Machine Learning and Binary Visualization for Classifying Files in Network Traffic

Baccalaureate Research Project
Winter 2018

Cyril Vicherek
024-971-145
And
Thomas Zuk
038-893-145

School of Information & Communications Technology
Seneca College of Applied Arts and Technology
Toronto, Canada

ABSTRACT

Everyday hundreds of new malware samples are being discovered. The sheer amount of data makes these samples difficult to analyze with traditional tools which rely on the known structure of data to determine classes or families of malware. “Symantec discovered more than 430 million new unique pieces of malware in 2015, up 36 percent from the year before”(Symantec, 2016). Therefore, there is an opening for tools and techniques which can analyze the data in network traffic, without knowing the structure of the data.

We propose a tool that extracts unencrypted network data from packet captures files and uses binary visualization techniques, namely: entropy, natural translation, and hilbert curve to create images from the data. Those images are then used as a training set for the machine learning process. We showcase the classifications of image and evaluate the accuracy of various models. We conclude with a summary of our findings.

TABLE OF CONTENTS

1.0 Introduction **4**
- **1.1 Objective** **5**
- **1.2 Approach** **6**
- **1.3 Organization of this Paper** **6**

2.0 Literature Review **7**
- **2.1 Data Visualization** **7**
- **2.2 Binary Visualization** **10**

3.0 Methodology **13**
- **3.1 Visualization Types** **13**
- **3.2 Generating Visualizations** **14**
  - **3.2.1 Graphics** **15**
  - **3.2.2 Audio** **18**
  - **3.2.3 Portable Document Format** **21**
  - **3.2.4 Non-Malicious Executables (Windows)** **24**
  - **3.2.5 Non-Malicious Executables (Linux)** **27**
  - **3.2.6 Malicious Executables** **30**
- **3.3 Image Classification with Machine Learning** **33**
  - **3.3.1 TensorFlow** **34**
  - **3.3.2 Server Specifications and Setup** **34**

4.0 Implementation **37**
- **4.1 Retraining** **37**
- **4.2 Classifying** **37**

5.0 Experiment **38**
- **5.1 Experiment Setup** **38**
  - **5.1.1 Collecting Samples** **39**
  - **5.1.2 Creating and capture network traffic** **39**
  - **5.1.3 Parsing and extracting artifacts (files) from network traffic** **40**
  - **5.1.4 Creating visualizations** **42**
- **5.2 Hypothesis** **43**
- **5.3 Observations/Results** **43**
  - **5.3.1 Model Version 1** **45**
  - **5.3.2 Model Version 1.1** **51**
  - **5.3.3 Model Version 2.0** **52**
  - **5.3.4 Model Version 2.1** **56**

5.3.5 Model Version 3 **60**
5.3.6 Model Version 3.1 **64**
5.3.5 Summary **65**
5.4 Discussion **66**

6.0 Future Work **68**

References **70**

1.0 Introduction

Many security research organizations have reported hundreds of new malware samples each day, and millions of new samples each month, many of which originate from the same class or families of malware. According to the "2017 State of Malware Report", Malwarebytes examined over one billion malware incidents between June and November, 2016. (Malwarebytes Labs, 2017) This presents a problem for malware analyst's, the traditional tools do not allow them to keep up with the new types of data, as traditional tools rely on knowing the structure of the data rather than being capable of processing the data regardless of structure. This leads to a slow and tedious malware analysis process.

As the number of malware variants and types increase year over year new tools need to be developed in order to streamline the analysis process. Malware authors are employing anti-analysis techniques such as anti-disassembly, anti-debugging, obfuscation, and virtual machine detection, which renders traditional analysis methods impractical and time consuming. (Sikorski, Honig. 2012) News tools which can process data regardless of the structure of that data, and provide features for analysis is a problem which binary visualization aims to solve. "In our era in the 70´s the first infographics appeared in journals and magazines in order to summarize information and create a great impact in a massive way. Several practitioners and academicians in diverse areas need to present data graphically. From geographers, to economists, military, statisticians, engineers, biologists, to many others fields, many professionals need to see and understand data graphically"(Aparicio, Costa. 2015.) Binary visualization is a technique that can be used to understand data graphically and is capable of determining patterns in data without

knowing the structure of the data, and leveraging these techniques for malware analysis leads to faster analysis. Although binary visualization, for the most part, has been applied to malware analysis there are many fields which can leverage these techniques.

Due to the increasing threat of malware, techniques which can analyze the data in packet capture files, in an automated way, need to be developed. This paper presents a tool for analyzing unencrypted network traffic and determining artifacts (key, distinguishable pieces of data) in the traffic. We present a tool which can extract and group similar data (e.g. data from the same TCP stream), and create a visual representation of the data. Our method utilizes binary visualization techniques to create images representing the traffic, which can be fed through a machine learning algorithm to determine patterns and highlight artifacts in the traffic.

1.1 Objective

The objective of this paper is to determine if binary visualization technique, combined with machine learning algorithms, can be used to detect file types in packet capture files that contains unencrypted network traffic. Our experiments assume that the network traffic is using the unsecured application layer protocol, HTTP, and should be able to identify: image files, audio files, pdf documents, linux binaries, windows binaries, and windows executable malware. Secured protocols such as HTTPS and TLS are not within the scope of the project.

Results would open doors for tools which would be capable of detecting files (both malicious and non-malicious) in network traffic in an automated and scalable way, cutting down the time it takes to analyze packet capture files during incident response activities.

1.2 Approach

The approach to our experiments can be split up into three phases. The first phase involves gathering datasets of different file types and visualization them using three different visualization types. Next, we split up the datasets by file type and visualization type and retrained them on the InceptionV3 model. Finally we tested our new models on visualizations not seen by the neural network and record the results.

1.3 Organization of this Paper

Chapter 2 of this paper is a literature review of related works. Chapter 3 explores the methods and techniques we used in our experiments. Chapter 4 outlines the architecture and implementation of our experiments. This includes the system and hardware component specifications used to generate the images and perform the machine learning tasks. Chapter 5 examines the experiments we performed to validate our work as well as the results of the experiments. Chapter 6 will conclude our project with future work.

2.0 Literature Review

Malware is becoming significantly more rampant making Malware Analysis a very time consuming and tedious process, and thus, tools are needed which can quickly and accurately detect and classify malicious files (Wagner et al. 1). To combat the rising threat of malware, new approaches to analysis need to be developed which do not rely on known structures of files, but can analyze files independent of structure and type; a task which binary and data visualization techniques excel at.

We examine two methodologies for visualization: 1) data visualization, techniques which can be used to easily summarize data in various formats such as graphs and charts. 2) binary visualization, specific techniques which act upon binary data as opposed to general data. We examine techniques from both methodologies which can be used to analyze large data sets, as well as how those sets can be used in automated machine learning algorithms.

2.1 Data Visualization

Data visualization techniques offer a rich set of features that allow data to be explored in unique and powerful ways to gain a better understanding of the information being analyzed. Data visualization is used in many fields of work and research, the primary goal is to help communication information using recognizable visual cues to clearly and efficiently portray information about data which can include statistics, trends and results. Data visualization methods include graphs, charts, plots, or diagrams which help analysts understand information

easier than looking at the raw data (Godfrey, Gryz and Piotr 1). However, this is not a foolproof solution and some challenges do arise. Godfrey, Gryz and Piotr (1) classify these challenges into two categories: (1) Visual Scalability. As datasets become larger it becomes difficult to plot all the data points and thus, new methods need to be found to accurately convey the entire dataset. (2) Data Processing Scalability. Large datasets need to be stored in databases capable of handling the volume of data, and so, efficiency and performance are critical in making data visualization tools valuable.

Visualizations can reveal the change in architecture and structure of software binaries, an area where traditional tools fall short. Panas (185) showed how software binaries can be visualized using Abstract Syntax Trees (AST) which allow for easy control-flow and data-flow graphs for both source code and software binaries (Panas 185). In addition to visualizations not depending on the known structure, visualization techniques can be applied to a large number of metrics to aid in analysis of different binary data. One way Panas (186) overcome the problem of large binary sizes which in turn increased the size of the three dimensional visualizations was performing summary operations like diff and modulo which helped manage larger samples as well as colour coding for a fourth dimension.

Sadana and others (1993) created OnSet a pixel matrix visualization tool to examine large-scale binary set data. To address scalability they introduce visual and interactive techniques to help identify commonalities and differences between binary set data. (Sadana et al. 1993) Some interactive features that were in OnSet that may prove useful include hover-over

highlighting, use highlights, labels and buttons for displaying different information and properties, zoom and pan operations to rearrange visible elements and layers, summary operations such as logical AND, OR, XOR combined with colour gradients to represent the states based on the operations. (Sadana et al. 1998) Some of the interactive features also help address the issue of scalability and present large amounts of data, specifically the summary operations, zooming, panning in combination with multi-layered visualizations. Another feature presented by Sadana et al. is bands that link visualizations together that represented the similarity based on thickness. Features not included in OnSet that may be helpful for binary visualization analysis include the options to include or exclude specific parts of data, the ability to create graphs based on the visualizations. The tool also does not provide the ability to look at the data contained in the binary sets, it only examines relations between set contents.

Similarly Conti and others (3) present an automated way of identifying distinct regions of large binary files(larger than 2 GB) by analyzing the byte distribution of the data using various algorithms including: Mean, Shannon's Entropy, CHI SQUARE, and Hamming Weight (Conti et al. 4-5). Regions of low values indicated uncompressed data (e.g. text files or images) while regions of high data indicated compressed or encrypted data (Conti et al. 5).

These techniques presented above will require exploration and examination to determine their specific implementation and usability for examining artifacts in web traffic on a binary level. Web traffic can generate large amounts of data with packet capture easily rising into the

Gigabyte level in relatively short time spans, scalability and the ability to summarize some of the data will be important.

2.2 Binary Visualization

At the lowest hardware and software levels all data is stored as binary data, which is data represented in the binary numeral system. Text files, PDFs, music, videos, all files are at the lowest levels represented binary formats but are of most use to us in their final format. Computer software interprets the binary data and displays or presents it to us accordingly such as audio for music and readable language for text documents. However, not all data is intended to be presented to humans in a readable format - e.g. such as executable software. Executable software is structured for how a computer can best interpret and utilize it. Binary visualization is the statistical representation of data at its lowest (binary) level. Binary format is not suitable for human interpretation, which is where binary visualization comes in.

Conti and others (10) explores ways of visualizing binary data in three unique ways: (1) The Byteview representation, a method which maps each byte in the binary data to a pixel on the screen. For example, the first byte would be plotted at (1, 1). The pixels are then color coded based on their value (e.g. a byte value of 00 would be black, while a byte value of FF would be green). (2) The byte presence view, a method which colors pixels in rows based and columns based on the value in the byte-view. (3) The dot plot view, a method borrowed from bioinformatics which uses a matrix to compare values. In the example provided by Conti and others (10) a single file was plotted on the x and y axis. The values that were the same on the x

and y values were highlighted. This allowed binary data to be viewed in a very concise way while still revealing important information about the binary data.

Due to the increasing quantity of data that analysts need to sift through, traditional tools are proving to be ineffective, because they rely on known structure of data. On the other hand, binary visualization techniques do not depend on structure, making them an effective way analyzing known and unknown data types (Domas 14). Domas builds upon the work of Conti and others (11-12) and demonstrates an additional method of visualizing data using digraphs, a method of plotting a pair of consecutive bytes on on x,y axis to create an image representing the binary data (Domas 17-22). This reveals an interesting characteristic of data; similar types of data have similar patterns which can be used to identify the data type (e.g image files have similar patterns, but are distinguishable from audio files or ascii text) (Domas 17-22). Furthermore, if binaries contained other types of data, those types could also be identified; for example, the images, text, and raw binary data contained in proprietary pieces of software, such as Microsoft's Notepad, could also be identified (Domas 33).

Hartstack propelled these ideas forward and showed, not only are binary files distinguishable, but fragments of files also have distinguishable patterns (Hartstack 29). Given a specific sized chunk of the original file (512 bytes, for example) and plotted using the digraph technique exhibited similar characteristics as the original file (e.g. 512 bytes of an audio file showed similar characteristics as the complete audio file which the fragment was taken from) (Hartstack 29). Moreover over 51,000 file fragments from 13 different files types including:

Audio Interleave files (.avi), Microsoft Word Document (.doc), Window Executable File (.exe), Graphical Interchange Format (.gif), Joint Photographic Experts Group (.jpg), Apple QuickTime Movie (.mov), Portable Document Format (.pdf), Powerpoint Presentation (.ppt), Plain Text file (.txt), WAVE Audio file (.wav), Windows Media Audio file, (.wma), Windows Media Video file (.wmv), and Zipped files (.zip) were used in machine learning software, like Weka, to classify file types (Hartstack 26).

3.0 Methodology

The methodology section describes how data was formatted to prepare for visualization, how we parsed data out of the packet capture files, the three types of visualizations that we used, results of the visualizations on different file types, and machine learning image classification used to identify files types in network traffic.

3.1 Visualization Types

We use three visualization types for generating visualizations. First, the entropy visualization, allows us to see areas of high and low entropy, which is useful for determining what type of data certain regions contain. The shannon's entropy values for chunks of data are calculated and those chunks are color coded based on the calculated entropy values. Regions of higher entropy normally indicate compression or encryption and will be brighter (in our case, a bright pink), while regions of low entropy which indicate no or very little randomness will be darker (dark blue, purple, or black). Second, the hilbert curve, a type of space-filling curve which preserves spatial locality and clusters similar areas together. Third, the natural visualization which plots bytes in succession, linearly. Color coding is performed based on the hexadecimal values: 0xFF is assigned white, 0x00 is assigned black, ASCII values are assigned red, and all other values are assigned blue.

Using these three visualizations will allow us to discover which visualizations types will yield the best results for machine learning. Chapter 5 examines the tests we performed as well

which visualizations proved the most effective and had the most accurate machine learning classification.

3.2 Generating Visualizations

In section 3.1 we explained the three visualization types we used. This section covers how we generated those visualizations, performance issues with the process, and examples of the visualizations to see the patterns in various files for each of the visualization types.

Our visualization script created takes 3 options: the input file, output file, and the type of visualization which can be entropy, hilbert, natural, or all for all 3 types. Visualization speed is dependent on the size of the file as well as hardware specifications such as CPU and memory speed. Performance could be improved by switching to a lower level language such as C or Rust to generate the visualizations.

```
tz@ubuntu:~/Documents$ time python maml-scripts/maml.py visualize -i Russian_Blue_38.jpg -o ./Russian_Blue_38 -V all

real    0m25.219s
user    0m25.132s
sys     0m0.040s
tz@ubuntu:~/Documents$
```

Figure 1: A .JPEG image was used for input for the three visualization types. Duration was 25 seconds.

The following sections provide examples of the visualizations of graphics (jpg files), audio (mp3 files), portable document format files which can contain textual and graphical data, non-malicious linux and windows executables, and malicious windows executables.

3.2.1 Graphics

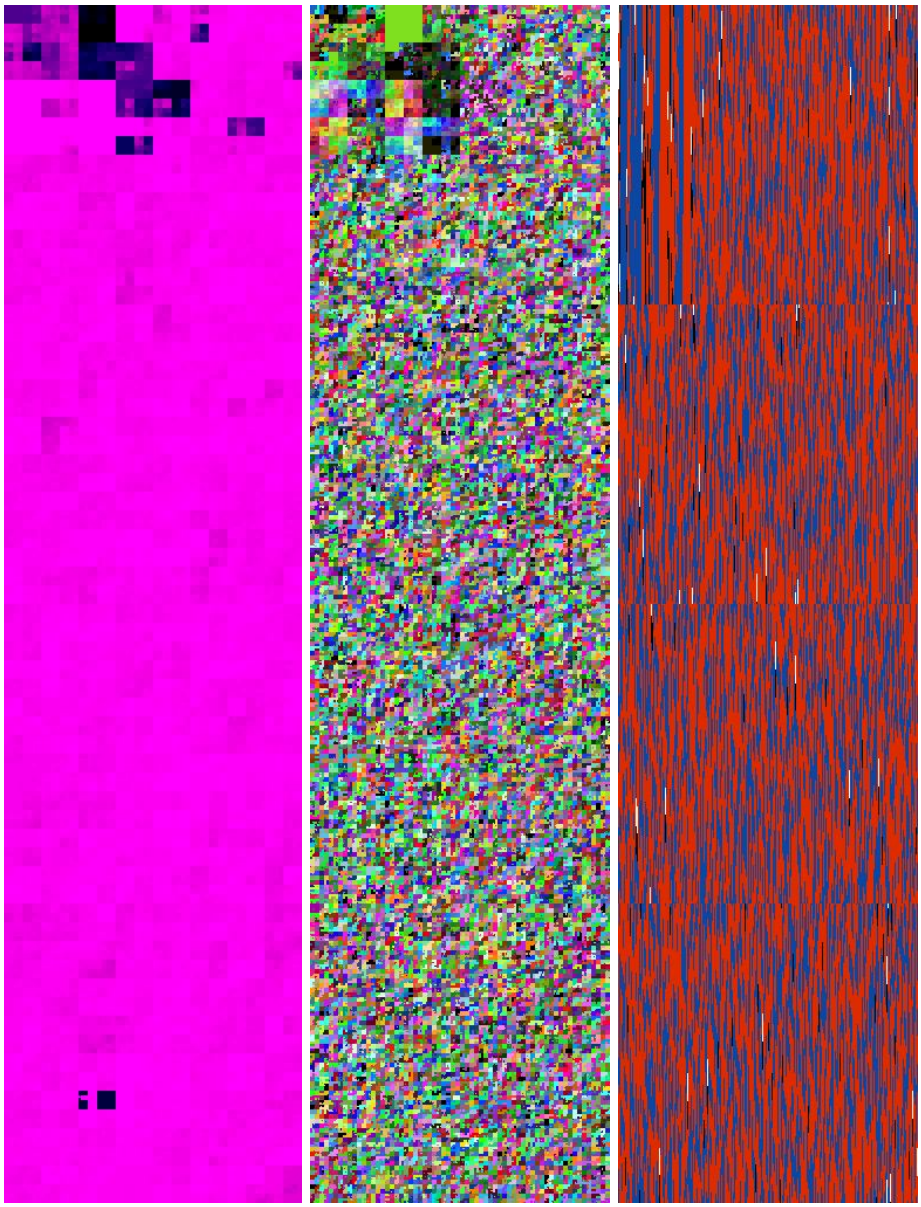

Figure 2: Sample 1 visualizations of what a jpg image looks like.

