IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 26 March 2024, accepted 11 April 2024, date of publication 22 April 2024, date of current version 1 May 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3392251

RESEARCH ARTICLE

MalSSL—Self-Supervised Learning for Accurate and Label-Efficient Malware Classification

SETIA JULI IRZAL ISMAIL[1,2], HENDRAWAN[1], (Member, IEEE), BUDI RAHARDJO[1], (Member, IEEE), TUTUN JUHANA[1], (Member, IEEE), AND YASUO MUSASHI[3], (Member, IEEE)

[1]School of Electrical Engineering and Informatics, Institut Teknologi Bandung, Bandung, West Java 40132, Indonesia
[2]School of Applied Science, Telkom University, Bandung, West Java 40257, Indonesia
[3]Research and Education Institute for Semiconductors and Informatics, Kumamoto University, Kumamoto 860-0862, Japan

Corresponding authors: Hendrawan (hend@itb.ac.id) and Setia Juli Irzal Ismail (33220007@mahasiswa.itb.ac.id)

This work was supported in part by the Institut Teknologi Bandung (ITB) International Research Program [Grant number 37/IT1.B07.1/SPP-LPPM/I/2024] and in part by Japanese Student Service Organization (JASSO) Scholarship.

ABSTRACT Malware classification with supervised learning requires a large dataset, which needs an expensive and time-consuming labeling process. In this paper, we explore the efficacy of self-supervised learning techniques for malware classification. We propose MalSSL, a self-supervised learning-based method utilizing image representation to classify malware. MalSSL classifies unlabeled malware images using contrastive learning and data augmentation. The model is initially trained on an unlabeled Imagenette dataset as a pretext task and subsequently retrained on an unlabeled malware dataset in downstream tasks. Two downstream tasks were employed to evaluate the system: 1) malware family classification and 2) malware benign classification. The obtained results include an accuracy of 98.4% for the malware family classification experiment on the Malimg dataset and an accuracy of 96.2% for the malware and benign dataset (Maldeb dataset). Our findings suggest that the proposed system accurately classifies malware without the need for labeled data, displaying higher accuracy compared to other self-supervised methods. This research not only contributes to advancing the state-of-the-art in malware classification but also underscores the potential of self-supervised learning methods as a viable solution for addressing the dynamic landscape of malware threats.

INDEX TERMS Image representation, malware, malware classification, self-supervised learning.

I. INTRODUCTION

Malware poses a significant threat to the Internet, with an average of 588 malware attacks occurring every minute, as reported by the antimalware company McAfee [1]. WannaCry ransomware infected 200,000 computers across 150 countries within just 3 days in 2017, resulting in millions of dollars in losses [2]. Apart from causing economic impacts, the Mirai botnet paralyzed internet networks in Europe and North America [3], whereas the Stuxnet malware successfully sabotaged the Natanz nuclear installation in Iran [4].

Traditionally, antivirus solutions have relied on signature-based and heuristic-based detection techniques [5]. Although effective, these methods involve manual compilation of malware signatures and heuristic rules by skilled malware analysts, demanding both time and specialized expertise [6]. In response to the growing volume of malware, an automated malware detection process using machine learning has been implemented [7].

Despite the advantages, the integration of machine learning in malware detection faces two key challenges. Firstly, it necessitates a large dataset [5]. Secondly, the dataset labeling process is time-consuming [8]. The largest labeled malware dataset to date is Ember, comprising 1.1 million samples [9]. However, this remains relatively small compared to the staggering 1.2 trillion malware samples reported by AV-Test [10].

One of the challenges in implementing machine learning for malware detection is the expensive dataset labeling process [11]. The process of labeling malware datasets begins with malware analysis. For known malware, the analysis process can be carried out using tools like Virustotal [12], with

The associate editor coordinating the review of this manuscript and approving it for publication was Kathiravan Srinivasan.

a maximum analysis time of 1 hour per sample. However, for new malware, manual analysis becomes imperative. This process demands time and expertise from malware analysts. The complexity of the malware affects the time needed to perform malware analysis. Manual malware analysis can span between 4 hours to 6 weeks per sample [13]. Hence, if we have 1000 malware samples, it will take quite a while to carry out the dataset labeling process.

In response to this challenge, we propose MalSSL, a self-supervised learning (SSL)-based malware detection system with image representation, to classify malware without the need to label the dataset. Leveraging self-supervised learning, MalSSL processes unlabeled data to learn representations and construct a model [14]. MalSSL involves contrastive learning, and the resulting model can be effectively utilized in downstream tasks for malware classification. Our contributions are:

1) We introduce MalSSL, an SSL-based malware detection system with image representation. MalSSL achieves precise malware classification without dependency on labeled data. MalSSL is capable of accurately classifying malware using a single GPU (Graphics Processing Unit) with a high accuracy of 98.4%.
2) We have developed models for classifying malware using advanced SSL methods such as MoCo (Momentum Contrast), SimCLR (Simple Framework for Contrastive Learning of Visual Representation), SimSiam (Simple Siamese), and SwAV (Swapping Assignments between Views). These models serve as benchmarks for evaluating the performance of MalSSL.
3) We curated a malware and benign dataset named Maldeb. Maldeb serves as an evaluation platform for malware-benign classification tasks. MalSSL achieves 96.2% accuracy in classifying the Maldeb dataset.

This paper is structured as follows: Section II presents related works on malware detection and SSL. Section III describes our proposed method. Section IV explains the results of our experiments. Section V consists of a discussion, and Section VI summarizes the conclusions.

II. RELATED WORKS

A. MALWARE DETECTION WITH IMAGE REPRESENTATION

Nataraj et al. [15] attempted to transform malware samples into a grayscale image representation. The binary sequence of the sample is organized into an 8-bit vector and then converted to decimal. Subsequently, the decimal value is translated into grayscale, where 0 corresponds to black and 255 to white. Machine learning is then employed to classify the malware family based on the similarity of the visual pattern of the image with k-nearest neighbors (k-NN).

Another methodology involves examining texture and intensity features, utilizing the support vector machine (SVM) algorithm [16]. Luo and Lo [17] explored the use of local binary pattern features in images trained with SVM and k-NN. Kalash et al. [18] used a convolutional neural network (CNN) for image classification with global image descriptor (GIST) features.

Not only grayscale, but representations in the form of RGB (Red, Green, and Blue) also have been investigated using Random Forest, k-NN, and SVM algorithms [19]. A CNN classification algorithm for RGB images has also been developed by [20]. Verma et al. [21] utilized binary textures from grayscale images with GLCM (Grey Level Co-occurrence Matrix) for multiclass malware classification. Makandar and Patrot proposed the Gabor wavelet as a feature with the SVM algorithm [22]. Nisa et al. used fused SFTA (Segmentation-based Fractal Texture Analysis) with DCNN (Deep CNN) [23].

Guo et al. introduced the Malware Entropy Sequences Reflect the Family (MESRF) with the discrete wavelet decomposition algorithm [24]. Bensaoud and Kalita applied a multi-task learning approach with PReLU (Parametric Rectifier Linear Unit) to detect obfuscation methods [25]. In our work, we adopt GIST image representation, differing from all previous approaches by utilizing unlabeled datasets.

Aslan and Yilmaz proposes the integration of two pre-trained network models, ResNet-50 and AlexNet. The features obtained from these models were combined to generate a feature vector of 4096 dimensions, which was then passed through the SoftMax layer and fully connected layers for normalization [26].

Al-Khater and Al-Madeed addresses the problem of imbalanced and inadequate malware datasets using the Fast and Adaptive Bi-dimensional Empirical Mode Decomposition (FABEMD) technique [27]. FABEMD extracts different intrinsic mode function (IMF) images to increase the training dataset.

AlGarni et al. proposes the use of transfer learning with pre-trained EfficientNet models on the ImageNet dataset [28]. Alam et al. introduces SREMIC: Spatial Relation Extraction-based Malware Image Classification. They extract spatial relations as features from images, utilize image augmentation, and propose a spatial convolutional network to classify malware [29]. Mitsuhashi and Shinagawa studied 120 different deep learning models with 5 levels of fine-tuning parameters to classify malware image representation and concluded that EfficientNetB4 finetuned by freezing had the best performance [30].

B. SELF-SUPERVISED LEARNING

The self-supervised learning method can be divided into two stages [31]. The initial stage is the pretext task, wherein the dataset's representation of the dataset is studied from the unlabeled dataset, and subsequently, a model is constructed. This model is designed to recognize the relationship between the data and remain resilient to nuisance factors. The model is then deployed in the downstream task stage, which in this study is to classify malware.

