3International Journal of Advanced Smart Convergence Vol.12 No.2 67-75 (2023)
http://dx.doi.org/10.7236/IJASC.2023.12.2.67

IJASC 23-2-7

Performance Comparison According to Image Generation Method in NIDS (Network Intrusion Detection System) using CNN

Sang Hyun, Kim

Professor, Department of Cyber Security, Youngsan University, Yangsan Campus, 288 Junam-ro, Yangsan, Gyeongnam, 50510, Korea
ksh50@ysu.ac.kr

Abstract

Recently, many studies have been conducted on ways to utilize AI technology in NIDS (Network Intrusion Detection System). In particular, CNN-based NIDS generally shows excellent performance. CNN is basically a method of using correlation between pixels existing in an image. Therefore, the method of generating an image is very important in CNN. In this paper, the performance comparison of CNN-based NIDS according to the image generation method was performed. The image generation methods used in the experiment are a direct conversion method and a one-hot encoding based method. As a result of the experiment, the performance of NIDS was different depending on the image generation method. In particular, it was confirmed that the method combining the direct conversion method and the one-hot encoding based method proposed in this paper showed the best performance.

Keywords: *Image Generation, NIDS, CNN, Direct Conversion, One-hot Encoding*

1. Introduction

Recently, many studies have been conducted on methods using AI (artificial intelligence) techniques in NIDS (network intrusion detection system) [1-3]. AI techniques are well suited to analyzing large amounts of network traffic data and identifying patterns. AI models that have learned from existing attack patterns can identify and respond to new malicious behaviors. NIDS can use this ability of AI to predict and prepare for new security threats.

Unbiased datasets are essential for learning and testing these AI techniques. Representative datasets that have been released so far include KDD CUP99 and NSL-KDD. The KDD CUP99 dataset is created based on data collected from the DARPA'98 NIDS evaluation program. It is a dataset consisting of a total of 42 columns, including a total of 41 attributes and a class representing the type of attack for each record.

NSL-KDD is a dataset proposed to improve the data bias and redundancy of KDD CUP'99 [4, 5]. The KDD CUP'99 dataset contains many duplicate recorders. NSL-KDD deletes the duplicate records of KDD

Manuscript Received: March. 26, 2023 / Revised: April. 2, 2023 / Accepted: April. 7, 2023
Corresponding Author: ksh50@ysu.ac.kr
Tel: +82-55-380-9522, Fax: +82-55-380-9249
Professor, Department of Cyber Security, Youngsan University, Korea

CUP'99 and includes attack types that are not in the training dataset in the test set [6, 7].

The NSL-KDD dataset often shows very high performance in the training process due to overfitting, but very low prediction performance in the testing process. The reason is that the test dataset includes attack types that are not in the training dataset. To solve this problem, various AI techniques are being utilized in NIDS. These attempts can be largely classified into two types: unsupervised learning and supervised learning. In general, unsupervised learning uses autoencoder methods, and supervised learning uses CNN-based learning.

First, in an environment where new attacks are developed every day and intrusion patterns are rapidly changing accordingly, unsupervised learning can be more effective in detecting new attacks than supervised learning, which assigns the correct answer to a dataset [8]. In particular, in the NSL-KDD dataset, since the test dataset includes attack types that are not in the training dataset, the unsupervised learning method that learns without correct answers may be more useful than the supervised learning method that learns based on correct answers. In [9-11], an unknown abnormal network traffic or intrusion on network datasets such as NSL-KDD and IoT datasets is detected using the various autoencoders. In these studies, various attempts are being made to find the optimal autoencoder model. Nevertheless, it is not certain that the autoencoder model, which compresses the input data and expands it again, can be applied to various situations.

Next, among the supervised learning methods, the CNN-based AI technique that automatically extracts the features of the dataset from these NSL-KDD datasets shows relatively better performance than general DNN (deep neural network) [12, 13]. In order to build a CNN-based NIDS, it is necessary to convert the NSL-KDD dataset into an image, and there is a large difference in performance depending on the image generation method. Direct conversion in [12] and one-hot encoding-based method in [13] are representative methods of generating images. In direct conversion, CNN is applied by repeatedly arranging 41 attributes of each record in the dataset to create a 28*28 size image. In this method based on one-hot encoding, character data is one-hot encoded among the attributes to expand 41 attributes to 122 attributes. Then, PCA (principal component analysis) is applied to the 122 attributes to obtain 100 or 121 attributes. From this, a 10*10 or 11*11 size images are created and CNN is applied.

