Computers & Security 140 (2024) 103769

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

Overcoming the lack of labeled data: Training malware detection models using adversarial domain adaptation

Sonam Bhardwaj [a,*], Adrian Shuai Li [b], Mayank Dave [a], Elisa Bertino [b]

[a] National Institute of Technology Kurukshetra, Department of Computer Engineering, Kurukshetra, India
[b] Purdue University, Department of Computer science, West Lafayette, IN, USA

ARTICLE INFO

Keywords:
Transfer learning
Malware detection
Malware images
CNN
Domain adaptation
Generative adversarial networks

ABSTRACT

Many current malware detection methods are based on supervised learning techniques, which however have certain limitations. First, these techniques require a large amount of labeled data for training which is often difficult to obtain. Second, they are not very effective when there are differences in domain distribution between new malware and known malware. To address these issues, we propose MD-ADA – a malware detection framework that leverages adversarial domain adaptation (DA). DA allows one to adapt a training malware dataset available at a domain, referred to as the source, for training a classifier in another domain, referred to as the target. DA, typically used when the target has limited training malware data available, maps the source and target datasets into a common latent space. As we use an image representation for malware binaries, MD-ADA uses a convolution neural network (CNN) providing a lossless image embedding for the source and target datasets. MD-ADA also employs a generative adversarial network (GAN) for malware classification that is suitable for scenarios with few target-labeled data where the distribution of the features is similar (homogeneous) or different (heterogeneous). We have carried out several experiments to assess the performance of MD-ADA. The experiments show that MD-ADA outperforms the fine-tuning approach with an accuracy of 99.29% on the BODMAS dataset, 89.3% for the Malevis dataset on homogeneous feature distribution, and 90.12% on the CICMalMem2022 dataset (Target) and 83.23% on the Microsoft Kaggle dataset (Target) for heterogeneous feature distribution. The observed F1-scores of 99.13% and 87.5% for homogeneous feature distributions and 91.27% and 81.7% for heterogeneous distributions indicate that the MD-ADA performance is satisfactory for both data distributions when the target has very few labeled data.

1. Introduction

Malicious software (malware) has been present since the earliest days of technology, but its sophistication and innovation have grown over time. Recent reports indicate that malware is expanding at an alarming rate. In 2022, the SonicWall Capture Advanced Threat Protection (ATP) report (Sonic Wall Threat Report, 2023) identified 119,549 PDF-based threats, a 35% increase from 2021. The number of infected Microsoft Office files increased marginally to 54,371. Cryptojacking and IoT malware are the contributing attacks for the rise in malware up to 5.5 billion in 2022.

To deal with constantly evolving malware, rapid and accurate detection and classification of malware is critical. However, such activities often require expensive manual analyses (Moser et al., 2008). It may take a long time (hours to weeks, depending on how intricate the malware is) to do such an investigation manually (Anderson et al., 2014).

To address the problem of malware detection and classification, recent approaches have used deep learning (DL) techniques with encouraging outcomes (Ni et al., 2018) (Cui et al., 2018) (Darem et al., 2021) (Jian et al., 2021). DL can extract features automatically to eliminate manual feature extraction. Some approaches employ DL to improve malware detection by converting bytecodes into images and categorizing the images using techniques such as deep residual networks, deep belief networks, deep convolutional neural networks, and deep recurrent networks (Lu and Li, 2019). The basic idea of these approaches is to leverage the distinguishing patterns in malware images. Furthermore, by analyzing the similarities between images of malware from the same family, the image representation helps in identifying correlations between different families of malware. It is feasible for a DL

* Corresponding author.
E-mail addresses: bhardw24@purdue.edu (S. Bhardwaj), li3944@purdue.edu (A.S. Li), mdave@nitkkr.ac.in (M. Dave), bertino@purdue.edu (E. Bertino).

https://doi.org/10.1016/j.cose.2024.103769
Received 24 August 2023; Received in revised form 14 December 2023; Accepted 9 February 2024
Available online 19 February 2024

model to automatically recognize key patterns and extract relevant features from images that belong to the same family because these images share similar features. Some approaches also use global and local features to generate malware images (Zhou et al., 2020). By supplying an entire image to a pre-trained model and aggregating the outputs of intermediate layers, global features (image shape, size, and color) are extracted. Instead of characterizing the entire image region, local features (objects, edges, and points) describe subregions of the image.

In order to train a DL model for malware image classification, local and global regions representing malware features must be identified. However, a tremendous amount of labeled data is needed for training DL models to accurately detect malware. Also, in some scenarios, it is necessary to update models regularly to prevent erroneous detection due to the evolution of malware and changes in its features. However, it may be time-consuming and expensive to collect and categorize data for many emerging malware families. The lack of labeled data makes supervised model learning ineffective. When using methods like SMOTE (Bhagat and Patil, 2015), oversampling techniques (Gosain and Sardana, 2017) can frequently cause model bias and, in turn, lead to classification errors. Additionally, labeled data from one domain might not be suitable for DL-model training in another domain. For example, a model trained on images of Android malware might not be able to correctly identify IoT malware. The problem of low model performance is made worse by an inappropriate distribution of data for testing and training.

Such drawbacks do not only affect malware classifiers; they also affect models used for other security functions, such as Network Intrusion Detection (NID). For example, a NID model trained for the detection of malicious flows in a conventional network may not be directly used for detecting malicious flows in an IoT network. To address those shortcomings, Singla et al. (2020) proposed an architecture for training a NID classifier in a minimal labeled data scenario that utilizes an Adversarial Domain Adaptation (ADA) strategy. The framework is trained to learn the mapping between the source and target domains and predict whether the samples belong to an attack or benign class. ADA utilizes GANs to achieve domain-invariant mapping between source and target datasets.

Therefore, in our work we extend the GAN approach by Singla et al., wherein, ADA is used for mapping different malware distributions from source and target, and an enhanced GAN architecture is used for malware detection. The GAN takes as input a lossless encoding of the input images generated by a convolutional neural network (CNN), which transforms the images from a 2-D matrix representation into a 1-D matrix representation.

The key contributions of our work are:

- We propose a framework, referred to as MD-ADA, for malware detection using adversarial ADA in which the malware binaries are represented as images.
- MD-ADA employs the GAN model for malware detection for similar (homogeneous) and distinct features (heterogeneous) distribution.
- We evaluate the performance of MD-ADA when it is trained on target samples that comprise fewer samples than source data.
- MD-ADA eliminates the need for Principal Component Analysis (PCA) to reduce the data dimensionality. This is a major improvement because PCA is often used to lower the number of dimensions in data, but it loses information and makes things more complicated.

The rest of this article is organized as follows. Section 2 provides background concepts underlying our framework. Section 3 introduces and explains the proposed MD-ADA framework. Section 4 describes the experimental design and datasets used to evaluate the framework's efficacy. Section 5 presents and analyzes the experimental results. Section 6 reports the performance evaluation on different datasets. Section 7 discusses related works. Section 8 compares the proposed approach with the state-of-art approaches. Section 9 concludes the paper with a discussion of future work.

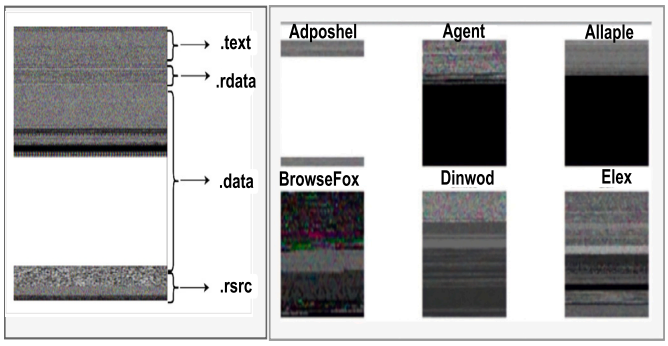

Fig. 1. Malware and malware family representation as grayscale images.

2. Background

In this section, we first provide some background on the image representation of malware. We then introduce ML concepts relevant to the design of the MD-ADA framework.

2.1. Malware as images

For representing malware binaries as grayscale images, the binaries are first converted into 8-bit unsigned vectors. These vectors are then transformed into a 2-D array that represents a grayscale image with pixel values between 0 and 255. Fig. 1 shows how grayscale images of malware binaries convey information and sections similar to portable executable (PE) file formats. It is also noteworthy how the size of the binary files corresponds to a specific range of image widths. For example, the image of a file with size less than 10 kB has a width of 32, the image of a file with a size of 60–100 kB has a width of 256, and the image of file with size more than 1000 kB has a width of 1024 (Nataraj et al., 2011).

