中華民國第 59 屆中小學科學展覽會

作品說明書

高級中等學校組　電腦與資訊學科

052506

使用 AI 科技之肺炎診斷

學校名稱：高雄市立高雄高級工業職業學校

作者：	指導老師：
高二 蘇廷恩	王志偉
高二 黃亮維	蔡振貴
高二 林子皓	

關鍵詞：機器學習、人工神經網路、 醫療成像判斷

壹、摘要

此次科展製作，我們團隊使用了機器學習的方式進行肺炎診斷，共分成了兩個階段，首先將肺部 X 光的圖片預處理，接著再將預處理後的 X 光圖輸入神經網路，以得到是否罹患肺炎的判斷結果。此方法可使誤判率大量減少，也讓醫師的判斷時間大幅減少，平均醫師判定一張 X 光平均需要 5 分鐘，而機器僅需數秒即可完成，使醫療效率增加。

貳、研究動機

在這個時代，醫療技術日新月異，但肺炎仍位居國人十大死因中的第三名。因此我們萌發出了一個想法，為何不使用資訊技術來協助醫療呢？因此我們做了一個神經網路協助醫師診斷，判斷病患是否有肺炎，使醫療效率最大化，並減少誤診及漏診等狀況的發生機率，以增進人民福祉。

參、研究設備及器材

硬體：

電腦一：

處理器：Intel® Core™ i5-7400 CPU @ 3.00GHZ

RAM：8.00GB

顯示卡：Nvidia GeForce GTX 1050 2G

電腦二：

處理器：Intel® Core™ i5-4460 CPU @ 3.20GHZ

RAM：12.00GB

顯示卡：Nvidia GeForce GTX 960 2G

電腦三：

處理器：Intel® Core™ i5-2520m CPU @ 2.50GHZ

RAM：8.00GB

顯示卡：Intel® HD Graphics 3000

軟體：

作業系統：Windows 10

整合開發環境：Visual Studio Code、PyCharm

使用語言：Python

模型框架：Keras、TensorFlow

肆、研究方法

一、機器學習框架

(一)、TensorFlow

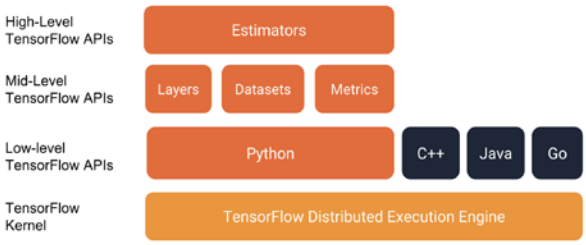


圖 1：TensorFlow 結構

TensorFlow 是 Google Brain Team 所開發的開源軟體庫，用於機器學習及深度學習，使用 Python、C++ 及 CUDA 所編寫，低層核心引擎由 C++所實現，如圖 1。被許多團隊用於研究及開發商業產品，如語音辨識、機器翻譯 及 圖像辨識等，其中也包括了 AlaphGo 和 AlphaGo Zero。

(二)、Keras

Keras 是基於 TensorFlow 和 Theano 所製作的深度學習框架，特點是程式碼非常簡潔、編寫快速及易於理解，相較於 TensorFlow ，Keras 較適合想要快速入門的初學者，圖 2 是 Keras 的範例。

```
model = Sequential()  # 新增一初始模型
model.add(Dense(128, input_shape=(1, 784),
                activation='relu'))  # 第一層隱藏層
model.add(Dense(32, activation='relu'))  # 第二層隱藏層
model.add(Dense(10, activation='softmax'))  # 輸出層
```

圖 2：Keras 實現四層全連接層

二、機器學習演算法

(一)、梯度下降

在機器學習中，梯度下降法是為了尋找一個最佳解，而最佳解就是模型輸出結果與實際答案相同，也就是與實際答案沒有誤差。透過調整參數來減少誤差，並以計算誤差後當前狀況的斜率來決定是否調整變數，圖 3 是使用均方誤差(MSE) 計算後的立體圖，設定最佳結果為 [0.2, 0.5]，a 和 b 均為變數且初始值為[2, 4.5]，欲使 a、b 調整為[0.2, 0.5]（紅色線條為變數調整紀錄）。

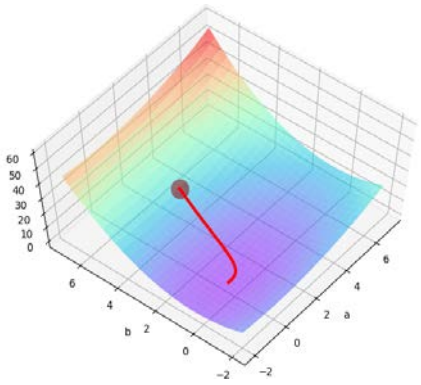


圖 3：梯度下降演示立體圖

(二)、全連接層

全連接層（Fully-connected layer）最常用於機器學習的輸出層，有分類器的作用，可運算特徵的加權和。由神經元組成，每個神經元都有個 W(weight 權重) 和 b(bias 偏差值)，兩者皆為變數。大多都會搭配激勵函式使用。圖 4 為全連接神經網路結構圖，圖 5 為加上激勵函式 ReLU 後的方程。

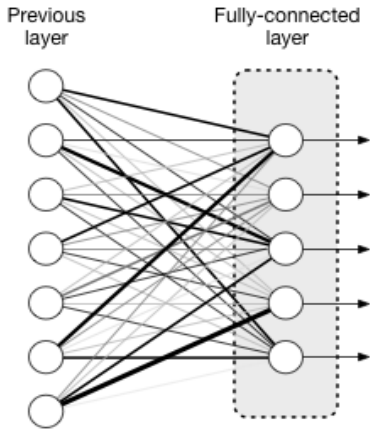


圖 4：全連接層

$$\sum_{i=1}^{inputsize} ReLU\left(W_i x_i\right) + b$$

圖 5：全連接層結合 ReLU 運算方程

(三)、二維捲積層

捲積層（Convolutional layer）由許多的濾波器組成，設濾波器中的變數為 W，則濾波

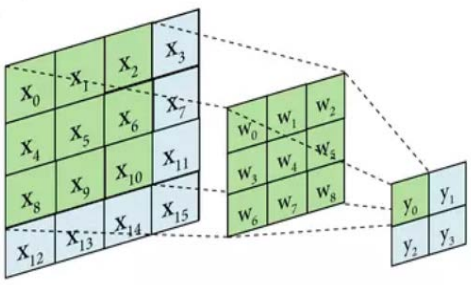


圖 6：二維捲積運作原理

器運算式為 $\sum_{i=1}^{kenrel\ size^2}(W_i * x_i)$ 每做完一次運算後就會往下一格移動，直到整張圖片掃描完畢，如圖 6 所示。

捲積層對於圖像辨識是非常重要的存在，可以經由訓練學習提取出一些重要特徵，藉由帶有變數的正方形濾波器，過濾出圖像的顏色、邊緣、細節等，如圖 7 所示。

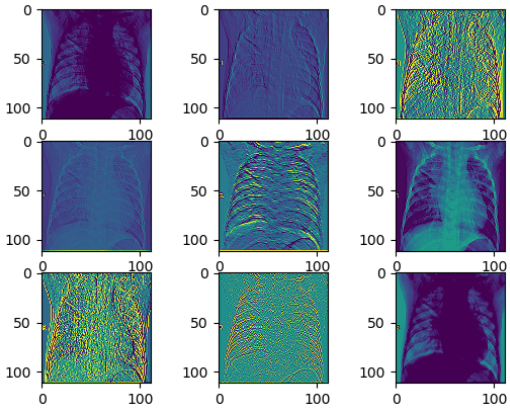


圖 7：捲積後圖片特徵

(四)、Depthwise Separable Convolution

隨著深度學習的發展，模型變得非常巨大，使用的變數也變得非常可觀，但電腦的效能不一定能負荷這麼龐大的運算量，而深度學習絕大部分的運算都是在捲積中，因此研發了 Depthwise Separable Convolution 這種新的捲積方法來減少電腦的負擔，並加快運算效率。

在原本 Convolutional 捲積中，原圖會乘上所有的 kernel map，如圖 8 所示，這在反覆捲積後會造成非常龐大的運算。而在 Depthwise Separable Convolution 中，所有的圖都會有相對應的 kernel map，如圖 9 所示，這樣的設計使運算量大幅減少，且準確度也不會因此降低。

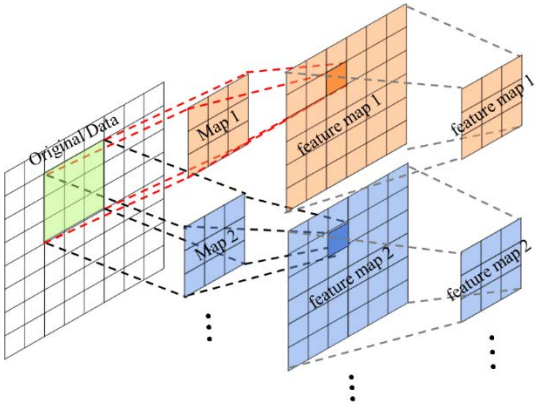


圖 8：原捲積方式

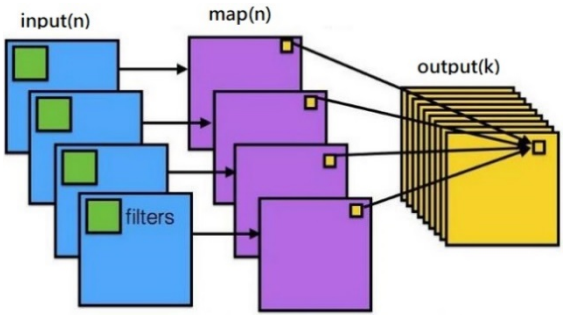


圖 9：Depthwise Separable Convolution

設輸入資料: $\mathrm{W}_{in} * \mathrm{H}_{in} * N_{ch}$

Kernel Map: $\mathrm{k} * \mathrm{k} * N_k$

輸出資料: $\mathrm{W}_{out} * H_{out} * N_k$

在原本的 Convolutional 中計算量如下

$\mathrm{W}_{in} * \mathrm{H}_{in} * N_{ch} * \mathrm{k} * \mathrm{k} * N_k$

Depthwise Separable Convolution 的計算量

$\mathrm{W}_{in} * \mathrm{H}_{in} * N_{ch} * \mathrm{k} * \mathrm{k} + N_{ch} * N_k * \mathrm{W}_{in} * \mathrm{H}_{in}$

