Arm Cortex-M 微控制器的機器學習

主任工程師Naveen Suda
機器學習演算法總監Danny Loh

arm

白皮書

機器學習 (ML) 演算法因各種考量如延遲、耗電量、成本、網路頻寬、可靠性、隱私與安全，正朝物聯網終端移動。因此，開發類神經網路 (NN) 解決方案、將它們部署在例如Arm Cortex-M微控制器系統這類低功率終端裝置的興趣與日俱增。為促成這類部署，我們提出了CMSIS-NN。這是一種經優化的軟體核心開放原始碼函式庫，可極大化Cortex-M核心的NN效能，而且只需使用最小的經常性記憶體。我們進一步提出NN架構探索的方法，以CIFAR-10數據集的影像檔分類為例，開發適用在條件相當受限裝置的模型。

—關鍵字：機器學習、深度學習、類神經網路、嵌入式系統、物聯網、節能、Cortex-M

I. 概論

連網裝置或物聯網 (IoT)過去幾年內快速擴展，到2035年，各個區隔市場的總數預測將達到1兆個[1] 。這些物聯網終端裝置通常包含數個可以搜集數據的感測器，包括音訊、視訊、溫度、溼度、GPS 位置與加速等。通常，大多數感測器搜集到的數據是由雲端的分析工具進行處理，以便運作各式的應用，如工業監控、家庭自動化與健康照護。不過，隨著物聯網節點數目增加，對網路的頻寬帶來相當的負擔，同時也增加物聯網應用的延遲性。此外，對雲端的依賴也讓在網路連線不穩定或有限的區域部署物聯網應用，相當具有挑戰性。這個問題的解決方案之一是終端運算，這種運算在數據的源頭、也就是物聯網的終端節點進行，因此可以降低延遲，並節省數據通訊消耗的能源。

類神經網路（NN）架構的解決方案，針對如影像分類、語音辨識與自然語言處理等複雜機器學習應用，已經展現出與人類一樣水準的準確性。有鑑於運算的複雜性與資源的需求，NN的執行絕大多數都侷限於高效能伺服器CPU，或專用硬體（如GPU或加速器）的雲端運算，但會讓物聯網應用增加延遲。在數據的源頭（通常是微控制器）立即進行分類，可以降低整體的延遲以及物聯網終端與雲端間數據通訊的能源消耗。不過，在微控制器部署NN，有下列挑戰：

- **有限的記憶體使用量：**微控制器系統通常只有10幾到100多KB的可用記憶體。整個類神經網路模型，包括輸入/輸出、權重與啟動，都必須在如此小的記憶體上限內運行。
- **有限的運算資源：**許多分類任務都有不斷線啟動與即時的要求，這會限制每個類神經網路推論的總運算次數。

這些挑戰可以從裝置與演算法的角度加以應對。我們一方面在執行類神經網路工作負載時，可以靠優化低階運算核心來達成更佳的效能與更小的記憶體使用量，並藉此提升這些微控制器的機器學習能力。這可以讓微控制器處理更大型與更複雜的NN。另一方面，關於目標硬體平台，類神經網路可以靠NN架構的探索來進行設計與優化。這樣可以在固定的記憶體與運算配置上限內，提升NN的品質(也就是準確性)。