For image-based malware classification, pre-trained DL models perform better than non-image-based classification models (like KNN) (Bhodia et al., 2019). The models compute the image texture features for classification, identifying areas with less texture and more texture (noisy) information as malicious. The region with less texture has a lower entropy, whereas the regions that contain malware code are considered to be noisy and have higher entropy. Images generated from static features are used to find malware variants from the same family by identifying visually similar patterns. Even though the texture information is helpful for obtaining the static features, it is essential to collect information about how the malware operates in the context of the environment in which it is being executed. Therefore, in a behavior-based technique, the behavior of malware is turned into images by building a behavior-to-color mapping based on the captured behavioral features (API calls, sequences, etc.) and assigning each captured behavior a specific color in the range [0,255]. The images created from the static or dynamic features are used to discover variants of malware by looking for patterns that are visually similar and originate from the same family of malicious software (Shaid and Maarof, 2014).

2.2. Generative adversarial networks (GANs) and adversarial domain adaptation

During the training phase of GAN, the generator module creates false or fake image samples that look realistic (like real images). The discriminator module learns to recognize the generator's fake image samples. Therefore, the performance of the GAN depends on how well the discriminator is trained to distinguish the fake and real image samples.

The training of the discriminator depends on how it learns on the labeled data provided. Therefore, training the model in case of minimum or limited labeled data is a challenge and also labeling the unlabeled data in a target domain is challenging. Even with a small amount of unlabeled or sparse data, the transferability of a machine-learning model can be improved through the process of domain adaptation (DA). For this, some related information about the data is required to handle the data distribution of target and source data over the sample space or domains. A classifier is able to modify its conclusions in order to improve its ability to generalize by using the information that is available. The impacts of domain shifts to avoid biased classification are mitigated by domain adaptability. The most recent approaches to DA either combine both domains into a common feature space or reconstitute the target domain using relevant information. Domain shift parameters such as mean discrepancy or correlation distances can be decreased by improving the representation of the domain samples.

The relevant DA concepts underlying the design of our MD-ADA framework can be summarized as follows.

Definition 1 *(Domain).* The feature space 'χ' and the marginal distribution P constitute the whole of a domain D denoted by $D = \{\chi, P(X)\}$.

Assume to have a feature space 'χ' of color images of size 32*32, each pixel in the range [0-255], the training set is represented by 'X' where

$$X = \left\{\mathbf{x}^{(i)}\right\}_{i=1}^{\mathbf{N}},\ \mathbf{x}^{(i)} \in \chi \tag{1}$$

The training set follows a certain probability distribution, denoted by P(X) or $P_X(\mathbf{x})$. If the dataset 'X' contains RGB color images, for instance, the probability of all-black images should be rather low.

Definition 2 *(Task).* A task $\mathcal{T}$ is composed of a label space $\mathcal{Y}$ and a probability decision function $f(x) : P(Y|X)$, such that, $\{\mathcal{T} = \mathcal{Y}, P(Y|X)\}$. The decision function f is assumed to be implicit and learns from the sample data.

Our label space, $\mathcal{Y}$, includes possible labels for samples in our problem. For example, $\mathcal{Y} = \{0, 1\}$ in binary classification denotes the malicious class with '0' and the normal class with '1'.

We also have the set of labels for our dataset represented as Y:

$$Y = \left\{\mathbf{y}^{(i)}\right\}_{i=1}^{\mathbf{N}},\ \mathbf{y}^{(i)} \in \mathcal{Y} \tag{2}$$

We want to learn the target function $\hat{y} = f(x)$ and from a probabilistic perspective learn $P(\hat{y}|x)$

Therefore, based on Definition 1 and Definition 2, we can define the domains and tasks for source and target (Pan and Yang, 2009) as:

Definition 3 *(Source domain and task).* A source domain is defined by $\mathcal{D}_S = \{\chi_S, P(X_S)\}$ and source learning task as $\mathcal{T}_S = \{\mathcal{Y}_S, P(Y_S|X_S)\}$.

Definition 4 *(Target domain and task).* A target domain is defined by $\mathcal{D}_T = \{\chi_T, P(X_T)\}$ and target learning task as $\mathcal{T}_T = \{\mathcal{Y}_T, P(Y_T|X_T)\}$.

Therefore, Transfer Learning(TL) is associated with the learning of the target function $P(Y_T|X_T)$ using knowledge of the source domain (D_S) and source tasks (T_S) when $D_S \neq D_T$ or $T_S \neq T_T$ with the constraint of minimal or no label on the target domain.

In our work, we consider the two different cases of homogeneous and heterogeneous feature spaces with minimal labeled data in the target domain.

Case 1: In the case of homogeneous transfer learning, $X_T = X_S$ and $Y_T = Y_S$ denote that the feature space and label space are similar. The aim is to minimize the information gap between these two domains, $P(X_T) \neq P(X_S)$ or $P(Y_T|X_T) \neq P(Y_S|X_S)$.

Case 2: Since the source and target domains may not share some features or labels, heterogeneous TL involves feature spaces $X_T \neq X_S$ (often non-overlapping) or $Y_T \neq Y_S$.

Heterogeneous TL approaches transform the initial problem into a homogeneous TL problem by bridging the gap that exists between the two feature spaces. In this new problem, the only divergences that need to be adjusted are those that persist in the distribution being either marginal or conditional.

A set of loss functions is employed to train our Generative Adversarial Network (GAN) framework. Specifically, we utilize the domain loss (D_{Loss}) and classification loss (C_{Loss}), which are mathematically defined in equations (3) and (4). The domain loss function quantifies the variance between the actual domain values and the predicted values generated by the discriminator during domain prediction. The term "domain prediction" pertains to the computational procedure of estimating the likelihood that a particular data sample originates from either the source distribution or the target distribution. The classification loss function operates in a manner akin to the D_{Loss} function, with the distinction that the C_{Loss} pertains to the disparity between the true value and the value anticipated by the classifier (class prediction). In this context, "class prediction" refers to the likelihood that a given data sample belongs to either the malware or benign category.

The domain loss is calculated as:

$$D_{Loss}(G, D) = -E_{X_S}[log\, D(G(X_S))] - E_{X_T}[log(1 - D(G(X_T))] \tag{3}$$

Here, E_{X_S} and E_{X_T} demote the predicted values of source and target domain samples, respectively. $D(G(X_S))$ is the probability that the discriminator will correctly identify the source domain sample. $D(G(X_T))$ is the probability of discriminator for a target domain sample. The classification loss is calculated as:

$$C_{loss}(G) = -E_{X_{malware}}[log(G(x))] - E_{x_{normal}}[log(1 - G(x))] \tag{4}$$

where the predicted values of malware and normal samples for true data distributions are denoted by $E_{X_{malware}}$ and $E_{X_{normal}}$, respectively. G(x) is the generator probability of classifying a sample as malicious and $1 - G(x)$ is the probability for the generator determining a sample to be normal over the total distribution. Therefore, the model is trained with a goal to minimize the discriminator loss (D_{Loss}) and classifier loss (C_{Loss}) simultaneously. The categorical cross-entropy function is a built-in function in the tensor-flow package, and it is used in the evaluation of the loss functions that are used by the model. The next section describes in detail all the components of the MD-ADA framework.

3. The MD-ADA framework

MD-ADA operates in two main phases, namely, malware image preparation and malware Detection (see Fig. 2). The first phase involves the collection of malware binaries, the conversion of binaries to malware images, and the pre-processing of these images for malware detection. The second phase trains the GAN architecture that uses TL to classify the images based on their labels and domains. The trained model is then used to make predictions on the test data that have not yet been seen in the training set.