In the Natural Language Processing (NLP) domain, the SSL method has demonstrated successful implementation. Pretext tasks involve training models such as BERT

(Bidirectional Encoder Representations from Transformers) [32], RoBERTa (Robustly Optimized BERT Approach) [33], and XLM-R (Cross-lingual Language Model-Robust) [34] on unlabeled datasets. The resultant model is then applied to downstream tasks. For instance, in NLP, the SSL model can proficiently complete some missing words in a sentence.

In the field of computer vision, numerous studies have explored the application of the SSL method. The SSL methods for image representation can be categorized into two main groups: 1) similarity maximization and 2) redundancy reduction [31]. In similarity maximization, input images undergo augmentation through two different data augmentation techniques. These augmented images are then fed into the encoder, and the similarity between the two inputs is calculated [35]. The network is trained to maximize this similarity and produce a feature model that effectively represents the images. On the other hand, the redundancy reduction method involves calculating the cross-correlation matrix of the two embeddings. Subsequently, the matrices of the two embeddings are optimized to be as close as possible to the identity matrix [36].

Similarity maximization can be divided into three distinct approaches: 1) contrastive learning; 2) Clustering; and 3) Distillation. Contrastive learning aims to learn the dataset feature from the embedding of the corresponding image [37]. In this method, embeddings from related images (positive) should be closer than embeddings from unrelated images (negative). The primary objective of contrastive learning is to bring together positive embeddings while pushing apart negative embeddings. However, a notable challenge of contrastive learning is preventing a trivial solution, which occurs when the system produces the same feature for all input images.

Several implementations of contrastive learning include PIRL (Pretext-Invariant Representation Learning) [38], SimCLR [39], and MoCo [40]. PIRL aims to achieve invariance over data augmentations rather than predicting data augmentation [38]. SimCLR used a contrastive loss to maximize agreement between different augmented views [39]. MoCo utilizes a memory bank and employs two forward passes to prevent trivial solutions [40]. In our work, we modified SimCLR and MoCo and implemented them in the context of malware classification problems.

Clustering takes a different approach to grouping samples than contrastive learning, creating groups in the feature space [41]. SwAV is a clustering method that uses equipartition constraint and soft assignment to prevent a trivial solution [42]. AVID-CMA (Audio Visual Instance Discrimination with Cross-Modal Agreement) combines contrastive learning and clustering techniques [43].

The distillation method involves naming the neural network as the student-teacher network. It prevents trivial solutions by employing an asymmetric learning rule and an asymmetric architecture between the student and teacher [44]. BYOL (Bootstrap Your Own Latent) is a distillation technique with an additional predictor on the student network [44]. SimSiam employs the same set of weights between the student and teacher networks [45]. Barlow Twins implements the efficient coding hypothesis and measures the cross-correlation matrix between the outputs of two identical networks [46].

Recent studies have explored SSL applications across various domains, including biomedicine. Del Pup and Atzori conducted a comprehensive survey on the applications of SSL to biomedical signals, highlighting its potential for extracting meaningful representations from diverse biomedical data sources [47]. This is particularly relevant considering the inherent challenge faced by both the biomedical and cybersecurity domains in acquiring labeled data at scale, owing to factors such as privacy concerns, data scarcity, and the need for domain expertise. As such, leveraging SSL techniques becomes imperative for effectively learning from unlabeled data in scenarios where labeled data is limited, a challenge shared by both biomedical research and malware classification.

C. SELF-SUPERVISED LEARNING ON MALWARE CLASSIFICATION

Dib et al. proposed EVOLIoT, a self-supervised contrastive learning framework for detecting and characterizing IoT malware variants [48]. Our work differs from EVOLIot in two ways: 1) EVOLIoT focuses on IoT (Internet of Things) malware, and 2) they extract features from assembly instructions and utilize pre-trained language models BERT. Seneviratne et al. presented Sherlock, a self-supervised model with a Vision Transformer architecture [49]. Sherlock focuses on Android malware using the Vision Transformer architecture. In contrast, our work is centered around PE (Portable Executables) malware.

III. MATERIAL AND METHOD

A. DESIGN

The approach employed in this study is to design a malware classification system with a self-supervised learning approach named MalSSL. There are two stages in our proposed system, namely the pretext task and the downstream task, as illustrated in Fig. 1.

In the downstream task stage, the model is further trained to recognize the image representations of malware families using the Malimg dataset [15]. Once the training process is completed, the system is tested to classify unknown input files into malware families. Additionally, we trained and tested the system with another downstream task, classifying inputs into malware and benign classes using the Maldeb Dataset [50].

B. DATASET

In this paper, we utilize the Imagenette [51], *Malimg* [15], and Maldeb Dataset [50]. Imagenette is a subset of the larger ImageNet dataset [52]. ImageNet consists of millions of labeled images across thousands of categories. In contrast,

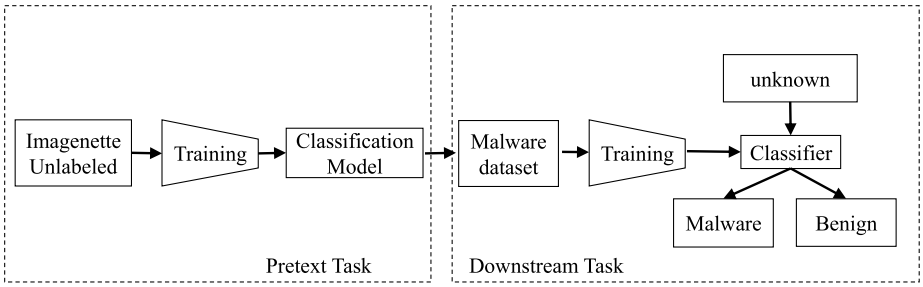

FIGURE 1. Proposed model of MalSSL: In the pretext task, MalSSL is trained on the unlabeled Imagenette dataset; the resulting model is then further trained on a malware dataset and tested for malware classification.

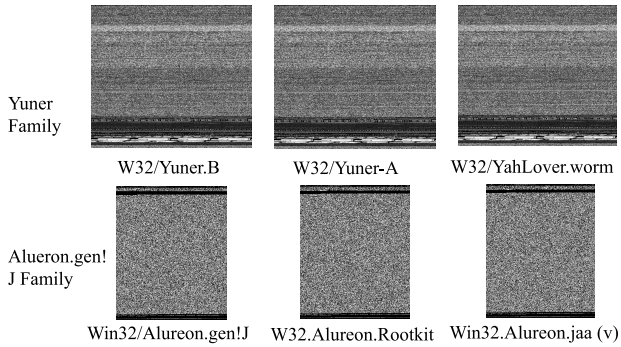

FIGURE 2. Malware family image representation of six different malware samples belonging to the malware families Yuner and Alueron.gen!J. Each family has a similar visual representation.

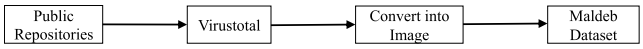

FIGURE 3. Maldeb dataset collection process: samples were collected from various malware repositories, validated using Virustotal, and then converted into images.

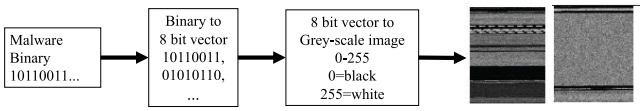

FIGURE 4. Conversion of malware binary to image: Malware binary is grouped into an 8-bit vector. Each vector is converted into grayscale.

Imagenette is a curated subset consisting of ten classes, for faster image classification models. We use the 160-pixel version of the Imagenette dataset. Imagenette is chosen for more efficient SSL training with a single GPU.

The Malimg dataset [15] is a collection of malware images representing 25 malware family classes. A malware family is a group of malwares that share similar program codes. Malware within the same family is considered a variant of a single malware. Often, malware authors share or sell their code on the dark web [53], leading to the development of malware variants by other parties.

The Malimg dataset is widely employed for constructing a malware classification system. Within the Malimg dataset, image representations of malware within the same class exhibit visual similarity, as depicted in Fig. 2.

The image representation of six distinct malware samples from two malware families, Yuner and Alueron.gen!J, illustrates this visual similarity. Each malware family comprises three different samples with comparable visual representations.

We collected our dataset, the Maldeb dataset, specifically for the testing phase [50]. This dataset differs from Malimg as it comprises only two classes: malware and benign. The purpose of gathering the Maldeb dataset is to evaluate the system with newer malware samples, considering that Malimg was published around 2011. We intend to test the system not only for classifying malware families but also for distinguishing between malware and benign samples.

