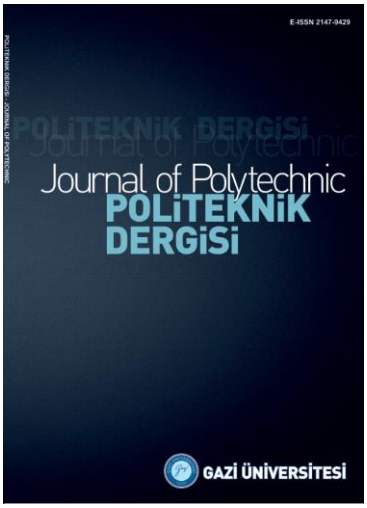

POLİTEKNİK DERGİSİ

JOURNAL of POLYTECHNIC

ISSN: 1302-0900 (PRINT), ISSN: 2147-9429 (ONLINE)
URL: http://dergipark.org.tr/politeknik

EfficientNet-based deep learning for malware classification: A dynamic distribution adaptation approach

Kötü amaçlı yazılım sınıflandırması için efficientnet tabanlı derin öğrenme: Dinamik dağıtım uyarlama yaklaşımı

Yazar(lar) (Author(s)): *Ceren Umay ÖZTEN[1], Adem TEKEREK[2]*

ORCID[1]: 0000-0002-6962-9259

ORCID[2]: 0000-0002-0880-7955

To cite to this article: Özten C. U. and Tekerek A., "EfficientNet-Based Deep Learning for Malware Classification: A Dynamic Distribution Adaptation Approach", ***Journal of Polytechnic***, 28(3): 845-866, (2025).

Bu makaleye şu şekilde atıfta bulunabilirsiniz: Özten C. U. ve Tekerek A., "EfficientNet-Based Deep Learning for Malware Classification: A Dynamic Distribution Adaptation Approach", ***Politeknik Dergisi***, 28(3): 845-866, (2025).

Erişim linki (To link to this article): http://dergipark.org.tr/politeknik/archive

DOI: 10.2339/politeknik.1536669

EfficientNet-Based Deep Learning for Malware Classification: A Dynamic Distribution Adaptation Approach

Highlights

- *Malware Classification using the EfficientNet and Dynamic Distribution Adaptation Network approach*
- *Applying data preprocessing*
- *Detailing the model training and validation processes*
- *Analyzing results with performance evaluation metrics*

Graphical Abstract

In this study, the EfficientNet deep learning model was used to classify malware images.

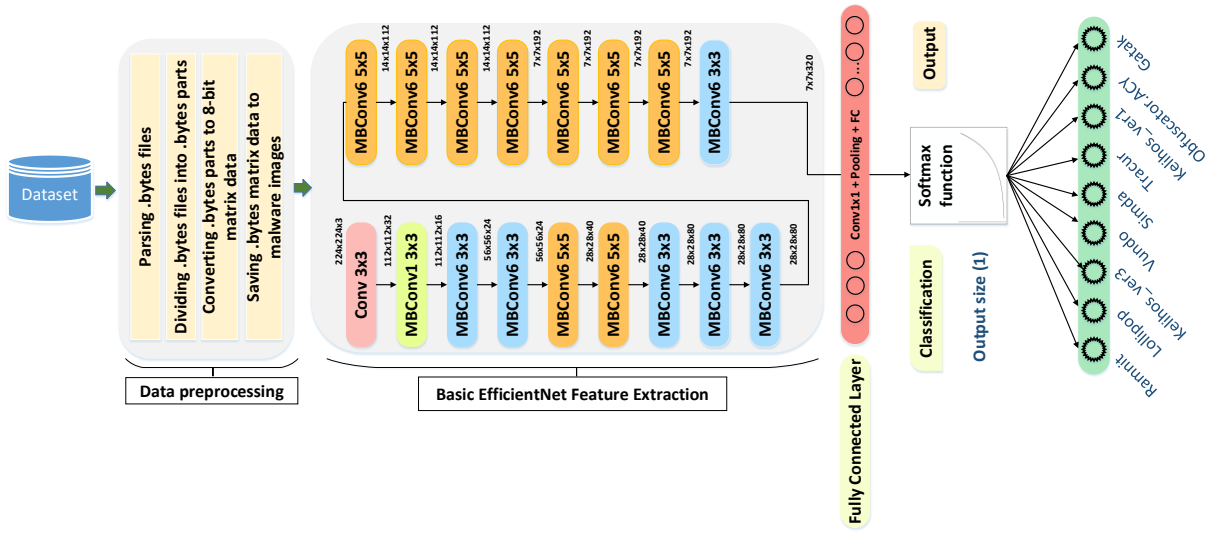

Figure. Proposed Methodology

Aim

To investigate the effectiveness of EfficientNet deep learning model in detecting classifying malware images.

Design & Methodology

Steps included dataset analysis, data preprocessing, EfficientNet model and Dynamic Distribution Adaptation Network approach and performance evaluation.

Originality

This study is one of the rare works successfully applying EfficientNet model to classify malware images.

Findings

EfficientNet models are applied succesfully to malware classification.

Conclusion

EfficientNet has proven the effectiveness of deep learning in security by classfiying malware.

Declaration of Ethical Standards

The authors of this article declare that the materials and methods used in this study do not require ethical committee permission and/or legal-special permission.

EfficientNet-Based Deep Learning for Malware Classification: A Dynamic Distribution Adaptation Approach

Araştırma Makalesi / Research Article

Ceren Umay ÖZTEN[1], Adem TEKEREK[1]

[1] Computer Engineering Deparment, Technology Faculty, Gazi University, Türkiye
(Geliş/Received: 21.07.2024 ; Kabul/Accepted: 07.10.2024 ; Erken Görünüm/Early View : 13.10.2024)

ABSTRACT

Malware is a general name given to all malicious software that threatens and prevents the use of information systems. Computers, which have become mandatory in daily life, are constantly under the threat of malware as well as facilitating human life. Therefore, the detection of malware that threatens computer systems is important. This study focuses on the classification of malware. In the study, a deep learning model based on the EfficientNet architecture and the Dynamic Distribution Adaptation Network approach were proposed and these proposed models were tested using the Microsoft Malware Classification Challenge (MMCC) and Dumpware10 datasets. In the study, the process of converting malware into images was discussed and the EfficientNet model was used as the basis for the classification of these images. The EfficientNet backbone-based Dynamic Distribution Adaptation Network achieved 97% accuracy in the MMCC dataset and 96% accuracy in the Dumpware10 dataset. As a result, the EfficientNet architecture proved the effectiveness of deep learning in the classification of malware and cybersecurity.

Keywords: EfficientNet architecture, malware classification, data preprocessing, compound scaling.

Kötü Amaçlı Yazılım Sınıflandırması için EfficientNet Tabanlı Derin Öğrenme: Dinamik Dağıtım Uyarlama Yaklaşımı

ÖZ

Malware, bilişim sistemlerini tehdit eden ve kullanımını engelleyen tüm kötü amaçlı yazılımlara verilen genel bir addır. Günlük hayatta kullanımı zorunlu hale gelen bilgisayarlar, insan hayatını kolaylaştırmanın yanı sıra kötü amaçlı yazılımların da sürekli tehdidi altındadır. Bu nedenle bilgisayar sistemlerini tehdit eden kötü amaçlı yazılımların tespiti önemlidir. Bu çalışmada kötü amaçlı yazılımların sınıflandırılması üzerine odaklanılmıştır. Çalışmada, EfficientNet mimarisine dayalı bir derin öğrenme modeli ve Dynamic Distribution Adaptation Network yaklaşımı önerilmiş ve bu önerilen modellemeler, Microsoft Kötü Amaçlı Yazılım Sınıflandırma Mücadelesi (MMCC) ve Dumpware10 veri kümeleri kullanılarak test edilmiştir. Çalışmada, kötü amaçlı yazılımların görüntülere dönüştürülme süreci ele alınmış ve bu görüntülerin sınıflandırılmasında EfficientNet modeli taban alınmıştır. EfficientNet backbone tabanlı Dynamic Distribution Adaptation Network, MMCC veri kümesinde %97, Dumpware10 veri kümesinde ise %96 doğruluk elde etmiştir. Sonuç olarak, EfficientNet mimarisi kötü amaçlı yazılımların sınıflandırılmasında ve siber güvenlikte derin öğrenmenin etkinliğini kanıtlamıştır.

Anahtar Kelimeler: EfficientNet mimarisi, malware sınıflandırma, veri ön işleme, mimari ölçeklendirme.

1. INTRODUCTION

The transformation of the first historic ARPANet network into today's virtual network of millions of servers increases cyber threats. The first “bug” definition of a computer virus has evolved into specific cyber-attack software, and cyber-attack software has become widespread, divided into variable malware families. Malware families can infiltrate information system infrastructures, mobile electronic devices, digital blockchains and government databases. This situation requires the correct classification of malware families. In this context, incorrect malware classification can render cyber security analysis dysfunctional. The goal of this research is to utilize EfficientNet architecture to distribute malware image transformations to the correct malware family classes. In this research, EfficientNet Inverted Residual (MBConv) layer blocks are defined to MaxPooling experiments with Transfer Learning. Malware dataset collection, malware preprocessing, EfficientNet model development and experimental result analysis are included in the method modeling of the research. In the method modeling, malware datasets are collected, malware samples belonging to the malware sets are decomposed into tensor blocks in preprocessing, and the tensor blocks are converted into three-channel RGB and two-channel grayscale malware images. The

*Sorumlu Yazar (Corresponding Author)
e-mail : atekerek@gazi.edu.tr

malware images are then trained on EfficientNet architecture. In the Transfer Learning and MaxPooling experiments, layer depth, input resolution and layer width settings are studied and MaxPooling final layer replacement is tested.

2. LITERATURE REVIEW

In the literature, malware preprocessing, implementation of convolutional layer sequences, CNN - BiLSTM two-tier model, LSTM layer integration with Convolution Dense layer, adaptation of Autoencoder neural network to grayscale malware images and random partitioning of malware dataset are investigated.

Huaxin Deng, et al., used Markov transfer matrices in malware data preprocessing. In Markov transfer matrices, the team assigned the probabilities of combinations of both consecutive letters or numbers of machine codes, the first letter and the last two letters of opcode fragments to three matrix image channels, and obtained three-channel malware images. And the malware images are entered into 1 fully-connected dense layered architecture with 4 convolution layers, 4 pooling filtering layers, and 4 convolution layers. The Markov method achieves 99.4% accuracy on the Microsoft Malware Challenge dataset [1]. Mumtaz Ahmed, et al., converted the malware data into a two-channel grayscale image. The team converted the malware byte files into hexadecimal numbers, transferred the binary hexadecimal numbers to pixels, and performed min-max normalization on the total pixel image. They then trained the pixel images on the InceptionV3 model with hidden layers frozen. The InceptionV3 model achieved 98.76% accuracy on the Microsoft Challenge BIG15 test set [2]. Sanjeev Kumar and Kajal Panda combined the feature outputs of VGG16, VGG19, ResNet50 and InceptionV3 models in the SDIF-CNN method into a single feature output in horizontal space and filtered the redundant data from the feature output. The filtered feature output was passed to KNN, SVM, Random Forest, Multi-layer Perceptron (MLP), Extra Tree and Gaussian Naive Bayes classifiers. The Multi-layer Perceptron model recorded 98.55% accuracy rate, 99% precision rate, 99% recall rate and 99% f1-score rate on the MalImg dataset [3]. Seok-Jun Bu and Sung-Bae Cho solved the malware structure for classification in an evolutionary ternary network and optimized the mixed malware variation into groups of inter-representation distances using a genetic algorithm. In the genetic algorithm, the team entered the byte malware image into the evolutionary triple network and generated new weight-sharing convolutional networks in the space of weight changes. The triple triplet loss due to the genetic algorithm brought similar malware samples closer together while pushing different malware samples away [4]. In the multi-view multidimensional feature fusion approach, Rajasekhar Chaganti, et al. combined static, dynamic and image feature sets of malware. In the multidimensional feature fusion learning of the models, each feature set presents the discriminative semantic characteristics of the malware. The team fused PE Section and PE Import, dynamic PE API and PE Image malware attribute files into a convolutional neural network. The connected convolutional neural network included the convolution1D-Maxpooling-Dense layer block. The dynamic PE API based feature set achieved 99% accuracy [5]. In the two-stage hybrid approach, Seungyeon Baek et al. vectorized opcode sequences in the static stage and decomposed them into benign file attributes in the Bi-LSTM model transition, and in the dynamic stage, they extracted process memory and API calls attributes by running the decomposed benign file attributes in a virtual Cuckoo Sandbox environment. Then, it transformed the attributes into a three-dimensional tensor structure with process memory, category and API calls channels. Finally, the three-dimensional tensor structure was classified in the EfficientNet-b3 model. The EfficientNet-b3 model achieved an accuracy rate of 94.98% [6]. Manish Kumar integrated convolutional neural network CNN with Bi-LSTM network for malware detection. With dynamic malware API calls, process execution signals are converted into process tree vectors. The high-level vector output is decomposed in the embedded layer and the low-level vector fragments are passed through the CNN1-BiLSTM1-CNN2-BiLSTM2-Dense-Softmax chaining layer pattern. Convolution layers filtered the feature for the LSTM layers, and the dual CNN and dual BiLSTM modeling showed high success. The binary CNN-BiLSTM modeling recorded an accuracy rate of 0.99 [7]. In the grayscale autoencoder approach, Xiaofei Xing, et al. encoded the APK code of the malware and benign file into decimal byte data and fixed it into a grayscale two-dimensional matrix, and passed the grayscale malware image data through AE-1 and AE-2 autoencoder structures. The AE-1 autoencoder structured the feature extraction of the grayscale image into the actual malware classification. The AE-2 autoencoder handled the malware discrimination of the classified benign file. The AE-1 autoencoder architecture has convolution, filtering and upsampling layers. The AE-2 autoencoder architecture has a multi-layer perceptron network in addition to the AE-1 architecture layers. The multi-layer perceptron network achieved 96% accuracy [8].

