DEEP LEARNING FINAL PROJECT REPORT

CSCI 5931

SPRING 2024

Sankeerthan Sornapoodi

111101112

DENVER, CO, UNITED STATES

CROP DISEASE DETECTION USING DEEP LEARNING TECHNIQUES

1.Abstract

Crop disease detection is an essential component of contemporary agriculture that affects global economic stability and food security. The necessity for automated and precise detection systems is underscored by the fact that conventional illness diagnosis techniques are frequently labor-intensive and subject to human mistakes. In this field, deep learning approaches have showed promise by using sophisticated neural network designs to evaluate plant photos and accurately identify symptoms of disease,we develop a Convolutional Neural Network (CNN) architecture for the detection of plant diseases using images of plant leaves. a predictive system is built, allowing users to input an image of a plant leaf, and the system predicts the class of disease present based on the trained model. It provides a comprehensive framework for leveraging deep learning techniques to address the critical challenge of crop disease detection in agriculture.

Keywords: Crop,Disease,diagnosis,neural network,plants,deep learning model,datasets.

2.Problem Statement

The world relies on agriculture for food, and various synthetics, so healthy lifecycle of crops is much more important than we give credit for, in many countries, especially agrarian ones where the agriculture plays a dominant role in growth of a country’s economy,Most of the world’s food and fabrics comes from agriculture, and agriculture when combined with technology brings out the finest results. Crop disease detection systems can solve multi-fold problems, saving farmer resources, preventing economic loss, early detection of diseases etc. In this project we aim to design a deep learning model that can detect the crop disease with higher accuracy h.Crop disease detection systems can address a variety of issues, including early disease identification, preventing financial loss, and conserving farmer resources. With the help of a simple photo of a plant's leaf, wn for a deep learning model that will be able to identify crop diseases in real-time environments with greater accuracy.

3.Introduction

The problem is interesting because of its complex intricate nature of detecting plant diseases that are fast spreading, hard to detect, and the mutations and resistance they develop against the medications. The difficulty in identifying rapidly spreading, difficult-to-detect plant illnesses, as well as the mutations and drug resistance they generate, Diseases like blight, black spot, powdery mildew, and downy mildew vary in stages and are typically undetectable until much later. This results in complete losses for farmers and the agriculture sector, delaying significant agro-economic losses.The problem may be more difficult to apply in deep learning since; to train the model, we must factor enormous datasets of crops and leaves in both healthy and diseased states. The limiting factor is that, to achieve high accuracy, we need multiple images taken from different angles, with different lighting and environmental conditions, and with different disease types. The Huge datasets, reliable model training, precise crop, condition, and disease classifications, and the ability to predict the best course of action are all necessary for the deployment of such a model.

Plant diseases may also impact the environment if more pesticides are required to contain outbreaks. The project lessens the environmental effect of crop management by supporting more focused and sustainable agriculture practices through the facilitation of early disease identification. While we aren't used to understanding how illnesses affect crops and we utilize user-friendly solutions, a variety of crops—both genetically and naturally grown—spark curiosity. The initiative intends to democratize access to cutting-edge agricultural technology, helping farmers of all sizes.

4.Literature Review

1.DeepCrop-Deep Learning based crop disease prediction with web application.

https://www.sciencedirect.com/science/article/pii/S2666154323002715#:~:

In order to achieve a higher accuracy rate, a novel deep learning-based model for crop disease identification was proposed in this project. ResNet-50 was used to identify the diseases, and the data was extracted using a variety of image processing and machine learning approaches.A web application was developed that can analyze the data from the image, and a real-world dataset containing 10,000 images of different crops, including potatoes, tomatoes, and spinach, at different stages of their lives, was also created. Multiple deep learning models, including resnet50, vgg, and others, have been used to create a framework that evaluates performance.However, the model's inability to localize the area of leaf disease diagnosis can be further enhanced by implementing a hybrid deep-learning architecture.