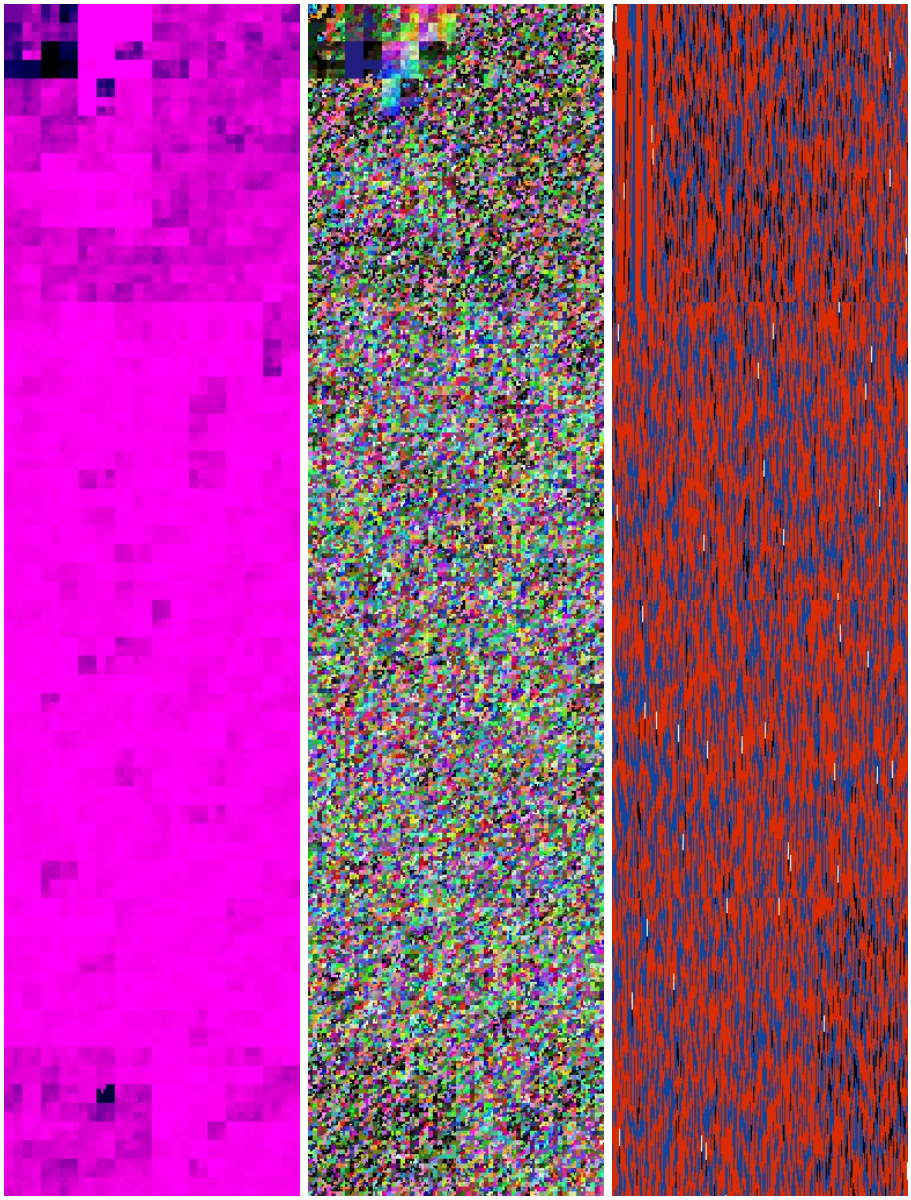

Figure 3: Sample 2 visualizations of what a jpg image looks like.

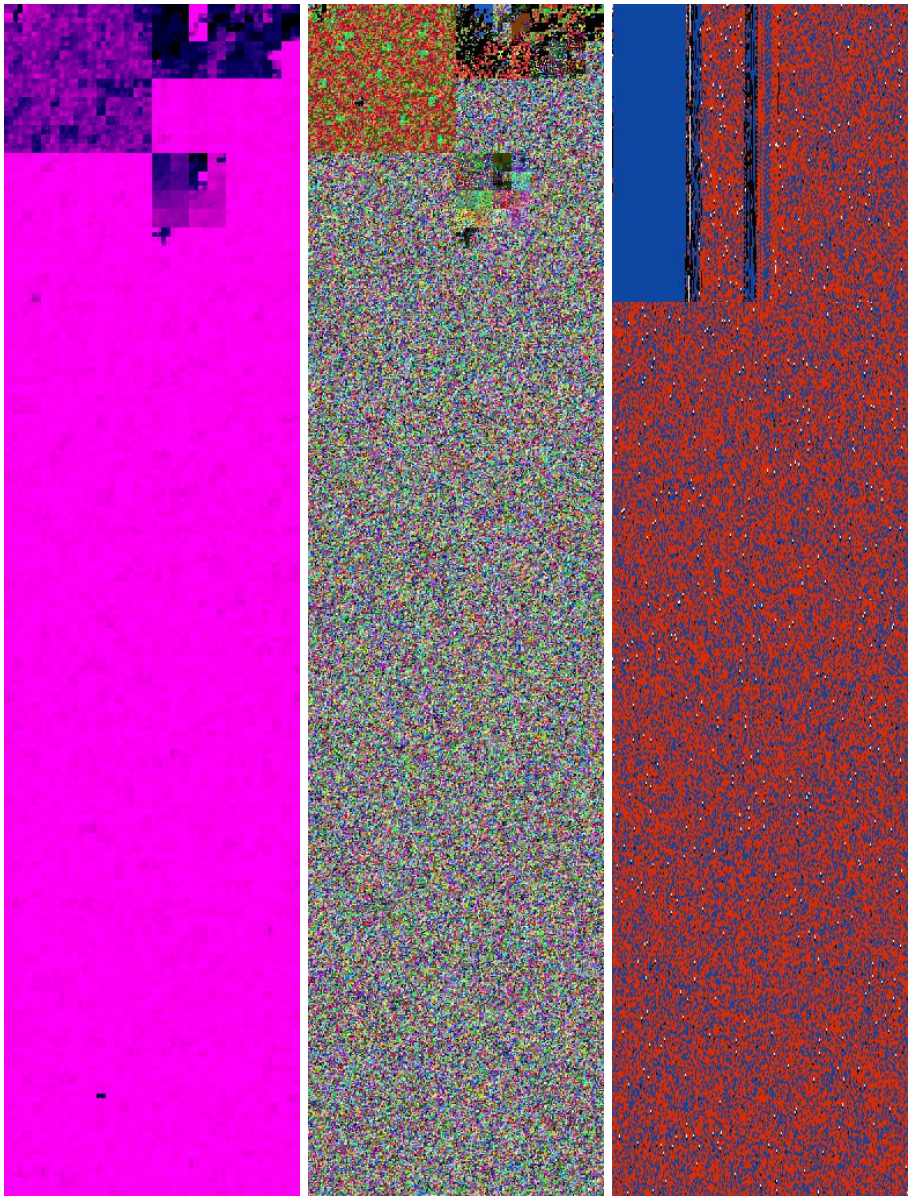

Figure 4: Sample 3 visualizations of what a jpg image looks like.

3.2.2 Audio

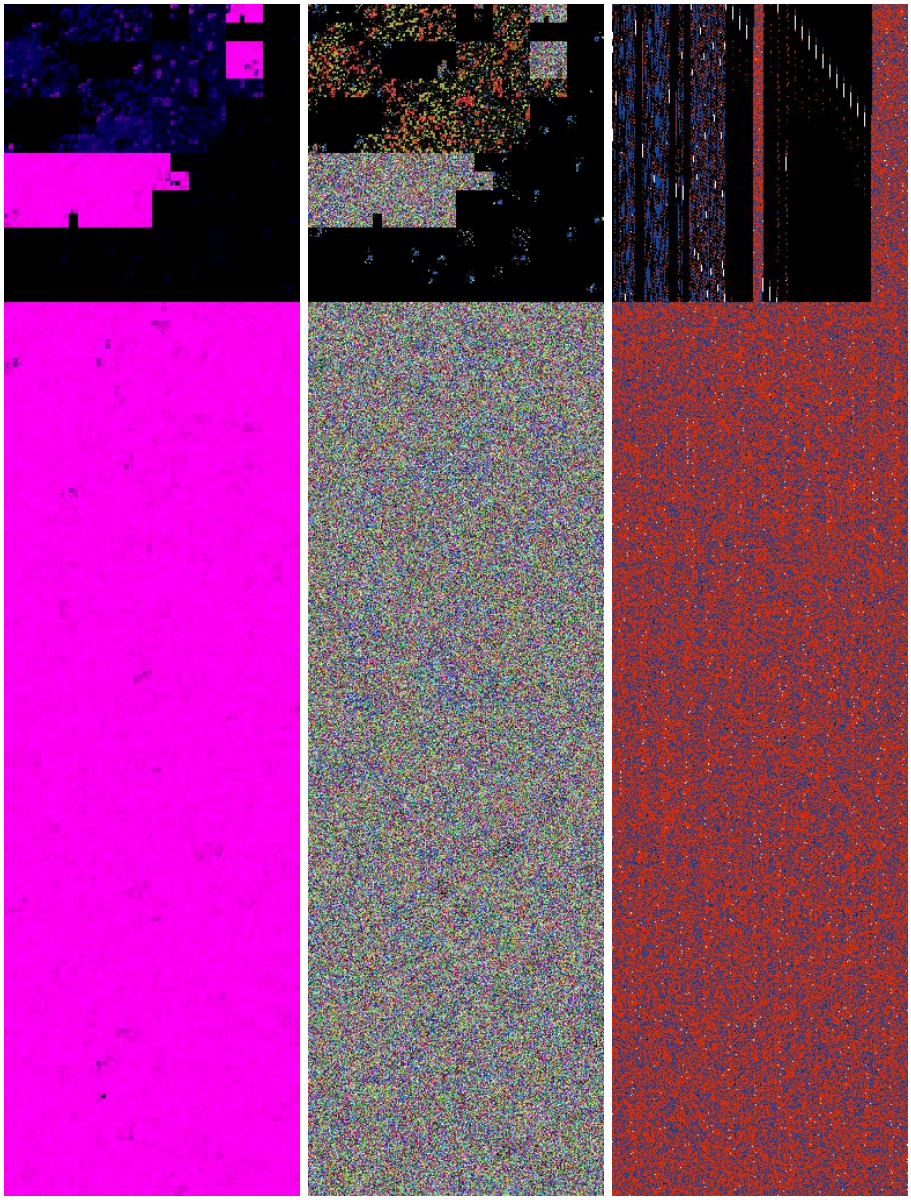

Figure 5: Sample 1 visualizations of what a mp3 audio file looks like.

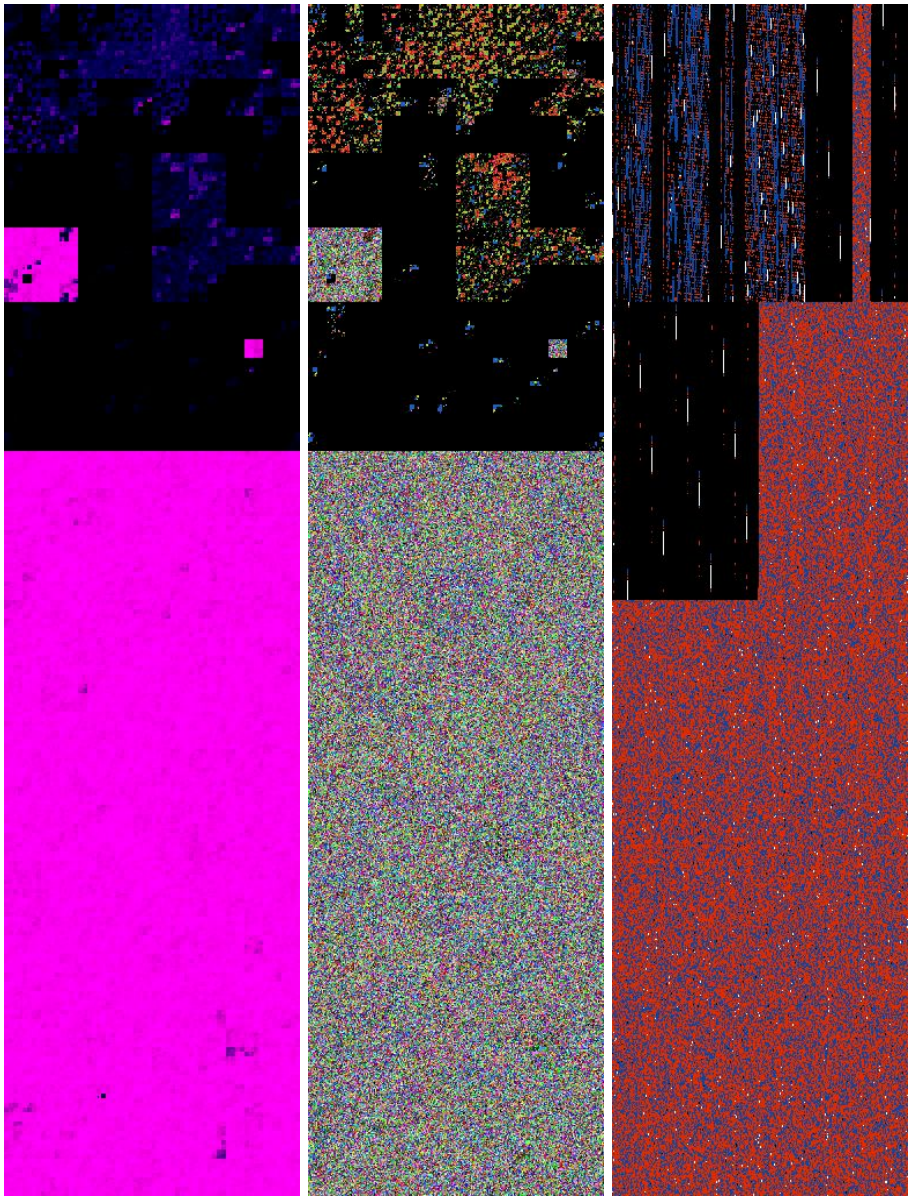

Figure 6: Sample 2 visualizations of what a mp3 audio file looks like.

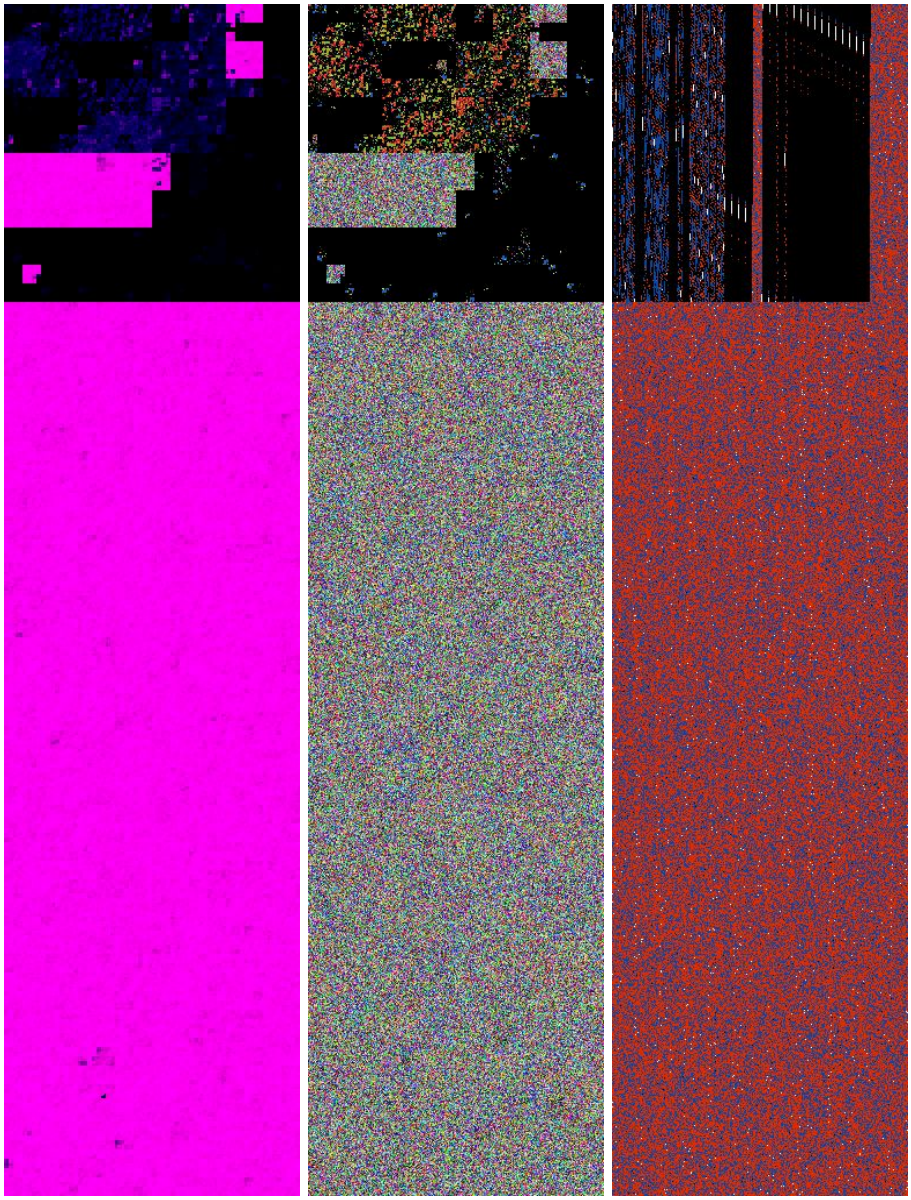

Figure 7: Sample 3 visualizations of what a mp3 audio file looks like.

3.2.3 Portable Document Format

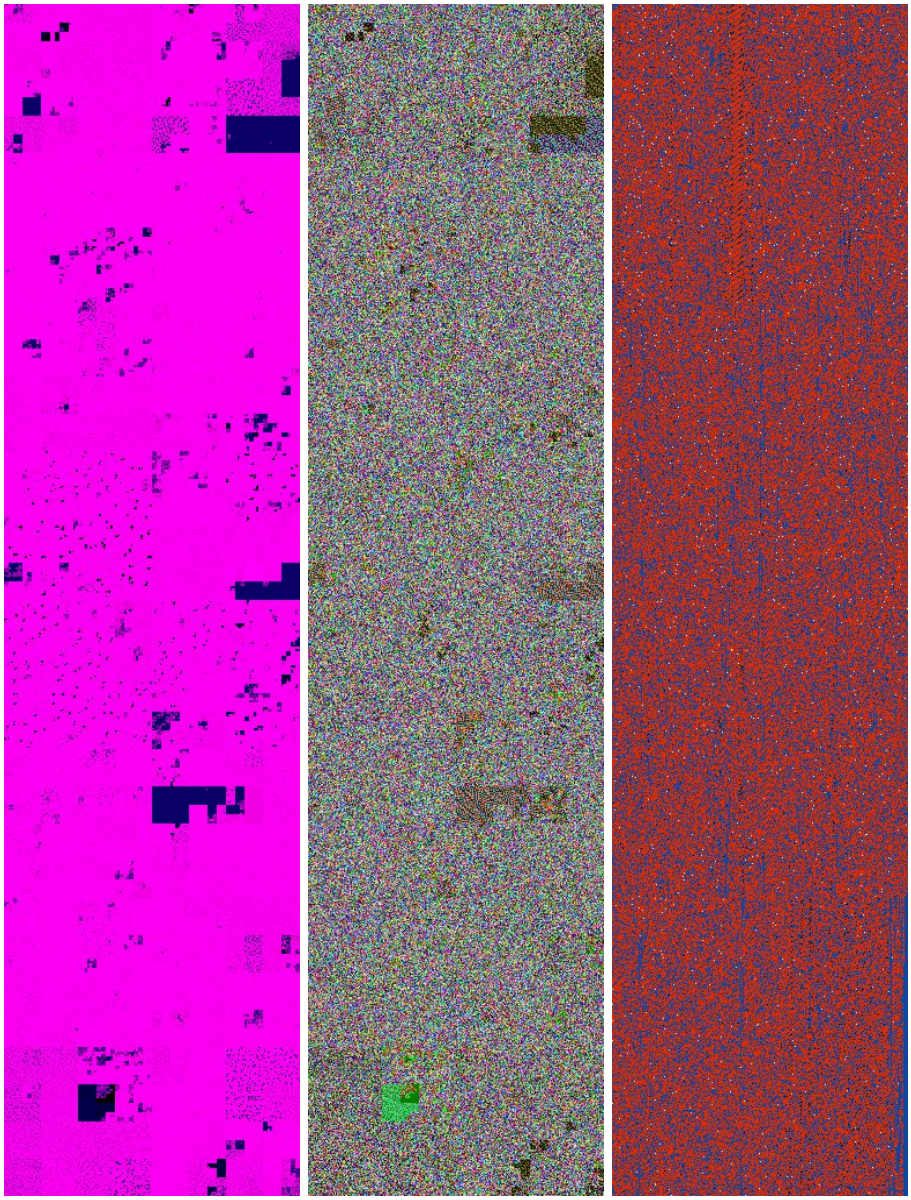

Figure 8: Sample 1 visualizations of what a pdf file looks like.

