Study on Network Intrusion Detection Method Using Discrete Pre-processing Method and Convolution Neural Network

JiHoon Yoo¹, ByeongJun Min¹, Sangsoo Kim², Dongil Shin¹, Dongkyoo Shin¹

¹Department of Computer Engineering, Sejong University, Seoul 05006, South Korea
²Agency for Defense Development, Daejeon 305600, South Korea

Corresponding author: Dongkyoo Shin (e-mail: shindk@sejong.ac.kr).

This work was supported in part by the Defense Acquisition Program Administration, and in part by the Agency for Defense Development under Contract UD2000014ED.

ABSTRACT The network intrusion detection system is a core technology of network security that detects packets for malicious activities occurring in the network and is an essential element for stable services in extended network environments such as big data and IoT. These network intrusion detection systems have been studied together with machine learning and deep learning, but performance is not guaranteed in the actual environment or the class balance problem has not been solved. Therefore, in this study, we investigate the performance of a discretization preprocessing method with a CNN-based classifier on the class imbalance problem of network traffic data. The preprocessing method adds a discretization algorithm for continuous variables in the commonly used conventional preprocessing method and converts 1D network packet vectors into 2D image vectors to improve relational analysis and generalization performance. Since the convolution neural network has immutability to the input data, it improves statistical efficiency in learning network packets converted into images. To evaluate the proposed model, we compared the computational complexity and generalization performance using NSL-KDD and CSE-CIC-IDS 2018, which is a representative network packet data. As a result of the experiment, it was confirmed that in the case of computational complexity, training time and parameters were reduced compared to the model designed similarly to the proposed model, and accuracy and F1 score improved in generalization performance.

INDEX TERMS Convolution Neural Network (CNN), CSE-CIC-IDS 2018, Discretization, NSL-KDD, Network Intrusion Detection System (NIDS)

I. INTRODUCTION

A Network Intrusion Detection System (NIDS) is a key technology in network security that detects packets of malicious or unwanted abnormal activity occurring in the network. [1] It has become an essential element to provide stable services in extended network environments such as big data and the internet of things (IoT) as well as general networks. [2, 3] Fig. 1 shows how these NIDS detected packets for abnormal activity. The NIDS intrusion detection method works very similarly to the classification systems of machine learning and deep learning, and it is like a real-time automatic knowledge generation problem that automatically generates knowledge by applying a learning algorithm to data. [4] The application of learning algorithms to NIDS over the past few years has achieved remarkable success and progress in intrusion detection. However, intrusion detection studies applying a learning algorithm to these NIDSs still have the following problems:

1) The performance of learning-based NIDS cannot be guaranteed in a real environment. Most learning-based NIDS studies are based on shortened, balanced datasets rather than large, unbalanced original datasets. Therefore, you are heavily dependent on the learned data, which prevents you from catching persistent problems with the real world. [5, 6]

2) In the case of network packet data, it is composed of mixed features, making it difficult for the model to analyze the relationship of predictors.
3) It doesn’t solve the class imbalance problem. It is a problem that is misclassified by a majority of an unbalanced dataset and corresponds to the generalization performance of the training model. Certain classes of a lot of data show high classification performance, while minority classes have very low classification performance, indicating a high overall average performance. However, if you use generalized performance metrics for a model such as F1-Score, you can see that the model is trained to be biased to a specific class. [7, 8]

To solve this problem, many studies have been conducted in the pre-processing part, but only the general pre-processing method is used and other pre-processing methods are not performed. [9] Also, compared to various learning and deep learning studies, there is a lack of research on CNNs. [10] For preprocessing, use it to prevent incomplete data such as data entry errors, measurement errors, experimental errors, and natural outliers from being heavily reflected in analytical and statistical modeling results. Therefore, for network packets composed of various mixed variables, each pre-processing according to the composed variables can greatly contribute to performance improvement.

The CNN model is a deep learning model that has been proven to be a good classifier in many fields. Since one trained filter performs a convolution operation while moving the entire image vector (Parameter Sharing), a single filter can features can be extracted. Because the same features can be seen in multiple places, it has the effect of learning more data, enabling efficient learning that consumes less memory and computation on data. [11, 12]

In this study, to solve this problem, we propose a discrete preprocessing method and NIDS using CNN. Contributions to this study include:

1) To reflect the network packet data in the real world, we use NSL-KDD and CSE-CIC-IDS 2018 data composed of unbalanced data. Especially for the CSE-CIC-IDS 2018 data set, the data collector is available as open-source, which could help problems with the real world.

2) For network traffic composed of mixed features, the relationship between various feature relationships is converted into data that can be easily analyzed by adding a discretization process for continuous variables in the preprocessing process. It also converts 1D network packet data into 2D image vectors for training on CNN model.

3) We evaluate whether the generalization performance of unbalanced data is improved by comparing Accuracy, Precision, Recall, and F1-Score performance by learning the processed and unprocessed data through the method proposed in the CNN model.

The composition of this paper is as follows. Section 2 introduces the related works, and Section 3 describes the analysis of the NSL-KDD Dataset, which will be used for experimentation and evaluation. Section 4 describes the approach and structure proposed to this study, and Section 5 shows the experiment. Section 6 summarizes the findings and provides directions for future works.
II. RELATED WORKS

As mentioned earlier, research that combines machine learning and deep learning-based NIDS has been conducted through various approaches. In the case of supervised learning, it learns a pattern to classify and label the target data by learning labeled data such as attack or normal on network traffic; in the case of unsupervised learning, feature extraction on unlabeled data; Perform Principal Component Analysis (PCA) and Clustering. [13] Guan et al. (2003) [14] proposed a new clustering method through the Y-mean algorithm modified from K-mean. The proposed model was tested for the KDD CUP 99 dataset and showed high detection rate and false alarm rate. Yao et al. (2006) [15] achieved better performance than the existing Support Vector Machine (SVM) by using a weighted kernel function that fit the data features of the SVM. Lakhina et al. (2010) [16] A model that combines the PCA algorithm and ANN Classifier was proposed. As a result of experimenting the proposed model with the NSL-KDD dataset, it was confirmed that reducing the input characteristics through the PCA algorithm reduce the resources and time required for training and results in better classification performance. Chae et al. (2013) [17] proposed Feature Selection using attribute ratio (AR) and compared it with other Feature Selection. DT is used as the classifier model, and it can be seen that the entire data feature is used or higher accuracy than other feature selection methods.

