

The background features a light gray dashed line forming a large circle. Various colored circles are scattered around: a large teal circle with a white center in the top-left, a small teal circle below it, a large lime green circle in the top-right, a small lime green circle below it, a large orange circle in the bottom-right, a small orange circle below it, a large green circle in the bottom-left, a small green circle below it, and a small pink circle in the middle-right.

# Edge AI

## 耐能棒使用介紹

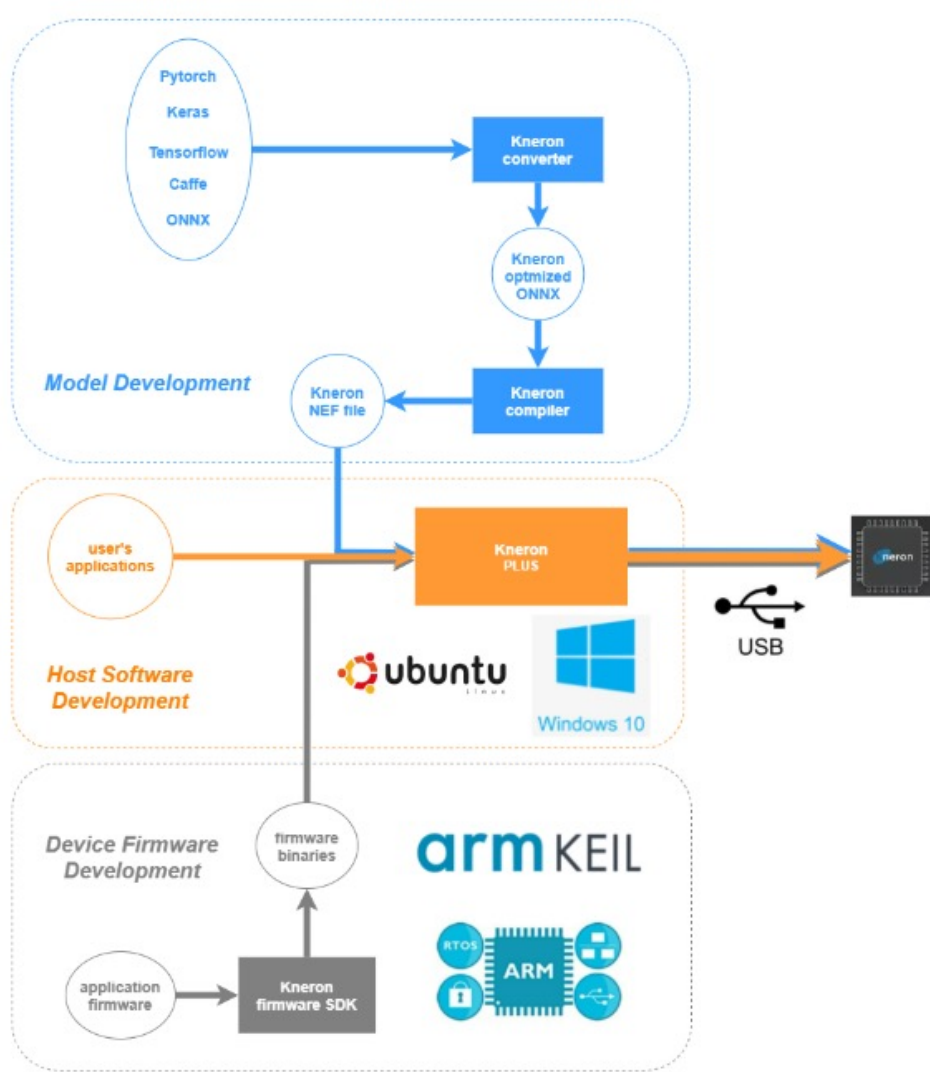
2022/05/25



# 耐能棒

## Kneron KL520 AI Dongle

- 結合耐能KL-520神經網路加速晶片 (NPU)
- 超低功耗的邊緣運算方案
- 高效能的神經網路推理運算能力
- 支持Win10, Linux 和樹莓派
- 支持Kneron KNEO 智能邊緣運算網路



The background is white and decorated with various geometric shapes. In the top left, there is a large orange circle with a dashed red outline, partially overlapping a solid yellow circle. Below the yellow circle is a small solid pink circle. In the top right, there is a solid green circle with a small white dot in the center, and below it is a solid yellow circle with a dashed yellow outline. In the bottom left, there is a solid green circle with a dashed green outline, and below it is a large solid yellow circle with a small solid cyan circle next to it. In the bottom right, there is a large solid cyan circle with a white dot in the center, and below it is a solid cyan circle with a dashed cyan outline. A large, faint dashed blue circle is centered in the background, framing the text.

