LAB5-Resnet18

國立雲林科技大學
夏世昌 特聘教授 / 電子工程系
王斯弘 助理教授 / 前瞻學位學士學程

2022, Fall Semester

訓練開始

START TRAINING!!

Import library

```
import tensorflow as tf
tf.__version__
```

```
'1.15.0'
```

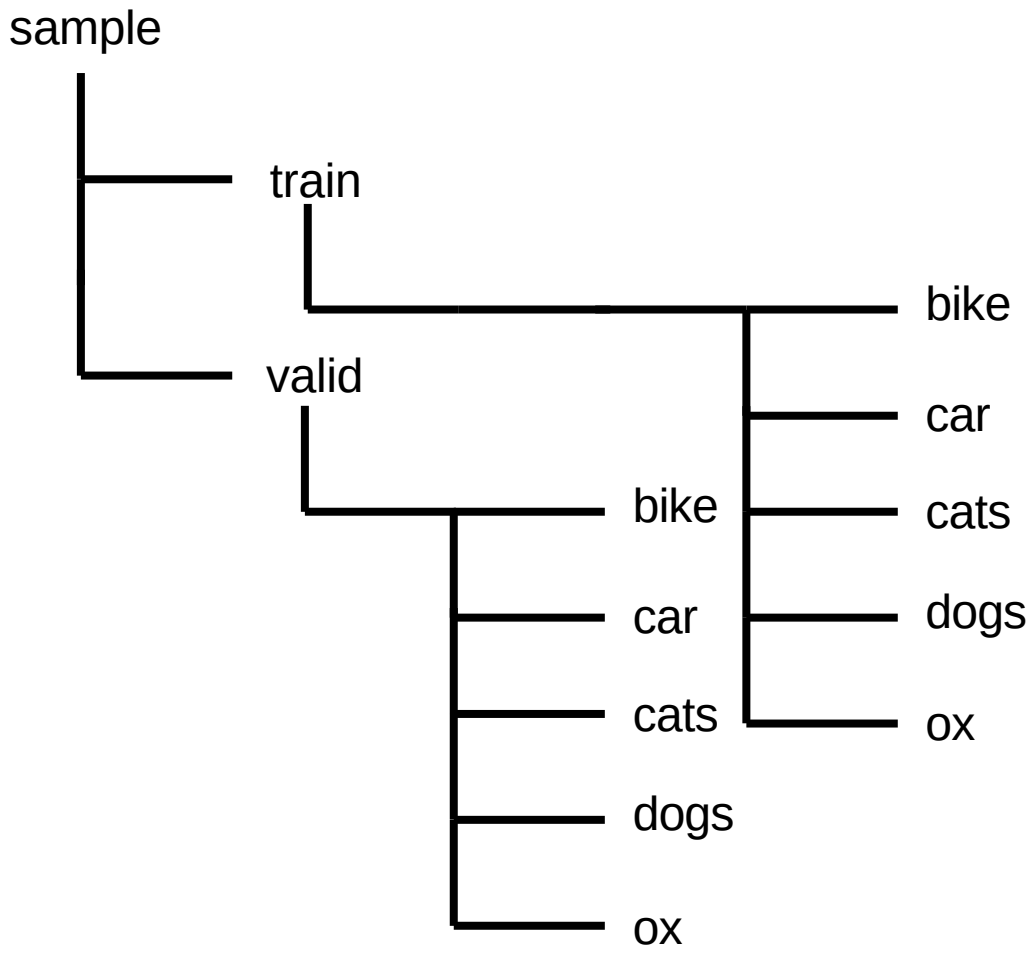
Import library

```
from tensorflow.keras import backend as K
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Flatten,Dense,Dropout,Activation,BatchNormalization,ReLU,add
from tensorflow.keras.layers import Conv2D,MaxPooling2D,Input,GlobalAveragePooling2D
from tensorflow.keras.optimizers import Adam,SGD
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from matplotlib import pyplot as plt
from tensorflow.keras.utils import plot_model
from tensorflow.keras import optimizers, regularizers
from tensorflow.keras.callbacks import CSVLogger ,EarlyStopping ,ReduceLROnPlateau
```