The core of deep learning is the study of extracting feature expressions to approximate the expected output based on data, and extracting feature expressions that are more abstract than existing machines for attack and general network features. Potluri et al. (2016) [18] proposed a DNN network composed of convolutional, max-pooling and fully connected layers. It was confirmed that the proposed model can effectively analyze the learning time and detection for NSL-KDD data. Y. Mirsky et al. (2018) [19] was proposed by Kitsune's core algorithm (KitNET), which consists of an ensemble of autoencoders(AE) and statistical pattern monitoring of network traffic. Through the actual IP camera video network data, it detected local network attacks in an efficient online manner without supervision, and showed sufficiently efficient and powerful detection capabilities. Aceto et al. (2019) [20] was proposed by multi-modal deep learning-based mobile traffic classification (MIMETIC), a new multimodal deep learning framework of encrypted traffic classification. In this study, an effective solution to an encrypted traffic classification that can support difficult mobile scenarios and overcome the performance limitations of the existing single-modal deep learning-based traffic classification was presented. Sharma et al. (2019) [21] proposed a model using t-stochastic neighbor embedding (t-SNE, similarity measurement) and kernel principal component analysis (KPCA, dimensional reduction) algorithms to learn non-image data into the CNN algorithm. With the proposed model, cancer cell vectors are compressed into two-dimensional dimensions and transformed into pixels using Cartesian plane coordinates.

The transformed cancer cell samples image showed very promising results from the CNN algorithm. Buturovic et al. (2020) [22] transposes a two-dimensional compressed data matrix through t-SNE for tabular data, maps feature of two-dimensional space, and creates images that reflect feature similarities. Sample images of blood data onto patients with infectious diseases were created and tested for CNN and Residual Neural Network (ResNet), and very high-performance classification results were shown. Lopez-Martin et al. (2020) [23] presents a method of performing supervised learning based on the deep reinforcement learning (DRL) framework. This study presents the results of applying the four algorithms deep q-network (DQN), double deep q-network (DDQN), policy gradient (PG) and actor-critic (AC) included in the DRL framework. Bovenzi et al. (2020) [24] proposed a hierarchical hybrid intrusion detection (H2D) model in which anomaly detection based on multimodal deep auto-encoder and soft-max classifier are combined. The proposed model was evaluated using the recently released Bot-IoT dataset, and provides high efficiency and flexibility required for IoT scenarios as well as performance.

Discretization preprocessing is used to solve problems with data with mixed features and unbalanced classes. This makes it easy to analyze the relationship between various feature relationships and aims to show more advanced generalization performance. [25] Aziz et al. (2014) [26] performed discretization using equal width interval binning for continuous variables, and it can be confirmed that improved performance is shown by using this for learning the GA search algorithm based on the concept of genetics. Shen et al. (2020) [27], proposed an algorithm that evolved data discretization preprocessing into MFPOF (Maximum Frequency Pattern Factor) based on FPFOF (Frequent Pattern Factor). This study showed good performance by applying discretization maximum frequent pattern factor data mining to NSL-KDD and UNSW-NB15 data. Zhang et al. (2019) [28] tried to solve the problem that an unbalanced dataset causes a bias toward multiple classes for a learning classification model. This study conducted an experiment on how a data set with appropriate discretization applied to continuous variables works in a cost-sensitive logistic regression model. From various performance evaluation results, it was confirmed that discretized variables provide reasonable coefficient estimates for model performance. Elhilbawi et al. (2021) [29] investigated the importance of discretization as a pretreatment step and argued that it achieves better classification performance compared to using continuous attributes. This paper compared the performance of several parametric and nonparametric discretization methods along with several machine learning classifiers on the problem of predicting intensive care unit (ICU) mortality. As a result of the experiment, the classification accuracy was 89.19% and 0.38, respectively, using the discretized data, whereas the classification accuracy F1 score was 86.19% and 0.08 when the continuous attribute was used.
This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

