Garbage CREW

AI-Powered Waste Sorting Robot

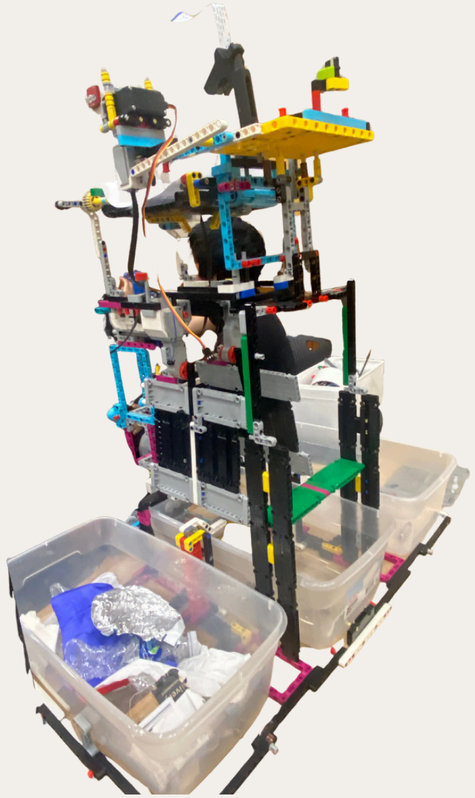
Machine Learning, Robotics, Embedded Systems, and Computer Vision

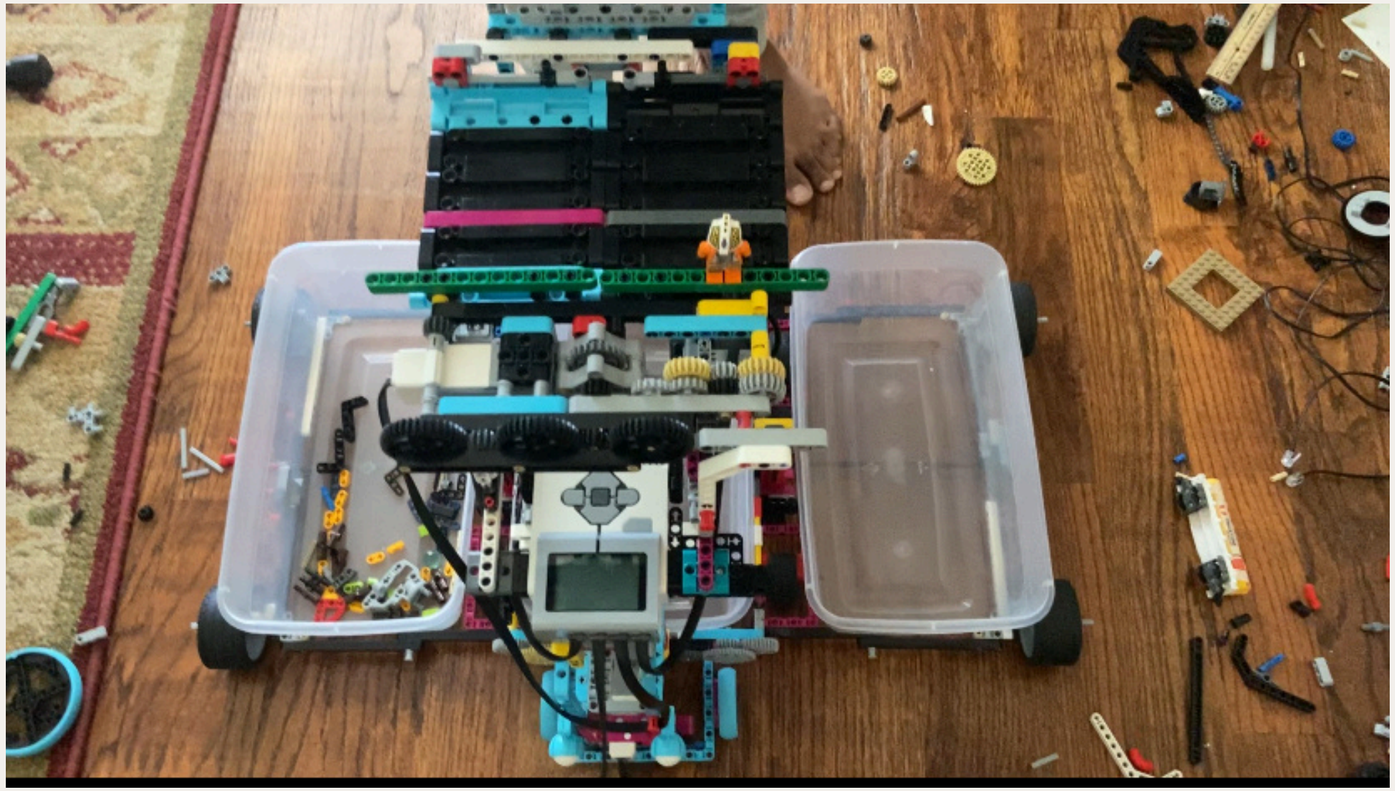
What is the issue we are solving?

Current waste sorting is slow, costly, and often inaccurate, causing recyclables to end up in landfills.

What did we do to solve this issue?

We developed a robot using an EV3 from scratch and with a lot of work integrated it with a raspberry pi for the image classification.





Project overview

01

What is the objective?

To automatically classify and sort waste into Trash, Compost, or Recyclables using AI and robotics.

02

What was it deployed on?

