Reducing Malware labeling Efforts Through Efficient Prototype Selection

Guanhong Chen, Shuang Liu
College of Intelligence and Computing, Tianjin University, Tianjin, China
chenguanhong@tju.edu.cn, shuang.liu@tju.edu.cn

2022 26th International Conference on Engineering of Complex Computer Systems (ICECCS) | 978-1-6654-0162-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICECCS54210.2022.00011

***Abstract*—Malware detection and malware family classification are of great importance to network and system security. Currently, the wide adoption of deep learning models has greatly improved the performance of those tasks. However, deep-learning-based methods greatly rely on large-scale high-quality datasets, which require manual labeling. Obtaining a large-scale high-quality labeled dataset is extremely difficult for malware due to the domain knowledge required. In this work, we propose to reduce the manual labeling efforts by selecting a representative subset of instances, which has the same distribution as the original full dataset. Our method effectively reduces the workload of labeling while maintaining the accuracy degradation of the classification model within an acceptable threshold. We compare our method with the random sampling method on two widely adopted datasets and the evaluation results show that our method achieves significant improvements over the baseline method. In particular, with only 20% of the data selected, our method has only a 2.68% degradation in classification performance compared to the full set, while the baseline method has a 6.78% performance loss. We also compare the effects of factors such as training strategy and model structure on the final results, providing some guidance for subsequent research.**

***Index Terms*—malware, machine learning, prototype selection**

I. Introduction

Malware (i.e., malicious software) has always been one of the major threats to network security. Driven by the high benefits, malicious developers have been trying to use malware for purposes, which has caused severe damage to both personal computers and network systems.

To evade detection, malicious files belonging to the same malware family are constantly modified and/or obfuscated using various tactics, resulting in many different files in each family [15]. As a result, a few original malicious files are expanded into a large number of different malicious files [15]. Therefore, classifying malware into families based on the forms of malicious behavior for further analysis has been an important task in this domain.

Machine learning, especially deep learning, has made a big splash in various fields in recent years. Due to its excellent performance, researchers have also tried to use various machine learning approaches to solve the malware family classification problem and achieved impressive results [13], [18], [21], [22], [26]. However, the training of machine learning models relies on a large amount of labeled data, and lacking labeled datasets has become a bottleneck limiting the advancement of the field [6]. There are websites, such as VirusTotal, VirusShare, VX heaven, that offer to share malware and provide detection results from many Anti-Virus engines. Unfortunately, it is often difficult for Anti-Virus engines to reach a consensus on the family label of the same malware file [16], mainly due to two reasons, i.e. different Anti-Virus engines usually have different detection results and sometimes different vendors (which provide the Anti-Virus engines) even have different rules to name a malware family. As rules used by different engines may not be comprehensive enough, a more detailed manual analysis of the documents is more reliable, yet is also more costly.

Unlike the other fields, such as natural language processing and computer vision, which usually obtain labeled data through crowd sourcing, the annotation of malware samples requires expert domain knowledge and it is hard to find qualified labelers [1]. Based on our preliminary survey, there is a gap on effective solutions to such problems. Therefore, there are currently few relevant datasets for the domain of malware classification, and some of the approaches do not public their datasets, which discourages further research and industrial applications.

To relieve the manual efforts of labeling malware, and at the same time, maintain high classification accuracy. We propose to adopt a prototype selection process to select a subset of instances that have the same distribution of the full set of unlabeled data, and then label the subset of selected instances to train deep learning models. We pre-process the malware into a specific format and further adopt the prototype selection method to select a subset of the original dataset such that the subset has the same distribution as the original dataset. In this way, we can reduce the amount of data that needs to be labeled, and thus reduce the cost of labeling malware, with an acceptable loss of classification accuracy.

