元智大學資訊管理學系
第三十屆專業實習報告

公司代號:A

實習單位:中央研究院 數位文化中心

輔導老師:陳志成教授

姓　名:吳孟儒

學　號:1111709

目錄

零、簡介

本報告用於期末實習報告，將會詳細描述並說明學生吳孟儒在本次實習中於2025年2月開始至11月為期9個月的實習內容。

此次實習於中央研究院-數位文化中心，工作組別為AI組，主要參與研發中央研究院數位文化中心的古籍OCR系統。系統上線於中央研究院文字辨識與校對平台[https://ocr.ascdc.tw/]，平台提供合作的學術機構快速、精準且便利的文本辨識，辨識範圍覆蓋17380字，包含簡繁體及各種異體字，為數位典藏、古籍數位化及數位化人文研究提供幫助，也有日文、草書、越南漢字等跨國合作項目。

因為目標之古籍包含大量古語及符號、文字等，中文任務上類別高達17380類，越南漢字更是高達35000類，遠遠高於目前市面上OCR的覆蓋度，龐大的類別數量，再加上常用字與古字數量分布懸殊，導致資料集的建構相當困難。長尾訓練以及模型驗證使這項任務充滿挑戰。

整個OCR系統可以劃分成多個部分：從GAN生成資料開始，Text Detection，區塊排序，Classification，一直到LLM修字。每個區塊由不同組別負責，本次主要負責的任務是OCR的核心任務：分類任務，工作內容主要負責文字分類模型的開發。

目前處理的語言種類眾多，包含中文、中文草書、日文、喃字、標點符號，同時也有提供客製化模型的需求。

(喃字因為目前尚不公開，因此將不做說明。)

壹、工作內容

一、工作環境介紹

1. 個人電腦：每位工讀生都有自己的工位以及一台文書電腦。

2. Sever端VM：所有Code 的編譯以及執行都以遠端的方式連到Sever端以Linux系統運行，每個人都有分配到的VM，可以透過SSH以及VS Code內建的SSH功能進行操作，並且可以開多個Container來控制環境。

3. 硬體設備介紹：我個人分配到4張顯卡，每張顯卡的CPU及RAM效能從Sever上劃分，並配有local端的SSD以及NAS共用的HDD。

   配置1：

   GPU：GeForce RTX 4090 24GB

   RAM：128G

   用途：中文17380類訓練

   配置2：

   GPU：GeForce RTX 3090 24GB *2

   RAM：128G

   用途：日常實驗

   配置3：

   GPU：GeForce RTX 1080 Ti 11GB

   RAM：128G

   用途：小模型訓練以及檔案處理

二、工作詳述

1. 進度會議：

每兩周一次固定的進度會議，需要準備簡報報告兩周的工作進度，主管會給予大概的建議或者分配新的任務。

2. 工作日誌：

每天工作都需要寫工作日誌，大概說明當天的工作內容以及進度，也可以是閱讀文獻時的筆記，週會報告時需要說明。

3. Fine-tune任務：

有史語所、戶簿所等合作，需要提供不同的客製化模型。會先收到一筆人工標記的資料，交由我手上後進行資料的處理，模型fine-tune的規劃及執行。

這個過程中主要考量的是fine-tune用的資料分布不均所導致的災難性遺忘問題，透過調用不同的資料，並且設計不同的Augmentation來平衡資料。也需要對資料做前處理來保證訓練資料的乾淨。

Fine-tune後的模型打包成 .onnx 的模型檔案交出。Fine-tune後的模型會用於新的一輪資料採集，並再一次取得資料後進行下一輪Fine-tune。

4. 新任務的模型建構：

對新的任務進行模型建構，如喃字及日文。從資料取得、生成開始，模型挑選、實驗，微調模型架構，數據分析、觀察。到跌代優化模型能力，閱讀論文尋找新的解決方案。

基本上就是負責一個小專案的進行及規劃。

5. 日常工作：

針對已有模型進行優化，方式手段不限，可以自由發揮，主管通常鼓勵我們發揮創造力不斷嘗試，尋找不同可能性。

可以(包括但不限於)：閱讀新的論文並嘗試復現 、嘗試不同的資料分配方式(如:Cross Valid)、嘗試不同的Augmentations、進行不同的前處理、優化模型效率、優化模型架構、提出全新的解決方案等等……

