


Article

# MLP-Mixer-Autoencoder: A Lightweight Ensemble Architecture for Malware Classification

Tuan Van Dao <sup>\*</sup> , Hiroshi Sato and Masao Kubo

Department of Computer Science, National Defense Academy of Japan, Yokosuka 239-8686, Japan

<sup>\*</sup> Correspondence: tuan2011nda@gmail.com

**Abstract:** Malware is becoming an effective support tool not only for professional hackers but also for amateur ones. Due to the support of free malware generators, anyone can easily create various types of malicious code. The increasing amount of novel malware is a daily global problem. Current machine learning-based methods, especially image-based malware classification approaches, are attracting significant attention because of their accuracy and computational cost. Convolutional Neural Networks are widely applied in malware classification; however, CNN needs a deep architecture and GPUs for parallel processing to achieve high performance. By contrast, a simple model merely contained a Multilayer Perceptron called MLP-mixer with fewer hyperparameters that can run in various environments without GPUs and is not too far behind CNN in terms of performance. In this study, we try applying an Autoencoder (AE) to improve the performance of the MLP-mixer. AE is widely used in several applications as dimensionality reduction to filter out the noise and identify crucial elements of the input data. Taking this advantage from AE, we propose a lightweight ensemble architecture by combining a customizer MLP-mixer and Autoencoder to refine features extracted from the MLP-mixer with the encoder-decoder architecture of the autoencoder. We achieve overperformance through various experiments compared to other cutting-edge techniques using Maling and Malheur datasets which contain 9939 (25 malware families) and 3133 variant samples (24 malware families).

**Keywords:** Malware classification; MLP-mixer; autoencoder; information security



**Citation:** Dao, T.V.; Sato, H.; Kubo, H. MLP-Mixer-Autoencoder: A Lightweight Ensemble Architecture for Malware Classification. *Information* **2023**, *14*, 167. <https://doi.org/10.3390/info14030167>

Academic Editor: Jiguo Li

Received: 18 January 2023

Revised: 11 February 2023

Accepted: 3 March 2023

Published: 6 March 2023



**Copyright:** © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

In recent years, malware has become a significant threat to security in cyberspace. New variants of malware are constantly appearing to challenge antivirus companies. More dangerous, several attackers and organizations have abused malware to attack businesses, governments, financial institutes, health providers, etc. [1]. According to the AV-TEST Institute [2], more than 450,000 new malicious programs and potentially unwanted applications are registered daily, and over 1356 million malware were discovered in May of 2022. This number is seven times higher than ten years ago.

Traditional malware classification works are based on signature and behavior approaches [3]. This approach is fragile in address with polymorphic and metamorphic malware. Moreover, because of the rapid growth of the development of several automatic malware creation tools [4], these methods cannot catch up to the speed of malware generation.

Machine learning has become more potent because its highly developed algorithms can solve most problems encountered in almost every field. There are two main approaches: Natural Language Processing (NLP) and Computer Vision. Several studies use NLP to classify malicious files by extracting features from malicious software, such as API Calls, Windows Registry, and Strings [5–7]. However, malware writers often use packing or obfuscation to make their files more difficult to classify or analyze. In addition, this malware contains very few strings, tokens, and other information that is not readable. As a result, NLP-based approaches face difficulty in collecting features of malware.

Another alternative to the machine learning-based method for malware classification is to employ computer vision approaches on file binaries [8–23]. Although attackers use obfuscation techniques (such as encryption, packing, metamorphism, and polymorphism) to achieve spoofing, malware variants from the same family still maintain similar code and data order, which may not appear in the exact location. On the other hand, after packing with the same packer, the images of malware variants belonging to different families were different [24], and the unpacked variants of different families were utterly different from their packed variants [13]. Conti et al. [8] have proposed a method to visualize malware binaries into a grayscale image and noticed that visual analyses of malware binary help distinguish various data regions from the image. The advantage of the malware visualization analysis is that it does not require using any decompilers or a dynamic running environment and does not need to pick out particular statistical features for classification intentionally.

Several studies that apply Convolutional Neural Networks (CNN) to classify malware have achieved high accuracy in standard datasets, such as Malimg and Malheur. Some take advantage of a pre-trained model to enhance the performance of the proposed model [25,26]. The growth of high-performance computing, coupled with the enormous CNNs architectures, made it possible to process images at a higher level of complexity. However, recent studies indicate that fewer parameters with a simple network structure give relatively satisfactory results and can be applied to low-profile devices like IoT [13,27,28], or smartphones [29]. On the other hand, to improve the performance of CNN, optimizing a large number of the hyper-parameters, such as kernel (filter size), padding, stride, and the number of channels, is also a problem with CNN [30]. Although several studies [26,31–33] attempt to apply state-of-the-art CNN models, the performance is still not outstanding.

Recently, Multilayer Perceptron (MLP) has been paid attention again to by the scientific community in the Computer Vision field by modifying traditional MLP. Tolstikhin et al. [34] introduce MLP-mixer with simple architecture based entirely on MLPs. MLP-mixer merely achieves strong performance; however, it still needs to beat some specialized CNN architectures.

Autoencoders are unsupervised deep learning algorithms with a unique neural network structure. An autoencoder (AE) transforms the input into an output with minimal reconstruction errors. They have several applications, such as dimensionality reduction, preprocessing for classification, and identifying only the essential elements of input data.

That has motivated us to empower MLP-mixer by concatenating with AE to gain more prosperous and more selective features of malware samples. While the global averaging layer of MLP-mixer is used to minimize overfitting by reducing the total number of parameters in the model and aggregating all local space in each channel across patches, Autoencoder creates latent dimension space, ensuring only the main structured part of the information can be reconstructed. In our proposed method, Autoencoder plays a role in refining the logical space of each sample after processing through MLP-Mixer.

The main contribution of this paper is providing a lightweight image-based malware classification system through feature synthesis from MLP-mixer and AE. Because the processing is merely dependent on images, the system does not require in-depth knowledge of the malware and the environment to determine its behavior. Moreover, some classifiers can give the result in under a second, so our model can be applied in real-time countermeasures against malware.

