applied sciences

MDPI

Article

Detection of Exceptional Malware Variants Using Deep Boosted Feature Spaces and Machine Learning

Muhammad Asam [1], Shaik Javeed Hussain [2,*], Mohammed Mohatram [2], Saddam Hussain Khan [1], Tauseef Jamal [1], Amad Zafar [3,*], Asifullah Khan [1], Muhammad Umair Ali [4,*] and Umme Zahoora [1]

1 Pattern Recognition Lab, Department of Computer & Information Sciences, Pakistan Institute of Engineering & Applied Sciences (PIEAS), Nilore, Islamabad 45650, Pakistan; asim2k994@gmail.com (M.A.); hengrshkhan822@gmail.com (S.H.K.); jamal@pieas.edu.pk (T.J.); asif@pieas.edu.pk (A.K.); farqualeet59@gmail.com (U.Z.)
2 Department of Electrical and Electronics, Global College of Engineering and Technology, Muscat 112, Oman; m.mohatram@gcet.edu.om
3 Department of Electrical Engineering, The Ibadat International University, Islamabad 54590, Pakistan
4 Department of Unmanned Vehicle Engineering, Sejong University, Seoul 05006, Korea
* Correspondence: s.javeedhussain@gcet.edu.om (S.J.H.); amad.zafar@iiui.edu.pk (A.Z.); umair@sejong.ac.kr (M.U.A.)

Abstract: Malware is a key component of cyber-crime, and its analysis is the first line of defence against cyber-attack. This study proposes two new malware classification frameworks: Deep Feature Space-based Malware classification (DFS-MC) and Deep Boosted Feature Space-based Malware classification (DBFS-MC). In the proposed DFS-MC framework, deep features are generated from the customized CNN architectures and are fed to a support vector machine (SVM) algorithm for malware classification, while, in the DBFS-MC framework, the discrimination power is enhanced by first combining deep feature spaces of two customized CNN architectures to achieve boosted feature spaces. Further, the detection of exceptional malware is performed by providing the deep boosted feature space to SVM. The performance of the proposed malware classification frameworks is evaluated on the MalImg malware dataset using the hold-out cross-validation technique. Malware variants like Autorun.K, Swizzor.gen!I, Wintrim.BX and Yuner.A is hard to be correctly classified due to their minor inter-class differences in their features. The proposed DBFS-MC improved performance for these difficult to discriminate malware classes using the idea of feature boosting generated through customized CNNs. The proposed classification framework DBFS-MC showed good results in term of accuracy: 98.61%, F-score: 0.96, precision: 0.96, and recall: 0.96 on stringent test data, using 40% unseen data.

Keywords: malware classification; detection; deep learning; deep features; convolutional neural networks; transfer learning; SVM

Citation: Asam, M.; Hussain, S.J.; Mohatram, M.; Khan, S.H.; Jamal, T.; Zafar, A.; Khan, A.; Ali, M.U.; Zahoora, U. Detection of Exceptional Malware Variants Using Deep Boosted Feature Spaces and Machine Learning. *Appl. Sci.* **2021**, *11*, 10464. https://doi.org/10.3390/app112110464

Received: 24 September 2021
Accepted: 31 October 2021
Published: 8 November 2021

1. Introduction

Software designed with a malicious purpose to harm users or systems falls under the category of malware. Malware may harm the system without user knowledge of any level of damage; it may range from gaining system access, deleting files, ransom demands or even complete sabotage. An substantial increase in credential harvesting using malware and well-established tactics has been noted in the recent past. Just during the course of the COVID-19 pandemic, Microsoft reported 16 different state-level actors who targeted commercial and academic institutions for stealing vaccine-related research knowledge. These threat actors have rapidly become more sophisticated over the past years. They are skilled, persistent and can launch attacks which are harder to spot. The AV-TEST Institute reported more than one billion infected files in January 2021 [1]. These malware files are often morphed into different combinations and variations to evade detection. Anti-

malware techniques are employed for protection against malware invasion. In this regard, categorization of a malware sample is necessary for the precise anti-malware solution [2].

Generally, the challenge of malware classification is addressed using static and dynamic feature extraction-based techniques. The static malware classification approach tries to find registered signatures in the files. In this approach, malware is identified by a sequence of bytes known as a malware signature. File hashes are also used for malware detection, generally resulting in a lower false positive (FP) rate [3,4]. However, these signatures can be altered to evade detection from malware scanners. Signature-malware detection is fast and effective in detecting malware but incapable of recognizing newly released malware [5]. Malware obfuscation is another challenge faced by the static malware classification technique. The obfuscation techniques may include code manipulation, instruction substitution, register reassignment and dead-code insertion [6].

Dynamic malware classification examines malicious activities and footprints during execution in a controlled environment [7]. This technique is proficient at detecting obfuscated and new malware. However, since the dynamic techniques look for anomalies in software behaviours at runtime, they can also produce large FPs. Time and excess resource utilization are other drawbacks of behaviour-based classification [8]. Traditional anti-malware techniques are unable to detect complex variants and new attacks. However, classification using image-based malware data has shown to be very promising, whereby the malware samples are first converted into images using standard conversion methods [9]. Consequently, the malware pattern generated in image representations of the samples is then detected using machine learning (ML) techniques.

Recently, ML, specifically its subfield deep learning (DL), has achieved breakthrough accuracy and scalability in various fields [10]. This evolution has played a radical role in shifting from traditional to artificial intelligence-powered anti-malware strategies. DL-based intensive efforts have been utilized for malware analysis. DL methodologies exploit the hierarchical representation of the input sample and thus can solve even more complex malware classification problems. Thus, DL's ability is largely due to the learning of data representations at multiple and deep feature extraction stages [11]. DL algorithms discover representations for the target class in terms of input data distribution, which is performed at multiple levels [12]. Higher-level learns features that are extracted from lower level features. This hierarchical learning makes the computer acquire the complicated concept of the problem by understanding the simpler one. Also, DL, and specifically, its variant deep convolutional neural network (CNN), is very prospective in image-based classification problems. CNNs have been successfully employed to classify malware families [13]. In our work, customized CNNs are used in an end-to-end manner and as a feature extractor. The feature hierarchies learned from customized CNNs are then assigned to SVM for malware classification. Additionally, in order to increase the diversity in the feature space, deep features extracted from customized transfer learning (TL)-based fine-tuned CNN (ResNet-18 and DenseNet-201) are concatenated to get boosted features space. The boosted features space is provided to the traditional ML classifier for exceptional malware detection.

The significant contributions of our work are below:

(1) Two new CNN-based frameworks—DFS-MC and DBFS-MC—are proposed for effective malware family classification using the MalImg dataset. Deep features are extracted from the customized CNNs and individually provided to the SVM classifier in the proposed DBF-MC framework. Furthermore, TL has been introduced in the customized CNN to reduce malware misclassification.

(2) In the proposed DBFS-MC framework, deep boosted features are obtained by fusing the deep features of customized ResNet-18 and DenseNet-201 to detect exceptional malware classes. In the proposed DBFS-MC framework, residual learning and the concept of blocks in CNN are incorporated, exploiting diverse and discriminative enrich information to learn the effective feature representation of malware classes.

(3) The DBFS-MC framework outperformed the customized CNN models by significantly improving the true prediction of the harder-to-classify malware variants (exceptional malware classes) Autorun.K, Swizzor.gen!I, Wintrim.BX and Yuner.A.

The rest of the paper is organized as follows: the next section highlights related work in the field of malware classification. Section 3 explains our novel classification framework methodology, while Section 4 discusses the experimental setup. Section 5 presents the results and discussions of our work. Section 6 concludes the paper.

