Classifying Recyclable items using Machine Learning

Byung Woong Ko, Jay Siegfried

Problem Statement

- Trash disposal is critical component of longevity of human race
  - Most U.S. waste is sent to landfills or incinerated
  - Recycling helps reduce carbon emission and allows us to reduce net waste
  - Singapore, Korea have infrastructure where classifying items are in place
- US does not have infrastructure set to recycle
  - Many recyclable items are dumped into same bin
    - Each different items (plastic, paper, glass) needs to be processed differently from each other
  - It **cost** alot to implement these infrastructures
  - People need to "**join in**" to create these infrastructure → will they?
    - It's hard to make people change their norms in short time
- **We need a low-cost solution that works with the current system**

Solution: Let machine do it

- **ML applied IOT device that classifies recyclable materials**
  - **Cheaper:**
    - Korea: recycling compound per apartment → expensive
    - IOT device in an industrial classification pipeline is cheaper
  - **Does not involve changing behavior:**
  - People will not have to change how they recycle

Our proposal

- Recycle item classifier using transferred learning: MobileNetV2, EfficientNetB0 model
- Trained to classify 4 recyclable categories: Can, Glass, Paper, Plastic
- Train using combination of 2 datasets:
  - Trashnet dataset: https://www.kaggle.com/datasets/feyzazkefe/trashnet
  - Recycle_classification dataset: https://www.kaggle.com/datasets/jinfree/recycle-classification-dataset
- Final model deployed to Jetson Nano, a low-cost, low-power edge AI device
  - Pretrained model is loaded onto the device for real-time trash recognition
- Designed to work at point of disposal without requiring changes in user behavior

Experiment

Training and Optimization Process

- Employed **transfer learning** with MobileNetV2 and EfficientNetB0
- **Step 1 – Feature extraction**:
  - Transfer Model base **frozen**
  - Added custom classification head (GlobalAveragePooling, Dense, Dropout, Softmax)
  - Trained only the top layers initially

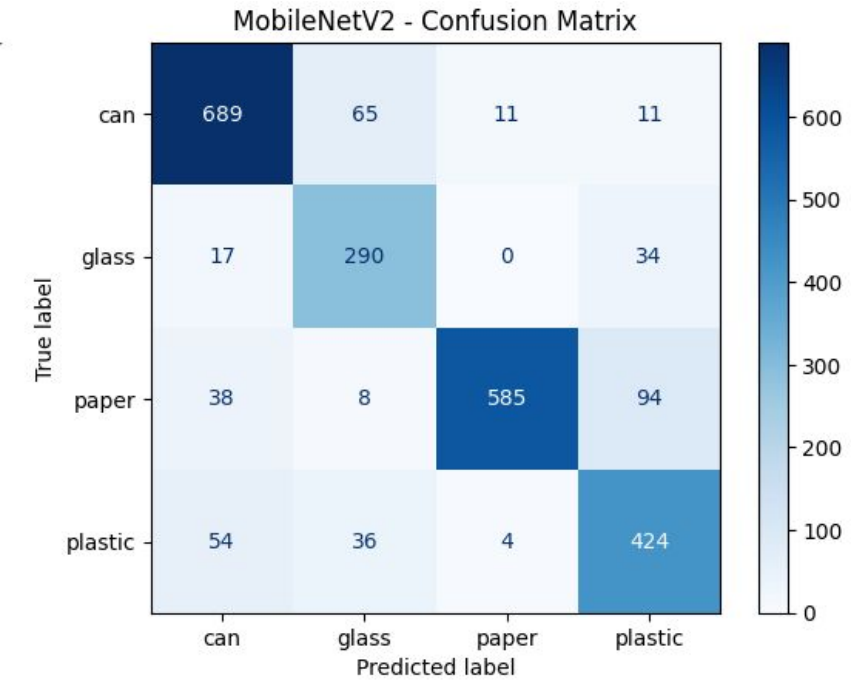
- **Step 2 – Fine-tuning**:
  - **Unfroze base model** for deeper adaptation to our trash datasets
  - Lowered learning rate to protect pretrained weights

