肺腺癌病理切片影像之腫瘤氣道擴散偵測競賽

II：運用影像分割作法於切割STAS輪廓

報告說明文件

- 參賽隊伍需詳細說明系統流程、演算法、工具、訓練資料與外部資源等。主辦單位會邀請專業人士審查，若發現方法說明有不清的部分，將請參賽隊伍補充，經發現有違規者，將取消獲獎資格。
- 請使用 A4 紙橫式打字，中文字體使用標楷體，英文、數字與符號使用 Times New Roman 字體。
- 版面設定：邊界上下各 2.54CM，左邊 3.17CM、右邊 3.0CM。
- 字體大小：題目 20（粗體）、內文 12，單行間距。
- 繳交程式碼檔案與報告，請 Email 至：xinyan9712@gmail.com，亦可同時副本至：t_brain@trendmicro.com
- 繳交期限至 2022 年 6 月 13 日，逾期將不予受理

壹、 環境

訓練環境：

作業系統：TWCC - cm.2xsuper (4GPU)

映像檔：pytorch-22.02-py3:latest

CUDA Version : 11.6

語言：

Python 3.8.12 (映像檔預設)

主要套件：

MONAI version: 0.8.1

Pytorch version: 1.11.0a0+17540c5

Segmentation_Models_Pytorch version: 0.3.0-dev

AdaBelief-pytorch : 0.2.0

Pillow version: 9.0.0

TorchVision version: 0.12.0a0

tqdm version: 4.62.3

lmdb version: 1.3.0

psutil version: 5.9.0

einops version: 0.4.1

預訓練模型(pretrain model) :

皆使用 Segmentation Models Pytorch[1] (後續簡稱 smp)這個套件中提供的預訓練模型，smp 提供完善的介面 ImageNet 的預訓練模型，此外還

可以使用 timm 的分類模型來做為 Encoder。

額外資料集: 無，僅使用官方提供的訓練資料集

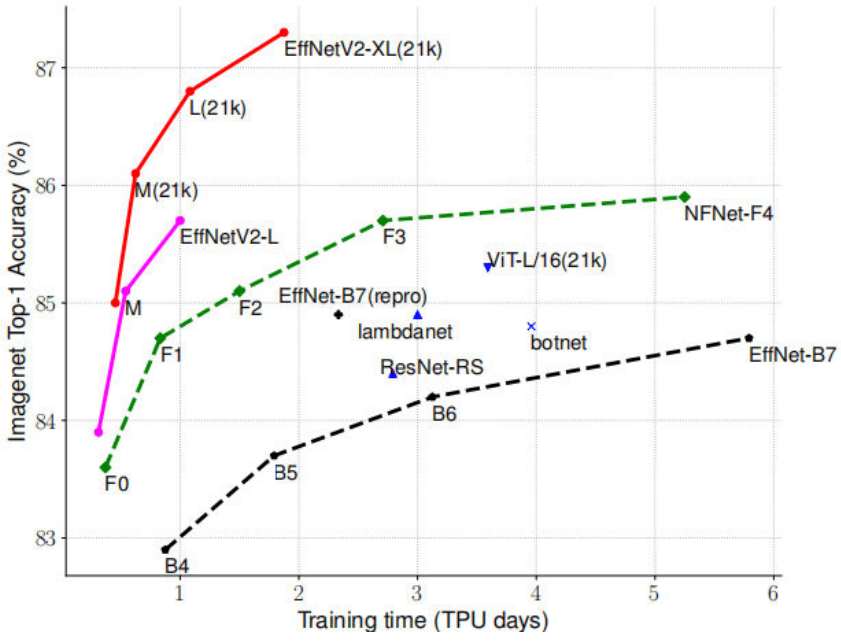
貳、 演算方法與模型架構

模型的部分，本次是採用 Convolutional Encoder-Decoder 架構來做圖片切割模型。一開始使用的是老師提供 example code 中的 Monai 既有的 Unet，但是 Dice 分數約 6、7 成。後來在查資料時查到了 smp 這個好用的函式庫，他可以快速且方便的建模型。

smp 中提供了許多 Decoder 與 Encoder 可以做選擇，此外 smp 還支援 timm 的分類模型作為 smp 其 Encoder 做使用，因為總類繁多所以我先固定 encoder (皆為 Resnet34)，並且從 UNet、FPN 試到 PAN、DeepLabV3Plus，發現 PAN 與 DeepLabV3Plus 的結果較好，所以從這兩個下手，並更改 encoder 來觀察其結果。Encoder 方面，起初因沒經驗便把老師提過的都放進去，如 Resnet152、Desnet201、Xception 等，但是常常因記憶體不足，導致無法訓練。

剛好，我想起了老師在課堂中提過的一個網站，叫做 Paper With Code，這個網站會存放各種模型在不同 Dataset 上的表現及分數，還會解釋模型的架構還有其參數的數量，我找了幾個高分且參數不多且的分類模型來測試 (如圖一)，最後我找到了幾個候選模型來作為 encoder :

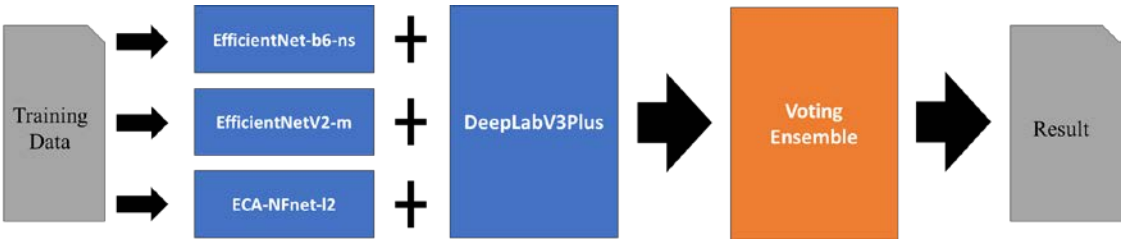
1. EfficientNet (Noise Student) 系列 :
2. Efficient Channel Attention (ECA) NFNet 系列
3. EfficientNetV2 系列



