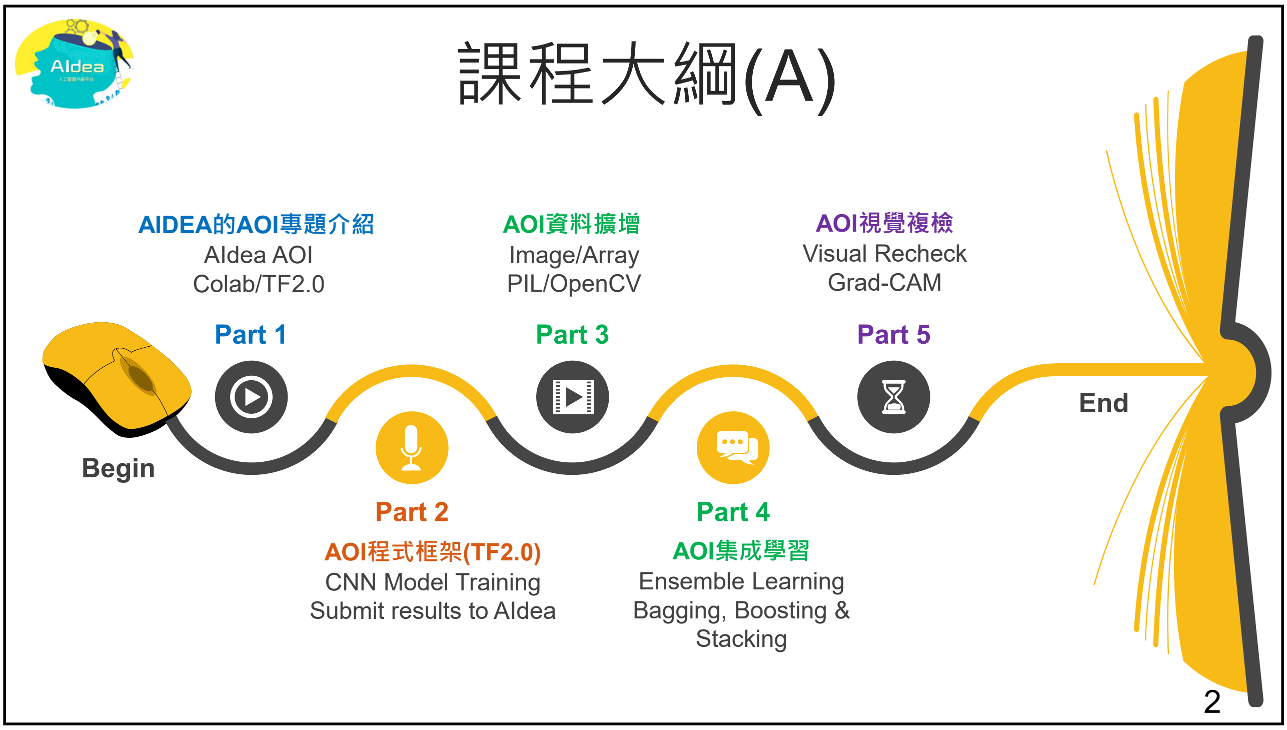
工研院產業學院
「AOI自動光學檢測瑕疵分類實作班」

(A)

亞洲大學朱學亭副教授
2020/07/01



課程大綱(A)

Part 1
AIDEA的AOI專題介紹
AIdea AOI
Colab/TF2.0

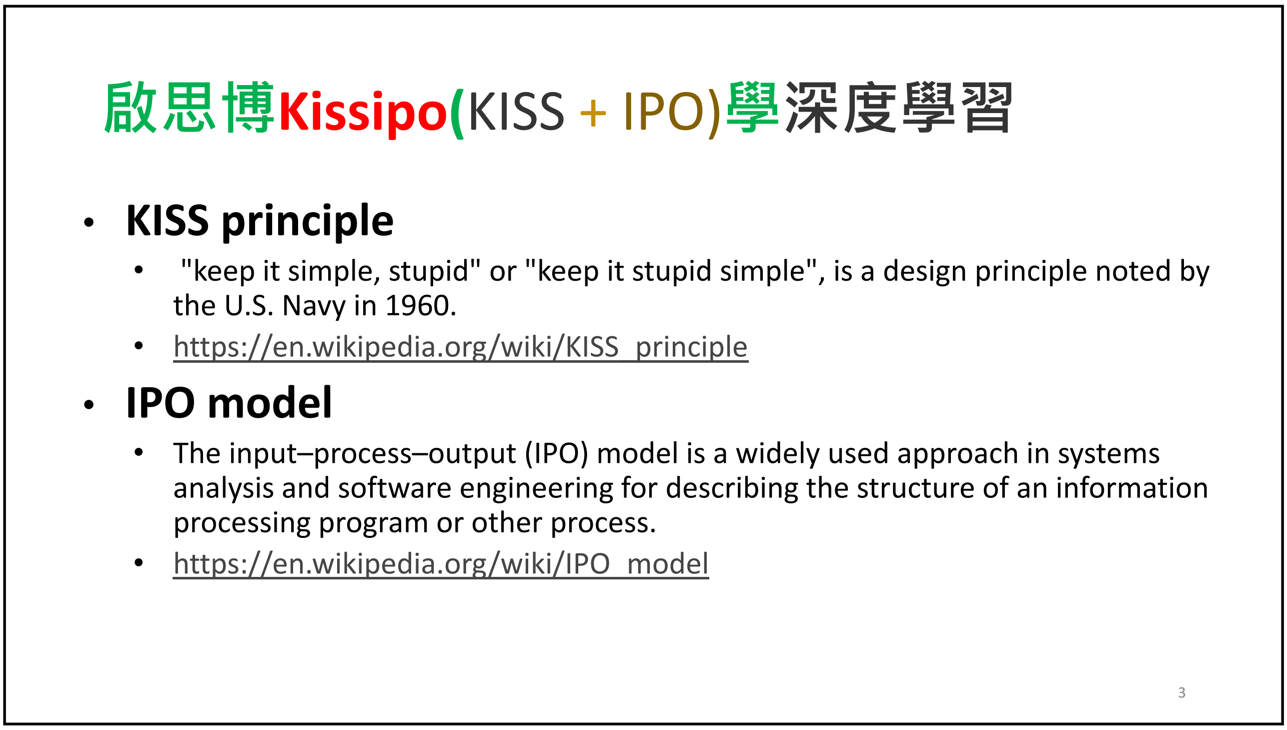
Part 2
AOI程式框架(TF2.0)
CNN Model Training
Submit results to AIdea

Part 3
AOI資料擴增
Image/Array
PIL/OpenCV

Part 4
AOI集成學習
Ensemble Learning
Bagging, Boosting &
Stacking

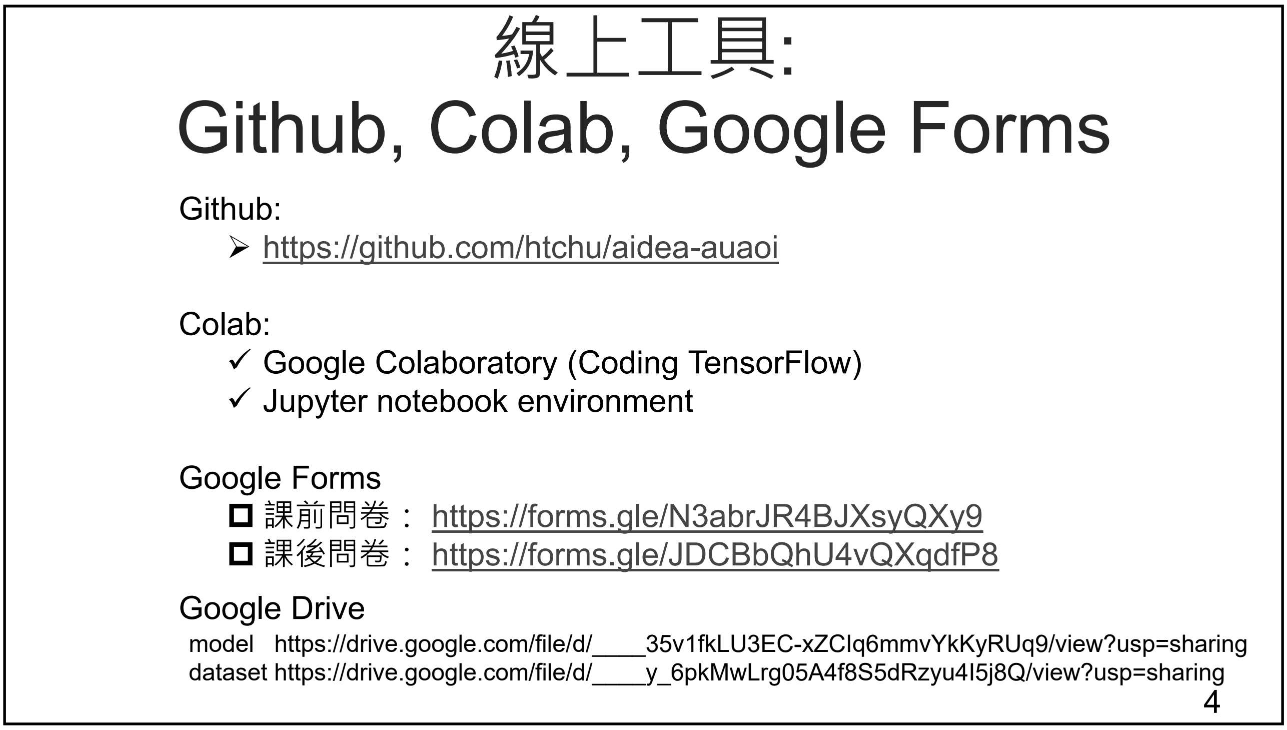
Part 5
AOI視覺複檢
Visual Recheck
Grad-CAM

2



啟思博Kissipo(KISS + IPO)學深度學習

- **KISS principle**
  - "keep it simple, stupid" or "keep it stupid simple", is a design principle noted by the U.S. Navy in 1960.
  - https://en.wikipedia.org/wiki/KISS_principle
- **IPO model**
  - The input–process–output (IPO) model is a widely used approach in systems analysis and software engineering for describing the structure of an information processing program or other process.
  - https://en.wikipedia.org/wiki/IPO_model



線上工具:
Github, Colab, Google Forms

Github:
- ➢ https://github.com/htchu/aidea-auaoi

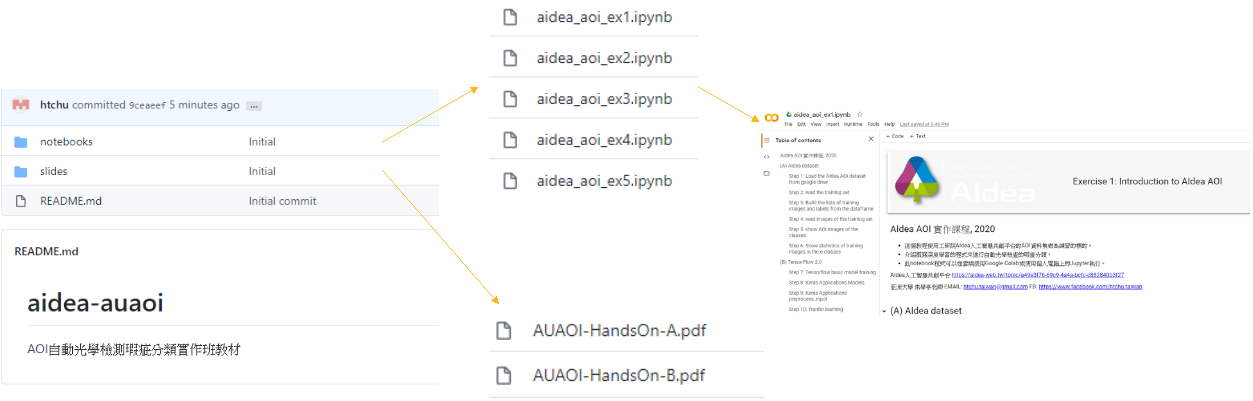
Colab:
- ✓ Google Colaboratory (Coding TensorFlow)
- ✓ Jupyter notebook environment

Google Forms
- □ 課前問卷： https://forms.gle/N3abrJR4BJXsyQXy9
- □ 課後問卷： https://forms.gle/JDCBbQhU4vQXqdfP8

Google Drive

model https://drive.google.com/file/d/____35v1fkLU3EC-xZCIq6mmvYkKyRUq9/view?usp=sharing
dataset https://drive.google.com/file/d/____y_6pkMwLrg05A4f8S5dRzyu4I5j8Q/view?usp=sharing

