Application of Neural Network Accelerator for Coffee Beans Detection

神經網路加速器應用於咖啡豆檢測

專題領域：系統組
組　　別：A287
指導教授：鄭桂忠 教授
組員姓名：謝承志、簡健翔、郭哲維

Abstract

傳統上，分辨發霉咖啡豆往往需要經驗與肉眼人工辨別，但其過程繁瑣且仰賴專業人力，若是不慎誤食，將對人體產生巨大的風險。因此本專題將在溫度和濕度控制環境下不同發霉程度的咖啡豆氣味作為數據，透過演算法消除訊號的雜訊，並將訊號進行特徵提取，最後，訓練卷積神經網路（CNN）來預測其結果，準確率高達96.1%。

為了減少推導過程的計算量與功耗，我們設計了硬體語言模型，其好處是透過電路的特性和大量的並行運算，加速推導過程，並將模型部屬在 FPGA 上。最後，我們比較幾種硬體加速模型的速度與功耗，也將設計過的硬體模型與軟體模型的效率進行比較，驗證其加速效果。

本專題期待能發揮深度學習和硬體語言兩者的優點，軟硬結合地去解決現實問題，另外開發過程中也考慮了通用性，透過載入不同參數，該模型可不僅應用在咖啡，也可以解決茶葉、農作物，食品工廠等等其他領域。

Introduction

根據國際咖啡組織（ICO）調查，台灣人一年共喝掉約28.5億杯咖啡，平均每人每年喝掉約120杯咖啡[1]。依據財政部資料顯示，2021年咖啡豆的進口量達40866公噸，進口金額突破2億美元[2]。在如此龐大的咖啡市場裡，咖啡豆的品質控管就相當重要了。

咖啡豆的品質由許多因素影響，最重要的就是「保存」方面了，如果咖啡豆或咖啡粉儲存環境不當，就容易造成赭麴黴菌的汙染，進而產生赭麴毒素，然而發霉的咖啡豆因為與原本的顏色相近，肉眼難以分辨，容易造成民眾誤食。

赭麴毒素分為 A、B、C 三種，其中又以赭麴毒素 A 毒性最強。赭麴毒素 A 具腎臟毒性，且於實驗證明具有致癌性，國際癌症研究組織（The International Agency for Research on Cancer，IARC）於1993年將赭麴毒素 A 列為2B 類（對人可能致癌之物

質）。根據食藥署規範，赭麴毒素 A 限量為5 ppb 以下，與國際食品法典委員會（Codex）、歐盟及中國相同，主要是因台灣的咖啡豆大部分仰賴進口，若食藥署執行邊境查驗發現赭麴毒素 A 超標，就會立即停止進口。歐盟曾評估發現，其成員國民眾平均每週經食物攝取45 ng/kg b.w 的赭麴毒素 A，其中50%來自穀物，13%來自酒類，10%來自咖啡，8%來自香辛類，1%來自肉品，代表人體攝取到的赭麴毒素 A 有不小的比例源自於咖啡[3]。

然而目前檢測發霉咖啡豆的方式還是十分困難，一般檢測方法有兩種，破壞性檢測和非破壞性檢測。破壞性檢測方式如衛福部食藥署110年修正公告之「食品中黴菌毒素檢驗方法—赭麴毒素 A 之檢驗」方法[4]，使用高效液相層析儀（High Performance Liquid Chromatography, HPLC），該方法是先將檢測物體固相萃取及淨化後，再進行高效液相層析測定毒素濃度，但這種方式分析步驟較為繁瑣，並且需要儀器和相關知識才能操作；非破壞性檢測方式不外乎人工目測檢測和氣體檢測兩種方式，如上段所述，人工目測的方式會因為顏色相近難以識別咖啡豆是否發霉，並且判斷的標準不一，氣體檢測的方式可以利用市面上的氣體檢測系統對咖啡豆氣體進行分析。

電子鼻系統是一種利用多種不同的氣體感測器來模仿哺乳類動物嗅覺的感測系統，屬於非破壞性的氣體檢測，氣體通過不同的感測器時，會在其表面產生各種電訊號，再經由電路傳輸，最後透過演算法來進行分析。電子鼻系統與動物嗅覺系統對應關係如下圖1：鼻腔之於氣體反應腔室，提供反應空間給吸入的氣體；嗅覺細胞之於不同的感測器，是氣體與感測器反應得到電訊號變化；神經元之於讀取電路，用來傳輸，擷取訊號；大腦之於辨識演算法，對感測器之訊號變化進行學習與辨識[5]。

