

110 學年度國立新竹科學園區 實驗高級中等學校自主學習

AI 人工智慧辨識 Covid-19 之病例

指導老師：江天瑋

班 級：一年三班

學生姓名：蔣承諺

目錄

第一章 前言	5
1.1 計畫動機與目的	5
1.2 學習方法與執行策略	5
1.3 預期成果	5
第二章 資料來源和開發環境介紹	6
2.1 資料來源 Kaggle Datasets 介紹	6
2.2 COVID-19 Radiography Database 介紹	6
2.3 Colab 開發環境介紹	7
第三章 訓練資料的統計與分析	8
3.1 說明	8
3.2 資料集與圖像資訊	8
3.3 圖像預處理與數據增強	9
第四章 深度學習神經網路建構說明	11
4.1 遷移學習之目的說明	11
4.2 微調學習之目的說明	12
4.3 預訓練模型 VGG16 介紹	13
4.4 深度學習神經網路建模之完整程式碼	14
第五章 數據成果分析與問題討論	18
第六章 自主學習歷程與心得反思	23
6.1 自主學習歷程	23
6.2 心得反思	24
第七章 結論	25
第八章 參考資料	26

自主學習研究建模之完整程式碼連結:

[自主學習 VGG16 辨識模型 Covid and Viral Pneumonia classify](#)

自主學習計劃書

竹科實中 110 學年度第 2 學期 學生自主學習計劃

班級：3 座號：35 姓名：蔣承諤

計畫名稱	AI 人工智慧辨識 Covid-19 之病例
個人/團體	<input checked="" type="checkbox"/> 個人計畫 <input type="checkbox"/> 團體計畫，組員：
計畫模式	<input type="checkbox"/> 校訂專題實作(高二限定) <input checked="" type="checkbox"/> 學術研究 <input type="checkbox"/> 線上學習 <input type="checkbox"/> 實體微課程(依實際開課項目選課) <input type="checkbox"/> 活動策劃 <input type="checkbox"/> 藝文創作 <input type="checkbox"/> 服務學習 <input type="checkbox"/> 其他：
學科領域	<input type="checkbox"/> 語文領域 <input type="checkbox"/> 數學領域 <input type="checkbox"/> 社會領域 <input checked="" type="checkbox"/> 自然領域 <input type="checkbox"/> 藝術領域 <input checked="" type="checkbox"/> 科技領域 <input type="checkbox"/> 綜合領域 <input type="checkbox"/> 健康領域 <input type="checkbox"/> 其他：
學習環境與設備需求	<input checked="" type="checkbox"/> 安靜閱讀區 <input checked="" type="checkbox"/> 電腦資訊檢索區 <input type="checkbox"/> 團體討論區 <input type="checkbox"/> 視聽學習區 <input type="checkbox"/> 其他：
計畫動機與目的	由於疫情肆虐全球，因此人類們正不斷的研發疫苗以及快篩來保護跟檢測我們的健康。另一方面，由於人工智慧的普及，它不僅能輔助醫生判讀，並且能有效地預測疾病，從而讓我想製作能判別 Covid-19 之病例的 AI 模型。
學習方法與執行策略	透過閱讀有關機器學習的相關書籍以及利用網路資源探索並下載資源。
預期成果	撰寫一個 AI 模型讓機器達到深度學習並能完成準確判讀的效果。

【自主學習規劃】

週次	進度規劃表	學習時數
第 1 週	閱讀《學 AI 真簡單 1—初探機器學習》之第一章-第四章並使用電腦實際測試與製作	1
第 2 週	閱讀《學 AI 真簡單 1—初探機器學習》之第一章-第四章並使用電腦實際測試與製作	1
第 3 週	閱讀《學 AI 真簡單 1—初探機器學習》之第一章-第四章並使用電腦實際測試與製作	1
第 4 週	閱讀《學 AI 真簡單 2—動手做深度學習》之第一章-第四章並使用電腦實際測試與製作	1
第 5 週	閱讀《學 AI 真簡單 2—動手做深度學習》之第一章-第四章並使用電腦實際測試與製作	1
第 6 週	閱讀《學 AI 真簡單 2—動手做深度學習》之第一章-第四章並使用電腦實際測試與製作	1
第 7 週	閱讀《學 AI 真簡單 3—輕鬆學自然語言處理》之第一章-第七章並使用電腦實際測試與製作	1
第 8 週	閱讀《學 AI 真簡單 3—輕鬆學自然語言處理》之第一章-第七章並使用電腦實際測試與製作	1
第 9 週	閱讀《學 AI 真簡單 3—輕鬆學自然語言處理》之第一章-第七章並使用電腦實際測試與製作	1
第 10 週	閱讀《學 AI 真簡單 3—輕鬆學自然語言處理》之第一章-第七章並使用電腦實際測試與製作	1
第 11 週	閱讀《學 AI 真簡單 4—探究巨量資料》之第一章-第四章並使用電腦實際測試與製作	1
第 12 週	閱讀《學 AI 真簡單 4—探究巨量資料》之第一章-第四章並使用電腦實際測試與製作	1
第 13 週	閱讀《學 AI 真簡單 4—探究巨量資料》之第一章-第四章並使用電腦實際測試與製作	1
第 14 週	整合前面所學之機器學習基本知識，設計及建構 AI 模型，並上網查找資料	1
第 15 週	設計及建構 AI 模型，並上網查找資料	1
第 16 週	建構 AI 模型，並上網查找資料	1
第 17 週	自網路蒐集 Covid-19 之胸腔 X-ray 病例圖資料集，並輸入病徵之相關資料用以訓練機器之辨識能力	1
第 18 週	測試 AI 之辨識能力並修改不足之處，增進辨識準確度	1

自主學習成果

☒靜態展示：
☐口頭報告：
☐其他：

初審意見：

初審結果：☐通過 ☐請修訂

複審結果：☐通過 ☐請修訂

教務處	學務處	輔導室
圖書館	學生簽名	家長簽名

其他注意事項

每週進度表

110 學年度第 2 學期國立竹科實中自主學習每週成果紀錄表

班級：103 座號：35 姓名：蔣永諤

學習計畫名稱：AI 人工智慧辨識 Covid-19 之病例

學習場地：國中電腦教室

週數	學習內容與進度	自我檢核	指導老師確認
1	閱讀完〈學AI真簡單1〉之第一章與第二章	<input type="checkbox"/> 優良 <input checked="" type="checkbox"/> 尚可 <input type="checkbox"/> 待努力	教師蔣江天瑋
2	閱讀完〈學AI真簡單1〉之第三章	<input type="checkbox"/> 優良 <input checked="" type="checkbox"/> 尚可 <input type="checkbox"/> 待努力	教師蔣江天瑋
3	閱讀完〈學AI真簡單1〉之第四章	<input type="checkbox"/> 優良 <input checked="" type="checkbox"/> 尚可 <input type="checkbox"/> 待努力	教師蔣江天瑋
4	上網查找可用AI模型	<input type="checkbox"/> 優良 <input checked="" type="checkbox"/> 尚可 <input type="checkbox"/> 待努力	教師蔣江天瑋
5	建立該模型之座標環境，並安裝所需套件	<input checked="" type="checkbox"/> 優良 <input type="checkbox"/> 尚可 <input type="checkbox"/> 待努力	教師蔣江天瑋
6	安裝所需套件，上網查詢相關資料	<input checked="" type="checkbox"/> 優良 <input type="checkbox"/> 尚可 <input type="checkbox"/> 待努力	教師蔣江天瑋
7	安裝所需套件，上網查詢相關資料	<input type="checkbox"/> 優良 <input checked="" type="checkbox"/> 尚可 <input type="checkbox"/> 待努力	教師蔣江天瑋
8	環境、程式 Debug	<input type="checkbox"/> 優良 <input checked="" type="checkbox"/> 尚可 <input type="checkbox"/> 待努力	教師蔣江天瑋
9	環境、程式 Debug	<input type="checkbox"/> 優良 <input checked="" type="checkbox"/> 尚可 <input type="checkbox"/> 待努力	教師蔣江天瑋
10	執行 AI 模型，問題無法解決，上網查資料	<input type="checkbox"/> 優良 <input checked="" type="checkbox"/> 尚可 <input type="checkbox"/> 待努力	教師蔣江天瑋
11	上網查找資料，嘗試解決還是失敗	<input type="checkbox"/> 優良 <input checked="" type="checkbox"/> 尚可 <input type="checkbox"/> 待努力	教師蔣江天瑋
12	請教學長解決辦法	<input checked="" type="checkbox"/> 優良 <input type="checkbox"/> 尚可 <input type="checkbox"/> 待努力	教師蔣江天瑋
13	向學長請教圖像辨識模型，上網查找資料	<input checked="" type="checkbox"/> 優良 <input type="checkbox"/> 尚可 <input type="checkbox"/> 待努力	教師蔣江天瑋
14	上網選定可用資料庫，並修改AI辨識模型	<input checked="" type="checkbox"/> 優良 <input type="checkbox"/> 尚可 <input type="checkbox"/> 待努力	教師蔣江天瑋
15	修改AI辨識模型	<input type="checkbox"/> 優良 <input checked="" type="checkbox"/> 尚可 <input type="checkbox"/> 待努力	教師蔣江天瑋
16	修改、測試AI辨識模型	<input type="checkbox"/> 優良 <input checked="" type="checkbox"/> 尚可 <input type="checkbox"/> 待努力	教師蔣江天瑋
17	修改、測試AI辨識模型	<input checked="" type="checkbox"/> 優良 <input type="checkbox"/> 尚可 <input type="checkbox"/> 待努力	教師蔣江天瑋
18	訓練、產出AI辨識模型	<input checked="" type="checkbox"/> 優良 <input type="checkbox"/> 尚可 <input type="checkbox"/> 待努力	教師蔣江天瑋

