Table of Contents

[一、簡介 2](#_Toc93078373)

[二、動機 3](#_Toc93078374)

[三、系統規格 4](#_Toc93078375)

[一、神經網路之設計 4](#_Toc93078376)

[二、神經網路之實作 13](#_Toc93078377)

[一、Convolution Layer的Schematic Design 15](#_Toc93078378)

[二、Convolution 2D的Schematic Design 16](#_Toc93078379)

[三、Convolution Kernel的Schematic Design 19](#_Toc93078380)

[四、Max Pooling的Schematic Design 21](#_Toc93078381)

[五、Dense Layer的Schematic Design 22](#_Toc93078382)

[六、Full Connect的Schematic Design 23](#_Toc93078383)

[三、浮點數運算器之實作 25](#_Toc93078384)

[一、加法器 26](#_Toc93078385)

[二、乘法器 26](#_Toc93078386)

[三、整數轉浮點 26](#_Toc93078387)

[四、浮點轉整數 26](#_Toc93078388)

[四、實驗結果 27](#_Toc93078389)

[五、討論 28](#_Toc93078390)

[一、LUT過小 28](#_Toc93078391)

[二、BRAM過小 28](#_Toc93078392)

[三、臨時更改架構 28](#_Toc93078393)

[六、結論 29](#_Toc93078394)

