

An Efficient Technique for Recognizing Tomato Leaf Disease Based on the Most Effective Deep CNN Hyperparameters

Md. Rajibul Islam^{1,*}, Md. Asif Mahmud Tusher Siddique², Md Amiruzzaman³, M. Abdullah-Al-Wadud⁴, Shah Murtaza Rashid Al Masud⁵ and Alope Kumar Saha⁵

¹Bangladesh University of Business and Technology, Dhaka, Bangladesh

md.rajibul.islam@bubt.edu.bd

²Leeds Beckett University, Leeds, United Kingdom

M.Siddique2525@student.leedsbeckett.ac.uk

³West Chester University, West Chester, Pennsylvania, USA

mamiruzzaman@wcupa.edu

⁴King Saud University, Riyadh, Saudi Arabia

mwadud@ksu.edu.sa

⁵University of Asia Pacific, Dhaka, Bangladesh

murtaza@uap-bd.edu; aloke@uap-bd.edu

*Correspondence: md.rajibul.islam@bubt.edu.bd

Received: 26th March 2022; Accepted: 24th September 2022; Published: 1st January 2023

Abstract: Leaf disease in tomatoes is one of the most common and treacherous diseases. It directly affects the production of tomatoes, resulting in enormous economic loss each year. As a result, studying the detection of tomato leaf diseases is essential. To that aim, this work introduces a novel mechanism for selecting the most effective hyperparameters for improving the detection accuracy of deep CNN. Several cutting-edge CNN algorithms were examined in this study to diagnose tomato leaf diseases. The experiment is divided into three stages to find a full proof technique. A few pre-trained deep convolutional neural networks were first employed to diagnose tomato leaf diseases. The superlative combined model has then experimented with changes in the learning rate, optimizer, and classifier to discover the optimal parameters and minimize overfitting in data training. In this case, 99.31% accuracy was reached in DenseNet 121 using AdaBound Optimizer, 0.01 learning rate, and Softmax classifier. The achieved detection accuracy levels (above 99%) using various learning rates, optimizers, and classifiers were eventually tested using K-fold cross-validation to get a better and dependable detection accuracy. The results indicate that the proposed parameters and technique are efficacious in recognizing tomato leaf disease and can be used fruitfully in identifying other leaf diseases.

Keywords: Convolutional Neural Network; Deep Learning; Disease Recognition; Multi-label Classification; Tomato Leaves

1. Introduction

Biologically called *Solanum Lycopersicon*, tomato is a commonly harvested crop around the world, which is high in principle antioxidants like Vitamin 'C' and 'A' accompanying beta carotene. There is an increasing trend in the production and consumption of tomatoes throughout the globe resulting in 38.54 million tons of production for the year 2020¹. Tomatoes can be grown in any well-drained wet soil with a

¹ "WPTC preliminary 2020 global crop estimate", last modified 2020, <https://www.tomatonews.com/en/wptc-preliminary-2020-global-crop-estimate-2-1175.html>.

sunlight. Tomatoes help improve our immunity and for many diseases, doctors directly prescribe tomatoes to increase antibodies in the body. Given the necessity of this vegetable in both economic and nutritional contexts, it is essential to enhance the cultivation and quality of products using various techniques. However, diseases are a major hindrance to the production of tomatoes. Together with misinterpretation from farmers by over or underuse of pesticides and water, elevated moisture and temperature naturally favour disease development. Here, most of the diseases have visible symptoms and pathologists usually diagnose these diseases through practical observation of leaves. But pathologists require excellent observation skills to detect subtle differences between different diseases. Furthermore, excessive variations, differences because of climates and regions, and new types of diseases make it very difficult to accurately detect the right affliction. Therefore, a precise and real-time disease recognition technology is necessary [1, 2].

In recent years through self-learned mechanisms, convolutional neural networks have progressed significantly in classification and identification tasks. In a variety of applications, CNN has been successfully used, including biomedical photo analysis [3], scene text recognition [4, 5], skin lesions classification [6], License Plate Detection and Recognition [7], and has consistently outperformed the competition. Furthermore, taking into account the universal context data of regions, CNN can extricate more strapping and discriminative features. The result of which is the rapid emergence of many powerful CNN architectures such as DenseNet [8], AlexNet [9], VGGNet [10], and many more. Currently, in applications like plant leaf disease analysis, one of the most common practices is to use deep convolutional networks with a cross-entropy loss function. As the likelihood of plant diseases developing is so low and it is widespread in plant disease datasets, many of them are unbalanced and in the case of unbalanced datasets, these techniques do not perform very well. Furthermore, huge amounts of labelled training data, extensive knowledge, as well as powerful computing and memory resources are required to train a deep neural network from scratch. In addition, problems with overfitting and convergence during deep CNN training typically necessitate repeated changes to the network's design or learning parameters in order to guarantee that all layers are advancing at a similar rate. A viable alternative to starting from scratch is to improve a CNN that has already been trained utilizing a huge collection of natural labeled images.

In this study, we first used various convolutional neural network models namely DenseNet 121 [8], DenseNet 169 [8], ResNet 50 [11], VGG 16 [10], VGG 19 [10], EfficientNet b7 [12] to identify different diseases of tomato leaf and compared their results. After performance analysis, the architecture of network with the highest efficiency among those models was chosen and investigations on the effects of three hyperparameters (i.e. learning rate, optimizer, and classifier) were carried out in terms of accuracy. Based on previous experiments, we realized the network with optimized hyperparameters to talk about how different parameters affect recognition tasks. Lastly, K-fold cross-validation is adopted in order to compare the effectiveness of the networks with improved hyperparameters. To the best of our knowledge, this research is the first to examine and evaluate the relationship between the learning rate, optimizer, and classifier hyperparameters to maximize the effectiveness of deep CNN. This, we believe, makes sense for researchers who want to fine-tune pre-trained algorithms for those other comparable problems.

The remainder of this work is structured as follows: The "Related work" section provides a comprehensive assessment of literary works published in the field of plant leaf disease detection. The "Materials and methods" section shows the datasets, deep neural networks, and tuning hyperparameters required to complete this work. The section "Experiments and Results" offers the method assessment and performance metrics, followed by fivefold cross-validation on the resulting metrics. The "Conclusion" section discusses the work's conclusions and future scope.

2. Related Work

The identification of plant diseases has been debated for many years. A noticeable effect of disease on the leaf is a sign of most plant diseases. With the use of machine learning techniques, several investigators created numerous acceptable structures and contributed their ideas to detect leaf diseases. The color

information that can be recovered from a single-color component is constrained since plant leaf pictures are complicated due to the background. As a result, the feature extraction approach produces less reliable information. Therefore, the high identification accuracy of CNN has attracted many researchers.

