CHAPTER IV
EXPERIMENT, RESULT, AND ANALYSIS

In this chapter, this undergraduate thesis presents the result and analysis of the models that were made from the experiment conducted. After a thorough iterations of experiments on the implementation of Machine Learning (ML) on Intrusion Detection System (IDS), it is mandatory to test the model as an evaluation on how the full-scale system works and to analyse whether the ideal model has been achieved in respect to performance metric scores such as accuracy and precision.

In addition, another metric will be added to complement the overall analysis which is time value. This is because to achieve and implement the most desired and efficient IDS model, the length of process time is inevitable for judging the quality of the IDS model. However, the time metric showed on this chapter does not represent the actual IDS performance but merely just an estimation on how the IDS model would perform in real environment. This is due to the fact, except batch size parameter tweaking, the time metric presented forthcoming only covers the testing process of the model without implementing it on real network traffics. The time metric on batch size parameter tweaking however, covers both training and testing process since batch size parameter is heavily correlated with training performance.

Furthermore, the final model result through fine tuning or parameter tweaking will be analysed to obtain conclusions on advantages, weaknesses, and limitations of the model to the desired results.

The overall result of this experiment in this chapter including its analysis is divided into several points in sub-chapters. Those points are as follows:

1. The scenario on how this while experiment is conducted.
2. Final result of parameter tweaking of the implemented ML approach.
3. Model evaluation with k-fold cross-validation technique.
4. The overall analysis of the whole experiment.

4.1 Experiment Scenario

In general, the goal of this experiment is to finding the ideal conditions of the implemented ML approach by tweaking the designated parameters. Those parameters are convolution layer's filter value, dropout layer's value, number of the neural

network's dense layer nodes, and model's training batch size. Furthermore, after the model with the most optimal result obtained by parameters tweaking process, the model will be evaluated with cross-validation technique. In this case, k-fold cross-validation technique is chosen with the value k of 3, and 2. The range values of each parameter that will be fine-tuned in this experiment is represented on Table 4.1 and are described as follows:

1. The range values of the convolution layer's filter values are 8 and 16. The reason the numbers chosen are less than 32 which is the conservative value is because this experiment is dealing with relatively small resolution or dimension size of data input. Thus, smaller values will be more effective. For simplicity, the usage of this parameter will be abbreviated with letter 'C' on further figures and tables.

2. The range values of the dropout layer are 0.3, 0.5, and 0.7. In this case, 0.3 and 0.7 are also chosen to show which leaning bias is better between smaller and bigger dropout value. For simplicity, the usage of this parameter will be abbreviated with 'DO' on further figures and tables.

3. The range values of the neural network's dense layer are 16, 32, 64, and 128. Similar with the value of convolution, this experiment would not need a big value of neural network nodes. Since the input data of the ML model is relatively small and bigger nodes is also mean a waste of resource and more computational time which leads to inefficiency. For simplicity, the usage of this parameter will be abbreviated with 'De' on further figures and tables.

