IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 1, 2021, accepted October 8, 2021, date of publication October 13, 2021, date of current version October 20, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3119655

Plant Disease Detection in Imbalanced Datasets Using Efficient Convolutional Neural Networks With Stepwise Transfer Learning

MOBEEN AHMAD, MUHAMMAD ABDULLAH, HYEONJOON MOON, AND DONGIL HAN, (Member, IEEE)
Department of Computer Engineering, Sejong University, Seoul 05006, South Korea

Corresponding author: Dongil Han (dihan@sejong.ac.kr)

This work was support in part by the Cooperative Research Program for Agriculture Science and Technology Development through the Rural Development Administration, Republic of Korea under Project PJ015686, and in part by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government [Ministry of Science and ICT (MSIT)] under Grant 2021R1F1A106168711.

ABSTRACT Convolutional neural networks have demonstrated state-of-the-art performance in image classification and various other computer vision tasks. Plant disease detection is an important area of deep learning which has been addressed by many recent methods. However, there is a dire need to optimize these solutions for resource-constrained portable devices such as smartphones. This is a challenging problem because deep learning models are resource extensive in nature. This paper proposes an efficient method to systematically classify plant disease symptoms using convolutional neural networks. These networks are memory efficient and when coupled with the proposed training configuration it enables rapid development of industrial applications by reducing the training times. Another critical problem arises with the improper distribution of samples among classes known as the class imbalance problem, which is addressed by employing a simple statistical methodology. Transfer learning is a known technique for training small datasets which transfers pre-trained weights learned on a large dataset. However, during transfer learning, negative transfer learning is a common problem. Therefore, a stepwise transfer learning approach is proposed which can help in fast convergence, while reducing overfitting and preventing negative transfer learning during knowledge transfer across domains. The system is trained and evaluated on two plant disease datasets i.e., PlantVillage (a publicly available dataset) and pepper disease dataset provided by the National Institute of Horticultural and Herbal Science, Republic of Korea. The pepper dataset is particularly challenging as it contains images from different parts of the plant such as the leaf, pulp, and stem. The proposed system has dominated the previous works on the PlantVillage dataset and achieved 99% and 99.69% accuracy on the Pepper dataset and PlantVillage datasets, respectively.

INDEX TERMS Disease detection, convolutional neural networks, image classification, MobileNet, Internet-of-Things, transfer learning.

I. INTRODUCTION

Agricultural yield is vulnerable to various biotic stresses which incur significant losses in terms of reduced production. Food safety, nutrition, and agricultural economy are interlinked in a vicious cycle [1], which poorly affects the developing and underdeveloped countries and leads to health and economic problems. In many under-developed countries, major crop yield is produced by small farmers for example, only in Africa, 80% of crops are produced by small farmers [2]. It is an alarming situation as lesser resources are available for smallholder farmers, which implies that the majority of agricultural production of under-developed countries is in danger. To properly manage a disease, early action is required which necessitates an early-stage disease diagnosis [3]. However, it is quite challenging due to the unavailability of expert opinion which usually delays the preliminary steps to mitigate the disease at earlier stages [3]. This delay is responsible

The associate editor coordinating the review of this manuscript and approving it for publication was Aysegul Ucar.

for significant damage to the crops every year. According to [4], worldwide annual crop losses due to plant diseases are estimated to be $60 billion.

Plant diseases are diagnosed based on their visual symptoms which usually appear on different regions of the plant such as leaf, stem, and pulp. However, expert knowledge is required to correctly diagnose disease classes. It is difficult to reach out the rural areas especially in underdeveloped countries where most of the crop is produced by smallholder farmers. With the use of advanced technology, it is possible to get an expert-level diagnosis. For instance, smartphones are now vastly available at affordable prices; this along with widespread coverage of the internet can be a suitable platform for a smartphone-based service for disease diagnosis. As crop diseases are diagnosable based on visual symptoms that might appear at different parts, a farmer can take a picture of the part of the plant, and the mobile disease detection system will identify and label the disease. This can help reduce the crop losses by minimizing the involved steps in a usual diagnostic process which involves an expert visiting the farm. Furthermore, the scope of the proposed system can be extended beyond the correct identification of plant disease by suggesting possible remedies as well as serve advertisements from various agricultural product vendors.

In the past various machine learning, and advanced deep learning techniques have been applied to the problem of plant disease detection. Some work [5] is done in the domain of plant disease detection, [6] and [7] address this problem by applying step-by-step image processing techniques such as image acquisition, image pre-processing, image segmentation, feature extraction, and classification. Most of these methods utilized hand-crafted features and conventional machine learning techniques. The hand-crafted features such as SIFT, HoG, color, and shape-based feature extraction are used to extract discriminant information from images which is then fed to a classifier. The classifier is then trained to learn the discriminant features by learning a complex distribution which can differentiate among different classes. Sannakki *et al.* [8] used K-Means clustering to extract color features to diagnose disease in pomegranate leaves. Patil and SZambre [9] extracted shape and color-based features which were then classified using Support Vector Machine (SVM). Conventional methods often involve extensive image preprocessing which poses an overhead as compared to the advanced deep learning approaches. Such preprocessing includes image resizing, denoising by applying Gaussian or some smoothing filter, etc. These preprocessing steps induce an overhead in the disease detection pipeline.

Conventional methods are still relevant where the data is sparser, and features are well-defined. However, if data availability is not a problem, advanced deep learning methods tend to surpass conventional hand-crafted feature methods because Convolutional Neural Networks (CNN) can extract the most discriminant features from the images. CNNs extract information from image pixels at different levels, starting from very basic color, shape information to very high-level information such as texture, object shapes, curves, etc. Furthermore, the preprocessing steps such as noise removal are also not required. The recent advent of deep learning [10] has helped the disease detection problem and advanced deep learning-based solutions [11], [12], and [13] have been proposed in the domain of plant disease detection. To name a few [14]–[16], [12] have shown good performance on public datasets and private datasets. In [17], the authors used two popular CNN architectures, GoogleNet [18] and AlexNet [19] to classify disease on leaf images and report results on a publicly available PlantVillage [20] dataset. In [21], the authors used apple leaf images from the PlantVillage dataset and achieved 90.4% accuracy for black rot and healthy leaf classification. In [16], the authors tested on a preliminary version of the PlantVillage dataset and achieved 99.53% accuracy.

However, the majority of the work is done on datasets with limited samples which pose a risk of poor performance in a real-world scenario. Apart from that, some datasets such as PlantVillage are focused on images taken in a lab environment which may not be the accurate representation of real-world data. The real-world scenario is comparatively challenging, due to an immense number of varying factors that can interfere with the image capturing process. For example, illumination, the direction of the incident light, occlusion from other leaves, and shadows caused by other leaves may result in an image with varying illumination at different regions. All these factors are responsible for the degradation of plant disease detection systems. Furthermore, most of the recent work is specific to leaf images, stem images, or fruit images. A unified deep learning system is required for ease of use in practical application.

Class imbalances play a major role in the degradation of the training phase [22]. In the case of plant disease detection, most of the available datasets suffer from class imbalance. For instance, PlantVillage [20] contains 54,309 images. 39,218 images belong to 5 major classes of diseases e.g., fungi, mold, virus, and mite, whereas 15,085 images belong to healthy plant images. As seen in figure 1 (a), a clear class imbalance can be seen among several PlantVillage classes. Similarly, an imbalance can be observed among samples that belong to various classes of plants as seen in figure 1 (b).

Therefore, a system needs to be developed that can accommodate real-world factors such as poor lighting, background noise, class imbalance, as well as challenges posed by hardware limitations of hand-held devices. Moreover, the lack of enough training data needs to be compensated by employing data augmentation methods. The impact of such technologies is so immense that it will not only help to increase the overall yield by small farms, but it will allow governments and large corporations to scale farming at a larger level while working with a smaller number of experts and lower cost of human resource. The above-mentioned issues are addressed in this paper with the following contributions:

1. An efficient CNN-based plant-disease detection method is proposed which can diagnose diseases in plants in an uncontrolled environment.
2. A statistical class-balancing method is proposed which can balance the class frequencies in an imbalanced dataset.
3. The stepwise transfer learning method is used to efficiently train CNNs on small datasets by enabling the rapid development of industrial applications.
4. A MobileNet based method is proposed which can be deployed on hand-held devices due to reduced hardware resource requirements because, it does not have a dense layer, which makes it suitable for hand-held devices.
5. A stepwise transfer learning approach is proposed which can help to reduce the convergence time of any CNN architecture. It is observed that it can help reduce overfitting as well as negative transfer learning.
6. Finally, the proposed model has achieved accuracy comparable to state-of-the-art methods with less resource requirement.

The rest of the paper is organized as follows. Description of our proposed method is provided in section II with its experimental validation in section III. Finally, we discuss the proposed methods in section IV and conclude the paper in section V with key findings and potential future directions.

II. METHODOLOGY

The purpose of this work is to allow computer vision technologies to help address common problems faced in classifying plant disease symptoms. The techniques proposed are not only applicable for plant disease detection problems but can be utilized for most industrial applications where rapid development of machine learning algorithms is desired. In the past advanced convolutional neural networks have performed well on such tasks given that, enough data is available for training. However, it is not always possible to get huge amounts of images for a specific crop disease pair. This data shortage problem and class imbalance problem are addressed by using data augmentation techniques which can make CNNs able to learn representative features of disease classes. Moreover, the data collected in the wild may suffer from various deficiencies like bad lighting, low image quality, and confusing symptoms which result in misclassifications. Also, the typical CNNs are not well-suited for resource-constrained hand-held devices due to their higher computational costs, large model size, and higher running time. In this paper, we carefully designed the experimental setup to analyze the effect of these conditions and try to overcome these challenges in our proposed method. To reduce the model size, a CNN is used without a dense layer that eliminates a large number of add-multiplication operations resulting in fast and small model size. In upcoming sections, we will first discuss the datasets and their challenging characteristics as well as the approaches we utilized to enhance the performance of our proposed system.

