1. Overview

在本次的報告實驗中，第二章節解釋了兩個網路的架構，UNet 的設計方式和 Residual learning 的概念，第三章則是說明在 UNet 論文中所提到的資料擴展，但是並未實作在本實驗中，第四章是比較兩個網路的實驗數據，第五章是訓練時所用的指令，第六章則是探討兩個架構的差異。

2. Implementation Details

Details of training, evaluating, inferencing code

在 training 時，本實驗報告寫了一個主要的 function 去執行訓練每一個 epoch 的動作，會事先載入 oxford_pet 資料集，並分成 9:1 的訓練(train)資料集跟驗證(valid)資料集，使用 torch 的 DataLoader 套件對資料分成不同的 batch size，在讀取時可以拿到 batch 數量的不同 image，做一次訓練(training ,optimization ,backpropagation)。

在 training 時每次的 epoch 都會有一個平均的 train loss 和 valid loss，用來評估 model 在訓練時是否有 overfitting 或 underfitting 的情況。

在 evaluate 的過程中也會顯示透過 Dice similarity Coefficient[1]所算出來的 Dice Score，用來判斷訓練時的效果和在測試集上的表現狀況，Dice Score 的數值範圍介於 0 到 1 之間，其算法為：

$$Dice\ score = \frac{2 * (number\ of\ common\ elements)}{(number\ of\ elements\ in\ set\ A + number\ of\ elements\ in\ set\ B)}$$

在 inference 時大致上和 evaluate 的方式差不多，同樣會求出在測試集上$mask_{pred}$和$mask_{ground-truth}$的 loss 與 Dice Score，不一樣的點在於 inference 提供了顯示 mask 的功能，在使用 inference.py 時可以指定是否要顯示每個 batch 時每個圖片的 mask 效果，透過 PyTorch 所提供的 draw_segmentation_masks[2]可以很方便地達到幫圖片畫上 segmentation 的方式。

Details of UNet model

UNet 是一個 Encoder-Decoder 的架構[3]，在圖.1 中為 UNet 論文中所提到的架構[4]，分為 Encoder 與 Decoder 部分，在 Python 的實作上則是以

圖.2 所示，分成很多個小區塊。本實驗的實作都是以同樣大小，也就是維持相同 padding 的情況下，換句話說就是輸入的圖片與輸出大小是一致的，如果一個圖片大小為 572 x 572 進入 UNet 之後會變成 388 x 388，可能會造成 50%左右的大小損失[3]。在 UNet 的架構中，左側會先對圖片做更高為度的特徵提取，在 CNN 層以更多的 channel 去提取特徵，到最後會有 1024 個 channel 開始做 up-conv，將高維度的特徵 Decode 成原本圖形大小的資料，在圖.1 的架構中有看到會有一條 copy and crop 的 path，每次在 up-conv 時都會與先前資料做結合，保留了原先在 encode 時的部分特徵(高維度特徵與低維度特徵的結合)，這樣做不只可以提升在 segmentation 時的精確度，也能避免原先在 decode 時的資料損失問題，有點類似 ResNet 中的 skip connection[5]，能做到特徵融合與保留原先資料細節的功能。

