深度學習
Deep Learning

模型壓縮

Instructor: 林英嘉 (Ying-Jia Lin)
2025/05/05

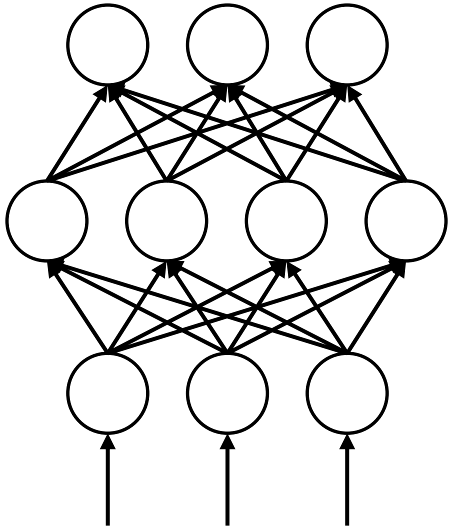
Course GitHub

Slido # DL_0505

Outline

- Model compression techniques
  - Knowledge Distillation
  - Pruning
  - Quantization

Why do we need model compression?



參數這麼多

- ViT: 86M
    - 86 x 4 bytes = 344M = 344000000 bytes = 344MB
- GPT-3: 175B = 175000M
    - 175000 x 4 bytes = 700000M = 700000000000 bytes = 700GB

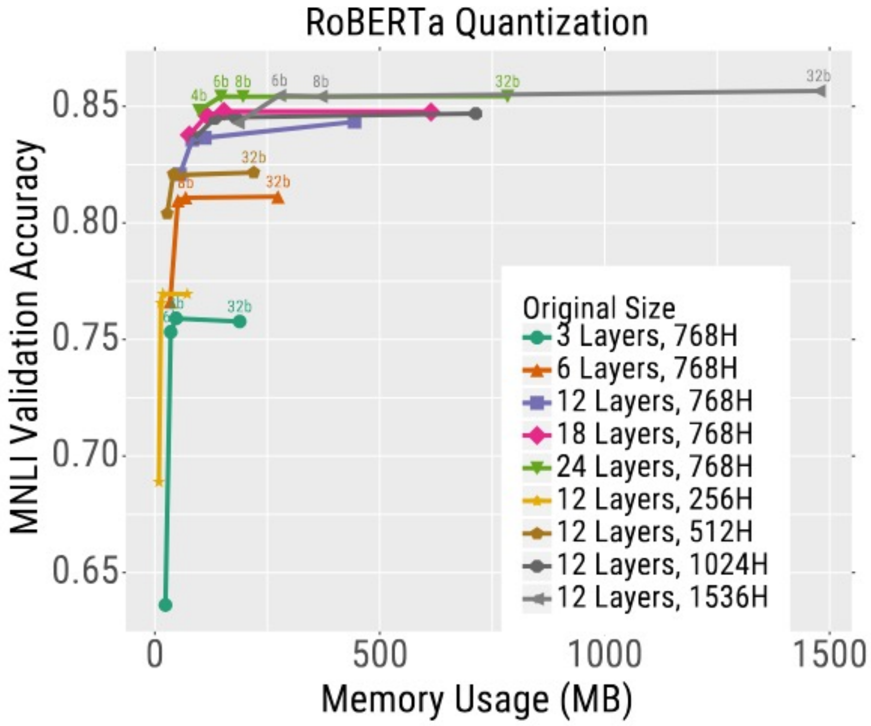
為什麼我們不直接訓練一個小模型？

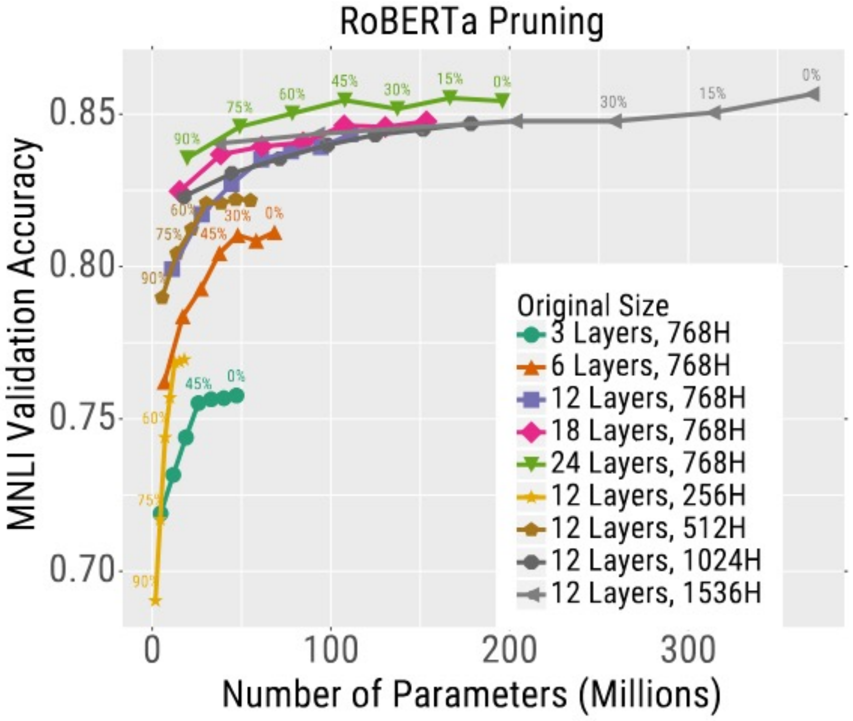
- 目前邏輯：先訓練大模型，再進行模型壓縮，但為什麼？
  - 小模型單獨無法達到大模型的表現，但是小模型可以透過學習大模型來達到大模型的表現 (Ba and Caruana, *NeurIPS* 2014)
  - 相較於大模型，小模型在訓練階段的收斂速度慢且不穩定 (Martinez et al., *EMNLP Findings* 2024)

[Ba and Caruana, NeurIPS 2014] Do Deep Nets Really Need to be Deep?
[Martinez et al., EMNLP Findings 2024] Tending Towards Stability: Convergence Challenges in Small Language Models
https://datascience.stackexchange.com/questions/86395/do-smaller-neural-nets-always-converge-faster-than-larger-ones

