Deep Learning: Image Classification of Plant Disease Based on ResNet and Transfer Learning

Fikayo Idowu

Email: Idowufikayo@gmail.com

ABSTRACT

In plant diseases, symptoms are typically visible which is why the accepted technique for diagnosing an infected plant is for an experienced plant pathologist to visually examine the infected plants. Unfortunately, this manual process is time consuming, and accuracy is dependent on the expertise of the pathologist. This problem presents an excellent opportunity for a computer-aided diagnosis. In this work, the utilization of deep learning architecture for plant disease classification is proposed. The dataset is multi-class in nature containing 38 classes. A custom **RESNET9 model** based on residual neural network is built and compared with other state of the art models. A plant disease dataset containing 87,867 images is used for the training of all the models. The results obtained from the training indicates that all models tested are efficient in classification of plant disease with the ResNET9 model achieving the highest accuracy of 99.83%, followed closely by MobileNet with 99.35% accuracy and EfficientNet with 99.00%. Evaluation metrics such as precision, recall, F1-score, and ROC-AUC was used to evaluate the results of the experiment. This study contributes to the development of plant disease identification using deep learning models.

INTRODUCTION

Plant diseases, if not managed properly, can have a rippling effect on the growth of their respective species (Gregory et al., 2009). On a larger scale, it could impact the global food production capabilities (Savary et al., 2012). Crops infected with diseases can experience reduced yields, and lower quality of produce. In the case where it affects crops that are already threatened or going into extinction, it could wipe out the existence of such crops and cause a complete crop failure (Carretero et al., 2010). This will result in significant economic losses for farmers and every player along the agricultural value chain, with possible trigger effects including food shortages and price increases for consumers (Savary & Willocquet, 2020).

The impact of plant diseases on global food production can be seen in various ways. Overall, plant diseases can have a significant impact on global food production, and efforts to prevent and control these diseases are crucial to ensuring food security for populations around the world. To reduce the possibility of these problems happening, it is important that disease identification be made early.

Several machine learning models have been deployed for the detection and classification of plant diseases. In recent decades, technological advancement has proven that Deep Learning, a more advanced subset of machine learning, has shown increased accuracy in the detection and classification of images (Kamilaris & Prenafeta-Boldú, 2018). Deep learning algorithms can learn and recognize patterns in large datasets and automatically analyze images of plant diseases, reducing the need for manual labor and expertise (Kamilaris & Prenafeta-Boldú, 2018). Consequently, this will reduce the timing and costs associated with disease

diagnosis and treatment while enhancing accurate diagnosis, allowing for more precise treatment and control of plant diseases (Sun et al., 2017). Also, deep learning algorithms can be easily modified and scaled to work for a varying number of crops (Muhammad Hammad Saleem et al., 2019).

Many deep learning models are implemented along with visualization techniques that detect and classify the symptoms of plant diseases. Across these models, there are several performance metrics used for the evaluation of the results including top –1%/top -5% error, classification accuracy and loss function (He et al., 2016a). In this research, we review past literature on the subject and compare their results with the aim of finding the best approach for our own experiment. Most recently, the best results for image classification have been gotten from algorithms built on convolutional neural networks (CNN) ((Argüeso et al., 2020). The effectiveness of CNN has mostly been attributed to its millions of tunable parameters. However, the limit of this model is that many annotated datasets is needed to learn visual features that characterize each disease in a plant image.

LITERATURE REVIEW

Most of the modern deep learning architectures for detection and classification we know today came into existence after the introduction of AlexNet (Alex Krizhevsky et al., 2017). Since that time, there have been constant upgrades of older models and the introduction of newer models. As deep learning began to gain momentum in research, the deployment into industries for real-life application began almost immediately, including in the agricultural industry.

(Hall et al., 2015) examined the robustness of common features for fine-grained leaf classification. Their aim was to move from using hand-crafted features (HCFs) which have been taken in controlled environments. Instead, they made use of more rigorous datasets like those on the field with varying shape, color and texture due to changing viewpoints and occlusion. Their research found that CovNet performed better than traditional hand-crafted features and that their combination gave the best result at 97.3% average accuracy, compared to 91.2% for HCFs. However, (Kamilaris & Prenafeta-Boldú, 2018) found that the method was not good at detecting occluded plants.

(Itzhaky et al., 2018) focused their research on leaf detection and counting using a dataset of tobacco and Arabidopsis plants images. They employed convolutional neural network (CNN) and parsed the dataset with each image annotated by a box. Using two methods for leaf counting, the first method takes the counting task as direct regression. They found that using multi scale representations of the image and predicting the leaf number using each of them improves the counting accuracy. Secondly, they suggested using leaf center annotation as the basic truth.

(Ferentinos, 2018) trained over 87,000 images, with 25 different plants with 58 different classes of plant, disease combinations, including plants that were healthy with different CNN architectures including AlexNet, AlexNetOWTbn, GoogLeNet, Overfeat, and VGG. Their research concluded that VGG outperformed all other models, gaining 99.53% with top-1 error rate of 0.47%. Another breakthrough in their research is that the training model requires low computational power of about 2ms on a single GPU. This makes real-life deployment possible for smartphones, drones, and other agricultural machinery that farmers and agronomists already know how to use. The high performance of CNN algorithms proves that it is largely dependable for plant image classification. However, the training could become unreliable if the training images used were not captured in cultivation fields.

Using over 7,000 cucumber leaf images including healthy ones and those infected with varying viruses, (Fujita et al., 2016) found that using a classification system based on CNN gave an average of 82.3% accuracy using the 4-fold cross validation strategy. The photographs were taken on site while ensuring all images followed a simple specification that each image must contain a leaf roughly at its center, thereby providing the research team with a lot of varying appearances that are differentiated by angle, lightning, and background.

(Türkoğlu & Hanbay, 2019) researched systems based on deep CNNs and developed models for the detection of plant diseases and pests as a classification problem. The research used a dataset consisting of real plant disease and pest images from Turkey. They obtained deep features from deep learning models and classified the training dataset using SVM, ELM, and KNN. Based on transfer

learning, the deep learning models were then fine-tuned. Their results showed that the deep learning models performed better when compared to the traditional methods. The evaluation for deep feature extraction was better than transfer learning methods with the experiment based on deep feature extraction and classifier methods producing better accuracy scores compared to the other layers. The best accuracy score was 97.86%, gotten from ResNet-50 model and SVM classifier. However, their research, like many deep learning research did not utilize the deep learning visualization, making most of the models' black boxes with no way to understand their workings.

To ensure the transparency of deep learning architectures, (Brahimi et al., 2018) made use of saliency maps as a visualization method to understand and interpret the CNN classification algorithm. The researchers tested multiple convolutional neural network models on a public dataset for plant diseases classification. The results showed that CNN models outperform the state-of-the-art results of plant diseases classification with an accuracy as high as 99.76%.

(Cruz et al., 2017) set out with their research to prove that transfer learning can be very effective when it is impossible to collect new plant images. Transfer learning is the storing of knowledge for the purpose of re-application when solving a new problem. The research uses deep learning algorithm to fuse the data at each level of abstraction to better the convergence of knowledge gained from each iteration. The models also discovered fractional features from data that could be used to detect veins and colors of symptomatic leaves.

(Picon et al., 2019) analyzed the performance of deep residual neural network models in the early detection of plant disease like Septoria (*Septoria triciti*), Tan

Spot (*Drechslera triciti-repentis*) and Rust (*Puccinia striiformis* & *Puccinia recondita*) with more than 8,000 images. Their results showed that Balanced Accuracy values (BAC) leaped to *0.78* on the classical approach to an average BAC of *0.84* when using a Residual Neural Network without further improvement. However, there were issues of low specificities that will not deliver high reliability in the detection of same plants in real case scenario.

Neural networks have been known to solve problems in deep learning, analyzing complex dataset. However, its combination with hyperspectral image analysis (HSI) is often less discussed. HSI is "a combination of digital imaging and spectroscopy. It involves the process of extracting spectral information to distinguish and identify spectrally unique materials where the wavelength bands are narrow and contiguous" (Kerkech et al., 2018). Essentially, the number and quality of data in both neural network and hyperspectral image analysis remains a central problem for the development of effective HSI-DL solutions that are adaptable across industries (Signoroni et al., 2019).

Convolutional Neural Networks

Convolutional Neural Network (CNN) is one of the mainstream deep learning models for image classification. It is known for its effectiveness in extracting features from the original images so that CNN can recognize the feature rules directly from the original pixels with minimum preprocessing (Fang et al., 2019). The key idea behind CNNs is that it leverages the spatial structure and local correlations present in the input data. Unlike traditional fully connected neural networks, CNNs exploit the concept of convolution, which involves applying a set of learnable filters (or kernels) to the input data. These filters systematically slide

over the input, performing element-wise multiplications and summations to produce feature maps that capture local patterns and features.

Transfer Learning

Transfer learning is a deep leaning technique that focuses on transferring knowledge across domains. The concept of transfer learning may have come from the generalization theory in psychology which explains that the knowledge of a skill can be the basis of learning another skill but both skills must be related in some way (F. Zhuang et al., 2021).

Essentially, transfer learning leverages the presence of a pre-trained model as a starting point to solve related problems for a new dataset instead of training another model from scratch. Transfer learning is beneficial when you have a limited amount of labeled data for your specific task. It makes leveraging knowledge and representations learned by models trained on large datasets, reducing the need for extensive training on specific tasks. This method can save time in training and preserve computational resources while achieving almost comparable or even better results.

