國立臺灣大學電機資訊學院電子工程學研究所

碩士論文

Graduate Institute of Electronics Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master's Thesis

高效能二值化權重神經網路之設計與實現

Design and Implementation of Efficient Binary-Weighted Neural Networks

李子筠

Tzu-Yun Lee

指導教授：闕志達 博士

Advisor: Tzi-Dar Chiueh, Ph.D.

中華民國 113 年 1 月

January, 2024

doi:10.6342/NTU202400902

doi:10.6342/NTU202400902

國立臺灣大學碩士學位論文
口試委員會審定書
MASTER'S THESIS ACCEPTANCE CERTIFICATE
NATIONAL TAIWAN UNIVERSITY

高效能二值化權重神經網路之設計與實現

Design and Implementation of Efficient Binary-Weighted Neural Networks

本論文係李子筠(R10943024)在國立臺灣大學電子工程學研究所完成之碩士學位論文，於民國 113 年 1 月 30 日承下列考試委員審查通過及口試及格，特此證明。

The undersigned, appointed by the Graduate Institute of Electronics Engineering on 30 January 2024, have examined a Master's Thesis entitled above presented by Tzu-Yun Lee (R10943024) candidate and hereby certify that it is worthy of acceptance.

口試委員 Oral examination committee:

陳志達　楊家驤　蔡佩芸

（指導教授 Advisor）

系（所、學位學程）主管 Director: 江介宏

doi:10.6342/NTU202400902

doi:10.6342/NTU202400902

致謝

此論文的完成表示我碩士生活的結束。回首這段時期的點點滴滴，無疑充斥著種種艱難與挑戰，也有著我人生中至今為止最大的變故。能夠克服這些難關，我要感謝一路走來許多人的幫助與陪伴，在他們的支持下，使我能學習與進步，完成這篇碩士論文。

首先，我由衷感謝我的指導教授，闕志達教授。感謝教授整頓好實驗室的氛圍、提供充沛的研究資源。也感謝教授在我研究陷入困境，找不到方向而原地踏步時，為我指點迷津，照亮研究的道路。此外不僅限於研究上的提點，教授也交會了我如何把事情做得更好，不放過每個細節。在待人處事上，我從教授身上學到許多。期許我在人生接下來的道路上能謹記老師的教誨，在未來持續彌補自己的不足。

感謝微系統實驗室的所有成員，謝謝柏彥學長、品翔學長、育豪學長、帝宇學長、冠緯學長、國洋學長，感謝你們營造良好的研究環境，讓我能適應實驗室的生活，樹立研究典範。謝謝名善、柏廷、正邦，在研究道路上互相扶持、奮鬥。感謝俊瑋、丕全、國偉、嘉芯、嘉宏、呈顥、惟真，感謝學弟妹們協助處理在實驗室中的各種雜物，期盼你們在研究的道路上順遂。

最後感謝我家人們。雖然你們不知道我的研究具體做了什麼，但還是無條件地在我背後默默付出。感謝你們的關愛與支持，讓我毫無後顧之憂地走完碩士的這段路，我也會持續努力，不負你們的期待。

doi:10.6342/NTU202400902

謹以此論文獻給我的貴人、家人與摯友，謝謝你們。

李子筠 謹誌

中華民國 113 年 1 月 25 日

doi:10.6342/NTU202400902

摘要

近年來，神經網路逐漸被開發用於各領域之任務，並且達到了與人類相當甚至有所超越的表現。這些成果使得基於神經網路的商品逐漸出現在我們的生活中，如何在計算資源有限的終端裝置進行神經網路推論運算顯得十分重要。因此，在維持模型表現下，以降低推論運算的計算成本成為了我們的研究重點。

本文採用了各種技巧，在不進行人工修改模型架構的前提下，基於量化感知訓練將權重進行二值化。在各種模型、應用中，我們的二值化訓練流程都能達到不錯的準確率。

在此之上，我們發現了二值化訓練中存在不穩定的震盪情況。針對這個問題，我們以 Hopfield 模型作為理論基礎，在二值化神經網路量化感知訓練中加入遲滯。使用我們以統計值作為閾值之遲滯二值化，訓練時的震盪情形得以解決。在各種模型與任務上都能有進一步的提升。

接著我們以 TPC-NAS 演算法進行神經網路架構搜尋。在數分鐘內的計算時間下進行自動架構修改，在相同的計算量限制下找到比人工設計的模型更好的架構，進一步提高二值化後模型之準確率。

最後我們以邏輯閘級模擬進行能效分析，相較於 8W8A 之精度，進行二值化後的 1W4A 能夠節省 3.57 倍的面積與 3.07 倍的功耗。並且在乘加運算上提出了基於 OAI 之近似方法，並以近似感知訓練初步證明其近似準確率。

doi:10.6342/NTU202400902

關鍵字：二值化神經網路、遲滯、圖像分類、物件偵測、自然語言處理、神經網路架構搜尋

doi:10.6342/NTU202400902

Abstract

In recent years, neural networks have gradually been applied to various tasks, achieving performances comparable to or surpassing those of humans. These advancements have led to the emergence of neural network-based products in our daily lives. The challenge of performing neural network inference computations on resource-constrained terminal devices has become crucial. Therefore, our research focuses on maintaining model performance while reducing the computational cost of inference.

This thesis employs various techniques to binarize weights based on quantization-aware training without manually modifying the model architecture. The binarization training process demonstrates good accuracy across various models and applications.

Furthermore, we identify oscillations during binarization training. To address this issue, we incorporate hysteresis into binarized quantization-aware training using the theoretical foundation of the Hopfield model. We propose using statistical values as hysteresis thresholds, which resolves oscillation issues during training and leads to further improvements in various models and tasks.

Next, we utilize the TPC-NAS algorithm for neural network architecture search. Through automated architecture modifications within a few minutes of computational time, the algorithm identifies architectures superior to manually designed models under the same computational constraints, further enhancing the accuracy of binarized models.

doi:10.6342/NTU202400902

Finally, we conduct energy efficiency analysis using logic gate-level simulation. Compared to the precision of 8 bits with 8-bit activations, the binarized model with 1-bit weights and 4-bit activations achieves a 3.57 times reduction in area and a 3.07 times reduction in power consumption.

Additionally, we propose an approximate method based on OAI for multiplicationaccumulate operations and preliminarily demonstrate its approximate accuracy through approximate-aware training.

Keywords: Binary-weighted Neural Networks, Hysteresis, Image Classification, Object Detection, Natural Language Processing, Neural Architecture Search

doi:10.6342/NTU202400902

目次

doi:10.6342/NTU202400902

doi:10.6342/NTU202400902

doi:10.6342/NTU202400902

doi:10.6342/NTU202400902

圖次

doi:10.6342/NTU202400902

doi:10.6342/NTU202400902

doi:10.6342/NTU202400902

doi:10.6342/NTU202400902

表次

doi:10.6342/NTU202400902

doi:10.6342/NTU202400902

第一章 緒論

1.1 研究背景

隨著硬體計算能力的提升，基於神經網路的機器學習模型也高速發展。時至今日，在各個應用中神經網路已經能產出人類認可的結果，並以各種產品的形式與我們的生活緊密結合。在影像處理方面被應用在智慧型手機的計算攝影以及影像編輯中，擴散模型(Diffusion model)近期的成果也超越了以往大眾的預期。在自然語言處理的領域，ChatGPT 在各項任務中傑出的表現，也使得大型語言模型的訓練成為了各公司競逐的目標，再度掀起了一波人工智慧的熱潮。

為了提升模型表現，神經網路的架構和功能也逐漸變得複雜，其計算需求也飛速成長。權重修剪(Weight pruning)與神經網路量化(Quantization)透過降低模型的精度，犧牲模型的精確度，以降低推論運算成本。神經網路權重二值化(Binarization)即是在量化方面做到極致的結果。此外，神經網路架構搜尋(Neural architecture search, NAS)能夠自動化調整、設計架構。搜尋所得之架構能在更低的計算成本下超越人工設計架構之表現，也能根據不同的硬體資源限制修改模型大小，適應不同的計算裝置。

在需要穩定、隱私保護、低延遲的任務上，邊緣運算相比雲端運算會更為合適。然而相對於雲端伺服器，邊緣裝置的計算資源有限。因此上述的研究對於邊緣運算至關重要，使我們能更快速、有效率地在邊緣裝置上進行神經網路推論的運算。

doi:10.6342/NTU202400902

1.2 論文動機與目標

目前在神經網路二值化上已經有非常多的研究，但是這些研究仍有不足之處。例如，[1]雖然被後續多個研究採用，但是需要在每層卷積層加入 fp32 skip-connection，不利於硬體運算。[2]雖然在二值化後能夠維持不錯的準確率，但在網路架構上需要針對二值化做人工的設計、調整。此外，大多數的研究仍然使用卷積神經網路在影像辨識的應用來評估二值化表現。

本研究的重點在於提出適合硬體計算的二值化權重神經網路量化方法。在沒有人工設計、調整模型下，透過各種技巧以及加入遲滯(Hysteresis)提高二值化量化感知訓練時的穩定性，盡可能維持量化前之準確率。並且透過結合 TPC-NAS 演算法進行自動架構搜尋，進一步縮短二值化權重網路與原始網路之表現差距。

除此之外，也將這個二值化與架構搜尋方法應用於多種模型與應用上，證明這個方法的泛用性。使用相同的流程，不需要人工設計架構，即可應用在基於卷積神經網路與 transformer 之模型上。以便在硬體上達到低功耗、低複雜度之推論運算。

doi:10.6342/NTU202400902

1.3 論文組織與貢獻

本篇論文共分為六章，第一章簡短說明研究背景、研究動機及目標，與論文組織與貢獻。第二章會介紹二值化神經網路的基本訓練方式，以及所使用到的訓練技巧。第三章介紹在二值化神經網路訓練中以 Hopfield 模型作為理論基礎加入遲滯(hysteresis)，以改善訓練穩定性以及提高模型表現。第四章將說明透過 TPC-NAS 改善模型架構，降低量化後正確率之耗損，並記錄在多種應用上的實驗結果。第五章則在硬體層面分析二值化神經網路推論，並進一步做乘加運算的近似。第六章總結本論文，並提出未來展望。

本篇論文的主要貢獻如下：

- 在二值化神經網路訓練中加入遲滯，在不影響推論計算成本下，改善訓練穩定性並提高模型表現。

- 結合 TPC-NAS 改善既有模型架構，降低量化後模型與量化前表現差距。

- 在二值化神經網路運算中提出近似電路，進一步降低推論計算面積與功耗。

doi:10.6342/NTU202400902

doi:10.6342/NTU202400902

第二章 二值化神經網路訓練簡介

本章節介紹將神經網路量化至二值化權重的訓練方法，以及為了維持量化後準確率，在量化感知訓練中採用的各種技巧。

2.1 二值化神經網路量化

本章節介紹訓練二值化神經網路之作法，以及我們在量化時對權重以及 activation 採用的量化函數。

在神經網路量化中，除了將權重以及 activation 量化為定點數型態，方便硬體運算之外，我們還會透過 scale 與 zero point 這兩個浮點數的參數協助我們進行線性量化(Linear quantization)。

$$x_q = \text{clamp}\left(\text{round}\left(\frac{x}{s} + z\right), q_{min}, q_{max}\right)$$

$$\text{clamp}(x, a, b) = \begin{cases} a, & x < a \\ x, & a \le x \le b \\ b, & b < x \end{cases}$$

式(2.1)

式(2.1)為線性量化的定義，其中x_q為量化後的值，x為未量化的值。s, z分別是前面提到的 scale 以及 zero point。q_{min}, q_{max}則是依據目標精度定點數最大最小值決定的量化後的整數範圍。

其中 scale 以及 zero point 能讓我們調整輸入的動態範圍，使其更接近量化後的範圍。通常這兩個參數會在一整層或是一整個通道進行共用，降低傳輸參數時的記憶體用量。

doi:10.6342/NTU202400902

在量化模型時，我們讓網路中每一層的權重在輸出的各個通道使用各自的 scale (per-channel scale)，並且將 zero point 設為 0。對於 activation 則讓整個張量共用 scale (per-tensor scale)以及 zero point。使用 per-channel scale 能夠個別調整每個 channel 的動態範圍，量化後的模型表現會比 per-tensor scale 來得好。

根據上面的量化設定，當量化後的模型在硬體推論時，能夠將運算拆分為整數部分與浮點數部分的運算，如式(2.2)所示：

$$
\begin{aligned}
\mathbf{Y} &= \mathbf{WX} + \mathbf{B} \\
&= (\mathbf{S_w W_q}) \left[s_x \left(\mathbf{X_q} - z_x \right) \right] + \mathbf{B} \\
&= \left(s_x \mathbf{S_w} \right) \left(\mathbf{W_q X_q} - \mathbf{W_q} z_x \right) + \mathbf{B} \\
&= \left(s_x \mathbf{S_w} \right) \left(\mathbf{W_q X_q} \right) + \left(\mathbf{B} - s_x z_x \mathbf{S_w W_q} \right) \\
&= \left(s_x \mathbf{S_w} \right) \left(\mathbf{W_q X_q} \right) + \mathbf{B}'
\end{aligned}
$$

式(2.2)

表 2.1　量化模型運算式(2.2)參數定義

	Full precision (fp32)	Quantized (integer)
Input	$\mathbf{X}$	$\mathbf{X_q}$
Output	$\mathbf{Y}$	-
Weight	$\mathbf{W}$	$\mathbf{W_q}$
Bias	$\mathbf{B}$	-
Weight scale (per-channel)	$\mathbf{S_w}$	-
Input scale (per-tensor)	s_x	-
Input zero point (per-tensor)	z_x	-

式(2.2)是將神經網路運算以矩陣乘法形式表現。線性層可以直接對應到矩陣乘法，卷積層則是可以透過 image to column 的方式重排為矩陣乘法。根據前述線性量化之定義式(2.1)以及矩陣運算之線性，我們可以將推論運算拆成紅色的整數部分以及剩餘的浮點數。其中藍色部分的值在訓練完畢後是定值，因此可以在推論前事先算出，簡化成$\mathbf{B}'$。

doi:10.6342/NTU202400902

$$\begin{aligned}\mathbf{Y} = s_y\left(\mathbf{Y_q} - z_y\right) &= \mathbf{WX} + \mathbf{B} \\ &= (s_x\mathbf{S_w})(\mathbf{W_qX_q}) + \mathbf{B}' \\ \mathbf{Y_q} &= \frac{s_x\mathbf{S_w}}{s_y}\mathbf{W_qX_q} + \left(\frac{\mathbf{B}'}{s_y} + z_y\right) \\ &= \mathbf{S_{merge}}\mathbf{W_qX_q} + \mathbf{B_{merge}}\end{aligned}$$

式(2.3)

表 2.2 量化模型運算式(2.3)參數定義

	Full precision (fp32)	**Quantized (integer)**
Output	**Y**	$\mathbf{Y_q}$-
Output scale (per-tensor)	s_y	-
Output zero point (per-tensor)	z_y	-

根據式(2.2)的結果，我們可以得到重新量化(re-quantize)的計算式(2.3)。其中$\mathbf{S_{merge}}$與$\mathbf{B_{merge}}$在訓練完畢後皆為定值，因此主要的運算為中間的整數矩陣乘法，接著再乘上$\mathbf{S_{merge}}$並加上$\mathbf{B_{merge}}$，得到量化後的輸出傳給下一層計算。

如此一來在計算時主要傳遞整數部分的資料，只有在網路的第一層和最後一層會需要使用 scale 以及 zero point 進行整數和浮點數之間的轉換。在中間的部分只需要計算整數部份以及輸出時進行重新量化。

此外因為整個過程仍為線性，因此位於線性層或卷積層之後的正規化層(normalization)以及簡單的激活函數(activation function)可以合併計算。常見的合併運算有 Conv + BN, Conv + ReLU, Conv + BN + ReLU 等。

doi:10.6342/NTU202400902

2.1.1 量化感知訓練簡介

神經網路量化主要有兩種做法來決定先前提到的 scale 以及 zero point，分別是訓練後量化(Post training quantization, PTQ)以及量化感知訓練(Quantization aware training, QAT)。

訓練後量化是在訓練完 fp32 模型後，根據訓練所得之權重計算出使量化誤差最小化的 scale 以及 zero point。由於從權重無法直接得出 activation 的分布，通常會使用部分的訓練資料進行校準(calibration)，實際將資料輸入模型中，計算每一層 activation 的資料分布，以統計方式得出適當的 activation scale 及 zero point。

訓練後量化只牽涉到模型的推論，不需要重新調整權重、訓練模型，因此可以在短時間內得到量化後的模型。然而這種做法的資料分布受限於預訓練權重，而預訓練時並沒有考慮到量化時的影響，因此最終得到的量化後模型的表現相較原始的模型通常有較大的差距。

為了進一步縮短量化前後模型的表現差距，量化感知訓練重新訓練模型，並在訓練過程中加入量化對模型的影響，同時動態調整權重以及 activation 之 scale 以及 zero point。

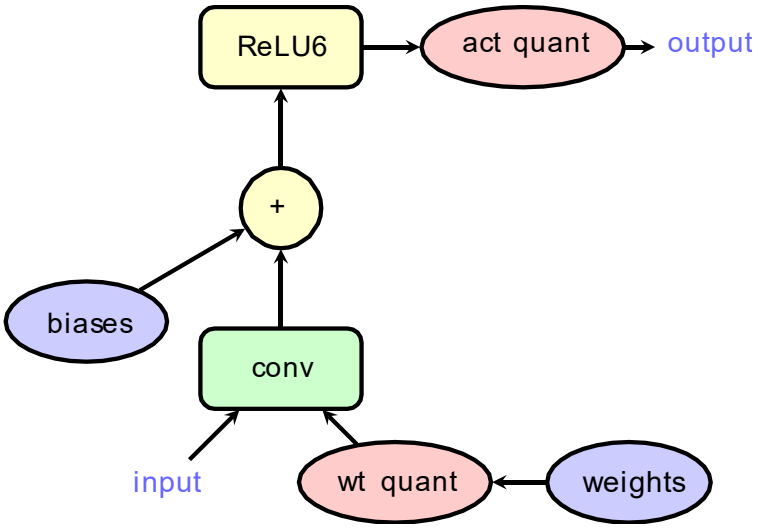


圖 2.1 量化感知訓練模擬量化卷積示意圖[3]

doi:10.6342/NTU202400902

圖 2.為量化感知訓練應用於卷積層之示意圖。在訓練時會保留 fp32 的權重，用於累積梯度之更新量，稱為權重的 fp32 master copy。在計算會動態對權重以及 activation 量化，但是並不會將整數部分與 scale, zero point 分開計算，而是組合成 fp32 的值x_f繼續往下傳遞，如式(2.4)所示。

$$\begin{aligned} x_f &= (x_q - z_x)\, s_x \\ &= \left[\text{clamp}\left(\text{round}\left(\frac{x}{s_x} + z_x\right), q_{min}, q_{max}\right) - z_x\right] s_x \end{aligned}$$

式(2.4)

組合成單一的值後在計算圖上可以看成 fp32 master copy 經過式(2.4)的函數輸出成量化後的權重，進行卷積運算。如此一來可以使用與訓練普通神經網路相同的反向傳播(Back propagation)，透過微分之連鎖率(chain rule)逐步計算損失函數對各層權重的梯度，進而用梯度下降法(Gradient descent)更新權重之 fp32 master copy、scale，以及 zero point。

在計算式(2.4)之梯度時，其中的clamp以及round之組合函數並非是連續可微分的。即使運用分段計算微分，round函數還是處處微分為 0，使得梯度經過量化後無法正常傳播。為了解決這個問題，在量化時會人為定義量化函數之梯度計算方式。常用的梯度計算方式為 Straight-through estimator (STE) [4]，STE 的概念是將round函數視為是同一(identity)函數，因此微分= 1，相當於是將離散的值視為原本量化前連續的值來計算梯度。

在處理 activation 的部分與權重稍有不同，最早期的量化感知訓練中會使用統計方式決定 scale 與 zero point。透過指數移動平均(Exponential moving average)計算多次推論下資料的最大值及最小值，對應到量化的範圍q_{min}, q_{max}，可以算出 scale s以及 zero point z，如式(2.5)。

$$\begin{cases} s &= \dfrac{x_{max} - x_{min}}{q_{max} - q_{min}} \\ z &= q_{min} - \text{round}\left(\dfrac{x_{min}}{s}\right) \end{cases}$$

式(2.5)

doi:10.6342/NTU202400902

2.1.2 權重二值化

權重二值化的部分是基於前一章節的量化感知訓練而進行，目標是將權重量化至±1，並找出適當的 per-channel scale。

最主要的改變是式(2.4)中的round函數會改為使用符號(sign)函數。與一般定義稍有不同，這邊定義符號函數為式(2.6)，在0的部分定義輸出為−1，避免權重量化結果產生三種值。

$$\operatorname{sign}(x) = \begin{cases} 1, & x \geq 0 \\ -1, & \text{otherwise} \end{cases}$$

式(2.6)

至於式(2.4)中的clamp函數，在許多二值化權重的論文中都將量化後的值再做一次clamp至[−1, 1]的範圍內，使得超出這個範圍的權重對應到的梯度變為0。

我們在權重二值化的部分主要使用整流箝位單元(Rectified Clamp Unit, ReCU)[5]作為基礎。ReCU 觀察到 fp32 master copy 在訓練時主要會是平均為0的 Laplace 分布。因為二值化時使用式(2.6)，只需要跨過 0 的邊界即可使量化後的值變號，而 fp32 master copy 落在 distribution tail 的值距離 0 較遠，因此比較難跨越邊界改變量化後的正負號。如此一來就相當於存在許多不太會變動的權重，在[5]之中被稱為 dead weight。

為了解決這個問題，[5]在權重二值化使用了整流箝位單元，主要的概念是根據 fp32 master copy 的分位數(quantile)作為式(2.4)中clamp函數的最大最小值，如式(2.7)。

$$\operatorname{ReCU}(w) = \max\left(\min\left(w, Q_\tau\right), Q_{1-\tau}\right) = \operatorname{clamp}\left(w, Q_{1-\tau}, Q_\tau\right)$$
$$0.5 < \tau \leq 1$$

式(2.7)

doi:10.6342/NTU202400902

在使用了 ReCU 後，假設 fp32 master copy 的分布為$Laplace(0, b)$，b為 Laplace 分布中的 scale 參數，這裡以估計的方式計算$b = \text{Mean}(|\mathbf{W}|)$。在量化時採用 scale 為$\alpha = \mathbb{E}(|\text{ReCU}(W)|)$，此時可以計算量化誤差(Quantization error)的期望值：

$$\begin{aligned}\text{QE}(\tau) &= \int_{-\infty}^{\infty} f(w)\left(w - \alpha\,\text{sign}\,(w)\right)^2\, dw \\ &= (a-b)^2\left(1+\exp\left(-\frac{Q_\tau}{b}\right)\right) + b^2 - \left((b+Q_\tau)^2 - 2\alpha Q_\tau\right)\exp\left(-\frac{Q_\tau}{b}\right) \\ &+ 2(1-\tau)\left(Q_\tau - \alpha\right)^2\end{aligned}$$

式(2.8)

式(2.8)根據上述的設定，會得到量化誤差的期望值是τ的函數，且這是一個在$\tau \approx 0.82$達到最小值的凸函數。

這裡另外計算權重的資訊熵(Information entropy)來衡量權重的多樣性(diversity)，權重多樣性越高的 BNN，表現越好。根據前面的定義可以計算資訊熵式(2.9)：

$$\begin{aligned}H(\tau) &= -\int_{-Q_\tau}^{Q_\tau} f(w)\ln\left(f\left(w\right)\right)\, dw - \sum_{|w|=Q_\tau} f(w)\ln\left(f\left(w\right)\right) \\ &= 2(\ln b + 1)\tau + \ln\frac{2}{b} - 1\end{aligned}$$

式(2.9)

雖然量化誤差在$\tau \approx 0.82$達到最小值，但是這只是讓量化後的權重與 fp32 master copy 更接近。繼續增加τ根據式(2.9)能夠繼續使資訊熵提高，所以[5]使用指數排程將τ從 0.85 逐漸增加至 1，取得量化誤差與資訊熵的平衡。

此外因為權重 fp32 master copy 分布在訓練過程中會逐漸改變，在權重二值化前先做標準化，可以使得式(2.9)的資訊熵維持在較高的值，提高量化後模型的表現。

doi:10.6342/NTU202400902

2.1.3 習得步長量化(Learned Step Size Quantization)

在 activation 的部分我們使用 LSQ+[6]進行量化。LSQ+的主要概念是透過梯度下降(Gradient descent)的方式更新 scale 以及 zero point。在量化的參數設定上與式(2.1)有些許不同，是以式(2.10)的方式實現：

$$x_q = \text{clamp}\left(\text{round}\left(\frac{x-z}{s}\right), q_{min}, q_{max}\right)$$

式(2.10)