The datasets were collected from several malware repositories, including TekDefense [54], TheZoo [55], The-Malware-Repo [56], Malware Database [57] and Malware Bazaar [58]. The benign samples were collected from Microsoft Windows 10 and 11 system apps and several open-source software repositories, including CNET [59], Sourceforge [60], Fileforum [61], and PortableFreeware [62]. The process of collecting is detailed in Fig. 3. Validation of the collected samples was performed by scanning them using the Virustotal malware scan service [63]. Samples that received validation confirming their status as malware were included in the dataset and categorized under the malware class.

The samples underwent pre-processing by converting the malware binary into grayscale images. For benign samples (not malware), they were grouped under the benign class. After validation, we classified 20,854 samples into two classes: malware (10,427 samples) and benign (10,427 samples). The conversion of benign samples into images follows the method proposed by Nataraj et al. [15], as depicted in Fig. 4.

Initially, the binary data is organized into an 8-bit vector. These vectors are then converted into grayscale within a range of [0-255], with 0 representing black and 255 representing white pixels. The image dimensions vary according to the Nataraj et al. method [15]. Depending on the size of the benign sample, the width of the image representation ranges from 32 to 1,024. Samples of the Maldeb dataset can be seen in Fig. 5.

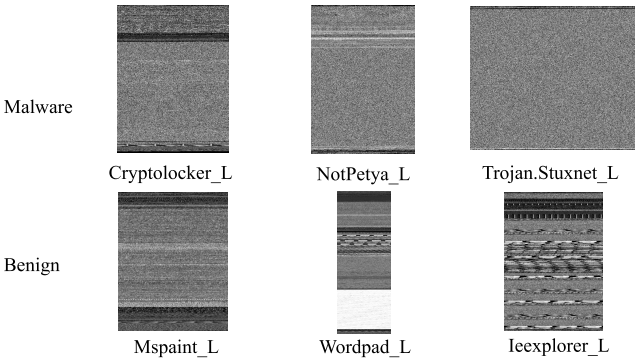

FIGURE 5. Maldeb dataset samples. The Maldeb dataset consists of two classes: malware and benign. Each class has a broad diversity of image representations.

TABLE 1. Malimg and maldeb dataset comparison.

Dataset Name	Malimg Dataset	Maldeb Dataset
Classes	Malware Family	Malware and Benign
Number of Classes	25	2
Number of samples	9548	1905 (1033 Malware; 872 benign)

Table. 1 present a comparison between the Malimg dataset and the Maldeb dataset. Notably, Maldeb does not share any common samples with the Malimg dataset.

C. PRETEXT-TASK

The pretext task aimed to construct a classifier model for use in the downstream task. In this phase, machine learning was trained on the unlabeled Imagenette dataset, which initially underwent random crop augmentation. The purpose of augmentation was to train the system to recognize image representations under various transformations.

Each input underwent two augmentations to create two corresponding images. The unlabeled augmented data were then trained on a dual Siamese Convolution Network (encoder) ResNet-18 [64]. The resulting encoder produced a general representation. A non-linear projection from the image representation was computed using a fully connected network in the projection head, which is a Multi-Layer Perceptron (MLP). The projection head helps aid the network in identifying invariant features and recognizing different transformations of the same image.

For the contrastive learning task, a contrastive loss function was implemented. The loss function for the two images on different networks was calculated, and a stochastic gradient descent calculation was performed to update the Convolutional Neural Network (CNN) and MLP. The goal was to minimize the loss function. In this process, the network learned to identify correlated images (from the same image groups) and distinguish them from non-correlated images. The outcome was a model capable of recognizing images by maximizing the level of similarity between augmented images.

From Fig. 6, the input x is augmented into two different images, x_i and x_j, which form a positive pair (correlated). A negative pair consists of images that are not from the same image (not correlated). A neural network base encoder $f(\cdot)$, extracts the representation vector from augmented images to obtain $h_i = f(x_i) = ResNet(x_i)$, where h_i is the output from the average pooling layer. A neural network projection head $g(\cdot)$, maps the image representation to space.

Then we added a multi-layer perceptron and one hidden layer, to obtain $z_i = g(h_i) = W^{(2)}\sigma(W^{(1)}h_i)$, where σ is a rectified linear unit (ReLU). For a given set x_k, where x_i and x_j are also part of it, the loss function calculation is:

$$\ell_{i,j} = -\log \frac{\exp(\frac{(z_i \cdot z_j)}{\tau})}{\sum_{k=1}^{N} \exp(\frac{(z_i \cdot z_k)}{\tau})} \tag{1}$$

With the following parameters:

- i, j = augmented samples
- k = positive + negative samples
- τ = temperature parameter = 0.5
- N = Total Samples

and

$$z_i = g(h_i) = W^{(2)}\sigma(W^{(1)}h_i) \tag{2}$$

The structure of the model follows the ResNet-18 architecture, which consists of 18 layers [64]. We modify the standard first layer of the 7×7 kernel to 3×3 with no stride and no MaxPool2d to make the model faster. We did not split the dataset for the pretext task; instead, we utilized the entire unlabeled dataset for training. The training was carried out on a PC server with the following specifications: Intel Core i9-11900K x 16; GPU NVIDIA (RTX3060 10GB); Disk 3TB; Operating System Ubuntu 23.10 LTS. The average training time required for the pretext task is three hours. The resulting model is deployed to perform downstream tasks.

D. DOWNSTREAM TASK

The trained model from the pretext task process, is employed for malware classification training using the transfer learning method. From the pretext task model in Fig. 6, we remove the projection head and extract the representation for the downstream task. The projection head is only used to calculate the contrastive loss in the pretext task. In the downstream task stage, the model is trained to classify malware families from the Malimg dataset. The downstream task employs a similar method to the pretext task, with the distinction of employing a classification head instead of a projection head, as illustrated in Fig. 7. First, the input Malimg dataset is augmented with a random crop, as illustrated in Fig. 8.

In the downstream task, we replace the projection head with a classification head. The classification head encoded the representation (features) from the encoder network into specific categories, such as malware and benign or malware families. The classification head is a simple linear layer, followed by the SoftMax activation function.

We conducted two different downstream tasks to adapt to two distinct malware classification machine learning tasks: a) malware family classification with the Malimg dataset and

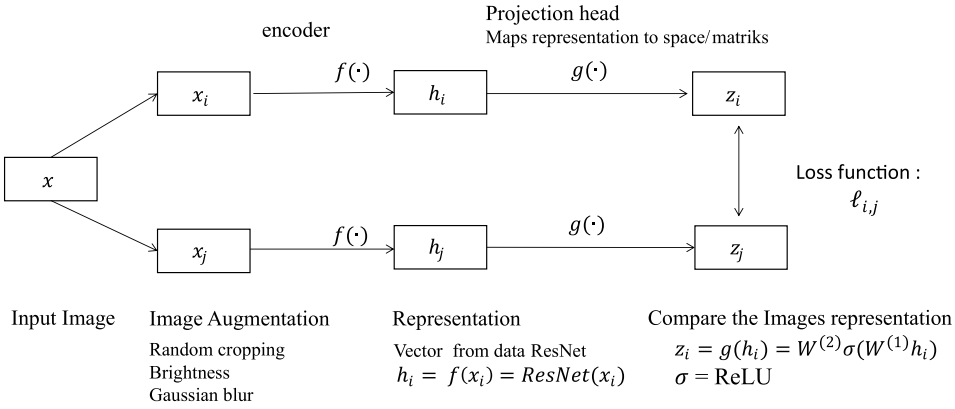

FIGURE 6. Illustration of the pretext task: The unlabeled dataset is augmented twice and trained on a dual-Siamese convolution network. The encoder maps the augmented images into a high-dimensional feature space. A projection head then maps the high-dimensional feature vectors into a projection space. The contrastive loss is computed, and the model minimizes this loss to learn the feature representations of the input images.

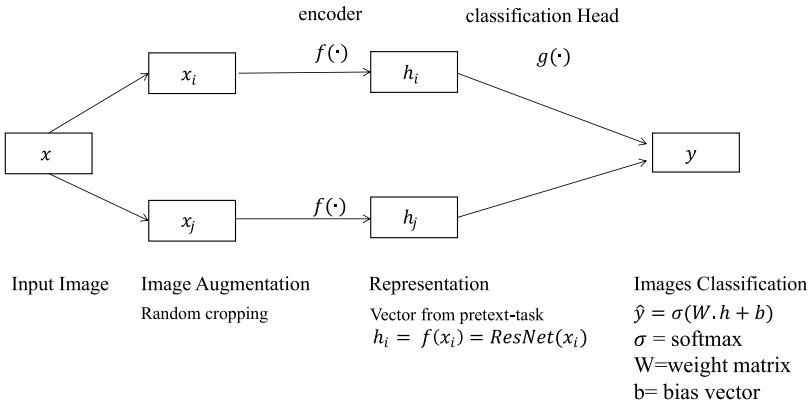

FIGURE 7. The illustration of the downstream task. The model from the pretext task is trained with a malware dataset and classified into malware benign or malware families.