三、實習期間完成之進度與學習

大部分的工作內容以實驗、研究為主。並且大多都包含了我的學習內容，可以說進度就是我的學習，因此這邊會包含大多部分在工作中的學習，並以模型的不同部分為主體，說明進行了什麼樣的研究以及優化，並且說明大致的思路以及結果。

(A)Datasets(資料集):

資料處理可以說是精進模型的核心，模型的能力很大一部分取決於資料的質跟量。有別於一般任務使用公開資料集，中文文字辨識任務中並沒有能夠直接使用的資料集能夠覆蓋我們的範圍。因此建構資料集，並且有效的使用手上的資料是一大重點。*避免篇幅過長，以下僅以中文任務為主說明

資料的來源大概有幾種:

(1)字型檔生成資料

利用PIL中的ImageFont、TTFont等套件搭配tcc字型檔，並讀取pkl檔中整理好的text list來繪製字體，這個過程中因為需要繪製將近100萬張圖片，因此搭配multiprocessing來進行多行程處理。最後利用cv2來進行圖片的裁切、置中等處理。

圖:程式示意圖

```
def main():
    t0 = time.time()
    pool = mp.Pool(16)
    pool.map(generateImages, font_paths)
    print("-------------------")
    t1 = time.time()
    print("Computer processed in %f seconds" % (t1-t0))
```

圖:繪製圖片示意圖

(2)GAN生成資料

字型黨生成的資料並不能完整覆蓋所有字，因此我們使用GAN模型來補全字型黨無法生成的生僻字，GAN的model由其他人製作，這邊僅負責運用。

對於越南漢字這種幾乎只出現在古籍裡的字有很好的效果。

(3)真實資料

真實資料是真實從古籍上擷取下來的資料，是最珍貴的訓練及驗證資料。包含7000多個類別，大約100萬張圖片，但是資料的品質參差不齊，錯誤標助的資料也存在不少，最重要的是分布不均。頭部的1500左右的常見字佔據了資料集的80%，而尾部的資料則是不到5張，甚至有10000個類別是直接沒有。

(4)公開資料集

不用於訓練，不說明。

資料集規劃及擴增:

以上大體就是目前所有的資料來源，資料依任務目前分為:中文、中文草書、日文、越南漢字、標點符號，資料構成基本一致。這段期間我嘗試了很多種不同的資料集劃分方式。需要劃分出訓練集、驗證集、測試集，最大程度的提供訓練集資料的同時，確保測試集中的資料能夠真實反應模型能力。下面說明後續資料集的劃分策略。

Testing data:

中文的部分會陸續收到小批量的資料作為fine-tune客製化模型用。這些資料通常量比較少，並且字體與訓練用的真實資料差異較

大，因此不混入訓練資料，而在主模型訓練中作為測試資料來評斷模型的范化能力。

在此其中，標點符號考慮到對於分類模型較為簡單，真實資料不用於訓練，全部用於測試資料。越南漢字則因為真實資料量極少，對於訓練幫助極小，則僅用於測驗。

Valid data:

驗證資料有兩種:

- 從訓練用的生成資料中劃分:

  從生成資料劃分，參考價值並不大，主要用來檢查是否over fitting。每個類別最多劃分5筆資料。

- 從訓練用的真實資料中劃分:

  劃分真實資料的方式主要以資料均勻為主，避免個別類別能力過強影響整體。

  劃分時單類別最多劃分5張，並且訓練資料中類別資料數量小於5後不在劃分。用於確認模型對真實資料的學習。

Training data:

訓練資料的部分我以字形檔生成資料為基底，以GAN生成資料為輔助將資料補齊。然後最多混入50張真實資料。

Augmentation

資料擴增的部份，我對每一種資料重新設計了個別的Augmentaion，包含侵蝕、膨脹、二值化、噪化(salt and paper)、仿射矩陣、高斯模糊等等多種擴增方式，並且使用imguag搭配cv2來完成。

