IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 9, 2022, accepted March 21, 2022, date of publication April 1, 2022, date of current version April 8, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3164104

Multi-Class Intrusion Detection Using Two-Channel Color Mapping in IEEE 802.11 Wireless Network

MUHAMAD ERZA AMINANTO[1,2], (Member, IEEE), R. SATRIO HARIOMURTI WICAKSONO[3], ACHMAD ERIZA AMINANTO[1], HARRY CHANDRA TANUWIDJAJA[2], LIN YOLA[1], AND KWANGJO KIM[4,5], (Senior Member, IEEE)

[1]Graduate School of Strategic and Global Studies, University of Indonesia, Jakarta 16424, Indonesia
[2]National Institute of Information and Communication Technology, Tokyo 184-8795, Japan
[3]Faculty of Science and Technology, Universitas Al-Azhar Indonesia, Jakarta 12110, Indonesia
[4]School of Computing, Korea Advanced Institute of Science and Technology, Daejeon 34141, South Korea
[5]International Research Institute for Cyber Security, Seongnam 13591, South Korea

Corresponding author: Muhamad Erza Aminanto (erza.aminanto@ui.ac.id)

This work was supported by the Innovations, Technology and Social Changes Group of School of Strategic and Global Studies, Universitas Indonesia, under a contract of 2021 Research Cluster Grant.

ABSTRACT The rise of interconnected devices through wireless networks provides two sides consequences. On one side, it helps many human tasks; on the other hand, the prone wireless medium opens the vulnerable system to be exploited by adversaries. An Intrusion Detection System (IDS) is one method to inspect the network traffic by leveraging state-of-the-art anomaly detection techniques. Deep learning models have been utilized to distinguish the benign and malicious traffic. However, projecting the tabular data into images before the image classification has been the main challenge of leveraging deep learning for IDS purposes. We propose the novel projection of tabular data into 2-coded color mapping for IDS purposes. The proposed method employs a feature selection method to ensure optimal dimensionality. We examined the different number of attribute subsets to obtain the relationship between the attributes. Furthermore, it takes advantage of the Convolutional Neural Network (CNN) model to classify the Wi-Fi attacks. We evaluate the proposed model using the most common Wi-Fi attacks dataset, Aegean Wi-Fi Intrusion Dataset (AWID2). The proposed method achieved an F1 score of 99.73% and a false positive rate of 0.24%. This study highlights the importance of addressing the mapping procedures from tabular data into grid-based data before deep learning training and validates the effectiveness of CNN to detect multiple types of wireless network attacks.

INDEX TERMS Wireless attacks, intrusion detection system, convolutional neural network, anomaly detection.

I. INTRODUCTION

Lately, the Internet of Things (IoT) has developed very fast [1]. One particular characteristic of IoT devices is widely interconnected through a wireless network with limited power and storage resources [2]. That being the case, information security issues have become more prominent and need to be addressed seriously. As the number of Internet users grows, not all people have decent digital literacy knowledge; adversaries have identified this phenomenon. Adversaries have developed several attacks over wireless networks. Lightweight devices such as fridges, routers, smart doors, and webcams are vulnerable to security threats [3]. An Intrusion Detection System (IDS) provides one of the efficient systems to ensure the confidentiality, integrity, and availability of an Internet network by proactively inspecting the network traffic [4]. Furthermore, the emergence of Deep Learning (DL) allows a DL-based IDS as a promising solution to Internet security problems [5].

IDS can be classified into two classes: signature-based and anomaly-based. Signature-based IDS uses the attack signature to detect a particular attack that matches the pattern. Antivirus developers initially used this methodology. A list of Indicators of Compromise (IOCs), which is kept updated

The associate editor coordinating the review of this manuscript and approving it for publication was Baozhen Yao.

from time to time, is required to ensure detection accuracy. However, signature-based IDS is vulnerable to unknown attacks. Slight modification on the attack signature or traffic encryption is sufficient to bypass this IDS type. On the other hand, behavior-based IDS monitors inbound network traffic to check any suspicious behavior. Behavior-based IDS increases the likelihood of unknown attack mitigation, while signature-based IDS identifies particular attack signatures. The behavior-based IDS usually leverages state-of-the-art artificial intelligence models to distinguish between benign and malicious traffic.

Popular deep learning classifiers, such as Decision Tree (DT), Random Forest (RF), Support Vector Machine (SVM), and Naïve Bayes (NB), have been utilized by several IDS research [6]. DT organizes data in a tree structure that splits data based on the feature value [7]. RF is a group of decision trees that choose features randomly during the classification process [8]. The randomized feature selection can generalize the data and prevent overfitting. SVM classifies data by dividing them using hyper-planes as the boundaries [9]. Finally, the naïve Bayes method is based on the Bayes theorem that computes the probability of defined classes on a dataset [10].

However, the methods listed above have similar limitations. They cannot perform well on a complex dataset with high dimensionality, especially since most IDS datasets are tabular. Because of this reason, we want to propose a deep learning framework for IDS that can cope with the complex dataset. Recursive Feature Elimination and Cross-Validation Selection (RFECV) is a wrapper-type feature selection that recursively removes irrelevant features based on the validation scores [11]. The RFECV algorithm aims to reduce the features on a high-dimensional dataset, providing a solution to address overfitting, computation time, and learning accuracy. Residual Networks-50 (ResNet50) [12] is a variant of the ResNet model that has 48 convolutional layers, 1 max-pooling layer, and 1 average pool layer [13]. This framework can control the error rate of ultra-deep neural networks by introducing the deep residual learning framework.

This paper proposes an IDS framework capable of handling tabular IDS datasets using an image classification approach. One noticeable challenge is converting the tabular data to an image to be fed into the Convolutional Neural Network (CNN) classifier. The framework consists of three main parts: 1. Feature selection using RFECV-SVM, 2. Image projection to convert the data into RGB format, and 3. CNN model training & testing using Resnet50. In this study, we examine the most common Wi-Fi network attacks dataset, namely Aegean Wi-Fi Intrusion Dataset (AWID2) [14]. To the best of our knowledge, this paper is the first DL-based IDS that projects IDS data into 2-color coded mapping with the combination of RFECV-SVM and Resnet50 CNN using the AWID2 dataset. Compared to previous work, our main contributions are listed below:

- We propose a CNN-based IDS that can handle the dataset with a minimum trade-off between the number of features, instances, and performances.
- Our framework exploits the spatial relationship through convolution methods which tabular data cannot do.
- We inspect the correlation between features inside matrices to capture the underlying spatial feature.
- Our proposed system performs feature selection on tabular data, converts it into an image, and feeds the converted image into Resnet50, the first DL-based IDS that utilizes 2-coded color mapping for the tabular IDS dataset.
- We provide performance analysis by highlighting the F1 score of our system. On a vast imbalance dataset, the F1 score represents our model's performance more accurately than the accuracy metric.

The rest of this study is organized as follows: Previous studies on reducing threat alerts are discussed in Section II. Section III overviews the dataset used in this study. The process of data preparation and the proposed method are explained in Section IV. The experimental results are presented in Section V. Section VI compares the proposed method with other machine learning models. Finally, Section VII concludes the paper and outlines directions for future work.