| Author (Year) | Research Direction | Input Data | Methods/Technique | Proposal Contribution |
|---------------|--------------------|------------|-------------------|-----------------------|
| Y. Guan (2003) | Machine learning   | KDD CUP 99, | Y-means clustering | Overcoming K-mean’s shortcomings, dependence on the number of clusters, and degeneracy. Data set can be automatically partitioned into an appropriate number of clusters. This feature detects known and unknown intrusions without labeling. Experiments with the KDD-99 data set confirmed that the proposed method is an effective way to partition large data spaces. |
| J. Yao (2006)  |                    | KDD CUP 99, | SVM               | Propose an improved SVM model that adds weights to the SVM kernel function. The performance is similar to the existing SVM, but the false-negative rate is lower. Much less cost and execution time than traditional SVM. |
| S. Lakhina (2010) |                  | NSL-KDD    | PCA, ANN          | We proposed the PCANNA algorithm, which is a combination of PCA and ANN algorithms. The proposed algorithm achieved 80.4% data reduction, 40% training time reduction, and 70% test time reduction for the NSL-KDD dataset. |
| H. Chae (2013)  |                    | NSL-KDD    | Feature Selection, Information Gain, Gain Ratio, ARF, DT | In this study, the performance of feature selection was evaluated to increase the computational efficiency of IDS construction. We compared the performance of various feature selection methods on the NSL-KDD data set, and among them, 22 features through the decision tree showed the best computational efficiency. |
| S. Potluri (2016) |                  | NSL-KDD    | DNN               | By proposing an accelerated DNN based on a parallel computer, the effect of anomaly detection on network traffic was confirmed. The proposed method conducts experiments on the NSL-KDD dataset, which can see an increase in training time and efficiency of the detection mechanism. |
| Y. Mirsky (2018) |                  | IP-Camera  | Kitsune, AE, Ensemble | Proposed KtINET consisting of an ensemble of AE and statistical pattern monitoring of network traffic. The attack was detected using real IP camera video network data, and it showed a sufficiently efficient and powerful detection function. |
| G. Aceto (2019) |                    | mobile encrypted traffic | MIMETIC        | It implicitly conveys information in a multi-modal manner, reflecting complex patterns extracted from traffic features. A new multi-mode deep learning framework of encrypted traffic classification that overcomes the performance limitations of the existing single-mode deep learning-based traffic classification proposal and supports difficult mobile scenarios. Multi-modality of traffic classification provides an effective solution to mobile scenarios as traffic can be inspected in a complementary view. |
| A. Sharma (2019) | Deep Learning     | TCGA, TIMIT, Relate | t-SNE, KPCA, A, CNN | Through the decision tree showed the best computational efficiency. |
| L. Buturovic (2020) |                  | Blood sample from patients with infectious diseases | t-SNE, CNN, ResNet | After transposing the two-dimensional compressed data matrix through T-SNE, features are mapped in a two-dimensional space, and an image reflecting feature similarity is created. Blood sample data on infectious disease patients are converted into images and classified through CNN and ResNet. |
| M. Lopez-Martin (2020) |                  | NSL-KDD, AWID | DRL Framework: DQN, DDQN, PG, AC | Proposal of a new algorithm that improves intrusion detection results compared to existing machine learning and deep learning technologies. Provide a comparative study of the four DRL algorithms and provides a way to apply them to datasets labeled as intrusions instead of interacting with the live network environment. |
| G. Bovenzi (2020) |                  | Bot-IoT    | M2-DAE            | The recently released Bot-IoT data set was used to evaluate the performance of four related categories of attacks and unknown attack inference. Attack classification by combining anomaly detection and soft output classifier through a new lightweight solution based on M2-DAE. |
| A. S. A. Aziz (2014) |                  | NSL-KDD    | interval binning, GA | New solutions such as genetic algorithms of biological systems have been proposed for network anomaly detection systems. A discretization method was used for selecting continuous features to generate some homogeneity between values with different data types. We experiment with the intrusion detection system performance on the NSL-KDD dataset to provide high performance of the proposed method and recommendations for future experiments. |
| L. Shen (2020)  | Discretization Methods | NSL-KDD, UNSW-NB15 | MFPOF based on FPOF | Since the data type of the anomaly detection model should be discrete data, the empirical basis was placed on transforming continuous variables into reliable and accurate data suitable for data mining through data discretization. In this paper, we propose the concept of MFPOF based on the FPOF of the FindFPOF algorithm and show that it is possible to further reduce the time complexity and high detection performance compared to the existing algorithm by discretizing two data sets. |
| Lili Zhang (2020) |                  | Imbalanced Credit Data | Variable Discretization | In order to improve the model performance on unbalanced data, each effort was made in terms of predictors and modeling algorithms. A detailed study of the credit score dataset demonstrates that appropriate variable discretization and class-dependent cost-sensitive logistic regression with best class weights help to reduce model bias and improve variance based on ROC curves and AUC. |
| Hanan Elhilwabti (2021) |                  | ICU Mortality | Discretization Methods | Discretization was argued to be a key technique used in various applications of machine learning and data mining, and to verify this, the effect of discretizing continuous attributes in ICU mortality prediction data was investigated. Various experimental results have shown that using discrete properties is better than using continuous properties. |
III. DATASET FEATURE ANALYSIS

To evaluate the model proposed in this study, we use two data sets: NSL-KDD and CSE-CIC-IDS 2018. The NSL-KDD data improved the loss of overload TCP dump packet, lack of attack definition, and many redundant and meaningless records raised by McHugh from the KDD CUP 1999 data developed by the Defense Advanced Research Projects Agency (DARPA). As listed in Table 2, they are classified into 5 classes: Normal, denial of service (DoS), pre-operation for vulnerability analysis before intrusion (probe), attempting unauthorized access from remote (R2L), and unauthorized access to take over root authority (U2R). Looking at the data ratio in Table 3, it can be seen that the class imbalance of the training and test data is not correct. [32]

| Dataset | Class | Attacks | Num |
|---------|-------|---------|-----|
| DoS     | apache2, back, land, Neptune, mailbomb, pod, processtable, smurf, teardrop, udpstorm, worm | 2 |
| Probe   | ipsweep, mscan, mmap, portscan, saint, stan, buffer_overflow | 6 |
| NSL KDD | U2R   | loadmodule, perl, ps, rootkit, sqattack, xterm ftp_write, guess-passwd, http tunnel, imap, multihop, named, phf, sendmail, snmpgetattack, spy, snmpguess, warezclient, warezmaster, xlock, xsnmp | 7 |
| R2L     | Normal | Normal traffic | 15 |

CSE-CIC-IDS 2018 data is a data set created for analysis and testing of intrusion detection systems focused on network anomaly detection in a joint project between the Communications Security Establishment (CSE) and the Canadian Institute for Cybersecurity (CIC). As listed in Table 2, they are classified into 7 classes: Normal, DoS, Distributed Denial of Service (DDoS), Botnet (Bot), BruteForce Attack (BruteForce), Infiltration (Infilter), Web Attack. Looking at the data rate in Table 3, it can be seen that the number of attack packets is very small compared to normal packets considering the actual network. Among them, the Infilter and Web Attack classes have very large imbalances with 0.09 and 0.005 respectively. In this study, considering these aspects, we validate a mode

| TABLE 3 DATASET CLASS DISTRIBUTION RATIO
| Dataset | Class | Number | Rate |
|---------|-------|--------|------|
| Total   | 125,973 | 100% |
| Normal  | 67,343  | 53%  |
| DoS     | 45,927  | 37%  |
| Probe   | 11,656  | 9.11% |
| U2R     | 52      | 0.04% |
| R2L     | 995     | 0.85% |
| Total   | 22,544  | 100% |
| Normal  | 9,711   | 43%  |
| Probe   | 7,458   | 33%  |

