LIFT: Discriminant Classification Approach of Malware Family on Time Consistent Open Set

Yuxia Sun(✉), Siyi Pan, Yang Yang, Ziying Huo, and Yuqin Hong

College of Information Science and Technology, Jinan University, Guangzhou, China
tyxsun@email.jnu.edu.cn

Abstract. In recent years, the rapid evolution of malware, including the emergence of new variants and families, has posed a significant challenge to antivirus defenses. To address this, some approaches to Malware Open Set Recognition (MOSR) have been introduced, aiming to categorize known malware families and detect new ones. Nevertheless, current MOSR research tends to neglect the effect of time inconsistency within datasets, which can result in an overly optimistic evaluation of MOSR effectiveness. To mitigate this issue, a new MOSR method named LIFT has been developed, focusing on a time-consistent division of malware datasets. LIFT employs the self-attention mechanism to understand the correlations among known malware families. It also integrates a linear probe and a unique regularization term to enhance the separability of deep representations. In the recognition phase, LIFT implements a feature truncation tactic to adjust the dimensional values in the samples' deep representation vectors, thereby enhancing their distinctiveness. Tests on time-consistent open datasets demonstrate that LIFT significantly outperforms existing techniques in open set recognition efficiency.

Keywords: Malware family classification · Time-consistent dataset · Open set recognition · Discriminative model

1 Introduction

By 2024, over 1.3 billion malicious programs (malware) and potentially unwanted applications (PUA) had been identified, with more than 300,000 new malware samples detected daily [1]. It encourages more malware detection research to accurately identify both existing and emerging malware families. Malware Open Set Recognition (MOSR) aims to classify each known family and identify new malware families using a single classifier. Many existing MOSR methods overlook real-world data distributions over time, leading to temporal bias. This bias occurs when models use future data for training, learning future knowledge that is unavailable at the time of the actual deployment of the model [2], often resulting in exaggerated malware detection results [3].

To adapt to changing environments, models should be trained with time-consistent datasets. In this paper, we propose a discriminative MOSR framework called LIFT. We utilize the self-attention mechanism of Swin Transformer [24, 25] to capture the data relevance of known families, enhance deep representation distinguishability with

D.-S. Huang et al. (Eds.): ICIC 2024, LNAI 14875, pp. 37–49, 2024.
https://doi.org/10.1007/978-981-97-5663-6_4

a linear probe technique, and revise the values of the representation vectors to their typical values with a feature truncation strategy to improve their distinctiveness. In the recognition phase, LIFT classify with a distance-based approach, ultimately achieving favorable open set recognition accuracy.

2 Related Work

2.1 Malware Open Set Recognition (MOSR)

Existing malware family classification methods can be categorized into three main types based on file format: image-based [4], binary-based [5], and disassembly-based methods [6]. Open set recognition (OSR) originated from the field of computer vision and addresses scenarios where test sets include classes unseen during training. This requires classifiers to not only accurately categorize known classes but also effectively distinguish unknown ones [7]. There are relatively few existing studies on malware open set recognition [8–11]. Depending on the composition of the training set, OSR methods can be divided into those requiring additional data [12, 13] and those that do not [14, 15]. The method with additional data further splits into using real data or generated data, with the latter usually employ generative adversarial networks or sampling techniques to create or mimic samples of unknown classes.

Guo et al. [10] proposed a probabilistic-based method with generating additional data for MOSR. To conservatively synthesize malware instances, Guo jointly trained the GAN with the classifier so that the classification probability of generated samples is close to a uniform distribution. In addition, Guo et al. proposed a local regularization for minimizing the batch-level recognition probabilities. This method utilizes generated data to produce malware static feature images of unknown families. Compared to the methods with additional data, discriminative approaches do not rely on additional data. It focuses on analyzing data from known categories, aiming to improve the separability between representations of known categories, thereby leaving more distinct spaces for unknown categories. Our LIFT method is a discriminative approach.

2.2 Malware Recognition with Temporal Bias

Pendlebury et al. [2] emphasized the significance of temporal consistency in assessing the effectiveness of malware classifiers. Experimental evidence from Fu et al. [16] and Cai et al. [3] has demonstrated that temporal bias within Android malware datasets results in diminished performance of malware detectors. Despite its importance, temporal bias has been infrequently addressed in malware family identification. For instance, Yang et al. [17] proposed a framework called CADE, which utilizes reconstruction loss and comparison loss to identify unknown family samples through hidden vectors generated by the encoder. Yuan et al. [18] developed a dual-headed neural network strategy to identify anomalous malware samples. These samples are then removed from the malware dataset, enabling the refined dataset to further train a multiclass malware family classifier.

