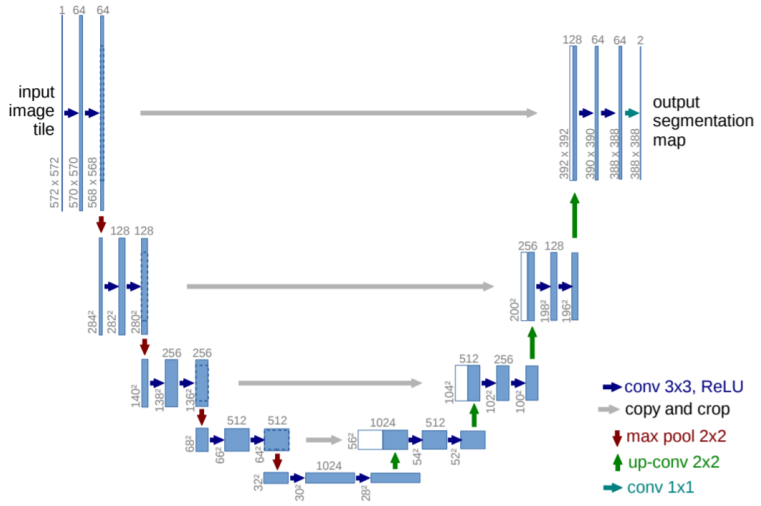
I. Question

從脊椎 X 光片中自動地將脊椎骨切割出來。

II. Method

3.1. Residual U-Net

以 U-Net 為基礎進行改良設計，在每層做第一次 conv 3x3 前用 shortcut 將輸入連到第二次 conv 3x3 的輸出(圖左)，採 conv 1x1 對輸入 feature map 做降維，使其可與通過兩次 conv 3x3 的 feature map 接合(圖右)，如此一來學習過程會自動判斷是否要將輸入的 feature map 直接通過亦或是進行兩次卷積提取特徵。



(圖左)

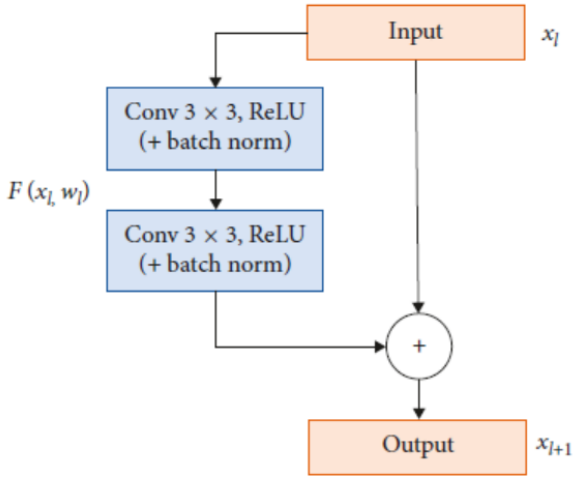


FIGURE 12: Residual block [28].

(圖右, ref. [1])

3.2. Loss Function

考量的損失函數定義為 L = BCE + λ × DL，λ設定為 0.001，使模型在學習過程中也考量目前整體的 DC 值(越大越好)，作為 Regular 項，其各項分別如下，

Binary Cross Entropy

$$\ell(x,y)=BCE=\{l_1,\ldots,l_N\}^\top,\ l_n=-w_n[y_n\cdot\log\sigma(x_n)+(1-y_n)\cdot\log(1-\sigma(x_n))],$$

Dice loss

$$\mathrm{DL}\left(p,\hat{p}\right)=1-\frac{2p\hat{p}+1}{p+\hat{p}+1}\quad \text{where } p\in\{0,1\} \text{ and } 0\leq\hat{p}\leq 1.$$