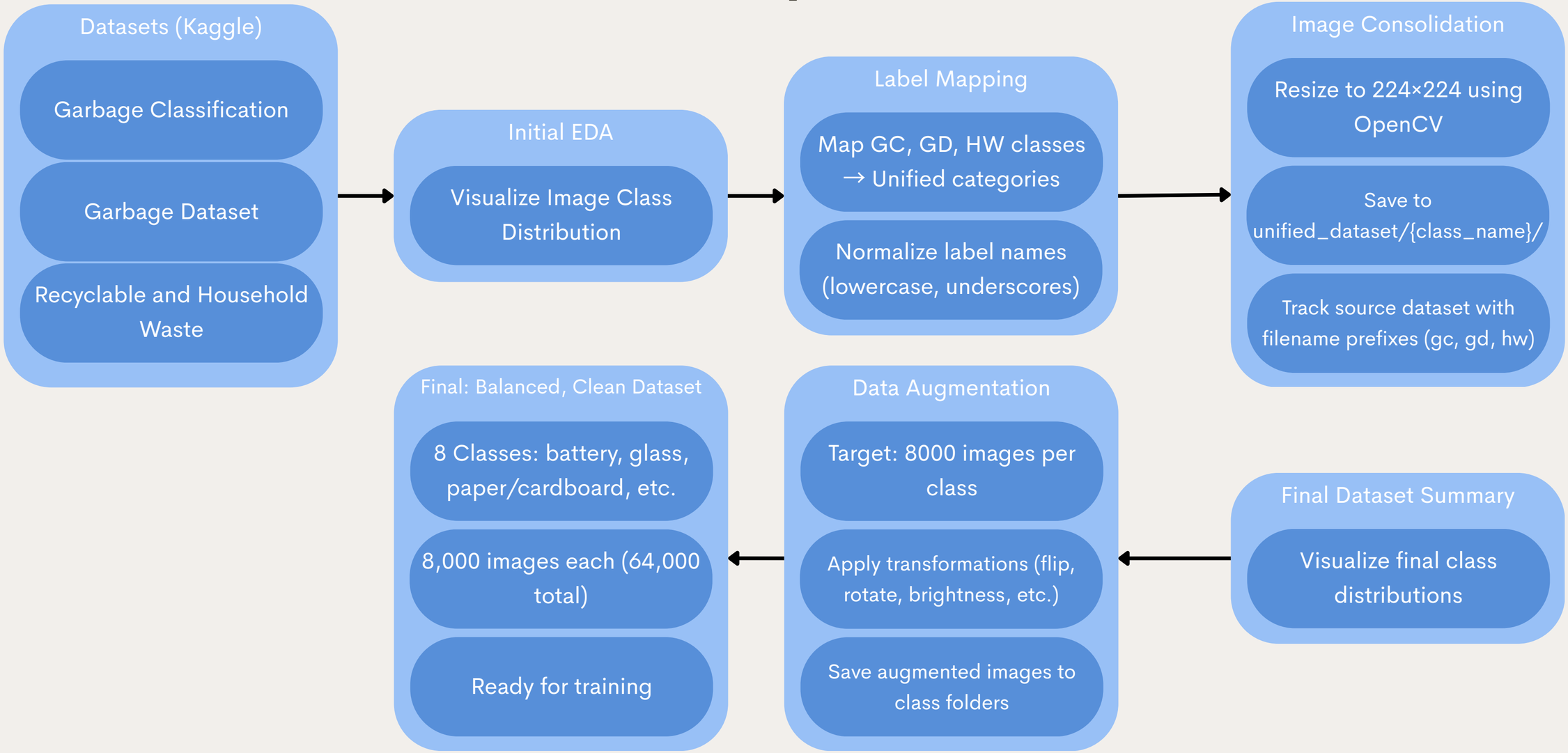
Raspberry Pi + LEGO Mindstorms EV3 + Camera

