

ÖZYEĞİN UNIVERSITY

FACULTY OF INDUSTRIAL ENGINEERING

DEPARTMENT OF DATA SCIENCE

**CS 566**

**Fall 2024-2025**

**SENIOR PROJECT REPORT**

**Potato Leaf Disease Classification**

**Using Deep Learning Models**

**By**

**DEMBA SOW**

**Supervised By**

**Prof. Dr. Hasan Fehmi Ateş**

***POTATO LEAF DISEASE CLASSIFICATION USING DEEP LEARNING MODEL***

Demba Sow

[dastech1998@gmail.com](mailto:dastech1998@gmail.com)

# Abstract

Potato crops are crucial for global food security, making disease detection a vital task. This study focuses on classifying potato leaf diseases using deep learning models, specifically CNN, MobileNetV2, ResNet50, and DenseNet121. We conducted experiments on the Potato Disease Leaf Dataset (PLD) and evaluated models using precision, recall, F1-score, accuracy, and loss metrics.

# Introduction

Potatoes are a staple food worldwide. Early detection of diseases like early blight, late blight, and identifying healthy leaves is critical for improving yield and reducing economic losses.

Traditional methods are labor-intensive and prone to error, prompting the need for automated approaches using deep learning.

### Objectives:

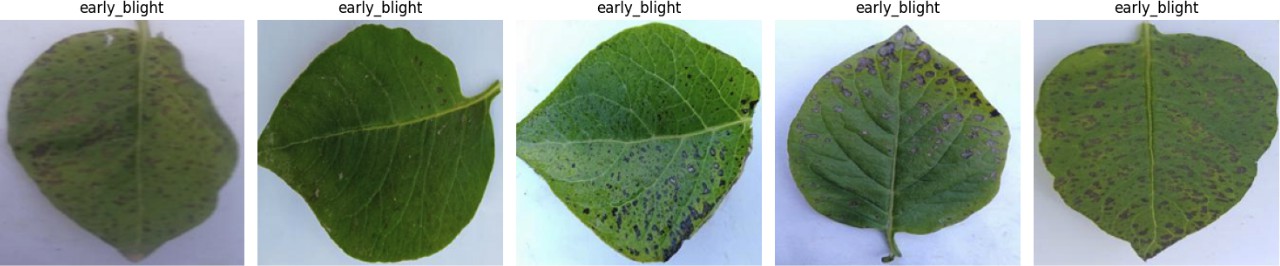
* Apply deep learning models for potato leaf disease classification.
* Compare model performances and improve them through fine-tuning.
* Evaluate classification metrics and recommend the best model

# Methodology

## Dataset Description

The Potato Disease Leaf Dataset (PLD) consists of 4,072 images collected from the Central Punjab region of Pakistan. It includes three classes: **early blight**, **late blight**, and **healthy** leaves.

* **Early Blight:** Leaves affected by fungal infections, showing brown or black lesions.



*Figure: Sample images of early blight disease*

* **Late Blight:** Leaves infected with water-soaked, rapidly spreading spots caused by a microorganism.



*Figure: Sample images of late blight disease*

* **Healthy**: Leaves with no visible signs of disease.



*Figure: Sample images of healthy disease*

## Dataset Structure:

* 4027 JPEG Images (256 x 256)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Subset** | **Early Blight** | **Late Blight** | **Healthy** | Total by Subset |
| Training | 1303 | 1132 | 816 | **3251** |
| Validation | 163 | 151 | 102 | **416** |
| Testing | 162 | 141 | 102 | **405** |
| Total by Class | **1628** | **1424** | **1020** | **4072** |

*Table: Original dataset structure*

## Data Preprocessing:

* Corrupted Image Check: No corrupted images were found.
* Image resizing: Resized to 224x224 pixels.
* Normalization: Pixel values scaled to a 0-1 range.
* Data augmentation: Rotation, zooming, flipping, and brightness adjustments applied.