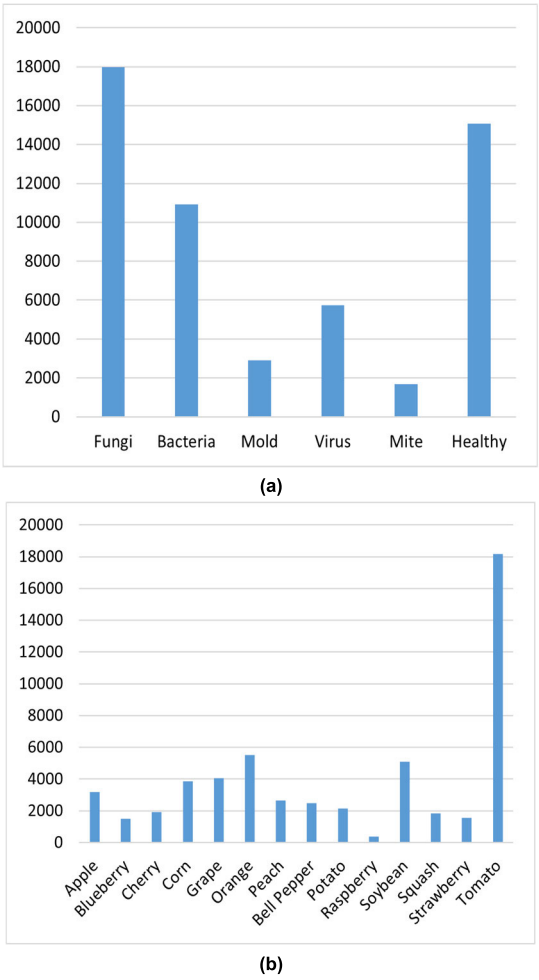


FIGURE 1. (a) Sample frequencies in PlantVillage dataset per disease category. (b) Sample frequencies in Plant Village dataset by plant classes.

A. DATASET DESCRIPTION

For experiments, we have used two datasets. The first dataset is provided by the National Institute of Horticultural and Herbal Science, Republic of Korea, while the other one is publicly available as PlantVillage [20] dataset.

1) PEPPER DATASET

This dataset is built to assist the training for the visual symptoms of the described diseases which pose vulnerability towards pepper plants. It includes a total of 99,507 images belonging to 24 classes of diseases precisely annotated with the help of experts and the support provided by the National Institute of Horticultural and Herbal Science, Republic of Korea. Since the application is targeted at common users and handheld devices, the images are captured using common digital cameras under varying daylight conditions. The dataset covers the scope of plant diseases more comprehensively as it features diseases on the different parts of the plants, e.g., stem, leaf, and pulp. Among 24 categories, 6 diseases are related to the pulp, 6 are related to stem, 9 are

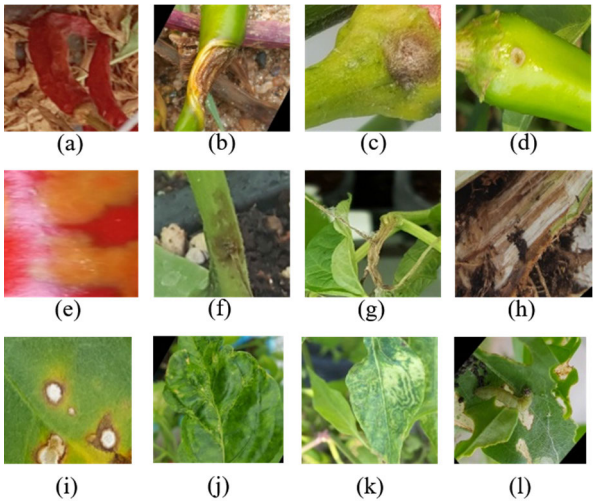


FIGURE 2. Some sample images from our pepper dataset. (a) Phytophthora blight (pulp), (b) Biter rot (pulp), (c) Gray mold (pulp), (d) Moth damage (pulp), (e) Tomato spotted wilt virus (pulp), (f) Tomato spotted wilt virus (stem), (g) Gray mold (stem), (h) Bacterial wilt (stem), (i) Bacterial leaf spot, (j) Damping off (leaf), (k) Tomato spotted wilt virus (leaf), (l) Gray mold (leaf).

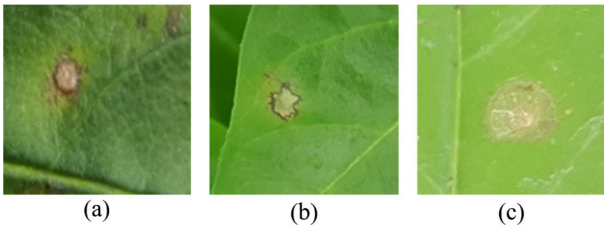


FIGURE 3. Some of the diseases show similar symptoms which makes them difficult to differentiate. (a) Bacterial spot (b) Bitter rot (c) Gray mold.

related to leaf, 2 are larva based, and 1 class combines healthy plant images from all three parts i.e., leaf, pulp, and stem. Some sample images from the said dataset can be seen in figure 2.

Dataset is challenging particularly because of two contrasting reasons: (i) Visually identifiable symptoms of some diseases are significantly like those of other diseases. For instance, bacterial spot, bitter rot, and gray mold show very similar visual symptoms on the leaf as shown in figure 3. On the contrary, (ii) Some diseases show very diverse visual symptoms which can be sometimes challenging to classify as the same class. For example, the disease southern blight tends to show diverse symptoms as shown in figure 4. Furthermore, detection of larvae-based diseases also poses a challenging task due to significant similarities among them as shown in figure 5.

2) PLANTVILLAGE DATASET

PlantVillage [20] dataset is a public dataset containing 38 crop disease pairs. In total there are 14 crops with 26 disease classes and 14 healthy plant classes. However, some plants do not have disease classes. The purpose of using

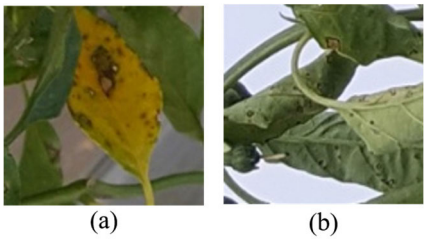


FIGURE 4. Some of the diseases show varying symptoms which makes it challenging to detect. Both (a) and (b) represent southern blight in leaf, hence pose a challenging problem for disease detection system.

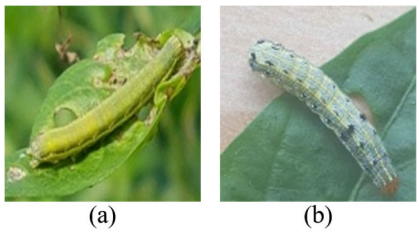


FIGURE 5. Some of the larvae-based diseases are also very challenging due to significant resemblance. (a) Spodoptera exigua larva (b) Spodoptera litura larva.

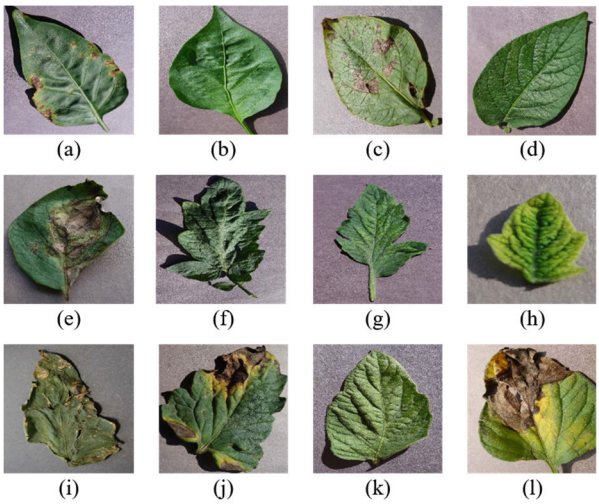


FIGURE 6. Some sample images from PlantVillage dataset. (a) Pepper bell – bacterial spot, (b) Pepper bell – healthy, (c) Potato – Early blight, (d) Potato – healthy, (e) Potato – late blight, (f) Tomato – target spot, (g) Tomato – tomato mosaic virus, (h) Tomato – tomato yellow leaf curl virus, (i) Tomato – bacterial spot, (j) Tomato – early blight, (k) Tomato – healthy, (l) Tomato – late blight.

a publicly available dataset was to evaluate our system for comparative analysis with other existing methods. Sample images are shown in figure 6.

