Network Intrusion Detection via Flow-to-Image Conversion and Vision Transformer Classification

CHI MAI KIM HO1, KIN-CHOONG YOW2, (Senior Member, IEEE), ZHONGWEN ZHU2, (Senior Member, IEEE), AND SARANG ARAVAMUTHAN3

1Faculty of Engineering and Applied Sciences, University of Regina, Regina, SK S4S 0A2, Canada
2Global AI Accelerator (GAIA) Team, Ericsson Canada Inc., Saint-Laurent, QC H4S 0B6, Canada
3Global AI Accelerator (GAIA) Team, Ericsson Inc., Santa Clara, CA 95054, USA

Corresponding author: Kin-Choong Yow (kin-choong.yow@uregina.ca)

This work was supported by the Ericsson—Global Artificial Intelligence Accelerator in Montreal through the Mitacs Accelerate Program under Grant IT16753.

ABSTRACT In recent years, computer networks have become an indispensable part of our life, and these networks are vulnerable to various types of network attacks, compromising the security of our data and the freedom of our communications. In this paper, we propose a new intrusion detection method that uses image conversion from network data flow to produce an RGB image that can be classified using advanced deep learning models. In this method, we proposed to use the decision tree algorithm to identify the important features, and a windowing and overlapping mechanism to convert the varying input size to a standard size image for the classifier. We then use a Vision Transformer (ViT) classifier to classify the resulting image. Our experimental results show that we can achieve 98.5% accuracy in binary classification on the CIC IDS2017 dataset, and 96.3% on the UNSW-NB15 dataset, which is 8.09% higher than the next best algorithm, the Deep Belief Network with Improved Kernel-Based Extreme Learning (DBN-KELM) method. For multi-class classification, our proposed method can achieve a testing accuracy of 96.4%, which is 5.6% higher than the next best method, the DBN-KELM.

INDEX TERMS Network intrusion detection, flow-to-image conversion, convolutional neural networks, vision transformers, image classification.

I. INTRODUCTION

In 2019, the Accenture company [1] published a report on the cost of cybercrime. Over the last five years, the cost of cybercrime has increased by 67%. More than 16 essential industry groups became prime targets of cyber attacks with losses up to $13.0 million in 2018. In recent years, along with the continuous development of new network technologies, more sophisticated and dangerous new types of cyberattacks are emerging. These malware can attack via many different platforms, and they have the ability to hide themselves in the network for a long time without being detected. Therefore, ensuring network security over the network layers becomes extremely important. To mitigate these attacks, Intrusion Detection Systems (IDSs) are used to continuously monitor the network and to classify or predict potential malicious actions.

There are many different types of IDSs and currently, the Signature detection based IDSs and Anomaly detection based IDSs are the most popular and most effective in network security. Research on signature based IDSs and anomaly detection based IDSs started a long time ago [2] but is still an active area because of the diversity of components [3] that create the IDSs such as the model [4], [5], the research method [6] or the framework [7]. Besides, there is also no consensus on the evaluation and validation methods of IDSs as researchers use many diverse approaches such as experiments [9], [10], simulations [11], or both [12] to evaluate their models. The method of using evaluation metrics on the test datasets is still the most popular and the most widely used today. Our project can be used to improve both Signature detection based IDSs and Anomaly detection based IDSs.
The performance of the IDSs is only as good as the datasets that are used to train them. A variety of data sets on network intrusions have been developed in recent years, and the two most popular aggregated network intrusion datasets today, CIC-IDS2017 and CSE-CIC-IDS2018, from the Canadian Institute for Cybersecurity (CIC) in the University of New Brunswick (UNB), have aggregated more than 78 features of a single network request. These datasets are built on the evaluation criteria of Gharib et al. [8] and can be used to detect a wide range of attacks.

In recent years, machine learning is applied to increase the accuracy of IDSs, mostly to improve the detection algorithm and the input network flow review. One of the earliest machine learning methods, KNN (aka traditional clustering) has been in use since 1996. More recently, advanced detection and classification algorithms based on statistical, signal processing, information theory [13], as well as digital signal processing [4], [14] have been proposed. However, these technologies are still reporting low accuracies in multi-class classification and high false alarm rates [15]. According to the authors in [16], the network flow data is massive and has all the characteristics of time series data which can easily lead to packet loss and interruptions during processing, severely affecting detection accuracies and false alarm rates.

A. PROBLEM DESCRIPTION

One of the main challenges of Network IDSs (NIDS) is the massive data size of the network flows that needs to be handled. For each flow, there are many possible features (e.g. inter-arrival time, packet size, etc.) that can be extracted, but not every feature is useful in determining whether an attack is underway or not. Another challenge is the variety of network intrusions and the ambiguity of each type of intrusion. There are many different types of attacks and some of them share similar features, leading to mis-classifications. A third challenge is that there are always new types of attacks that have not been seen before by the NIDS, and thus the NIDS is not able to recognize that an attack is in progress.

The questions are therefore how can we learn which features are important in detecting such intrusions on a set of real-time network data? How can we increase the self-learning ability of the system to be able to discriminate between different types of attacks? And how can we make the most accurate classification between the malicious and benign network flows so that we can detect new types of attacks?

To improve the performance and accuracy of our NIDS, we propose a method that converts a network flow pattern within a specific time interval into a two-dimensional image. One dimension of the image is the various measurements of the network flow (i.e. features), and the other dimension represents the values of these measurements over time. We will then use image processing techniques to classify these images as malicious or benign. This approach allows all or majority available information (i.e. other measurements or features that may not be directly related to the attacks) to be used in the classification to reduce false positives.