2.PlantDiseaseNet: Two-stage novel architecture.

Solving current limitations of Deep Learning based approaches for plant disease detection.

https://www.mdpi.com/2073-8994/11/7/939

Summary:

In this paper, a novel two-stage architecture was proposed that used one of the largest dataset containing the largest dataset of plant images—more than 79,925 leaf images—that were taken in a variety of lighting and environmental conditions was used in this paper to propose a novel two-stage architecture. The images were enhanced in two ways: first, using a traditional method that involved blurring and interrupting the images to introduce distortion to the photos, and second, training generative adversarial networks to generate syntactic data based on the datasets. After training and testing the datasets, a novel real-world architecture centered on plant disease detection was introduced.In order to recognize and classify leaves by species and attempt to detect diseases in real time, the architecture employs two phases, PDNET1 and PDNET2, but typically results in accurate plant leaf detection and reduced disease detection. The trained model achieved an accuracy of 93.67%.

3.Deep Learning based approach for automated plant disease classification using vision transformer.

https://www.nature.com/articles/s41598-022-15163-0

Summary:

In order to provide farmers with visual information for preventive measures, this work provides a lightweight deep learning strategy for real-time automated plant disease categorization utilizing Vision Transformer (ViT). Many models are trained and assessed on numerous datasets, including ViT, traditional CNN techniques, and CNN plus ViT combinations. Attention barriers decrease down prediction speed but boost accuracy. CNN blocks combined with attention blocks can make up for speed without compromising accuracy. Both hybrid models and convolutional-based and ViT-based architectures were compared. Class inequalities and dataset limits are challenges. The ViT-based model achieves better accuracy than CNN and hybrid models with fewer parameters, equal or even exceeding published findings. On all devices, attention blocks perform less quickly than convolutional blocks. integrating convolutional and attentional blocks Combining attention and convolutional blocks achieves faster prediction with higher accuracy,particularly in complex datasets like Plant Village and Wheat Rust Classification Dataset (WRCD). Theorder of this combination has minimal impact on prediction speed, but using attention blocks followed by convolutional blocks slightly improves accuracy.

4.MobileNet:Plant Leaf Diseases Detection using Deep Learning algorithms

https://link.springer.com/chapter/10.1007/978-981-19-5868-7_17

Summary:

This paper addresses financial losses, this study tackles the problem of plant leaf diseases and damaging insects in agriculture by suggesting quicker and more precise prediction techniques. The researchers created conventional neural network (CNN) models to recognize and diagnose plant leaf diseases using simple photos of both healthy and injured plants, taking advantage of current developments in deep learning.

To identify diseases in leaf photos, four deep learning models were used: Inception-v3, MobileNet, AlexNet, and basic sequential model. For training and testing, a new plant diseases dataset with 38 different classes—including simple leaf photos of healthy and diseased plants—was utilized.Furthermore, the trained models were applied to Internet-sourced plant leaf photos for testing. The MobileNet model outperformed the other models that were examined on the new plant. Among the evaluated models, the MobileNet model demonstrated high performance on the new plant diseases dataset, achieving training and validation accuracies of 99.07% and 97.52%, respectively.

5.Proposed Methodology

5.1 Dataset

The PlantVillage dataset stands as a publicly accessible resource offering a diverse array of plant disease categories. This dataset includes 38 classes, featuring a comprehensive collection of 54,305 images. In our experimental setup, we meticulously partitioned this dataset into distinct subsets for training, testing, and validation purposes. Specifically, the pre-trained models underwent training utilizing 80% of the PlantVillage dataset, while the remaining 20% was allocated for validation and testing. Within this framework, the total count of samples attributed to plant classes amounted to 54,305. Of these, 43,955 samples were earmarked for training, with 4,902 samples set aside for validation, and 5,488 samples designated for testing. Importantly, each of these delineated sets encapsulates the entirety of the 38 classes spanning various plant diseases.

