IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 12 March 2024, accepted 8 April 2024, date of publication 18 April 2024, date of current version 29 April 2024.

*Digital Object Identifier 10.1109/ACCESS.2024.3391022*

RESEARCH ARTICLE

# Classifying Malware Traffic Using Images and Deep Convolutional Neural Network

**R. E. DAVIS JR., (Member, IEEE), JINGSHENG XU, AND KAUSHIK ROY, (Senior Member, IEEE)**
Department of Computer Science, North Carolina Agricultural and Technical State University, Greensboro, NC 27411, USA
Corresponding author: R. E. Davis Jr. (redavis@aggies.ncat.edu)

This work was supported by the ONR under Award N00014-22-1-2724.

**ABSTRACT** Network traffic classification plays a crucial role in detecting malware threats. However, most existing research focuses on extracting statistical features from the network traffic, ignoring the rich information contained within raw packet capture (pcap) files. To achieve higher accuracy in malware traffic classification, a novel approach is proposed that fully utilizes the information contained in the pcap files by representing them with images and then training deep Convolutional Neural Networks (CNN) to learn the features automatically and classify them with higher accuracy. Selected fields of the IP headers in network sessions are transformed into 50 × 50 RGB images. These images serve as input to CNN, and malware samples are grouped by class or malware name. The model is initially trained and validated on the MCFP dataset with more than 140 malware classes and subsequently tested on separate datasets, namely USTC-TFC2016, Taltech.ee MedBIoT, and IEEE-Mirai. The macro F1 scores and accuracy of this method are significantly higher than the baseline statistical-feature based approach both in the validation dataset and in the test datasets from different sources. The results of this research have the potential to be extended beyond malware classification to enable the classification of various types of network traffic data.

**INDEX TERMS** Malware classification, network traffic classification, deep learning, convolutional neural network.

## I. INTRODUCTION

The detection and classification of malware in network traffic are crucial for ensuring the security and integrity of computer networks. Traditional approaches often rely on extracting specific features from network data, such as packet statistics, flow characteristics, or payload content. However, these methods may not fully exploit the valuable information embedded in the raw packet capture (pcap) files. In this study, we propose a unique approach for malware classification using Convolutional Neural Networks (CNN).

The distinction of this work is attributed to the approach of converting elements of the IP header from network sessions in pcap files into binary data, which serves as the basis for generating 50 × 50 RGB images. These images act as visual representations of the raw network traffic, akin to item barcodes found in retail or grocery stores. By transforming the pcap data into a visual format, the potential of CNNs in image classification is leveraged to accurately classify malware samples.

There are papers [1], [2], and [3] in the literature that achieve 99% plus accuracy with various feature selection and machine learning algorithms. It is very hard to distinguish which approach is better. Several research showed that even baseline methods achieved near perfect accuracy [2], [3]. This could be caused by the datasets used in these research that makes the classification problem easy. Additionally, most of the papers identified in the literature train and test on the same dataset. It is unclear whether the model trained on one dataset will work well on other datasets.

According to the research, there are datasets from different sources with the same malware label. There was rarely any attempt in testing the model trained on one dataset to other datasets from different sources. In this paper, the Malware Capture Facility Project dataset (MCFP) [4] was chosen as the training and validation dataset. MCFP is the largest malware dataset we could find. It contains more than 140 malware classes and 460 gigabytes of packet capture (pcap) files. The

The associate editor coordinating the review of this manuscript and approving it for publication was Md. Moinul Hossain.

distribution of samples over these malware classes varies significantly. These factors make it much harder to achieve high accuracies compared to the datasets used in literature. Our test data come from three different sources: USTC-TFC2016 [1], Taltech.ee MedBIoT [5], and IEEE-Mirai [6]. These dataset were chosen because they have overlapping malware labels with MCFP dataset.

In this paper, a novel methodology for transforming pcap data into visual representations is presented and its effectiveness for malware classification is demonstrated. The remainder of this paper is organized as follows: Section II describes related work. Section III describes our methodology including data collection, image generation, CNN model creation, baseline method for comparison, training, and evaluation. Section IV describes detailed result on validation and test datasets of our method and baseline method. Discussion, conclusion, and future work are presented in Section V.

## II. RELATED WORK

Most of the traffic classification research work in the literature is based on selecting the best features from network traffic data and use machine learning algorithms designed for structured data [7], [8], [9], [10], [11], [12], [13]. Features are usually statistical information about network flows or sessions such as average packet sizes, number of packets per second, and more advanced statistical features. Recently, with success of deep neural networks in research work, new approaches start to emerge.

Wang et al. [1] generated 28 × 28 grayscale images from each network flow and network session. Their images include data from the Ethernet layer header to the application layer. They randomized the identifying information, such as IP addresses and MAC addresses. However, these noise data not only make the image contain a much smaller number of packets, but also make it hard for convolutional network to learn. They built a convolutional neural network that is very similar to LeNet-5 [14] for training and testing. They were able to achieve about 99. 41% accuracy using the USTC-TFC2016 [1] dataset with 20 malware classes, which is a significantly smaller dataset compared to MCFP [4]. Their train and test data were from the same dataset.

Bendiab et al. [15] used Binvis [16], a binary file visualization tool, to generate images from pcap files. They generated only 1,000 images of dimension 1024 × 256 pixels representing 8 different traffic classes from their own traffic dataset. The images are generated from chunks of pcap files. Therefore, images are not associated with flows or sessions. Also, the images seem to include payload data as well as IP addresses. They used Resnet 50 [17] to train and validate these images and achieved a 94. 5% F1 score.

