

AI CUP 2024 春季賽

以生成式 AI

建構無人機於自然環境偵察時所需之導航資訊

競賽 I — 影像資料生成競賽

報告說明文件

- ◆ 參賽隊伍需詳細說明系統的流程、演算法、工具與外部資源、**技術模型原創性或改良成效等。請注意，程式之可執行性及可驗證性將影響報告評分。**主辦單位會請專業學者、專家仔細審查，若發現分數有問題或方法說明不清之部分，請參賽隊伍補充說明，經發現有違規者，將取消獎項資格。
- ◆ 請使用 A4 紙直式打字，中文字體使用標楷體，英文、數字與符號使用 Times New Roman 字體。
- ◆ 版面設定：邊界上下各 2.54CM，左邊 3.17CM、右邊 3.0CM。
- ◆ 字體大小：題目 20（粗體）、內文 12，單行間距。
- ◆ 作者聯絡資料表：詳細資料請填寫於附件的「作者聯絡資料表」。
- ◆ 須依照以下大綱及內容說明撰寫。不可自訂標題、修改內容順序，或合併段落，但可搭配圖片說明。
- ◆ 附件空白報告範本可參考使用。
- ◆ **報告大綱與撰寫規定（字數不得低於各段落規定下限）**

壹、環境

建議字數：100~600 字。

內容規定：請說明使用的作業系統、語言、套件（函式庫）、預訓練模型、額外資料集等。如使用預訓練模型及額外資料集，請逐一列出來源。

貳、演算法與模型架構

建議字數：300~1200 字。

內容規定：說明演算法設計、模型架構與模型參數，包括可能使用的特殊處理方式。

參、技術模型原創性或改良成效

建議字數：200~1200 字。

內容規定：說明演算法、模型之原創性或者改良外部資源哪一部分後所帶來之成效。

肆、資料分析與處理過程

建議字數：200~1500 字。

內容規定：說明對資料集的理解與後續處理或擴增的方式，例如對資料集包含的影像內容、資料可能的刪減、更正或增補。

伍、訓練方式

建議字數：300~1000 字。

內容規定：說明模型的訓練方法與過程。

陸、結果分析與結論

建議字數：300~2500 字。

內容規定：分析所使用的模型及其成效，簡述未來可能改進的方向。分析必須附圖，可將幾個成功和失敗的例子附上並說明之。

柒、程式碼(未於 2024/06/04 23:59 前繳程式碼連結者，將失去獲頒獎金/獎狀資格)

內容規定：實作程式碼與本報告文件需釋出至程式碼託管平台(如 github)並於此提供下載連結。如檔案太大以致託管平台不接受，需額外上傳至其他網站並提供下載連結。實作程式碼包含前處理程式碼、訓練程式碼、生成程式碼、各項參數之設定(包括訓練權重)等。程式碼應附 README.md 檔案交代安裝配置環境，重要模塊輸出/輸入，以讓第三方使用者可以除錯、重新訓練與重現結果。繳交前請確認連結有效且有開啟瀏覽權限，如連結失效視同未交。

捌、使用的外部資源與參考文獻及生成式語言模型 (如 ChatGPT 等)、生成式繪圖模型 (如 Midjourney) 使用說明 (若參賽隊伍無使用生成式語言模型則無須撰寫)

內容規定：

一、參考文獻請以 APA 格式為主。

二、「生成式模型使用說明」請提供以下內容：

- (1) 輸入給生成式語言模型的提示詞 (Prompts; Input instructions; Prefix)
- (2) 完整的生成式語言模型於本競賽 Private test set 之預測結果。
- (3) 於程式碼連結中附上應用生成式語言模型於本競賽任務之程式碼(請自行遮蔽 API key，如使用網頁服務則無須提供，但須說明操作方式)
- (4) 字數不拘

