Group 3

109060013張芯瑜 109062328 吳邦寧

cnn猜拳機

Logic Design Laboratory Final Project

Table of Contents

[一、簡介 3](#_Toc93094051)

[二、動機 4](#_Toc93094052)

[三、系統規格 5](#_Toc93094053)

[(1)神經網路之設計 5](#_Toc93094054)

[(2)神經網路之實作 14](#_Toc93094055)

[1. Convolution Layer的Schematic Design 16](#_Toc93094056)

[2. Convolution 2D的Schematic Design 17](#_Toc93094057)

[3. Convolution Kernel的Schematic Design 20](#_Toc93094058)

[4. Max Pooling的Schematic Design 22](#_Toc93094059)

[5. Dense Layer的Schematic Design 23](#_Toc93094060)

[6. .Full Connect的Schematic Design 24](#_Toc93094061)

[(3)浮點數運算器之實作 26](#_Toc93094062)

[1. 加法器 27](#_Toc93094063)

[2. 乘法器 30](#_Toc93094064)

[(4) 遊戲進行與螢幕顯示相關設計 32](#_Toc93094065)

[1. TOP\_SLAVE 32](#_Toc93094066)

[2. STATE\_CHANGE 33](#_Toc93094067)

[3. RANDOM 34](#_Toc93094068)

[4. SLAVE 35](#_Toc93094069)

[5. VGA 37](#_Toc93094070)

[四、實驗結果 40](#_Toc93094071)

[(1) 光線對神經網路的敏感性 40](#_Toc93094072)

[(2) 背景對神經網路的敏感性 40](#_Toc93094073)

[(3) 其他因素對神經網路的敏感性 40](#_Toc93094074)

[五、討論 41](#_Toc93094075)

[(1)LUT過小 41](#_Toc93094076)

[(2)BRAM過小 41](#_Toc93094077)

[(3)合成過慢 41](#_Toc93094078)

[六、結論 42](#_Toc93094079)

# 一、簡介

本計畫將以現場可程式化邏輯閘陣列 (Field Programmable Gate Array, FPGA) 實作電腦視覺 (Computer Vision, CV)，並以該技術實作手勢 偵測式 (Gesture Detection) 的剪刀石頭布遊戲。

首先，FPGA 會從鏡頭輸入影像資料，經影像預處理 (Image Preprocessing) 後，將資料傳遞給卷積神經網路 (Convolution Neural Network, CNN)，並交由神經網路判斷手勢，再與對手決定勝負，最終於周 邊裝置 (Peripherals) 輸出遊戲結果。

若是玩家獲勝；若是玩家敗北；若是玩家平手。

在本專案中，我們做出了以下貢獻

(1) 設計一份 VGA 遊戲

(2) 設計一份神經網路

(3) 對神經網路進行較為深度的調教

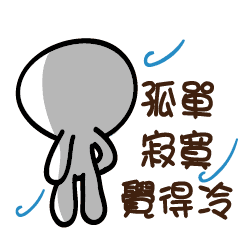
(4) 設計並公開一份全 Combinatorial Circuit 的 FPU (Floating point Processing Unit)

(5) 利用兩片 FPGA 以最大化利用硬體資源

# 二、動機

在這冷冷的冬天裡，宿舍的室友卻都和男朋友跑去約會了QQ，找不到人陪我玩遊戲的我只好突發奇想，製作一款可以自己和自己玩的遊戲，CNN猜拳機就這樣誕生。

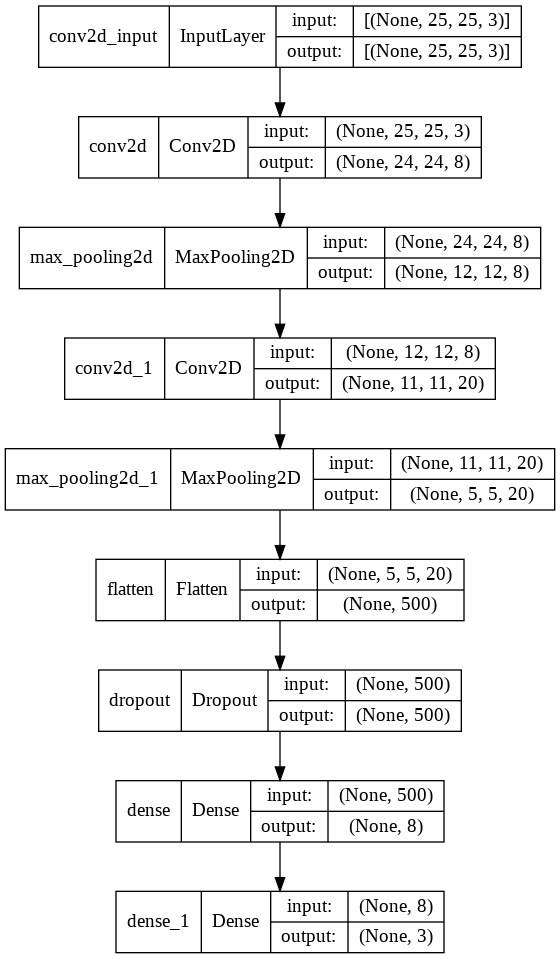
一開始只是想玩剪刀石頭布，但只用輸入選項的方式太無趣了，於是決定利用影像辨識的方式進行實作，這樣就能夠體驗出拳的爽感，另外為了增加遊戲的刺激性，也增加了亂數的對手與玩家較勁，看看到底是誰比較厲害，於是成品就慢慢的成形了！