Agrafiotis et al. [2] generated 28 × 28 grayscale images almost identically to Wang's paper [1]. However, they applied to a different dataset called CIC-IDS2017 [18]. Three different machine learning algorithms were used: CNN, vision transformer, and random forest. The images were the input to CNN and vision transformer. CiC-Flow-Meter [19] tool extracted features from the pcap files were the input to the random forest algorithm. They did binary classification of benign and malicious traffic and achieved the best 0.999 accuracy with the vision transformer. The random forest result is 0.985, which is just slightly worse than the vision transformer. They used the same dataset for training and testing.

Bo et al. [3] used the neural networks initially created for natural language modeling of malicious traffic. They treated each byte in the raw packet as a word input to a Gated Recurrent Units (GRU) neural network to classify the malware traffic. Their datasets include ISCX2012 [20], USTC-TFC2016 [1], and CICIDS2017 [18]. They achieved high accuracy on these datasets. Specifically, the accuracy on USTC-TFC2016 [1] was 99.94%. Similar to Wang et al. [1], their train and test were all performed within the same data set.

There are several papers utilizing the MCFP dataset. However, most of the work only used a small subset of the entire dataset. For example, Liu et al. [21] used 8 of the MCFP malware classes in their experiments to generate malware traffic fingerprints for classification. Zhang et al. [22] also used 8 MCFP malware classes in detecting encrypted malicious traffic. Li et al. [23] selected a random set of 20 MCFP malware classes for unknown malware class detection. Zhao et al. [24] used 10 MCFP malware classes in their prototype based learning method in malware classification. Rong et al. [25] used more MCFP classes than the research work above. A total of 42 malware classes were used by the paper. However, they experimented with 5-way and 20-way classifications. The F1 classification score of on 20-way classification was about 97% with their few-shot learning algorithm. In this paper, we used all 142 malware classes in MCFP and achieved a similar F1 score.

## III. METHODOLOGY

### A. MACHINE LEARNING PIPELINE

Figure 1 shows the full machine learning pipeline of this research. The process begins with downloading network traffic data from four sources, including the Malware Capture Facility Project (MCFP), USTC-TFC2016, MedBIoT, and IEEE-Mirai datasets. Datasets were stored in a tree directory structure organized by source and malware class labels. The next stage, the pipeline diverges into image generation and statistical feature extraction. Images generated from the dataset were used to train a CNN model and numeric features extracted from previous stages were used to train random forest, svm, and xgboost models. Finally, the evaluation stage compared F1 scores of CNN versus traditional ML models. In the subsequent sections, each of these steps is described in detail.

### B. DATASETS

#### 1) MCFP DATASET [4]

The Malware Capture Facility Project, affiliated with the Stratosphere IPS Project, captures malware and network data

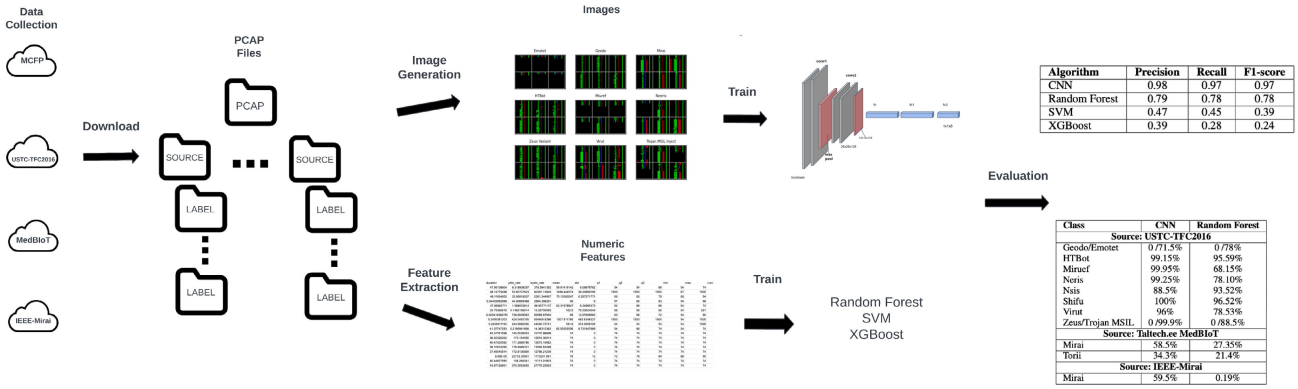



**FIGURE 1.** An overview of malware classification pipeline.

to refine machine learning algorithms in network security. They utilize Malware, Normal, and Background traffic to assess detection and algorithm performance. Each dataset includes pcap files and password-protected zip files related to the malware execution.

From the diverse file types in the MCFP, the focus was solely on raw pcap files representing malware. A script was created to navigate the dataset's JSON structure, and pcaps corresponding to each malware type were extracted, subsequently organizing them into folders named after the malware class.

The dataset represent over 140 classes, to include such malware classes as Neeris, Wootbot, Sogou, RBot, NSIS Agent, Zbot, Trojan MSIL Inject, Sirefef, Dridex, and many others. Pcap file sizes range from 4KB to 70GB in size. This dataset was used for training and validation. The dataset can be downloaded here [26].

#### 2) USTC-TFC2016 DATASET [1]

The USTC-TFC2016 dataset was created by Wang et al. for their malware classification research. contains both benign and malicious network traffic. The malicious traffic is based on CTU dataset [26] collected from 2011 to 2015. The authors made some modifications on the dataset. The dataset included a total of 20 classes, of which 10 were malware classes such as Cridex, Geodo, Htbot, Virut, and Zeus. Pcap file sizes range from 2.5MB to 282MB. This was used as a test dataset. Find the dataset here [27].

