A Generalized Unknown Malware Classification

Nanda Rani, Ayushi Mishra, Rahul Kumar, Sarbajit Ghosh, Sandeep K. Shukla, and Priyanka Bagade(✉)

Department of Computer Science and Engineering, Indian Institute of Technology, Kanpur, Kanpur, India
{nandarani,ayushim,rahulkumar,sarbajitg,sandeeps,pbagade}@cse.iitk.ac.in

Abstract. Although state-of-the-art image-based malware classification models give the best performance, these models fail to consider real-world deployment challenges due to various reasons. We address three such problems through this work: limited dataset problems, imbalanced dataset problems, and lack of model generalizability. We employ a prototypical network-based few-shot learning method for a limited dataset problem and achieve 98.71% accuracy while training with only four malware samples of each class. To address the imbalanced dataset problem, we propose a class-weight technique to increase the weightage of minority classes during the training. The model performs well by improving precision and recall from 0% to close to 60% for the minority class. For the generalized model, we present a meta-learning-based approach and improve model performance from 48% to 72.06% accuracy. We report performances on five diverse datasets. The proposed solutions have the potential to set benchmark performance for their corresponding problem statements.

Keywords: Malware classification · Deep learning · Cyber Security · Malware

1 Introduction

The advancement in malware development leads to rapidly evolving malware families. Several image-based deep learning (DL) models, such as ResNet, MLP, LSTM, and GRU, are available that perform well for the corresponding datasets. The recent state-of-the-art models offer close to 99–100% accuracy on some of the datasets [1–3,7]. Although these models perform well, they lack with following capabilities, which we address through this research work:

1. Limited Dataset - The traditional deep learning model requires more data to learn significant features from given image sets. There will always be fewer samples for newly evolved malware classes. In such cases, traditional DL models fail to identify the patterns of the latest evolved malware families. Collecting more newly evolved malware samples to train the traditional model

F. Li et al. (Eds.): SecureComm 2022, LNICST 462, pp. 793–806, 2023.
https://doi.org/10.1007/978-3-031-25538-0_41

may cause a delay in recognizing malware family classes. Hence, there is a need for a model that can learn features from limited malware samples that are newly evolved.

2. Imbalanced Dataset - The datasets used to train various malware classification models [1,5,7] as well as publicly available datasets [1,3,7] are highly imbalanced. Imbalance data may lead to overfitting the model or may result in a biased model. Generating synthetic malware samples may not be a good option for dealing with the imbalanced dataset problem as the actual malware may differ in functionality. Hence, there is a need for a learning methodology that gives importance to classes with fewer samples in the dataset during the training phase to overcome overfitting.
3. Lack of model generalizability - It is always considered that training and testing datasets belong to the same distribution. However, there is a significant chance that unseen test set malware may belong to out-of-distribution from the training sample's distribution. Hence, there is a need for a generic model that can generalize the malware family class and identify patterns from unseen out-of-distribution malware samples. Based on our knowledge, there is no research work for generic malware classification models that can detect out-of-distribution samples.

We introduce three approaches based on advanced deep learning methods to address these problems. First, we implement a prototypical network-based few-shot learning model that can learn new sophisticated malware sample patterns by only seeing very few samples of each malware class. Second, we propose a class weight technique to provide weightage to minority classes and significantly improve the precision-recall of the minority class. Lastly, we employ meta-learning to propose generic models that can detect out-of-distribution malware samples. We utilize five malware datasets of different distributions and achieve better results. We verify the differences between distributions of multiple datasets by implementing out-of-distributions techniques proposed by Zaeemzadeh et al. [12]. Our contributions through this research work are the following:

1. We present a model based on few-shot learning that can learn patterns of newly evolved malware with very few samples.
2. We propose a class weight-based model that can balance the imbalance effect in the malware dataset during training.
3. We introduce a generic model that can classify out-of-distribution samples belonging to the training set's malware class.

We discuss related research work for image-based malware classification in Sect. 2. Section 3 explains the background of malware analysis techniques and how it evolves. We elaborated the implemented methodology to address mentioned research gaps in Sect. 4. Section 5 discusses the experiment implementation and the results. Further, we conclude the contribution and future scope of this research in Sect. 6.

2 Related Work

Recently, image-based malware classification has gained popularity over traditional malware classification methods, i.e., static and dynamic malware analysis [2,6]. However, the textural differences between malware images make DL models suitable for identifying malware patterns. Many research experiments are available which implement deep learning models on diverse datasets and exploring several possible feature sets for image-based malware classification. This section discusses a summary of recently proposed malware classification methods.

