**PROJECT : "Potato Leaf Disease Classification Using CNN"**

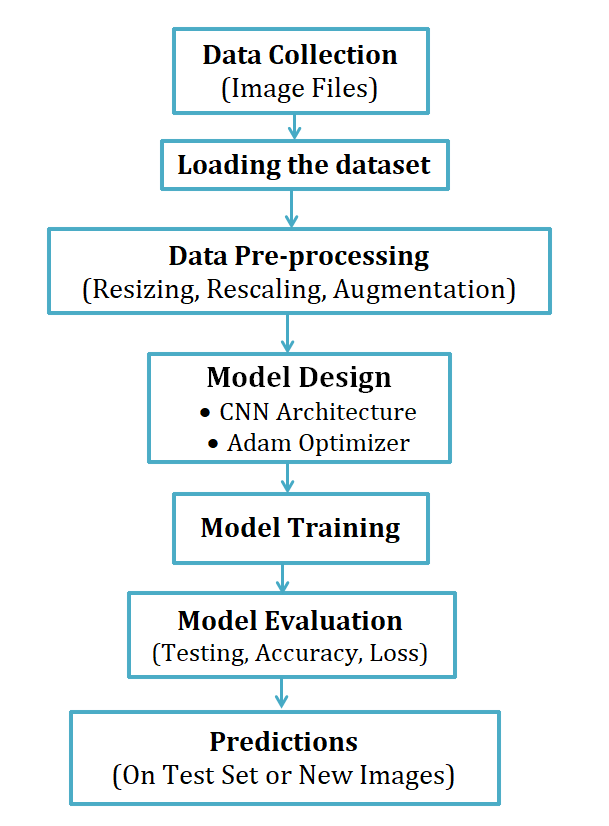
|  |  |  |
| --- | --- | --- |
| **Snehashish Das(233)** | **Rohan Dutta(223)** | **Kushal Dalal(215)** |

**PROGRESS REPORT AS ON 11.12.2024**

**1. Explanation of Work**

* **Objective**: Developing a machine learning-based system to detect Potato Leaf Diseases (Healthy, Late Blight and Early Blight)
* **Importance**: Early disease detection helps farmers take timely action, minimizing crop losses and ensuring food security.
* **Approach for this work in brief**:
  + Collecting a data set of potato leaf images from Kaggle.
  + Adjusting the image size and performing augmentation and splitting the data-set based on training, validation and testing.
  + Creating own CNN architecture and using an optimizer(adam) for compilation.
  + Train a model on the training dataset on a specified no. of epochs and using the validation dataset for monitoring.
  + Evaluate the model and generate predictions and plotting the confusion matrix.

**2. Block Diagram of Work**

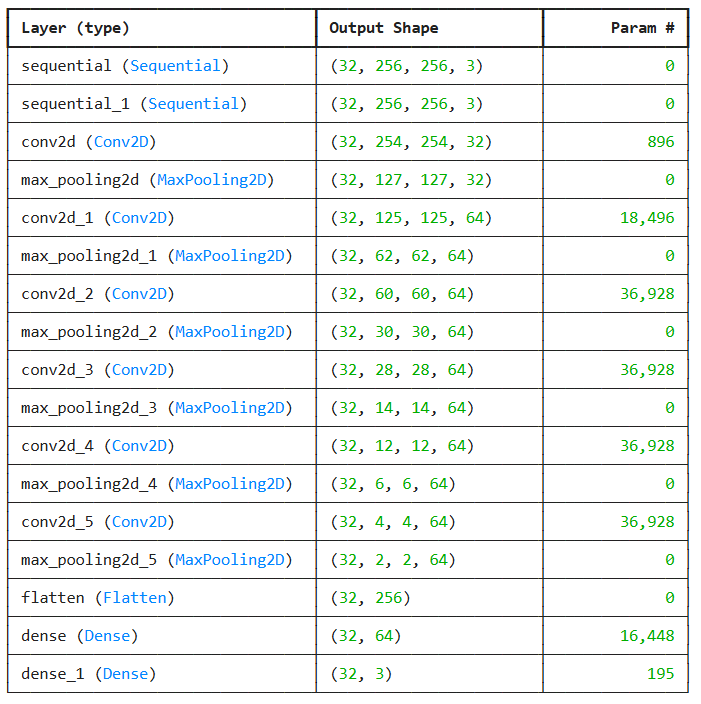


**3. Technology Details**

* **Programming Language:** Python
* **Platform:** Jupyter Notebook**.**
* **Frameworks:** TensorFlow/Keras for building and training the model.
* **Libraries:** Numpy, Matplotlib, Seaborn, Scikit-learn
* **Hardware Used:** AMD Ryzen 5 4600G, 3200Mhz Ram 8GB

**4. ML Model**

* **Model Type**: Convolutional Neural Network (CNN) for image classification.
* **Input Layer**:
* Images resized to *256x256* pixels and rescaled to [0, 1] range.
* **Data Augmentation**:
* Random horizontal/vertical flips and rotations for better generalization.
* **Convolutional Layers**:
* 6 Conv2D layers with ReLU activation for feature extraction.
* Kernel size: (*3, 3*) in each Conv2D layer.
* MaxPooling after each Conv2D to reduce spatial dimensions.
* **Flatten Layer**:
  + Flattens the 3D feature maps into 1D vectors.
* **Fully Connected Layer**:
  + 1 Dense layer with 64 units and ReLU activation.
* **Output Layer**:
  + Dense layer with 3 units (one per class) and softmax activation for multi-class classification.
* **Loss Function**:
  + Sparse Categorical Crossentropy, as the labels are integers (not one-hot encoded).
* **Optimizer**:
  + Adam optimizer for efficient training.