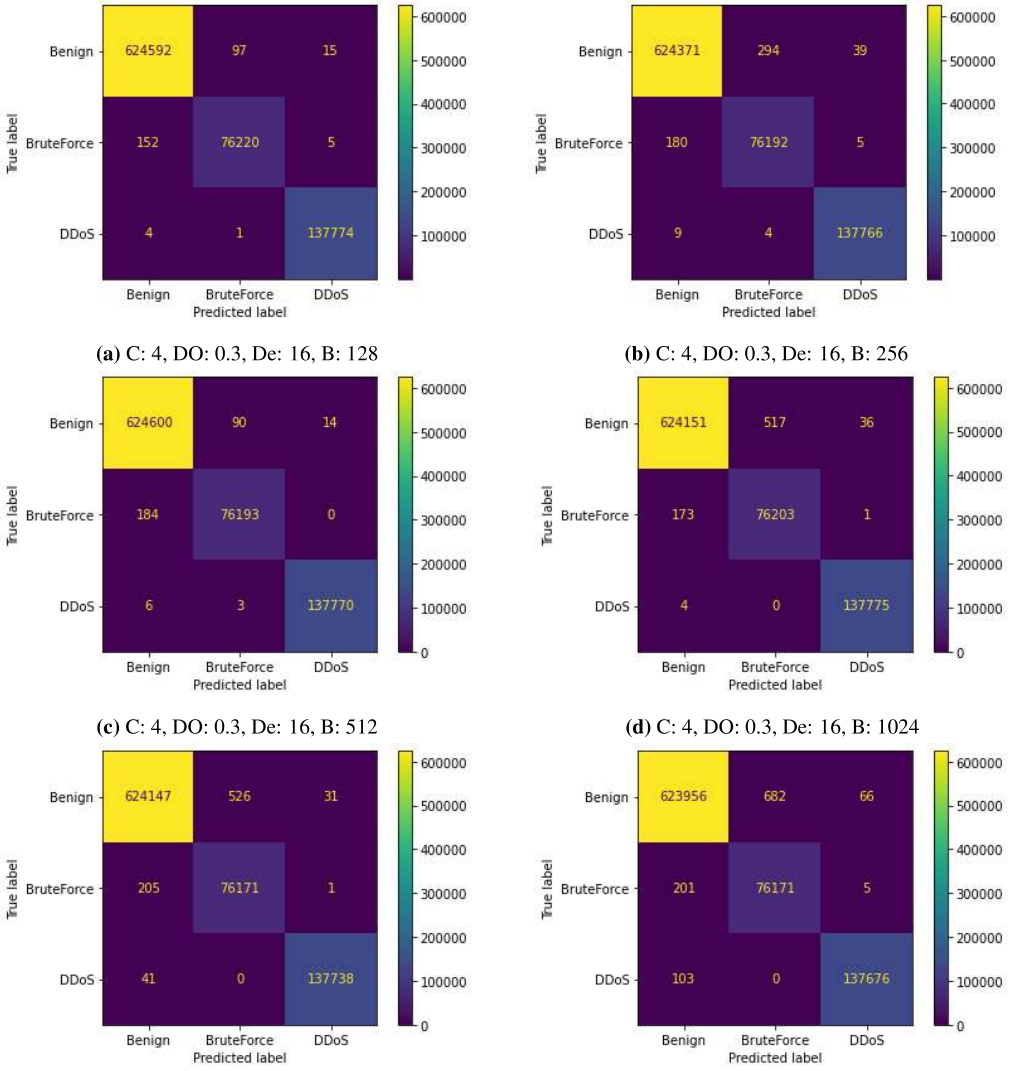
4. The range values of the training's batch size on the ML model are 128, 256, 512, 1024, and 2048. The reason 2048 is chosen for the highest value is because the bigger the size of the batch, the more inaccurate the model will be. For simplicity, the usage of this parameter will be abbreviated with letter 'B' on further figures and tables.

Table 4.1 Parameter that will be tweaked on this experiment.

Parameter name	**Abbreviation**	**Value**
Convolution filter	C	8, and 16.
Dropout layer	DO	0.3, 0.5, and 0.7
Dense layer	De	16, 32, 64, and 128.
Batch size	B	128, 256, 512, 1024, and 2048

4.2 Training and Testing Result

The process to find the optimal IDS model is done by manually train and test the dataset with reiteration for each parameter values. In this case, the experiment started with the lowest value for each parameter and reiterated by changing each of values in respect to the model performance. The model performance observed are performance metrics and time to complete the process. The parameter tweaking stage will be conducted with simple training and testing scheme. The author choses 8:2 data ratio, which means 20% of the dataset are randomly chosed as testing set. The reason is because the model will be evaluated with higher testing ratio by cross-validation technique later on the evaluation stage. Fig. 4.1 shows this first stage's confusion matrices.



(a) C: 4, DO: 0.3, De: 16, B: 128

(b) C: 4, DO: 0.3, De: 16, B: 256

(c) C: 4, DO: 0.3, De: 16, B: 512

(d) C: 4, DO: 0.3, De: 16, B: 1024

(e) C: 4, DO: 0.3, De: 16, B: 2048

(f) C: 4, DO: 0.3, De: 16, B: 4096

Figure 4.1 Plots of first stage's confusion matrices.

As one can observed above, the confusion matrices show a quite similar raw result among different batch size values with staggering high-rate classification success. However, to get more information out of the confusion matrices, performance metrics need to be calculated. As the concept and the technicalities of how to obtain performance metric of ML model has been explained and presented in Chapter 3, the following Table 4.2 shows the result of batch size values tweaking with each and other values in their smallest designated sizes.