3.1. Malware image preparation

This phase is responsible for providing an image embedding compatible with the GAN architecture. It comprises the following stages:

- **Collection and creation of raw malware images**: At this stage, we collect the malware binaries and convert them into 8-bit unsigned vectors. These vectors are then transformed into a 2-D array representing the grayscale images with pixel values ranging in the interval [0-255]. Since these images are very large and variable in size, ranging from 600 pixels to 1000 pixels, they are intricate

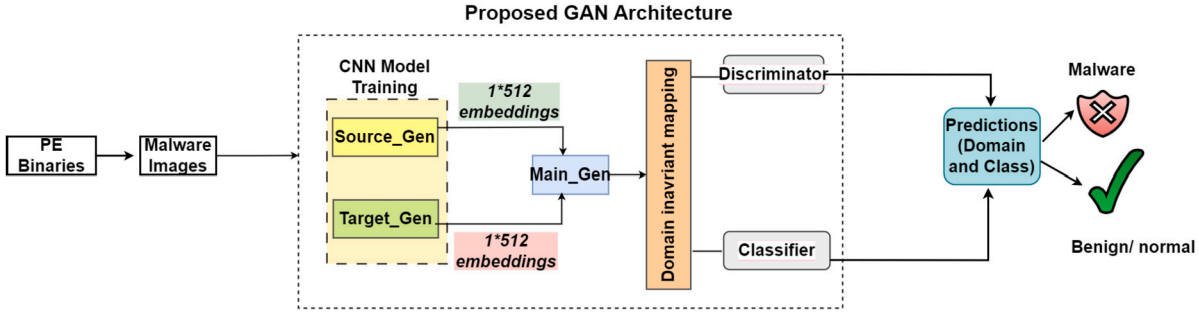

Fig. 2. Overview of the MD-ADA framework.

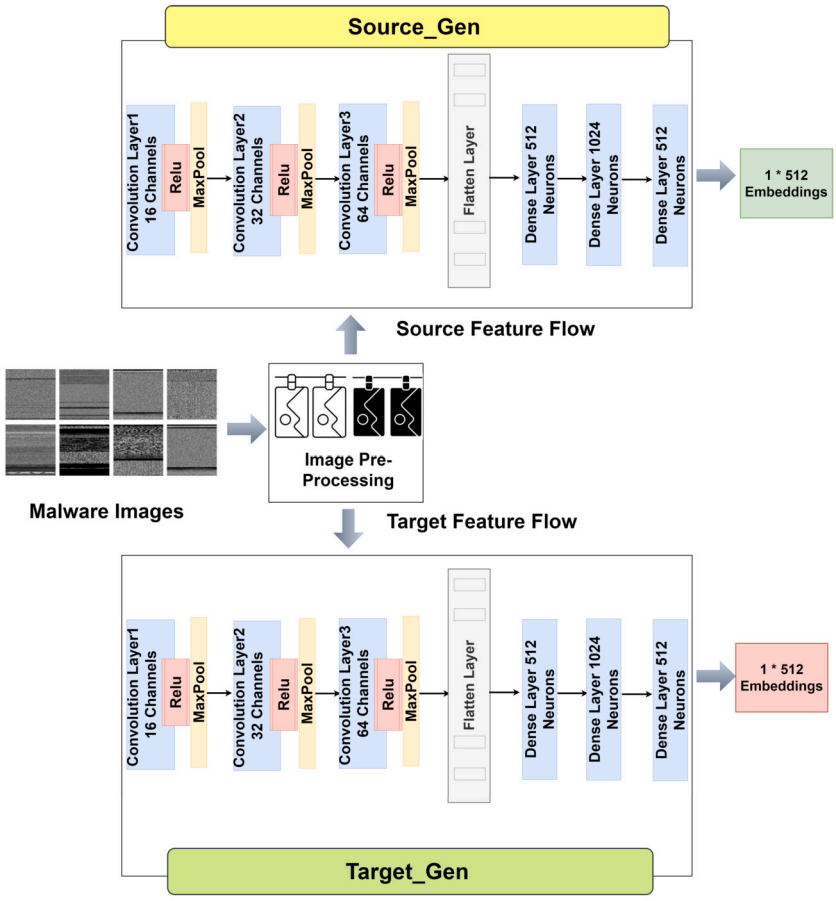

Fig. 3. Preparation of image data by source generator and target generator through the CNN-based Training Model.

to handle. Thus, we use image scaling and image resizing techniques as our image pre-processing method to obtain images of width 224*224. These images are then forwarded to the CNN for creating image embeddings for the main generator of GAN architecture.

- **CNN Model Training**: At this stage, we create processed images for malware detection as shown in Fig. 3. The CNN is integrated into both of the generators, source_gen, and target_gen, in order to prepare the image embeddings. Our CNN architecture contains three convolution layers, one flattened layer, three fully connected dense layers, and an output layer. The first convolution layer consists of 16 channels, the ReLu activation function, and the max-pooling layer with a size of 4. The second and third convolution layers are similar to the first layer except that the second layer contains 32 channels and the third layer contains 64 channels. The flattened layer transforms the existing data into a 1-D array that is forwarded to the dense layers. Dense layers process the received 1D array to produce the 1*512 image embeddings. Each of the three dense layers contains 512 neurons, 1024 neurons, and 512 neurons, respectively. In all the dense layers, we use the ReLu activation function. These embeddings are then fed to the main generator architecture.

3.2. Malware detection

This phase is responsible for malware detection using the enhanced GAN architecture that receives 1*512 image embedding created using source_gen and target_gen. As shown in Fig. 2, the architecture comprises three generators (source, target, and main), a discriminator, and a classifier. The source (source_gen) and target generators (target_gen) accept the source data and target data, respectively. Both generators receive as input data from the different domains, i.e., the source domain and the target domain. The corresponding output from the two generators is utilized as the input for the main generator, which subsequently

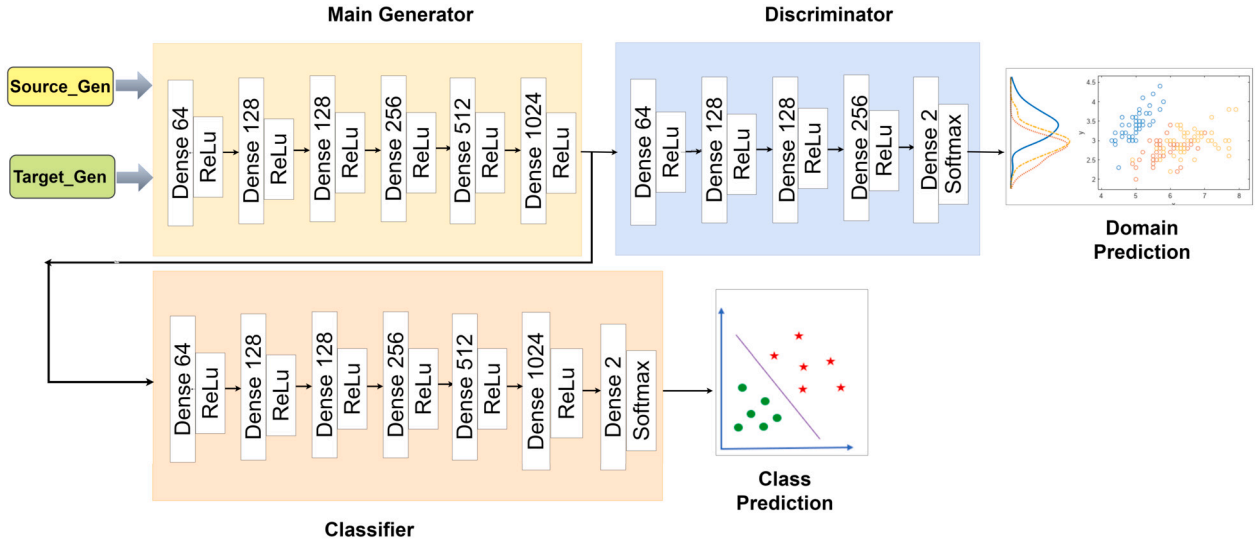

Fig. 4. Detailed Architecture of MD-ADA.

generates images for both the classifier and discriminator through the implementation of domain mapping operation.

The classifier classifies the generated images into binary classes i.e., malware image or benign software image. The role of the discriminator is to identify the source and the target domain of the images. It updates the weights via back-propagation. When a sample is processed, its output is mapped to establish whether it comes from the source domain or the target domain. The adversarial learning component is only trained on the target data, while the distribution of the source data is held constant throughout the process. This concept can be contrasted to the model in which the real data distribution is preserved while the generated data distribution is trained.

