AI BUILDER COURSE

CAPSTONE PROJECT REPORT

LeafLens

Farmer’s Botanical Ally : A plant leaf disease classification model

Introduction

The agricultural sector plays a key role in supplying quality food and makes the greatest contribution to growing economies and populations. Plant disease may cause significant losses in food production and eradicate diversity in species. Early diagnosis of plant diseases using accurate or automatic detection techniques can enhance the quality of food production and minimize economic losses. In recent years, deep learning has brought tremendous improvements in the recognition accuracy of image classification and object detection systems.

Table of Contents :

Problem Statement

Plant diseases pose significant threats to agricultural productivity, leading to substantial economic losses and jeopardizing food security worldwide. Timely and accurate identification of plant diseases is crucial for implementing effective management strategies, such as targeted pesticide application, crop rotation, and genetic resistance breeding. However, traditional methods of disease identification rely heavily on human expertise, which can be time-consuming, subjective, and prone to errors. Additionally, in regions with limited access to expert knowledge or diagnostic facilities, the timely detection and control of plant diseases become even more challenging.

Therefore, there's a critical need for an automated plant disease identification system that utilizes advanced technologies like machine learning and computer vision. This system should provide rapid and precise diagnoses based on visual symptoms, empowering farmers and agricultural stakeholders to effectively manage crop diseases and protect global food supplies.

Motivation :

Plant disease identification is a laborious task and at the same time less accurate and geographically limited. Whereas an automatic prediction technique will take less effort, less time and give more accurate results and could be easily accessible across global networks. This model will help the farmers detect the plant diseases so that they can be cured on time and can assure better plant growth.

Brief overview of our project :

Our model utilizes computer vision to classify plant leaves as healthy or diseased. By analyzing leaf images, it provides insights into disease causes, prevention, and treatment. Leveraging convolutional neural networks, it extracts intricate features from leaf images and accurately identifies various diseases across plant species. This empowers farmers with timely interventions to safeguard crop yields and ensure food security. Ultimately, our model revolutionizes plant disease management, supporting sustainable agriculture practices globally.

Objective

Early vision :

Developing a deep learning architecture to classify leaf images foremost into healthy or unhealthy and proceeding with categorizing unhealthy ones into the following –

- Rusty
- Powdery

Comparing the model's performance based on the Accuracies, Robustness to variations, Efficiency and Generalizability calculated on implementing various machine learning algorithms.

Review the results for various hyperparameters.

Aim for societal advancement :

Plant disease identification is a laborious task and at the same time less accurate and geographically limited. Our project will take less effort, less time and give more accurate results. This model will help the farmers/gardeners detect the plant diseases so that they can be cured on time and can assure better plant growth.

Literature Overview

a) About DataSet :

The plant disease detection dataset is a comprehensive collection of images labeled with information about the plant and the specific disease affecting it.

We are using two different type of data sets for plant leaf disease detection : Here, the dataset is mounted from the drive link :

1. **Data_Set 1 : 3 classes -> " Healthy, Powdery, Rusty " divided into 1322 training, 150 testing and 60 validation images.**

- **Kaggle - Kaggle-plant-disease-recognition-dataset**
- **Drive - Drive-plant-disease-recognition-dataset**

2. **Data_Set 2 : 38 classes, divided into 70329 training, 74 testing and 17421 validation images.**

- **Kaggle -Kaggle-new-plant-disease-recognition-dataset**
- **Drive - Drive-new-plant-disease-recognition-dataset**

b) About Models :

Here is a brief description of the models we've used for training for our image classification dataset :

1. VGG19 (Pre-Trained) :
    a. VGG19 has already learned intricate visual features from vast image datasets, making it adept at recognizing various patterns.
    b. However, VGG19's computational complexity can be prohibitive, requiring significant computational resources and time for training and inference, especially with large-scale datasets.
2. Sequential CNN

a. Sequential Convolutional Neural Networks (CNNs) are tailored for processing image data, excelling in capturing spatial dependencies and hierarchical patterns.
b. However, Sequential CNNs are prone to overfitting, particularly with small or noisy datasets, necessitating careful regularization techniques to ensure generalization.

