

摘要

本研究中，我們利用 Google 所提供的 API 收集來自台灣及中國的地形高度圖(heightmap)和衛星空照圖。我們利用收集的圖像訓練 pix2pix cGAN (conditional Generative Adversarial Network) 模型，並將人工繪製的高度圖加上真實山脈應有的細節(包含尖銳的山脊、山壁上的紋路、連續的河流網路……等)，透過訓練的 pix2pix 模型生成更接近真實山脈的效果。為了提升擬真地形的生成效果，我們在原先 pix2pix 模型的基礎下，額外加入「動態權重層」來達成。最後利用這些經訓練的 pix2pix 模型，開發了 API、Unity 客戶端及網頁客戶端，使我們訓練的 pix2pix 模型更加實用。本研究除了能應用於遊戲開發，使生成擬真山脈的流程更為簡化，也可將其應用於將低解析度之高度圖轉換為高解析度之高度圖，同時保持高度圖的真實性，幫助地形資料的收集作業。

壹、 研究動機

隨著 3C 的進步，遊戲已經成為現代人打發時間及社交的必需品；隨著科技技術的進步，對於遊戲畫質的要求也越高，而在製作各種遊戲時，常常會需要生成擬真的地形作為場景。

我們曾嘗試用 Unity 開發冒險遊戲，冒險遊戲的地形是使用 Unity 的 terrain 工具用筆刷繪製，發現若要透過人工繪製出擬真的山脈地形，須將大致的架構畫出來後，還需花費不少時間捏出山脊和挖出河流等細節部分。因此我們希望簡化人工繪製的過程，透過訓練模型生成擬真山脈地形的成果。

貳、 研究目的

本研究目的期望能簡化遊戲製作者在生成擬真的山脈地形的流程。製作**地形擬真模型**、**空照圖模型**兩個 cGAN 模型，**地形擬真模型**能有效地將人工設計的山脈架構高度圖自動轉換成**擬真山脈地形**高度圖；**空照圖模型**則依其生成出的擬真高度圖，生成相對應的空照圖，作為擬真高度圖 3D 渲染時的貼圖。

參、 研究設備與器材

一、 軟體環境：

Python 3.7、Pytorch 1.4、Flask、OpenCV (opencv-python)、Jupyter Notebook、p5.js、three.js、Unity

二、訓練資料來源：

Google Elevation API、Google Maps Static API

三、硬體規格：

(一) 工作站：

1. CPU：Intel(R) Core™ i7-6800K CPU@3.80 GHz
2. 記憶體：128.0 GB
3. GPU：NVIDIA GeForce GTX 1080 Ti * 2
4. 作業系統：Ubuntu 16.04

肆、研究過程與方法

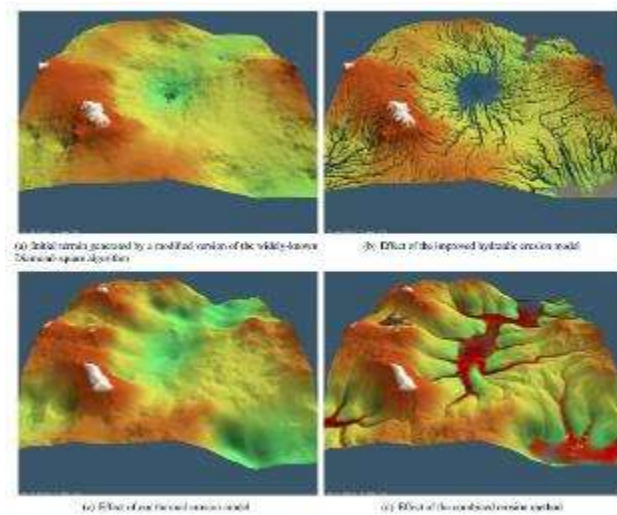
一、文獻探討：

(一) 物理侵蝕模型[1]：

此物理模型是使用 GPU 模擬水流在輸入地形上侵蝕與堆積的方式，實現改變地形樣貌的效果。

其做法是將地表分成正方形的網格，使用歐拉法求地表上每格的水深、含沙量和網格間的水流速，並根據水量和流速進行侵蝕和堆積，疊代多次後可得出侵蝕一段時間後的地面和水面高度圖。一次疊代的步驟如下：

1. 在每個網格加上等量的水，模擬均勻的降雨
2. 更新流速(加速度受坡度和阻力影響)
3. 根據流速讓水流到鄰近的格子，同時搬運等比例的砂土
4. 根據水量和流速進行侵蝕，增加水中含沙量，降低地面高度
5. 將水中一定比例的沙土堆積到地面
6. 移除每格一定比例的水，模擬蒸發



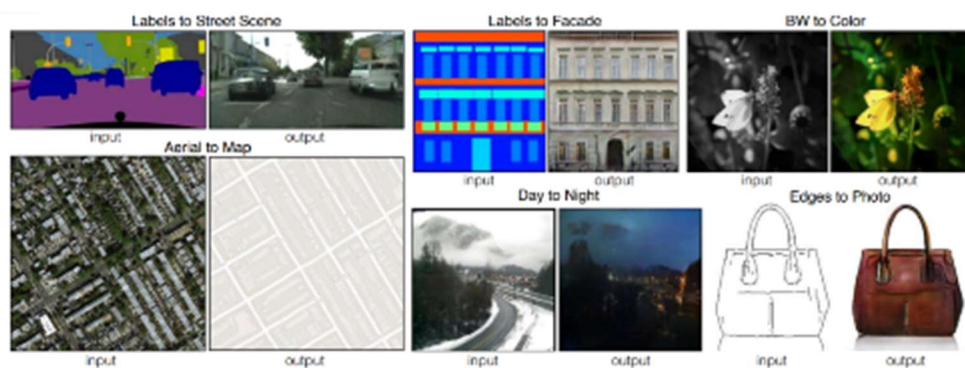
圖一、物理侵蝕模型的效果 (取自[2])

我們利用 Unity 的 compute shader 實現該模型，以將其與我們訓練的 pix2pix 模型進行真實度及實用性的比較。

另外，此模型需要調整許多複雜的參數，如單位時間、面積的雨量、侵蝕的速率、一次疊代的時間步長、水中每單位時間沉澱的沙土比例及蒸發速率等。

(二) pix2pix 模型[2]：

pix2pix 模型是一個 Conditional adversarial networks(cGAN)，訓練時將一對影像當作輸入，而模型的目標則是將第一張圖片轉換為第二張圖片。例如下圖，模型的目標是將左圖做為輸入，輸出右邊的圖像。

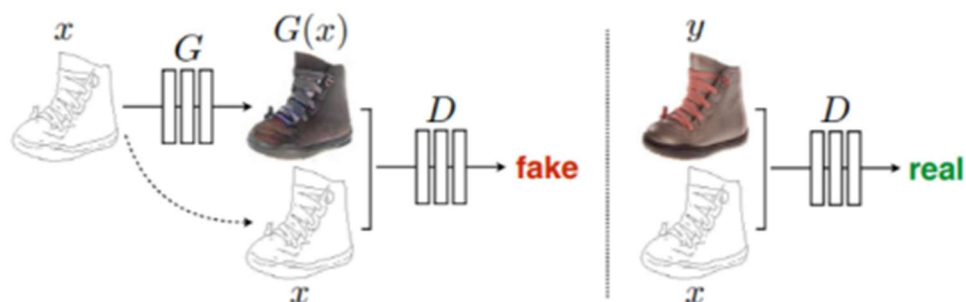


圖二、pix2pix 可做到的圖像風格轉換(取自[2])

pix2pix 由 generator 和 discriminator 兩個部分組成。generator 的結構為 U-Net，訓練時會嘗試把輸入圖像轉換為目標圖像。discriminator 則是一個分類器，訓練時會嘗試分辨哪些圖是 generator 生成的圖，哪些是真的目標圖像。兩者會同時訓練，generator 生成的圖越不容易被 discriminator 分辨出來，就代表 generator 表現得越好。利用這

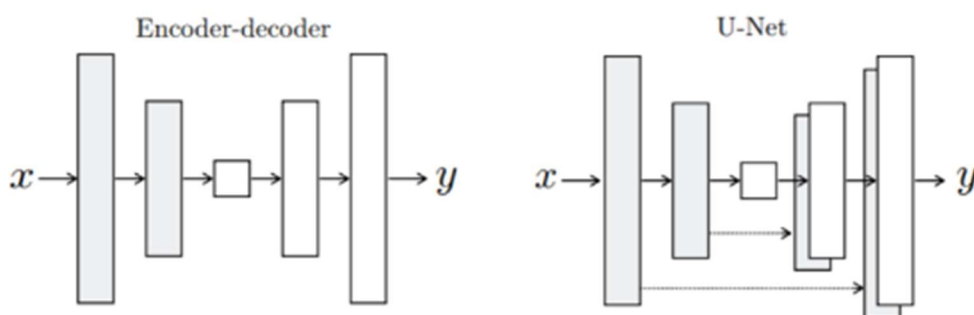
點來訓練 generator，就能讓它的輸出盡可能的真實，在此研究中，我們要利用的就是把訓練好的 generator 來將高度圖加上山脈細節，使其更為擬真。

如圖三中，discriminator 的任務是辨認出哪些圖片是由 generator 所生成(如左)，哪些是原始圖像(如右)。



圖三、訓練 pix2pix 將鞋子的邊緣圖生成實際鞋子的圖像。(取自[2])