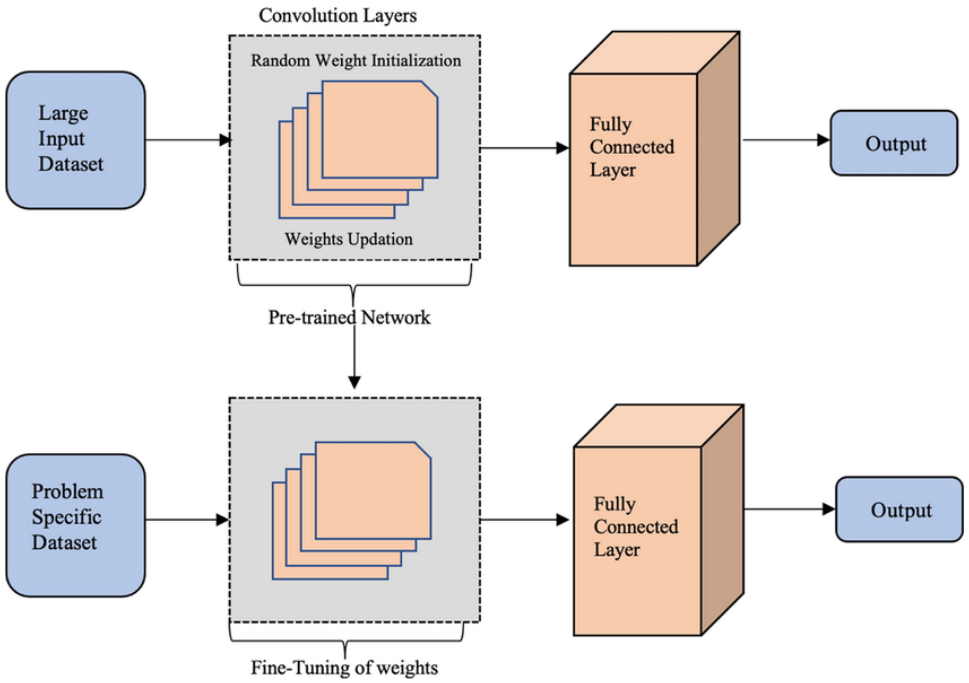


Figure 1: Architecture of Transfer Learning Model

METHOD AND EXPERIMENT SETUP

Here, we layout the structure of our proposed experiment as in Fig xxx.

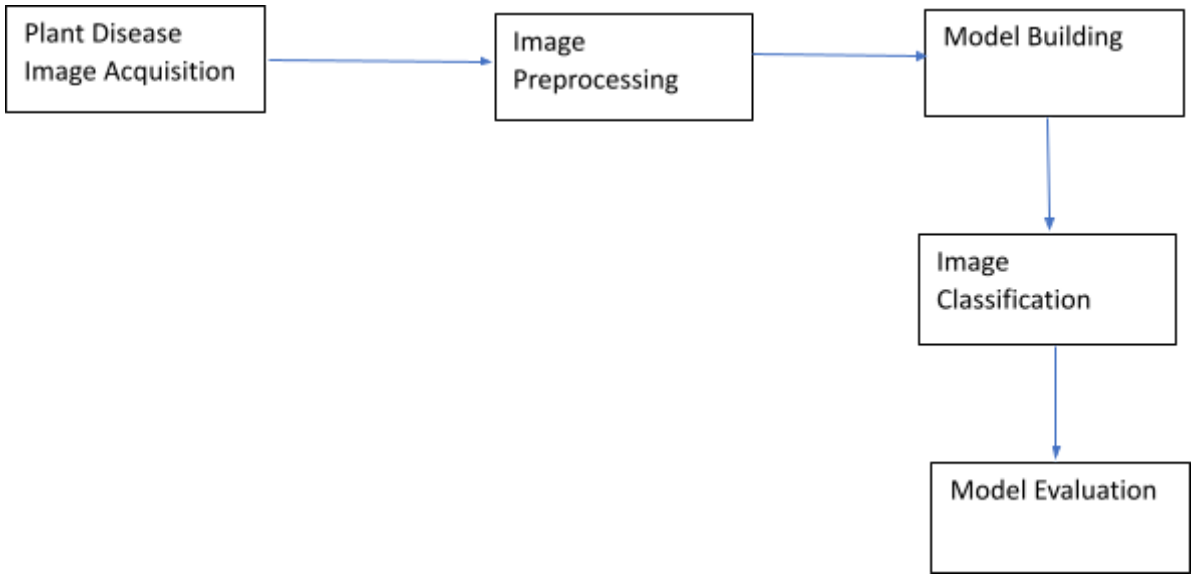


Image Dataset Acquisition

Gathering our own image dataset would have taken us more time than the timeline of this project allowed. Therefore, we decided to make use of data from the internet. We got our data from the PlantVillage dataset on Kaggle (Kaggle,

2016). We then did some augmentation to enlarge the dataset units and make it fit for our experiment. The final dataset used consists of about 87,000 RGB images of healthy and diseased crop leaves. The diseased crops are categorized into 38 different classes.

We divided the dataset into training and validation at an 80/20 ratio, respectively. We used the train/test spilt method in Scikit learn to ensure there is balance in the splitting.

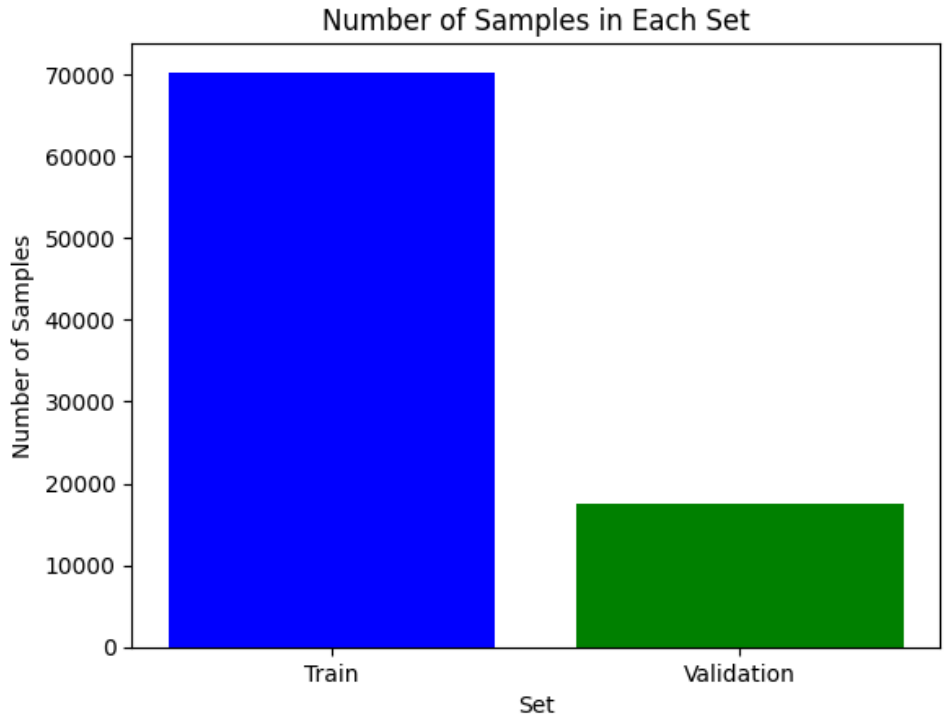


Figure 3: Distribution of Train and Test Dataset.

For our test dataset, we had another directory with 33 images which will be used for the final prediction. Further in exploring our data, we found that there are 14 unique plants and 26 unique diseases. To get a sense of how many images we

have for each disease, we plotted the graph in fig xxx, signifying that each disease is appropriately represented and there would be no case of underrepresentation leading to imbalanced data.

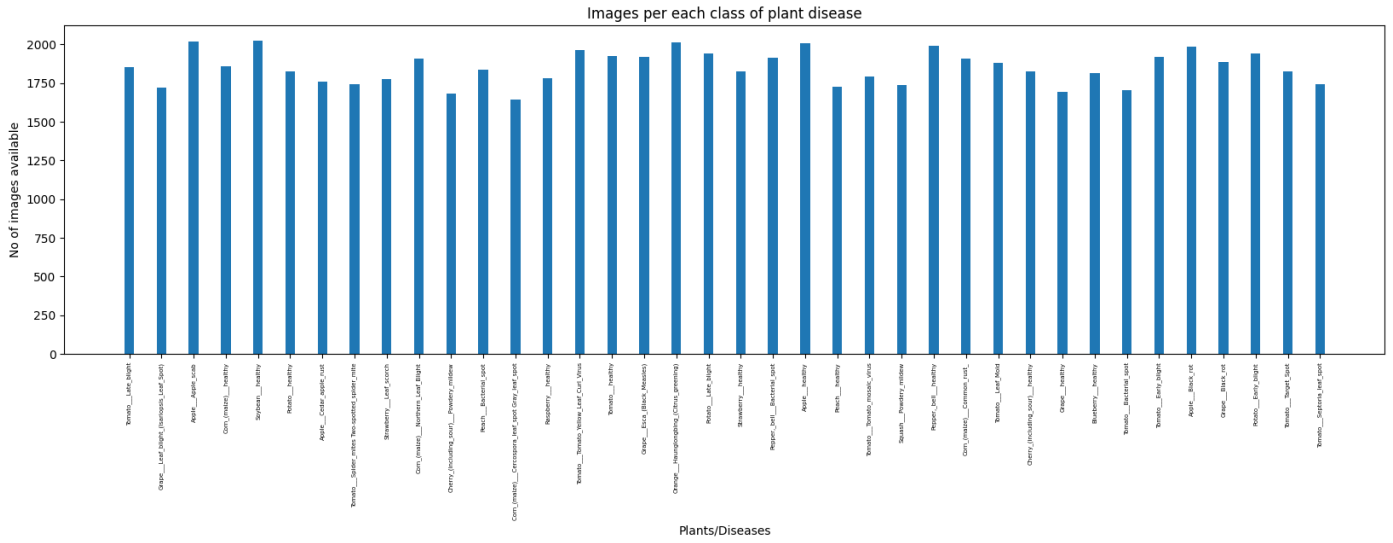


Figure 4: Distribution of Images per Class.

Image Data Pre-processing

Both datasets for training and validation were converted to tensors. *torchvision.datasets* is the most widely used class for loading datasets. It is also utilized in loading custom datasets. For our experiment, however, we have used a subclass *torchvision.datasets.ImageFolder* which helps in loading the image data when the data is arranged in this format: *root/dog/xxx.png* or *root/cat/nsdf3.png*. The shape of our data remains at 256 x 256 resolution.

After loading the dataset, we needed to transform the pixel values of each image (0-255) to 0-1 as neural networks work better with normalized data. The entire array of pixel values is converted to a torch tensor and then divided by 255. The reason why normalization is important in neural networks is that If in your

dataset, there is a feature that is all positive or all negative, learning will become harder for the nodes in the layer that follows. However, If the data is transformed, such that it has a mean close to zero, we'll be sure that both positive values and negative ones are represented in the values. Another reason normalization helps is regarding the scale of the inputs. Normalization ensures that the magnitude of the values that a feature assumes is the same across board.