#### 3) MEDBIOT DATASET [5]

The MedBIoT dataset was developed by Manzanares, et al. from the Department of Software Science at the Tallinn University of Technology, Estonia. The dataset incorporates a combination of real and emulated IoT devices (a total of 83 devices) and providing genuine malware network data, specifically from three notable botnet malwares: Mirai, Bash-Lite, and Torii. Mirai and Torii were used, where pcap file sizes range from 25MB to 635MB. These were utilized as the test dataset. Find the dataset here [28].

#### 4) IEEE-MIRAI DATASET

Mirai-based Multi-class dataset [6] created by Gebrye, et al. Derived from prior IoT intrusion data, this dataset emphasizes benign traffic and three specific Mirai botnet attacks: SYN-Flooding, ACK-Flooding, and HTTP-Flooding, all treated under the general umbrella of the Mirai class. Pcap file sizes range from 12MB to 87MB. This was used as a test dataset. Find the dataset here [29].

### C. IMAGE GENERATION

For each session in a PCAP file, a 3-channel RGB image of size $50 \times 50$ pixels are generated. Each row of the image represents exactly one packet. Therefore, the first 50 packets of a session are used in generating the image. For each packet, 50 pixels were used to present the following fields in the IP header: Type of Service (TOS), Total Length, Flags, Time To Live (TTL), Flags, and Protocol. Because source and destination IP addresses are identifying data, which should be not used in training, they were not included in the images. Additionally, other fields in the IP header, including checksum, packet identification, and fragment offsets, were not included. These fields are either irrelevant to the malware or potentially identifying. To make it easier for the neural networks classify the images, we used different colors to represent adjacent fields in the IP header.

The following Figure 2 shows 6 random generated sample images arranged in $2 \times 3$ grid for 9 malware classes. Images in the same malware class do look more similar to each other compared to the images from different classes. However, there are visible variations among the images from the same class. In the example of Miuref and Neeris, the variations are quite significant. For some malware, the length of session is much less than 50 packets. For example, Emotet has a lot empty rows in the images, which means short sessions.

Next, the detailed steps of generating these images from the PCAP files are described. Scapy [30] Python library was used to extract the sessions from the PCAP files. However, the library is extremely slow in handling large PCAP files and time complexity seems to be more than linear time. Many of

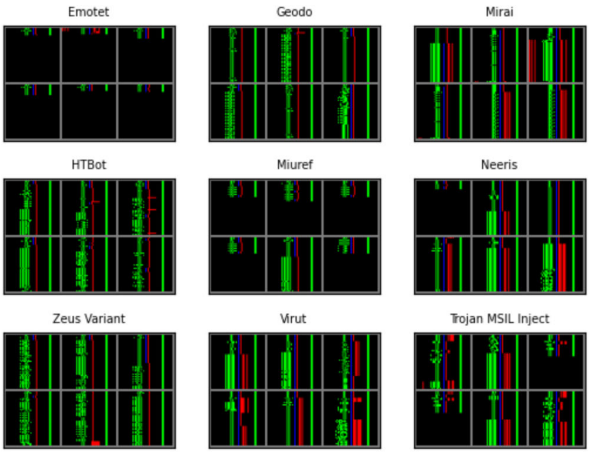



**FIGURE 2.** Sample malware images generated from MCFP dataset.

PCAP files in MCFP dataset are several gigabytes in size. To make the image generation faster, we used 'tcpdump' command to split the large PCAP files into a set of smaller PCAP files of 25 Megabtyes each. The Scapy [30] library is able to handle this size reasonably quickly. A Python script was written to iterate through all the sessions identified by the Scapy library and generate an image for each session. For each session, the Python script reads the first 50 packets in the session and extract the fields mentioned above and use PIL python library to generate a $50 \times 50$ PNG image. To limit the number of total images generated for reasonable training time later, the first of the 25 Megabyte split files was used. After all these steps we generated a total of 1,553,086 images for 142 Malware classes.

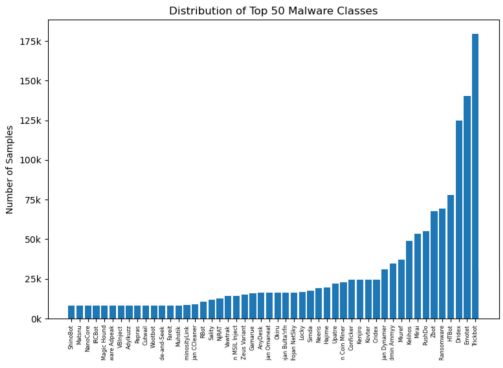



**FIGURE 3.** Image distribution top 50 of malware classes of MCFP dataset.

Figures 3 and 4 show the distributions of 50 largest malware classes and 50 smallest malware classes. The largest malware class Trickbot has about 175 thousand images, while the smallest malware class jRAT Has less than 10 images. The wide distribution of number of samples makes it more challenging to classify with machine learning algorithms.