In addition, some of the advances in image classification techniques can also help to improve processing speeds. For example, if we use convolutional neural networks (CNNs), we can adjust the size of the mask and the stride of the convolution to achieve better computational efficiency. Similarly, using the multi-head attention mechanisms in Vision Transformer (ViT), we can also focus our computational efforts only in regions of interest.

B. CONTRIBUTIONS

The novelty in this work is that our proposed method selects the most important features in the network flows based on the most common types of network attacks, and converts them into an RGB image representation. We then use state-of-the-art image processing techniques to classify these images, which uses regional information to improve computational efficiency and to reduce false positives. The contributions in this paper are:

- **Feature Selection.** We proposed an approach using decision trees to rank the various network flow measurements (i.e. features) for the different kinds of attacks and select the most important ones for our classification algorithm.
- **Flow-to-Image Conversion.** We proposed an approach to convert the network flows into an image by selecting the most important measurements and converting them into an RGB image representation.
- **Classification using ViT.** We investigated two approaches to classify the resulting image, CNN and ViT. We show that the results using ViT is the best and it surpassed other state-of-the-art algorithms.

II. RELATED WORK

In this research, detection and mitigation of security attacks will be our main focus. There are four main types of network attack traffic that are described in the KDDCup 1999 dataset: Denial of Service (DoS), Probe, Remote to Local (R2L), and User to Root (U2R). DoS refers to all types of attacks that create excessive network traffic and disrupt the operations of certain targeted nodes. DoS can be detected by examining network features such as source bytes, packet rate, the total number of packets sent, etc. [17]. Probe attacks are attacks conducted by sending empty packets to gain knowledge about the network. They are usually detected by features such as connection duration or source bytes. R2L refers to remote access attacks in which an attacker tries to gain access to a remote system. Related features include connection duration, requested service, or failed login. U2R is a type of attack where an attacker tries to log in to a standard account and then gain root administrator access. They are usually identified by features like the number of files created, or the number of shell prompts called.
There are many machine learning approaches developed to detect these kind of attacks. Tan et al. [18] applied real-time network abnormal detection techniques such as the Euclidean Distance Map (EDM) to display changes in the observed data objects (e.g. in a flooding attack, we want to observe the source of the attack packets). These changes in the EDM will show the occurrences of malicious instructions. Another technique is the Collaborative anomaly detection framework (CADF) from Moustafa et al. [19], which captures and logs network data. It consists of 3 modules: capturing and logging data, pre-processing data, and decision module for intrusion detection. In the decision module, Gaussian Mixture Models (GMM) and interquartile range take the role of the primary abnormal pattern identification tool. The architecture deploying this framework also uses Software as a Service (SaaS) to make installation on cloud systems easier.

Algorithms using neural network architectures have also been proposed by researchers. For example, the enhancement of resilient backpropagation artificial neural network (ERBP) algorithms to train a multi-layer perception was proposed by Reyadh et al. [20]. ERBP makes direct adjustments to the weight settings based on the local gradient depending only on the sign of the derivative. This system used the NSL-KDD99 dataset for training and testing. In another work, Deep Convolutional Neural Networks (DCNNs) for the network anomaly detection problem was proposed by Sheraz and Yasir [21]. DCNNs were trained using GPUs on the NSL-KDD99 training dataset. The system was compared with other classification algorithms using metrics such as Receiver operating characteristics (RoC) curve, Area under RoC curve (AuRoC), Accuracy, precision-recall-curve, and mean Average precision.

To detect R2L, U2R, DoS, and Probe attacks, Thaseen and Kumar [22] used a Multi-class support vector machine (SVM) and chi-square feature selection. The Multi-class SVM shows an advantage in reducing training and testing time. The results performed on the NSL-KDD99 dataset, which is an enhanced version of the KDDCup 1999 dataset, showed that the model results in high detection rates and low false alarm rates in comparison to other traditional approaches. In another work, Manzoor and Morgan [23] used Support Vector Machines on Apache storm-based-IDS in the real-time mode. The Multi-class SVM shows an advantage in reducing training and testing time. The results performed on the NSL-KDD99 dataset, which is an enhanced version of the KDDCup 1999 dataset, showed that the model results in high detection rates and low false alarm rates in comparison to other traditional approaches.

Another interesting model was proposed by Mowla et al. [24] and built based on IDSs for Medical Cyber Physical Systems. The idea in this project is to convert a 5-class problem (4 types of attacks plus a normal non-attack scenario) into a number of 2-class problems. First, the incoming input traffic is classified into whether it is malicious or benign. If it is malicious, it will then be sent to a different neural network and classified to whether it is a probing attack or not. This process is then repeated for the R2L attack and the U2R attack. However, this approach has the shortcoming that it should test for the most probable type of attacks first, otherwise the mis-classification rate can be significant. The results of this approach have been shown to be comparable with the traditional multi-class approach, but it takes less time to run.

A more recent approach is to convert the network features into a 2D image and then apply CNNs to perform the classification [25]. In this approach, the authors converted each request flow to a 2D matrix, where the size of the matrix corresponds to the total number of data features to be examined (for example, for a request flow of 121 features, they convert that into a $11 \times 11$ matrix). Then they will convert 2D matrix to 2D image. After the 2D image is formed, the authors will then apply CNN techniques to train it and perform classification. Another approach from [26] tried to convert each request to an image, using CNNs to perform the classification.

Despite the emergence of many new deep learning models, almost all of these models are still only processing data on a request-by-request basis which cannot take advantage of the correlation between data requests in neighboring time frames. The combination and co-processing of these data, within a certain time frame, is the key to increasing the classification accuracy and decreasing the false alarm rate.