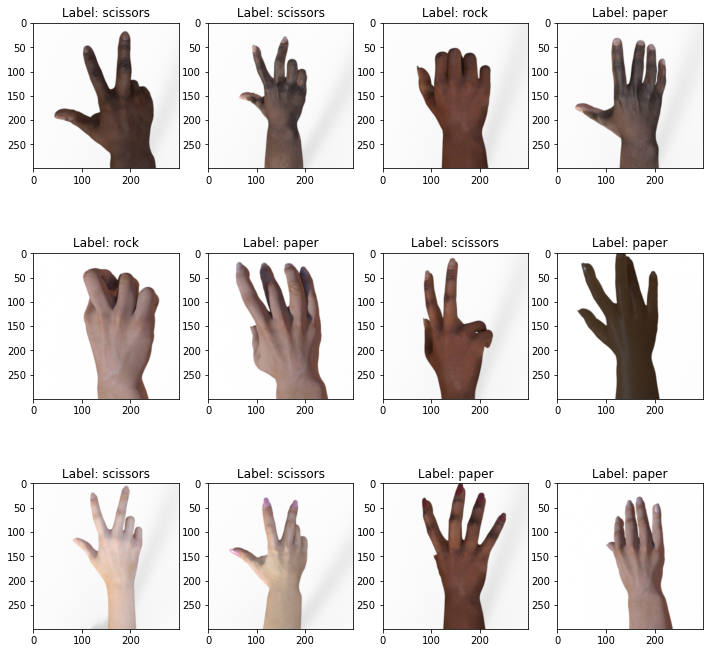
# 三、系統規格

## (1)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。



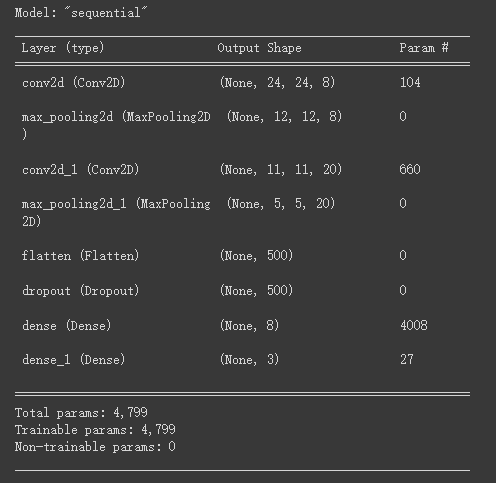
下圖為訓練用的壓縮後圖片，可以見到壓縮後只有25x25的解析度。



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。



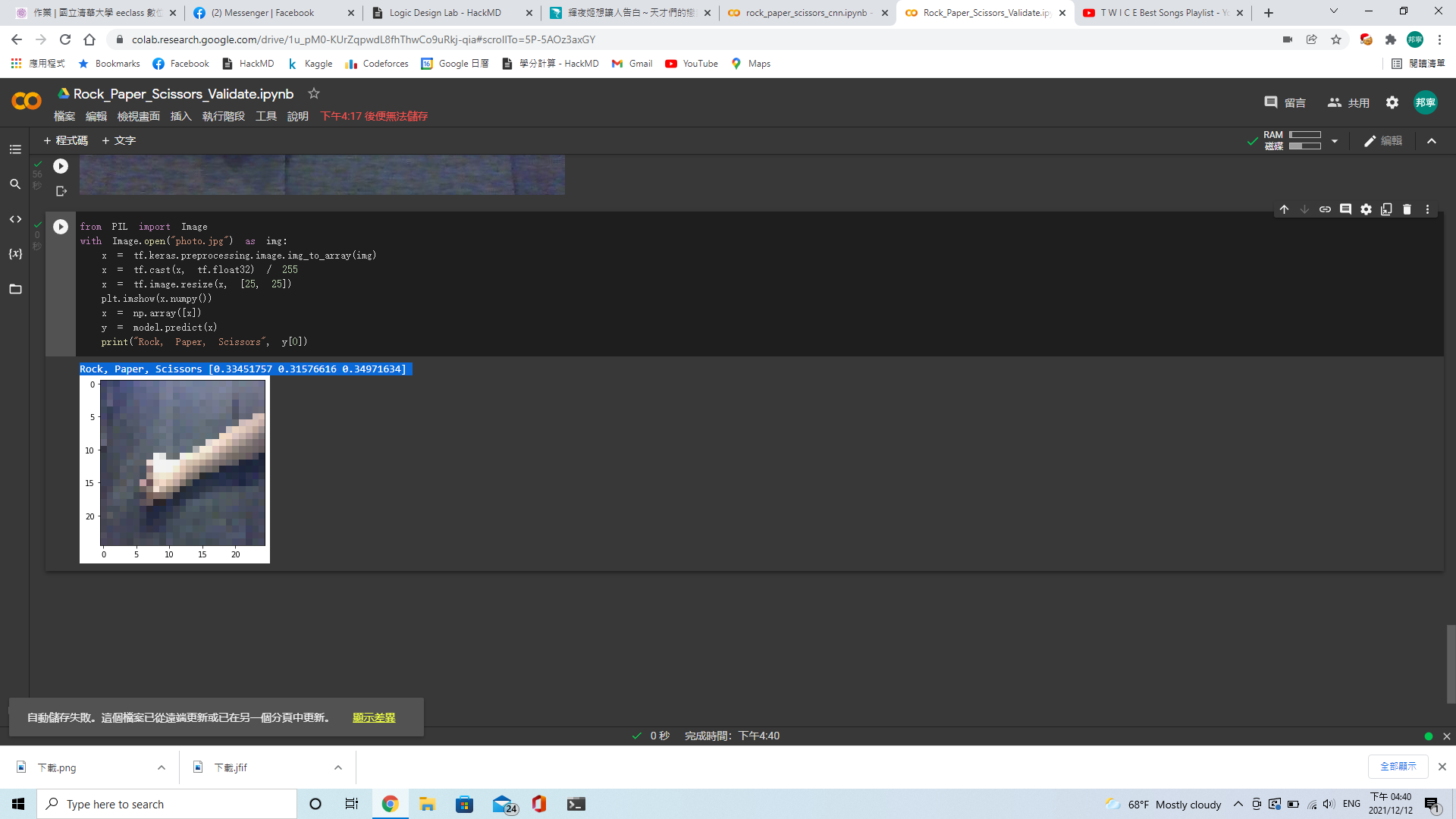
模型採用Adam進行最佳化，學習率採0.001，並訓練30個Epoch，下圖為訓練過程。由圖可見，訓練資料集與測試資料集無明顯差距，由此可以排除模型過擬合(Overfitting)之可能。

