```
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                                                  size
--------------------------------------------------------------------  ------------------------------------------------  ---------
atharvasoundankar/chocolate-sales                                     Chocolate Sales Data 📊🍫                                 14
abdulmalik1518/mobiles-dataset-2025                                   Mobiles Dataset (2025)                                 20314
mahmoudelhemaly/students-grading-dataset                              Student Performance & Behavior Dataset                520428
atharvasoundankar/global-water-consumption-dataset-2000-2024          Global Water Consumption Dataset (2000-2024) 🌍💧          17
adilshamim8/student-depression-dataset                                Student Depression Dataset                            467020
atharvasoundankar/global-food-wastage-dataset-2018-2024               🌍 Global Food Wastage Dataset (2018-2024) 🍽️              1080
atharvasoundankar/global-energy-consumption-2000-2024                 Global Energy Consumption (2000-2024) 🔥⚡                  258
parsabahramsari/wdi-education-health-and-employment-2011-2021         WDI: Education, Health & Employment (2011-2021)       136185
bhargavchirumamilla/netflix-movies-and-tv-shows-till-2025             Netflix Movies and TV shows till 2025                6471169
aniruddhawankhede/mental-heath-analysis-among-teenagers               Mental_Heath_Analysis_Among_Teenagers                 177089
salahuddinahmedshuvo/ecommerce-consumer-behavior-analysis-data        Ecommerce Consumer Behavior Analysis Data              44265
smayanj/netflix-users-database                                        Netflix Users Database                                362559
willianoliveiragibin/grocery-inventory                                Grocery Inventory                                      50801
atharvasoundankar/global-music-streaming-trends-and-listener-insights Global Music Streaming Trends & Listener Insights       97474
atharvasoundankar/viral-social-media-trends-and-engagement-analysis   🚀 Viral Social Media Trends & Engagement Analysis       1072
anandshaw2001/imdb-movies-and-tv-shows                                IMDb Movies and TV Shows                             2548638
brsahan/genomic-data-for-cancer                                       Genomic Data for Cancer                                 9134
amanrajput16/olympics-medal-list-1896-2024                            Olympic Medal List (1896-2024)                         11101
miadul/brain-tumor-dataset                                            Brain Tumor Dataset                                   872531
adilshamim8/student-performance-on-an-entrance-examination            Student Performance on an Entrance Examination          4402
```

```
!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
import os
from datetime import datetime
from packaging import version

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.applications import EfficientNetB0
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
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
```

```
from tensorflow.keras.applications.efficientnet import preprocess_input

# Data augmentation for training
train_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input,  # EfficientNetB0-specific preprocessing
    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_datagen = ImageDataGenerator(preprocessing_function=preprocess_input)
```

```
# Load dataset
train_generator = train_datagen.flow_from_directory(
    train_folder,
    target_size=(224, 224),
    batch_size=32,
    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=32,
    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))}
```

```
# Load EfficientNetB0 base model (pre-trained on ImageNet)
base_model = EfficientNetB0(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
```

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

```
base_model.trainable = False
```