B. OVERVIEW OF CLASSIFICATION PROCESS

A sophisticated methodology is proposed by incorporating various established processes to facilitate the process of classification and identification of diseases in crops. The classification pipeline consists of various steps as shown in figure 7, which is initiated by first manually analyzing

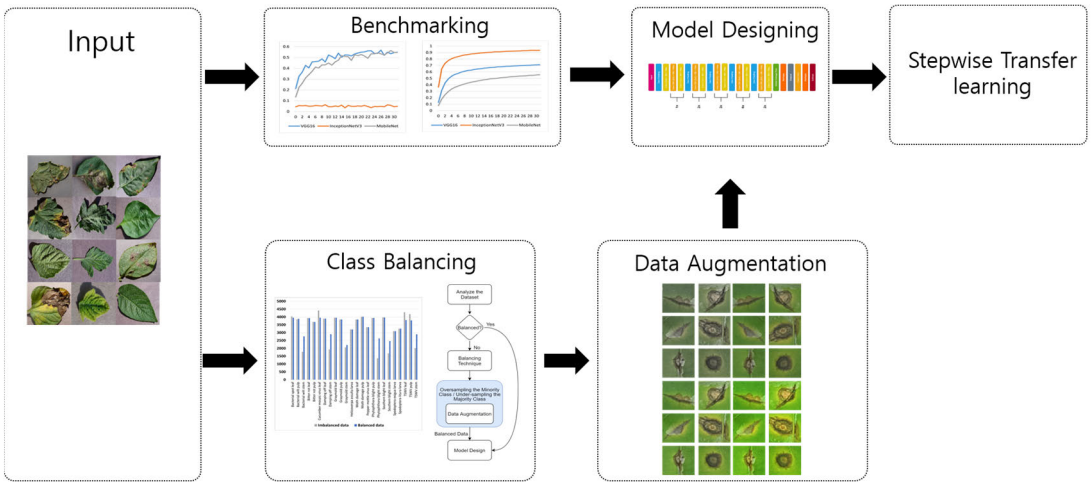


FIGURE 7. An overview of proposed methodology.

the dataset for class imbalances. Class imbalance occurs when some classes are under-represented as compared to others. This can cause the model to predict biased results. This problem is eradicated by employing data augmentation techniques for under-represented classes as well as removing some samples from the over-represented classes. After class-balancing next crucial step is model selection. Due to rapid advancement in machine learning, several state-of-the-art Convolutional Neural Networks (CNNs) are available which make it difficult to choose the most appropriate model for the problem at hand. We selected three different types of CNNs which are known to perform well on the ImageNet [23] dataset and benchmarked them on plant disease datasets. This way model selection step can be systemized instead of the hit-and-trial method. After the initial benchmarking and model selection, we train our dataset from scratch i.e., without using transfer learning techniques to realize the model capacity as compared to the given data. A balance between the capacity of the CNN model being used and the complexity of training data is important. Any imbalance can lead to the overfitting or underfitting of a model. Later, the underlying representations learned from ImageNet [23] dataset are hierarchically transferred to our model. The transfer learning experiments are further explained in the experiment section. An overview of the classification pipeline can be seen in figure 7.

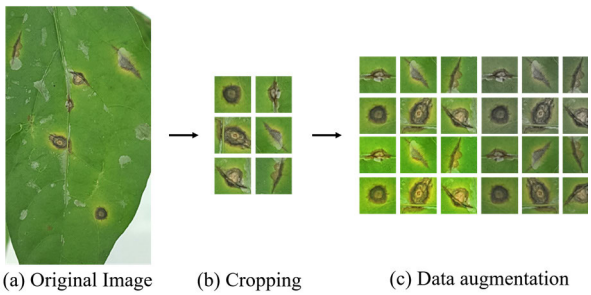


FIGURE 8. An original image is cropped such that the background information is neglected, and the symptoms are brought into focus. Furthermore, these cropped images are then rotated, flipped, applied different illumination variations for data augmentation.

1) DATA PREPARATION

The original image resolution is 1600 × 1200 for the pepper dataset, which is quite large as shown in figure 8(a). Therefore, the images were first cropped to remove the background and focus mostly on the diseased area as shown in figure 8(b). These crops were then randomly split into three portions, for training, validation, and testing. Finally, data augmentation is done to increase the sample size of the dataset. The augmentation techniques included, rotation, flipping, cropping, saturation fluctuation, and illumination as shown in figure 8(c). It is to be noted that the final results were also evaluated on 5-fold cross-validation to eliminate any skewness that might occur due to data preparation steps such as augmentation.

2) CLASS BALANCING

As discussed in section I, a class imbalance can lead to biased training, so the input dataset is analyzed for class imbalances. Any imbalances in class distribution can make the model 6 biased towards a specific class which can pose a deficiency in model accuracies and hence, degrading the overall performance of the model. For example, in the case of our Pepper crop dataset; phytophthora blight, damping off, bacterial wilt, and southern blight are under-represented as compared to other classes as shown in figure 10, so it is natural for the trained model to get biased towards other classes.

We can balance the dataset based on the sample frequencies by over-sampling the minority class by employing some data synthesis and augmentation methods and for the class with a large number of samples we under-sample it by

deleting some samples. In the case of over-sampling for under-represented classes, we need some image generation techniques. Two approaches are used to oversample the data: Adaptive synthetic sampling approach for imbalanced learning [24] and the Synthetic Minority Oversampling Technique [25] which involves over-sampling the minority class and under-sampling of the majority class to get the best results. Another technique that can be used for balancing the data is using Generative Adversarial Networks (GAN) for data augmentation [26]. Since, GAN-based augmentation is an overhead for a simple classification task, a more traditional methodology was used. Finally, we merge the results and do some further data augmentation to increase the overall number of training data, and then this balanced data is used for the model design phase as shown in figure 8.

3) DATA AUGMENTATION

Besides using more advanced techniques for data augmentation, some basic augmentation techniques were applied to all the classes to increase the number of samples and robustness towards unseen data. Images were scaled to incorporate the symptoms in different resolutions. Further augmentation steps include flipping, rotation, translation, adding noise, changing lighting conditions as shown in figure 8(c).

4) BENCHMARKING AND MODEL SELECTION

In this era of highly advanced deep learning technology which is aided by a huge number of powerful architectures, it is quite difficult to select one appropriate model that suites the dataset and target application best. Most of the advanced deep learning models are designed to exploit the huge computational power of modern hardware therefore they are resource-intensive. Handheld devices and various embedded systems in IoT environments still suffer from a lack of computational power and storage space, therefore, it is essential to perform initial benchmarking on mobile versions of deep learning models as well as large deep learning architectures. Initial benchmarking was done on the input dataset by employing multiple models with different levels of computational complexities trained with different configurations. The configurations include training from scratch, transfer learning, transfer learning with frozen layers, fine-tuning, and transfer learning for feature extraction. The experiments performed are shown in table 1. Among these models, we selected three models for further experiments. i.e., VGG-16 [27], Inception V3 [18], and different versions of MobileNet [28]. A summarized architecture of basic MobileNet is shown in figure 12. The results are analyzed to measure the extent to which the high-end model is overfitting, and the basic model is underfitting. Performance of selected benchmark models was also observed on the ImageNet dataset as presented in table 2. It can be seen that models with varying complexities are included in this study. The complexity is based on a total number of multiply-addition operations and a total number of trainable parameters as shown in table 2. It is to be noted

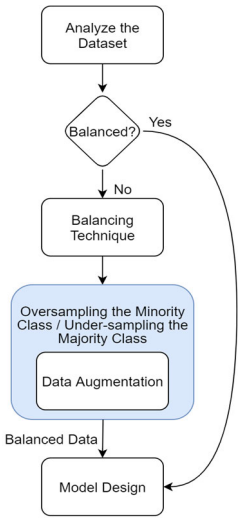


FIGURE 9. Flowchart representing class balancing algorithm where data augmentation is employed to oversample the minority class whereas majority class is under-sampled by simple sample removal procedure.

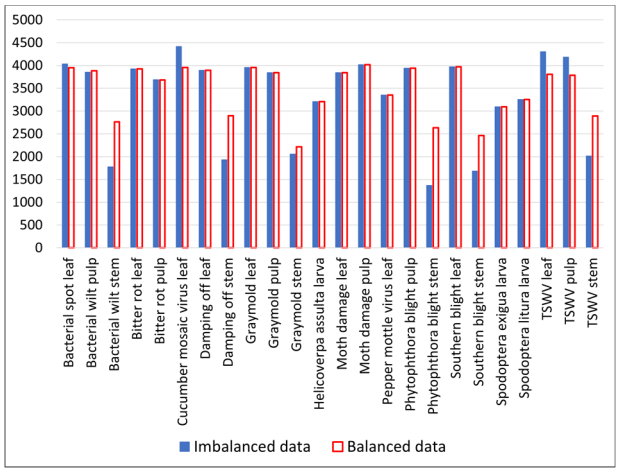


FIGURE 10. The number of images per class in the Pepper dataset before and after class balancing.

that the results of random initialization and initialization with ImageNet weights do not have a significant difference.

This suggests that generic datasets such as ImageNet are not suitable to be used as a feature extractor for specific domain applications such as plant disease detection. For initial benchmarking on the pepper dataset, these models were trained for 32 epochs as shown in figure 11. VGG-19 being the most computationally expensive model to train achieved 71% training accuracy but got only 54.9% validation accuracy. Inception V3 has a significantly smaller number of parameters as compared to VGG-16. Due to very deep architecture, it was able to achieve a very high training accuracy of 93.5%, however, the validation accuracy was very low. This demonstrates that this network is prone to overfitting as our dataset is not large enough to train a deep architecture. Finally, MobileNet was trained, and it achieved mediocre training accuracy of 55.68% and a

TABLE 1. Models benchmarked on the PlantVillage dataset during the initial model selection stage. All the experiments reported were conducted on a single GTX 1080 Ti GPU except Inception-V3, GoogleNet, and VGG-16_bn. Random initialization: the network was trained from scratch without freezing any layers. Finetuning (TL): the network was initialized with pre-trained weights and conventional transfer learning was used. Feature extraction (random): the network was initialized with random weights, and all the layers were frozen. Feature extraction (TL) states that the network was initialized with trained weights, all the layers remained frozen throughout the experiment and the network was simply used as a feature extractor.