In the LSTM-Dense method, Esraa Saleh Alomari et al. presented malware detection based on feature selection with deep learning and feature selection in the correlation matrix by processing datasets. Datasets of variable attribute selections were trained in the LSTM model with dense dense layer. In the training, 5 hidden layers in the Dense Dense Layer model are defined for ReLU activation between the input and output layers. The LSTM model replaced the first Dense layer of the Dense dense layer model with the ReLU activated LSTM layer. Narrowing the datasets by feature selection met the performance of almost the entire dataset [9]. R.

Vinayakumar et al. removed bias by applying different separations to the datasets along the bias-deep learning line. Removing bias from the datasets made the malware detection model training independent. Light GBM 100-tree modeling, convolutional 1D layers and LSTM hybrid MalConv variants were investigated on the Ember dataset. Flexible and real-time hybrid deep learning models are used for malware preprocessing and classification. Ember dataset was randomly split into 60% training and 40% test sets and introduced to the models [10]. Handhika Yanuar Pratama and Jeckson Sidabutar apply EfficientNet models - EfficientNet-b0, EfficientNet-b1, EfficientNet-b2, EfficientNet-b3, EfficientNet-b4, EfficientNet-b5, EfficientNet-b6 and EfficientNet-b7 - to two-channel grayscale and three-channel RGB malware images from the Malware Classification Challenge (BIG 2015) dataset. EfficientNet models are able to perform successful deep learning classification on the ImageNet dataset. In the study experiments, EfficientNet-b7 architecture applied to three-channel RGB malware images achieves 99.63% accuracy rate, 98.36% precision rate, 98.35% recall rate, 98.34% F1-score rate and 98.30% AUC rate [11]. Cyber breach detection systems are being developed to protect against DDoS attacks in SDN-based SCADA systems. Oyucu et al. proposed a Decision Tree-based Ensemble Learning technique that detects DDoS attacks in SDN-based SCADA systems by distinguishing between normal data flow and DDoS attack. The proposed hybrid model using machine learning classification methods includes dataset generation, feature refinement, normalization and classification stages. For the training and testing of ensemble learning models, normal traffic and DDoS attack data flow traffic are obtained from specific experimental network topology simulation. Minimum Redundancy Maximum Relevance (MrMR) method is adopted for feature balance in the dataset. Feature selection and hyperparameter tuning are used to optimize decision tree ensemble models. The experiments show that feature selection, different combinations of decision tree ensemble models, and hyperparameter tuning can lead to better detection performance against DDoS attacks. The team's Ensemble Boosted Trees method showed the highest accuracy performance of 92.9% [12]. Polat et al. proposed a multi-stage learning model for DDoS attack detection in SDA-based SCADA systems by combining 1-dimensional convolutional neural network (1D-CNN) and decision tree based classification. In the proposed model, the feature extracted from the 1D-CNN convolutional neural network model is input to the decision tree model. While the 1D-CNN network model performs deeper and more complex feature extraction, the decision tree model defines the features into the decision structure. A new dataset of specific experimental network topology based on varying attack scenarios is used to train and test the model. The proposed model achieved an accuracy of 97.8% in DDoS attack detection [13]. In addition to the literature studies, the proposed model utilizes all EfficientNet-b0-EfficientNet-b7 models for malware classification. In this study, two-channel grayscale malware images and three-channel RGB malware images were generated by processing bytes files of different datasets.

In this study, two-channel grayscale malware images of the Microsoft Malware Classification Challenge (BIG 2015) dataset are more successful with the EfficientNet-b7 model, while three-channel RGB malware images are more successful with the EfficientNet-b5 model. For the Dumpware10 dataset, the EfficientNet-b6 architecture is more successful. In the model development, a deep learning model was created by using convolutional bottleneck and depthwise separable convolution in the inverse residual block structure. The EfficientNet architecture of the study effectively applies the layer structure that reduces the gradient computation while approaching full convolution to malware image datasets. The combination of the Microsoft Malware Classification Challenge (BIG 2015) dataset and the proprietary Dumpware10 dataset offers different perspectives in experimental testing. While the Microsoft Malware Classification Challenge (BIG 2015) dataset serves as a general benchmark, the Dumpware10 dataset provides a different benchmark evaluation of the working model.

3. MATERIAL and METHOD

3.1. Methods

For malware detection, a deep learning model is built using convolutional bottleneck and depthwise separable convolution in the inverse residual block structure of the EfficientNet architecture.

3.1.1. Transfer learning

Transfer learning is the transfer of the activation hardware of the deep neural network architecture, previously subjected to the training cycle, to different tasks. The final Dense layer of the deep neural network architecture is changed according to the classification distribution of the task. The deep neural network layer architecture performs activation learning by pre-training on a large benchmark dataset such as ImageNet. The deep neural network layers can transfer the activation learning experienced on the ImageNet dataset to different classification tasks belonging to different datasets. For example, the weighted DenseNet121 neural network architecture, pre-trained on the ImageNet dataset, can be adapted to a dataset containing human facial expressions and aiming to classify human emotions - happiness, anger, sadness, neutral, surprise, fear, disgust. The DenseNet121 neural network architecture applies the edge, shape, color and associated texture pixel feature information learned from ImageNet image data to human emotion classification of facial expression images. The last Dense layer of the DenseNet121 neural network

architecture is replaced by a softmax layer which is divided into 7 human emotion classes. In the DenseNet121 architecture, the layers carrying low-level feature information do not participate in gradient generalization during training, but the last Dense layers carrying high-level feature information participate in gradient generalization. For successful softmax classification, activation function, optimization function, learning rate, momentum, number of epochs and weight decay function hyperparameters are added to improve the gradient generalization. As a result, the weights of the pre-trained DenseNet121 neural network architecture are task domain specific in the transfer learning domain. The hyperparameters drive the gradient generalization of the deep neural network training loop. Optimization functions such as Adam, SGD, Adagrad, RMSProp regulate the activation flow of the neural network architecture layers. The training cycle of the neural network architecture translates into more successful gradient generalization. Adjusting the learning rate, momentum, weight decay function and number of epochs shape the effect of optimization functions on the training cycle. Dense layer swapping and hyperparameter configuration are prominent in the transfer learning neural network architecture. High-performance neural network architectures such as DenseNet121, ResNet50, InceptionV3, Xception, AlexNet, EfficientNet-b0-b7 transform Dense layer exchange and hyperparameter configuration in the transfer learning extension. It performs different training cycles according to the datasets. In this context, the transfer learning method approximates the neuron activations of pattern neural network architectures to the gradient generalization of the dataset and shows successful results in classification tasks. In this method, the neural network layers fix the gradient computation up to the Dense layer, or the gradient computation of layer blocks that process only low-level feature information.

Transfer learning domain connected with deep neural networks through deeper layers of feature extraction and neural network based adaptation of big volume of prevalent knowledge domain to small volume of intuitive knowledge domain. For example, a researcher can use existent biological protein structure information with artificial intelligence for predicting protein structures. Information knowledge transfer through deep neural networks has deep transfer learning model-based methods with variety of frozen pre-trained layer blocks, added new layer blocks and regulation of layer gradient computations. Deep transfer learning model-based methods are divided into finetuning, freezing low-level CNN layers, and progressive learning approaches.

Finetuning generalizes the neural network model, which has been pre-trained with data close to the target task scope, to the dataset of the target task. In this respect, Finetuning is the most common deep transfer learning approach. The finetuning approach can reduce the computational cost of the training cycle for the dataset and addresses the need for a large dataset for the target task. However, finetuning faces the problem of gradient loss during the training cycle. Freezing low-level CNN layers means that the low-level convolution layers of the neural network model are frozen and do not participate in the gradient calculation of the training cycle. Only the intermediate fully-connected layers participate in the gradient generalization in the training cycle of the target dataset. The low-level CNN layers undertake the feature extraction of the dataset, while the intermediate fully-connected layers undertake the classification of the feature extraction. Progressive learning uses part or all of the layers of a pre-trained neural network model without entering the gradient generalization of the training cycle. The new layer configuration added to the neural network model is trained on the target dataset. The triple layer block autoencoder structure minimizes the mismatch between training and test feature data by applying a maximum mismatch term to the features of the training and target data [14].

In figure 1 previously explained deep transfer learning methods of finetuning, frozen CNN layers and progressive learning are depicted.

3.1.2. EfficientNet

The EfficientNet architecture consists of “compound scaling” modeling of resolution, layer width and layer depth in MBConv structures. This architecture gives MBConv structures the flexibility of multidimensional compound scaling expansion. The MBConv inverted residual block structure follows a convolutional layer path that shrinks at the beginning, expands in the middle, and shrinks again at the end. Initially it follows (1x1) convolution filtering, followed by (3x3) depthwise convolution block filtering. Then the (1x1) convolution filtering reduces the number of parameters in the middle layer. The MBConv block is an inverted residual block modeling that includes an inverted layer transformation with performance impact.

MBConv block

In figure 2 MBConv block structure and connections are shown. MBConv has a convolutional block architecture. MBConv convolutional block structure consists of depthwise separable convolutions, BatchNormalization, Squeeze and Excitation module, Projection phase, BatchNormalization, activation and Skip connection internals layer components. The full convolutional layer architecture is replaced by linear bottlenecks. The full convolutional operator neural network layer of the Depthwise separable convolutions block architecture decomposes the convolutional layer into two separate layers by factorization. The first layer is the depthwise convolution layer. This layer applies a lightweight single convolutional filtering for each input channel. The second decomposed layer is the (1x1) pointwise convolution layer, which combines new feature outputs

from linear computational combinations of input parameters.

The standard convolution layer takes the input hixwixdi and Li tensor structure and processes K convolutional kernel filtering to produce the output Lj tensor structure hixwixdj. The standard convolution layer has a computational cost of hi*wi*di*dj*k*k*k, while the depthwise separable convolution layer has a lower computational cost in the formal convolution functionality. Depthwise and (1x1) pointwise convolution layers have lower running cost compared to the standard convolution layer. Depthwise separable convolution layer reduces the running cost by a factor of k^2 compared to traditional layers [15].

Inverted residual block

In figure 4 inverted residual block has (1x1) Conv2d and (3x3) Depthwise blocks applying inverted residual block connecting bottleneck expansions. Bottleneck blocks have usage for reducing parameter density while maintaining sufficient portion of model's feature extraction capability. Bottleneck blocks closely resemble the structure of residual block architectures. According to Sandler's research team a residual block has first widening then narrowing and widening again layer pipeline and begins with several bottleneck layers immediately after the input layer, while inverted residual block has first narrowing then widening and narrowing again pipeline connecting (1x1) Conv2d - (3x3) Depthwise block and (1x1) Conv structures. Bottleneck layers are connected to following expansion layer. While the bottleneck layers capture essential feature information, the expansion layers are responsible for reviving non-linear feature details without weighting. In this setup, "shortcut" connections are established between the bottleneck layers to ensure smooth information flow.