2. Related Work

Static and dynamic malware analysis techniques have been applied for detection and classification problems. The evolution of ML and DL techniques has opened many horizons for the analysis and prediction of malware analysis using malware binary or hex files. These files have been converted into images because DL has been extensively used on images or grid-like data. In 2011, Nataraj et al. [14] identified the texture features in the malware image file. These files have been obtained by interpreting the byte code of the portable executable (PE) binary file to the grey level of the pixel value in the image. In [14], texture features have been extracted from the malware image using wavelet decomposition and provided to the K-nearest neighbour classifier. This technique achieved 98% accuracy on the benchmark malware dataset consisting of 25 malware families. In [15], the artificial neural network has been trained on the malware texture features and SVM is used for classification.

A lightweight malware classification technique for the Internet-of-Things (IoTs) has been reported in [16]. Lightweight CNN has been employed on channel binaries collected from the IoTs network. This approach reached an accuracy of 94.0% for DDoS malware. Android systems are not far away from malware attacks. A malware detection framework has been proposed for the android environment using neural networks [17].

A hierarchical ensemble neural network is proposed to exploit trace detection in Intel processors [18]. These traces have been converted into images data for training deep CNN. Ni et al. [19] extracted malicious op-code sequences and converted them to images. Virtual data of the malware attacks have been generated using generative adversarial networks [20]. It helped in achieving the capability of zero-day malware attack detection and reported an accuracy of 95.74%. Le et al. [21] proposed a deep learning approach using LSTM-CNN and achieved an accuracy of approximately 98% along with improved time efficiency.

However, generally, the reported techniques lack to tackle the following aspects:

(1) Most of the reported works have been evaluated in terms of accuracy on the validation dataset. However, precision and recall measures are considered as better performance metrics than accuracy for the imbalance dataset. Moreover, evaluation of these performance metrics on the test dataset is needed for the robustness of the detection model.
(2) Previous techniques largely misclassified malware variants like Autorun.K, Swizzor.gen!I, Wintrim.BX, and Yuner.A that feature minor interclass differences.

In this study, we have addressed the above-mentioned research gaps. In this regard, we proposed a feature boosting based DBFS-MC framework for the identification of exceptional malware families and evaluated the results on stringent unseen MalImg dataset in terms of accuracy, precision, recall, and F-score.

3. Methodology

Malware categorization is a complex mapping problem after malware identification. Classification of malware using a single original feature representation is challenging because of the diverse properties of different malware [13]. We have introduced a new classification framework based on deep feature learning and ML techniques for the discrimination of malware families. The proposed malware classification framework can be viewed in two main phases, data augmentation and classification. In the classification phase, four different deep CNN based techniques are employed. These are (i) Deep Feature

Space-based Malware classification (DFS-MC), (ii) the proposed framework Deep Boosted Feature Space-based Malware Classification (DBFS-MC), (iii) implementation of existing CNN; training from scratch and incorporation of TL based fine-tuning.

The well-established CNN architectures are implemented in two ways: Deep feature-based and Softmax probabilistic-based classification. The brief detail of the proposed malware classification setup is shown in Figure 1, while the detailed work is shown in Figure 2 (training phase) and Figure 3 (testing phase).

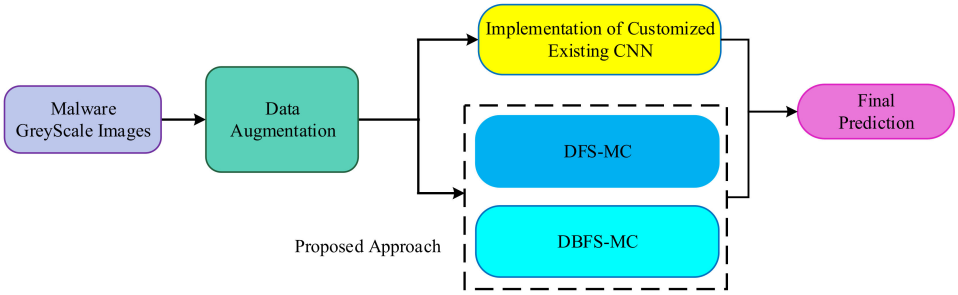

Figure 1. Brief overview of the proposed malware classification framework.

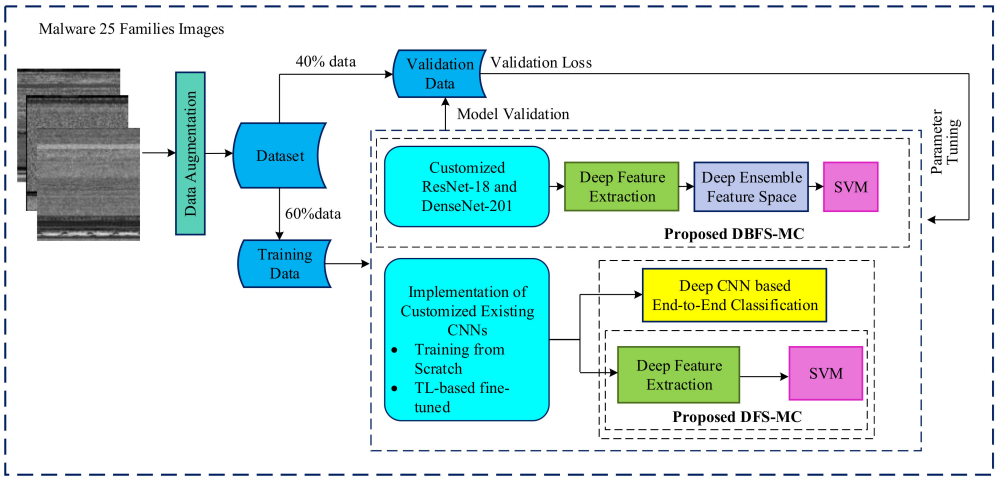

Figure 2. Training phase of malware classification approaches.

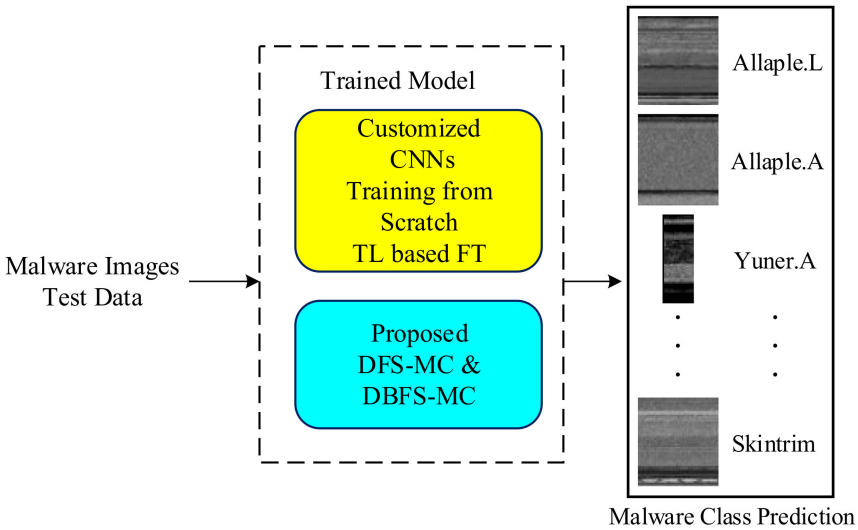

Figure 3. Testing phase of malware classification approaches.

3.1. Data Augmentation

Deep CNN architectures required a large quantity of labelled data. Insufficient training data may lead to poor generalization. Data augmentation is the image dataset that refers to artificially creating new modified samples of the same label by transforming samples [22,23]. Data augmentation is incorporated into the training dataset to improve the generalization and robustness of the proposed classification framework. The implemented augmentation parameters are shown in Table 1.

Table 1. Dataset augmentation details.

