

PRE-PROCESSING

Image augmentation

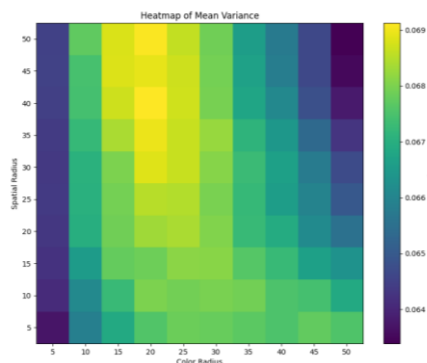
均值偏移(Mean Shift)

Mean shift 會將相似顏色或是空間距離相近的 pixel cluster 成一塊。因為要看色彩，所以在做 mean shift 前我們會先將 color space 轉成 Lab。另外，我們還要定義色彩和空間上的 threshold，也就是 spatial radius 和 color radius。我們使用一種特別的找法，目標是盡量讓每一點對模型而言差異越大越好。

首先，我們會先定義一個 spatial radius 和 color radius 的範圍 (5 ~ 50)。然後從這個範圍中取出 spatial radius 和 color radius 並對圖片做 mean shift。接下來，會將圖片加上 32*32 個 seedpoint，然後從每一個 seedpoint 開始做 floodfill。每做一個就和真實的 mask 做 IOU，這樣一張圖片就會有 32*32 個 IOU (如下圖)。



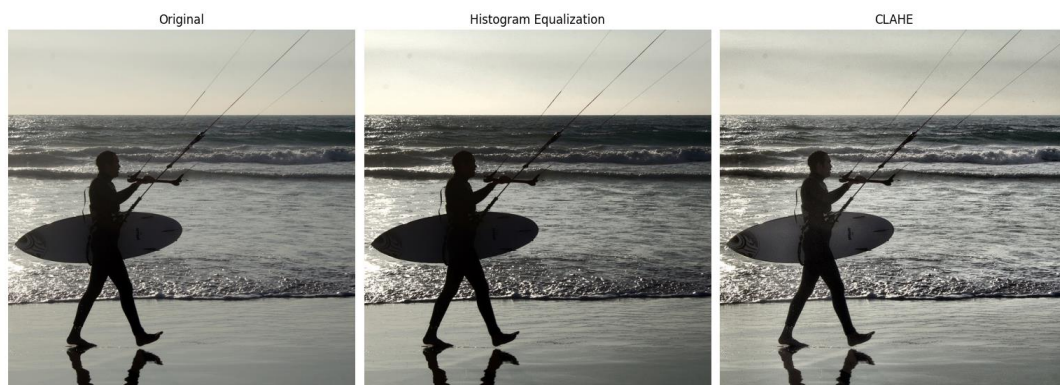
圖上越黑的部分代表 IOU 值越大。我們的目標就是讓這些 IOU 值分布越不均勻越好，因此，我們採用 variance 作為衡量標準。掃過所有 spatial radius 和 color radius 的 variance heat map 如下：



可以發現當 spatial radius = 40; color radius = 20 時 variance 最大，因此處理 mean shift 會採用此參數。

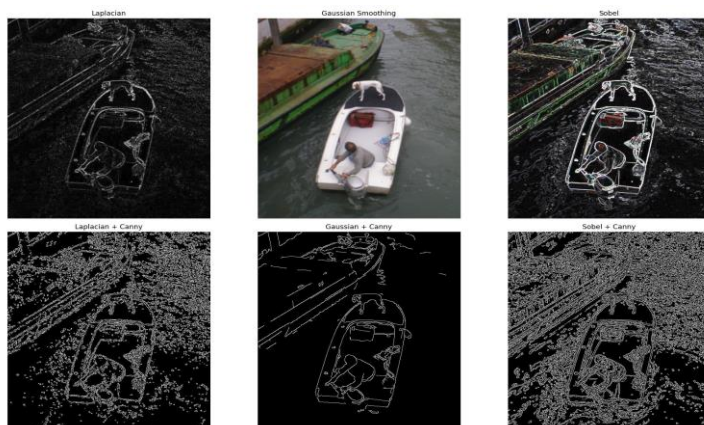
對比度加強

水體以視覺來說會是連接在一起的一整體，但受到光線亮暗的變化、反射的程度，進而使得圖片的取得中，水體產生了明顯錯差，為此為了避免光線影響到傳統影像處理方法對於水體的辨識，我們在過程中使用了兩種增強對比度的方法，在之前，我們先將圖像轉至HSV空間來對V (明度) 通道來進行增加，首先是上課所提的 histogram equalization，其核心原理是將影像的亮暗拉平，與 HW1 的實作相同，但其缺點也很明顯，因為對於資料處理的數據不佳選擇，會因此使得背景雜訊對比度的增加，並同時降低有用訊號的對比度，為此為了避免這種情形，我也使用了限制對比度自適應直方圖均衡化 (CLAHE)，能透過限制放大程度，來減少於過度放大圖像相對均勻區域中的噪聲，另外後者也能對參數進行調整，提供了設定上的彈性度