由上述兩個式子我們可以推導出原 Convolution 和 Depthwise Separable Convolution 計算量的差距如圖 10。

$$\frac{\mathrm{W}_{in} * \mathrm{H}_{in} * \mathrm{N}ch * k * k + \mathrm{N}ch * \mathrm{N}k * \mathrm{W}\_in * \mathrm{H}\_in}{\mathrm{W}\_in * \mathrm{H}\_in * \mathrm{N}ch * k * k * \mathrm{N}k} = \frac{1}{\mathrm{N}k} + \frac{1}{k * k}$$

圖 10：Depthwise Separable Convolution 與原 Convolution 運算量差距

從圖 10 中，我們可以得知當 Kernel Map 越大和數量越多時，Depthwise Separable Convolution 節省的時間也會跟著增加。例如，當 kernel 形狀為 3*3*256 時，Depthwise Separable Convolution 僅有一般捲積的$\frac{1}{256} + \frac{1}{9} = 0.115$倍計算量。

(五)、全局平均池化

全局平均池化(Global Average Pooling)，又簡稱為 GAP，是為了解決全連接層的問題，首次出現在論文 Network In Network（2013），主要是將捲積過後的各張特徵圖裡的像素值做平均的動作，使網路結構做正則化以防止過擬合的問題，如圖 11。可取代全連接層，以避免全連接層參數過多造成過擬合及訓練時間過長。

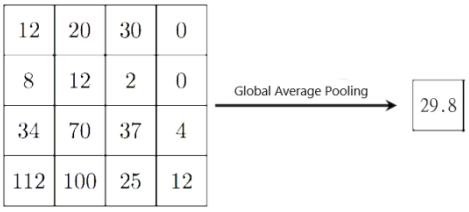


圖 11：全局平均池化運作原理

(六)、激勵函式

如果沒有非線性函數，輸入與輸出的關係只是一組線性的組合，跟最原始的感知器（Perceptron）一樣。激勵函式是藉著非線性的特性讓神經網路不單只能判斷一個點是否大於或小於一個值來分辨，而是判斷出是否存在於指定區域。藉由這個特點，使它跟真實的神經更相似。本次所使用的是 Sigmoid 及 ReLU 兩激勵函式，Sigmoid 方程式為$\mathbf{Sigmoid(x)} = \frac{1}{1+e^{-x}}$，而 ReLU 方程式為 $\mathbf{ReLU(x)} = \mathbf{max(0,x)}$，圖 12 及圖 13 各為激勵函式 ReLU 及 Sigmoid。

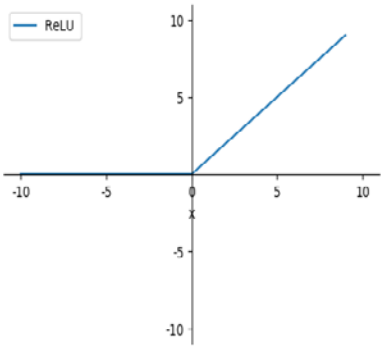


圖 12：$\mathbf{ReLU(x)} = \mathbf{max(0,x)}$

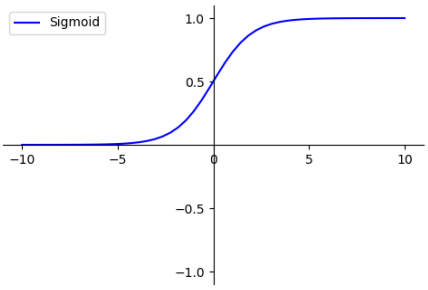


圖 13：$\mathbf{Sigmoid(x)} = \frac{1}{1+e^{-x}}$

(七)、誤差函數

誤差函數（Loss Function）負責誤差的計算，對於梯度下降來說是不可缺的條件，藉由誤差函數可生成梯度，不同的誤差函數會改變梯度及神經網路的訓練速度及收斂速度，進而影響準確率，本次所使用的誤差函數為 Binary Cross-Entropy，適用於僅需分類兩種結果的神經網路之訓練，誤差計算式為 $\mathbf{Loss} = -\sum_{i=1}^{n}(y_{true(i)} * \log(y_{pred(i)}) + (1 - y_{true(i)})\log(1 - y_{pred(i)}))$，其中$y_{pred}$為神經網路輸出結果，$y_{true}$為真實標籤，n 為輸出結果數。

(八)、反向傳播

反向傳播（Back Propagation），是由博士生 Paul Werbos 在論文中提出，其想法是為了改善神經網路的變數更新方式，被大量用於監督式學習的分類訓練，此算法圍繞在梯度及鏈式法則的應用，分為兩階段：

第 1 階段：激勵傳播

1.將訓練資料輸入神經網路以取得輸出，稱為前向傳播

2.將前向傳播之輸出和正確標籤帶入損失函數以求出誤差

第 2 階段：權重更新

對於每個神經元上的權重照以下步驟更新：

1.使用鏈式法則求出各權重之影響及梯度

2.將此梯度乘上一指定參數並取反值（因目標是將誤差縮小，而非增加）

(九)、RMSprop

優化器（Optimizer）控制著梯度下降的學習效率及誤差收斂方向，使用適合的優化器可使訓練結果最佳化、減少訓練所使用的時間，本次實驗使用 Root Mean Square prop（RMSprop）於 2012 年首次被 Geoffrey Hinton 發表，以反向傳播的概念來做為變數更新的方法，與以往的優化器相比，RMSprop 可使每個變數有著各自對應的學習效率，並以觀察過去的梯度來決定變數的更新值，其計算方式如下：

設x為輸入、y為輸出目標、神經網路輸出方程為$f(x)$、衰減速率為γ、變數為θ、學習效率為η、常數λ(使計算不產生錯誤，通常為$1e-6$)、t 代表目前狀態

設定損失函數：$Loss(x) = -\sum_{i=1}^{1}(f(x) * \log(y) + (1 - f(x))\log(1 - y))$

計算梯度：$g_t = Loss(x)$

累積平方梯度：$E[g^2]_t = \gamma \cdot E[g^2]_{t-1} + (1-\gamma) g_t \cdot g_t$

計算變數更新量：$\Delta\theta = -\frac{\eta}{\sqrt[2]{\lambda + E[g^2]_t}} \cdot g_t$

變數應用更新：$\theta_{t+1} = \theta_t + \Delta\theta$

(十)、批正則化

批正則化（BatchNormalization）是將數據分布歸一化成平均值為 0，方差為 1 的分布。在訓練神經網路時，各層輸出值時常因過大而造成下層變數無法適度的收斂，在

層與層之間加上批正則化後，各輸出差距範圍減少，可使變數收斂加速，方程為：

$$y_n = \frac{x_n - \frac{\sum_{i=1}^{\text{input size}} x_i}{\text{input size}}}{\sqrt[2]{\frac{1}{\text{input size}}\sum_{i=1}^{\text{input size}}(x_i - \bar{x})^2}}$$ ，其中x為輸入。

(十一)、過擬合

過擬合是一種訓練神經網路時產生的過度適應訓練資料的情形，如圖 14，產生的原因有：

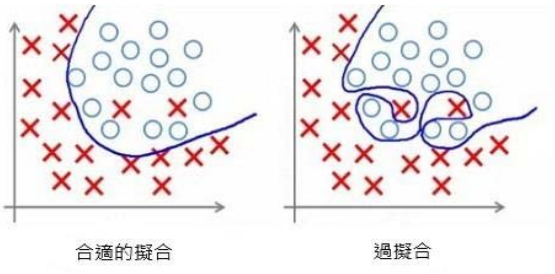


圖 14：過擬合情況

1、訓練的數據不足

數據不足時，很多重要的特徵會被忽略，結果的可信度也會降低，進而造成圖形的辨識錯誤。

2、訓練的數據有雜訊

有雜訊的訓練數據會造成神經網路的學習到錯誤的特徵，使準確率降低。

3、從訓練的數據擷取過多特徵

當程式從訓練數據擷取過多非必要特徵時，會造成神經網路對於訓練資料有過度的專一性，有許多正確的特徵被判斷成錯誤，進而造成低準確率。

有以下幾種方法可以避免過擬合發生：

1. 提早停止訓練

提早停止訓練可避免模型過於擬合於訓練數據及一些非必要特徵，避免準確率下降。

2. 丟棄法

丟棄法（dropout）是防止過擬合最常用的方法，將神經元的變數以特定機率暫時歸零，使神經網路不會過度依賴於某個神經元的結果，避免神經元使用錯誤特徵進行判斷。

3. 交叉驗證

交叉驗證（Cross-validation）是在每批訓練後，以少量測資進行測試，並記錄準確率，可從記錄中判斷何時開始出現過擬合。

4. 正則化及正規化

正則化（Normalization）是將數據做預處理，從而使數據歸一化，減少數據之間的差距，使模型收斂時更有效率，並避免因賦予同個神經元過多的權重所造成過擬合。而正規化（Regularization）是對於反向傳播時的變數變化量做限制，藉由損失函數上添加懲罰項，以限制訓練時的變化量，進而使模型傾向簡單化，避免因模型過於複雜所造成過擬合。

(十二)、Class Activation Mapping

Class Activation Mapping （CAM），是一種使捲積神經網路輸出可視化的方法，可產生熱度圖，用以觀察神經網路的判斷重點。要使用 CAM 首先要將輸出層中的權重取出，並與最後捲積層輸出之特徵圖相乘，以產生熱度圖，最後調整透明度並和原圖重疊，流程圖如圖 15。