★註 1：請確認上述資料與 AI CUP 報名系統中填寫之內容相同。自 2023 年起，獎狀製作將依據報名系統中填寫內容為準，有特殊狀況需修正者，請主動於報告繳交期限內來信 moe.ai.ncu@gmail.com。報告繳交截止時間後將不予修改。

★註 2：上傳程式碼檔案與報告至程式碼託管平台或是雲端等網站，將相關連結寄信至：jamesouo@g.nccu.edu.tw，並同時副本至：t_brain@trendmicro.com 與 moe.ai.ncu@gmail.com，缺一不可。報告檔名與信件主旨請寫「競賽報告與程式碼/TEAM_????/OO 競賽」，????為參賽者所屬團隊名稱，OO 為建構無人機於自然環境偵察時所需之導航資訊競賽 I — 影像資料生成。

AI CUP 2024 春季賽

以生成式 AI

建構無人機於自然環境偵察時所需之導航資訊

競賽 I — 影像資料生成競賽

競賽報告

隊伍：TEAM_5333

隊員：余振揚(隊長)、蔣沅均、江宇浩、林欣誠

Private leaderboard：128.0601 / Rank 13

GitHub: <https://github.com/LittleFish-Coder/gen-ai-uav>

壹、環境

- 作業系統：Ubuntu 18.04.6LTS
- GPU：NVIDIA Tesla V100 32GB
- CUDA Version：11.6
- Python：conda python 3.10
- Packages (如 GitHub requirements.txt):

```
opencv-python-headless (for resizing images)
albumentations (for data augmentation)
torch
torchvision
dominate
visdom (for webserver, we do not use this)
wandb (weight and bias, we do not use this)
transformers (for super resolution)
```

```
opencv-python-headless
albumentations
torch==1.4.0
torchvision==0.5.0
dominate==2.4.0
visdom==0.1.8.8
wandb

# For Super Resolution
transformers
```

- 額外資料集：無
- 預訓練模型：無

我們使用 miniconda 作為我們的虛擬環境管理工具，由於我們的 GPU CUDA version 較舊，所以在安裝 pytorch 時，需要安裝舊版本的 torch，我們參考了 PyTorch 官方的 [document](#)，最後使用以下指令來安裝我們的環境：

```
conda install pytorch==1.13.1 torchvision==0.14.1 torchaudio==0.13.1
pytorch-cuda=11.6 -c pytorch -c nvidia
```

貳、演算法與模型架構

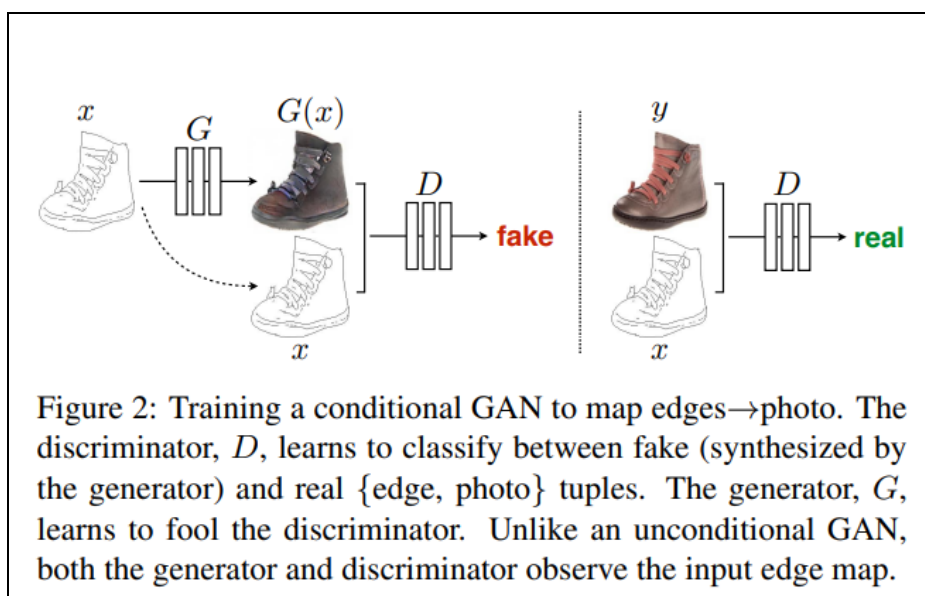
- 演算法設計：

Pix2Pix 是一種條件生成對抗網路(Conditional Generative Adversarial Network, cGAN)，主要用於影像翻譯(image translation)任務，簡單來說就是把一張圖片轉換成另一張圖片。