真實資料因為資料常有破損且解析度低，實驗中發現侵蝕膨脹、二值化等比較激進的操作，極為容易對資料集造成汙染，因此Aug的

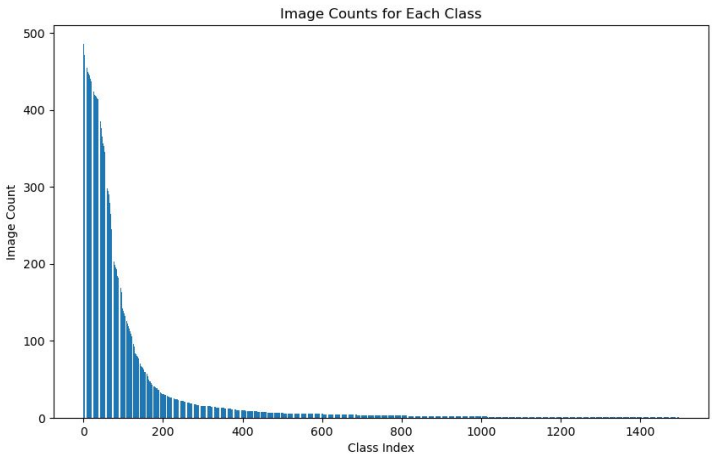
幅度會保守一點，並且並不加入侵蝕膨脹等操作。生成資料則因為解析度高且不會有破損的情況，因此Aug的幅度會比較激進。

資料集的問題:

資料集中最大的問題在於長尾訓練，前面也提到了因為常用字與生僻字極大的分布差異，導致了訓練資料不平衡的問題 : The long tail effect

這個問題主要出現在真實資料上，雖然使用生成資料並且減少頭部的資料可以適當的平衡這個問題，但是純生成資料的類別訓練出來泛化能力就是差很多。

長尾訓練會使得模型對頭部過於自信，且權重極高。對尾部的學習幾乎不理不睬，即使透過調整尾部權重或是資料擴增等方法來平衡，反而容易對尾部的學習過強，且尾部資料不足而導致over fitting。



Long Tail實驗:

關於這個問題[1][2]中也有提到，兩者的思路很相似，都是將模型拆分成Feature Extractor 跟Classifier兩個部分，為Feature

Extractor提供最大量的資料，訓練更好的表達能力，為Classifier提供最平衡的資料，確保平衡的分類能力。後者的思路更簡單明瞭，可以簡單理解成先用原本沒有平衡過的資料做一次訓練，在凍結Feature Extractor單獨對Classifier用小部分的資料訓練，有一點Fine-tune的味道在。

在10中旬開始進行這個實驗，我在Unet上將生成資料與多餘的真實資料不管平衡直接投入訓練，並且凍結Head後使用真實資料(無真實資料則用生成資料填補)fine-tune分類層，目前還在實驗中，結果暫不說明。

(B)Optimization(演算法或稱優化器):

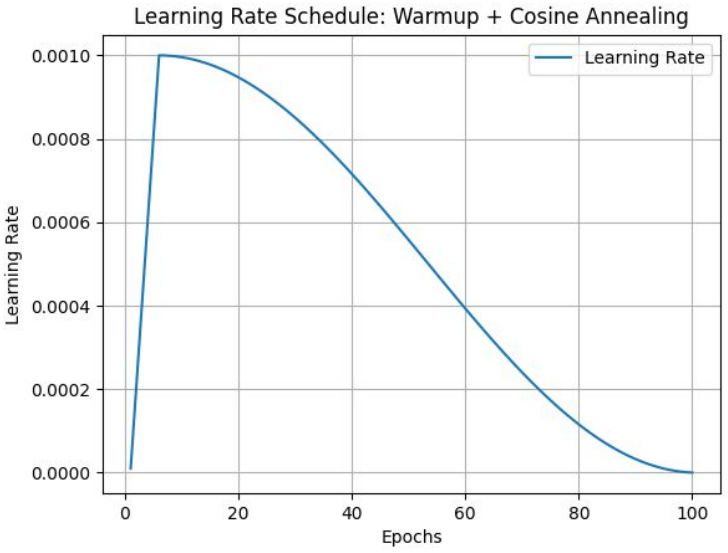
目前主流的作法有：SGD、AdamW

大致上現在的分類模型訓練上使用都以AdamW為主，但深入研究後，可以觀察到，因為資料集以及模型的體量不斷擴張，大部分的研究開始往效率、速度為核心研究。而AdamW作為目前自適應Optimization的最優解，對於高參數量的模型有很強的效率優勢。並且對個別參數的學習率自適應能力使其根本上的解決了高參數模型中參數朝向不同的局部最優解優化時的分歧問題，因此而成為了現在幾乎是預設選項的優化器。