### D. CNN ARCHITECTURE

VGG [31] was initially used to classify images from the ImageNet [32] dataset. The dataset contains more than

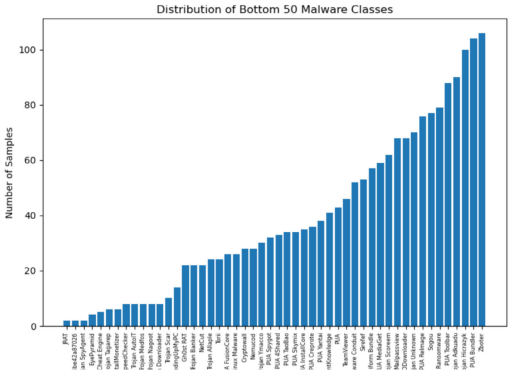



**FIGURE 4.** Image distribution bottom 50 of malware classes of MCFP dataset.

14 million images and 1,000 classes. Packet images are simpler than those of ImageNet. The number of classes 142 is also less than ImageNet. Therefore, it should be less challenging to classify compared to ImageNet. The decision was made to simplify VGG to meet the project's requirements.

The following Figure 5 shows the architecture of our CNN model. The input to the network was $50 \times 50$ pixel RGB images. This was followed by a sequence of convolutional blocks consisting of convolutional layers interspersed with max pooling layers. The convolutional filter sizes were $3 \times 3$ and the number of filters in the convolutional layers were 64, 64, 128, and 128 respectively. To reduce overfitting, dropout layers with a rate of 0.5 were added after every convolutional block. The convolutional features were finally fed into a flatten layer and two fully connected dense layers with 4096 units each before an output layer with 142 channels for 142 malware classes. Figure 6 is the summary of the Deep Convolutional Neural Network which is more detailed description of Figure 5. There are over 93 trainable parameters in this neural network.

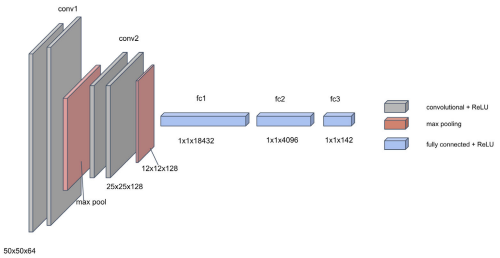



**FIGURE 5.** Deep convolutional neural network architecture diagram.

### E. TRAINING CNN MODEL

A random 80-20 split was performed on the MCFP dataset, resulting in 1,242,468 images in the training set and 310,618 images in the validation dataset. AdamW optimizer in Pytorch library was used with $1.0 \times 10^{-5}$ as the learning rate and 0.01 as the weight decay factor. Cross-Entropy was used as the loss function. The batch size was 128. The CNN models were implemented in Pytorch and trained on 4 Nvidia

```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
├─Sequential: 1-1                        [-1, 64, 50, 50]          --
|    └─Conv2d: 2-1                       [-1, 64, 50, 50]          1,792
|    └─BatchNorm2d: 2-2                  [-1, 64, 50, 50]          128
|    └─ReLU: 2-3                         [-1, 64, 50, 50]          --
├─Sequential: 1-2                        [-1, 64, 25, 25]          --
|    └─Conv2d: 2-4                       [-1, 64, 50, 50]          36,928
|    └─BatchNorm2d: 2-5                  [-1, 64, 50, 50]          128
|    └─ReLU: 2-6                         [-1, 64, 50, 50]          --
|    └─MaxPool2d: 2-7                    [-1, 64, 25, 25]          --
├─Sequential: 1-3                        [-1, 128, 25, 25]         --
|    └─Conv2d: 2-8                       [-1, 128, 25, 25]         73,856
|    └─BatchNorm2d: 2-9                  [-1, 128, 25, 25]         256
|    └─ReLU: 2-10                        [-1, 128, 25, 25]         --
├─Sequential: 1-4                        [-1, 128, 12, 12]         --
|    └─Conv2d: 2-11                      [-1, 128, 25, 25]         147,584
|    └─BatchNorm2d: 2-12                 [-1, 128, 25, 25]         256
|    └─ReLU: 2-13                        [-1, 128, 25, 25]         --
|    └─MaxPool2d: 2-14                   [-1, 128, 12, 12]         --
├─Sequential: 1-5                        [-1, 4096]                --
|    └─Dropout: 2-15                     [-1, 18432]               --
|    └─Linear: 2-16                      [-1, 4096]                75,501,568
|    └─ReLU: 2-17                        [-1, 4096]                --
├─Sequential: 1-6                        [-1, 4096]                --
|    └─Dropout: 2-18                     [-1, 4096]                --
|    └─Linear: 2-19                      [-1, 4096]                16,781,312
|    └─ReLU: 2-20                        [-1, 4096]                --
├─Sequential: 1-7                        [-1, 142]                 --
|    └─Linear: 2-21                      [-1, 142]                 581,774
==========================================================================================
Total params: 93,125,582
Trainable params: 93,125,582
Non-trainable params: 0
Total mult-adds (M): 420.69
==========================================================================================
```

**FIGURE 6.** Detailed summary of CNN model.

GPUs. The model converged fairly quickly within 2 epochs. To show the gradual progress of learning, instead of every epoch, we sampled loss and accuracy for every 100 batches of training, which is equivalent to every 12,800 samples. A random 100 batches in the validation dataset is selected when computing the validation loss and validation accuracy. The following Figures 7 and 8 show the progress of training and validation loss and accuracy on every 100 batches as a unit. Validation loss and validation accuracy is computed after every 100 batches of training in the beginning. This explains why validation result is better than training in the beginning. However, as the loss function and accuracy converge, the difference between the two is very small. It is observed that there is no sign of overfitting. Both training and validation loss stayed stable in the later stage of training. A more detailed result of training and validation will be shown in the result section of this paper.