The rest of the paper is organized as follows: Section 2 discusses the related work concerning some popular and recent malware classification techniques. Section 3 illustrates the proposed model in detail. Section 4 evaluates the performance of the proposed approach. Finally, we summarize our work in Section 5.

## 2. Related Work

This session investigates various new studies on image-based malware classification with CNN and CNN-free models. In terms of the former, recent studies focus on deep and

complex architecture and combining it with others to enhance performance. In the latter, some alternative candidate is used to replace CNN.

For the first time, Nataraj et al. [10] proposed a novel approach for visualizing and classifying malware using image processing techniques. They visualized malware as a gray-scale image based on the observation that images of the same class were very similar in layout and texture. They utilize the GIST descriptor, based on the wavelet decomposition of an image, as a feature extractor and k-nearest neighbor (kNN) as a classifier. The paper achieved an accuracy of 97.18% on their introduced dataset: Maling, which contains 9339 malware samples related to 25 different malware families.

### 2.1. CNN-Based Models

Hammad et al. [17] used both hand-engineering Tamura and CNN models of Google-LeNet techniques to extract the feature from 2D malware images. The authors employed k-Nearest Neighbor (KNN), Support Vector Machine (SVM), and Extreme Learning Machine (ELM). The authors' proposed method outperformed both existing Malware classifications on the Maling dataset with an accuracy of 95.42% for Tamura feature extraction, followed by the ELM classifier, and GoogleNet with 96.84%, followed by the KNN classifier.

Lin et al. [18] proposed byte-level one-dimensional CNNs to explore informative features from the one-dimensional structure of binary executables. They achieved comparable results with less computational cost than 2D CNNs regarding the amount of multiply-add operations. They achieved a high performance even with small image sizes, such as  $32 \times 32$  and  $64 \times 64$ , with 96.95% and 98.37%, respectively.

Falana et al. [23] utilized an ensemble technique consisting of a Deep Convolutional Neural Network and a Deep Generative Adversarial Neural Network that can generate novel malware from the train malware data-sets to prevent the adversarial attack. Through experiments on the Maling dataset, the authors achieved an overall classification accuracy of 95.63%, a precision of 95.34%, a recall of 95.30%, and an F1 score of 94.98%.

Wang et al. [20] proposed a novel design of Multiscale Attention Adaptive Module (MSAAM) and CliqueNet that can combine local and global multiscale information in the spatial domain of malware images. The authors adjust the gray-scale image size of the model input to  $256 \times 256$ . Experiment results showed that Attention mechanisms improved accuracy slightly by 0.6% from 98.6% to 99.2% on the Maling dataset.

Rezende et al. [11] transferred the first 49 layers of ResNet-50 on ImageNet to the malware classification task. Frozen layers can be seen as learned feature extraction layers. The author replaced the last layer with 1000 fully connected softmax with 25 fully connected ones according to the number of classes on the Maling dataset. After 750 epochs, the paper reached an average accuracy of 98.62% with 10-fold cross-validation. They also compare features extracted from Deep CNN (DCNN) with GIST features using the same kNN classifier. The experimental result showed that ResNet-50 performed better than handcrafted GIST by 0.52% with 98.00% and 97.48%, respectively.

Kumar et al. [28] applied deeper transfer learning with the knowledge from the already trained ImageNet (which contains 1.2 million images with 1000 classes) for the malware classification task; they improved the ResNet50 model by altering the last layer with a fully connected dense layer. They achieved high accuracy of 99.12% with the Adam optimizer and 99.23% with the NAdam optimizer.

Silva et al. [35] inject a sequence of random bytes into non-executable sections such as ".data" into the files. The authors faced challenges when applying this method to some specific instances, usually packed or obfuscated, ensuring that keeping its functionality intact is a challenging task. The usage of CNNs for the image-based approach is increasing, and their work shows how adding small perturbations to a malware file can reduce the accuracy of these CNNs by almost 50%. The authors also declare that not only relying on a fixed preprocessing method more dynamic approach, such as the Attention-based method, can be helpful.

Awan et al. [9] applied spatial convolutional attention called dynamic spatial convolution on VGG19 Network. This attention utilized a global average pooling (GAP) mechanism, rescaled the output of GAP by lambda layer, and fed into dropout of rate 0.25 before fully connected layer, and the author utilized Softmax as a traditional classifier of CNNs. The performance was evaluated on the Malimg dataset and achieved an accuracy of 97.68%.

## 2.2. CNN-Free Models

Barros et al. [19] developed a novel method that captures pairwise information from data pairs based on a new similarity measure designed for the context of malware identification. The authors generate the Siamese Networks input by pairing up the new image with every image of the training dataset. This method overperforms traditional image feature extraction such as Garbor, HOC, and GIST but is lower than SoTa CNN models.

Naeem et al. [13] extracted malware features from the grayscale image through SIFT-GIST Malware (CSGM) description that could reduce computational time and improve classification accuracy. Note that CSGM features consisted of local and global features of the malware image, making authors more informative than the traditional method, as well as simple CNN models extracting local features of the image. They achieved high performance in both Malimg and Malheur datasets with an accuracy of 98.40% and 97.50%, respectively.

Son et al. [22] found that the main texture of the malware image is spread in the vertical orientation so that the horizontal size of the malware image can be reduced without too much computation complexity and training time. The authors' proposed method, which decreases the dimension of the normalized input images, is higher than the GIST-based malware classification model, reducing the computational complexity and saving significant training time. The authors achieved the highest accuracy of 98.49% and 95.79% in Malimg and Malheur datasets with the SVM classifier, respectively.

Ly et al. [36] propose a novel approach to encoding and arranging bytes from binary files into images. The authors take semantic information into account and represent it as color images. They constrain the size of generated images to  $256 \times 1024$ , 8-bit RGB color, and use XGBoost as a classifier. As a result, the accuracy improves by 2.16% from 84.20% to 86.36% with Grayscale, the proposed color encoding method, respectively.