為什麼我們不直接訓練一個小模型？

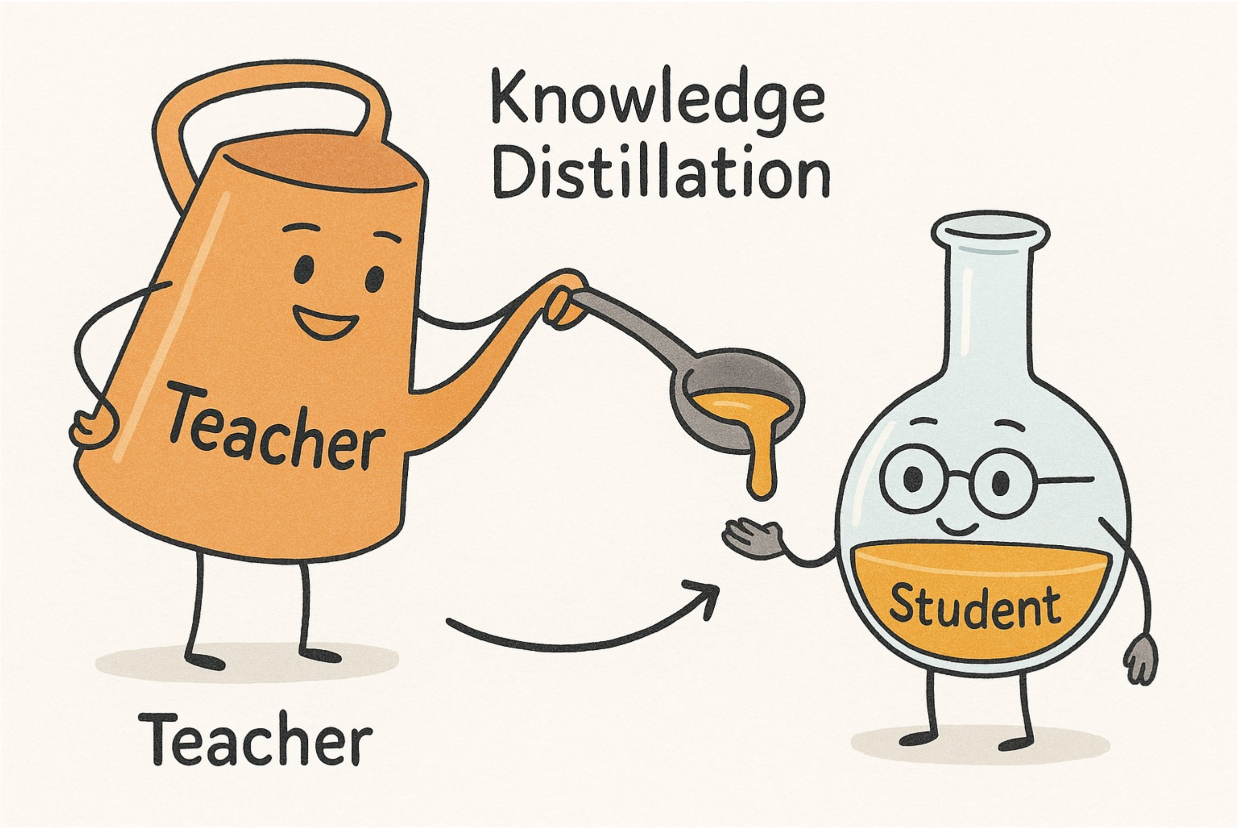
- 目前邏輯：先訓練大模型，再進行模型壓縮，但為什麼？
  - 實驗表明這樣的流程比較好





Li, Zhuohan, et al. "Train big, then compress: Rethinking model size for efficient training and inference of transformers." ICML, 2020.

Knowledge Distillation



[Recap] 模型輸出的後處理

x_i: 資料集中第 i 張影像
y_j: label，在此範例中 $j = 1,2,3$

Cross-entropy: $\mathcal{L}_i = -\sum_j y_j \log \hat{P}(y_j|x_i)$

x_i

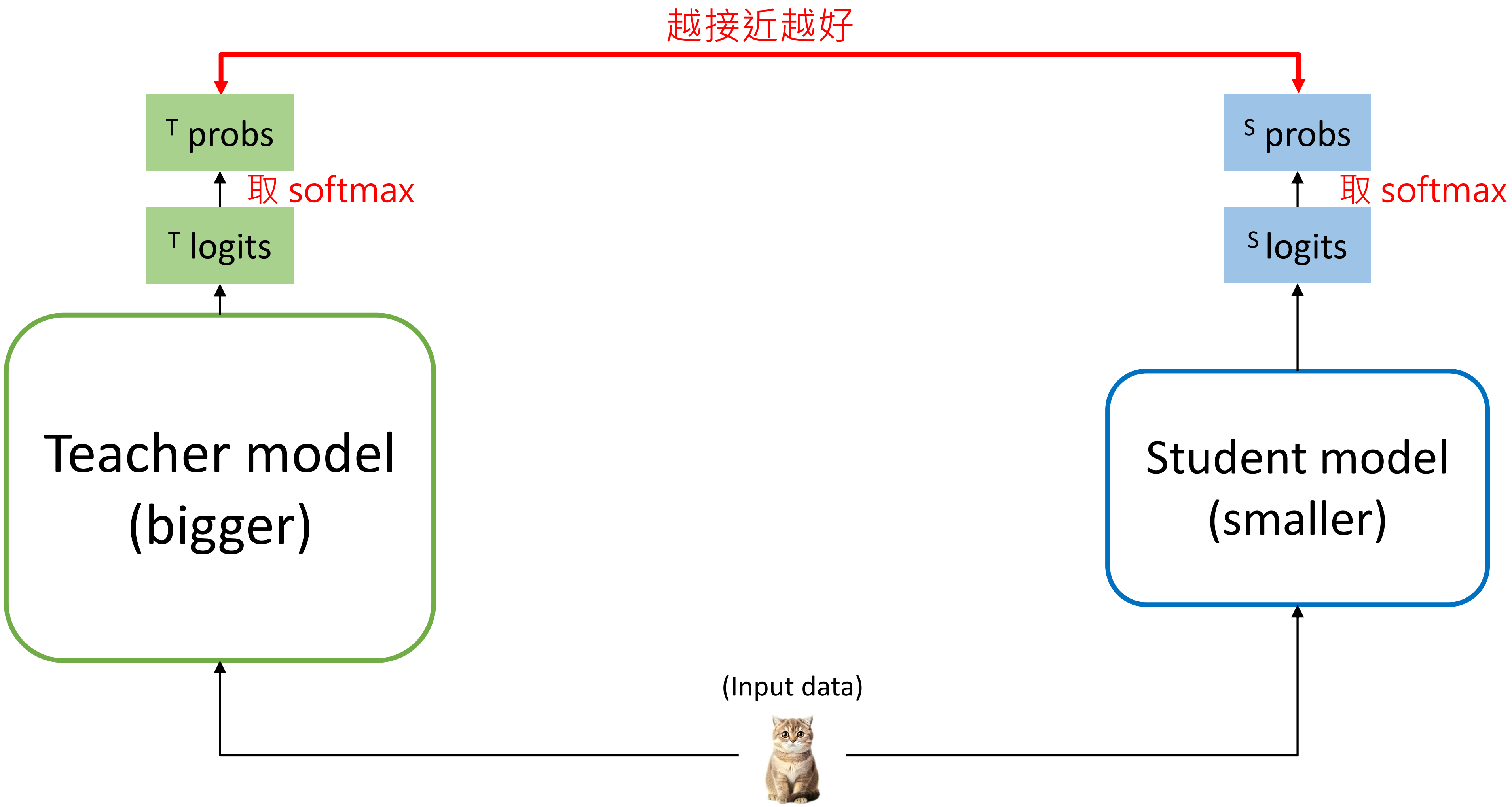
→ Model →

	Model output (logits)	Softmax $\hat{P}(y_j \mid x_i)$	label (y_j)	Cross-entropy (loss)
y_1 Cat	0.5	0.225	1	$-\log 0.225$
y_2 Dog	0.7	0.275	0	0
y_3 Apple	1.3	0.500	0	0

各類別加總：$\mathcal{L}_i = -\log 0.225$

Teacher model and student model

Hinton, Geoffrey, Oriol Vinyals, and Jeff Dean. "Distilling the knowledge in a neural network." *arXiv preprint arXiv:1503.02531* (2015).