5.2 Data Preprocessing

This demonstrates the initial preprocessing steps for a grape leaf image from the PlantVillage dataset. It employs the Matplotlib library to read and visualize the image. By using matplotlib.image.imread, the image is loaded from the specified file path. Then, imshow functions from Matplotlib are utilized to display the image on the plot. Additionally, the shape of the image array is printed, providing insight into its dimensions, such as height, width, and color channels. This preprocessing step is crucial for understanding the structure and characteristics of the input image before applying further analysis or feeding it into a deep learning model for disease detection

5.3 Data Curation

This focuses on data curation, specifically loading an image from a file path using Matplotlib's imread function. The image is then printed to display its parameters, such as pixel values and color channels. Afterward, parameters such as img_size and batch_size are initialized. These parameters are essential for preprocessing and feeding the image data into a deep learning model. img_size typically represents the desired dimensions of the input image, while batch_size determines the number of images processed simultaneously during training or inference.

5.4 Training Testing and Splitting

This prepares image data for training and validation by utilizing the ImageDataGenerator class from the Keras library. :

Image Data Generators Initialization: An instance of ImageDataGenerator is created to generate batches of tensor image data. It's configured with various parameters, including rescale set to 1./255, which normalizes pixel values to the range [0,1], and validation_split set to 0.2, indicating that 20% of the data will be reserved for validation.

Training and Validation Data Generation:

Train Generator: The flow_from_directory method is used to create a generator for training data. It specifies the directory containing the images (base_dir), the target size for resizing the images to (img_size, img_size), the batch size (batch_size), and sets the subset parameter to 'training' to indicate that it's generating data for training. Additionally, class_mode is set to 'categorical' to handle multi-class classification tasks.

Validation Generator: Another generator is created for validation data using the same flow_from_directory method. The only difference is that the subset parameter is set to 'validation', indicating that it's generating data for validation.

Data Availability Information: The output displays the number of images found in each subset (Found), along with the number of classes detected in the dataset. This information is crucial for ensuring that the data splitting process has been executed correctly and that both training and validation sets are adequately populated.

By utilizing these data generators, the code streamlines the process of loading, preprocessing, and augmenting image data, making it suitable for training deep learning models for image classification tasks, such as crop disease detection.

5.5 Building Convolutional Neural network

This is a Convolutional Neural Network (CNN) model for image classification, specifically for detecting crop diseases.

Model Architecture: The CNN model is defined using the Sequential API provided by Keras. It consists of several layers:

Convolutional Layers: Two convolutional layers are added using the Conv2D function. The first convolutional layer has 32 filters of size (3, 3), and the second convolutional layer has 64 filters of the same size. ReLU activation functions are applied after each convolution operation.

Max Pooling Layers: After each convolutional layer, a max-pooling layer with a pool size of (2, 2) is added using the MaxPooling2D function. This helps in reducing the spatial dimensions of the feature maps while retaining the most important information.

Flatten Layer: The output of the last convolutional layer is flattened into a 1D array using the Flatten layer. This prepares the data for input into the fully connected layers.

Fully Connected Layers: Two dense (fully connected) layers are added using the Dense function. The first dense layer has 256 units with ReLU activation, and the output layer has units equal to the number of classes in the dataset, with softmax activation. Softmax ensures that the output values represent class probabilities.

Model Summary: The summary method is called on the model to display a summary of its architecture, including the type of each layer, the output shape of each layer, and the number of parameters (weights and biases) associated with each layer. This summary provides insights into the model's structure and complexity.

Parameter Calculation: The total number of parameters in the model is calculated, including trainable parameters and non-trainable parameters. This information is essential for understanding the computational requirements of the model and optimizing its performance.

CNN architecture is suitable for processing image data and performing multi-class classification tasks, such as detecting various types of crop diseases based on leaf images. The model's architecture is designed to extract relevant features from input images and make accurate predictions about the presence of different diseases.