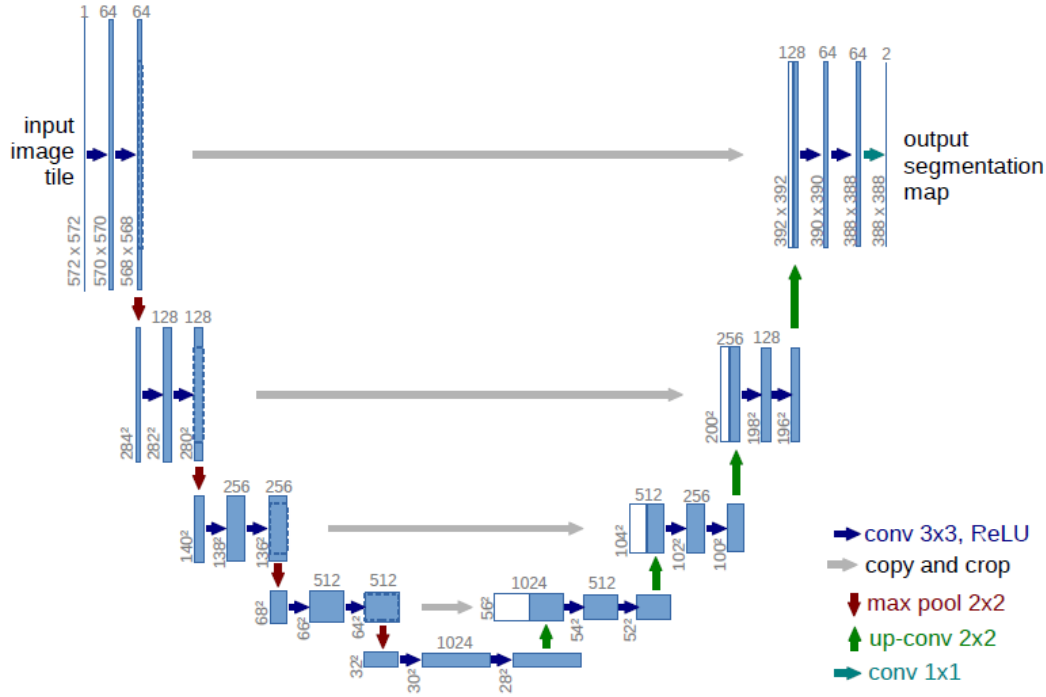


圖.1、UNet 架構

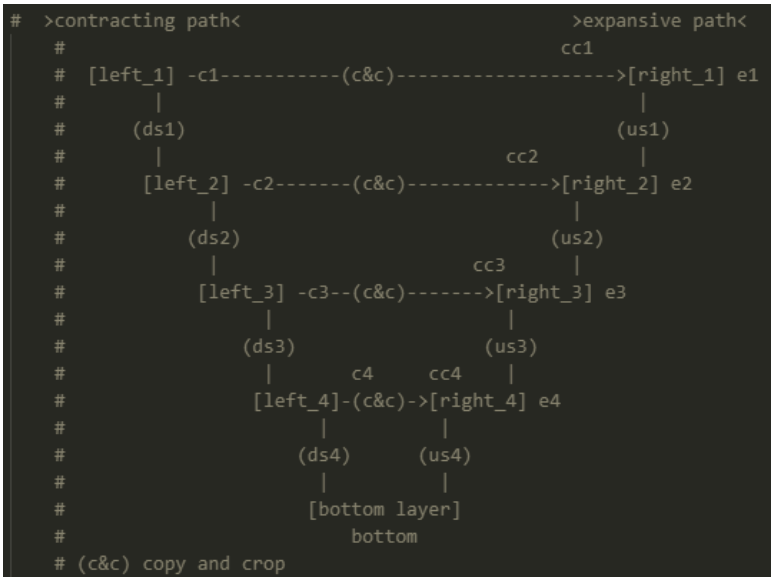


圖.2、實驗中設計的 UNet 架構

在實作上以圖.2 的方式去設計網路，整體的 input channel 設定是三層(R、G、B)，輸出則是設定為一層(由於本實驗為 binary semantic segmentation)著重在把 segmentation 框出來的問題上。此外由於問題設定在單純的 object segmentation，並沒有針對物件去做區分類別的動作，因此在 loss function 的取用上是使用二元交叉商 BCEWithLogitsLoss，在只有單個類別(binary semantic segmentation)的使用上較為常見[3, 6, 7]。

Details of ResNet34_UNet model

ResNet34_UNet 在 encode 階段使用 ResNet34[5]的架構，後半段的 decode 則是使用 UNet 原本的架構，跟 UNet 一樣在 encode 時為了結合高為度的特徵值和避免資料損失的情況，同樣有一條類似 skip connection 的資料跟 up-sample 時的資料做結合[8]。