# PART 1

## 邊緣運算的發展由來

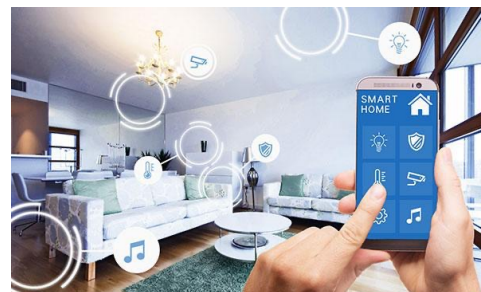
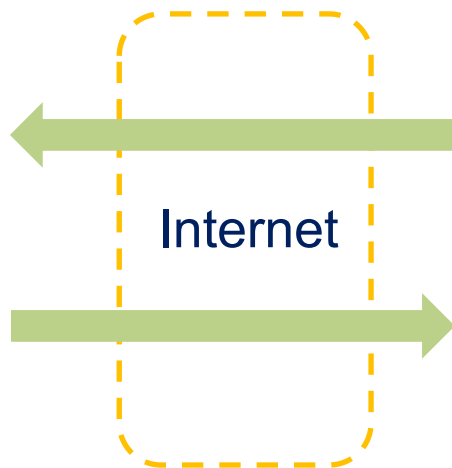
# 物聯網與人工智慧的興起 (AIoT)



# 物聯網與人工智慧的興起 (AIoT)



傳統作法：  
將訓練好的模型部署至雲端進行推論



# 雲端運算可能造成的問題

- 因大量的資料量傳輸，需要足夠的網路頻寬，成本高
- 可能因網路的延遲，無法即時得到模型分析結果
  - 自駕車或駕駛輔助系統延遲，車禍的機率提升
  - 工廠瑕疵檢測的時間成本增加，使生產效率降低
- 有網路安全的疑慮

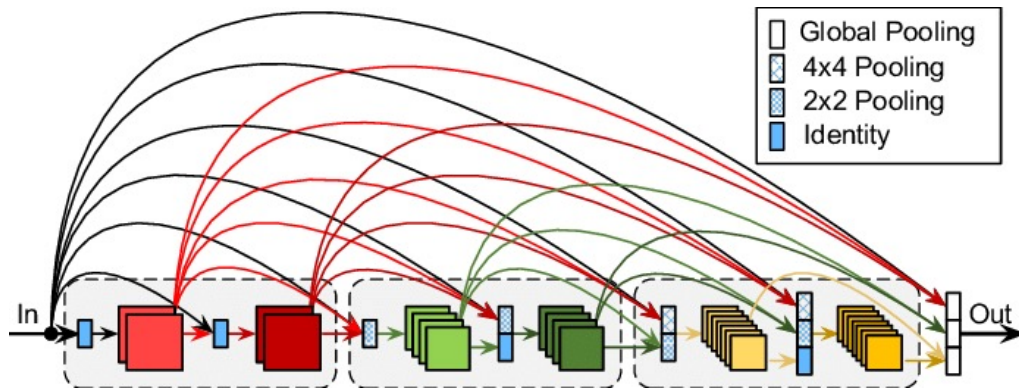
# 邊緣運算

- 是一種分散式的運算架構
- 運算過程盡可能靠近資料來源以減少延遲和頻寬使用
- 減少駭客入侵的機率，保護客戶隱私
- 由於近年硬體發展良好，使邊緣運算的可行性變高



