中原大學

資訊工程學系

109 學年度專題實驗報告

Exploration of Dataflow and Systolic Array Configuration on 3D-CNNs

組員

資訊工程學系四年甲班 1067147 楊于姍

資訊工程學系四年乙班 1067220 白欣怡

指導教授 鄭維凱

2020/12/4

摘要

專題中我們研究了論文「SCALE-Sim: Systolic CNN Accelerator Simulator」提出的 2D CNN 架構效能檢測工具 ScaleSim，及 ScaleSim 中使用的一種減少記憶體訪問的運算方式 Systolic array，以及 OS(output stationary)、IS(input stationary)、WS(weight stationary)三種 dataflow。以 ScaleSim 為基礎，並融合了論文「Systolic Cube: A Spatial 3D CNN Accelerator Architecture for Low Power Video Analysis」提出的加入時間維度的三維 Systolic cube 架構，以此架構實作出一套檢測 3D CNN 架構效能的工具。我們目前將以 OS dataflow 為主，來探索各個 3D CNN 架構之效能，並分析各層卷積層適合的 PE（process element）架構，將來會嘗試以 IS 及 WS dataflow 來探索 3D CNN 架構。我們目前使用的 3D CNN 為論「Learning Spatiotemporal Features with 3D Convolutional Networks」提出的 C3D 架構及論文「3D Convolutional Neural Networks for Human Action Recognition」提出的 3D CNN 架構來進行架構探索，實驗方式分別為調整三維的 PE array size 及 Buffer size，得到運行結果後我們會進一步分析各層的 PE 和 Buffers 的使用率來找出適合該 3D CNN 架構的 PE size 和 Buffer size，避免硬體空間的浪費。

目錄

圖目錄

第一章、相關研究

1.1 3D-CNN 之相關研究

3D-CNN 組成架構與 2D-CNN 差異不大，皆有卷積層做卷積運算，Subsample 層做 Pooling 降維運作，而 3D-CNN 主要特徵為可以捕捉時空特徵，代表 3D-CNN 可以捕捉連續禎上的時間訊息，換句話說 3D-CNN 可以藉由輸入多張連續圖像來捕捉連續動作，去進一步分析與預測，以 Fig.1 和 Fig.2 為例，在 2D-CNN 中輸入 Fig.1，2D-CNN 可以判斷出一個人穿著粉色衣服，這個人的手腳向前伸出，在 3D-CNN 中以 Fig.2 中的連續圖片當作輸入，3D-CNN 可以判斷出一個人穿著粉色衣服，這個人在跳體操。

Fig. 1: 2D-CNN 輸入為一張圖片[3]

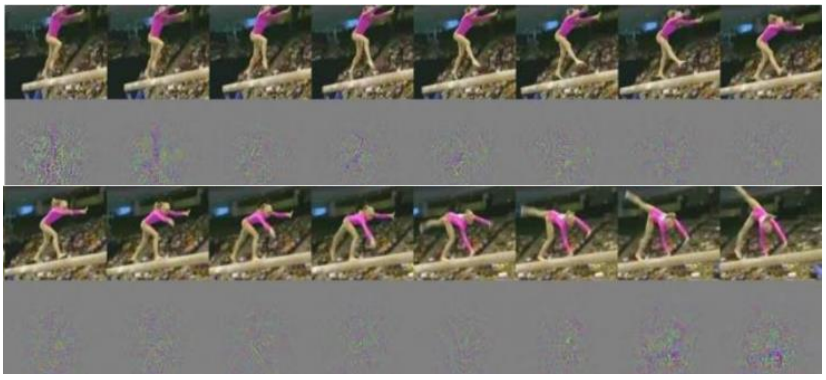

Fig. 2: 3D-CNN 輸入為連續多張圖片[3]

1.2 2D-CNN 與 3D-CNN 之間的差異

A. 2D-CNN 和 3D-CNN 優缺點與應用比較

Table1 簡述 2D-CNN 和 3D-CNN 優缺點與應用比較表

	2D-CNN	3D-CNN
優點	計算較簡單、資料量較少	可以捕捉時空特徵
缺點	無法捕捉時空特徵(連續動作)	計算較複雜、資料量較多
應用	物體識別與偵測	動作識別與物體追蹤

B. 2D-CNN 和 3D-CNN 架構運作比較

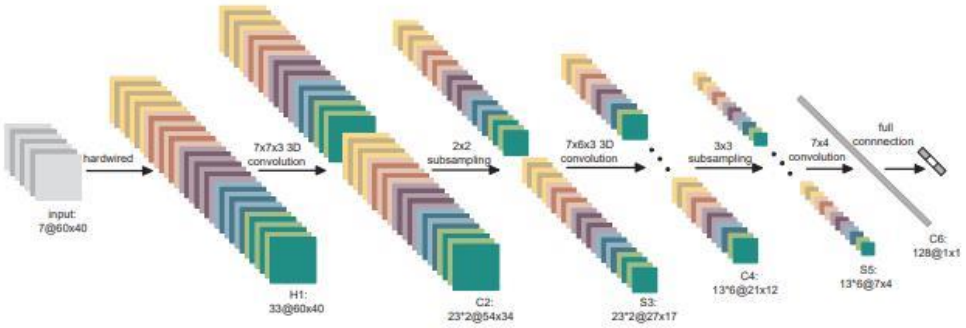

Fig. 3: 3DCNN 架構[4]

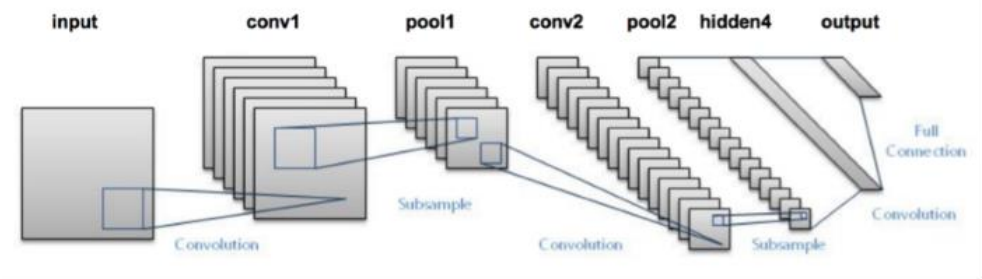

Fig. 4: 2D Convolutional Neural Networks 架構圖,圖片來源: Visualizing parts of Convolutional Neural Networks using Keras and Cats

