**#IMPORT PACKAGES**

import os

import numpy as np

import matplotlib.pyplot as plt

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from sklearn.model\_selection import train\_test\_split

**#CONNECT THE GOOGLE DRIVE WITH THE DATASET**

from google.colab import drive

drive.mount('/content/drive')

dataset\_path = "/content/drive/My Drive/skindisease\_dataset"

**# PREPROCESS THE IMAGES IN THE DATASET**

img\_size = (128, 128)

batch\_size = 32

# Validation split = 20%, so training split = 80% (i.e., 1 - 0.2 = 0.8)

datagen = ImageDataGenerator(

    rescale=1.0/255,

    validation\_split=0.2

)

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

    dataset\_path,

    target\_size=img\_size,

    batch\_size=batch\_size,

    class\_mode='categorical',

    subset='training'  # 80% of data

)

val\_generator = datagen.flow\_from\_directory(

    dataset\_path,

    target\_size=img\_size,

    batch\_size=batch\_size,

    class\_mode='categorical',

    subset='validation'  # 20% of data

)

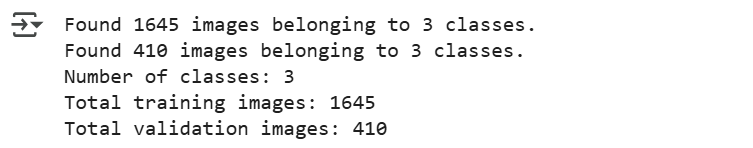
num\_classes = len(train\_generator.class\_indices)

print(f"Number of classes: {num\_classes}")

print(f"Total training images: {train\_generator.samples}")

print(f"Total validation images: {val\_generator.samples}")

**output:**



**#CHOOSE THE MODEL AND TRAIN**

model = Sequential([

    Conv2D(32, (3,3), activation='relu', input\_shape=(128, 128, 3)),

    MaxPooling2D(2,2),

    Conv2D(64, (3,3), activation='relu'),

    MaxPooling2D(2,2),

    Conv2D(128, (3,3), activation='relu'),

    MaxPooling2D(2,2),

    Flatten(),

    Dense(128, activation='relu'),

    Dropout(0.5),

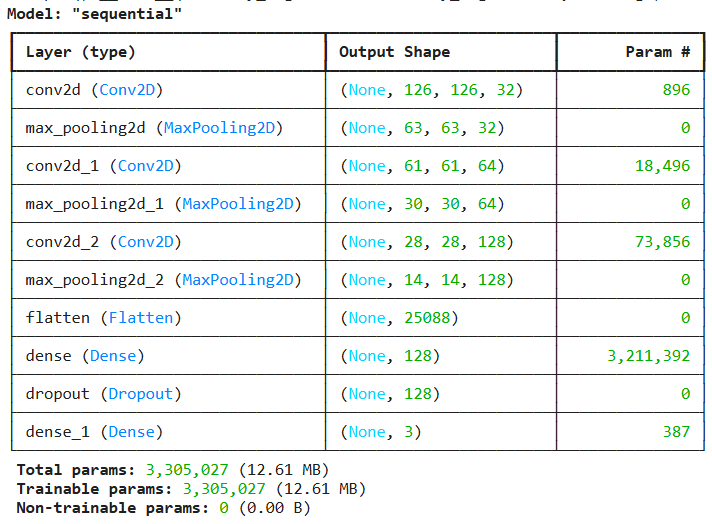
    Dense(num\_classes, activation='softmax')  # Output layer

])

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

model.summary()

