LAB MANUAL

Unit V – Deep Learning

Disclaimer: The content is curated from online/offline resources and used for educational purpose only

Unit V – Deep Learning

Lab 3. To develop a CNN model to classify images of Organic or Recyclable

Objective

- Collect and preprocess a dataset of labeled images categorized as "Organic" and "Recyclable."
- Design a CNN architecture tailored to extract features from waste images, including spatial patterns and textures.
- Train the CNN model on the preprocessed dataset using a training-validation split.
- Evaluate the effectiveness of the model in reducing manual waste segregation efforts.
- Demonstrate how the model can support recycling initiatives and improve environmental sustainability.

Problem

Develop a Convolutional Neural Network (CNN) model to classify images into two categories: 'Organic' and 'Recyclable.' The system will analyze and categorize waste images, aiding in efficient waste management and promoting sustainability practices.

Solution

we'll go through the following steps:

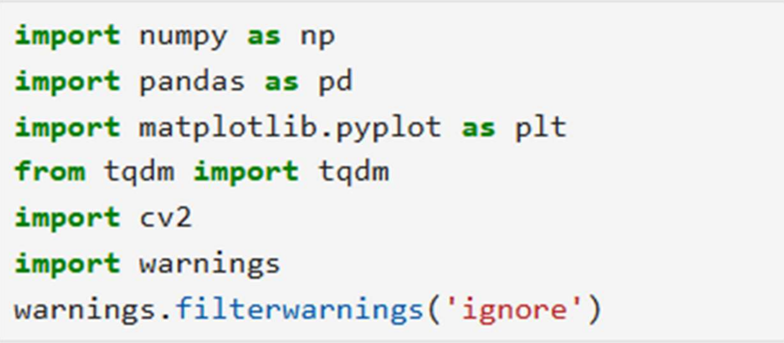
1. Import required libraries
2. Load Dataset
3. Collections Count
4. Show data
5. Scale the data for better performance
6. Find number of classes in Data
7. Build the CNN Model
    a. Initialize the model

   b. Convolution, Activation and Pooling layers
   c. Flatten, Dropout and output layers
   d. Compile the model

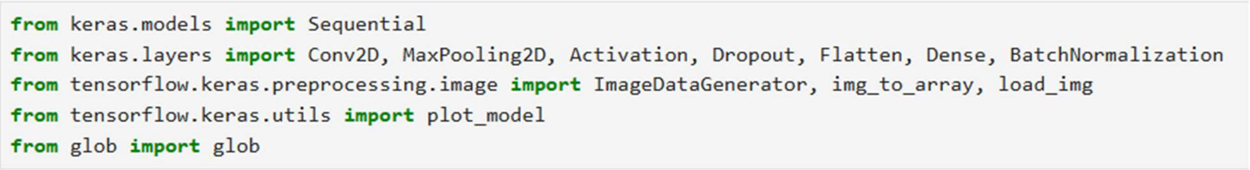
8. Model Summary
9. Normalization
10. Train and Test data generation
11. Train the Model
12. Plot Training Accuracy
13. Plot Loss
14. Model Prediction

Procedures

1. Import required libraries

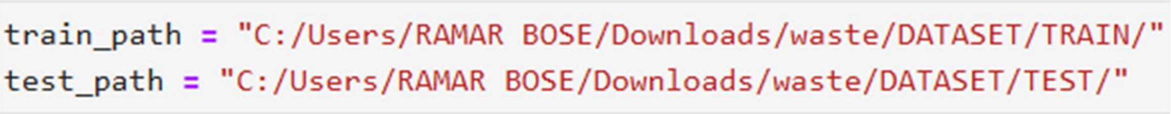


```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from tqdm import tqdm
import cv2
import warnings
warnings.filterwarnings('ignore')
```

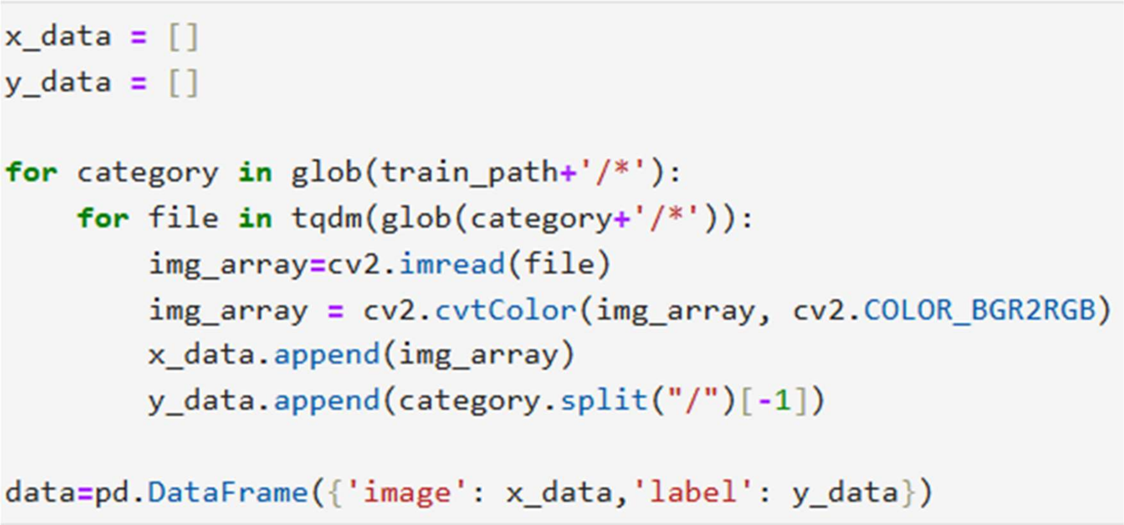


```python
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Activation, Dropout, Flatten, Dense, BatchNormalization
from tensorflow.keras.preprocessing.image import ImageDataGenerator, img_to_array, load_img
from tensorflow.keras.utils import plot_model
from glob import glob
```

