人工智慧導論及實作期末報告

一、Group info:

Team names: TEAM_5655,TEAM_5576

Final Score: 0.675906

Members: 楊憲閔、李以路、張郡中

二、Leaderboard:

29	TEAM_5655	1	9	0.675906	5/26/2024 6:44:38 PM
43	TEAM_5576	1	4	0.499079	5/25/2024 12:58:33 PM

三、Approaches:

首先是切割 training data 與 validation data，作為評估模型的效果

```
# 指定 training set 和 validation set 的路徑
tr_image_root = "/kaggle/input/aicup-data/train/train/Training_dataset/img"
tr_label_root = "/kaggle/input/aicup-data/train/train/Training_dataset/label_img"

train_image_list = sorted([os.path.join(tr_image_root, f) for f in os.listdir(tr_image_root) if f.endswith('.jpg')])
train_label_list = sorted([os.path.join(tr_label_root, f) for f in os.listdir(tr_label_root) if f.endswith('.png')])

# # 因需切割 train data 為 train 與 validation，所以需先讀入data並切割
# # 讀資料
# import glob
# # 找出 train_path 底下所有檔案
# train_image_list = glob.glob(os.path.join(tr_image_root , '*.*') , recursive = True)
# train_label_list = glob.glob(os.path.join(tr_label_root , '*.*') , recursive = True)
l = list(zip(train_image_list , train_label_list))

# 切割
from sklearn.model_selection import train_test_split
train_path_list , val_path_list = train_test_split(l , test_size = 0.2 , random_state = 42)
```

嘗試的方法可以分為前中後期作為討論

1. 前期:

前期的時候不知道該怎麼下手，因此先從範例程式碼的參數做調整試試

a. 先不改模型、超參數等等做訓練:

2024-05-21 12:26:54	0.454444	0.0	Scoring success.

b. 降低 learning rate:

嘗試降低 learning rate 到 0.0001，其餘不變

2024-05-21 06:21:36	0.481996	0.0	Scoring success.

c. 加大 epoch:
尝試加大 epoch 數，從 30 加大到 60，發現其實沒什麼效果

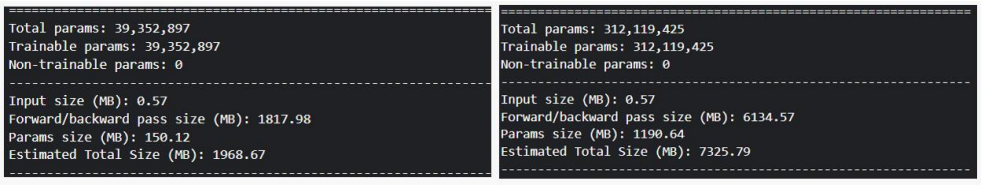
2024-05-26 11:39:16	0.499079	0.0	Scoring success.

2. 中期:
中期的時候思考到微調參數應該沒有什麼大幅度的增長，因此改以嘗試修改此 model 或是更換其他 model

a. 先更改範例程式碼裡的 backbone，原先是 resnet34，先改成 resnet18，一來可以測試修改 backbone 是否有效，二來也能讓訓練快些(因為要學習的參數降低了)，結果分數還是在 0.4 多徘徊

submission_file.zip resnet18_50_epoch 只有改train與test路徑 上傳成員 楊 憲閔	2024-05-22 09:44:54	0.471610	0.0	Scoring success.

b. 認為參數減少影響了他的表現，因此將 backbone 改為 resnet50，參數量約變為 resnet34 的 10 倍，訓練時長也增長了不少，結果分數還比以前低



（左圖為 resnet34，右圖為 resnet50）

submission_file.zip resnet50_15epochs 上傳成員 楊 憲閔	2024-05-24 03:17:00	0.446967	0.0	Scoring success.

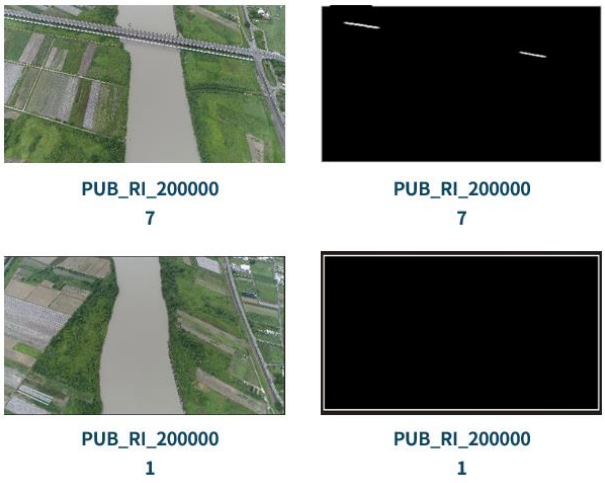
c. 認為此 model 好像不適合此比賽，翻閱了比賽的討論區，思考語意分割或許能有效解決此問題，因此就查詢關於語意分割的模型，發