指導教師核章：

教務處核章：

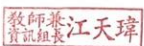

教師蔣江天瑋

自主學習成果報告書

國立竹科實中自主學習成果報告

(此紀錄表於期末繳交)

110學年度 第2學期

填寫日期	6/23	班級	103
座號	35	姓名	蔣承諤
自主學習主題	AI人工智慧辨識Covid-19之病例		
計畫模式	<input checked="" type="checkbox"/> 個人 <input type="checkbox"/> 團體 (成員:)		
學習成果呈現方式	<input checked="" type="checkbox"/> 靜態展示 <input type="checkbox"/> 動態演出 <input type="checkbox"/> 海報發表 <input type="checkbox"/> 影片上傳 (須附連結) <input type="checkbox"/> 其他:		
自主學習計畫成果簡要說明	<p>此次的自主學習中，透過課外書籍中的資料、網路上的資源以及學長的講解，讓我能夠順利地做出此次自主學習成果報告。透過匯入新冠肺炎感染以及未感染的X-ray測試資料集到我訓練好的模型當中 (20個模型)，經過判斷分析後，模型會給出它測驗出來的結果，包括準確值和損失函數值。而我的所有模型準確值均為0.625，損失函數不一 (最小值為0.5289)，由此可判斷此模型的準確度在可接受的鑑別力範圍。</p>		
自主學習計畫成果自評	<p><input type="checkbox"/>符合預期 <input checked="" type="checkbox"/>部分符合預期 <input type="checkbox"/>未符合預期</p> <p>簡要說明： 對於模型的結果其實我已是相當滿意，因為對於新接觸的我，能夠產出一個模型就算是達成我心中的目標了。而準確值的部分也算符合預期，不至於太過低落，仍具有有效預測能力。只不過我原本預期要閱讀完課外讀物來學習機器學習，可惜在短期的時間內成效不佳 (對我來說有點難)，只能夠轉換方法來完成自主學習。</p>		
學習歷程省思	<p>在短短的自主學習當中，我認為我要從零開始學習機器學習並產出完美的成果是一件非常有挑戰性的事情。而正如我想的那樣，對於沒有基礎的我處理龐大的資訊量實在有點力不從心。因此我開始向資研社有經驗的學長請教這方面的問題，透過不斷的討論以及他的講解，讓我最終能夠產出一個滿成功的機器學習模型，而能夠判斷出感染新冠肺炎的機率值。</p>		
核章	指導老師	教務處核章	
			

備註：1. 表格如不敷使用，可自行增列。

2. 若有照片／影片成果者亦可貼上成果照片，或提供影片連結，再加上敘述。

第一章 前言

1.1 計畫動機與目的

近兩年來在新冠疫情的衝擊下，全世界各個角落可說是無一幸免於難，不管是經濟、人文、政治，抑或是個人衛生和個人習慣都因而受到了極大的影響。每天全世界的確診人數都在不斷的攀升，而有些國家的醫療資源以及人力資源卻相當的缺乏，而導致病患沒能在患病的第一時間受到完善的照顧與診斷。同一時間醫護人員的分配與調度也成了大問題，除了需要照顧新冠確診者之外，還須同時兼顧到其他的傷症病患，讓醫護人員忙得不可開交。因此這些問題都是有待解決的。

在疫情嚴峻的同時，科技的發展也是日新月異，AI 人工智慧、機器學習、深度學習和互聯網等技術更是愈發的成熟。許多的跨領域結合便是利用了人工智慧抑或是機器學習等技術，將其應用於交通、經濟與醫療等多個面向，藉此輔助人類判斷及預測某些事物發展的機率等，都能夠大幅的提高準確度，更能夠降低繁雜的時間成本，可說是利益良多。而為了解決上述的醫療問題，已經有研究人員訓練出了多個能夠有效辨別新冠確診者的深度學習模型，並且準確度也是相當的高，可實際運用於病例當中。

因此綜合以上的問題與實際運用的例子，我決定要利用網路上的資源以及書籍當中的知識，來自己創建出一個能夠藉由肺部 X-ray 的影像來輔助醫生判斷病患是否為新冠確診者的初步模型，藉此也讓自己對機器學習和深度學習這塊領域能夠有更深入的學習與探究。

1.2 學習方法與執行策略

透過閱讀 AI 機器學習的入門書籍來增進自己對機器學習、AI 以及深度學習的基礎知識，並實際跟著書中的範例實作來增加自己的印象。除此之外，我也打算利用網路上廣泛的資料以及資源作為書中內容的補充，讓自己能夠順利地創建出深度學習的模型。

1.3 預期成果

我希望在本次自主學習的時間當中，能夠創建出如上述所言的一個藉由肺部 X-ray 的影像來輔助醫生判斷病患是否為新冠確診者的深度學習模型，並對機器學習與深度學習這塊領域能有更深入的了解，同時產生學習的熱忱。

第二章 資料來源和開發環境介紹

2.1 資料來源 Kaggle Datasets 介紹

Kaggle 是一個數據建模和數據分析的競賽平台，也是全世界公認最大的資料科學社群。企業和研究者可在上面發布數據，進而提出一些實際需要解決的問題，而統計學者、數據挖掘專家和業界人士等參賽者則在上面進行競賽、解決問題以產出最好的機器學習模型，抑或是為了在眾多參賽者當中贏得高額的獎金。

另外，Kaggle 上面還設有了 datasets 專區，也就是說它裡面有許多的資料集，而且都是經由別人整理過的資料並且提供每個學習者和參賽者下載，是一個非常好的開源資料集。除此之外，有些研究人員或是資料學家會在資料集下方分享自己的研究成果，讓大眾能夠有更好的學習方向。

在我此次的自主學習研究當中需要用到新冠確診者胸腔的 X-ray 資料集。由於醫療方面的資料集可能會因為病患隱私的關係，而較難取得大量且完善的資料，導致無法產出較具參考價值的深度學習模型。而 Kaggle 的 Datasets 正能夠解決這方面的問題，其中便涵蓋了多種研究人員上傳的資料集讓學習者能夠下載，因此最終才能夠完成此次的自主學習研究。

2.2 COVID-19 Radiography Database 介紹

此開源數據資料集內包含了正常、細菌性肺炎、病毒性肺炎以及 Covid-19 陽性患者的胸腔 X-ray 圖像，並且有研究人員會不時上傳、更新與維護此資料庫中的資料，具有一定的可信度。而在此次的自主學習研究當中我將進行 Covid-19 陽性和病毒性肺炎的 X-ray 判讀比較，因此以下僅介紹此兩個資料集的資料來源。