3D-CNN 架構如圖 Fig.3 經由 hardwired 層將輸入分為五個通道訊息，分別為光流 x、光流 y、斜度 x、斜度 y 和灰度，灰度為使輸入由彩色變成灰階，以綠色立方為例，60*40*6 綠色立方輸入至第一層卷積層，與兩個 7*7*3 卷積核卷積運算後輸出兩個 54*34*4 綠色立方，經 POOLING 層後由 27*17*4 輸入至第二層卷積層與三個 7*6*3 卷積核卷積運算後輸出六個 21*12*2 綠色立方以此類推，而 2D-CNN 架構如圖 Fig.4 皆由二維輸入與二維卷積核卷積運算後輸出二維輸出。兩者除了在輸入與卷積核皆由二維變成三維架構外，3D-CNN 經由卷積核一次卷積運算後的輸出在整個 Feature map 上表示為一個三維架構，而 2D-CNN 經由卷

積核一次卷積運算後的輸出在整個 Feature map 上表示為一個 pixel，簡而言之一次卷積運算後的輸出也會由二維變成三維架構。

1.3 SCALE-Sim Simulator 之相關研究

在論文[1]提出的一套 2D CNN 效能檢測工具，模擬 2D CNN 運算時，PE(process element) 的使用狀況。其中運用了 2D systolic array 的技術並加入了 Buffer 在各個 PE 中，這個技術可以減少記憶體訪問的次數進而達到速度提升的功用，我們將擴充此架構來達到 3D CNN 架構測試效能的效果。

1.4 Weight、Ìnput、Òutput Stationary Dataflow 之相關研究

Dataflow 是資料的處理方式，影響 CNN 是否能有效率地完成運算，若 Dataflow 無法有效率地運行，就算 PE 大小設定很大，硬體資源使用率依然會很低。現今的 Dataflow 皆在強調 Data Reuse，讓同一筆數據盡量在 On-Chip 端重複利用，避免重複去外部的 DRAM 讀取或最小化 Partial Sum (Psum)，減少寫回 DRAM 的次數進而優化數據傳輸。

常見的 Dataflow[6]可以分為以下幾種:

1. Weight Stationary(WS)

如 Fig.5 所示，WS 是使 Weight 最大化重複使用，通過將 Weight 固定在 PE，減少從 Buffer 讀取 Weight 的次數和 Weight 的移動次數，Ifmap(Act)從 Buffer 讀取，進入 PE 與 Weight 做運算，Psum 則從 Buffer 讀取並與各 PE 的 Register 累加後寫回 Buffer。

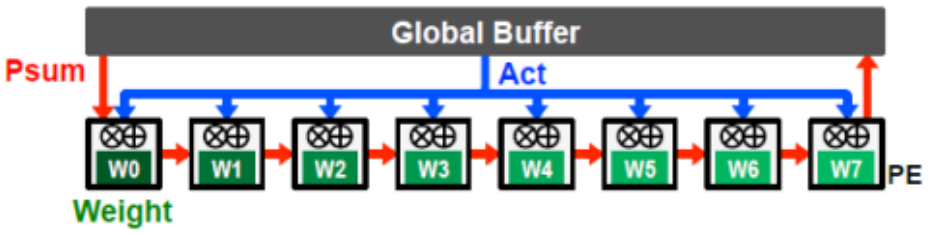

Fig. 5: Weight Stationary(WS)[5]

2. Input Stationary(IS)

IS 是使 Ifmap 最大化重複使用，與 Weight Stationary(WS)相似，唯一差別在原本是 Weight 固定在 PE，Ifmap 和 Psum 從 Buffer 讀取並寫回，改為 Ifmap 固定在 PE，Ifmap 和 Psum 從 Buffer 讀取並寫回。

3. Output Stationary(OS)

如 Fig.6 所示，OS 是將 Psum 最大化重複使用，改為 Psum 固定在各個 PE 中並與各 PE Register 進行累加, Weight 和 Ifmap 則從 Buffer 讀取至各個 PE 進行運算。

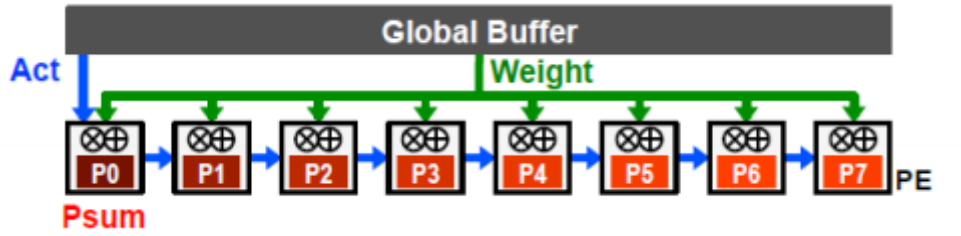

Fig. 6: Output Stationary(OS)[5]

OS 又可細分為三種，如 Fig.7 所示:

A. OS_A

單一 Output Channel 與多 Output Activations 進行運算, 固定一個 Filter,和 Ifmap 全部算完,才換下一個 Filter。

B. OS_B

多 Output Channels 與多 Output Activations 進行運算, 多個 Filter 和多個 Ifmap 計算,SCALE-Sim 的 OS Dataflow 屬於這一類型。

C. OS_C

多 Output Channels 與單一 Output Activations 進行運算, 同一批 Ifmap 與全部 Filter 算完，才換下一批 Ifmap。

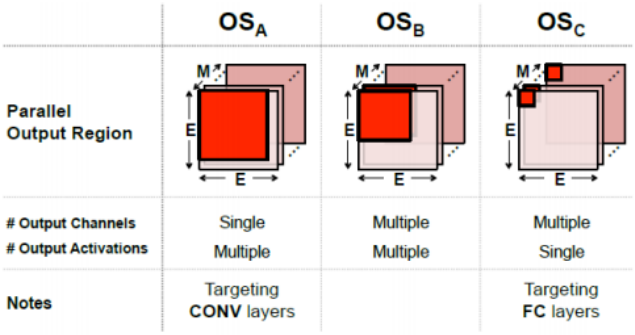

Fig. 7: Variations of Output Stationary [5]

第二章、研究動機

隨著影片資源日益漸增，隨之而來的問題是如何有效捕捉時序上的訊息，由上述 1.2 小節可知由於 2D-CNN 無法有效捕捉時序上的訊息，所以基於 3D-CNN 的需求會逐漸增加。

在尋找適合自己資料的神經網路模型架構中，我們除了注重神經網路的準確率外，同時也希望能夠找到其中效能最高的神經網路架構，準確率計算是可以通過數據對比驗證的，而效能的驗證卻不是那麼簡單的事情，畢竟 CNN 是一個黑盒子，過程中 Buffer 與 PE 使用情況是未知的，所以我們想要做出一個可以驗證 3D-CNN 效能的檢測工具。