Natraj et al. [6] propose the first idea to visualize the malware binaries and classify malware based on images. They released the very first image-based dataset named Malimg publicly. Microsoft released malware samples in one of the Kaggle challenges for image-based malware classification [14]. Both datasets, Malimg and Microsoft, have a total of 9,339 and 20,860 samples, respectively.

Singh et al. [1] demonstrate a novel method for malware classification by implementing a pre-trained ResNet50 model and achieved an accuracy of 99.40%. However, they mentioned that their model gives low accuracy for previously unseen malware [1]. Bozkir et al. [5] recognize malware from RGB images generated from a memory dump of the suspicious process by utilizing GIST and HOG features. Among all implemented learning models, the best accuracy is 96.39%. They perform manual feature extraction, which may not be a good option when deploying a model in real time. Similarly, Dhavalle et al. [2] extract features from the images, including energy, entropy, contrast, dissimilarity, homogeneity, and correlation and build various machine learning classifiers. The random forest model achieves the highest accuracy of 96.7%. Bhodia et al. [7] implement several variants of ResNet to identify malware patterns. Even though the authors use several DL models to identify malware, none of them is a generic model which can detect unseen malware. In [3], the authors present diverse deep learning models, including MLP, CNN, LSTM and GRU, to utilize image-based features and opcode features to train the models. VGG19 and ResNet152 perform well with 92% accuracy, but their model is not validated for unknown malware variants. Kim et al. [4] employs a hybrid deep generative model which exploits local and global features of malware images and uses variational autoencoders. The model doesn't validate generalizability with diverse datasets [4].

Zhu et al. [10] implement a Siamese Neural Network-based few-shot learning method to detect malware with few samples, detecting ransomware attacks with few samples. However, their work is limited to ransomware malware class only. Matching networks and prototypical network-based methods for malware classification are introduced by Tran et al. [19]. They claimed performance on a very limited number of experiments and matching network-based model performs poor for Malimg dataset as well as both proposed model performs poor for Microsoft dataset. All of the above research works primarily focus on model performance accuracy and overlook class-wise precision and recall. Also, there is not much work on malware classification with limited malware samples and work which discusses model generalizability for malware classifications. Therefore, we address these research problems in this paper.

3 Background

Malware is a malicious code that aims to conduct various destructive operations without the victim's consent and awareness. Attackers introduce several malware families; some notorious families are Trojans, Backdoors, Ransomware, and Worms. With the newly emerging cyber threats, malware is growing exponentially.

As malware evolves rapidly, It is necessary to have intelligent algorithms to identify malware patterns effectively. Primarily, two main analysis techniques are there for malware analysis: static and dynamic. Static analysis can identify malicious patterns without running the executables, and by observing the code structures [17]. However, this method is ineffective against obfuscated malware.

Dynamic code analysis executes the code in a sandbox environment to reveal the runtime behavior [18]. Although it is better than static analysis, dynamic analysis consumes more time and resources. Also, some sophisticated malware may not demonstrate malicious behaviour while running in the sandbox. On the other hand, image-based malware classification provides a novel way to classify the malware without executing the code and performing manual code investigation. The image-based classification techniques provide an emerging exploration method to classify malware with respect to texture variations between malware classes. The texture difference between malware families is shown in Fig. 1.

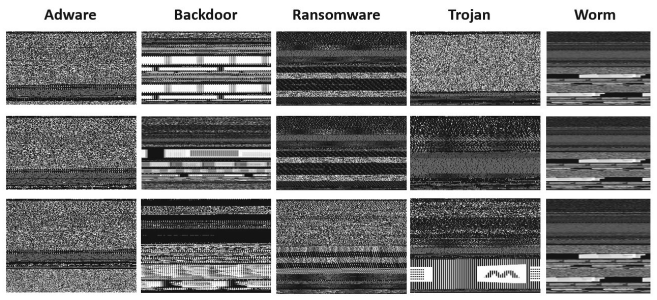

Fig. 1. Malware Families representation in Grayscale images

4 Proposed Methodology

4.1 Dataset

The Malimg [6] and Microsoft challenge datasets [14] are the only publicly available. These datasets contain old malware sample variants that may not contain recent malware patterns. Therefore, we create our dataset by collecting recent malware samples from MalwareBazaar[1]. MalwareBazaar is a platform that aims to share malware samples with the cyber community for study purposes. Our

[1] https://bazaar.abuse.ch/.

dataset contains 11,273 malware samples belonging to different malware classes, including ransomware, worm, trojan, adware and backdoor, with 813, 4012, 3926, 426 and 2096 samples, respectively.