Once normalization was complete, we set the seed value to 7 and passed the training and validation data into the data loader. Data loader is a subclass which comes from *torch.utils.data*. It helps in loading large and memory-consuming datasets in a batch size of 32. *Batch size* is the total number of images passed as input in one forward propagation of the CNN. Essentially, batch size defines the number of samples that will be propagated through the network.

Neural Networks Implementation

Residual Neural Network - RESNET9

Residual Neural Network (ResNet) is a type of convolutional neural network (CNN). It was introduced by (He et al., 2016b) to resolve the vanishing gradient problem. The model learns residual functions with reference to the layer inputs, instead of unreferenced functions. Thereby replacing the probability that each stacked layer will directly fit a desired underlying mapping, with an assuredness that each layer fits a residual mapping. The residual blocks are piled on each other. A ResNet-100 will have a hundred layers using blocks.

For this research, a convolutional neural network architecture called ResNet9 for image classification tasks was built. ResNet9 architecture is a smaller version of the ResNet architecture that includes nine layers. Convolutional blocks, residual blocks, and a classifier layer. It applies convolutional operations, residual connections, and pooling operations to extract features and perform down sampling.

While the underlying mapping was formerly denoted as;

$$H(x)$$

(1)

The stacked nonlinear layer will fit another mapping of

$$F(x) := H(x) - x$$

(2)

The original mapping will then be reframed to

$$F(x) + x$$

(3)

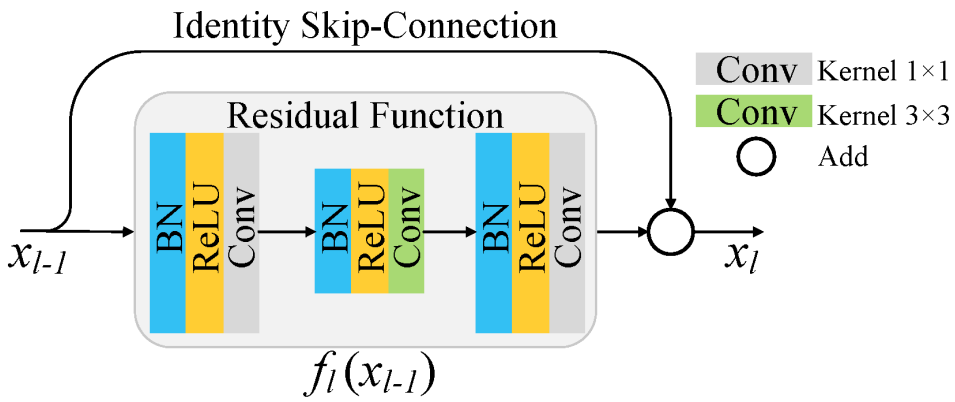


Figure 5: Custom RESNET (RESNET9) Architecture.

The breakdown of the RESNET9 architecture is as follows:

ConvBlock

This function defines a basic convolutional block, which consists of a convolutional layer, batch normalization layer, and ReLU activation function. It takes three arguments: **in_channels** (number of input channels), **out_channels** (number of output channels), and **pool** (a boolean flag indicating whether to apply max pooling after the convolutional block). If the **pool** flag is set to **True**, a max pooling layer is appended to the **layers** list. Finally, the function returns a **nn.Sequential** module that sequentially executes the layers in the **layers** list.

Batch normalization

Batch normalization is a technique used to normalize the intermediate activations in a neural network. It helps in stabilizing and accelerating training, reducing

internal covariate shift, and acting as a form of regularization. The **nn.BatchNorm2d** was used as the batch normalization in this project.

Activation function (ReLU)

Rectified Linear Unit (ReLU) is a commonly used activation function in deep neural networks. It applies an element-wise activation to the input, setting negative values to zero and leaving positive values unchanged. It helps in capturing and enhancing the important features while suppressing irrelevant or noisy information.

Pooling layer (Max Pooling)

The pooling layer used in the project is the max pooling layer, implemented using the **nn.MaxPool2d** module. Max pooling is a down sampling operation that reduces the spatial dimensions of the input tensor while retaining the most salient features.

The **nn.MaxPool2d** module operates on 2D input tensors (e.g., images), performs max pooling along the spatial dimensions (width and height) and is used to apply max pooling after certain convolutional layers. The **nn.MaxPool2d.** It takes several arguments, including the kernel size (**kernel_size**), stride (**stride**), padding (**padding**), and dilation (**dilation**), among others.

Classifier Layer

The classifier layer is defined as **nn.Linear(512, num_diseases)** within the **ResNet9** class. It is responsible for the final classification step in the network, transforming

the features extracted by the convolutional layers into predictions for different diseases. The input to the classifier layer has 512 features, which corresponds to the number of output channels from the last convolutional layer (**conv4**).

The output size of the classifier layer is determined by the **num_diseases** argument passed to the **ResNet9** constructor, representing the number of classes or diseases the model is trained to classify. The **nn.Linear** layer performs a linear transformation of the input features, applying matrix multiplication and bias addition.

Optimizers (Adam)

Adam optimizer is an adaptive optimization algorithm commonly used for training deep neural networks. It incorporates the concept of momentum, which helps accelerate gradient descent in the relevant direction and dampens oscillations.

Loss Function (Cross-entropy)

The cross-entropy loss measures the dissimilarity between the predicted class probabilities and the true class labels. It is particularly suitable for multi-class classification problems like plant disease image classification.

Table 1: Summary of RESNET9 Parameter.

Layer (Type)	Output Shape	Parameters
Conv2d-1	[-1, 64, 256, 256]	1,792
BatchNorm2d-2	[-1, 64, 256, 256]	128
ReLU-3	[-1, 64, 256, 256]	0
Conv2d-4	[-1, 128, 256, 256]	73,856
BatchNorm2d-5	[-1, 128, 256, 256]	256
ReLU-6	[-1, 128, 256, 256]	0
MaxPool2d-7	[-1, 128, 64, 64]	0
Conv2d-8	[-1, 128, 64, 64]	147,584
BatchNorm2d-9	[-1, 128, 64, 64]	256
ReLU-10	[-1, 128, 64, 64]	0
Conv2d-11	[-1, 128, 64, 64]	147,584
BatchNorm2d-12	[-1, 128, 64, 64]	256
ReLU-13	[-1, 128, 64, 64]	0
Conv2d-14	[-1, 256, 64, 64]	295,168
BatchNorm2d-15	[-1, 256, 64, 64]	512
ReLU-16	[-1, 256, 64, 64]	0
MaxPool2d-17	[-1, 256, 16, 16]	0
Conv2d-18	[-1, 512, 16, 16]	1,180,160
BatchNorm2d-19	[-1, 512, 16, 16]	1,024

ReLU-20	[-1, 512, 16, 16]	0
MaxPool2d-21	[-1, 512, 4, 4]	0
Conv2d-22	[-1, 512, 4, 4]	2,359,808
BatchNorm2d-23	[-1, 512, 4, 4]	1,024
ReLU-24	[-1, 512, 4, 4]	0
Conv2d-25	[-1, 512, 4, 4]	2,359,808
BatchNorm2d-26	[-1, 512, 4, 4]	1,024
ReLU-27	[-1, 512, 4, 4]	0
MaxPool2d-28	[-1, 512, 1, 1]	0
Flatten-29	[-1, 512]	0
Linear-30	[-1, 38]	19,494

MobileNet and EfficientNet architecture were employed for the transfer learning technique in this research.

Mobile Net

The convolutional neural network does not work well with real-time applications and low-memory portable devices because of the large amount of computational power needed. A possible solution is to compress and accelerate the deep convolutional neural networks to reduce parameters, computation cost, and lower power consumption. This is where MobileNet is efficient.

MobileNet is a lightweight convolutional neural network developed by Google researchers in 2017 to achieve a balance between accuracy and latency in computation. The architecture gained popularity due to its characteristics of low memory consumption, low computational overhead, and fast inference speed while maintaining competitive accuracy (L. Ou & K. Zhu, 2022). Due to these characteristics, the MobileNet architecture has seen a growing demand in the deployment of intelligent vision systems on low-cost microcomputers (Glegoła et al., 2021).

The key idea behind MobileNet is to use depth-wise separable convolutions, which reduce the computational cost of convolutions while preserving the representational capacity of the network.

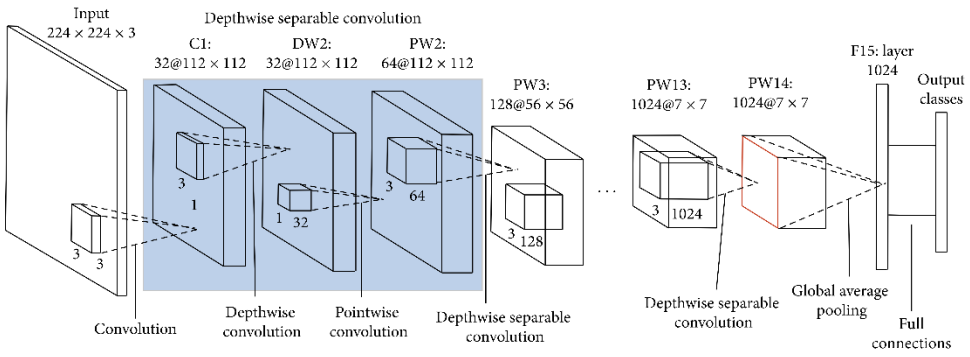


Figure 6: Architecture of MobileNET model.

EfficientNET-B2

(Tan & Le, 2019)introduced the EfficientNet architecture to provide higher performance for classification, with a smaller number of parameters and computational time. When compared with other CNN models, the EfficientNet architecture uses a new activation function called Swish instead of the Rectifier Linear Unit (ReLU) activation function.

Essentially, EfficientNet was designed to systematically achieves more efficient results by uniformly scaling depth, width, and resolution while scaling down the model. By scaling these dimensions in a principled manner, EfficientNet can effectively capture complex patterns and features while being computationally efficient (Atila et al., 2021).

The model also uses a combination of efficient building blocks, such as inverted residual blocks and squeeze-and-excitation blocks, to improve model efficiency. These blocks reduce the number of parameters and computations while maintaining or even enhancing representational power. It has become a popular choice for many computer vision tasks. Its scalable nature allows users to choose the appropriate model size based on their computational constraints, making it applicable across different devices and deployment scenarios.