參數設定

```
DATASET_PATH = 'sample'

IMAGE_SIZE = (224,224)

NUM_CLASSES = 5

BATCH_SIZE = 256

# Epoch 數
NUM_EPOCHS = 60

# 模型輸出儲存的檔案
WEIGHTS_FINAL = 'model-resnet18.h5'
```


dataset


sample
train
valid
bike
car
cats
dogs
ox
bike
car
cats
dogs
ox

dataset

課程有幫各位準備資料集的圖片
請各位前往雲端下載

dataset

```
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)
```

dataset

```
valid_datagen = ImageDataGenerator()
valid_batches = valid_datagen.flow_from_directory(DATASET_PATH + '/valid',
                                                  target_size=IMAGE_SIZE,
                                                  interpolation='bicubic',
                                                  class_mode='categorical',
                                                  shuffle=False,
                                                  batch_size=BATCH_SIZE)
```

dataset

```
for cls, idx in train_batches.class_indices.items():
    print('Class #{} = {}'.format(idx, cls))
```

```
Class #0 = bike
Class #1 = car
Class #2 = cats
Class #3 = dogs
Class #4 = ox
```

Model

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

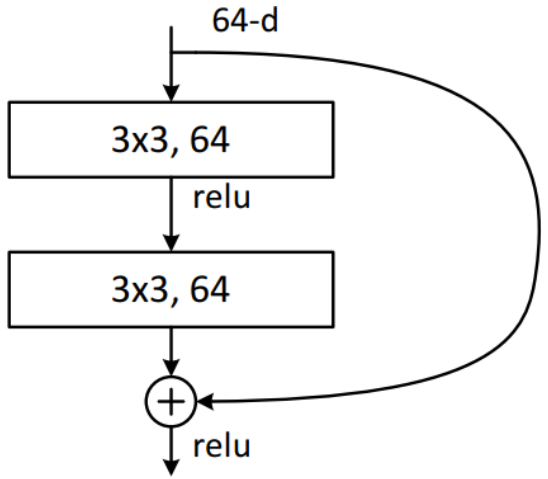
ures for ImageNet. Building blocks are shown in brackets (see also Fig. 5), with the numbers of block

定義Model

```
def resnet18(x):
    x = Conv2D(filters=64,kernel_size=(7,7),strides=(2, 2),padding='same')(x)
    x = MaxPooling2D()(x)
    x = block(x,out_filters=64,downsample=0)
    x = block(x,out_filters=64,downsample=0)
    x = block(x,out_filters=128,downsample=1)
    x = block(x,out_filters=128,downsample=0)
    x = block(x,out_filters=256,downsample=1)
    x = block(x,out_filters=256,downsample=0)
    x = block(x,out_filters=512,downsample=1)
    x = block(x,out_filters=512,downsample=0)
    x = GlobalAveragePooling2D()(x)
    x = Dense(5,activation='softmax')(x)
    return x
```

定義block

```
def block(x,out_filters,k_size=(3,3),downsample=0):
    if(downsample==1):
        x1 = Conv2D(filters=out_filters,kernel_size=(1,1),strides=(2, 2),padding='same')(x)
        x=Conv2D(filters=out_filters,kernel_size=k_size,strides=(2, 2),padding='same')(x)
    else:
        x1 = x
        x=Conv2D(filters=out_filters,kernel_size=k_size,strides=(1, 1),padding='same')(x)
    x = BatchNormalization()(x)
    x = ReLU()(x)
    x = Conv2D(filters=out_filters,kernel_size=k_size,strides=(1, 1),padding='same')(x)
    x = BatchNormalization()(x)
    x = add([x1,x])
    x = ReLU()(x)
    return x
```



定義Model架構

```
img_input = Input(shape=(IMAGE_SIZE[0],IMAGE_SIZE[1],3))
output = resnet18(img_input)
model = Model(img_input,output)
print(model.summary())
```

