***Comparative Analysis of DenseNet, ResNet, and Xception on Potato Health Detection***

**Project Overview**  
This project investigates the use of deep learning techniques for classifying potato plant conditions using the PlantVillage dataset. The focus is on three specific classes: **Potato\_\_\_Early\_blight**, **Potato\_\_\_Late\_blight**, and **Potato\_\_\_healthy**. To achieve this, three models—**DenseNet**, a custom **ResNet** (implemented from scratch), and **Xception**—are developed and evaluated. By comparing their performance, the study aims to identify the most effective architecture for accurate plant disease diagnosis, contributing to advancements in precision agriculture and plant health management.

1. **Introduction to DenseNet**



DenseNet, introduced in the paper *"Densely Connected Convolutional Networks"* (Huang et al., CVPR 2017), is an architecture known for its efficiency in parameter usage. It improves information flow by connecting each layer to every other layer, ensuring feature reuse. This is achieved through **dense blocks** that concatenate outputs instead of summing them (as in ResNet).

**Key Reference:**

* Paper: [Densely Connected Convolutional Networks](https://arxiv.org/abs/1608.06993)

**2. Step-by-Step Explanation of Code**

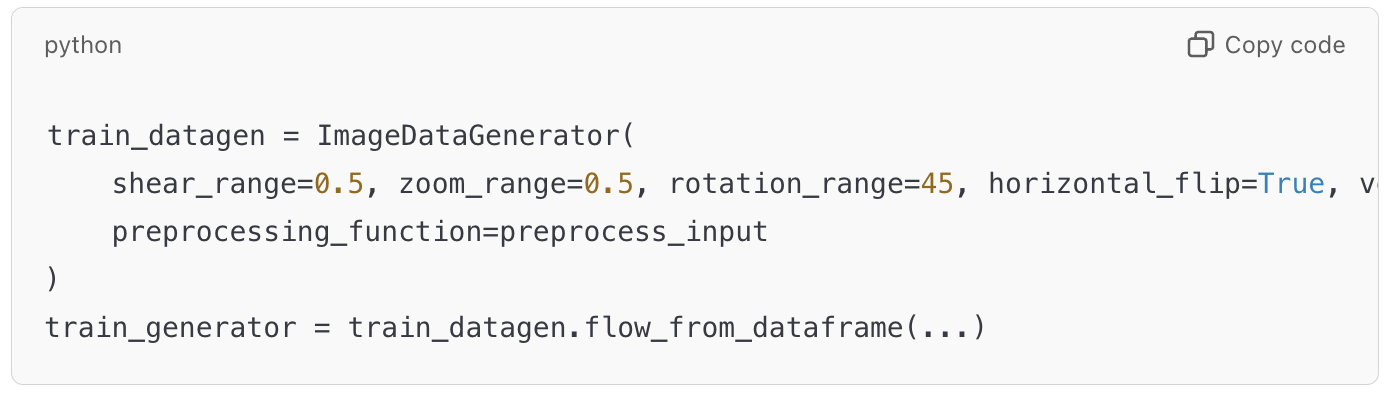
**a. Data Loading and Preprocessing**

* **Dataset Creation**:A screenshot of a computer

  Description automatically generated

The code processes subdirectories containing "Potato" images, creating a DataFrame with file paths and labels.

* **Data Splitting**:A close-up of a computer code

  Description automatically generatedThe dataset is split into training, validation, and test sets, maintaining class balance using stratify.
* **Augmentation and Generators**:The ImageDataGenerator applies data augmentation (rotation, flipping, etc.) and uses preprocess\_input for DenseNet.

**b. Model Architecture**

* **DenseNet Base**:The pre-trained DenseNet121 is loaded without its top layers. This serves as a feature extractor.
* **Custom Head**:A screenshot of a computer code

  Description automatically generated
  + **GlobalAveragePooling2D**: Reduces the spatial dimensions of the feature maps.
  + **Dense Layer**: Adds a fully connected layer with L2 regularization to prevent overfitting.
  + **Dropout**: Randomly drops neurons during training, reducing overfitting.
  + **Output Layer**: Softmax layer for multi-class classification.