### III. METHODS

#### A. OVERALL ARCHITECTURE

Figure 1 shows the overall architecture of our system. Our proposed solution consists of 3 main modules: the Data Pre-processing module, the Flow to Image Converter module and the Classifier module. During the training phase of the system, we apply the decision tree classification process to the training data to find the importance of the features so that we can choose the most important features to be used in the conversion of the data flow to an image. During normal execution, the data that comes in will first go through the Data Pre-processing module where we will extract the important features that were found during the training phase. The pre-processed output will then go through the Flow to Image Converter module and be converted to a series of RGB images. These images are then sent to the Classifier module and will be classified by the deep learning classifier into whether it contains benign or malicious flows.

#### B. DATA PRE-PROCESSOR

To build the trained model, we need a dataset that has already been labeled. Popular datasets that are available to the intrusion detection research community includes the CIC IDS2017 [28] and the UNSW-NB15 [29]. These datasets...
store all their network flow features in a CSV file, which makes it easy for us to filter the data by the arrival time of each request, or by the server that received it.

1) DETERMINING THE IMPORTANT FEATURES
Before we can use the Data Pre-processor module for intrusion detection, we need to determine what are the important features that we want to extract from the CSV file. Each row of the CSV file represents one network flow, and each column represents a particular network feature. For example, in the CIC IDS2017 dataset, there are 78 features and hence the CSV file has 78 columns.

To determine which features are important in identifying the attacks, the decision tree classification algorithm was applied to choose the most important features for each type of attack. We used the Mahalanobis Distance-based Oversampling (MDO) technique [30] to balance the distance of each class so as to increase the accuracy of the decision tree. MDO is the oversampling technique using the Mahalanobis distance to generate synthetic samples. This method is proven to be effective in multi class and multi-label classification models using imbalance datasets for training.

For example, in the CIC IDS2017 dataset, there are 9 different classes of network flow data (8 malicious and 1 benign). They are the Patator, Benign, DDoS, DoS, Bot, PortScan, Heartbleed, Infiltration. We build a decision tree for combination of all types of attack with benign requests and we look at the information gain (IG) of 24 features which have the highest value.

By selecting features that have an IG value of at least 0.001 in at least one type of attack, we obtained 24 features that we will use in our NIDS. These features include Avg Fwd Seg Size, Subflow Fwd Bytes, Dest Port, Avg Pkt Size, Source Port, Bwd Pkt Length Min, Idle Min, Fwd Pkt Length Mean, PSH Flag Count, Flow Duration, Fwd Header Length, Fwd IAT Max, Fwd IAT Std, Idle Max, Init Win Bytes Bwd, Min Seg Size Fwd, Fwd IAT Min, Active Mean, Active Max, Act Data Pkt Fwd, Bwd Pkt Length Mean, Fwd Packets/sec, and Pkt Length Mean. The source IP is also encoded to facilitate the conversion to image form.

Table 1 shows the important features which are extracted from CIC IDS2017 [28] and Table 2 shows the important features which are extracted from UNSW-NB15 [29].

2) PREPARING THE DATA FOR FLOW TO IMAGE CONVERSION
After determining the important features needed for our algorithm, we can now use the Pre-processor module for intrusion detection. During the execution of the intrusion detection process, our Data Pre-processor module will read the input from the CSV file, where the rows of the CSV file represent the network flows, and the columns represent the 78 features.

First, the number of rows in the CSV files will be reviewed and checked if there are enough rows to generate a square image (this is because the final module (i.e. the classifier module) in our intrusion detection process requires a square image). Since we have identified 24 important features for the attacks that we are interested in (i.e. 24 columns), we need to make sure that the input CSV file have at least 24 rows too). If not, new empty rows with all 0 values for all features will be added to the CSV file. Next, we remove all other columns in the CSV file except the 24 columns that we have identified as important features for our intrusion detection system. This new CSV file will become the input to the next module, the Flow to Image Converter module.

C. FLOW TO IMAGE CONVERTER
The next step is to convert the network flows in the new CSV files to images. As the new CSV files are of different sizes (the number of columns corresponds to the number of features
TABLE 1. The important features that are extracted from CIC IDS2017.

| Feature name: | Detail: | IG: |
|---------------|--------|----|
| 1 Source IP   | IP number of source machine | 0.9 |
| 2 Avg Fwd Seg Size | Average size observed in the forward direction | 0.806 |
| 3 Subflow Fwd Bytes | The average number of bytes in a sub flow in the forward direction | 0.787 |
| 4 Dest Port Port number of destination machine | 0.62 |
| 5 Avg Pkt Size | Average sizes of packet | 0.61 |
| 6 Source Port | Port number of source machine | 0.527 |
| 7 Bwd Pkt Length Min | Minimum size of packet in backward direction | 0.518 |
| 8 Idle Min Minimum time a flow was idle before becoming active | 0.487 |
| 9 Fwd Pkt Length Mean | Average size of packet in forward direction | 0.469 |
| 10 PSH Flag Count | Count number of PSH flag | 0.4 |
| 11 Flow Duration Flow duration | 0.389 |
| 12 Fwd Header Bytes Total bytes used for headers in the forward direction | 0.346 |
| 13 Fwd IAT Max Maximum time between two packets sent in the forward direction | 0.328 |
| 14 Fwd IAT Std Standard deviation time between two packets sent in the forward direction | 0.324 |
| 15 Idle Max Maximum time a flow was idle before becoming active | 0.187 |
| 16 Init Win Bytes # of bytes sent in the initial window | 0.127 |
| 17 Min Seg Size Fwd Minimum segment size observed in the forward direction | 0.0122 |
| 18 Fwd IAT Min Minimum time between two packets sent in the forward direction | 0.0122 |
| 19 Active Mean Mean time a flow was active before becoming idle | 0.0122 |
| 20 Active Max Maximum time a flow was active before becoming idle | 0.0122 |
| 21 Act Data Pkt # of packets with at least 1 byte of TCP data payload in the forward direction | 0.0122 |
| 22 Bwd Pkt Length Mean Mean size of packet in backward direction | 0.00973 |
| 23 Fwd Packets/sec Number of forward packets per second | 0.00847 |
| 24 Pkt Mean Mean length of a flow | 0.00584 |