可視化Model架構

```
print(model.summary())
```

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 224, 224, 3)	0	
conv2d (Conv2D)	(None, 112, 112, 64)	9472	input_1[0][0]
max_pooling2d (MaxPooling2D)	(None, 56, 56, 64)	0	conv2d[0][0]
conv2d_1 (Conv2D)	(None, 56, 56, 64)	36928	max_pooling2d[0][0]
batch_normalization (BatchNorma	(None, 56, 56, 64)	256	conv2d_1[0][0]
re_lu (ReLU)	(None, 56, 56, 64)	0	batch_normalization[0][0]
conv2d_2 (Conv2D)	(None, 56, 56, 64)	36928	re_lu[0][0]
batch_normalization_1 (BatchNor	(None, 56, 56, 64)	256	conv2d_2[0][0]
add (Add)	(None, 56, 56, 64)	0	max_pooling2d[0][0] batch_normalization_1[0][0]
re_lu_1 (ReLU)	(None, 56, 56, 64)	0	add[0][0]
conv2d_3 (Conv2D)	(None, 56, 56, 64)	36928	re_lu_1[0][0]
batch_normalization_2 (BatchNor	(None, 56, 56, 64)	256	conv2d_3[0][0]
re_lu_2 (ReLU)	(None, 56, 56, 64)	0	batch_normalization_2[0][0]
conv2d_4 (Conv2D)	(None, 56, 56, 64)	36928	re_lu_2[0][0]
batch_normalization_3 (BatchNor	(None, 56, 56, 64)	256	conv2d_4[0][0]
add_1 (Add)	(None, 56, 56, 64)	0	re_lu_1[0][0] batch_normalization_3[0][0]
re_lu_3 (ReLU)	(None, 56, 56, 64)	0	add_1[0][0]
conv2d_6 (Conv2D)	(None, 28, 28, 128)	73856	re_lu_3[0][0]
batch_normalization_4 (BatchNor	(None, 28, 28, 128)	512	conv2d_6[0][0]

可視化Model架構

```
print(model.summary())
```

add_5 (Add)	(None, 14, 14, 256)	0	re_lu_9[0][0] batch_normalization_11[0][0]
re_lu_11 (ReLU)	(None, 14, 14, 256)	0	add_5[0][0]
conv2d_16 (Conv2D)	(None, 7, 7, 512)	1180160	re_lu_11[0][0]
batch_normalization_12 (BatchNo	(None, 7, 7, 512)	2048	conv2d_16[0][0]
re_lu_12 (ReLU)	(None, 7, 7, 512)	0	batch_normalization_12[0][0]
conv2d_17 (Conv2D)	(None, 7, 7, 512)	2359808	re_lu_12[0][0]
conv2d_15 (Conv2D)	(None, 7, 7, 512)	131584	re_lu_11[0][0]
batch_normalization_13 (BatchNo	(None, 7, 7, 512)	2048	conv2d_17[0][0]
add_6 (Add)	(None, 7, 7, 512)	0	conv2d_15[0][0] batch_normalization_13[0][0]
re_lu_13 (ReLU)	(None, 7, 7, 512)	0	add_6[0][0]
conv2d_18 (Conv2D)	(None, 7, 7, 512)	2359808	re_lu_13[0][0]
batch_normalization_14 (BatchNo	(None, 7, 7, 512)	2048	conv2d_18[0][0]
re_lu_14 (ReLU)	(None, 7, 7, 512)	0	batch_normalization_14[0][0]
conv2d_19 (Conv2D)	(None, 7, 7, 512)	2359808	re_lu_14[0][0]
batch_normalization_15 (BatchNo	(None, 7, 7, 512)	2048	conv2d_19[0][0]
add_7 (Add)	(None, 7, 7, 512)	0	re_lu_13[0][0] batch_normalization_15[0][0]
re_lu_15 (ReLU)	(None, 7, 7, 512)	0	add_7[0][0]
global_average_pooling2d (Globa	(None, 512)	0	re_lu_15[0][0]
dense (Dense)	(None, 5)	2565	global_average_pooling2d[0][0]

```
Total params: 11,189,637
Trainable params: 11,181,957
Non-trainable params: 7,680
```