然而容易忽視的是SGD在尋找平滑最優解時其實是同樣優秀的選擇，在[3]中提到在一個局部定義的區域中SGD能夠相較於Adam更高效率的逃脫，也就不容易陷入Local minima，這是因為SGD梯度噪聲的更傾向於Heavy-tail的分布，使得SGD在尾端時更容易出現large jump。而不論是Adam或是AdamW，因為參數個別學習率的過度優化，雖然在遇到Saddle point時可以透過累積單一軸的動量來逃出這個梯度接近0的點，但是一旦遇到一個很深的Local minima時會義無反顧地栽進去。而且一旦進去就會持續深入，並且參數的動量會越來越小，也就越來越難以逃脫。此時雖然對訓練集跟驗證集的效果可能會很好，但是對於測試集的泛化能力就會快速的下降。

結論是AdamW在高參數模型上的效率很高，但容易陷入局部最佳解，導致模型更容易出現Overfitting。尤其在小模型或較容易擬合的模型上更為容易發生。而SGD雖然效率較低但更能夠找到平滑最優解，也就是泛化能力更強。

綜上所述，我採用了AdamW、SGD的接替做法，AdamW會參與前面80%的Epoch，如果訓練超過設定最多Epoch的80%且模型還沒有完全收斂的話，SGD會接替，並完成訓練。如果未達80%就已經收斂SGD則會直接接替直到模型再次收斂或者達到上限。學習率的部分AdamW會有一個Warm up的過程，從設定學習率的-1次方開始攀升到設定學習率再開始曲線下滑，而SGD接替時會直接接替學習率。如圖:



實驗範例:

驗證集:

Ch-real-mix-valid、ch-real-twbook-aug-limit100

測試集:

Ch-real-hre-limit100

這是一個在EfficientFormer上不凍結backbone的fine-tune的範例(fine-tune在後續會提到)，可以看到在80後的epoch明顯對測試集跟驗證集的泛化能力有很大的提升。

在EfficientFormer與Unet的實驗中，AdamW跟SGD輪替的方式都起到了很大的作用，整體對測試集的泛化能力提高了5%上下。

(C)Loss Function (損失函數) :

損失函數的部分，為應對長尾訓練，本專案已經採用Balanced Binary Cross Entropy，並且對Binary Cross Entropy以及一般的Cross Entropy都有過實驗。

在此之上我使用了Label Smoothing，這個作法將原本的one-hot Label:

$$\begin{cases} 1, if\ (i = y) \\ 0, if\ (i \neq y) \end{cases}$$

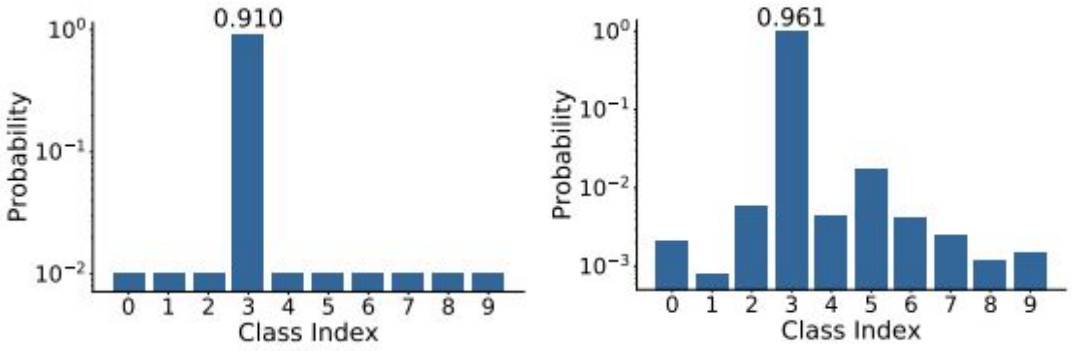
加上noise變成:

$$\begin{cases} (1 - factor), if\ (i = y) \\ \dfrac{factor}{Total\ categories}, if\ (i \neq y) \end{cases}$$