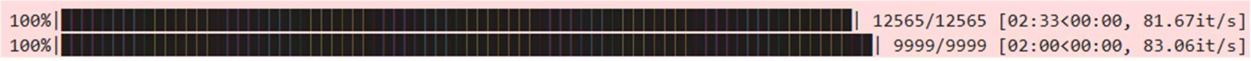
2. Load Dataset



```python
train_path = "C:/Users/RAMAR BOSE/Downloads/waste/DATASET/TRAIN/"
test_path = "C:/Users/RAMAR BOSE/Downloads/waste/DATASET/TEST/"
```

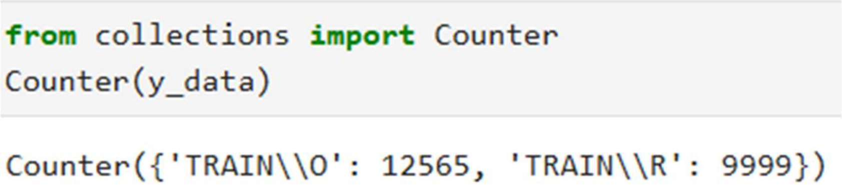


```
x_data = []
y_data = []

for category in glob(train_path+'/*'):
    for file in tqdm(glob(category+'/*')):
        img_array=cv2.imread(file)
        img_array = cv2.cvtColor(img_array, cv2.COLOR_BGR2RGB)
        x_data.append(img_array)
        y_data.append(category.split("/")[-1])

data=pd.DataFrame({'image': x_data,'label': y_data})
```



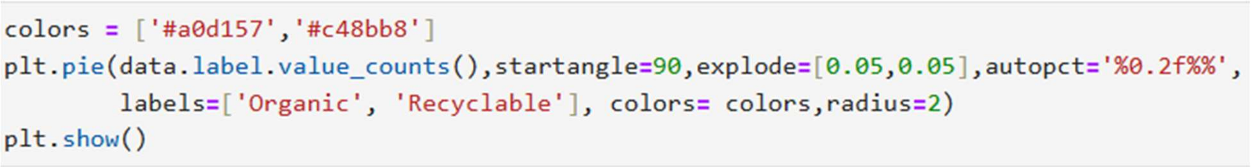
```
100%|██████████████████████████████████████████████████████████████████████| 12565/12565 [02:33<00:00, 81.67it/s]
100%|████████████████████████████████████████████████████████████████████████| 9999/9999 [02:00<00:00, 83.06it/s]
```

3. Collections Count

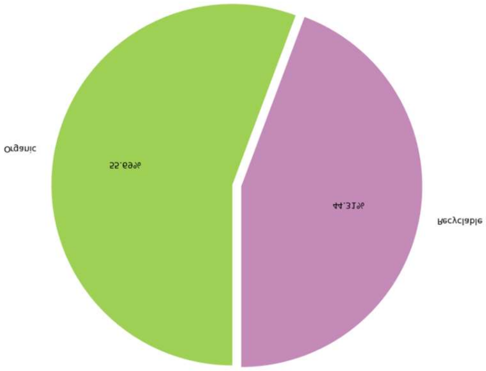


```
from collections import Counter
Counter(y_data)
```