Github- AUAOI



htchu committed 9ceaeef 5 minutes ago

notebooks	Initial
slides	Initial
README.md	Initial commit

README.md

aidea-auaoi

AOI自動光學檢測瑕疵分類實作班教材

aidea_aoi_ex1.ipynb

aidea_aoi_ex2.ipynb

aidea_aoi_ex3.ipynb

aidea_aoi_ex4.ipynb

aidea_aoi_ex5.ipynb

AUAOI-HandsOn-A.pdf

AUAOI-HandsOn-B.pdf

aidea_aoi_ex1.ipynb

AIdea AOI 實作課程, 2020

(A) AIdea dataset

Step 1: Load the AIdea AOI dataset from google drive

Step 2: read the training set

Step 3: Build the lists of training images and labels from the dataframe

Step 4: read images of the training set

Step 5: show AOI images of the classes:

Step 6: Show statistics of training images in the 6 classes

(B) TensorFlow 2.0

Step 7: Tensorflow basic model training

Step 8: Keras Applications Models

Step 9: Keras Applications preprocess_input

Step 10: Tranfer learning

Exercise 1: Introduction to AIdea AOI

AIdea AOI 實作課程, 2020

- 這個教程使用工研院AIdea人工智慧共創平台的AOI資料集做為練習的標的。
- 介紹撰寫深度學習的程式來進行自動光學檢查的瑕疵分類。
- 此notebook程式可以在雲端使用Google Colab或使用個人電腦上的Jupyter執行。

AIdea人工智慧共創平台 https://aidea-web.tw/topic/a49e3f76-69c9-4a4a-bcfc-c882840b3f27

亞洲大學 朱學亭老師 EMAIL: htchu.taiwan@gmail.com FB: https://www.facebook.com/htchu.taiwan

(A) AIdea dataset

From Github to Google Colab

https://github.com/htchu/aidea-auaoi

Colab Limits

1. The GPUs available in Colab often include Nvidia K80s, T4s, P4s and P100s. There is no way to choose what type of GPU you can connect to in Colab at any given time.

2. Overall usage limits as well as idle timeout periods, maximum VM lifetime, GPU types available, and other factors vary over time.

aidea_aoi_ex1.ipynb

AIdea AOI 實作課程, 2020

(A) AIdea dataset

Step 1: Load the AIdea AOI dataset from google drive

Step 2: read the training set

Step 3: Build the lists of training images and labels from the dataframe

Step 4: read images of the training set

Step 5: show AOI images of the classes:

Step 6: Show statistics of training images in the 6 classes

(B) TensorFlow 2.0

Step 7: Tensorflow basic model training

Step 8: Keras Applications Models

Step 9: Keras Applications preprocess_input

Step 10: Tranfer learning

Exercise 1: Introduction to AIdea AOI

AIdea AOI 實作課程, 2020

- 這個教程使用工研院AIdea人工智慧共創平台的AOI資料集做為練習的標的。
- 介紹撰寫深度學習的程式來進行自動光學檢查的瑕疵分類。
- 此notebook程式可以在雲端使用Google Colab或使用個人電腦上的Jupyter執行。

AIdea人工智慧共創平台 https://aidea-web.tw/topic/a49e3f76-69c9-4a4a-bcfc-c882840b3f27

亞洲大學 朱學亭老師 EMAIL: htchu.taiwan@gmail.com FB: https://www.facebook.com/htchu.taiwan

(A) AIdea dataset

1.AIdea的AOI專題介紹

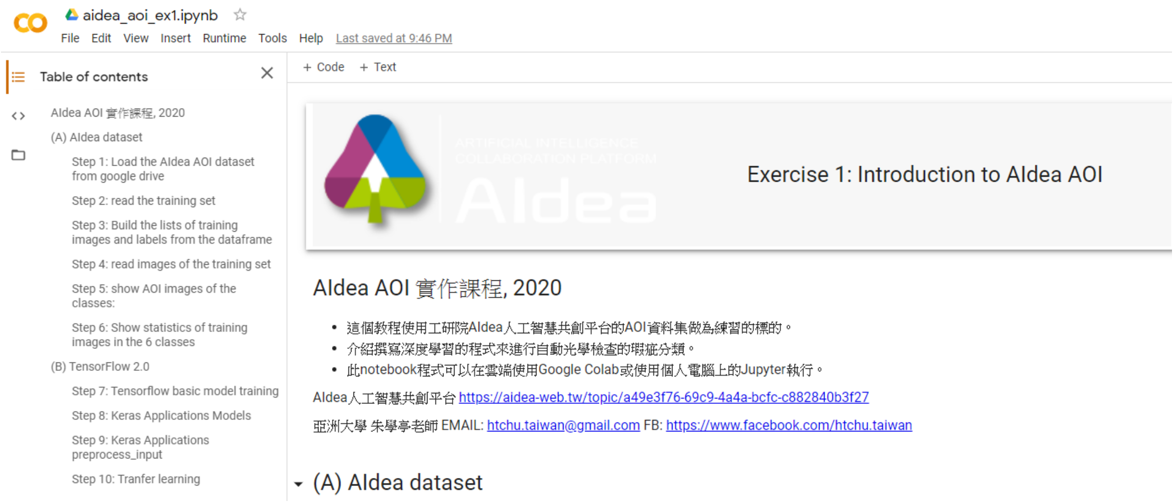


Table of contents

- AIdea AOI 實作課程, 2020
  - (A) AIdea dataset
    - Step 1: Load the AIdea AOI dataset from google drive
    - Step 2: read the training set
    - Step 3: Build the lists of training images and labels from the dataframe
    - Step 4: read images of the training set
    - Step 5: show AOI images of the classes:
    - Step 6: Show statistics of training images in the 6 classes
  - (B) TensorFlow 2.0
    - Step 7: Tensorflow basic model training
    - Step 8: Keras Applications Models
    - Step 9: Keras Applications preprocess_input
    - Step 10: Tranfer learning

Exercise 1: Introduction to AIdea AOI

AIdea AOI 實作課程, 2020

- 這個教程使用工研院AIdea人工智慧共創平台的AOI資料集做為練習的標的。
- 介紹撰寫深度學習的程式來進行自動光學檢查的瑕疵分類。
- 此notebook程式可以在雲端使用Google Colab或使用個人電腦上的Jupyter執行。

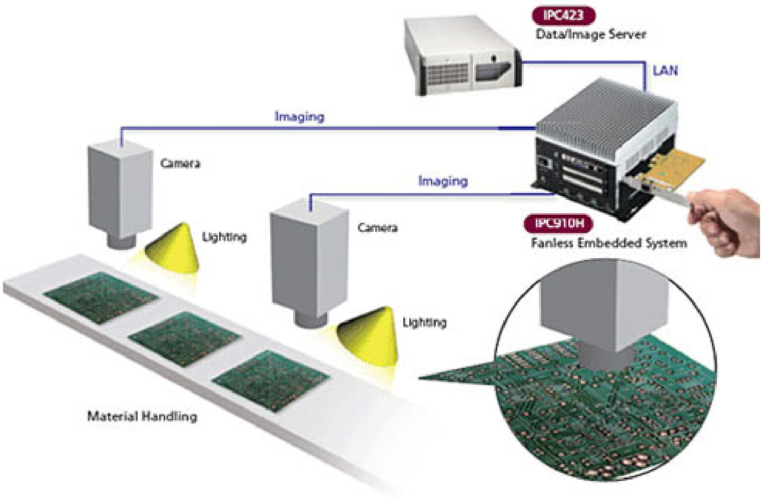
AIdea人工智慧共創平台 https://aidea-web.tw/topic/a49e3f76-69c9-4a4a-bcfc-c882840b3f27

亞洲大學 朱學亭老師 EMAIL: htchu.taiwan@gmail.com FB: https://www.facebook.com/htchu.taiwan

(A) AIdea dataset

7

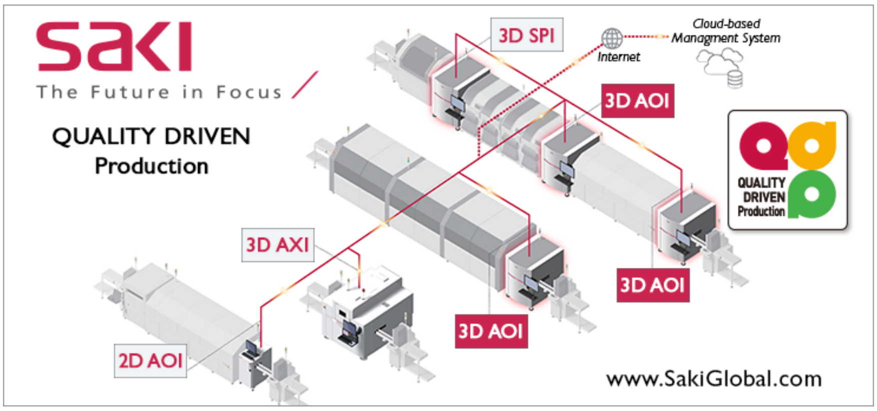
AOI system architecture



8

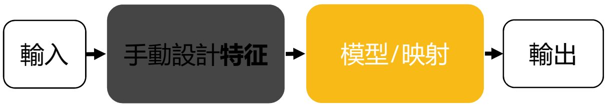
視覺檢測種類:SPI, AOI, AXI

SPI (Solder Paste Inspection)
AOI (Auto Optical Inspection)
AXI (Automatic X-ray Inspection)

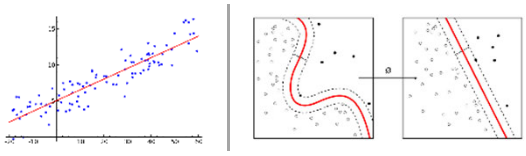


AOI工作流程差異

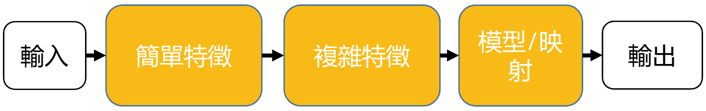
傳統機器學習的AOI[1990 年至今]


輸入 → 手動設計**特征** → 模型/映射 → 輸出

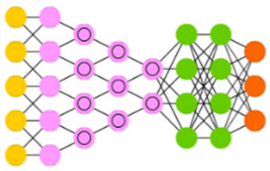
示例 [回歸和 SVM]



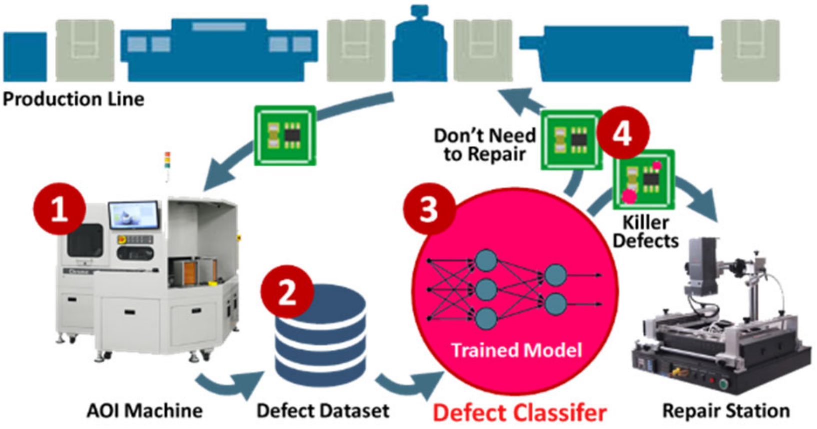
深度/端到端學習的AOI [2012 年至今]


輸入 → 簡單特徵 → 複雜特徵 → 模型/映射 → 輸出

示例 [卷積網路]

