Malware Classification based on Image and N-gram OpCode Features

Peiyi Yang (py5yy), Zice Wei (zw9ga), Rongyu Wang (rw9fs)

Abstract

Malware classification is the first key step towards efficiently analyzing their impact. In this project, we extracted two types of features from hexadecimal malware files, malware images and N-gram OpCode features, and fed them (a single type of feature as well as combined features) into classification models such as SVM, Random Forest and Neural Networks and achieved high accuracy. We also discussed some of the technical challenges we encountered and our solutions. The code link and how we split the work are given in the appendix.

Introduction

Anti-malware software is constantly being updated to recognize common viruses. A single virus strain might only have a relatively short life span before it is recognized. So cyber criminals are also constantly modifying and obfuscating malware to make malware of the same class with similar behaviors look like different files, to trick anti-malware software and evade detection, increase the lifespan of malware and to get as far as possible with them. As a result, anti-malware software must deal with a large number of files and evaluate their potential malicious intent. In order to efficiently analyze malware, the first step is to identify their class or “family”.

Dataset

We adopted our dataset from the Microsoft Malware Classification Challenge (BIG 2015) on Kaggle [1]. It consists of malware files from 10 classes. Each malware sample is identified by a 20-character hash value, and an integer class label specifying which class the sample belongs. For each file, there is a hexadecimal file that represents its raw content (a .bytes file), and another assembly file that is generated from the hex file (a .asm file) by the IDA disassembler, as shown in Figure 1. The original dataset is over 35GB in size and overwhelming for the purpose of this course, so we sampled 10% from each class to make training and testing faster. Sample sizes vary from 300KB to 10MB. Table 1 shows the details of our dataset.

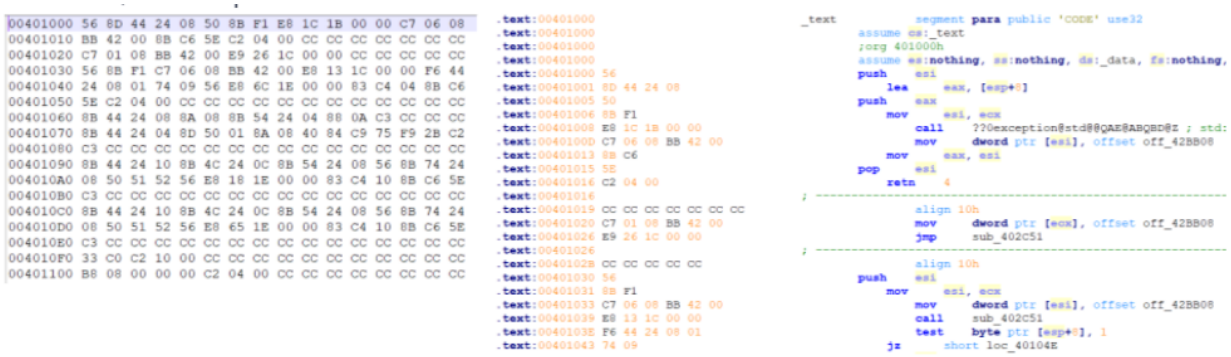

Figure 1. A malware sample. Left: hexadecimal file of raw content. Right: text file of assembly code generated by IDA.

Table 1. Statistics of our sampled dataset.

Malware Class	Number of Samples	Label	Type
Ramnit	154	1	Worm
Lollipop	247	2	Adware
Kelihos_ver3	294	3	Backdoor
Vundo	47	4	Trojan
Simda	4	5	Backdoor
Tracur	75	6	Trojan Downloader
Kelihos_ver1	39	7	Backdoor
Obfuscator.ACY	122	8	Obfuscated malware
Gatak	101	9	Backdoor

Methods

In this section we discuss how we engineered the features for classification and what classification models are used.