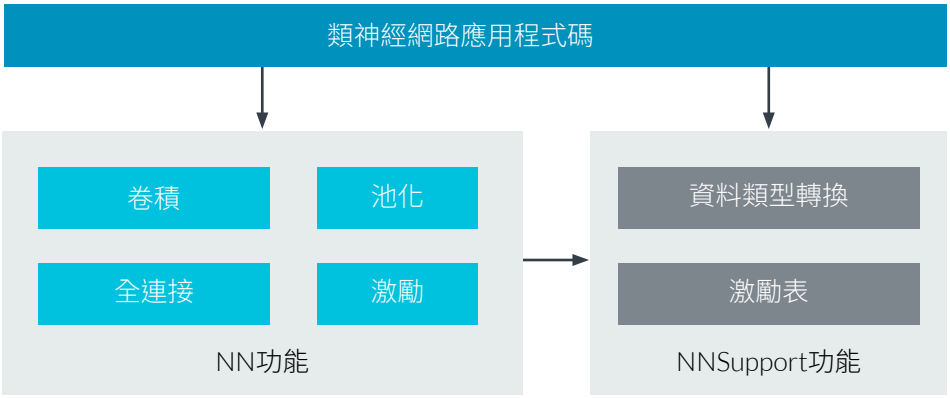
在這份報告中的第二單元提出CMSIS-NN[2]。CMSIS-NN是大量的高效類神經網路核心，開發目的是讓鎖定智慧物聯網終端裝置的Arm Cortex-M處理器核心上的類神經網路，極大化它的效能並極小化記憶體的使用量。架構在CMSIS-NN核心基礎上的類神經網路推論，可以達成4.6倍的runtime/數據吞吐量提升，以及4.9倍的能源效率提升。

在第三單元中，我們使用CIFAR-10數據集的影像分類應用為例，針對微控制器記憶體/運算限制，提出搜尋類神經網路架構的技巧。

II. CMSIS-NN

CMSIS-NN類神經網路核心的總覽，如圖表一顯示。核心編碼包含兩個部份：NNFunctions與NNSupportFunction。NNFunctions包含實作常見類神經網路網路層類型的函數，如卷積、深度可分離卷積結構、全連接 (也就是內積)、池化與啟動。這些函數可以讓應用程式碼使用，以實作類神經網路的推論應用。核心API則刻意保持簡單，以便針對TensorFlow、Caffe 或PyTorch等所有機器學習框架輕鬆重新鎖定。NNSupportFunctions包含公用的程式函數，例如NNFunctions使用的數據轉換與激勵函數表。應用程式碼也可使用這些函數來建構更複雜的NN模組，如長短期記憶 (LSTM) 或閘控再流裝置 (GRU) 單元。

圖表1：CMSIS-NN
類神經網路核心總覽



對於某些核心，如全連接與卷積的核心，我們會實作不同版本的核心函數。提供的是一個針對所有網路層參數不必改變就可以通用的基本版本。我們也實作包括進一步優化技巧的其它版本，它們不是具有變型輸入，就是對網路層參數有某些限制。

A. 固定點量化

研究顯示，即便是低精密度定點表示法，NN的運作依然良好[3]。固定點量化可以協助避免進行昂貴的浮點運算，並降低儲存權重與啟動的記憶體使用量，這對資源受限的平台極為關鍵。儘管不同網路或網路網路層的精密度需求可能不同[4]，CPU很難運行帶有不同位元寬度的資料類型。在這裡，我們開發了同時支援8位元與16位元數據的核心。

核心採用跟CMSIS-DSP裡使用的資料類型格式相同，也就是把q7_t　當作int8、把q15_t當作int16　，並把q31_t　當作int32。執行量化時，我們假定固定點格式具有兩次方的定標。量化格式以Qm.n代表，而代表值是Ax2-n，其中的A是整數值而n是Qm.n的一部份，代表該數字針對分數部份使用的位元數，也就是顯示數基點的所在地。我們跳過偏差用的定標因素，並把它以參數輸出至核心；因為是二次方定標的關係，定標的實作按位元移位操作。

在NN的運算期間，代表不同數據，也就是輸入、權重、偏差與輸出的固定點可能不同。bias_shift 與out_shift這兩個輸入參數，用來替運算調整不同數據的定標。