3.3. Data Augmentation

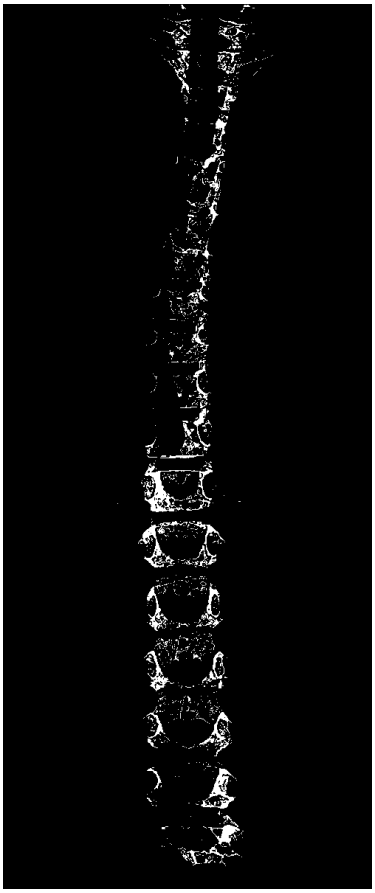
為解決資料不足的問題，學習過程對資料進行隨機的 affine 變換、水平翻轉及亮度、對比度、飽和度與色調調整，隨機的 affine 可增加不同角度脊椎骨的資料，而隨機影像屬性調整可增加 X 光片影像品質不一致時的資料。

3.4. Pre-train Model

使模型的權重在訓練前有一個好的初始值，故先提取訓練集資料的特徵作為初始訓練集，(圖左)為進行 3x3 Sobel 濾波後的結果，(圖中)為原圖減去(圖左)的結果，(圖右)為提取脊椎的結果，方法參考[1]。

(圖左)

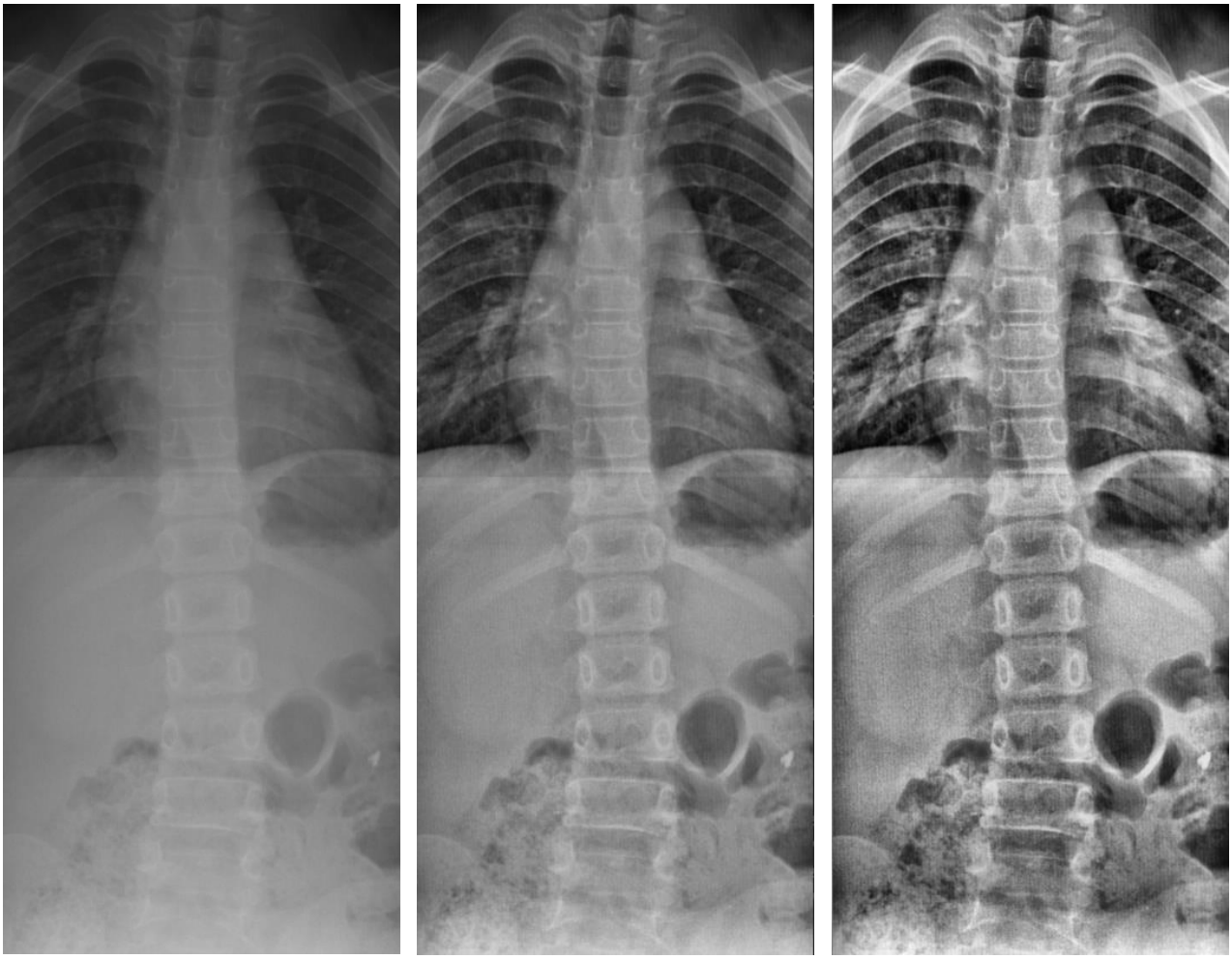
(圖中)