## Deep Learning Models

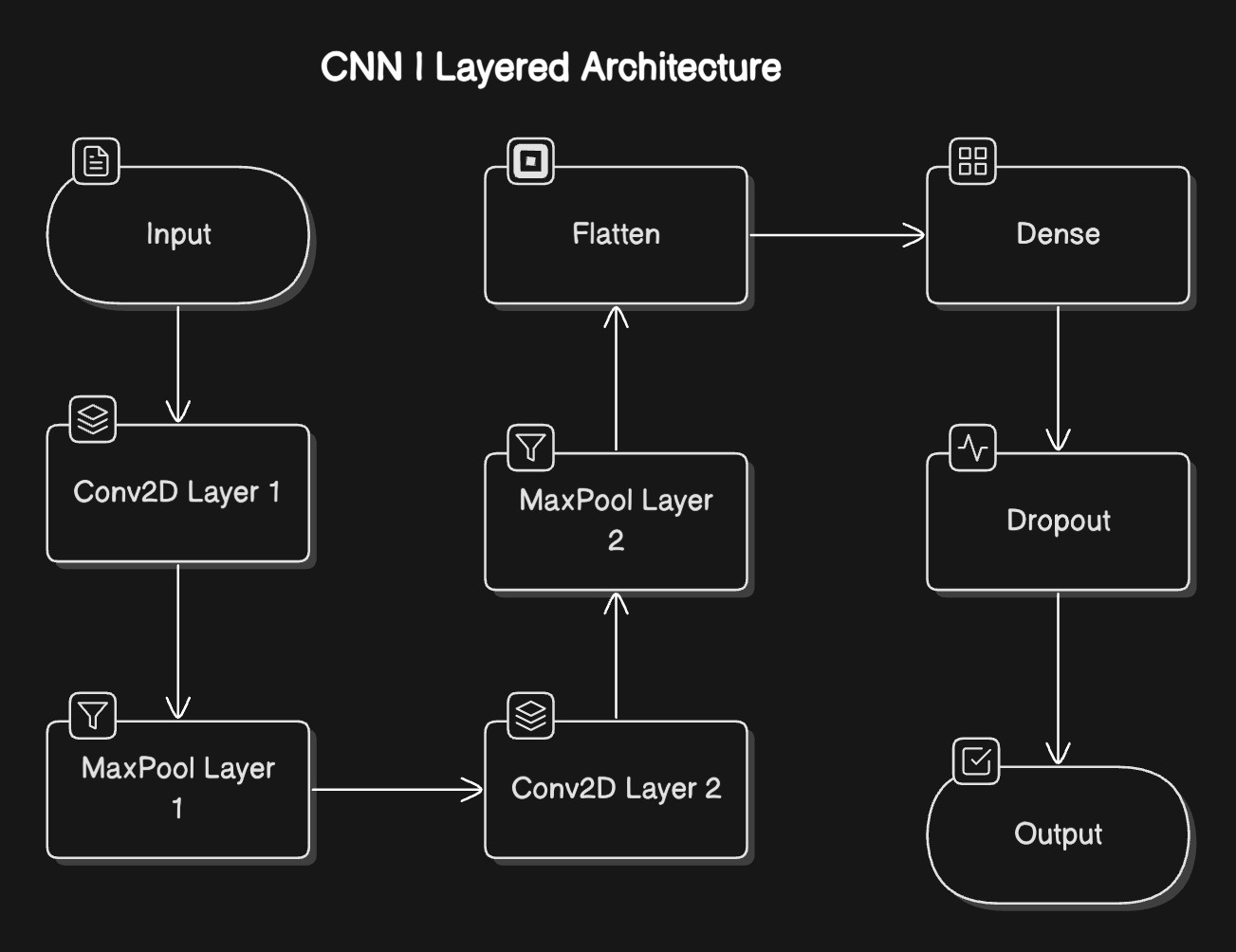
We used the following models:

### Custom CNN Architecture I Total layers: 8

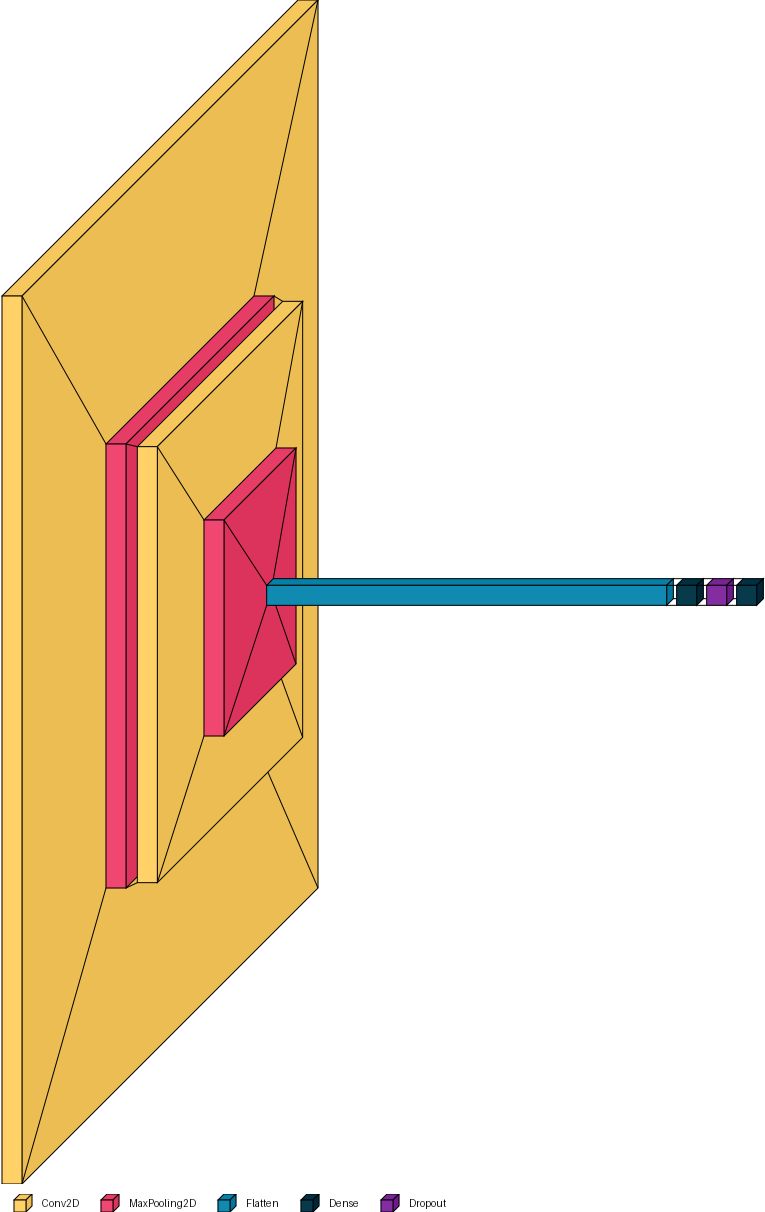
* + - * **2 Conv2D layers** (1st and 2nd layers)
      * **2 MaxPooling2D layers** (after 1st and 2nd Conv2D)

