

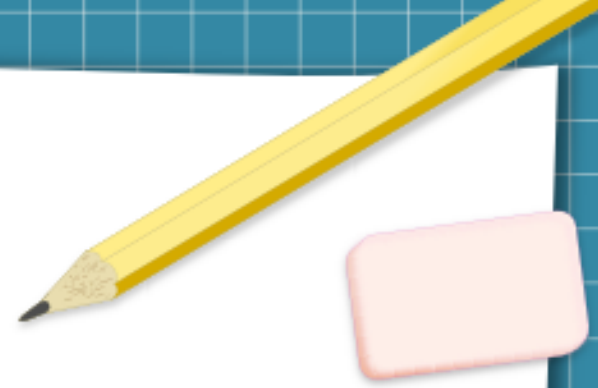


Aidea Edge AOI 瑕疵檢測

自動光學檢查（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。

執行環境



硬體：使用 NVIDIA Jetson Nano ，限制 3GB RAM ，最多執行 20 分鐘。

軟體：程式使用 Docker 來執行，Dockerfile 請參考下載檔案。

注意事項：

程式執行過程中，若發生 out-of-memory ，超過 20 分鐘，視為失敗。

軟體框架僅限於 Dockerfile 中所提供，若使用未提供之框架 / 軟體套件，會造成程式錯誤。

Docker file :

FROM nvcr.io/nvidia/l4t-tensorflow:r32.5.0-tf2.3-py3

RUN python3 -m pip install --upgrade Pillow

RUN python3 -m pip install --upgrade glob2

RUN mkdir -p /app

WORKDIR /app

ADD . /app

競賽模式

- 在平台上限制程式的記憶體使用 **(3GB)**，也限制最長的執行時間 **(20 分鐘)**，若超過限制，將會視為失敗。

- 加權總分滿分為 100，佔比如下：

- Accuracy 項目：60
- Time (loading model) 項目：10
- Time (inference) 項目：30

- Time (loading model) 項目計分

起始時間 (秒) (包含)	結束時間 (秒) (不包含)	分數
0	8	10
8	16	9
16	24	8
24	32	7

- Time (inference) 項目計分

起始時間 (秒) (包含)	結束時間 (秒) (不包含)	分數
0	15	30
15	30	27
30	45	24
45	60	21

模型打包成 zip 檔上傳

- 程式撰寫請參考 [program_template.zip](#) 中的檔案。
 - 程式執行的進入點是 [start.py](#)，但您無須、也不能修改此檔案。
 - 請修改 [toolkit.py](#) 程式，並將您的功能 implement 於此檔案中。

```
# Import your library

class Toolkit(object):
    def __init__(self):
        # No code is allowed in this function
        pass

    def load_model(self):
        # Implement here

    def perform_inference(self):
        # Implement here
```

模型調整

- 使用簡易 **CNN** 架構觀察訓練難易度 - 得分 **84.9**

Accuracy : 0.865

Size : 10M

Time (inference) : 0:00:34

Time (loading model) : 0:00:11

- 增加 **Conv** 層 (參考使用 **VGG**) - 得分 **85.7**

Accuracy : 0.895

Size : 3M

Time (inference) : 0:00:34

Time (loading model) : 0:00:17

- 發現雖然準確率提高
但多增設的 **conv** 層，讓 **loading time** 也隨之增加

模型調整 - 2

- 改善圖片讀取方式和使用前處理 - 得分 **89.1**

Accuracy : 0.935

Size : 5M

Time (inference) : 0:00:30

Time (loading model) : 0:00:10

- 發現使用旋轉會降低準確率，也觀察 **AOI** 照片拍攝視固定方式拍攝。
所以最後將旋轉角度前處理拿掉
- 刪除更多 **conv** 層 (有效減少 **inference time** 但準確率下降) - 得分 **89.7**

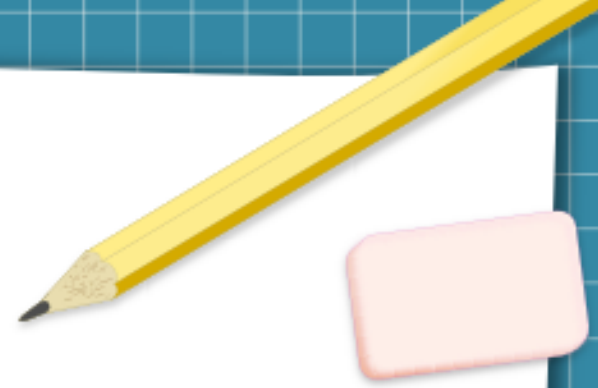
Accuracy : 0.895

Size : 10M

Time (inference) : 0:00:26

Time (loading model) : 0:00:10

- 發現雖然準確率降低，但 **inference** 時間減少反而總分增加



模型調整 - 3

- 減少及改善 **conv** 層 - 得分 **90.6**

Accuracy : 0.96

Size : 3M

Time (inference) : 0:00:38

Time (loading model) : 0:00:12

- 發現使用 **filter** 數量越多會提高準確率，但也會影響 **inference time**
- 繼續調整 **conv** 層並控制 **acc** 不要調降 - 得分 **96.3**
 - 轉換使用 **tf.data** 方式讀取資料

Accuracy : 0.955

Size : 517K

Time (inference) : 0:00:10

Time (loading model) : 0:00:13

- 使用 **tf.data** 可以快速讀高讀取數度但整體架構重新建構花了不少心力

比賽結果

- 最終成績 - 97.8

Accuracy : 0.98

Size : 8M

Time (inference) : 0:00:13

Time (loading model) : 0:00:11

- 在這次比賽當種使用 **windows** 安裝 **ubuntu** 雙環境下執行，
- 並使用 **Dockerfile** 執行 **.py** 當查看程式狀況，
- 調整模型中卷基層參數並觀察反應 (參考 **VGG** 內容)，
- 手寫 **learning rate** 起伏調整參數，增加準確率穩定性
- 使用 **tf.data** 架構大幅度減少 **inference time**
- 在有限的資源及框架中找出最適合的模型，
配合競賽方環境建構模型訓練環境，
採用多種不同的讀取寫入及儲存方式，
並找出現階段最適合的模型架構