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

作業環境：

使用老師上課時所提供的 NVIDIA 伺服器。

作業系統：Ubuntu 18.04.4 LTS

記憶體：16 GB

GPU：Tesla T4

CUDA version：11.0

程式語言：

Python 3.6.9

使用套件：

MONAI version: 0.8.1

Numpy version: 1.22.2

Pytorch version: 1.11.0a0+17540c5

Segmentation_Models_Pytorch version: 0.3.0-dev

adabelief-pytorch : 0.2.0

Pillow version: 9.0.0

Tensorboard version: 2.8.0

TorchVision version: 0.12.0a0

tqdm version: 4.62.3

lmdb version: 1.3.0

psutil version: 5.9.0

pandas version: 1.3.5

einops version: 0.4.1

預訓練模型：

Segmentation Models Pytorch[1] (SMP)

Pytorch Image Models (timm)

額外資料集：

無，皆使用官方所提供之資料集。

貳、 演算方法與模型架構

本次比賽所使用的架構為編碼-解碼器結構(Encoder-Decoder Structure)，其中編碼器使用的為 ECA-NFNet-l0 及 EfficientNetV2-s，解碼器使用的為 PAN 及 DeepLabV3+，皆使用預訓練模型，並組合成三種不同的子模型，如圖一所示，並會在後續介紹。

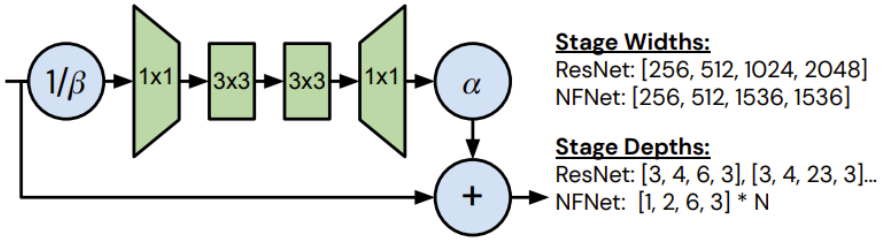
訓練完的三種子模型會再經由集成投票(Ensemble Voting)結合，而得出更高的準確率。

圖一、 模型架構

1. 編碼器(Encoder)

   a. Efficient Channel Attention -Normalizer-Free ResNets-L1

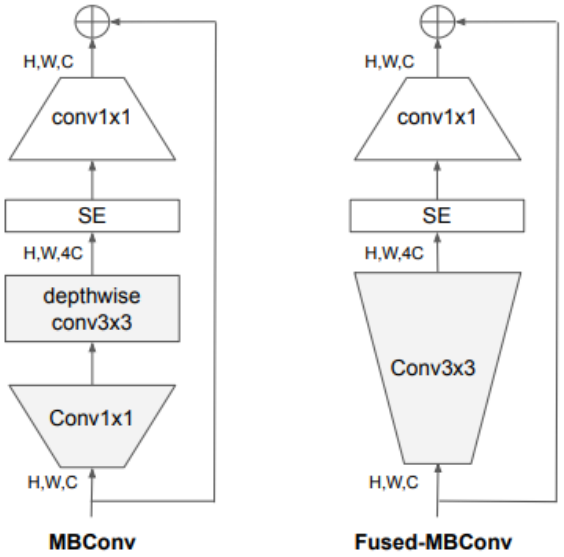
   NFNet 是一種基於 ResNets、但不使用批次標準化(Batch Normalization，BN)的神經網路模型，架構如圖二所示。為了改善不使用批次標準化造成的不穩定，提出自適應梯度修剪(Adaptive Gradient Clipping，AGC)。其訓練速度提高了 8.7 倍，且最佳模型準確率達到 89.2%。而本次比賽所使用的 L1 為原始版本中的 F1 利用高效通道注意(Efficient Channel Attention，ECA)的方式所建構出的精簡版。



圖二、 NFNet 架構[2]

b. EffcientNetV2-S

EfficientNetV2 是 EfficientNet 的第二代版本。其將第一代版本中所使用的 MBConv 替換為 Fused-MBConv，因而得到更快的訓練速度與更小的體積，此差別由圖三所示。在本次比賽中使用體積最小的 S 版本。