### 1 Flatten layer

* + - * **1 Dropout** (50%)
      * **1 Dense layers** (128 neurons)
      * **Output Dense Layer (**3 neurons)



*Figure: CNN custom layered architecture I*



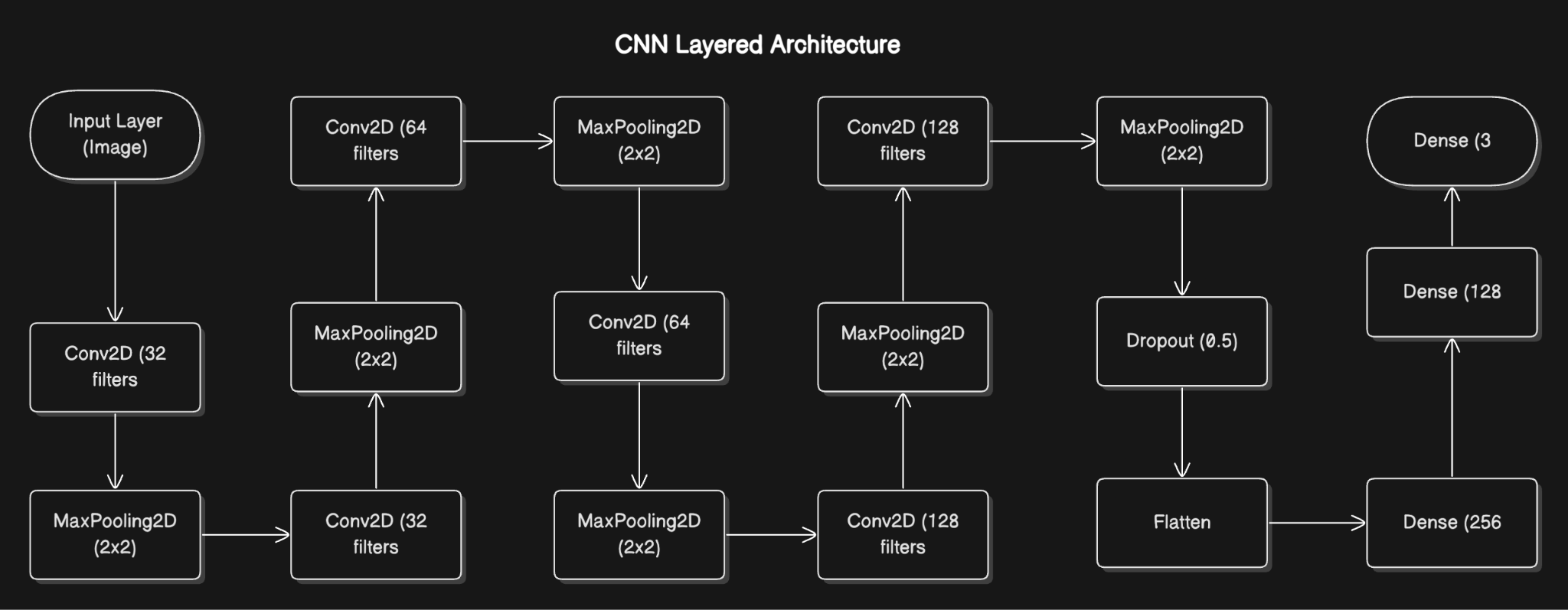
*Figure: CNN I layered architecture 3D representation*

