PLANT DISEASE DETECTION MODEL USING DEEP LEARNING

AUGUST 2024

MORINGA SCHOOL DATA SCIENCE: PART TIME 6
Group 3.

Contents

TOMATO DISEASE DETECTION MODEL USING DEEP LEARNING.

1. OVERVIEW

Tomatoes are one of the most important crops in Kenya, both economically and nutritionally. They are a staple in the diet of many Kenyans and a significant source of income for smallholder farmers. However, tomato production in Kenya faces numerous challenges, among which diseases are the most important. Tomato plants are susceptible to various diseases caused by fungi, bacteria, viruses, and pests. These diseases can lead to substantial yield losses, reduced produce quality, and increased production costs due to the need for pesticides and other control measures.

In Kenya, the impact of tomato diseases is exacerbated by several factors:

1. Limited Access to Knowledge and Resources: Many smallholder farmers lack access to information on disease identification and management practices. Extension services are often inadequate, leaving farmers without the necessary support to effectively combat tomato diseases.
2. Climate and Environmental Conditions: Kenya's diverse climate and environmental conditions can create favourable conditions for the spread and persistence of various tomato diseases. Changes in weather patterns due to climate change further complicate disease management.
3. Economic Constraints: The financial constraints of smallholder farmers limit their ability to invest in disease management solutions such as resistant seed varieties, appropriate chemicals, and modern farming techniques.
4. Inadequate Diagnostic Tools: Traditional methods of disease diagnosis are often slow, inaccurate, and labour-intensive. There is a need for rapid, accurate, and cost-effective diagnostic tools that can be easily used by farmers in the field.

The consequences of unchecked tomato diseases are severe. Yield losses can range from 20% to 100%, depending on the type and severity of the disease. This not only affects the income of farmers but also threatens food security and nutrition for millions of Kenyans. Addressing tomato diseases effectively is therefore crucial for enhancing agricultural productivity, improving the livelihoods of smallholder farmers, and ensuring food security in Kenya.

2 . BUSINESS PROBLEM

a. Business Problem Statement:

Agriculture is a cornerstone of Kenya's economy, with a majority of the population relying on it for their livelihood. However, plant diseases pose a significant threat to crop yields, leading to economic hardships for farmers and contributing to food insecurity. Traditional methods of disease detection are often slow, costly, and require expert knowledge. An automated plant disease detection system using image data can empower Kenyan farmers to quickly and accurately identify diseases, take timely actions, and improve crop productivity.

b. Project Objectives:

The goal of this project is to develop an automated system for detecting plant diseases in Kenya using image recognition technology. This system will enable farmers to identify various plant diseases from images of leaves, providing accurate and timely diagnostic information to help mitigate crop losses.

Project specific objectives are

I. Data Collection and Pre-processing. Gather and pre-process a diverse dataset of tomato leaf images.
II. Model Development. Build and train deep learning models for disease classification.
III. Model Evaluation. Assess the model's performance using metrics such as accuracy.
IV. Deployment of our model on a web-based application.

3. DATA UNDERSTANDING

a. Data Source:

Our dataset was downloaded from a Kaggle repository authored by Plant Village. PlantVillage is an organization hosted by Penn State University, is a research initiative focused on enhancing agricultural resilience and sustainability, particularly in the context of climate change. The platform employs advanced technologies such as artificial intelligence and data analytics to address various agricultural challenges including plant disease, which will be the focus of this project. For this project we shall focus on Tomato Leaves.

The link to the repository is here:
https://www.kaggle.com/datasets/emmarex/plantdisease?select=PlantVillage

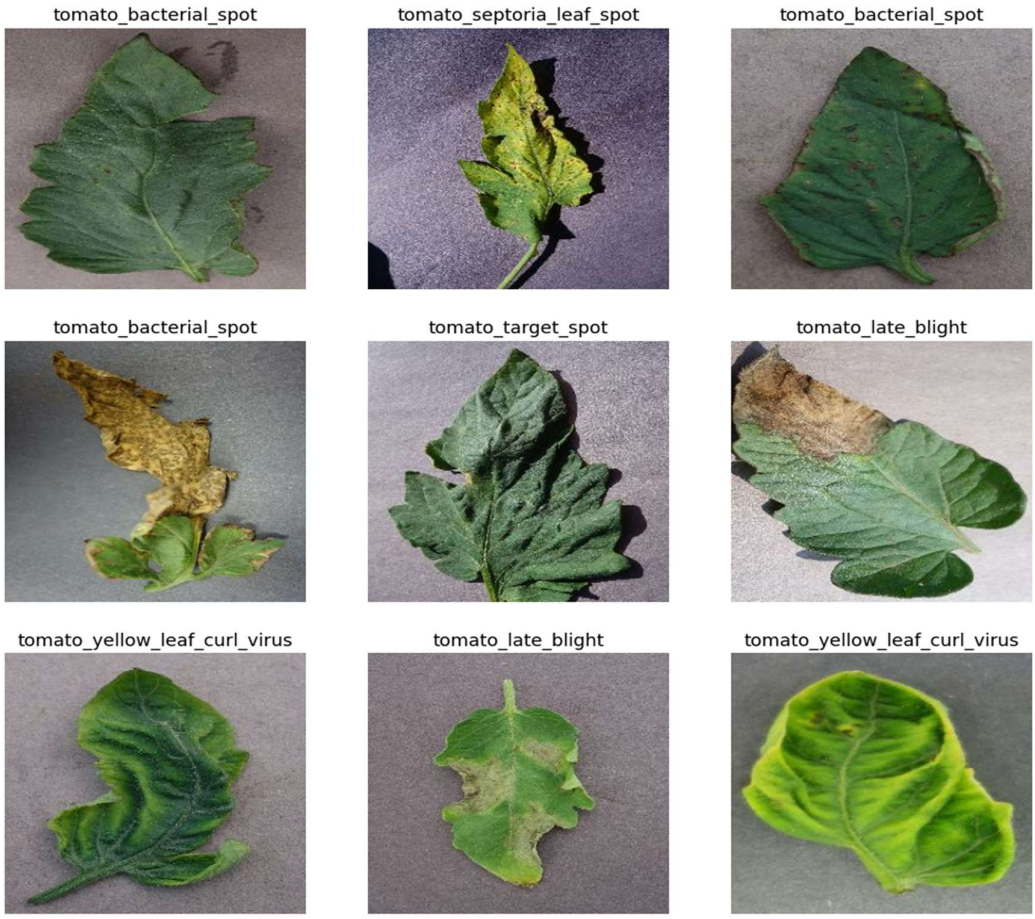
Dataset Author Website: https://plantvillage.psu.edu/