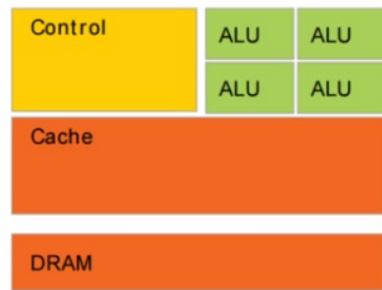
# 邊緣運算—Edge AI

- 可分為兩種情境：模型訓練(聯邦式學習)、**終端模型推論(今天議題)**
- 但現今的神經網路的架構越來越龐大，要使部屬至邊緣裝置的模型推論速度增加？



# 邊緣運算—Edge AI

- 壓縮模型，如剪枝、量化、蒸餾，但精度有可能下降
- 使用NPU(Neural Processing Unit)進行模型推論



CPU



GPU



FPGA(需硬體設計背景)

# 邊緣運算—Edge AI



各家廠牌邊緣裝置

The background is white and decorated with various colorful circles and dashed lines. In the top left, there is a large orange circle with a dashed red outline, overlapping a solid yellow circle. Below them is a small pink circle. In the top right, there is a green circle with a white center, a small orange circle, and a yellow circle with a dashed green outline. In the bottom left, there is a large yellow circle, a small teal circle, and a green circle with a dashed green outline. In the bottom right, there is a large teal circle with a white center, a small teal circle with a dashed blue outline, and a teal circle with a dashed blue outline. A large, faint dashed blue circle is centered behind the text.














# PART 2

## 耐能棒的使用流程

# 準備雲端資料

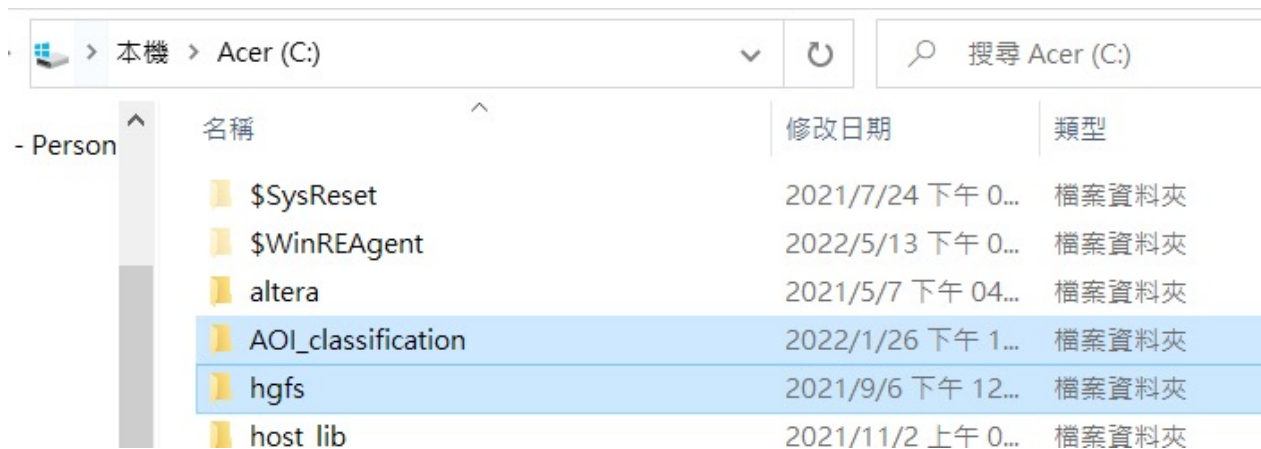
- 請先將 Aidea\_AOI 資料夾整份下載至本機，

並在雲端建立一份自己的資料夾，將以上三份檔案在上傳至此資料夾

與我共用 > Aidea_AOI ▾		    	
名稱 ▾	擁有者	我上次修改的時間	檔案大小
 hgfs	mumu		—
 AOI_classification	mumu		—
 train.csv 	mumu		47 KB
 train_images.zip 	mumu		341.5 MB
 Edge AI Lab-AOI-ToStudents.ipynb 	mumu		18 KB

