|  |
| --- |
| from tensorflow.keras.models import Model,Sequential |
|  | from tensorflow.keras.layers import Flatten, Dense, Dropout |
|  | from tensorflow.keras.applications.resnet50 import ResNet50 |
|  | from tensorflow.keras.optimizers import Adam |
|  | from tensorflow.keras.preprocessing.image import ImageDataGenerator |
|  | import matplotlib.pyplot as plt |
|  | import time |
|  | from tensorflow.keras.callbacks import ModelCheckpoint |
|  | from tensorflow.keras.models import load\_model |
|  | t1 = time.time() |
|  |  |
|  | import tensorflow as tf |
|  |  |
|  | gpus = tf.config.experimental.list\_physical\_devices('GPU') |
|  | for gpu in gpus: |
|  | tf.config.experimental.set\_memory\_growth(gpu, True) |
|  |  |
|  | DATASET\_PATH = 'C:/Users/user/Desktop/學校/109-2/巨量課程2/Data/image\_data/' |
|  | IMAGE\_SIZE = (224, 224) |
|  | NUM\_CLASSES = 5 |
|  | BATCH\_SIZE = 16 |
|  | NUM\_EPOCHS = 10 |
|  | WEIGHTS\_FINAL = 'model-resnet50-final.h5' |
|  |  |
|  | train\_datagen = ImageDataGenerator(rotation\_range=40, |
|  | width\_shift\_range=0.2, |
|  | height\_shift\_range=0.2, |
|  | shear\_range=0.2, |
|  | zoom\_range=0.2, |
|  | channel\_shift\_range=10, |
|  | horizontal\_flip=True, |
|  | fill\_mode='nearest') |
|  |  |
|  | train\_batches = train\_datagen.flow\_from\_directory(DATASET\_PATH + '/train', |
|  | target\_size=IMAGE\_SIZE, |
|  | interpolation='bicubic', |
|  | class\_mode='categorical', |
|  | shuffle=True, |
|  | batch\_size=BATCH\_SIZE) |
|  |  |
|  | valid\_datagen = ImageDataGenerator() |
|  | valid\_batches = valid\_datagen.flow\_from\_directory(DATASET\_PATH + '/validation', |
|  | target\_size=IMAGE\_SIZE, |
|  | interpolation='bicubic', |
|  | class\_mode='categorical', |
|  | shuffle=False, |
|  | batch\_size=BATCH\_SIZE) |
|  |  |
|  | # for cls, idx in train\_batches.class\_indices.items(): |
|  | # print('Class #{} = {}'.format(idx, cls)) |
|  |  |
|  | rest\_layer = ResNet50(include\_top=False, weights='imagenet', |
|  | input\_shape=(224,224,3)) |
|  |  |
|  | Res\_model = Sequential() |
|  | Res\_model.add(rest\_layer) |
|  | Res\_model.layers[0].trainable = False |
|  | Res\_model.add(Flatten()) |
|  | Res\_model.add(Dropout(0.5)) |
|  | Res\_model.add(Dense(5, activation='softmax', name='prediction')) |
|  |  |
|  | Res\_model.compile(optimizer=Adam(learning\_rate=1e-5), |
|  | loss='categorical\_crossentropy', metrics=['accuracy']) |
|  | # print(net\_final.summary()) |
|  | best\_model\_path = 'TL\_best\_model.h5' |
|  | checkpoint\_callback = ModelCheckpoint( |
|  | best\_model\_path, |
|  | monitor='val\_accuracy', |
|  | save\_best\_only=True |
|  | ) |
|  |  |
|  | history\_res = Res\_model.fit\_generator(train\_batches, |
|  | steps\_per\_epoch = train\_batches.samples // BATCH\_SIZE, |
|  | validation\_data = valid\_batches, |
|  | validation\_steps = valid\_batches.samples // BATCH\_SIZE, |
|  | callbacks=[checkpoint\_callback], |
|  | epochs = NUM\_EPOCHS) |
|  | history\_dict = history\_res.history |
|  | loss\_values = history\_dict['loss'] |
|  | val\_loss\_values = history\_dict['val\_loss'] |
|  | epochs = range(1, len(loss\_values) + 1) |
|  | plt.plot(epochs, loss\_values, 'bo-', label='Training loss') |
|  | plt.plot(epochs, val\_loss\_values, 'r--', label='validation loss') |
|  | plt.xlabel('Epochs') |
|  | plt.ylabel('Loss') |
|  | plt.legend(loc='upper right') |
|  | plt.show() |
|  |  |
|  | acc\_values = history\_dict['accuracy'] |
|  | val\_acc\_values = history\_dict['val\_accuracy'] |
|  | plt.plot(epochs, acc\_values, 'b', label="Training accuracy") |
|  | plt.plot(epochs, val\_acc\_values, 'r', label="Validation accuracy") |
|  | plt.xlabel('Epochs') |
|  | plt.ylabel('Accuracy') |
|  | plt.legend(loc='lower right') |
|  | plt.show() |
|  |  |
|  | t2 = time.time() |
|  | print('time elapsed: ' + str(round(t2-t1, 2)) + ' seconds') |
|  |  |
|  | Res\_model.load\_weights(best\_model\_path) |
|  | Res\_model.save(WEIGHTS\_FINAL) |
|  |  |
|  | from tensorflow.keras.models import load\_model |
|  | from tensorflow.keras.preprocessing import image |
|  | import numpy as np |
|  | import glob |
|  |  |
|  | import tensorflow as tf |
|  |  |
|  | gpus = tf.config.experimental.list\_physical\_devices('GPU') |
|  | for gpu in gpus: |
|  | tf.config.experimental.set\_memory\_growth(gpu, True) |
|  |  |
|  | file\_path = 'C:/Users/user/Desktop/學校/109-2/巨量課程2/Data/image\_data/test/' |
|  | f\_names = glob.glob(file\_path + '\*.jpg') |
|  |  |
|  | net = load\_model('model-resnet50-final.h5') #load model |
|  | # net = load\_model('model-VGG16-final.h5') |
|  | # net = load\_model('model-ResNet101V2-final.h5') |
|  |  |
|  | cls\_list = ['daisy', 'dandelion','rose','sunflower','tulip'] |
|  |  |
|  | #prediction |
|  | for f in range(10): |
|  | img = image.load\_img(f\_names[f], target\_size=(224, 224)) |
|  | x = image.img\_to\_array(img) |
|  | x = np.expand\_dims(x, axis=0) |
|  | pred = net.predict(x)[0] |
|  | top\_inds = pred.argsort()[::-1] |
|  | print(f\_names[f]) |
|  | for i in top\_inds: |
|  | print('{:.3f} {}'.format(pred[i], cls\_list[i])) |