b. Data Description:

a) Our data contains 16,031 images of 10 classes i.e., 9 classes of leaves indicating 9 different diseases and 1 class of healthy leaves. The diseases covered by the images are

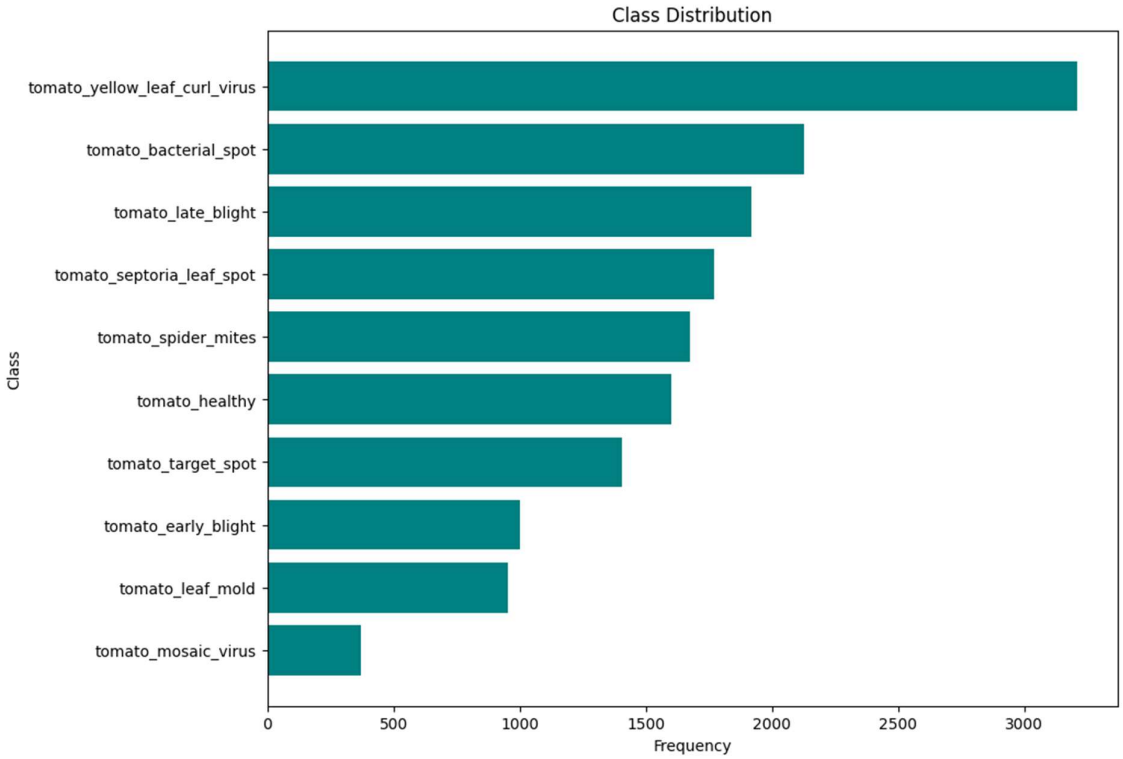
- Bacterial Spot
- Early Blight
- Late Blight
- Leaf Mold
- Septoria Leaf Spot
- Spider Mites
- Yellow Leaf Curl Virus
- Mosaic Virus

Here are samples of leaf images indicating presence of a disease



b. The class Distribution in ascending order is as follows

Mosaic Virus: 373, Leaf Mold: 952, Early Blight 1000, Target Spot: 1404, Healthy: 1601, Spider Mites: 1676, Septoria Leaf Spot: 1771, Late Blight: 1919, Bacterial Spot: 2127 And Yellow Leaf Curl Virus: 3208



4. METRICS OF SUCCESS

The metric of success is to create a model that achieves a classification accuracy of 90% and above.

5. DATA PREPARATION

Data Cleaning:

- Renamed one of the labels from Tomato_Spider_mites_Two_spotted_spider_mite to Tomato_Spider_mites. Futher, to ensure consistency in the labeling format, a single underscore was included between each word.
- Checked for any corrupt images in our data set. One corrupt image was found on the 1st run and removed.