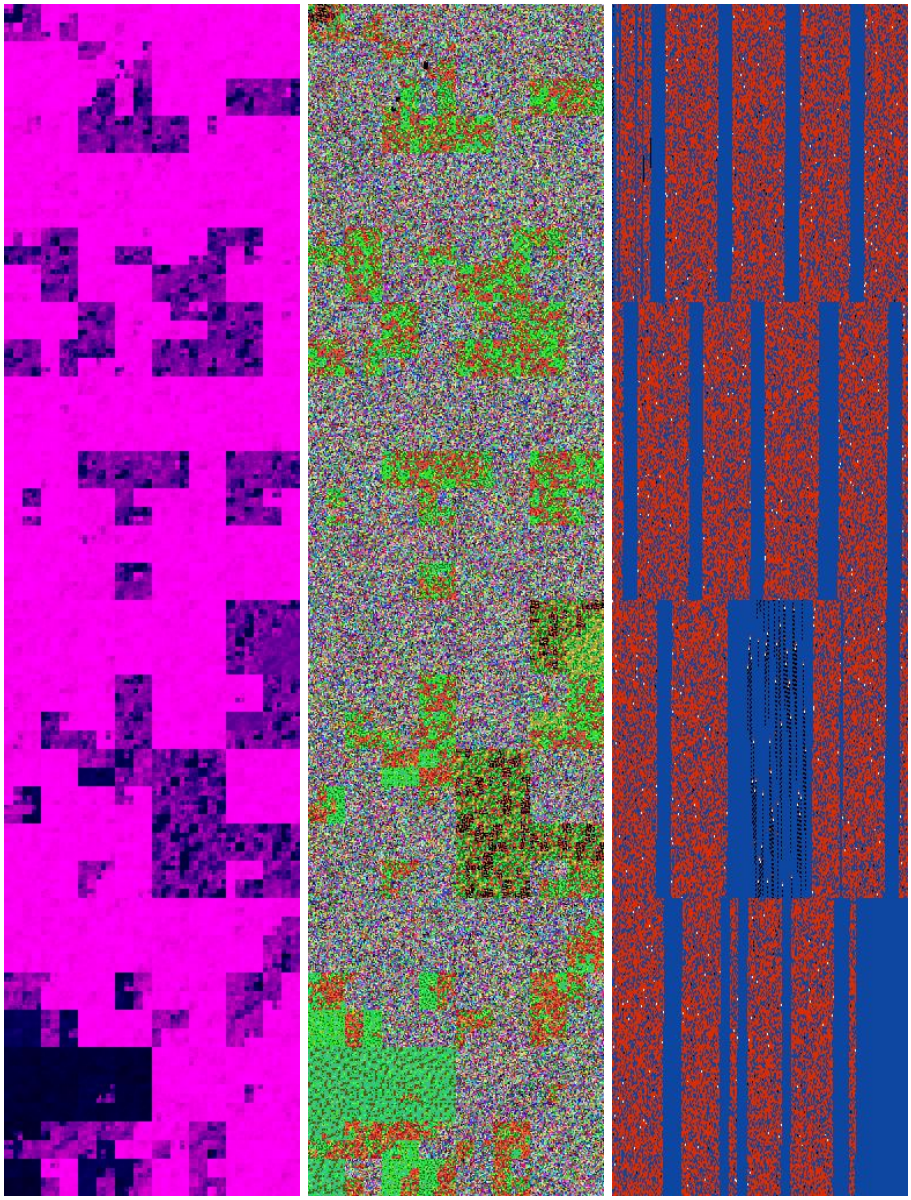

Figure 9: Sample 2 visualizations of what a mp3 audio file looks like.

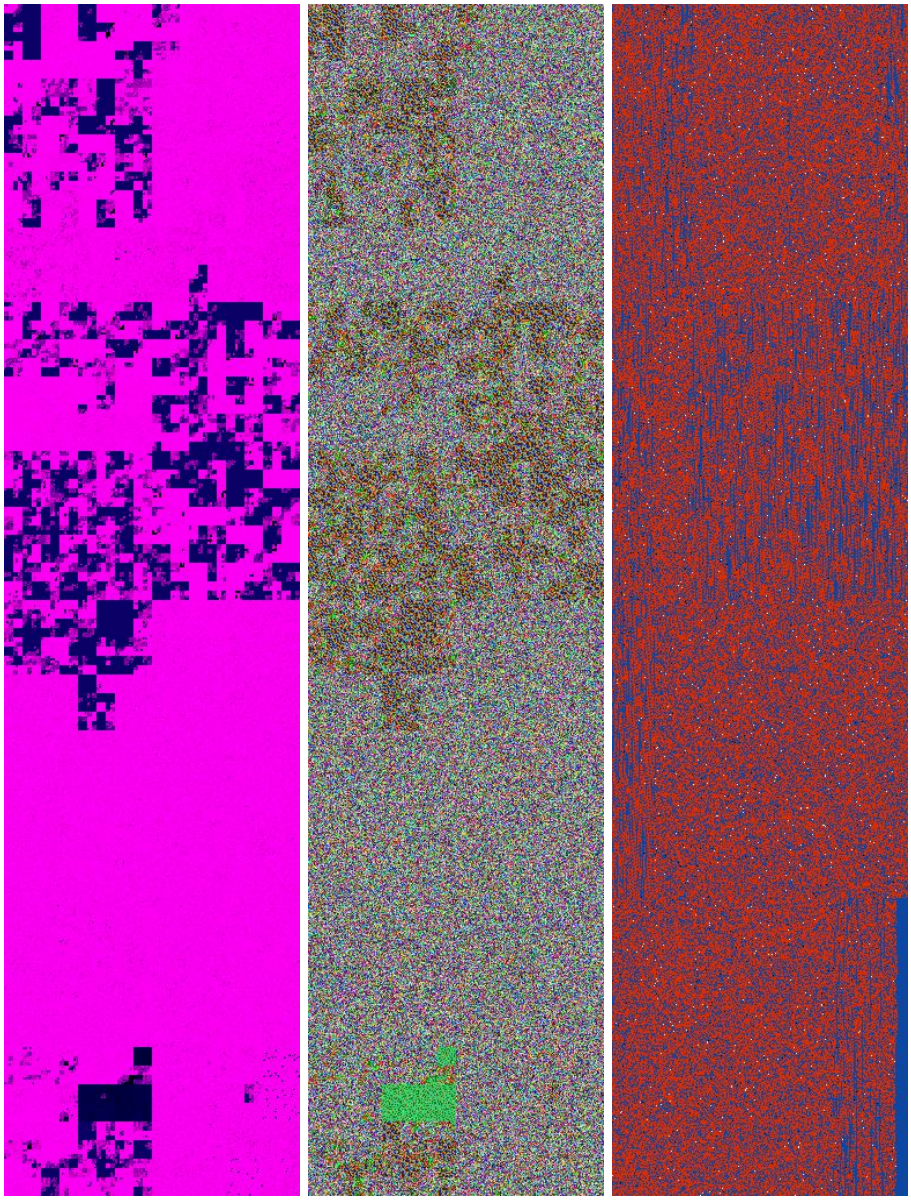

Figure 10: Sample 3 visualizations of what a mp3 audio file looks like.

3.2.4 Non-Malicious Executables (Windows)

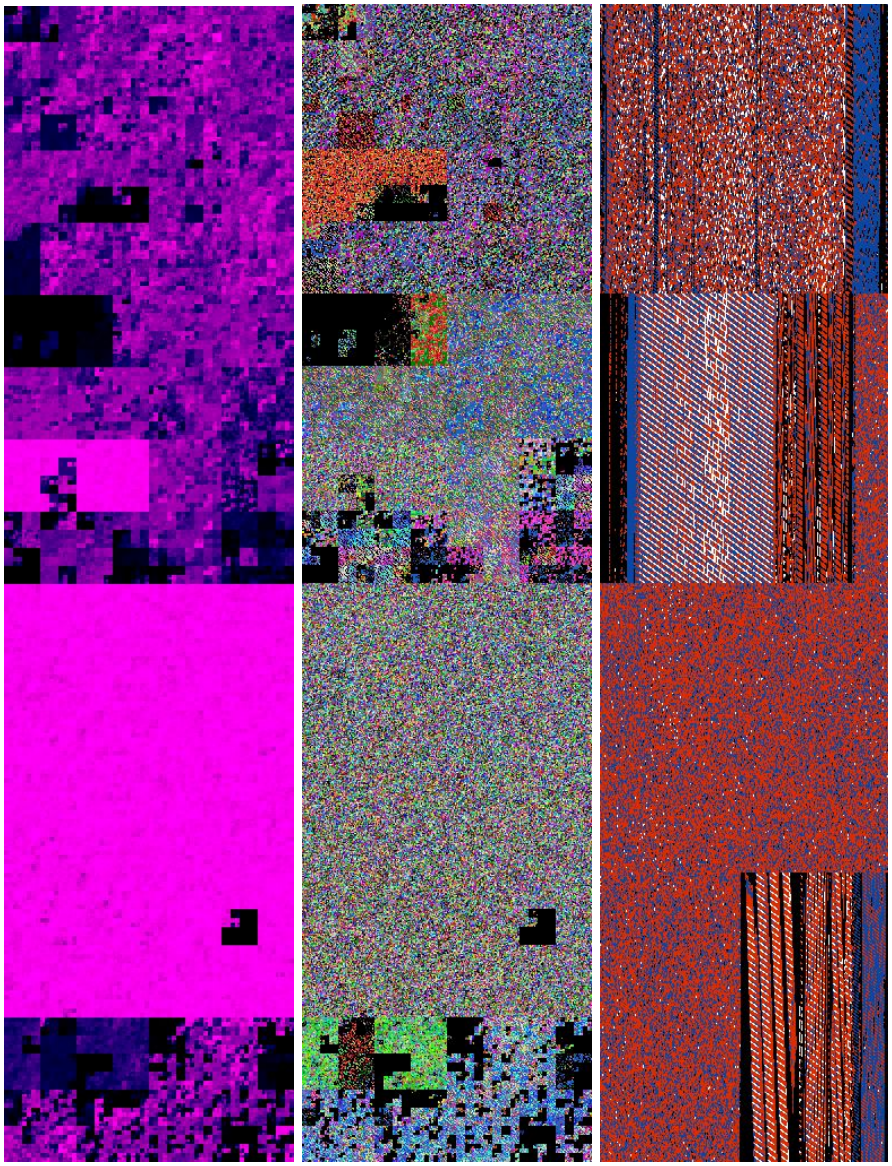

Figure 11: Sample 1 visualizations of what a Windows .exe file looks like.

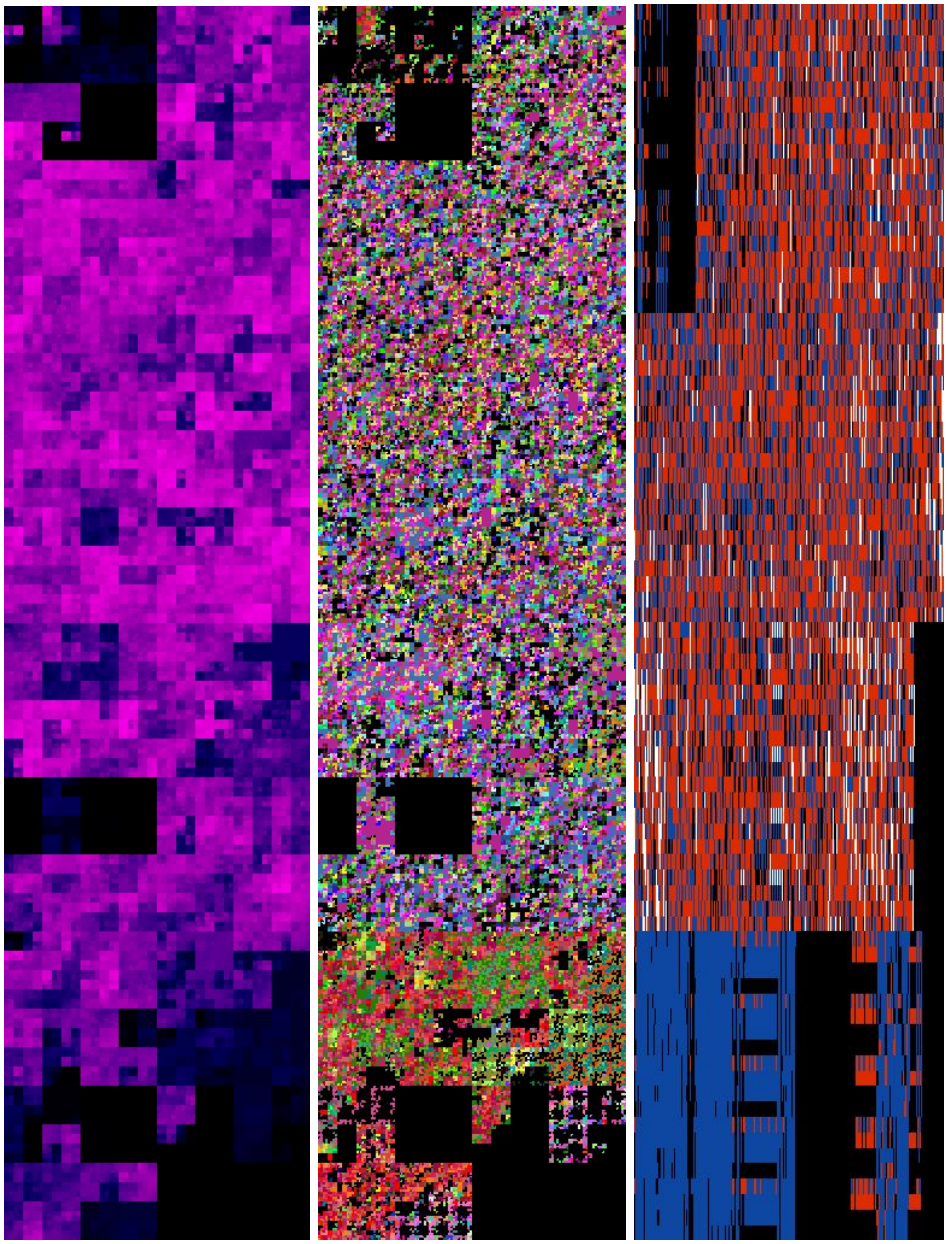

Figure 12: Sample 2 visualizations of what a Windows .exe file looks like.

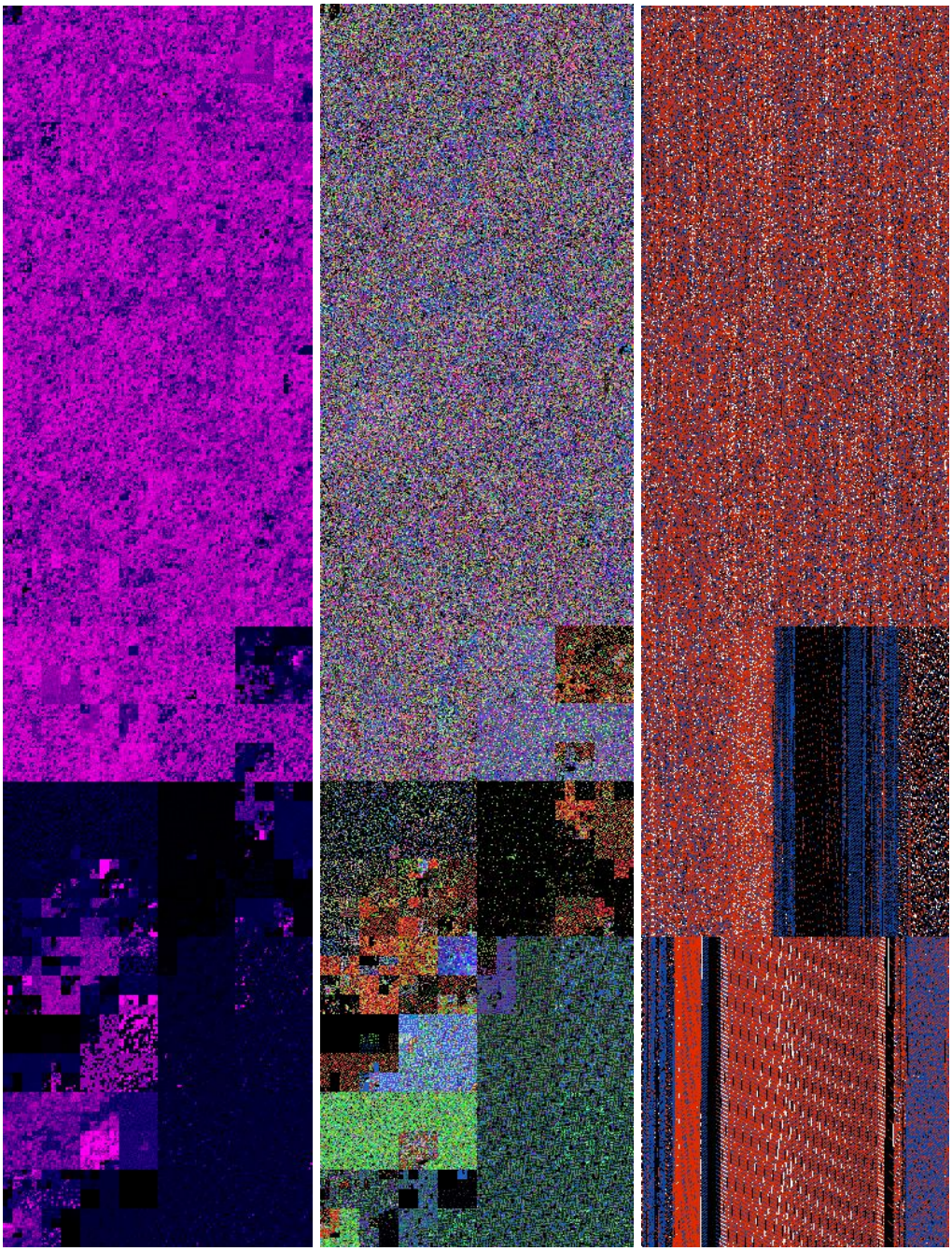

Figure 13: Sample 3 visualizations of what a Windows .exe file looks like.

