ANKARA UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

COM 4062 PROJECT REPORT

AN INTRUSION DETECTION SYSTEM FOR NEW CYBER ATTACKS

Ayça Nur VANLI

17290128

Asst. Prof. Ömer Özgür TANRIÖVER

May, 2021

ABSTRACT

This report is written for the Ankara University Computer Engineering Department in order to inform about my studies for my undergraduate research project. The aim was developing a successful Machine Learning (ML) model for network intrusion detection system. The CICIDS2017 dataset was used in this research. For the model, Convolution Neural Network (CNN) and Bidirectional Long Short-term Memory (BILSTM) are selected to conduct a hybrid model.

TABLE OF CONTENTS

TABLE OF FIGURES AND TABLES

1. INTRODUCTION

Intrusion Detection System (IDS) is a commonly used device for cyber security solutions. The aim of this device is to monitor the network traffic and alert the administrator for any malicious actions in the traffic. We can classify three types of IDS based on their input data which are Network IDS (NIDS) that takes input from the network packets, Host-based IDS (HIDS) that takes input from the specific host's traffic, and Application IDS that takes input as high-risk applications traffic [1].

With the soaring expansion of the Internet over the years, every network has been facing a tremendous amount of traffic with an exponential growth rate. For this reason, it has become inevitable to abandon the traditional network IDS. This traditional approach to detecting an intrusion is called signature-based IDS where the IDS's database tries to match the actions with the already detected attacks. Hence, the industry started to consider a more intellectual way to deal with their big data and they started to improve the efficiency of their IDS. When we are considering big data like network traffic, ML algorithms can be a remedy o handle them. Therefore, machine learning algorithms can be designed to detect the bad intended traffic in a network. So, the other approach to detect an intrusion is called the anomaly-based approach where the IDS is capable to determine the obscure attacks based on abnormal behaviors.

2. LITERATURE REVIEW

When we search about the applicable dataset to use, one of the latest and diverse datasets is CICIDS2017. In the article called Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization [2] we can have a in depth look about how the dataset is formed. This dataset is created by the Canadian Institute for Cybersecurity in 2017. And the dataset satisfies the evolution framework for IDS’s.

First, they created two kind of network. One is attacker network and the other one is the victim network. In attacker network, they have the machines that will initiate the attack towards to the victim network. The attacker network consists of some Kali computers. On the other hand, they tried to resonate the victim to the legitimate network as much as possible. Hence, they put a Domain Controller, DNS server, Windows and Ubuntu servers, Windows, Mac and Ubuntu user computers, Fortinet

Firewall. All of the benign traffic is created using java agents and the protocols are SSH, HTTP/S, FTP and email protocols. The attack traffic consists of 7 different attack profiles. Brute Force, Heart Bleed Attack, Botnet, DoS (Denial of Service) Attack, DDoS (Distributed Denial of Services) Attack, Web Attack, and Infiltration Attack. After creating the flow of network, they extract 80 features by processing the .pcap files using CICFlowMeter. Finally, the CICIDS2017 dataset is completed. In the "An Efficient Feature Selection Based Bayesian and Rough set approach for intrusion detection" article [5], the study calculated the reality quality of the multiple IDS datasets using Fuzzy Logic System. They conclude that out the accuracy of the CICIDS2017 dataset to a legitimate network traffic is ~%75.

When we need to look at the CICIDS2017 dataset's flaws, "A detailed analysis of CICIDS2017 dataset for designing Intrusion Detection Systems – 2018" [3] can lead us to three major fatal error within the dataset. The first one is the size of the dataset. Since the dataset is relatively big, it increases the processing time and decreases the efficiency of the algorithms. The second one is incomplete and redundant of the data inside the dataset. For example, in the total of 3119345 instances, 288602 of them does not have a class. This makes harder to attain the assign accurate weights. The third of them is the high-class imbalance in the dataset. For instance, the benign class dominates with the %83.34 percentage meanwhile Heartbleed class has only %0.00039. This creates model to be a biased towards particular class.

In the article "Evaluating the CIC IDS-2017 Dataset Using Machine Learning Methods and Creating Multiple Predictive Models in the Statistical Computing Language R" [4] we get to overcome the obstacle of the dataset. Such as, they assigned the maximum value of the column wherever "Infinite" occurs, and they assigned the average value of the column wherever "NaN" (Not a Number) occurs. Moreover, they omit the instances that lack data. In the article "Building an efficient intrusion detection system based on feature selection and ensemble classifier" [6], we can see that using building ensemble is classifiers can be an approach such as ensemble classifier that consist of C4.5 Random Forest, ForestPA. Also, in the article they use hybrid feature selection that includes Correlation Based Filter Selection Algorithm and Bat Algorithm.

3. METHODS & TOOLS

In the preprocessing stage of the dataset, First I assigned the appropriate data type of each value such as int8, float64 or categorical. Then I remove the “Infinity”, “NaN” and empty values. Since the attacks occurs chronically not random, I removed the any column related to the date and time since the model would simply learn which attack occurs in which time and would make a shallow decision. Then I scaled numerical attributes then I standardize them so that the mean value is zero and the standard deviation is one. Since ML algorithms can understand the categorical data raw, it must be transformed into a numerical form. Hence, I used One Hot Encoder for the categorical values.