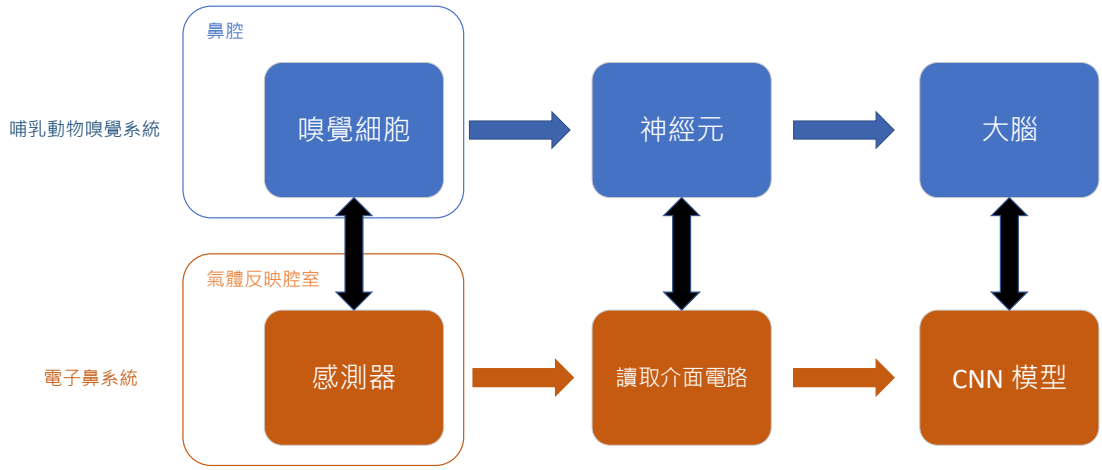

圖1 電子鼻系統與動物嗅覺系統對應關係

為了使咖啡店家和一般咖啡買家方便對手邊的咖啡豆發霉程度進行判斷，本專題使用實際的咖啡豆氣體樣本，建立一個簡單的神經網路模型模擬電子鼻系統內的大腦，辨識咖啡豆的新鮮、半發霉、發霉三種狀態。因為這個架構只預測咖啡豆的發霉狀態，比起市面上複雜氣體檢測系統，能以**更快的速度、更小的面積、更小的能耗**對咖啡豆的發霉狀態進行單一的預測，更好地去因應近年來日漸龐大的咖啡市場。