We generate grayscale images by converting the executable samples from our dataset. As executable binaries are sequences of 0's and 1's, we use this sequence to group consecutive 8 bits and calculate their corresponding decimal value. The sequence of decimal values is stored in an array and considered as a pixel value for generating the grayscale image for executables. The generated images are visually distinguishable between different malware families (Fig. 1). We set the width of the image size to 256 and let the length be flexible according to the size of executables [1]. We explain all these steps in Algorithm 1.

Algorithm 1. Executable to grey-scale image conversion

```
0: procedure BINARY_TO_IMAGE($file$) {Convert binary executable to grayscale image}
1: Width = 256
2: Length = File_Size/Width
3: Calculate decimal values for each consecutive 8 bits.
4: Create a 2-D array by fixing width as 256.
5: Convert 2-D array into grayscale images.
```

We obtain datasets from Singh et al. [1] and Prajapati et al. [3]. We use five datasets (Maling [6], Microsoft [14], Singh et al. [1], Prajapati et al. [3] and our dataset) to perform experiments. We address the research gaps discussed in Sect. 1 by introducing few-shot learning for limited dataset problems (Sect. 4.2), class weight method for imbalanced dataset problems (Sect. 4.3) and meta-learning for model generalizability (Sect. 4.4).

4.2 Limited Dataset

We present a prototypical network-based few-shot learning model to address the limited dataset problem. The prototypical network is based on the idea that there is an embedding in which several points cluster around a single prototype representation for each class, present in Fig. 2. The goal is to learn per-class prototypes using feature space sample averaging. In this implementation, we assume 'm' labelled datasets as a support set, where 'm' is as small as one or two samples. The support set is represented as S = $(x_1, y_1), ..., (x_m, y_m)$, in which $x_i \in \mathbb{R}^D$ is D-dimensional feature vector and $y_i \in 1, ..., n$ is corresponding labels. We can denote the support set as S_n, represented as data samples with class l. The prototype for each class is represented as C_n, where $C_n \in \mathbb{R}^m$, an embedding function $f_\theta : \mathbb{R}^D \rightarrow \mathbb{R}^m$ having ϕ as the learnable parameter. Each prototype represents the average of the embedded support points in its class [13].

$$C_n = \frac{1}{|S_n|} \sum_{(x_i, y_i) \epsilon S_n} f_\phi(X_i) \tag{1}$$

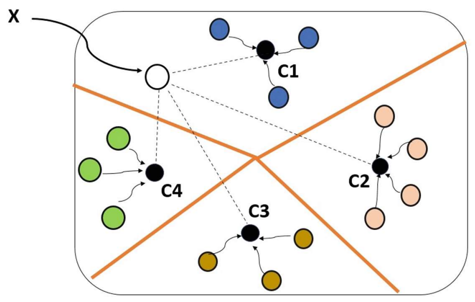

Fig. 2. Prototypical networks [13]

The learning phase of Prototypical Networks is to perform by minimizing the negative log-probability $J(\phi)$, often known as log-softmax loss [13].

$$J(\phi) = -log(p_\phi(y = n|x)) \text{ for true class k}$$

where,

$$p_\phi(y = n|x) = \frac{exp(-d(f_\phi(x), c_n))}{\Sigma_{n'} exp(-d(f_\phi(x), c_{n'}}$$

Here d represents euclidean distance.

The key benefit of utilizing a logarithm is that the loss is drastically increased when model fails to predict the correct class labels. The query images classify by measuring the distance between each unlabeled image and prototypes using Euclidean distance. After computing distances, we apply softmax to obtain the probability of belonging to each class, then assign a class based on greater the probability, shorter the distance.