Model	Training configuration	Training accuracy (%)	Epoch with best train accuracy (%)	5-fold cross-validation accuracy (%)	Time taken (200 epochs) (hours)	Model complexity (GMACs)	Trainable params	Total params
VGG-16*	Random initialization	98.67	33	**98.19**	67.01	15.53	134.42 M	134.42 M
	Finetuning (TL)	98.60	155	**98.03**	67.91			
	Feature extraction (random)	83.01	104	82.24	21.54		155.69 K	
	Feature extraction (TL)	83.51	38	82.60	21.36			
VGG-19_bn	Random initialization	98.57	43	97.96	118.72	19.69	139.73 M	139.73 M
	Finetuning (TL)	98.64	120	97.99	114.97			
	Feature extraction (random)	82.26	131	81.79	38.76		155.69 K	
	Feature extraction (TL)	82.46	51	81.67	39.26			
Resnet-50	Random initialization	99.23	81	98.59	49.89	4.12	23.59 M	23.59 M
	Finetuning (TL)	99.21	71	98.53	49.36			
	Feature extraction (random)	82.79	102	82.25	22.60		77.86 K	
	Feature extraction (TL)	82.70	37	82.00	19.18			
DenseNet-121	Random initialization	98.95	182	98.24	40.08	2.88	6.99 M	6.99 M
	Finetuning (TL)	98.72	68	98.30	45.28			
	Feature extraction (random)	78.90	198	78.14	16.82		38.95 K	
	Feature extraction (TL)	79.01	43	78.19	18.13			
SqueezeNet-1_0	Random initialization	93.85	169	93.43	16.91	0.75	754.92 K	754.92 K
	Finetuning (TL)	93.97	53	93.33	15.12			
	Feature extraction (random)	80.80	149	80.13	11.58		19.49 K	
	Feature extraction (TL)	81.18	80	80.56	11.30			
Inception-V3*	Random initialization	**99.57**	199	**99.00**	52.27	2.85	24.45 M	24.45 M
	Finetuning (TL)	**99.56**	186	**98.98**	53.49			
	Feature extraction (random)	78.61	130	78.11	16.30		107.08 K	
	Feature extraction (TL)	78.60	66	77.72	16.54			
GoogleNet*	Random initialization	98.02	195	97.18	15.57	1.51	5.64 M	5.64 M
	Finetuning (TL)	97.94	197	97.26	15.74			
	Feature extraction (random)	69.72	194	69.08	**6.37**		38.95 K	
	Feature extraction (TL)	69.42	167	68.59	6.43			
MobileNet-V1	Random initialization	97.13	26	96.51	20.05	0.58	3.25 M	3.25 M
	Finetuning (TL)	98.52	35	96.22	20.11			
	Feature extraction (random)	76.30	148	75.38	12.17		48.68 K	
	Feature extraction (TL)	75.83	164	74.29	12.06			
MobileNet-V2	Random initialization	98.27	74	97.78	18.18	0.32	2.27 M	2.27 M
	Finetuning (TL)	98.25	137	97.62	17.43			
	Feature extraction (random)	81.06	117	80.56	11.76		48.68 K	
	Feature extraction (TL)	81.69	78	81.21	11.19			
MobileNet-V3-Small	Random initialization	95.88	82	95.38	11.80	**0.06**	**1.56 M**	**1.56 M**
	Finetuning (TL)	96.12	113	95.48	11.20			
	Feature extraction (random)	76.10	124	75.99	**10.45**		**38.95 K**	
	Feature extraction (TL)	76.61	36	76.25	10.49			
MobileNet-V3-Large	Random initialization	98.12	183	**97.42**	15.93	0.23	4.25 M	4..25 M
	Finetuning (TL)	98.11	32	97.27	15.59			
	Feature extraction (random)	81.37	29	80.47	11.17		48.68 K	
	Feature extraction (TL)	81.35	**24**	80.71	11.10			

* These experiments were performed on a different machine with a GTX 2080 GPU.

TABLE 2. Comparison of selected benchmarked models on ImageNet dataset.

Model	ImageNet Accuracy	Million Multi-Adds	Million Parameters
VGG-16	71.5 %	15,300	138
Inception-V3	**81.2 %**	5,000	23.2
MobileNet_V1	70.6 %	569	4.2
MobileNet V2	72.8 %	300	3.4
MobileNet V3 Large	73.3 %	155	**7.5**
MobileNet V3 Small	58.0 %	**21**	**1.56**

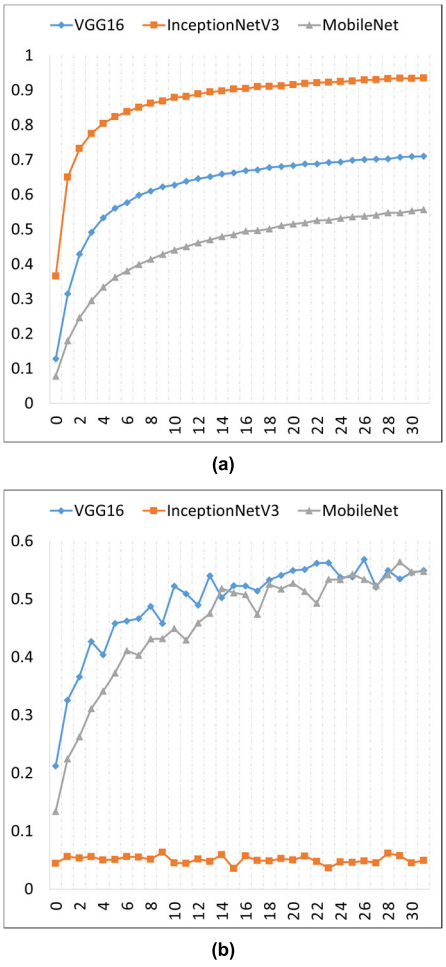


FIGURE 11. Benchmark comparison of (a) training and (b) validation accuracies using different models on Pepper disease recognition dataset.

validation accuracy of 54.76%. It is comparable to the validation accuracy of VGG-19, which is 33 times larger in terms of the number of parameters and has 27 times more multiplication and addition operations. In the light of benchmarking results shown in figure 11, we design a model based on MobileNet_V3_Large to maintain a balance between model complexity and dataset complexity. MobileNet architecture can be seen in figure 12. Such a model may efficiently learn the features in our data and achieve good classification results due to its unique structure and deep architecture while having

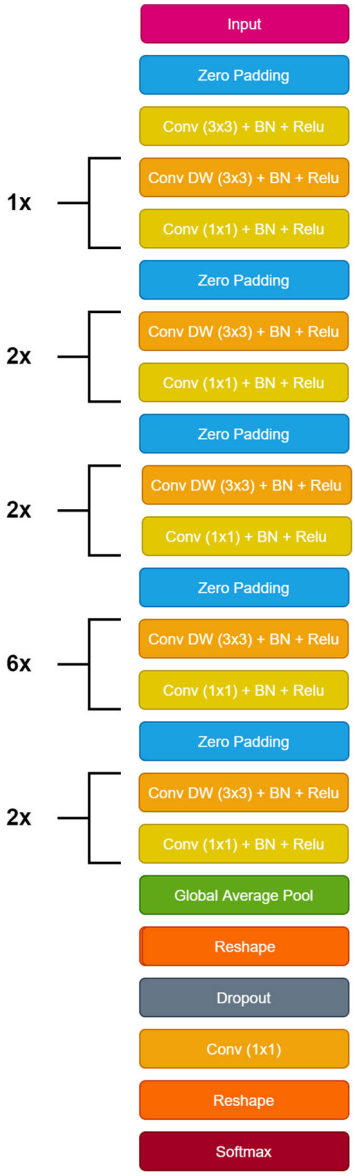


FIGURE 12. Basic MobileNet architecture.

a small number of trainable parameters. Having a smaller number of parameters makes a model less prone to overfitting, which is a major concern while learning representations on small-scale datasets.

5) TRANSFER LEARNING

Transfer learning [29] is a technique used for transferring pre-trained weights from a source domain to a target domain. Source domain is usually a large dataset used for initially training the model. The target domain is the dataset on which it is desired to apply that model. The intuition behind using transfer learning is that the basic features extracted from data are similar even in different domains given the data type is the same across domains. Disregarding the diversity of images from different domains, the basic low-level features extracted at initial layers are essentially similar. Moreover,

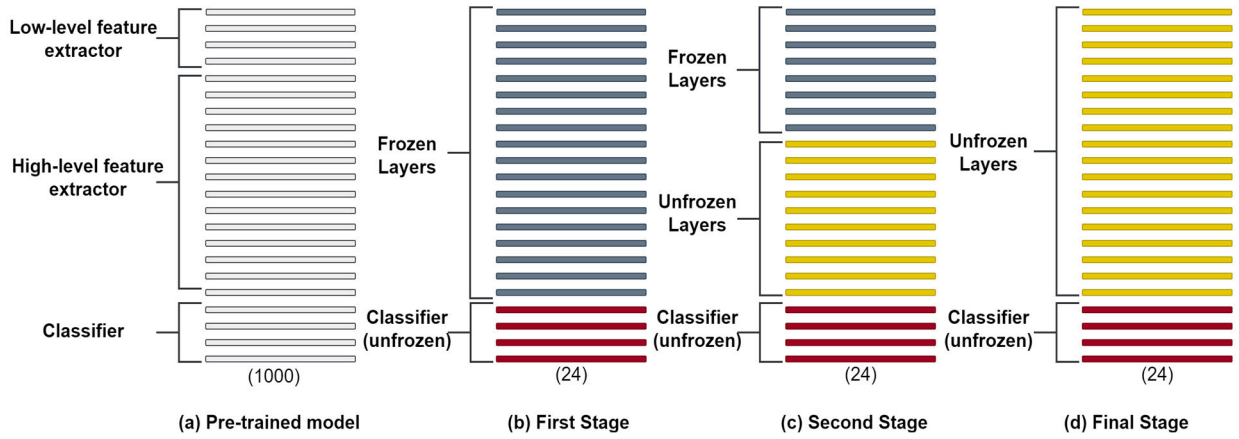


FIGURE 13. A schematic diagram representing the proposed stepwise transfer learning approach. (a) Model trained on source domain i.e., ImageNet. (b) Replaced the Softmax layer to match the target domain's classes and trained only the classifier while keeping the feature extraction layers frozen. (c) Unfroze the high-level feature layers and trained the model to adapt with the target domain. (d) Unfroze rest of the layers and trained for a few epochs for further fine-tuning.

in most cases there are not enough samples available to train a high-complexity model from scratch, hence, it is likely to overfit the training data.

