Garbage Classification

Nandaj Jayaraj
nandajofc@gmail.com
+91 7306603029

GitHub Link for this project: https://github.com/nandajjay/Garbage-Classification

Aim: To train a model to classify waste into categories.

Inference: The model trained can identify objects and classify them into one of the baskets which we defined.

(Basket= Battery, Biological, Brown glass, Carboard, Clothes, Green Glass, Metal, paper, Plastic, Shoes, Trash and White glass.)

- We were able to obtain a precision rating above 90(0.9) for most of the objects.

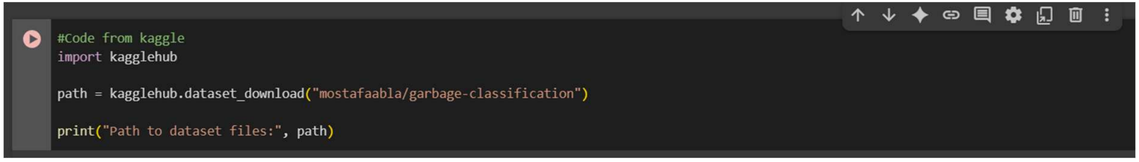
Dataset Link: https://www.kaggle.com/datasets/mostafaabla/garbage-classification

Steps:

1. Data pre-processing
2. Data analysis (EDA).
3. Model Training.

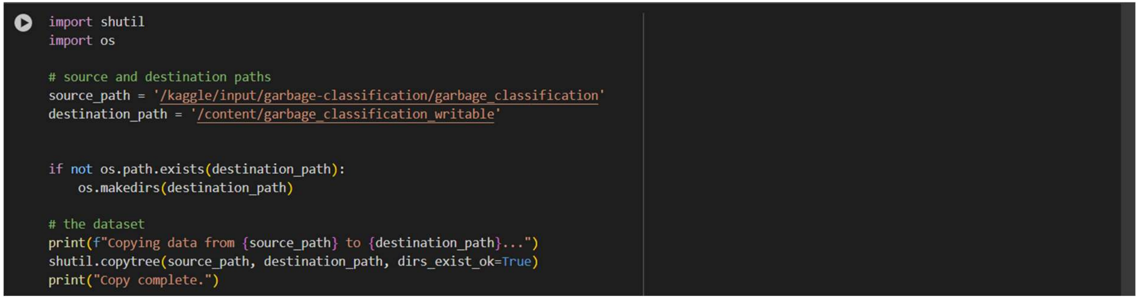
Code:

1. Importing Data



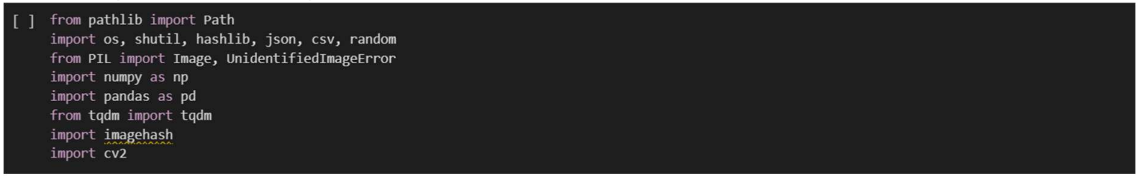
```
#Code from kaggle
import kagglehub

path = kagglehub.dataset_download("mostafaabla/garbage-classification")

print("Path to dataset files:", path)
```