After these preprocessing, my first trial was running the model without any feature selection to see the outcome. I decided to use CNN for my model. CNN is an Artificial Neural Network that is inspired by the human brain’s ability to process visuals. CNN is multi-layered and feed-forwarded. CNN model has three layers which are sequentially convolutional layer, pooling layer, and fully connected layer.

We can see the summary of the model in the Figure 1. For the trial #1, I used batch size as 256 and I set the number of epochs to 100.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 70, 1)]	0
conv1d (Conv1D)	(None, 70, 32)	128
max_pooling1d (MaxPooling1D)	(None, 35, 32)	0
dropout (Dropout)	(None, 35, 32)	0
batch_normalization (BatchNo	(None, 35, 32)	128
conv1d_1 (Conv1D)	(None, 35, 64)	6208
max_pooling1d_1 (MaxPooling1	(None, 17, 64)	0
dropout_1 (Dropout)	(None, 17, 64)	0
batch_normalization_1 (Batch	(None, 17, 64)	256
flatten (Flatten)	(None, 1088)	0
dense (Dense)	(None, 32)	34848
dense_1 (Dense)	(None, 15)	495

Total params: 42,063
Trainable params: 41,871
Non-trainable params: 192

Figure 1: CNN Model Summary

Then I run the Random Forest Classifier and see the rank of the importance of the features. After that, I applied the feature selection. I used Recursive Feature Elimination. I diminished the number of features to 40.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_7 (InputLayer)         [(None, 40, 1)]           0
_________________________________________________________________
conv1d_13 (Conv1D)           (None, 40, 32)            128