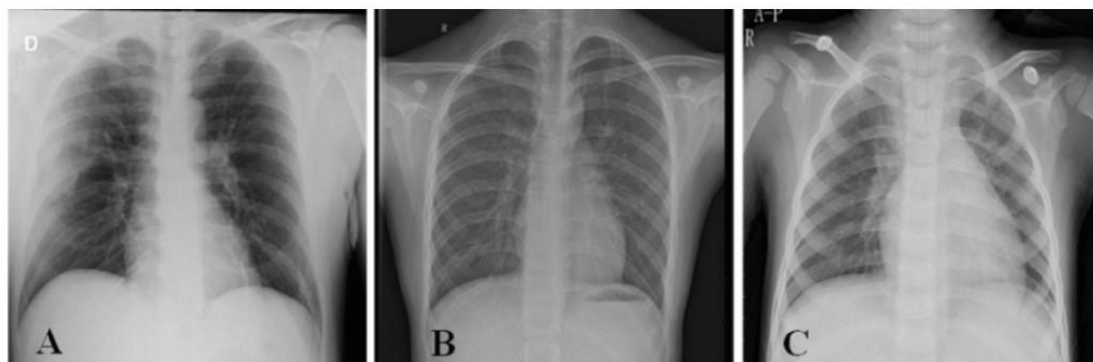
Covid-19 數據集：

此數據集是由 3616 張 Covid-19 CXR(Chest X-ray)陽性圖像所組成，而這些圖像是從不同的公開可用數據集、線上資源和已發表的文章中所收集的。其中包含了來自 BIMCV-COVID19+數據集、德國醫學院、意大利醫學放射學會(SIRM)、GitHub、Kaggle、Twitter 以及另一個 COVID-19 CXR 的儲存庫。而在於非 COVID 和 COVID 類別之間的主要區別是透過 CXR 圖像中的肺不透明度分別是由其他肺部相關疾病引起，還是 COVID-19 所導致的。

Viral Pneumonia(病毒性肺炎)數據集：

此數據集是由 1345 張病毒性肺炎 CXR 圖像所構成。其中有一部分的圖像

是來自北美放射學會(RSNA)所舉辦的一項人工智能肺炎檢測挑戰賽，在該數據庫中可獲得無肺部感染的正常胸部 X 光圖和非 Covid 的肺炎 X 光圖像。而另外一部分的圖像則是來自 Kaggle 胸部 X 光數據庫，裡面包含了正常、細菌性和病毒性肺炎的胸部 X 光圖像。最終此數據庫作者將這些蒐集到的圖像彙整成了一個新的數據庫，以便各方人員進行研究。



(圖 A)Covid-19 陽性 X 光圖像 (圖 B)正常 X 光圖像 (圖 C)病毒性肺炎 X 光圖像

2.3 Colab 開發環境介紹

Colab 是一個在雲端運行的編輯執行環境，它支援了 Python 程式以及機器學習的多個套件，並且使用者不須做任何的環境設定以及下載步驟，全部都是在瀏覽器上執行而且是完全免費。不僅如此，Colab 中的程式碼預設會直接儲存在開發者的 Google Drive 雲端硬碟中，並且使用 Google 所提供的虛擬機以及免費的 GPU、TPU 進行強大的運算，不會占用到本機的資源，還能夠跟其他人輕鬆共用，可說是相當方便。只不過由於許多資源(像是 GPU、TPU)是免費提供的關係，因此可用的內存量會隨著時間以及用戶端進行變化，如果時常閒置或是使用量超過免費提供上限的話，就會導致暫時無法利用 GPU 來進行計算，抑或是程式執行被中斷，而需要再重新執行一次程式碼。

在綜合與考量以上的優缺點後，我便決定使用 Colab 筆記本來編寫此次自主學習的內容。雖然我也是初次使用 Colab，對於一些操作方式以及運作功能還尚未熟悉，但經過多次使用與上網查找資料後，便覺得漸漸順手，操作起來也是相當的流暢，方才體會其便利性。而我認為最有感的好處便是 Colab 所提供擁有強大運算能力的 GPU，它讓我在進行深度學習模型訓練時的執行速度大幅提升，而不至於訓練一次模型就需要花上好幾個小時，十分方便。

Python 3 Google Compute Engine 後端 (GPU)

目前顯示自 上午8:48以來的資源



左圖為使用 Colab 筆記本時，可運用的免費 GPU 資源，幫助電腦能夠快速運算與訓練深度學習模型

第三章 訓練資料的統計與分析

3.1 說明

本次自主學習研究我將使用兩種不同類別的訓練資料集分別是: Covid-19 陽性 X 光圖像資料集和 Viral Pneumonia(病毒性肺炎)X 光圖像資料集，而接下來我將個別統計與分析各個資料集的大小以及圖像的各類資訊，並以圖表呈現。

3.2 資料集與圖像資訊

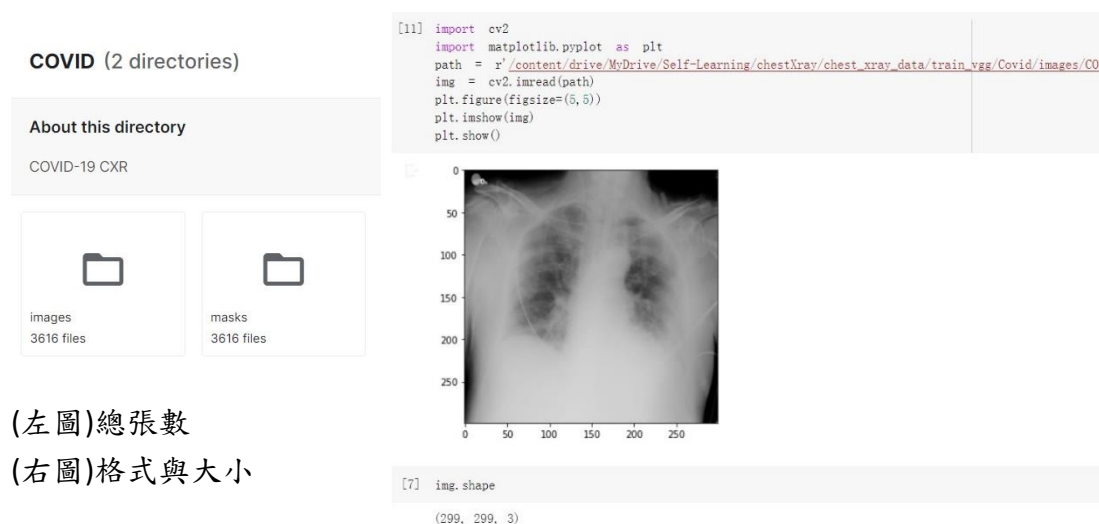
Covid-19 資料集:

內部共有兩個資料夾，分別是 X 光圖像和肺部切割圖，在本次研究當中只會使用到 X 光圖像的資料夾而已。

Covid-19 陽性胸腔 X 光圖像

總張數: 3616 張

格式與大小: 均為 PNG 文件格式，解析度為 299*299 像素



Viral Pneumonia(病毒性肺炎)資料集:

內部共有兩個資料夾，分別是 X 光圖像和肺部切割圖，在本次研究當中只會使用到 X 光圖像的資料夾而已。

Viral Pneumonia(病毒性肺炎)胸腔 X 光圖像

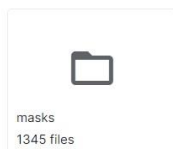
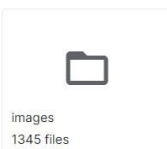
總張數: 1345 張

格式與大小: 均為 PNG 文件格式，解析度為 299*299 像素

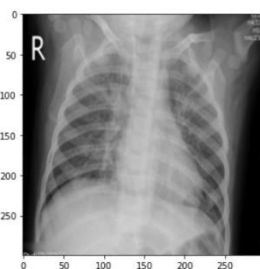
Viral Pneumonia (2 directories)

About this directory

Viral Pneumonia CXR



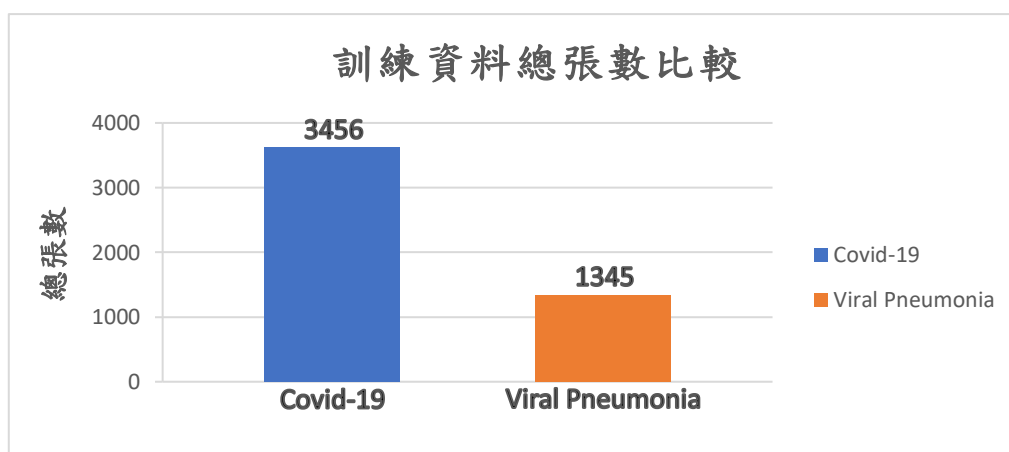
```
[14] import cv2
import matplotlib.pyplot as plt
path = r'./content/drive/MyDrive/Self-Learning/chestXray/chest_xray_data/train_vgs/Viral_Pneumonia/images/'
img = cv2.imread(path)
plt.figure(figsize=(5,5))
plt.imshow(img)
plt.show()
```



