

1.Implementation Details

A. Unet 架構模型

本次實作先建構了一個元件 CC (2 次 conv)，有別於原始 Unet 的捲積單位，本次實作對於 2 次的捲積中，將 channel 的數量做漸進式遞增或遞減，避免過程中突然大量流失資訊，另外本次實作了參數 $\text{feature}=[64, 128, 256]$ 做為 Unet 整體架構中每一層的通道數量，並可以透過增加 feature 的數量，來增加 Unet 模型的深度。

B. Resnet34_Unet

本次實作延續 CC 元件，額外增加 Resnet34 架構中的殘差捲積元件 BasicBlock，在 encoder 中所有架構和 channel 數量皆和 Resnet34 相同，decoder 和 skip 的操作參考 Lab2 敘述中的圖片架構。

C. 訓練過程

loss 函數採用 dice loss，希望 training 和 validation 所要達成的目標一致。optimizer 使用 Adam，並使用 scheduler 調控 learning rate，也有做 early stop 機制。在執行 train.py 時，可以有 --model 參數，此參數為 0 時訓練 Unet 為 1 時訓練 Resnet34_Unet

D. utils

a. EarlyStopping:

參數 patience，當 validation loss 超過固定 epochs 沒有改善，就中斷訓練。

參數 min_delta，作為 validation loss 最小需改善的幅度。

參數 path，模型暫存路徑。

b. dice_score:

epsilon 為 0.1，避免除以 0，可支援多筆資料，返還 dice score 的平均值。

c. dice_loss:

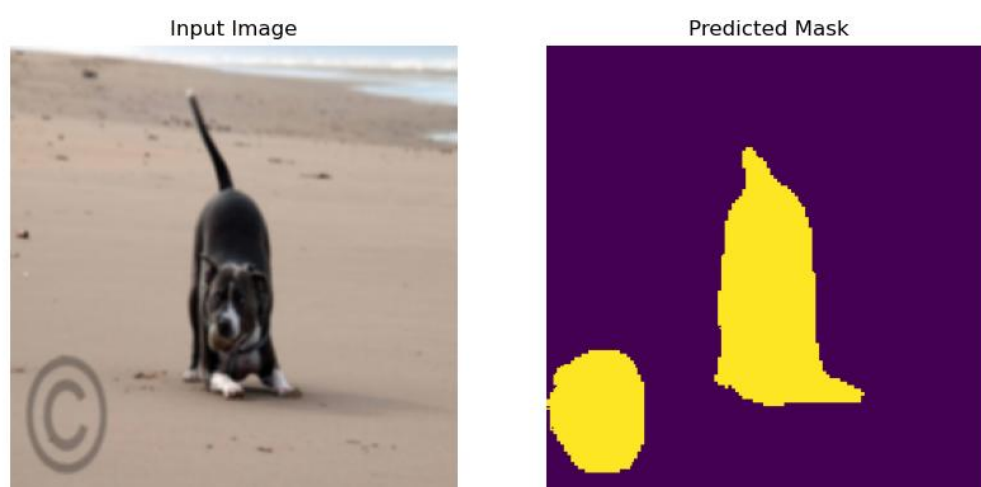
和 `dice_score` 對應，回傳 $1 - \text{dice_score}$ 。

d. `check_loss_line`:

給訓練過程中的 `train loss` 和 `validation loss`，顯示訓練 `loss` 曲線圖。

2. Data Preprocessing

在觀察資料當中，有發現有些照片有浮水印的標籤，以及些許的簽名，甚至有些圖片還有一些噪點，但是這些照片的邊界都非常清晰明確。對於這些和模型預測無關或不穩定的因素，本次實作將資料先經過標準化後，套上一個低通濾波器，抹平雜訊的干擾和將浮水印與背景融合



除此之外也有把 training data 做資料擴增，將其左右翻轉，使資料量變為 2 倍，但是這麼做也要考慮是否會 overfitting，因此如果使用資料擴增，可將 batch size 調高一些。

3. Analyze the experiment results

總體來說 Unet 和 Resnet34_unet 收斂後 validation data 的 dice loss 皆落在 0.15 到 0.15 之間，也就是 dice score 介於 0.85~0.9 之間。