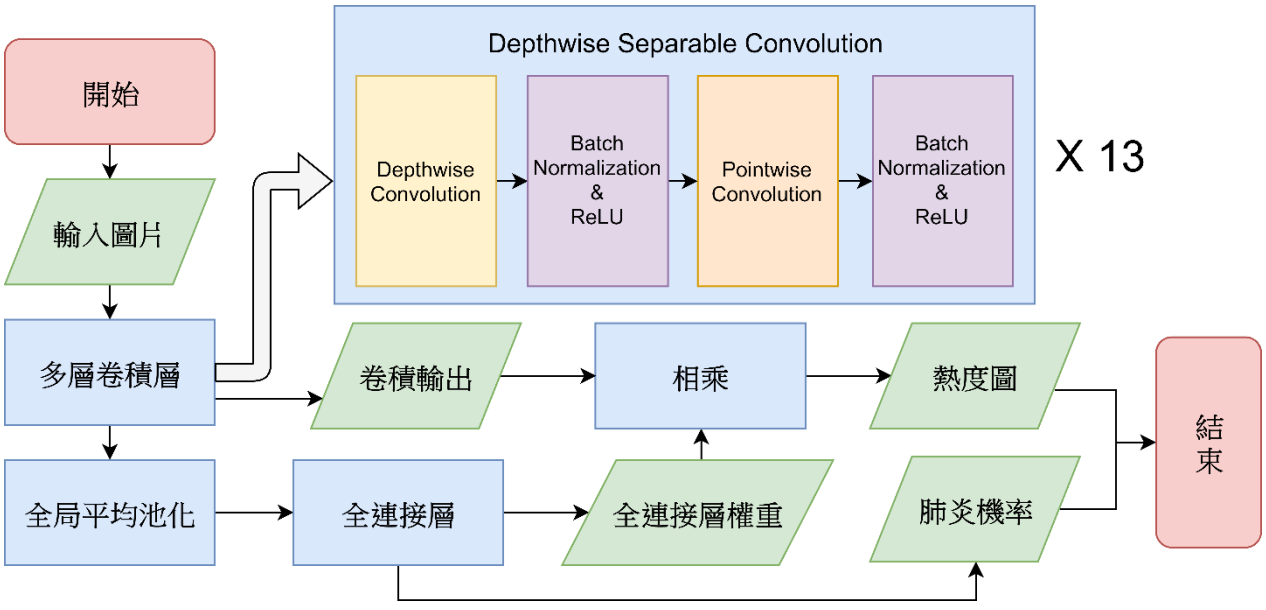


圖 15：熱度圖產生流程圖

三、硬體

(一)、處理器

我們使用 Intel Core i5-4460 做為處理器（CPU），雖然在機器學習訓練的運算速度上，跟中高階的顯示卡比起來較為遜色，不過在數據的前置處理上的處理還是要依賴於處理器。也可以搭配 Intel Distribution for Python 來增加 Python 的運行速度，使神經網路訓練更快。表 1 為 Intel Core i5-4460 的詳細資訊。

表 1：Intel Core i5-4460 詳細資訊

效能	核心數量	4
	執行緒數量	4
	處理器基本頻率	3.2G Hz
	最大超頻	3.4G Hz
	快取記憶體	6MB SmartCache
	散熱設計功率（TDP）	84W
記憶體規格	最大記憶體大小 （取決於記憶體類型）	32GB
	記憶體類型	DDR3-1333/1600, DDR3L-1333/1600 @ 1.5V
	最大記憶體通道數量	2
	最大記憶體頻寬	25.6 GB／s

(二)、顯示卡

我們所使用的顯示卡（GPU）是 Nvidia GeForce GTX 960，因為 TensorFlow 支援 Nvidia 所推出的 CUDA（Compute Unified Device Architecture）整合技術和 cuDNN（CUDA Deep Neural Network library）深度神經網路加速庫，此技術可使用於 GeForce 8

系列後的顯示卡，可大幅加速神經網路的訓練速度。表 2 為 Nvidia GeForce GTX 960 的詳細資訊。

表 2：Nvidia GeForce GTX 960 詳細資訊

引擎規格	CUDA 核心	1024
	基本時脈	1127M Hz
	加速時脈	1178M Hz
	貼圖填充率	720 億／秒
記憶體規格	記憶體時脈	7Gbps
	標準記憶體配置	2GB
	記憶體介面	GDDR5
	記憶體界面寬度	128-位元
溫度與功率規格	GPU 最高溫度	98 C
	繪圖卡功率	120W
	建議的系統電源	400W

四、架設神經網路

(一)、神經網路結構

近年捲積神經網路在 ImageNet 競賽中，為了追求分類準確度，讓模型深度愈來愈深愈複雜，但在實際層面上的應用，這些大且複雜的模型是難以駕馭，為了讓模型在實際上能夠應用，目前研究方向分成兩個方向：

1. 對訓練好的複雜模型壓縮得到縮小的模型
2. 直接設計小模型訓練

Google 於近期提出了一種小巧且高效的捲積神經網路模型，其名為 MobileNet，屬於上述之第二者。我們將此模型結構修改為適用於判斷罹患肺炎的 X 光片，並搭配遷移學習進行訓練。

我們使用步數為 2 的 Depthwise Convolution 取代池化，改善池化會丟失部分特徵的缺失，經由捲積後圖片輸出得到 1024 張 7 * 7 的特徵，之後使用 Global Average Pooling 將每張圖片平均運算，使輸出圖片變成 1 * 1 * 1024 的陣列，再由全連接層輸出肺炎之機率，結構如表 3。

表 3：此次科展所使用 MobileNet 結構表

Layer	Filter Shape	Input Size
Conv / s2	3 * 3 * 3 * 32	224 * 224 * 3
Conv dw / s1	3 * 3 * 32 dw	112 * 112 * 32
Conv / s1	1 * 1 * 32 * 64	112 * 112 * 32
Conv dw / s2	3 * 3 * 64 dw	112 * 112 * 64
Conv / s1	1 * 1 * 64 * 128	56 * 56 * 64
Conv dw / s1	3 * 3 * 128 dw	56 * 56 * 128
Conv / s1	1 * 1 * 128 * 128	56 * 56 * 128
Conv dw / s2	3 * 3 * 128 dw	56 * 56 * 128
Conv / s1	1 * 1 * 1 * 128 * 256	28 * 28 * 128
Conv dw / s1	3 * 3 * 256 dw	28 * 28 * 256
Conv / s1	1 * 1 * 256 * 256	28 * 28 * 256
Conv dw / s2	3 * 3 * 256 dw	28 * 28 * 256
Conv / s1	1 * 1 * 256 * 512	14 * 14 * 256
5 * Conv dw / s1 Conv / s1	3 * 3 * 512 dw 1 * 1 * 512 * 512	14 * 14 * 512 14 * 14 * 512
Conv dw /s2	3 * 3 * 512 dw	14 * 14 * 512
Conv / s1	1 * 1 * 512 * 1024	7 * 7 * 512
Conv dw / s2	3 * 3 * 1024 dw	7 * 7 * 1024
Conv / s1	1 * 1 * 1024 * 1024	7 * 7 * 1024
Avg Pool / s1	Pool 7 * 7	7 * 7 * 1024
FC / s1	1024	1 * 1 * 1024

(二)、圖片預處理

在訓練之前，先把圖片縮小為 224*224，以符合神經網路之輸入大小，圖片如圖 16。再把圖片的像素點之值除以 255（因為圖片的每個像素點之值介於 0~255 之間），處理後圖片的每個像素點之值在 0~1 之間，使梯度下降時可以更快收斂。圖 17 為圖片預處理之主要程式碼。

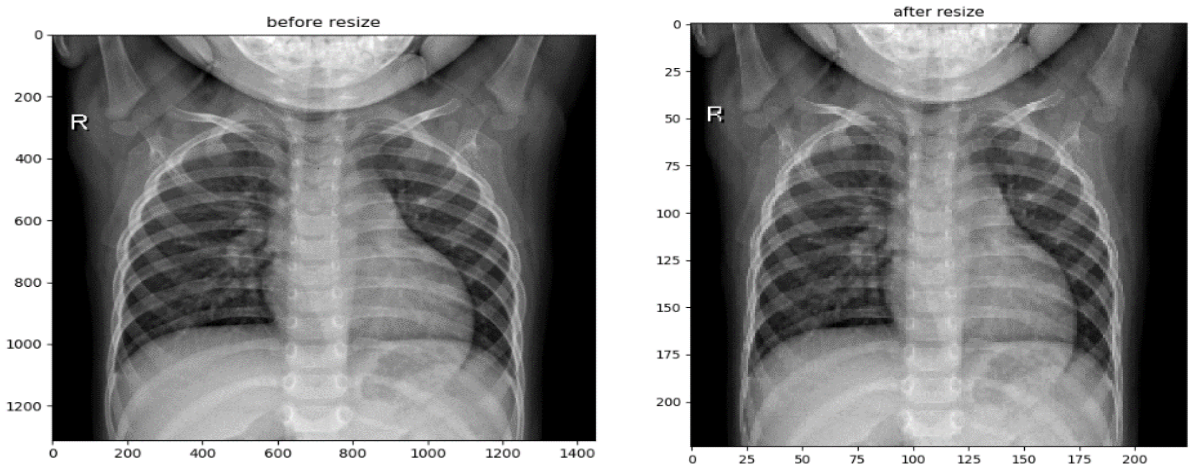


圖 16：圖片重新塑形後結果

```
img_path = r"D:\Users\Administrator\Pictures\datasets\Pneumonia"

datagen = ImageDataGenerator(rescale=1. / 255)

train_data = datagen.flow_from_directory(img_path + '\\train', target_size=(224, 224),
                                         classes=['NORMAL', 'PNEUMONIA'],
                                         batch_size=16)
validation_data = datagen.flow_from_directory(img_path + '\\val', target_size=(224, 224),
                                              classes=['NORMAL', 'PNEUMONIA'],
                                              batch_size=16)
test_data = datagen.flow_from_directory(img_path + '\\test', target_size=(224, 224),
                                        classes=['NORMAL', 'PNEUMONIA'],
                                        batch_size=16)
```

圖 17：圖片預處理之主要程式碼

(三)、訓練過程

1. 第一次訓練

第一次訓練使用了 15360 張圖片進行訓練，圖 18 為訓練經網路之主要程式碼，訓練了 30 批，每批 512（16*32）張，使用了批正則化 及 全局平均池化以避免過擬合之發生，並使用交叉驗證以觀察過擬合，結果如圖 19 所示。

```
train_step = model.fit_generator(train_data, steps_per_epoch=32, epochs=30,
                                 validation_data=validation_data, validation_steps=50,
                                 )
```