# 一、簡介

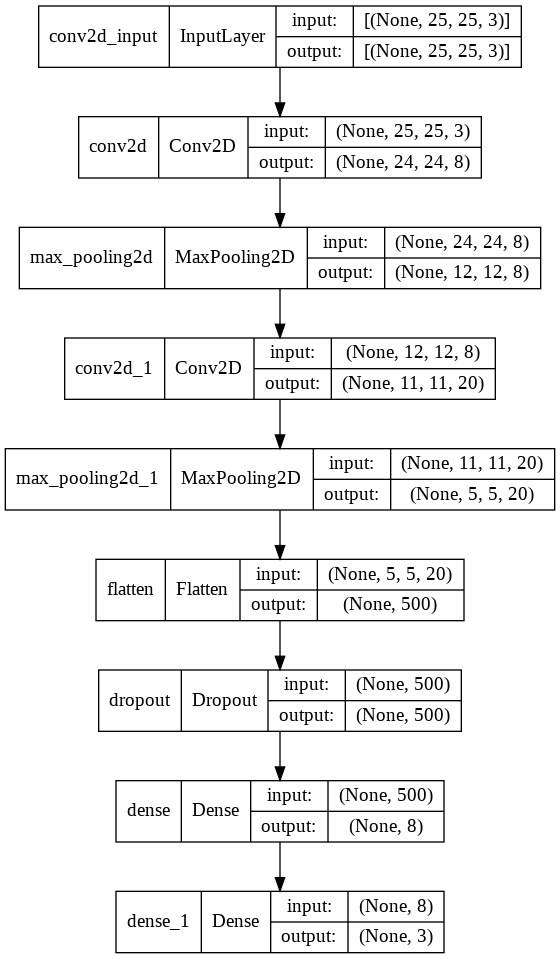
本期末專題為基於CNN的猜拳機，

# 二、動機

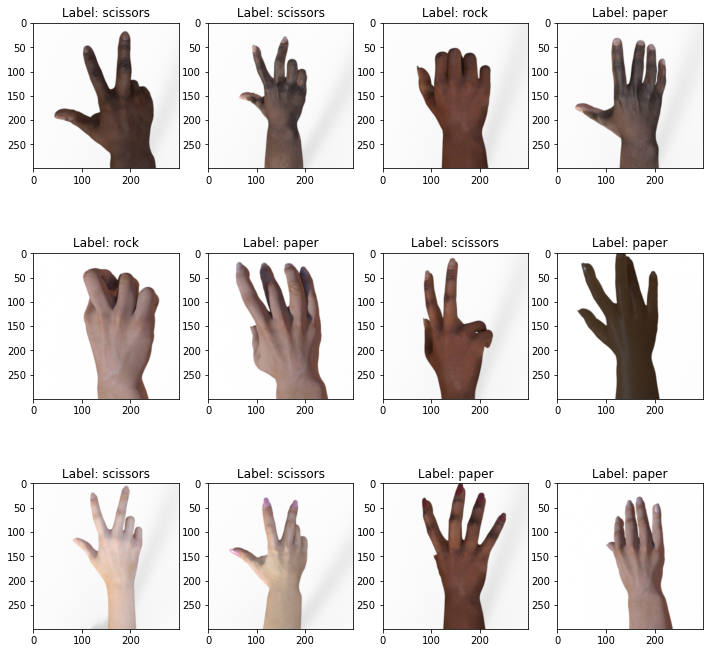
# 三、系統規格

## 一、神經網路之設計

下圖為神經網路之結構圖，由圖可見該網路使用了雙層的卷積、池化層，並於神經網路的末端加入全連接層。



下圖為原始訓練資料，由於FPGA運算資源相當有限，因此必須先降低解析度，方可將神經網路燒入FPGA中。下圖為訓練用的原始圖片，可以發現原始圖片為300x300的手勢資料。



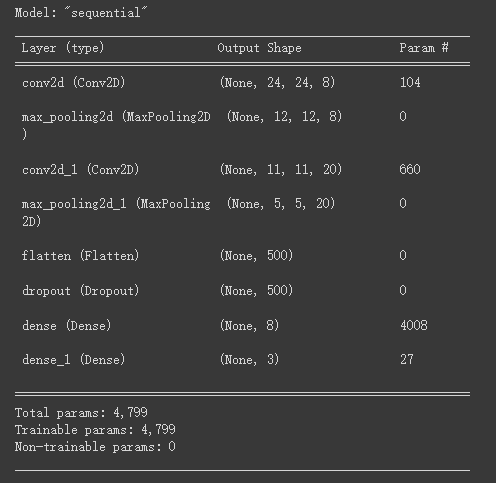
下圖為訓練用的壓縮後圖片，可以見到壓縮後只有25x25的解析度。



為了增加模型的穩健度，我們對訓練資料進行資料擴增 (Data Augmentation)，下圖為擴增後的訓練資料。



由於FPGA的記憶體容量相當有限，本神經網路只使用約略5000個參數，每個參數都是一個32位元浮點數。



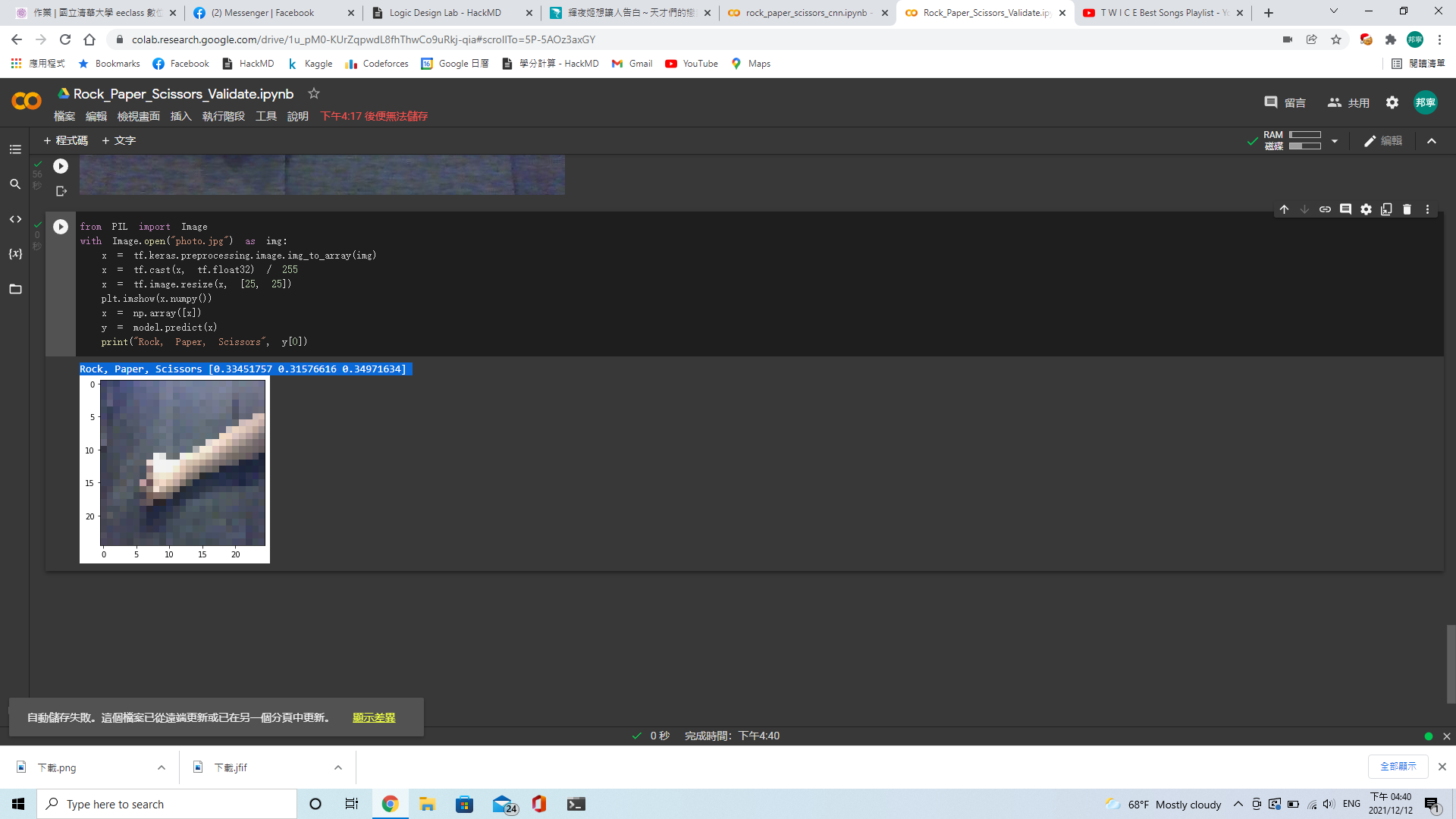
模型採用Adam進行最佳化，學習率採0.001，並訓練30個Epoch，下圖為訓練過程。由圖可見，訓練資料集與測試資料集無明顯差距，由此可以排除模型過擬合(Overfitting)之可能。

