

CS 316 Phase (2) Screen Shots

```
[1] ✓ 25s
# Install & Import Libraries
!pip install tensorflow matplotlib opendatasets

import os
import cv2
import numpy as np
import matplotlib.pyplot as plt
import opendatasets as od
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from google.colab import drive

Requirement already satisfied: tensorflow in /usr/local/lib/python3.12/dist-packages (2.19.0)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-packages (3.10.0)
Collecting opendatasets
  Downloading opendatasets-0.1.22-py3-none-any.whl.metadata (9.2 kB)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (25.9.23)
Requirement already satisfied: gast!=0.5.0,!0.5.1,!0.5.2,>=0.2.1 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (0.2.8)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (18.1.1)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (3.4.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.12/dist-packages (from tensorflow) (25.0)
Requirement already satisfied: protobuf!=4.21.0,!4.21.1,!4.21.2,!4.21.3,!4.21.4,!4.21.5,<6.0.0dev,>=3.20.3 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (5.29.5)
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (2.32.4)
Requirement already satisfied: setuptools in /usr/local/lib/python3.12/dist-packages (from tensorflow) (75.2.0)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (1.17.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (3.1.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (4.15.0)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (2.0.8)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (2.17.0)
Requirement already satisfied: tensorboard>=2.19.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (2.19.0)
Requirement already satisfied: keras>=3.5.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (3.10.0)
Requirement already satisfied: numpy>=2.2.0,>=1.26.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (2.0.2)
Requirement already satisfied: h5py>=2.11.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (3.15.1)
Requirement already satisfied: ml-dtypes<1.0.0,>=0.5.1 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (0.5.3)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (1.3.3)
Requirement already satisfied: cypher>=0.10 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (4.60.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (1.4.9)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (11.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (3.2.5)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (2.9.0.post0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages (from opendatasets) (4.67.1)
Requirement already satisfied: kaggle in /usr/local/lib/python3.12/dist-packages (from opendatasets) (1.7.4.5)
Requirement already satisfied: click in /usr/local/lib/python3.12/dist-packages (from opendatasets) (8.3.0)
Requirement already satisfied: wheel<1.0.0,>=0.23.0 in /usr/local/lib/python3.12/dist-packages (from astunparse>=1.6.0->tensorflow) (0.45.1)
Requirement already satisfied: rich in /usr/local/lib/python3.12/dist-packages (from keras>=3.5.0->tensorflow) (13.9.4)
Requirement already satisfied: namex in /usr/local/lib/python3.12/dist-packages (from keras>=3.5.0->tensorflow) (0.1.0)
Requirement already satisfied: optree in /usr/local/lib/python3.12/dist-packages (from keras>=3.5.0->tensorflow) (0.17.0)
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests<3,>=2.21.0->tensorflow) (3.4.4)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests<3,>=2.21.0->tensorflow) (3.11)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests<3,>=2.21.0->tensorflow) (2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests<3,>=2.21.0->tensorflow) (2025.10.5)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.12/dist-packages (from tensorflow>=2.19.0->tensorflow) (3.9)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow>=2.19.0->tensorflow) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from tensorflow>=2.19.0->tensorflow) (3.1.3)
Requirement already satisfied: bleach in /usr/local/lib/python3.12/dist-packages (from kaggle->opendatasets) (6.2.0)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.12/dist-packages (from kaggle->opendatasets) (8.0.4)
Requirement already satisfied: text-unidecode in /usr/local/lib/python3.12/dist-packages (from kaggle->opendatasets) (1.3)
Requirement already satisfied: webencodings in /usr/local/lib/python3.12/dist-packages (from kaggle->opendatasets) (0.5.1)
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.12/dist-packages (from werkzeug>=1.0.1->tensorflow>=2.19.0->tensorflow) (3.0.3)
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.12/dist-packages (from rich->keras>=3.5.0->tensorflow) (4.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.12/dist-packages (from rich->keras>=3.5.0->tensorflow) (2.19.2)
Requirement already satisfied: mdurl>=0.1 in /usr/local/lib/python3.12/dist-packages (from markdown-it-py>=2.2.0->rich->keras>=3.5.0->tensorflow) (0.1.2)
Downloaded opendatasets-0.1.22-py3-none-any.whl (15 kB)
Installing collected packages: opendatasets
Successfully installed opendatasets-0.1.22
```

[2]
✓ 19s

```
# Download Dataset from Kaggle

od.download("https://www.kaggle.com/datasets/asdasdasdas/garbage-classification")

# Adjust dataset path
dataset_path = "/content/garbage-classification/Garbage classification/Garbage classification"
print(os.listdir(dataset_path)) # will show folders like cardboard, glass, etc

→ Please provide your Kaggle credentials to download this dataset. Learn more: http://bit.ly/kaggle-creds
Your Kaggle username: "shajanmokyel"
Your Kaggle Key: .....
Dataset URL: https://www.kaggle.com/datasets/asdasdasdas/garbage-classification
Downloading garbage-classification.zip to ./garbage-classification
100%|██████████| 82.0M/82.0M [00:00<00:00, 951MB/s]

['paper', 'glass', 'trash', 'plastic', 'cardboard', 'metal']
```

[3]
✓ 7s

```
# Load & Preprocess Images

image_size = 224
data, labels = [], []

class_names = ['cardboard', 'glass', 'metal', 'paper', 'plastic', 'trash']
class_map = {name: idx for idx, name in enumerate(class_names)}

for class_name in class_names:
    class_path = os.path.join(dataset_path, class_name)
    for img_name in os.listdir(class_path):
        img_path = os.path.join(class_path, img_name)
        img = cv2.imread(img_path)
        if img is not None:
            img = cv2.resize(img, (image_size, image_size))
            img = img / 255.0
            data.append(img)
            labels.append(class_map[class_name])

data = np.array(data)
labels = np.array(labels)
```