Augmentation	Parameters
Rotate	[0, 360] degrees
Scale	[0.5, 1]
Reflection-X	±1
Reflection-Y	±1
Shear	±0.05

3.2. Classification Schemes

The following three schemes are applied for the classification of the malware family.

3.2.1. Implementation of Customized CNN

We customized different standard deep CNN models according to the compatibility of our proposed malware challenge. These customized architectures are DenseNet-201, GoogleNet, InceptionV3, Xception, Resnet-18, ResNet-50, AlexNet, VGG16 and VGG19 [11,24–27] The details of the CNN layers are given in Table 2. Initial and final layers of networks are customized according to the compatibility of the input data set and the target multi-class (25-classes). We implemented the customized existing CNNs in two different ways, softmax probability-based classification and deep feature learning for ML-based classification (DFS-MC). The Softmax probabilistic activation function in CNNs decides the class of the input data. The activation function accepts the input vectors in the form of real numbers and normalizes the inputs to a probability distribution. The output is proportional to the exponentials of its input numbers.

Table 2. Details of deep CNN models.

Models	Depth (Convolutional + Fully Connected Layers)
DesNet-201	203 (201 + 2)
ResNet-18	22 (20 + 2)
Google Net	59 (57 + 2)
Inception-V3	96 (94 + 2)
Xception	75 (74 + 2)
ResNet50	55 (53 + 2)
AlexNet	8 (5 + 3)
VGG-16	16 (13 + 3)
VGG-19	19 (16 + 3)

The customized existing CNNs are trained from scratch and TL-based fine-tuned. Training the CNN model is the only way for adjusting the weights and biases of the model [28]. These architectures are optimized for malware images by training them from scratch on malware datasets by randomly initializing weights from a uniform distribution. Weights are adjusted with the help of optimization algorithms and backpropagation using the training dataset.

Transfer Learning

The CNN models are eager to find a large amount of labelled data for the best training. In case the labelled data in the target domain is limited, the concept TL is useful. TL incorporates the concept of initializing the weights of the models with the parameters

of already trained models. TL gives a good initial estimation of the parameters and reduces the overfitting. This training scheme has shown enhanced performance in the field of malware classification challenges. Pre-trained CNN models of the source domain (Image Classification, ImageNet) are adopted for the target domain (Malware Classification, MalImg) with the help of fine-tuning. This fine-tuning is carried out with the help of additional layers or dissecting the existing layers accordingly. This concept is known as domain adaption [27]. We have introduced the concept of TL in the customized existing deep CNN models and fine-tuned on malware dataset. TL uses the knowledge of the existing domain for a new target domain.

3.2.2. The Proposed Deep Feature Space-Based Malware Classification (DFS-MC)

The malware classification challenge is addressed by a hybrid learning scheme and named as Deep Feature Space-based Malware Classification (DFS-MC). The classification ability of the proposed framework is enhanced by combining the benefits of both empirical and structural risk minimization [29]. CNNs are high-capacity learning models and follow the principles of empirical risk minimization learning theory, which focuses on minimizing training loss. Sometimes, this training may lead to overfitting. Contrary to this, a classical ML classifier like SVM is based on a structural risk minimization principle. This principle improves the SVM's generalization ability by looking at a test error [30]. Therefore, features are extracted from the second last fully connected layer (FC) of customized CNNs and individually provided to SVM for classification. The feature extraction layer and its respective dimensions are given in Table 3.

$$\sigma(\mathrm{v})_i = \frac{e^{V_i}}{\sum_{j=1}^{K} e^{V_j}} \tag{1}$$

$$w^T \mathrm{x} + \mathrm{b} = 0 \tag{2}$$

Table 3. Deep feature matrix dimensions for each CNN model.

Pre-Trained		Trained from Scratch		Deep Boosted	
Feature Layer	Feature Dim.	Feature Layer	Feature Dim.	Feature Layer	Feature Dim.
Last fc	64×25	Last fc	64×25	Last fc	128×25

In Equation (1), σ represents the softmax activation function, whereas v are features from the last layer of customized CNNs and input to softmax. The exponential function for input and normalization vectors are represented by e^{Vi} and e^{Vj}. K represents the number of classes. In Equation (2), x is a feature space, whereas w^T and b are weight and bias, respectively.

3.2.3. The Proposed DBFS-MC Framework

In the aforementioned DFS-MC framework, we have carefully selected the best two customized CNNs, ResNet-18 and DenseNet-201. We have utilized the hybrid learning capacity of these two customized CNN in the proposed deep boosted learning and named the Deep Boosted Feature Space-based Malware Classification (DBFS-MC) framework. In the proposed DBFS-MC approach, ResNet-18 and DensNet-201 are customized and TL based fine-tuned according to our problem space. Deep features arisen in the second last layer classification layer of these customized CNNs are merged to produce a boosted feature space and provided to SVM. An overview of this framework is shown in Figure 4. The proposed deep boosted learning scheme exploited the deep feature learning capability of customized CNN and the discrimination power of SVM.

In Equation (3), $\mathrm{x}_{Boosted}$ shows the boosted feature space that are generated by concatenation of deep feature spaces of $f_{ResNet}(\mathrm{x})$ and $f_{DenseNet}(\mathrm{x})$. These boosted features are then fed to SVM for classification. SVM creates an optimal decision boundary by minimizing the misclassification rate and maximizing the margin between the samples [31]. ζ repre-

sents the misclassified samples, while *C* is the cost that establishes the trade-off between misclassification rate and the model's generalization as depicted in Equations (4) and (5):

$$x_{Boosted} = f_b(f_{ResNet}(x) \mid f_{DensNet}(x)) \tag{3}$$

$$w^T x_{Boosted} + b = 0 \tag{4}$$

$$\min_{w,\xi_i} C \sum_{i}^{N} \zeta_i + \frac{1}{2}\|w\|^2 \tag{5}$$

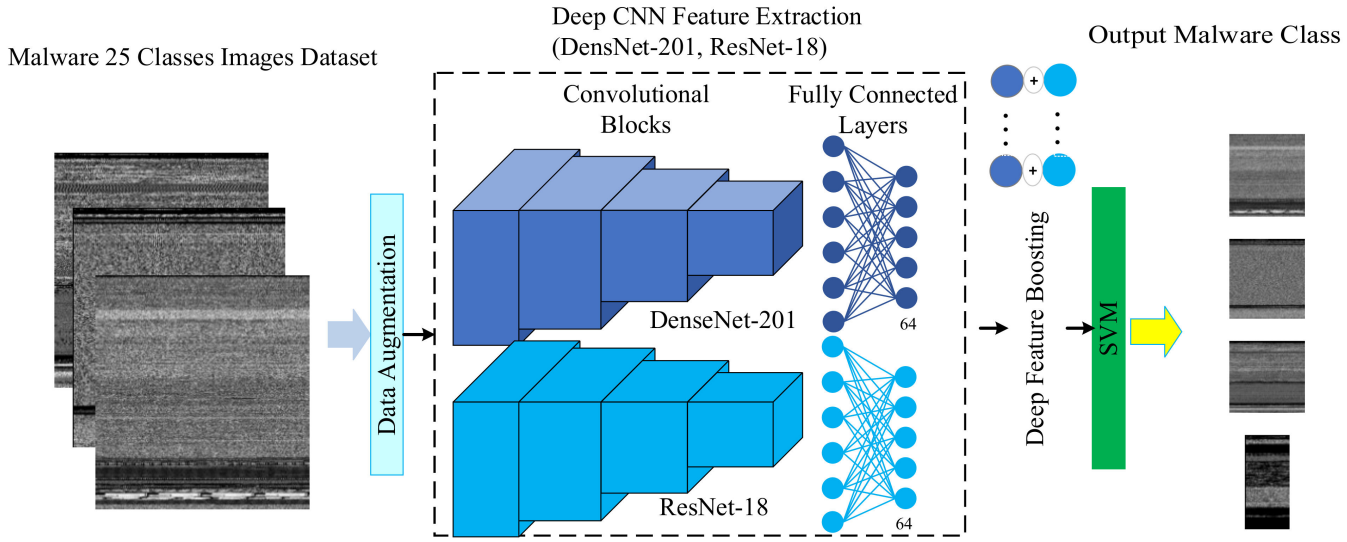

Figure 4. Flow diagram of the proposed DBFS-MC malware classification approach.

