Recycling with Deep Learning

Group 3
Pin-Hao Pan, Ching Hsuan Lin , Kayla Candice Huang, Zhe-Yu Lin

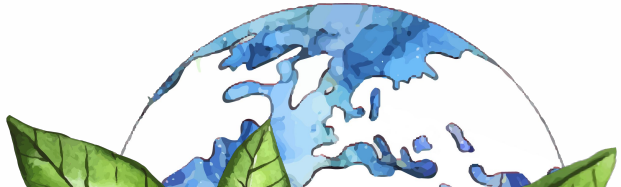
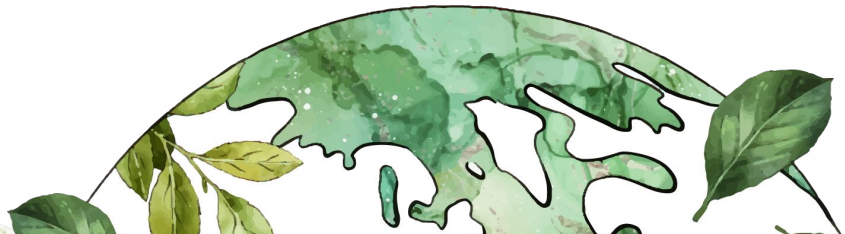


TABLE OF CONTENTS

A Glimpse into Everyday Waste Disposal

01

MOTIVATION



WHAT is happening?

- **Enormous landfill footprint:** The United States has lost approximately 1.8 million acres of land to active landfills.
- **Increased environmental harm:** Municipal solid waste (MSW) landfills are the third-largest source of human-related methane emissions in the United States[3].

Difficulties in encouraging recycling:

- **Higher municipal costs:** In many areas, the cost of recycling now exceeds that of landfill disposal, highlighting the urgent need for improvement.
- **Low diversion rates:** Only 32.1% of American waste is recycled or composted[1], and the National Recycling Goal is 50% by 2030[2].

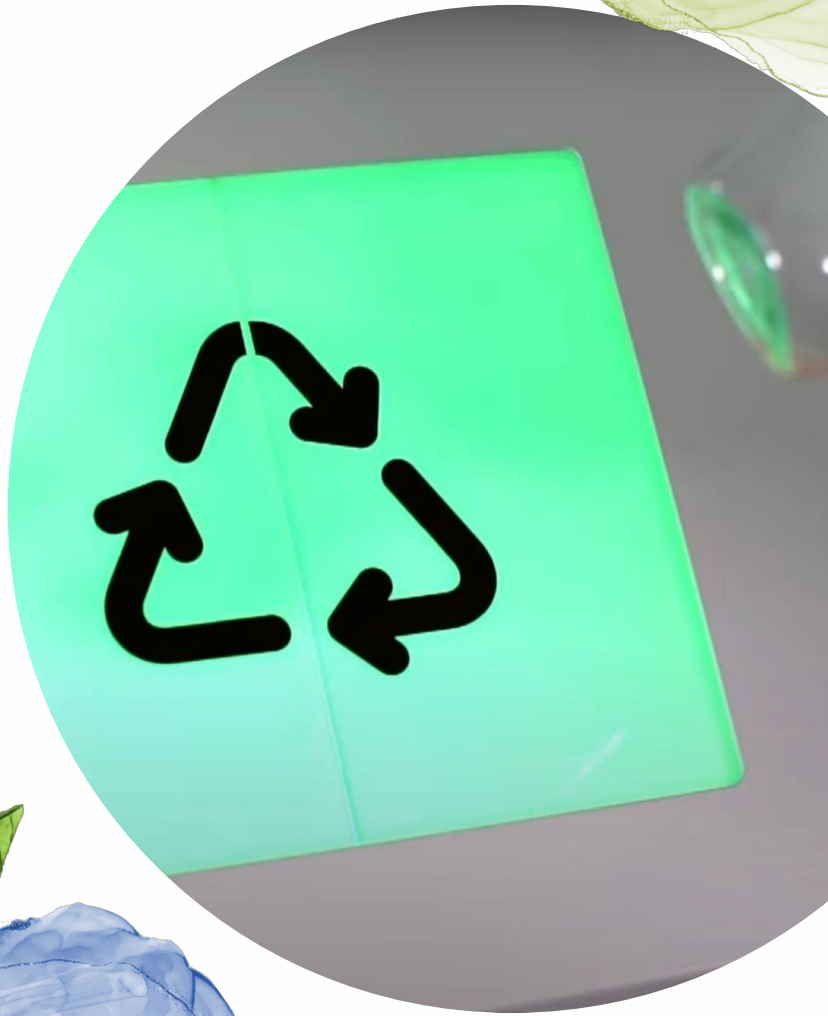
1.Blanco, Christian, Calvin Spanbauer, and Sara Stienecker. "America's Broken Recycling System." California Management Review, May 30, 2023. https://cmr.berkeley.edu/2023/05/america-s-broken-recycling-system/
2.U.S. Environmental Protection Agency. (n.d.). U.S. National Recycling Goal. EPA. Retrieved April 26, 2025, from https://www.epa.gov/circulareconomy/us-national-recycling-goal