_________________________________________________________________
max_pooling1d_11 (MaxPooling (None, 20, 32)            0
_________________________________________________________________
dropout_11 (Dropout)         (None, 20, 32)            0
_________________________________________________________________
batch_normalization_11 (Batc (None, 20, 32)            128
_________________________________________________________________
conv1d_14 (Conv1D)           (None, 20, 64)            6208
_________________________________________________________________
max_pooling1d_12 (MaxPooling (None, 10, 64)            0
_________________________________________________________________
dropout_12 (Dropout)         (None, 10, 64)            0
_________________________________________________________________
batch_normalization_12 (Batc (None, 10, 64)            256
_________________________________________________________________
flatten_5 (Flatten)          (None, 640)               0
_________________________________________________________________
dense_10 (Dense)             (None, 32)                20512
_________________________________________________________________
dense_11 (Dense)             (None, 40)                1320
=================================================================
Total params: 28,552
Trainable params: 28,360
Non-trainable params: 192
_________________________________________________________________
```

Figure 2: CNN Model with Feature Selection Summary

Then for the trial #3, I decided to double the batch size from 256 to 512. For the trial #4, I decided to implement an CNN-BILSTM model. For the trial #5, I tried to run the model with only 10 features. For the last trial, I tried to improve the 10 features by adding extra convolutional layers after each one and tuning hyperparameters. The final model can be seen in Figure 3. All of the trials' summary can be seen in the Table 1.

Trial #	Model	Feature Selection	Number of Features	Batch Size
1	CNN	No	70	256
2	CNN	Yes	40	256
3	CNN	Yes	40	512
4	CNN-BILSTM	Yes	40	512
5	CNN-BILSTM	Yes	10	512
6	CNN-BILSTM	Yes	10	512

Table 1: Trial's Summary

Layer (type)	Output Shape	Param #
input_8 (InputLayer)	[(None, 10, 1)]	0
conv1d_17 (Conv1D)	(None, 10, 32)	128
conv1d_18 (Conv1D)	(None, 10, 16)	1552
max_pooling1d_13 (MaxPooling	(None, 5, 16)	0
dropout_13 (Dropout)	(None, 5, 16)	0
bidirectional_7 (Bidirection	(None, 5, 140)	48720
batch_normalization_13 (Batc	(None, 5, 140)	560
conv1d_19 (Conv1D)	(None, 5, 64)	26944
conv1d_20 (Conv1D)	(None, 5, 32)	6176
max_pooling1d_14 (MaxPooling	(None, 2, 32)	0
dropout_14 (Dropout)	(None, 2, 32)	0
batch_normalization_14 (Batc	(None, 2, 32)	128
flatten_6 (Flatten)	(None, 64)	0
dense_12 (Dense)	(None, 32)	2080
dense_13 (Dense)	(None, 10)	330

Total params: 86,618
Trainable params: 86,274
Non-trainable params: 344

Figure 3: CNN-BILSTM Model

4. RESULTS & DISCUSSION

The first result I obtained was about the usage of CNN model without any feature selection which would be 70. Even if I got 80.35% accuracy for Trial #1, the loss function can be seen in the Figure 5 tell us the instability of our model.

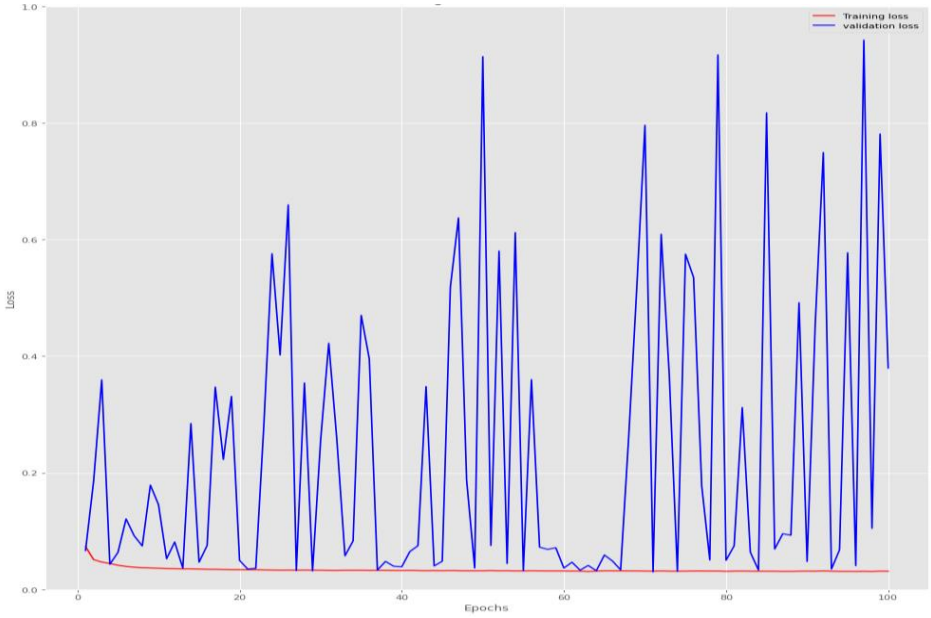