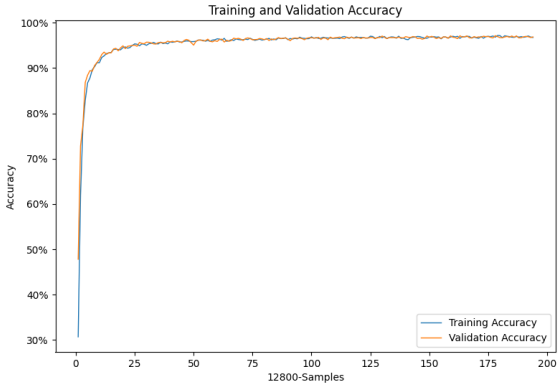



**FIGURE 7.** Training and validation accuracy history.

### F. BASELINE METHOD

The performance of the image-based CNN model was chosen for comparison against a commonly used method in the literature, where statistical features are extracted from the packet captures and machine learning algorithms are applied to this structured dataset. Netml [33] is a Python library that extracts 10 of the most common statistics in the literature: flow duration, number of packets per second, number of bytes per second, and the following statistics of packet sizes: mean, standard deviation, the first to the third quantiles, the minimum and the maximum. A Python script written with netml library to extract these statistical features from the same MCFP dataset. Please, note that the first 25 megabytes was used in large PCAP files for generating images. We did the same when extracting the statistical features. Another thing to note is that these statistical features are flow-based and images are generated based on sessions. Therefore, the statistically based dataset has 4,472,196 samples, which is about twice the number of samples as the images. Random Forest, Support Vector Machine, and XGBoost are applied on this statistically based dataset. A detailed result will be shown in the following result section.

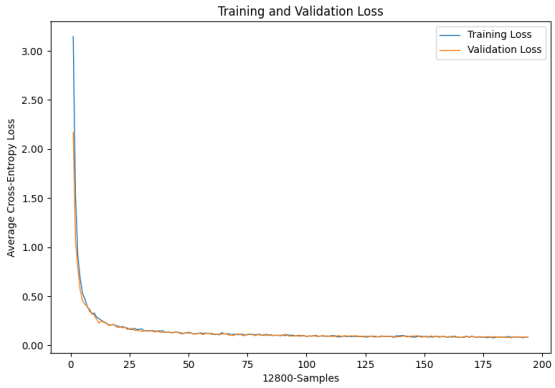



**FIGURE 8.** Training and validation loss history.

### G. EVALUATION METHODOLOGY

Accuracy is not a good measure when the classes are not evenly distributed. Figures 3 and 4 show the distribution of classes is very uneven among different malware classes in our datasets. The decision was made to use the macro F1 score to evaluate the performance of machine learning models. To compute the macro F1 score, precision and recall values are computed for each individual class. For each class $i$, where $1 \leq i \leq n$, and $n$ is the total number of classes, precision, recall, and F1 score are defined as follows:

$$\text{Precision}_i = \frac{TP_i}{TP_i + FP_i} \tag{1}$$

$$\text{Recall}_i = \frac{TP_i}{TP_i + FN_i} \tag{2}$$

$$\text{F1}_i = 2 \cdot \frac{\text{Precision}_i \cdot \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i} \tag{3}$$

$TP_i$, $FP_i$, and $FN_i$ represent the true positives, false positives, and false negatives for malware class $i$, respectively. F1 score for the individual class is defined as the harmonic mean of precision and recall values. Harmonic mean gives higher

weight to the worse performing metric between precision and recall. The macro F1 score is the mean of individual F1 scores for all classes. Macro F1 score treats each class equally without giving higher weights to bigger classes. The macro precision, recall, and F1 score, considering all $n$ classes, are calculated as the arithmetic mean of the individual metrics for each class:

$$\text{Macro Precision} = \frac{1}{n}\sum_{i=1}^{n}\text{Precision}_i \tag{4}$$

$$\text{Macro Recall} = \frac{1}{n}\sum_{i=1}^{n}\text{Recall}_i \tag{5}$$

$$\text{Macro F1} = \frac{1}{n}\sum_{i=1}^{n}\text{F1}_i \tag{6}$$

To evaluate the CNN model and the baseline models, classification reports were first run on the MCFP validation dataset. For each of 142 malware classes, the precision, recall, and F1 score will be computed. The macro precision, recall, and F1 scores will be computed and compared. To further test the performance of the CNN model and the baseline, classification reports will be run on test datasets from different sources including USTC-TFC2016 and Taltech.ee MedBIoT, and IEEE-Mirai. Because these test datasets are captured independently from one another, the evaluation results will be more convincing in determining the performance of our image based approach. A detailed evaluation result will be presented in the next result section.

## IV. RESULTS

The results demonstrate that using images to represent malware packet sessions and training a Deep Convolutional Neural Network (CNN) model generates significantly better accuracy in malware classification compared to traditional methods using numerical features from sessions. Specifically, training a deep CNN on a large dataset of millions of labeled images representing 142 malware classes from the Malware Capture Facility Project (MCFP). This achieved a 97% macro F1 score on the validation set, far exceeding the maximum 74% macro F1 score from Random Forest models trained on numerical features from the same MCFP dataset. Table 1 shows the summary of the classification report in the MCFP validation dataset. CNN on image data generated from malware sessions achieved macro precision, recall, and F1 scores of 98%, 97% and 97%, respectively. Random Forest on aggregated numerical feature dataset extracted from the same packet sessions achieved macro precision, recall, and F1 scores of 79%, 78%, and 78% respectively. Other machine learning algorithms, including XGBoost and SVM, were also tried on the aggregated numerical feature dataset. XGBoost's macro F1 score was only 24%. Due to the higher time complexity of SVM with an RBF kernel, completion of training on the entire training dataset was not achieved. However, on the much smaller subset of the training dataset, the macro F1 score was 40%.