圖一、 EffcientNetV2 之表現圖

而本次我最高分的模型，Decoder 皆為 DeepLabV3Plus，並在 Encoder 方面選用了 EfficientNet-b6-ns、EfficientNetv2-m、ECA-NFNet-l2，然後將這三個模型做 Voting Ensemble，並輸出最後結果。

示意圖(圖二)：

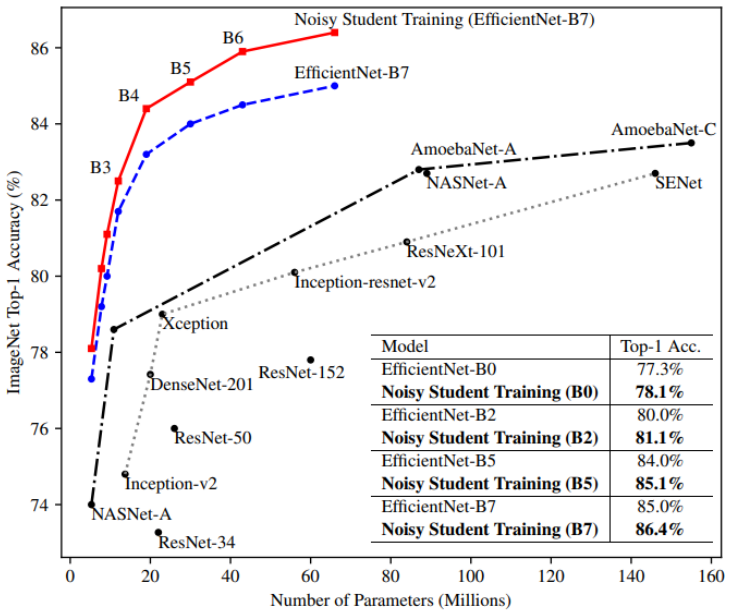


圖二、 模型示意圖

Encoder1 →EfficientNet-b6-ns :

EfficientNet-b6 為 EfficientNet[4]家族第 7 代(從 b0 開始)，EfficientNet 的特點是將原始的 Convolution Layer 改成 MBConv Convolution Layer，使其在相同的功能下，參數卻少了非常多，也代表計算量下降、可以連接更多層，進行更複雜的運算。也有相關的研究使用 EfficientNet 去做癌症分析[2]。

Noise Student 是一種對無標籤的照片的一種訓練方法，原理為先用有標籤的照片訓練出一個模型，然後將沒有標籤的圖片給它去預測並得出標籤，然後再將原本沒有標籤的照片，用剛剛預測出來的標籤作為偽標籤(pseudo label)，一起放入原先模型訓練。偽標籤作為"Noise"能解決 Overfiiting 的問題，讓新模型的分數比原本更高(圖三)，而我們使用的權重為 EfficientNet-b6 在 ImageNet 資料集上使用 Noise Student 方法訓練出來最好的

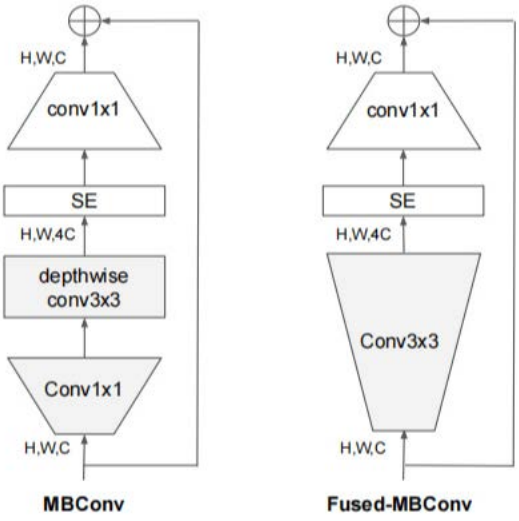


圖三、 Noise Student 表現

模型權重。

Encoder2 →EfficientNetV2-m:

EfficientNetV2[5]為 EfficientNet 的進階版，它將 MBConv 改成 Fused-MBConv。在 MBConv 的網路中，Depthwise convolutions 計算速度會很慢，因為它通常無法充分的利用 GPU 加速，因此第二代將 MBConv 改良成 Fused-MBConv，雖然參數量變多了，但是計算變快，準確率也變高了，如圖四。

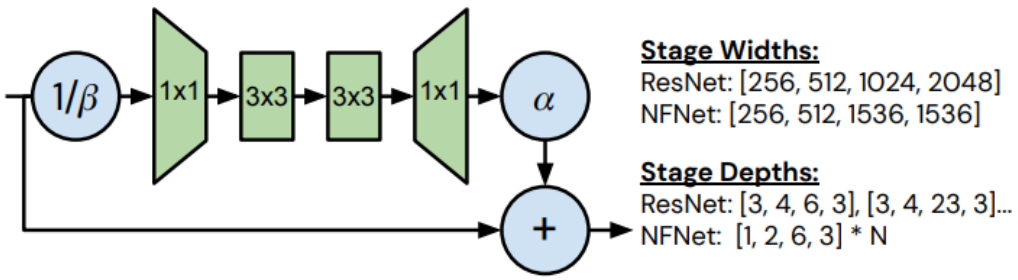


圖四、 EfficientNet 一、二代差異

這次比賽選用的是 Medium 型號，因為 Large 及 Xlarge 模型太大了，並且使用的 Pretrain Model 是 EfficientNetV2 在 ImageNet-21K 資料集上訓練出來的模型。