Method

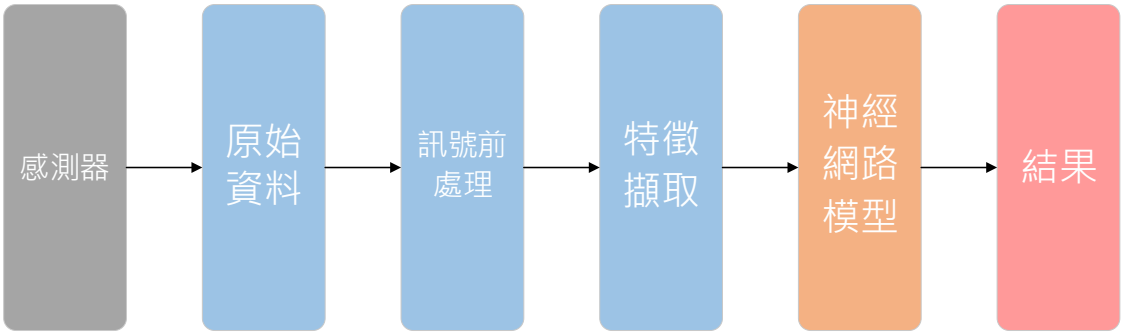

圖2 Operating flow

1. 感測器（sensor）

本專題使用資料是來自14顆金屬氧化物感測器，感測器的型號與目標氣體分別為

表1 氣體感測器型號與目標氣體

Number	Sensor	Target gas
1	TGS-2600	Methane, Carbon monoxide, Hydrogen, Iso-butane
2	TGS-2602	Hydrogen, Ammonia, Ethanol, Toluene
3	TGS-2603	H2S, Hydrogen, Trimethyl amine, Methyl mercaptan
4	TGS-2610 COO	Ethanol, Hydrogen, Methane, Iso-butane/Propane
5	TGS-2611 COO	Ethanol, Methane, Iso-butane
6	TGS-2620	Methane, CO, Iso-butan, Ethanol
7	TGS-2612	Ethanol, Methane, Iso-Butane/Propane
8	FIS-SB5100	Hydrogen sulfide, Carbon monoxide, Hydrogen, Ethanol
9	FIS-SB 5300	Hydrogen, Carbon monoxide, Isobutene, Ethylene, Ethanol, Ammonia
10	FIS-SB-AQ1-06	Methane, Iso-butane, Hydrogen, Ethanol, CO
11	FIS-SB-30-04	Iso-butane, Hydrogen, Ethanol
12	FIS-SP3S-AQ2-01	Methane, Iso-butane, Hydrogen, Ethanol, CO
13	FIS-SP5-3B00	Hydrogen, Carbon momoxide, Iso-butane, Ammonia, Methane, Ethanol
14	FIS-SP3-6100	Ozone, Iso-butane, Hydrogen, CO, NO/NO2, Ethyl alcohol

（Reference: Tang, Chang-Lin. (2022). Development of a Non-Destructive Moldy Coffee Beans Detection System Based on Electronic Nose. National Tsing Hua University, Hsinchu, Taiwan. pp.28）

接著，透過介面電路將感測器的阻值變化轉換為類比電壓變化，最後設定訊號擷取的頻率，把感測器的電訊號讀出，產生原始資料。

2. 資料處理（data processing）

因為感測器測出來的訊號可能會因為環境因素產生偏差值或是雜訊，所以要先透過分數比例法（fractional difference）將感測誤差進行消除，分數比例法公式如下：

$$R_{FD} = \frac{R_s - R_0}{R_0} = \frac{\Delta R_s}{R_0}$$

R_s：氣體感測器的反應後穩態電阻值

R_0：反應前基線電阻值

將14個感測器的資料做為14個特徵（feature），分別對每個特徵做標準化（standardize），標準化可以提升神經網路模型的收斂速度、增加精準度、防止模型梯度爆炸，標準化公式如下：

$$X_{std} = \frac{X-\mu}{\sigma}$$

X：感測器資料

μ：感測器資料的平均值 σ：感測器資料的標準差

3. 神經網路模型（Neuron Network）

3.1 架構（architecture）

本專題採用的模型是類似於 Lenet 的架構，由三層的卷積層（convolution layer）和兩層的全連接層（fully connected layer）構成，輸入（input）為14筆感測器測出來的特徵（1-D vector），輸出（output）為0（新鮮）、1（半發霉）、2（發霉）三種狀態，選用卷積層的架構可以使權重數量大幅下降，比起都是全連接層的架構，更能節省暫存器和記憶體的空間。

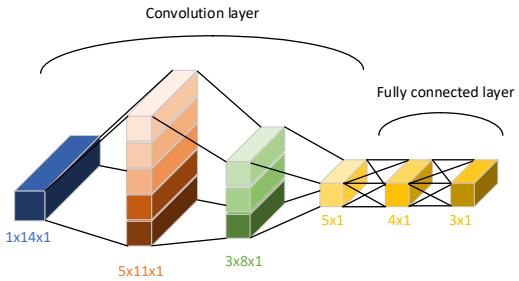

圖3 CNN 神經網路架構

表2 CNN 模型各層架構

Layer	Input size	Output size	Weight size	Stride
CONV_1	1 x 14 x 1	5 x 11 x 1	Weight: 5 x 1 x 4 x 1 Bias: 5 x 1	1
CONV_2	5 x 11 x 1	3 x 8 x 1	Weight: 3 x 5 x 4 x 1 Bias: 3 x 1	1
CONV_3	3 x 8 x 1	1 x 5 x 1	Weight: 1 x 3 x 4 x 1 Bias: 1 x 1	1
FC_4	5 x 1	4 x 1	Weight: 4 x 5 x 1 Bias: 4 x 1	
FC_5	4 x 1	3 x 1	Weight: 3 x 4 x 1 Bias: 3 x 1	