03

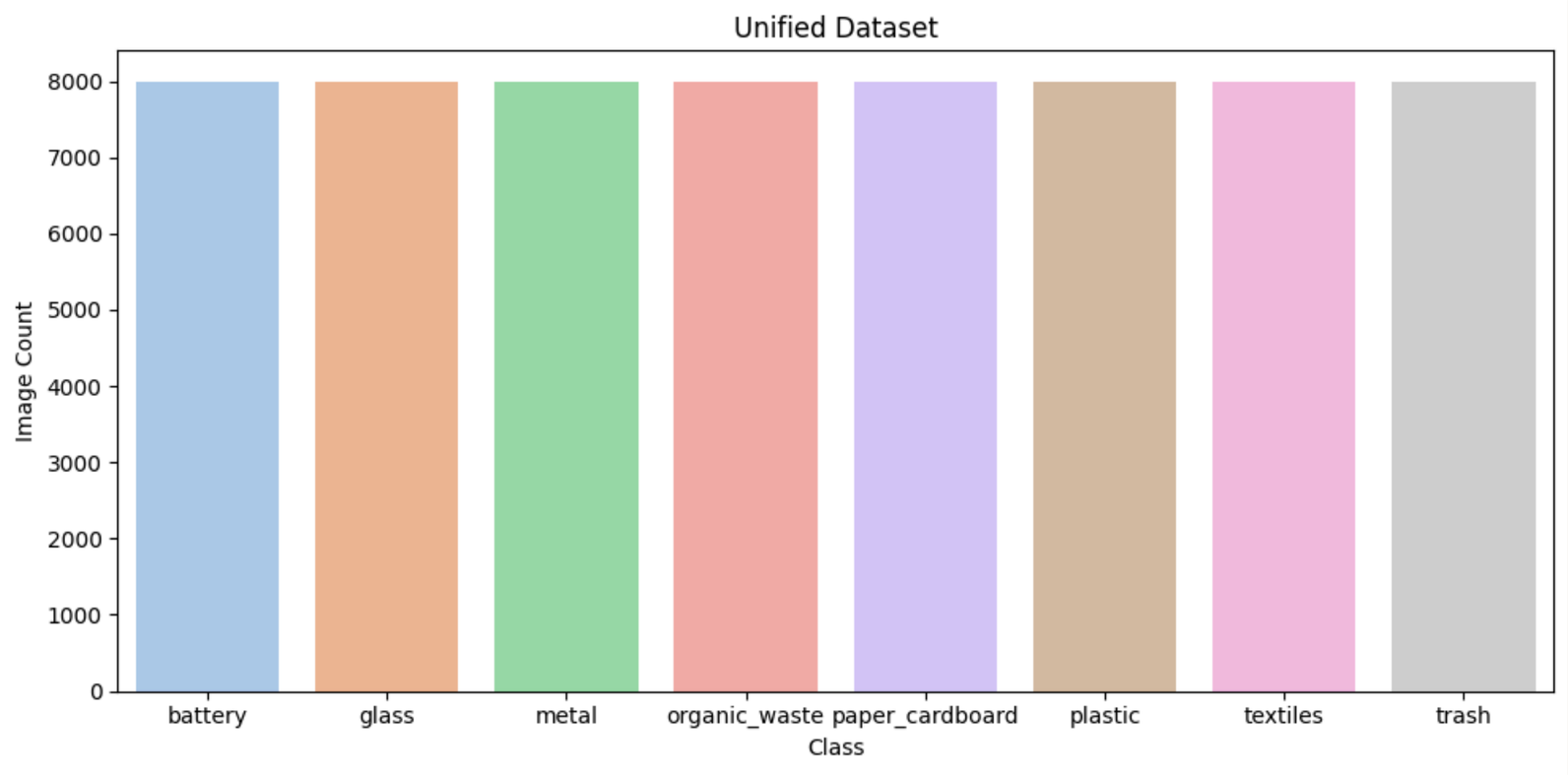
MAIN LIBRARIES USED

torch, torchvision, opencv_python, albumentations, timm, ultralytics, numpy, matplotlib, seaborn, gpiozero