Lee et al. [16] illustrate the effectiveness of autoencoder by applying multiple AEs. Each AE model classifies only one type of malware and is trained using only samples from the corresponding family. As a result, the author achieves an accuracy of 94.03% for a system with the same AE network structure and 97.75% for various AEs. Moreover, the model achieves a 0.46% improvement from 97.75% to 98.21% when combining similar classes.

## 3. Proposed Method

### 3.1. Image Representation for Malware

To visualize a malware sample as an image, we must interpret every byte as one pixel in an image. Notice that binary files are the hexadecimal representation of the PE of malware in Figure 1. The first row is the offset of the memory address. The second one represents the pair of hexadecimal. Each hexadecimal pair is treated as a single decimal number that serves as the image's pixel value, so this conversion is no loss of information. The resulting array must be organized as a 2-D array, and values must be in the range [0, 255] (0: black, 255: white). The size of the image depends on the binary file's size. Table 1 presents different heights for malware images due to the different sizes of malware files while fixing the width of images. Table 1 also illustrates that converting malware into grayscale images does not require a long time; standard malicious codes less than 1 Mb in size only take no more than 0.01 s to convert.

```

00000001`80000000: 4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00
00000001`80000010: B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00
00000001`80000020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000001`80000030: 00 00 00 00 00 00 00 00 00 00 00 00 E0 00 00 00
00000001`80000040: 0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68
00000001`80000050: 69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F
00000001`80000060: 74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20
00000001`80000070: 6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00
00000001`80000080: 6B 55 4B 17 2F 34 25 44 2F 34 25 44 2F 34 25 44
00000001`80000090: 34 A9 8F 44 7C 34 25 44 34 A9 BB 44 26 34 25 44
00000001`800000A0: 26 4C B6 44 2C 34 25 44 2F 34 24 44 7F 34 25 44
00000001`800000B0: 34 A9 8E 44 38 34 25 44 34 A9 BE 44 2E 34 25 44
00000001`800000C0: 34 A9 BF 44 2E 34 25 44 34 A9 B8 44 2E 34 25 44
00000001`800000D0: 52 69 63 68 2F 34 25 44 00 00 00 00 00 00 00 00
00000001`800000E0: 50 45 00 00 64 86 06 00 29 57 14 59 00 00 00 00
00000001`800000F0: 00 00 00 00 F0 00 22 20 0B 02 0A 00 00 7C 00 00
00000001`80000100: 00 58 50 00 00 00 00 00 EC 15 00 00 00 10 00 00
00000001`80000110: 00 00 00 80 01 00 00 00 00 10 00 00 00 02 00 00
    
```

Figure 1. The structure of a binary file.

For preprocessing, we utilize Keras’s ImageDataGenerator library to read images from the dataset. Before inputting to the neural network, it is necessary to align different-sized malware to the same size (e.g., 32 × 32) so the whole loss occurs by discarding the extra part or compressing. We accept this loss for training and evaluating Neural Networks like in previous studies.

On the other hand, new variants are often created by changing a small part of the code. Therefore, if the predecessor is reused, the result would be very similar. Furthermore, it is possible to detect small changes by converting malware into an image while keeping the comprehensive structure of samples belonging to the same family [37].

Table 1. Image height for different malware file sizes.

File Size	Image Height	Time Convert (ms)
<10 kB	32	0.105
10 kB–30 kB	64	0.312
30 kB–60 kB	128	0.428
60 kB–100 kB	256	0.571
100 kB–200 kB	384	0.748
200 kB–500 kB	512	0.665
500 kB–1 Mb	768	0.814
>1 Mb	1024	2.85

### 3.2. MLP-Mixer

Tolstikhin et al. [34] proposed MLP-mixer, a conventional MLP inspired. The authors divided the input image into small mini patches. Unlike in CNNs, the MLP-mixer processes input data on these mini patches through the network. Like in the transformer, each patch fed to the MLP architecture is unrolled to a vector with C channels, and each of these vectors is then stacked upon each other and can be seen as a table. There are two types of MLP-mixer layers: token-mixing MLPs and channel-mixing MLPs. In the former, the table is transposed, and each row is fed into the same MLP layer. All the weights are shared across the same channel of different patches. In the latter, the table is reversed trick again and flipped back into patches, then does the same shared computation for all the patches. Individual rows are processed independently. MLP-mixer learns image features by mixing spatial and channel information obtained by image segmentation for each patch. MLP mixer merely leverages MLP to operate all patches. Skip-connections are used in both token-mixing blocks and channel-mixing blocks to avoid gradient vanishing phenomena, a common problem in Deep Learning during training. The activation function is GELU. MLP-mixer block is repeated n times and then reduced dimension using Global Average Pooling (GAP), followed by a fully connected layer before the classification task.

### 3.3. Autoencoder

The basic structure of an AE consists of three components: the encoder, latent space, and decoder. The encode compresses the input and produces a latent space, preventing the neural network from memorizing and overfitting the data. The encoding output represents the original data's feature while reducing the dimensions by setting the number of neurons in the encoder layer to less than the number in the input layer. Conversely, the decoder reconstructs the input only using this latent space. The autoencoder is trained to make the input and decoding data as similar as possible while never becoming precisely the same. Notice that the autoencoder is a data-specific feature extractor. It can only extract meaningful features from data identical to what it has been trained. We take advantage of these characteristics of the AE to refine features extracted from the MLP-mixer.

### 3.4. MLP-Mixer-AE

Although MLP-mixer has optimized the structure of the Neural network through token mixing and channel mixing blocks, the necessary features to identify malicious code have not been purified; Ref. [34] only uses GAP to summarize the channel through all patches, then fed into a fully-connected layer. It was discovered in [38] that adding a small amount of attention makes the model more efficient. The reason for using attention here is to reselect important features from inputs. Here, instead of using complex attention, we choose autoencoder, an artificial neural network familiar with machine learning in unsupervised learning to refine features after going through the MLP-mixer. The encoder-decoder structure helps us extract the most crucial features from images in the form of data and establish valuable correlations between various inputs within the network.