Figure 4:Trial #1's Loss Graph

When I run the Random Forest Classifier for to examine the importance of the 70 features, I saw a divergent number of features that can have less impact in my model. It can be seen in a descendent sorted manner in Figure 4. This encourages to do feature selection in the model. After the results, I diminished the number of features to half of the original number of features which led to 40. After trying with 40 features on trial #2, I got accuracy of 88,10%. The loss function that can be seen in the Figure 6 was better than the trial #1's. This showed me the improvement in my model.

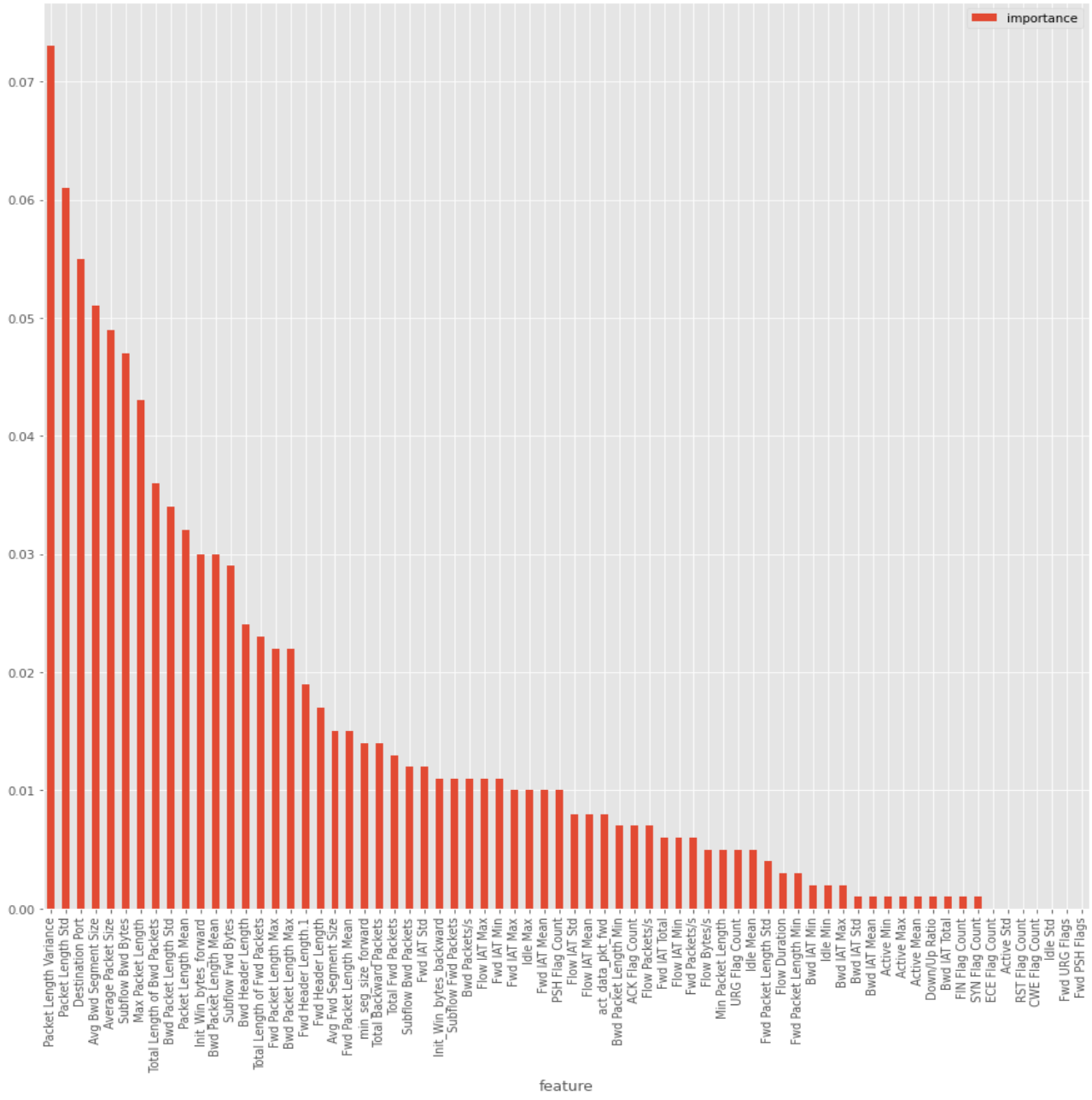

Figure 5: Importance of Features

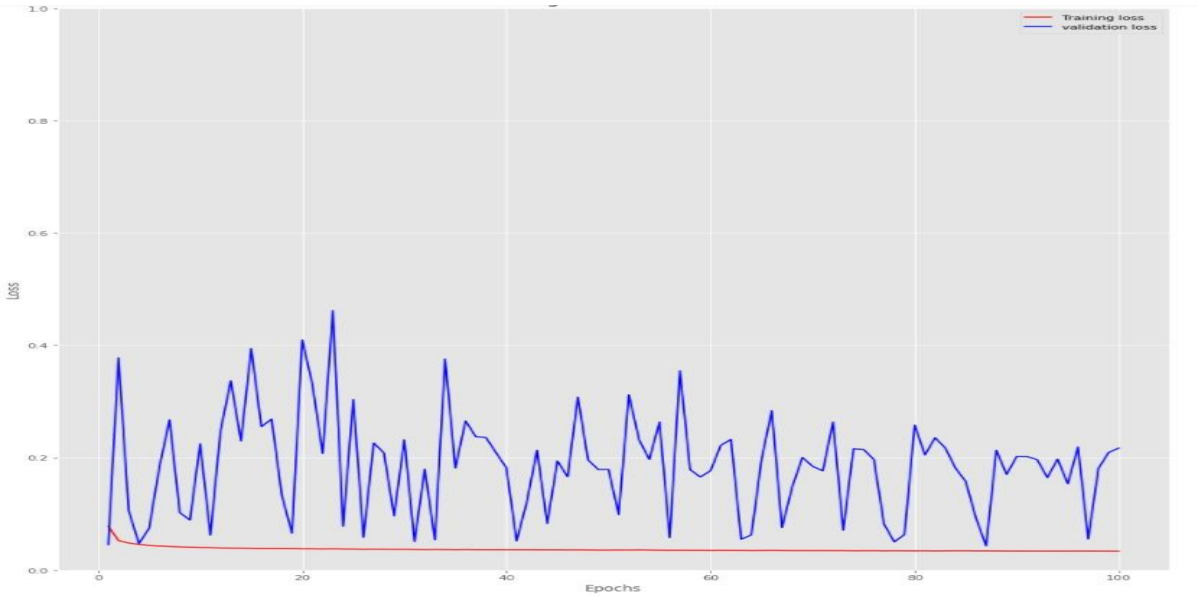

Figure 6: Trial #2's Loss Graph

I decided to make the batch size to 512 from 256 because of the amount of time my model takes for each trial. I run the model with the batch size 512 for trial #3 and I got the accuracy of 93.74%. In contrast to increase in accuracy, the loss function that can be seen in the Figure 7 was less stable than the trial #2's. When we look at the accuracy function of the Trial #3's that can be seen in Figure 8, we can see the harsh zig-zag in the model's accuracy.

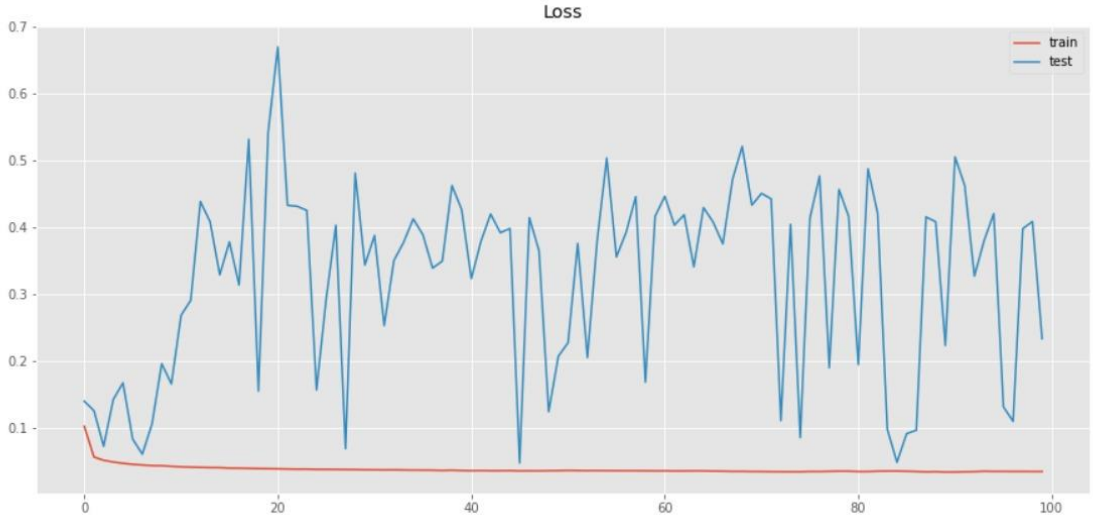

Figure 7: Trial #3's Loss Graph

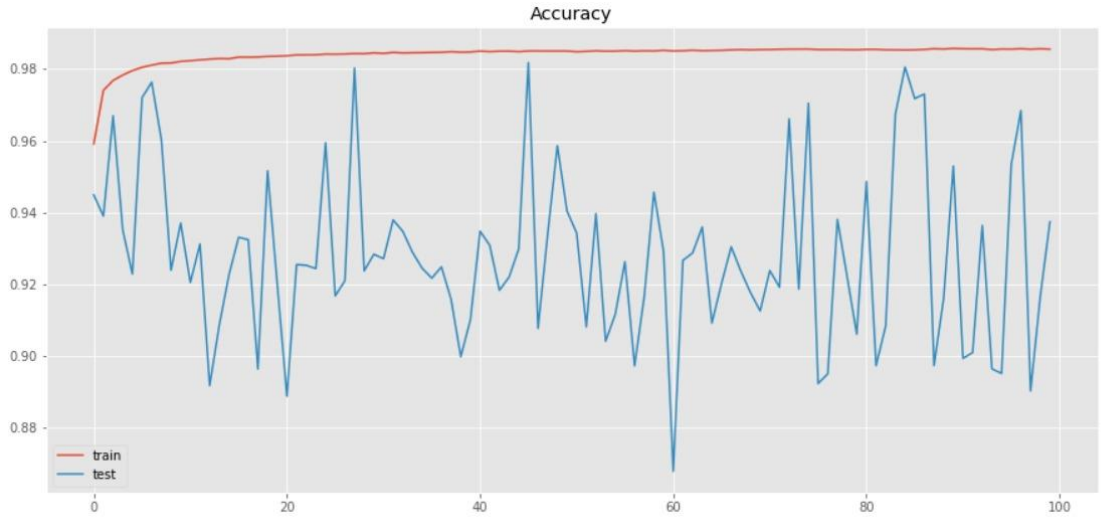

Figure 8: Trial #3's Accuracy Graph

Because of the instable form of the CNN model, I decided to implement BILSTM layers in our CNN model. After implementing the the hybrid system, I obtained a lot more stable form of my model. The accuracy was 97,53% which was the peak in my experiment so far. The trial #4's loss graph can be seen in the Figure 9 and the accuracy graph can be seen in the Figure 10.