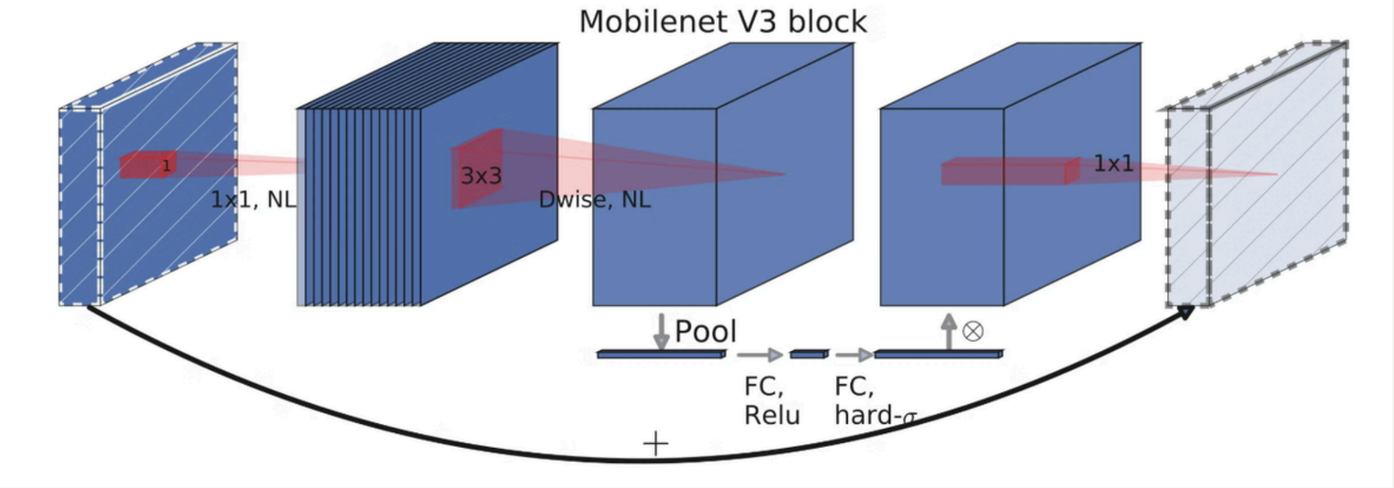
Data Preparation

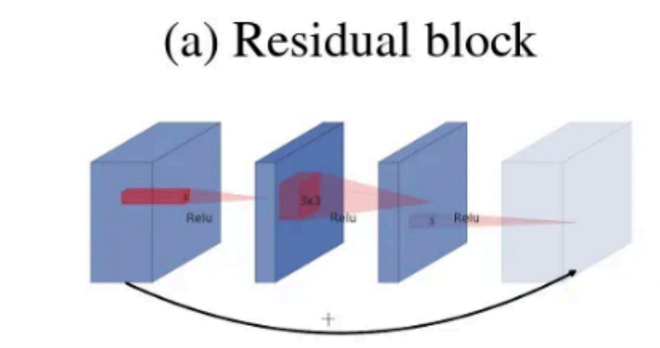


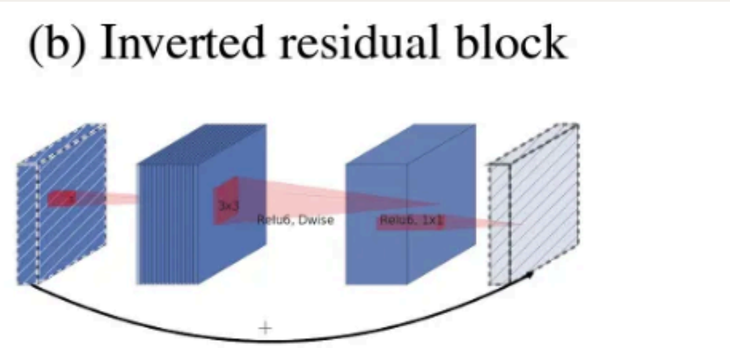
Dataset Class Distribution

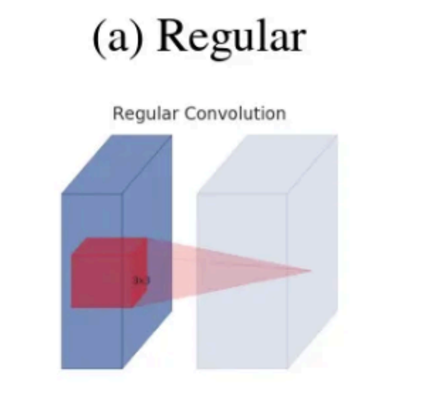


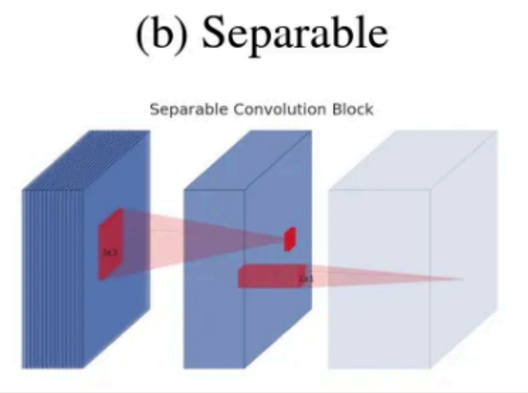
Model Architecture- MobileNetV3-Large

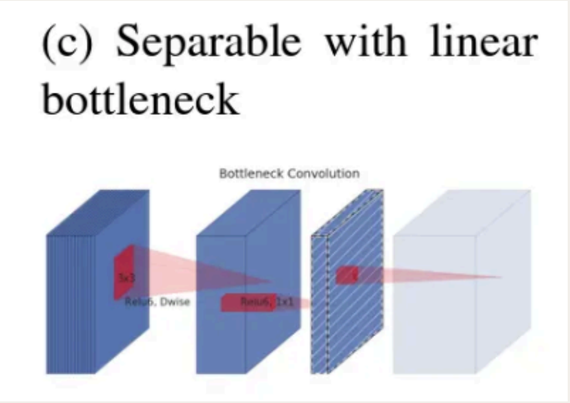


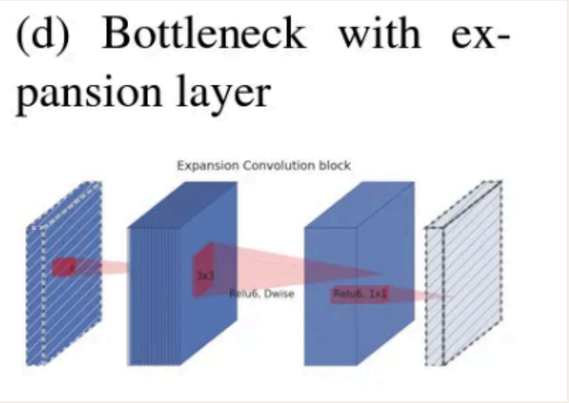












Images from https://medium.com/@RobuRishabh/understanding-and-implementing-mobilenetv3-422bd0bdfb5a

THE MODEL WE USED

MobileNetV3 Large (Pre-Trained on ImageNet)

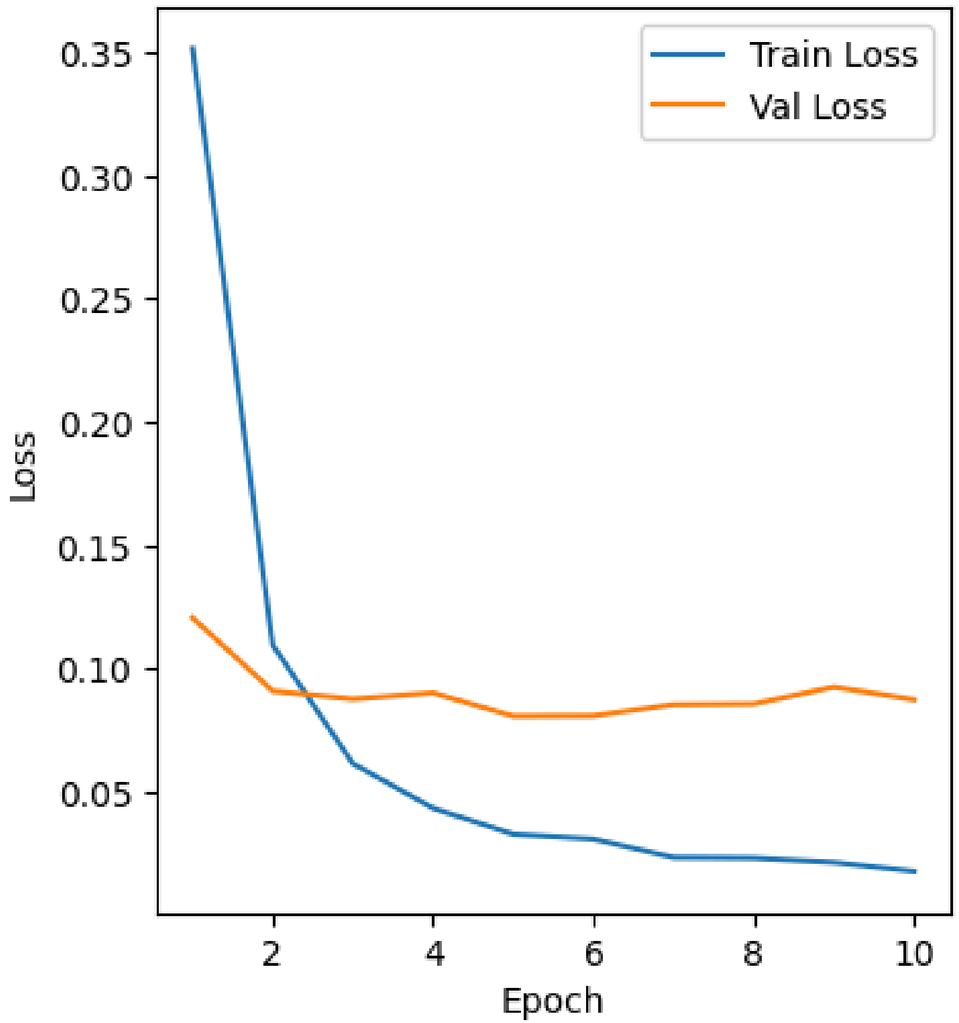
- Why?: Lightweight, fast, and optimized for deployment on mobile/edge devices
- Modification: Final classification layer changed to output 8 waste categories
  - battery
  - glass
  - metal
  - organic_waste
  - paper_cardboard
  - plastic
  - textiles
  - trash
- These are later grouped into: Recyclable, Compost, Trash