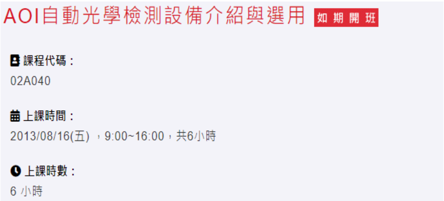


AOI with machine learning models



Classical ML-based AOI Machine

財團法人自強工業科學基金會
Tze-Chiang Foundation of Science & Technology



AOI自動光學檢測設備介紹與選用 如期開班

課程代碼：
02A040

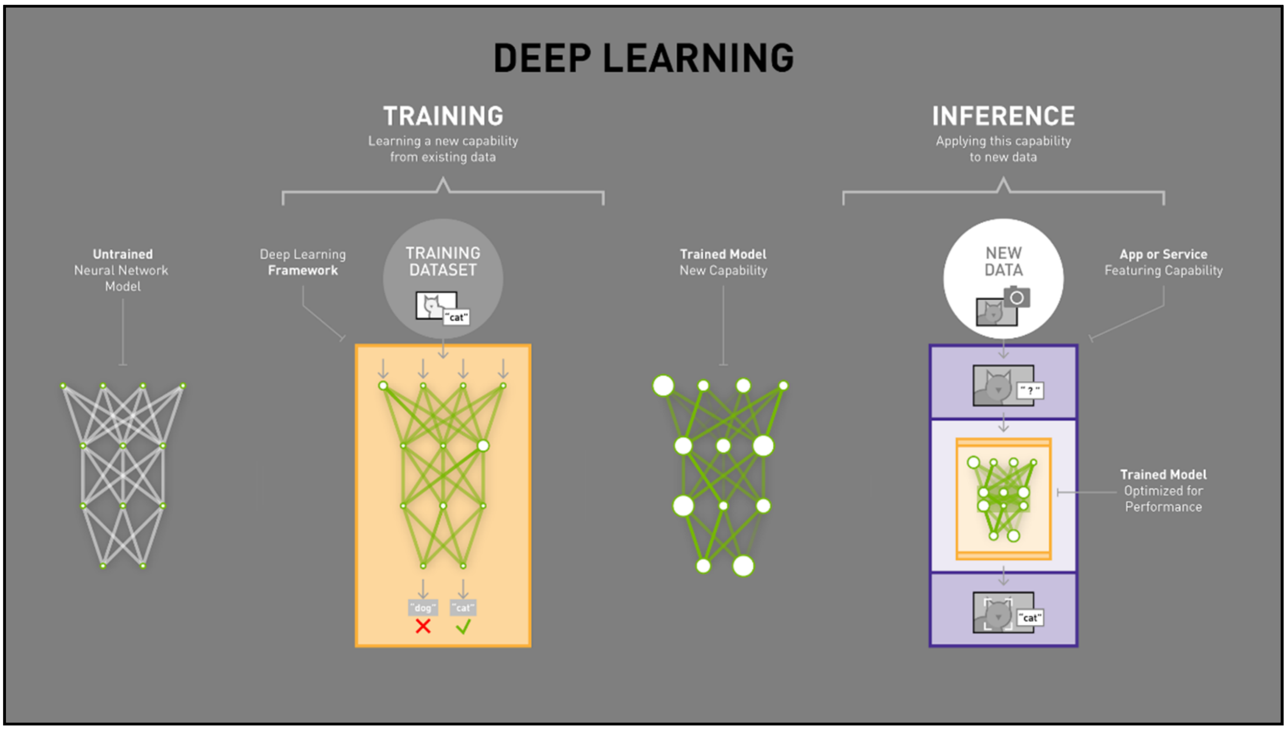
上課時間：
2013/08/16(五) ，9:00~16:00，共6小時

上課時數：
6 小時

課程大綱：
1.影像分析：
影像前處理：影像的運算、迴旋積運算、直方圖、直方圖修正、影像濾波處理
影像分割：二值化、多值化、自動閥值
影像後處理：邏輯運算、像素的近鄰、形態處理、骨架化、及物件標號

2. 影像量測：
邊界偵測：邊界、邊界點、梯度、一階導數運算子(Sobel、Prewitt、Robert)、二導數運算子(Laplacian、LOG)、鏈碼輪廓追蹤法
尺寸量測：最小平方法、霍式轉換法、直線搓合、圓及圓弧的搓合、線寬線距的量測、圓形物件的量測、矩形物體的量測。

3.影像辨識：
特徵抽取：特徵值(周長、離心率、真圓度、粗糙度、長短距離比、角數、平均距離、距離標準差、尤拉數、灰階標準差)
物件識別與分類:最近鄰分類器、K-NN分類器、類神經網路分類

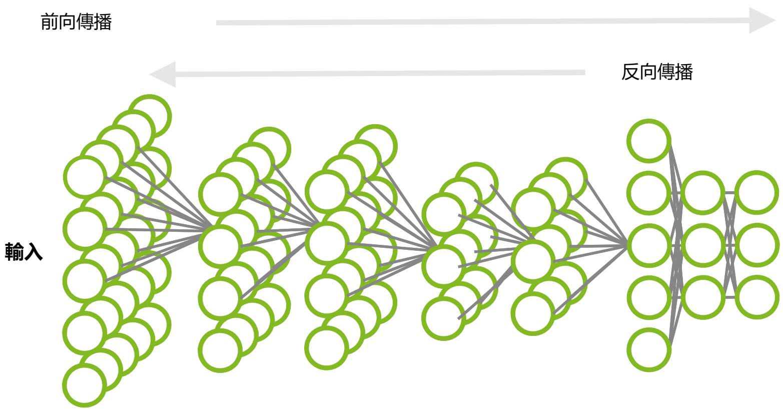

DEEP LEARNING
TRAINING
Learning a new capability from existing data
INFERENCE
Applying this capability to new data
Untrained Neural Network Model
Deep Learning Framework
TRAINING DATASET
"cat"
"dog"
"cat"
Trained Model New Capability
NEW DATA
App or Service Featuring Capability
"?"
Trained Model Optimized for Performance
"cat"

New DL-based AOI Machine

台達視覺檢測的 DAVS 以人工智慧為核心系統，結合既有的 AOI 系統，讓既有設備可延長使用年限以此保障製造業者過去的投資，而人工智慧與 AOI 整合的模式，也提升了產品的檢出率。

https://buzzorange.com/techorange/2019/09/05/delta-aoi-system/

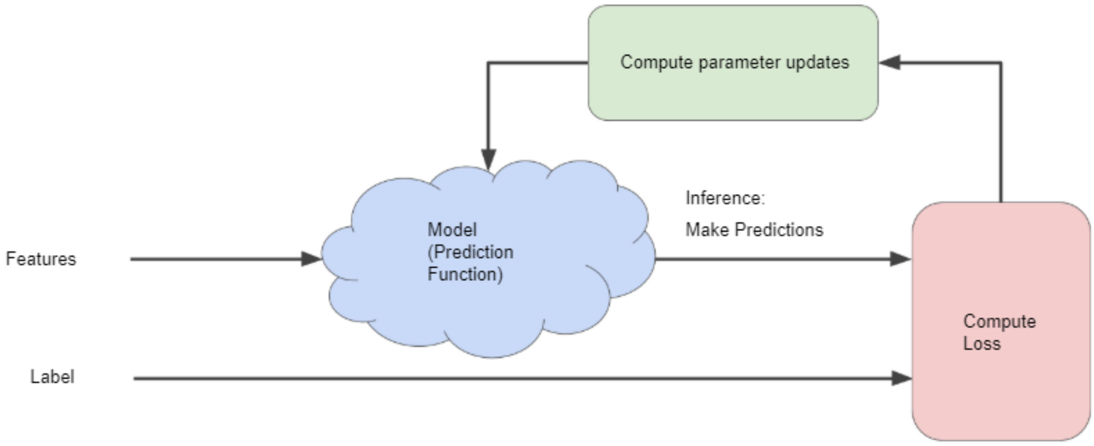
深度學習方法 - 訓練



流程

- **前向傳播為每個訓練圖像生成一個推斷標籤**
- **使用損失函數計算每個圖像的已知標籤與預測標籤之間的差異**
- **在後向傳播過程中調整權重**
- **重複該流程**

Reducing Loss: An Iterative Approach



Convolutional neural network(CNN)

Add convolution and pooling layers before feedforward neural network

卷積計算

- 步伐(stride)
- 填充(padding)

池化運算

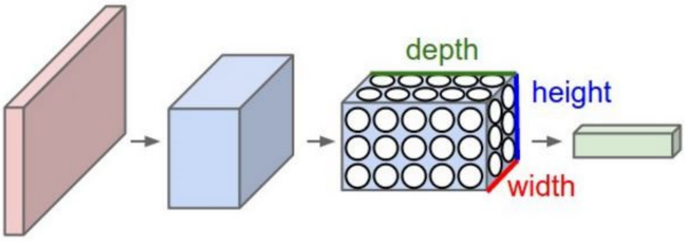
```
keras.layers.Conv2D(filters, kernel_size, strides=(1, 1),
padding='valid', data_format=None, dilation_rate=(1, 1),
activation=None, use_bias=True,
kernel_initializer='glorot_uniform', bias_initializer='zeros',
kernel_regularizer=None, bias_regularizer=None,
activity_regularizer=None, kernel_constraint=None,
bias_constraint=None)

keras.layers.MaxPooling2D(pool_size=(2, 2),
strides=None, padding='valid', data_format=None)
```

CNN History

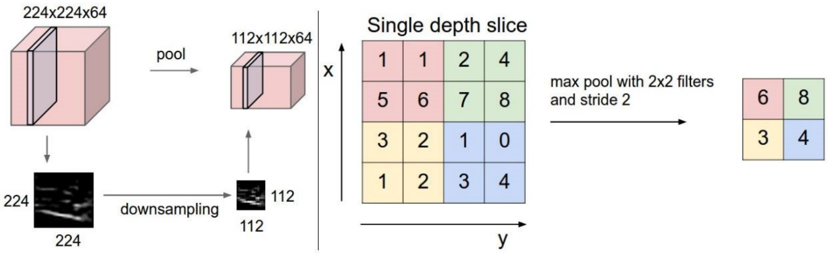
楊立昆
Yann LeCun, Professor of Computer Science
The Courant Institute of Mathematical Sciences
New York University
Room 1220, 715 Broadway, New York, NY 10003, USA.
(212)998-3283   yann@cs.nyu.edu

In 1995, Yann LeCun and Yoshua Bengio introduced the concept of convolutional neural networks.

Convolutional layer



Pooling layer



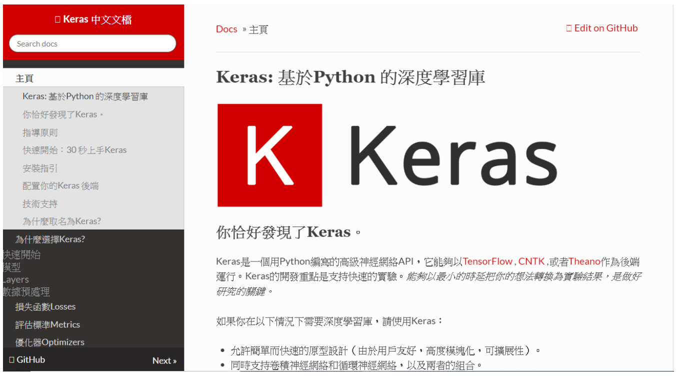
Single depth slice

1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

max pool with 2x2 filters and stride 2

6	8
3	4

Keras



Keras: 基於Python 的深度學習庫

你恰好發現了Keras。

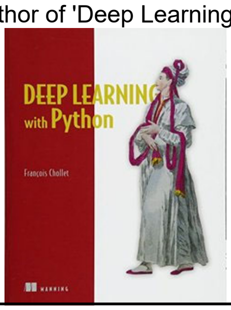
Keras是一個用Python編寫的高級神經網絡API，它能夠以TensorFlow , CNTK ,或者Theano作為後端運行。Keras的開發重點是支持快速的實驗。*能夠以最小的時延把你的想法轉換為實驗結果，是做好研究的關鍵。*

如果你在以下情況下需要深度學習庫，請使用Keras：

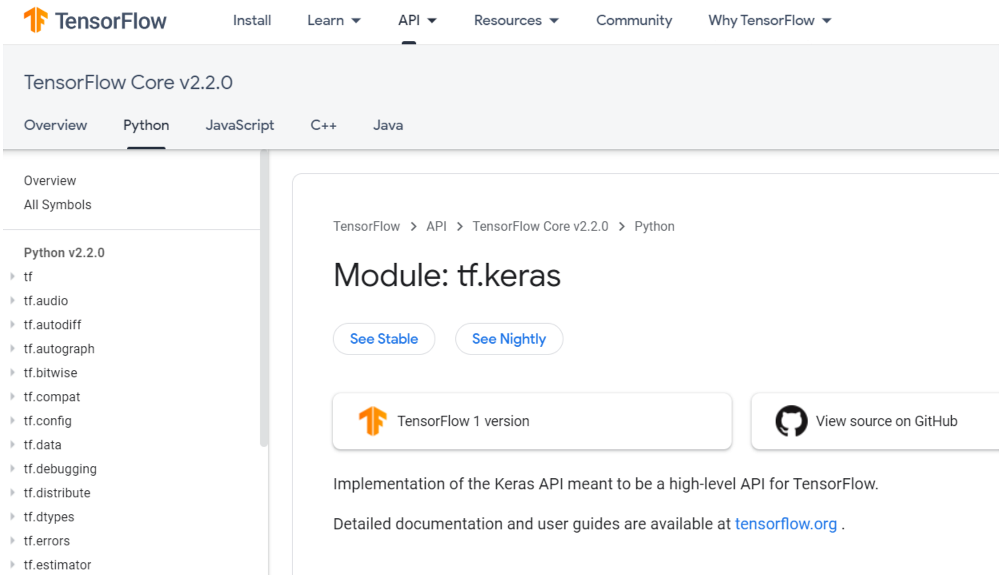
- 允許簡單而快速的原型設計（由於用戶友好，高度模塊化，可擴展性）。
- 同時支持卷積神經網絡和循環神經網絡，以及兩者的組合。

https://keras.io/zh/

François Chollet 弗朗索瓦·喬萊特
Deep learning @google
Creator of Keras, neural networks library.
Author of 'Deep Learning with Python'.