3.2.5 Non-Malicious Executables (Linux)

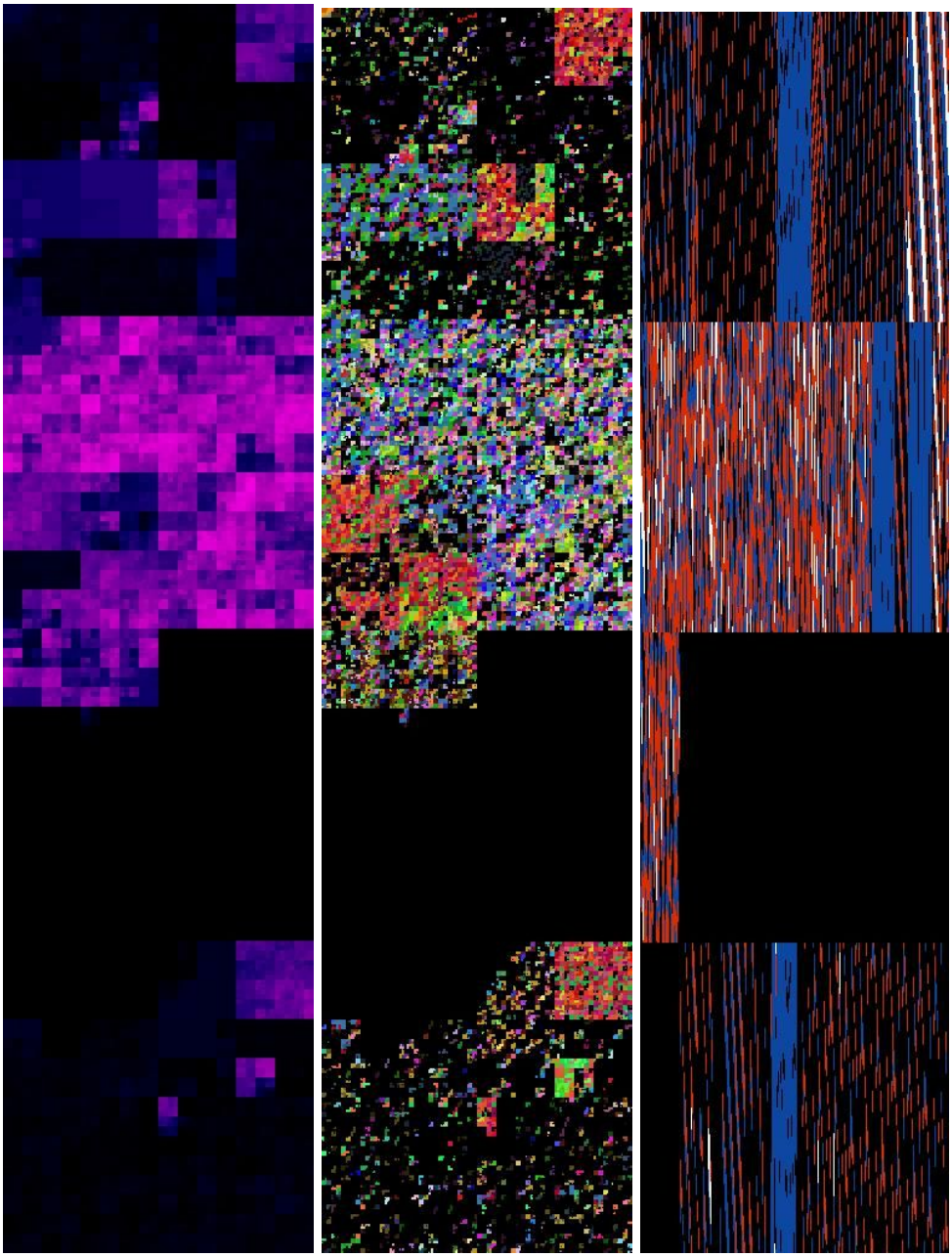

Figure 14: Sample 1 visualizations of what a Linux Binary file looks like.

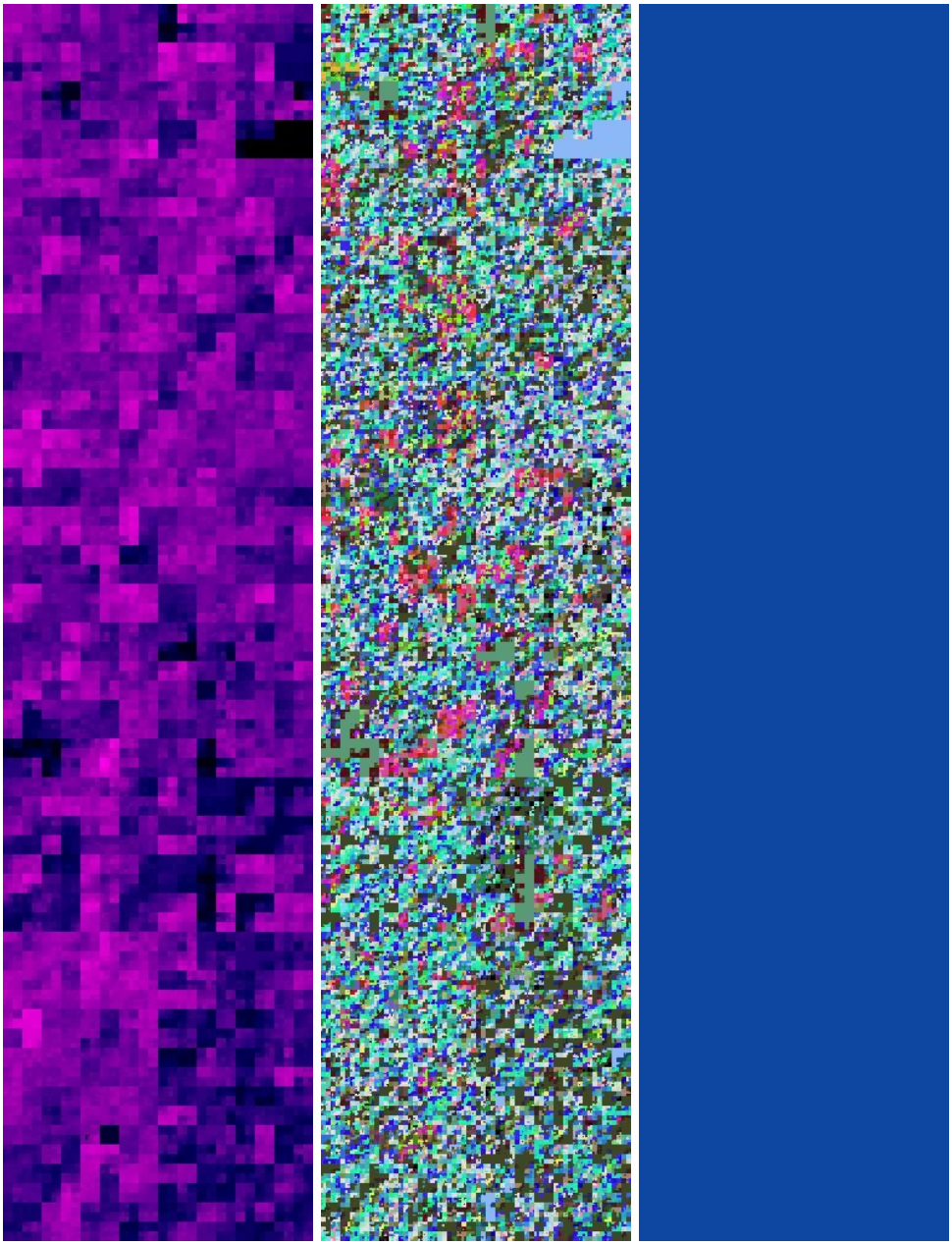

Figure 15: Sample 2 visualizations of what a Linux Binary file looks like.

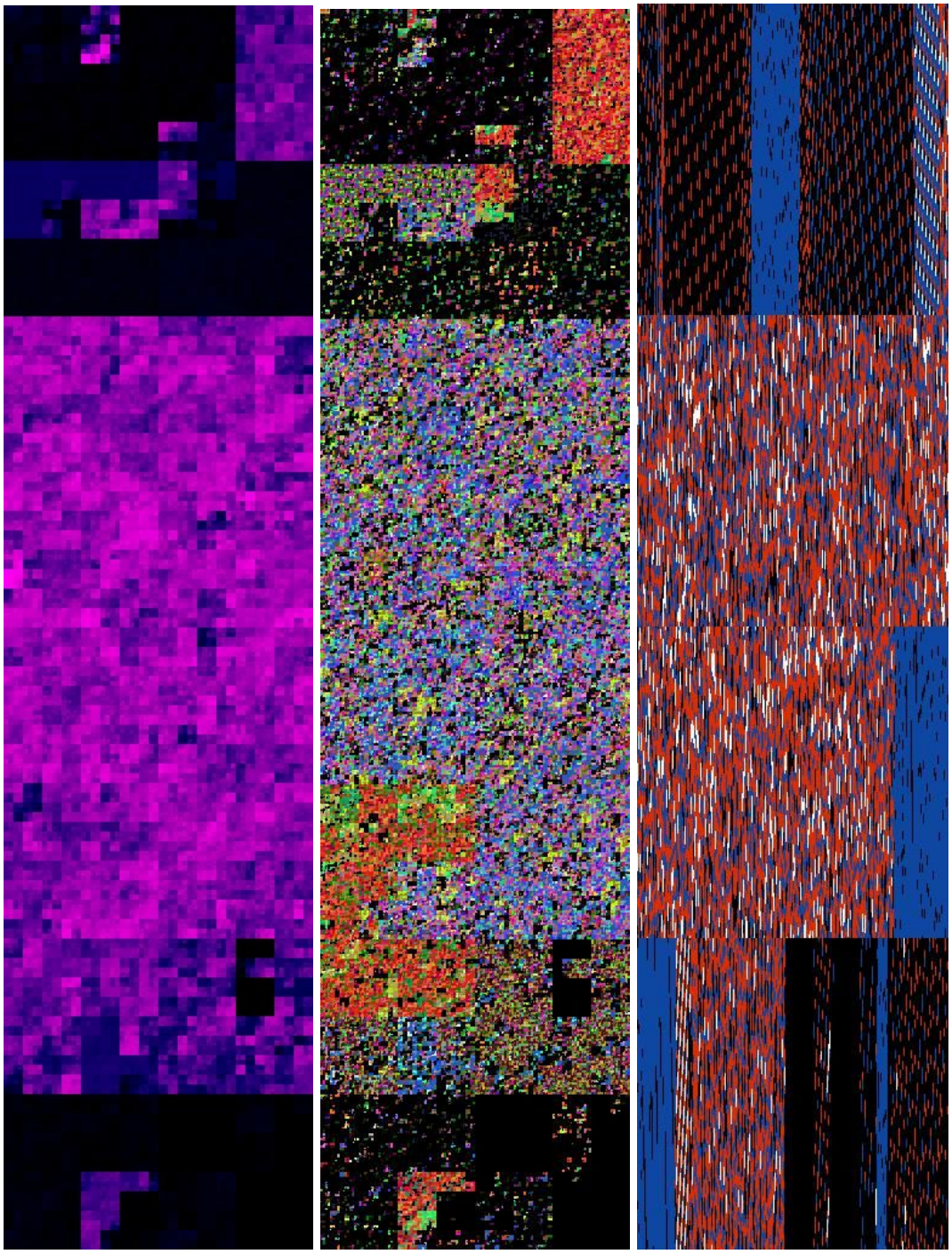

Figure 16: Sample 3 visualizations of what a Linux Binary file looks like.

3.2.6 Malicious Executables

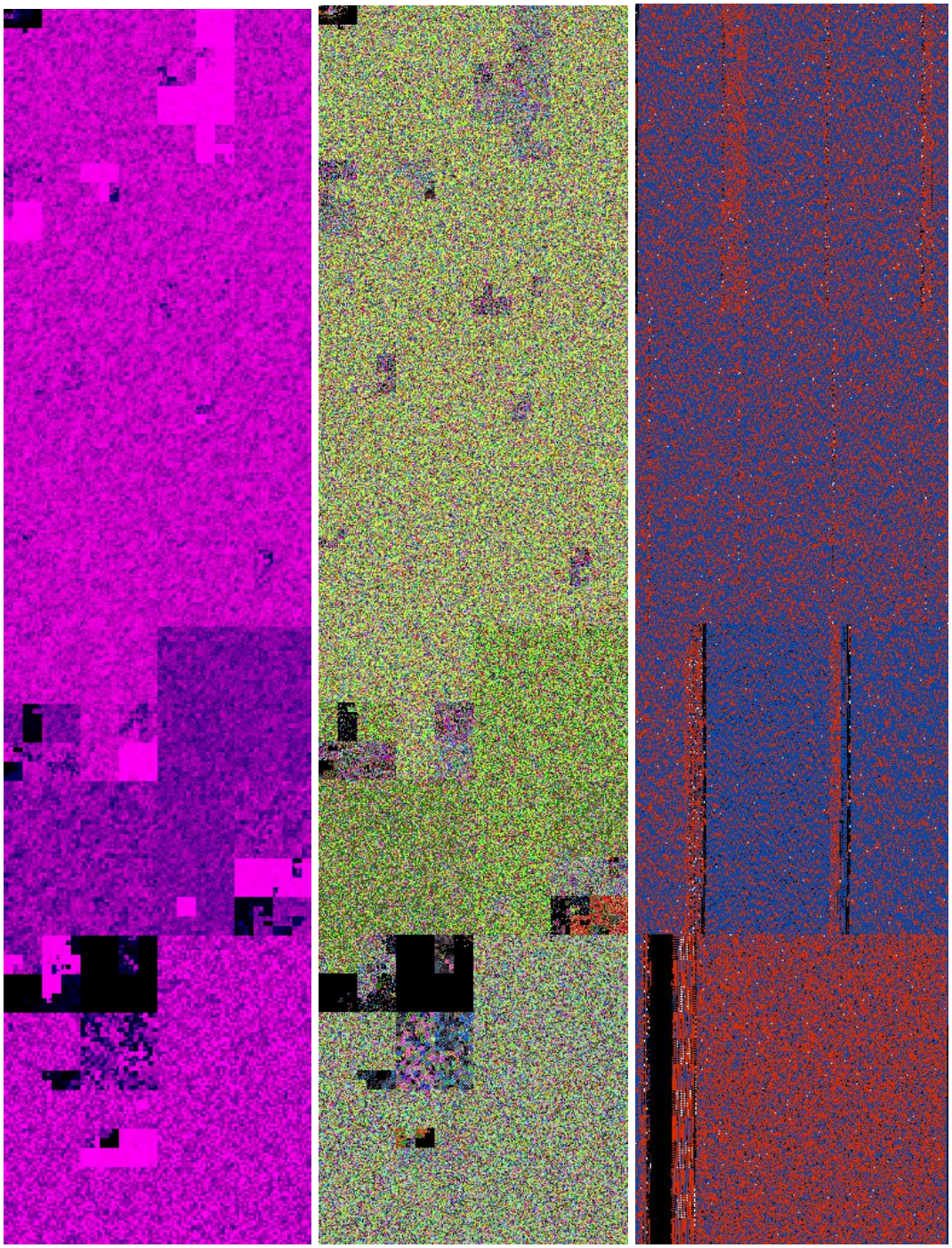

Figure 17: Sample 1 visualizations of what a Malicious Windows Executable file looks like.

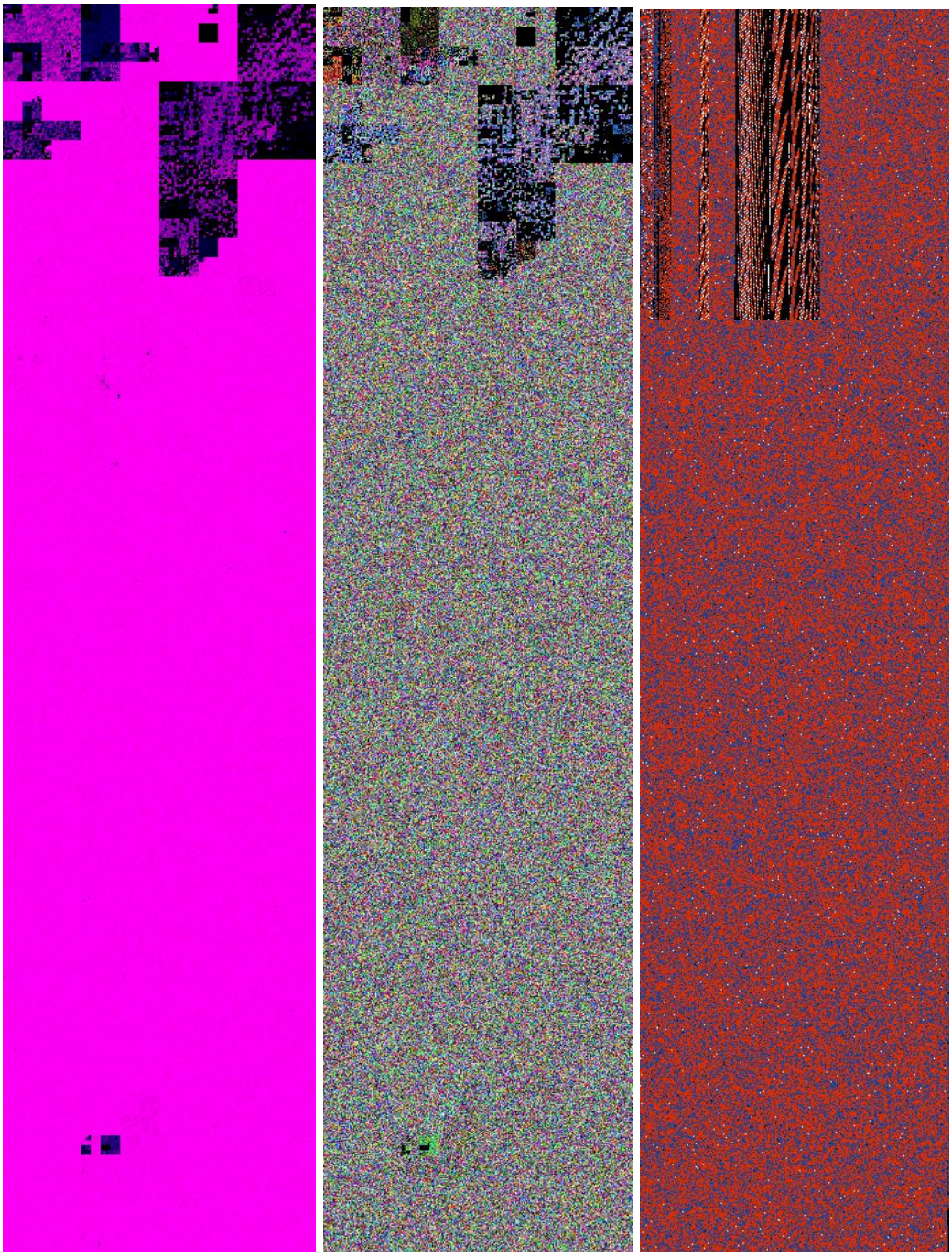

Figure 18: Sample 2 visualizations of what a Malicious Windows Executable file looks like.