HOW we aim to solve?

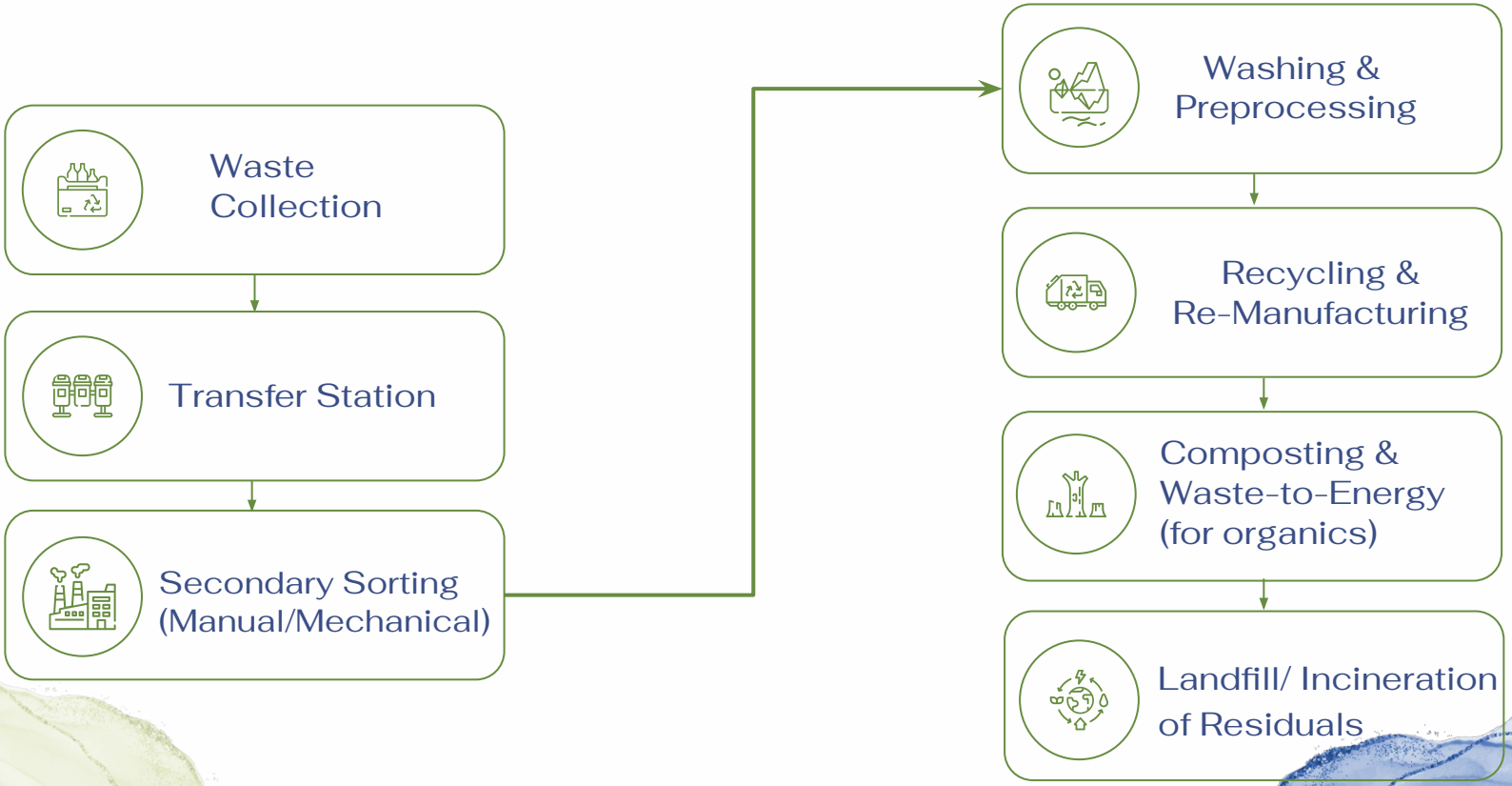
Create a fine-tuned **CNN-based image classification model** for garbage, that can be embedded into smart waste bins that help the public dispose of items more accurately in the future.