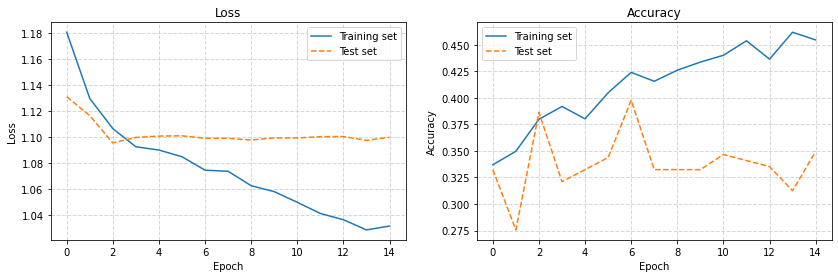


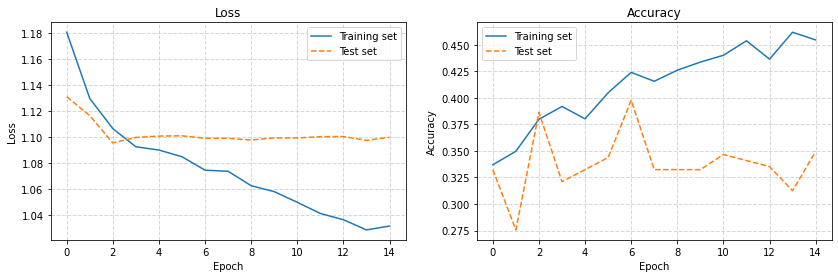


在現實世界測試(Real Life Testing)時，剪刀與布都能被準確的辨識，唯獨石頭無法被精確辨識；不僅如此，測試結果也顯示模型對光照角度、光線強度相當敏感。因此，於實際應用時，應維持穩定光源。下圖展示石頭無法被準確辨識。



