LEAF DISEASE DETECTION

Abhishek Malpotra & Siddharth Naidu

Leaf Disease Classification

Abstract

This project aims to develop an AI and ML-based system for accurate identification and classification of leaf diseases through image processing techniques. The system will analyze leaf images and symptoms to detect and diagnose diseases, enabling farmers to take appropriate measures for disease prevention and management, ultimately ensuring healthy crop yields. The project will focus on using deep learning algorithms and convolutional neural networks to accurately classify the images of the leaves into different categories of diseases. Challenges include the availability and quality of data, overfitting, interpretability. The project will provide a valuable tool for farmers to identify and manage leaf diseases, ultimately increasing crop yields and improving food security

Background

The background for the development of an AI and ML-based system for leaf disease identification lies in the growing need to improve crop yields and food security. Plant diseases can significantly reduce crop yields, and traditional methods of disease identification and management can be time-consuming and labor-intensive. By leveraging the power of AI and ML, it is possible to develop automated systems for disease identification and management that are more efficient and accurate, enabling farmers to take timely action to prevent and manage disease outbreaks. Advances in computer vision, deep learning, and image processing techniques have made it possible to accurately classify and identify plant diseases based on visual symptoms, leading to the development of such systems..

Detailed Approach

Data Collection

As part of this project, we needed to collect a dataset that included a wide range of images of healthy leaves as well as leaves with various types of diseases, captured under different lighting conditions and from different angles. The dataset comprises of:

1. Images of leaves affected by different stages of the disease, from early to advanced stages, as this can impact the accuracy of disease detection.
2. Equal number of images for each type of disease and for healthy leaves, to prevent bias towards certain categories.
3. Enough data points to allow for training a deep learning model with high accuracy.
4. Data source is: https://www.kaggle.com/datasets/dev523/leaf-disease-detection-dataset

Exploratory Data Analysis

A crucial step in understanding the data, the data was subjected to various analysis methods. The data exploration was was divided into two parts: Statistical analysis and Visualization

Statistical Analysis

Our statistical analysis involved a few steps.

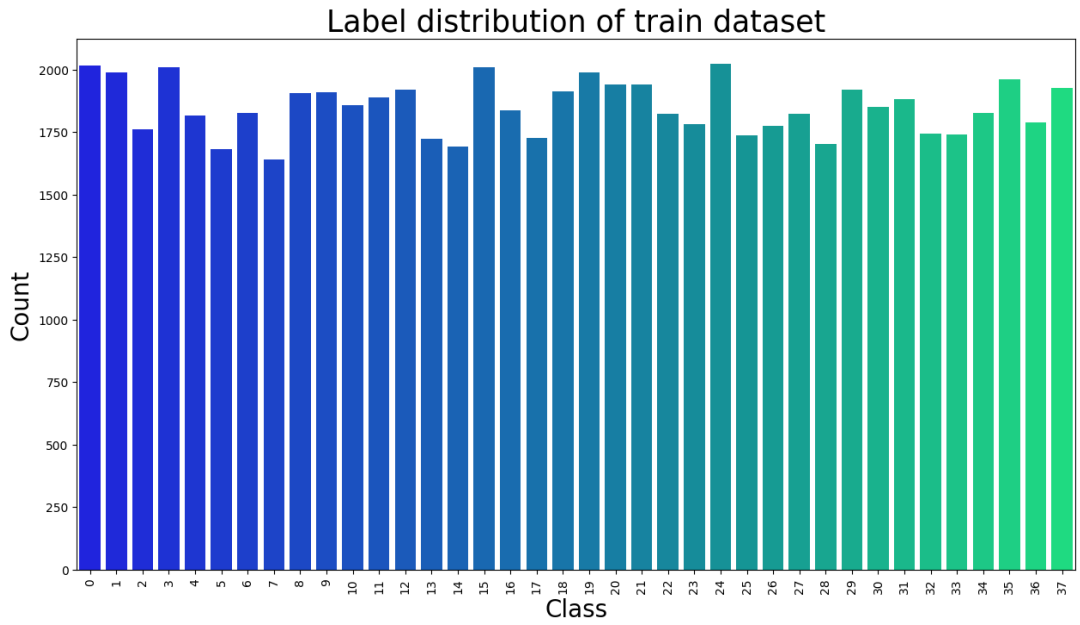
1. "Our data contains {num_classes} classes": This line simply prints out the number of unique classes or categories in the dataset. In the context of leaf disease identification, this would refer to the number of different types of diseases or conditions that the model is trained to identify.
2. "Our training data has {num_train_images} images": This line prints out the number of images in the training dataset. The training dataset is used to train the deep learning model to recognize patterns in the data. The number of images in each dataset is important to know as it can affect the model's ability to generalize to new, unseen images.

```
Found 70295 images belonging to 38 classes.
Found 17572 images belonging to 38 classes.
Our data contains 38 classes
Our training data has 70295 images
Our validation data has 17572 images
Shape of train data: (224, 224, 3)
Shape of validation data: (224, 224, 3)
```