**output:**



**#TRAIN THE MODEL**

epochs = 20

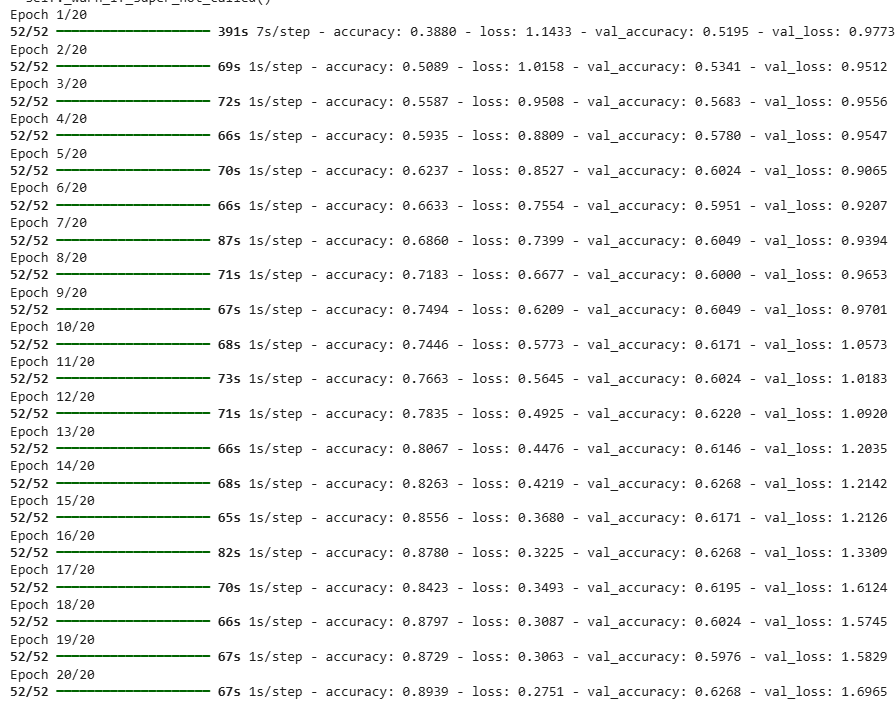
history = model.fit(

    train\_generator,

    validation\_data=val\_generator,

    epochs=epochs

)



**#PLOT THE GRAPHS**

acc = history.history['accuracy']

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

loss = history.history['loss']

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

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

plt.subplot(1, 2, 1)

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

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

plt.legend()

plt.title('Training vs Validation Accuracy')

plt.subplot(1, 2, 2)

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

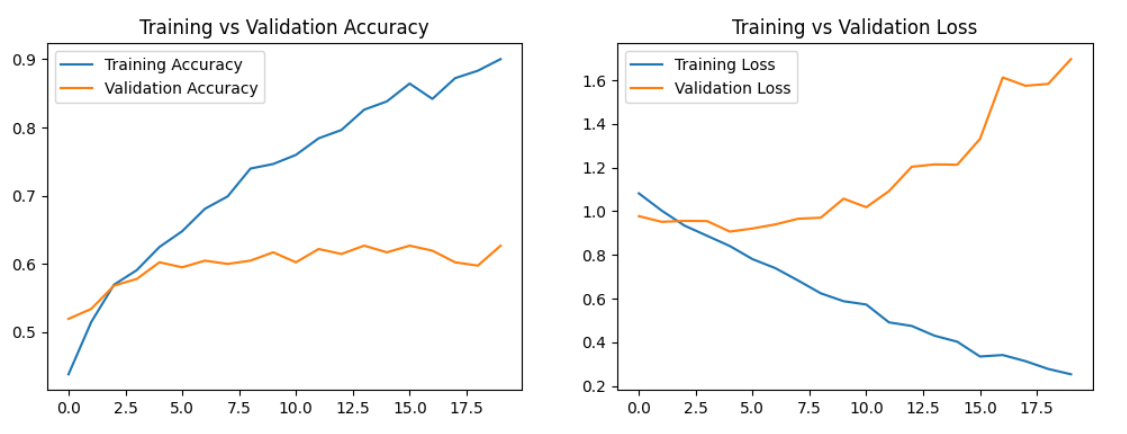
plt.plot(val\_loss, label='Validation Loss')

plt.legend()

plt.title('Training vs Validation Loss')

plt.show()