According to the work of Ma et al. [11], image-based malware classification techniques have achieved the best performance among all types of similar approaches. Moreover, image-based approaches are more time-efficient and can be universally applied in that they only require binary files to obtain the corresponding image files. Experiments [21] have also shown that this conversion method is resistant to escaping techniques, such as code obfuscation, to some extend. Therefore, we adopt the image-based malware classification approach, which takes RGB images (converted from malware files) as input, leveraging on the feature extraction layers of

Shuang Liu is the corresponding author.

978-1-6654-0162-3/22/$31.00 ©2022 IEEE
DOI 10.1109/ICECCS54210.2022.00011

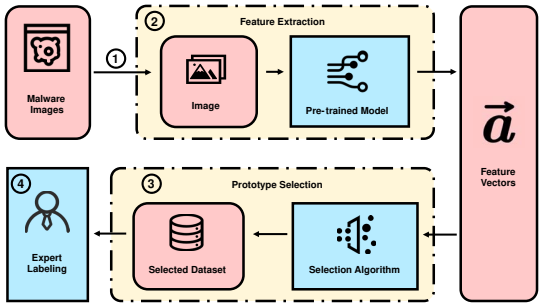


Fig. 1. Overview of our approach

the CNN classification models (e.g., VGG [17], Resnet [8] and InceptionV3 [19]) to obtain the hidden layer's output as feature vectors.

Our approach does not assume any label information before performing prototype selection, and thus a prototype selection algorithm that does not rely on labels is desired. Considering this, we adopt the prototype selection algorithm MMD-critic [9] to select prototypical examples from the full dataset represented in the feature vector space obtained from the previous step. MMD-critic is a state-of-the-art example-based prototype selection algorithms, which can effectively use the distribution of the vector space to select a subset that better represents the original set without label information. Lastly, the selected instances are labeled by experts and used to train a malware classification model.

To summarize, our work makes the following contributions:

1) We take the first attempt to tackle the dataset curation problem in the domain of malware family classification and propose to eliminate the labeling efforts with prototype selection;
2) We propose a complete process, including pre-processing, feature extraction and prototype selection, to select a subset of the dataset for manual labeling;
3) We conduct experiments with two widely adopted malware classification datasets. The results show that our method is effective in selecting a small subset (20%) of representative instances. The models trained with the selected subset has only 2.68% F1-score degradation on the classification task compared with the models trained with the full set of training data, and 4.1% F1-score improvement compared with the models trained with the random sampled dataset;

II. Our Approach

A. Overview

Figure 1 shows the overview of our approach, which consists of four steps, i.e., (1) pre-processing, (2) feature extraction, (3) prototype selection and (4) expert labeling. We will introduce each step in detail in the following sections.

B. Pre-processing

The malware usually needs to be converted into a form that the program can recognize, for it is not practical to use the entire file as the input of the analysis process. In the field of malware family classification, depending on the features extracted, we can classify all the works as binary-based methods, assembly-based methods, and image-based methods [11].

According to the experiment study by Ma et al. [11], image-based malware classification methods achieve state-of-the-art performance with low overhead. Therefore, we adopt the image-based methods in our approach. The method of converting malicious files into images based on binary bits enables information to be extracted quickly, without resorting to any expert knowledge [21]. We adopt the same pre-processing method as IMCFN [21], where each binary file is treated as a sequence of 8-bit unsigned integers. This sequence is then scaled to a 2D array depending on the file size, which can then be viewed as a grayscale map. Finally, this grayscale map is mapped to an RGB image using a color-map.

C. Feature Extraction

Since the prototype selection algorithm needs to process the data in vector space and cannot process the images directly, we need to convert the images obtained from the pre-processing step into vectors. As CNN models have achieved impressive results in the domain of image processing and computer vision, we are motivated to adopt CNN models as feature extraction models in our work. In particular, we adopt the three CNN models, i.e., VGG16, Resnet50, and InceptionV3 for feature extraction. As a supervised learning model, the CNN model must be trained with labelled data. Therefore, before performing the feature extraction process, we need to train the parameters of the CNN models. However, CNN models with a large number of parameters are easily over-fitted when trained on small datasets, resulting in insufficient training. Since transfer learning has made great advance in improving the performance of malware family classification task [21], [22]. We use transfer learning to train the model. Transfer learning means storing knowledge gained while solving one problem and applying it to a different but related problem [23]. In particular, we employ a fine-tuning strategy from transfer learning to train our feature extraction model.