In the proposed framework, we exploited the potential of residual and block-based learning to generate prominent features from malware visual images as shown in Figures 5 and 6. The residual and block learning attributes of both the architectures are highly appreciated in the form of experimental results. These two architectures are described below.

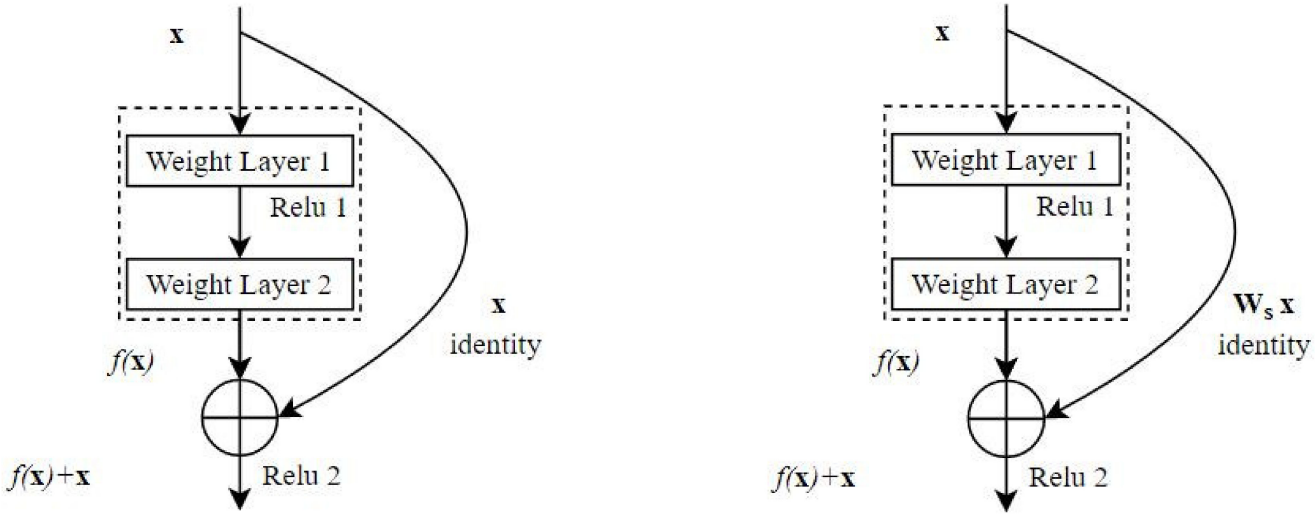

Figure 5. Skip Connection Building Block.

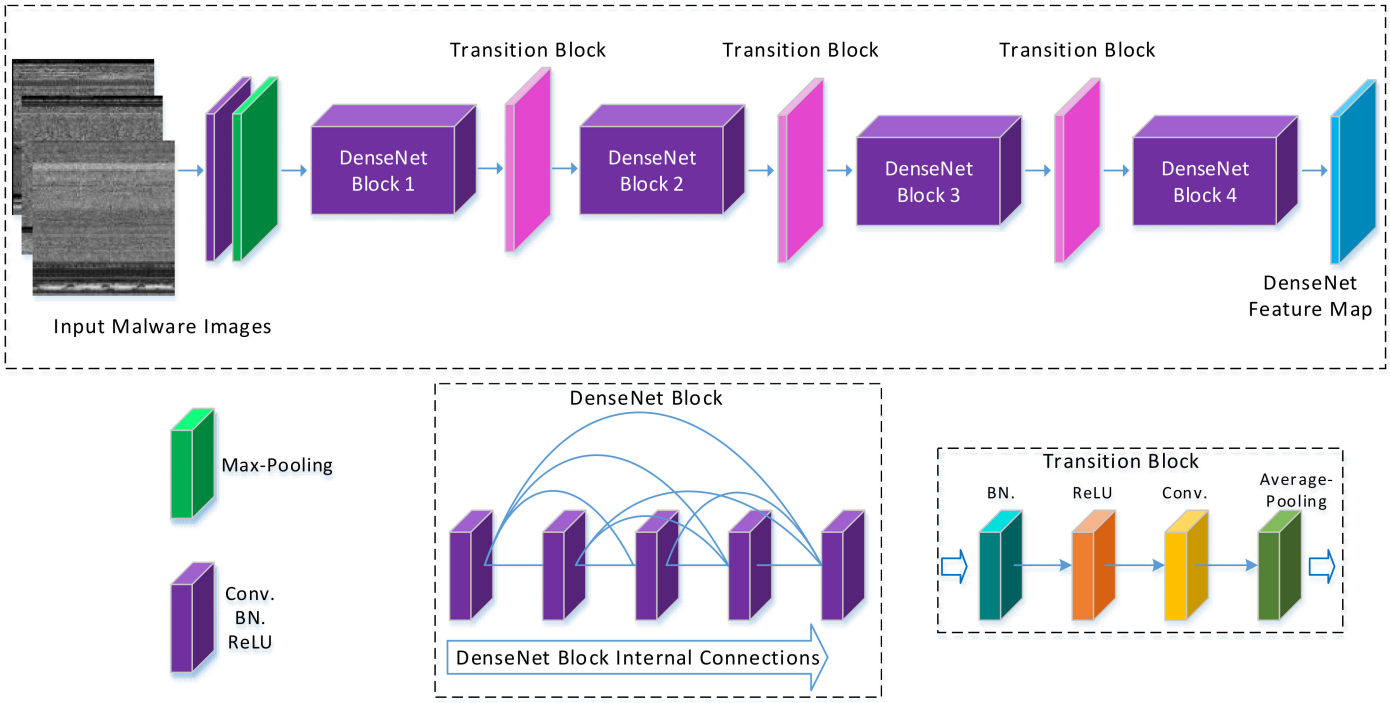

Figure 6. DenseNet-201 realization using channel wise concatenation.

ResNet-18

Residual network (ResNet-18) for the first time solved vanishing gradient problem by introducing skip connections, also known as identity connection [32], as shown in Figure 5. Skip connections help the gradient to flow through an alternate path and allow the model to learn identity hypothesis function. They provide a mean for information flow from earlier to later layers of the model. ResNet-18 uses a bottleneck residual block design to increase the performance of the network. ResNet-18 is 18 layers deep:

$$x_{s,t} = \sum_{a=1}^{m} \sum_{b=1}^{n} x_{s+a-1,t+b-1} w_{a,b} \tag{6}$$

$$y = f(x, \{w_i\}) + x \tag{7}$$

$$y = f(x, \{w_i\}) + w_s x \tag{8}$$

Convolutional operation for the symmetric filter is shown in Equation (6). Here, x represents the input and convolved feature map and w represents the kernel.

Equations (7) and (8) describe the residual operation, where w_i and w_s represents the weight layers of 3×3 and 1×1 (skip connection) convolutional operation, respectively.