其演算法設計的核心思路是使用對抗訓練，讓生成網路(Generator)產生越來越真實的合成影像，而判別網路(Discriminator)則被訓練來區分真實影像與合成影像。

在訓練過程中，生成網路(Generator)接收條件影像(如邊緣圖)和噪聲向量(noise)作為輸入，並將其映射到目標影像的空間。判別網路(Discriminator)則接收生成網路產生的合成影像和條件影像，以及配對的真實影像和條件影像，並輸出每個輸入是真實或合成影像的概率。

生成網路和判別網路相互對抗，生成網路(Generator)試圖產生足以欺騙判別網路(Discriminator)的合成影像，而判別網路則試圖更好地區分真實和合成影像。透過這種對抗訓練，生成網路可以學習到從條件影像生成逼真目標影像的映射。



● **模型架構：**

Pix2Pix 模型包含以下主要部分：

1 生成網路(Generator)：

- 1.1 編碼器(Encoder)：U-Net 架構，將條件影像和噪聲向量編碼為特徵向量(Embedding)。
- 1.2 解碼器(Decoder)：使用反卷積層，將編碼器的特徵向量解碼為目標影像。
- 1.3 跳躍連接(Skip Connections)：將編碼器的特徵圖與解碼器中相應尺寸的特徵圖連接，幫助保留細節資訊。

2 判別網路(Discriminator)：

- 2.1 輸入分為真實影像對(真實影像和條件影像)和假影像對(生成影像和條件影像)。
- 2.2 使用卷積層提取特徵，最後輸出一個概率分數，表示輸入是真實還是假影像。

3 損失函數(Loss)：



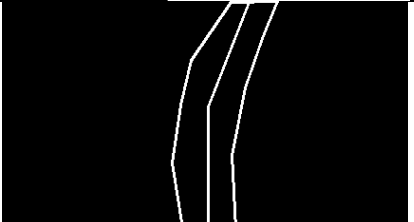

- 3.1 生成網路損失：使用對抗損失(adversarial loss)和 L1 損失(L1 loss)的加權和。
- 3.2 判別網路損失：二元交叉熵損失(Binary Cross Entropy loss)。

4 訓練方式：

交替訓練生成網路和判別網路，使用隨機梯度下降法最小化各自的損失函數。

參、技術模型原創性或改良成效

我們的主要任務即是要將一個黑白草圖轉換成空拍機影像圖，如下：

Domain Type	Draft Image	Drone Image
Road		
River		

而我們的任務還可以 breakdown 成以下子任務:

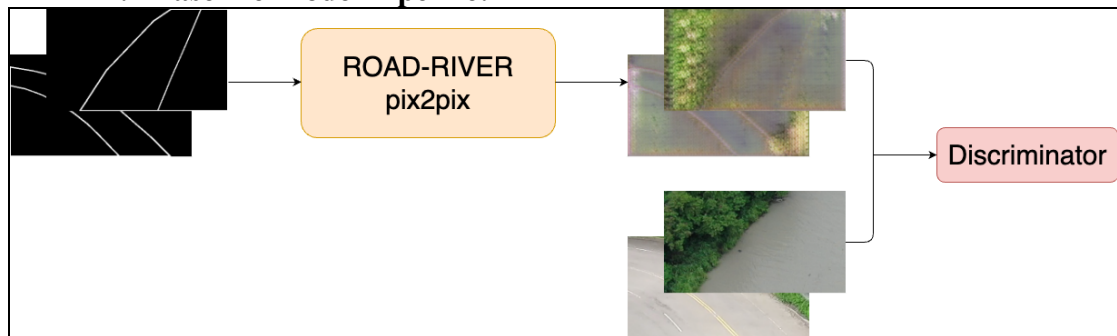
1. 將一個道路草圖轉換成道路空拍機影像圖
2. 將一個河川草圖轉換成河川空拍機影像圖

