Plant Disease Classification using CNN

Presented By

- Shehab Haj Yahia
- Rafat Balaum

AI for Plant Disease Classification

This presentation introduces an innovative deep learning model designed for the rapid and accurate identification of plant diseases. Our initiative directly addresses critical challenges in food security and economic stability within the agricultural sector.

By leveraging advanced artificial intelligence, we aim to significantly reduce global crop losses, potentially by up to 40%, safeguarding livelihoods and ensuring food availability for communities worldwide.

Objective

- Build a deep learning model to detect diseases from leaf images
- Classification of plant leaf diseases using texture features

- Leveraging CNNs

The technology leverages Convolutional Neural Networks (CNNs) to precisely identify complex disease patterns from images, detecting subtle signs often missed by the human eye.

- Improve early intervention in agriculture

Methodology

- **Dataset**: kaggle-plantdisease or custom leaf image dataset
- **Preprocessing**: Image resizing, normalization, data augmentation
- **Model**: Convolutional Neural Network (CNN) architecture
- **Training**: With TensorFlow/Keras

Tools & Technologies

- **Language**: Python
- **Framework**: TensorFlow with Keras API
- **Libraries**: NumPy, Matplotlib, scikit-learn
- **Environment**: Jupyter Notebook / Google Colab

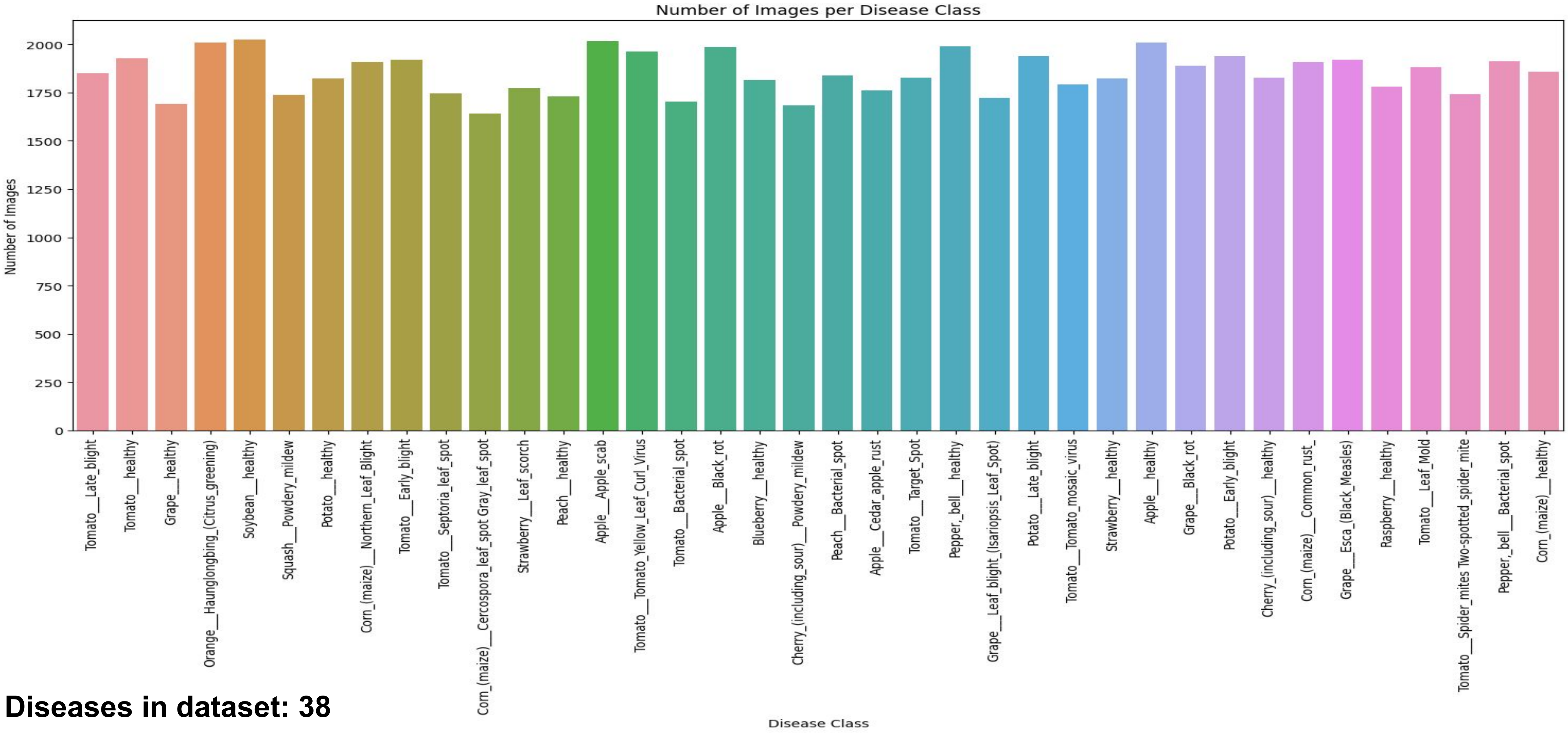