In this paper, we focused on CNN-based NIDS. Performance comparison was performed according to the image generation method in CNN-based NIDS, which shows relatively excellent accuracy for the NSL-KDD dataset among supervised learning methods. Additionally, a new method combining the two image generation methods used in the performance comparison experiment was presented. For the experiment, we first implemented a DNN with three hidden layers. Direct conversion and one-hot encoding-based methods were used as image generation methods. The size of the image used in the experiment was similar to the image size previously used in each method, and two types of sizes, 12*12 and 28*28, were used. In addition, in order to make the image size 28*28 in the one-hot encoding-based method, a method of creating an image by combining direct conversion was proposed. As a result of the experiment, it was confirmed that the CNN-based methods showed relatively improved performance by about 10% compared to previous methods such as random forest and general DNN, which are machine learning methods. Among the image generation methods, it was confirmed that the method combining one-hot encoding and direct conversion showed the best performance.

2. Materials and Methods

2.1 NSL-KDD Dataset

NSL-KDD is a widely used dataset in the field of network intrusion detection and protection research [6, 7]. The NSL-KDD dataset was created by improving the limitations of the KDD CUP99 dataset, such as

duplicate records, limited variety of attack types, and unrealistic traffic scenarios. The NSL-KDD dataset retains the basic structure and functionality of the KDD CUP99 dataset, but removes duplicate records while introducing new attack types.

The NSL-KDD data set consists of network traffic data collected in a LAN (local area network) environment. This includes both normal network traffic and various types of attacks such as DoS (denial of service), R2L (remote to local), U2R (user to root) and probing attacks. The data set is divided into training and test sets. In this paper, we use train+ as a training data set and test+ as a test set for AI model learning. Table 1 shows the composition of the train+ dataset and the test+ dataset. Here train+ has a total of 125,973 records and test+ has 22,544 records respectively. It can be confirmed that test+ has more U2R and R2L attack type data than train+. For this reason, when the AI model is trained, it shows high performance, but when the actual trained model is tested with the test set, it can be seen that the prediction accuracy is relatively low.

Table 1. Construction of the NSL-KDD dataset

		Train+		Test+	
Normal	Normal	67,343		9,711	
Attack	Dos	45,927	58,630	7,458	12,833
	Probe	11,656		2,421	
	U2R	52		200	
	R2L	995		2,754	
Total		125,973		22,544	

Figure 1 shows the average values of numerically expressed attributes among 41 attributes in the NSL-KDD dataset. Here, the value range of the 5th and 6th attributes is too large, so the size is limited to "300". Since some attribute values locally have very large values in the figure, it is essential to normalize each attribute value by pre-processing before training or testing for successful AI model training.

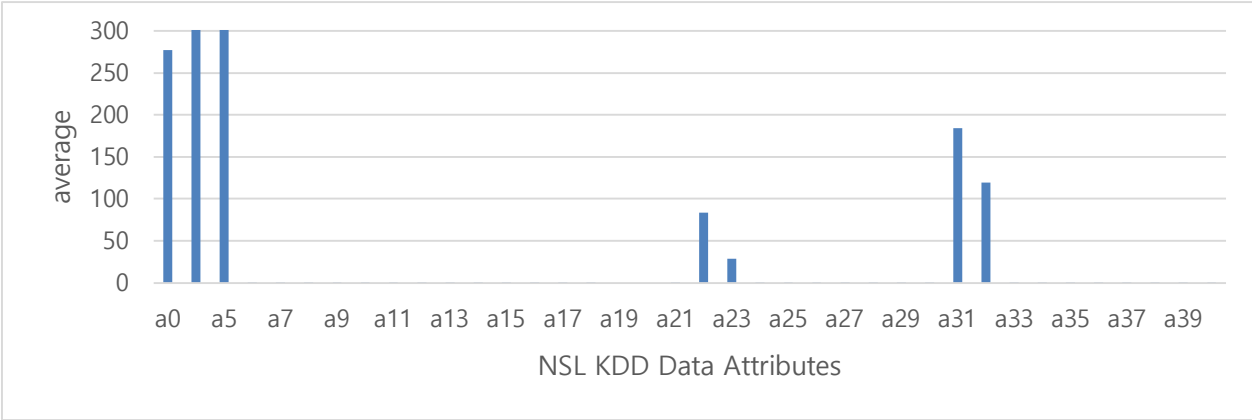


Figure 1. Average value distribution of NSL_KDD dataset attributes

2.2 Image Generation Based on Direct Conversion

As shown in Figure 1, the NSL-KDD dataset requires pre-processing because the average values of 41 attributes are distributed locally. In this study, each attribute value is processed as follows as a pre-processing to generate an image [12].