Fine-tuning The fine-tuning strategy uses the parameters of the model that trained on the pre-training dataset to initialize the target model and uses the dataset of the target problem to tune the parameters. Following the standard, we use the ImageNet dataset [4], a benchmark dataset in the computer vision domain which contains over one million images, as the pre-training dataset. After the pre-training, all the fully connected layers of the adopted models are replaced with three fully-connected layers and were fine-tuned using a malware dataset. In the fine-tuning process, the malware dataset is used to train the added fully connected layers, while the other pre-trained layers are frozen at first. We will then open up the

TABLE I
EVALUATING RESULT ON BIG-15

Classification Model	Selection Method	10% Data			20% Data		
		Percision	Recall	F1-score	Percision	Recall	F1-score
VGG16	All	97.36	97.33	97.27	97.36	97.33	97.27
	VGG16+MMD-critic	91.08(-6.45)	91.15(-6.35)	91.01(-6.44)	**94.02(-3.43)**	**94.38(-3.03)**	**94.16(-3.2)**
	Resnet50+MMD-critic	**92.42(-5.08)**	**92.26(-5.21)**	**92.27(-5.14)**	93.75(-3.71)	94.01(-3.41)	93.84(-3.53)
	InceptionV3+MMD-critic	91.32(-6.21)	91.43(-6.06)	91.26(-6.18)	94.02(-3.44)	94.38(-3.03)	94.15(-3.21)
	Random-avg	87.77(-9.86)	87.65(-9.95)	86.65(-10.92)	92.96(-4.53)	92.95(-4.50)	92.86(-4.53)
Resnet50	All	97.65	97.61	97.56	97.65	97.61	97.56
	VGG16+MMD-critic	**87.68(-10.21)**	**86.54(-11.33)**	**86.55(-11.28)**	92.17(-5.61)	91.98(-5.76)	91.99(-5.71)
	Resnet50+MMD-critic	87.4(-10.49)	84.76(-13.16)	86.24(-11.60)	90.43(-7.39)	90.6(-7.18)	90.24(-7.50)
	InceptionV3+MMD-critic	85.21(-12.74)	85.99(-11.90)	85.27(-12.60)	**92.82(-4.94)**	**92.81(-4.91)**	**92.64(-5.04)**
	Random-avg	83.08(-14.91)	79.28(-18.77)	79.22(-18.80)	90.54(-7.28)	90.23(-7.56)	90.13(-7.62)
InceptionV3	ALL	98.45	98.43	98.42	98.45	98.43	98.42
	VGG16+MMD-critic	**93.64(-4.89)**	**93.83(-4.68)**	**93.68(-4.81)**	95.08(-3.42)	95.48(-3.00)	95.24(-3.23)
	Resnet50+MMD-critic	92.16(-6.39)	92.44(-6.09)	92.2(-6.32)	94.57(-3.95)	94.47(-4.03)	94.37(-4.11)
	InceptionV3+MMD-critic	93.31(-5.22)	93.55(-4.96)	93.3(-5.20)	**96.84(-1.64)**	**96.96(-1.50)**	**96.85(-1.59)**
	Random-avg	90.42(-8.16)	83.4(-15.28)	83.23(-15.43)	90.37(-8.21)	83.23(-15.45)	83.41(-15.24)

We use ALL to represent the case where the full set is used, and the number in brackets is the proportional performance drop compared to the full set. Random-avg is the value averaged using ten different randomly selected seeds. The best performance corresponding to each classification model (other than the full set) has been indicated in **bold**.