起初我們的第一個想法即是訓練一個 Baseline 模型，並將所有的 Draft Image 當作 domainA 的資料(並未 Road River 分開處理)。

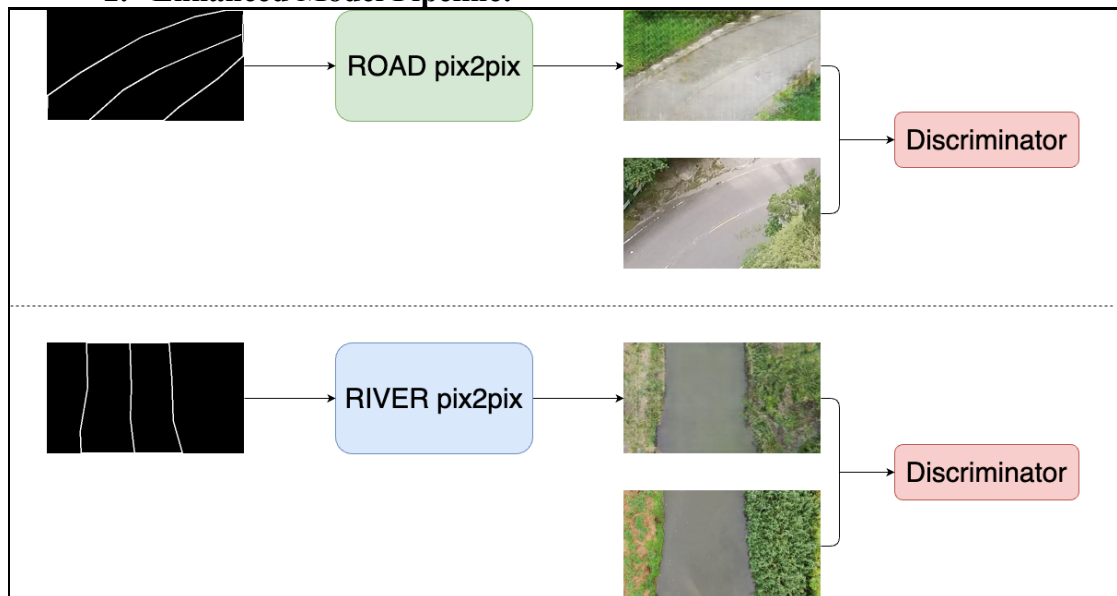
在一開始的 Public Score 上，我們發現 Baseline Model 分數不盡理想，後來提出 Enhanced Method，即分別訓練 ROAD_pix2pix 以及 RIVER_pix2pix，來分別處理不同 domain type 的輸入草圖(Draft Image)，這是我們對該任務提出的改良方法，使用該方法後 FID Score 有著明顯的下降。

以下是我們的改良模型的前後對照圖(並沒有額外去調模型內的 backbone model，僅針對訓練的方式做改良)：

1. Baseline Model Pipeline:



2. Enhanced Model Pipeline:



肆、資料分析與處理過程

1 Training Dataset Format:

訓練資料包含了 2 個子資料夾：label_img 以及 img

- label_img：黑白草圖 (black-and-white draft image)
- img：對應空拍機影像圖 (corresponding ground-truth drone image)