Some of the parameters used in the algorithm are as follows:
- **RGB_image**: initial (N × 24 × 3) image
- **RGB_feature_value**: the value of feature which is represented in the range of RGB color (0 to 0xFFFFFF)
- **Color_range**: equal 0xFFFFFF in decimal
- **row**: the value of the height of the csv file following time-frame, in this proposal, it is N
- **cols**: the value of the weight of the csv file following time-frame, in this proposal, it is defined by 24
- **feature_Max**: the maximum value of feature in input csv file
- **feature_Min**: the minimum value of feature in input csv file
- **Pixel_color**: the RGB value of one pixel which is represented as (RR, GG, BB)
- **conversion_value**: this value is calculated by RGB_feature_value divide 256(0xFF) and get integer.

Every 8 bits (i.e. 2 digits in hexadecimal) of the 24-bit RGB value is the value of each color component. The first 8 bits (the first two digits in hexadecimal) is the value of the Red component. The next 8 bits is the value of the Green component, and the last 8 bits is the value of Blue component. By taking modulo division of the 24-bit value by...
**Algorithm 1** The Algorithm of Mapping Flow Data to RGB Values

1: Input Nx24 csv file
2: **RGB_image** = (N, 24, 3)
3: col = 1
4: row = 1
5: while col <= 24 do
6:     while row <= N do
7:         **RGB_feature_value** = \[
8:             \text{Color\_range} \times \frac{\text{feature\_value}}{\text{Max} - \text{Min}}, \text{feature\_value}\%
9:     \]
10:     Pixel\_color = (int(conversion\_value/256)\%256), (conversion\_value\%256), (RGB\_feature\_value\%256))
11:     **RGB_image[col][row]** = Pixel\_color
12:     row = row + 1
13: end while
14: col = col + 1
15: end while
16: Output image

0xFF (i.e., 256 in decimal) twice, we can get the values of the individual red, green, and blue color components. These color components are then added into the corresponding location in the RGB image. This process is then repeated for all the selected features that are measured within a targeted time period, which was reported in the CSV file.

The training labels for each N \times 24 \times 3 image is based on whether an attack flow exists during the timeframe represented by the rows of the image. If an attack exists, the type of the attack will be taken as the training label for this image.

2) WINDOWING AND OVERLAPPING MECHANISM

The image that is generated in the previous section is an N\times24 \times 3 image, where N is the number of rows in the CSV file, 24 is the number of features that we selected from the Decision Tree algorithm, and 3 is the number of channels (Red, Green, Blue). As N will be different in every CSV file, we propose to reshape the image from N\times24 \times 3 to a square 24 \times 24 \times 3 image. This will make the design of the classifier in the next module easier as the dimensions will be fixed. We will use a windowing and overlapping mechanism to reshape the images as shown in Figure 2 below.

In Figure 2, we have an image as an example, we take the raw dataset from CSV file (left side) and slice those raw dataset into 7 different 24 \times 24 \times 3 images with an overlapping size of 12 rows (i.e., half of the square 24 \times 24 \times 3 image). The first square image will be extracted from the first row of the initial image (i.e., position height_0 = 0 and width_0 = 0) to the last row of the square output image (i.e., position height_0 = 24 and width_0 = 24). The second image will be cropped from the halfway point of the first image (i.e., position height_1 = height_0 + 12 and width_1 = 0) to a position 24 rows below that (i.e., position height_1 = height_0 + 12 + 24 and width_1 = 24). We will repeat this action until the row number exceeds the height of initial image. Then the final square image will be cropped from the last position of the initial image (i.e., position height_n = 78 - 24 and width_n = 0) to the end of initial image (i.e., position height_n = 78 and width_n = 24). The algorithm of windowing and overlapping mechanism is shown in Algorithm 2 below.

Some of the parameters used in the algorithm are as follows:
- **image_size** includes the number of column and row of the network flow which needs to be converted to image (N \times 24)
- **row** is the value of the height of the initial image (N \times 24 \times 3), in this proposal, it is N
- **cols** is the value of the weight of the initial image (N \times 24 \times 3), in this proposal, it is defined by 24
- **h** is the shifting value which is used for overlapping image
Algorithm 2 The Algorithm of Windowing and Overlapping Mechanism

1: Input image  
2: rows, cols = image_size  
3: if (h + 24) ≤ (rows - 1) then  
4: output_image = cropfrom(0, h) to (cols, h + expected_output_height)  
5: else  
6: output_image = cropfrom(0, rows−expected_output_height)to(cols, rows)  
7: end if  
8: h = h + overlappingSize  
9: Export square image  
10: if (h > rows) then  
11: Go to step 3  
12: end if

• expected_output_height is the height of the output image, in this proposal, the output image must be square so the expected_output_height is equal cols (= 24)

D. CLASSIFIER

This module is the classifier that will classify the 24 × 24 × 3 image into one of the output classes. We propose to use the ViT as the classifier. However, we will first introduce the more commonly used CNNs so that we will have a basis of comparison later.