圖 18：第一次訓練之主要程式碼

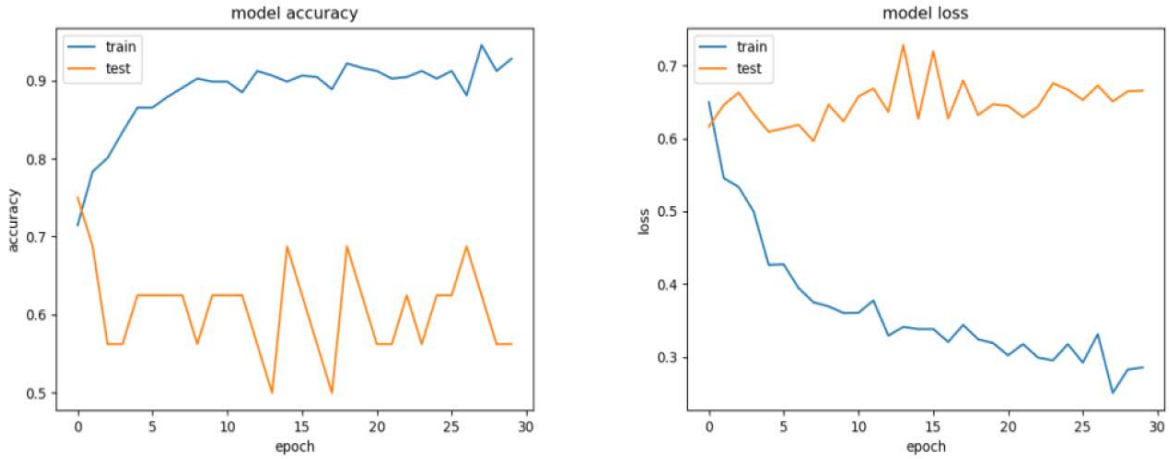


圖 19：第一次訓練準確率及誤差

由圖 19 可觀察到，於第 7 批後，雖然訓練誤差及準確率有明顯進步，但測試誤差不減反增，且測試準確率也沒有提高，可知於第 7 批後開始出現了過擬合。

2. 過擬合之原因及解決

由第一次訓練的結果推斷，造成的原因可能為過量的訓練以造成模型對於訓練數據過度擬合，無法判斷正確的特徵，從而使其無法順利進行泛化。為了降低過擬合的發生率，我們使用了 dropout 及 正規化，並將訓練批數減少至 7 批，使訓練時發生過擬合的機率大幅減少。

3. 第二次訓練

第二次訓練時為避免過擬合，只使用了 3584 張圖片，訓練了 7 批，每批也是 512 張，並且使用 dropout 及正則化於輸出層上，以降低過擬合之發生率，結果如圖 20。使用 4000 張僅用於測試的圖片進行測試後，得到了 91.58%的準確率，主要程式碼如圖 21。

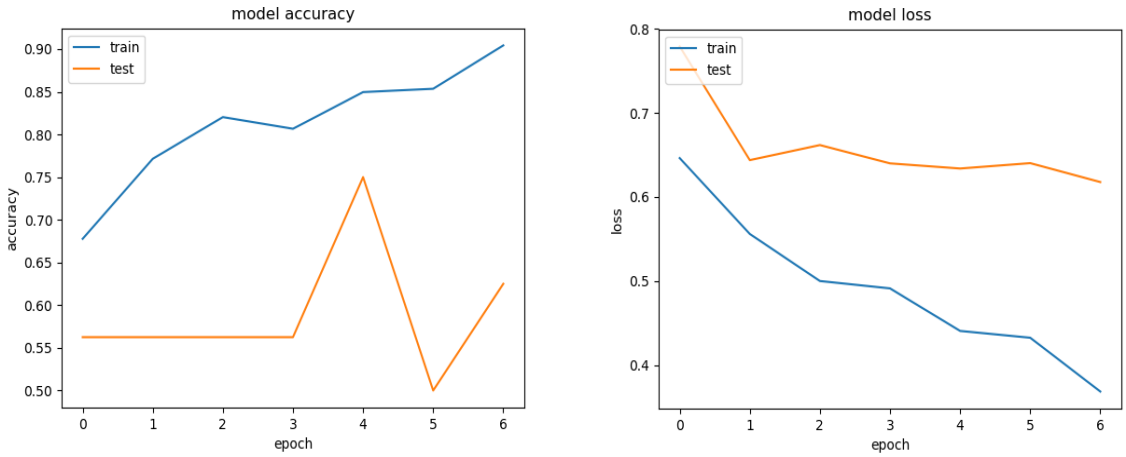


圖 20：第二次訓練準確率及誤差

```
test_info = model.evaluate_generator(test_data, steps=250)
print('test loss:', test_info[0], '  test accuracy:', test_info[1])
```

圖 21：第二次訓練後測試用主要程式碼

伍、研究結果

我們使用了正確標籤與神經網路輸出結果進行比對，先將肺部 X 光圖片預處理後再輸入神經網路，以得出患有肺炎之機率及熱度圖，並在熱度圖下方加上正確標籤，以便觀察輸出結果與正確標籤，最後再將熱度圖顯示於螢幕，最後結果如圖 22。

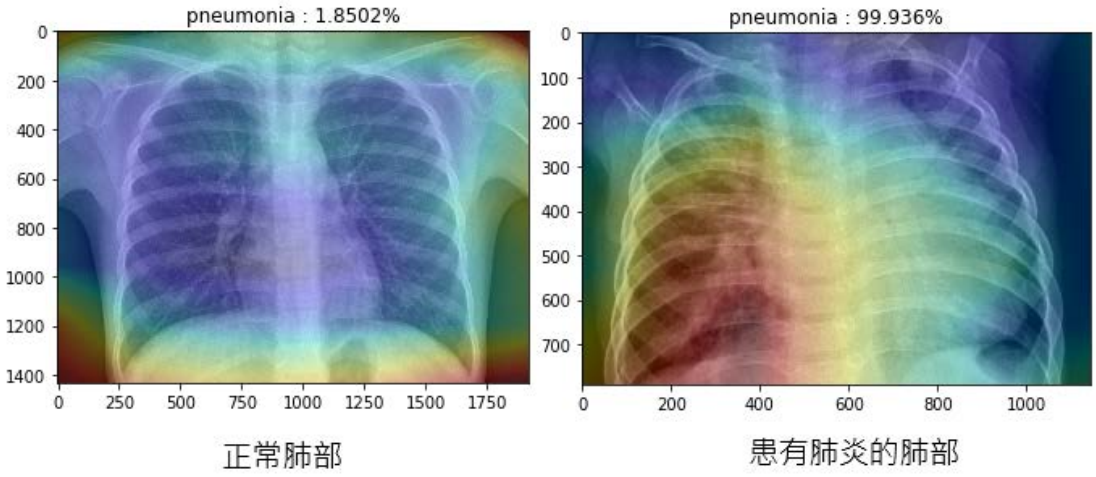


圖 22：熱度圖與正確標籤之比對
（圖片上方為神經網路判斷出的肺炎機率，圖片下方為正確標籤）

再將患有肺炎的 X 光圖輸入神經網路，並將生成的熱度圖與醫師診斷之結果比對，如圖 23。

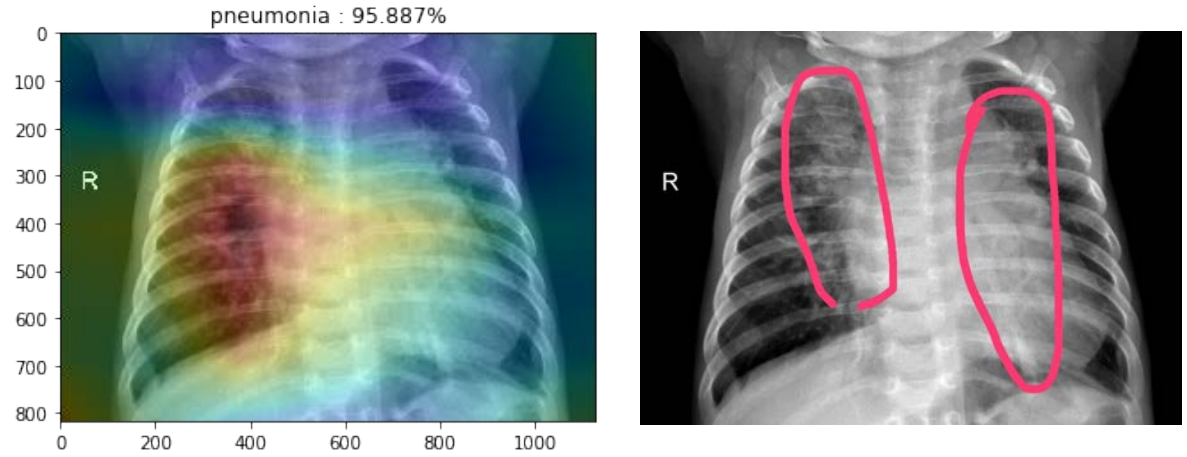


圖 23：熱度圖激活位置與醫師診斷之比對

但有時熱度圖仍會標示有誤，雖然輸出機率正確，但判斷依據卻是在肝臟，而非肺部位置，如圖 24 所示，可能因 X 光圖片過於混濁或有些許雜訊，例如鋼釘或心律調整器等，皆會使熱度圖激活位置錯誤。