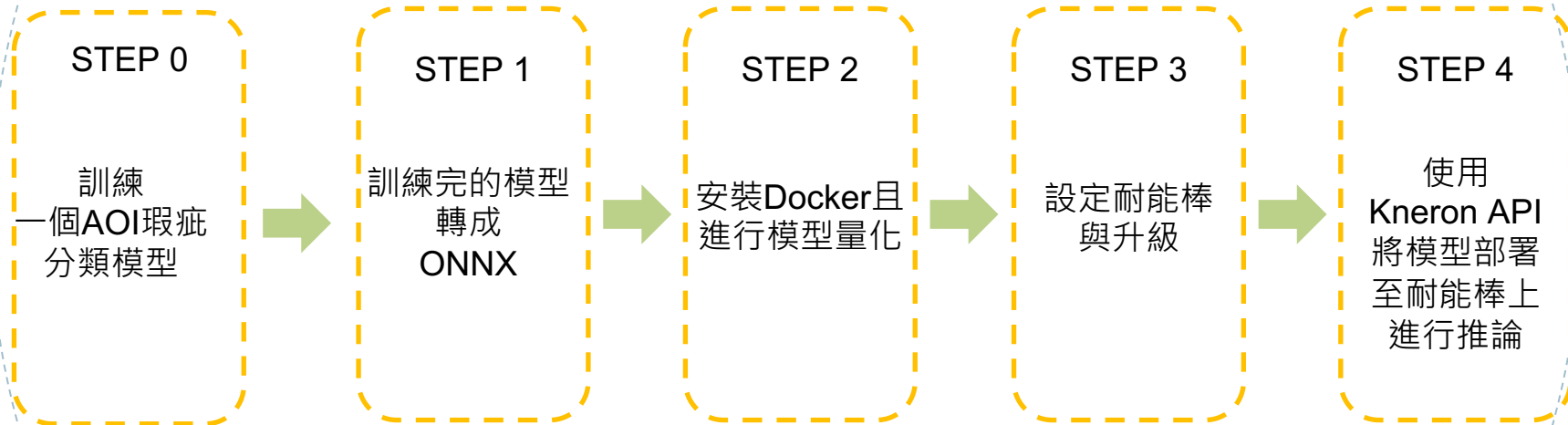
# 準備雲端資料

- 將 hgfs 與 AOI\_classification 資料夾從download移至 C 槽



本機 > Acer (C:)			
搜尋 Acer (C:)			
- Person	名稱	修改日期	類型
	\$SysReset	2021/7/24 下午 0...	檔案資料夾
	\$WinREAgent	2022/5/13 下午 0...	檔案資料夾
	altera	2021/5/7 下午 04...	檔案資料夾
	AOI_classification	2022/1/26 下午 1...	檔案資料夾
	hgfs	2021/9/6 下午 12...	檔案資料夾
	host lib	2021/11/2 上午 0...	檔案資料夾

# 流程步驟



# AOI瑕疵分類

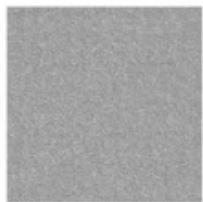
- 自動光學檢查 ( Automated Optical Inspection , 簡稱 AOI ) , 為高速高精度光學影像檢測系統 , 運用機器視覺做為檢測標準技術 , 可改良傳統上以人力使用光學儀器進行檢測的缺點
- 應用層面包括從高科技產業之研發、製造品管 , 以至國防、民生、醫療、環保、電力...等領域。
- 工研院電光所投入軟性電子顯示器之研發多年 , 在試量產過程中 , 希望藉由 AOI 技術提升生產品質。
- 本次邀請各界資料科學家共襄盛舉 , 針對所提供的 AOI 影像資料 , 來判讀瑕疵的分類 , 藉以提升透過數據科學來加強 AOI 判讀之效能。



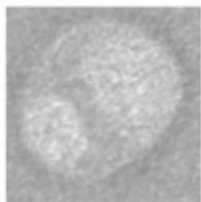
# 資料說明

- train\_images.zip : 訓練與測試影像資料
- train.csv: 包含2個欄位，ID和Label
- ID: 影像檔名
- Label: 瑕疵分類類別
- 0: normal
- 1: void
- 2: horizontal defect
- 3: vertical defect
- 4: edge defect
- 5: particle

