```
import os

# Create a directory for Kaggle config
os.makedirs("/root/.kaggle", exist_ok=True)

# Upload `kaggle.json`
from google.colab import files
files.upload()  # Select and upload the downloaded kaggle.json file

# Move `kaggle.json` to the correct directory
!mv kaggle.json /root/.kaggle/

# Set permissions
!chmod 600 /root/.kaggle/kaggle.json

# Verify Kaggle API works
!kaggle datasets list
```

Choose Files kaggle.json

- **kaggle.json**(application/json) - 67 bytes, last modified: 3/20/2025 - 100% done

```
Saving kaggle.json to kaggle.json
ref                                                                     title                                                si
----------------------------------------------------------------------  ---------------------------------------------------  --------
atharvasoundankar/chocolate-sales                                       Chocolate Sales Data 📊🍫
atharvasoundankar/global-food-wastage-dataset-2018-2024                 🌍 Global Food Wastage Dataset (2018-2024) 🍽️            10
abdulmalik1518/mobiles-dataset-2025                                     Mobiles Dataset (2025)                                   203
adilshamim8/student-depression-dataset                                  Student Depression Dataset                              4670
mahmoudelhemaly/students-grading-dataset                                Student Performance & Behavior Dataset                  5204
atharvasoundankar/global-water-consumption-dataset-2000-2024            Global Water Consumption Dataset (2000-2024) 🌍💧
parsabahramsari/wdi-education-health-and-employment-2011-2021           WDI: Education, Health & Employment (2011-2021)         1361
atharvasoundankar/global-energy-consumption-2000-2024                   Global Energy Consumption (2000-2024) 🔥⚡                  2
aniruddhawankhede/mental-heath-analysis-among-teenagers                 Mental_Heath_Analysis_Among_Teenagers                   1770
salahuddinahmedshuvo/ecommerce-consumer-behavior-analysis-data          Ecommerce Consumer Behavior Analysis Data                442
smayanj/netflix-users-database                                          Netflix Users Database                                  3625
willianoliveiragibin/grocery-inventory                                  Grocery Inventory                                        508
atharvasoundankar/global-music-streaming-trends-and-listener-insights   Global Music Streaming Trends & Listener Insights        974
atharvasoundankar/viral-social-media-trends-and-engagement-analysis     🚀 Viral Social Media Trends & Engagement Analysis       107
anandshaw2001/imdb-movies-and-tv-shows                                  IMDb Movies and TV Shows                               25486
rzgiza/pokdex-for-all-1025-pokemon-w-text-description                   Pokédex For All 1025 Pokémon (+ text descriptions)       757
brsahan/genomic-data-for-cancer                                         Genomic Data for Cancer                                   91
amanrajput16/olympics-medal-list-1896-2024                              Olympic Medal List (1896-2024)                           111
miadul/brain-tumor-dataset                                              Brain Tumor Dataset                                     8725
adilshamim8/student-performance-on-an-entrance-examination              Student Performance on an Entrance Examination            44
```

```
!kaggle datasets download -d jaiharish11499/wastedata
```

```
Dataset URL: https://www.kaggle.com/datasets/jaiharish11499/wastedata
License(s): CC0-1.0
```

```
import zipfile

with zipfile.ZipFile("wastedata.zip", 'r') as zip_ref:
    zip_ref.extractall("waste_data")
```

```
import os
print(os.listdir("/content/"))
```

```
['.config', 'wastedata.zip', 'waste_data', 'sample_data']
```

```
train_folder = "/content/waste_data/d/Train"
test_folder = "/content/waste_data/d/Test"
```

```
import pandas as pd
import numpy as np
import glob
from datetime import datetime
from packaging import version

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.applications import VGG19
from tensorflow.keras.preprocessing import image_dataset_from_directory
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.keras.callbacks import ModelCheckpoint, History
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import Conv2D, Lambda, MaxPooling2D, Dense, Dropout, Flatten, GlobalAveragePooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.utils import to_categorical

from skimage.io import imread, imshow
from skimage.transform import resize
from IPython import display
import matplotlib.pyplot as plt
import seaborn as sns
from seaborn import heatmap
from sklearn.metrics import confusion_matrix
```

```
# Data augmentation for training
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=30,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest')
```

```
# No augmentation for validation/test
test_datagen = ImageDataGenerator(rescale=1./255)
```

```
train_generator = train_datagen.flow_from_directory(
    train_folder,
    target_size=(224, 224),
    batch_size=64,
    class_mode='binary'
)
```

```
Found 336 images belonging to 2 classes.
```

```
test_generator = test_datagen.flow_from_directory(
    test_folder,
    target_size=(224, 224),
    batch_size=64,
    class_mode='binary',
    shuffle=False
)
```

```
Found 64 images belonging to 2 classes.