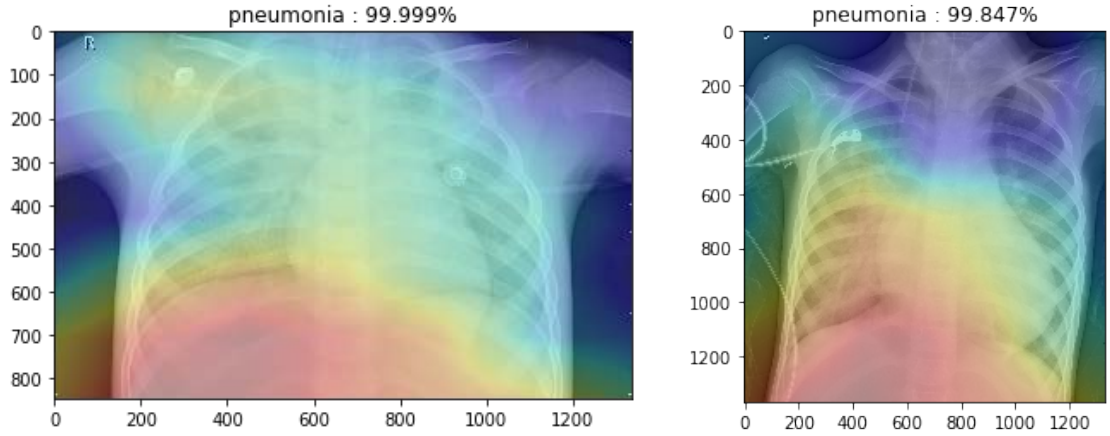


圖 24：熱度圖激活位置為肝臟

陸、討論

一、研究特色

我們使用機器學習的方式進行肺炎的判定，藉由機器協助醫師的診斷，並以熱度圖來表示可能患有肺炎（Pneumonia）的部位，以減少醫師出現誤診以及漏診的狀況。目前可判斷病毒型（Viral）及細菌型（Bacterial）肺炎，此神經模型架構也可用於 CT 電腦斷層掃描 及 NMRI 核磁共振成像 等醫療檢查成像的診斷，準確率及速度可以比醫師的診斷更佳，增加醫療的效率。

二、未來研究

(一)、將此程式移植進手機，方便醫師使用。

(二)、透過自編碼器（AutoEncoder）來進行圖片特徵篩選，以達到更高的準確率。

(三)、增加更多疾病的判定及預測疾病未來可能的變化，協助醫師對症下藥。

(四)、將模型藉由 TFLite 進行固化及壓縮，以增加運行速度及減少 GPU 啟動延遲。

(五)、將此作法應用於 CT 電腦斷層掃描 及 NMRI 核磁共振成像 等醫療檢查成像的診斷。

柒、結論

本次科展使用人工智慧來診斷疾病，相較於醫師診斷，此研究不僅能減少誤診率，也能大大提高醫療的效率。特別是醫療資源不足、偏僻鄉村地區，此研究可以使醫師擁有更好的決策診斷能力，也可以避免醫師在工作時過度疲勞而影響判斷能力。

對於胸部 X 光醫學成像，人工智慧可以迅速分類並標出可疑的患部位置。台灣 AI 教父杜奕瑾曾指出台灣擁有全世界最高品質的健保資料，未來若能跟健保大數據建立 AI 醫療診斷體系，則神經網路能夠判斷的範圍將不再僅限於肺炎。

目前可準確判斷病毒型及細菌型的肺炎，並標示出患部位置，若能有更高階的運算資源以及各類疾病更完整的特徵集，神經網路模型訓練效果將不止於當前研究，勢必能將各類疾病一一找出。

捌、參考資料及其他

一、引用資料：

圖 1：TensorFlow 結構 來源：Nix Wang(2018)。TensorFlow 学习笔记（二）—— 初步使用。2018 年 11 月 6 號，取自 https://nixwang.com/2018/03/12/learning-tensorflow-2/

圖 4：全連接層 來源：Matthijs Hollemans（2017）。Matrix Multiplication with Metal Performance Shaders。2018 年 11 月 7 日，取自 https://pse.is/DZTTU

圖 6：二維捲積運作原理 來源：Lukas Mosser（2017）。Stochastic Reconstruction of an Oolitic Limestone by Generative Adversarial Networks。2018 年 11 月 7 日，取自 https://pse.is/FCSZE

圖 8：原捲積方式 來源：Tommy Huang Mar （2018）。捲積神經網路(Convolutional neural network, CNN) — CNN 運算流程。2018 年 11 月 7 日，取自 https://reurl.cc/r5xj4

圖 9： Depthwise Separable Convolution 來源：汪思颖（2017）。变形卷积核、可分离卷积？卷积神经网络中十大拍案叫绝的操作。2018 年 11 月 7 日，取自 https://pse.is/FNYHF

圖 14：過擬合情況 來源：Alice（2016）。Regularization 正规化。2018 年 11 月 13 日，取自 https://morvanzhou.github.io/tutorials/machine-learning/theano/3-5-regularization/

表 1：Intel Core i5-4460 詳細資訊 來源：Intel 產品規格。2018 年 11 月 13 日，取自 https://ark.intel.com/zh-tw/products/80817/Intel-Core-i5-4460-Processor-6M-Cache-up-to-3-40-GHz-

表 2：Nvidia GeForce GTX 960 詳細資訊 來源：Nvidia GeForce 產品規格。2018 年 11 月 13 日，取自 https://www.geforce.com.tw/hardware/desktop-gpus/geforce-gtx-960/specifications

肺部 X 光圖片 來源：Daniel Kermany（2018）。Chest X-Ray Labeled Optical Coherence Tomography (OCT) and Chest X-Ray Images for Classification。2018 年 10 月 31 日，取自 https://data.mendeley.com/datasets/rscbjbr9sj/2 （CC 許可證：https://creativecommons.org/licenses/by/4.0/deed.en）

二、參考論文及書籍：

Aurélien Géron.（2017）.*Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems 1st Edition*. Sebastopol:O'Reilly Media.

Marvin Minsky & Seymour Papert .（1969）. *Perceptrons: An Introduction to Computational Geometry.* MA:MIT press.

Andrew G. Howard. （2017）. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications.

Lukasz Kaiser. （2017）. Depthwise Separable Convolutions for Neural Machine Translation.

Yann LeCun. （1998）. Gradient-based learning applied to document recognition.

Min Lin. （2013）. Network In Network

【評語】052506

此作品使用機器學習的方式進行肺炎診斷，共分成兩個階段，首先將肺部 X 光的圖片預處理，接著再將預處理後的 X 光圖輸入神經網路，以得到是否罹患肺炎的判斷結果。此作品使用類神經網路方法來做模型訓練和影像辨識的應用，對類神經網路方法的技術沒有改進而只是使用，雖然有探討過度擬合的問題且使用一些簡單的方法來減輕此問題，但這些方法都是已知且常見的方法，創新性不足。

建議未來加強能提升辨識率的創新研究，讓這個作品更具有實用性。

作品海報

壹、研究動機

在這個時代，醫療技術日新月異，但肺炎仍位居國人十大死因中的第三名。因此我們萌發出了一個想法，為何不使用資訊技術來協助醫療呢？因此我們做了一個神經網路協助醫師診斷，判斷病患是否罹患肺炎，可減少看診時間，使醫療效率最大化，並減少誤診及漏診等狀況的發生機率，增進人民福祉。

貳、研究目的

所以我們需要一種快速、精準、方便檢測的工具，在現在的社會，可以說是人工智慧世代，而AI的應用性即可成為一個隨時可診斷的萬能醫師助理，只要將圖片輸入進電腦，就能透過以訓練好的神經網路模型來檢測是否罹患肺炎。

參、研究過程與方法

一、梯度下降

在機器學習中，梯度下降法是為了尋找一個最佳解，而最佳解就是模型輸出結果與實際答案相同，也就是與實際答案沒有誤差。透過調整參數來減少誤差，並以計算誤差後當前狀況的斜率來決定是否調整變數。圖1是使用均方誤差 (MSE) 計算後的立體圖，紅色線條為變數調整紀錄。

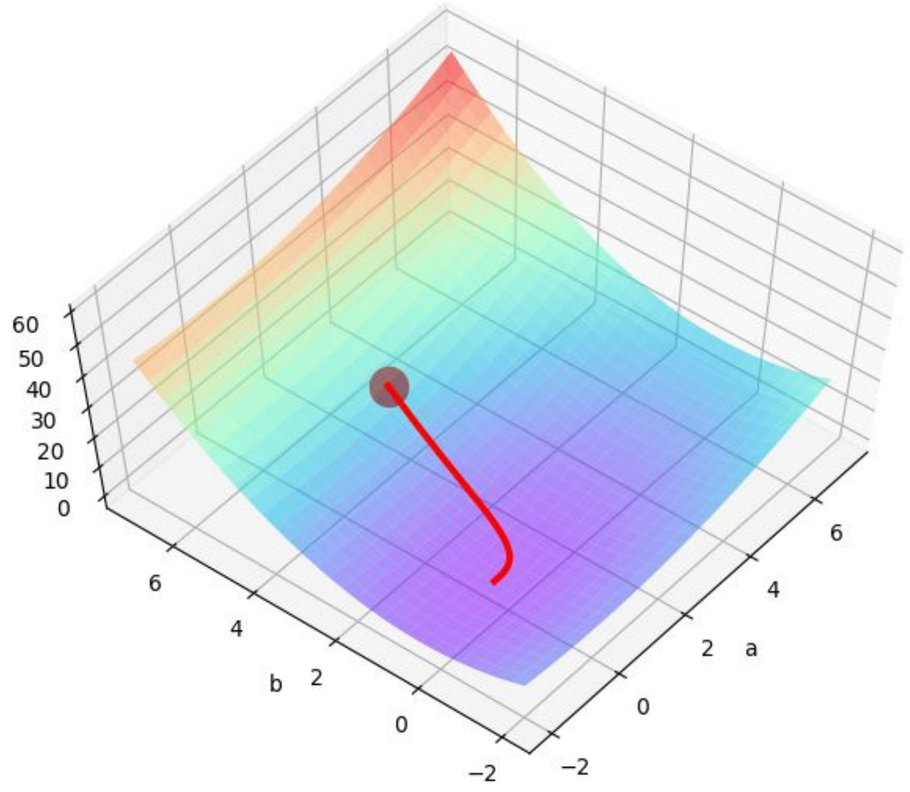


圖1：梯度下降演示立體圖

二、全連接層

全連接層（Fully-connected Layer）最常用於機器學習的輸出層，有分類器的作用，可運算特徵的加權和。由神經元組成，每個神經元都有個W(weight 權重) 和 b(bias 偏差值)，兩者皆為變數。大多都會搭配激勵函式使用。