2. Copying data to make it writable.

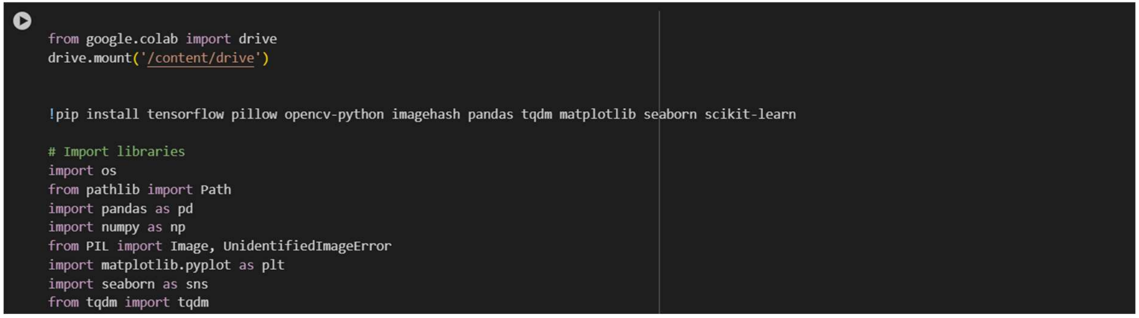


```
import shutil
import os

# source and destination paths
source_path = '/kaggle/input/garbage-classification/garbage_classification'
destination_path = '/content/garbage_classification_writable'


if not os.path.exists(destination_path):
    os.makedirs(destination_path)

# the dataset
print(f"Copying data from {source_path} to {destination_path}...")
shutil.copytree(source_path, destination_path, dirs_exist_ok=True)
print("Copy complete.")
```

3. Installing and importing required libraries:



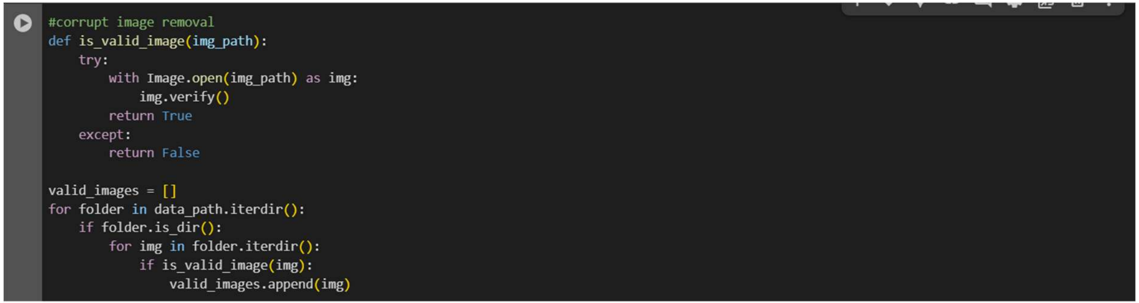
```
from pathlib import Path
import os, shutil, hashlib, json, csv, random
from PIL import Image, UnidentifiedImageError
import numpy as np
import pandas as pd
from tqdm import tqdm
import imagehash
import cv2
```



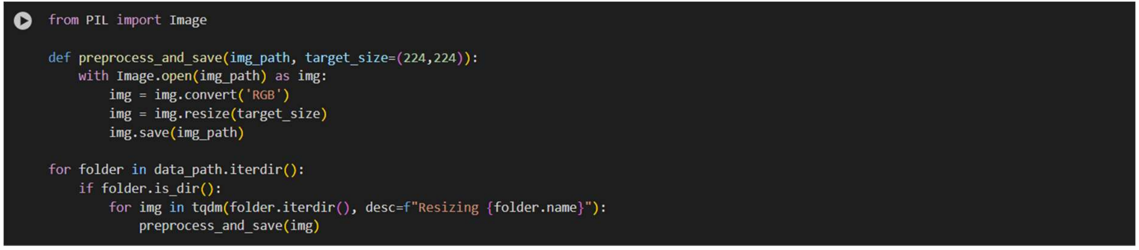
```
from google.colab import drive
drive.mount('/content/drive')


!pip install tensorflow pillow opencv-python imagehash pandas tqdm matplotlib seaborn scikit-learn

# Import libraries
import os
from pathlib import Path
import pandas as pd
import numpy as np
from PIL import Image, UnidentifiedImageError
import matplotlib.pyplot as plt
import seaborn as sns
from tqdm import tqdm
```

4. Cleaning Data:

- Corrupt data removal.
- Resizing images.

