**Plant Disease Detection Using Deep Learning**

**Project Overview**

This project focuses on developing a deep learning model to detect plant diseases using image data. The system classifies plant leaves into various categories (healthy or infected with specific diseases) based on an image. It leverages transfer learning using the VGG16 model and fine-tuning techniques to achieve high accuracy and robust classification.

**Objectives**

* Create a model that classifies plant leaves as healthy or infected with specific diseases.
* Achieve an accuracy of at least 80%.
* Address challenges like class imbalance and poor precision/recall for minority classes.
* Implement preprocessing techniques to enhance the quality of input data.
* Optimize training for better generalization to unseen data.

**Dataset**

* The dataset consists of images of plant leaves, categorized into 15 classes (e.g., healthy leaves, bacterial spot, early blight, late blight, etc.).
* **Class Distribution:** Imbalanced, with some classes being significantly underrepresented.
* **Dataset Link :** https://www.kaggle.com/datasets/emmarex/plantdisease

**Preprocessing**

1. **Resizing:** All images resized to 224x224 pixels to match the input size of VGG16.
2. **Normalization:** Pixel values normalized to the range [0, 1].
3. **Data Augmentation:** Applied techniques such as random rotations, flips, zooms, and brightness adjustments to create diverse training samples.

**Model Architecture**

**Base Model**

* **Pre-trained Model:** VGG16 (transfer learning)
  + Pre-trained on ImageNet for feature extraction.
  + Top layers replaced to adapt to the current classification task.

**Custom Layers**

* Global Average Pooling layer.
* Fully connected dense layers.
* Dropout layers to prevent overfitting.
* Output layer with 15 nodes and softmax activation.

**Optimization**

* **Loss Function:** Categorical Cross-Entropy
* **Optimizer:** Adam with learning rate scheduling.
* **Metrics:** Accuracy, Precision, Recall, F1-Score

**Training and Validation**

**Hyperparameters**

* Batch Size: 32
* Epochs: 30
* Initial Learning Rate: 1e-4

**Class Balancing**

* Used compute\_class\_weight to assign weights to underrepresented classes.

**Evaluation Metrics**

* Validation Accuracy: Achieved **86.95%**.
* Precision, Recall, and F1-scores calculated for individual classes to evaluate per-class performance.

**Challenges and Solutions**

**1. Class Imbalance**

* **Issue:** Minority classes had poor precision and recall.
* **Solution:**
  + Oversampling minority classes using data augmentation.
  + Applied class-weighted loss during training.

**2. Model Overfitting**

* **Issue:** High training accuracy but low validation performance.
* **Solution:**
  + Added dropout layers.
  + Reduced learning rate for fine-tuning.
  + Increased data augmentation to generalize better.

**3. Poor Performance on Certain Classes**

* **Issue:** Classes like "Potato\_\_\_healthy" had low F1-scores.
* **Solution:** Focused augmentation and rebalanced the dataset.

**Results**

**Model Performance**

* **Validation Accuracy:** 86.95%
* **Confusion Matrix:** Analyzed misclassifications to improve predictions further.
* **Precision-Recall Analysis:** Focused on improving metrics for underrepresented classes.

**Class-wise Metrics**

| **Class Name** | **Precision** | **Recall** | **F1-Score** | **Support** |
| --- | --- | --- | --- | --- |
| Pepper\_\_bell\_\_\_Bacterial\_spot | 0.05 | 0.05 | 0.05 | 199 |
| Pepper\_\_bell\_\_\_healthy | 0.06 | 0.06 | 0.06 | 295 |
| Potato\_\_\_Early\_blight | 0.06 | 0.06 | 0.06 | 200 |
| Potato\_\_\_Late\_blight | 0.06 | 0.07 | 0.06 | 200 |
| Potato\_\_\_healthy | 0.00 | 0.00 | 0.00 | 30 |
| ... | ... | ... | ... | ... |

**Future Improvements**

1. **Advanced Architectures:**
   * Experiment with other architectures like ResNet, EfficientNet, or Vision Transformers.
2. **Ensemble Learning:**
   * Combine predictions from multiple models to improve accuracy and robustness.
3. **Explainability:**
   * Use techniques like Grad-CAM to visualize why the model made specific predictions.
4. **Hyperparameter Tuning:**
   * Perform grid search or Bayesian optimization for finding optimal hyperparameters.

**Code Implementation**

For the complete code, including preprocessing, model training, evaluation, and visualization, please refer to the attached Python scripts.

**Conclusion**

This project successfully built a deep learning pipeline for detecting plant diseases using image classification. Despite challenges like class imbalance and underrepresented categories, the model achieved a validation accuracy of **86.95%**. Further improvements and refinements can enhance the system's precision and recall, making it a reliable tool for agricultural applications.