$$ReLU\left(\sum_{i=1}^{inputsize} W_i x_i + b\right)$$

圖2：全連接層結合ReLU運算方程

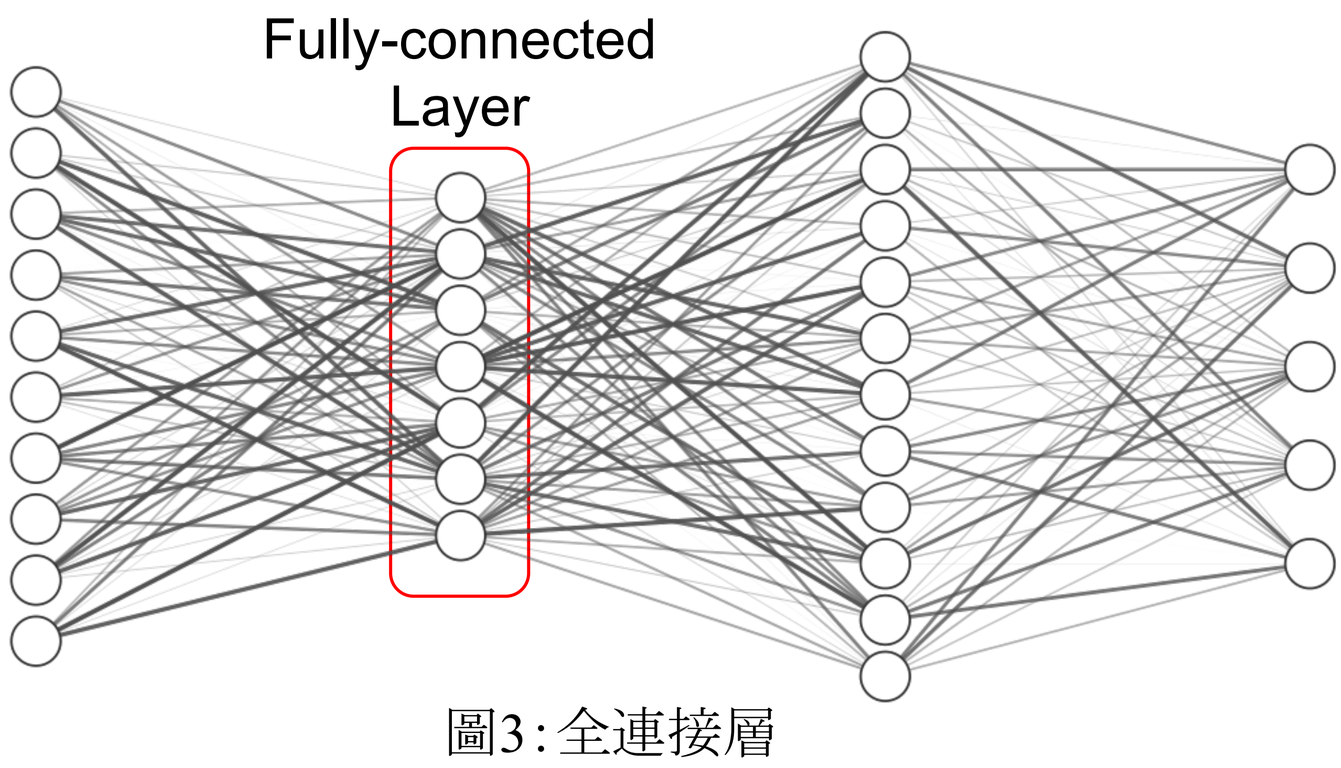


圖3：全連接層

三、二維捲積層

捲積層（Convolutional layer）由許多的濾波器組成。捲積層對於圖像辨識是非常重要的存在，可以經由訓練學習提取出一些重要特徵，如顏色、邊緣、細節等，如圖4所示。

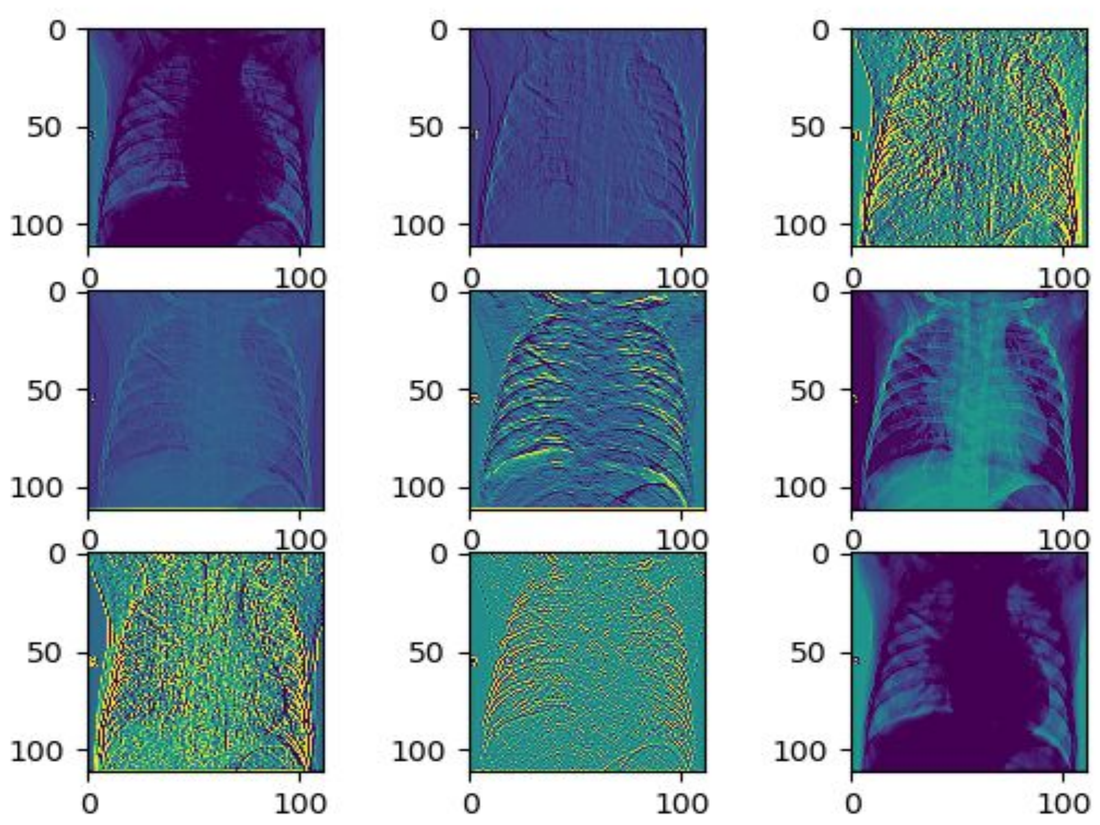


圖4:捲積後圖片特徵

此次科展我們使用的是Depthwise Separable Convolution，這是Google在2017年所發布的新捲積計算方法。在神經網路中，大部分的計算都是在捲積層，此捲積計算方法便是在此方面進行優化。在傳統的捲積計算中，原圖會乘上所有的捲積核，而此種捲積計算差別於每張輸入圖都會有相對應的捲積核，如圖6所示，這樣的設計使運算量大幅減少，且準確度也不會因此降低。

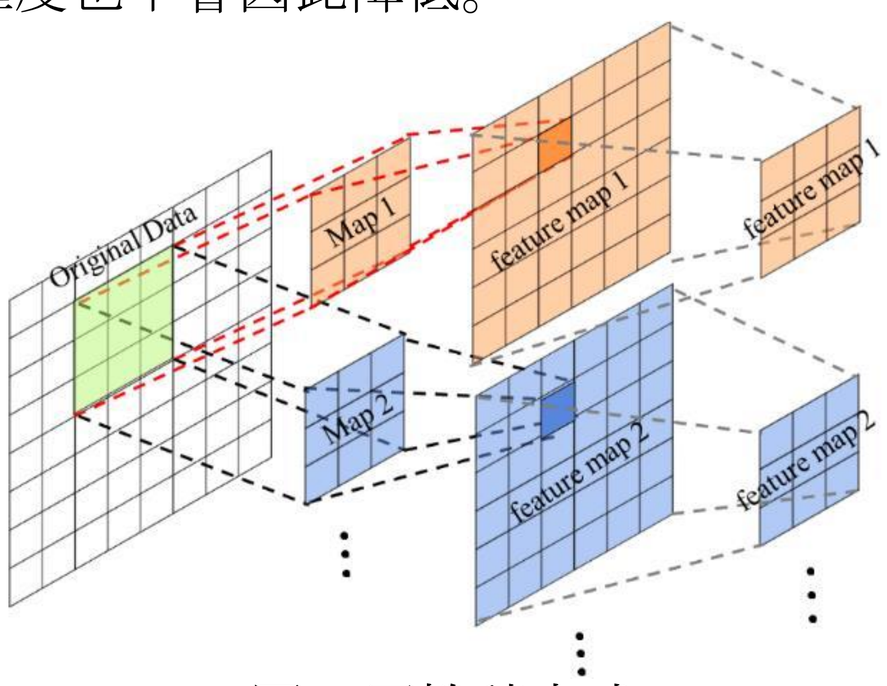


圖5：原捲積方式

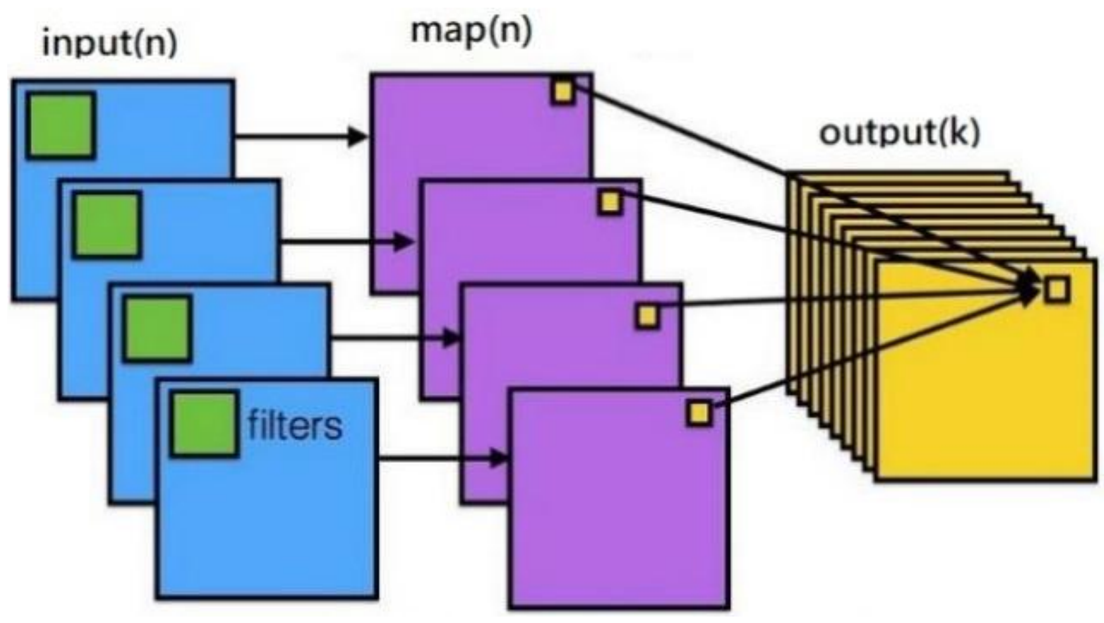


圖6：Depthwise Separable Convolution

四、全局平均池化

全局平均池化(Global Average Pooling), 簡稱為GAP, 主要是將捲積過後的各張特徵圖裡的像素值做平均的動作, 使網路結構做正則化以防止過擬合的問題, 如圖7。可取代全連接層, 以避免全連接層參數過多造成過擬合及訓練時間過長的問題。

