Journal of Parallel and Distributed Computing 172 (2023) 32–50

Contents lists available at ScienceDirect

Journal of Parallel and Distributed Computing

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

Malware detection using image representation of malware data and transfer learning

Furqan Rustam [a], Imran Ashraf [b,*], Anca Delia Jurcut [c,*], Ali Kashif Bashir [d,e], Yousaf Bin Zikria [b]

[a] School of Computer Science, University College Dublin, Dublin, D04 V1W8, Ireland
[b] Information and Communication Engineering, Yeungnam University, Gyeongsan 38541, Republic of Korea
[c] School of Computer Science, University College Dublin, Ireland
[d] Department of Computing and Mathematics, Manchester Metropolitan University, Manchester, United Kingdom
[e] Woxsen School of Business, Woxsen University, Hyderabad, India

ARTICLE INFO

Article history:
Received 26 April 2022
Received in revised form 11 August 2022
Accepted 6 October 2022
Available online 13 October 2022

Keywords:
Malware detection
Transfer learning
Stacked model
Hybrid features

ABSTRACT

With the increased proliferation of internet-enabled mobile devices and large internet use, cybercrime incidents have grown exponentially, often leading to huge financial losses. Most cybercrimes are launched through malware attacks, phishing attacks, denial/distributed denial of attacks, looting people's money, stealing credential information for unauthorized use, personal identity thefts, etc. Timely detection of malware can avoid such damage. However, it requires an efficient and effective approach to detecting such attacks. This study attempts to devise a malware detection approach using transfer learning and machine learning algorithms. A hybrid approach has been adopted where pre-trained models VVG-16 and ResNet-50 extract hybrid feature sets from the data to be used with the machine learning algorithms. In doing so, this study contrives the Bi-model architecture where the same models are combined sequentially in the stacked form to obtain higher performance as the output of the first model is used to train the second model. With the Bi-model structure, 100% accuracy is obtained for a 25 classes problem. Performance comparison with state-of-the-art models and T-test proves the superior performance of the proposed approach.

1. Introduction

The proliferation of many internet-enabled mobile devices, the majority of which are smartphones, has increased the number of internet interconnections. According to [1], approximately 6.648 billion use smartphone devices which makes 83.96% of the total population of the world. Additionally, the accelerated digital transformation and dependence on internet-connected infrastructure have made these devices open to cyberattacks. Furthermore, the ease of use and 24/7 access to online applications to perform everyday tasks like education, business, banking, social communication, surveillance, medical, etc., require continuous use of the internet and are highly vulnerable to cyberattacks. Cyberattacks are launched in malware, phishing, internet of things (IoT), and distributed denial of service (DDoS) attacks to steal personal and financial information for financial and other benefits. According to [10,19], a trillion dollars loss has been made to the world's economy as a result of cybercrimes. With increased digital transformation and wide use of internet-based services, acceleration has been observed in the rate of cybercrimes. According to the latest report, cybercrime costs 20 billion dollars in 2021, which is 57 times higher than in 2015, and it is expected to grow to 10.5 trillion by 2025 [2].

Malware is a widely used approach for cybercrimes, and it can be in the form of trojan, rootkit, adware, spyware, ransomware, etc. [25]. Malware is a software/piece of code used to access unauthorized systems to record key logs, screens, and other sensitive information with the intent to steal such information or damage or disrupt the normal functioning of the system [10]. Malware can be categorized into different types such as spyware, virus, adware, ransomware, worms, trojan virus, file-less malware, etc. [31]. In addition, new variants emerge over time. For example, recent malware remains invisible during the transaction and affects the user system silently by stealing credentials [21]. Malware attacks have increased largely during recent years and necessitate continuous research efforts to reduce the threat of such attacks. Recently, machine learning-based models and tools have shown superiority for

* Corresponding authors.
E-mail addresses: imranashraf@ynu.ac.kr (I. Ashraf), anca.jurcut@ucd.ie (A.D. Jurcut).

https://doi.org/10.1016/j.jpdc.2022.10.001
0743-7315/

detecting such attacks. Trained on large datasets of malware attacks and periodic updating can provide a higher level of security against these attacks.

In describing the importance of machine learning models for malware attack detection, we advocate the efficacy of such models against malware and similar attacks to steal personal and financial credentials. Despite the availability of many existing sophisticated models, with emerging attacks, the security provided by these models is not enough, and new techniques need to be contrived. In this regard, this study endeavors a malware detection approach for 25 types of malware and makes the following contributions.

- A novel approach is presented that leverages transfer learning for feature extraction to train machine learning models. For this purpose, features from ResNet-50 and VVG-16 are combined to train models.
- The concept of the Bi-model is used where the same model is stacked. The conceding model is trained on the combined features, while the preceding model is trained on the output generated from the conceding model, thus helping to refine the prediction performance.
- Extensive experiments are performed to evaluate the performance of the approach from several perspectives using standalone models, pre-trained models, individual features from VVG16 and ResNet-50, and pre-trained models InceptionV3 and EfficientNetB0. Experiments are also carried out for retrained models and state-of-the-art approaches for a fair comparison.

The rest of the paper is divided into the following sections. Section 2 contains the literature review with a discussion of several important works. Section 3 provides the description and foundation of the proposed approach. Results are presented in Section 4, followed by the concluding remarks in Section 5.

2. Related work

Several recent works can be found in the literature that focuses on malware detection using machine learning, and deep learning-based approaches [6,11,13,14,33] (see Table 1 for a comparative summary of existing literature on malware detection). For example, a recent survey on cyber security threats in the maritime industry can be found in [9]. The study outlines the recent threats to the industry in the context of the IoT-enabled industry. The study [13] proposed an approach for malware detection using the convolutional neural network (CNN) variant ResNeXt. The study deployed the approach on the Malimg dataset, where the proposed method transfers the attribute of raw Malimg into grayscale images. ResNeXt achieved a significant accuracy of 98.32% and 98.86% on the Malimg and modified Malimg datasets. Along with the same directions, [33] proposed an approach for malware classification using colored malware images. A CNN model is developed and finetuned to perform better using colored malware images. Experiments are performed on the Malimg malware dataset with 9435 samples and the IoT-Android mobile (IoT-AM) dataset with 7219 samples to make the binary classification.

Similarly, the study [11] proposed an ensemble approach for malware classification using several neural networks and a support vector machine (SVM). The authors deployed CNN-SVM, gated recurrent unit (GRU)-SVM, and multilayered perceptron (MLP)-SVM for experiments on the Malimg dataset. Results show an accuracy of 98.52% using the proposed approach. The study [23] used the features fusion where the extracted features from the AlexNet and Inception-v3 are used for segmentation-based fractal texture analysis (SFTA). The extracted features are fed into SVM, K nearest neighbor (KNN), and decision tree (DT) for the final prediction. Experiments with the Malimg dataset yield an accuracy of 0.99. A malware detection scheme for industrial IoT is presented in [32], which follows the architecture of a classified behavior. The study uses a graph-based model for malware detection and considers Delf, Obfuscated, Small, and Zlob classes for experiments. The proposed approach obtains high accuracy of up to 99% of the true prediction rate while avoiding the high computational cost of graph matching.

Malware detection performance is often affected when the models are used with imbalanced data distributions for different classes. The authors propose a deep learning-based malware detection approach, called DenseNet in [16], with a reweighted class-balanced loss function in the last layer to improve the results on the imbalanced data. Experiments involving the Malimg, BIG, MaleVis, and Malicia datasets yield accuracy scores of 98.23%, 98.46%, 98.21%, and 89.48%, respectively. An ensemble model is proposed in [12] that uses random forest (RF) with CapsNet model. The approach uses capsule neural networks and bootstrap aggregation techniques to mitigate the influence of imbalanced data. Experiments performed on the Malimg and Microsoft BIG 2015 show the highest F1 score of 96.7% on the Malimg dataset. The impact of dataset imbalance can also be catered to by using the synthetic samples generated using resampling. For example, [20] used the data augmentation to make a balanced dataset and resolve the model overfitting problem for malware detection.

The study [29] proposed an approach for malware detection using the combination of machine learning and deep learning models. The study used CNN with gradient boosting (XG-Boost) for classification, where the CNN-based features are passed to train the XG-Boost model for prediction. The proposed approach CNN-XGBoost shows an accuracy score of 98.7% on the Maligm dataset. Similarly, a hybrid model for malware detection is presented in [10] that combines multiple deep learning models. Two deep learning models AlexNet and ResNet-152 extract the feature vector from the dataset and then use a fully connected neural network to classify the malware variants. The proposed approach was tested on the two publicly available datasets: the Malimg and Microsoft BIG 2015 dataset, with the highest accuracy of 97.78% for the Malimg dataset.

The study deployed a transfer learning approach with CNN and the Bi-directional long short-term memory (Bi-LSTM) model. Experiments are carried out on the Malimg and OBF datasets, with the proposed approach achieving a 98.5% accuracy score. The study [18] proposed deep transfer learning for malware image classification (DTMIC) approach for malware classification. For feature extraction, the authors used several CNN variants such as VVG16, VVG19, ResNet50, and Google's inceptionV3 models. DTMIC approach achieved an accuracy of 98.92% on the Malimg dataset and 93.19% on the Microsoft dataset. The study [7] also proposed a malware image classification using the CNN model and automated transmutation. They deployed state-of-the-art CNN architecture for malware prediction and the proposed approach achieved a significant precision of 98.97%.