我們將 label_img 視為 domainA 資料，img 視為 domainB 資料，則整個 raw training dataset 可以用以下結構圖來表示：

```
training_dataset
├── label_img (trainA)
│   ├── TRA_RI_1000000.png
│   ├── TRA_RI_1000001.png
│   └── ...
└── img (trainB)
    ├── TRA_RO_1000000.jpg
    ├── TRA_RO_1000001.jpg
    └── ...
```

2 Training Dataset 資料前處理(Data Preprocess):

NOTE:

我們最佳的成果，並沒有 apply 以下所有的方法，但我們都有提供以下 data preprocess method 在我們的程式碼裡面，使用者可以依照自己的喜好進行以下方法

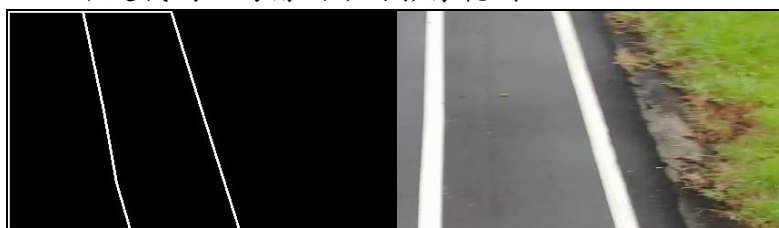
若要直接復現我們的最佳成果，我們也有提供一份 [notebook](#) 供參考

2.1 Data Filter

我們將模糊的影像刪除，以避免模型學到錯誤的 pattern。

然而我們也不曉得此 preprocess 方法，是否可以 match private image 的 ground truth，或許 private image 就是想要模糊的影像也說不定。但我們相信以一個使用者角度來看，正常人應該不會想要一張不清楚的影像，所以我們認為事先刪除模糊影像是必要的。

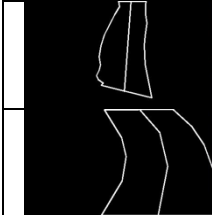


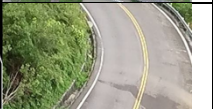
以下是我們認為需刪除的影像範例：



刪除模糊影像的條件，是由同學人工主觀判定的，我們並沒有特別訂製什麼指標來決定一張影像的好壞，我們在程式裡有提供[欲刪除影像的檔案名單](#)。

2.2 Data Augmentation

原始的 data，Road, River 大約各 2000 張，我們利用 horizontal flip 以及 vertical flip 來增加我們的資料集。

RAW	Method	Augmented
	Horizontal Flip	
	Vertical Flip	

2.3 Dataset Split

如上一節提到的，我們將資料集猜分成 Road 資料集以及 River 資料集，意即準備 2 domain-specific dataset。

以下是資料集的資料夾結構：

- Dataset Split (Enhanced Model's Architecture):

split the dataset into RIVER and ROAD domains.

```
dataset
├── train_ROAD
│   ├── trainA (Draft Images)
│   └── trainB (Drone Images)
└── train_RIVER
    ├── trainA (Draft Images)
    └── trainB (Drone Images)
```

3 Testing Dataset Format:

Testing Data 僅提供 [label_img](#)(黑白影像草圖)，我們的任務就是要將這些 domainA 的照片轉譯成 domainB 的照片，下表左圖是 raw testing dataset 的資料夾結構。