**TABLE 1. Macro Precision, Recall, F1 Score on MCFP validation dataset.**

| Algorithm | Precision | Recall | F1-score |
|---|---|---|---|
| CNN | 0.98 | 0.97 | 0.97 |
| Random Forest | 0.79 | 0.78 | 0.78 |
| SVM | 0.47 | 0.45 | 0.39 |
| XGBoost | 0.39 | 0.28 | 0.24 |

Tests were also conducted on datasets from different sources, and it was found that the image-based deep learning approach consistently outperformed traditional methods by a wide margin. Both trained models were tested on USTC-TFC2016 [3], and Taltech.ee MedBIoT [34], and Mirai-based Multi-Class [29] datasets. They are much smaller datasets than the MCFP dataset containing only about 11 malware classes which are a subset of MCFP's malware classes. The following tables 2 and 3 show the classification reports of these malware classes on MCFP validation datasets. Table 2 shows the result of the Random Forest algorithm on the aggregated numerical feature dataset extracted from the MCFP validation dataset that focuses only on these 11 malware classes from the test datasets. Table 3 shows the result of CNN algorithm on image extracted from the MCFP validation data set that focuses on the same set of malware classes. Please note that these tables show the results on validation dataset.

**TABLE 2. Precision, Recall, F1 Scores Selected 11 Malware classes with random forest.**

| Class | Precision | Recall | F1-score |
|---|---|---|---|
| Emotet | 0.72 | 0.87 | 0.79 |
| Geodo | 0.74 | 0.70 | 0.72 |
| HTBot | 0.85 | 0.91 | 0.88 |
| Mirai | 0.82 | 0.74 | 0.77 |
| Miuref | 0.92 | 0.74 | 0.82 |
| Neeris | 0.69 | 0.68 | 0.68 |
| NSIS Agent | 0.85 | 0.70 | 0.77 |
| Shifu | 0.87 | 0.89 | 0.88 |
| Virut | 0.70 | 0.63 | 0.66 |
| Trojan MSIL Inject | 0.94 | 0.91 | 0.93 |
| Zeus Variant | 0.49 | 0.43 | 0.46 |

**TABLE 3. Precision, Recall, F1 Scores Selected 11 malware classes with image based CNN.**

| Class | Precision | Recall | F1-score |
|---|---|---|---|
| Emotet | 0.99 | 0.87 | 0.93 |
| Geodo | 0.99 | 0.99 | 0.99 |
| HTBot | 1.00 | 1.00 | 1.00 |
| Mirai | 1.00 | 1.00 | 1.00 |
| Miuref | 0.99 | 1.00 | 1.00 |
| Neeris | 0.99 | 0.99 | 0.99 |
| NSIS Agent | 0.85 | 0.88 | 0.87 |
| Shifu | 0.94 | 1.00 | 0.97 |
| Virut | 0.99 | 0.98 | 0.99 |
| Trojan MSIL Inject | 1.00 | 1.00 | 1.00 |
| Zeus Variant | 0.99 | 1.00 | 1.00 |

Table 4 shows the accuracy results of the trained CNN and Random Forest models on three test datasets are shown. Overall, the results indicate that the CNN model trained on image representations of malware sessions provides superior performance in malware classification on test datasets from

different sources. Except for Nsis and Emotet, CNN has outperformed Random Forest by a significant margin. On a closer look on these 2 malware classes, we can find (in Table 1) that CNN's F1 scores on MCFP validation dataset were also the lowest for Nsis and Emotet compared to all other classes. For Emotet, CNN's Recall score on the validation dataset was only 0.87, which is much lower than the Precision score of 0.99. This could mean that the CNN model has relatively high false negatives causing the low accuracy score on the test dataset.

**TABLE 4. Accuracy results of test datasets with CNN and random forest.**

| Class | CNN | Random Forest |
|---|---|---|
| **Source: USTC-TFC2016** | | |
| Geodo/Emotet | 0 /71.5% | 0 /78% |
| HTBot | 99.15% | 95.59% |
| Miruef | 99.95% | 68.15% |
| Neris | 99.25% | 78.10% |
| Nsis | 88.5% | 93.52% |
| Shifu | 100% | 96.52% |
| Virut | 96% | 78.53% |
| Zeus/Trojan MSIL | 0 /99.9% | 0 /88.5% |
| **Source: Taltech.ee MedBIoT** | | |
| Mirai | 58.5% | 27.35% |
| Torii | 34.3% | 21.4% |
| **Source: IEEE-Mirai** | | |
| Mirai | 59.5% | 0.19% |

Because test datasets are from different sources, they may label the same malware differently from MCFP. In Table 4 Geodo from USTC-TFC2016 dataset had a CNN and Random Forest accuracy score of 0%. During testing, most of the test samples were predicted as Emotet. Emotet is also known as Geodo [35]. Testing the CNN model for Emotet, yielded 71.5% accuracy, whereas testing Random Forest model for Emotet yielded an accuracy of 78%. Similar situation exists for Zeus and Trojan MSIL Inject. Zeus had accuracy of 0% for both CNN and Random Forest. Both CNN and Random Forest predicted them as Trojan MSIL Inject instead. The MCFP dataset has a malware label for Zeus Variant, and not the original Zeus. The USTC-TCF2016 dataset had a label for Zeus, instead of Zeus Variant. There is a good chance what USTC- TCF2016 dataset has labeled as Zeus is Trojan MSIL Inject instead. Kaspersky labels Zeus as a Trojan [36]. CNN's accuracy on Mirai was not good compared to other malware classes. Random Forest's accuracy was even much worse. The could have been caused by Mirai malware having many variants [37]. One of the Mirai variants is call Okiru, which is another malware class in MCFP dataset. In general, it is much more challenging to apply a model learned in one dataset to other datasets. However, the CNN based model consistently outperformed the baseline method.

