Machine Learning Approach for Intrusion Detection System to Mitigate Distributed Denial of Service Attack Based on Convolutional Neural Network Algorithm

1st Muhammad Raksi Erlambang
Department of Telecommunications Engineering
Telkom University
Bandung, Indonesia
mraksie@student.telkomuniversity.ac.id

2nd Ida Wahidah Hamzah
Department of Telecommunications Engineering
Telkom University
Bandung, Indonesia
wahidah@telkomuniversity.ac.id

3rd Favian Dewanta
Department of Telecommunications Engineering
Telkom University
Bandung, Indonesia
favian@telkomuniversity.ac.id

Abstract*— The rapid development of data communications alongside its nature to be protected and secured have resulted in a long on-going research and development of Intrusion Detection System (IDS). One of many approaches for improving IDS is by using Machine Learning (ML) method. This research proposes to build an IDS model using Convolutional Neural Network (CNN) algorithm which is a specialized type of ML. This research is conducted by converting CSE-CIC-IDS2018 samples into a pixelate image as an input to the IDS model to classify between benign and malicious network traffic. The best model will be chosen by comparing performance metrics of each model on different parameter combinations and the final model will be evaluated with k-fold Cross-validation technique to make sure the finest performance is obtained. The results obtained on this research are performance metrics scores that is higher than 93% for all of the parameter combinations. Based on the final result obtained, the authors concluded that the model proposed on this research is not only successful, but also is better compared to other traditional ML-based IDS in terms of performance metrics.

***Keywords*— Intrusion Detection System, Machine Learning, Convolutional Neural Network**

I. INTRODUCTION

Along with the development of information technology and the convergence of technology with current digital lifestyle today, information security has become more relevant and needed than ever. By 2023 there will be 66% of global population that has internet access where the numbers of Internet Protocol (IP) connected devices will reach 29.3 billion caused by Internet of Things up from 18.4 billion in 2018 [1]. The inevitable trend of information technology development which on track with the increasing dependencies on robust and reliable data communications network makes information security a crucial aspect of functioning modern society. This phenomenon will occur worldwide as it is an essential nature to apply security approaches on wider and more crucial range of fields than before such as automated driving, healthcare, and energy. All of this is just the beginning of a more sophisticated form of digitalization.

With all of this advanced development, these huge number of statistics raise new challenges as the development of information security also in tune with the rise of new and more sophisticated threats such as exploits and vulnerabilities in worldwide data network. One of those challenges is an anomalous network dataflow that is also known as intrusion or a breach on a network. A breach on a network nowadays means lots of possibilities ranging from an information gets stolen to an attack to the network functionality and availability widely known as Distributed Denial of Service (DDoS) [2]. According to Cisco, 1Gbps DDoS attack alone is enough to take most organizations completely offline [1]. Added with 23% of those attacks are greater than 1Gbps, a dynamic and scalable approach to protect network and information from unwanted hands has become one of the utmost importance. Intrusion Detection System (IDS) is one of many methods to overcome this problem.

Intrusion Detection System (IDS) is a dedicated system formed as a software or hardware to run the intrusion process automatically. The ultimate goal of an IDS is to be able to identify different kinds of malicious intrusion with high detection rate and low false alarm rate, which cannot be achieved by firewall. IDS has a long history of appliance with traditional rule-based or pre-determined set of rules to differentiate between mundane and malignant traffic. This approach however is not reliable in the long run since it will unable to detect malicious traffic intelligently and continuously.

Due to its rapid demand and the need to adapt to more complex threat in the future, Machine Learning (ML) technique can be used as a solution to reinforce an IDS. ML can improve conventional IDS methods in terms of effectiveness and efficiency by its capability to process and learn new patterns automatically [3]. ML is sought to be implemented on IDS because of the rapid nature of the development of ML technologies by both academia and industry. ML also has been used and applied in many other technological disciplines and has shown significant positive transformations. The versatile and learnability characteristic of ML is expected to strengthen and improving the existing conventional IDS in an unimaginable possibility [4].