(1) Continuous data: In Figure 1, the 5th and 6th attributes, "src/dst data", are properties with actual values. And the range of values is very wide. Therefore, the range of values that each property can have is normalized between 0 and 255. The above processing is performed to express the value of each pixel of the image in 8-bit, 256 gray scale.
(2) Symbolic data: In the NSL-KDD dataset, the second to fourth attributes, that is, attributes with specific string values such as "service" or "protocol type" are preprocessed as follows. It measures the total number of possible symbols and assigns a value between 0 and 255. As an example, the "protocol type" attribute value has three symbolic values, which in this case are normalized to 0, 128, and 255.

Next, an image is created based on the pre-processed data. The basic concept of image generation is to express each attribute value as one pixel of the image. In this study, the specific details of the image generation method called direct conversion are as follows. In Figure 2, after dividing the packet data into attribute, each attribute is made into an 8-bit vector by the preprocessing method described above. After that, each attribute value is sequentially inserted into one pixel. Since there are 41 attribute values, attribute values can be repeatedly arranged to create a 28*28 image.

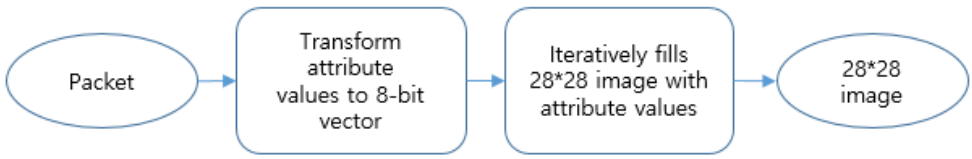


Figure 2. Image generation based on Direct conversion

This method has the characteristic that the combination of pixels expressing attribute values applied to filtering in the CNN convolution process can be configured in various ways. On the other hand, as the size of the image increases, too many iterations are performed when generating the image, so that the same pixel pattern continues to occur, so there may be a limit to performance improvement.

2.3 Image Generation Based on One-Hot Encoding

In this study, the process of making the NSL-KDD data set into an image without repetition using one-hot encoding is as follows [13].

(1) One-hot encoding: 3 character-type attribute values among 41 attributes are converted into numbers using the one-hot encoding method. For example, the “protocol type” values (tcp, udp, and icmp) in the second column convert to binary vectors [1, 0, 0], [0, 1, 0], [0, 0, 1] respectively. If this one-hot encoding method is applied to all three character-type attribute values, 41 attribute values are changed into a total of 122 attribute values.
(2) Normalization: Except for 3 character-type attributes, the remaining attributes are mapped in the [0, 1] range using min-max normalization. Since the attributes of the one-hot encoded character type have a value of 0 or 1, the remaining attributes are mapped in the [0, 1] range without disturbing the linear relationship between each attribute.
(3) 2D vector mapping: The initial 1D data (1∗122) is preprocessed and converted into a 2D feature vector to apply to CNN. That is, a 1*122 vector is converted into n*n image data. At this time, the dimension of the 1-dimensional vector (1*122) is reduced using PCA (principal component analysis) to make a

1*121 or 1*100 vector. The values of this reduced 1-dimensional vector are mapped to pixels in a 1:1 ratio and finally reconstructed into a 2D image of 11*11 or 10*10 size without repetition.