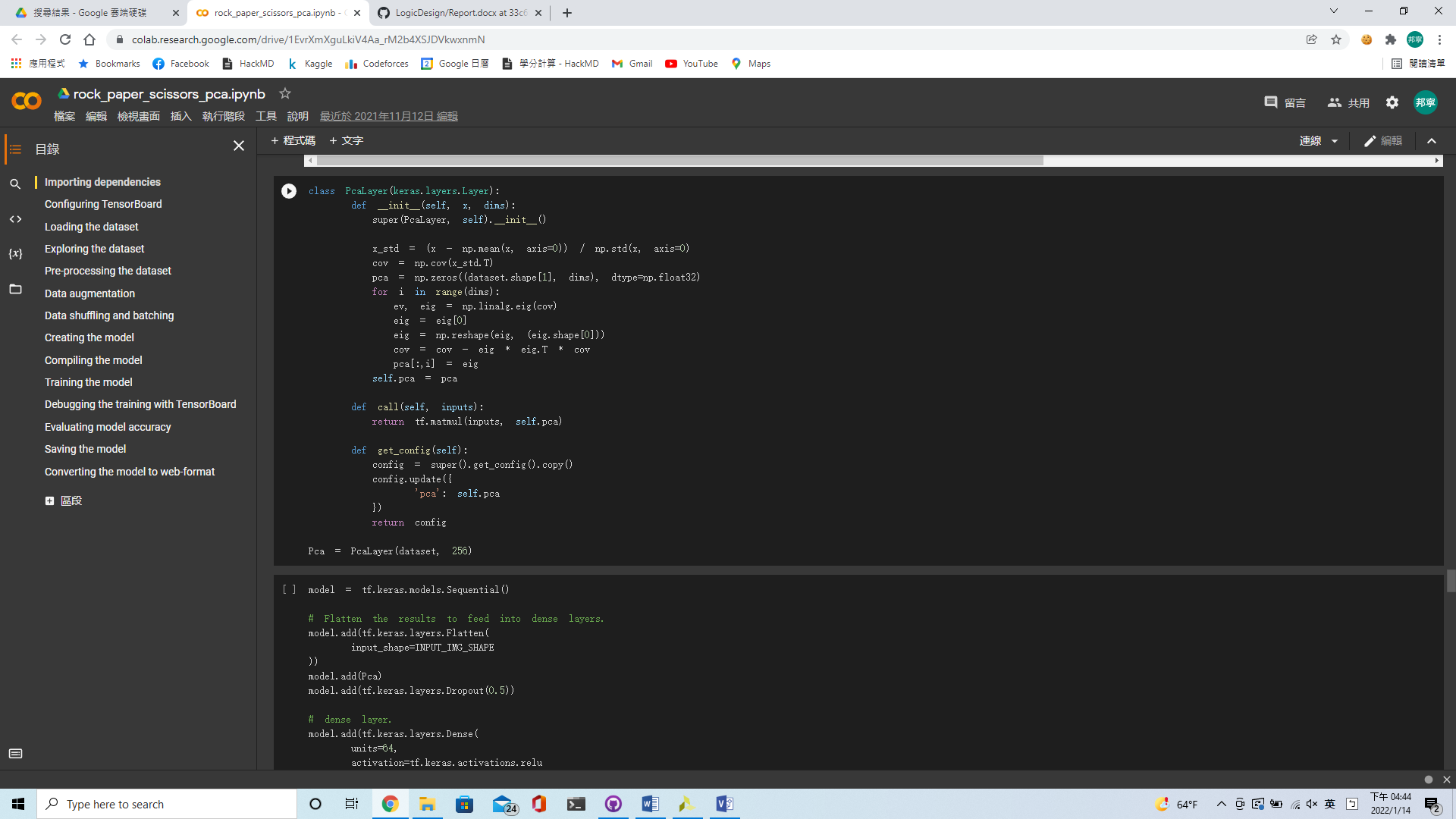


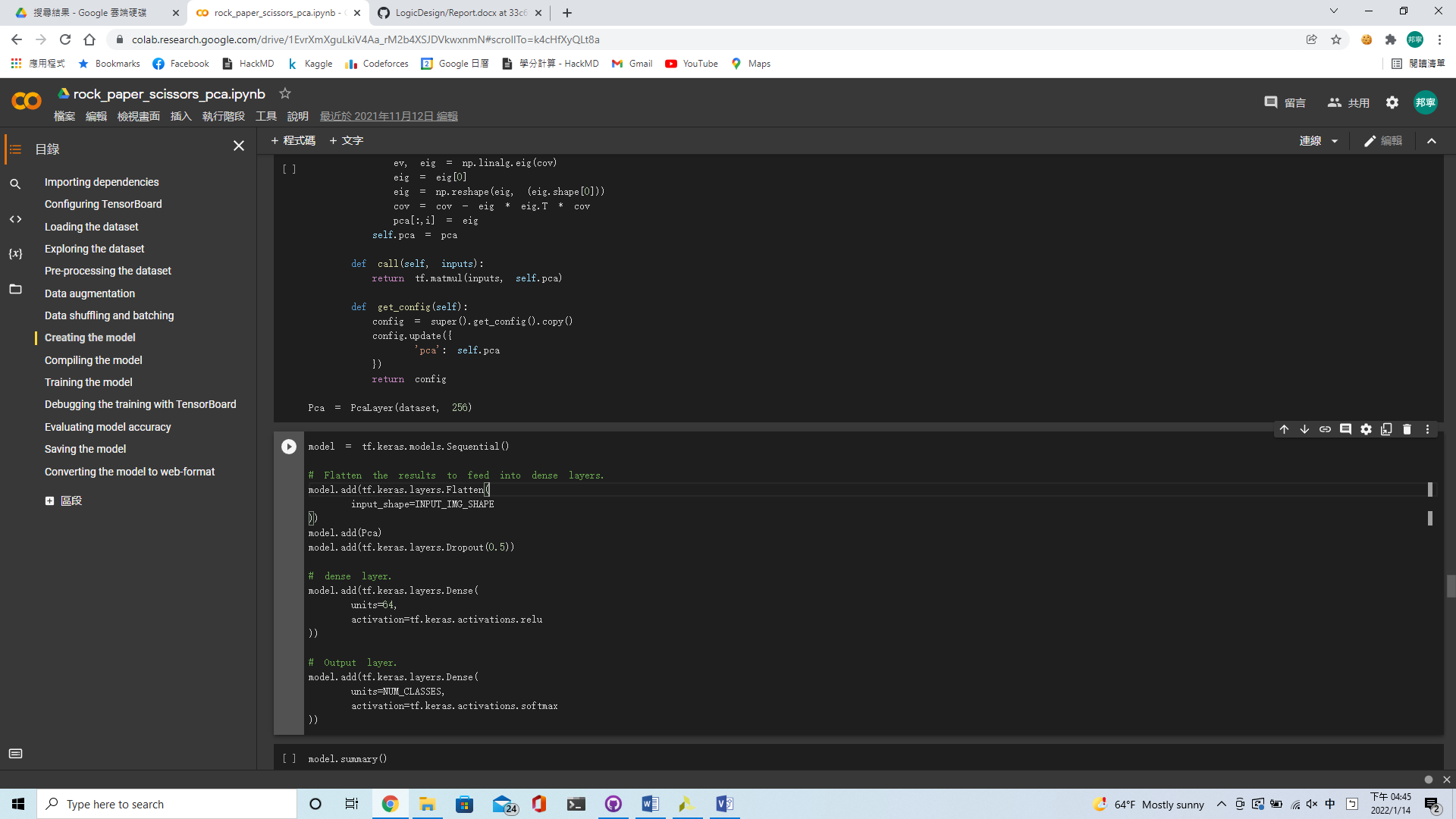
為降低神經網路的參數使用量，我們曾採用PCA (Primary Content Analysis, 主成分分析)去提起圖片特徵，不幸的是，計算PCA需要過高的時間複雜度，且小規模測試後的實驗結果不佳，因此，最後並無採用該計算方法。