3.2 卷積並行數據流技巧探討

卷積是 CNN 架構裡最主要的運算，它包含了輸入特徵（input feature maps）、權重（kernel weights）三個維度的乘積和累加，主要由四層的迴圈（loops）實現：

1. **Loop-1**：在一個 kernel window 內做一對一的相乘，再用 adder tree 進行累加
2. **Loop-2**：input feature maps 間的移動也可以視為 input channel 的交換
3. **Loop-3**：kernel window 在 input feature map 上的滑動
4. **Loop-4**：output feature maps 間的移動也可以視為 output channel 的交換

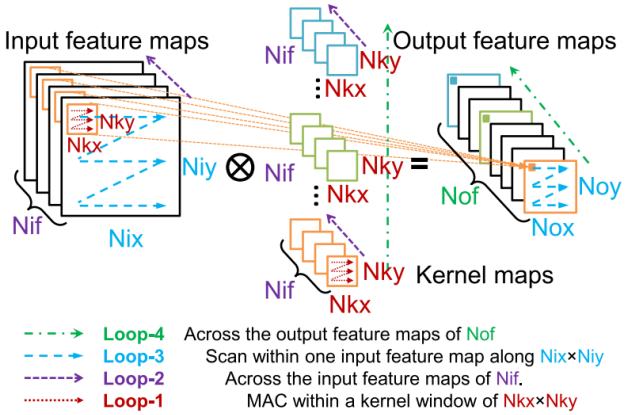

圖 4 卷積的四層迴圈[6]

根據上述的描述，如果將不同的 Loop 做並行化處理（展開），會造成截然不同的卷積運算，這會造成不同的 PE 架構、資料的重複利用（data reuse）和記憶體的拿取（memory access），首先就來探討不同 Loop 展開會有什麼區別：

1. **Loop-1 unrolling**：每個權重與相同大小的輸入（kernel window）進行內部乘積展開，並對所有乘積進行相加，所以需要與 kernel window 內權重數相同的 PE 數量，在一個 cycle 內對一個 kernel window 內的所有值進行相乘和相加。
2. **Loop-2 unrolling**：對相同位置的輸入和相同位置的權重，不同的 input channel 進行展開，所以需要與 input channel 相同數量的 PE，在一個 cycle 內對不同的 input channel 進行相乘和相加。
3. **Loop-3 unrolling**：對相同 kernel window 大小，不同位置的 input feature map 進行展開，所以需要與單一 output feature map 相同數量的 PE 和寄存器，在一個 cycle 內對每個不同位置的 input feature map 進行相乘再與寄存器內的值累加。
4. **Loop-4 unrolling**：對相同位置的輸入和相同位置的權重，不同的 output channel 進行展開，所以需要與 output channel 相同數量的 PE 與寄存器，在一個 cycle 內對不同的 output channel 進行相乘再與寄存器內的值相加。

本專題主要是利用 unroll Loop-2 和 unroll Loop-4，對 input channel 和 output channel 做展開，除了會比較兩種 unrolling 的效果，還會將兩種 unroll 的方式做結合，最後再加上我們自己調整的架構做效能探討。

3.3 Stationary

Stationary 的數據流主要分為三種[7]：

1. weight stationary（WS）：指的是最小化 weight 讀出的功耗，透過讓 weight 都從 RF（register file）讀出，因為從 RF 讀出功耗較從 DRAM 讀出低。
2. output stationary（OS）：指的是最小化 partial sums 讀寫的功耗，透過將要累加的 partial sums 存在 RF，使得不必在 DRAM 和計算單元間傳輸資料。
3. row stationary（RS）：指的是最大化所有 reuse 並在 RF level 進行累加，不同於 output stationary 和 weight stationary，只優化 weight 和 partial sums 的部分，是最小化整體讀寫的功耗。

本次專題因為架構小的因素，能實現 row stationary 的數據流，在一維的卷積下，保持每列 filter 的 weight stationary，再將 input activations 串行傳入 PE，在一個 clock cycle 內對 kernel window 內的值進行相乘，累加成 partial sum 放入存儲空間，這樣就能實現 1-D 的 row stationary。

3.4 FPGA 架構

本專題使用的 FPGA 板為 PYNQ-Z2，為了善用 FPGA 板 PS 端（processing system）和 PL 端（programmable logic）相互溝通的優勢，我們設計的架構有以下幾點特徵：