Previously, there was numbers of research on this topic regarding on how to apply an ML method on IDS. With stunning results and scores, these researches have managed to raise new standard and opened new pathway for other researcher to striving on this topic. However, these researches were mainly dominated by the combination usage of traditional ML method and old type of datasets as a training and validation subject. To fill in the gap of this issue, this Final Undergraduate Thesis proposes the using of newer type of ML method which are Convolutional Neural Network (CNN) algorithm and Neural Network type of ML on a newest and designated dataset of a captured network traffic that consisted of detailed descriptions of intrusions and abstract distribution models of applications, protocols, and lower-level network entities. This is due to the fact that Neural Network with CNN method has gained massive success and popularity among many fields of study recently, even though the implementation of CNN is mainly used in computer vision field [5]. One of the biggest challenges of this undergraduate thesis is to make this implementation and experimentation possible through designing a set of rules of program to apply CNN method on a network traffic dataset. Another main reason to choose Neural Network over another traditional ML method is because Neural Network offers more advantages such as better feature representation, performance, and less false alarm.

II. THEORITICAL REVIEW

In this experiment, we design an IDS that has been trained with CNN and Neural Network algorithm. The traditional method to train an IDS is usually done by purposely generate a personal or dedicated real network traffic by oneself. Although it's possible to do so, most of handcrafted network traffics are usually rather limited in coverage and considerably debatable in their integrity. Public dataset comes to resolve this issue. In this case, this experiment uses CSE-CIC-IDS2018 as it is one of the newest public traffic datasets available in this research field. Furthermore, all of the experiment process was conducted on Google Colab platform. Colab Pro tier is chosen in consideration that this experiment needs a more dedicated size of Random Access Memory (RAM) and resource priority [8].

FIG. 1.
VISUALIZATION OF FIVE CONVERTED INSTANCES

In general, there are two parts that are going to be conducted in this experiment; Pre-processing and building the ML model by training and evaluation [3]. The pre-processing is one of the most important processes to transform the various raw dataset feature attributes into a meaningful and valuable data that the ML model would understand. This is due to the key elements to build an IDS with CNN algorithm is by mapping the one-dimensional instances of the dataset into a type of data that is accepted into the ML model, which is image type of data. Another key reason is because the CNN model is far better for processing image type of data compared to another type of data [5]. This experiment will exploit one of the advantages of CNN and Neural Network algorithm by making this experiment relies on the concept of image classification. Started with 80 features from each instance in CSE-CIC-IDS2018 dataset, this experiment will end up on 79 features that will transformed into 13x6 pixels image type data excluding its label feature as shown on Fig. 1 which is the acceptable input for CNN and Neural Network model to execute.

A. *System Block Diagram*

The ML model on IDS starts when the model is provided with input data which come from previous pre-processing step. Generally, the IDS core system is a method to distinguish normal and malicious traffics as shown in Fig. 2. After the pre-processing step, the next step is to split the CSE-CIC-IDS2018 dataset into training and testing block. Then, the core IDS system of this experiment would be consisted on training the CSE-CIC-IDS2018 block by a combined CNN and Neural Network algorithm to form a classification model algorithm.

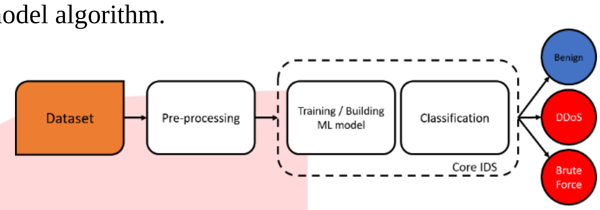

FIG. 2.
DIAGRAM OF CORE IDS

The 5,12 gigabytes out of 6,41 gigabytes of raw training dataset used on this experiment are labelled normal or various types of intrusion. In this case, this experiment categorizes the intrusion labels into two types which are DDoS and Brute Force. DDoS group of attack consists of over than 680,000 High Orbit Ion Canon (HOIC) attack and 1730 Low Orbit Ion Canon (LOIC) attack. On the other hand, Brute Force group of attack consists of over than 190,000 File Transfer Protocol (FTP) Brute Force attack, 180,000 Secure Shell Protocol (SSH) Brute Force attack, 611 Brute Force attack on Web, 230 Brute Force attack on Cross Site Scripting (XSS), and 87 Structured Query Language (SQL) Injection attack. Table 1 shows the representation of CSE-CIC-IDS2018 data used on this experiment.