Our proposed method is illustrated in Figure 2. After resizing the malware image in pre-processing, it is divided into  $S$  patches (we set the patch size as  $8 \times 8$ ). Then patches are normalized in Layer Norm, and the malware image is turned into a table of patches and channels ( $S, C$ ). The table is then transposed ( $C, S$ ) to feed into token-mixing MLPs, with hidden node  $D_s$  along with GELU activation. Afterward, the table is transposed to return to the initial form and aggregate with the initial table before feeding into Channel-mixing MLPs. In these blocks, the hidden node of channel-mixing is  $D_c$ , and the activation is GELU. Same with token-mixing blocks, skip-connections are used to avoid gradient vanishing. MLP-mixer blocks are repeated  $n$  times, followed by the GAP layer. In our implementation, we utilize a lightweight MLP-mixer with  $C = 128$ ,  $D_s = 64$ , and  $D_c = 512$  to reduce computation cost compared to  $C = 256$ ,  $D_s = 256$ , and  $D_c = 2048$  of the smallest model scale of the original paper [34]. The number of epochs is 50.

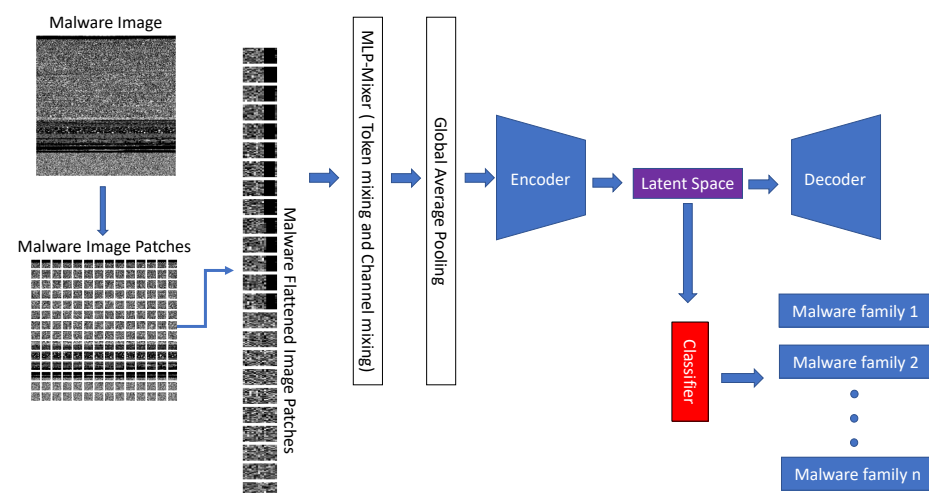


Figure 2. Overview of the proposed method.

Our proposed approach continuously brings the feature vector after the GAP layer from the MLP-mixer model into a simple autoencoder model with two layers of the encoder and two layers of the decoder. The node of the first layer of the encoder is  $2 * C$  channel, followed by  $C$  channel and  $C/2$  in latent space. We train the AE with mean squared error (MSE) loss function and 50 epochs; we use the latent space for classifying with typical classifiers algorithm of machine learning such as Decision Tree, k-Nearest Neighbors, Naïve Bayes, Nearest Centroid, Random Forest, and SVM to evaluate our system.

To evaluate our method, we utilize 10-fold Cross-Validation. One of the ten subsamples is held as validation data, and the remaining nine subsamples are used as training data. This process is repeated ten times with each of the ten subsamples used as validation. The average of ten results is the quality of the method.

## 4. Experiments

### 4.1. Dataset

This study evaluates our model adopts the Maling Dataset [10], consisting of 9339 malware image samples of 25 different families, and Malheur Dataset [39], consisting of 3133 malware binaries of 24 different families; in this study, they are converted into images by following the rule described in Table 1. The number of malware in each class is illustrated in Tables 2 and 3.

**Table 2.** Maling Dataset.

Class	Family Name	No. of Samples	Percentage (%)
0	Adialer.C	122	1.31
1	Agent.FYI	116	2.12
2	Allapple.A	2949	31.58
3	Appapple.L	1591	17.04
4	Alueron.gen!J	198	2.12
5	Autorun.K	106	1.14
6	C2LOP.gen!g	200	2.14
7	C2LOP.P	146	1.56
8	Dialplatform.B	177	1.89
9	Dontovo.A	162	1.73
10	Fakerean	381	4.08
11	Instantaccess	431	4.62
12	Lolyda.AA1	213	2.28
13	Lolyda.AA2	184	1.97
14	Lolyda.AA3	123	1.32
15	Lolyda.AT	159	1.70
16	Malex.gen!J	136	1.46
17	Obfuscator.AD	142	1.52
18	Rbot!gen	158	1.69
19	Skintrim.N	80	0.86
20	Swizzor.gen!E	128	1.37
21	Swizzor.gen!I	132	1.41
22	VB.AT	408	4.58
23	Wintrim.BX	97	1.04
24	Yuner.A	800	8.57
	<b>Total</b>	<b>9339</b>	

### 4.2. Evaluation Metrics

To evaluate the performance of the proposed CNN-based model, four standard performance metrics that are widely used in the existing research community were applied: accuracy, precision, recall, and F1-score. The four indicators are explained with the aid of the four parameters in Table 4, where the class being evaluated is positive, and the remaining classes are negative.

**Table 3.** Malheur Dataset.

Class	Family Name	No. of Samples	Percentage (%)
0	Adultbrowser	262	8.36
1	Allapple	300	9.58
2	Bancos	48	1.53
3	Casino	140	4.47
4	Dorfdo	65	2.07
5	Ejik	168	5.36
6	Flystudio	33	1.05
7	Ldpinch	43	1.37
8	Looper	209	6.67
9	Magiccasin	174	5.55
10	Podnuha	300	9.58
11	Posion	26	0.83
12	Porndialer	98	3.13
13	Rbot	101	3.22
14	Rotator	300	9.58
15	Sality	85	2.71
16	Spygames	139	4.44
17	Swizzor	78	2.49
18	Vapsup	45	1.44
19	Vikingdll	158	5.04
20	Vikingdz	68	2.17
21	Virut	202	6.45
22	Woikoiner	50	1.59
23	Zhelatin	41	1.31
	<b>Total</b>	<b>3133</b>	

**Table 4.** Performance metric parameters.

Parameter	Description
True positive (TP)	The number of positive class samples that are correctly classified
True Negative (TN)	The negative class is correctly classified into the negative class
False Positive (FP)	The number of negative class samples misclassified into the positive class
False Negative (FN)	The number of positive class samples misclassified into the negative class

*Accuracy* is defined as the ratio between the number of correctly classified samples to the total in the test dataset.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

*Precision* is defined as the ratios of true positive among the samples classified as positive.