In summary, research on open-set recognition in malware remains scarce, with most studies on MOSR neglecting the impact of temporal bias. This paper proposes an effective discriminant malware family open-set classification model capable of classifying known families and identifying unknown families on time-consistent datasets.

3 Methodology

3.1 Overview

To capture enough information from malware samples evolving over time, LIFT takes malware feature vectors as inputs and utilizes the framework as shown in Fig. 1 to obtain a target classifier to classify samples into one of the known families or an unknown family. LIFT contains four components: the primary model module, which includes the feature extractor Swin Transformer Z_{tran} and the classification head Z_{cls}; the linear probe module Z_{prob}; the regularization module $RegM$; and the truncation processing module TrM.

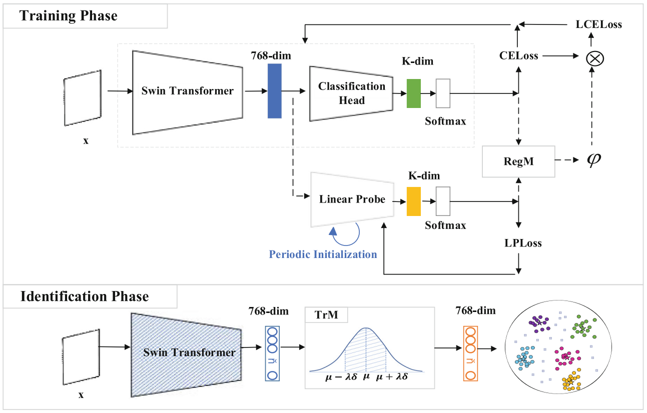

Fig. 1. LIFT Framework.

Throughout the training phase, the Swin Transformer extracts deep representations from malware static feature images. Subsequently, these representations are fed into the classification head and the linear probe respectively. After calculating the classification probability and cross-entropy loss in both networks, the regularization module determines the regularization coefficient by assessing the disparity between the two probability distributions. Then the regularization coefficient is used to adjust the magnitude of the cross-entropy loss of classification head. The feature extractor, classification head, and linear probe are trained together to obtain unique deep representations from the feature extractor. In recognition phase, the malware static feature images from test set are fed into the trained feature extractor, followed by the truncation processing module that corrects the extracted representations and accomplishes the recognition process using a distance-based approach.

3.2 Feature Extraction Based on Linear Probe Boosted Swin Transformer

During training, the Transformer-based malware classification model may encounter the issue of overfitting. However, there is still potential for enhancing the distinguishability of deep representations among different families. Linear probes are frequently employed to assess the ability of feature extractors [19]. Incorporating linear probes during training can help models focus more on challenging samples and reduce overfitting [20]. Therefore, LIFT jointly train the Swin Transformer feature extractor, the classification head and the linear probe, where the weight parameters of the linear probe are periodically randomly initialized. The regularization coefficients dynamically adjust the cross-entropy loss of each training sample, enhancing the discriminability of the deep representations produced by the feature extractor. The detail of the model and the procedure for updating the parameters during the training process are as follows.

Primary Model. The feature extractor Z_{tran} receives a malware static feature image as input and produces a 768-dimensional feature vector for the classification head. The classification head Z_{cls} produces an embedding with the same dimensions as the number of known families in the training set:

$$f_i = Z_{tran}(x_i) \tag{1}$$

$$z_i = Z_{cls}(f_i) \tag{2}$$

The softmax layer takes the embedding z_i as input and outputs the predicted probability $P_i = \left(p_i^1, p_i^2, \ldots, p_i^K\right)$, where each element p_i represents the probability of sample i belonging to one of the K known families. Subsequently, the cross-entropy loss is computed based on the predicted probability and the true family label of this sample:

$$CELoss_i = -\sum_{j=1}^{K} y_i^j log\left(p_i^j\right) \tag{3}$$

where y_i^j equals 1 if j represents the true family label, and 0 otherwise.