此外，一般的 generator 都是使用 Encoder-decoder 結構，而 pix2pix 模型的 generator 則使用了特殊的 U-Net 結構，其為 Encoder-decoder 結構的改良。其在不同層之間加上了跳躍連接(skip-connections)，如圖四。而 U-Net 在圖像分割任務上表現十分良好。我們的研究將使用 pix2pix 作為基礎模型。



圖四、Encoder-decoder 與 U-Net 結構比較 (取自[2])

二、收集訓練模型所需之圖像資料：

(一) 地形擬真模型之訓練資料：

這份高度圖資料收集範圍為橫斷山脈的一矩形，四點經緯度座標分別為：

28.15°N, 98.75°E、25.65°N, 98.75°E、25.65°N, 101.25°E、28.15°N, 101.25°E，長寬皆為

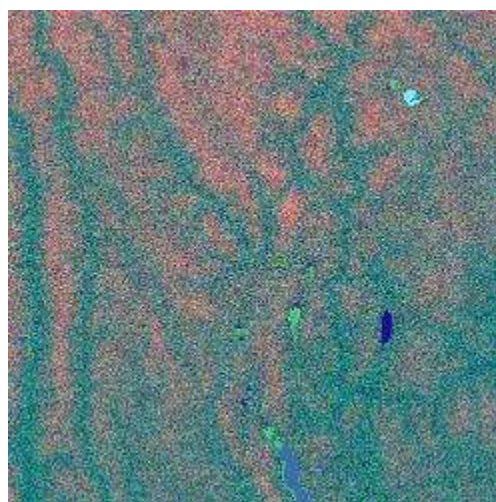
2.5 經/緯度。

我們利用 Google Elevation API 每隔 0.001 經/緯度在上述選取範圍內得到特定點的高度值，收集完這 2500 x 2500 個高度值後，我們再將其高度最大值對應的像素值設為 $2^{24} - 1$ ，高度最小值對應的像素值則設為 0，其他高度值則根據最大與最小值範圍進行線性映射，最後就會得到一張長寬皆為 2500 像素的 24 bits 的高度圖。

我們之後會利用此張高度圖，生成出 256 x 256 的灰階高度圖資料集，訓練地形擬真模型。



圖五、橫斷山脈之收集範圍



圖六、橫斷山脈之 24 bits 高度圖(長寬為 2500 像素)

(二) 空照圖與高度圖之間轉換之模型的訓練資料：

訓練資料是從台灣的中央山脈及中國的東南丘陵兩個區域蒐集的。中央山脈的收集範圍為一四邊形，其頂點經緯度座標分別為：24.663513°N, 121.182540°E、24.534208°N, 121.665832°E、23.188591°N, 120.532673°E、23.012812°N, 121.095572°E

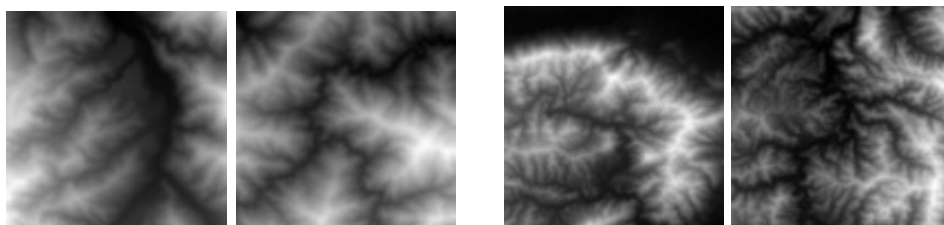


圖七、中央山脈及東南丘陵的選取範圍

°E；東南丘陵的收集範圍也為一四邊形，其頂點經緯度座標分別為：27.162822°N, 117.929151°E、27.494665°N, 113.754346°E、24.952233°N, 117.819288°E、24.772807°N, 113.820264°E。

因為空照圖模型需要將高度圖與衛星空照圖互相轉換，所以我們同時蒐集相互對應的空照圖和高度圖。

首先收集高度圖，我們將四邊形分割為長寬皆為 0.1 經/緯度的正方形，並在這些正方形內，利用 Google Elevation API，每隔 0.001 經/緯度紀錄一次此地的高度值，當正方形內的所有數據點都被記錄後，我們將數據點最大值的對應像素值定為 255，高度最小值的對應像素顏色則定為 0。其他數據點則根據最大最小值做線性調整，最後就會得到一張 100 x 100 的 8 bits 高度圖。



圖八、台灣中央山脈之灰階高度圖 圖九、中國東南丘陵之灰階高度圖

再來收集衛星空照圖，Google Maps Static API 可以回傳特定經緯度的衛星空照圖，但因為回傳的衛星空照圖的周圍會有 Google 的商標與地圖的資訊，會使得無法與灰階高度圖相互對應，故我們分別調整回傳的圖像大小及地圖縮放倍率來將其中位置的衛星空照圖與灰階高度圖相互對應。

經過測試，我們將圖像大小設為 640 x 640 像素，縮放倍率則設為 11，並在此空照圖的中央位置裁切出一個 320 x 320 的圖像，再將此圖像縮小成 100 x 100 (0.1 經/緯度)，與灰階高度圖相互對應，如圖十。



圖十、API 回傳之空照圖、裁切後的空照圖及該位置的灰階高度圖。

至此，本研究所需之高度圖及空照圖項目已完成收集，收集圖像數量如下

表：

表一、高度圖及空照圖的圖像數量(單位：張)

地區 \ 類型	灰階高度圖	衛星空照圖 (100 x 100)
中央山脈	100 (100 x 100)	100
東南丘陵	677 (100 x 100)	677
橫斷山脈	1 (2500 x 2500)	0

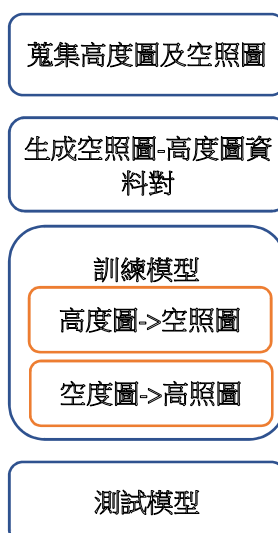
三、pix2pix 模型的訓練方式：

pix2pix 模型之訓練方式是輸入一對照片，其中一張為輸入，另一張則為目標輸出。pix2pix 模型的目標即是將模型輸出盡量接近目標輸出。

我們將訓練模型的圖像集分為三個部分，分別為 train、val 及 test，train 是用於訓練模型，val 則是在訓練過程中驗證模型的正確性，test 則是在訓練完成後，用來測試模型。

四、訓練空照圖與高度圖之間轉換之模型：

下圖為空照圖與高度圖之間轉換之模型的訓練流程，蒐集高度圖及空照圖以及生成空照圖-高度圖資料對



圖十一、空照圖與高度圖之間轉換之模型的訓練流程

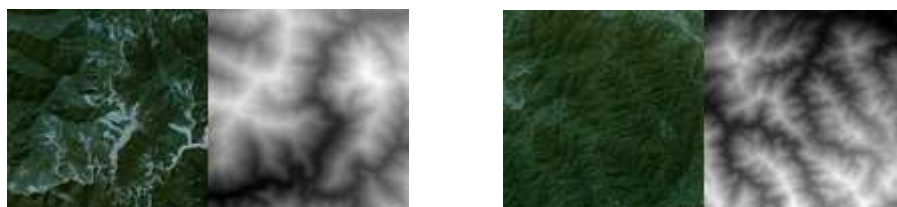
(一) 將空照圖轉換為高度圖之模型：

此模型的目標是將輸入的空照圖生成相對應的高度圖，故我們的訓練資料對即是空照圖及高度圖，其中空照圖為輸入，高度圖為目標輸出。

我們將中央山脈及東南丘陵兩個地區分為兩個訓練集，分開訓練，故總共會訓練出兩個模型。兩訓練集之 train、val、test 的圖像數量如下表：

表二、空照圖與高度圖互相轉換模型之 train、val、test 的圖像數量(單位：張)

地區 \ 類型	train	val	test	總和
中央山脈	50	25	25	100
東南丘陵	338	169	170	677



圖十二、中央山脈(左)及東南丘陵(右)模型(一)之訓練資料

分類好資料集後，我們利用 pix2pix 模型所提供的 train.py 程式碼進行訓練，訓練指令如下：

```
python train.py --model pix2pix --name datasets/ChinaAerialTrain --direction AtoB
```

(二) 將高度圖轉換為空照圖之模型：

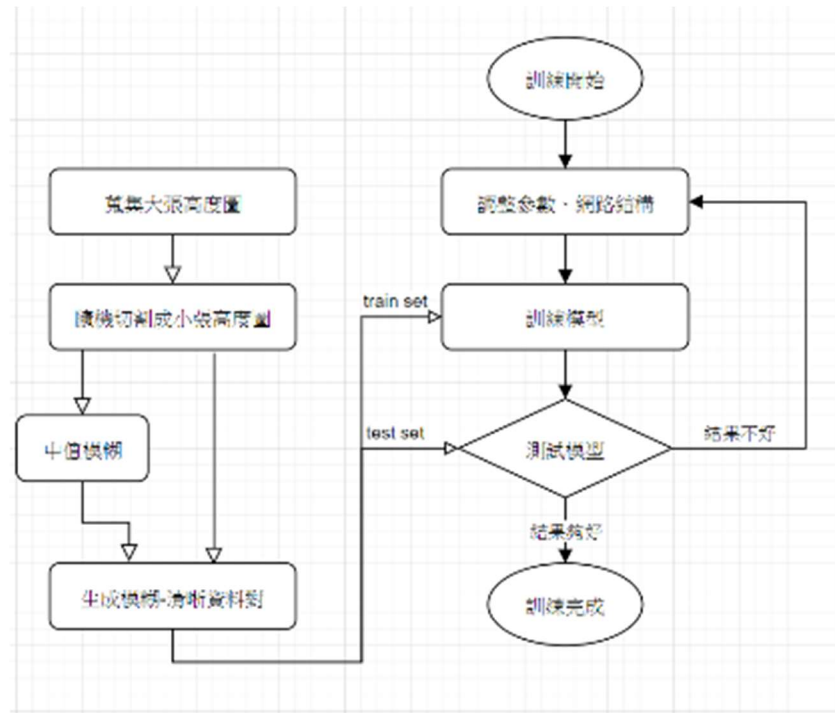
在 pix2pix 模型中，有一個參數(direction，預設為 AtoB)可以設定輸入及目標輸出顛倒過來，也就是將原本的輸入變為目標輸出、目標輸出變為輸入。故我們只要將模型(二)的 direction 參數調整為 BtoA，即可訓練出此模型。為了比較，在此模型中，我們只訓練中國東南丘陵的圖像。訓練指令如下：

```
python train.py --model pix2pix --name datasets/ChinaAerialTrain --direction BtoA
```