Pandian *et al.* [13] applied an innovative 14 layered deep CNN (14-DCNN) on a massive open dataset of leaves. Their research indicates that 14-DCNN is well suited to automated plant disease identification. A customized CNN model significantly outperformed a pre-trained model, as shown by their study. Developing an effective CNN model to get higher detection accuracy is a difficult task. Zhang *et al.* [14] suggested a three-channel CNN model that combines RGB color components to recognize disease in vegetable leaves. Sibiya *et al.* [15] utilized CNN to classify maize plants' diseases. They demonstrated the model's impact using histogram approaches. They were able to obtain an overall model accuracy of 92.85%. For identifying diseases in tomato leaf, Zhang *et al.* [16] investigated a few CNN architectures such as ResNet, AlexNet, and GoogleNet. The maximum accuracy of ResNet was 92.28%, outperforming other networks. In the study presented by Amara *et al.* [17], the LeNet CNN model was utilized to identify banana leaf diseases. Here, the authors test the model using grayscale and color images utilizing the CA and F1- scores.

Ferentinos [18] used AlexNet, GoogleNet, and VGG CNN architecture to compare the classification accuracy of the leaf disease. The VGG surpassed all other networks with the plant, obtaining 99.53 percent disease performance. Yamamoto *et al.* classified tomato diseases using CNN utilizing high, low, and super-resolution to compare super-resolution accuracy to other approaches [19]. The paper's results showed that the super-resolution approach surpassed traditional methods by a great proportion in terms of detection accuracy. Durmus *et al.* [20] used pre-trained networks AlexNet and SqueezeNet V1.1 to classify tomato plant disease. AlexNet, on the other hand, outperforms with a disease classification accuracy of 95.65%.

According to the review, deep neural networks have been effectively employed for learning in plant disease detection applications. The architecture of the network, where it is critical to accurately edge weights and map nodes from the input to the output, is the primary issue involved with developing deep neural networks. To train deep neural networks, it is necessary to fine-tune their network parameters using a procedure that maps the input layer to the output layer and gets better over time. In our work, some pre-trained deep models were used as a starting point and fine-tuned it using three hyperparameters: learning rate, optimizer, and classifier. The capability to use deep models with limited sample numbers is the main benefit of such transfer learning in image classification [21]. Lastly, outstanding values of hyperparameters that contributed the most to improving detection accuracy were recorded using a fivefold cross-validation approach.

3. Materials and Methods

In this study, images of plant leaves were used as input to neural networks. DenseNet 121, DenseNet 169, ResNet 50, VGG 16, VGG 19, and EfficientNet b7 were among the cutting-edge neural networks evaluated. Based on the literature, these models performed well in image classification [8, 10-12].










A computer program can learn from data using deep learning. The learning process is the means through which the ability to conduct the classification with high precision is attained. The aim is to use pre-trained models to identify and classify ten types of plant disease using the ImageNet dataset. The classification job instructs the computer program to determine which of k categories a given input belongs. The learning algorithm is tasked with creating the function $f: \mathbb{R}^n \rightarrow \{1, \dots, k\}$. The model allocates an input defined by a vector x to a category specified by a numerical value y when $y = f(x)$. In this study, ten different classes were used, nine of which were for leaf illnesses and one for healthy leaves.


3.1. Dataset
The dataset of diseased tomato plant leaves was collected from the well-known Plant Village dataset². The dataset contains 56,048 images of plant leaves of 14 different species such as Apple, Blueberry, Cherry, Corn, and Grape. Among these, we chose tomato leaves in this experiment. The dataset of tomato

² "Plant disease detection", last modified 2020, <https://github.com/kevalnagda/plant-disease-detection>.

leaf is composed of images of 9 non-identical classes and 1 healthy class as shown in Table 1 along with a brief information. The 9 diseased classes are Early blight, Bacterial spot, Leaf mold, Late blight, Septoria leaf spot, Target spot, Two-spotted spider mite, Tomato yellow leaf curl virus, and Tomato mosaic virus. There are 18,160 images in total, and 1591 of them are images of healthy tomato leaves.

Table 1. Tomato leaf disease dataset

Label	Category	No. of samples	Symptoms	Illustration
1	Bacterial Spot	2127	Spots are dark brown to black; spots rarely develop to more than 3mm in diameter.	
2	Early Blight	1000	There are black or brown dots, and leaf spots frequently have a yellow or green concentric ring pattern.	
3	Healthy	1591	Bright green color and grown leaves are generally 10 inches in length, new leaves can be 3 inches in length.	
4	Late Blight	1909	The water-soaked region emerges and quickly expands to produce purple-brown, oily-looking patches.	
5	Leaf mold	952	Pale greenish-yellow spots, normally smaller than .25 inches. Olive-green to brown velvety mold on the lower part.	
6	Septoria leaf spot	1771	Spots are circular, marginal brown, chlorotic yellow. 1/16 to 1/4 in diameter with a dark brown margin.	
7	Two Spotted Spider Mites	1676	Numerous yellow or white tiny, granulated spots, blade black netting.	
8	Target Spot	1404	Small necrotic lesions with light brown cores and black borders grow from spots.	
9	Tomato Mosaic virus	373	On the leaves, the mittens are light and dark green. Curled, deformed, or swollen leaves are possible.	

10	Yellow leaf curl virus	5357	Stunting, decrease in leaf size, upward cupping or curling of leaves, and chlorosis on leaves and flowers are all symptoms of severe stunting.	
----	------------------------	------	--	---

In the first and second part of the experiment, the dataset was split into training and testing datasets in an 8:2 ratio by randomizing pictures from the dataset based on the group label ratio. In the third part of the research, we did five-fold cross-validation. For that, the whole dataset was equally divided into five folders, where one of the folders was used for validation and the other four for training. In all cases, the images have all been downsized to the target size (64×64). The dataset was normalized before being divided into training and validation sets.

3.2. State-of-the-art CNN Architectures Analysis

In the first part of the research, we evaluated six well-known pre-trained convolutional neural network architectures to detect and classify disease in tomato leaves. The models are DenseNet 121 [8], DenseNet 169 [8], ResNet 50 [11], VGG 16 [10], VGG 19 [10], EfficientNet b7 [12]. The following is a quick summary of these pre-trained models:

3.2.1. DenseNet

DenseNet was first introduced in the paper [8]. In a feed-forward manner, it connects each layer to any other layer. This network has $L(L+1)$ direct connections between each layer and its following layer, whereas most conventional convolutional networks have just one link in between layer and its following layer. DenseNet design provides several advantages, including improving feature propagation, relieving the vanishing gradient problem, and, most importantly, lowering the parameter count.

3.2.2. ResNet

In the paper [11] ResNet was first introduced. This architecture was proposed primarily to solve the problem of numerous non-linear layers not being able to learn identity mapping and to address the degradation problem. There are three types of layers in the ResNet model, and they are 50, 101, and 152. Among those, ResNet50 is the most efficient and effective.

Thus, in this experiment, we choose ResNet50. This is a network-within-a-network design built on a large number of stacked residual units. Residual units serve as the foundation of the ResNet design. Convolution and pooling layers make up these residual units. This is kind of similar to the VGG [10] architecture but 8 times deeper. In this experiment, we loaded the pre-trained network and finally added a softmax layer in the end to perform image classification.

3.2.3. VGG

VGG [10], developed by the University of Oxford's Visual Geometry Group, placed second in the classification assignment at the ILSVRC-2014. The most astonishing feature of this architecture is that it consistently has the same convolution layer that uses 3×3 filters. We employed two of the best performing VGG architectures, VGG 16 and VGG 19, in this experiment. VGG-16 contains 13 convolution layers followed by 3 completely connected layers, whilst VGG-19 has a stack of 19 convolutional layers linked to a fully connected layer. In this case, we loaded pre-trained VGG-16 and VGG-19 weights and created an output layer with ten dimensions, which correspond to the ten tomato disease classes.

