import os

import numpy as np

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import layers

from tensorflow.keras.applications import EfficientNetV2B0

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from sklearn.utils.class\_weight import compute\_class\_weight

# Dataset Path

dataset\_path = "/content/drive/MyDrive/Melanoma and BCC /melanoma and BCC dataset"

# Image Settings

img\_size = (224, 224)

batch\_size = 32

# Data Preprocessing (Augmentation + Rescaling)

train\_datagen = ImageDataGenerator(

rescale=1./255,

validation\_split=0.2,

rotation\_range=30,

width\_shift\_range=0.3,

height\_shift\_range=0.3,

zoom\_range=0.3,

horizontal\_flip=True

)

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

dataset\_path,

target\_size=img\_size,

batch\_size=batch\_size,

class\_mode='binary',

subset='training',

shuffle=True

)

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

dataset\_path,

target\_size=img\_size,

batch\_size=batch\_size,

class\_mode='binary',

subset='validation',

shuffle=False

)

# Compute Class Weights

labels = train\_generator.classes

class\_weights = compute\_class\_weight('balanced', classes=np.unique(labels), y=labels)

class\_weights\_dict = {i: class\_weights[i] for i in range(len(class\_weights))}

# Build EfficientNetV2 Model

def build\_model():

base\_model = EfficientNetV2B0(weights='imagenet', include\_top=False, input\_shape=(224, 224, 3))

base\_model.trainable = True

model = keras.Sequential([

base\_model,

layers.GlobalAveragePooling2D(),

layers.Dense(512, activation='relu'),

layers.Dropout(0.5),

layers.Dense(1, activation='sigmoid')

])

model.compile(optimizer=keras.optimizers.Adam(learning\_rate=0.00003),

loss='binary\_crossentropy',

metrics=['accuracy'])

return model

# Initialize Model

model = build\_model()

# Training Callbacks

from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau

early\_stopping = EarlyStopping(monitor='val\_accuracy', patience=5, restore\_best\_weights=True)

lr\_scheduler = ReduceLROnPlateau(monitor='val\_loss', factor=0.5, patience=2, min\_lr=1e-7)

# Train Model

epochs = 10

history = model.fit(

train\_generator,

validation\_data=val\_generator,

epochs=epochs,

steps\_per\_epoch=len(train\_generator),

validation\_steps=len(val\_generator),

class\_weight=class\_weights\_dict,

callbacks=[early\_stopping, lr\_scheduler]

)

print("\n Model trained successfully!")

# Save Model

model.save('/content/drive/MyDrive/melanoma\_bcc\_efficientnetv2\_algorithm.optimized.h5')

print("\n Model saved successfully!")

# Display Final Accuracy

final\_train\_acc = history.history['accuracy'][-1] \* 100

final\_val\_acc = history.history['val\_accuracy'][-1] \* 100

print(f"\nFinal Training Accuracy: {final\_train\_acc:.2f}%")

print(f"Final Validation Accuracy: {final\_val\_acc:.2f}%")