In transfer learning there are three main research issues: 1) What to transfer, 2) How to transfer, and 3) When to transfer [29]. The key to address these issues is based upon the extent to which the source and target domains are similar. If the domain and target classes are very similar, then the complete model can be transferred by replacing the classification head having the number of classes respective to the target domain. However, if the domains are not that similar then a lesser number of layers can be transferred successfully. Hence, the similarity between domains is the key indicator of when and how to transfer. In the case of transferring from ImageNet to Plant disease dataset, the visual similarity in high-level features is very low. The results of only changing the classifier can be seen in benchmark experimentation in section II.B.4. Based on these results, it is crucial to find the right strategy for transfer learning.

6) STEPWISE TRANSFER LEARNING

Brock *et al.* [30] proposed the Freezeout method, where they progressively freeze the layers once a layer's learning rate has reached zero. The main idea behind their approach is to exclude the layers from backpropagation which will result in a faster convergence time. In this study, a similar approach is proposed, however, the intuition behind both methodologies is contrasting. Instead of starting with all the layers set to trainable as in Freezeout, we start training with frozen layers except the classifier layers. Another distinguishing factor is that this study focuses on finding an optimal method for transferring knowledge from the source domain whereas, Freezeout is focused on faster training. In contrast, we propose an automatic algorithmic technique (stepwise transfer learning) which is a very different approach as compared to freezeout as it unfreezes the layers once the loss saturates for a specific number of epochs. Several experiments have shown that stepwise transfer learning has achieved faster training time and accumulative trainable parameters are lower than freezeout. The accumulative trainable parameters can be computed by summing up the trainable parameters at all steps divided by total steps during training. Freezeout focuses mainly on training from scratch. It cannot be applied during transfer learning because it starts training by updating weights for all the layers, which will overwrite all the weights copied from the pre-trained model. However, stepwise transfer learning will preserve the pre-trained weights by freezing the layers from the beginning and only unfreeze when the current model capacity is unable to learn further features.

The proposed transfer learning is divided into three stages. In the first step, weights are transferred from the source domain and are frozen except the classifier is replaced as shown in figure 13 (b). This helps the classifier to learn some mappings from features to labels. In the next step, the loss is computed after every training iteration. If the loss computed has not decreased in the last 10 epochs, another layer block is unfrozen as shown in figure 13 (c). The stepwise transfer learning algorithm is triggered every time the loss is not decreased for the last 10 epochs. Finally, all the layers are unfrozen if they are left frozen due to a continuous decrease in loss as shown in figure 13 (d). This allows for minor fine-tuning towards the end of the training. This was done for the last 5 epochs of training. The complete stepwise transfer learning algorithm is described in table 3.

When the layers are unfrozen prematurely, there is a tendency to overly modifying the weights, especially the low-level features. The preservation of low-level features is important because they form the basis of high-level features. Subsequently, mid-to-high level features need to be more domain-specific as opposed to low-level features. The unfreezing of top layers allows learning the domain-specific features. If the weights from the source are not tweaked, it can

cause negative transfer learning. Negative transfer learning is caused when transfer learning degrades model performance because of a mismatch between underlying features of source and target domains. Finally, in the last few epochs, we drop the learning rate which restricts the gradient to take big steps. In this stage, it is safe to unfreeze all the layers and helps us fine-tune the model for the target domain as shown in figure 13(c). This last step has shown a 1 to 2

III. EXPERIMENTATION AND RESULTS

This section provides details about the experimental setup, statistics of datasets, evaluation metrics used in this study followed by several experiments and their results conducted in this study. Finally, a comparative analysis with other methods is detailed.

A. EXPERIMENTAL SETUP AND DATA STATISTICS

All the experiments were performed using the PyTorch framework with a single NVIDIA 1080 Ti GPU. We used an 80%-10%-10% data split for all our experiments. 80 percent of data is used for training, 10 percent is used for validation, and the remaining 10 percent is used for testing. Table 4 summarizes the statistics of the training, validation, and test splits of the PlantVillage and Pepper Disease dataset. It is to be noted that the experiments performed during initial benchmarking were trained and tested on imbalanced datasets and without tweaking i.e., all the models were used in their original form with original tuning parameters and losses.

Experiments performed in this study can be categorized roughly into three categories for ease of readability. Experiments were conducted on the pepper dataset, experiments conducted on the PlantVillage dataset, and finally transfer-learning experiments. To analyze the performance of the proposed method and comparison with other methods, evaluation strategies are briefly explained in the next section.

B. EVALUATION STRATEGIES AND METRICS

The performance of deep learning-based classification systems is usually assessed based on the model's training and validation accuracy. Training accuracy is the accuracy we get when the model is applied to training data while validation accuracy is the one, we get after applying the model to validation data. Validation accuracy is one of the most commonly used performance metrics. Validation accuracy gives us accuracy on the subset of the dataset which is excluded from the training data and is unseen by the model. Although, this metric is reasonable in many scenarios and very common to use in classification systems. Yet, it fails to evaluate a lot about a model, for example, a model can have very high accuracy yet underperforming on some specific classes, especially in the case of imbalanced datasets. Validation loss is also an important metric to evaluate a model's performance as it accumulates all the losses incurred on each example in the validation set. The problem with validation accuracy is that some underperforming classes might be disregarded, so more metrics should be considered. Another problem In this study precision, recall, and F1-score per class were recorded. Per-class measurement reflects the true performance of a model and can provide insights into the underperforming classes. These metrics will not only reflect the overall performance but will give a more in-depth view of the quality of the model's performance.

C. TRAINING CONFIGURATION

The training configurations employed for the experiments conducted in this study are explained in this section. The training strategy involves several components such as loading pre-trained weights, freezing layers, scheduling learning rate, employing stepwise transfer learning algorithm, and finally validation on unseen data.

The pre-trained weights can be used in various settings to achieve different results. For instance, these weights can be used to jump-start the training process as it eliminates the need for learning all the basic features from scratch. These weights can also be used to extract features, therefore can be used as a feature extractor, where these features are simply used as input for some classifier or as a backbone for other complex applications, for instance, object detection, semantic segmentation, and instance segmentation.

In our case, the pre-trained weights were used for two purposes for the sake of comparisons. Firstly, the weights were used as a feature extractor to generate the results of the baseline. These results were used to evaluate the performance gains achieved from employing different training optimization methods such as freezeout, and our proposed stepwise transfer learning method.

Then the freezing layers is a very common and known procedure, it prevents the gradients for that specific layer to be computed, hence the weights for that particular layer remain unchanged. Different techniques have been proposed to utilize this method for better training performance. The most common usage of freezing layers is when they are used in combination with pre-trained weights loaded during the initialization stage. After the weights are loaded, all the layers are frozen, essentially stopping the model to learn anything new. Then there are more sophisticated approaches proposed which freeze layers to achieve their respective goals. Brock *et al.* [30] proposed a method to progressively freeze the layers as the training continues. They used decaying learning rates which is a common technique for better convergence. As soon as a layer's learning rate approaches zero, that layer's gradients are not computed in further training steps.

On the other hand, in this study the layers are frozen at the initialization stage as in the case of feature extractor, however, as the training loss starts to saturate, a layer is unfrozen, and the training continues until the loss keeps on decreasing. This process is repeated until 60 % of layers remain frozen. This percentage was achieved by running randomized trials. These trials included different configurations. First, the models were allowed to train until all the layers are frozen, second, the training was stopped at different percentages of frozen layers. The best results were achieved at 60% frozen layers.

The Adam optimizer was used to reduce the cross-entropy loss in all the experiments. The starting learning rate was set to 10^{-4}. The learning rate was scheduled to decay per 20 training steps. The gamma or decay rate of 0.1 was used throughout. Experiments were conducted with a step size of 7 but that achieved inferior results as compared to a step size of 20. The learning rate was scheduled for all the experiments using the same scheduling algorithm because the scope of this study is to find the optimal method for training large networks with minimum computational resources in as little time as possible for the rapid development of real-world applications.

Finally, stepwise transfer learning as described in section II.6. was performed on all the models which show superior results in comparison with the conventional transfer learning methods as shown in table 3.

TABLE 3. Pseudocode of stepwise transfer learning algorithm.

```
loss = []
layer_block_index = −1
model = load_model(pretrained = True)
for param in model.parameters():
    param.requires_grad = False
for epoch in range(0, max_epoch):
    train()
    loss.append(train.loss())
    if loss[epoch − 10] − current_loss < threshold:
        layers = model.layers[layer_block_index]
        for param in layers.parameters():
            param.requires_grad = True
            layer_index = layer_index – 1
    if epoch − max _epoch ≤ 5:
        for param in model.parameters():
            param.requires_grad = True
```

D. TRANSFER LEARNING EXPERIMENTS

After initial benchmarking, we selected MobileNet_V3_Large for further experiments due to factors such as less overfitting, a smaller number of multiply-addition operations, and a smaller number of trainable parameters. As discussed in section II.B.4, none of the models performed satisfactorily as shown in figure 11. However, MobileNet_V3_Large seems promising due to less overfitting given the model size and efficiency. Therefore, MobileNet_V3_Large was selected for further experiments with transfer learning (TL) as it is practically inefficient to train a CNN from scratch on small datasets.

As mentioned before, the results presented in table 1 are from 4 training configurations which can be broadly categorized into two i.e., 1) random initialization, and 2) initialization with pre-trained weights from ImageNet. Further, each configuration was coupled with a feature extraction flag which controls whether the gradients for the model will be computed or not. It is interesting to note that the ImageNet weights have little impact on the overall performance when compared against random initialization. It can be concluded that the weights learned on ImageNet are not better than the random initialization for the specific case of plant disease detection due to significant differences across the source and target domains.