3.2.4. EfficientNet

EfficientNet [12] was introduced first to achieve more effective performance by evenly scaling width, depth, and resolution parameters utilizing a remarkably effective composite coefficient while scaling down the model. Unlike other CNN models, which employ ReLU as the activation function, this one proposes a unique activation function called Switch. EfficientNet has eight models ranging from B0 to B7. When the number of models increases the accuracy increases considerably while the quantity of estimated parameters does not increase that much. In this experiment, we have used the latest one, EfficientNet B7.

The inverted bottleneck MBConc is the primary building block of the EfficientNet. Under similar FLOPS constraints, EfficientNet performs much better than most other neural network models by giving significantly better accuracy numbers. Here, we used the native model architecture to extract features for the output FC layer.

3.3. Hyperparameters Tuning

Hyperparameters are a set of parameters that can influence the model's learning. These parameters include the number of epochs, layers, activation functions, optimizers, learning rate, etc. The hyperparameter configuration utilized in the second half of the investigation is detailed below. After multiple tries, the authors advanced the effective learning rate, optimizer, and the activation function to the third stage of the experiment, which is the K-fold cross-validation procedure.

3.3.1. Learning rate

A hyperparameter, the main purpose of which is to change the model concerning the approximated error every time the weights of the model are updated is known as the Learning rate. Determining a fixed value for the learning rate is strenuous. Selecting a very tiny value might lead to a lengthy training process and even can get stuck. On the other hand, choosing a too big value might lead to an unstable training process or too fast learning of a sub-optimal set of weights. While configuring a neural network it might be one of the most important hyperparameters. To combat the problem of choosing a hyperparameter manually for each given learning session in the learning rate schedule, there are various adaptive gradient descent algorithms, including Adadelta, Adam, and RMSpro. But as we have seen from this experiment, choosing a suitable learning rate is important even for those adaptive learning rates, especially while working with fewer epochs [22, 23].

3.3.2. Optimizer

Optimizer plays an important role while iteratively updating the parameters of all the layers in the training of the deep CNN model [24, 25]. Optimization is quite important in training a neural network, as is responsible for reducing losses and providing the most accurate results. Gradient descent is a prominent approach for doing optimization in a neural network. This is used frequently in linear regression and classification algorithms. Moreover, the gradient descent algorithm is responsible for backpropagation in neural networks. Even though it is easy to implement and compute, it has a few drawbacks such as may often trap in local minima and requiring large memory to calculate the gradient descent of the whole dataset. In this article, we have worked with stochastic gradient descent, Adam, AdaBound, RMSProp, AdaDelta, AdaGrad, Nadam, and Ftrl to see their effect on our dataset.

3.3.3. Activation Functions

The main work of an Activation function or classifier is to sort data into labeled classes or categories. It mainly affects the outcome of deep learning models, including their performance and accuracy. The activation functions have a significant influence on the capacity and speed of neural networks to converge [26-28]. Moreover, activation functions help to normalize the output between -1 and 1 for any input. As weight and bias are essentially linear transformations, a neural network is simply a linear regression model with no activation function. Activation functions are available in a variety of forms, including Binary step, Linear, ReLU, Sigmoid, and many more. In the second part of the experiment, we experimented on Softmax, ReLU, SeLU, ELU, Exponential, Nadam, Softsign, Tanh, and Sigmoid.

3.4. K-fold Cross-Validation

K-fold cross-validation is a statistical method for measuring the ability of machine learning models. In the third part of experiment, the highest values of the hyperparameters acquired in the second part of the experiment were assessed using 5-fold cross-validation. This process aims to analyze the performance and relationship of these hyperparameters in enhancing classification accuracy.

4. Experiments and Results

4.1. Algorithms Evaluation

The CNN models considered in this study were executed in a machine equipped with Ryzen 3600x processor, AMD radeon RX 550, and 16 GB RAM. All codes were realized with keras 2.4.3 framework, written in python 3.9.5, and executed in Jupyter Notebook. For every experiment, we used categorical cross-entropy loss and accuracy metrics for evaluation. A similar layout was taken for every model and each experiment was run for 50 epochs. A dense layer "Softmax" activation function was employed for classification at the output. "Adam" was the optimizer we used with a learning rate of 0.01. The accuracy and loss of training and validation datasets are shown in Table 2. Furthermore, recall, precision, and F1 score are also shown are of weighted average. The average time (in seconds) taken for each epoch is also shown in Table-2.

Table 2. Results analysis of each pre-trained models

Model	Training set		Validation Set		Precision	Recall	F1 score	Average Time per epoch (Seconds)
	Accuracy	Loss	Accuracy	Loss				
DenseNet 121	0.9955	0.0137	0.9912	0.0460	0.991279	0.991189	0.991194	410
DenseNet 169	0.9920	0.0256	0.9471	0.1989	0.954496	0.947137	0.945015	495
ResNet 50	0.9892	0.0341	0.9876	0.0411	0.987682	0.987610	0.987604	570
VGG 16	0.9752	0.0758	0.9763	0.0904	0.978826	0.978524	0.978428	600
VGG 19	0.9840	0.0520	0.9788	0.0658	0.979296	0.978800	0.978823	575
EfficientNet B7	0.9667	0.0993	0.9628	0.1163	0.964157	0.962830	0.962865	1380

In the case of DenseNet 121, after 50 epochs we achieved an accuracy score of 99.55% in the training set and 99.12% in the validation set. The weighted average of recall, precision, and F1 score were 0.9912, 0.9911, 0.9911 consecutively. The average time it took for each epoch to complete was 410 seconds. DenseNet 169 performed almost similarly to DenseNet 121. Even though this architecture has more layers it performed worse with it. As a result, the average time of execution of each epoch increased to 495 seconds. After 50 epochs it achieved an accuracy score of 94.71% and loss was 19.89%. ResNet 50 has the closest results to the DenseNet 121. Its accuracy in both the train and validation set was almost the same, near 98%. In the case of the training set after 50 epochs, it achieved an accuracy of 98.92% and in the validation set, it achieved 98.76%. Both VGG 16 and VGG 19 performed similarly on the basis of the accuracy of the validation set which was close to 97%. Their average time per epoch was also almost adjacent. The weighted results of precision, sensitivity, and F1 score are shown in Table 3 for each type of diseases.