DenseNet-201

The DenseNet-201 architecture connects each layer in a feed-forward manner. Features of the proceeding layers are used as input to the current layer, and features extracted in the current layer is used as input to all subsequent layers. A DenseNet network is closer to ResNet as inputs are concatenated instead of addition. This small change has encouraged feature reusability, strengthened feature propagation, reduced the number of parameters and substantially improved vanishing-gradient problem [33]. DenseNet-201 realization for our proposed method is shown in Figure 6. DenseNet-201 follows simple connectivity rules. This help to integrate the identity mapping and deep supervision properties. The internal representation of this network is very compact. Therefore, feature reusability decreased the effect of feature redundancy, so this the reason why DenseNet-201 emerged to be a fine feature extractor for computer vision problems:

$$x^l = H_l(x^0 \left| x^1 \right| x^2, \ldots, x^{l-1}) \quad (9)$$

In Equation (9), $H_l(.)$ is the non-linear transformation, and $(x^0 \left| x^1 \right| x^2, \ldots, x^{l-1})$ is concatenated feature space up to layer l.

4. Experimental Setup

4.1. Dataset

The malware MalImg dataset used consisted of 9342 images of 25 families [14]. Originally the dataset was in binary portable executable format, but CNN is famous for its images-like data. Therefore, these files were converted into grayscale images, and each image was resized to 128 × 128 pixels. The dataset for this experimental setup is imbalanced in nature. The malware class name with their respective number of instances is presented in Figure 7.

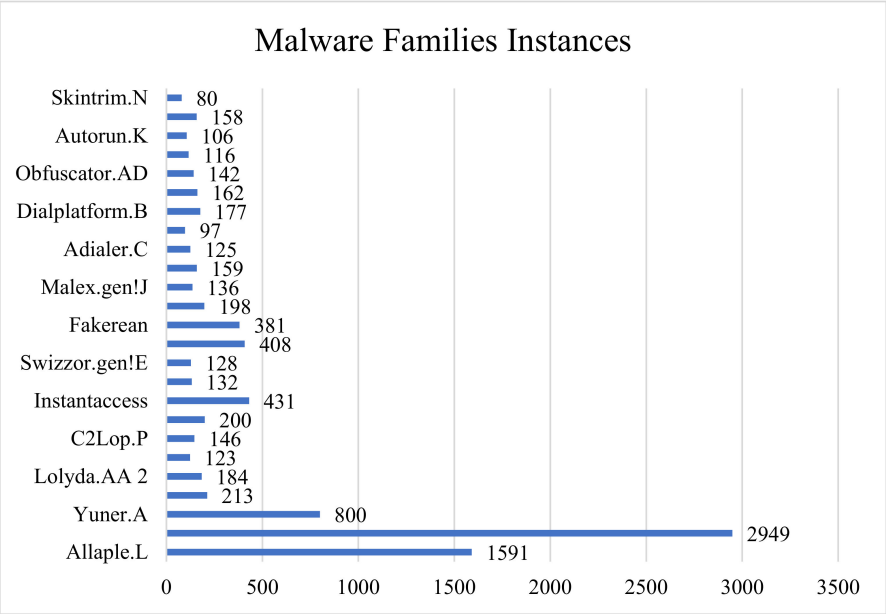

Figure 7. MalImg dataset families.

4.2. Implementation Details

In our work, we employed a 60–40% dataset partitioning scheme for training and testing phases. Usually, 80–20%, 75–25% or sometimes 70–30% training and validation dataset partitioning are present in the literature. In our case, the MalImg, malware-based dataset is limited and highly imbalanced, as shown in Figure 4. Therefore, we used the scheme (60–40) to make our model more robust for unseen malware data. The training samples were further divided for training samples and validation samples for hyper-parameters selection using a hold-out cross-validation scheme. The detail of hyper-parameters is available in Table 4.

Table 4. Hyperparameter details.

Hyperparameters	Values
Optimization Method	SGD
Momentum Value	0.95
Weight Decay	0.0005
Learning Rate	0.0001
Epoches	10
Loss Function	cross-entropy
Batch-size	16

The simulations were carried out on MATLAB-2021a using a Dell Core I i5-7500 with GPU-enabled Nvidia®® GTX 1060-T. It took ~1–2 h to train a model on the said settings. One epoch took 7–10 min on Nvidia-Tesla K-80, while a single malware image took approximately 2 s for analysis.

4.3. Performance Evaluation Metrics

These trained models are evaluated using standard performance metrics, including accuracy, recall precision, and F1-Score as shown in Equations (10)–(13). The details of performance metrics are presented in Table 5. Predicted class refers to the class of interest, and the negative class refers to the combination of the rest of the classes:

$$\text{Acc} = \frac{\text{Predicted Postive Class } + \text{ Predicted Negative Class}}{\text{Total Samples}} \times 100 \tag{10}$$

$$\text{R} = \frac{\text{Predicted Postive Class}}{\text{Total Postive Class Samples}} \times 100 \tag{11}$$

$$\text{P} = \frac{\text{Predicted Postive Class}}{\text{Predicted Postive Class } + \text{Incorrectly Predicted Postive Class}} \times 100 \tag{12}$$

$$\text{F1} - \text{Score} = 2 \times \frac{\text{P} \times \text{R}}{\text{P} + \text{R}} \tag{13}$$

Table 5. Performance metrics details.

Metric	Symbol	Description
Accuracy	Acc	% of total number of correct detections
Recall	R	Proportion of correctly identified classes and actual negative classes
Precision	P	Ratio of correctly detected classes close to the actual class
F1 Score	F1-Score	Harmonic mean of P and R.

5. Results and Discussion

The malware classification experiments are performed using: (i) Softmax probability-based implementation of customized CNN models, (ii) the proposed DFS-MC, and (iii) the proposed DBFS-MC frameworks. These malware classification models are implemented using training from scratch and TL-based fine-tuning on the MalImg dataset. The evaluation of malware classification in terms of accuracy, recall, precision, and F1-Score are reported in Tables 6 and 7.

Table 6. Softmax probabilistic-based implementation of customized CNN for malware classification.

Model	Training Scheme							
	Training from Scratch				Transfer Learning Based			
	Acc %	Recall	Precision	F1-Score	Acc %	Recall	Precision	F1-Score
DenseNet-201	96.57	0.9054	0.9080	0.9067	98.13	0.9411	0.9373	0.9392
Resnet-18	96.41	0.9203	0.9176	0.9189	98.13	0.9416	0.9374	0.9395
GoogleNet	87.20	0.8505	0.8772	0.8637	97.11	0.9178	0.9199	0.9189
Inception-V3	95.72	0.8941	0.9025	0.8983	96.36	0.8905	0.8905	0.8905
Xception	94.48	0.9148	0.9153	0.9150	96.01	0.8809	0.9025	0.8916
ResNet-50	94.86	0.8580	0.8934	0.8753	96.71	0.8984	0.8840	0.8911
AlexNet	88.38	0.8169	0.8643	0.8399	97.91	0.9329	0.9299	0.9314
VGG-16	93.41	0.8525	0.8891	0.8705	97.13	0.9192	0.9268	0.9230
VGG-19	94.97	0.8629	0.8939	0.8782	97.46	0.9259	0.9224	0.9241

It is evident from Table 6 that the TL-based approach outperformed the training from-scratch implementation. The TL-based approach received weights from the pre-trained architectures already trained on a variety of data and then fine-tuned according to our

challenge. Furthermore, the deep features extracted from customized CNNs are fed to ML outperformed softmax-based implementation. This malware classification approach seeks training error minimization. This scheme tailors the model towards the peculiarities of the dataset and may result in weak generalization. The trade-off between training error minimization and improved generalization is leveraged by deep feature extraction and applying conventional ML for classification.

Table 7. Deep feature extracted from customized CNN models and SVM classification (DFS-MC).