5.6 Model Training

This is responsible for training the previously defined Convolutional Neural Network (CNN) model using the specified training and validation data.

Training Process: The fit method is called on the model object to start the training process. This method takes several arguments:

train_generator: The generator object that generates batches of training data. It provides the model with batches of input samples and their corresponding target labels during training.

steps_per_epoch: The number of steps (batches) to yield from the generator before declaring one epoch finished. This value is calculated as the total number of samples in the training set divided by the batch size.

epochs: The number of epochs (iterations over the entire training dataset) to train the model.

validation_data: The generator object for validation data, providing batches of validation samples and their labels.

validation_steps: The number of steps (batches) to yield from the validation generator in each epoch. Similar to steps_per_epoch, this value is calculated based on the total number of samples in the validation set and the batch size.

Training Progress: During training, the model prints progress updates for each epoch, including the epoch number, the average loss, and the accuracy on both the training and validation datasets. These metrics are computed based on the performance of the model on the current batch of data.

Epoch Results: After each epoch, the training and validation loss values (measured using categorical cross-entropy) and accuracy values are displayed. These values provide insights into the model's performance and its ability to generalize to unseen data.

Training Duration: The time taken for each epoch (in seconds) is also displayed. This information helps monitor the training progress and identify potential issues, such as slow convergence or overfitting.

this code efficiently trains the CNN model on the provided dataset, iteratively updating its weights and biases to minimize the loss function and improve accuracy on both the training and validation datasets. The training process typically involves multiple epochs to allow the model to learn complex patterns in the data and achieve optimal performance.

5.7 Model Evaluation

This is responsible for evaluating the trained model's performance on the validation dataset.

Model Evaluation: The evaluate method is called on the model object to assess its performance on the validation data. This method takes the validation generator as input and computes the loss and accuracy metrics on the provided validation dataset. The steps argument specifies the number of batches to process from the generator.

Validation Results: After evaluation, the validation loss and accuracy are printed to the console. The validation accuracy is a crucial metric that indicates the model's ability to correctly classify unseen data samples from the validation set. It is calculated as the ratio of correctly classified samples to the total number of samples in the validation set, expressed as a percentage.

Output: The validation accuracy is displayed as a percentage, providing insight into the model's overall performance on unseen data. A higher validation accuracy indicates better generalization ability and suggests that the model has learned meaningful patterns from the training data.

In this specific case, the trained model achieved a validation accuracy of 88.6%, indicating that it correctly classified approximately 88.6% of the samples in the validation dataset. This result demonstrates the effectiveness of the trained model in accurately identifying plant diseases based on input leaf images.

5.8 Building a predictive system

This defines functions for building a predictive system using a trained deep learning model.
Load and Preprocess Image Function (load_and_preprocess_image):
This function takes an image path as input and loads the image using the Pillow library (Image.open).
It resizes the image to the specified target size using the resize method.

The image is then converted to a NumPy array using np.array.

A batch dimension is added to the image array using np.expand_dims.
Finally, the pixel values of the image are scaled to the range [0, 1] by dividing by 255 and converting to float32.

Predict Image Class Function (predict_image_class):
This function takes the trained model, image path, and class indices as input.
It preprocesses the image using the load_and_preprocess_image function.

The preprocessed image is passed to the model's predict method to obtain class probability predictions.

The index of the class with the highest probability is extracted using np.argmax.

The predicted class name corresponding to the index is retrieved from the provided class indices dictionary.

The predicted class name is returned as the output of the function.

These functions enable the prediction of the class label for an input image using a trained deep learning model. They encapsulate the image preprocessing and prediction steps, making it easy to integrate the model into a predictive system for plant disease detection.

5.9 Executing test images

This demonstrates the practical implementation of a predictive system for plant disease detection using a trained deep learning model. It includes:

Loading and preprocessing an input image using the Pillow library.