Linear Probe Module. The Z_{prob} module takes the deep representation extracted by the feature extractor Z_{tran} as input and outputs the embedding $\hat{z}_i$:

$$\hat{z}_i = Z_{prob}(f_i) \tag{4}$$

Similarly, the softmax layer of linear probe like the softmax layer of primary model, generating the predicted probability $\widehat{P}_i = \left(\hat{p}_i^1, \hat{p}_i^2, \ldots, \hat{p}_i^K\right)$. The model weight parameters of the linear probe are updated using cross-entropy loss:

$$LPLoss_i = -\sum\nolimits_{j=1}^{K} y_i^j log\left(\hat{p}_i^j\right) \tag{5}$$

It is important that the independence of the linear probe is maintained by preventing gradients backpropagated to the feature extractor during training. If the linear probe can accurately classify training samples with its inherently limited classification capability, it indicates that the deep representations are sufficiently distinct. To avoid the linear probe overfitting, the model parameters of the linear probe are randomly re-initialized every two epochs during training.

Regularization Module. LIFT have a regularization coefficient φ_i that utilizes the predicted probabilities of primary model and linear probe, which can better measure the gap between the classification performance of the two models. To improve the main model, LIFT integrates the probability distributions from the classification head P_i and the linear probes $\widehat{P}_i$ into the regularization processing module *RegM*. This module computes the regularization coefficients φ_i, which dynamically adjust the cross-entropy loss of the main model. The coefficients are calculated using the following steps.

The first step is to calculate the difference information $info_i^j$ for sample x_i about family j in the classification head and linear probe:

$$info_i^j = p_i^j * log\left(p_i^j / m_i^j\right) + \widehat{p}_i^j * log\left(\widehat{p}_i^j / m_i^j\right) \tag{6}$$

Here, p_i^j is the probability that the classification head predicts sample x_i as belonging to family j, and $\widehat{p}_i^j$ is the probability that the linear probe predicts it as belonging to family j. The mean of these two probabilities, m_i^j, is defined as $m_i^j = \left(p_i^j + \widehat{p}_i^j\right)/2$.

To prevent the model from falling into an under-optimized state, $info_i^j$ is divided by the sum of the probabilities that sample x_i is predicted by both the classification head and the linear probe as belonging to its true family y_i, referred to as φ_i^j:

$$\varphi_i^j = \left(\frac{info_i^j}{p_i^{y_i} + \widehat{p}_i^{y_i}}\right)^{\gamma} \tag{7}$$

where γ is an integer greater than 1, used to adjust the magnitude of φ_i^j. The regularization coefficient φ_i is the average of $\varphi_i^1, \varphi_i^2, \ldots, \varphi_i^K$, where K is the number of known families:

$$\varphi_i = \frac{1}{K}\sum\nolimits_{j=1}^{K} \varphi_i^j \tag{8}$$

After calculating the regularization coefficients for the samples, LIFT multiplies the regularization coefficient φ_i by the cross-entropy loss of the sample in the main model's classification head as a regularization term:

$$LCELoss_i = \varphi_i * CELoss_i \tag{9}$$

During the training process, the gradient of the regularization term coefficient φ_i is not backpropagated to the main model or the linear probes. Therefore, this coefficient only affects the magnitude of $CELoss_i$. When there is a significant difference between the two probability distributions, it indicates that the linear probe is unable to accurately classify the current sample, which means that deep representation lacks sufficient discriminability. Consequently, the regularization coefficient increase, to prompt the model to focus more on this sample. If both two networks perform poorly in classification, the value of $p_i^{y_i} + \widehat{p}_i^{y_i}$ decreases, leading to a higher regularization factor. This adjustment helps prevent the main model from remaining under-optimized.

Total Training Loss. LIFT computes the total training loss using the loss functions from the three modules and updates the model parameters according to Eq. (10).

$$L = \frac{1}{n}\sum_{i=1}^{n} CELoss_i + LPLoss_i + LCELoss_i \tag{10}$$

where n is the number of training samples in a batch. In summary, in this paper, the main model module is used to ensure the basic classification performance, the linear probe module with periodically initialized is used to detect the quality of the deep representations extracted by the current feature extractor, and the regularization module is used to dynamically adjust the degree of attention paid by the model to each training sample based on the probability distributions of the classification head and the linear probe, and ultimately improve the distinguishability of the deep representations.