第三章、研究方法

3.1 將原 SCALE-Sim Simulator 2D Dataflow 架構改為 3D 架構

我們使用的 dataflow 為 Systolic Cube Dataflow，Dataflow 說明分為 Input(輸入)和 Weight(卷積核)兩部份。

A. 3D Systolic Cube Input Dataflow

Input 皆由 3D Systolic Cube 最上層第一行進入，根據 Weight Channel 決定各批 input 在同個 PE 內停留次數，當 input 停留數滿或最上層第一行有空的 PE 時，就會進行資料傳遞或進新一批 input，資料傳遞是由左至右、前往後、上至下傳遞，如圖 Fig.9，由於資料是由左至右傳遞所以當新進 input 次數等於 3D Systolic Cube row 數時，之後的 input 皆由最上層第一行最後一個 PE 進入。

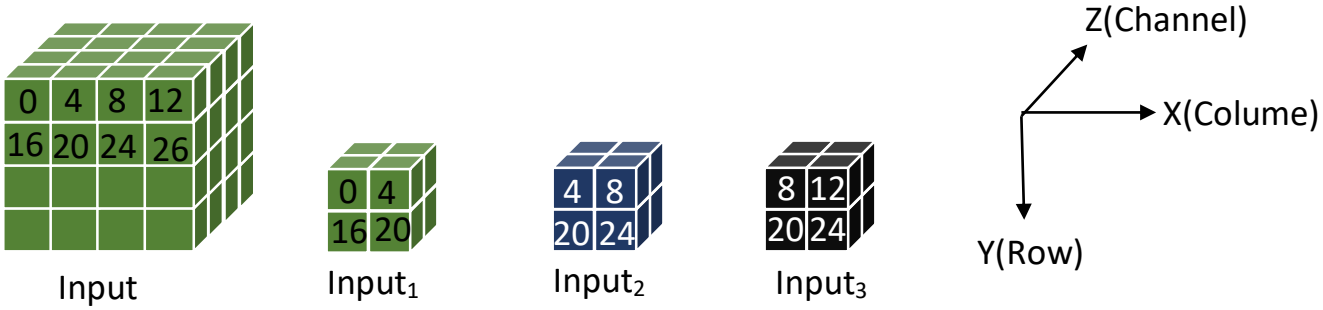

Fig. 8: input size 為 4*4*4，$Input_n$ size 根據 weight size 決定，n∈$\mathcal{N}$，這裡假設 weight size 為 2*2*2

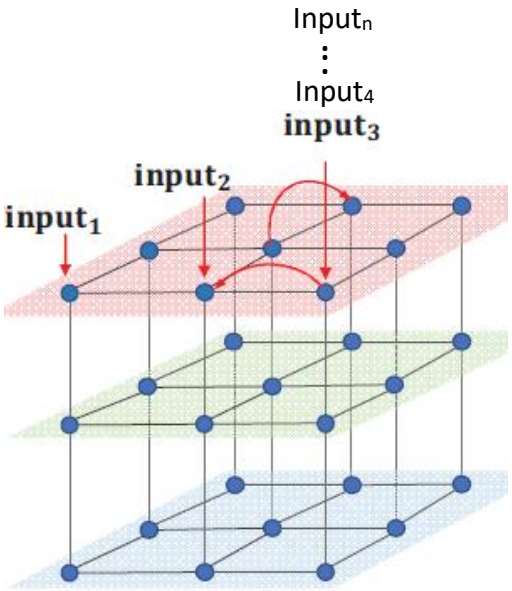

Fig. 9: 3D Systolic Cube Input Dataflow[2]

B. 3D Systolic Cube Weight Dataflow

一層各有一個 weight，各層 weight 從各層最左下角的 PE 進入 3D Systolic Cube，weight 在 PE 間傳遞為右至左和前至後，如圖 Fig.10，weight 會一直重複進入 PE 直到 input 計算完畢，若是有其他組 weight 需要計算就接續前一組 weight 進入 PE 中。

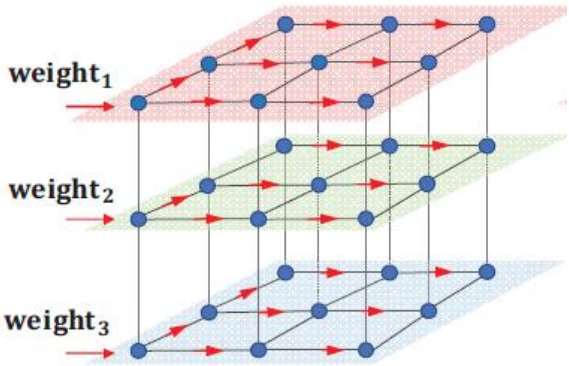

Fig. 10: 3D Systolic Cube Weight Dataflow[2]

3.2 透過更改 Buffer 與 PE 設定探索 3D CNN 之間的效能

更改 Net 設定與 config 設定檔，在使用多個不同 3D CNN 情況下，分別根據不同大小的 Buffer 與不同大小的 PE 設定，透過數據比較找出最適合此 Net 架構的 Buffer 和 PE 大小。

第四章、實驗結果

4.1 實驗設置

使用論文[1][2]提出的 3DCNN 架構 Net，如 Fig.3，及 C3D Net，如 Fig.11 做為測試對象，比較兩個 Net 在不同 Buffer 和不同 PE 設置時，兩者對 Buffer 和 PE 的使用率，Buffer 分為 Ifmap、Filter、Ofmap 三種 Buffer，分別存輸入、卷積核、輸出的在 Dram 內的 Byte 數。

A. 3D CNN Net

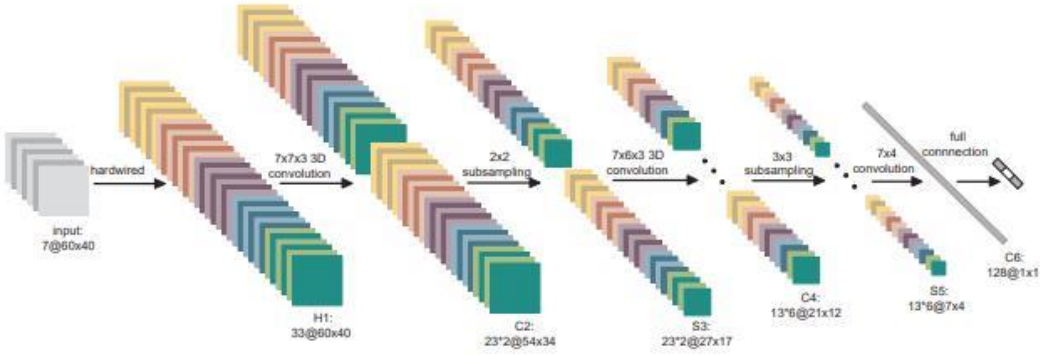

Fig.3 : 3DCNN 架構[4]