CSE-CIC-IDS 2018 Test

| Dataset | Class | Features | Information |
|---------|-------|----------|-------------|
| Total   | 12,986,354 | 100% |
| Normal  | 10,787,766 | 83.07% |
| DDoS    | 1,011,146 | 7.78% |
| DoS     | 523,440  | 4.03% |
| Bot     | 228,953  | 1.76% |
| BruteForce | 304,759 | 2.44% |
| Infilter | 129,547 | 0.09% |
| Web Attack | 743 | 0.005% |

CSE-CIC-IDS 2018 Train

| Dataset | Class | Features | Information |
|---------|-------|----------|-------------|
| Total   | 3,246,589 | 100% |
| Normal  | 2,696,942 | 83.07% |
| DDoS    | 252,787  | 7.78% |
| DoS     | 130,860  | 4.03% |
| Bot     | 57,238   | 1.76% |
| BruteForce | 76,190 | 2.44% |
| Infilter | 32,387  | 0.09% |
| Web Attack | 185 | 0.005% |

| TABLE 4 DATASET FEATURE INFORMATION
| Dataset | Features | Information |
|---------|----------|-------------|
| NSL KDD | Network Packet Header | 1-9 |
|         | Network Packet Payload | 10-22 |
|         | Time-Based Feature | 23-31 |
|         | Host-Based Feature | 32-41 |
|         | Label | 42 |

| CSE CIC IDS 2018 | Features | Information |
|------------------|----------|-------------|
| 46-63            | Network Subflow | 64-67 |
| 68-79            | Traffic features | 68-79 |
| 80-82            | General purpose traffic features | 80-82 |
| 83               | Label | 83 |

| TABLE 5 NSL-KDD FEATURE TYPE CLASSIFICATION
| Dataset | Class | Features |
|---------|-------|----------|
| Total   | 1-4   | Network Connections |
| Normal  | 5-16  | Network Packets |
| DoS     | 17-22 | Network Flows |
| Probe   | 23-45 | Statistic Network Flow |
| U2R     | 46-63 | Content-related traffic features |
| R2L     | 64-67 | Network Subflow |
| Total   | 68-79 | Traffic features |
| Normal  | 80-82 | General purpose traffic features |
| Probe   | 83    | Label |
The NSL-KDD and CSE-CIC-IDS data include various attribute information as shown in Table 4. NSL-KDD data includes not only information about the network, but also information about time and the host to add diversity to the data. The properties from 1 to 22 are typical values for the network, and the properties from 23 to 41 are composed of the number and ratio of the host and input traffic analyzed for 2 s. Because of these various features, some cases have been applied not only to NIDS studies but also to host instruction detection system (HIDS) studies. The CSE-CIC-IDS 2018 data set includes flow control such as bidirectional flow (forward and backward), and unlike the NSL-KDD data set, it is all made up of network-related properties. When the data set generated by Table 4 is classified as a feature type that can be used for machine learning, it is classified as shown in Table 5. Network packet characteristics of NSL-KDD are evenly distributed into four categories, and time-based and host-based characteristics consist of number and ratio, indicating that it mainly belongs to discrete type. It can be seen that the CSE-CIC-IDS 2018 data set is simply composed of binary and continuous variables, except for unnecessary attributes that are not required for training. If the difference in the range of data by the type of feature, data consisting of strings, and data consisting of various features are learned as they are, pre-processing is absolutely necessary because normal learning cannot be performed using machine learning and deep learning models.

IV. DESIGN AND IMPLEMENTATION OF DISCRETE PRE-PROCESSING METHOD AND CNN BASED NIDS

This paper proposes the system structure shown in Fig. 2 to improve the generalization performance of learning models for NSL-KDD and CSE-CIC-IDS 2018 datasets. First, after identifying the data characteristic type of each dataset, pre-processing was conducted by adding a continuous variable discretization algorithm to the general pre-processing method used in the proposed machine learning. In order to more clearly distinguish the relationship between different data, a square matrix is used to transform the network packet vector into an image pixel format with spatial characteristics. Finally, the accuracy, precision, recall, and F1-score of the machine learning and deep learning models were used to evaluate the performance of the proposed model.
A. Discretization of Continuous Variable

There are infinitely many Degrees of Freedom (DoF) in mathematical problems for continuous data, and these problems complicate model learning by creating nonlinear correlations for predictors because calculations cannot last forever. As shown in Fig. 3, the packet size feature, which is the most significant feature that distinguishes DoS attacks, is applied to the machine learning model, they are used in various ways between values of 0 and 100,000. In this case, it is easy to fall into the DoF problem where the number of variables that can be freely changed increases. In order to solve this problem, a characteristic consisting of continuous values is discretized and delivered as ‘1 when the packet size is large’ and ‘0 when the packet size is small’, which reduces the free fluctuation of the continuity variable and only information is transmitted. When discretized in this way, it is expressed in a linear relationship with the predictor variable and converted into data are is easy for the model to analyze, compared to a continuity variable with a large DoF.

B. Data pre-processing

Therefore, this study investigated a pre-processing method that adds a discrete algorithm for continuous properties to the general machine learning pre-processing process. Fig. 4 shows a pre-processing method that does not include the discretization algorithm, and a detailed description of the process is as follows.

1) Remove Unnecessary attributes from data attributes, such as zero standard deviation.
2) Nominal (categorical features) data composed of the object type are all encoded as the integer type and converted into a one-hot vector representation. Through this process, 3 nominal features of NSL-KDD are converted into 84 features.
3) For numeric (continuous features) data, because the image color channel value of [0-255] should be used as a feature, the data are normalized in the range of [0-255] through the min-max scaler.
4) Binary data are composed of 0 and 1, and used without special pre-processing.
5) Merges 4 preprocessed feature types into one. This translates 41 features in the NSL-KDD data set into 121 features and 72 in the 83 CSE-CIC-IDS 2018 data sets.
5) The above process merges the preprocessed feature types into one. The NSL-KDD data is converted to 179 features by the first method and 189 features by the second method. The CSE-CIC-IDS 2018 data is converted to 362 features using only the second method.