ResNet34 的基本架構是使用多層的 Residual Block，Residual Block 可以很好的避免梯度消失、爆炸的情況發生[5]，圖.3 為一個兩層 Conv 組成的 Residual Block，從圖.3 中可以看到跟一般的神經網路不太一樣，假設$H(x)$是一個目標函數，示我們預期希望達到的一個 residual block 能學習到的結果，可以表示為 $H(x) := F(x) + x$，而在透過訓練時其實是學習從中的$F(x)$，論文中稱作 residual function，$F(x) := H(x) - x$。舉一個簡單的例子，假設預期的輸出是 30，$H(x) = 30$，但是現在的值是 10，$x = 10$，也就是說要修正的值就是$F(x) := 30 - 10 = 20$。Residual learning 學的就變成是$F(x)$，有效的避免了，像是在很深層的網路更新時，會造層的梯度消失/梯度爆炸問題。如果一層網路不更新的話可以對內部的$F(x)$不去做任何

的修正，也就是$H(x) = x$，內部的權重網路並沒有需要被訓練到。

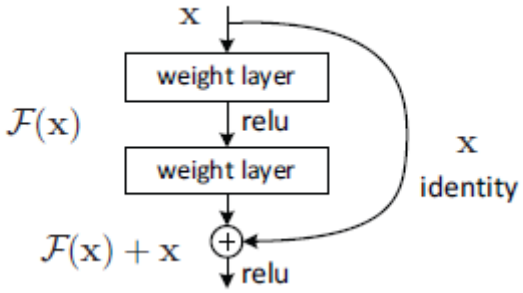


圖.3、Residual Block，原文中表示為 Residual learning 表示為$F(x)$

ResNet34_Unet 的結構如圖.4 所示，結合了 UNet 架構和加入了 Residual learning 的特點，能更有效的去學習，避免了梯度消失等問題，也應用在 segmentation 的問題上，甚至可以使用在多類別(multi-classes)，或者更深層的 Encode-Decode 架構上。

在實作上架構跟 UNet 的實作上很像，同樣是以一個類似 U 型的網路架構去做設計，圖.5 表示了實作 ResNet34_UNet 的網路架構，因為 ResNet34 在轉到 512 channel 時會經過 5 次的 downsampling，因此為了保持輸入輸出圖片大小一致，在 decode 時也會需要做 5 次的 upsample(up-conv)，圖.4 中的網路架構設計如表.1 所示。