五、訓練地形擬真模型：

我們希望地形擬真模型能將人工繪製的大致山脈架構高度圖(以下稱為手繪圖)轉換成擬真山脈地形高度圖。也就是這個模型是以手繪圖做為輸入，再加上真實山脈

應有的細節和特徵後進行輸出，其流程如圖十三。



圖十三、地形擬真模型的訓練過程

(一) 準備訓練資料：

因為我們使用 pix2ix 模型進行訓練，所以訓練的資料對共有兩張圖像，分別做為輸入及目標輸出。目標輸出圖片的生成方式是從先前蒐集的橫斷山脈高度圖上隨機取 256 x 256 的正方形。每次選取的正方形的位置和角度都是隨機的，以增進訓練資料的一般性。

至於輸入圖像，如果要人工模仿每張目標輸出畫出手繪圖，會花費很多時間。所以我們使用中值模糊(kernel size 為 29)來快速生成輸入圖片。因為中值模糊的特性，經過中值模糊後，真實高度圖中的大山脊上的小河谷和大河谷上的小凸起物會被抹除，剩下的線條類似手繪圖的大致架構，符合模型輸入是山脈的大致架構這項條件。

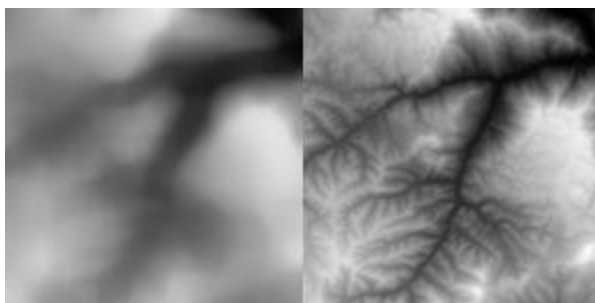
實務上，我們會使用 OpenCV 內建之 medianBlur 函數將原始圖片進行中值模糊。

```
1 import cv2
2 im = cv2.imread("/home/host/scifair/test.png")
3 print(im)
4 cv2.imwrite("blur.png", cv2.medianBlur(im, 29))
```

圖十四、中值模糊的程式碼

其程式碼如圖十四。

最後，我們將中值模糊過的真實高度圖和原始的真實高度圖組成模糊-清晰資料對，用來訓練地形擬真模型。



圖十五、橫斷山脈的訓練資料對

訓練集之 train、val、test 的圖像數量如下表。

表三、地形擬真模型之 train、val、test 的圖像數量(單位：張)

地區 \ 類型	train	val	test	總和
橫斷山脈	1024	256	25	1305

(二) 對 pix2pix 模型進行調整：

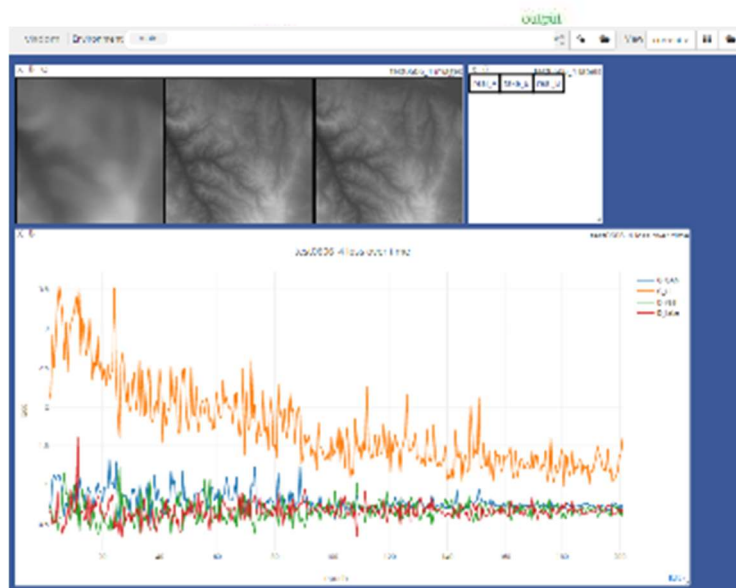
pix2pix 模型原本是針對「一般圖片的風格轉換」這項工作設計的，這裡指的是像素值有固定上下界且通常有 RGB 三數值的圖片。這和我們的目標—「生成擬真山脈」性質上有些許不同，所以我們調整 pix2pix 模型使其符合需求：

1. 高度圖只有一個數值，所以設定模型的輸出及輸入數值數量皆為 1。
2. 高度圖的像素值沒有上下界的限制。為了讓輸出不因線性映射造成結果過大的失真，讓模型能均勻調整產生的細節，我們把 generator 輸出層的激活函數 tanh 去除。
3. 將山脈加上細節和一般的風格轉換有很大的差別：前者的輸出會沿用輸入的像素「值」作為架構(不只沿用空間架構)，後者輸入和輸出的像素值則不必然有直接關係。所以我們在 generator 的最外層(也就是輸入和輸出層)加上額外的 skip connection 讓此層的輸出直接與模型輸入相加，成為最終的模型輸出(原本的 generator 第二層以下才有 skip connection)。因此，模型只需要學習輸

入和目標輸出的差，也就是哪裡要增高、哪裡要降低，而不用學習如何重建整個高度圖。

(三) 初步訓練及測試：

對模型架構進行了以上的調整後，其他參數我們全部採用 pix2pix 的預設參數訓練模型，以先前準備好的橫斷山脈模糊-清晰資料集訓練。訓練進行時可以使用 Visdom 觀察模型的 loss(橘、綠、紅、藍線)變化情形，越小越好。



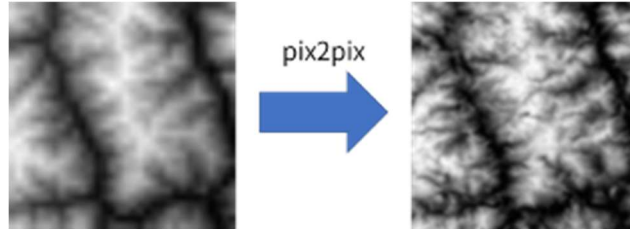
圖十六、Visdom 的介面

此次訓練的表現相當理想，經過測試，test 資料集的誤差(L1 loss)為 0.023。

但是在以手繪圖測試此訓練結果時，我們發現當輸入圖片像素值在空間上的頻率過高(細節過多)時，模型輸出會變得很不真實。我們推測這是因為訓練資料的輸入圖都是經過 29 x 29 中值模糊(對 256 x 256 的圖片來說是嚴重的模糊)，中值模糊的特性會將所有高頻細節消除，透過模型學習，要在輸入圖加上很多高頻計算出的細節作為模型輸出。若測試時的手繪圖全部或部分區域已經含有高頻細節，則模型訓練後會再加一堆細節上去，使輸出高度圖變的雜亂、不真實。如果希望得到較佳的結

果，就要限制輸入手繪圖只能含有概略的資訊。

這是需要改進的地方，因為模型接收到模糊的高度圖後，會依訓練結果計算出細節的樣子(例如小河谷的位置)，而模型的計算對使用者來說是無法預期的。如果使用者想要在輸入圖的某些區域指定精確的細節(例如小河谷的密集程度或形狀)，則必須讓模型不要在該處進行運算。也就是模型必須學習在越模糊的地方加越多細節，

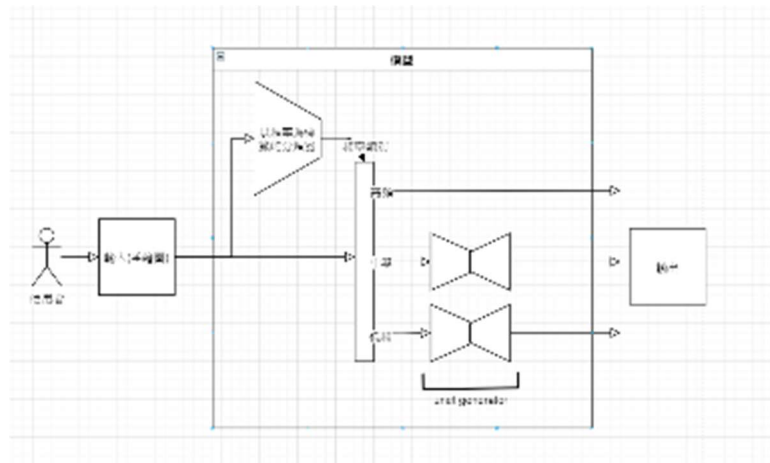


圖十七、真實高度圖做為輸入會使輸出雜亂

已經有夠多細節的地方就維持原樣輸出。

(四) 建構能適應不同細節量的輸入的模型：

為了建構能適應不同細節量輸入的模型，開始的構想是，先訓練 n 個模糊轉清晰 pix2pix 模型，每個模型的訓練資料中輸入圖的中值模糊的模糊程度不同，以得到多個適應不同模糊程度的 generator。再額外訓練訓練一個以輸入圖的像素值在空間上的頻率(也就是分類模糊程度)為標籤的小型分類器(下以 c 表示)。