### Custom CNN Architecture II Total layers: 17

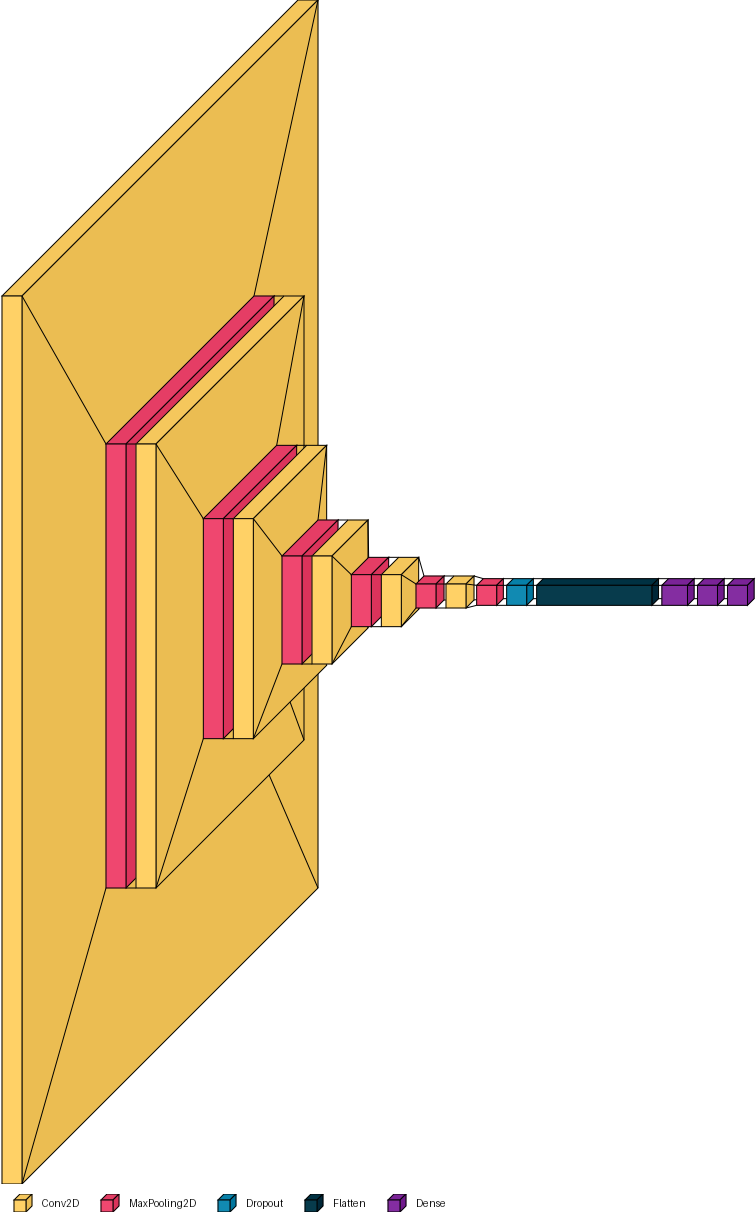
* + - * **6 Conv2D layers** (1st, 2nd, 3rd, 4th, and 5th layers)
      * **6 MaxPooling2D layers** (after each Conv2D layer)
      * **1 Dropout layer** (before flattening)

### 1 Flatten layer

* + - * **2 Dense layers** (one with 256 neurons and one with 128 neurons)
      * **Output Dense layer** (with 3 neurons)



*Figure: CNN custom layered architecture II*



*Figure: CNN custom layered architecture II* 3D representation

### MobileNetV2

MobileNetV2 is a lightweight, efficient convolutional neural network architecture designed for mobile and edge devices. It uses depthwise separable convolutions and an inverted residual

structure with linear bottleneck layers, making it fast and resource-efficient while maintaining a relatively high accuracy. MobileNetV2 is ideal for applications that require low-latency inference and a small memory footprint, such as real-time image classification and object detection on

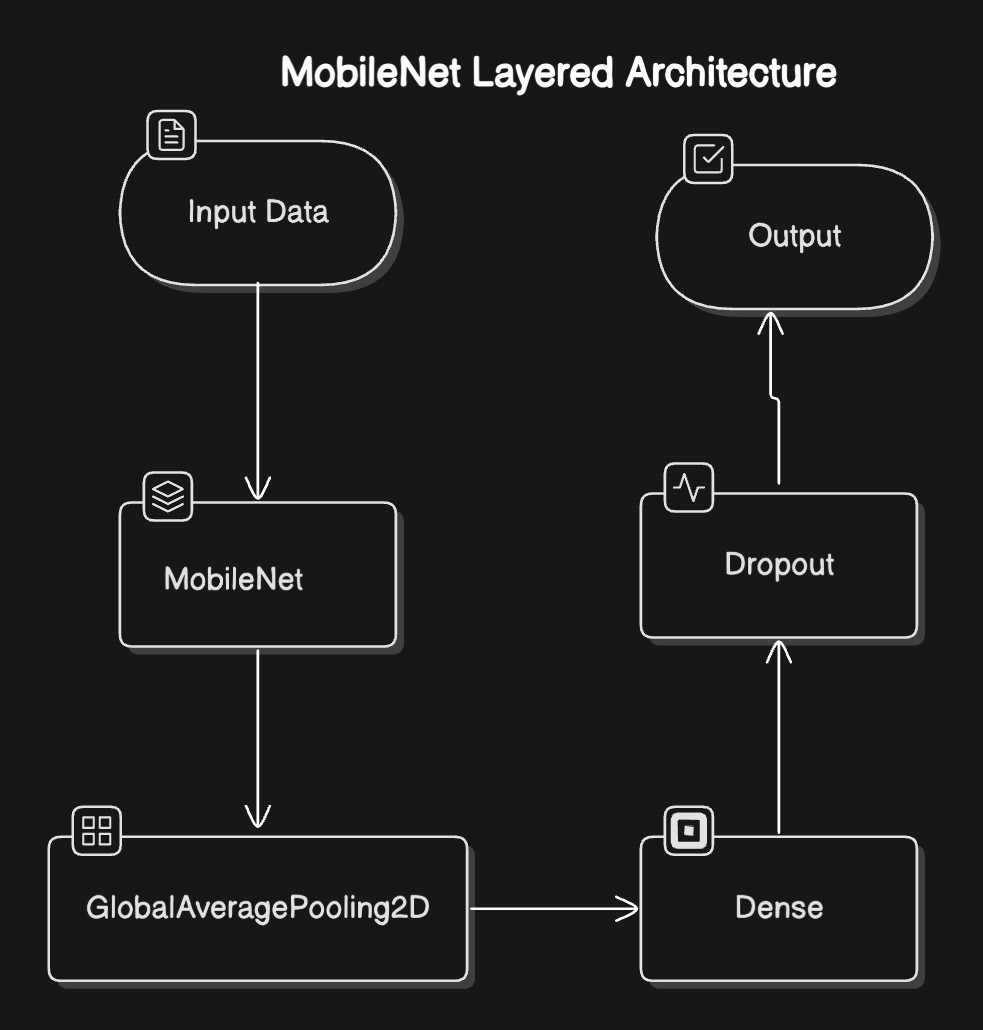
mobile devices.