LOSS與優化器

```
model.compile(optimizer=Adam(lr=0.001),
                    loss='categorical_crossentropy', metrics=['acc'])
```

callback

```
csv_logger = CSVLogger('training.csv')
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2,
                              patience=10, min_lr=1e-7)
earlystop = EarlyStopping(monitor = 'val_acc', patience = 30, verbose = 1)
cbks = [csv_logger,reduce_lr]
```

訓練

```
history=model.fit_generator(train_batches,
                            steps_per_epoch = train_batches.samples // BATCH_SIZE,
                            validation_data = valid_batches,
                            validation_steps = valid_batches.samples // BATCH_SIZE,
                            epochs = NUM_EPOCHS,
                            callbacks = cbks
                           )
```

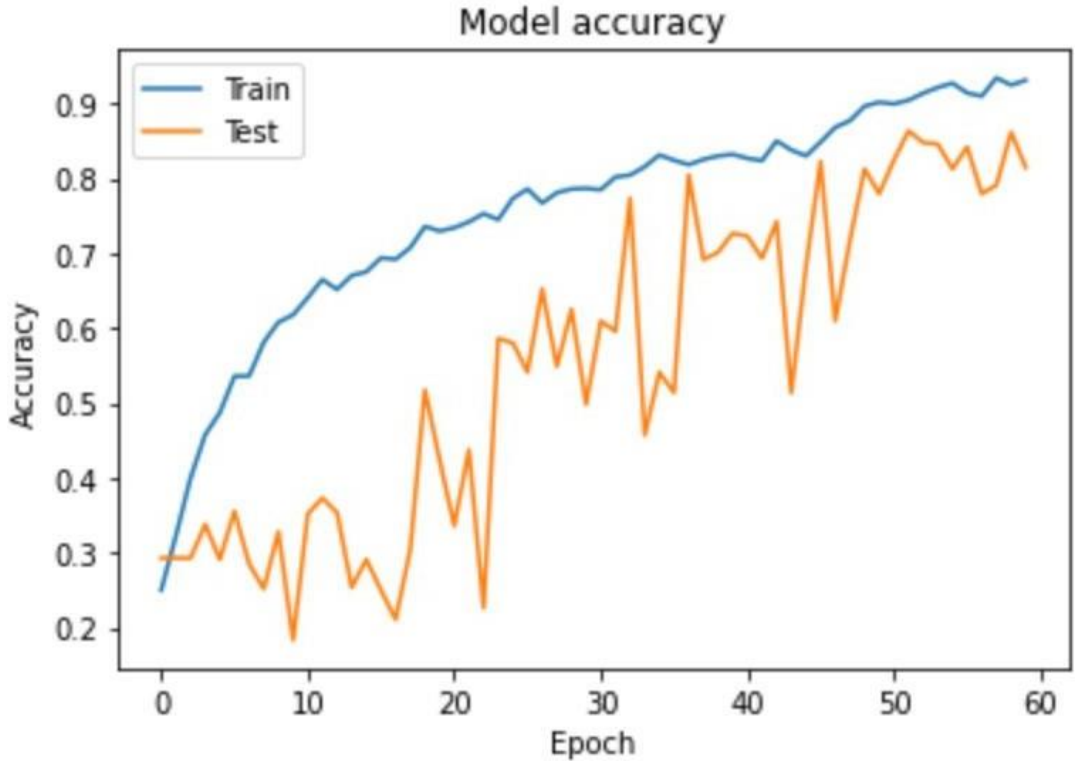
成果

```
9/9 [==============================] - 27s 3s/step - loss: 0.3319 - acc: 0.8768 - val_loss: 1.0301 - val_acc: 0.7168
Epoch 49/60
8/9 [=========================>....] - ETA: 2s - loss: 0.2967 - acc: 0.8955Epoch 1/60
9/9 [==============================] - 28s 3s/step - loss: 0.2956 - acc: 0.8963 - val_loss: 0.4999 - val_acc: 0.8125
Epoch 50/60
8/9 [=========================>....] - ETA: 2s - loss: 0.2589 - acc: 0.9054Epoch 1/60
9/9 [==============================] - 27s 3s/step - loss: 0.2686 - acc: 0.9015 - val_loss: 0.6908 - val_acc: 0.7793
Epoch 51/60
8/9 [=========================>....] - ETA: 2s - loss: 0.2764 - acc: 0.8999Epoch 1/60
9/9 [==============================] - 28s 3s/step - loss: 0.2756 - acc: 0.8993 - val_loss: 0.5237 - val_acc: 0.8242
Epoch 52/60
8/9 [=========================>....] - ETA: 2s - loss: 0.2563 - acc: 0.9029Epoch 1/60
9/9 [==============================] - 28s 3s/step - loss: 0.2532 - acc: 0.9042 - val_loss: 0.3718 - val_acc: 0.8633