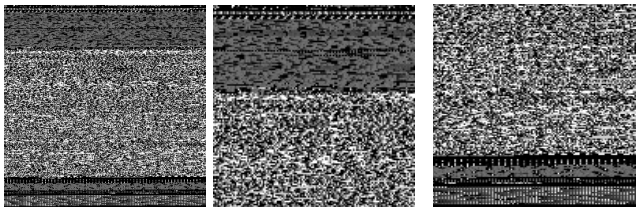

FIGURE 8. The augmentation example of image representation from malware twain32L. From left to right: 1) original image representation of twain_32_L; 2-3) random crop example.

b) malware benign classification with the Maldeb dataset. In each experiment, we modify the classification head according to the number of classes in the classification task. In the first experiment, the system is tasked with classifying the input into 25 classes of malware families. We adjust the classification head into 25 classes. In the second experiment, malware benign classification, the classification head is adjusted into two classes.

After the downstream task had been successfully executed, a malware classification test was conducted. We performed several tests with different batch sizes (8, 16, 32, 64, 128, 256, and 512) and different epoch sizes (10, 50, 100, 200, and 300). We split the dataset into three sets randomly: training 80%, validation 10%, and testing 10%. The system is built using the PyTorch Lightly platform [65]. The following hyperparameters were used for training: a) optimizer: stochastic gradient descent (SGD); b) learning rate: 0.06; c) momentum: 0.9; d) regularization: weight decay = 0.0005; e) loss function: cross-entropy loss; f) model architecture: ResNet-18; g) learning rate schedule: cosine annealing; h) metrics: accuracy, false positive rate, and false negative rate. We share the code and dataset from our experiment on the GitHub repository [66].

E. COMPARISON WITH OTHER SSL METHOD

We have conducted an evaluation of MalSSL using four advanced self-supervised learning techniques for images, namely MoCo [40], SimCLR [39], SimSiam [45], and SwAV [42]. The reason why these four methods were chosen is that they share a common objective with MalSSL, which is to maximize similarity between the original image and the

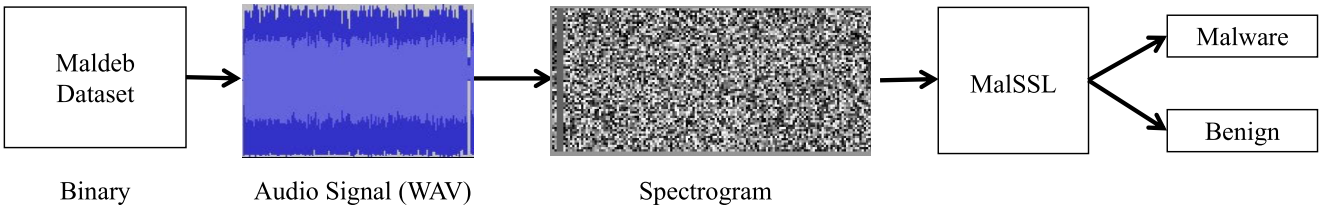

FIGURE 9. Testing MalSSL with audio representation. Maldeb datasets were preprocessed into audio representations and Spectrograms, trained with MalSSL, and tested to classify malware and benign.

augmented one. We have studied MoCo, SimCLR, SimSiam, and SwAV methods and developed four models for malware classification. The models were developed using PyTorch and Lightly [65]. Each model underwent hyperparameter fine-tuning to achieve the best possible results. Our malware classification models might differ from the original works as we have adapted them to a single GPU environment.

MoCo [40], short for Momentum Contrast, is a contrastive method that utilizes a memory bank to maintain activity momentum. This method employs two encoders. The main advantage of MoCo is that it does not require storing the entire dataset, making memory usage management easier. However, MoCo has some drawbacks, such as requiring two forward passes and additional memory to store parameters or features. MalSSL and MoCo are two different technologies. The main distinction between them is that MalSSL does not require a memory bank, whereas MoCo does.

SimCLR [39] is a method that learns a general representation by maximizing the similarity between transformed views of the same image and minimizing the similarity of different images. MalSSL differs from SimCLR in the augmentation techniques used as well as the simpler ResNet architecture of ResNet-18. MalSSL does not use color jitter augmentation, as SimCLR does. We employ a simpler contrastive loss calculation adapted from MoCo.

The Swapping Assignments between Views (SwAV) algorithm is a clustering method that operates online. Its main objective is to increase the similarity between an image and its augmentation. This ensures that both the original image and its augmentation are placed in the same cluster. The main difference between SwAV and MalSSL is that SwAV uses the Sinkhorn-Knopp algorithm to calculate the similarity of embeddings.

SimSiam uses a distillation approach with the concept of the student-teacher network. The architecture used for the student and the teacher is asymmetrical. In this approach, the embedding of the student is calculated for the original image, and the embedding of the teacher is calculated for the augmented image. The similarity of the two embeddings is forced. Additionally, a prediction head is added to the student network. On the other hand, MalSSL uses a symmetric network and does not add a prediction head.

F. TESTING WITH AUDIO REPRESENTATION

We are expanding our approach to include different types of malware representations beyond image-based formats. Diversifying the representations used in our approach could offer valuable insights and potentially enhance the robustness and effectiveness of our model. Previously, we studied the possibility of utilizing the BERT (Bidirectional Encoder Representations) language model, SSL, and text representation to classify malware [67]. In this section, we are conducting experiments with audio representations, specifically using spectrograms, as illustrated in Fig. 9. Spectrograms provide a visual representation of the frequency content of audio signals over time and have been successfully utilized in various machine-learning tasks, including audio classification.

The dataset used in this experiment is the binary form of the Maldeb dataset. To enable audio-based analysis, we preprocessed the dataset by converting the malware and benign binaries into an audio signal. The audio signal was then converted to spectrogram representations to capture the frequency content of the audio data over time. Subsequently, we employed MalSSL to classify the spectrograms as malware or benign.

The malware and benign samples were converted to audio signals with PCM (pulse code modulation) and the following parameters: sample rate 44100 Hz, sample size 16-bit, mono channel, without compression, and WAV file format. The spectrogram representations were generated following the Mel Spectrogram method with the following parameters: sample rate = 16000 Hz, STFT (short-time Fourier transform), n_fft = 400, hop length = 160, and n_mels = 128. The MalSSL model was trained using a batch size of 256, a learning rate of 0.06, and 300 epochs.

IV. RESULT

A. PERFORMANCE OF MALSSL

We conducted a comprehensive set of experiments on a diverse malware dataset to evaluate the effectiveness of the proposed MalSSL methods for malware classification. The primary evaluation metric used in this study is accuracy, reflecting the models' ability to correctly classify malware samples into their respective families in the malware family classification task, or the models' ability to correctly classify samples into malware and benign in the malware-benign classification task. We conducted tests on the Malimg dataset to classify malware families, and the results are presented in Fig. 10.

During our experiments, we systematically varied the batch sizes for training the MalSSL model, spanning a range from 8 to 512. This exploration aimed to discern the impact

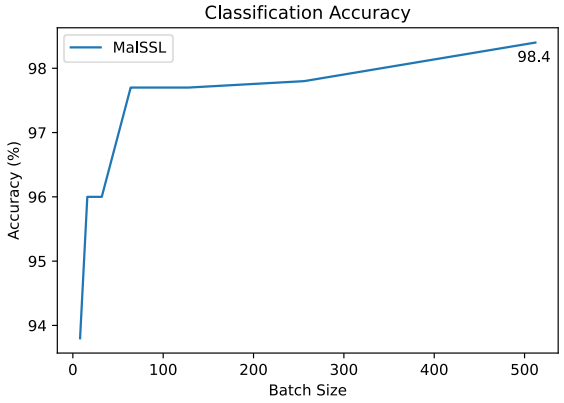

FIGURE 10. MalSSL accuracy and batch size performance in classifying the Malimg dataset. The highest accuracy of 98.4% is achieved with a batch size of 512.