Train-Test Split

- The dataset was split into 80% training, 10% validation, and 10% test sets using a custom data splitting function. This method ensures that the splits are mutually exclusive and collectively exhaustive, meaning that all data points

are allocated to one of the three sets without overlap, and all data points are included in the splits.

- Our models will be trained by the augmented train dataset, validated using the validation dataset and tested using the unseen test dataset

Data Augmentation:

- Data was applied using a function that takes an image and its corresponding label as input and applies augmentation transformation to the image and returns the augmented image along with its label. Further, we applied these transformations to the train dataset using the *.map.*
- Arguments aapplied are random rotations (0.2), zoom, and horizontal and vertical flips to augment the training data.
- The purpose of augmentation is to improve model generalization, increase the effectiveness of the training data size, reduces underfitting and simplifies our pipeline.

6. MODELLING

Model Training and Selection:

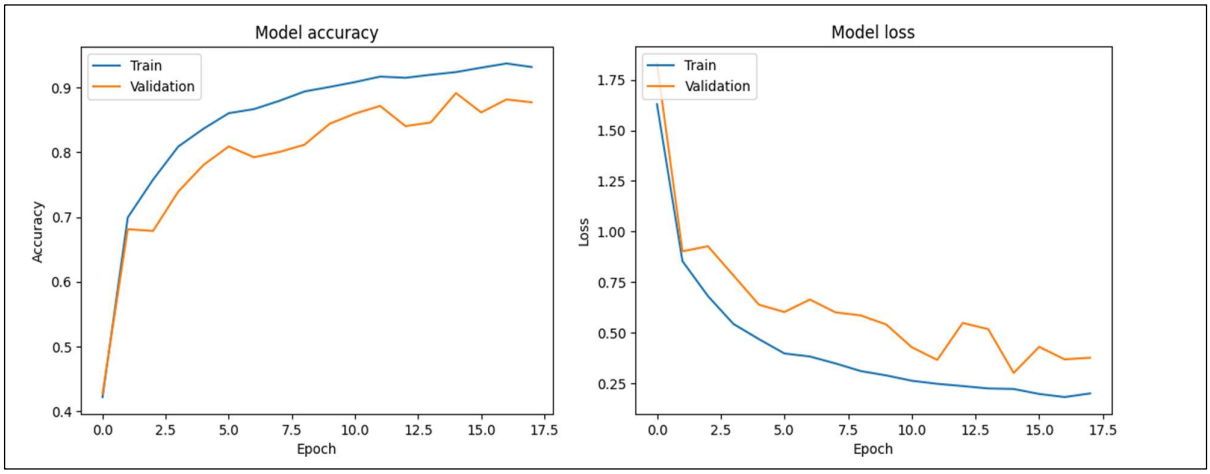
- Tested various models including Simple CNN, VGG19, ResNet50, EfficientNetB0 and MobileNetV2 to identify the best performing architecture.
- The models were trained using the categorical crossentropy loss function. This is because we are dealing with a multiclass categorization model
- Early stopping with a patience of 3 was set. The monitoring metric was the validation accuracy.
- The training and validation history of all the models was saved in *.json* format, and the trained models saved in the .h5 format

Here is a breakdown of each of the models that were tested:

Simple CNN

Model Training:

A Convolutional Neural Network (CNN) model is built, consisting of several layers. Convolutional layers with ReLU activation followed by max-pooling layers. Dropout layers to prevent overfitting. Flatten layer to convert the 2D matrix to a 1D vector. Dense layers for the final classification using SoftMax activation.



Model Evaluation:

Achieved a training accuracy of 93.20% and a validation accuracy of 87.75%.

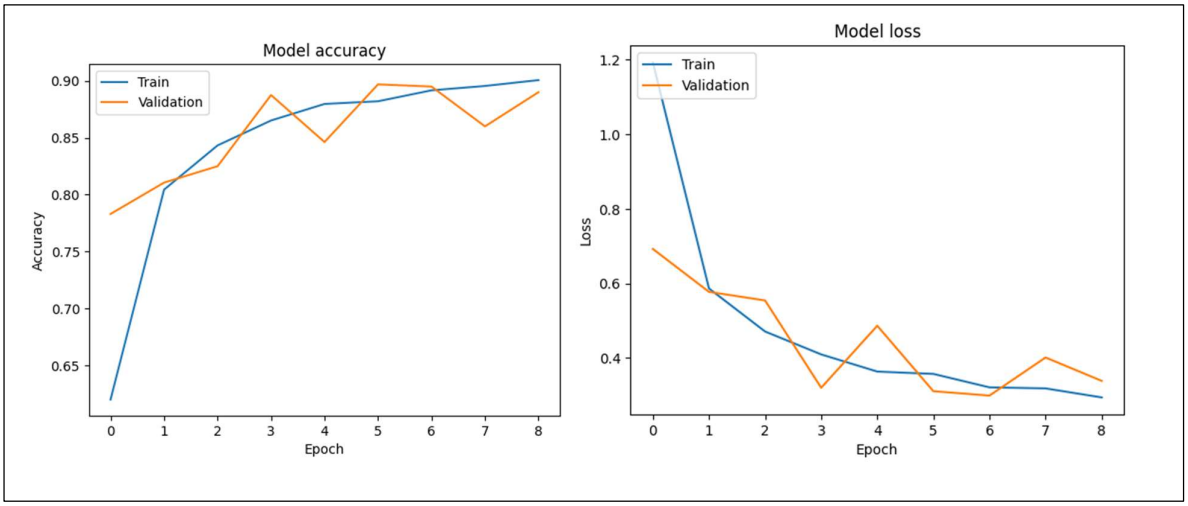
The model's performance has improved significantly from the first epoch to the 18th epoch. Initially, both the training and validation losses were high with moderate accuracy, but over time, the training loss decreased and accuracy increased, indicating better model performance and generalization.