(圖右)

3.5. Image Preprocessing & Training set

脊椎骨在腹部及心臟位置具有相似的灰階值，採用”限制對比度的自適應直方圖均衡(CLAHE)”方法，可藉由提高影像區域性的對比度來區隔前、後景，改善影像品質，使模型更易於訓練，故採用執行一次的 CLAHE 結果(圖中)當作訓練集。



(原圖)　　(一次 CLAHE)　　(二次 CLAHE)

3.6. Ensemble Learning

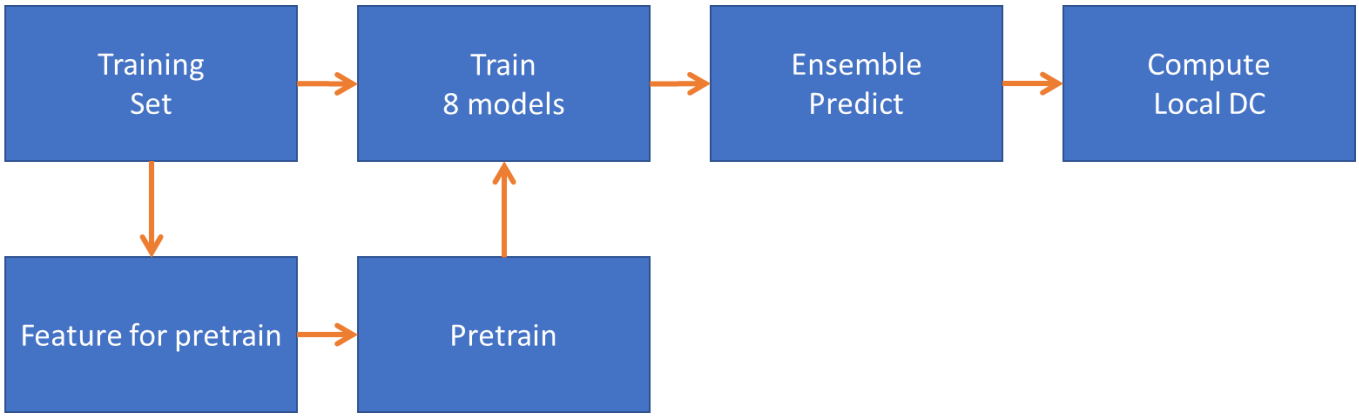
由於訓練過程用資料增強來隨機生成資料集，故不能保證模型學到的資料品質能與未知的測資相似，因此同時訓練多個模型來應付各種不同訓練集的情況，準備了四個有 pretrain 及四個沒有 pretrain 的模型，輸出的結果取平均後，高於一個閥值(通常設 0.5)的 pixel 視為脊椎骨。

3.7. Parameter Setting

Epochs=512
Batch size=4
Down sample scale=0.2
Learning Rate=0.001
Learning Rate Decay= LR * (0.8 ** (epoch // 128))

```
train_net(net,
          train_dataset,
          device,
          epochs=512,
          batch_size=4,
          lr=0.001,
          val_percent=0,
          save_cp=False,
          img_scale=img_scale,
          data_augment=True)
```

3.8. Flow-Chart



3.9. Model

```
self.inc = ResidualDoubleConv(n_channels, 32)
self.down1 = Down(32, 64)
self.down2 = Down(64, 128)
self.down3 = Down(128, 256)
self.down4 = Down(256, 512)
self.down5 = Down(512, 512)
self.up1 = Up(1024, 512, bilinear)
self.up2 = Up(768, 256, bilinear)
self.up3 = Up(384, 128, bilinear)
self.up4 = Up(192, 64, bilinear)
self.up5 = Up(96, 32, bilinear)
self.outc = OutConv(32, n_classes)
```