II. RELATED WORK

IDS-related studies have been done for relatively a long time. Detecting and classifying the type of attacks, especially on wireless networks, required considering many aspects and factors. These aspects could be interpreted as different attributes in respect to each record. In this research, we used Wi-Fi intrusion dataset, AWID2, built by Kolias *et al*. in 2015 [14]. With 154 attributes for each record, the AWID2 dataset could be considered a high-dimensional dataset. In previous research, AWID2 was used to solve a binary classification problem or multiclass classification problem based on attack labels. Thing [15] proposed a Deep Learning model with 98.66% accuracy on a 4-class classification problem using 154 attributes. Another approach on 4-class classification was made by Kasongo and Sun [16] with Feed-Forward Deep Neural Network (FFDNN) model and obtained a 99.77% accuracy score. Unlike Kolias, FFDNN proposed by Kosongo *et al.* used only 26 out of 154 attributes. Kosongo *et al*. used this model to classify the AWID2 dataset as a binary problem and obtained 99.66% accuracy. Regarding the binary classification problem, Aminanto *et al.* [17] propose the Deep-Feature Extraction and Selection (D-FES) method with better performance, 99.97% accuracy, and 99.94% F1 score.

Problems related to high-dimensional data are usually associated with many deep learning architectures. One of many architectures used in deep learning implementation is CNN, which is known for an effective technique to find and learn patterns in grid-like topology data [18]. Hence CNN is commonly used to solve computer vision tasks. Residual Neural Network (ResNet) variance, Proposed by He *et al*. [12] in 2016, is the most frequently used architecture as a base network [19]. A recent study done by

Wicaksono *et al.* [20] used ResNet50 as a base network to compare multiple contrastive learning methods. On the other hand, Firat and Hanbay [21] also proposes the usage of ResNet50 to tackle the different problems related to hyperspectral images. The superiority of ResNet50 compared to other Deep Learning architectures is shown in a study done by Septiandri *et al.* [22]. This study suggests that ResNet50 performs better than other architectures such as DenseNet121, DenseNet169, MobileNetV2, Xception. Also, ResNet50 outperforms other ResNet variants such as ResNet18, ResNet34, and ResNet101 in medical images.

As mentioned before, CNN is used to process grid-like topology data [18]. It suggests that converting IDS Dataset into spatial-related format is a prerequisite before implementing CNN to the dataset. Duan *et al.* [23] mentioned in their study that data conversion is a crucial part of their research using the AWID2 dataset to train a classifier model. Al-Turaiki and Altwaijry *et al.* [24] also converted the IDS data format into a more suitable format for CNN. They projected the NSL-KDD and UNSW data into 2D images encoded in grayscale. Each projected image contains small squares with a specific grayscale value mapped from the attribute value in the preprocessed dataset. Sun *et al.* [25] suggested another conversion method in their study. The proposed method is 2D word embedding that maps attributes into literal writing in an image. Even though Sun *et al.* [25] do not use IDS-related datasets, their conversion method is still compatible with CNN architecture such as ResNet, SE-Net, and PolyNet.

Our research adopted the preprocessing method proposed by Al-Turaiki and Altwaijry *et al.* [24] and enhanced it by doubling the range of conversion value. In the original idea proposed by Al-Turaiki and Altwaijry *et al.* [24], attributes were converted into grayscale, between 0 to 255 [26]. These grayscale values were used to generate small square images and placed sequentially. In our case, we utilized RGB channels to convert each attribute. We adjusted the ''Red'' and ''Blue'' channels accordingly to attribute value. Since each channel has a range value from 0 to 255, this approach allows us to utilize twice as much range compared to grayscale. The wider range also allows us to generate images with high color contrast for better representation of multiple attributes with adjacent values. For better image representation, we keep the number of total attributes equal to the square number, representing the NxN dimension of the image. Therefore, we also implemented feature selection using RFECV-SVM to get the square number of total attributes based on returned ranking. As for the network, we used ResNet50 as our base network.

III. DATA DESCRIPTION

In this study, we chose the IEEE 802.11 wireless network intrusion dataset, called AWID2 [14], because it represents the latest IEEE 802.11 network with simulated attacks in a real network. The dataset emulated the physical environment containing ten clients and one node as an adversary.

TABLE 1. Class composition on the original dataset.

Class [represented as:]	Training	Testing
Normal: "0"	1,633,190	530,785
Impersonation: "1"	48,522	20,079
Injection: "2"	65,379	16,682
Flooding: "3"	48,484	8,097
Total Original Dataset	1,795,575	575,643

All devices were connected through an access point protected with Wired Equivalent Privacy (WEP) encryption.

The dataset used in this study comprises four classes as shown in Table 1. First is the ''normal'' class, represented as class 0 during our experiment. This class indicates a network without any attack and dominates the entire data by ratio 10:1 compared to others combined. Other types corresponding to three major attack types are ''impersonation,'' ''injection,'' and ''flooding.'' Those classes were represented as class 1, class 2, and class 3 sequentially during our experiment.

There are four different attacks that belong to Impersonation class, namely Caffe Latte [27], Hirte [28], Evil Twin [29] and Rogue Access Point (AP) [30]. Caffe Latte and Hirte attacks are launched remotely by stealing the WEP key. Clients usually keep their list of previously connected networks for seamless connection in the future. Then the adversary will capture the WEP key in the middle of the handshake protocol. The difference between Caffe Latte and Hirte is that the latest attack uses a different approach, which is a fragmentation attack. Meanwhile, both Evil Twin and Rogue AP attacks exploit the behavior of the victims to look for an already connected network nearby by name. The attacker masquerades as the legal AP to the victim.

Injection group consists of three attacks: ARP Injection [31], Chop-Chop [32] and Fragmentation [33]. The ARP Injection manipulates the network to generate the Initialization Vector (IV) in large numbers to be captured for the key cracking purposes. The chop-chop attack can reveal m last bytes in the keystream so that the adversary can deduce the cipher without knowing the key. The fragmentation attack exploits the 802.11 protocol which any packet exceeding the defined maximum length should be fragmented into smaller packets and can be delivered independently.

The last group is Flooding attacks, which comprise at least 13 different attacks such as Beacon Flooding [34], Request to Send (RTS) Flooding [35], and Fake Power Saving [36]. The Beacon Flooding attack will transmit fake beacons with either non-existing or particular spoofed identifiers. The transmission will cause overflow to the clients. The RTS Flooding attack exploits RTS/CTS (Request to Send/Clear to Send) protocol. The adversary will transmit many RTS frames with a long duration window to keep the clients busy. Finally, the Fake Power Saving attack abuses the power-saving protocol, where the clients are enforced to be in sleep mode, then the access points ignore them.