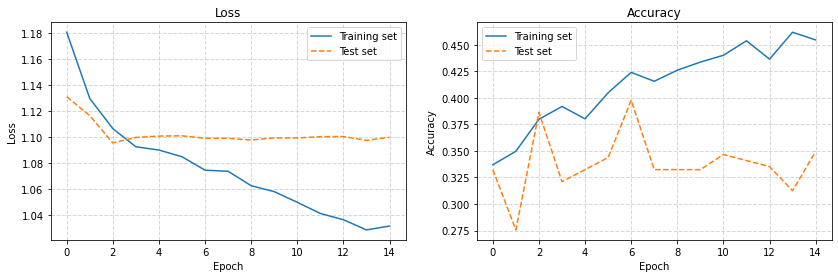


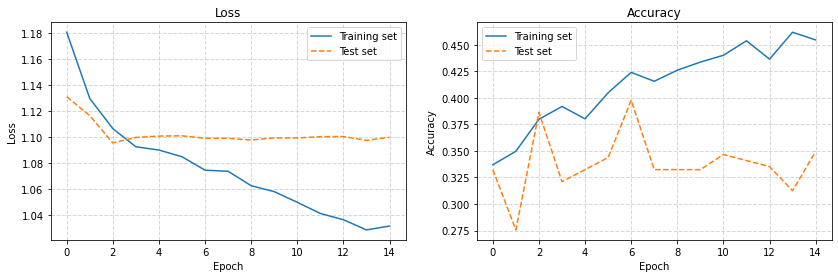


在現實世界測試(Real Life Testing)時，剪刀與布都能被準確的辨識，唯獨石頭無法被精確辨識；不僅如此，測試結果也顯示模型對光照角度、光線強度相當敏感。因此，於實際應用時，應維持穩定光源。下圖展示石頭無法被準確辨識。