TensorFlow v2.2.0



TensorFlow Core v2.2.0

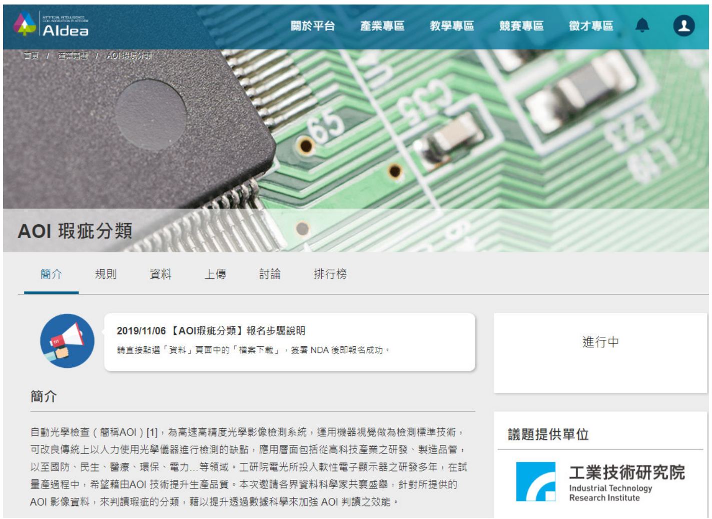
TensorFlow > API > TensorFlow Core v2.2.0 > Python

Module: tf.keras

Implementation of the Keras API meant to be a high-level API for TensorFlow.

Detailed documentation and user guides are available at tensorflow.org .

AIdea AOI Project



AOI 瑕疵分類

2019/11/06 【AOI瑕疵分類】報名步驟說明

請直接點選「資料」頁面中的「檔案下載」，簽署 NDA 後即報名成功。

簡介

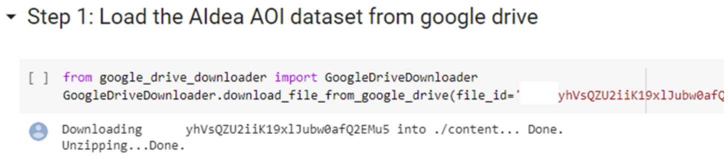
自動光學檢查（簡稱AOI）[1]，為高速高精度光學影像檢測系統，運用機器視覺做為檢測標準技術，可改良傳統上以人力使用光學儀器進行檢測的缺點，應用層面包括從高科技產業之研發、製造品管，以至國防、民生、醫療、環保、電力…等領域。工研院電光所投入軟性電子顯示器之研發多年，在試量產過程中，希望藉由AOI 技術提升生產品質。本次邀請各界資料科學家共襄盛舉，針對所提供的 AOI 影像資料，來判讀瑕疵的分類，藉以提升透過數據科學來加強 AOI 判讀之效能。

議題提供單位

https://aidea-web.tw/topic/a49e3f76-69c9-4a4a-bcfc-c882840b3f27

Step 1: Load the dataset from google drive

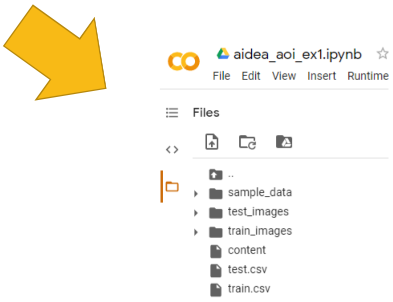
AUAOI Ex1.



Step 1: Load the AIdea AOI dataset from google drive

```
[ ] from google_drive_downloader import GoogleDriveDownloader
    GoogleDriveDownloader.download_file_from_google_drive(file_id='      yhVsQZU2iiK19xlJubw0afQ
```

```
Downloading      yhVsQZU2iiK19xlJubw0afQ2EMu5 into ./content... Done.
Unzipping...Done.
```

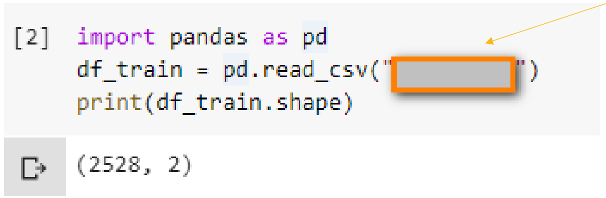


Step 2:

AUAOI Ex1.

Step 2: read the training set

train.csv

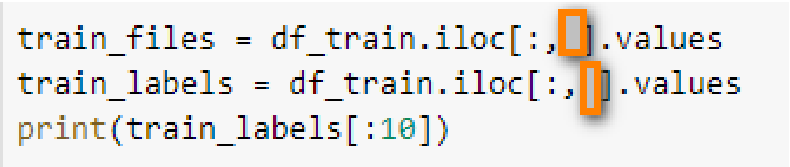


```
[2] import pandas as pd
    df_train = pd.read_csv("")
    print(df_train.shape)
```

```
(2528, 2)
```

Step 3: Build the lists of training images and labels

AUAOI Ex1.



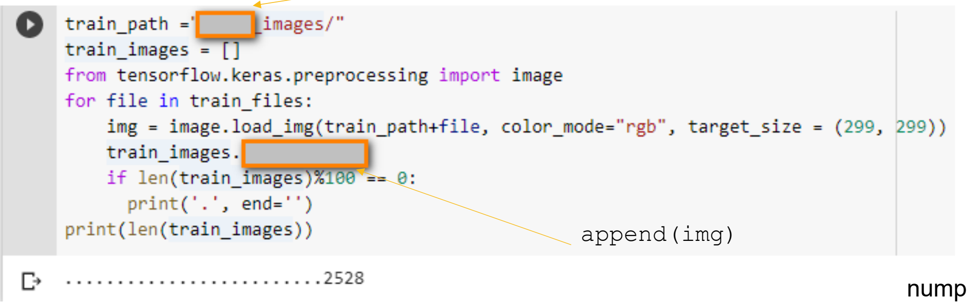
```
train_files = df_train.iloc[:, ].values
train_labels = df_train.iloc[:, ].values
print(train_labels[:10])
```

- •train_images.zip：訓練所需的影像資料（PNG格式），共計 2,528 張。
- •train.csv：包含 2 個欄位，ID 和 Label。
    - ID：影像的檔名。
    - Label：瑕疵分類類別（0 表示 normal，1 表示 void，2 表示 horizontal defect，3 表示 vertical defect，4 表示 edge defect，5 表示 particle）。
- •test_images.zip：測試所需的影像資料（PNG格式），共計 10,142 張。
- •test.csv：包含 2 個欄位，ID 和 Label。
    - ID：影像的檔名。
    - Label：瑕疵分類類別（其值只能是下列其中之一：0、1、2、3、4、5）。

Step 4: read images of the training set

AUAOI Ex1.

train_images/



```
train_path ='     images/"
train_images = []
from tensorflow.keras.preprocessing import image
for file in train_files:
    img = image.load_img(train_path+file, color_mode="rgb", target_size = (299, 299))
    train_images.
    if len(train_images)%100 == 0:
      print('.', end='')
print(len(train_images))
```

```
........................2528
```

append(img)

numpy.ndarray（值0-255）

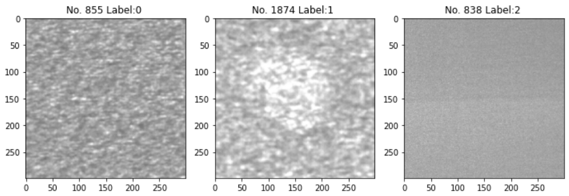
```
from tensorflow.keras.preprocessing import image
img = image.load_img(file, color_mode="rgb", target_size = (299, 299))
```

```
from PIL import Image
img = Image.open(file)

import matplotlib.pyplot as plt
img = plt.imread(file)
```

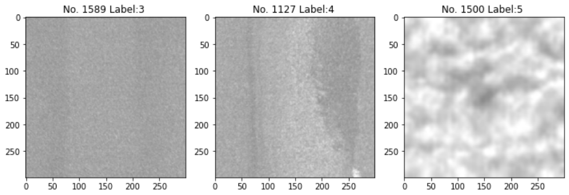
```
from skimage import io
img = io.imread(file)
import cv2
img = cv2.imread(file) #OpenCV is BGR, Pillow is RGB
#imread is deprecated in SciPy 1.0.0
```

Step 5: show AOI images of the classes

AUAOI Ex1.



0 (normal), 1 (void), 2 (horizontal defect)



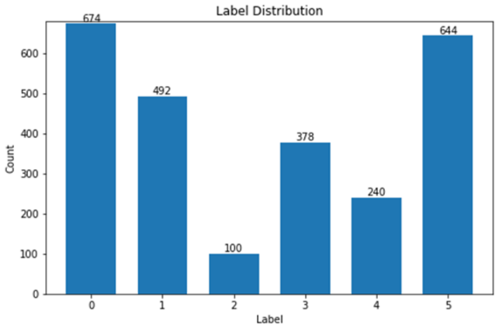
3 (vertical defect), 4 (edge defect), 5 (particle)

Step 6: Show statistics of training images in the 6 classes

AUAOI Ex1.

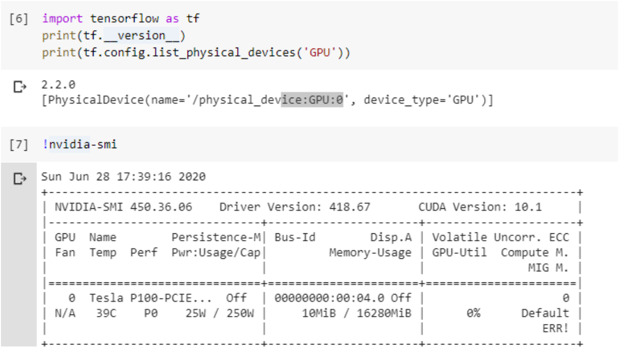
train_labels

```
import numpy as np
labels, counts = np.unique(□, return_counts=True)
print(labels, counts)
```



Step 7a: Tensorflow basic model training

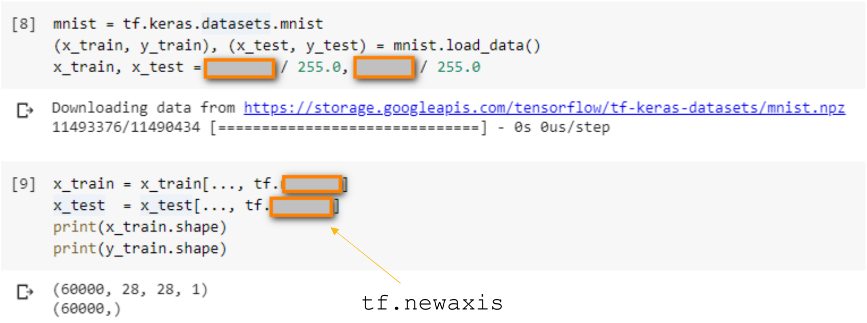
AUAOI Ex1.



```
[6] import tensorflow as tf
    print(tf.__version__)
    print(tf.config.list_physical_devices('GPU'))

    2.2.0
    [PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]

[7] !nvidia-smi

    Sun Jun 28 17:39:16 2020
    +-----------------------------------------------------------------------------+
    | NVIDIA-SMI 450.36.06    Driver Version: 418.67       CUDA Version: 10.1     |
    |-------------------------------+----------------------+----------------------+
    | GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
    | Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
    |                               |                      |               MIG M. |
    |===============================+======================+======================|
    |   0  Tesla P100-PCIE...  Off  | 00000000:00:04.0 Off |                    0 |
    | N/A   39C    P0    25W / 250W |     10MiB / 16280MiB |      0%      Default |
    |                               |                      |                 ERR! |
    +-------------------------------+----------------------+----------------------+
```

Step 7b:

AUAOI Ex1.