VGG19

Model Training:

A pre-trained VGG19 model was used to build a custom image classification model. The VGG19 model's weights are frozen to retain its pre-trained features.

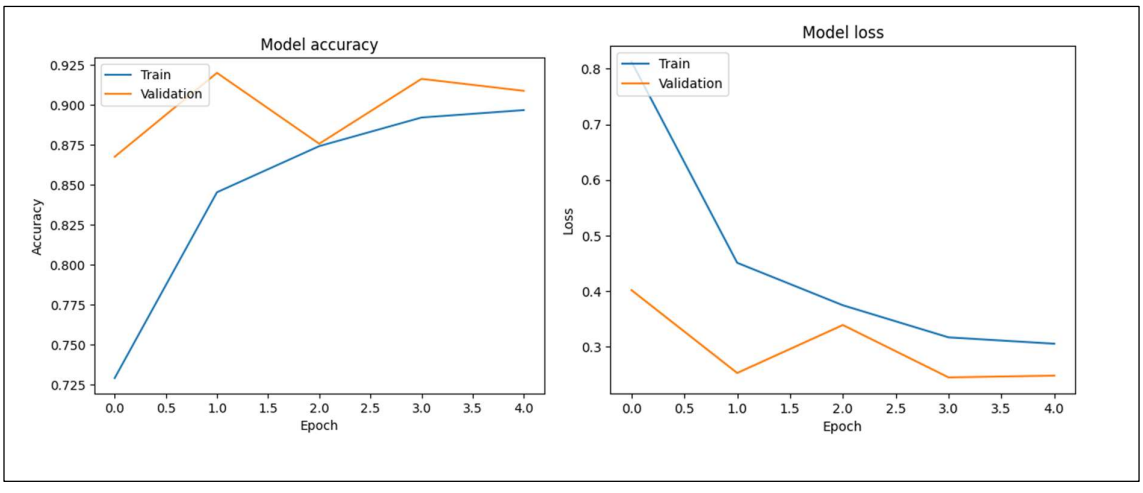
Convolutional layers with ReLU activation followed by max-pooling layers. Dropout layers to prevent overfitting. Flatten layer to convert the 2D matrix to a 1D vector. Dense layers for the final classification using SoftMax activation.



ResNet50

Model Training:

The ResNet50 model's weights are frozen to retain its pre-trained features, and a custom classifier consisting of a global average pooling layer, a dense layer, a dropout layer, and an output layer is added on top. The model's architecture and parameters are then summarized. This approach helps to quickly fine-tune a powerful pre-trained model for a specific classification task with minimal additional training.



Model Evaluation:

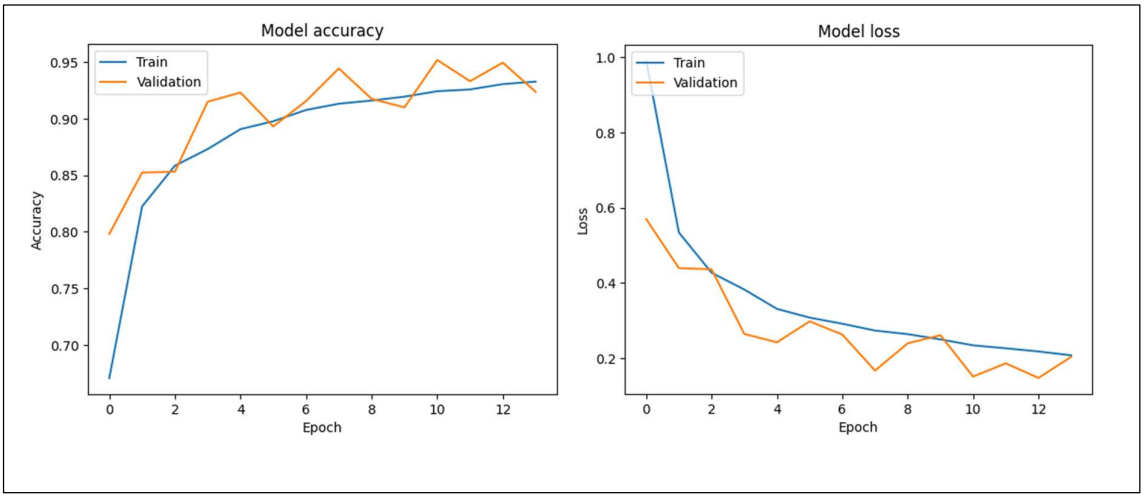
The model shows good learning capabilities, with consistent improvements in both training and validation accuracy. The training loss started at 0.8127 and decreased to 0.3053 by epoch 5. The validation loss started at 0.4017 and fluctuated, ending at 0.2478. The training accuracy improved from 72.90% to 89.67% over the 5 epochs. The validation accuracy started at 86.75% and varied slightly, ending at 90.87%.

The decrease in training loss and general decrease in validation loss further support the model's effectiveness in learning from the data.