B. C3D Net

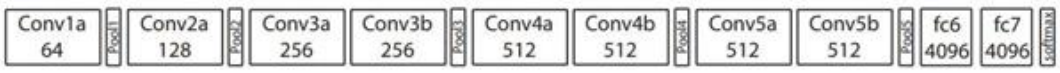

Fig. 11: 3D-CNN 架構[3]，共 8 層卷積層，2 層全連接層，各卷積層 Filter 數量分別為 64、128、256、512、512、512、512，皆由 2*2*2 做 subsampling，只有最後一層由 1*2*2 做 subsampling

4.2 實驗結果與分析

A.PE 大小探索

三種 Buffer 大小固定為 Ìfmap Buffer=512，Filter Buffer=512，Ofmap Buffer=256，PE 大小分為 16*16*16、32*32*32、32*16*16、16*32*16、16*16*32，分別用 3DCNN Net 和 C3D Net 做比較。

Table2: 3DCNN PE size:16*16*16

	average utilization(%)	cycles	DRAM IFMAP READ BW(Bytes)	DRAM Filter READ BW(Bytes)	DRAM OUTPUT Write BW(Bytes)
conv1	99.5561	590659	2400	147	18360
conv1a	99.5561	590659	2400	147	18360
conv1b	99.5561	590659	2400	147	18360
conv1c	99.4495	404855	2400	147	14688
conv1d	99.4495	404855	2400	147	14688
conv2	96.3428	30246	396	126	2268
conv2a	96.3428	30246	396	126	2268
conv2b	96.3428	30246	396	126	2268
conv2c	94.698	16138	270	126	1512
conv2d	94.698	16138	270	126	1512
conv2e	96.3428	30246	396	126	2268
conv2f	96.3428	30246	396	126	2268
conv2g	96.3428	30246	396	126	2268
conv2h	94.698	16138	270	126	1512
conv2i	94.698	16138	270	126	1512
conv3	10.88	10934	140	28	78

Table3: 3DCNN PE size:32*32*32

	average utilization(%)	cycles	DRAM IFMAP READ BW(Bytes)	DRAM Filter READ BW(Bytes)	DRAM OUTPUT Write BW(Bytes)
conv1	96.3446	295352	920	147	18360
conv1a	96.3446	295352	920	147	18360
conv1b	96.3446	295352	920	147	18360
conv1c	95.6236	202877	765	147	14688
conv1d	95.6236	202877	765	147	14688
conv2	80.8806	15142	154	126	2268
conv2a	80.8806	15142	154	126	2268
conv2b	80.8806	15142	154	126	2268
conv2c	75.3911	8090	115	126	1512
conv2d	75.3911	8090	115	126	1512

conv2e	80.8806	15142	154	126	2268
conv2f	80.8806	15142	154	126	2268
conv2g	80.8806	15142	154	126	2268
conv2h	75.3911	8090	115	126	1512
conv2i	75.3911	8090	115	126	1512
conv3	4.6623	6566	84	28	78

Table4: 3DCNN PE size:32*16*16

	average utilization	cycles	DRAM IFMAP READ BW	DRAM Filter READ BW	DRAM OUTPUT Write BW
conv1	98.7301	295352	1797	147	18360
conv1a	98.7301	295352	1797	147	18360
conv1b	98.7301	295352	1797	147	18360
conv1c	98.4305	202877	1452	147	14688
conv1d	98.4305	202877	1452	147	14688
conv2	91.2944	15142	238	126	2268
conv2a	91.2944	15142	238	126	2268
conv2b	91.2944	15142	238	126	2268
conv2c	87.3041	8090	160	126	1512
conv2d	87.3041	8090	160	126	1512
conv2e	91.2944	15142	238	126	2268
conv2f	91.2944	15142	238	126	2268
conv2g	91.2944	15142	238	126	2268
conv2h	87.3041	8090	160	126	1512
conv2i	87.3041	8090	160	126	1512
conv3	8.301	6566	140	28	78

Table5: 3DCNN PE size:16*32*16

	average utilization(%)	cycles	DRAM IFMAP READ BW(Bytes)	DRAM Filter READ BW(Bytes)	DRAM OUTPUT Write BW(Bytes)
conv1	52.8875	590659	2400	147	18360
conv1a	52.8875	590659	2400	147	18360
conv1b	52.8875	590659	2400	147	18360
conv1c	52.8305	404855	2400	147	14688
conv1d	52.8305	404855	2400	147	14688

conv2	51.1604	30246	396	126	2268
conv2a	51.1604	30246	396	126	2268
conv2b	51.1604	30246	396	126	2268
conv2c	50.2769	16138	270	126	1512
conv2d	50.2769	16138	270	126	1512
conv2e	51.1604	30246	396	126	2268
conv2f	51.1604	30246	396	126	2268
conv2g	51.1604	30246	396	126	2268
conv2h	50.2769	16138	270	126	1512
conv2i	50.2769	16138	270	126	1512
conv3	5.44	10934	140	28	78

Table6: 3DCNN PE size:16*16*32

	average utilization(%)	cycles	DRAM IFMAP READ BW(Bytes)	DRAM Filter READ BW(Bytes)	DRAM OUTPUT Write BW(Bytes)
conv1	97.2717	590659	1782	147	18360
conv1a	97.2717	590659	1782	147	18360
conv1b	97.2717	590659	1782	147	18360
conv1c	96.6597	404855	1438	147	14688
conv1d	96.6597	404855	1438	147	14688
conv2	85.7023	30246	225	126	2268
conv2a	85.7023	30246	225	126	2268
conv2b	85.7023	30246	225	126	2268
conv2c	80.9463	16138	145	126	1512
conv2d	80.9463	16138	145	126	1512
conv2e	85.7023	30246	225	126	2268
conv2f	85.7023	30246	225	126	2268
conv2g	85.7023	30246	225	126	2268
conv2h	80.9463	16138	145	126	1512
conv2i	80.9463	16138	145	126	1512
conv3	9.4527	10934	84	28	78

Table7: C3D PE size:16*16*16

	average utilization(%)	cycles	DRAM IFMAP READ BW(Bytes)	DRAM Filter READ BW(Bytes)	DRAM OUTPUT Write BW(Bytes)
conv1a	99.9979	32197834	21888	1.08E+02	19079424
conv2a	99.9955	15126862	5292	2.16E+02	8963584
conv3a	99.9764	1416878	1230	432	1677312
conv3b	99.9732	1247478	1092	432	1477632
conv4a	99.7813	75598	218	864	180224
conv4b	99.6961	54982	161	864	129024
conv5a	93.7359	1774	25	864	0
conv5b	96.3651	1774	10	565	0