Table 3. Test results of algorithms for each type of disease

Model	Score	Yellow Leaf Curl Virus	Mosaic virus	Target Spot	Two-spotted spider mite	Septoria leaf spot	Leaf Mold	Late blight	Healthy	Early blight	Bacterial spot
DenseNet 121	Precision	0.9990	1	0.9704	0.9872	0.9774	1	1	0.9853	0.9829	0.9976
	Recall	0.9990	1	0.9820	0.9747	1	1	0.9817	0.9970	0.9942	0.9816
	F1 Score	0.9990	1	0.9761	0.9809	0.9886	1	0.9908	0.9911	0.9885	0.9896
DenseNet 169	Precision	0.9894	1	0.9804	0.7665	0.9883	0.9597	0.9891	0.9626	0.7857	0.9817
	Recall	0.9961	0.9736	0.6017	0.9842	0.9740	0.9896	0.9505	0.9940	0.9482	0.9839
	F1 Score	0.9927	0.9866	0.7458	0.8618	0.9811	0.9744	0.9694	0.9781	0.8593	0.9828
ResNet 50	Precision	0.9980	0.9743	0.9696	0.9809	0.9942	0.9948	0.9921	0.9627	0.9767	0.9953
	Recall	0.9970	1	0.9580	0.9747	1	0.9948	0.9843	0.9970	0.9655	0.9862
	F1 Score	0.9975	0.9870	0.9638	0.9778	0.9971	0.9948	0.9882	0.9795	0.9710	0.9908
VGG 16	Precision	0.9932	0.9866	0.9932	0.9031	0.9939	0.9597	0.9816	0.9940	0.9367	0.9730
	Recall	1	0.9736	0.8862	1	0.9452	0.9896	0.9765	0.9970	0.9367	0.9908
	F1 Score	0.9966	0.9801	0.9367	0.9491	0.9689	0.9744	0.9791	0.9955	0.9367	0.9818

VGG 19	Precision	0.9951	0.9743	0.9905	0.9491	0.9824	0.9648	0.9810	0.9940	0.8864	0.9838
	Recall	1	1	0.9401	1	0.9682	0.9948	0.9427	0.9940	0.9425	0.9748
	F1 Score	0.9975	0.9870	0.9646	0.9738	0.9753	0.9795	0.9614	0.9940	0.9136	0.9793
EfficientNet B7	Precision	0.9990	0.95	0.9171	0.9656	0.9505	0.9895	0.9865	0.8842	0.8950	0.9881
	Recall	0.9922	1	0.8952	0.8864	0.9971	0.9792	0.9531	0.9970	0.9310	0.9542
	F1 Score	0.9956	0.9743	0.9060	0.9243	0.9732	0.9843	0.9695	0.9372	0.9126	0.9708

EfficientNet B7 performed most poorly in terms of average time per epoch. Whereas other algorithms took less than 600 seconds to complete each epoch, EfficientNet took more than double time, around 1400 seconds to finish each epoch. Moreover, its accuracy score in the validation set was the second lowest of the bunch. Similar trends can be seen in training set accuracy, loss, precision, recall, F1 score.

The graph below (Figure 1) depicts the accuracy and loss of the models on classifying the tomato leaf diseases.

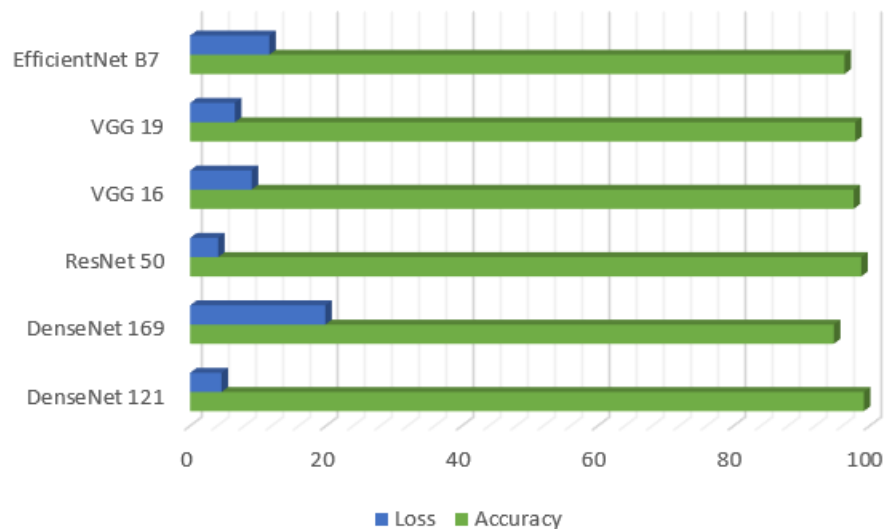


Figure 1. Accuracy and loss of the tested models

4.2. Performance Metrics Evaluation over Hyperparameters

From the above result analysis, we can see that DenseNet 121 surpassed other pre-trained models for the Tomato leaf disease diagnosis. To do further analysis, we tried tweaking different parameters and tried to find out if learning rate, optimizer, or activation functions had an impact on the overall effectiveness of the DenseNet architecture as depicted in table 4. If so, what are the optimal metrics for learning rate, optimizer, and classifier hyperparameters to use for the DenseNet 121 model? For that first started by changing the learning rate. We started with a 0.002 learning rate and kept gradually increasing to 0.0009. As the results were getting worse, we stopped there and then kept gradually decreasing the learning rate. Then we selected the learning rate at which the pre-trained model performed best. Then we tried other popular optimizers out there and analyzed the results. Finally, we selected the optimizer that performed best among those and tried different classifiers. Results of all these are given below Table 4.

Table 4. Results analysis of different metrics for Learning rate, Optimizer, and Classifier

Model Name	Learning Rate	Optimizer	Classifier	Accuracy	Loss	Precision	Recall	F1 Score
DenseNet 121	0.002	Adam	Softmax	0.9725	0.0894	0.972592	0.972467	0.972244
DenseNet 121	0.005	Adam	Softmax	0.9791	0.0711	0.979726	0.979075	0.979143
DenseNet 121	0.0001	Adam	Softmax	0.891	0.3416	0.9036	0.890969	0.892355
DenseNet 121	0.0005	Adam	Softmax	0.8802	0.4666	0.901473	0.880231	0.880055
DenseNet 121	0.0009	Adam	Softmax	0.8588	0.5263	0.888472	0.858756	0.856682
DenseNet 121	0.001	Adam	Softmax	0.9631	0.1129	0.943885	0.936399	0.936122

DenseNet 121	0.05	Adam	Softmax	0.9893	0.0432	0.99506	0.995044	0.995036
DenseNet 121	0.01	Adam	Softmax	0.9912	0.046	0.991279	0.991189	0.991194
DenseNet 121	0.5	Adam	Softmax	0.9579	0.1449	0.958487	0.957874	0.957903
DenseNet 121	0.1	Adam	Softmax	0.9876	0.0454	0.98764	0.98761	0.987561
DenseNet 121	1	Adam	Softmax	0.2817	39.8967	0.115794	0.281663	0.130671
DenseNet 121	0.01	AdaBound	Softmax	0.9931	0.0395	0.99506	0.995044	0.995036
DenseNet 121	0.01	SGD	Softmax	0.9664	0.114	0.968755	0.967236	0.967159
DenseNet 121	0.01	RMSProp	Softmax	0.9902	0.0398	0.990472	0.990363	0.990374
DenseNet 121	0.01	AdaDelta	Softmax	0.9788	0.0665	0.979477	0.9788	0.978811
DenseNet 121	0.01	AdaGrad	Softmax	0.9675	0.1095	0.968946	0.967511	0.967583
DenseNet 121	0.01	Nadam	Softmax	0.9414	0.2211	0.951478	0.941355	0.942128
DenseNet 121	0.01	Ftrl	Softmax	0.2844	2.1597	0.080893	0.284416	0.12596
DenseNet 121	0.01	AdaMax	Softmax	0.9771	0.0831	0.977674	0.977148	0.977208
DenseNet 121	0.01	AdaBound	Softplus	0.9904	0.0374	0.990562	0.990363	0.990369
DenseNet 121	0.01	AdaBound	Selu	0.3695	1.86	0.240685	0.369493	0.214595
DenseNet 121	0.01	AdaBound	Relu	0.2844	Nan	0.080893	0.284416	0.12596
DenseNet 121	0.01	AdaBound	Elu	0.3312	2.1844	0.202258	0.331222	0.229642
DenseNet 121	0.01	AdaBound	Exponential	0.2844	Nan	0.080893	0.284416	0.12596
DenseNet 121	0.01	AdaBound	Nadam	0.9413	0.2211	0.951478	0.941355	0.942128
DenseNet 121	0.01	AdaBound	Softsign	0.2901	8.7462	0.120583	0.105796	0.055687
DenseNet 121	0.01	AdaBound	Tanh	0.0985	8.7652	0.018213	0.098568	0.027964
DenseNet 121	0.01	AdaBound	Sigmoid	0.9625	0.1655	0.964853	0.961729	0.961471