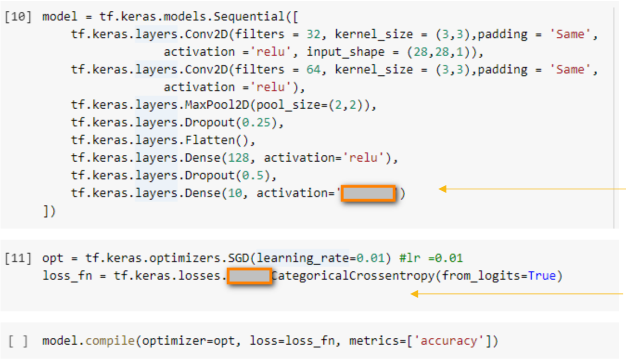


```
[8] mnist = tf.keras.datasets.mnist
    (x_train, y_train), (x_test, y_test) = mnist.load_data()
    x_train, x_test =        / 255.0,        / 255.0

    Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
    11493376/11490434 [==============================] - 0s 0us/step

[9] x_train = x_train[..., tf.       ]
    x_test  = x_test[..., tf.       ]
    print(x_train.shape)
    print(y_train.shape)

    (60000, 28, 28, 1)
    (60000,)
```

tf.newaxis

Step 7c:

AUAOI Ex1.



```
[10] model = tf.keras.models.Sequential([
         tf.keras.layers.Conv2D(filters = 32, kernel_size = (3,3),padding = 'Same',
                  activation ='relu', input_shape = (28,28,1)),
         tf.keras.layers.Conv2D(filters = 64, kernel_size = (3,3),padding = 'Same',
                  activation ='relu'),
         tf.keras.layers.MaxPool2D(pool_size=(2,2)),
         tf.keras.layers.Dropout(0.25),
         tf.keras.layers.Flatten(),
         tf.keras.layers.Dense(128, activation='relu'),
         tf.keras.layers.Dropout(0.5),
         tf.keras.layers.Dense(10, activation='       ')
     ])
```

```
[11] opt = tf.keras.optimizers.SGD(learning_rate=0.01) #lr =0.01
     loss_fn = tf.keras.losses.      CategoricalCrossentropy(from_logits=True)
```

```
[ ]  model.compile(optimizer=opt, loss=loss_fn, metrics=['accuracy'])
```

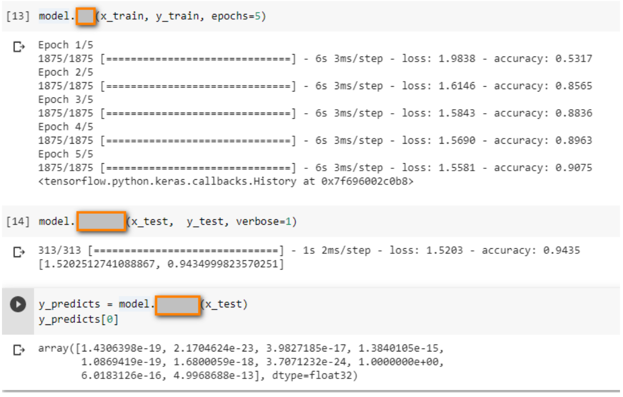
softmax

SparseCategoricalCrossentropy

31

Step 7d:

AUAOI Ex1.



```
[13] model.   (x_train, y_train, epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 6s 3ms/step - loss: 1.9838 - accuracy: 0.5317
Epoch 2/5
1875/1875 [==============================] - 6s 3ms/step - loss: 1.6146 - accuracy: 0.8565
Epoch 3/5
1875/1875 [==============================] - 6s 3ms/step - loss: 1.5843 - accuracy: 0.8836
Epoch 4/5
1875/1875 [==============================] - 6s 3ms/step - loss: 1.5690 - accuracy: 0.8963
Epoch 5/5
1875/1875 [==============================] - 6s 3ms/step - loss: 1.5581 - accuracy: 0.9075
<tensorflow.python.keras.callbacks.History at 0x7f696002c0b8>
```

```
[14] model.        (x_test,  y_test, verbose=1)
```

```
313/313 [==============================] - 1s 2ms/step - loss: 1.5203 - accuracy: 0.9435
[1.5202512741088867, 0.9434999823570251]
```

```
y_predicts = model.       (x_test)
y_predicts[0]
```

```
array([1.4306398e-19, 2.1704624e-23, 3.9827185e-17, 1.3840105e-15,
       1.0869419e-19, 1.6800059e-18, 3.7071232e-24, 1.0000000e+00,
       6.0183126e-16, 4.9968688e-13], dtype=float32)
```

fit

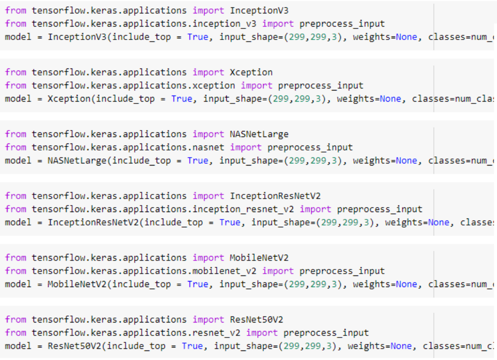
evaluate

predict

32

Step 8: Keras Applications Models

AUAOI Ex1.



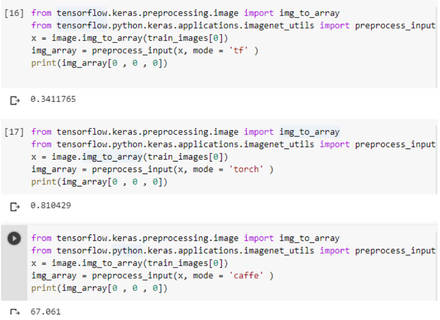
```
from tensorflow.keras.applications import InceptionV3
from tensorflow.keras.applications.inception_v3 import preprocess_input
model = InceptionV3(include_top = True, input_shape=(299,299,3), weights=None, classes=num_

from tensorflow.keras.applications import Xception
from tensorflow.keras.applications.xception import preprocess_input
model = Xception(include_top = True, input_shape=(299,299,3), weights=None, classes=num_cla

from tensorflow.keras.applications import NASNetLarge
from tensorflow.keras.applications.nasnet import preprocess_input
model = NASNetLarge(include_top = True, input_shape=(299,299,3), weights=None, classes=num_

from tensorflow.keras.applications import InceptionResNetV2
from tensorflow.keras.applications.inception_resnet_v2 import preprocess_input
model = InceptionResNetV2(include_top = True, input_shape=(299,299,3), weights=None, classe

from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
model = MobileNetV2(include_top = True, input_shape=(299,299,3), weights=None, classes=num_

from tensorflow.keras.applications import ResNet50V2
from tensorflow.keras.applications.resnet_v2 import preprocess_input
model = ResNet50V2(include_top = True, input_shape=(299,299,3), weights=None, classes=num_c
```

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.900	143,667,240	26
ResNet50	98 MB	0.749	0.921	25,636,712	-
ResNet101	171 MB	0.764	0.928	44,707,176	-
ResNet152	232 MB	0.766	0.931	60,419,944	-
ResNet50V2	98 MB	0.760	0.930	25,613,800	-
ResNet101V2	171 MB	0.772	0.938	44,675,560	-
ResNet152V2	232 MB	0.780	0.942	60,380,648	-
InceptionV3	92 MB	0.779	0.937	23,851,784	159
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
MobileNet	16 MB	0.704	0.895	4,253,864	88
MobileNetV2	14 MB	0.713	0.901	3,538,984	88
DenseNet121	33 MB	0.750	0.923	8,062,504	121
DenseNet169	57 MB	0.762	0.932	14,307,880	169
DenseNet201	80 MB	0.773	0.936	20,242,984	201
NASNetMobile	23 MB	0.744	0.919	5,326,716	-
NASNetLarge	343 MB	0.825	0.960	88,949,818	-
EfficientNetB0	29 MB	-	-	5,330,571	-
EfficientNetB1	31 MB	-	-	7,856,239	-
EfficientNetB2	36 MB	-	-	9,177,569	-
EfficientNetB3	48 MB	-	-	12,320,535	-
EfficientNetB4	75 MB	-	-	19,466,823	-
EfficientNetB5	118 MB	-	-	30,562,527	-
EfficientNetB6	166 MB	-	-	43,265,143	-
EfficientNetB7	256 MB	-	-	66,658,687	-

33

Step 9: Keras Applications preprocess_input

AUAOI Ex1.



```
[16] from tensorflow.keras.preprocessing.image import img_to_array
     from tensorflow.python.keras.applications.imagenet_utils import preprocess_input
     x = image.img_to_array(train_images[0])
     img_array = preprocess_input(x, mode = 'tf' )
     print(img_array[0 , 0 , 0])

     0.3411765

[17] from tensorflow.keras.preprocessing.image import img_to_array
     from tensorflow.python.keras.applications.imagenet_utils import preprocess_input
     x = image.img_to_array(train_images[0])
     img_array = preprocess_input(x, mode = 'torch' )
     print(img_array[0 , 0 , 0])

     0.810429

     from tensorflow.keras.preprocessing.image import img_to_array
     from tensorflow.python.keras.applications.imagenet_utils import preprocess_input
     x = image.img_to_array(train_images[0])
     img_array = preprocess_input(x, mode = 'caffe' )
     print(img_array[0 , 0 , 0])

     67.061
```

	Input size	Data format	mode
Xception	299x299	channels_last	tf
VGG16	224x224	channels_first / channels_last	caffe
VGG19	224x224	channels_first / channels_last	caffe
ResNet50	224x224	channels_first / channels_last	caffe
InceptionV3	299x299	channels_first / channels_last	tf
InceptionResNetV2	299x299	channels_first / channels_last	tf
MobileNet	224x224	channels_last	tf
DenseNet	224x224	channels_first / channels_last	torch
NASNet	331x331 / 224x224	channels_first / channels_last	tf
MobileNetV2	224x224	channels_last	tf

*mode = **caffe***

(will convert the images from RGB to BGR, then will zero-center each color channel with respect to the ImageNet dataset)

減去ImageNet平均 BGR [103.939, 116.779, 123.68]

*mode = **tf***

(will scale pixels between -1 and 1) **除以127.5，然後減 1。**

*mode = **torch***

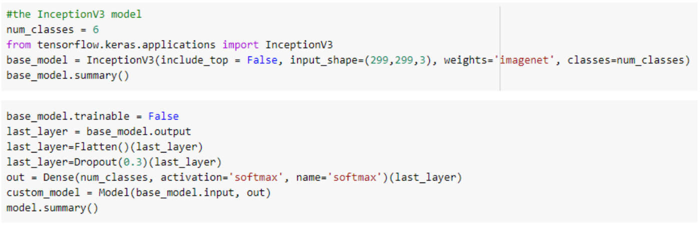
(will scale pixels between 0 and 1 and then will normalize each channel with respect to the ImageNet dataset)