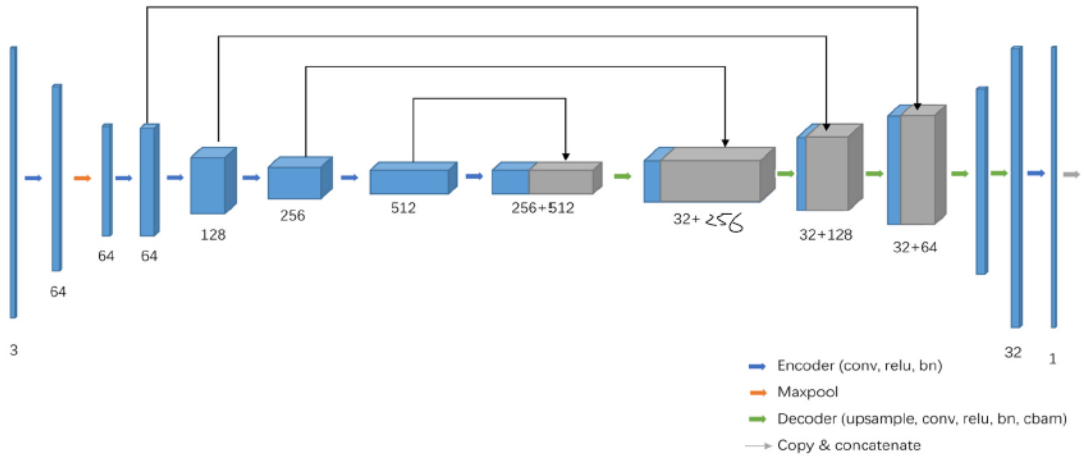


圖.4、ResNet34_UNet 架構

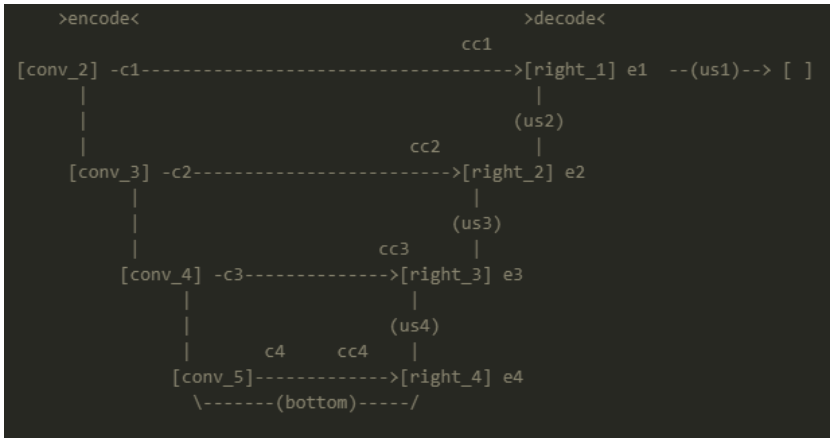


圖.5、網路實作上的 ResNet34_UNet

3. Data Preprocessing

Preprocessing data

在實作上並沒有將 Data Preprocessing 實作出來，但是針對 UNet 的 paper 中有提到可以針對特定圖片做 transform，去產生新的資料(data augmentation)[4, 9]，UNet 的 paper 中提到先在一個 3x3 的網格上產生隨機的位移值，這些位移來自標準差為 10 像素的高斯分布。通過雙三次插值(bicubic interpolation)來算每個像素的變形量(算出具體的位移量)，進行生成變形的圖像，同時也有在 contracting path (encoder 時) 做 dropout layer 來做另類的數據擴增(data augmentation)，但是這種情況只能在少量樣本時，提升模型的性能和穩定性。

此外也有嘗試使用在載入前先 resize 圖片，試著去提升一次 training 時能接受的 batch_size 數量，但是測試下來的結果是事先 resize 並不會影響最終進到 model 裡面的計算量。

4. Analyze on the experiment results

What did you explore during the training process

訓練時所用的超參數有，Batch size 為 16、Optimizer 為 AdamW、learning rate 為 0.0005，圖.6 和圖.7 中分別是 ResNet34_UNet 和 UNet 的訓練過程。在 Loss 的比較圖上，ResNet34_UNet 跟 UNet 差異看起來並不大，反倒是圖.8 的 dice score 比較圖更能看得出差異，ResNet34_UNet 在訓練時拿到的效果相對來說都是比 UNet 還要好一點的。