圖十八、建構適應不同細節量輸入模型的初步構想

在應用時，先把手繪圖輸入頻率分類器，再依照分類器給出的類別決定要用使哪一個 generator，最後以該 generator 的輸出為整個模型的輸出。依照此概念可以寫出：

$$\hat{y} = [c(x) \cdot G](x) \quad (\text{式一})$$

其中 x 為模型輸入， $c(x)$ 為頻率分類器的輸出(為 1-hot-encoding)， G 為 n 個 generator，

$$G = [g_1, g_1, \dots, g_n]^T \quad (\text{式二})$$

但是這個模型架構過於複雜而且不連續，訓練和使用起來效率較低，所以我們以這個假想模型的概念為基礎建構出效率更高的模型。

Generator(下以 g 表示)是由 encoder (下以 e 表示)和 decoder (下以 d 表示)部分組成的，精確來說 $g(x) = d \circ e(x)$ 。所以上式可以寫成：

$$\hat{y} = \sum_{i=1}^n c(x)_i d_{g_i}[e_{g_i}(x)] \quad (\text{式三})$$

注意到 c 和 e 皆為 feature extractor 的 CNN，輸入也都是 x 。我們受到 <https://distill.pub/2020/circuits/>[5]中 high-low freq filter 的啟發，文中提到 CNN 可以辨別高頻和低頻(相當於模糊程度)的區域，所以我們把 c 的工作交給 e ，讓 e 除了原本的 feature 外，額外輸出關於模糊程度的 feature，並在 d 中引入「動態權重模組」以消化這些 feature，

$$\hat{y} = \sum_{i=1}^n d'_{g_i}[e'_{g_i}(x)] \quad (\text{式四})$$

e' 的輸出訊息包含了原本 c 和 e 的輸出訊息。因為 d 和 e' 之間有 skip connection，前者可以選擇從內層進入到 d ，並藉由我們在 d 中引入的「動態權重模組」控制 d 中每個 feature 的權重；相較之下，原先的模型只能跟 d 的輸出相乘，決定整個 d 的輸出要不要被採用。改良後的方法能增加模型學習的自由度。

「動態權重層」是我們設計的神經網路模組。它就像 1x1 convolution，但是把權重換成冪(exponent)，而通道間的相加換成相乘，其在 PyTorch 的實現如下：

```
class Mult(nn.Module):
    def __init__(self, nc):
        super(Mult, self).__init__()
        self.register_parameter(name='exp',
                                param=torch.nn.Parameter(torch.diag(torch.ones(nc)).unsqueeze(-1).unsqueeze(-1)))
        self.register_parameter(name='bias',
                                param=torch.nn.Parameter(torch.zeros(nc).unsqueeze(-1).unsqueeze(-1)))
        self.relu = nn.ReLU()

    def forward(self, x):
        x = self.relu(x) + 0.1
        return x.unsqueeze(-3).pow(self.exp).prod(1) + self.bias
```

圖十九、動態權重層在 PyTorch 的實現

此模組中 exp 參數為單位矩陣時與原先的模型不會有任何差別，但模型會透過學習此參數的情形，來決定哪個通道要和哪個通道以多少冪次相乘，就像是動態的

權重一樣。故能夠確保加入這個模組後，模型的表現至少會和原本的一樣(當 exp 為單位矩陣時)，或者更好(當模型有成功學習到 exp 參數時)。

而式四 $\hat{y} = [c(x) \cdot G](x)$ (式一中的每個 generator 雖然學習處理不同模糊程度的輸入，但因目標相近，所以或多或少會有 feature 是相通的，尤其是靠近外層的低階 feature。所以我們進一步把所有 generator 合併，以節省總通道數，讓模型簡化為只有一個 unet 作為 generator。

$$\hat{y} = d'[e'(x)] = g'(x) \quad (\text{式五})$$

這邊的 g' 就是我們最終得到的新 generator。這裡剩下一個 U-Net，但因為有動態權重層，能在增加少許參數的情況下適應不同模糊程度的輸入。而幫助 generator 訓練的 discriminator 也同樣要加上相同的動態權重層，同時提升訓練效果。

因為訓練模式的改變，原本供不同 generator 學習的資料集也需要合併，用來訓練 generator。在資料集中，我們把中值模糊的 kernel size 從原本固定 29 改成 1 到 29 之間隨機的數字，使每張輸入圖像有不同模糊程度，讓每組資料對的輸入圖的模糊程度都不一樣。

另外，我們也通過調整 U-Net 的層數、generator 的 filter 數量(參數 ngf)及有無「動態權重層」來訓練並比較各種不同的模型，期望找到一個效果最好的模型。

我們共訓練了七個模型，其參數分別如下表，其訓練結果我們會在研究結果中說明。

表四、各擬真山脈模型的參數

模型編號	U-Net 的層數	ngf	有無動態權重	中值模糊的模糊程度
1 (即為初步訓練之模型)	8	64	無	29
2	8	48		29
3	8	48		1 ~ 29
4	8	64		1 ~ 29

5	7	64	有	1 ~ 29
6	8	48		1 ~ 29
7	8	64		1 ~ 29

伍、 研究結果

一、 空照圖與高度圖互相轉換之模型訓練結果：

以下圖像皆為來自 test 資料集之圖片，也就是說模型並沒有在訓練過程中”看過”這些圖像。藉由觀察這些圖像，我們能更好的評斷模型的學習程度。

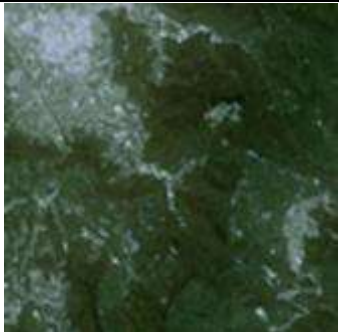
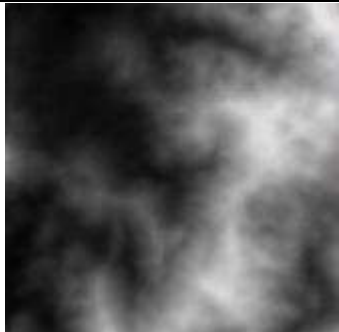
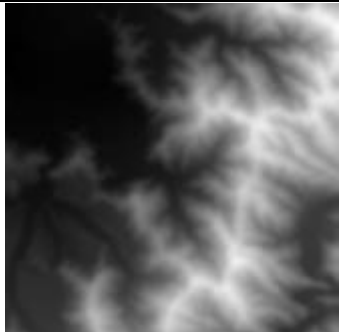
以下圖像皆以三個為一組，分別代表輸入圖片、模型輸出圖片及目標輸出圖片。

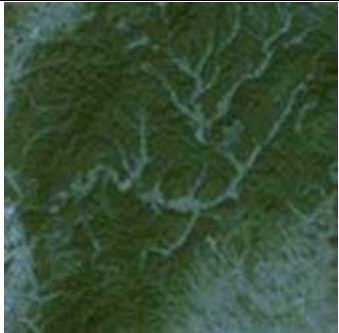
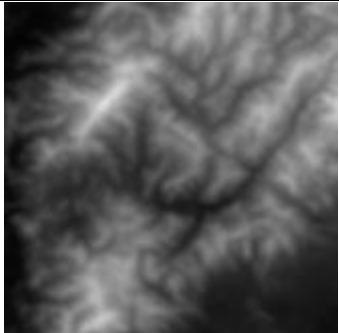
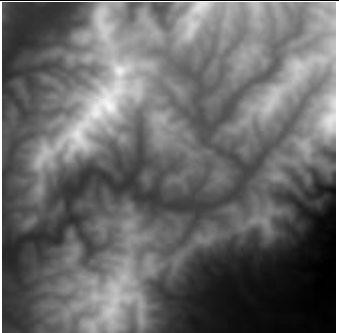
(一) 依照空照圖生成相對應之灰階高度圖(長寬為 100 像素)之模型：

在表五中，能發現兩個模型都可以根據 100 x 100 的衛星空照圖生成其相對應的灰階高度圖，且與目標輸出圖片相差不遠，若將模型輸出圖像與目標輸出圖像共同排放在一起，一般人應無法分辨哪個才是真實的灰階高度圖。

另外，我們可以發現中國東南丘陵的輸入圖像的右下角有一部分成白色，其為人口較多的區域，通常地勢也較平坦。對應到模型輸出圖像中右下角呈現黑色，即是對應到較平坦的區域。