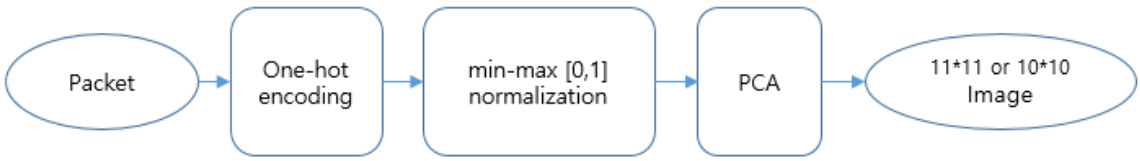


Figure 3. Image generation based on One-hot encoding

This method can simply create an image without repetition because the image applied to the convolution process of the CNN is filled with 1*122 attribute values. However, there is a disadvantage of losing the original feature value of an image by reducing the dimension using PCA to create an image. In order to improve the performance of CNN, correlation between attribute values in an image should be used. Therefore, it is necessary to create an image without reducing the dimensionality or to enlarge the image. If a 28*28 image is created by repeatedly utilizing the attribute values as in the direct conversion method mentioned above, it is expected that the image can be effectively created without excessive repetition and the performance of the CNN can be improved because there are already 122 attribute values.

3. Proposed Method for Performance Comparison

The following CNN model was implemented to compare the detection performance of the CNN-based NIDS system according to the image generation method. In the figure, the two types of image sizes used in the image generation process were 12*12 and 28*28. Direct conversion based method and one-hot encoding based method were used as image generation methods. In the convolution layer, since the image is small, one layer applying a 3*3 filter is implemented. In DNN, a fully connected layer with three hidden layers was constructed.

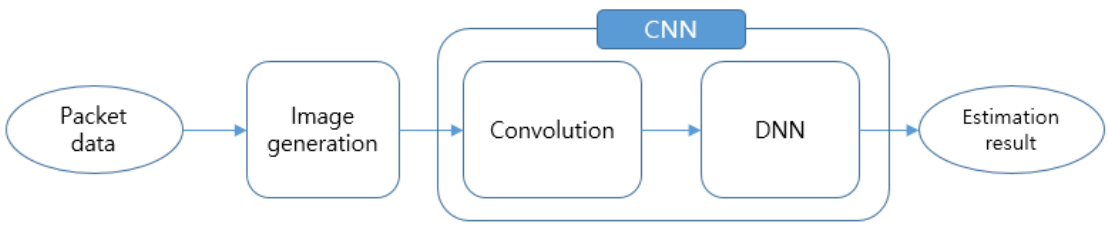


Figure 4. Proposed method for performance evaluation of NIDS according to image generation method

(1) Image generation: A total of four image generation methods were used by combining image size and image generation method.
CNN-IG1: 12*12 image size and Direct conversion
CNN-IG2: 12*12 image size and One-hot encoding with zero padding
CNN-IG3: 28*28 image size and Direct conversion
CNN-IG4: 28*28 image size and One-hot encoding with Direct conversion
In CNN-IG4, in order to generate a 28*28 size image, a one-dimensional vector of 1*784 (28*28) is

obtained by repeatedly placing one-hot encoded 1*122 1-D vectors using the direct conversion method.

(2) Convolution layer: Since the image size is small, a convolution layer with one layer applying a 3*3 filter is implemented. The number of filters used is 32, and the activation function is “relu”. The size of the image was reduced using Max-Pooling with a filter size of 2.

(3) DNN: It is implemented as a sequential dense type model (FCL: fully connected layer) with three hidden layers. The number of neurons used in each hidden layer is 60, 30, and 10, and the activation function is 'relu'. The attack type was binarized into “normal” and “attack” for training and testing.

Figure 5 shows a detailed block diagram of the proposed CNN method. The figure also shows the number of feature parameters created at each stage when a 28*28 size image is input.

