***Edge AI Prototype Development Report***

**Introduction**

This project focused on developing an **Edge AI image classification prototype** using TensorFlow Lite. The goal was to train a lightweight convolutional neural network (CNN) capable of classifying plant leaf diseases from images, convert the model to a TensorFlow Lite format, and demonstrate its potential deployment on resource-constrained devices such as Raspberry Pi or mobile phones. Edge AI solutions like this are essential for enabling real-time, privacy-preserving, and offline-capable AI applications.

**Dataset and Preprocessing**

The prototype used the Beans dataset from TensorFlow Datasets, consisting of images categorized into three classes related to healthy and diseased bean leaves. To suit Edge AI constraints, images were resized to 128x128 pixels and normalized to pixel values between 0 and 1. This preprocessing step ensures efficient model training and inference while reducing computational requirements.

**Model Architecture and Training**

A compact CNN architecture was designed to balance accuracy with efficiency. The model consisted of two convolutional layers with ReLU activations and max-pooling, followed by a global average pooling layer and a dense softmax output layer for classification into three categories. This relatively simple architecture is ideal for edge devices due to its low parameter count and computational overhead.

The model was trained for 5 epochs using the Adam optimizer and sparse categorical cross-entropy loss function. Training achieved a test accuracy of approximately 87%, demonstrating sufficient performance for a prototype intended for real-time applications.

**TensorFlow Lite Conversion**

To optimize the model for deployment on edge devices, it was converted to TensorFlow Lite (.tflite) format using TensorFlow’s built-in converter. This conversion reduces the model size and optimizes it for low-latency inference on devices with limited memory and processing power.

The .tflite model was successfully saved and made ready for deployment. Such models can run on devices like Raspberry Pi, Android/iOS smartphones, or microcontrollers, enabling local AI processing without reliance on cloud infrastructure.

**Benefits of Edge AI in Real-Time Applications**

Edge AI offers several critical advantages over traditional cloud-based AI:

**Reduced Latency:** By performing inference locally on the device, Edge AI minimizes delay, enabling instantaneous responses crucial for applications like autonomous drones or smart agricultural sensors.

**Enhanced Privacy:** Sensitive data (e.g., images of plants or medical data) stays on-device, mitigating privacy and security risks associated with transmitting data to remote servers.

**Offline Operation:** Edge AI enables devices to function reliably in remote or network-constrained environments, essential for field deployments such as rural farms or disaster zones.

**Deployment Considerations**

Deploying the .tflite model involves transferring the file to the target device and using a TensorFlow Lite interpreter to load and run inferences on input data from connected sensors or cameras. This workflow empowers real-time decision-making on the edge, such as alerting farmers about crop diseases or automating quality control in recycling plants.

**Conclusion**

This project successfully demonstrated how an Edge AI prototype could be developed using TensorFlow Lite and publicly available datasets. By balancing model complexity and accuracy, the solution showcases the feasibility of deploying AI models on resource-limited devices for real-time, privacy-aware applications. Edge AI represents a transformative shift toward decentralized AI, promising impactful benefits across industries like agriculture, healthcare, and environmental monitoring.