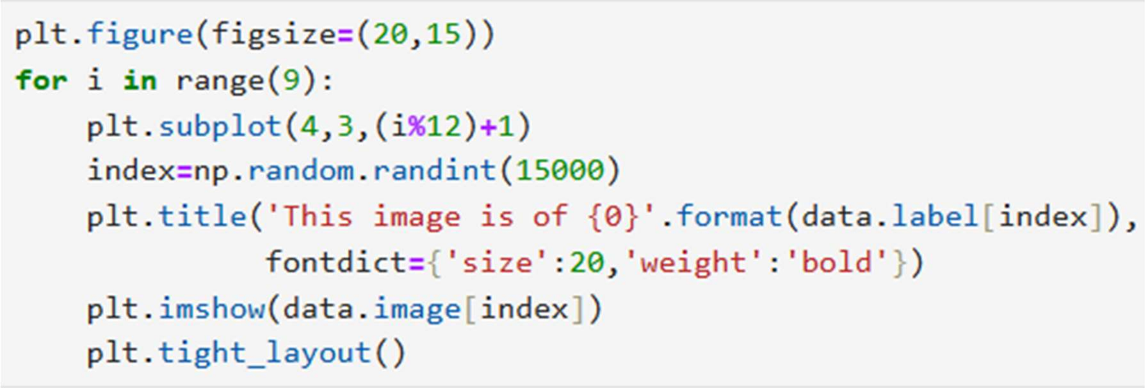
```
Counter({'TRAIN\\O': 12565, 'TRAIN\\R': 9999})
```



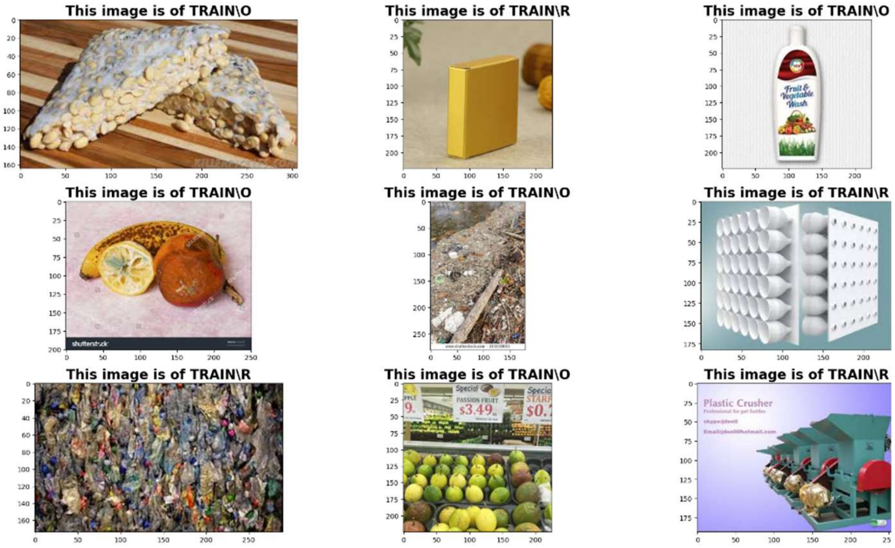
```
colors = ['#a0d157','#c48bb8']
plt.pie(data.label.value_counts(),startangle=90,explode=[0.05,0.05],autopct='%0.2f%%',
        labels=['Organic', 'Recyclable'], colors= colors,radius=2)
plt.show()
```



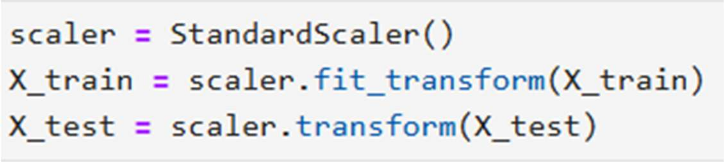
4. Show data



```
plt.figure(figsize=(20,15))
for i in range(9):
    plt.subplot(4,3,(i%12)+1)
    index=np.random.randint(15000)
    plt.title('This image is of {0}'.format(data.label[index]),
              fontdict={'size':20,'weight':'bold'})
    plt.imshow(data.image[index])
    plt.tight_layout()
```

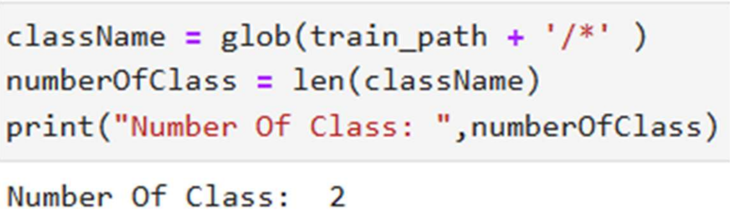


5. Scale the data for better performance



```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

6. Find number of classes in Data

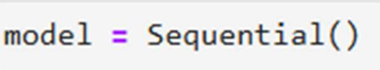


```
className = glob(train_path + '/*' )
numberOfClass = len(className)
print("Number Of Class: ",numberOfClass)
```