銳利度加強

為了增加後面使用 Canny edge detection 與 Harris Corner detector 抓取圖像邊緣特徵的能力，在做完亮暗度後會使用 Laplacian operator, Gaussian smoothing, Sobel operator 來加強銳利度，也就是增強高頻訊號，下圖為經過三種方式的原圖，與使用 Canny operator 後的結果，能發現使用 Gaussian smoothing 效果相對最佳，除了能抓到特徵外，也能減少對於如水波、漣漪的特徵抓取，避免錯誤資訊提供給 model。



彩度加強

這部分對於影像的加強，主要是希望能增強影像鮮豔度上的對比，幫助模型辨認，以視覺來說，天空與水明顯是不同的兩個整體，然後以像素點的角度來觀察，兩者事實上十分相似，以傳統作法及深度學習做法都容易難以區分，為此，提升對比度的方法能加強兩者的差異性，設法避免錯誤辨認，我使用的做法是先將圖像轉至 HSV 顏色空間，並對飽和度進行 CLAHE，結果如下圖:



Generate the image

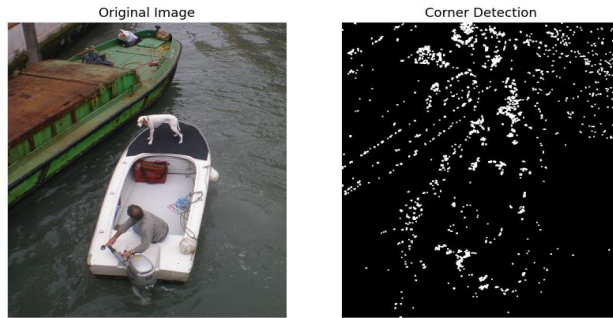
因為此次深度模型所使用的訓練圖片較少，所以送入模型的圖片希望能有包含明顯的特徵以及相關資訊，讓模型能夠更加清楚的分辨水體位置，為此，我使用了下列的方法

邊緣偵測(Canny operator)

我使用上方所提的影像加強方法加強圖片，並設定閾值來使用 Canny operator 抓住邊緣特徵，也就是像素值明顯出現變化的位置，除了調整閾值抓出符合條件的特徵外，我也設法調整影像加強技巧，盡量希望能找出符合以下條件的圖片，首先圖片特徵、物體的邊緣能盡量被抓到，第二，避免不必要特徵的抓取，如水波、漣漪，避免模型錯誤的辨識，理想的輸出圖片如上方銳利增強圖片的下二，使用影像加強技巧中，能最符合上述條件的是 kernel size 適中的 Gaussian smoothing。

角點偵測(Harris Corner detector)

實際實作中，水體並非都是平坦的，可能會有水花、水波、漣漪、波浪等非理想情況需要判斷，為此，與邊緣相比，我們認為角點的存在並非是水體與其連帶效應容易出現的，因此使用上課所提的 Harris Corner detector，尋找圖片中的各式角點，角點分布越密之處，通常為物體的分布之處，希望能讓模型理解這是最不可能出現水體的位置，並加以判斷水體，其抓取效果如下



找尋輪廓(Find the contour)

邊緣找到後，若不經過處理仍然是斷斷續續的片段，以視覺角度來說確實能以此為根據抓到影像特徵，但對於電腦角度仍不明顯，所以需要先做邊緣的連結，我使用的方法是 Morphology 方法中的 deflation，先設定固定的 kernel，沿著邊緣不斷放大，設法連接所有斷裂的邊緣，畫出更大的曲線，並使用內建函式找出圖像中重點資訊的輪廓

過濾輪廓(Filter the contour)

經過上方的步驟後，圖像會出現大小、週長皆不同的輪廓，有可能是大輪廓沒辦法一次抓到而被迫分解的，也有可能是上述因為不想要的特徵如水波、白雲、波浪、漣漪等所導致，但這些特徵通常是細長且較為不明顯的，因此距離主要特徵較遠的部分所圍繞出的輪廓通常在面積上、週長上都會較小，我們就能以此為基準濾掉大部分不想看到的特徵，但同時這個方法也是一把雙面刃，若取得邊緣特徵十分瑣碎，進而使得輪廓抓取也仍瑣碎，大部分相關的資訊都被消除乾淨了，因此，找尋輪廓對於參數、以及影像增強的技巧選擇十分要求。

Image for training

edge detection(下左一)

經過影像增強後，使用 Canny operator 找出邊緣，並未經過連接

Harris Corner detector(下左二)

經過影像增強後，使用 Harris Corner detector 抓出角點

Draw contour(下左三)