(左圖)總張數

(右圖)格式與大小

```
[15] img.shape
(299, 299, 3)
```



3.3 圖像預處理與數據增強

圖像預處理:

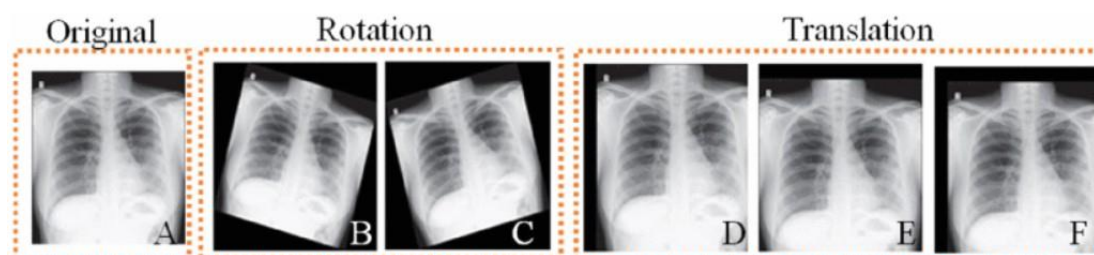
我們需要對數據進行預處理以調整 X 光圖像的大小，讓其能夠符合各種 CNN 模型的輸入影像大小要求。例如:本次實作所使用的 VGG16 模型為 224*224 像素；EfficientNetB1 為 240*240 像素；ResNet125 為 224*224 像素；而 MobileNetV2 為任何大於 32*32 像素的圖像均可，圖像越大則模型的表現越佳。

數據增強:

根據文獻資料，數據增強的目的在於我們能夠透過增強現有數據的方式而不是另外蒐集新的數據，以達到提升深度學習演算法的分類準確性之目的。數據增強可以顯著地增加可用於訓練模型的數據的多樣性，使模型可以適應更多種不同的場景進而增加泛化能力，並且在原始資料較少時能夠擴大訓練集的數量。而當數據集不平衡(數據集中各個類別的數量分布不均衡)時，則更需要使用到數據增強的方式來處理。不過在此次的研究當中，雖然我們 Covid-19 的資料

集與 Viral Pneumonia 的資料集是不平衡的，但我們均對這兩個資料集做同等地擴充。

而我在此次數據增強的部分則是從 `keras.preprocessing.image` 的函式庫當中引入了 `ImageDataGenerator` 的函數，只需要在此函數當中調整參數的模式，便能夠批量的產生經過數據增強的圖像來擴充訓練資料集以及驗證資料集。



(圖 A)原始胸腔 X 光圖 (圖 B)順時針旋轉 15 度後的圖像 (圖 C)逆時針旋轉 15 度後的圖像 (圖 D)往右水平平移 5%後的圖像 (圖 E)往下垂直平移 5%後的圖像 (圖 F)經過水平平移與垂直平移 5%後的圖像

```
[ ] train_datagen = ImageDataGenerator(  
    rescale=1./255, #指定將影像張量的數字縮放  
    shear_range=0.2, #剪切強度(以弧度逆時針方向剪切角度)  
    zoom_range=0.2, #隨機縮放範圍  
    rotation_range=30, #影像旋轉(可填入0到180度的範圍，並以順時針旋轉)  
    width_shift_range=0.2, #水平平移，相對總寬度的比例  
    height_shift_range=0.2, #垂直平移，相對總高度的比例  
    horizontal_flip=True, #隨機水平翻轉  
    preprocessing_function=preprocess_input, #資料預處理函式  
    fill_mode="nearest", #填充新建立畫素的方法  
    validation_split=0.1 #將總資料切分成驗證資料集的比例  
)
```

上圖為透過 Keras 中的函數來進行數據增強，以達到擴充訓練資料的目的

```
train_generator = train_datagen.flow_from_directory(r'/content/drive/MyDrive/Self-Learning/chestXray/chest_xray_data/train_vgg', #目標目錄的路徑  
    target_size=(224, 224), #圖像產生的尺寸大小  
    color_mode='rgb', #顏色模式(可選擇'rgb'或是'grayscale')  
    batch_size=batch_size, #每一批資料的大小  
    class_mode='categorical', #決定返回的标签数组的类型:'categorical'是2D one-hot編碼標籤  
    subset='training' #數據子集類別  
)
```

上圖為讀取資料集中的圖像，並產出我們所設定的格式的批量資料(圖像預處理)以對深度學習模型進行訓練

第四章 深度學習神經網路建構說明

4.1 遷移學習之目的說明

基於資料集、任務(要解決的問題)以及圖像分類等問題的相似性，我們能夠將預訓練好的模型(底部特徵提取層的網路權重)應用於新資料集上進行學習。簡單來說便是開發人員將用於某一項任務已訓練好(參數值固定)的深度學習模型，只使用其中的特徵提取層來提取特徵，然後在這些提取層的基礎上建構新的模型而運用於另一個新的領域或是待解決問題當中。而以下為遷移學習的各項優點與通常用來解決的問題：

1. 處理大量未標記的資料(例如：利用預訓練好的模型來辨識貓狗的照片，並對資料集進行標籤)
2. 降低大量資料的訓練成本：當我們遇到大量性質相似的資料集時，我們能夠利用遷移學習來提高訓練的效率，進而節省時間以及硬體資源等。
3. 醫療應用方面的需求：許多醫療影像會因為隱私權的關係，而使得可運用且具有標籤的資料集樣本稀缺

以下為較嚴謹的定義：

Transfer learning allows you to transfer knowledge from one model to another. For example, you could transfer image recognition knowledge from a cat recognition app to a radiology diagnosis. Implementing transfer learning involves retraining the last few layers of the network used for a similar application domain with much more data. The idea is that hidden units earlier in the network have a much broader application which is usually not specific to the exact task that you are using the network for. In summary, transfer learning works when both tasks have the same input features and when the task you are trying to learn from has much more data than the task you are trying to train.

資料來源：[Deep Learning Specialization by Andrew Ng — 21 Lessons Learned](#)

綜合以上優點且正好本次的自主學習研究符合上述的第三點，因此我便決定利用遷移學習的方法來訓練此次的模型，藉此來提升訓練時的效率，節省訓練時間。而本次我所匯入的預訓練模型為 CNN 經典模型之一的 VGG16 模型。

```
[ ] base_model = tf.keras.applications.VGG16(  
    include_top=False,  
    weights="imagenet",  
)  
x=base_model.output  
x=GlobalAveragePooling2D()(x)  
x=Dense(512, activation='relu')(x)  
preds=Dense(2, activation='softmax')(x)
```

(左圖)匯入預訓練好的 VGG16 模型，並選擇只匯入 13 層卷積層(包含權重值)

(右圖)自行創建我們所需的全連接層(輸出層)

4.2 微調學習之目的說明

微調學習實際上便是遷移學習的一種實現方法。它能夠在訓練我們自己建構的全連接層的同時，重新訓練預訓練模型的頂層，也就是特徵提取層的頂層而得到一個新的權重值。此種作法的好處便是能夠得到更加適合於我們現有資料集的權重，而非單純的使用模型預訓練時所調整出來的權重值，如此一來，我們模型的辨識準確度相對而言也會有所提升。以下為微調學習的幾個特點：

1. 微調學習需要調整原有神經網路的權重，所以我們需要對預訓練模型中的一部分頂層重新訓練，得到更加適合現有資料集的權重。
2. 在預訓練模型當中底層的卷積層往往是實現較簡單且通用的特徵提取，而這些功能在所有的影像上都是通用的。而卷積層的位置越高，則其功能就越是針對於我們所訓練的資料集(相關性越高)，因此通常微調只會選擇重新訓練頂層的部分。
3. 微調學習的目標:讓預訓練模型的頂層更適用於現有的新資料集，而非徹底改變預訓練模型的全部權重。