3.3 Feature Truncation Based on Distance

To enhance the model's ability to distinguish between samples from known and unknown families, this paper applies statistical principles to constrain deep representations during identification phase. The deep representations of the training set generally follow a Gaussian distribution across different dimensions. Based on whether the feature values fall within high-probability or low-probability regions, features can be categorized as either typical feature or extreme feature. Extreme features, which appear less frequently during training, may lead to vague and uncertain estimations. In contrast, classifiers are better at processing typical features. By limiting feature values to typical ranges, the influence of extreme features on recognition is minimized [21]. Consequently, we employ a feature truncation approach to refine deep representations.

Zhu et al. [21] proposed a feature rectification method that reinputs constrained deep representations back into the classification layer to obtain the predicted probability of the modified test samples. However, by feeding the constrained representation vectors back into the classification layer, this approach can lead to compression of useful information and loss of the expressive power of deep representations. In fact, after the model has been trained, there is inherent similarity between deep representations from the same family, so the test samples can be classified into either a known or unknown family based on the Euclidean distance between the deep representation vectors. Therefore, in this paper, the rectified deep representations and a distance-based approach are used to accomplish the MOSR task.

Step 1. After the training phase, training samples are reinput to the feature extractor Z_{tran} to acquire their deep representation vectors, denoted as f_{train}. These vectors constitute the set FT, which encompasses the deep representations of all training samples. Then, the mean and standard deviation across all dimensions of the deep representations in the set FT are computed, resulting in the mean vector $\mu = \left[\mu^1, \mu^2, \ldots, \mu^d\right]$ and the standard deviation vector $\sigma = \left[\sigma^1, \sigma^2, \ldots, \sigma^d\right]$, where d represents the dimension of the deep representations.

Step 2. The Truncation Processing Module (*TrM*) plays a crucial role in calculating the reference center, determining the distance threshold, and recognizing test samples. *TrM* takes the deep representations f of samples as input and outputs the modified deep representations $\tilde{f}$. Specifically, *TrM* adjusts the value of each dimension of the deep representation vector f based on the mean vector μ and standard deviation vector σ. If the value of the j th dimension of f, denoted as f^j, exceeds the interval range $\left[\mu^j - \lambda\sigma^j, \mu^j + \lambda\sigma^j\right]$, it is modified to the boundary value of that interval; otherwise, it remains. As described in Eq. (11), the processed f^j is represented as $\widetilde{f^j}$, where $j \in [1, d]$, and λ is a hyperparameter.

$$\widetilde{f^j} = \begin{cases} \mu^j - \lambda\sigma^j, f^j \le \mu^j - \lambda\sigma^j \\ f^j, \qquad\quad \mu^j - \lambda\sigma^j < f^j < \mu^j + \lambda\sigma^j \\ \mu^j + \lambda\sigma^j, f^j \ge \mu^j + \lambda\sigma^j \end{cases} \tag{11}$$

Step 3. For each known family $k \in F$ in the training set, find its reference center $\widetilde{c'_k}$ and distance threshold $\widetilde{\delta_k}$. Deep representations of all training samples $x_1, x_2, \ldots, x_N$ of family k in the training set are extracted using the feature extractor Z_{tran}, and these representations are corrected using *TrM*. The mean vector of the corrected deep representations is the reference center $\widetilde{c'_k}$:

$$\widetilde{c'_k} = \frac{1}{N}\sum_{i=1}^{N} TrM\left(Z_{tran}(x_i)\right) \tag{12}$$

The distance threshold $\widetilde{\delta_k}$ is the maximum distance from the modified deep representations of N training samples in family k to the reference center $\widetilde{c_{k'}}$:

$$\tilde{\delta}_k = \max_{i=1,\ldots,N} \left\| TrM\left(Z_{tran}(x_i)\right) - \tilde{c}'_k \right\|_2 \tag{13}$$

Step 4. For the test sample x^t, first use Z_{tran} to extract its deep representation. Then, the deep representation is fed into *TrM* for correction processing. Next, calculate the minimum distance between the corrected deep representation and the reference centers of K known families, obtaining the closest known family k. The calculation formulas are shown in Eqs. (14) and (15).

$$minDist(x_t) = \min_{j=1,\ldots,K} \| TrM\left(Z_{tran}(x^t)\right) - \widetilde{c'_j} \|_2 \tag{14}$$

$$k = \underset{j=1,\ldots,K}{argmin} \| TrM\left(Z_{tran}(x^t)\right) - \widetilde{c'_j} \|_2 \tag{15}$$

