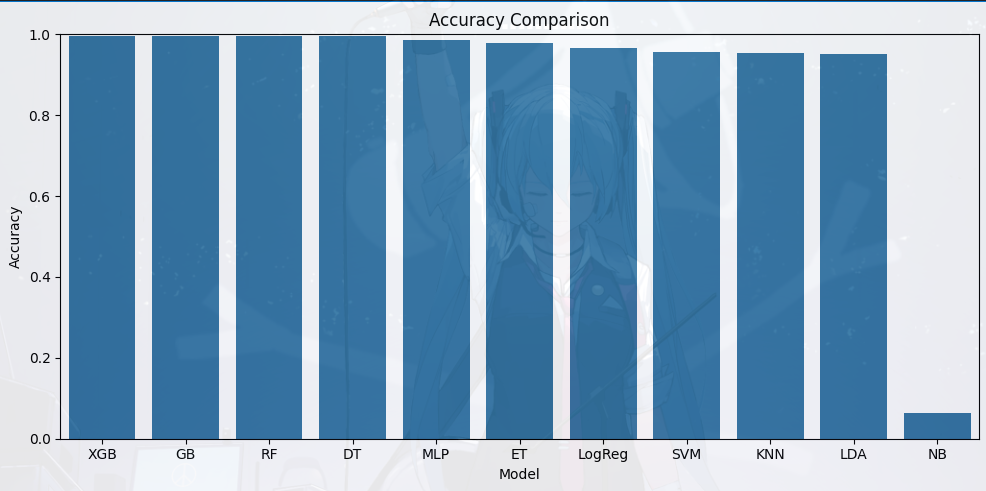
**海大資工AI機器學習作業報告**

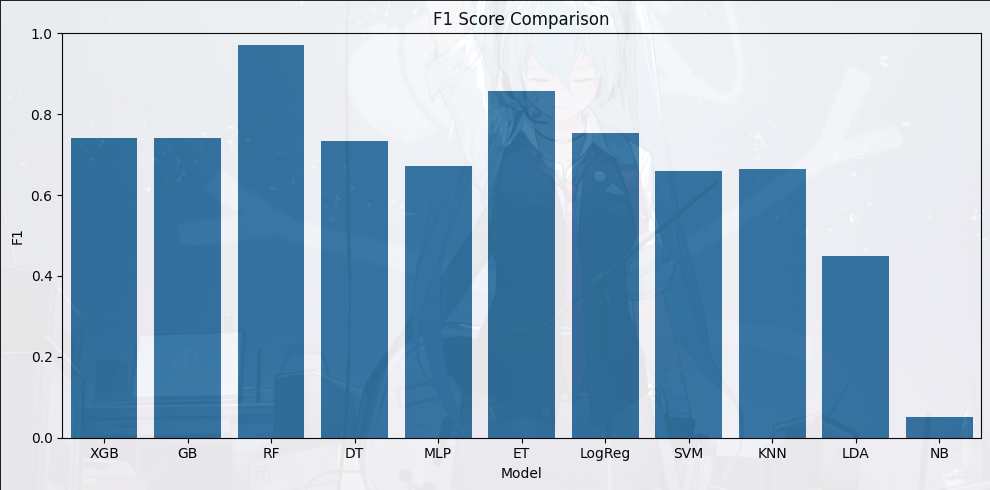
　　以下時間皆以訓練資料80%，測試資料20%為前題設計。

1. **機器學習實驗結果, F1以0~1區間表示**

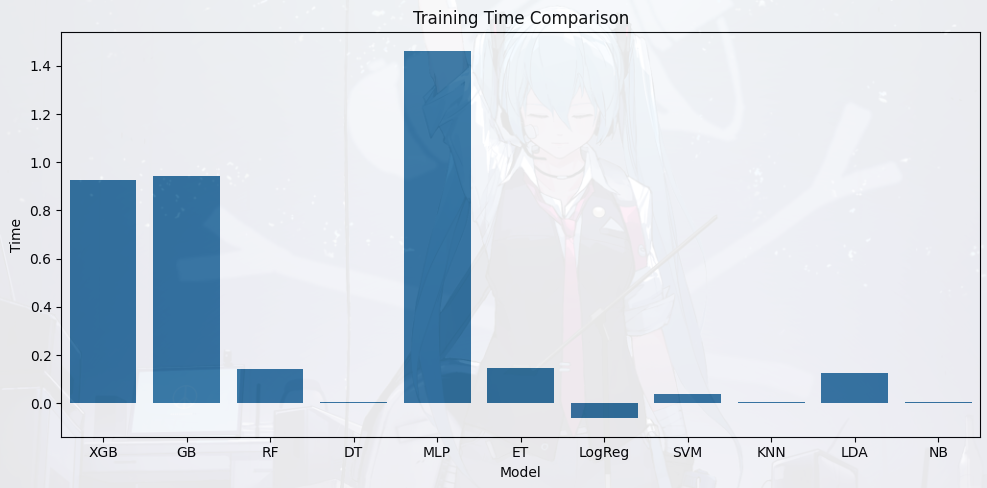
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 系統 | 分類器 | 系統設定 | Accuracy | F1 | Time (s) |
| A | LogReg | max\_iter=1000, 其餘預設 | 96.6176% | 0.7529 | -0.0625 |
| B | KNN | 預設 | 95.2941% | 0.6643 | 0.00440 |
| C | DT | 預設 | 99.5588% | 0.7327 | 0.00687 |
| D | RF | 預設 | 99.7159% | 0.9714 | 0.14040 |
| E | GB | 預設 | 99.7059% | 0.7415 | 0.94173 |
| F | XGB | 預設 | 99.7059% | 0.7415 | 0.92828 |
| G | ET | 預設 | 97.7941% | 0.8585 | 0.14812 |
| H | SVM | 預設 | 95.7353% | 0.6592 | 0.04027 |
| I | NB | 預設 | 6.3235% | 0.0512 | 0.00465 |
| J | MLP | max\_iter=1000, 其餘預設 | 98.5294% | 0.6729 | 1.46302 |
| K | LDA | 預設 | 95.1471% | 0.4486 | 0.12631 |



圖(一)：各方法正確性結果圖



圖(二)：各方法F1結果圖



圖(三)：個方法訓練時間結果圖