As we can see from Table 4 that for learning rate there is a range or a fixed point for which the algorithm performs well above which the accuracy decreases, and below which accuracy also drops. In our experiment, we observed the worst results when the learning rate was increased to 1. Here, accuracy dropped below 29%, and the F1 score was just 13%. In this study for a learning rate of 0.01, the algorithm performed best. Accuracy, in this case, was just above 99%, loss observed was 0.046, and F1 score was also above 99% mark. In the case of optimizers, seven out of nine algorithms scored more than 95%. Among them, Adabound's accuracy score was the highest. It had an accuracy score of 99.31% which was just above Adam's 99.12%. Its loss was also less than Adam's. Its precision, recall, and F1 score was 0.99506. RMSProp also performed well here, the accuracy score of which was 99.04%. Among all the classifiers tested Ftrl optimizer had the worst performance, with accuracy just above 28% the F1 score was just 0.1259. After selecting 0.01 as the learning rate and Adabound as the optimizer we tested on different activation functions. Here, in total four activation functions scored more than 90%, Softmax, Softplus, Nadam, and Sigmoid. Among them, the score of softmax was the highest. Tanh scored least in terms of accuracy with just 9.85%. So, overall, we found optimum results when the learning rate is 0.01, Optimizer is AdaBound, and activation function is Softmax.

The Figure 2 below illustrates the confusion matrix of the results when the learning rate is 0.01 and the optimizer is AdaBound. Here, in the vertical line is the actual label of the images of each disease and the horizontal line is the model predicted classification of the images:

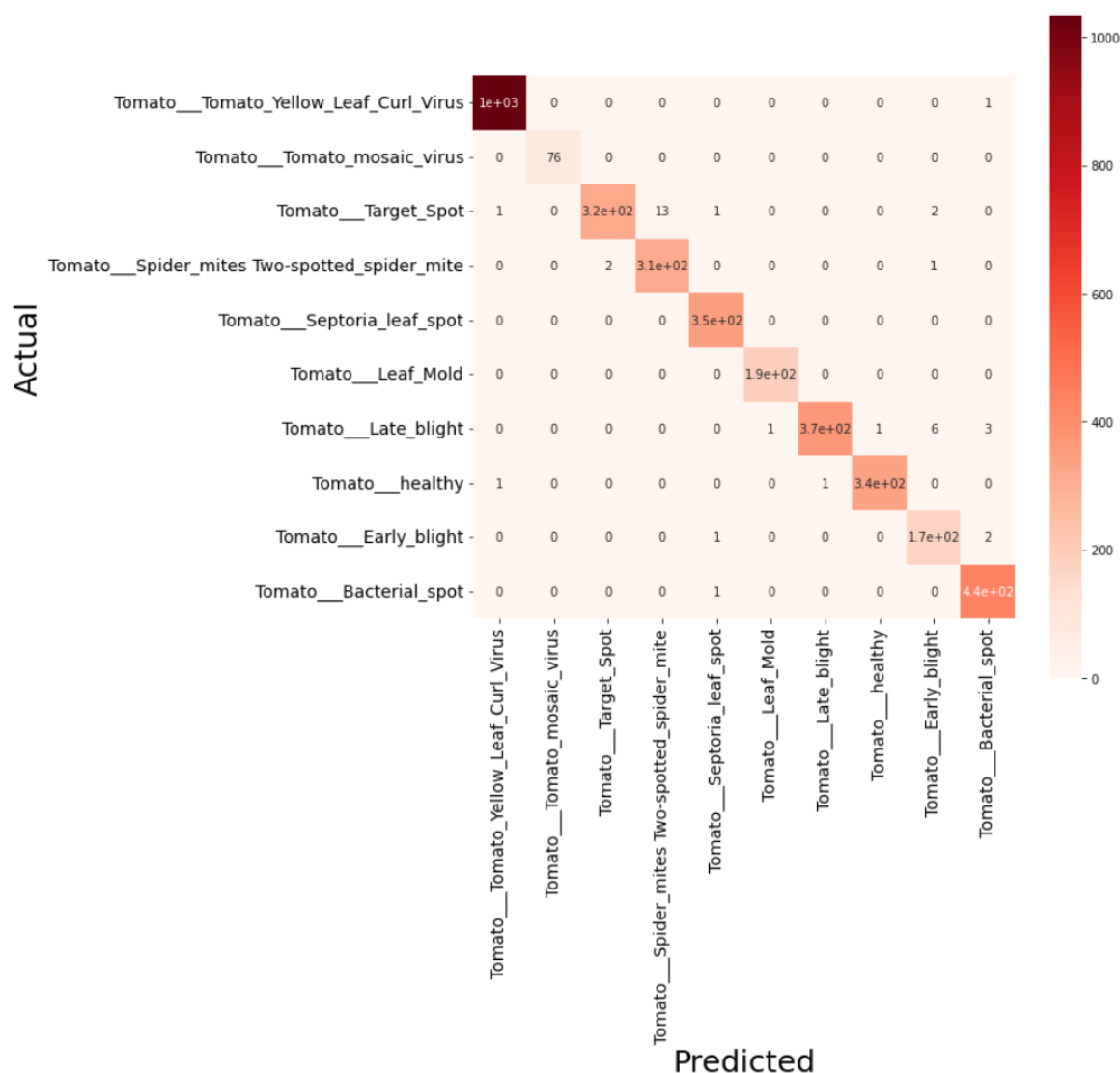


Figure 2. Confusion matrix when the optimizer is Adabound

4.3. K-fold Cross Validation

Table 4 shows 3 potential combinations of hyperparameters that produced the maximum accuracy, or 99%, in this case. Those are a combination of, (i) AdaBound optimizer and Softmax classifier with a learning rate of 0.01, (ii) Adam optimizer and Softmax classifier with a learning rate of 0.01, and (iii) AdaBound optimizer and Softplus classifier with a learning rate of 0.01. To check the authenticity of these results we further did five-fold cross validation on the dataset using hyperparameters that exhibited the highest performance metrics scores according to Table 4. The end result is shown in Table 5 below:

Table 5. Performance metrics scores over hyperparameters using Five-fold cross validation

Model Name	Fold	Learning rate	Optimizer	Classifier	Accuracy	Loss	Precision	Recall	F1 Score
DenseNet 121	1	0.01	Adabound	Softmax	0.9929	0.0173	0.992952	0.992885	0.992895
DenseNet 121	2	0.01	Adabound	Softmax	0.9939	0.0226	0.99392	0.993889	0.993885
DenseNet 121	3	0.01	Adabound	Softmax	0.9882	0.0392	0.988344	0.988154	0.988177
DenseNet 121	4	0.01	Adabound	Softmax	0.9937	0.0168	0.993759	0.993673	0.993683
DenseNet 121	5	0.01	Adabound	Softmax	0.9734	0.4688	0.966071	0.965385	0.965499
DenseNet 121	1	0.001	Adabound	Softmax	0.9674	0.1002	0.969183	0.967433	0.967716
DenseNet 121	2	0.001	Adabound	Softmax	0.9592	0.1571	0.959167	0.959167	0.959167

DenseNet 121	3	0.001	Adabound	Softmax	0.7573	1.0255	0.867438	0.7573	0.768661
DenseNet 121	4	0.001	Adabound	Softmax	0.8366	0.7323	0.889585	0.836589	0.836541
DenseNet 121	5	0.001	Adabound	Softmax	0.8703	0.5724	0.917959	0.879121	0.881876
DenseNet 121	1	0.1	Adabound	Softmax	0.5446	1.2905	0.424256	0.544609	0.470178
DenseNet 121	2	0.1	Adabound	Softmax	0.2972	nan	0.088341	0.297222	0.1362
DenseNet 121	3	0.1	Adabound	Softmax	0.3237	647538	0.348753	0.323691	0.28699
DenseNet 121	4	0.1	Adabound	Softmax	0.2944	nan	0.086648	0.29436	0.133885
DenseNet 121	5	0.1	Adabound	Softmax	0.2947	nan	0.08641	0.293956	0.13356
DenseNet 121	1	0.01	Adam	Softmax	0.9871	0.0668	0.936839	0.987658	0.982842
DenseNet 121	2	0.01	Adam	Softmax	0.9735	0.0975	0.883684	0.965478	0.959866
DenseNet 121	3	0.01	Adam	Softmax	0.9812	0.0728	0.926514	0.981212	0.981432
DenseNet 121	4	0.01	Adam	Softmax	0.9564	0.6998	0.912693	0.956584	0.956823
DenseNet 121	5	0.01	Adam	Softmax	0.9234	0.7249	0.894126	0.914677	0.913799
DenseNet 121	1	0.01	AdaBound	Softplus	0.9669	0.2168	0.903615	0.965743	0.963426
DenseNet 121	2	0.01	AdaBound	Softplus	0.9548	0.0878	0.899624	0.956767	0.957685
DenseNet 121	3	0.01	AdaBound	Softplus	0.9856	0.1746	0.932275	0.988796	0.983435
DenseNet 121	4	0.01	AdaBound	Softplus	0.9233	0.9987	0.862627	0.912374	0.918768
DenseNet 121	5	0.01	AdaBound	Softplus	0.8963	1.4367	0.889144	0.875587	0.885743

Here, we can see that for our metrics in the case of all 5 folds the accuracy score was more than 97%, it got more than 99% accuracy in three out of five folds for learning rate 0.01, AdaBound optimizer, and softmax classifier. However, the accuracy score reached as high as 99.39% in the second fold. When we applied the same experiment for 0.001 and 0.1 learning rates, the results were much worse. Especially for the learning rate of 0.1, the accuracy was below 60%, and in three out of five cases; it was even below 30%. The other two combinations of Adam optimizer and Softmax classifier with a learning rate of 0.01, and AdaBound optimizer and Softplus classifier with a learning rate of 0.01 have not score more than 99% accuracy. Moreover, some folds of these combinations even score close to 92% accuracy. Therefore, we found that for learning rate 0.01, AdaBound optimizer, and softmax classifier the model performs best.

The Figure 3 below depicts the model accuracy and model loss of each epoch while the model was learning from the dataset for learning rate 0.01, AdaBound optimizer, and Softmax classifier. As we can see from the 2nd diagram, the model loss did not change a lot after the 7th or 8th epoch and stayed almost the same as the train set loss. In the case of the accuracy, it fluctuated a lot before it stabilized at the 35th or 36th epoch, then it was almost as same as the train set accuracy.

So, we can see that indeed for the learning rate 0.01, Softmax classifier and AdaBound optimizer the DenseNet performs best.

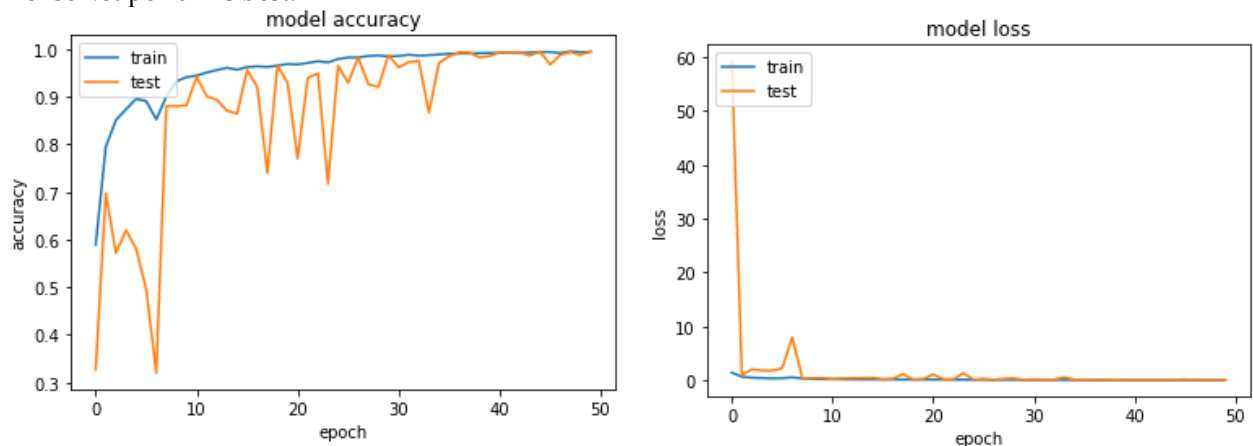


Figure 3: Model Accuracy and Model loss in cross validation

5. Conclusions

This paper analyzed networks that are based on pre-trained deep convolutional networks of DenseNet, ResNet, EfficientNet, and VGG. Here, in the first step, we compared those networks with Adam Optimizer, 0.01 learning rate, and Softmax Activation function. The highest result was achieved with DenseNet. Then a performance evaluation was done with different optimizers, learning rates, and classifiers that was affecting the results of the DenseNet. We found out that a range of learning rates between 0.001 and 0.1 gives good results where above and below are not effective. In the case of activation functions with Softmax and Softplus activation functions, the best results were obtained. When different optimizers were evaluated Adam, AdaBound, and RMSProp performed well. Here, the best overall result was observed with a 0.01 learning rate, Softmax activation function, and AdaBound optimizer. Our study reveals a significant information that there is a relationship among learning rate, optimizer, and classifier in improving detection accuracy. In the third part of the experiment, a K-fold cross-validation check further justified those parameters. Using the most effective deep CNN hyperparameters realized, this work might be extended to a variety of leaf disease detection applications. Despite the fact that this study obtained the highest detection accuracy, performance evaluation with multiple hyperparameters consumes a substantial amount of time and computer power. In the future, the convolutional neural network (CNN) pruning strategy may be explored to solve this issue.