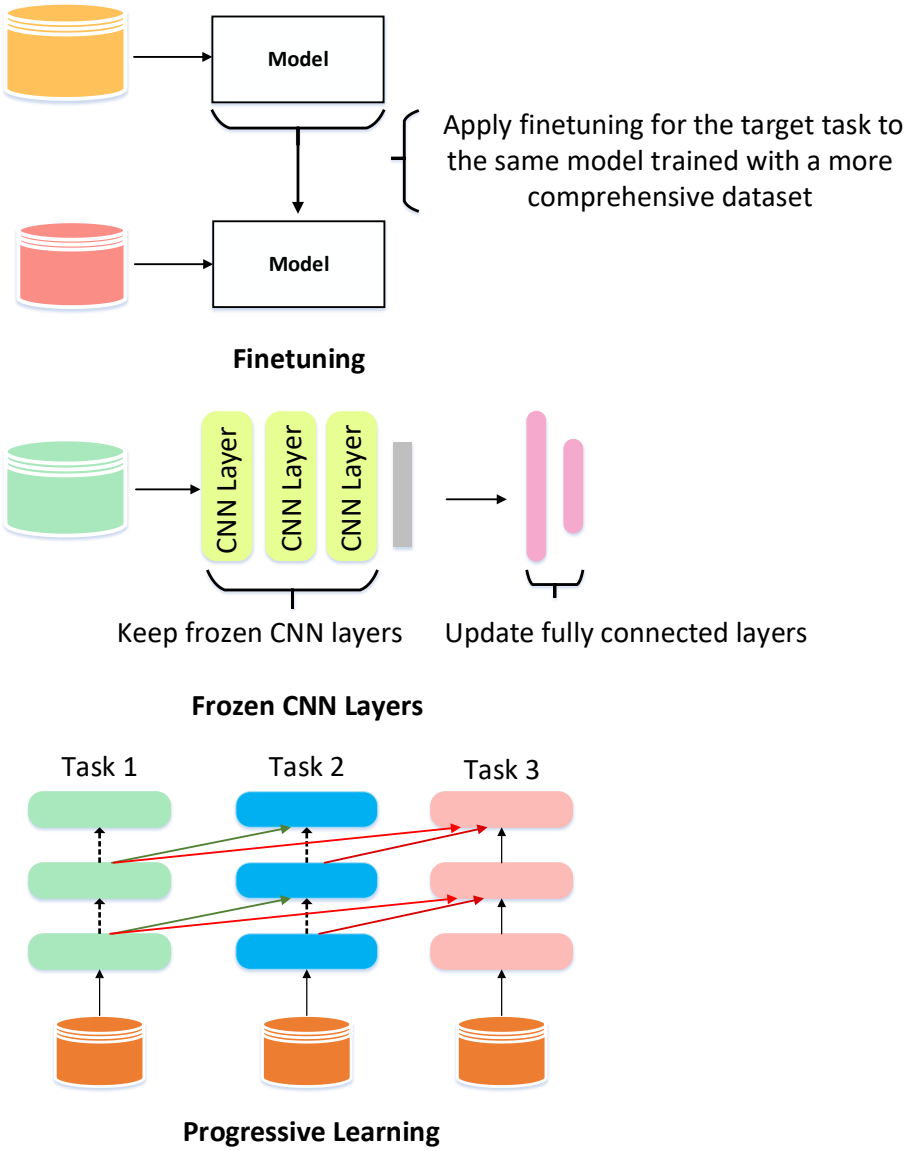

Figure 1. Deep transfer learning model-based methods

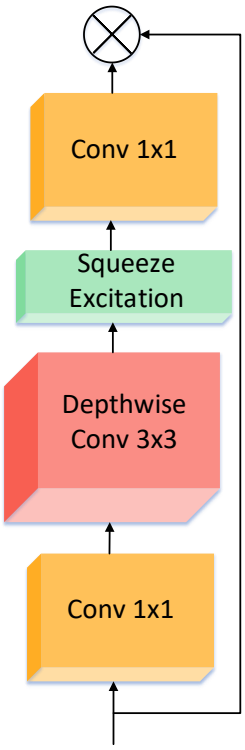

Figure 2. MBConv Block

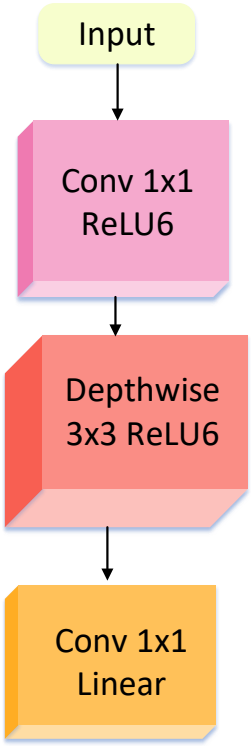

Figure 3. Linear Bottleneck Block

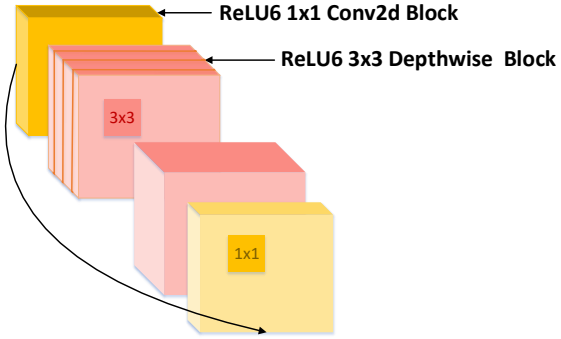

Figure 4. Inverted Residual Block

EfficientNet architecture scaling

EfficientNet architecture scaling enhances model performance by proportionally increasing the depth, width, and resolution of the neural network layers. This scaling is governed by the EfficientNet compound coefficient, which adjusts these dimensions simultaneously. At the same time, the neural network structure is expanded to the EfficientNet model series such as shallower EfficientNet-b0 and more complicated EfficientNet-b7. The convolutional design analyzes the transformation of neural network layers through layer width, channel count, input height and width parameters in a flexible manner. The depth (d), width (w), and resolution (r) scaling of the layers are interconnected with distinctive parameters. For instance, depth scaling is connected to layer number of relevant depth, width scaling is connected to neuron channel unit count and resolution scaling is connected to input width and height parameters. While EfficientNet-b0 has 224 resolution scaling and depth scaling that has 237 number of layers, EfficientNet-b7 has 600 resolution and depth scaling that has 813 number of layers.

Depth (d)

Scaling the depth of neural networks is frequently used in convolutional structures. With increasing depth, convolutional structures (ConvNet) can capture more complex and rich feature information. It can generalize better to unexperienced tasks. However, deep neural networks have a more difficult training cycle due to the vanishing gradient problem. Skip connections and batch normalization add-on dilute the gradient problem and reduce the training accuracy of deep neural networks [16].

Width (w)

Increase the width scaling of a neural network is a common technique, particularly for smaller models. Width scaling is influenced by neuron channel unit count across layers and increasing neuron channels improves feature extraction of model compatible with findings by Zagoruyko and Komodakis, wider networks are more effective at capturing detailed feature information and are easier to train. However, as the network width growth

forces model complexity and approximates training accuracy to plateau as it reaches the limit.

Resolution (r)

Convolutional layers can extract richer feature details through higher input resolution. Input resolution is connected to width and height of input image. First going through smaller (224x224) input resolution to higher (300x300) input resolutions, convolutional layers bind higher resolution image pixels to create feature which has better classification accuracy. Excessive input resolution causes poor accurracy increasement over time.

Compound scaling

In figure 5 compound scaling is depicted as width scaling, depth scaling and resolution scaling of baseline layer dimension parameters. The scaling of layer dimension parameters in a neural architecture is interdependent. For higher input resolution, increasing the network depth scaling enhances feature gain of neuron channel units and related neuron channel units capture similar feature patterns in images with higher pixel densities. This situation necessitates a joint approach to scaling, as scaling only one dimension without scaling other layer dimensions result in inadequate model performance. Increasing the neuron channel unit count connected to layer depth and input resolution increasement. For instance, going through from EfficientNet-b0 to EfficientNet-b7 architecture, layer depth and input resolution is increased together with convolution blocks of neuron channel units.

Balancing the layer depth, width and resolution dimensions of the neural network structure is critical for more effective accuracy performance. The compound scaling method scales the layer depth, width and resolution dimensions of the neural network structure consistently with the help of the compound coefficient. Neural network layer depth, width and resolution dimension constants can be determined by "grid search" research. The compound coefficient is the dynamic value that controls the resource distribution in the scaling of the neural network structure. Depth, width and resolution dimension constants are the values that determine how the resource distribution will be transferred to the network depth, width and resolution [16].

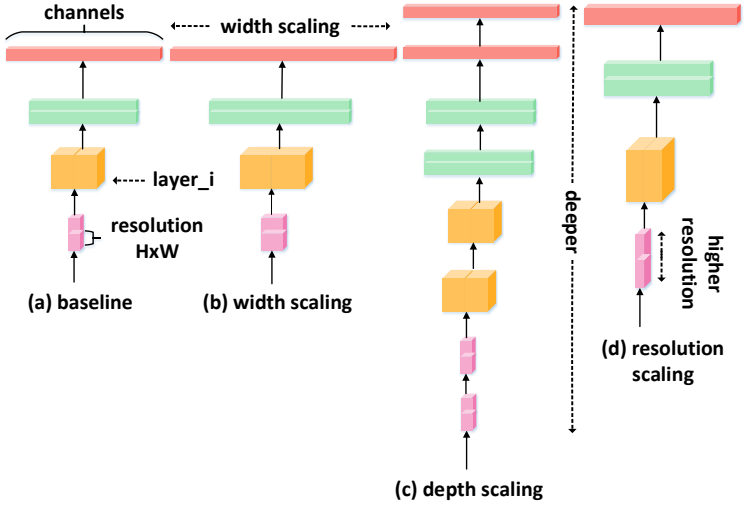

Figure 5. Compound scaling

3.1.3. Dynamic distribution adaptation network (DDAN) and CORAL transfer learning algorithm

For improving fluctuation of validation datasets and model's generalization ability, we examine research studies about transfer learning algorithms and domain adaptations. In this research we chose dynamic distribution adaptation network with CORAL loss applied to backbone neural network.

In effectiveness analysis of transfer learning for the concept drift problem in malware detection research study, malware samples were divided into source domain and target domain fields via temporal split. VirusShare dataset's malware samples from the years 2015, 2017, 2019 and 2020 were transferred to source domain and target domain. Source domain contains malware samples belong to specific earlier year band, while target domain contains malware samples belong to later year band. Transfer learning algorithms were applied to the relevant source domain and target domain. Especially CORAL transfer learning algorithm approximates the feature covariance of the source domain to the feature covariance of the target domain. This is achieved by whitening the source data (reducing the feature correlation to 0) and activating the covariance of the target domain. The original feature domain is not changed. In the related research study, transfer learning algorithms were presented to develop new malware detections despite the insufficiency of labeled malware samples [17]. CORAL loss defines transferable measure of target domain and source domain in dynamic distribution adaptation network.

In transfer learning with dynamic distribution adaptation research study, Dynamic Distribution Adaptation (DDA) method is presented. Dynamic Distribution Adaptation method evaluates the quantitative weight of each feature distribution of data domains. Dynamic Distribution Adaptation can participate in the structural risk minimization of feature transitions in solution of transfer learning problems. The research study proposed Manifold Dynamic Distribution Adaptation (MDDA) for traditional transfer learning and Dynamic Distribution Adaptation Network (DDAN) learning algorithms for deep transfer learning on the basis of Dynamic Distribution Adaptation. Especially in deep transfer learning, the Dynamic Distribution Adaptation Network (DDAN) performs end-to-end learning of the feature g(.) learning function and the classification function f. DDAN learns feature representations with the end-to-end training cycle of deep neural networks. Backbone network applies domain adaptation with DDA method while learning useful feature representations [18].

In the Dynamic Distribution Adaptation Network (DDAN) architecture, data samples from source domain and target domain are input to deep neural networks. CNN networks such as AlexNet and ResNet extract high-level features from the data samples. The high-level features pass through the fully-connected layer and are assigned to the softmax classification. The unique architectural part is the convergence of the feature distributions of the source domain and the target domain using the dynamic distribution alignment. The DDAN architecture incorporates the mini-batch Stochastic Gradient Descent (SGD) algorithm into the deep neural network training cycle. Dynamic distribution adaptation is computed over batch parts of the domain [18].

Deep and adversarial transfer learning

Deep transfer learning has improved with more enhanced feature extraction capability of deep neural networks and parametric functions such as loss functions or optimization functions. Especially loss functions have become evaluation model for transfering source domain knowledge to feature separation of target domain. In addition, adaptation of source domain and target domain has realized through loss function. For instance, the Deep Domain Confusion (DDC) method introduced MMD loss into deep networks, facilitating adaptation between domains. Similarly, Deep Adaptation Networks (DAN) integrated a multi-kernel MMD framework based on first-order formulation while the Deep CORAL network included CORAL loss based on second-order formulation. CORAL loss has usage in Dynamic Distribution Adaptation Network and measures adaptation of source domain and target domain during network training cycle. Apart from Dynamic Distribution Adaptation Network and loss functions such as CORAL, MMD, the adversarial learning approach promotes the learning of representative feature characteristics that have more potential to transfer between target domain and source domain. The Domain Adversarial Neural Network (DANN) uses domain adversarial loss rather than relying on loss functions like MMD connected to specific Maximum Mean Discrepancy distance distribution formulation. In this situation network can learn more distinctive feature characteristics between domains.

3.2. Datasets

3.2.1. Dumpware10 dataset

The Dumpware10 dataset produced by Hacettepe University was created for the detection of malware with an image-based approach. It was combined with image descriptors such as GIST and Histogram of Gradients (HOG). Four different resolutions ranging from 224 to 4096 pixels were used in the creation of malware images belonging to the Dumpware10 dataset. GIST and HOG image descriptors were evaluated both separately and together within the scope of information fusion. UMAP, a dimensional reduction and multi-faceted learning technique, was used within the scope of malware image transformation problems. This dataset has a total of 11 classes, including 10 malware families and one benign software. The Dumpware10 dataset has a total of 4294 data samples, 3433 training and 861 validation samples. The dataset contains files belonging to 10 different malware families, including Adposhel, Allaple.A, Amonetize, AutoRun-PU, BrowseFox, Dinwod, InstallCore.C, MultiPlug, VBA, and Vilsel.

3.2.2. Microsoft malware classification challenge dataset

Microsoft Malware Classification Challenge is a dataset for malware classification. The dataset provides 10868 malware byte files as training data. Each byte file contains raw byte sequences representing a specific type of malware. This dataset helps researchers develop malware detection algorithms. The data is particularly suitable for research aimed at malware classification based on static file features.