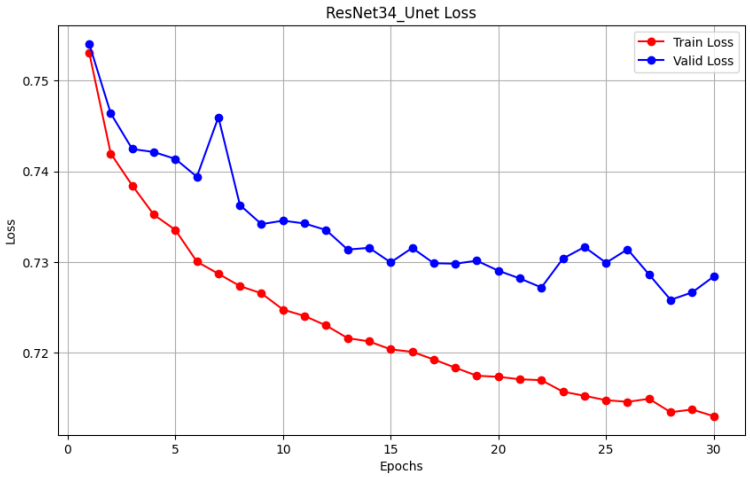


圖.6、ResNet34_UNe 訓練結果圖

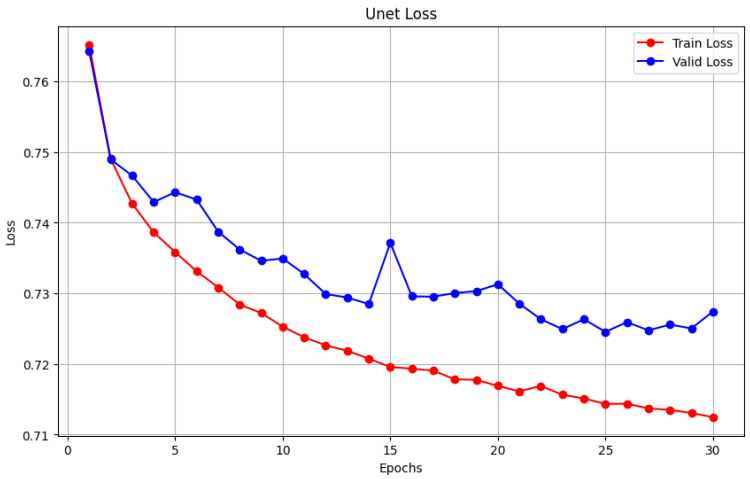


圖.7、UNet 的訓練結果圖

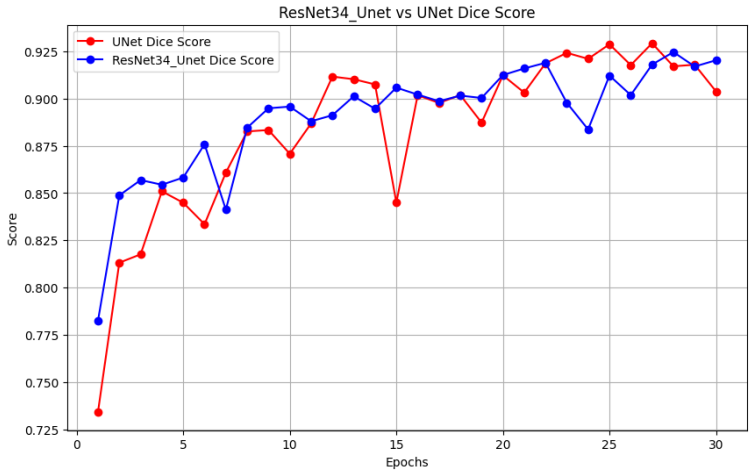
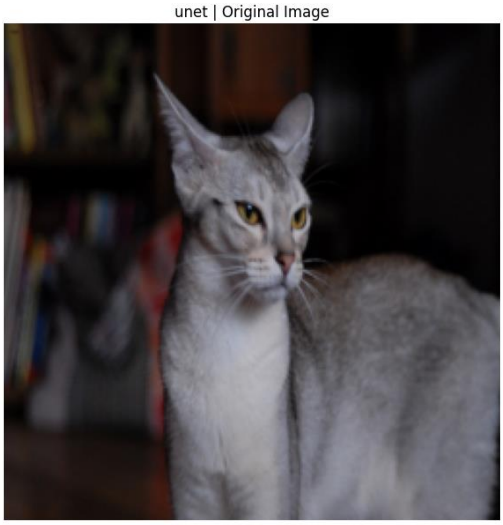
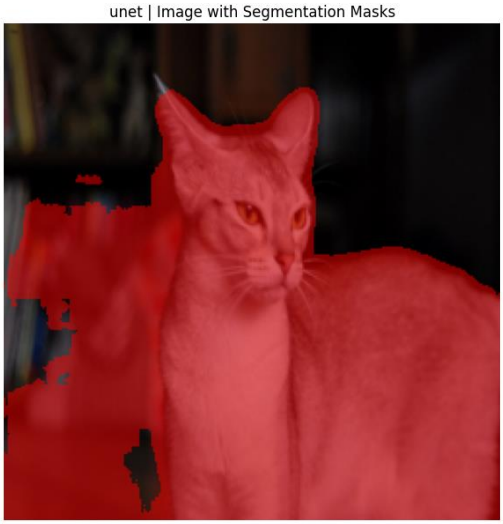