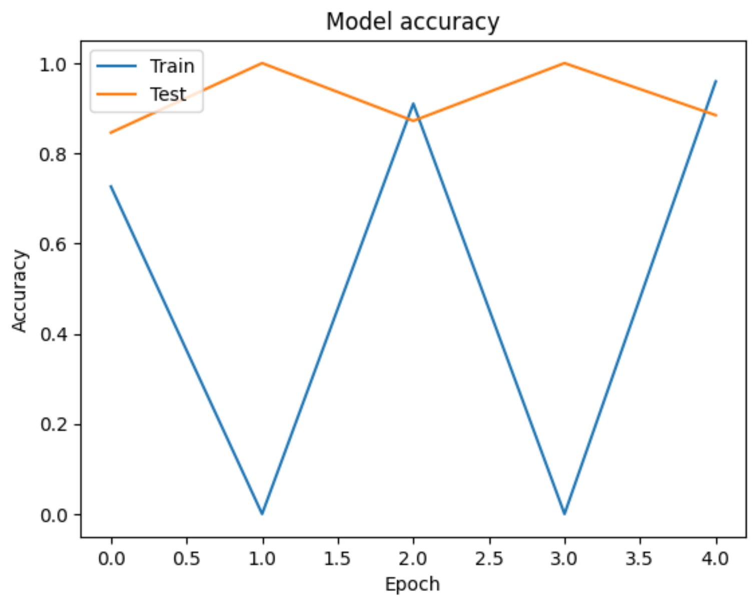
Predicting the class label of the preprocessed image using the trained convolutional neural network (CNN) model.

Outputting the predicted class name, which represents the detected plant disease, to the console.

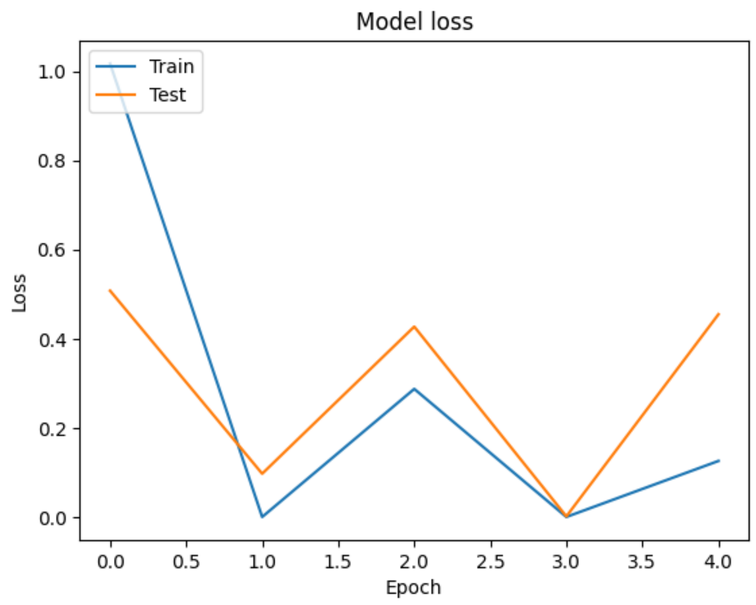
The predict_image_class function encapsulates the prediction process, where the input image undergoes preprocessing before being fed into the CNN model for inference. The predicted class name is then determined based on the highest probability prediction from the model output. Finally, the predicted class name is printed as the result of the prediction process.

6.Results

A deep learning model for the precise and effective identification of plant diseases based on input photos of plant leaves can be developed. Enhancing crop health management techniques, optimizing resource allocation, and facilitating informed decision-making can all be made possible by the system's high accuracy, real-time detection capabilities, and intuitive interface, which ultimately contribute to increased production and sustainable agriculture.