不過由於本次自主學習所使用的資料集樣本較為稀缺，所需的訓練時間也會相對較少，因此我採用的方法調整為重新訓練整個模型的權重。而與此同時，我也會進行只對我所架構的全連接層重新訓練的測試，而不調整原來預訓練模型中的權重，讓兩者進行比對。結果於第五章中進行分析比較。

```
[ ] # # 遷移學習
# 只訓練我們自己加入的分類層(全連接層)，鎖定原本base_model的權重
# for layer in base_model.layers:
#     layer.trainable = False

# # 模型微調 all
# 解開所有base_model的層來訓練
for layer in base_model.layers:
    layer.trainable = True

# # 模型微調 partial
# 檢查所有層
for i, layer in enumerate(base_model.layers):
    print(i, layer.name)

# # 使用層的數字來決定要讓哪些層可以被訓練
# for layer in model.layers[:10]:
#     layer.trainable = False
# for layer in model.layers[10:]:
#     layer.trainable = True
```

(左圖)

透過程式碼的編寫，來選擇我們想要訓練的神經網路層，例如:只訓練我們架構的全連接層，或是訓練整個神經網路，抑或是選擇只訓練特定的某幾層等

#未來展望：藉由不斷的測試找到哪些神經網路層才需要重新訓練、調整權重值，而剩下的則套用原來的參數進而達到最高的準確度，且降低訓練的時間成本

(上圖)

在未來我們能夠進行更深入的測試，像是找到只需調整權重的特定層，便能達到最高的準確度，藉此來減少訓練時所消耗的資源與時間成本

4.3 預訓練模型 VGG16 介紹

說明:

VGG 是英國牛津大學 Visual Geometry Group 的縮寫，主要貢獻是使用了更多的隱藏層進行大量的圖片訓練，並提高準確率至 90%，因而證明了增加神經網路的深度能夠在一定程度上影響模型的最終性能。其中 VGG 有兩種結構，分別是 VGG16 和 VGG19，而唯一的差別便是在於神經網路的深度不同而已。而 VGG16 之所以為 16 的原因是因為它是由 13 個卷積層加上 3 個全連接層所堆疊出來的，並且每一個卷積層當中的 filter 大小均為 3×3 ，而池化層中的池化核則是採用 2×2 大小。

VGG16 原理:

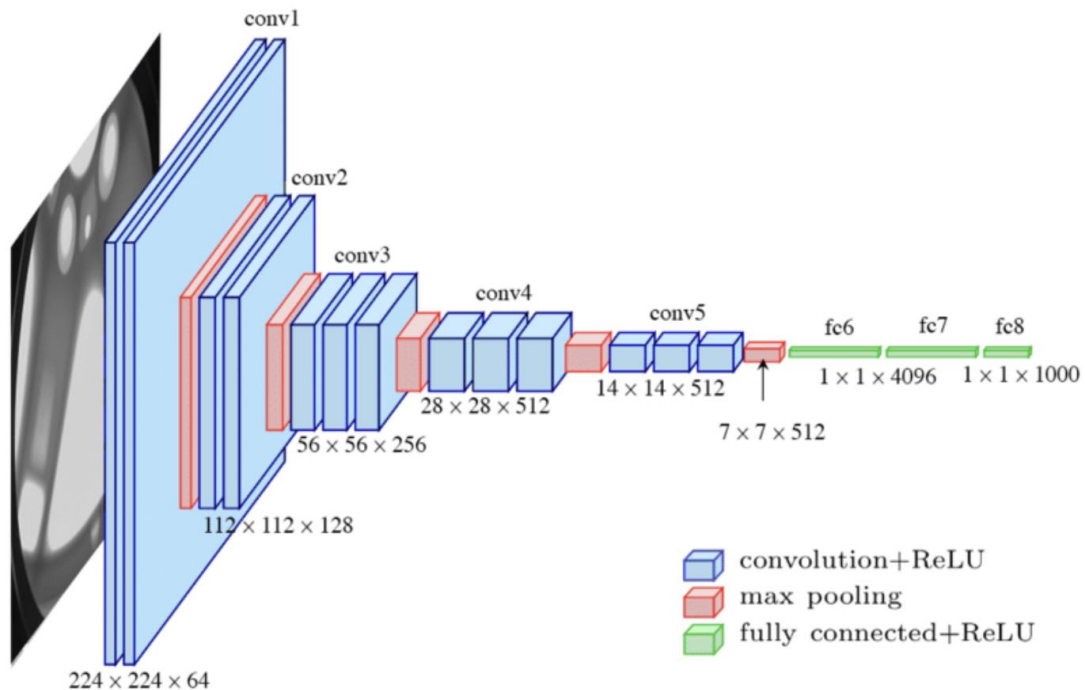
VGG16 採用的是連續的多個 3×3 大小的卷積核來代替某些神經網路內較大的卷積核(像是 11×11 、 7×7 、 5×5)，對於我們給定的感受野(與輸出有關的輸入圖片的局部大小)來說，採用堆積的小卷積核是優於大卷積核的，因為透過多層的非線性層能夠有效地增加神經網路深度來保證學習更為複雜的模式，並且小卷積核所需的參數也比大卷積核還來的更少。以 3×3 和 7×7 為例，3 個步長為 1 的 3×3 卷積核的疊加作用可看成一個 7×7 大小的卷積核，因此 3 個 3×3 卷積核所使用的總參數為 3×9 個參數，而 7×7 卷積核則是使用了 49 個參數，所以我們可以明顯地看出小卷積核大幅地減少了使用參數，進而提升訓練模型的效率。

VGG16 優點:

1. VGG16 的結構十分簡潔，整個神經網路均是使用了同樣大小的卷積核尺寸(3×3)與最大池化核尺寸(2×2)
2. 利用多個小濾波器(3×3)卷積層的組合比一個大濾波器(5×5 或 7×7)卷積層的組合還來的好
3. VGG 驗證了通過不斷加深神經網路結構可以提升性能的理論

VGG16 缺點:

1. VGG 相對其他神經網路會耗費更多的計算資源，並且使用了更多的參數，而導致了更多的內存占用(140M)。其中絕大多數的參數都是來自於第一層的全連接層。
2. 由於具有大量參數的關係，因此訓練整個 VGG 模型時所耗費的時間也是相當龐大的。所幸具有預訓練的 VGG 模型，只需要將其匯入即可得到已調整好的權重值。



上圖為 VGG16 的結構圖，藍色為卷積層，紅色為最大池化層，綠色為全連接層

4.4 深度學習神經網路建模之完整程式碼



以下附上本次自主學習研究的完整程式碼與執行結果

打開連結後為 Colab 執行環境，請另開副本以利進行檢閱

[自主學習_VGG16 辨識模型_Covid and Viral Pneumonia classify](#)



說明:

此程式碼為本次自主學習研究的主軸，目的是要產出一個能夠有效區分出 **Covid-19 的確診病例與病毒性肺炎病例** 的一個深度學習模型。為了要能夠順利訓練出一個有效的模型，各個步驟與處理都是至關重要的，因此我將資料處理到模型產出的整個過程分為以下各個部分:

1. 引入所有所需套件
2. 資料預處理(包含資料剖析與檢查)
3. 學習率模式設定與調整
4. 匯入預訓練模型 VGG16
5. 更改全連接層，並選擇所要訓練的神經網路層
6. 模型的組裝與檢查
7. Callback 回調函數設定
8. 利用訓練集與驗證集訓練與調整模型
9. 利用繪圖套件繪製出訓練成果
10. 利用測試集測試模型準確度

流程講解:

由於我已在各部分程式碼的上方，還有程式碼當中標註了各項程式碼的功能或是作用了，因此在這邊只作步驟說明與講解:

第一步驟:

程式的套件以及函式庫均是由專業人士編寫完成的，內容包含了各種方便的功能，像是矩陣運算、繪製圖表、檔案資料預處理等。當我們要使用到這些功能時，我們就不必每次都要花時間自行編寫，而是只要引入相對應的函式庫就能夠輕鬆使用各項功能了，像是 Keras、TensorFlow 這些都是初學者便能輕易上手的開源軟體庫。

第二步驟:

資料預處理是一個相當重要的環節。假如我們對於手上的資料集不熟悉，抑或是資料集處理的過程不完善，便會直接的影響我們所訓練出來的模型的成效。因此，當我們拿到一個新的資料集時，我們都需要觀察圖像的各個資訊以方便之後進行模型的設置。資料預處理包含了圖像標準化、缺失值處理、數據增強與圖像擴充等，這些處理技巧都能夠讓我們有一個完善的輸入能運用於模型，使我們在訓練模型時擁有更好的表現。

第三步驟:

學習率在訓練模型時扮演著極為重要的角色。它是一個負責控制模型梯度下降的速度的參數。當我們學習率設置過小時，便會導致我們的模型收斂速度較慢，且容易發生過擬合的問題；而當學習率設置過大時，又會容易導致模型學習過快而無法收斂，容易產生震盪甚至是發散的情況。不過一般而言，初始學習率都是依照個人經驗去設置的，因此「試出」一個好的學習率是相當重要的。而我此次的自主學習所使用的學習率是根據我調整的參數，隨著執行步數而改變其大小值，如此一來便能使模型達到更好的收斂效果。

第四步驟:

如同第四章內容中的介紹，我們能夠利用遷移學習的方式來匯入預訓練好的 VGG16 模型。當我們遇到大量性質相似的資料集時，便能夠透過遷移學習的方式來提高訓練的效率，進而節省時間以及硬體資源等。

第五步驟:

一樣如同第四章中的內容，當我們匯入預訓練好的模型後，我們能夠更改其中的全連接層來改變輸出模式。此外利用微調學習的方式，我們能夠選擇我們所要重新訓練的神經網路層，進而使模型更加貼合新的資料集。

第六步驟:

當我們建構完成以及選擇要重新訓練的神經網路層後，我們還須將整個模型整合(組裝)起來，並且設置模型訓練時所要使用的優化器以及損失函數等。通常我們需要經過不斷地測試才能夠找出最佳的優化器，而損失函數則是要根據資料預處理時的資料型態來選擇相對應的損失函數類型。

第七步驟:

Callback 回調函數為一個函數的合集，能夠用來查看訓練模型時的內在狀態和統計資料。我們可以透過調整內部的函數以及設置條件來決定訓練模型時的停止點或是存取狀態，也能夠將訓練的紀錄以 Tensorboard 的檔案格式存取，方便之後進行數據繪圖。

第八步驟:

將我們處理過的訓練集與驗證集輸入進模型當中，使模型進行訓練，並同步地調整其中的參數值。而此次自主學習研究中，我總共利用了兩種訓練方式，分別是遷移學習(只訓練全連接層)以及微調學習(全部重新訓練)這兩種，在接來的數據分析中兩者會進行比對，進而衍生出更多的討論。

第九步驟:

利用他人編寫好的繪圖套件繪製出兩種訓練模型的方式的訓練結果。透過呈現出圖表的方式，我們便能夠清晰地瞭解訓練過程的變化趨勢，進而對模型的參數設定進行調整。除此之外，也能夠讓他人快速地瞭解訓練成果，而不是只有一堆密密麻麻的文字來呈現。

第十步驟:

最後我們還需要檢查模型的訓練成效是否具可信度，作為神經網路模型的最終評估。由於模型於訓練時有可能會對訓練集和驗證集產生過擬合的現象，而導致模型只在訓練時表現良好，事實上在未看過的測試集上卻是天差地遠，所以最終的測試也是至關重要的一環。

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, None, None, 3)]	0
block1_conv1 (Conv2D)	(None, None, None, 64)	1792
block1_conv2 (Conv2D)	(None, None, None, 64)	36928
block1_pool (MaxPooling2D)	(None, None, None, 64)	0
block2_conv1 (Conv2D)	(None, None, None, 128)	73856
block2_conv2 (Conv2D)	(None, None, None, 128)	147584
block2_pool (MaxPooling2D)	(None, None, None, 128)	0
block3_conv1 (Conv2D)	(None, None, None, 256)	295168
block3_conv2 (Conv2D)	(None, None, None, 256)	590080
block3_conv3 (Conv2D)	(None, None, None, 256)	590080
block3_pool (MaxPooling2D)	(None, None, None, 256)	0
block4_conv1 (Conv2D)	(None, None, None, 512)	1180160
block4_conv2 (Conv2D)	(None, None, None, 512)	2359808
block4_conv3 (Conv2D)	(None, None, None, 512)	2359808
block4_pool (MaxPooling2D)	(None, None, None, 512)	0
block5_conv1 (Conv2D)	(None, None, None, 512)	2359808
block5_conv2 (Conv2D)	(None, None, None, 512)	2359808
block5_conv3 (Conv2D)	(None, None, None, 512)	2359808
block5_pool (MaxPooling2D)	(None, None, None, 512)	0
global_average_pooling2d (Gl	(None, 512)	0
dense (Dense)	(None, 512)	262656
dense_1 (Dense)	(None, 2)	1026
Total params: 14,978,370		
Trainable params: 14,978,370		
Non-trainable params: 0		

上圖為此次模型的所有神經網路層

第五章 數據成果分析與問題討論

數據分析說明 I:

以下將依序呈現出微調學習與遷移學習的訓練結果，並進行比較與討論。

<pre>" VGG16 with sparse, train all basemodel layer" Fine tune batch_size= 64 Epoch 1/20 Epoch 00001: LearningRateScheduler setting learning rate to tf.Tensor(1e-04, shape=(), dtype=float32). 55/55 [=====] - 106s 2s/step - loss: 0.3725 - accuracy: 0.8204 - val_loss: 0.3864 - val_accuracy: 0.8016 Epoch 2/20 Epoch 00002: LearningRateScheduler setting learning rate to tf.Tensor(9.045084e-05, shape=(), dtype=float32). 55/55 [=====] - 53s 967ms/step - loss: 0.0957 - accuracy: 0.9686 - val_loss: 0.0427 - val_accuracy: 0.9886 Epoch 3/20 Epoch 00003: LearningRateScheduler setting learning rate to tf.Tensor(6.545085e-05, shape=(), dtype=float32). 55/55 [=====] - 52s 950ms/step - loss: 0.0433 - accuracy: 0.9868 - val_loss: 0.0475 - val_accuracy: 0.9879 Epoch 4/20 Epoch 00004: LearningRateScheduler setting learning rate to tf.Tensor(3.454914e-05, shape=(), dtype=float32). 55/55 [=====] - 50s 900ms/step - loss: 0.0199 - accuracy: 0.9948 - val_loss: 0.0395 - val_accuracy: 0.9906 Epoch 5/20 Epoch 00005: LearningRateScheduler setting learning rate to tf.Tensor(9.549147e-06, shape=(), dtype=float32). 55/55 [=====] - 41s 732ms/step - loss: 0.0195 - accuracy: 0.9954 - val_loss: 0.0333 - val_accuracy: 0.9926 Epoch 6/20 Epoch 00006: LearningRateScheduler setting learning rate to tf.Tensor(8e-05, shape=(), dtype=float32). 55/55 [=====] - 35s 639ms/step - loss: 0.0269 - accuracy: 0.9917 - val_loss: 0.0419 - val_accuracy: 0.9906 Epoch 7/20 Epoch 00007: LearningRateScheduler setting learning rate to tf.Tensor(7.804226e-05, shape=(), dtype=float32). 55/55 [=====] - 35s 636ms/step - loss: 0.0298 - accuracy: 0.9914 - val_loss: 0.0699 - val_accuracy: 0.9805 Epoch 8/20 Epoch 00008: LearningRateScheduler setting learning rate to tf.Tensor(7.236068e-05, shape=(), dtype=float32). 55/55 [=====] - 112s 2s/step - loss: 0.0244 - accuracy: 0.9934 - val_loss: 0.0421 - val_accuracy: 0.9913 CPU times: user 5min 1s, sys: 30.3 s, total: 5min 31s Wall time: 8min 13s</pre>	
微調學習 因為會重新訓練整個模型的每一層，因此訓練時的收斂速度較慢，但最終辨識正確率會最高。 0.9926	
<pre>## 這是training log 請不要執行 ## " VGG16 with sparse, freeze all basemodel layer" Transfer learning batch_size= 64 Epoch 1/20 Epoch 00001: LearningRateScheduler setting learning rate to tf.Tensor(1e-04, shape=(), dtype=float32). 55/55 [=====] - 95s 2s/step - loss: 0.1033 - accuracy: 0.9643 - val_loss: 0.0701 - val_accuracy: 0.9785 Epoch 2/20 Epoch 00002: LearningRateScheduler setting learning rate to tf.Tensor(9.045084e-05, shape=(), dtype=float32). 55/55 [=====] - 42s 753ms/step - loss: 0.0207 - accuracy: 0.9960 - val_loss: 0.0675 - val_accuracy: 0.9792 Epoch 3/20 Epoch 00003: LearningRateScheduler setting learning rate to tf.Tensor(6.545085e-05, shape=(), dtype=float32). 55/55 [=====] - 35s 637ms/step - loss: 0.0211 - accuracy: 0.9951 - val_loss: 0.0677 - val_accuracy: 0.9792 Epoch 4/20 Epoch 00004: LearningRateScheduler setting learning rate to tf.Tensor(3.454914e-05, shape=(), dtype=float32). 55/55 [=====] - 36s 652ms/step - loss: 0.0176 - accuracy: 0.9960 - val_loss: 0.0718 - val_accuracy: 0.9778 Epoch 5/20 Epoch 00005: LearningRateScheduler setting learning rate to tf.Tensor(9.549147e-06, shape=(), dtype=float32). 55/55 [=====] - 113s 2s/step - loss: 0.0209 - accuracy: 0.9945 - val_loss: 0.0708 - val_accuracy: 0.9778 CPU times: user 2min 57s, sys: 14.2 s, total: 3min 12s Wall time: 5min 22s</pre>	
遷移學習 因為僅訓練模型的最後一層，因此訓練時的收斂速度較快，但辨識正確率會略低。 0.9792	