## V. DISCUSSION AND CONCLUSION

The results show that the image-based deep convolutional neural network model performs significantly better than the various machine learning algorithms that use baseline statistical features of flows or sessions in network traffic. The image-based CNN was able to achieve 97% macro F1 score on classifying 142 malware classes in MCFP dataset. The superior performance is evident on both validation and test datasets. The reason for this performance gain could be caused by images preserving the information of individual packet while this information is lost in the statistical features. Even though only the first 50 packets of a session is used for generating images, it was enough to significantly outperform the statistical features. Convolutional neural networks are known to be excellent in automatically learning the features in the unstructured image data. By encoding the packets into images, convolutional neural networks can be enabled to detect the unique patterns in different malware packet images. There are some limitations to our current image generation algorithm. Statistical features include time related information such as flow duration and average number of packets per second. Although the current images do not include this information, it is possible to encode the time gap between packets into images in order to capture the timing information. Also, the image generation method is based only on IP headers. In the next stage of research, the images can be generated to include TCP header and application layer headers. The method presented in this paper could be used not only in malware classification, but also in general network traffic classification tasks.

## ACKNOWLEDGMENT

The views and conclusion contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of U.S. Government.

## REFERENCES

[1] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jan. 2017, pp. 712–717.

[2] G. Agrafiotis, E. Makri, I. Flionis, A. Lalas, K. Votis, and D. Tzovaras, "Image-based neural network models for malware traffic classification using PCAP to picture conversion," in *Proc. 17th Int. Conf. Availability, Rel. Secur.*, Aug. 2022, pp. 1–7.

[3] B. Wang, Y. Su, M. Zhang, and J. Nie, "A deep hierarchical network for packet-level malicious traffic detection," *IEEE Access*, vol. 8, pp. 201728–201740, 2020.

[4] S. Garcia. *Malware Capture Facility Project*. [Online]. Available: https://www.stratosphereips.org/datasets-malw

[5] A. Guerra-Manzanares, J. Medina-Galindo, H. Bahsi, and S. Nõmm, "MedBIoT: Generation of an IoT botnet dataset in a medium-sized IoT network," in *Proc. 6th Int. Conf. Inf. Syst. Secur. Privacy*, 2020, pp. 207–218. [Online]. Available: https://cs.taltech.ee/research/data/medbiot

[6] H. Gebrye. (2023). *Mirai-Based Multi-Class Dataset*. [Online]. Available: https://dx.doi.org/10.21227/h4ac-wr38

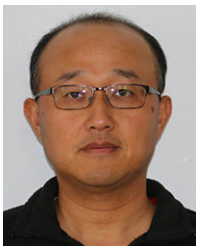
[7] Y. Dhote, S. Agrawal, and A. J. Deen, "A survey on feature selection techniques for internet traffic classification," in *Proc. Int. Conf. Comput. Intell. Commun. Netw. (CICN)*, Dec. 2015, pp. 1375–1380.

[8] E. de la Hoz, E. de la Hoz, A. Ortiz, J. Ortega, and A. Martínez-Álvarez, "Feature selection by multi-objective optimisation: Application to network anomaly detection by hierarchical self-organising maps," *Knowl.-Based Syst.*, vol. 71, pp. 322–338, Nov. 2014.

[9] H. Zhang, G. Lu, M. T. Qassrawi, Y. Zhang, and X. Yu, "Feature selection for optimizing traffic classification," *Comput. Commun.*, vol. 35, no. 12, pp. 1457–1471, Jul. 2012.