除以255，減去ImageNet平均[0.485, 0.456, 0.406] ，除以標準差[0.229, 0.224, 0.225]。

34

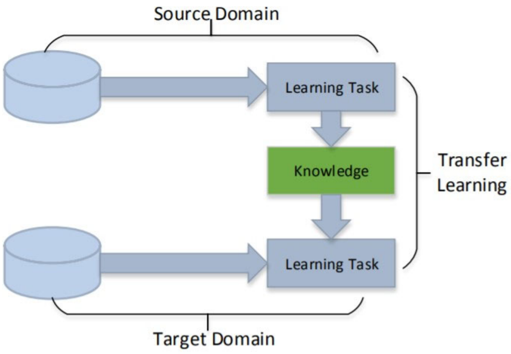
Step 10: Tranfer learning

AUAOI Ex1.



```
#the InceptionV3 model
num_classes = 6
from tensorflow.keras.applications import InceptionV3
base_model = InceptionV3(include_top = False, input_shape=(299,299,3), weights='imagenet', classes=num_classes)
base_model.summary()
```

```
base_model.trainable = False
last_layer = base_model.output
last_layer=Flatten()(last_layer)
last_layer=Dropout(0.3)(last_layer)
out = Dense(num_classes, activation='softmax', name='softmax')(last_layer)
custom_model = Model(base_model.input, out)
model.summary()
```



https://www.coderbridge.com/@gueiyajhang/54584ea6d4c240aeb3b8ae4af3a0531a

2. TF 2.0 AOI程式框架

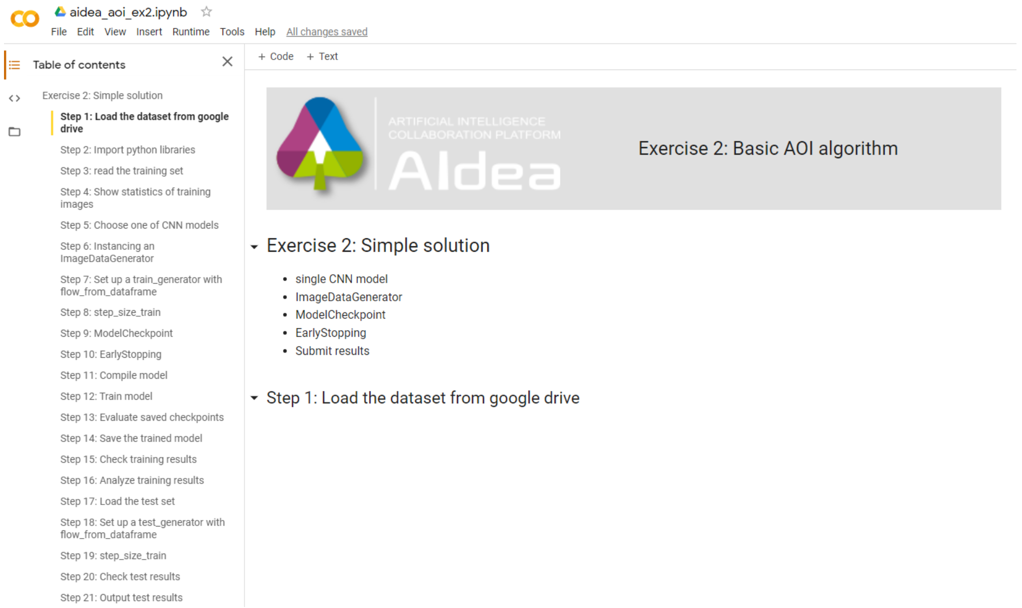

aidea_aoi_ex2.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
ARTIFICIAL INTELLIGENCE COLLABORATION PLATFORM
AIdea

Table of contents

Exercise 2: Basic AOI algorithm

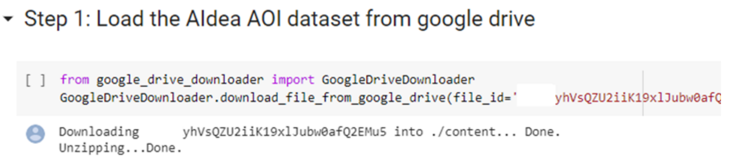
Exercise 2: Simple solution

- single CNN model
- ImageDataGenerator
- ModelCheckpoint
- EarlyStopping
- Submit results

Step 1: Load the dataset from google drive

Step 1: Load the dataset from google drive

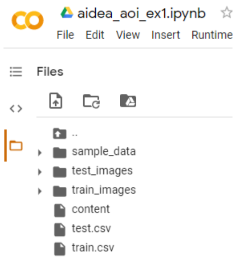
AUAOI Ex2.



▾ Step 1: Load the AIdea AOI dataset from google drive

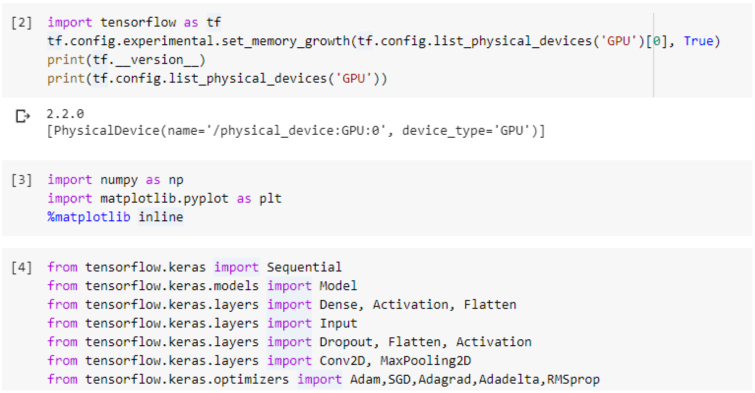
```
[ ]  from google_drive_downloader import GoogleDriveDownloader
     GoogleDriveDownloader.download_file_from_google_drive(file_id='      yhVsQZU2iiK19xlJubw0afQ
```

```
Downloading      yhVsQZU2iiK19xlJubw0afQ2EMu5 into ./content... Done.
Unzipping...Done.
```



Step 2: Import python libraries

AUAOI Ex2.



```
[2]  import tensorflow as tf
     tf.config.experimental.set_memory_growth(tf.config.list_physical_devices('GPU')[0], True)
     print(tf.__version__)
     print(tf.config.list_physical_devices('GPU'))
```

```
2.2.0
[PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]
```

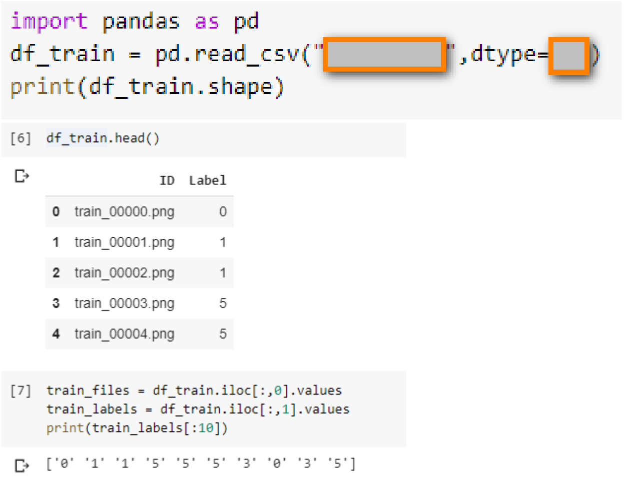
```
[3]  import numpy as np
     import matplotlib.pyplot as plt
     %matplotlib inline
```

```
[4]  from tensorflow.keras import Sequential
     from tensorflow.keras.models import Model
     from tensorflow.keras.layers import Dense, Activation, Flatten
     from tensorflow.keras.layers import Input
     from tensorflow.keras.layers import Dropout, Flatten, Activation
     from tensorflow.keras.layers import Conv2D, MaxPooling2D
     from tensorflow.keras.optimizers import Adam,SGD,Adagrad,Adadelta,RMSprop
```

Step 3: read the training set

AUAOI Ex2.

train.csv str



```
import pandas as pd
df_train = pd.read_csv("          ",dtype=   )
print(df_train.shape)
```

```
[6] df_train.head()
```

	ID	Label
0	train_00000.png	0
1	train_00001.png	1
2	train_00002.png	1
3	train_00003.png	5
4	train_00004.png	5

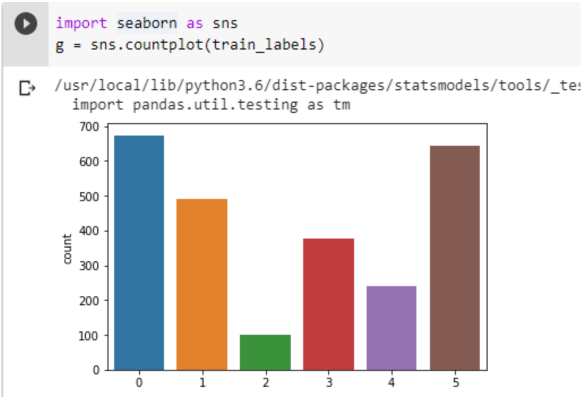
```
[7] train_files = df_train.iloc[:,0].values
    train_labels = df_train.iloc[:,1].values
    print(train_labels[:10])
```

```
['0' '1' '1' '5' '5' '5' '3' '0' '3' '5']
```

39

Step 4: Show statistics of training images

AUAOI Ex2.



```
import seaborn as sns
g = sns.countplot(train_labels)
```

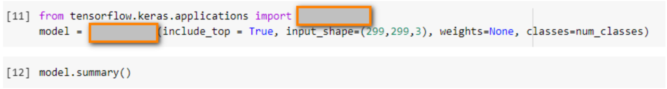
```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_te
  import pandas.util.testing as tm
```

```
num_classes=6
```

40

Step 5: Choose one of CNN models

AUAOI Ex2.



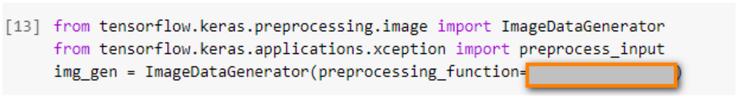
```
[11] from tensorflow.keras.applications import 
     model =           (include_top = True, input_shape=(299,299,3), weights=None, classes=num_classes)

[12] model.summary()
```

- DenseNet121(...): Instantiates the Densenet121 architecture.
- DenseNet169(...): Instantiates the Densenet169 architecture.
- DenseNet201(...): Instantiates the Densenet201 architecture.
- InceptionResNetV2(...): Instantiates the Inception-ResNet v2 architecture.
- InceptionV3(...): Instantiates the Inception v3 architecture.
- MobileNet(...): Instantiates the MobileNet architecture.
- MobileNetV2(...): Instantiates the MobileNetV2 architecture.
- NASNetLarge(...): Instantiates a NASNet model in ImageNet mode.
- NASNetMobile(...): Instantiates a Mobile NASNet model in ImageNet mode.
- ResNet101(...): Instantiates the ResNet101 architecture.
- ResNet101V2(...): Instantiates the ResNet101V2 architecture.
- ResNet152(...): Instantiates the ResNet152 architecture.
- ResNet152V2(...): Instantiates the ResNet152V2 architecture.
- ResNet50(...): Instantiates the ResNet50 architecture.
- ResNet50V2(...): Instantiates the ResNet50V2 architecture.
- VGG16(...): Instantiates the VGG16 model.
- VGG19(...): Instantiates the VGG19 architecture.
- Xception(...): Instantiates the Xception architecture.

Step 6: Instancing an ImageDataGenerator

AUAOI Ex2.

preprocess_input



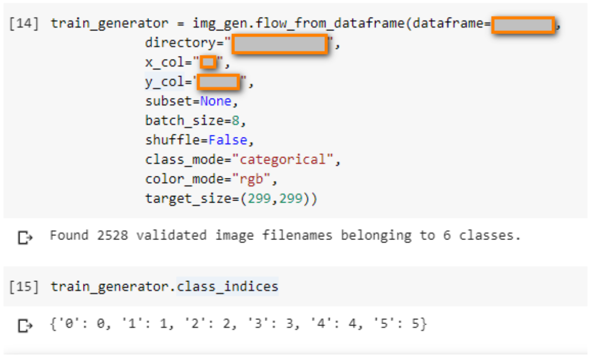
```
[13] from tensorflow.keras.preprocessing.image import ImageDataGenerator
     from tensorflow.keras.applications.xception import preprocess_input
     img_gen = ImageDataGenerator(preprocessing_function=            )
```

```
tf.keras.preprocessing.image.ImageDataGenerator(
    featurewise_center=False, samplewise_center=False,
    featurewise_std_normalization=False, samplewise_std_normalization=False,
    zca_whitening=False, zca_epsilon=1e-06, rotation_range=0, width_shift_range=0.0,
    height_shift_range=0.0, brightness_range=None, shear_range=0.0, zoom_range=0.0,
    channel_shift_range=0.0, fill_mode='nearest', cval=0.0, horizontal_flip=False,
    vertical_flip=False, rescale=None, preprocessing_function=None,
    data_format=None, validation_split=0.0, dtype=None
)
```

https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator

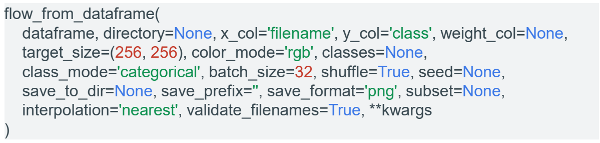
Step 7: Set up a train_generator with flow_from_dataframe

AUAOI Ex2.



```
[14] train_generator = img_gen.flow_from_dataframe(dataframe=      ,
                 directory="            ",
                 x_col="  ",
                 y_col="     ",
                 subset=None,
                 batch_size=8,
                 shuffle=False,
                 class_mode="categorical",
                 color_mode="rgb",
                 target_size=(299,299))

   Found 2528 validated image filenames belonging to 6 classes.

[15] train_generator.class_indices

   {'0': 0, '1': 1, '2': 2, '3': 3, '4': 4, '5': 5}
```

```
df_train
train_images
ID
Label
```



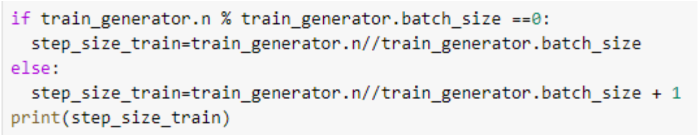
```
flow_from_dataframe(
    dataframe, directory=None, x_col='filename', y_col='class', weight_col=None,
    target_size=(256, 256), color_mode='rgb', classes=None,
    class_mode='categorical', batch_size=32, shuffle=True, seed=None,
    save_to_dir=None, save_prefix='', save_format='png', subset=None,
    interpolation='nearest', validate_filenames=True, **kwargs
)
```

https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator

43

Step 8: step_size_train

AUAOI Ex2.



```
if train_generator.n % train_generator.batch_size ==0:
  step_size_train=train_generator.n//train_generator.batch_size
else:
  step_size_train=train_generator.n//train_generator.batch_size + 1
print(step_size_train)
```

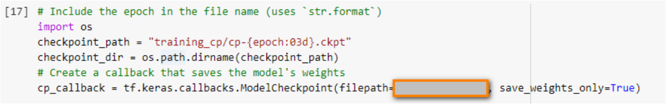
44

Step 9:

ModelCheckpoint

Callback to save the Keras model or model weights at some frequency.

