ML Final Report - Human Protein Atlas Image Classification

劉家豪 B04504042
蘇軒 B04203058
陳柏瑞 B04901143

Introduction

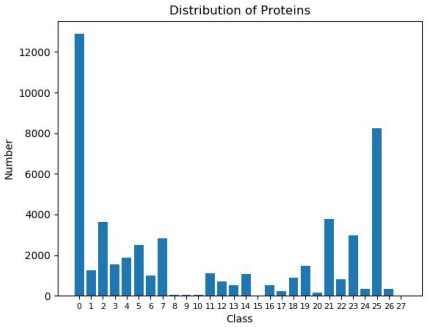
Task description : 這次的 Task是要找出一個適合的 model 去做蛋白質圖片的分類，每張圖片中會有1到4種蛋白質不等，model必須能分辨出一張圖片中包含的所有蛋白質。

Data description: 總共有28種蛋白質，標示為0-27，所有 microscope images samples會先經過四種 filters: the protein of interest (green) plus three cellular landmarks: nucleus (blue), microtubules (red), endoplasmic reticulum (yellow).四張圖片代表一個蛋白質的data。

在訓練的過程中，會遇到Imbalanced data的問題，也就是部分data佔了整體data的大多數，而少部分的data數量很少，以二元分類為例，如果兩個 class 中 class 0占了百分之95.5，而 class 1 占了百分之0.05這樣就算model直接把所有 testing data 都預測成 class 0 這樣 accuracy 也高達0.955 如此將會造成嚴重誤差。本次data 蛋白質種類數量分布如下圖與下表，其中class0, 2, 21, 25 佔了大多數，需要對剩下的class做特殊的處理，在Motivation 中會提到。

本次選用pytorch實作，用已經寫好的pretrained model - bninception，與kaggle中找到的其中一個kernel(連結付在reference)作為基礎，自己加上augmentation與over sampling來增加model的準確度。

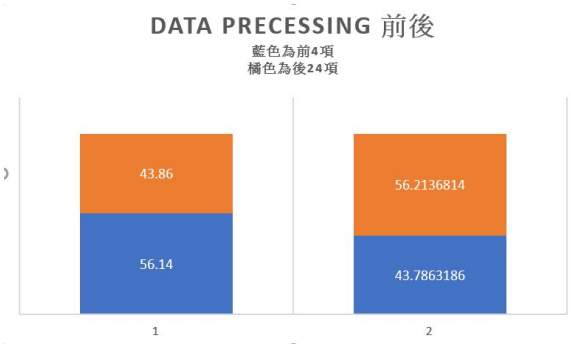
id	count	%	id	count	%	id	count	%	id	count	%
0	12885	25.37	7	2822	5.55	14	1066	2.10	21	3777	7.44
1	1254	2.46	8	53	0.10	15	21	0.04	22	802	1.58
2	3621	7.13	9	45	0.09	16	530	1.04	23	2965	5.84
3	1561	3.07	10	28	0.06	17	210	0.41	24	322	0.63
4	1858	3.65	11	1093	2.15	18	902	1.78	25	8228	16.20
5	2513	4.94	12	688	1.35	19	1482	2.92	26	328	0.65
6	1008	1.98	13	537	1.05	20	172	0.34	27	11	0.02



Data Preprocessing/Feature Engineering

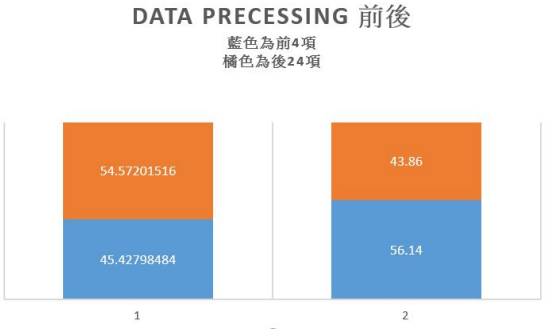
為了避免Imbalanced data造成的預測誤差，處理方法通常有種，第一是複製或重組現有的數量較少的class 的圖片以增加這些class的比例，第二是刪除部分數量較多的class的圖片，第三是使用不受數量影響準確性的training 方式，例如decision tree，第四是調整在model output的activation function的判斷值，將data較多的class的threshold調大，model 預測出來要更確定是屬於這個class才能做出預測。我們嘗試的方法是第一種、第二種與第四種，跟data processing 有關的是第一及第二種。

