Deep Neural Network FPGA Accelerator
深度神經網路FPGA加速器

組別：B128　　指導教授：鄭桂忠 教授

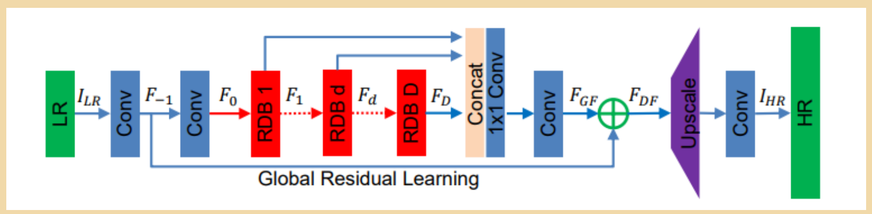
組員姓名：廖威竑、郭柏辰　　Mentor: 鄭揚翰、李兆鈁

摘要

在機器學習中，因為計算方式有大量重複處理的卷積運算(Convolution)，因為CPU不擅長處理大量平行運算，在此不是理想的運算單元，因而有使用GPU、FPGA加速運算的應用。由於GPU和FPGA的架構中都具備多顆核心，儘管單一核心性能不比CPU，卻可以利用其平行運算之特性，使運算速度提升。考量到功耗問題，FPGA可以同時實現低功耗且加快驗證速度，此專題中我們選用FPGA做為加速器。在此專題中，我們結合深度學習的方法，針對圖片進行超高解析(Image Super Resolution)，藉由輸入低解析度(Low Resolution)的圖片，經FPGA中的深度學習模型運算後，得到高解析度(High Resolution)的圖片。並比較「經FPGA運算」與「全部經CPU運算」之所需時間，並驗證兩張圖片的一致性，達成加速之成效。所使用之模型為Residual Dense Network (RDN)，藉由Vivado、DNNDK、PetaLinux等工具將模型轉化成FPGA可應用之形式，並且利用軟體語言，如C++、Python，控制FPGA板之運作。

Residual Dense Network (RDN)模型

理論上較多層的網路模型的訓練效果應該至少與較少層的網路模型相同，但實驗結果並非如此，模型若只是單純堆疊層數反而會使驗證結果變差，會有退化問題(degradation problem) [1] [2]。使用殘差(Residual)網路，在每經過幾層就給予一個捷徑連結(Shortcut connections) 確保較深網路訓練結果不會比較淺層網路差。在Residual Dense Blocks (RDB)中，每一層feature map皆會與其後方每一層feature map連接，傳統上L層的網路架構只會存在L條線連接，但在此架構模型之下，可以產生L (L+1)/2 條線，在文中也提到此作法可以有效減少所需要訓練的變數、減緩梯度消失(vanishing-gradient)以及增加變數重複使用率[2]



分析方法 (PSNR、SSIM)

PSNR (Peak Signal and Noise Ratio)分析待測圖片與參考圖片之間受雜訊干擾的比率，單位為dB。方法是將兩張圖片的每個像素點做相減的總和，並與圖片的最大值比較。

$$PSNR = 10\log(\frac{MAX_I^2}{MSE}) = 10\log\left(\frac{255^2}{\frac{1}{N}\sum_{i=1}^{N}\left(I(i)-\hat{I}(i)\right)^2}\right)(dB)$$

$I(i)$： pixels of high resolution , $\hat{I}(i)$：pixels of super resolution

SSIM(Structural Similarity Index)則是比較待測圖片與參考圖片之間的平均、標準差與共變異數，藉由較整體性的比較分析兩張圖片在結構上產生的差異。

$$SSIM(I,\hat{I}) = [C_l(I,\hat{I})]^{\alpha}[C_c(I,\hat{I})]^{\beta}[C_s(I,\hat{I})]^{\gamma}$$

$C_l = \frac{2\mu_I\mu_{\hat{I}}+C}{\mu_I^2+\mu_{\hat{I}}^2+C}$ (mean), $C_c = \frac{2\sigma_I\sigma_{\hat{I}}+C}{\sigma_I^2+\sigma_{\hat{I}}^2+C}$(standard deviation), $C_s = \frac{\sigma_{I\hat{I}}+C}{\sigma_I\sigma_{\hat{I}}+C}$ (covariance)

α, β, γ : parameters for controlling importance.

系統方法與設計

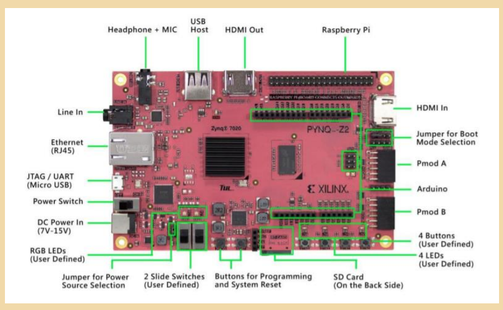
此次使用的FPGA版本為Xilinx TUL PYNQ™-Z2 board。其內部核心晶片型號為ZYNQ XC7Z020-1CLG400C，是一塊以Python + Zynq為概念的開發板。板子上可以分為PS和PL兩個部分。