表五、模型(一)中台灣中央山脈與中國東南丘陵之輸入、輸出及目標輸出

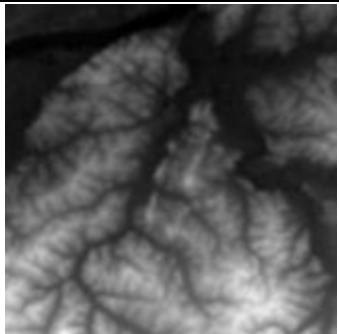
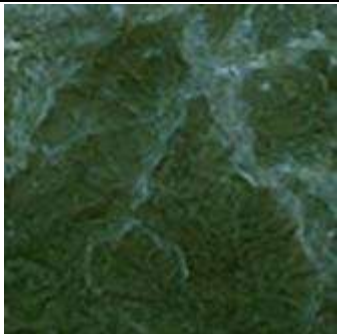
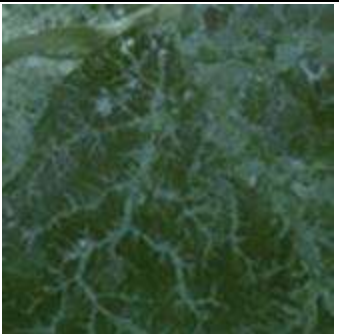
地區	輸入圖像	模型輸出圖像	目標輸出圖像
台灣 中央 山脈			

中國 東南 丘陵			
----------------	---	--	---

(二) 依照灰階高度圖生成相對應之空照圖(長寬為 100 像素)之模型：

我們發現在空照圖轉灰階高度圖的訓練中，pix2pix 模型所生成空照圖雖與真實空照圖有些差別，但我們認為生成效果仍相當不錯，例如較平坦的地方通常是河流或居住地，在空照圖的表現常是偏白或偏藍，這點在模型輸出非常明顯。

表六、模型(二)中中國東南丘陵之輸入、輸出及目標輸出

地區	輸入圖像	模型輸出圖像	目標輸出圖像
中國 東南 丘陵			

二、地形擬真模型的訓練結果：

與上個模型相同，地形擬真模型的測試圖像都是來自 test 資料集(共有 25 張圖像)。測試完成後，我們會將模型輸出與目標輸出計算 L1 Loss，藉此來判定每個模型的表現。


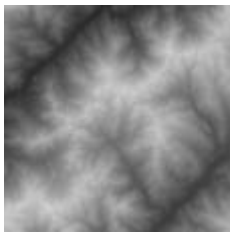
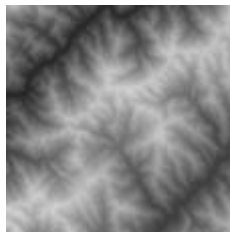

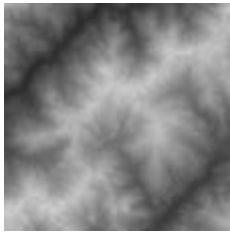
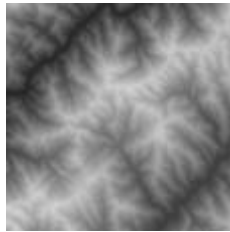

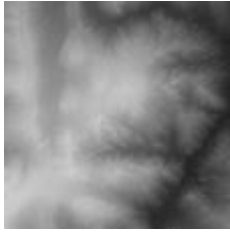
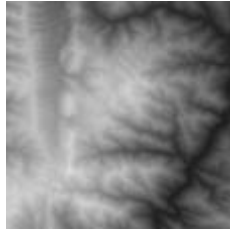

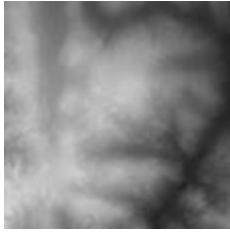
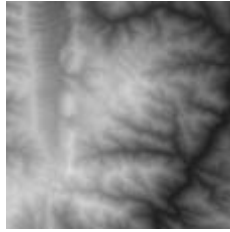
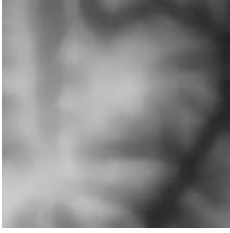
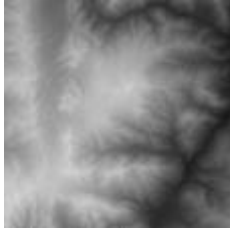
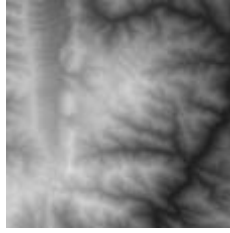
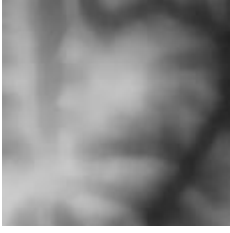
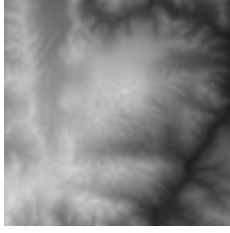
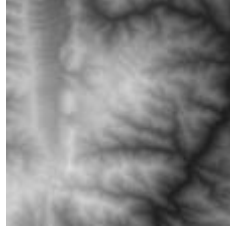


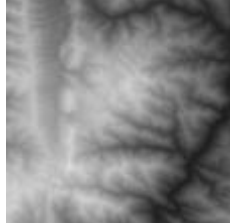
我們會將高度圖由原本的 8 bits 的形式經過線性映射轉換為 0~1 的浮點數，255 映射到 1；0 則映射到 0。映射後，再計算 L1 Loss，每張圖 L1 Loss 的定義如下：

$$\frac{1}{256 \times 256} \left(\sum_{i=1}^{256} \sum_{j=1}^{256} |x_{i,j} - y_{i,j}| \right) \quad (\text{式六})$$

其中 $x_{i,j}$ 與 $y_{i,j}$ 分別代表模型輸出及目標輸出在位置像素 i, j 的浮點數值。表七中的 L1 Loss 為 25 張圖平均之後的結果。

(一) 各模型輸入、模型輸出及目標輸出圖像的比較：

表七、各地形擬真模型的輸入、模型輸出及目標輸出圖像

模型編號	輸入	模型輸出	目標輸出
1 (即為初步訓練之模型)			
2			
3			
4			
5			
6			
7			

表八、各地形擬真山脈圖形模型輸出與目標輸出的平均 L1 Loss 及其標準差

模型編號	L1 Loss	標準差
1 (即為初步訓練之模型)	0.023	0.0018
2	0.027	0.0062
3	0.021	0.0025
4	0.021	0.0077
5	0.018	0.0078
6	0.020	0.0079
7	0.016	0.0066

從 L1 Loss 可以發現，模型 7 的效果最好，在這個模型中，我們加入了動態權重模組，且使用結構較為龐大的 U-Net(深度為 7、ngf 為 64)，使這個模型的整體效果較其他模型較為良好。

(二) 人工手繪圖的比較：

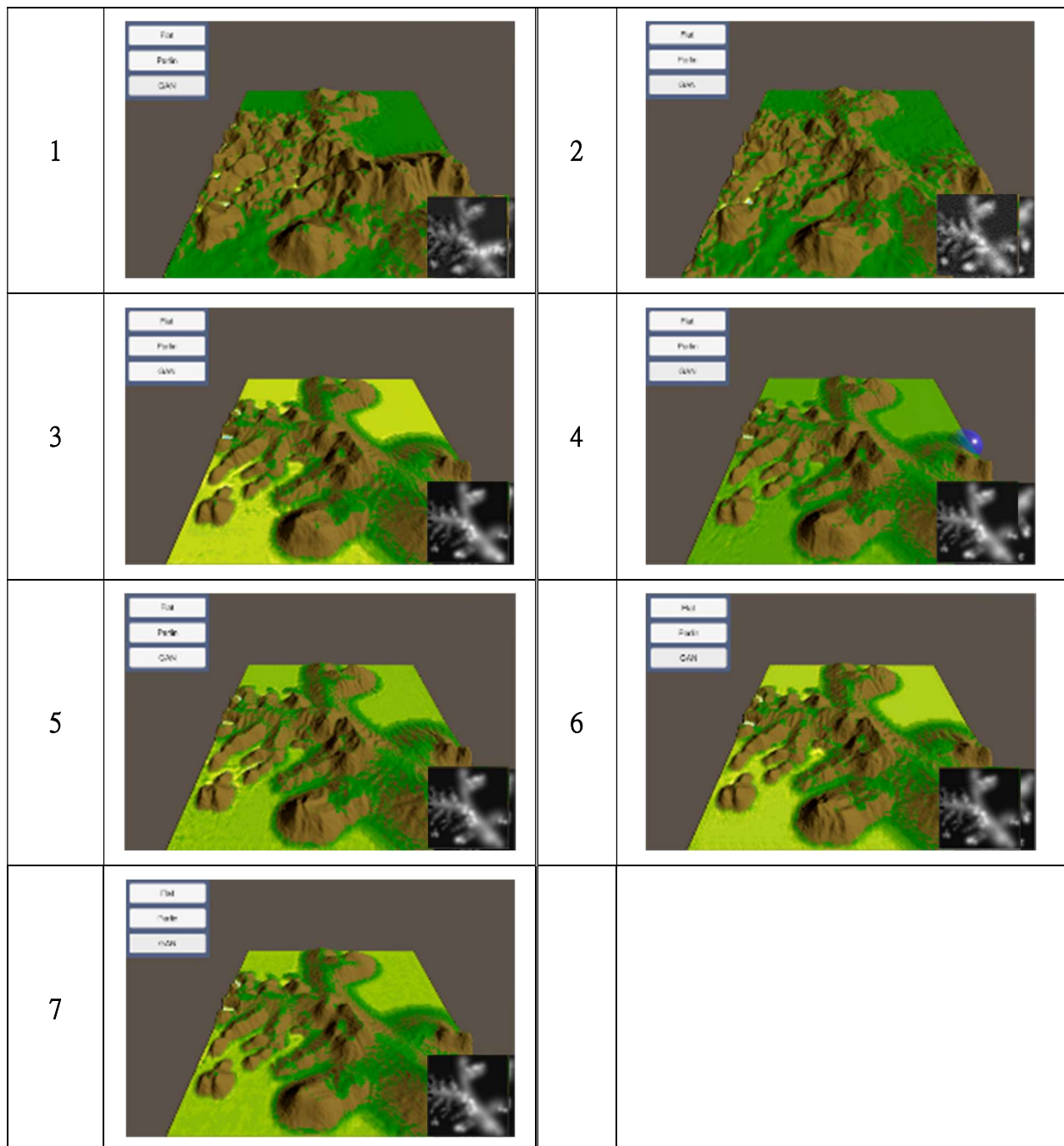
如同研究目的中所提到，地形擬真模型的目標是要能將人工手繪的大致地形架構加上山脈細節，所以我們也測試了若干不同模型，輸入相同高度圖，觀察模型輸出會有甚麼差別。如圖二十，此 3D 模型是由我們人工手繪出的山脈大致架構生成之結果。其中，右下角的灰階高度圖將作為我們輸入圖像，輸入到上述 7 個模型中。



