**Final Programming Exam. (part 2)**

**Deep Learning, 2022**

姓名：吳佳展 學號：110318517

import numpy as np

import pandas as pd

import os

import matplotlib.pyplot as plt

os.environ["CUDA\_VISIBLE\_DEVICES"] = "2"

from tensorflow import keras

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.models import Model

from tensorflow.keras.layers import (Dense, Dropout, Activation,

                                     Flatten, GlobalAveragePooling2D)

import tensorflow as tf

data\_path = 'Data\_2/train/'

x\_data\_list = []

y\_data\_list = []

for roots, dirs, files in os.walk(data\_path):

    for each in files:

        if each.find('checkpoint') == -1:

            x\_data\_list.append(os.path.join(roots.split("/")[-1], each))

            y\_data\_list.append(roots.split("/")[-1])

            data\_list = pd.DataFrame({})

data\_list['img\_path'] = x\_data\_list

data\_list['label'] = y\_data\_list

data\_list['label'] = data\_list['label'].astype('string')

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

train\_list, test\_list = train\_test\_split(data\_list,

                                         test\_size=0.1,

                                         random\_state=40,

                                         stratify=data\_list['label'].values)

img\_size = 256

num\_class = len(data\_list['label'].unique())

from tensorflow.keras.applications.resnet import (ResNet50,ResNet101, preprocess\_input)

resnet\_model = ResNet101(weights='imagenet', include\_top=False,

                     input\_shape=(img\_size, img\_size, 3))

x = GlobalAveragePooling2D()(resnet\_model.output)

x = Dropout(0.27)(x)

outputs = Dense(num\_class, activation='softmax')(x)

model = Model(inputs=resnet\_model.inputs, outputs=outputs)

model.summary()

learning\_rate = 1e-5

optimizer = keras.optimizers.Adam(learning\_rate=learning\_rate)

model.compile(loss='categorical\_crossentropy',

              optimizer=optimizer,

              metrics=['accuracy'])

batch\_size = 32

num\_steps = len(train\_list) // batch\_size + 1

num\_epochs = 100

train\_datagen = ImageDataGenerator(preprocessing\_function=preprocess\_input)

test\_datagen = ImageDataGenerator(preprocessing\_function=preprocess\_input)

img\_shape = (img\_size, img\_size)

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

                                                    dataframe=train\_list,

                                                    directory=data\_path,

                                                    x\_col="img\_path",

                                                    y\_col="label",

                                                    target\_size=img\_shape,

                                                    batch\_size=batch\_size,

                                                    class\_mode='categorical')

valid\_generator = test\_datagen.flow\_from\_dataframe(

                                                    dataframe=test\_list,

                                                    directory=data\_path,

                                                    x\_col="img\_path",

                                                    y\_col="label",

                                                    target\_size=img\_shape,

                                                    batch\_size=batch\_size,

                                                    class\_mode='categorical',

                                                    shuffle=False)

model\_dir = 'Q2\_model/'

if not os.path.exists(model\_dir):

    os.makedirs(model\_dir)

# logfiles = model\_dir + '/{}-{}'.format('basic\_model',

#                                        model.\_\_class\_\_.\_\_name\_\_)

# model\_cbk = keras.callbacks.TensorBoard(log\_dir=logfiles,

#                                         histogram\_freq=1)

modelfiles = model\_dir + 'vbn.h5'

model\_mckp = keras.callbacks.ModelCheckpoint(modelfiles,

                                             monitor='val\_accuracy',

                                             save\_best\_only=True)

earlystop = keras.callbacks.EarlyStopping(monitor='val\_loss',

                                          patience=20,

                                          verbose=1)

# callbacks\_list = [model\_cbk, model\_mckp, earlystop]

callbacks\_list = [model\_mckp, earlystop]

history = model.fit\_generator(train\_generator,

                              steps\_per\_epoch=num\_steps,

                              epochs=num\_epochs,

                              validation\_data=valid\_generator,

                              callbacks=callbacks\_list)

loss, acc = model.evaluate\_generator(valid\_generator, verbose=2)

train\_history = ['loss', 'val\_loss', 'accuracy', 'val\_accuracy']

name\_history = ['training\_loss', 'val\_loss', 'training\_acc', 'val\_acc']

plt.figure(figsize=(10, 6))

for eachx, eachy, i in zip(train\_history, name\_history, range(4)):

    if i % 2 == 0:

        plt.subplot(1, 2, i//2+1)

    l\_x = len(history.history[eachx])

    plt.plot(np.arange(l\_x), history.history[eachx], label=eachy)

    plt.legend(loc='best')

    plt.title('model'+eachy)

plt.savefig(f'Q2\_model/type2.png')

plt.show()