**c. Training**

* **Compiling the Model**: **A screen shot of a computer code

  Description automatically generated**Uses a low learning rate for fine-tuning the pre-trained model.
* **Early Stopping and Checkpointing**:A screenshot of a computer code

  Description automatically generatedPrevents overfitting by stopping early if validation loss doesn’t improve.

**d. Evaluation**

* **Accuracy and Loss**:A white background with black and green text

  Description automatically generatedPlots training and validation metrics over epochs to visualize model performance.
* **Confusion Matrix**:A white background with black text

  Description automatically generatedEvaluates model predictions class-wise.
* **ROC Curve and AUC**:A white background with black text

  Description automatically generatedMeasures how well the model distinguishes between classes.

**e. Fine-Tuning**

* **Unfreezing Layers**:A white background with black text

  Description automatically generatedGradually unfreezes layers for fine-tuning, starting with the last 20 layers.
* **Class Weights**:A close-up of a word

  Description automatically generatedBalances class distribution in the loss function to handle imbalance in the dataset.

**3. Key Concepts of DenseNet**

**1. Dense Connectivity**

* Unlike traditional CNNs where each layer passes its output only to the next layer, in DenseNet, **each layer receives input from all previous layers** and passes its output to all subsequent layers.
* This dense connectivity allows better feature reuse and reduces the problem of vanishing gradients.
* If there are *L* layers, each layer has *L(L+1)/2* direct connections.

**2. Growth Rate (kk)**

* The **growth rate** is the number of output feature maps (channels) each layer produces.
* For example, if *K*= 32, each layer contributes 32 feature maps to subsequent layers.
* This keeps the network compact because the number of new feature maps added is controlled by *K*.

**3. Bottleneck Layers**

* To reduce computational complexity, DenseNet uses a **bottleneck layer** (1x1 convolution) before the main convolution in each dense block. This reduces the number of input feature maps.

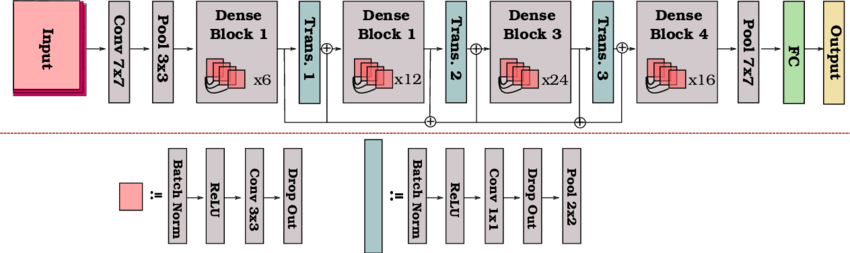
**4. Compression**

* To further reduce the size of the network, DenseNet employs a **transition layer** between dense blocks. These layers perform down-sampling using 1X1 convolutions followed by pooling.

**5. Dense Blocks**

* The network is divided into **dense blocks**, where the dense connectivity is implemented.
* Each block is followed by a **transition layer** (except the last one), which reduces the spatial dimensions and the number of feature maps.

**DenseNet Architecture**



The architecture can be summarized as follows:

1. **Input Layer**:
   * A 7 x 7 convolution followed by a max-pooling operation to reduce spatial dimensions.
2. **Dense Block**:
   * A dense block is a sequence of layers where each layer receives input from all previous layers and contributes its output to the subsequent layers.
   * Each layer consists of:
     + Batch Normalization (BN)
     + ReLU activation
     + 1 x 1 Convolution (bottleneck layer)
     + Batch Normalization (BN)
     + ReLU activation
     + 3 x 3 Convolution
   * The outputs of all layers are concatenated along the feature dimension.
3. **Transition Layer**:
   * Between dense blocks, a transition layer reduces the number of feature maps using 1 x 1 convolution and down-samples the feature maps using 2 x 2 average pooling.
4. **Growth Rate** (*K*):
   * Determines how many new feature maps are added by each layer in a dense block.
   * If the initial number of feature maps is *F0*, and the block has *L* layers, the final number of feature maps will be *F0* + *K* × L
5. **Classification Layer**:
   * At the end of the network:
     + Global average pooling (reduces spatial dimensions to 1x1).
     + Fully connected layer with softmax activation for classification.