Feature Extraction

We designed two types of features: image features and N-gram OpCode features.
Imaged Features. The rationale is to visualize malware as gray-scale images and to classify them based on those images [2]. We believe malware of the same class shares the same pattern, layout and texture that can distinguish them from one another. Just like snakes, the pattern of their skin embeds enough features for experts to tell their species. To represent

malware as images, the simplest way is to treat every 8 bits or 1 byte from the hex file as a pixel, as they have the same range of [0, 255] (00~FF for hex representations). We demonstrate two sets of malware image examples in Figure 2 and Figure 3, in which we set the width to a fixed value of 512 pixels and got samples with varying height.

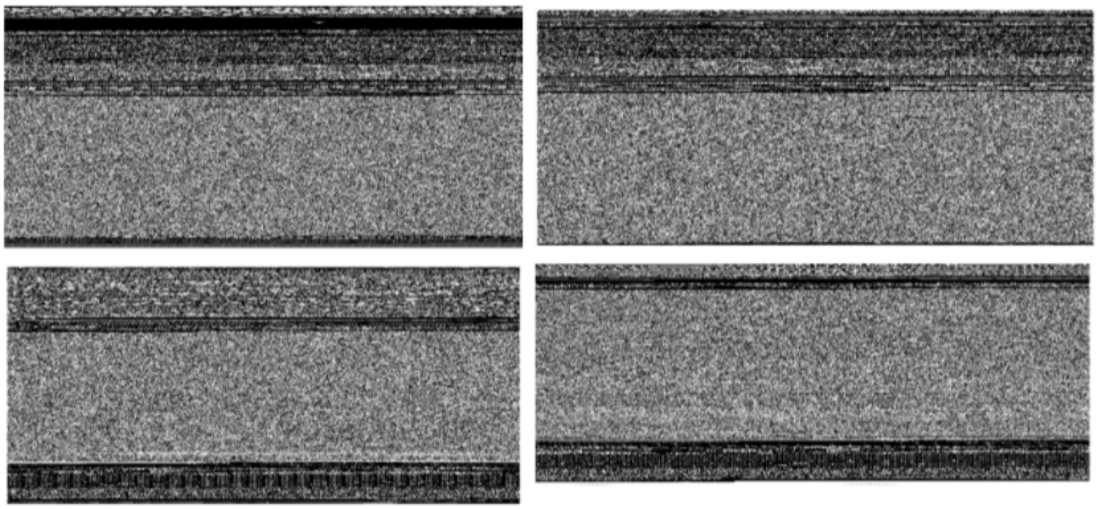

Figure 2. Four malware from class 1 with very similar pattern and texture.

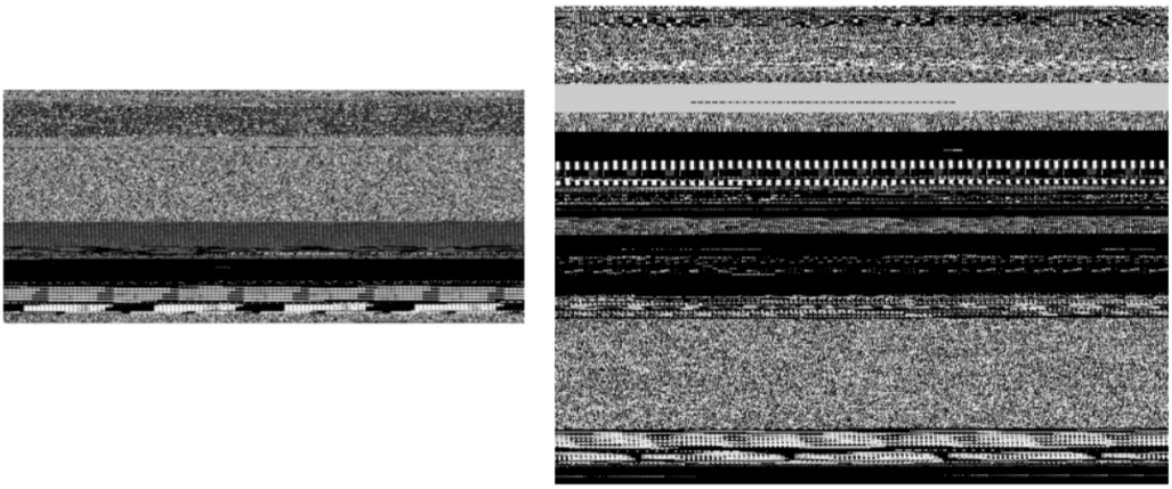

Figure 3. Two malware from class 8. Note the black and white stripes at the bottom of these images are not seen in previous examples and might be a distinguishing feature.