```

```
from sklearn.utils.class_weight import compute_class_weight
# Compute class weights to address imbalance
class_labels = np.array(train_generator.classes)
class_weights = compute_class_weight(class_weight='balanced', classes=np.unique(class_labels), y=class_labels)
class_weight_dict = {i: class_weights[i] for i in range(len(class_weights))}
```

```
base_model = VGG19(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
base_model.trainable = False
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg19/vgg19_weights_tf_dim_ordering_tf_kernels_n
80134624/80134624 ━━━━━━━━━━━━━━━━━━━━ 2s 0us/step
```

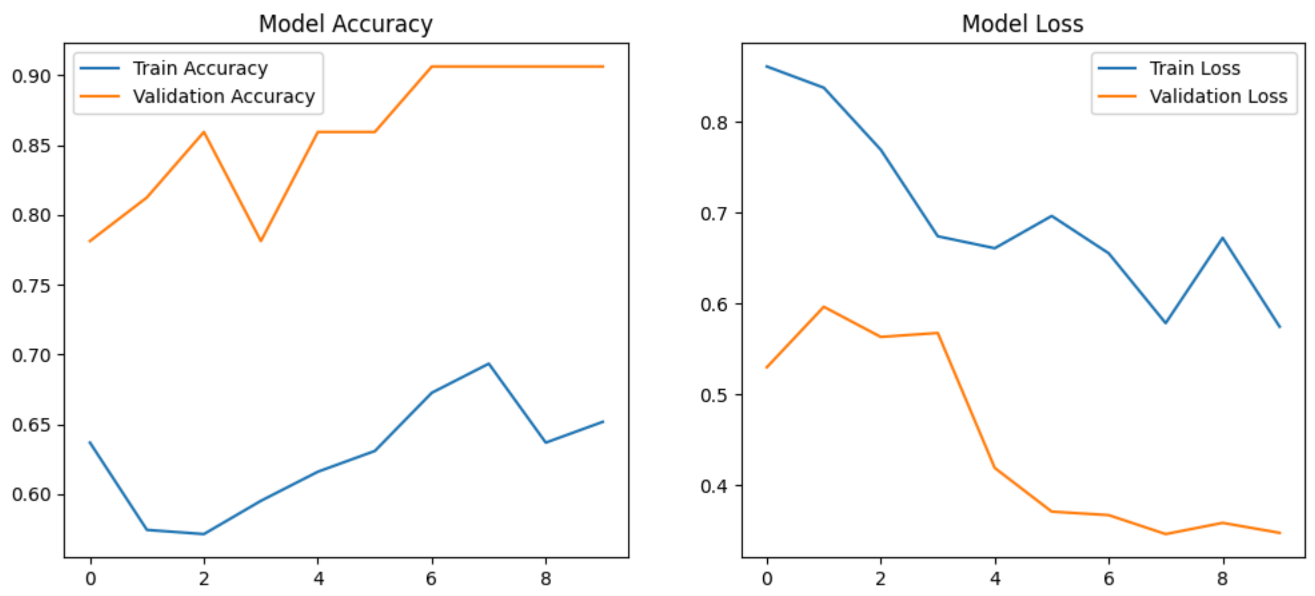
```
model = keras.Sequential([
    base_model,
    keras.layers.Flatten(),  # Instead of GlobalAveragePooling2D()
    keras.layers.Dense(256, activation='relu'),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(1, activation='sigmoid')
])
```

```
from tensorflow.keras.optimizers import Adam
#Compile the model
model.compile(optimizer=Adam(learning_rate=0.0001), loss='binary_crossentropy', metrics=['accuracy'])
```

```
# Train model
history = model.fit(
    train_generator,
    epochs=10,
    validation_data=test_generator,
    class_weight=class_weight_dict)
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` cl
  self._warn_if_super_not_called()
Epoch 1/10
6/6 ━━━━━━━━━━━━━━━━━━━━ 308s 51s/step - accuracy: 0.5507 - loss: 0.8832 - val_accuracy: 0.7812 - val_loss: 0.5300
Epoch 2/10
6/6 ━━━━━━━━━━━━━━━━━━━━ 338s 58s/step - accuracy: 0.5837 - loss: 0.8952 - val_accuracy: 0.8125 - val_loss: 0.5965
Epoch 3/10
6/6 ━━━━━━━━━━━━━━━━━━━━ 388s 66s/step - accuracy: 0.5820 - loss: 0.7672 - val_accuracy: 0.8594 - val_loss: 0.5634
Epoch 4/10
6/6 ━━━━━━━━━━━━━━━━━━━━ 302s 51s/step - accuracy: 0.6108 - loss: 0.7149 - val_accuracy: 0.7812 - val_loss: 0.5677
Epoch 5/10
6/6 ━━━━━━━━━━━━━━━━━━━━ 330s 52s/step - accuracy: 0.5872 - loss: 0.7347 - val_accuracy: 0.8594 - val_loss: 0.4191
Epoch 6/10
6/6 ━━━━━━━━━━━━━━━━━━━━ 302s 50s/step - accuracy: 0.6388 - loss: 0.6812 - val_accuracy: 0.8594 - val_loss: 0.3710
Epoch 7/10
6/6 ━━━━━━━━━━━━━━━━━━━━ 344s 59s/step - accuracy: 0.6954 - loss: 0.6429 - val_accuracy: 0.9062 - val_loss: 0.3672
Epoch 8/10
6/6 ━━━━━━━━━━━━━━━━━━━━ 301s 50s/step - accuracy: 0.7057 - loss: 0.5326 - val_accuracy: 0.9062 - val_loss: 0.3463
Epoch 9/10
6/6 ━━━━━━━━━━━━━━━━━━━━ 308s 52s/step - accuracy: 0.6297 - loss: 0.6761 - val_accuracy: 0.9062 - val_loss: 0.3586
Epoch 10/10