4.3 Imbalanced Dataset

We employ class weight techniques to balance the imbalance effect of data samples during training. We assign low and high-weight values to the majority and minority classes. We calculate the weight for each class by using the following formula:

$$w_i = \frac{n\_samples}{(n\_classes * n\_samplesi)} \tag{2}$$

where,
w_i is the weight for each class (i represents the class)
$n\_samples$ is the total number of samples in the dataset
$n\_classes$ is the total number of unique classes
$n\_samplesi$ is the total number of samples of the respective class

The aim is to give the minority class more weight throughout the training phase. We implement a weighted log loss function to achieve the same.

$$log\_loss = \frac{1}{N}\sum_{i=1}^{N}[-(\omega_0(y_i * log(\hat{y_i})) + \omega_1((1 - y_i) * log(1 - \hat{y_i})))] \tag{3}$$

where,
ω_0 is the class weight for zero class.
ω_1 is the class weight for one class.

The goal of the objective function in Eq. 3 is to penalize the minority class for misclassification by assigning them a higher class weight while the majority class gets a lower weight. We implement this technique with the ResNet50 model using the weight and loss function shown in Eq. 3.

4.4 Model Generalization

We present a meta-learning-based model generalization method to build a generic model. The aim is to build a generic model which can identify malware patterns in the dataset whose distribution was not present in the training dataset. To the best of our knowledge, there are no generic models available that can get trained on one distribution dataset and detect samples of datasets with different distributions. All state-of-the-art model training and testing perform on the same distribution, i.e., the same dataset only [1,3,6,7]. We had a total of five datasets, and before proceeding with generalization, we first needed to verify whether the distribution of all five datasets was different or not. Therefore, we verify the distribution differentiation by implementing out-of-distribution detection.

Out of Distribution Detection. In real-world scenarios, test samples may not contain the data from train samples. Therefore, detecting the out-of-distribution (OOD) samples that do not belong to the training classes is desirable. We implement the union of 1-dimensional spaces [12] to perform the OOD detection.

OOD samples can get recognized more robustly and with a greater probability if the feature vectors of the training samples are in a 1-dimensional subspace [12]. Given a training dataset of N-sample label pairings from L available classes, the goal is to train a neural network to detect out-of-distribution samples during testing (not belonging to any known L classes).

Making the distribution of known classes as compact as possible reduces the risk of error. The error probability is given by $p_e = \sqrt{p_l p_o}\exp(-\mathscr{B})$, where $\mathscr{B}$ is the Bhattacharyya distance [15] and is given by:

$$\mathscr{B} = \frac{1}{8}\Delta^T(\frac{\Sigma_l + \Sigma_o}{2})^{-1}\Delta + \frac{1}{2}\log\frac{(\det\frac{\Sigma_l+\Sigma_o}{2})}{\sqrt{\det\Sigma_l \det\Sigma_o}} \quad (4)$$

$$\Delta = (\mu_l - \mu_o)$$

The distance $\mathscr{B}$ is a combination of the Mahalanobis distance and a measure of the compactness of the distributions [12]. We distinguish OOD and IN distributions during test time using the cosine similarities between the test samples and the singular vector corresponding to each class. The cosine similarity is:

$$\cos(\theta_{ln}) = \frac{w_l^T x_n}{||w_l||\,||x_n||} \quad (5)$$

OOD test perform by computing the angular minimum distance of test feature vector x_n, which is given by

$$\phi_n = \min_l \arccos(\frac{|x_n^T v_1^{(l)}|}{||x_n||}) \tag{6}$$

Meta Learning Based Model Generalization. We implement a meta-learning-based model generalization method. For generalization, Li et al. [11] present a novel meta-learning approach to synthesize virtual training and test domain for validation during training for achieving generalization, which we follow for this work. Our model generalization method provides a model agnostic training method that enhances the domain generalizability of a base learner. We use ResNet18 as a base learner for this work. We divide a set of different distribution datasets in a train, val and test set so that all have different distributions dataset. We train the base learner on a train set distribution and, at the same time, validate on a validation set distribution(which is different from the training distribution). The meta-optimization goal is to reduce the loss on the training distributions while simultaneously ensuring that the direction is taken to accomplish this also improves the validation loss (having a different distribution than the train set). Once a model learns to minimize the loss on the validation set (different distribution), we can evaluate performance on the test set (having novel different distribution data). We illustrate the model generalization learning flow in Fig. 3 (Different symbols represent different distributions of data samples).

