Tomato Plant Disease Detection

Title

Tomato Plant Disease Detection using TinyML: Edge-Based Disease Detection with Optimized CNNs.

Indroduction

This project aims to develop a edge-deployable machine learning model capable of accurately identifying tomato plant diseases from leaf images. By leveraging TensorFlow Lite and Edge Impulse, we'll create a **compact and optimized** model that can run directly on resource-constrained devices, enabling real-time, on-site disease diagnosis for enhanced agricultural decision-making.

Dataset

Plant Village dataset containing **16011** tomato leaf images **10** disease categories: Bacterial spot, Early blight, Healthy, Late blight, Leaf Mold, Septoria leaf spot, Target Spot, Tomato mosaic virus, Tomato yellow leaf curl virus, Two-spotted spider mite.

Methodology

1. Data Preprocessing and Exploration

A. Loading and Visualization: Load the Plant Village dataset containing **16011** tomato leaf images. Visualize sample images to understand data distribution and potential issues.

B. Image Augmentation: Apply techniques like rotation, flipping, brightness adjustments, and cropping to increase dataset diversity and reduce overfitting.

C. Data Splitting: Divide the dataset into training (**80%**), validation (**10%**), and testing (**10%**) sets using a suitable splitting strategy.

2. Model Development

A. Architecture Design: Choose a Convolutional Neural Network (CNN) architecture considering both accuracy and efficiency for edge deployment.

B. Implementation: Build the model using TensorFlow, defining layers, activation functions, and output layers.

C. Training: Train the model on the training set using optimizers (e.g., Adam), loss functions (e.g., categorical cross-entropy), and batch sizes. Monitor training progress using metrics like accuracy and loss, and visualize them.

D. Hyperparameter Tuning: Experiment with different hyperparameters (e.g., learning rate, batch size, number of epochs) to find the best configuration for model performance.

3. Model Conversion and Deployment

A. Quantization: To make the model smaller and more efficient for deployment on resource-constrained devices, we'll apply techniques like quantization.

B. Model Pruning: We can further reduce the model size by removing redundant connections or neurons that have minimal impact on prediction.

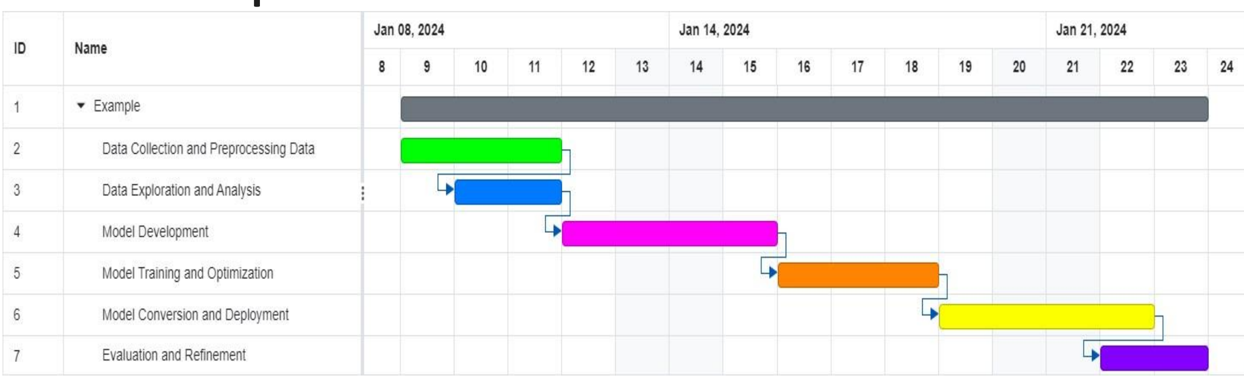
C. Packaging and Deployment: Finally, we'll use Edge Impulse to package the optimized TensorFlow Lite model and deploy it to your chosen edge device.

4. Evaluation and Validation

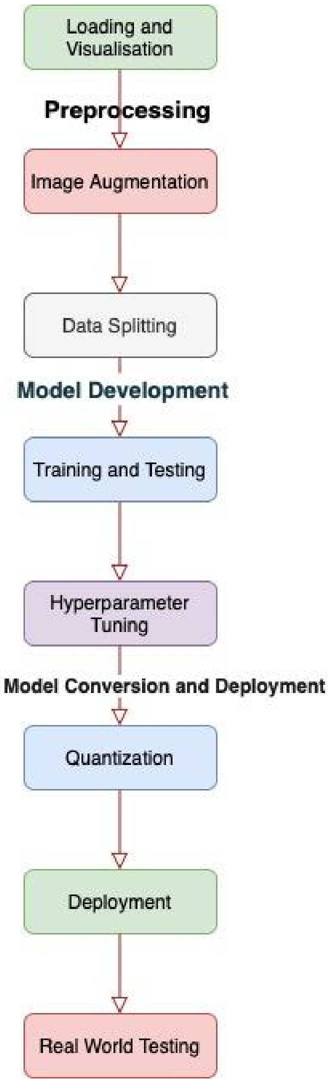
A. Evaluation on Validation and Testing Sets: Evaluate the model's performance on the validation and testing sets using metrics like accuracy.