Step 5. In the deep representation space, if the distance from the input test sample x^t to the reference center $\widetilde{c_{k'}}$ of the nearest known family k is less than or equal to the distance threshold $\tilde{\delta}_k$ for that family, then x^t is classified to the known family k. Otherwise it is classified to the unknown family. The decision criterion is defined as:

$$decision = \begin{cases} ''unseen''\ family, & if\ minDist\left(x^t\right) > \tilde{\delta}_k \\ family\ ''k'', & if\ minDist\left(x^t\right) \le \tilde{\delta}_k \end{cases} \tag{16}$$

4 Experiment

4.1 Dataset

The BODMAS dataset, a public dataset for PE malware created by Yang et al. [22], comprises static feature vectors of 2381 dimensions from 57,293 malware samples across 581 families, with each sample's first appearance timestamp on VirusTotal [23] from August 2019 to September 2020. For our analysis, we transformed these vectors into 54x54 malware feature images. To ensure time consistency, Yang et al. divide samples at a cutoff date of January 10, 2020. Samples dated before this cutoff were assigned to the training set, and those after to the testing set. This split resulted in fewer than 30 families in the training set having more than 100 samples each. To ensure sufficient samples for training, only the top N families with the most samples were selected as known families, with N set at 5, 10, and 15. Table 1 details this time-consistent BODMAS, listing the number of known families N_{seen}, unknown families N_{unseen}, samples N_{sam}, and total families N_{fam}, with the degree of openness O calculated accordingly:

$$O = 1 - \sqrt{\frac{2 \times N_{seen}}{2 \times N_{seen} + N_{unseen}}} \tag{17}$$

4.2 Evaluation Metrics

MOSR faces two classification challenges: multi-classification among K known families, and binary classification to distinguish between known and unknown families. This study evaluates the performance using three metrics: $F1_K$, $AUROC$, and $F1_{K+1}$. The $AUROC$ represents the area under the ROC curve for the detection of unknown families, while the F1 score is the harmonic mean of the classification precision and recall. $F1_K$ score is the F1 score for closed set K. The $F1_{K+1}$ comprehensively evaluates the overall performance of open-set recognition for both known and unknown families, considering the unknown family as the $K + 1$ th family. It involves calculating the F1 score across all $K + 1$ families, including both known and unknown families:

$$F1_{K+1} = \frac{2 \times Recall_{K+1} \times Precision_{K+1}}{Recall_{K+1} + Precision_{K+1}} \tag{18}$$

Table 1. Time-consistent BODMAS.

N_{seen} : N_{unseen}	Set	Seen Family			Unseen Family		
		N_{sam}	N_{fam}	Time	N_{sam}	N_{fam}	Time
5:177 O = 76.88%	Train	7077	5	2019-08-29 ~2020-01-10			
	Test	1205	5	2020-01-10 ~2020-02-10	3163	177	2020-01-10 ~2020-02-10
10:172 O = 67.73%	Train	9522	10	2019-08-29 ~2020-01-10			
	Test	1771	10	2020-01-10 ~2020-02-10	2597	172	2020-01-10 ~2020-02-10
15:167 O = 60.98%	Train	11257	15	2019-08-29 ~2020-01-10			
	Test	2418	15	2020-01-10 ~2020-02-10	1950	167	2020-01-10 ~2020-02-10

4.3 Experimental Settings

The experiments in this paper were conducted on a server equipped with an RTX 3090 GPU, Ubuntu 18.04 OS, and 30 GB of RAM. The Swin Transformer uses the Swin-Tiny version. We obtained the model, pre-trained on the ImageNet-1K dataset, from the original authors' GitHub and fine-tuned it. Both the classification head and the linear probe are structured as a single fully connected layer, with their output dimensions corresponding to the number of known families in the training dataset.

4.4 Baselines

Liang [20]. Liang et al. proposed a closed-set classification method that uses linear probes to correct the cross-entropy loss for each training sample. However, it only considers the probability of a sample being predicted as a true family when calculating the regularization coefficients.

Zhu-MSP [21]. Zhu et al. developed a method for detecting out-of-distribution (OOD) images, which reinputs truncated deep representations into the classification head to compute the probabilities of a test sample across known categories. The maximum probability value (MSP) is used as the OOD score for calculating the $AUROC$ metric. Given its focus on OOD detection, this study compares the method solely in terms of the closed-set multi-classification metric $F1_K$ and the threshold-independent metric $AUROC$, excluding any comparison with the open-set classification metric $F1_{K+1}$.