C. Network Traffic Image Conversion

The data for which the process was completed was expressed as data having a range of [0-255], which could be expressed in the black and white and color channels for an image. To process this as a convolutional operation, it was converted into a square matrix as shown in Fig. 6 and then converted into a pixel image using the process described in Fig. 7. It takes O(n) time complexity by default because it traverses the one-dimensional network traffic data to transform it into a pixel image. However, since additional iterations are required to create the 2D image vector during the traversal, the algorithm ultimately takes O(n^2) time complexity. 11 x 11 square matrices can be formed by transforming the 121 properties of NSL-KDD transformed in the preprocessing method without applying a discrete algorithm and 72 properties of CSE-CIC-IDS 2018 are transformed into a 9 x 9 square matrix. 179 and 189 features of NSL-KDD transformed in the preprocessing method using the discretization algorithm can form a 14 x 14 square matrix with zero paddings applied. 362 features of CSE-CIC-IDS 2018 transform into a 20 x 20 square matrix with zero paddings. In the process of matching with a square matrix, it created two types of image data to be used for training. The first was a color image composed of three color channels, red, green, and blue (RGB) colors overlapped to form an M x N x 3-pixel array. The second created an image with one grayscale color channel and was matched with an M x N x 1-pixel array. When the above pre-processing process was completed, the learning and evaluation data set shown in Table 6 was formed. In the case of M-1 and M-2, the data were completed through pre-processing without using the discretization algorithm. It is widely used in general machine learning and to compare the data results when the discretization algorithm is applied. In the case of D-1, D-2, CD-1, and CD-2, this was the data obtained through pre-processing with the discretization algorithm applied.

### Table 6

| Name | Pre-processing | Channel | Feature |
|------|----------------|---------|---------|
| M-1  | Non Discretization Algorithm | Gray Scale | 121 (NSL-KDD) |
| M-2  | Color            |         | 72      |
| D-1  | Discretization of Continuous Feature | Gray Scale | 179 (NSL-KDD) |
| D-2  | Discretization of Continuous and Discrete Feature | Color | 362 (CSE-CIC-IDS) |
| CD-1 | Discretization of Continuous and Discrete Feature | Gray Scale | 189 (NSL-KDD) |
| CD-2 | Discretization of Continuous and Discrete Feature | Color | 362 (CSE-CIC-IDS) |

### Algorithm: Imaging network traffic

```python
# x: Input Data (Network Traffic Array)  # bound: Square Matrix with Image
Input: x, bound
x_len = len(x)

x_img = np.zeros ((x_len, bound, bound))

for i = 0 to x_len do
    out = np.zeros (bound, bound))
    x_arr_len = len[i]
    sqr_x = int (x_arr_len ** 0.5)
    if sqr_x * x < arr_len:
        for i = 0 to bound do
            p = s + k % x_arr_len
            x_img[i] = out
        end
    for i = bound do
        p = (s + k % x_arr_len)
    end

return x_img
End
```

### Figure 5. Pre-processing with discretization algorithm

### Figure 6. Imaging network traffic using square matrix

### Figure 7. Algorithm: Imaging network traffic
**D. Convolution Neural Network Learning Model**

A CNN is a transformed neural network that uses convolution and aims to learn the feature representation of data, it has the following differences compared to a DNN, which is the most basic neural network used for deep learning. The basic layer structure of a CNN consists of a convolutional layer and a pooling layer, and various outputs such as classification and distance calculations between features can be used with a fully connected layer depending on the purpose of the learning. The pooling layer reduces the parameters connected between the convolutional layers, thereby reducing the amount of computation and improving the acceptance field of the subsequent convolutional layers.

A CNN can configure many convolution kernels by creating different feature maps in each layer. Because each region of neighboring neurons in the layer is connected to the neurons in the feature map of the next layer, the kernel can be shared in all spaces of the input. At this point, by tying adjacent shifts of the same weight together in a similar way to a filter, the convolutional layer can force feature learning with invariances to shifts in the input vector. This allows you to improve generalization for data sets by reducing the SGD optimization complexity required to represent equivalent shift-invariant features using fully connected layers. Because CNN's with this feature show high performances in the field of image and signal processing, this study, selected and designed a performance evaluation model for converting network traffic data into images. The proposed model had a basic CNN structure, where the structure and parameters of each layer used when constructing the model are specified in Table 7. As the input of the model, grayscale or color images were received as input data, and a performance evaluation was performed through multi-class classification for five normal and attack labels through the soft-max function of the output layer.

**V. EXPERIMENT**

An experiment was conducted by comparing the performance of the pre-processing method proposed in section 4 and the designed model with the performances reported in existing studies.

**A. Experiment Environment**

The data used for training for performance evaluation were NSL-KDD and CSE-CIC-IDS 2018 discussed in Section 3, and experiments are conducted based on the preprocessing and designed model proposed in Section 4. The experimental environment and parameters used in the experiment are presented in Tables 8 and 9, respectively.

**TABLE 8**

| Environment | Name |
|-------------|------|
| Language    | Python |
| Library     | Tensorflow 2.1.0 |
|             | Keras 2.2.4-tf |
| GPU         | Geforce RTX 2070 Super |
| Memory      | 64GB |

**TABLE 9**

| Parameter       | Value |
|-----------------|-------|
| Epoch           | 50    |
| Batch           | 256   |
| Optimizer       | Adam  |
| Learning Rate   | 2e-4  |
| Loss            | Sparse Categorical Cross entropy |

In order to evaluate the quantified performance, performance evaluation was performed using the four performance indicators that are used as general performance measures in the machine learning and deep learning classification models listed in Table 10: the accuracy, precision, recall, and harmonic average.
B. Experiment Result : Computational Complexity

In general, computational complexity is measured in "big-o notation," and is related to the amount of computation the model handles as the input data grows in time. However, competing neural network architectures in deep learning generally apply the same algorithm to the same type of problem, and most architectures use similar computational elements. Therefore, it is common practice to compare metrics such as learning time and parameters as a stand-in for evaluating computational complexity.