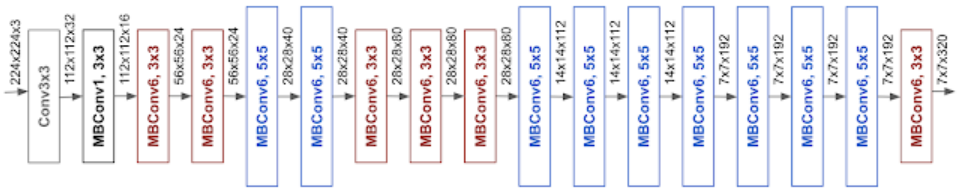


Figure 7: Architecture of EfficientNET-B2

Evaluation Criteria

Evaluation Criteria provide insights into different aspects of a deep learning model's performance, such as accuracy, precision, recall, trade-offs between true positives and false positives, and overall performance across different thresholds or classes. The choice of evaluation method depends on the specific task and the desired evaluation criteria.

For this project, precision, recall, F1 Score, Confusion Metrics are the evaluation metrics used.

Precision

Precision measures the proportion of correctly predicted positive instances out of the total predicted positive instances. Precision is particularly useful when the focus is on minimizing false positive predictions.

The precision metric is calculated using the following formula:

Precision

$$TP/(TP + FP)$$

(4)

where:

- TP (True Positives) represents the number of instances that are correctly predicted as positive by the model.
- FP (False Positives) represents the number of instances that are wrongly predicted as positive by the model when they are negative.

Recall

Recall is an evaluation metric used in deep learning to assess the model's ability to correctly identify positive instances. It measures the proportion of correctly predicted positive instances out of the total actual positive instances. A high recall value indicates that the model has a low rate of missing malignant tumors (false negatives).

The recall metric is calculated using the following formula:

$$TP/(TP + FN)$$

(5)

F1 Score

The F1-score is an evaluation metric used in deep learning that combines precision and recall into a single measure. It provides a balance between these two metrics and is particularly useful when both precision and recall need to be considered simultaneously. The harmonic mean places more weight on low values. As a result, the F1-score will be high only if both precision and recall are high. If either precision or recall is low, the F1-score will also be low.

The F1-score is calculated as the harmonic mean of precision and recall:

$$F1 - score = 2 * (precision * recall)/(precision + recall)$$

(6)

Confusion metrics

A confusion matrix is a table that summarizes the performance of a classification model by showing the counts of true positive, true negative, false positive, and false negative predictions. It allows for a more nuanced evaluation of the model's performance beyond a single metric. It helps identify the types of errors made by the model and provides insights into the strengths and weaknesses of the classification model.

RESULTS

Model Implementation

In our experiments, all models were run on Kaggle to make use of the platform's GPU. ResNet9, MobileNet, and EfficientNet were all implemented with Adam as the optimizer and CrossEntropyLoss as the loss function because the data is multiclass.

Table 2: Summary of Models Implementation.

	ResNet9	MobileNet	EfficientNet
Batch Size	32	32	32
Epochs	30	30	32
Image Size	256	256	256
Learning Rate	0.01	0.01	0.01
Loss function	CrossEntropyLoss	CrossEntropyLoss	CrossEntropyLoss
Optimizer	Adam	Adam	Adam

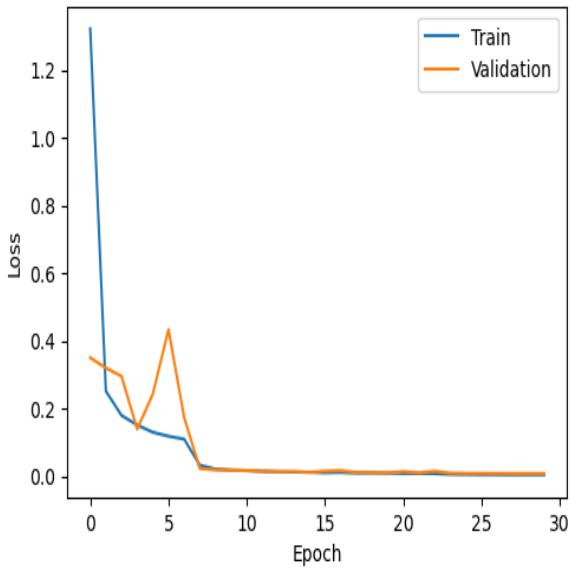
Evaluation of All Models Performance

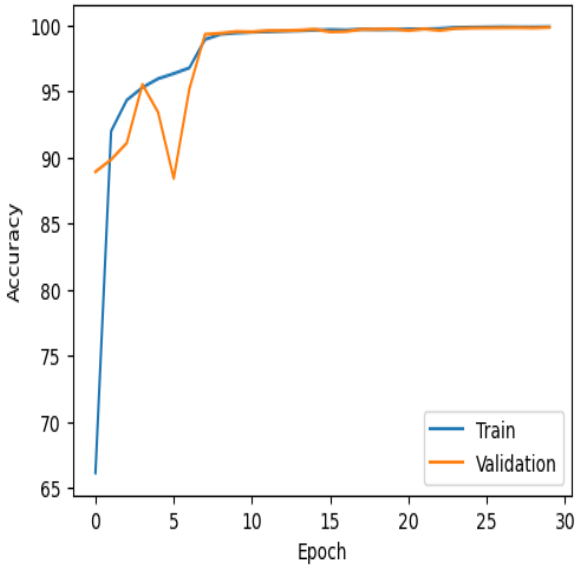
Analysis of Train loss/accuracy and Test loss/accuracy for all models

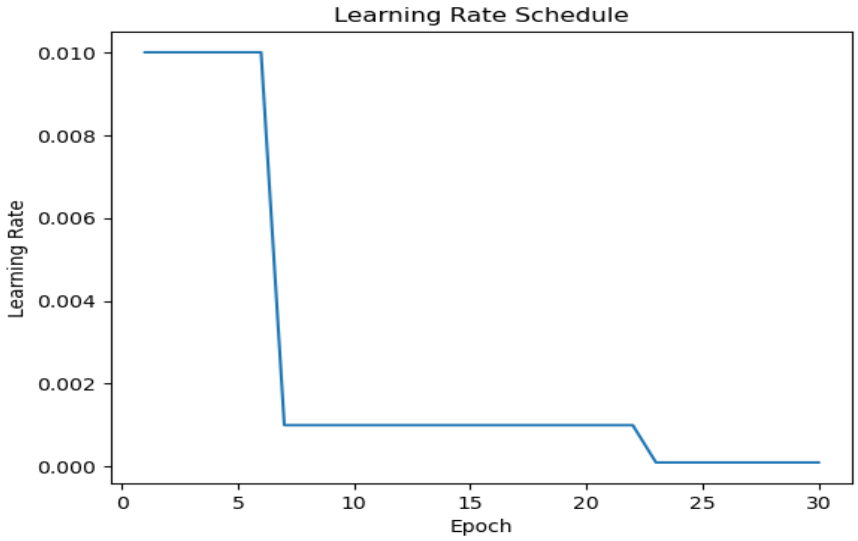
The custom RESNET9 model achieved the highest training accuracy of 99.88% and validation accuracy of 99.83% after 30 epochs. The MobileNet model also

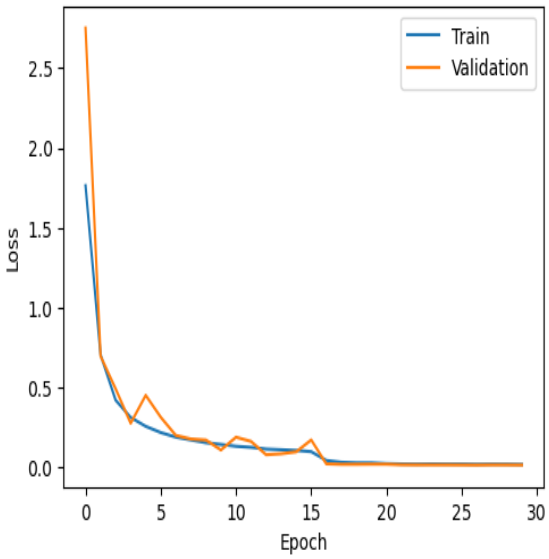
achieved a train accuracy of 99.88% with validation accuracy of 99.60% while the EfficientNet-b2 model achieved a training accuracy of 99.40% and validation accuracy of 99.60%. From this, it is concluded that a residual neural network is slightly more effective than some pretrained model in the classification of plant diseases. The learning rate for all models as shown in figure 8 – figure 10 remains constant for all models at initial epochs, however due to the implementation of 'ReduceLROnPlateau' scheduler, the learning rate gradually reduces based on the validation loss plateauing. This helped the models to converge better and finding a better minimum. Overall, all models show good generalization as their validation loss decreases steadily.

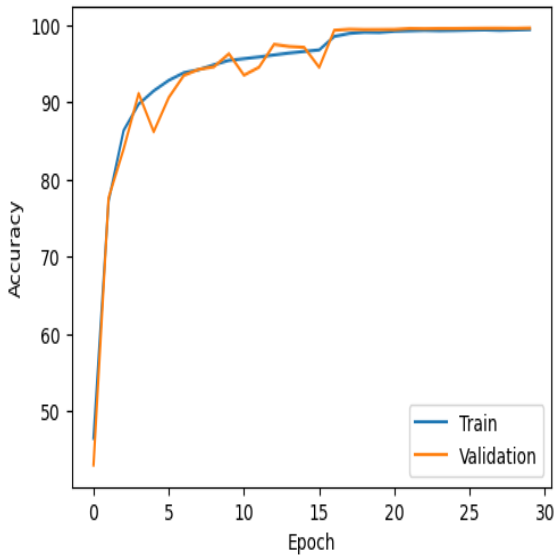
In terms of computational cost, The MobileNet performed best as it took a short time of 134 minutes to complete while the custom RESNET9 and the pretrained EfficientNet-b2 took about 70% longer with 253 minutes and 252 minutes respectively. From this observation, MobileNet is recommended as it generalizes well and takes little computational cost.