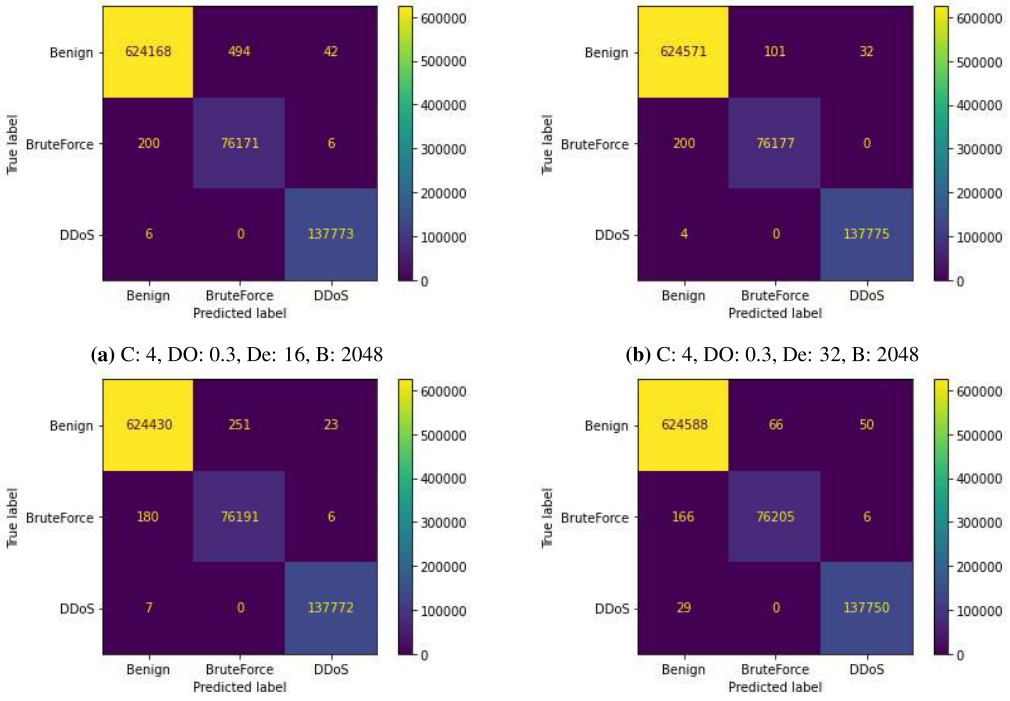
Table 4.2 Performance metrics of batch size value tweaking.

	Acc. (%)	**Prec. (%)**	**Rec. (%)**	**F1 (%)**	**Loss (%)**	**Train & testing**
C = 4 DO = 0.3 De = 16 **B = 128**	99.9673	99.9673	99.9673	99.9673	0.2	10 mins 30 secs
C = 4 DO = 0.3 De = 16 **B = 256**	99.9367	99.9367	99.9367	99.9367	0.51	5 mins 13 secs
C = 4 DO = 0.3 De = 16 **B = 512**	99.9646	99.9646	99.9646	99.9646	0.24	3 mins 17 secs
C = 4 DO = 0.3 De = 16 **B = 1024**	99.9129	99.9129	99.9129	99.9129	0.4	2 mins 18 secs
C = 4 DO = 0.3 De = 16 **B = 2048**	99.9042	99.9042	99.9042	99.9042	0.52	1 min 37 secs
C = 4 DO = 0.3 De = 16 **B = 4096**	99.8740	99.8740	99.8740	99.8740	1.03	1 min 18 secs

From Table 4.2 above, the performance metrics shows that the initial proposed ML model has already attained stunning results with a steady above 99% on scores in each and every metric. However, the only significant difference from this stage

process is the time to process each model. A smaller batch size of training data tends to slow the building process of the IDS model. This is due to a small batch size of training data inputted continuously to the ML model also means the bigger the iteration number takes thus will lead to longer time needed for the model to finish all the training process. Furthermore, the batch size value of 2048 has the best time performance even though statistically, the smaller the batch size value, the better the accuracy. The author has also tried 4096 as the batch size value, but the process time has become stagnant with only 19 seconds time improvements and less metric scores compared to the value of 2048.

As it is better to choose smaller batch size value in respect to both time and performance, the author choses 2048 as the parameter value. The drawbacks of this decision is expected to be improved on the next three parameter tweaking stages. The next parameter tweaking stage is solely purposed to tweak the neural network's dense layer parameter in respect to the performance of batch size value of 2048. Fig. 4.2 shows the second stage's confusion matrices.