AUAOI Ex2.



```
[17] # Include the epoch in the file name (uses `str.format`)
     import os
     checkpoint_path = "training_cp/cp-{epoch:03d}.ckpt"
     checkpoint_dir = os.path.dirname(checkpoint_path)
     # Create a callback that saves the model's weights
     cp_callback = tf.keras.callbacks.ModelCheckpoint(filepath=            , save_weights_only=True)
```

`checkpoint_path`

```
tf.keras.callbacks.ModelCheckpoint(
    filepath, monitor='val_loss', verbose=0, save_best_only=False,
    save_weights_only=False, mode='auto', save_freq='epoch', **kwargs
)
```

https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/ModelCheckpoint

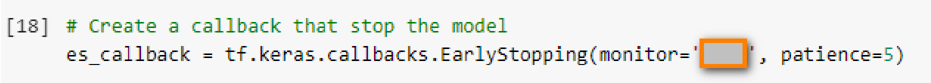
45

Step 10:

EarlyStopping

Stop training when a monitored metric has stopped improving.

AUAOI Ex2.



```
[18] # Create a callback that stop the model
     es_callback = tf.keras.callbacks.EarlyStopping(monitor='    ', patience=5)
```

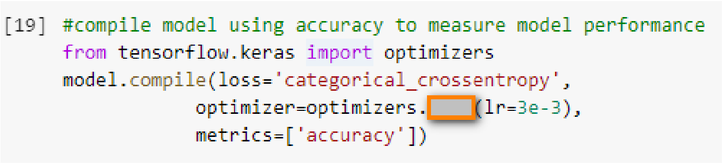
`loss`

```
tf.keras.callbacks.EarlyStopping(
    monitor='val_loss', min_delta=0, patience=0, verbose=0, mode='auto',
    baseline=None, restore_best_weights=False
)
```

46

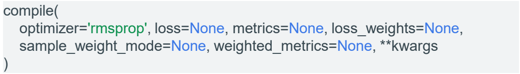
Step 11: Compile model

AUAOI Ex2.



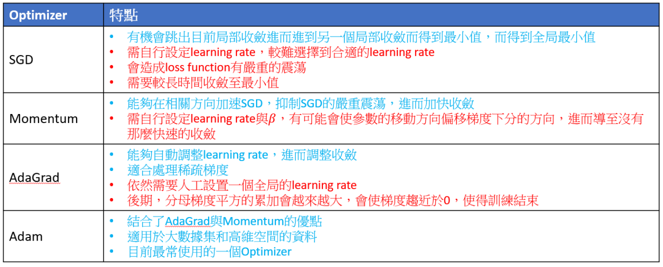
```
[19] #compile model using accuracy to measure model performance
     from tensorflow.keras import optimizers
     model.compile(loss='categorical_crossentropy',
                   optimizer=optimizers.    (lr=3e-3),
                   metrics=['accuracy'])
```

Adam/SGD



```
compile(
    optimizer='rmsprop', loss=None, metrics=None, loss_weights=None,
    sample_weight_mode=None, weighted_metrics=None, **kwargs
)
```

tf.keras.optimizers



Optimizer	特點
SGD	• 有機會跳出目前局部收斂進而進到另一個局部收斂而得到最小值，而得到全局最小值 • 需自行設定learning rate，較難選擇到合適的learning rate • 會造成loss function有嚴重的震蕩 • 需要較長時間收斂至最小值
Momentum	• 能夠在相關方向加速SGD，抑制SGD的嚴重震蕩，進而加快收斂 • 需自行設定learning rate與β，有可能會使參數的移動方向偏移梯度下分的方向，進而導至沒有那麼快速的收斂
AdaGrad	• 能夠自動調整learning rate，進而調整收斂 • 適合處理稀疏梯度 • 依然需要人工設置一個全局的learning rate • 後期，分母梯度平方的累加會越來越大，會使梯度趨近於0，使得訓練結束
Adam	• 結合了AdaGrad與Momentum的優點 • 適用於大數據集和高維空間的資料 • 目前最常使用的一個Optimizer

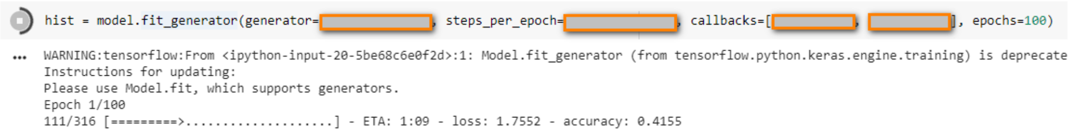
47

Step 12: Train model

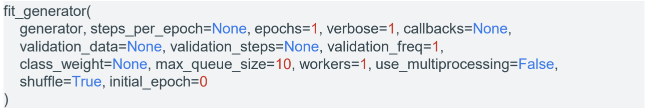
AUAOI Ex2.

cp_callback, es_callback

train_generator step_size_train



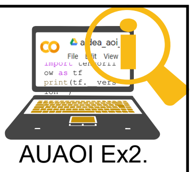
```
hist = model.fit_generator(generator=          , steps_per_epoch=          , callbacks=[          ,          ], epochs=100)

WARNING:tensorflow:From <ipython-input-20-5be68c6e0f2d>:1: Model.fit_generator (from tensorflow.python.keras.engine.training) is deprecate
Instructions for updating:
Please use Model.fit, which supports generators.
Epoch 1/100
111/316 [=========>....................] - ETA: 1:09 - loss: 1.7552 - accuracy: 0.4155
```

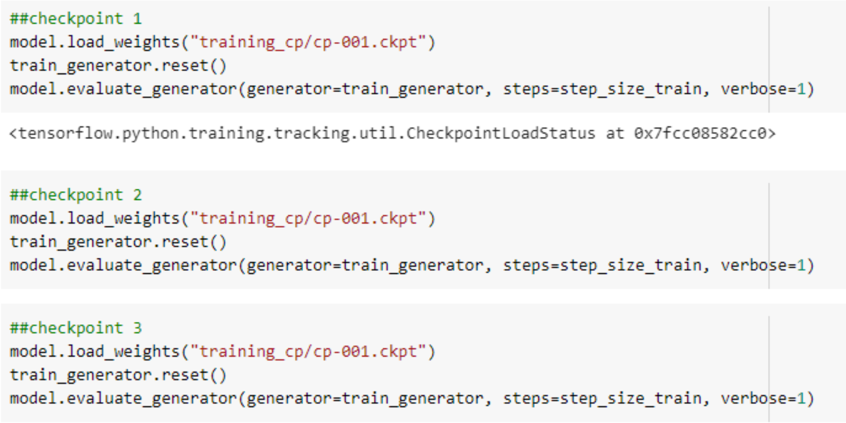


```
fit_generator(
    generator, steps_per_epoch=None, epochs=1, verbose=1, callbacks=None,
    validation_data=None, validation_steps=None, validation_freq=1,
    class_weight=None, max_queue_size=10, workers=1, use_multiprocessing=False,
    shuffle=True, initial_epoch=0
)
```

48

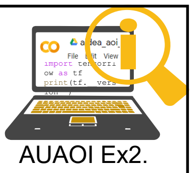
Step 13: Evaluate saved checkpoints

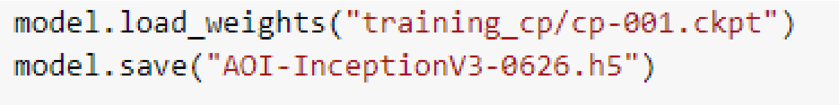

AUAOI Ex2.



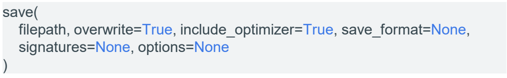
```
##checkpoint 1
model.load_weights("training_cp/cp-001.ckpt")
train_generator.reset()
model.evaluate_generator(generator=train_generator, steps=step_size_train, verbose=1)

<tensorflow.python.training.tracking.util.CheckpointLoadStatus at 0x7fcc08582cc0>

##checkpoint 2
model.load_weights("training_cp/cp-001.ckpt")
train_generator.reset()
model.evaluate_generator(generator=train_generator, steps=step_size_train, verbose=1)

##checkpoint 3
model.load_weights("training_cp/cp-001.ckpt")
train_generator.reset()
model.evaluate_generator(generator=train_generator, steps=step_size_train, verbose=1)
```

Step 14: Save the trained model


AUAOI Ex2.

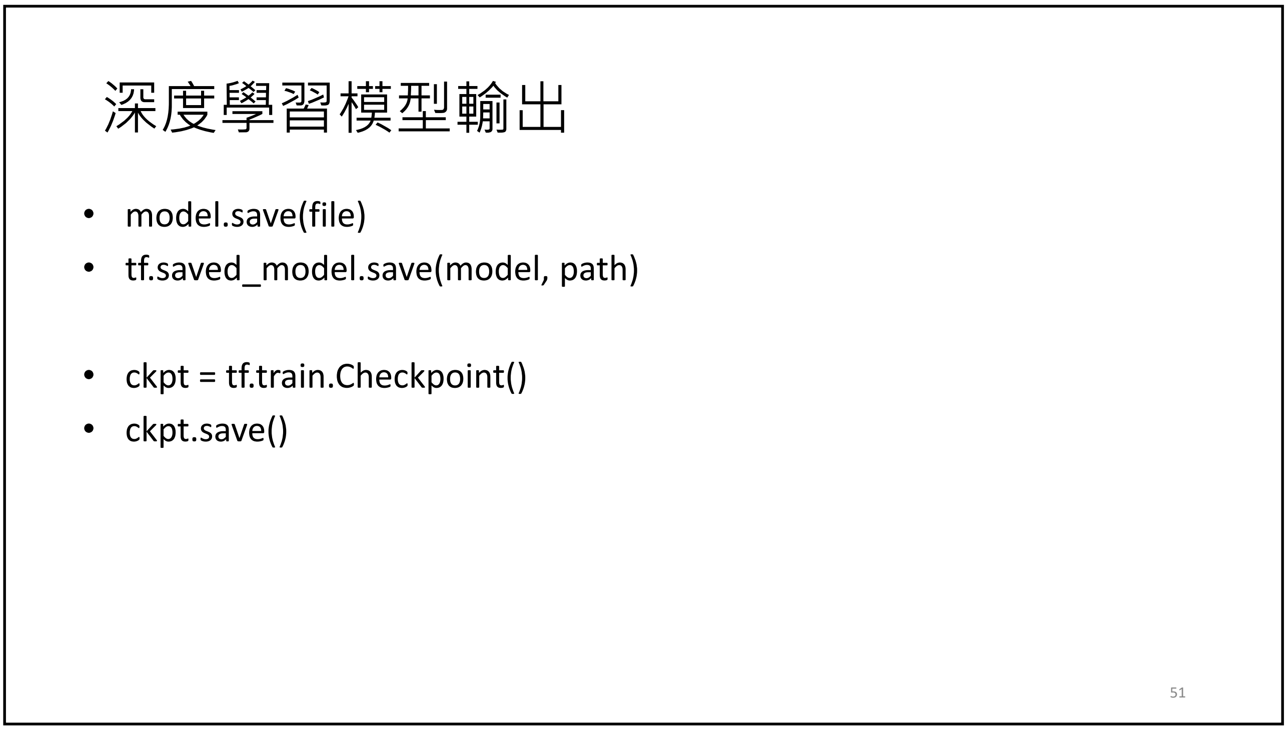


```
model.load_weights("training_cp/cp-001.ckpt")
model.save("AOI-InceptionV3-0626.h5")
```



```
save(
    filepath, overwrite=True, include_optimizer=True, save_format=None,
    signatures=None, options=None
)
```