1) CONVOLUTIONAL NEURAL NETWORKS (CNNs)

In this study, we will use the AlexNet Deep CNN architecture to perform binary classification for our NIDS. Deep CNN is an algorithm that has shown superiority in solving image processing and image classification problems. By taking advantage of the spatial correlation between pixels, CNNs are able to process complex input structures with fewer parameters than other deep learning architectures.

AlexNet is a popular architecture which uses 5 convolutional layers and three fully-connected layers. AlexNet uses Rectified Linear Units (ReLU) as activation functions which has the advantage of a shorter training time and is able to optimize the performance better than the tanh function. AlexNet also allows for multi-GPU training by putting half of the model’s neurons on one GPU and the other half on another GPU, so this model can process the large datasets faster. The architecture of AlexNet is shown in Figure 3.

2) VISION TRANSFORMER (ViT)

The transformer model was first introduced by Ashish et al. [31] in 2017. By using self-attention layers to capture long-term dependencies, the transformer easily learns more diverse interactions between spatial locations and process input asynchronously which helps the performance of the model become faster. After announcing the transformer’s outstanding results in the field of natural language processing (NLP), ViT [32] has also been proposed and applied in the field of image processing. ViT has been shown to compare somewhat better than the convolutional models.

The architecture of ViT includes the Transformer encoder with a Multiple Layer Perceptron (MLP) head for the classification and linear projection of flattened patches in the image so as to embed their positions in each part of the image. The input image is split into fixed-size patches, and each patch is linearly embedded, slotted, and then transformed into tuples (i.e. resulting vector sequences) which are then fed to a standard Transformer Encoder. A learnable “classification token” is then added to the string and then given to the classifier to classify the images. In the ImageNet Real input dataset, the ViT model produces the highest result of 90.72%, compared to 90.55% of EfficientNet L2. The application of the ViT model in our NIDS is shown in Figure 4.

We train our ViT in the same way as the linear ViT model of A. Dosovitskiy et al. [31]. Our ViT is first pre-trained with the data of ImageNet-21K, and then fine tuned by Stochastic gradient descent (SGD) with a momentum of 0.9. We then train all the ViT models using the AdamW optimizer with a learning rate of 2e-5.

E. DATASET DESCRIPTION

There are various datasets created to train NIDSs that are based on Machine learning. Examples include CIC-IDS-2017 [28], CSE-CIC-IDS-2018 [33], DARFA [34], KDD CUP 99 [9], NSL-KDD [35], UNSW-NB15 [29], ISCX2012 [36], and AFDA [37]. Table 3 shows the comparison between the different datasets for a deep learning based NIDS.

In this research, the two datasets that we used are the CIC IDS2017 and UNSW-NB15. We chose these two datasets not only because of the variety of attack types in them, but also because they have all the features related to server identification and arrival times that we need.

F. EVALUATION

There are various indicators that can be used to evaluate the performance of an NIDS, such as: True Positive (TP): the instances are properly predicted as attacks; True Negative (TN): properly predicted as normal instances; False Positive

| Dataset  | Features | Attack types                          |
|----------|----------|---------------------------------------|
| DARPA    | 41       | DoS, U2R, R2L, Probe                  |
| KDD CUP 99 | 41     | DoS, U2R, R2L, Probe                  |
| NSL-KDD  | 41       | DoS, U2R, R2L, Probe                  |
| UNSW-NB15  | 48      | DoS, worms, Backdoors, and Fuzzers    |
| ISCX2012 | 48 Flows | DoS, DDoS, Brute-force, Infiltration   |
| APDA     | System calls | Zero-day attacks, Stealth attack, C100 Webshell attack |
| CIC-IDS-2017 | 80     | DoS, DDoS, Brute-force, Infiltration, Portscan, Botnet, Web |
| CSE-CIC-IDS-2018 | 80   | DoS, DDoS, Brute-force, Infiltration, Portscan, Botnet, Web |
From these indicators, we use the following metrics to benchmark our NIDS:

- **Recall** ($RC$), is the percentage of properly predicted attacks of the sum of intrusion instances, formulated as

\[
RC = \frac{TP}{TP + FN} \times 100
\]  

- **Precision** ($P$), is the percentage of all normal and abnormal instances made out of all positive predictions that could have been made:

\[
P = \frac{TP}{TP + FP} \times 100
\]  

- **False Positive Rate** (FPR), is the percentage of normal instances of the sum of normal instances misidentified as
attacks, computed as:

$$FPR = \frac{TP}{TP + TN} \times 100$$  \hfill (3)$$

Accuracy (ACC), is the percentage of all normal and abnormal instances properly classified, defined as

$$ACC = \frac{TP + TN}{FP + FN + TP + TN} \times 100$$  \hfill (4)$$

IV. EXPERIMENTAL RESULTS

We take the CSV files of the CIC IDS2017 and UNSW-NB15 datasets and convert them to RGB images using the algorithm described in the previous section. Based on 24 most important features which is selected and sorted by decision tree algorithm, we generated the RGB image dataset. In the case of the CIC IDS2017 dataset, from 24 selected features (Table 1), the total images generated is 135,090 images when we use an overlap size of 12. Out of the 135,090 images, 38,562 images were benign (i.e. does not contain any malicious flows) and 96,528 images were malicious (i.e. contain at least one malicious flow). In the case of the UNSW-NB15 dataset, the image dataset has a huge imbalance between the Benign images and the Malicious images (the number of Benign images is 710,175 and the number of Malicious images is 78,038). To balance this dataset for Binary classification, we have to remove over 80% of the Benign images, resulting in only 135,159 images for the entire UNSW-NB15 dataset. This corresponds to a new “image” dataset that we can use to train and evaluate our algorithm. Table 4 summarizes the binary class distribution of the images generated by both the CIC IDS2017 and UNSW-NB15 datasets.

