

```
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

Warning: Looks like you're using an outdated API Version, please consider updating (server 1.7.4.2 / client 1.6.17)

ref	title	size	last modified
atharvasoundankar/chocolate-sales	Chocolate Sales Data 🇮🇳🍫	14KB	20%
atharvasoundankar/global-food-wastage-dataset-2018-2024	Global Food Wastage Dataset (2018-2024) 🌍🗑️	106KB	20%
abdulmalik1518/mobiles-dataset-2025	Mobiles Dataset (2025)	20KB	20%
adilshamim8/student-depression-dataset	Student Depression Dataset	456KB	20%
mahmoudelhemy/students-grading-dataset	Student Performance & Behavior Dataset	508KB	20%
atharvasoundankar/global-water-consumption-dataset-2000-2024	Global Water Consumption Dataset (2000-2024) 🌍💧	17KB	20%
atharvasoundankar/global-energy-consumption-2000-2024	Global Energy Consumption (2000-2024) 🔥⚡	252KB	20%
aniruddhawankhede/mental-heath-analysis-among-teenagers	Mental Heath Analysis Among Teenagers	173KB	20%
salahuddinahmedshuvo/ecommerce-consumer-behavior-analysis-data	Ecommerce Consumer Behavior Analysis Data	43KB	20%
smayanj/netflix-users-database	Netflix Users Database	354KB	20%
willianoliveiragabin/grocery-inventory	Grocery Inventory	50KB	20%
alikalwar/heart-attack-risk-prediction-cleaned-dataset	Heart Attack Risk Prediction Cleaned Dataset	671KB	20%
atharvasoundankar/global-music-streaming-trends-and-listener-insights	Global Music Streaming Trends & Listener Insights	95KB	20%
atharvasoundankar/viral-social-media-trends-and-engagement-analysis	🔥 Viral Social Media Trends & Engagement Analysis	105KB	20%
brsahan/genomic-data-for-cancer	Genomic Data for Cancer	9KB	20%
amanrajput16/olympics-medal-list-1896-2024	Olympic Medal List (1896-2024)	11KB	20%
miadul/brain-tumor-dataset	Brain Tumor Dataset	852KB	20%
adilshamim8/student-performance-on-an-entrance-examination	Student Performance on an Entrance Examination	4KB	20%
atharvasoundankar/big-4-financial-risk-insights-2020-2025	Big 4 Financial Risk Insights (2020-2025)	3KB	20%
shantanugarg274/lung-cancer-prediction-dataset	Lung Cancer Prediction Dataset	127KB	20%

```
!kaggle datasets download -d preetishah/waste-classificationorganic-and-recyclable
```

 Warning: Looks like you're using an outdated API Version, please consider updating (server 1.7.4.2 / client 1.6.17)

Dataset URL: <https://www.kaggle.com/datasets/preetishah/waste-classificationorganic-and-recyclable>

License(s): apache-2.0

Downloading waste-classificationorganic-and-recyclable.zip to /content

52% 13.0M/25.0M [00:00<00:00, 42.5MB/s]

100% 25.0M/25.0M [00:00<00:00, 69.8MB/s]

```
import zipfile

with zipfile.ZipFile("waste-classificationorganic-and-recyclable.zip", 'r') as zip_ref:
    zip_ref.extractall("waste_classification")
```

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

```
 ['.config', 'waste-classificationorganic-and-recyclable.zip', 'waste_classification', 'sample_data']
```

```
train_folder = "/content/waste_classification/wasteclassification/train"
test_folder = "/content/waste_classification/wasteclassification/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 666 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 32 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\_n80134624/80134624 — 0s 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

```

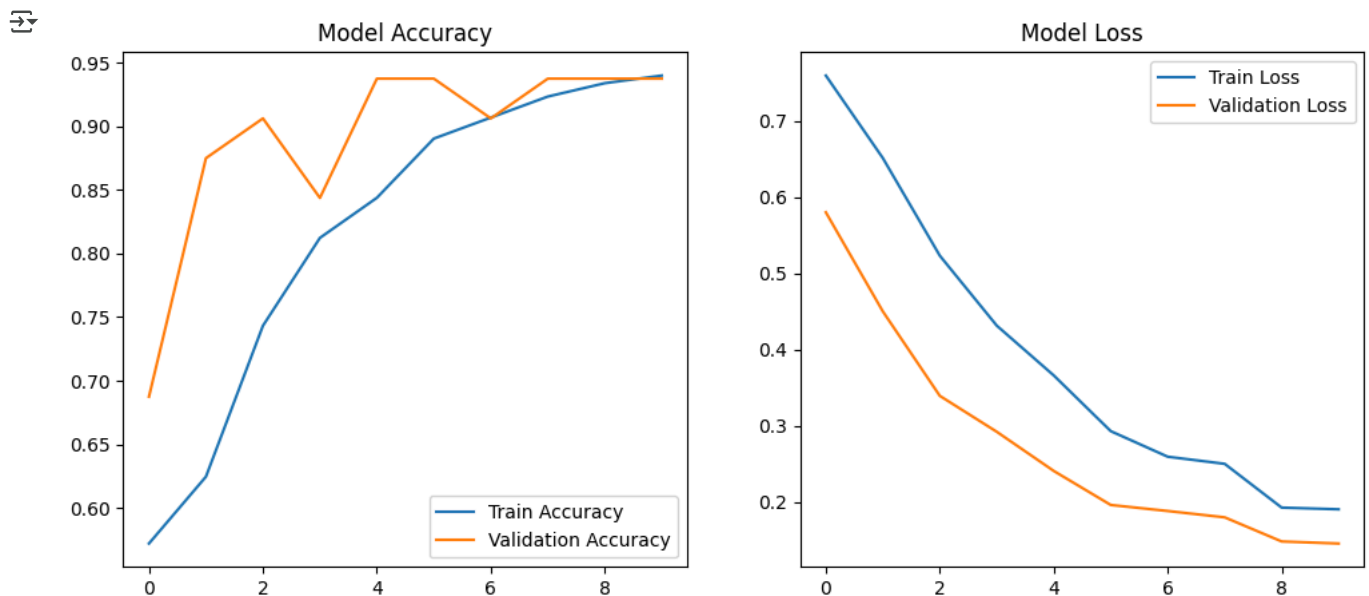
```
model.compile(optimizer=Adam(learning_rate=0.0001), loss='binary_crossentropy', metrics=['accuracy'])
```

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