EfficientNetB0

Model Training:

The EfficientNetB0 model's weights are frozen to retain its pre-trained features, and a custom classifier consisting of pre-processing steps, a global average pooling layer, several dense layers, a dropout layer, and an output layer is added on top. The model's architecture and parameters are then summarized. This approach helps to quickly fine-tune a powerful pre-trained model for a specific classification task with minimal additional training.



Model Evaluation:

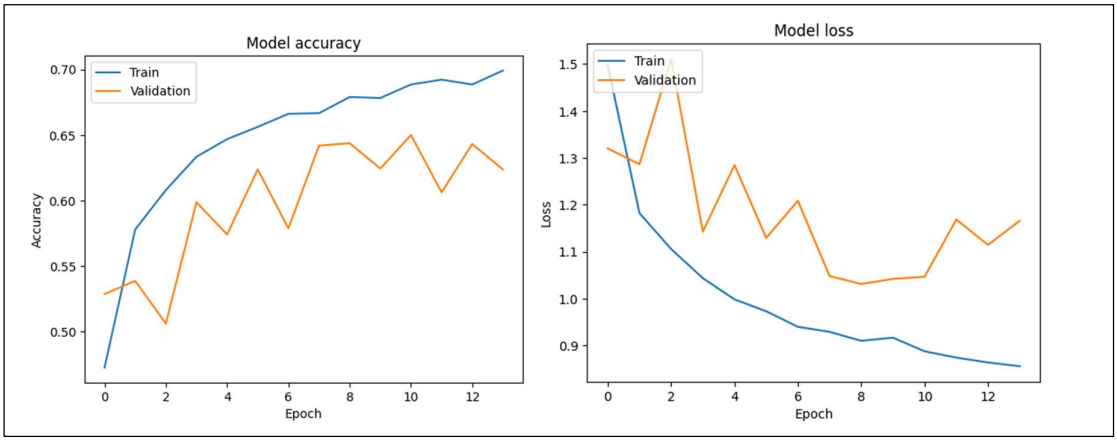
The training loss started at 0.9908 and decreased to 0.2071 by epoch 14. The training accuracy improved from 67.07% to 93.28% over the same period. Validation accuracy increased from 79.81% to 92.37%.

Generally, the model shows good learning and generalization capabilities, with steady improvement in both accuracy and loss metrics.

MobileNetV2

Model Training:

Trained the MobileNetV2 model using categorical crossentropy loss function



Model Evaluation:

The model shows a gradual improvement in training accuracy, rising from 47.27% to 69.91%, but validation accuracy fluctuates between 50% and 65% with inconsistent loss values. This suggests that while the model is learning better on the training data, it struggles to generalize effectively to the validation set, indicating potential overfitting or issues with model performance on unseen data.

7. EVALUATION

The Model was tested using the test dataset to assess their accuracy on unseen data. Below was the summary of the model performances:

- Test accuracy for EfficientNetB0: 95.15 %
- Test accuracy for CNN: 91.42 %
- Test accuracy for ResNet50: 90.78 %
- Test accuracy for VGG19: 88.54 %
- Test accuracy for MobileNetV2: 64.15 %

Based on the test accuracy performance, we went ahead to further tune the ResNet50, CNN and EfficientNetB0 to further examine if we can get better performance.

8. MODEL HYPER PARAMETER TUNING.

- We used a tool called Weights & Biases (wandb) to track the training of CNN and ResNet50 models which had the highest accuracies and generate report

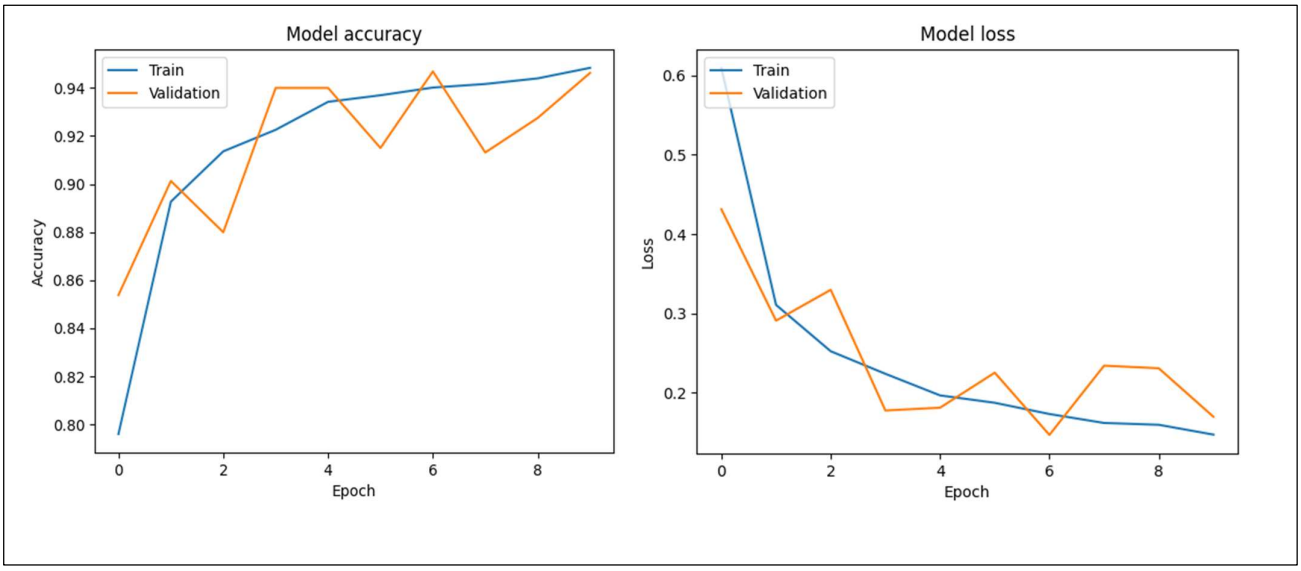
- Wandb visualizes the results of different hyperparameter combinations to identify which configuration leads to the best validation accuracy.

