聊聊 SpinQuant

CS Chen

正交(旋轉)矩陣

- 用方陣討論就好, 比較方便
- 正交/旋轉 矩陣: $RR^T = R^T R = I$
- $y = Rx$, 保長

$$\|y\|^2 = \|Rx\|^2 = (Rx)^{\mathrm{T}}(Rx) = x^T R^T Rx = x^T x = \|x\|^2$$

- $x' = Rx, y' = Ry$, 保角

$$\langle x', y' \rangle = x'^T y' = x^T R^T Ry = x^T y = \langle x, y \rangle$$

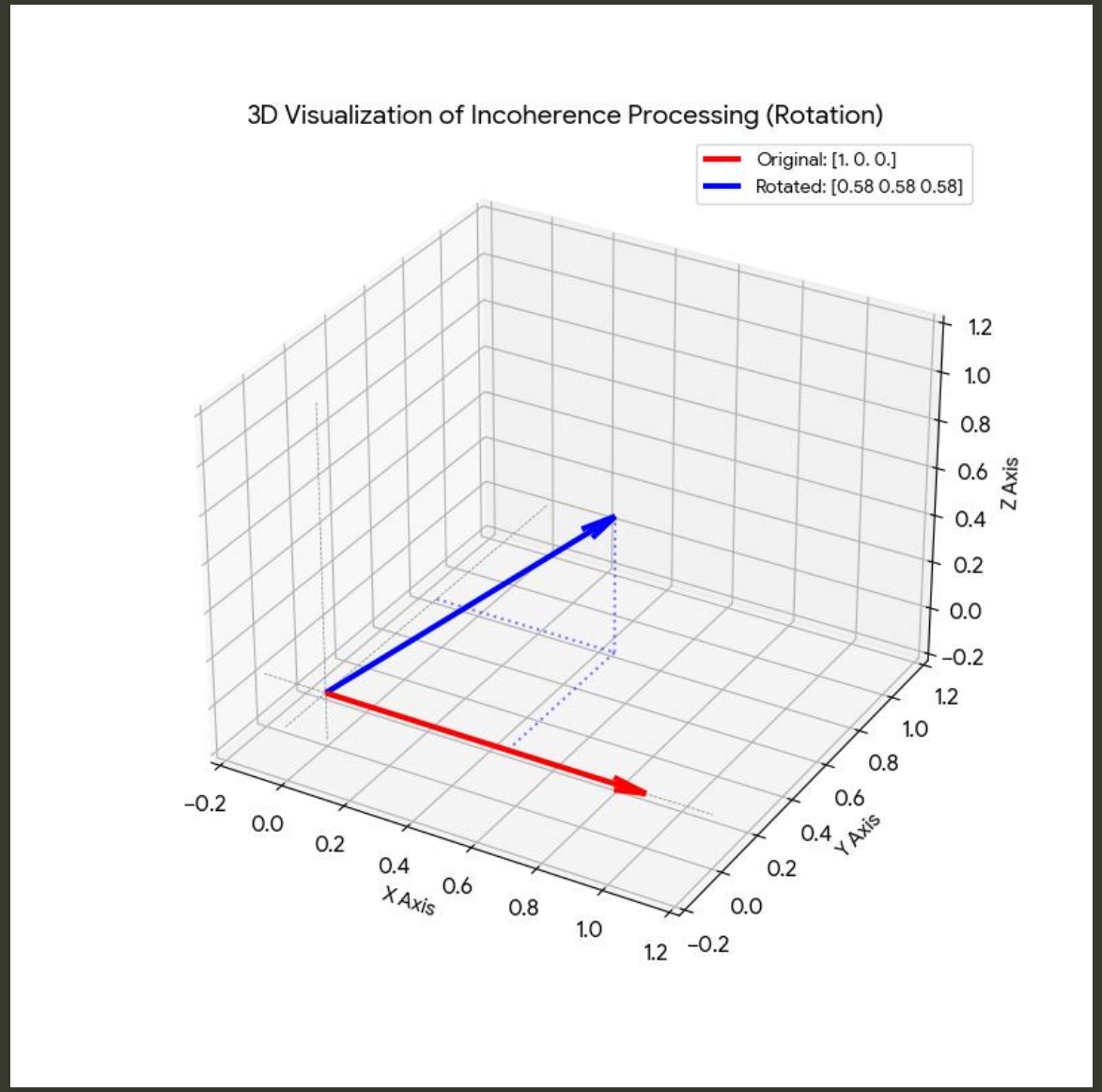
- 所以 $y = Rx$ 相當於 "鋼體" 座標旋轉

有極值的 Tensor

- 不管 per-tensor, per-channel, per-group, 共享同一組 qparam 的 tensor 其值分布如果有 outliers 則對量化不友善
- 極端情況為 one-hot vector: e.g. $x^T = [1, 0, 0]^T$
- 但如果先套用正交矩陣, $y = Rx$