不過只需要使$z' = -\frac{z}{s}$即可轉換為與式(2.1)相同的設定。同理，式(2.4)會變成式(2.11)的形式。

$$x_f = sx_q + z$$

式(2.11)

結合式(2.10)以及式(2.11)，可以得到式(2.12)：

$$x_f = \begin{cases} q_{min}, & \frac{x-z}{s} < q_{min} \\ \text{round}\left(\frac{x-z}{s}\right), & q_{min} \le \frac{x-z}{s} \le q_{max} \\ q_{max}, & q_{max} < \frac{x-z}{s} \end{cases}$$

式(2.12)

將上式分別對 scale s以及 zero point z微分，即可使用梯度下降法學習兩者的值，如式(2.13)：

$$\frac{\partial x_f}{\partial s} = \frac{\partial x_q}{\partial s}s + x_q = \begin{cases} q_{min}, & \frac{x-z}{s} < q_{min} \\ -\frac{x-z}{s} + \text{round}\left(\frac{x-z}{s}\right), & q_{min} \le \frac{x-z}{s} \le q_{max} \\ q_{max}, & q_{max} < \frac{x-z}{s} \end{cases}$$

$$\frac{\partial x_f}{\partial z} = \frac{\partial x_q}{\partial z}s + 1 \ = \begin{cases} 0, & q_{min} \le \frac{x-z}{s} \le q_{max} \\ 1, & \text{otherwise} \end{cases}$$

式(2.13)

doi:10.6342/NTU202400902

在使用梯度下降之前，需要先對 scale 和 zero point 進行初始化。

在較深的網路，在 fp32 預訓練權重加入 activation 量化後，在不好的 scale 以及 zero point 條件下，會使得 activation 在前向傳播(forward propagation)的過程中逐漸偏移 fp32 模型所產生的輸出，導致訓練無法從好的起點開始訓練，有時候甚至會產生 NaN。

為了解決這個問題，我們採用三階段的調整 scale 以及 zero point。在第一個階段中，直接以每層的 activation 最大最小值計算出 scale 和 zero point。重複多次迭代後，網路中的 scale 和 zero point 會逐漸修正成適合預訓練參數的大小，並且錯誤的 activation 也會被修正。通常在第一個階段並不會跑完整個訓練資料集，因此在第二個階段中我們計算 activation 最大最小值的指數移動平均，將 scale 以及 zero point 改為更具有統計意義的值。

當統計完成後進入第三個階段，此時不再觀察最大最小值，而是使用梯度下降法更新 scale 和 zero point。

在 scale 以及 zero point 的學習率(learning rate)的設定則是參考原始的 LSQ[7]，設為$\frac{1}{N_f Q_{max}}$，其中N_f為 activation 張量的大小。這樣設定是考慮了兩個因素：當精度越高，量化的區間越多，因此學習率要隨之調小，所以引入Q_{max}。當要量化的數值越多，計算 gradient 時會有越多項加總，因此學習率也要調小，引入N_f來調整學習率。

doi:10.6342/NTU202400902

2.2 二值化神經網路訓練技術

在上個章節中描述了二值化量化感知訓練如何進行，以及量化時採用的量化函數。這個章節中主要是說明在二值化神經網路中我們使用了那些技術來進一步提高二值化後的模型表現。

2.2.1 Adam 最佳化演算法

Algorithm: Gradient Descent with Adam Optimizer

Data: Step size: α, Exponential decay rate: β_1, β_2, Objective function: $\mathtt{f}(\theta)$, Time step: $\mathtt{dt}$, Initial parameter: θ_0

Result: Resulting parameters $\theta_\mathtt{t}$

$m_0 \leftarrow 0$; // Initialize 1st moment vector

$v_0 \leftarrow 0$; // Initialize 2nd moment vector

$t \leftarrow 0$;

while $\theta_\mathtt{t}$ *not converged* **do**

  $\mathtt{t} \leftarrow \mathtt{t} + 1$;

  $\mathtt{g_t} \leftarrow \nabla_\theta \mathtt{f_t}(\theta_{\mathtt{t}-1})$; // Compute gradients

  $\mathtt{m_t} \leftarrow \beta_1 \cdot \mathtt{m_{t-1}} + (1-\beta_1) \cdot \mathtt{g_t}$;

  $\mathtt{v_t} \leftarrow \beta_2 \cdot \mathtt{v_{t-1}} + (1-\beta_2) \cdot \mathtt{g_t}^2$;

  $\widehat{\mathtt{m_t}} \leftarrow \frac{m_t}{1-\beta_1^t}$; // Compute bias-corrected 1st moment estimate

  $\widehat{\mathtt{v_t}} \leftarrow \frac{v_t}{1-\beta_2^t}$; // Compute bias-corrected 2nd moment estimate

  $\theta_\mathtt{t} \leftarrow \theta_{\mathtt{t}-1} - \alpha \cdot \frac{\widehat{\mathtt{m_t}}}{\sqrt{\widehat{\mathtt{v_t}}} + \epsilon}$; // Update parameters

end

圖 2.2 Adam 最佳化演算法[8]

Adam (Adaptive moment estimation)[8]是根據歷史梯度資訊個別調整每個參數的更新量的演算法，相當於是個別調整每個參數的學習率。

在進行梯度下降時，如果落在損失函數較陡的區域，將學習率調小可以避免越過局部極小值。落在較平緩的區域時則可以將學習率調大加速收斂。Adam 即是透過統計梯度的一、二階矩(first, second moment)調整每個參數的更新量。

doi:10.6342/NTU202400902

對於 fp32 模型訓練來說，雖然 Adam 在學習率沒有仔細調整的情況下能有不錯的表現，但是如果使用隨機梯度下降法(Stochastic gradient descent, SGD)結合一階矩(first moment)，在學習率調整得好的情況下，最終收斂的模型表現會比 Adam 好。

但是在二值化神經網路訓練中，使用 Adam 往往比 SGD 能得出準確率更高的網路。因此我們也使用 Adam 進行二值化神經網路訓練。

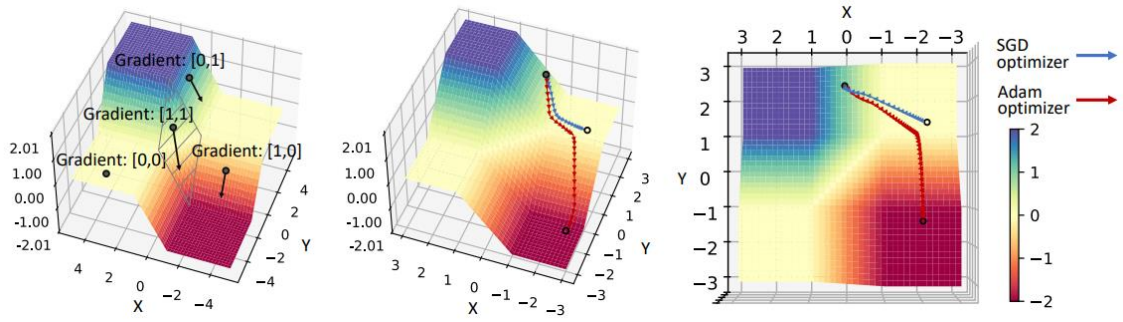


圖 2.3　Adam 與 SGD 於 BNN 訓練 Loss landscape 之視覺化[9]

圖 2.3 是[9]探討 Adam 訓練效果優於 SGD 原因所畫的圖，分別畫出了兩種演算法在 loss landscape 上的軌跡。這邊採用的量化函數是基於式(2.1)，使用符號函數(sign)量化為±1後，做clamp(x, −1, +1)將範圍外的梯度設為 0。如此一來相比原先 fp32 平滑的 loss surface，二值化訓練下的 loss surface 會更不平滑，存在較多平坦的區域。從圖上可以看出，在平坦的區域下只有一階矩的 SGD 無法有效降到損失函數的局部極小值。而加入了二階矩的 Adam 能夠在平坦的區域放大更新量，進而使損失函數降到極小值。

這個現象導致 Adam 能夠公平的訓練每個參數，SGD 則容易有許多參數卡在平緩的區域，只有部分參數有在更新。[9]額外觀察了實際訓練出的權重，使用 SGD 的確有較多的 fp32 master copy 落在[−1, +1]之外。並且每個通道權重的絕對值平均的標準差也較大，顯示每個通道並沒有公平地被訓練。

doi:10.6342/NTU202400902

2.2.2 漸進訓練(Progressive Training)

在二值化權重量化感知訓練中，一次將所有權重進行二值化會導致 loss surface 如前一章節提到的變得不平滑，這使得二值化權重訓練更加困難。

VAQF[10]為了解決這個問題，引入了漸進訓練(progressive training)的作法。漸進訓練的作法是在量化感知訓練的過程中逐漸提高二值化權重的比例，直到完全將所有權重二值化。

式(2.4)的量化函數將原始的浮點數權重x量化為整數與浮點數結合的x_f。要實現漸進訓練，我們引入qr的參數，代表權重量化比例(quantization ratio)。實際在訓練時會依據式(2.13)進行訓練：

$$x' = M_{qr} \cdot x_f + (1 - M_{qr}) \cdot x, \;\; 0 \leq qr \leq 1$$

式(2.14)

其中M_{qr}是針對每個權重，依據隨機數是否小於qr而決定是否要進行量化。如果要進行量化則為 1，否則為 0。

實際上訓練時會線性增加qr，直到經過預先決定的迭代次數使得$qr = 1$。目前的設定上網路中所有層都共用單一qr，根據實驗結果，個別指派每層獨立的qr並分開調整並沒有太大的差異。為了降低超參數(hyperparameter)數量，這裡只使用統一的qr。

使用漸進訓練的第二個好處是當我們使用 fp32 預訓練參數進行初始化時，最初$qr = 0$會完全與未量化模型相同，因此不會因為量化導致損失函數的值一次上升太多，導致初始化狀態偏離預訓練的模型。

doi:10.6342/NTU202400902

2.2.3 知識蒸餾(Knowledge Distillation)

知識蒸餾(Knowledge distillation, KD)是常被用於模型壓縮或是將訓練好的模型轉移到另一個資料集上的技術。知識蒸餾的特徵是會存在 teacher 以及 student 模型，因此也常被稱為 teacher-student learning。

最原始的知識蒸餾是根據分類問題所建構[11]，在神經網路進行分類時，會藉由最後一層輸出經過softmax函數產生每個類別的機率預測，透過在softmax中加入溫度T參數，我們可以控制模型輸出分布的陡峭程度。式(2.15)為加入溫度後透過softmax計算第i個類別的機率。

$$p_i = \frac{\exp\left(x_i/T\right)}{\sum_j \exp\left(x_j/T\right)}$$

式(2.15)

在分類訓練時在有標註的資料集上會計算 student 模型輸出與真實標註的交叉熵(cross-entropy)作為損失函數。在知識蒸餾的情況下損失函數會額外加入另一項 student 與 teacher 模型的差異。計算方式則是由 teacher 輸出溫度較高的 soft distribution 與 student 輸出的分布計算交叉熵。

雖然經過溫度較高的softmax後，teacher 輸出的信心水準會下降。但是相比於真實類別的標註，teacher 輸出的 soft distribution 具有更多資訊。真實標註是 one-hot 編碼的向量，我們只知道真實的類別是哪一個。而 soft distribution 中每個類別都給出一個機率，透過機率的大小關係，我們可以得出在所有類別中哪些比較接近，哪些類別則差異較大。因此在損失函數中加入 student 與 soft distribution 的差異可以提高訓練效果。

doi:10.6342/NTU202400902

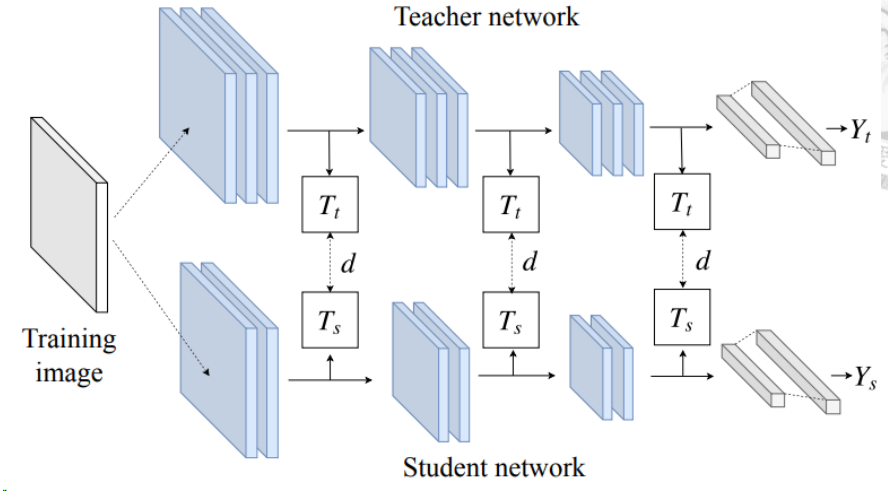


圖 2.4 Feature distillation 訓練示意圖[12]

除了使用網路輸出計算 teacher-student 損失函數之外，圖 2.4 是使用網路中間層經過轉換後計算損失函數。訓練的方法一樣是將額外的知識蒸餾損失加入原本的損失函數中。式(2.16)為 feature distillation 之損失函數，分別將 teacher, student 中間輸出$\boldsymbol{F}_t, \boldsymbol{F}_s$經過$\boldsymbol{T}_t, \boldsymbol{T}_s$轉換後，計算兩者間距離$d(\cdot)$。

$$L_{distill} = d\left(T_t\left(\boldsymbol{F}_t\right), T_s\left(\boldsymbol{F}_s\right)\right)$$

式(2.16)

最簡單的轉換只處理 teacher 與 student 中間輸出的維度差異，在卷積神經網路中可以透過1×1卷積進行通道維度的轉換。計算損失函數的部分因為不再對應到機率分布，因此會改為使用均方誤差(Mean square error, MSE)計算 teacher 與 student 輸出的差異。

在二值化神經網路訓練下，我們選擇 teacher 為預訓練的高精度神經網路，student 則會是相同架構的二值化神經網路。因為網路架構相同，在轉換的部分可以使用同一(identity)函數進行。

doi:10.6342/NTU202400902

在較為複雜的任務上訓練二值化網路時，使用兩步驟訓練可以提高訓練後的二值化神經網路正確率。[13]採用的訓練分為兩步：

1. 使用 fp32 模型作為 teacher，student 則是相同架構但量化 activation 的模型。
2. 使用前一步驟的 student 作為 teacher，student 的部分對權重和 activation 皆做了二值化。

分成兩部訓練可以避免同時加入權重及 activation 量化後帶來的巨幅誤差。降低訓練的困難程度。

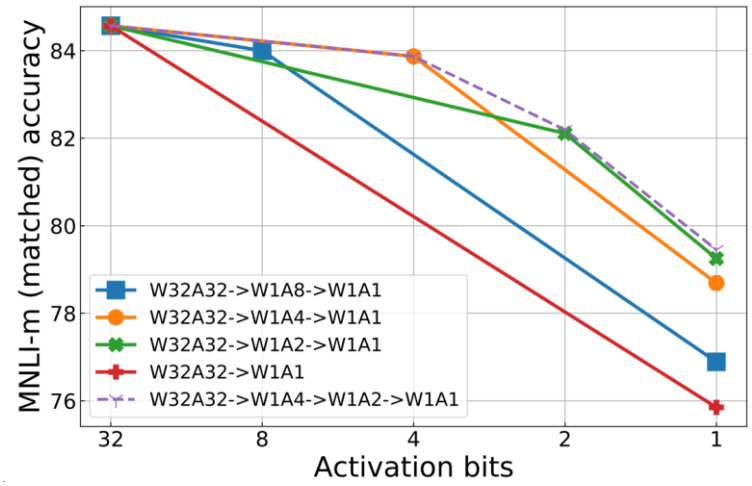


圖 2.5 Multi-distillation 示意圖[14]

BiT[14]則在這個基礎上進一步延伸提出 multi-distillation，從 fp32 模型訓練二值化權重、multi-bit activation 模型，再逐步以知識蒸餾的方式訓練，降低 activation 精度，直到到達二值化 activation。

圖 2.5 是使用 multi-distillation 在 GLUE benchmark[15]中的 MNLI 任務圖 2.5 Multi-distillation 示意圖[14]訓練 BERT[16]的結果。可以看到不同的 multi-distillation 路徑最終到達的 1W1A 模型之準確率皆不相同，路徑上步驟越多，得到的 1W1A 準確率越高。

doi:10.6342/NTU202400902

逐步使用多次知識蒸餾訓練可以降低每次訓練中，teacher 與 student 之間的差異。因此分成越多步驟，最終的準確率越高。

但是每次多分出一個步驟就需要花費更多訓練時間，在 BERT 這種大模型下影響更明顯。再來細分步驟帶來的準確率提昇會有邊際效應，所以 BiT 最終是選擇使用兩步驟的訓練，先從 fp32 模型訓練 1W2A，再從 1W2A 訓練 1W1A。

然而即便使用了 multi-distillation，1W1A 的 BiT 訓練最終的準確率還是跟原始的 BERT 相差了 10.4%。我們還是需要使用較高精度的 activation 維持正確率，因此我們沒有採用 multi-distillation 進行訓練。

2.3 第二章總結

本章節介紹了二值化神經網路量化感知訓練的細節以及提升準確率的技巧。

總結來說，我們訓練時權重量化主要使用了符號(sign)函數進行二值化，並且用 STE 計算梯度。為了維持準確率則使用了高精度的 activation，這裡採用 LSQ+學習對應的 per-tensor scale 及 zero point。訓練時主要採用 Adam 進行權重更新，加入了漸進訓練逐漸將權重二值化，並且使用 fp32 模型作為 teacher 進行知識蒸餾。在較為複雜的訓練中，如物件偵測時，會採用兩步驟訓練降低訓練困難度。

在下個章節中會介紹在上述設定之外，我們提出遲滯用來進一步改善二值化訓練的理論以及實驗結果。

doi:10.6342/NTU202400902

第三章 結合遲滯與二值化神經網路

本章節描述在神經網路二值化訓練中加入遲滯的方法以及提高訓練穩定性的效果。最後將神經網路二值化轉換成組合的最佳化問題，透過 Hopfield 模型來達成最佳化，為遲滯訓練提供理論基礎。

3.1 遲滯與二值化神經網路權重

在神經網路二值化訓練中，可以觀察到準確率會有震盪的情況，在兩個 epoch 間的準確率有大幅度的變化，模型無法繼續收斂到更好的準確率。

為了更好地觀察這個現象，這邊將問題以及模型做簡化。最佳化問題改為三維空間中多個點的二元分類，分類的基準是隨機產生的平面。在模型的部分採用兩層線性模型，隱藏層的維度設定為 9。

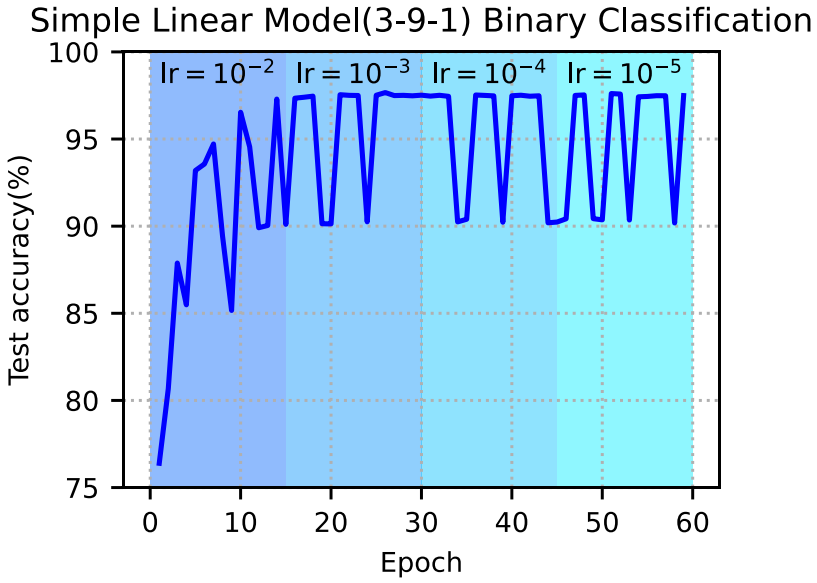


圖 3.1 簡單線性模型進行二元分類於測試集之準確率震盪情形

doi:10.6342/NTU202400902

由圖 3.1 可以看到在訓練過程中，模型於測試資料集之分類準確率會有震盪的情況發生。實際觀察網路訓練過程中的權重以及梯度，可以發現在網路第一層 $3\times9=27$個參數中，大部分參數對應的 fp32 master copy 的數量級在10^{-1}，但是其中一個參數的數量級為10^{-7}。這導致了在多次更新後，只有其中一個參數的正負號改變。當$+1,-1$兩個狀態下的梯度互為反向時，會使得二值化後的權重在$+1,-1$間切換，產生準確率震盪的情況。

因為計算梯度時採用了 Straight-through estimator (STE) [4]，導致梯度只受二值化後的權重影響。降低學習率僅會改變 fp32 master copy 的值，因此當$+1,-1$跨過損失函數的局部極小值時，則會產生互為反向的梯度。這個問題無法透過降低學習率來解決，從圖 3.1 也可以發現降低學習率後震盪的問題還是持續發生。

為了解決這個問題，我們在二值化訓練中加入遲滯。將權重二值化的量化函數改為斯密特觸發器(Schmitt trigger)即可在二值化訓練中引入遲滯的效果。

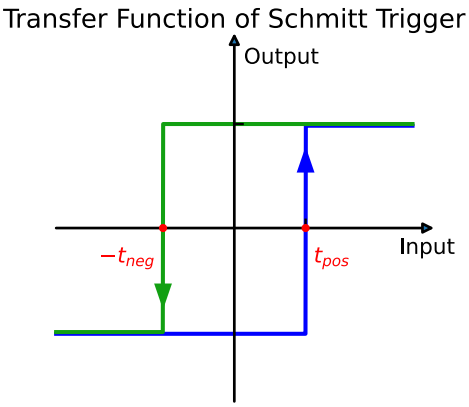


圖 3.2　斯密特觸發器之轉移函數

斯密特觸發器的轉移函數如圖 3.2 所示，遲滯相當於有記憶效應，依據前一次量化結果在兩條曲線中做選擇。而組成轉移函數的兩條曲線則各是偏移後的 sign 函數，$-t_{neg}, t_{pos}$兩個閾值控制了 sign 函數的偏移。

doi:10.6342/NTU202400902

以w_t代表目前的 fp32 master copy，$w_{q,t-1}, w_{q,t}$代表前一次更新與目前量化後的權重，可以將斯密特觸發器寫成如式(3.1)的函數：

$$w_{q,t} = f(w_{q,t-1}, w_t) = \begin{cases} \text{sign}(w_t - t_{pos}), & w_{q,t-1} = -1 \\ \text{sign}(w_t + t_{neg}), & w_{q,t-1} = +1 \end{cases}$$

式(3.1)

式(3.1)可以進一步化簡為：

$$w_{q,t} = f(w_{q,t-1}, w_t) = \begin{cases} -w_{q,t-1}, & w_{q,t-1} = -1 \text{ and } w_t > t_{pos} \\ -w_{q,t-1}, & w_{q,t-1} = +1 \text{ and } w_t < -t_{neg} \\ w_{q,t-1}, & \text{otherwise} \end{cases}$$

式(3.2)

由式(3.2)可以看出只有當 fp32 master copy 超出閾值之外時，二值化後的權重才會改變，否則會維持前一次量化後的值。這代表當有部分權重在 0 附近震盪時，並不會超出閾值之外，在二值化後權重並不會改變，也就不會對梯度產生影響。使得其他數量級較大的權重有機會能夠更新，進而脫離震盪。

3.2 Hopfield 模型

Hopfield 模型[17]是一種遞歸神經網路(Recurrent neural network, RNN)，其神經元為二值化的數值或是連續變數。

在模型參數滿足條件下，對 Hopfield 模型之神經元進行迭代更新，可以保證模型系統的能量會隨著時間逐步降低，最終收斂到能量函數的局部極小值。

此特性使得 Hopfield 模型能夠作為密集關聯記憶體(Dense associative memory)，或是用於解組合的最佳化問題(Combinatorial optimization)。

離散 Hopfield 模型包含N個二值化的神經元V_i，由神經元j之輸出到神經元i輸入的強度W_{ij}，在每個神經元上有外部輸入I_i以及更新閾值U_i。

doi:10.6342/NTU202400902

在更新離散 Hopfield 模型時會使用式(3.3)做為更新規則：

$$V_{i,t} = \begin{cases} +1, & \text{if } \sum_j W_{ij} V_{j,t-1} + I_i > U_i \\ -1, & \text{otherwise} \end{cases}$$

式(3.3)

而其對應到的能量函數為式(3.4)：

$$E(\mathbf{V}) = -\frac{1}{2} \sum_i \sum_j W_{ij} V_i V_j - \sum_i I_i V_i + \sum_i U_i V_i$$