- Once we obtained the optimized parameters, the model was trained, the history saved in .json and the trained model saved in .h5 format

1. ResNet50

Model Training and Evaluation

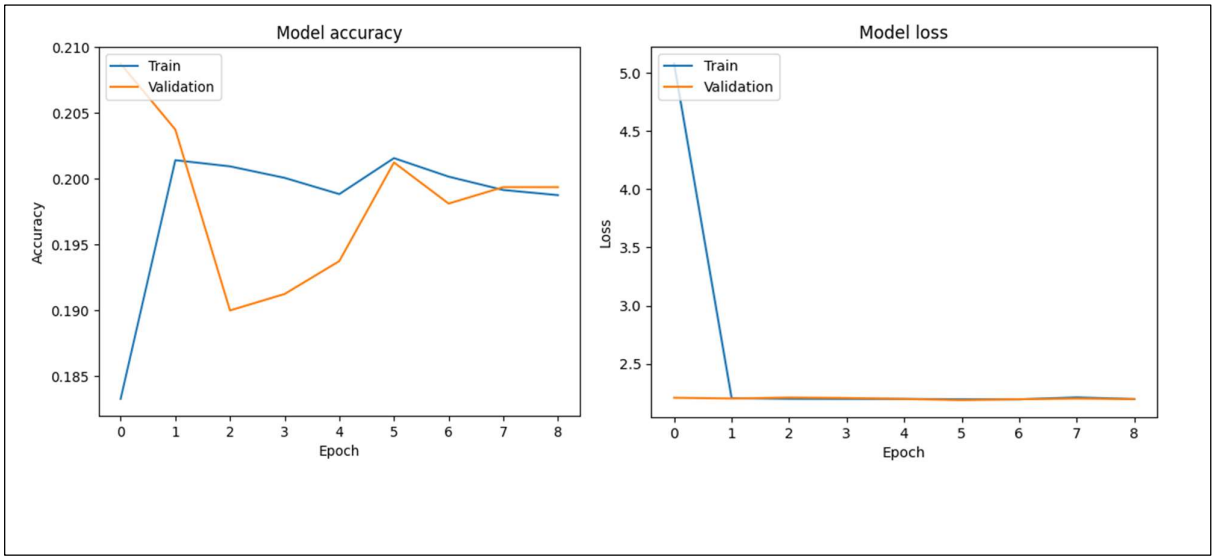
- Accuracy: 0.94836, meaning the model correctly predicted about 94.8% of the training data.
- Best Epoch: Epoch 6 was the best in terms of validation loss.
- Best Validation Loss: 0.14682, indicating the lowest error on the validation set at the best epoch.
- Total Epochs: 9 epochs were run in total.
- Final Loss: 0.14717, showing the training error at the end.
- Validation Accuracy: 0.94625, the accuracy on the validation set, showing good generalization.
- Validation Loss: 0.16966, the error on the validation set at the end.