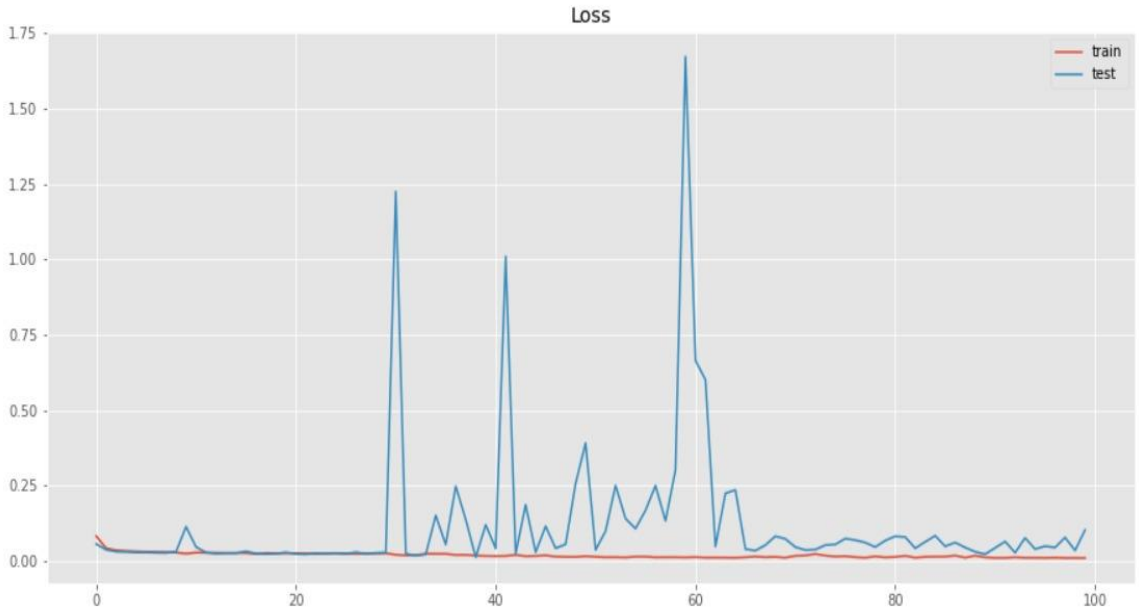

Figure 9: Trial #4's Loss Graph

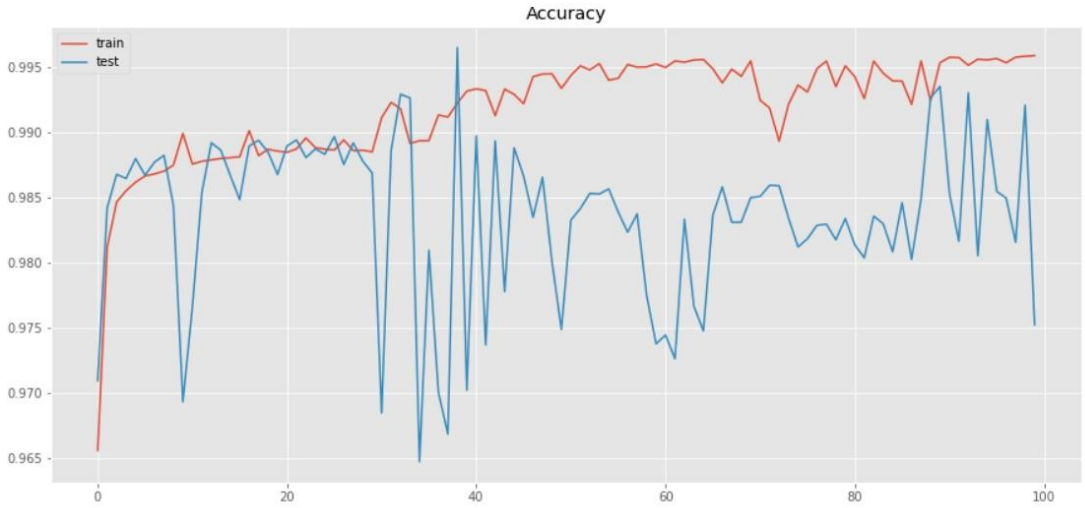

Figure 10: Trial #4's Accuracy Graph

I decided to experiment with the number of features so I decreased the number of features from 40 to its quarter amount, 10. When I run the trial #5, I got much worse results than the previous trial. The loss graph that can be seen in Figure 11 was the worst loss graph in my experiment. Also, accuracy diminished from 97,53% to 89,13%. The accuracy graph can be seen in the Figure 12.

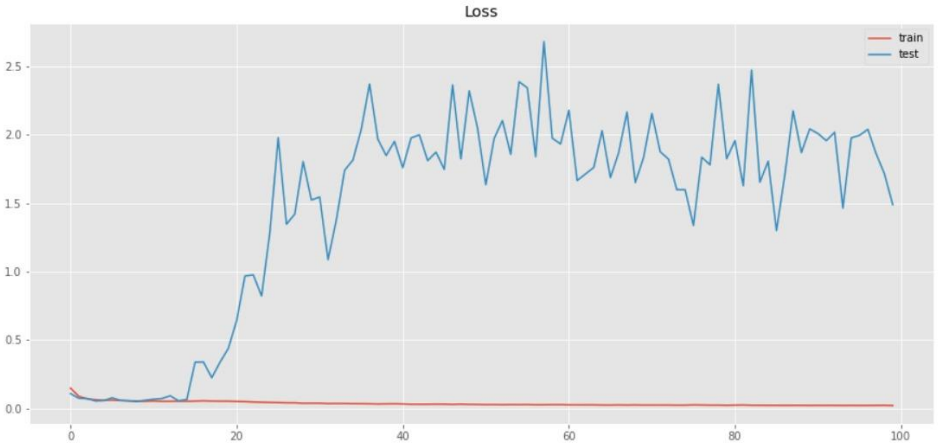

Figure 11: Trial #5's Loss Graph

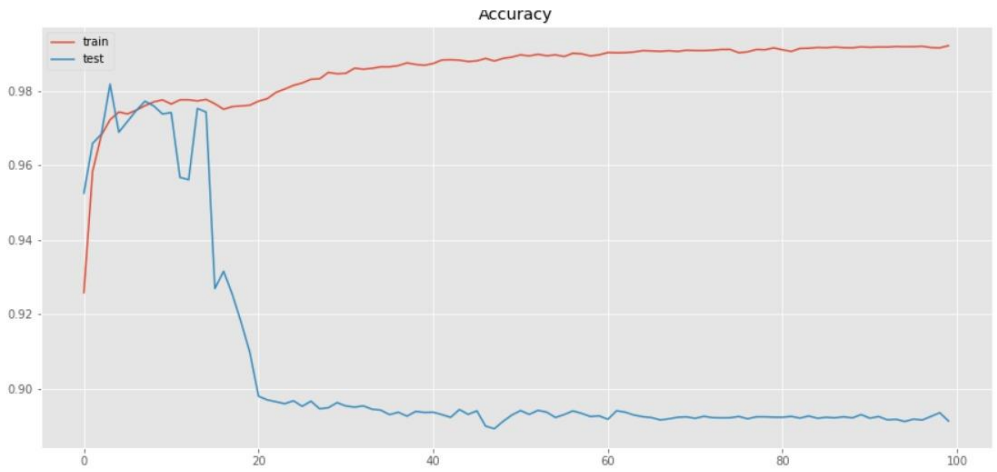

Figure 12: Trial #5's Accuracy Graph

Trial #5's behaviour made me realize that I can optimize my model's hyperparameters and structure of layers. I doubled the number of convolutional layers by adding one more convolutional layer after each one. Also, I tuned the hyperparameters of the model. With these changes, trial #6's performance sharpened increasingly. The loss graph, can be seen in the Figure 13, was more synchronized with the reality. I reached the minimum loss in the trial #6 with 0.07. The accuracy graph was a lot better too. The improvement from the previous trial were impressive with increase from 89,13% to 96,39%. The accuracy graph can be seen in the Figure 14.

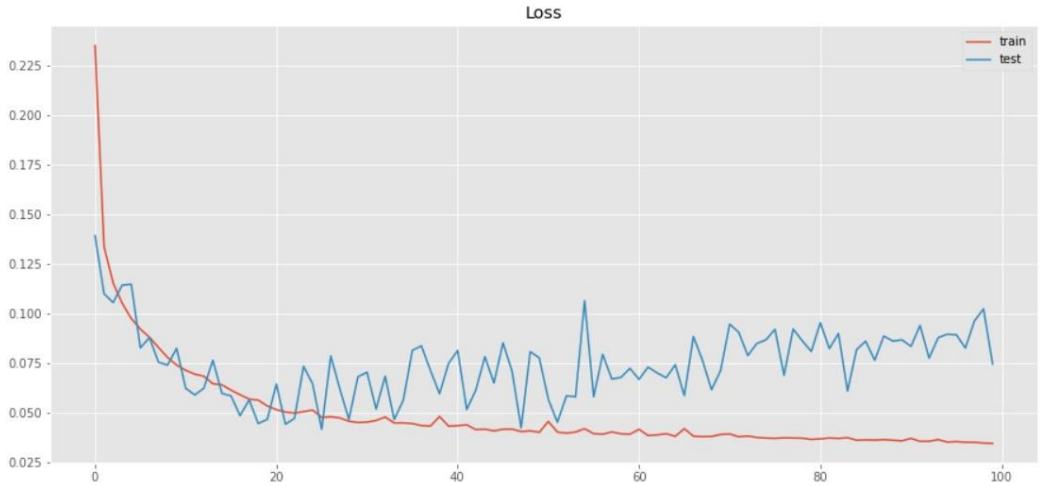

Figure 13: Trial #6's Loss Graph

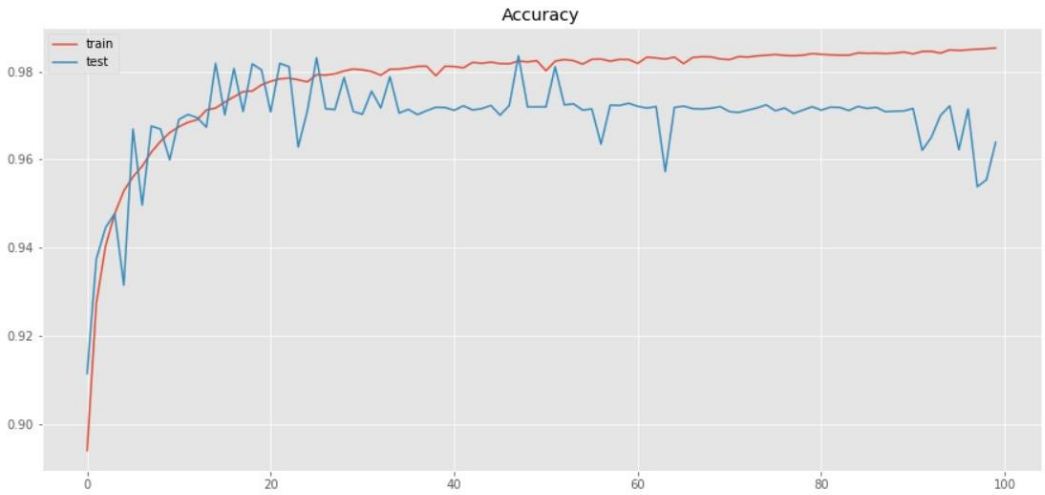

