RCOEM
Shri Ramdeobaba College of
Engineering and Management, Nagpur

Shri Ramdeobaba College of Engineering & Management, Nagpur

Department of Computer Science & Engineering(AIML)

Session 2024-25

SEMINAR

ON

ML-DRIVEN CROP DISEASE PREDICTION SYSTEM

Presented By:
Group No:5
Group members:
1) SUJAL TRIVEDI
2) ARYA WANKHADE
3) VINAYAK SAHU
4) AKASH PUROHIT

Guided By:
(Dr.Amit Pimpalkar)

Introduction

Project Overview: This project develops a machine learning-based system to detect plant diseases from images for accurate and timely diagnosis.

Dataset: Using the Kaggle "New Plant Disease Detection Dataset," we trained our model on 25 classes of diseases across 5 different crops.

Technology Stack: The system is built with Convolutional Neural Networks (CNN) using TensorFlow, Keras, and OpenCV.

Motivation: Early and accurate disease detection is essential to reducing crop losses and enhancing agricultural productivity, meeting a critical need in sustainable farming.

OBJECTIVES

1 **ML Model for Disease Detection**

CNN to classify 25 diseases across 5 crops, helping farmers manage crop health.

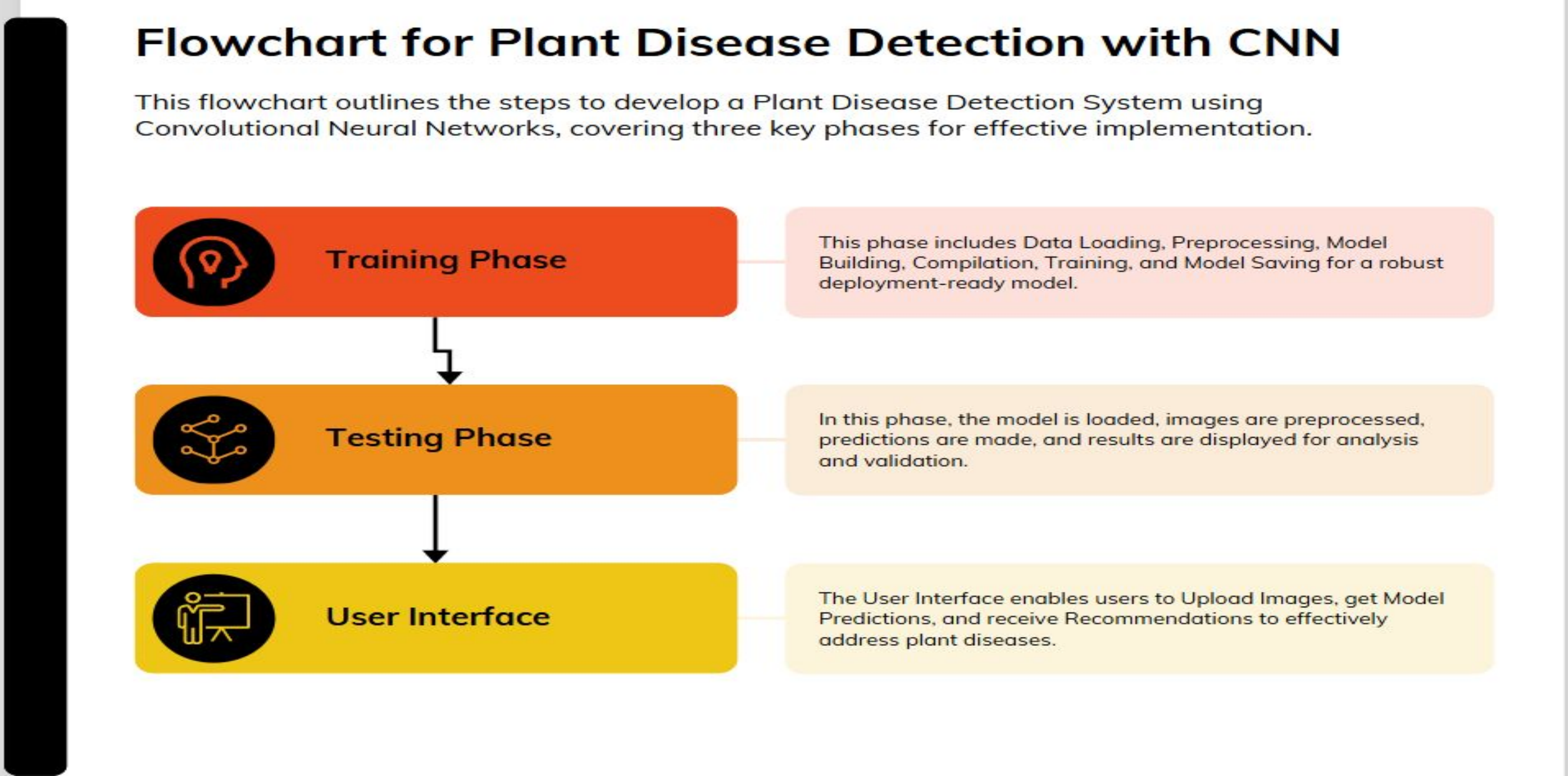
2 **Efficient Image Processing**

Techniques like resizing and normalization for accurate disease detection.