TABLE II
EVALUATING RESULT ON MALIMG

Classification Model	Selection Method	10% Data			20% Data		
		Percision	Recall	F1-score	Percision	Recall	F1-score
VGG16	All	98.98	98.82	98.83	98.98	98.82	98.83
	VGG16+MMD-critic	96.36(-2.64)	96.25(-2.60)	96.19(-2.68)	97.29(-1.71)	97.32(-1.52)	97.26(-1.60)
	Resnet50+MMD-critic	**96.98(-2.02)**	**97.11(-1.74)**	**96.92(-1.94)**	**97.54(-1.46)**	**97.54(-1.30)**	**97.47(-1.38)**
	InceptionV3+MMD-critic	96.49(-2.52)	96.25(-2.60)	96.30(-2.57)	97.26(-1.74)	97.11(-1.74)	97.01(-1.85)
	Random-avg	95.84(-3.17)	95.61(-3.25)	95.42(-3.45)	96.90(-2.10)	96.68(-2.17)	96.64(-2.22)
Resnet50	All	97.81	98.07	97.86	97.81	98.07	97.86
	VGG16+MMD-critic	91.34(-6.61)	93.03(-5.14)	91.91(-6.08)	95.30(-2.57)	95.28(-2.84)	94.71(-3.22)
	Resnet50+MMD-critic	91.34(-6.62)	**93.14(-5.03)**	91.72(-6.27)	95.23(-2.64)	95.28(-2.84)	94.87(-3.06)
	InceptionV3+MMD-critic	**92.36(-5.58)**	93.03(-5.14)	**92.39(-5.59)**	**95.71(-2.14)**	**95.39(-2.73)**	**95.00(-2.92)**
	Random-avg	88.31(-9.71)	86.64(-11.66)	83.92(-14.24)	91.83(-6.11)	91.93(-6.27)	90.86(-7.15)
InceptionV3	ALL	98.24	98.18	98.14	98.24	98.18	98.14
	VGG16+MMD-critic	93.99(-4.33)	93.68(-4.59)	93.27(-4.97)	**96.71(-1.55)**	**96.79(-1.42)**	**96.64(-1.53)**
	Resnet50+MMD-critic	93.74(-4.58)	91.96(-6.33)	92.07(-6.19)	95.52(-2.77)	95.61(-2.62)	95.30(-2.90)
	InceptionV3+MMD-critic	**95.22(-3.07)**	**94.75(-3.49)**	**94.49(-3.72)**	96.53(-1.74)	96.57(-1.64)	96.46(-1.72)
	Random-avg	88.74(-9.67)	89.64(-8.70)	88.11(-10.22)	94.62(-3.68)	94.75(-3.49)	94.26(-3.96)

convolution layers of the pre-trained model to be trained with the data, but the parameters of the added fully connected layers will be frozen.

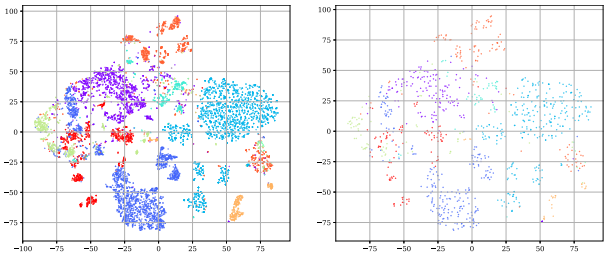
After the model is fine-tuned, the output of the fully connected layer before the classification layer in the model will be used as a feature vector for the malware image to be used in the next step of prototype selection.

D. Prototype Selection

In order to select prototype samples, the MMD-critic algorithm [9] is adopted in our work. MMD-critic provides two options, a $local$ mode, which requires the label information, and a $global$ mode, which only requires unlabeled data. Recall that in our scenario, all the datasets that need to be selected do not have any label. Therefore the global mode of MMD-critic is applied to our process. In our approach, we only need to set the number of prototypes to be selected, and the MMD-critic algorithm generates the corresponding number of prototypes to represent the original data. The selection ratio should be adjusted according to the actual situation, and how the selection ratio affects the results will be discussed in the subsequent experimental section.