3. Material and methods

This research work proposes an approach for malware detection using transfer learning with Bi-models. Extensive experiments are performed to evaluate the performance of the stacked model.

3.1. Proposed approach

The architecture of the proposed approach is illustrated in Fig. 1. As the first step, a publicly available dataset is obtained, which in this case is the Malimg dataset [22]. The dataset contains

Table 1
A comparative summary of existing literature on malware detection.

Ref	Year	Approach	Dataset	Method	Significance
[13]	2020	DL	Malimg & modified Malimg	ResNeXt	Simple CNN using grayscale images, high accuracy
[33]	2020	DL	Malimg, IoT-AM	Fine-tuned CNN	Colored malware images with simple CNN, high accuracy
[11]	2020	DL/ML	Malimg	CNN-SVM, GRU-SVM, MLP-SVM	Ensemble classification models, high accuracy
[23]	2020	DL/ML	Malimg	AlexNet, InceptionV3, SFTA with SVM, KNN and DT	Ensemble of DL & ML
[16]	2021	DL	Malimg, BIG, MaleVis and Malicia	DenseNet with reweighted class-balanced loss function	High accuracy with imbalanced datasets
[29]	2021	DL/ML	Malimg	CNN with XG-Boost	Hybrid approach of ML & DL
[10]	2021	DL	Malimg, BIG	AlexNet and ResNet-152	Hybrid feature vector extraction
[12]	2021	DL/ ML	Malimg, BIG	Capsule NN and bootstrap aggregation.	Ensemble approach, high accuracy with imbalanced datasets
[20]	2021	DL	Malimg, OBF	Transfer learning with CNN and Bi-LSTM	Augmentation and transfer learning
[18]	2022	DL	Malimg, BIG	Transfer learning with VVG16, VVG19, ResNet50, and InceptionV3	An approach with high accuracy for malware detection
[7]	2022	DL	Malimg	Transfer learning with CNN	An approach with high accuracy for malware detection

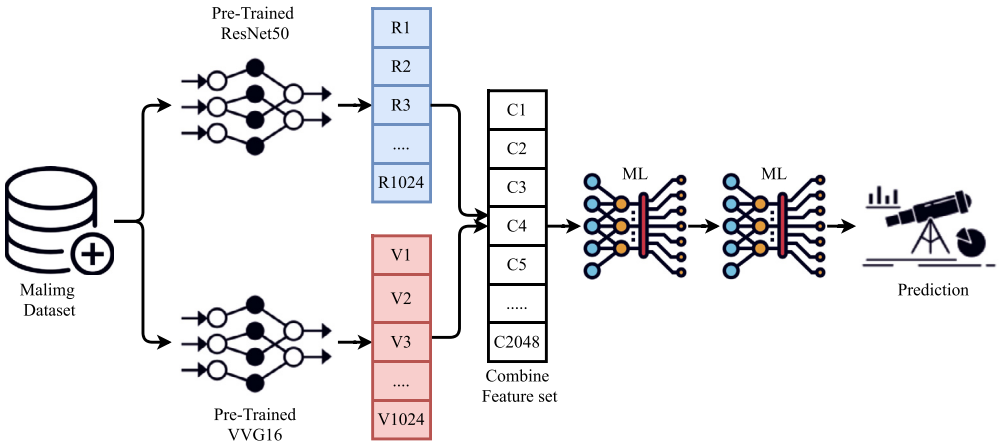

Fig. 1. Architecture of the proposed approach.

grayscale images passed to two pre-trained CNN variants VVG16 and ResNet50. Both models take the Malimg dataset separately as input to extract the features. Each of these pre-trained models generates a feature set size of 1024. These feature sets are combined to make a large feature set for further processing to obtain better training for the Bi-models. In the bi-model architecture, Model-1 takes the hybrid feature set as input and provides the prediction probabilities of each class. The Model-2 is then trained on the predicted probabilities generated by the Model-1, which helps in enhancing the classification. Here, Bi-models are used, indicating that both Model-1 and Model-2 are the same models like support vector classifier (SVC), random forest (Rf) and logistic regression (LR), etc. Experiments are performed with 0.8 to 0.2 train-test split ratios.

Table 2
Details of the target class, number of samples, and encoded labels.

Target	Label	Samples	Target	Label	Samples
Adialer.C	0	125	Lolyda.AA2	13	184
Agent.FYI	1	116	Lolyda.AA3	14	123
Allaple.A	2	2949	Lolyda.AT	15	159
Allaple.L	3	1591	Malex.gen!J	16	136
Allueron.gen!J	4	198	Obfuscator.AD	17	142
Autorun.K	5	106	Rbot!gen	18	158
C2Lop.P	6	146	Skintrim.N	19	80
C2Lop.gen!g	7	200	Swizzor.gen!E	20	128
Dialplatform.	8	177	Swizzor.gen!I	21	132
Dontovo.A	9	162	VB.AT	22	408
Fakerean	10	381	Wintrim.BX	23	97
Instantaccess	11	431	Yuner.A	24	800
Lolyda.AA1	12	213			

3.2. Maligm dataset

This study used a publicly available dataset named Malimg, which contains images for 25 malware variants. This dataset is generated using malware binaries as 1D malware binary is transformed into image features [22].

Table 2 shows the variants of malware in the Malimg dataset and the count of images for each malware variant. The dataset contains 25 classes, each with a different number of samples. The encoded labels range from 0 to 24. Each target class represents a variant of the recorded malware attacks. The dataset has been selected for its public availability. It uses in several other important works, and it is easy to compare the performance of the current approach with other state-of-the-art approaches. Fig. 2 shows the sample images of malware variants from the Maligm dataset.

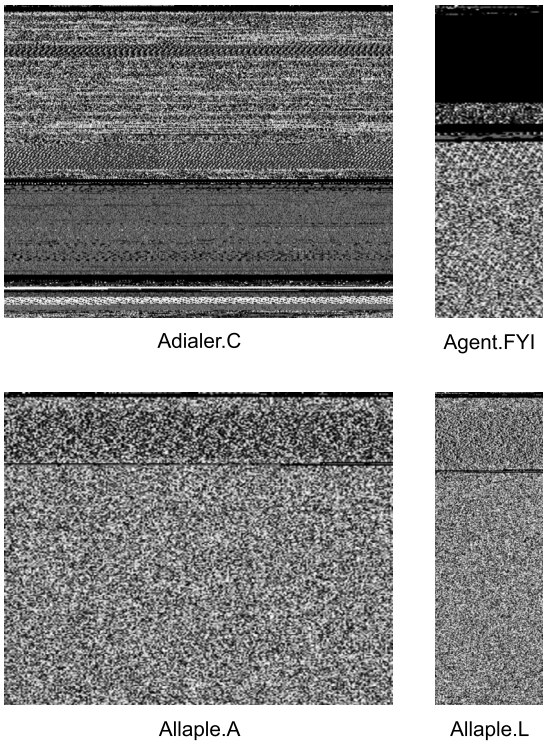

Fig. 2. Malimg dataset malware variants sample visualization.

3.3. Transfer learning

Transfer learning (TL) is a machine learning technique in which a pre-trained model is finetuned and used for a different task than the one for which it has been trained. This study uses two pre-trained models VVG-16 and ResNet-50 for feature extraction which are then passed to selected machine learning models for training. Other than that, such models can also extract important features to train models.

3.3.1. VVG16

VVG16 is a variant of the CNN model proposed by the Oxford University large-scale image recognition [30]. The VVG16 is trained over 14 million images from 1000 categories. It is an improved model compared to AlexNet and contains large kernel-size filters. The first convolutional layer has 11 kernel size filters, while the second layer has 5 kernel size filters. Following the convolutional layer is a 3x3 size of kernels. VGG16 is trained over weeks using NVIDIA Titan Black graphical processing unit (GPU) [3]. The architecture of VVG16 is given in Fig. 3 indicating its structure of 13 convolutional layers, 4 max-pooling layers, and 3 dense layers.

3.3.2. ResNet-50

ResNet-50 or Residual net belongs to the ResNet family and is based on the CNN architecture. It goes deeper than compared to VVG16 to improve the accuracy. ResNet-50 tackles the vanishing Gradient issue, as well as achieves significant results than other models [15]. ResNet applies residual fitting on all layers because it is easier to fit the actual mapping. All layers are in the stack architecture so that the model can perform significantly better than other models [4]. The architecture of ResNet-50 is shown in Fig. 4.

3.4. Supervised machine learning models

Features extracted from VVG16 and ResNet-50 are used to train machine learning models. This study deploys four machine learning models, including SVC, LR, RF, and KNN, to optimize the results

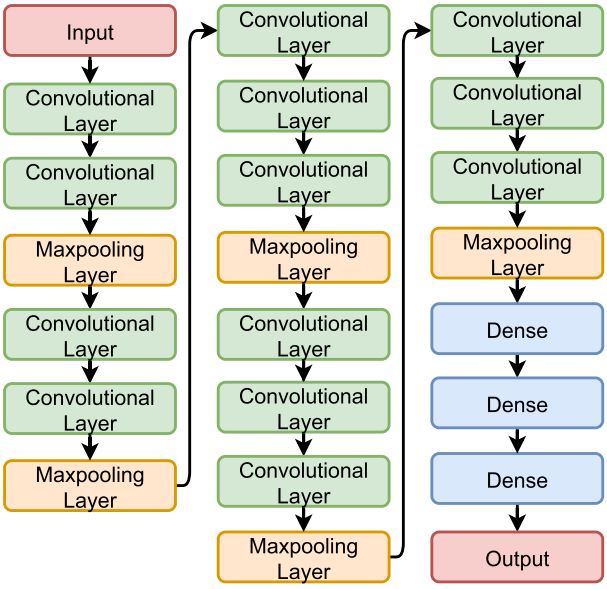

Fig. 3. Architecture of VVG16 [3,30].