式(3.4)

在一次迭代更新後，其中一個神經元由V_i更新為V_i'，以$\mathbf{V}, \mathbf{V}'$表示更新前後的所有神經元。可以得到對應的能量變化式(3.5)：

$$\begin{aligned} \Delta E =& E(\mathbf{V}') - E(\mathbf{V}) \\ =& -\frac{1}{2} \left[\sum_{l,l\neq i} V_l W_{li} (V_i' - V_i) + \sum_{j,j\neq i} (V_i' - V_i) W_{ij} V_j + W_{ii} (V_i' V_i' - V_i V_i) \right] \\ & - I_i (V_i' - V_i) + U_i (V_i' - V_i) \end{aligned}$$

式(3.5)

令$\Delta V_i = V_i' - V_i$，權重矩陣$\mathbf{W}$滿足對角線為 0 且為對稱矩陣時，可化簡為

$$\Delta E = -\left(\sum_{j,j\neq i} W_{ij} V_j + I_i - U_i \right) \Delta V_i$$

式(3.6)

根據離散 Hopfield 模型的更新規則式(3.3)，可知式(3.6)等號右邊兩項必為同號，因此$\Delta E \leq 0$。再加上E存在下界，因此在迭代更新過程中離散 Hopfield 模型會逐漸降低能量，直到到達局部極小值。

doi:10.6342/NTU202400902

連續的 Hopfield 模型[18]則是基於類比電路模擬生物系統，所有神經元都加上了輸入電阻R_i與輸入電容C_i，使得神經元的值u_i會落後於其他神經元的輸出V_i。每個神經元的輸出都加上了放大器來實現激活函數(activation function)$V_i = g(u_i)$。透過在輸出與輸入端連接對應阻值的電阻$R_{ij} = W_{ij}^{-1}$，可以實現 Hopfield 模型的權重矩陣。完整的電路實現如圖 3.3 所示。

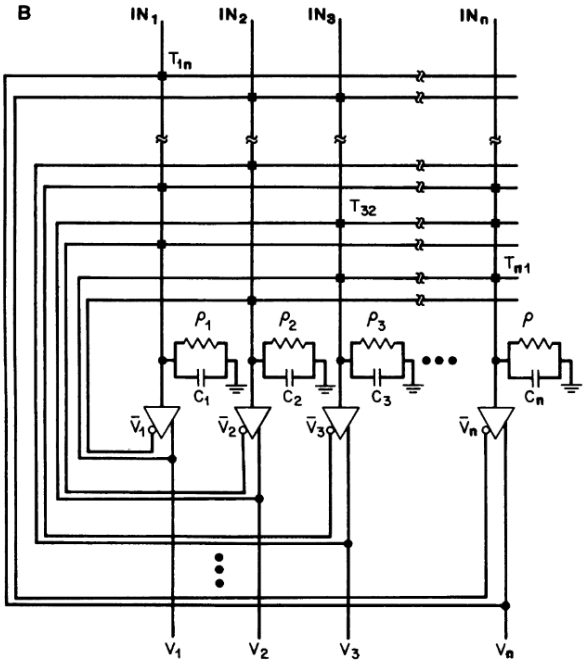


圖 3.3　連續 Hopfield 模型之類比電路實現[19]

對於以上電路，根據克希何夫電流定律(Kirchhoff current law)，可以得到式(3.7)的微分方程描述模型的狀態變化：

$$\begin{cases} C_i \dfrac{du_i}{dt} = \sum_j W_{ij} V_j + I_i - \dfrac{u_i}{R_i} \\ V_i = g_i(u_i) \end{cases}$$

式(3.7)

doi:10.6342/NTU202400902

連續 Hopfield 模型的能量函數定義為式(3.8)

$$E(\mathbf{V}) = -\frac{1}{2}\sum_i\sum_j V_i W_{ij} V_j - \sum_i I_i V_i + \sum_i \frac{1}{R_i}\int_0^{V_i} g_i^{-1}(V)\,dV$$

式(3.8)

將能量對於時間求導，可得式(3.9)

$$\begin{aligned}\frac{dE}{dt} &= -\sum_i \frac{dV_i}{dt}\left(\sum_j W_{ij}V_j + I_i - \frac{u_i}{R_i}\right)\\ &= -\sum_i C_i \frac{du_i}{dt}\frac{dV_i}{dt}\\ &= -\sum_i C_i \frac{dg_i^{-1}(V_i)}{dt}\frac{dV_i}{dt}\\ &= -\sum_i C_i g_i^{-1\prime}(V_i)\left(\frac{dV_i}{dt}\right)^2\end{aligned}$$

式(3.9)

因為激活函數與其反函數為單調遞增，因此加總的每一項皆為非負值，可得

$$\begin{aligned}&\frac{dE}{dt} \le 0\\ &\frac{dE}{dt} = 0 \iff \forall i,\ \frac{dV_i}{dt} = 0\end{aligned}$$

式(3.10)

式(3.10)顯示連續 Hopfield 模型的系統隨著時間演化會逐漸趨向能量的局部極小值，最終在該處停下。

此外如果使$V_i = g_i(\lambda u_i)$，式(3.8)第三項變為$\sum_i \frac{1}{\lambda R_i}\int_0^{V_i} g_i^{-1}(V)\,dV$。在$\lambda \to \infty$的情況下，該連續系統的能量會相等於離散系統的能量，兩者有相同的極值。可以將離散模型視為高增益(λ)的連續模型。

doi:10.6342/NTU202400902

根據式(3.8)，計算能量對V_i之梯度：

$$\begin{aligned}\nabla \mathbf{E}(\mathbf{v}) &= -\mathbf{W}\mathbf{v} - \mathbf{i} + \frac{\mathbf{1}}{\mathbf{r}}\mathbf{u} \\ &= -\mathbf{c}\frac{d\mathbf{u}}{dt}\end{aligned}$$

式(3.11)

可以得知神經元的更新是朝著能量梯度的反向進行，代表更新連續 Hopfield 模型相當於是在做梯度下降演算法(Gradient descent)，目標是降低系統能量。

3.2.1 Hopfield 模型用於最佳化

透過選擇適當的權重W_{ij}、外部輸入I_i，我們能夠將想要解的最佳化問題之限制以及目標函數轉換成 Hopfield 模型之能量函數。透過 Hopfield 模型的更新逐步降低能量，最終找到滿足限制並最小化目標函數的神經元之值。

以旅行推銷員問題(Traveling salesman problem, TSP)為例，TSP 是給定數個城市以及城市間的距離下，試圖找出最短且走過每個城市各一次，回到起始城市的路徑。根據所有路徑的總數，可以知道使用暴力搜尋的複雜度為O($N!$)，因此不太可行。使用 Hopfield 模型進行最佳化可以在很短的執行時間內找到不錯的解，使用不同的初始值重複多次搜尋有機會能夠找到最佳解。

$$\begin{aligned}E = &\overbrace{\frac{A}{2}\sum_x\sum_i\sum_{j\neq i}V_{x,i}V_{x,j}}^{\text{Visit each city once (row)}} + \overbrace{\frac{B}{2}\sum_i\sum_x\sum_{y\neq x}V_{x,i}V_{y,i}}^{\text{Visit 1 city at a time (col)}} \\ &+ \underbrace{\frac{C}{2}\left(\sum_x\sum_i V_{x,i} - N\right)^2}_{\text{Visit N cities in total}} + \underbrace{\frac{D}{2}\sum_x\sum_{y\neq x}\sum_i d_{x,y}V_{x,i}(V_{y,i-1}+V_{y,i-1})}_{\text{Minimize path length}}\end{aligned}$$

式(3.12)

TSP 相當於是找出N個城市的置換矩陣(Permutation matrix)$\mathbf{P}_{\boldsymbol{\pi}}$，矩陣中的元素$P_{\pi i,j}$代表了第 j 個拜訪的城市為城市i。給定城市數量為N，城市x, y間距離為$d_{x,y}$根據置換矩陣的限制以及 TSP 的目標可以寫出式(3.12)[20]的能量函數。

doi:10.6342/NTU202400902

透過比較係數可以得知式(3.12)對應到有N^2個神經元的 Hopfield 模型，而其對應到之權重矩陣以及外部輸入為式(3.13)[20]：

$$W_{xi,yj} = -A\delta_{x,y}(1-\delta_{i,j}) - B\delta_{i,j}(1-\delta_{x,y}) - Cn - Dd_{x,y}(\delta_{j,i-1}+\delta_{j,i+1})$$
$$I_{xi} = +Cn$$
$$\delta_{i,j} = \begin{cases} 1, & \text{if } i=j \\ 0, & \text{otherwise} \end{cases}$$

式(3.13)

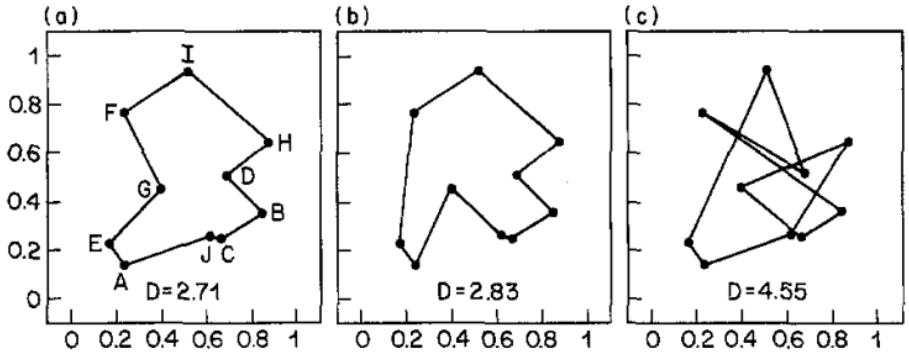


圖 3.4　TSP 最佳解以及 Hopfield 模型尋得之解[20]

圖 3.4 顯示了 TSP 問題的解。其中(a)為暴力搜尋而得之最佳解。(b)為連續 Hopfield 模型收斂到的解。(c)則是離散 Hopfield 模型收斂到的解。

可以觀察到在 TSP 這種離散的組合最佳化問題中引入連續變數進行最佳化，可以得到比單純使用離散變數更好的解。這是因為離散模型只能在合法的解，也就是的N^2維超立方體(Hypercube)的邊角上移動。而連續模型在合法的解之外，還提供了不是合法的解的中間狀態。這使得 Hopfield 模型的狀態演化能夠更平滑地進行。

doi:10.6342/NTU202400902

3.2.2 遲滯 Hopfield 模型

遲滯 Hopfield 模型(Hysteretic Hopfield model)[21]透過在激活函數中加入歷史狀態的影響在 Hopfield 模型中引入遲滯的效應。

對於解組合最佳化問題而言，遲滯 Hopfield 模型相比原始的 Hopfield 模型更常收斂到合法的解，並且能夠使用更少的迭代次數達到收斂。

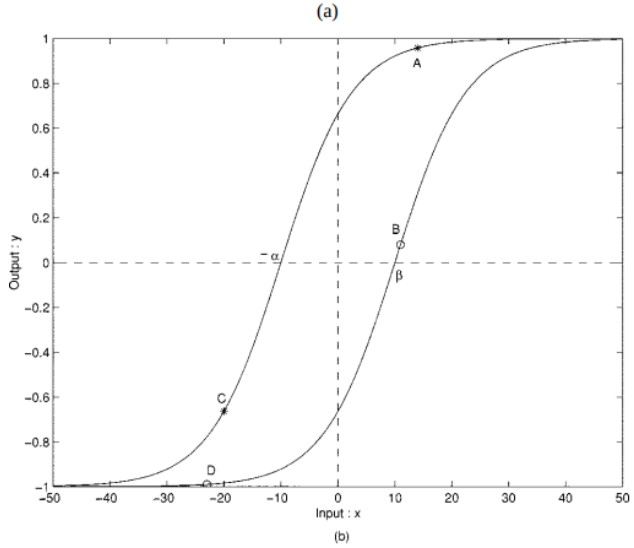


圖 3.5　遲滯雙曲正切(tanh)激活函數[21]

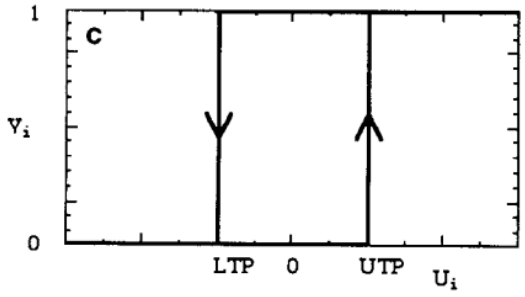


圖 3.6　遲滯符號(sign)激活函數[22]

圖 3.5、圖 3.6 是兩種常見的遲滯激活函數。其中符號激活函數在平移後與斯密特觸發器相等(圖 3.2)，並且可以視為是將雙曲正切函數在增益趨近無窮大後平移的結果。

doi:10.6342/NTU202400902

參考[23]對於遲滯符號激活函數用於 Hopfield 模型後的收斂條件證明，可以用於圖 3.2 之施密特觸發器上：

首先根據對稱且對角線為零的權重矩陣限制，令閾值$U_i = 0$，式(3.6)變為：

$$\begin{aligned}\Delta E &= -\left(\sum_j W_{ij}V_j + I_i\right)\Delta V_i \\ &= -u_i'\Delta V_i\end{aligned}$$

式(3.14)

此時會有兩種情況使得更新前後$\Delta V_i \neq 0$：

1. $V_i = -1, V_i' = +1, \Delta V_i = 2$

根據定義，此時輸入u_i'跨越閾值t_{pos}，可知$u_i' > t_{pos}$，因此：

$$\Delta E = -2u_i' < -2t_{pos}$$

式(3.15)

若$2t_{pos} \geq 0$，可使$\Delta E < 0$。

2. $V_i = +1, V_i' = -1, \Delta V_i = -2$

同樣根據定義，此時輸入u_i'跨越閾值$-t_{neg}$，可知$u_i' < -t_{neg}$，因此：

$$\Delta E = 2u_i' < -2t_{neg}$$

式(3.16)

若$2t_{neg} \geq 0$，可使$\Delta E < 0$。

因此，遲滯 Hopfield 模型在採用遲滯符號激活函數下，收斂到局部極小值之條件為$t_{pos} \geq 0$且$t_{neg} \geq 0$。

在後續的 BNN 訓練以及遲滯 Hopfield 模型中，使用了符號遲滯激活函數，並將其閾值設為$t_{pos} = t_{neg}$，透過標準差或變異數乘上一恆正常數確保系統能滿足收斂到局部極小值之條件。

doi:10.6342/NTU202400902

SUMMARY OF EXPERIMENTAL RESULTS FOR $N = 50$-QUEEN PROBLEM. ANN: ARTIFICIAL NEURAL NETWORK; HNN: HOPFIELD NEURAL NETWORK; HHNN: HYSTERETIC HOPFIELD NEURAL NETWORK

ANN	Standard energy function		Modified energy function	
	No tuning	Tuning	No tuning	Tuning
HNN	61%		83%	
HHNN	71%	72%* / 63%**	78%	90%* / 82%**

* All parameters were tuned using gradient descent.
** Only (α,β) parameters were tuned using gradient descent.

圖 3.7　遲滯 Hopfield 模型於 N-皇后問題收斂比率比較[21]

圖 3.7 為[21]使用包含雙曲正切遲滯激活函數之 Hopfield 模型用於解 N-皇后(N-Queen)問題，將N^2個神經元對應到棋盤上的位置，限制 N 個皇后要擺放在互不衝突的位置。由該結果可以得知，加入遲滯後能夠得到更高的收斂比率，並且調整遲滯激活函數之平移量(圖 3.5 中的α, β)能夠進一步提高收斂比率。

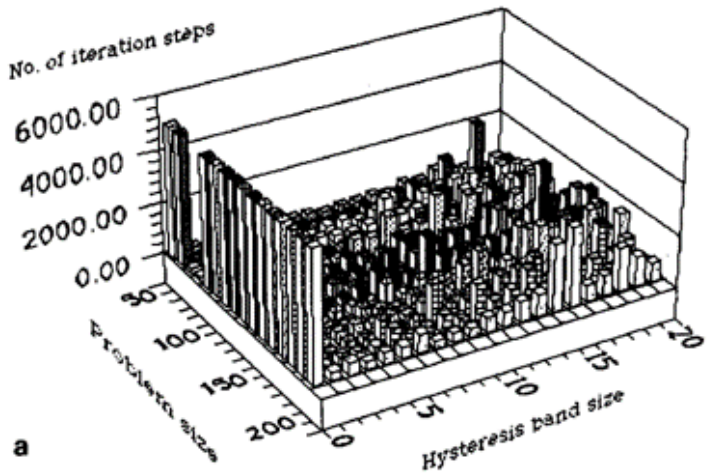


圖 3.8　遲滯 Hopfield 模型於縱橫交換機問題收斂迭代次數比較[22]

doi:10.6342/NTU202400902

[22]將包含符號遲滯激活函數之 Hopfield 模型用於解$N \times N$縱恆交換機(Crossbar switch)問題。圖 3.8 是不同問題大小、遲滯區域大小下所需的迭代次數。可以觀察到在遲滯區域=0 時，等同於原始不包含遲滯的 Hopfield 模型，會需要最多的迭代次數，只要加入部分的遲滯即可顯著降低迭代次數。

同樣可以發現，隨著遲滯區域變大，所需的更新次數又會再度上升，因此遲滯區域的決定需要特別仔細調整。這邊則是嘗試各種區域大小後選擇表現最佳的那個，對於複雜度更高的問題來說這個方法是不太可行的。

3.2.3 遲滯 Hopfield 模型與二值化神經網路訓練

二值化神經網路訓練可視為是一種組合最佳化問題：找出一組二值化的權重，使得損失函數的值最小化。

這邊以單一一層線性層(Linear layer)之神經網路訓練作為範例，對應到遲滯 Hopfield 模型之演化。此處的任務為線性迴歸，先隨機產生二值化權重作為N維超平面(hyperplane)之係數，此平面即為迴歸的最佳解。接著再隨機產生M個N維浮點數訓練資料點$x_{i,j}$，帶入超平面得到目標值y_i。訓練的目標是降低神經網路輸出與目標值之均方誤差(Mean square error, MSE)。

令神經網路之權重為W_i，可寫出損失函數為式(3.17)：

$$Loss = \frac{1}{M}\sum_{i=1}^{M}\left(\sum_{j=1}^{N} W_j \cdot x_{i,j} - y_i\right)^2$$

式(3.17)

將式(3.17)展開並分離平方項，可得式(3.18)：

$$Loss = \frac{1}{M}\sum_{i=1}^{M} - \left(\sum_{j=1}^{N}\sum_{k=1,k\neq j}^{N} x_{i,j}x_{i,k}W_jW_k - \sum_{j=1}^{N} x_{i,j}y_iW_j\right) + \sum_{j=1}^{N}(x_{i,j}W_j)^2 + y_i^2$$

式(3.18)

doi:10.6342/NTU202400902

因為W_j二值化後只能是+1, −1，式(3.18)後的平方項為定值，因此可以去除掉平方項，最小化剩餘一次項的部分相當於最小化原始的均方誤差。

將式(3.18)對應到N個神經元V_i的 Hopfield 模型之能量函數，可以得到對應的權重矩陣W_{ij}以及外部輸入I_i：

$$\begin{cases} W_{ij} &= \frac{1}{M}\sum_{i=1}^{M} x_{i,j}x_{i,k} \\ I_i &= \frac{1}{M}\sum_{i=1}^{M} x_{i,j}y_i \end{cases}$$

式(3.19)

給定權重矩陣W_{ij}以及外部輸入I_i後，對式(3.7)進行求解即可計算 Hopfield 模型之狀態演化。我們採用歐拉法(Euler's method)求解常微分方程，整體演算法之虛擬碼由圖 3.9 所示：

Algorithm: Continuous Hopfield Model with Euler's Method.

Data: Weight matrix: `W`, External input: `I`, Activation function: `g`, Time step: `dt`
Result: Converged state `u` of the Hopfield model

Random initialize `u` ;
repeat
  Compute $\texttt{V} = \texttt{g}(\texttt{u})$;
  Compute $\texttt{du[i]} = \left(-\texttt{u[i]} + \sum_j \texttt{W[i, j]V[j]} + \texttt{I[i]}\right) \times \texttt{dt}$;
  $\texttt{u} \leftarrow \texttt{u} + \texttt{du}$;
until *convergence*;

圖 3.9　歐拉法求解連續 Hopfield 模型之虛擬碼

doi:10.6342/NTU202400902

除了原始的 Hopfield 模型之外，這邊比較了採用圖 3.5 遲滯雙曲正切(tanh)激活函數之遲滯 Hopfield 模型。在參數的部分正切函數的斜率為一固定值，平移量則分別採用兩種做法進行比較：

1. 根據[21]使用梯度下降演算法(Gradient descent)，計算能量函數對平移量之梯度式(3.20)，用於更新平移量λ_i之值。該方法之虛擬碼如圖 3.10 所示。

$$\begin{cases} \alpha(t+1) & = \alpha(t) - \eta \dfrac{\partial E}{\partial V_i} \dfrac{dV_i}{d\alpha(t)} \\ \beta(t+1) & = \beta(t) - \eta \dfrac{\partial E}{\partial V_i} \dfrac{dV_i}{d\beta(t)} \end{cases}$$

式(3.20)

Algorithm: Continuous Hysteretic Hopfield Model with Euler's Method.

