DDoS Attack Detection for Vehicles-to Infrastructure-Communication Using a Combination Scheme of CNNs and Decision Tree

Salma Elgendy
Faculty of Engineering, School of Electrical and Computer Science
University of Ottawa
Cairo, Egypt
selge047@uottawa.ca

Mayar Attawiya
Faculty of Engineering, School of Electrical and Computer Science
University of Ottawa
Cairo, Egypt
matta057@uottawa.ca

Omar Haridy
Faculty of Engineering, School of Electrical and Computer Science
University of Ottawa
Cairo, Egypt
ohari043@uottawa.ca

Ahmed Farag
Faculty of Engineering, School of Electrical and Computer Science
University of Ottawa
Cairo, Egypt
afara109@uottawa.ca

Prof. Paula Branco
Faculty of Engineering, School of Electrical and Computer Science
University of Ottawa
Ottawa, Canada
pbranco@uottawa.ca

***Abstract*— Distributed Denial Service (DDoS) attacks became the most widely spread attack due to their ease of design and execution. Such attacks are challenging to detect and mitigate due to the diversity of DDoS attack modes and the variable size of attack traffic. This makes the research on DDoS attack detection become extremely important. Machine learning techniques are used to detect various DDoS attacks with complex and dynamic patterns. However, such techniques require doing extensive pre-processing and feature engineering to the data to achieve acceptable results. On the other hand, deep learning techniques can achieve acceptable results without the need for such prior preparations.**

This paper proposes a combination scheme between deep learning models, to benefit from their ability to extract features without the need to prior preparations, and machine learning models, that have high performance on similar problems. The solution proposed uses two convolutional neural networks (CNNs) to classify between 10 types of DDoS attacks and uses their prediction results to enhance the performance of one machine learning model on the same classification task. The results of the experiments taken by this study show that the proposed solution can score 20.5% accuracy higher than the classification results of the machine learning model on its own.

Keywords— distributed denial of service (DDoS), machine learning, deep learning, convolutional neural networks (CNN)

I. INTRODUCTION

Distributed Denial of Service (DDoS) attacks are one of the major cyber threats on communications networks and have dramatically increased over the past years. The high frequency of DDoS attacks is because they are simple and cheap to initiate but at the same time, their impact on the victim can be potentially severe. These attacks overwhelm the victim and make it unable to provide its services normally [1].

One of the examples of DDoS attacks is in the field of autonomous and connected vehicles that share information with vehicles and infrastructures using wireless communication. The risk of attacks increases, as vehicles become more connected through wireless networks. These networks allow the vehicle to infrastructure communication (V2I). Attacks on V2I communication can have serious consequences if the systems are not properly secured. V2I applications have a lot of vulnerabilities that attract hackers [2]. One example of V2I communication is that applied in the autonomous parking in smart garages, which relies on the vehicle to garage communication (V2G) putting the whole system under the risk of potential DDoS attacks through Wi-Fi. Attacks on autonomous vehicles are very dangerous as the car might become blind in the presence of pedestrians or other vehicles leading to huge damages.

Detection of these attacks is one of the main defense mechanisms. The traditional security solutions such as firewalls and intrusion detection systems are unable to detect the complex DoS and DDoS attacks; most of these solutions filter the normal and suspicious traffic based only upon the static predefined rules [3]. Using artificial intelligence-based solutions have enabled researchers to detect DDoS attacks with complex and dynamic patterns.

Machine learning is used widely in detecting DDoS and DoS attacks. Many machine learning methods such as Naïve Bayes, Decision trees, and K-Nearest Neighbors were used to detect DDoS and DoS attacks. Decision Trees (DT) has also been used in some of the recent works to detect these attacks [4]. DT is either used on its own or in combination with other models [5]. However, machine learning models need extensive pre-processing, feature engineering, and prior preparations. On the other hand, deep learning doesn't require feature engineering since it can achieve good performance in classification by directly feeding the data into the network.

Selecting the dataset is a crucial step to ensure the good performance of the selected model. Some of the publicly available datasets that fit the classification problem was reviewed. The DDoS Evaluation Dataset from the Canadian Institute for Cybersecurity (CICDDoS2019) was selected.

The research team found that there is still room for improvement in the detection performance reported in the state-of-the-art section, the proposed solution builds on current detection models available making a combination between deep learning which has high performance, and machine learning models that can be tested to find the best models meeting the requirements resulting in a new detection model to achieve effective detection of DDoS attacks. For deep learning, the state-of-the-art in [3] and [8] will be used to

build a model using convolutional neural networks (CNN). For the machine learning part, tuned Decision Tree will be used as it has shown great performance in solving similar problems.

II. Related Work

Machine learning is used widely these days in detecting attacks. Using machine learning-based solutions have enabled researchers to detect DDoS attacks with complex and dynamic patterns. Traditional solutions like firewalls are no longer able to detect complex DoS and DDoS attacks.

NN plus biological danger theory was used in [9] to reduce DDoS attacks in Software Defined Network (SDN). In [10] and [11], authors used radial basis function support vector machine (RBF-SVM) and linear classification SVM for DDoS detection; the accuracy rates in these studies reached 95.11% and 97.60%, respectively. The authors of [12] reached accuracy of 97.6% with a false positive rate of 3.85% using multiple linear SVM + SOM. Also, in [13] they used several machine learning techniques to reach high accuracies which are decision tree and support vector machine with a linear kernel, and they were comparing their work to [14] in which gradient boosting was used "XGBOOST", using CICIDS2017 and a sort of feature engineering, they reached accuracy around 100%. As a result, it can be deduced that the detection effect of machine learning is more successful than traditional methods. However, it always needs feature engineering and extensive prior preparations.