III. Result

4.1. 3-Fold Cross-Validation

```
train_net(net,
          train_dataset[i],
          device,
          epochs=256,
          batch_size=4,
          lr=0.001,
          val_percent=0,
          save_cp=False,
          img_scale=0.2,
          data_augment=False)
```

	U-Net	Residual U-Net
K=1	0.9026980608701706	0.8928840607404709
K=2	0.9251346111297607	0.926760533452034
K=3	0.9173149526119232	0.9216688871383667
Average	0.9150492082039515	0.9137711604436238
Parameters	13,394,177	20,878,753

4.2. Evaluation

4.2.1. Global Dice Coefficient

$$\text{dice similarity coefficient (DSC)} = \frac{2|GT \cap SR|}{|GT| + |SR|},$$

```
class DiceCoeff(Function):
    """Dice coeff for individual examples"""

    def forward(self, input, target):
        self.save_for_backward(input, target)
        eps = 0.0001
        self.inter = torch.dot(input.view(-1), target.view(-1))
        self.union = torch.sum(input) + torch.sum(target) + eps

        t = (2 * self.inter.float() + eps) / self.union.float()
        return t
```

4.2.2. Local Dice Coefficient

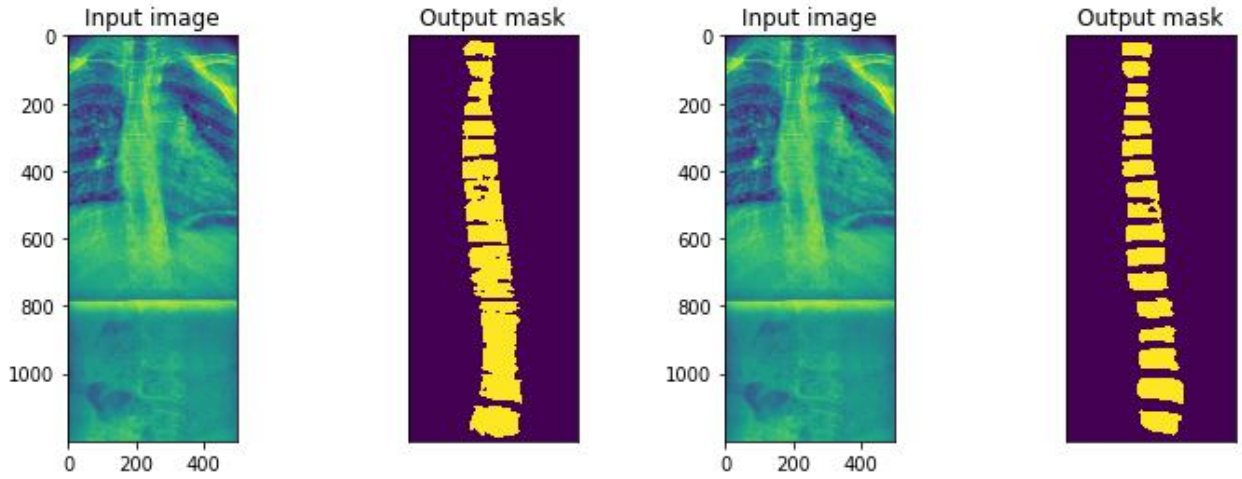
使用 opencv 的 Find Contours 找出 Predict 的每個脊椎骨並求出各自的面積，若 Predict 的脊椎數目與 Ground Truth 相同，則逐一取出計算 DC，交集面積= (predict + ground truth) - 聯集，再用 Global Dice 的算法計算出單脊椎骨的 DC。

```
# compute the Area
area_p = cv2.contourArea(contours_p[i])
area_g = cv2.contourArea(contours_g[i])

# mapping to new space
temp1 = np.zeros((1200, 500, 3), np.uint8)
temp1.fill(0)
temp2 = np.zeros((1200, 500, 3), np.uint8)
temp2.fill(0)
cv2.drawContours(temp1, contours_p, i, (0, 255, 0), 1)
cv2.drawContours(temp2, contours_g, i, (0, 0, 255), 1)

# get the union area
union = np.zeros((1200, 500, 3), np.uint8)
union.fill(0)
union = cv2.add(temp1, temp2)
union_gray = cv2.cvtColor(union, cv2.COLOR_BGR2GRAY)
_, contours_u, hierarchy_u = cv2.findContours(union_gray, cv2.RETR_TREE,\
                                              cv2.CHAIN_APPROX_SIMPLE)
area_u = cv2.contourArea(contours_u[0])

# compute the dc score
cross = area_g + area_p - area_u
total = 2 * (cross) / (area_g + area_p)
```