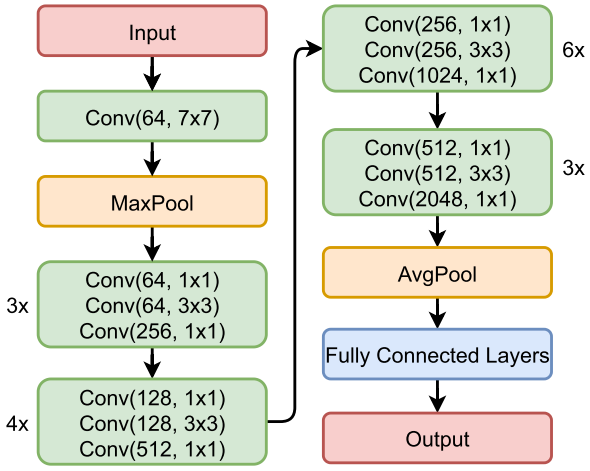

Fig. 4. Architecture of ResNet [15,17].

with settings of different hyperparameters. An appropriate range of these parameters is found from different high-accuracy reporting studies from the literature, and the range is used to select the best-performing parameter sets empirically. The range of parameters and the values with the best performance are listed in Table 3.

3.4.1. Support vector classifier

SVC is a linear model used for the classification and tends to perform better when the feature set for the training is large [24,26]. It is specifically selected for experiments as this study combines the features from VVG16 and ResNet-50 to increase the feature set. SVC draws multiple hyperplanes in the feature space to classify the data. The hyperplanes that classify the data with the best margin will be selected for classification. We used poly kernel with SVC, which gives better results when the training data is normalized [5]. The cost function used to determine the hyperplanes is given in equation (1)

$$J(\theta) = \frac{1}{2}\sum_{j=1}^{n}\theta_j^2 \tag{1}$$

Table 3
Hyperparameters used for optimizing the performance of models.

Model	Hyperparameters	Tuning range
SVC	kernel = 'poly', C=2.0	kernel= {'linear', 'poly', 'sigmoid'}, C={1.0 to 5.0}
RF	n_estimators = 300, max_depth = 300	n_estimators = {50 to 500}, max_depth = {50 to 500}
LR	solver = saga, C = 2.0	solver = {sag, saga, newton-cg} C = {1.0 to 5.0}
KNN	n_neighbors= 5	n_neighbors={1.0 to 10.0}

3.4.2. Random forest

RF is a tree-based model used for classification and regression tasks. RF combines many decision trees to formulate the final decision under majority voting criteria [27]. Each decision tree is trained on random samples from the input data, and each tree makes the prediction. RF performs voting between these predictions as the target class with more predictions will be the final class. We used 300 decision trees in this voting criteria with a max 300 level depth which helps to reduce the complexity and overfitting. Mathematically, RF is defined as

$$tree_1, tree_2, tree_3, ...tree_n \epsilon\, rf \tag{2}$$

Here, $tree_1, tree_2, tree_3$ are trees in RF while we can define RF prediction as:

$$prediction = mode\{tree_1, tree_2, tree_3\} \tag{3}$$

or,

$$prediction = mode\{\sum_{i=1}^{N} tree_n\} \tag{4}$$

3.4.3. Logistic regression

LR is a statistical model used for the classification of data when the feature set for training is large [28]. LR is used to describe data and explain the relationship between one dependent variable and one or more nominal, ordinal, interval, or ratio-level independent variables. LR uses the Sigmoid function to classify the data as shown in equation (5). We used LR with a multinomial variable which makes it suitable for multi-class data and binary class data.

$$sigmoid\ function = \frac{1}{1 + e^{-(\beta_o + \beta_i)}} \tag{5}$$

3.4.4. K nearest neighbor

KNN is a simple machine learning algorithm used for classification and regression tasks. Also known as the lazy learner [8], its computational cost is very low and architecture very simple. It matches new data with training data classes using a distance metric like the Euclidean distance (ED). It labels new data according to the training data label more closely. We used KNN with one hyper-parameter n_neighbors with value 7, defining the number of neighbors required for each sample.

3.4.5. Bi-models

For the proposed approach, we leverage Bi-models, where models are combined sequentially with themselves. In this approach, the first model generates prediction probabilities using the hybrid feature generated by the transfer learning approach, and the second model is trained on the first model's output probabilities to make the final prediction. Therefore, we use SVC to make Bi-SVC, RF to make Bi-RF, etc.

3.5. Mathematical definition for proposed approach

The proposed approach first uses the pre-trained VVG16 and ResNet-50 to extract the features from the Maligm dataset. The feature extraction can be defined as

$$X_i = Malimg\_Images \tag{6}$$

Where X_i are image features from 25 target classes from the Maligm dataset. Equation (7) shows the VVG16 extracted feature $VVG-16_f$ while equation (8) shows the ResNet-50 features. Both feature sets have the same size of 1024.

$$VVG16_f = \begin{pmatrix} V_1 & V_2 & V_3 & .. & V_{1024} \\ .. & .. & .. & .. & .. \\ .. & .. & .. & .. & .. \\ .. & .. & .. & .. & .. \end{pmatrix} = VVG16(X_i) \tag{7}$$

$$ResNet50_f = \begin{pmatrix} R_1 & R_2 & R_3 & .. & R_{1024} \\ .. & .. & .. & .. & .. \\ .. & .. & .. & .. & .. \\ .. & .. & .. & .. & .. \end{pmatrix} = ResNet50(X_i) \tag{8}$$

The extracted features from VVG16 and ResNet-50 are combined to make a hybrid feature (HF) set. After the combination, the feature set size is enlarged to 2048.

$$HF = \begin{pmatrix} V_1 & V_2 & V_3 & .. & V_{1024} \\ .. & .. & .. & .. & .. \\ .. & .. & .. & .. & .. \\ .. & .. & .. & .. & .. \end{pmatrix} . \begin{pmatrix} R_1 & R_2 & R_3 & .. & R_{1024} \\ .. & .. & .. & .. & .. \\ .. & .. & .. & .. & .. \\ .. & .. & .. & .. & .. \end{pmatrix} \tag{9}$$

or,

$$HF = \begin{pmatrix} H_1 & H_2 & H_3 & .. & H_{2048} \\ .. & .. & .. & .. & .. \\ .. & .. & .. & .. & .. \\ .. & .. & .. & .. & .. \end{pmatrix} \tag{10}$$

This hybrid feature set is then used with the Bi-model scheme, in which the first model will use a hybrid feature set to give probability features for the training of the second model.

$$Prob = \begin{pmatrix} P_1 & P_2 & .. & P_{25} \\ .. & .. & .. & .. \\ .. & .. & .. & .. \\ .. & .. & .. & .. \end{pmatrix} = model_1(HF) \tag{11}$$

Here, in Fig. 5, and equations (10) and (11), H_1, H_2,..., H_{2048} show the features generated using hybrid feature approach which are 2048 in total while P_1, P_2,..., P_{25} are probabilities generated by the 1st model in Bi-Model approach which are 25 in total. To get this small probability-based feature set we do not drop the features from the feature set, rather we transfer a large image feature set into a small feature set using the pre-trained models. It shows that a large set of information is converted into small and correlated information. This small feature set has the same information

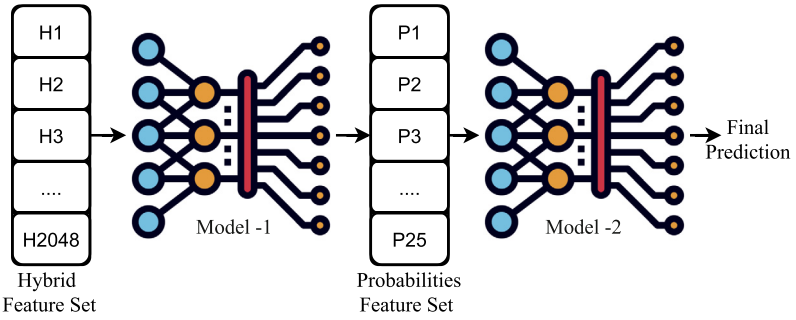

Fig. 5. Architecture of Bi-model approach.

as the original feature set but with a more significant and correlated feature set with respect to the target class. The steps carried out for transforming the feature set are given in Algorithm 1.

$$final\_prediction = model_2(Prob) \tag{12}$$

Algorithm 1 Proposed approach.

Input: Malware Images
Output: Malware Prediction
initialization;
imgs ⟵ Load(Malimg)
VVG_F ⟵ Pre-Trained_VVG-16 (imgs)
$ResNet_F$ ⟵ Pre-Trained_ResNet-50 (imgs)
HF ⟵ Concatenate (VVG_F, $ResNet_F$)
$prob$ ⟵ Model_1(HF)
Malware_Prediction ⟵ Model_1($prob$)

4. Results and discussion

The proposed approach is deployed on an Intel Core i7 11th generation machine with Windows operating system. The machine consists of 16 GB RAM and 2 GB NVIDIA GPU. We implement the proposed methodology on the Jupyter notebook in Python with several libraries such as Tensorflow, Keras, and Sci-kit learn. We evaluate the performance of models in terms of accuracy, precision, recall, and F1 score. As the used dataset is imbalanced, the models can be overfitted for majority class data, and thus, in that case, we prefer an F1 score over other evaluation parameters.