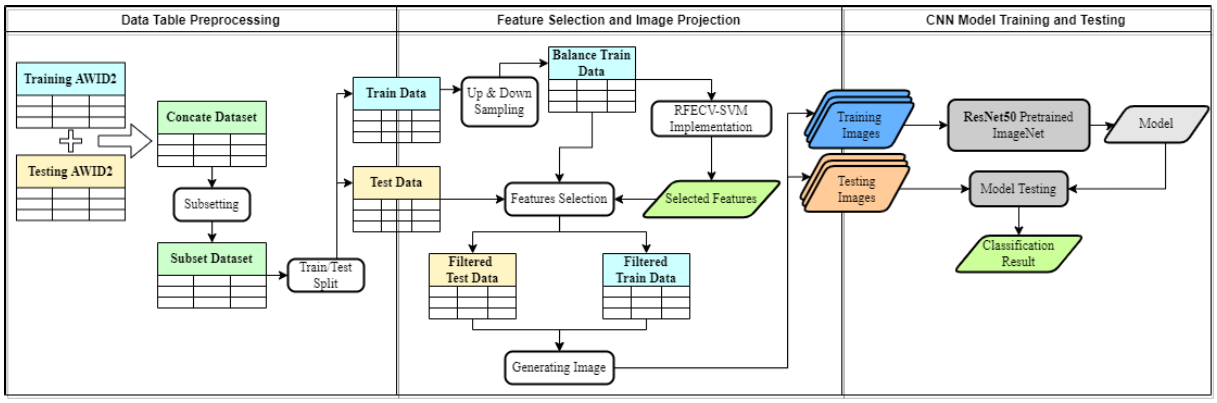


FIGURE 1. Overview of the proposed method with three stages procedures.

IV. METHODOLOGY

In this section, we explain the process of data preparation and data pre-processing. Then the process is followed by the feature selection step and classification using CNN.

Our methodology could be defined as three sequences as depicted in Fig. 1. The first sequence is Data Table Preparation, in which we process the normalized AWID2 datasets. Next, we combined the normalized training and testing datasets into one massive dataset containing 4-class data to reduce bias in the following sequence. We only utilized 0.5% of the entire combined data. The subsetting process is necessary to avoid high memory usage during generating images and training a model. Then, the subset data was split into a new train and test data table.

Since we use AWID-CLS-R 4-classes data, it is required to balance the class distribution of the training dataset by upsampling and downsampling to avoid overfitting. We used RFECV feature selection to extract a square number of attributes as we were required to project the selected attributes into images with NxN dimension. RFECV itself is classified as an exhausted algorithm but considerably straightforward. RFECV algorithm will run a model using a defined estimator and all the features in the given input data. Then, it validates the result and removes the less important feature based on validation. As the name suggests, RFECV will take the new subset with removed features, retrain, and revalidate until the model defines all features based on their importance. RFECV using 5-fold cross-validation and SVM estimator were then implemented into balanced distributed train data to estimate the best-fit selected feature to represent the train data. We need to mention that the RFECV fitting duration is proportional to the input data, so we used 1,000 data points sampled from balanced train data for the RFECV fitting to keep processing time to a minimum. For better understanding and comparison, we took the square values of 25, 36, 49, 64, 91, 100, 121, and 144 to represent the total attributes used in the image out of 154 attributes. As mentioned before, we chose these values to keep small squares evenly plotted in the generated image with N x N dimension. By default, RFECV did not return the constant number of selected features since it depended on random state and input data used during the process. These selected features are regarded as first rank or first attributes. The unselected attributes returned with the features rank that corresponds to the ranking position of the "i-th" attributes. We took the square value total attributes by slicing the sorted attributes based on their ranking.

We used the selected attributes to filter the train and test data. These filtered train and test data tables generated images for the CNN process. Train and test data tables contain the attribute values in 0 to 1. We multiply these values by 510 since we use two channels of RGB instead of three. The multiplication results greater than 255 converted to the Red channel. Simultaneously, multiplication results equal or less than 255 converted to the Blue channel. We then generate square images based on these values and place these images left-right and top-down based on their attributes ranking to form a new bigger image. Finally, the generated images were grouped based on their model training and testing labels.

CNN is a neural network that should have at least one convolutional layer. CNN is highly correlated to three principles. The first is sparse interaction, which means that the nodes in CNN are not always fully-connected because of the presence of kernel. The second is parameter sharing, which means that several parameters are used together in a single model. The third is equivariant, which means that the change in input data is highly reflected on the output since some parameters are shared [18]. ResNet50 is one of many types of CNN architecture. By default, ResNet50 consists of 50 hidden layers with 1,000 end-output on a fully-connected layer. During the experiment, we replaced the fully-connected layer with another sequential layer consisting of the Softmax activation layer at the end to map the distribution probability of outputs into four classes. The overview of these layers can be seen in Fig. 2.

CNN processes usually use three types of data: training, validation, and testing datasets. We made these datasets using generated images from the previous sequence. Our datasets

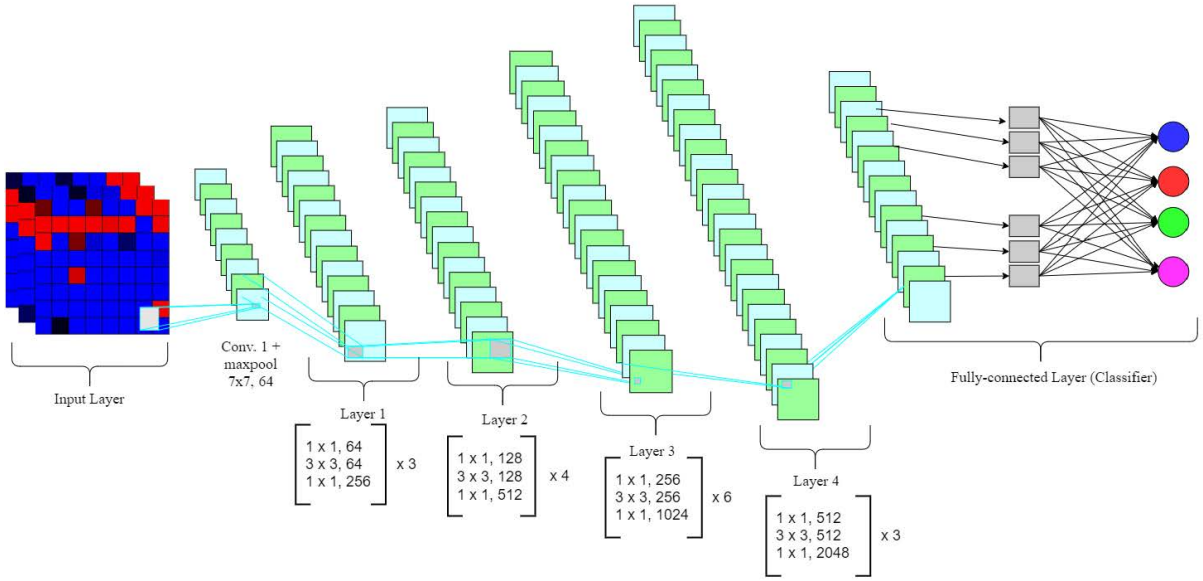


FIGURE 2. Overview of the ResNet50 architecture.

consist of 6,669 images of the training dataset, 741 images of the validation dataset, and 2,223 images of the testing dataset. We used ResNet50 as our base network and Stochastic Gradient Descent (SGD) as an optimizer with eight epochs, a learning rate of 0.005, and a momentum of 0.9. We used One Cycle LR with a 0.005 maximal learning rate as our scheduler. For better observation of model robustness, we repeat the model training and testing using the same dataset three times and then calculate the mean value and standard deviation of each model for three iterations.