4.2.3. Ablation Study

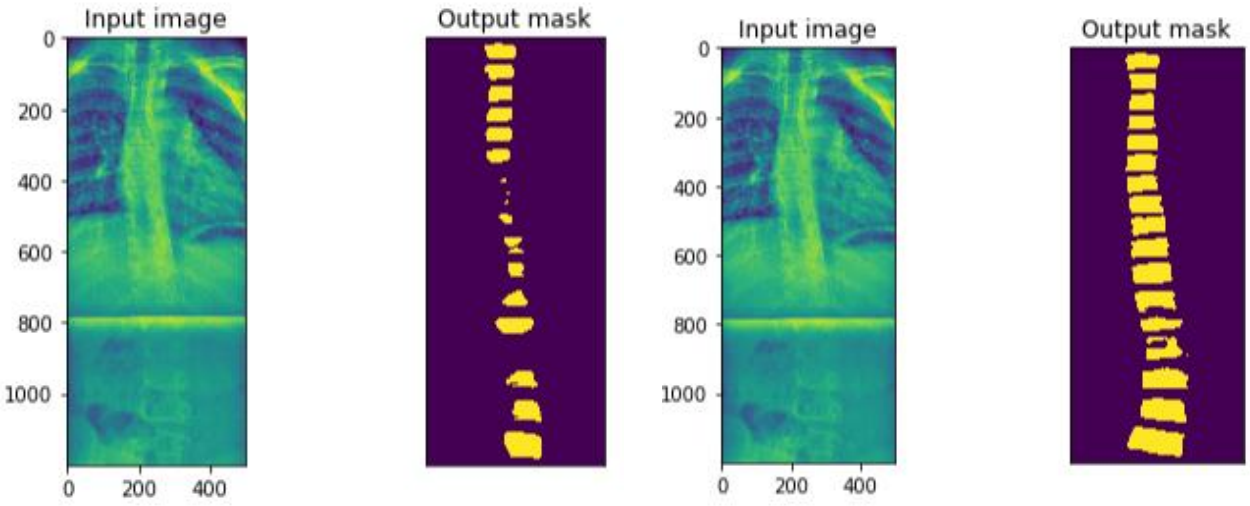
固定其他參數情況下做調整功能塊同時觀察其變化，而這裡不去計算 DC，並希望說預測的結果儘量是沒有破碎與相連的情況，如此才能穩定的計算出 Local DC，以下皆是 validation set 的結果(非進行訓練的資料)。