Figure 14: Trial #6's Accuracy Graph

The summary of the all of the trials' results are shown in the Table 2. Additionally, the improvement of the accuracy and the loss is graphed in the Figure 15.

Trial #	Model	Feature Selection	Number of Features	Loss	Accuracy
1	CNN	No	70	0.37	0.8035
2	CNN	Yes	40	0.21	0.8810
3	CNN	Yes	40	0.23	0.9374
4	CNN-BILSTM	Yes	40	0.10	0.9753
5	CNN-BILSTM	Yes	10	1.49	0.8913
6	CNN-BILSTM	Yes	10	0.07	0.9639

Table 2: Trial's Results

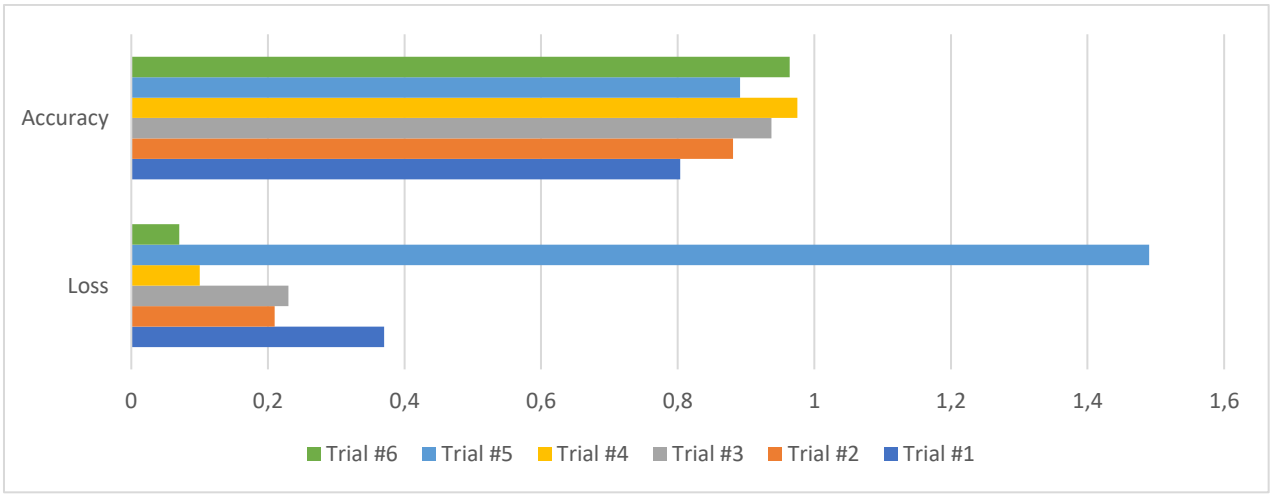

Figure 15: Loss and Accuracy Graph

5. CONCLUSION

As I examined the literature about CICIDS2017 dataset, I found the pros and cons of the data it contains such as 3 major details that must be cared for preprocessing. These are the size of the dataset, high-class imbalance, the amount of lack information and redundancy within the instances. With these cautions in my head, I begin with working with it using a sequential model called CNN.

At first, I run the model without any feature selection. The model's accuracy and loss were bad. So, I use feature selection and reduce the number of features to half amount. In this second trial, the model had begun to behave more desirably. Since the amount of time the model was taking for each trial was relatively high, I doubled the batch size. With the new batch size, the model resulted in better accuracy in this third trial. For improving the architecture of the model, I correlated BILSTM layers. In this fourth trial, the loss and the accuracy gave me the best results so far. For the experimental point, I cut the number