Dataset Overview

- Source: New Plant Diseases Dataset
- Number of Classes: 38
- Image Dimensions: 224x244x3 pixels
- Train/Validation Split: 80/20
  - `Training Images: 56251 images`
  - `Validating Images: 14044 images`
  - `Test Images: 17572 images`

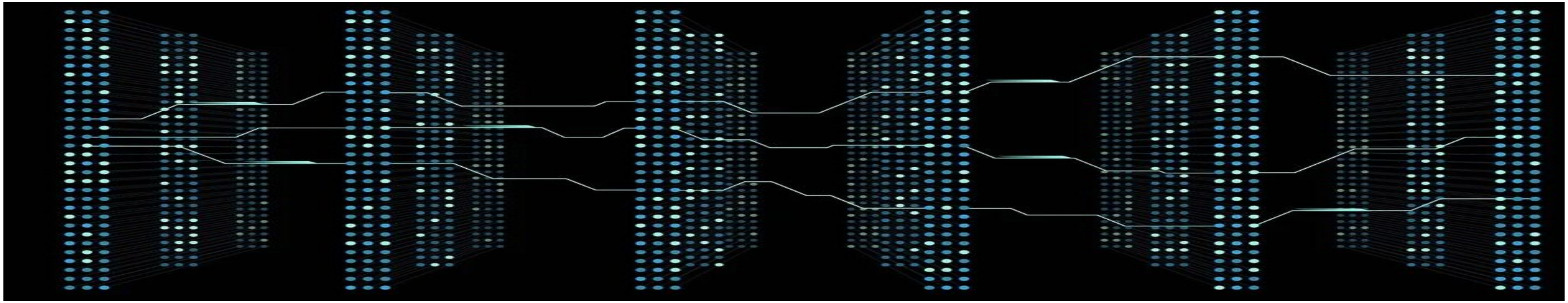
- Preprocessing: Normalization and One-Hot Encoding

Data understanding & Visualization

Number of different plants is:14



Diseases in dataset: 38



Our Model: Architecture and Training Data

Model Architecture

We have built and trained from scratch CNN model

In addition, we have selected **MobileNetV2**, a highly efficient Convolutional Neural Network (CNN) architecture. This choice is optimized for efficient inference, making it ideal for deployment on *edge devices like smartphones or drones* in agricultural settings.

Dataset & Training

Our model was trained on the extensive **PlantVillage dataset**, comprising over 56,251 images across 14 crop species and 38 distinct disease classes. We utilized a standard 80/20 split for training, validation, and 17,572 for testing.

CNN Architecture

- Input Layer: Image tensors (224x224x3)
- Conv2D Layer + ReLU
- MaxPooling Layer
- Dropout Layer
- Flatten Layer
- Dense Layer with ReLU
- Output Dense Layer with Softmax (38 classifications)

Model Summary

- Total parameters: 744,966
- Trainable parameters: 743,046
- Total number of layers(`model.layers`):23
- Model depth: 11