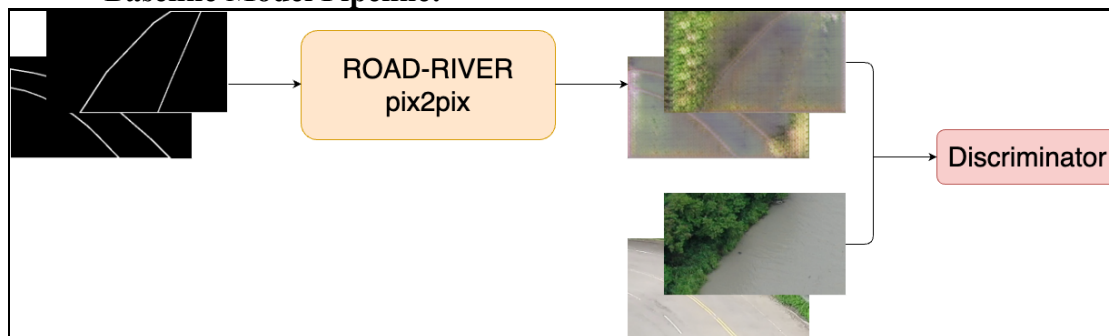
而因為我們之後有訓練了 2 個 domain-specific 的模型，一個用來看 Road 的照片，一個負責看 River 照片，所以我們要再將 testing dataset 做分類(如下表右圖)：

Raw Testing Dataset Folder Structure	2 domain-specific Testing Dataset Folder Structure
<p>Testing Dataset Folder Structure:</p> <pre>testing_dataset ├── label_img (testA) │ ├── PRI_RI_1000000.png │ ├── PRI_RI_1000001.png │ └── ...</pre>	<p>After Dataset Split:</p> <pre>dataset ├── test_ROAD │ ├── testA (Draft Images) │ │ ├── PRI_RO_1000000.png │ │ ├── PRI_RO_1000001.png │ │ └── ... │ └── test_RIVER │ ├── testA (Draft Images) │ │ ├── PRI_RI_1000000.png │ │ ├── PRI_RI_1000001.png │ │ └── ...</pre>

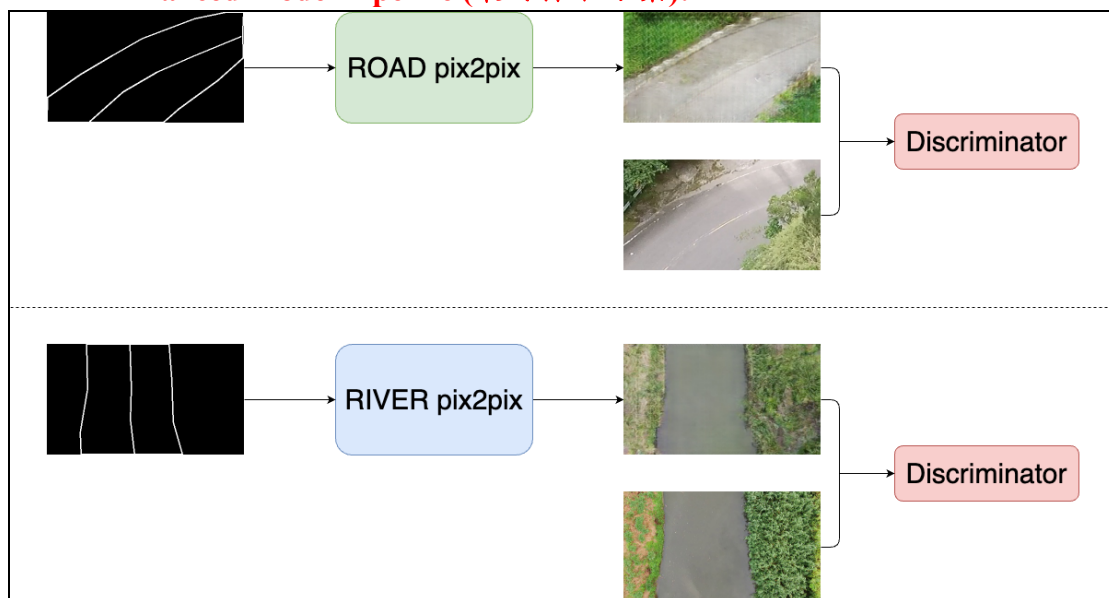
伍、訓練方式

如上述所提到，我們分別提出 Baseline 以及 Enhanced Model(Final 方案)