V. EVALUATION

We employed the AWID2 dataset containing more than two million Wi-fi network data collection and classified this dataset into four classes. We concerted eight configurations based on the square values used to filter the number of attributes. The pipeline sequences implemented towards each configuration are identical. Our experiment was entirely done using Python, supported by Pytorch as the framework for better tuning and adjustment during CNN model training and testing. We run our code on an Nvidia Tesla K80 Graphic Card using 12 GB of RAM. The complete source code for this study is available online.[1]

A. DATA PREPROCESSING

Before data-to-images projection and CNN model training, it is necessary to adjust the existing dataset. We normalized the entire dataset between values zero and one using the mean range method. The value normalization is essential during value conversion in data-to-image projection and CNN model training to reduce bias.

[1]https://github.com/satrio-hw/IDS_research_CNN

The dataset that we used has 2,371,218 records of Wi-fi network data, divided into a training and testing set with a ratio of 75:25. As mentioned in the table, the class ratio between the training and testing dataset is quite different. We combine training and testing datasets into one massive dataset to reassure data ratio consistency. As we need to convert the records into images, this amount of data costs high memory usage and extensive time to process. Therefore, we reduced the data by sampling 0.5% of the original amount to overcome this high cost. We kept a similar distribution of subset to the original dataset. The reduction result is 11,856 records, which split into a new train and test dataset with a ratio of 75:25.

As the "normal" class dominated 10-to-1 compared to other classes, it is necessary to balance the training dataset using upsampling and downsampling. A balanced training dataset lowered the possibility of model overfitting during model training. We projected each record from both balance training and imbalance testing dataset into an image using the RGB channel.

Our proposed methodology aims to capture a pattern formed by data-to-image projection and produces a model with a high understanding of these patterns. We choose RGB channels to project the data value into images as it is visually discernible to the human eye. By definition, the RGB channel is an array with three values, each ranging between zero and two hundred fifty-five. Each value represents the colors red, green, and blue. We only used red and blue channels to produce images with higher contrast. This approach means that we doubled the range of one color channel to project a red-black-blue color gradient. In addition, we implemented a rounding function, as shown in Algorithm 1, based on the number of attributes used during the projection to increase the

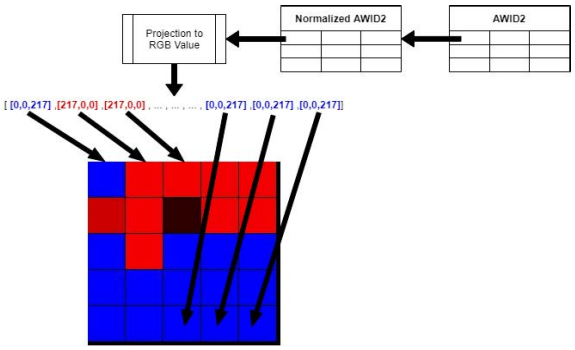


FIGURE 3. Image Projection from given attributes value. We implement value down rounding in projection function to increase the contrast between the produced small squares (see also algorithm on Table. 2).

TABLE 2. Data-to-image algorithm.

Algorithm 1 Projection to RGB Value Function

function Projection to RGB Value ($DataTable$)
$divider$ = 510 / $TotalAttr$
foreach row in $DataTable$:
  $RowNum$ = 0
  $ColNum$ = 0
  foreach $value$ in row:
    $MultiResult$ = (rounddown(($value \times 510$) / $divider$))
    $MultiResult$ = $MultiResult \times divider$
    if ($value \times 510$) < 255:
      $PixelVal$ = $[0, 0, 255 - MultiResult]$
    else:
      $PixelVal$ = $[MultiResult - 255, 0, 0]$
    endif
    SmallSquareImage = GenerateImage($PixelVal$)
    position[$RowNum$, $ColNum$] = SmallSquareImage
    if $ColNum$ == sqrt($TotalAttr$):
      $ColNum$ = 0
      $RowNum$ += 1
    else:
      $ColNum$ += 1
    endif
  endforeach
endforeach

contrast between adjacent values. Fig. 3 shows visualization related to our projection method to convert tabular data into an image.

The CNN training and testing were separated into two sequences. The first sequence is the initial training and testing using the combined train and test sets. The second sequence is a stress test, using a data-to-image projection of only the data testing subset. This stress test assures the model robustness and better performance evaluation between multiple models produced. We used a 69:8:23 ratio for the train-validation-test data images in the first sequence. We then generated 15,000 new images from the subset test dataset regarding each configuration and used the produced model from the first sequence to classify these new images.

B. EVALUATION METRICS

The imbalanced data distribution on multi-class datasets became our primary consideration for a fair evaluation of each model's performance. We used Accuracy (Acc) as an additional metric in our evaluation since it gives the overall insight into the correct prediction. Unfortunately, the correct prediction is not enough to measure the model performance since we have an extremely imbalanced testing dataset. Accuracy as a metric is unable to capture the minority class and lean toward the dominant class. As this problem occurred, we focused on the F1 score (F1) as our primary metric since it gives a better perspective on how well model prediction is based on precision and recall. Precision is a metric that calculates the true positive prediction against the total predicted positive (True Positive + False positive). Recall, also known as Detection Rate, calculates the true positive prediction against the total actual positive(True Positive + False Negative).

Since we used a multi-class dataset, a direct F1 score and accuracy will not be sufficient since they measure binary problems. Due to this, True Positive, True Negative, False Positive, and False Negative definitions heavily depend on the reflected class during the metrics calculation. Therefore, we used the weighted F1 score and weighted accuracy to calculate metrics for each label and found their average weighted by the number of samples from the corresponding class.

We also measure False Alarm Rate(FAR), which showed the number of a particular instance classified as another class, and False Negative Rate(FNR), which is the number of false-negative cases divided by the total number of actual positive instances. The smaller FAR and FNR values reflect better model performance. The following equations can define the above metrics.

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$Acc_{weighted} = \frac{\sum_{i=0}^{n} freq_{cls_i} \cdot Acc_{cls_i}}{\sum_{i=0}^{n} freq_{cls_i}} \tag{2}$$

$$F_1 = \frac{2TP}{2TP + FP + FN} \tag{3}$$

$$F1_{weighted} = \frac{\sum_{i=0}^{n} freq_{cls_i} \cdot F1_{cls_i}}{\sum_{i=0}^{n} freq_{cls_i}} \tag{4}$$

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

$$FAR = \frac{FP}{TN + FP} \tag{7}$$

$$FNR = \frac{FN}{FN + TP} \tag{8}$$

C. EXPERIMENTAL RESULTS

We evaluated our experiment based on each sequence from preprocessing to stress test. As we used eight configurations, each evaluation contained eight different values to compare.

TABLE 3. RFECV selected attributes per configuration.