(3 * 3*3 +1) * 32 = 896

Feature Extraction

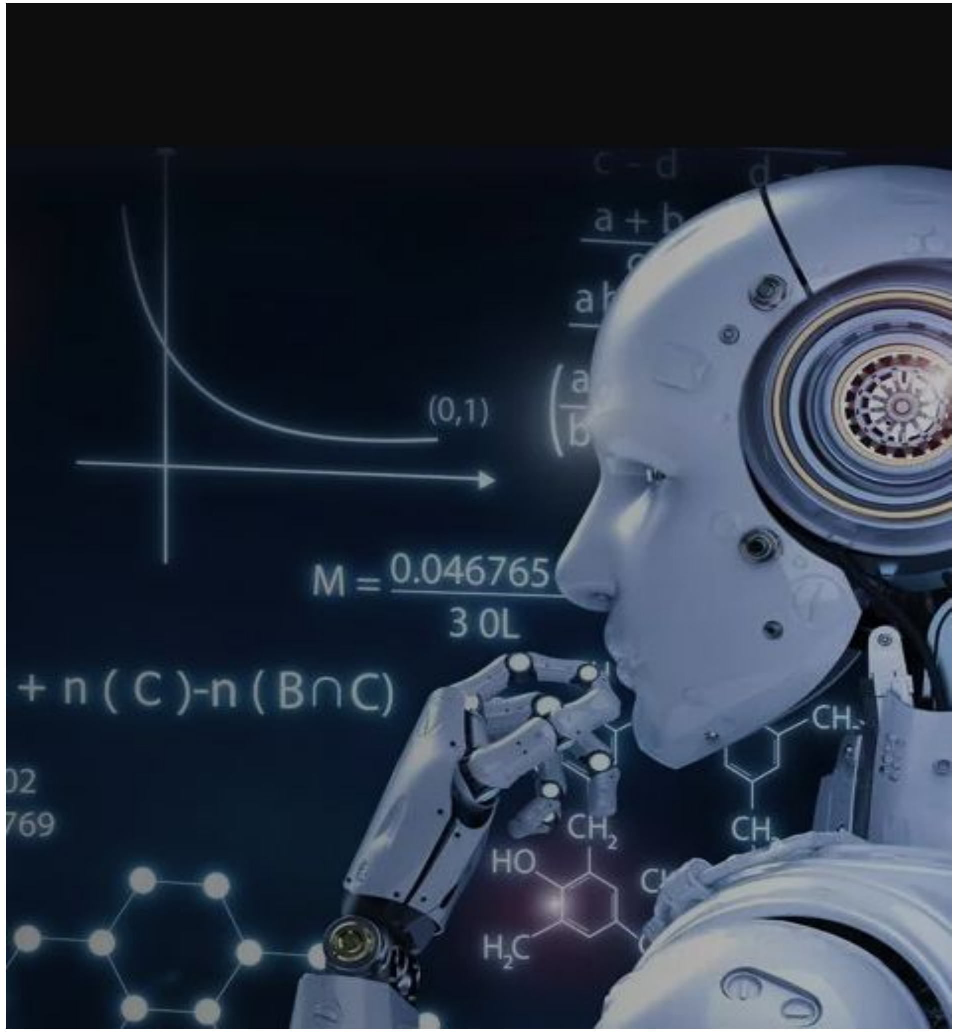
Flatten Layer

Dense Layer with ReLU

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 222, 222, 32)	896
conv2d_1 (Conv2D)	(None, 220, 220, 32)	9,248
max_pooling2d (MaxPooling2D)	(None, 110, 110, 32)	0
conv2d_2 (Conv2D)	(None, 108, 108, 64)	18,496
conv2d_3 (Conv2D)	(None, 106, 106, 64)	36,928
max_pooling2d_1 (MaxPooling2D)	(None, 53, 53, 64)	0
conv2d_4 (Conv2D)	(None, 51, 51, 128)	73,856
conv2d_5 (Conv2D)	(None, 49, 49, 256)	295,168
max_pooling2d_2 (MaxPooling2D)	(None, 24, 24, 256)	0
global_average_pooling2d (GlobalAveragePooling2D)	(None, 256)	0
dense (Dense)	(None, 512)	131,584
batch_normalization (BatchNormalization)	(None, 512)	2,048
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 256)	131,328
batch_normalization_1 (BatchNormalization)	(None, 256)	1,024
dropout_1 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 128)	32,896
batch_normalization_2 (BatchNormalization)	(None, 128)	512
dropout_2 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 64)	8,256
batch_normalization_3 (BatchNormalization)	(None, 64)	256
dropout_3 (Dropout)	(None, 64)	0
dense_4 (Dense)	(None, 38)	2,470