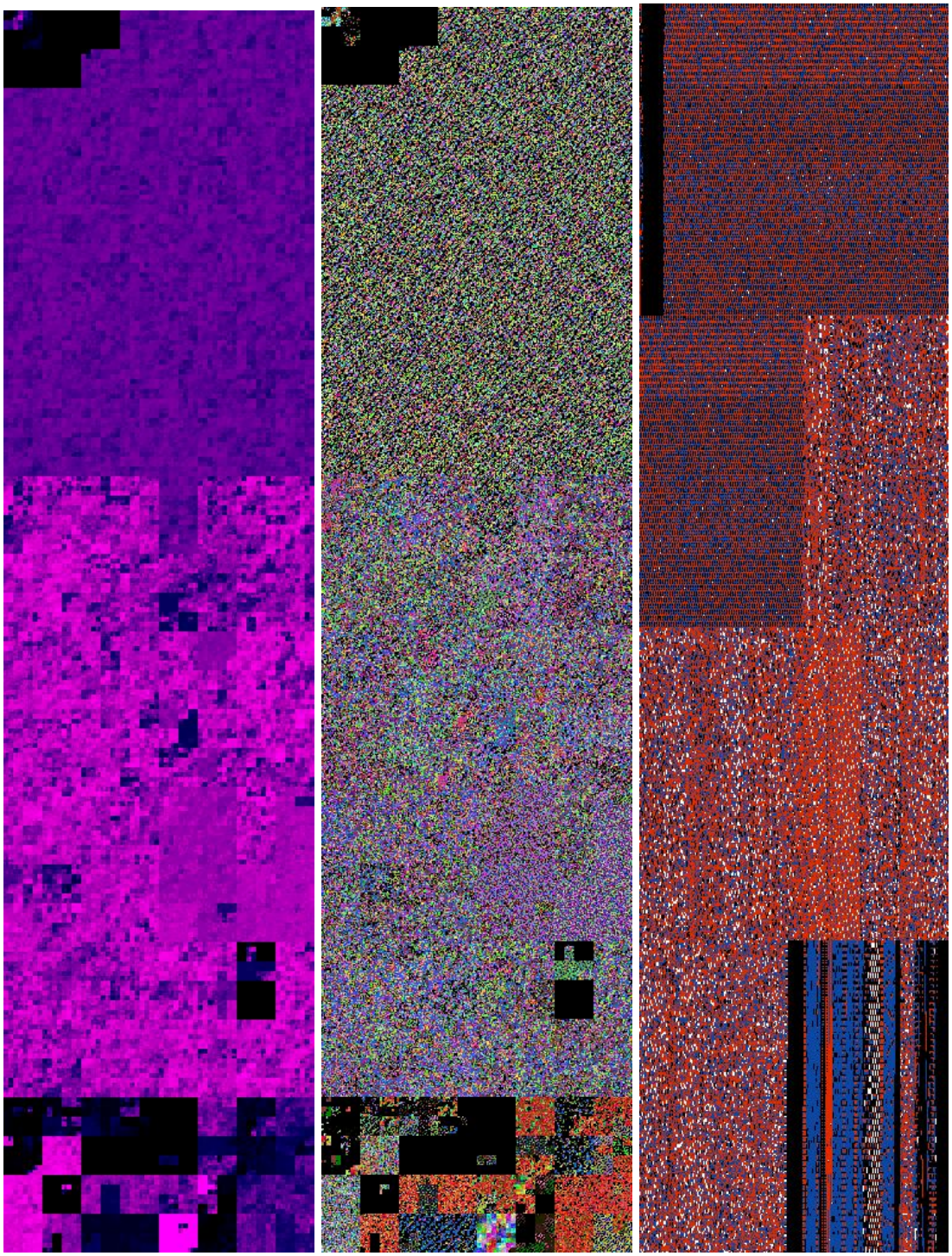

Figure 19: Sample 3 visualizations of what a Malicious Windows Executable file looks like.

3.3 Image Classification with Machine Learning

Over the last several years the study of machine learning has made significant progress in numerous application areas, one of which has been image recognition and classification. Convolutional neural networks, a structure of machine learning model has been found to achieve high performance on visual recognition that can match and sometimes exceed human performance. The Tensorflow Inception image recognition neural network supports transfer learning, which only requires the final layer of the neural net to be retrain, this technique that removes the necessity to fully train a model and works well for many image classification applications.

We used three main models to train on our data which consisted of about 1000 samples of various file types. The first model, version 1, included: graphic files, audio files, and portable document format files, linux binaries, and windows binaries. The second model, version 2, included the previously mentioned files and added on windows malware. The third model, version 3, consisted only of linux binaries, windows binaries, and windows executable malware. Furthermore, we created version 1.1, 2.1, and 3.1 which included the same datasets but increased the number of training steps to 8000 in order to determine if an increase in training steps would result in better classification accuracy for various models. The results of our finding can be found in Chapter 5.

3.3.1 TensorFlow

Following the Tensorflow installation tutorial, downloads and installs all the prerequisites needed to retrain the final layer of Inception for image classification. To Retrain the model we gathered the generated visualization images for training into directories named after their class and passed the directory to the retrain script. The following images show the directory classes containing visualizations, and the retrain script running with 12517 entropy visualization images as input.

3.3.2 Server Specifications and Setup

Retraining was performed on a Digital Ocean Virtual Private Server (VPS) with 2 virtual cores, 80gb SSD, 4GB of memory and took approximately 2 hours to complete.

```
root@rearp15-ml:/home/science/visualization_output/ent# ls
image  music  pdf  winexe
root@rearp15-ml:/home/science/visualization_output/ent# ls image/ | head
-5
Abyssinian_101_ent.jpg
Abyssinian_117_ent.jpg
Abyssinian_119_ent.jpg
Abyssinian_121_ent.jpg
Abyssinian_123_ent.jpg
root@rearp15-ml:/home/science/visualization_output/ent# python3 tensorflo
w/examples/image_retraining/retrain.py --image_dir /home/science/visualiz
ation_output/ent/
```

Figure 20: Each of the directories listed contain images to be classified, the directory is named based on the class of image within the directory. Finally the retrain command is run specifying the directory containing each of the class directories.

```
root@rearp15-ml:/home/science/tensorflow# time python3 tensorflow/examples/image_retraining/retrain.py --image_dir /home/science/visualization
_output/ent/
Not extracting or downloading files, model already present in disk
Model path:  /tmp/imagenet/classify_image_graph_def.pb
INFO:tensorflow:Looking for images in 'image'
INFO:tensorflow:Looking for images in 'pdf'
INFO:tensorflow:Looking for images in 'music'
INFO:tensorflow:Looking for images in 'winexe'
2018-03-27 21:49:04.033925: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary wa
s not compiled to use: AVX2 FMA
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/winexe/Chess_ent.jpg_inception_v3.txt
2018-03-27 21:49:05.014972: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithGlobalNormalization is deprecated. It will cease t
o work in GraphDef version 9. Use tf.nn.batch_normalization().
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/winexe/rm_ent.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/winexe/bthudtask_ent.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/winexe/InetMgr_ent.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/winexe/bcdboot_ent.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/winexe/WindowsAnytimeUpgradeui_ent.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/winexe/pgawk_ent.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/winexe/bootcfg_ent.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/winexe/w32tm_ent.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/winexe/ieUnatt_ent.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/winexe/perl5_ent.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/winexe/tzutil_ent.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/winexe/BitLockerWizard_ent.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/winexe/ComSvcConfig_ent.jpg_inception_v3.txt
```

Figure 21: Running the retrain script on the Entropy visualizations. Creating bottleneck values is the layer just before the final output layer in classification.

```
INFO:tensorflow:2018-03-27 23:42:31.844936: Step 3999: Train accuracy = 98.0%
INFO:tensorflow:2018-03-27 23:42:31.845824: Step 3999: Cross entropy = 0.084711
INFO:tensorflow:2018-03-27 23:42:32.088613: Step 3999: Validation accuracy = 99.0% (N=100)
INFO:tensorflow:Final test accuracy = 99.0% (N=388)
INFO:tensorflow:Froze 2 variables.
Converted 2 variables to const ops.

real    113m40.059s
user    119m34.952s
sys     7m17.176s
root@rearp15-ml:/home/science/tensorflow#
root@rearp15-ml:/home/science/tensorflow#
```

Figure 22: Retraining took just under 120 minutes to complete in this instance.

Once training is completed we can pass an image that was not used in the training process to be classified. Tensorflow provides a label_image script that takes the trained graph, and trained labels as input, as well as the image intended for classification. The script only takes a few seconds to run for a single classification.

```
root@rearp15-ml:/home/science/tensorflow# python3 tensorflow/examples/label_image/label_image.py --graph=/tmp/output_graph.pb --labels=/tmp/ou
tput_labels.txt --input_layer=Mul --output_layer=final_result --input_mean=128 --input_std=128 --image=/home/science/Russian_Blue_38_ent.jpg
2018-03-27 23:45:51.562732: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary wa
s not compiled to use: AVX2 FMA
2018-03-27 23:45:52.734497: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithGlobalNormalization is deprecated. It will cease t
o work in GraphDef version 9. Use tf.nn.batch_normalization().
image 0.9978387
winexe 0.0013192457
pdf 0.00059561734
music 0.00024651844
root@rearp15-ml:/home/science/tensorflow#
```

Figure 23: An entropy visualization of a .JPG image that was not used in the training process was passed to the classify script. The retrained model correctly classified the image with an estimate accuracy of 99.78387%.

The training was completed three times, one for each visualization type, Natural, Hilbert Curve and Entropy, each model was saved for use in the experimentation phase for comparison against each other.

4.0 Implementation

4.1 Retraining

```
science@rearp15-ml:~/retrained_models$
science@rearp15-ml:~/retrained_models$
science@rearp15-ml:~/retrained_models$
science@rearp15-ml:~/retrained_models$ cat retrain.sh
#!/bin/bash

selectedModel=$1
version=$2

if [ $selectedModel == "entropy" ]; then
        model="/home/science/retrained_models/ent_inceptionv3_"
        imageDir="ent_mal"
elif [ $selectedModel == "natural" ]; then
        model="/home/science/retrained_models/nat_inceptionv3_"
        imageDir="nat_mal"
elif [ $selectedModel == "hilbert" ]; then
        model="/home/science/retrained_models/hil_inceptionv3_"
        imageDir="hil_mal"
else
        echo "Invalid Model Selected. Please select model type: entropy, natural or hilbert."
fi


echo "Retraining: "$model$version

python3 /home/science/tensorflow/tensorflow/examples/image_retraining/retrain.py --output_graph=$model$version"/output_graph.pb" --output_labels=$model
$version"/output_labels.txt" --bottleneck_dir=$model$version"/bottleneck/" --model_dir=$model$version --image_dir="/home/science/visualization_output/"
$imageDir"/" --summaries_dir=$model$version"/logs/" --how_many_training_steps=4000
science@rearp15-ml:~/retrained_models$
```

Figure 24: Retrain Script.

4.2 Classifying

```
science@rearp15-ml:~/retrained_models$
science@rearp15-ml:~/retrained_models$
science@rearp15-ml:~/retrained_models$
science@rearp15-ml:~/retrained_models$ cat classify.sh
#!/bin/bash

selectedModel=$1
version=$2
classifyImage=$3

if [ $selectedModel == "entropy" ]; then
        model="/home/science/retrained_models/ent_inceptionv3_"
elif [ $selectedModel == "natural" ]; then
        model="/home/science/retrained_models/nat_inceptionv3_"
elif [ $selectedModel == "hilbert" ]; then
        model="/home/science/retrained_models/hil_inceptionv3_"
else
        echo "Invalid Model Selected. Please select model type: entropy, natural or hilbert."
fi


echo "Using model: "$model$version"/outputgraph.pb to classify "$classifyImage

python3 /home/science/tensorflow/tensorflow/examples/label_image/label_image.py --graph=$model$version"/output_graph.pb" --labels=$model$version"/outpu
t_labels.txt" --input_layer=Mul --output_layer=final_result --input_mean=128 --input_std=128 --image=$classifyImage
science@rearp15-ml:~/retrained_models$
```

Figure 25: Classify Script.

5.0 Experiment

This section covers the experiments we performed, namely results from the classifying our data on our different models. We describe our setup for the experiments, state our hypothesis and what we expect our results to be, and we examine scenarios to confirm our hypothesis. Finally, we summarize our findings and close with our thoughts.

5.1 Experiment Setup

This section gives a detailed description of how we setup our experiments including the number of samples we used of each file type, server-client setup for capturing network traffic in a controlled environment, tools we used to extract the the HTTP objects from the packet capture files, and finally the process of generating and classifying the visualizations using our models.

The machine learning was done on our Digital Ocean VPS server with: 2 virtual cores, 80gb Solid State Drive, and 4GB of RAM. For our use case, this option had the storage capacity and CPU requirements needed to perform the machine learning in a reasonable time. The file extraction and visualization were done on a local machine running *Ubuntu 16* in a Virtual Machine.

5.1.1 Collecting Samples

For machine learning to be trained accurately a high number of samples is required, often in the hundreds for basic small scale trials and ideally well into thousands of samples. For our experimentation we collected 6800 samples of various files types. The breakdown is as follows:

- JPEG/Image files
  - 1000
- Music
  - 875
- Portable Document Format
  - 1170
- Windows Portable Executables
  - 1126
- Linux Binaries
  - 1184
- Malicious Executables
  - 1472

5.1.2 Creating and capture network traffic

A web server was created to host each of the samples so that network traffic captures could be performed when a client is downloading the files from the server. The packet capture can then be parsed and extract all of the artifacts (files). To generate and capture the traffic a simple *bash* script was created. The network traffic capture is performed with *tcpdump* and the web traffic is generated using *curl*.

5.1.3 Parsing and extracting artifacts (files) from network traffic

One of the outcomes of our project is to create a command line tool that can parse packet captures to extract files and create binary visualizations. We determined that *tshark*, a command line network protocol analyzer that comes with any default install of *wireshark* and is capable of capturing packet data, replaying packet captures and performing various analysis on the traffic inside a network capture file, would be the easiest to use.

We utilize tshark for parsing capabilities, as it has an export-objects option that will extract various objects such as files transferred over HTTP, FTP, as well as reconstruct VOIP calls. However, this method does come with some limitations. First, the latest version of *tshark* does not allow us to choose which files we want to extract, although the documentation hints that this may be a future functionality. Second,while there are *Python* libraries that are wrappers for *tshark*, they are very limited in their capabilities and do not have the ability to export objects, leaving us to have to do manually. While the first limitation will not be addressed in our implementation, the second limitation can easily be solved by using *Python*'s *subprocess* library which is an easy to use and flexible library for spawning processes.

Our script provides the *parse* command line option and accepts an input pcap file and an output directory to dump the exported objects.

```
tz@ubuntu:~/Documents/maml-scripts/parser$ cat parser.py
import os
import sys
import subprocess

System Settings r(object):

        Functions to take a pcap file and extract the sessions from
        it, outputting it to a file
    """

    def __init__(self, data=None):
        self.data = data


    def extract_http_sessions(self, pcap_input, out_directory):
        p = subprocess.Popen(["tshark","-r", pcap_input, "--export-objects", "http,"+out_directory]).wait()


tz@ubuntu:~/Documents/maml-scripts/parser$
```

Figure 26: The script takes an input pcap extracts the HTTP objects to the designated output directory.

```
tz@ubuntu:~/Documents$ time python maml-scripts/maml.py parse -i /home/tz/Downloads/ie_aurora/ie_aurora_exploitWin7.pcap -o output
    1   0.000000 192.168.146.1 → 192.168.146.131 TCP 66 51314 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
    2   0.001052 192.168.146.131 → 192.168.146.1 TCP 66 80 → 51314 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 SACK_PERM=1 WS=32
    3   0.001129 192.168.146.1 → 192.168.146.131 TCP 54 51314 → 80 [ACK] Seq=1 Ack=1 Win=65700 Len=0
    4   0.002462 192.168.146.1 → 192.168.146.131 HTTP 572 GET / HTTP/1.1
    5   0.004090 192.168.146.131 → 192.168.146.1 TCP 54 80 → 51314 [ACK] Seq=1 Ack=519 Win=6912 Len=0
    6   1.035772 192.168.146.131 → 192.168.146.1 TCP 1514 HTTP/1.1 200 OK  [TCP segment of a reassembled PDU]
    7   1.036036 192.168.146.131 → 192.168.146.1 TCP 1514 80 → 51314 [ACK] Seq=1461 Ack=519 Win=6912 Len=1460 [TCP segment of a reassembled PD
U]
    8   1.036067 192.168.146.1 → 192.168.146.131 TCP 54 51314 → 80 [ACK] Seq=519 Ack=2921 Win=65700 Len=0
    9   1.036232 192.168.146.131 → 192.168.146.1 TCP 1514 80 → 51314 [ACK] Seq=2921 Ack=519 Win=6912 Len=1460 [TCP segment of a reassembled PD
U]
   10   1.036535 192.168.146.131 → 192.168.146.1 TCP 1514 80 → 51314 [ACK] Seq=4381 Ack=519 Win=6912 Len=1460 [TCP segment of a reassembled PD
U]
   11   1.036570 192.168.146.1 → 192.168.146.131 TCP 54 51314 → 80 [ACK] Seq=519 Ack=5841 Win=65700 Len=0
   12   1.036787 192.168.146.131 → 192.168.146.1 HTTP 156 HTTP/1.1 200 OK  (text/html)
   13   1.115506 192.168.146.1 → 192.168.146.131 HTTP 490 GET /ASXHgUIrZAbwfJXPqVHDxGgtqQKmqULMTxjHNTSxRdOSxvXWIfteGfzKNBKdQgPHlAZaCDQIykPrtcR
MOhudEyvPWSZkuQDMNUC.gif HTTP/1.1
   14   1.115923 192.168.146.131 → 192.168.146.1 TCP 54 80 → 51314 [ACK] Seq=5943 Ack=955 Win=8000 Len=0
   15   1.320113 192.168.146.131 → 192.168.146.1 HTTP 201 HTTP/1.1 200 OK  (GIF89a)
   16   1.520104 192.168.146.1 → 192.168.146.131 TCP 54 51314 → 80 [ACK] Seq=955 Ack=6090 Win=65448 Len=0
   17  13.918537 192.168.146.1 → 192.168.146.131 TCP 54 51314 → 80 [RST, ACK] Seq=955 Ack=6090 Win=0 Len=0

real    0m0.387s
user    0m0.044s
sys     0m0.144s
tz@ubuntu:~/Documents$
```

Figure 27: Example of the script running and the output from the script.

```
tz@ubuntu:~/Documents/output$ ls
%2f
%2f(1)
ASXHgUIrZAbwfJXPqVHDxGgtqQKmqULMTxjHNTSxRdOSxvXWIfteGfzKNBKdQgPHlAZaCDQIykPrtcRMOhudEyvPWSZkuQDMNUC.gif
_Hye44r5jDk.js
images%3fq=tbn%3aANd9GcRLrhKw25fiUk4TRtgDEUvd50jcq7rDlfZWAnPFtVbYv8e0b29nAg2j
images%3fq=tbn%3aANd9GcRp70F8sUxll4rS77BqBudf9oE3zETOAsmgCiOlmeOUbDJyNaqtlUwb
images%3fq=tbn%3aANd9GcRqmCDG9AvHhWAp95irbQfb8cayYW49VFrxznpI9BZfPr_RuKRqfureLA
images%3fq=tbn%3aANd9GcTfeBaXae2K7DFL28qQ3ztmNw8n7gLfVDPnVZ-R3bEgV8Tvs9Q1P5oo
jNEnVtXTHfvqDpsgnQOAGQJaSzSNfGmovBbmOJJBIIZpbCqjqthaK.gif
SdYBs6GkBMtsDkULa93wPx0hL04UBi6oixWxouSvNqn-Whxxl4bsOfesh82PZYF49s1APQ=s109
SdYBs6GkBMtsDkULa93wPx0hL04UBi6oixWxouSvNqn-Whxxl4bsOfesh82PZYF49s1APQ=s30
search%3ftbs=sbi%3aAMhZZitsihTY3DBPKxKwRNm9nDV6V3JwxzskmdQaoHUWhJOLQ5grqTu7MuQssbFl7Q_16SbW_18ooWSLlyJUMOtNPCvYAJ12uSAl5OQv-ionD9adKX9xXv1PxBK
Po597L7XP1vg0wz0F6xMmZd553yzYQmBOr6SXWH7iIuINEI_1r96g7Zjs8CzPRyUP-875TZdIEVdO3kqK85ho84F6iV8wnjBnqFgkRF27l
upload
upload(1)
tz@ubuntu:~/Documents/output$
```

Figure 28: Example objects extracted from the pcap file.

