```
import os
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import EfficientNetB0
from tensorflow.keras import layers, models
from tensorflow.keras.callbacks import EarlyStopping
import matplotlib.pyplot as plt
```

```
#  Mount Google Drive
from google.colab import drive
import os

drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
# Set path to the dataset
dataset_path = "/content/drive/MyDrive/dataset-resized"
```

```

#  Image Settings
IMG_SIZE = (224, 224)
BATCH_SIZE = 32
EPOCHS = 10

# Data Augmentation
datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2,
    horizontal_flip=True,
    zoom_range=0.2,
    rotation_range=30
)

train_gen = datagen.flow_from_directory(
    dataset_path,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    subset='training'
)

val_gen = datagen.flow_from_directory(
    dataset_path,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    subset='validation'
)
```

```
Found 2024 images belonging to 6 classes.
Found 503 images belonging to 6 classes.
```

```
#  Build the CNN Model using EfficientNetB0
base_model = EfficientNetB0(include_top=False, weights='imagenet', input_shape=(*IMG_SIZE, 3))
base_model.trainable = False
model = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.3),
    layers.Dense(train_gen.num_classes, activation='softmax')
])

```

```
Downloading data from https://storage.googleapis.com/keras-applications/efficientnetb0_notop.h5
16705208/16705208 ━━━━━━━━━━━━━━━━━━━━ 2s 0us/step
```

```
# Compile the Model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Train the Model
history = model.fit(
    train_gen,
    validation_data=val_gen,
    epochs=EPOCHS,
    callbacks=[EarlyStopping(monitor='val_loss', patience=3, restore_best_weights=True)]
```

```
)
```

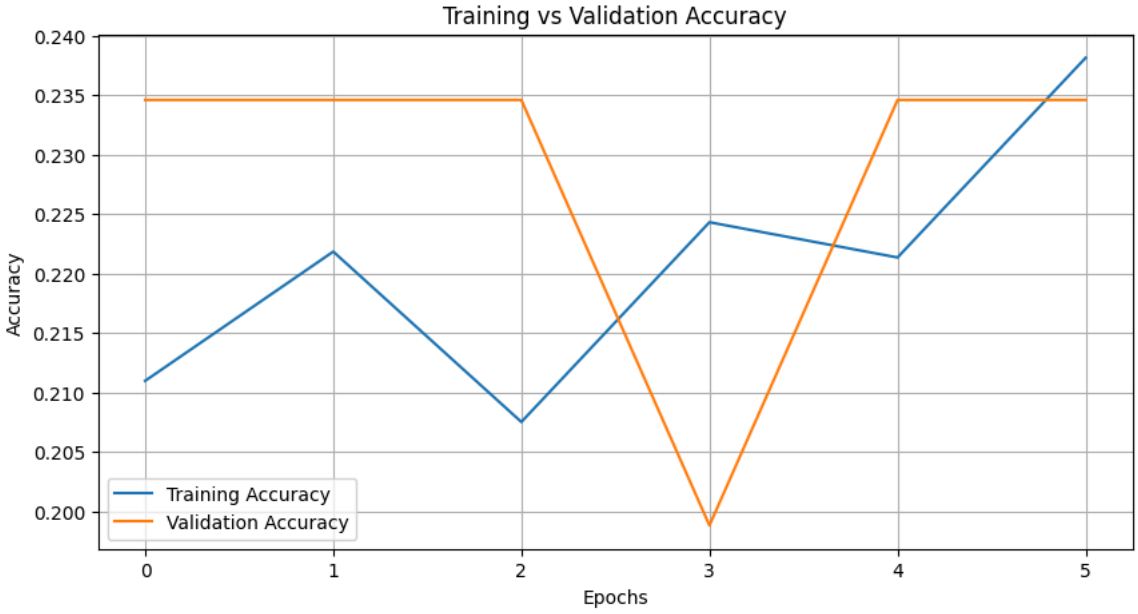
```
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `P
  self._warn_if_super_not_called()
Epoch 1/10
64/64 ━━━━━━━━━━━━━━━━━━━━ 1420s 22s/step - accuracy: 0.2058 - loss: 1.7719 - val_accuracy: 0.2346 - val_loss: 1.7236
Epoch 2/10
64/64 ━━━━━━━━━━━━━━━━━━━━ 37s 586ms/step - accuracy: 0.2196 - loss: 1.7434 - val_accuracy: 0.2346 - val_loss: 1.7242
Epoch 3/10
64/64 ━━━━━━━━━━━━━━━━━━━━ 41s 587ms/step - accuracy: 0.1788 - loss: 1.7465 - val_accuracy: 0.2346 - val_loss: 1.7236
Epoch 4/10
64/64 ━━━━━━━━━━━━━━━━━━━━ 38s 588ms/step - accuracy: 0.2239 - loss: 1.7393 - val_accuracy: 0.1988 - val_loss: 1.7251
Epoch 5/10
64/64 ━━━━━━━━━━━━━━━━━━━━ 38s 598ms/step - accuracy: 0.2106 - loss: 1.7314 - val_accuracy: 0.2346 - val_loss: 1.7249
Epoch 6/10
64/64 ━━━━━━━━━━━━━━━━━━━━ 38s 598ms/step - accuracy: 0.2521 - loss: 1.7266 - val_accuracy: 0.2346 - val_loss: 1.7249
```

```
#  Evaluate the Model
val_loss, val_acc = model.evaluate(val_gen)
print(f"\n Final Validation Accuracy: {val_acc:.4f}")

# Plot Training History
plt.figure(figsize=(10, 5))
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title("Training vs Validation Accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.grid(True)
plt.show()
```

```
16/16 ━━━━━━━━━━━━━━━━━━━━ 7s 432ms/step - accuracy: 0.2562 - loss: 1.7384

 Final Validation Accuracy: 0.2346
```