Table8: C3D PE size:32*16*16

	average utilization(%)	cycles	DRAM IFMAP READ BW(Bytes)	DRAM Filter READ BW(Bytes)	DRAM OUTPUT Write BW(Bytes)
conv1a	99.9938	16099786	21888	108	19079424
conv2a	99.9869	7565134	5292	216	8963584
conv3a	99.9302	708174	1230	432	1077312
conv3b	99.9209	625398	1092	432	1477632
conv4a	99.359	37102	132	864	180224
conv4b	99.1109	26854	104	864	129024
conv5a	96.6372	1774	28	826	0
conv5b	93.2372	1774	10	397	0

Table9: C3D PE size:16*32*16

	average utilization(%)	cycles	DRAM IFMAP READ BW(Bytes)	DRAM Filter READ BW(Bytes)	DRAM OUTPUT Write BW(Bytes)
conv1a	74.9987	16098944	21888	5.40E+01	19079424
conv2a	62.4998	7563466	5292	1.08E+02	8963584
conv3a	56.2689	708470	1230	216	1677312
conv3b	56.2702	623774	1092	216	1477632

conv4a	53.5985	38197	234	432	180224
conv4b	53.7685	27526	176	432	129024
conv5a	66.5885	1333	28	432	0
conv5b	89.1071	910	10	432	0

Table10: C3D PE size:16*16*32

	average utilization(%)	cycles	DRAM IFMAP READ BW(Bytes)	DRAM Filter READ BW(Bytes)	DRAM OUTPUT Write BW(Bytes)
conv1a	99.9827	32197834	21888	108	19079424
conv2a	99.9632	15126862	5292	216	8963584
conv3a	99.8044	1416878	1230	432	1677312
conv3b	99.7781	1247478	1092	432	1477632
conv4a	98.2367	75598	119	864	180224
conv4b	97.6041	54982	90	864	129024
conv5a	76.434	1774	23	864	0
conv5b	96.5014	1774	10	864	0

Table11: C3D PE size:32*32*32

	average utilization(%)	cycles	DRAM IFMAP READ BW(Bytes)	DRAM Filter READ BW(Bytes)	DRAM OUTPUT Write BW(Bytes)
conv1a	99.9829	8049920	21888	54	19079424
conv2a	99.9637	3782602	5292	108	8963584
conv3a	99.8068	354134	1230	216	1677312
conv3b	99.7809	312734	1092	216	1477632
conv4a	98.2579	18606	135	432	180224
conv4b	97.6037	13478	107	432	129024
conv5a	81.1059	910	28	432	0
conv5b	89.3592	910	10	383	0