No Data Aug v.s. Data Aug



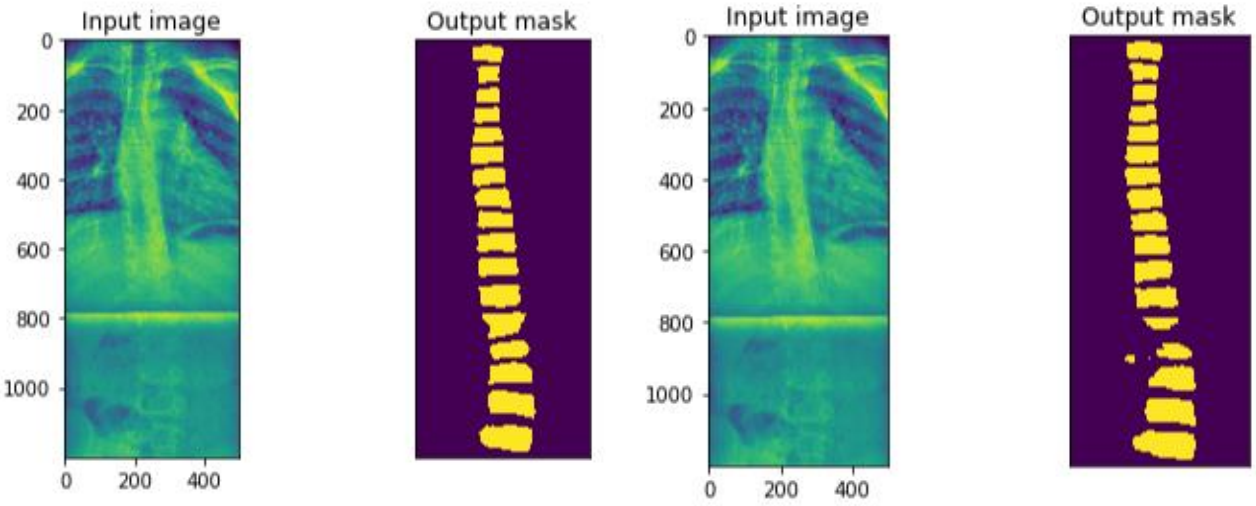
若無進行資料增強時，遇到影像品質不佳的 X 光片，模型較難成功分割脊椎骨。

No Pretrain v.s. With Pretrain



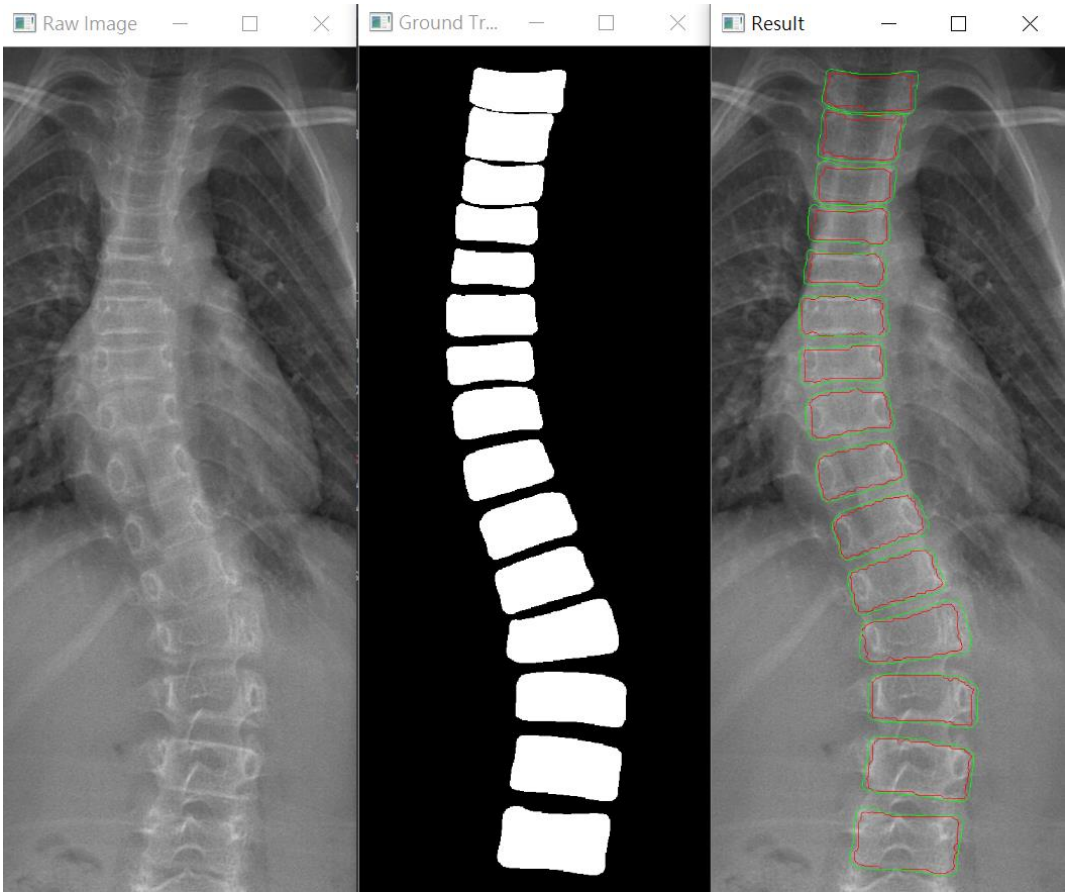
由於 Pretrain 會預先學習脊椎骨特徵，所以在相同 epoch 下可以收斂地更快，Predict 的結果也較好。