Data: Weight matrix: `W`, External input: `I`, Learning rate: `lr`, Hysteretic activation function: `g`, Time step: `dt`
Result: Converged state `u` of the Hopfield model

Random initialize `u` ;
Zero initialize threshold α, β ;
Zero initialize `du` ;
repeat
  Compute $\texttt{V} = \texttt{g}(\texttt{u}, \texttt{du})$;
  `t[i]` $\leftarrow \alpha$[i] or β[i] based on the sign of `du[i]` ;
  Compute $\texttt{du[i]} = \left(\texttt{t[i]} - \texttt{u[i]} + \sum_j \texttt{W[i, j]V[j]} + \texttt{I[i]} \right) \times \texttt{dt}$;
  α[i] $\leftarrow \alpha$[i] $- \texttt{lr} \times \frac{dE}{d\alpha[\texttt{i}]}$;
  β[i] $\leftarrow \beta$[i] $- \texttt{lr} \times \frac{dE}{d\beta[\texttt{i}]}$;
  $\texttt{u} \leftarrow \texttt{u} + \texttt{du}$;
until *convergence*;

圖 3.10 歐拉法求解連續遲滯 Hopfield 模型結合梯度下降調整平移量之虛擬碼

2. 採用各神經元 u_i 值之統計量(變異數或標準差)乘上一常數後作為平移量。

由於激活函數之輸入為u_i，採用統計量來控制平移量可以根據u_i之分布控制落在平移量範圍之外的比例。

doi:10.6342/NTU202400902

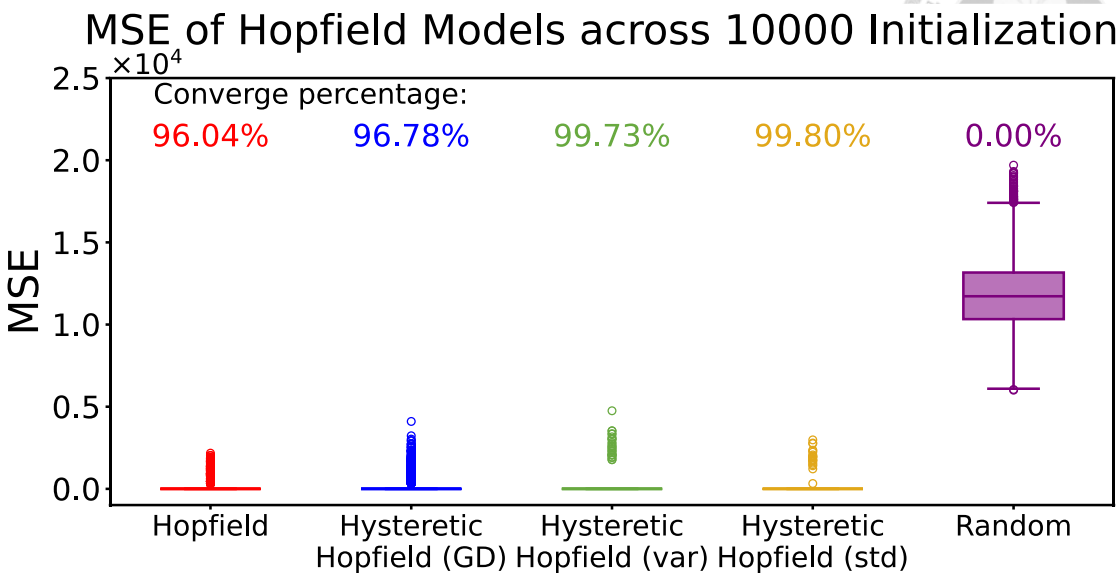


圖 3.11　不同遲滯 Hopfield 模型平移量調整方法之均方誤差

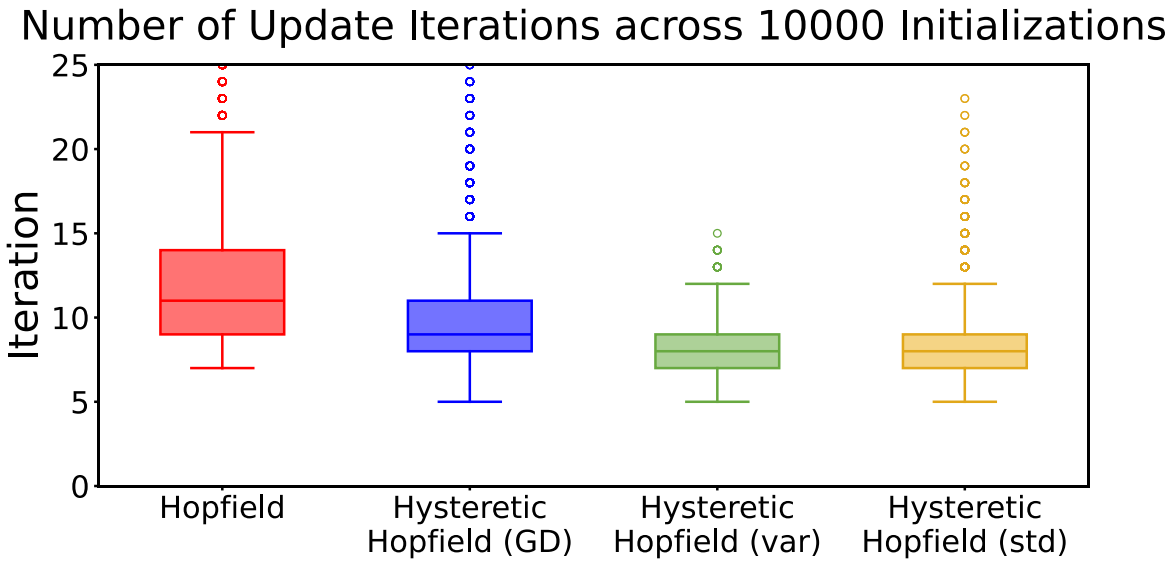


圖 3.12　不同遲滯 Hopfield 模型平移量調整方法所需之迭代次數

doi:10.6342/NTU202400902

圖 3.11、圖 3.12 模擬不同調整遲滯激活函數之平移量在線性迴歸問題下的均方誤差以及收斂所需之迭代次數。對於所有方法採用相同的時間步長$dt = 10^{-3}$，並且使用相同的10,000組隨機數初始化u_i進行迭代，收斂的條件也固定為連續三次迭代沒有使能量降低即終止。對於有調整平移量之方法會調整該方法之參數，如梯度下降法之學習率$\eta = 0.01$，以及統計量乘上之常數(變異數乘上0.4，標準差則乘上0.2)。圖 3.11 圖中隨機的方法則是直接隨機產生二值化權重，不做任何迭代更新。

從圖 3.11 我們可以發現 Hopfield 模型收斂後的均方誤差顯著比隨機產生的二值化權重低，並且加入遲滯後的 Hopfield 模型又比沒有遲滯的 Hopfield 模型有更低的均方誤差。而比較三種遲滯 Hopfield 模型，使用統計量作為平移量的方法又比使用梯度下降法更新平移量的方法有更高的比例會收斂到最佳解。

在圖 3.12 中可以觀察到同樣是使用統計量計算平移量的方法比起原始的 Hopfield 模型以及梯度下降法更新平移量之遲滯 Hopfield 模型更快達到收斂。值得注意的是，雖然在均方誤差下，使用標準差計算平移量的收斂比例比使用變異數更高。但是在最差情況下，使用標準差計算平移量到達收斂所需的迭代次數會遠比使用變異數來得高。

以上結果顯示，我們可以使用連續 Hopfield 模型之u_i之統計量來調整遲滯激活函數中的平移量，比起原始不加遲滯的 Hopfield 模型以及使用梯度下降調整平移量能更快達到收斂，且收斂時的能量更低。

與梯度下降法相比較，以統計量調整平移量的方法在超參數(hyperparameter)的個數上與之相等，皆為一個超參數。但是梯度下降法對於N個神經元採用 $2N$ 組個別的平移量，且平移量之梯度與訓練時的損失函數相關，難以各自調整。

doi:10.6342/NTU202400902

對應到二值化神經網路量化，我們可以將 Hopfield 模型之神經元u_i對應到量化感知訓練中的 fp32 master copy，V_i對應到二值化後的權重，能量函數對應到訓練時的損失函數。

此時當遲滯激活函數之增益趨近無窮大時，雙曲正切函數會接近於符號函數，也就是我們在二值化神經網路訓練中加入的斯密特觸發器圖 3.2。此時遲滯激活函數之平移量對應到斯密特觸發器之閾值(t_{neg}, t_{pos})。

因此根據 Hopfield 模型之實驗結果，我們可以證實在單一一層線性二值化網路進行線性迴歸的訓練中，以統計量計算遲滯之閾值能夠改善量化感知訓練時的收斂速度，以及收斂到正確率更高的模型。

根據以上結論，我們將這個概念應用在更複雜的模型、更困難的任務上。在模型的部分可以是多層線性層、卷積層組成的卷積神經網路、Transformer。並且兩層神經元中間插入非線性激活函數。在任務的部分則有影像辨識、物件偵測、自然語言處理之應用。在進行以上模型之二值化量化感知訓練時，我們使用遲滯符號激活函數以 fp32 master copy 之統計量計算閾值加入遲滯效應，試圖提高二值化後模型的表現。

doi:10.6342/NTU202400902

3.3 實驗數據與結果

本章節以前述遲滯 Hopfield 模型為理論基礎，在二值化神經網路訓練中加入符號遲滯激活函數。並根據先前的實驗結果，使用 fp32 master copy 之統計量計算遲滯之閾值。在不同的模型以及應用上比較遲滯帶來的正確率提升。

3.3.1 簡單線性模型實驗結果

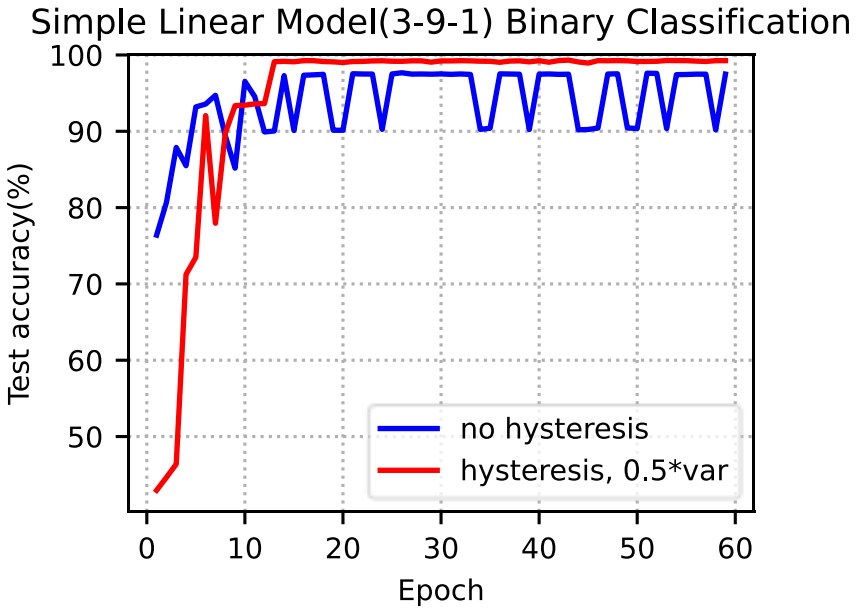


圖 3.13 線性模型進行二元分類加入遲滯前後於測試集之準確率

回到最初採用遲滯的動機，在圖 3.1 中使用兩層線性層的簡單模型做二元分類時會有震盪的情況。這邊以相同的訓練設定，在訓練時加入遲滯符號激活函數，並以 0.5 乘上每一層權重對應的 fp32 master copy 之變異數作為遲滯符之閾值。

從圖 3.13 的訓練曲線可以看到，在加入了遲滯後，從第 13 個 epoch 開始即收斂到 99%以上的分類正確率。並且在收斂後不會再產生震盪的情形。

與前面使用 Hopfield 模型推導的線性迴歸不同，這裡神經網路的輸出經過非線性的 sigmoid 函數。顯示在非線性的激活函數下遲滯也能改善二值化訓練。

doi:10.6342/NTU202400902

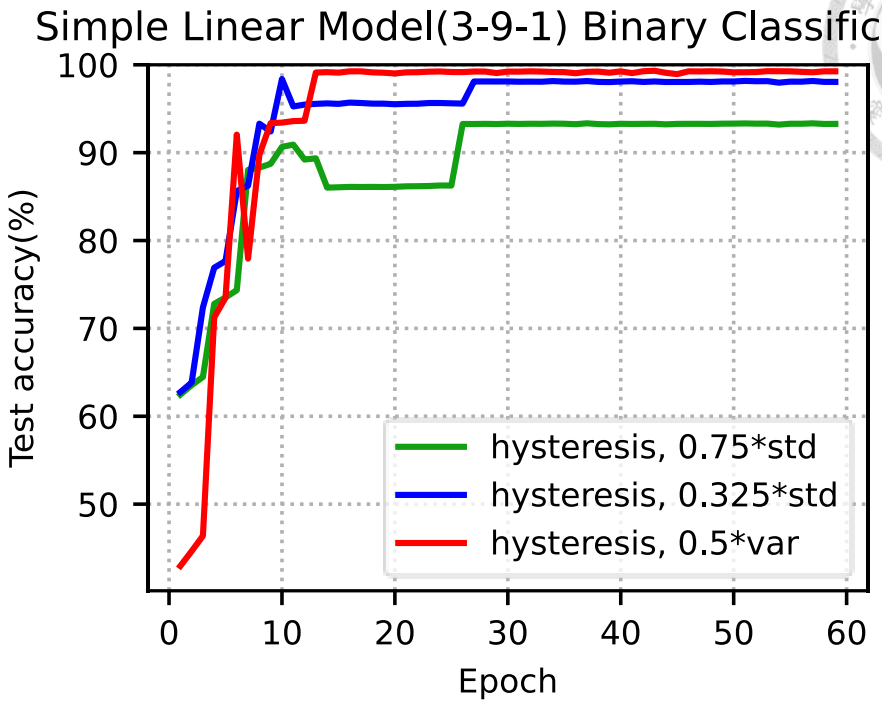


圖 3.14　線性模型進行二元分類以不同閾值之遲滯於測試集之準確率

圖 3.14 比較了不同閾值設定下的遲滯用於二元分類問題下的測試準確率。這裡分別使用乘上 0.325 與 0.75 常數的標準差作為閾值，相比變異數的閾值，在初始值的部分一個會小於變異數閾值，另一個則大於變異數閾值。

在最終的正確率表現上，三者皆沒有出現震盪的情形。但是相比於變異數作為閾值的作法，使用標準差的兩次訓練在最終的正確率都無法收斂到 99%。

根據實驗結果發現使用變異數作為閾值來訓練二值化神經網路能夠達到更高的測試準確率，因此後續的實驗中主要是基於變異數決定遲滯之閾值。

doi:10.6342/NTU202400902

3.3.2 卷積神經網路(CNN)實驗結果

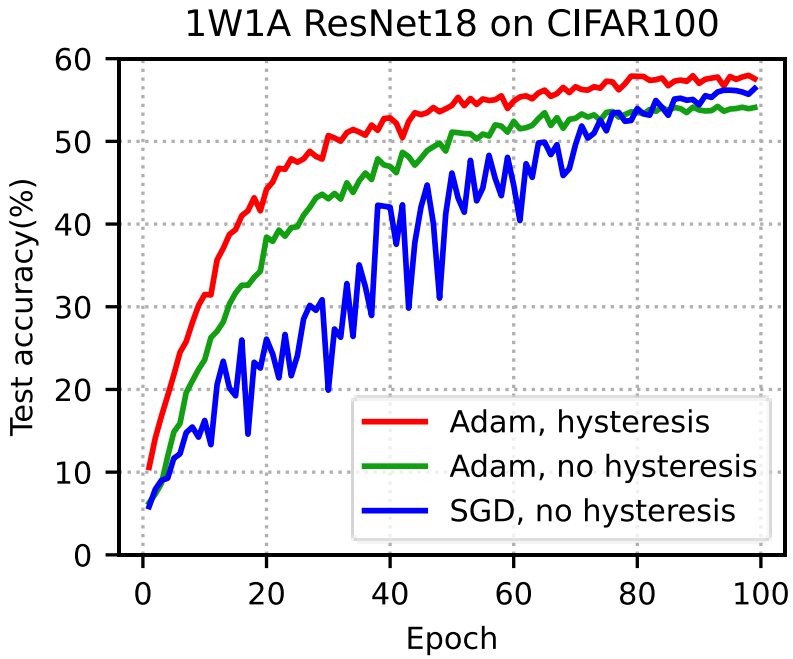


圖 3.15 1W1A 量化後 ResNet18[24]於 CIFAR100[25]測試集之準確率

圖 3.15 是將 ResNet18[24]進行二值化後在 CIFAR100[25]的測試準確率。在沒有遲滯的情況下，雖然 Adam 比起 SGD 較為穩定，但是最終收斂到的測試準確率比 SGD 低了 2.3%。而加入了遲滯後的 Adam 在收斂速度上比原始的 Adam 更快，並且最終的正確率比 SGD 高了 1.2%。

表 3.1 ResNet18[24]於 CIFAR100[25]進行 1W4A 量化感知訓練結果

Precision	Hysteresis	Test Accuracy (%)	Diff (%)
fp32	No	74.7	-
1W4A	No	72.2	-2.5
1W4A	Yes	**72.8**	-1.9

表 3.1 將 activation 量化精度改為 4-bit 後，並且使用預訓練 fp32 權重作為初始化之訓練結果。最終的準確率相比 fp32 模型只相差 1.9%。

doi:10.6342/NTU202400902

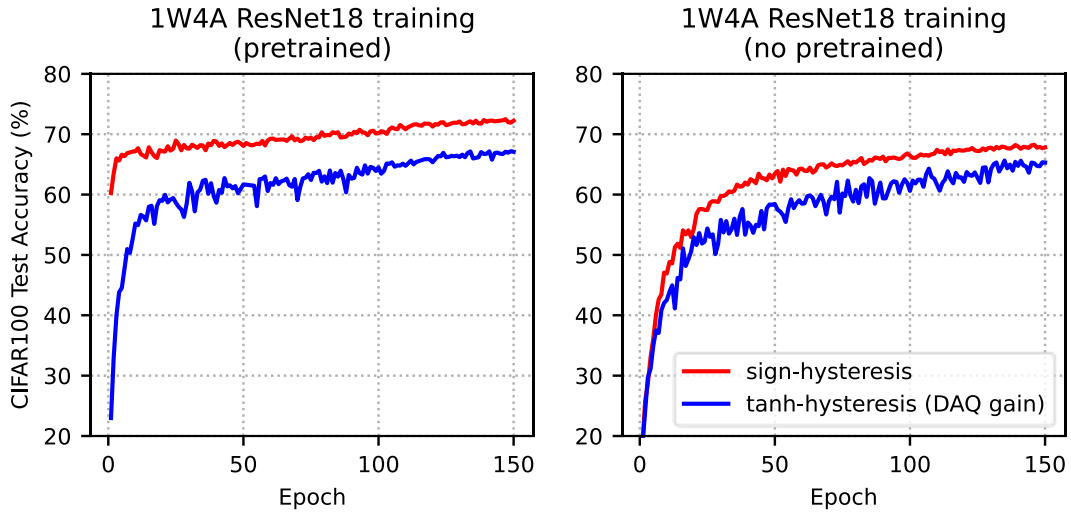


圖 3.16　ResNet18[24]使用不同遲滯量化至 1W4A 於 CIFAR100[25]之準確率

圖 3.16 比較使用雙曲正切遲滯函數與符號遲滯函數於二值化神經網路訓練上之效果。在訓練設定上除了使用不同的遲滯函數進行二值化外，其餘的超參數設定皆相同。

其中雙曲正切函數是將原先遲滯使用的符號函數sign(x)改為$\tanh(k \cdot x)$，其中k為控制斜率之參數，當k越接近無窮大，雙曲正切函數會越接近符號函數。

在參數k的調整上使用了 DAQ[26]的排程，透過每個權重對應的 fp32 master copy 與相鄰的兩個量化等級的距離調整k值，調大接近 0 的權重對應的k值。此外，因為雙曲正切函數可能會將權重量化到±1以外的值，因此在測試集推論時會採用符號函數進行二值化。

從圖中正確率可以看出使用符號遲滯函數的二值化訓練比雙曲正切函數更好，在使用預訓練參數進行初始化的情況下，使用符號函數的測試準確率比起雙曲正切函數高了 5.2%，在隨機初始化的情況下則高了 2.5%。並且符號函數的測試準確率曲線比雙曲正切函數更為穩定。

doi:10.6342/NTU202400902

雙曲正切函數表現較差的原因推測是訓練與推論時二值化方法不同而導致訓練效果不佳。雖然目前有動態調整k值降低這個差異，但是從圖 3.16 左圖可以看出，在使用預訓練參數進行初始化的情況下，符號函數能夠從 60%的準確率繼續訓練，而雙曲正切函數卻使模型偏離預訓練參數，準確率降低至 23%，顯示在動態調整k值的情況下，訓練時使用雙曲函數的 soft binarization 與測試時使用符號函數的 hard binarization 還是存在差異。這個差異最終導致訓練曲線不穩定，並且最後收斂到較低的準確率。

因為符號函數表現更好，並且去除了調整k值的額外麻煩，所以在訓練二值化神經網路時，我們使用基於符號函數實現遲滯訓練。

表 3.2　ResNet18[24]於 ImageNet1k[27]進行 1W4A 量化感知訓練結果

Precision	Test Accuracy (%)	Diff (%)
fp32	69.6	-
1W4A	66.3	-3.3

表 3.2 是使用遲滯量化 ResNet18 至 1W4A 精度之準確率。結合了第二章提到的各種二值化訓練技巧以及以符號激活函數引入遲滯，最終在 ImageNet1k 達到 66.3%的準確率，相比 fp32 模型只降低 3.3%。

doi:10.6342/NTU202400902

Pascal VOC[28]是常用於物件偵測(object detection)與影像分割(image segmentation)的資料集。學術界通常在 2005 年至 2012 年的競賽資料中，取出 07 與 12 年的訓練資料進行訓練，並在 07 年的測試資料上進行評估，這邊我們也採用相同的設定訓練。

Single Shot Multibox Detector (SSD)[29]為單階段(one-stage)物件偵測模型。如 Faster-RCNN[30]兩階段模型，需要先找出物件再分類。雖然準確率較高，但是兩階段的運算比較慢，單階段模型比較能達到實時偵測的需求。

原始的 SSD 模型採用 VGG[31]模型作為骨幹(backbone)，這邊則採用表現更好、訓練更容易的 ResNet50 做為骨幹，並進行二值化訓練。

表 3.3　ResNet50-SSD 於 Pascal VOC[28]進行 1W4A 量化感知訓練結果

Precision	Hysteresis	mAP (%)	Diff (%)
fp32	No	74.1	-
1W4A	No	61.4	-12.7
1W4A	Yes	**71.0**	-3.1

表 3.3 列出了二值化後的 ResNet50-SSD 於 Pascal VOC 2007 測試集之準確率。在加入了遲滯後，1W4A 量化後的 ResNet50-SSD 之準確率為 71.0，相比無遲滯訓練提高了 9.6%，與 fp32 模型來到 3.1%的準確率差異。

相比於單純的影像辨識任務，ResNet50-SSD 的結果顯示在物件偵測模型中加入遲滯也能改善二值化訓練。

doi:10.6342/NTU202400902

3.3.3 可分離式卷積(Depthwise Separable Convolution) 網路實驗結果

MobileNet[32]是基於可分離式卷積(depthwise separable convolution)設計，為了嵌入式與行動裝置設計的神經網路模型。可分離式卷積是將普通卷積分離成 pointwise 與 depthwise 兩個卷積層，其中 depthwise 卷積對輸入特徵圖的個別通道進行卷積運算，而 pointwise 則聚集各通道的資訊，透過 1x1 尺寸的卷積核對特徵圖進行升降維，改變輸出通道個數。

在較近期的 EfficientNet[33]中承襲了 MobileNet 的設計，在其 inverted residual block 中也採用了可分離式卷積的設計。使用了可分離式卷積後在參數量和計算量上比起原始的卷積層更少，讓 EfficientNet 以及 MobileNet 可以使用更少的計算成本達到與其他模型相近的表現。然而，這些模型精簡的架構也使他們比起普通的卷積神經網路更難在量化後維持準確率。

本章節將遲滯二值化訓練應用於包含可分離式卷積之神經網路模型中，比較加入遲滯後對於訓練穩定性以及模型表現的改善。

doi:10.6342/NTU202400902

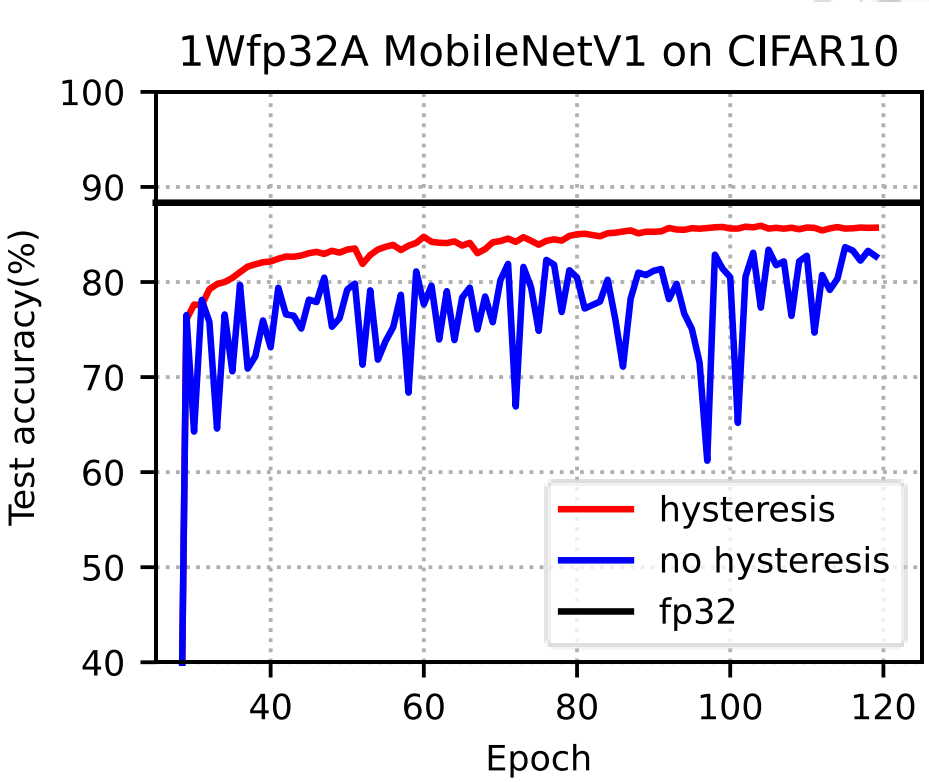


圖 3.17　MobileNetV1[32]進行二值化後於 CIFAR10[25]之準確率

圖 3.17 為 MobileNetV1[32]進行權重二值化於 CIFAR10 訓練之測試準確率。在訓練時採用了第二章提過的漸進訓練(progressive training)，因此到訓練中期準確率才提升到 70%以上。

從圖中可以看到，因為 depthwise 卷積更少的參數，使得二值化權重在訓練時，單一權重正負號翻轉造成該層網路輸出結果變動的幅度更大，導致訓練時準確率不穩定，收斂到較差的準確率。

