Final Programming Exam. (part 2)
Deep Learning, 2022

姓名：吳佳展　　學號：110318517

此題的答案的整體架構為幾大步驟分別為：a.讀取資料 b.切分 train data 和 test data c.設計模型架構 d.訓練模型 e.劃出訓練模型的 loss f.使用模型進行產出預測圖片分類 g.將其測試集資料和分類值合併匯出 csv 檔案，詳細步驟下方將會逐一說明各項過程。

整體架構圖：

1. 第 1 步先讀取各個資料夾的檔案照片當作 data。

```
# read data
data_path = "Data_1/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])
```

2. 第 2 步驟分割成 train data 和 test data

```
train_list, test_list = train_test_split(data_list,
                                         test_size = 0.1,
                                         random_state = 30,
                                         stratify = data_list['label'].values)
```

3. 第三步驟設計 imgsize 大小並 create model。

```
# create model
img_size, Sequential, num_class = 224, 123, len(data_list['label'].unique())
vgg_model = VGG16(weights='imagenet', include_top=False,
                  input_shape=(img_size, img_size, 3))

x = Flatten()(vgg_model.output)
x = Dropout(0.2)(x)
outputs = Dense(num_class, activation = 'softmax')(x)
model = Model(inputs = vgg_model.inputs, outputs=outputs)
model.summary()

vgg_model.trainable = False
vgg_model.trainable = True

trainable_layer = 3
for layer in vgg_model.layers[:-trainable_layer]:
    layer.trainable = False

# for layer in model.layers:
#     print(layer, layer.trainable)
learning_rate = 1e-3
opti = tf.keras.optimizers.Adam(learning_rate = 1e-3)
model.compile(loss='categorical_crossentropy', optimizer = opti, metrics = ['accuracy'])
```

模型架構及參數如下圖所示

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
dropout (Dropout)            (None, 25088)             0
_________________________________________________________________
dense (Dense)                (None, 4)                 100356
=================================================================
Total params: 14,815,044
Trainable params: 14,815,044
```

4. 第四步驟將 TRAINING 資料放入模型訓練，並將模型存取，而模型訓練的參數如下表所示。

參數名稱	Loss	Optimizer	Epochs	Batch_size
所使用參數	categorical_crossentropy	adam	30	32

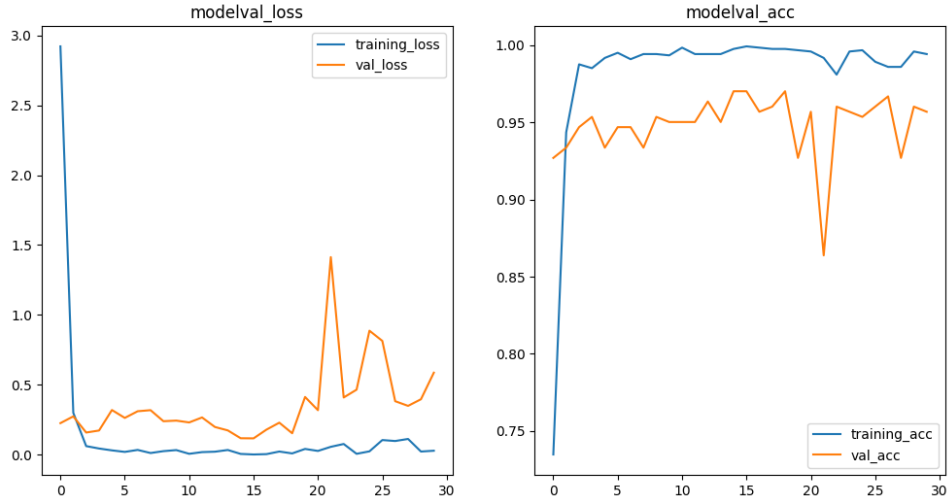
```
modelfiles = f'{model_dir}/animal.h5'
model_mckp = ModelCheckpoint(modelfiles,monitor='val_accuracy',save_best_only=True)
earlystop = EarlyStopping(monitor='val_loss',patience=10,verbose=1)
callbacks_list = [model_mckp, earlystop]
history = model.fit(train_generator,
                    steps_per_epoch = num_steps,
                    epochs = num_epochs,
                    validation_data = valid_generator,
                    callbacks = callbacks_list)
loss, acc = model.evaluate_generator(valid_generator, verbose=2)
```

```
logfiles = model_dir + '/{}-{}'.format('basic_model', model.__class__.__name__)
modelfiles = f'{model_dir}/vbn.h5'
model_cbk = TensorBoard(log_dir=logfiles, histogram_freq=1)
model_mckp = ModelCheckpoint(modelfiles, monitor='val_accuracy', save_best_only=True)
earlystop = EarlyStopping(monitor='val_loss',patience=5, verbose=1)
callbacks_list = [model_cbk, model_mckp, earlystop]
history = model.fit(train_generator, steps_per_epoch=num_steps,epochs=num_epochs,
                    validation_data=valid_generator,
                    callbacks=callbacks_list)
```

5. 第五步驟為將所訓練過程的 learn error 用圖表所示，判斷是否有 overfitting 或是 underfitting 的狀態。

```
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']

# plot train model loss
plt.figure(figsize=(11, 6))
for each_x, each_y, 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[each_x])
    plt.plot(np.arange(l_x), history.history[each_x], label=each_y)
    plt.legend(loc='best')
    plt.title('model'+ each_y)
plt.savefig(f'{model_dir}/animals_loss.png')
plt.show()
```



6. 第六步驟將 test_data 使用 test.py 進行預測，進而產生出預測分類。

```
model = tf.keras.models.load_model('animal.h5')
y_pred_all = model.predict_generator(test_generator)
y_pred_all = y_pred_all.argmax(-1)
labels = {0 : 'buffalo', 1 : 'elephant' ,2 : 'rhino',3 : 'zebra'}
predictions = [labels[k] for k in y_pred_all]
```

7. 第七步驟為將 test_data 和預測值組合起來輸出成“test2.csv”。

```
# export csv
df_new = pd.DataFrame()
df_new['img_path'] = x_data_list
df_new['predecict'] = predictions
df_new.to_csv('test2.csv', index=False)
```