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。





## (2)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。

### Convolution Layer的Schematic Design



### Convolution 2D的Schematic Design











### Convolution Kernel的Schematic Design







### Max Pooling的Schematic Design



下圖為Relu的Schematic Design。



### Dense Layer的Schematic Design



### .Full Connect的Schematic Design









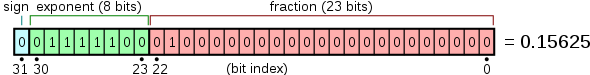


## (3)浮點數運算器之實作

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。



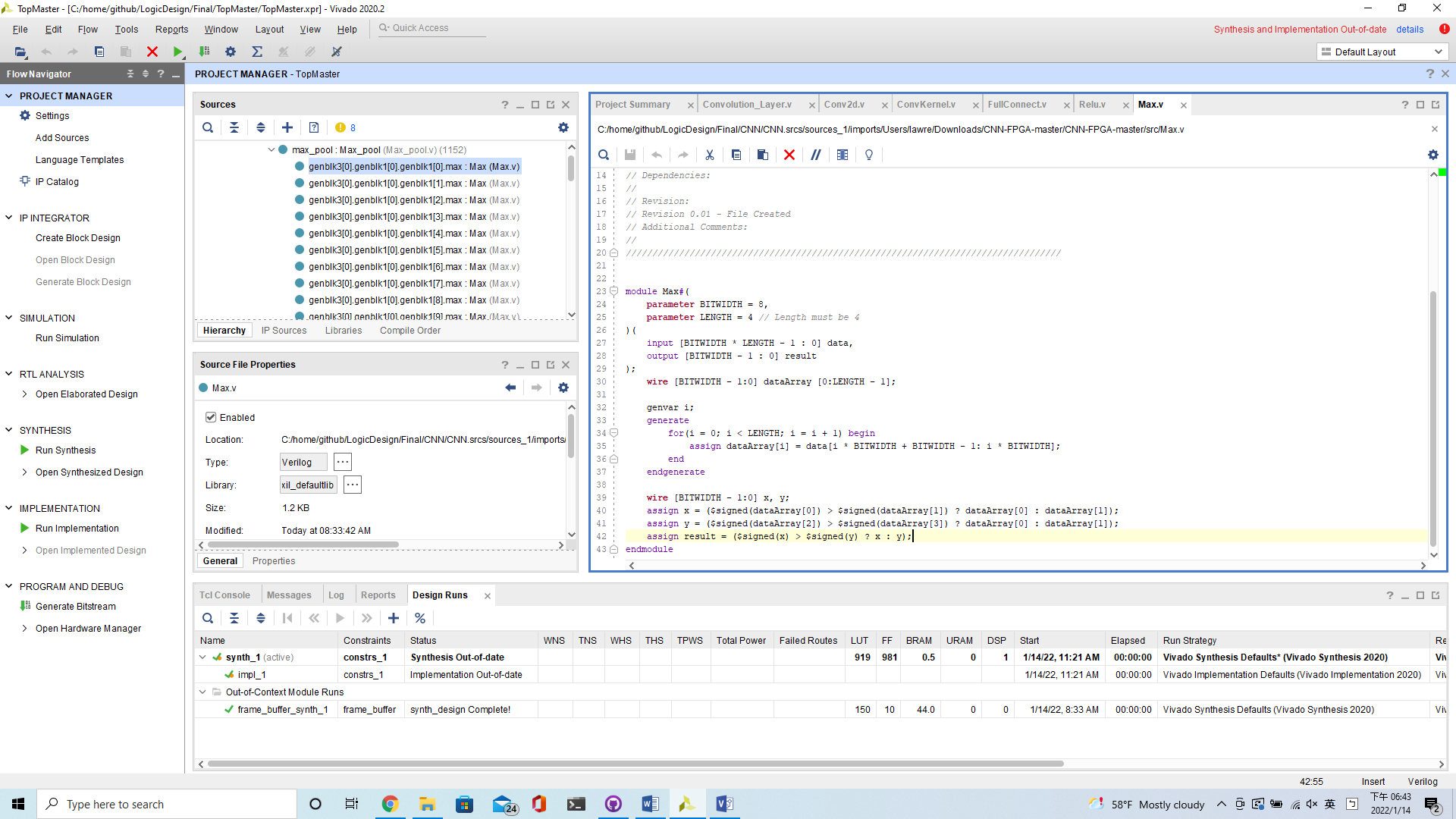
上圖中，Exponent部分為，而Fraction部分為，而，由此規律，我們可以找出浮點數轉整數的方法。