Encoder3 →ECA-NFnet-l2:

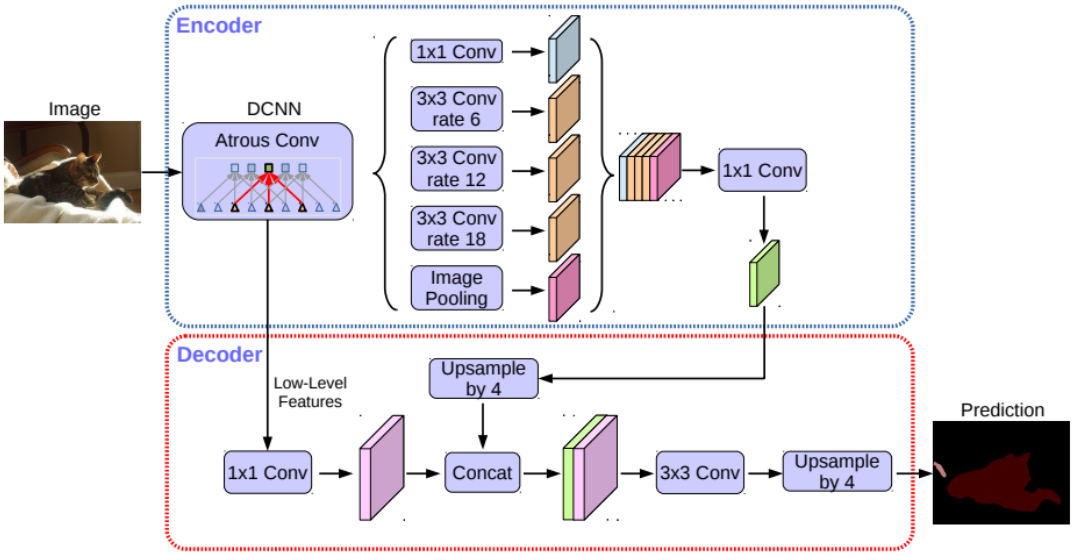
Efficient Channel Attention -Normalizer-Free ResNets – L2 簡稱 ECA-NFnet-l2，NFNet 是一種基於 ResNet 但沒有使用 Batch Normalization，而是使用 Adaptive Gradient Clipping (AGC)的一種模型(如圖五[6])。其好處是可以減少記憶體使用量與模型訓練時間，也可以讓模型對 BatchSize 的大小不那麼敏感。這次使用的是原本 NFNet-F2 透過 Efficient Channel Attention (ECA)的方式所改良的 Light 版，此改良能讓模型進一步的縮小，且訓練更加快速。