N-gram OpCode. Programs are essentially sequences of assembly instructions or Op(eration)Codes. Since malware of the same class often behave similarly, they probably have similar sets of operations and OpCodes. Different classes of malware, on the other hand, might have very different sets of OpCodes. That gives us a new direction: we can measure the frequency of different sets of OpCodes and use them as a feature. To be specific, we group every N OpCodes together and count their frequency in each malware. Basically it's the same idea of N-gram Bag of Words representation in Natural Language Processing. In our case, a bag-of-OpCodes is a representation of malware that describes the occurrence of

OpCode executed by the malware. We set N to be 3, filtered out some bags-of-OpCodes that are rare and got a 8809-dimensional vector. Figure 4 gives an overview of the N-gram OpCode feature we obtained.

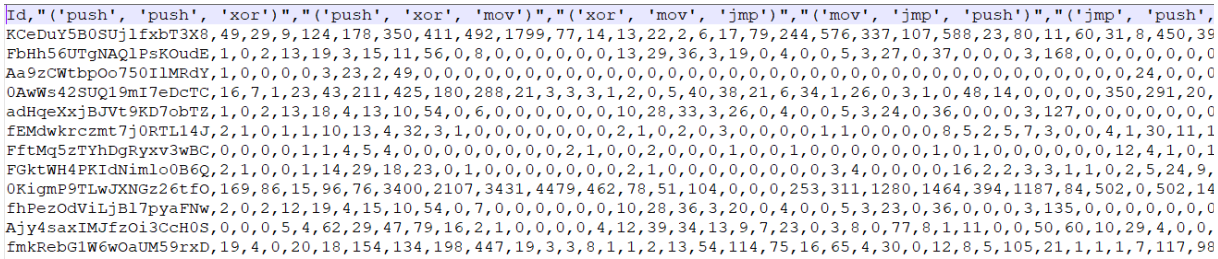

Figure 4. 3-gram OpCode feature we obtained.

Classification Algorithms

In this section, we introduced four classification models used in this study. Both conventional machine learning approaches and neural networks were adopted to distinguish malwares according to features we generated.

Random Forest

Random Forest (RF) is a type of ensemble learning, which can be seen as an extension of decision tree algorithms with bootstrap aggregation. Each decision tree is generated from a part of the original dataset. For each node of the tree, a part of features is chosen to provide the best segmentation. Compared to the K-Nearest-Neighbors(KNN) algorithm, RF is more efficient and has a great control of over-fitting. Since features generated in this project especially n-gram features has considerably high dimensions(i.e. Image features: >200, n-gram features: >8000), RF contributes to the feature selection by assigning different relative importance to these features.
Both in the experiments of image features and n-gram features, we used 650 estimators to achieve the best performance.

Support Vector Machine

Support Vector Machine(SVM) is also popular for its ability to deal with high-dimensional data, especially when the dimension of features is much higher than the sample size. It is a supervised learning algorithm that takes advantage of the kernel function to project non-linear samples to a higher dimension and use hypothesis gaps to divide data. SVM approach is well-known for its high efficiency on memory. We used linear core as the kernel function for SVM.

Neural Network

Apart from the traditional machine learning methods, we also implement two different neural networks related methods to perform the malware classification. Both networks are trained for 50 epochs.

We use VGG-16 [3], a pre-built Convolutional Neural networks (CNN) taking images converted from the execution file as input, which contains 16 layers and the overview is shown in the Figure 5 below.