4.1. Results using single models with combined features

Initial experiments are performed using single models trained on the features extracted from VVG16 and ResNet-50 models and combined to enlarge the feature set. Table 4 shows the results of SVC using the hybrid feature set extracted by pre-trained models. The results of SVC show significant performance with the transfer learning approach as it achieves a 0.99 accuracy score and 0.97 F1 score. SVC has a significantly higher accuracy of 100% on all variants predictions except for 'C2Lop.P', 'C2Lop.gen!g', 'Swizzor.gen!E' and 'Swizzor.gen!I' classes. It gives 3 wrong predictions for 'C2Lop.P', 1 wrong prediction for 'C2Lop.gen!g', 8 for 'Swizzor.gen!E', and 8 for 'Swizzor.gen!I' variants because these classes have fewer samples for training as compared to other classes. While 'Swizzor.gen!E' and 'Swizzor.gen!I' both have similar features, it is difficult for the model to distinguish between them correctly. Overall the performance of SVC with transfer learning is significantly better, with a 0.97 F1 score which can be attributed to a feature set generated by the transfer learning approach.

Table 5 shows the results of RF using the transfer learning features. RF has a significantly lower performance than SVC as it achieves a 0.94 F1 score with a 0.98 accuracy score. It performs poorly in the higher number of classes compared to SVC. It gives more wrong predictions like 16 and 18 for 'Swizzor.gen!E' and 'Swizzor.gen!I', respectively. These results show that the ruled-based tree models do not perform well on the Maligm dataset with linear features set generated by the transfer learning approach. In comparison, the results using the KNN are better with features from VVg16 and ResNet-50 than the RF model, as shown in Table 6. It obtains a 0.95 F1 Score with a 0.98 accuracy score slightly better than RF. Despite being simple, KNN shows better results than RF. Although it has a higher number of miss-classifications like 110 for 'Swizzor.gen!E' and 18 for the 'Swizzor.gen!I' classes, these wrong predictions are lower than RF's.

The performance of the LR model is better than both KNN and RF, as indicated in Table 7. Significantly better with a 0.96 F1 score and a 0.98 accuracy score, it made fewer wrong classifications than both KNN and RF. The sparse feature set is suitable for linear models like LR, which results better. The wrong predictions from the LR belong to the same classes for which SVC made the wrong prediction, i.e., 10 and 13 for 'wizzor.gen!E' and 'Swizzor.gen!I' classes, respectively. Results show that linear models LR and SVC perform better than KNN and RF.

4.2. Results using bi-models with features from VVG16 and ResNet-50

To analyze the Bi-models' efficiency and superiority, additional experiments are performed using the Bi-models structures of SVC, RF, KNN, and LR. With this approach, Model-1 is trained using combined features from VVG16 and ResNet-50, while Model-2 is trained on the output of Model-1 in the form of class probabilities feature set.

Table 8 contains the results of Bi-SVC using transfer learning, indicating the substantial increase in the performance of SVC with a 100% accuracy. It happens due to two factors: initially, the features are combined, which helps in the better training of Model-1. For Model-2, the best fit is obtained when it is trained on the probability feature set of Model-1. The sequentially combined SVC uses these probabilities to make final predictions, and the results are significantly better than using the single SVC model. The shortcoming of wrong predictions for 'wizzor.gen!E' and 'Swizzor.gen!I' classes is no more present with the Bi-SVC model. Bi-SVC achieved a 1.00 F1 score with 1.00 accuracy.

The performance of Bi-RF and Bi-LR models has also been improved significantly, and both models obtain a 100% accuracy, as shown in Tables 9 and 11. Bi-KNN shows comparatively poor performance with a 0.96 F1 score and 0.99 accuracy score. The class-wise predictions are shown in Table 10 indicating 6 wrong predictions for 'Swizzor.gen!E' and 7 for 'Swizzor.gen!I'. Also, the wrong predictions for 'C2Lop.P' and 'C2Lop.gen!g' are 6 each, reducing its overall accuracy and F1 score compared to Bi-SVC, Bi-RF, and Bi-LR.

Fig. 6 shows the comparison between the Bi-Model Approach and the stand-alone approach in terms of accuracy.

Table 4
SVC results using hybrid features extracted using pre-trained VVG-16+ResNet-50.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	Prec.	Rec.	F1
0	**22**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
1	0	**28**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
2	0	0	**590**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
3	0	0	0	**315**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
4	0	0	0	0	**44**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
5	0	0	0	0	0	**26**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
6	0	0	0	0	0	0	**25**	2	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0.89	0.89	0.89
7	0	0	0	0	0	0	1	**34**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.92	0.97	0.94
8	0	0	0	0	0	0	0	0	**34**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
9	0	0	0	0	0	0	0	0	0	**24**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
10	0	0	0	0	0	0	0	0	0	0	**87**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
11	0	0	0	0	0	0	0	0	0	0	0	**98**	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
12	0	0	0	0	0	0	0	0	0	0	0	0	**47**	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
13	0	0	0	0	0	0	0	0	0	0	0	0	0	**27**	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**22**	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**27**	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**20**	0	0	0	0	0	0	0	0	1.00	1.00	1.00
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**29**	0	0	0	0	0	0	0	1.00	1.00	1.00
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**34**	0	0	0	0	0	0	1.00	1.00	1.00
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**17**	0	0	0	0	0	1.00	1.00	1.00
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**23**	8	0	0	0	0.74	0.74	0.74
21	0	0	0	0	0	0	2	1	0	0	0	0	0	0	0	0	0	0	0	0	8	**16**	0	0	0	0.64	0.59	0.62
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**83**	0	0	1.00	1.00	1.00
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**25**	0	1.00	1.00	1.00
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**148**	1.00	1.00	1.00
																							accuracy					0.99
																							macro avg			0.97	0.97	0.97

Table 5
RF results using hybrid features extracted using pre-trained VVG-16+ResNet-50.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	Prec.	Rec.	F1
0	**22**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
1	0	**28**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
2	0	0	**590**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.99	1.00	1.00
3	0	0	0	**315**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
4	0	0	0	0	**44**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
5	0	0	0	0	0	**26**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
6	0	0	0	0	0	0	**26**	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.87	0.93	0.90
7	0	0	0	0	0	0	1	**33**	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0.69	0.94	0.80
8	0	0	0	0	0	0	0	0	**33**	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1.00	0.97	0.99
9	0	0	0	0	0	0	0	0	0	**24**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
10	0	0	1	0	0	0	0	0	0	0	**86**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	0.99	0.99
11	0	0	0	0	0	0	0	0	0	0	0	**98**	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
12	0	0	0	0	0	0	0	0	0	0	0	0	**47**	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
13	0	0	0	0	0	0	0	0	0	0	0	0	0	**27**	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**22**	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**27**	0	0	0	0	0	0	0	0	0	0.96	1.00	0.98
16	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	**16**	0	0	0	0	0	0	0	0	1.00	0.80	0.89
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**29**	0	0	0	0	0	0	0	1.00	1.00	1.00
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**32**	0	0	1	1	0	0	1.00	0.94	0.97
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**17**	0	0	0	0	0	1.00	1.00	1.00
20	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	**15**	7	0	0	0	0.63	0.48	0.55
21	0	0	0	0	0	0	3	4	0	0	0	0	0	0	0	0	0	0	0	0	9	**9**	2	0	0	0.50	0.33	0.40
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**83**	0	0	0.97	1.00	0.98
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**25**	0	1.00	1.00	1.00
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**148**	1.00	1.00	1.00
																							accuracy					0.98
																							macro avg			0.94	0.94	0.94

Table 6
KNN results using hybrid features extracted using pre-trained VVG-16+ResNet-50.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	Prec.	Rec.	F1
0	**22**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
1	0	**28**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
2	0	0	**589**	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
3	0	0	0	**315**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
4	0	0	0	0	**44**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
5	0	0	0	0	0	**26**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
6	0	0	0	0	0	0	**25**	2	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0.86	0.89	0.88
7	0	0	0	0	0	0	1	**33**	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0.77	0.94	0.85
8	0	0	0	0	0	0	0	0	**33**	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1.00	0.97	0.99
9	0	0	0	0	0	0	0	0	0	**24**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
10	0	0	1	0	0	0	0	0	0	0	**86**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	0.99	0.99
11	0	0	0	0	0	0	0	0	0	0	0	**98**	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
12	0	0	0	0	0	0	0	0	0	0	0	0	**47**	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
13	0	0	0	0	0	0	0	0	0	0	0	0	0	**27**	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**22**	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**27**	0	0	0	0	0	0	0	0	0	0.96	1.00	0.98
16	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	**19**	0	0	0	0	0	0	0	0	1.00	0.95	0.97
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**29**	0	0	0	0	0	0	0	1.00	1.00	1.00
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**34**	0	0	0	0	0	0	0.94	1.00	0.97
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**17**	0	0	0	0	0	1.00	1.00	1.00
20	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	**21**	5	0	0	0	0.64	0.68	0.66
21	0	0	0	0	0	0	2	3	0	0	0	0	0	0	0	0	0	0	2	0	11	**9**	0	0	0	0.60	0.33	0.43
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**83**	0	0	1.00	1.00	1.00
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**25**	0	1.00	1.00	1.00
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**148**	1.00	1.00	1.00
																							accuracy					0.98
																							macro avg			0.95	0.95	0.95