E. Expert labeling

Finally, the selected data will be carefully analyzed and annotated by experts. The labeled data can be used for downstream tasks. Since usually only a small subset (10%-20% in our experiment) of the original dataset selected can achieve a good performance on downstream tasks, the number of data to be annotated, and thus the time and labor cost can be greatly reduced.



(a) All points (b) Selected points

Fig. 2. Visualization of selected set and full set of BIG-15 based on the feature extracted by Resnet model

III. Evaluation

We report our evaluation results in this section. All of our experiments are conducted on a server with 2 Intel Xeon Platinum 8260 CPU @2.30GHz, 8*Nvidia GeForce RTX2080 Ti GPU (11GB memory for each), and 512GB RAM. Only one GPU is used in our experiments. We implement our approach in Python, leveraging on built-in deep learning models in Tensorflow and Keras.

A. Datasets

We adopt BIG-15 [15] and Malimg [12] to validate our approach, as both datasets are widely used to evaluate the malware family classification methods.

BIG-15 was collected and published by Microsoft and consists of a total of 10,868 samples of malware from 9 families. The raw data for each file of malware was converted to a hexadecimal representation of the binary content with the headers removed.

Malimg contains only grey-scale images converted from malicious samples and is widely used in the field of malware resolution using image-based classification methods. A total of 9,435 greyscale images are included in this dataset, forming 25 malware families.

B. Evaluation settings

To evaluate the performance of our prototype selection process, we conduct quantitative evaluation, in which we train the malware family classification models with the selected subset and compare the model with the model trained with the full set of data on the classification performance. We also compare our selection approach with the random selection approach. In this way, we use the classification results as an indirect measurement to evaluate the quality of the selected subset of data.

We also adopt VGG16, Resnet50, InceptionV3 models as the classification models. Note that although we use the same set of models in the feature extraction step, they are served as totally different purposes. For the feature extraction step, we only use the parameters of the feature extraction layers to obtain the feature vectors of the given malware images. The training purpose of feature extraction is to obtain accurate feature representations so as to conduct prototype selection.

When training the feature extraction model, the batch size is set to 64, and the number of training epochs is set to 100. We adopt the Adam optimizer and the learning rate is set to $1 * 10^{-4}$. The MMD-critic algorithm uses the global strategy, i.e., ignoring the label information, and the kernel function is the radial basis function (RBF) function with γ set to $2.6 * 10^{-2}$. The classification model is trained with a batch size of 64 and trained for 300 epochs. The optimizer is Adam with learning rate of $1 * 10^{-4}$.

We randomly shuffled the original dataset and used 10% of the malware samples as testing set. The remaining 90% is used as the full dataset for prototype selection and random selection. In the following, the full set is used to refer to the set that excludes the test set. This random selection process was performed 10 times with different random seeds and the final result was averaged over ten runs to eliminate the effect of randomness. Following the standard, we partition the dataset (i.e., randomly selected, prototype selected or the full dataset) into 8:1 for training and validation. The testing set is the previously selected 10% of the full dataset and is used for all three trained models.

Note that when training the feature extraction models with fine-tuning strategy, we need to use a dataset that is different from the dataset that needs to be selected as the training set to fine-tune the models. So we use the BIG-15 dataset as the training set of feature extraction models for the Malimg dataset and vice versa. And the full dataset was departed to training set, validation set and test set with the standard rate 8:1:1. We treat the feature extraction procedure as the default setting, and subsequent datasets are referred to as the datasets to be selected unless specifically claimed.