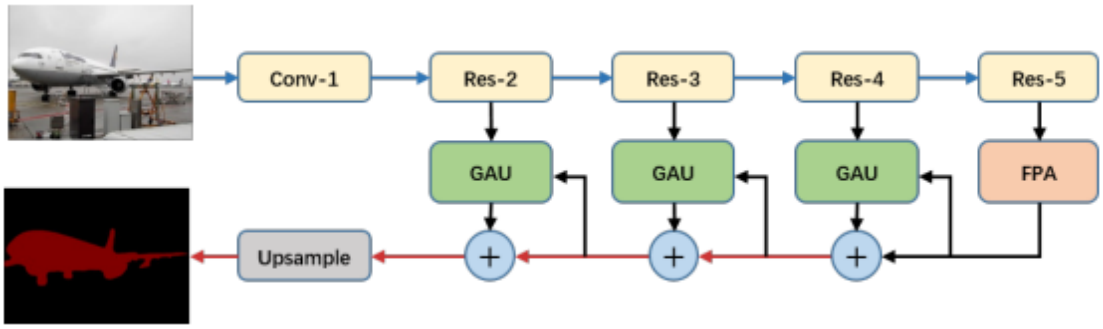


圖三、 MBConv 與 Fused-MBConv 架構[3]

2. 解碼器(Decoder)

a. Pyramid Attention Network

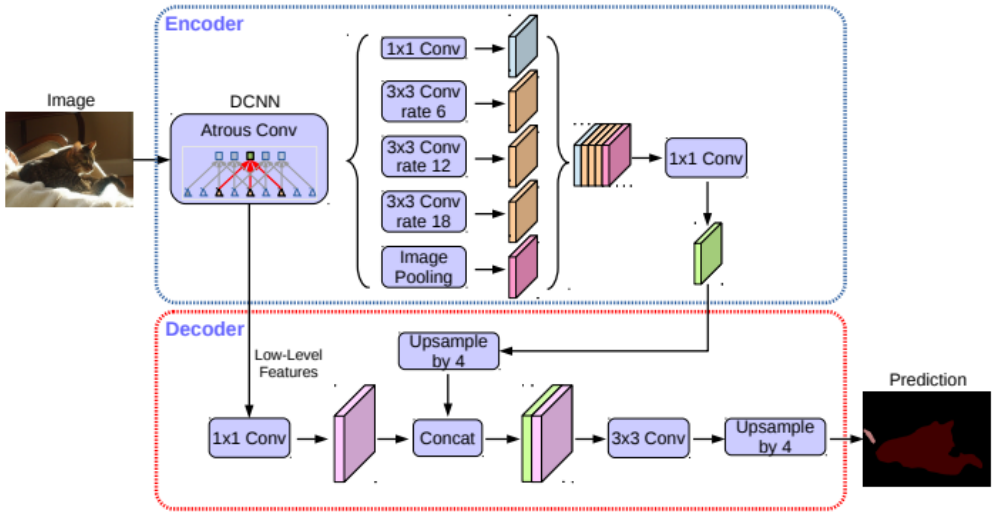
PAN 為結合注意力機制和空間金字塔提取用於像素分類的特徵的模型，並提出 Feature Pyramid Attention module(FPA)和 Global Attention Upsample module(GAU)兩種模組，架構如圖四所示。



圖四、 PAN 架構[4]

b. DeepLabV3+

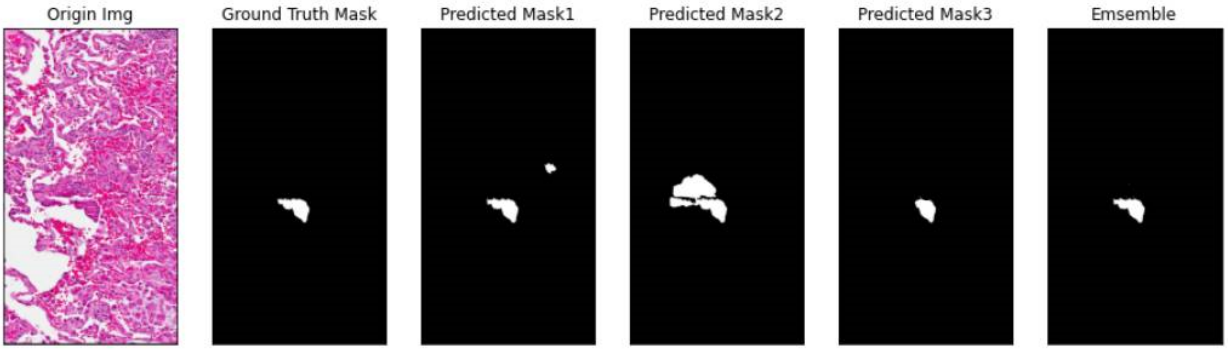
DeppLabV3+結合了空間金字塔池化模組及編碼-解碼器兩種用於語義分割的方法，且相較於 DeepLabV3 擁有更有效率的解碼器，架構如圖五所示。此模型在 PASCAL VOC 2012 和 Cityscapes 的測試資料上達到 89.0%及 82.1%的正確率。