Fig. 4 shows the detailed GAN architecture. Here, the main generator, the classifier, and the discriminator consist of three parts: an input layer, an output layer, and a pair of fully connected layers. For the main generator, the first dense layer is composed of 64 neurons, the second layer and the third layer include 128 neurons, the fourth layer has 256 neurons, the fifth layer has 512 neurons, and the sixth layer (output layer) has 1024 neurons. Each layer uses ReLu activation function.

The classifier receives the input from the main generator and performs class prediction and label prediction. Its architecture is similar to the architecture of the main generator with an additional softmax layer for classification. It consists of six dense layers and one output layer, where the ReLu function is used for dense layers and the softmax function is used for the output layer to classify the data that include both malicious and normal classes.

The discriminator is responsible for domain prediction. It contains five dense layers including one output layer. The first layer contains 64 neurons, the second and the third layer contain 128 neurons, the fourth layer contains 256 neurons, and the fifth layer (output layer) contains 2 neurons for domain discrimination. The ReLu activation function is used for dense layers and softmax for the output layer.

4. Experimental details

For our experiments, we have deployed MD-ADA on a T4 GPU with 16 GB of memory and a 1.59 GHz clock speed and delivers 8.1 times the performance of a dual-core CPU with 12 GB of RAM, and 512 gigabytes of storage space. The model is trained on both homogeneous and heterogeneous distribution of images. To achieve this the data loaders for the mini-batch training are created with a batch size of 16, and a random seed value of 123.

Table 1
Representation of different classes and distribution (%) of samples in the Malevis dataset.

Malware Class Name	Distribution (%)	Malware Class Name	Distribution (%)
Adposhel	3.47	Injector	3.47
Agent	3.30	InstallCore	3.51
Allaple	3.36	MultiPlug	3.50
Amonetize	3.49	NeoreKlami	3.51
Androm	3.51	Neshta	3.49
AutoRun	3.46	Regrun	3.40
BrowseFox	3.46	Sality	3.50
Dinwod	3.50	Snarasite	3.51
Elex	3.51	stantinko	3.51
Expiro	3.52	VBA	3.51
Fasong	3.51	VBKrypt	3.48
HackKMS	3.50	Vilsel	3.48
HLUX	3.51	other (Normal)	12.87

4.1. Dataset description and pre-processing

In our experiments, we have used two datasets: Malevis (2013) and Microsoft-Kaggle (2023). Malevis, used for experiments on the homogeneous case, has image representations for 26 classes, where 25 classes are malicious classes and 1 class is designated as benign. These images are used to create homogeneous malware datasets. There are two different image sizes in the data, measuring in at width 224x224 pixels and 300x300 pixels. From a pool of 25 malware families, 15 were chosen as the source, while the remainder contributed to the target data. Source and target each received half of the benign class data. There are a total of 5,814 images in the source, while the target has 3,875 images. Table 1 provides information on the types and distribution of samples in the Malevis dataset.

In the case of heterogeneous feature space, Malevis is used as the source data, and Microsoft-Kaggle as the target data. There are 5,814 images in the source dataset and 1,099 image samples in the target dataset. The corrupted images and lower-sized images were eliminated from the data collection as they were unfit for the model. After fil-

tering the corrupted images from the dataset, the remaining images were downsized to 320x320 and 240x240 without losing any information. Here, huge-size images were examined using the GANs, starting at 320x320 and scaling up to 600x600, while previous efforts only evaluated 32x32-sized images. The pre-processing of datasets has been performed for image datasets, w.r.t. homogeneous and heterogeneous sources. For both the feature spaces, images were handled specifically. Firstly, the PE binaries are converted to images. The images were converted from csv through the OpenCV python library (OpenCV Python Library, 2023), after which the images were resized for the model. The model processes the specific size of images. For this, the process involves mapping each bit to a pixel value between 0 and 255. When a pixel's value is 0, the corresponding bit value is 0, and when it's 1, the corresponding bit value is 255. The bits are transformed into a grayscale image by first arranging the n-pixel dots in a matrix. Then this matrix is transformed into a specific image by allocating appropriate grayscale pixel value to the image. The image structure is preserved by performing the pixel allocation row-by-row (Bensaoud et al., 2020). After attaining the image, additional image enhancement is done by noise reduction and image resizing.

5. Result analysis and discussion

To evaluate the performance of MD-ADA, we compare it with a TL approach based on fine-tuning applied to the images obtained from the PE binaries. The models are evaluated for both homogeneous and heterogeneous feature spaces. The accuracy and F1-score metrics are used to evaluate the performance:

- **Accuracy:** The accuracy measures the overall correctness of our classification model by considering both true positive and true negative predictions:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}$$

- **Precision:** Precision quantifies the ratio of correctly predicted positive observations to the total predicted positives:

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **Recall (Sensitivity):** Recall calculates the ratio of correctly predicted positive observations to the total actual positives:

$$\text{Recall} = \frac{TP}{TP + FN}$$

- **F1-score:** The F1-score is the harmonic mean of precision and recall, providing a balanced evaluation of the model's performance:

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Here, the terms are defined as follows:

- **TP (True Positive):** Instances correctly predicted as positive.
- **FP (False Positive):** Instances incorrectly predicted as positive.
- **TN (True Negative):** Instances correctly predicted as negative.
- **FN (False Negative):** Instances incorrectly predicted as negative.

Fine-Tuning Approach.

The fine-tuning approach uses the CNN-based ResNet-50 model (Koonce, 2021) trained on the source dataset and then fine-tuned on the target samples for approximately 500 epochs. We first train the ResNet-50 with source data using Adam optimizer with a learning rate of 0.0005. Then we retrain the last two fully connected layers of ResNet50 and freeze the top layers to allow the gradient to only back-propagate through the fully-connected layers. The back-propagation was restricted to the fully-connected layers to learn patterns of the discriminative convolution layers. We fine-tune with the target data at a much lower learning rate of 0.0001 using Adam optimizer. For testing the fine-tuned model, we randomly select 200 samples each (not included in the training set) from both source and target data.

Table 2
Results for the homogeneous feature space scenario.

Samples used for training from Target Dataset	Accuracy		F1-Score	
	Fine Tuning-Target	MD-ADA-Target	Fine Tuning-Target	MD-ADA-Target
100	0.59	0.865	0.47	0.845
200	0.67	0.865	0.60	0.847
500	0.72	0.857	0.69	0.853
800	0.747	0.879	0.736	0.859
1000	0.762	0.881	0.764	0.864
2000	0.791	0.885	0.784	0.867
3500	0.835	0.893	0.8413	0.875

5.1. Homogeneous feature space

Datasets. From a total of 25 malware families in the Malevis Dataset, 15 were selected as the source classes, and 10 as the target classes. It is done for the purpose of testing the outcomes on the source as well as the target for homogeneous feature space data. In comparison, the target data only has a total of 3,875 samples whereas the source data consists of 5,814 samples. The images of different sizes are included in the dataset ranging from 224 to 300 pixels. These images were resized to 224*224 width for the model. We model the malware classification as binary i.e. predicting whether the image samples of the source data belong to malware or normal category.

Experiments. MD-ADA has been trained on the target samples up to 450 epochs for 100, 200, 500, 800, 1000, 2000, and 3,500 target samples since we have only 3,875 samples in the target. The learning rate (α) for performing the experiments for homogeneous feature space was set at 0.001. The training time was 4950 seconds. The main generator was fed with the array 1*512 generated from the CNN model architecture representing the images of source and target generators.

Performance results on the source dataset. The experiments were performed to test the model on the source data and achieved an accuracy rate of 92% and an F1-score of 0.916 on the source data images. The results show that MD-ADA outperforms the fine-tuning approach in accuracy by 23% on 100 samples and 8.5% on 3,500 samples (maximum sample images). A significant difference in F1-score is observed when tested for 100 data samples obtaining a score of 0.47 with fine-tuning approach and 0.81 with MD-ADA. In the case of 3500 target samples, MD-ADA has a gain of 7.47% in the F1-score.

Performance results on the target dataset. When observed on the target test samples for the variable training samples fed to the model, the model is able to achieve a considerable accuracy of 89.3% for 3,500 target training samples. Fig. 5 shows that the model achieves a huge gain of 27.5% on 100 samples and improves constantly with an increase in the target training samples achieving the maximum accuracy of 89.3% which is 5.8% better than the accuracy achieved from the fine-tuning approach. The F1- score of 87.5% has a positive effect on the performance of the target task, leading to more precise and reliable predictions in a scenario of minimal labeled samples. When compared with fine-tuning approach, the gain in F1-score observed is 3.37% for 3,500 samples.

