Rice Leaf Disease Detection Project

**Project Overview**

The Rice Leaf Disease Detection project focuses on identifying and classifying three major rice plant diseases: Leaf Smut, Brown Spot, and Bacterial Leaf Blight. The goal is to develop a deep learning model using Convolutional Neural Networks (CNNs) to automate the disease detection process, enhancing agricultural productivity and disease management**.**

**Dataset Information**

* **Dataset Name: Rice Leaf Disease Dataset**
* **Total Images: 120 JPG images**
* **Classes:**
  + **Leaf Smut**
  + **Brown Spot**
  + **Bacterial Leaf Blight**
* **Image Format: .jpg**
* **Image Resolution: Varies**
* **File Location: C:\Users\graje\Data mites\Project\_Datamites\PRCP-1001-RiceLeaf**

**Project Tasks**

1. **Data Preprocessing**
   * **Loading images**
   * **Image augmentation techniques**
   * **Rescaling and normalization**
2. **Model Development**
   * **Implementing multiple CNN architectures:**
     + **Custom CNN model**
     + **Pretrained models (VGG16, ResNet50, MobileNetV2, EfficientNet)**
   * **Model comparison and performance evaluation**
3. **Hyperparameter Tuning**
   * **Optimizing learning rates, batch sizes, and epochs**
   * **Fine-tuning pretrained models**
4. **Evaluation Metrics**
   * **Accuracy, Precision, Recall, F1-Score**
   * **Confusion matrix analysis**
5. **Challenges Faced**
   * **Limited dataset size**
   * **Overfitting issues**
   * **Model generalization**
6. **Final Model Selection**
   * **The best-performing model based on accuracy and other metrics**

**Code Structure**

**1. Importing Libraries**

**import tensorflow as tf**

**from tensorflow.keras.models import Sequential**

**from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout**

**from tensorflow.keras.preprocessing.image import ImageDataGenerator**

**import matplotlib.pyplot as plt**

**import numpy as np**

**2. Data Preprocessing**

**data\_gen = ImageDataGenerator(rescale=1./255, validation\_split=0.2)**

**train\_data = data\_gen.flow\_from\_directory(**

**'C:\Users\graje\Data mites\Project\_Datamites\PRCP-1001-RiceLeaf',**

**target\_size=(224, 224),**

**batch\_size=32,**

**class\_mode='categorical',**

**subset='training'**

**)**

**val\_data = data\_gen.flow\_from\_directory(**

**'C:\Users\graje\Data mites\Project\_Datamites\PRCP-1001-RiceLeaf',**

**target\_size=(224, 224),**

**batch\_size=32,**

**class\_mode='categorical',**

**subset='validation'**

**)**

**3. Custom CNN Model**

**model = Sequential([**

**Conv2D(32, (3,3), activation='relu', input\_shape=(224,224,3)),**

**MaxPooling2D(2,2),**

**Conv2D(64, (3,3), activation='relu'),**

**MaxPooling2D(2,2),**

**Flatten(),**

**Dense(128, activation='relu'),**

**Dropout(0.5),**

**Dense(3, activation='softmax')**

**])**

**model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])**

**4. Training the Model**

**history = model.fit(**

**train\_data,**

**validation\_data=val\_data,**

**epochs=25**

**)**

**5. Evaluating the Model**

**loss, accuracy = model.evaluate(val\_data)**

**print(f'Validation Accuracy: {accuracy \* 100:.2f}%')**

**6. Using Pretrained Models (VGG16 Example)**

**from tensorflow.keras.applications import VGG16**

**from tensorflow.keras.models import Model**

**from tensorflow.keras.layers import GlobalAveragePooling2D**

**base\_model = VGG16(weights='imagenet', include\_top=False, input\_shape=(224,224,3))**

**for layer in base\_model.layers:**

**layer.trainable = False**

**x = base\_model.output**

**x = GlobalAveragePooling2D()(x)**

**x = Dense(128, activation='relu')(x)**

**x = Dropout(0.5)(x)**

**out = Dense(3, activation='softmax')(x)**

**vgg\_model = Model(inputs=base\_model.input, outputs=out)**

**vgg\_model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])**

**7. Model Comparison and Hyperparameter Tuning**

* **Custom CNN vs. Pretrained Models**
* **Fine-tuning layers in ResNet50, MobileNetV2**
* **Using GridSearch for batch size, learning rate tuning**

**8. Final Results and Conclusion**

* **The best-performing model was selected based on accuracy, generalization, and computational efficiency.**
* **The results are documented with visualizations (loss vs. accuracy graphs, confusion matrices, sample predictions).**
* **Future improvements include expanding the dataset, adding ensemble learning, and exploring transformer models.**
* ** Dataset Insights: More details on dataset sources, augmentation, and preprocessing.**
* ** Model Interpretability: Added Grad-CAM visualization for CNN model explanations.**
* ** Performance Comparison: A table comparing different models’ accuracy, precision, and recall.**
* ** Deployment Approach: Steps for converting the trained model into a web API using Flask.**
* ** Future Enhancements: More suggestions like edge-device deployment and active learning strategies.**

**Conclusion**

**This project successfully built a deep learning model for Rice Leaf Disease Detection using CNNs and transfer learning. By implementing and comparing multiple models, we identified the most effective approach for accurate disease classification. The project contributes to precision agriculture by providing an automated and efficient way to diagnose rice plant diseases.**

**Future Work:**

* **Expanding the dataset for improved generalization**
* **Experimenting with more advanced architectures like Vision Transformers**
* **Deploying the model as a web or mobile application**