Deep learning, on the other hand, doesn't require feature engineering. Good performance can be achieved by directly putting data into the network and deep learning in many fields has been more efficient than traditional machine learning. The first paper discussed for their state-of-the-art in deep learning uses ResNet in IoT DoS and DDoS detection [3]. It is also crucial to mention that they are working on the same dataset chosen to apply the proposed solution "CICDDoS2019". During the last years, deep learning, especially convolutional neural networks (CNN), made high accomplishments in the image processing field so the authors concluded that the potential of using CNNs in DDoS attacks detection will be efficient by converting the network traffic data into three-channel image form. They proposed a methodology to convert the network traffic data into image form and trained a state-of-the-art CNN model, i.e., ResNet over the converted data.

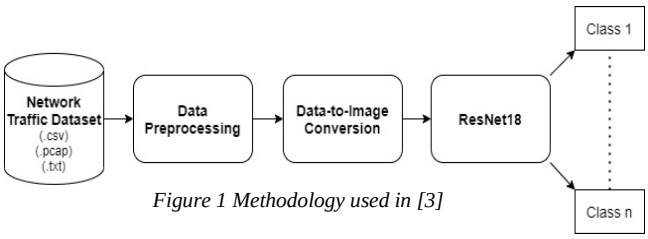

Figure 1 Methodology used in [3]

For the Data Preprocessing, the tabular data must be cleaned, the records with missing data are dropped, columns of high correlation and with malformed values are dropped to be ready to be converted into the image shape, after the preprocessing stage, the dataset ended up with 60 features.
For the Data Conversion, each feature is normalized, and the dataset is separated into two data frames, attack and normal. Data frames are divided into chunks of 180 records, each record consists of 60 pre-processed features and are finally converted into a 60x60x3 image shape to be finally fed to the ResNet18 for classification.
Using this methodology, they reached 99.9% accuracy and 87% average precision recognizing eleven types of DoS and DDoS attack patterns which is 9% higher as compared to the state-of-the-art as shown in Table 1.

Table 1 - COMPARISON BETWEEN THE SOLUTIONS IN [3] AND [15]

Method	Precision	Recall	F1-Score
Sharafaldin et al. [15]	**78%**	**65%**	**69%**
Faisal Hussain et al. [3]	**87%**	**86%**	**86%**

For the latest paper using the same dataset "CICDDoS2019", Wang H et al. designed a hybrid neural network DDosTC structure, combining efficient and scalable transformers and a CNN to detect DDoS attacks on software-defined networking (SDN), tested on the dataset "CICDDoS2019" [16]. They conducted several experiments on different state-of-the-art deep learning models in the field of DDoS intrusion detection. The experimental results show that the average AUC of DDosTC is 2.52% higher than the current optimal model and that DDosTC is more successful than the current optimal model in terms of average accuracy, average recall, and F1 score. Their solution was to use deep learning transformers plus CNNs DDoS attack-detection model "DDosTC". The detection module is implemented on the control plane, as it is considered the brain of the SDN. For the transformer, it is composed of an encoder and decoder. Each encoder block has two layers: multi-head attention and feed-forward. A decoder has three layers: multi-head attention, multi-head attention (encoder-decoder attention), and feed-forward. And finally, a dense layer using a sigmoid function.

Based on the accurate results obtained by using Machine Learning models such as RBF-SVM in [10] and [11], DT in [4] and [5], that needs extensive feature engineering, and the novel CNN-based method in [3] and [16] that converts the network traffic tabular data into images, the proposed solution is designed to form a combination scheme between both DT and CNNs to eliminate the need for feature engineering, in addition to, enhancing the state-of-the-art accuracy obtained by the DTs for solving DDoS attacks Detection problem.

The proposed combination scheme is to use several CNNs to get the probability of occurrence of each attack, then use the output of each CNN to form feature vectors that can be concatenated as feature columns to the original tabular network traffic dataset, and then feed the modified dataset to the Decision Tree Classifier.

III. Proposed Methodology

The proposed method is to combine deep learning and machine learning models to benefit from the advantages of each. First off, CNN will be used and since it is a deep learning model it does not require feature extraction to be done before feeding the data to it. CNN is specifically chosen to take advantage of its great performance on image datasets, by converting the chosen dataset to be 3-channel images [3]. The solution consists of two stages, stage 1 is the part that includes

converting the traffic data to images to be fed to the selected CNNs to form new feature vectors, that will be used then by stage 2. Stage 2 starts with concatenating the new features generated by the two CNNs to the original dataset; hence, a new modified dataset is formed that will then be fed to the Decision Tree to output the final classification prediction. The two selected CNN models are Efficientnet and Xception which will be used for stage 1 of the solution.

This section is divided to three main topics: Data acquisition, Data preprocessing, and Intrusion detection at which these modules are thoroughly explained.

