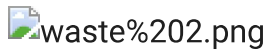


Waste Sorting Using Computer Vision using MobileNet



1. Problem Statement

Imagine if every piece of trash in the world could automatically find the right bin to go to! That's the magic of computer vision with waste sorting. Sorting waste is essential for keeping our planet clean. When we sort waste correctly:

Recyclables (like plastic bottles and cans) can be reused, which saves resources.

Organic waste can be composted, helping plants grow.

Non-recyclables go to landfills, but it's better if we minimize them.

By training a computer to recognize waste, we can speed up sorting and reduce mistakes, helping the environment and recycling process.

[+ Code](#)[+ Text](#)

2. Objectives

In this project, we will:

Use computer vision to help identify types of waste.

Train a model called MobileNet to "see" and categorize images as Organic (O) or Recyclable (R).

Use a dataset of waste images, practice sorting them, and observe how the computer learns to differen

3. Dataset Information

Our dataset contains images of waste items that belong to two categories:

O (Organic): For food scraps, leaves, or other natural waste.

R (Recyclable): For plastics, metals, and items that can be reused.

This dataset is organized into two folders:

Train Folder: Where images are used to teach the computer what each type of waste looks like.

Test Folder: To check if the computer has learned to sort images correctly.

Download Dataset Manually from Kaggle Website

Go to the Kaggle website, navigate to the dataset you want (e.g., the "Waste Classification Data").

URL for dataset used in this project : <https://www.kaggle.com/datasets/techsash/waste-classification->

Click the URL and click Download to manually download the dataset as a ZIP file.

Once downloaded, unzip the file and place it in your working directory.

4. Understanding MobileNet

Think of MobileNet as a quick learner that's lightweight and fast. MobileNet is designed for mobile devices, making it perfect for applications where we don't have a lot of computing power, like on phones or tablets. It can recognize objects by extracting features from images and comparing them to what it has learned.

Lightweight: Works on mobile devices and runs fast.

Good Accuracy: Efficiently identifies objects, like if an item is recyclable or organic.

Feature Extraction: Takes essential details from images (like shapes and colors).

✓ 5. Code Implementation

Let's walk through the code to see how we train MobileNet to sort waste images.

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img, img_to_array
from sklearn.metrics import classification_report, confusion_matrix

# Paths to directories and model
base_dir = r"C:\Users\user\Desktop\Edunet_Content_Usha\Computer Vision\Datasets\Waste Classi
train_dir = base_dir + r"\TRAIN"
test_dir = base_dir + r"\TEST"
model_path = 'mobilenet_waste_classifier.h5'
```

```
test_image_path = r"C:\Users\user\Desktop\Edunet_Content_Usha\Computer Vision\Datasets\Waste

# Data preparation
train_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)
train_data = train_datagen.flow_from_directory(train_dir, target_size=(224, 224), batch_size=
test_data = test_datagen.flow_from_directory(test_dir, target_size=(224, 224), batch_size=32

# Load and configure MobileNetV2
mobilenet_model = tf.keras.applications.MobileNetV2(input_shape=(224, 224, 3), include_top=F
mobilenet_model.trainable = False

# Add custom layers for waste classification
model = tf.keras.Sequential([
    mobilenet_model,
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Train the model
model.fit(train_data, validation_data=test_data, epochs=2)
model.save(model_path)

# Evaluate the model
loss, accuracy = model.evaluate(test_data)
print("Model Accuracy:", accuracy)

# Generate classification report and confusion matrix
y_pred_prob = model.predict(test_data).flatten()
y_pred = (y_pred_prob > 0.5).astype(int)
y_true = test_data.classes
print("Classification Report:\n", classification_report(y_true, y_pred, target_names=["Organic", "Recyclable"])
print("Confusion Matrix:\n", confusion_matrix(y_true, y_pred))

# Load and prepare a single test image
model = tf.keras.models.load_model(model_path)
test_img = load_img(test_image_path, target_size=(224, 224))
test_img_array = img_to_array(test_img) / 255.0
test_img_array = np.expand_dims(test_img_array, axis=0)

# Predict the class of the test image
predicted_prob = model.predict(test_img_array)[0][0]
predicted_class = 'Organic' if predicted_prob < 0.5 else 'Recyclable'
plt.imshow(test_img)
plt.title(f"Predicted: {predicted_class}")
plt.axis('off')
plt.show()
```

```

Found 22564 images belonging to 2 classes.
Found 2514 images belonging to 2 classes.
Epoch 1/2
706/706 [=====] - 583s 823ms/step - loss: 0.2373 - accuracy: 0.
Epoch 2/2
706/706 [=====] - 672s 952ms/step - loss: 0.1710 - accuracy: 0.
79/79 [=====] - 67s 849ms/step - loss: 0.2219 - accuracy: 0.911
Model Accuracy: 0.9116945266723633
79/79 [=====] - 73s 909ms/step
Classification Report:

```

	precision	recall	f1-score	support
Organic	0.56	0.60	0.58	1401
Recyclable	0.44	0.40	0.42	1113
accuracy			0.51	2514
macro avg	0.50	0.50	0.50	2514
weighted avg	0.51	0.51	0.51	2514

```

Confusion Matrix:
[[842 559]
 [669 444]]
1/1 [=====] - 1s 892ms/step

```


Predicted: Recyclable



✓ 6. Conclusion

By learning to identify waste, MobileNet becomes a helpful tool in sorting. This project highlights how computer vision can make real-life processes like waste sorting easier and faster. Using this

setup, we can eventually create applications for recycling plants or even home use, contributing to a cleaner and more sustainable world.

waste%201.png