模型輸出的後處理 for KD

x_i: 資料集中第 i 張影像
y_j: label，在此範例中 $j = 1,2,3$

KL-divergence: $\mathcal{L}_{\mathrm{KL}}(x_i) = \mathrm{KL}(\hat{P}_T || \hat{P}_S) = \sum_j \hat{P}_T(y_j|x_i) \log \frac{\hat{P}_T(y_j|x_i)}{\hat{P}_S(y_j|x_i)}$

x_i

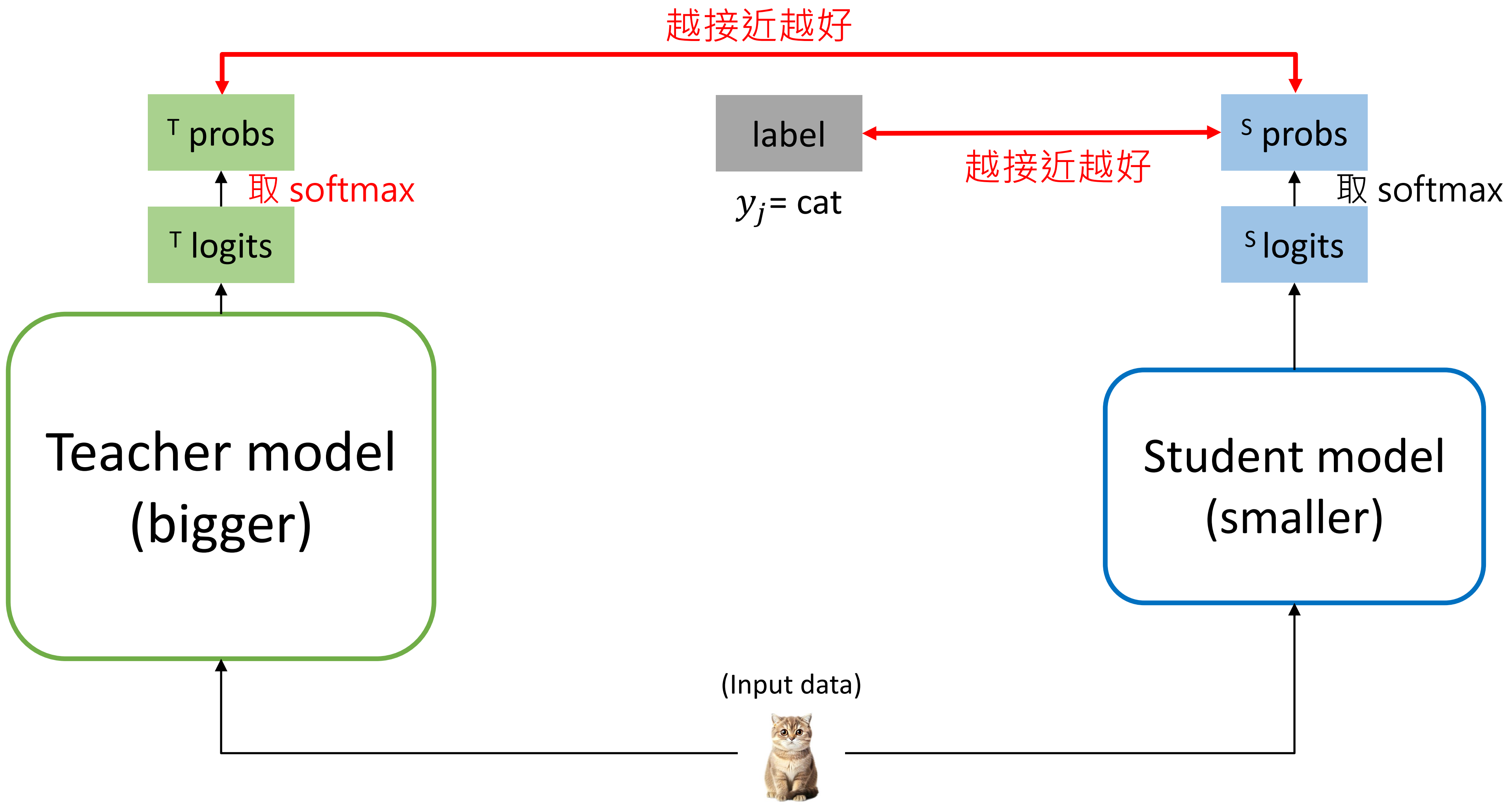
→ Model →

	Teacher prob $\hat{P}_T(y_j\|x_i)$	Student prob $\hat{P}_S(y_j\|x_i)$	KL-divergence (loss)
y_1 Cat	0.500	0.225	$0.5 \log \frac{0.5}{0.225}$
y_2 Dog	0.275	0.275	$0.275 \log \frac{0.275}{0.275}$
y_3 Apple	0.225	0.500	$0.225 \log \frac{0.225}{0.5}$

各類別加總：$\mathcal{L}_{\mathrm{KL}}(x_i) = 0.5 \log \frac{0.5}{0.225} + 0.275 \log \frac{0.275}{0.275} + 0.225 \log \frac{0.225}{0.5}$

Optimizing a student model

Hinton, Geoffrey, Oriol Vinyals, and Jeff Dean. "Distilling the knowledge in a neural network." *arXiv preprint arXiv:1503.02531* (2015).



KD 目標函數

Hard Targets

label

S probs

$$\mathcal{L}_i = -\sum_j \alpha \cdot y_j \log \hat{P}\left(y_j \middle| x_i\right) + (1-\alpha) \cdot \mathrm{KL}\left(\hat{P}_T || \hat{P}_S\right)$$

T probs

S probs

Soft Targets

α 是一個超參數 (調整兩種 losses 的比例)

Problems of Knowledge Distillation

- We always need a pre-initialized student model.
- Training with unsupervised data is time-consuming.

Pruning

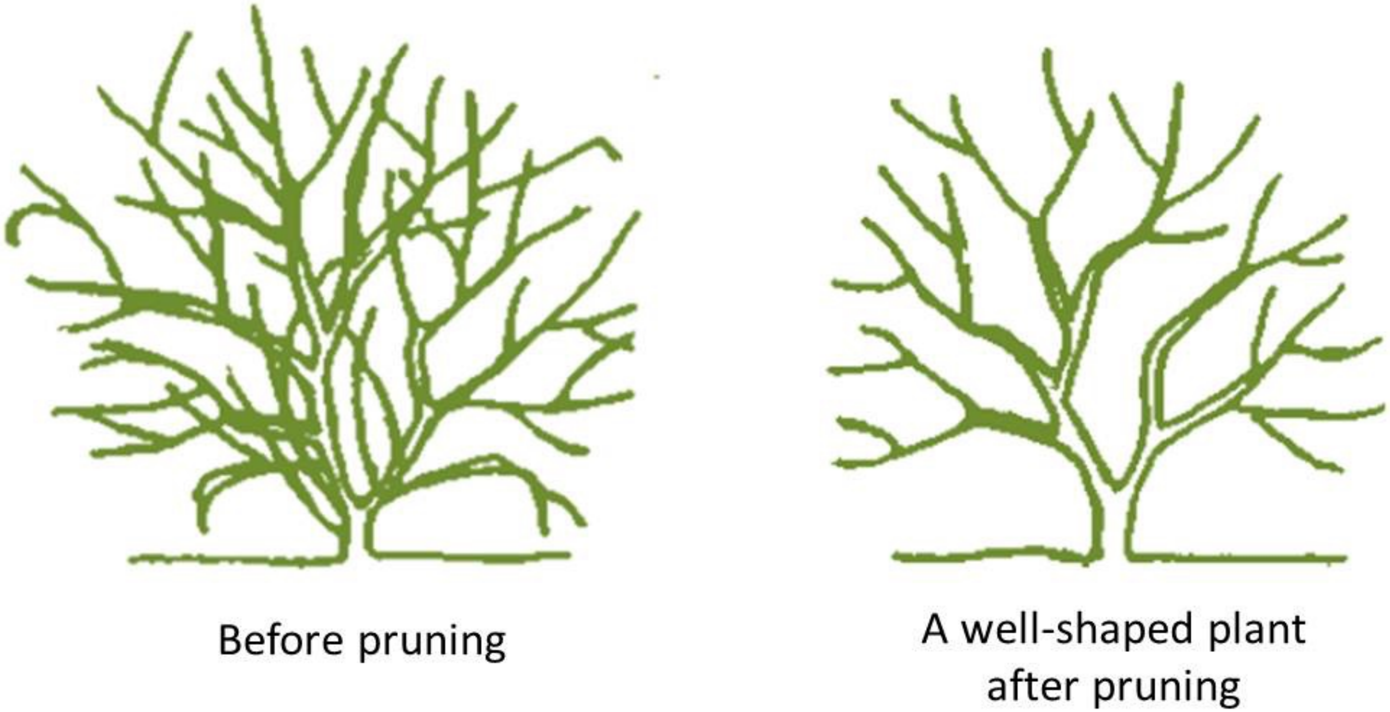
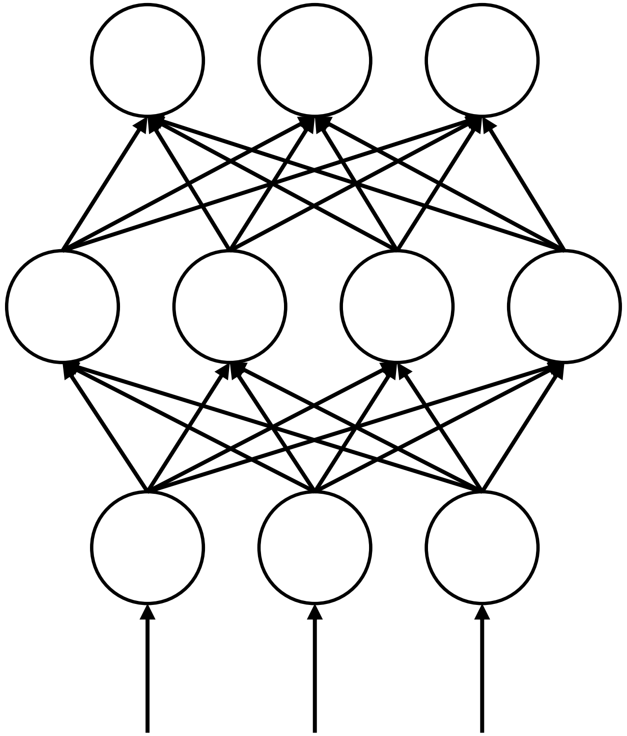