Figure 5. Overview of VGG-16

We also use a Deep Neural Network (DNN) taking N-gram features as input, we use 6 dense layers and 2 dropout layers. The overview of the network is shown in the Figure 6 below.

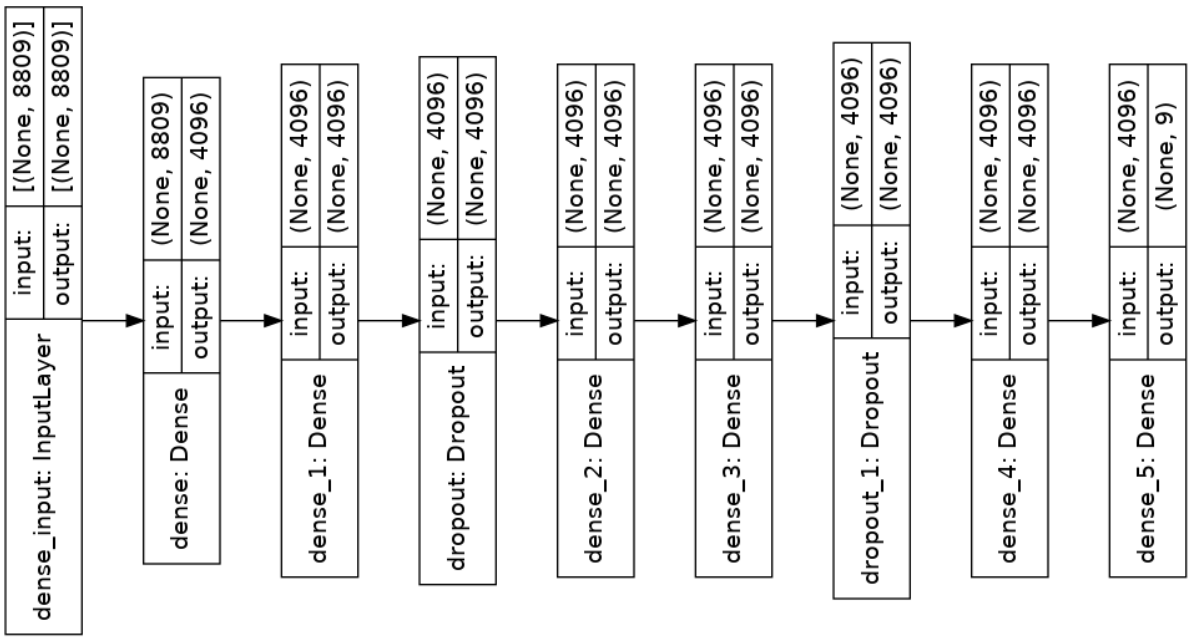

Figure 6. Overview of DNN

Evaluation

Traditional Machine Learning Algorithms

As for RF, when using image features to distinguish malwares' types, we obtained 83.75% accuracy. When using 3-gram features, we obtained a much higher accuracy of 97.12%. We also explored combined features and achieved 96.11% classification accuracy. Figure 7. shows a part of one decision tree generated in the experiment of image features. As indicated by Table 2. & 3, features' contributions varied for both n-gram features and image features. It is noticed that the relative significance of different image features are almost averaged, while for 3-gram features, feature No.1105 is more vital.

Table 2. Top 5 Important 3-gram Features

Top 5 Important 3-gram Features		
Rankings	Feature	Relative Importance
1.	1105	0.008577
2.	6432	0.007061
3.	4729	0.006862
4.	4034	0.006778
5.	4000	0.006452

Table 3. Top 5 Important Image Features

Top 5 Important Image Features		
Rankings	Feature	Relative Importance
1.	1907	0.002983
2.	1818	0.002717
3.	0	0.002691
4.	182	0.002562
5.	183	0.002295

In the experiments with SVM, for image features, our model achieved 76.43% accuracy. And same as performance of RF, we got higher accuracy on 3-gram features of 94.82%. The integrated features also showed significant accuracy of 86.27%.