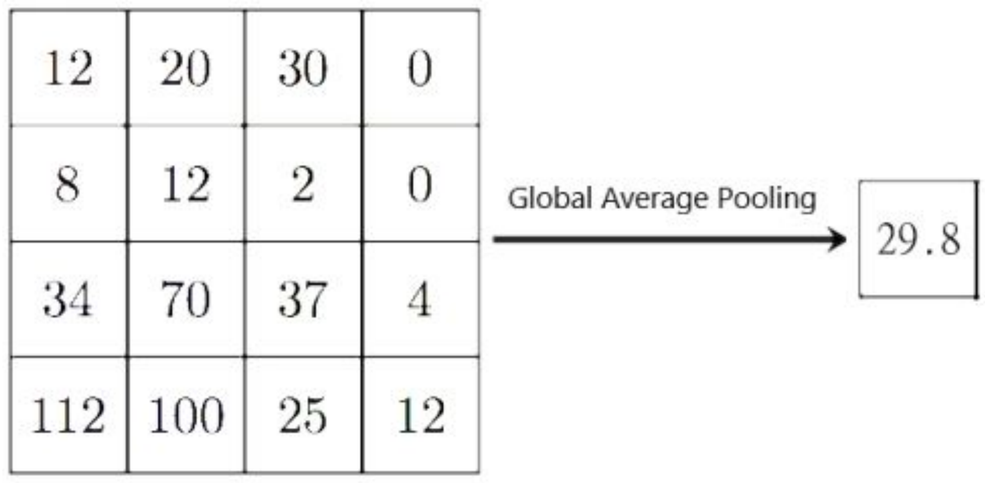


圖7:全局平均池化

五、過擬合

過擬合是一種訓練神經網路時產生過度適應訓練資料的情形, 如圖8, 產生的原因有:

(一)、訓練的數據不足

(二)、訓練的數據有雜訊

(三)、從訓練的數據擷取過多特徵

此次實驗使用以下方法避免過擬合發生:

(一)、正規化

(二)、丟棄法

(三)、減少訓練批數

(四)、交叉驗證

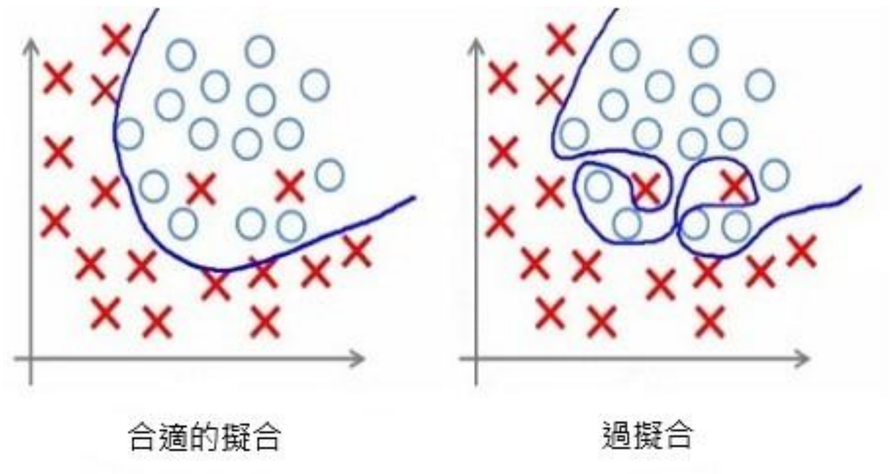


圖8:過擬合

六、Class Activation Mapping

Class Activation Mapping (CAM), 是一種使捲積神經網路輸出可視化的方法, 可產生熱度圖。要使用CAM首先要將輸出層中的權重取出, 並與最後捲積層輸出之特徵圖相乘, 以產生熱度圖, 流程圖如圖9。

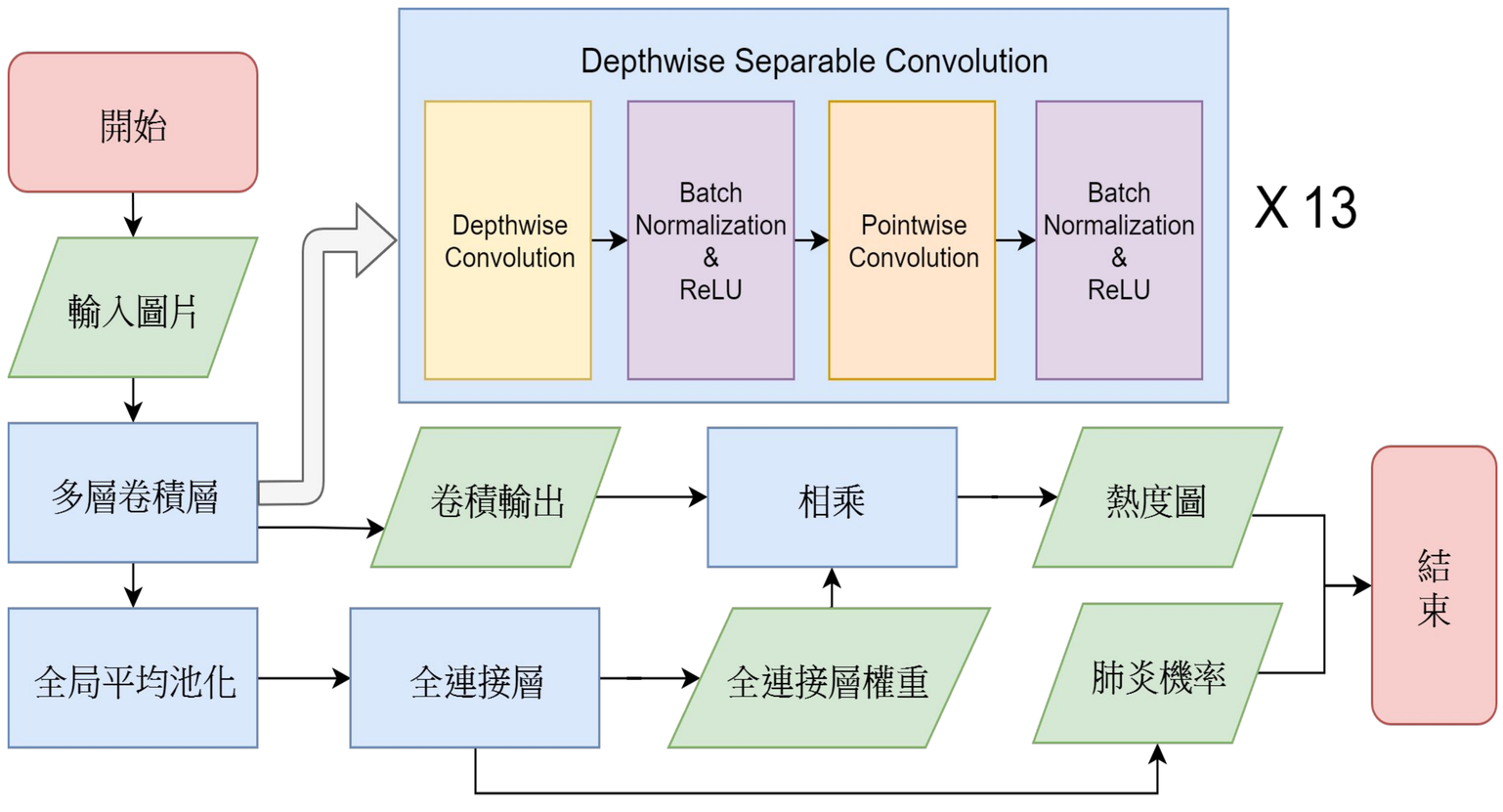


圖9:流程圖

七、硬體

Intel Core i5-4460

效能	核心數量	4
	執行緒數量	4
	處理器基本頻率	3.2G
	快取記憶體	6MB SmartCache
記憶體規格	記憶體類型	DDR3-1333/1600 DDR3L-1333/1600

Nvidia GeForce GTX 960

引擎規格	CUDA核心	1024
	基本時脈	1127M Hz
記憶體規格	記憶體時脈	7Gbps
	標準記憶體配置	2GB
	記憶體介面	GDDR5
	記憶體介面寬度	128-位元

