**Image Processing Homework 2 Report**

**P76111296 張中龍 使用語言: python**

***Questions:***

給一個資料夾，內含許多張照片，每張照片有三個class ，分別是powder\_uncover, powder uneven及scratch，將defect種類辨識並segment出來。

***Method***:

**訓練方法:**

**Yolov5**: train 300 epochs with batch size 4, image size 640\*640.

**U-Net**: train 175 epochs with batch size4, image size 300\*400.

另外，在U-Net訓練時，會將所有yolo偵測的box裁切下來，再依照位置將box對應的mask也裁切下來，作為U-Net的input。

**取得結果圖片:**先使用cv2及shutil套件，把scratch的照片複製並移動一個要給yolov5辨識照片的資料夾，而uncover及uneven由於圖片大小約為3384\*3330，因此先resize(長寬各\*1/3)後再複製並移動。接下來yolov5會辨識資料夾中的照片，並把辨識到defect的種類以及位置寫進txt。等yolov5完成辨識的動作，根據檔名開啟txt並讀取預測到的每個bounding box的class以及位置，再依據這些座標及class畫出對應的detection result。對於每張圖片，將其每個bounding box根據座標裁剪下來並一張一張(\*)resize後，送進U-Net，U-Net會產生對應的segment mask圖片，將此圖片依據預先紀錄好地bounding box class資訊變成對應的顏色，另外再創造一個與圖片一樣大小的全黑的mask，將每個產生的segment依照紀錄的bounding box座標還原到對應的位置與mask進行OR運算，等每個bounding box都裁減運算完，此mask即為整張圖的segmentation result。

(\*若是width>heigh，則會rotation 90度再送進去，之後會segment完再轉回來，以避免扁的圖片被resize成長的圖片)

**取得metric:**

**IOU**:在一張圖片中，將預測到的bounding box的位置與Json內的每個ground truth進行運算，取最好的值(且class相同)作為此box的的iou，將所有box的iou加總並除以box數即為圖片的IOU。

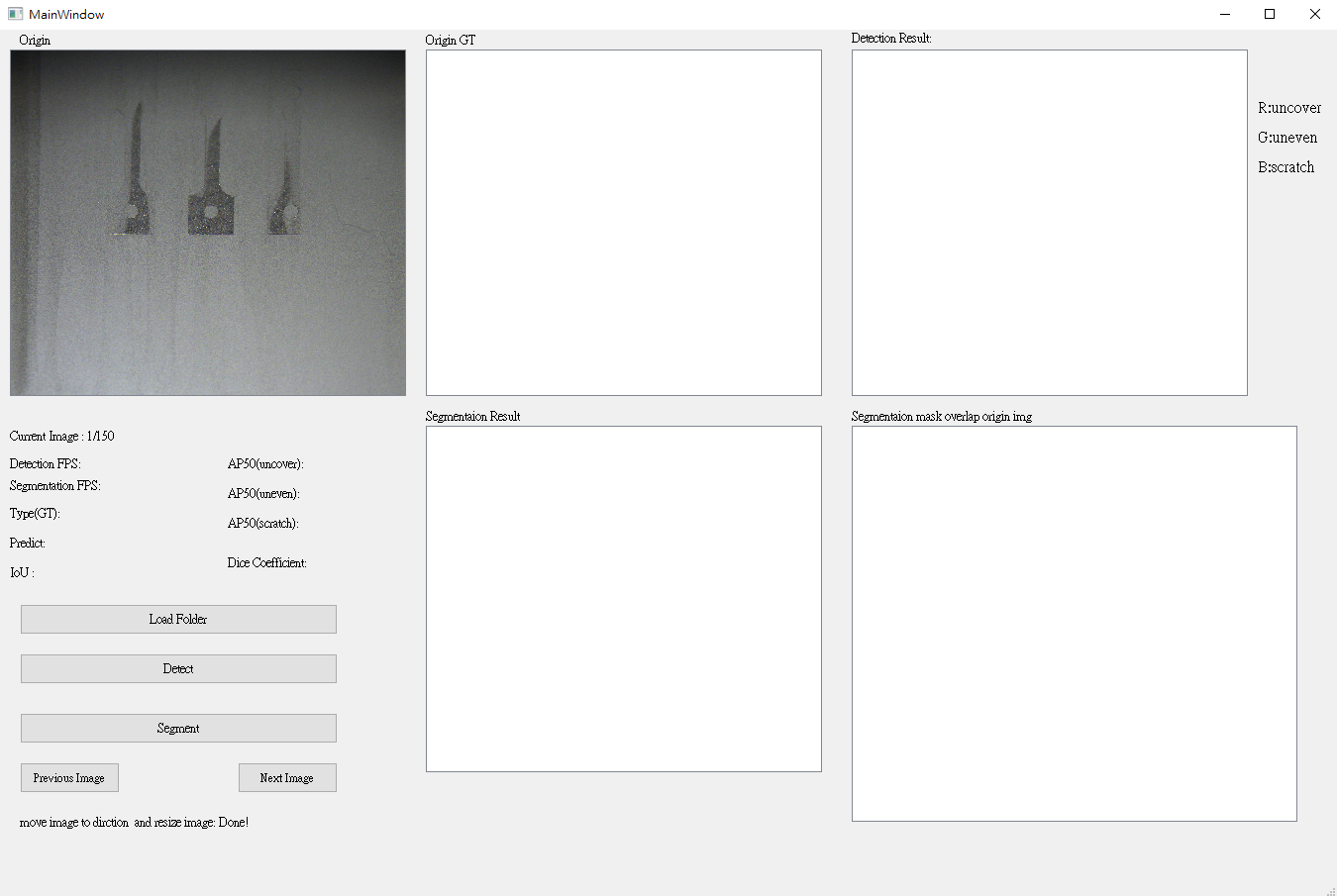
**AP:** 與IOU類似，在檢測每個box iou時，當box最好的iou>0.5則TP=TP+1，否則Fp=FP+1，對於每一類，皆有自己的TP及FP，所有image共享這兩個參數，在所有影像接辨識完後，一類的AP則為對應的。