圖五、 NFNet 示意圖

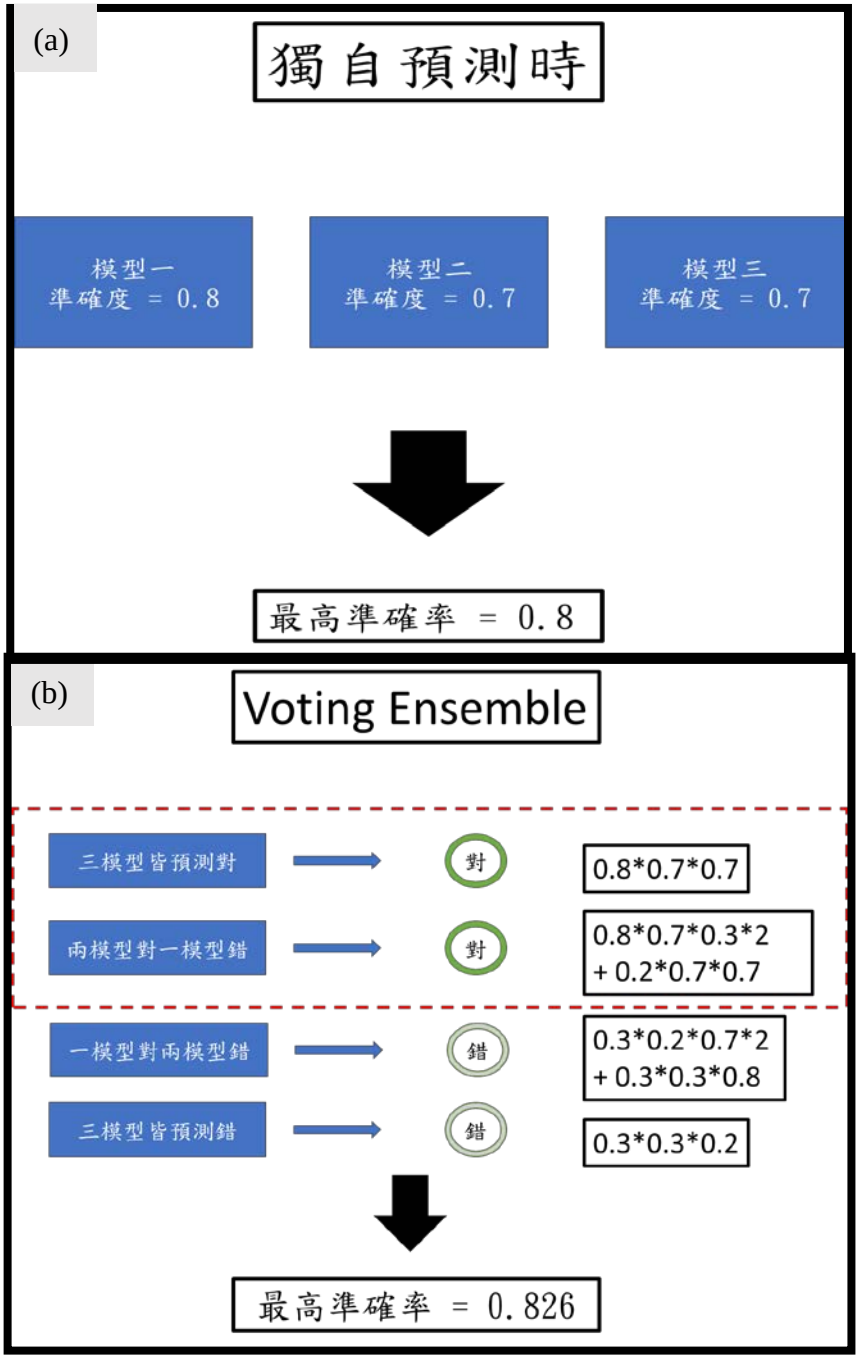
Decoder →DeepLabV3Plus

空間金字塔池化模塊(Spatial pyramid pooling module)與編碼解碼器結構常用於語義分割任務。前者能夠通過以多種速率和多個有效視野使用過濾器或池化操作探測輸入特徵來編碼多尺度上下文訊息；而後者則可以透過逐漸恢復空間信息來捕獲更清晰的對象邊界。DeepLabV3Plus[7]結合兩種方法的優點。具體來說， DeepLabV3Plus 透過添加一個簡單而有效的解碼器模塊來擴展 DeepLabPlus，以細化分割結果，尤其是沿著對象邊界，其結構如圖六。



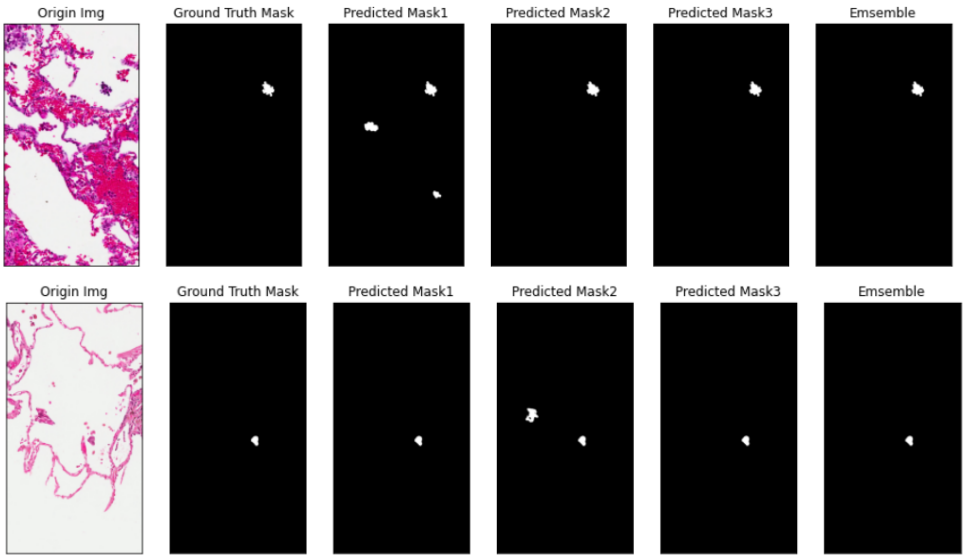
圖六、 DeepLabV3Plus 示意圖

接下來介紹我使用的 Voting Ensemble：假設有三個二元分類之模型做 Voting Ensemble，則每次 output 的結果，都是由三個人下去投票後多數決的結果。例如模型一與模型二預測出 1，模型三預測出 0，則最後的結果就是 1。如圖七(a)的例子，有三個彼此獨立的模型，其準確度為 0.7、0.7、0.8，當三個模型獨自預測時，最高的準確率是 0.8；但是當進行 Voting Ensemble 時，則可以得到最高 0.826 的準確度，相較於獨自預測時高出許多。



圖七、 Ensemble 示意圖（a）無 Ensemble（b）有 Ensemble

雖然實作中較難做到 Voting Ensemble 的模型皆獨立，無法像上述例子一樣提升這麼多的準確度，但是一般來說是會提升的，例如這次比賽實際的例子:



圖八、 Ensemble 示意圖

圖八的上圖很明顯的可以看到第一個模型的預測結果與實際圖相差甚遠，圖八的下圖很明顯的可以看到第二個模型的預測結果與實際圖相差甚遠，若模型皆單獨使用，則模型一與模型二會各有一張照片 Dice 分數極低。但是藉由 Ensemble，可以看到因為只有單一個人畫錯很多其他都幾乎正確，因此該人的錯誤會被隊友彌補，對於最後的 Ensemble 模型來說，這兩張預測的照片的 Dice 分數皆會被提高。

參、 資料處理

前處理的部分，是參照老師在深度學習課堂上提供的參考資料下去修改，主要包含:

1. 將遮罩的 json 檔轉換成 png：由資料夾中引入官方提供的標準答案的 json 檔，並透過 PIL 這個函式庫，將其轉為 png，這部分為固定程序所以並未對老師的程式做改寫。
2. 撰寫 Transform，以套用到讀入的照片上，我這裡使用了七種 Transform，有六種式圖片前處理，只有一種是資料增強，主要有:
   a. **LoadImaged (img、seg)**
      為 Monai 這個函式庫中的函式，它的強大在於它可以透過更改 reader 來讀取不同的資料類型，例如.bmp 或是.npy。
   b. **AsChannelFirstd (img)、AddChanneld (seg)**
      前者的目的是要將三維的圖片，其 Channel 所在的位置改到最前面，如(1716, 942, 3)變成(3, 1716, 942)，以迎合 Pytorch 的格式；後者是為了讓遮罩的維度與模型的輸出吻合，例如遮罩的照片讀進來只有二

維，如 (1716, 942)，但是模型的輸出為三維(1, 1716, 942)這兩者會因維度大小不同被系統判定無法比較。

c. **ScaleIntensityd (img、seg)**

將強度縮放到一個區域內，以消除每一張照片的個體差異。

d. **Resized (img、seg)**

之前有跟老師討論過要用 RandomCrop 還是 Resize，老師說這次比賽是游離的癌細胞，因為是游離，必須看該癌細胞的周圍才知道他是否為游離癌細胞，還是有與組織相黏，所以 Resize 會比 Crop 適合;圖片大小的部分，老師給的範例是 Resize 成 800x800，我也做過相似大小的測試，從 768x768 到 896x896 每 32 為一單位，會設 32 的原因是因為模型在做 Maxpool 或 Avgpool 時圖片大小會降成一半，因為有些模型會進行五次 pooling，代表圖片會變三十二分之一，所以我就設定 32 的倍數，但是不管是調大到 896x896 或調小到 768x768，Dice 分數都沒差多少，所以就不更改 800x800 這個分辨率。此外也試過將照片縮小到 512x512 甚至 256 x 256，這樣的好處是批次大小可以變大 2~3 倍，但是造成 Dice 分數很不理想：例如有些腫瘤區域很小的照片，會被預測成一張全黑的圖，表示這個腫瘤直接被忽略，這樣是不行的，所以我仍使用大張的照片做訓練。

e. **RandRotate90d (img、seg)**

為資料增強，機率設為 0.3，代表每一批圖片進來都有 0.3 的機率，順時針旋轉 90 度。

f. **EnsureTyped (img、seg)**

為對 transform 中的圖片做最後確認，主要是確認圖片的資料型態是否正確。一般是 torch.FloatTensor，但有時候會因為一些 Transform 的操作導致圖片或遮罩變成不正確的資料型態，這時就要用這個 EnsureTyped 來更改回來。

3. 製作訓練、驗證以及要上傳的 Dataset，並轉成 Dataloader 形式，且套用上面寫好的 Transform，以便後續作訓練。此外這個階段也會將照片分成 Train 跟 Validation，以便後續模型的訓練以及驗證。

a. **Dataset**

使用的是 Monai 函式庫中的 monai.data.Dataset，他的方便之處在於：我們只要將圖片以及遮罩的路徑用字典檔寫好，並用一個 list 包裝起來丟給它，便能改寫成 torch 的 Dataset，之後就可以放入 torch 的 Dataloader；此外在製作 Dataset 時，因為會將圖片路徑裝入 list，此時可以對 list 進行切割(indice)，即可輕易的分得 Train 跟 Validation 兩堆圖片了。

b. **Dataloader**

Dataloader 是使用 torch.utils.data.DataLoader 即 torch 函式庫既有的函式，Dataloader 的好處是將你的資料作成類似 generate 的資料型態，

generate 跟 list 最大的差別在於，list 會一次將其中的所有元素讀取好，因此你可以指定你要哪一個元素；但 generate 並不會將全部讀取進來，一次只會讀取第一批資料，用完再取第二批，這樣做的好處是 generate 較不佔用記憶體。使用 Dataloader 只需管一個 Batchsize 的圖片會不會讓記憶體過載，而 list 需要看全部的資料量，因此，使用 Dataloader 可以用更大更多的照片去訓練。

肆、 訓練方式

1. 損失函數(Loss function)

本次使用的損失函數(Loss function)為官方指定的 Dice 函數，使用的為 Monai 上現有的函式 DiceMetric。訓練方面，三個模型為分開訓練，出各自的預測結果之後，在最後將結果套用 Ensemble。

2. 批次大小(BatchSize)

批次大小的部分， tf_efficientnetv2_m_in21ft1k 的批次大小為 16、tf_efficientnet_b6_ns 為 16、eca_nfnet_l2 為 24。

3. 優化器(Optimizer)

優化器的方面，三個模型皆使用 AdaBelief[8, 9]，他是適應性法（Adaptive Method）的系列作，與 Adam、AdamW 屬於同系列。AdaBelief 的特點在於其在 Adam 的公式上將二階動量項做了改寫，也引入了一個超參數 ε，改動 ε 可以讓 AdaBelief 二階動量項被改動，讓其預測準確度會高於 Adam 而達到 SGD 的準確度，表一為 AdaBelief 的表現圖，而且 AdaBelief 用的是適應性法，因此速度會比 SGD 快上很多。

AdaBelief	SGD	AdaBound	Yogi	Adam	MSVAG	RAdam	AdamW
70.08	70.23†	68.13†	68.23†	63.79† (66.54‡)	65.99	67.62‡	67.93†