$$R = \begin{bmatrix} 0.577 & 0.707 & 0.408 \\ 0.577 & -0.707 & 0.408 \\ 0.577 & 0.000 & -0.816 \end{bmatrix}$$

  - 則 $y^T = [0.577, 0.577, 0.577]^T$
  - y 的分布非常的量化友善!


3D Visualization of Incoherence Processing (Rotation)
Original: [1. 0. 0.]
Rotated: [0.58 0.58 0.58]
X Axis
Y Axis
Z Axis
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2

矩陣乘法套用旋轉

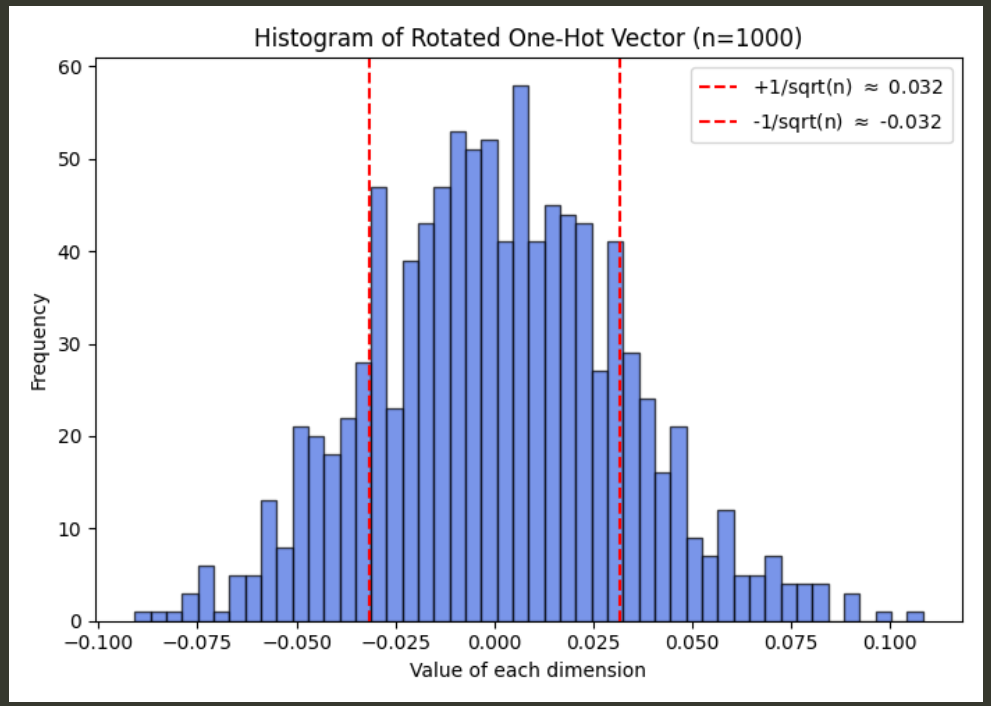
- $y = Wx$, 其中 $W \in \mathbb{R}^{m \times n}$, input $x \in \mathbb{R}^{n \times 1}$, output $y \in \mathbb{R}^{m \times 1}$
- 考慮 $y = Wx = (U^T U)W(V^T V)x = U^T(UWV^T)(Vx) = U^T \tilde{W} \tilde{x}$
- 透過 V 旋轉, 能使得 $\tilde{x} = Vx$ 好量化
- 透過 U 旋轉, 能使得 $\tilde{W} = UWV^T$ 好量化
- 則 $\tilde{W}\tilde{x}$ 因為量化誤差很小, 所以很精準, 此時多乘一個 U^T 即可還原回 Wx

- PS: 或是更極致的情況, $Wx = (WV^T)(Vx)$, 用一個 V 就同時讓 WV^T 和 Vx 都好量化

怎麼選擇 R?