4. PROPOSED MODEL

4.1. Data Preprocessing

In the data preprocessing of the research, malware byte files belong to the Microsoft Malware Classification Challenge (BIG 2015) dataset are subjected to two-channel grayscale malware image conversion and three-channel RGB malware image conversion. Connected malware image conversions have paralel implementation as B2IMG algorithm.

In figure 6 parsing of .bytes files, calculation of (a,b) data array size, converting sized data arrays to 8-bit 2D format and saving 8-bit 2D formats as (256x256) sized .jpg files phases of two-channel grayscale malware image conversion are depicted. During two-channel grayscale malware image conversion, byte files are assigned to 8-bit two-channel grayscale format by converting to (a,b) matrix modeling over a 16-column array data and saved in fixed-size image files with .jpg or .png extensions [19].

Data arrays are sized to 256x256 for fixed-size grayscale malware image generation.

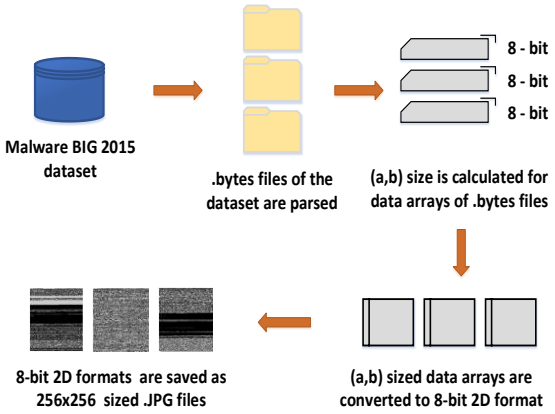

Figure 6. Two-channel Grayscale malware image conversion

In figure 7 parsing of .bytes files, dividing bytes data into RGB channels, combining RGB channels into meaningful RGB data and saving RGB data as .png files phases of three-channel RGB image conversion are depicted. During three-channel RGB malware image conversion, the binary data processed from the byte files are divided into triple RGB channel blocks. Then, RGB channel blocks are combined to create new RGB data and the RGB data are saved in .png image files [19].

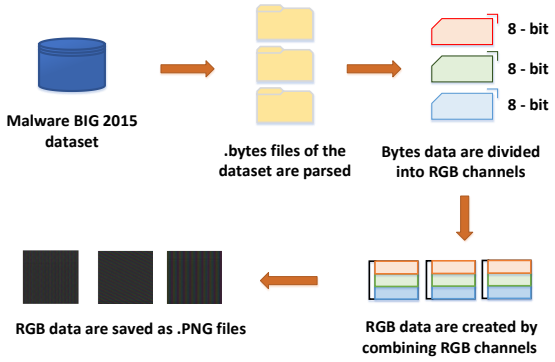

Figure 7. Three-channel RGB malware image conversion

In figure 8 modelling of EfficientNet neural network implementation is partitioned into data processing and EfficientNet feature extraction phases. And connected phases are detailed consecutively with steps and model architecture structures such as layers and functions.
In data preprocessing, the malware dataset is parsed into byte files and converted to 8-bit matrix format and saved to malware images. Malware images are transferred to EfficientNet architecture and subjected to feature extraction. Data preprocessing and EfficientNet feature extraction ends with the softmax function connected to the fully-connected layer. The softmax function assigns the feature extractions to the malware family counterparts. EfficientNet feature extraction produces the classification characteristics of malware images.

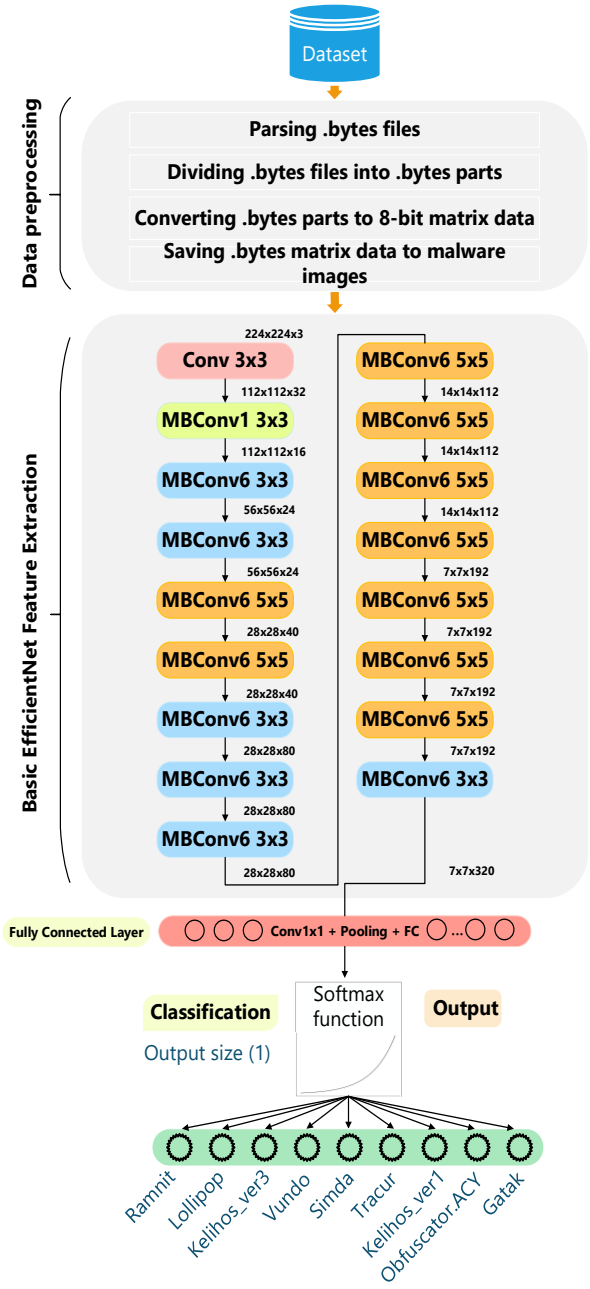

Figure 8. EfficientNet neural network implementation with data preprocessing

5. EXPERIMENTAL RESULTS

10868 malware byte files of the Microsoft Malware Classification Challenge (BIG 2015) dataset are introduced to EfficientNet-b3 and EfficientNet-b4 neural network architectures. Stochastic Gradient Descent (SGD) and Adam optimizations with 0.01 learning rate are used in EfficientNet model trainings. Pre-trained ImageNet-1K dataset weights are used in EfficientNet transfer learning and during experiments Pre-trained ImageNet-1K dataset weights were specifically trained for EfficientNet architectures and pre-trained ImageNet-1K dataset weights knowledge is transferred into EfficientNet malware classification by finetuning. Malware train data are converted to two-channel

grayscale malware images and three-channel RGB malware images and included in the EfficientNet model training cycle. Three-channel RGB malware images and two-channel grayscale malware images are separated into 80% train set and 20% test set. Train set of malware images are trained with 20% validation separation. Grayscale malware images are 256x256 in size. RGB malware images are assigned to 32, 64, 128, 256, 384, 512, 768, 1024 and 224 sizes according to variable file sizes.

Table 1 results show that EfficientNet-b4 architecture trained with grayscale malware images achieves 0.90 accuracy in SGD optimization. EfficientNet-b3 architecture trained with RGB malware images achieves 0.91 accuracy in SGD optimization. Adam optimization shows lower accuracy compared to SGD optimization. Deepening EfficientNet-b3 architecture to EfficientNet-b4 architecture increases the accuracy rate in Adam optimization while decreasing it in SGD optimization for training RGB malware images.

Table 1. EfficientNet-b3 and EfficientNet-b4 models Microsoft Malware Classification Challenge validation accuracies

Malware image	Model	Learning rate	Optimization	Accuracy
grayscale malware	Efficient Net-b3	0.01	SGD	0.897642
grayscalemalware	Efficient Net-b4	0.01	SGD	0.907418
grayscalemalware	Efficient Net-b3	0.01	Adam	0.753306
grayscalemalware	Efficient Net-b4	0.01	Adam	0.840713
rgb malware	Efficient Net-b3	0.01	SGD	0.917194
rgb malware	Efficient Net-b4	0.01	SGD	0.897067
rgb malware	Efficient Net-b3	0.01	Adam	0.826337
rgb malware	Efficient Net-b4	0.01	Adam	0.885566

Table 2 results show that EfficientNet-b3 and EfficientNet-b4 architectures trained with SGD optimization on grayscale malware image type have the highest test accuracy.

Table 2. Microsoft Malware Classification Challenge EfficientNet-b3 and EfficientNet-b4 test accuracies

Malware image	Model	Learning rate	Optimization	Accuracy
grayscale malware	Efficient Net-b3	0.01	SGD	0.89
grayscale malware	Efficient Net-b4	0.01	SGD	0.89
grayscale malware	Efficient Net-b3	0.01	Adam	0.70
grayscale malware	Efficient Net-b4	0.01	Adam	0.80
rgb malware	Efficient Net-b3	0.01	SGD	0.88
rgb malware	Efficient Net-b4	0.01	SGD	0.86
rgb malware	Efficient Net-b3	0.01	Adam	0.80
rgb malware	Efficient Net-b4	0.01	Adam	0.86

Table 2. (cont.) Microsoft Malware Classification Challenge EfficientNet-b3 and EfficientNet-b4 test

We use Microsoft Malware Classification Challenge (MMCC) grayscale malware images belong to the Malware Classification and Visualization Using EfficientNet and B2IMG Algorithm research study. Related grayscale malware images are assigned to equal width and height values of 32, 64, 128, 256, 384, 512, 768, 1024 according to their file sizes [11]. In this research related research study's grayscale malware images had already passed through B2IMG data preprocessing algorithm -paralel to our data preprocessing- just without (256x256) size fixation and are split into 80% train set and 20% test set. Then the grayscale malware images are inputted to EfficientNet-b0...b7 architectures. The train set and test set are distributed into Gatak, Kelihos_ver1, Kelihos_ver3, Lollipop, Obfuscator.ACY, Ramnit, Simda, Tracur, Vundo 9 malware families. Train set of grayscale malware images was trained in 120 epochs with a 20% validation separation. This train cycle has SGD optimization with a 0.01 learning rate. The highest train accuracy was obtained in the EfficientNet-b7 architecture.

In this study during going though EfficientNet pre-transformed Microsoft Malware Classification Challenge (MMCC) RGB malware images are sized to (224x224) random resized crops. Random parts of malware image are cropped and resized to (224x224). By these random resized crops EfficientNet model can interpret better characteristic textural analysis of malware image. RGB malware images are inputted to EfficientNet-b0, EfficientNet-b1, EfficientNet-b2, EfficientNet-b3, EfficientNet-b4, EfficientNet-b5, EfficientNet-b6, EfficientNet-b7 architectures. RGB malware images are separated into 80% train set and 20% test set. In this research, the train set of RGB malware images is trained in 120 epoch cycles with 20% validation separation. This training cycle has SGD optimization with a learning rate of 0.01.

Table 3 shows Microsoft Malware Classification Challenge grayscale malware images achieve the most successful validation accuracy in the EfficientNet-b7 architecture.

Table 3. Validation accuracy of grayscale two-channel malware images in the EfficientNet-b7 architecture

Malware image	Model	Learning rate	Optimization	Accuracy
grayscale malware	EfficientNet-b7	0.01	SGD	0.952792

Table 4 shows Microsoft Malware Classification Challenge RGB malware images achieve the most successful validation accuracy in the EfficientNet-b5 architecture.

Grayscale malware are less noisy than RGB data and focuses on the textural patterns of malware data. In this case, deeper EfficientNet-b7 can capture finer malware image details without complexity. RGB malware has three color channels and is more complex than grayscale malware. In this case, the simpler EfficientNet-b5 can resolve the added channel complexity into meaningful malware image details and realize successful performance metrics.

Table 4. Validation accuracy of three-channel RGB malware images in EfficientNet-b5 architecture

Malware image	Model	Learning rate	Optimization	Accuracy
RGB malware	EfficientNet-b5	0.01	SGD	0.944796

For Dumpware10 malware images, 300-pixel resolution and (300x300) image size were preferred. Dumpware10 malware images were divided into 3433 training sets and 861 test sets and entered into EfficientNet-b0, EfficientNet-b1, EfficientNet-b2, EfficientNet-b3, EfficientNet-b4, EfficientNet-b5, EfficientNet-b6, EfficientNet-b7 architectures. In this research, Dumpware10 malware images were trained in 120 epoch cycles with a 20% validation separation. The training set and the test set were distributed to Adposhel, Allaple, Amonetize, AutoRun, BrowseFox, Dinwod, InstallCore, MultiPlug, Other, VBA, Vilsel classes.

Table 5 shows Dumpware10 RGB malware images achieve the most successful validation accuracy in the EfficientNet-b6 architecture. Dumpware10 RGB images are focused on the simpler EfficientNet architecture than the Microsoft Malware Classification Challenge (MMCC) RGB malware images.