C. Evaluation results

Table I and Table II show the performance of malware classification models trained with the selected subset of malware samples on BIG-15 and Malimg, respectively. Following the standard, we use precision, recall and F1-score as measurements and bold the best results in each classification model. The classification model column lists the models which are trained with the selected subset of malware samples. The selection method column lists the models used in the feature extraction step. All indicate the full set of malware samples are used for training and Random-avg represents the average of running the random sampling strategy 10 times.

We use the decrease of F1-score compared to the F1-score of the full set as the criteria of performance loss. Our method has only an average 6.06% performance degradation when selecting 10% of the data compared with using the full set. On the Malimg dataset, the samples selected by our method even only get 4.39% performance loss. This indicates that the data selected by our process can even represent the distribution of the original dataset well, and the classifier can learn the distribution characteristics of the data well even if the number of the selected subset is small.

We tried to select a larger number of data to observe the changes that occurred. When 20% of all data was selected as

TABLE III
TIME TO COVERAGE

	BIG-15			Malimg		
	epochs	time(s)	speed	epochs	time(s)	speed
VGG16	42	81.34	1.94	37	64.75	1.75
Resnet50	154	285.93	1.87	164	260.52	1.56
InceptionV3	108	124.21	1.15	96	112.91	1.17

Epochs means the number of epochs that the models need to coverage. Time records how many seconds were consumed by the model to trigger the early stop mechanism. Speed indicates the time consumed per epoch.

TABLE IV
THE INFLUENCE OF FINE-TUNE

Dataset	Strategy	VGG16	Resnet50	InceptionV3
BIG15	ImageNet	84.2	83.54	91.74
	Malware	87.43	83.22	90.65
	ImageNet+Malware	91.01	86.55	93.68
Malimg	ImageNet	91.31	89.23	90.43
	Malware	92.73	87.95	88.06
	ImageNet+Malware	96.19	91.91	93.27

ImageNet means that we use the models pre-trained by ImageNet as the feature extraction models. Malware means we only use a malware dataset to train the models. The last row refers to the models that were trained with fine-tune strategy.

prototype samples, our method only had a performance loss of 2.68%. After selecting 20% of the data, we are able to train very accurate classifiers, and the best one has achieved an F1-score of 97.47%, improving the accuracy by 3.38% over selecting 10% of the data.

In contrast, there is a very large performance penalty for using randomly selected data. Randomly selecting 10% of the data results in an average loss of 12.17% of classification performance. Even when 20% of the data is selected, random selection loses 6.78% of the classification performance, a performance that is not even as good as when our method selects 10% of the data.

Overall, our selection process can effectively extract the distribution of the data, as evidenced by the fact that the classification performance can be retained to the maximum extent. Random selection, however, results in a very large performance loss and cannot be used as a method to reduce the dataset size. Note that limited by the size of the datasets used, 20% of samples were chosen in this paper. In practical situations, the proportion chosen can be adjust according to the large amount of data.

D. Discussion

1) Visualization of the experiment results: We use MMD-critic to select the prototypes based on the feature extracted by the CNN classification models. The feature spaces extracted by the CNN models can greatly affect the performance of MMD-critic. Figure 2 shows a visual representation of the BIG-15 feature space extracted by the Resnet model, where we have mapped the features by a dimensional reduction algorithm [20] to a 2D space. Figure 2(a) shows all the data in the full set of BIG-15. All the points that belong to the same classes share the same color. From the figure, we can see that most of the points belonging to the same category tend to cluster together. This reflects that our feature extraction model can extract valid information that contains the differences between malware classes and the characteristics of the classes. The visualization of the selected prototypes is shown in Figure 2(b). The distribution of points in the figure looks very similar to the distribution of the full set, which shows that the selected prototypes can fit the original distribution well. **This result shows that our feature extraction model does extract the key features to help us select the prototype samples.** This is one of the reasons why our feature extraction method combined with MMD-critic can achieve relatively good results.

2) Compare between three CNN models: From Table 1 and Table 2, we find that the three different feature extraction model structures also show differences in classification performance when the same classification model is used. The Resnet model does not seem to perform well when used as a classification model, but the effectiveness of the classification model is beyond in the scope of this paper.