3. Logistic Regression
    a. With probabilistic predictions, Logistic Regression offers simplicity and interpretability, making it valuable for binary classification tasks, including image classification in certain scenarios.
    b. However, its linear assumption may limit its effectiveness in capturing the nonlinear relationships inherent in complex image data.
4. LDA
    a. Linear Discriminant Analysis (LDA) is a dimensionality reduction technique and a classification algorithm where the classes are well-separated and the feature space is high-dimensional.
    b. However, LDA's performance may degrade when the assumptions of Gaussian distributions and equal class covariances are not met.
5. KNN
    a. K-Nearest Neighbors (KNN) is a powerful non-parametric classification algorithm based on the majority class of the k-nearest neighbors in the feature space, widely used in image classification tasks.
    b. However, KNN's computational complexity with large dataset and choice of the hyperparameter impact its performance, making it less efficient for large-scale image datasets.
6. Decision Tree Classifier
    a. Decision trees partition the feature space into simple decision regions based on feature values, making them easy to interpret and visualize. They are also robust to outliers and missing values.

    b. However, decision trees are prone to overfitting, particularly with deep trees and complex datasets.
7. Random Forest Classifier
    a. Random Forest Classifier is an ensemble learning method that combines multiple decision trees to improve classification accuracy and robustness.
    b. However, Random Forests may suffer from computational inefficiency and increased memory usage, especially with large numbers of trees.
8. GaussianNB
    a. Gaussian Naive Bayes (GaussianNB) is a probabilistic classifier based on Bayes' theorem assuming features are conditionally independent given the class, making it computationally efficient and well-suited for high-dimensional data.
    b. However, GaussianNB's assumption of feature independence may not hold in practice for complex image data
9. Support Vector Classifier (SVC)
    a. SVC is a powerful supervised learning algorithm that constructs a hyperplane or set of hyperplanes in a high-dimensional feature space to maximize the margin between classes, making it effective in capturing complex decision boundaries and handling non-linear data.
    b. However, SVC's performance heavily depends on the choice of the kernel function and its associated hyperparameters.
10. XGBoost Classifier
    a. XGBoost (Extreme Gradient Boosting) Classifier combines the strengths of gradient boosting and decision tree algorithms, offering high prediction accuracy, scalability, and robustness against overfitting.
    b. However, XGBoost's hyperparameter tuning can be challenging, requiring careful optimization to achieve optimal performance.

Project Plan

a) Milestones Targets :

i) Literature Survey

ii) Data Collection & Preprocessing

iii) Model Training & Evaluation

iv) Novel Architecture Development

v) Documentation

b) Management Plan :

i) Project ideation and Exploration:

1) Decided to build a Model on Plant Disease Detection as the project idea.
2) Explored various resources including YouTube tutorials, articles, Kaggle datasets, and GitHub repositories to gather insights and details for the selected project idea.

ii) Dataset Preparation and Model Selection:

1) Worked on a dataset comprising 3 classes divided into Train, Test, and Validation sets.
2) Experimented with various classifiers including Logistic Regression, Decision Trees, Random Forests, SVM, and CNN.
3) Found the CNN Model to be the most appropriate one, providing the highest accuracy among all classifiers tested.

iii) Model Evaluation and Reporting:

1) Implemented graphical representations of the model performance using confusion matrix, accuracy graphs, and classification reports.
2) Utilized these visualizations to analyze and interpret the model's behavior and performance.
3) Compiled the findings and insights into the Final Report and Presentation, summarizing the project's methodology, results, and conclusions.

c) Team Management :

i) Decision Making - Emphasized equal participation and open communication in project direction and decision-making. Held regular team meetings to discuss progress, challenges, and solutions.

ii) Task allocation and accountability - Established clear deadlines and milestones to ensure accountability and timely completion.

iii) Support and Collaboration - Encouraged knowledge sharing and skill development through peer learning and mentorship, where members freely shared ideas, offered assistance, and provided constructive feedback to one another.