5.2. Heterogeneous feature space

Datasets. For different feature spaces, the tests have been conducted on image representations of malware using Malevis as the source data, which has 5,814 samples, and the Microsoft-Kaggle dataset, which contains 1,099 image samples, as the target data. For evaluation, the target

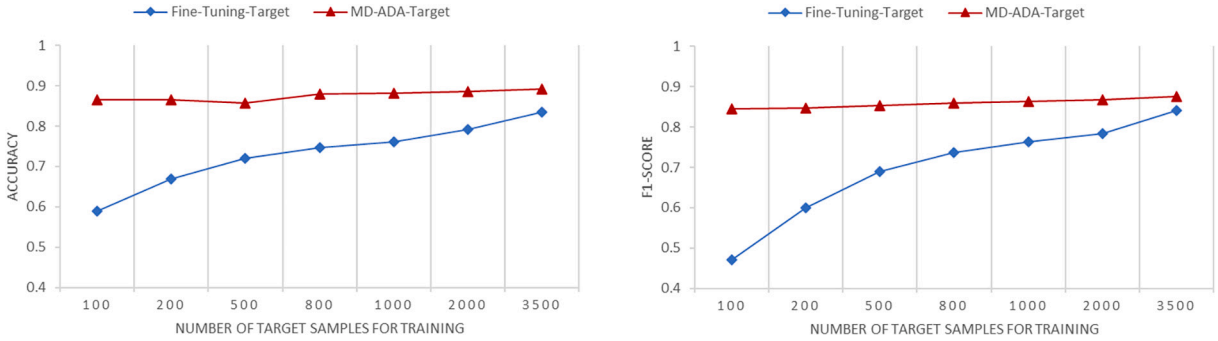

Fig. 5. Performance Analysis for the homogeneous feature space scenario.

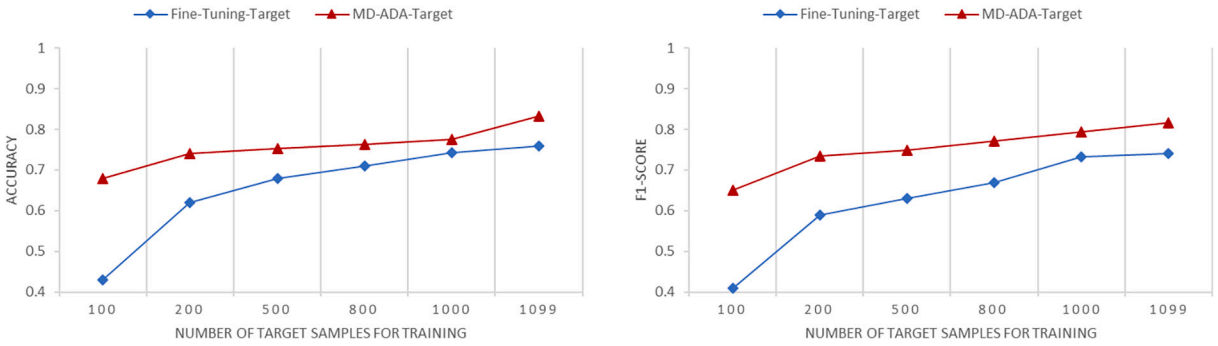

Fig. 6. Performance Analysis for the heterogeneous feature space scenario.

Table 3
Results for the heterogeneous feature space scenario.

Samples used for training from Target Dataset	Accuracy		F1-Score	
	Fine Tuning-Target	MD-ADA-Target	Fine Tuning-Target	MD-ADA-Target
100	0.43	0.68	0.41	0.65
200	0.62	0.741	0.59	0.735
500	0.68	0.7528	0.63	0.749
800	0.71	0.763	0.67	0.771
1000	0.742	0.776	0.732	0.794
1099	0.759	0.8323	0.741	0.817

data contains a various number of samples, and the source data contains 5,814 images. We performed experiments for 100, 200, 500, 800, 1000, and 1,099 samples from the target dataset.

Experiments. We prepared the images using our CNN architecture to a 1*512 embedding from the source and target generators and then applied the main generator for classification and domain prediction. The source and target data were used to train the model and tested the source images and target images with varying target samples. We feed the data for the source (Malevis) and target (Microsoft-Kaggle) to MD-ADA for classifying the sample images as malware or normal. As the fine-tuning approach works only for the data belonging to the same feature space, we resized the source and target data to the same dimension.

Performance results on the source dataset. MD-ADA achieves good results on image datasets compared with the fine-tuning approach with the accuracy on the source being 88% and F1-score 0.866 as depicted in Table 3. Fig. 6 presents the results on the accuracy and F1-score for the different feature space scenarios. The MD-ADA framework achieves a 0.004% gain in accuracy and 0.045% in F1-score on 100 target samples as compared to the fine-tuning approach. This proves MD-ADA is suitable for detecting the target samples and performs well for unseen malware and from the different feature sets.

Performance results on the target dataset. Conducting experiments on the target samples proved to be the most challenging aspect of the experiment due to the limited quantity of samples available. The observed accuracy for the different feature space scenario achieved is 68% which is 2.5% better than the fine-tuning approach for 100 target samples. When fed with maximum samples the model achieved an accuracy of 83.23% being 7.3% better than the accuracy by fine-tuning approach. The model's capacity to adapt and learn from the available data may be restricted by inadequate initial sample sizes of 100 and 200 samples. Incrementally increasing the sample size can be useful as the model gradually adjusts and gains knowledge from additional data points. The challenge of achieving generalization during fine-tuning arises due to the domain shift between the two domains, leading to a decreased accuracy of the fine-tuning approach. The F1-score achieved by MD-ADA on target data on 1,099 training samples is observed to be 81.7% being 7.6% better than the fine-tuning approach. Domain discrepancies between source and target might cause imperfect adaptation, reducing precision and recall. This can be the reason for the lower fine-tuning F1-score.

6. Performance evaluation on BODMAS and Malmem-2022

The model uses the BODMAS dataset (BODMAS, 2023), which includes binaries, feature vectors, and metadata for 57,293 malware files and 77,142 benign Windows PE files, to train the model on homogeneous data. The BODMAS data consists of 2,381 discrete features. Duplicate and missing values are checked for in the data. We further looked for the presence of a set of defining features. We used a probability-based sampling method to choose 90k samples for the source data and 44,435 samples for the target data. With regards to heterogeneous data sources, We selected the CIC-MalMem-2022 (CIC-MalMem-2022, 2023) target dataset containing 55 features and the BODMAS dataset as our source data due to the abundance of malware samples it included. There are a total of 58,596 records in the database, with an even split between the two categories (29,298 records). We selected a quarter of both the source and target datasets for our tests.

Table 4
Result for Homogeneous feature space-BODMAS.

Data	no. of samples	F1-Score	Accuracy (class)	Loss	Accuracy (Domain)
Source	90 K	0.9913	0.9929	0.8575	0.4780
Target	44,435	0.8996	0.9507	0.8575	0.3637

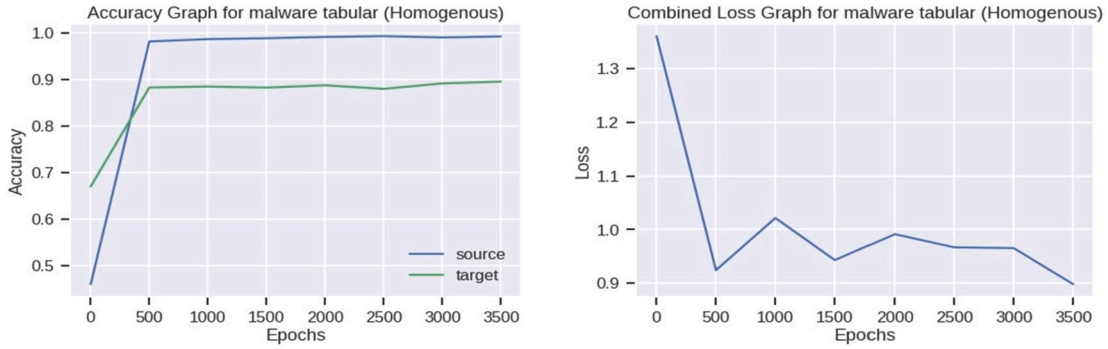

Fig. 7. Accuracy and loss rate for homogeneous tabular (non-image) data.