這樣的改變避免了模型在單一類別上過度自信，參考[3][4]，其中也說明了Label Smoothing 對分類模型泛化能力的提升。

並且，我認為可以利用中文相似字中的相似性，對於Label Smoothing做出變化，也就是根據類別的相似度分配soft label。

左圖:一般的Label Smoothing 右圖:根據類別不同分布



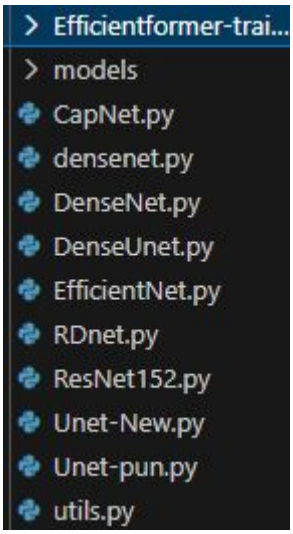
論文[5]中初步實驗了這樣的可能性:

Algorithm 1 The pipeline of the proposed OLS

Input: Dataset $\mathcal{D}_{\texttt{train}} = \{(\boldsymbol{x_i}, y_i)\}$, model f_θ, training epochs T
Initialize: Soft label matrix $\boldsymbol{S^0} = \frac{1}{K}\boldsymbol{I}$, $\boldsymbol{I}$ denotes unit matrix, K denotes the number of classes
for current epoch $t = 1$ **to** T **do**
  Initialize: $\boldsymbol{S^t} = \boldsymbol{0}$
  for $iter = 1$ **to** $iterations$ **do**
    Sample a batch $\mathcal{B} \subset \mathcal{D}_{\texttt{train}}$, input to f_θ
    Obtain predicted probabilities $\{f(\theta, \boldsymbol{x_i}), \boldsymbol{x_i} \in \mathcal{B}\}$
    Compute loss by Eqn. (4), backward to update the parameter θ
    for $i = 1$ **to** $|\mathcal{B}|$ **do**
      Update $\boldsymbol{S^t_{y_i}} \leftarrow \boldsymbol{S^t_{y_i}} + f(\theta, \boldsymbol{x_i})$
    end for
  end for
  Normalize $\boldsymbol{S^t}$ at each column
end for

根據這篇論文提供的Code，我完成了這個變體的實裝，但是實驗中發現，因為類別數量高達17380類，文中的做法導致運算資源的需求因為類別數量急速攀升，且對訓練的效率產生了很大的影響。一輪訓練的時間幾乎從一周延長至三周。

後續將等到異體字的統計資料到手，以改善這樣的問題。



(D)Model(模型)

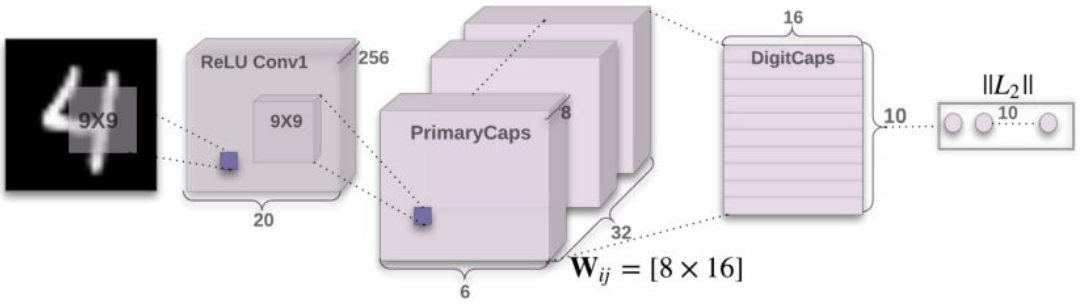
在這段期間，我進行了多種模型的實裝與研究。包含CapsuleNet、ResNet、DenseNet、DenseUnet、Unet、EfficientNetV2、EfficientFormerV2。