This could benefit:

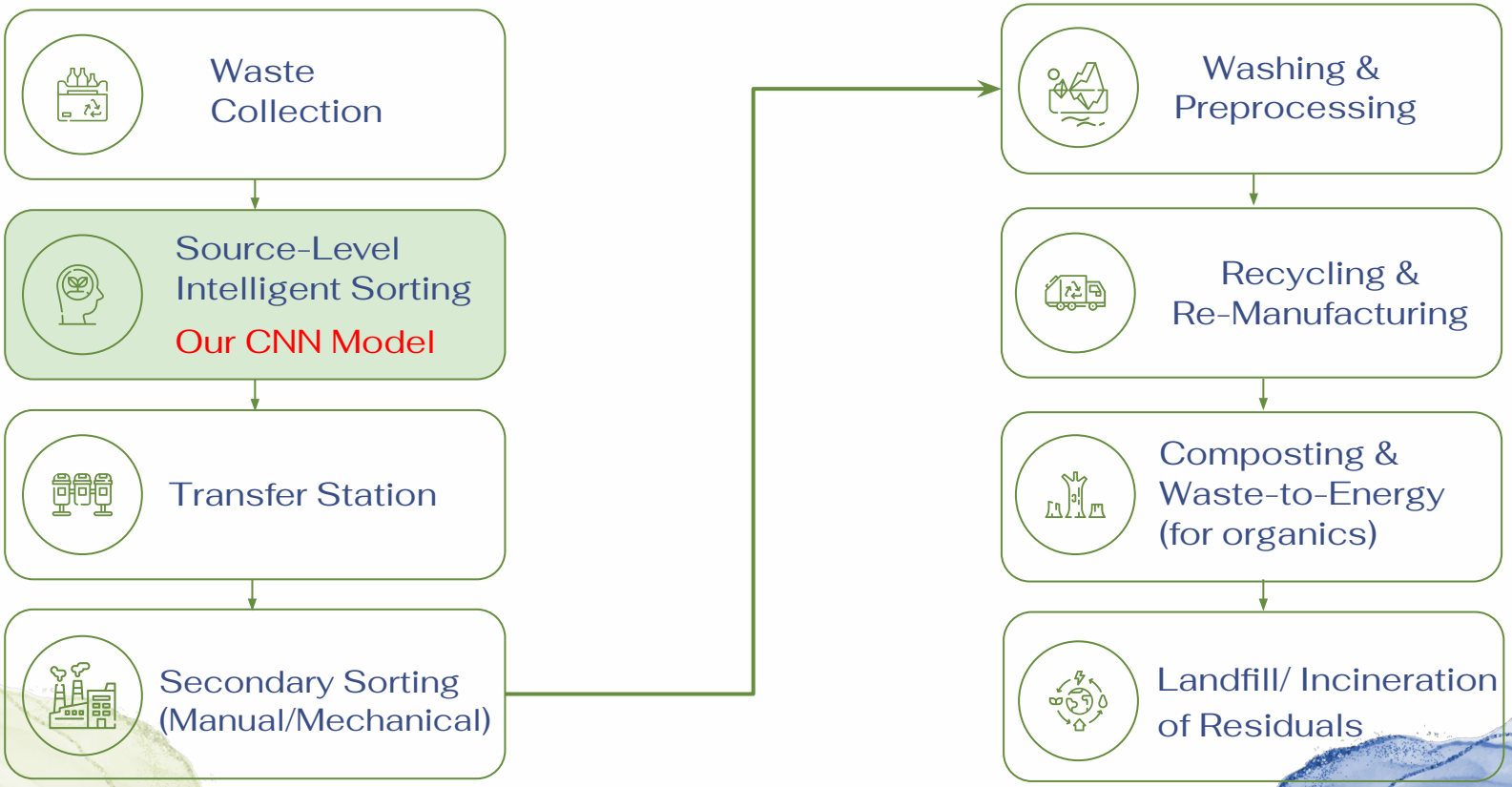
- **Government and Public Sector** - Policy promotion and environmental goals
- **Waste Management and Recycling Companies** - Improve sorting efficiency and reduce waste management costs
- **Apartment buildings and residential communities** - Improve residents' sorting habits and reduce waste management fines



Waste Management Process Flowchart



Apply our model into process



Potential Real-world Application

02

DATASET & PREPROCESSING

HOW we get our dataset?