### Total layers: 5

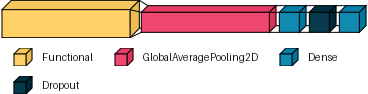
* + - * **1 MobileNetV2 base model** (pre-trained, not counting individual layers inside it, but considered as 1 layer in this context)

### 1 GlobalAveragePooling2D layer

* + - * **1 Dense layers** (128 neurons)
      * **1 Dropout layer** (50%)
      * **Output layer** (3 neurons)



*Figure: MobileNetV2 layered architecture*

**

*Figure: MobileNetV2 layered architecture 3D representation*

### ResNet50

ResNet50 is a deep residual network with 50 layers, based on the concept of residual connections or "skip connections" that allow the network to learn residual mappings instead of directly learning the desired underlying mapping. This architecture is effective for training very deep

networks by mitigating the vanishing gradient problem, helping in the learning process of deeper models. ResNet50 is commonly used in image classification tasks but may struggle with small

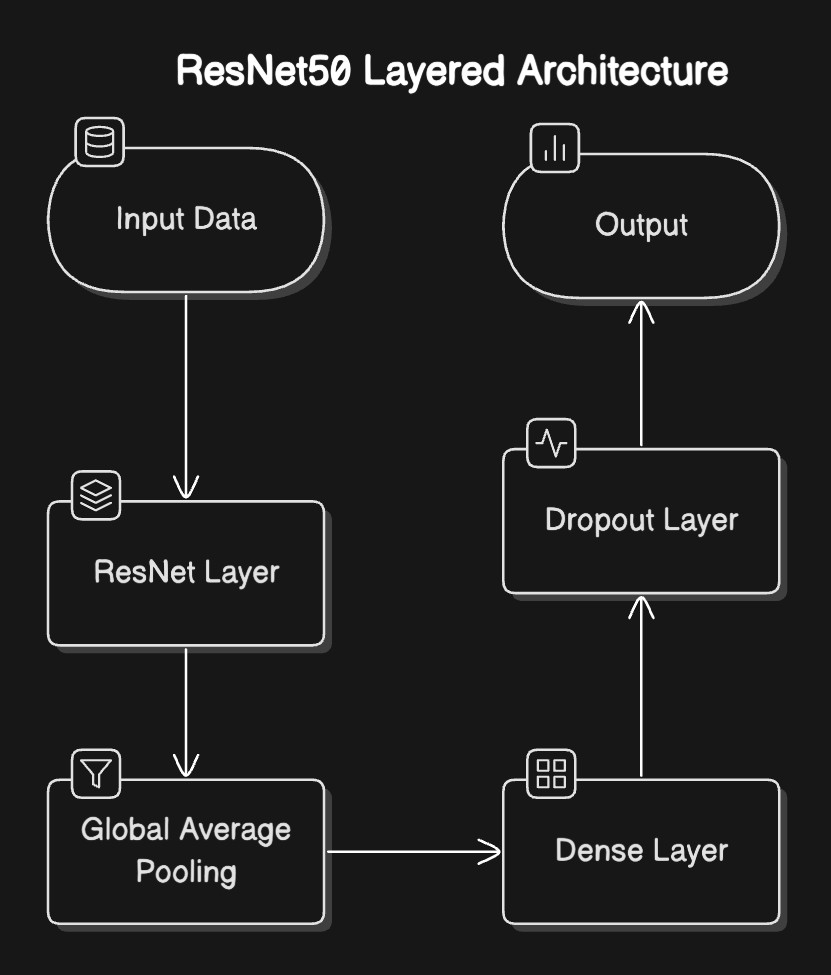
datasets due to its depth and complexity.