A. Data Acquisition

Innovation in networking and wireless systems has indeed opened new ways for threats to system's security. To be able to detect DDOS attacks through Wi-Fi, numerous data has to be collected and lots of experiments must be done which will be time and money consuming. Instead of this inefficient method, there are lots of publicly available datasets for network intrusions that can be quite beneficial. Selecting the dataset is a crucial step to ensure the good performance of the selected model. The research team reviewed some of the publicly available datasets that fit the problem. The DDoS Evaluation Dataset from the Canadian Institute for Cybersecurity (CICDDoS2019) was selected since it consists of real-time network traffic, is extensive and versatile, and contains both inbound and outbound traffic of the latest DoS and DDoS attacks. There are different modern reflective DDoS attacks in this dataset including PortMap, NetBIOS, LDAP, MSSQL, UDP, UDP-Lag, SYN, NTP, DNS, and SNMP; this dataset contains six different types of DDoS attacks and others resulting in 11 different attack types in total plus normal traffic, so 12 classes in total. More about the dataset can be found in [6] and [7].

B. Data Preprocessing

After getting the data, it must be cleaned from unwanted samples and features. It also must be prepared for the CNN models.

1) Taking a Subset

CICDDoS2019 dataset is one of the most well-designed datasets. It contains numerous amounts of data which makes it very efficient, but on the other hand, it is extremely large "~20GB" which makes it hard to fit a model using the entire dataset using the available hardware. To solve this problem, a subset is taken of the dataset by taking 100,000 samples from each attack type, all the benign data available, and discarding "TFTP" data which was 9 GB alone. As a result, the dataset that will be used in this paper has 11 classes in total.

2) Data Cleaning

The CICDDoS2019 dataset contains 88 feature columns but some of these features are unimportant like IPs and time of extracting the data. Those features were Flow ID, Source IP, Source Port, Destination IP, Destination Port, Protocol, and Timestamp. Using correlation with the labels, an insight of how much every feature contributes to the resulting label will be known. The feature which shows a very small correlation value was dropped. In addition, after visualizing the data, it appeared that some features have only zero or constant values, so they were also dropped. Features that were dropped because of small correlation or zero values are BwdPSHFlags, FwdURGFlags, BwdURGFlags, FINFlagCount, PSHFlagCount, ECEFlagCount, FwdAvgBytes/Bulk, FwdAvgPackets/Bulk, FwdAvgBulkRate, BwdAvgBytes/Bulk, BwdAvgPackets/Bulk, and BwdAvgBulkRate. There was a feature containing some characters although it was numerical not categorical, it was "SimillarHTTP", and it was dropped too. Finally, after the data cleaning process, 66 unique and important features remained. After that, the samples containing noisy values were dropped, such as NaNs and inf. Encoding was applied to 'Label' column to convert it into numerical type. The final number of samples after preprocessing is 983484 data samples.

3) Feature Extraction

To further improve the accuracy of the DT model, a linear feature extraction algorithm was applied. Feature Extraction reduces the number of features in a dataset by creating new features from the existing ones. In other words, feature extraction algorithms create a summarized version of the original features that can be used to train the model, instead of the original dataset, to help improve the performance of the model. Linear feature extraction methods perform a linear mapping of the data to a lower-dimensional space. The feature extraction algorithm used is Principal Component Analysis (PCA). PCA is an unsupervised algorithm that finds the direction of maximum variation in data, to obtain important features (in the form of components) from a large set of features available in a dataset, and minimizing the reconstruction error by looking at pair wised distances. In PCA, data is projected into a set of orthogonal axes and each of the axes gets ranked in order of importance. PCA was used with 20 components which means the 66 features were further reduced into only 20 new features extracted by PCA.

4) Data Conversion

As mentioned above, CNNs have showed great performance throughout the years on images, so it will be an added value to use CNNs in intrusion detection modules. However, there is a challenge to benefit from CNNs in intrusion detection problems, which is the fact that CNNs require its input in image format while the dataset is in csv format. Since the data is in csv format, it has to be converted to RGB images to be committable with CNNs. To achieve this, the pipeline illustrated in fig. 2 was used. First, the dataset was sorted the by labels so that the data is in sequence where the samples of each class are after one another. Then, the feature columns are normalized using min-max normalization by the means of the equation below (1).

$$X' = \frac{X - \min(X)}{\max(X) - \min(X)} * 255 \quad (1)$$

After that, the samples need to be arranged in image format. Since the dataset has 66 features, 66 samples are required to form one image channel of size 66x66. And since the selected CNN models require RGB images, two more channels need to be created, which means 198 samples from the same class in total will be needed to form an image of size 66x66x3. From the sorted dataset, a chunk of 198 samples is taken from the same class to form each image. Since there are 983484 samples in the original dataset which will be divided into chunks of 198 samples, a total of 4796 images will be formed. Fig. 2 and 3 show samples of the formed images.