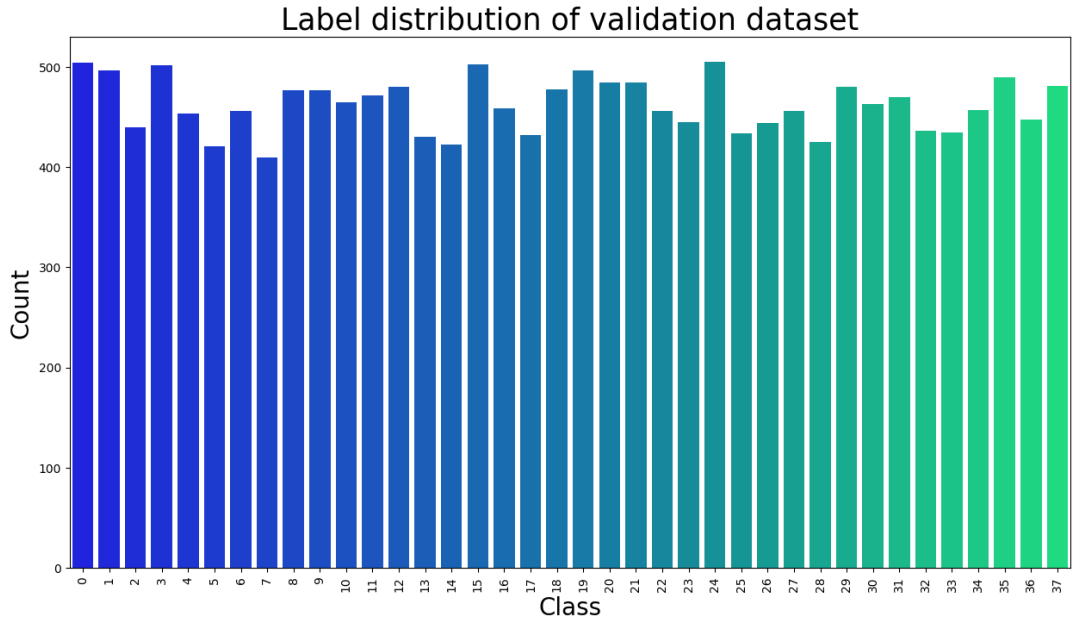
Visualization

For our visualizations, we plotted two count-plots to see whether the data is imbalanced

1. The label distribution of the training dataset is plotted using the "plot_label_distribution" function with the title "Label distribution of train dataset".



2. The label distribution of the validation dataset is plotted using the "plot_label_distribution" function with the title "Label distribution of validation dataset".



Data Preparation

Defined Constants: For the data preparation stage, we defined several constants that will be used throughout the project, including the image size, batch size, number of epochs, train and validation folders, and model path. These constants ensure consistency in the data preparation process and allow for easier modification in case of future changes.

Augmented data: To increase the diversity of the dataset, we applied data augmentation techniques to reduce overfitting and improve model performance by increasing the number of training examples.

Train and Validation set load: We loaded the training and validation set, ensuring that the dataset is balanced with an equal number of images for each class to prevent bias towards specific categories.

Modeling

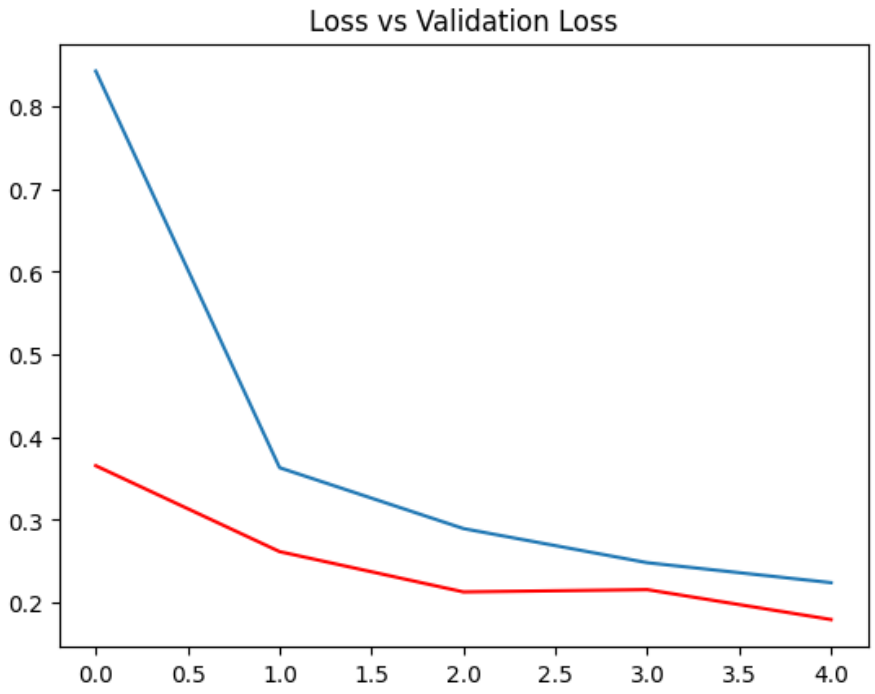
The model is based on Convolutional Neural Networks (CNNs) and uses the MobileNetV2 architecture, which is pre-trained on a large image dataset. The weights are defaulted to the pre-trained values and include_top is set to False, meaning that the last fully connected layer is not included.

