Final Report – DenseNet121 Model

1. Introduction

In this project, a plant disease image classification model based on **DenseNet121** was built and evaluated using a subset of the **PlantVillage dataset**. DenseNet121 is a deep convolutional neural network where each layer is connected to all previous layers, which helps the model learn useful features more efficiently and improves training stability.

The goal of this project was to train a model to classify plant diseases, evaluate its performance, and understand why the model achieved high accuracy during training.

---

2. Model Architecture: DenseNet121

DenseNet121 was chosen because:

- It is **pretrained on ImageNet**, which allows transfer learning.
- Dense connections help the model learn **important visual details** in plant leaves.
- It performs well even with **limited training data**.

The original classification layer was removed, and a custom classification head was added:

- Global Average Pooling
- Fully connected layer with **L2 regularization**
- Dropout to reduce overfitting
- Softmax output layer for multi-class classification

---

3. Step-by-Step Methodology

Step 1: Data Preparation

- The PlantVillage dataset was used with **five selected classes**.
- Images were resized to a smaller size to work efficiently on **Google Colab CPU**.
- A tf.data.Dataset pipeline was created with batching and prefetching for better performance.

Step 2: Transfer Learning (Feature Extraction)

- DenseNet121 pretrained weights were loaded.
- All convolutional layers were **frozen** at first.
- Only the newly added classification head was trained.

This helped the model use already learned visual features without overfitting at the start.

Step 3: Overfitting Control

To reduce overfitting, the following techniques were used:

- Data augmentation (rotation, zoom, shifting)
- Dropout with a rate of 0.6
- L2 weight regularization
- EarlyStopping based on validation loss

Step 4: Fine-Tuning

- After initial training, the **last layers of DenseNet121** were unfrozen.
- A **very small learning rate** was used.
- The model was trained for a few more epochs.

This allowed the model to better adapt to plant disease features and improved accuracy.

Step 5: Model Saving

- The best model was saved automatically using **ModelCheckpoint**.
- This ensured evaluation was done using the model with the **lowest validation loss**.

Step 6: Evaluation

The model was evaluated using:

- Accuracy
- Precision
- Recall
- F1-score
- Confusion Matrix
- Grad-CAM for explainability

True labels were extracted from the dataset and converted from one-hot encoding to class indices to calculate the metrics correctly.

---

4. Why Training and Validation Accuracy Are Very High

The model achieved very high training and validation accuracy (around 99–100%) because:

- DenseNet121 is a **strong pretrained model**.
- Only **five classes** were used, making the task easier.
- Fine-tuning allowed the model to specialize in plant disease features.
- Regularization methods helped stabilize training.
- Training and validation data came from the **same distribution**, which can lead to very high validation accuracy.

---

5. Test Performance and Generalization

Although training and validation accuracy were very high, test accuracy was much lower. This indicates **limited generalization**, which can be explained by:

- Differences between training and test data
- Using different image types (color, grayscale, segmented)
- High model capacity compared to dataset size
- Limited computational resources (CPU-only training)

This behavior is common in deep learning and highlights the importance of careful evaluation.

---

6. Explainability Using Grad-CAM

Grad-CAM was used to visualize which parts of the images influenced the model's predictions. The heatmaps showed that the model focused mainly on **leaf areas and disease regions**, which confirms that predictions were based on meaningful visual information.

---

7. Conclusion

In this project, DenseNet121 was successfully applied to plant disease classification using transfer learning and fine-tuning. A structured training process and regularization techniques resulted in very high training and validation accuracy. However, test results showed generalization challenges, emphasizing the importance of proper dataset splitting and evaluation.

Future improvements may include:

- Using more diverse training data
- Creating a clearer separation between training, validation, and test sets
- Training on GPU for deeper fine-tuning