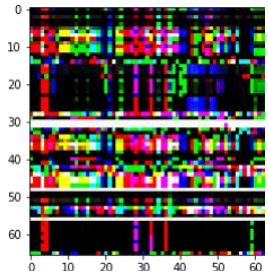

Figure 2 Image created from BENIGN class

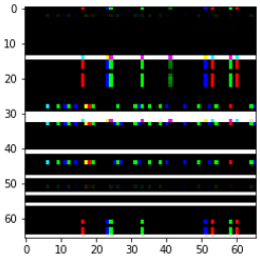

Figure 3 Image created from DDoS_MSSQL class

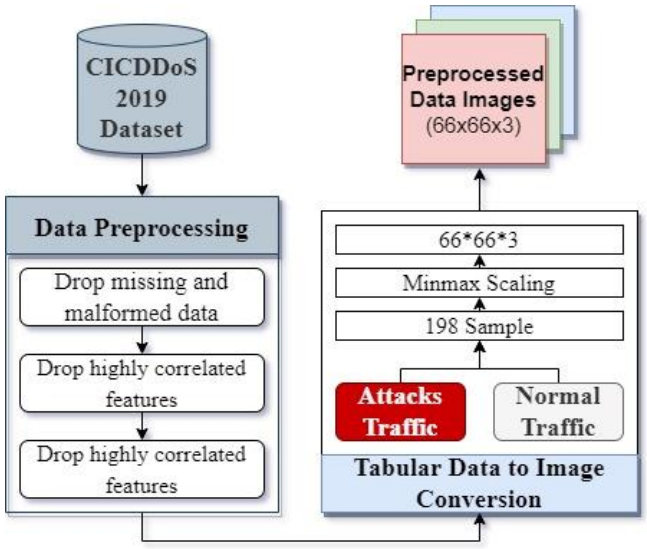

Figure 4 - Data preparation pipeline

C. Intrusion Detection

1) Machine Learning Model Only

As stated above, Decision Tree is selected as the machine learning model. Decision tree work by breaking down the data through making decisions based on multiple questions at each level. This makes it suitable for multiclass classification and work fine in intrusions detection. Compared with other machine learning algorithms, DT is one of the quickest ways to identify relationships between variables and the most significant variable. Data preparation during pre-processing in a decision tree requires less effort and does not require normalization or scaling of data. First step in the proposed methodology is to use the DT only with the original data samples as the base-line model, to compare it with the proposed solution. The parameters used for the DT model are listed in table 2.

Table 2 DT PARAMETERS

Parameter	**Value**
n_estimators	100
criterion	"gini"
max_depth, max_leaf_nodes, n_jobs, random_state, class_weight, max_samples	None
min_samples_split	2
min_samples_leaf	1
min_weight_fraction_leaf,ccp_alpha, min_impurity_decrease	0.0
max_features	"auto"
bootstrap	True
oob_score, warm_start	False
verbose	0

Before applying DT, Principal Component Analysis (PCA) feature extraction algorithm is applied to the original dataset. By using PCA, feature data are simplified to 20 features instead of 66 in the original dataset.

2) CNNs Only

CNNs were selected with the goal of achieving the highest accuracy on the required classification task. The CNN models selected were chosen for their capability of extracting features efficiently, each by its own unique method, with suitable number of parameters that allows the CNN to be efficiently trained on the available hardware.

The first CNN model is EfficientNet-B0, a mobile-sized CNN model. EfficientNet-B0 was built with a goal to optimize both accuracy and floating point operations (FLOPS) represented as: $ACC(m)x(FLOPS(m)\ /\ T)^{\omega}$, where $ACC(m)$ and $FLOPS(m)$ denote the accuracy and FLOPS of model, T is the target FLOPS, and $\omega = -0.07$ is a hyperparameter responsible for the trade-off between accuracy and FLOPS [17]. The architecture of EfficientNet-B0 is similar to MnasNet however, EfficientNet-B0 is slightly bigger due to the larger FLOPS target. EfficientNet-B0 architecture is shown in Table 3.

Table 3 EFFICIENTNET-B0 NETWORK ARCHITECTURE

Stage i	**EfficientNet-B0 Architecture**			
	Operator $\hat{F}_i$	***Resolution*** $\hat{H}_i x \hat{W}_i$	***#Channels*** $\hat{C}_i$	***#Layers*** $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

EfficientNet-B0 requires input images of size 224×224×3. As a result, the output images from the data conversion module need to be resized from 66×66×3 to 224×224×3. EfficientNet has more versions other than EfficientNet-B0, such as EfficientNet-B1 to EfficientNet-B7. However, the other versions require larger input sizes starting from 240×240×3 to 600×600×3. Due to hardware limitations, EfficientNet-B0 was selected as it requires the smallest input size and has the least number of parameters (5,330,571 parameters). The convolutional layers' structure of EfficientNet-B0 is used unchanged however, several layers were added to make the model fit the requirements of the current classification problem. The additional layers are shown in table 4.

Table 4 ADDED LAYERS TO EFFICIENTNET-B0 ARCHITECTURE

Stage i	**Added Layers to EfficientNet-B0 Architecture**	
	Operator $\hat{F}_i$	***Corresponding Parameter***
1	Dropout regularization	Rate = 0.2
2	Fully Connected Layer (Dense)	Output neurons = 11, Activation = Softmax
3	L2 regularization	Rate = 0.0001

All parameters of EfficientNet-B0 are retrained on the dataset using stochastic gradient descent (SGD) optimizer and

categorical cross entropy loss function. Learning rate was empirically selected to be equal to 0.001.