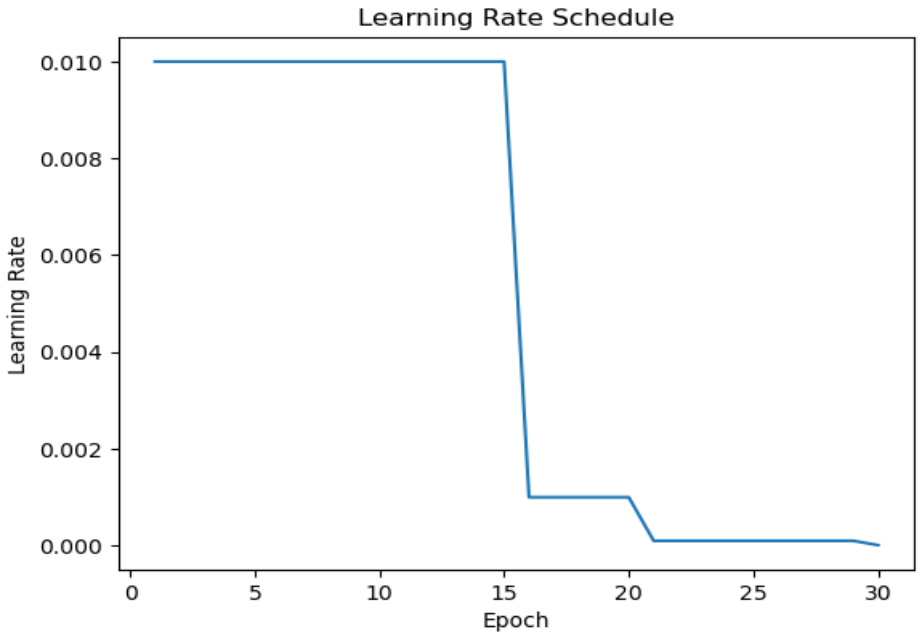

Train
Validation
Loss
Epoch
0.0
0.2
0.4
0.6
0.8
1.0
1.2
0
5
10
15
20
25
30

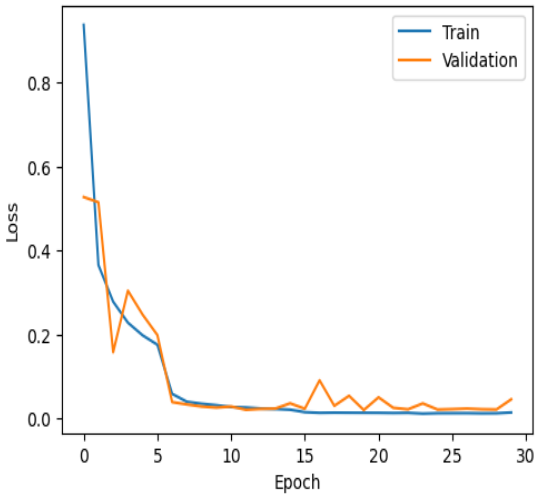

Train
Validation
Accuracy
Epoch
65
70
75
80
85
90
95
100
0
5
10
15
20
25
30

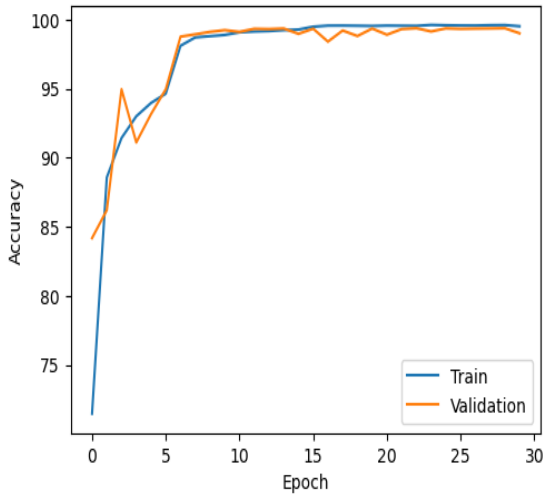

Learning Rate Schedule
Learning Rate
Epoch
0.000
0.002
0.004
0.006
0.008
0.010
0
5
10
15
20
25
30

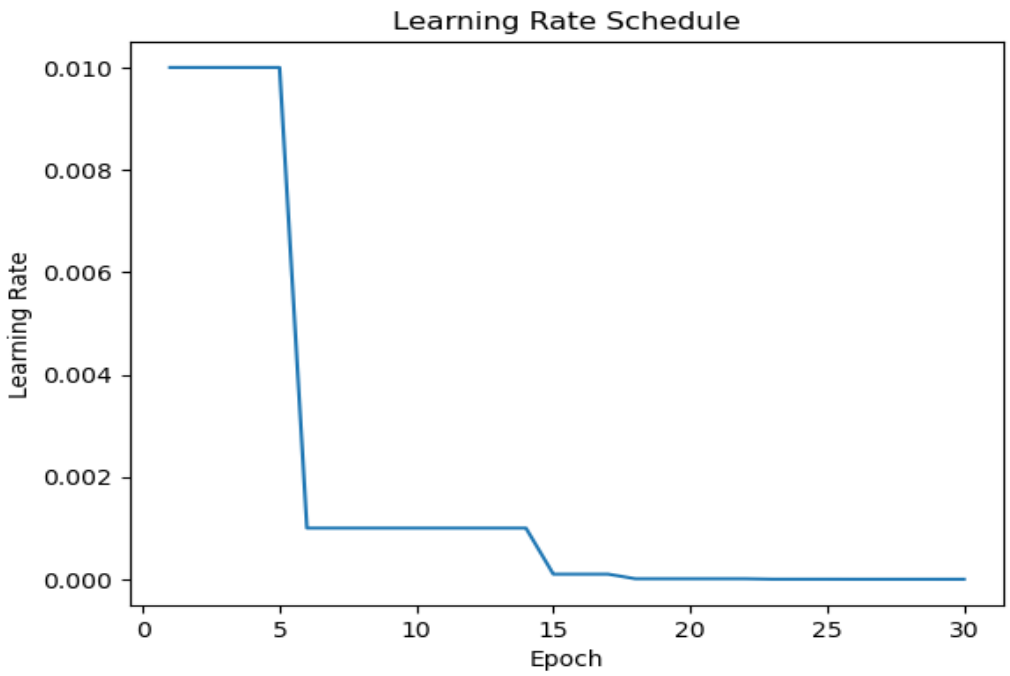

Train
Validation
Loss
Epoch
0.0
0.5
1.0
1.5
2.0
2.5
0
5
10
15
20
25
30


Train
Validation
Accuracy
Epoch
50
60
70
80
90
100
0
5
10
15
20
25
30


Learning Rate Schedule
Learning Rate
0.010
0.008
0.006
0.004
0.002
0.000
0
5
10
15
20
25
30
Epoch


Train
Validation
Loss
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20
25
30
Epoch


Accuracy
100
95
90
85
80
75
0
5
10
15
20
25
30
Epoch
Train
Validation


Learning Rate Schedule
0.010
0.008
0.006
0.004
0.002
0.000
Learning Rate
0
5
10
15
20
25
30
Epoch

Analysis of Confusion Matrix for all models.

To properly understand the prediction ability of each model, the confusion matrix was plotted to aid visualization. The dark diagonal squares in the matrix represents the number of true positives predicted by each model while the lighter squares represent the wrong classification. The higher the values in the diagonal squares, the more precise the prediction for that class is.

All models in this research performed reasonably well within the same range according to the matrix as shown in figure 11 to figure 13. There are minimal false predictions across board.