圖.8、兩個網路的 Dice Score 比較圖

Found any characteristics of the data

圖.9 和圖.10 中表示了不同圖片在兩個架構中的推論(inference)結果，並且將推論出來的結果顯示出來。從 UNet 的圖中可以看到比起 ResNet34_UNet 的結果，更容易誤判區域，但是 ResNet34_UNet 的結果相對較好。

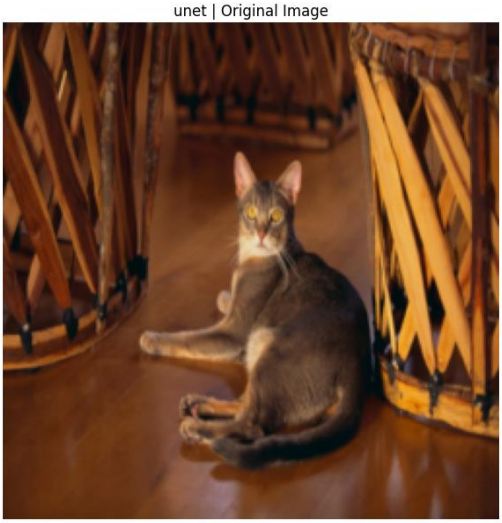




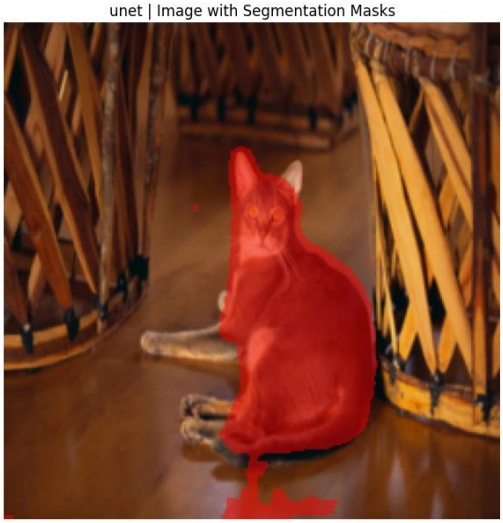
unet | Original Image

unet | Image with Segmentation Masks

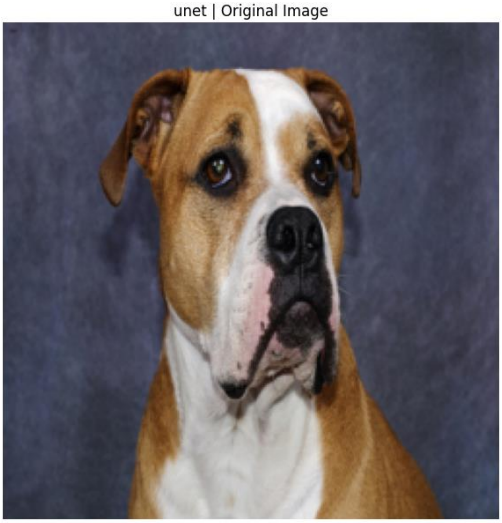