5.1.4 Creating visualizations

The command line tool can be used to generate each of the three visualization types, natural, entropy and hilbert by passing the keyword “visualize”, the input file to be visualized, output visualization file name and lastly the visualization type by passing one of the visualization type keywords or the word “all” for all three. Generating all three visualizations will result in three files with the visualization type appended to the file name.

```
tz@ubuntu:~/Documents$ time python maml-scripts/maml.py visualize -i Russian_Blue_38.jpg -o ./Russian_Blue_38 -V all

real    0m25.219s
user    0m25.132s
sys     0m0.040s
tz@ubuntu:~/Documents$
```

Figure 29:The above figure shows the visualization generation of three visualization types on a single JPEG image.

```
root@ubuntu:/home/tz/Documents# time ./maml-bulkgen.sh test/images_top250ls.txt /home/tz/Documents/test/output/ all
DONE!
[1]+  Killed                  ./maml-bulkgen.sh test/images_top250ls.txt /home/tz/Documents/test/output/ all

real    106m48.631s
user    106m43.320s
sys     0m3.772s
root@ubuntu:/home/tz/Documents#
```

Figure 30: The above figure shows the visualization generation of three visualization types on 250 JPEG images.

```
tz@ubuntu:~/Documents$ time ./maml-bulkgen.sh data/pdflist.txt data/pdf_output/ all
DONE!
[2]+  Killed                  ./maml-bulkgen.sh data/pdflist.txt data/pdf_output/ all

real    511m12.773s
user    510m34.216s
sys     0m20.708s
tz@ubuntu:~/Documents$
```

Figure 31: The above figure shows the visualization generation of three visualization types on 1100 PDF documents.

5.2 Hypothesis

Different file types structure data in various different ways, we believe that the visualization types we have selected can be used in tandem with machine learning to reliably and accurately identify the file types entirely based on the summary visualizations.

5.3 Observations/Results

In order to retrain the InceptionV3 image classifier the generated visualizations were grouped based on the visualization type and the class/type of file. A directory was created for each visualization type, entropy, natural and hilbert. Within each visualization directory was a subdirectory for each of the classes with their respective visualizations in it. Retraining and classification was performed several times to cross compare accuracy with different classes/file types and variations to the number of training steps.

The First machine learning model, called version 1.0 and version 1.1 contained the following classes: Images (JPEG files), PDFs, MP3, Windows Executables and Linux executables/binaries.

The second model, version 2.0 and 2.1 contain the same classes as model 1.0 with the addition of malware. For all model version x.0 use the default InceptionV3 training steps of

4000. Version x.1 uses 8000 training steps, which in theory should improve the accuracy of the trained model.

The third model was trained on the following classes: Windows executables, malicious executables and linux executables/binaries.

5.3.1 Model Version 1

```
INFO:tensorflow:2018-03-28 18:58:47.395621: Step 3999: Cross entropy = 0.084390
INFO:tensorflow:2018-03-28 18:58:47.686804: Step 3999: Validation accuracy = 96.0% (N=100)
INFO:tensorflow:Final test accuracy = 97.3% (N=510)
INFO:tensorflow:Froze 2 variables.
Converted 2 variables to const ops.

real    115m34.235s
user    158m12.704s
sys     21m42.020s
root@rearp15-ml:/home/science/retrained_models# time ./retrain.sh entropy
```

Figure 32: Entropy Retraining.

```
INFO:tensorflow:2018-03-28 21:44:22.839377: Step 3999: Train accuracy = 100.0%
INFO:tensorflow:2018-03-28 21:44:22.839747: Step 3999: Cross entropy = 0.061739
INFO:tensorflow:2018-03-28 21:44:23.021495: Step 3999: Validation accuracy = 97.0% (N=100)
INFO:tensorflow:Final test accuracy = 95.9% (N=536)
INFO:tensorflow:Froze 2 variables.
Converted 2 variables to const ops.

real    98m6.753s
user    138m36.568s
sys     15m31.484s
root@rearp15-ml:/home/science/retrained_models#
```

Figure 33: Hilbert Retraining.

```
INFO:tensorflow:2018-04-08 02:47:38.323318: Step 7999: Train accuracy = 87.0%
INFO:tensorflow:2018-04-08 02:47:38.324014: Step 7999: Cross entropy = 0.269833
INFO:tensorflow:2018-04-08 02:47:38.519394: Step 7999: Validation accuracy = 88.0% (N=100)
INFO:tensorflow:Final test accuracy = 81.4% (N=597)
INFO:tensorflow:Froze 2 variables.
Converted 2 variables to const ops.

real    96m52.127s
user    140m58.320s
sys     9m56.664s
root@rearp15-ml:/home/science/retrained_models# time ./retrain.sh natural 1.0
```

Figure 34: Natural Retraining.

```
science@rearp15-ml:~/retrained_models$ time ./classify.sh entropy 1.0 ../visualization_output/ent/image/english_cocker_spaniel_24_ent.jpg
Using model: /home/science/retrained_models/ent_inceptionv3_1.0/outputgraph.pb to classify ../visualization_output/ent/image/english_cocker_spaniel_24_
ent.jpg
2018-04-14 18:44:41.913868: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not com
piled to use: AVX2 FMA
2018-04-14 18:44:42.667075: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithGlobalNormalization is deprecated. It will cease to work in
 GraphDef version 9. Use tf.nn.batch_normalization().
image 0.9981781
winexe 0.0011031299
pdf 0.00051101693
music 0.00020146162
linux binaries 6.341765e-06

real    0m7.484s
user    0m7.200s
sys     0m1.108s
science@rearp15-ml:~/retrained_models$
```

Figure 35: Entropy - Image (JPEG) classification.

```
science@rearp15-ml:~/retrained_models$ time ./classify.sh hilbert 1.0 ../visualization_output/hil/image/english_cocker_spaniel_24_hil.jpg
Using model: /home/science/retrained_models/hil_inceptionv3_1.0/outputgraph.pb to classify ../visualization_output/hil/image/english_cocker_spaniel_24_
hil.jpg
2018-04-14 18:51:35.244985: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not com
piled to use: AVX2 FMA
2018-04-14 18:51:36.053709: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithGlobalNormalization is deprecated. It will cease to work in
 GraphDef version 9. Use tf.nn.batch_normalization().
image 0.99693525
pdf 0.002251963
linux binaries 0.00036836966
winexe 0.00036006124
music 8.42079e-05

real    0m7.051s
user    0m6.752s
sys     0m1.184s
science@rearp15-ml:~/retrained_models$
```

Figure 36: Hilbert - Image (JPEG) classification.

```
science@rearp15-ml:~/retrained_models$ time ./classify.sh natural 1.0 ../visualization_output/nat/image/english_cocker_spaniel_24_nat.jpg
Using model: /home/science/retrained_models/nat_inceptionv3_1.0/outputgraph.pb to classify ../visualization_output/nat/image/english_cocker_spaniel_24_
nat.jpg
2018-04-14 18:52:26.617645: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not com
piled to use: AVX2 FMA
2018-04-14 18:52:27.650634: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithGlobalNormalization is deprecated. It will cease to work in
 GraphDef version 9. Use tf.nn.batch_normalization().
image 0.99399924
pdf 0.0056985063
music 0.00021634772
winexe 8.0176105e-05
linux binaries 5.7170864e-06

real    0m8.332s
user    0m7.748s
sys     0m1.332s
science@rearp15-ml:~/retrained_models$
```

Figure 37: Natural - Image (JPEG) classification.

```
science@rearp15-ml:~/retrained_models$ time ./classify.sh entropy 1.0 ../visualization_output/ent/music/06-Typical_Male_ent.jpg
Using model: /home/science/retrained_models/ent_inceptionv3_1.0/outputgraph.pb to classify ../visualization_output/ent/music/06-Typical_Male_ent.jpg
2018-04-14 18:58:08.882934: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not com
piled to use: AVX2 FMA
2018-04-14 18:58:09.890776: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithGlobalNormalization is deprecated. It will cease to work in
 GraphDef version 9. Use tf.nn.batch_normalization().
music 0.99789524
winexe 0.0015914065
image 0.00023307321
pdf 0.00020259853
linux binaries 7.761675e-05

real    0m8.067s
user    0m7.568s
sys     0m1.268s
science@rearp15-ml:~/retrained_models$
```

Figure 38: Entropy - Audio (MP3) classification.

```
science@rearp15-ml:~/retrained_models$ time ./classify.sh hilbert 1.0 ../visualization_output/hil/music/06-Typical_Male_hil.jpg
Using model: /home/science/retrained_models/hil_inceptionv3_1.0/outputgraph.pb to classify ../visualization_output/hil/music/06-Typical_Male_hil.jpg
2018-04-14 18:59:03.744920: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not com
piled to use: AVX2 FMA
2018-04-14 18:59:04.667607: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithGlobalNormalization is deprecated. It will cease to work in
 GraphDef version 9. Use tf.nn.batch_normalization().
music 0.99963546
winexe 0.0002041007
image 0.00011773484
linux binaries 3.9502065e-05
pdf 3.2172045e-06

real    0m7.213s
user    0m6.812s
sys     0m1.220s
science@rearp15-ml:~/retrained_models$
```

Figure 39: Hilbert - Audio (MP3) classification.

```
science@rearp15-ml:~/retrained_models$ time ./classify.sh natural 1.0 ../visualization_output/nat/music/06-Typical_Male_nat.jpg
Using model: /home/science/retrained_models/nat_inceptionv3_1.0/outputgraph.pb to classify ../visualization_output/nat/music/06-Typical_Male_nat.jpg
2018-04-14 18:59:51.572006: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not com
piled to use: AVX2 FMA
2018-04-14 18:59:52.321331: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithGlobalNormalization is deprecated. It will cease to work in
 GraphDef version 9. Use tf.nn.batch_normalization().
image 0.5230938
music 0.47489402
winexe 0.0010219798
linux binaries 0.0008985193
pdf 9.161838e-05

real    0m7.283s
user    0m6.928s
sys     0m1.144s
science@rearp15-ml:~/retrained_models$
```

Figure 40: Natural - Audio (MP3) classification.

```
science@rearp15-ml:~/retrained_models$ time ./classify.sh entropy 1.0 ../visualization_output/ent/pdf/A_Durable_Main-Memory_Index_Using_Flash_ent.jpg
Using model: /home/science/retrained_models/ent_inceptionv3_1.0/outputgraph.pb to classify ../visualization_output/ent/pdf/A_Durable_Main-Memory_Index_
Using_Flash_ent.jpg
2018-04-14 19:11:23.045385: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not com
piled to use: AVX2 FMA
2018-04-14 19:11:24.026313: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithGlobalNormalization is deprecated. It will cease to work in
 GraphDef version 9. Use tf.nn.batch_normalization().
pdf 0.999746
winexe 0.00024536386
linux binaries 6.928746e-06
image 1.656811e-06
music 3.867665e-08

real    0m7.518s
user    0m7.080s
sys     0m1.144s
science@rearp15-ml:~/retrained_models$
```

Figure 41: Entropy - PDF classification.

```
science@rearp15-ml:~/retrained_models$ time ./classify.sh hilbert 1.0 ../visualization_output/hil/pdf/A_Durable_Main-Memory_Index_Using_Flash_hil.jpg
Using model: /home/science/retrained_models/hil_inceptionv3_1.0/outputgraph.pb to classify ../visualization_output/hil/pdf/A_Durable_Main-Memory_Index_
Using_Flash_hil.jpg
2018-04-14 19:12:10.552139: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not com
piled to use: AVX2 FMA
2018-04-14 19:12:11.686504: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithGlobalNormalization is deprecated. It will cease to work in
 GraphDef version 9. Use tf.nn.batch_normalization().
pdf 0.9973144
linux binaries 0.0016721666
winexe 0.00083199376
music 9.1313916e-05
image 9.0124246e-05

real    0m7.541s
user    0m7.068s
sys     0m1.244s
science@rearp15-ml:~/retrained_models$
```

Figure 42: Hilbert - PDF classification.

```
science@rearp15-ml:~/retrained_models$ time ./classify.sh natural 1.0 ../visualization_output/nat/pdf/A_Durable_Main-Memory_Index_Using_Flash_nat.jpg
Using model: /home/science/retrained_models/nat_inceptionv3_1.0/outputgraph.pb to classify ../visualization_output/nat/pdf/A_Durable_Main-Memory_Index_
Using_Flash_nat.jpg
2018-04-14 19:13:25.860327: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not com
piled to use: AVX2 FMA
2018-04-14 19:13:26.886906: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithGlobalNormalization is deprecated. It will cease to work in
 GraphDef version 9. Use tf.nn.batch_normalization().
pdf 0.9555449
image 0.017881243
music 0.014400721
winexe 0.011378454
linux binaries 0.00079476734

real    0m7.074s
user    0m6.816s
sys     0m1.100s
science@rearp15-ml:~/retrained_models$
```

Figure 43: Natural - PDF classification.

```
science@rearp15-ml:~/retrained_models$ time ./classify.sh entropy 1.0 ../visualization_output/ent/winexe/netiougc_ent.jpg
Using model: /home/science/retrained_models/ent_inceptionv3_1.0/outputgraph.pb to classify ../visualization_output/ent/winexe/netiougc_ent.jpg
2018-04-14 19:20:46.259147: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not com
piled to use: AVX2 FMA
2018-04-14 19:20:47.133587: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithGlobalNormalization is deprecated. It will cease to work in
 GraphDef version 9. Use tf.nn.batch_normalization().
winexe 0.97942424
linux binaries 0.020486815
pdf 4.9304555e-05
music 2.818625e-05
image 1.1576166e-05

real    0m7.207s
user    0m7.012s
sys     0m0.980s
science@rearp15-ml:~/retrained_models$
```

Figure 44: Entropy - WinExe classification.

```
science@rearp15-ml:~/retrained_models$ time ./classify.sh hilbert 1.0 ../visualization_output/hil/winexe/netiougc_hil.jpg
Using model: /home/science/retrained_models/hil_inceptionv3_1.0/outputgraph.pb to classify ../visualization_output/hil/winexe/netiougc_hil.jpg
2018-04-14 19:21:20.431905: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not com
piled to use: AVX2 FMA
2018-04-14 19:21:21.284212: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithGlobalNormalization is deprecated. It will cease to work in
 GraphDef version 9. Use tf.nn.batch_normalization().
winexe 0.98125505
linux binaries 0.018590419
music 7.670124e-05
pdf 4.7338657e-05
image 3.0464029e-05

real    0m7.012s
user    0m6.568s
sys     0m1.180s
science@rearp15-ml:~/retrained_models$
```

Figure 45: Hilbert - WinExe classification.

```
science@rearp15-ml:~/retrained_models$ time ./classify.sh natural 1.0 ../visualization_output/nat/winexe/netiougc_nat.jpg
Using model: /home/science/retrained_models/nat_inceptionv3_1.0/outputgraph.pb to classify ../visualization_output/nat/winexe/netiougc_nat.jpg
2018-04-14 19:22:03.712482: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not com
piled to use: AVX2 FMA
2018-04-14 19:22:04.616690: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithGlobalNormalization is deprecated. It will cease to work in
 GraphDef version 9. Use tf.nn.batch_normalization().
winexe 0.99570626
linux binaries 0.0031370167
image 0.00076082617
pdf 0.00031560686
music 8.019538e-05

real    0m7.271s
user    0m6.772s
sys     0m1.304s
science@rearp15-ml:~/retrained_models$
```

Figure 46: Natural - WinExe classification.

```
science@rearp15-ml:~/retrained_models$ time ./classify.sh entropy 1.0 ../visualization_output/ent/linux_binaries/ppmtogif_ent.jpg
Using model: /home/science/retrained_models/ent_inceptionv3_1.0/outputgraph.pb to classify ../visualization_output/ent/linux_binaries/ppmtogif_ent.jpg
2018-04-14 19:16:16.293321: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not com
piled to use: AVX2 FMA
2018-04-14 19:16:17.061440: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithGlobalNormalization is deprecated. It will cease to work in
 GraphDef version 9. Use tf.nn.batch_normalization().
linux binaries 0.99531627
winexe 0.004531464
pdf 9.250544e-05
music 5.158882e-05
image 8.1373e-06

real    0m6.751s
user    0m6.436s
sys     0m1.216s
science@rearp15-ml:~/retrained_models$
```

Figure 47: Entropy - Linux Binary Classification.

```
science@rearp15-ml:~/retrained_models$ time ./classify.sh hilbert 1.0 ../visualization_output/hil/linux_binaries/ppmtogif_hil.jpg
Using model: /home/science/retrained_models/hil_inceptionv3_1.0/outputgraph.pb to classify ../visualization_output/hil/linux_binaries/ppmtogif_hil.jpg
2018-04-14 19:17:13.006921: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not com
piled to use: AVX2 FMA
2018-04-14 19:17:14.087731: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithGlobalNormalization is deprecated. It will cease to work in
 GraphDef version 9. Use tf.nn.batch_normalization().
linux binaries 0.947448
winexe 0.045455027
pdf 0.0050079213
image 0.0017716888
music 0.00031728615

real    0m7.569s
user    0m7.312s
sys     0m1.316s
science@rearp15-ml:~/retrained_models$
```

Figure 48: Hilbert - Linux Binary Classification.

```
science@rearp15-ml:~/retrained_models$ time ./classify.sh natural 1.0 ../visualization_output/nat/linux_binaries/ppmtogif_nat.jpg
Using model: /home/science/retrained_models/nat_inceptionv3_1.0/outputgraph.pb to classify ../visualization_output/nat/linux_binaries/ppmtogif_nat.jpg
2018-04-14 19:17:51.499597: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not com
piled to use: AVX2 FMA
2018-04-14 19:17:52.348002: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithGlobalNormalization is deprecated. It will cease to work in
 GraphDef version 9. Use tf.nn.batch_normalization().
linux binaries 0.9986852
winexe 0.0012523863
image 3.414057e-05
music 2.2863474e-05
pdf 5.35724e-06

real    0m7.638s
user    0m7.288s
sys     0m1.304s
science@rearp15-ml:~/retrained_models$
```

Figure 49: Natural - Linux Binary Classification.