Table 11 shows the structure of the 1D-CNN model used to compare with the proposed CNN model, and it can be seen that it has the same structure as the proposed model. The two CNN models used in the experiment were trained using the training parameters shown in Table 9. In the case of 1D-CNN, data that has not been converted into an image is learned, and in the case of the proposed CNN, the data converted into an image is learned.

The experimental results from computational complexity are given in Table 12. In the case of 1D-CNN(x) of Method, x means M, D, and CD generated through the proposed preprocessing method. The difference here is that 1D-CNN models are trained without transforming the data into images for training. Looking at the experimental results, it can be seen that in the case of learning time/s, about 2-3 times the learning time is saved when the proposed method is used. For the parameters used for training, the M method, which combines basic pre-processing and image transformation, used more parameters than before. However, you can see that the number of parameters is reduced in the D and CD method, which combines discretization and image conversion. Through the above experimental results, it was confirmed that learning time/s and parameters can be reduced in CNN training combined with the proposed discretization and image conversion.

B. Experiment Result : Generalization Performance

The experimental results for each performance indicator are presented in Tables 13–16. Specifically, Fig. 9, 10 and Table 13 show the average and accuracy of the overall performance. It can be seen that the performance of the model trained

---

**TABLE 10**

**PERFORMANCE MATRIX USED IN EXPERIMENT**

| Matrix  | Equation          |
|---------|-------------------|
| Accuracy| \( \frac{TP + TN}{TP + FN + FP + TN} \) |
| Precision| \( \frac{TP + FP}{TP} \) |
| Recall  | \( \frac{TP + FN}{Precision + Recall} \) |
| F1-Score| \( 2 \times \frac{Precision \times Recall}{Precision + Recall} \) |

**TABLE 11**

**PARAMETERS OF 1D-CNN MODEL**

| Type       | Parameter  | Shape                  |
|------------|------------|------------------------|
| Input      | Filter: 32 | 72, 121, 179, 189, 362 |
|            | Kernel: 2  | 121, 32                |
|            | Stride: 1  | 189, 32                |
|            | L = 32, 32 | 362, 32                |
| Conv_layer1| Filter: 64 | 72, 32                 |
|            | Kernel: 2  | 121, 32                |
|            | Stride: 1  | 189, 32                |
|            | L = 32, 32 | 362, 32                |
| Leaky_ReLu| Alpha=0.3  | 36, 32                 |
| Max Pooling| Pool size : 2 | 89, 32             |
| Dropout    | Rate: 0.2  | 94, 32                 |
|            | Filter: 64 | 181, 32                |
| Conv_layer2| Filter: 64 | 89, 46                 |
|            | Kernel: 2  | 94, 46                 |
|            | Stride: 1  | 181, 46                |
| Leaky_ReLu| Alpha=0.3  | 18, 64                 |
| Max Pooling| Pool size : 2 | 44, 64             |
| Dropout    | Rate: 0.2  | 47, 64                 |
|            | Filter: 64 | 90, 64                 |
| Conv_layer3| Filter: 64 | 18, 128                |
|            | Kernel: 2  | 30, 128                |
|            | Stride: 1  | 44, 128                |
| Leaky_ReLu| Alpha=0.3  | 47, 128                |
|           |            | 90, 128                |

**TABLE 12**

**DETAILED RESULT: COMPUTATIONAL COMPLEXITY**

| Method            | NSL-KDD | CSE-CIC-IDS |
|-------------------|---------|-------------|
| Time/s Parameter  | Time/s Parameter |
| 1D-CNN(M)         | 180     | 13,243.9    |
| 1D-CNN(D)         | 229.6   | 36,903      |
| 1D-CNN(CD)        | 243.3   | 312,415     |
| M-1               | 62.5    | 6,385.1     |
| M-2               | 65.1    | 44,903      |
| D-1               | 79.3    | 6,415.9     |
| D-2               | 72.2    | 45,159      |
| CD-1              | 79.9    | 10898.1     |
| CD-2              | 82.9    | 63,719      |
| Flatten Input     | 5,7     | Softmax     |
| 1D-CNN(x)         |         |             |
| Time/s Parameter  |         |             |
| 1D-CNN(M)         | 39,973  | 36,903      |
| 1D-CNN(D)         | 48,933  | X           |
| 1D-CNN(CD)        | 50,853  | X           |
| M-1               | 43,877  | 44,903      |
| M-2               | 44,133  | 45,159      |
| D-1               | 47,077  | X           |
| D-2               | 47,333  | X           |
| CD-1              | 47,077  | 63,719      |
| CD-2              | 47,333  | 63,975      |

The experimental results from computational complexity are given in Table 12. In the case of 1D-CNN(x) of Method, x means M, D, and CD generated through the proposed preprocessing method. The difference here is that 1D-CNN models are trained without transforming the data into images for training. Looking at the experimental results, it can be seen that in the case of learning time/s, about 2-3 times the learning time is saved when the proposed method is used. For the parameters used for training, the M method, which combines basic pre-processing and image transformation, used more parameters than before. However, you can see that the number of parameters is reduced in the D and CD method, which combines discretization and image conversion. Through the above experimental results, it was confirmed that learning time/s and parameters can be reduced in CNN training combined with the proposed discretization and image conversion.
using the CD-1 data was the highest for all four average indicators. In addition, the CD-2 data, which were divided into grayscale and color, showed a similar performance. This is common with other pretreatment methods, and it confirmed that there was not much difference in performance for grayscale and color.

Table 13 lists the precision performance for each class and the percentage of data in which the predicted and actual values matched the correct answer in the predicted target. This corresponded to a false positive from the viewpoint of intrusion detection and was the ratio of classifying normal network traffic as attack traffic. Because false positives have a lower priority than false positives in intrusion detection, the false-negative performance was important. From Table 14, it can be seen that the results of each model were different for each class of precision. It can be seen that the R2L and U2R classes with a small number of training data samples in the NSL-KDD data set show the best performance in the models trained with D-2 and M-1 data. In the case of Infilter and Web Attack, which has a small number of training data samples in the CSE-CIC-IDS 2018 dataset, it can be seen that the model trained with CD-1 data shows the best performance. However, it can be seen that Web Attack is not classified at all in some models.

