Title: Edge AI Prototype — Recyclable Item Classification (TF-Flowers)

Dataset: TF-Flowers (converted from 5 classes → binary: labels 0–2 = non-recyclable, 3–4 = recyclable). Clean dataset; no corrupted files.

Model: Lightweight CNN (Conv2D, DepthwiseConv2D, GlobalAveragePooling), ~ small parameter count for edge deployment.

Training:

- Train/Val/Test split: 80/10/10 (seeded)
- Image size: 128×128
- Batch size: 32
- Epochs: 8 (prototype)

Evaluation (example placeholders — run script for exact values):

- Test accuracy: reported by script (expect ~70–90% depending on run)
- Precision, recall, F1: see classification_report output
- Confusion matrix: see printed matrix

TFLite conversion:

- Converted with tf.lite.TFLiteConverter and converter.optimizations = [tf.lite.Optimize.DEFAULT].
- Saved model: tf_flowers_edge_model.tflite (size: printed by script).

Edge AI benefits demonstrated:

- Low latency: Inference on device (Pi) avoids round-trip to cloud.
- Privacy: Image data never leaves device.
- Offline operation: Model runs without internet.
- Cost saving: Avoids repeated cloud inference calls.

Limitations & next steps:

- Mapping TF-Flowers classes' - recyclable is synthetic; for a real recycling system collect domain-specific labeled images (paper, plastic, metal, glass, organic).
- Use data augmentation and/or transfer learning (MobileNetV2) for better accuracy.

- Use representative dataset for full integer quantization to improve speed and reduce model size on Pi.

Quick troubleshooting & tips

- If dataset download is slow in Colab, enable GPU runtime (Runtime → Change runtime type → GPU). But for this small model CPU is fine.
- If you want deterministic results, set seeds for TF and NumPy (included) but note absolute determinism across TF versions/hardware can still vary.
- For very small edge targets (microcontrollers), use TensorFlow Lite Micro and further quantize/convert via tflite_convert with representative dataset and target_spec.supported_ops = [tf.lite.OpsSet.TFLITE_BUILTINS_INT8].