現 Unet 應該不錯，就直接套用 Unet 上去，分數確實有提升但並沒有顯著上升(註 1)

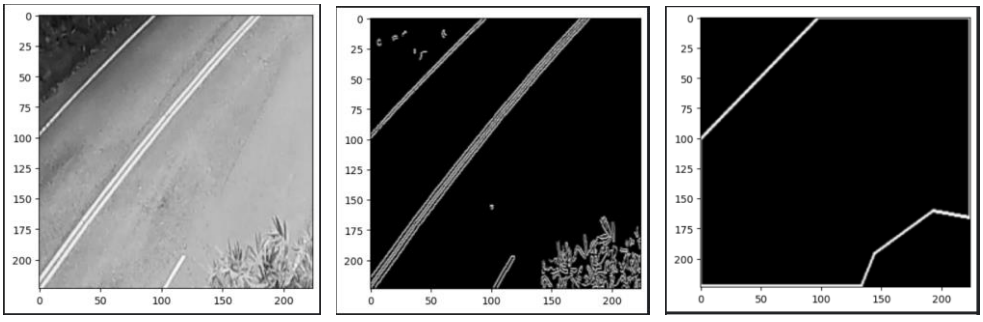
submission_file (2).zip unet_用resnet50 loss = 1.2798 上傳成員 楊 憲閔	2024-05-24 05:05:12	0.513418	0.0	Scoring success.

3. 後期:

經過了前面那些嘗試後，認為好像哪裡怪怪的，因此就翻了一下上傳檔案，確認是否輸出的是我們要的，結果發現模型根本沒學到



因此我們就想到若模型不知道該學哪邊，就告訴模型該學習哪裡即可，因此就想到先對輸入進行 Edge detection(canny，註 2)，再餵給模型



（由左而右分別為輸入的灰階圖、經過 Edge detection 的圖、label）

```
        image = cv2.imread(self.images[index] , cv2.IMREAD_GRAYSCALE)
#         image = Image.open(self.images[index]).convert('RGB')# 打開圖像文件，轉換為RGB
#         gray_image = Image.open(self.images[index]).convert('L')# 打開圖像文件，轉換為RGB
#         label = Image.open(self.labels[index]).convert('L')# 打開圖像文件，轉換為灰階影像
        label = cv2.imread(self.labels[index] , cv2.IMREAD_GRAYSCALE)

        # 若為訓練模式，進行資料擴增
        kernel_size = 3
        image = cv2.GaussianBlur(image,(kernel_size, kernel_size), 0)
        image = cv2.Canny(image , 100 , 200 , L2gradient = True)
```

（canny）

a. 先從範例程式搭配 canny 試試看，將其 backbone 改回 resnet34，並修改 dataset 中對圖片的前處理(canny)，並修改輸入通道=1(因為輸入變為灰階圖而非 RGB 圖)，分數而有了明顯提升

submission_file (4).zip simple_model_resnet34 先對其進行 canny loss = 1.3681 上傳成員 楊 憲閔	2024-05-25 09:06:00	0.639538	0.0	Scoring success.

b. 將 model 改為 Unet，一樣搭配 canny，其餘不變，得到比範例程式搭配 canny 還高些的分數，並嘗試加大 epoch，發現已經上不去了

submission_file (7).zip unet_resnet50_150epoch loss = 1.3244 上傳成員 楊 憲閔	2024-05-26 06:44:38	0.675905	0.0	Scoring success.
submission_file (6).zip unet_resnet50_100epoch_canny loss = 1.3258 上傳成員 楊 憲閔	2024-05-26 04:26:35	0.675846	0.0	Scoring success.
submission_file (5).zip unet_resnet50_canny loss = 1.3463 上傳成員 楊 憲閔	2024-05-25 10:59:06	0.669649	0.0	Scoring success.

四、Discussion or issue:

1. 再次檢查 testing dataset 與 submission file，可發現 model 有把注意力集中了，雖然有些 testing dataset 畫出來還是只有邊框有白線或是只有一點點線條，但比例明顯下降許多

(左圖為成功繪出的案例，右圖為無法繪出的案例)