TABLE 1
Distribution of CSE-CIC-IDS2018 dataset used in this experiment

Traffic type	**Traffic numbers**	**Distribution (%)**
Benign	3124681	74.49
DDOS attack-HOIC	686012	16.35
FTP-BruteForce	193360	4.6
SSH-BruteForce	187589	4.4
DDOS attack-LOIC-UDP	1730	0.04124
Brute Force -Web	611	0.01456
Brute Force -XSS	230	0.00548
SQL Injection	87	0.00207
Total	**4194300**	**100**

B. *System Work Flowchart*

As mentioned before, in general, this experiment consists of two major parts. The first part is pre-processing. After pre-processing, the next part will be model building. In model building, the data will be trained on the ML algorithm to attained the desired model. These model eventually will get evaluated in testing process with fine tuning until the most satisfactory result achieved. In this case, the ideal result is performance metrics score with above than 90% rate and lowest loss score.

The full pre-processing part consisted of five steps of data cleaning. Those five steps start with filling the null values of each instance with zero value. This is because a dataset that is consisted of null values is redundant and could cause major inaccuracy on the IDS model. The next step is translating each and every infinity values on the dataset into an appropriate numeric value. Due to the dataset comes with a few of inconsistent value, we need to make sure the data is suitable for the ML model to process. In this case, all of infinity values on this dataset will be transformed into each attributes' maximum value. After that, any irrelevant attribute such as 'Timestamp' will be dropped as this attribute doesn't bring any intrinsic value and relevance on the ML model. Furthermore, all of 78 features contained in each of dataset's instances are operated in normalization process. Normalization process is a feature scaling procedure to obtain same output value of each attribute of continuous data into the range between 0 and 1. This normalization process will be conducted with Min-max scaler technique. Accordingly, all of these instances' label will be converted into a meaningful numerical value using Label-encoder technique. This is due to the fact that every label attributes on each instance are on string type of data and the ML model expects numerical data as an input [6]. Lastly, all and each of instance is converted from vector type of data into a form which represents an image, which is matrix type of data. This experiment converts all the 78 features into a 13×6 matrix instance. Fig. 3 shows the flowchart of this experiment.