Table 5. Validation accuracy of Dumpware10 RGB malware images on EfficientNet-b6 architecture

Malware image	Model	Learning rate	Optimization	Accuracy
RGB malware	EfficientNet-b6	0.01	SGD	0.934498

In training of grayscale malware images belong to the Malware Classification and Visualization Using EfficientNet and B2IMG Algorithm study, the EfficientNet-b7 architecture reaches the highest accuracy with a value of 95%. In training of RGB malware images belong to our research study, the EfficientNet-b5 architecture reaches the highest accuracy with a value of 94%. The EfficientNet-b6 architecture has the highest accuracy with of 93% for RGB malware images of the Dumpware10 dataset.

Table 6 shows EfficientNet-b7 SGD optimization of Microsoft Malware Classification Challenge grayscale images achieve the highest success values.

In figure 9 train loss of Microsoft Malware Classification Challenge (Grayscale) EfficientNet-b7 and in figure 10 train accuracy of Microsoft Malware Classification Challenge (Grayscale) EfficientNet-b7 is depicted. In figure 11 validation accuracy and in figure 12 validation loss of Microsoft Malware Classification Challenge (Grayscale) EfficientNet-b7 is depicted. The validation loss and accuracies are more zigzag shaped. In this case, train data approaches a more consistent curve by being memorized in the model while validation data is more inconsistent as a result of overfitting. The model training has difficulty generalizing to unpredictable validation data. The overfitting status of the train data is reflected in the validation data.

Table 6. Microsoft Malware Classification Challenge and Dumpware10 test accuracy, F1-score, recall and precision results

	Model	Malware image	Learning Rate	Optimization	Accuracy	F1 Score	Recall	Precision
MS	EfficientNet-b7	Gray Scale	0.01	SGD	0.93	0.8963	0.9163	0.8815
	EfficientNet-b5	RGB	0.01	SGD	0.91	0.8646	0.8777	0.8550
Dumpware10	EfficientNet-b6	RGB	0.01	SGD	0.91	0.8786	0.8843	0.8811

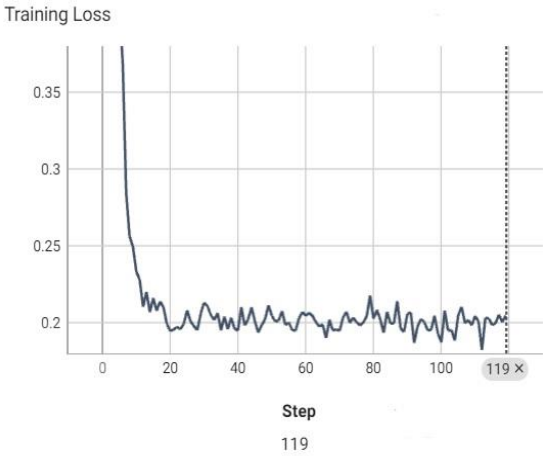

Figure 9. Microsoft Malware Classification Challenge (Grayscale) EfficientNet-b7 train loss

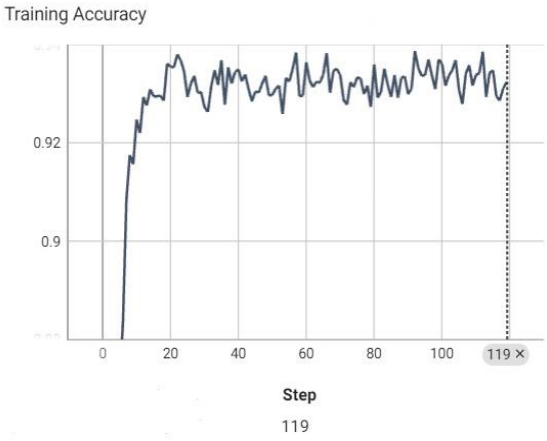

Figure 10. Microsoft Malware Classification Challenge (Grayscale) EfficientNet-b7 train accuracy

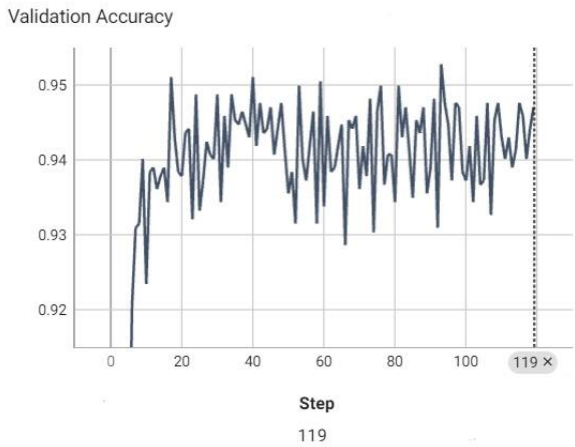

Figure 11. Microsoft Malware Classification Challenge (Grayscale) EfficientNet-b7 validation accuracy

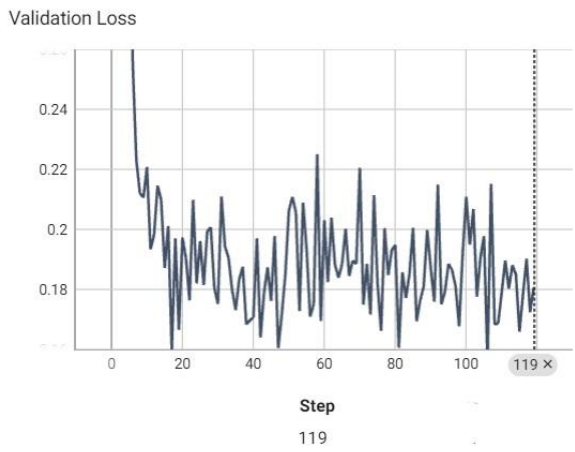

Figure 12. Microsoft Malware Classification Challenge (Grayscale) EfficientNet-b7 validation loss

In figure 13 confusion matrix of Microsoft Malware Classification Challenge (Grayscale) EfficientNet-b7 is depicted. When Microsoft Malware Classification Challenge (Grayscale) EfficientNet-b7 confusion matrix is examined, the lowest accuracy is seen in the Simda class, while the highest accuracy is seen in the Kelihos_ver1 class. In particular, the fact that the Simda class has a lower accuracy is due to insufficiency of data for the Simda class.

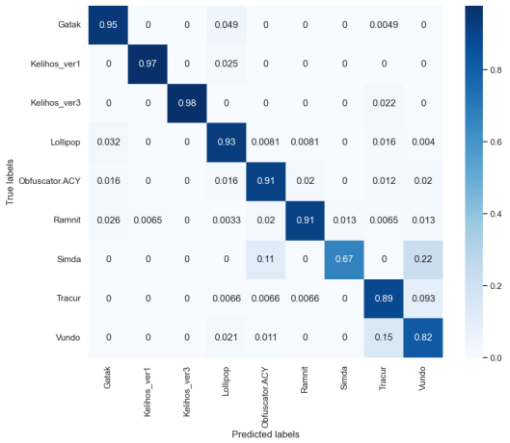

Figure 13. Microsoft Malware Classification Challenge (Grayscale) EfficientNet-b7 confusion matrix

In figure 14 train loss of Microsoft Malware Classification Challenge (RGB) EfficientNet-b5 and in figure 15 train accuracy of Microsoft Malware Classification Challenge (RGB) EfficientNet-b5 is depicted. In figure 16 validation loss and in figure 17 validation accuracy of Microsoft Malware Classification Challenge (RGB) EfficientNet-b5 is depicted. When the figures are examined, it is seen that the validation loss and accuracies are more inconsistent than the train loss and accuracies. This situation transforms the overfitting problem of the train data into a zigzag curve in the validation data. RGB malware images are more consistent compared to grayscale malware images.

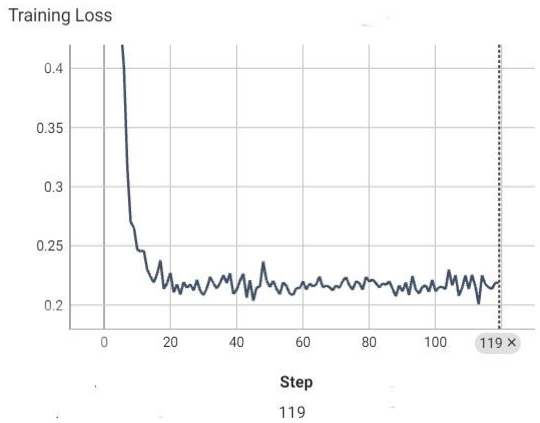

Figure 14. Microsoft Malware Classification Challenge (RGB) EfficientNet-b5 train loss

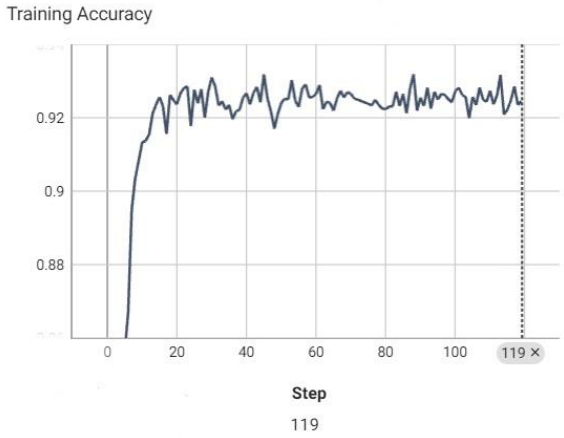

Figure 15. Microsoft Malware Classification Challenge (RGB) EfficientNet-b5 train accuracy

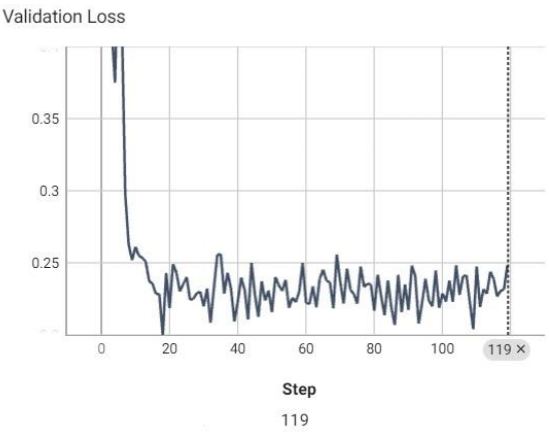

Figure 16. Microsoft Malware Classification Challenge (RGB) EfficientNet-b5 validation loss

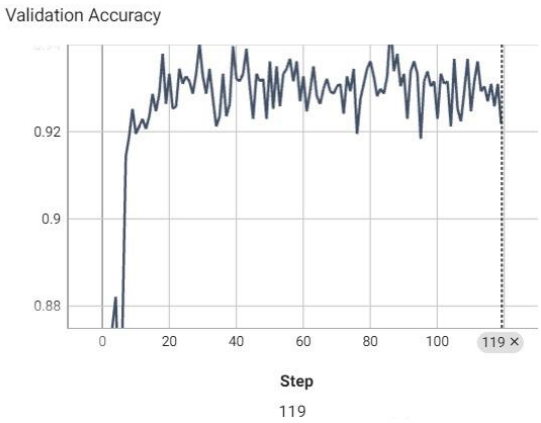

Figure 17. Microsoft Malware Classification Challenge (RGB) EfficientNet-b5 validation accuracy

In figure 18 confusion matrix of Microsoft Malware Classification Challenge (RGB) EfficientNet-b5 is depicted. When the Microsoft Malware Classification Challenge (RGB) EfficientNet-b5 confusion matrix is examined, the lowest accuracy belongs to the Simda class while the highest accuracy belongs to the Kelihos_ver3 class. This is due to the imbalance in the data distribution.

Figure 18. Microsoft Malware Classification Challenge (RGB) EfficientNet-b5 confusion matrix

In figure 19 validation loss of Dumpware10 EfficientNet-b6 and in figure 20 validation accuracy of Dumpware10 EfficientNet-b6 is depicted. In figure 21 train loss and in figure 22 train accuracy of Dumpware10 EfficientNet-b6 is depicted. When Dumpware10 EfficientNet-b6 validation accuracies and losses and train accuracies and losses are examined, it is shown that the train data has reached overfitting and the validation data has difficulty in generalization. In the model training, the batch parts can not learn the malware image characteristics in the gradient loop. This creates inconsistent curves in the validation data.