Table 7
LR results using hybrid features extracted using pre-trained VVG-16+ResNet-50.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	Prec.	Rec.	F1
0	**22**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
1	0	**28**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
2	0	0	**590**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
3	0	0	0	**315**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
4	0	0	0	0	**44**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
5	0	0	0	0	0	**26**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
6	0	0	0	0	0	0	**23**	2	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0.92	0.82	0.87
7	0	0	0	0	0	0	1	**33**	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0.92	0.94	0.93
8	0	0	0	0	0	0	0	0	**34**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
9	0	0	0	0	0	0	0	0	0	**24**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
10	0	0	0	0	0	0	0	0	0	0	**87**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
11	0	0	0	0	0	0	0	0	0	0	0	**98**	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
12	0	0	0	0	0	0	0	0	0	0	0	0	**47**	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
13	0	0	0	0	0	0	0	0	0	0	0	0	0	**27**	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**22**	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**27**	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**20**	0	0	0	0	0	0	0	0	1.00	1.00	1.00
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**29**	0	0	0	0	0	0	0	1.00	1.00	1.00
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**33**	0	0	1	0	0	0	0.97	0.97	0.97
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**17**	0	0	0	0	0	1.00	1.00	1.00
20	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	**21**	9	0	0	0	0.64	0.68	0.66
21	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	11	**14**	0	0	0	0.52	0.52	0.52
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**83**	0	0	1.00	1.00	1.00
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**25**	0	1.00	1.00	1.00
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**148**	1.00	1.00	1.00
																							accuracy					0.98
																							macro avg			0.96	0.96	0.96

Table 8
Bi-SVC results using hybrid features extracted using pre-trained VVG-16+ResNet-50.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	Prec.	Rec.	F1
0	**25**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
1	0	**25**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
2	0	0	**590**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
3	0	0	0	**298**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
4	0	0	0	0	**54**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
5	0	0	0	0	0	**23**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
6	0	0	0	0	0	0	**23**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
7	0	0	0	0	0	0	0	**43**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
8	0	0	0	0	0	0	0	0	**35**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
9	0	0	0	0	0	0	0	0	0	**34**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
10	0	0	0	0	0	0	0	0	0	0	**77**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
11	0	0	0	0	0	0	0	0	0	0	0	**88**	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
12	0	0	0	0	0	0	0	0	0	0	0	0	**42**	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
13	0	0	0	0	0	0	0	0	0	0	0	0	0	**33**	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**27**	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**41**	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**27**	0	0	0	0	0	0	0	0	1.00	1.00	1.00
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**27**	0	0	0	0	0	0	0	1.00	1.00	1.00
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**27**	0	0	0	0	0	0	1.00	1.00	1.00
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**20**	0	0	0	0	0	1.00	1.00	1.00
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**22**	0	0	0	0	1.00	1.00	1.00
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**34**	0	0	0	1.00	1.00	1.00
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**80**	0	0	1.00	1.00	1.00
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**15**	0	1.00	1.00	1.00
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**158**	1.00	1.00	1.00
																								accuracy				1.00
																								macro avg		1.00	1.00	1.00

Table 9
Bi-RF results using hybrid features extracted using pre-trained VVG-16+ResNet-50.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	Prec.	Rec.	F1
0	**21**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
1	0	**18**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
2	0	0	**586**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
3	0	0	0	**312**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
4	0	0	0	0	**36**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
5	0	0	0	0	0	**21**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
6	0	0	0	0	0	0	**37**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
7	0	0	0	0	0	0	0	**37**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
8	0	0	0	0	0	0	0	0	**36**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
9	0	0	0	0	0	0	0	0	0	**33**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
10	0	0	0	0	0	0	0	0	0	0	**74**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
11	0	0	0	0	0	0	0	0	0	0	0	**88**	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
12	0	0	0	0	0	0	0	0	0	0	0	0	**45**	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
13	0	0	0	0	0	0	0	0	0	0	0	0	0	**35**	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**26**	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**33**	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**34**	0	0	0	0	0	0	0	0	1.00	1.00	1.00
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**28**	0	0	0	0	0	0	0	1.00	1.00	1.00
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**37**	0	0	0	0	0	0	1.00	1.00	1.00
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**20**	0	0	0	0	0	1.00	1.00	1.00
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**20**	0	0	0	0	1.00	1.00	1.00
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**25**	0	0	0	1.00	1.00	1.00
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**71**	0	0	1.00	1.00	1.00
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**21**	0	1.00	1.00	1.00
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**174**	1.00	1.00	1.00
																								accuracy				1.00
																								macro avg		1.00	1.00	1.00

Table 10
Bi-KNN results using hybrid features extracted using pre-trained VVG-16+ResNet-50.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	Prec.	Rec.	F1
0	**20**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
1	0	**26**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
2	0	0	**594**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
3	0	0	0	**322**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
4	0	0	0	0	**34**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
5	0	0	0	0	0	**24**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
6	0	0	0	0	0	0	**27**	1	0	0	0	0	0	0	0	0	0	0	0	0	1	4	0	0	0	0.87	0.82	0.84
7	0	0	0	0	0	0	3	**22**	0	0	0	0	0	0	0	0	0	0	0	0	1	2	0	0	0	0.96	0.79	0.86
8	0	0	0	0	0	0	0	0	**25**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
9	0	0	0	0	0	0	0	0	0	**30**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
10	0	0	0	0	0	0	0	0	0	0	**83**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
11	0	0	0	0	0	0	0	0	0	0	0	**80**	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
12	0	0	0	0	0	0	0	0	0	0	0	0	**46**	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
13	0	0	0	0	0	0	0	0	0	0	0	0	0	**34**	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**25**	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**36**	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**32**	0	0	0	0	0	0	0	0	1.00	1.00	1.00
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**32**	0	0	0	0	0	0	0	1.00	1.00	1.00
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**33**	0	0	0	0	0	0	1.00	1.00	1.00
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**17**	0	0	0	0	0	1.00	1.00	1.00
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**17**	6	0	0	0	0.68	0.74	0.71
21	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	6	**18**	0	0	0	0.60	0.72	0.65
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**74**	0	0	1.00	1.00	1.00
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**21**	0	1.00	1.00	1.00
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**171**	1.00	1.00	1.00
																								accuracy				0.99
																								macro avg		0.96	0.96	0.96

Table 11
Bi-LR results using hybrid features extracted using pre-trained VVG-16+ResNet-50.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	Prec.	Rec.	F1
0	**25**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
1	0	**18**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
2	0	0	**545**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
3	0	0	0	**318**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
4	0	0	0	0	**38**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
5	0	0	0	0	0	**22**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
6	0	0	0	0	0	0	**35**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
7	0	0	0	0	0	0	0	**52**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
8	0	0	0	0	0	0	0	0	**36**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
9	0	0	0	0	0	0	0	0	0	**33**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
10	0	0	0	0	0	0	0	0	0	0	**77**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
11	0	0	0	0	0	0	0	0	0	0	0	**92**	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
12	0	0	0	0	0	0	0	0	0	0	0	0	**51**	0	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
13	0	0	0	0	0	0	0	0	0	0	0	0	0	**31**	0	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**16**	0	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**38**	0	0	0	0	0	0	0	0	0	1.00	1.00	1.00
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**23**	0	0	0	0	0	0	0	0	1.00	1.00	1.00
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**27**	0	0	0	0	0	0	0	1.00	1.00	1.00
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**38**	0	0	0	0	0	0	1.00	1.00	1.00
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**19**	0	0	0	0	0	1.00	1.00	1.00
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**26**	0	0	0	0	1.00	1.00	1.00
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**30**	0	0	0	1.00	1.00	1.00
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**77**	0	0	1.00	1.00	1.00
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**22**	0	1.00	1.00	1.00
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**179**	1.00	1.00	1.00
																								accuracy				1.00
																								macro avg		1.00	1.00	1.00

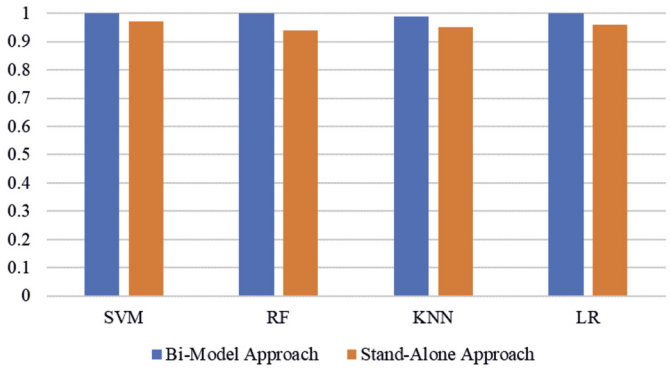

Fig. 6. Comparison between the Bi-model approach and stand-alone models.

Table 12
Results of models using pre-trained VVG-16 extracted features.

Model	Accuracy	Precision	Recall	F1
SVC	0.98	0.92	0.93	0.92
RF	0.97	0.94	0.93	0.93
KNN	0.98	0.95	0.95	0.95
LR	0.97	0.92	0.92	0.92

Table 13
Results of models using pre-trained ResNet-50 extracted features.