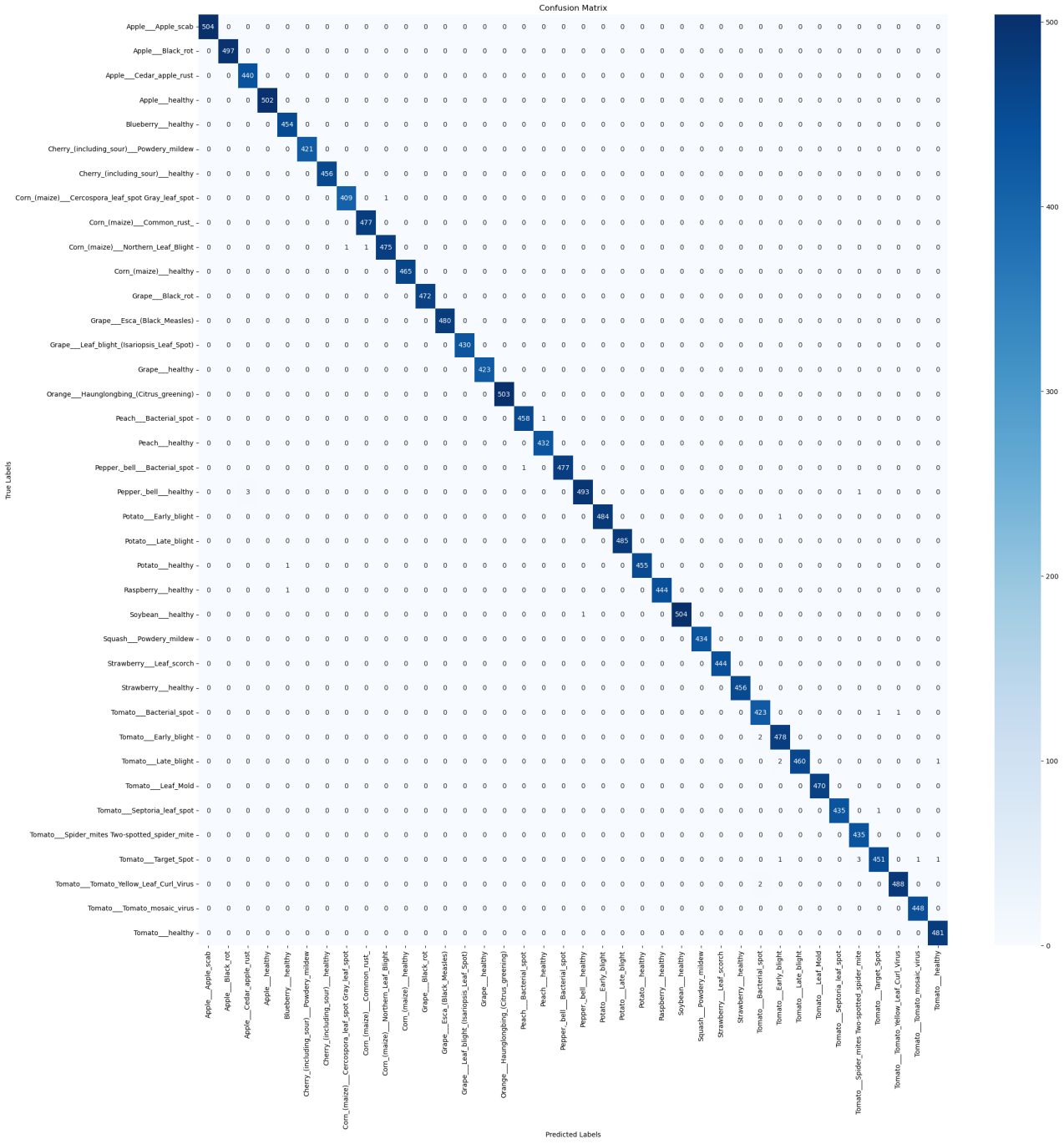


Figure 11: Confusion Matrix for RESNET9

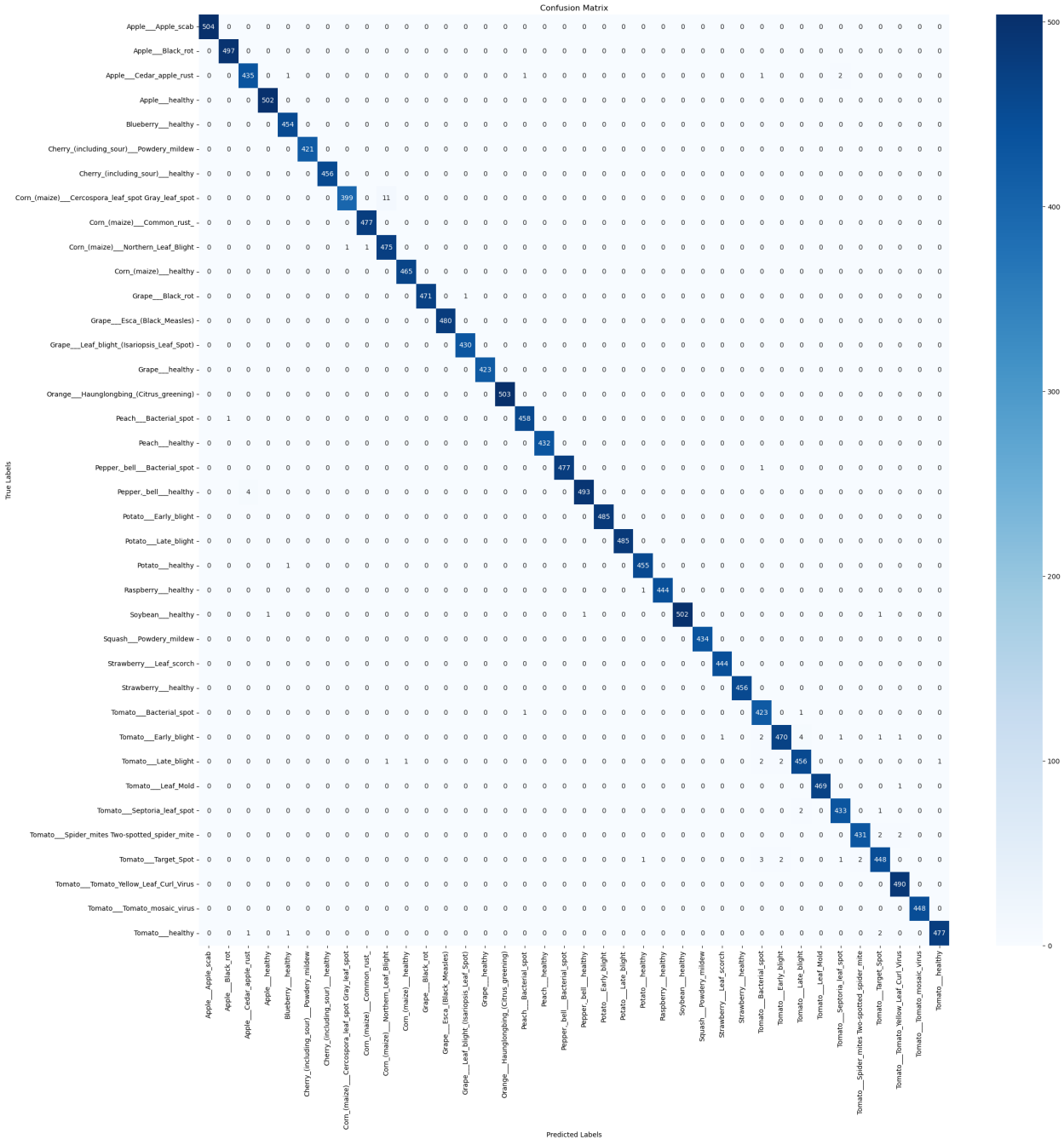


Figure 12: Confusion Matrix for MobileNET

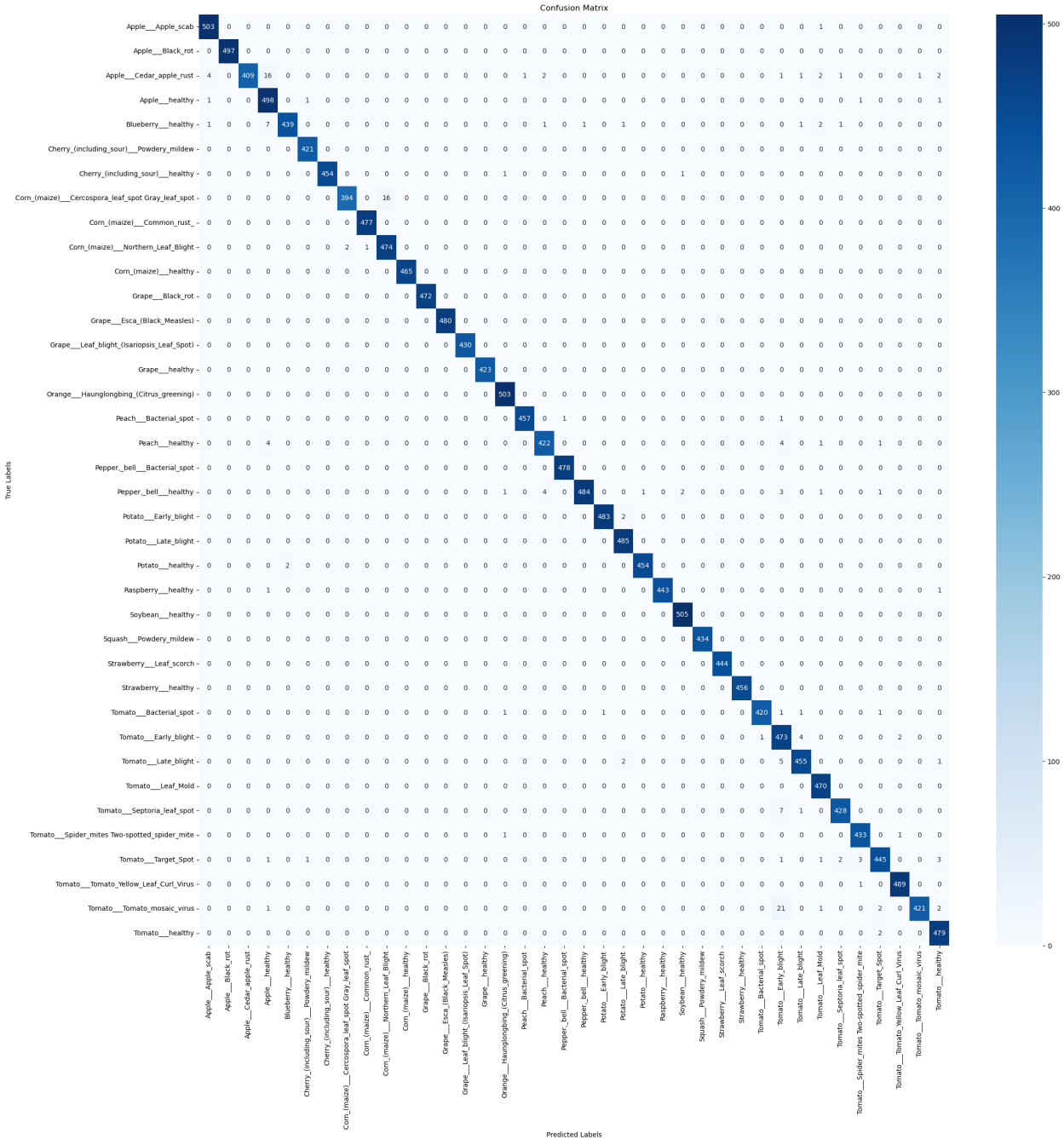


Figure 13: Confusion Matrix for EfficientNET-B2

Comparative Analysis of Precision, Recall and F1 Score for all Models.

The dataset contains 38 classes of plant disease and precision, recall and F1 score metrics was utilized to assess each model's performance The evaluation results was aggregated and analysed to provide insights into the effectiveness of each model in accurately identifying plant diseases. The following is a breakdown of the analysis:

A. RESNET9 MODEL: This model demonstrates outstanding performance in classifying plant diseases by achieving an overall accuracy of 99.84%. The weighted average precision, recall and F1 score was 99.84 which shows a high score for the model. The macro-average precision, recall and F1 score are 99.831%, 99.836% and 99.834% respectively. The support value which indicates the number of instances in each class is consistent at 17,572. Overall, these results suggests that RESNET9 model performs exceptionally well across the 38 diseases classes.

Table 3: Summary of Aggregated Precision, Recall and F1 Score for RESNET9 Model

RESNET 9				
	Precision	Recall	F1-score	Support
Accuracy	0.998350	0.99835	0.998350	0.99835
Macro Avg	0.998318	0.998318	0.998337	17572

Weighted Avg	0.998355	0.998350	0.998349	17572

B. MobileNET MODEL: This model has exhibited commendable performance in classifying plant diseases by achieving an accuracy of 99.266%. The weighted average precision, recall and F1 score stands at 99.266% while the macro average is 99.27%, 99.24% and 99.25% respectively.

Table 4: Summary of Aggregated Precision, Recall and F1 Score for MobileNET Model.