可能的方法:未來可以試試修改參數，或是進行較多樣的 data augmentation 來獲得更多的 training data

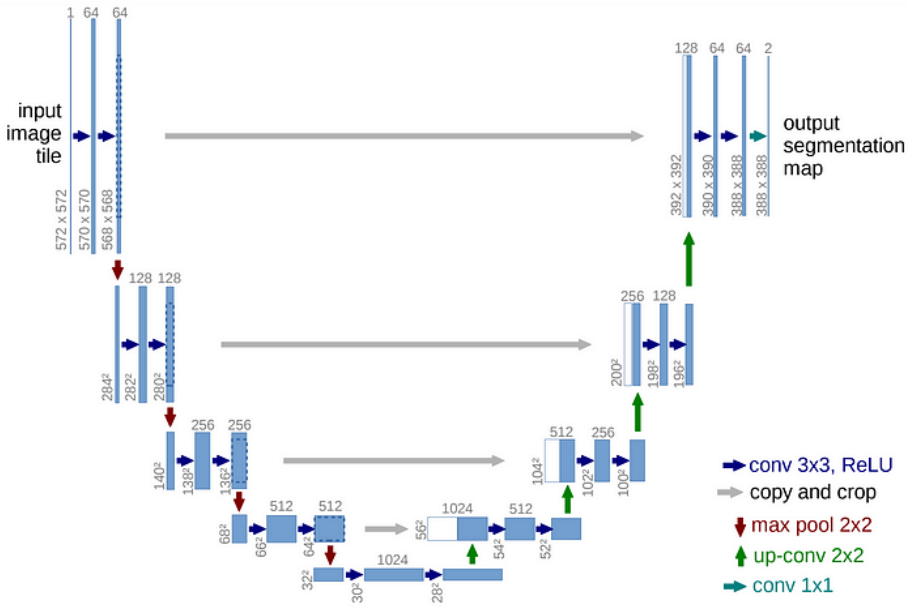
2. 欲使用 canny 會使得只能用 cv2 讀入、進行邊緣偵測，在 data augumentation 時會報錯，導致無法對 training data 進行 data augumentation

可能的方法:在範例程式中，是利用 Image.open()進行讀檔，然後對其進行 data augumentation，其中此檔的類型為 PIL 檔，與 cv2 讀進來的格式(ndarray)不同，因此可以先對其進行轉檔，轉成 PIL 檔應該就能與範例程式一樣進行 data augumentation

3. 在這次的比賽中，我們學到了面對問題時需要先去分析問題，要用什麼方法才能達到我們要的，就例如說這次邊緣偵測使我們能較為掌握到問題所要的，而非模型表現不好就換模型，導致浪費了許多時間

五、備註:

註 1:Unet 架構



可以分為前段(Encoder)、中段(Bridge)與後段(Decoder):

Encoder:圖片進入模型後，持續對其進行卷積運算，來萃取出圖片中的重要特徵(低維向量)，以傳到接下來的 bridge 做傳遞

Bridge:作為前段與後段的聯繫者，將含有重要特徵的低維向量傳給 Decoder，來讓 Decoder 進行後續的行為

Decoder:藉由 Encoder 萃取出的重要特徵，來重建成與原圖一樣大小的新圖像，並測試是否能顯著地還原，並用輸出與原圖做對比來進行學習

其中最重要的是「Skip Connection」，會在 Decode 的過程將細微特徵(Decode 時 upsampling 的結果)與粗糙特徵(Encode 過程中的成果)拼接，用以避免遺失重要資訊，進而提高辨識的品質

註 2:canny 邊緣偵測

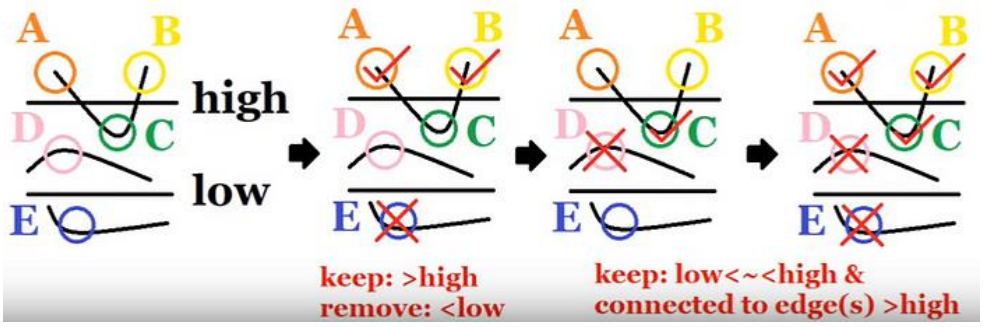
是一個邊緣偵測演算法，優點是錯誤率低、定位準確、解析度高，實際上是經過了以下四步驟:

Filter:運用 Gaussian Filter 濾除雜訊

Compute Gradient:透過 Sobel 濾波器濾出邊緣的強度及方向

Non-maximum suppression:尋找最大梯度方向與最大值抑制

connect weak edge:將線連起來，設定 high threshold 與 low threshold，找出邊緣



六、參考資料:

https://tomohiroliu22.medium.com/%E6%B7%B1%E5%BA%A6%E5%AD%B8%E7%BF%92paper%E7%B3%BB%E5%88%97-05-u-net-41be7533c934

https://martin12345m.medium.com/3d-unet-%E5%B0%8F%E7%B0%A1%E4%BB%8B-%E6%9E%B6%E6%A7%8B-1-%E5%BE%9E2d%E9%96%8B%E5%A7%8B-669cacd2f813

https://hackmd.io/@tarostudent99/BkN18VVJu

https://medium.com/@bob800530/opencv-%E5%AF%A6%E4%BD%9C%E9%82%8A%E7%B7%A3%E5%81%B5%E6%B8%AC-canny%E6%BC%94%E7%AE%97%E6%B3%95-d6e0b92c0aa3