* 1. **系統比較**

　　為確保模型能順利收斂並取得穩定的分類結果，將 LogisticRegression 與 MLPClassifier 的最大迭代次數 (max\_iter) 調整為 1000。

　　在RF中，正確率高，且透過F1分數也可知泛用性高，適合分類的任務。反觀XGB與GB，雖正確率高，但F1分數偏低，顯示泛用性有待商確，且訓練時間偏高。ET的F1數值顯示，這是個在資源有限的高泛用性模型。DT在訓練的時間極短，容易過擬合，泛用性差。MLP雖準確率高，但訓練時間久，F1值也不是非常優秀，泛用性需多加斟酌。LogReg展現了穩定可靠的泛用性，且訓練速度極快，適合初步的快速分析。SVM雖訓練時間短，但不管是正確性抑或是泛用性，都不是很好，顯示在分類這工作的能力是較不足的。KNN則不同，在訓練時間相比SVM短非常多的前提下，正確率與其不相上下，泛用性更是反超SVM。LDA與NB在其他方法的比較下，則顯示不適合分類的工作，尤其NB，在正確性上幾乎可說沒有任何優勢。

* 1. **結論**

　　在Ensemble的方法中(RF, GB, XGB, ET)整體表現較優，是在分類工作上的好選擇。Boosting的方式(GB, XGB)雖準確率高，但在F1的數值相比Ensemble的方法來說，低了不少，顯示在少數類別的預測能力上有所不足。傳統的模型(LogReg, LDA, NB)雖訓練速度快，但面對複雜資料時，表現較差。

1. 機器學習細部調整

前情提要：

　　在資料的前處理上，與之前相同，將訓練與驗證資料集採4:1方式分配，隨意分配，無特別考慮四個分類各佔比，僅將資料按比數分配。

實驗方式：

　　以下十一種機器訓練皆採用本實驗方式。每種模型具有各自的調整方式，故本部分探討的是自己本身調整對結果的差異，而非模型間的比較。那每種參數皆會被訓練到，如A模型具有4種B參數、5種C參數，在一輪的訓練中總共會進行4\*5種訓練，而本人要求訓練十輪，並繪製出圖表，觀察穩定性。對於評判的指標，本人採取Accuracy、Loss、F1三種指標觀察最佳的正確性、最低損失、最泛用者其參數為何。