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

*Recall* is defined as the ratio of samples classified as positive among true positives.

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

*F1-score* is used to evaluate the quality of the model.

$$F1\text{-score} = 2 * \frac{Precision \times Recall}{Precision + Recall} \quad (4)$$



### 4.3. Evaluation Results and Discussion

The model is assessed using a 10-fold Cross-Validation process to produce standard performance metrics. By randomly dividing the data into 70% training and 30% testing, the training and testing sets can be created. We utilized common classifiers such as Decision Tree, k-NN, Naive Bayes, Nearest Centroid, Random Forest, and SVM for both datasets.

Table 5 shows the performance of both datasets according to the various image sizes. It can be seen that the image size is a significant impact on the performance. Even with the smallest image size of  $32 \times 32$ , our proposed model gives better accuracy with the original MLP-mixer and ResNet50 to almost 10%. Compared to other classifiers, SVM overperform in almost evaluation metrics, with the same result as the paper [22]. Furthermore, this result shows that compressing the image does not adversely affect the classification performance.

From the results in Table 5, we have received the same results as the original paper [34] that MLP-mixer itself is not overperforming the CNN model of ResNet. In most cases, the accuracy of the MLP-mixer is lower than ResNet50, from 0.5% to a maximum of 1.93% in the Malimg dataset and 4.04% in the Malheur dataset. CNN takes advantage of positional invariance capability to get object information even in other places of images. It makes CNN models superior and perform better than traditional MLPs. However, binary files, as well as malicious codes, have fixed architecture, merely different between the ratio of sections [40]. As a result, CNN cannot exploit its strengths in processing images generated from malware. As shown in Table 6, CNN ties numerous parameters to learn attributes from input images. At the same time, MLP-mixer only needs less than 20 times (compared with the typical ResNet50 model) but still gives the same high performance as CNNs.

To evaluate the objective performance of the proposed method, the classification accuracy, precision, recall, and F1-score were compared with the results of other studies performed on the Malimg dataset (Table 6) and Malheur dataset (Table 7).

In our proposed method, we refine representation space created from a lightweight MLP-mixer model with a simple and effective Autoencoder model. Our parameter method is comparable to the lightweight CNN model in [14] with merely 0.83 M parameters. Still, the performance does not change significantly since the first-time dataset was published by Nataraj et al. [10] by only 0.31%. On the other hand, typical CNN models like ResNet [11,12], VGG19 [9,19], and DensNet [26] utilize enormous parameters and improved performance slightly; however, they require more computational power. Nevertheless, using significantly fewer and sufficient parameters, our proposed achieves higher performance than previous studies.

A series of previous studies show that more than pure CNN is needed for the malware image-based approach. Combining CNN with other methods, such as VAE and Attention mechanism as in [37] improves performance but makes the model more complex. Ref. [37] is only higher than 0.06% accuracy but requires more parameters by 1.57 M than this paper.

Regarding the Malheur dataset, Kim et al. [41] achieved a percentage of recall higher than us by 2.16%, but precision is noticeably lower at 8.08%. As a result, overall performance calculated by F1-score ours' is higher by 0.23%.

Table 5. Performance comparison according to input image changes.

Input Size	Methods	Classifiers	Maling Dataset				Malheur Dataset			
			Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
32 × 32	MLP-Mixer-AE	Decision Tree	76.88	69.79	72.10	70.96	89.08	83.71	83.20	82.96
		k-Nearest Neighbors	89.75	91.14	85.36	86.92	98.15	97.46	95.96	96.43
		Naïve Bayes	79.53	79.70	75.79	77.09	97.32	95.24	95.89	95.33
		Nearest Centroid	89.72	91.58	93.31	92.09	98.18	96.92	96.28	96.40
		Random Forest	88.67	90.36	79.88	81.67	97.83	97.56	95.36	96.22
		<b>SVM</b>	<b>94.67</b>	<b>95.19</b>	<b>93.42</b>	<b>94.12</b>	<b>98.37</b>	<b>97.78</b>	<b>96.44</b>	<b>96.71</b>
	MLP-Mixer	Softmax	84.62	-	-	-	94.47	-	-	-
	ResNet50	Softmax	84.94	-	-	-	91.38	-	-	-
64 × 64	MLP-Mixer-AE	Decision Tree	91.61	82.33	82.29	81.50	85.37	76.24	75.46	74.72
		k-Nearest Neighbors	97.27	95.22	93.06	93.72	96.74	94.03	94.03	94.61
		Naïve Bayes	96.04	91.38	91.45	91.24	95.82	94.35	94.35	93.32
		Nearest Centroid	98.27	95.79	96.17	95.91	97.35	95.88	95.88	95.55
		Random Forest	97.45	95.48	93.51	94.17	96.52	93.37	93.37	94.14
		<b>SVM</b>	<b>98.66</b>	<b>97.06</b>	<b>96.61</b>	<b>96.77</b>	<b>97.89</b>	<b>96.70</b>	<b>96.70</b>	<b>96.75</b>
	MLP-Mixer	Softmax	95.82	-	-	-	93.09	-	-	-
	ResNet50	Softmax	98.11	-	-	-	96.06	-	-	-
96 × 96	MLP-Mixer-AE	Decision Tree	92.98	84.99	85.76	85.24	87.61	80.38	80.29	79.73
		k-Nearest Neighbors	98.52	96.79	96.26	96.45	97.64	96.92	95.13	95.62
		Naïve Bayes	96.41	72.74	93.23	92.85	96.36	93.65	95.02	94.02
		Nearest Centroid	98.49	96.59	96.83	96.66	97.92	96.64	<b>96.23</b>	96.18
		Random Forest	98.31	96.45	95.67	95.97	97.51	96.74	94.48	95.36
		<b>SVM</b>	<b>99.05</b>	<b>97.82</b>	<b>97.58</b>	<b>97.66</b>	<b>98.02</b>	<b>97.05</b>	96.05	<b>96.31</b>
	MLP-Mixer	Softmax	97.25	-	-	-	93.30	-	-	-
	ResNet50	Softmax	98.43	-	-	-	97.34	-	-	-
224 × 224	MLP-Mixer-AE	Decision Tree	95.41	90.19	90.34	89.78	88.50	84.10	82.63	81.98
		k-Nearest Neighbors	99.06	97.85	97.73	95.75	97.92	<b>97.44</b>	95.71	96.13
		Naïve Bayes	98.21	96.18	96.19	96.09	97.03	94.89	95.37	94.90
		Nearest Centroid	98.68	97.15	97.29	97.16	98.05	97.03	<b>96.43</b>	96.42
		Random Forest	99.12	97.95	97.84	97.91	97.70	97.22	95.17	95.98
		<b>SVM</b>	<b>99.34</b>	<b>98.38</b>	<b>98.26</b>	<b>98.29</b>	<b>98.15</b>	97.24	96.38	<b>96.50</b>
	MLP-Mixer	Softmax	97.75	-	-	-	94.79	-	-	-
	ResNet50	Softmax	99.14	-	-	-	97.87	-	-	-