- 隨機就好!
- 維度 1000 的 one hot vector 做實驗

```
import numpy as np
from scipy.stats import ortho_group
np.random.seed(42)
n = 1000
v = np.zeros(n)
v[0] = 1.0
# 隨機生成一個 n x n 的正交矩陣 (均勻分佈於 O(n) 空間)
U = ortho_group.rvs(dim=n)
# 將隨機正交矩陣乘上原本的向量
v_rotated = U @ v
```



- 大部分的值已經落在 ±0.032 之間了

隨機的數學保證

- 為什麼隨機可以?
- 機率理論中有個不等式: Concentration inequality
  - 在探討什麼樣的情況, random variable 偏離平均值太遠的機率, 不會太高
- QuIP 論文的 **[Lemma 9]:**
  - x 是單位球上表面的一個隨機向量, i.e. $\|x\| = 1$.
  - 對任意 1-Lipschitz 函數 $F: x \mapsto \mathbb{R}$, 存在 A, C 與維度 n 無關, 滿足

$$P(F(x) - \mathbb{E}[F(x)] \geq t) \leq C\exp\left(-\frac{nt^2}{A}\right)$$

- 白話來說, $F(x)$偏離期望值超過 t的機率不會太大 (不會超過 $C\exp(-nt^2/A)$)

隨機的數學保證

- 為什麼隨機可以?
- 機率理論中有個不等式: Concentration inequality
  - 在探討什麼樣的情況, random variable 偏離平均值太遠的機率, 不會太高
- QuIP 論文的 **[Lemma 10]:**
  - 設 $B \in \mathbb{R}^{m \times n}$, 且 x 是 $\mathbb{R}^n$ 單位球表面上的一個隨機向量, i.e. $\|x\| = 1$.
  - 存在 A, C 與維度 n 無關, 滿足

$$P\left(\left(\frac{\|Bx\|}{\|B\|_F}\right)^2 \geq \frac{A'}{n}\log\frac{C'}{\delta}\right) \leq \delta$$

- 白話來說, 投影向量超過一個值的機率不會太大.

隨機的數學保證 (Conti.)

- QuIP 論文的 **[Lemma 10]:**
  - 設 $B \in \mathbb{R}^{m \times n}$, 且 x 是 $\mathbb{R}^n$ 單位球表面上的一個隨機向量, i.e. $\|x\| = 1$.
  - 存在 A, C 與維度 n 無關, 滿足

$$P\left(\left(\frac{\|Bx\|}{\|B\|_F}\right)^2 \geq \frac{A'}{n}\log\frac{C'}{\delta}\right) \leq \delta$$