* 1. XGB

　　看到十次的訓練，結果非常穩定，正確率高達0.995，損失也低，只有0.015，泛用性也有0.95，顯示模型不只穩定，還相當不錯。

　　在參數的設定上，也看得出來是非常的”保守”，疊帶次數不多，只是50、100，深度也不多，且學習曲線也設定0.1、0.05以防止學習過快的過擬合。

　　在各指標的最佳成績中，拋開疊帶次數，基本上可以定調為學習曲線0.1，深度5會有較佳的實驗結果。

* 1. GB

　　在十次的訓練中，只有損失在0.07~0.06間浮動，其餘皆非常穩定，具有0.997的正確性，0.74的泛用性。可以看到在XGB、GB這兩個相近的模型中，XGB還是較佳的，雖然犧牲點正確性，但換來的是更低的損失、更高的泛用性。

　　參數的設計上也與XGB大差不差，以各指標最佳的結果來說，參數的設計已經有了非常明顯的結果，疊帶次數50，學習曲線0.05，深度5。

* 1. RF

　　以十次的訓練結果來看，可說是非常的穩定，正確性落於0.995，損失在0.03，泛用性也高達0.95，是個相當不錯的結果。

　　在參數的設計上，以跌代的次數提高穩定性，設定在50~200，在最大深度的設計上以10~20防止過擬合，但也好奇如果不限制會怎樣，多了個None，同時限制分裂最小樣本數、葉節點最小樣本數，最後以分裂準則的不同進行實驗比對。

　　在三個評分指標中，令我意外的是全部最佳的成績皆出現在沒對高度進行限制的方式中，這也許隱含了0.997的正確率是嚴重的過擬合。畢竟結果高於總訓練的0.95太多。先拋開過擬合，以三筆數據來看，比較正確性顯然是沒有意義的，以評判的標準來看，個人叫趨向於entropy，畢竟較快的gini再以經過擬合的數據中，某程度來說加速了過擬合的發生，故以增長較慢的entropy來看也許較為合適。其損失在三者間無大差異，損失還比其他兩者低0.01。

* 1. DT

　　經過多次實驗，本人發現，在給定隨機狀態為42後，實驗結果真的會相當不錯。結果論來說，正確率有0.997，損失也夠低，只有0.005，泛用性也高，有0.977。

　　參數的設計上較RF簡單，僅設最高深度、分裂最小樣本數、葉節點最小樣本數及分裂準則。

　　以三個評分標準而言，首先看到最大高度不再是RF的”無限制”，顯示過擬合的現象或許沒有發生，比較的準則也是個人較偏好的entropy，並且不管在哪種評分標準中，參數的設至除了最少葉節點的不同，其餘皆有了共識。而在此個人偏好Loss最低的設置，雖然犧牲0.01的泛用性，可在損失看來，低了0.05，單論數據而言，是比較漂亮的。

* 1. ET

　　在十次訓練中，可以明顯看到ET的結果相較DT而言，是全盤皆輸的情況。正確率0.98，損失0.06，泛用性0.88。

　　參數的設計僅有對疊帶次數、深度、分裂準則做限制，本身可預期到結果不慎理想。

　　在三種評分標準下，除了疊帶次數不同，其餘皆有取得共識，甚至單筆數據拿下來正確率最高、最高泛用性兩者。但數據著實不太漂亮，深度全是不多做限制者，過擬合的發生是需要被考慮的，以gini的分裂標準面對深度不限制的樹，著實也是令人聞風喪膽的組合。不畏言，ET著實不是在分類這項工作上的好方法。

* 1. MLP

　　在以上實驗中，本組可說是浮動數一數二大的，雖然正確率約坐落在0.98左右，損失約在0.03左右，但泛用性非常不穩定，從0.65~0.95都有，可說結果是極其不穩定。