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

1. 正負號
2. Exponent，也就是指數部分
3. 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。



### 加法器













### 乘法器











## (4) 遊戲進行與螢幕顯示相關設計

### 1. TOP\_SLAVE

* 設計SPEC：

input clk

input rst

input mid\_but

input [3-1:0]data\_in

input request

input valid

output notice\_slave

output ack

output [3:0] vgaRed

output [3:0] vgaGreen

output [3:0] vgaBlue

output hsync

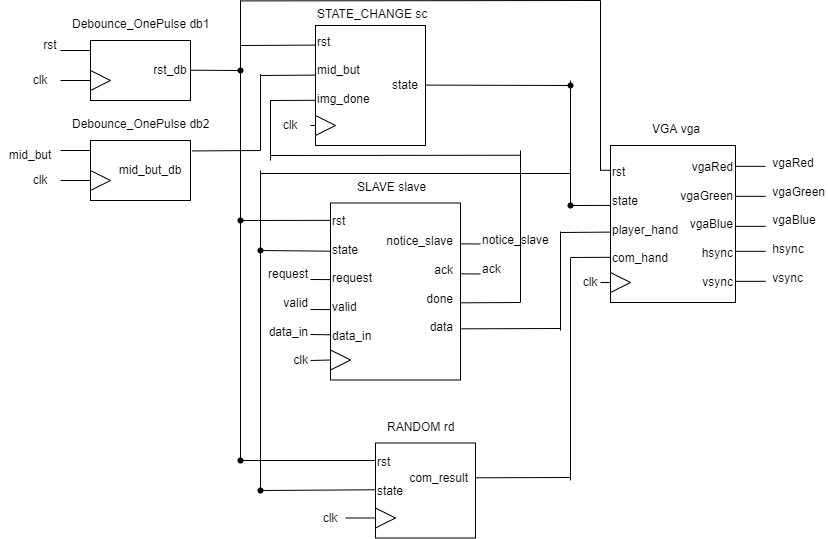
output vsync

* 設計說明：

此Module為TOP module，主要將所有module接線接好，包含以下：

|  |  |
| --- | --- |
| Module | 功能 |
| Debounce\_OnePulse | 把雜訊處理掉 |
| STATE\_CHANGE | 進行STATE的改動 |
| RANDOM | 產生FPGA出拳亂數 |
| VGA | 進行VGA螢幕顯示 |
| SLAVE | 接收Master回傳之辨識結果 |

* 設計圖：



### STATE\_CHANGE

* 設計SPEC：

STATE\_CHANGE module:

Input clk

Input rst (已經進行完debounced and one pulsed的reset)

Input mid\_but (已經進行完debounced and one pulsed的mid\_but)

Input img\_done (當手勢被辨識完)

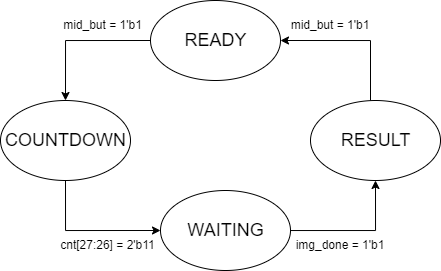
Output [1:0] state

* 設計說明：

總共設計4個STATE，包含READY、COUNTDOWN、WAITING、和RESULT：

|  |  |  |
| --- | --- | --- |
| STATE | 說明 | 換下一個STATE條件 |
| READY | 開頭畫面準備開始遊戲 | mid\_but被按下 |
| COUNTDOWN | 倒數畫面讓玩家有時間將手固定在畫面前 | 倒數完3秒 |
| WAITING | 等待TOP\_Master回傳辨識結果 | 當值回傳完，也就是img\_done被拉起時 |
| RESULT | 顯示辨識結果、亂數結果、和輸贏 | mid\_but被按下 |

* 設計圖：



### RANDOM

* 設計SPEC：

RANDOM module:

Input clk

Input rst (已經進行完debounced and one pulsed的reset)

Input [1:0] state

Output [1:0] com\_result (利用2bits去輸出這次亂數之猜拳結果)

LFSH module:

Input clk

Input rst (已經進行完debounced and one pulsed的reset)

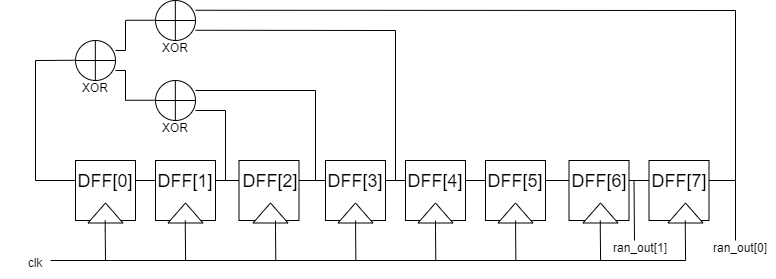
Output [1:0] ran\_out (利用2bits去輸出這次亂數結果)

* 設計說明：

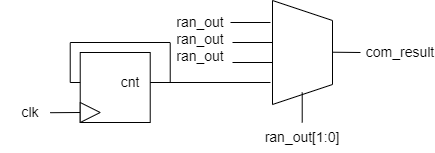
利用LFSH去獲取一2bits亂數，但由於猜拳只會有三種可能性，所以當亂數結果為2’b11時，我們根據register cnt當時的值進行輸出。其中cnt是包含每clk會加1’b1且不會超過2’b10的性質，所以RANDOM不會輸出一非剪刀、石頭或布的結果。