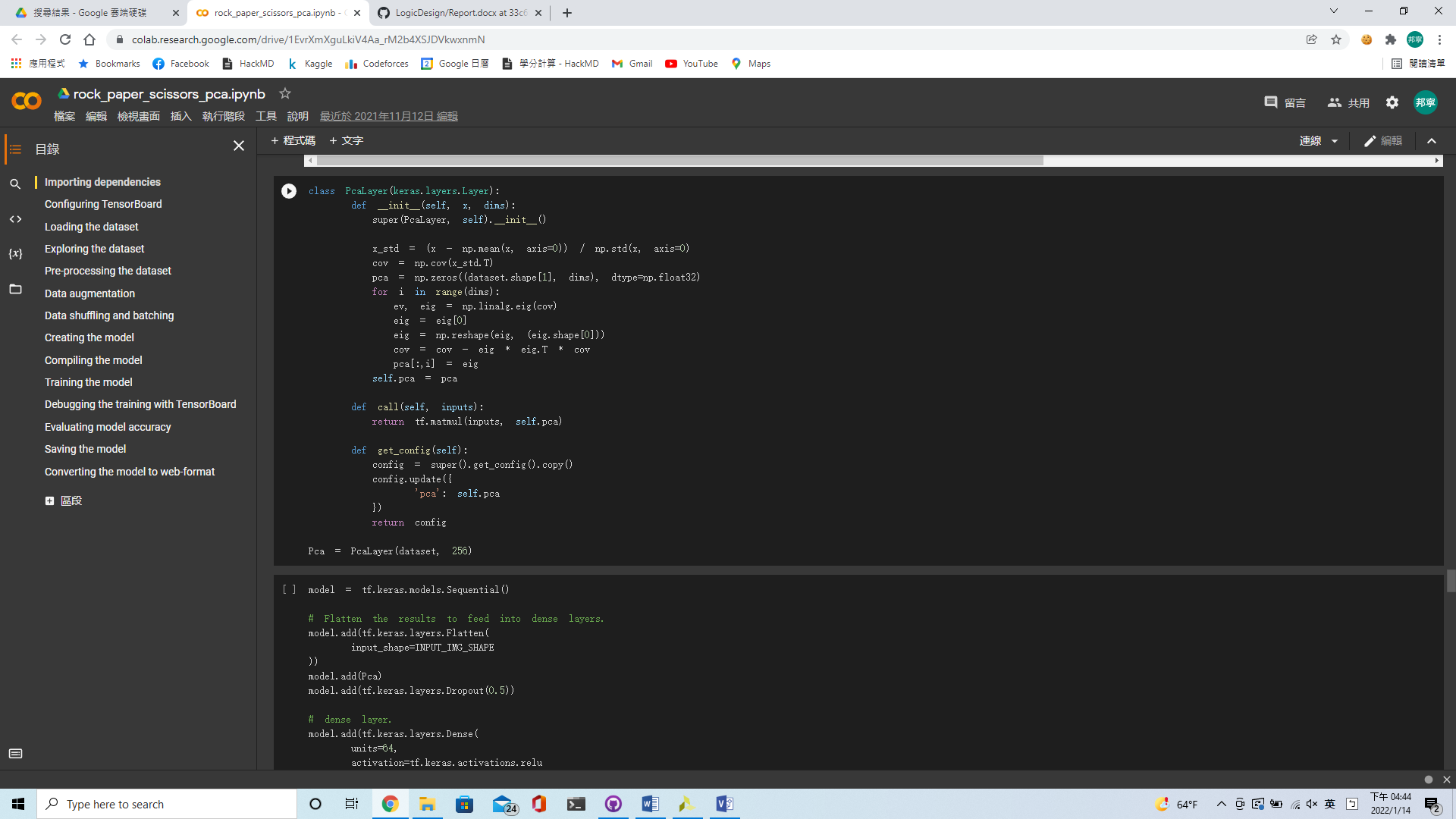


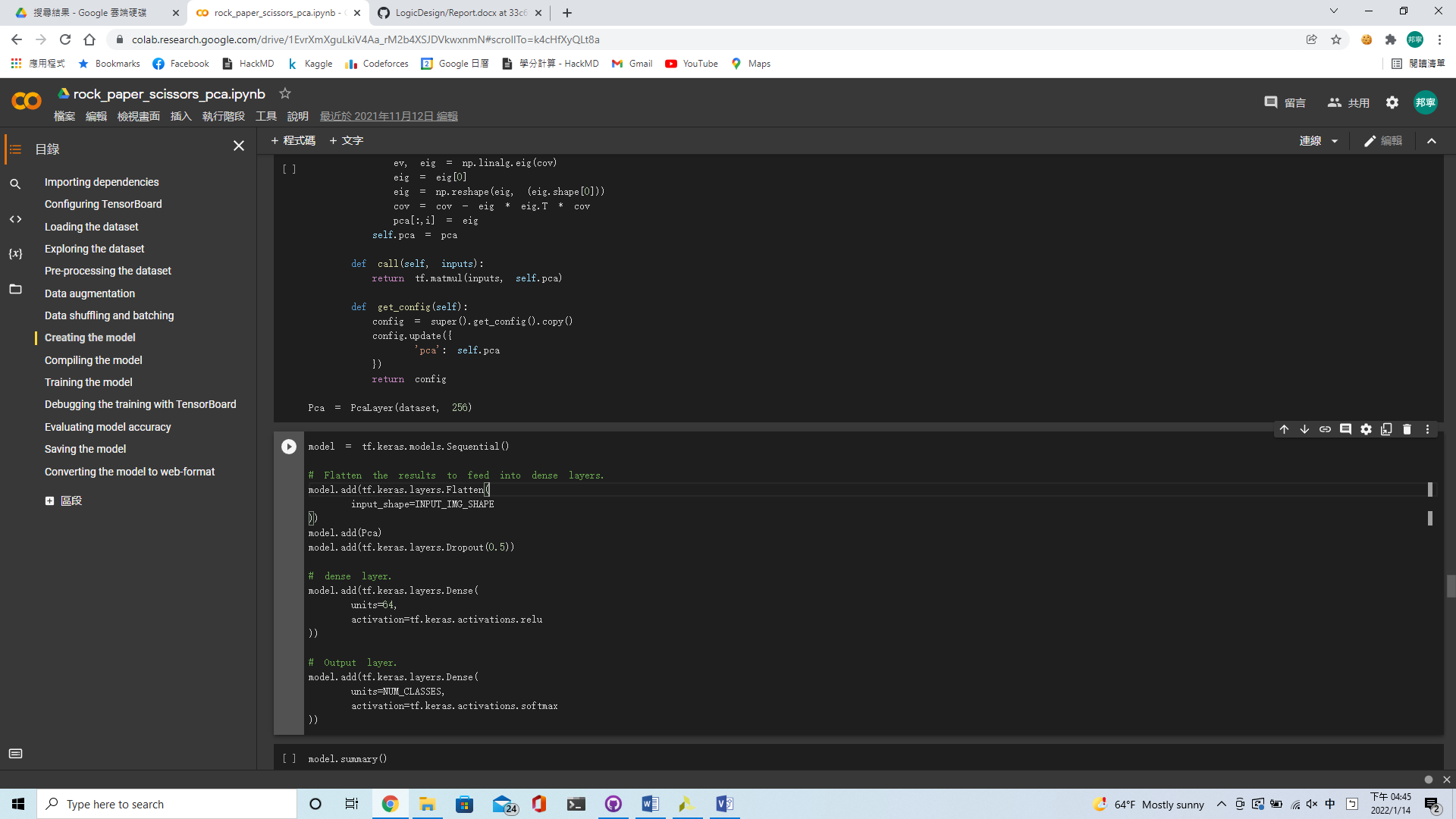
為降低神經網路的參數使用量，我們曾採用PCA (Primary Content Analysis, 主成分分析)去提起圖片特徵，不幸的是，計算PCA需要過高的時間複雜度，且小規模測試後的實驗結果不佳，因此，最後並無採用該計算方法。



可由上述圖表得知，PCA化簡後的特徵不足以代表整個資料集，進而導致過度擬和。

不幸的是，Keras並未內建PCA，而為了計算PCA，必須自行以Keras套件實作一層PCA。PCA的計算過程並不複雜，可轉化為計算特徵值、特徵向量的問題，下圖為PCA的實作方法。





## 二、神經網路之實作

大方向來說，我們有以下這張樹狀結構圖。

至於各式神經網路相關的模組，可以用下圖粗略表示。

### 一、Convolution Layer的Schematic Design



### 二、Convolution 2D的Schematic Design











### 三、Convolution Kernel的Schematic Design







### 四、Max Pooling的Schematic Design



下圖為Relu的Schematic Design。



### 五、Dense Layer的Schematic Design



### 六、Full Connect的Schematic Design









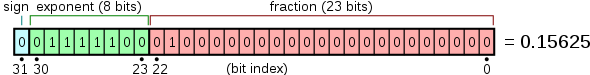


## 三、浮點數運算器之實作

本浮點數運算器為Pure Combinatorial Circuit，沒有任何一個Flip Flop，也沒有任何一個Latch，完全為了效率而生，也是完全不在乎晶片面積的設計方式。並且，本浮點數運算器採IEEE-754規範設計而成，下圖為浮點數的表達方式。