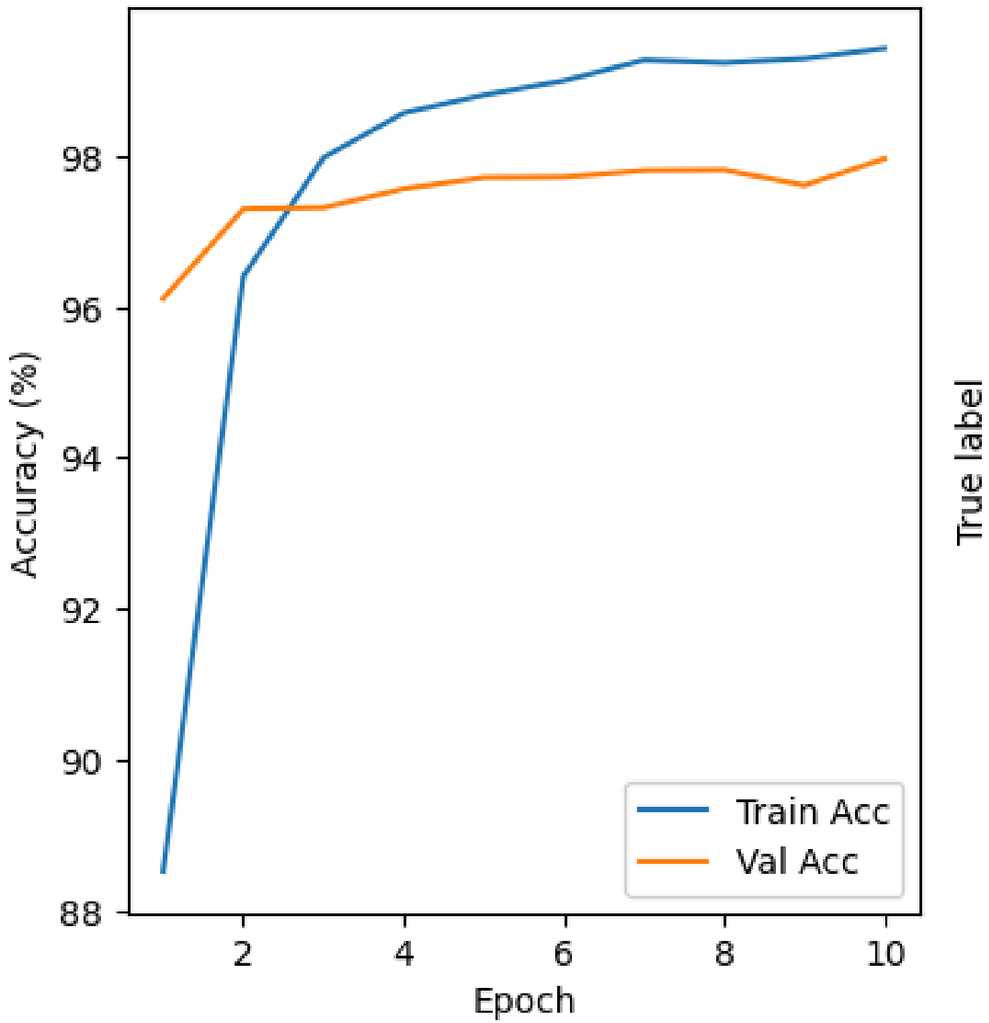
Training Details

- Input Size: 224×224
- Batch Size: 32
- Epochs: 10
- Loss Function: Cross Entropy Loss
- Optimizer: Adam (learning rate = 1e-4)
- Train/Val Split: 80/20, stratified by class

Final Train Accuracy: 99.34%

Validation Accuracy: **97.98%**

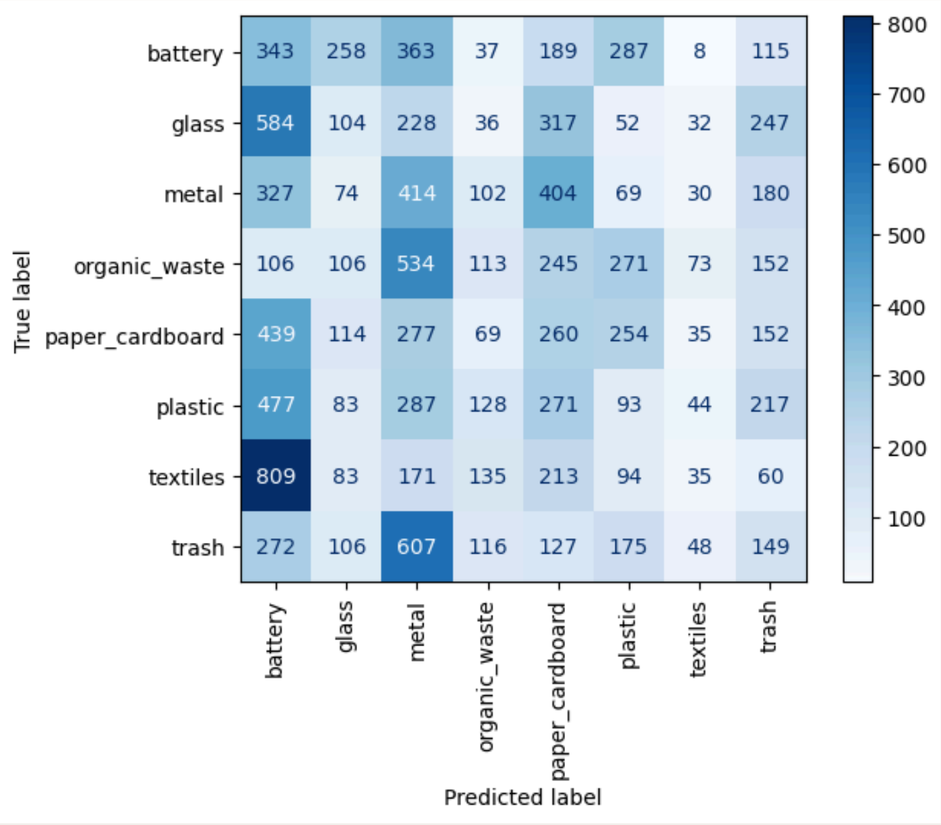

Loss Over Epochs
Train Loss
Val Loss
0.35
0.30
0.25
0.20
0.15
0.10
0.05
Loss
2
4
6
8
10
Epoch


Accuracy Over Epochs
98
96
94
92
90
88
Accuracy (%)
Train Acc
Val Acc
2
4
6
8
10
Epoch
True label