下列方程式可以用來估算移動值：

$$bias\_shift = n_{input} + n_{weight} - n_{bias} \quad (1)$$

$$out\_shift = n_{input} + n_{weight} - n_{output} \quad (2)$$

其中的n_{input}, n_{weight}, n_{bias} *與* n_{output} ，分別是輸入、權重、偏差與輸出中的分數的位元數。

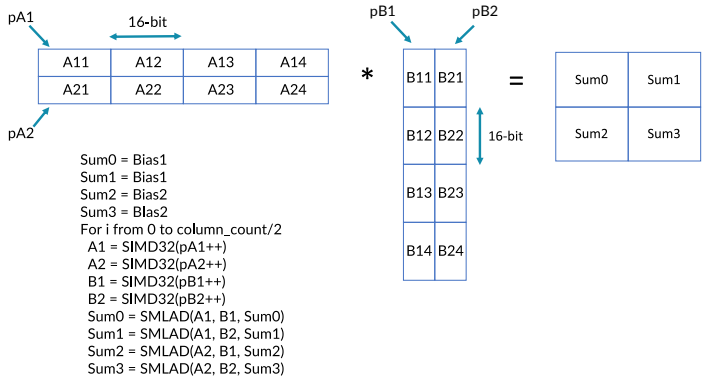
B. 軟體核心優化

在這個單元中我們凸顯了在CMSIS-NN裡已進行的優化工作，以提升效能並降低記憶體的使用量。

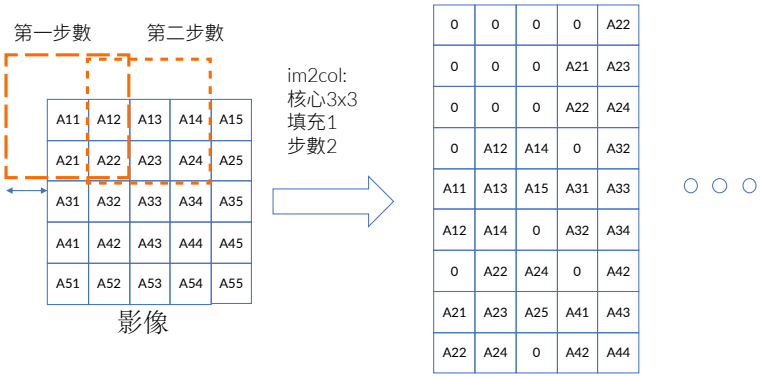
1) 矩陣乘法：矩陣乘法是類神經網路中最重要的運算核心。這個工作的實作，是基於CMSIS-DSP內的mat_mult核心。如圖表2顯示，矩陣乘法核心是以2x2核心實作，與CMSIS的實作類似。因此可以允許部份數據再次使用，也可以節省載入指令的總筆數。累積是使用q31_t資料類型完成，而兩個運算元都屬於q15_t資料類型。我們使用相對應的偏差值，讓累加器初始化。運算的執行則是使用專用的SIMD MAC指令_SMLAD。

2) 卷積：卷積網路層藉由在輸入特徵映射中運算過濾器權重與小接受區之間的點乘積，擷取新的特徵映射。通常來說，CPU架構的卷積實作可以解構成輸入紀錄、擴展(也就是im2col與image-to-column) 以及矩陣乘法操作。im2col是把類影像的輸入轉化成「行」，而「行」則代表每個卷積過濾器需要的數據。圖表3顯示im2col的一個範例。