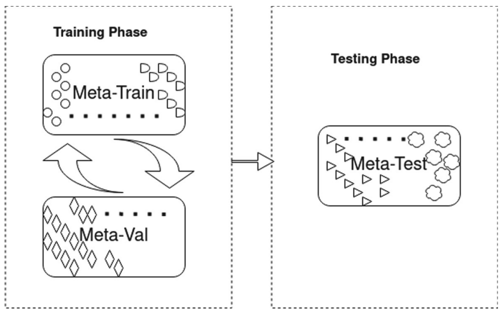

Fig. 3. Model generalization model learning method

We assume there are total n_d different distributions of malware datasets in the training set. The test dataset belongs to different distribution, which is not present in the training set. The malware class for all distribution datasets must be the same to compare generalizability. Therefore, we compare the malware class of all available datasets and found three common malware families in all five datasets. These are trojan class, worm class and backdoor class. Hence we validate our implementation with the three-class classification. The steps follow to implement the generic model are following:

- We define a base model, ResNet18, parameterized as θ and hyperparameters α, β, γ. to solve the malware classification task. Where, α is meta step size, β is meta val beta and γ is decay rate.
- For each iteration, we follow the below steps:
  - Split the n_d distribution into $\hat{n_d}$ and $\bar{n_d}$.
  - For meta-train we calculate gradients

    $$\nabla_\theta = \mathcal{F}^{'}(\hat{n_d}; \theta)$$

    where $\mathcal{F}$ is the cross-entropy loss function.
  - We update the model parameter with $\theta^{'} = \theta - \alpha \nabla_\theta$
  - We calculate cross entropy loss by using $\bar{n_d}$ and $\theta^{'} i.e., \mathcal{G}(\bar{n_d}; \theta^{'})$
  - For meta-optimization we update the model by using

    $$\theta = \theta - \gamma \frac{\partial(\mathcal{F}(\hat{n_d}; \theta) + \beta \mathcal{G}(\bar{n_d}; \theta - \alpha \nabla_\theta))}{\partial \theta}$$

We minimize the following objective function for this algorithm.:

$$\mathcal{F}(\theta) + \beta \mathcal{G}(\theta - \alpha \mathcal{F}^{'}(\theta)) \tag{7}$$

5 Experiment and Results Discussion

5.1 Limited Dataset Results

We implement few-shot learning on various datasets available, and the model achieves best performance considerably by seeing very few malware samples. Malimg, Microsoft, Singh et al. [6] and our dataset has seven, six, eight and five classes, respectively. Hence we run them for the number of different class way n shots, where n can be small, i.e. no newly evolved samples available. The accuracy achieved by model on the datasets is present in Table 1.

Table 1. Result of limited dataset problem

Our dataset	Acc (%)	Malimg	Acc (%)	Microsoft	Acc (%)	Singh et al. [1]	Acc (%)
5-way 1-shot	53.85	6-way 1-shot	86.96	7-way 1-shot	91.24	8-way 1-shot	63.04
5-way 2-shot	53.57	6-way 2-shot	87.12	7-way 2-shot	90.43	8-way 2-shot	**71.59**
5-way 4-shot	56.86	6-way 4-shot	95	7-way 4-shot	**98.71**	8-way 4-shot	70
5-way 8-shot	**65.85**	6-way 8-shot	**96.88**	7-way 8-shot	95.50	8-way 8-shot	69.53

5.2 Imbalanced Dataset Results

We implement a class weight-based methodology to resolve the imbalance dataset problem. To assess the model performance, we need to verify whether class-wise precision-recall of minority classes is improving or not, as accuracy may not always be a suitable indicator of model performance with an imbalanced dataset. The implemented method improves minority class performance significantly. Hence, Improvement in minority class performance in terms of precision and recall is present in Tables 2, 3, 4 and 5.

Table 2. Result of imbalanced problem on Microsoft Dataset

	Without proposed method		With proposed method	
Class name	Precision	Recall	Precision	Recall
Adware	0.97	0.98	0.94	0.99
Backdoor	**0.00**	**0.00**	**0.25**	**0.60**
Virtool	**0.85**	0.91	**0.86**	0.91
Worm	**0.96**	0.88	**0.97**	0.94
Trojan	**0.87**	0.84	**0.95**	0.76
Botnet	1.00	1.00	0.99	1.00

Table 3. Result of imbalanced problem on Our Dataset

	Without proposed method		With proposed method	
Class name	Precision	Recall	Precision	Recall
Adware	0.86	0.60	0.85	0.73
Backdoor	0.94	**0.89**	0.87	**0.91**
Ransomware	0.80	0.79	0.80	0.79
Worm	**0.00**	**0.00**	**0.32**	**0.50**
Trojan	**0.46**	0.60	**0.67**	0.58