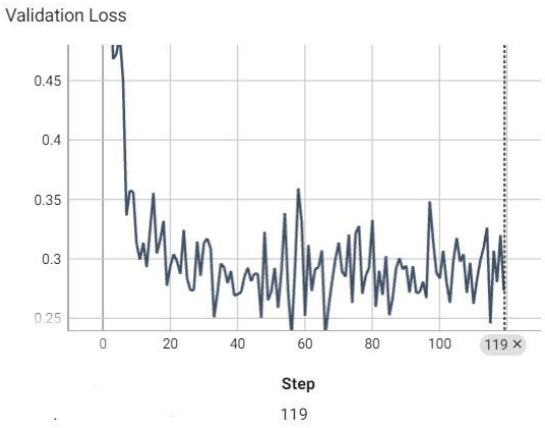

Figure 19. Dumpware10 EfficientNet-b6 validation loss

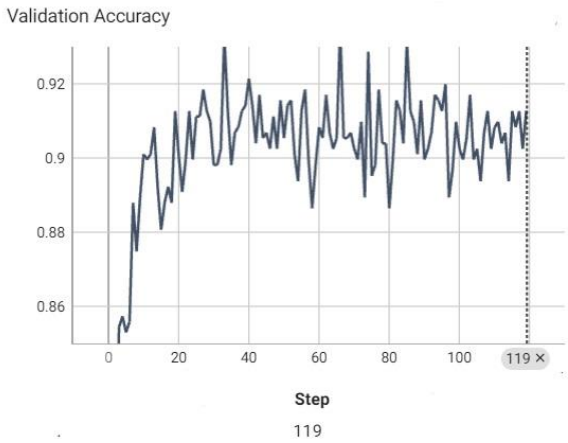

Figure 20. Dumpware10 EfficientNet-b6 validation accuracy

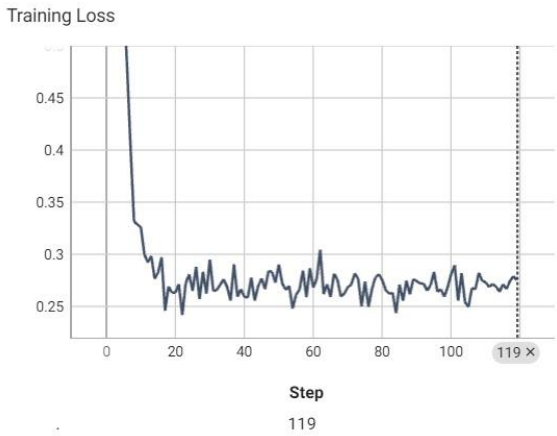

Figure 21. Dumpware10 EfficientNet-b6 train loss

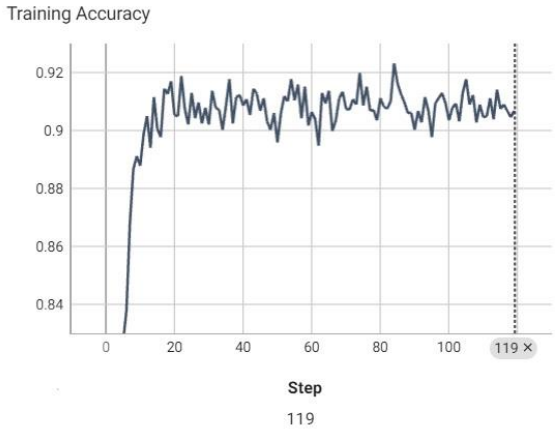

Figure 22. Dumpware10 EfficientNet-b6 train accuracy

In figure 23 confusion matrix of Dumpware10 EfficientNet-b6 is shown. In Dumpware10 EfficientNet-b6 confusion matrix, the lowest accuracy belongs to Dinwod class while the highest accuracy belongs to VBA class. Imbalance of data distribution between classes causes this issue.

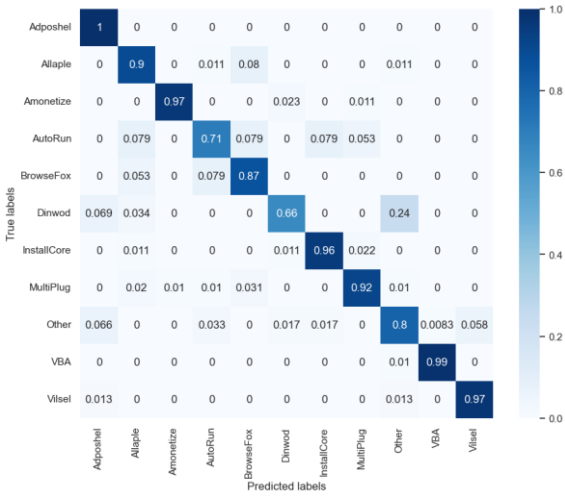

Figure 23. Dumpware10 EfficientNet-b6 confusion matrix

In transfer learning approach for malware classification research study, a convolutional transfer learning application (TL-CNN) was utilized for Android malware image classification. The transfer learning convolutional neural network architecture separated benign images and malicious malware images. In the transfer learning model of the research work, the pre-trained ResNet-50 classifier component was replaced with the original classification component. The classification component removed the ImageNet 1000 image classes and integrated the fully-connected layer contains 25 malware classes. In the transfer learning ResNet-50 model, the MaxPooling layer with padding, (2x2) kernel filter size and 2 stride is introduced before the fully-connected layer of the classification component. The fully-connected layer and

the Softmax function performed the final malware classification [20].

In this research, based on the transfer learning approach for malware images classification, the last Global Average Pooling layer of the feature extraction block of the EfficientNet architecture is replaced with the MaxPooling layer and connected to the classifier. EfficientNet architecture can be adapted to the MaxPooling layer with (7x7) kernel size and 1 step. The classifier classifier of the EfficientNet architecture requires (1x1) plane-sized input and the EfficientNet architecture does not meet the Max Pooling setting with (2x2) kernel filter size and 2 steps.

MaxPooling EfficientNet-b7 (Microsoft Malware Classification Challenge - grayscale malware image) and EfficientNet-b5 (Microsoft Malware Classification Challenge - RGB malware image) EfficientNet-b6 (Dumpware10 dataset) models were trained with 120 epoch cycles in SGD optimization with 0.01 learning rate.

Table 7 shows EfficientNet-b5 (RGB) has the highest evaluation accuracy among MaxPooling EfficientNet model trainings. When the MaxPooling change occurred, RGB malware images produced higher validation accuracy than grayscale malware images.

Table 7. MaxPooling EfficientNet model validation accuracies

Malware image	Model	Learning rate	Optimization	Accuracy
Grayscale malware	EfficientNet-b7	0.01	SGD	0.918250
RGB malware	EfficientNet-b5	0.01	SGD	0.926394
RGB malware	EfficientNet-b6	0.01	SGD	0.903930

Table 8 shows EfficientNet-b7 (Grayscale) has the highest test accuracy. The fact that the EfficientNet-b7 precision value is higher than the recall value shows that the correct class matching is high in the total correct class inference of the model.

Table 8. MaxPooling EfficientNet model experiment test accuracy F1-score, recall and precision results

Model	Malware Image	Learning rate	Optimization	Accuracy	F1-score	Recall	Precision
EfficientNet-b7	Grayscale	0.01	SGD	90%	0.8529	0.8434	0.8893
EfficientNet-b5	RGB	0.01	SGD	89%	0.8288	0.8407	0.8207
EfficientNet-b6	RGB	0.01	SGD	88%	0.8299	0.8275	0.8469

In figure 24 confusion matrix of MaxPooling EfficientNet-b5 Microsoft Malware Classification Challenge (RGB) is depicted with class accuracy distributions. In MaxPooling EfficientNet-b5 Microsoft Malware Classification Challenge (RGB) confusion matrix, the highest accuracy belongs to the Kelihos_ver3 class, while the lowest accuracy belongs to the Simda class. This is due to the data imbalance between classes. Simda class has insufficient number of data sample.

Figure 24. MaxPooling EfficientNet-b5 Microsoft Malware Classification Challenge (RGB) confusion matrix

In figure 25 MaxPooling EfficientNet-b7 Microsoft Malware Classification Challenge (Grayscale) confusion matrix shows that the accuracy of the Simda class is the lowest, while the accuracy of the Kelihos_ver3 class is the highest. Negative matching shifts are observed in the Simda class, and this breaks the accuracy.

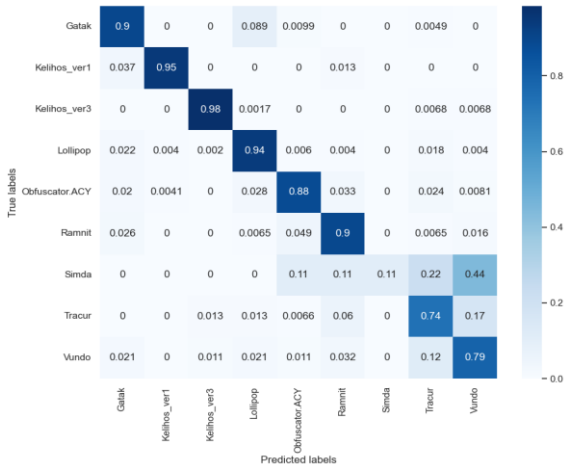

Figure 25. MaxPooling EfficientNet-b7 Microsoft Malware Classification Challenge (Grayscale) confusion matrix

In figure 26 MaxPooling EfficientNet-b6 Dumpware10 (RGB) confusion matrix shows that the highest accuracy belongs to the VBA class, while the lowest accuracy

belongs to the AutoRun class. The unbalanced distribution of Dumpware10 malware images among the classes affects accuracy.

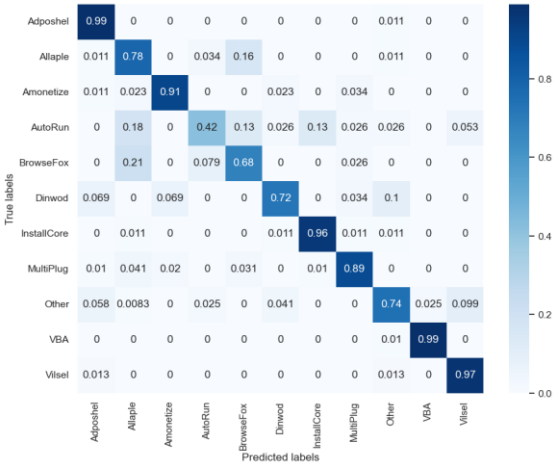

Figure 26. MaxPooling EfficientNet-b6 Dumpware10 (RGB) confusion matrix

MaxPooling experiment results shows that EfficientNet-b7 (Grayscale) extension of the Transfer Learning experiment has the highest success performance. The MaxPooling experiment did not provide a noticeable improvement in test accuracies and the GlobalAveragePooling layer of the EfficientNet architecture produced more successful results than the MaxPooling layer.

CORAL transfer learning algorithm uses the source domain and target domain distinctions of the dataset under domain adaptation. Dumpware10, Microsoft Malware Classification Challenge RGB and Microsoft Malware Classification Challenge grayscale datasets are divided into source domain with 60% and target domain with 40%. Then the source domain and target domains are divided into “test domain” and “validation domain” with a rate of 20%. The source domain, target domain and validation domain of the datasets participate in the training of the Dynamic Distribution Adaptation Network with CORAL loss in domain partition logic of CORAL transfer learning algorithm.

The backbone deep neural network (ResNet) in the Dynamic Distribution Adaptation Network extension is replaced by the EfficientNet architecture of this research work. Dumpware10 domain data is input to EfficientNet-b6, Microsoft Malware Classification Challenge RGB domain data is input to EfficientNet-b5 and Microsoft Malware Classification grayscale domain data is input to EfficientNet-b6. The training cycle of EfficientNet architectures has SGD optimization with EarlyStopping support and 0.01 learning rate. Batch size is 16. A bottleneck layer has been added to the Dynamic Distribution Adaptation Network Backbone architecture, improving performance. Dynamic Distribution Adaptation Backbone architecture assigns CORAL loss of source domain and target domain to transfer loss, CrossEntropy loss to classification loss, and the sum of transfer loss and classification loss to total loss. In the Dynamic Distribution Adaptation training cycle, the source domain and target domain get closer to each other and the transfer loss decreases.

Table 9 shows that Microsoft Malware Classification Challenge grayscale EfficientNet-b6 backbone model has the highest validation accuracy rate.

Table 9. Dynamic distribution adaptation network with CORAL loss experiment validation accuracies

Malware dataset	Malware image	Backbone model	Accuracy
Microsoft Malware Classification Challenge	Grayscale	EfficientNet-b6	0.9706
Microsoft Malware Classification Challenge	RGB	EfficientNet-b5	0.9442
Dumpware10	RGB	EfficientNet-b6	0.9663

Table 10 shows that the most successful results in test domain accuracy, precision, recall and F1-score metrics belong to Dumpware10 RGB EfficientNet-b6 model.

Table 10. Dynamic distribution adaptation network with CORAL loss experiment test tesults

Malware dataset	Malware image	Backbone model	Accuracy	Precision	Recall	F1-score
Microsoft Malware Classification Challenge dataset	Grayscale	EfficientNet-b6	0.95854	0.9009	0.8554	0.8647
Microsoft Malware Classification Challenge dataset	RGB	EfficientNet-b5	0.95257	0.9484	0.8478	0.8596
Dumpware10	RGB	EfficientNet-b6	0.96064	0.9460	0.9556	0.9491

In figure 27 validation accuracy of DDAN with CORAL loss experiment Microsoft Malware Classification Challenge Grayscale EfficientNet-b6 is depicted and in figure 28 DDAN with CORAL loss experiment Microsoft Malware Classification Challenge (Grayscale) EfficientNet-b6 classification loss is depicted. Figure 29 and figure 30 consecutively show transfer loss and total loss of DDAN with CORAL loss experiment Microsoft Malware Classification Challenge (Grayscale) EfficientNet-b6. When connected figures are examined, we see that the validation accuracy figure contains less zigzag changes. This situation demonstrates consistency of training. Also there is a sudden decreasement of accuracy that connected to model anomaly. However EarlyStopping mechanism had been already saved the model weight that highest accuracy rate. Therefore model weight did not influenced by sudden drop in accuracy.