Model	Training Scheme							
	Training from Scratch				Transfer Learning Based			
	Acc %	Recall	Precision	F1-Score	Acc %	Recall	Precision	F1-Score
DenseNet	97.70	0.9286	0.9286	0.9286	98.39	0.9483	0.9452	0.9468
ResNet18	98.07	0.9387	0.9368	0.9377	98.37	0.9463	0.9427	0.9445
GoogleNet	97.54	0.9284	0.9232	0.9258	97.60	0.9270	0.9250	0.9259
Inception	97.54	0.9262	0.9268	0.9265	96.95	0.8965	0.9860	0.9391
Xception	97.21	0.9170	0.9141	0.9156	98.31	0.9421	0.9467	0.9444
ResNet50	97.59	0.9289	0.9258	0.9273	97.72	0.9301	0.9284	0.9292
AlexNet	97.59	0.9279	0.9286	0.9282	98.15	0.9417	0.9378	0.9397
VGG16	97.43	0.9246	0.9221	0.9234	97.91	0.9355	0.9304	0.9329
VGG19	97.64	0.9281	0.9284	0.9283	98.23	0.9436	0.9420	0.9428

These experiments concluded that two of the architectures are performing well in our customized framework. Based upon the results from the experiments performed following conclusion can be made based upon accuracy and F1-score.

- DenseNet-201 and ResNet-18 > all other CNNs
- Deep Feature-Based SVM Classification > SoftMax Probabilistic-based Classification
- TL Based Model > Training from Scratch Models

5.1. Performance Analysis of Proposed Frameworks on Exceptional Malware Classes

Exceptional classes in the dataset are those classes for which the discrimination is difficult due to smaller interclass differences. The Softmax-based implementation of customized CNN is performing well for a set of malware classes while missing exceptional malware. Therefore, in the proposed DFS-MC malware analysis approach, we emphasised reducing misclassification for exceptional malware. The customized ResNet-18 and DenseNet-201 correctly classified Autorun.K/Swizzor.gen!I and C2Lop.P/WintrimBX, respectively.

Both the architectures improved the precision and detection rate compared to the existing CNN, as shown in Figures 8 and 9. However, the customized ResNet-18 and DenseNet-201 misclassified some exceptional classes (C2Lop.P/WintrimBX and Autorun.K/Swizzor.gen!I, respectively) when employed individually. Hence, as an outcome of the experiment, we selected two models instead of one to tackle the risk of misclassification.

Therefore, we developed a novel DBFS-MC framework, which significantly reduced the misclassification of exceptional malware classes. The proposed DBFS-MC efficiently learned most of the exceptional malware classes by concatenating the decision of both the networks, as shown in Figure 10. The deep feature boosting helps merge the potentials of customized ResNet-18 and DenseNet-201 architectures while improving the deficiency of the single model. Precision and recall analysis for the exceptional malware classes are 0.94 and 0.95, as shown in the Figure 10.

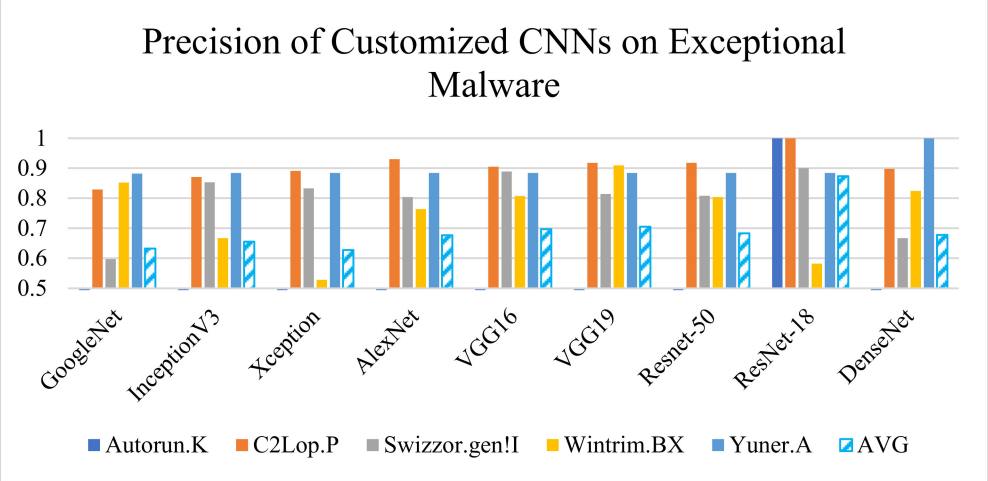

Figure 8. Performance (precision) comparison of implemented CNN models.

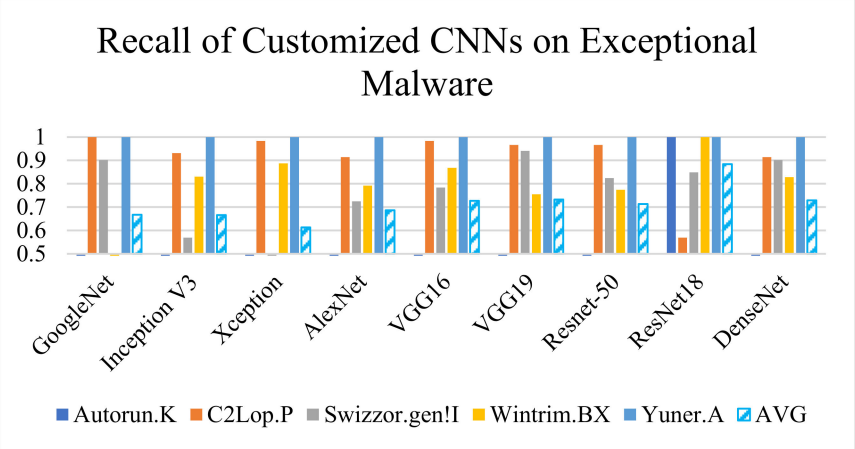

Figure 9. Performance (recall) comparison of implemented CNN models.

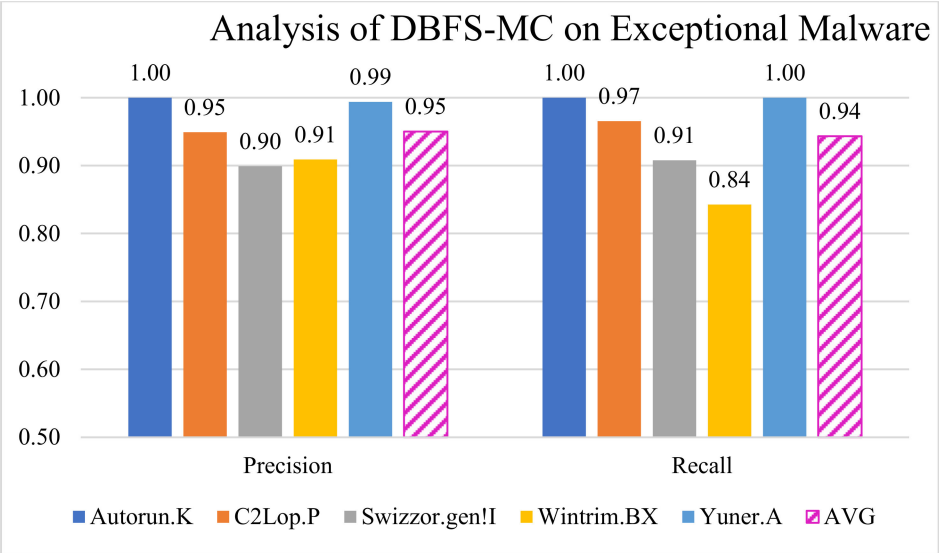

Figure 10. Performance of DBFS-MC for exceptional malware.