我會先簡單介紹這些模型的架構、概念跟特色，並且在最後介紹在這些模型上的嘗試。

(1)CapNet(膠囊網路)

CapsuleNet又名膠囊網路，由AI教父Geoffrey Hinton提出，理念是希望能夠解決CNN的對空間關係與方向上的短版。原理也很簡單，CNN的捲積網路在擷取特徵後變成一個scalar，只能記得這個特徵

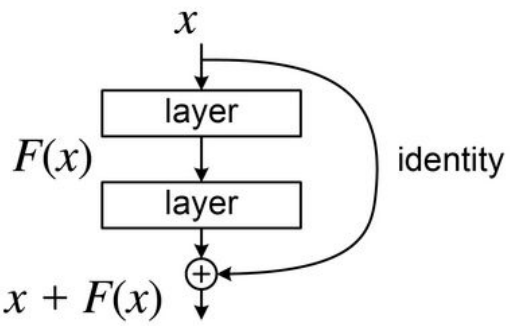
是什麼。而膠囊網路則是直接將這個捲積看到的東西拉直變成一個向量，透過向量的方向、大小、位置來記錄物件的空間關係。



理論上這個網路對於文字的處理能力可能會很不錯，因為很多古字或異體字的部首換個位置或換個方向就完全不是同一個字。但是因為最後的DigitCaps會因類別數量提升而擴大，導致模型占用的VRAM會跟著類別數量上升而上升，這也是為了保留空間關係而導致的問題。

(2)ResNet(殘差網路)、DenseNet(稠密網路)

ResNet是目前深度學習領域中最具影響力的卷積神經網路架構之一，其核心理念可以用一句話概括:"深，還要更深"。一般的卷積神網路在深度增加時，會遇到梯度消失、梯度爆炸以及訓練誤差反而升高的現象，使得模型無法有效學習。但 ResNet 透過在每個 Block 中加入一條簡單但有效的 skip connection，解決了這個問題，從而開啟了[深]度學習的時代。