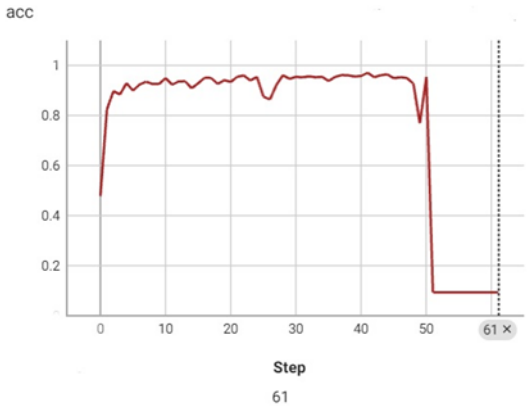

Figure 27. DDAN with CORAL loss experiment Microsoft Malware Classification Challenge (Grayscale) EfficientNet-b6 validation accuracy

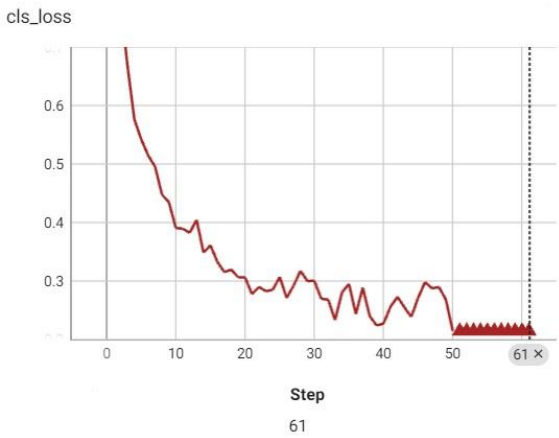

Figure 28. DDAN with CORAL loss experiment Microsoft Malware Classification Challenge (Grayscale) EfficientNet-b6 classification loss

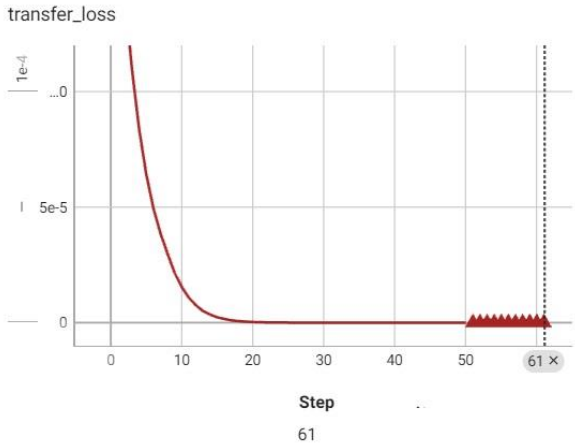

Figure 29. DDAN with CORAL loss experiment Microsoft Malware Classification Challenge (Grayscale) EfficientNet-b6 transfer loss

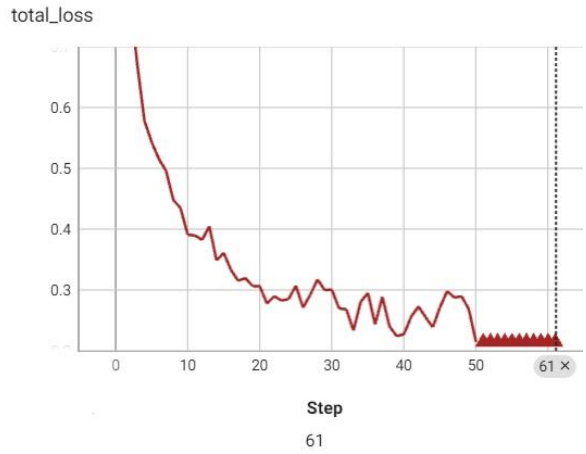

Figure 30. DDAN with CORAL loss experiment Microsoft Malware Classification Challenge (Grayscale) EfficientNet-b6 total loss

In figure 31 validation accuracy of DDAN with CORAL loss experiment Microsoft Malware Classification Challenge (RGB) EfficientNet-b5 is depicted and in figure 32 DDAN with CORAL loss experiment Microsoft Malware Classification Challenge (RGB) EfficientNet-b5 classification loss is depicted. Figure 33 and figure 34 consecutively show transfer loss and total loss of DDAN with CORAL loss experiment Microsoft Malware Classification Challenge (RGB) EfficientNet-b5. When connected figures are examined, we see that fluctiations in validation accuracy figure is lesser than the previous experiments' figures. Also examined that there is a abrupt drop in validation accuracy which shows a model anomaly. However EarlyStopping mechanism had been already saved mode weights that include the highest accuracy rate. Therefore sudden drop in validation accuracy did not affect the model weights.

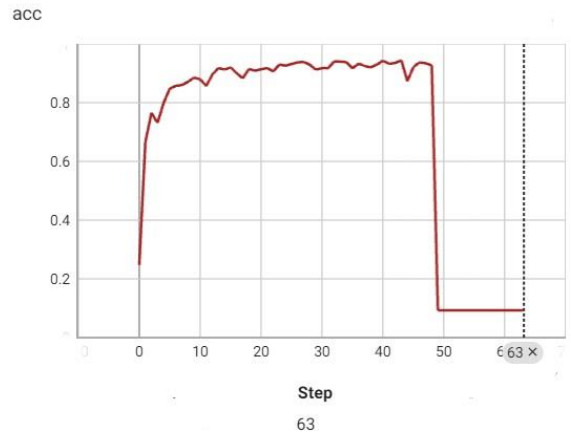

Figure 31. DDAN with CORAL loss experiment Microsoft Malware Classification Challenge (RGB) EfficientNet-b5 validation accuracy

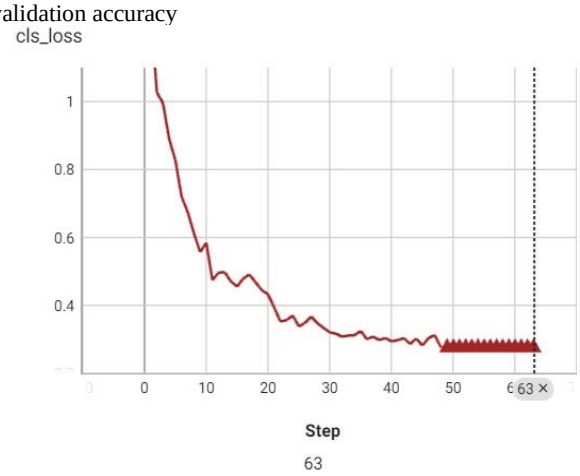

Figure 32. DDAN with CORAL loss experiment Microsoft Malware Classification Challenge (RGB) EfficientNet-b5 classification loss

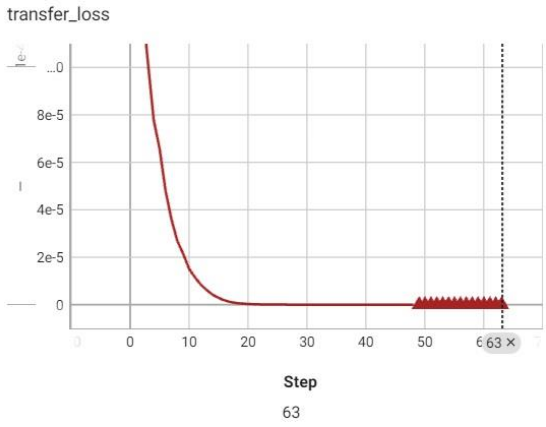

Figure 33. DDAN with CORAL loss experiment Microsoft Malware Classification Challenge RGB EfficientNet-b5 transfer loss

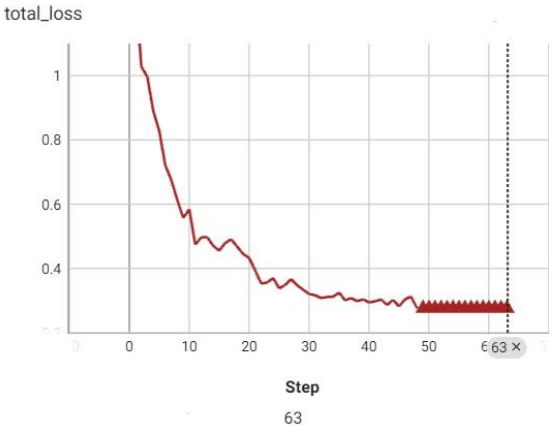

Figure 34. DDAN with CORAL loss experiment Microsoft Malware Classification Challenge RGB EfficientNet-b5 total loss

In figure 35 validation accuracy of DDAN with CORAL loss experiment Dumpware10 EfficientNet-b6 is depicted and in figure 36 DDAN with CORAL loss experiment Dumpware10 EfficientNet-b6 classification loss is depicted. Figure 37 and figure 38 consecutively show transfer loss and total loss of DDAN with CORAL loss experiment Dumpware10 EfficientNet-b6. When connected figures are examined, we see that validation accuracy figure graph has less fluctiations beside the previous experiments' figure graphs excluding DDAN with CORAL loss experiment. However sudden accuracy drop did not happened in DDAN with CORAL loss experiment Dumpware10 EfficientNet-b6.

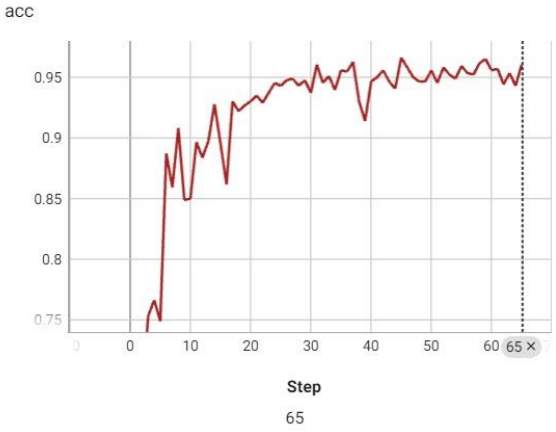

Figure 35. DDAN with CORAL loss experiment Dumpware10 EfficientNet-b6 validation accuracy

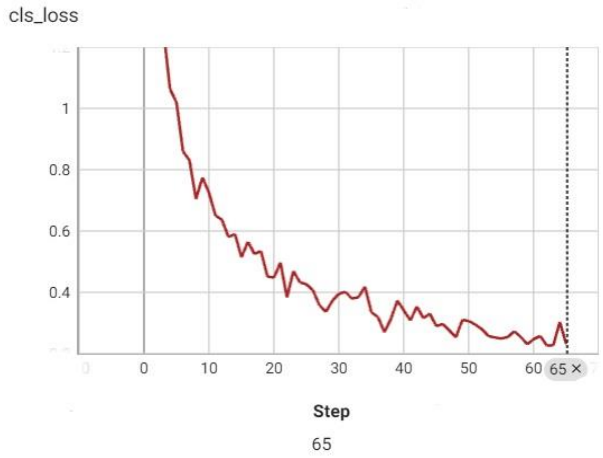

Figure 36. DDAN with CORAL loss experiment Dumpware10 EfficientNet-b6 classification loss

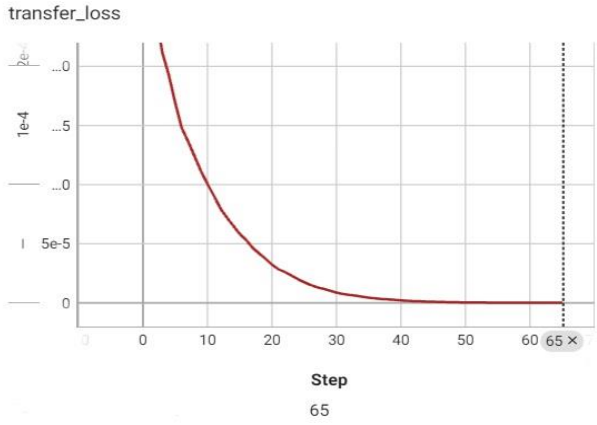

Figure 37. DDAN with CORAL loss experiment Dumpware10 EfficientNet-b6 transfer loss

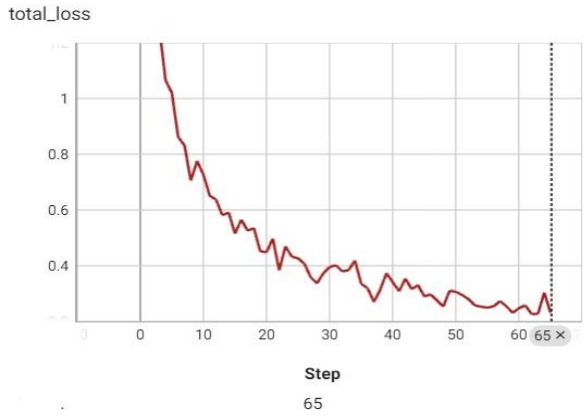

Figure 38. DDAN with CORAL loss experiment Dumpware10 EfficientNet-b6 total loss

In figure 39 DDAN with CORAL loss Microsoft Malware Classification Challenge (Grayscale) EfficientNet-b6 confusion matrix, we see that the highest accuracy is in Kelihos_ver3 and the lowest accuracy is in Simda.