**5. Algorithm**

#### ****Step 1: Data Preparation****

* Load the image dataset using **tf.keras.preprocessing.image\_dataset\_from\_directory** and split it into training, validation, and test sets.
* Resize the images to a uniform size of **256x256** and normalize pixel values to the range [0, 1].
* Apply data augmentation techniques such as random flipping and rotation to enhance the dataset and improve model generalization.

#### ****Step 2: Model Design****

* Define a sequential CNN model consisting of resizing, rescaling, and data augmentation layers.
* Add convolutional layers with **ReLU** activation for feature extraction and **MaxPooling** layers to reduce spatial dimensions.
* Flatten the features into a 1D vector, followed by dense layers, including the output layer with **softmax** activation for multi-class classification.

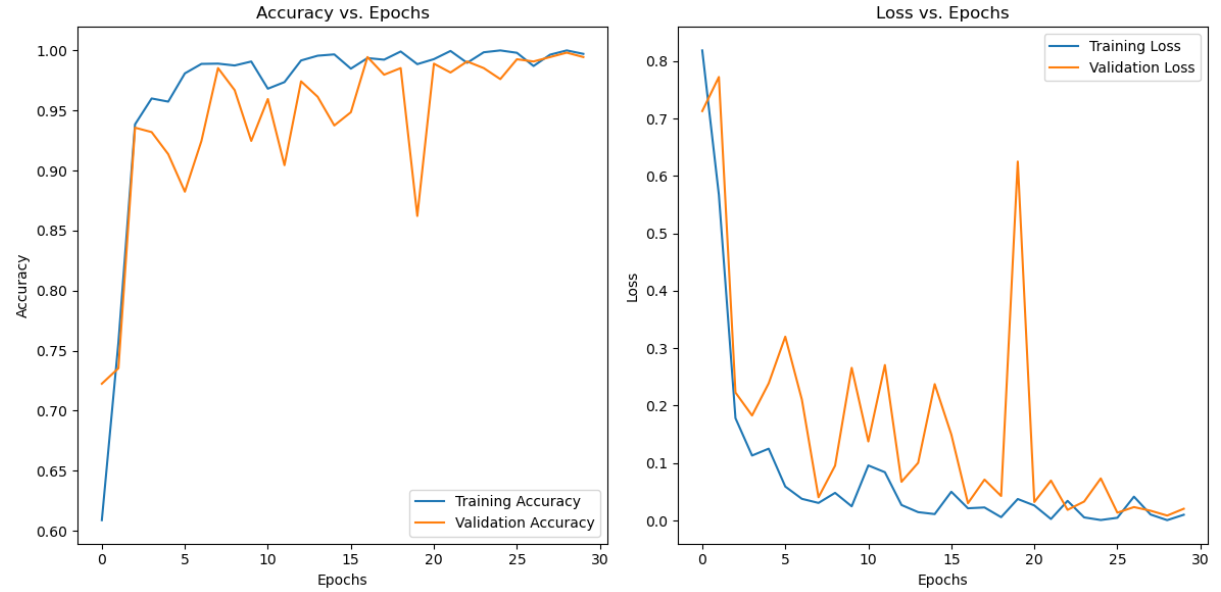
#### ****Step 3: Model Compilation****

* Compile the model with the **Adam** optimizer for adaptive learning.
* **Sparse Categorical Crossentropy** loss function for multi-class classification, and accuracy as the evaluation metric to track performance.

#### ****Step 4: Model Training****

* Train the model using the training dataset with the fit() method and monitor its performance on the validation dataset.
* Iterate through **30** epochs to optimize the weights for achieving better accuracy and minimizing the loss.

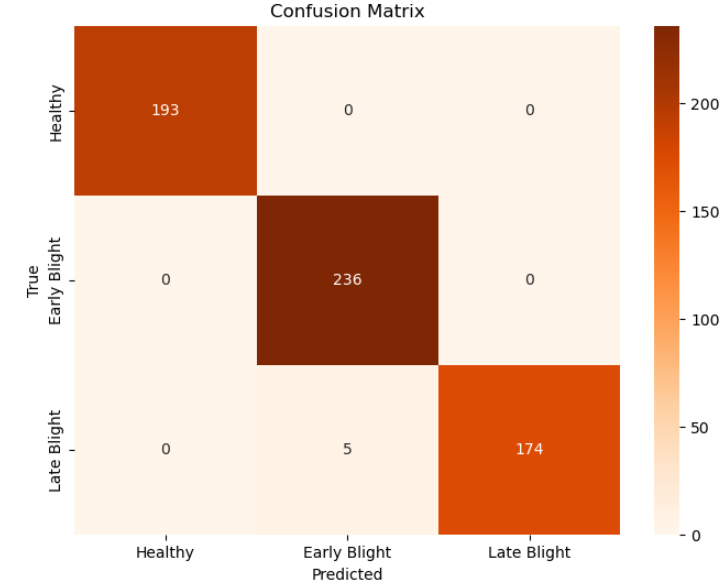
#### ****Step 5: Model Evaluation****

* Evaluate the model on the test dataset using the evaluate() method to measure its final performance in terms of accuracy and loss.
* A **99.18%** accuracy is achieved using CNN Model.
* 

#### ****Step 6: Predictions****

* Processed each image (resize, rescale).
* Passed the image through the trained model.
* Obtained the predicted class and confidence score.

#### ****Step 7: Performance Analysis****

* Compute a confusion matrix by comparing true labels with predicted labels to evaluate the classification performance.
* Visualize the confusion matrix using a heatmap to identify areas where the model performs well and where it may need improvement.
* 
* 