The second CNN model is Xception, a CNN model inspired by Inception modules. Inception modules are used in CNN to allow for more efficient computation and deeper networks through a dimensionality reduction with stacked 1×1 convolutions, which makes the CNN capable of learning richer representations with less parameters. These modules work by ordering multiple kernel filter sizes to operate on the same level (in parallel), rather than stacking them sequentially. Xception is based on an 'extreme' version of the Inception architecture; thus, its name stands for 'Extreme Inception'. This 'extreme' version of an Inception module first uses a 1×1 convolution to map cross-channel correlations, and then separately map the spatial correlations of every output channel. In other words, Xception is a CNN based entirely on depthwise separable convolution layers and it makes the hypothesis that the mapping of cross-channels correlations and spatial correlations in the feature map of convolutional neural networks can be entirely decoupled. This CNN architecture contains 22,910,480 parameters. More details about Xception and its architecture can be found in its original paper [18]. The convolutional layers' structure of Xception is used unchanged however, several layers were added to make the model fit the requirements of the current classification problem. The additional layers are shown in table 5.

Table 5 ADDED LAYERS TO XCEPTION ARCHITECTURE

Stage i	Added Layers to EfficientNet-B0 Architecture	
	Operator $\hat{F}_i$	***Corresponding Parameter***
1	Fully Connected Layer (Dense)	Output neurons =2048, Activation= Relu
2	Batch normalization	-
3	Dropout regularization	Rate = 0.2
4	Fully Connected Layer (Dense)	Output neurons =2048, Activation= Relu
5	Batch normalization	-
6	Dropout regularization	Rate = 0.2
7	Fully Connected Layer (Dense)	Output neurons =2048, Activation= Relu
8	Fully Connected Layer (Dense)	Output neurons =11, Activation= Softmax

All parameters of Xception are retrained on the dataset using Adam optimizer and categorical cross entropy loss function. Learning rate was empirically selected to be equal to 0.0001. The tuning of the hyperparameters of the additional layers of both models were done by first setting the number of neurons of both models empirically such that both models overfit. After that, regularization techniques such as drop out, L2 and batch normalization are used to tune the models and obtain acceptable results.

3) Combination Scheme

Finally, taking advantage of the great performance of the DT classifiers on this type of problem, the output of the several CNN models, columns of probabilities for each model, will be concatenated as feature vectors to the original dataset and the DT model will be trained on the new dataset. Fig. 5 and 6 provide an overview of the proposed methodology.

Now is the time to make use of the sorted images data, the original dataset, the machine learning model, and the CNNs to apply the proposed solution. As illustrated in fig. 5, EfficientNet-B0 is trained with the converted images, while saving the current indices before splitting to training and validation, then we take the probabilities produced from the prediction using the validation set and getting the predictions again but using the training set this time. After that, the two sets of probabilities are sorted with the indices saved before; in order to have them with the same order of the original dataset "the 983484 samples". Then, each vector of the sorted probabilities is repeated 198 times, because each image consists of 198 samples as discussed earlier. Finally, these probabilities are concatenated to the original features creating a new feature set of size 983484 x 77. Repeat the stated steps again but with the Xception model. The result is a feature set of size 983484 x 88. The last step is to split this new dataset into training and testing and train the decision tree.

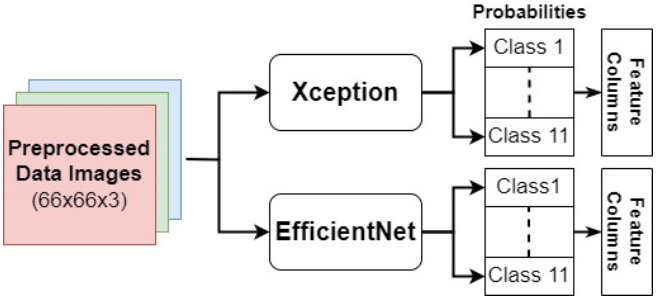

Figure 2 - Stage 1 of the classification pipeline (CNN Training)

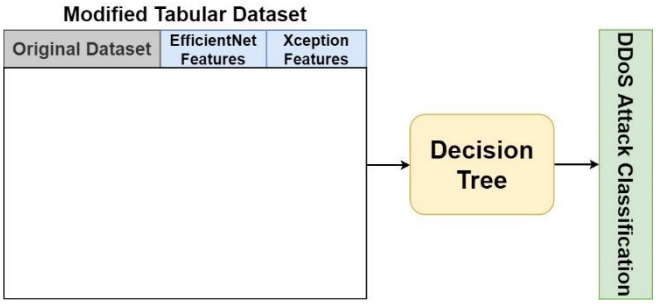

Figure 3 – Stage 2 of classification pipeline: Features generated by the two CNN models (EfficientNet and Xception) are concatenated to the original dataset to be fed to Decision Tree Classifier

Finally, the performance of the proposed methodology will be compared to the performance of an DT model trained on the original dataset.

IV. EXPERIMENTS AND RESULTS

This section shows the results of implementing the proposed solution. All experiments took place on Tesla P100 and 25 Giga RAM. The metrics used to evaluate the models' performance are accuracy, precision, recall, and F2 scores. Accuracy is the ratio between the correctly predicted samples and the total number of samples, Precision quantifies the number of positive class predictions that belong to the positive class. Recall quantifies the number of positive class predictions made from all positive examples in the dataset. F1 score is the harmonic mean of precision and recall. Since the classification problem contains 11 class as described earlier, precision, recall, and f1 scores are calculated using the weighted average which calculates these scores for each class independently, multiply the score of each class by a weight that depends on the number of true labels of that class in the dataset, and finally calculates the average score of all classes. The subset taken from the dataset, described in section III is very slightly imbalanced (almost balanced) which means that

accuracy will still be a useful metric in comparing the performance of the models. Precision, recall, and F1 score will also give an indication of the weakness spots in each model. Since the recall score is affected by the number of positive samples (attacks) being misclassified, solving this classification problem requires having the recall score as high as possible.