Table 5
Result for heterogeneous feature space.

Data	no. of samples	F1-Score	Accuracy (class)	Loss	Accuracy (Domain)
Source	1,34,435	0.9127	0.9012	0.96	0.4522
Target	58,596	0.8764	0.8714	0.96	0.4203

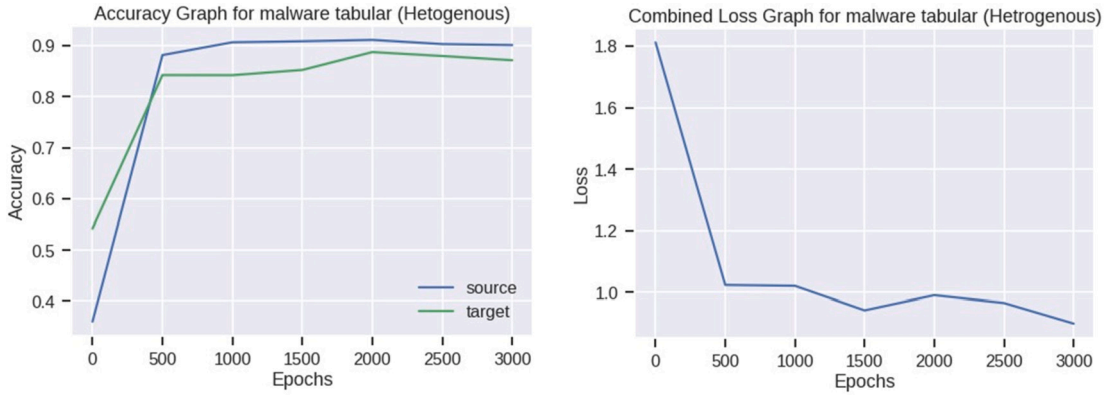

Fig. 8. Accuracy and loss rate for heterogeneous Feature space: CICMalmem-2022 and BODMAS.

We present the results for both homogeneous and heterogeneous feature spaces as discussed below:

As a result of the fact that we had a significant quantity of tabular data, we discovered that it was feasible for the model to be executed throughout a substantial range of epochs. The conclusions drawn from the BODMAS dataset in tabular form. The total number of epochs for the experiments was 3500, and by the time they were through, we had reached an accuracy of 99.29% for the source and 95.07% for the target, respectively is shown in Table 4. The F1-score of 99.13 and 89.9, which demonstrates how effectively our model processes data that is imbalanced across classes. Fig. 7 shows the accuracy curve for source and target set and the loss rate of the model for 3500 epochs. The CIC-Mal-mem dataset had 58,596 samples that were intended to be used as target data, whereas the BODMAS dataset comprised 57,293 malicious software samples and 77,142 benign software samples to be used as source data. When using a learning rate of 0.001 and doing experiments for up to 3500 epochs, a satisfactory accuracy of 90.1% is reached on the source data, and 87% is achieved on the target samples as presented in Table 5. The F1-score of 0.91 on the source, and 0.87 on the target, respectively, demonstrates the usefulness of the model across the various feature spaces. In Fig. 8 the curves for accuracy (left image) and loss rates (image on right) are depicted for 3000 epochs.

7. Related works

Approaches have been recently proposed that apply TL techniques and also the fusion of GANs in different security application domains.

Singla et al. (2020) proposed a GAN-based DA method to train DL classification models for network intrusion detection with minimal labeled data in the target domain. In this approach, the features required for training the model were selected using the principal component analysis (PCA). However, the use of PCA has some drawbacks, namely it requires information to be normalized and also to trade-off information loss versus dimensionality.

To address such issues, MD-ADA does not rely on PCA. Instead, MD-ADA uses a CNN architecture that consists of max-pooling layer (of size

4*4) to deal with dimensionality reduction in images. The dimensions are reduced based on the size and stride of the pooling regions. Selecting the maximum values within each pooling region helps to extract the most dominant features from images. It reduces the width and height of the feature map while preserving the most important information.

A relevant approach related to malware is the Xception model (Lo et al., 2019), which uses a DeepCNN to categorize malware families. The Xception model is available on Keras and has been pre-trained on ImageNet data. Lo et al. (2019) utilized the model to train on the Malimg and Microsoft kaggle datasets. They use a three-layer ensemble model that does not involve feature engineering. Their model achieves a reduced log loss and an accuracy of 99.94% on predictions from '.bytes' and '.asm' files. The major drawback of Xception is that it does not work for unseen malware datasets. To address such drawback, Kim et al. (2017) proposed an approach that uses a transferable deep convolutional GAN to detect zero-day malware.

A deep autoencoder is used to learn the detector, which captures malware characteristics from actual and synthetic data for stabilizing discriminator training.

The model was utilized to analyze a Microsoft-Kaggle dataset that had been converted into distorted-size malware images for validation purposes. The model was able to achieve a detection accuracy rate of 95.74%. However, auto-encoders used for dimension reduction often fail to learn the important features because of directly passing the input to the output layer. Deep autoencoders are prone to overfitting, especially when the training dataset is small or noisy, whereas, MD-ADA proves to be an efficient model when trained with minimal labeled data. By using binary reconstruction, malicious functionality can probably be identified. This indicates using a technique known as "DeepReflect" (Downing et al., 2021) to find malware in software binaries comprehending their inherent functioning by employing deep learning techniques. The aforementioned procedure encompasses the extraction of patterns, features, and behaviors from binary code identifying benign and malicious software. To achieve effective generalization, deep learning techniques typically necessitate a substantial quantity of diverse and accurately labeled training data.

The aforementioned techniques necessitate a significant amount of annotated data to effectively train the model and produce desirable outcomes. Also, the model might underperform in the case when it is fed with different domain data, whereas the ADA technique used in MD-ADA overcomes such limitation by performing domain adaptation requiring only a small amount of labeled data for training.

Traditional ML algorithms are unable to withstand domain shifts and perform poorly when data distribution changes during DA. Wang et al. (2022) proposed an unsupervised DA model for malware detection by correlating the malware data distribution for known and unknown samples to address this issue. Their approach consists of two stages: the first involves minimizing the distribution divergence between domains through adversarial learning on the extracted features, and the second involves aligning the pseudo-labeled target domain and class centers of the labeled source domain to extract more information from the unlabeled target domain, thereby minimizing class-level data divergence. In the residual network, the self-attention mechanism is employed to extract more accurate features. The model's success may be compromised if the distributions diverge or if there are significant differences between known and unknown malware samples, unlike MD-ADA which can perform in both similar and different feature distribution scenarios.

Malware detection in devices running on Android using ML techniques is based on the API features and functions used by the malware. The features of an API include features like the API's call sequences and access controls Fu et al. (2021). The source code and comments present are sometimes disregarded during the examination of Android APK, yet they are a vital step in spotting android malware. Therefore, Huang et al. (2022) introduced AndroidSEM, which is focused on a transformational architecture that provides source code comments for pretraining the framework and optimizing using GANs. The linear regression model is used to get high-quality feedback related to semantic improvement. After that, the Quantum SVM is used to categorize the malicious android code with a classification accuracy of 99.01% with the use of quantum feature mapping. The model performs poorly when target classes overlap due to noise and in spaces with a high number of dimensions. The number of training instances available may not be enough to accurately represent the underlying data distribution (Gupta et al., 2016). Therefore, MD-ADA is best suitable for such scenarios where it is showing good accuracy in minimal labeled situations. Alotaibi's 2022 suggested a multidimensional DeepGAN model to identify mobile malware with an average accuracy of 96.2%. The pattern and sequence information provided by the API is pixel-to-pixel processed by conditional GAN inside a multi-face framework that also makes use of a hybrid GoogleNet and LSTM. GoogleNet employs nine inception modules, which is the maximum allowed by the framework, to achieve optimal success in deep learning. As a model's parameters increase, its susceptibility to overfitting increases. The parameter explosion at the inception levels is undesirable and requires the model to reduce feature representations for the target variable to accommodate within the available feature space. To encounter this, autoencoders aid in lowering the feature dimension that may be used to distinguish malicious software from benign software. Based on the autoencoder's reconstruction error, the approach by Xing et al. (2022) for DL-based malware detection evaluates the efficacy of the malware image representation. The malware was identified from benign training using the multi-layer perceptron (MLP) learned by extracting high-dimensional characteristics and using the output of the pre-trained auto-encoder-1 (AE1) with 96% accuracy. However, Smmarwar et al. (2022) developed a three-stage deep learning-based malware detection system for IoT-based smart agriculture that makes advantage of the fusion of GAN and discrete wavelet transform to overcome the constraint of detecting malware variants (DWT). Characteristics are retrieved using discrete wavelet transform, and then those features are used by a GAN to identify malicious software. In order to improve the classification accuracy, the DWT extracts the approximation coefficients, and the detail coefficients and associates them with the generator and the discriminator, respectively, to use the image fusion in the GAN model. Multi-class malware family identification is achieved by using a lightweight CNN model. While some of the prior research employs the concept of expressing the malware as images, few address the problem of domain adaptability and the availability of low-labeled data for training, limiting their usefulness for malware detection. To the authors' knowledge, no prior works have attempted to tackle the malware detection problem using a unified framework that considers malware images, low-labeled training data, and domain adaptation, without the need of PCA for handling high dimensions.