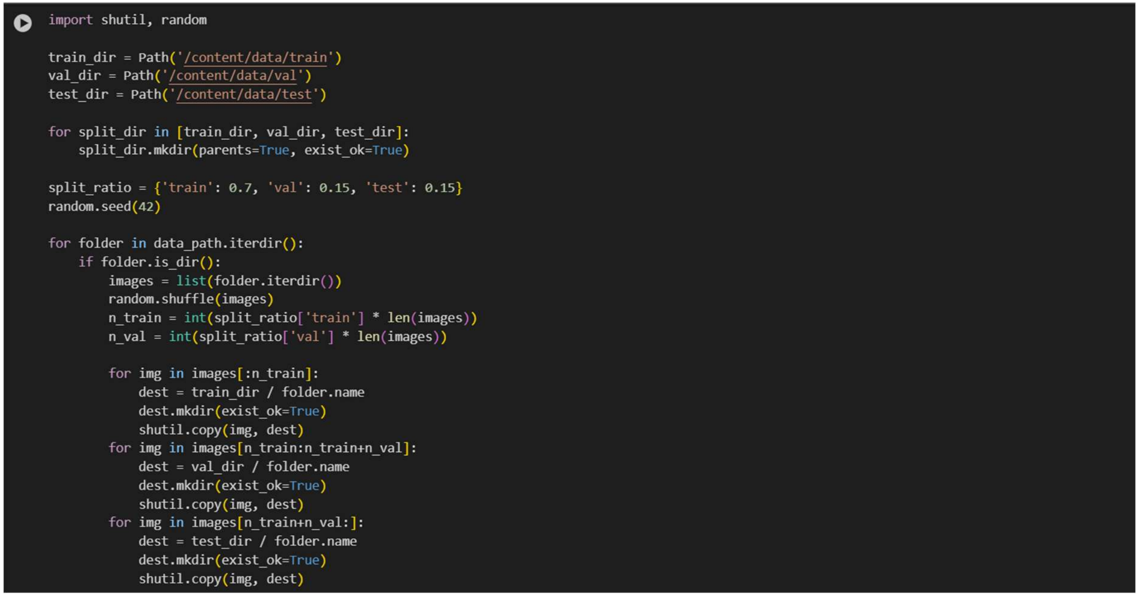


```
#corrupt image removal
def is_valid_image(img_path):
    try:
        with Image.open(img_path) as img:
            img.verify()
        return True
    except:
        return False

valid_images = []
for folder in data_path.iterdir():
    if folder.is_dir():
        for img in folder.iterdir():
            if is_valid_image(img):
                valid_images.append(img)
```



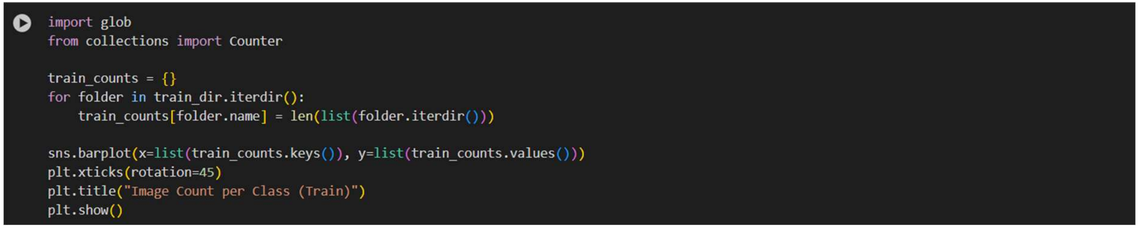
```
from PIL import Image

def preprocess_and_save(img_path, target_size=(224,224)):
    with Image.open(img_path) as img:
        img = img.convert('RGB')
        img = img.resize(target_size)
        img.save(img_path)

for folder in data_path.iterdir():
    if folder.is_dir():
        for img in tqdm(folder.iterdir(), desc=f"Resizing {folder.name}"):
            preprocess_and_save(img)
```