Since the subset of the dataset is relatively large, the holdout method was used to estimate the models' performance, where the subset is split into 80% training and 20% testing.

This study's experiments are divided into 1. Experimenting with machine learning models 2. Experimenting with deep learning models and 3. Experimenting with combination scheme.

1) Experimenting with DT

The dataset was cleaned, as described in section III, and split into 80% training data and 20% testing data. PCA feature extraction technique was applied to the cleaned data to form a new dataset, as described in section III. Decision tree is then trained and tested on this new dataset resulting in training accuracy of 83.6% and testing accuracy of 79%. The weighted average scores of precision, recall, and F1 for the testing dataset are 81%, 80%, and 79% respectively. Table 6 displays the precision, recall, and F2 scores of each class.

Table 6 DETAILED SCORES OF BASELINE MODEL

Label: Class	**Precision**	**Recall**	**F1-score**
0: BENIGN	0.99	0.99	1.00
1: DrDoS_DNS	0.91	0.88	0.90
2: DrDoS_LDAP	0.82	0.74	0.79
3: DrDoS_MSSQL	0.96	0.95	0.96
4: DrDoS_NTP	0.99	1.00	1.00
5: DrDoS_NetBIOS	1.00	1.00	1.00
6: DrDoS_SNMP	0.8	0.91	0.86
7: DrDoS_SSDP	0.64	0.56	0.60
8: DrDoS_UDP	0.61	0.69	0.65
9: Syn	0.52	0.94	0.66
10: UDP-lag	0.67	0.13	0.21

2) Experimenting with CNNs

The cleaned data (before splitting) was converted into images to be fed to the CNN models, as described in section III. After that, the formed images were split into 80% training and 20% testing. EfficientNet-B0 was trained and tested on the converted dataset which resulted in 94% testing accuracy. The weighted average scores of precision, recall, and F1 for the testing dataset are 94%, 94%, and 94% respectively. Table 7 displays the precision, recall, and F2 scores of each class.

Table 7 DETAILED SCORE OF EFFICIENTNET-B0 MODEL

Label: Class	**Precision**	**Recall**	**F1-score**
0: BENIGN	1.00	1.00	1.00
1: DrDoS_DNS	0.99	1.00	1.00
2: DrDoS_LDAP	1.00	1.00	1.00
3: DrDoS_MSSQL	1.00	1.00	1.00
4: DrDoS_NTP	1.00	1.00	1.00
5: DrDoS_NetBIOS	1.00	1.00	1.00
6: DrDoS_SNMP	1.00	0.99	0.99
7: DrDoS_SSDP	0.55	0.85	0.67
8: DrDoS_UDP	0.80	0.46	0.58
9: Syn	0.93	0.99	0.96
10: UDP-lag	0.99	0.92	0.95

Fig. 7 and fig. 8 show the learning curves of EfficientNet-B0. It can be observed that as the model trains on more epochs, its training and testing accuracies and losses become closer in value.

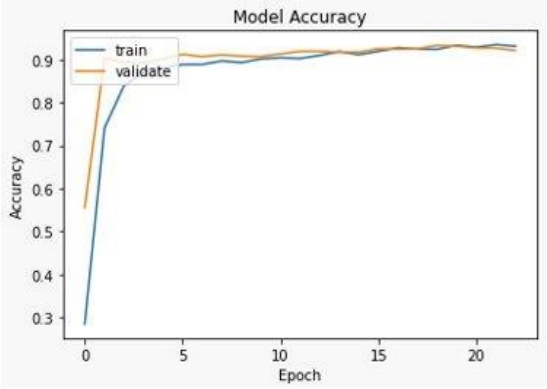

Figure 7 EfficientNet-B0 accuracy learning curve

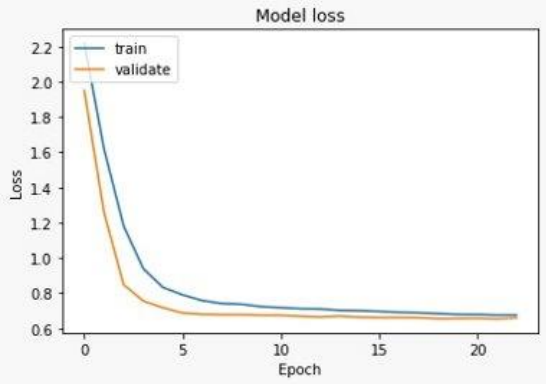

Figure 8 EfficientNet-B0 loss learning curve

Xception was also trained and tested on the converted dataset which resulted in 91% testing accuracy. The weighted average scores of precision, recall, and F1 for the testing dataset are 92%, 91%, and 91% respectively. Table 8 displays the precision, recall, and F2 scores of each class.