* 設計圖：

LFSH：



RANDOM：



### SLAVE

* 設計SPEC：

與Lab6 chip2chip之設計相同，下面用粗體標出同處：

SLAVE module：

input clk

input rst\_n

**input [1:0] state**

input [3-1:0]data\_in

input request

input valid

output notice\_slave

output ack

**output done**

**output [1:0]data**

slave\_control module ：

input clk

input rst\_n

**input [1:0] top\_state**

input request

input [3-1:0] data\_in

input valid

output reg ack

output reg notice

**output reg [1:0] data**

**output reg all\_done**

* 設計說明：

利用Lab6 sample code進行更改，僅在slave\_control中加入top\_state來判斷現在的state，讓其能只在”WAITING”的state收取資料。

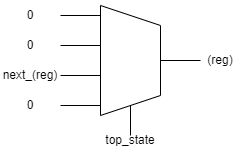
另外在slave\_control中多加了輸出all\_done，也就是SLAVE中的done，進行收取資料完成的輸出，以告訴STATE資料以收取完畢。

而由於剪刀石頭布只有三種可能性，slave\_control以及SLAVE中的data改為2bits。

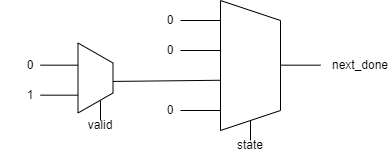
* 設計圖：

下方以新增及改動的部分為主：

每一clk傳值會依據top\_state進行更改



增加done及next\_done



### VGA

* 設計SPEC：

基本上與之前的sample code相同，下面用粗體標出有改動的地方。

VGA module ：

input clk

input rst (已經進行完debounced and one pulsed的reset)

**input [1:0] state**

**input [1:0] com\_hand**

**input [1:0] player\_hand**

output [3:0] vgaRed

output [3:0] vgaGreen

output [3:0] vgaBlue

output hsync

output vsync

mem\_addr\_gen module ：

input clk

input rst (已經進行完debounced and one pulsed的reset)

**input [1:0] state**

**input [1:0] com\_hand**

**input [1:0] player\_hand**

input [9:0] h\_cnt

input [9:0] v\_cnt

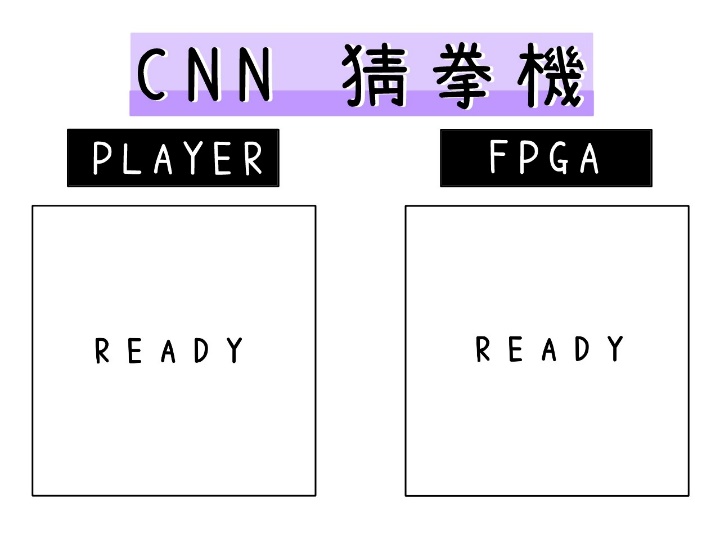
output reg [16:0] pixel\_addr

* 設計說明：

實作方式為利用IP 匯入一個coe檔，此coe檔包含了所有圖片的RGB資料，當變到不同的state時，改變畫面的呈現方式：

1. READY

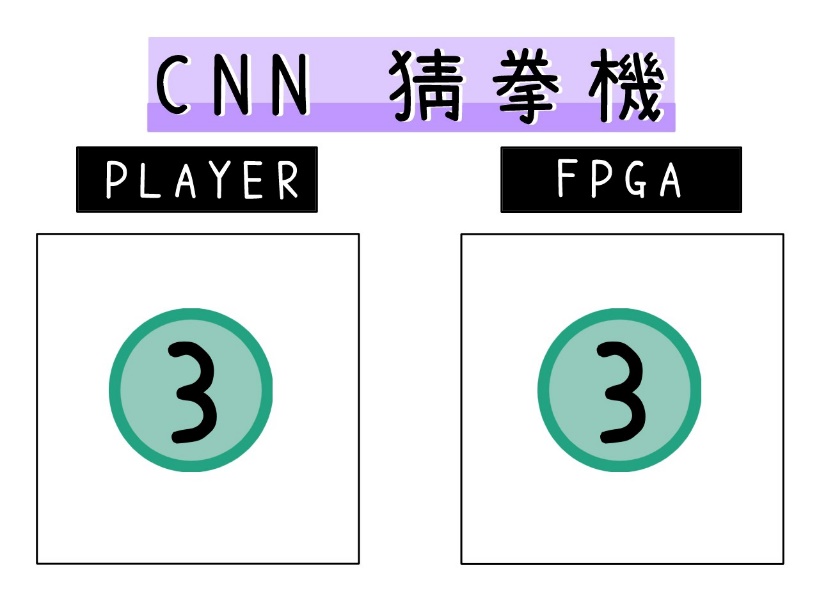
顯示遊戲主畫面以及”ready”等字樣，而字樣呈現方式則是在或是、且時，將pixel\_addr的輸出改為後面已存好的圖的addr。