圖表2 ：具有2x2 核心的矩陣乘法的內迴圈。每個迴圈運算兩行與兩列點乘積結果，也就是產生四個輸出。



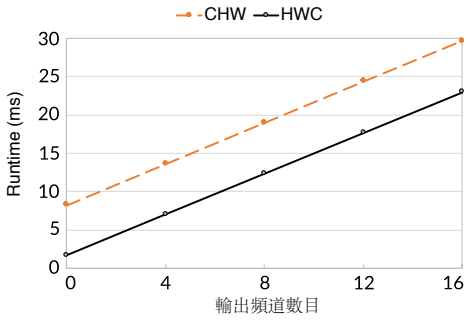
圖表3：具3x3核心、填充1與步數2的im2col的2D影像範例



im2col 主要的挑戰之一是記憶體使用量的增加，因為輸入影像中的畫素在im2col 輸出矩陣中重複。為了紓解記憶體使用量問題、同時維持im2col 的效能優點，我們的卷積核心實作了部份的im2col。核心一次只會擴展兩行，這已經足夠從矩陣乘法核心取得極大化的效能提升，同時把記憶體負擔維持在最小。影像數據格式也會影響卷積的效能，特別是im2col的效率。兩種最常見的影像數據格式是頻道- 寬度- 高度 (CHW ；也就是頻道為最後)，與高度- 寬度- 頻道 (HWC；也就是頻道為第一)。維度的順序則與數據步數的順序一樣。在HWC格式中，頻道的數據以步數1儲存，沿著橫向寬度的數據則是以頻道數的步數儲存；沿著縱向高度的數據，則以 (頻道數x影像寬度) 步數儲存。

只要權重與影像的維度順序一樣，數據的布局對於矩陣乘法的運作就沒有影響。im2col只會與寬度及高度的維度一起運作。HWC式樣的布局可以促成高效率的數據移動，因為每個畫素的數據 (也就是同樣的x與y位置) 是連續地儲存，並且可以用SIMD指令有效率地進行複製。為了驗證這一點，我們實作了CHW與HWC版本，並比較它們在Arm Cortex-M7的runtime。我們在圖表4把結果凸顯出來；把HWC輸入固定為16x16x16，並很快的輸出頻道數目。當輸出頻道值為零時，代表軟體只執行im2col，並沒有進行任何矩陣乘法的運作。與CHW布局相比，HWC擁有較短的im2col runtime，但矩陣乘法效能卻相同。因此，我們用HWC數據布局來實作卷積核心。

圖表4：CHW與HWC數據布局卷積執行時間的比較。兩種布局都有同樣的矩陣乘法runtime，但HWC的im2col runtime比較短。



C. CMSIS-NN 的結果

我們測試CNN的CMSIS-NN核心，CNN則利用CIFAR-10數據集進行訓練。數據集包含6萬個32x32的彩色影像，並分為十個輸出類別。網路拓撲是基於Caffe內提供的內建範例，具有三個卷積網路層與一個完全連結的網路層。所有網路層的權重與激勵數據都量化成q7_t格式。runtime則是用具有一顆時脈216 MHz的Arm Cortex-M7核心的STMicroelectronics NUCLEO-F746ZG Mbed開發板進行測量。

圖表1：數據吞吐量與能源效率的提升

網路層類型	基準線 runtime	CMSIS-NN runtime	SRAM 快取記憶體	
			數據吞吐量	能源效率
卷積	443.4 ms	96.4 ms	4.6 X	4.9 X
池化	11.83 ms	2.2 ms	5.4 X	5.2 X
ReLU	1.06 ms	0.4 ms	2.6 X	2.6 X
總計	456.4 ms	99.1 ms	4.6 X	4.9 X

整個影像分類中的每個影像，大約花費99.1微秒(相當於每秒10.1張影像)。CPU在這個網路運行的運算吞吐量，大約是每秒249百萬運算(MOps)。預先量化的網路針對CIFAR-10測試集達成80.3%的準確率。用Arm Cortex-M7核心運行的8位元量化網路，則達成79.9%的準確率。使用CMSIS-NN核心的最大記憶體使用量約為133KB，此時我們用部份的im2col來實作卷積以節省記憶體，接下來則進行矩陣乘法。少了部份im2col的記憶體使用量，大約為332 KB，此時神經網路無法在開發板上使用。為了量化CMSIS-NN核心對既有解決方案帶來的優點，我們也使用一個1D卷積函數(來自CMSIS-DSP的arm_conv)、類Caffe池化與ReLU，實作一個基準線版本。

針對CNN應用，圖表1總結基準線函數與CMSIS-NN核心的比較結果。與基準線函數相比，CMSIS-NN核心的runtime與吞吐量分別提升2.6倍與5.4倍。節能方面的提升也與吞吐量的提升相若。