(a) C: 4, DO: 0.3, De: 16, B: 2048

(b) C: 4, DO: 0.3, De: 32, B: 2048

(c) C: 4, DO: 0.3, De: 64, B: 2048

(d) C: 4, DO: 0.3, De: 128, B: 2048

Figure 4.2 Plots of second stage's confusion matrices.

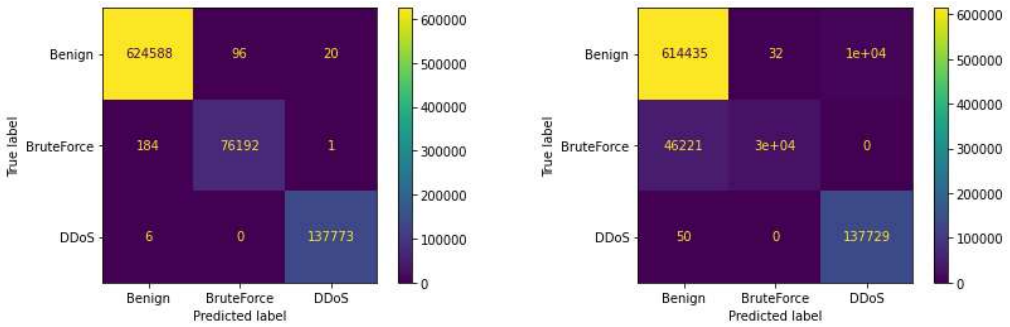
Similar with first stage's performance matrices in Fig. 4.1, performance matrices of the second stage in Fig. 4.2 also shows a predominantly high-scores in each of label classification. Furthermore, the performance metrics to evaluate this stage's parameter tweaking is shown on Table 4.3

Table 4.3 Performance metrics of dense layer value tweaking.

	Acc. (%)	Prec. (%)	Rec. (%)	F1 (%)	Loss (%)	Testing
C = 4 DO = 0.3 **De = 32** B = 2048	99.9598	99.9598	99.9598	99.9598	0.48	1 min 12 secs
C = 4 DO = 0.3 **De = 64** B = 2048	99.9443	99.9443	99.9443	99.9443	0.88	1 min 12 secs
C = 4 DO = 0.3 **De = 128** B = 2048	99.9622	99.9622	99.9622	99.9622	0.4	1 min 12 secs

The performance metrics on Table 4.3 shows a pretty similar behaviour with the first stage's parameter tweaking due to each and every scores are consistently above 99%. As one can observe, the value of 32 and 64 could reach 99.95% and 99.94% accuracy respectively. However, the dense layer value of 128 has slightly better accuracy and time performance compared to other two values with only four seconds time difference. The author also has decided to not continue with a dense layer value bigger than 128 as this could lead to an overfitting condition. Ultimately, the author chooses 128 for the value of this parameter tweaking stage and moves forward toward the next stage which is dropout layer parameter tweaking.

The dropout layer parameter tweaking stage is conducted in respect to the batch size value of 2048 and dense layer value of 128. Fig. 4.3 shows the confusion matrices of this stage's parameter tweaking.



(a) C: 4, DO: 0.5, De: 128, B: 2048

(b) C: 4, DO: 0.7, De: 128, B: 2048

Figure 4.3 Plots of third stage's confusion matrices.

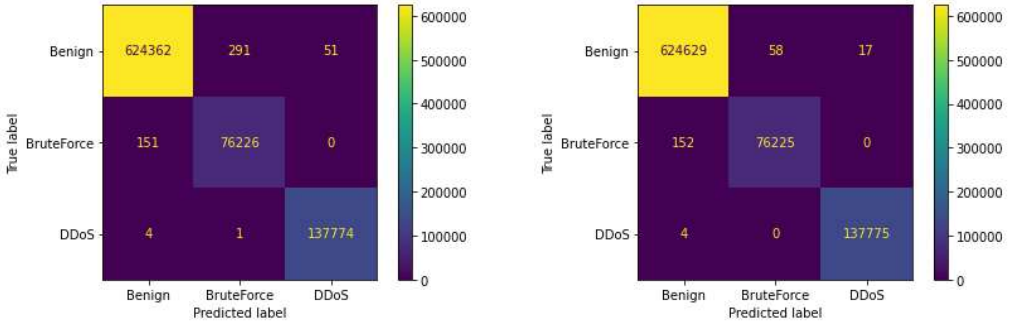
In Fig. 4.3, deviation could be observed as the value of 0.5 and 0.7 show contrast results. On dropout value of 0.7, the model has a very bad performance on detecting BruteForce attack with over than 46000 BruteForce attack is predicted as Benign. In this stage, clearly the dropout value of 0.5 is outperforming the dropout value of 0.7. But in order to analyse these models deeper, the following performance metrics are shown on Table 4.4

Table 4.4 Performance metrics of dropout value tweaking.