unet | Original Image



unet | Image with Segmentation Masks



unet | Original Image



unet | Image with Segmentation Masks

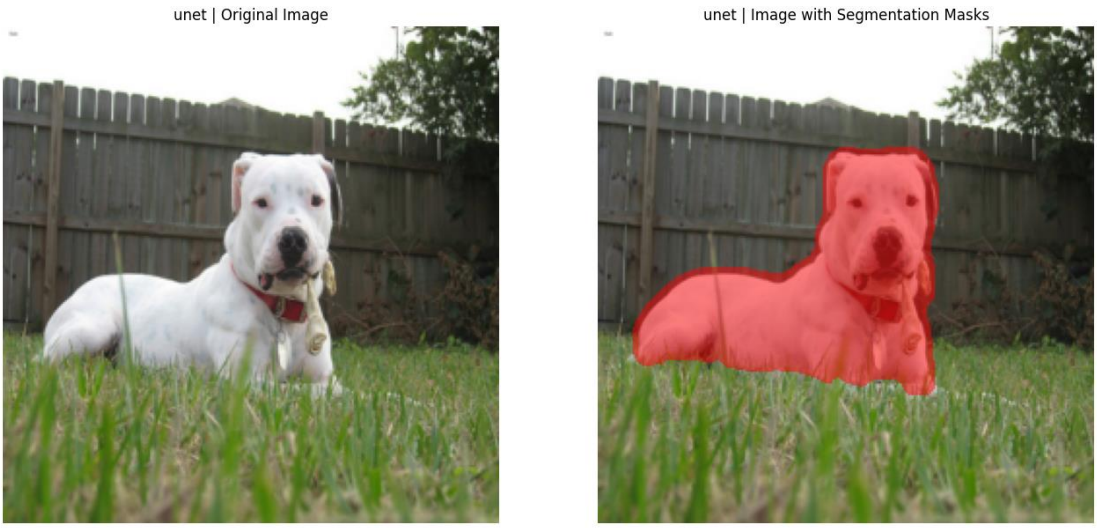
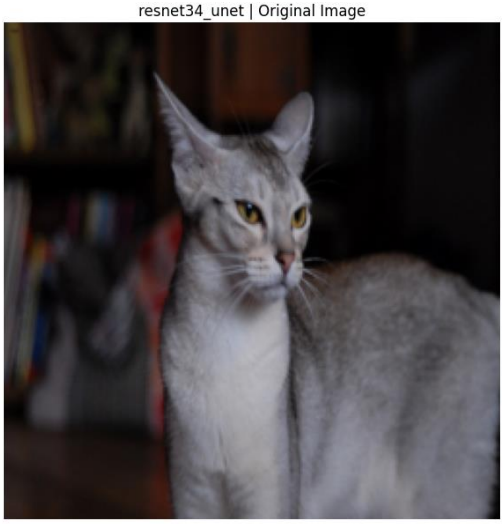
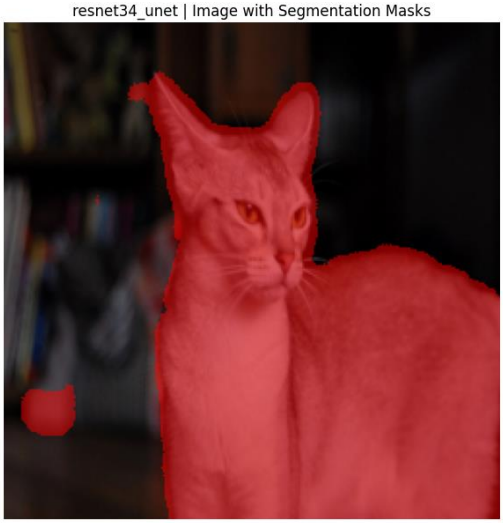


圖.9、UNet 在推論時的結果

resnet34_unet | Original Image



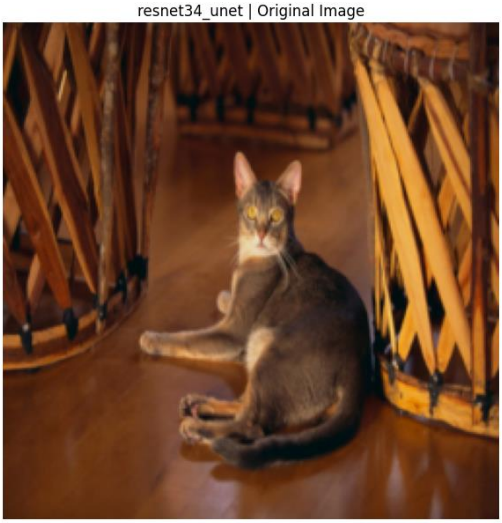
resnet34_unet | Image with Segmentation Masks