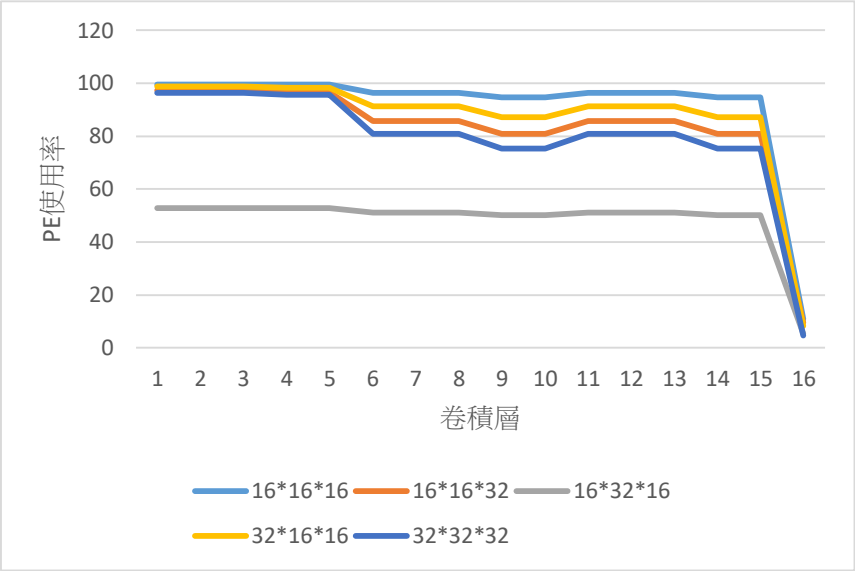

Fig. 12: 3DCNN 卷積層 PE 使用率比較

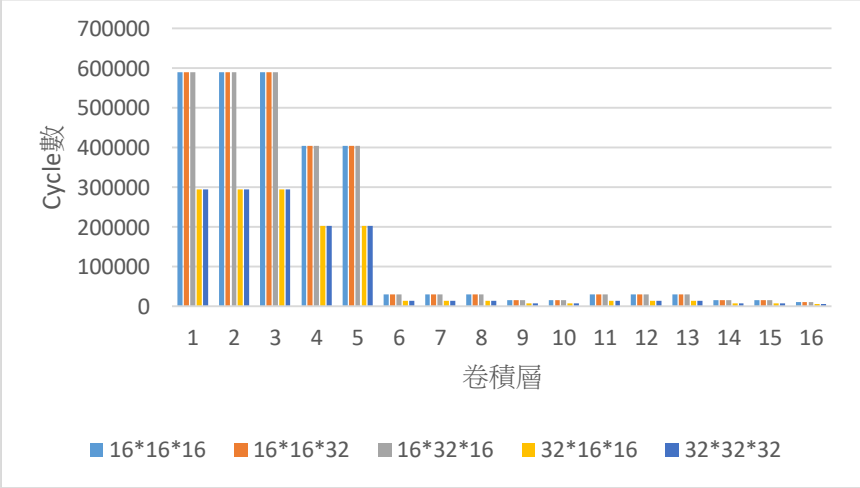

Fig. 13: 3DCNN Cycle 數比較

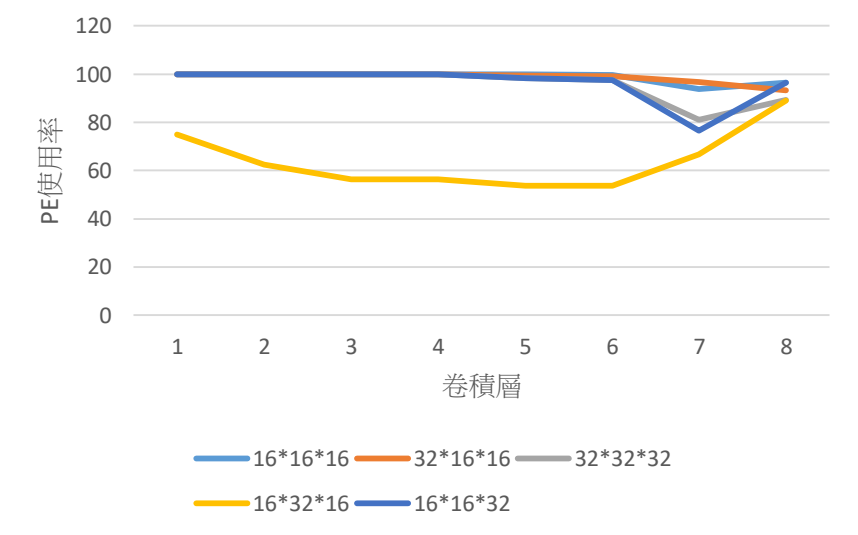

Fig. 14: C3D 卷積層 PE 使用率比較

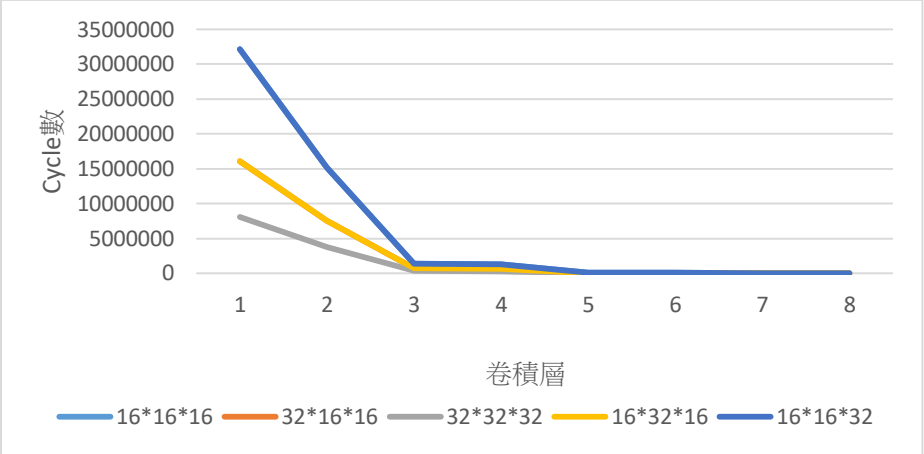

Fig. 15: C3D Cycle 數比較

Cycle 數比較

當 Filter 數大於 PE 的 Row 數時，如 Fig.15，決定 Cycle 兩大關鍵是 PE 的 Row 數和 Column 數，PE 的 Row 數決定一次可以放多少個 Filter，而當 PE 的 Column 數越大，代表可以連續放入越多的 Input，所以 PE 的 Row 數和 Column 數越大，Cycle 數會越小，而 PE 的 Row 數對 Cycle 數影響較 Column 數大，以 Fig.13 16*16*16 與 16*16*32 為例，因兩者的 PE 的 Column 數和 Row 數一樣，所以 Cycle 數不變，以 16*16*16 和 16*32*16 為例，16*16*16 Cycle 數較大，因為 PE 的 Row 數較 16*32*16 PE 的 Row 數小，當 Filter 數小於等於 PE 的 Row 數時，如 Fig.13，決定 Cycle 關鍵只有 PE 的 Column 數，因為 DataFlow 是一層一個 Filter 獨立運作，所以在 Filter 數小於等於 PE 的 Row 數時，使用到的 PE 的 Row 數是一樣的。

PE 使用率比較

1. 卷積核數量對 PE 使用率的影響

由 Fig.12 可知 3DCNN 到最後一層卷積層時，使用率大幅下降，這是因為 3DCNN 最後一層卷積層的 Filter 數只有一個，而因為 Systolic Cube Filter Dataflow 是各層獨立運作，所以在最後一層卷積層時，只會使用到最上層的 PE，導致最後一層卷積層時，使用率大幅下降，所以由此得知卷積核數量會影響 PE 的使用率。

2. 每層的 PE 數量對 PE 使用率的影響

由 Fig.12 和 Fig.14 看出 16*32*16 使用率皆是最低的，這是因為總輸入是固定，而我們每層的 PE 數量會影響總 Cycle 數，每層的 PE 數量越多，總 Cycle 數越少，反之總 Cycle 數越多。加上我們使用率的計算是用每次 Cycle 的 PE 比率去做加總再平均，例:32*32*32 和 16*32*16 在每次 Cycle 的未使用的 PE 比率是一樣的，但是因為 32*32*32 的 Cycle 數會較少，所以未使用的 PE 數計算次數也會比較少，導致 16*32*16 使用率較低，再加上上述所說的 Systolic Cube Filter

Dataflow 是各層獨立運作，又因為 16*32*16 相較 16*16*16 Row 數較多，導致使用率較低，所以 16*32*16 使用率最低。

B.Buffer 大小探索

Table12 : 3DCNN: PE=16*16*16、Ifmap Buffer=1、Filter Buffer=1、Output Buffer=256

	average utilization	cycles	DRAM IFMAP READ BW	DRAM Filter READ BW	DRAM OUTPUT Write BW
conv1	99.5561	590659	3575	147	18360
conv1a	99.5561	590659	3575	147	18360
conv1b	99.5561	590659	3575	147	18360
conv1c	99.4495	404855	2877	147	14688
conv1d	99.4495	404855	2877	147	14688
conv2	96.3428	30246	396	126	2268
conv2a	96.3428	30246	396	126	2268
conv2b	96.3428	30246	396	126	2268
conv2c	94.698	16138	270	126	1512
conv2d	94.698	16138	270	126	1512
conv2e	96.3428	30246	396	126	2268
conv2f	96.3428	30246	396	126	2268
conv2g	96.3428	30246	396	126	2268
conv2h	94.698	16138	270	126	1512
conv2i	94.698	16138	270	126	1512
conv3	10.88	10934	140	28	78

Table13 : 3DCNN: PE=32*16*16、Ifmap Buffer=1、Filter Buffer=2、Output Buffer=256

	average utilization	cycles	DRAM IFMAP READ BW	DRAM Filter READ BW	DRAM OUTPUT Write BW
conv1	98.7301	295352	1818	147	18360
conv1a	98.7301	295352	1818	147	18360
conv1b	98.7301	295352	1818	147	18360
conv1c	98.4305	202877	1473	147	144688
conv1d	98.4305	202877	1473	147	144688
conv2	91.2944	15142	238	126	2268
conv2a	91.2944	15142	238	126	2268
conv2b	91.2944	15142	238	126	2268
conv2c	87.3041	8090	160	126	1512
conv2d	87.3041	8090	160	126	1512

conv2e	91.2944	15142	238	126	2268
conv2f	91.2944	15142	238	126	2268
conv2g	91.2944	15142	238	126	2268
conv2h	87.3041	8090	160	126	1512
conv2i	87.3041	8090	160	126	1512
conv3	8.301	6566	140	28	78