5. Splitting Data for Training, Validation and Testing:

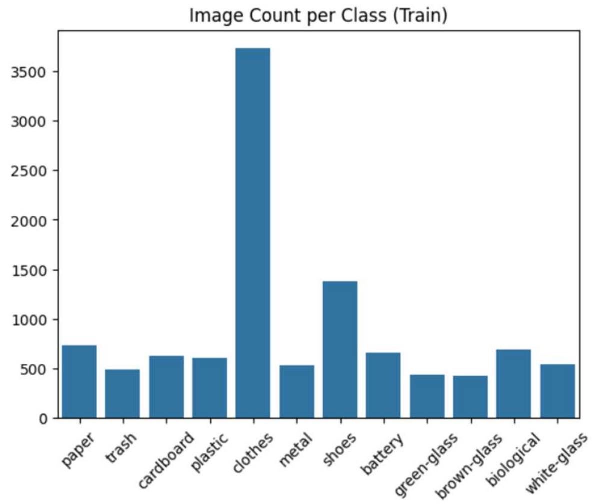


```
import shutil, random

train_dir = Path('/content/data/train')
val_dir = Path('/content/data/val')
test_dir = Path('/content/data/test')

for split_dir in [train_dir, val_dir, test_dir]:
    split_dir.mkdir(parents=True, exist_ok=True)

split_ratio = {'train': 0.7, 'val': 0.15, 'test': 0.15}
random.seed(42)

for folder in data_path.iterdir():
    if folder.is_dir():
        images = list(folder.iterdir())
        random.shuffle(images)
        n_train = int(split_ratio['train'] * len(images))
        n_val = int(split_ratio['val'] * len(images))

        for img in images[:n_train]:
            dest = train_dir / folder.name
            dest.mkdir(exist_ok=True)
            shutil.copy(img, dest)
        for img in images[n_train:n_train+n_val]:
            dest = val_dir / folder.name
            dest.mkdir(exist_ok=True)
            shutil.copy(img, dest)
        for img in images[n_train+n_val:]:
            dest = test_dir / folder.name
            dest.mkdir(exist_ok=True)
            shutil.copy(img, dest)
```

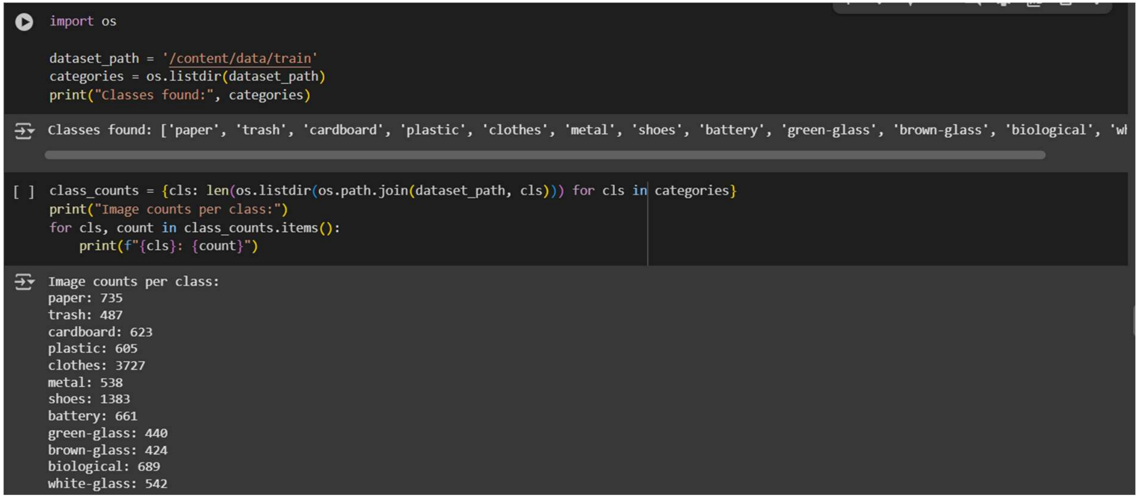
6. EDA (Analysing Data):



```
import glob
from collections import Counter

train_counts = {}
for folder in train_dir.iterdir():
    train_counts[folder.name] = len(list(folder.iterdir()))

sns.barplot(x=list(train_counts.keys()), y=list(train_counts.values()))
plt.xticks(rotation=45)
plt.title("Image Count per Class (Train)")
plt.show()
```