Epoch 53/60
8/9 [=========================>....] - ETA: 2s - loss: 0.2434 - acc: 0.9136Epoch 1/60
9/9 [==============================] - 29s 3s/step - loss: 0.2418 - acc: 0.9136 - val_loss: 0.4665 - val_acc: 0.8477
Epoch 54/60
8/9 [=========================>....] - ETA: 2s - loss: 0.2276 - acc: 0.9217Epoch 1/60
9/9 [==============================] - 27s 3s/step - loss: 0.2286 - acc: 0.9212 - val_loss: 0.4609 - val_acc: 0.8457
Epoch 55/60
8/9 [=========================>....] - ETA: 2s - loss: 0.2026 - acc: 0.9251Epoch 1/60
9/9 [==============================] - 28s 3s/step - loss: 0.1991 - acc: 0.9269 - val_loss: 0.5399 - val_acc: 0.8125
Epoch 56/60
8/9 [=========================>....] - ETA: 2s - loss: 0.2328 - acc: 0.9146Epoch 1/60
9/9 [==============================] - 27s 3s/step - loss: 0.2329 - acc: 0.9140 - val_loss: 0.4287 - val_acc: 0.8418
Epoch 57/60
8/9 [=========================>....] - ETA: 2s - loss: 0.2330 - acc: 0.9102Epoch 1/60
9/9 [==============================] - 29s 3s/step - loss: 0.2387 - acc: 0.9097 - val_loss: 0.6278 - val_acc: 0.7793
Epoch 58/60
8/9 [=========================>....] - ETA: 2s - loss: 0.1942 - acc: 0.9372Epoch 1/60
9/9 [==============================] - 27s 3s/step - loss: 0.1970 - acc: 0.9341 - val_loss: 0.6287 - val_acc: 0.7910
Epoch 59/60
8/9 [=========================>....] - ETA: 2s - loss: 0.1971 - acc: 0.9268Epoch 1/60
9/9 [==============================] - 28s 3s/step - loss: 0.2083 - acc: 0.9245 - val_loss: 0.4578 - val_acc: 0.8613
Epoch 60/60
8/9 [=========================>....] - ETA: 2s - loss: 0.1872 - acc: 0.9331Epoch 1/60
9/9 [==============================] - 28s 3s/step - loss: 0.1949 - acc: 0.9309 - val_loss: 0.5815 - val_acc: 0.8145
```