Conf.	Square Value	Features Selected
I	25	[5, 7, 8, 46, 50, 63, 65, 66, 67, 69, 71, 72, 75, 77, 89, 90, 93, 97, 98, 101, 103, 108, 111, 117, 121]
II	36	[3, 4, 5, 6, 7, 8, 15, 28, 37, 46, 49, 50, 51, 60, 61, 63, 65, 66, 67, 68, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 81, 89, 90, 92, 93, 95]
III	49	[6, 7, 8, 37, 46, 49, 50, 63, 65, 66, 67, 69, 72, **74**, 75, 77, 79, 80, 89, 90, 91, 92, 93, **94**, 95, **96**, 97, 98, 99, 103, 106, 107, 111, 117, 118, 119, 120, 121, 124, **129**, 138, 139, 140, 141, 142, 143, 144, 145, 153]
IV	64	[3, 4, 5, 6, 7, 8, 13, 14, 15, 16, 17, 19, 37, 42, 46, 47, 49, 50, 60, 63, 65, 66, 67, 68, 69, 70, 71, 72, 74, 75, 76, 77, 78,79, 80, 81, 89, 90, 91, 92, 93, 94, 95, 96, 97, 99, 100, 101, 103, 105, 106, 107, 108, 109, 111, 117, 118, 119, 120, 121, 122, 125, 126, 127]
V	81	[3, 4, 5, 6, 7, 8, 9, 13, 14, 15, 16, 17, 19, 25, 28, 37, 42, 46, 47, 49, 50, 60, 61, 63, 65, 66, 67, 68, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 81, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 103, **104**, 106, 107, 108, 109, 111, 117, 118, 119, 120, 121, 122, 125, 126, 127, 128, 129, 137, 138, 139, 140, 141, 142, 143, 144, 145, 153]
VI	100	[3, 4, 5, 6, 7, 8, **9**, **10**, **11**, **12**, **13**, 14, 15, 16, 17, **18**, **19**, **20**, **21**, **24**, 25, 28, 37, **42**, 46, **47**, 49, 50, **51**, **54**, **55**, 60, **61**, 63, **64**, 65, 66, 67, 68, 69, 70, 72, **73**, 74, 75, 76, 77, 78, 79, 80, 81, **88**, 89, 90, 91, 92, 93, 95, 96, 97, 98, 99, 100, 101, 103, 105, 106, 107, 108, 109, 111, **113**, **114**, **115**, **116**, 117, 118, 119, 120, 121, 122, 125, 126, 127, 128, 129, **133**, **134**, 137, 138, 139, 140, 141, 142, 143, 144, 145, **149**, **150**, 153]
VII	121	[3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, **26**, **27**, 28, 29, 30, 31, 37, 42, 45, 46, 47, 48, 49, 50, 51, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 97, 100, 101, 103, 106, 107, 108, 109, 110, 111, **112**, 117, 118, 119, 120, 121, 122, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, **147**, **148**, 149, 150, 151, 153]
VIII	141	[3, 4, 5, 6, 7, 8, **9**, **10**, **11**, **12**, 13, 14, 15, 16, 17, **18**, 19, **20**, **21**, **22**, **23**, **24**, 25, **26**, **27**, 28, **29**, **30**, **31**, **32**, **33**, **34**, **35**, **36**, 37, **38**, **39**, **40**, **41**, 42, **43**, 46, 47, 49, 50, 51, **53**, **54**, **55**, **56**, **57**, **58**, **59**, 60, 61, **62**, 63, **64**, 65, 66, 67, 68, 69, 70, **71**, 72, **73**, 74, 75, 76, 77, 78, 79, 80, 81, **82**, **83**, **84**, **85**, **86**, **87**, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, **99**, 100, 101, **102**, 103, **104**, **105**, 106, 107, 108, 109, **110**, 111, **112**, **113**, **114**, **115**, 117, 118, 119, 120, 121, **122**, **123**, **124**,125, 126, 127, 128, 129, **130**, **131**, **132**, **133**, **134**, **135**, 137, 138, 139, 140, 141, 142, 143, 144, 145, **146**, **147**, **148**, **150**, **151**, **152**, 153]

1) FEATURE SELECTION

As mentioned before, we used RFECV to select the essential attributes and filtered the dataset accordingly. The treatment of electing which feature to use depends on returned RFECV value, whether only by selected features or combined by the ranked features. The column on the right in Table 3 shows the attributes based on the RFECV features selection used in our experiments. We ordered the selected attributes based on the index number and made a subset if RFECV returned more than the required attributes. Conversely, as shown in the right column, we used selected attributes combined with the adjacent best attributes based on feature rank. The attributes combination could be seen in Conf. III, V, VI, VII, and VIII, indicated by bold index number.

The number of optimal attributes returned by RFECV ranges from 45 to 116 for every iteration, causing the number of combined attributes utterly varied. We argue that this is due to the RFECV process done after multiple sampling on different sessions for each configuration. However, despite the inconsistency, the attributes' appearance throughout every configuration shows a certain level of consistency. For example, the average appearance of attributes for all configurations on another configuration with more attributes, in general, is up to 93.37% $\pm$ 9.65%. For better illustration, this means that 48 out of 49 attributes in Conf. III are also present in Conf. V. Fig. 4 gives a general overview of these configurations by calculating the average of attributes appearance between each configuration to another.

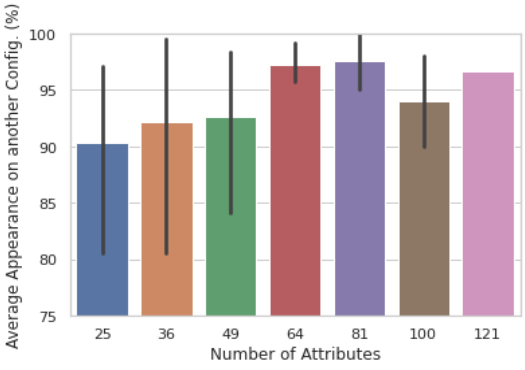


FIGURE 4. Average of attributes appearance compared to another configuration with more attributes.

2) CNN

Our study aims to train a robust model to learn the pattern of projected images. Since we have eight different configurations, we train our model based on each configuration separately. To measure the robustness of each model, as mentioned before, we run two different tests. The first test is a part of the train-validate-test sequence. We iterated the whole sequence three times. Therefore, we could calculate the average and standard deviation of each metric. We use the model from the last iteration to classify 15,000 images produced regarding related configuration. Table 4 shows the F1 score and accuracy from each model for both tests.

The train-validate-test results are the average value with standard deviation from three iterations in the first sequence. The comparison between the two sequences shows that each model performed better in the stress test than the train-validate test. Even though we could not define whether or not the difference between the two sequences is significant enough, we can still conclude that the models produced are pretty robust.

We highlight three out of eight configurations since these models produce the top-3 results in general. Models trained

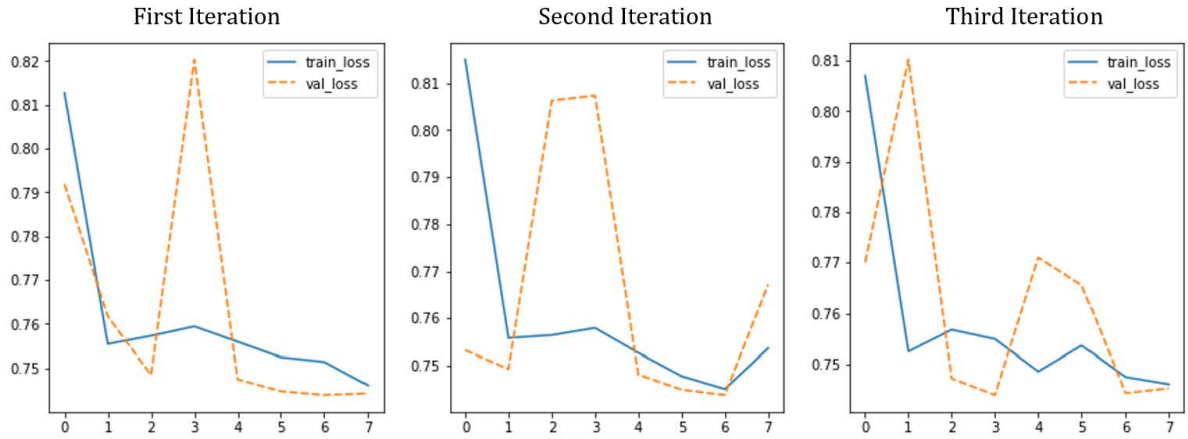


FIGURE 5. Loss during model training using 64 attributes.