1. 16-bit 定點數量化：將32-bit 浮點數轉為16-bit 定點數，節省運算與存儲資源
2. Direct Memory Access（DMA）：PS 端與 PL 端的溝通協議，使用 GP 接口控制系統，HP 接口實現資料串行（data streaming）
3. 並行計算：透過 unroll 不同的 loops，加速計算的運行
4. Input & Weight reuse：因為架構小，將資料存在 RF（register file）或 BRAM（block RAM），節省資料傳輸時間，並且達到 row stationary
5. 資料處理放在 PS 端：方便使用演算法對資料進行處理
6. CNN 架構放在 PL 端：方便並行化處理，提升運行速率

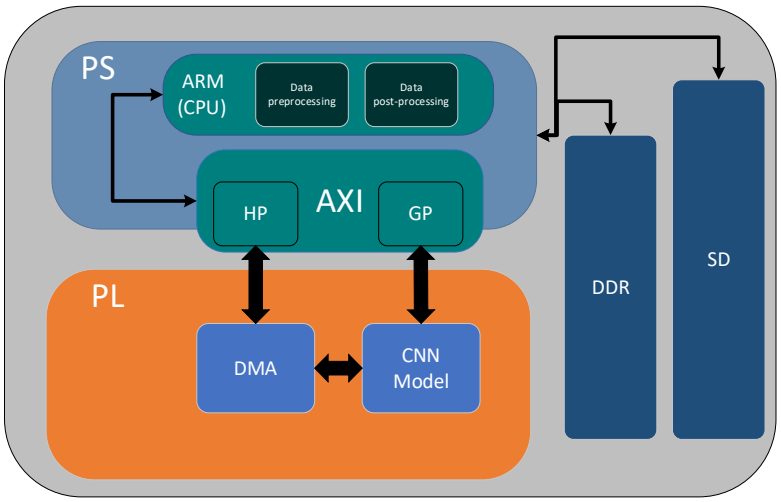

圖5 FPGA 架構示意圖

Results

1. 數據流比較

1.1 資源使用率（Utilization）

我們比較的 FPGA 板上資源，主要有以下四種：

LUT（lookup table）：跟地址查找有關，根據輸入去找到相應位置的信號，然後做輸出

FF（flip flop）：1bit 的存儲器，是 register 的基本組成單元

BRAM（block RAM）：RAM 的一種，具有很高的運行速率，但有限的資源數量

DSP：MAC計算單元，跟所有加乘運算有關

從下圖6可以發現對不同 loop 做並行化所消耗的資源是不同的，關於 LUT 的使用量，因為並行化的架構都需要同時對不同值做查找，所以有做並行化的架構必定比沒做並行化的使用率高，另外，在我們自己調整的架構中「our work」，有對某些 kernel window 做展開，所以會造成 LUT 的大量使用；關於 FF 和 BRAM 的使用量，因為模型架構小和資料量小，所以各組的 FF 和 BRAM 的使用量是差不多的，但有並行化的結構的架構還是會比沒有的略高一些；關於 DSP 的使用量，對內層展開越多同時需要的計算單元也就越多，如果同時展開 Loop-2&4 對計算資源的使用量會最大，「our work」因為沒有對所有內層的 loop 進行展開，所以 DSP 的使用量與「original work」差不多。

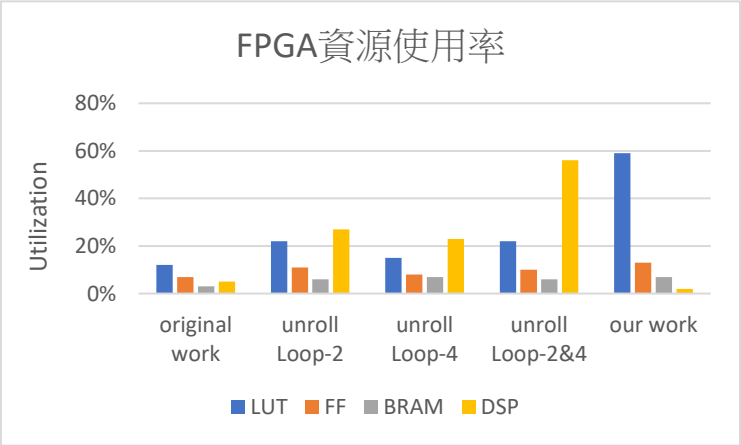

圖6 FPGA 資源使用率比較

1.2 能耗、時間、正確率

觀察平均消耗時間會發現有做並行化的架構通常每次預測所花的時間都比沒做並行化（original work）的低，但 unroll loop-2 的時間卻跟 original work 差不多，我們推測是因為我們的 CNN 架構中 input channel 的數量太少，導致 unroll loop-2 的效果無法發揮，甚至比原本的還要差。