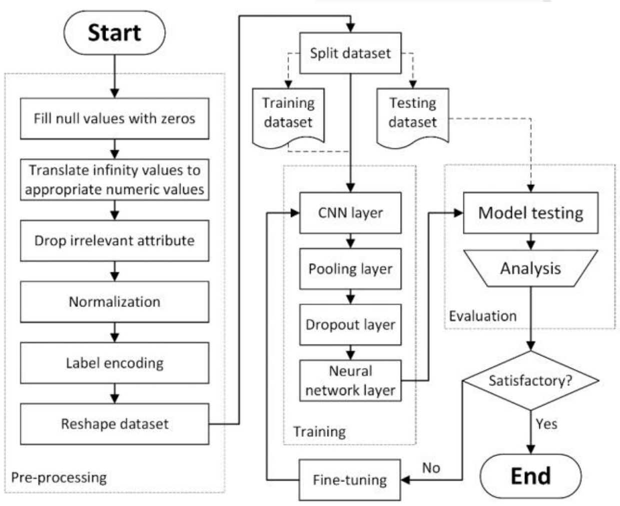

FIG. 3
FLOWCHART OF THIS EXPERIMENT

The full pre-processing part consisted of five steps of data cleaning. Those five steps start with filling the null values of each instance with zero value. This is because a dataset that is consisted of null values is redundant and could cause major inaccuracy on the IDS model. The next step is translating each and every infinity values on the dataset into an appropriate numeric value. Due to the dataset comes with a few of inconsistent value, we need to make sure the data is suitable for the ML model to process. In this case, all of infinity values on this dataset will be transformed into each attributes' maximum value. After that, any irrelevant attribute such as 'Timestamp' will be dropped as this attribute doesn't bring any intrinsic value and relevance on the ML model. Furthermore, all of 78 features contained in each of dataset's instances are operated in normalization process. Normalization process is a feature scaling procedure to obtain same output value of each attribute of continuous data into the range between 0 and 1. This normalization process will be conducted with Min-max scaler technique. After that, all of these instances' label will be converted into a meaningful numerical value using Label-encoder technique. This is due to the fact that every label attributes on each instance are on string type of data and the ML model expects numerical data as an input [6]. Lastly, all and each of instance is converted from vector type of data into a form which represents an image, which is matrix type of data. This experiment converts all the 78 features into a 13×6 matrix instance.

C. *Building the IDS Model*

After pre-processing, the next step is to build the core of the IDS. This experiment builds the IDS by training and evaluating an ML model. In general, the ML architecture of this experiment is shown in Fig. 4. This experiment's ML model starts with channeling each transformed instance into the first layer of the ML model, the CNN layer. The convolutional layer acted as the first input layer, followed by a pooling layer, dropout layer, and finally proceeded to a Neural Network layer with only a single dense layer that is fully linked to an output of three classes, namely Normal traffic, DDoS attack traffic, and BruteForce attack traffic.

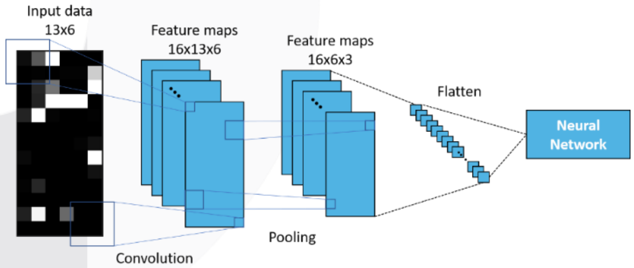

FIG. 4
VISUAL REPRESENTATION OF THIS EXPERIMENT ML MODEL

The first layer of this experiment's ML model, the CNN layer consists of filters, kernels, and activation function parameter. The filters are usually less abstract and generally emulates basic feature detectors to extract input data. The conservative values of CNN filter are 16 and 32. Researcher usually don't want a huge number of filters applied to the CNN layer due to its nature that the model will be more likely redundant if the filter value is too high. The kernel size of CNN layer determines of how large the filter matrix that moves over the convoluted data. The kernel size of a two-dimensional CNN algorithm usually ranges from 2x2 and 3x3 [5]. The last variable of this ML model's first layer, the activation function acts as an additional component on the network which adds non-linearity to the data. This is due to the nature of linear transformation function alone couldn't capture complex relationships. Eventually, this experiment uses Rectified Linear Unit (ReLU) as activation function. ReLU has become the de facto and default activation function for many researchers because it's easier to use and frequently achieves better performance compared to another types of activation function [5].

The second and third layer of this experiment's ML model is pooling and dropout layer. Quite similar with CNN layer, the pooling layer essentially reduces the spatial size of the input data without sacrificing its major features. Ultimately, the objective of pooling layer is to make the computational power more efficient through dimensionality reduction [7]. This experiment uses Max Pooling technique as this pooling type is also capable of being a noise suppressant to the data. Done with pooling, the dropout layer serves as a deterrent to overfitting the model which leads in poor performance when the model is evaluated in the benchmarking process. The value of dropout layer lies between 0 and 1. The best value of dropout layer is dependent on its dataset and ML model. This means there are no fixated or default value for this layer. However, usually the value of dropout layer performs better on the range between 0.3 and 0.8.

The fourth and last layer of this experiment is a single layer of Forward Propagation Neural Network that is going to be iterated for over a series of epochs. Abstractly in the background, this final layer executes a matrix-vector multiplication. The most important parameter for this layer is unit parameter. This parameter uses a positive integer as its input and represents the output size of the layer. Followed by activation function parameter, this layer also utilizes ReLU. Finally, the output of this layer is transmitted to the last output layer formed as the same single dense layer with three units. But different compared to its precedent, this layer utilizes Softmax as its activation function. This is due to the performance of Softmax is better for multi-class classification compared to other activation function.