Methodology

a) Data Collection :

This project utilized a publically accessible dataset available on Kaggle .

b) Data Preprocessing & Augmentation :

It involved several key steps to ensure the data is appropriately formatted and prepared for training a machine learning model. Typical preprocessing steps are Data collection, Image resizing, Data Augmentation, Normalization, Splitting the dataset, label encoding, data loading and more.

- Data Collection : Collect diverse images of healthy and diseased leaves.
- Image Resizing: Resize all images *(typically to 256x256 pixels)*to a uniform size to ensure consistency in input dimensions for the model.
- Data Augmentation: Augment the dataset by applying transformations such as rotation, flipping, zooming, and shifting to create variations of the original images. This helps to *reduce overfitting.*
- Normalization: Normalize pixel values to a common scale *(e.g., 0 to 1)* to ensure stability and improve convergence.
- Splitting the Dataset: Split dataset into training, validation, and test sets for training, hyperparameter tuning, and final model evaluation, respectively.
- Label Encoding: Encode class labels numerically., such as one-hot encoding, to facilitate model training.
- Data Loading: Use data generators or pipelines to efficiently load batches of preprocessed images into memory during model training. This helps in handling large datasets that may not fit into memory entirely.

d) Model Training :

Building a Convolutional Neural Network (CNN) involves stacking different layers to create a deep learning model capable of learning hierarchical representations of data.

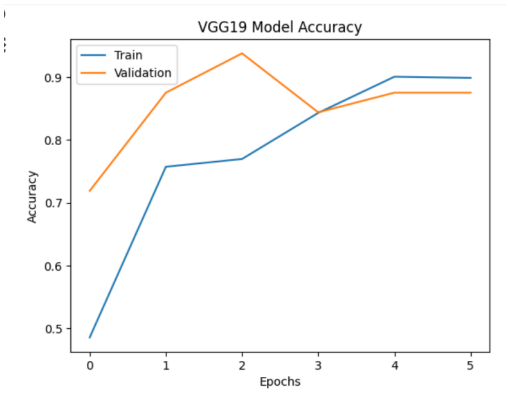
Here is the description of layers used in our model:

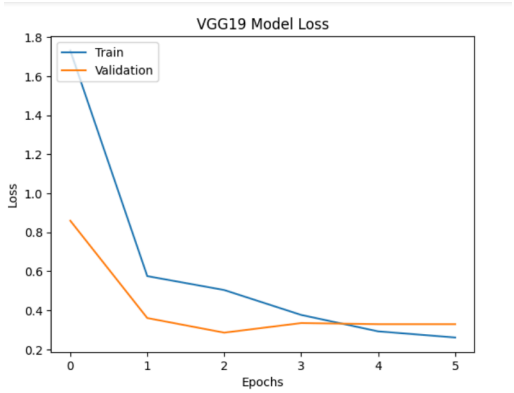
- Convolutional Layer (Conv2D): The Conv2D layer performs convolutional operations on the input image. The parameters used is explained as follows:
    i) 32: Number of filters/kernels to be used in the convolution operation.
    ii) (3, 3): Size of the convolutional kernel/filter.
    iii) activation='relu': Rectified Linear Unit (ReLU) activation function is applied to introduce non-linearity.
    iv) input_shape=(img_width, img_height, 3): Input shape of the images, which is (img_width, img_height, 3) representing width, height, and 3 channels (RGB).
- Max Pooling Layer (MaxPooling2D): The MaxPooling2D layer performs downsampling by taking the maximum value within a window.The parameters used is explained as follows:
    i) 32: Number of filters/kernels to be used in the convolution operation.
    ii) (2, 2): Pool size (2x2 window).
- Two additional Conv2D layers are added with increasing filter sizes (64 and 128). Each layer extracts more abstract features from the input images.
- Two additional MaxPooling2D layers follow the convolutional layers, performing downsampling to reduce spatial dimensions and computational complexity.
- The Flatten layer converts the 2D feature maps from the previous convolutional layers into a 1D feature vector. This prepares the data for input into the fully connected layers.
- Dense Layers (Fully Connected Layers):
    i) Two Dense layers are added for classification.
    ii) The first Dense layer has 128 units with ReLU activation, which introduces non-linearity to the network.
    iii) The second Dense layer has 3 units (equal to the number of classes) with softmax activation, which outputs the probabilities of each