下圖為範例。



上圖中，Exponent部分為，而Fraction部分為，而，由此規律，我們可以找出浮點數轉整數的方法。

由於乘法與加法在浮點數上可粗略的構成一個阿貝爾群，且神經網路僅需用到乘法與加法，無須實作除法，亦無需實作其他運算單元。

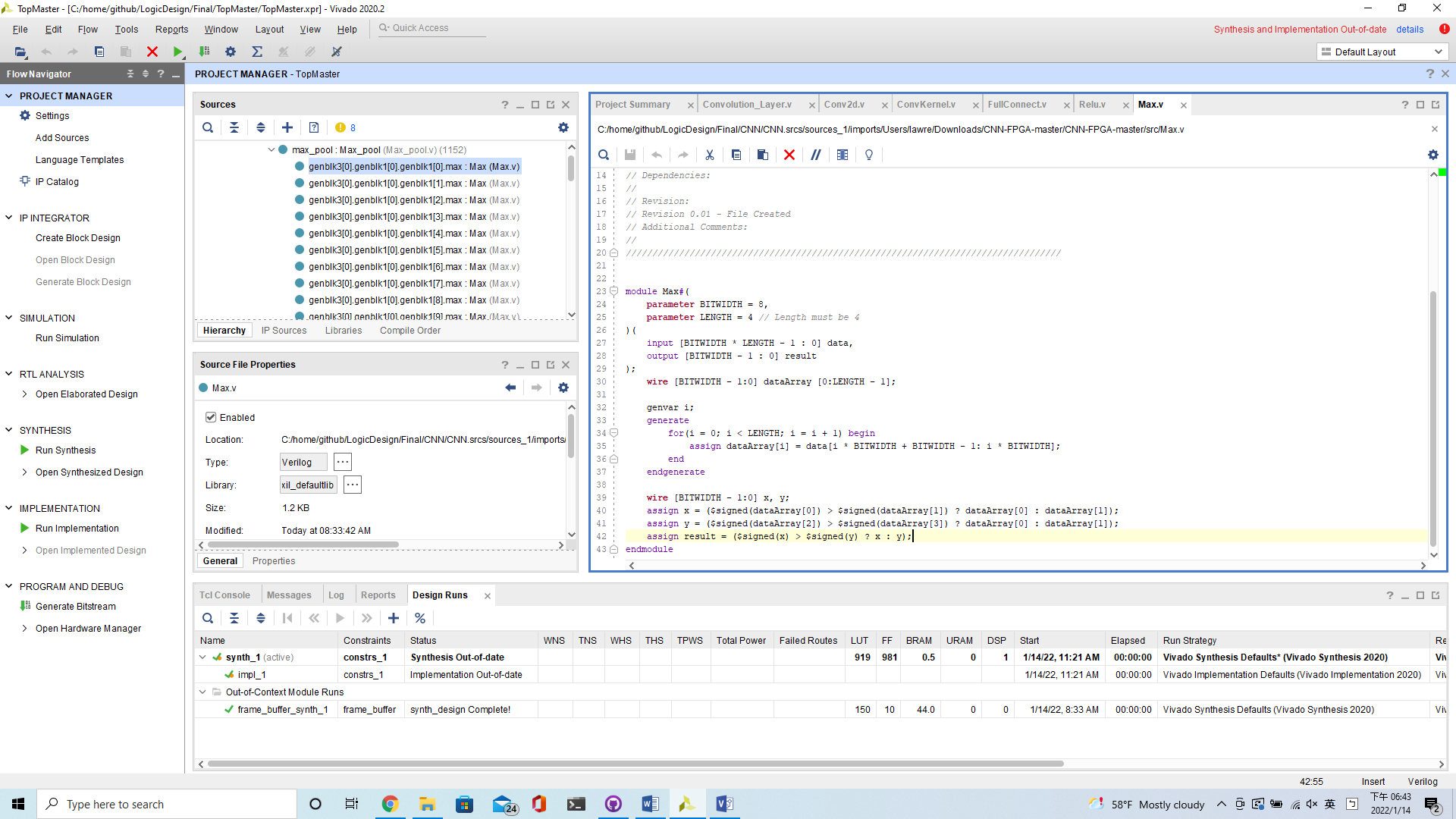
有鑑於IEEE-754的特性，執行浮點數比大小時，可直接利用帶正負號整數之比較器，這是因為浮點數的比較順序為