5.2. Performance of the Proposed DBFS-MC Framework

We have developed the deep hybrid learning strategy for malware classification, which exploit the learning capability of both deep feature and ML classifier. Despite their outperformance, there exists the risk of misclassification of exceptional malware classes. Therefore, we proposed DBFS-MC deep boosted strategy, which utilizes the diverse and discriminative feature spaces of the customized CNNs. The performance improvement is achieved due to the concatenation of deep feature space of two best performing customized models (ResNet-18 and DenseNet-201). Moreover, the proposed approach (DBFS-MC) achieved the highest classification performance in terms of accuracy (98.61%), recall (0.96), precision (0.96) and F-score (0.96), as shown in Table 8. The performance comparison of DBFS-MC with customized ResNet-18 and DenseNet-201 is also depicted in Figure 11.

Table 8. Performance of the proposed DBFS-MC.

Model	Accuracy %	Recall	Precision	F1-Score
Proposed DBFS-MC	98.61	0.9632	0.9627	0.9630

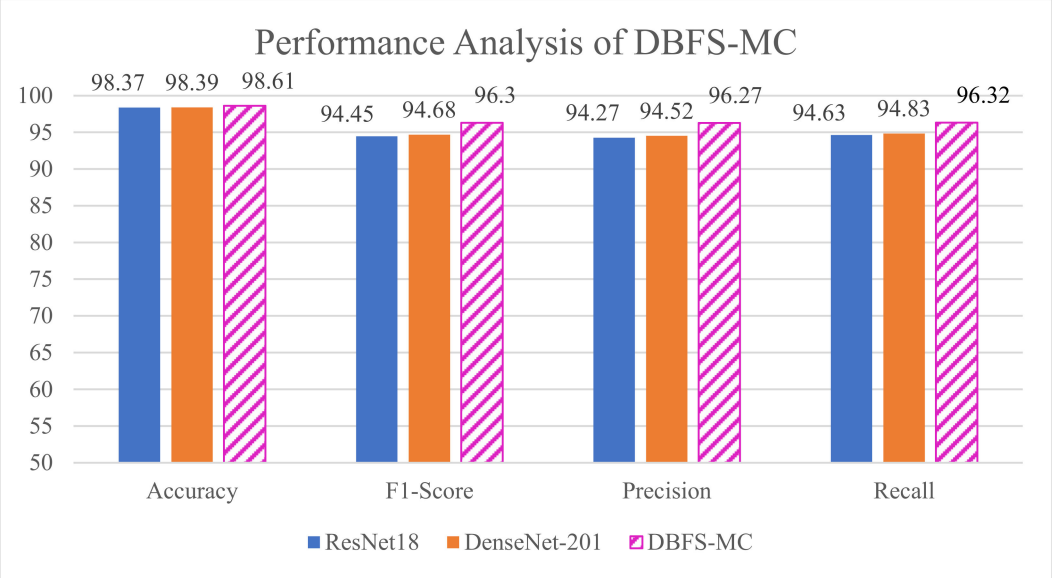

Figure 11. Performance improvement of the proposed DBFS-MC.

5.3. Comparative Analysis with the Reported Work

Most of the reported research has been performed using dataset partitioning of 70–30 or 80–20 in terms of accuracy. Therefore, in Table 9, the proposed framework is compared with the reported malware classification models in terms of accuracy. The standard measure for an imbalanced dataset is precision, recall, and F-score [34]. However, Most of the previous work has not reported the F1-score, but Cui et al. [35] showed their performance by quoting precision and recall.

Table 9. Comparative analysis of proposed DBFS-MC with the reported work.

Technique	%Accuracy	F-Score	Precision	Recall
Natraj et al. [14]	98.08	-	-	-
Cui et al. [35]	94.50	-	0.9460	0.9450
LGMP-2018 (encoder based) [36]	90.23	-	-	-
LGMP-2018 (cluster based) [36]	89.58	-	-	-
NSGA-II [37]	97.60	-	-	0.8840
VGG, end-to-end [38]	90.77	-	-	-
VGG, SVM [38]	92.29	-	-	-
S. Lad et al. (CNN + SVM) [39]	98.03	-	-	-
Proposed DBFS-MC	98.61	0.9632	0.9627	0.9630

6. Conclusions

Static anti-malware products can only identify already registered malware and are incapable of recognizing modified or newly-released malware. Dynamic malware classification examines malicious activities and footprints during execution in a controlled environment but may be resource and time intensive. Deep learning-based techniques can identify complex attack patterns in malware images and, thus, recognize new malware in a confined environment. In this regard, two new frameworks, DFS-MC and DBFS-MC, have been proposed for hybrid learning and feature boosting to identify malware using a large stringent unseen dataset. In the DBFS-MC framework, deep features are generated from the TL-based customized ResNet-18 and DenseNet-201. These deep feature spaces are concatenated for achieving a single boosted feature space and provided to an SVM. The feature boosting merged the potentials of customized ResNet-18 and DenseNet-201 models while leveraging the discrepancies of the single model. Furthermore, it efficiently classified the exceptional malware classes, i.e., Autorun.K, Swizzor.gen!I, Wintrim.BX and Yuner.A, which were previously not addressed. In addition, the 40:60 partitioning scheme of the dataset further increases the robustness. The DBFS-MC framework achieved the highest accuracy (98.61%) and F1-score (0.9630) and improved precision (0.9632) and recall (0.9627). Moreover, our findings revealed that the proposed DBFS-MC framework outperforms TL-based and training from scratch customized CNNs. In the future, the proposed frameworks may be adapted to new malware attacks using the standard benchmark dataset like android and IoT malware. Furthermore, this study may be enhanced by developing an anti-malware application for Microsoft Windows OS that can analyse the traffic on FTP for malware analysis in real-time scenarios.

Author Contributions: Conceptualization, M.A., S.H.K. and T.J.; Formal analysis, M.A. and S.J.H.; Funding acquisition, S.J.H. and M.U.A.; Investigation, A.K.; Methodology, A.K. and U.Z.; Project administration, M.A., M.M. and M.U.A.; Resources, M.A.; Supervision, A.Z.; Validation, M.A. and U.Z.; Visualization, T.J.; Writing—original draft, M.A., S.H.K. and T.J.; Writing—review & editing, S.J.H., M.M., A.Z. and M.U.A. All authors have read and agreed to the published version of the manuscript.

Funding: We acknowledge the Global college of Engineering and Technology, Muscat for funding this research under an internal research funding grant.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All the scripts that are developed for the simulations are available from the corresponding author on reasonable request.

Acknowledgments: This work was conducted with the support of the PAEC program. We also thank the Pattern Recognition Lab (PR-Lab) and Pakistan Institute of Engineering, and Applied Sciences (PIEAS), for providing necessary computational resources and a healthy research environment.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. AV-Test, "AV-TEST Report". Available online: https://www.av-test.org/en/statistics/malware/ (accessed on 15 June 2021).
2. Sihwail, R.; Omar, K.; Ariffin, K.A.Z. A Survey on Malware Analysis Techniques: Static, Dynamic, Hybrid and Memory Analysis. *Int. J. Adv. Sci. Eng. Inf. Technol.* **2018**, *8*, 1662–1671. [CrossRef]
3. Damodaran, A.; Di Troia, F.; Visaggio, C.A.; Austin, T.; Stamp, M. A comparison of static, dynamic, and hybrid analysis for malware detection. *J. Comput. Virol. Hacking Tech.* **2015**, *13*, 1–12. [CrossRef]
4. Souri, A.; Hosseini, R. A state-of-the-art survey of malware detection approaches using data mining techniques. *Human-Cent. Comput. Inf. Sci.* **2018**, *8*, 3. [CrossRef]
5. Preda, M.D. Code Obfuscation and Malware Detection by Abstract Interpretation. Available online: https://www.di.univr.it/documenti/AllegatiOA/allegatooa_03534.pdf (accessed on 10 November 2020).
6. You, I.; Yim, K. Malware Obfuscation Techniques: A Brief Survey. In Proceedings of the 2010 International Conference on Broadband, Wireless Computing, Communication and Applications, Fukuoka, Japan, 4–6 November 2010; pp. 297–300. [CrossRef]