7. Calculating count per class:



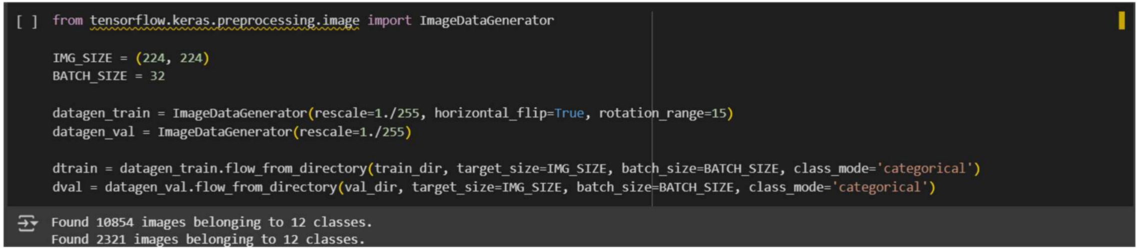
```
import os

dataset_path = '/content/data/train'
categories = os.listdir(dataset_path)
print("Classes found:", categories)
```

```
Classes found: ['paper', 'trash', 'cardboard', 'plastic', 'clothes', 'metal', 'shoes', 'battery', 'green-glass', 'brown-glass', 'biological', 'w
```

```
class_counts = {cls: len(os.listdir(os.path.join(dataset_path, cls))) for cls in categories}
print("Image counts per class:")
for cls, count in class_counts.items():
    print(f"{cls}: {count}")
```

```
Image counts per class:
paper: 735
trash: 487
cardboard: 623
plastic: 605
clothes: 3727
metal: 538
shoes: 1383
battery: 661
green-glass: 440
brown-glass: 424
biological: 689
white-glass: 542
```

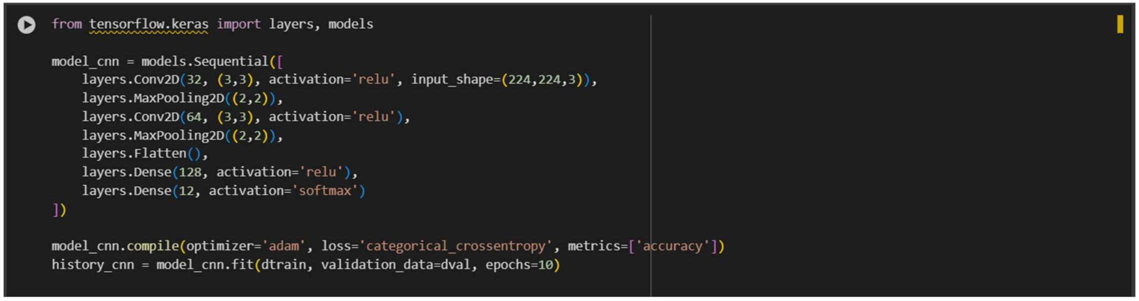
8. Training CNN using Keras:



```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

IMG_SIZE = (224, 224)
BATCH_SIZE = 32

datagen_train = ImageDataGenerator(rescale=1./255, horizontal_flip=True, rotation_range=15)
datagen_val = ImageDataGenerator(rescale=1./255)

dtrain = datagen_train.flow_from_directory(train_dir, target_size=IMG_SIZE, batch_size=BATCH_SIZE, class_mode='categorical')
dval = datagen_val.flow_from_directory(val_dir, target_size=IMG_SIZE, batch_size=BATCH_SIZE, class_mode='categorical')
```