- 當 $B \in \mathbb{R}^{1 \times n}$為一個 row vector, 令 $b = B^T$, 則

$$\frac{\|Bx\|}{\|B\|_F} = \left(\frac{b}{\|b\|}\right)^T x$$

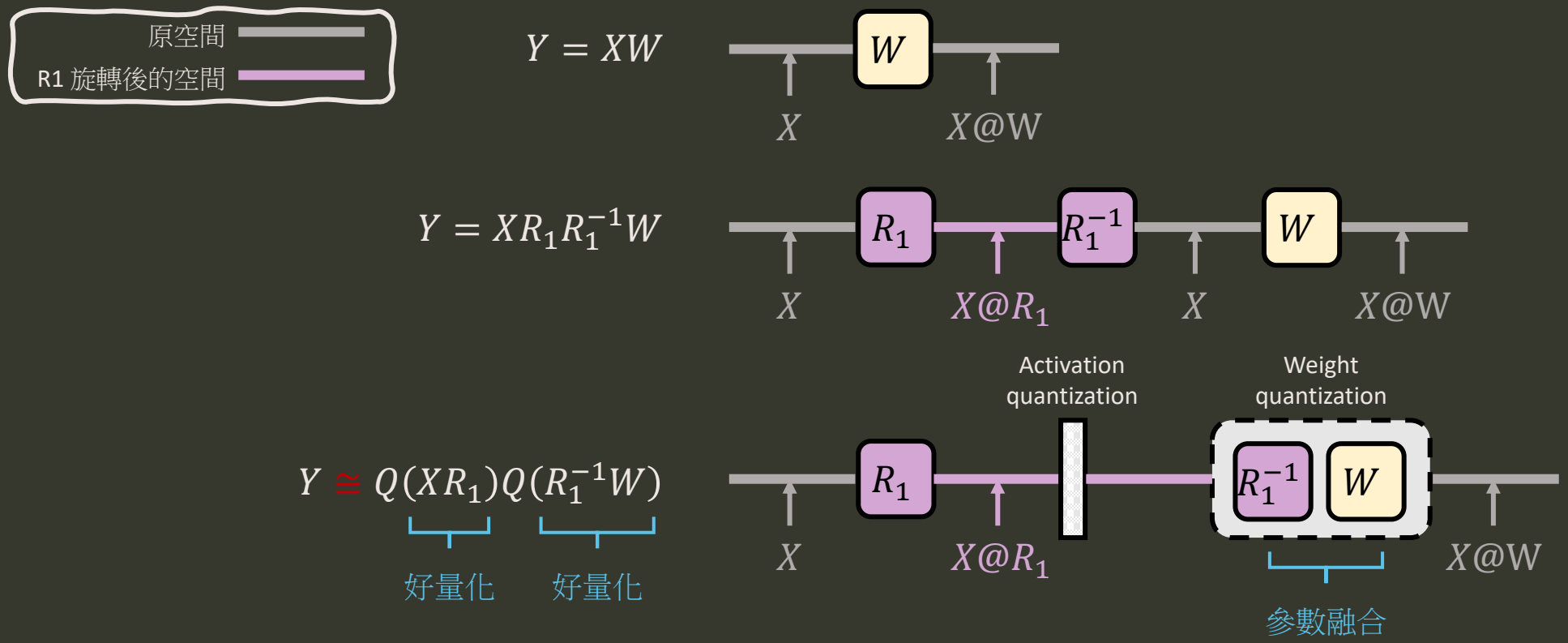
物理意義: 將 x 投影到 b 方向上

- 隨機投影的長度平方, 要異常大的機率是極度微小的!
- 角色對調: "B 給定, x 隨機" 等價於 "x 給定, B 隨機" ➔ 隨機旋轉矩陣

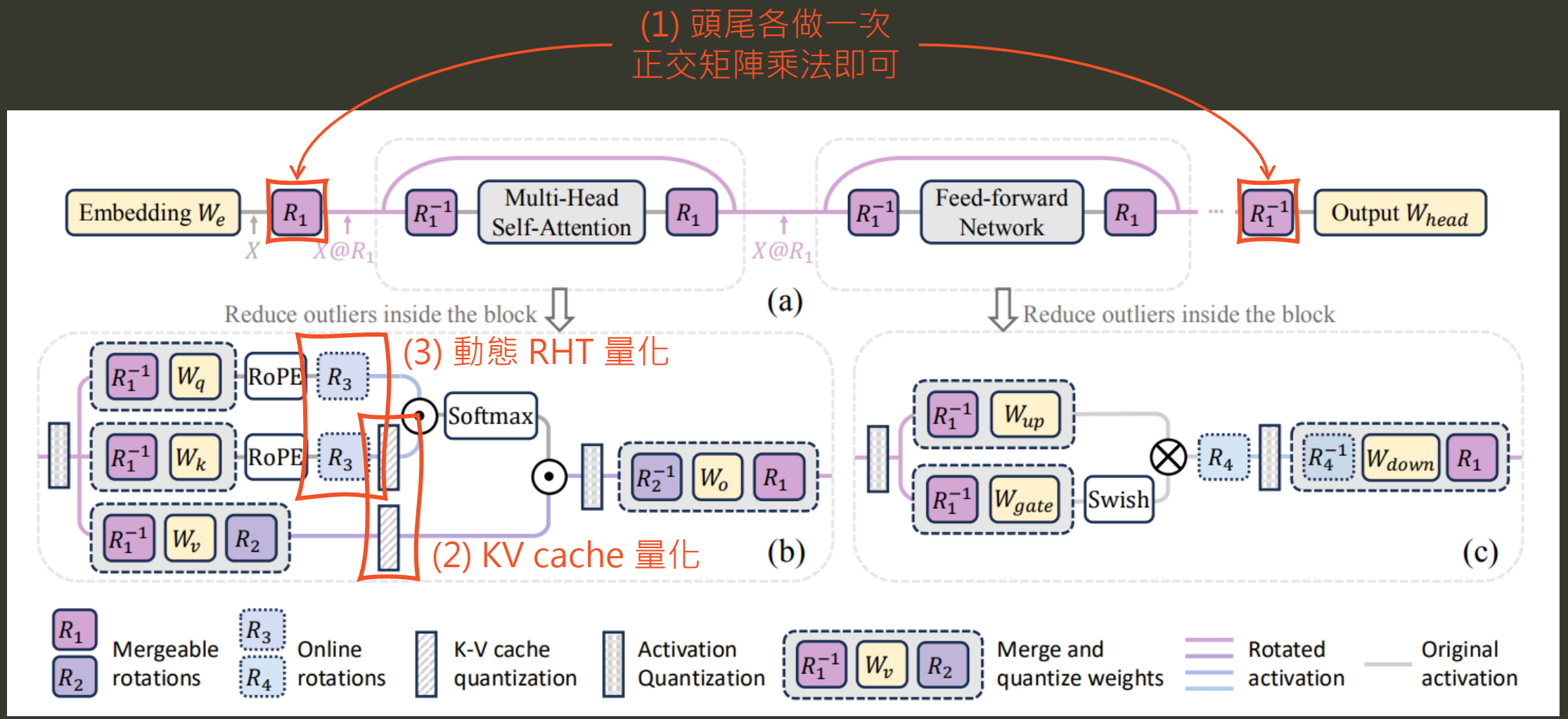
SpinQuant Linear Layer 架構

- 注意到我們用改右乘表示 $Y = XW$
- 用一個 R 就同時讓 WV^T 和 Vx 都好量化: $Y = XW = (XR)(R^TW)$

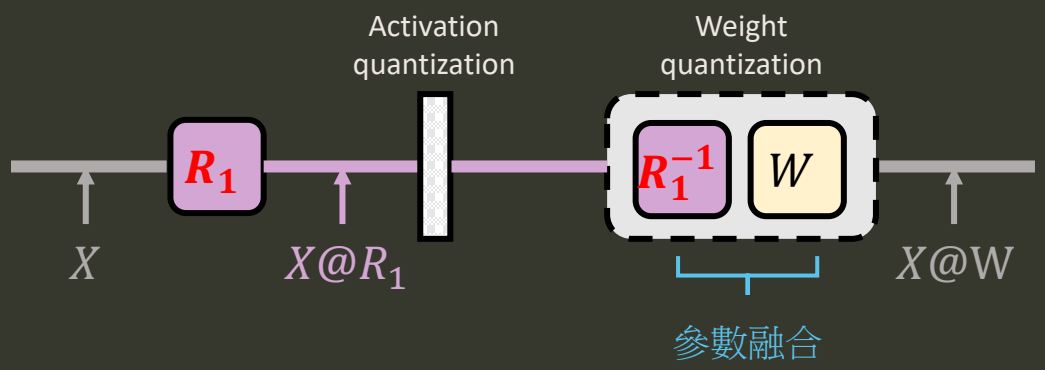
https://arxiv.org/abs/2405.16406



SpinQuant 全架構