Table14 : 3DCNN: PE=5*8*10、Ifmap Buffer=8、Filter Buffer=5、Output Buffer=256

	average utilization	cycles	DRAM IFMAP READ BW	DRAM Filter READ BW	DRAM OUTPUT Write BW
conv1	99.9577	1889250	2400	147	18360
conv1a	99.9577	1889250	2400	147	18360
conv1b	99.9577	1889250	2400	147	18360
conv1c	99.9471	1295663	2400	147	14688
conv1d	99.9471	1295663	2400	147	14688
conv2	99.6745	95763	459	126	2268
conv2a	99.6745	95763	459	126	2268
conv2b	99.6745	95763	459	126	2268
conv2c	99.5186	50910	459	126	1512
conv2d	99.5186	50910	459	126	1512
conv2e	99.6745	95763	459	126	2268
conv2f	99.6745	95763	459	126	2268
conv2g	99.6745	95763	459	126	2268
conv2h	99.5189	50910	459	126	1512
conv2i	99.5189	50910	459	126	1512
conv3	28.3179	34947	224	28	78

Ifmap Bytes

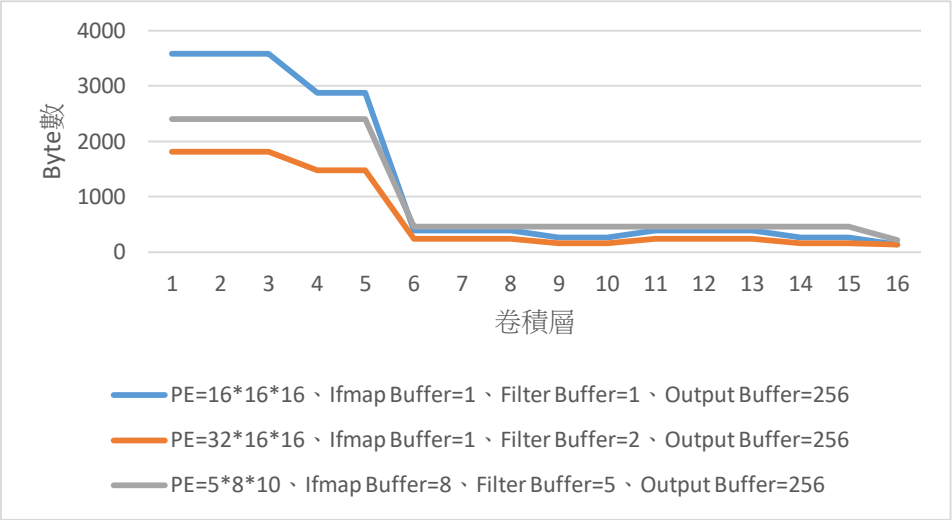

Fig. 16 3DCNN Ifmap Bytes 比較

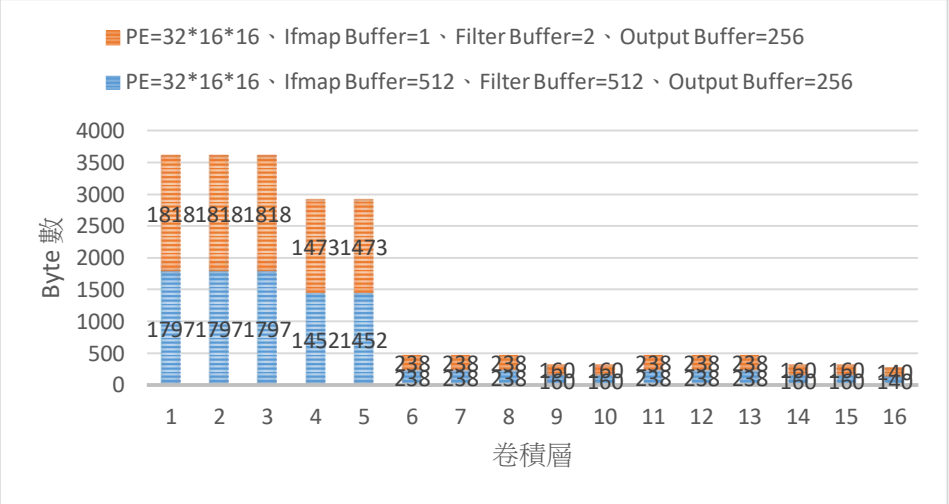

Fig. 17: 3DCNN 同 PE 架構下 Buffer Size 不同 Ifmap Bytes 比較

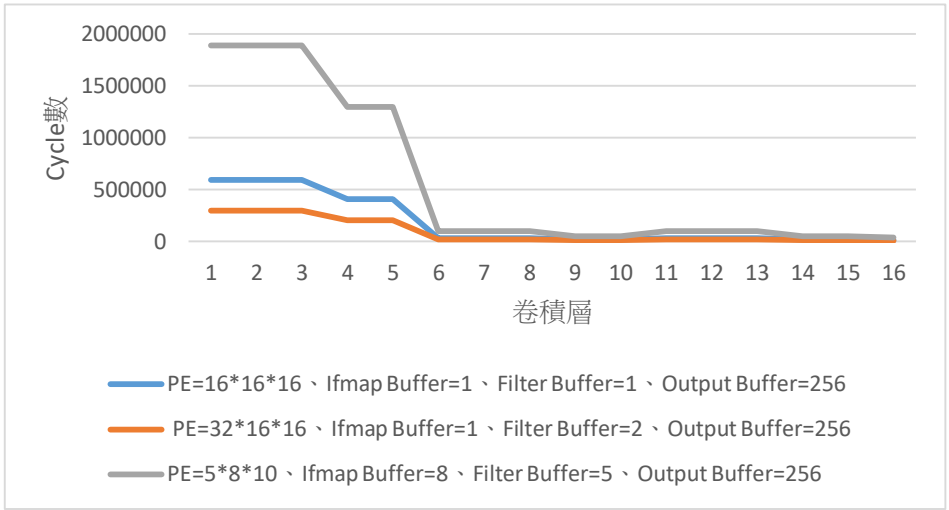

Fig. 18: 3DCNN 不同架構下 Cycle 數比較

1. Input 大小越小，Byte 數減少

由 Fig.16 看出隨著卷積次數越多 Byte 數減少，因為卷積次數越多代表輸入會越少，所以 Byte 數減少，以第 5 層至第 6 層卷積層 Byte 數下降幅度增加為例，因為在 3DCNN 架構中第 5 層至第 6 層經過 2*2 Subsampling 層後，導致 Input Colume 和 Row 由 60*4 變為 27*17，所以 Byte 數下降幅度增加。