TABLE 4. CNN test results for each configuration.

Config.	Attr. used	F1 Score	
		train-val-test	stress test
1	25	96.33% ±0.12%	96.84%
2	36	98.71% ±0.41%	99.35%
3	49	99.2% ±0.29%	99.61%
4	**64**	**99.54%** ±**0.15%**	**99.73%**
5	**81**	**99.61%** ±**0.06%**	99.71%
6	100	99.21% ±0.21%	96.26%
7	121	99.49% ±0.11%	99.56%
8	**144**	**99.6%** ±**0.11%**	99.42%
Config.	**Attr. used**	**Accuracy**	
		train-val-test	**stress test**
1	25	95.82% ±0.17%	96.38%
2	36	98.59% ±0.48%	99.32%
3	49	99.25% ±0.32%	99.60%
4	**64**	**99.54%** ±**0.15%**	**99.73%**
5	**81**	**99.61%** ±0.06%	99.70%
6	100	99.18% ±0.22%	99.30%
7	121	99.48% ±0.12%	99.55%
8	**144**	**99.6%** ±**0.11%**	99.40%

using Conf. V and VIII gave the highest F1 score and accuracy during the first test, while models trained using 64 attributes came in the third place. Regardless of the close results between Conf. V and VIII, the results tend to decrease after using more than 81 attributes. The stress test shows that models trained using 144 attributes produce the worst performance after Conf. I and II. The model of the Conf. VIII is also the only model that shows declined results, whether F1 score or accuracy. We argue that the drop in performance of the model trained using 144 attributes is due to overfitting and the inability to perceive a bigger variance of the stress test dataset. Alternately, the model trained using 64 attributes came in the first place, with the highest F1 score and accuracy. We took Conf. VI with 64 attributes as our benchmark for deeper analysis and model comparison.

We trained our model using eight epochs and iterated the entire process three times. Each iteration to train models took on average 20 minutes to complete. We keep track of model loss during this process, as shown in Fig. 5. The loss during model training indicates a lack of validation data since our validation loss fluctuated. The fluctuation was due to our relatively small validation set compared to our training set; therefore, it could not validate the general representation of the data. However, we argue that our model was good enough since our training loss was converged from the first epoch onward. This statement was further proven by the stress test, using more than double the size of our training data.

TABLE 5. Stress test results.

Metric	Normal	Impersonation	Injection	Flooding
$Precision$	99.97%	96.81%	91.03%	100%
DR	99.73%	99.80%	99.02%	100%
FAR	0.24%	0.12%	0.14%	0.0%
FNR	0.26%	0.19%	0.98%	0.0%

We performed the stress test using a total of 15,000 images, consisting of 13,797 images of the "Normal" class, 518 images of "Impersonation" class, 205 images of "Injection" class, and 480 images of "Flooding" class, sampled randomly from the original test set. The confusion matrix in Fig. 6 shows relatively good results, especially in class 3, representing the "Flooding" attack with a 100% detection rate. Other classes also have a high detection rate above 99%, with the lowest at 99.02% for "injection". Concurrently, the false-alarm rates and false-negative rates for all respective classes are under 1%, as shown in Table 5. The highest value for false alarm rates is 0.24% for the "normal" class, but we argue this is due to imbalance class distribution with the domination of the "normal" class on 11-to-1 compared to other classes combined. The overall results of the stress test confirm the high performance produced by our model given training dataset.

VI. COMPARISON WITH STATE-OF-THE-ART MODELS

We have the best-fit configuration to classify our projected data table; we took several statistical machine learning algorithms to classify the same data as a comparison.

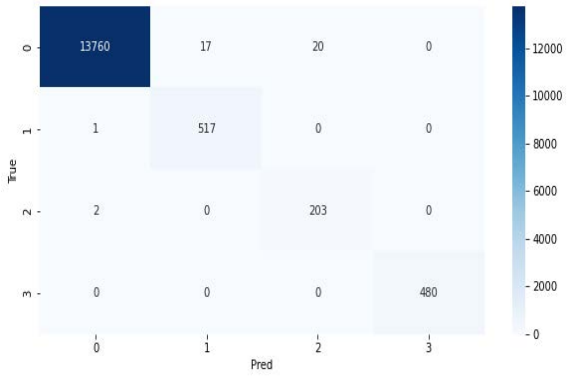


FIGURE 6. Stress test confusion matrix. 0 represent "Normal," 1 represent "Impersonation," 2 represent "Injection," and 3 represent "Flooding."

TABLE 6. SVM results for each kernel.

Config.	F1 Score		
[attr. used]	Linear	Polynomial	RBF
1 [25]	94.33%	93.89%	92.24%
2 [36]	96.58%	95.68%	96.05%
3 [49]	95.22%	94.32%	93.61%
4 [64]	97.59%	97.20%	96.10%
5 [81]	96.73%	95.57%	95.16%
6 [100]	96.64%	95.57%	95.17%
7 [121]	96.74%	95.54%	95.20%
8 [144]	96.74%	95.56%	95.21%
Config.	**Accuracy**		
[attr. used]	**Linear**	**Polynomial**	**RBF**
1 [25]	93.06%	92.52%	89.99%
2 [36]	96.13%	94.85%	95.53%
3 [49]	94.39%	93.15%	92.21%
4 [64]	97.36%	96.91%	95.59%
5 [81]	96.27%	94.67%	94.28%
6 [100]	96.73%	94.66%	94.29%
7 [121]	96.74%	94.62%	94.33%
8 [144]	96.74%	94.64%	94.34%

The classification used similar data points as the CNN stress test. We process the data through different algorithms, namely SVM, Decision Tree, and XGBoost. SVM was chosen as it is an estimator parameter used on RFECV. The way SVM works highly depend on the kernel implemented to separate the data into different classes. We used three different kernels and chose the best results for comparison to other algorithms. SVM model with Linear kernel produced the best result in all configurations compared to other kernels, namely RFB and Polynomial, as shown in Table 6. We chose the Decision Tree as the second algorithm since it is generally used to handle multi-class problems. We also added XGBoost with default hyper-parameter for better comparison.