III. METHOD

In general, the goal of this experiment is to finding the ideal conditions of the proposed 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 batch size. Furthermore, after the model with optimum result obtained by tweaking the parameters, 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 2 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 size of data input. Thus, smaller values will be more effective. For simplicity, the usage of this parameter will be abbreviated with '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 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 'B' on further figures and tables.

IV. RESULT AND DISCUSSION

The process to find the optimal IDS model is done by manually training and testing 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.

A. *Training and Testing Results*

From Table 2, the performance metrics shows that the initial proposed ML model has already attained stunning results with a steady 99% of 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. 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.

TABLE 2
PERFORMANCE METRICS OF BATCH SIZE VALUE TWEAKING.

	Acc. (%)	Prec. (%)	Rec. (%)	F1 (%)	Loss (%)	Time
C=4 DO=0.3 De=16 **B=128**	99.9673	99.96	99.96	99.96	0.2	10m 30s
C=4 DO=0.3 De=16 **B=256**	99.9367	99.93	99.93	99.93	0.51	5m 13s
C=4 DO=0.3	99.9646	99.96	99.96	99.96	0.24	3m 17s

De=16 **B=512**						
C=4 DO=0.3 De=16 **B=1024**	99.9129	99.91	99.91	99.91	0.4	2m 18s
C=4 DO=0.3 De=16 **B=2048**	99.9042	99.90	99.90	99.90	0.52	1m 37s
C=4, DO=0.3 De=16 **B=4096**	99.8740	99.87	99.87	99.87	1.03	1m 18s

The performance metrics on Table 3 shows a pretty similar behavior 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. 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.

TABLE 3
PERFORMANCE METRICS OF DENSE LAYER VALUE TWEAKING.

	Acc. (%)	**Prec. (%)**	**Rec. (%)**	**F1 (%)**	**Loss (%)**	**Time**
C=4 DO=0.3 **De=32** B=2048	99.9598	99.95	99.95	99.95	0.48	1m 37s
C=4 DO=0.3 **De=64** B=2048	99.9443	99.94	99.94	99.94	0.88	1m 37s
C=4 DO=0.3 **De=128** B=2048	99.9622	99.96	99.96	99.96	0.4	1m 33s

Moving to dropout value tweaking on Table 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 as shown on Table 4.

TABLE 4
PERFORMANCE METRICS OF DROPOUT VALUE TWEAKING.

	Acc. (%)	**Prec. (%)**	**Rec. (%)**	**F1 (%)**	**Loss (%)**	**Time**
C=4 **DO=0.5** De=128 B=2048	99.9634	99.96	99.96	99.96	0.91	1m 35s
C=4 **DO=0.7** De=128 B=2048	93.2599	93.25	93.25	93.25	31	1m 34s

Lastly, on Table 5, the convolution filter value of 16 has shown a slightly improvements of 0.03% compared to the value of 4 and 0.01% compared to the value of 8. This little improvement also come with a stable performance time. Furthermore, the convolution filter value of 16 also has shown the lowest loss metric score compared to all iterations conducted from the beginning of this experiment. With this result, the author finally chooses the combination of the convolution filter value of 16, the dropout value of 0.3, the neural network's dense layer value of 128, and the batch size value of 2048 for all the model parameter. This combination further will be evaluated by k-fold cross-validation technique in the next subchapter.

TABLE 5
PERFORMANCE METRICS OF CONVOLUTION FILTER VALUE TWEAKING.

	Acc. (%)	**Prec. (%)**	**Rec. (%)**	**F1 (%)**	**Loss (%)**	**Time**
C=8 DO=0.3 De=128 B=2048	99.9406	99.94	99.94	99.94	0.22	1m 34s
C=16 DO=0.3 De=128 B=2048	99.9725	99.97	99.97	99.97	0.16	1m 37s

B. *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 regularized / stratified k-fold cross-validation. Randomization and regularization however, are just additional k-fold characteristics 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.