	Acc. (%)	**Prec. (%)**	**Rec. (%)**	**F1 (%)**	**Loss (%)**	**Testing**
C = 4 **DO = 0.3** De = 128 B = 2048	99.9622	99.9622	99.9622	99.9622	0.4	1 min 12 secs
C = 4 **DO = 0.5** De = 128 B = 2048	99.9634	99.9634	99.9634	99.9634	0.91	1 min 11 secs
C = 4 **DO = 0.7** De = 128 B = 2048	93.2599	93.2599	93.2599	93.2599	31	1 min 22 secs

From Table 4.4, it is clear that the dropout value of 0.7 has a decreasing performance with 93% accuracy compared to all IDS model from the beginning of this experiment. The loss metric on dropout value of 0.7 also shows an outlier result of 31% which is far less than the acceptable IDS performance metrics in general. On this stage, the author has drawn conclusions that the bigger the dropout value added on the model, the less accurate the IDS will be while the performance time doesn't change much. Furthermore, the author has suggested the best dropout value for this IDS model lies between the value of 0.3 and 0.5. For simplicity, the next convolution stage ahead will use the dropout value of 0.3.

The next and the final parameter tweaking stage deals with finding the best convolution filter value in respect to the batch size value of 2048, the dense layer value of 128, and the dropout value of 0.3. The confusion matrices of the convolution filter value parameter tweaking are shown on Fig. 4.4 below.



(a) C: 8, DO: 0.3, De: 128, B: 2048

(b) C: 16, DO: 0.3, De: 128, B: 2048

Figure 4.4 Plots of fourth stage's confusion matrices.

For the last plot of this experiment stages, one can observe that the convolution tweaking for both values has very similar result. The convolution value of 16 also has slightly better performance for classifying benign traffic compared to value of 8. To analyse this stage deeper, the performance metrics of this stage's parameter tweaking is shown on Table 4.5

Table 4.5 Performance metrics of convolution value tweaking.

	Acc. (%)	**Prec. (%)**	**Rec. (%)**	**F1 (%)**	**Loss (%)**	**Testing**
C = 8 DO = 0.3 De = 128 B = 2048	99.9406	99.9406	99.9406	99.9406	0.22	1 min 16 secs
C = 16 DO = 0.3 De = 128 B = 2048	99.9725	99.9725	99.9725	99.9725	0.16	58.1 secs

From Table 4.5 above, the convolution filter value of 16 has shown a slightly improvements compared to the value of 8. This little improvements also come with a stable performance time. Furthermore, the convolution filter value of 16 also has shown the lowest loss and testing time metric score compared to all iterations conducted from the beginning of this experiment.

It is also noted that with total of 838,623 traffic packets, both convolution filter value of 8 and 16 has a testing performance of 76 and 58.1 seconds respectively. This proves that the model could test a packet for as fast as 0.069 ms which is better than the International Telecommunication Union (ITU) standard of end-to-end delay time with no more than 150 ms / packet delay. This final model further will be evaluated by k-fold cross-validation technique in the next subchapter.

4.3 Evaluation with k-fold Cross-validation

After the ideal model parameters has been achieved, the result of this experiment is still need to be evaluated to make sure that the IDS is able to predict new data traffics that was outside of the training dataset. In this case, the use of cross-validation technique is important as it will decrease the chance of selection bias and exhibit an insight on how the IDS will behave towards independent or unknown instance. This evaluation process will be conducted by a randomized and stratified k-fold cross-validation as detailed and described on Chapter 3. Randomized and stratified however, are just additional k-fold characteristic added to the k-fold process to preserve the same class ratio throughout all of the iterations to the ratio of the whole original dataset while maintaining its randomness.

This evaluation process will utilize the k value of 3 and 2. This means the cross-validation process will utilize 33% and 50% of its whole dataset as testing dataset respectively. The reason smaller k value is chosen is because the lower the k value means that the IDS model is trained on a limited training dataset and tested on a bigger testing dataset thus will lead to a high error prediction on average. This was expected as the model has already shown staggering result with nearly perfect classifications on only 20% testing dataset that was conducted and presented in the beginning of this chapter.

