from google.colab import drive

import os

import tensorflow as tf

from tensorflow.keras.preprocessing.image import ImageDataGenerator, load\_img, img\_to\_array

import numpy as np

from PIL import Image

# Step 1: Connect Google Drive

drive.mount('/content/drive')

# Step 2: Ensure Correct Dataset Path

dataset\_path = "/content/drive/MyDrive/Melanoma and BCC /melanoma and BCC dataset" # Fixed extra space issue

# Step 3: Image Preprocessing

img\_size = (224, 224) # Resize all images to 224x224

batch\_size = 32

# Initialize ImageDataGenerator with rescaling and validation split

train\_datagen = ImageDataGenerator(rescale=1./255, validation\_split=0.2)

# Load Training Images

train\_generator = train\_datagen.flow\_from\_directory(

dataset\_path,

target\_size=img\_size,

batch\_size=batch\_size,

class\_mode='binary',

subset='training',

shuffle=True # Ensure shuffling for training

)

# Load Validation Images

val\_generator = train\_datagen.flow\_from\_directory(

dataset\_path,

target\_size=img\_size,

batch\_size=batch\_size,

class\_mode='binary',

subset='validation',

shuffle=False # Keep validation order fixed

)

# \*Fix: Get Correct Class Labels\*

class\_labels = {v: k for k, v in train\_generator.class\_indices.items()} # Reverse mapping

# Dataset Overview

total\_train\_images = train\_generator.samples

total\_val\_images = val\_generator.samples

total\_images = total\_train\_images + total\_val\_images

print(f"\n \*Dataset Overview:\*")

print(f" Total Images: {total\_images}")

print(f" Training Images: {total\_train\_images}")

print(f" Validation Images: {total\_val\_images}")

print(f" Class Labels Mapping: {train\_generator.class\_indices}\n")

# Function to Display Before & After Preprocessing

def display\_preprocessing\_details(generator, dataset\_type):

print(f"\n \*Preprocessing Details for {dataset\_type} Dataset:\*")

img\_count = 0

for batch\_images, batch\_labels in generator:

for i in range(len(batch\_images)):

img = batch\_images[i] # Get preprocessed image

label = int(batch\_labels[i]) # Get assigned label

filename = generator.filenames[img\_count] # Get file name

# Construct full path to original image

original\_path = os.path.join(dataset\_path, filename)

# Ensure file exists before loading

if os.path.exists(original\_path):

original\_img = load\_img(original\_path) # Load original image

original\_img\_array = img\_to\_array(original\_img) # Convert to array

print(f"\n Image {img\_count+1}: {filename}")

# Display Before Preprocessing Details

print(f" \*Before Preprocessing:\*")

print(f" - Original Shape: {original\_img\_array.shape}") # Original shape

print(f" - Min Pixel Value: {original\_img\_array.min()}")

print(f" - Max Pixel Value: {original\_img\_array.max()}")

# Display After Preprocessing Details

print(f" \*After Preprocessing:\*")

print(f" - Resized Shape: {img.shape}") # Resized shape

print(f" - Min Pixel Value (after rescale): {img.min():.4f}")

print(f" - Max Pixel Value (after rescale): {img.max():.4f}")

print(f" - Assigned Label: {label} ({class\_labels[label]})")

else:

print(f"\n Image Not Found: {filename}")

img\_count += 1

if img\_count >= generator.samples: # Stop after processing all images

break

if img\_count >= generator.samples:

break # Stop looping batches

# Display Before & After Preprocessing for ALL Training and Validation Images

display\_preprocessing\_details(train\_generator, "Training")

display\_preprocessing\_details(val\_generator, "Validation")

print("\n \*All Images Processed Successfully!\* ")