　　參數的設計上首先定義神經的層數、每層的神經元數目，激活函數，最佳化方式，正則化強度及學習曲線。可以看到由於本身硬體的限制，本人還是相當保守的，在神經元的數目上基本上很少，層數也低，激活函數基本上就挑幾個常見的，最佳化選用適合大多數情況的adam。

　　既然本實驗數據波動較大，相較於其他模型，討論三種評分最佳者的意義就顯得沒那麼重要，畢竟結果是如此的不穩定。且可以看到基本上就是第72號數據為基準，三者正確率都約0.98但泛用性實在落差過大，猜測激活函數帶來的影響頗大。

* 1. LogReg

　　以實驗十次的結果而言，非常的穩定，正確率落在0.969，損失0.097，泛用性0.78。

　　參數的設定上與penalty脫不了干係，以C控制penalty程度，並選用l2。最後挑選常用的lbfgs演算法，加上由於本人好奇擅長處理稀疏矩陣的saga方法在分類上的功力，增加的實驗。

　　從三者評分指標可見參數的設定上只有C有不同，而此模型對於泛用性來說，相較於其他，或許在分類上皆顯得不那麼合適。

* 1. SVM

　　首先聲明，此比較有機會更深入，如果我的荷包夠深或時間夠多。本人發現只要參數動到kernel，訓練時間就會暴漲，這要跑十回合，本人對於一回合跑兩小時還沒結束的結果實在沒法等待。

　　就目前結果而言，還是相當穩定的，正確率約0.96，損失約0.098，泛用性約0.75。再次強調這結果如果套用更大量的運算，如poly，或許在三種指標上都會有所改變。

　　在三種評分指標底下，此實驗基本由第52號數據拿下，可以看到正確性、泛用性數據是相同的，而損失是三者中最小。比較需要注意的是正則化參數是10，可能因設置太大而存在過擬合現象。

* 1. KNN

　　在十次的實驗中，這結果非常穩定，準確率落在0.96，損失高得嚇人，有0.64，泛用性也偏差，僅0.74。

　　參數的設計上，設定鄰居數量，權重的決定方式，矩陣的運算模式。

　　在本人看來，7號數據是KNN的最佳解，雖然泛用性稍低於其他兩者0.02，但損失降低的程度非常顯著。且鄰居較多，跑出來的結果應該較為穩定。整體而言，KNN是一個較不是於被應用在此分類工作上的模型。

* 1. NB

　　以實驗的結果來說，雖然是非常穩定的，但也是相當差勁的。正確率低落，僅0.28，損失高達8.29，泛用性也僅0.17。

　　參數上僅對影響模型穩定性的浮點樹作調整。以三種評分的指標來看，全由3號數據拿下。且從全部的紀錄中可以看到浮點數越小，結果就越差。整體而言，NB就不該拿來做分類工作。

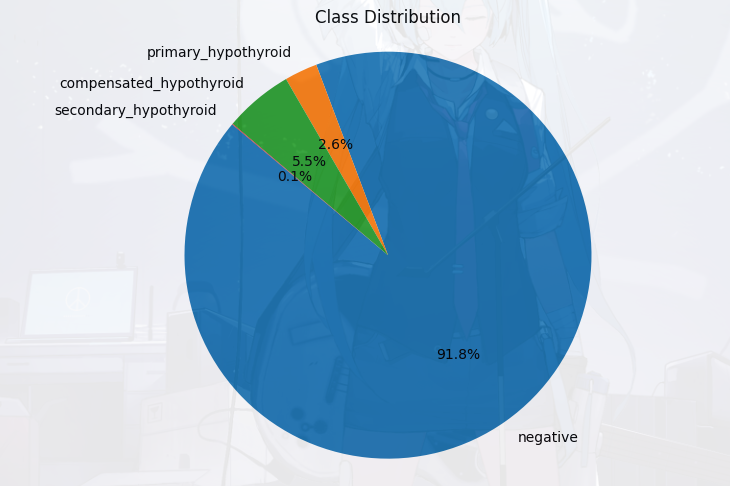
* 1. LDA

　　以十次的實驗而言，結果是非常穩定的，正確率有0.95，損失0.21，泛用性偏低，落在0.44。

　　參數的設定上有選擇數值解法的最小平方法、特徵值解法，並控制共變異矩陣的正則化強度防止過擬合發生。三種評分指標皆指出數據0即最佳解。整體而言，尚可，如損失可再低，泛用性可再提高再好不過。