Acknowledgments

We would like to thank the Institute of Energy, Environment, Research, and Development (IEERD, UAP) and the University of Asia Pacific for financial support.

References

- [1] Muhammad Hammad Saleem, Johan Potgieter and Khalid Mahmood Arif, "Plant Disease Detection and Classification by Deep Learning", *Plants*, ISSN: 2223-7747, pp. 468-489, Vol. 8, No. 11, 31st October 2019, Published by MDPI, DOI: 10.3390/plants8110468, Available: <https://doi.org/10.3390/plants8110468>.
- [2] Raihan Kabir, Salman Jahan, Md Rashedul Islam, Nabila Rahman and Md. Rajibul Islam, "Discriminant Feature Extraction using Disease Segmentation for Automatic Leaf Disease Diagnosis", in *Proceedings of the International Conference on Computing Advancements (ICCA 2020)*, 10-12 January 2020, Dhaka, Bangladesh, ISBN: 9781450377782, DOI: 10.1145/3377049.3377100, Article 32, pp. 1-7, Published by Association for Computing Machinery (ACM), Available: <https://dl.acm.org/doi/10.1145/3377049.3377100>.
- [3] Zongwei Zhou, Jae Shin, Lei Zhang, Suryakanth Gurudu, Michael Gotway *et al.*, "Fine-tuning convolutional neural networks for biomedical image analysis: Actively and incrementally", in *Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 21-26 July 2017, Honolulu, HI, USA, Electronic ISBN: 978-1-5386-0457-1, Print ISSN: 1063-6919, DOI: 10.1109/CVPR.2017.506, pp. 4761-4772, Published by IEEE, Available: <https://ieeexplore.ieee.org/document/8099989>.
- [4] Youbao Tang and Xiangqian Wu, "Scene text detection and segmentation based on cascaded convolution neural networks", *IEEE Transactions on Image Processing*, Print ISSN: 1057-7149, Electronic ISSN: 1941-0042, pp. 1509-1520, Vol. 26, No. 3, 20th January 2017, Published by IEEE, DOI: 10.1109/TIP.2017.2656474, Available: <https://ieeexplore.ieee.org/document/7828014>.
- [5] Ziwei Liu, Ping Luo, Xiaogang Wang and Xiaoou Tang, "Deep learning face attributes in the wild", in *Proceedings of the 15th IEEE International Conference on Computer Vision*, 7-13 December 2015, Santiago, Chile, Electronic ISSN: 2380-7504, USB ISBN: 978-1-4673-8390-5, DOI: 10.1109/ICCV.2015.425, pp. 3730-3738, Published by IEEE, Available: <https://ieeexplore.ieee.org/document/7410782>.
- [6] Nazia Hameed, Antesar Shabut, Fozia Hameed, Silvia Cirstea, Sorrel Harriet *et al.*, "Mobile-based Skin Lesions Classification Using Convolution Neural Network", *Annals of Emerging Technologies in Computing (AETiC)*, Print ISSN: 2516-0281, Online ISSN: 2516-029X, pp. 26-37, Vol. 4, No. 2, 1st April 2020, Published by International Association for Educators and Researchers (IAER), DOI: 10.33166/AETiC.2020.02.003, Available: <http://aetic.theiaer.org/archive/v4/v4n2/p3.html>.
- [7] J.Andrew Onesimu, Robin D.Sebastian, Yuichi Sei and Lenny Christopher, "An Intelligent License Plate Detection and Recognition Model Using Deep Neural Networks", *Annals of Emerging Technologies in Computing (AETiC)*, Print ISSN: 2516-0281, Online ISSN: 2516-029X, pp. 23-36, Vol. 5, No. 4, 1st October 2021, Published by