MOBILENET				
	Precision	Recall	F1-score	Support
Accuracy	0.996016	0.996016	0.996016	0.996016
Macro Avg	0.995991	0.995945	0.995956	17572
Weighted Avg	0.996036	0.996016	0.996015	17572

C. EfficientNET-B2 MODEL: This model demonstrates competitive performance in the classification problem at hand with an accuracy of 99.004%. The weighted average precision, recall and F1 score was aggregated to 99.004% while the macro average is 99.060%, 98.976% and 99.004% respectively.

Table 5: Summary of Aggregated Precision, Recall and F1 Score for EfficientNET -B2 Model.

EFFICIENTNET - B2				
	Precision	Recall	F1-score	Support
Accuracy	0.990041	0.990041	0.990041	0.990041
Macro Avg	0.990600	0.989760	0.990042	17572
Weighted Avg	0.990337	0.990041	0.990051	17572

In summary, all three models showcase remarkable performance in classifying plant diseases, achieving high accuracy, precision, recall and F1 score. The RESNET9 model exhibits the highest accuracy, precision, recall and F! score amongst the three models and its closely followed by MobileNET and EfficientNET -B2. These findings highlight the sustainability of these models for plant disease classification tasks which is valuable for researchers in the field of agriculture and pathology.

Analysis of ROC curve for all models.

The ROC-AUC score is a widely used metric for evaluating the performance of binary classification models. As can be seen in figure 14 – figure 16, all models achieved an exceptional score of 1 across all 38 classes of plant diseases. The exceptional ROC-AUC score of 1 indicates that all three models exhibit outstanding discriminative ability and are capable of distinguishing between different plant

disease classes with high accuracy. Obtaining a perfect ROC-AUC score for all 38 disease classes is a remarkable achievement, highlighting the models' effectiveness in capturing the discriminative features specific to each disease. Such high scores indicate that the models have learned complex patterns and representations from the input images, enabling them to accurately differentiate between various diseases.

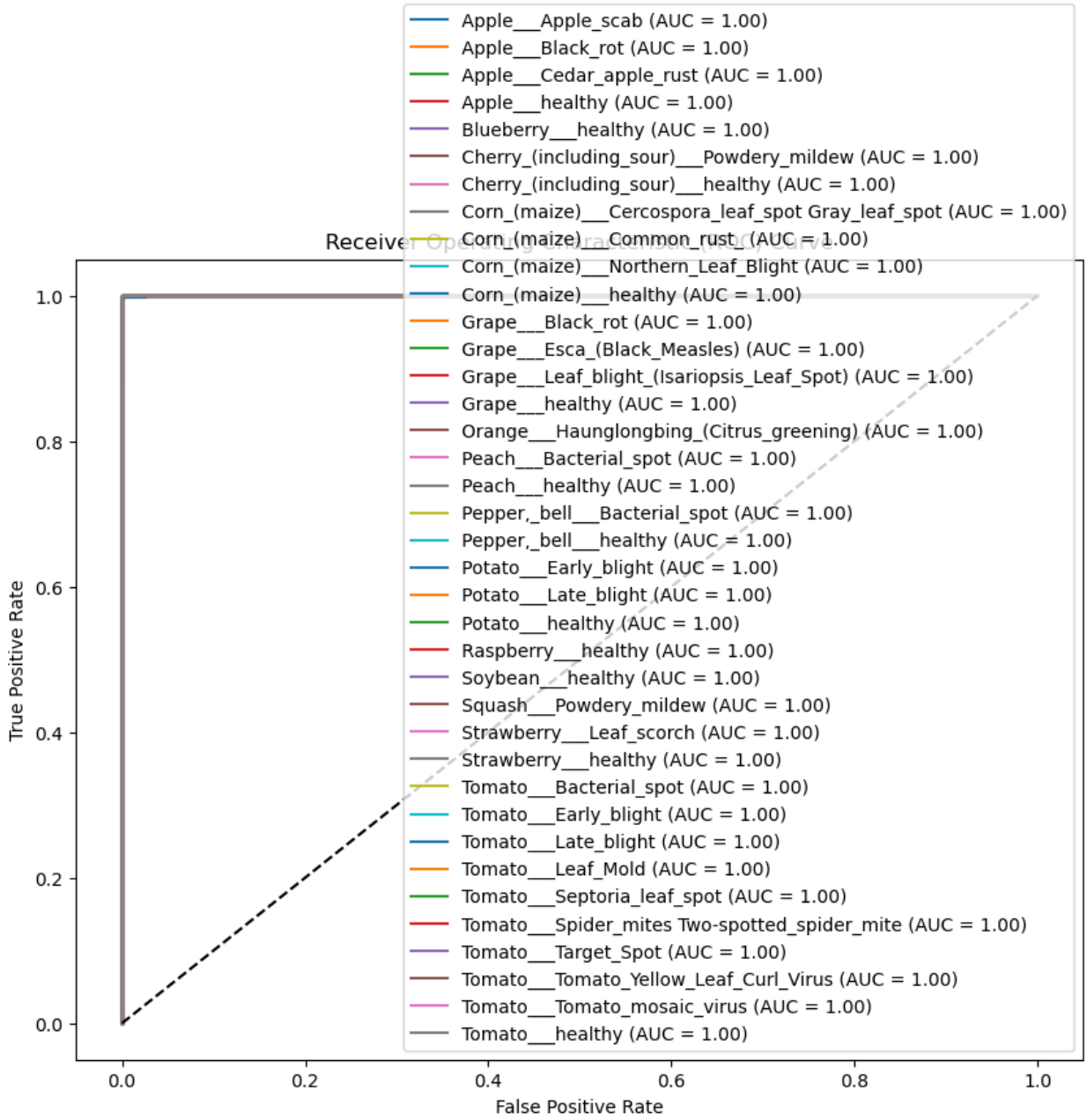


Figure 14: ROC curve for RESNET9 Model.

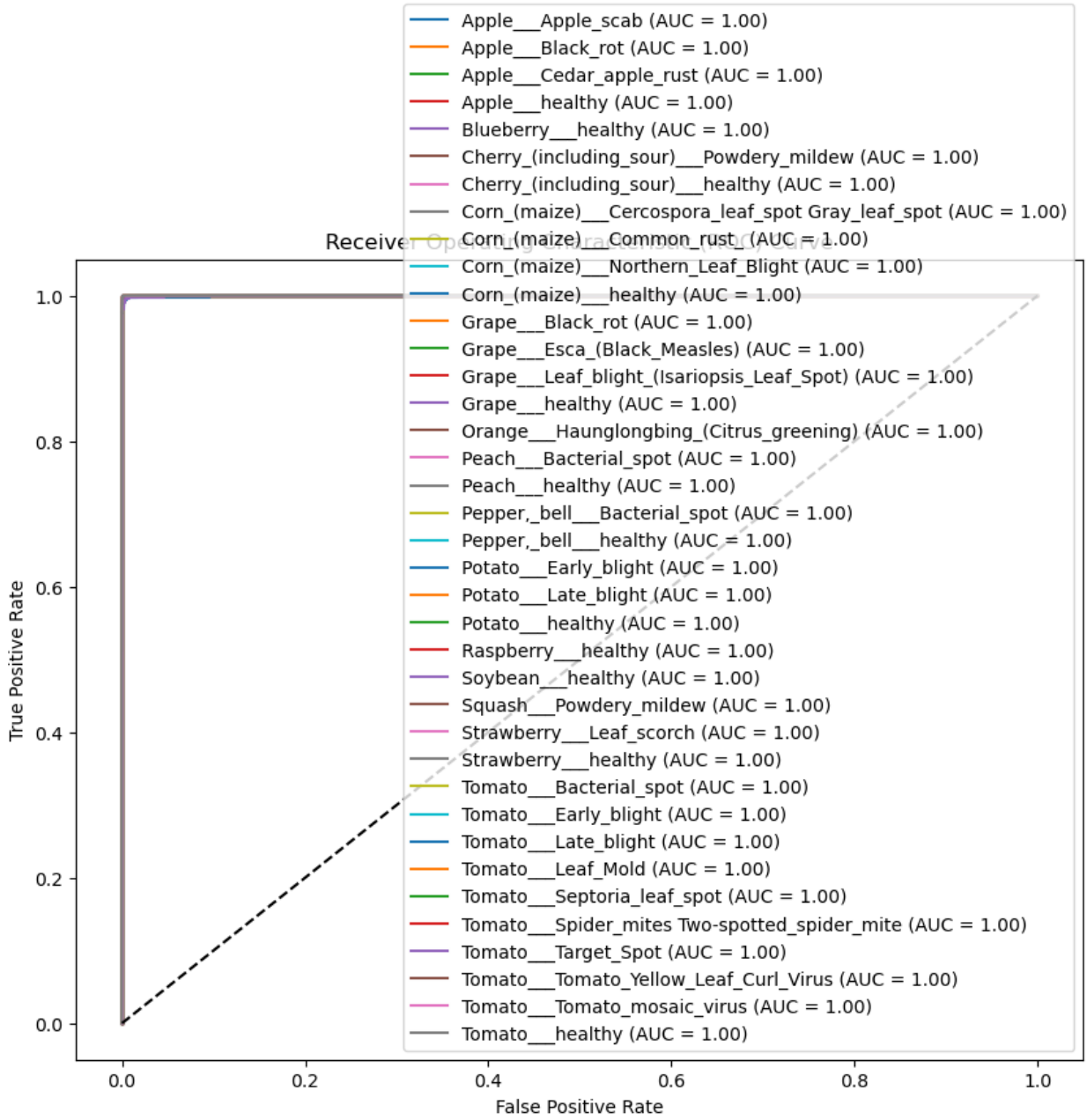


Figure 15: ROC Curve for MobileNET Model.