Table 7 shows the result comparison between the three algorithms. The SVM model delivered the worst performance with the highest results, around 97% for accuracy and F1 score, despite using the optimal kernel compared to other options. Simultaneously, the Decision Tree delivered better results on 99.70% for accuracy and F1-score with 49 attributes. Finally, the best results of all statistical models were acquired by the XGBoost model, with a paper-thin difference compared to the Decision Tree. XGBoost also shows

TABLE 7. Statistical model results comparison for each configuration.

attr.	SVM		Decision Tree		XGBoost	
	F1 score	Acc.	F1 score	Acc.	F1 score	Acc.
25	94.33%	93.06%	99.34%	99.33%	98.93%	98.88%
36	96.58%	96.12%	99.49%	99.49%	99.65%	99.64%
49	95.22%	94.39%	99.70%	99.70%	99.66%	99.66%
64	97.59%	97.36%	99.52%	99.50%	**99.73%**	99.71%
81	96.73%	96.26%	99.55%	99.54%	99.72%	99.72%
100	96.73%	96.26%	99.44%	99.42%	99.72%	99.72%
121	96.74%	96.27%	99.32%	99.30%	99.72%	99.72%
144	96.74%	96.27%	99.37%	99.36%	99.72%	99.72%

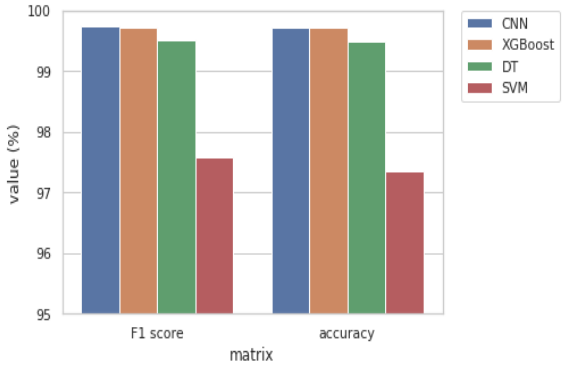


FIGURE 7. Comparison between CNN and statistical weighted metrics.

alignment with our CNN result with an F1 score of 99.73% and an accuracy of 99.71% on the 6th configuration.

Fig. 7 shows the direct comparison between CNN and all statistical methods. Performance-wise, all methods deliver good results since the F1 score and accuracy are above 95%. If we only look at the best result from CNN and statistical approach, the difference is quite slim, with CNN higher than XGBoost 0.01%. Therefore, we could conclude that CNN models perform comparably to the statistical method, based on only the weighted matrix F1 score and accuracy. Meanwhile, if we also consider the execution time during training and testing, statistical methods are preferable since they only need execution time less than 20 minutes. Despite learning data features and delivering exceptional performance, the flaw in execution time becomes a major drawback to CNN implementation on tabular data.

VII. CONCLUSION AND FUTURE WORK

This paper present a novel mapping method of tabular data into grid-based data that convolutional neural networks can learn for IDS purposes. We map the tabular data of wireless attacks into images by exploiting the sequences of attributes placed in a matrix. The matrix values are represented by two-coded color mapping, blue and red. This approach outperforms the accuracy of previous work [24], which employs grayscale mapping. With the optimized size of attributes using RFECV feature selection and ResNet50 CNN classifier, we achieve the best performance with 99.73% of

F1 score. We also successfully keep the false alarm rate low, about 0.24%.

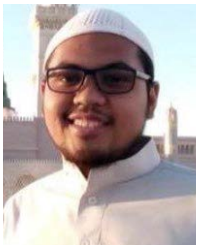
In the near future, the sequence of attribute placement in the matrix would be interesting to examine. CNN-based classifier exploits the underlying spatial information. Then different placement might affect the learning results. In addition, lightweight processing should be the main consideration when using CNN-based classifiers. Finally, different datasets can verify that the proposed method is suitable for different domain knowledge.