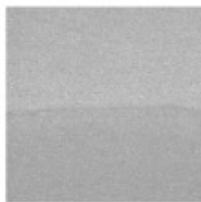
# 資料說明



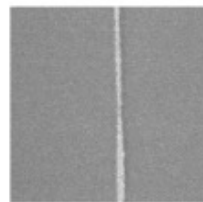
Normal



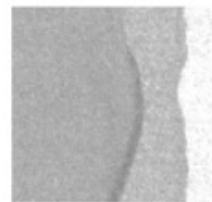
Void



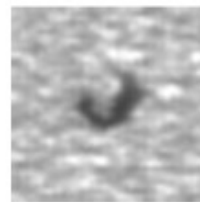
Vertical Defect



Horizontal Defect



Edge Defect



Particle

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

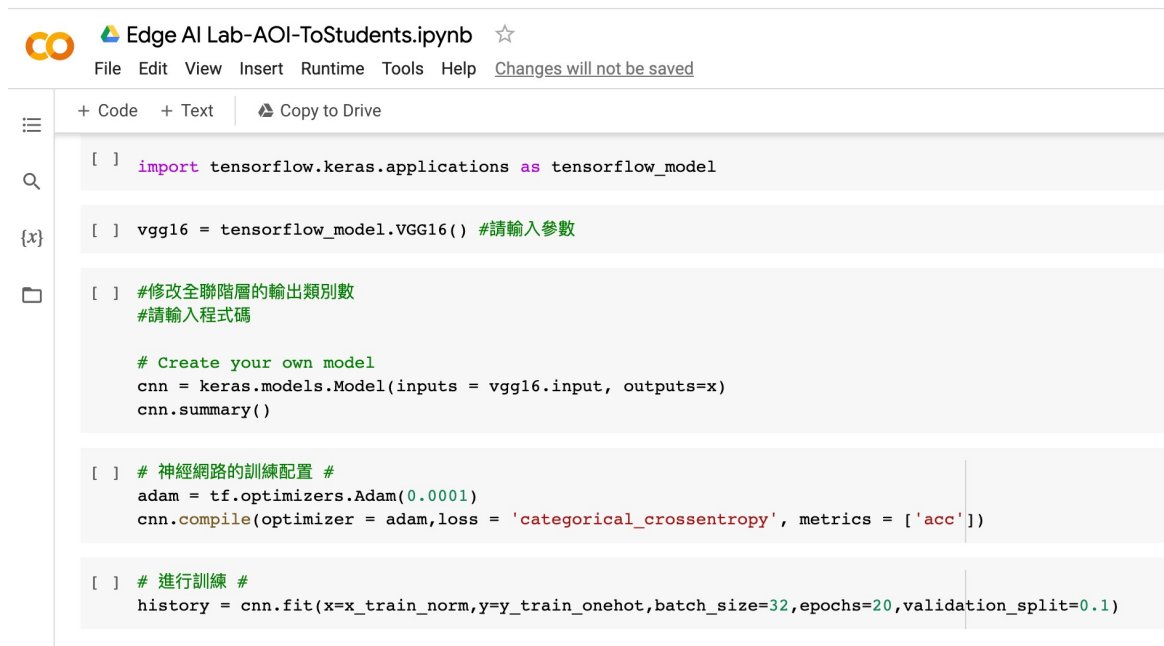
## 資料說明

train.csv

ID	Label
train_00000.png	0
train_00001.png	1
train_00002.png	1
train_00003.png	5
train_00004.png	5
train_00005.png	5
train_00006.png	3
train_00007.png	0
train_00008.png	3
train_00009.png	5
train_00010.png	3
train_00011.png	5
train_00012.png	3
train_00013.png	3
train_00014.png	1
train_00015.png	1
train_00016.png	1
train_00017.png	1

# STEP 0

- 使用Colab開啟Edge AI Lab-AOI-ToStudents.ipynb



```
[ ] import tensorflow.keras.applications as tensorflow_model

[ ] vgg16 = tensorflow_model.VGG16() #請輸入參數

[ ] #修改全聯階層的輸出類別數
#請輸入程式碼

# Create your own model
cnn = keras.models.Model(inputs = vgg16.input, outputs=x)
cnn.summary()

[ ] # 神經網路的訓練配置 #
adam = tf.optimizers.Adam(0.0001)
cnn.compile(optimizer = adam,loss = 'categorical_crossentropy', metrics = ['acc'])

[ ] # 進行訓練 #
history = cnn.fit(x=x_train_norm,y=y_train_onehot,batch_size=32,epochs=20,validation_split=0.1)
```