In terms of the final classification results, there is not much difference in the effectiveness of the three feature extraction models. However, VGG16 and InceptionV3 are in general more likely to achieve better results compared to Resnet50.

In addition, we examined the model convergence speed to compare the training time cost of the model. We adopt an early-stop strategy when training the feature selection model. When the value of the model's loss function does not decrease for 20 epochs, we terminate the training process and record the number of rounds and time spent on training. The results are shown in Table III, where VGG16 takes only a short time to converge and is significantly faster than the other two models.

Taking into account the effect and convergence speed both, we prefer VGG16 as the feature selection model.

3) The influence of transfer learning: To avoid under-training the models, we used ideas from transfer learning to train the feature extraction models with a fine-tuning strategy. We designed some experiments here to verify whether this training strategy can bring an improvement in effectiveness. We only changed the training strategy of the feature extraction model, and the other steps remain the same, to extract 10% of the data. Then the F1 value of the VGG16 classification model is used as the criteria to measure the effectiveness of our selected data.

As is shown in Table IV, the fine-tuned models achieve the best results when compared with other strategies. Considering that pre-trained models can still achieve good results, the performance loss of using pre-trained models can be ignored in some scenarios where the training costs need to be saved. In particular, when there is not enough dataset for fine-tuning, using pre-trained models directly is also a reasonable approach. **However, in general, fine-tuning strategies using migration learning can considerably improve the performance of feature extraction models.**

IV. Related Works

A. Malware Family Classification

Learning-based approaches, especially deep learning algorithms [13], [14], [18], [21], [22], [26] have shown great potential in the problem of malware family classification. We provide a detailed discussion on this line of approaches, which our work is directly related to.

Some works [13], [14] take the binary code of the malware as a feature and feeds it into the model for classification. Due to the difficulty of obtaining the source code of malware, some methods [18], [26] attempt to use disassembly tools to restore the logic of the software and thus make a classification. The direct conversion of bytes to grey-scale picture is also a possible extraction method, as is the strategy adopted by IMCEC [22], IMCFN [21].

B. Prototype Selection

Prototype selection was proposed for reducing the size of training set or explain the decision of models [3].

Most prototype selection methods [2], [5], [7], [10], [24], [25] assume that the labels of instances are given. We aim to solve the problem of reducing labeling efforts, in which labels are not accessible at the time of prototype selection. In addition, the main goal of our application prototype selection is to restore the original distribution with as few instances as possible, therefore a distribution-based algorithm is more relevant to our scenario. MMD-critic [9] does not require labelled data and targets obtaining a subset having the same distribution as the original set.

V. Conclusion

In this work, we present a process for extracting prototype samples from large amounts of malware, with the purpose of reducing manually labeling efforts. Our process can effectively reduce the number of malware samples to be labeled for training a malware family classification model. This approach is based on CNN classification models and prototype selection algorithms. Our experiments confirmed that this process can effectively reduce the amount of data while ensuring the classification performance of the classification model. In particular, three interesting conclusions emerged from our analysis: 1) CNN feature extraction models do extract the key feature information to help us select the prototypes; 2) We recommend using VGG16 as the feature extraction model as it provides a good balance of accuracy and speed; 3) Fine-tuning strategies can help us train better feature extraction models. But in cases where training conditions are constrained, using pre-trained models only can achieve good results too.

Acknowledgment

The work is supported by the National Natural Science Foundation of China (U1836214, 61802275) and Innovation Fund of Tianjin University 2020XRG-0022.