表一、 AdaBelief 表現

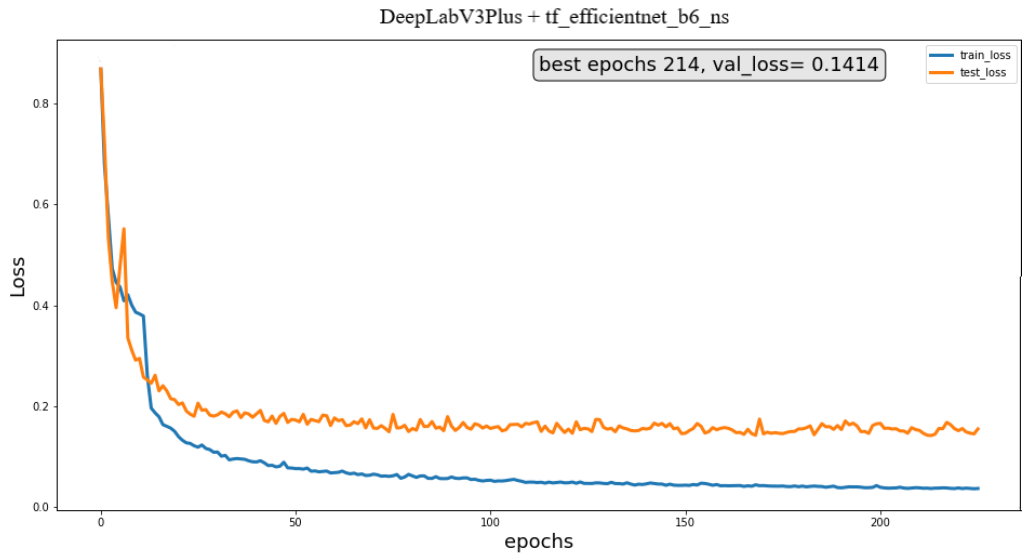
此外，三個模型使用的 Dataset、Transfrom 皆相同。

伍、 分析與結論

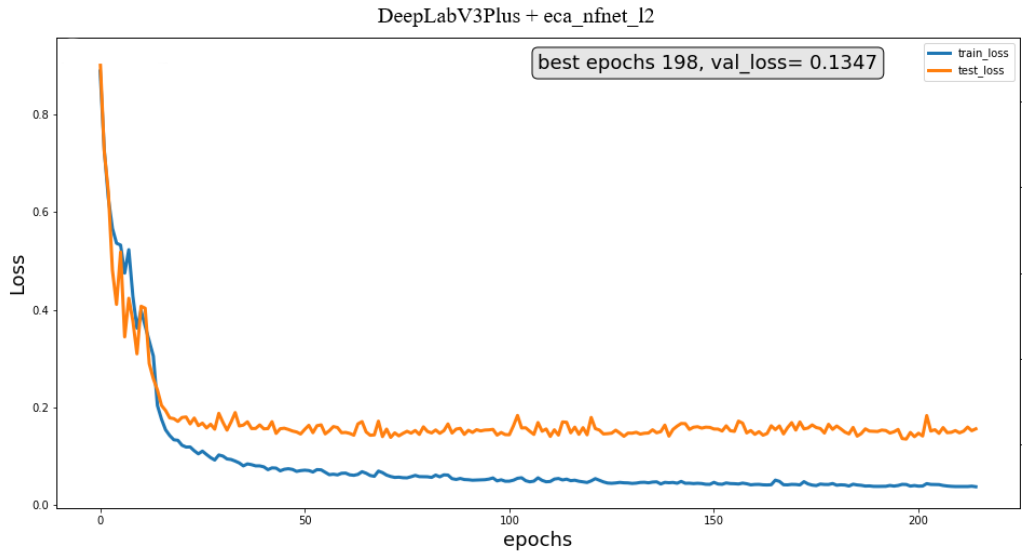
所有的模型都訓練 200 個 Epoch 以上，並儲存在驗證資料集(Validation)上 loss 最低的模型之權重。

結果展示：

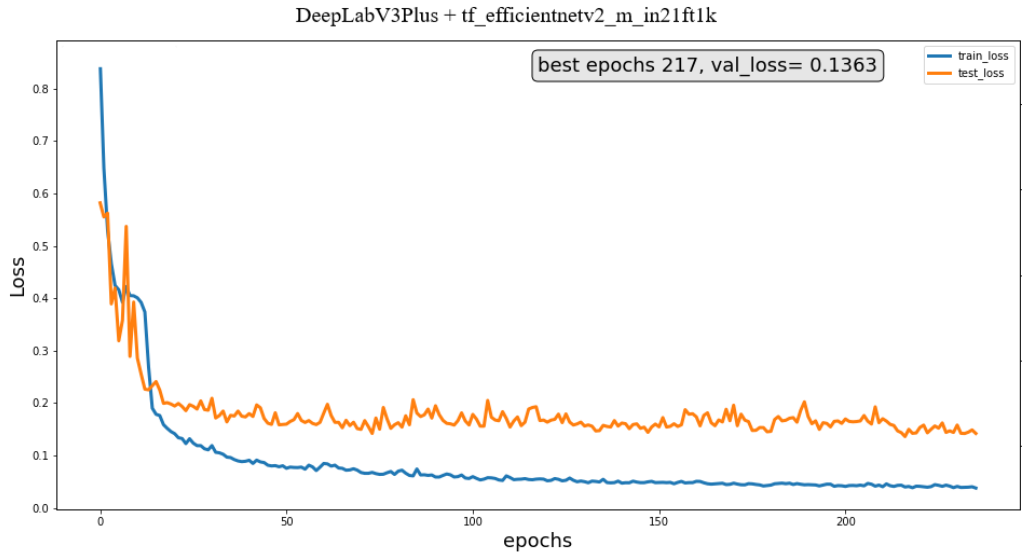
1. DeepLabV3Plus + tf_efficientnet_b6_ns