```
Number Of Class:  2
```

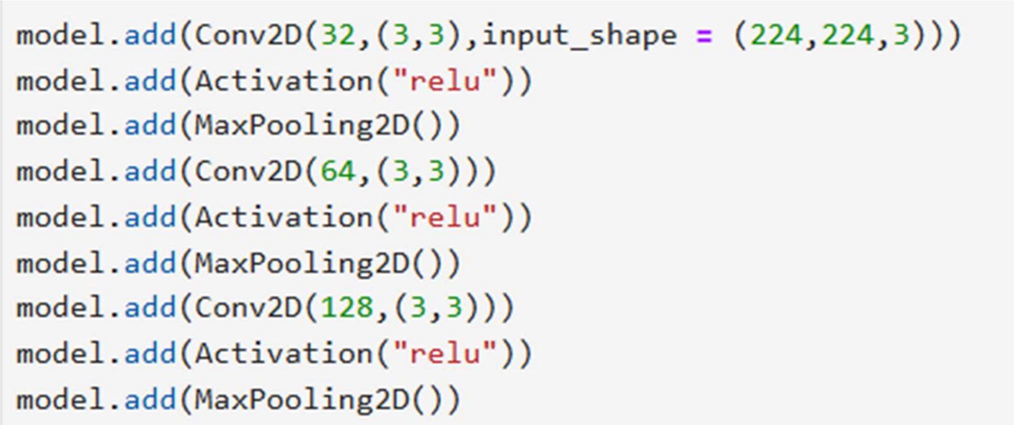
7. Build the CNN Model

    a. Initialize the model



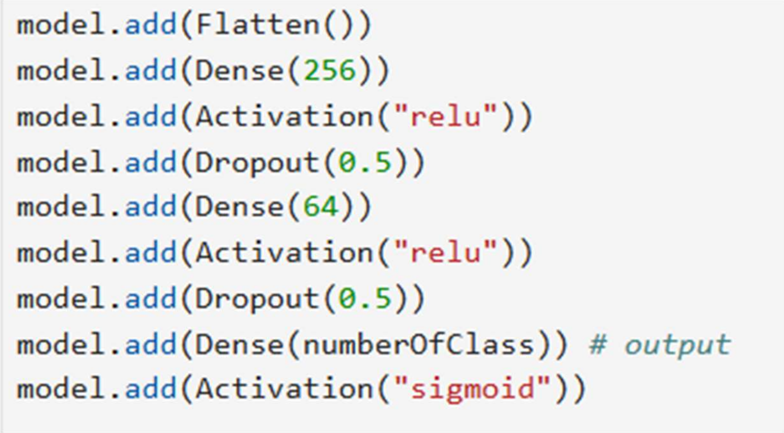
```
model = Sequential()
```

    b. Convolution, Activation and Pooling layers



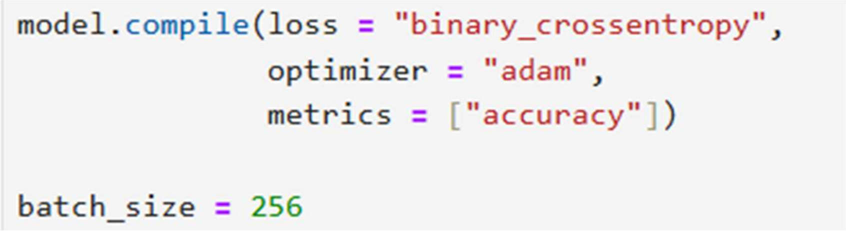
```
model.add(Conv2D(32,(3,3),input_shape = (224,224,3)))
model.add(Activation("relu"))
model.add(MaxPooling2D())
model.add(Conv2D(64,(3,3)))
model.add(Activation("relu"))
model.add(MaxPooling2D())
model.add(Conv2D(128,(3,3)))
model.add(Activation("relu"))
model.add(MaxPooling2D())
```

    c. Flatten, Dropout and output layers



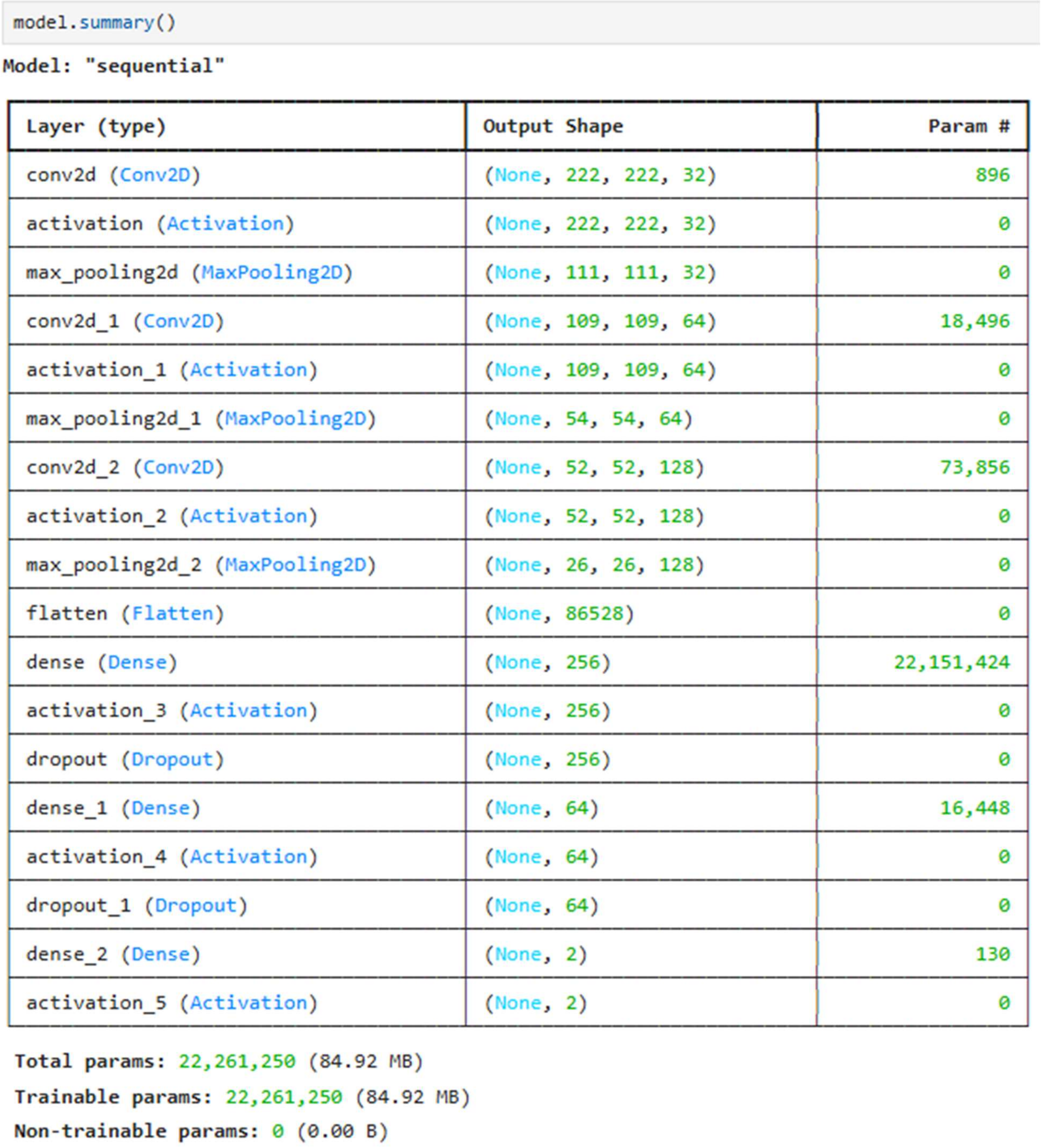
```
model.add(Flatten())
model.add(Dense(256))
model.add(Activation("relu"))
model.add(Dropout(0.5))
model.add(Dense(64))
model.add(Activation("relu"))
model.add(Dropout(0.5))
model.add(Dense(numberOfClass)) # output
model.add(Activation("sigmoid"))
```

d. Compile the model



