國立清華大學 電機工程學系

實作專題研究成果報告

Convolutional Neural Network Processor for Animation Super Resolution

適用動畫超解析之

卷積神經網路處理器設計

專題領域：系統組

組　別：　A49

指導教授：黃朝宗

組員姓名：林俊曄、張嘉祐、陳昭霖

研究期間：108年7月1日至108年5月底止，共計11個月

Abstract

Nowadays with advanced technology, many people use smart phones or tablets to watch animations. Without restrictions, they can watch their favorite animations anytime and anywhere. However, sometimes in condition of poor network, the transmission rate is low, resulting in decline of resolution, which reduces the quality of watching anime for viewers. If we download high-resolution videos to our smart phones, we not only have to wait for the download time but also use a lot of precious memory space. Therefore, we want to train a super-resolution CNN network for animation so that the resolution of images with poor quality can be improved and become clearer after processed by this network. Then, we will implement the network on hardware and complete a CNN hardware accelerator in order to achieve the throughput of animation standard (output 24 images per second). Our ultimate goal is to apply this hardware accelerator to a portable electronic product such as smart phones or tablets. From this, we can watch high-resolution anime by using a small amount of memory space to store weight, bias, and the activation of the computing process. Not only can it save memory space, but also maintain a high-quality viewing experience when the signal is not that good.

In this research, we decided to train an SR-CNN model for One Piece and implement it on hardware. Our research plan is a complete system design, covering software simulation, hardware simulation and synthesis, and APR. The software part includes the analysis and selection of Hyperparameters, Data Quantization and Dynamic fixed point, and other extraction strategies. The purpose is to ensure that the performance of hardware will not reduce much compared to the performance of software; The hardware part includes architecture design, simulation and optimization; APR part includes using Synopsys Design Compiler to do synthesis, using IC Compiler to produce APR Layout in the environment of TSMC 40nm process, and finally analyzing the Timing, Area, and Power of the APR Layout and the synthesis circuit.

Following is our APR result. Timing is 5.4ns, Area is 936079μm^2, Dynamic Power is 50.96mW, Leakage Power is 4.46mW, and the number of cycles to output a high-resolution image is 10,715,029. Therefore, the throughput of our design is 17.28 images per second. The PSNR of output image compared to the high-resolution image used to training is 29.11dB in average. In contrast to the resolution of the input image, the resolution of the output image is improved significantly.

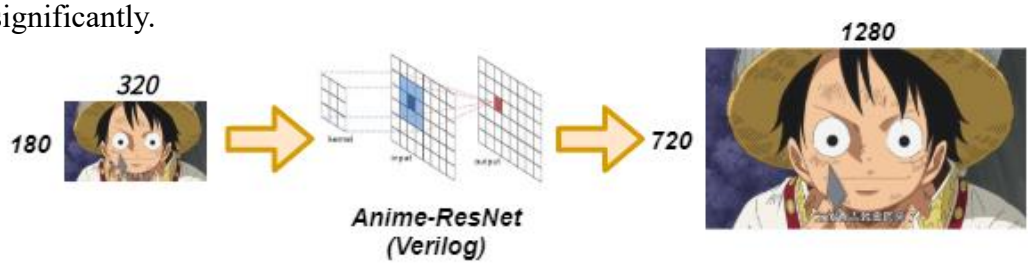


Figure 1. Super Resolution of Anime

摘要

在這個科技發達的世代，很多人使用手機或平板看動畫，不受限制，隨時隨地都可以觀賞自己喜歡的動畫，但有時候在網路不好的狀況下，傳輸速率變差，導致畫質下降，降低了我們的觀看品質。如果先行下載高畫質的影片至手機內，不僅要等待下載的時間而且會占掉許多寶貴的記憶體空間。因此我們想訓練出一個針對動畫風格的超解析之 CNN 網路，讓畫質較差的圖片經過這個網路後解析度能提高，變得更加清晰，並將這個網路以硬體實作出來，使 Throughput 能達到動畫等級的規格(每秒輸出24張圖)，完成一個 CNN 硬體加速器。最終目標為將此硬體應用於手機或平板這種方便攜帶的電子產品上，讓我們以少量的記憶體空間存放 Weight、Bias、運算過程的 Activation，便能觀賞高畫質的動畫。不僅可以節省記憶體空間，也能在網路傳輸速率不佳的情況下保有高畫質的觀賞體驗。

在這個專題研究中，我們決定訓練一個針對海賊王畫風的 SR-CNN Model，並實作在硬體上。我們的研究計畫是一個完整的系統設計，涵蓋軟體模擬、硬體模擬與合成、APR 三大部分。軟體部分包含 Hyperparameters 的分析與選擇，Data 的 Quantization 和 Dynamic Fixed Point 等提取策略，目的都是為了讓原本在軟體上的架構移駕到硬體上時表現不會降低太多；硬體部分包含架構的設計、模擬驗證及優化；APR 部分包含利用 Synopsys Design Compiler 合成電路、在 TSMC 40nm 製程的環境下使用 IC Compiler 製作 APR Layout，最後分析合成電路及 APR Layout 的 Timing、Area、Power。

我們最終 APR 結果的 Timing 是5.4ns，Area 是936079μm^2，Dynamic Power 是50.96mW，Leakage Power 是4.46mW，輸出一張高畫質圖片的總 Cycle 數為10,715,029個，由此可算出 throughput 為每秒17.28張。輸出圖片與用於訓練卷積神經網路的高清圖片比較後的 PSNR 平均為29.11dB，以人眼觀測輸入與輸出的圖片畫質有顯著的提升。