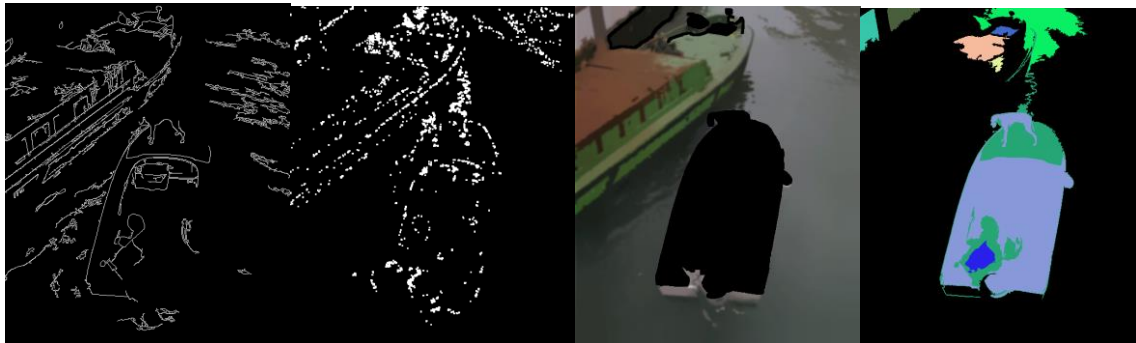
經過影像增強、邊緣偵測，使用形態學連接邊緣

Fill color(下左四)

結合以下的影像增強效果所產生的圖片所抓取的特徵來結合

1. 原圖
2. 原圖經 sobel operator
3. 原圖經高斯模糊後，再經 sobel operator

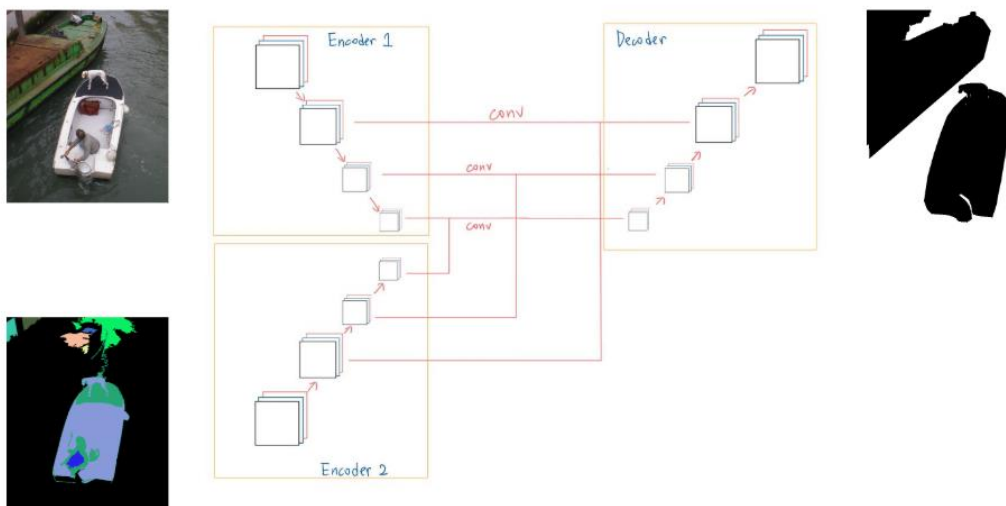
圖像分別經過上述不同影像增強效果後，使用簡單的灰值二值化，並以此數據進行輪廓的抓取，進行濾除，並將輪廓所包圍住的區域進行塗色



Model Design

Unet:

Image segmentation 最大的特點就是輸出的維度非常大。因此，如果沒有在模型中間降低 dimension，可能會導致模型參數量過大。我採用 Unet 作為本次專題的骨幹。Unet 有一個 encoder 和 decoder。Encoder 負責將圖片 map 到高維空間，而 decoder 負責 reconstruction。此外，會將 encoder 和 decoder 間做 skip connection。另外，除了使用傳統的 Unet 以外，因為本次訓練的數據量非常少，因此需要額外對模型做一些 constraint。我們額外將前面預處理的圖片丟進去模型做訓練，因此 Unet 必須同時讀兩張圖片，也就是兩個 encoder。一張是原圖或是 mean shifted 後的圖片，另一張是預處理後的圖片 (如 edge detection 等等)。經過兩個 encoder 各自壓縮後，在經過一層 convolution 將兩張圖片的特徵融合並送給 decoder，模型架構如下圖：

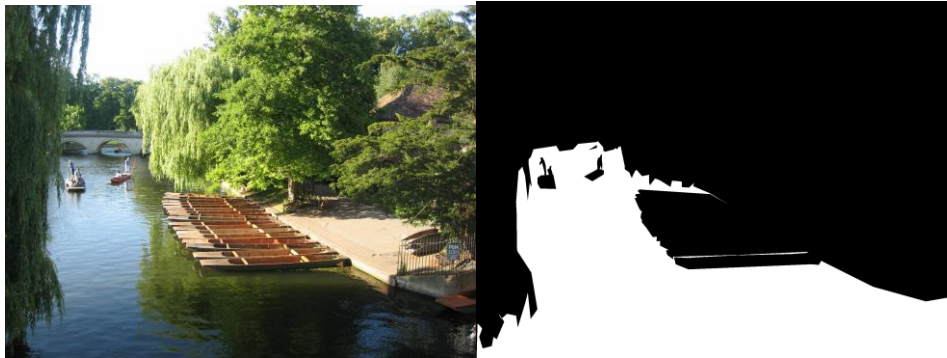


Training method:

訓練方法就大同小異，包括使用 learning rate scheduler，還有使用 weight decay 讓模型的參數更接近 0 做 regularization，並且在 decoder 階段使用 dropout。

Training result:

下面的表格數字為 testing set 的 平均 IOU，下圖為範例的 image 和 golden mask







Input Image:

在輸入圖片方面，總共分為原圖或經過 mean shift 的圖片為一組，經過 edge detection 或 corner detector 等等的圖片為一組。

使用原圖:

fill color	draw contour	edge detection	corner detector
0.3308	0.2921	0.3312	0.3248
<div>Predicted Mask </div>	<div>Predicted Mask </div>	<div>Predicted Mask </div>	<div>Predicted Mask </div>

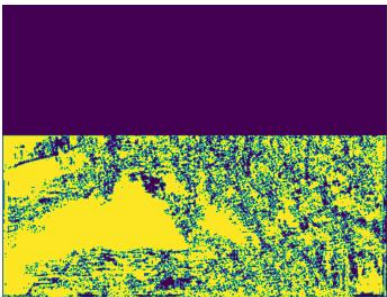

使用 mean shifted 後的圖片：

fill color	draw contour	edge detection	corner detector
0.4232	0.3099	0.4066	0.3583
			

可以發現使用 mean shifted 後的圖片 IOU 皆比較高，推測是因為 mean shift 雜訊濾掉，讓模型能夠更專注於更主要的邊緣。

Model structure:



這裡主要改變的是兩個 feature 融合時的 kernel size，image encoder 和 decoder 的大小保持不變。

fuse kernel size = 1	fuse kernel size = 3
0.3312	0.4066
	

當 kernel size 為 3 的時候，可以發現 IOU 提升不少，推測是因為圖片對於其他圖片的附近位置有所關聯，不僅是和同一個位置的 pixel 有關。

Data augmentation:

這裡主要探討是否要將圖片做 vertical flip，從訓練資料中可以發現，水和天空往往特徵相當類似，但水大部分出現在圖片的下方而非上方，因此，我們會希望模型去學習到水在下方的 bias，進一步提升預測的準確度。

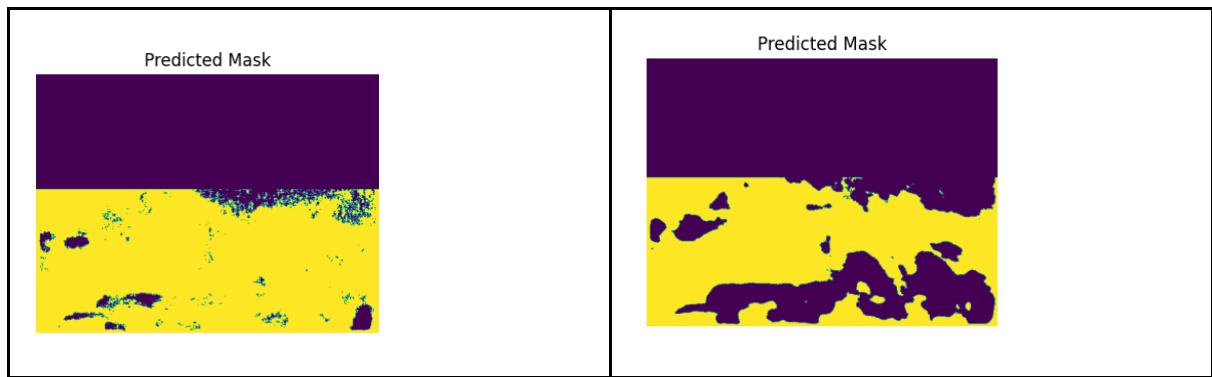
wo flipping	with flipping
0.4066	0.3321
<p>Predicted Mask</p> 	<p>Predicted Mask</p> 

loss function:

loss function 對於模型的訓練也很重要。這裡比較兩種 loss function，一種是最基本 binary cross entropy loss，另外一種是 dice loss。

Dice loss 比較專注於整張圖片，而 binary cross entropy loss 較偏向 pixel wise，在這個資料集當中，可以發現 class imbalance 的問題較嚴重，且 dice loss 優化的目標與 IOU 相似。因此，也可以從結果發現，dice loss 相比 binary cross entropy loss 表現較高。此外，也可以發現 binary cross entropy loss 的結果當中含有較多的噪點。