Dataset

Our dataset contains **677** images across six different types of waste

Take photos

We took photos of garbage in our daily life

Label

We manually classified and labeled our image data

Split

We split our data into train, validation, and test sets

WHAT are the labels?

Plastic

79/17/18

Glass

70/15/16

Metal

79/17/18

General Waste

88/19/20

Organic Waste

84/18/19

Paper

70/15/15

HOW we process the data?

Steps	Actions	Reasons
Format Filtering	Convert image to jpeg files	Avoid errors or inconsistencies caused by other file formats
Orientation Correction	Correct image orientation	Automatically corrects sideways or upside-down photos
Mode Conversion	Convert image to RGB (remove transparency)	Prevent transparency-related issues when saving as jpeg file
Resizing	Scale down images if larger than 224x224 but no upscaling	Standardize the image dimensions
Center on White Background	Center resized image and add white padding to non-square images	Avoid distortion and ensure standardized input size for models.
Rename and Save	Unified file names and saved them	Standardizes file name for easier processing later

Example of Plastic Waste

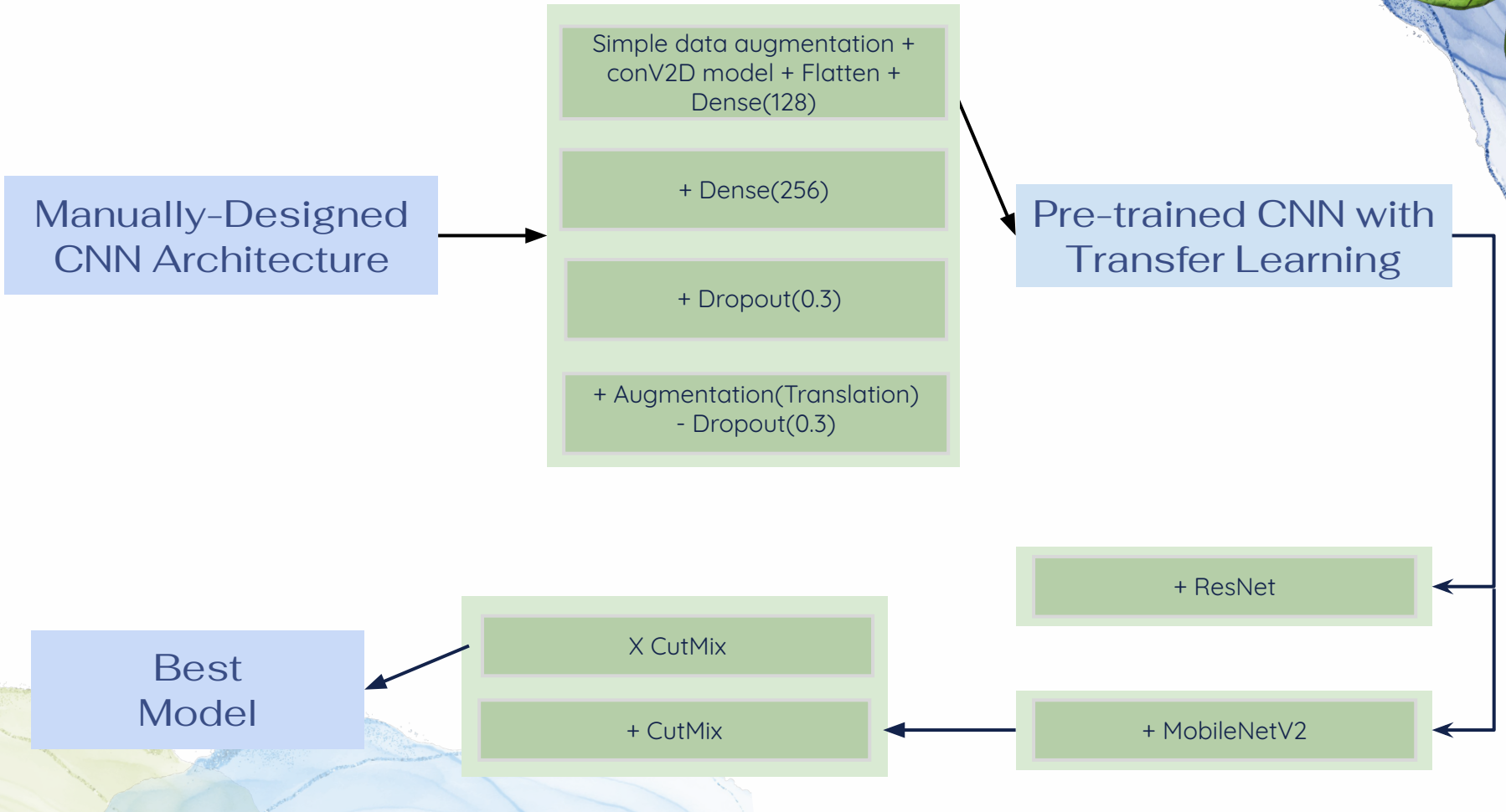
Example of Plastic Waste after down scaling