Table 15 lists the recall performances for the individual classes. The recall rate was the ratio predicted by the model as the correct answer among the actual correct answers and represented the ratio of falsely detecting an attack as normal with a false voice from the intrusion detection point of view. It is a concept that correlates with precision, but because the important indicators can vary depending on the viewpoint of the performance evaluation, higher values for both performance indicators indicate a better model. Looking at the recall performance by class of the NSL-KDD data set, it can be seen that R2L and U2R classes with small data
samples are not classified at all in some models. However, the CD-1 model and Siamese model showed good performance compared to other models at 69% and 50% for the R2L and U2R models, respectively. Similarly, for the CSE-CIC-IDS 2018 data set, it can be seen that in some models, web attacks with a small data sample are not classified at all. However, in the MLP and LSTM models, the Web Attack class showed high performance at 98%, and the CD-1 model showed the highest performance at 41% for the Infiler class.

Table 16 list the F1-score performance, which were evaluation indicators that could accurately evaluate when the class had an unbalanced structure. Even if there was a priority for precision and recall, if both indicators had a significant influence, it could be used as the most effective performance indicator as an evaluation value that was not biased to either side. In the end, because the precision and recall have a trade-off relationship, higher values for both indices, produce a higher F1-score. Looking at the performance of each class on the F1-score of the NSL-KDD data set, it can be seen that in the model using the CD-1 data, 4 classes except the U2R class show the best performance. In particular, it was confirmed that data with very few training data samples, such as the Infiler class, showed good performance at 58%. However, the model using CD-1 data did not show optimal performance in the case of the Web Attack class because it did not show good performance in Recall.

| Data set   | Method | Normal | DoS | Probe | R2L | U2R |
|------------|--------|--------|-----|-------|-----|-----|
| NSL KDD    | M-1    | 95%    | 77% | 59%   | 10% | 3%  |
|            | M-2    | 96%    | 80% | 58%   | 10% | 4%  |
|            | D-1    | 92%    | 80% | 67%   | 33% | 12% |
|            | D-2    | 92%    | 79% | 71%   | 38% | 12% |
|            | CD-1   | 94%    | 86% | 75%   | 69% | 10% |
|            | CD-2   | 91%    | 88% | 68%   | 63% | 16% |
|            | DNN [33]| 99%    | 73% | 49%   | 0%  | 7%  |
|            | Siamese[34] | 89%    | 86% | 77%   | 32% | 50% |

| Data set    | Method | Class | M-1 | M-2 | CD-1 | CD-2 | MLP [35] | LSTM [35] |
|-------------|--------|-------|-----|-----|------|------|---------|-----------|
| CSE CIC IDS 2018 | Normal | 100% | 100% | 100% | 100% | 100% | 99%     | 99%       |
|              | Botnet | 1%   | 1%   | 100% | 100% | 100% | 100%    | 100%      |
|              | Brute Force | 100% | 100% | 98%  | 100% | 94%  | 98%     | 98%       |
|              | DDoS   | 13%  | 13%  | 100% | 100% | 100% | 100%    | 100%      |
|              | DoS    | 71%  | 70%  | 81%  | 79%  | 66%  | 99%     | 99%       |
|              | Infiler | 12%  | 5%   | 41%  | 18%  | 19%  | 17%     | 17%       |
|              | Web Attack | 0%   | 0%   | 35%  | 47%  | 98%  | 98%     | 98%       |

| Data set    | Method | Class | M-1 | M-2 | CD-1 | CD-2 | MLP [35] | LSTM [35] |
|-------------|--------|-------|-----|-----|------|------|---------|-----------|
|              | Normal | 94%  | 94%  | 99%  | 99%  | 93%  | 93%     | 93%       |
|              | Botnet | 3%   | 3%   | 99%  | 100% | %    | 99%     | 99%       |
|              | Brute Force | 75%  | 75%  | 85%  | 84%  | 88%  | 95%     | 95%       |
|              | DDoS   | 23%  | 23%  | 100% | 100% | 100% | %       | 100%      |
|              | DoS    | 83%  | 82%  | 89%  | 88%  | 76%  | 98%     | 98%       |
|              | Infiler | 24%  | 7%   | 58%  | 24%  | 31%  | 28%     | 28%       |
|              | Web Attack | 0%   | 0%   | 51%  | 64%  | 84%  | 58%     | 58%       |

These results of the experiment confirmed that the M-1, M-2, D-1, D-2, CD-1, and CD-2 algorithms, which converted data into images and processed them, improved the performance compared to the algorithms that are generally used to learn network traffic. Among them, the CD-1 and CD-2 algorithms, which applied discrete preprocessing to continuous and discrete variables, showed good performance in all data. For the NSL-KDD data set, the F1-score values for the R2L class were increased to 71% and 70%, respectively, in the model trained using the CD-1 and CD-2 data, respectively, in the CSE-CIC-IDS 2018 data set increased the Infiler class to 58% in the model using CD-1 data. This is because the nonlinear relationship between continuous and discrete variables for existing sparse classes was converted into a linear relationship that is easy to interpret by the model through discretization preprocessing. In addition, in the case of the NSL-KDD dataset F1-score performance, it was a number that was affected by the precision and recall, and it could be seen that a performance difference occurred as a result of changes in the two performance indicators, which had a complementary relationship with each other. Looking at the feature of each performance indicator described in section 5.B, the precision
and recall had a complementary relationship. Thus, as the recall performance of CD-1 and CD-2 improved compared to M-1 and M-2, the precision performance decreased. However, in the case of the U2R class of the NSL-KDD data set with the smallest data sample and the Web Attack of the CSE-CIC-IDS 2018 data set, it can be seen that the performance improvement using the proposed method and the performance increase/decrease ratio according to the complementarity relationship are abnormal.

VI. CONCLUSION
In this study, a Discrete Pre-processing Method and NIDS using a Convolution Neural Network were proposed to solve the problems occurring in the learning-based NIDS. Contributions from this study include:

1) Use the NSL-KDD and CSE-CIC-IDS 2018 data sets composed of unbalanced data that occurs similarly to the real situation.