**Table 6.** Comparison with the latest malware classification models on the Maling dataset.

Studies	Year	Techniques	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	# Parameters (M)
Nataraj et al. [10]	2011	GIST feature + kNN	97.18	-	-	-	-
Rezende et al. [11]	2017	ResNet-50 + Softmax	98.62	-	-	-	25.56
Burks et al. [12]	2019	ResNet-18 + VAE	85.00	83.0	83.0	83.0	12.46
Naeem et al. [13]	2019	Combined SIFT-GIST	98.40	-	-	-	-
Roseline et al. [14]	2020	Lightweight CNNs	97.49	97.0	97.0	97.0	<b>0.83</b>
Awan et al. [9]	2021	VGG19 + Attention	97.62	97.68	97.50	97.20	143.67
Hemalatha et al. [26]	2021	DensNet + Reweighted Loss	98.23	97.78	97.92	97.85	~7.98
Sudhakar [28]	2021	ResNet50 + Transfer Learning	99.23	98.3	97.88	98.08	~25.56
Nisa et al. [15]	2021	SFTA + Cosine kNN	98.70	-	97.0	-	88.26
Lee et al. [16]	2021	Multiple Autoencoders	97.75	95.0	94.0	93.0	23.81
Hammad et al. [17]	2022	Feature Extraction Tamura	95.42	-	-	-	-
		Feature Extraction GoogleNet	96.48	-	-	-	4.00
Lin et al. [18]	2022	Bit-level sequences + CNNs	98.70	-	-	-	-
		Byte-level sequences + CNNs	98.91	-	-	-	-
Barros et al. [19]	2022	VGG19 + Zero-shot Learning	97.76	97.84	97.76	97.69	143.67
Wang et al. [20]	2022	CliqueNet + Multiscale	99.2	98.0	97.9	97.9	-
		Attention	99.2	98.0	97.9	97.9	-
Zhong et al. [21]	2022	CNN + gray	96.0	95.3	96.0	95.2	-
Son et al. [22]	2022	Dimension Reduction + SVM	98.51	-	-	-	-
Falana et al. [23]	2022	DNN + DGAN	95.63	95.34	95.30	94.98	-
Tuan et al. [37]	2022	CNN + AVAE	<b>99.40</b>	-	-	-	3.62
<b>This paper</b>	<b>2023</b>	<b>MLP-mixer Autoencoder</b>	<b>99.34</b>	<b>98.38</b>	<b>98.26</b>	<b>98.29</b>	<b>2.05</b>

**Table 7.** Comparison with the latest malware classification models on the Malheur dataset.

Studies	Year	Techniques	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	# Parameters (M)
Hurier et al. [42]	2017	Euphony	-	90.06	83.86	86.85	-
Naeem et al. [13]	2019	Combined SIFT-GIST	97.50	-	-	-	-
Sebastian et al. [43]	2020	AV labels	-	90.81	88.45	89.61	-
Kim et al. [41]	2022	Multiple AV	-	89.70	<b>98.60</b>	93.94	-
Son et al. [22]	2022	Dimension Reduction + SVM	95.79	-	-	-	-
<b>This paper</b>	<b>2023</b>	<b>MLP-mixer Autoencoder</b>	<b>98.37</b>	<b>97.78</b>	<b>96.44</b>	<b>96.71</b>	<b>1.39</b>

## 5. Conclusions

Currently, malware classification through image processing has attracted the attention of researchers because it does not need statics and dynamic analysis but still achieves high efficiency. Although CNN has become a popular tool, recent studies have made new changes to replace it with MLP or other CNN-free models, such as single or multi descriptors, the attention mechanism in Vision Transformer. However, it has not yet achieved the expected results like state-of-the-art CNN models.

The birth of the MLP-mixer has confirmed the strength of the neural network model again compared to CNNs in computer vision today. In the problem of processing images made from malicious code, due to the structural characteristics of the malicious code, CNN cannot take advantage of positional invariance capability. However, with a neural network that shares parameters via patch and channel, the performance gap between MLP and CNN has decreased significantly. The problem of feature vector re-purification has been solved by a simple but highly effective Autoencoder network. Experimental results show that our ensemble method is superior to the CNN-free models and has improved performance compared to various typical models of pure CNN such as VGG19, ResNet50, GoogleNet, DensNet, CliqueNet, and DGAN. Furthermore, our lightweight ensemble architecture utilizes fewer and sufficient parameters compared to the original MLP-mixer model and the complex combination architecture of CNN, and VAE with the Attention mechanism. However, it still achieves high performance through various experiments. This study does not use grid search hyperparameter tuning as in [44] to save computational time.