[10] A. Fahad, Z. Tari, I. Khalil, A. Almalawi, and A. Y. Zomaya, "An optimal and stable feature selection approach for traffic classification based on multi-criterion fusion," *Future Gener. Comput. Syst.*, vol. 36, pp. 156–169, Jul. 2014.
[11] Z. Liu, R. Wang, M. Tao, and X. Cai, "A class-oriented feature selection approach for multi-class imbalanced network traffic datasets based on local and global metrics fusion," *Neurocomputing*, vol. 168, pp. 365–381, Nov. 2015.
[12] P. Wei, Y. Li, Z. Zhang, T. Hu, Z. Li, and D. Liu, "An optimization method for intrusion detection classification model based on deep belief network," *IEEE Access*, vol. 7, pp. 87593–87605, 2019.
[13] M. Moradi and M. Zulkernine, "A neural network based system for intrusion detection and classification of attacks," in *Proc. IEEE Int. Conf. Adv. Intell. Syst. Theory Appl.*, May 2004, pp. 15–18.
[14] Y. LeCun, L. D. Jackel, L. Bottou, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, U. A. Müller, E. Sackinger, and P. Simard, "Learning algorithms for classification: A comparison on handwritten digit recognition," *Neural Netw., Stat. Mech. Perspective*, vol. 261, no. 276, p. 2, 1995.
[15] G. Bendiab, S. Shiaeles, A. Alruban, and N. Kolokotronis, "IoT malware network traffic classification using visual representation and deep learning," in *Proc. 6th IEEE Conf. Netw. Softwarization (NetSoft)*, Jun. 2020, pp. 444–449.
[16] A. Cortesi, "Binvis.io: Visual analysis of binary files," Tech. Rep., 2016.
[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[18] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy*, 2018, pp. 108–116.
[19] A. Habibi Lashkari, G. Draper Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of tor traffic using time based features," in *Proc. 3rd Int. Conf. Inf. Syst. Secur. Privacy*, 2017, pp. 253–262.
[20] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Comput. Secur.*, vol. 31, no. 3, pp. 357–374, May 2012.
[21] J. Liu, Q. Xiao, L. Xin, Q. Wang, Y. Yao, and Z. Jiang, "M3F: A novel multi-session and multi-protocol based malware traffic fingerprinting," *Comput. Netw.*, vol. 227, May 2023, Art. no. 109723.
[22] X. Zhang, M. Zhao, J. Wang, S. Li, Y. Zhou, and S. Zhu, "Deep-forest-based encrypted malicious traffic detection," *Electronics*, vol. 11, no. 7, p. 977, Mar. 2022.
[23] X. Li, J. Fei, J. Xie, D. Li, H. Jiang, R. Wang, and Z. Qi, "Open set recognition for malware traffic via predictive uncertainty," *Electronics*, vol. 12, no. 2, p. 323, Jan. 2023.
[24] L. Zhao, L. Cai, A. Yu, Z. Xu, and D. Meng, "Prototype-based malware traffic classification with novelty detection," in *Proc. ICICS*, Beijing, China. Cham, Switzerland: Springer, 2019, pp. 3–17.
[25] C. Rong, G. Gou, C. Hou, Z. Li, G. Xiong, and L. Guo, "UMVD-FSL: Unseen malware variants detection using few-shot learning," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2021, pp. 1–8.
[26] Stratosphere. (2016). *CTU University, The Stratosphere IPS Project Dataset*. [Online]. Available: https://www.stratosphereips.org/dataset.html
[27] Y. Lu. (2016). *USTC-TFC2016 Dataset*. [Online]. Available: https://github.com/yungshenglu/USTC-TFC2016
[28] Group. (2022). *Medbiot Dataset*. [Online]. Available: https://cs.taltech.ee/research/data/medbiot/
[29] I. DataPort. (2021). *Mirai-based Multi-class Dataset*. [Online]. Available: https://ieee-dataport.org/documents/mirai-based-multi-class-dataset
[30] P. Biondi and contributors. *Scapy*. [Online]. Available: https://scapy.net/
[31] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.
[32] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.
[33] K. Yang, S. Kpotufe, and N. Feamster, "Feature extraction for novelty detection in network traffic," 2020, *arXiv:2006.16993*.
[34] A. Guerra-Manzanares, J. Medina-Galindo, H. Bahsi, and S. Nõmm, "Using MedBIoT dataset to build effective machine learning-based IoT BotNet detection systems," in *Proc. Int. Conf. Inf. Syst. Secur. Privacy*. Cham, Switzerland: Springers, 2020, pp. 222–243.
[35] S. Kuraku and D. Kalla, "Emotet malware—A banking credentials stealer," *IOSR J. Comput. Eng*, vol. 22, pp. 31–41, 2020.
[36] Kaspersky Resource Center. (2023). *The Zeus Virus*. [Online]. Available: https://www.kaspersky.com/resource-center/threats/zeus-virus
[37] TechTarget. (2023). *Okiru Malware: How Does This Mirai Malware Variant Work?* [Online]. Available: https://www.techtarget.com/searchsecurity/answer/Okiru-malware-How-does-this-Mirai-malware-variant-work

**R. E. DAVIS JR.** (Member, IEEE) received the master's degree in computer science with a focus on cybersecurity and artificial intelligence. He is currently pursuing the Ph.D. degree with the Department of Computer Science, North Carolina Agricultural and Technical State University (NC A&T). He has more than 20 years of experience in the tech industry, working on a variety of projects from small startups to large corporations in the public and private sectors. He is passionate about using technology to solve real-world problems. In his previous roles, he was responsible for providing systems engineering, technical training, support, and consulting to customers on a variety of platforms. He has a deep understanding of cloud computing services and is a passionate advocate for using cloud computing to solve business problems. He is a highly skilled and experienced engineer with a proven track record of success in the tech industry. He is confident that he can make a significant contribution to any team or organization. In his spare time, he enjoys technical evangelism and training, public speaking, and spending time with his family.



**JINGSHENG XU** received the B.S. degree from Nanjing University, China, the M.S. degree from Peking University, China, and the Ph.D. degree from Michigan State University. He is currently an Associate Professor with the Department of Computer Science, North Carolina Agricultural and Technical State University (NC A&T). He is participated in numerous funded projects in cyber security research and education. He teaches Python for data science, network security, data structures, advanced algorithms, and other courses. He has published many peer-reviewed research papers on the topics of cyber security. His research interests include network security, machine learning, and modeling and simulation.

**KAUSHIK ROY** (Senior Member, IEEE) is currently a Professor and the Chair of the Department of Computer Science, North Carolina Agricultural and Technical State University (NC A&T). He is also the Jefferson-Pilot/Ron McNair Endowed Chair with the Department of Computer Science, NC A&T. He has more than 160 publications, including 45 journal articles and a book. His research is funded by the National Science Foundation (NSF), the Department of Defense (DoD), and the Department of Energy (DoE). He is the Director of the Center for Cyber Defense (CCD) and the Center for Trustworthy AI. He also leads the Cyber Defense and AI Laboratory. His current research interests include cybersecurity, cyber identity, biometrics, machine learning (with a focus on deep learning), data science, cyber-physical systems, and big data analytics.