### Total layers: 5

* + - * **1 ResNet50 base model** (pre-trained, not counting individual layers inside it, but considered as 1 layer in this context)

### 1 GlobalAveragePooling2D layer

* + - * **1 Dense layers** (128 neurons)
      * **1 Dropout layer** (50%)
      * **Output layer (**3 neurons)



*Figure: ResNet50 layered architecture*

**

*Figure: ResNet50 layered architecture 3D representation*

### DenseNet121

DenseNet121 is a deep convolutional neural network that features dense connectivity, meaning each layer receives input from all previous layers. This architecture has 121 layers, and the dense connections allow it to reuse features effectively, improving the flow of gradients during training. DenseNet121 typically delivers high accuracy by encouraging feature reuse and reducing the number of parameters compared to traditional convolutional networks. However, its dense connectivity can make it more computationally expensive and resource-intensive.

### Total layers: 5

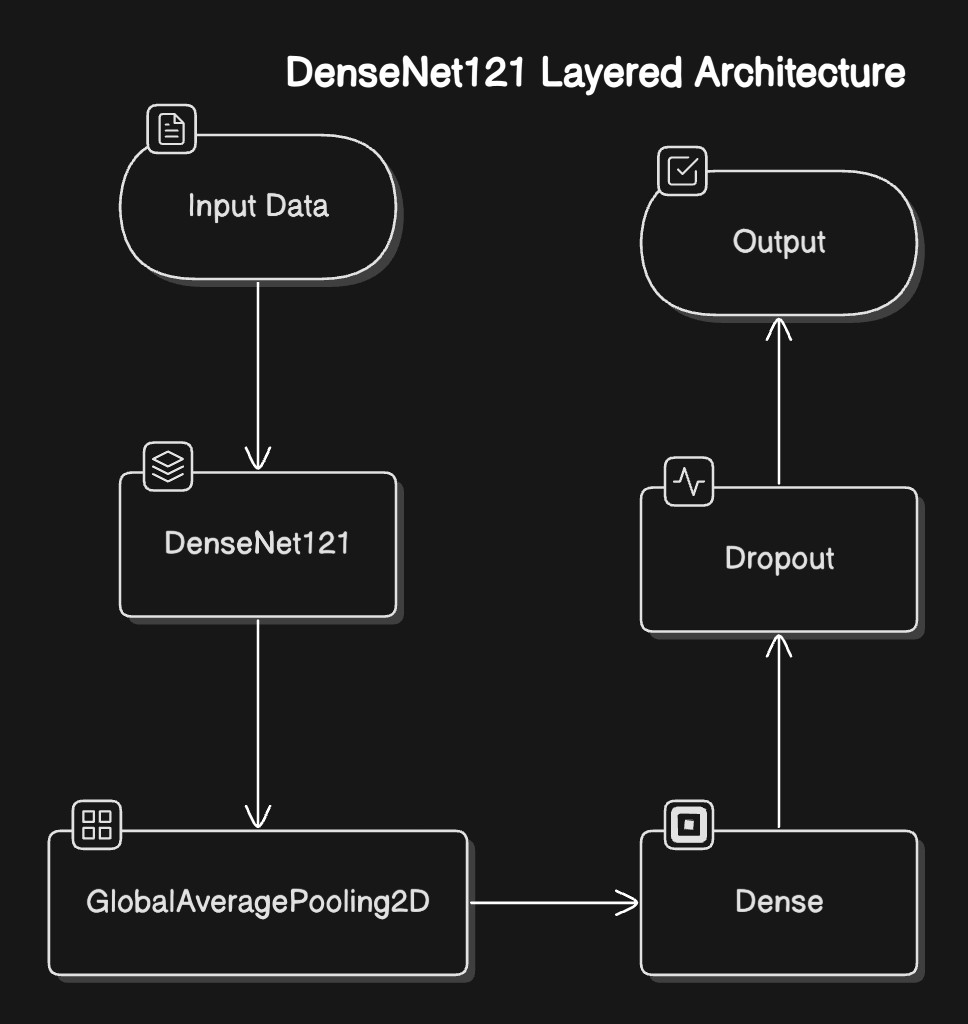
* + - * **1 DenseNet121 base model** (pre-trained, not counting individual layers inside it, but considered as 1 layer in this context)

### 1 GlobalAveragePooling2D layer

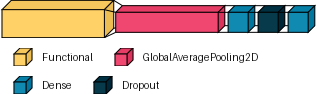
* + - * **1 Dense layers** (128 neurons)

### 1 Dropout layer

* + - * **Output layer** (3 neurons)



*Figure: DenseNet121 layered architecture*

**

*Figure: DenseNet121 layered architecture 3D representation*

### Training Process

* Loss function: Categorical Cross-Entropy
* Optimizer: Adam
* Learning rate: 0.001, reduced with a learning rate scheduler.
* Batch size: 32
* Early stopping applied to prevent overfitting.
* Epoches: 20

# Results and Analysis

### Evaluation Metrics

* **Accuracy:** This metric measures the proportion of correct predictions made by the model across the entire dataset. It is calculated as the ratio of true positives (TP) and true

negatives (TN) to the total number of samples.