The contribution of each type of attack of generated images from CIC IDS2017 dataset is as shown in Figure 5.

We run our model on a server with 2x Intel Xeon Silver 4214 CPUs and 8x NVIDIA RTX 5000 GPUs for both types of classifier architecture (i.e. CNN and ViT). The Tensorflow 2.3.0 framework is used to construct the classifiers. We experiment with binary classification on the RGB image dataset generated from both the CIC IDS2017 and UNSW-NB15 datasets, and multi-class classification on the images generated from only the CIC IDS2017 dataset.

A. BINARY CLASSIFICATION

We perform experiments with binary classification based on two labels of the data: Benign and Malicious. We separated the dataset to three parts: training (70%), testing (20%) and validation (10%), and we run the training for 50 epochs on both the CNN and ViT architecture. The final accuracy and the false positive rate for our proposed model on both the CIC IDS2017 and UNSW-NB15 datasets with an overlapping size of 12 is shown in Table 5 below.

We then compare our results with some of the latest state-of-the-art methods for NIDSs that use machine learning or deep learning, such as BP [38], CNN [25], SVM [39], DBN [38] and DBN-KELM [38]. The comparison of our proposed method with these other methods for the binary classification task is shown in Table 6.

The Graphical comparison accuracy ratio of binary classification is shown in Fig. 9.

From Table 6, we observe that our method using the ViT classifier has outperformed all the other methods in terms of precision and accuracy in both datasets (UNSW-NB15 and CIC IDS2017), except for the accuracy in the CIC IDS2017.
dataset. Our proposed method achieved 98.5% accuracy on the CIC IDS2017 dataset and 96.3% accuracy on the UNSW-NB15 dataset. Although the performance of our proposed method on accuracy in the CIC-IDS2017 dataset is lower than the SVM model [39], it only falls short by 0.4% and is still higher than all the other methods.

B. MULTI-CLASS CLASSIFICATION

We perform experiments with multi-class classification on the CIC IDS2017 and UNSW-NB15 dataset which is balanced. In the CIC IDS2017 dataset, the imbalance of the generated image dataset is quite severe. The Benign and the DoS attack types account for more than 75% of all network flow types.

The multi-class classification on CIC IDS2017 is implemented on nine classes, which are the Patator, Benign, DDoS, DoS, Bot, PortScan, Heartbleed, Infiltration, and Web attack. The performance of the multi-class classification on 20,000 test samples using our proposed method with the ViT classifier on the CIC IDS2017 dataset is shown in Table 7 below. The confusion matrix for multi-class classification on CIC IDS2017 is shown in Figure 10.

Table 8 shows the comparison of our method with other state-of-the-art methods. From the results, we observe that our proposed model achieved 100% precision for both the Patator attack and DDoS attack on the CIC IDS2017 dataset. The Graphical comparison accuracy ratio of multi-class classification is shown in Fig. 11. The accuracy of our proposed
TABLE 7. The performance of the multi-class classification on 20,000 test samples using our proposed method with the ViT classifier on the CIC IDS2017 dataset.

|                      | Patotar | Benign | DDoS | Bot | DoS  | PortScan | Heartbleed | Infiltration | Web attack |
|----------------------|---------|--------|------|-----|------|----------|------------|--------------|------------|
| Number of Samples    | 170     | 8723   | 687  | 67  | 8943 | 2        | 1323       | 2            | 81         |
| Recall               | 99%     | 95%    | 100% | 96% | 98%  | 100%     | 97%        | 0%           | 89%        |
| Precision            | 100%    | 99%    | 100% | 84% | 95%  | 50%      | 98%        | 0%           | 93%        |
| Accuracy             | 100%    | 94%    | 100% | 99% | 94%  | 50%      | 98%        | 0%           | 100%       |

TABLE 8. Comparison with other methods for multi-class classification.

|                      | Patotar | Benign | DDoS | Bot | DoS  | PortScan | Heartbleed | Web attack |
|----------------------|---------|--------|------|-----|------|----------|------------|------------|
| BP [38]              | 75.4%   | 77.6%  | 70.4%| 94.4%| 40.4%| 70.1%    | 80.9%      |            |
| DBN [38]             | 75.9%   | 69.2%  | 80.6%| 64.7%| 73.2%| 82.4     | 84.9%      |            |
| CNN [25]             | 69.5%   | 10.2%  | 82.4%| 49.5%| 66.4%| 80.1%    | 82.1%      |            |
| DBN-KELM [38]        | 77.8%   | 62.4%  | 86.4%| 73.8%| 76.4%| 79.9%    | 90.8%      |            |
| Our method using the ViT classifier | 100%    | 99%    | 100% | 84% | 95%  | 50%      | 89%        | 93%        | 96.4%      |

TABLE 9. The performance of the multi-class classification on 20,000 test samples using our proposed method with the ViT classifier on the UNSW-NB15 dataset.

|                      | Unknown | Benign | Fuzzers | Worms | DoS  | Shellcode | Exploits | Generic | Recon |
|----------------------|---------|--------|---------|-------|------|-----------|---------|---------|-------|
| Number of Samples    | 524     | 9202   | 1601    | 17    | 404  | 138       | 3087    | 3644    | 127   |
| Recall               | 55.30%  | 98.07% | 77.76%  | 98.75%| 52.84%| 61.13%    | 61.29%  | 76.45%  | 83.30%|
| Precision            | 54.72%  | 98.39% | 81.75%  | 57.50%| 50.59%| 64.60%    | 77.05%  | 94.30%  | 88.85%|
| Accuracy             | 55.34%  | 98.07% | 77.76%  | 98.74%| 52.94%| 61.13%    | 61.39%  | 76.45%  | 89.30%|

FIGURE 10. The confusion matrix for multi-class classification on CIC IDS2017.