- International Association for Educators and Researchers (IAER), DOI: 10.33166/AETiC.2021.04.003, Available: <http://aetic.theiaer.org/archive/v5/v5n4/p3.html>.
- [8] Gao Huang, Zhuang Liu, Laurens Van Der Maaten and Kilian Q. Weinberger, "Densely connected convolutional networks", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2017)*, 21-26 July 2017, Honolulu, HI, USA, Electronic ISBN:978-1-5386-0457-1, Print on Demand(PoD) ISBN:978-1-5386-0458-8, Print ISSN: 1063-6919, DOI: 10.1109/CVPR.2017.243, pp. 4700-4708, Published by IEEE, Available: <https://ieeexplore.ieee.org/document/8099726>.
 - [9] Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton, "Imagenet classification with deep convolutional neural networks", *Communications of the ACM*, ISSN: 0001-0782, EISSN: 1557-7317, pp 84-90, Vol. 60, No. 6, 24th May 2017, Published by ACM, DOI: 10.1145/3065386, Available: <https://dl.acm.org/doi/10.1145/3065386>.
 - [10] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition", in *Proceedings of the 3rd International Conference on Learning Representations (ICLR2015)*, 7-9 May 2015, San Diego, USA, pp. 1-14, Available: <https://www.robots.ox.ac.uk/~vgg/publications/2015/Simonyan15/simonyan15.pdf>.
 - [11] Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun, "Deep Residual Learning for Image Recognition", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016)*, 27-30 June 2016, Las Vegas, NV, USA, Electronic ISBN: 978-1-4673-8851-1, Print on Demand(PoD) ISBN: 978-1-4673-8852-8, Electronic ISSN: 1063-6919, DOI: 10.1109/CVPR.2016.90, pp. 770-778, Published by IEEE, Available: <https://ieeexplore.ieee.org/document/7780459>.
 - [12] Mingxing Tan and Quoc Le, "Efficientnet: Rethinking model scaling for convolutional neural networks", in *Proceedings of the 36th International Conference on Machine Learning*, PMLR 97, 09-15 June 2019, Long Beach, California, USA, pp. 6105-6114, Available: <http://proceedings.mlr.press/v97/tan19a.html>.
 - [13] J. Arun Pandian, V. Dhilip Kumar, Oana Geman, Mihaela Hnatiuc, Muhammad Arif *et al.*, "Plant Disease Detection Using Deep Convolutional Neural Network", *Applied Sciences*, ISSN: 2076-3417, pp. 01-17, Vol. 12, No. 14, Article ID. 6982, 10 July 2022, Published by MDPI, DOI: 10.3390/app12146982, Available: <https://www.mdpi.com/2076-3417/12/14/6982>.
 - [14] Shanwen Zhang, Wenzhun Huang and Chuanlei Zhang, "Tree-channel convolutional neural networks for vegetable leaf disease recognition", *Cognitive Systems Research*, Print ISSN: 2214-4366, Online ISSN: 1389-0417, pp. 31-41, Vol. 53, January 2019, Published by Elsevier B.V., DOI: 10.1016/j.cogsys.2018.04.006, Available: <https://www.sciencedirect.com/science/article/pii/S1389041717303236>.
 - [15] Malusi Sibiya and Mbuyu Sumbwanyambe, "A Computational Procedure for the Recognition and Classification of Maize Leaf Diseases Out of Healthy Leaves Using Convolutional Neural Networks", *AgriEngineering*, ISSN: 2624-7402, pp. 119-131, Vol. 1, No. 1, 13th March 2019, Published by MDPI, DOI: 10.3390/agriengineering1010009, Available: <https://www.mdpi.com/2624-7402/1/1/9>.
 - [16] Keke Zhang, Qiufeng Wu, Anwang Liu and Xiangyan Meng, "Can Deep Learning Identify Tomato Leaf Disease?", *Advances in Multimedia*, Print ISSN: 1687-5680, Online ISSN: 1687-5699, pp. 1-10, Vol. 2018, Article ID: 6710865, 26th September 2018, Published by Hindawi, DOI: 10.1155/2018/6710865, Available: <https://doi.org/10.1155/2018/6710865>.
 - [17] Jihen Amara, Bassem Bouaziz and Alsayed Algergawy, "A Deep Learning-based Approach for Banana Leaf Diseases Classification", in *Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017*, B. Mitschang *et al.* (Eds.): BTW 2017 – Workshopband (Workshops), 6-10 March 2017, Stuttgart, Germany, pp. 79-88, Available: http://btw2017.informatik.uni-stuttgart.de/slidesandpapers/E1-10/paper_web.pdf.
 - [18] K. P. Ferentinos, "Deep learning models for plant disease detection and diagnosis", *Computers and Electronics in Agriculture*, ISSN: 0168-1699, pp. 311- 318, Vol. 145, February 2018, DOI: 10.1016/j.compag.2018.01.009, Available: <https://www.sciencedirect.com/science/article/abs/pii/S0168169917311742>.
 - [19] Kyosuke Yamamoto, Takashi Togami and Norio Yamaguchi, "Super-resolution of plant disease images for the acceleration of image-based phenotyping and vigor diagnosis in agriculture", *Sensors*, ISSN: 1424-8220, pp. 1-13, Vol. 17, No. 11, Article ID. 2557, 6th November 2017, Published by MDPI, DOI: 10.3390/s17112557, Available: <https://www.mdpi.com/1424-8220/17/11/2557>.
 - [20] Halil Durmuş, Ece Olcay Güneş and Mürvet Kırıcı, "Disease detection on the leaves of the tomato plants by using deep learning", in *Proceedings of the 2017 6th International Conference on Agro-Geoinformatics*, 7-10 August 2017, Fairfax, VA, USA, Electronic ISBN:978-1-5386-3884-2, USB ISBN:978-1-5386-3883-5, Print on Demand (PoD) ISBN:978-1-5386-3885-9, DOI: 10.1109/Agro-Geoinformatics.2017.8047016, pp. 1-5, Published by IEEE, Available: <https://ieeexplore.ieee.org/document/8047016>.
 - [21] Diane Larsen-Freeman, "Transfer of learning transformed", *Language Learning*, Online ISSN: 1467-9922, pp. 107-129, Vol. 63, No. s1, 13th February 2013, Published by Wiley, DOI: 10.1111/j.1467-9922.2012.00740.x, Available: <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9922.2012.00740.x>.

- [22] Anil Johnny and K. N. Madhusoodanan, "Dynamic Learning Rate in Deep CNN Model for Metastasis Detection and Classification of Histopathology Images," *Computational and Mathematical Methods in Medicine*, Print ISSN: 1748-670X, Online ISSN: 1748-6718, Vol. 2021, Article ID. 5557168, 26 Oct. 2021, pp. 1-13, Published by Hindawi, DOI: 10.1155/2021/5557168, Available: <https://doi.org/10.1155/2021/5557168>.
- [23] Pankaj Singh Rathore, Naveen Dadich, Ankit Jha and Debasish Pradhan, "Effect of Learning Rate on Neural Network and Convolutional Neural Network", *International Journal of Engineering Research & Technology (IJERT)*, ISSN (Online) : 2278-0181, Vol. 06, No. 17, Paper ID : IJERTCONV6IS17007, 5th January 2019, pp. 1-8, Published by IJERT, DOI: 10.17577/IJERTCONV6IS17007, Available: <https://www.ijert.org/research/effect-of-learning-rate-on-neural-network-and-convolutional-neural-network-IJERTCONV6IS17007.pdf>.
- [24] Muhammad Yaqub, Jinchao Feng, M. Sultan Zia, Kaleem Arshid, Kebin Jia *et al.*, "State-of-the-Art CNN Optimizer for Brain Tumor Segmentation in Magnetic Resonance Images", *Brain Sciences*, ISSN: 2076-3425, Vol. 10, No. 7, 3rd July 2020, pp. 427-446, Published by MDPI, DOI: 10.3390/brainsci10070427, Available: <https://doi.org/10.3390/brainsci10070427>.
- [25] S. Vani and T. V. Madhusudhana Rao, "An Experimental Approach towards the Performance Assessment of Various Optimizers on Convolutional Neural Network", in *Proceedings of the 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, 25 April 2019, Tirunelveli, India, Electronic ISBN: 978-1-5386-9439-8, DOI: 10.1109/ICOEI.2019.8862686, pp. 331-336, Available: <https://ieeexplore.ieee.org/abstract/document/8862686>.
- [26] Yingying Wang, Yibin Li, Yong Song and Xuwen Rong, "The Influence of the Activation Function in a Convolution Neural Network Model of Facial Expression Recognition", *Applied Sciences*, ISSN: 2076-3417, Vol. 10, No. 5, 10th March 2020, pp. 1897-1916, Published by MDPI, DOI: 10.3390/app10051897, Available: <https://doi.org/10.3390/app10051897>.
- [27] Gianluca Maguolo, Loris Nanni and Stefano Ghidoni, "Ensemble of convolutional neural networks trained with different activation functions", *Expert Systems with Applications*, ISSN: 0957-4174, Vol. 166, Article ID. 114048, 15 March 2021, pp. 1-8, Published by Elsevier, DOI: 10.1016/j.eswa.2020.114048, Available: <https://doi.org/10.1016/j.eswa.2020.114048>.
- [28] Jianli Feng and Shengnan Lu, "Performance Analysis of Various Activation Functions in Artificial Neural Networks", *Journal of Physics: Conference Series*, Online ISSN: 1742-6596, Print ISSN: 1742-6588, Vol. 1237, No. 2, Article ID. 022030, Jun 2019, pp.1-6, Published by IOP Publishing Ltd, DOI:10.1088/1742-6596/1237/2/022030, Available: <https://doi.org/10.1088/1742-6596/1237/2/022030>.



© 2023 by the author(s). Published by Annals of Emerging Technologies in Computing (AETiC), under the terms and conditions of the Creative Commons Attribution (CC BY) license which can be accessed at <http://creativecommons.org/licenses/by/4.0>.