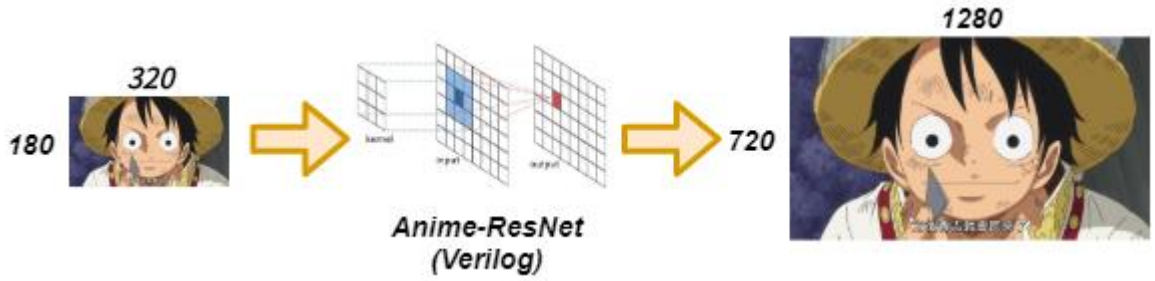


圖1　Super Resolution of Anime

目錄

圖目錄

表目錄

一、前言

我們的研究計畫是一個完整的系統設計。首先利用 Pytorch 訓練出針對海賊王動畫畫風的超解析之 CNN 網路，接著以 Verilog 設計這個網路的數位電路，最後使用 TSMC 40nm 的製程完成 APR 分析。現在的日本動畫，多數以每秒24幀來計算，也就是每秒24張圖片，而有時為了節省成本，會每兩幀或三幀用同一張圖片，也就是每秒12張或8張圖片。因此我們的目標 throughput 為每秒輸出24張高解析度的圖，確保我們的設計能實際應用於動畫上。

在 model 中我們使用 ResNet 架構，從較新的動畫集數中隨機收集約1000張的高清圖片，經過模糊處理後形成1000組 Data pairs(Low Resolution, High Resolution)作為 dataset 去訓練 Anime-ResNet Model，並以 PSNR 作為 Performance 的指標，最後從所有訓練出的 Model 中選取一個最適合的 Model 實作在硬體上。

在數位電路設計中，我們將訓練好的參數(Weight、Bias)存進 SRAM 裡，並用另外兩個 SRAM 交互存放運算過程中的 Activation。輸入為一張大小為320×180 pixels 的海賊王圖片，輸出為一張大小1280×720 pixels 的高解析度海賊王圖片。最後我們使用 TSMC 40nm 的製程來完成 APR，並對結果的 Timing、Area、Power 進行分析與討論。

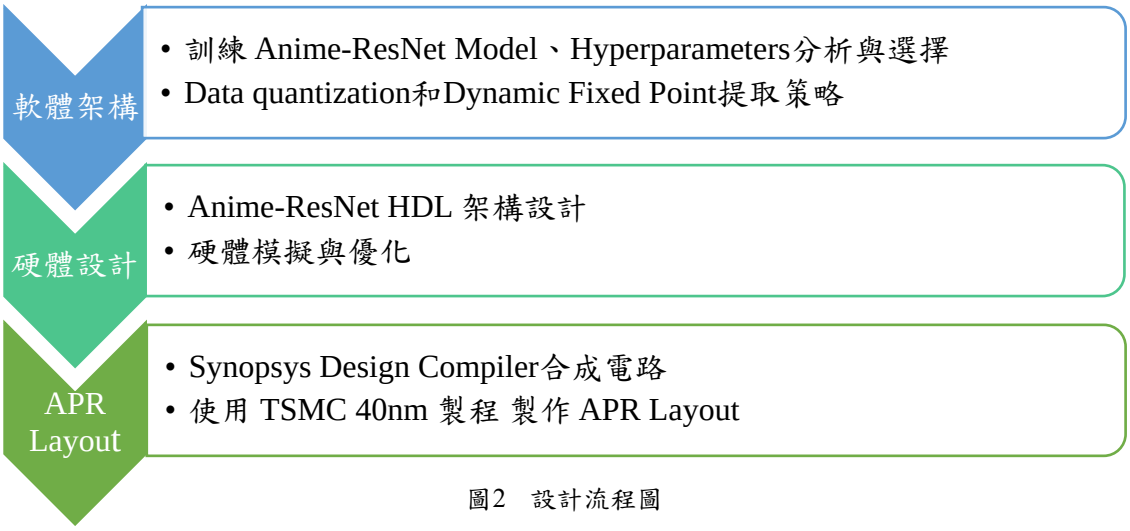


圖2　設計流程圖

二、Anime-ResNet Model Training:

1. Data Set 的選擇與處理:

為了提高舊版動畫的解析度，我們從 One Piece 新版動畫中，利用 Matlab 將一集動畫分成數十張的截圖，撇去一些不良的畫面後(黑幕、大量字幕)，我們總共提取1000多張圖片當作 Data Set，每張的大小是1280 pixel ×720 pixel。為了模擬出舊版動畫畫質較差的效果，我們透過 Adobe Photoshop 將照片做高斯模糊，並縮小成原圖面積的1/16倍，再將模糊後的照片 LR(Low Resolution)與原圖(HR)形成一組組 Data Pairs，並採用 PSNR 為 Performance 的指標，讓 Training 的過程中的 Output 有依據可以比對，不斷修正 Parameters，降低 Loss，使 Model 的 Performance 變高。

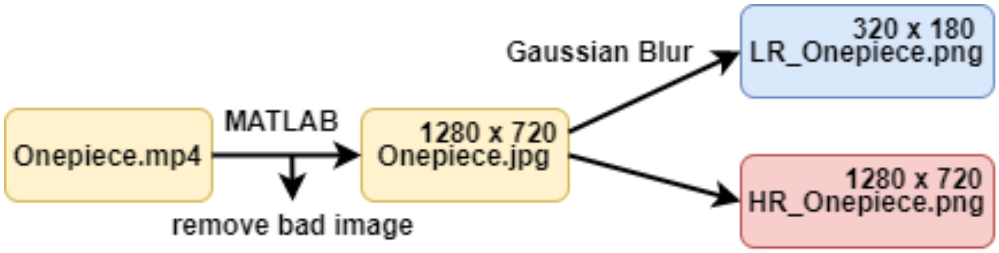


圖3　One Piece Data Set 處理過程

2. Anime-ResNet Model 的選擇與調整:

在軟體上，我們可以自由地調整 Hyperparameters (i.e. ResBlock 的數目 、Feature Map 的層數)，而且隨著 RES Block 跟 Feature Map 數量的增加，可以得到 PSNR 較好的結果；但對於硬體來說，需要額外考慮記憶體的限制，如果為了增加 PSNR，一味增加 RES Block、Feature map 的數量，會造成最終晶片的面積、power 大幅增加，使增加的 PSNR 不符合經濟效益。