Model	Accuracy	Precision	Recall	F1
SVC	0.98	0.96	0.96	0.96
RF	0.97	0.94	0.93	0.93
KNN	0.98	0.95	0.94	0.95
LR	0.97	0.96	0.95	0.95

4.3. Experiments using separate features from VVG16 and ResNet-50

Without combining the features, further analysis is carried out using separate features from VVG16 and ResNet-50 models. Single features are used. In comparison with the proposed approach, we compared the performance of selected models using the individual VVG-16 features and ResNet-50 features. Table 12 shows the results of using VVG-16 features. The performance of KNN is significant compared to other models because KNN requires small features set for a good fit compared to LR and SVC, which work better using a large feature space. VVG-16 generates 1028 features that are enough for KNN but not good for other LR, SVC, and RF models. Consequently, the performance of SVC, RF, and LR is slightly poor than that of KNN in terms of F1 score.

Table 13 contains the results of machine learning models using features from the ResNet-50 model. The performance of all models is good with ResNet-50 compared to VVG16 features. SVC is significantly better with a 0.96 F1 score, followed by the KNN and LR with a 0.95 F1 score. These results show the effectiveness of KNN on the small feature set and show the linear model results for more correlated linearly separable data.

For a better understanding, the feature sets from VVG16, and ResNet-50 combined features of these models, and the features from the original Maligm dataset are illustrated in Fig. 7. The visualization reveals that features become more separable when combined for VVG16 and ResNet-50, providing the more appropriate features set for the models' training. Consequently, the models' training is efficient, leading to better performance than when trained using individual features from VVG16 and ResNet-50.

Table 14
Results of models using Gray Scale features.

Model	Accuracy	Precision	Recall	F1
SVC	0.97	0.95	0.95	0.95
RF	0.97	0.94	0.93	0.93
KNN	0.48	0.77	0.67	0.66
LR	0.97	0.96	0.95	0.95

Table 15
Results of deep learning models with original features set.

Model	Accuracy	Precision	Recall	F1
CNN	0.97	0.96	0.96	0.96
InceptionV3	0.97	0.92	0.92	0.92
EfficientNetB0	0.97	0.96	0.96	0.96

4.4. Performance using original features from maligm dataset

Table 14 shows the performance of models with the features set obtained from the original gray-scale images of the Maligm dataset. The number of features in the original dataset is higher than VVG-16, ResNet-50, and a combination of both. A total of 67500 features is available from the dataset, a large feature set and good for linear models such as LR and SVC. As a result, each SVC and LR obtained an F1 score of 0.95, followed by RF with a 0.93 F1 score. The KNN shows the worst performance due to the large feature set and obtains only a 0.66 F1 score.

4.5. Results using deep learning models

For a better comparison, three deep learning models are also implemented with one custom CNN model while two pre-trained models, including InceptionV3 and EffiencientB0.

These models are trained using the original features from the Maligm dataset, and performance is evaluated regarding the accuracy, precision, recall, and F1 Score. The models are trained using 50 epochs, and accuracy and validation for each epoch are shown in Fig. 8.

Table 15 contains the results of deep learning models indicating a 0.96 F1 score each from the CNN and EfficientNetBo whereas the F1 score for the InceptionV3 is 0.92. Despite the large feature set, the performance of these models is inferior to the proposed Bi-model architecture where the models can obtain state-of-the-art performance.

Fig. 9 shows the comparison between machine learning and deep learning models. According to the results, machine learning models outperform in terms of all evaluation parameters. Deep learning models are not significant as our Bi-Model approach because our Bi-Model approach used two models sequentially to achieve significant accuracy.

4.6. Results with K-fold cross-validation

We also performed 10-fold cross-validation with the proposed approach to show its significance. The performance of all models is significant with 10-fold cross-validation as Bi-SVC, Bi-LR, and Bi-RF show a 1.00 mean accuracy score and 0.00 standard deviation (SD). At the same time, Bi-KNN has a 0.99 mean accuracy and 0.001 SD, as shown in Table 16. Usually KNN is not good on a large feature set but in the proposed approach, since the feature set is reduced by transforming a large feature set into a small feature set, KNN also performs and obtains superior results.

4.7. Computational time

The computational complexity of the models is presented here in terms of the training time. The training time for Bi-Models and

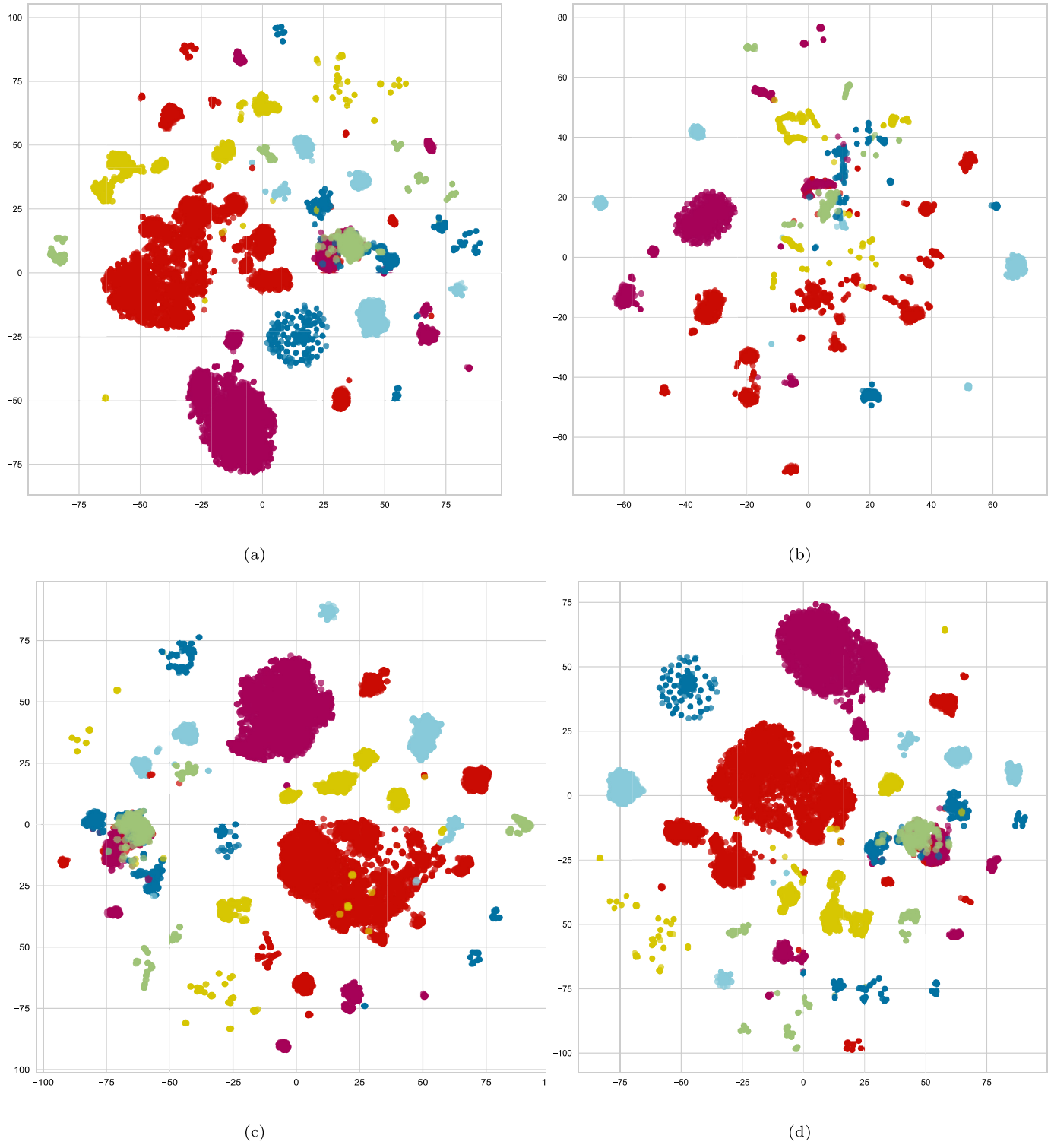

Fig. 7. Feature set visualization, (a) Combined features from VVG16 and ResNet-50, (b) Features from the original Maligm dataset, (c) For ResNet-50, and (d) For VVG16.

stand-alone models is presented using 10-fold cross-validation. Table 17 shows the computational time (in seconds) of learning models with different approaches. Results indicate that Bi-Models take more time as compared to stand-alone models because Bi-models are working sequentially. The first model generates features for the second model to be trained on and this whole procedure takes more time as compared to training a single model. The computational cost of Bi-Models, of course, is higher than compared to an individual model but for malware detection applications, accuracy is more important as compared to time. So, we can comprise on time for an efficient approach in terms of accuracy.