```
model.compile(loss = "binary_crossentropy",
              optimizer = "adam",
              metrics = ["accuracy"])

batch_size = 256
```

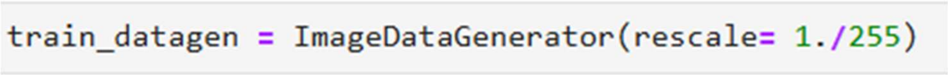
8. Model Summary



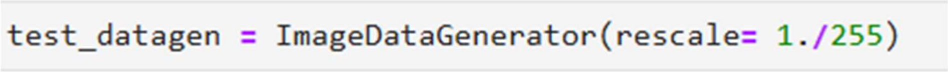
```
model.summary()
```

```
Model: "sequential"

┌─────────────────────────────────┬────────────────────────┬───────────────┐
│ Layer (type)                    │ Output Shape           │       Param # │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d (Conv2D)                 │ (None, 222, 222, 32)   │           896 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ activation (Activation)         │ (None, 222, 222, 32)   │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d (MaxPooling2D)    │ (None, 111, 111, 32)   │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_1 (Conv2D)               │ (None, 109, 109, 64)   │        18,496 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ activation_1 (Activation)       │ (None, 109, 109, 64)   │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d_1 (MaxPooling2D)  │ (None, 54, 54, 64)     │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_2 (Conv2D)               │ (None, 52, 52, 128)    │        73,856 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ activation_2 (Activation)       │ (None, 52, 52, 128)    │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d_2 (MaxPooling2D)  │ (None, 26, 26, 128)    │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ flatten (Flatten)               │ (None, 86528)          │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense (Dense)                   │ (None, 256)            │    22,151,424 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ activation_3 (Activation)       │ (None, 256)            │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout (Dropout)               │ (None, 256)            │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_1 (Dense)                 │ (None, 64)             │        16,448 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ activation_4 (Activation)       │ (None, 64)             │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout_1 (Dropout)             │ (None, 64)             │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_2 (Dense)                 │ (None, 2)              │           130 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ activation_5 (Activation)       │ (None, 2)              │             0 │
└─────────────────────────────────┴────────────────────────┴───────────────┘

 Total params: 22,261,250 (84.92 MB)

 Trainable params: 22,261,250 (84.92 MB)

 Non-trainable params: 0 (0.00 B)
```

9. Normalization

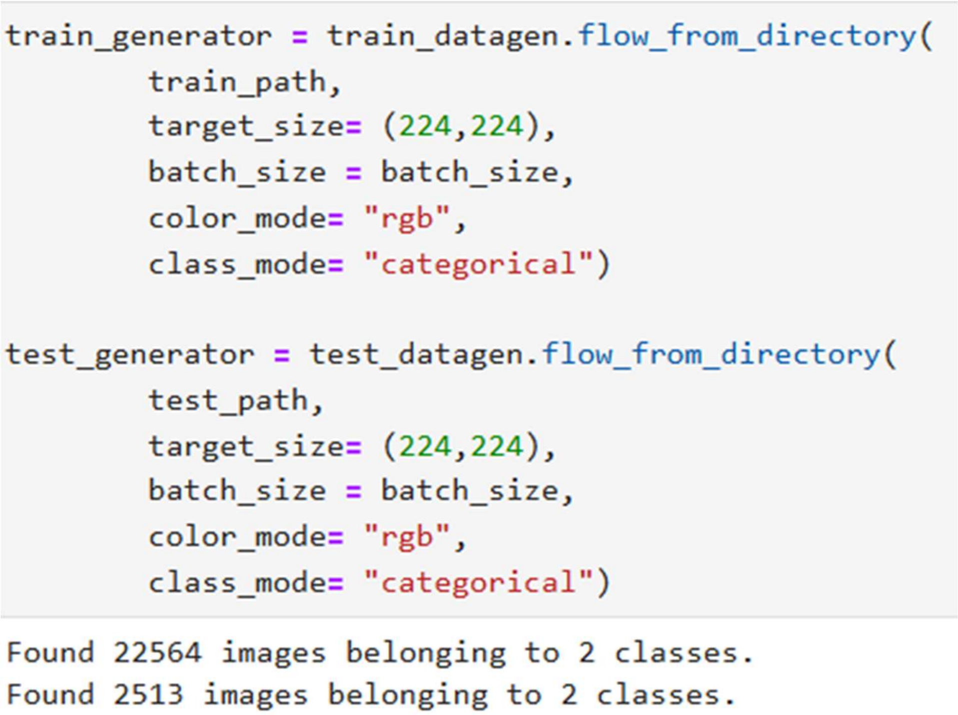


```
train_datagen = ImageDataGenerator(rescale= 1./255)
```



```
test_datagen = ImageDataGenerator(rescale= 1./255)
```

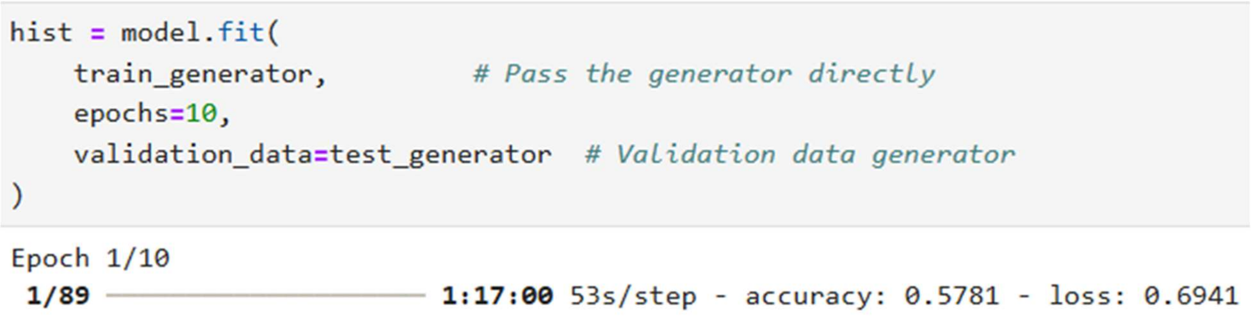
10. Train and Test data generation



```
train_generator = train_datagen.flow_from_directory(
        train_path,
        target_size= (224,224),
        batch_size = batch_size,
        color_mode= "rgb",
        class_mode= "categorical")

test_generator = test_datagen.flow_from_directory(
        test_path,
        target_size= (224,224),
        batch_size = batch_size,
        color_mode= "rgb",
        class_mode= "categorical")
```

```
Found 22564 images belonging to 2 classes.
Found 2513 images belonging to 2 classes.
```

11. Train the Model

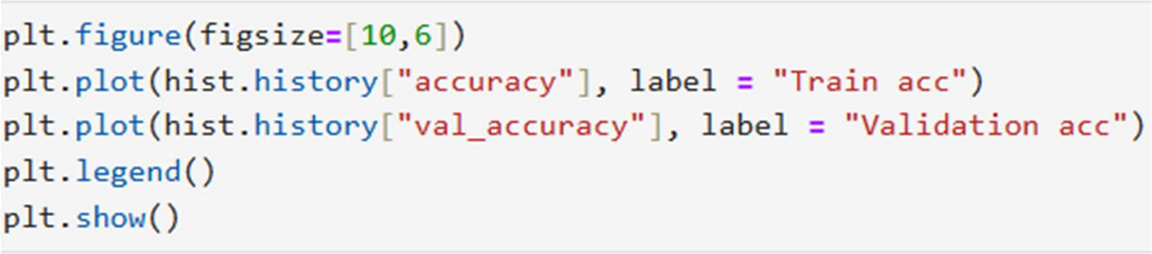


```
hist = model.fit(
    train_generator,         # Pass the generator directly
    epochs=10,
    validation_data=test_generator  # Validation data generator
)
```

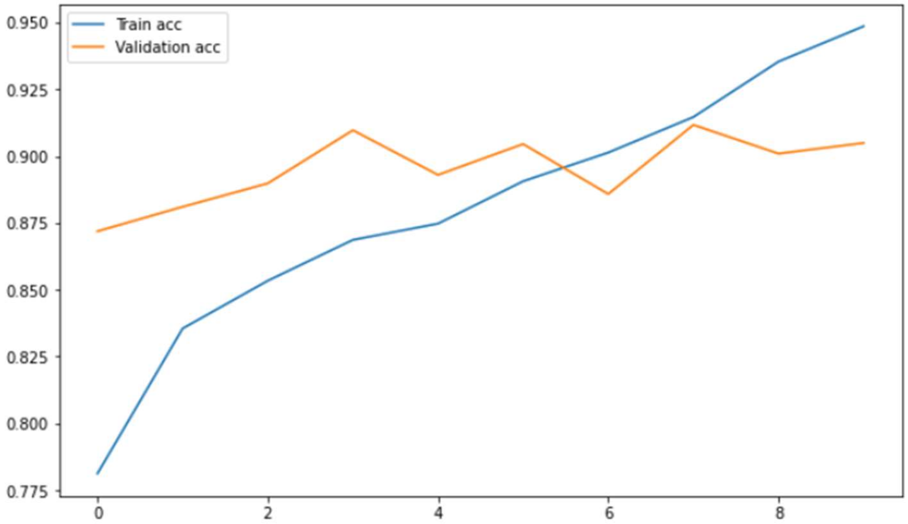
```
Epoch 1/10
 1/89 ━━━━━━━━━━━━━━━━━━━━ 1:17:00 53s/step - accuracy: 0.5781 - loss: 0.6941
```

Running until 10th epoch.

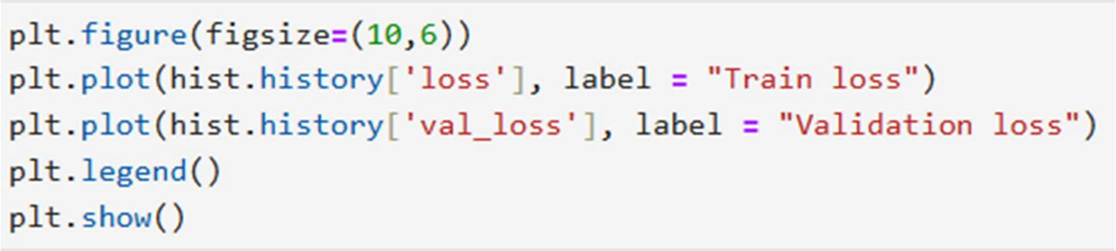
12. Plot Training Accuracy



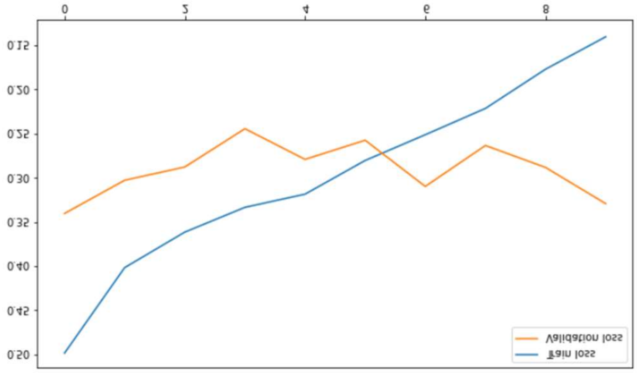
```
plt.figure(figsize=[10,6])
plt.plot(hist.history["accuracy"], label = "Train acc")
plt.plot(hist.history["val_accuracy"], label = "Validation acc")
plt.legend()
plt.show()
```



13. Plot Loss