我們分別訓練下列這些 Model，以20張不在 Data Set 內的圖片測試，平均後得出其相對應的 PSNR(如圖4所示)，並衡量 Performance 與該 Model 實作在硬體上所需的記憶體空間。我們最終決定採用 ResBlock*4 和 nFearture=24 的架構。此外，Model 在大約 nEpochs =100時會收斂，我們選用 nEpochs=200作為最終版本。(見圖5)

Parameter	nFeat	nResblock	nEpochs
Value	24	4	200

表1　Anime-ResNet Model 規格

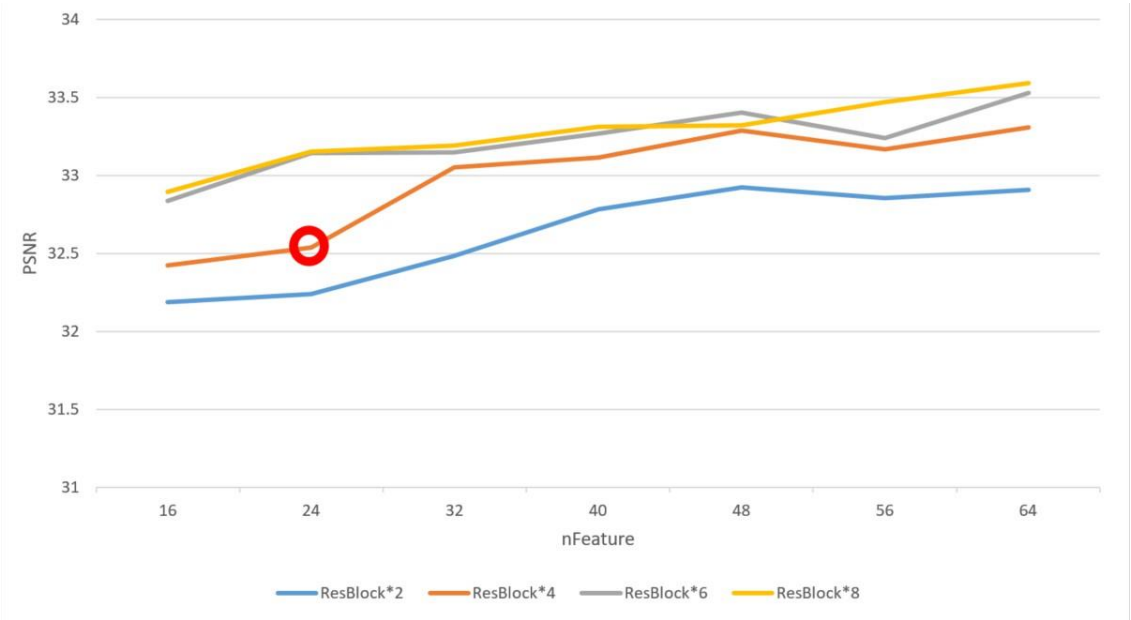


圖4　不同規格 model 的比較（紅圈是我們的 model）

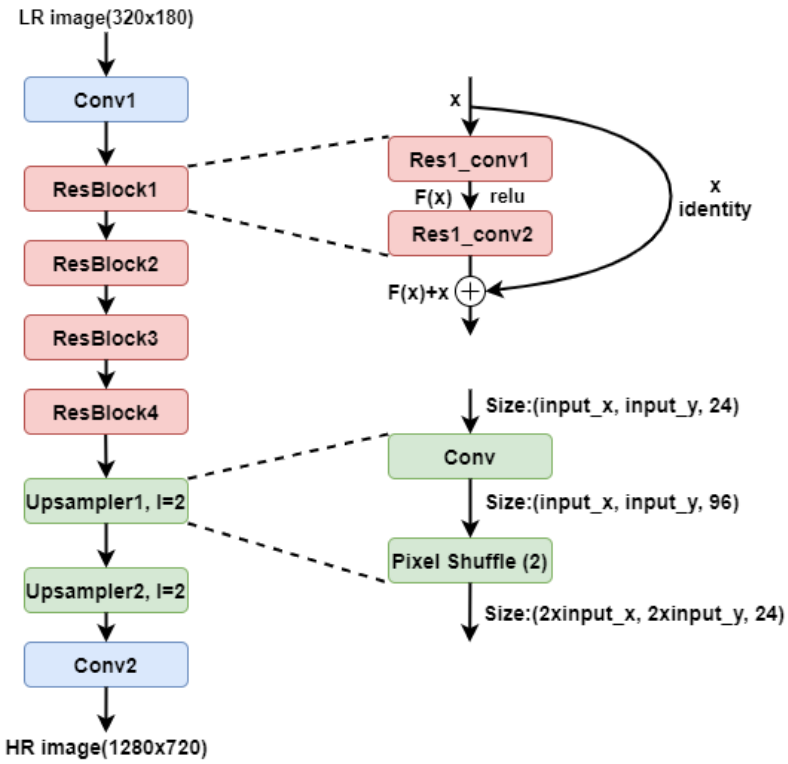


圖5　Anime-ResNet Model 整體架構

3. 針對硬體架構採取的 Quantization 策略:

首先，為了驗證硬體架構的正確性，我們取 Activation : 20 bits、Parameter: 8bits 作為第一版的 Golden 供硬體測試，可以包含運算過程中的最大與最小數值，不會有 Overflow 的情形發生。