Scale 0.2 v.s. Scale 0.5

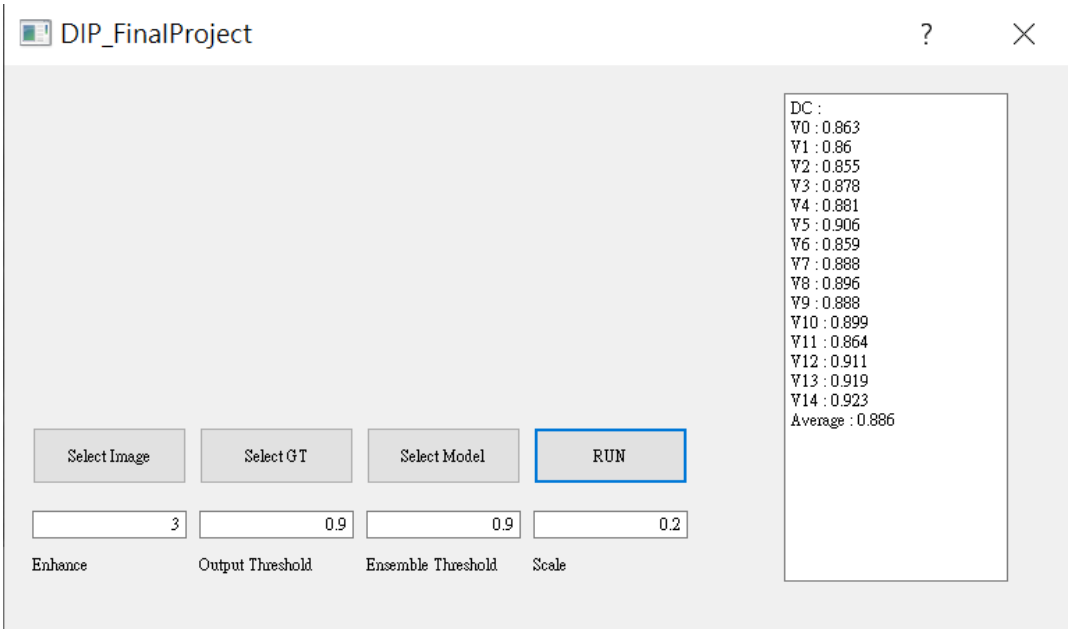


Scale 設 0.2 的效果較好之可能原因是能讓模型學到用簡單的特徵來分割出脊椎骨，而 Scale 0.5 將原圖 down sample 到一半仍可能保留較多複雜的特徵，不易模型學習，且參數大導致訓練時收斂的時間較久，因此採用 Scale 0.2 來當作訓練參數。

4.3. Show



(原圖)　　(Ground Truth)　　(結果，紅線為 predict)



(GUI)

IV. Discussion

先前有嘗試過加深網路以及提高尺寸約 50x100 的濾波器個數，原因是想說增加提取更細節的濾波器所以加深網路，而 50x100 尺寸大約是脊椎骨的尺寸，增加其濾波器個數期望可學習到更多不同面向的脊椎骨，以上設計皆期望可提高準確度，但最後兩者效果並不明顯，可能原因是採用 CLAHE 已可顯著地改善影像的品質及脊椎特徵，故用參數較少的 UNET 也可學習的很好，進行預測時，若有無法準確切割的情形，此時對圖片再執行多次的 CLAHE 則可以使模型切割脊椎骨出來，所以用 CLAHE 除了可幫助學習以外，也可提高預測準確率，加上有 pretrain 的話可加快學習的收斂速度，而採用資料增強可明顯提高切割的成果。

V. Conclusion

1. 用 CLAHE 可顯著地改善訓練與預測的成效

2. 資料增強可以彌補影像拍攝時造成品質不一致的情況

3. 加深 UNET 網路對此任務無太大的幫助

4. 用特徵當作 pretrain 可加速訓練收斂速度

5. Ensemble 可更客觀地預測結果

Reference:

[1] Cobb Angle Measurement of Spine from X-Ray Images Using Convolutional Neural Network, Ming-Huwi Horng, Chan-Pang Kuok, Min-Jun Fu, Chii-Jen Lin, and Yung-Nien Sun