成果圖

```python
import matplotlib.pyplot as plt


plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```

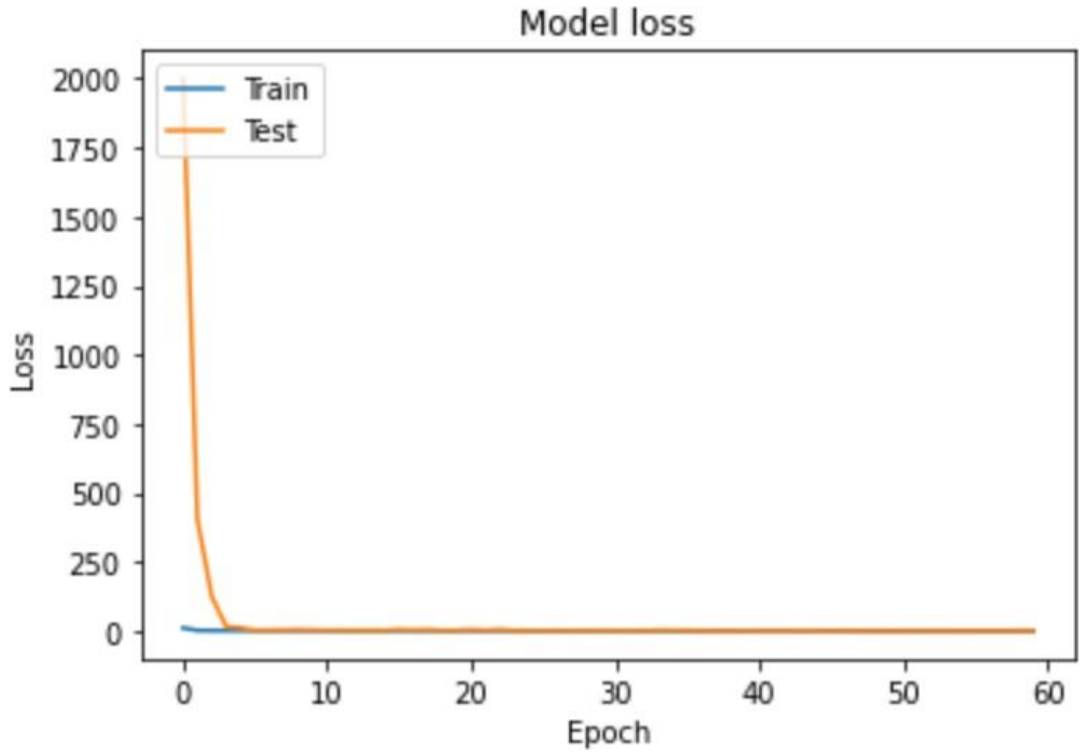
成果圖


Model accuracy
Train
Test
Accuracy
Epoch
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
0
10
20
30
40
50
60

成果圖

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```

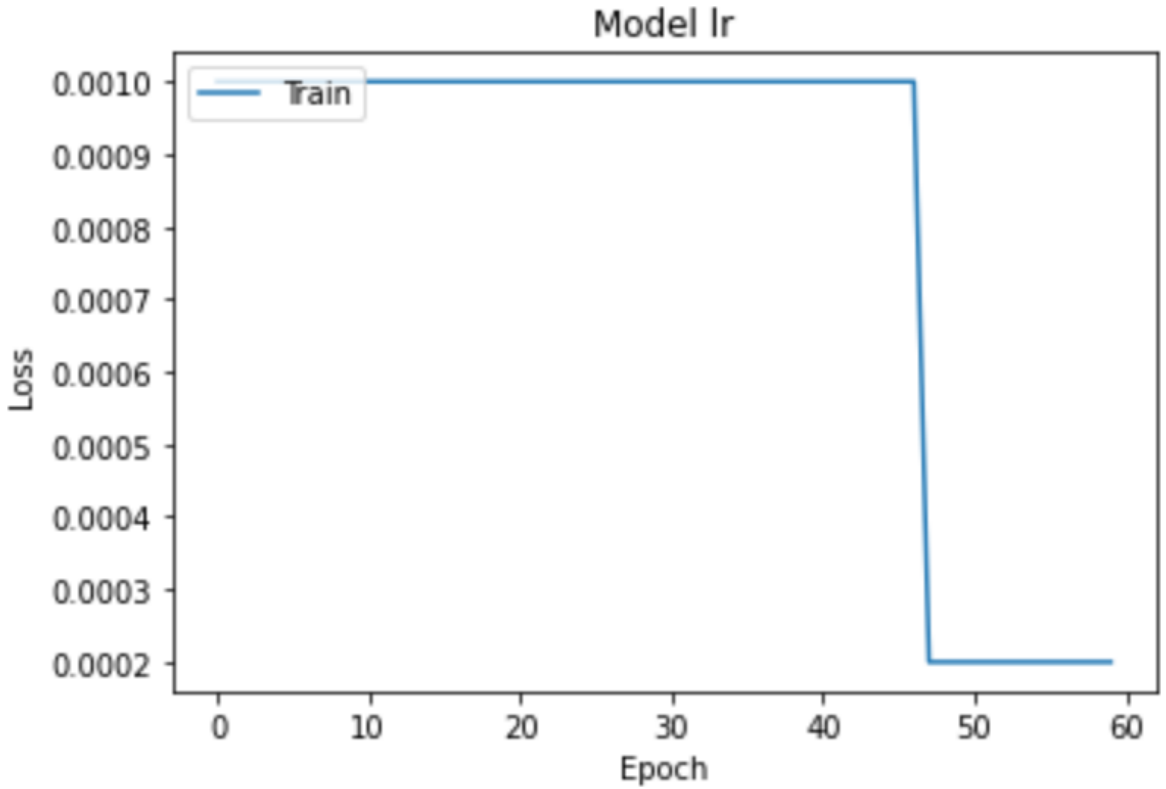
成果圖


Model loss
Train
Test
Loss
Epoch
2000
1750
1500
1250
1000
750
500
250
0
0
10
20
30
40
50
60

成果圖