Figure source:
https://www.uky.edu/Ag/Horticulture/QRLabels/Pruning.html

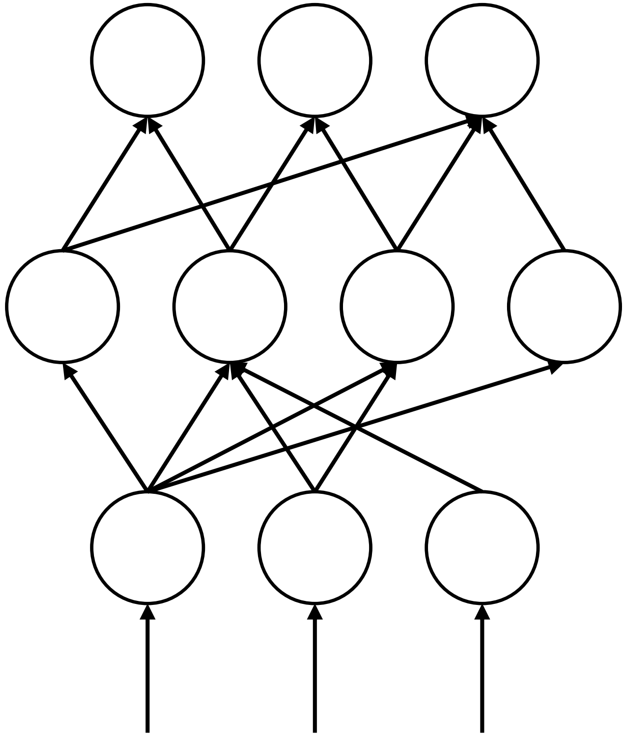
Overview of Pruning

Han, Song, et al. "Learning both weights and connections for efficient neural network." NeurIPS 2015.