**output:**

****

**#IMPORT THE INPUT IMAGE AND PREPROCESS AND TRAIN THE IMAGE**

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

from tensorflow.keras.preprocessing.image import ImageDataGenerator

import os

dataset\_path = "/content/drive/My Drive/skindisease\_dataset"

train\_datagen = ImageDataGenerator(

    rescale=1.0/255,

    rotation\_range=20,

    width\_shift\_range=0.2,

    height\_shift\_range=0.2,

    shear\_range=0.2,

    zoom\_range=0.2,

    horizontal\_flip=True,

    validation\_split=0.2

)

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

    dataset\_path,

    target\_size=(128, 128),

    batch\_size=32,

    class\_mode="categorical",

    subset="training"

)

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

    dataset\_path,

    target\_size=(128, 128),

    batch\_size=32,

    class\_mode="categorical",

    subset="validation"

)

model = Sequential([

    Conv2D(32, (3,3), activation='relu', input\_shape=(128, 128, 3)),

    MaxPooling2D(2,2),

    Conv2D(64, (3,3), activation='relu'),

    MaxPooling2D(2,2),

    Conv2D(128, (3,3), activation='relu'),

    MaxPooling2D(2,2),

    Flatten(),

    Dense(256, activation='relu'),

    Dropout(0.5),

    Dense(len(train\_generator.class\_indices), activation='softmax')  # Output layer

])

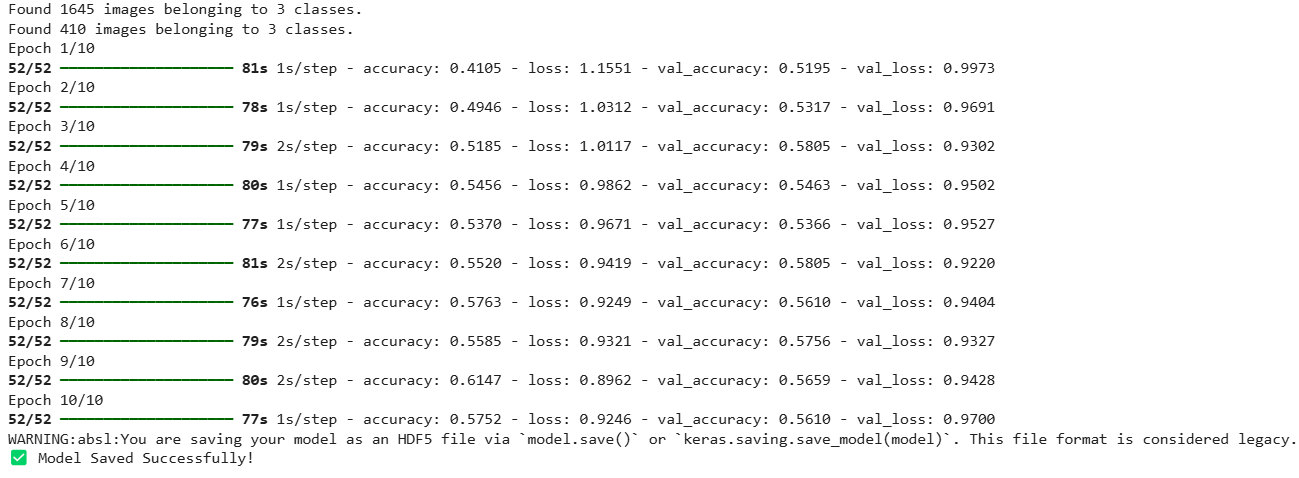
model.compile(optimizer="adam", loss="categorical\_crossentropy", metrics=["accuracy"])

history = model.fit(train\_generator, validation\_data=val\_generator, epochs=10)

model.save("skin\_disease\_model.h5")

print("Model Saved Successfully!")