圖五、 DeepLabV3+架構[5]

3. 集成投票(Ensemble Voting)

集成投票是一種將不同模型所預測出的結果組合在一起，以得到更好的泛化性。圖六是在本次比賽所用的模型中，其中一個使用集成投票的成果：在一開始三個模型所預測出的結果與原始標記相差甚大，但在經過集成投票後，可看出最終結果與原始標記非常相似。



圖六、 集成投票效果

參、 資料處理

資料處理的部分以參照老師所提供的程式碼為主，可以分成三個部分：Shape to Mask、Transform 及 Create DataLoader，順序如圖七所示。

圖七、 資料處理順序

1. Shape to Mask

   將官方所提供的遮罩檔案利用 PIL 函式庫，將檔案由.json 檔轉成.png 的形式。此部分為標準程序，因此並未對老師所提供的檔案多加改寫。

2. Transform

   Transform 使用 MONAI 的函式庫，在本次的比賽中，使用了下列功能。

   a. LoadImaged

      讀取影像資料，且可以讀取不同類型的圖形檔案。

   b. AsChannelFirstd

      將圖形的通道維度更改為第一維度，以符合 PyTorch 的格式。

   c. AddChanneld

      在輸入影像中添加長度為一的通道維度，讓二維資料的遮罩變成三維資料，以利後續處理。

   d. ScaleIntensityd

      將輸入影像的強度縮放到固定的範圍內，以消除每張照片的個體差異。

   e. Resized

      為一種資料增強的方法。從中心裁剪或擴展影像大小。
      使用此方式是因為在本次的比賽所要區分的為游離的癌細胞，需要判斷標記的周圍是否有與組織相黏，因此不建議使用有切割的資料增強方法避免失去標記意義。

   f. EnsureTyped

      確保輸入資料為 PyTorch Tensor 或 numpy array，避免在轉換過程中造成資料型態錯誤。

3. Dataset、DataLoader

a. Dataset

使用 MONAI 函式庫的 Dataset。可以將以字典檔撰寫的影像及遮罩路徑包裝後，轉成一般 PyTorch 的 Dataset 的形式。

b. DataLoader

使用 PyTorch 函式庫的 DataLoader。用以控制每次訓練的影像數量，避免記憶體超載。

肆、 訓練方式

本次比賽中的三個模型皆為獨立訓練後，再將最終結果作集成投票。每個模型訓練時的資料處理方式、資料集皆相同。批次大小(Batch Size)、損失函數(Loss Function)、優化器(Optimizer)如下。

1. 批次大小

由於記憶體大小有限，因此每個模型的批次大小皆為 4。

2. 損失函數

MONAI 函式庫的 DiceMetric。Dice 係數即為 F1-Score，計算真陽性率(True Positive Rate, TPR)及陽性預測值(Positive Predict Value, PPV)的調和平均值(Harmonic Mean)，公式如下：

$$DC = \frac{2 \times TP}{2 \times TP + FP + FN}$$

3. 優化器

優化器皆使用 AdaBelief。AdaBelief 結合 Adam 的快速收斂性及穩定性及 Stochastic Gradient Descent(SGD)的泛化性。表一為不同優化器的準確率，可以看出 AdaBelief 與 SGD 的分數相當接近，但訓練時間縮短很多。