Example of Metal Waste

Example of Metal Waste after down scaling

03 Modeling

Model Flowchart



Manually-Designed CNN Architecture

Simple data augmentation + conV2D model + Flatten + Dense(128)

```
# --- Standard augmentation ---
data_augmentation = keras.Sequential([
    layers.RandomFlip("horizontal"),
    layers.RandomRotation(0.1),
    layers.RandomZoom(0.2)
])

# --- Model definition ---
inputs = keras.Input(shape=(IMG_SIZE, IMG_SIZE, 3))
x = data_augmentation(inputs)
x = layers.Conv2D(filters=32, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=64, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=128, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Flatten()(x)
outputs = layers.Dense(NUM_CLASSES, activation="softmax")(x)

model = keras.Model(inputs=inputs, outputs=outputs)

# --- Compile model ---
model.compile(
    optimizer="rmsprop",
    loss="sparse_categorical_crossentropy",
    metrics=["accuracy"]
)
```

Manually-Designed CNN Architecture

Model 1 + Dense(256)

```
# --- Standard augmentation ---
data_augmentation = keras.Sequential([
    layers.RandomFlip("horizontal"),
    layers.RandomRotation(0.1),
    layers.RandomZoom(0.2)
])

# --- Model definition ---
inputs = keras.Input(shape=(IMG_SIZE, IMG_SIZE, 3))
x = data_augmentation(inputs)
x = layers.Conv2D(filters=32, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=64, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=128, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=256, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Flatten()(x)
outputs = layers.Dense(NUM_CLASSES, activation="softmax")(x)

model = keras.Model(inputs=inputs, outputs=outputs)

# --- Compile model ---
model.compile(
    optimizer="rmsprop",
    loss="sparse_categorical_crossentropy",
    metrics=["accuracy"]
)
```

Manually-Designed CNN Architecture

Model 1 + Dropout(0.3)

```
# --- Standard augmentation ---
data_augmentation = keras.Sequential([
    layers.RandomFlip("horizontal"),
    layers.RandomRotation(0.1),
    layers.RandomZoom(0.2)
])

# --- Model definition ---
inputs = keras.Input(shape=(IMG_SIZE, IMG_SIZE, 3))
x = data_augmentation(inputs)
x = layers.Conv2D(filters=32, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=64, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=128, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=256, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Flatten()(x)
x = layers.Dropout(0.3)(x)
outputs = layers.Dense(NUM_CLASSES, activation="softmax")(x)

model = keras.Model(inputs=inputs, outputs=outputs)

# --- Compile model ---
model.compile(
    optimizer="rmsprop",
    loss="sparse_categorical_crossentropy",
    metrics=["accuracy"]
```

Manually-Designed CNN Architecture

Model 3 + Augmentation(Translation) - Dropout(0.3)

```
# --- Standard augmentation ---
data_augmentation = keras.Sequential([
    layers.RandomFlip("horizontal"),
    layers.RandomRotation(0.1),
    layers.RandomZoom(0.2),
    layers.RandomTranslation(0.1, 0.1)
])

# --- Model definition ---
inputs = keras.Input(shape=(IMG_SIZE, IMG_SIZE, 3))
x = data_augmentation(inputs)
x = layers.Conv2D(filters=32, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=64, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=128, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=256, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Flatten()(x)
outputs = layers.Dense(NUM_CLASSES, activation="softmax")(x)

model = keras.Model(inputs=inputs, outputs=outputs)

# --- Compile model ---
model.compile(
    optimizer="rmsprop",
    loss="sparse_categorical_crossentropy",
    metrics=["accuracy"]
)
```