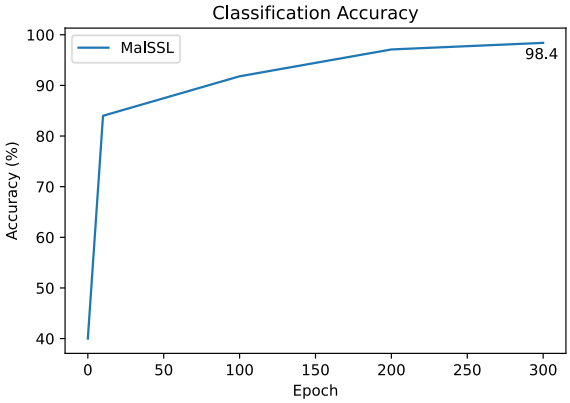

FIGURE 11. MalSSL performance accuracy at different epochs. The highest accuracy is 98.4% at epoch 300.

of batch size on the accuracy of the self-supervised learning approach for malware classification. We observed a positive correlation between batch size and classification accuracy. As the batch size increased from 8 to 512, the accuracy also increased, indicating a consistent improvement in the model's performance with larger batches. To comprehensively explore the impact of training duration on the performance of our MalSSL model, experiments were conducted by varying the number of epochs from 10 to 300. This investigation aimed to discern the relationship between training duration and the model's ability to capture intricate patterns within the unlabeled malware dataset. The results are presented in Fig. 11.

The highest accuracy of 98.4% was attained at epoch 300. This finding suggests that extended training duration fosters a more comprehensive understanding of the inherent characteristics of malware, resulting in good classification performance. The best result of our experimentation occurred at a batch size of 512, where the MalSSL model achieved an accuracy of 98.4%. The observed trend prompts a closer examination of the dynamics associated with varying batch sizes. Larger batch sizes likely facilitate a more comprehensive exploration of the feature space, enabling the model to discern intricate patterns in the unlabeled malware data.

TABLE 2. MalSSL performance compared with other SSL methods on malware family classification with unlabeled malimg dataset.

Method	Accuracy (%)	FPR (%)	FNR (%)	Training time (minutes)
MoCo	61.5	10.9	27.1	12
SimCLR	89.9	2.6	7.4	13
SWaV	86.3	5.8	7.9	20
SimSiam	97.3	1.0	1.6	42
MalSSL	98.4	0.6	0.9	39

B. MALWARE FAMILY CLASSIFICATION RESULTS

In Table. 2, we present a comprehensive comparison of various self-supervised (SSL) methods applied to the task of malware family classification with the Malimg dataset.

We build malware classification models with different SSL methods, including MoCo, SimCLR, SwAV, SimSiam, and MalSSL. The comparison is based on classification accuracy and training time running on the same machine. The training time in question is the duration required to perform training on downstream tasks. MalSSL outperforms other methods with an accuracy of 98.4% in a training time of 39 minutes. MoCo exhibits the lowest accuracy at 61.5%. SimSiam achieves a high accuracy of 97.3% but at the cost of a longer training time of 42 minutes. MalSSL demonstrates its ability to capture patterns within unlabeled malware datasets and its efficiency in training time. MalSSL has produced a low false positive rate (FPR) of 0.6% and a false negative rate of 0.9%. A low FPR and FNR rate in these experiments indicates that MalSSL accurately classifies malware samples into their respective families with minimal errors.

C. MALWARE BENIGN CLASSIFICATION RESULTS

We present the results of our experiments comparing different self-supervised (SSL) methods for the task of malware-benign classification in Table. 3. We build malware classification models with MoCo, SimCLR, SwAV, SimSiam, and our proposed MalSSL. The evaluation metrics are accuracy and training time, and the models were tested using the unlabeled Maldeb dataset.

Malware-benign classification poses a significant challenge compared to malware family classification. This increased difficulty is attributed to the diversity and variability within the malware and benign classes, as we can see in Fig. 5. Unlike the distinct and well-defined patterns found in malware families, benign samples cover a wide spectrum of legitimate software, making it more challenging to distinguish differences.

The challenge in malware-benign classification is reflected in the varied accuracies observed across SSL methods. Whereas SimCLR and SimSiam exhibit notable accuracies of 90.8% and 88.5%, respectively, MoCo and SwAV encounter difficulties, achieving accuracies of 69.2% and 74.3%, respectively. MalSSL attains the highest accuracy of 96.2% with a training time of 35 minutes. In contrast, MoCo requires an extended training time of 73 minutes, whereas SimSiam achieves a notably shorter training time of

TABLE 3. MalSSL accuracy and training time compared with other SSL methods on unlabeled Maldeb dataset.

Method	Accuracy (%)	FPR (%)	FNR (%)	Training time (minutes)
MoCo	69.2	14.1	16.7	73
SimCLR	90.8	3.9	5.2	41
SWaV	74.3	12.5	13.2	39
SimSiam	88.5	3.7	7.6	20
MalSSL	96.2	1.4	2.2	35

TABLE 4. MalSSL K-Fold cross-validation testing result.

Dataset	Average Accuracy (%)	Standard Deviation (%)
Malimg	98.1	1.3
Maldeb	95.5	2.6

20 minutes. MalSSL achieved a low false positive rate (FPR) of 1.4% and a false negative rate of 2.2%. In these experiments, the false positive rate (FPR) represents the percentage of benign files incorrectly identified as malware by MalSSL. The false negative rate (FNR) shows the proportion of actual malware samples mistakenly labeled as benign. A low FPR and FNR rate signify that MalSSL can effectively distinguish between malware and benign with minimal error.

D. K-FOLD CROSS-VALIDATION TESTING

We evaluated the performance of MalSSL using k-fold cross-validation (k = 5). The dataset was split into training (80%) and testing (20%) in each fold. The model was trained on the training set and evaluated on the testing set. This process was repeated five times, and the results were averaged.

Table. 4 displays the outcome of the K-fold cross-validation testing. MalSSL attained an average accuracy of 95.5% on the Maldeb dataset and 98.1% on the Malimg dataset. These findings reveal a high accuracy and minimal standard deviation, indicating that MalSSL demonstrates strong performance and consistency across folds.

E. AUGMENTATION EXPERIMENTS

In this section, we present the results of experiments conducted to evaluate the impact of data augmentation on the performance of our malware classification model. Specifically, we compare the classification accuracy achieved with and without data augmentation on the Malimg and Maldeb datasets. For the experiments with data augmentation, we employed random cropping as the primary augmentation technique. Random cropping was applied to the input images during training to introduce variability and improve the robustness of the classifier. The results of our experiments are summarized in Table. 5.

As shown in Table. 5, the classification accuracy achieved with data augmentation is substantially higher compared to the results obtained without augmentation. Specifically, the model trained with augmentation achieves an accuracy of 98.4% in the Malimg dataset and 96.2% in the Maldeb dataset. In contrast, the model trained without augmentation

TABLE 5. MalSSL classification accuracy comparison with and without data augmentation (%).

Dataset	With Augmentation	Without Augmentation
Malimg	98.4	79.3
Maldeb	96.2	74.7

TABLE 6. MalSSL classification result of Maldeb dataset audio representation.

Epoch	Batch Size	Accuracy (%)	With Augmentation?	Time (m)
100	256	85.9	v	45
100	256	81.2	x	39
300	512	82.7	v	58
300	512	81.6	x	47

exhibits a drastic drop in accuracy, achieving only 79.3% in the Malimg dataset and 74.7% in the Maldeb dataset. The results demonstrate the critical role of data augmentation in enhancing the performance of our malware classification model. Augmentation techniques such as random cropping enable the model to learn more robust and generalizable features from malware image representation.

F. AUDIO REPRESENTATION EXPERIMENTS

The classification of malware-benign samples using audio representation with MalSSL yielded promising results. The results of the classification experiment are summarized in Table. 6.

We conducted experiments using varying numbers of epochs (100, 300) and batch sizes (256, 512), with and without augmentation. The highest accuracy achieved is 85.9% with 100 epochs, a batch size of 256, and with augmentation. These results suggest that augmentation may not significantly impact the classification of malware audio representations. Despite the better performance of image representation in classification compared to audio representation, our results indicate the potential of MalSSL in classifying the audio representation of malware and benign samples.

V. DISCUSSION

Our proposed method, MalSSL, has a different approach compared to state-of-the-art malware detection systems. Most papers on malware detection systems with image representation utilize supervised learning with labeled datasets. In Table. 7, we compare our results with the state-of-the-art malware family classification system on the Malimg dataset. Table. 7 reveals that our proposed method (98.4%) outperforms the original work by Nataraj et al. (97.2%) [15]. We are not far from the best performance method from Guo et al. (99.9%) [24]. The detailed approach of supervised methods listed in Table. 7 is explained in Section II.

The Maldeb dataset is a new dataset that consists of newer malware samples than Malimg, which was published in 2011. In Table. 8, we compare our proposed method's performance on the Maldeb dataset with the experiment conducted by our lab member Khairul for his thesis project [69]

TABLE 7. Accuracy comparison on the Malimg dataset.