所有的實驗圖片數據皆在.\src\result 當中

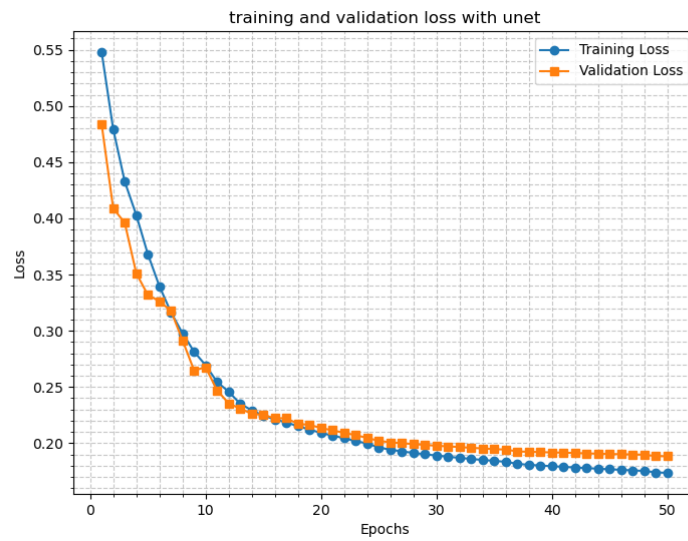
A. Unet

a. feature layer

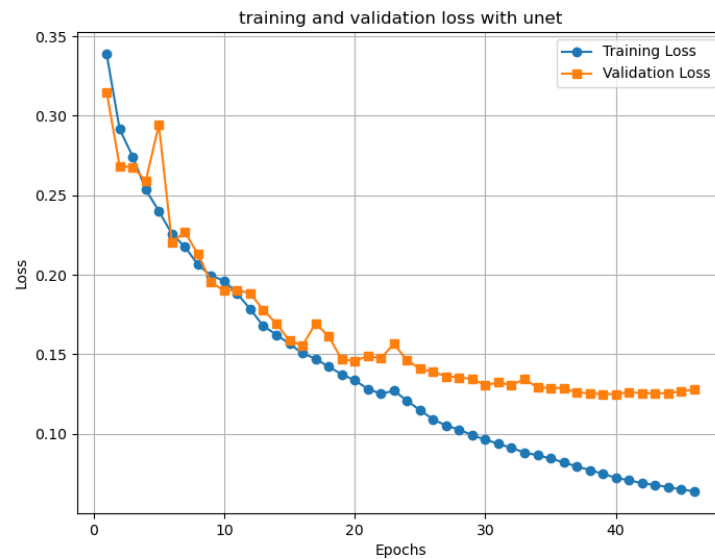
測試 Unet 整體架構的深度影響

```
criterion = utils.dice_loss
optimizer = optim.Adam(model.parameters(), lr=1e-3)
scheduler = torch.optim.lr_scheduler.StepLR(optimizer, step_size=12, gamma=0.6)
early_stopping = utils.EarlyStopping(patience=7, min_delta=0.001, path=path)
```

I. feature = [4, 8, 16]



II. feature = [64, 128, 256]



III. little conclusion

架構越深代表模型能夠掌握更多數據，又因為 Unet 的特性，不會因為架構變深而遺忘先前資訊。

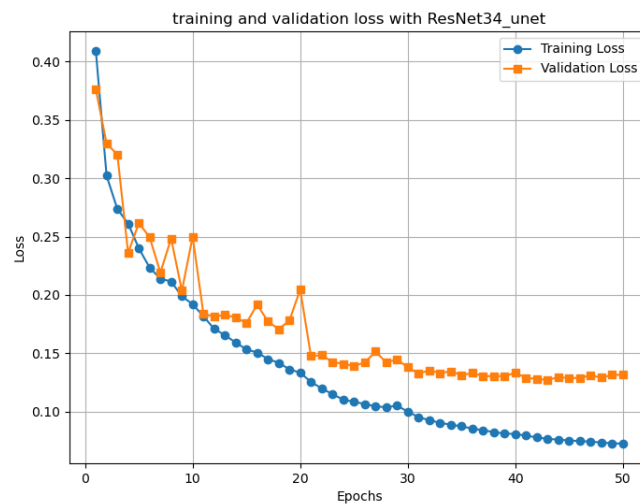
B. Resnet34_unet

a. learning rate

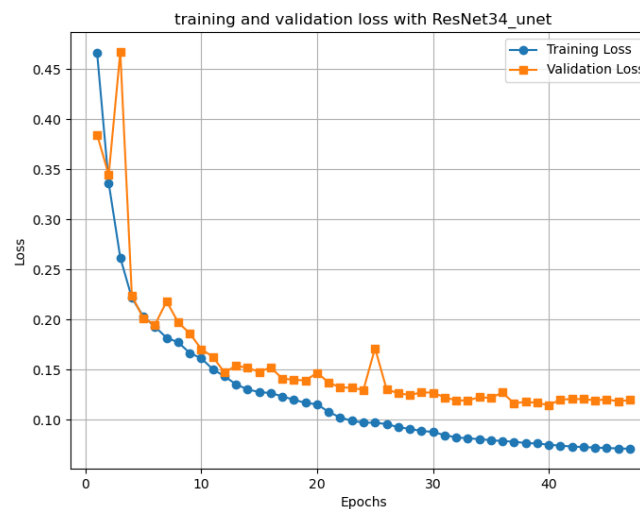
測試 lr 對模型收斂的影響

```
# hyperparameters
criterion = utils.dice_loss
optimizer = optim.Adam(model.parameters(), lr=1e-4) # adjust learning rate
scheduler = torch.optim.lr_scheduler.StepLR(optimizer, step_size=10, gamma=0.5)
early_stopping = utils.EarlyStopping(patience=7, min_delta=0.001, path=path)
```