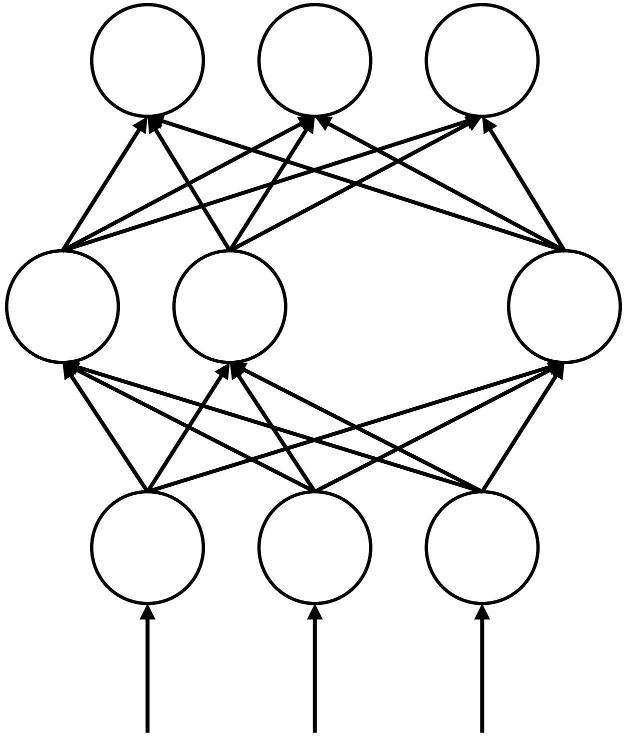
Before Pruning



方式1: Weight Pruning

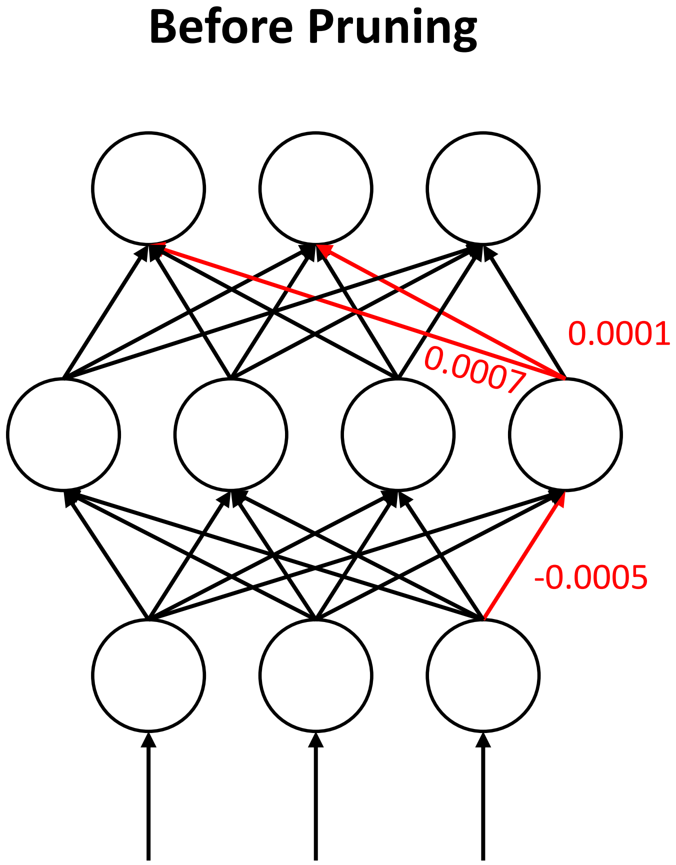


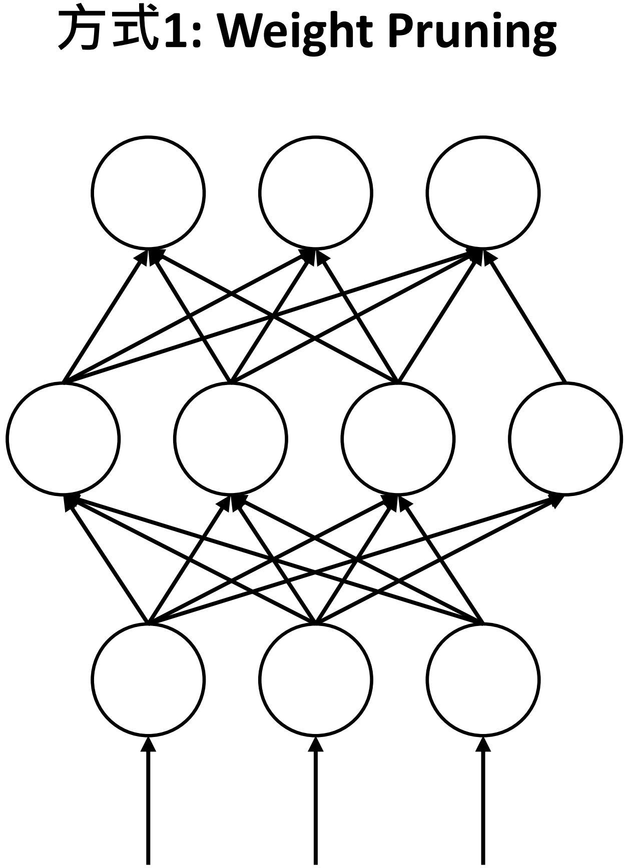
方式2: Node Pruning



Unstructured Pruning (方法)

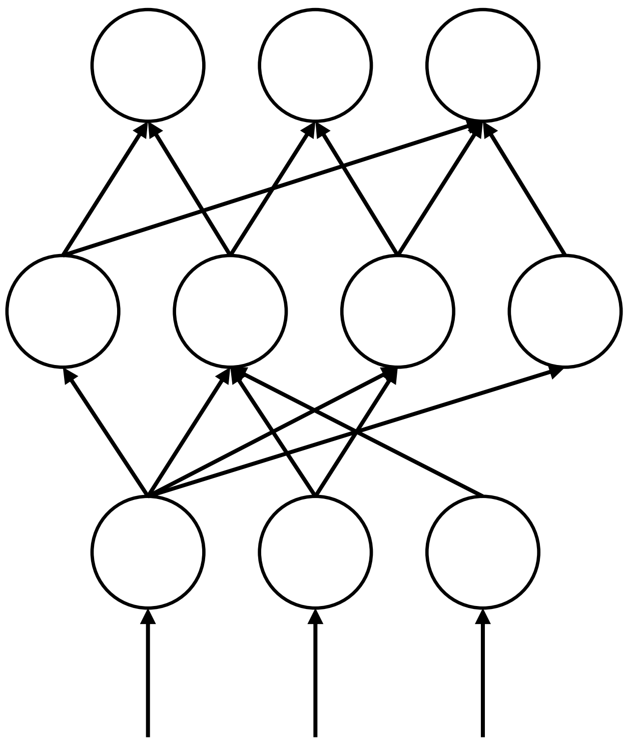
移除 |weight 數值| < threshold 的 weights
例如：threshold = 0.001





Unstructured Pruning (優缺點)

方式1: Weight Pruning

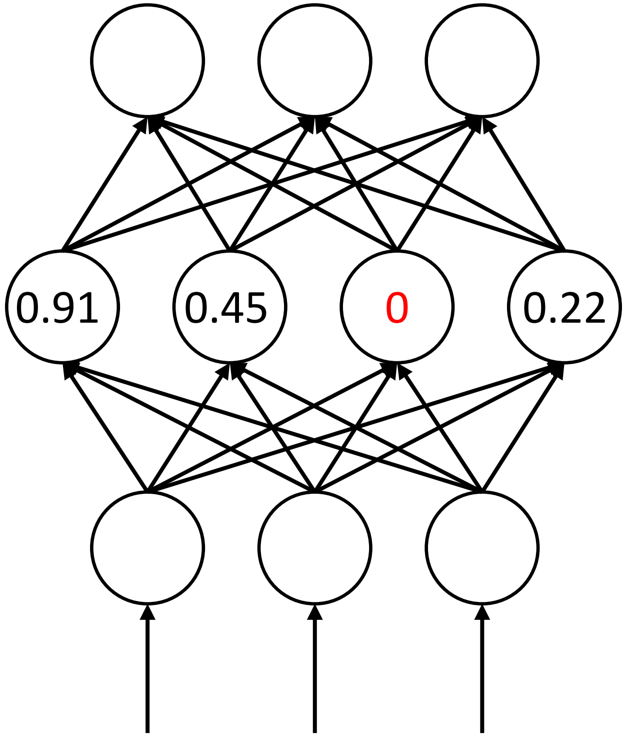


- **優點：**
  - 剪的是單個權重，可以任意刪除來達到很高的壓縮率
  - 剪掉的是影響小的權重，因此通常和原本模型相比的效能損失可能較小
- **缺點：**
  - 難以構成統一的平行化矩陣，例如：
    - 有的 Nodes 有 2 個 outputs；有的 Node 只有 1 個 input
    - 有的 Nodes 有 2 個 inputs；有的 Nodes 有 1 個 input
  - 雖然參數變少，但可能難以平行化，故速度可能慢

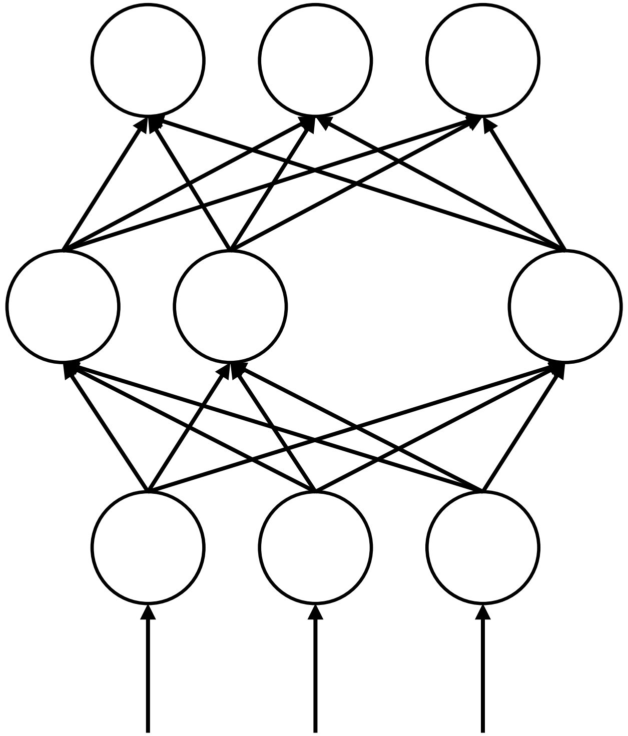
Structured Pruning (方法)

移除 | activations 數值| < threshold 的 neurons
例如：threshold = 0.001

Before Pruning

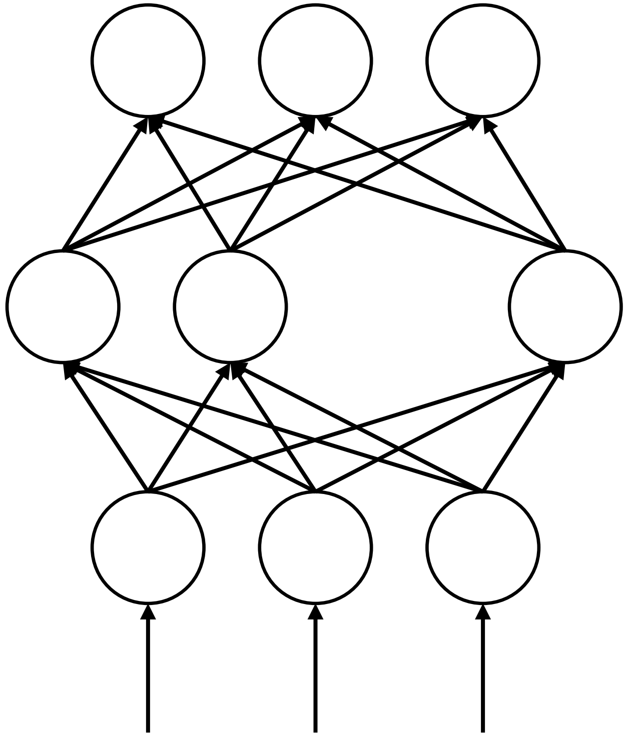


方式2: Node Pruning



Structured Pruning (優缺點)