- **Baseline Model Pipeline:**



- **Enhanced Model Pipeline (最終採取方案):**



Private Leaderboard 最佳成果的所有 training 流程

- **Data Preprocess**
僅 Training Dataset Split，無 Data Filter 及 Data Augmentation。
- **HyperParameters Tuning**
我們僅微調以下的 args

```
n_epochs = 200  
n_epochs_decay = 200  
batch_size = 1 (default 1)  
netG = unet_256 (default unet_256)
```

*詳細流程可以直接跑 GitHub Repo 提供的 [workshop.ipynb](https://github.com/0x00b000/workshop.ipynb)

陸、結果分析與結論

1. netG backbone:

作者在 Pix2Pix 模型中，預設使用 unet256 為 backbone，另外也提供了 Resnet9block 的選項，優勢在於 input size 如果是 428x240，則 output 也會是 428x240。

我以預設參數(`n_epochs=100`, `n_epochs_decay=100`)，並將 netG 設為 Resnet9block 後，並在兩個資料集(Road & River)上分別做 training，其結果在 public 排行榜上僅得 172.1164。

而使用參數(`n_epochs=100`, `n_epochs_decay=100`, `netG=UNET256`)，並直接 train 一個大雜燴模型(Road River 一起 train)，其結果在 public 排行榜上有 142.1900。

另外也額外 train 了 2 domain-specific 的 model (`n_epochs=200`, `n_epochs_decay=200`, `netG=UNET256`)，其結果在 public 排行榜有 124.7482 分

比起 resnet backbone，可發現 unet256 backbone，有較好的表現，猜測與 unet 的 skip connection 有很大的關係。

2. Epoch:

我們在上一個環節，train 了 2 domain-specific 的 model (`n_epochs=200`, `n_epochs_decay=200`, `netG=UNET256`)，取得不錯的成績。並且以這個標準進行更多次的 training epoch tuning。

我們發現 training epoch 到後面，不管 train 多少圈，其結果並沒有大幅的提升。

3. Data Filter and Training Size:

因為比賽時間即將接近尾聲，我們在最後的資料準備上，直接 apply 了 Data Filter(Remove 模糊影像)，並且嘗試使用 Augmentation 來增加我們的樣本多樣性，使我們的 dataset 從原本的 4000 筆增加到 8000 筆，而因為樣本數變多之後，所以我們在後續的 training args 有更動 Model Training 的 Batchsize 來加速 training，將在下一個環節提到。

4. Batch Size:

當我們發現 training option 有提供 Batchsize 調整時，已經接近 competition 的尾聲，而也發現越大的 batchsize 可以加速 training process，於是我們臨時整了 hyperparameters(`n_epochs=900`, `n_epochs_decay=100`, `batchsize=64`)。

然而，我們將 train 了 1000 圈的 weight 進行 inference 時，成果反而比 (`n_epochs=200`, `n_epochs_decay=200`, `batchsize=1`) 效果還差，這顯現了 GAN 在 training 時 batchsize 是一個重要的考量，而我們疏忽了。

這導致我們無法衡量增加 training_size(增加資料集的豐富度)是否能改進我們的成果，因為我們兩者個 hyperparameters 是不相同的

應該以同樣的 hyperparameters 來去比較不同資料集對成果的影響
例如 fix hyperparameters: (`n_epochs=200`, `n_epochs_decay=200`, `batchsize=1`)
並比較兩者資料集的結果(若有時間我們應該要這樣設定才對)：

Train_Road (2000 samples) + Train_River (2000 samples)
versus
Train_Road_Aug (4000 samples) + Train_River_Aug (4000 samples)

5. Super Resolution:

由於使用 OpenCV 直接將 256x256 的 img 進行線性 enlarge 會有失細緻度，我在 [other](#) 的資料夾裡，有額外寫了 Super Resolution 的 code，主要善用現有的 SwinIR 將 256x256 的影像進行 2 倍的放大(512x512)，之後再使用 OpenCV 將影像從 512x512 縮小成 428x240。