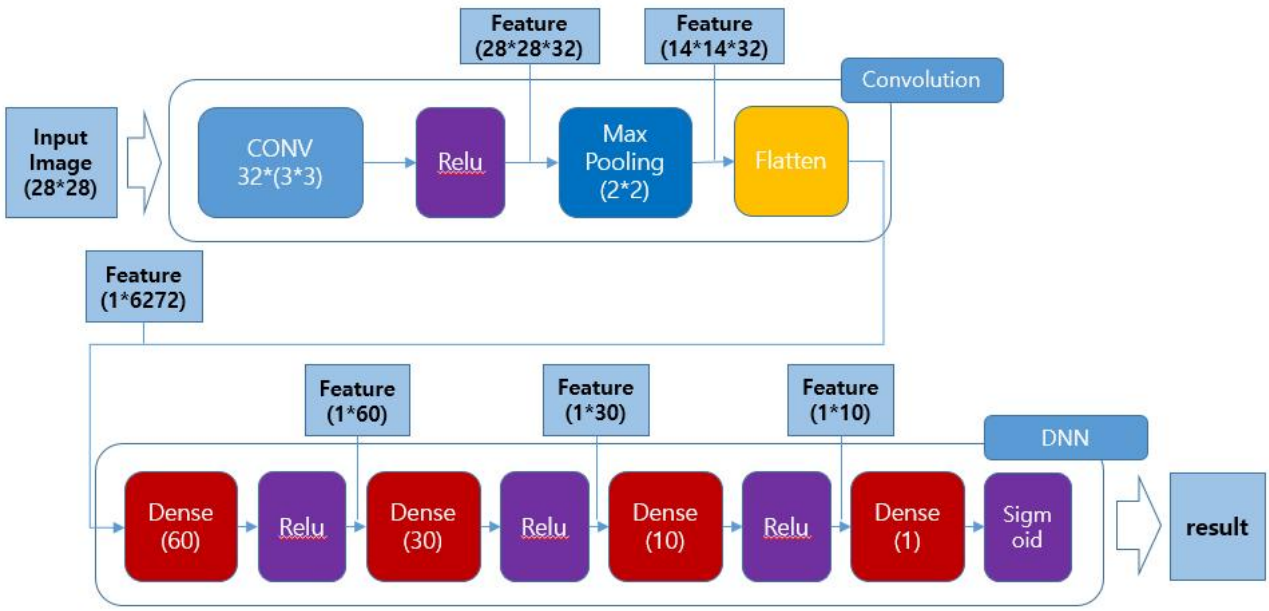


Figure 5. Detailed block diagram of the proposed CNN method

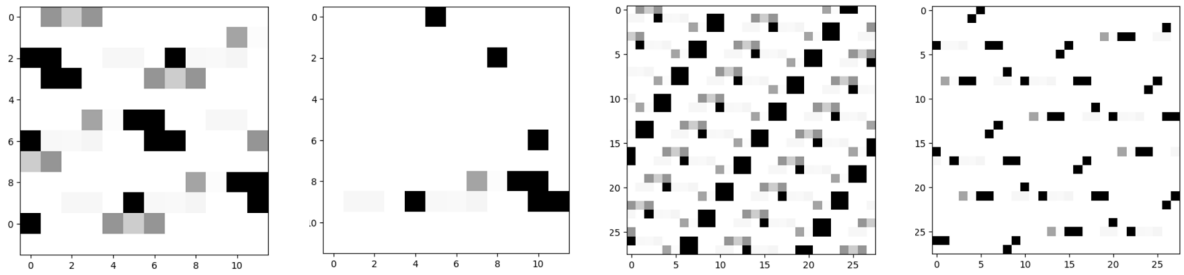
4. Results and Discussion

In this paper, the NSL-KDD dataset's training dataset (train+) and test dataset (test+) were used in the experiment to compare the CNN-based NIDS performance according to the image generation method. For binary supervised learning, the attack types in each dataset are unified as “Attack”. That is, data with two labels, “Attack” and “Normal”, were used for training and testing.

As an experimental environment for performance evaluation, Tensorflow and keras were used in Google Colab. First, 'binary_crossentropy' was used as a loss function and 'adam' was used as an optimizer for training in deep learning. The default value of 0.001 was used as the learning rate of 'adam'. The number '0' was set as a random number seed value for initializing the weights of the deep learning neural network. The number of training iterations, epoch, was set to 100, and batch_size was set to 50. Next, in Random Porest, the number of decision trees was set to ‘10’, the max_depth of the decision tree was set to ‘2’, and the random_state was set to ‘0’.

Figure 6 shows images of the 1000th training data using each image generation method. In the figure, the

brightness value of the pixel is normalized to the range of [0, 255], and black and white are reversed. Figure 6(a) is the result of the CNN-IG1 method. It is shown that the same pattern is repeated about 3 times to create a 12*12 size image with 41 attributes. The image created by the CNN-IG3 method is shown in Figure 6(c). It can be seen that many repetitions appear. Figure 6(b) is an image generated by the CNN-IG3 method. This image was generated from 122 attributes created by one-hot encoding of character-type attributes and min-max normalization of the remaining attributes. The image is made without repetition. Figure 6(d) is an image generated by the CNN-IG4 method. Although the attributes were repeatedly arranged to create a relatively large image of 28*28, it can be seen that the number of repetitions is smaller than in Figure 6(c).