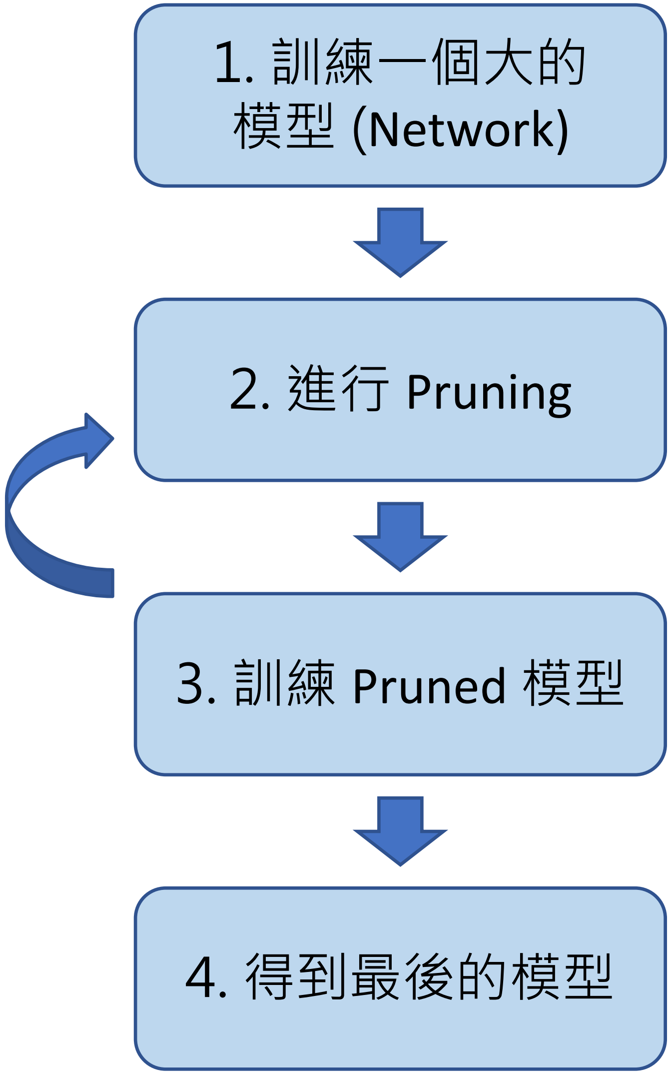
方式2: Node Pruning



- **優點：**
    - 等同於 hidden size 變小，因此可以透過GPU進行平行化
- **缺點：**
    - 一次刪除部分 Node(s) 可能使模型效能損失較大
    - 壓縮率和 Unstructured Pruning 比起來較不彈性

Pruning 訓練流程

Han, Song, et al. "Learning both weights and connections for efficient neural network." NeurIPS 2015.



- 一般來說，我們不會一次把模型剪掉太多參數
  - 所以 Step 2. 和 Step 3. 會重複進行
  - 直到參數量足夠小 (可自行決定)

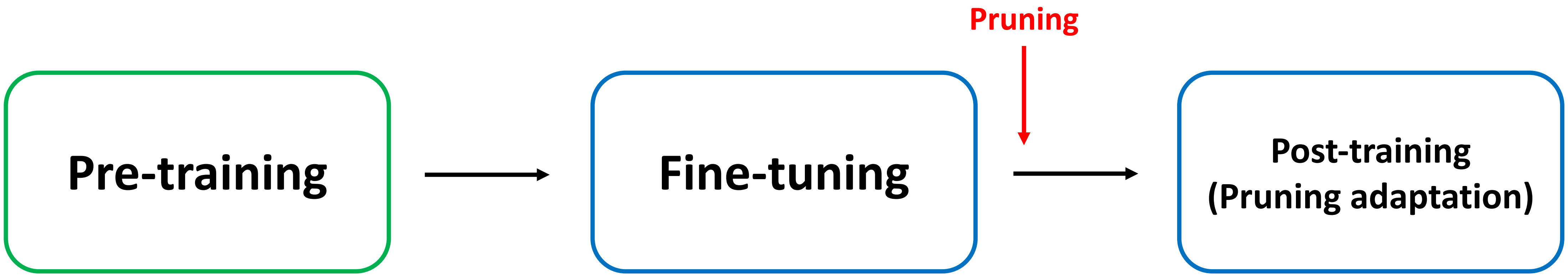
[Recap] 先 pre-training，再 Fine-tuning

Pre-training → **Fine-tuning**

在大量資料上進行訓練，通常是自監督式 (Self-Supervised Training)

在目標資料上 (Down-stream tasks, 下游任務) 進行訓練，通常是監督式 (Supervised Training)，也就是需要有標註的資料才能進行模型訓練

Pre-training, Fine-tuning, and Post-training



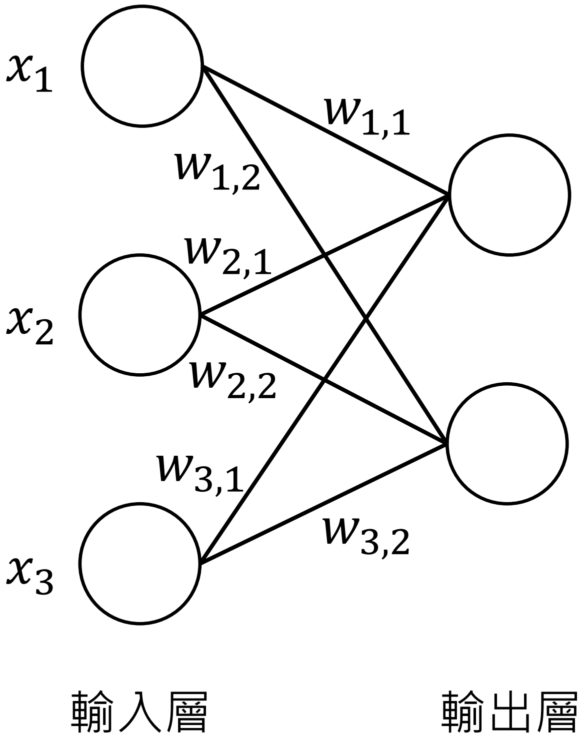
在大量資料上進行訓練，通常是自監督式 (Self-Supervised Training)

在目標資料上 (Down-stream tasks, 下游任務) 進行訓練，通常是監督式 (Supervised Training)，也就是需要有標註的資料才能進行模型訓練