I. Quantization Bits 數的考量:

確認完硬體的正確性後，為了節省硬體的面積、能量與 I/O Pin 數目，需要縮小 Activation Bits 和 Parameter Bits 數，以及考量運算過程中 Overflow 的情形。我們在每層 layer 後面加上 Saturation 和 Rounding 的處理，減少因 bits 數下降而損失的圖片品質。我們最終選擇 Activation: 16bits、Fractional Length: 7bits 當作最終版的的 bits 分配。

II. Dynamic Fixed Point 原理:

利用 Dynamic Fixed Point 可以將硬體的資源最大化，概念如圖6。在硬體中架構的設計與限制，不適合使用32bits 的浮點數作運算。因此我們分析每層 Parameters，根據其最大和最小值，判斷每層 Parameter 的小數點位置，擷取出最佳的分布區間，將每個 bit 的效用發揮到最大。 同樣 Activation 的部分也是相同的概念，從每層 Layer 的分布找出最佳分布區間，不同的地方為還要考慮上下層小數點的位置再額外作調整。

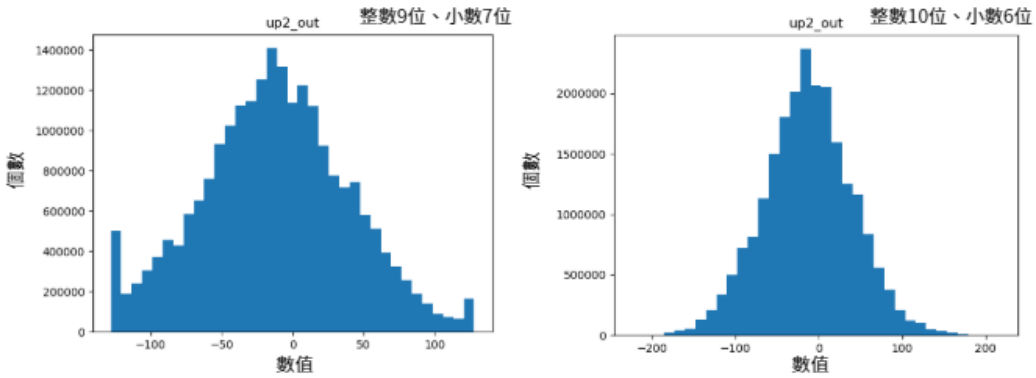


圖6　Dynamic Fixed Point Operation:

左圖有 Saturation，如果整數部分多1bit 表示(如右圖)，PSNR 會提高

4. 成果:

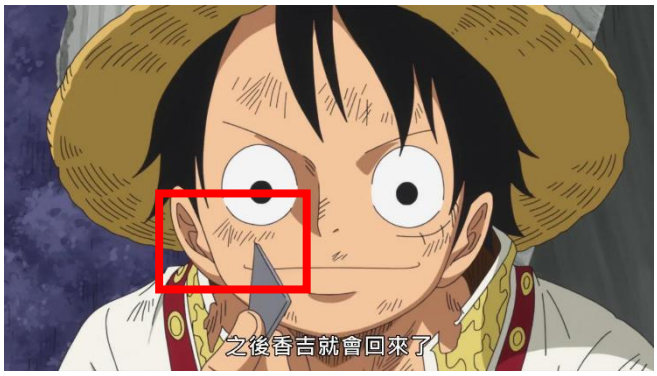


Image01 from the latest animation

HR (PSNR)

Anime-ResNet (Pytorch)

(34.736dB)

Gaussian Blur

Anime-ResNet (Hardware)

(30.254dB)

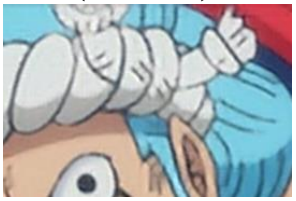
Image02 from the latest animation

HR (PSNR)

Anime-ResNet (Pytorch)

(34.891dB)

Gaussian Blur



Anime-ResNet (Hardware)

(29.978dB)

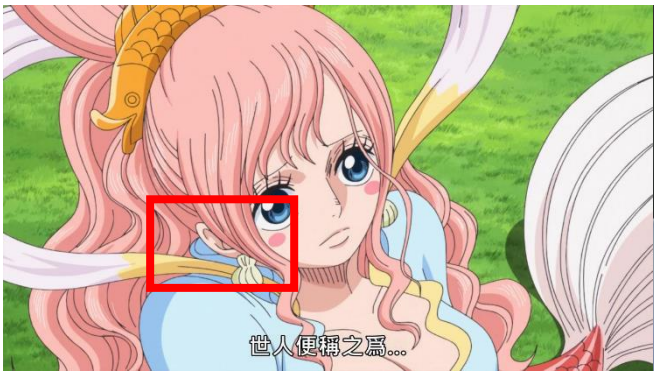
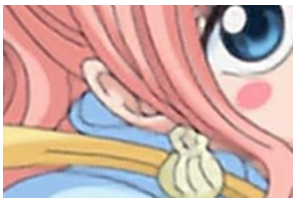


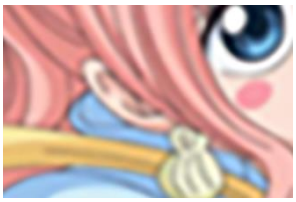
Image03 from the latest animation

HR (PSNR)

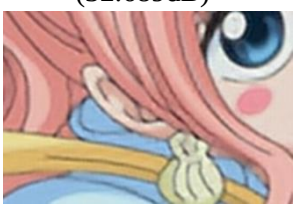


Anime-ResNet (Pytorch)

(32.689dB)



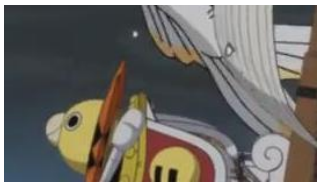
Gaussian Blur



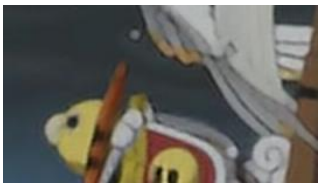
Anime-ResNet (Hardware)