The overall result of this evaluation process is divided into the following two different sub-chapters, namely Sub-chapter 4.3.1. for the k value of 3 and Sub-chapter 4.3.2. for the k value of 2.

4.3.1 Evaluation with k value of 3

The evaluation result with k value of 3 is shown by confusion matrices on Fig. 4.5. From this confusion matrices one can observe this model still holds a robust and solid score across all labels of benign, BruteForce, and DDoS classifications with 33% of its whole dataset as testing dataset. This is shown by the high numbers for both true negatives and true positives scores presented on the plot below. From this confusion matrices, the analysed scores are presented on Table 4.6

From Table 4.6 the evaluation process shows that this IDS model also has a similar and consistent performance compared to the model in Sub-chapter 4.2. stages. With more than 99% on all metric scores and identical losses, the proposed IDS model has passed the evaluation process with the value k of 3 and eligible to proceed for the evaluation with k value of 2 in the next Sub-chapter.

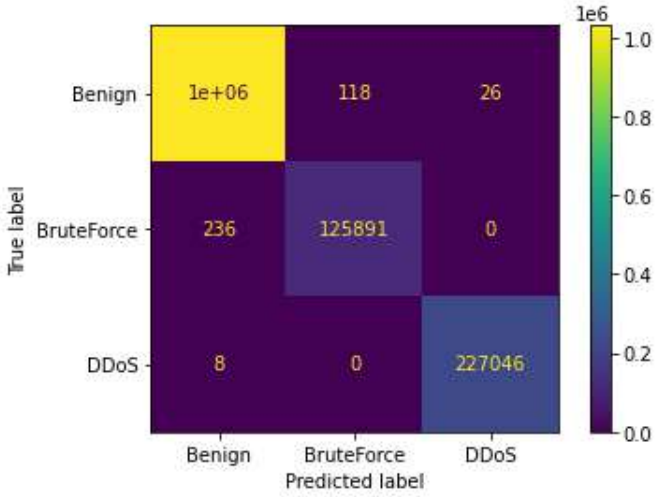
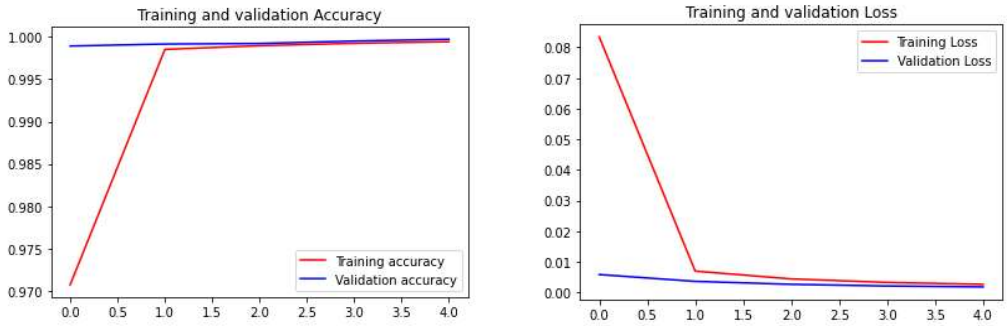


Figure 4.5 Confusion matrices for evaluation with k value of 3.

Table 4.6 Performance metrics of evaluation with k value of 3.

	Accuracy(%)	**Precision(%)**	**Recall(%)**	**F1(%)**	**Loss (%)**
C = 16 DO = 0.3 De = 128 B = 2048 k **= 3**	99.9720	99.9720	99.9720	99.9720	0.18

Furthermore, Fig. 4.6 also shows the visual graphics of the accuracies and losses of the model with 33% of its dataset as testing set. This plot clearly visualize that this model was already achieving its maximum performance in the second epoch or repetition and just became stagnantly high until the process finished. This proves that this model is not just successful in terms of final performance scores, but also has a completely steady and stable execution throughout the process.



(a) Accuracies

(b) Losses

Figure 4.6 Plots of the evaluation accuracies and losses with k value of 3.

4.3.2 Evaluation with *k* value of 2

Fig. 4.7 shows the confusion matrices for the evaluation process with *k* value of 2 which means 50% of the whole dataset acts as testing set. Likewise, the confusion metrices are still showing the robustness of the proposed IDS model. This is proved by a relatively similar and high scores for all of classifications of the labels.