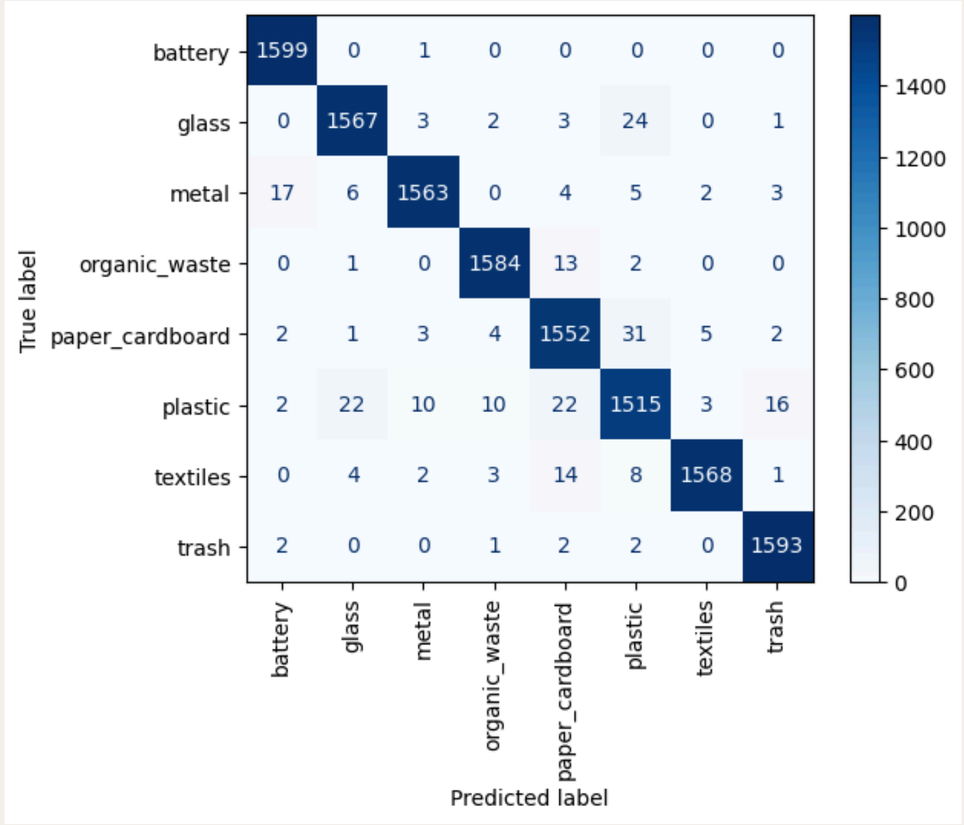
Sid

Bharath

Confusion Matrices



Pretrained



Finetuned

Sid

Metrics

category	precision	recall	f1-score
battery	10.22%	21.44%	13.84%
glass	11.21%	6.50%	8.23%
metal	14.37%	**25.88%**	**18.48%**
organic_waste	**15.35%**	7.06%	9.67%
paper_cardboard	12.83%	16.25%	14.34%
plastic	7.18%	5.81%	6.42%
textiles	11.48%	2.19%	3.67%
trash	11.71%	9.31%	10.38%
accuracy	11.80%	11.80%	11.80%
macro avg	11.79%	11.80%	10.63%
weighted avg	11.79%	11.80%	10.63%

Pre-Trained

category	precision	recall	f1-score
battery	98.60%	**99.90%**	**99.30%**
glass	97.90%	97.90%	97.90%
metal	98.80%	97.70%	98.20%
organic_waste	98.80%	99.00%	98.90%
paper_cardboard	96.40%	97.00%	96.70%
plastic	95.50%	94.70%	95.10%
textiles	**99.40%**	98.00%	98.70%
trash	98.60%	99.60%	99.10%
accuracy	98.00%	98.00%	98.00%
macro avg	98.00%	98.00%	98.00%
weighted avg	98.00%	98.00%	98.00%

Fine-Tuned

How Did We Optimize Our Model?

OPTIMIZATION STRATEGIES :

Torchscript compilation:

```
    else:
        model = torch.load(
            "./models/mobilenetv3_garbage_classifier.pt",
            map_location=DEVICE,
            weights_only=False,
        )
    model = torch.jit.script(model)

model = torch.jit.optimize_for_inference(model)
model.to(DEVICE)
model.eval()
```

Post-Training Static Quantization

```
model.qconfig = torch.quantization.get_default_qconfig("qnnpack")
torch.quantization.prepare(model, inplace=True)

num_samples = 0
max_samples = 500

with torch.no_grad():
    for images, _ in val_loader:
        batch_size = images.size(0)
        model(images)
        num_samples += batch_size
        if num_samples >= max_samples:
            break

torch.quantization.convert(model, inplace=True)
```