- Applied **data augmentation**: horizontal flip, rotation, zoom, shifts

- Compared between the 2 models to get the best model

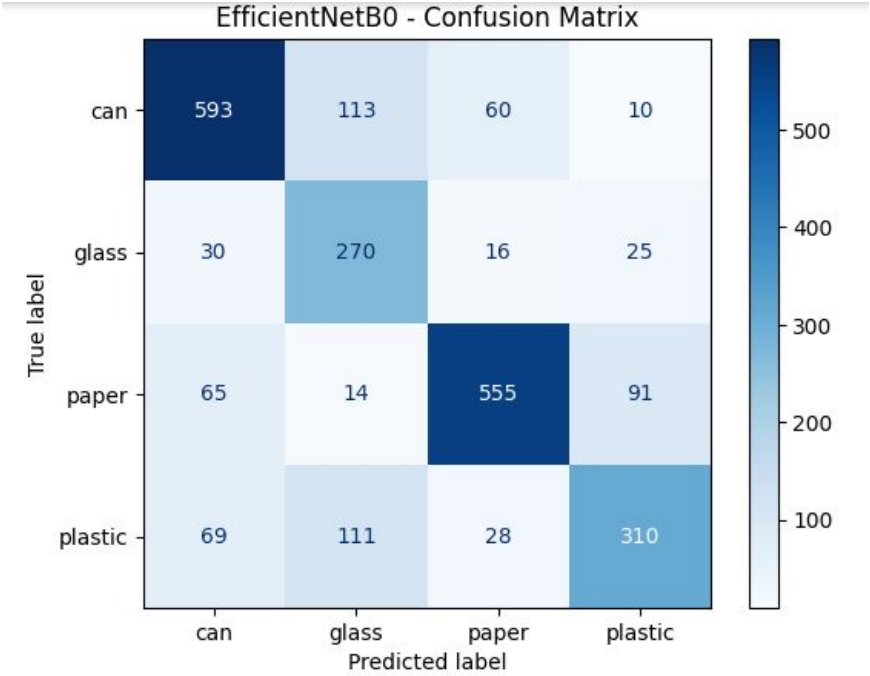
- Evaluated performance using **confusion matrix**, **per-class accuracy**, and **classification report**

Results: MobileNetV2



	precision	recall	f1-score	support
can	0.8634	0.8879	0.8755	776.0000
glass	0.7268	0.8504	0.7838	341.0000
paper	0.9750	0.8069	0.8830	725.0000
plastic	0.7531	0.8185	0.7845	518.0000
accuracy	0.8424	0.8424	0.8424	0.8424
macro avg	0.8296	0.8409	0.8317	2360.0000
weighted avg	0.8537	0.8424	0.8446	2360.0000

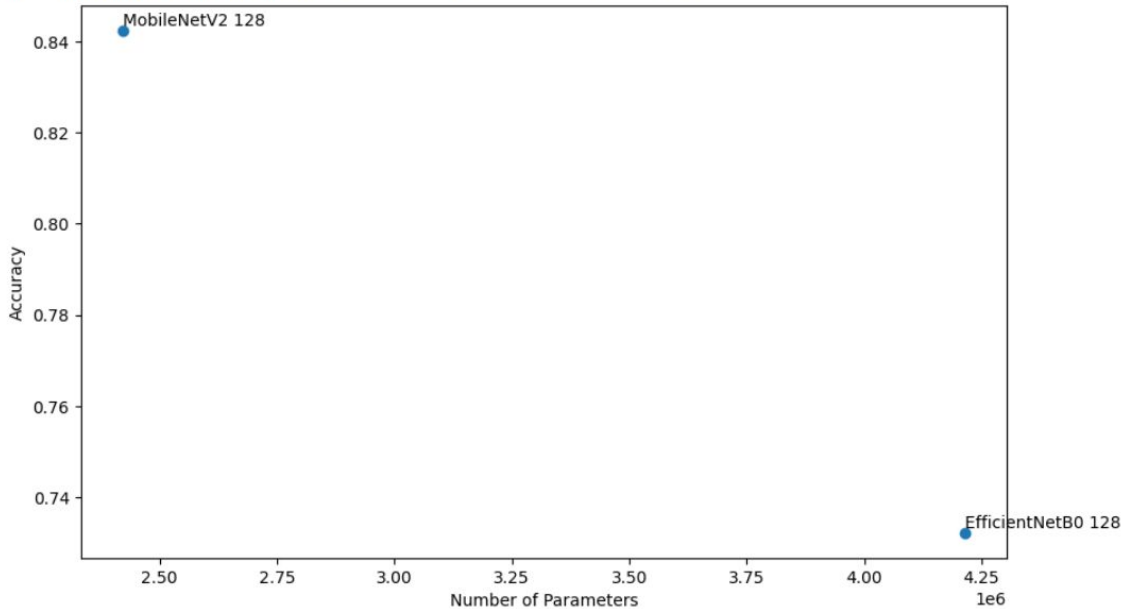
Results: EfficientNetB0



	precision	recall	f1-score	support
can	0.7834	0.7642	0.7736	776.0000
glass	0.5315	0.7918	0.6360	341.0000
paper	0.8422	0.7655	0.8020	725.0000
plastic	0.7110	0.5985	0.6499	518.0000
accuracy	0.7322	0.7322	0.7322	0.7322
macro avg	0.7170	0.7300	0.7154	2360.0000
weighted avg	0.7492	0.7322	0.7353	2360.0000

Results: Model Performance Comparison

```
Model Performance Summary:
name       description            input size  parameter count  accuracy
MobileNetV2 Multiclass TrashNet (FT)          128          2422468   0.84237
EfficientNetB0 Multiclass TrashNet (FT)          128          4214055   0.73220
```



- Clearly, MobileNetV2 is better

Deployment on Board

- Model is saved as a h5 file, but converted into onnx file (Open Neural Network Exchange)
- Both save a model and can be used, but onnx is more lightweight and is supported by more ML libraries
- ONNX Runtime optimizes performance and is supported by NVIDIA
- Transferred over USB