(上圖)為微調學習(全部重新訓練)的訓練結果

(下圖)為遷移學習(只訓練全連接層)的訓練結果

每張圖皆包含了每次訓練時的訓練、驗證損失函數以及訓練、驗證的準確度

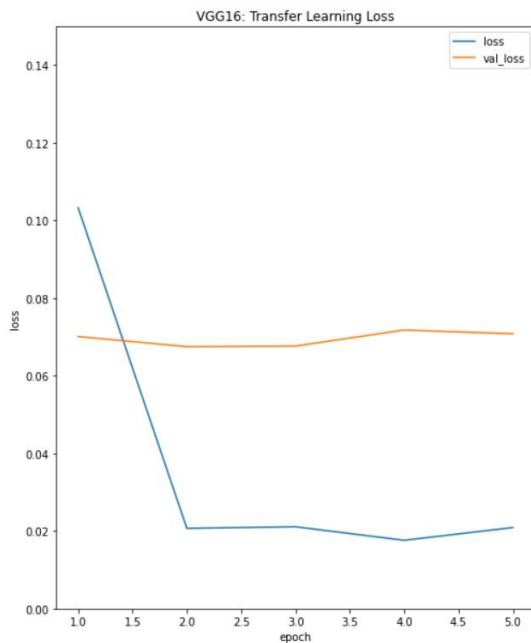
數據討論 I:

從上面兩張圖的數據中可以發現，微調學習的準確度(0.9926)是略高於遷移學習的準確度(0.9792)。這是由於微調學習是重新訓練所有的神經網路層，因此每一層的神經網路均會重新調整其權重值，使此模型能夠更加符合新的資料集，而使得模型在辨識準確度方面相對遷移學習來說擁有更好的表現。但也因為如此，微調學習在訓練時的收斂速度會比遷移學習時來得慢，因此進行訓練的時間也會比較長。

數據分析說明 II:

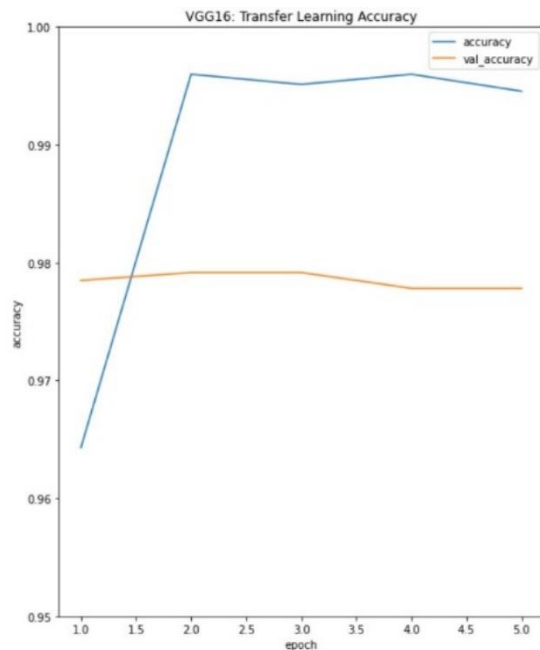
以下將會依次呈現出本次模型的訓練結果折線圖，總共有四張圖，分別是：

1. 遷移學習的訓練和驗證損失函數(圖一)
2. 遷移學習的訓練和驗證準確度(圖二)
3. 微調學習的訓練和驗證損失函數(圖三)
4. 微調學習的訓練和驗證準確度(圖四)



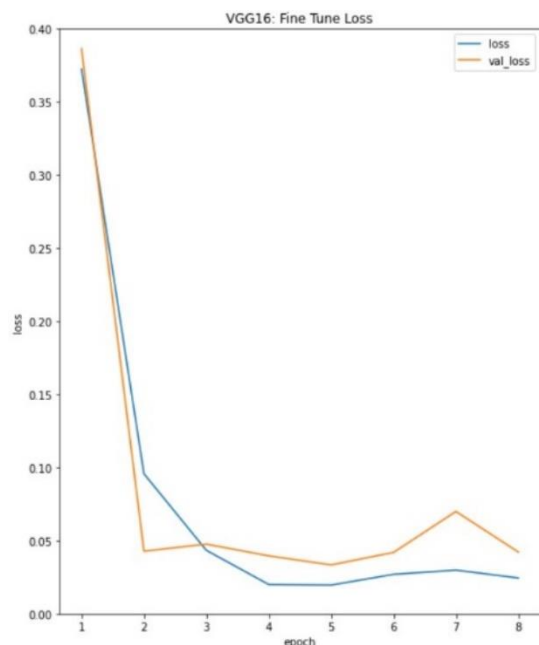
(圖一)

遷移學習的訓練和驗證損失函數



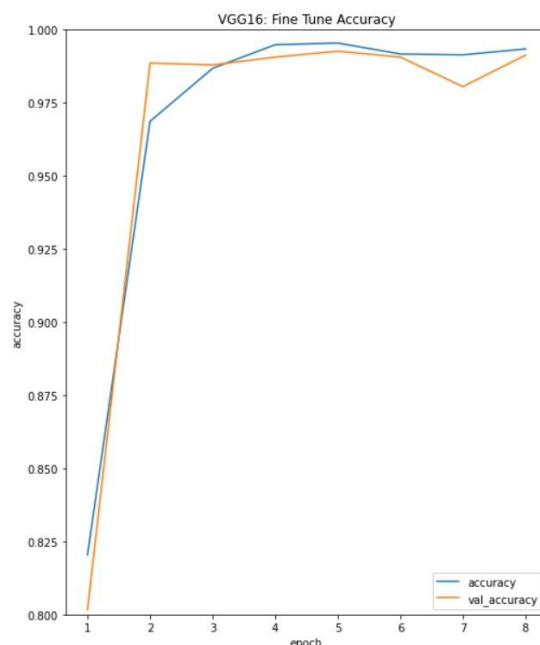
(圖二)

遷移學習的訓練和驗證準確度



(圖三)

微調學習的訓練和驗證損失函數



(圖四)

微調學習的訓練和驗證準確度

數據討論 II:

上面四張圖中的黃色折線均為驗證時的損失函數或是準確度資料，而藍色折線則為訓練時的損失函數或是準確度資料。

圖一：我們可以看到遷移學習驗證的損失函數(黃色)幾乎是呈現水平的狀態，而訓練的損失函數(藍色)則是在第一次訓練時較大，過了第一次訓練以後便大幅下降，並呈現接近水平的狀態。而訓練的損失函數在後來之所以較驗證時低是因為模型是通過驗證的方式來調整權重，使誤差降低，因此訓練的損失函數相較驗證時來說會有較低的輸出。

圖二：我們一樣可以看到遷移學習驗證的準確度(黃色)是維持在高水平的狀態，而訓練的準確度則是在第一次訓練之後大幅上升，並且超越了驗證時的準確度。其中的原理也是相同的概念，由於驗證就像是對模型的訓練成果進行小考，因此經過多次小考並調整之後，訓練的成效也會越來越好。

圖三：我們可以清楚看到微調學習時，不管是驗證還是訓練的損失函數均是在第一次訓練時最高，而後大幅降低，在第三次訓練時，訓練損失函數(藍色)便開始低於驗證的損失函數(黃色)。這個現象是由於我們採用微調學習的方式而導致，因為我們須重新訓練所有的神經網路層，因此權重都仍在進行調整，使得損失函數一開始高，而後才大幅降低。

圖四：我們一樣可以看到準確度的部分也是相同的現象，由於是重新訓練的關係，因此準確度在第一次訓練時表現不佳，而後便迅速增加，到了第三次訓練時，訓練的準確度便保持在驗證的準確度之上。