Sid

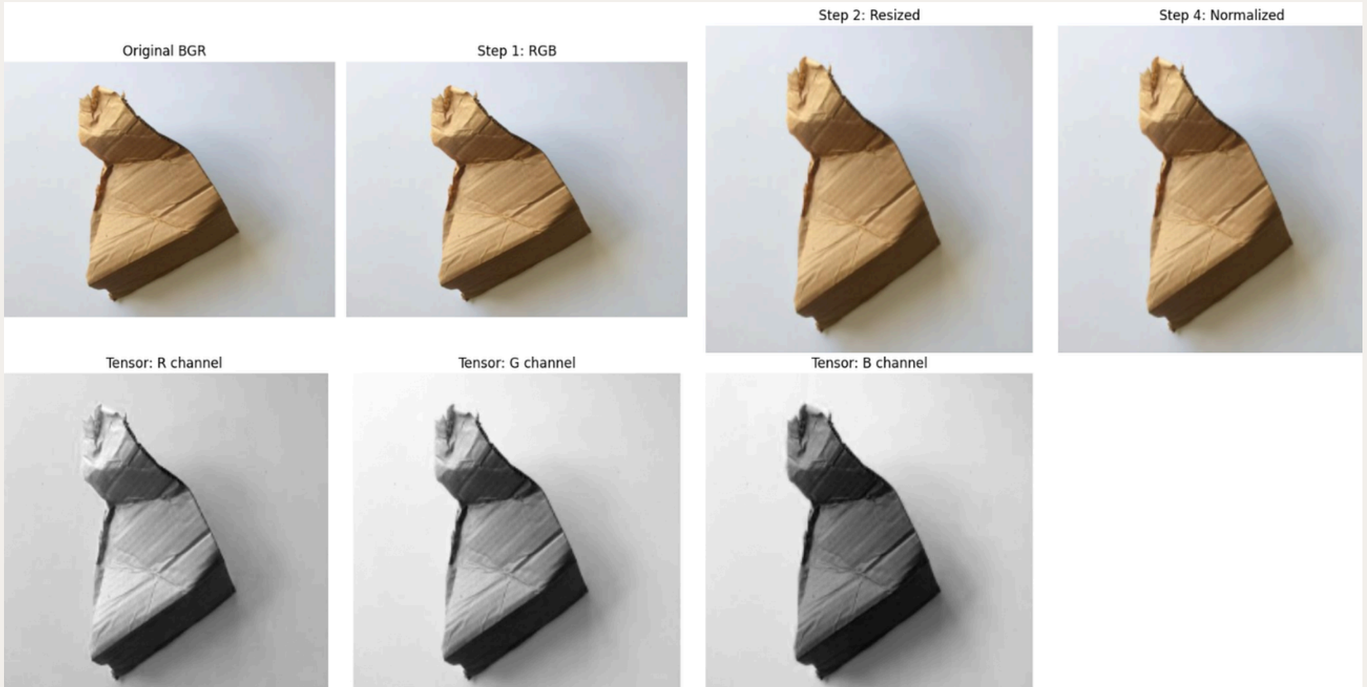
REAL-TIME INFERENCE PIPELINE

```python
cap = cv2.VideoCapture(0, cv2.CAP_V4L2)
cap.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter_fourcc(*'MJPG'))
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 600)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)
cap.set(cv2.CAP_PROP_FPS, 36)
```

```python
    model = torch.jit.script(model)

model = torch.jit.optimize_for_inference(model)
model.to(DEVICE)
model.eval()
```

REAL-TIME INFERENCE PIPELINE

```python
def preprocess_cv2(image_bgr):
    image_rgb = cv2.cvtColor(image_bgr, cv2.COLOR_BGR2RGB)
    image_resized = cv2.resize(image_rgb, (224, 224))
    image_tensor = torch.from_numpy(image_resized).float() / 255.0
    image_tensor = image_tensor.permute(2, 0, 1)
    image_tensor = transforms.Normalize(
        mean=[0.485, 0.456, 0.406],
        std=[0.229, 0.224, 0.225]
    )(image_tensor)
    return image_tensor.unsqueeze(0)
if frozen and snapshot_mode and not classified:
    input_tensor = preprocess_cv2(freeze_frame).to(DEVICE)
    outputs = model(input_tensor)
    pred = torch.argmax(outputs, 1).item()
    label = CLASSES[pred]
    category = CLASS_TO_CATEGORY[label]
```




```python
CLASSES = [
    "battery", "glass", "metal", "organic_waste",
    "paper_cardboard", "plastic", "textiles", "trash"
]

CLASS_TO_CATEGORY = {
    "battery": "trash",
    "glass": "trash",
    "metal": "recyclable",
    "organic_waste": "compost",
    "paper_cardboard": "recyclable",
    "plastic": "recyclable",
    "textiles": "trash",
    "trash": "trash",
}
```

REAL-TIME INFERENCE PIPELINE

```python
def move_servo(angle, delay):
    try:
        servo.angle = angle
        time.sleep(delay)
        servo.angle = 360
    except Exception as e:
        print(f"Servo error: {e}")

def sort_to_compost():
    move_servo(90, 1.5)

def sort_to_recyclable():
    move_servo(90, 1.0)

def sort_to_trash():
    move_servo(90, 0.5)
```

```python
print(f"Detected: {label} - Category: {category}")
if category == "compost":
    sort_to_compost()
elif category == "recyclable":
    sort_to_recyclable()
elif category == "trash":
    sort_to_trash()
```

System Architecture Diagram

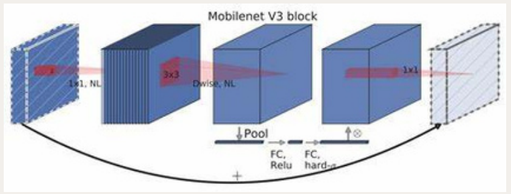
This is the architecture of our project and how the individual components interact with each other:

Raspberry Pi Camera Module 2

Video Feed

Data Preprocessing

Convolutional Neural Network



High-Torque Servo

Recycling

Compost

Trash

EV3 Large Motor

EV3 Control Brick

EV3 Touch Sensor

Ainesh and Aman

Code for the EV3 Brick hub

Here is the Code of the EV3 control Brick that causes the EV3 motors to categorize between 3 different categories based on the time pressed on the Touch Sensor

```python
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import Motor, TouchSensor
from pybricks.parameters import Port
from pybricks.tools import wait

# Initialize EV3 brick and devices
ev3 = EV3Brick()

# Motors
left_motor = Motor(Port.A)      # LEFT wheel
right_motor = Motor(Port.B)     # RIGHT wheel
motor_1 = Motor(Port.C)         # Possibly sorting arm 1
motor_2 = Motor(Port.D)         # Possibly sorting arm 2

# Touch sensors
touch_sensor = TouchSensor(Port.S1)  # Object detection or trigger sensor
release_sensor = TouchSensor(Port.S2)  # Used for release detection (optional)
```