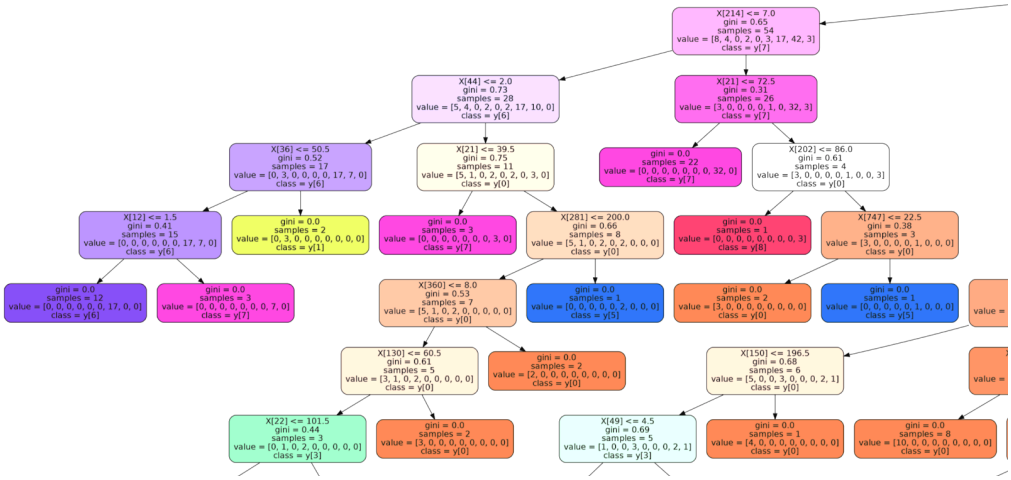

Figure 7. Partial decision tree of the image features classifier

Neural Networks

Figure 8 shows the training accuracy and the validation accuracy for both networks within 50 epochs. For CNN, the final accuracy rate is around 98%. The training accuracy increases rapidly from 0th epochs to 10th epoch, then it gradually increases up to 30th epoch. At around the 32th epoch, there is a drop possibly because we do not have enough data, then it gradually increases to 98%. For DNN, the final accuracy rate is around 90%. The training accuracy decreases at the beginning of the first epoch, then gradually increases up to the 20th epoch. The accuracy then slightly increases to 90%.

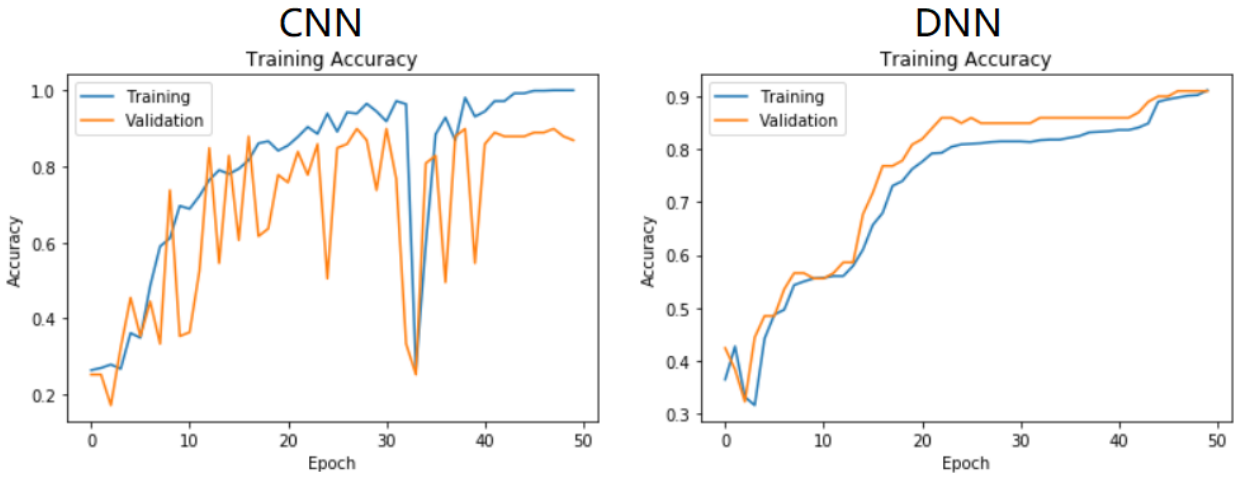

Figure 8. Training plots.