B. Real-World Testing: deploy the model on a physical device to assess its accuracy and resource usage in a practical setting.

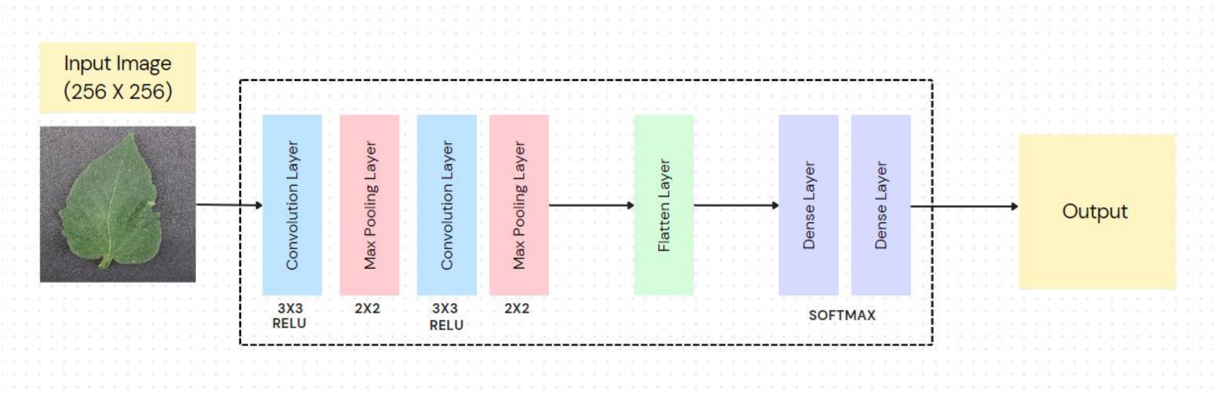
Timeline Map



Flow Chart



Loading and Visualisation
Preprocessing
Image Augmentation
Data Splitting
Model Development
Training and Testing
Hyperparameter Tuning
Model Conversion and Deployment
Quantization
Deployment
Real World Testing



Input Image
(256 X 256)
Convolution Layer
3X3
RELU
Max Pooling Layer
2X2
Convolution Layer
3X3
RELU
Max Pooling Layer
2X2
Flatten Layer
Dense Layer
Dense Layer
SOFTMAX
Output


Actual: Tomato_Early_blight,
Predicted: Tomato_Early_blight.
Confidence: 94.53125%
Actual: Tomato__Target_Spot,
Predicted: Tomato__Target_Spot.
Confidence: 93.75%
Actual: Tomato__Target_Spot,
Predicted: Tomato__Target_Spot.
Confidence: 87.890625%
Actual: Tomato__Tomato_YellowLeaf__Curl_Virus,
Predicted: Tomato__Tomato_YellowLeaf__Curl_Virus.
Confidence: 98.828125%
Actual: Tomato_Leaf_Mold,
Predicted: Tomato_Septoria_leaf_spot.
Confidence: 64.0625%
Actual: Tomato__Tomato_YellowLeaf__Curl_Virus,
Predicted: Tomato__Tomato_YellowLeaf__Curl_Virus.
Confidence: 99.609375%
Actual: Tomato_healthy,
Predicted: Tomato_healthy.
Confidence: 57.421875%
Actual: Tomato_Early_blight,
Predicted: Tomato_Early_blight.
Confidence: 99.21875%
Actual: Tomato__Tomato_YellowLeaf__Curl_Virus,
Predicted: Tomato__Tomato_YellowLeaf__Curl_Virus.
Confidence: 99.609375%

Edge Impulse Deployment

Step 2: Process "tf_lite_quantized_model.tflite"

Configure model settings for optimal processing.

Model input
Input shape: (128, 128, 3)
Image (RGB)

How is your input scaled?
Pixels ranging 0..255 (not normalized)
Input should be in RGB format (one value per pixel). If your model uses a different channel order, or is scaled differently, then select "Other".

Model output
Output shape: (10)
Classification

Output labels (10)
Enter labels for your model separated by ','.
Tomato_Bacterial_spot, Tomato_Early_blight