問題與討論：

在本次製作模型的過程中，由於大部分的內容對於我來說都是沒有接觸過的，因此在查找資料、閱讀網路上的文章與教學，還有實際實作的部分都需要花費龐大的時間與精力去完成，並且將這些資訊清晰地統整起來。過程中也不乏遇到各種問題，像是程式語法、資料處理等，都需要上網閱讀相關的文章與註解，因此以下我將本次製作模型過程中的幾個問題與解決辦法一一的條列說明。

問題一：由於本次自主學習的目標是要能夠產出具有辨識 Covid-19 病例與 Viral Pneumonia(病毒性肺炎)病例的一個深度學習模型，因此最基本的便是要取得 Covid-19 和病毒性肺炎的資料集數據，才能夠對模型進行訓練。不過由於這是醫療影像數據集的關係，因此要找到樣本數且具完整性的資料集實屬困難。

解決辦法一：透過不懈地努力，我發現了一個具有龐大且完整的開源資料集 (Kaggle Datasets)，裡面不僅有研究人員所上傳的資料，而且都是經過整理可直接下載使用的資料集，為我省去了相當多的時間。

問題二：在得到數據集後，資料預處理也是相當重要的一環。不過，由於我還是一個初學者的關係，因此對於資料讀入的程式語法或是觀念都相當薄弱，在一開始訓練模型時，便因此大大地影響了我的模型準辨識確度與運作效能。

解決辦法二：由於訓練成果不佳，因此我便上網查找相關的文章來探討資料預處理的各個過程以及程式語法的編寫。不僅如此，我還因此學到了數據增強以及圖像擴充等技巧，透過 Keras 的官方文檔來了解各個函式庫以及函式的功能，進而使資料集樣本稀缺的問題得以解決，讓模型能夠有更好的訓練結果。

問題三：由於本次選擇的預訓練模型為 VGG16，因此在訓練時所消耗的時間也會相對較長。除此之外，包含多層全連接層的神經網路以及多餘的數據增強設定均會使得模型的訓練效能不佳，而又再次拖慢了訓練模型所需的時間，導致我每次都需要等接近一小時的時間才能夠了解模型的訓練狀況，進而再對模型進行重新的調整。

解決辦法三：經過了多次的測試與資料查詢之後，我發現在全連接層的部分只需要建構一層 GlobalAveragePooling 便能夠替代掉許多層的全連接層，因此大大的降低了參數的使用，使得模型的訓練效率能夠更上一層樓。除此之外，數據增強的部分也會影響訓練時的時間。經測試後發現，水平與垂直的隨機位移事實上對於模型的準確率並沒有太大的提升，而且還間接導致了 GPU 的效能沒辦法發揮到最高效，因此我便將這部分的函數進行了刪減，最終讓訓練的效能有了顯著地提升。

第六章 自主學習歷程與心得反思

6.1 自主學習歷程

在本次自主學習研究當中，我原本的計畫是要讀完四本有關 AI 機器學習的書籍，藉此來增進我對這塊領域的了解，並且有能力自己製作出一個具辨識能力的模型。不過，當我讀完了第一本書後便覺得內容對我來說有點難度，有些觀念仍是相當模糊，而且程式編寫的教學也是少之又少，因此我便決定改變學習的策略，轉而直接上網檢閱相關的範例程式碼，同時查詢各個部分的程式碼的運作模式與功能等。但計畫永遠趕不上變化，我的想法馬上就被一行行艱深的程式碼給潑了冷水。我這才發覺對於毫無基礎的我來說要能夠讀懂解釋與每一行程式碼仍是一項挑戰。

自此我對於我的自主學習成果感到相當的茫然。而由於我在學校資研社的關係，所以我便向有做過深度學習專題的學長請教了相關的程式碼與建構的一些建議和想法，讓我有了一些許的方向。而我認為最重要的便是我爸爸的幫助，當他聽到我陷入製作瓶頸時，他便提供了我[政大開放式課程影音網上蔡炎龍老師的課程講解](#)，內容不僅包括了深度學習的各項重要概念，還示範了程式碼建構的過程，讓我有能力自己去建構本次的模型，並且擁有清晰地架構與觀念，讓我在閱覽他人文章或是程式碼時更加順利。

而建構程式碼的過程也耗費了我大量的時間，大多我都在上網查找程式碼的講解，抑或是函式庫和套件的各個功能與設定等。不過，在這個不斷地查詢資料、閱讀文檔的過程當中，我也明顯地發現了我的進步，慢慢地我閱讀他人文章的速度不斷加快，各個專有名詞也已深深地烙印在我的腦海裡，從毫無觀念與技巧再到熟悉與深刻，我認為這個過程便是本次自主學習最大的收穫。

而程式碼建構完成後的最大困難點便是提升模型的準確度以及訓練時的執行效率。對於這些問題，我需要重新地檢視我的模型設定與各行程式碼。與此同時我也向媽媽的同事請教了優化模型的辦法，他不僅提供了我許多的建議以及實作辦法，還幫我檢視了程式設定的不足之處。經過了不斷地討論與嘗試之後，最終遷移學習模型的準確度與微調學習模型的準確度分別達到了 **97.92%**和 **99.26%**，均是相當不錯的成果。

6.2 心得反思

我認為要能夠自己建構出一個完整的模型包括了程式碼編寫的能力以及深度學習的概念與實作技巧。對於初學者的我來說，直接進入實作的過程並不是一個明智的選擇，而是應該要從最一開始的深度學習理論與程式碼編寫技巧開始訓練和學習，而不至於白白浪費掉寶貴的時間，導致中間毫無成效的摸索。

過程中的種種困難也讓我不斷地去嘗試和學習，從上網查詢各類文獻資料的過程當中，我漸漸地了解了深度學習內各類的專有名詞、原理以及程式寫法。除此之外，我也透過了開放式影音平台上的教學影片清晰地認識了深度學習這塊領域，讓我最終能夠完成此次的自主學習研究。

雖然整個自主學習研究花了我許多的時間嘗試與摸索，但我認為只要抱持著一顆熱忱的心便會擁有走下去的無限動力，最終才能夠產出如此完美的成果，並且從中學到了受用一生的知識與技巧。而藉由此次的研究也讓我對深度學習產生了無比的熱忱，希望在未來能夠進行更深入的研究。

第七章 結論

在本次自主學習研究當中，我總共訓練出了兩種模型，分別是遷移學習(僅訓練全連接層)的模型，還有微調學習(全部重新訓練)的模型。

其中遷移學習的模型辨識準確率高達 97.92%，而微調學習的模型辨識準確率更是高達 99.26%，經過測試集的最終測試後，其準確率高達 100%。這是由於我們準備的測試集資料的樣本數較少，抑或是可能測試集與訓練集的樣本十分接近而導致準確率能夠高達 100%。為了要再次檢驗此次自主學習研究的模型是否真的具有可信度以及實用性，因此我將訓練集以及驗證集再次匯入模型當中進行檢驗，結果準確度分別是 99.37%和 99.13%，可見本次的模型成果是相當成功的，說不定在未來也能夠實際應用於醫療產業上。

而在未來，我也希望能夠對模型進行更深入的測試，像是匯入不同的預訓練模型來比較訓練出來的準確度，進而找出一個最能符合此資料集的模型和神經網路，抑或是在訓練時，找出只需調整權重的特定層，便能達到最高的準確度的研究，藉此來減少訓練時所消耗的資源與時間成本，這些都是我們能夠在未來進行改善或是嘗試的方向。

第八章 參考資料

Kaggle Datasets:

[Kaggle 介紹](#)

[Kaggle 入門](#)

[\[資料分析&機器學習\] 第 1.3 講：Kaggle 介紹](#)

COVID-19 Radiography Database:

[Can AI Help in Screening Viral and COVID-19 Pneumonia?](#)

[Exploring the effect of image enhancement techniques on COVID-19 detection using chest X-ray images](#)

Colab:

[Google Colab 教學！新手 Python 開發環境推薦【新手 Python 練習】](#)

[Google Colaboratory - 適合 Python 初學者的雲端開發環境](#)

[Google Colaboratory 筆記](#)

圖像預處理:

[圖像預處理 Keras 文檔](#)

[訓練模型-Keras Application 重要函數](#)

[資料預處理](#)

[資料預處理和特徵工程](#)

[機器學習之 資料預處理 \(sklearn preprocessing\)](#)

數據增強:

[【深度學習】ImageDataGenerator 的使用](#)

[用 Keras API 實作資料強化 Data Augmentation](#)

遷移學習和微調學習:

[使用 Tensorflow 實現遷移學習和微調\(構建 CNN 影像分類模型\)](#)

[訓練模型-遷移學習](#)

VGG16:

[深入理解 VGG16 模型](#)

[VGG16 結構圖](#)

[VGG16 和 VGG19 介紹](#)