SpinQuant Rotation Matrix 選擇

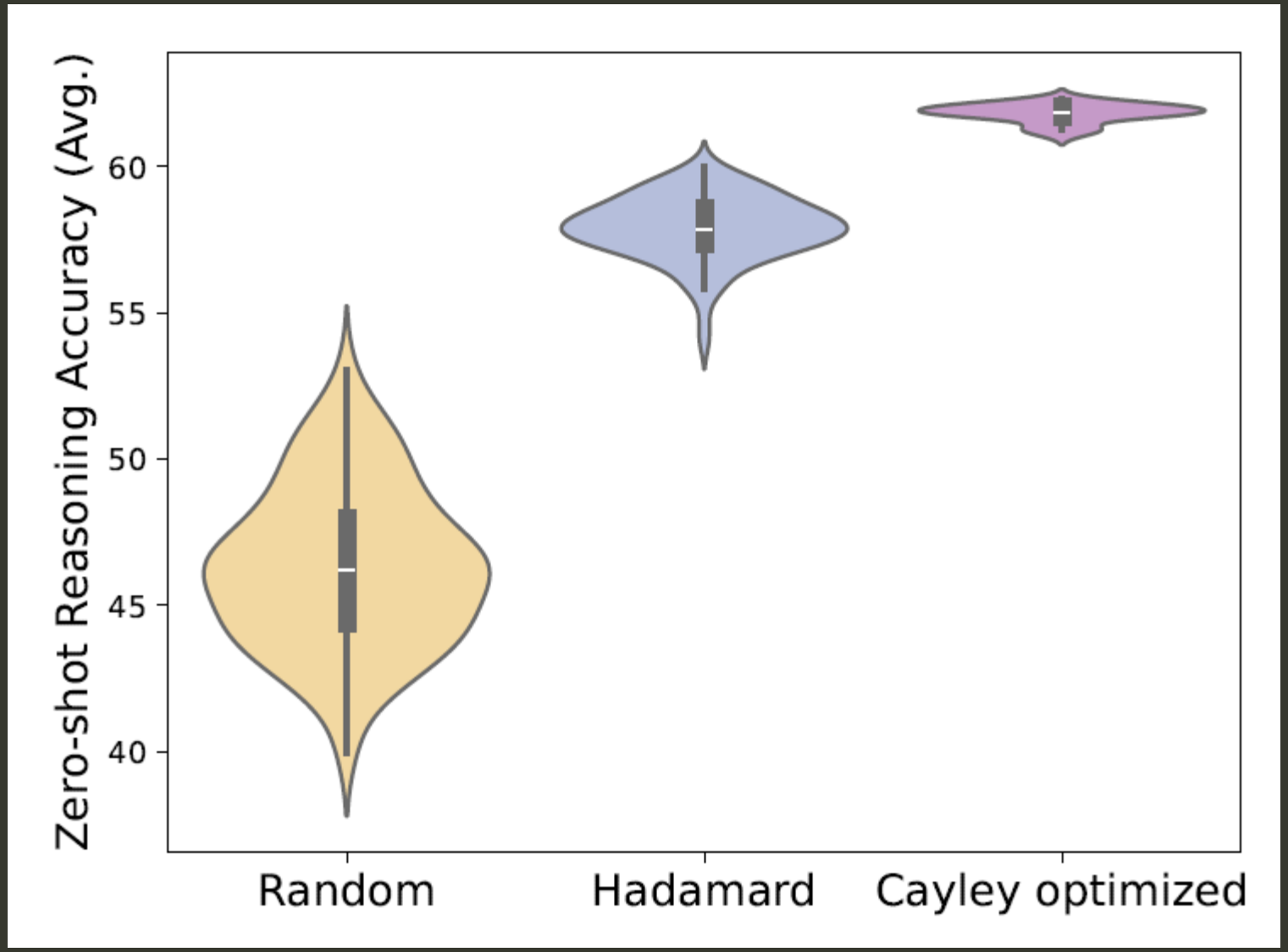


$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$ (最基本的阿達瑪矩陣，其他阿達瑪矩陣都基於此來建造)

$$H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

$$H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

- R_1 的選擇:
  1. 全隨機正交矩陣: dim 一大的話, 矩陣相乘 cost 高. (QuIP 論文有提一個解法, 這裡步展開)
  2. 隨機 Hadamard Transform, (RHT): RHT 仍是正交矩陣, 且元素只有 $\{1, -1\}$, 所以不用浮點乘法運算 (QuIP# 論文)
  3. SpinQuant 提出**直接用學的**: 難點在於 SGD 怎麼保證更新後仍”正交”？使用 Cayley SGD 方法. PS: 量化 OP 使用 STE 技巧.


Zero-shot Reasoning Accuracy (Avg.)
60
55
50
45
40
Random
Hadamard
Cayley optimized

Summary

- Cayley SGD 方法: 參考 GitHub "facebookresearch/SpinQuant" 的 optimizer.py 即可
- 搭配使用 STE

- SpinQuant 在 2025 提出時, 在 W4A4KV4 設定下的 LLaMA-2 7B/13B/70B models 跟 full precision 很接近, 當時最好的結果

- 還有一個 PTQ 方法稱 PrefixQuant: 針對 KV-cache 可以再壓縮 (token-wise level, 而 SpinQuant 是 channel-wise)

- Google 的 TurboQuant 其中一步驟也是利用隨機旋轉矩陣 (論文裡稱 PolarQuant), 有趣的是, 可以證明隨機旋轉後, 每個維度互相獨立且必為 Beta 分佈. 既然已知必定為 Beta 分布, 那就可以事先決定好量化參數了. 利用這種做法(加其他步驟)對 KV cache 極大的量化!