# 1. IMPORT LIBRARIES

import numpy as np

import matplotlib.pyplot as plt

import tensorflow as tf

from tensorflow.keras.preprocessing import image\_dataset\_from\_directory

from tensorflow.keras.applications import MobileNetV2

from tensorflow.keras.layers import GlobalAveragePooling2D, Dropout, Dense

from tensorflow.keras.models import Sequential

from sklearn.metrics import classification\_report, confusion\_matrix

import seaborn as sns

import os

import pathlib

# 2. LOAD & PREPROCESS DATA

dataset\_path = "/content/rice\_dataset" # Update to your path

batch\_size = 32

img\_size = (224, 224)

train\_ds = image\_dataset\_from\_directory(

dataset\_path,

validation\_split=0.2,

subset="training",

seed=123,

image\_size=img\_size,

batch\_size=batch\_size

)

val\_ds = image\_dataset\_from\_directory(

dataset\_path,

validation\_split=0.2,

subset="validation",

seed=123,

image\_size=img\_size,

batch\_size=batch\_size

)

class\_names = train\_ds.class\_names

AUTOTUNE = tf.data.AUTOTUNE

train\_ds = train\_ds.prefetch(buffer\_size=AUTOTUNE)

val\_ds = val\_ds.prefetch(buffer\_size=AUTOTUNE)

# 3. BUILD TRANSFER LEARNING MODEL (MobileNetV2)

base\_model = MobileNetV2(input\_shape=(224, 224, 3),

include\_top=False,

weights='imagenet')

base\_model.trainable = False

model = Sequential([

base\_model,

GlobalAveragePooling2D(),

Dropout(0.3),

Dense(128, activation='relu'),

Dropout(0.3),

Dense(len(class\_names), activation='softmax')

])

model.compile(optimizer='adam',

loss='sparse\_categorical\_crossentropy',

metrics=['accuracy'])

model.summary()

# 4. TRAIN MODEL

epochs = 10

history = model.fit(train\_ds, validation\_data=val\_ds, epochs=epochs)

# 5. VISUALIZE PERFORMANCE

acc = history.history['accuracy']

val\_acc = history.history['val\_accuracy']

loss = history.history['loss']

val\_loss = history.history['val\_loss']

epochs\_range = range(epochs)

plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)

plt.plot(epochs\_range, acc, label='Training Accuracy')

plt.plot(epochs\_range, val\_acc, label='Validation Accuracy')

plt.legend(loc='lower right')

plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)

plt.plot(epochs\_range, loss, label='Training Loss')

plt.plot(epochs\_range, val\_loss, label='Validation Loss')

plt.legend(loc='upper right')

plt.title('Training and Validation Loss')

plt.show()

# 6. EVALUATE MODEL

y\_true = []

y\_pred = []

for images, labels in val\_ds:

preds = model.predict(images)

y\_true.extend(labels.numpy())

y\_pred.extend(np.argmax(preds, axis=1))

# Confusion Matrix

cm = confusion\_matrix(y\_true, y\_pred)

plt.figure(figsize=(8, 6))

sns.heatmap(cm, annot=True, fmt='d', xticklabels=class\_names, yticklabels=class\_names)

plt.xlabel('Predicted')

plt.ylabel('True')

plt.title('Confusion Matrix')

plt.show()

# Classification Report

print("Classification Report:\n", classification\_report(y\_true, y\_pred, target\_names=class\_names))

# 7. PREDICT SAMPLE IMAGE

from tensorflow.keras.preprocessing import image

img\_path = '/content/sample\_rice.jpg' # Replace with your own image

img = image.load\_img(img\_path, target\_size=(224, 224))

img\_array = image.img\_to\_array(img)

img\_array = tf.expand\_dims(img\_array, 0) # Create batch axis

predictions = model.predict(img\_array)

predicted\_class = class\_names[np.argmax(predictions[0])]

confidence = 100 \* np.max(predictions[0])

print(f"Predicted: {predicted\_class} ({confidence:.2f}% confidence)")