加入基於統計量的遲滯後，準確率抖動的情況大幅減少，並且最終準確率比原先高 2.1%，進一步縮短與未量化的 fp32 模型之差距。

doi:10.6342/NTU202400902

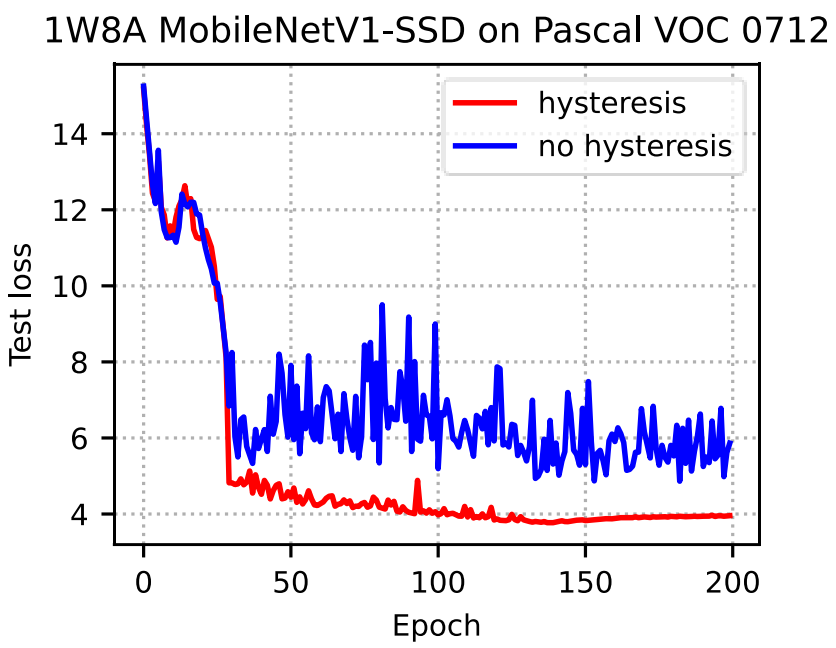


圖 3.18　MobileNetV1-SSD 進行二值化後於 Pascal VOC 之準確率

表 3.4　MobileNetV1-SSD 於 Pascal VOC 進行 1W8A 量化感知訓練結果

Precision	Hysteresis	mAP (%)	Diff (%)
fp32	No	67.4	-
1W8A	No	35.2	**-32.2**
1W8A	Yes	52.7	**-14.7**

這邊以 MobileNetV1 為骨幹之 SSD 模型在 Pascal VOC 上進行物件偵測。由於 MobileNetV1 中 depthwise 卷積的存在，加入遲滯後從圖 3.18 中可以看出其對於訓練之穩定性提升。表 3.4 列出了二值化後的準確率，在遲滯改善訓練穩定性後，於 Pascal VOC 上準確率提升了 17.5%。

總結而言，對於存在 depthwise 卷積的模型，加入遲滯能夠顯著改善訓練穩定性，進一步提升二值化模型表現。

doi:10.6342/NTU202400902

3.3.4 Transformer 實驗結果

DeiT(Data-efficient image transformers)[34] 在架構上與原始的 vision transformer[35]相同，藉由調整訓練設定以及知識蒸餾等技巧以更微縮的模型達到更高的辨識準確率，同時降低訓練資料的需求。

這裡使用 DeiT-base 大小的模型進行影響辨識的二值化訓練。在資料集的部分從 ImageNet1k[27]中取出前 100 個類別的資料稱為 ImageNet100 作為訓練資料集，資料量降低為原先的約 1/10，大幅增加了訓練速度。

在量化的設定上除了第一層的卷積層和最後一層線性層以外都將權重進行了二值化。在 activation 的部分大部分為 8-bit，在注意力層(attention layer)的部分對其中的 Q, K, V 做更低精度的量化，Q 與 K 矩陣進行 1-bit 量化，V 矩陣則量化至 4-bit。這裡寫成 1W4/8A*表示。

$$\text{softmax}\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) \cdot V$$

式(3.21)

表 3.5 DeiT-base[34]於 ImageNet100 進行 1W4/8A*量化感知訓練結果

Precision	Hysteresis	Test Accuracy (%)	Diff (%)
fp32	No	87.4	-
1W4/8A*	No	83.8	-3.6
1W4/8A*	Yes	85.1	-2.3

加入遲滯後，二值化之 DeiT-base 於 ImageNet100 之測試準確率提高了 1.3%，顯示在 vision transformer 上加入遲滯同樣也能有效改善二值化訓練。

doi:10.6342/NTU202400902

DaViT(Dual attention vision transformers)[36]相比 DeiT 是較新的 vision transformer 模型。在架構上 DaViT 採用了與 Swin transformer[37]相同的階層式(hierarchical)架構，透過多層 patch embedding 層使用卷積對特徵進行降採樣(downsampling)以取出不同大小的特徵。此外 DaViT 還引入了雙注意機制(dual-attention)，除去原始在圖片寬高維度的 window attention 機制，DaViT 在通道維度加上了 channel attention 機制，能夠更好的把握全局資訊。

在其他 binary transformer 訓練中通常如[10]沒有量化 patch embedding 層，但是在 DaViT 中使用了多層 patch embedding，這邊則是一樣進行權重二值化，但是在比較難量化的 patch embedding 層中使用兩倍的權重。其餘量化的精度設定與前述 DeiT 相同，大部分採用 8-bit activation，在 attention 層中則使用 1-bit Q, K 以及 4-bit V。

表 3.6 DaViT-tiny[36]於 ImageNet100 進行 1W4/8A*量化感知訓練結果

Precision	Hysteresis	Test Accuracy (%)	Diff (%)
fp32	No	85.1	-
1W4/8A*	No	79.8	-5.3
1W4/8A*	Yes	81.8	-3.3

表 3.7 DeiT-base[34]與 DaViT-tiny[36]模型 QAT 後 ImageNet100 準確率比較

Model	Params. (M)	FLOPs (G)	Accuracy (%)	Diff (%)
DeiT-base	86.6	17.6	85.1	-2.3
DaViT-tiny	28.3	4.5	81.8	-3.3

doi:10.6342/NTU202400902

表 3.6 為 DaViT-tiny 在 ImageNet100 訓練後的準確率。在加入了遲滯後二值化後的準確率提高了 2%。顯示了在較新的 vision transformer 架構且較微縮的模型大小下，加入遲滯也能改善二值化訓練之準確率。

先前使用的任務皆為影像為主的任務，這裡以自然語言處理(Natural Language Processing, NLP)測試遲滯二值化訓練。

BERT(Bidirectional Encoder Representations from Transformers)[16]是用於自然語言處理的 transformer 模型，是採用 encoder 部分將文字取出文字特徵，應用於下游任務(downstream tasks)中。BERT 是第一個使用統一的架構應用於多種 NLP 任務中，並超越許多專為單一任務設計的模型表現。

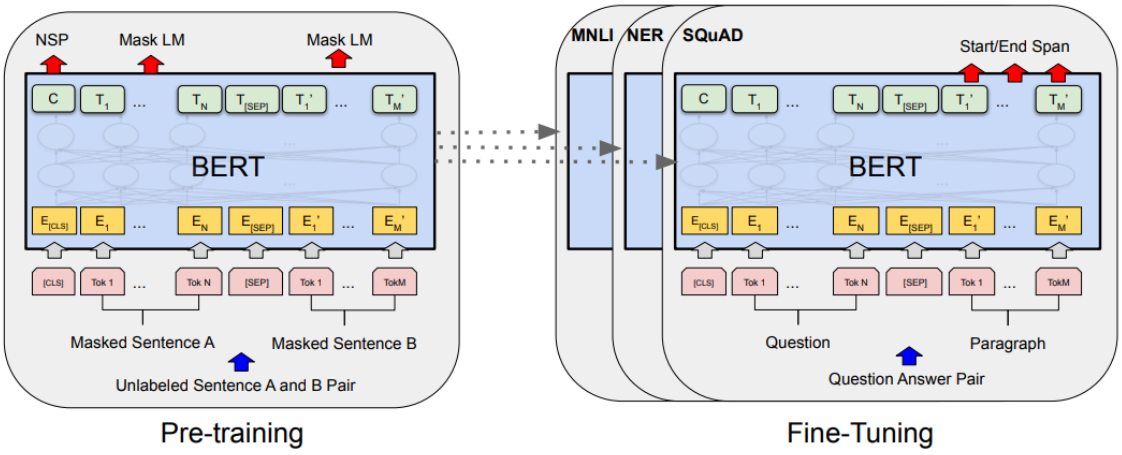


圖 3.19 BERT 兩階段訓練示意圖[16]

圖 3.19 是 BERT 的兩階段訓練過程，首先會先以無標註的訓練資料進行 masked language modeling (Masked LM)以及 next sentence prediction (NSP)的預訓練。Masked LM 讓模型預測被遮蓋住的 token，NSP 則讓預測兩個句子是否連續出現在文本中。兩個任務讓 BERT 能學習出如何表示輸入的文字。第二階段則是讓預訓練完的模型在下游任務中進行 fine-tune，針對個別應用調整權重。

doi:10.6342/NTU202400902

在二值化訓練中，因為第一階段預訓練的時間成本過高，因此我們採用未量化的預訓練模型進行初始化，並在個別下游任務 fine-tune 的同時對整體網路做二值化。在量化的設定上 encoder 皆為二值化權重，embedding 與 activation 量化為 4-bit，用於下游任務的最後一層線性層則維持在高精度。

GLUE(The General Language Understanding Evaluation)[15]是常用於測試 NLP 模型的資料集。其中包含了多種自然語言處理之任務，例如由前提推論假設是否成立的 MNLI (Multi-Genre Natural Language Inference)、判斷兩個問題句語意是否相同的 QQP (Quora Question Pairs)、判斷單一句子是正面還是負面情緒的 SST-2 (Stanford Sentiment Treebank)等任務。

我們在 GLUE 的各個任務上進行 fine-tune 以及二值化訓練，以相同的訓練設定測試加入遲滯前後的正確率差異。在任務的選擇上以 BiT[14]為基礎，排除掉不穩定的 WNLI(Winograd Natural Language Inference)任務。

表 3.8　BERT[16]於 GLUE[15]各任務 1W4A 遲滯量化感知訓練結果

Precision Hysteresis	MNLI m/mm	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	Avg.
fp32 No	84.9/85.5	91.4	92.1	93.2	59.7	90.1	86.3	72.2	**83.9**
1W4A No	83.0/83.6	90.0	89.5	91.6	35.5	85.7	82.8	58.1	**77.8**
1W4A Yes	83.4/83.9	91.1	90.7	91.9	38.2	85.9	83.5	60.6	**78.8**

在加入遲滯後，二值化後於各任務上之平均準確率改善約 1%，與原始未量化模型差距為 5.1%。這顯示了除了影像任務之外，在自然語言處理中加入遲滯後也能改善二值化後之準確率。

doi:10.6342/NTU202400902

3.3.5 與其他 1W1A BNN 比較

在本章節中我們將遲滯二值化訓練與其他 BNN 技術[1, 2, 38-41]相結合，比較加入遲滯前後訓練後模型之準確率。

在這些 BNN 技術中，[39]使用 activation 絕對值的平均來計算 per-tensor scaling factor，提高二值化後的模型表現。[1]修改網路架構，在每層二值化卷積層加入 fp32 skip-connection，改善二值化後模型表現能力，但是與原始 fp32 模型仍有 14.2%之準確率差異。[42]則使用 RSign 與 PReLU，透過增加些許參數的方式來更好的學習 activation 二值化。

現有的 BNN 技術則在架構上引入更多人工設計與修改。例如[2]使用了 Info-RCP 架構來進行 per-channel scaling，使損失函數更加平滑。

在以上的 BNN 技術中，都沒有引入遲滯來提高訓練穩定性。而遲滯二值化可以透過在 weight binarization 中加入記憶前一次二值化結果來實現。在實驗設定上，我們選擇基於 ResNet18[24]之 BNN 模型，在 ImageNet100 上分別訓練原始不含遲滯的版本與加入遲滯二值化後的版本。兩次訓練的設定維持與原始設定相同。

doi:10.6342/NTU202400902

表 3.9 ResNet18[24]於 ImageNet100 1W1A 加入遲滯量化前後訓練結果

Method	Param. (MB)	Ops (10^8)	ImageNet1k Accuracy(%)	ImageNet100 Accuracy(%)		Accuracy Gain(%)
				No hysteresis	With hysteresis	
fp32	46.8	18.2	69.6	79.16	-	**-**
BNN[38]	4.2	1.47	42.2	48.74	49.36	**+0.62**
XNOR-Net[39]	4.2	1.47	51.2	71.82	72.42	**+0.40**
XNOR-Net++[40]	4.2	1.47	57.1	70.86	71.30	**+0.44**
BiRealNet-18[1]	4.2	1.65	56.4	70.40	71.44	**+1.04**
ReActNet-Bir18[41]	4.2	1.89	65.9	78.90	79.30	**+0.40**
BNext18[2]	2.2	0.43	67.9	81.08	81.38	**+0.30**

表 3.9 為其他 BNN[1, 2, 38-41]加入遲滯二值化前後的訓練準確率比較。其中運算量 Ops 計算方式參考[2]，透過 CPU 一個 cycle 內能夠執行之計算換算不同精度之運算，1W1A 運算為 BOPs，換算後可得$\text{OPs} = \frac{\text{BOPs}}{64} + \text{FLOPs}$ 。

基於 ResNet18 二值化模型在 ImageNet100 加入遲滯後，準確率提高了 0.3%~1.0%。顯示我們提出的遲滯二值化訓練能夠與現有 BNN 技術相結合，進一步提高二值化訓練後模型表現。

doi:10.6342/NTU202400902

3.4 第三章總結

第三章介紹了 Hopfield 模型的定義以及數學特性，並將遲滯 Hopfield 模型應用於單一線性層二值化網路的線性迴歸訓練，提供在二值化訓練加入遲滯的理論基礎，並從結果可以看出使用統計量決定平移量的優勢。

接著我以遲滯在各個模型及應用的二值化訓練中引入遲滯。結果顯示遲滯能有效去除震盪，提高訓練穩定性。在卷積神經網路以及 transformer 應用在影像辨識、物件偵測，與自然語言處理上都能提高二值化後的準確率。

doi:10.6342/NTU202400902

doi:10.6342/NTU202400902

第四章 結合 TPC-NAS 與二值化神經網路

在前兩個章節的二值化神經網路都使用與 fp32 模型相同的架構，在不做架構修改的情況下進行二值化訓練。在這個章節中，我們使用 TPC-NAS[43]作為神經網路架構搜尋(Neural architecture search, NAS)演算法在相同硬體限制下修改架構，提高模型表現，以彌補二值化後的準確率損失。

4.1 TPC-NAS 演算法總覽

神經網路架構搜尋(NAS)演算法是在給定搜尋空間(search space)、硬體計算限制(budget)下，找出表現最好、準確率最高的神經網路架構。

NAS 演算法按照訓練模型的次數可以分為三類：multi-shot, one-shot, zero-shot NAS。

1. Multi-shot NAS

Multi-shot NAS 主要是在搜尋空間中取樣出架構後實際訓練以評估該架構之表現。在搜尋演算法上可以採用基因演算法、貝氏最佳化(Bayesian optimization)[44]、強化學習(Reinforcement learning)[45]等。

雖然在演算法上有許多改進空間，透過降低訓練 epoch 數、減少訓練資料，加入模型表現預測器(performance predictor)等。但 Multi-shot NAS 花費的時間主要還是在訓練模型的部分，整個演算法在 GPU 會花上千小時的時間進行搜尋。

doi:10.6342/NTU202400902

2. One-shot NAS

One-shot NAS 主要將搜尋空間的所有架構組合出一個 supernet，只需要進行一次 supernet 的訓練即可完成 NAS 演算法。

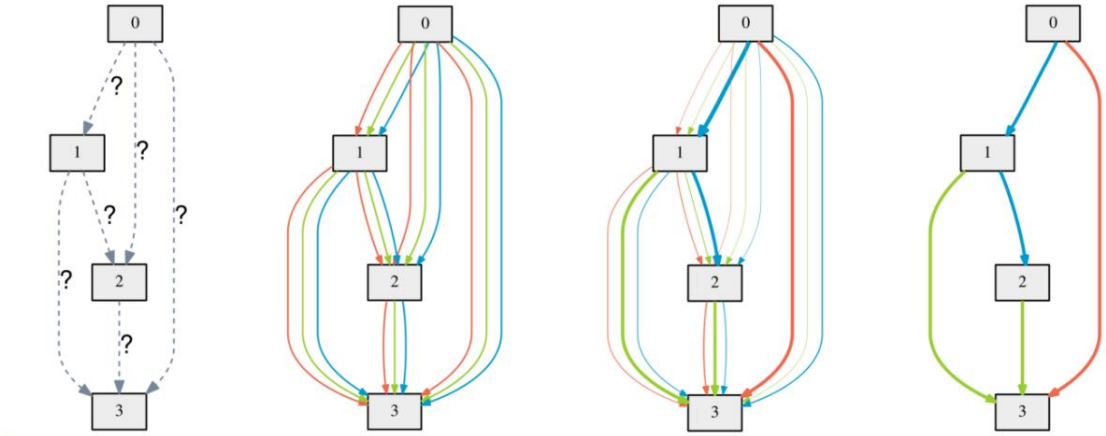


圖 4.1 DARTS 演算法示意圖[46]

DARTS[46]透過softmax將離散的搜尋空間變為連續可微分的 supernet，在 supernet 訓練完畢後根據softmax找出分數最高的運算，組合出 NAS 搜尋結果。

雖然 DARTS 的作法相較於多次訓練的作法能夠減少整體搜尋時間，但是在使用上面 supernet 的構建方法會使用到大量的記憶體，導致這種方法難以拓展到更大的模型上。

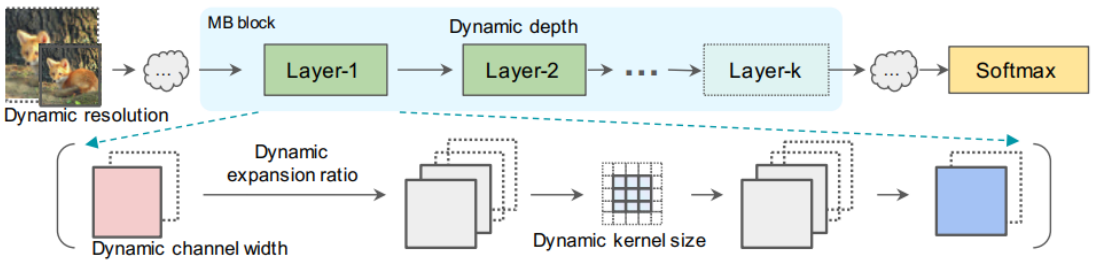


圖 4.2 兩階段 NAS 訓練示意圖[47]

doi:10.6342/NTU202400902

相對於 DARTS 的 supernet，另外一種 one-shot NAS 的做法是兩階段(two-stage)NAS。首先透過共用權重(weight-sharing)的方式構建 supernet，訓練完 supernet 後再根據硬體限制找出符合條件、表現最好的子網路(sub-network)。

圖 4.2 是兩階段 NAS 訓練的示意圖。在共用權重的部分，不同通道數的卷積權重可以共用，大卷積核之權重也能與小的卷積核共用，不同深度的網路也可以共用權重。在訓練過程中也不是直接訓練整個 supernet，而是在其中取樣出子網路訓練、更新權重。如此一來相較於 DARTS 可以大幅減少訓練 supernet 時的記憶體用量與訓練時間。

在第二個階段則可以使用基因演算法，根據子網路在目標任務推論的表現，找出符合硬體限制的架構。此時顯示了兩階段 NAS 的另一個優點，當搜尋空間不變，只改變硬體限制時，不需要重新訓練 supernet，只需要重新進行第二階段的搜尋即可。相當於在一次訓練下可以針對多種硬體限制進行搜尋。

不過共用權重的訓練也被發現存在缺陷，訓練時子網路的取樣策略必須謹慎決定，確保每個子網路能被公平地訓練。此外，不同子網路間的權重更新也會互相干擾。從訓練完畢的 supernet 採樣出的子網路跟單獨訓練的子網路表現不盡相同，也會影響兩階段 one-shot NAS 最終的搜尋結果。

3. Zero-shot NAS

Zero-shot NAS 不需要進行訓練，而是透過分析架構、進行一兩次的推論或是梯度計算來評估模型的表現，計算一個量化的分數來表示架構的好壞。這個分數配合如基因演算法的搜尋，可以在搜尋空間中找出符合硬體限制，分數最高的架構。因為省去了訓練的時間，zero-shot NAS 的搜尋時間通常在數個小時以下。

doi:10.6342/NTU202400902

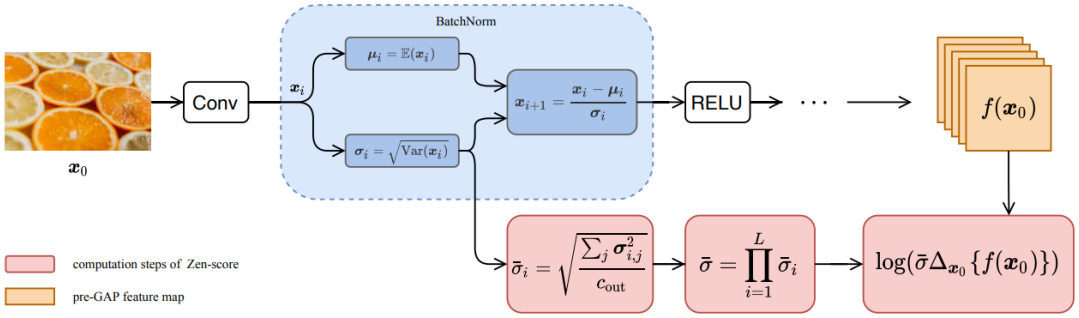


圖 4.3 Zen-Score 計算流程圖[48]

Zen-NAS[48]透過計算 Gaussian complexity 的期望值來評估架構的表達能力(Expressivity)。計算上在輸入加入 Gaussian noise，觀察推論後輸出的差異來近似梯度。並考慮了梯度爆炸的問題，乘上了網路中批正規化層(Batch normalization layer)中變異數的影響。

總路徑數神經網路架構搜尋(TPC-NAS)[43]屬於 zero-shot NAS 的分類。透過觀察架構中輸入到輸出的路徑總數計算總路徑數分數(TPC score)，藉由 TPC score 來衡量一個架構的表達能力(Expressivity)。一個架構的路徑總數越多，就能進行越複雜的轉換，因此表達能力更強。

因為計算 TPC score 不需要進行反向傳播或推論，所以在每一次分數的計算上比起其他 zero-shot NAS 更加迅速，能夠在相同時間內評估更多架構。在影像辨識、物件偵測、超解析度成像、自然語言處理等應用上，都能在數分鐘內完成神經網路架構搜尋，並在相近的計算量限制下，達到接近或是超越其他 NAS 演算法架構的正確率。

doi:10.6342/NTU202400902

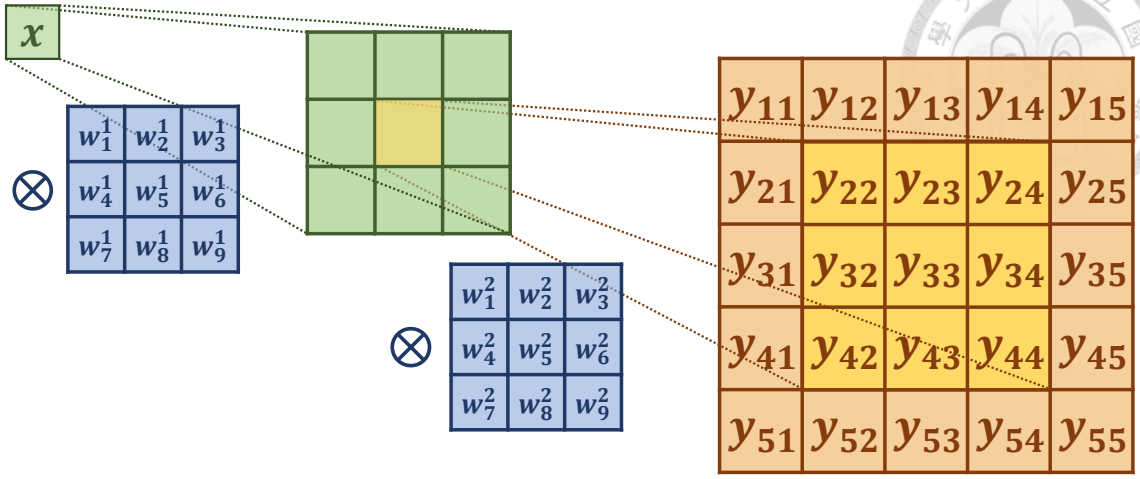


圖 4.4　兩層卷積層之 CNN[43]

考慮圖 4.中兩層卷積層的神經網路，每層皆為 3x3 卷積，各有一個卷積核(kernel)。根據上圖可以得到輸出y_{ij}與輸入x的關係式(4.1)：

$$y_{ij} = \sum_{(k,l)\in P_{ij}} x \cdot w_k^1 \cdot w_l^2$$

式(4.1)

其中P_{ij}為輸入x到輸出y_{ij}的所有路徑之集合。輸入乘上在每個路徑上第一層卷積核、第二層卷積核對應的權重後進行累加，計算出輸出之值。

而式(4.1)中的P_{ij}是輸入到單一輸出的路徑，要計算總路徑數，可以針對所有輸出節點加總式(4.2)：

$$\text{Total path count} = \sum_{ij} |P_{ij}|$$

式(4.2)

圖 4.中，每個卷積核使輸入影響到 9 個輸出，因此兩層卷積層總共有81條路徑，總路徑數為 81。

doi:10.6342/NTU202400902

網路中的批正規化層、skip-connection 等結構主要是為了幫助網路在訓練時收斂，並不會影響網路的表達能力。因此 Zen-NAS[48]計算時不考慮這些架構，只留下包含卷積層、線性層、激活函數的骨架(backbone)網路，稱為簡樸神經網路(Vanilla neural network, VNN)。

TPC-NAS 以相同的方式考慮整個網路的表達能力，經過上述的簡化，只需要考量每一層每個輸入到輸出的路徑數，再將網路中每一層的路徑數相乘，即可計算出整個網路的總路徑數。其中每個輸入到輸出的路徑數即是一個神經元連接到下一層神經元的出度(outdegree)。

對於卷積層而言，可以根據其設定的參數計算出對應的出度，如式(4.3)。其中 O_i 為網路中第 i 層卷積層之出度，$C_{out,i}$ 為卷積層的輸出通道個數，k_i 為卷積核的大小，s_i 為卷積之步長，g_i 為卷積分組計算的組數。

$$O_i = \frac{C_{out,i} \cdot k_i^2}{s_i^2 \cdot g_i}$$

式(4.3)

為了避免路徑總數在相乘後數值過大的問題，TPC score 取每一層的總路徑數的log再進行累加，達到累乘的效果。圖 4.5 為 TPC score 的計算流程。

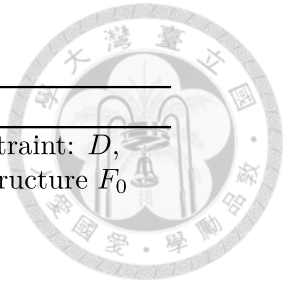
Algorithm: TPC Score Computation.