(28.139dB)

Image04 from the latest animation

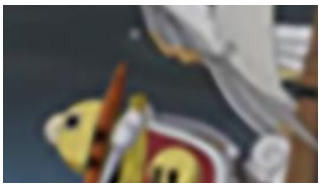


HR (PSNR)

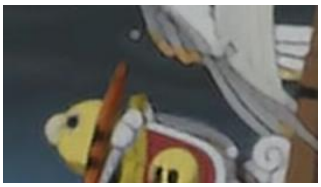


Anime-ResNet (Pytorch)

(33.970dB)



Gaussian Blur



Anime-ResNet (Hardware)

(31.425dB)

Image05 from the latest animation

HR (PSNR)

Anime-ResNet (Pytorch)

(33.486dB)

Gaussian Blur

Anime-ResNet (Hardware)

(29.978dB)

Image06 from the latest animation

HR (PSNR)

Anime-ResNet (Pytorch)

(35.208dB)

Gaussian Blur

Anime-ResNet (Hardware)

(28.750dB)

Image07 from the latest animation

HR (PSNR)

Anime-ResNet (Pytorch)

(34.392dB)

Gaussian Blur

Anime-ResNet (Hardware)

(28.056dB)

Image08 from the latest animation

HR (PSNR)

Anime-ResNet (Pytorch)

(35.006dB)

Gaussian Blur

Anime-ResNet (Hardware)

(28.612dB)

圖7　軟體硬體成果比較

三、硬體架構設計

1. 乘法器數量的估計與選擇

由 Model 中每一層的 Input 圖片大小、Output 圖片大小及 Channel 數，可以先算出輸出一張圖片總共需要做約 9×10^9次的乘法運算。接著我們的目標是每秒輸出24張圖片，並預估 Timing 為4ns，可以算出輸出一張圖片的 Cycle 數約為10^7個。將輸出一張圖片的總乘法運算數除以總 cycle 數即可得知一個 Cycle 要做900個乘法運算，也就是需要900個乘法器。由於 model 大多由24個 Channel 的3×3 Convolution 所組成，我們決定使用$9 \times 4 \times 24 = 864$個乘法器，如此可以在$3 \times 3$的 Convolution 運算中一個 Cycle 輸出4個值存進 SRAM。

2. 記憶體大小選擇

在硬體規格的設計上，最占用面積的其實是記憶體。Weight、Bias 及運算過程中產生的中間值(Activation)數量很多，若皆以暫存器存放，會占用非常大量的面積，因此我們使用 SRAM 來暫存這些值。由於 Model 的 Output 是1280×720 pixels 的圖片、運算過程中最多需要存放24個 Channel 的資料、Activation bitwidth 為16 bits，再加上我們處理 Padding 的方式是將 Padding 的0存在 SRAM 內，因此 SRAM 的大小為 $\frac{1284 \times 724 \times 24 \times 16}{8} \approx 45\text{MB}$。運算過程中我們使用兩組相同規格的 SRAM 交互存放，如圖8所示。由於我們設計一個 cycle 能輸出四個值存進 SRAM，因此我們將兩組 SRAM 的每個 Address 等分為四個 Bank，每個 Bank 大小為$2 \times 2 \times 24 \times$ Activation bitwidth，以簡化存取 SRAM 的設計。

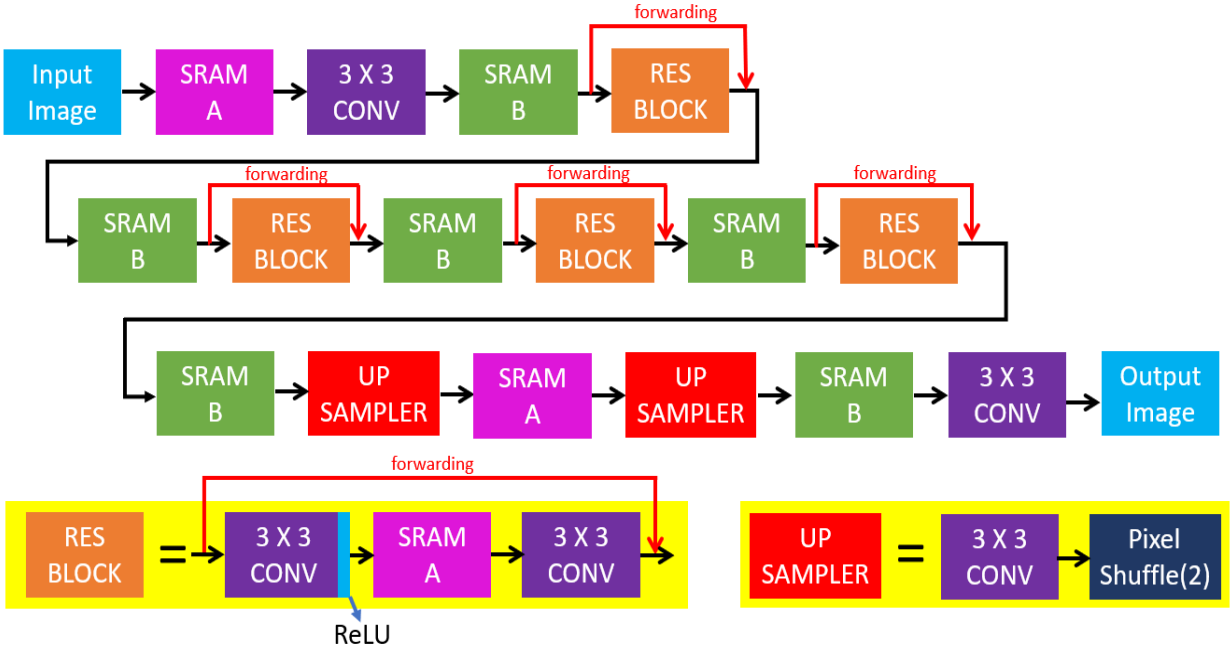


圖8　Data Flow

Input	320x180 image	SRAM for Activation	45MB x2 個 (SRAM_A & SRAM_B)
Output	1280x720 image	SRAM for weight	88KB x1 個
乘法器數量	864 個	SRAM for bias	0.4KB x1 個