**Dice coefficient**:將result segmentation mask(x)及answer segmentation mask(mask資料夾裡的正確答案(y)轉為單通道，並用threshold使有顏色的值皆為255其餘仍為0，將兩個mask做and運算，可取得intersection mask(x and y)，將這些mask的值除以255，再利用numpy.sum()則可得出mask白色的pixel數，也就是可以得知|x|, |y|,|x and y|，而Dice則可通過 計算得出。

**FPS:**利用time.process\_time()取得執行前和執行後的時間，兩者相減即為FPS

***Results:***

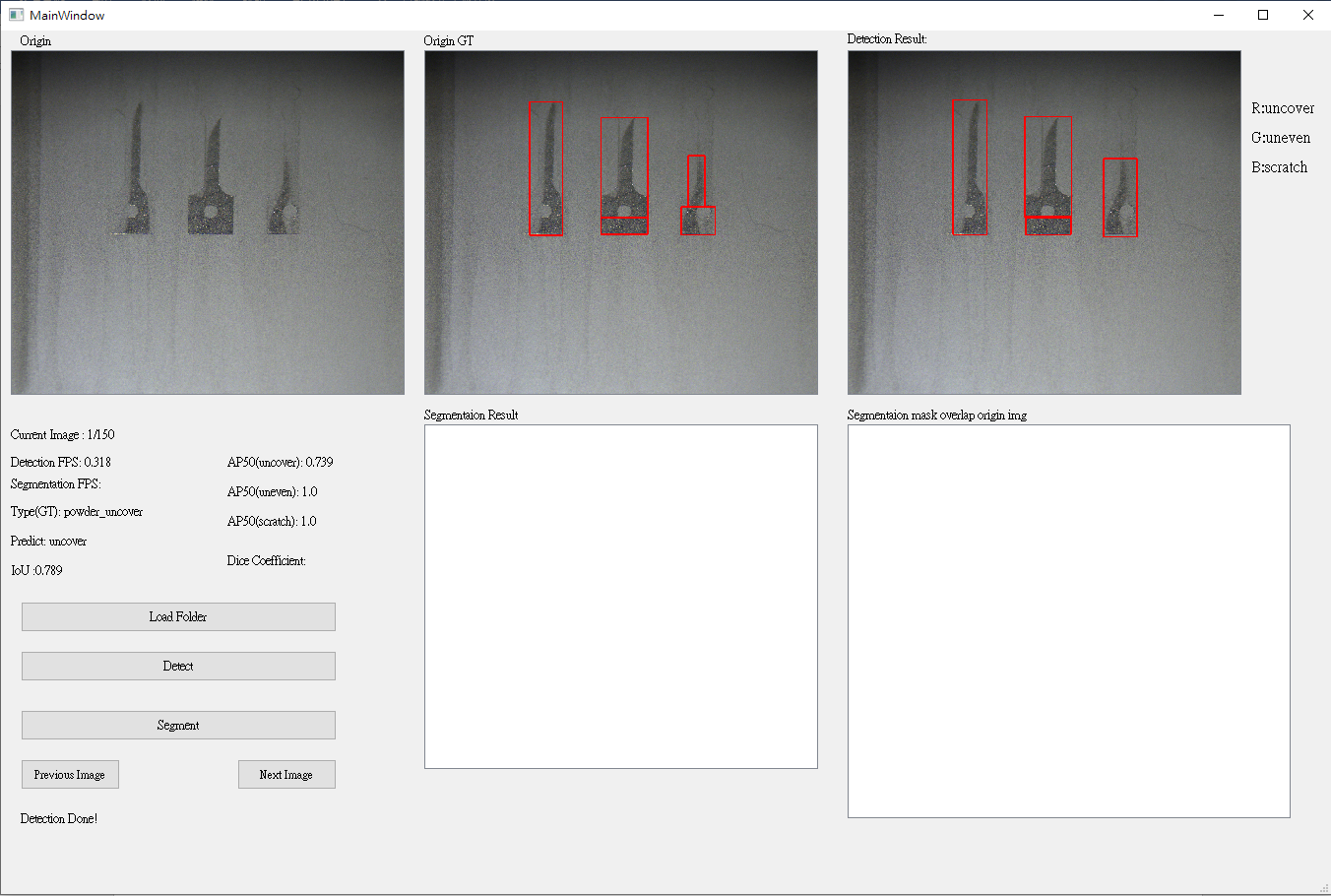
**Initial window**(after push Load Folder button)