Data: Parameter of all convolution layers: $C_{out,i}, k_i, s_i, g_i$
Result: TPC score $\mathtt{S_t}$ of the model

$\mathtt{S_t} \leftarrow 0$;
Remove all residual links in the model ;
foreach *convolution layer* `i` *in the model* **do**
    $\mathtt{S_t} \leftarrow \mathtt{S_t} + \log \frac{C_{out,i} \cdot k_i^2}{s_i^2 \cdot g_i}$;
end

圖 4.5　TPC score 計算演算法[43]

doi:10.6342/NTU202400902



Algorithm: TPC-NAS Algorithm.

Data: Search space: S, Hardware constraint: K, Depth constraint: D, Number of iteration: M, Population size: N, Initial structure F_0
Result: Model with largest TPC score $\mathtt{F}_{\mathtt{max}} \in S$

```
1  Initialize population P ← {F_0} ;
2  for m = 1 to M do
3  |  F ← Randomly select one model from P ;
4  |  F̂ ← MUTATE(F, S) ;
5  |  if F̂ does not satisfy K or has more than D layers then
6  |  |  go to line 3 ;
7  |  else
8  |  |  Calculate TPC score of F̂ ;
9  |  |  Insert F̂ into population P ;
10 |  end
11 |  if |P| > N then
12 |  |  Remove the model with lowest TPC score from P ;
13 |  end
14 end
15 F_max ← the model with highest TPC score in P ;
```

圖 4.6　TPC-NAS 演算法[43]

Algorithm: MUTATE

Data: Search space: S, Structure F
Result: Mutated structure $\hat{\mathtt{F}}$

```
1 Randomly select a block b in F ;
2 Change the block type, kernel size, width, and depth of b randomly
    within search space S ;
```

圖 4.7　TPC-NAS 基因演算法之架構突變[43]

圖 4.6 為 TPC score 結合基因演算法後的 TPC-NAS 演算法流程。首先我們會在搜尋空間中建構一個非常小的模型F_0作為初始族群，使得演化初期容易產生符合硬體限制與深度限制的網路架構。接著依據圖 4.對架構進行突變，產生新的架構後再檢查是否有符合限制，只有符合硬體與深度限制的架構才會被計算 TPC score 並加到族群內。淘汰的方式則是選取 TPC score 最低的架構，將其從族群中移除。

最終族群內 TPC score 最高之架構即為 TPC-NAS 搜尋結果。

doi:10.6342/NTU202400902

4.2 實驗數據與結果

在此章節中，我們會使用 TPC-NAS 改進現有人工設計的模型，再進行遲滯二值化訓練。在相同硬體限制下，以 TPC-NAS 補償二值化帶來的正確率損失。

4.2.1 卷積神經網路(CNN)實驗結果

在搜尋空間的設定上，針對卷積神經網路，我們的搜尋空間會包含一般的卷積層，加上以ResNet[24]提出的瓶頸卷積為基礎的ConvK1KxK1模組。

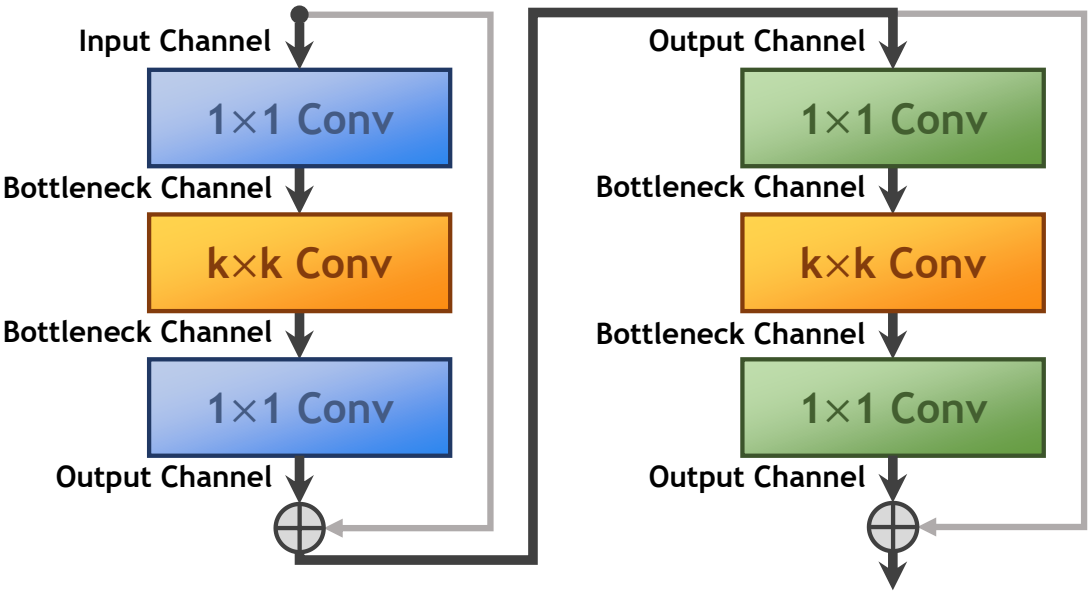


圖 4.8 ConvK1KxK1 模組示意圖

圖 4.8 為ConvK1KxK1之示意圖，圖中省略了所有批正規化層與激活函數。瓶頸卷積層主要是將普通的卷積層替換為三層卷積，首先先透過1×1卷積將輸入通道數降低至瓶頸通道數(bottleneck channel)，接著再經過k×k卷積實際進行捲積運算，最後再透過1×1卷積將結果還原回輸出通道數(output channel)。這樣設計下，複雜的k×k運算之輸入輸出通道數會被降低至瓶頸通道數，因此可以降低整體的計算量。

doi:10.6342/NTU202400902

4.2.1.1 影像辨識(Image Classification)實驗結果

使用卷積神經網路進行影像辨識時，TPC-NAS 主要搜尋卷積層的部分。在網路的最後一層會按照 TPC-NAS 搜尋結果中最後一層卷積層輸出維度加入對應的線性層以完成影像辨識的任務。

在資料集的部分這邊採用 CIFAR100 進行測試，FLOPs 是依據輸入 CIFAR100 中32×32大小的影像來計算。

表 4.1 ResNet18[24]於 CIFAR100[25]進行 TPC-NAS 結果

Model	TPC score	CIFAR100 fp32 Test Accuracy (%)	Params. (M)	FLOPs (M)
ResNet18	126	74.7	11.2	555.48
ResNet18 (TPC-NAS)	481	**81.3**	9.9	555.46

表 4.1 為使用 TPC-NAS 以 ResNet18 之參數量與計算量為限制搜尋之結果。在更少的參數量、更低的 FLOPs 下，TPC-NAS 搜尋到了更高分數(481)的架構，並且實際在 CIFAR100 訓練後準確率比原本高了 6.6%，顯示 TPC-NAS 能成功應用於影像辨識，改善 ResNet18 表現。

表 4.2 TPC-NAS ResNet18[24]於 CIFAR100[25]進行 1W4A 訓練結果

Model	TPC score	fp32 Test Acc. (%)	1W4A Test Acc. (%)	Diff. (%)
ResNet18	126	74.7	72.8	-1.9
ResNet18 (TPC-NAS)	481	81.3	**79.8**	-1.5

在量化至 1W4A 後，準確率下降至 79.8%，相較於使用 TPC-NAS 前二值化到 1W4A 的準確率也改善了 7%。

doi:10.6342/NTU202400902

這邊我們考慮權重與 activation 之位寬(bitwidth)，引入混合精度量化(Mixed precision quantization)常用的指標BitOps[49]。式(4.4)為BitOps的計算方式，為網路中運算的浮點運算量，乘上該運算量化後的權重與 activation 之位寬即可得到BitOps。BitOps雖然不能精準的評估能量與計算延遲，但是能夠避免不同硬體造成的影響，使不同量化方法能夠比較公平地被比較。

$$BitOps(m) = FLOPs(m) \cdot \text{weight-bit}(m) \cdot \text{act-bit}(m)$$

式(4.4)

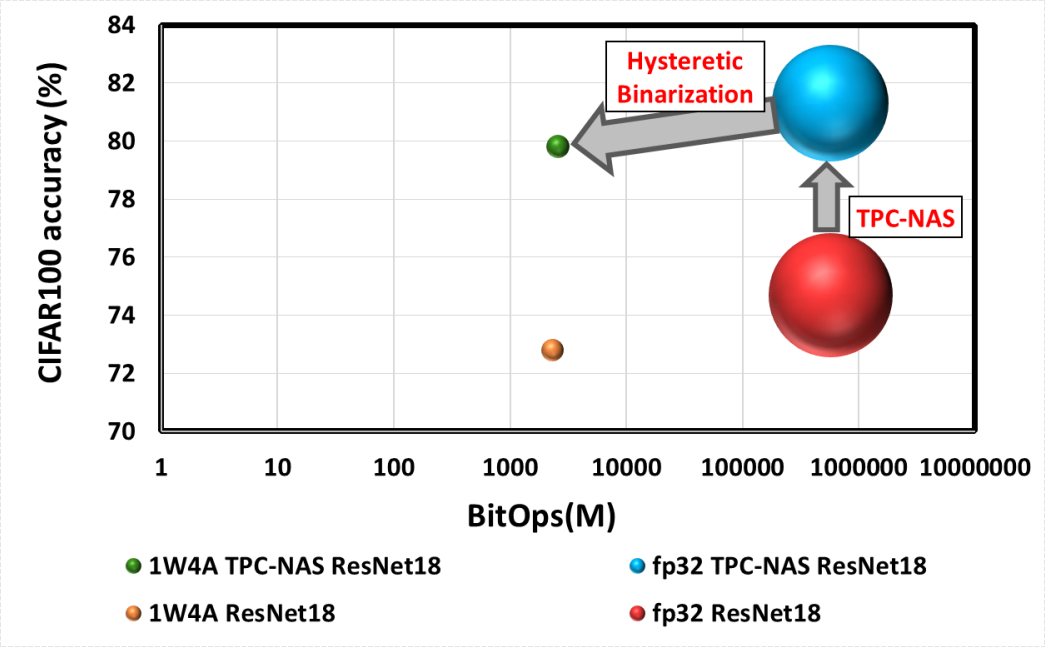


圖 4.9　ResNet18 TPC-NAS 結合遲滯二值化於 CIFAR100 訓練結果

透過以上BitOps的定義，我們畫出圖 4.9 的泡泡圖。其中橫軸為BitOps，縱軸為在 CIFAR100 訓練後的測試準確率，泡泡大小為模型參數大小(MB)。

結合了 TPC-NAS 和遲滯二值化訓練後，我們能夠節省相較於原始 ResNet18 220 倍的 BitOps 的計算量。在模型大小方面，考慮 scale 以及 zero point 等非二值化參數後，相較原本的 ResNet18 也小了 30.4 倍。二值化網路的準確率到達 79.8%，也比原先高了 5.1%。

doi:10.6342/NTU202400902

4.2.1.2 物件偵測(Object Detection)實驗結果

在使用卷積神經網路進行物件偵測的部分還是使用先前提到的 SSD[29]為基礎，選擇以 ResNet50[24]為骨幹(backbone)的 ResNet50-SSD 作為比較對象。

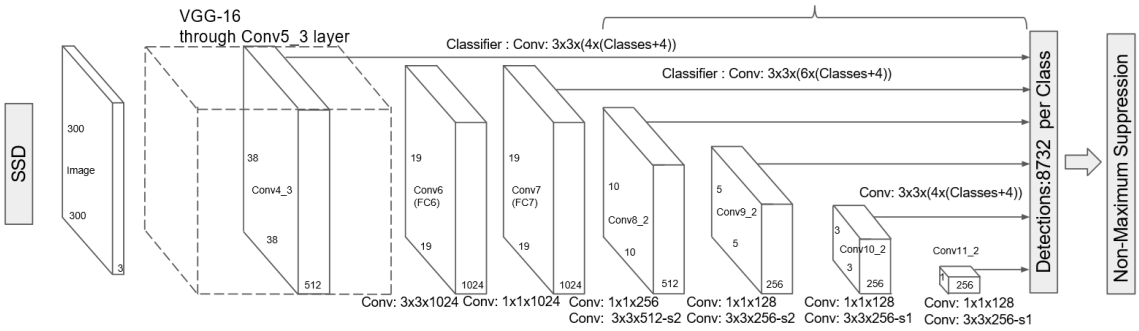


圖 4.10　VGG16-SSD 架構[29]

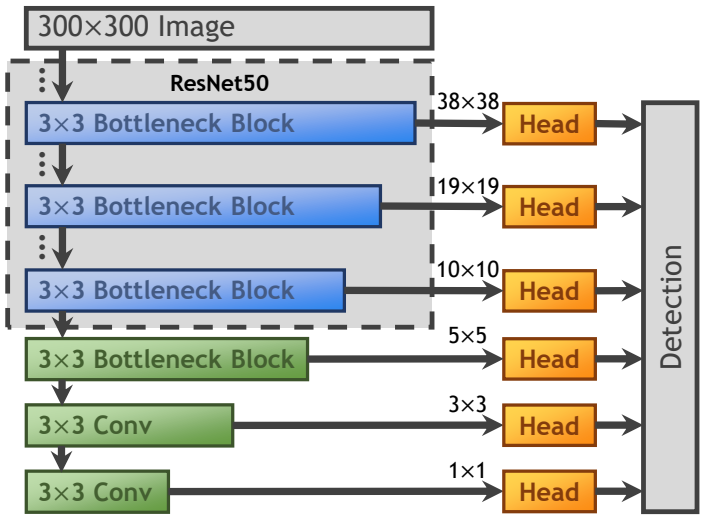


圖 4.11　ResNet50-SSD 架構

圖 4.10 與圖 4.11 分別為 VGG16-SSD 與 ResNet50-SSD 之架構圖。其中骨幹網路指的是圖 4.10 中的 VGG16 與後續降採樣至1×1圖片大小的卷積層。在圖 4.11 則是 ResNet50 與綠色卷積的部分。在兩個架構中都是由骨幹網路取出對應另種圖片大小的特徵，經過 head 預測出 bounding box 座標與大小，以及物件所屬之類別。而這裡主要用 TPC-NAS 搜尋骨幹網路，在搜尋結果取出與上圖相同圖片大小之特徵後，head 的輸入維度即被決定，因此不需要搜尋 head 之架構。

doi:10.6342/NTU202400902

表 4.3　ResNet50-SSD 於 Pascal VOC[28]進行 TPC-NAS 結果

Model	TPC score	Pascal VOC fp32 Test Accuracy (%)	Params. (M)	FLOPs (G)
ResNet50-SSD	367	74.1	39.6	8.07
ResNet50-SSD (TPC-NAS)	399	**79.2**	37.9	7.15

表 4.3 是以 ResNet50-SSD 參數量及 FLOPs 為限制下使用 TPC-NAS 進行架構搜尋後得到的結果，表中參數量與 FLOPs 包含骨架網路與 head。TPC-NAS 搜尋後得到 399 的 TPC score 的架構，並且在 fp32 測試準確率上提升了 5.1%。

表 4.4　TPC-NAS ResNet50-SSD 於 Pascal VOC 進行 1W4A 訓練結果

Model	TPC score	fp32 Test Acc. (%)	1W4A Test Acc. (%)	Diff. (%)
ResNet50-SSD	367	74.1	71.0	-3.1
ResNet50-SSD (TPC-NAS)	399	79.2	**74.8**	-4.4

表 4.4 將 TPC-NAS 搜尋結果分別進行二值化，二值化後準確率下將約為 3-4%。與原始 1W4A ResNet50-SSD 相比，進行 TPC-NAS 後準確率高了 3.8%，比原先 fp32 的 ResNet50-SSD 還高。

doi:10.6342/NTU202400902

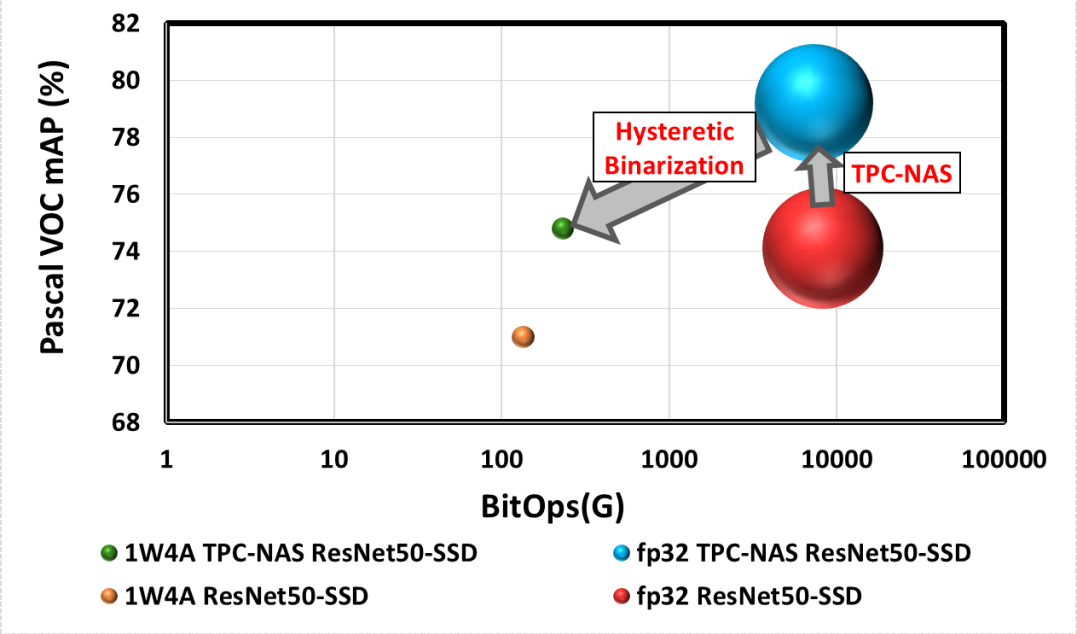


圖 4.12　ResNet50-SSD TPC-NAS 結合遲滯二值化於 Pascal VOC 訓練結果

計算BitOps後，主要的計算量落在網路最後一層使用較高精度的 head 上，也使 TPC-NAS 搜尋到的模型在二值化後整體運算量比原始的 ResNet50-SSD 高。

不過相比 fp32 的 ResNet50-SSD，經過了 TPC-NAS 搜尋以及二值化訓練後，我們能夠節省 31.3 倍的計算量、35.74 倍的參數大小，並改進 0.7%的準確率。

4.2.2 Transformer 實驗結果

前一個章節將 TPC-NAS 結合遲滯二值化訓練，應用於卷積神經網路模型的影像辨識與物件偵測上。

本章節以 transformer 模型為基礎，應用於影像辨識與自然語言處理之應用，評估 TPC-NAS 結合遲滯二值化訓練的成效。

doi:10.6342/NTU202400902

4.2.2.1 影像辨識(Image Classification)實驗結果

在影像辨識任務上採用 ImageNet100 資料集，模型的部分選擇使用 FastVit[50]。FastVit 使用重新參數化(reparameterization)的技巧將訓練時不同路徑的權重合併，在推論時只會留下單一分支，能夠避免 skip-connection 造成推論效率降低。此外 FastViT 為了提高參數效率以及降低推論延遲，採用了混合架構(hybrid architecture)，在網路前期使用較多卷積層，與 transformer block 結合能更好的捕捉局部與全局訊息。

表 4.5　不同 transformer 模型 ImageNet100 準確率比較

Model	Params. (M)	FLOPs (G)	ImageNet100 Accuracy (%)
DeiT-base	86.6	17.6	87.4
DaViT-tiny	28.3	4.5	85.1
FastVit-sa24	**20.6**	**3.8**	85.6

表 4.5 為目前用於影像辨識之三種 vision transformer 模型之比較。在準確率、參數量、計算量上，FastViT-sa24 都優於 DaViT-tiny。而相比 DeiT-base，FastViT 之參數量少了 4.2 倍，計算量少了 4.6 倍。

與先前卷積神經網路不同，transformer 模型主要由線性層組成，且包含了注意力層(attention layer)。在線性層的部分根據 TPC score 的定義使用輸出通道數計算分數。在注意力層中 Q, K, V 額外的運算因為只是輸出的矩陣轉換，並沒有額外資料參與計算，所以不會增加 TPC score。

doi:10.6342/NTU202400902

表 4.6 FastViT-sa24[50]於 ImageNet100 進行 TPC-NAS 結果

Model	TPC score	ImageNet100 fp32 Test Accuracy (%)	Params. (M)	FLOPs (G)
FastViT-sa24	519	85.6	20.6	3.8
FastViT-sa24 (TPC-NAS)	634	**85.8**	19.8	4.1

表 4.7 TPC-NAS FastViT-sa24 於 ImageNet100 進行 1W4A 訓練結果

Model	TPC score	fp32 Test Acc. (%)	1W4A Test Acc. (%)	Diff. (%)
FastViT-sa24	519	85.6	70.1	-15.5
FastViT-sa24 (TPC-NAS)	634	85.8	**79.3**	-6.3

表 4.6 為 TPC-NAS 搜尋出來的架構與原始的 FastViT-sa24 比較，雖然 TPC score 有顯著的提升，但是在測試準確率的提升不如之前高。

表 4.7 則將兩個模型各自進行二值化訓練，結果顯示 TPC-NAS 尋找出的模型在二值化後比較能維持其準確率。

觀察 TPC-NAS 的搜尋結果可以發現，相較於原先的 FastViT-sa24，TPC-NAS 將網路前端的卷積層寬度增加，使這部分的參數增加為原先的 5.14 倍，運算量增加為原先的 6.21 倍。由[51]可以知道增加卷積層寬度可以改善低精度神經網路量化後之準確率，而在這裡 TPC-NAS 將網路後期寬度較寬的部分略為縮減，將計算量分配給前期寬度較窄的卷積層，改善了二值化網路的準確率。

doi:10.6342/NTU202400902

4.2.2.2 自然語言處理(Natural Language Processing, NLP)實驗結果

這邊使用與[43]相同的設定，以 BERT-tiny[52]為限制，使用 TPC-NAS 搜尋線性層中的輸入輸出通道數、multi-head attention 中的 head 個數等。在搜尋完架構後使用 Common Crawl 與 Wikipedia dataset 進行預訓練。最後以 GLUE[15]的多項任務進行 fine-tune，評估模型表現。

表 4.8　BERT-tiny[52]進行 TPC-NAS 結果

Model	TPC score	GLUE Avg.	FLOPs (M)
BERT-tiny	66	67.2	38.6
BERT-tiny (TPC-NAS)	133	**72.6**	38.4

表 4.9　TPC-NAS BERT-tiny[52]於 GLUE[15]各任務 1W4A 訓練結果

Model Precision	MNLI m/mm	QQP	QNLI	SST-2	MRPC Acc.	MRPC F1	RTE	WNLI	Avg.
BERT-tiny fp32	63.0/62.7	81.0	60.8	81.6	68.4	81.2	56.0	49.8	**67.2**
BERT-tiny (TPC-NAS) fp32	68.1/67.6	85.0	84.6	81.3	70.5	81.6	58.1	56.3	**72.6**
BERT-tiny (TPC-NAS) 1W4A	67.4/67.6	82.8	73.0	83.3	71.3	81.5	58.4	56.3	**71.3**