3. EfficientNetB0

Model Training and Evaluation

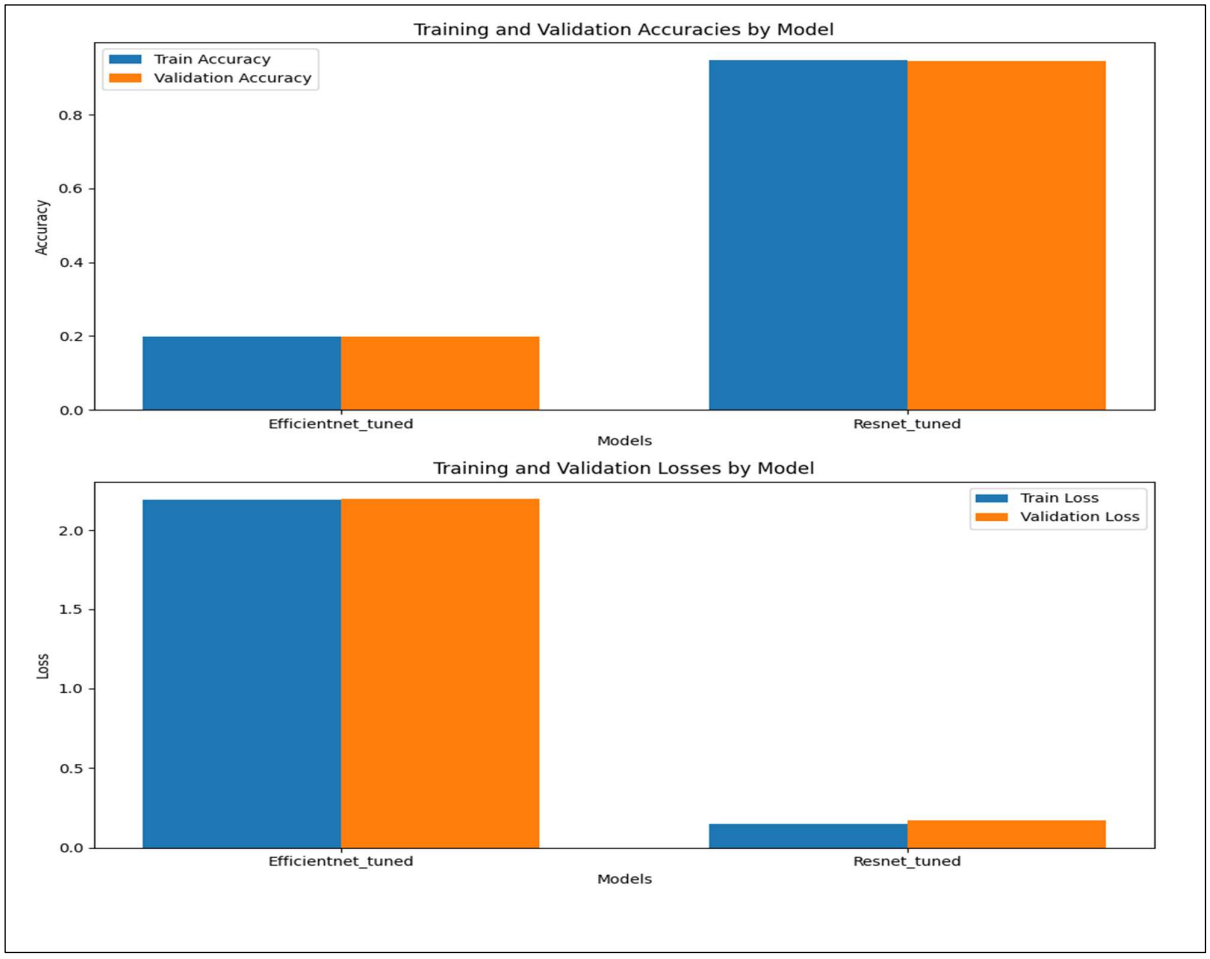
- The loss value of 2.2025 indicates how well the model's predictions match the actual labels. The high loss value generally implies poor model performance.
- An accuracy of 20.47% on the test set is relatively low, indicating that the model is only correctly predicting the class about 20% of the time.



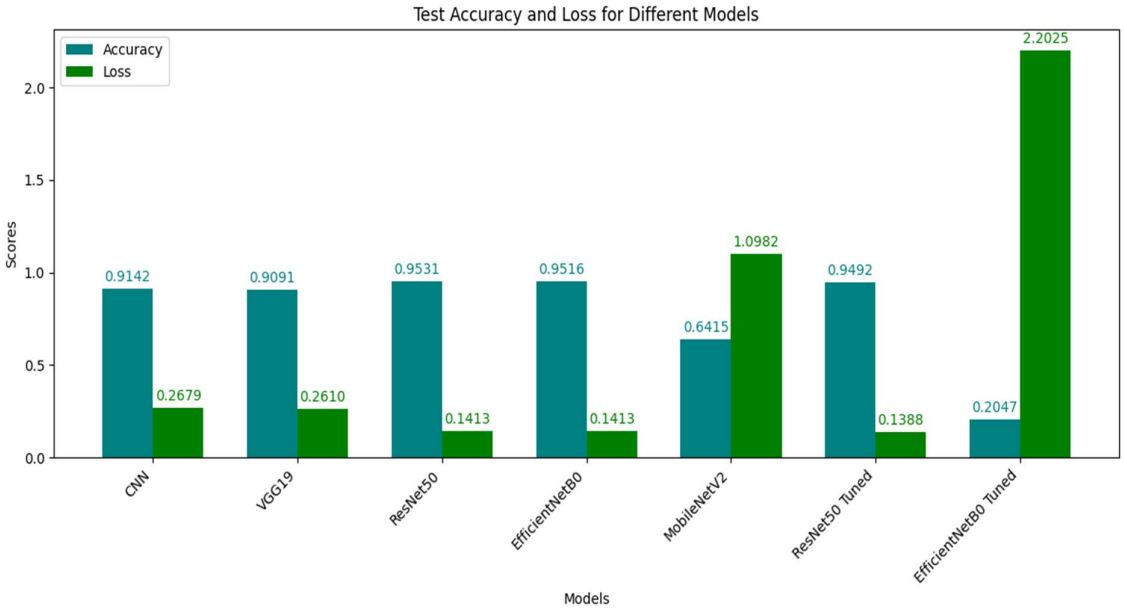
- Low Accuracy: Both training and validation accuracies are very low (~20%), suggesting that the model is not performing well. This indicates that the model is struggling to learn the patterns in the data.
- High Loss: The high training and validation loss values further indicate that the model's predictions are far from the actual values.
- Fluctuations in Validation Metrics: The fluctuations in validation accuracy and loss suggest that the model is inconsistent in its performance on the validation set, potentially due to overfitting or noise in the data.