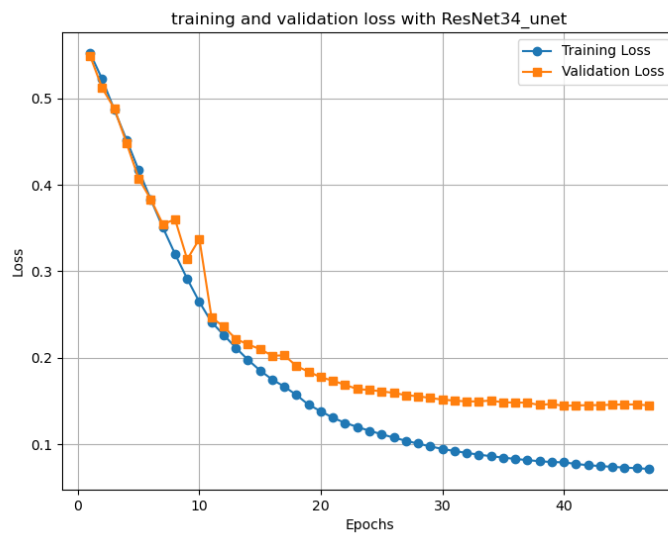
I. $lr=1e-2$



II. $lr=1e-3$



III. $lr=1e-4$



```
Epoch 44/50, Loss: 0.07404502309285678
  Evaluate Dice Loss: 0.1457910438378652
Epoch 45/50, Loss: 0.07320094108581543
  Evaluate Dice Loss: 0.1461658775806427
Epoch 46/50, Loss: 0.07242397620127751
  Evaluate Dice Loss: 0.14612159132957458
Epoch 47/50, Loss: 0.07167728703755599
  Evaluate Dice Loss: 0.14502437909444174
Early stopping...
```

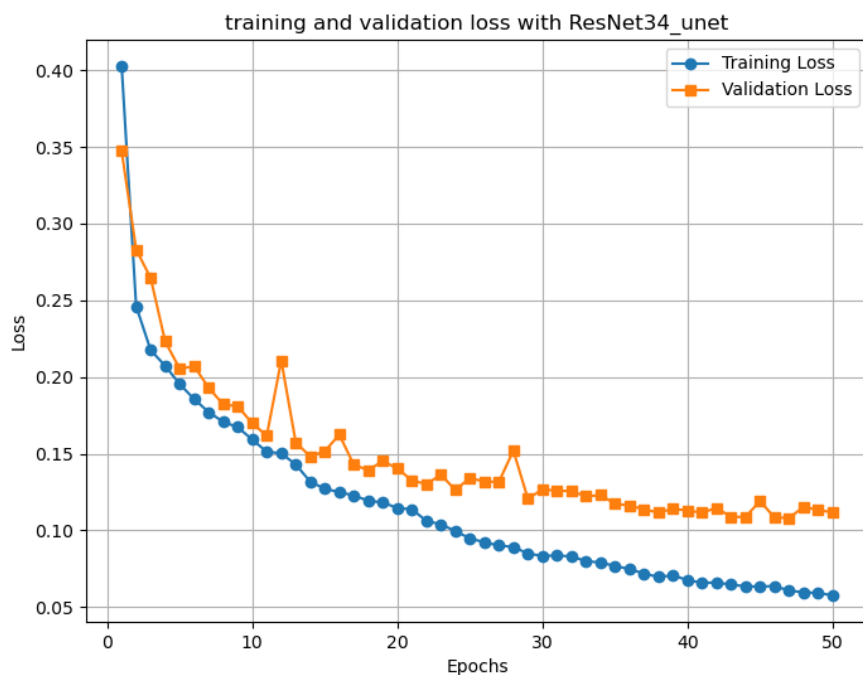
IV. little conclusion

可以發現越小的 lr 模型越趨於穩定，但不代表模型表現會比較好。

C. 其餘的單一實驗

a. 去掉陰影

在觀察模型的 inference，發現有些有陰影的情況模型不好預測，因此在做資料前處理的時候多加上了一層 `remove_shadow` 避免干擾，結果 dice score 沒有明顯提升。



4. Execution steps

A. Training

training 中新增了 `-m` 參數選擇要訓練哪種模型，0 是 Unet，1 是 Resnet34_unet。

B. Inference

新增 `-model_type` 參數選擇要使用哪種模

型，0 是 Unet，1 是 Resnet34_unet。

執行 inference.py 時候，會隨機挑出 10 張圖片顯示標記狀況。

5. Discussion

當模型進行計算時，U-Net 的核心設計就是在於保留先前圖片的資訊。如果是需要在圖片上標記區塊的任務，若僅使用一般 CNN，特徵經過多層 CNN 處理後可能已經和原始圖片相差甚遠，導致無法很好標記。而 U-Net 透過 Skip Connections，將 encoder 中對應層的特徵直接傳遞到 decoder，這樣可以補充該層先前計算的細節，確保模型在慢慢恢復原始圖片大小的過程中仍能保有完整的空間資訊，而提升標記能力。而論文中也有佐證說 U-Net 架構下對於像是醫學方面，不僅只需要少量數據即可訓練，推理速度也相較為快速(512x512 可以在 1 秒內完成)。

reference: <https://arxiv.org/abs/1505.04597>