III. 硬體條件受限的NN模型

在這個單元裡，我們使用影像分類應用為範例，凸顯了為部署應用的硬體平台 (也就是微控制器) 選擇正確類神經網路架構的重要性。為此，我們需要瞭解微控制器的硬體限制。

微控制器通常包含處理器核心、一個當成主記憶體的靜態隨機存取記憶體(SRAM)，以及用來儲存編碼與數據的嵌入式快閃記憶體。表格2顯示具有Arm Cortex-M核心的一些市售微處理器開發板，它們擁有不同的運算與記憶體容量。

微處理器系統中的記憶體數量，會限制系統可運行的類神經網路模型的大小。除了記憶體限制，類神經網路的龐大運算需求也會為在微控制器上運行NN，帶來另一個關鍵限制：它們為了維持低耗電，通常都以低時脈運行。因此，我們必須選擇對的NN架構，來配合部署NN模型的硬體在記憶體與運算上的限制。為了評估在不同硬體限制條件下神經網路的準確性，我們選擇三個同不同尺寸的系統配置，並導出每個配置需要的類神經網路需求，如表格III顯示。在這裡，我們假定每秒標稱可進行10個影像的分類推論 (也就是每秒10幀)，以便導出神經網路的需求。

表格II：現成的Arm Cortex-M

Arm Mbed™ 平台	處理器	時脈	靜態隨機存取記憶體	快閃記憶體
LPC11U24	Cortex-M0	48 MHz	8 KB	32 KB
nRF51-DK	Cortex-M0	16 MHz	32 KB	256 KB
LPC1768	Cortex-M3	96 MHz	32 KB	512 KB
Nucleo F103RB	Cortex-M3	72 MHz	20 KB	128 KB
Nucleo L476RG	Cortex-M4	80 MHz	128 KB	1MB
Nucleo F411RE	Cortex-M4	100 MHz	128 KB	512 KB
FRDM-K64F	Cortex-M4	120 MHz	256 KB	1MB
Nucleo F746ZG	Cortex-M7	216 MHz	320 KB	1MB

表格III：神經網路的限制

NN 尺寸	NN 記憶體限制	每秒運算次數/推論限制
小	80 KB	6 MOps
中	200 KB	20 MOps
大	500 KB	80 MOps

A. 影像分類用的神經網路架構

1) 卷積類神經網路(CNN)：CNN是電腦視覺應用最受歡迎的類神經網路架構。CNN包含多個依規格化散布的卷積網路層、池化與非線性激勵網路層。卷積網路層將輸入的影像解構到不同的特徵映射，從初始網路層中如邊緣、線條與曲線等低階特徵，到後面網路層的高階/抽象特徵。當代最頂尖的CNN包含100多個到1,000多個這種卷積網路層，而最後擷取的特徵則由完全連結的分類網路層分類至輸出類別。卷積運作是CNN最關鍵的運作，並且非常耗時，有超過九成的時間都花在卷積網路層上。

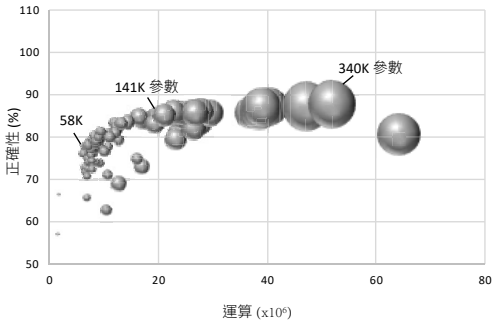
2) 近期的高效NN架構：為了降低CNN的運算複雜性，有人提議[5]用深度可分離卷積網路層當成標準卷積運作的高效率替代品。有人提議[6]利用2-D深度卷積接著1-D逐點卷積，取代標準的3-D卷積，並提出名為MobileNets的高效率NN類別。ShuffleNets [7]利用混合頻道上的深度卷積以及群組軟體1x1的卷積，來提升緊湊模型的準確性。MobileNets-V2 [8]藉由增加捷徑連接進一步提升效率，並協助深度網路的收斂。整體來說，我們已經有許多高效率的神經網路架構提案，可以用來開發符合特定硬體預算的NN模型。