9. MODEL SELECTION.

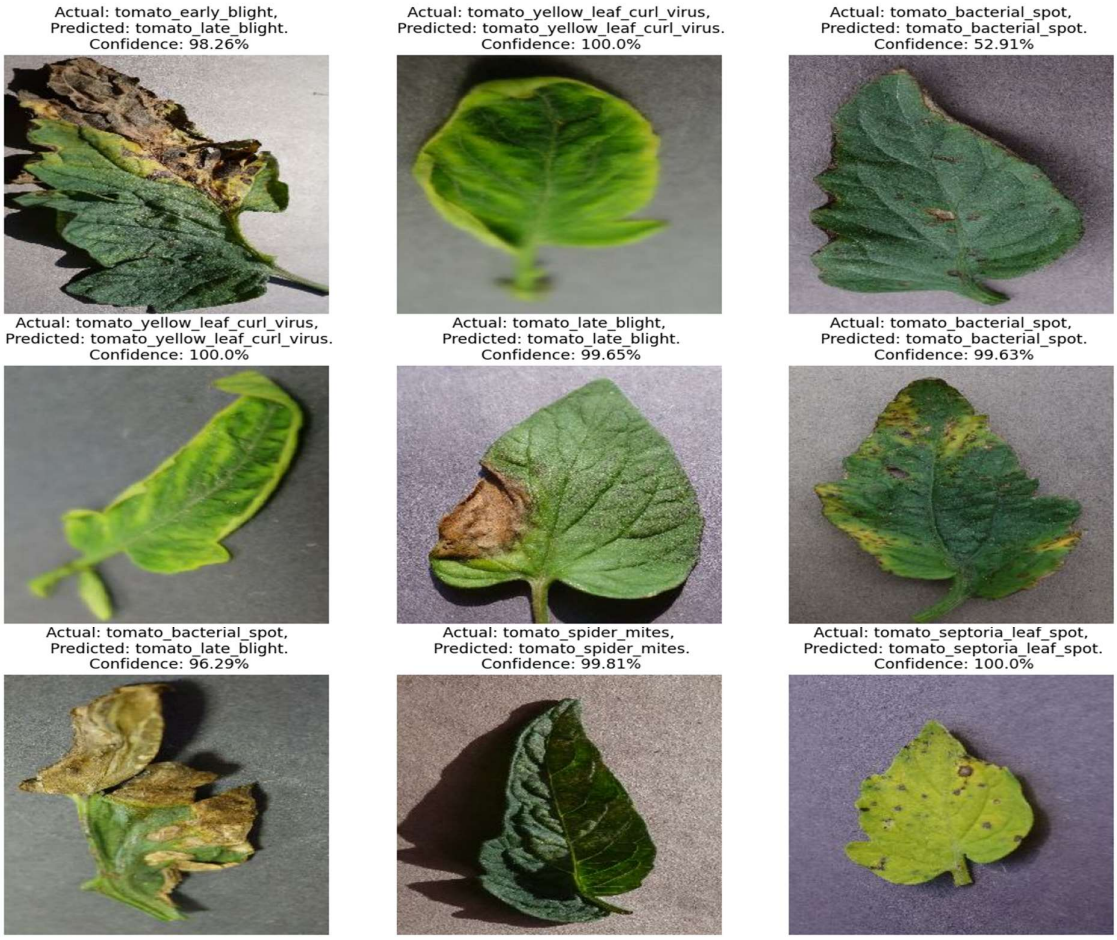
Our Resnet50 Model was selected as the best model as it performance is consistent with the results we are looking for in a model. Despite the EfficientNeB0 model having a higher accuracy before, further tuning led to a loss in accuracy and high validation loss as show in the graphs below. The reason for this was yet to be established, and will form part of future research areas in the project.



Below is a graphical summary of the model performance after tuning.



To further evaluate the performance of our best model, ResNet50, we used it to make predictions using the test data and check the confidence level of the model's prediction.



Finally, we obtained a random from the internet to run a prediction using our model as seen below;

```
[ ]
    img_path='./tomato_late_blight.jpeg'#loading a random image for prdecition using the the best model
    image = tf.keras.preprocessing.image.load_img(img_path, target_size=(256, 256))
    img_array = tf.keras.preprocessing.image.img_to_array(image)
    img_array = tf.expand_dims(img_array, 0)

    predictions = model_restune2.predict(img_array)
    predicted_class = class_names[np.argmax(predictions[0])]
    confidence = round(100 * (np.max(predictions[0])), 2)

    print(f"Predicted: {predicted_class}.\n Confidence: {confidence}%")
```

```
1/1 [==============================] - 0s 82ms/step
Predicted: tomato_late_blight.
 Confidence: 99.95%
```

- The model correctly predicted the image as tomato_late_blight with 99.95% confidence.

9. DEPLOYMENT

Deployment Plan:

The model was deployed in Flask App Model Deployment using ngrok, allowing farmers to upload images via a mobile app for real-time disease diagnosis.

Deployment Link: https://python-evident-ram.ngrok-free.app/

Performance Monitoring:

Will set up logging and monitoring to track model performance and retrain the model periodically with new data.

10. CONCLUSION AND FUTURE WORK

Summary:

Successfully developed an image classification model for tomato leaf disease detection with an accuracy of 94%.

Additionally, creation of a web-based app that allows farmers to upload leaf images and a prediction on the health status of the plant is displayed.

This model can assist farmers in early disease detection and management.

Challenges:

- One challenge was obtaining high-quality labelled images for training.
- Computational limitations in training our model.

Future Work:

- Creating a mobile app (on IOS and Android) that captures the leaf image and returns the prediction result of the disease if any that a plant may have.
- Future work includes expanding the dataset with more diverse images including that of different plants and diseases, integrating weather data for better predictions, and improving model accuracy through transfer learning.

11. APPENDICES

Code Listings:

- The complete code is available at: https://github.com/KeyMoney22/Plant_disease_classification-