1. **深度學習實驗結果**

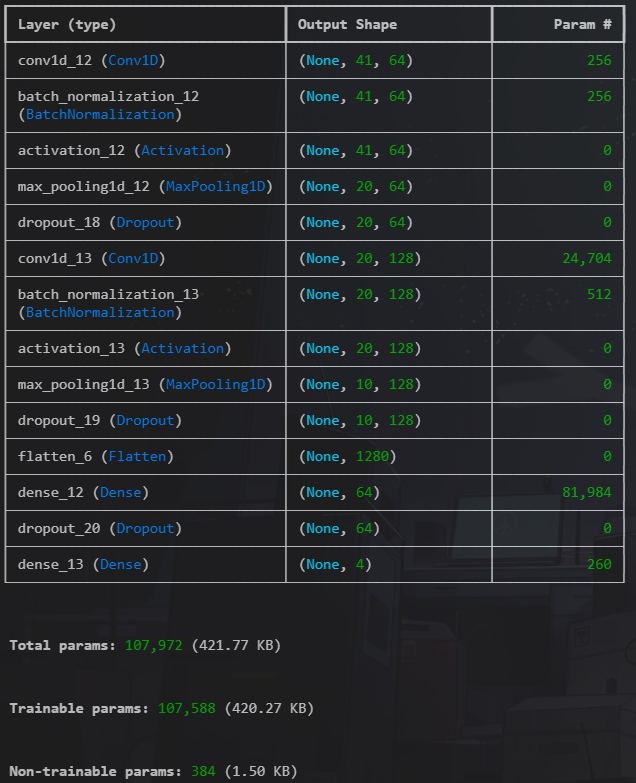
　　在資料處理時，列出蒐集到的資料結果分布圖，顯示原始資料分布。



圖(四)：蒐集資料結果分布圖

* 1. CNN模型

　　對於模型，首先建立Conv1D(64)的卷積層用來抽取局部特徵，處理結構化、序列型資料。利用Batch Normalization + Activation + Pooling保持數值穩定，降低維度。再建Convid(128)的卷積層已抽取更深層特徵。隨後以Flatten將多為輸出轉為向量給Dense(64)使用，以Dense + Dropout建立分類邏輯避免過擬合。最後輸出分類結果。



圖(五)：CNN模型結構

　　從左圖可見訓練與驗證的損失函數趨勢向下，學習過程大致上趨於穩定;右圖的正確性函數在訓練時學習效果良好，呈現趨勢向上，無明顯停滯或震盪，訓練初期穩定，中期雖出現震盪，但整體正確性依然是相當高的，是個作為分類的不錯選擇，雖CNN本身適合圖片分析，可在此實驗呈現出文字類的分類工作也有不錯的成效。

|  |  |
| --- | --- |
|  |  |
| 圖(六)：CNN損失函數 | 圖(七)：CNN正確性函數 |

* 1. LSTM模型

　　在模型設計上，首先用LSTM(64)處理時間序列資料，學習長期關係如病患過去的症狀。而後使用Batch Normalization來穩定模型的訓練速度與表現避免梯度消失問題。加上Dropout防止過擬合，試圖提高泛用性。再使用LSTM(32)健力第二層時間記憶，提高時間的依賴特徵，再加上Batch Normalization與 Dropout提高穩定性。第三次使用Dense(64) + Activation 將時間特徵轉為高為抽象特徵，最後經由Dropout後使用Dense(4)分類至四個結果。