Method	Algorithm/Feature	Labeled	Accuracy (%)
[15]	kNN	v	97.2
[18]	M-CNN	v	98.2
[20]	IMFCN	v	98.8
[22]	Gabor Wavelett+SVM	v	98.9
[23]	SFTA+DCNN	v	99.3
[24]	MESRF	v	99.9
[25]	MTL+PReLU	v	97.8
[26]	SVM+hybrid feature	v	97.8
[27]	FABEMD	v	99.6
[28]	CNN+Transfer Learning	v	99.9
[29]	SREMIC	v	99.8
[30]	EfficientNetB4	v	98.9
[68]	GIST+Deep Learning	v	98.0
MalSSL	SSL	x	98.4

TABLE 8. Accuracy comparison on the Maldeb dataset.

Method	Algorithm/Feature	labeled	Accuracy (%)
[69]	CNN	v	95.0
LinearSVC	SVM	v	90.2
MalSSL	SSL	x	96.2

and our experiment with LinearSVC. The results showed that MalSSL could detect the Maldeb dataset with good performance. This experiment proved that MalSSL can be employed on two different malware classification tasks: 1) malware family classification and 2) malware benign classification.

Malware-benign classification, characterized by a more diverse malware-benign class, presents different challenges compared to malware family classification. MalSSL's consistent performance across these tasks underscores its versatility and capacity in different malware classification scenarios.

The observed trade-offs between accuracy and training time (from Tables. 2 - 4) underscore the importance of balancing efficiency and performance in practical deployment scenarios. MalSSL achieves high accuracy (over 90%) on three different malware classification scenarios with an efficient training time, making it a practical and resource-efficient solution for real-world deployment scenarios.

We opt for contrastive learning as it can learn the representation of data points by discovering their similarity. Our proposed model reduces the dependency on large volumes of labeled data traditionally required for supervised learning approaches. Whereas access to larger datasets could enhance the model's capability to classify a wider range of malware and improve overall performance, the primary advantage of our approach lies in its ability to effectively utilize unlabeled data for training.

While acknowledging the diverse conditions and training datasets across different SSL methods, it is important to highlight that MalSSL demonstrated competitive performance despite being executed on a single GPU. For example, SimCLR utilized 128-core TPUs [37], MoCo was trained on 64 GPUs [38], SwAV on 4 GPUs [40], and SimSiam was trained on 8 GPUs. Despite these variations' conditions, our comparison highlights the efficiency of MalSSL in achieving strong performance with relatively modest computational resources, indicating its potential practical ability. The possible cause is that we utilize a smaller architecture, ResNet-18 compared to the ResNet-50 architecture of SimCLR, MoCo, SwAV, and SimSiam.

A test conducted with malware audio representations demonstrates the potential of implementing MalSSL with various other malware representations, which warrants further investigation. Present Antivirus solutions utilize various machine learning methodologies, including Random Forest [70], Support Vector Machines [71], Decision Tree [72], LSTM [73], Deep Learning [74], [75], Clustering [76], [77], and Ensemble learning [78], among others. The implementation of MalSSL has the potential to advance Antivirus technology by reducing the need for huge, labeled datasets. MalSSL could be deployed in the real world by integrating into existing antivirus systems or as an additional layer of protection alongside antivirus solutions. In the future, we will explore collaborations with industry partners or cybersecurity organizations to further test and validate MalSSL in real-world scenarios.

We also recognized that the Malimg and Maldeb datasets may not reflect real-world malware. Thus, in the future, we want to test the system with other datasets and assess its resistance to other adversarial attacks. Further work could focus on fine-tuning different hyperparameters to enhance the performance of each SSL method in diverse classification scenarios. Additionally, investigating the interplay between epochs, batch size, and accuracy could provide deeper insights into optimizing the model.

VI. CONCLUSION

In this study, we explored the application of self-supervised learning methods in the domain of malware classification, addressing the challenges posed by diverse classification tasks. The proposed malware classification system with self-supervised learning (MalSSL) does not require dataset labeling and avoids the need for large computations.

It is tested on two different machine learning tasks: a) malware family classification and b) malware and benign classification. For the malware family classification task, it achieved a good accuracy result of 98.4% on the Malimg dataset. In the malware and benign classification task, we collected a new malware and benign dataset named the Maldeb dataset, consisting of two classes: malware and benign samples. MalSSL achieved an accuracy of 96.2% for classifying malware and benign tasks with the Maldeb dataset. MalSSL consistently outperformed other SSL methods: MoCo, SimCLR, SwAV, and SimSiam, achieving high accuracy with efficient training times.

Our research contributes to advancing the state-of-the-art in malware classification, demonstrating effective classification without the need for labeling the dataset first. We anticipate that this approach will accelerate and reduce the cost of malware classification.

ACKNOWLEDGMENT

Debi Amalia Septiyani collected malware and benign samples and validated the malware dataset. Halimul Hakim Khairul tested the Maldeb dataset with the CNN algorithm. Dani Agung Prastiyo collected benign samples and tested the malware audio representation.

REFERENCES

[1] C. Beek. (2021). *McAfee Labs Threat Report 2021*. [Online]. Available: https://www.hsdf.org/wp-content/uploads/2021/06/rp-quarterly-threats-apr-2021.pdf

[2] S. Ghafur, S. Kristensen, K. Honeyford, G. Martin, A. Darzi, and P. Aylin, "A retrospective impact analysis of the WannaCry cyberattack on the NHS," *NPJ Digit. Med.*, vol. 2, no. 1, p. 98, Oct. 2019, doi: 10.1038/s41746-019-0161-6.

[3] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, "Understanding the Mirai botnet," in *Proc. USENIX Secur.*, 2017, pp. 1093–1110. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis

[4] R. Langner, "Stuxnet: Dissecting a cyberwarfare weapon," *IEEE Secur. Privacy*, vol. 9, no. 3, pp. 49–51, May 2011, doi: 10.1109/MSP.2011.67.

[5] M. Al-Asli and T. A. Ghaleb, "Review of signature-based techniques in antivirus products," in *Proc. Int. Conf. Comput. Inf. Sci. (ICCIS)*, Apr. 2019, pp. 1–6, doi: 10.1109/ICCISci.2019.8716381.

[6] J. Scott. (2017). *Signature Based Malware Detection is Dead*. [Online]. Available: https://informationsecurity.report/Resources/Whitepapers/920fbb41-8dc9-4053-bd01-72f961db24d9_ICIT-Analysis-Signature-Based-Malware-Detection-is-Dead.pdf

[7] E. Raff and C. Nicholas, "A survey of machine learning methods and challenges for windows malware classification," 2020, *arXiv:2006.09271*.

[8] B. Kolosnjaji, A. Demontis, B. Biggio, D. Maiorca, G. Giacinto, C. Eckert, and F. Roli, "Adversarial malware binaries: Evading deep learning for malware detection in executables," in *Proc. 26th Eur. Signal Process. Conf. (EUSIPCO)*, Sep. 2018, pp. 533–537, doi: 10.23919/EUSIPCO.2018.8553214.

[9] H. S. Anderson and P. Roth, "EMBER: An open dataset for training static PE malware machine learning models," 2018, *arXiv:1804.04637*.

[10] A. Marx and M. Morgenstern. (2023). *Malware Statistics*. [Online]. Available: https://www.av-test.org/en/statistics/malware/

[11] D. Gibert, C. Mateu, and J. Planes, "The rise of machine learning for detection and classification of malware: Research developments, trends and challenges," *J. Netw. Comput. Appl.*, vol. 153, Mar. 2020, Art. no. 102526, doi: 10.1016/j.jnca.2019.102526.

[12] B. Quintero. (2004). *VirusTotal—Analyse Suspicious Files*. Chronicle Secur. Accessed: Apr. 17, 2024. [Online]. Available: https://www.virustotal.com/

[13] K. Zetter, *Countdown to Zero Day*. New York, NY, USA: Crown, Sep. 2014.

[14] R. Balestriero, M. Ibrahim, V. Sobal, A. Morcos, S. Shekhar, T. Goldstein, F. Bordes, A. Bardes, G. Mialon, Y. Tian, A. Schwarzschild, A. G. Wilson, J. Geiping, Q. Garrido, P. Fernandez, A. Bar, H. Pirsiavash, Y. LeCun, and M. Goldblum, "A cookbook of self-supervised learning," 2023, *arXiv:2304.12210*.

[15] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, "Malware images," in *Proc. 8th Int. Symp. Visualizat. Cyber Secur.*, Jul. 2011, pp. 1–7, doi: 10.1145/2016904.2016908.

[16] K. Kancherla and S. Mukkamala, "Image visualization based malware detection," in *Proc. IEEE Symp. Comput. Intell. Cyber Secur. (CICS)*, Apr. 2013, pp. 40–44, doi: 10.1109/CICYBS.2013.6597204.