**output:**

****

**#UPLOAD AND PREDICT THE IMAGE**

from google.colab import files

files.download("skin\_disease\_model.h5")

import tensorflow as tf

model = tf.keras.models.load\_model("skin\_disease\_model.h5")

print("Upload an image for prediction:")

uploaded = files.upload()

uploaded\_filename = list(uploaded.keys())[0]

model\_path = "skin\_disease\_model.h5"

model = tf.keras.models.load\_model(model\_path)

print(" Model Loaded Successfully!")

img\_size = (128, 128)

img = image.load\_img(uploaded\_filename, target\_size=img\_size)

img\_array = image.img\_to\_array(img) / 255.0

img\_array = np.expand\_dims(img\_array, axis=0)

predictions = model.predict(img\_array)

predicted\_class = np.argmax(predictions)

class\_names = list(train\_generator.class\_indices.keys())

predicted\_label = class\_names[predicted\_class]

plt.imshow(img)

plt.axis("off")

plt.title(f"Predicted: {predicted\_label}")

plt.show()

print(f"🔹 Model Prediction: {predicted\_label}")

# ✅ Evaluate Model Accuracy and Loss on Validation Data

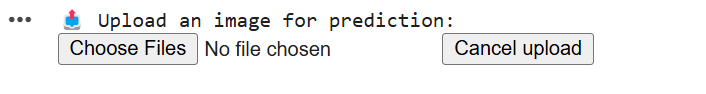
loss, accuracy = model.evaluate(val\_generator, verbose=0)

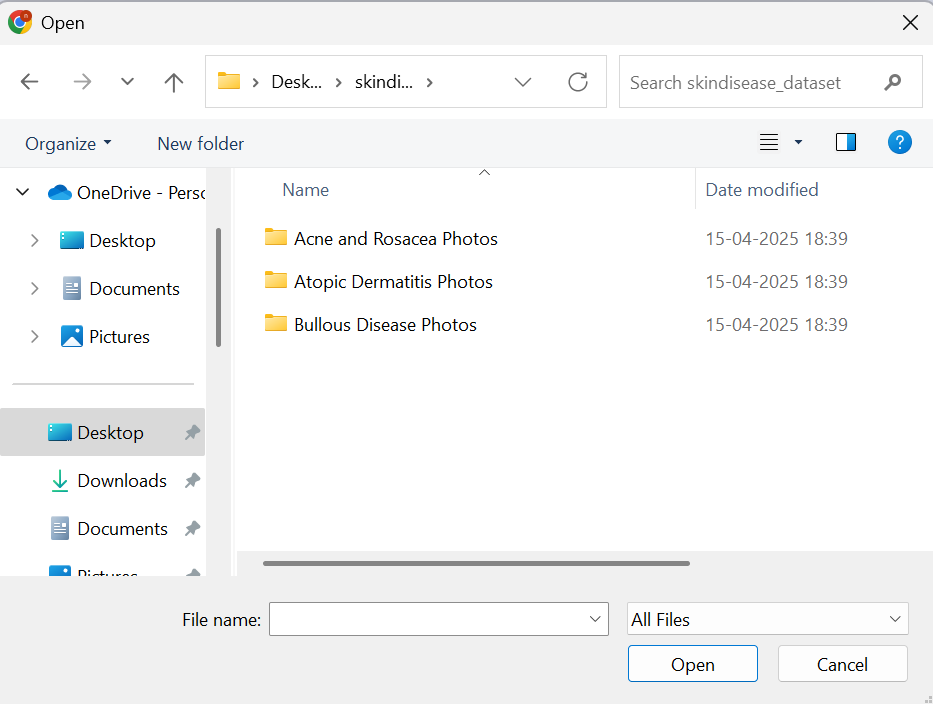
print("\n📊 Model Evaluation on Validation Data:")

print(f"✅ Accuracy: {accuracy \* 100:.2f}%")

print(f"❌ Loss: {loss:.4f}")

**OUTPUT:**

****

****