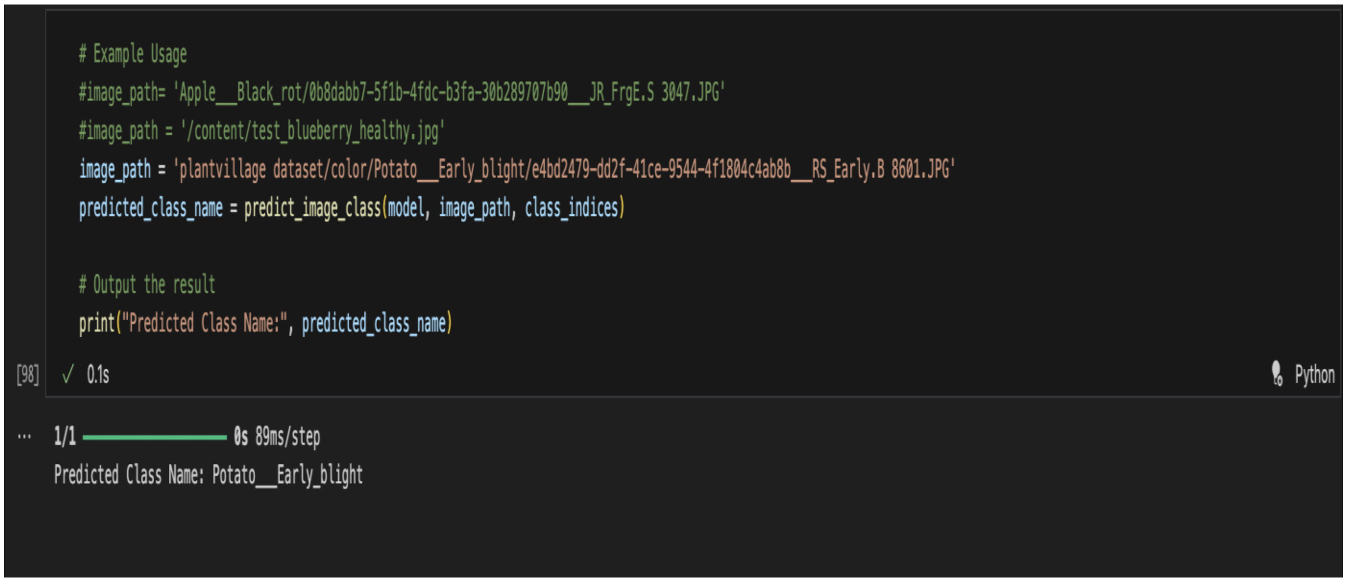


Graph demonstrating Training and Testing Model Accuracy



Graph demonstrating Loss function of the model for every epoch

OUTPUT

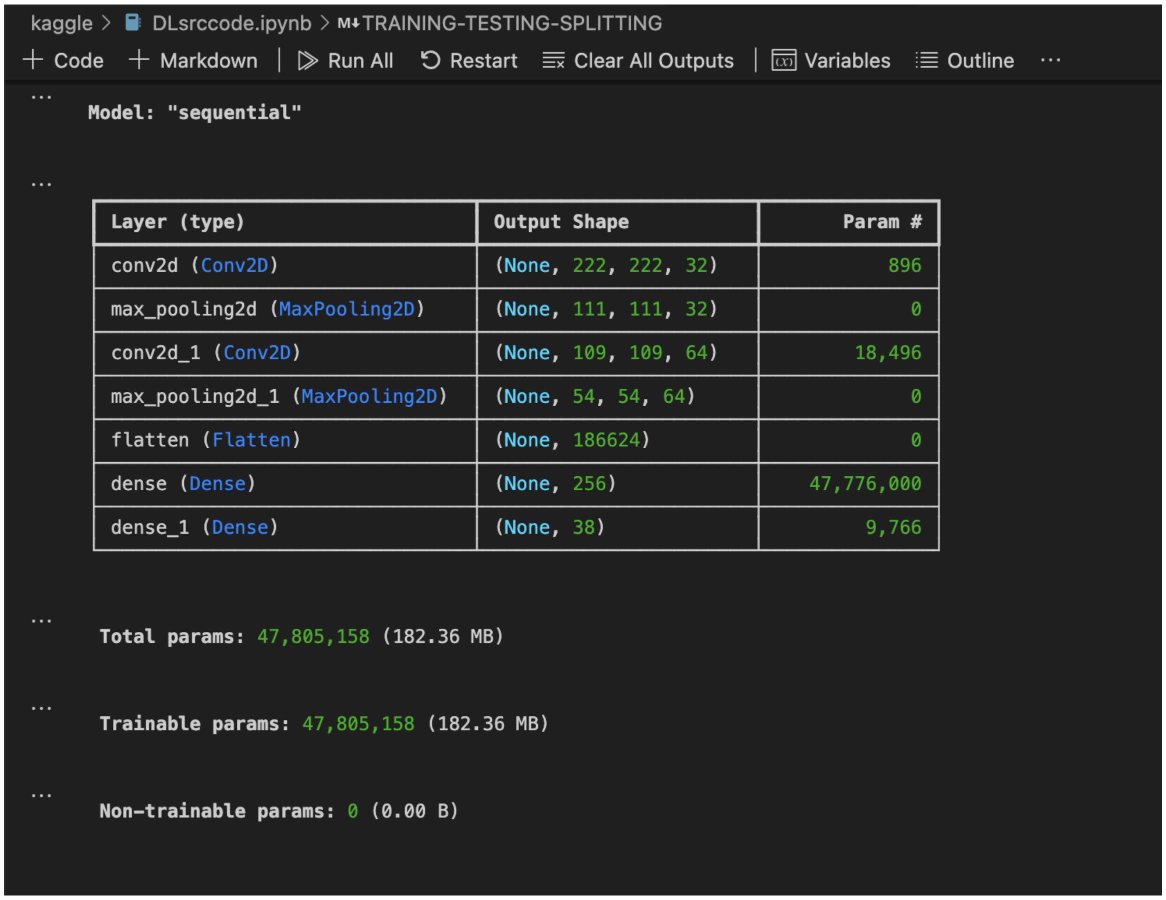


```
# Example Usage
#image_path= 'Apple__Black_rot/0b8dabb7-5f1b-4fdc-b3fa-30b289707b90__JR_FrgE.S 3047.JPG'
#image_path = '/content/test_blueberry_healthy.jpg'
image_path = 'plantvillage dataset/color/Potato__Early_blight/e4bd2479-dd2f-41ce-9544-4f1804c4ab8b__RS_Early.B 8601.JPG'
predicted_class_name = predict_image_class(model, image_path, class_indices)

# Output the result
print("Predicted Class Name:", predicted_class_name)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 89ms/step
Predicted Class Name: Potato__Early_blight
```

Executing the code based on test images.


kaggle > DLsrccode.ipynb > M↓TRAINING-TESTING-SPLITTING
+ Code + Markdown | Run All Restart Clear All Outputs | Variables Outline ···

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 222, 222, 32)	896
max_pooling2d (MaxPooling2D)	(None, 111, 111, 32)	0
conv2d_1 (Conv2D)	(None, 109, 109, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 54, 54, 64)	0
flatten (Flatten)	(None, 186624)	0
dense (Dense)	(None, 256)	47,776,000
dense_1 (Dense)	(None, 38)	9,766

```
Total params: 47,805,158 (182.36 MB)

Trainable params: 47,805,158 (182.36 MB)

Non-trainable params: 0 (0.00 B)
```

Convolutional Neural Network Model.