表 4.9 為 TPC-NAS 結合遲滯二值化訓練結果，TPC-NAS 找到 TPC score 且計算量相近的模型，在實際訓練後準確率高了 5.4%。進行 1W4A 訓練後準確率降低了 1.3%，但還是比最初的 fp32 BERT-tiny 高了 4.1%。

doi:10.6342/NTU202400902

4.3 第四章總結

在第四章中，首先介紹了神經網路架構搜尋的演算法種類，顯示 TPC-NAS 的優點，接著描述 TPC-NAS 基於總路徑數來評估架構的表達能力。

最後結合先前的遲滯二值化訓練，在卷積神經網路應用於影像辨識、物件偵測，以及 transformer 用於影像辨識、自然語言處理上，TPC-NAS 都能改善二值化後網路的準確率，在其中三個實驗中達到超越原始 fp32 模型的表現。

doi:10.6342/NTU202400902

doi:10.6342/NTU202400902

第五章 二值化神經網路推論電路

本章節探討二值化神經網路推論運算於電路上實作之效能，以邏輯閘級模擬(gate-level simulation)結果比較不同精度、近似後之面積與功耗。

5.1 推論電路效能分析

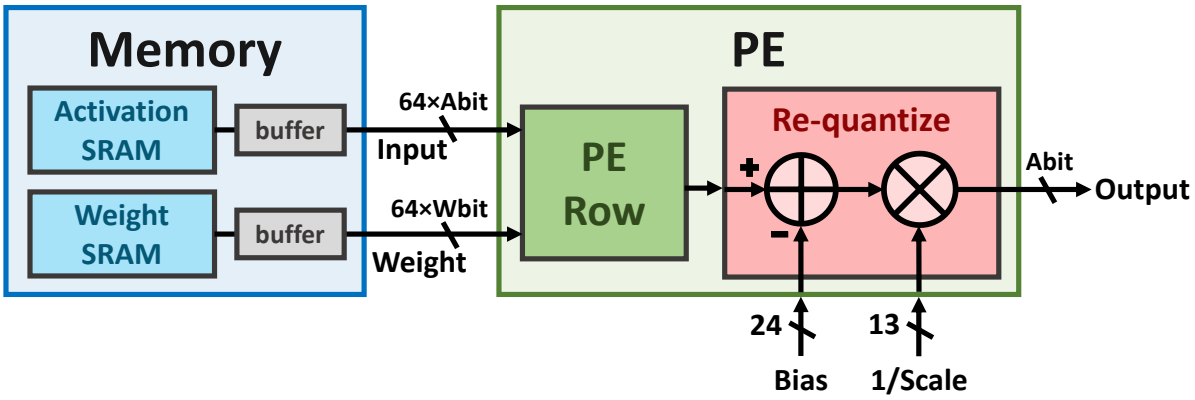


圖 5.1 量化神經網路推論計算架構

這邊主要計算神經網路推論運算中權重矩陣與輸入矩陣相乘的運算，卷積層的部分可以使用 image to column 重排為矩陣乘法。圖 5.1 為比較不同精度推論運算之硬體架構圖，權重與 activation 會存放於 SRAM 中，在 PE row 中進行 64 組權重與 activation 之內積，最終再進行式(2.2)之重新量化(re-quantize)將結果重新量化為整數。

其中重新量化時使用的 bias 以及 scale 經過軟體模擬推論誤差後，將 fp32 的值量化為 24-bit 以及 13-bit 之定點數。Scale 的部分會事先取倒數以將除法轉為乘法計算。

doi:10.6342/NTU202400902

這裡分別使用 1W1A、1W4A、8W8A 與 fp32 之精度，在 TSMC .13 製程上以 Design Compiler 合成並以 PrimeTime 分析多組輸入資料下的功耗。

在不同精度中會採用相同的吞吐量(throughput)來比較，為了滿足吞吐量，在高精度下會使用較多的 SRAM 以提高同時間讀取權重與輸入的數量。

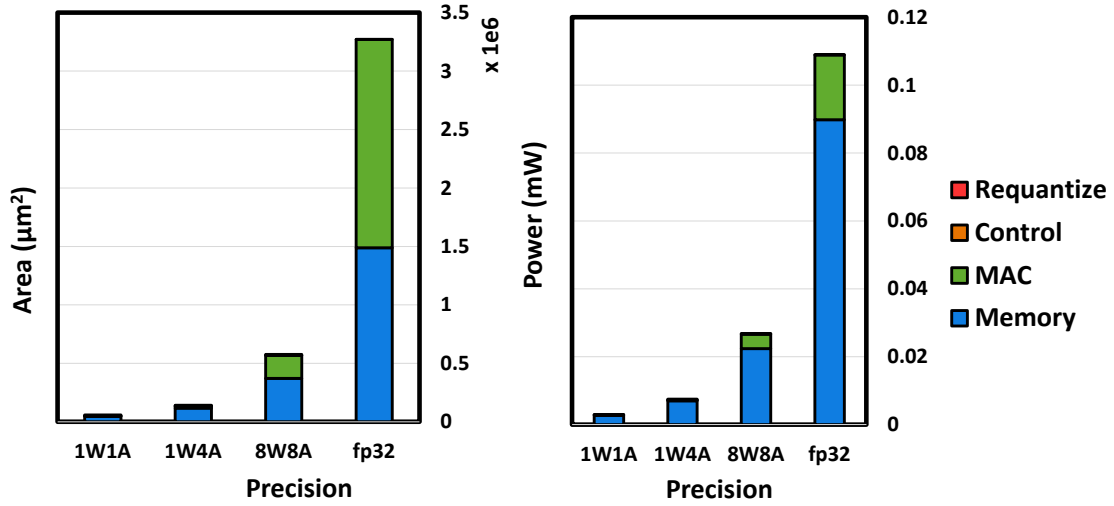


圖 5.2　不同精度下推論電路面積與功耗比較(TSMC .13 Process)

相比 fp32，1W1A、1W4A 與 8W8A 之推論電路在面積與功耗上皆有明顯的改進。將 fp32 之運算量化至 8W8A 後，運算單位變為 8-bit 整數且多出重新量化的部分，不過這部分的運算相比 fp32 還是節省了 18.7 倍的面積與 43.0 倍的功耗。

在 memory 的部分由 fp32 降至 8W8A 可以節省 4 倍，再降至 1W4A 則可以再節省 3.2 倍，在降至 1W1A 則節省 2.5 倍。

整體而言相比 fp32，1W4A 的面積節省了 30.9 倍，功耗則省了 34.3 倍。1W1A 則省 55.7 倍的面積與 37.59 倍的功耗。

doi:10.6342/NTU202400902

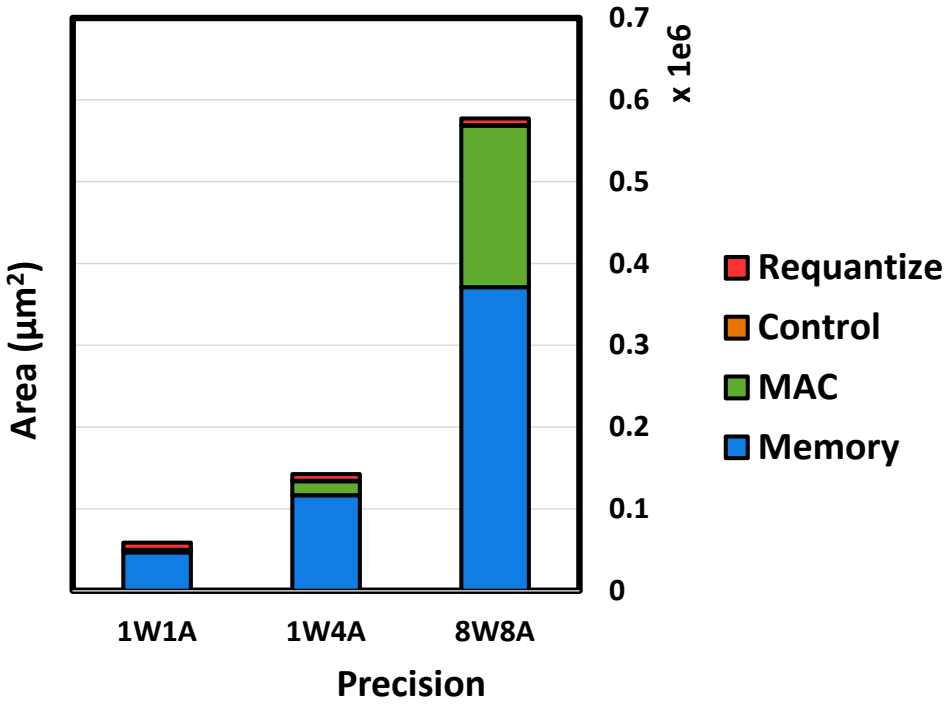


圖 5.3　1W1A、1W4A 與 8W8A 推論電路面積比較(TSMC .13 Process)

在面積的部分 1W4A 電路整體比 8W8A 節省了 3.57 倍。主要佔較多面積的是計算內積以及 memory 的部分，控制電路與重新量化的部分相對較小。1W1A 則相比 1W4A 又再省了 2.42 倍的面積。

若排除 memory，單獨比較計算與控制電路，1W4A 的面積比 8W8A 省了 4.66 倍，1W1A 則相比 1W4A 省了 2.2 倍。而 1W4A 的精度在相同的吞吐量下可以節省 3.2 倍的 memory，因此精度下降至 1W4A 後 memory 佔整體電路的面積由 66.8% 提升至 74.6%。在 1W1A 電路中 memory 則占整體的面積 79.7%。

doi:10.6342/NTU202400902

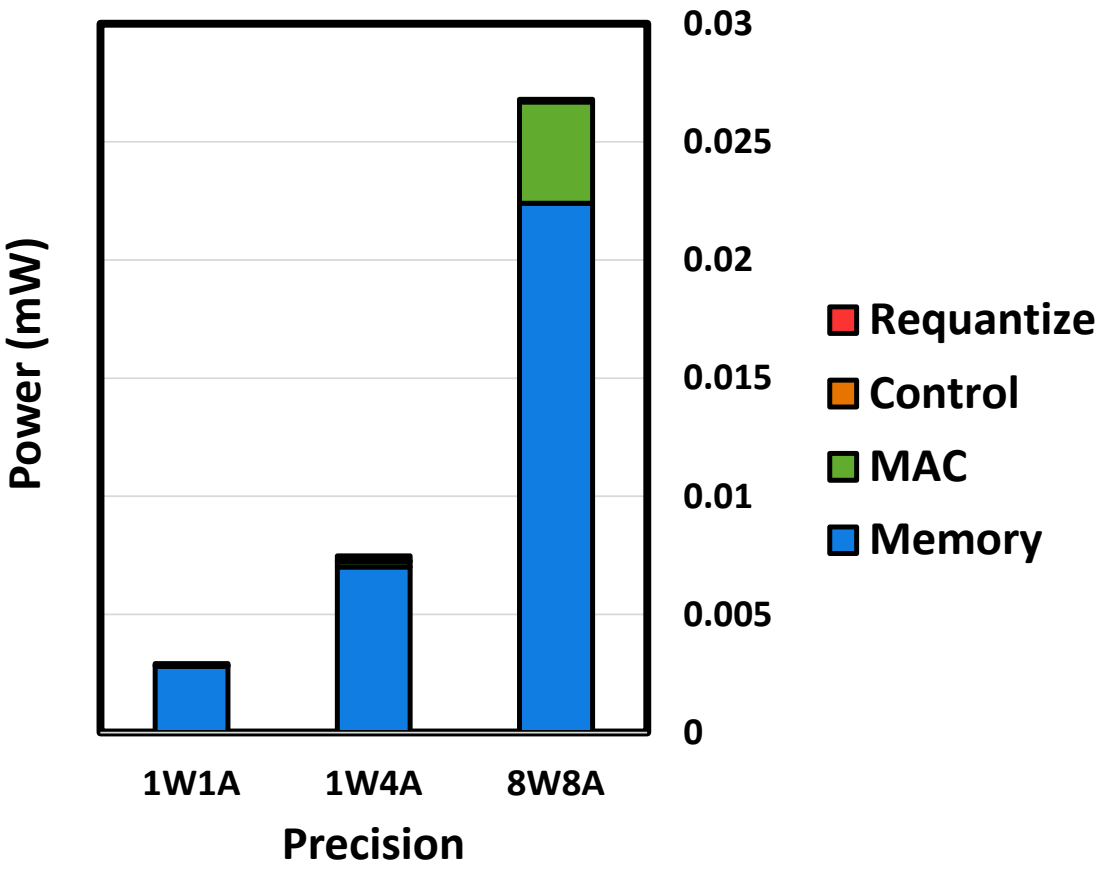


圖 5.4　1W1A、1W4A 與 8W8A 推論電路功耗比較(TSMC .13 Process)

在功耗上整體 1W4A 比 8W8A 省了 3.07 倍，1W1A 則又比 1W4A 省了 2.41 倍。不過在功耗的主要還是 memory 佔大部分，從 8W8A 到 1W4A 後 memory 的功耗降低了 3.24 倍，1W4A 降至 1W1A 的 memory 功耗則降低 2.50 倍。

整體而言，精度從 8W8A 降低至 1W4A 後可以節省 3.57 倍的面積與 3.07 倍的功耗，8W8A 降至 1W1A 則可以節省 9.82 倍的面積與 9.31 倍的功耗。顯示二值化後對於面積與功耗皆能有顯著的改進。

doi:10.6342/NTU202400902

5.2 乘加運算近似電路

本章節介紹二值化網路推論之實作，以及提出基於 OAI 運算之近似電路。

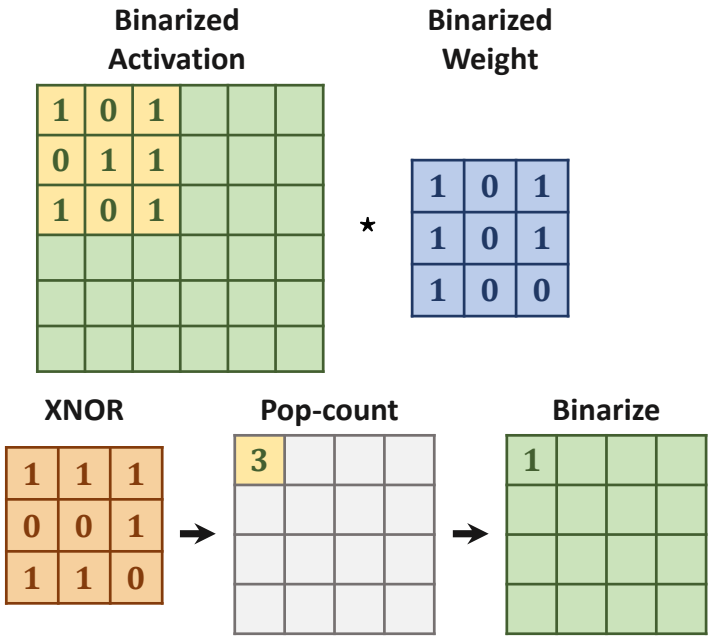


圖 5.5 XNOR pop-count 實現二值化卷積示意圖

在二值化神經網路運算的部分，XNOR-Net[39]以 0 代表−1，1 代表+1，當權種與 activation 皆被量化至 1-bit 時，二值化的乘加運算可以被 XNOR 與 pop-count 運算取代。

圖 5.5 為二值化卷積以 XNOR 與 pop-count 實現之示意圖。首先±1之乘法剛好能夠對應到 XNOR 運算，當權重與 activation 之正負號相同時，結果則為正，否則結果為負。接著再根據 pop-count 做累加，圖中 pop-count 結果有 6 個+1，減去剩餘的 3 個−1得到結果為 3，最後再進行二值化得到 1。實際在運算時不需要計算出−1的個數，只需要計算+1的個數乘以 2 再減去累加的個數即可。

doi:10.6342/NTU202400902

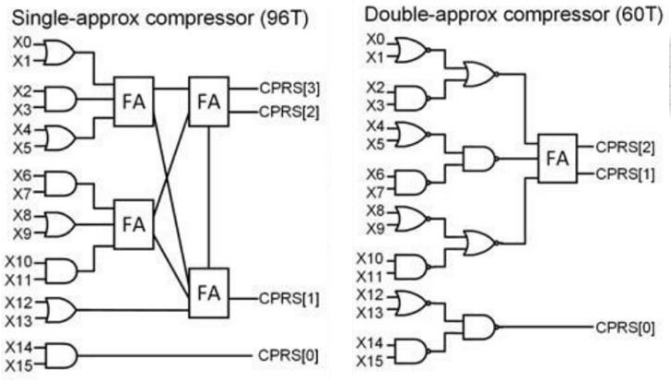


圖 5.6　以 NAND, NOR 近似加法樹示意圖[53]

[53]使用 SRAM，以記憶體內運算(In-memory computing)的方式實現前述 XNOR 的部分。在加法樹的部分則採用了近似運算，圖 5.6 為近似運算之示意圖，其中加法樹的第一級或第二級分別由交錯的 NAND, NOR 運算取代。兩種邏輯閘取代全加器，將兩個輸入結合成 1-bit 輸出，其中 AND 代表無條件捨去，OR 代表無條件進位。

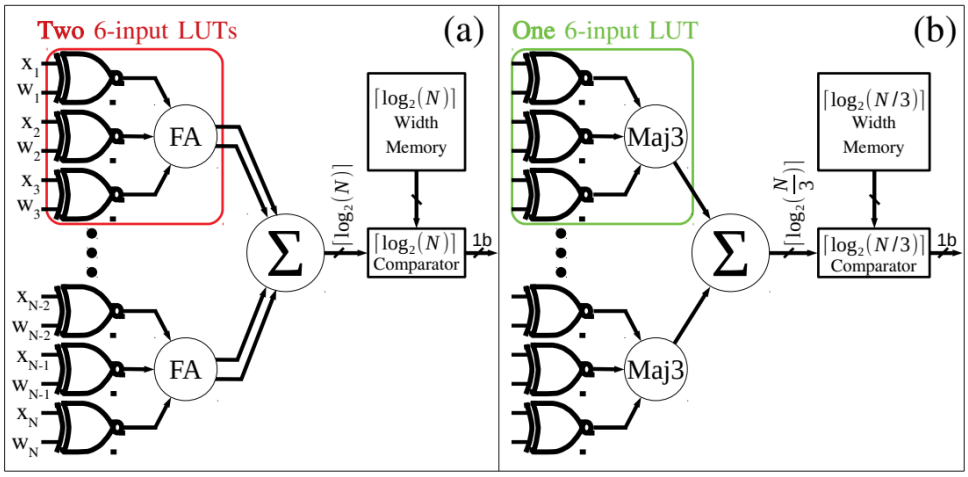


圖 5.7　以 3-input majority 近似加法樹示意圖[54]

[54]則採用了 3-input majority (Maj3)近似加法樹。每輸入三個 bit，其加總的結果可能為0, 1, 2, 3，Maj3 則將結果近似為0, 2，以出現次數最多之數值近似總和。此外 majority 近似可以透過減去平均誤差的方式進一步提升近似準確率。

doi:10.6342/NTU202400902

表 5.1 Maj3, OAI21 近似加總之比較

Inputs (A0, A1, B0)	Sum	Maj3.	OAI21 (inputs inverted)
(0,0,0)	0	0	0
(0,0,1)	**1**	**0**	**2**
(0,1,0)	**1**	**0**	**0**
(0,1,1)	2	2	2
(1,0,0)	**1**	**0**	**0**
(1,0,1)	2	2	2
(1,1,0)	2	2	2
(1,1,1)	**3**	**2**	**2**

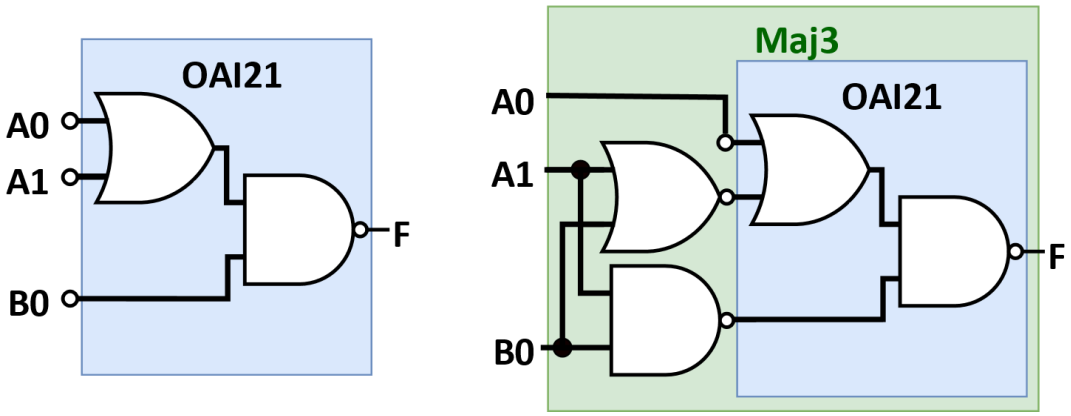


圖 5.8 OAI21 近似與 Maj3 實現

在標準元件庫中包含了 OAI21 的 cell，主要是先經過 OR，再做 AND，最後 invert。表 5.1 比較了 Maj3 與輸入反向後的 OAI21 相對於真實加總的差異。粗體的部分為與真實加總不同之輸出，可以看到 Maj3 與 OAI21 之錯誤個數是相同的，而他們兩者不同的地方在於輸入為(0,0,1)時，此時兩者皆為錯誤的輸出。

而 OAI21 的好處在於它是 standard cell，在實現上會比 Maj3 簡單。從圖 5.8 可以看出以 AND, OR 為基礎實現 Maj3 的邏輯會比 Maj3 複雜，雖然 Maj3 不只有一種實現方式，但是基本上都會比 OAI21 占用較多面積。

doi:10.6342/NTU202400902

無論是 Maj3 還是 OAI21，都是從 XNOR 後的輸出開始近似。如果我們在 XNOR 運算時將 XNOR 改為以其他 cell 替代，則可以進一步做化簡，與後面的邏輯做結合。

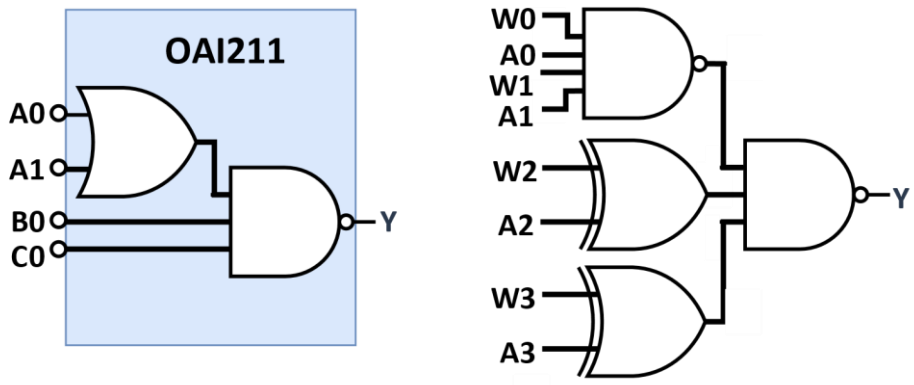


圖 5.9　OAI211 與 8-1 乘加近似

將 OAI21 替換為 4-input 的 OAI211 後，並將前兩個 input 對應的 XOR 與 OR 替換為 4-input NAND 得到 8-1 乘加進似電路。由於少去了兩個 XOR，相比 OAI211 能進一步降低面積與功耗。

5.2.1 邏輯閘級模擬結果

這邊以邏輯閘級模擬各種近似電路，比較各個電路計算 1W1A 內積時的面積、功耗以及時間。

在誤差部分取近似值與實際內積值之均方誤差(MSE)以及計算訊噪比(signal-to-noise ratio, SNR)。

doi:10.6342/NTU202400902

表 5.2　512-bit 1W1A 內積近似電路誤差、面積與功耗(TSMC .13 Process)

Approximation	MSE	SNR (dB)	Area (μm²)				Total Power (mW)
			Prod.	Approx.	Adder Tree	Total	
No approximation	0	Inf.	6083	0	17181	23264	0.725
XNOR, 1 lvl. Maj3[54]	42.7	31.87	6083	3758	5579	15420	0.418
XNOR, 1 lvl. OAI21	74.5	**29.45**	6083	1159	5579	**12821**	**0.340**
XNOR, 2 lvl. Maj3[54]	100.1	28.17	6083	5016	1767	12866	0.320
XNOR, 2 lvl. NAND NOR[53]	184.3	25.52	6083	1954	4186	12223	0.279
XNOR, 1 lvl. OAI211	78.4	**29.23**	6083	1086	4186	**11355**	**0.265**
1 lvl. 8-1 Prod-Sum	92.2	**28.52**	4997		4186	**9183**	**0.216**

表 5.2 比較了各種近似方法的誤差、面積與功耗，表格中各方法以功耗降序排序。兩級的 NAND NOR 近似雖然在功耗上降低約 2.6 倍，但是在誤差上是所有方法中最高的，實際測試後發現交替使用 NAND NOR 使得平均誤差接近為 0，無法透過扣除平均誤差來改善近似誤差。