由PL端進行DPU的架設以及載入與運作模型：

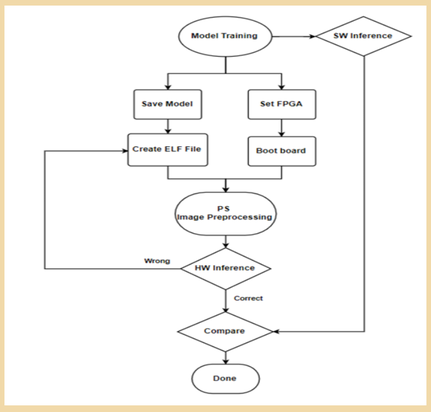
1. Freeze Model：載入相應的model以及訓練完的weight變數轉存成常數以便進行量化。
2. Quantization & Compile ：將訓練所得之權重 (weights) 由32 bits浮點數轉換為8 bits整數儲存並編譯成FPGA板可以讀取之格式。
3. Implement DPU on FPGA：載入DPU至PL端並整合相應到PS端。

由PS端控制圖片I/O與預處理與後處理：

1. Co-Compile：以CPU做圖片先後處理以及控制PL端之模型使用。
2. Test：進行PSNR、SSIM分析以及計時。



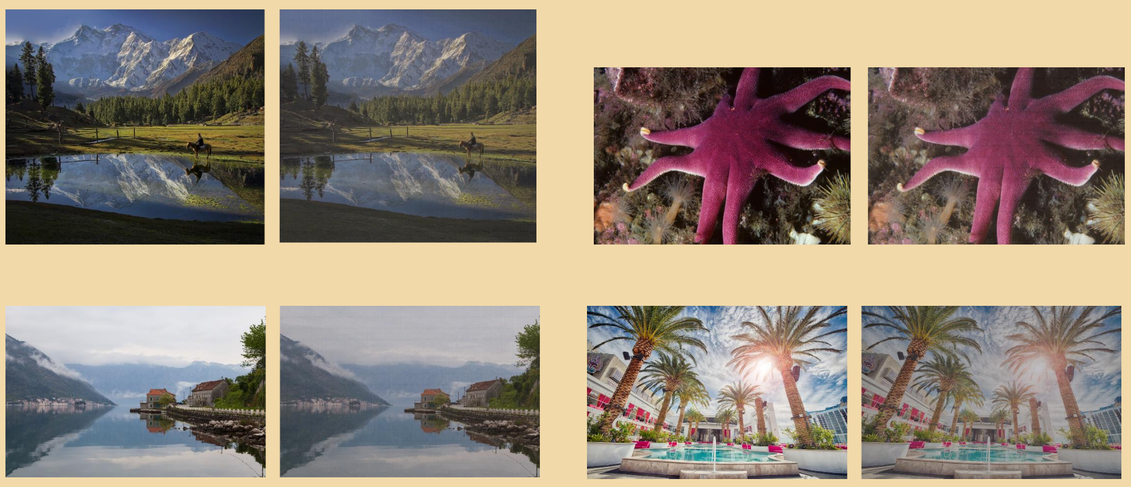
PYNQ-Z2 FPGA board



Flow chart

實驗成果

四組不同圖片經過FPGA運算之後結果，(左)使用DIV2K原高解析圖片、(右)利用DIV2K提供使用Bicubic的方法做2x downscaling的低解析度圖片，經過FPGA運算後的SR圖片。



由下表比較使用FPGA資源做運算與單純用CPU之速度比較。
其實驗結果為FPGA的 DPU相較於單純使用CPU有良好的加速效果。

Photo	FPGA Time(s)	Intel(R) Xeon(R) CPU @ 2.20GHz Time(s)	Speedup
0001	176.71	531.99	66.78%
0002	242.84	672.90	63.91%
0003	176.72	502.95	64.86%
0004	176.60	492.07	64.11%
0005	209.83	596.01	64.79%
0006	176.62	505.63	65.07%
0007	176.65	506.02	65.09%
0008	176.68	505.47	65.05%
0009	198.77	567.01	64.94%
0010	209.84	609.43	65.57%

由下表針對運算過後的SR圖片進行PSNR、SSIM分析。PSNR運算的結果皆在27~28之間，代表峰值訊號與雜訊約差1000倍，人眼判斷上並沒有看到雜訊的干擾。在結構分析(SSIM)上則數值相差較大，其原因推測為整體亮度經運算後變暗，與原圖相差較多；另外在patches連接出仍有細微的黑色條紋，也是造成SSIM數值較低的原因之一。

Photo	PSNR(dB)	SSIM
0001	28.066	0.672
0002	28.346	0.584
0003	27.833	0.694
0004	27.976	0.765
0005	28.588	0.801
0006	28.178	0.695
0007	28.109	0.590
0008	27.978	0.671
0009	27.867	0.670
0010	28.439	0.708

結論

1. 利用FPGA上的DPU進行圖片的運算，可以將運算速度提升約60%。對於單一patch而言，CPU因為擁有較高的操作頻率，速度高於DPU運算，因此較小張的圖片加速成效較不明顯；對於較大型的圖片而言，FPGA加速效果顯著，CPU則因為無法進行有效的平行運算，運算速度大幅上升。
2. 圖片在經過運算後沒有產生過多雜訊，但在結構性的問題上產生誤差。經過FPGA運算之圖片會產生整體亮度較暗的問題；在patches的連接處有相接時的黑色線條。最終成像品質的問題。

參考資料

[1] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770-778, June 2016.

[2] G. Huang, Z. Liu, L. V. D. Maaten and K. Q. Weinberger, "Densely Connected Convolutional Networks," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. pp. 2261-2269, 2017.

[3] E. Agustsson and R. Timofte, "NTIRE 2017 Challenge on Single Image Super-Resolution: Dataset and Study," in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2017.