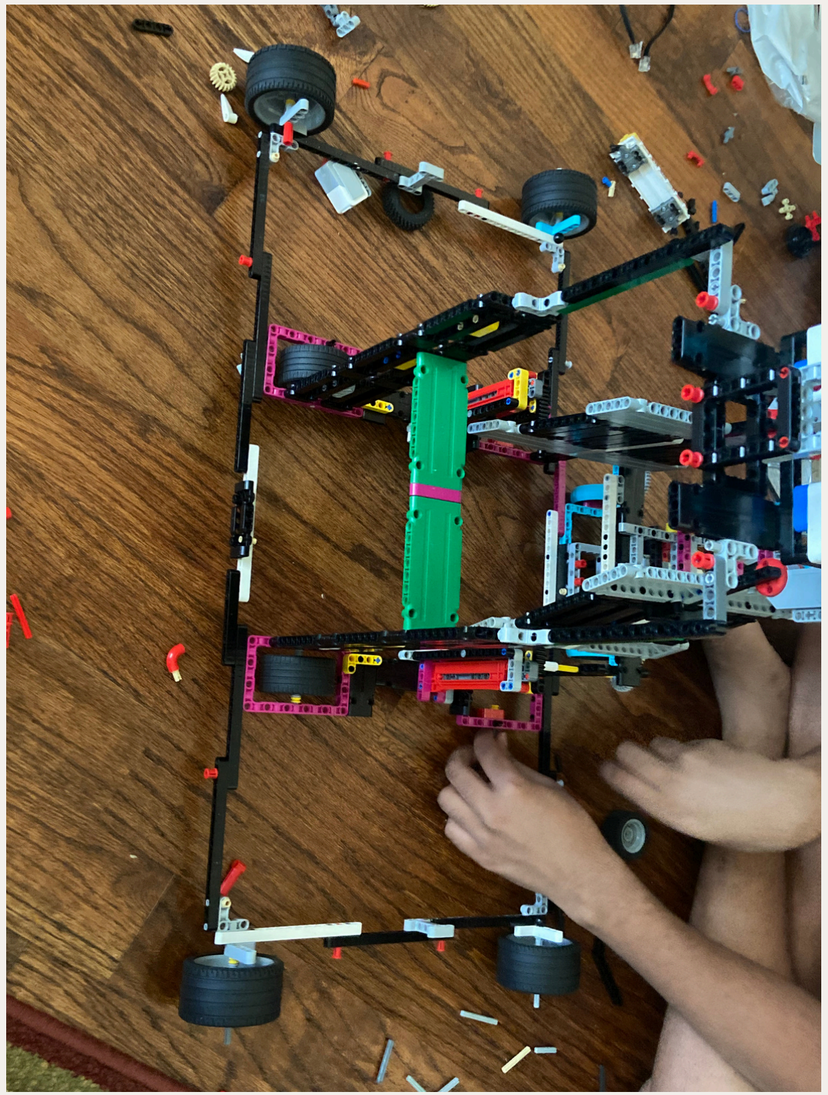
```python
# Main loop logic
while True:
    # Reset all motors
    left_motor.reset_angle(0)
    right_motor.reset_angle(0)
    motor_1.reset_angle(0)
    motor_2.reset_angle(0)

    # Wait for object to be placed (sensor pressed)
    while not touch_sensor.pressed():
        wait(10)

    # Wait for sensor to be released
    while touch_sensor.pressed():
        wait(10)

    # Simulate a variable reset
    TimeAlpha = 0

    # Example object detection logic
    object_detected = True  # You can replace this with real sensor-based condition

    if object_detected:
        # Rotate in place to scan/align
        left_motor.run(-100)
        right_motor.run(100)

        # Wait for a condition or time to pass
        wait(500)  # Wait 0.5s (or until another sensor condition is met)
```

```python
    # Stop motors
    left_motor.stop()
    right_motor.stop()

    # Reset counter
    Counter = 0

    # Sorting motion
    motor_1.run_angle(500, 90)   # e.g. swing right
    motor_2.run_angle(500, -90)  # swing left or retract

else:
    # Alternate condition — e.g. wrong object, error state
    alternate_condition = False

    if alternate_condition:
        motor_1.run_angle(500, -180)
    else:
        motor_2.run_angle(500, 180)

# Final motor reset (optional)
motor_1.reset_angle(0)
motor_2.reset_angle(0)
```

Ainesh

Future improvements

- Improvements to models accuracy in real life applications
- Scale of the model to be able to accomodate any type of waste
- Background Removal/Object Extraction

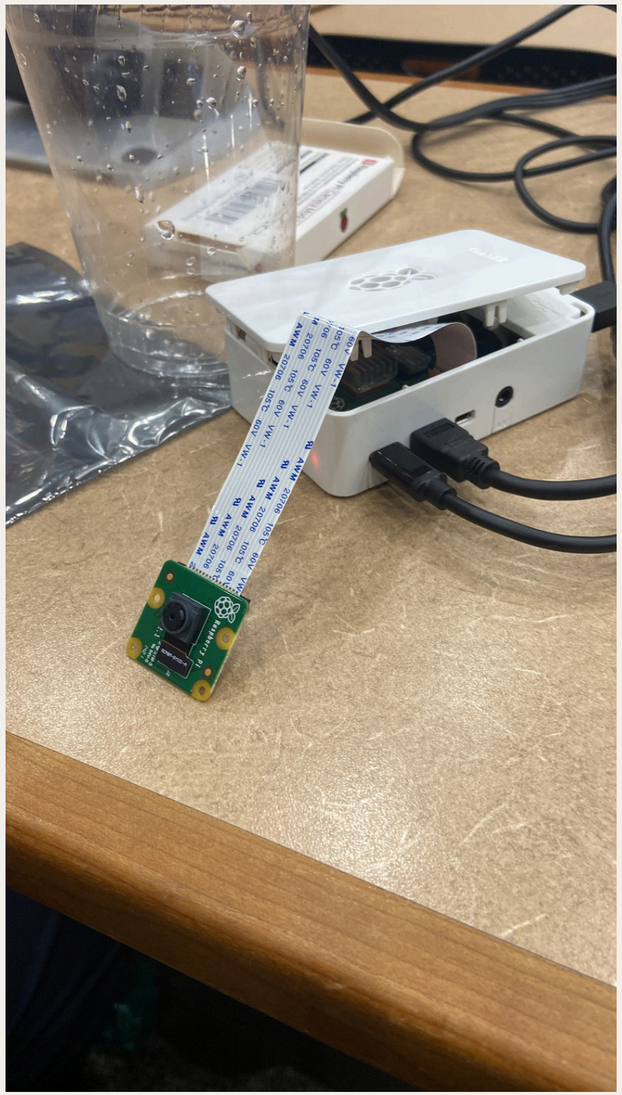


Bharath

Limitations and Challenges

- Limitations include the size, weight and type of object

- Throughout this poject we had many challenges like having to contruct a whole robot from scratch and the espectially tough part was finding out how to connect the raspberry pi to the ev3 controller.



Bharath

The End

github: https://github.com/Garbage-Crew/Garbage-Crew