To benefit from transfer learning, there should be a high resemblance between both domains, therefore the models trained on the PlantVillage dataset are used as pre-trained weights for training pepper dataset which tends to be comparatively challenging as the benchmarking results on pepper (figure 11) are poor in comparison with PlantVillage dataset (table 1).

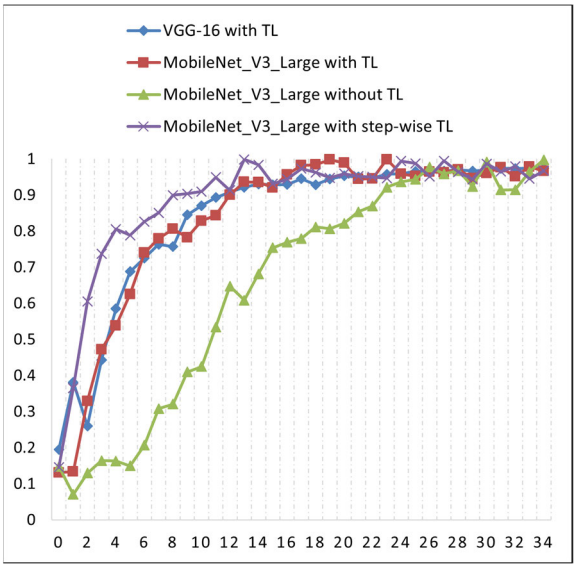
1) STEPWISE TRANSFER LEARNING

First of all, we attempted to use the weights of the ImageNet as a starting point of our training, hence using ImageNet as the source domain and plant disease classification as the target domain in the transfer learning process. To use pre-trained weights, the classification layer of 1000 classes was replaced with the classification layer of 24 classes for the pepper dataset, while keeping the rest of the layers of our model identical to that of a MobileNet_V3_Large model. This was achievable to copy the weights for each layer from the pre-trained MobileNet_V3_Large model.

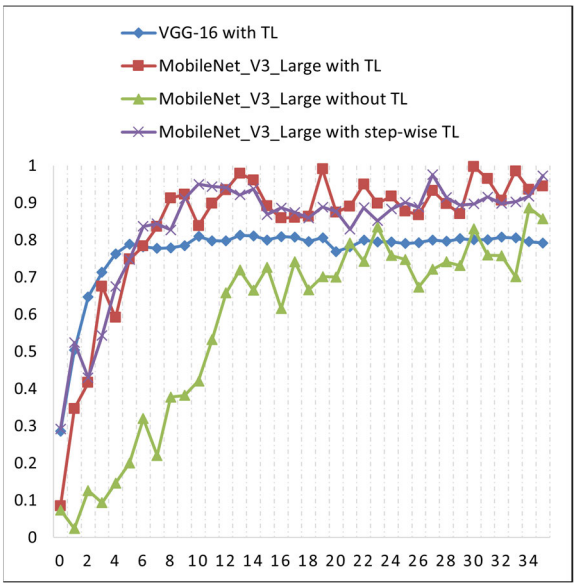
After copying the weights, the layers for which the weights are copied are frozen so that the network can take the advantage of already learned feature representations as used by ImageNet. This step of freezing the layers eliminates the risk of forgetting the learned features. At the beginning of training, the gradient takes big steps which can easily change or overwrite the pre-trained weights. Therefore, it is vital to keep those layers frozen during the initial epochs. This step trains the classification layer to use the feature representation as provided by pre-trained weights for target domain data. ImageNet is a huge dataset, therefore, there is no need to extensively learn low-level basic features again for the plant disease dataset. The pre-trained weights are slightly optimized to classify plants and disease symptoms along with hundreds of other classes.

However, the model can be further fine-tuned, if it discards the unnecessary information about other classes, and use additional parameters to learn domain (plant diseases) related features. To achieve this, the last few layers up until block 12 (onwards last zero-padding layer as shown in figure 12) were unfrozen and trained for a longer period until training accuracy stopped improving. This way model is optimized to learn high-level features that are specific to the target while keeping the low-level features intact.

Apart from MobileNet_V3_Large, other models used in benchmarking were also trained with stepwise TL. This comparison validates the process of stepwise transfer learning as the accuracy of VGG-16 and Inception-V3 with stepwise



(a)



(b)

FIGURE 14. (a) Training and (b)) validation accuracies measured in three experiments with MobileNet_V3_Large (with and without Transfer learning (TL) on Pepper disease dataset and comparison with VGG-16 with TL.

TL also improved significantly. It is discussed in detail in section IV. The results showed the significance of the proposed transfer learning method. Hence, this approach can be used with any type of model.

E. RESULTS

In this section, we present the results of our proposed method on the Pepper disease dataset and PlantVillage dataset. The proposed model was trained using three different configurations. Firstly, the model was trained from scratch i.e., without transfer learning. Then the model was trained with simple transfer learning and finally, the model was trained in stepwise transfer learning configuration.

A slow convergence rate was observed during training without transfer learning configuration compared to other configurations. Surprisingly, VGG-16 with TL showed fast convergence as compared to MobileNet_V3_Large without TL. However, after approximately 30 epochs both the models converged to almost the same accuracies. Then, MobileNet_V3_Large was trained with TL which showed a significant improvement over the previous configuration as shown in figure 14(a). Finally, MobileNet_V3_Large was trained using a third configuration where the weights were transferred in a stepwise manner. As can be seen in figure 14(a), the training accuracy converged surprisingly fast. We argue that the stepwise transfer of pre-trained weights has helped in training the models faster. We hypothesize that due to frozen layers, all the gradients were backpropagated to higher layers which helped in learning plant or disease-specific features in an efficient manner instead of wasting epochs on learning basic features from scratch. Furthermore, the improvement observed from simple TL to stepwise TL is surprisingly significant. To our understanding, this is caused by the unfreezing of the lower layers in later epochs which helps in fine-tuning the basic features. As the network has already learned plant or disease-specific high-level features, it is now less prone to overwriting the low-level features, instead of at this stage, the low-level features are further tweaked according to the specific domain. The models were evaluated using validation accuracy as shown in figure 14(b) and three other evaluation metrics mentioned in section III.B.

Apart from training and validation accuracy, we present results using precision, recall, and F1-score for the pepper disease dataset in table 5.

These results show excellent performance on the dataset. However, these metrics lack the insight, so detailed results are also reported in which all the metrics per disease class are provided as shown in table 5. This can help in enhancing the model performance as individual class performance can be analyzed. It is helpful especially in the case of a class-imbalance problem.

The same experiments were performed on the PlantVillage dataset to evaluate our method on a public dataset. The comparative training accuracies are shown in figure 15(a). The trend is almost identical to that of the pepper disease dataset. The validation accuracies can be seen in figure 15(b). Per class, analysis is also provided as shown in table 8. A comparison of some evaluation metrics is provided in table 6. This shows that the proposed method has dominated the previous methods for the PlantVillage dataset. Further experiments could be done on other public datasets, but unfortunately, there are not enough datasets available to test on public datasets.

IV. DISCUSSION

In this paper, the chosen base architecture is MobileNet_V3_small. Other architectures may also be employed for this purpose. However, our goal is to make this system viable

TABLE 4. Statistics of training and testing data used in this study.

Data source	Training data	Validation data	Testing data	Total classes
PlantVillage	43448	5430	5430	15
Pepper dataset	79621	9943	9943	24

TABLE 5. Comparison of our method with previous methods on the pepper disease dataset.

Methods	Cross-validation accuracy	Recall	F1-score
VGG-16 with TL	79 %	58 %	83 %
1.0-MobileNet w/o TL	77 %	66 %	66 %
MobileNet_V1 with TL	97 %	84 %	83 %
MobileNet_V1 with stepwise transfer learning	99 %	98 %	98 %
MobileNet_V3_Large with stepwise transfer learning	**99%**	**99%**	**99%**

TABLE 6. Comparison of our method with previous methods on PlantVillage dataset.

Methods	Cross-validation accuracy	Recall	F1-score
AlexNet [17]	97.86 %	97.82 %	97.82 %
GoogleNet [17]	98.39 %	98.37 %	98.37 %
VGG-16 with TL	71.40 %	56.10 %	82.43 %
MobileNet_V3_Large w/o TL	73.10 %	62.05 %	63.59 %
MobileNe_V3_Large with TL	98.50 %	98.37 %	97.12 %
MobileNet_V3_Large with stepwise transfer learning	**99.69 %**	**99.40 %**	**99.62 %**

to be used on hand-held devices, its selection compared to other implementations of deep CNN for recognition is the trade-off between accuracy and memory efficiency. For instance, architectures such as VGG and Inception-V3 have a large model capacity, but they tend to overfit small datasets. The motivation behind the selected model is to make this system able to perform well in resource-constrained environments such as hand-held devices so that it can empower the common farmer and help increase the overall crop production.

Furthermore, experiments were conducted to analyze the performance of the proposed stepwise transfer learning (TL) method. We used the two models from the benchmarking stage i.e., VGG-16 and Inception-V3. During simple training of Inception-V3 on our dataset, extreme overfitting was observed, however, the same model was trained in stepwise TL configuration. The model tends to overfit during the

TABLE 7. Per class precision, recall, and F1-score with a total number of samples (support) on the pepper disease dataset using our proposed model.