```
plt.plot(history.history['lr'])
plt.title('Model lr')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train'],loc='upper left')
plt.show()
```

成果圖


Model lr
Train
Loss
0.0010
0.0009
0.0008
0.0007
0.0006
0.0005
0.0004
0.0003
0.0002
0
10
20
30
40
50
60
Epoch

儲存model

```
model.save(WEIGHTS_FINAL)
```

model-resnet18.h5 21 hours ago

Inference

因為有儲存model，所以請打開一個新的檔案開始Inference。

START！

預測

import

```
from tensorflow.python.keras import backend as K
from tensorflow.python.keras.models import load_model
from tensorflow.python.keras.preprocessing import image
import sys
import numpy as np
import glob
```

預測

載入model

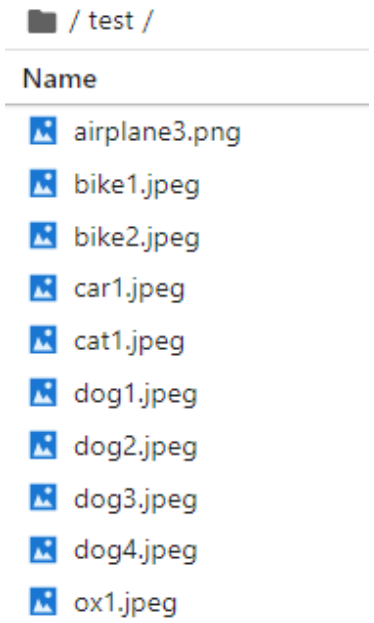
```
model = load_model('model-resnet18.h5')
```

預測

設定預測圖片路徑

```
files = glob.glob('./test/*')
print(files)
```

```
['./test\\bike1.jpeg', './test\\bike2.jpeg', './test\\car1.jpeg', './test\\cat1.jpeg', './test\\dog1.jpeg', './test\\dog2.jpeg', './test\\dog3.jpeg', './test\\dog4.jpeg', './test\\ox1.jpeg']
```



根據你訓練的資料集找圖片預測！！

預測

預測多張圖片

```
cls_list = ['bike', 'car', 'cats', 'doge', 'ox']

for f in files:
    img = image.load_img(f, target_size=(224, 224))
    if img is None:
        continue
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis = 0)
    pred = model.predict(x)[0]
    top_inds = pred.argsort()[::-1][:5]
    print(f)
    for i in top_inds:
        print('  {:.3f}  {}'.format(pred[i], cls_list[i]))
```

預測成果

```
test\bike1.jpeg
    0.987  bike
    0.013  dogs
    0.000  car
    0.000  ox
    0.000  cats
test\car1.jpeg
    1.000  car
    0.000  ox
    0.000  bike
    0.000  cats
    0.000  dogs
test\ox1.jpeg
    0.990  ox
    0.007  car
    0.002  cats
    0.000  dogs
    0.000  bike
```

```
test\cat1.jpeg
    1.000  cats
    0.000  ox
    0.000  car
    0.000  dogs
    0.000  bike
test\dog4.jpeg
    0.997  dogs
    0.002  cats
    0.001  bike
    0.000  car
    0.000  ox
```

END