圖二十、人工手繪的山脈大致架構

表九、輸入手繪圖時，各模型的輸出比較

模型編號	模型輸出	模型編號	模型輸出
------	------	------	------



可以發現訓練資料只有一種模糊程度的模型 1 與模型 2 的模型輸出較為雜亂，生成出的圖片中有些地方沒有符合原本的架構，我們推測是因為輸入圖像與 kernel size 為 29 的中值模糊圖像相差較大，導致模型沒有辦法很好的預測輸出。

訓練資料集中具有多種模糊程度的模型 3 ~ 7 都可以很好的保留原本的山脈架構，可以發現具有動態權重層的模型在河流的刻畫上會比沒有動態權重層的模型(模型 1 ~ 4)表現較佳，這也顯示在具有動態權重層的模型其 L1 Loss 比較低。

三、 pix2pix 伺服器：

為了避免每次需要應用 pix2pix 模型處理高度圖時都要重新載入模型，我們利用 Python 的 Flask 套件建立了一個 API 伺服器，並將其建置於工作站上。用戶可以直接上傳高度圖，並在伺服器上用訓練好的 pix2pix 模型進行處理。

這個 API 可以快速的在模型之間切換，且支援本研究中所有訓練的模型。

四、用戶端程式：



圖二十一、所有用戶端程式

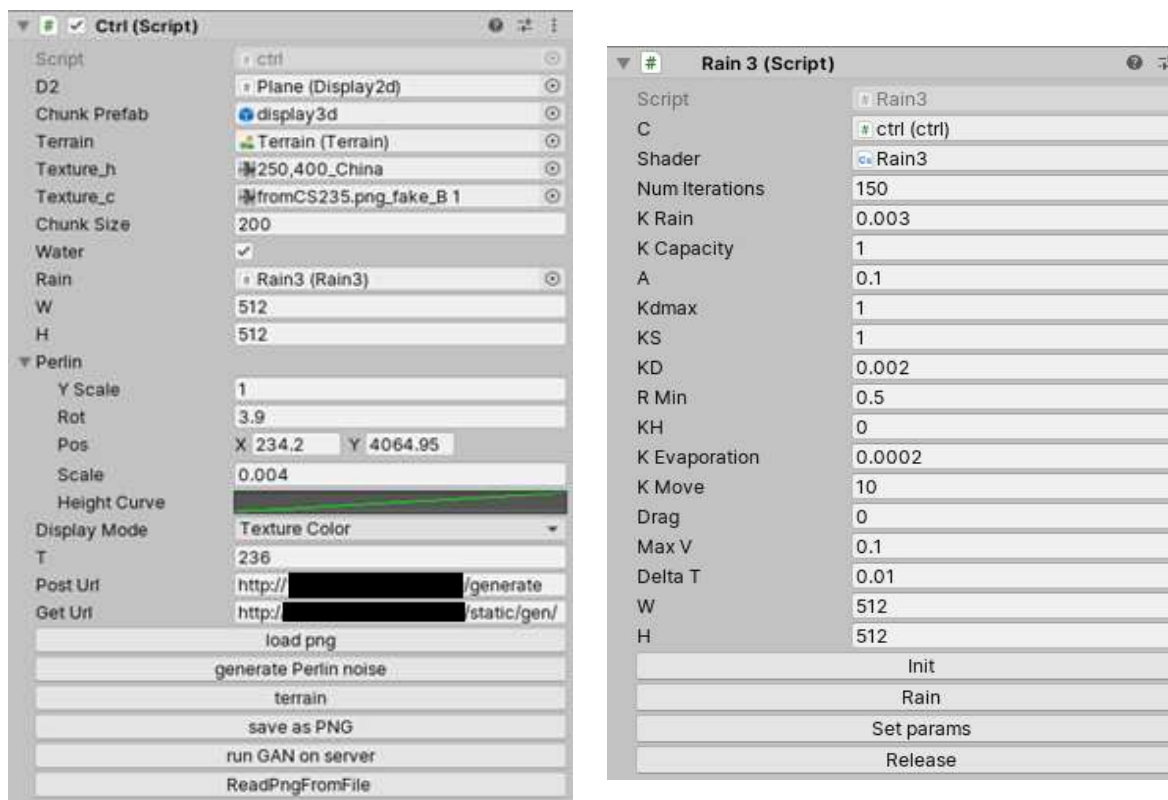
我們利用 Unity 及 web 技術分別開發了兩個獨立的客戶端程式：Unity 及網頁客戶端。客戶端可以讓用戶在圖形使用者介面上直接使用模型，而不用自行輸入複雜的指令。

其中 Unity 可以直接繪出 3D 模型，而網頁端則是在黑色畫布上由用戶自行繪製灰階高度圖。以下將詳細描述兩客戶端的詳細功能。

(一) Unity：

在 Unity 中，我們寫了兩個 script：Ctrl 和 Rain，讓使用者可以在 Unity editor 的 edit mode 直接對地形進行操作。使用者可以選擇用 PNG 灰階高度圖、Perlin noise 或 Unity 的 terrain 物件上畫好的地形作為初始地形，並按“run GAN on server”按鈕用伺服器上的 pix2pix 模型處理地形，或按 Rain script 的“Rain”按鈕模擬論文[1]中的物理侵蝕。使用者對地形每做一個操作後都會更新顯示出來的地形。操作完成後可

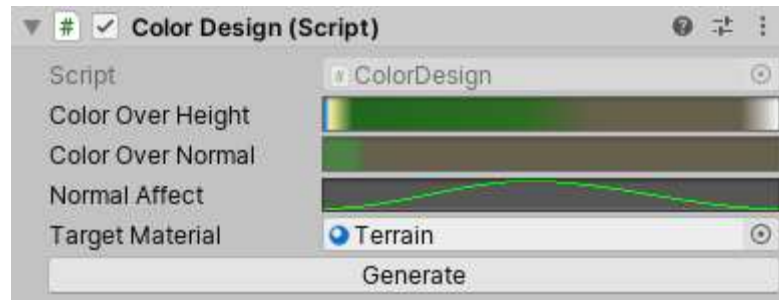
以按” save as PNG” 輸出新的高度圖。



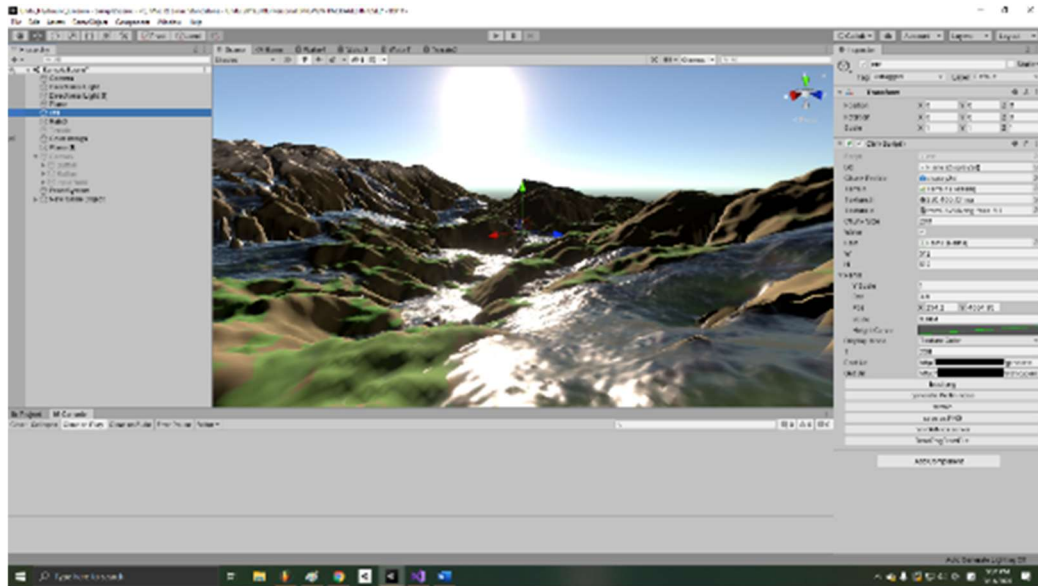
圖二十二、Unity 用戶端之操作介面

因為在 Unity 中生成過大的 mesh 會造成錯誤，所以我們把地面高度圖和水面高度圖分別分成多個邊長約為 200 的正方形區塊，為每個區塊生成一個物件並指定它們的 mesh，並拼在一起。