此外 OAI211 能夠達到接近兩級 Maj3 近似之準確度，但面積與功耗更低。與未近似前相比，OAI211 能節省 2.04 倍的面積與 2.73 倍的功耗。額外近似了 XOR 部分的 8-1 乘加近似則可以進一步降低計算成本，減少 2.53 倍的面積與 3.35 倍的功耗。

doi:10.6342/NTU202400902

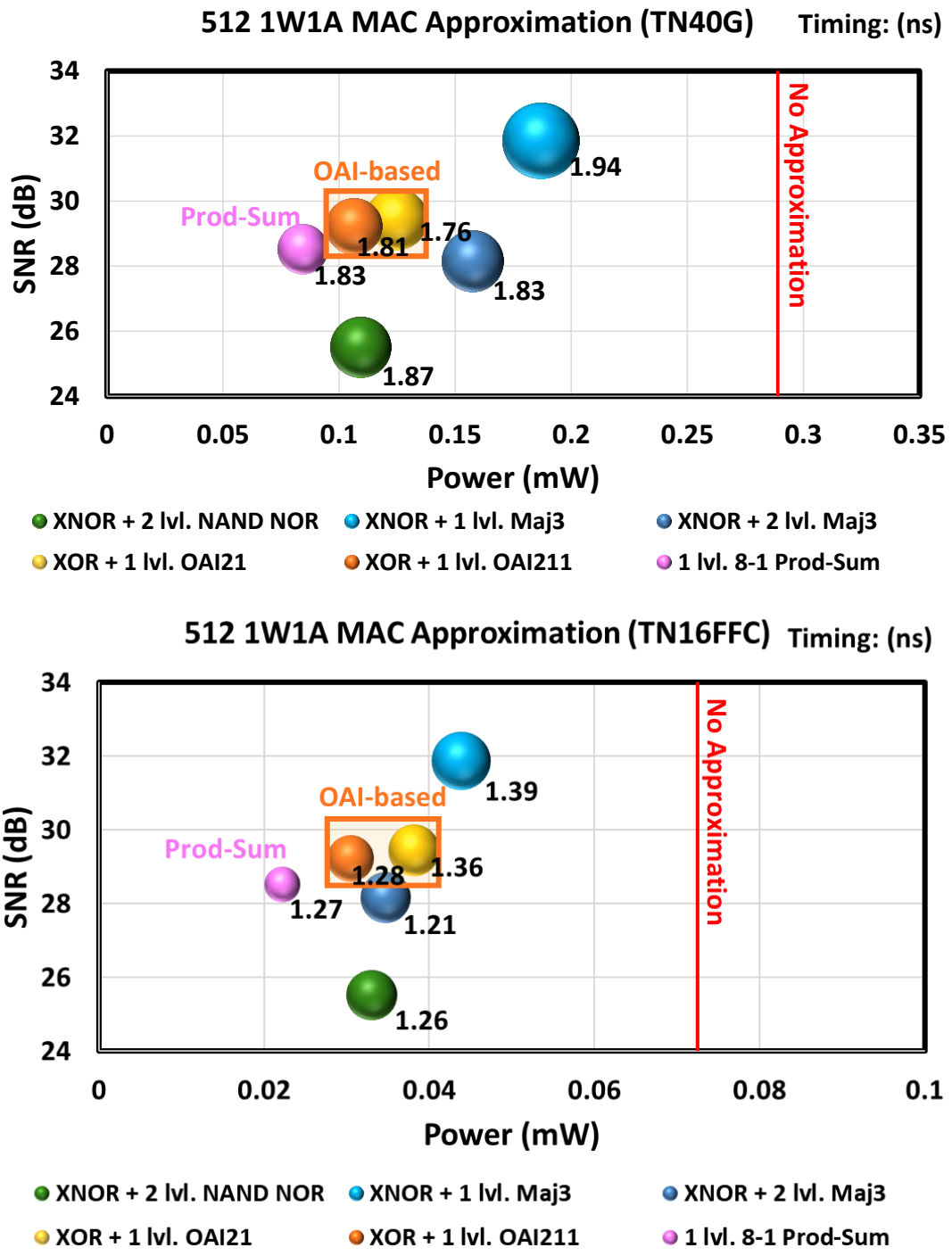


圖 5.10　512-bit 1W1A 內積近似電路於不同製程合成結果

doi:10.6342/NTU202400902

這裡分別於 TSMC.13、TN40G、TN16FFC 三製程上進行電路模擬，近似電路的相對關係並沒有太大的變動，因此圖 5.10 列出 40、16nm 製程之模擬結果。在使用微縮的製程下，近似電路仍然有不錯的表現。8-1 乘加電路在 16nm 製程相對於不做近似的內積電路仍然節省 2.36 倍的面積與 3.25 倍的功耗。與其他近似電路的相對關係也沒太多變化。

5.2.2 近似感知訓練

先前的模擬中使用均方誤差與訊噪比來評估近似誤差，但是實際應用於神經網路推論應該以準確率來評估。

為了確保加入近似後能夠維持準確率，這邊採用了近似感知訓練。

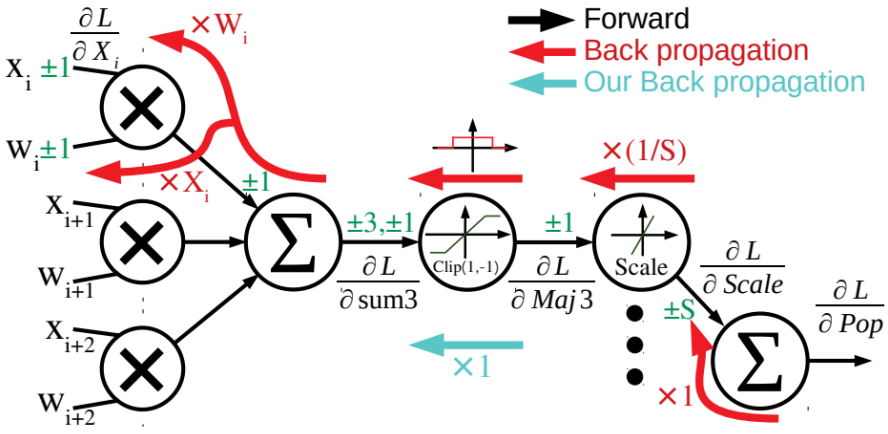


圖 5.11 1W1A Maj3 內積近似感知訓練正向反向傳播計算[54]

圖 5.11 為[54]訓練一級 Maj3 近似採用之方法，要模擬 Maj3 需要將矩陣乘法中的內積拆成三個一組的加總，再經過clip將值限制在±1以內，模擬 Maj3 計算。

在反向傳播時經過clip，則可以使用其定義計算梯度，取代 Maj3 的微分。

doi:10.6342/NTU202400902

對於輸入經過反向的 OAI21，我們可以參考模糊邏輯(fuzzy logic)將邏輯對應到等效的函數。對於分別對應到0, 1之輸入±1，inverter 相當於是取負號，而 AND 可以對應到取最小值，OR 則是取最大值。因此我們的輸入反向 OAI21 會變為式(5.1)，此時可以計算近似函數之導數，以用於梯度下降演算法。

$$\mathrm{OAI21}(\neg a_0, \neg a_1, \neg b_0) = -\min\left(\max\left(-a_0, -a_1\right), -b_0\right)$$

式(5.1)

其中最大值對應之導數可以由式(5.2)計算，最小值計算方式依此類推。

$$\max(a,b) = \begin{cases} a, & a \geq b \\ b, & a < b \end{cases}$$
$$\frac{\partial \max(a,b)}{\partial a} = \begin{cases} 1, & a \geq b \\ 0, & a < b \end{cases}$$
$$\frac{\partial \max(a,b)}{\partial b} = \begin{cases} 0, & a \geq b \\ 1, & a < b \end{cases}$$

式(5.2)

依據式(5.2)可以求得 OAI21 之偏微分式(5.3)：

$$\frac{\partial \mathrm{OAI21}(\neg a_0, \neg a_1, \neg b_0)}{\partial a_0} = \begin{cases} 1, & a_1 \geq a_0 \geq b_0 \\ 0, & \text{otherwise} \end{cases}$$
$$\frac{\partial \mathrm{OAI21}(\neg a_0, \neg a_1, \neg b_0)}{\partial a_1} = \begin{cases} 1, & a_0 > a_1 \geq b_0 \\ 0, & \text{otherwise} \end{cases}$$
$$\frac{\partial \mathrm{OAI21}(\neg a_0, \neg a_1, \neg b_0)}{\partial b_0} = \begin{cases} 1, & b_0 > \min(a_0, a_1) \\ 0, & \text{otherwise} \end{cases}$$

式(5.3)

doi:10.6342/NTU202400902

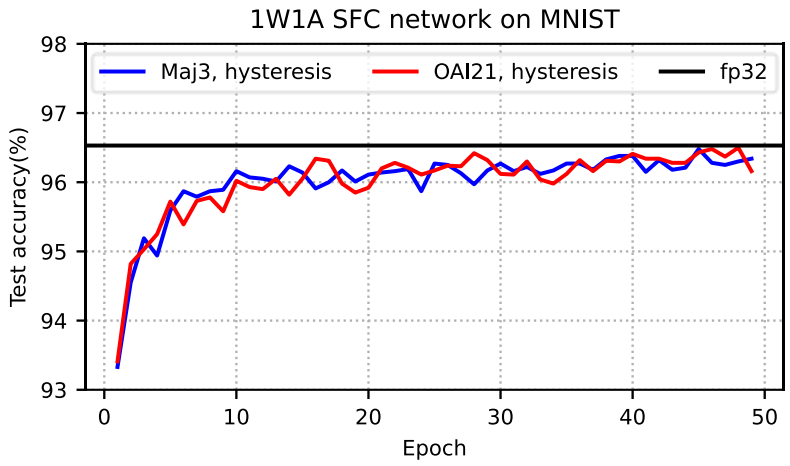


圖 5.12　1W1A SFC[54]模型於 MNIST 進行近似感知訓練結果

圖 5.12 為 OAI21 近似與 Maj3 近似應用於四層線性層之 SFC[54]模型於 MNIST 進行近似感知訓練之測試準確率。在使用遲滯二值化的訓練下，兩種近似方法皆能達到 fp32 模型之準確率。

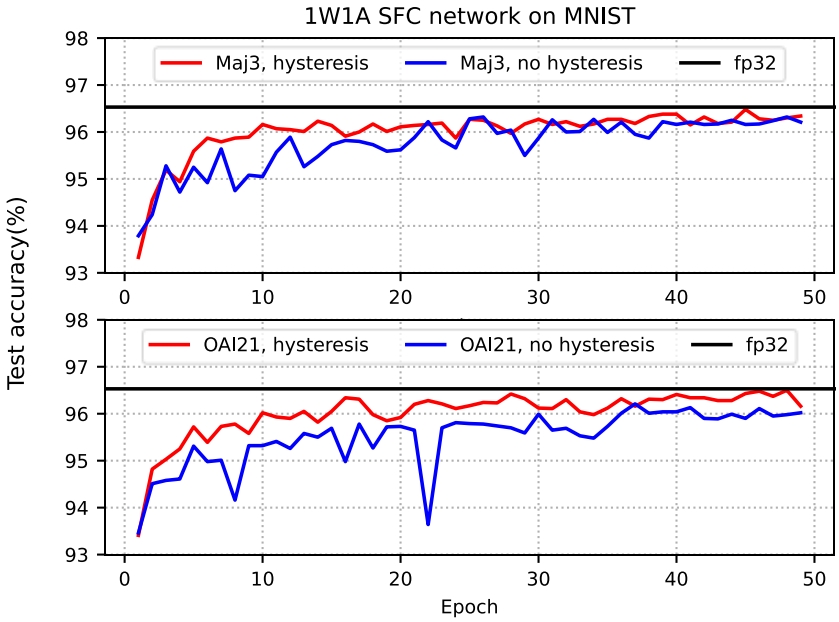


圖 5.13　1W1A SFC[54]模型加入遲滯前後進行近似感知訓練結果

圖 5.13 能看出在加入遲滯後對於 Maj3 以及 OAI21 之近似感知訓練之穩定性能有所提升。

doi:10.6342/NTU202400902

5.3 第五章總結

在本章節中首先比較 fp32、8W8A、1W4A、1W1A 精度下進行推論運算在硬體上的面積與功耗，顯示權重二值化之優勢。在面積上 1W4A 比 8W8A 省了 3.57 倍，在功耗上則省了 3.07 倍。1W1A 則相較 8W8A 省 9.82 倍的面積與 9.31 倍的功耗。

接著以 XNOR pop-count 為基礎，模擬各種 1W1A 推論近似硬體。並在不同製程上進行模擬，顯示我們基於 OAI 提出的近似方法能夠有效節省面積與功耗，並在不同製程下都能維持其優勢。

最後於 MNIST 上進行近似感知訓練，以模糊邏輯的方式計算邏輯運算函數之梯度，透過訓練時模擬近似函數以改進近似後準確率。並且以 OAI21 近似為例訓練，與 Maj3 近似相比較。證明 OAI21 是可行的近似方式，能達到甚至超越 Maj3 之表現，並且在硬體實現上比 Maj3 面積更小且功耗更低。

doi:10.6342/NTU202400902

第六章 結論與展望

隨著神經網路發展，各類應用逐漸影響我們的生活。在商品化的同時，如何在終端裝置以高能效的方式進行神經網路推論運算，同時維持準確率成為了研究的重心之一。

因應上述需求，本文基於整數線性量化結合各種壓縮模型之訓練技巧，實現了在不改變模型架構下將權重進行二值化之流程。並且於影像辨識、物件偵測、自然語言處理之任務上，使用卷積神經網路、transformer 之架構皆能在二值化的高度壓縮後維持可接受的準確率。此外為了提高二值化訓練時的穩定性，去除震盪的情型，我們在二值化時加入了遲滯之效應，在各種模型以及任務上測試都能相對於普通的二值化有所改進。

為了進一步提高二值化模型的表現，我們結合了 TPC-NAS 對現有架構進行神經網路架構搜尋，在數分鐘內的 CPU 執行時間下，我們可以找到相近運算量、TPC score 更高的架構。並且經過訓練後無論是 fp32 還是二值化後的準確率都能比原先的模型高。這部分同樣在影像辨識、物件偵測、自然語言處理之任務上，使用卷積神經網路、transformer 之架構進行驗證。

最後我們探討在實際硬體實現上二值化神經網路之能效，相比於 8W8A 的精度，1W4A 的運算能夠節省 3.57 倍的面積與 3.07 倍的功耗。在計算乘加計算時則有近似的空間，我們提出基於基本元件庫的 OAI 近似方法，經過邏輯閘級模擬能夠達到比 Maj3 與 NAND NOR 近似更低的面積與功耗。在近似感知訓練的結果也能證明在實際網路的推論運算中，使用 OAI 近似能夠達到 fp32 之準確率。

doi:10.6342/NTU202400902

未來，我們會將二值化應用於更多任務上。目前的二值化模型主要還是屬於分類與回歸任務，而現今最熱門的神經網路應用主要是生成式模型，如何在二值化後維持生成式模型之表現是相當重要的。近年來大型語言模型的蓬勃發展也令我們必須尋找在量化感知訓練以外，要如何在有效的計算資源與時間內對大型生成式模型進行量化，並維持可接受的準確率。

在演算法方面，目前的 activation 位元數皆為人工決定。與其他量化的研究相似，在這方面也可以進行混合精度搜尋(Mixed-precision search)，如何在維持模型表現的同時最小化 activation 整體的位元數也是可行的研究方向。

在硬體方面，從效能分析可以看出隨著精度降低，記憶體所佔之面積與功耗也會隨之提升，此時記憶體內運算(In-memory computing)就顯得格外重要。同時這部分也能結合近似電路，在記憶體內運算降低資料傳輸之成本後，近似電路能夠讓我們更有效率地進行計算。在這部分乘加近似感知訓練能夠維持近似後之準確率，但目前的訓練方式還需要進行修改才能用於 multi-bit activation 量化中。

doi:10.6342/NTU202400902

參考文獻

[1] Z. Liu, B. Wu, W. Luo, X. Yang, W. Liu, and K.-T. Cheng, "Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm," in *Proc. European Conference on Computer Vision (ECCV)*, Sep. 2018, pp. 722-737.

[2] N. Guo, J. Bethge, C. Meinel, and H. Yang, "Join the high accuracy club on imagenet with a binary neural network ticket," *arXiv preprint arXiv:2211.12933,* 2022.

[3] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, USA, Jun. 2018, pp. 2704-2713.

[4] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432,* 2013.

[5] Z. Xu, M. Lin, J. Liu, J. Chen, L. Shao, Y. Gao, Y. Tian, and R. Ji, "Recu: Reviving the dead weights in binary neural networks," in *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2021, pp. 5198-5208.

[6] Y. Bhalgat, J. Lee, M. Nagel, T. Blankevoort, and N. Kwak, "Lsq+: Improving low-bit quantization through learnable offsets and better initialization," in *Proc.*

doi:10.6342/NTU202400902

IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Jun. 2020, pp. 696-697.

[7] S. K. Esser, J. L. McKinstry, D. Bablani, R. Appuswamy, and D. S. Modha, "Learned step size quantization," *arXiv preprint arXiv:1902.08153,* 2019.

[8] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980,* 2014.

[9] Z. Liu, Z. Shen, S. Li, K. Helwegen, D. Huang, and K.-T. Cheng, "How do adam and training strategies help bnns optimization," in *Proc. International Conference on Machine Learning*, Jul. 2021: PMLR, pp. 6936-6946.

[10] M. Sun, H. Ma, G. Kang, Y. Jiang, T. Chen, X. Ma, Z. Wang, and Y. Wang, "Vaqf: Fully automatic software-hardware co-design framework for low-bit vision transformer," *arXiv preprint arXiv:2201.06618,* 2022.

[11] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531,* 2015.

[12] B. Heo, J. Kim, S. Yun, H. Park, N. Kwak, and J. Y. Choi, "A comprehensive overhaul of feature distillation," in *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, Seoul, Korea, Oct. 2019, pp. 1921-1930.

[13] B. Martinez, J. Yang, A. Bulat, and G. Tzimiropoulos, "Training binary neural networks with real-to-binary convolutions," *arXiv preprint arXiv:2003.11535,* 2020.

[14] Z. Liu, B. Oguz, A. Pappu, L. Xiao, S. Yih, M. Li, R. Krishnamoorthi, and Y. Mehdad, "Bit: Robustly binarized multi-distilled transformer," *Proc. Advances in*

doi:10.6342/NTU202400902

Neural Information Processing Systems (NeurIPS), vol. 35, pp. 14303-14316, 2022.

[15] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "Glue: A multi-task benchmark and analysis platform for natural language understanding," *arXiv preprint arXiv:1804.07461,* 2018.

[16] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805,* 2018.

[17] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. National Academy of Sciences,* vol. 79, no. 8, pp. 2554-2558, Apr. 1982.

[18] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proc. National Academy of Sciences,* vol. 81, no. 10, pp. 3088-3092, May. 1984.

[19] J. J. Hopfield and D. W. Tank, "Computing with neural circuits: A model," *Science,* vol. 233, no. 4764, pp. 625-633, Aug. 1986.

[20] J. J. Hopfield and D. W. Tank, "“Neural” computation of decisions in optimization problems," *Biological cybernetics,* vol. 52, no. 3, pp. 141-152, Jul. 1985.

[21] S. Bharitkar and J. M. Mendel, "The hysteretic hopfield neural network," *IEEE Transactions on Neural Networks,* vol. 11, no. 4, pp. 879-888, Jul. 2000.

doi:10.6342/NTU202400902

[22] Y. Takefuji and K. Lee, "An artificial hysteresis binary neuron: A model suppressing the oscillatory behaviors of neural dynamics," *Biological Cybernetics,* vol. 64, no. 5, pp. 353-356, Mar. 1991.

[23] G. Xia, Z. Tang, Y. Li, and R. Wang, "Hopfield neural network with hysteresis for maximum cut problem," *Neural Information Processing-Letters and Reviews,* vol. 4, no. 2, pp. 19-26, Aug. 2004.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 770-778.

[25] A. Krizhevsky and G. Hinton, *Learning multiple layers of features from tiny images*. Technical report, CIFAR, 2009.

[26] D. Kim, J. Lee, and B. Ham, "Distance-aware quantization," in *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2021, pp. 5271-5280.

[27] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, and M. Bernstein, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision,* vol. 115, pp. 211-252, Apr. 2015.

[28] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision,* vol. 88, pp. 303-338, Sep. 2010.

doi:10.6342/NTU202400902

[29] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Proc. European Conference on Computer Vision (ECCV)*, Oct. 2016: Springer, pp. 21-37.

[30] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Proc. Advances in Neural Information Processing Systems (NeurIPS),* vol. 28, 2015.

[31] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556,* 2014.

[32] 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.

[33] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *Proc. International Conference on Machine Learning*, Jun. 2019: PMLR, pp. 6105-6114.

[34] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," in *Proc. International Conference on Machine Learning*, Jul. 2021: PMLR, pp. 10347-10357.

[35] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, and S. Gelly, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929,* 2020.

doi:10.6342/NTU202400902

[36] M. Ding, B. Xiao, N. Codella, P. Luo, J. Wang, and L. Yuan, "Davit: Dual attention vision transformers," in *Proc. European Conference on Computer Vision (ECCV)*, Nov. 2022: Springer, pp. 74-92.

[37] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2021, pp. 10012-10022.

[38] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1," *arXiv preprint arXiv:1602.02830,* 2016.

[39] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *Proc. European Conference on Computer Vision (ECCV)*, Sep. 2016: Springer, pp. 525-542.

[40] A. Bulat and G. Tzimiropoulos, "Xnor-net++: Improved binary neural networks," *arXiv preprint arXiv:1909.13863,* 2019.

[41] Z. Liu, Z. Shen, M. Savvides, and K.-T. Cheng, "Reactnet: Towards precise binary neural network with generalized activation functions," in *Proc. European Conference on Computer Vision (ECCV)*, Aug. 2020: Springer, pp. 143-159.

[42] Z. Liu, Z. Shen, M. Savvides, and K.-T. Cheng, "Reactnet: Towards precise binary neural network with generalized activation functions," *arXiv pre-print server,* 2020-07-13 2020, doi: None
arxiv:2003.03488.

doi:10.6342/NTU202400902

[43] M. Huang, "Efficient neural architecture and mixed-precision search for quantized neural networks using model path counts," M.S. thesis, National Taiwan University, Taipei, 2023.

[44] J. Bergstra, D. Yamins, and D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *Proc. International Conference on Machine Learning*, Atlanta, USA, Jun. 2013: PMLR, pp. 115-123.

[45] B. Zoph and Q. Le, "Neural architecture search with reinforcement learning," in *Proc. International Conference on Learning Representations (ICLR)*, 2016.

[46] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," *arXiv preprint arXiv:1806.09055,* 2018.

[47] D. Wang, M. Li, C. Gong, and V. Chandra, "Attentivenas: Improving neural architecture search via attentive sampling," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2021, pp. 6418-6427.

[48] M. Lin, P. Wang, Z. Sun, H. Chen, X. Sun, Q. Qian, H. Li, and R. Jin, "Zen-nas: A zero-shot nas for high-performance image recognition," in *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2021, pp. 347-356.

[49] B. Wu, Y. Wang, P. Zhang, Y. Tian, P. Vajda, and K. Keutzer, "Mixed precision quantization of convnets via differentiable neural architecture search," *arXiv preprint arXiv:1812.00090,* 2018.

doi:10.6342/NTU202400902

[50] P. K. A. Vasu, J. Gabriel, J. Zhu, O. Tuzel, and A. Ranjan, "Fastvit: A fast hybrid vision transformer using structural reparameterization," *arXiv preprint arXiv:2303.14189,* 2023.

[51] A. Mishra, E. Nurvitadhi, J. J. Cook, and D. Marr, "Wrpn: Wide reduced-precision networks," *arXiv preprint arXiv:1709.01134,* 2017.

[52] I. Turc, M.-W. Chang, K. Lee, and K. Toutanova, "Well-read students learn better: On the importance of pre-training compact models," *arXiv preprint arXiv:1908.08962,* 2019.

[53] D. Wang, C.-T. Lin, G. K. Chen, P. Knag, R. K. Krishnamurthy, and M. Seok, "Dimc: 2219tops/w 2569f2/b digital in-memory computing macro in 28nm based on approximate arithmetic hardware," in *Proc. IEEE International Solid-State Circuits Conference (ISSCC)*, San Francisco, USA, Feb. 2022, vol. 65: IEEE, pp. 266-268.

[54] S. Rasoulinezhad, S. Fox, H. Zhou, L. Wang, D. Boland, and P. H. Leong, "Majoritynets: Bnns utilising approximate popcount for improved efficiency," in *Proc. International Conference on Field-Programmable Technology (ICFPT)*, Tianjin, China, Dec. 2019: IEEE, pp. 339-342.

doi:10.6342/NTU202400902