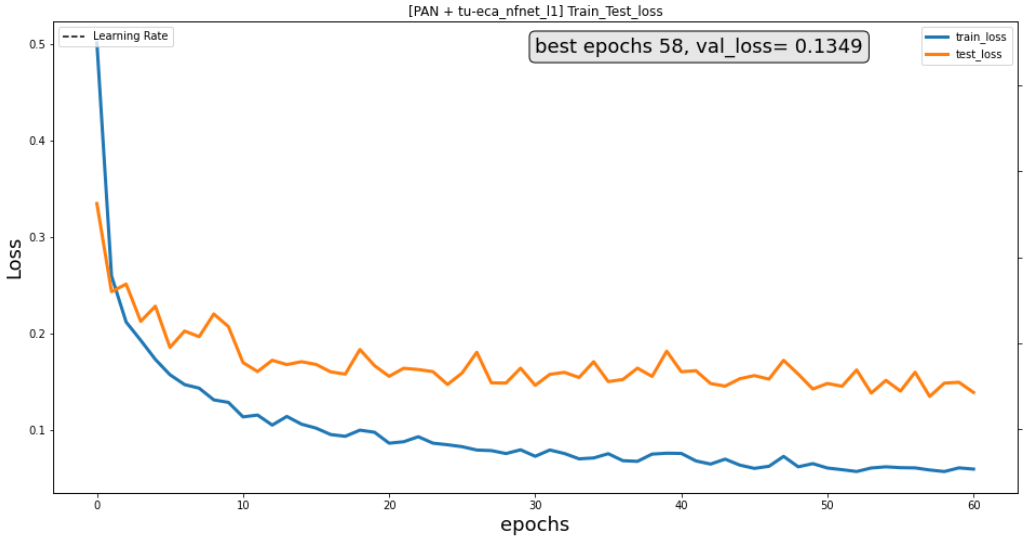
AdaBelief	SGD	AdaBound	Yogi	Adam	MSVAG	RAdam	AdamW
70.08	70.23†	68.13†	68.23†	63.79† (66.54‡)	65.99	67.62‡	67.93†

表一、 不同優化器的準確率比較[6]

伍、 分析與結論

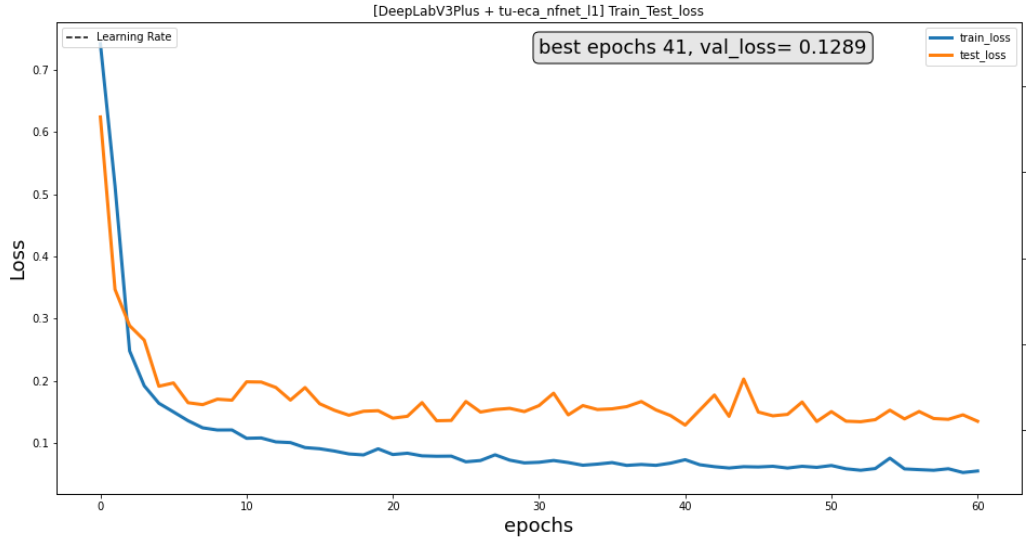
下列圖八到十為三個模型各自訓練時的損失函數圖，每次訓練大約跑 60 個 Epoch。