但其結果好像沒有進步非常多，猜測與本身 inference 之後的影像的品質有關，況且 256x256 enlarge 成 428x240 也不算跨太多尺度(scale)，所以我們猜測 Super Resolution 對於該任務不算有太大的影響力。

6. 改善方法：

我們其實找到了一些更好的方法來執行此 image to image translation 的任務，以下是我們找到的相關資源：

- pix2pixHD: <https://github.com/NVIDIA/pix2pixHD>
- img2img-turbo: <https://github.com/GaParmar/img2img-turbo>
- scepter: <https://github.com/modelscope/scepter>
- controlnet: <https://github.com/lllyasviel/ControlNet>

img2img-turbo 是作者團隊於 4 月左右提出的架構，我有試著去建立該模型的環境，但礙於 V100 GPU 限制，我們的 VRAM 不足以支撐我們進行實驗。

而 Pix2PixHD 是作者與 NVIDIA 共同開發的專案，其 HD 代表 High Dimension，意味著可以處理更高解析度的照片，但也意味著 training 要花更多的時間。由於前期花太多時間在搞懂 Pix2Pix，導致我們喪失了絕佳的時機採用 Pix2PixHD 的架構，當我們發現該模型時，Competition 即將進入尾聲。

後續雖然我們沒有使用到任何其他的模型架構，但以現有的 Pipeline 設計使我們在 Private Leaderboard 取得 Rank 13 的成績，我們認為非常幸運。

至於 Pix2PixHD 因為我們還是很想嘗試其效果，所以 Leader 本人在其他專案有使用到該模型，其專案在於將 pose 骨架資訊轉移到使用者跳舞姿態(A pose 2 dance translation application)，有興趣可以到該 GitHub 觀賞成果

LittleFish-Coder/yyds-dance-generator:
<https://github.com/LittleFish-Coder/yyds-dance-generator>

柒、程式碼

Github 連結：<https://github.com/LittleFish-Coder/gen-ai-uav>
(GitHub README 以英文撰寫為主，亦有附上操作教學)

復現最佳成績

若要直接復現 Private Leaderboard 上最好的成績，並還原當時的 Data Preprocess 過程以及模型權重，可以直接跑該 GitHub Repo 提供的 [workshop.ipynb](#)，將會提供當時的訓練參數以及 pre-trained 好的 model

客製化 model：Data Preparation + Train Model + Test Model

可參考 GitHub 的 README，我們提供了客製化的 Training Process，詳情可以參考 [#Usage](#)：

1. 先執行 [dataset/preprocess_dataset.ipynb](#) 以客製化 trainig dataset
2. 執行 [train_model.ipynb](#) 以客製 training model 流程
3. 執行 [test_model.ipynb](#) 來測試模型結果

小組合作

另外，此次比賽有認識其他隊伍 TEAM_5574，我們有進行討論且結果不相同，code 有些許雷同但僅作為合作用途，合作 GitHub Repo 如下：

Sherry2580: <https://github.com/Sherry2580/AI-cup-2024-spring>

捌、使用的外部資源與參考文獻

本隊伍於此次比賽期間，未使用任何生成式模型進行資料擴增，但在撰寫 code 時有使用 Microsoft copilot 協助。

參考的文獻如下：

- Medium - GAN:
https://tomohiroliu22.medium.com/深度學習_paper系列-03-generative-adversarial-networks-a64e89387b62
- Medium - Conditional GAN:
https://tomohiroliu22.medium.com/深度學習_paper系列-08-conditional-gan-ea75376580a6
- Medium - Pix2PixHD
https://tomohiroliu22.medium.com/深度學習_paper系列-09-pix2pix-hd-a188aecfcfd
- CSDN:
<https://blog.csdn.net/JNingWei/article/details/78218837>
- Stack Overflow:
<https://stackoverflow.com/questions/23853632/which-kind-of-interpolation-best-for-resizing-image>