****

**After push detection button**

**Origin GT**: show image that with given ground truth in Json

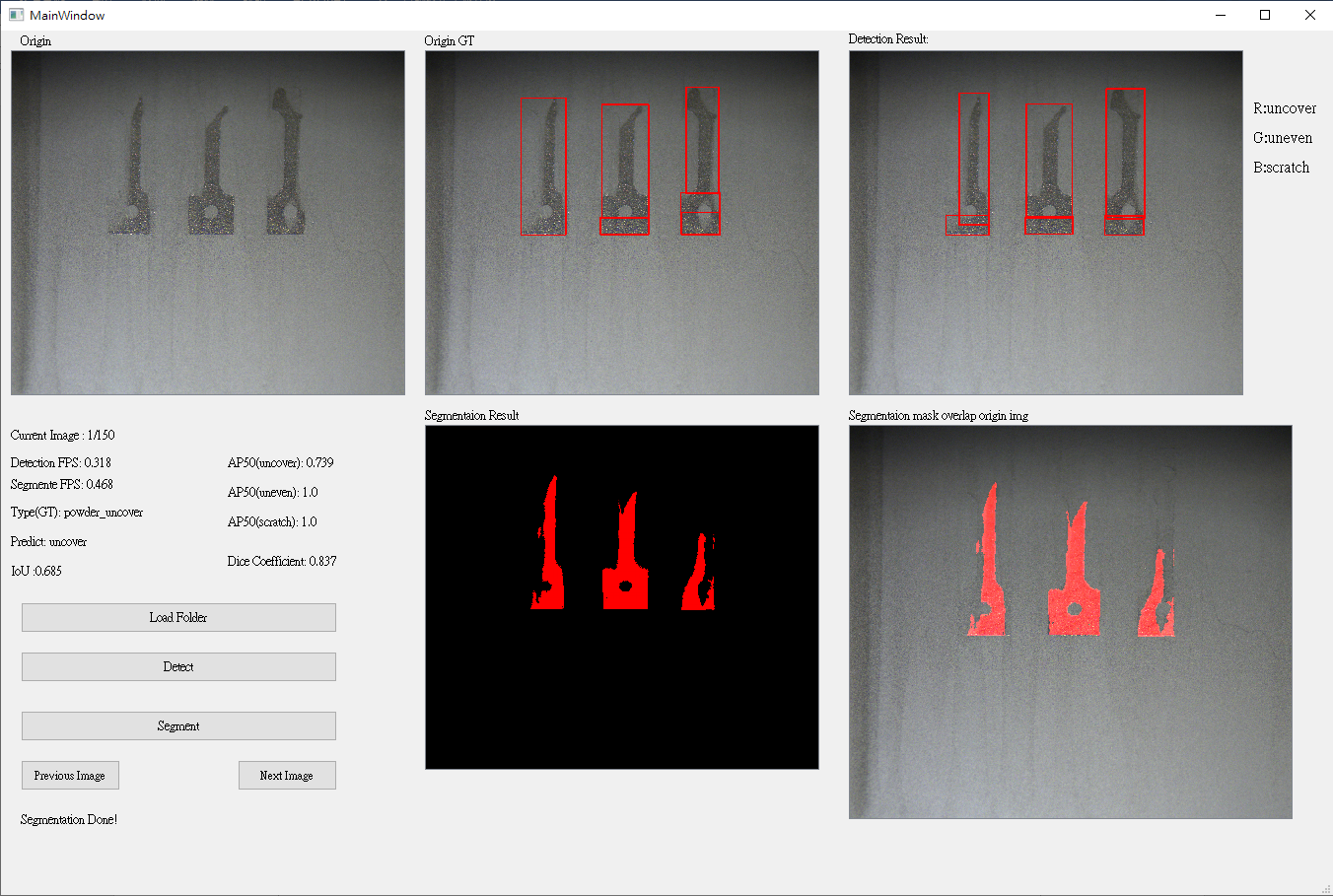
**Detection Result**: show image that with yolov5 detect bounding box