適應模型剪枝

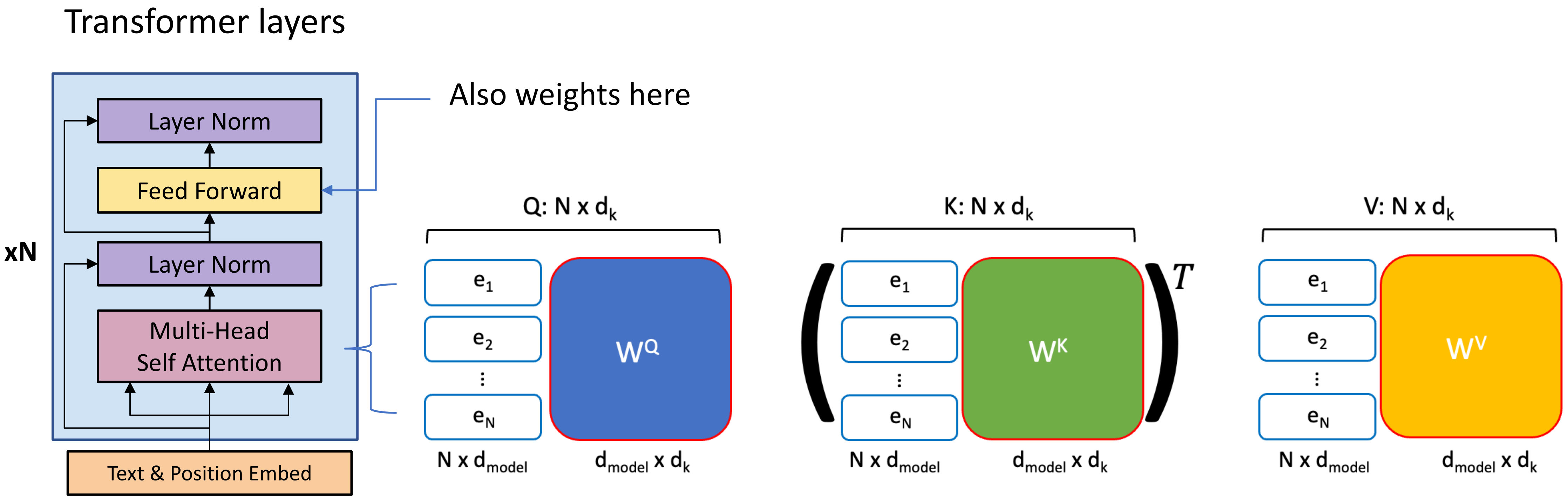
Quantization
量化

[Recap] MLP is composed of weight matrices



$$\underset{\mathbb{R}^{1\times 3}}{\begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}} \times \underset{\mathbb{R}^{3\times 2}}{\boxed{\begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \\ w_{3,1} & w_{3,2} \end{bmatrix}}} + \underset{\mathbb{R}^{2}}{\text{bias}}$$

Weights in Transformer layers



$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

看看 PyTorch 怎麼存 tensors

如果是浮點數的話， PyTorch 預設以 float32 (FP32) 儲存數值

```
>>> a = torch.tensor([[1., -1.], [1., -1.]])
>>> a.dtype
torch.float32
```

```
>>> a = torch.tensor([[1, -1], [1, -1]])
>>> a.dtype
torch.int64
```

數值範圍比較

數值類型名稱	PyTorch dtype	數值範圍	Bit數	Byte數
float32	torch.float32	約±3.4e38	32	4
float16	torch.float16	約±6.5e4	16	2
int8	torch.int8	-128 ~ 127	8	1
uint8	torch.uint8	0 ~ 255	8	1
int4	x	-8 ~ 7	4	0.5
uint4	x	0 ~ 15	4	0.5

u: unsigned (無號，代表數值只有正的)

Introduction to Quantization

(這頁只是示意圖，數值不精準)

$$\begin{bmatrix} -0.4 & 1.3 & 3.73 \\ -4.7 & -3.2 & -6.4 \\ 8.5 & 14.3 & 13.5 \end{bmatrix} \xrightarrow[\text{FP32} \;\to\; \text{int8}]{\text{降低儲存精度}} \begin{bmatrix} 0 & 1 & 4 \\ -5 & -3 & -6 \\ 9 & 14 & 14 \end{bmatrix}$$

36 bytes

32-bit floating point (FP32): 1個值需要**4個bytes**才能儲存

8 bytes

8-bit Integer (int8): 1個值需要**1個bytes**才能儲存

誤差：
$$\begin{bmatrix} 0.4 & -0.3 & 0.27 \\ -0.3 & 0.2 & 0.4 \\ 0.5 & -0.3 & 0.5 \end{bmatrix}$$

計算機概論 / C 語言

- int8: 整數 (integer) 使用 8 個位元 (bits) 來儲存數值，可以分成 Unsigned int8 和 signed int8
- Unsigned int8 (無號整數):

bit 位數	7	6	5	4	3	2	1	0	
二進位 (0或1)	**1**	**0**	**0**	**1**	**1**	**1**	**1**	**1**	
十進位	2^7			2^4	2^3	2^2	2^1	2^0	=128+16+8+4+2+1=159

bit 位數	7	6	5	4	3	2	1	0	
二進位 (0或1)	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	
十進位	2^7								=128

- 因此，Unsigned int8 的範圍為 0 到 255

計算機概論 / C 語言

- int8: 整數 (integer) 使用 8 個位元 (bits) 來儲存數值，可以分成 Unsigned int8 和 signed int8
- Signed int8 (有號整數):

bit 位數	7	6	5	4	3	2	1	0	
二進位 (0或1)	**1**	**0**	**0**	**1**	**1**	**1**	**1**	**1**	
十進位	-2^7			2^4	2^3	2^2	2^1	2^0	=-128+16+8+4+2+1=-97

bit 位數	7	6	5	4	3	2	1	0	
二進位 (0或1)	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	
十進位		2^6	2^5	2^4	2^3	2^2	2^1	2^0	=64+32+16+8+4+2+1=127

- 因此，signed int8 的範圍為 -128 到 +127

算出負數的二進位 (電腦背後作法)

- 假設現在要算-5的二進位
- 先算+5的二進位：

十進位						2^2		2^0	=4+1=5
二進位	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**1**	

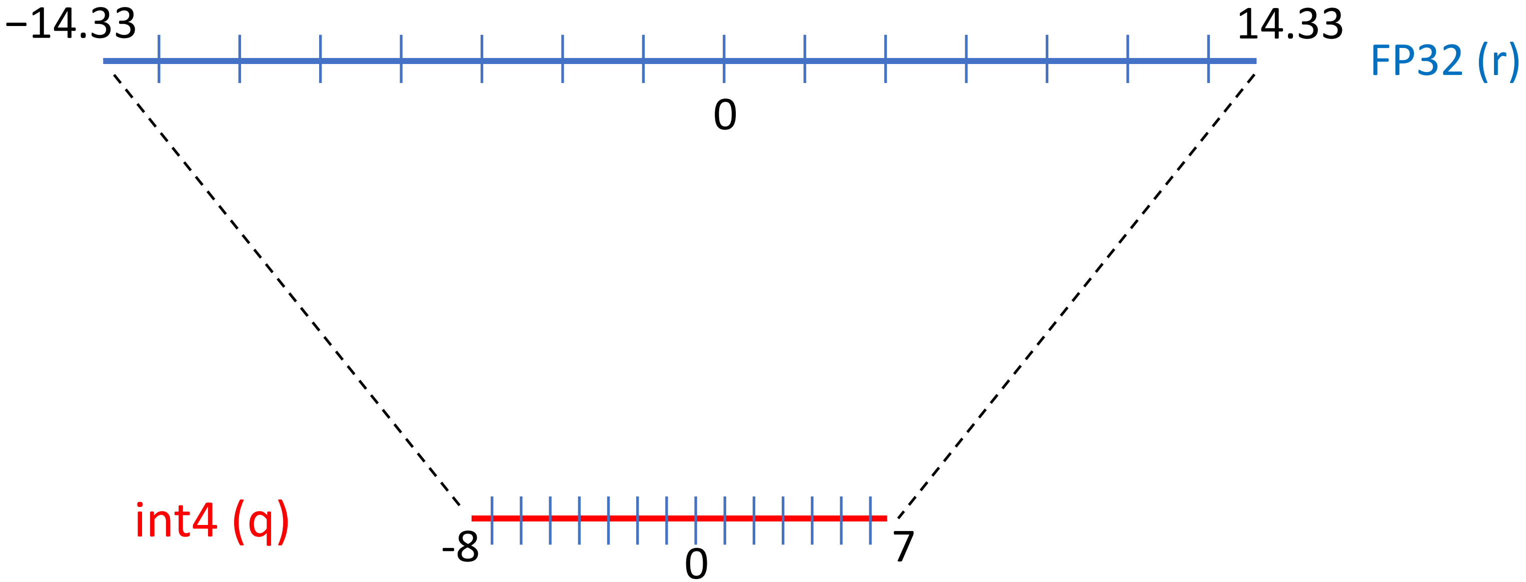
- 取反 (1換成0，0換成1)：

二進位	**1**	**1**	**1**	**1**	**1**	**0**	**1**	**1**	
	-2^7	2^6	2^5	2^4	2^3		2^1	+1	=-128+64+32+16+8+2+1=-5