Other Adjustments We've Tried

Data Augmentation Techniques	RandomContrast(0.2) RandomBrightness(0.2)
Feature Extraction Layers	GlobalAveragePooling2D
Regularization Techniques	BatchNormalization Dropout(0.15)
Fully Connected Layers	Dense(256) / Dense(128)

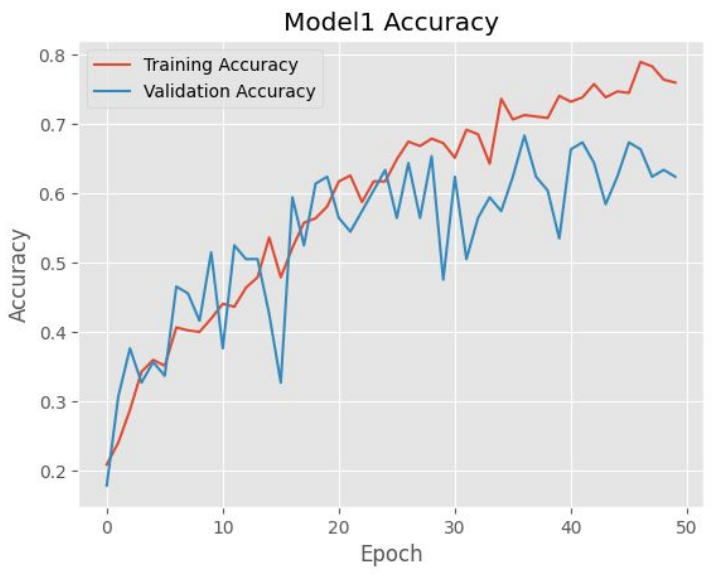
Using LIME to Explain Model

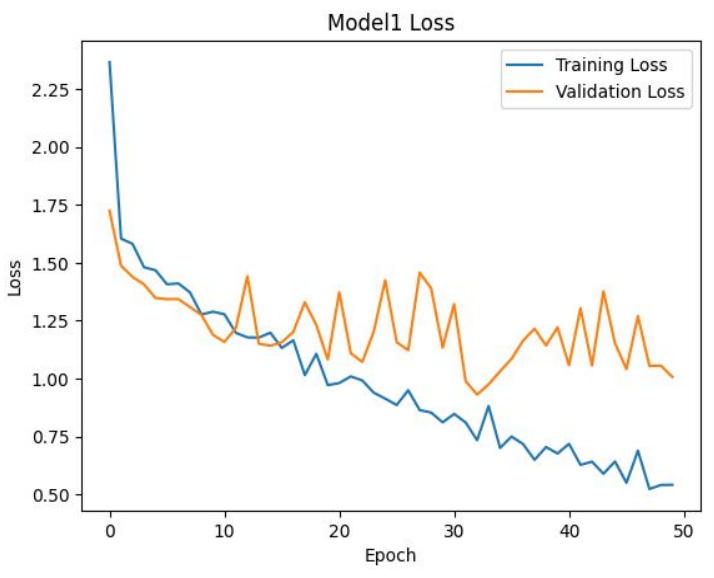
Original

LIME for class 4

Predicted: Paper
Confidence: 100.00%

Model 1 Accuracy and Loss





CNN model and results

	Model Structure	Accuracy
Model 1	Simple data augmentation + conV2D model + Flatten + Dense(128)	0.7075
Model 2	Model 1 + Dense(256)	0.6792
Model 3	Model 2 + + Dropout(0.3)	0.6415
Model 4	Model 3 + Augmentation(Translation) - Dropout	0.6604

ResNet50

- *ResNet*
  - **Pros**: Top-tier accuracy on large datasets; strong feature learning.
  - **Cons**: Heavy; slower inference; risk of overfitting

Our Model Pretrained Base: ResNet50 (ImageNet weights)
+
Model 1 head (custom layers on top) and Key modifications:

1. GlobalAveragePooling2D instead of MaxPooling2D(pool_size=2) + Flatten():
Why: By summarizing each feature map into one value, GAP drastically cuts the number of parameters (→ less overfitting) and enforces spatial invariance across the entire map.