八、訓練過程

第一次訓練使用了15360張圖片進行訓練, 訓練了30批, 每批512(16*32)張, 結果如圖10所示, 大約在第10批訓練左右開始有了過擬合的情況。

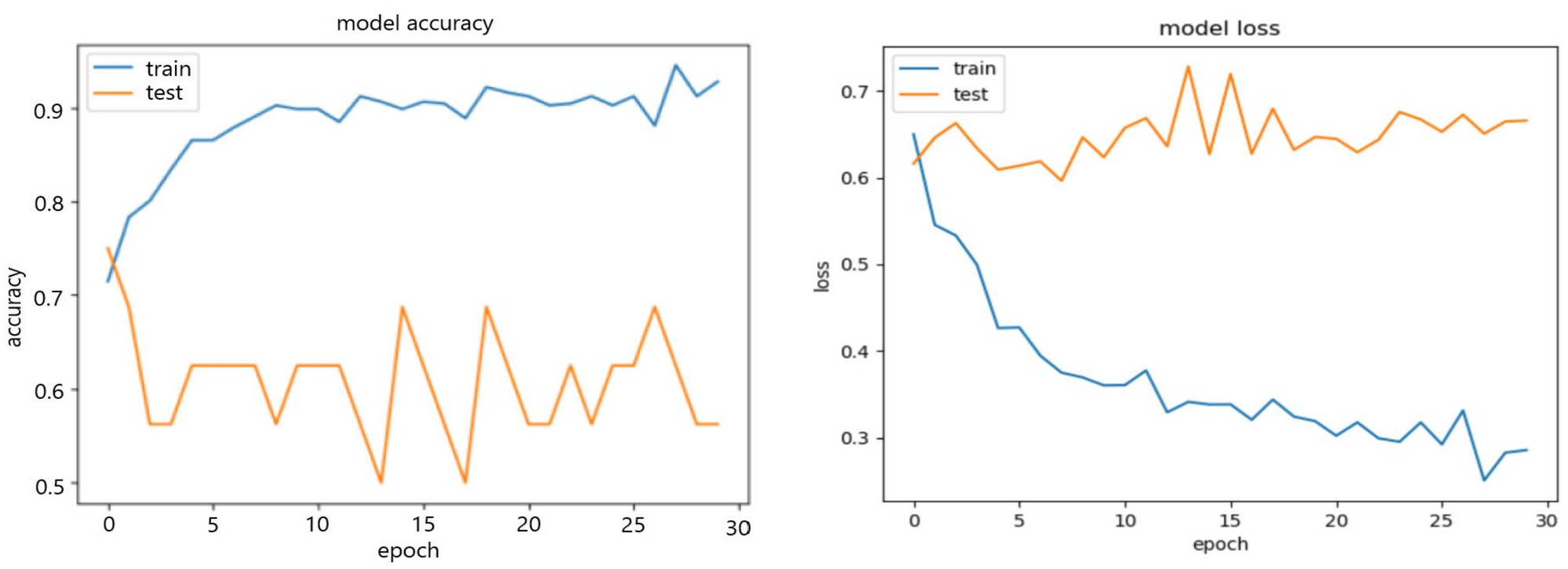


圖10:第一次訓練之準確率及誤差

第二次訓練時使用了3584張圖片，只訓練了7批，每批也是512張，結果如圖11。使用4000張僅用於測試的圖片進行測試後，得到了91.58%的準確率。

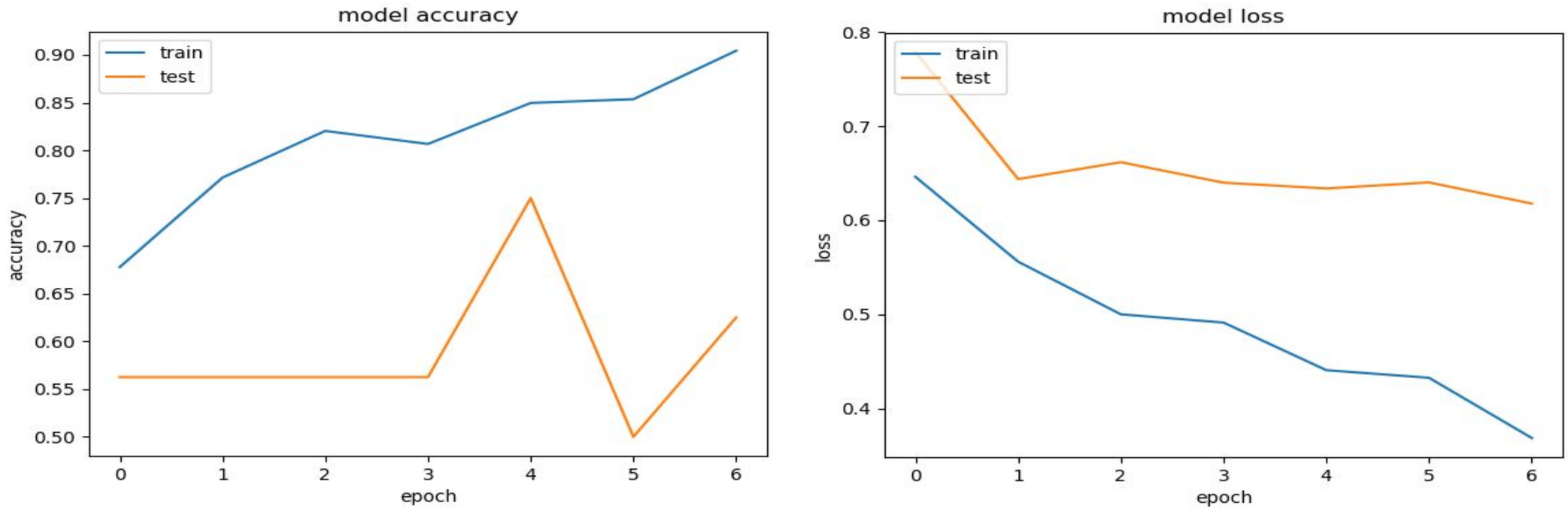
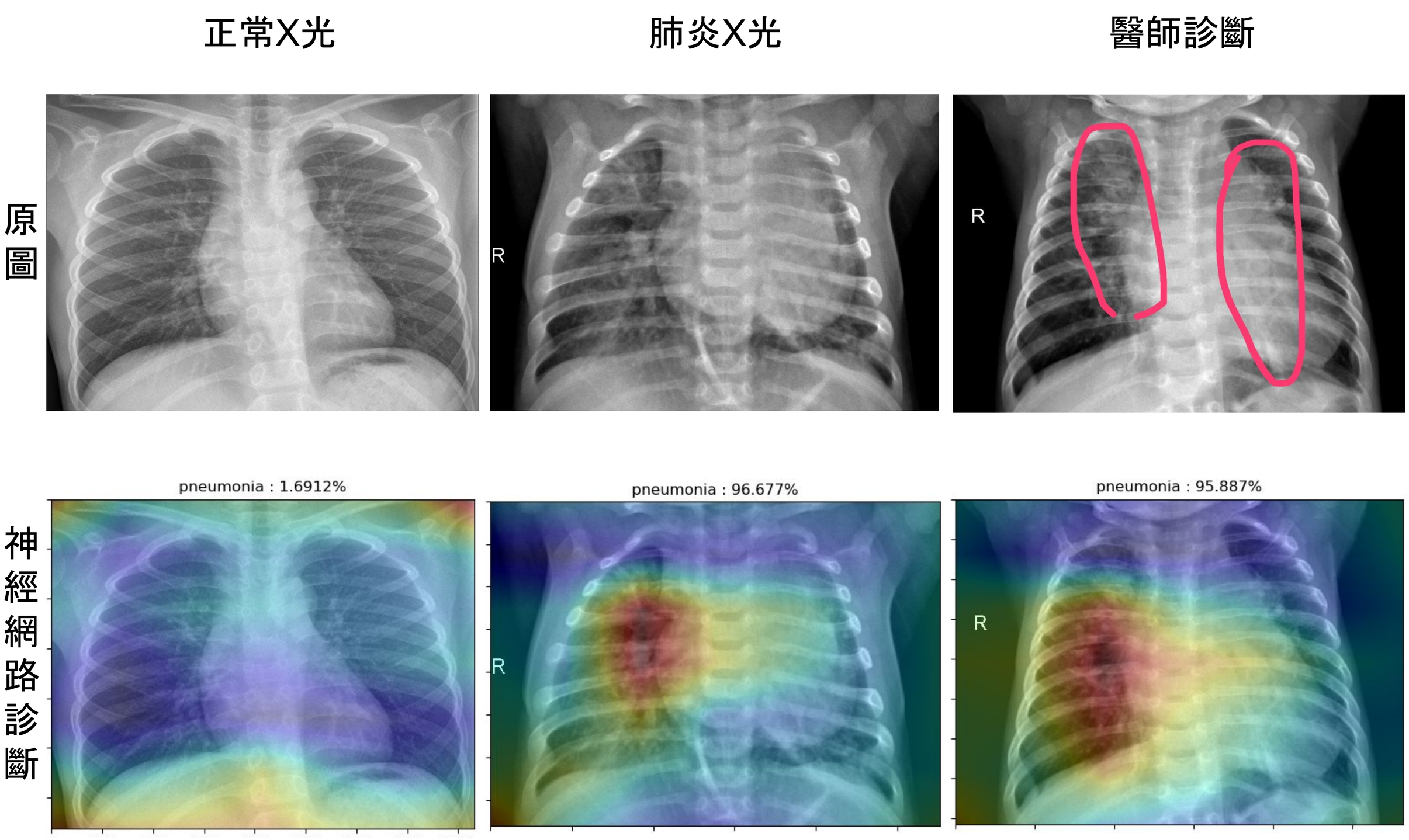


圖11:第二次訓練之準確率及誤差

肆、研究結果



伍、討論

研究特色

(一)、AI技術產出熱度圖:標示可能患有肺炎的部位。

(二)、使用輕量化的模型以減少運算量及增加運行速度，且設備需求低。

(三)、醫師協助判斷圖片正確性，驗證系統的可靠性。

(四)、可判斷病毒型(Viral)及細菌型(Bacterial)肺炎。

未來研究

(一)、藉由TFLite進行固化及壓縮，以增加運行速度及減少GPU啟動延遲。

(二)、透過自編碼器進行圖片特徵篩選，以達到更高的準確率。

(三)、增加能判斷的疾病種類，協助醫師對症下藥。

(四)、繼續研發手機APP，方便醫師使用。

(五)、應用於CT電腦斷層掃描及 NMRI核磁共振成像等醫療檢查。

陸、結論

本次科展使用人工智慧來診斷疾病，相較於醫師，此研究不僅能減少誤診率，也能大大提高醫療的效率。此研究可以使醫療資源不足、偏僻鄉村地區的醫師擁有更好的決策診斷能力，也可以避免醫師在工作時過度疲勞而影響判斷能力。

目前可準確判斷病毒型及細菌型的肺炎，並標示出顯在的患部位置。因設備需求不足，故使用輕量型神經網路作為模型，若能有更高階的運算資源以及各類疾病更完整的特徵集，神經網路模型訓練效果將不止於當前研究，勢必能將各類疾病一一找出。