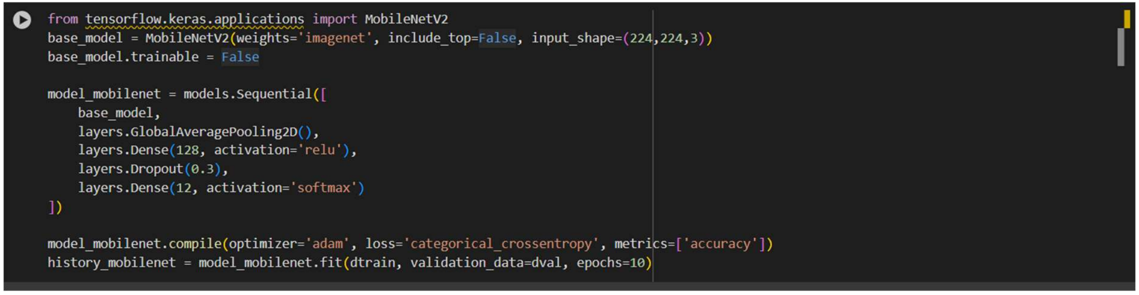
```
Found 10854 images belonging to 12 classes.
Found 2321 images belonging to 12 classes.
```

9. Setting up data pipeline for training and validation:



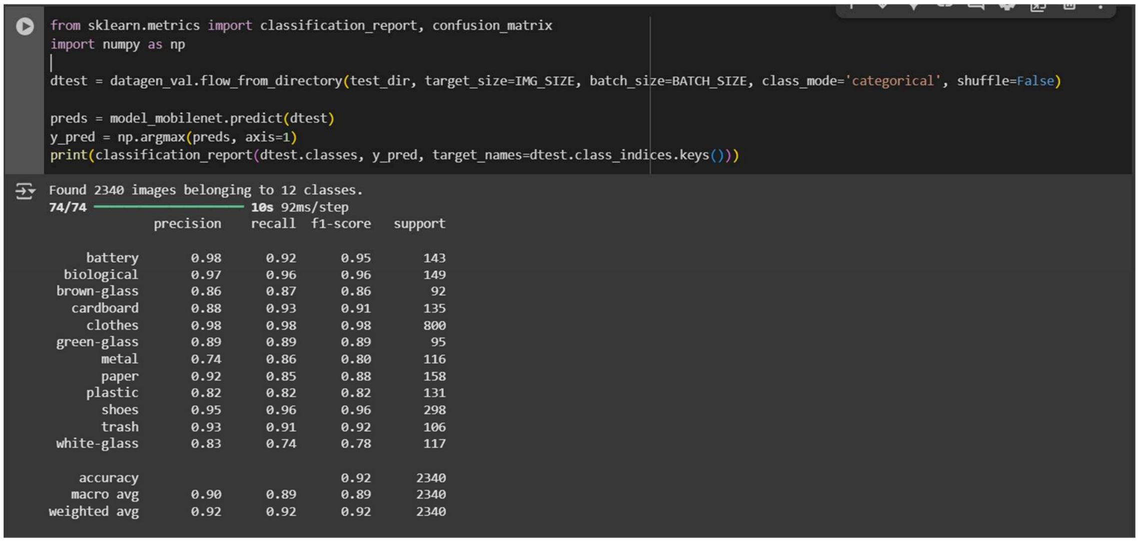
```
from tensorflow.keras import layers, models

model_cnn = models.Sequential([
    layers.Conv2D(32, (3,3), activation='relu', input_shape=(224,224,3)),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64, (3,3), activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(12, activation='softmax')
])

model_cnn.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
history_cnn = model_cnn.fit(dtrain, validation_data=dval, epochs=10)
```