Training Configuration

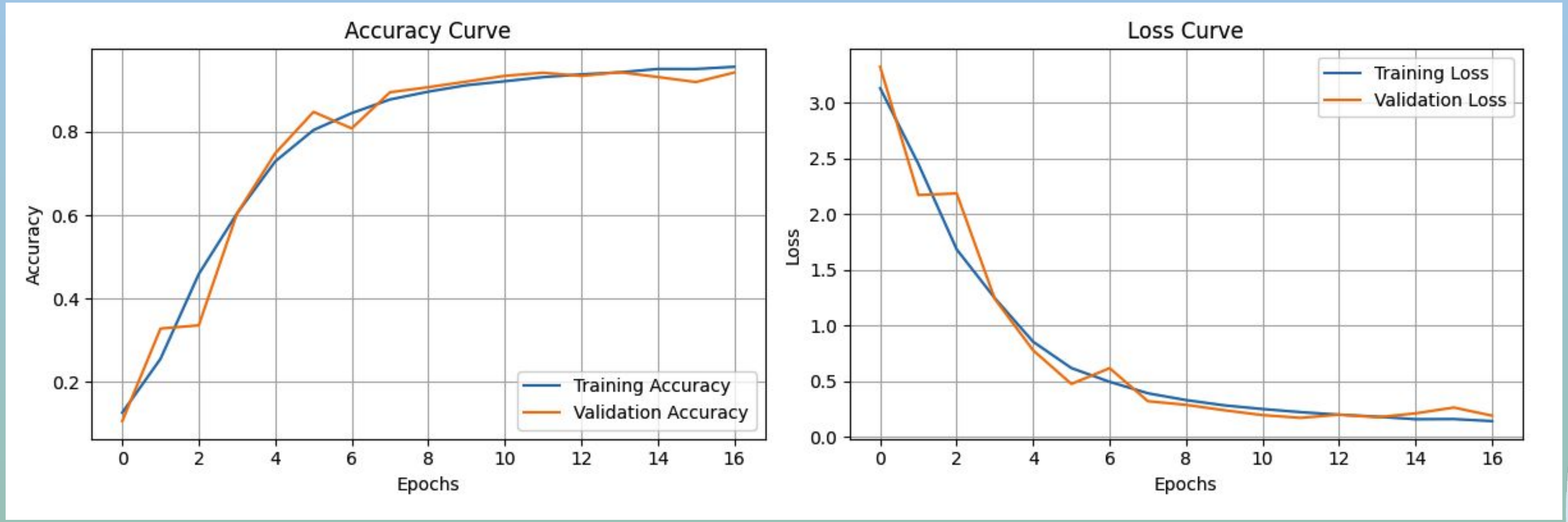
- Loss Function: Categorical Crossentropy
- Optimizer: Adam
- Metrics: Accuracy
- Epochs: 25
- Batch Size: 64
- Validation Split: 0.2
- Callbacks
  - ModelCheckpoint (*val_accuracy*)
  - EarlyStopping (*val_accuracy*)




```
# train the model
model_checkpoint = ModelCheckpoint('/content/working/cnn_model.keras', monitor='val_accuracy', save_best_only=True, mode='max', verbose=1)
early_stopping = EarlyStopping(monitor='val_accuracy', patience=3, verbose=1, mode='max', restore_best_weights=True)

history = model.fit(train_data,
                    validation_data=valid_data,
                    epochs=epochs,
                    batch_size=batch_size,
                    callbacks=[model_checkpoint, early_stopping])
```