resnet34_unet | Original Image

resnet34_unet | Image with Segmentation Masks

resnet34_unet | Original Image



resnet34_unet | Image with Segmentation Masks

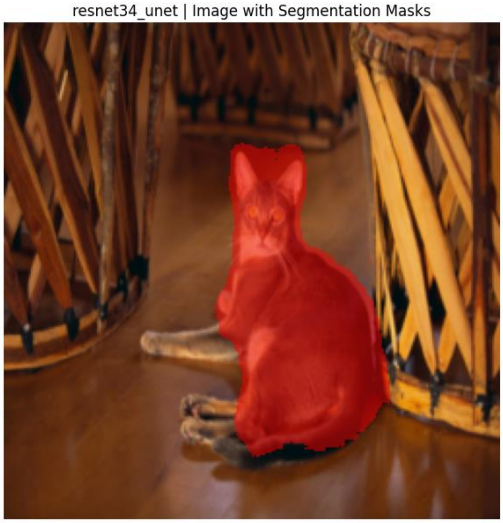


圖.10、ResNet34_UNet 在推論時的結果

5. Execution command

The command and parameters for the training process

在訓練時可以使用此指令

```
python train.py
```

能使用的參數有

- --data_path ，資料集的位置
- --model ， 要載入的 pretrain model
- --batch_size ，一次訓練的 batch 大小
- --model_name ，是使用 unet 還是 resnet34_unet (存檔時會以這個當作前輟)
- --learning_rate ，學習率
- --epochs ，訓練的次數

The command and parameters for the inference process

在訓練時可以使用此指令

```
python inference.py
```

能使用的參數有

- --data_path ，資料集的位置
- --model ， 要載入的 pretrain model
- --batch_size ，一次訓練的 batch 大小
- --model_name ，是使用 unet 還是 resnet34_unet
- --show_inference ，使否要啟用顯示圖片的推論結果(會依照 batch_size 的大小一次顯示多張圖片)

6. Discussion

What architecture may bring better results

ResNet34_UNet 的架構以實驗數據上來看比 UNet 要來的好，如果以架構來看的話，ResNet34_UNet 在訓練圖片時應該要能達到更好的效果，如第二章所提到的，使用了 Residual learning 的方式，可以支援更深層的網路，擷取更多、更高為度的特徵點。

What are the potential research topics in this task

我個人認為可以針對多個 classes 的情況，因為目前只是針對 binary semantic segmentation，在 ResNet34_UNet 應該能在 mutli-classes 時比 UNet 本身的架構還要來的好，甚至是使用更深層的 ResNet，如 ResNet101 或 ResNet152 等當作這個架構的 Encoder，嘗試去找出更多每個像素上的特徵點。

7. Reference

1. OECD.AI. *Dice score*. Available from: https://oecd.ai/en/catalogue/metrics/dice-score.
2. PyTorch. *draw_segmentation_masks*. Available from: https://pytorch.org/vision/stable/generated/torchvision.utils.draw_segmentation_masks.html.
3. Tran, M. *Understanding U-Net*. Nov 15, 2022; Available from: https://towardsdatascience.com/understanding-u-net-61276b10f360.
4. Ronneberger, O., P. Fischer, and T. Brox. *U-net: Convolutional networks for biomedical image segmentation*. in *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*. 2015. Springer.
5. He, K., et al. *Deep residual learning for image recognition*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.
6. pzq666666. *图像分割评判系数整理（基础）*. 2023-04-17; Available from: https://blog.csdn.net/weixin_57923537/article/details/130209335.
7. Espressius, D. *3 Common Loss Functions for Image Segmentation*. Jan 30, 2022; Available from: https://dev.to/_aadidev/3-common-loss-functions-for-image-segmentation-545o.
8. Huang, Z., et al., *Deep learning-based pelvic levator hiatus segmentation from ultrasound images.* 2022. **9**: p. 100412.
9. disapear1997. *[day-24] U-net Data Augmentation*. 2020-10-09; Available from: https://ithelp.ithome.com.tw/articles/10250219.