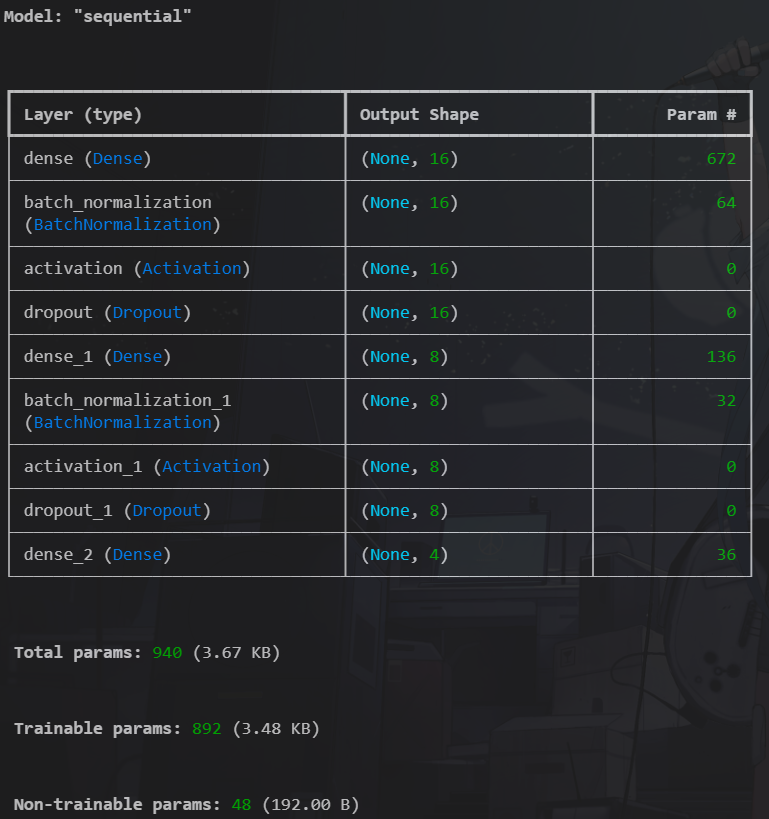
圖(八)：LSTM模型結構

　　先看到左圖的損失函數，再訓練時，確實表現出學習的樣子，趨勢向下，但驗證集卻劇烈波動，顯示出嚴重的過度擬合，猜測某些epoch無法有效的泛化。這情況也在右方的正確性中顯示，訓練時表現也沒啥問題，可驗證集呈現出劇烈波動，猜測模型過度技藝訓練資料，泛化能力不足。

|  |  |
| --- | --- |
|  |  |
| 圖(九)：LSTM損失函數 | 圖(十)：LSTM正確性函數 |

* 1. MLP模型

　　首先利用Dense將輸入特徵映射到hidden layer中，經過Batch Normalization加速收斂。藉由ReLU的非線性轉換，增強模型的表達能力，透過Dropout防止過擬合的發生。再將以上過程複製一次，最終經由Dense輸出至四個分類中。