[17] J.-S. Luo and D. C. Lo, "Binary malware image classification using machine learning with local binary pattern," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2017, pp. 4664–4667, doi: 10.1109/BIGDATA.2017.8258512.

[18] M. Kalash, M. Rochan, N. Mohammed, N. D. B. Bruce, Y. Wang, and F. Iqbal, "Malware classification with deep convolutional neural networks," in *Proc. 9th IFIP Int. Conf. New Technol., Mobility Secur. (NTMS)*, Feb. 2018, pp. 1–5, doi: 10.1109/NTMS.2018.8328749.

[19] J. Fu, J. Xue, Y. Wang, Z. Liu, and C. Shan, "Malware visualization for fine-grained classification," *IEEE Access*, vol. 6, pp. 14510–14523, 2018, doi: 10.1109/ACCESS.2018.2805301.

[20] D. Vasan, M. Alazab, S. Wassan, H. Naeem, B. Safaei, and Q. Zheng, "IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture," *Comput. Netw.*, vol. 171, Apr. 2020, Art. no. 107138, doi: 10.1016/j.comnet.2020.107138.

[21] V. Verma, S. K. Muttoo, and V. B. Singh, "Multiclass malware classification via first- and second-order texture statistics," *Comput. Secur.*, vol. 97, Oct. 2020, Art. no. 101895, doi: 10.1016/j.cose.2020.101895.

[22] A. Makandar and A. Patrot, "Malware class recognition using image processing techniques," in *Proc. Int. Conf. Data Manage., Analytics Innov. (ICDMAI)*, Feb. 2017, pp. 76–80, doi: 10.1109/ICDMAI.2017.8073489.

[23] M. Nisa, J. H. Shah, S. Kanwal, M. Raza, M. A. Khan, R. Damaševičius, and T. Blažauskas, "Hybrid malware classification method using segmentation-based fractal texture analysis and deep convolution neural network features," *Appl. Sci.*, vol. 10, no. 14, p. 4966, Jul. 2020, doi: 10.3390/app10144966.

[24] H. Guo, S. Huang, C. Huang, Z. Pan, M. Zhang, and F. Shi, "File entropy signal analysis combined with wavelet decomposition for malware classification," *IEEE Access*, vol. 8, pp. 158961–158971, 2020, doi: 10.1109/ACCESS.2020.3020330.

[25] A. Bensaoud and J. Kalita, "Deep multi-task learning for malware image classification," *J. Inf. Secur. Appl.*, vol. 64, Feb. 2022, Art. no. 103057, doi: 10.1016/j.jisa.2021.103057.

[26] Ö. Aslan and A. A. Yilmaz, "A new malware classification framework based on deep learning algorithms," *IEEE Access*, vol. 9, pp. 87936–87951, 2021, doi: 10.1109/ACCESS.2021.3089586.

[27] W. Al-Khater and S. Al-Madeed, "Using 3D-VGG-16 and 3D-Resnet-18 deep learning models and FABEMD techniques in the detection of malware," *Alexandria Eng. J.*, vol. 89, pp. 39–52, Feb. 2024, doi: 10.1016/j.aej.2023.12.061.

[28] M. D. AlGarni, R. AlRoobaea, J. Almotiri, S. S. Ullah, S. Hussain, and F. Umar, "An efficient convolutional neural network with transfer learning for malware classification," *Wireless Commun. Mobile Comput.*, vol. 2022, pp. 1–8, Oct. 2022, doi: 10.1155/2022/4841741.

[29] I. Alam, M. Samiullah, U. Kabir, S. Woo, C. K. Leung, and H. H. Nguyen, "SREMIC: Spatial relation extraction-based malware image classification," in *Proc. 18th Int. Conf. Ubiquitous Inf. Manage. Commun. (IMCOM)*, Jan. 2024, pp. 1–8, doi: 10.1109/IMCOM60618.2024.10418339.

[30] R. Mitsuhashi and T. Shinagawa, "Deriving optimal deep learning models for image-based malware classification," in *Proc. 37th ACM/SIGAPP Symp. Appl. Comput.*, Apr. 2022, pp. 1727–1731, doi: 10.1145/3477314.3507242.

[31] Y. LeCun and I. Misra. (2021). *Self-Supervised Learning: The Dark Matter of Intelligence*. [Online]. Available: https://ai.facebook.com/blog/self-supervised-learning-the-dark-matter-of-intelligence/

[32] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North*, vol. 1, Oct. 2019, pp. 4171–4186, doi: 10.18653/v1/n19-1423.

[33] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A robustly optimized BERT pretraining approach," 2019, *arXiv:1907.11692*.

[34] S. Ruder, A. Søgaard, and I. Vulić, "Unsupervised cross-lingual representation learning," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics, Tutorial Abstr.*, 2019, vol. 1911, no. 02116, pp. 31–38, doi: 10.18653/v1/p19-4007.

[35] C. Doersch, A. Gupta, and A. A. Efros, "Unsupervised visual representation learning by context prediction," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1422–1430, doi: 10.1109/ICCV.2015.167.

[36] H. Barlow, "Redundancy reduction revisited," *Network: Comput. Neural Syst.*, vol. 12, no. 3, pp. 241–253, Mar. 2001, doi: 10.1088/0954-898x/12/3/301.

[37] D.-H. Lee, "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in *Proc. ICML Workshop Challenges Represent. Learn.*, 2013, pp. 1–6. [Online]. Available: https://www.kaggle.com/blobs/download/forum-message-attachment-files/746/pseudo_label_final.pdf

[38] I. Misra and L. van der Maaten, "Self-supervised learning of pretext-invariant representations," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 6706–6716, doi: 10.1109/CVPR42600.2020.00674.

[39] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," 2020, *arXiv:2002.05709*.