2. Sram Buffer Size 越小 Byte 數增加

同架構下，若是 Sram Buffer 初始設定值越小，存入 Dram Buffer Byte 數則增加，如 Fig.17

OUTPUT Byte 數

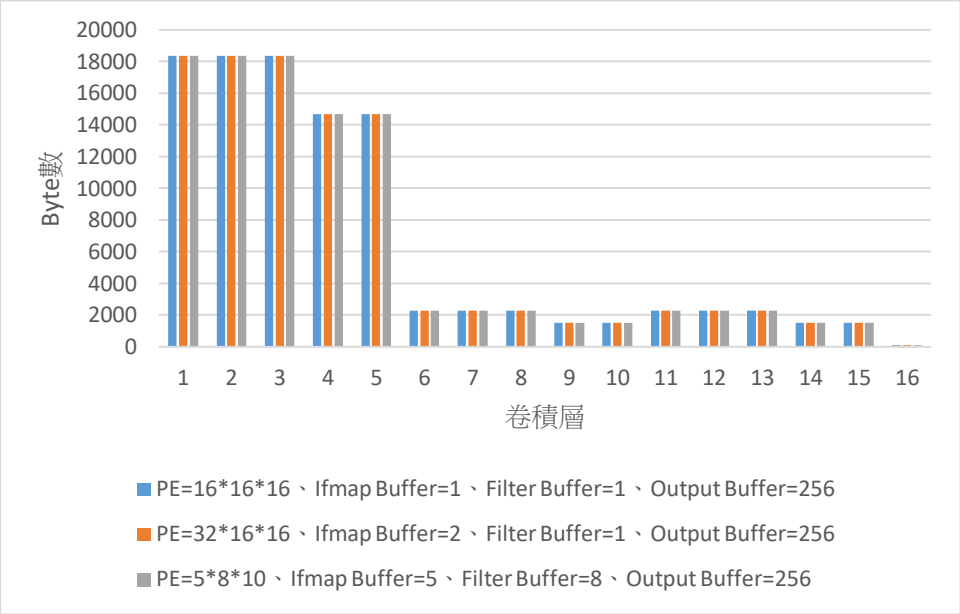

Fig. 19: 3DCNN OUTPUT Byte 數比較

同 Net OUTPUT Byte 數在各個卷積層是相同的，因為卷積後的 Feature map 大小同 Net 架構下是固定的，如 Fig.19

第五章、結論

1. PE 的 Row 數和 Column 越大，Cycle 數越小，反之越大，其中 Row 數對 Cycle 數影響較大
2. PE 的 Row 數越接近 Filter 數，PE 使用率越高，所以若是 3D-CNN 架構 Filter 數不互質時，當 PE 的 Row 數等於 3D-CNN 架構 Filter 數的最大公因數時，PE 使用率會提升，若是 3D-CNN 架構 Filter 數互質時，則 PE 的 Row 數等於 3D-CNN 架構 Filter 數最大值時，使用率會提升，如 Fig.20

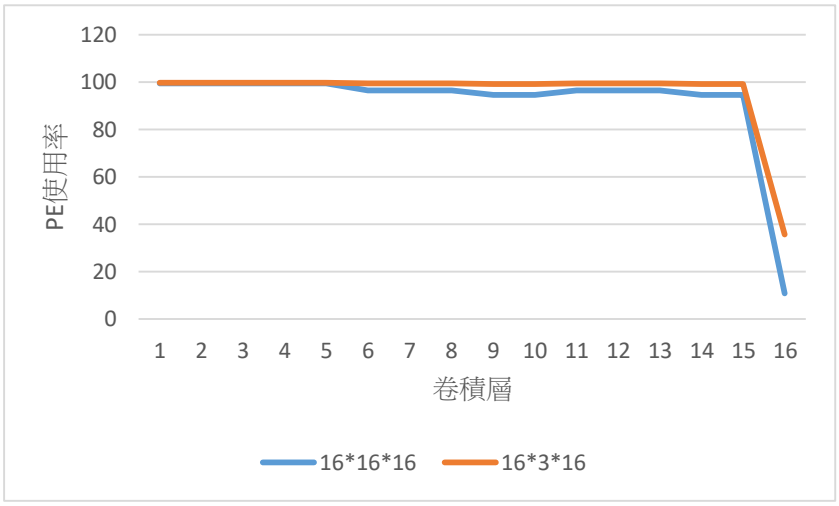

Fig. 20: 3DCNN PE 的 Row 數等於 Filter 數的最大公因數時，使用率比較

3. Input 大小越大 Sram Buffer Size 越小，Ifmap Byte 越多，而 ofmap Byte 數量依 3D-CNN 架構決定，產生的 Output Feature map 越大 ofmap Byte 數量越多

第六章、未來展望

目前我們使用的 dataflow 為 OS(output stationary)，將來還可以在 IS(input stationary) 及 WS(weight stationary)的 dataflow 基礎再做延伸探索。

參考文獻

[1] Ananda Samajdar, Yuhao Zhu, Paul Whatmough, Matthew Mattina, and Tushar Krishna, “SCALE-Sim: Systolic CNN Accelerator Simulator”, arXiv preprint arXiv:1811.02883,2018.

[2] Yongchen Wang, Ying Wangl, Huawei Li, Cong Shi, and Xiaowei Li, “Systolic Cube: A Spatial 3D CNN Accelerator Architecture for Low Power Video Analysis, ”

[3] Du Tranl, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri, “Learning Spatiotemporal Features with 3D Convolutional Networks”, arXiv:1412.0767v4, Oct 2015

[4] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu, “3D Convolutional Neural Networks for Human Action Recognition”, IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 35, NO. 1, JANUARY 2013

[5] Vivienne Sze, Senior Member, Yu-Hsin Chen, Student Member, Tien-Ju Yang, Student Member, Joel Emer, Fellow, “Efficient Processing of Deep Neural Networks: A Tutorial and Survey”, arXiv:1703.09039v2 [cs.CV], Aug 2017

[6]Y.-H. Chen, J. Emer, and V. Sze, “Using Dataflow to Optimize Energy Efficiency of Deep Neural Network Accelerators,” IEEE Micro’s Top Picks from the Computer Architecture Conferences, vol. 37, no. 3, May-June 2017.