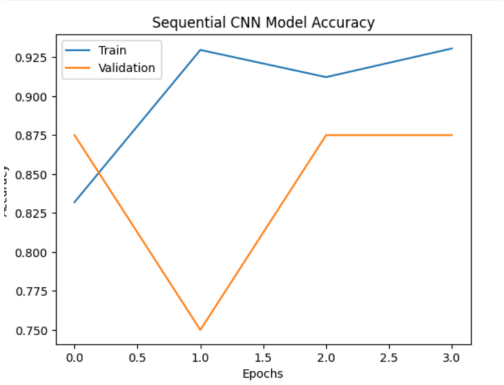
class. Softmax ensures that the sum of probabilities for all classes is 1, making it suitable for multi-class classification problems.

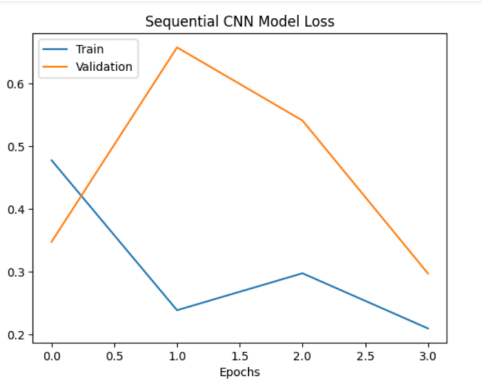
- Model Compilation
    1) Optimizer: Adam optimizer is used, which adapts the learning rate for each parameter, providing better convergence.
    2) Loss Function: The loss function measures the difference between the model's predictions and the actual target labels. Categorical cross-entropy loss is used since the target labels are one-hot encoded. It computes the cross-entropy loss between the true labels and the predicted probabilities.
    3) Metrics: Accuracy is used as the evaluation metric.
- Training
    - The model is trained using the fit_generator method.
    - 16 steps per epoch are used, considering the number of batches(32) in the training set.
    - Training is performed for 10 epochs.
    - Validation data is provided to monitor model performance.
    - EarlyStopping and ModelCheckpoint callbacks are utilized during training to ensure efficient training and save the best model.
    - Training and validation loss and accuracy are stored in history_seq.
    - Best model based on validation accuracy is saved as 'best_model_seq.h5' using ModelCheckpoint callback.

Glimpses of Training Our model









d) Model Evaluation :

- Accuracy Assessment: Evaluating the proportion of correctly classified samples.
- Confusion Matrix Analysis: Assessing the model's classification performance across different disease classes.
- Graphical Representation: Visualizing precision-recall curves, ROC curves, and class-wise accuracy graphs.
- Classification Report: Generating detailed metrics for precision, recall, F1-score, and support for each class.

Experimental Analysis

Accuracy

```
[44] vgg_acc = vgg_model.evaluate_generator(validation_set)[1]
     seq_acc = s_model.evaluate_generator(validation_set)[1]
     print(f"The accuracy of the VGG19 model is = {vgg_acc*100}%")
     print(f"The accuracy of the Sequential CNN model is = {seq_acc*100}%")
```

```
<ipython-input-44-685ef4b5661c>:1: UserWarning: `Model.evaluate_generator` i
  vgg_acc = vgg_model.evaluate_generator(validation_set)[1]
<ipython-input-44-685ef4b5661c>:2: UserWarning: `Model.evaluate_generator` i
  seq_acc = s_model.evaluate_generator(validation_set)[1]
The accuracy of the VGG19 model is = 94.9999988079071%
The accuracy of the Sequential CNN model is = 93.33333373069763%
```

```
Accuracy of Different Classifiers:
                        Classifier  Accuracy
0                Logistic Regression   0.46875
1  Linear Discriminant Analysis   0.65625
2                K Nearest Neighbors   0.37500
3                     Decision Tree   0.40625
4                     Random Forest   0.50000
5              Gaussian Naive Bayes   0.50000
6            Support Vector Machine   0.50000
7                           XGBoost   0.59375
```

Review of Models used

VGG19 has already learned intricate visual features from vast image datasets, making it adept at recognizing various patterns.