一、正負號

二、Exponent，也就是指數部分

三、Fraction, Mantissa，也就是小數部分

因此，無須額外設計電路，僅需要將浮點數視為帶正負號整數，即可進行比較。



### 一、加法器













### 二、乘法器











# 四、實驗結果

### 一、光線對神經網路的敏感性

根據實驗，我們知道光線能夠大幅影響神經網路的輸出，如果鏡頭背光，那麼神經網路會十分不準；如果環境光源是帶有顏色的光線，那也會對神經網路造成影響；如果是多點光源，也會對神經網路造成影響。

造成上述現象的原因有下面三點。

一、資料集的光源都是單點光源

二、資料集的光源都是無色光源

三、鏡頭的輸入與模型的訓練資料有差異

### 二、背景對神經網路的敏感性

根據實驗，我們發現白底與黑底的背景對神經網路輸出影響力不大，這是因為神經網路的的訓練資料集有經過資料擴充；但是，我們發現具有斑點或是條紋的背景，會對神經網路造成影響，這是因為資料集沒有受過這些背景的訓練。

### 三、其他因素對神經網路的敏感性

一、左手右手

由於所有的訓練資料都是右手，因此，使用左手來猜拳會造成失準；雖然左右手形狀類似，但是有拇指的差異。因此，左手的準確率較右手為遜色。

二、外套

本次期末專題位於冬季，而人們習慣在冬季穿著外套，而穿著外套時，神經網路會受到額外的干擾。我們的硬體背景是黑色，若使用者穿著黑色外套，則神經網路只能辨別非黑色的手掌，無法取得手臂資訊。因此，穿著外套會對神經網路造成影響。

三、手環

不少人有配戴手環的習慣，而配戴手環容易造成額外的干擾。神經網路在訓練時，不會預期手上會有一圈東西。因此，手環對神經網路也是額外的干擾。

四、膚色

值得注意的是，膚色其實對神經網路的影響不大。這是因為神經網路有經過資料擴充，資料擴充後，不論是什麼顏色的膚色，都有少量樣本，因此神經網路可以成功辨認。

五、指甲油

指甲油對神經網路的影響力相當大，尤其是顏色鮮豔的。這是因為神經網路不期待指尖上會有額外的色塊，進而造成模型失真。

# 五、討論

## 一、LUT過小

不幸的，由於LUT過小，我們無法大量複製浮點數運算單元。原先預計要以浮點運算單元為最小顆粒的Pipeline無法實際燒錄，如果以符點運算單元為Pipeline最小顆粒，會使得晶片面積過大，進而使得LUT被塞爆。

為了處理上述問題，我們選擇讓同一個浮點運算單元重複計算，也就是說，一個Convolution Kernel中，只有常數個浮點運算單元，並且計算時間跟輸入資料大小成線性。藉由改變電路結構，並且延長計算時間，我們能夠更有效的運用晶片面積，並使得燒錄進FPGA變成可行。

## 二、BRAM過小

由於鏡頭模組將耗費大量記憶體於儲存鏡頭畫面，VGA模組亦將耗費大量記憶體於儲存輸出螢幕畫面，BRAM成為一項珍稀資源。然而，在努力壓縮記憶體用量後，硬體資源仍然不敷使用。

為了增加硬體資源，我們決定使用兩片FPGA，藉此取得雙倍的硬體資源，並利用Chip 2 Chip的互動功能進行資訊傳遞。

## 三、合成過慢

由於神經網路十分複雜，而且浮點運算單元不是簡單的晶片，交由Vivado合成時相當緩慢，短則數十分鐘，長則數十小時。為了解決該問題，我們在Synthesis時採用Runtime Optimized選項，並於Implementation時採用Flow Quick選項。

雖然使用最快的演算法必然不會給出最優的結果，但是，本專案並非尋求最佳的解決方案，而是在有限的時間內給出可接受的解答，因此，以晶片面積換取執行時間不失為一種好辦法。

# 六、結論