7. Bazrafshan, Z.; Hashemi, H.; Fard, S.M.H.; Hamzeh, A. A survey on heuristic malware detection techniques. In Proceedings of the 5th Conference on Information and Knowledge Technology, Shiraz, Iran, 28–30 May 2013; pp. 113–120. [CrossRef]
8. Asad, M.; Asim, M.; Javed, T.; Beg, M.O.; Mujtaba, H.; Abbas, S. DeepDetect: Detection of Distributed Denial of Service Attacks Using Deep Learning. *Comput. J.* **2019**, *63*, 983–994. [CrossRef]
9. Gandotra, E.; Bansal, D.; Sofat, S. Malware Analysis and Classification: A Survey. *J. Inf. Secur.* **2014**, *5*, 56–64. [CrossRef]
10. Gibert, D.; Mateu, C.; Planes, J. The rise of machine learning for detection and classification of malware: Research developments, trends and challenges. *J. Netw. Comput. Appl.* **2020**, *153*, 102526. [CrossRef]
11. Khan, A.; Sohail, A.; Zahoora, U.; Qureshi, A.S. A survey of the recent architectures of deep convolutional neural networks. *Artif. Intell. Rev.* **2020**, *53*, 5455–5516. [CrossRef]
12. Ucci, D.; Aniello, L.; Baldoni, R. Survey of machine learning techniques for malware analysis. *Comput. Secur.* **2018**, *81*, 123–147. [CrossRef]
13. Rafique, M.F.; Ali, M.; Qureshi, A.S.; Khan, A.; Kim, J.Y.; Mirza, A.M. Malware classification using deep learning based feature extraction and wrapper based feature selection technique. *arXiv* **2019**, arXiv:1910.10958.
14. Nataraj, L.; Karthikeyan, S.; Jacob, G.; Manjunath, B.S. Malware images. *ACM Int. Conf. Proc. Ser.* **2011**. [CrossRef]
15. Makandar, A.; Patrot, A. Malware Image Analysis and Classification using Support Vector Machine. *Int. J. Adv. Trends Comput. Sci. Eng.* **2015**, *4*, 1–3.
16. Su, J.; Vasconcellos, V.D.; Prasad, S.; Daniele, S.; Feng, Y.; Sakurai, K. Lightweight Classification of IoT Malware Based on Image Recognition. In Proceedings of the 8th IEEE International Workshop on Network Technologies for Security, Administration, and Protection (NETSAP 2018), Tokyo, Japan, 23–27 July 2018. [CrossRef]
17. Karbab, E.B.; Debbabi, M.; Derhab, A.; Mouheb, D. MalDozer: Automatic framework for android malware detection using deep learning. *Digit. Investig.* **2018**, *24*, S48–S59. [CrossRef]
18. Chen, L.; Sultana, S.; Sahita, R. HeNet: A Deep Learning Approach on Intel® Processor Trace for Effective Exploit Detection. In Proceedings of the 2018 IEEE Security and Privacy Workshops (SPW), San Francisco, CA, USA, 24–24 May 2018. [CrossRef]
19. Ni, S.; Qian, Q.; Zhang, R. Malware identification using visualization images and deep learning. *Comput. Secur.* **2018**, *77*, 871–885. [CrossRef]
20. Kim, J.-Y.; Bu, S.-J.; Cho, S.-B. Zero-day malware detection using transferred generative adversarial networks based on deep autoencoders. *Inf. Sci.* **2018**, *460–461*, 83–102. [CrossRef]
21. Le, Q.; Boydell, O.; Mac Namee, B.; Scanlon, M. Deep learning at the shallow end: Malware classification for non-domain experts. *Digit. Investig.* **2018**, *26*, S118–S126. [CrossRef]
22. Shorten, C.; Khoshgoftaar, T.M. A survey on Image Data Augmentation for Deep Learning. *J. Big Data* **2019**, *6*, 60. [CrossRef]
23. Wang, J.; Perez, L. The Effectiveness of Data Augmentation in Image Classification using Deep Learning. *arXiv* **2017**, arXiv:1712.04621.
24. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv* **2014**, arXiv:1409.1556.
25. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 1–9. [CrossRef]
26. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the Inception Architecture for Computer Vision. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Francisco, CA, USA, 18–20 June 1996; pp. 2818–2826. [CrossRef]
27. Khan, S.H.; Sohail, A.; Khan, A.; Lee, Y.S. Classification and Region Analysis of COVID-19 Infection Using Lung CT Images and Deep Convolutional Neural Networks. September 2020. Available online: http://arxiv.org/abs/2009.08864 (accessed on 20 June 2021).
28. Khan, S.H.; Sohail, A.; Khan, A. COVID-19 Detection in Chest X-ray Images using a New Channel Boosted CNN. 2020. Available online: http://arxiv.org/abs/2012.05073 (accessed on 20 July 2021).
29. Khan, S.H.; Sohail, A.; Khan, A.; Hassan, M.; Lee, Y.S.; Alam, J.; Basit, A.; Zubair, S. COVID-19 detection in chest X-ray images using deep boosted hybrid learning. *Comput. Biol. Med.* **2021**, *137*, 104816. [CrossRef]
30. Faris, H.; Hassonah, M.A.; Al-Zoubi, A.M.; Mirjalili, S.; Aljarah, I. A multi-verse optimizer approach for feature selection and optimizing SVM parameters based on a robust system architecture. *Neural Comput. Appl.* **2017**, *30*, 2355–2369. [CrossRef]
31. Khan, S.H.; Yousaf, M.H.; Murtaza, F.; Velastin, S. Passenger detection and counting for public transport system. *NED Univ. J. Res.* **2020**, *2*, 35–46. [CrossRef]
32. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778. [CrossRef]
33. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely Connected Convolutional Networks. In Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, 21–26 July 2017; pp. 2261–2269. [CrossRef]
34. How Can the F1-Score Help with Dealing with Class Imbalance? Available online: https://sebastianraschka.com/faq/docs/computing-the-f1-score.html (accessed on 21 June 2021).
35. Cui, Z.; Xue, F.; Cai, X.; Cao, Y.; Wang, G.-G.; Chen, J. Detection of Malicious Code Variants Based on Deep Learning. *IEEE Trans. Ind. Inform.* **2018**, *14*, 3187–3196. [CrossRef]

36. Naeem, H.; Naeem, M.R. Visual Malware Classification Using Local and Global Malicious Pattern. *J. Comput.* **2020**, *30*, 73–83. [CrossRef]
37. Cui, Z.; Du, L.; Wang, P.; Cai, X.; Zhang, W. Malicious code detection based on CNNs and multi-objective algorithm. *J. Parallel Distrib. Comput.* **2019**, *129*, 50–58. [CrossRef]
38. Rezende, E.; Ruppert, G.; Carvalho, T.; Theophilo, A.; Ramos, F.; de Geus, P. Malicious Software Classification Using VGG16 Deep Neural Network's Bottleneck Features BT—Information Technology—New Generations. 2018, pp. 51–59. Available online: https://w3.lasca.ic.unicamp.br/media/publications/2018-ITNG-edmar.rezende-MaliciousClassifVGG16.DeepNeural.BottleneckFeatures.pdf (accessed on 30 October 2020).
39. Lad, S.S.; Adamuthe, A.C. Malware Classification with Improved Convolutional Neural Network Model. *Int. J. Comput. Netw. Inf. Secur.* **2020**, *12*, 30–43. [CrossRef]