4.8. Comparison with state-of-the-art approaches

The performance of the proposed approach is compared with several state-of-the-art approaches that utilized the same dataset, i.e., the Malimg dataset for experiments. Predominantly, these studies used pre-trained models in their approach such as [10]

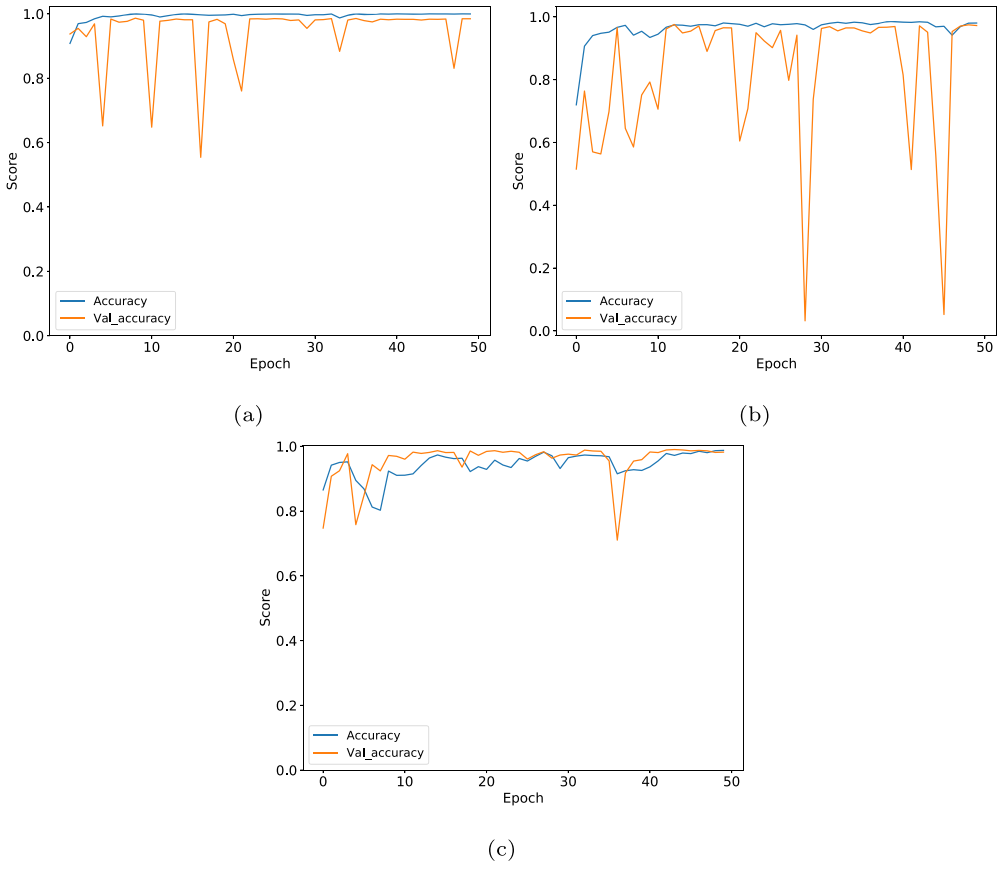

Fig. 8. Per epoch accuracy and validation accuracy for, (a) CNN, (b) InceptionV3 and (c) EfficientNetB0.

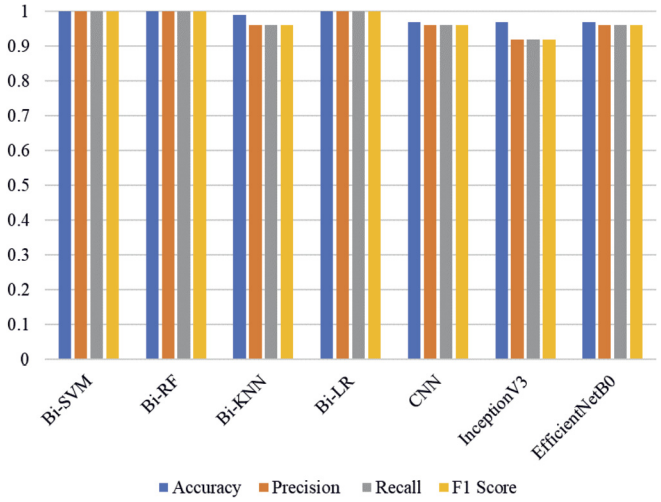

Fig. 9. Comparison between machine learning and deep learning approach.

Table 16
Results of Bi-Models using 10 fold cross validation.

Model	Accuracy	SD
Bi-SVC	1.00	+/- 0.00
Bi-RF	1.00	+/- 0.00
Bi-KNN	0.99	+/- 0.01
Bi-LR	1.00	+/- 0.00

used AlexNet, ResNet-152 while [13] used ResNeXt for malware detection. Whereas some studies combined machine learning and deep learning approaches to obtain higher performance, such as [11] combined MLP and SVM while [23] combined SFTA with KNN and [29] combined CNN with XG-Boost. Despite the best results from these studies on the same dataset, the results from the proposed approach surpass those performances as shown in Table 18.

Table 17
Computational cost of learning models using hybrid feature set.

Model	Time (sec)	Model	Time (sec)
Bi-SVM	71.143	Bi-KNN	18.928
SVM	0.221	KNN	0.665
10 Fold SVM	119.139	10 Fold KNN	21.539
Bi-RF	188.306	Bi-LR	394.631
RF	4.529	LR	2.509
10 Fold RF	231.674	10 Fold LR	523.74

4.9. Statistical significant T-test

We deployed a statistical T-test to show the significance of the proposed approach Bi-Models with a transfer learning hybrid feature set. We implement a T-test on the results of the Bi-Model with combined features and other approaches. T-test gives results in terms of null hypothesis acceptance or rejection.

- **Null Hypothesis (H_o):** T-test accepts the null hypothesis when comparison results are statistically equal.
- **Alternative Hypothesis (H_a):** T-test rejects the null hypothesis and accepts the alternative hypothesis when comparison results are statistically significant.

T-test gives output as t-statistic score (t), and critical value (CV). If the t score is less than CV in the T-test, it rejects the

Table 18
Performance comparison with state-of-the-art approaches.

Ref.	Year	Approach	Accuracy
[13]	2020	ResNeXt	98.32%
[11]	2020	MLP-SVM	98.52%
[23]	2020	SFTA with KNN	99%
[16]	2021	Dense-Net model with re-weighting	98.23%
[29]	2021	CNN with XG-Boost	98.7%
[10]	2021	AlexNet and ResNet-152 with fully connected layers	97.78%
[12]	2021	Random CapsNet forest	96.7%
[20]	2021	CNN with Bi-LSTM	98.5%
[18]	2022	Deep transfer learning	98.92%
Current	2022	Transfer learning with SVC	99%
		Transfer learning with Bi-KNN	99%
		Transfer learning with Bi-SVC	100%
		Transfer learning with Bi-RF	100%
		Transfer learning with Bi-LR	100%

Table 19
T-test results to show the significance of the proposed approach.

Comparison	t	CV	H_o
Bi-Models + HF vs Models + HF	6.44	6.74 e^{-17}	Rejected
Bi-Models + HF vs Models + VVG-16 Features	6.48	6.74 e^{-17}	Rejected
Bi-Models + HF vs Models + ResNet-50 Features	7.41	6.74 e^{-17}	Rejected
Bi-Models + HF vs Models + Original Features	3.41	6.74 e^{-17}	Rejected

H_o indicating that the results are statistically significant. In our case T-test rejects the null hypothesis for all cases when we compared the proposed approach results with the other approaches, as shown in Table 19.

5. Conclusion

The wide use of internet-enabled mobile devices and expanding digital transformation have increased their proneness to malware attacks. In addition, with emerging and evolving attack paradigms over time, the numbers of such attacks have alarmingly increased. Consequently, the need for efficient, improved, and accurate malware detection approaches has been multifold. As a response to this need, a novel Bi-model architecture is presented that adopts the stacked structure of the same model to increase malware detection performance. For better training, features from VVG16 and ResNet-50 have been combined to enlarge the feature set. The conceding model in the Bi-model structure is trained on the combined features, while the preceding model utilizes the conceding model's predicted class probabilities to produce better results. Extensive experiments are performed involving 25 classes using individual features each from VVG16 and ResNet-50, combined features, using Bi-model, standalone models, deep learning, and pre-trained models like InceptionV3 and EfficientNetB0. The proposed model outperforms these models and recent state-of-the-art models on the Maligm dataset. A 100% accuracy is obtained using the combined features with Bi-SVC, Bi-RF, and Bi-LR using 25 classes of the Maligm dataset. In future work, we intend to enlarge the dataset by incorporating additional recent attack types in the dataset and perform experiments on an even higher number of classes. Currently, the proposed approach has a high computational cost in terms of training time which we intend to improve. We also intend to incorporate more data for recent attacks and analyze the performance of the proposed approach.

CRediT authorship contribution statement

Furqan Rustam: Conceptualization, Methodology, Writing – original draft. **Imran Ashraf:** Conceptualization, Supervision, Writing – review & editing. **Anca Delia Jurcut:** Data curation, Software, Visualization. **Ali Kashif Bashir:** Investigation, Supervision, Validation. **Yousaf Bin Zikria:** Investigation, Project administration, Validation.