[4]
✓ 0s

```
# Split Dataset
X_train, X_temp, y_train, y_temp = train_test_split(data, labels, test_size=0.3, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)
```

[5]
✓ 2s

▶

```
# Data Augmentation
train_datagen = ImageDataGenerator(
    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'
)
train_datagen.fit(X_train)
```

[6]
✓ 25s

▶

```
# Mount Google Drive
drive.mount('/content/drive')
model_path = "/content/drive/MyDrive/waste_model_v2.h5"
```

➡ Mounted at /content/drive

```
[7] 5s
# Train Model (Using Transfer Learning)
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import GlobalAveragePooling2D, Input, Dense, Dropout
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau

model_path = "/content/drive/MyDrive/waste_model_v2.h5"

if os.path.exists(model_path):
    print("نموذج موجود بالفعل، سيتم تعيينه")
    model = load_model(model_path)
    print("تم تعيين النموذج بنجاح، لا حاجة لتدريبه")
else:
    print("Transfer Learning")

# 1. Load the previously trained base model (MobileNetV2)
base_model = MobileNetV2(
    input_shape=(image_size, image_size, 3),
    include_top=False, # We do not want to include the final ImageNet classifier layer
    weights='imagenet'
)

# 2. Freeze the base model
base_model.trainable = False # We don't want to re train the layers that already learned the features

# 3. Adding a classifier "head"
# We implemented the Keras functional API
inputs = Input(shape=(image_size, image_size, 3))
x = base_model(inputs, training=False)
x = GlobalAveragePooling2D()(x)
x = Dropout(0.5)(x)
x = Dense(128, activation='relu')(x)
outputs = Dense(6, activation='softmax')(x) # 6 classes

# 4. Create new model
model = Model(inputs, outputs)

model.compile(optimizer=Adam(learning_rate=0.001),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# 5. Defining when to stop
early_stop = EarlyStopping(
    monitor='val_accuracy',
    patience=5, # Stop after 5 epochs of no improvement
    restore_best_weights=True # Keep the best version of the model
)
# If the model gets "stuck", reduce the learning rate
reduce_lr = ReduceLROnPlateau(
    monitor='val_loss',
    factor=0.2,
    patience=3,
    min_lr=1e-6
)
callbacks_list = [early_stop, reduce_lr]

# 6. Train model
history = model.fit(
    train_datagen.flow(X_train, y_train, batch_size=32),
    epochs=50, # Increased epochs with the help of EarlyStopping to find the best one
    validation_data=(X_val, y_val),
    callbacks=callbacks_list # Use the callbacks from step 5
)

# 7. Save model in google drive to prevent re training every time
model.save(model_path)
print("تم حفظ النموذج في Google Drive")
```

قراص محفوظ موجود بالفعل، سيتم تعيينه
 WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
 تم تعيين المفروض بنجاح، لا حاجة لعملية التدريب

```
# Evaluate Model

test_loss, test_acc = model.evaluate(X_test, y_test)
print(f"دقة النموذج: {test_acc*100:.2f}%")

12/12 ----- 14s 957ms/step - accuracy: 0.9434 - loss: 0.2108
93.42 دقة النموذج:
```

```
# Predict with Arabic Labels

arabic_labels = {
    'cardboard': 'كرتون',
    'glass': 'زجاج',
    'metal': 'معدن',
    'paper': 'ورق',
    'plastic': 'بلاستيك',
    'trash': 'نفايات'
}

def predict_image(img_path):
    img = cv2.imread(img_path)
    if img is None:
        print(f"خطأ: تعتذر تحميل الصورة من {img_path}")
        return
    img = cv2.resize(img, (image_size, image_size))
    img = np.expand_dims(img, axis=0) / 255.0
    prediction = model.predict(img)
    class_idx = np.argmax(prediction)
    class_name = class_names[class_idx]
    print(f"الفئة المتوقعة: {arabic_labels[class_name]} ({class_name})")

# Example usage:
predict_image("/content/garbage-classification/Garbage classification/Garbage classification/glass/glass1.jpg")
```

```
1/1 ----- 1s 822ms/step
(glass) الفئة المتوقعة: زجاج
```

```

import gradio as gr

def classify_image_text_only(image):
    import cv2, numpy as np
    img = cv2.resize(image, (image_size, image_size))
    img = np.expand_dims(img, axis=0) / 255.0
    prediction = model.predict(img)
    class_idx = np.argmax(prediction)
    class_name = class_names[class_idx]
    arabic_label = arabic_labels[class_name]
    return f"النوع: {arabic_label}"

interface_text = gr.Interface(
    fn=classify_image_text_only,
    inputs=gr.Image(type="numpy", label="تحميل صورة النباتات"),
    outputs=gr.Textbox(label="النتيجة بالعربية"),
    title="نظام تصنیف النباتات بالصور",
    description="اختر صورة نباتات معروفة نوعها (ورق، بلاستيك، زجاج، كربون، معدن، نباتات)"
)

interface_text.launch()

```

It looks like you are running Gradio on a hosted Jupyter notebook, which requires `share=True` . Automatically setting `share=True` (you can turn this off by setting `share=False` in `launch()` explicitly).

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`

* Running on public URL: <https://3a3219790ec5faad8f.gradio.live>

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working directory to deploy to Hugging Face Spaces (<https://huggingface.co/spaces>)

نظام تصنیف النباتات بالصور

اختر صورة نباتات معروفة نوعها (ورق، بلاستيك، زجاج، كربون، معدن، نباتات).