Disease	Precision	Recall	F1-score	Support
Bacterial spot leaf	1	0.99	0.99	504
Bacterial wilt pulp	1	1	1	482
Bacterial wilt stem	0.97	1	0.98	222
Bitter rot leaf	1	1	1	490
Bitter rot pulp	0.99	1	0.99	461
CMV^{+} leaf	1	1	1	551
Damping off leaf	0.98	1	0.99	486
Damping off stem	1	0.97	0.98	241
Gray mold leaf	0.98	1	0.99	494
Gray mold pulp	1	1	1	480
Gray mold stem	1	0.99	0.99	256
Helicoverpa assulta larva	1	1	1	401
Moth damage leaf	0.82	1	0.9	480
Moth damage pulp	1	1	1	502
Pepper mottle virus leaf	1	1	1	419
Phytophthora blight pulp	1	1	1	493
Phytophthora blight stem	0.99	0.88	0.93	171
Southern stem blight leaf	1	1	1	496
Southern stem blight stem	0.98	1	0.99	210
Spodoptera exigua larva	1	0.73	0.84	386
Spodoptera litura larva	1	1	1	407
TSWV* leaf	0.99	1	1	537
TSWV* pulp	1	1	1	522
TSWV* stem	1	1	1	252
Average/Total	**0.99**	**0.98**	**0.98**	**9943**

* TSWV - (Tomato Spotted Wilt Virus), + CMV - Cucumber mosaic virus.

TABLE 8. Per class precision, recall, and F1-score with a total number of samples (support) on PlantVillage dataset using our proposed model.

Disease	Precision	Recall	F1-score	Support
B Pepper Bacterial spot	1	1	1	99
B Pepper healthy	0.99	1	1	147
P Early blight	1	1	1	100
P healthy	0.99	0.99	0.99	100
P Late blight	0.98	0.99	0.98	15
T Bacterial spot	0.99	0.99	0.99	212
T Early blight	1	0.94	0.97	100
T healthy	0.99	0.99	0.99	190
T Late blight	1	0.91	0.95	95
T Leaf Mold	0.95	1	0.97	177
T Septoria leaf spot	0.99	0.97	0.98	167
T Spider mites	0.95	1	0.97	140
T Target Spot	1	0.99	0.99	320
T mosaic virus	1	1	1	37
T Yellow Leaf Curl Virus	0.99	1	0.99	159
Average/Total	**0.99**	**0.99**	**0.99**	**2058**

B – Bell pepper, P – Potato, T – Tomato

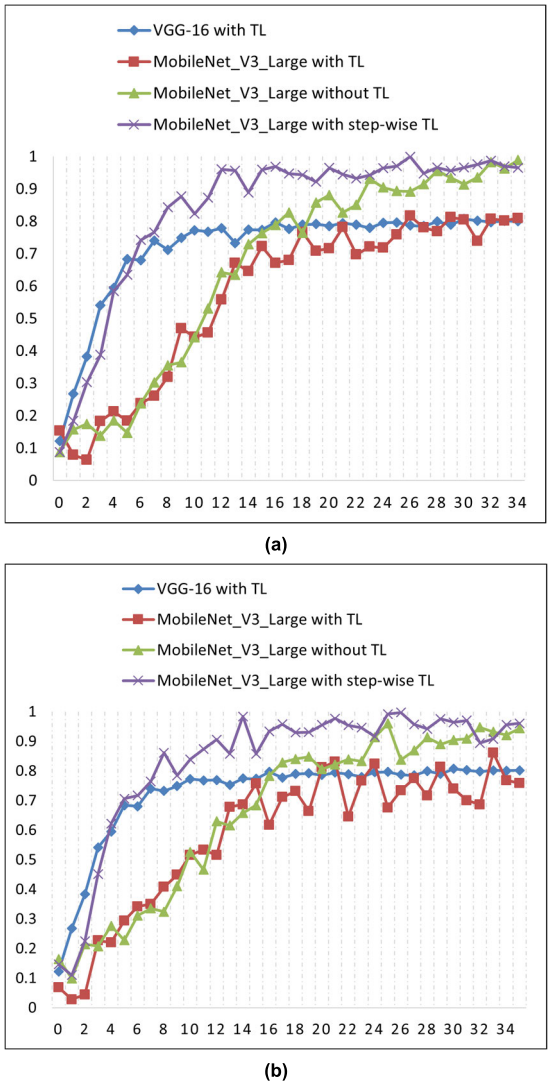


FIGURE 15. (a) Training and (b)) validation accuracies measured in three experiments with MobileNet_V3_Large (with and without Transfer learning (TL) on PlantVillage dataset and comparison with VGG-16 with TL.

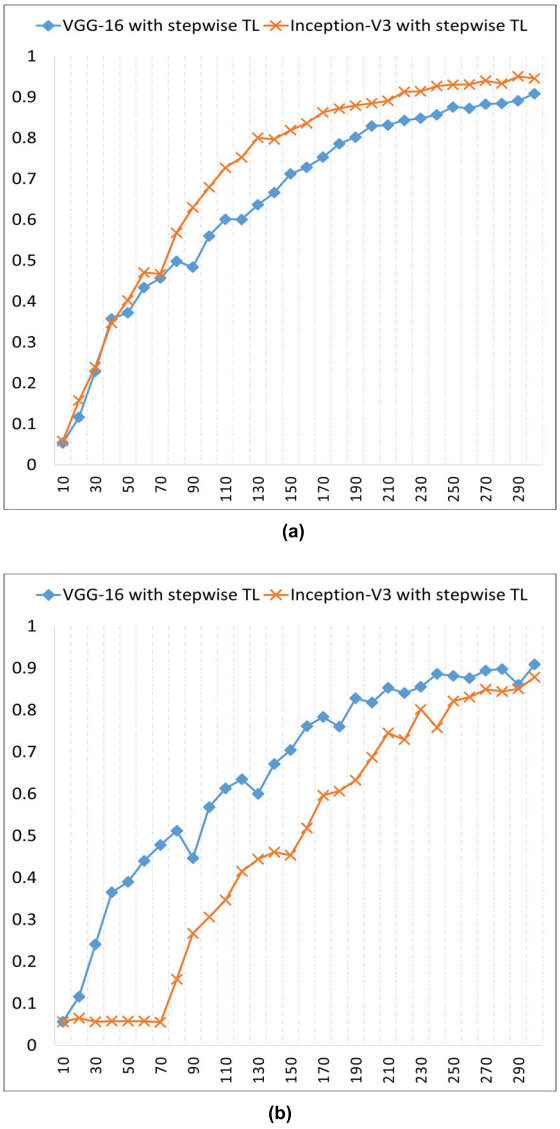


FIGURE 16. (a) Training and (b) Validation accuracies of benchmark models not selected for the next stage. However, when trained with stepwise transfer learning, they have achieved improved validation accuracy and overfitting is reduced to significantly.

initial epochs but if we train long enough after unfreezing the initial layers model starts learning. Training and validation accuracies are provided in figure 16 for both VGG-16 and Inception-V3 trained with stepwise transfer learning. Further comparison is among different transfer learning approaches is provided in table 9. All three methods i.e., Stepwise transfer learning, FreezeOut, and simple transfer learning are also evaluated on a held-out test set as shown in table 9. Moreover, it is important to have an explanation regarding the model's performance. In some scenarios, model accuracy is satisfactory but that might be caused by irrelevant features, for example, in some cases, models are trained on some recurring background features. Therefore, visualization of activations is generated using Grad-cam [31] in figure 17, which has shown that the models are focusing on the right regions. In the horizontal axis, the target category is mentioned, on which the model is conditioned to predict. This results in regions that are activated for that specific target category

Another perspective to understand the intuition behind the success of the proposed method could be that the stepwise transfer learning algorithm simplifies the problem for the gradient descent algorithm. During the initialization, due to frozen layers, there are a reduced number of trainable parameters, which is much easier to optimize if compared with a large number of parameters. As it is commonly known that it is easier to train shallower networks (fewer parameters) as compared to deeper networks (a large number of parameters) due to gradient vanishing and other problems. An analogy can be drawn from this fact to understand the effect of stepwise transfer learning as it is essentially reducing the depth of a

TABLE 9. Comparison of Stepwise transfer learning, Freezeout, and conventional transfer learning with MobileNet_V3_Large model on Pepper dataset.

Method	Training accuracy (%)	5-fold Cross-Validation accuracy (%)	Test accuracy (%)	Trainable Params	Training Time (hours)	Model Complexity (GMACs)	Inference Time
Transfer Learning	91.20	88.24	86.58	4.25 M	3.12	0.23	0.03s
FreezeOut [30]	95.00	92.37	93.26	3.5 M	1.82		
Stepwise Transfer Learning (ours)	**99.00**	**98.99**	**97.84**	2.8 M	1.47		

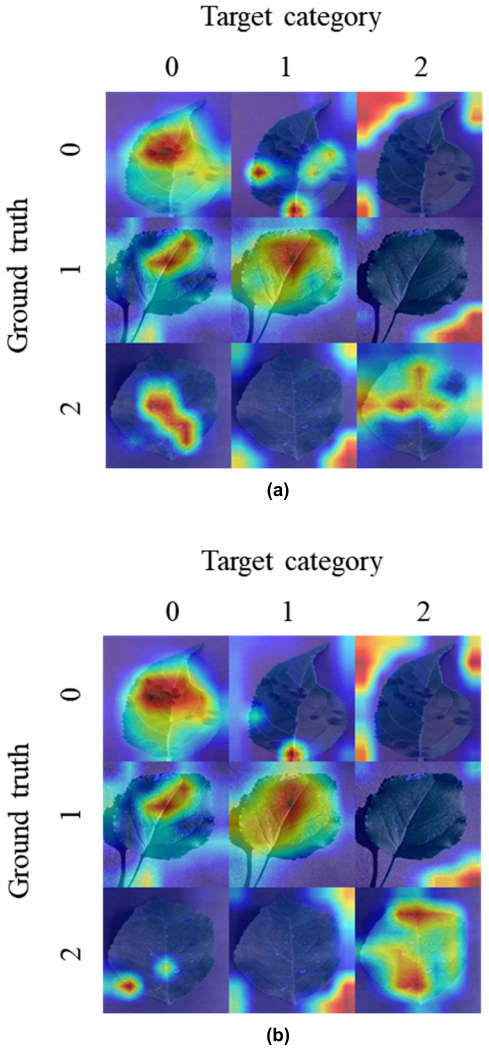


FIGURE 17. Activation maps generated by using Grad-CAM. (a) MobileNet_V3_Large and (b) MobileNet_V3_Large trained with proposed training configuration (stepwise transfer learning). The highest activations were found to be relevant with the ground truth. Where ground truth and target category were mismatched, most of the activations were found in non-leaf regions such as background.