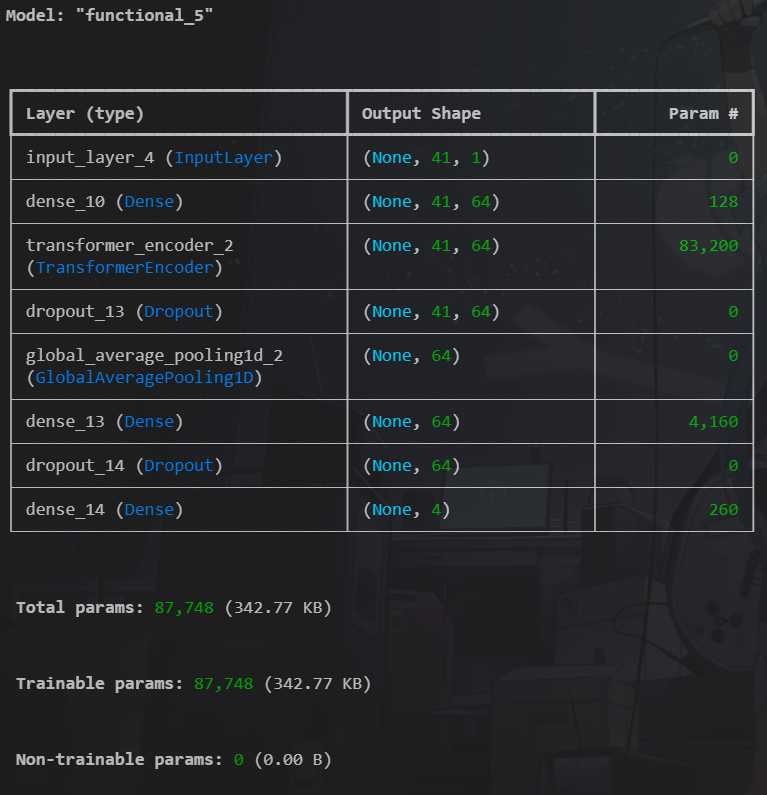
圖(十一)：MLP模型結構

　　左圖的損失函數訓練與驗證集皆呈現出趨勢向下的結果，可訓練的loss始終高於驗證集，猜測可能為Dropout影響。右圖的正確性整體呈現出趨勢向上，且訓練與驗證集差距不大，未發生過擬合，整體而言收斂良好，適合此分類的工作。

|  |  |
| --- | --- |
|  |  |
| 圖(十二)：MLP損失函數 | 圖(十三)：MLP正確性函數 |

* 1. Transformer模型

　　首先需要提及的是，這是個有關於時間的模型，本人將41個特徵當作41個時間點為前提，在第一層的input接收41個特徵，將輸入映射至高維空間(Dense\_10)，再經由TransformerEncoder捕捉時間的全局依賴關係，加上個Dropout防止過擬合。壓縮時間維度成為向量，送入全連階層，而後再經過一次Dropout，最終輸出至四個對應結果。



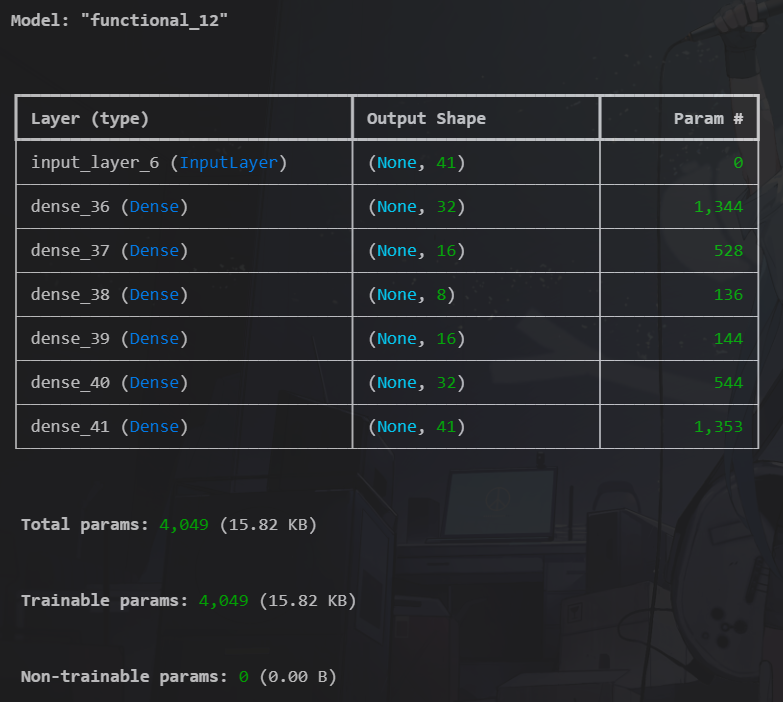
圖(十四)：Transformer模型結構

　　左圖中的損失函數整體而言表現出趨勢向下，本人觀察到訓練集與實驗集差距漸漸縮短，表現出學習效果良好。右圖的正確性函數在訓練時整體而言穩定向上，可驗證時卻不是如此，雖然趨勢也是向上，但在0.93~0.96區間震動較大，反映出了不穩定性，雖是如此，整體而言正確性依然是相當高的，表示Transformer在分類的工作上具有出色的泛化能力。

|  |  |
| --- | --- |
|  |  |
| 圖(十五)：Transformer損失函數 | 圖(十六)：Transformer正確函數 |

* 1. Autoencoder模型

　　首先在輸入層將原始資料(41特徵)對應到41維度，經過第一層降維提取特徵，第二層進一步壓縮取出特徵，第三層維最小瓶頸曾，保留最重要資訊，而後開始重建還原原始維度特徵結構，最終輸出層與輸入層同為41維度，用以誤差的計算。