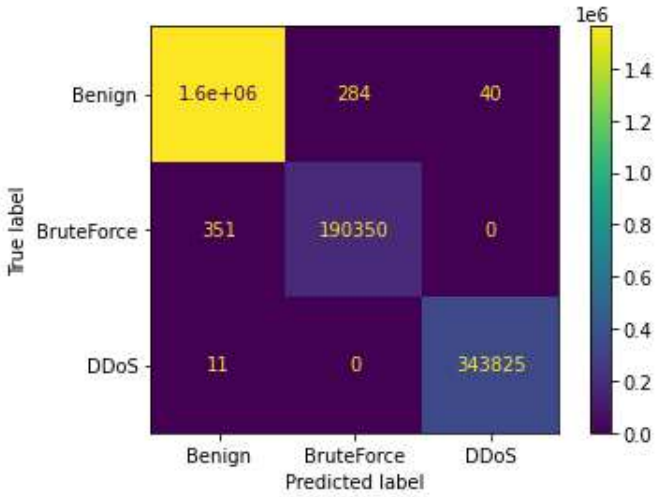


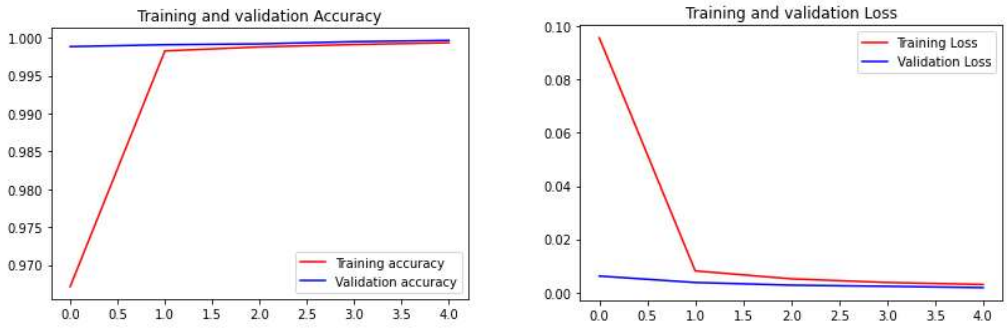
Figure 4.7 Confusion matrices for evaluation with k value of 2.

From Table 4.7, the author has concluded that this experiment has generate the most optimum and ideal IDS model for the proposed approach. Since 50% of its dataset has been used for testing set, performance metric scores still showing a result that is undeniably high and similar from the beginning.

Table 4.7 Performance metrics of evaluation with k value of 2.

	Accuracy(%)	**Precision(%)**	**Recall(%)**	**F1(%)**	**Loss(%)**
C = 16 DO = 0.3 De = 128 B = 2048 ***k* = 2**	99.9673	99.9673	99.9673	99.9673	0.18

Finally, Fig. 4.8 shows the visual graphics of the accuracies and losses of the model with 50% of its dataset as testing set. This plot shows that the model still performs a relatively steady and stable accuracies and losses throughout both the training and testing process, just like the precedent result shown in Sub-chapter 4.3.1.



(a) Accuracies

(b) Losses

Figure 4.8 Plots of the evaluation accuracies and losses with k value of 2.

4.4 Experiment Analysis

From the set of stages of experiment above, the author has gathered some final analysis on the result of this undergraduate thesis. Those analysis has been summarized on the following points:

1. Among all of the parameter used in this experiment, the value of batch size plays a very important role on determining the training and testing time which correlates heavily on efficiency for both computational and human resources.

2. Although a higher batch number value is expected which leads to compromised metric scores, the tweaking process of another parameter could be utilized to improve it.

3. Even though different value of dense layer parameter has resulted on similar performance metric, the higher dense layer value is still preferred since it offers more metric scores and less losses compared to other.

4. The dropout parameter tweaking process has shown that the model proposed has better performance with lower dropout value. Hence, the author has preferred the range between 0.3 and 0.5 as the dropout value since this range has resulted on higher metric scores compared to other.

5. Higher convolution filter value has shown better performance which implies that it can be used to handle the compromising of large batch value problem.

6. The model proposed on this undergraduate thesis has exceed expectations as the evaluation result with k-fold cross-validation technique shown a stable and consistent high result. This also means that the model implemented is scalable and accurate enough to be implemented continuously in the future.