Binary Cross Entropy loss	Dice Loss
0.3763	0.4066

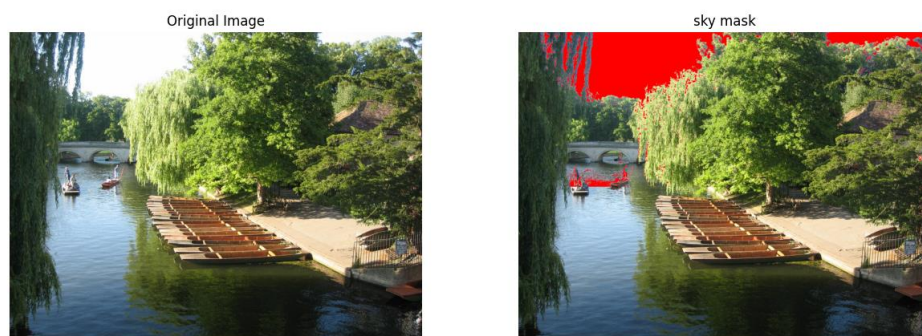


POST PROCESSING

由於測試資料過少，純粹使用深度學習的做法效果仍有限，為此，我們能藉由一些傳統作法來根據原始圖片上既定的特徵以及一些我們知道的常理來對深度學習所產生的結果進行修正，提升整體的效果，以下幾種為我們所使用的方法

a. 排除天空成分 (skymask)

天空與水有近乎相同的色彩，即使眼睛能判斷，資料量不夠大的模型事實上是難以區分的，兩者的顏色差異會隨著光線亮暗、反射、漣漪水波以及波浪等外部效應所影響，沒辦法以一個絕對的通道數值區間來將兩者區分，因此，將兩者區分的方法對於效果的提升會顯得非常關鍵，首先我們先對於天空進行簡單的概述，天空具有一定的藍色色彩，且通常是圖片中的最為光亮的部分，另外天空高機率會出現在圖片的上方，我們能根據上述的條件建立遮罩，只檢測圖片上方，同時檢測圖片光亮處以及從 HSV 顏色空間中檢測偏藍處，此方法能盡量抓到天空的特徵，與模型生成的圖用 AND 方式結合，大概效果如下



b. 排除非藍色成分 (post_process mask)

找尋水存在的部分是我們在此問題的核心主軸，但若將a所提之遮罩用於全圖，效果實際上會非常不理想，最主要的原因是水與天空在色彩上的呈現過於相似，對於天空與水的界定十分複雜，容易造成使用傳統方法容易造成水體辨識的過度解讀，進而使得不可逆的後果發生，為此，我們組別選擇換個思路來解決，找尋最不可能為水的位置，此方法使用傳統方法能解決的圖像問題相對能更為準確，提供深度學習模型生成結果修正檢測，我們組別所使用的步驟如下，首先，為了避免光線使得判斷結果出現誤差，我們先將圖像由 non-linear RGB 轉至 HSV 通道，並於明度通道 (V) 使用了限制對比度自適應直方圖均衡化 (CLAHE) 增強光線的對比度，再來去除最亮、最暗部分，同時選擇最不接近藍色位置，即偏紅綠色，此步驟能根據所選通道數值範圍簡單排除，最後設計出另一種遮罩，同時對於影像進行 AND 運算，如下圖所示，藍色是高亮位置，而紅色則是偏紅綠色位置，可以發現當找尋極亮部分，能高機率去除天空成分



c. 附加情況

另外還有一種遮罩，是將前處理所運作的技巧投入到最後來進行加強，若前處理做得很好，透過前面所提的 contour，我們能用 contour 抓取各類物體，而抓取的物體為非水出現的位置，一樣透過 AND 操作，我們就能將從非水的部分開始去除。

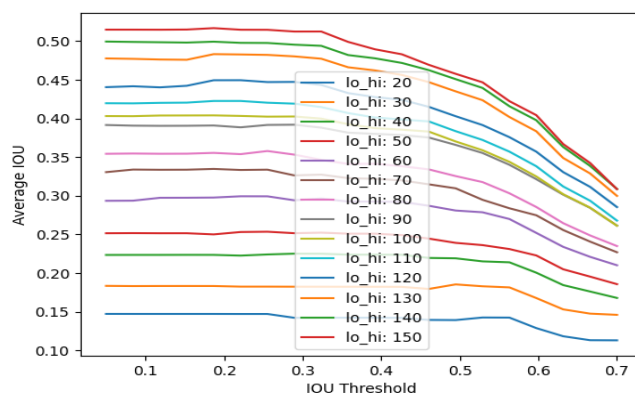


Other method

在直接用 Unet 訓練之前，我們曾嘗試將傳統方法與深度學習結合。因為水體通常的變化不大，所以在傳統方法當中，對圖片做 mean shift 然後做 floodfill 就能到不錯的效果。然而，傳統做法最大的問題在於不知道 seedpoint 在哪裡且不知道應該要有幾個，而且不知道判斷是否為同一個區塊的 threshold 為多少。如果能找到 seedpoint 和判斷是否要繼續填的 threshold，就能抓到水體且效果非常非常好。因此，我們打算使用深度學習模型來學習 seedpoint 的位置和對應的 threshold value。