5.3.2 Model Version 1.1

```
low:2018-04-17 04:40:09.228027: Step 7999: Cross entropy = 0.215410
INFO:tensorflow:2018-04-17 04:40:09.441815: Step 7999: Validation accuracy = 94.0% (N=100)
INFO:tensorflow:Final test accuracy = 90.7% (N=397)
INFO:tensorflow:Froze 2 variables.
Converted 2 variables to const ops.

real    62m7.279s
user    73m31.456s
sys     5m49.404s
root@rearp15-ml:/home/science/retrained_models# time ./retrain.sh entropy 1.1
```

Figure 50: Entropy Retraining.

```
INFO:tensorflow:2018-04-17 02:35:57.513622: Step 7999: Cross entropy = 0.135027
INFO:tensorflow:2018-04-17 02:35:57.740302: Step 7999: Validation accuracy = 93.0% (N=100)
INFO:tensorflow:Final test accuracy = 90.2% (N=388)
INFO:tensorflow:Froze 2 variables.
Converted 2 variables to const ops.

real    66m18.273s
user    78m30.800s
sys     6m22.568s
root@rearp15-ml:/home/science/retrained_models# time ./retrain.sh hilbert 1.1
```

Figure 51: Hilbert Retraining.

```
INFO:tensorflow:2018-04-17 01:26:30.964645: Step 7999: Cross entropy = 0.160894
INFO:tensorflow:2018-04-17 01:26:31.220024: Step 7999: Validation accuracy = 93.0% (N=100)
INFO:tensorflow:Final test accuracy = 91.0% (N=401)
INFO:tensorflow:Froze 2 variables.
Converted 2 variables to const ops.

real    64m9.886s
user    76m14.164s
sys     6m13.820s
root@rearp15-ml:/home/science/retrained_models# time ./retrain.sh natural 1.1
```

Figure 52: Natural Retraining.

5.3.3 Model Version 2.0

```
INFO:tensorflow:2018-04-08 21:02:25.619605: Step 3999: Validation accuracy = 96.0% (N=100)
INFO:tensorflow:Final test accuracy = 91.3% (N=654)
INFO:tensorflow:Froze 2 variables.
Converted 2 variables to const ops.

real    16m5.981s
user    15m36.580s
sys     0m34.932s
root@rearp15-ml:/home/science/retrained_models#
```

Figure 53: Entropy Retraining.

```
INFO:tensorflow:2018-04-08 22:34:48.595635: Step 3999: Cross entropy = 0.224123
INFO:tensorflow:2018-04-08 22:34:48.801228: Step 3999: Validation accuracy = 94.0% (N=100)
INFO:tensorflow:Final test accuracy = 90.7% (N=685)
INFO:tensorflow:Froze 2 variables.
Converted 2 variables to const ops.

real    14m34.425s
user    14m10.980s
sys     0m29.136s
root@rearp15-ml:/home/science/retrained_models# time ./retrain.sh hilbert 2.0
```

Figure 54: Hilbert Retraining.

```
INFO:tensorflow:2018-04-08 21:56:06.764983: Step 3999: Train accuracy = 87.0%
INFO:tensorflow:2018-04-08 21:56:06.765828: Step 3999: Cross entropy = 0.384666
INFO:tensorflow:2018-04-08 21:56:06.963291: Step 3999: Validation accuracy = 75.0% (N=100)
INFO:tensorflow:Final test accuracy = 81.9% (N=744)
INFO:tensorflow:Froze 2 variables.
Converted 2 variables to const ops.

real    15m7.523s
user    14m46.188s
sys     0m30.460s
root@rearp15-ml:/home/science/retrained_models# time ./retrain.sh natural 2.0
```

Figure 55: Natural Retraining.

```
science@rearp15-ml:~/retrained_models$ time ./classify.sh entropy 2.0 ../visualization_output/ent/exe_malware_ent/7b6cd93a-2702-11e8-900e-80e65024849a_
ent.jpg
Using model: /home/science/retrained_models/ent_inceptionv3_2.0/outputgraph.pb to classify ../visualization_output/ent/exe_malware_ent/7b6cd93a-2702-11
e8-900e-80e65024849a_ent.jpg
2018-04-14 19:25:52.347092: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not com
piled to use: AVX2 FMA
2018-04-14 19:25:53.138781: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithGlobalNormalization is deprecated. It will cease to work in
 GraphDef version 9. Use tf.nn.batch_normalization().
exe malware ent 0.87561685
winexe 0.050536625
music 0.048454713
image 0.017269038
pdf 0.0045767883

real    0m7.029s
user    0m6.728s
sys     0m1.112s
science@rearp15-ml:~/retrained_models$
```

Figure 56: Entropy - EXE Malware classification.

```
science@rearp15-ml:~/retrained_models$ time ./classify.sh hilbert 2.0 ../visualization_output/hil/exe_malware_hil/7b6cd93a-2702-11e8-900e-80e65024849a_
hil.jpg
Using model: /home/science/retrained_models/hil_inceptionv3_2.0/outputgraph.pb to classify ../visualization_output/hil/exe_malware_hil/7b6cd93a-2702-11
e8-900e-80e65024849a_hil.jpg
2018-04-14 19:27:50.032592: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not com
piled to use: AVX2 FMA
2018-04-14 19:27:51.029621: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithGlobalNormalization is deprecated. It will cease to work in
 GraphDef version 9. Use tf.nn.batch_normalization().
exe malware hil 0.8183908
winexe 0.15452072
linux binaries 0.020964907
music 0.00287711
image 0.0027301176

real    0m7.791s
user    0m7.248s
sys     0m1.336s
science@rearp15-ml:~/retrained_models$
```

Figure 57: Hilbert - EXE Malware classification.

```
science@rearp15-ml:~/retrained_models$ time ./classify.sh natural 2.0 ../visualization_output/nat/exe_malware_nat/7b6cd93a-2702-11e8-900e-80e65024849a_
nat.jpg
Using model: /home/science/retrained_models/nat_inceptionv3_2.0/outputgraph.pb to classify ../visualization_output/nat/exe_malware_nat/7b6cd93a-2702-11
e8-900e-80e65024849a_nat.jpg
2018-04-14 19:26:47.845366: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not com
piled to use: AVX2 FMA
2018-04-14 19:26:48.792037: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithGlobalNormalization is deprecated. It will cease to work in
 GraphDef version 9. Use tf.nn.batch_normalization().
winexe 0.54120326
exe malware nat 0.37643525
pdf 0.055077035
linux binaries 0.025507858
music 0.001177072

real    0m7.932s
user    0m7.284s
sys     0m1.548s
science@rearp15-ml:~/retrained_models$
```

Figure 58: Natural - EXE Malware classification.

```
science@rearp15-ml:~/retrained_models$ time ./classify.sh entropy 2.0 ../visualization_output/ent/winexe/amdocl_as32_ent.jpg
Using model: /home/science/retrained_models/ent_inceptionv3_2.0/outputgraph.pb to classify ../visualization_output/ent/winexe/amdocl_as32_ent.jpg
2018-04-14 19:30:26.987911: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not com
piled to use: AVX2 FMA
2018-04-14 19:30:28.308167: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithGlobalNormalization is deprecated. It will cease to work in
 GraphDef version 9. Use tf.nn.batch_normalization().
winexe 0.7587212
exe malware ent 0.1888306
linux binaries 0.04211186
pdf 0.009950723
music 0.0003639726

real    0m7.502s
user    0m7.192s
sys     0m1.156s
science@rearp15-ml:~/retrained_models$
```

Figure 59: Entropy - WinExe classification.

```
science@rearp15-ml:~/retrained_models$ time ./classify.sh hilbert 2.0 ../visualization_output/hil/winexe/amdocl_as32_hil.jpg
Using model: /home/science/retrained_models/hil_inceptionv3_2.0/outputgraph.pb to classify ../visualization_output/hil/winexe/amdocl_as32_hil.jpg
2018-04-14 19:31:16.107562: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not com
piled to use: AVX2 FMA
2018-04-14 19:31:17.020197: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithGlobalNormalization is deprecated. It will cease to work in
 GraphDef version 9. Use tf.nn.batch_normalization().
linux binaries 0.46191993
winexe 0.410864
exe malware hil 0.1264267
image 0.00049239135
music 0.00017159771

real    0m7.794s
user    0m7.164s
sys     0m1.388s
science@rearp15-ml:~/retrained_models$
```

Figure 60: Hilbert - WinExe classification.

```
science@rearp15-ml:~/retrained_models$ time ./classify.sh natural 2.0 ../visualization_output/nat/winexe/amdocl_as32_nat.jpg
Using model: /home/science/retrained_models/nat_inceptionv3_2.0/outputgraph.pb to classify ../visualization_output/nat/winexe/amdocl_as32_nat.jpg
2018-04-14 19:32:05.129439: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not com
piled to use: AVX2 FMA
2018-04-14 19:32:06.114815: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithGlobalNormalization is deprecated. It will cease to work in
 GraphDef version 9. Use tf.nn.batch_normalization().
winexe 0.7868112
linux binaries 0.13133217
exe malware nat 0.08039649
pdf 0.0007025188
music 0.00054232834

real    0m7.649s
user    0m6.944s
sys     0m1.116s
science@rearp15-ml:~/retrained_models$
```

Figure 61: Natural - WinExe classification.

```
science@rearp15-ml:~/retrained_models$ time ./classify.sh entropy 2.0 ../visualization_output/ent/image/Ragdoll_149_ent.jpg
Using model: /home/science/retrained_models/ent_inceptionv3_2.0/outputgraph.pb to classify ../visualization_output/ent/image/Ragdoll_149_ent.jpg
2018-04-14 19:38:01.100716: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not com
piled to use: AVX2 FMA
2018-04-14 19:38:02.268811: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithGlobalNormalization is deprecated. It will cease to work in
 GraphDef version 9. Use tf.nn.batch_normalization().
image 0.9969593
winexe 0.0009697761
exe malware ent 0.0009595209
music 0.00067790505
pdf 0.00041101692

real    0m8.499s
user    0m8.044s
sys     0m1.152s
science@rearp15-ml:~/retrained_models$
```

Figure 62: Entropy - Image (JPEG) classification.

```
science@rearp15-ml:~/retrained_models$ time ./classify.sh hilbert 2.0 ../visualization_output/hil/image/Ragdoll_149_hil.jpg
Using model: /home/science/retrained_models/hil_inceptionv3_2.0/outputgraph.pb to classify ../visualization_output/hil/image/Ragdoll_149_hil.jpg
2018-04-14 19:38:39.044089: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not com
piled to use: AVX2 FMA
2018-04-14 19:38:39.818629: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithGlobalNormalization is deprecated. It will cease to work in
 GraphDef version 9. Use tf.nn.batch_normalization().
image 0.98825383
pdf 0.008415705
exe malware hil 0.0019527028
linux binaries 0.0008018608
winexe 0.00030031794

real    0m7.054s
user    0m6.824s
sys     0m1.248s
science@rearp15-ml:~/retrained_models$
```

Figure 63: Hilbert - Image (JPEG) classification.

```
science@rearp15-ml:~/retrained_models$ time ./classify.sh natural 2.0 ../visualization_output/nat/image/Ragdoll_149_nat.jpg
Using model: /home/science/retrained_models/nat_inceptionv3_2.0/outputgraph.pb to classify ../visualization_output/nat/image/Ragdoll_149_nat.jpg
2018-04-14 19:39:08.555367: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not com
piled to use: AVX2 FMA
2018-04-14 19:39:09.427752: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithGlobalNormalization is deprecated. It will cease to work in
 GraphDef version 9. Use tf.nn.batch_normalization().
image 0.9973537
pdf 0.0017384362
exe malware nat 0.00051530293
music 0.0002827931
winexe 0.00010572458

real    0m6.599s
user    0m6.252s
sys     0m1.084s
science@rearp15-ml:~/retrained_models$
```

Figure 64: Natural - Image (JPEG) classification.

5.3.4 Model Version 2.1

```
INFO:tensorflow:2018-04-09 00:23:11.902902: Step 7999: Cross entropy = 0.142236
INFO:tensorflow:2018-04-09 00:23:12.065212: Step 7999: Validation accuracy = 93.0% (N=
INFO:tensorflow:Final test accuracy = 92.0% (N=654)
INFO:tensorflow:Froze 2 variables.
Converted 2 variables to const ops.

real    28m42.220s
user    28m1.556s
sys     0m52.912s
root@rearp15-ml:/home/science/retrained_models# time ./retrain.sh entropy 2.1
```

Figure 65: Entropy Retraining.

```
INFO:tensorflow:2018-04-09 02:34:05.701601: Step 7999: Cross entropy = 0.097478
INFO:tensorflow:2018-04-09 02:34:05.879221: Step 7999: Validation accuracy = 94.0% (N=100)
INFO:tensorflow:Final test accuracy = 91.4% (N=685)
INFO:tensorflow:Froze 2 variables.
Converted 2 variables to const ops.

real    27m59.110s
user    27m17.984s
sys     0m51.580s
root@rearp15-ml:/home/science/retrained_models# time ./retrain.sh hilbert 2.1
```

Figure 66: Hilbert Retraining.

```
INFO:tensorflow:2018-04-09 02:02:27.806490: Step 7999: Validation accuracy = 85.0% (N=100)
15.0.4420.1017 w:Final test accuracy = 81.5% (N=744)
w:Froze 2 variables.
Converted 2 variables to const ops.

real    28m2.118s
user    27m18.140s
sys     0m53.352s
root@rearp15-ml:/home/science/retrained_models#
```

Figure 67: Natural Retraining.

```
science@rearp15-ml:~/retrained_models$ time ./classify.sh entropy 2.1 ../visualization_output/ent/image/great_pyrenees_61_ent.jpg
Using model: /home/science/retrained_models/ent_inceptionv3_2.1/outputgraph.pb to classify ../visualization_output/ent/image/great_pyrenees_61_ent.jpg
2018-04-14 21:13:10.238020: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not com
piled to use: AVX2 FMA
2018-04-14 21:13:11.101528: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithGlobalNormalization is deprecated. It will cease to work in
 GraphDef version 9. Use tf.nn.batch_normalization().
image 0.9961941
exe malware ent 0.0029067849
music 0.0005765567
winexe 0.00028173768
pdf 3.448985e-05

real    0m7.294s
user    0m6.880s
sys     0m1.188s
science@rearp15-ml:~/retrained_models$
```

Figure 68: Entropy - Image (JPEG) classification.

```
science@rearp15-ml:~/retrained_models$ time ./classify.sh hilbert 2.1 ../visualization_output/hil/image/great_pyrenees_61_hil.jpg
Using model: /home/science/retrained_models/hil_inceptionv3_2.1/outputgraph.pb to classify ../visualization_output/hil/image/great_pyrenees_61_hil.jpg
2018-04-14 21:13:59.791115: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not com
piled to use: AVX2 FMA
2018-04-14 21:14:00.657369: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithGlobalNormalization is deprecated. It will cease to work in
 GraphDef version 9. Use tf.nn.batch_normalization().
image 0.9968124
exe malware hil 0.0013445178
pdf 0.0008647539
winexe 0.00084226363
music 6.99344e-05

real    0m8.401s
user    0m7.752s
sys     0m1.484s
science@rearp15-ml:~/retrained_models$
```

Figure 69: Hilbert - Image (JPEG) classification.

```
science@rearp15-ml:~/retrained_models$ time ./classify.sh natural 2.1 ../visualization_output/nat/image/great_pyrenees_61_nat.jpg
Using model: /home/science/retrained_models/nat_inceptionv3_2.1/outputgraph.pb to classify ../visualization_output/nat/image/great_pyrenees_61_nat.jpg
2018-04-14 21:14:52.545540: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not com
piled to use: AVX2 FMA
2018-04-14 21:14:53.337504: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithGlobalNormalization is deprecated. It will cease to work in
 GraphDef version 9. Use tf.nn.batch_normalization().
image 0.99108475
exe malware nat 0.005450285
music 0.0023260447
winexe 0.00077046535
pdf 0.000351018

real    0m7.027s
user    0m6.548s
sys     0m1.132s
science@rearp15-ml:~/retrained_models$
```

Figure 70: Natural - Image (JPEG) classification.

```
science@rearp15-ml:~/retrained_models$ time ./classify.sh entropy 2.1 ../visualization_output/ent/winexe/xsltc_ent.jpg
Using model: /home/science/retrained_models/ent_inceptionv3_2.1/outputgraph.pb to classify ../visualization_output/ent/winexe/xsltc_ent.jpg
2018-04-14 21:16:57.248055: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not com
piled to use: AVX2 FMA
2018-04-14 21:16:58.168195: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithGlobalNormalization is deprecated. It will cease to work in
 GraphDef version 9. Use tf.nn.batch_normalization().
winexe 0.6511801
exe malware ent 0.33604506
linux binaries 0.011845351
music 0.0006276648
pdf 0.00019365479

real    0m7.136s
user    0m6.808s
sys     0m1.076s
science@rearp15-ml:~/retrained_models$
```

Figure 71: Entropy - WinExe classification.

```
science@rearp15-ml:~/retrained_models$ time ./classify.sh hilbert 2.1 ../visualization_output/hil/winexe/xsltc_hil.jpg
Using model: /home/science/retrained_models/hil_inceptionv3_2.1/outputgraph.pb to classify ../visualization_output/hil/winexe/xsltc_hil.jpg
2018-04-14 21:17:45.414003: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not com
piled to use: AVX2 FMA
2018-04-14 21:17:46.332177: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithGlobalNormalization is deprecated. It will cease to work in
 GraphDef version 9. Use tf.nn.batch_normalization().
winexe 0.8389932
exe malware hil 0.15206496
linux binaries 0.00567354
pdf 0.002676901
music 0.0003264797

real    0m7.359s
user    0m6.840s
sys     0m1.272s
science@rearp15-ml:~/retrained_models$
```

Figure 72: Hilbert - WinExe classification.

```
science@rearp15-ml:~/retrained_models$ time ./classify.sh natural 2.1 ../visualization_output/nat/winexe/xsltc_nat.jpg
Using model: /home/science/retrained_models/nat_inceptionv3_2.1/outputgraph.pb to classify ../visualization_output/nat/winexe/xsltc_nat.jpg
2018-04-14 21:18:34.457093: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not com
piled to use: AVX2 FMA
2018-04-14 21:18:35.236453: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithGlobalNormalization is deprecated. It will cease to work in
 GraphDef version 9. Use tf.nn.batch_normalization().
winexe 0.5503858
linux binaries 0.35823876
exe malware nat 0.083859324
pdf 0.005918793
image 0.00088036014

real    0m6.949s
user    0m6.660s
sys     0m1.180s
science@rearp15-ml:~/retrained_models$
```

Figure 73: Natural - WinExe classification.

```
science@rearp15-ml:~/retrained_models$ time ./classify.sh entropy 2.1 ../visualization_output/ent/exe_malware_ent/bd7dc738-3766-11e7-acd3-80e65024849a_
ent.jpg
Using model: /home/science/retrained_models/ent_inceptionv3_2.1/outputgraph.pb to classify ../visualization_output/ent/exe_malware_ent/bd7dc738-3766-11
e7-acd3-80e65024849a_ent.jpg
2018-04-14 21:21:11.854350: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not com
piled to use: AVX2 FMA
2018-04-14 21:21:12.845672: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithGlobalNormalization is deprecated. It will cease to work in
 GraphDef version 9. Use tf.nn.batch_normalization().
exe malware ent 0.9212645
winexe 0.054256156
pdf 0.024229653
linux binaries 0.0001480531
image 7.0615315e-05

real    0m6.833s
user    0m6.608s
sys     0m1.036s
science@rearp15-ml:~/retrained_models$
```

Figure 74: Entropy - Malware classification.

```
science@rearp15-ml:~/retrained_models$ time ./classify.sh hilbert 2.1 ../visualization_output/hil/exe_malware_hil/bd7dc738-3766-11e7-acd3-80e65024849a_
hil.jpg
Using model: /home/science/retrained_models/hil_inceptionv3_2.1/outputgraph.pb to classify ../visualization_output/hil/exe_malware_hil/bd7dc738-3766-11
e7-acd3-80e65024849a_hil.jpg
2018-04-14 21:22:07.406734: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not com
piled to use: AVX2 FMA
2018-04-14 21:22:08.424574: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithGlobalNormalization is deprecated. It will cease to work in
 GraphDef version 9. Use tf.nn.batch_normalization().
exe malware hil 0.99088895
winexe 0.008951713
linux binaries 4.6986064e-05
pdf 4.272336e-05
music 3.9847244e-05

real    0m7.578s
user    0m7.264s
sys     0m1.100s
science@rearp15-ml:~/retrained_models$
```

Figure 75: Hilbert - Malware classification.

```
science@rearp15-ml:~/retrained_models$ time ./classify.sh natural 2.1 ../visualization_output/nat/exe_malware_nat/bd7dc738-3766-11e7-acd3-80e65024849a_
nat.jpg
Using model: /home/science/retrained_models/nat_inceptionv3_2.1/outputgraph.pb to classify ../visualization_output/nat/exe_malware_nat/bd7dc738-3766-11
e7-acd3-80e65024849a_nat.jpg
2018-04-14 21:22:54.171958: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not com
piled to use: AVX2 FMA
2018-04-14 21:22:54.910133: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithGlobalNormalization is deprecated. It will cease to work in
 GraphDef version 9. Use tf.nn.batch_normalization().
exe malware nat 0.8995686
winexe 0.06889172
linux binaries 0.031201907
image 0.0001493086
pdf 9.5354044e-05

real    0m7.195s
user    0m6.936s
sys     0m1.216s
science@rearp15-ml:~/retrained_models$
```

Figure 76: Natural - Malware classification.