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

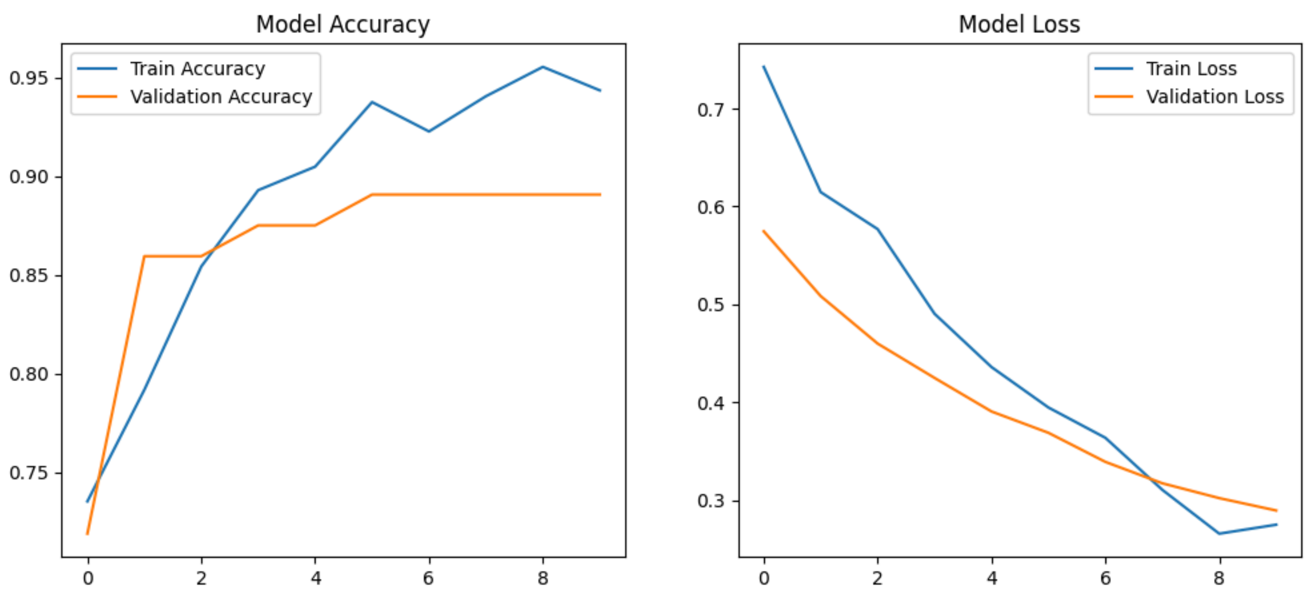
```
from tensorflow.keras.optimizers import Adam
#Compile the model
```

```
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
11/11 ━━━━━━━━━━━━━━━━━━━━ 55s 4s/step - accuracy: 0.7120 - loss: 0.7327 - val_accuracy: 0.7188 - val_loss: 0.5745
Epoch 2/10
11/11 ━━━━━━━━━━━━━━━━━━━━ 39s 3s/step - accuracy: 0.7916 - loss: 0.5690 - val_accuracy: 0.8594 - val_loss: 0.5083
Epoch 3/10
11/11 ━━━━━━━━━━━━━━━━━━━━ 43s 4s/step - accuracy: 0.8416 - loss: 0.6057 - val_accuracy: 0.8594 - val_loss: 0.4598
Epoch 4/10
11/11 ━━━━━━━━━━━━━━━━━━━━ 37s 3s/step - accuracy: 0.8762 - loss: 0.4989 - val_accuracy: 0.8750 - val_loss: 0.4247
Epoch 5/10
11/11 ━━━━━━━━━━━━━━━━━━━━ 42s 4s/step - accuracy: 0.9229 - loss: 0.4073 - val_accuracy: 0.8750 - val_loss: 0.3905
Epoch 6/10
11/11 ━━━━━━━━━━━━━━━━━━━━ 38s 3s/step - accuracy: 0.9439 - loss: 0.4057 - val_accuracy: 0.8906 - val_loss: 0.3687
Epoch 7/10
11/11 ━━━━━━━━━━━━━━━━━━━━ 37s 4s/step - accuracy: 0.9272 - loss: 0.3505 - val_accuracy: 0.8906 - val_loss: 0.3390
Epoch 8/10
11/11 ━━━━━━━━━━━━━━━━━━━━ 37s 3s/step - accuracy: 0.9409 - loss: 0.3186 - val_accuracy: 0.8906 - val_loss: 0.3171
Epoch 9/10
11/11 ━━━━━━━━━━━━━━━━━━━━ 45s 4s/step - accuracy: 0.9550 - loss: 0.2778 - val_accuracy: 0.8906 - val_loss: 0.3020
Epoch 10/10
11/11 ━━━━━━━━━━━━━━━━━━━━ 39s 4s/step - accuracy: 0.9411 - loss: 0.2749 - val_accuracy: 0.8906 - val_loss: 0.2893
```

```
# 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()
```



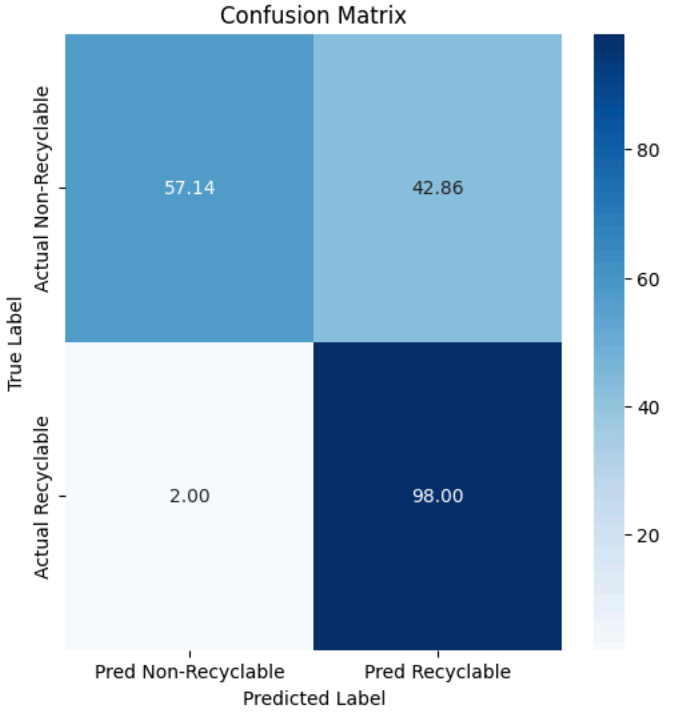

```
# Confusion matrix
y_true = test_generator.classes
y_pred = model.predict(test_generator) > 0.5
cm = confusion_matrix(y_true, y_pred)
```

```
2/2 ━━━━━━━━━━━━━━━━━━━━ 9s 2s/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.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       0.89      0.57      0.70        14
    Recyclable       0.89      0.98      0.93        50

      accuracy                           0.89        64
     macro avg       0.89      0.78      0.81        64
  weighted avg       0.89      0.89      0.88        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: 94.35%
Final Validation Accuracy: 89.06%
Final Training Loss: 27.48%
Final Validation Loss: 28.93%
```