表2　Hardware Specification

3. 運算架構設計

在 Resblock Unit 中有一個比較特別的部分是 Forwarding，也就是在 Resblock unit 中的第二層3 × 3 Convolution 運算完後要加上第一層3 × 3 Convolution 的 Input。由於我們使用兩組 SRAM(Group A、Group B)交互存取資料，第二層3 × 3 Convolution 運算完後要寫入 SRAM B，而第一層的 Input 也是存在 SRAM B，我們使用先讀後寫的方法，利用 Pipeline 錯開讀值與寫值的時間，先將 Forward 的值從 SRAM B 讀出來，處理完加法後再寫回去 SRAM B 的同一位置。如此雖然會有多一個加法運算的 Overhead，但就不必先存進 SRAM 再讀回來進行 Forwarding 的加法運算，可以節省大量的 Cycle，以及不用再多開一組 SRAM 去存值。

在 Upsampler 中要先做一層3 × 3的 Convolution 再做 Pixel Shuffle(2)，而3 × 3 Convolution 的 Input Channel 數為24，Output Channel 數為96，Pixel Shuffle(2)的 Output Channel 數為24。如果分兩階段做，先將第一層的輸出存進 SRAM 再讀出來做 Pixel Shuffle(2)，會非常浪費 Cycle 數，且 SRAM 的大小也會變為4倍，因此我們在硬體架構上設計成一個階段一次完成，做完 Convolution 運算寫回 SRAM 時直接寫入該 Activation 在 Pixel Shuffle 後的位置。

4. Pipeline 與 Cycle Time 考量

為了讓 Cycle Time 壓低至 4ns 以內，我們 Pipeline 切了四級(五個階段)，第一階段是讓從 SRAM 讀進來的資料 Delay 一個 Cycle，以避免外部訊號進入電路的時間過長；第二階段是每個 Input Channel(有24個)與3x3 Kernel 做完 Convolution 的總和；第三階段是將前一級的24個 Output 值分成一半，每12個相加；第四階段是前一級的兩個 Output 值相加並做 Rounding、Saturate、ReLU 的處理；最後一階就是輸出值。

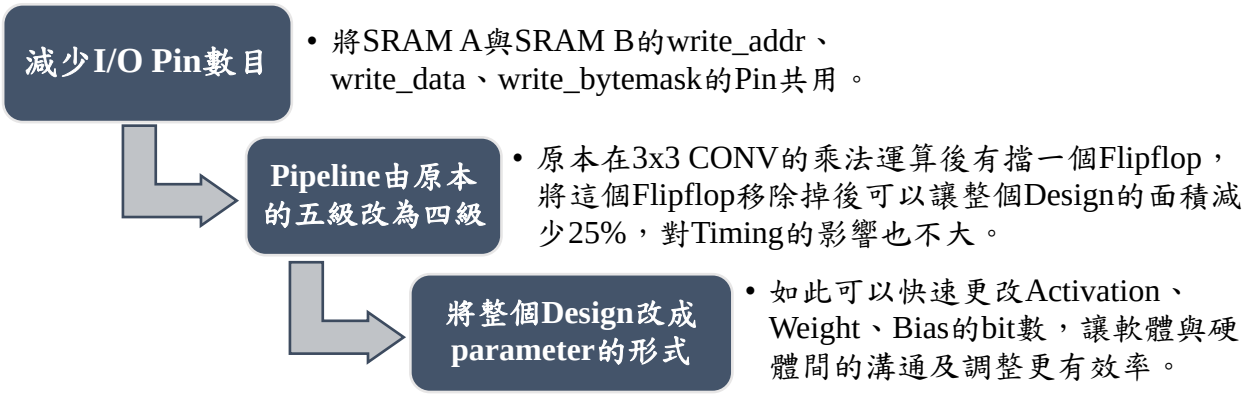


圖9　硬體優化流程

四、合成結果與 APR 分析

在前面軟體使用 Pytorch 建立 Anime-ResNet Model 及硬體使用 Verilog 完成硬體架構設計，並透過 Simulation 比對確定電路正確性後，我們利用 Synopsys Design Compiler 去合成電路產生 Netlist 及驗證 Gate-level Simulation。之後於 TSMC 40nm 製程環境下使用 IC Compiler 製作 APR Layout，而這一次 APR 為 Macro-Level Layout，使用了 EDA Cloud 實作，最終我們的硬體設計以 Activation 分為16bits 及20bits 兩種硬體設計，16bits 的版本有做 Quantization 的處理，20bits 的版本則沒有，並探討其 Timing、Area、Power。

1. Floorplan 規劃

我們最終將 I/O pin 的擺放如以下，將不同 Bank 相關的 Pin 擺在一起，藍色方塊表示將不同 bank(B0~B3)的 write enable, bytemask, data, address 擺在一起，照這樣的擺放方式能使讀寫 SRAM 資料更有效率，並且硬體設計方面將 SRAM A 跟 SRAM B 的 bytemask, data, address 共用 pin，也使 Floorplan 減少了8000多根 pin，大幅減少了我們合成的面積。

Output: Red　　Input: Green

1. SRAM A read data
2. SRAM A read address
3. SRAM A read enable
4. SRAM B write enable
5. SRAM B bytemask
6. SRAM B write data
7. SRAM B write address
8. SRAM B read data
9. SRAM B read address
10. Weight data
11. Weight address
12. Bias data
13. Bias address