From Table 6, the evaluation process shows that this IDS model also has a similar and consistent performance compared to the model in parameter tweaking 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.

TABLE 6
PERFORMANCE METRICS OF CONVOLUTION FILTER VALUE TWEAKING.

	Acc. (%)	Prec. (%)	Rec. (%)	F1 (%)	Loss (%)
C=16 DO=0.3 De=128 B=2048 ***k*=3**	99.9720	99.97	99.97	99.97	0.18

From Table 7, the author has concluded that this experiment has generated 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 are still showing results that are undeniably high and similar to the previous experiment.

TABLE 7
PERFORMANCE METRICS OF CONVOLUTION FILTER VALUE TWEAKING.

	Acc. (%)	Prec. (%)	Rec. (%)	F1 (%)	Loss (%)
C=16 DO=0.3 De=128 B=2048 ***k*=2**	99.9673	99.96	99.96	99.96	0.18

C. Analysis

From the set of stages of experiment above, the author has gathered some final analysis on how the parameter tweaking and application of CNN and ML model on IDS could achieve high performance. Those analysis has been summarized on the following points:

1. Among all of the parameters 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 the efficiency of computational resources.
2. Although a higher batch size value is expected which resulted on compromised loss scores, the tweaking process of other parameters 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 authors have 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. A higher convolution value has shown a better performance which we suggest that it can be used to handle the compromising large batch size value.
6. The model proposed on this research has exceed expectations as the evaluation result with k-fold cross-validation technique has shown a stable and consistent prominent result.

V. CONCLUSION

From the set of stages of experiment above, the authors have summarized the following points:

1. Based on the experiment, the result demonstrates that the application of Convolutional Neural Network algorithm can be used as the main tool to build an Intrusion Detection System through the technique of converting network traffic data into image form of data as a model input.
2. The approach proposed in this research has proven to detect the characteristic of a captured network traffic, namely benign and malicious traffics.
3. The model proposed in this research has proven to achieve higher accuracy compared to other traditional machine learning models on the same dataset.
4. Based on each of models that have been built, the scores of the model's performance metrics are highly dependant on each of the value of the machine learning parameters.
5. The parameter of batch size highly correlates with the model's time performance, while the parameter of convolution plays a role in determining the accuracy metric of the IDS model.

REFERENCE

[1] Cisco and/or its affiliates. 2020. *Cisco Annual Internet Report (2018-2023) White Paper.* [Online] Available at: https://www.cisco.com/c/en/us/solutions/collateral/executiveperspectives/annual-internet-report/white-paper-c11-741490.html

[2] B. A. Forouzan, 2007. *Data Communications and Networking, 4th Edition*. McGraw Hill Higher Education, 2007.

[3] K. Kim, M. E. Aminanto, and H. C. Tanuwidjaja, 2018. *Network Intrusion Detection using Deep Learning: A Feature Learning Approach*. Springer.

[4] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, 2019. Survey of intrusion detection systems: techniques, datasets, and challenges. *Cybersecurity*. **2:20** 1-2.

[5] S. Albawi, T. A. Mohammed, and S. Al-Zawi, 2017. Understanding of a Convolutional Neural Network. *International Conference on Engineering and Technology (ICET)*. 1-6.

[6] W. -H. Lin, H. -C. Lin, P. Wang, B. -H. Wu and J. -Y. Tsai, 2018. Using convolutional neural networks to network intrusion detection for cyber threats.

IEEE International Conference on Applied System Invention (ICASI). 1107-1110.

[7] A. Ajit, K. Acharya and A. Samanta, 2020. A Review of Convolutional Neural Networks. *International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)*. 1-5.

[8] T. Carneiro, R. V. Medeiros Da N´oBrega, T. Nepomuceno, G. -B. Bian, V. H. C. De Albuquerque and P. P. R. Filho, 2018. Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications. *IEEE Access, vol. 6*. 61677-61685.