Table 4. Result of Imbalanced problem on Singh et al. [1] Dataset

	Without proposed method		With proposed method	
Class name	Precision	Recall	Precision	Recall
Trojan Dropper	1.0	1.0	1.0	0.99
Virus	0.94	0.77	0.84	0.77
Trojan Downloader	0.89	0.88	0.79	0.93
Backdoor	0.84	0.76	0.83	0.62
Virtool	**0.00**	**0.00**	**0.59**	**0.40**
Worm	1.0	1.00	0.99	1.00
Trojan	0.78	0.89	0.75	0.92
Rogue	0.00	0.00	0.00	0.00

Table 5. Result of imbalanced problem on Malimg Dataset

	Without proposed method		With proposed method	
Class name	Precision	Recall	Precision	Recall
Trojan Downloader	0.98	**0.95**	0.92	**0.98**
Backdoor	1.00	1.00	1.00	1.00
Worm	1.00	1.00	1.00	1.00
Trojan	**0.95**	0.97	**0.99**	0.91
PWS	1.00	0.99	1.00	0.99
Dialer	0.99	1.00	0.99	1.00
Rogue	0.98	1.00	0.98	1.00

5.3 Model Generalization Results

We first verify whether the available dataset is in different distribution or not before moving towards generic model implementation.

Out of Distribution Results. We implement an OOD detection method to test the distribution difference between datasets by measuring FPR@95TPR and AUC (Area under the ROC Curve). FPR@95TPR is the probability that a negative (out-of-distribution) example gets miscategorized as positive (in-distribution) when the true positive rate (TPR) is as high as 95%. The lower value for FPR@95TPR and higher value for AUC proves the in-data and out-data taken belongs to different distributions. We verified the distribution difference by implementing the method explained in Sect. 4.4 and obtained FPR@95TPR value as 0.2573 and AUC value as 0.93616. The lower value of FPR@95TPR and higher value of AUC proves that the taken datasets are of different distributions.

Meta Learning for Domain Generalization Results. To create a baseline for comparing proposed method performance, we evaluate model performance without implementing the proposed meta-learning approach. We implement a commonly used ResNet50 neural network and train on our dataset and Microsoft dataset, and test it on the Malimg dataset and receive only 48% accuracy. Which is not preferable and needs to have a generic model which can perform better than the baseline. Hence, we trained the meta-learning model on the set of Our dataset and Microsoft dataset, and tested on Malimg dataset. Initially, the model achieved accuracy of 68.21%, which is an improvement form the baseline, but we experiment more to improve the performance. Next, in order to increase the diversity of the training set, we train it with the set consists our dataset, Microsoft dataset, and Singh et al. [1] dataset and test on Malimg dataset. This time model improves the accuracy to 69.83%. Finally, we use all five datasets and selected our dataset, Microsoft dataset, Singh et al. [1], and Prajapati et al. [3] dataset for training and Malimg dataset for testing. This time the model improved the accuracy significantly and achieved 72.06%. We can see a high jump compared to the baseline ResNet model from 48% to 72.06% by adding a meta-learning approach on top of the base learner. The result of implemented meta-learning proves that by adding a more diverse distribution dataset in the training phase, the model is learning unknown dimensions of data and generalizing more towards pattern detection for unseen and different distribution malware samples. We summarize the performance of meta-learning for generic models in Table 6.

Table 6. Result of model generalization

Cases number	Training set	Testing set	Acc.
Case 1 (Baseline)	Microsoft	Malimg Dataset	**48%**
Case 2	Microsoft and Our dataset	Malimg Dataset	68.21%
Case 3	Microsoft, our dataset and Singh et al. [1]	Malimg Dataset	69.83%
Case 4	Microsoft, our dataset, Singh et al. [1] and Prajapati et al. [3]	Malimg Dataset	**72.06%**

6 Conclusions and Future Scope

This research addresses the critical gaps present in the current malware classification using image representation. The key contribution of this research work is to handle limited and imbalanced dataset problems and present a robust generic malware classification model. Our work focuses on implementing image-based malware classification techniques rather than static and dynamic analysis. This research highlights the malware classification model generalization, which has the potential to provide a new research dimension in malware classification research. As a part of the future work, Hyperparameter tuning can be one of the possible modifications to improve model generalization algorithm performance. The proposed class weight-based technique significantly improves the precision and recall of minority classes and effectively balances the imbalance data effect. The presented model for the limited dataset problem can also further improve by using a zero-shot deep learning technique. In a nutshell, this research work has the strength to serve as a foundation for several future studies on the imbalance and limited malware dataset problem and enhancement of the malware classification model's generalization ability.