CNN model in the initial iterations and training a very shallow network.

V. CONCLUSION

In this paper, an efficient deep learning-based approach is presented to classify plant diseases in an uncontrolled environment. The dataset is first analyzed for class imbalances which are known to be a deterrent in achieving good classification results. A careful analysis of the dataset has played a key role in improving the overall performance of the system by the class-balancing method. The stepwise transfer learning has helped in reducing the convergence time of CNNs. It has worked not only the proposed model but two other models are also trained with stepwise transfer learning configuration and have shown significant improvement. This is observed that stepwise transfer learning has not only helped in faster convergence but also in reducing negative learning. A plant disease classification system is proposed keeping in view the challenges specific to the plant disease detection problem particularly keeping in view, the feasibility of deploying the classifier on hand-held devices for a practical solution. Such devices still lack high-end hardware so, it is important to design efficient solutions. The proposed classification system is evaluated on the Pepper crop dataset as well as a publicly available dataset PlantVillage dataset. This work can be further extended to other crops and diseases as well as more advanced deep learning techniques can be employed for practical applications. CNN-based computer vision tasks have no doubt achieved a milestone in terms of high accuracies however, there is a need to focus on practical solutions so that industry and consumers both benefit from cutting-edge research. This system if deployed properly can help mitigate losses to small farms and eventually play an important role in increasing crop yield. Finally, this technique can also be used in several industrial applications where rapid development of machine learning algorithms is desired.

REFERENCES

[1] *Food Safety*. Accessed: Dec. 9, 2020. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/food-safety
[2] *Enabling Poor Rural People to Overcome Poverty Smallholders, Food Security, and the Environment*, UNEP, Nairobi, Kenya, 2013.
[3] *Agricultural Biotechnology*, ISAAA, Ithaca, NY, USA, 2015.
[4] S. M. Allard and S. A. Micallef, "The plant microbiome: Diversity, dynamics, and role in food safety," in *Safety and Practice for Organic Food*. Amsterdam, The Netherlands: Elsevier, 2019, pp. 229–257.
[5] V. Singh and A. K. Misra, "Detection of plant leaf diseases using image segmentation and soft computing techniques," *Inf. Process. Agricult.*, vol. 4, pp. 41–49, Mar. 2017.
[6] V. Singh and A. K. Misra, "Detection of unhealthy region of plant leaves using image processing and genetic algorithm," in *Proc. Int. Conf. Adv. Comput. Eng. Appl.*, Mar. 2015, pp. 1028–1032.
[7] S. D. Khirade and A. B. Patil, "Plant disease detection using image processing," in *Proc. Int. Conf. Comput. Commun. Control Autom.*, Feb. 2015, pp. 768–771.

[8] S. S. Sannakki, V. S. Rajpurohit, and V. B. Nargund, "SVM-DSD: SVM based diagnostic system for the detection of pomegranate leaf diseases," in *Advances in Intelligent Systems and Computing*, vol. 174. Chicago, IL, USA: AISC, 2013, pp. 715–720.
[9] S. P. Patil and R. S. Zambre, "Classification of cotton leaf spot disease using support vector machine," *Int. J. Eng. Res.*, vol. 4, no. 3, pp. 1511–1514, 2014.
[10] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.
[11] A. F. Fuentes, S. Yoon, J. Lee, and D. S. Park, "High-performance deep neural network-based tomato plant diseases and pests diagnosis system with refinement filter bank," *Frontiers Plant Sci.*, vol. 9, p. 1162, Aug. 2018.
[12] A. Ramcharan, K. Baranowski, P. McCloskey, B. Ahmed, J. Legg, and D. P. Hughes, "Deep learning for image-based cassava disease detection," *Frontiers Plant Sci.*, vol. 8, p. 1852, Oct. 2017.
[13] A. Ramcharan, P. McCloskey, K. Baranowski, N. Mbilinyi, L. Mrisho, M. Ndalahwa, J. Legg, and D. P. Hughes, "A mobile-based deep learning model for cassava disease diagnosis," *Frontiers Plant Sci.*, vol. 10, p. 272, Mar. 2019.
[14] J. Ahmad, B. Jan, H. Farman, W. Ahmad, and A. Ullah, "Disease detection in plum using convolutional neural network under true field conditions," *Sensors*, vol. 20, no. 19, p. 5569, Sep. 2020.
[15] M. Arsenovic, M. Karanovic, S. Sladojevic, A. Anderla, and D. Stefanovic, "Solving current limitations of deep learning based approaches for plant disease detection," *Symmetry*, vol. 11, no. 7, p. 939, Jul. 2019.
[16] K. P. Ferentinos, "Deep learning models for plant disease detection and diagnosis," *Comput. Electron. Agricult.*, vol. 145, pp. 311–318, Feb. 2018.
[17] S. P. Mohanty, D. P. Hughes, and M. Salathé, "Using deep learning for image-based plant disease detection," *Frontiers Plant Sci.*, vol. 7, p. 1419, Sep. 2016.
[18] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, Inception-ResNet and the impact of residual connections on learning," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017.
[19] A. Krizhevsky, *Learning Multiple Layers of Features From Tiny Images*. Toronto, ON, Canada: Univ. of Toronto, 2009
[20] D. P. Hughes and M. Salathe, "An open access repository of images on plant health to enable the development of mobile disease diagnostics," *arXiv:1511.08060*, 2015. [Online]. Available: https://arxiv.org/abs/1511.08060
[21] G. Wang, Y. Sun, and J. Wang, "Automatic image-based plant disease severity estimation using deep learning," *Comput. Intell. Neurosci.*, vol. 2017, pp. 1–8, Jul. 2017.
[22] J. M. Johnson and T. M. Khoshgoftaar, "Survey on deep learning with class imbalance," *J. Big Data*, vol. 6, p. 27, Dec. 2019.
[23] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, and A. C. Berg, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.
[24] H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," in *Proc. IEEE Int. Joint Conf. Neural Netw. World Congr. Comput. Intell.*, Jun. 2008, pp. 1322–1328.
[25] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, no. 1, pp. 321–357, 2002.
[26] G. Mariani, F. Scheidegger, R. Istrate, C. Bekas, and C. Malossi, "BAGAN: Data augmentation with balancing GAN," 2018, *arXiv:1803.09655*. [Online]. Available: https://arxiv.org/abs/1803.09655
[27] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: https://arxiv.org/abs/1409.1556
[28] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*. [Online]. Available: http://arxiv.org/abs/1704.04861
[29] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.
[30] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, "FreezeOut: Accelerate training by progressively freezing layers," 2017, *arXiv:1706.04983*. [Online]. Available: https://arxiv.org/abs/1706.04983
[31] R. R. Selvaraju *et al.*, "Choose your neuron: Incorporating domain knowledge through neuron-importance," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 526–541.

MOBEEN AHMAD received the B.S. degree in electrical (computer) engineering from COMSATS University, Lahore, Pakistan, in 2013, and the M.S. degree in robotics and intelligent machine engineering from the National University of Science and Technology, Islamabad, Pakistan, in 2016. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, Sejong University, Seoul, South Korea. His research interests include machine learning, AutoML, and computer vision.

MUHAMMAD ABDULLAH received the B.S. degree in electrical (computer) engineering from COMSATS University, Lahore, Pakistan, in 2014. He is currently pursuing the M.S. degree leading to Ph.D. degree with the Department of Computer Science and Engineering, Sejong University, Seoul, South Korea. From 2014 to 2015, he worked as a Business Intelligence Developer at Mobilink Inc., Pakistan. From 2015 to 2016, he worked as a Software Engineering at Mentor Graphics. His research interests include image processing, signal processing, machine learning, AutoML, and computer vision.

HYEONJOON MOON received the B.S. degree in electronics and computer engineering from Korea University, in 1990, and the M.S. and Ph.D. degrees in electrical and computer engineering from the State University of New York at Buffalo, in 1992 and 1999, respectively. From January 1996 to October 1999, he was a Senior Research in electro-optics/infrared image processing branch with the U.S. Army Research Laboratory (ARL), Adelphi, MD, USA. He developed a face recognition system evaluation methodology based on the Face Recognition Technology (FERET) Program. From November 1999 to February 2003, he was a Principal Research Scientist at Viisage Technology, Littleton, MA, USA. He has extensive background on still image and real-time video-based computer vision and pattern recognition. Since March 2004, he has been with the Department of Computer Science and Engineering, Sejong University, where he is currently a Professor and the Chairperson. His current research interests include research and development is on real-time facial recognition system for access control, surveillance, big database applications, image processing, biometrics, artificial intelligence, and machine learning.

DONGIL HAN (Member, IEEE) received the B.S. degree in electronics and computer engineering from Korea University, Seoul, South Korea, in 1988, and the M.S. and Ph.D. degrees in electrical and electronics engineering from the Korea Advanced Institute of Science and Technology, Seoul, in 1990 and 1995, respectively. From 1995 to 2003, he was the Chief Research Engineer with the Digital TV Research and Development Laboratories, LG Electronics Inc., Seoul. He is currently a Professor with the Department of Computer Science and Engineering, Sejong University, Seoul. His research interests include image processing, display quality enhancement for digital TV, system on chip, and robot vision.