Save model

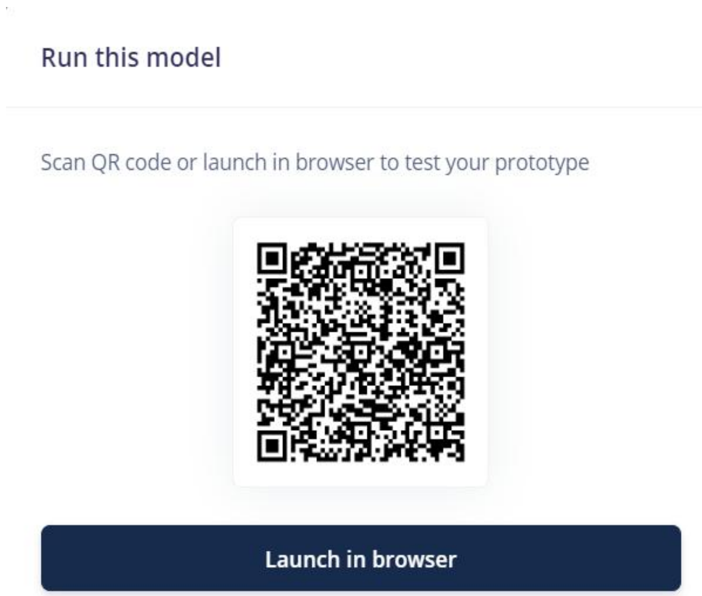
On-device performance

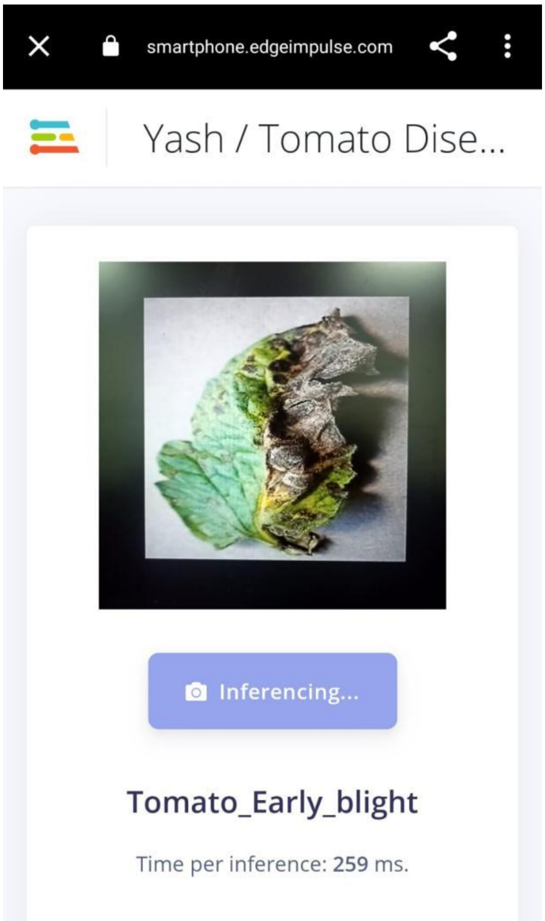
MCUs

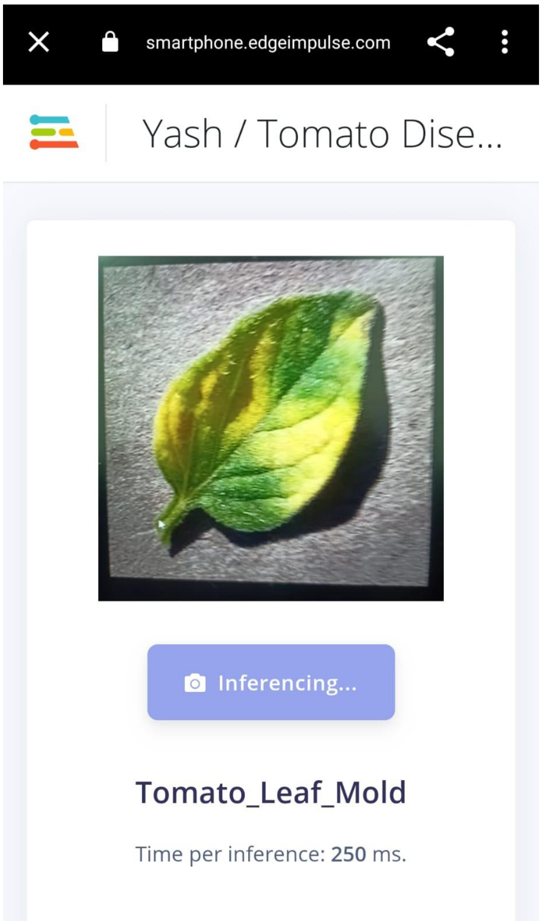
		EON COMPILER		TFLITE	
DEVICE	LATENCY	RAM	ROM	RAM	ROM
Low-end MCU	19,458,911 r	-0.0K	3.6M	2.9M +498.5K	3.6M +18.9K
High-end MCU	264,752 ms.	-0.0K	3.6M	2.9M +497.5K	3.6M +21.1K
+ AI accelerator	264,752 ms.	-0.0K	3.6M	2.9M +497.5K	3.6M +21.1K

Microprocessors

DEVICE	LATENCY	MODEL SIZE
MPU	4,557 ms.	3.5M
GPU or accelerator	760 ms.	3.5M







Results

We have able to achieve an accuracy of **89.6%**.

Expected Outcomes

Development of a high-performing, lightweight ML model for tomato plant disease identification Successful deployment of the model on an edge device using Edge Impulse Demonstration of the model's effectiveness in real-world settings Potential contributions to agricultural decision-making and disease management practices.

Real-World Use

Real-Time Plant Health Prediction on Your Phone:

Launch the Edge Impulse app on your phone. Point your phone's camera at a tomato leaf. The model will process the image in real-time, analyzing it for disease signatures based on the trained classification categories. You'll receive an instant prediction on your phone screen, indicating the identified disease or confirming a healthy leaf.