The multi-class classification on UNSW-NB15 is implemented on nine classes, which are Unknown, Benign, Fuzzers, Worms, DoS, Shellcode, Exploits, Generic, and Recon. Due to this small data size, the resulting training set and test set sizes for the PortScan attack are very small, leading to poor training and testing accuracies. The misclassification errors are shown in the confusion matrix in Figure 10. We can see from Figure 10 that the highest misclassifications are those of DoS being misclassified as Benign, and vice versa. The reason is that at the initial and the end part of a DoS attack, the number of attack packets are relatively low and thus they can be easily mistaken as Benign packet flows. Apart from these two categories, the misclassifications in the other attack categories are all very low.

The multi-class classification on UNSW-NB15 is implemented on nine classes, which are Unknown, Benign, Fuzzers, Worms, DoS, Shellcode, Exploits, Generic, and Recon. Due to this small data size, the resulting training set and test set sizes for the PortScan attack are very small, leading to poor training and testing accuracies. The misclassification errors are shown in the confusion matrix in Figure 10. We can see from Figure 10 that the highest misclassifications are those of DoS being misclassified as Benign, and vice versa. The reason is that at the initial and the end part of a DoS attack, the number of attack packets are relatively low and thus they can be easily mistaken as Benign packet flows. Apart from these two categories, the misclassifications in the other attack categories are all very low.

The multi-class classification on UNSW-NB15 is implemented on nine classes, which are Unknown, Benign, Fuzzers, Worms, DoS, Shellcode, Exploits, Generic, and Recon. Due to this small data size, the resulting training set and test set sizes for the PortScan attack are very small, leading to poor training and testing accuracies. The misclassification errors are shown in the confusion matrix in Figure 10. We can see from Figure 10 that the highest misclassifications are those of DoS being misclassified as Benign, and vice versa. The reason is that at the initial and the end part of a DoS attack, the number of attack packets are relatively low and thus they can be easily mistaken as Benign packet flows. Apart from these two categories, the misclassifications in the other attack categories are all very low.
 Reconnaissance. The performance of the multi-class classification on 20,000 test samples using our proposed method with the ViT classifier on the UNSW-NB15 dataset is shown in Table 9 below.

The accuracy of our proposed method using ViT classifier is also over 11% higher than the next best method, the CNN-BiLSTM model [40].

Although we tried to balance dataset before classifying, UNSW-NB15 dataset still contains more than 80% Benign and Generic requests. The sample number of attack types such as Worms, Shellcode or Unknown (Analysis and Backdoor) is still too small, so their accuracy is impacted seriously.

V. CONCLUSION

The emergence of many new types of cyberattacks has recently led to an increase in demand for new intrusion detection technologies in NIDSS. In this paper, we have proposed a novel method that performs intrusion detection by converting the input networks flows into an RGB image, and then apply the ViT technique to classify that image. To decide what network flow features are important for the classification, we use the decision tree algorithm to compute the importance of each feature. We then select features that have an Information Gain (IG) value of at least 0.001 in at least one type of attack.

Using the features that we have selected from the decision tree algorithm, we extract these features from the input network flows and map them to a color value between 0×000000 to 0xFFFFFF (i.e. 24-bit color). From this color value, we will then extract the Red, Green and Blue components and use them to generate an RGB image from the network flow.

To standardize the size of the image so that we can design a classifier with a fixed number of inputs, we propose a windowing and overlapping technique to extract square images to be fed to the classifier. We then design a classifier based on ViT to classify this image.

Our experimental results have shown that our proposed method outperforms other state-of-the-art algorithms. For the binary classification problem, our proposed method also outperformed the other state-of-the-art methods. Our method has an accuracy of 96.4%, which is 5.6% higher than the next best method, the DBN-KELM [37]. Also, our proposed method achieved 100% precision in two of the object classes, the Patator and DDoS.

For the multi-class classification problem, our proposed method also outperformed the other state-of-the-art methods. Our method has an accuracy of 96.4%, which is 5.6% higher than the next best method, the DBN-KELM [37]. Also, our proposed method achieved 100% precision in two of the object classes, the Patator and DDoS.

However, the data imbalance in the dataset makes the results obtained from the multi-class classification inaccurate. Some types of attacks in the dataset occurs quite infrequently so we do not have enough data to train the classifier correctly. Besides, the feature selection step in our method is also dependent on the data size in each category. If the data size of a category is small, the features that differentiate it from other categories would have a lower information gain and thus be considered less significant. As a result, these features will not get selected by our algorithm and the performance of our network intrusion detection method will be decreased for attacks in these categories. The future work will be focused on how to improve the proposed solution work for imbalance dataset across different categories.

REFERENCES

[1] B. Kelly. (Mar. 6, 2019). *Ninth Annual Cost of Cybercrime Study Unlocking the Value of Improved Cybersecurity Protection the Cost of Cybercrime Contents*.[Online]. Available: https://www.accenture.com/_acmedia/PDF-96/Accenture-2019-Cost-of-Cybercrime-Study-Final.pdf#zoom=50

[2] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, “Network anomaly detection: Methods, systems and tools,” *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 303–336, 1st Quart., 2014.

[3] S. Rahmč, Y. Labit, F. Gouaisbaut, and T. Floquet, “Sliding modes for anomaly observation in TCP networks: From theory to practice,” *IEEE Trans. Control Syst. Technol.*, vol. 21, no. 3, pp. 1031–1038, May 2013.