8. Comparison with state-of-art approaches

To show the efficacy of the model, MD-ADA approach is compared with the state-of-art approaches depicted in Table 6. The study conducted by Yang et al. (2021) presents DeepMal that aims to maintain harmful attributes while training models for static malware detection through adversarial instruction learning. DeepMal utilizes adversarial training to produce adversarial samples that preserve malicious characteristics while evading static detection algorithms. The work seeks to overcome the constraints of current static malware detection techniques by developing models that can accurately distinguish between benign and malicious samples.

Muneer et al. (2022) propose a novel method that combines deep learning techniques to find anomalies in high-dimensional data without the need for labeled examples. The work aims to efficiently detect anomalies in datasets without relying on labeled training samples. The suggested approach integrates deep learning approaches to tackle the difficulties given by high-dimensional data. The authors provide a comprehensive description of the hybrid deep learning model and its uti-

Table 6
Comparison of MD-ADA Approach with state-of-art approach.

Model	F1-Score	Accuracy	PCA needed
MD-ADA (Source)	0.9127	0.9012	×
MD-ADA (Target)	0.8764	0.8714	×
DANN +SGD (Muneer et al., 2022)	88.23	86.99	✓
DeepMal Resnet Classifier (Yang et al., 2021)	0.9215	0.9213	✓

lization in anomaly detection. The objective of the work is to improve the identification of irregularities in intricate datasets by employing a fusion of advanced deep learning and PCA.

The MD-ADA framework exhibits robust performance in both the source and destination domains. MD-ADA demonstrates a high level of performance in the source domain, achieving an F1-Score of 0.9127 and an accuracy of 0.9012 on the BODMAS dataset. In the target domain, it maintains strong performance with an F1-Score of 0.8764 and an accuracy of 0.8714. It is important to highlight that PCA is not necessary for MD-ADA in any area, highlighting its effectiveness in managing high-dimensional data without reducing its dimensions. The DANN with SGD model, demonstrates competitive performance, achieving an F1-Score of 88.23 and an accuracy of 86.99. Nevertheless, it necessitates the use of PCA, indicating a dependence on dimensionality reduction methods. Similarly, the DeepMal Resnet Classifier (Yang et al., 2021) achieves remarkable performance, with an F1-Score of 0.9215 and an accuracy of 0.9213. However, it also requires the use of PCA. The inclusion of PCA in these models necessitates a balance between performance and the intricacy resulting from the reduction of data dimensionality. MD-ADA is noteworthy for its efficacy in both source and target domains without the need for PCA.

9. Conclusion

The lack of data from a wide range of malware domains is a major limitation for supervised learning to succeed when applied to malware detection. There is a chance that a supervised learning model falsely identifies a threat because of the wide variety of obfuscation techniques and lack of data used to train it. Due to the growing sophistication of malware evasion strategies, it is essential to have accurate models that can learn and expand over time when there is an increase in the amount of unknown data. In order to detect novel forms of malware, researchers have proposed GAN-based models, trained using deep learning methods to learn and evolve over time.

In order to build a powerful discriminator and classifier that can identify threats in real-time with greater accuracy than a supervised learning model, we proposed a GAN framework that uses images of malware as input. To narrow the gap between domains, we use domain adaptation for training three generators and a discriminator. Our experimental results show that our framework has superior performance in identifying malware based on its image representations. The measures of accuracy and F1-score show that our proposed model works best with low or no labeled data in both homogeneous (uniform) and heterogeneous feature spaces. The key differentiator of our model is that the model does not require any PCA or dimension reduction technique to filter the features for model training. MD-ADA is highly effective in both the source and target domains, even when compared with state-of-art approaches delivering impressive performance without the need for PCA.

In future work, we plan to consider malware data from systems running Android, the internet of things (IoT), and Linux and use this data for further experiments. Such experiments will allow us to further evaluate the flexibility of our framework when the source organization and the target organization uses different data distribution formats. However, this can be a challenging task to address. Also, when malware data is made accessible, there is the possibility that data could be leaked to attackers. We will thus investigate the use of data-preserving DA techniques to prevent source data leakages.

CRediT authorship contribution statement

Sonam Bhardwaj: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Adrian Shuai Li:** Conceptualization, Formal analysis, Investigation, Methodology, Supervision, Validation, Writing – original draft, Writing – review & editing. **Mayank Dave:** Conceptualization, Investigation, Supervision, Visualization, Writing – original draft, Writing – review & editing. **Elisa Bertino:** Investigation, Project administration, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgement

The authors gratefully acknowledge the support from Science and Engineering Research Board (SERB), India through the SERB-OVDF 2022 Program, and the support of the National Science Foundation under grant No. 2229876.

References

Dataset link [BODMAS] for homogenous tabular; Available at: https://github.com/whyisyoung/BODMAS. (Accessed 9 December 2023).
Dataset link [CIC-MalMem-2022] for heterogenous; Available at: https://www.unb.ca/cic/datasets/malmem-2022.html. (Accessed 9 December 2023).
Image Homogenous and Heterogenous Source Data. Retrieved May 19, 2023 from https://web.cs.hacettepe.edu.tr/selman/malevis/.
Image Heterogenous Target Data. Retrieved May 19, 2023 from https://www.kaggle.com/datasets/matthewfields/malware-as-images.
Alotaibi, Fahad Mazaed, 2022. A multifaceted deep generative adversarial networks model for mobile malware detection. Appl. Sci. 12 (19), 9403. https://doi.org/10.3390/app12199403.
Anderson, Blake, Storlie, Curtis, Yates, Micah, McPhall, Aaron, 2014. Automating reverse engineering with machine learning techniques. In: Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop, pp. 103–112.
Bensaoud, Ahmed, Abudawaood, Nawaf, Kalita, Jugal, 2020. Classifying malware images with convolutional neural network models. Int. J. Netw. Secur. 22 (6), 1022–1031. https://doi.org/10.6633/IJNS.202011_22(6).17.
Bhagat, Reshma C., Patil, Sachin S., 2015. Enhanced SMOTE algorithm for classification of imbalanced big-data using random forest. In: 2015 IEEE International Advance Computing Conference (IACC), pp. 403–408.
Bhodia, Niket, Prajapati, Pratikkumar, Di Troia, Fabio, Stamp, Mark, 2019. Transfer learning for image-based malware classification. arXiv preprint. arXiv:1903.11551. https://doi.org/10.48550/arXiv.1903.11551, 2019.
Cui, Zhihua, Xue, Fei, Cai, Xingjuan, Cao, Yang, Wang, Gai-ge, Chen, Jinjun, 2018. Detection of malicious code variants based on deep learning. IEEE Trans. Ind. Inform. 14 (7), 3187–3196. https://doi.org/10.1109/TII.2018.2822680.
Darem, Abdulbasit, Abawajy, Jemal, Makkar, Aaisha, Alhashmi, Asma, Alanazi, Sultan, 2021. Visualization and deep-learning-based malware variant detection using OpCode-level features. Future Gener. Comput. Syst. 125, 314–323. https://doi.org/10.1016/j.future.2021.06.032.
Downing, E., Mirsky, Y., Park, K., Lee, W., 2021. DeepReflect: discovering malicious functionality through binary reconstruction. In: 30th USENIX Security Symposium (USENIX Security 21), pp. 3469–3486.
Fu, Zhangjie, Ding, Yongjie, Godfrey, Musaazi, 2021. An LSTM-based malware detection using transfer learning. J. Cybersecurity 3 (1), 11. https://doi.org/10.32604/jcs.2021.016632.
Gosain, Anjana, Sardana, Saanchi, 2017. Handling class imbalance problem using oversampling techniques: a review. In: 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 79–85.