3 **Real-Time Prediction Interface**

Web tool for instant disease diagnosis, confidence scores, and management tips.

FLOWCHART



Flowchart for Plant Disease Detection with CNN

This flowchart outlines the steps to develop a Plant Disease Detection System using Convolutional Neural Networks, covering three key phases for effective implementation.

Training Phase

This phase includes Data Loading, Preprocessing, Model Building, Compilation, Training, and Model Saving for a robust deployment-ready model.

Testing Phase

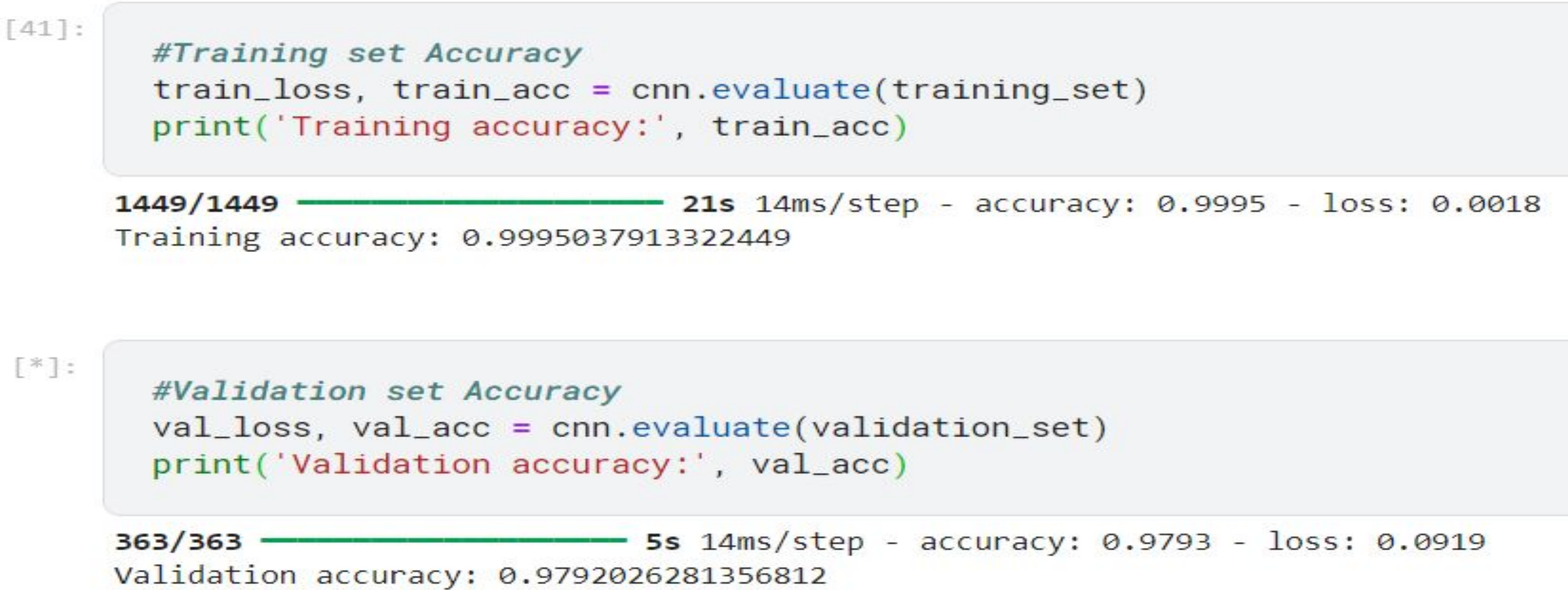
In this phase, the model is loaded, images are preprocessed, predictions are made, and results are displayed for analysis and validation.

User Interface

The User Interface enables users to Upload Images, get Model Predictions, and receive Recommendations to effectively address plant diseases.