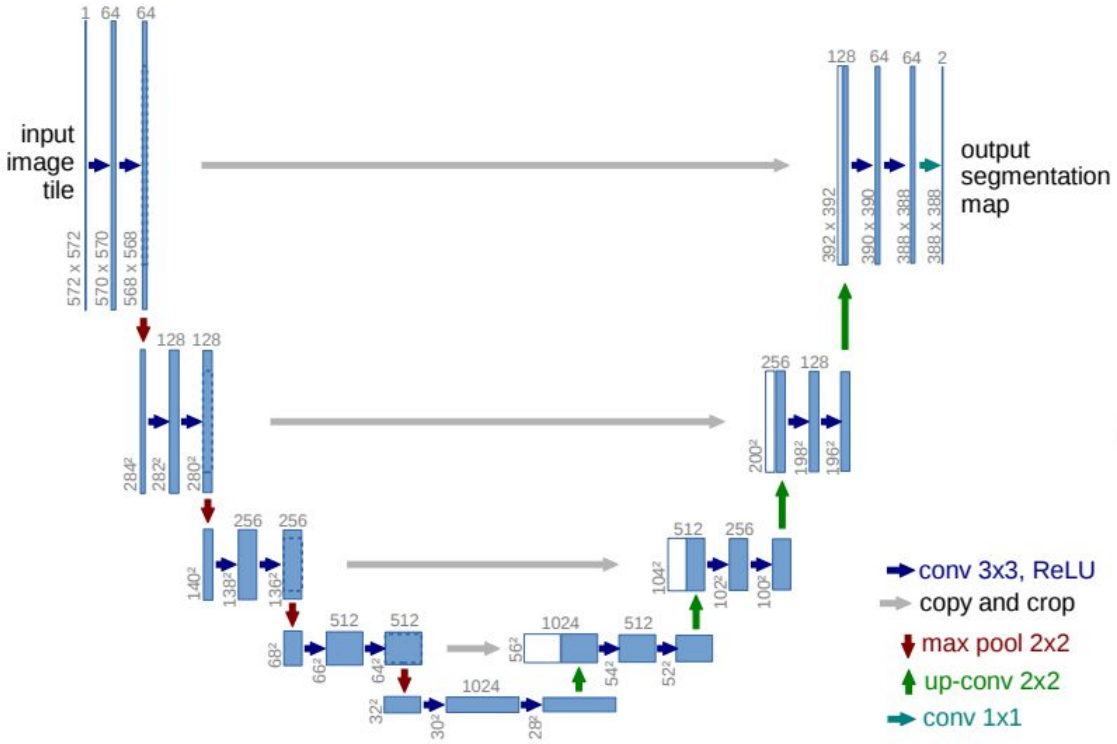


ResNet甚至可以訓練到1202層這種誇張的數字，而後續的幾乎所有模型也都引用了這樣的設計。

隨後還有DenseNet透過concat將不同維度的特徵結合，來提高對不同維度特徵的重視程度，在Segmentation的任務上作為Unet的Backbone有很好的效果。

(3)Unet

關於Unet的這個模型我覺得其實是最有意思的，Unet作為一個醫療用Segmentation模型被廣為人知，其模型的架構就像一個巨大的U形結構，因而得名Unet



Unet可以理解為一個Encoder、Decoder的概念，由編碼器及解碼器組成，編碼器負責提取輸入資訊的高層次特徵，並壓縮資訊。解碼器負責將特徵恢復成原本的圖片。訓練過程中編碼器需要學會如何將圖片轉換成一個僅包含重要資訊的特徵值，而解碼器需要學會如何將這個特徵值還原成我們要的圖片。這個過程跟最早期的GAN有一點相似，但是GAN要的是Decoder的部分產出隨機的圖片。而Unet則規範這些圖片的基礎以及生成圖片的樣式。

使用Unet作為分類實驗室上一屆學長提出的方案，他認為decoder的結構有助於監督encoder結構的學習。而這樣的方式也許可以強迫encoder在訓練時不會丟失一些重要的細部資訊，並且decoder出來的圖片還可以做為評估模型能力的指標。而我接續了這項實驗。

後續實驗中以基礎的Unet架構進行實驗，也證實了這個有趣的想法，有加入decoder比沒有加入decoder(純CNN)的模型收斂更快，效率更高，並且對細微差異的觀察能力更強。並且對於日文、標點符號一類圖像結構簡單的任務跟EffiecientFormer一類TransFormer結構的能裡表現相像。

(4)EfficientNetV2

EfficientNetV2 是 EfficientNet 系列的後續改良版，整體的設計核心仍然圍繞在「模型效能與運算成本之間取得最佳平衡」這個理念上，但相較於第一代，它在訓練速度、記憶體使用率，以及對不同資料集的適應力上都有明顯提升。

整體來說 EfficientNetV2 的訓練速度比 V1 快上許多，尤其在大 Batch 或大型資料集時效果最明顯。在實際實驗過程中，特別是在中文 OCR 這種大類別、多樣化影像樣式的任務上，EfficientNetV2 的 Feature Extractor 有相當穩定的表現。

(5)EfficientFormer

EfficientFormer 是一個以輕量化為目標設計的 Vision Transformer，其核心理念是在保持 Transformer 架構特性的前提下，大幅降低計算成本。它的主要特點是將大部分的 early stage 改以 MLP 與輕量卷積進行特徵混合，而不是使用計算量較高的 self-attention。只有在最後的 stage，模型才加入真正的多頭自注意力，以補足 global information 的建模能力。

這樣的設計使 EfficientFormer 在速度與精度之間取得平衡：前面階段運算便宜且容易在低效能硬體上優化，後段再利用 attention 補強語意表達。整體來說，它適合在運算資源有限、但仍需要 Transformer 架構特性的場景中擔任 backbone，例如 mobile vision 或即時應用。

(E)模型訓練實驗:

在模型實驗上我們進行了很多嘗試，以下僅呈現目前可以公開的部分，並且只有中文模型的部分。

在CapsuleNet模型測試中，我嘗試實作了這個模型並於中文資料集上實驗，但是對VRAM的佔用率遠超我的預期在Batch size=10的情況下幾乎把4090 24GB的VRAM吃滿。並且單一epoch的訓練時間就要23個小時，最後放棄了這個嘗試。