Table 8 DETAILED SCORES OF XCEPTION MODEL

Label: Class	**Precision**	**Recall**	**F1-score**
0: BENIGN	1.00	0.97	0.99
1: DrDoS_DNS	0.99	0.98	0.99
2: DrDoS_LDAP	0.99	1.00	1.00
3: DrDoS_MSSQL	1.00	0.99	0.99
4: DrDoS_NTP	1.00	1.00	1.00
5: DrDoS_NetBIOS	1.00	0.99	0.99
6: DrDoS_SNMP	0.99	1.00	0.99
7: DrDoS_SSDP	0.55	0.80	0.65
8: DrDoS_UDP	0.76	0.51	0.61
9: Syn	0.92	0.88	0.90
10: UDP-lag	0.88	0.92	0.90

Fig. 9 and fig. 10 show the learning curves of Xception. It can be observed that as the model trains on more epochs, its training and testing accuracies and losses become closer in value.

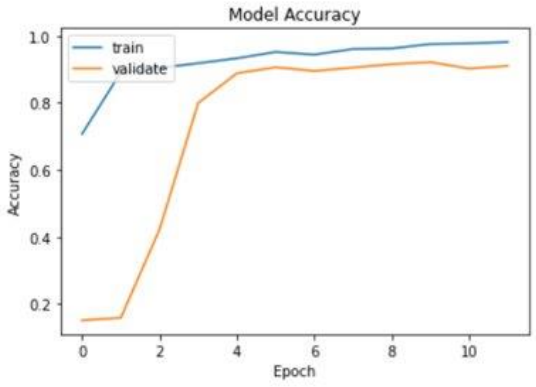

Figure 9 Xcpetion accuracy learning curve

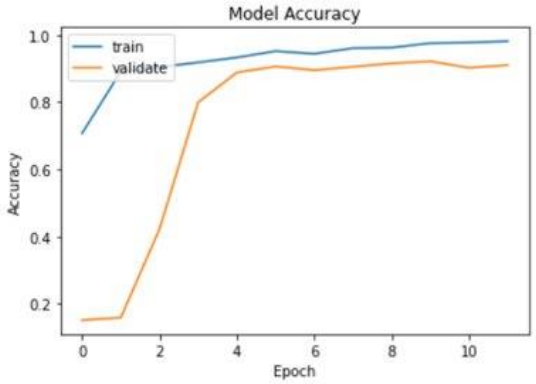

Figure 10 Xcpetion loss learning curve

3) Experimenting with combination scheme

The probabilities estimated by EfficientNet-B0 and Xception were saved and concatenated with the original cleaned data as additional features. Decision tree is then trained and tested on the new dataset and resulted in 100% training accuracy and 99.9 % testing accuracy. The weighted average scores of precision, recall, and F1 for the testing dataset are 99.9%, 99.9%, and 99.9% respectively. Table 9 displays the precision, recall, and F2 scores of each class.

Table 9 DETAILED SCORES OF COMBINATION SCHEME

Label: Class	**Precision**	**Recall**	**F1-score**
0: BENIGN	1	1	1
1: DrDoS_DNS	1	1	1
2: DrDoS_LDAP	1	1	1
3: DrDoS_MSSQL	1	1	1
4: DrDoS_NTP	1	1	1
5: DrDoS_NetBIOS	1	1	1
6: DrDoS_SNMP	1	1	1
7: DrDoS_SSDP	1	1	1
8: DrDoS_UDP	1	1	1
9: Syn	1	1	1
10: UDP-lag	1	1	1

V. DISCUSSION OF RESULTS

The results of DT on the training and testing data extracted by PCA shows that the model did not overfit the data but still needs considerable improvement to efficiently detect the attacks. This performance may be the result of the inability of the decision tree to distinguish between similar attacks, or to identify classes with complex features, resulting in low F1 scores to some classes. DT classified classes 6 to 10 with F1 scores less than 90% where the lowest F1 score was 21%.

On the other hand, CNNs showed better results in classifying the attacks. The learning curves of EfficientNet-B0 show that the model is performing with acceptable results on the training data and is generalizing well on the testing data. However, EffieicentNet-B0 had difficulties in distinguishing some classes, such as classes 7 and 8, where it scored F1-scores below 90% and the lowest F1 score was equal to 58%. Xception also showed better results than the baseline model and its learning curves show that the model is performing with acceptable results on the training data and generalizing well on the testing data. However, after a certain epoch value, its generalization gap starts increasing, so early stopping has to be applied to avoid overfitting. Xception also had some difficulties in classification of classes 7 and 8, where it scored F1-scores below 90% and the lowest F1 score was equal to 61%.

The combination scheme used the strength points of DT and of the CNN models to classify the attacks resulting in 100% F1-score in classifying all classes. Such performance proves the success of combining machine learning and deep learning models in classification problems generally, and DDoS attacks classification problems specifically.

CNN models were able to extract the important features efficiently from the images which made it easier for the DT to break down the data, make decisions, and classify the attacks regardless of how similar or different these attacks were and regardless of the complexity of the attack's features.

The performance of DT baseline model can be enhanced to match that of the combination scheme as discussed in section II however, this will require tremendous efforts to construct feature engineering procedures that extract only the most important and relevant features from the samples in order to enable the DT model to classify the attacks with acceptable results. This step was not necessary in the combination scheme since the CNN models were able to efficiently extract the important features needed to improve the performance of DT.

However, the proposed solution depends on converting the data samples into images, which requires sufficient computational resources and may cause some latencies. In addition, the data conversion may cause some delays if the solution is tested on a real-time data stream. The solution also is developed to work only on the given set of features and does not automatically adapt to new features or new attack types.