REFERENCES

[1] A. Ghasempour, "Internet of Things in smart grid: Architecture, applications, services, key technologies, and challenges," *Inventions*, vol. 4, no. 1, p. 22, Mar. 2019.
[2] M. Nivaashini and P. Thangaraj, "Computational intelligence techniques for automatic detection of Wi-Fi attacks in wireless IoT networks," *Wireless Netw.*, vol. 27, no. 4, pp. 2761–2784, May 2021.
[3] M. Hassaballah, M. A. Hameed, A. I. Awad, and K. Muhammad, "A novel image steganography method for industrial Internet of Things security," *IEEE Trans. Ind. Informat.*, vol. 17, no. 11, pp. 7743–7751, Nov. 2021.
[4] Z. Ahmad, A. Shahid Khan, C. Wai Shiang, J. Abdullah, and F. Ahmad, "Network intrusion detection system: A systematic study of machine learning and deep learning approaches," *Trans. Emerg. Telecommun. Technol.*, vol. 32, no. 1, p. e4150, Jan. 2021.
[5] S. N. Mighan and M. Kahani, "A novel scalable intrusion detection system based on deep learning," *Int. J. Inf. Secur.*, vol. 20, no. 3, pp. 387–403, Jun. 2021.
[6] A. Thakkar and R. Lohiya, "A survey on intrusion detection system: Feature selection, model, performance measures, application perspective, challenges, and future research directions," *Artif. Intell. Rev.*, vol. 55, pp. 453–563, Jul. 2021.
[7] D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE Trans. Syst., Man, Cybern.*, vol. 21, no. 3, pp. 660–674, May 1991.
[8] M. Pal, "Random forest classifier for remote sensing classification," *Int. J. Remote Sens.*, vol. 26, no. 1, pp. 217–222, 2007.
[9] J. A. K. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Process. Lett.*, vol. 9, no. 3, pp. 293–300, Jun. 1999.
[10] I. Rish, "An empirical study of the naive Bayes classifier," in *Proc. Workshop Empirical Methods Artif. Intell. (IJCAI)*, 2001, vol. 3, no. 22, pp. 41–46.
[11] X. Lin, F. Yang, L. Zhou, P. Yin, H. Kong, W. Xing, X. Lu, L. Jia, Q. Wang, and G. Xu, "A support vector machine-recursive feature elimination feature selection method based on artificial contrast variables and mutual information," *J. Chromatography B*, vol. 910, pp. 149–155, Dec. 2012.
[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[13] D. Theckedath and R. R. Sedamkar, "Detecting affect states using VGG16, ResNet50 and SE-ResNet50 networks," *Social Netw. Comput. Sci.*, vol. 1, no. 2, pp. 1–7, Mar. 2020.
[14] C. Kolias, G. Kambourakis, A. Stavrou, and S. Gritzalis, "Intrusion detection in 802.11 networks: Empirical evaluation of threats and a public dataset," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 184–208, 1st. Quart., 2016.
[15] V. L. L. Thing, "IEEE 802.11 network anomaly detection and attack classification: A deep learning approach," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Mar. 2017, pp. 1–6.
[16] S. M. Kasongo and Y. Sun, "A deep learning method with wrapper based feature extraction for wireless intrusion detection system," *Comput. Secur.*, vol. 92, May 2020, Art. no. 101752.
[17] M. E. Aminanto, R. Choi, H. C. Tanuwidjaja, P. D. Yoo, and K. Kim, "Deep abstraction and weighted feature selection for Wi-Fi impersonation detection," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 3, pp. 621–636, Mar. 2018.
[18] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.
[19] L. Jing and Y. Tian, "Self-supervised visual feature learning with deep neural networks: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 11, pp. 4037–4058, Nov. 2021.
[20] R. S. H. Wicaksono, A. A. Septiandri, and A. Jamal, "Human embryo classification using self-supervised learning," in *Proc. 2nd Int. Conf. Artif. Intell. Data Sci. (AiDAS)*, Sep. 2021, pp. 1–5.
[21] H. Firat and D. Hanbay, "Classification of hyperspectral images using 3D CNN based ResNet50," in *Proc. 29th Signal Process. Commun. Appl. Conf. (SIU)*, Jun. 2021, pp. 1–4.
[22] A. A. Septiandri, A. Jamal, P. A. Iffanolida, O. Riayati, and B. Wiweko, "Human blastocyst classification after *in vitro* fertilization using deep learning," in *Proc. 7th Int. Conf. Advance Inform., Concepts, Theory Appl. (ICAICTA)*, Sep. 2020, pp. 1–4.
[23] Q. Duan, X. Wei, J. Fan, L. Yu, and Y. Hu, "CNN-based intrusion classification for IEEE 802.11 wireless networks," in *Proc. IEEE 6th Int. Conf. Comput. Commun. (ICCC)*, Dec. 2020, pp. 830–833.
[24] I. Al-Turaiki and N. Altwaijry, "A convolutional neural network for improved anomaly-based network intrusion detection," *Big Data*, vol. 9, no. 3, pp. 233–252, Jun. 2021.
[25] B. Sun, L. Yang, W. Zhang, M. Lin, P. Dong, C. Young, and J. Dong, "SuperTML: Two-dimensional word embedding for the precognition on structured tabular data," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2019, pp. 2973–2981.
[26] M. Popescu and A. Naaji, "Detection of small tumors of the brain using medical imaging," in *Handbook of Decision Support Systems for Neurological Disorders*, H. D. Jude, Ed. New York, NY, USA: Academic, 2021, ch. 3, pp. 33–53. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B978012822271300013X
[27] M. Ahmad and V. Ramachandran, "Cafe latte with a free topping of cracked WEP retrieving WEP keys from road warriors," in *Proc. Conf. ToorCon*, San Diego, CA, USA, 2007.
[28] (2018). *Airbase-NG*. Accessed: Feb. 9, 2020. [Online]. Available: https://www.aircrack-ng.org/doku.php?id=airbase-ng#how_does_the_hirte_attack_work
[29] Y. Song, C. Yang, and G. Gu, "Who is peeping at your passwords at Starbucks?—To catch an evil twin access point," in *Proc. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jul. 2010, pp. 323–332.
[30] R. Beyah and A. Venkataraman, "Rogue-access-point detection: Challenges, solutions, and future directions," *IEEE Secur. Privacy Mag.*, vol. 9, no. 5, pp. 56–61, Sep. 2011.
[31] E. Tews, R.-P. Weinmann, and A. Pyshkin, "Breaking 104 bit WEP in less than 60 seconds," in *Proc. Int. Workshop Inf. Secur. Appl.* Jeju-si, South Korea: Springer, 2007, pp. 188–202.
[32] E. Tews and M. Beck, "Practical attacks against WEP and WPA," in *Proc. 2nd ACM Conf. Wireless Netw. Secur. (WiSec)*, 2009, pp. 79–86.
[33] A. Bittau, "The fragmentation attack in practice," in *Proc. IEEE Symp. Secur. Privacy, IEEE Comput. Soc.*, Sep. 2005, pp. 1–13.
[34] A. Mart, U. Zurutuza, R. Uribeetxeberria, M. Fern, J. Lizarraga, A. Serna, and I. Vélez, "Beacon frame spoofing attack detection in IEEE 802.11 networks," in *Proc. 3rd Int. Conf. Availability, Rel. Secur.*, Mar. 2008, pp. 520–525.
[35] S. S. Sawwashere and S. U. Nimbhorkar, "Survey of RTS-CTS attacks in wireless network," in *Proc. 4th Int. Conf. Commun. Syst. Netw. Technol.*, Apr. 2014, pp. 752–755.
[36] L. F. Meiners, "But…My station is awake! Power save denial of service in 802.11 networks," Core Secur. Technol., Eden Prairie, MN, USA, Tech. Rep., 2009.

MUHAMAD ERZA AMINANTO (Member, IEEE) received the Ph.D. degree from the School of Computing, Korea Advanced Institute of Science and Technology (KAIST), South Korea, in 2018. He is currently a Lecturer with the University of Indonesia (UI) and a Cooperative Visiting Researcher at the Cybersecurity Research Institute, National Institute of Information and Communications Technology (NICT), Japan. His current research interests include machine learning, IDS, and cybersecurity.

R. SATRIO HARIOMURTI WICAKSONO received the B.S. degree in informatics from Universitas Al Azhar Indonesia, Indonesia, in 2021. He is currently a Product Data Analyst at PT. Multidaya Teknologi Nusantara (eFishery). His current research interest includes machine learning, computer vision, and natural language processing.



ACHMAD ERIZA AMINANTO received the B.S. degree in statistics from Universitas Indonesia, Indonesia, in 2019, where he is currently pursuing the master's degree with the School of Strategic and Global Studies. His current research interest includes machine learning, statistics, and information security.

HARRY CHANDRA TANUWIDJAJA received the B.S. and M.S. degrees in electrical engineering from the Bandung Institute of Technology (ITB), Indonesia, in 2013 and 2015, respectively, and the Ph.D. degree from the School of Computing, Korea Advanced Institute of Science and Technology (KAIST), South Korea, in 2021. He is currently a Researcher with the National Institute of Information and Communication Technology (NICT), Japan. His current research interests include machine learning, web phishing detection, malware detection, privacy-preserving, and intrusion detection systems.

LIN YOLA is currently a Senior Lecturer with the School of Strategic and Global Studies, Universitas Indonesia, with a strong academic and professional background in urban, energy, and technology innovation. She is best known for her model of ''Climatically Responsive Urban Configuration.'' She is currently the Head of the Center for Spatial Data and Analysis, which mainly focuses on the simulation model and integrated eco-spatial GNSS technologies. Besides the significant cross-discipline studies, her projects and publications impose technology innovation as the strategic methodology in addressing global development issues.

KWANGJO KIM (Senior Member, IEEE) received the B.Sc. and M.Sc. degrees in electronic engineering from Yonsei University, Seoul, South Korea, in 1980 and 1983, respectively, and the Ph.D. degree from the Division of Electrical and Computer Engineering, Yokohama National University, Yokohama, Japan, in 1991.

He was a Visiting Professor with the Massachusetts Institute of Technology, Cambridge, MA, USA; the University of California at San Diego, La Jolla, CA, USA, in 2005; and the Khalifa University of Science, Technology and Research, Abu Dhabi, UAE, in 2012, and an Education Specialist with the Bandung Institute of Technology, Bandung, Indonesia, in 2013. He is currently an Emeritus Professor with the School of Computing, Korea Advanced Institute of Science and Technology, Daejeon, South Korea; the President of the International Research institute for Cyber Security; and the honorable President of the Korea Institute of Information Security and Cryptography (KIISC). His current research interests include the theory of cryptology, information security, and its applications. He served as a Board Member of the International Association for Cryptologic Research (IACR) from 2000 to 2004, the Chairperson of the Asiacrypt Steering Committee from 2005 to 2008, and the President of KIISC in 2009. He is a fellow of IACR and a member of IACR, IEICE, and ACM. He served as the General Chair for Asiacrypt2020 and PQCrypto2021 held in a virtual conference.