(a) (b) (c) (d)

Figure 6. Image results made using each image generation method (a) CNN-IG1(12*12); (b) CNN-IG2(12*12); (c) CNN-IG3(28*28); (d) CNN-IG4(28*28)

Table 2 shows the performance comparison results of the NIDS system applying various AI techniques. Here, the AI techniques used are Random Forest, DNN, and CNN when image generation is executed in the four methods described above. Table 2 shows four values of accuracy, precision, recall, and f1-score of each method for performance comparison. Random Forest shows the lowest performance on both training and test sets. Next, the performance of DNN is low. The NSL-KDD dataset contains many attack types in the test set that are not in the training set. Therefore, these simple methods create an AI model that is overfitted to the training set, making it difficult to show good performance on the test set.

In Table 2, it can be seen that the precision and recall values are significantly different. The reason why the precision value is low is that the test set includes many attack types that are not in the training set, so a model that overfits the training set identifies many attacks in the test set as normal. On the other hand, the recall value appears high because normal is mostly judged as normal.

In order to confirm that the model is overfitting to the training set, the change in accuracy for the training set and the change in the loss value for the test set, which occur during the DNN training process, are shown in Figure 7 In the figure, it can be seen that the accuracy of the training set converges to almost 100%. However, the change of the loss value for the test set does not converge and oscillates. These experimental results confirm overfitting of the model.

Table 2. Performance comparison results of the NIDS system

	Train-set	Test-set			
	Accuracy	Accuracy	Precision	Recall	F1_score
Random Forest	0.9183	0.7282	0.6162	0.9790	0.7563
DNN	0.9873	0.7557	0.6443	0.9960	0.7731
CNN(IG1) 12*12	0.9978	0.8037	0.6946	0.9714	0.8099

CNN(IG2) 12*12	0.9982	0.8105	0.7032	0.9692	0.8151
CNN(IG3) 28*28	0.9977	0.8020	0.6918	0.9747	0.8092
CNN(IG4) 28*28	0.9982	0.8211	0.7303	0.9273	0.8171

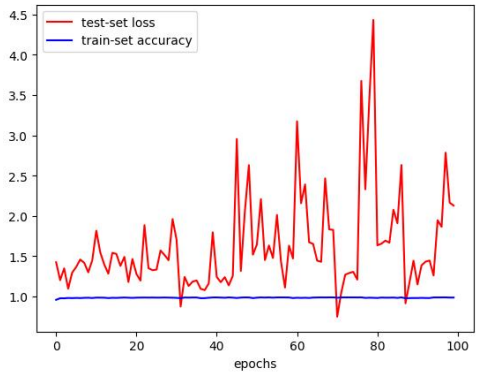


Figure 7. Accuracy on the training set and loss on the test set in the DNN method

Here, NIDS based on CNN shows relatively high performance. The CNN method extracts the correlation between pixels in a data set as a new feature by using an image processing technique in the convolutional layer. The features extracted by the image processing method can be regarded as more general features such as the overall distribution and shape of the dataset. We believe that CNN partially solves the overfitting problem that occurs during training due to these characteristics. Therefore, the process of generating images for applying CNN is very important.

CNN-IG1 method is a direct conversion method. This method constructs a 12*12 image by replacing 41 attributes with 41 pixels and then repeatedly arranging them. CNN-IG3 method constructs 28*28 images in the same way. The performance of these two methods is very similar even when the image is enlarged more than two times. The reason for this is that the pattern of pixels is already repeated several times when constructing a 12*12 image, and even if a 28*28 image is created again, the same patterns are added, so it is thought that there is no significant difference in performance.

On the other hand, CNN-IG2 method utilizes one-hot encoding and zero padding to construct an image using 122 attributes without repetition. CNN-IG4 method creates a 28*28 image by repeatedly arranging 122 attributes made by one-hot encoding using the direct conversion method. When creating a 28*28 image in CNN-IG4 method, attributes are repeatedly placed, and at this time, new patterns not found in the 12*12 image are created in the 28*28 image. Since additional features can be extracted from these patterns, it is determined that the performance of the CNN-IG4 method is better than that of the CNN-IG2 method.