Acknowledgement. We thank to the C3i (Cyber Security and Cyber Security for Cyber-Physical Systems) Innovation Hub for partially funding this research project.

References

1. Singh, A., Handa, A., Kumar, N., Shukla, S.K.: Malware classification using image representation. In: Dolev, S., Hendler, D., Lodha, S., Yung, M. (eds.) CSCML 2019. LNCS, vol. 11527, pp. 75–92. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-20951-3_6
2. Dhavlle, A., Shukla, S.: A novel malware detection mechanism based on features extracted from converted malware binary images, ArXiv, vol. abs/2104.06652 (2021)
3. Prajapati, P., Stamp, M.: An empirical analysis of image-based learning techniques for malware classification. In: Stamp, M., Alazab, M., Shalaginov, A. (eds.) Malware Analysis Using Artificial Intelligence and Deep Learning. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-62582-5_16
4. Kim, J.Y., Cho, S.B.: Obfuscated malware detection using deep generative model based on global/local features. Comput. Secur. **112**, 102501 (2022). ISSN 0167-4048. https://doi.org/10.1016/j.cose.2021.102501
5. Bozkir, A., Tahillioglu, E., Aydos, M., Kara, I.: Catch them alive: a malware detection approach through memory forensics, manifold learning and computer vision. Comput. Secur. **103**, 04 (2021)
6. Nataraj, L., Karthikeyan, S., Jacob, G., Manjunath, B.S.: Malware images: visualization and automatic classification. In: Proceedings of the 8th International Symposium on Visualization for Cyber Security, VizSec 2011. Association for Computing Machinery, New York (2011)
7. Bhodia, N., Prajapati, P., Di Troia, F., Stamp, M.: Transfer learning for image-based malware classification. In: Proceedings of the 5th International Conference on Information Systems Security and Privacy (2019)
8. Vasan, D., Alazab, M., Wassan, S., Safaei, B., Zheng, Q.: Image-based malware classification using ensemble of cnn architectures (imcec). Comput. Secur. **92**, 101748 (2020)
9. Lu, Y., Li, J.: Generative adversarial network for improving deep learning based malware classification. In: 2019 Winter Simulation Conference (WSC), pp. 584–593 (2019)
10. Zhu, J., Jang-Jaccard, J., Singh, A., Welch, I., AI-Sahaf, H., Camtepe, S.: A few-shot meta-learning based siamese neural network using entropy features for ransomware classification. Comput. Secur. **117**, 102691 (2022)
11. Li, D., Yang, Y., Song, Y.-Z., Hospedales, T.M.: Learning to generalize: meta-learning for domain generalization (2017)
12. Zaeemzadeh, A., Bisagno, N., Sambugaro, Z., Conci, N., Rahnavard, N., Shah, M.: Out-of-distribution detection using union of 1-dimensional subspaces. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 9452–9461 (2021)
13. Snell, J., Swersky, K., Zemel, R.S.: Prototypical networks for few-shot learning (2017)
14. Ronen, R., Radu, M., Feuerstein, C., Yom-Tov, E., Ahmadi, M.: Microsoft malware classification challenge (2018)
15. Bhattacharyya, A.: On a measure of divergence between two statistical populations defined by their probability distributions. Bull. Calcutta Math. Soc. **35**, 99–109 (1943)
16. Tran, T.K., Sato, H., Kubo, M.: One-shot learning approach for unknown malware classification. In: 2018 5th Asian Conference on Defense Technology (ACDT), pp. 8–13 (2018). https://doi.org/10.1109/ACDT.2018.8593203

17. Chen, L.: Understanding the efficacy, reliability and resiliency of computer vision techniques for malware detection and future research directions (2019)
18. Saurabh, A.M., Static, A.U., Methodology, D.: International Conference on Advanced Computation and Telecommunication (ICACAT) 2018, pp. 1–5 (2018). https://doi.org/10.1109/ICACAT.2018.8933769
19. Tran, T.K., Sato, H., Kubo, M.: Image-based unknown malware classification with few-shot learning models. In: Seventh International Symposium on Computing and Networking Workshops (CANDARW) 2019, pp. 401–407 (2019). https://doi.org/10.1109/CANDARW.2019.00075