```
1 /usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` c
self._warn_if_super_not_called()
```

```
Epoch 1/10
11/11 ----- 542s 49s/step - accuracy: 0.5606 - loss: 0.7867 - val_accuracy: 0.6875 - val_loss: 0.5800
Epoch 2/10
11/11 ----- 550s 50s/step - accuracy: 0.5836 - loss: 0.7067 - val_accuracy: 0.8750 - val_loss: 0.4501
Epoch 3/10
11/11 ----- 549s 50s/step - accuracy: 0.7203 - loss: 0.5427 - val_accuracy: 0.9062 - val_loss: 0.3393
Epoch 4/10
11/11 ----- 559s 50s/step - accuracy: 0.8187 - loss: 0.4286 - val_accuracy: 0.8438 - val_loss: 0.2922
Epoch 5/10
11/11 ----- 546s 50s/step - accuracy: 0.8414 - loss: 0.3737 - val_accuracy: 0.9375 - val_loss: 0.2409
Epoch 6/10
11/11 ----- 535s 48s/step - accuracy: 0.8910 - loss: 0.2965 - val_accuracy: 0.9375 - val_loss: 0.1963
Epoch 7/10
11/11 ----- 535s 49s/step - accuracy: 0.9164 - loss: 0.2558 - val_accuracy: 0.9062 - val_loss: 0.1884
Epoch 8/10
11/11 ----- 535s 48s/step - accuracy: 0.9280 - loss: 0.2575 - val_accuracy: 0.9375 - val_loss: 0.1800
Epoch 9/10
11/11 ----- 536s 49s/step - accuracy: 0.9283 - loss: 0.2065 - val_accuracy: 0.9375 - val_loss: 0.1484
Epoch 10/10
11/11 ----- 535s 49s/step - accuracy: 0.9547 - loss: 0.1777 - val_accuracy: 0.9375 - val_loss: 0.1457
```

```
# 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()
```



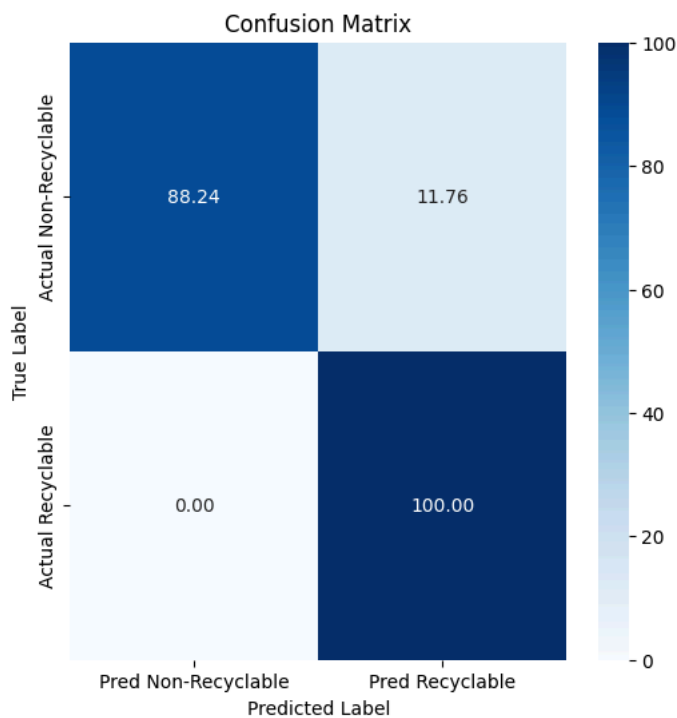
```
# Confusion matrix
y_true = test_generator.classes
y_pred = model.predict(test_generator) > 0.5
cm = confusion_matrix(y_true, y_pred)
```

```
1 / 1 ----- 25s 25s/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	1.00	0.88	0.94	17
Recyclable	0.88	1.00	0.94	15
accuracy			0.94	32
macro avg	0.94	0.94	0.94	32
weighted avg	0.94	0.94	0.94	32

```
# 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: 93.99%  
 Final Validation Accuracy: 93.75%  
 Final Training Loss: 19.06%  
 Final Validation Loss: 14.57%