功耗方面，我們有去觀察更細部的功耗報告，發現五種架構每個項目（clocks, signals, logic, BRAM, DSP, PS7）的功耗佔比與資源使用率差不多，「our work」的 signals power 較高，原因是本架構將 input 做展開，不再使用 stream 的方式傳入資料，所以 signals 的能耗會比其他四種架構高。

正確率方面，因為都是同樣的量化長度 16bit fixed point，所以計算出的結果理應相同，正確率亦相同。

表3 能耗、時間、正確率比較

	Power consumption	Average time (per data)	Correction
Original work	1.479W	1.742ms	95.39%
Unroll Loop-2	1.577W	1.744ms	95.39%
Unroll Loop-4	1.559W	1.699ms	95.39%
Unroll Loop-2&4	1.652W	1.688ms	95.39%
Our work	1.758W	1.685ms	95.39%

表4 功耗佔比

	clocks	signals	logic	BRAM	DSP	PS7
Original work	2%	2%	1%	1%	1%	93%
Unroll Loop-2	2%	3%	3%	1%	3%	87%
Unroll Loop-4	2%	3%	2%	1%	3%	88%
Unroll Loop-2&4	2%	4%	3%	1%	6%	83%
Our work	3%	10%	8%	1%	<1%	77%

2. FPGA 加速

實現硬體加速是本專題的主要目標之一，從下圖7就可以很明顯的發現，用 PL 端處理 CNN model 的優勢，圖中的 PS 指的是使用 numpy 在 PS 端處理 CNN model，右邊的五種都是在 PL 端使用不同並行化架構處理 CNN model，使用 PL 端與 PS 端相比最大速率能相差 8.8 倍。

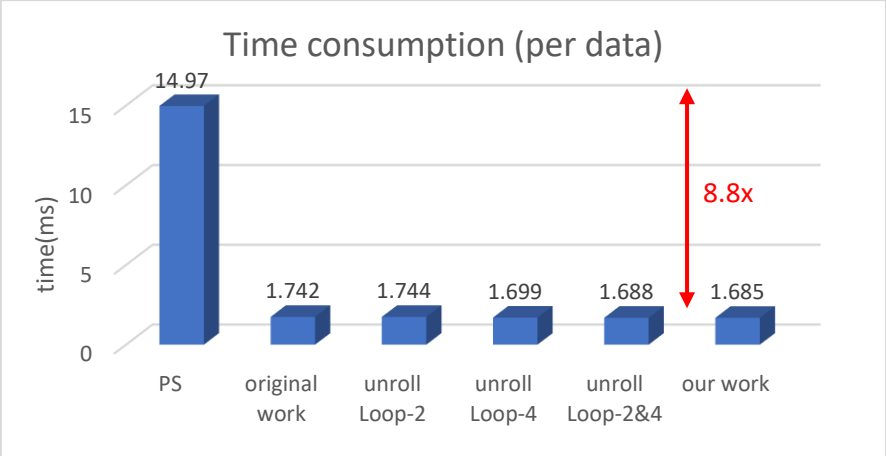

圖7 硬體加速比較

3. 量化前後的差異

對資料和權重進行量化，將 32bit 的浮點數轉為 16bit 的定點數，好處是加速運算和節省記憶體，因為 fixed point 的乘法較 floating point 簡單，缺點是降低了精度，造成預測的正確率降低，但以0.7%的正確率換取一半的儲存空間是相當划算的，如下表3-2。

表5 量化前後的比較

	量化前	量化後
正確率	96.09%	95.39%
資料讀取量（all test data）	24.3 KB	12.2 KB
記憶體使用量（per inference）	392 bytes	196 bytes

