import pandas as pd

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.applications.resnet import ResNet50

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

from tensorflow.keras.models import Sequential

from tensorflow.keras.optimizers import Adam

def load\_train(path):

labels = pd.read\_csv(path + 'labels.csv')

train\_datagen = ImageDataGenerator(

validation\_split=0.25,

horizontal\_flip=True,

rescale=1./255)

train\_gen\_flow = train\_datagen.flow\_from\_dataframe(

dataframe=labels,

directory=path + 'final\_files/',

x\_col='file\_name',

y\_col='real\_age',

target\_size=(224, 224),

batch\_size=32,

class\_mode='raw',

subset='training',

seed=12345)

return train\_gen\_flow

def load\_test(path):

labels = pd.read\_csv(path + 'labels.csv')

test\_datagen = ImageDataGenerator(

validation\_split=0.25,

rescale=1./255)

test\_gen\_flow = test\_datagen.flow\_from\_dataframe(

dataframe=labels,

directory=path + 'final\_files/',

x\_col='file\_name',

y\_col='real\_age',

target\_size=(224, 224),

batch\_size=32,

class\_mode='raw',

subset='validation',

seed=12345)

return test\_gen\_flow

def create\_model(input\_shape):

backbone = ResNet50(

input\_shape=input\_shape,

weights='imagenet',

include\_top=False,

)

backbone.trainable = False

model = Sequential([backbone,

GlobalAveragePooling2D(),

Flatten(),

Dense(units=40, activation='tanh'),

Dense(units=10, activation='softmax'),

Dense(1, activation='relu')])

optimizer = Adam(lr=0.01)

model.compile(

optimizer=optimizer,

loss='mse',

metrics=['mae']

)

return model

def train\_model(model, train\_data, test\_data, batch\_size=None, epochs=20,

steps\_per\_epoch=None, validation\_steps=None):

if steps\_per\_epoch is None:

steps\_per\_epoch = len(train\_data)

if validation\_steps is None:

validation\_steps = len(test\_data)

model.fit(

train\_data,

validation\_data=test\_data,

batch\_size=batch\_size,

epochs=epochs,

steps\_per\_epoch=steps\_per\_epoch,

validation\_steps=validation\_steps,

verbose=2

)

return model