2. DeepLabV3Plus + eca_nfnet_l2



3. DeepLabV3Plus + tf_efficientnetv2_m_in21ft1k



成績一覽

Encoder	EfficientNet-b6-ns	ECA-NFNet-l2	EfficientNetv2-m*	Ensemble
Public/Private	0.86/0.90	0.88/0.90	x	0.90/0.91

*EfficientNetv2-m 因上傳次數不足未上傳

分析與討論：

由結果圖可以看出 200 回合後 Loss 幾乎已收斂，且 Training Loss 與 Validation Loss 幾乎平行，這表示沒有太嚴重的 Overfitting。此外透過觀察各個模型的成績，可以看出做 Ensemble 後，Dice 分數有顯著的提升。

本次最高分的模型，都不是該系列的最高階版本，eca_nfnet_l2 之上還有 eca_nfnet_l3，tf_efficientnetv2_m_in21ft1k 之上還有 tf_efficientnetv2_l_in21ft1k ，tf_efficientnet_b6_ns 之上還有 tf_efficientnet_b7_ns。不選用的原因為：eca_nfnet_l3 在 timm 上並無預訓練模型，沒有預訓練的模型會導致訓練時間變很久，訓練 200 個 Epochs 後 Public 分數卻僅有 0.81，故在當時我就沒用採用這個模型；而後兩者是模型實在是太大了，在同數量的 GPU 環境下，若採用上位版，批次大小則必須縮小一半或變三分之二，此時的訓練就會變得較久，且在 Public 分數未有顯著的提升，故使用的是較小的版本。

至於優化器的方面，選擇 AdaBelief 是因為老師上課提供的 NVIDIA DLI 課程內的伺服器一次只能跑八個半小時，且其上傳下載皆很耗時間導致無法存下模型之後接續跑。因此我的訓練都是以八小時為限，不管它跑了多少的 Epochs 都必須停止，因此收斂較慢的 SGD 效果很差，收斂較快的 Adam 分數上不去。後來上 Github 與 Kaggle 找尋新優化器，找到了既快又分數不錯

的 AdaBelief。不過後來 TWCC 來之後，我並未測試 SGD，就一直使用 AdaBelief，所以也許給 SGD 多一點的訓練時間，其結果會更好。

結論:

AdaBelief 做為新穎的優化器，它可以讓模型在有限的計算資源(或訓練時間)內，得到較的成果。

最後做個三點總結，第一點模型並非越新越好或是參數越多越好，這攸關你的任務，如果訓練的是大型的東西的輪廓如自駕車系統，那使用近兩三年很紅的，內含 Transformer 的 CNN 模型也許不錯，但是若像這次比賽照片，有些腫瘤很大但有些是小到剩一個點，而且形狀還不固定的就不太適合，此外還考量你的訓練資源是否足以支持你使用如此巨大的模型；第二點是能使用 Pretrain Model 就使用 Pretrain Model，或使用 Pretrain Model 來做微調也可以，這麼做的好處是可以減少訓練時間，還可以減少訓練所需的資料數；第三點是如果你試了一種新模型，或與你之前常使用的訓練方法不同的模型，既使準確度可能不佳，都可以保留下來，也許經過 Ensemble 之後能得到榜上有名的成績。

未來可改進之處:

上面提到我使用 SGD 因為訓練時長過短導致效果不佳，所以未來可以嘗試 SGD 使用 SGD 並訓練較長時間，並看其結果有無更好。此外就是有些模型是因為太大跑不動而無法使用，但未來如果有幸使用到強大的計算資源(如台灣杉二號)時也許可以試試看。

再來是模型的部分，未來可以嘗試更多的模型做 Ensemble，彼此互補達到更好的分數，或是進行 Pseudo-Labelling。Pseudo-Labelling 是一種 semi-supervised learning，其做法是先使用原本的 Training image 訓練出一個模型，並用此模型預測 Public 及 Private dataset，之後將 Public 及 Private 與模型出來的 mask 一起放入模型中訓練。雖然 Public 及 Private 的 Label 並非完全正確，但卻有助於區分各類別，讓模型泛化能力較高[10, 11]。

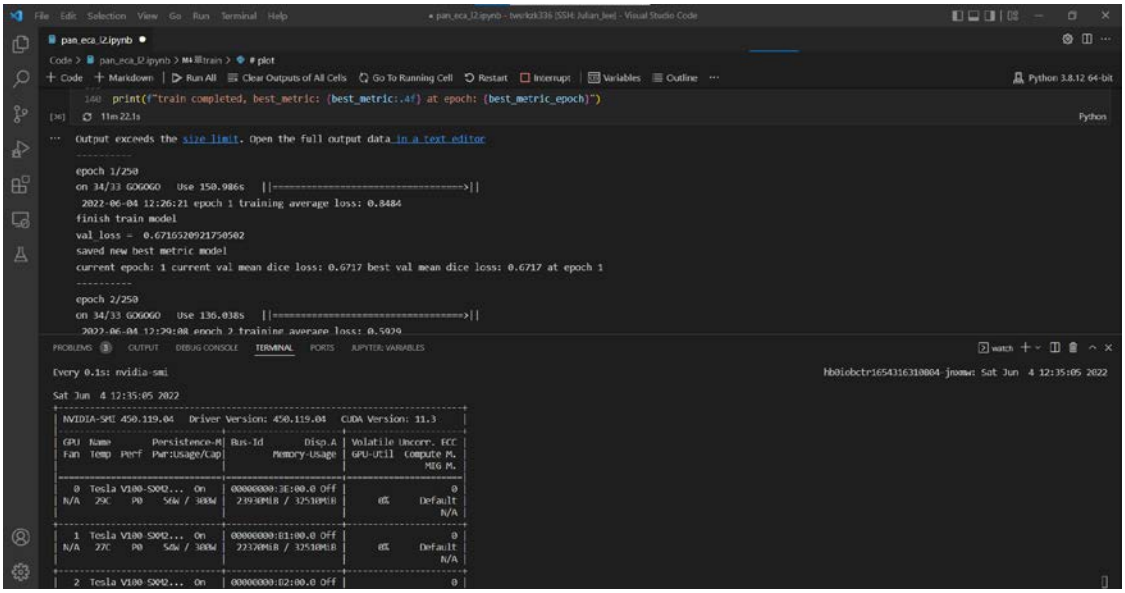
心得 :