第一種方法嘗試增比例低的class的數量。我們除了原本給定的512x512的31072張圖外，又下載了kaggle中有提到的2048x2048的data 82374張，總共113446張來增加數量，又把數量較少的class{ 1, 3, 4, 6, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 22, 24, 26, 27}的數量透過複製來增加，原本，前四多的class{ 0, 2, 21, 25 }加起來超過55%，我們希望複製data後能讓前四多的class下降到40%，並大幅提升原本只有0.01~0.1的class，再對所有的training data做augmentation，最後我們的classs比例如下圖：



Class	Original %	Sampling %
8	0.10	0.77
9	0.09	1.47
10	0.06	1.4
15	0.04	0.39
27	0.02	0.55

第二種方法是將data中標記為0與標記為25的data隨機刪去一部分，為了不要刪到包含其他標記的data，我們取出只有標記0與只有標記25的data刪除，刪除後各個class比例如下，training 結果會在expirement and discussion 中討論：

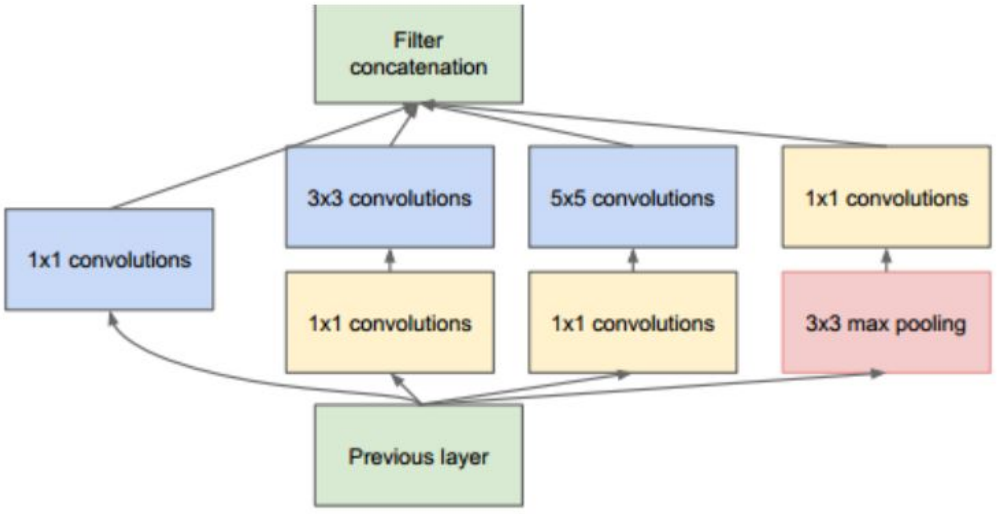


Class	Original %	Sampling %
8	0.10	0.82
9	0.09	1.58
10	0.06	1.51
15	0.04	0.42
27	0.02	0.6

Model Description

Model1 :

我們在model結構上，一開始使用了 pytorch 內建的 pretrain model(bninception) 來當作我們model 最一開始的架構。



上圖為一個基本incpetion的架構，因為傳統的 Conv Net 是將 Convolution Layer stack 再一起， 而 Incpetion Layer 最大的改變就是將Convolution Layer用不一樣甚至平行的方式堆疊在一起，形成一個稀疏的結構，從上圖可以得知 Inception 分別將 1x1, 3X3, 5X5 的 conv 和 3X3 的 pooling stack 一起，一方面增加網路的 width， 另一方面增加網路對尺度的適應性，主要的特點是提高網路內部計算資源的利用性。