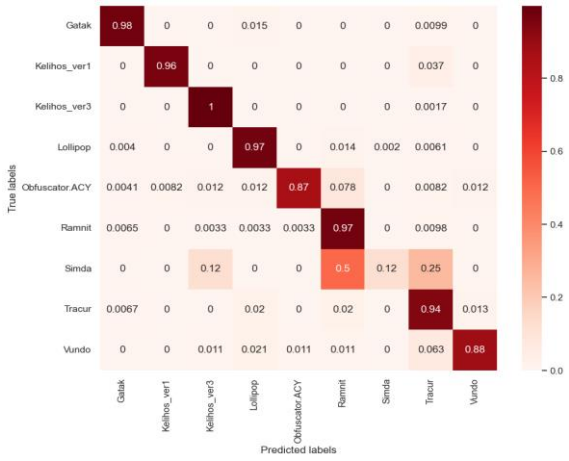

Figure 39. DDAN with CORAL loss experiment Microsoft Malware Classification Challenge (Grayscale) EfficientNet-b6 confusion matrix

In figure 40 DDAN with CORAL loss Microsoft Malware Classification Challenge (RGB) EfficientNet-b5 confusion matrix, we see that Kelihos_ver3 has the highest accuracy while Simda has the lowest accuracy.

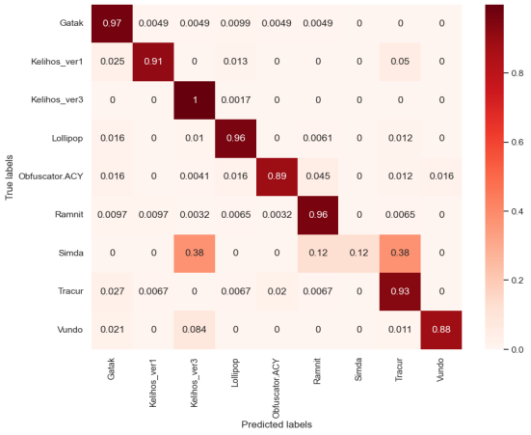

Figure 40. DDAN with CORAL loss experiment Microsoft Malware Classification Challenge (RGB) EfficientNet-b5 confusion matrix

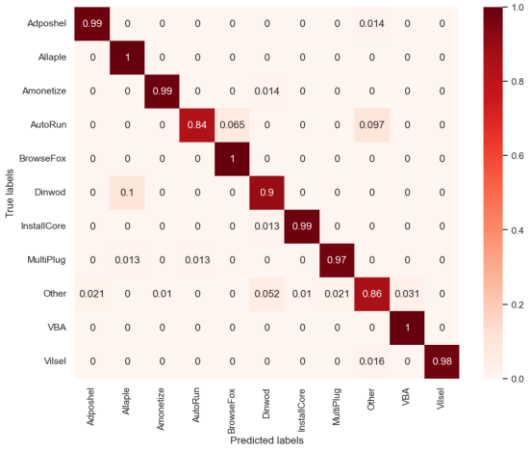

Figure 41. DDAN with CORAL loss experiment Dumpware10 EfficientNet-b6 confusion matrix

In figure 41 DDAN with CORAL loss experiment Dumpware10 EfficientNet-b6 confusion matrix, we see that Allaple, BrowserFox and VBA have the highest accuracies while AutoRun has lowest accuracy.

In table 11 summarizes the highest performance metrics for the Transfer Learning and MaxPooling experiments conducted in the research. The experiments were focused on malware classification using different EfficientNet models (EfficientNet-b5, b6, and b7) applied to both grayscale and RGB malware images. The datasets used for evaluation include the Microsoft Malware Classification Challenge (MMCC) and Dumpware10. The table presents key performance metrics such as test accuracy, F1 Score, recall, and precision, along with validation accuracy. EfficientNet-b7 (Grayscale) achieved the highest validation accuracy (95.27%) in the transfer learning experiments, with strong test accuracy (93%) and high precision (88.15%). EfficientNet-b6 (RGB) on the Dumpware10 dataset performed well across both transfer learning and DDAN experiments, achieving the best test accuracy (96%) and strong precision (94.60%) with a validation accuracy of 96.63%. MaxPooling experiments yielded slightly lower results than transfer learning, with EfficientNet-b7 (Grayscale) showing a test accuracy of 90%. DDAN approach improved the test and validation accuracy metrics across both RGB and grayscale images, particularly excelling in handling the Dumpware10 dataset.

Table 11. Most successful results of the Transfer Learning and MaxPooling experiments

Transfer Learning	Model	Malware image	Test Accuracy	F1 Score	Recall	Precision	Validasyon Accuracy
MS	EfficientNet-b7	Grayscale	0.93	0.8963	0.9163	0.8815	0.952792
	EfficientNet-b5	RGB	0.91	0.8646	0.8777	0.8550	0.944796
Dumpware10	EfficientNet-b6	RGB	0.91	0.8786	0.8843	0.8811	0.934498
MaxPooling							
MS	EfficientNet-b7	Grayscale	0.90	0.8529	0.8434	0.8893	0.918250
	EfficientNet-b5	RGB	0.89	0.8288	0.8407	0.8207	0.926394
Dumpware10	EfficientNet-b6	RGB	0.88	0.8299	0.8275	0.8469	0.903930
DDAN							
MS	EfficientNet-b5	RGB	0.95	0.8596	0.8478	0.9484	0.9442
	EfficientNet-b6	Grayscale	0.95	0.8647	0.8554	0.9009	0.9706
Dumpware10	EfficientNet-b6	RGB	0.96	0.9491	0.9556	0.9460	0.9663

In Table 12 compares the results obtained from various experiments, including Transfer Learning, MaxPooling, and the Dynamic Distribution Adaptation Network (DDAN), for malware classification. EfficientNet-b7 (grayscale) achieved the highest validation accuracy (95.27%) for the Microsoft Malware Classification Challenge. The RGB-based EfficientNet-b5 performed slightly lower with a validation accuracy of 94.48%. MaxPooling technique did not yield significantly better results compared to Transfer Learning. For grayscale images, EfficientNet-b7 scored 91.82% validation accuracy, and for RGB images, EfficientNet-b5 achieved 92.63%. DDAN approach produced the highest results across the board. EfficientNet-b6 (grayscale) reached a validation accuracy of 97.06%, while for RGB images, EfficientNet-b5 and EfficientNet-b6 achieved validation accuracies of 94.42% and 96.63%, respectively.

Table 12. Comparison of academic studies with DDAN experiment results

Studies	Dataset	Accuracy	DDAN EfficientNet-b6 (Grayscale) accuracy rate
[11]	MMCC	99.63% (train)	97% (validation) 95% (test)
[2]	MMCC	99.06% (train)	97% (validation) 95% (test)
[1]	MMCC	99.44%	97% (validation) 95% (test)
[4]	MMCC	99.58%	97% (validation) 95% (test)
			DDAN EfficientNet-b6 (RGB) accuray rate
[19]	Dumpware10	99.60%	96% (validation) 96% (test)
[21]	Dumpware10	97%	96% (validation) 96% (test)

6. CONCLUSIONS

In this study, a comprehensive research was conducted on the classification and visualization of malware using a model based on the EfficientNet deep learning architecture. The model created with EfficientNet's inverse residual block configuration achieved high accuracy rates, especially in the classification of malware images. Experiments conducted on the Microsoft Malware Classification Challenge and Dumpware10 datasets used in the study show that the model gives successful results. 97% validation accuracy and 95% test accuracy were obtained in the Microsoft Malware Classification Challenge dataset, and 96% validation and test accuracy were obtained in the Dumpware10 dataset. EfficientNet's deep learning architecture showed high performance in classification studies on malware images. In particular, the methods used in the preprocessing of data and image transformations made significant contributions to the success of the model. The balanced scaling of EfficientNet's layer depth, width and resolution dimensions increased the accuracy of the model. This study reveals that deep learning techniques can be used effectively in malware detection in the field of cyber security. The flexibility of the EfficientNet architecture and the use of transfer learning techniques have increased the accuracy and generalization ability of the model. In future studies, the performance of the model can be further improved with different datasets and more complex malware detection methods. In conclusion, this study shows that the EfficientNet architecture can be successfully applied in the field of cybersecurity and can provide solutions for malware detection.

DECLARATION OF ETHICAL STANDARDS

The author(s) of this article declare that the materials and methods used in this study do not require ethical committee permission and/or legal-special permission.

AUTHORS' CONTRIBUTIONS

Adem TEKEREK: Wrote the manuscript and performed the model experiments and experiments' analysis, Data preprocessing process and guided through study research background and model experiments.

Ceren Umay ÖZTEN: Wrote the manuscript and performed the model experiments and experiments' analysis, Data preprocessing process and guided through study research background and model experiments.

CONFLICT OF INTEREST

There is no conflict of interest in this study.

REFERENCES

[1] Deng H., Guo C., Shen G., Cui Y., and Ping Y., "MCTVD: A malware classification method based on three-channel visualization and deep learning", ***Computers & Security***, 126, (2023).

[2] Ahmed M., Afreen N., Ahmed M., Sameer M. and Ahamed J., "An inception V3 approach for malware classification using machine learning and transfer learning", ***International Journal of Intelligent Networks***, 4: 11-18, (2023).

[3] Kumar S. and Panda K., "SDIF-CNN: Stacking deep image features using fine-tuned convolution neural network models for real-world malware detection and classification", ***Applied Soft Computing***, 146, (2023).

[4] Bu S.-J. and Cho S.-B., "Malware classification with disentangled representation learning of evolutionary triplet network", ***Neurocomputing***, 552, (2023).

[5] Chaganti R., Ravi V. and Pham T. D., "A multi-view feature fusion approach for effective malware classification using Deep learning", ***Journal of Information Security and Applications***, 72, (2023).

[6] Baek S., Jeon J., Jeong B. and Jeong Y.-S., "Two-stage hybrid malware detection using Deep learning", ***Human-centric Computing and Information Sciences***, 11, (2021).

[7] Kumar M., "Scalable Malware Detection System Using Distributed Deep Learning", ***Cybernetics and Systems***, 54: 619–647, (2022).

[8] Xing X., Jin X., Elahi H., Jiang H. and Wang G., "A malware detection approach using autoencoder in deep learning", ***IEEE Access***, 10: 25696-25706, (2022).

[9] Alomari E. S., Nuiaa R. R., Alyasseri Z. A. A., Mohammed H. J., Sani N. S., Esa M. I. and Musawi B. A., "A. Malware detection using deep learning and correlation-based feature selection", ***Symmetry***, 15:123, (2023).

[10] Vinayakumar R., Alazab M., Soman K. P., Poornachandran P. and Venkatraman S., "Robust intelligent malware detection using deep learning", ***IEEE Access***, 7: 46717-46738, (2019).

[11] Pratama H. Y. and Sidabutar J., "Malware classification and visualization using EfficientNet and B2IMG algorithm", ***2022 International Conference on Advanced Computer Science and Information Systems (ICACSIS)***, Depok, Indonesia, 75-80, (2022).

[12] Oyucu S., Polat O., Türkoğlu M., Polat H., Aksöz A. and Ağdaş M. T., "Ensemble Learning Framework for DDoS Detection in SDN-Based SCADA Systems", ***Sensors***, 24: 155, (2024).

[13] Polat O., Türkoğlu M., Polat H., Oyucu S., Üzen H., Yardımcı F. and Aksöz A., "Multi-Stage Learning Framework Using Convolutional Neural Network and Decision Tree-Based Classification for Detection of DDoS Pandemic Attacks in SDN-Based SCADA Systems", ***Sensor***s, 24: 1040, (2024).

[14] Iman M., Arabnia H. R. and Rasheed K., "A review of deep transfer learning and recent advancements", ***Technologies***, 11: 40, (2023).

[15] Sandler M., Howard A., Zhu M., Zhmoginov A., and Chen L.-C., "Mobilenetv2: Inverted residuals and linear bottlenecks", ***2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)***, Salt Lake City, Utah, 4510-4520, (2018).

[16] Tan M. and Le Q., "EfficientNet: Rethinking model scaling for convolutional neural networks", ***International Conference on Machine Learning***, Long Beach Convention Center, California, 6105-6114, (2019).

[17] Escudero García D., DeCastro-García N. and Muñoz Castañeda A. L., "An effectiveness analysis of transfer learning for the concept drift problem in malware detection", ***Expert Systems with Applications***, 212, (2023).

[18] Wang J., Chen Y., Feng W., Yu H., Huang M. and Yang Q., "Transfer learning with dynamic distribution adaptation", ***ACM Transactions on Intelligent Systems and Technology***, 11: 6, (2020).

[19] Tekerek A. and Yapici M. M., "A novel malware classification and augmentation model based on convolutional neural network", ***Computers & Security***, 112, (2022).

[20] Bala Z., Zambuk F. U., Imam B. Y., Gital A. Y., Shittu F., Aliyu M. and Abdulrahman M. L., "Transfer learning approach for malware images classification on Android devices using deep convolutional neural network", ***Procedia Computer Science***, 212: 429-440, (2022).

[21] Prawiranata F. P. S. and Hadiprakoso R. B., "Comparison of Transfer Learning Performance in Image-Based Malware File Classification on the Dumpware10 Dataset", ***2023 IEEE International Conference on Cryptography, Informatics, and Cybersecurity (ICoCICs)***, Bogor, Indonesia, 252-257, (2023).