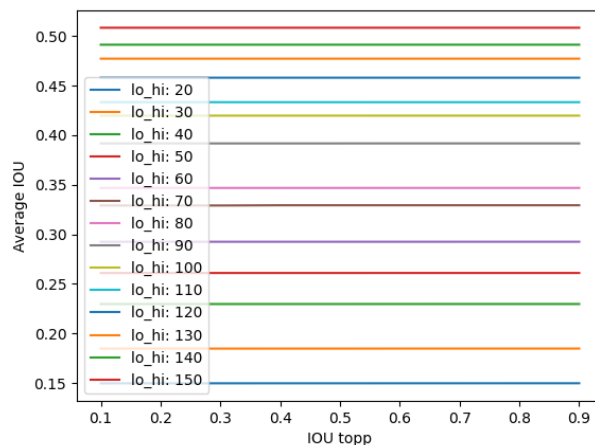


但在那之前，我們要先對每個圖片找出其對應的 seedpoint 的位置和對應的 threshold value。如下圖所示，我們會先將圖片打上 32*32 個 seedpoint。然後先預設一個 threshold value (50) 對於每個點去做 floodfill，做完後每一點都會有 IOU 值，越黑代表從該點的 IOU 值越大。得到這些 IOU 值後，我們要決定最終圖片要取那些點開始做 floodfill，因此。我們作了以下實驗。第一個實驗是直接取 IOU 值在 threshold 以上的 seedpoint，掃過部分的 threshold value 的結果如下：



可以發現只取 IOU 值為 0.2 的 seedpoint 可以讓 IOU 值最高。

第二個方案是只取前幾高的 seedpoints 。結果如下圖:



可以發現不論取 top 幾， IOU 的結果皆差不多，因此接下來會取前 10 % 的 seedpoints 來做 floodfill 。

決定好方案後，接下來要開始找正確答案，我們的做法採用類似 EM algorithm，會分兩步驟進行，目標是讓找到的 seedpoint 和 threshold value 使得 IOU 最大。E 步驟會計算在固定的 threshold value 下每個 seedpoint 所對應的 IOU 值。M 步驟嘗試各種 threshold 並從前面所計算的 IOU 值取前 10 % 做 floodfill，最後可以算出整張圖片的 IOU 值和其對應的 threshold 。

首先 E step 會先隨便取一個 threshold value，然後計算每個 seedpoint 所對應的 IOU 值。然後到 M step 算出最佳的 threshold。接下來再回到 E step，拿上一步 M step 所計算的 threshold 再計算一次 IOU 值。不斷的遞迴直到收斂。

```
Average IOU: 0.7238337796384621 for iteration 0
Starting Iteration 1
Finding seedpoints.....
Average IOU: 0.737822420389292 for iteration 1
Starting Iteration 2
Finding seedpoints.....
Average IOU: 0.7394840168693139 for iteration 2
Starting Iteration 3
Finding seedpoints.....
Average IOU: 0.7434506038571004 for iteration 3
```

結果如下圖，可以看到 IOU 值會隨著 iteration 不斷往上，也就是說，只要模型能夠學會前 10 % 的 seedpoint 的位置和其對應的 threshold，IOU 就能夠達到 0.75 左右。

然而，找到答案後，發現模型完全訓練不起來，也許更大的資料集和更好的訓練技巧能夠帶來更好的效果。

結論

傳統影像處理部分：

此次傳統影像處理技巧主要用於前處理及後處理上，也是此學期授課的主要內容，實作上，大部分情況能提供比少資料情況下的深度學習模型更佳的效果，但需要考慮不同情況的遮罩疊加，同時一定有部分區域，無論哪個遮罩都無法判斷的，此處就會變得複雜，另外，使用傳統作法對於水體的附加效應來說難以解決，如水波、波浪，漣漪等，很難以像素、邊緣檢測的方式明確定義平坦的水平面與波浪應該為同一實體，同時亦會受到光線的影響，另外，天空與水的區分也是一大挑戰，同時適用於部分圖片的遮罩對於其他圖片來說可能造成錯誤的誤導，找到一個通用且全面性的遮罩非常困難，另外，前處理的步驟選擇以及技巧使用則是另外一個挑戰，這會影響圖片後續的所有操作，整體來說，影像處理對於特徵十分明顯區塊十分有利，搭配形態學的做法，能有不錯的效果，但對於邊界判定不明顯、外部影響較大的圖片判定具有挑戰性。