2. Added Dense(128, activation='relu') before the final softmax:
Why: This extra hidden layer lets the model learn richer non-linear combinations of the pretrained ResNet50 features, boosting its capacity to separate classes more cleanly.

```
# --- Model definition with ResNet50 backbone + custom Conv head ---
inputs = keras.Input(shape=(IMG_SIZE, IMG_SIZE, 3))
x = data_augmentation(inputs)

# 1) ResNet50 feature extractor
base_model = keras.applications.ResNet50(
    input_shape=(IMG_SIZE, IMG_SIZE, 3),
    include_top=False,
    weights="imagenet"
)
base_model.trainable = False
x = base_model(x, training=False)

# 2) Conv2D + Pooling
x = layers.Conv2D(32, 3, padding="same", activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(64, 3, padding="same", activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(128, 3, padding="same", activation="relu")(x)
x = layers.GlobalAveragePooling2D()(x)

x = layers.Dense(128, activation="relu")(x)
outputs = layers.Dense(NUM_CLASSES, activation="softmax")(x)

model = keras.Model(inputs=inputs, outputs=outputs)
```

MobileNetV2(w/o CutMix)

- *MobileNetV2(w/o CutMix)*
  - **Pros**: Fast to train & infer; small footprint—ideal for edge/mobile.
  - **Cons**: Basic augmentation only; limited regularization
    → may still overfit small datasets.

```
# --- Model definition ---
inputs = keras.Input(shape=(IMG_SIZE, IMG_SIZE, 3))
x = data_augmentation(inputs)

base_model = keras.applications.MobileNetV2(
    input_shape=(IMG_SIZE, IMG_SIZE, 3),
    include_top=False,
    weights="imagenet"
)
base_model.trainable = False

x = base_model(x, training=False)
x = layers.Conv2D(filters=32, kernel_size=3, padding="same",activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=64, kernel_size=3, padding="same",activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=128, kernel_size=3 ,padding="same",activation="relu")(x)
x = layers.GlobalAveragePooling2D()(x)
x = layers.Dense(128, activation="relu")(x)
outputs = layers.Dense(NUM_CLASSES, activation="softmax")(x)
```

MobileNetV2(w/ CutMix)

- *MobileNetV2(w/ CutMix)*
  - **Pros:** Provides strong regularization and improved generalization, boosts robustness to occlusion and noise, all with zero inference overhead.
  - **Cons:** Requires careful tuning (mixing ratio, pipeline integration), assumes patch area scales linearly with class presence, may hurt localization, and adds slight training overhead.

```
# Apply CutMix during training using a map function
train_dataset = train_dataset.map(cutmix, num_parallel_calls=AUTOTUNE)

# --- Model definition ---
base_model = MobileNetV2(input_shape=(IMG_SIZE, IMG_SIZE, 3), include_top=False, weights="imagenet")
base_model.trainable = False

inputs = keras.Input(shape=(IMG_SIZE, IMG_SIZE, 3))
x = data_augmentation(inputs)
x = base_model(x, training=False)
x = layers.Conv2D(filters=32, kernel_size=3, padding="same", activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=64, kernel_size=3, padding="same", activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=128, kernel_size=3 ,padding="same", activation="relu")(x)
x = layers.GlobalAveragePooling2D()(x)
x = layers.Dense(128, activation="relu")(x)
outputs = layers.Dense(NUM_CLASSES, activation="softmax")(x)

model = keras.Model(inputs, outputs)

# --- Compile model ---
model.compile(
    optimizer="rmsprop",
    loss="sparse_categorical_crossentropy",
    metrics=["accuracy"]
)
```