****

**After push segment button**

**Segmentation mask:** show segment result with black background.

**Segmentation mask overlap original img:** show segment result overlap with original image

****

而Previous Image 和 Next Image button 可以選擇前後一張圖片顯示

***Discussion:***

1. 在segmentation時，下圖uneven的case會與uncover的較不相同，因此會先將其進行not運算再進入U-Net。我推測可能對於uncover是要將暗色的地方mask設255，但是以下這個case卻是亮的地方mask應該要設為255，若是不先not，則會導致segment錯地方。



Fig 1.original



Fig2.先進行not再detect



Fig3. 直接detect

1. 對於大部分uneven (如下圖)以及scratch，觀察mask後，發現似乎將整個Detect的box塗滿即可，因此這些case並不會給U-Net訓練或預測，在predict時，會在yolov5框出框後，通過長寬及class判斷出後，直接將整個Detect的box填對應的class顏色即可。另一方面，若是將這些圖片送進U-Net，對於訓練(uncover)會產生反效果。



1. 對於一些小的瑕疵，由於會先裁剪下來再resize成300\*400並送進U-Net，一開始我擔心小圖片若放大會造成訓練或辨識困難，所以對於小圖片，會將圖片置於中間，再將周圍padding圖片的gray level mean至150\*200，再放大成300\*400，以避面小圖片被放大太多，但最後我發先就算padding mean可能還是會對segment有些影響，效果不如沒有來的好，所以最後就還是採取統一放大或所小的方法。
2. 另外，不知道是否是給定的ground truth的關係，在給定的ground truth中，有時候會將一個defect用很多框框起來，有時候卻只用一個，這樣感覺yolo也不知道甚麼時候要全部框一起還是要分開框(一些聚集的uncover defects)，所以大部分yolo的detect與ground truth的數量會不一樣，這樣IOU的計算就不是很有代表性，有時候IOU會由於yolo多框而導致很低，但是整體segment效果仍會很好，因此我認為在這個project下，感覺Dice coefficient會比較能代表圖片的效果指數。
3. 由於有些剪裁過後的圖片會有點模糊，為了使結果更好，我參考了網路上的資料，使用cv2.GaussianBlur和cv2.addWeighted，使圖片銳利化，效果不是非常明顯，但是由於運算不多，所以還是保留下來。

***Conclusion:***

這次的功課比較不一樣的是需要銜接yolov5及U-Net兩個不同的部分，yolov5產生的input需要經過處理才能傳進去U-Net，而這中間處理的方法會大幅影響最終segmentation的結果，另外，要將圖片能準確且快速地顯示在UI也是需要另外思考，一些metric計算也可以通過一些公式或技巧來快速取得，因此在此功課的指標計算與顯示圖片上，也需考慮上整體的效率，最後，兩者的銜接更需要大量的cv2的處理，因此對於一些cv2的圖片處理技巧，也更加的熟悉了。