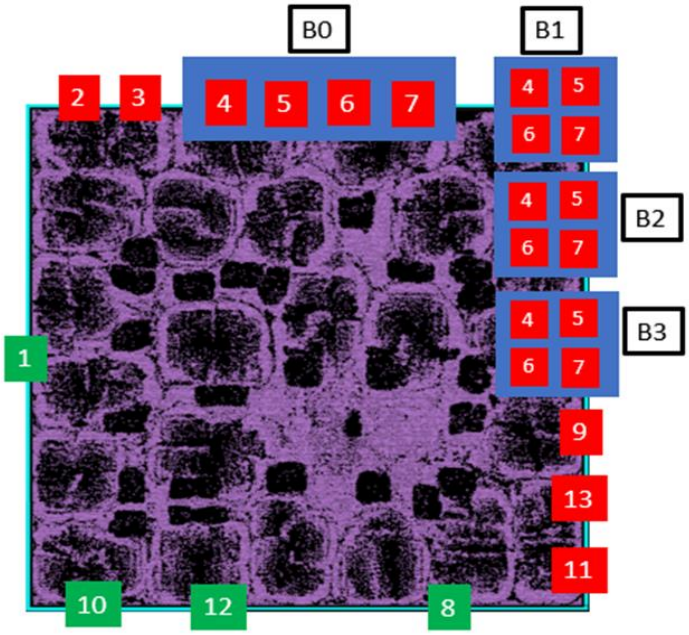


圖10 IO pin 擺放位置、Floorplan 規劃

透過 bytemask 能簡化寫值進入 SRAM 的設計，如圖 11，將 bytemask 特定的 bit 設為 0 便能夠改寫 SRAM 的資料，反之則保留原本 SRAM 中的資料。

Original SRAM data	32	23	15	17
Write data	-9	-16	-7	-1
Bytemask	0	1	1	0
SRAM output	-9	23	15	-1

圖11 4-bit bytemask 使用方式

2. Module Hierarchy

透過圖12可以看到我們的硬體設計主要是由 mul 和 map 兩種 Module 所構成，且相對應編號的 mul 和 map 會在相鄰的位置。而 mul 和 map 兩種 Module 的主要功能為：

map: 於一個 Cycle 從 SRAM 輸入的 activation 很多，

map 用於分配這些 activation 給相對應的 register，再傳給 mul 做運算。

mul: 做 activation 的乘法運算。

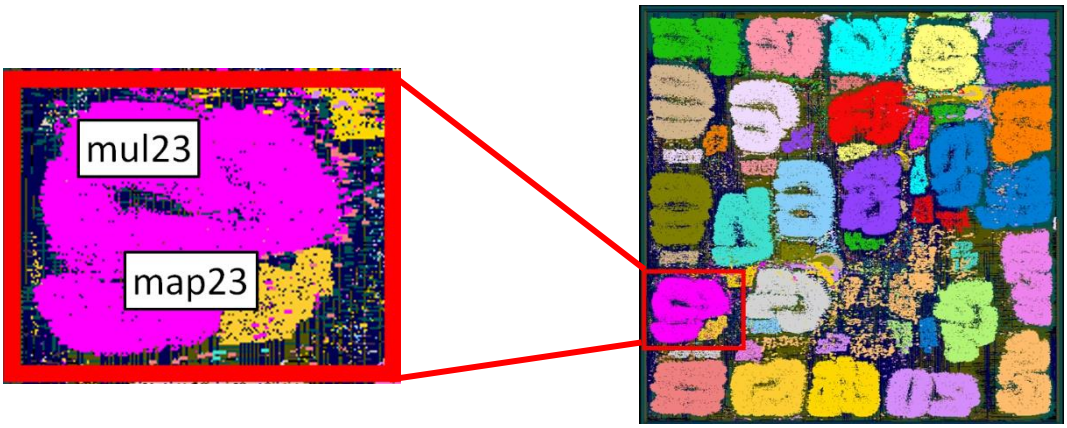


圖12 Hierarchy of different modules

3. Synthesis & APR 結果比較

結果分析我們分成20bits 的 Activation 以及16bits 的 Activation 作為探討，20bits 的版本目前已驗證 Gate Level Simulation 且完成 APR，而16bits 版本目前只有完成到 HDL Simulation 驗證，所以沒有完整的 APR Layout 無法與合成的結果比較，因此 16bits 的部分只討論合成後的結果。

規格:

—TSMC 40nm 製程

—Condition: SS corner

—Operating Voltage = 0.81V

Activation: 16bits	Synthesis
Timing	3.8ns
Area	$695695\mu m^2$
Power(dynamic):	$48.59mW$
Power(leakage):	$3.11 \times 10^3 \mu W$
Total Cycles	10,715,029 個
Throughput	24.56 image/sec

表3 Specification of 16 bits Activation

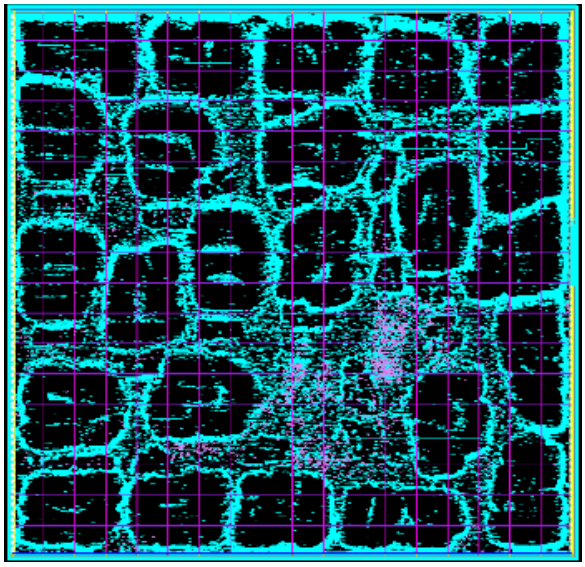


圖13 APR Layout

Activation: 20bits	Synthesis	APR (Utilization 0.5)
Timing	3.8ns	5.4ns
Area	$844568\mu m^2$	$936079\mu m^2$
Power(dynamic):	$56.046mW$	$50.96mW$
Power(leakage):	$3.57 \times 10^3 \mu W$	$4.46 \times 10^3 \mu W$
Total Cycles	10,715,029 個	10,715,029 個
Throughput	24.56 image/sec	17.28 image/sec