Zhu-Energy [21]. The OOD detection method for images proposed by Zhu et al. This method requires re-feeding the truncated deep representations into the classification head. This method calculates the energy values of the test samples and uses them as the out-of-distribution scores to calculate the $AUROC$ metrics.

Guo [10]. A probabilistic-based method with generating additional data for MOSR proposed by Guo at el.

4.5 Experimental Results

Model Evaluation. The open set recognition performances of our method and baselines on the BODMAS dataset are shown in Table 2, with the best results in bold and the second best in underlined.

Table 2. Experimental Results.

$N_{seen} : N_{unseen}$	Method	$F1_K$	$AUROC$	$F1_{K+1}$
5:177 (O = 76.88%)	Liang [20]	0.8848	0.8962	0.6137
	Zhu-MSP [21]	0.8677	0.5636	/
	Zhu-Energy [21]	/	0.5992	/
	Guo [10]	0.8042	0.6837	0.5966
	LIFT(ours)	**0.8850**	**0.9250**	**0.6295**
10:172 (O = 67.73%)	Liang [20]	0.8824	0.8942	0.6925
	Zhu-MSP [21]	0.8901	0.8005	/
	Zhu-Energy [21]	/	0.7962	/
	Guo [10]	0.8196	0.7762	0.6787
	LIFT(ours)	**0.9002**	**0.9087**	**0.7238**
15:167 (O = 60.98%)	Liang [20]	**0.8633**	0.9010	0.6933
	Zhu-MSP [21]	0.8585	0.7516	/
	Zhu-Energy [21]	/	0.7527	/
	Guo [10]	0.8071	0.7798	0.6858
	LIFT(ours)	0.8631	**0.9045**	**0.6950**

Compared to Liang, our LIFT shows improvement across all three splits of the BODMAS dataset in both $F1_{K+1}$ and $AUROC$ metrics. While in $F1_K$ metric, LIFT is slightly lower in some splits, the difference is marginal, less than 0.0005. LIFT effectively leverages the discrepancy between the probability distributions from the classification head of main model and the linear probe. It utilizes the classification information from each family beyond merely the probability of the true family, thereby enhancing the performance of LIFT over baseline methods.

Compared to Zhu, our method has a significant advantage on $AUROC$ which can be attributed to the inherent distinctiveness of the processed deep representations. This distinctiveness enhances the separability between known and unknown family samples in the representation space. Our method directly uses processed representations for distance-based identification, which yields better results in detecting malware from unknown families compared to methods that compress valuable information by feeding truncated representations back to the classification head.

Compared to Guo, our method improves on all three metrics.

Ablation Experiments. The LIFT framework proposed in this paper encompasses two key components: a linear probe module designed to enhance the distinguishability of the deep representations; and a feature truncation module designed to make the values of the deep representations more typical. To assess the impact of these components on MOSR performance, we conduct ablation experiments on the 5:177 split of the BODMAS dataset, and the results are shown in Table 3.

Table 3 illustrates that using the Swin Transformer without any modifications yields the lowest performance. Using only the feature truncation module slightly improves $F1_K$ and $AUROC$, but greatly enhances $F1_{K+1}$. Similarly, the linear probe module alone slightly boosts $F1_K$ and significantly increases $AUROC$ and $F1_{K+1}$. Both the linear probe and feature truncation modules in LIFT are essential for improving MOSR task performance, combining these two modules further improves $F1_{K+1}$.

Table 3. Ablation Experimental Results.

Model	$F1_K$	$AUROC$	$F1_{K+1}$
LIFT	**0.8850**	0.9250	**0.6295**
Swin Transformer+Z_{prob}	0.8850	**0.9260**	0.6252
Swin Transformer+*TrM*	0.8850	0.8942	0.6263
Swin Transformer	0.8845	0.8938	0.6159

5 Conclusion

In this paper, LIFT, a discriminant classification approach of malware family on time-consistent open set, is proposed to solve the MOSR problem without generating additional data. By utilizing the self-attention mechanism of Swin Transformer, LIFT can capture both global and local information more effectively. Additionally, it employs linear probe and feature truncation to further refine the deep representation. The experiments indicate that LIFT can effectively address the challenge of malware open-set recognition in time-consistent dataset.