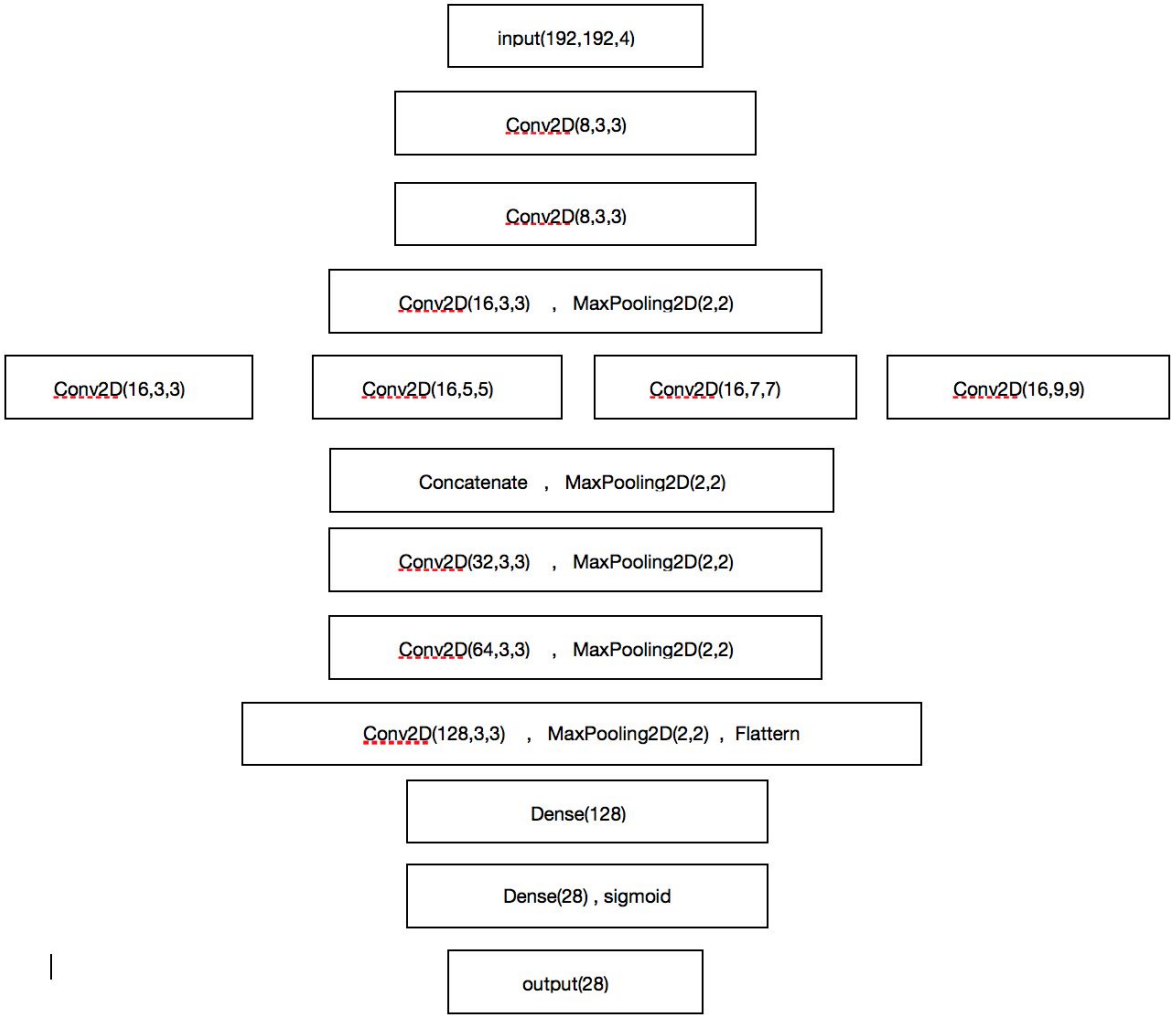
至於這次所使用到的 Bnincpetion 則是將多個 Inception Layer 堆疊再一起，並且加上了 Batch-Normalization， 架構如下圖:

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	double #3×3 reduce	double #3×3	Pool +proj
convolution*	7×7/2	112×112×64	1						
max pool	3×3/2	56×56×64	0						
convolution	3×3/1	56×56×192	1		64	192			
max pool	3×3/2	28×28×192	0						
inception (3a)		28×28×256	3	64	64	64	64	96	avg + 32
inception (3b)		28×28×320	3	64	64	96	64	96	avg + 64
inception (3c)	stride 2	28×28×576	3	0	128	160	64	96	max + pass through
inception (4a)		14×14×576	3	224	64	96	96	128	avg + 128
inception (4b)		14×14×576	3	192	96	128	96	128	avg + 128
inception (4c)		14×14×576	3	160	128	160	128	160	avg + 128
inception (4d)		14×14×576	3	96	128	192	160	192	avg + 128
inception (4e)	stride 2	14×14×1024	3	0	128	192	192	256	max + pass through
inception (5a)		7×7×1024	3	352	192	320	160	224	avg + 128
inception (5b)		7×7×1024	3	352	192	320	192	224	max + 128
avg pool	7×7/1	1×1×1024	0						

有了pretrain-model後，我們簡單的在最後面加上了一層Convolution Layer 後就是現在的Model1 了。

Model2:

至於model 2 經過我們的討論之後，決定效仿 Inception model 一樣自己手刻出一個由 Conv Layer 並排堆疊而成並且具有寬度的網路架構來進行訓練，如下圖，我們先將 input 經過三個 Conlution Layer 後，得到的 Feature Map 分別通過四個 CNN 架構來針對每一個不一樣的 Feature Map 更進一步訓練出屬於自己的 CNN 最後將四個 CNN的output concate 起來後最後通過連續的CNN就是我們這次的 Model 了。