表4 Specification of 20 bits Activation

五、討論

硬體部分目前只支援到Weight與Bias的Dynamic Fixed Point處理，軟體部分目前完成到Activation、Weight、Bias都可以透過Dynamic Fixed Point的策略來提高Performance，將Activation納入處理後，PSNR可以提高0.3dB。

由於我們是針對海賊王一種風格Training，且動畫線條及顏色與真實世界的影像相比簡單很多，因此相較於其他Super Resolution的Model，Anime-ResNet的規模小很多，所需的Data Set僅需1000張、Model的深度及Feature數不用太大就能有很好的效果，下表是我們的Anime-ResNet與參考文獻“Enhanced Deep Residual Networks for Single Image Super-Resolution”[1]中EDSR Model的規模比較。

透過16bits quantization的處理可以使Anime-ResNet的I/O Pin再減少約8000多根、合成電路的面積減少約$150000\mu m^2$，而SRAM大小只需原先的$\frac{4}{5}$，但目前16 bits Quantization 的版本只完成了HDL模擬，APR Layout尚未完成，期待往後能將其完成。

	Anime-ResNet	EDSR
# Residual Blocks	4	32
# Filters	24	256
# Parameters	93K	43M
Loss function	L1	L1

表5 Anime-ResNet 與 EDSR 規模比較

六、參考文獻

[1] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, Kyoung Mu Lee, “Enhanced Deep Residual Networks for Single Image Super-Resolution”, Jul 2017.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, "Deep Residual Learning for Image Recognition", Dec 2015.

七、計畫管理與團隊合作方式

1. 計劃管理

專題的一開始，我們在暑假利用線上網站熟悉 Python 的語法與練習。開學後針對機器學習以及卷積神經網路閱讀許多相關論文，藉此了解深度學習的基本知識，同時修習積體電路設計實驗，培養數位電路設計的能力。在決定研究主題之後，我們開始收集 Data Set，利用 Pytorch 訓練 Super Resolution 的 Model，接著從訓練出的 Model 中選出最適合實作在硬體上開始進行數位電路設計，最後使用 TSMC 40nm 製程合成電路與完成 APR。

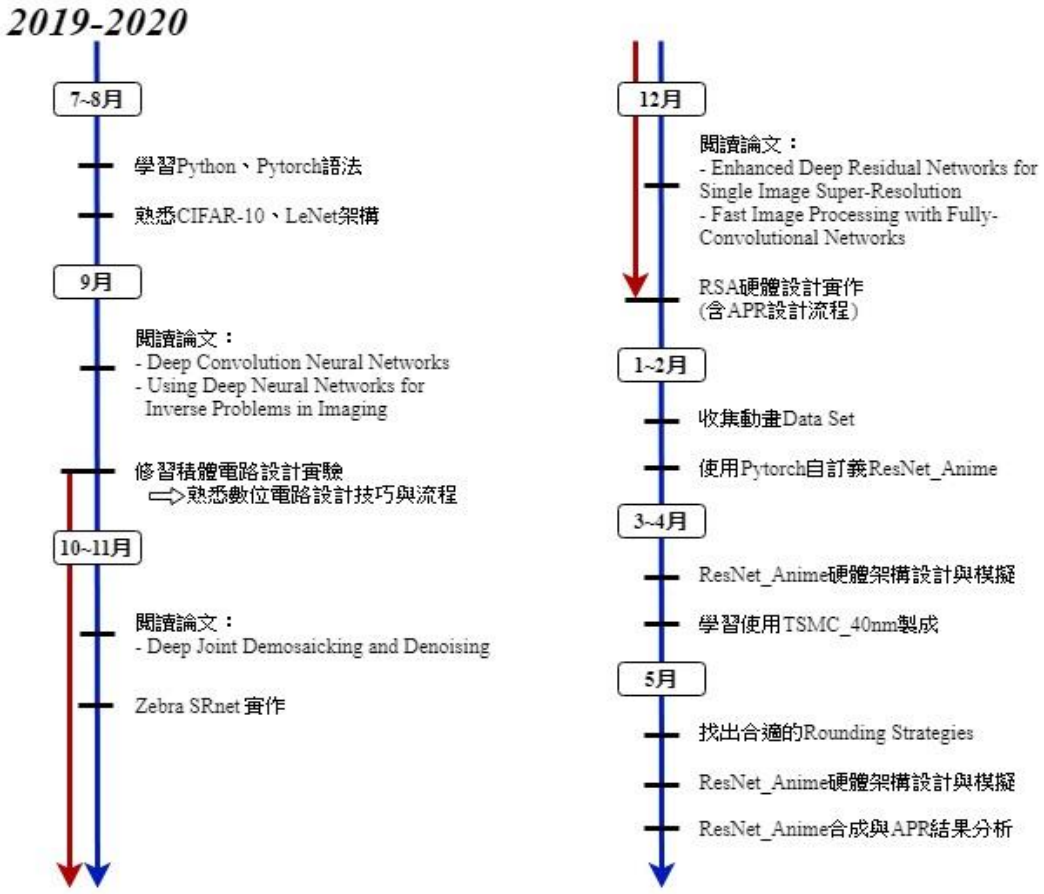


圖14 計劃管理流線圖

2. 團隊合作方式

基本上我們跟教授 meeting 的頻率約為一個月一次，教授會依據我們的進度給予指導與建議，訂定下一個月的目標，而每次 meeting 實驗室的學長姐也會在場，協助我們解決許多問題。在研究過程中我們依照軟體、硬體、APR 三個部分進行分工，優先負責各自份內的工作，當有人進度比較快時便會去支援其他組員。我們團隊內大約一周會討論一次，分享各自的進度與遇到的困難，如果討論後依然有疑惑便會將問題整理好，找時間詢問實驗室的學長姐。