From FP32 to int4

當前資料以 FP32 儲存的值取絕對值最大者的正負號作為兩端，為r的數值範圍

每格的級距s為：(14.33+14.33) / (7+8) = 1.9107



切成15格

14.33 / 7 = (14.33 - 1.9107) / x

x = (14.33 - 1.9107) * 7 / 14.33
= r * 1/s

r / s = 14.33 /1.9107 = 7.499869
量化後數值 q = round(r / s) = 7

Quantization 方式

🧊 Post-training Quantization (PTQ)

在模型訓練好之後，直接將權重轉成低精度，不重新訓練

🔥 Quantization-aware Training (QAT)

在訓練過程中就模擬量化的影響，讓模型學會適應低精度

PTQ 與 QAT 比較

	🧊 **PTQ**	🔥 **QAT**
何時做量化？	模型訓練完之後	訓練過程中就模擬量化
需不需再訓練？	❌ 有 pre-trained model 就可	✅
最後模型表現	較差 (轉換後精度越低，表現越差)	較好
實作難易度	較簡單	較難

Gemma 3 QAT

GEMMA / AI EDGE

Gemma 3 QAT Models: Bringing state-of-the-Art AI to consumer GPUs

APRIL 18, 2025

Edouard YVINEC
Research Scientist

Phil Culliton
ML Engineer

Share

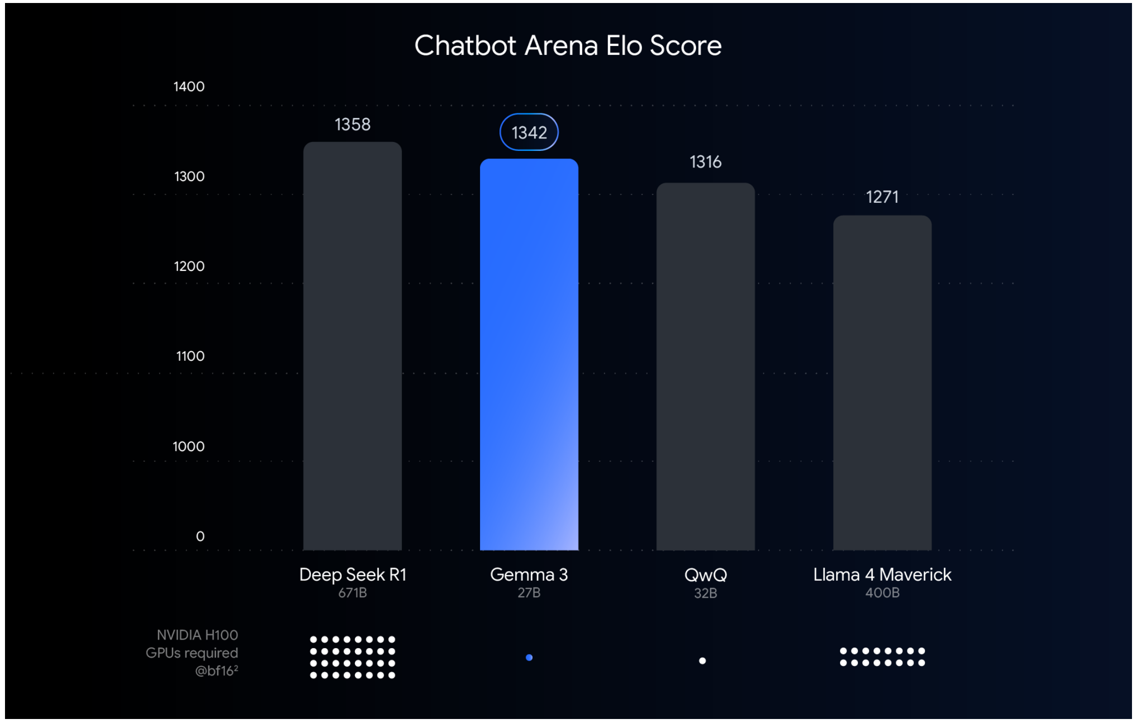


Figure source: https://developers.googleblog.com/en/gemma-3-quantized-aware-trained-state-of-the-art-ai-to-consumer-gpus/

Summary

Comparison: What is Model Compression?

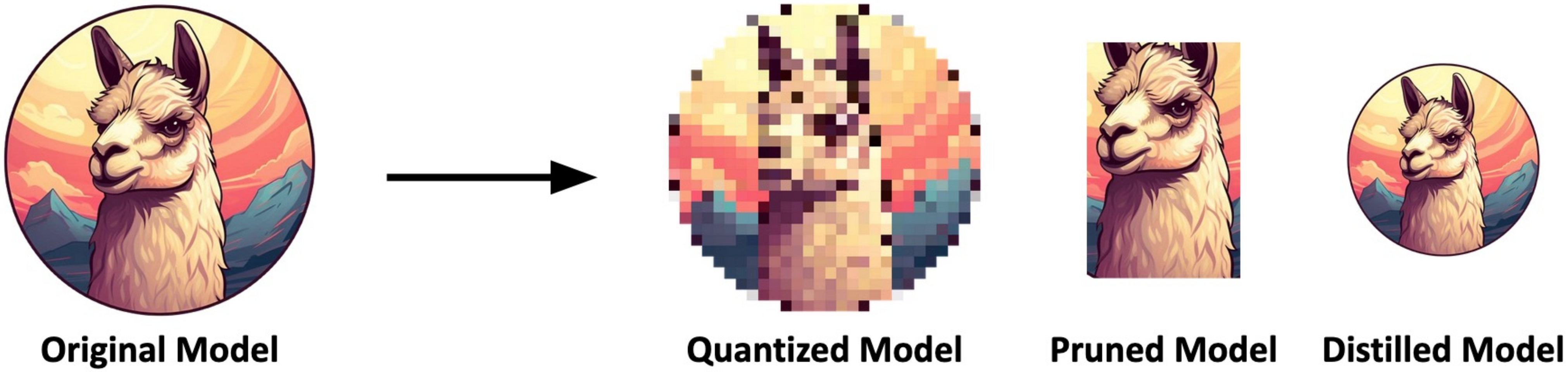


Figure credit: yxhong-tw

Apple Intelligence 最低 8GB 的 RAM?

對於 1B 的模型來說
Float32 -> 4GB
Float16 -> 2GB
8-bit -> 1GB
4-bit -> 0.5GB

模型	Gemma 3 4B	Gemma3 QAT	Llama 3.2 3B
參數量	4B	27B	3B
Memory	4 GB (8-bit)	13.5 GB (4-bit)	3 GB (8-bit)

Additional resources

- MIT EfficientML by Prof. Song Han
  - Pruning and Sparsity (Part I)
  - Pruning and Sparsity (Part II)
  - Quantization (Part I)
  - Quantization (Part II)
- bitsandbytes
  - https://huggingface.co/docs/transformers/en/quantization/bitsandbytes
- DeepLearning.AI course
  - https://www.deeplearning.ai/short-courses/quantization-fundamentals-with-hugging-face

Thank you!

Instructor: 林英嘉

✉️ yjlin@cgu.edu.tw

TA: 林君襄

✉️ becky890926@gmail.com