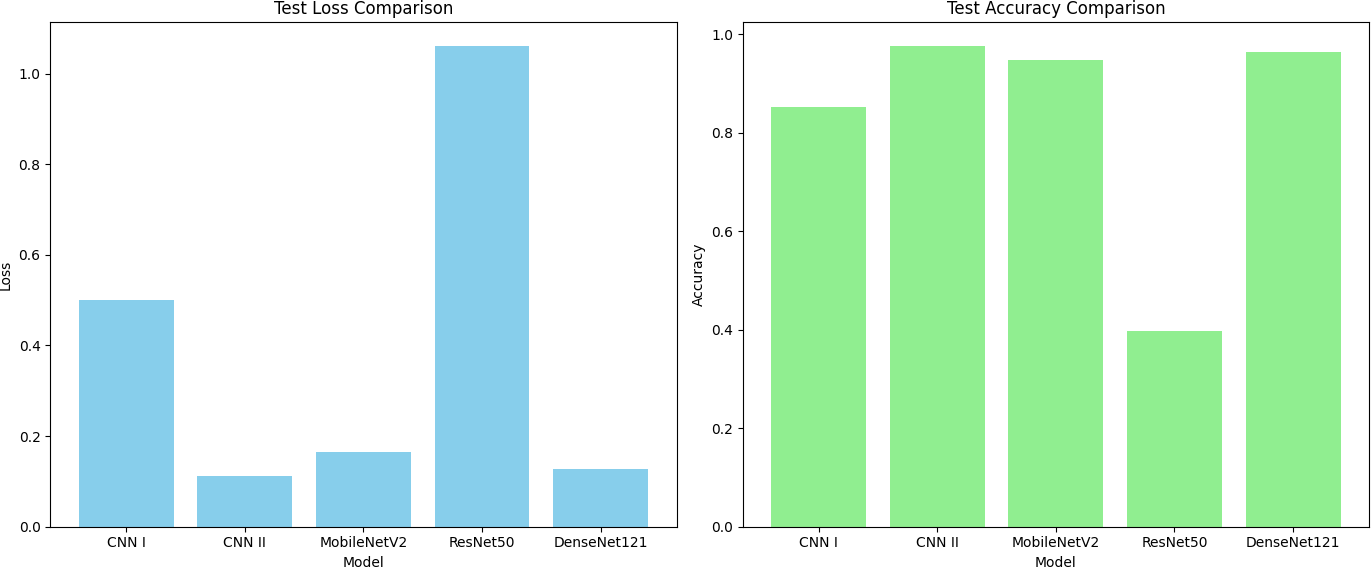
* **Loss:** Represents model error during training/testing; lower values indicate better performance.
* **Recall (Sensitivity):** Recall, also known as sensitivity or true positive rate, measures the proportion of true positive predictions among all actual positive instances. It is calculated as the ratio of TP to the sum of TP and false negatives (FN).
* **Precision:** Precision measures the proportion of true positive predictions among all

positive predictions made by the model. It is calculated as the ratio of TP to the sum of TP and false positives (FP).

* **F1-score:** F1 Score is a metric that balances precision and recall. It is calculated as the harmonic mean of precision and recall. F1 Score is useful when seeking a balance between high precision and high recall, as it penalizes extreme negative values of either component.

### Initial Model Performance

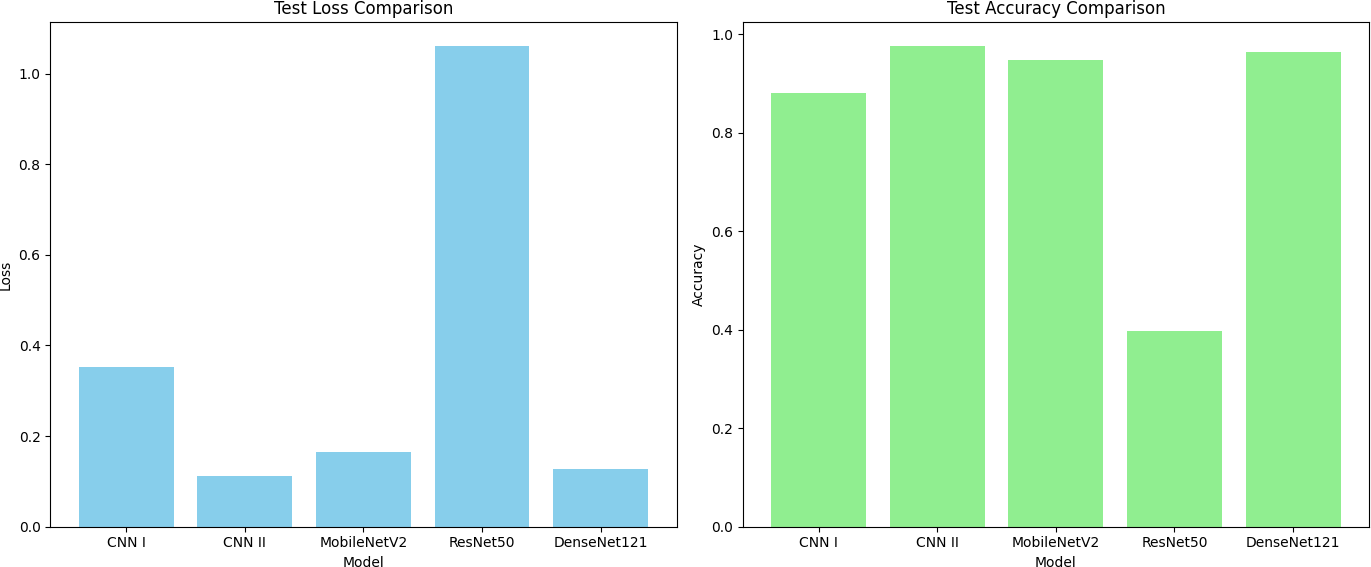
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model** | **Accuracy** | **Loss** | **Precision** | **Recall** | **F1-score** |
| CNN I | 80% | 0.65 | 0.85 | 0.86 | 0.85 |
| CNN II | 96% | 0.12 | 0.97 | 0.98 | 0.98 |
| MobileNetV2 | 94% | 0.18 | 0.95 | 0.95 | 0.95 |
| ResNet50 | 70% | 1.04 | 0.13 | 0.33 | 0.19 |
| DenseNet121 | 95% | 0.14 | 0.96 | 0.96 | 0.96 |