References

[1] Hyrum S Anderson and Phil Roth. Ember: an open dataset for training static pe malware machine learning models. *arXiv:1804.04637*, 2018.
[2] Joel Luis Carbonera and Mara Abel. A density-based approach for instance selection. In *ICTAI*, pages 768–774. IEEE, 2015.
[3] Gonzalo Cerruela-García, Aida de Haro-García, José Pérez-Parras Toledano, and Nicolás García-Pedrajas. Improving the combination of results in the ensembles of prototype selectors. *Neural Networks*, 118:175–191, 2019.
[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255. Ieee, 2009.
[5] Geoffrey Gates. The reduced nearest neighbor rule (corresp.). *IEEE transactions on information theory*, 18(3):431–433, 1972.
[6] Daniel Gibert, Carles Mateu, and Jordi Planes. The rise of machine learning for detection and classification of malware: Research developments, trends and challenges. *JNCA*, 153:102526, 2020.
[7] Peter Hart. The condensed nearest neighbor rule (corresp.). *TIT*, 14(3):515–516, 1968.
[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.
[9] Been Kim, Oluwasanmi Koyejo, Rajiv Khanna, et al. Examples are not enough, learn to criticize! criticism for interpretability. In *NIPS*, pages 2280–2288, 2016.
[10] Enrique Leyva, Antonio González, and Raúl Pérez. Three new instance selection methods based on local sets: A comparative study with several approaches from a bi-objective perspective. *Pattern Recognition*, 48(4):1523–1537, 2015.
[11] Yixuan Ma, Shuang Liu, Jiajun Jiang, Guanhong Chen, and Keqiu Li. A comprehensive study on learning-based pe malware family classification methods. In *FSE/ESEC*, pages 1314–1325, 2021.
[12] Lakshmanan Nataraj, Sreejith Karthikeyan, Gregoire Jacob, and Bangalore S Manjunath. Malware images: visualization and automatic classification. In *VizSec*, pages 1–7, 2011.
[13] Yanchen Qiao, Bin Zhang, and Weizhe Zhang. Malware classification method based on word vector of bytes and multilayer perception. In *ICC*, pages 1–6. IEEE, 2020.
[14] Edward Raff, Jon Barker, Jared Sylvester, Robert Brandon, Bryan Catanzaro, and Charles Nicholas. Malware detection by eating a whole exe. *arXiv:1710.09435*, 2017.
[15] Royi Ronen, Marian Radu, Corina Feuerstein, Elad Yom-Tov, and Mansour Ahmadi. Microsoft malware classification challenge. *arXiv:1802.10135*, 2018.
[16] Marcos Sebastián, Richard Rivera, Platon Kotzias, and Juan Caballero. Avclass: A tool for massive malware labeling. In *RAID*, pages 230–253. Springer, 2016.
[17] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, 2014.
[18] Guosong Sun and Quan Qian. Deep learning and visualization for identifying malware families. *IEEE TDSC*, 2018.
[19] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, pages 2818–2826, 2016.
[20] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.
[21] Danish Vasan, Mamoun Alazab, Sobia Wassan, Hamad Naeem, Babak Safaei, and Qin Zheng. Imcfn: Image-based malware classification using fine-tuned convolutional neural network architecture. *Computer Networks*, 171:107138, 2020.
[22] Danish Vasan, Mamoun Alazab, Sobia Wassan, Babak Safaei, and Qin Zheng. Image-based malware classification using ensemble of cnn architectures (imcec). *Computers & Security*, 92:101748, 2020.
[23] Jeremy West, Dan Ventura, and Sean Warnick. Spring research presentation: A theoretical foundation for inductive transfer. *Brigham Young University, College of Physical and Mathematical Sciences*, 1(08), 2007.
[24] D Randall Wilson and Tony R Martinez. Reduction techniques for instance-based learning algorithms. *Machine learning*, 38(3):257–286, 2000.
[25] Dennis L Wilson. Asymptotic properties of nearest neighbor rules using edited data. *IEEE SMC*, (3):408–421, 1972.
[26] Jiaqi Yan, Guanhua Yan, and Dong Jin. Classifying malware represented as control flow graphs using deep graph convolutional neural network. In *DSN*, pages 52–63. IEEE, 2019.