of features to its quarter to see the model's behaviour. In this fifth trial, model predicted so poorly that it was a dramatic change. To better the implementation of my model, I increase the number of convolutional layers to double and tuned the hyperparameters. In this sixth trial, model's loss and accuracy graphs were near to the expected results and gave me the lowest loss so far which is 0,07 and the accuracy was 96,39%. With the success of my model, I conclude my research successfully.

BIBLIOGRAPHY

[1] A. A. N. J. M. S. SH Kok, "A Review of Intrusion Detection System using Machine Learning Approach," *International Journal of Engineering Research and Technology,* vol. 12, no. 1, pp. 8-15, 2019.

[2] A. H. L. a. A. A. G. Iman Sharafaldin, "Toward Generating a New Intrusion Detection Dataset and Intrusion," in *ICISSP 2018 - 4th International Conference on Information Systems Security and Privacy*, 2018.

[3] S. T. a. K. D. M. Prasad, "An efficient feature selection based Bayesian and Rough set approach," *Applied Soft Computing Journal,* 2020.

[4] S. B. Ranjit Panigrahi, "A detailed analysis of CICIDS2017 dataset for designing Intrusion Detection Systems," *International Journal of Engineering & Technology,* vol. 7, no. 3.24, pp. 479-482, 2018.

[5] M. A. Zachariah Pelletier, "ISSN (Online): 2455-9024," *International Research Journal of Advanced Engineering and Science,* vol. 5, no. 2, pp. 187-191, 2020.

[6] G. C. a. S. J. Y. Zhou, "Building an efficient intrusion detection system based on feature selection and ensemble classifier," *Computer Networks,* 2020.