References

1. Malware statistics & trends report: Av-test. https://www.av-test.org/en/statistics/malware
2. Pendlebury, F., Pierazzi, F., Jordaney, R., Kinder, J., Cavallaro, L.:{TESSERACT}: eliminating experimental bias in malware classification across space and time. In: 28th USENIX Security Symposium (USENIX Security 19), pp. 729–746 (2019)

3. Cai, H.: Assessing and improving malware detection sustainability through app evolution studies. ACM Trans. Software Eng. Methodol. **29**(2), 1–28 (2020)
4. Chai, Y., Qiu, J., Yin, L., Zhang, L., Gupta, B.B., Tian, Z.: From data and model levels: Improve the performance of few-shot malware classification. IEEE Trans. Network Serv. Manage. **19**(4), 4248–4261 (2022)
5. Qiao, Y., Zhang, W., Du, X., Guizani, M.: Malware classification based on multilayer perception and word2vec for iot security. ACM Trans. Internet Technol. **22**(1), 1–22 (2021)
6. Wu, B., Xu, Y., Zou, F.: Malware classification by learning semantic and structural features of control flow graphs. In: 2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pp. 540–547. IEEE (2021)
7. Scheirer, W.J., de Rezende Rocha, A., Sapkota, A., Boult, T.E.: Toward open set recognition. IEEE Trans. Pattern Anal. Mach. Intell. **35**(7), 1757–1772 (2012)
8. Hassen, M., Chan, P.K.: Learning a neural-network-based representation for open set recognition. In: Proceedings of the 2020 SIAM International Conference on Data Mining, pp. 154–162. SIAM (2020)
9. Jia, J., Chan, P.K.: Mmf: a loss extension for feature learning in open set recognition. In: Farkaš, I., Masulli, P., Otte, S., Wermter, S. (eds) Artificial Neural Networks and Machine Learning – ICANN 2021. ICANN 2021. LNCS, vol. 12892. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-86340-1_26
10. Guo, J., Guo, S., Ma, S., Sun, Y., Xu, Y.: Conservative novelty synthesizing net-work for malware recognition in an open-set scenario. IEEE Trans. Neural Networks Learn. Syst. **34**(2), 662–676 (2021)
11. Jia, J., Chan, P.K.: Representation learning with function call graph transformations for malware open set recognition. In: 2022 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. IEEE (2022)
12. Bendale, A., Boult, T.E.: Towards open set deep networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1563–1572 (2016)
13. Oza, P., Patel, V.M.: C2ae: class conditioned auto-encoder for open-set recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2307–2316 (2019)
14. Kong, S., Ramanan, D.: Opengan: open-set recognition via open data generation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 813–822 (2021)
15. Yue, Z., Wang, T., Sun, Q., Hua, X.S., Zhang, H.: Counterfactual zero-shot and open-set visual recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 15404–15414 (2021)
16. Fu, X., Cai, H.: On the deterioration of learning-based malware detectors for an-droid. In: 2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), pp. 272–273. IEEE (2019)
17. Yang, L., et al.:{CADE}: Detecting and explaining concept drift samples for security applications. In: 30th USENIX Security Symposium (USENIX Security 21), pp. 2327–2344 (2021)
18. Yuan, C., Cai, J., Tian, D., Ma, R., Jia, X., Liu, W.: Towards time evolved malware identification using two-head neural network. J. Inform. Secur. Appl. **65**, 103098 (2022)
19. Asano, Y.M., Rupprecht, C., Vedaldi, A.: A critical analysis of self-supervision, or what we can learn from a single image. arXiv preprint arXiv:1904.13132 (2019)
20. Liang, Y., Zhu, L., Wang, X., Yang, Y.: A simple episodic linear probe improves visual recognition in the wild. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 9559–9569 (2022)
21. Zhu, Y., et al.: Boosting out-of-distribution detection with typical features. Adv. Neural. Inf. Process. Syst. **35**, 20758–20769 (2022)

22. Yang, L., Ciptadi, A., Laziuk, I., Ahmadzadeh, A., Wang, G.: Bodmas: An open dataset for learning based temporal analysis of pe malware. In: 2021 IEEE Security and Privacy Workshops (SPW), pp. 78–84. IEEE (2021)
23. VirusTotal. https://www.virustotal.com/gui/home/upload
24. Liu, Z., et al.: Swin transformer: hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 10012–10022 (2021)
25. GitHub-microsoft/Swin-Transformer. https://github.com/microsoft/Swin-Transformer