深度學習模型部分：

本次深度學習並沒有發揮到很大的好處，因為訓練資料較少，若再切 validation set 後會發現 validation set 的資料量非常少 (大概 8 張圖片) 造成模型的 loss 相當不穩定，訓練非常的困難。另外，訓練資料少也常常 overfitting，因此特別將 dropout 開到 0.5 才稍微緩解這個問題。在資料量少的情況下，只能對模型多一點限制，多輸入一張處理後過的圖片更能夠幫助模型去提取圖片的特徵。然而，和傳統做法的結論相同，模型對於主要邊界的判斷相當精準，但是判斷完邊界後要判斷該區塊是否是水對模型和傳統方法都是很大的難題。也因如此，如果該區塊判斷錯誤，縱使邊界抓得非常精準，對於 IOU 的 penalty 還是非常大。這樣會導致模型最後在少量的 testing set 的 IOU 表現相當不穩定，這是我們在深度學習部分遇到最大的難題。

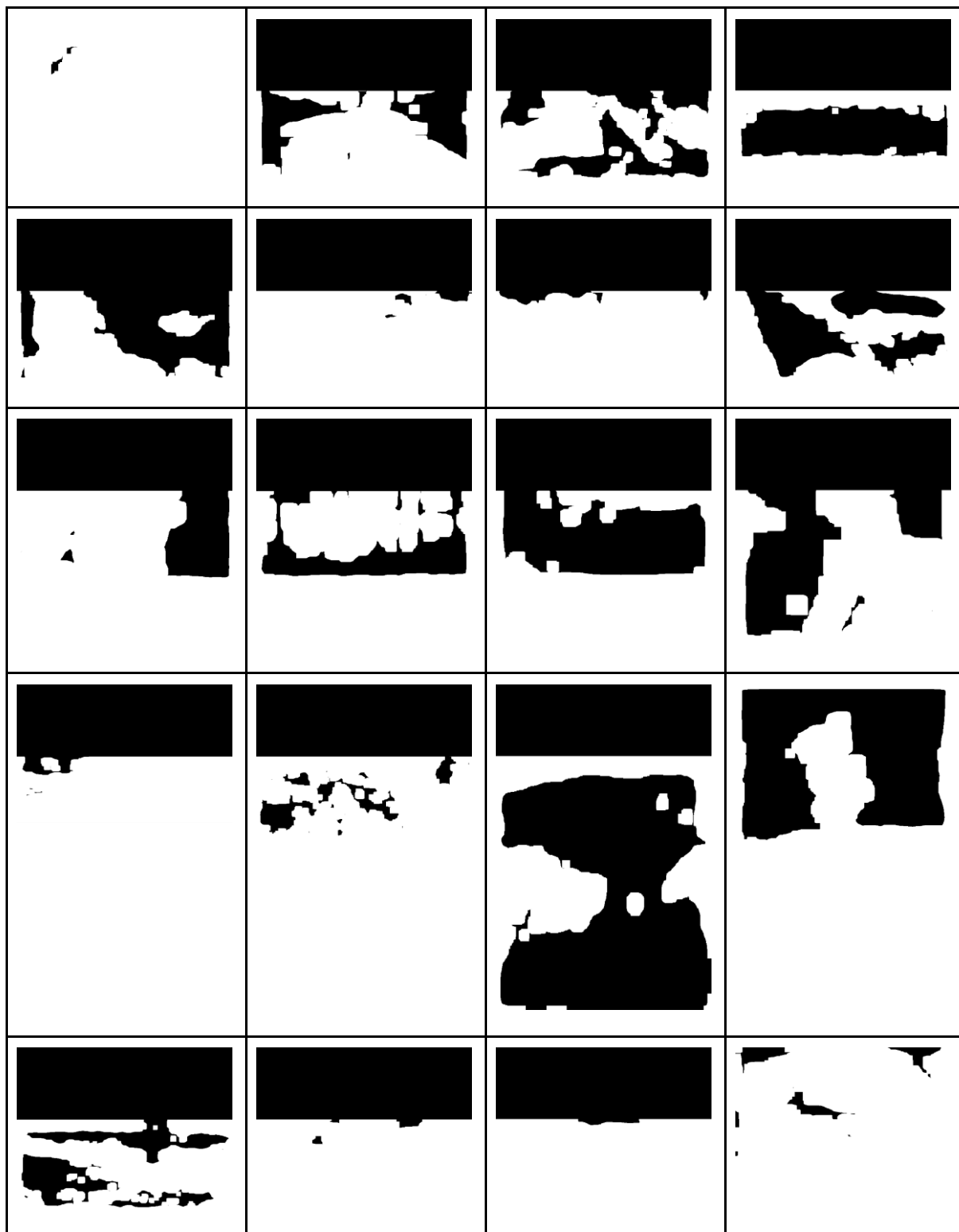
心得:

張語楹：本次作業一開始以為很簡單，但是當我打開作業的 zip 檔後，發現訓練的資料竟然只有 80 張，就知道這會是一個艱鉅的任務。我和隊友一開始就往深度學習的方向開始做，但做到一半後就後悔了，模型常常訓練不起來，loss 直接爆掉。最後也是一直嘗試各種排列組合才成功歸納出一些虛無縹緲的規律，只能說深度學習非常的玄。在資料量少的情況下傳統作法完全全提升了準確度。若單純用原本的圖片直接硬 train，IOU 完全卡在 0.2 上不去。但經過傳統做法的加持，IOU 提升了很多，也許這就是這門課學到的東西！