6/6 ━━━━━━━━━━━━━━━━━━━━ 300s 50s/step - accuracy: 0.6485 - loss: 0.5263 - val_accuracy: 0.9062 - val_loss: 0.3477
```

```
# Plot accuracy and loss
fig, axes = plt.subplots(1, 2, figsize=(12, 5))
axes[0].plot(history.history['accuracy'], label='Train Accuracy')
axes[0].plot(history.history['val_accuracy'], label='Validation Accuracy')
axes[0].set_title('Model Accuracy')
axes[0].legend()
axes[1].plot(history.history['loss'], label='Train Loss')
axes[1].plot(history.history['val_loss'], label='Validation Loss')
axes[1].set_title('Model Loss')
axes[1].legend()
plt.show()
```

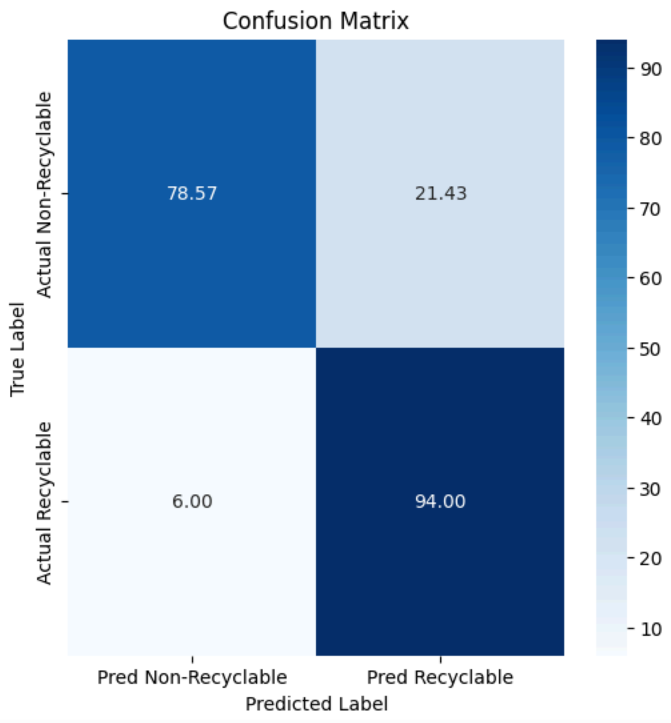


```
# Confusion matrix
y_true = test_generator.classes
y_pred = model.predict(test_generator) > 0.5
cm = confusion_matrix(y_true, y_pred)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 49s 49s/step
```

```
# Display confusion matrix with labels and percentages
fig, ax = plt.subplots(figsize=(6, 6))
cm_percent = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis] * 100
sns.heatmap(cm_percent, annot=True, fmt='.2f', cmap='Blues', xticklabels=['Pred Non-Recyclable', 'Pred Recyclable'],
            yticklabels=['Actual Non-Recyclable', 'Actual Recyclable'])
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix')
```

```
Text(0.5, 1.0, 'Confusion Matrix')
```



```
from sklearn.metrics import classification_report
# Classification report
print("Classification Report:")
print(classification_report(y_true, y_pred, target_names=['Non-Recyclable', 'Recyclable']))
```

```
Classification Report:
                precision    recall  f1-score   support

Non-Recyclable       0.79      0.79      0.79        14
    Recyclable       0.94      0.94      0.94        50

      accuracy                           0.91        64
     macro avg       0.86      0.86      0.86        64
  weighted avg       0.91      0.91      0.91        64
```

```
# Convert accuracy and loss to percentage
train_acc = [x * 100 for x in history.history['accuracy']]
val_acc = [x * 100 for x in history.history['val_accuracy']]
train_loss = [x * 100 for x in history.history['loss']]
val_loss = [x * 100 for x in history.history['val_loss']]
# Print accuracy and loss values
print("Final Training Accuracy: {:.2f}%".format(train_acc[-1]))
print("Final Validation Accuracy: {:.2f}%".format(val_acc[-1]))
print("Final Training Loss: {:.2f}%".format(train_loss[-1]))
print("Final Validation Loss: {:.2f}%".format(val_loss[-1]))
```

```
Final Training Accuracy: 65.18%
Final Validation Accuracy: 90.62%
Final Training Loss: 57.46%
Final Validation Loss: 34.77%
```