[4] C. Zhou, S. Huang, X. Xiiong, S. H. Yang, H. Li, Y. Qin, and X. Li, “Design and analysis of multimodel-based anomaly intrusion detection systems in industrial process automation,” *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 45, no. 10, pp. 1345–1360, Apr. 2015.

[5] C. Zhu, W. Sheng, and M. Liu, “Wearable sensor-based behavioral anomaly detection in smart assisted living systems,” *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 4, pp. 1225–1234, Oct. 2015.

[6] M. Mardani and G. B. Giannakis, “Estimating traffic and anomaly maps via network tomography,” *IEEE/ACM Trans. Netw.*, vol. 24, no. 3, pp. 1533–1547, Jun. 2016.

[7] A. G. Tartakovsky, A. S. Polunchenko, and G. Sokolov, “Efficient computer network anomaly detection by changepoint detection methods,” *IEEE J. Sel. Topics Signal Process.*, vol. 7, no. 1, pp. 4–11, Feb. 2013.

[8] A. Gharib, I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, “An evaluation framework for intrusion detection dataset,” in *Proc. Int. Conf. Inf. Sci. Secur. (ICISS)*, Dec. 2016, pp. 1–6.

[9] (Oct. 2007). *KDD Cup 1999.* [Online]. Available: http://kdd.ics.uci.edu/ databases/kddcup99/kddcup99.html
[10] J. McHugh, “Testing intrusion detection systems: A critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory.” ACM Trans. Inf. Syst. Secur., vol. 3, no. 4, pp. 262–294, 2000.

[11] D. E. Difallah, P. Cudre-Mauroux, and S. A. McKenna, “Scalable anomaly detection for smart city infrastructure networks,” IEEE Internet Comput., vol. 17, no. 6, pp. 39–47, Nov. 2013.

[12] H. Alipour, Y. B. Al-Nashif, P. Samat, and S. Hariri, “Wireless anomaly detection based on IEEE 802.11 behavior analysis,” IEEE Trans. Inf. Forensics Security, vol. 10, no. 10, pp. 2158–2170, Oct. 2015.

[13] Y. Zhang, N. Moustafa, and P. Havinga, “Outlier detection techniques for wireless sensor networks: A survey,” IEEE Commun. Surveys Tuts., vol. 12, no. 2, pp. 159–170, 2nd Quart., 2010.

[14] M. Ahmed, A. N. Mahmood, and J. Hu, “A study of network anomaly detection techniques,” J. Netw. Comput. Appl., vol. 60, pp. 19–31, Jan. 2016.

[15] K. Premnabu, B. Purnama, S. Sharipuddin, D. Darmawijoyo, D. Stiawan, S. Samsuryadi, A. Heryanto, and R. Budidarto, “Network anomaly detection research: A survey,” Indonesian J. Electr. Eng. Informat., vol. 7, no. 1, pp. 37–50, Mar. 2019.

[16] T. Vafeiadis, A. Papanikolaou, C. Ilioudis, and S. Charchalakis, “A Realistic Cyber Defense Dataset,” Proc. Mil. Commun. Inf. Syst. Conf. (MilCIS), vol. 79, pp. 1–6, Jun. 2016.

[17] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, “Toward generating a new intrusion detection dataset and intrusion traffic characterization,” in Proc. 8th Int. Symp. Inf. Secur. Defense Appl., vol. 9, pp. 16062–16091, 2021, doi: 10.1109/ACCESS.2021.3051074.

[18] Z. Wang, Y. Zeng, Y. Liu, and D. Li, “Deep belief network integrated kernel-based extreme learning machine for network intrusion detection,” IEEE Access, vol. 9, pp. 16062–16091, 2021, doi: 10.1109/ACCESS.2021.3051074.

[19] M. Lopez-Martin, A. Sanchez-Esguevillas, J. I. Arribas, and B. Carro, “Network intrusion detection based on extended RBF neural network with offline reinforcement learning,” IEEE Access, vol. 9, pp. 153153–153170, 2021.

[20] K. Jiang, W. Wang, A. Wang, and H. Wu, “Network intrusion detection combined hybrid sampling with deep hierarchical network,” IEEE Access, vol. 8, pp. 32464–32476, 2020, doi: 10.1109/ACCESS.2020.2975730.
ZHONGWEN ZHU (Senior Member, IEEE) received the B.Eng. degree in mechanical engineering and the M.Eng. degree in turbomachinery from Shanghai Jiao Tong University, Shanghai, China, and the Ph.D. degree in applied science from the Free University of Brussels, Belgium, in 1996. He joined with the Aerospace Laboratory, National Research Council of Canada (NRC), in 1997. One year later, he started to work at Bombardier Aerospace. Since 2001, he has been working at Ericsson Canada Inc., with different roles, e.g., a Data Scientist, a SW Designer, the System Manager, a System Designer, a System/Solution Architect, a Product Owner, and the Product Manager. His current research interests include 5G networks, network security, edge computing, reinforcement learning, machine learning for 2D/3D object detections, and the IoT (ultra-reliable low latency application). He was an Associate Editor of Journal of Security and Communication Network (Wiley). He was the invited Technical Committee Member of International Conference on Multimedia Information Networking and Security. He was a recipient of the Best Paper Award IEEE International Conference on Signal Processing and Multimedia Applications, in 2008.

SARANG ARAVAMUTHAN received the B.Tech. degree in electrical engineering from IIT, Chennai, in 1987, and the Ph.D. degree in applied mathematics from Cornell University, in 1995. He has several years of industry experience in cryptography and data science. Prior work experiences include stints in Microsoft, Google, and TCS. His current research interests include machine learning and cryptography and their applications to the telecommunication industry.