圖(十七)：Autoencoder模型結構

　　首先針對為何選用MAE函數而非正確性函數作出解釋，Autoencoder做為一個重建模型，MAE相比於正確性函數更能反映出重建誤差，準確率適合拿來作為分類模型的指標，而非重建的效果分析。

　　左圖的損失函數訓練與驗證集接穩定下降，表示模型重建穩定，沒有過擬合現象發生。MAE曲線大致走向與損失函數相同，表現出泛化能力良好。

|  |  |
| --- | --- |
|  |  |
| 圖(十八)：Autoencoder損失函數 | 圖(十九)：Autoencoder之MAE函數 |

* 1. 深度學習實驗總表

|  |  |  |
| --- | --- | --- |
|  | Accuracy (0~1) | Loss (0~1) |
| CNN | 0.9852 | 0.0497 |
| LSTM | 0.9694 | 0.0649 |
| MLP | 0.9440 | 0.1167 |
| Transformer | 0.9478 | 0.1083 |

|  |  |  |
| --- | --- | --- |
|  | MAE (0~1) | Loss (0~1) |
| Autoencoder | 0.2179 | 0.0675 |

　　根據上表，可知CNN在正確率及損失函數的表現上，皆呈現出最佳，為本次實驗的深度學習模型中的最適合擔任分類工作的模型。LSTM的表現也不錯。這結果屬實令我感到驚訝，CNN適合做的是抽取影像的特徵，而LSTM擅長的是有時間順序的資料，可在這甲狀腺的分類中，似乎兩者皆不太符合各自擅長領域，可正確率、損失函數的呈現結果又還不錯。

　　MLP和Transformer表現相比於前兩者就略遜不少，或許可將原因歸咎於資料量的筆數不夠大量，或許資料更充足，經過更充分的訓練，可將損失函數的表現呈現得更加出色，正確率也有所提升。

　　最後，來談談Autoencoder，這是本人有意挑選的模型，此模型目的在於重建，而不在分類，可預期到的結果為如上表所示，損失函數的呈現不是太良好。基於目的(重建)的不同，比較正確率是沒有意義的，轉而比較MAE函數，以反應出重建誤差。而即使損失函數呈現結果看起來不錯，也僅代表模型能還原輸入，無法保證對於不同類別的資料有不錯的分類能力。

1. 實驗感想

　　經過本次的實驗，我想我學習到最多的部分是如何比較，不管是機器學習抑或是深度學習，都遇到了參數設定上注定不可能完全一樣，意思是我無法準確控制到操作變因為模型不同，其餘參數都相同。每個模型各自著重的點都不同，一開始的想法是盡量以預設為前提，期望可以達到模型各自參數上的差別最小化，但深入了解模型各自的參數後，這根本就是種奢望，在機器學習的部分也可以看到我在兩種模型的最大推論次數設定1000，期望不會遇到無法收斂的問題，而其他模型並不見的會有這問題，抑或是壓根運作機制就不同，這就造成了點差異。

　　在深度學習上，我刻意每個模型都挑選了擅長不同領域的，期望看出在這資料分類的工作中有顯著的差異，可最明顯的是Autoencoder，而這是不用經過實驗就應該預期的到的，其他竟然正確率都有高達九成，這明顯與我期望看到的不太相符，或許真的是資料上的筆數差異造成不管是訓練、驗證都無法完美的審視模型的可靠性，只有在損失函數的部分凸顯出了MLP與Transformer在資料上的不足所出現的短版­­損失函數實驗結果不太好，其他皆沒有太大差異。

　　在這次的作業中，環境的設置上也是不少的問題，我第一次跑需要機器學習與深度學習的作業，不用想也知道tensorflow肯定會撞到一堆版本問題，在不搞砸自己電腦環境的前提下，我使用Docker devcontainer為開發工具，可沒料到的是，GPU的調用竟是如此的困難，在Rebuild無數次後，我舉了白旗，靠著資料筆數沒那麼大的前提下用CPU撐完了期末，我想這是我最困擾的部分，看著訓練的速度慢的可以…真的是想起GPU的好。