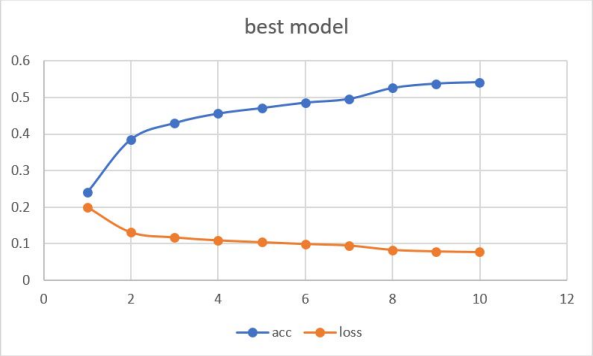
Experiment and Discussion

此小節分為兩個部分，第一個部分是我們最佳data的model結果與training時的accuracy, loss呈現，第二部分是我們達到最佳model前做的測試，總共有四種測試，分別是調整learning rate 、調整layer 深度、複製data調整各種class的在dataset 中的比例，刪減data set 中比例最高的兩個class其中的一些data與data set沒做任何改變做比較、調整predict時activation function threshold大小。

第一部分

public f1 score : 0.52868 private f1 score: 0.499

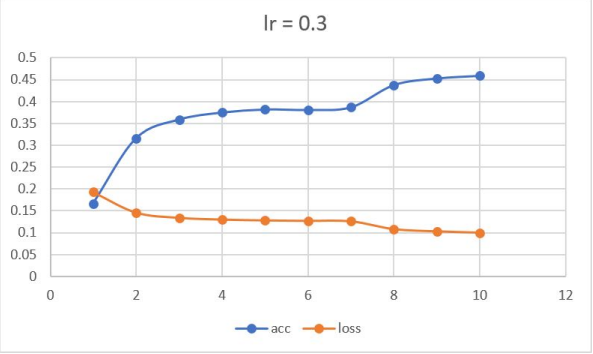
data precess: 複製數量少的class, 複製部分單獨標class0, class25的data，調整threshold

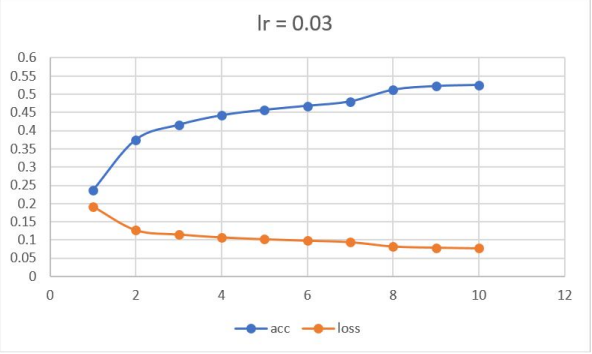


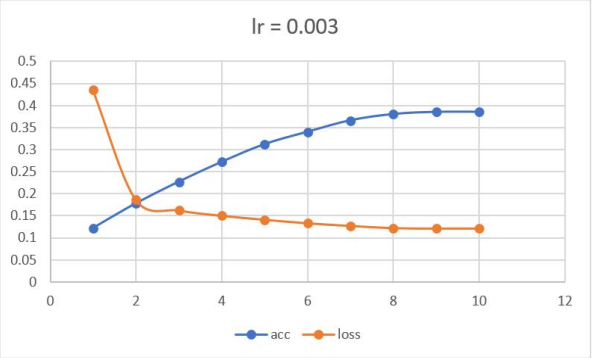
第二部分

1. 調整learning rate

我們將learning rate 調整為 0.3, 0.03, 0.003，可得到最好的learning rate應該是在0.03附近，所以之後training都採用0.03。







2. 不同深度之 model 比較

這個部份我們依照 Model Description 所敘述的兩個 model 進行實驗，我們將兩個 model分別用 externel data 去訓練，得到的結果如下:

	publice	private
bninception(pretrain)	0.501	0.479
Model2	0.091	0.075

從結果可以發現我們自己架的 model 2 效果遠比用pretrain model 的效果還要差的許多，經過我們討論後推估原因是，相較於 bnincpetion 的 pretrain model，我們自己架的 model 不管是深度或是寬度方面都小很多，使得它無法處理 imbalanced data 的問題。