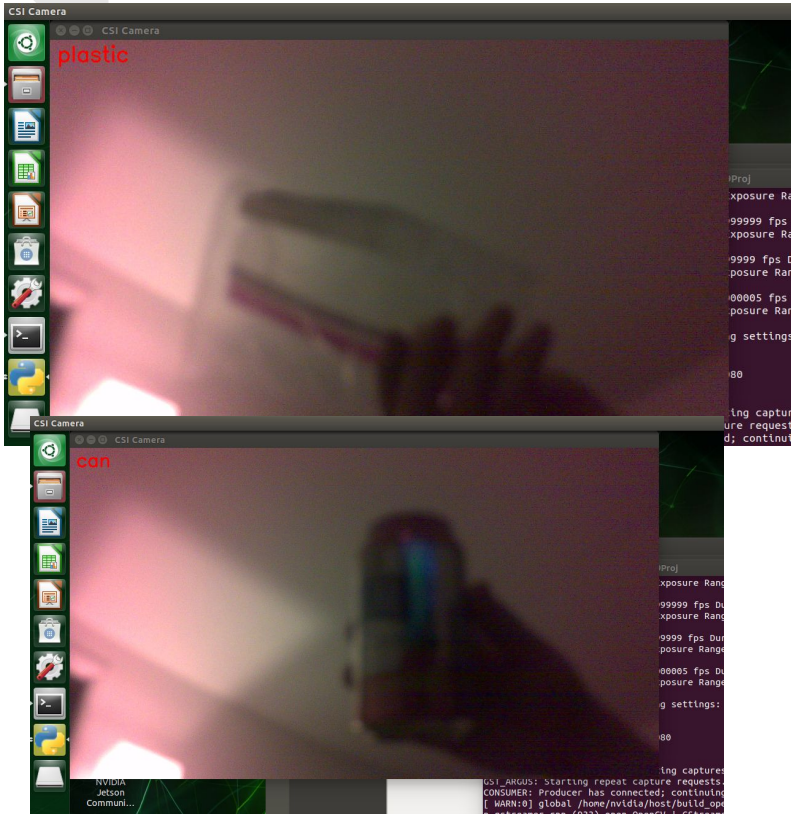
```
# Load the Keras model
model = './drive/MyDrive/MobileNetV2_trashnet.h5'
keras_model = load_model(model)

# Output Path
recyclenet = './drive/MyDrive/MobileNetV2_trashnet'
output_path = recyclenet + '.onnx'

# Input specifications
spec = (tf.TensorSpec((None, 128, 128, 3), tf.float32, name='input'),)

# Convert the keras h5 model to ONNX
model_proto, x = tf2onnx.convert.from_keras(keras_model, input_signature=spec, opset=13, output_path=output_path)

# Save the ONNX model
with open(output_path, "wb") as f:
    f.write(model_proto.SerializeToString())
```

Using the model

- Used “CSI-Camera” repository for camera interfacing
- Runs a live feed with the current prediction always displayed on the top left of the image
- Meant to simulate deployment at a recycle facility

```
ret_val, frame = video_capture.read()
# Resize the frame to the correct image size (128x128)
frame1 = cv2.resize(frame, (128, 128))
# Convert the frame to a numpy array and normalize it
image = np.asarray(frame1).astype(np.float32) / 255.0
# Add batch dimension
image = np.expand_dims(image, axis=0)
# Make a prediction on the image
probabilities = sess.run([output_name], {input_name: image})
# Save the predicted class
predicted_class_index = np.argmax(probabilities)
predicted_class_label = class_labels[predicted_class_index]
# Display the predicted class label on the frame
cv2.putText(frame, predicted_class_label, (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1,(0, 0, 255), 2)
# Display the frame
cv2.imshow('Recycle Detection', frame)
```

Results



- Model can be deployed and run in conjunction with a camera feed
- Camera response time can be slow, and the frame is tinted.
- Defaults to plastic typically, switching when object comes into view
    - Glass is difficult to classify correctly

Conclusions and Future work

- Image classification for recycled products can be deployed on an embedded device for practical use
- Using MobileNetV2 as a transfer layer works for accuracy and size.
- Future steps:
  - 2 Stage classifier
    - Stage 1: Trash or Recyclable
    - Stage 2: Classify Recyclable
  - Applying Federated Learning to dynamically update/ improve model from multiple devices
  - Improved image quality

Works Cited

[1]JetsonHacksNano. “Jetsonhacksnano/CSI-Camera.” GitHub, github.com/JetsonHacksNano/CSI-Camera. Accessed 7 May 2025.

[2]Ozkefe, Feyza. “Trashnet.” Kaggle, 19 Nov. 2021, www.kaggle.com/datasets/feyzazkefe/trashnet/data.

[3]Wang, Jinyeong. “Recycle_Classification_Dataset.” Kaggle, 23 Dec. 2020, www.kaggle.com/datasets/jinfree/recycle-classification-dataset.