Gupta, Parth, Banchs, Rafael E., Rosso, Paolo, 2016. Squeezing bottlenecks: exploring the limits of autoencoder semantic representation capabilities. Neurocomputing 175, 1001–1008. https://doi.org/10.1016/j.neucom.2015.06.091.

Huang, Yizhao, Li, Xingwei, Qiao, Meng, Tang, Ke, Zhang, Chunyan, Gui, Hairen, Wang, Panjie, Liu, Fudong, 2022. Android-SEM: generative adversarial network for Android malware semantic enhancement model based on transfer learning. Electronics 11 (5), 672. https://doi.org/10.3390/electronics11050672.

Jian, Yifei, Kuang, Hongbo, Ren, Chenglong, Ma, Zicheng, Wang, Haizhou, 2021. A novel framework for image-based malware detection with a deep neural network. Comput. Secur. 109, 102400. https://doi.org/10.1016/j.cose.2021.102400.

Kim, Jin-Young, Bu, Seok-Jun, Cho, Sung-Bae, 2017. Malware detection using deep transferred generative adversarial networks. In: International Conference on Neural Information Processing. Springer, Cham, pp. 556–564.

Koonce, Brett, 2021. ResNet 50. Convolutional Neural Networks with Swift for Tensorflow: Image Recognition and Dataset Categorization, pp. 63–72.

Lo, Wai Weng, Yang, Xu, Wang, Yapeng, 2019. An xception convolutional neural network for malware classification with transfer learning. In: 2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS), pp. 1–5.

Lu, Yan, Li, Jiang, 2019. Generative adversarial network for improving deep learning based malware classification. In: 2019 Winter Simulation Conference (WSC), pp. 584–593.

Moser, Andreas, Kruegel, Christopher, Kirda, Engin, 2008. Limits of static analysis for malware detection. In: Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007), pp. 421–430.

Muneer, A., Taib, S.M., Fati, S.M., Balogun, A.O., Aziz, I.A., 2022. A hybrid deep learning-based unsupervised anomaly detection in high dimensional data. Comput. Mater. Continua 70 (3).

Nataraj, Lakshmanan, Karthikeyan, Sreejith, Jacob, Gregoire, Manjunath, Bangalore S., 2011. Malware images: visualization and automatic classification. In: Proceedings of the 8th International Symposium on Visualization for Cyber Security, pp. 1–7.

Ni, Sang, Qian, Quan, Zhang, Rui, 2018. Malware identification using visualization images and deep learning. Comput. Secur. 77, 871–885. https://doi.org/10.1016/j.cose.2018.04.005.

OpenCV Python Library, 2023. Retrieved May 19, 2023 from https://pypi.org/project/opencv-python/.

Pan, Sinno Jialin, Yang, Qiang, 2009. A survey on transfer learning. IEEE Trans. Knowl. Data Eng. 22 (10), 1345–1359. https://doi.org/10.1109/TKDE.2009.191.

Shaid, Syed Zainudeen Mohd, Maarof, Mohd Aizaini, 2014. Malware behavior image for malware variant identification. In: 2014 International Symposium on Biometrics and Security Technologies (ISBAST), pp. 238–243.

Singla, Ankush, Bertino, Elisa, Verma, Dinesh, 2020. Preparing network intrusion detection deep learning models with minimal data using adversarial domain adaptation. In: Proceedings of the 15th ACM Asia Conference on Computer and Communications Security, pp. 127–140.

Smmarwar, Santosh K., Gupta, Govind P., Kumar, Sanjay, 2022. Deep malware detection framework for IoT-based smart agriculture. Comput. Electr. Eng. 104, 108410. https://doi.org/10.1016/j.compeleceng.2022.108410.

Sonic Wall Threat Report, 2023. Retrieved on May 19, 2023 from https://www.sonicwall.com/medialibrary/en/white-paper/2023-cyber-threat-report.pdf.

Wang, Fangwei, Chai, Guofang, Li, Qingru, Wang, Changguang, 2022. An efficient deep unsupervised domain adaptation for unknown malware detection. Symmetry 14 (2), 296. https://doi.org/10.3390/sym14020296.

Xing, Xiaofei, Jin, Xiang, Elahi, Haroon, Jiang, Hai, Guojun, Wang, 2022. A malware detection approach using autoencoder in deep learning. IEEE Access 10, 25696–25706. https://doi.org/10.1109/ACCESS.2022.3155695.

Yang, C., Xu, J., Liang, S., Wu, Y., Wen, Y., Zhang, B., Meng, D., 2021. DeepMal: maliciousness-preserving adversarial instruction learning against static malware detection. Cybersecurity 4, 1–14.

Zhou, Zhili, Lin, Kunde, Cao, Yi, Yang, Ching-Nung, Liu, Yuling, 2020. Near-duplicate image detection system using coarse-to-fine matching scheme based on global and local CNN features. Mathematics 8 (4), 644. https://doi.org/10.3390/math8040644.

Sonam Bhardwaj earned her B.Tech degree from Lingayas University, Faridabad, in 2013, followed by an M.Tech degree in Computer Science in 2016. With one year of research experience in Big Data Deduplication, she furthered her expertise as a visiting scholar at Purdue University from 2022 to 2023. During this period, her focus extended to Transfer Learning in Security and Attack Detection utilizing attack graphs. Currently pursuing her Doctorate in Network Forensics, Sonam boasts 5 years of rich experience in the field. Her areas of interest encompass investigating network attacks, graphical analysis of attack graphs, machine learning, cybersecurity, and the management of evidence in IoT. As a committed professional, she serves as a reviewer for several reputable journals and holds the esteemed status of being a student member of ACM and IEEE.

Adrian Shuai Li is a Ph.D. student in Computer Science at Purdue University, advised by Professor Elisa Bertino. Prior to that, he was a researcher at the Institute for Security, Privacy, and Information Assurance (ISPIA) at the University of Calgary from 2020 to 2021, advised by Professor Reihaneh Safavi -Naini. His research interests include Security, Privacy, and Artificial Intelligence/Machine Learning. In particular, his research focuses on building effective cyber security defenses using AI-based technologies. He received an M.Sc. from the University of Calgary, where he was a recipient of the Mitacs Globalink Graduate Fellowship. Before that, he conferred his bachelor's degree in computer science from Wuhan University. In the past, he has interned at TELUS Research, Aviatrix Systems and Cisco Research.

Mayank Dave is the Professor in the Department of Computer Engineering at National Institute of Technology Kurukshetra (NIT Kurukshetra), India with more than 32 years' experience of teaching and research. He received B.Tech degree from Aligarh Muslim University, Aligarh, India in 1989, MTech degree in Computer Science and Technology and the PhD degree from IIT Roorkee, India in 1991 and 2002, respectively. He has published approximately 200 research papers in various international/national journals and conferences. He has coordinated several research and development projects in the department and institute. He has written a textbook titled "Computer Networks" published by Cengage Learning, India. Prof. Mayank Dave has guided fifteen PhDs and currently guiding few more PhDs. His research interests include mobile networks, cyber security, cloud computing, software-defined networking, web technologies etc. He is a Senior Member of the IEEE and the IEEE Computer Society, and member of the ACM, IETE, Computer Society of India, and Institution of Engineers (India).

Elisa Bertino is the Samuel D. Conte Full Professor of Computer Science at Purdue University. Before joining Purdue, she was a Full Professor at the University of Milano. She has been working on data security and privacy for more than 35 years. More recently she has been working on security and privacy on 4GLTE and 5G cellular networks, security of mobile applications and IoT systems, zero-trust architectures, and machine learning techniques for cybersecurity. She is a Fellow member of IEEE, ACM, and AAAS. She received the 2002 IEEE Computer Society Technical Achievement Award for "For outstanding contributions to database systems and database security and advanced data management systems", the 2005 IEEE Computer Society Tsutomu Kanai Award for "Pioneering and innovative research contributions to secure distributed systems", and the 2019-2020 ACM Athena Lecturer Award.