The images are represented in RGB channels, and the model has the ability to train the pre-trained model set to False. Additional layers were added, including an average pooling layer, a dropout layer, and several dense layers for classification.

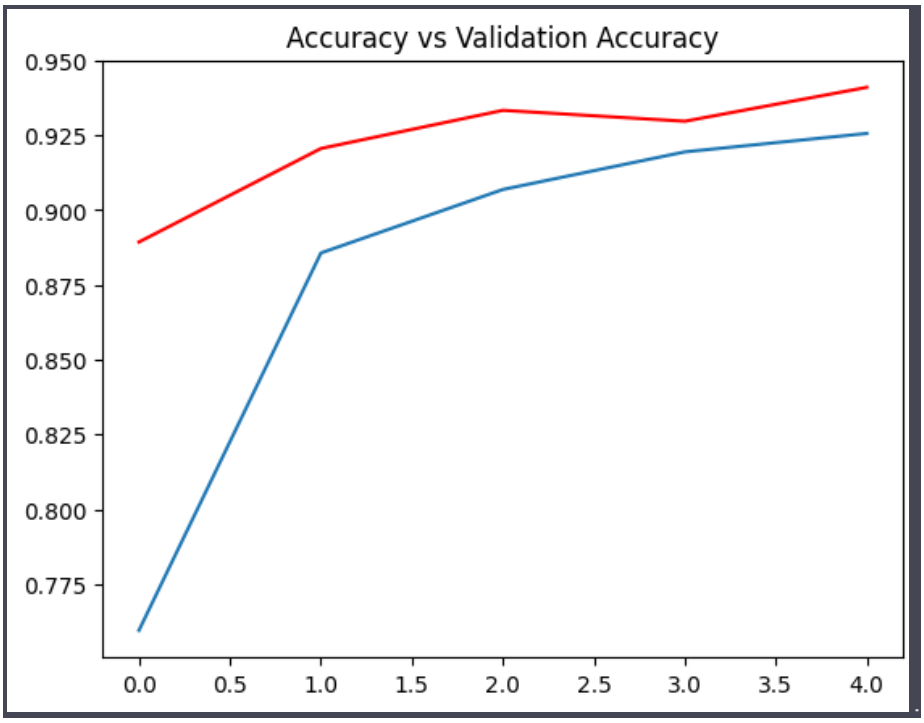
To finalize the model, the pre-trained model was combined with the additional layers for feature extraction and classification. The model was then compiled using a specified optimizer, loss function, and metrics to evaluate the model's performance.

Model Evaluation

The Loss across Epochs graph shows the performance of the model over the training and validation datasets. The graph displays how the model's loss (i.e., the difference between the predicted and actual values) decreases with each epoch, indicating that the model is improving its accuracy over time. The graph also shows the validation loss, which is the model's performance on the unseen data. A decreasing validation loss indicates that the model is generalizing well to new data.



The Accuracy across Epochs graph shows the model's accuracy over the training and validation datasets. The graph displays how the model's accuracy (i.e., the percentage of correct predictions) increases with each epoch, indicating that the model is improving its performance over time. The graph also shows the validation accuracy, which is the model's accuracy on the unseen data. An increasing validation accuracy indicates that the model is generalizing well to new data



Deployment

The deployment phase of the leaf disease detection project utilized the Flask web framework to create a user-friendly interface for our application. The following sections detail the implementation of the Flask application, which incorporates the pre-trained CNN model and provides a seamless experience for users to diagnose plant diseases.