5.3.5 Model Version 3

```
INFO:tensorflow:2018-04-09 17:49:13.078525: Step 3999: Cross entropy = 0.243178
INFO:tensorflow:2018-04-09 17:49:13.303227: Step 3999: Validation accuracy = 89.0% (N=100)
INFO:tensorflow:Final test accuracy = 89.4% (N=397)
INFO:tensorflow:Froze 2 variables.
Converted 2 variables to const ops.

real    19m11.821s
user    18m28.940s
sys     0m52.336s
root@rearp15-ml:/home/science/retrained_models#
```

Figure 77: Entropy Retraining.

```
INFO:tensorflow:2018-04-16 18:54:52.014136: Step 3990: Validation accuracy = 94.0% (N=100)
INFO:tensorflow:2018-04-16 18:54:54.893704: Step 3999: Train accuracy = 99.0%
INFO:tensorflow:2018-04-16 18:54:54.897331: Step 3999: Cross entropy = 0.112097
INFO:tensorflow:2018-04-16 18:54:55.178487: Step 3999: Validation accuracy = 95.0% (N=100)
INFO:tensorflow:Final test accuracy = 89.7% (N=388)
INFO:tensorflow:Froze 2 variables.
Converted 2 variables to const ops.
root@rearp15-ml:/home/science/retrained_models# ./retrain.sh hilbert 3.0
```

Figure 78: Hilbert Retraining.

```
INFO:tensorflow:2018-04-09 16:33:48.801709: Step 3999: Cross entropy = 0.225410
INFO:tensorflow:2018-04-09 16:33:49.076129: Step 3999: Validation accuracy = 85.0% (N=100)
INFO:tensorflow:Final test accuracy = 88.5% (N=401)
INFO:tensorflow:Froze 2 variables.
Converted 2 variables to const ops.

real    18m36.599s
user    18m1.296s
sys     0m44.168s
root@rearp15-ml:/home/science/retrained_models#
```

Figure 79: Natural Retraining.

```
science@rearp15-ml:~/retrained_models$ time ./classify.sh entropy 3.0 ../visualization_output/ent_mal/winexe/ranlib_ent.jpg
Using model: /home/science/retrained_models/ent_inceptionv3_3.0/outputgraph.pb to classify ../visualization_output/ent_mal/winexe/ranlib_ent.jpg
2018-04-14 20:21:57.621080: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not com
piled to use: AVX2 FMA
2018-04-14 20:21:58.582488: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithGlobalNormalization is deprecated. It will cease to work in
 GraphDef version 9. Use tf.nn.batch_normalization().
winexe 0.5726964
exe malware ent 0.40142858
linux binaries 0.025874984

real    0m7.840s
user    0m7.396s
sys     0m1.188s
science@rearp15-ml:~/retrained_models$
```

Figure 80: Entropy - WinEXE classification.

```
science@rearp15-ml:~/retrained_models$ time ./classify.sh hilbert 3.0 ../visualization_output/hil_mal/winexe/ranlib_hil.jpg
Using model: /home/science/retrained_models/hil_inceptionv3_3.0/outputgraph.pb to classify ../visualization_output/hil_mal/winexe/ranlib_hil.jpg
2018-04-14 20:22:45.769052: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not com
piled to use: AVX2 FMA
2018-04-14 20:22:46.642807: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithGlobalNormalization is deprecated. It will cease to work in
 GraphDef version 9. Use tf.nn.batch_normalization().
winexe 0.68711805
exe malware hil 0.296572
linux binaries 0.016309926

real    0m7.102s
user    0m6.888s
sys     0m1.112s
science@rearp15-ml:~/retrained_models$
```

Figure 81: Hilbert - WinEXE classification.

```
science@rearp15-ml:~/retrained_models$ time ./classify.sh natural 3.0 ../visualization_output/nat_mal/winexe/ranlib_nat.jpg
Using model: /home/science/retrained_models/nat_inceptionv3_3.0/outputgraph.pb to classify ../visualization_output/nat_mal/winexe/ranlib_nat.jpg
2018-04-14 20:23:29.655888: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not com
piled to use: AVX2 FMA
2018-04-14 20:23:30.445125: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithGlobalNormalization is deprecated. It will cease to work in
 GraphDef version 9. Use tf.nn.batch_normalization().
winexe 0.7572831
linux binaries 0.14588489
exe malware nat 0.09683198

real    0m7.958s
user    0m7.488s
sys     0m1.552s
science@rearp15-ml:~/retrained_models$
```

Figure 82: Natural - WinEXE classification.

```
science@rearp15-ml:~/retrained_models$ time ./classify.sh entropy 3.0 ../visualization_output/ent_mal/linux_binaries/udevadm_ent.jpg
Using model: /home/science/retrained_models/ent_inceptionv3_3.0/outputgraph.pb to classify ../visualization_output/ent_mal/linux_binaries/udevadm_ent.j
pg
2018-04-14 20:25:54.205035: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not com
piled to use: AVX2 FMA
2018-04-14 20:25:55.236595: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithGlobalNormalization is deprecated. It will cease to work in
 GraphDef version 9. Use tf.nn.batch_normalization().
linux binaries 0.9853001
winexe 0.012184976
exe malware ent 0.0025148368

real    0m7.299s
user    0m6.988s
sys     0m1.216s
science@rearp15-ml:~/retrained_models$
```

Figure 83: Entropy - Linux Binary classification.

```
science@rearp15-ml:~/retrained_models$ time ./classify.sh hilbert 3.0 ../visualization_output/hil_mal/linux_binaries/udevadm_hil.jpg
Using model: /home/science/retrained_models/hil_inceptionv3_3.0/outputgraph.pb to classify ../visualization_output/hil_mal/linux_binaries/udevadm_hil.j
pg
2018-04-14 20:26:30.087100: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not com
piled to use: AVX2 FMA
2018-04-14 20:26:30.958837: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithGlobalNormalization is deprecated. It will cease to work in
 GraphDef version 9. Use tf.nn.batch_normalization().
linux binaries 0.99793065
winexe 0.001203955
exe malware hil 0.0008653406

real    0m7.859s
user    0m7.364s
sys     0m1.376s
science@rearp15-ml:~/retrained_models$
```

Figure 84: Hilbert - Linux Binary classification.

```
science@rearp15-ml:~/retrained_models$ time ./classify.sh natural 3.0 ../visualization_output/nat_mal/linux_binaries/udevadm_nat.jpg
Using model: /home/science/retrained_models/nat_inceptionv3_3.0/outputgraph.pb to classify ../visualization_output/nat_mal/linux_binaries/udevadm_nat.j
pg
2018-04-14 20:27:14.914719: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not com
piled to use: AVX2 FMA
2018-04-14 20:27:16.128895: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithGlobalNormalization is deprecated. It will cease to work in
 GraphDef version 9. Use tf.nn.batch_normalization().
linux binaries 0.92091495
winexe 0.06742841
exe malware nat 0.011656652

real    0m8.436s
user    0m7.972s
sys     0m1.332s
science@rearp15-ml:~/retrained_models$
```

Figure 85: Natural - Linux Binary classification.

```
science@rearp15-ml:~/retrained_models$ time ./classify.sh entropy 3.0 ../visualization_output/ent_mal/exe_malware_ent/af24c017-5713-11e7-87ca-80e650248
49a_ent.jpg
Using model: /home/science/retrained_models/ent_inceptionv3_3.0/outputgraph.pb to classify ../visualization_output/ent_mal/exe_malware_ent/af24c017-571
3-11e7-87ca-80e65024849a_ent.jpg
2018-04-14 20:42:37.917314: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not com
piled to use: AVX2 FMA
2018-04-14 20:42:38.745516: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithGlobalNormalization is deprecated. It will cease to work in
 GraphDef version 9. Use tf.nn.batch_normalization().
exe malware ent 0.5052196
winexe 0.49441323
linux binaries 0.00036716493

real    0m8.888s
user    0m8.496s
sys     0m1.280s
science@rearp15-ml:~/retrained_models$
```

Figure 86: Entropy - EXE Malware classification.

```
science@rearp15-ml:~/retrained_models$ time ./classify.sh hilbert 3.0 ../visualization_output/hil/exe_malware_hil/af24c017-5713-11e7-87ca-80e65024849a_
hil.jpg
Using model: /home/science/retrained_models/hil_inceptionv3_3.0/outputgraph.pb to classify ../visualization_output/hil/exe_malware_hil/af24c017-5713-11
e7-87ca-80e65024849a_hil.jpg
2018-04-14 20:43:50.779087: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not com
piled to use: AVX2 FMA
2018-04-14 20:43:51.751791: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithGlobalNormalization is deprecated. It will cease to work in
 GraphDef version 9. Use tf.nn.batch_normalization().
exe malware hil 0.85569334
winexe 0.11452489
linux binaries 0.029781802

real    0m8.575s
user    0m7.920s
sys     0m1.368s
science@rearp15-ml:~/retrained_models$
```

Figure 87: Hilbert - EXE Malware classification.

```
science@rearp15-ml:~/retrained_models$ time ./classify.sh natural 3.0 ../visualization_output/nat/exe_malware_nat/af24c017-5713-11e7-87ca-80e65024849a_
nat.jpg
Using model: /home/science/retrained_models/nat_inceptionv3_3.0/outputgraph.pb to classify ../visualization_output/nat/exe_malware_nat/af24c017-5713-11
e7-87ca-80e65024849a_nat.jpg
2018-04-14 20:44:39.187145: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not com
piled to use: AVX2 FMA
2018-04-14 20:44:39.989736: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithGlobalNormalization is deprecated. It will cease to work in
 GraphDef version 9. Use tf.nn.batch_normalization().
exe malware nat 0.9759613
winexe 0.023002004
linux binaries 0.0010367348

real    0m7.318s
user    0m7.012s
sys     0m1.072s
science@rearp15-ml:~/retrained_models$
```

Figure 88: Natural EXE Malware classification.

5.3.6 Model Version 3.1

```
INFO:tensorflow:2018-04-16 22:27:44.356029: Step 7999: Cross entropy = 0.158154
INFO:tensorflow:2018-04-16 22:27:44.557644: Step 7999: Validation accuracy = 91.0% (N=100)
INFO:tensorflow:Final test accuracy = 90.4% (N=397)
INFO:tensorflow:Froze 2 variables.
Converted 2 variables to const ops.

real    40m43.677s
user    39m33.644s
sys     1m34.068s
root@rearp15-ml:/home/science/retrained_models#
```

Figure 89: Entropy Retraining.

```
INFO:tensorflow:2018-04-16 23:10:16.523669: Step 7999: Train accuracy = 97.0%
INFO:tensorflow:2018-04-16 23:10:16.526755: Step 7999: Cross entropy = 0.121287
INFO:tensorflow:2018-04-16 23:10:16.937979: Step 7999: Validation accuracy = 92.0% (N=100)
INFO:tensorflow:Final test accuracy = 91.0% (N=388)
INFO:tensorflow:Froze 2 variables.
Converted 2 variables to const ops.

real    38m45.228s
user    37m39.384s
sys     1m30.100s
root@rearp15-ml:/home/science/retrained_models# time ./retrain.sh hilbert 3.1
```

Figure 90: Hilbert Retraining.

```
INFO:tensorflow:2018-04-17 00:21:41.607662: Step 7999: Cross entropy = 0.182570
INFO:tensorflow:2018-04-17 00:21:41.811990: Step 7999: Validation accuracy = 92.0% (N=100)
INFO:tensorflow:Final test accuracy = 90.8% (N=401)
INFO:tensorflow:Froze 2 variables.
Converted 2 variables to const ops.

real    37m3.726s
user    36m0.740s
sys     1m27.692s
root@rearp15-ml:/home/science/retrained_models# time ./retrain.sh natural 3.1
```

Figure 91: Natural Retraining.

5.3.5 Summary

For a complete comparison of image classification accuracy across all of our models and between the three visualization types refer to the table below. In summary, the absolute highest accuracy model was Entropy Version 1.0 with an accuracy of 97.3%. The highest Hilbert model was Version 1.0, the highest Natural visualization model was Version 1.1. The highest accuracy for Version 2.x variants was Entropy 2.1 and 3.0 with an accuracy of 92.0%. This version of the model is most significant to our research as it contains malicious windows executables, detecting malware was a primary goal of our research. Across the visualization types accuracy was highest in Entropy, Hilbert and lastly Natural with a lowest score of 81.4% in Version 1.0.

	V1.0	**V1.1**	**V2.0**	**V2.1**	**V3.0**	**V3.1**
Entropy	97.3%	90.7%	91.3%	92.0%	92.0%	90.4%
Hilbert	95.9%	90.2%	90.7%	91.4%	91.4%	91.0%
Natural	81.4%	91.0%	81.9%	81.5%	81.5%	90.8%

5.4 Discussion

As our experiments show, entropy and hilbert visualizations yielded the highest accuracy, while natural visualizations yielded the lowest accuracy, due to lack of clustering that the hilbert and entropy visualizations provide. However, we did notice an increase in accuracy by increasing the training steps but in a significant amount to make it more efficient than the alternatives we tested.

The experiments showed that we could classify images in network traffic with a high accuracy, but in order for this to be a viable solution there are few issues that need to be addressed. First, on a low tier server the classification took quite a bit of time which would not be good for large packet capture files with a lot of data transfer. One way to solve this would be to upgrade to a higher tier. Second, the time it takes to create the visualizations would need to be reduced. In our experiments we used all three visualization types, but since entropy and hilbert visualizations produced the highest accuracy only one of those visualizations would be required, furthermore implementing the visualizations in a lower level language would yield faster results. Finally, we noticed lower accuracy when regular files, executables, and malware were combined as opposed to executables and malware alone. Collecting more samples might yield better results or finding a more efficient way of identifying binaries from regular files could yield better results.

Overall, our research proved successful but there are improvements which can be made to make it a viable option. With more experiments and advancements in the field of machine learning this could become a scalable way to efficiently analyze large packet capture files.

6.0 Future Work

Given the objective of our research, which was to determine if binary visualization techniques can be combined with machine learning to detect file types and malware within packet capture files; with the potential for being automated and scalable our visualization techniques should be as time and resource efficient as possible. The current toolset for parsing packet captures to extract files and visualize them uses *Python*. Lower level languages such as *Go* or *Rust* could be leveraged to reduce visualization time, which would contribute to an overall more efficient and scalable solution. Future enhancements to our work include creating *Go* and *Rust* variants of the visualization tools to compare which language can optimize the process for all future work.

In addition to efficiency gains expanding our research, specifically the machine learning process upon more file/artifact types would greatly increase the usefulness of our work. As it currently stands accuracy among the current file types is high, exceeding 90% in the cases of Hilbert curve and Entropy visualizations. Introducing more file types such as compressed archives (ZIP, RAR, Tar balls, and similar), video/movie files, and HTML documents would greatly expand use cases. In addition to file types conducting research into machine learning on network traffic itself to distinguish different protocols and whether the usage is malicious or not would increase benefit further. Being able to detect plain-text protocols such as HTTP, DNS, FTP and encrypted protocols including SSH, HTTPS, and distinguishing the difference between legitimate usage and malicious usage such as Command-and-Control (C2C) uses and data

exfiltration would be of interest.

Our current training of InceptionV3 used roughly 6800 visualizations of each of the three selected visualization types. While these numbers are above recommended amount of machine learning, to contribute further to accuracy expanding the number and variation of samples used to train the model, and for classification theoretically would improve results.

Lastly, experimenting with different machine learning models would be of additional value. Of the top image classification models the InceptionV3 model is 2nd highest in speed, the highest is ResNet-50. Conducting training and comparing results between InceptionV3, VGG16, ResNet50 and any new models could provide more accurate and/or quicker classification.

References

Aparicio, M. and Costa, C. (2015). *Data Visualization*.

Beingessner, Alexis. Carleton University, 2017.

Conti, Gregory, Erik Dean, Matthew Sinda, and Benjamin Sangster. "Visual reverse engineering of binary and data files." *Visualization for Computer Security* (2008): 1-17.

Conti, Gregory, Sergey Bratus, Anna Shubina, Benjamin Sangster, Roy Ragsdale, Matthew Supan, Andrew Lichtenberg, and Robert Perez-Alemany. "Automated mapping of large binary objects using primitive fragment type classification." *digital investigation* 7 (2010): S3-S12.

Domas, Christopher. "The future of RE: Dynamic binary visualization." *DerbyCon. Louisville, KY.. Web* 28 (2012).

Godfrey, Parke, Jarek Gryz, and Piotr Lasek. "Interactive visualization of large data sets." *IEEE Transactions on Knowledge and Data Engineering* 28, no. 8 (2016): 2142-2157.

Hartstack, Ellen Jordan. "VMIFF–Visualization metrics for the identification of file fragments." PhD diss., Iowa State University, 2013.

Internet Security Threat Report. (2016). *Symantec*, 21.

State of Malware Report. (2017). *Malwarebytes Labs*.

Panas, Thomas. "Signature visualization of software binaries." In *Proceedings of the 4th ACM symposium on Software visualization*, pp. 185-188. ACM, 2008.

Sadana, Ramik, Timothy Major, Alistair Dove, and John Stasko. "Onset: A visualization technique for large-scale binary set data." *IEEE Transactions on Visualization and Computer*

Graphics 20, no. 12 (2014): 1993-2002.

Sikorski, Michael, and Andrew Honig. *Practical malware analysis: the hands-on guide to dissecting malicious software*. no starch press, 2012.

Wright, Charles V., Fabian Monrose, and Gerald M. Masson. "Using visual motifs to classify encrypted traffic." In *Proceedings of the 3rd international workshop on Visualization for computer security*, pp. 41-50. ACM, 2006.

Wagner, Markus, Fabian Fischer, Robert Luh, Andrea Haberson, Alexander Rind, Daniel A. Keim, Wolfgang Aigner, R. Borgo, F. Ganovelli, and I. Viola. "A survey of visualization systems for malware analysis." In *EG Conference on Visualization (EuroVis)-STARs*, pp. 105-125. 2015.