```
# -*- coding: utf-8 -*-
"""
Created on Sat May  1 12:35:52 2021

@author: User
"""

import scipy.io
import matplotlib.pyplot as plt

import tensorflow as tf
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

import numpy as np
import numpy
import tensorflow.keras as keras
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelBinarizer
from sklearn.metrics import confusion_matrix
from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.utils import to_categorical
import seaborn as sns

from keras.preprocessing.image import ImageDataGenerator, array_to_img, img_to_array, load_img
from sklearn.model_selection import train_test_split


if __name__=="__main__":

    #import data
    test_data = scipy.io.loadmat('CAB420_Assessment_1B_Data\Data\Q1\q1_test.mat')
    train_data = scipy.io.loadmat('CAB420_Assessment_1B_Data\Data\Q1\q1_train.mat')

    # Load images and labels
    test_Y = np.array(test_data['test_Y'])
    test_X = np.array(test_data['test_X'])  /255.0

    train_Y = np.array(train_data['train_Y'])
    train_X = np.array(train_data['train_X']) /255.0

    # Check the shape of the data
    print(test_X.shape)
    print(train_X.shape)

    # Fix the axes of the images
    test_X = np.moveaxis(test_X, -1, 0)
    train_X = np.moveaxis(train_X, -1, 0)

    print(test_X.shape)
    print(train_X.shape)

    # Plot a random image and its label

    plt.imshow(train_X[350])
    plt.show()
    print(train_Y[350])
```

```
#reshape train Y to vector format
print(test_Y)

#replace 10 to 0s in ys
train_Y = np.where(train_Y==10, 0, train_Y)
test_Y = np.where(test_Y==10, 0, test_Y)

a = 5

print(test_Y[a])
print(test_Y[a+1])

def unique(list1):
    x = np.array(list1)
    print(np.unique(x))

print("unique")

unique(test_Y)

datagen = ImageDataGenerator(
                        rotation_range=10,
                        zoom_range=[0.9, 1.1],
                        height_shift_range=0.10,
                        shear_range=0.15,
                        #channel_shift_range=100,
                        #brightness_range=(0.1, 0.9)
                        )

model = keras.models.load_model('vgg_2stage_CIFAR_small.h5')
model.summary()

model.compile(loss=keras.losses.SparseCategoricalCrossentropy(from_logits=True),
          optimizer=keras.optimizers.SGD(),#(lr=1e-4, momentum=0.9),
          metrics=['accuracy'])
#model.fit_generator(datagen.flow(train_X, train_Y, batch_size=40),epochs=250)
model.fit(train_X,train_Y,batch_size = 40,epochs=50)
#model.fit(train_X, train_Y,
#      batch_size=128,
#      epochs=10,
#      validation_data=(test_X, test_Y))



def eval_model(model, x_test, y_test):
   test_scores = model.evaluate(x_test, y_test, verbose=2)
   print('Test loss:', test_scores[0])
   print('Test accuracy:', test_scores[1])

   pred = model.predict(x_test);
   indexes = tf.argmax(pred, axis=1)

   cm = confusion_matrix(y_test, indexes)
   fig = plt.figure(figsize=[20, 6])
   ax = fig.add_subplot(1, 2, 1)
   c = ConfusionMatrixDisplay(cm, display_labels=range(len(numpy.unique(y_test))))
```

```
    c.plot(ax = ax)

    ax = fig.add_subplot(1, 2, 2)
    ax.hist(y_test, bins=len(numpy.diagonal(cm)), rwidth=0.95)
    ax.plot(numpy.diagonal(cm))

eval_model(model, train_X, train_Y)
eval_model(model, test_X, test_Y)

pred = model.predict(test_X);
indexes = tf.argmax(pred, axis=1)
count = 0
print("_check_")
print(test_Y[9])
print(indexes[9].numpy())
print("____test")
for i in range(10000):
    if test_Y[i] == indexes[i].numpy():
        count = count + 1
print(count)
print(len(indexes))
print((count/len(indexes))*100)
print("____train")
pred = model.predict(train_X);
indexes = tf.argmax(pred, axis=1)
count = 0
for i in range(1000):
    if train_Y[i] == indexes[i].numpy():
        count = count + 1
print(count)
print(len(indexes))
print((count/len(indexes))*100)
```