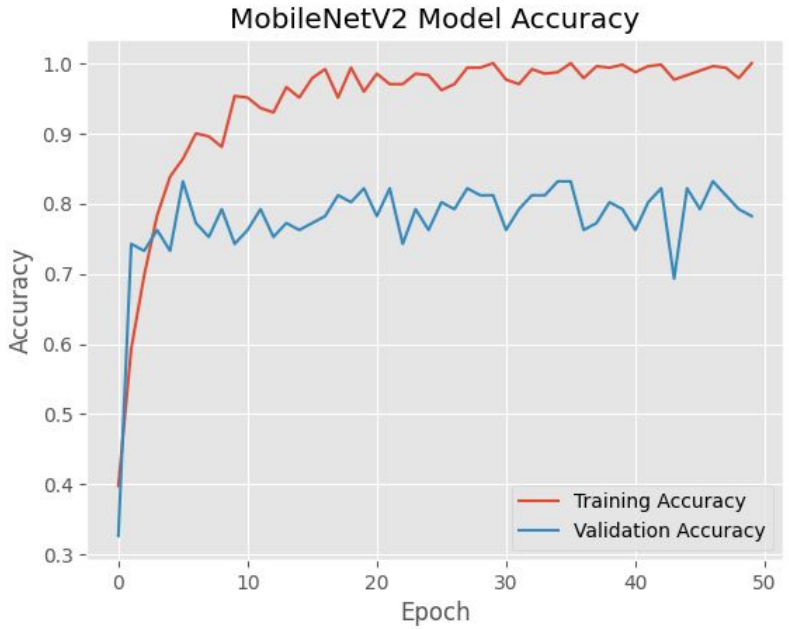
Using LIME to Explain Model

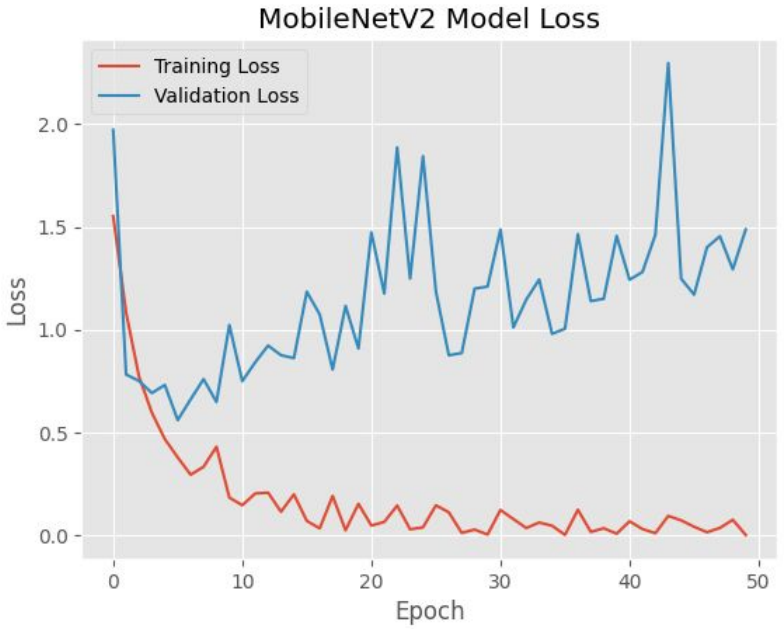
Original

LIME for class 5

Predicted: Plastic
Confidence: 69.31%

MobileNetV2 Accuracy and Loss



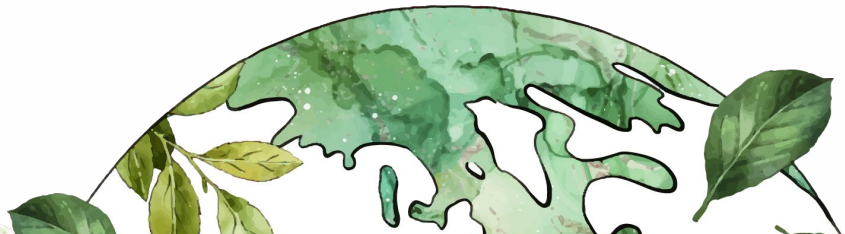


Test Accuracy of Pre-trained Models

	Accuracy
ResNet	0.2830
MobileNetV2 w/o CutMix	**0.8302**
MobileNetV2 w/ CutMix	0.8112

04

Conclusions



Conclusions

Even with limited data, we built a CNN model based on MobileNetV2, achieving 0.83 accuracy in classifying six types of waste.

This demonstrates that lightweight deep learning models can already bring tangible improvements to waste management. By leveraging our approach, we hope to:

Increase classification efficiency → Reduce the cost of recycling → Help reduce greenhouse gas emissions

Challenges

Inconsistent image formats

Different image formats can carry different types of information, leading to inconsistencies that cause problems during processing and model training.

Unable to handle multiple-object classification scenarios

Images containing multiple objects make it difficult for the model to perform single-label classification, resulting in poorer learning performance.

Limited data for training complex models

Although we had over 100 images per class, our dataset was still insufficient to train advanced models such as transformers.

Future Steps

1. ***Multi-Object Recognition***

- Recognize multiple classes within a single image

2. ***Real Time Inference***

- Capture video frames using OpenCV and feed them into an optimized, quantized model on edge hardware for live inference

3. ***Collect more data***

- Increase the diversity and quantity of the dataset

4. ***Try other transformer-based models***

- Such as Vision Transformer (ViT) and Data-efficient Image Transformer (DeiT)

Thanks!

Q & A