**DenseNet Variants**

DenseNet has several variants depending on the depth of the network:

* **DenseNet-121** (used in my project): Has 121 layers.
* **DenseNet-169**
* **DenseNet-201**
* **DenseNet-264**

The numbers (121, 169, etc.) indicate the total number of layers, including the dense blocks and transition layers.

**4. Graphs**

***-MODEL ACCURACY AND LOSS CURVE BEFORE FINE-TUNING***A comparison of a graph

Description automatically generated with medium confidence

***-MODEL ACCURACY AND LOSS CURVE AFTER FINE-TUNING***

A comparison of a graph

Description automatically generated with medium confidence

***-MODEL (precision, recall, f1-score, confusion matrix) BEFORE FINE-TUNING***

precision recall f1-score support

Potato\_\_\_Early\_blight 1.00 0.12 0.21 50

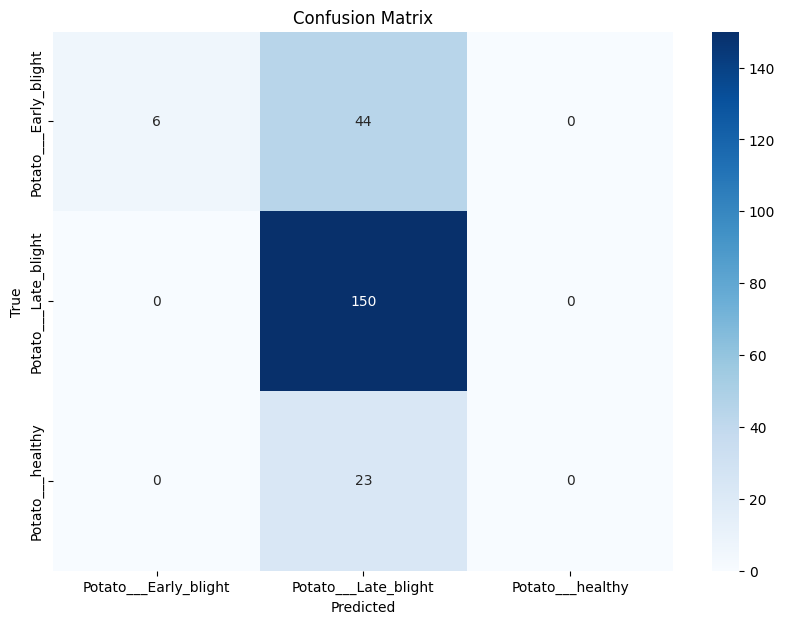
Potato\_\_\_Late\_blight 0.69 1.00 0.82 150

Potato\_\_\_healthy 0.00 0.00 0.00 23

accuracy 0.70 223

macro avg 0.56 0.37 0.34 223

weighted avg 0.69 0.70 0.60 223



***-MODEL (precision, recall, f1-score, confusion matrix) AFTER FINE-TUNING***

precision recall f1-score support

Potato\_\_\_Early\_blight 0.66 0.82 0.73 50

Potato\_\_\_Late\_blight 0.83 0.76 0.79 150

Potato\_\_\_healthy 0.26 0.26 0.26 23

accuracy 0.72 223

macro avg 0.58 0.61 0.59 223

weighted avg 0.73 0.72 0.72 223

A blue squares with white text

Description automatically generated

***-MODEL ROC AND AUC BEFORE FINE-TUNING***

A graph of a curve

Description automatically generated

***-MODEL ROC AND AUC AFTER FINE-TUNING***

A graph of a curve

Description automatically generated with medium confidence

**5. Enhancements**

* **Learning Rate Scheduling**: Introduce dynamic learning rate adjustments (e.g., ReduceLROnPlateau).
* **Batch Normalization Tuning**: Adjust BatchNorm layers for better convergence.
* **Data Augmentation**: Use more advanced augmentation techniques (e.g., CutMix or MixUp).
* **Ensemble Learning**: Combine DenseNet predictions with other models for improved results.