# STEP 0

- 載入 AOI瑕疵檢測 資料集的csv，並查看每類資料量是否不平均

```
#將壓縮檔複製到/content
! cp "/content/drive/MyDrive/train_images.zip" /content/

#解壓縮訓練集
! unzip /content/train_images > data_unzip.log

#read csv
import pandas as pd
AOI_data = pd.read_csv('/content/drive/MyDrive/AIdea_AOI/train.csv')

#查看訓練集每類別圖片數量並且分類
label = []
for i in range(6):
    temp = AOI_data[AOI_data['Label'] == i]
    label.append(temp.reset_index())
    print('第' + str(i) + '類張數: ' + str(len(label[i])))

第0類張數: 674
第1類張數: 492
第2類張數: 100
第3類張數: 378
第4類張數: 240
第5類張數: 644
```

# STEP 0

- 分出測試集(此程式碼不可更動，為評分用測試集)

```
import cv2
train_images = []
train_label = []
test_images = []
test_label = []
for i in range(6):
    #讀取圖片測試集圖片(這邊的程式碼不可更動)
    images_temp = []
    label_temp = [i] * 20
    for j in range(20):
        img = cv2.imread('/content/train_images/' + label[i]['ID'][j])
        images_temp.append(cv2.resize(img, (224, 224), cv2.INTER_AREA))
    test_images += images_temp
    test_label += label_temp
```

影像前處理可自由修改

# STEP 0

- 接著將剩餘圖片進行增量，使訓練集每類張數相同

```
import cv2
train_images = []
train_label = []
test_images = []
test_label = []
for i in range(6):
    #讀取圖片測試集圖片(這邊的程式碼不可更動)
    images_temp = []
    label_temp = [i] * 20
    for j in range(20):
        img = cv2.imread('/content/train_images/'+label[i]['ID'][j])
        images_temp.append(cv2.resize(img, (224, 224), cv2.INTER_AREA))
    test_images += images_temp
    test_label += label_temp

#將剩餘圖片進行增量，使訓練集每類張數相同
#請輸入程式碼

train_images += images_temp
train_label += label_temp
```

(訓練集張數可由同學決定)

# STEP 0

- 將格式轉成numpy，並且將訓練集進行shuffle

```
#將 list 轉成 array
import numpy as np
from sklearn.utils import shuffle
x_train = np.array(train_images)
x_test = np.array(test_images)
y_train = np.array(train_label)
y_test = np.array(test_label)

#將訓練集進行shuffle
import random
x_train , y_train = shuffle(x_train, y_train, random_state=random.seed())
```



# STEP 0

- 模型不一定要VGG16

```
import tensorflow as tf
import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten
from tensorflow.keras.layers import Conv2D, MaxPooling2D

import tensorflow.keras.applications as tensorflow_model

vgg16 = tensorflow_model.VGG16( ) #請輸入參數

#修改全聯階層的輸出類別數
num_classes = 6
x = vgg16.layers[-1].output
x = Flatten(name='flatten')(x)
x = Dropout(0.5)(x)
x = Dense(num_classes, activation='softmax', name='predictions')(x)

# Create your own model
cnn = keras.models.Model(inputs = vgg16.input, outputs=x)
cnn.summary()
```

# STEP 0

- 同學們的測試準確率要比以下準確率(89%)高哦!

```
# evaluate
test_loss, test_val = cnn.evaluate(x_test_norm, y_test_onehot)
print(' 測試資料損失值:', test_loss)
print(' 測試資料準確度:', test_val)
```

```
4/4 [=====] - 0s 34ms/step - loss: 0.3151 - acc: 0.9000
測試資料損失值: 0.3150562047958374
測試資料準確度: 0.8999999761581421
```

# HW

- 請每位同學自己訓練AOI瑕疵分類模型
- 將準確度截圖如下，準確度 > 89%

```
# evaluate
test_loss, test_val = cnn.evaluate(x_test_norm, y_test_onehot)
print(' 測試資料損失值:', test_loss)
print(' 測試資料準確度:', test_val)

4/4 [=====] - 0s 34ms/step - loss: 0.3151 - acc: 0.9000
測試資料損失值: 0.3150562047958374
測試資料準確度: 0.8999999761581421
```

- 上傳截圖與程式碼.py (or .ipynb)
- **Deadline: 2022/6/21 23:59**