*Figure: Accuracy and loss comparison across models*

### Fine-Tuning Results

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model** | **Accuracy** | **Loss** | **Precision** | **Recall** | **F1-score** |
| CNN I | 88% | 0.33 | 0.89 | 0.88 | 0.88 |
| CNN II | 96% | 0.12 | 0.97 | 0.98 | 0.98 |
| MobileNetV2 | 94% | 0.18 | 0.95 | 0.95 | 0.95 |
| ResNet50 | 70% | 1.04 | 0.13 | 0.33 | 0.19 |
| DenseNet121 | 95% | 0.14 | 0.96 | 0.96 | 0.96 |

****

*Figure: Accuracy and loss comparison across models after fine-tuning*

### Insight and Interpretations of Results

**Pre-Fine-Tuning Results**

* **CNN I** had moderate performance (**80% accuracy**, **0.65 loss**) with decent precision, recall, and F1-score (all **0.85-0.86**). However, it fell short compared to other models like CNN II and DenseNet121.
* **CNN II** was the standout performer with **96% accuracy**, **0.12 loss**, and the highest precision, recall, and F1-score (all **0.97-0.98**).
* **MobileNetV2** and **DenseNet121** delivered strong results (94%-95% accuracy) with well-balanced precision, recall, and F1-scores (~**0.95-0.96**).
* **ResNet50** underperformed significantly, with poor accuracy (**70%**) and a very high loss (**1.04**). Its metrics indicated major difficulties in classifying the data properly.

### Post-Fine-Tuning Results

* **CNN I** improved significantly after fine-tuning, reaching **88% accuracy** and reducing its loss to **0.33**, with better precision (**0.89**) and F1-score (**0.88**). This indicates that

fine-tuning optimized its learning and reduced overfitting or underfitting issues.

* **CNN II** maintained its excellent performance, showing no notable changes. Its results indicate a well-generalized model that was already highly optimized.
* **MobileNetV2** remained stable, with no notable improvements or declines. It continued to deliver strong performance (~94% accuracy and **0.95 metrics**), indicating its robustness.
* **DenseNet121** also remained consistent at **95% accuracy**, demonstrating its ability to maintain a high level of generalization even after fine-tuning.
* **ResNet50** showed no improvement, maintaining poor accuracy (**70%**) and very low precision, recall, and F1-score. This suggests that either the architecture is not suitable for the task or that fine-tuning was not sufficient to address its learning issues.

|  |  |  |  |
| --- | --- | --- | --- |
| Transfer Learning Model | Architecture Complexity | Strength | Weaknesses |
| MobilNetv2 | Lightweight | Fast, Efficient, Rebost | Slightly lower accuracy than DenseNet121 |
| ResNet | Deep | Handles deeper networks via skip connections | Poor generalization to small datasets |
| DenseNet121 | Dense connectivity | Reuses features effectively, robust | Slightly more resource-intensive |

### 4 Data Augmentation Effect on Models Performance

The purpose of this experiment is to investigate how the size of the data and addressing the underrepresentation of the healthy class can affect the performance of our models.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Subset** | **Early Blight** | **Late Blight** | **Healthy** | Total by Subset |
| Training | 11320 | 11320 | 11320 | **33960** |
| Validation | 1510 | 1510 | 1510 | **4530** |
| Testing | 1410 | 1410 | 1410 | **4230** |
| Total by Class | **14240** | **14240** | **14240** | **42720** |

*Table: Augmented dataset structure (10x Augmentation)*

Total: 1.5 GB