Performance & Real-World Impact



95.71%

Classification Accuracy

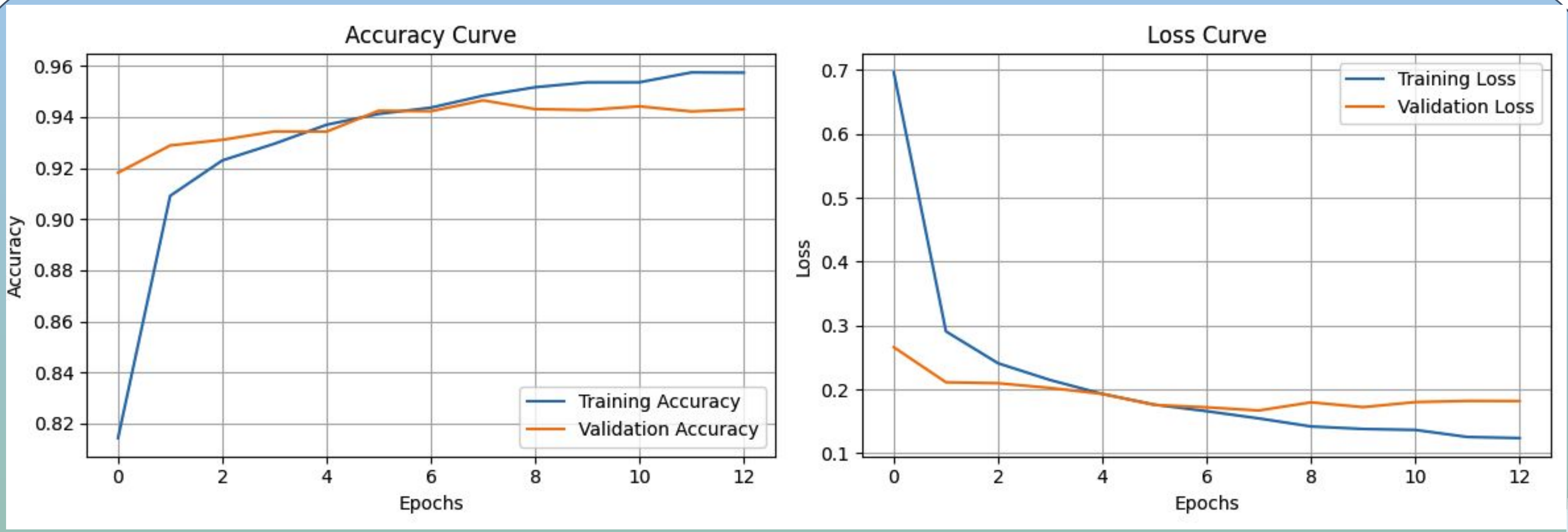
25ms

Average processing time per image on a standard GPU, allowing for real-time analysis and rapid decision-making in the field.