EVALUATION METRICS

Model Performance: Our model achieved an impressive **99.9% Training accuracy** and **97.9% Validation accuracy**, highlighting its precision in identifying plant diseases across various crops.



```
[41]:
#Training set Accuracy
train_loss, train_acc = cnn.evaluate(training_set)
print('Training accuracy:', train_acc)
```

```
1449/1449 ━━━━━━━━━━━━━━━━━━━━ 21s 14ms/step - accuracy: 0.9995 - loss: 0.0018
Training accuracy: 0.9995037913322449
```

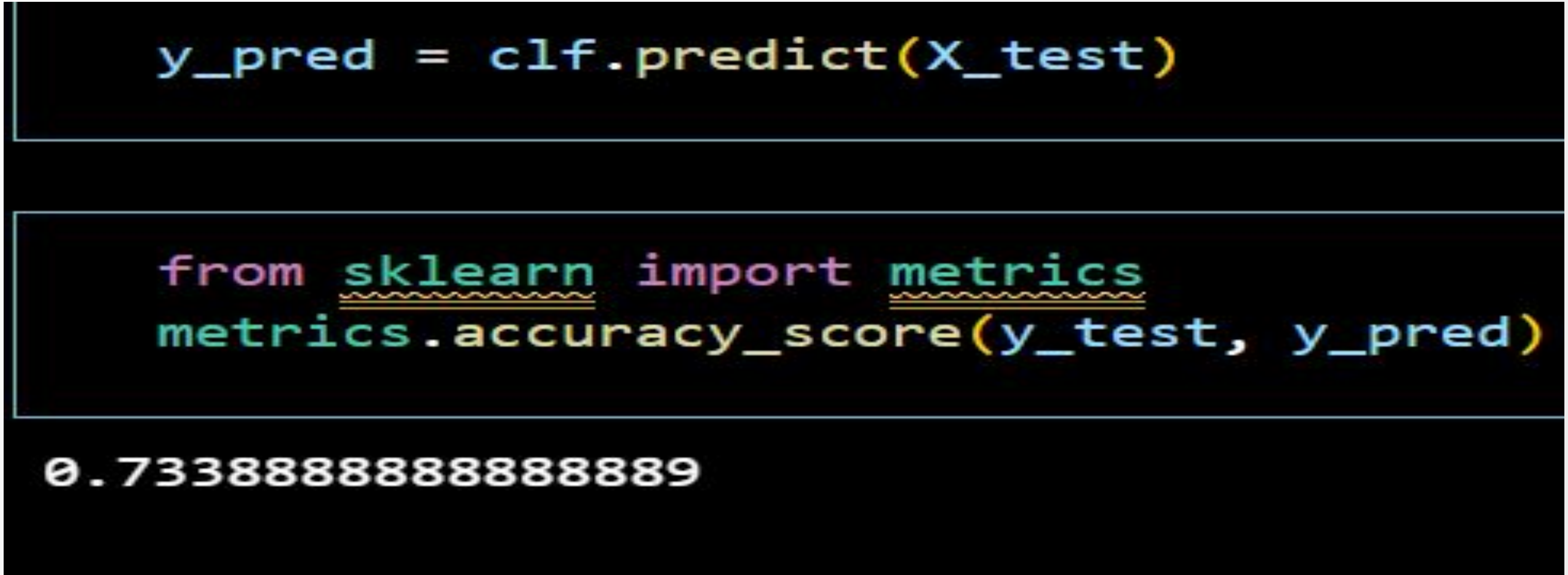
```
[*]:
#Validation set Accuracy
val_loss, val_acc = cnn.evaluate(validation_set)
print('Validation accuracy:', val_acc)
```

```
363/363 ━━━━━━━━━━━━━━━━━━━━ 5s 14ms/step - accuracy: 0.9793 - loss: 0.0919
Validation accuracy: 0.9792026281356812
```

WHY CNN?

Superior Accuracy: The CNN model achieved **99.9% training accuracy** and **97.9% validation accuracy**, significantly surpassing SVM's **73.3% accuracy**.

Automated Feature Extraction: CNNs automatically learn complex features from images, making them more effective for detecting plant diseases compared to SVM's reliance on manually defined features



```
y_pred = clf.predict(X_test)
```

```
from sklearn import metrics
metrics.accuracy_score(y_test, y_pred)
```

```
0.7338888888888889
```

Training Pipeline

1. **Data Loading**: Load and preprocess images from the "train" directory using `image_dataset_from_directory`, which automatically labels and batches the images.
2. **Model Building**: Initialize a CNN model using Keras, adding multiple convolutional, activation, and pooling layers.
3. **Model Compilation**: Compile the model, specifying the loss function, optimizer, and evaluation metrics.
4. **Model Training**: Train the model on the training dataset with validation on the validation dataset to monitor performance.
5. **Model Saving**: Save the trained model for future inference.

Testing Pipeline

1. **Model Loading**: Load the trained CNN model.
2. **Image Preprocessing**: Load and preprocess individual test images.
3. **Prediction**: Pass the test images through the model to get class predictions.
4. **Result Display**: Display the test image and the prediction results.

Streamlit Integration

1. **User Interface (UI)**: Use Streamlit to create an interactive front end where users can upload images and receive disease predictions with management recommendations.

Thank You