1. COUNTDOWN

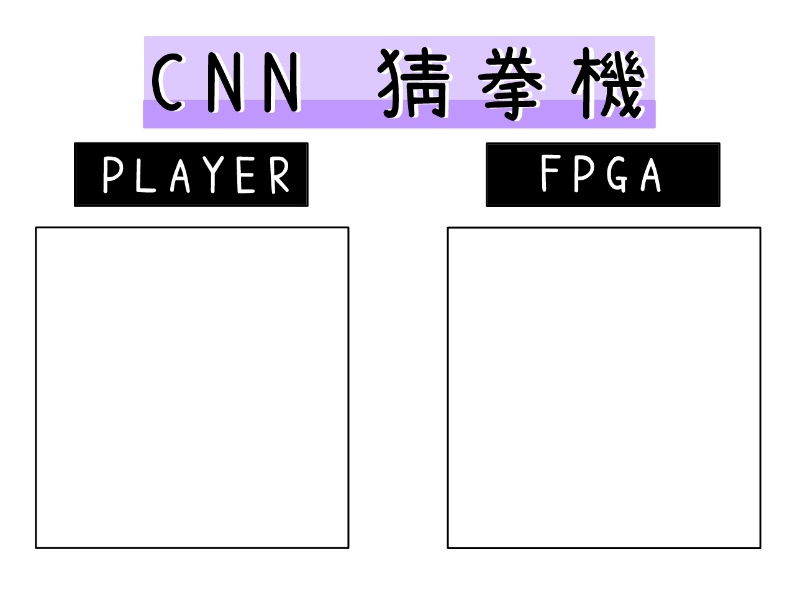
顯示遊戲主畫面以及倒數圖樣，而圖樣呈現方式則是在或是、且時，將pixel\_addr的輸出改為後面已存好的數字圖的addr，且利用正在倒數的cnt[27:26]去判斷現在應該輸出哪張圖片。

|  |  |
| --- | --- |
| cnt[27:26] | 圖片 |
| 00 | 3 |
| 01 | 2 |
| 10 | 1 |
| 11 | 不顯示(此時進行換state) |



1. WAITING

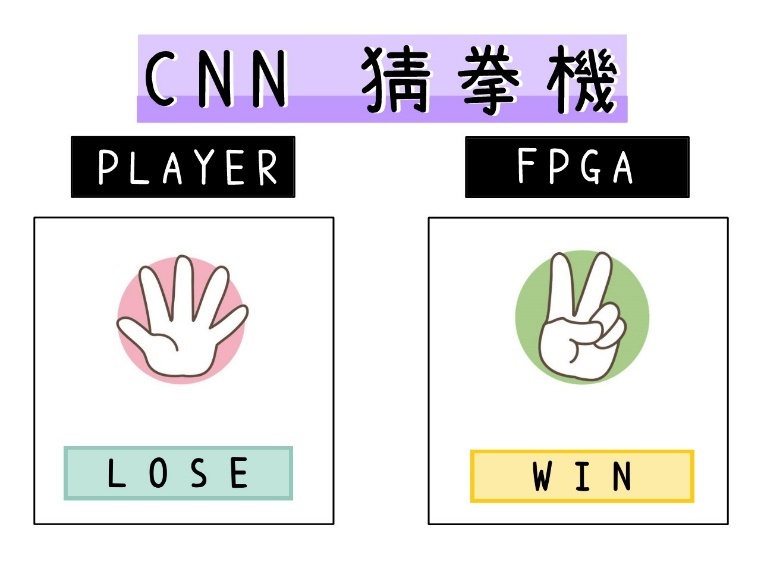
僅顯示遊戲主畫面。



1. RESULT

顯示遊戲主畫面以及倒數圖樣，而圖樣呈現方式則是在或是、且時，將pixel\_addr的輸出改為剪刀、石頭或布的addr，且利用input player\_hand和com\_hand去判斷現在應該輸出哪張圖片。

但當要顯示結果時，因為和包含於和中，所以在先前的判斷中，當時要將輸出改為結果的圖，而之後在和中野依樣判斷v\_cnt去進行輸出改動。



# 四、實驗結果

### (1) 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。

1. 資料集的光源都是單點光源

2. 資料集的光源都是無色光源

3. 鏡頭的輸入與模型的訓練資料有差異

### (2) 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。

### (3) 其他因素對神經網路的敏感性

1. 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。

2.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。

3.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。

4.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。

5.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

## (1)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。

## (2)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。

## (3)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

在本專案中，我們做出了以下貢獻

(1) 設計一份 VGA 遊戲

(2) 設計一份神經網路

(3) 對神經網路進行較為深度的調教

(4) 設計並公開一份全 Combinatorial Circuit 的 FPU (Floating point Processing Unit)

(5) 利用兩片 FPGA 以最大化利用硬體資源

本專案不僅設計了一台猜拳機，更為社會開源了一份公有智慧財，綜上所述，本專案實不愧對教授、助教本學期的諄諄教誨，亦為本學期的心血結晶！