這次很高興藉由系上開的深度學習基礎概論參加到這次的活動，也很幸運的拿到 Public 第四名、Private 第七名的成績，這次的參賽除了讓我學到影像切割的基本寫法，還讓我認識到了 Monai 這個好用的函數庫，最重要的是，我也藉著這個比賽，學習到了如何查資料、找尋更好的模型即演算法，並實際應用於比賽，好比說這次用的 smp 與預測結果的 Ensemble，也都是這次比賽才學會的，此外我也學到了上 Paper with Code 與 GitHub 查找資料的能力等，這份 Word 是我的訓練經歷與分享，希望能在最後的結果上得到不錯的成績，也期待未來也有類似的比賽可以來參加。

陸、 雲端使用

透過這次的肺腺癌切割比賽，我獲得了十多天的 TWCC 資源，也開啟了新世界的大門，首先方便的開發形容器(CSS)，以往若在自己的電腦或是課堂 Server，要切環境就只能用 Anaconda 建立新的 Enviroment，在 CSS 若需要 Pytorch 的環境就使用 Pytorch 映像檔，需要 TensorFlow 就使用 TensorFlow 的映像檔，此外還可以藉由選擇映像檔版本來切換不同的環境，如 Cuda11.6、Cuda11.3…等，可以快速的根據自己的需求選擇環境，也可以方便管理每個環境。附圖為 CSS 實際使用情況。

此外 TWCC 的好處除了可以租借很多顆高級 GPU 來跑模型外，還可以多開並同時跑不同的程式，以前都是訓練太慢一天三次上傳不完，現在是東西太多有時候都希望一天可以丟個 10 次之類的。還有就是 TWCC 的開發型容器也提供 SSH 連線，這就意謂這我可以使用我熟悉的軟體來進行訓練模型(如 VSCode)，這可以縮短熟悉新開發介面的時間，還有就是可以加裝 Coding 時提示的插件，使你輸入 np.ar 會自動跳出 np.array 等方便的功能，下圖為在 VSCode 上跑程式，並同時確認 GPU 的使用情況，以調整圖片大小或 Batchsize 等參數。



附上之前測試各種模型結果

https://github.com/JulianLee310514065/Miscellaneous/blob/main/TWCC/%E8%B7%91%E5%88%86.md

柒、 程式碼

GitHub : https://github.com/JulianLee310514065/AICUP_STAS_Segmentation

捌、 使用的外部資源與參考文獻

[1] Yakubovskiy, P. (2020). Segmentation Models Pytorch. GitHub. Retrieved from https://github.com/qubvel/segmentation_models.pytorch

[2] Wang, J., Liu, Q., Xie, H., Yang, Z., & Zhou, H. (2021). Boosted efficientnet: Detection of lymph node metastases in breast cancer using convolutional neural networks. *Cancers*, *13*(4), 661.

[3] Xie, Q., Luong, M. T., Hovy, E., & Le, Q. V. (2020). Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 10687-10698).

[4] Tan, M., & Le, Q. (2019, May). Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning* (pp. 6105-6114). PMLR.

[5] Tan, M., & Le, Q. (2021, July). Efficientnetv2: Smaller models and faster training. In *International Conference on Machine Learning* (pp. 10096-10106). PMLR.

[6] Brock, A., De, S., Smith, S. L., & Simonyan, K. (2021, July). High-performance large-scale image recognition without normalization. In *International Conference on Machine Learning* (pp. 1059-1071). PMLR.

[7] Chen, L. C., Zhu, Y., Papandreou, G., Schroff, F., & Adam, H. (2018). Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 801-818).

[8] Zhuang, J., Tang, T., Ding, Y., Tatikonda, S. C., Dvornek, N., Papademetris, X., & Duncan, J. (2020). Adabelief optimizer: Adapting stepsizes by the belief in observed gradients. *Advances in neural information processing systems*, *33*, 18795-18806.

[9] juntang-zhuang/Adabelief-Optimizer: Repository for NeurIPS 2020 Spotlight "AdaBelief Optimizer: Adapting stepsizes by the belief in observed gradients"

[10] Lee, D. H. (2013, June). Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML* (Vol. 3, No. 2, p. 896).

[11] Zoph, B., Ghiasi, G., Lin, T. Y., Cui, Y., Liu, H., Cubuk, E. D., & Le, Q. (2020). Rethinking pre-training and self-training. *Advances in neural information processing systems*, *33*, 3833-3845.