- Input Layer: Accepts input images (typically 256x256x3).
- Convolutional Layers (Conv1_1 to Conv5_4): Extract features using 3x3 filters with stride 1 and same padding.
- Max Pooling Layers (MaxPool): Downsample feature maps.
- Fully Connected Layers (FC6, FC7, FC8): Perform classification/regression tasks.
- Flatten Layers: Flatten feature maps into a vector.
- ReLU Activation: Introduce non-linearity after each layer.
- Model Parameters
    - Total parameters 20122691
    - Trainable parameters 98307

The sequential convolutional neural network (CNN) architecture consists of several layers:

- Convolutional Layers: Four Conv2D layers with increasing filter sizes (32, 64, 128) and (3, 3) kernel size, applying ReLU activation.
- Max Pooling Layers: Two MaxPooling2D layers with (2, 2) pool size.
- Flatten Layer: Converts the 2D feature maps into a 1D feature vector.
- Dense Layers:
    - Two Dense layers for classification.
    - The first Dense layer has 128 units with ReLU activation.
    - The second Dense layer has 3 units with softmax activation for multi-class classification.
- Model Parameters
    - Total parameters 14843878
    - Trainable parameters 14843878
    - No pre-trained weights used

In other classifiers dictionary we have used Logistic Regression, Linear Discriminant Analysis (LDA), K-Nearest Neighbors (KNN), Decision Tree Classifier, Random Forest Classifier, GaussianNB, Support Vector Classifier (SVC) and XGBoost Classifier.

For each classifier in the dictionary classifiers

- Cross-validation is performed using 5-fold cross-validation on the training data to evaluate the classifier's performance.
- The average cross-validation score is calculated.
- Each classifier is fitted to the entire training dataset.
- Predictions are generated on the validation dataset using the trained classifier.
- Confusion matrices are computed for each classifier's predictions compared to the true labels of the validation dataset.
- Accuracy scores are calculated for each classifier's predictions.
- Results, including classifier names and their corresponding accuracies, are stored for comparison.

Based on the confusion matrices:

- Linear Discriminant Analysis and Support Vector Machine show relatively balanced performance across classes.
- Decision Tree and K Nearest Neighbors struggle with misclassification and class imbalance.
- Logistic Regression, Random Forest, Gaussian Naive Bayes, and XGBoost demonstrate moderate performance but may require further tuning or handling of class imbalance for improved accuracy.

Conclusion

- Compared and contrasted deep learning models for plant disease prediction and built novel architectures for improved accuracy
- Literature survey laid the foundation for model selection with state-of-the-art techniques
- 1382 Leaf images split into training (1322) and external validation sets (60)
- Utilized 5 and 10-fold cross-validation for rigorous training and evaluation.
- Models demonstrated 3-class classification capabilities (Healthy, Powdery, Rusty)
- A new architecture having 3 residual blocks, a DSC block, 2 fire modules, a convolution layer with max-pooling, and dense layer (VGG19) was found most efficient.
- Accurate plant disease prediction improves plant outcomes and healthcare efficiency.
- Early detection and precise classification lead to timely interventions and better treatment planning
- Lasting positive impact on healthcare practices and inspiration for future deep learning research.

References

Articles :

- **https://www.mdpi.com/2073-4395/12/10/2395#:~:text=A%20CNN%2Dbased%20deep%20learning,classification%2C%20a%20CNN%20was%20used.**
- **https://github.com/ardendertat/Applied-Deep-Learning-with-Keras/tree/master**
- **https://www.frontiersin.org/journals/plant-science/articles/10.3389/fpls.2016.01419/full**

Data set 1 : https://www.kaggle.com/datasets/rashikrahmanpritom/plant-disease-recognition-dataset

Data set 2 : https://www.kaggle.com/datasets/vipoooool/new-plant-diseases-dataset/data

Youtube Resource :

- **https://youtu.be/amt9ZmGofJk?si=v4kQIy_pVjYCha4g**
- **https://youtu.be/148eu_foNo8?si=Ebb2LgFcAoyCx0vf**
- **https://youtu.be/zrHmtH8u3UM?si=PDDZV7__u9L8DCrr**