As a result of the experiment, it was confirmed that the CNN-based NIDS showed relatively superior performance compared to the previous methods for the NSL-KDD dataset. In addition, there is a difference in performance depending on the method of generating an image to use CNN. When creating an image, it is important to ensure that the patterns that express the characteristics of the data can appear as diversely as possible. In this paper, it was judged that the method using the combination of the one-hot encoding-based method and the direct conversion method expresses the characteristics of the data well.

5. Conclusion

In this paper, we evaluated the performance of CNN-based NIDS according to the image generation

method. The NSL-KDD dataset used in the experiment does not have many attributes. Many attack types that are not in the training set are included in the test set. Therefore, structurally, a model that is overfitted to the training set is inevitable. CNN-based NIDS shows relatively good performance on NSL-KDD dataset. The reason for this is as follows. CNN is a technique that finds the correlation between pixels using an image processing filter and uses it as a characteristic of a model. It is judged that the characteristics found in this way have versatility and are used adaptively in the NSL-KDD test set.

The performance of CNN can be improved when patterns of pixels exist as diversely as possible. In this paper, it was confirmed that the pattern of pixels in the image generation method combining direct conversion and one-hot encoding is relatively diverse. As a result of the experiment, the CNN-based NIDS that generated images in this way showed the best performance.

Acknowledgement

This work was supported by Youngsan University Research Fund of 2022.

References

[1] Ahmad, Zeeshan, et al. "Network intrusion detection system: A systematic study of machine learning and deep learning approaches," Transactions on Emerging Telecommunications Technologies 32.1 e4150, 2021. DOI: 10.1002/ett.4150.

[2] Althubiti, Sara A., Eric Marcell Jones, and Kaushik Roy. "LSTM for anomaly-based network intrusion detection," 2018 28th International telecommunication networks and applications conference (ITNAC). IEEE, 2018. DOI: 10.1109/ATNAC.2018.8615300.

[3] Atilla Özgür and Hamit Erdem, "A review of KDD99 dataset usage in intrusion detection and machine learning between 2010 and 2015," PeerJ Preprints 4:e1954v1, 2016. DOI:10.7287/PEERJ.PREPRINTS.1954.

[4] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," Computational Intelligence in Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on IEEE, pp.1-6, 2009. DOI: 10.1109/CISDA.2009.5356528

[5] KDD CUP 99 dataset available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[6] Laheeb M. Ibrahim, Dujan T. Basheer and Mahmod S. Mahmod, "A comparison study for intrusion database (KDD99, NSL-KDD) based on self organization map (SOM) artificial neural network," Journal of Engineering Science and Technology, Vol. 8, No.1, pp.107-119, 2013.

[7] NSL-KDD dataset available: https://github.com/defcom17/NSL_KDD.

[8] McHugh, John, Alan Christie, and Julia Allen. "Defending yourself: The role of intrusion detection systems," IEEE software 17.5, pp. 42-51, 2000. DOI: 10.1109/52.877859.

[9] Zavrak, Sultan, and Murat İskefiyeli. "Anomaly-based intrusion detection from network flow features using variational autoencoder," IEEE Access 8, 108346-108358, 2020. DOI: 10.1109/ACCESS.2020.3001350.

[10] Y. R. Song, S. W. Hyun, and Y. G. Cheong. "Analysis of autoencoders for network intrusion detection," Sensors 21.13, 4294. 2021. DOI: 10.3390/s21134294.

[11] Jiho Jang, Dongjun Lim, Changmin Seong, JongHun Lee, JongGeun Park, and YunGyung Cheong, "Evaluating Unsupervised Deep Learning Models for Network Intrusion Detection Using Real Security Event Data," International Journal of Advanced Smart Convergence Vol.11, No.4, pp. 10-19, 2022. http://dx.doi.org/10.7236/IJASC.2022.11.4.10.

[12] W. Y. Jo, et al. "Packet Preprocessing in CNN-based network intrusion detection system," Electronics 9.7 1151, 2020. DOI: 10.3390/electronics9071151.

[13] Yihan Xiao, Cheng Xing, Taining Zhang, and Zhongkai Zhao, "An intrusion detection model based on feature reduction and convolutional neural networks," IEEE Access. 7:42210-42219, 2019. DOI: 10.1109/ACCESS.2019.2904620