[40] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 9726–9735, doi: 10.1109/CVPR42600.2020.00975.
[41] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, "Deep clustering for unsupervised learning of visual features," in *Computer Vision—ECCV* (Lecture Notes in Computer Science, Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 11218. Cham, Switzerland: Springer, 2018, pp. 139–156, doi: 10.1007/978-3-030-01264-9_9.
[42] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, "Unsupervised learning of visual features by contrasting cluster assignments," 2020, *arXiv:2006.09882*.
[43] P. Morgado, I. Misra, and N. Vasconcelos, "Robust audio-visual instance discrimination," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 12929–12940, doi: 10.1109/CVPR46437.2021.01274.
[44] J. B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko, "Bootstrap your own latent a new approach to self-supervised learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 21271–21284, doi: 10.48550/arXiv.2006.07733goo.
[45] X. Chen and K. He, "Exploring simple Siamese representation learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 15745–15753, doi: 10.1109/CVPR46437.2021.01549.
[46] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny, "Barlow twins: Self-supervised learning via redundancy reduction," 2021, *arXiv:2103.03230*.
[47] F. Del Pup and M. Atzori, "Applications of self-supervised learning to biomedical signals: A survey," *IEEE Access*, vol. 11, pp. 144180–144203, 2023, doi: 10.1109/ACCESS.2023.3344531.
[48] M. Dib, S. Torabi, E. Bou-Harb, N. Bouguila, and C. Assi, "EVOLIoT," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, May 2022, pp. 452–466, doi: 10.1145/3488932.3517393.
[49] S. Seneviratne, R. Shariffdeen, S. Rasnayaka, and N. Kasthuriarachchi, "Self-supervised vision transformers for malware detection," *IEEE Access*, vol. 10, pp. 103121–103135, 2022, doi: 10.1109/ACCESS.2022.3206445.
[50] S. J. I. Ismail. (2024). *Maldeb Dataset*. Accessed: Apr. 17, 2024. [Online]. Available: https://ieee-dataport.org/documents/maldeb-dataset
[51] J. Howard. (2019). *Imagenette Dataset*. Accessed: Jan. 2, 2024. [Online]. Available: https://github.com/fastai/imagenette
[52] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255, doi: 10.1109/CVPR.2009.5206848.
[53] M. Mirea, V. Wang, and J. Jung, "The not so dark side of the darknet: A qualitative study," *Secur. J.*, vol. 32, no. 2, pp. 102–118, Jun. 2019, doi: 10.1057/s41284-018-0150-5.
[54] I. Ahl. (2011). *TekDefense Malware Samples*. Accessed: Jun. 11, 2023. [Online]. Available: http://www.tekdefense.com/downloads/malware-samples/
[55] Y. Nativ and S. Shalev. (2015). *The Zoo—A Live Malware Repository*. Accessed: Jun. 11, 2023. [Online]. Available: https://thezoo.morirt.com
[56] D. Vito and Aymen. (2021). *The Malware Repo*. Accessed: Jun. 11, 2023. [Online]. Available: https://github.com/Da2dalus/The-MALWARE-Repo
[57] Endermanch and Andrew. (2022). *Malware Database*. Accessed: Jun. 11, 2023. [Online]. Available: https://github.com/Endermanch/MalwareDatabase
[58] Spamhaus Technology. (2007). *The Malware Bazaar*. Accessed: Jun. 11, 2023. [Online]. Available: https://bazaar.abuse.ch/
[59] Red Ventures. (2024). *CNET*. Accessed: Mar. 18, 2024. [Online]. Available: https://download.cnet.com/windows/
[60] S. Media. (1999). *Sourceforge*. Accessed: Mar. 18, 2024. [Online]. Available: https://sourceforge.net/
[61] BetaNews. (1998). *Fileforum*. Accessed: Mar. 18, 2024. [Online]. Available: https://fileforum.com
[62] A. Lee. (2004). *The Portable Freeware Collection (TPFC)*. Accessed: Mar. 18, 2024. [Online]. Available: https://www.portablefreeware.com
[63] Virustotal. (2024). *VirusTotal—How It Works*. Accessed: Apr. 14, 2024. [Online]. Available: https://docs.virustotal.com/docs/how-it-works
[64] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778, doi: 10.1109/CVPR.2016.90.
[65] I. Susmelj and M. Heller. (2019). *Lightly*. Lightly AG. Accessed: Jun. 13, 2023. [Online]. Available: https://www.lightly.ai/
[66] S. Ismail. (2023). *Malware Detection System With Self Supervised Learning—Repository*. Accessed: Jun. 19, 2023. [Online]. Available: https://github.com/julismail/Self-Supervised
[67] S. J. I. Ismail, H. P. Gemilang, B. Rahardjo, and Hendrawan, "Self-supervised learning implementation for malware detection," in *Proc. 8th Int. Conf. Wireless Telematics (ICWT)*, Jul. 2022, pp. 1–6, doi: 10.1109/ICWT55831.2022.9935463.
[68] S. Yajamanam, V. R. S. Selvin, F. Di Troia, and M. Stamp, "Deep learning versus gist descriptors for image-based malware classification," in *Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy*, Jan. 2018, pp. 553–561, doi: 10.5220/0006685805530561.
[69] H. H. Khairul, "Design and implementation of images dataset malware generating system based on deep learning," Institut Teknologi Bandung, Bandung, Indonesia, Tech. Rep. 18118033, 2022. [Online]. Available: https://digilib.itb.ac.id/gdl/view/67891/DETEKSI-MALWARE-MENGGUNAKAN-ARSITEKTUR-CNN-DENGAN-HYPERPARAMETER-TUNING
[70] Avira Operation GmBH. (2017). *NightVision-Using Machine Learning to Defeat Malware*. [Online]. Available: https://www.webassetscdn.com/avira/prod/cache-buster-1598423379/assets/oem.avira.com/resources/todelete/whitepaper_NightVision_EN_20170704.pdf
[71] Bitdefender. (2017). *Bitdefender Machine Learning Technical Brief*. [Online]. Available: https://explore.bitdefender.com/business-security/machine-learning-technical-brief
[72] S. Afroz and R. Gupta, "AVAST-AI and machine learning," in *Proc. Enigma Usenix*, 2020, pp. 1–94. Accessed: Mar. 22, 2024. [Online]. Available: https://www.usenix.org/sites/default/files/conference/protected-files/enigma2020_slides_afroz.pdf
[73] T. Micro. (2024). *Machine Learning Masters*. [Online]. Available: https://www.trendmicro.com/content/dam/trendmicro/global/en/global/docs/infographic/ifg-machine-learning-masters.pdf
[74] J. Saxe and K. Berlin, "Deep neural network based malware detection using two dimensional binary program features," in *Proc. 10th Int. Conf. Malicious Unwanted Softw. (MALWARE)*, Oct. 2015, pp. 11–20, doi: 10.1109/MALWARE.2015.7413680.
[75] C. Huang and A. Karnik. (2021). *McAfee Labs: The Rise of Deep Learning for Detection and Classification of Malware*. McAfee Labs. Accessed: Mar. 23, 2024. [Online]. Available: https://www.mcafee.com/blogs/other-blogs/mcafee-labs/the-rise-of-deep-learning-for-detection-and-classification-of-malware/
[76] A. Patel. (2020). *F-Secure: Detection of Anomalous Process Creation Chains Using Word Vectorization, Normalization, and an Autoencoder*. Accessed: Mar. 22, 2024. [Online]. Available: https://blog.f-secure.com/process-creation-chains/
[77] S. O'sullivan, D. Vangel, and Y. Rhee. (2024). *Advanced Technologies At the Core of Microsoft Defender Antivirus*. [Online]. Available: https://learn.microsoft.com/en-us/microsoft-365/security/defender-endpoint/adv-tech-of-mdav?view=o365-worldwide
[78] E. Raff and C. K. Nicholas. (2021). *Machine Learning for Malware Detection*. Accessed: Mar. 22, 2024. [Online]. Available: https://media.kaspersky.com/en/enterprise-security/Kaspersky-Lab-Whitepaper-Machine-Learning.pdf

SETIA JULI IRZAL ISMAIL received the bachelor's degree in electrical engineering from STT Telkom, Bandung, Indonesia, in 2003, and the master's degree in computer science from Institut Teknologi Bandung, Indonesia, in 2014, where he is currently pursuing the Ph.D. degree in electrical engineering.

He was a Malware Analyst with Indonesia Computer Emergency Response Team (ID-CERT), Bandung, from 2012 to 2019. Since 2009, he has been a Lecturer with Telkom University, Bandung. In 2016, he received a grant from Asia Pacific Computer Emergency Response Team (AP-CERT) to present his research at the AP-CERT AGM Annual Meeting, Tokyo, Japan. He delivered guest lectures on malware, cryptography, and security at Universitat Autonoma de Barcelona, Spain, in 2018, funded by Erasmus. From 2022 to 2023, he conducted research with Kumamoto University, Japan, funded by Jasso. He is a member of the Honeynet Indonesia Chapter and Indonesia Academic Computer Security Incident Response Team (ACAD-CSIRT). His current research interests include malware detection, machine learning, and computer security.

HENDRAWAN (Member, IEEE) received the B.S. degree in electrical engineering from Institut Teknologi Bandung, Bandung, Indonesia, in 1985, the M.S. degree in telecommunication engineering and the Ph.D. degree in electrical engineering from the University of Essex, Essex, in 1990 and 1995, respectively. He has been a Full Professor with Institut Teknologi Bandung, since 2023. He is currently a Researcher and a Lecturer with the School of Electrical and Informatics, Institut Teknologi Bandung. He is also the Head of the Telecommunication Engineering Research Group. His research interests include wireless technology and machine learning.

BUDI RAHARDJO (Member, IEEE) received the B.S. degree in electrical engineering from Institut Teknologi Bandung, Bandung, Indonesia, in 1986, the M.S. and Ph.D. degrees in computer science from the University of Manitoba, Manitoba, Canada, in 1990 and 1997, respectively. He is currently a Researcher and a Lecturer with the School of Electrical Engineering and Informatics, Institut Teknologi Bandung. His current research interests include computer security and cryptography. He is the Founder and the Chairperson of Indonesia Computer Emergency Response Team (ID-CERT). He is one of the founders of Asia Pacific Computer Emergency Response Team (AP-CERT).

TUTUN JUHANA (Member, IEEE) received the bachelor's degree in electrical engineering, the master's degree in telecommunication engineering, and the Ph.D. degree from Institut Teknologi Bandung (ITB), Indonesia, in 1995, 1999, and 2011, respectively. He is currently an Associate Professor of telecommunication engineering with the School of Electrical Engineering and Informatics (SEEI), ITB. He is also the Dean of SEEI, ITB, and a member of the National Center for Sustainable Transportation Technology. His research interests include wireless ad-hoc networking, vehicular ad-hoc networks, the Internet of Things, and ambient assisted living.

YASUO MUSASHI (Member, IEEE) has been with the Information Processing Center, Kumamoto University, as a Cooperative Researcher and an Assistant Professor, since 1997. Since 2002, he has been an Associate Professor with the Center for Multimedia and Information Technologies. He was a Guest Scientist with the Johann Wolfgang Goethe Universität Frankfurt am Main, from January 2005 to July 2005. Since May 2014, he has been with the Center for Management and Information Technologies (CMIT), Kumamoto University. He has been a Full Professor with CMIT, since 2015, and the Research Education Institute for Semiconductors and Informatics (REISI), since 2023. His research interests include computer network security and developing security incident detection and prevention systems.