2) By using discretization and 2D image vector transformation for continuous variables in the preprocessing process, data consisting of mixed features were converted into data that can easily analyze the relationships between various feature relationships.

3) Using the least studied CNN-based NIDS model, we evaluated the generalization performance of the proposed preprocessing method.

For the designed model, NSL-KDD and CSE-CIC-IDS 2018 data, which is data representing network traffic, was used for training and evaluation. To evaluate the generalization performance of the proposed model, four indicators (accuracy, precision, recall, and f1-score) that are frequently used in machine learning and deep learning evaluation were used. To evaluate the computational complexity, we compared the learning time and trainable parameters of the CNN model similar to the proposed model. The results of the computational complexity experiment showed that the CNN model trained using D and CD through the preprocessing proposed in this study saves 2~3 times the learning time and reduces the trainable parameters.

As a result of the performance experiment, it was found that the CNN model trained using the CD-1 data generated through the preprocessing proposed in this study was the best overall. In particular, for the R2L class of the NSL-KDD dataset with a small training sample, the f1-score value was 71%, which was superior to other studies, and the InfILTER class of the CSE-CIC-IDS 2018 dataset showed a performance of 58%. This is because the model has been transformed into a linear relationship that is easy to learn through the proposed preprocessing method for properties with a non-linear relationship to a sparse class. However, it was also confirmed that the U2R class of the NSL-KDD data set and the Web Attack of the CSE-CIC-IDS 2018 data set did not show as much improvement as reported in previous studies. In the case of the Siamese model, compared with the proposed paper, it can be seen that, despite being an image-based model, it performs well for class imbalance for NSL-KDD data composed of tabular data. Also, in the case of MLP and LSTM models, it can be seen that the Web Attacker class of CSE-CIC-IDS 2018 data shows high performance compared to the proposed paper.

Therefore, future research is expected to show better results in class imbalance by using the Siamese model and time series learning method for the image data generated through the preprocessing method of the proposed paper.

REFERENCES

[1] Raghunath, and S. N. Mahadeo, “Network intrusion detection system (NIDS).”, 2008 First International Conference on Emerging Trends in Engineering and Technology. IEEE, pp. 1272-1277, 2008, https://doi.org/10.1109/icetc.2008.252

[2] Hurley Trace, E. Perdomo Jorge and Alexander Perez-pons, “HMM-Based Intrusion Detection System for software-defined networking”, 2016 15th IEEE Conference on Machine Learning and Application, pp. 617-621, 2016, https://doi.org/10.1109/ICMLA.2016.0108

[3] R. Patan, G. P. Ghantasala, R. Sekaran, D. Gupta and M. Ramachandran, “Smart healthcare and quality of service in IoT using grey filter convolutional based cyber physical system. “, Sustainable Cities and Society, vol. 59, 2020, https://dx.doi.org/10.1016/j.scs.2020.102141.

[4] G. Aceto et al., “Mobile encrypted traffic classification using deep learning: Experimental evaluation lessons learned and challenges”, IEEE Trans. Netw. Service Manag., vol. 16, no. 2, pp. 445-458, Jun. 2019, https://doi.org/10.1109/TNSM.2019.2890985

[5] Y. Cheong, K. Park, H. Kim, J. Kim and S. Hyun, “Machine learning based intrusion detection systems for class imbalanced datasets”, J. Korea Inst. Inf. Secur. Cryptol., vol. 27, no. 6, pp. 1385-1395, 2017, https://doi.org/10.1080/09391178.2017.12.66038

[6] I. Ahmad, M. Basheer, M. J. Iqbal and A. Rahim, “Performance comparison of support vector machine random forest and extreme learning machine for intrusion detection”, IEEE Access, vol. 6, pp. 33780-33795, 2018, https://doi.org/10.1109/ACCESS.2018.2841987

[7] Y. Sun and F. Liu, “SMOTE-NCL: A re-sampling method with filter for network intrusion detection”, Proc. 2nd IEEE Int. Conf. Comput. Commun. (ICCC), pp. 1157-1161, 2016, http://doi.org/10.1109/CompComm.2016.7924886

[8] B. Yan, G. Han, M. Sun and S. Ye, “A novel region adaptive SMOTE algorithm for intrusion detection on imbalanced problem”, Proc. 3rd IEEE Int. Conf. Comput. Commun. (ICCC), pp. 1281-1286, 2017, http://doi.org/10.1109/CompComm.2017.8327479

[9] S. Garcia, J. Luengo and F. Herrera, “Tutorial on practical tips of the most influential data preprocessing algorithms in data mining”, Knowl. Based Syst., vol. 98, pp. 1-29, 2016, https://doi.org/10.1016/j.knosys.2015.12.06

[10] E. Hodo et al., “Shallow and deep networks intrusion detection system- a taxonomy and survey”, Journal of Cryptography and Security, 2017, arXiv:1701.02145

[11] K. Wu, Z. Chen and W. Li, “A novel intrusion detection model for a massive network using convolutional neural networks”, IEEE Access, vol. 6, pp. 50850-50859, 2018, http://doi.org/10.1109/ACCESS.2018.2868993

[12] Hu, Jianwei, Chenshuo Liu, and Yanpeng Cui, “An Improved CNN taxonomy and survey”, Journal of Cryptography and Security, 2017, vol. 27, no. 6, pp. 1385-1395, 2017, https://doi.org/10.1007/s10805-017-0575-y

[13] P. Laskov, P. Düssel, C. Schäfer, K. Rieck, “Learning intrusion detection: supervised or unsupervised?”, IEEE Conference on Machine Learning and Application, pp. 617-621, 2016, http://doi.org/10.1109/ICMLA.2016.0108

[14] Y. Guan, A. A. Ghorbani and N. Belace, “Y-means: a clustering method for intrusion detection”, CCECE 2003-Canadian Conference on Electrical and Computer Engineering. Toward a Caring and Humane Technology, IEEE, vol. 2, pp. 1083–1086, 2003, https://doi.org/10.1109/CCECE.2003.1226084