10. Training based on MobileNetV2:



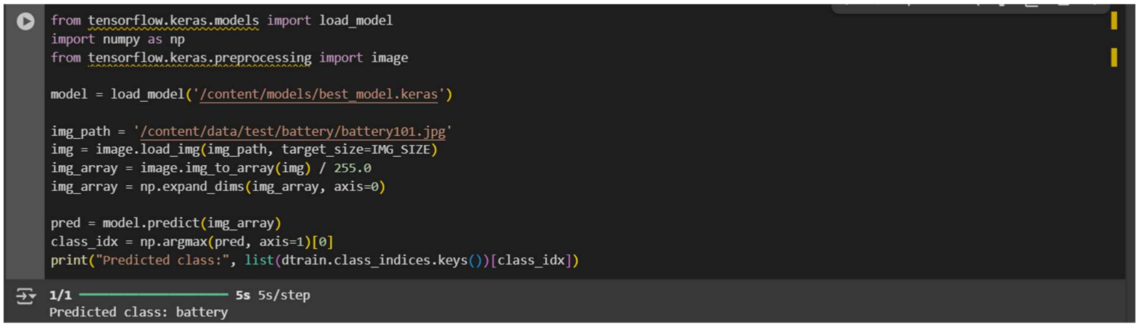
```
from tensorflow.keras.applications import MobileNetV2
base_model = MobileNetV2(weights='imagenet', include_top=False, input_shape=(224,224,3))
base_model.trainable = False

model_mobilenet = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.3),
    layers.Dense(12, activation='softmax')
])

model_mobilenet.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
history_mobilenet = model_mobilenet.fit(dtrain, validation_data=dval, epochs=10)
```

11. Evaluation of trained v2 model:



```
from sklearn.metrics import classification_report, confusion_matrix
import numpy as np

dtest = datagen_val.flow_from_directory(test_dir, target_size=IMG_SIZE, batch_size=BATCH_SIZE, class_mode='categorical', shuffle=False)

preds = model_mobilenet.predict(dtest)
y_pred = np.argmax(preds, axis=1)
print(classification_report(dtest.classes, y_pred, target_names=dtest.class_indices.keys()))
```

```
Found 2340 images belonging to 12 classes.
74/74 ━━━━━━━━━━━━━━━━━━━━ 10s 92ms/step
              precision    recall  f1-score   support

     battery       0.98      0.92      0.95       143
  biological       0.97      0.96      0.96       149
 brown-glass       0.86      0.87      0.86        92
   cardboard       0.88      0.93      0.91       135
     clothes       0.98      0.98      0.98       800
 green-glass       0.89      0.89      0.89        95
       metal       0.74      0.86      0.80       116
       paper       0.92      0.85      0.88       158
     plastic       0.82      0.82      0.82       131
       shoes       0.95      0.96      0.96       298
       trash       0.93      0.91      0.92       106
 white-glass       0.83      0.74      0.78       117

    accuracy                           0.92      2340
   macro avg       0.90      0.89      0.89      2340
weighted avg       0.92      0.92      0.92      2340
```



```
from tensorflow.keras.models import load_model
import numpy as np
from tensorflow.keras.preprocessing import image

model = load_model('/content/models/best_model.keras')

img_path = '/content/data/test/battery/battery101.jpg'
img = image.load_img(img_path, target_size=IMG_SIZE)
img_array = image.img_to_array(img) / 255.0
img_array = np.expand_dims(img_array, axis=0)

pred = model.predict(img_array)
class_idx = np.argmax(pred, axis=1)[0]
print("Predicted class:", list(dtrain.class_indices.keys())[class_idx])
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 5s 5s/step
Predicted class: battery
```

Note: This report is made for evaluation purpose only.