Model Training Verbose

```
Epoch 4: val_accuracy improved from 0.93107 to 0.93435, saving model to /content/working/cnn_model.keras
879/879 ━━━━━━━━━━━━━━━━━━━━ 156s 178ms/step - accuracy: 0.9333 - loss: 0.2094 - precision: 0.9483 - recall: 0.9179 - val_accuracy: 0.9343 - val_loss: 0.2024 -
Epoch 5/25
879/879 ━━━━━━━━━━━━━━━━━━━━ 0s 143ms/step - accuracy: 0.9371 - loss: 0.1900 - precision: 0.9521 - recall: 0.9251
Epoch 5: val_accuracy did not improve from 0.93435
879/879 ━━━━━━━━━━━━━━━━━━━━ 156s 177ms/step - accuracy: 0.9371 - loss: 0.1900 - precision: 0.9521 - recall: 0.9251 - val_accuracy: 0.9343 - val_loss: 0.1930 -
Epoch 6/25
879/879 ━━━━━━━━━━━━━━━━━━━━ 0s 143ms/step - accuracy: 0.9424 - loss: 0.1736 - precision: 0.9545 - recall: 0.9321
Epoch 6: val_accuracy improved from 0.93435 to 0.94247, saving model to /content/working/cnn_model.keras
879/879 ━━━━━━━━━━━━━━━━━━━━ 167s 190ms/step - accuracy: 0.9424 - loss: 0.1736 - precision: 0.9545 - recall: 0.9321 - val_accuracy: 0.9425 - val_loss: 0.1757 -
Epoch 7/25
879/879 ━━━━━━━━━━━━━━━━━━━━ 0s 144ms/step - accuracy: 0.9455 - loss: 0.1602 - precision: 0.9565 - recall: 0.9367
Epoch 7: val_accuracy did not improve from 0.94247
879/879 ━━━━━━━━━━━━━━━━━━━━ 157s 179ms/step - accuracy: 0.9455 - loss: 0.1602 - precision: 0.9565 - recall: 0.9367 - val_accuracy: 0.9423 - val_loss: 0.1720 -
Epoch 8/25
879/879 ━━━━━━━━━━━━━━━━━━━━ 0s 141ms/step - accuracy: 0.9489 - loss: 0.1502 - precision: 0.9591 - recall: 0.9402
Epoch 8: val_accuracy improved from 0.94247 to 0.94653, saving model to /content/working/cnn_model.keras
879/879 ━━━━━━━━━━━━━━━━━━━━ 155s 176ms/step - accuracy: 0.9489 - loss: 0.1502 - precision: 0.9591 - recall: 0.9402 - val_accuracy: 0.9465 - val_loss: 0.1671 -
Epoch 9/25
879/879 ━━━━━━━━━━━━━━━━━━━━ 0s 141ms/step - accuracy: 0.9524 - loss: 0.1380 - precision: 0.9606 - recall: 0.9449
Epoch 9: val_accuracy did not improve from 0.94653
879/879 ━━━━━━━━━━━━━━━━━━━━ 154s 175ms/step - accuracy: 0.9524 - loss: 0.1380 - precision: 0.9606 - recall: 0.9449 - val_accuracy: 0.9431 - val_loss: 0.1798 -
Epoch 10/25
879/879 ━━━━━━━━━━━━━━━━━━━━ 0s 140ms/step - accuracy: 0.9561 - loss: 0.1307 - precision: 0.9639 - recall: 0.9486
Epoch 10: val_accuracy did not improve from 0.94653
879/879 ━━━━━━━━━━━━━━━━━━━━ 153s 174ms/step - accuracy: 0.9561 - loss: 0.1307 - precision: 0.9639 - recall: 0.9486 - val_accuracy: 0.9428 - val_loss: 0.1723 -
Epoch 11/25
879/879 ━━━━━━━━━━━━━━━━━━━━ 0s 142ms/step - accuracy: 0.9540 - loss: 0.1322 - precision: 0.9625 - recall: 0.9467
Epoch 11: val_accuracy did not improve from 0.94653
879/879 ━━━━━━━━━━━━━━━━━━━━ 154s 176ms/step - accuracy: 0.9540 - loss: 0.1322 - precision: 0.9625 - recall: 0.9467 - val_accuracy: 0.9442 - val_loss: 0.1802 -
Epoch 12/25
879/879 ━━━━━━━━━━━━━━━━━━━━ 0s 144ms/step - accuracy: 0.9576 - loss: 0.1226 - precision: 0.9647 - recall: 0.9518
Epoch 12: val_accuracy did not improve from 0.94653
879/879 ━━━━━━━━━━━━━━━━━━━━ 157s 178ms/step - accuracy: 0.9576 - loss: 0.1226 - precision: 0.9647 - recall: 0.9518 - val_accuracy: 0.9422 - val_loss: 0.1821 -
Epoch 13/25
879/879 ━━━━━━━━━━━━━━━━━━━━ 0s 143ms/step - accuracy: 0.9598 - loss: 0.1192 - precision: 0.9660 - recall: 0.9537
Epoch 13: val_accuracy did not improve from 0.94653
879/879 ━━━━━━━━━━━━━━━━━━━━ 155s 176ms/step - accuracy: 0.9598 - loss: 0.1192 - precision: 0.9660 - recall: 0.9537 - val_accuracy: 0.9430 - val_loss: 0.1819 -
Epoch 13: early stopping
Restoring model weights from the end of the best epoch: 8.
```

Performance & Real-World Impact



96%

Classification Accuracy

25ms

Average processing time per image on a standard GPU, allowing for real-time analysis and rapid decision-making in the field.