1. ECA-NFNet-L1 + PAN



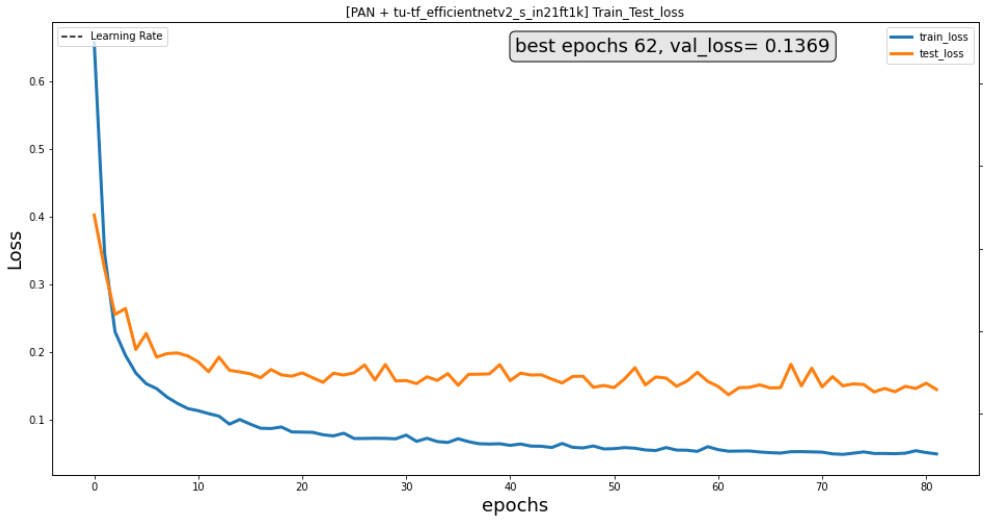
圖八、 ECA-NFNet-L1 + PAN 損失函數圖

2. ECA-NFNet-L1 + DeepLabV3+



圖九、 ECA-NFNet-L1 + DeepLabV3+ 損失函數圖

3. EfficientNetV2-S + PAN



圖十、 EfficientNetV2-S + PAN 損失函數圖

由於老師上課所提供的 NVIDIA 伺服器有時間上限，因此 Epoch 次數並未設太高。但由上圖看出，在訓練約 60Epoch 時，損失函數便已收斂，且訓練損失(Training Loss)及驗證損失(Validation Loss)接近平行，代表沒有發生過擬和(Overfitting)的現象。而在比賽後期拿到 TWCC 運算資源時，也曾經將每個模型的 Epoch 提高，但最終並未得到更好的結果。因此在這次試驗了解到，並非設置越多 Epoch 就一定能得到更好的結果。

各模型及經過集成投票後的 Public Score 如表二，可看出經過集成投票後，分數是有提升的。代表當我們結合不同弱分類器時，使用集成投票的方式能夠汲取各模型的優點，在表現上是有明顯的提升。

Model	ECA-NFNet-l1 PAN	ECA-NFNet-l1 DeepLabV3+	EfficientNetV2-s DeepLabV3+	Ensemble
Public Score	0.90	0.91	0.91	0.92

表二、 各模型及集成投票後的 Public Score

在使用 TWCC 運算資源的過程中，由於記憶體上限變高、無時間限制，因此也使用每個模型更進階的版本，例如 ECA-NFNet-L3、EfficientNetV2-M，但並未得到最好的結果。猜測的原因為這些模型是為了分割較大物件，例如：車、人物…等物件，而本次競賽區分的為尺度較小的癌細胞，因此即使使用更高階的版本也無法明顯改善模型表現。

優化器最終使用的為 AdaBelief，在一開始使用 NVIDIA 伺服器時，有效的減少達到收斂所需的時間及提高訓練分數。但在使用 TWC 資源時，並未比較 AdaBelief 與準確度較高但收斂較慢的 SGD。若未來有機會使用較好資源運算時，可以嘗試比較兩者的訓練時間與準確度。

結論

在這次比賽中使用的模型皆為預訓練模型，不但節省訓練時間，也能得到不錯的表現。而每個模型訓練時，在 60 Epoch 時損失函數便已開始收斂，因此在訓練中可以觀察開始收斂的訓練次數，可以避免訓練次數多而得不到表現顯著改善的結果。另外，選擇分類器時需要考慮任務的目的，因此並非使用參數越多的模型就一定能達到更好的效果。訓練完的模型可以利用集成投票的方式汲取各自的優點，且在最終結果上也有明顯的改善。最後，使用 AdaBelief 這個優化器對於有限的運算資源時，可以節省運算時間，並達到很好的準確率。

陸、雲端使用(建議至少一頁，若未提供則無法參加雲端運算應用評審獎的評核)

柒、 程式碼

Github: https://github.com/nicochang18/AICUP_STAS_II

捌、 使用的外部資源與參考文獻

[1]SMP

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

[2]NFNet

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.

[3]EfficientNetV2

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

[4]PAN

Li, H., Xiong, P., An, J., & Wang, L. (2018). Pyramid attention network for semantic segmentation. arXiv preprint arXiv:1805.10180.

[5]DeepLabV3+

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).

[6]AdaBelief

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.