B. 硬體條件受限的NN模型的搜尋

我們使用具捷徑連接的MobileNet架構，它類似讓硬體條件受限的類神經模型，進行搜尋的ResNet模型裡的架構。網路層的數量、每層網路層的特徵數量、卷積過濾器的維度與步數，被當成這次搜尋的超參數。訓練這些超參數的所有組合相當耗時，並且不太實際。因此，我們反覆執行超參數的竭盡式搜尋、計算模型的記憶體/運算需求，並且只訓練能配合硬體預算限制的模型。隨後我們從之前的集用場選擇超參數，以縮小搜尋空間，並繼續下一迭代的模型搜尋。圖表5顯示超參數搜尋的範例，這個範例顯示準確性、運算的數量，以及每個模型的參數。

經過幾個迭代後，我們在表格IV顯示於硬體條件限制內具有最高準確性的模型。請留意，由於這並不是對所有超參數進行的極盡式搜尋，我們在搜尋期間可能會漏掉一些符合硬體條件限制、且準確度極高的類神經網路模型。

圖表5：利用CIFAR-10數據集進行影像分類的類神經網路超參數搜尋 vs. 以泡泡尺寸顯示的運算及參數數量



表格IV ：來自超參數搜尋的最佳神經網路總結

NN 模型	準確性	記憶體	運算
S (80 KB, 6 MOps)	77.8%	58 KB	5.8 MOps
M (200 KB, 20 MOps)	84.7 %	141 KB	19.8 MOps
L (500 KB, 80 MOps)	87.7 %	340 KB	51.9 MOps

結果顯示這些模型擴大規模沒有問題，且針對不同的硬體預算，準確性在不同層級出現飽合。例如，針對200KB與每秒20百萬次運算(MOps)的記憶體/運算預算，模型的準確性大約在85%左右飽合，並且受到硬體的運算能力限制。瞭解類神經網路的準確性是否受運算或記憶體資源限制，對於硬體平台選擇的各種利弊得失，可提供關鍵的洞察。

IV. 結論

機器學習演算法已證實可以解決一些足以展現人類等級效能的複雜認知任務。在全新高效類神經網路架構與優化的NN軟體協助下，這些演算法正慢慢地朝物聯網的終端移動，以便讓類神經網路在這些終端裝置高效運作。我們在微控制器裝置常見的記憶體/運算限制下，提出執行NN模型搜尋的技巧，並使用影像分類為例。我們更進一步提出優化CMSIS-NN內的NN核心的方法，以便在最小的記憶體使用量下，極大化Cortex-M核心的神經網路效能。如您需要取用CMSIS-NN函式庫，敬請造訪：https://github.com/ARM-software/CMSIS_5.

參考資料

[1] Philip Sparks. “The route to a trillion devices,” – available online: https://community.arm.com/iot/b/blog/posts/whitepaper-the-route-to-a-trillion-devices.

[2] L. Lai, N. Suda, and V. Chandra, “CMSIS-NN: Efficient neural network kernels for Arm Cortex-M CPUs,” arXiv preprint arXiv:1801.06601.

[3] D. Lin, S. Talathi, and S. Annapureddy, “Fixed point quantization of deep convolutional networks,” In International Conference on Machine Learning (ICML) 2016, pp. 2849–2858.

[4] H. Sharma, J. Park, N. Suda, L. Lai, B. Chau, V. Chandra, and H. Esmaeilzadeh, “Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural network,” In ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA) 2018, pp. 764-775.

[5] François Chollet, “Xception: Deep learning with depthwise separable convolutions,” arXiv preprint arXiv:1610.02357, 2016.

[6] A.G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “Mobilenets: Efficient convolutional neural networks for mobile vision applications,” arXiv preprint arXiv:1704.04861, 2017.

[7] X. Zhang, X. Zhou, M. Lin, and J. Sun, “Shufflenet: An extremely efficient convolutional neural network for mobile devices,” arXiv preprint arXiv:1707.01083, 2017.

[8] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetV2: Inverted Residuals and Linear Bottlenecks,” In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4510-4520. 2018.