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

```
# Predict classes for validation set
val_gen.reset()
y_pred = model.predict(val_gen, verbose=1)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true = val_gen.classes
```

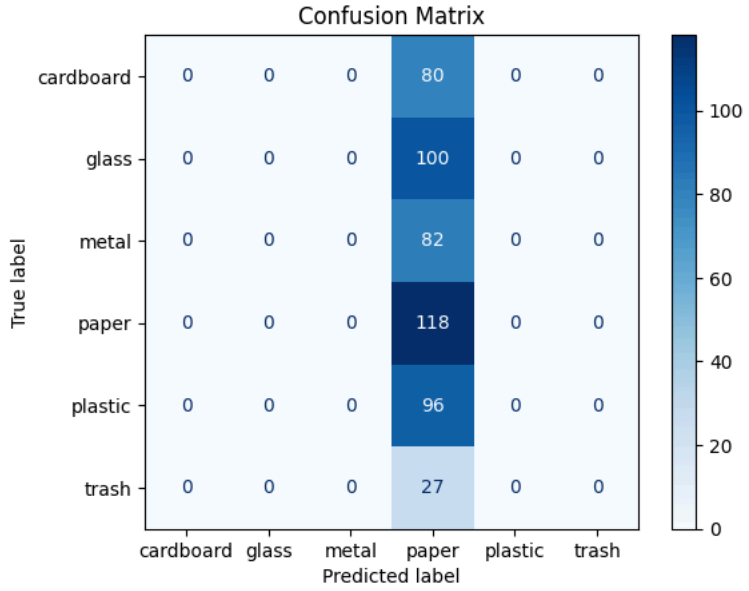
```
16/16 ━━━━━━━━━━━━━━━━━━━━ 19s 864ms/step
```

```
#  Confusion Matrix
cm = confusion_matrix(y_true, y_pred_classes)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=list(val_gen.
class_indices.keys()))

plt.figure(figsize=(8, 6))
disp.plot(cmap=plt.cm.Blues)
```

```
disp.plot(cmap=plt.cm.Blues)
plt.title('Confusion Matrix')
plt.grid(False)
plt.show()
```

```
<Figure size 800x600 with 0 Axes>
```




```
#  Classification Report
report = classification_report(y_true, y_pred_classes, target_names=list
(val_gen.class_indices.keys()))
print("\n🔎 Classification Report:\n")
print(report)
```

```
🔎 Classification Report:

              precision    recall  f1-score   support

   cardboard       0.00      0.00      0.00        80
       glass       0.00      0.00      0.00       100
       metal       0.00      0.00      0.00        82
       paper       0.23      1.00      0.38       118
     plastic       0.00      0.00      0.00        96
       trash       0.00      0.00      0.00        27

    accuracy                           0.23       503
   macro avg       0.04      0.17      0.06       503
weighted avg       0.06      0.23      0.09       503

/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```