Inference Speed

Random Sample Predictions

True: AppleCedarRust3
Predicted: Apple___Cedar_apple_rust

True: TomatoYellowCurlVirus5
Predicted: Tomato___Tomato_Yellow_Leaf_Curl_Virus

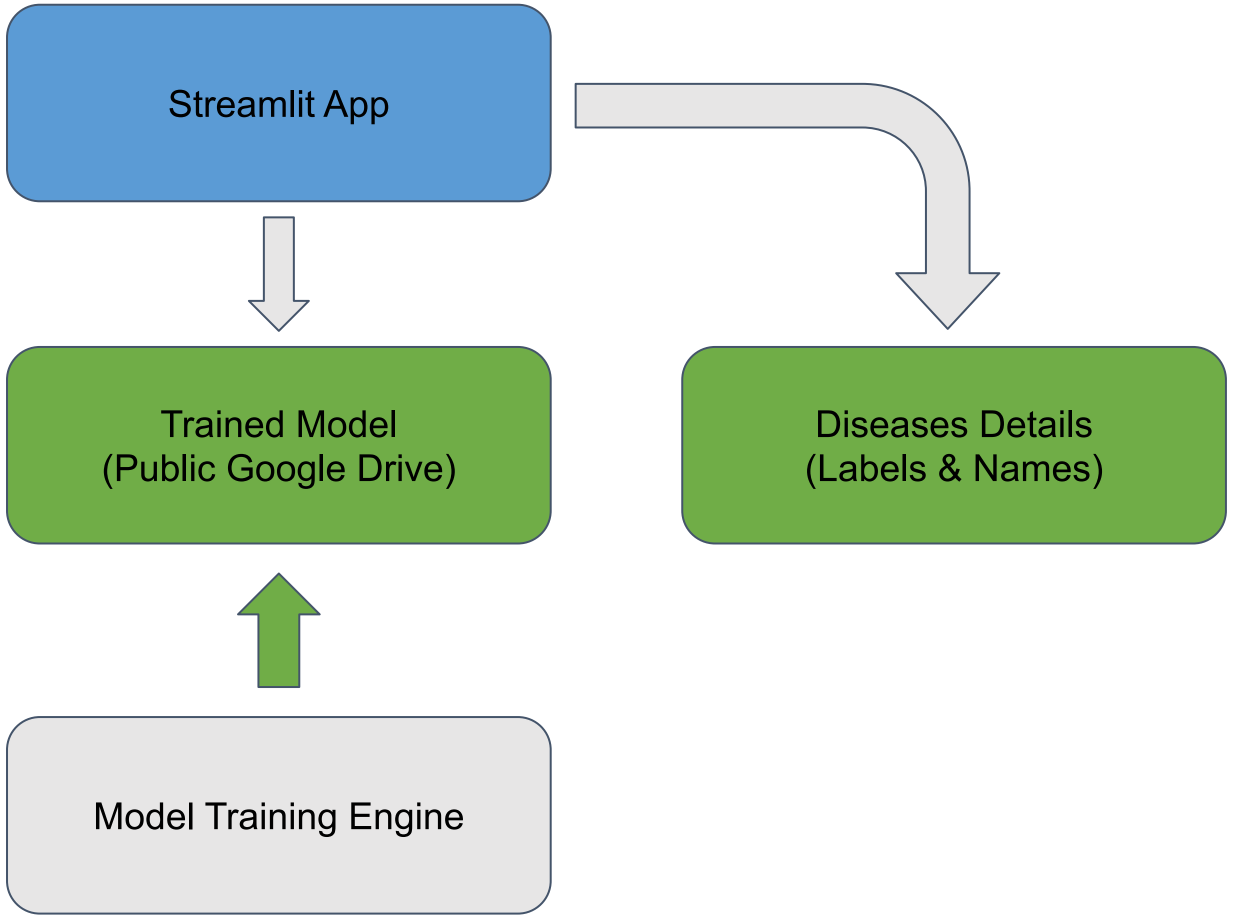
True: TomatoYellowCurlVirus2
Predicted: Tomato___Tomato_Yellow_Leaf_Curl_Virus

Challenges Faced

- Selecting the right architecture
- Hyperparameter tuning
- Limited dataset size
- Overfitting on training data
- GPU is not always available, therefore, it took a long time to train model

Plant Disease Classification App

High Level Design



Applications & Future Potential

Mobile Applications

Enables on-the-spot disease diagnosis for smallholder farmers using just a smartphone camera, providing immediate actionable insights.

Precision Agriculture

Integrates with drone-based monitoring systems for large farms, allowing the identification of localized outbreaks over vast areas, optimizing resource allocation.

Automated Farming

Seamless integration with robotic systems for targeted treatment and localized intervention, automating the response to disease detection.

New Disease Discovery

The model adapts to novel pathogens through continuous learning, ensuring its relevance and effectiveness in identifying emerging plant diseases.

Conclusion: Cultivating a Smarter, More Resilient Future

Our deep learning model for plant disease classification represents a significant leap forward for agriculture. It offers a scalable, accurate, and cost-effective solution to a critical global challenge. By empowering farmers with timely and precise disease identification, we can dramatically reduce economic losses and enhance global food security, fostering a more sustainable and intelligent agricultural future.

Learn More