7.Future Work

- The future work in this domain, would be to create better neural models that have access to larger datasets, with images trained in real time in all environments,lightings and angles,This would ensure that the system has better accuracy, and metrics that would help enable better results, in regards to detection, classification of plant diseases.
- There has to be more research in how the plants demonstrate diseases in various conditions, and climates, and the data has to be labelled and created into a bigger dataset,that can be used during training of the deep learning models.
- More emphasis on how to integrate this model into a real time web/mobile application,so it can be used by farmers, and agricultural industry,in real time scenarios, and the data can be gathered and studied to improve the efficiency of the model.

Technical Aspects(Future Work)

- Enhanced Model Architectures: Continued exploration and development of deep learning architectures tailored specifically for crop disease detection. This may involve novel network architectures, such as attention mechanisms or graph neural networks, to capture intricate patterns and dependencies within plant images more effectively.
- Data Augmentation and Synthesis: Further investigation into advanced data augmentation and synthesis techniques to address data scarcity issues. This could involve generating synthetic images using generative adversarial networks (GANs) or exploring unsupervised learning approaches for feature extraction.
- Transfer Learning and Domain Adaptation: Exploration of transfer learning strategies to leverage pre-trained models from related domains, such as general object recognition or medical imaging, and adapt them to crop disease detection tasks. Domain adaptation techniques can also be investigated to enhance model generalization across different environmental conditions and plant species.
- Interactive and Explainable Systems: Designing interactive and explainable crop disease detection systems that facilitate user interaction and provide transparent insights into model predictions. Incorporating attention mechanisms or saliency maps can help highlight regions of interest in images and improve model interpretability.
- Real-Time Deployment and Edge Computing: Investigation of lightweight model architectures and optimization techniques for real-time deployment of crop disease detection systems on edge devices, such as drones or smartphones. This involves efficient model compression, quantization, and hardware acceleration to minimize computational resources while maintaining performance.
- Large-Scale Deployment and Collaboration: Scaling up crop disease detection solutions for large-scale deployment in agricultural settings, potentially through collaborative platforms that enable data sharing and knowledge transfer among farmers, researchers, and agricultural stakeholders. This can facilitate continuous model improvement and adaptation to diverse farming practices and regions

8.Conclusion

In conclusion, the development and application of deep learning techniques for crop disease detection hold tremendous promise for revolutionizing modern agriculture. Through the analysis of various deep learning models and methodologies, it becomes evident that these technologies offer effective solutions to mitigate the challenges posed by plant diseases, ultimately benefiting farmers, agricultural stakeholders, and global food security.

The utilization of datasets such as PlantVillage, with its extensive collection of labeled images spanning multiple plant species and disease classes, underscores the importance of robust data resources in training accurate and generalizable deep learning models. Moreover, the experimental findings presented in this study demonstrate the efficacy of leveraging pre-trained models, advanced architectures, and data augmentation techniques to enhance disease detection accuracy and scalability.

Looking forward, future research endeavors in this field should focus on advancing model architectures, incorporating multi-modal data sources, addressing data scarcity through synthesis and augmentation, and promoting model interpretability and real-time deployment on edge devices. Furthermore, collaboration and knowledge-sharing platforms can foster a collaborative ecosystem where farmers, researchers, and industry experts collectively contribute to the continuous improvement and widespread adoption of crop disease detection solutions.

9.References & Citations

1. https://www.kaggle.com/datasets/abdallahalidev/plantvillage-dataset
2. Plant disease identification using Deep Learning: A reviewResearchGatehttps://www.researchgate.net › ... › PLANT DISEASES
3. Revolutionizing crop disease detection with computational ...National Institutes of Health (NIH) (.gov)https://www.ncbi.nlm.nih.gov › articles › PMC10894121Revolutionizing crop disease detection with computational ...National Institutes of Health (NIH) (.gov)https://www.ncbi.nlm.nih.gov › articles › PMC10894121
4. A versatile deep-learning model for accurate prediction of ...Phys.orghttps://phys.org › news › 2023-04-versatile-deep-l...
5. PlantVillageImage:imgres