Importing Necessary Libraries

The application imports essential libraries such as os, cv2, numpy, and Flask, as well as the TensorFlow Keras library for loading the model and processing images.

Loading the Pre-Trained Model

The trained CNN model is loaded using the load_model function from the TensorFlow Keras library. Constants such as UPLOAD_FOLDER and ALLOWED_EXTENSIONS are defined to specify the upload directory and allowed image formats, respectively. The Flask app is initialized and configured with the necessary settings.

Mapping Class Indices to Disease Names

A dictionary, class_indices, is created to map class indices to their corresponding leaf disease names. This mapping ensures that the model's output is translated into human-readable disease names.

Helper Functions

Two helper functions are defined to streamline the prediction process:

1. allowed_file: This function checks if the uploaded file has a valid extension (i.e., 'png', 'jpg', 'jpeg').
2. prediction: This function takes an image path and the pre-trained model as inputs and generates the disease prediction.

z

Flask Routes and Functions

The Flask app has two routes:

1. Root Route ('/'): This route renders the homepage, which consists of an image upload form.
2. Predict Route ('/predict'): This route handles the image uploading and prediction process. The predict function associated with the '/predict' route checks if the uploaded file is allowed, saves it securely to the upload folder, and then uses the prediction helper function to generate the disease prediction. The result is then displayed on the homepage along with the uploaded image.

Running the Flask Application

Finally, the Flask app is run with the debug=True flag, enabling developers to identify and fix any issues that may arise during deployment. This approach integrates the pre-trained CNN model with the Flask web framework to create an accessible and efficient leaf disease detection application, demonstrating the potential for advanced AI techniques in practical, real-world scenarios.

Results Explanation

We trained a convolutional neural network based on the MobileNetV2 architecture to classify leaf images for 38 different plant diseases. We augmented the data to increase diversity and trained the model on 70295 images with 5 epochs. The weights were set to pre-trained values and include_top was set to False. We also added additional layers of average pooling, dropout, and dense layers for classification. We achieved an accuracy of 93.97% on the validation set, with a loss of 0.1853 after the final epoch.

Conclusions

Summary

Based on the results obtained in this project, we can conclude that using convolutional neural networks and the MobileNetV2 architecture for leaf disease identification is an effective approach. The model achieved high accuracy in both training and validation sets, with a final validation accuracy of 93.97% after five epochs. These results demonstrate the potential of using machine learning for automated plant disease diagnosis, which can greatly benefit farmers by enabling early disease detection and targeted management.

However, further improvements could be made by increasing the diversity and size of the dataset, and exploring different architectures and hyperparameters.

Challenges and Limitations

1. **Data Quality**: The accuracy of the model depends on the quality of the data used for training, and the data used may not be representative of plant species from all over the world. For instance, the dataset used for training the model may only contain images of plant diseases prevalent in certain regions, thus reducing its effectiveness in identifying plant diseases that are not present in the dataset.
2. **Overfitting**: One of the major challenges with deep learning models is overfitting, which can lead to limited performance on new data. This occurs when the model is too complex and is trained to recognize patterns specific to the training data, rather than learning more generalizable features. As a result, the model may not perform well when tested on new, unseen data.
3. **Transparency**: Deep learning models are often complex and can be difficult to interpret, making it challenging to understand how the model is making predictions. As a result, it can be difficult to identify the specific features or factors that contribute to the model's decision-making process.

Future Work

Some potential future work for this project could include:

1. **Incorporating additional data sources:** Collecting more diverse and representative data from different regions around the world to improve the model's ability to accurately classify a wider range of plant diseases.
2. **Exploring different model architectures:** While the MobileNetV2 architecture used in this project performed well, there are many other deep learning architectures that could be explored to potentially improve performance further.
3. **Developing a mobile application:** Creating a user-friendly mobile application for farmers to easily take pictures of their plants and receive real-time disease diagnosis and prevention recommendations.
4. **Integrating with precision agriculture technologies:** Integrating the disease identification model with precision agriculture technologies, such as automated irrigation systems, to provide targeted treatments to affected plants.
5. **Continuing to refine and optimize the model:** Further fine-tuning and hyperparameter optimization can improve the model's accuracy and reduce the risk of overfitting

References

1. https://www.sciencedirect.com/science/article/pii/S2666285X22000218
2. https://www.frontiersin.org/articles/10.3389/fpls.2016.01419/full
3. https://www.mdpi.com/2073-4395/12/10/2395
4. https://plantmethods.biomedcentral.com/articles/10.1186/s13007-021-00722-9
5. https://www.nature.com/articles/s41598-022-21498-5
6. https://www.kaggle.com/datasets/dev523/leaf-disease-detection-dataset