CNN models almost leverage pre-trained weight from previous work to improve performance. In our image-based malware task, there is still a need to be an optimal pre-

trained model to apply transfer learning to CNN. When training the model from scratch, our method still works better even though the amount of data is not as large as in [11]. As a result, this study shows the potential of MLP for specific data, such as images created from binary files.

In future work, we will continue to experiment with data from different sources in the upcoming work, especially for IoT malware. Unlike CNN, which uses GPUs to accelerate processing through parallel data processing, MLP can run flexibly in many environments, which is highly practical. Besides, we will further strengthen our framework to distinguish between malware families by adding more information into the representation space, such as Static Characteristics, and investigate other machine learning methods, such as semi-supervised learning approaches. Recent research showed CNN's limitation method for injection attacks and adversarial attacks. In the upcoming work, we will measure the robustness of our method with these attacks.

**Author Contributions:** Conceptualization, T.V.D. and H.S.; methodology, T.V.D.; validation, H.S. and M.K.; writing—original draft preparation, T.V.D.; writing—review and editing, H.S. and M.K. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Data Availability Statement:** Data is already available by previous authors, not new dataset.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Malware Attacks Targeting Ukraine Government-Microsoft on the Issues. Available online: <https://blogs.microsoft.com/on-the-issues/2022/01/15/mstic-malware-cyberattacks-ukraine-government/> (accessed on 14 December 2022).
2. Malware Statistics & Trends Report | AV-TEST. Available online: <https://www.av-test.org/en/statistics/malware> (accessed on 14 December 2022).
3. Raghuraman, C.; Suresh, S.; Shivshankar, S.; Chapaneri, R. Static and dynamic malware analysis using machine learning. In Proceedings of the First International Conference on Sustainable Technologies for Computational Intelligence, Jaipur, India, 29–30 March 2019; pp. 793–806.
4. Ye, Y.; Li, T.; Adjeroh, D.; Iyengar, S.S. A survey on malware detection using data mining techniques. *ACM Comput. Surv. (CSUR)* **2017**, *50*, 1–40. [CrossRef]
5. Belaoued, M.; Mazouzi, S. A chi-square-based decision for real-time malware detection using PE-file features. *J. Inf. Process. Syst.* **2016**, *12*, 644–660.
6. Maulana, R.J.; Kusuma, G.P. Malware classification based on system call sequences using deep learning. *Adv. Sci. Technol. Eng. Syst. J.* **2020**, *5*, 207–216. [CrossRef]
7. Demirkiran, F.; Çayır, A.; Ünal, U.; Dağ, H. An ensemble of pre-trained transformer models for imbalanced multiclass malware classification. *Comput. Secur.* **2022**, *121*, 102846. [CrossRef]
8. Conti, G.; Dean, E.; Sinda, M.; Sangster, B. Visual reverse engineering of binary and data files. In Proceedings of the International Workshop on Visualization for Computer Security, Cambridge, MA, USA, 15 September 2008; pp. 1–17.
9. Awan, M.J.; Masood, O.A.; Mohammed, M.A.; Yasin, A.; Zain, A.M.; Damaševičius, R.; Abdulkareem, K.H. Image-Based Malware Classification Using VGG19 Network and Spatial Convolutional Attention. *Electronics* **2021**, *10*, 2444. [CrossRef]
10. Nataraj, L.; Karthikeyan, S.; Jacob, G.; Manjunath, B.S. Malware images: Visualization and automatic classification. In Proceedings of the 8th International Symposium on Visualization for Cyber Security, Pittsburgh, PA, USA, 20 July 2011; pp. 1–7.
11. Rezende, E.; Ruppert, G.; Carvalho, T.; Ramos, F.; De Geus, P. Malicious software classification using transfer learning of resnet-50 deep neural network. In Proceedings of the 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), Cancun, Mexico, 18–21 December 2017; pp. 1011–1014.
12. Burks, R.; Islam, K.A.; Lu, Y.; Li, J. Data augmentation with generative models for improved malware detection: A comparative study. In Proceedings of the 2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), New York City, NY, USA, 10–12 October 2019; pp. 0660–0665.
13. Naeem, H.; Guo, B.; Ullah, F.; Naeem, M.R. A cross-platform malware variant classification based on image representation. *KSII Trans. Internet Inf. Syst. (TIIS)* **2019**, *13*, 3756–3777.
14. Abijah Roseline, S.; Hari, G.; Geetha, S.; Krishnamurthy, R. Vision-based malware detection and classification using lightweight deep learning paradigm. In Proceedings of the International Conference on Computer Vision and Image Processing, Prayagraj, India, 4–6 December 2020; pp. 62–73.