Declaration of competing interest

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

References

[1] How many smartphones are in the world?, https://www.bankmycell.com/blog/how-many-phones-are-in-the-world. (Accessed 20 January 2022).
[2] Cybercrime to cost the world $10.5 trillion annually by 2025 https://cybersecurityventures.com/cybercrime-damages-6-trillion-by-2021/#::text=The%20latest%20forecast%20is%20for,every%2040%20seconds%20in%202016. (Accessed 20 January 2022).
[3] VGG16 - convolutional network for classification and detection, https://neurohive.io/en/popular-networks/vgg16/. (Accessed 20 January 2022).
[4] ResNet50 - pre-trained models for image classification, https://www.analyticsvidhya.com/blog/2020/08/top-4-pre-trained-models-for-image-classification-with-python-code/. (Accessed 20 January 2022).
[5] Svm kernels https://dataaspirant.com/svm-kernels//. (Accessed 20 January 2022).
[6] R. Abbasi, A. Kashif Bashir, J. Chen, A. Mateen, J. Piran, F. Amin, B. Luo, Author classification using transfer learning and predicting stars in co-author networks, Softw. Pract. Exp. 51 (3) (2021) 645–669.
[7] R. Agarwal, S. Patel, S. Katiyar, S. Nailwal, Malware classification using automated transmutation and CNN, in: Advanced Computing and Intelligent Technologies, Springer, 2022, pp. 73–81.
[8] G. Amato, F. Falchi, KNN based image classification relying on local feature similarity, in: Proceedings of the Third International Conference on Similarity Search and Applications, 2010, pp. 101–108.
[9] I. Ashraf, Y. Park, S. Hur, S.W. Kim, R. Alroobaea, Y.B. Zikria, S. Nosheen, A survey on cyber security threats in IoT-enabled maritime industry, IEEE Transactions on Intelligent Transportation Systems.
[10] Ö. Aslan, A.A. YILMAZ, A new malware classification framework based on deep learning algorithms, IEEE Access.
[11] A. Bensaoud, N. Abudawaood, J. Kalita, Classifying malware images with convolutional neural network models, Int. J. Netw. Secur. 22 (6) (2020) 1022–1031.
[12] A. Çayır, U. Ünal, H. Dağ, Random CapsNet forest model for imbalanced malware type classification task, Comput. Secur. 102 (2021) 102133.
[13] J.H. Go, T. Jan, M. Mohanty, O.P. Patel, D. Puthal, M. Prasad, Visualization approach for malware classification with ResNeXt, in: 2020 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2020, pp. 1–7.
[14] H. Hassan, A.K. Bashir, M. Ahmad, V.G. Menon, I.U. Afridi, R. Nawaz, B. Luo, Real-time image dehazing by superpixels segmentation and guidance filter, J. Real-Time Image Process. 18 (5) (2021) 1555–1575.

[15] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[16] J. Hemalatha, S.A. Roseline, S. Geetha, S. Kadry, R. Damaševičius, An efficient densenet-based deep learning model for malware detection, Entropy 23 (3) (2021) 344.
[17] M.N.S. Jahromi, P. Buch-Cardona, E. Avots, K. Nasrollahi, S. Escalera, T.B. Moeslund, G. Anbarjafari, Privacy-constrained biometric system for non-cooperative users, Entropy 21 (11) (2019) 1033.
[18] S. Kumar, B. Janet, DTMIC: deep transfer learning for malware image classification, J. Inf. Secur. Appl. 64 (2022) 103063.
[19] R. Lyer, The political economy of cyberspace crime and security, Academia.edu.
[20] N. Marastoni, R. Giacobazzi, M. Dalla Preda, Data augmentation and transfer learning to classify malware images in a deep learning context, J. Comput. Virol. Hacking Tech. (2021) 1–19.
[21] A. Moses, S. Morris, Analysis of mobile malware: a systematic review of evolution and infection strategies, J. Inf. Secur. Cybercrimes Res. 4 (2) (2021) 1–29.
[22] L. Nataraj, V. Yegneswaran, P. Porras, J. Zhang, A comparative assessment of malware classification using binary texture analysis and dynamic analysis, in: Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence, 2011, pp. 21–30.
[23] M. Nisa, J.H. Shah, S. Kanwal, M. Raza, M.A. Khan, R. Damaševičius, T. Blažauskas, Hybrid malware classification method using segmentation-based fractal texture analysis and deep convolution neural network features, Appl. Sci. 10 (14) (2020) 4966.
[24] W.S. Noble, What is a support vector machine?, Nat. Biotechnol. 24 (12) (2006) 1565–1567.
[25] A. Qamar, A. Karim, V. Chang, Mobile malware attacks: review, taxonomy & future directions, Future Gener. Comput. Syst. 97 (2019) 887–909.
[26] A.A. Reshi, F. Rustam, A. Mehmood, A. Alhossan, Z. Alrabiah, A. Ahmad, H. Alsuwailem, G.S. Choi, An efficient CNN model for COVID-19 disease detection based on x-ray image classification, Complexity (2021).
[27] F. Rustam, M.A. Siddique, H.U.R. Siddiqui, S. Ullah, A. Mehmood, I. Ashraf, G.S. Choi, Wireless capsule endoscopy bleeding images classification using CNN based model, IEEE Access 9 (2021) 33675–33688.
[28] F. Rustam, A.A. Reshi, W. Aljedaani, A. Alhossan, A. Ishaq, S. Shafi, E. Lee, Z. Alrabiah, H. Alsuwailem, A. Ahmad, et al., Vector mosquito image classification using novel RIFS feature selection and machine learning models for disease epidemiology, Saudi J. Biol. Sci. 29 (1) (2022) 583–594.
[29] S. Saadat, V.J. Raymond, Malware classification using CNN-XGBoost model, in: Artificial Intelligence Techniques for Advanced Computing Applications, Springer, 2021, pp. 191–202.
[30] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint, arXiv:1409.1556.
[31] J. Singh, J. Singh, A survey on machine learning-based malware detection in executable files, J. Syst. Archit. 112 (2021) 101861.
[32] Y. Sun, A.K. Bashir, U. Tariq, F. Xiao, Effective malware detection scheme based on classified behavior graph in IIoT, Ad Hoc Netw. 120 (2021) 102558.
[33] D. Vasan, M. Alazab, S. Wassan, H. Naeem, B. Safaei, Q. Zheng, IMCFN: image-based malware classification using fine-tuned convolutional neural network architecture, Comput. Netw. 171 (2020) 107138.

Furqan Rustam received the Master of Science degree in computer science from the Department of Computer Science, Khwaja Fareed University of Engineering and Information Technology (KFUEIT), Rahim Yar Khan, Pakistan. He worked as a Research Assistant with the Fareed Computing & Research Center, KFUEIT. He is currently working as a Lecturer with the Department of Software Engineering, University of Management and Technology, Lahore, Pakistan. His current research interests include data mining, machine learning, and artificial intelligence, mainly working on creative computing and supervised machine learning.

Imran Ashraf received his Ph.D. in Information and Communication Engineering from Yeungnam University, South Korea in 2018, and the M.S. degree in computer science from the Blekinge Institute of Technology, Karlskrona, Sweden, in 2010 with distinction. He has worked as a postdoctoral fellow at Yeungnam University, as well. He is currently working as an Assistant Professor at the Information and Communication Engineering Department, Yeungnam University, Gyeongsan, South Korea. His research areas include positioning using next-generation networks, communication in 5G and beyond, location-based services in wireless communication, smart sensors (LIDAR) for smart cars, and data analytics.

Anca Delia Jurcut (Member, IEEE) has been an Assistant Professor with the School of Computer Science, University College Dublin since 2015. She worked as a Postdoctoral Researcher with the University of Limerick and as a Software Engineer with IBM, Dublin, in the area of data security and formal verification. Her research interests include security protocols design and analysis, automated techniques for formal verification, network security, attack detection and prevention techniques, security for the Internet of Things, and applications of blockchain for security and privacy. She has several key contributions in research focusing on detection and prevention techniques of attacks over networks, the design and analysis of security protocols, automated techniques for formal verification, and security for mobile edge computing.

Ali Kashif Bashir (Senior Member, IEEE) received the B.Sc. degree (Hons.) in computer forensics and security from the Department of Computing and Mathematics, Manchester Metropolitan University, U.K., and the Ph.D. degree in computer science and engineering from Korea University, South Korea. His past assignments include an Associate Professor of ICT with the University of the Faroe Islands, Denmark; Osaka University, Japan; Nara College, National Institute of Technology, Japan; the National Fusion Research Institute, South Korea; Southern Power Company Ltd., South Korea; and Seoul Metropolitan Government, South Korea. He is currently a Senior Lecturer/an Associate Professor and the Program Leader of the Department of Computing and Mathematics, Manchester Metropolitan University. He is also with the School of Electrical Engineering and Computer Science, National University of Science and Technology (NUST), Islamabad, as an Adjunct Professor, and the School of Information and Communication Engineering, University of Electronics Science and Technology of China (UESTC) as an Affiliated Professor and the Chief Advisor of the Visual Intelligence Research Center. He has worked on several research and industrial projects of South Korean, Japanese and European agencies and Government ministries. In his career, he has received over 2.5 Million USD funding. He has authored over 180 research articles, received funding as the PI and the Co-PI from research bodies of South Korea, Japan, EU, U.K., and Middle East, and supervising/co-supervising several graduate (M.S. and Ph.D.) students. His research interests include the Internet of Things, wireless networks, distributed systems, network/cyber security, network function virtualization, and machine learning. Dr. Bashir is a member of the IEEE Industrial Electronic Society and ACM. He is leading many conferences as the chair (program, publicity, and track) and had organized workshops in flagship conferences, like IEEE Infocom, IEEE Globecom, and IEEE Mobicom. He is serving as the Editor in-Chief for the IEEE Future Directions Newsletter. He is also serving as an Area Editor for KSII Transactions on Internet and Information Systems, and an Associate Editor for IEEE Internet of Things Magazine, IEEE Access, Computer Science, IET Quantum Communication, and Journal of Plant Diseases and Protection. He is a Distinguished Speaker of ACM.

Yousaf Bin Zikria (SM' 17) is currently working as an Assistant Professor in the Department of Information and Communication Engineering, College of Engineering, Yeungnam University, Gyeongsan-Si, South Korea. He received a Ph.D. degree from the Department of Information and Communication Engineering, Yeungnam University, Korea, in 2016. He has more than ten years of experience in research, academia, and industry in the field of Information and Communication Engineering and Computer Science. He authored more than 60 scientific peer-reviewed journals, conferences, patents, book chapters.