至於山脈地形的呈現方式，使用者可以利用 ColorDesign(如圖二十三)調整特定高度、斜率要顯示甚麼顏色。



圖二十三、Unity Color Design 介面



圖二十四、Unity 整體介面呈現

但不一定每個使用者都有 Unity 的編輯器，所以我們也利用 Unity 內建的功能建置了.exe 版本的 Unity 客戶端，可以直接在電腦上執行且畫面更簡單、易用。有平地 and Perlin Noise 可選擇作為初始地形。使用者可以用鍵盤的 W(前)A(左)S(後)D(右)在地形上移動，直接用滑鼠建構地形。按下“GAN”用伺服器上的模型處理地形。



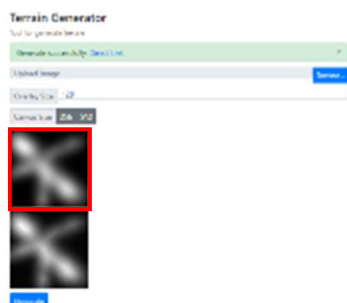
圖二十五、應用程式之起始畫面



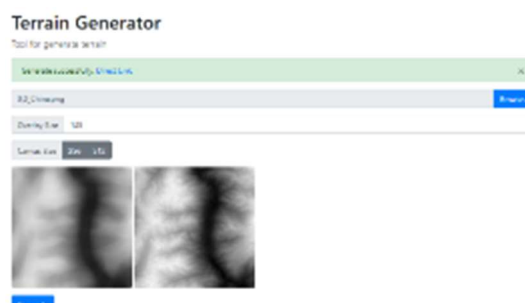
圖二十六、經由 pix2pix 模型後所產生的結果

(二) 網頁：

我們利用 HTML 和 JavaScript 與 JavaScript 套件 three.js(用於顯示 3D 山脈模型)及 p5.js(使用戶能自行在網頁端繪製灰階高度圖),建置出一個用戶能自行上傳灰階高度圖或是直接繪製灰階高度圖的網頁端,再利用伺服器端的 pix2pix 模型生成其加上山脈細節的灰階高度圖,並根據此灰階高度圖,在畫面下方顯示其 3D 模型。



圖二十七、用戶可以在黑色的畫布上自由繪畫(圖中紅框),繪製灰階高度圖



圖二十八、用戶可以將灰階高度圖上傳,網站會回傳灰階高度圖及一個可以拖拉的 3D 模型。

陸、 討論

一、 討論：

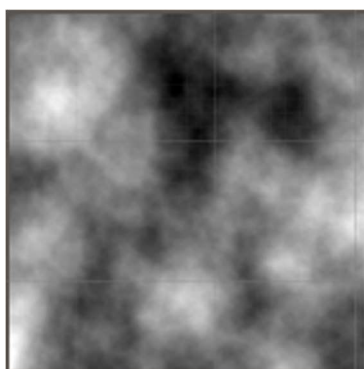
(一) pix2pix 模型與物理侵蝕模型的比較：

1. 雖然 pix2pix 模型訓練時需要較長的時間(在工作站上約需 3 小時),但只要模型訓練完成後,生成山脈地形的速度會非常的快,約 0.2 秒就可生成一張 256 x 256 像素的山脈圖。但侵蝕模型則相反,它不需要進行訓練,也不用收集真實的山脈資料,可是侵蝕模型在運算時需要疊代多次,無法達到快速運算的目標,如疊代 3000 次時需要約 10 秒。
2. 為了生成出最好的結果,侵蝕模型需要調整各種參數,需要不斷測試每一種參數組合才有可能達到效果。而 pix2pix 模型則不需要調整參數,雖然山脈的風格(河流的深度、一張圖內有幾條河流等等)會有所侷限,不過這可

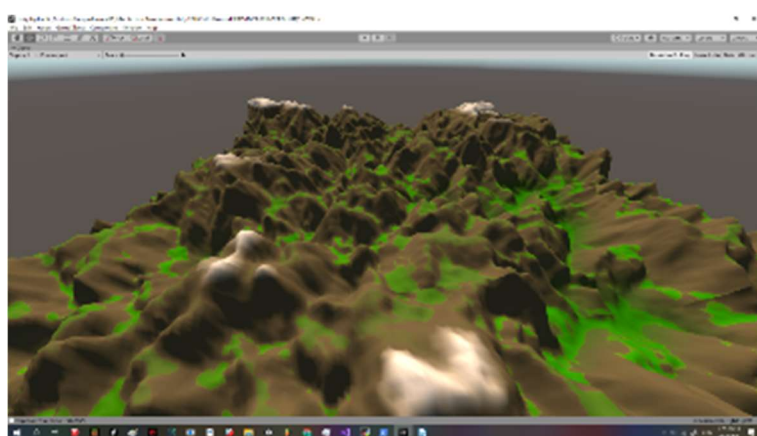
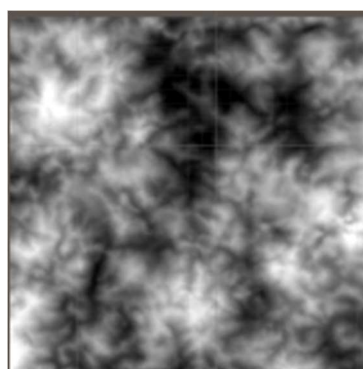
以藉由訓練時的圖像來改變。而且 pix2pix 模型還可以完成空照圖與灰階高度圖的相互生成，這是侵蝕模型無法做到的。

綜合以上，此研究中的 pix2pix 模型較傳統的侵蝕模型有以下幾點優勢：不須複雜的參數調整即有極好的效果、運算快速、模型的用途廣。

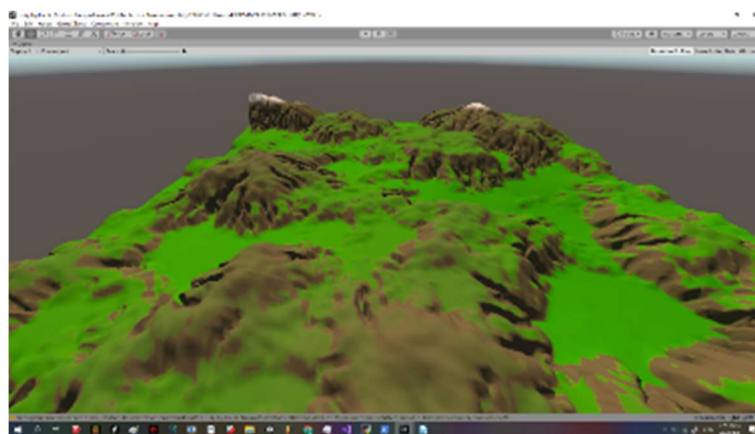
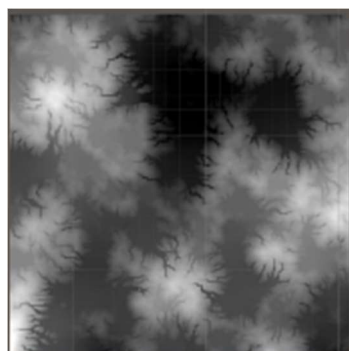
以下截圖為 pix2pix 模型及物理侵蝕模型的比較，兩個模型的輸入皆為相同



圖二十九、測試用的 256 x 256 像素之灰階高度圖



圖三十、pix2pix 模型所生成之地形



圖三十一、物理侵蝕模型所生成之地形

的灰階高度圖，如圖二十九。

在圖三十一中，可以發現 pix2pix 模型所生成的地形較接近真實的山脈地形。

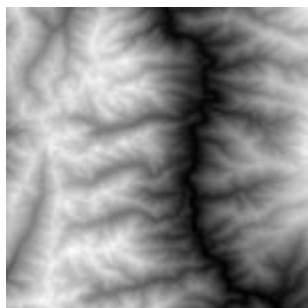
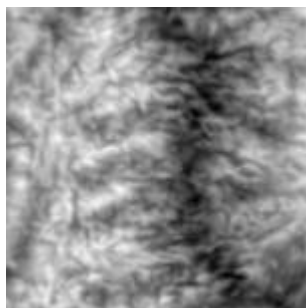
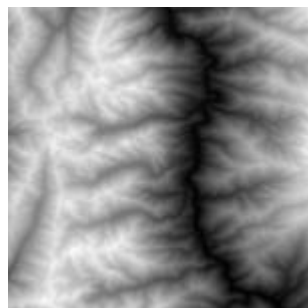
(二) 100 x 100 灰階高度圖及 256 x 256 灰階高度圖的比較：

我們訓練輸入長寬為 256 像素圖像的模型是因為長寬為 100 像素的圖像可能沒辦法在一張圖中得到更多的細節。訓練後要進行比較時因為圖像解析度過低也不太能夠準確的比較。採用 256 x 256 的灰階高度圖能有效避免上述的問題。

(三) 不同的訓練集是否可以使輸入圖像不須經過中值模糊即得到良好的效果：

為了驗證以上的討論，我們將一張未中值模糊過的真實高度圖當作輸入圖，輸入到模型 1 以及模型 4 中，這兩個模型唯一的差別只有訓練圖像集的模糊程度是否隨機。

表十、比較模型 1 與模型 4 在輸入圖像未經過中值模糊輸出的差異

測試用的 256 x 256 真實 高度圖 (輸入)	模型 1 輸出	模型 4 輸出
		

可以發現當輸入圖像並未經過中值模糊時，模型 1 的輸出會顯得十分的雜亂，但模型 4 的輸出與輸入圖基本相同。這代表模型 1 並沒有學習到如何正確的加上山脈的細節，使其變為更為真實。但模型 4 學到了當輸入圖像已經相當真實時，就不需要對此圖像進行額外的處理。