在Unet的架構上我嘗試了將backbone替換成DenseNet(DenseUnet)，並且跟ResNet-152、Unet以及DenseNet在同樣(64*64)條件下比較。下面是目前各模型的最佳結果。

最後對於中文真實資料驗證集的ACC:
Unet : 0.9654
Unet(去除Decoder) : 0.9168
ResNet-152 : 0.9536
DenseNet : 0.9438
DenseUnet : 0.9589
可以看到Unet單純的架構便能夠高於ResNet、DenseNet。並且訓練效率以及模型大小也遠勝於其他模型。

而後EfficientNetV2上我們使用論文中Midium Size的模型，並且以(224*224)為大小訓練。EfficientFormerV2也使用論文中S2的架構，以(224*224)為大小訓練。
最後對於中文真實資料驗證集的ACC:

EfficientNetV2 : 0.9631

EfficientFormerV2 : 0.9655

總結:

目前中文訓練的部分，我們認為，因為中文字體的結構相較於大型圖片簡單，並且訓練資料的問題，更寬更深的模型不一定是更好的模型，反而使模型的泛化能力不佳。

並且EfficientFormerV2因為其TransFormer的架構，在更大且解析度更高的圖片上效果更好。

四、工作當中扮演的角色

在這次實習中，我作為一名實習生，獨自負責一項專案的管理。在此之中，主管只給予我們大方向的目標以及建議。我需要依靠自己的能力尋找方向並且進步。

貳、自我評估與心得感想

在這次的實習中，祥安老師(主管)特別強調，認為中研院是一個研究機構，而我們的工作則是在中文古籍OCR上取得不一樣的進展，這個進展不一定要是大步向前邁進，不斷的嘗試、不怕錯誤，並且尋找可能存在的路徑是我們的任務。也因此工作中，往往都是一個大方向的目標，並且不會有明確的指示。

在這個過程中往往會遇到毫無頭緒的時候，沒有人可以直接告訴我正確的做法，也許也不存在正確的做法，我的學長稱這個階段為"無頭蒼蠅階段"，當任務推進到一個程度，要開始想辦法改善的時候總會遇到這麼一個瓶頸期。能做的事情就是觀察、嘗試，並且學習，通常都是隨便點開一篇標題引人注目的論文，然後延伸閱讀。也不知道有沒有幫助，套用學長的話"不知道怎麼辦，讀篇論文壓壓驚"這種沒有方向的漫無目的也許是這個工作最令人緊張的一點，要使模型進步有無數種可能的方向，但並沒有寫在課本上，ChatGPT也不會告訴我答案。

在這樣的實習中我也漸漸找到自己的腳步，以這次標點符號的實作為主，我踏實的從資料集開始做，對資料作分析、擴增。一步一步將模型建立，然後循環式尋找提升的空間，這種感覺就像在玩遊戲練等一樣，逐漸升級提升戰力。不一樣的是經常會倒著升級…但是馬上讀檔進行下一次嘗試這個過程，也越來越有意思。

整個過程我學到非常多東西，不論是實作上、理論上、還是觀念上的。如何作為一個研究者穩步提升自己的實力，是我最大的收穫。即便還有很多不成熟的地方，但我會繼續努力。

引用

[1]KANG, Bingyi, et al. Decoupling representation and classifier for long-tailed recognition. *arXiv preprint arXiv:1910.09217*, 2019.

[2]Zhang, J., Liu, L., Wang, P., & Shen, C. (2024). *To balance or not to balance: A simple-yet-effective approach for learning with long-tailed distributions*. arXiv:2402.17191.

[3]Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). *Rethinking the Inception architecture for computer vision*. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 2818–2826).

[4]MÜLLER, Rafael; KORNBLITH, Simon; HINTON, Geoffrey E. When does label smoothing help?. *Advances in neural information processing systems*, 2019, 32.

[5]ZHANG, Chang-Bin, et al. Delving deep into label smoothing. *IEEE Transactions on Image Processing*, 2021, 30: 5984-5996.