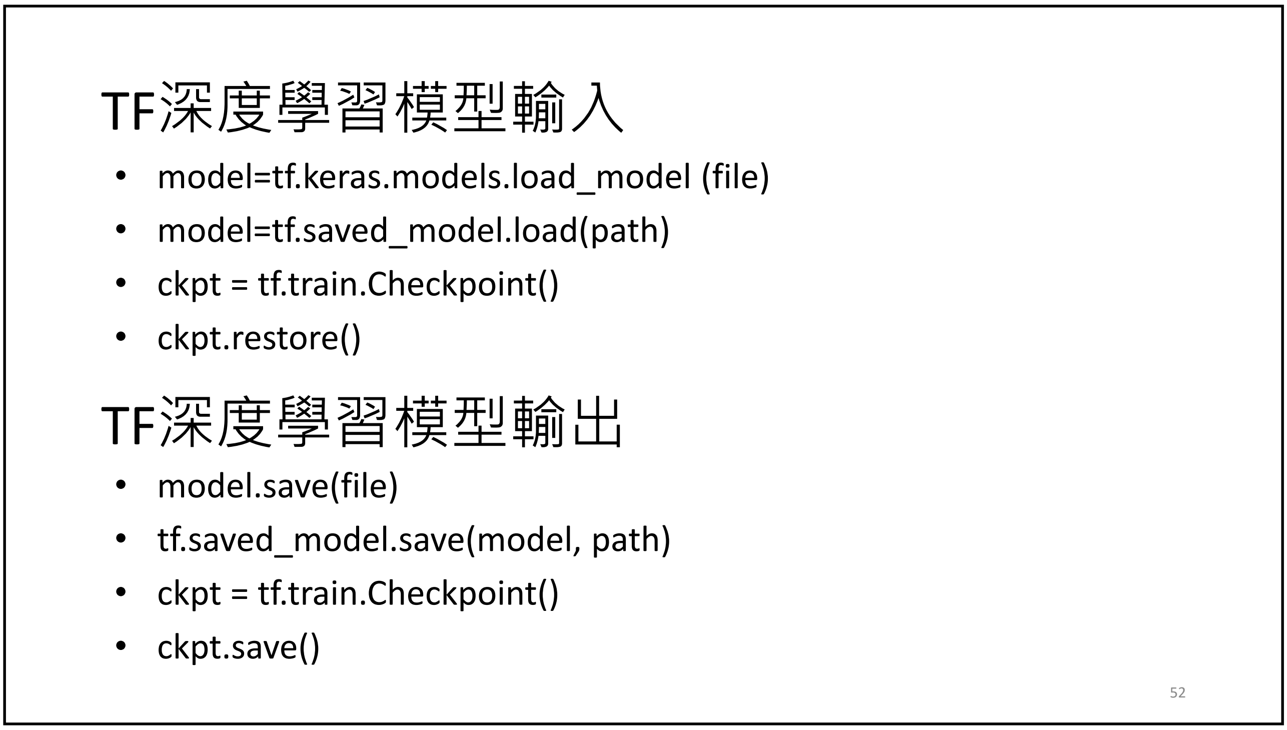
https://www.tensorflow.org/api_docs/python/tf/keras/Model#save



深度學習模型輸出

- model.save(file)
- tf.saved_model.save(model, path)

- ckpt = tf.train.Checkpoint()
- ckpt.save()



TF深度學習模型輸入

- model=tf.keras.models.load_model (file)
- model=tf.saved_model.load(path)
- ckpt = tf.train.Checkpoint()
- ckpt.restore()

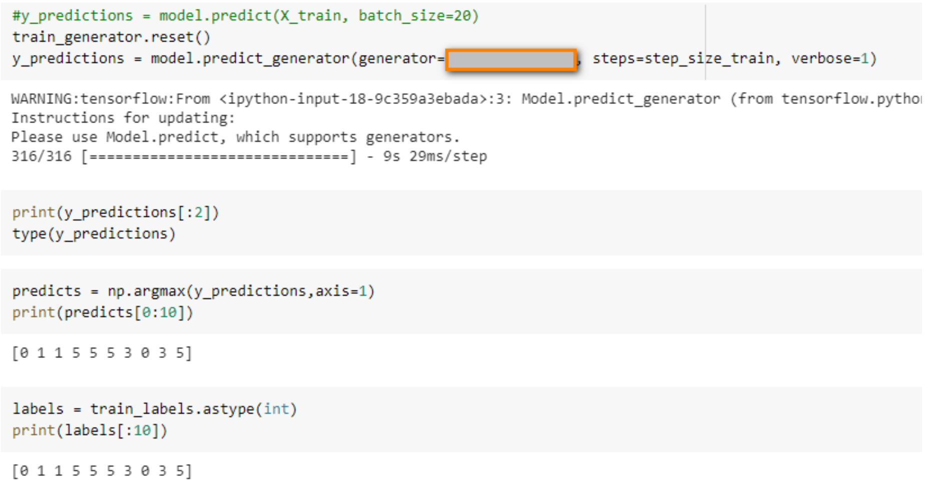
TF深度學習模型輸出

- model.save(file)
- tf.saved_model.save(model, path)
- ckpt = tf.train.Checkpoint()
- ckpt.save()

Step 15: Check training results

AUAOI Ex2.

train_generator



```
#y_predictions = model.predict(X_train, batch_size=20)
train_generator.reset()
y_predictions = model.predict_generator(generator=              , steps=step_size_train, verbose=1)
```

```
WARNING:tensorflow:From <ipython-input-18-9c359a3ebada>:3: Model.predict_generator (from tensorflow.pytho
Instructions for updating:
Please use Model.predict, which supports generators.
316/316 [==============================] - 9s 29ms/step
```

```
print(y_predictions[:2])
type(y_predictions)
```

```
predicts = np.argmax(y_predictions,axis=1)
print(predicts[0:10])
```

```
[0 1 1 5 5 5 3 0 3 5]
```

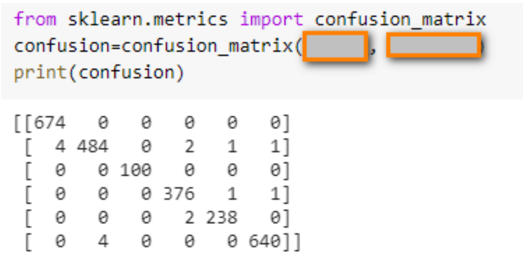
```
labels = train_labels.astype(int)
print(labels[:10])
```

```
[0 1 1 5 5 5 3 0 3 5]
```

Step 16: Analyze training results

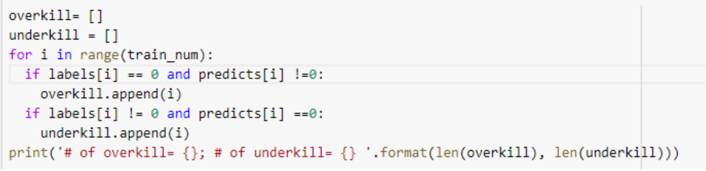
AUAOI Ex2.

labels predicts



```
from sklearn.metrics import confusion_matrix
confusion=confusion_matrix(      ,          )
print(confusion)
```

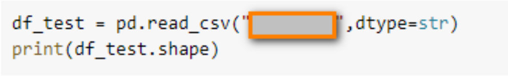
```
[[674   0   0   0   0   0]
 [  4 484   0   2   1   1]
 [  0   0 100   0   0   0]
 [  0   0   0 376   1   1]
 [  0   0   0   2 238   0]
 [  0   4   0   0   0 640]]
```



```
overkill= []
underkill = []
for i in range(train_num):
  if labels[i] == 0 and predicts[i] !=0:
    overkill.append(i)
  if labels[i] != 0 and predicts[i] ==0:
    underkill.append(i)
print('# of overkill= {}; # of underkill= {} '.format(len(overkill), len(underkill)))
```

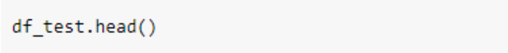
Step 17: Load the test set

AUAOI Ex2.

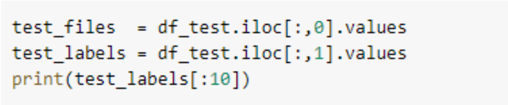


```
df_test = pd.read_csv("        ",dtype=str)
print(df_test.shape)
```

test.csv



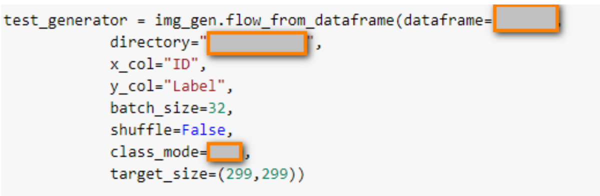
```
df_test.head()
```



```
test_files  = df_test.iloc[:,0].values
test_labels = df_test.iloc[:,1].values
print(test_labels[:10])
```

Step 18: Set up a test_generator with flow_from_dataframe

AUAOI Ex2.



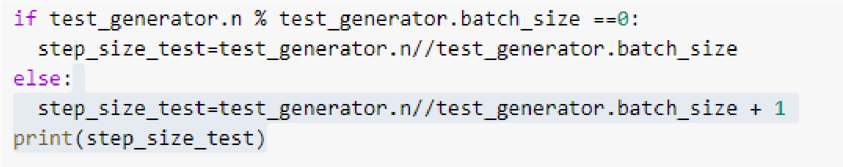
```
test_generator = img_gen.flow_from_dataframe(dataframe=      ,
            directory="          ",
            x_col="ID",
            y_col="Label",
            batch_size=32,
            shuffle=False,
            class_mode=    ,
            target_size=(299,299))
```

df_test
test_images

None

Step 19: step_size_test

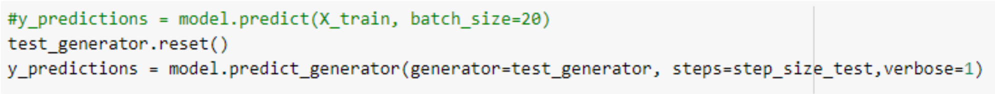
AUAOI Ex2.



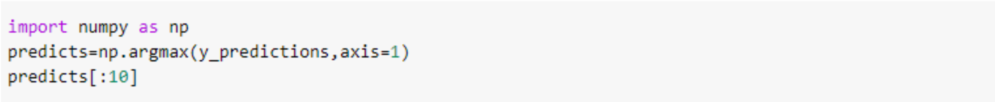
```
if test_generator.n % test_generator.batch_size ==0:
  step_size_test=test_generator.n//test_generator.batch_size
else:
  step_size_test=test_generator.n//test_generator.batch_size + 1
print(step_size_test)
```

Step 20: Check test results

AUAOI Ex2.



```
#y_predictions = model.predict(X_train, batch_size=20)
test_generator.reset()
y_predictions = model.predict_generator(generator=test_generator, steps=step_size_test,verbose=1)
```



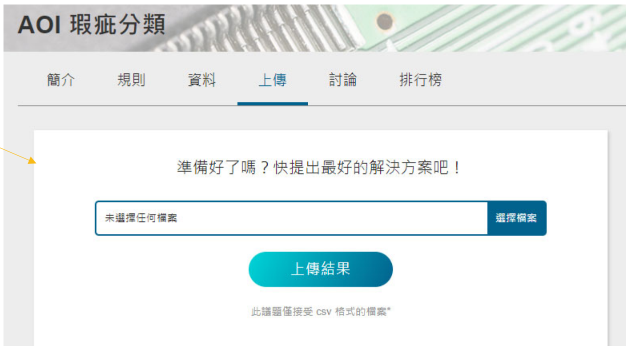
```
import numpy as np
predicts=np.argmax(y_predictions,axis=1)
predicts[:10]
```

Step 21:

AUAOI Ex2.

```
df_out = pd.DataFrame(df_test)
df_out.shape
```

```
df_out['Label'] = predicts
df_out.to_csv("0626-xception.csv", index=False)
```



準備好了嗎？快提出最好的解決方案吧！

未選擇任何檔案

此議題僅接受 csv 格式的檔案*

~~End~~

Thanks! Q&A

(1)Overfitting problem
(2)Training set/Validation set

行走江湖難免遇到八鴿...