The following Table 4 shows the training accuracy and testing accuracy for each network, it can be known that training accuracy for CNN (VGG-16) with binary images as input is 7.9% higher than the accuracy for DNN with 3-gram features as input. The difference between training accuracy and testing accuracy may be caused by the small amount of data; there is only a little data for the Simda malware class.

Table 4. Training and testing accuracy.

Network	Training Accuracy	Testing Accuracy
CNN (VGG-16)	**98.9%**	**91.0%**
DNN	91%	86.4%

Compared with the leaderboard [4] of Microsoft Malware Classification Challenge, our models achieved performance comparable to the top 30% candidates, especially the random forest method and VGG-16 method. It is noticed that some teams used similar approaches and features as ours but obtained higher performance, which may be owed to the slight difference of feature selection and long tuning process, especially for traditional machine learning methods including RF and SVM, they rely greatly on features. Our future work may include further comparison and exploration of that direction.

Challenges

DNN had poor performance on image features. Initially we used the DNN on both image features and N-gram OpCode features. It worked fine with the N-gram features, but the performance on image features were terrible (~50% accuracy). We thought the images did not embed useful features like it should have in theory and were about to abandon them. It turned out that the DNN we used, which is a MLP with some dropout layers, was not powerful enough to extract those features, and after switching to CNN/VGG it all worked out.

Varying and uncommon input sizes of images for popular neural networks during training. Since the malware sizes vary from 300KB to 10MB, the sizes of malware images also vary significantly. Some were similar to common image sizes like 512x512, others were very, very big (long) and very different from what popular neural networks take as inputs (E.g. VGG is pre-trained with 224x224 images). Also, normally during training it is required that the samples in the same batch should have the same size. But in our case nearly every sample has a unique size and often too large for . Our workaround was quite simple: we resized/shrinked all samples to 224x224 before feeding into VGG. And it seemed to work without losing too much information. But we do wonder if there is a better solution.

Conclusion

In conclusion, we perform the malware classification based on image and N-gram opcode features. The image features generated from binary files are helpful for distinguishing codes of different malware. Besides, N-gram features generated from OpCode catch more essential information of the malware classification. For classification methods, both Random Forest and Support Vector Machine are powerful enough when dealing with our generated features. Moreover, the CNN (VGG-16) with Image input has higher accuracy than DNN with N-gram features.

For future work, we plan to build a Hybrid Neural Network, which combines VGG-16 and DNN to increase the accuracy. Also, we would like to design a toolbox integrating each method and automate the classification process. Besides, in real-world cases, many malware is encrypted. We would like to find an efficient way to deal with the encrypted virus.

Reference

[1] Ronen, R., Radu, M., Feuerstein, C., Yom-Tov, E., & Ahmadi, M. (2018). Microsoft malware classification challenge. *arXiv preprint arXiv:1802.10135*.
[2] Nataraj, L., Karthikeyan, S., Jacob, G., & Manjunath, B. S. (2011, July). Malware images: visualization and automatic classification. In *Proceedings of the 8th international symposium on visualization for cyber security* (pp. 1-7).

[3] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
[4] Microsoft Malware Classification Challenge (BIG 2015). (n.d.). Retrieved from https://www.kaggle.com/c/malware-classification/leaderboard

Appendix

Work Split

- Peiyi Yang: Sampled original dataset. Generated malware images and n-gram features from hexadecimal files and assembly files.
- Zice Wei: Designed the CNN and DNN to perform malware classification based on malware images and n-gram features.
- Rongyu Wang: Applied and tuned RF and SVM to classify malware types according to images features and n-gram features and evaluated importance of features.

Project Link

Code: https://github.com/Kur1su0/network_proj
Data: /u/zw9ga/net_proj/data/subtrain on cs portal