15. Nisa, M.; Shah, J.H.; Kanwal, S.; Raza, M.; Khan, M.A.; Damaševičius, R.; Blažauskas, T. Hybrid malware classification method using segmentation-based fractal texture analysis and deep convolution neural network features. *Appl. Sci.* **2020**, *10*, 4966. [[CrossRef](#)]
16. Lee, J.; Lee, J. A Classification System for Visualized Malware Based on Multiple Autoencoder Models. *IEEE Access* **2021**, *9*, 144786–144795. [[CrossRef](#)]
17. Hammad, B.T.; Jamil, N.; Ahmed, I.T.; Zain, Z.M.; Basheer, S. Robust Malware Family Classification Using Effective Features and Classifiers. *Appl. Sci.* **2022**, *12*, 7877. [[CrossRef](#)]
18. Lin, W.C.; Yeh, Y.R. Efficient Malware Classification by Binary Sequences with One-Dimensional Convolutional Neural Networks. *Mathematics* **2022**, *10*, 608. [[CrossRef](#)]
19. Barros, P.H.; Chagas, E.T.; Oliveira, L.B.; Queiroz, F.; Ramos, H.S. Malware-SMELL: A zero-shot learning strategy for detecting zero-day vulnerabilities. *Comput. Secur.* **2022**, *120*, 102785. [[CrossRef](#)]
20. Wang, C.; Zhao, Z.; Wang, F.; Li, Q. MSAAM: A Multiscale Adaptive Attention Module for IoT Malware Detection and Family Classification. *Secur. Commun. Netw.* **2022**, *2022*, . [[CrossRef](#)]
21. Zhong, F.; Chen, Z.; Xu, M.; Zhang, G.; Yu, D.; Cheng, X. Malware-on-the-Brain: Illuminating Malware Byte Codes with Images for Malware Classification. *IEEE Trans. Comput.* **2022**, *72*, 438–451. [[CrossRef](#)]
22. Son, T.T.; Lee, C.; Le-Minh, H.; Aslam, N.; Dat, V.C. An enhancement for image-based malware classification using machine learning with low dimension normalized input images. *J. Inf. Secur. Appl.* **2022**, *69*, 103308. [[CrossRef](#)]
23. Falana, O.J.; Sodiya, A.S.; Onashoga, S.A.; Badmus, B.S. Mal-Detect: An intelligent visualization approach for malware detection. *J. King Saud-Univ.-Comput. Inf. Sci.* **2022**, *34*. [[CrossRef](#)]
24. Ban, T.; Isawa, R.; Guo, S.; Inoue, D.; Nakao, K. Efficient malware packer identification using support vector machines with spectrum kernel. In Proceedings of the 2013 Eighth Asia Joint Conference on Information Security, Seoul, Korea, 25–26 July 2013; pp. 69–76.
25. Wong, W.; Juwono, F.H.; Apriono, C. Vision-based malware detection: A transfer learning approach using optimal ECOC-SVM configuration. *IEEE Access* **2021**, *9*, 159262–159270. [[CrossRef](#)]
26. Hemalatha, J.; Roseline, S.A.; Geetha, S.; Kadry, S.; Damaševičius, R. An efficient densenet-based deep learning model for malware detection. *Entropy* **2021**, *23*, 344. [[CrossRef](#)]
27. Kim, H.M.; Lee, K.H. IIoT Malware Detection Using Edge Computing and Deep Learning for Cybersecurity in Smart Factories. *Appl. Sci.* **2022**, *12*, 7679. [[CrossRef](#)]
28. Kumar, S. MCFT-CNN: Malware classification with fine-tune convolution neural networks using traditional and transfer learning in Internet of Things. *Future Gener. Comput. Syst.* **2021**, *125*, 334–351.
29. Ding, Y.; Zhang, X.; Hu, J.; Xu, W. Android malware detection method based on bytecode image. *J. Ambient. Intell. Humaniz. Comput.* **2020**, 1–10. [[CrossRef](#)]
30. Bochinski, E.; Senst, T.; Sikora, T. Hyper-parameter optimization for convolutional neural network committees based on evolutionary algorithms. In Proceedings of the 2017 IEEE International Conference on Image Processing (ICIP), Beijing, China, 17–20 September 2017; pp. 3924–3928.
31. Choi, S.; Bae, J.; Lee, C.; Kim, Y.; Kim, J. Attention-based automated feature extraction for malware analysis. *Sensors* **2020**, *20*, 2893. [[CrossRef](#)]
32. Yakura, H.; Shinozaki, S.; Nishimura, R.; Oyama, Y.; Sakuma, J. Neural malware analysis with attention mechanism. *Comput. Secur.* **2019**, *87*, 101592. [[CrossRef](#)]
33. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv* **2010**, arXiv:2010.11929.
34. Tolstikhin, I.O.; Houthouwer, N.; Kolesnikov, A.; Beyer, L.; Zhai, X.; Unterthiner, T.; Yung, J.; Steiner, A.; Keysers, D.; Uszkoreit, J.; et al. Mlp-mixer: An all-mlp architecture for vision. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 24261–24272.
35. da Silva, A.A.; Pamplona Segundo, M. On Deceiving Malware Classification with Section Injection. *Mach. Learn. Knowl. Extr.* **2023**, *5*, 144–168. [[CrossRef](#)]
36. Vu, D.L.; Nguyen, T.K.; Nguyen, T.V.; Nguyen, T.N.; Massacci, F.; Phung, P.H. HIT4Mal: Hybrid image transformation for malware classification. *Trans. Emerg. Telecommun. Technol.* **2020**, *31*, e3789. [[CrossRef](#)]
37. Van Dao, T.; Sato, H.; Kubo, M. An Attention Mechanism for Combination of CNN and VAE for Image-Based Malware Classification. *IEEE Access* **2022**, *10*, 85127–85136.
38. Liu, H.; Dai, Z.; So, D.; Le, Q.V. Pay attention to mlps. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 9204–9215.
39. Rieck, K.; Trinius, P.; Willems, C.; Holz, T. Automatic analysis of malware behavior using machine learning. *J. Comput. Secur.* **2011**, *19*, 639–668. [[CrossRef](#)]
40. Huang, W.C.; Di Troia, F.; Stamp, M. Robust Hashing for Image-based Malware Classification. In Proceedings of the ICETE, Porto, Portugal, 26–28 July 2018; pp. 617–625.
41. Kim, S.; Jung, W.; Lee, K.; Oh, H.; Kim, E.T. Sumav: Fully automated malware labeling. *ICT Express* **2022**, *8*, 530–538. [[CrossRef](#)]
42. Hurier, M.; Suarez-Tangil, G.; Dash, S.K.; Bissyandé, T.F.; Le Traon, Y.; Klein, J.; Cavallaro, L. Euphony: Harmonious unification of cacophonous anti-virus vendor labels for android malware. In Proceedings of the 2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR), Buenos Aires, Argentina, 20–28 May 2017; pp. 425–435.

43. Sebastián, S.; Caballero, J. Avclass2: Massive malware tag extraction from av labels. In Proceedings of the Annual Computer Security Applications Conference, Austin, TX, USA, 7–11 December 2020; pp. 42–53.
44. Ghouti, L.; Iman, M. Malware classification using compact image features and multiclass support vector machines. *IET Inf. Secur.* **2020**, *14*, 419–429. [[CrossRef](#)]

**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.