```
plt.figure(figsize=(10,6))
plt.plot(hist.history['loss'], label = "Train loss")
plt.plot(hist.history['val_loss'], label = "Validation loss")
plt.legend()
plt.show()
```



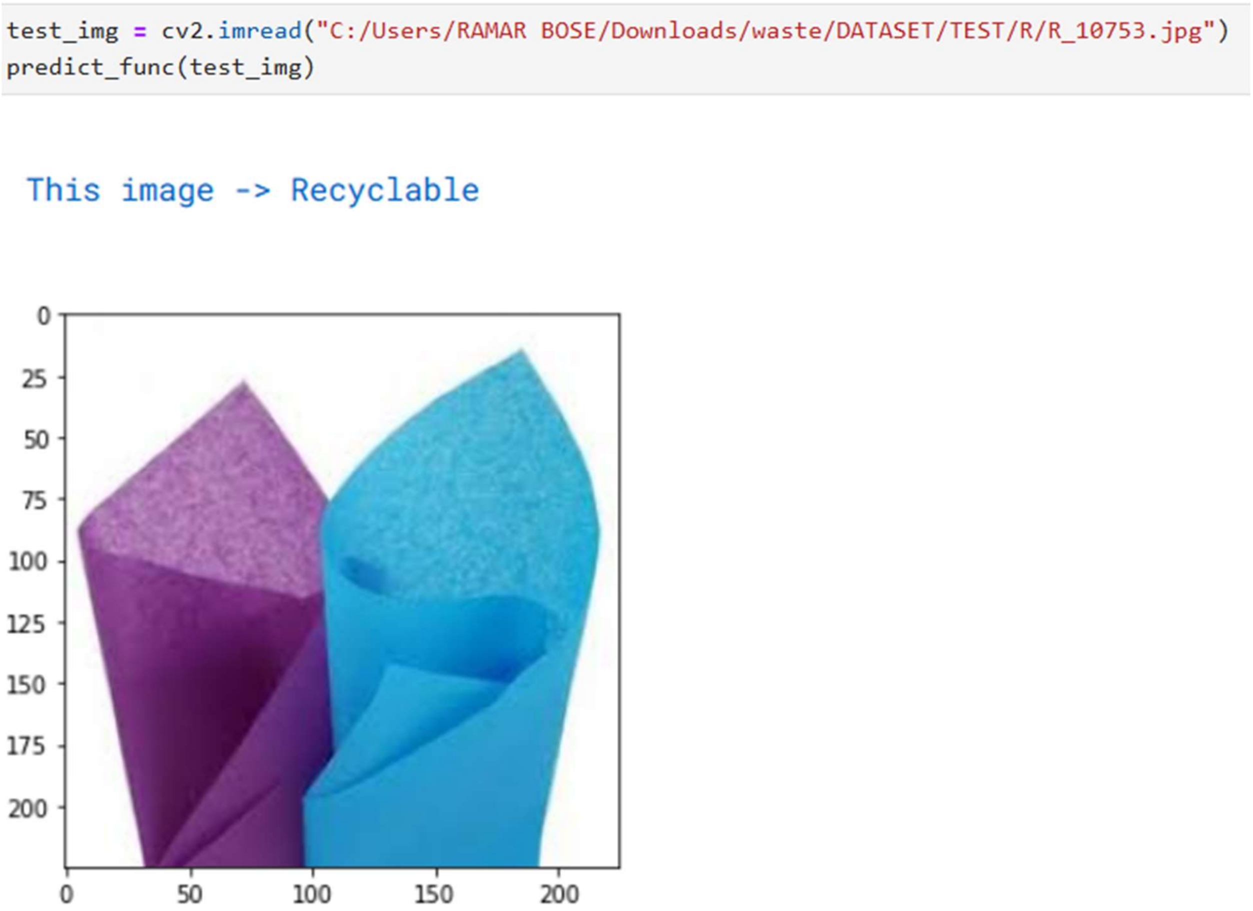
14. Model Prediction

```
def predict_func(img):
    plt.figure(figsize=(6,4))
    plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    plt.tight_layout()
    img = cv2.resize(img, (224, 224))
    img = np.reshape(img, [-1, 224, 224,3])
    result = np.argmax(model.predict(img))
    if result == 0: print("\033[94m"+"This image -> Recyclable"+"\033[0m")
    elif result ==1: print("\033[94m"+"This image -> Organic"+"\033[0m")
```

```
test_img = cv2.imread("C:/Users/RAMAR BOSE/Downloads/waste/DATASET/TEST/O/O_12573.jpg")
predict_func(test_img)
```

```
This image -> Organic
```

Disclaimer: The content is curated from online/offline resources and used for educational purpose only



```
test_img = cv2.imread("C:/Users/RAMAR BOSE/Downloads/waste/DATASET/TEST/R/R_10753.jpg")
predict_func(test_img)
```

```
This image -> Recyclable
```