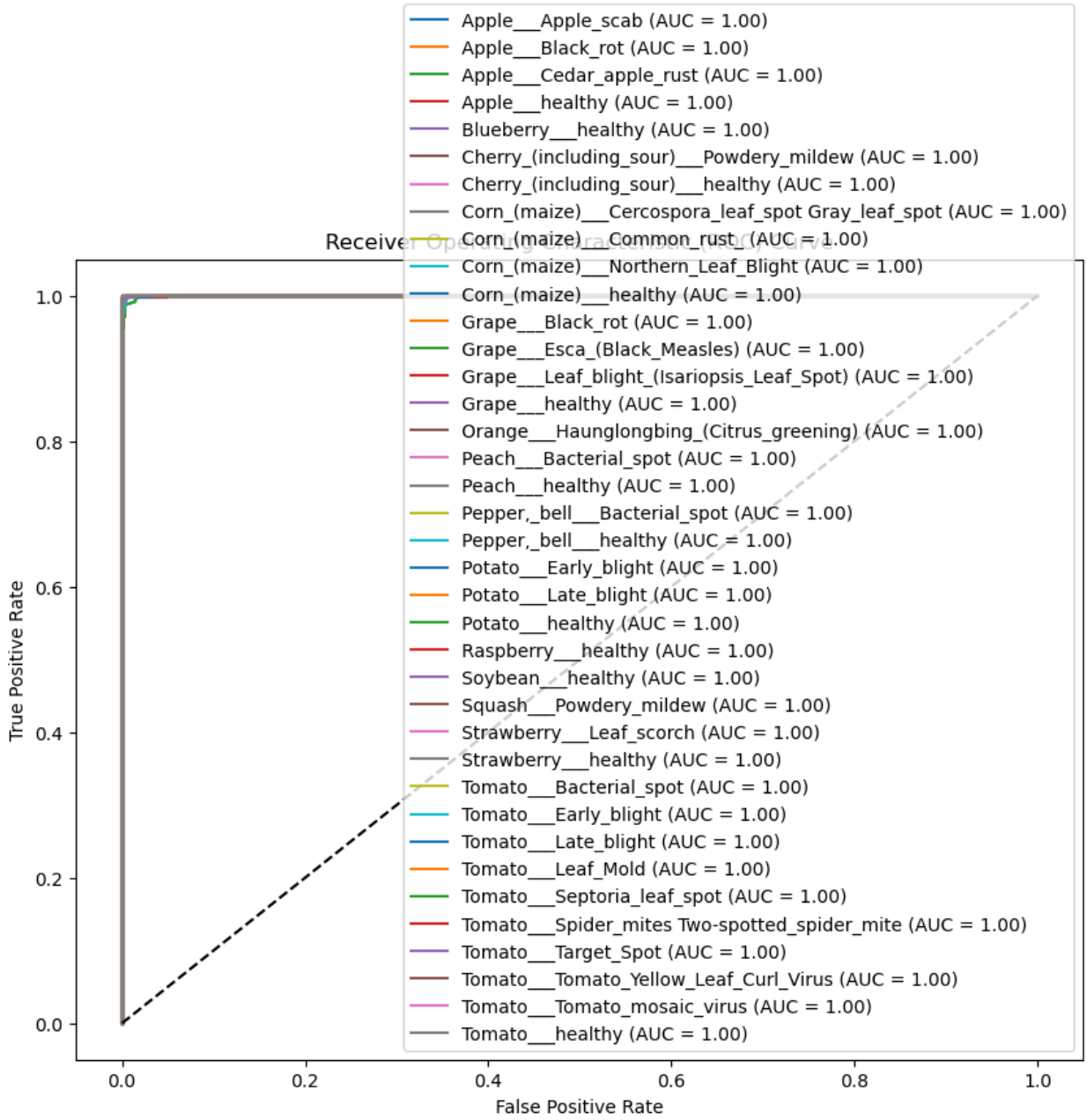


Figure 16: ROC curve for EfficienNET-B2 Model

CONCLUSION AND DISCUSSION

A residual neural network called RESNET9 was proposed for the identification of various plant diseases from images. The custom model was compared with two pretrained models, MobileNet and EfficientNet-b2 to further examine the benefit of transfer learning.

After 30 epochs each, the ResNET9 model achieved the highest accuracy of 99.83%, followed closely by MobileNet with 99.35% accuracy and EfficientNet with 99.00%.

Further evaluation using stated metrics across all models suggest that the RESNET9 model is very effective in the classification of plant diseases.

For future works, investigations could focus on exploring ensemble approaches or fine-tuning techniques to potentially enhance the models' performance even further.

Further research could focus on understanding the specific features or regions of interest that contribute most to the models' discriminative ability, potentially enhancing interpretability and guiding future efforts in disease detection and classification.

REFERENCES

Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton (2017) 'ImageNet classification with deep convolutional neural networks. Commun.', Available at: https://doi.org/10.1145/3065386.

Argüeso, D., Picon, A., Irusta, U., Medela, A., San-Emeterio, M.G., Bereciartua, A. and Alvarez-Gila, A. (2020) 'Few-Shot Learning approach for plant disease classification using images taken in the field', Computers and Electronics in Agriculture, 175, pp. 105542 Available at: https://doi.org/10.1016/j.compag.2020.105542.

Atila, Ü, Uçar, M., Akyol, K. and Uçar, E. (2021) 'Plant leaf disease classification using EfficientNet deep learning model', Ecological Informatics, 61, pp. 101182 Available at: https://doi.org/10.1016/j.ecoinf.2020.101182.

Brahimi, M., Arsenovic, M., Laraba, S., Sladojevic, S., Boukhalfa, K. and Moussaoui, A. (2018) 'Deep learning for plant diseases: detection and saliency map visualisation', Human and Machine Learning: Visible, Explainable, Trustworthy and Transparent, , pp. 93-117.

Carretero, R., Serrago, R.A., Bancal, M.O., Perelló, A.E. and Miralles, D.J. (2010) 'Absorbed radiation and radiation use efficiency as affected by foliar diseases in relation to their vertical position into the canopy in wheat', Field Crops Research, 116(1-2), pp. 184-195.

Cruz, A.C., Luvisi, A., De Bellis, L. and Ampatzidis, Y. (2017) Vision-based plant disease detection system using transfer and deep learning. American Society of Agricultural and Biological Engineers, pp. 1.

Denil, M., Shakibi, B., Dinh, L., Ranzato, M. and De Freitas, N. (2013) 'Predicting parameters in deep learning', Advances in neural information processing systems, 26.

F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong and Q. He (2021) A Comprehensive Survey on Transfer Learning, pp. 43-76.

Fang, W., Ding, Y., Zhang, F. and Sheng, V.S. (2019) 'DOG: A new background removal for object recognition from images', Neurocomputing, 361, pp. 85-91.

Ferentinos, K.P. (2018) 'Deep learning models for plant disease detection and diagnosis', Computers and Electronics in Agriculture, 145, pp. 311-318.

Fujita, E., Kawasaki, Y., Uga, H., Kagiwada, S. and Iyatomi, H. (2016) Basic investigation on a robust and practical plant diagnostic system. IEEE, pp. 989.

Glegoła, W., Karpus, A. and Przybyłek, A. (2021) 'MobileNet family tailored for Raspberry Pi', Procedia Computer Science, 192, pp. 2249-2258 Available at: https://doi.org/10.1016/j.procs.2021.08.238.

Gregory, P.J., Johnson, S.N., Newton, A.C. and Ingram, J.S. (2009) 'Integrating pests and pathogens into the climate change/food security debate', Journal of experimental botany, 60(10), pp. 2827-2838.

Hall, D., McCool, C., Dayoub, F., Sunderhauf, N. and Upcroft, B. (2015a) Evaluation of features for leaf classification in challenging conditions. IEEE, pp. 797.

Hall, D., McCool, C., Dayoub, F., Sunderhauf, N. and Upcroft, B. (2015b) Evaluation of features for leaf classification in challenging conditions. IEEE, pp. 797.

He, K., Zhang, X., Ren, S. and Sun, J. (2016a) Deep residual learning for image recognition. pp. 770.

He, K., Zhang, X., Ren, S. and Sun, J. (2016b) Deep residual learning for image recognition. pp. 770.

Itzhaky, Y., Farjon, G., Khoroshevsky, F., Shpigler, A. and Bar-Hillel, A. (2018) Leaf counting: Multiple scale regression and detection using deep CNNs. Newcastle, .

Kaggle (2016) New Plant Diseases Dataset. Available at: https://www.kaggle.com/datasets/vipoooool/new-plant-diseases-dataset (Accessed: .

Kamilaris, A. and Prenafeta-Boldú, F.X. (2018) 'Deep learning in agriculture: A survey', Computers and Electronics in Agriculture, 147, pp. 70-90.

Kerkech, M., Hafiane, A. and Canals, R. (2018) 'Deep leaning approach with colorimetric spaces and vegetation indices for vine diseases detection in UAV images', Computers and Electronics in Agriculture, 155, pp. 237-243.

L. Ou and K. Zhu (2022) Identification Algorithm of Diseased Leaves based on MobileNet Model. pp. 318.

McCulloch, W.S. and Pitts, W. (1943) 'A logical calculus of the ideas immanent in nervous activity', The Bulletin of mathematical biophysics, 5, pp. 115-133.

Muhammad Hammad Saleem, Johan Potgieter and Khalid Mahmood Arif (2019) 'Plants | Free Full-Text | Plant Disease Detection and Classification by Deep Learning', Muhammad Hammad Saleem 1, 2 andKhalid Mahmood Arif, Available at: https://www.mdpi.com/2223-7747/8/11/468#B43-plants-08-00468.

Picon, A., Alvarez-Gila, A., Seitz, M., Ortiz-Barredo, A., Echazarra, J. and Johannes, A. (2019) 'Deep convolutional neural networks for mobile capture device-based crop disease classification in the wild', Computers and Electronics in Agriculture, 161, pp. 280-290.

Savary, S., Ficke, A., Aubertot, J. and Hollier, C. (2012) 'Crop losses due to diseases and their implications for global food production losses and food security', Food security, 4(4), pp. 519-537.

Savary, S. and Willocquet, L. (2020) 'Modeling the impact of crop diseases on global food security', Annual Review of Phytopathology, 58, pp. 313-341.

Signoroni, A., Savardi, M., Baronio, A. and Benini, S. (2019) 'Deep learning meets hyperspectral image analysis: A multidisciplinary review', Journal of Imaging, 5(5), pp. 52.

Sun, Y., Liu, Y., Wang, G. and Zhang, H. (2017) 'Deep learning for plant identification in natural environment', Computational intelligence and neuroscience, 2017.

Tan, M. and Le, Q. (2019) Efficientnet: Rethinking model scaling for convolutional neural networks. PMLR, pp. 6105.

Türkoğlu, M. and Hanbay, D. (2019) 'Plant disease and pest detection using deep learning-based features', Turkish Journal of Electrical Engineering and Computer Sciences, 27(3), pp. 1636-1651.