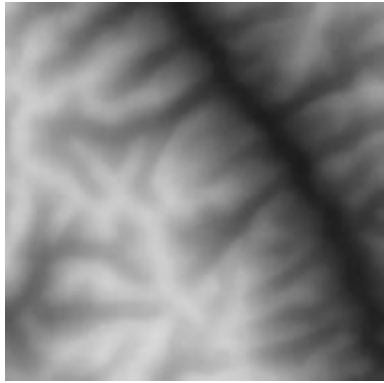
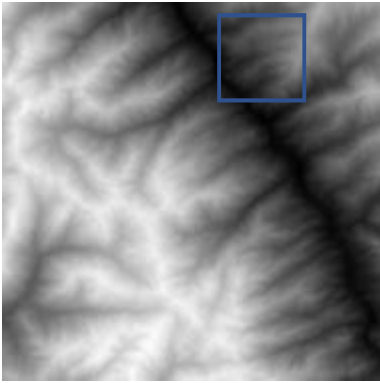
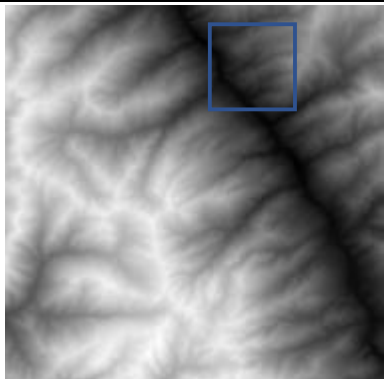
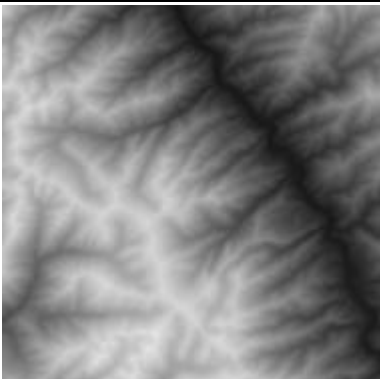
綜合以上，我們發現只要改變訓練集中模糊程度的差別，就可以很好的避免模型在已經具有豐富細節的圖像再加上一層細節。

(四) 加入「動態權重層」後的輸出效果差異：

為了驗證以上的討論，我們將一張經過中值模糊(kernel size 為 13)的圖像當作輸入圖，輸入到模型 4 及模型 6，這兩個模型的差別只有模型 6 動態權重層，

模型 4 則無，及模型 6 的 ngf 為 48、模型 4 則為 64，其他包括 U-Net 的深度及訓練集都完全相同。

表十一、模型 4 及模型 6 之輸出比較

輸入圖像	模型 4 輸出
	
模型 6 輸出	目標輸出
	

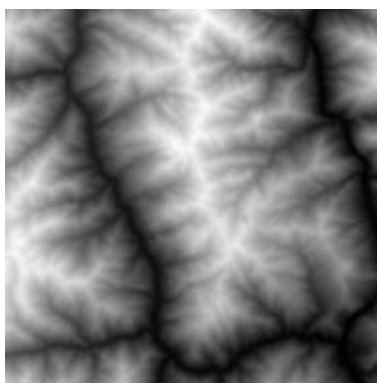
可以發現模型 4 與模型 6 的輸出中，模型 4 的輸出不會產生明確的，整體來說較為模糊，反觀模型 6，其河谷的部分較為明顯，這就說明了加上動態權重層的模型表現會較佳，且值得注意的是，模型 6 的 ngf 比模型 4 的還少，這就說明了加上動態權重層的會使模型在較少參數下，表現仍比較多參數但沒有動態權重層的模型好。(模型 4 與目標輸出的 L1 Loss 為 0.0500，模型 6 則為 0.0474)

(五) pix2pix 模型是否能將低解析度的灰階高度圖轉換為高解析度的灰階高度圖：

為了驗證以上的討論，我們將相同的 256 x 256 像素灰階高度圖(圖三十三)利用 OpenCV 內建的 resize 函數將圖片縮小成特定大小(分別為 128 x 128 像素、64 x 64 像素、32 x 32 像素、16 x 16 像素、8 x 8 像素、4 x 4 像素)，再利用 resize 函數將其放大回 256 x 256 像素之灰階高度圖，其中缺少的像素 resize 函數預設

會使用雙線性插值 (bilinear interpolation) 的方式進行線性插值。


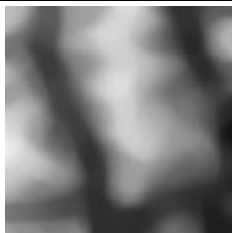
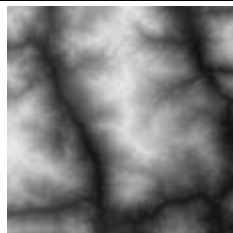


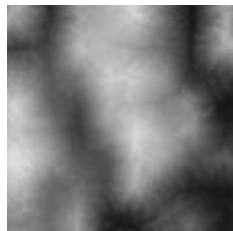
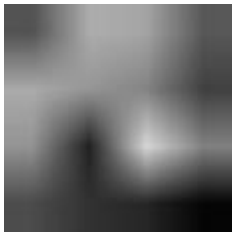
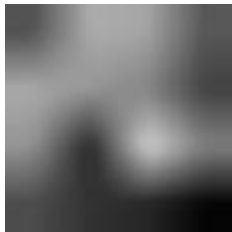
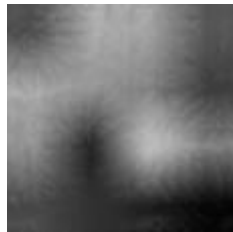
特別的是，經過雙線性插值的灰階高度圖在輸入到模型前，需要對其進行中值模糊，否則生成出的圖像效果會不好，我們推測其原因是因為中值模糊會將圖片的高頻與低頻部分去除，使得模型認為這張圖為缺少細節的高度圖，進而將它加上細節，反之則否。



圖三十二、原始灰階高度圖(長寬皆為 256 像素)

表十二、各解析度灰階高度圖輸出比較

縮小尺寸(像素)	經 resize 函數放大之高度圖	經中值模糊後之高度圖(輸入)	模型生成之高度圖(輸出)
128 x 128			
64 x 64			
32 x 32			

16 x 16			
8 x 8			
4 x 4			

在表十二中，我們能發現 pix2pix 模型在解析度為原圖的八分之一(32 x 32)時，仍能大致將低解析度的高度圖轉換為高解析的高度圖，且生成的圖片與原始圖片(目標輸出)相差不大，代表 pix2pix 模型是可以將經過處理(中值模糊)的低解析度高度圖轉換為高解析度的灰階高度圖，且效果十分良好。

此功能主要能應用在將較低解析度的灰階高度圖，轉換為高解析度的灰階高度圖，同時保證此高解析度的灰階高度圖是在一定程度上符合真實的情況。

二、 未來展望：

(一) 網頁用戶端的加強：

目前網頁用戶端只能在平面的畫布畫上灰階高度圖，相較 Unity 客戶端較不直覺，未來或許可以將其調整為直接繪出 3D model。

(二) 縮短利用 pix2pix 模型生成擬真地形的流程：

本研究將欲生成的擬真地形分開實作，所以達到一定的效果，但對應用上來說，仍嫌繁瑣，未來希望能再改善整個研究，讓從地形擷取開始，到最作後擬真地形的生成能以最簡單方式實作，不僅能簡化操作步驟，降低使用難度，亦能廣泛推廣。

柒、 結論

本研究中，我們把從 Google API 收集的灰階高度圖及衛星空照圖做為訓練資料，並根據 pix2pix 模型訓練了許多效果良好的模型，它們都可以有效的產出十分接近真實情況的圖像。

這些模型較傳統方法(物理侵蝕模型)具有生成快速、應用層面廣、生成出的山脈地形與真實的情形十分接近。我們也在模型中加上了自行研究的動態權重層，測試顯示，加入動態權重層會使模型輸出的整體效果變得更加的好。

為了使模型的使用更加簡單，而不是在終端機上打許多複雜的指令，我們將模型的使用包裝成兩個不同的客戶端：Unity 客戶端及網頁客戶端，兩個客戶端都可以在圖形使用者介面完成將模糊灰階高度圖加上山脈細節的工作，並能直接在畫面上顯示 3D 模型，雖然兩個功能有所不同，但基本上可以滿足大部分的用戶。

另外，我們也利用 Flask 套件編寫了簡單的 API 接口，使有需要的程式設計師或遊戲設計師能直接呼叫 API，使工作能更快的完成。

最後，本次研究的主要應用如下：縮短遊戲設計師在生成擬真山脈地形的流程及時間、將低解析度之地形高度圖轉換為高解析度之地形高度圖，同時保持高度圖的真實性。

捌、 參考資料與其他

一、 參考資料：

- [1] Jákó, B., & Tóth, B. (2011, November). Fast Hydraulic and Thermal Erosion on GPU. In *Eurographics (Short Papers)* (pp. 57-60).
- [2] Isola, P., Zhu, J. Y., Zhou, T., & Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1125-1134).
- [3] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems* (pp. 2672-2680).

- [4] Ronneberger, O., Fischer, P., & Brox, T. (2015, October). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention* (pp. 234-241). Springer, Cham.
- [5] Thread: Circuits
(<https://distill.pub/2020/circuits/>)
- [6] CycleGAN and pix2pix in PyTorch
(<https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix>)