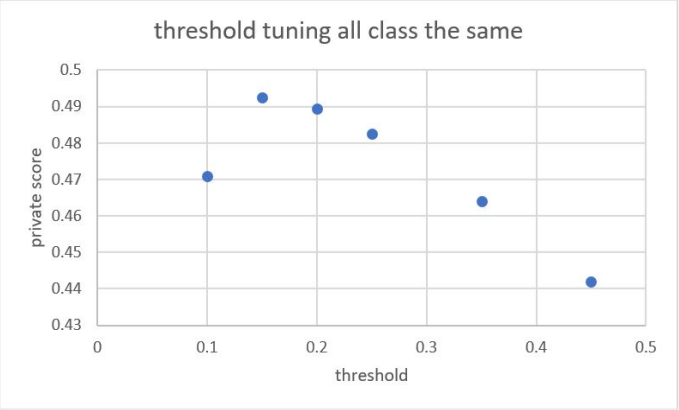
3. 複製data調整各種class的在dataset 中的比例，刪減data set 中比例最高的兩個class其中的一些data做比較

從下表可看出，在private上表現最好的是複製少量的class增加他們在data中的比例，而我們後來又再把單獨標0與25的data複製加入data中，對結果也有微幅的提升，可能是因為如此model能更準確地分辨是否是class0與class25。

	origin	copy data	delete data
public	0.53	0.52864	0.507
private	0.492	0.499	0.4785

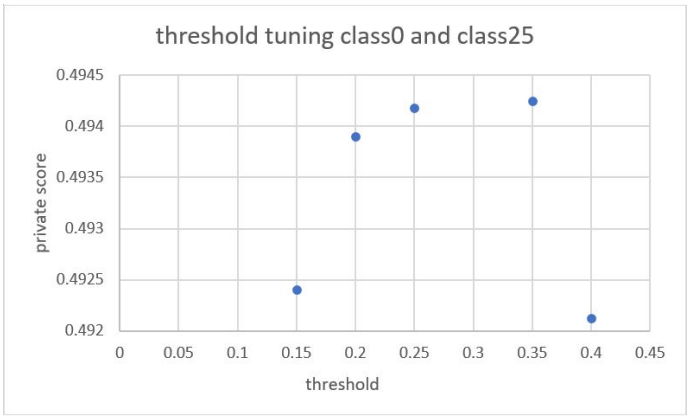
4. 調整predict時activation function threshold大小

exp1 所有class threshold 一起調整：(0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.45)



threshold	private score
0.1	0.471
0.15	0.4924
0.2	0.4895
0.25	0.4825
0.35	0.464
0.45	0.442

exp2 所有class threshold 維持0.15，調整 class0 class25 threshold(0.15, 0.2, 0.25, 0.35, 0.4)



threshold	private score
0.15	0.4924
0.2	0.4939
0.25	0.49418
0.35	0.49425
0.4	0.49212

由exp1結果可看出，所有threshold在0.15時能得到最好的score，這時候如果調整class0 與class25 能增加score，可能原因是dataset中大部分是class0 與class25，traint出來很容易判斷成這兩個class，這時候如果加大這兩個class的threshold 可以增加被判斷成0跟25的難度，所以f1 accuracy上升。

Conclusion

這次的 final project 是我們首次處理 imbalanced data、 multilabel classification 的問題，一開始我們針對資料量不平均這方面的問題下了很多功夫去研究也嘗試了很多網路上的方法，最後我們發現 over-sampling的效果最好，接著我們著手於bnincpetion 等相關pretrain 網路架構的研究，並且嘗試著模仿類似架構的網路並進

行訓練，可惜最後因為神經網路深度與寬度不夠導致效果不彰，經過這次的作業，我們發現在關 image multilabel classification 方面還有很多地方可以繼續努力，未來希望可以實踐看看之前在proposal提到的smote method 來處理 imbalanced data，而不是用比較基本的 under-sampling 或是 over-sampling， 另外在 model 方面我們未來會繼續嘗試看看是否有更好的model，並且試著從調整圖片下手或許可以增加 f1 score。

Reference

1. kaggle kernal(model 來源): https://www.kaggle.com/c/human-protein-atlas-image-classification/discussion/72812?fbclid=IwAR2qMwO4mplRTxowIGroe-8lSedKay9Jeiji6ugy7E6LmY5OrLHd00AvKEw
2. https://arxiv.org/pdf/1106.1813.pdf?fbclid=IwAR2LsjNE0gJzStoTZwoMRnjCqL1TQvfHSAtvksI62UjtES-cnyQw2NrVX_8
3. https://morvanzhou.github.io/tutorials/machine-learning/ML-intro/3-07-imbalanced-data/