import matplotlib

matplotlib.use("Agg")

from pyimagesearch.resnet import ResNet

from sklearn.preprocessing import LabelEncoder

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

from sklearn.metrics import classification\_report

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.optimizers import SGD

from tensorflow.keras.utils import to\_categorical

from imutils import paths

import matplotlib.pyplot as plt

import numpy as np

import argparse

import cv2

import os

ap = argparse.ArgumentParser()

ap.add\_argument("-d", "--dataset", required=True,

help="path to input dataset")

ap.add\_argument("-a", "--augment", type=int, default=-1,

help="whether or not 'on the fly' data augmentation should be used")

ap.add\_argument("-p", "--plot", type=str, default="plot.png",

help="path to output loss/accuracy plot")

args = vars(ap.parse\_args())

INIT\_LR = 1e-1

BS = 8

EPOCHS = 50

print("[INFO] loading images...")

imagePaths = list(paths.list\_images(args["dataset"]))

data = []

labels = []

for imagePath in imagePaths:

label = imagePath.split(os.path.sep)[-2]

image = cv2.imread(imagePath)

image = cv2.resize(image, (64, 64))

data.append(image)

labels.append(label)

data = np.array(data, dtype="float") / 255.0

le = LabelEncoder()

labels = le.fit\_transform(labels)

labels = to\_categorical(labels, 2)

(trainX, testX, trainY, testY) = train\_test\_split(data, labels,

test\_size=0.25, random\_state=42)

aug = ImageDataGenerator()

if args["augment"] > 0:

print("[INFO] performing 'on the fly' data augmentation")

aug = ImageDataGenerator(

rotation\_range=20,

zoom\_range=0.15,

width\_shift\_range=0.2,

height\_shift\_range=0.2,

shear\_range=0.15,

horizontal\_flip=True,

fill\_mode="nearest")

print("[INFO] compiling model...")

opt = SGD(lr=INIT\_LR, momentum=0.9, decay=INIT\_LR / EPOCHS)

model = ResNet.build(64, 64, 3, 2, (2, 3, 4),

(32, 64, 128, 256), reg=0.0001)

model.compile(loss="binary\_crossentropy", optimizer=opt,

metrics=["accuracy"])

print("[INFO] training network for {} epochs...".format(EPOCHS))

H = model.fit(

x=aug.flow(trainX, trainY, batch\_size=BS),

validation\_data=(testX, testY),

steps\_per\_epoch=len(trainX) // BS,

epochs=EPOCHS)

print("[INFO] evaluating network...")

predictions = model.predict(x=testX.astype("float32"), batch\_size=BS)

print(classification\_report(testY.argmax(axis=1),

predictions.argmax(axis=1), target\_names=le.classes\_))

N = np.arange(0, EPOCHS)

plt.style.use("ggplot")

plt.figure()

plt.plot(N, H.history["loss"], label="train\_loss")

plt.plot(N, H.history["val\_loss"], label="val\_loss")

plt.plot(N, H.history["accuracy"], label="train\_acc")

plt.plot(N, H.history["val\_accuracy"], label="val\_acc")

plt.title("Training Loss and Accuracy on Dataset")

plt.xlabel("Epoch #")

plt.ylabel("Loss/Accuracy")

plt.legend(loc="lower left")

plt.savefig(args["plot"])