VI. CONCLUSION

The combination scheme of the Machine Learning model (DT) and the Deep Learning Models (EfficientNet, Xception) has outperformed each of them when applied on the dataset independently, as well as the baseline model. Using the proposed method expands the room for enhancements and improvements for any tabular data machine learning applications. Converting the structured data into image-like shapes permits harnessing the feature extraction capabilities of the CNNs. Feeding the generated descriptive features to well-known ML models is proved to enhance the prediction capability of the model. Moreover, the expert knowledge needed to extract features from the raw data to be then fed to the ML model is not as crucially required as before, since the

introduced CNN automatically extracts the most important features, and allows a large room for parameter tuning to tailor the extraction capabilities to the dataset. The proposed model requires sufficient computational resources for training the model, however, the inference speed is no slower than the conventional methods. In addition, the adaptive capabilities of the model do not need periodic retraining on a whole modified dataset, incremental training would suffice. Harnessing the privileges of both ML and DL models allowed the improvement of the prediction capability.

VII. Future work

Upgrading the processor hardware and increasing the size of RAM memory will enable the model to run directly on the original dataset without having to work only on a subset. This way the model can be trained on more diverse cases of each attack type. The proposed solution can be further developed to fit real-time data streams and to be able to adapt to changing features and new types of DDoS attacks. This way the model will be ready to be tested on real-time data coming from V2I applications among others.

References

[1] Islam, M., et al. ”Cybersecurity Attacks in Vehicle-to-Infrastructure.” (2018).

[2] Semerci, M., Cemgil, A. T., & Sankur, B. (2018). An intelligent cyber security system against DDoS attacks in SIP networks. Computer Networks, 136, 137–154. doi:10.1016/j.comnet.2018.02.025

[3] F. Hussain, S. G. Abbas, M. Husnain, U. U. Fayyaz, F. Shahzad and G. A. Shah, "IoT DoS and DDoS Attack Detection using ResNet," 2020 IEEE 23rd International Multitopic Conference (INMIC), 2020, pp. 1-6, doi: 10.1109/INMIC50486.2020.9318216.

[4] Lakshminarasimman, S. & Ruswin, S. & K., Sundarakantham. (2017). Detecting DDoS attacks using decision tree algorithm. 1-6. 10.1109/ICSCN.2017.8085703.

[5] G. Lucky, F. Jjunju and A. Marshall, "A Lightweight Decision-Tree Algorithm for detecting DDoS flooding attacks," 2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C), 2020, pp. 382-389, doi: 10.1109/QRS-C51114.2020.00072.

[6] DDoS Evaluation Dataset (CIC-DDoS2019) https://www.unb.ca/cic/datasets/ddos-2019.html

[7] Iman Sharafaldin, Arash Habibi Lashkari, Saqib Hakak, and Ali A. Ghorbani, "Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy", IEEE 53rd International Carnahan Conference on Security Technology, Chennai, India, 2019

[8] Wang H, Li W. DDosTC: A Transformer-Based Network Attack Detection Hybrid Mechanism in SDN. Sensors (Basel). 2021;21(15):5047. Published 2021 Jul 26. doi:10.3390/s21155047.

[9] Mihai-Gabriel I., Victor-Valeriu P. Achieving DDoS resiliency in a software defined network by intelligent risk assessment based on neural networks and danger theory; Proceedings of the IEEE 15th International Symposium on Computational Intelligence and Informatics (CINTI); Budapest, Hungary. 19–21 November 2014; pp. 319–324.

[10] Kokila R.T., Selvi S.T., Govindarajan K. DDoS detection and analysis in SDN based environment using support vector machine classifier; Proceedings of the IEEE Sixth International Conference on Advanced Computing (ICoAC); Chennai, India. 17–19 December 2014; pp. 205–210.

[11] Phan T.V., van Toan T., van Tuyen D., Huong T.T., Thanh N.H. OpenFlowSIA: An optimized protection scheme for software-defined networks from flooding attacks; Proceedings of the IEEE Sixth International Conference on Communications and Electronics (ICCE); Ha Long, Vietnam. 27–29 July 2016; pp. 13–18.

[12] Phan T.V., Bao N.K., Park M. A Novel Hybrid Flow-based Handler with DDoS Attacks in Software-Defined Networking; Proceedings of the 2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress; Toulouse, France. 18–21 July 2016; pp. 350–357.

[13] Sarraf, Saman. (2020). Analysis and Detection of DDoS Attacks Using Machine Learning Techniques. American Scientific Research Journal for Engineering, Technology, and Sciences. 66. 95-104.

[14] M Devendra Prasad, P.B.V., C Amarnath, Machine Learning DDoS Detection Using Stochastic Gradient Boosting. International Journal of Computer Sciences and Engineering, 2019. 7(4): p. 157-16.

[15] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, “Developing realistic distributed denial of service (ddos) attack dataset and taxonomy,” in 2019 International Carnahan Conference on Security Technology (ICCST). IEEE, 2019, pp. 1–8.

[16] Wang H, Li W. DDosTC: A Transformer-Based Network Attack Detection Hybrid Mechanism in SDN. Sensors (Basel). 2021;21(15):5047. Published 2021 Jul 26. doi:10.3390/s21155047.

[17] Tan, Mingxing, and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks." International Conference on Machine Learning. PMLR, 2019.

[18] Chollet, François. "Xception: Deep learning with depthwise separable convolutions." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.