4. 結果探討

透過 FPGA 和硬體語言的優勢，我們可以很輕鬆地加速 CNN 模型計算，實驗的加速效果也達到了8.8倍，符合我們的實驗目的。

另外，因為卷積計算的性質，我們能設計不同的卷積架構和數據流進行進一步的比較，「our work」是基於我們的 CNN 模型的效率對各層並行化進行取捨的架構，如果單就速率層面來看「our work」是最好的，如果單就資源使用率來看，不做並行化能節省最多的資源，如果以整體效能來比較，output channel 的展開，也就是 unroll loop-4，能以較少的資源獲得最多的加速效果，各架構的取捨要以實際情形來決定。

最後，不同於數據流層面的優化，量化是在資料層面進行優化，以正確率作為代價，換取較低的儲存空間和較快的運算速度，在能接受的正確率範圍（≧95%）對資料進行量化，在本次實驗也驗證了這個結果。

Conclusion

本專題在咖啡豆氣味數據 14 個特徵的前提之下，建構專屬的 CNN 模型，利用深度學習的技術對咖啡豆氣味進行判別，並結合了硬體語言架構，加快預測速度，比較各種數據流與量化的效果，各種架構其實就是在資源使用率、速率、正確率之間進行取捨，該用什麼架構還是得以實際情況來做決定，另外，各架構實驗正確率都達 95% 以上，也驗證了我們 CNN 模型的可行性。

這次的模型因為架構小、資料量少，所以很多加速優化的方式都不能使用，或提升不大，但也因為架構簡單，消耗的資源、能量較小，能較容易實做在終端裝置上，並且因為 CNN 模型只是辨認的演算法，所以也可載入不同的 weight 對除了咖啡豆以外的氣味進行預測，增加泛用性。

Feedback

非常感謝鄭桂忠教授給予機會實作本次專題，讓我們可以了解深度學習的基本概念和其在硬體相關領域的發展，甚至透過一個專案的實作，徹底地了解有關數位 IC 的設計流程，也特別感謝陳彥文學長的帶領，通過一些論文了解實作上的細節，如何研究出更好的能耗表現與計算效率。

經過上學期的培訓，透過修課、論文和線上資源了解與專題相關的部分，下學期終於開始實作，從應用領域開始真正認識到一切不只是紙上談兵，專題架構的每一個部分都是專業領域，從嵌入式開發，CNN model，PS 和 PL 的交互，ip 的封裝，計算單元加速等等，可能耗費幾個晚上的絞盡腦汁，都只是相關專業的入門石而已，更認識到自己的不足，儘管過程一路崎嶇，不斷地的拐彎和試錯，卻也看到更多實作 FPGA 的細節，有時候納悶自己的 bug 會不會太冷門，想不到網路一查卻是熱門討論，或是 zynq-7000的實作注意事項，可能就隱藏在那本1000多頁的官方操作手冊中一小段裡，令人感到無言，不論如何，專題還是順利結束了，在此再次感謝鄭桂忠教授和陳彥文學長的鼎力相助。

Reference

[1] 鍾泓良. (2022, December 25). 咖啡豆進口額 十年增一倍. UDN. https://udn.com/news/story/7241/6862800

[2] Ministry of Finance, Taiwan. (n.d.). Ministry of Finance, Taiwan. https://www.mof.gov.tw/

[3] 台東醫院藥劑科. (2011, August 24). 認識食品中的赭麴毒素 A. https://www.tait.mohw.gov.tw/?aid=509&pid=0&page_name=detail&iid=343

[4] Food and Drug Administration, Taiwan. (2021, January 8). Method of Test for Mycotoxins in Foods - Test of Ochratoxin A. Food and Drug Administration, Taiwan. https://www.fda.gov.tw/tc/siteListContent.aspx?sid=103&id=38781

[5] Tang, Chang-Lin. (2022). Development of a Non-Destructive Moldy Coffee Beans Detection System Based on Electronic Nose. National Tsing Hua University, Hsinchu, Taiwan. pp. 13, 27-29.

[6] Yufei Ma, Yu Cao, Sarma Vrudhula, Jae-sun Seo. (2018, July). Optimizing the Convolution Operation to Accelerate Deep Neural Networks on FPGA. IEEE. pp. 2-4.

[7] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, Joel Emer. (2017, August). Efficient Processing of Deep Neural Networks: A Tutorial and Survey. IEEE. pp. 14-15