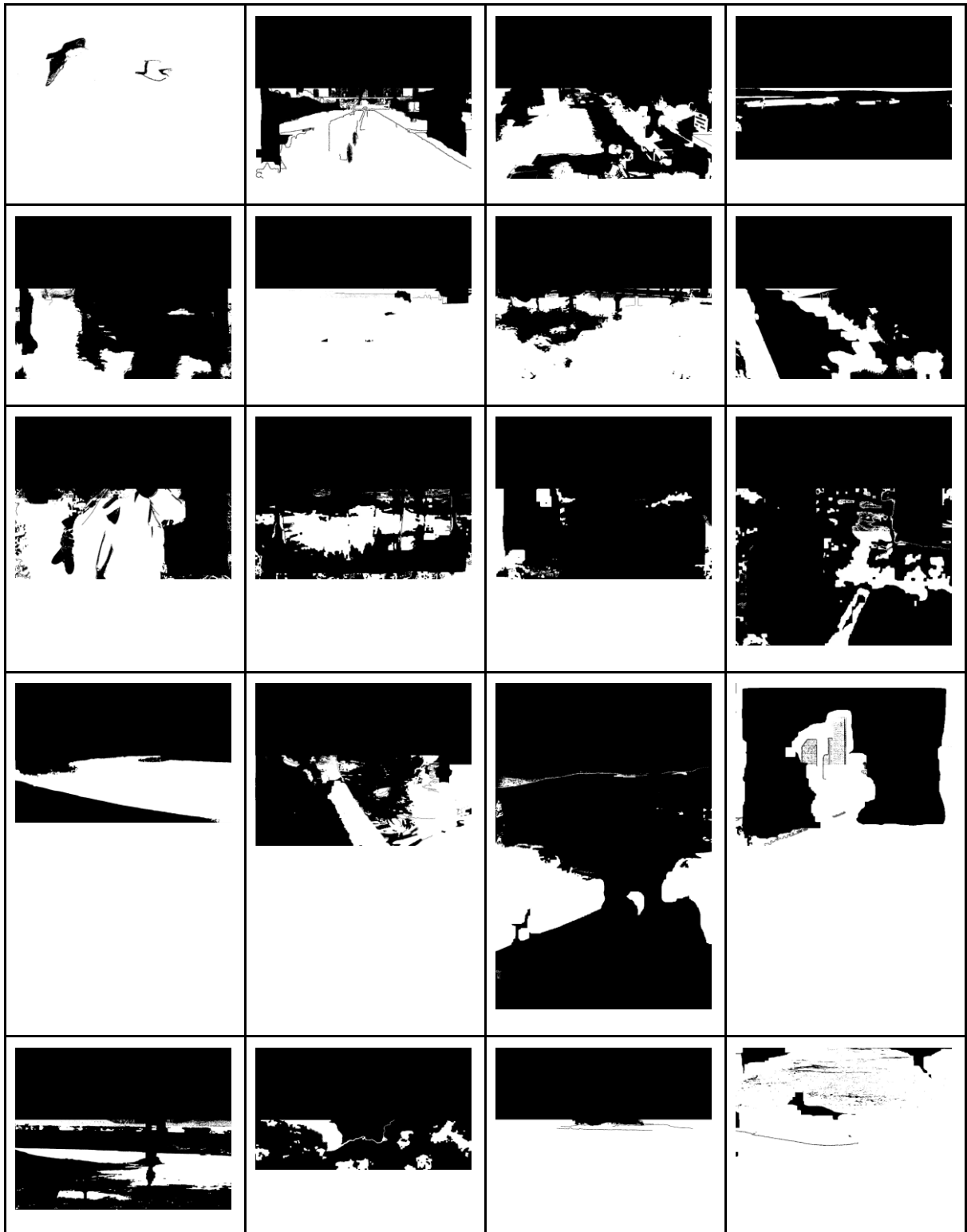
劉昱旻：這次作業有機會將上課大部分所提的各式影像處理技巧投入其中，從實作面上學習了不少，但是實際情況中，發現的問題也不少，首先圖像處理很能明確的指定特定對象做操作，也就是當對一個影像做操作時，很容易進而影響到周圍的像素點，造成不理想的效應發生，以Canny operator為例，肉眼很明顯能抓到物體位置，但實際進行邊緣偵測時，往往目標物難以完整抓取，同時進而抓取到如水波、漣漪等不理想效應，需要避免，就需要進行更多的前處理技巧及運作，整體來說，這次作業我們花了很多時間，無論在傳統影像處理技巧上或是深度學習方法，都嘗試了多種的排列組合，雖然實作上以及經驗上都十分不足，仍希望在未來能有機會將這學期所學投入在研究之中。

Appendix:

without mask (IOU = 0.4534)



with post_process mask (IOU = 0.4595)



original image:

