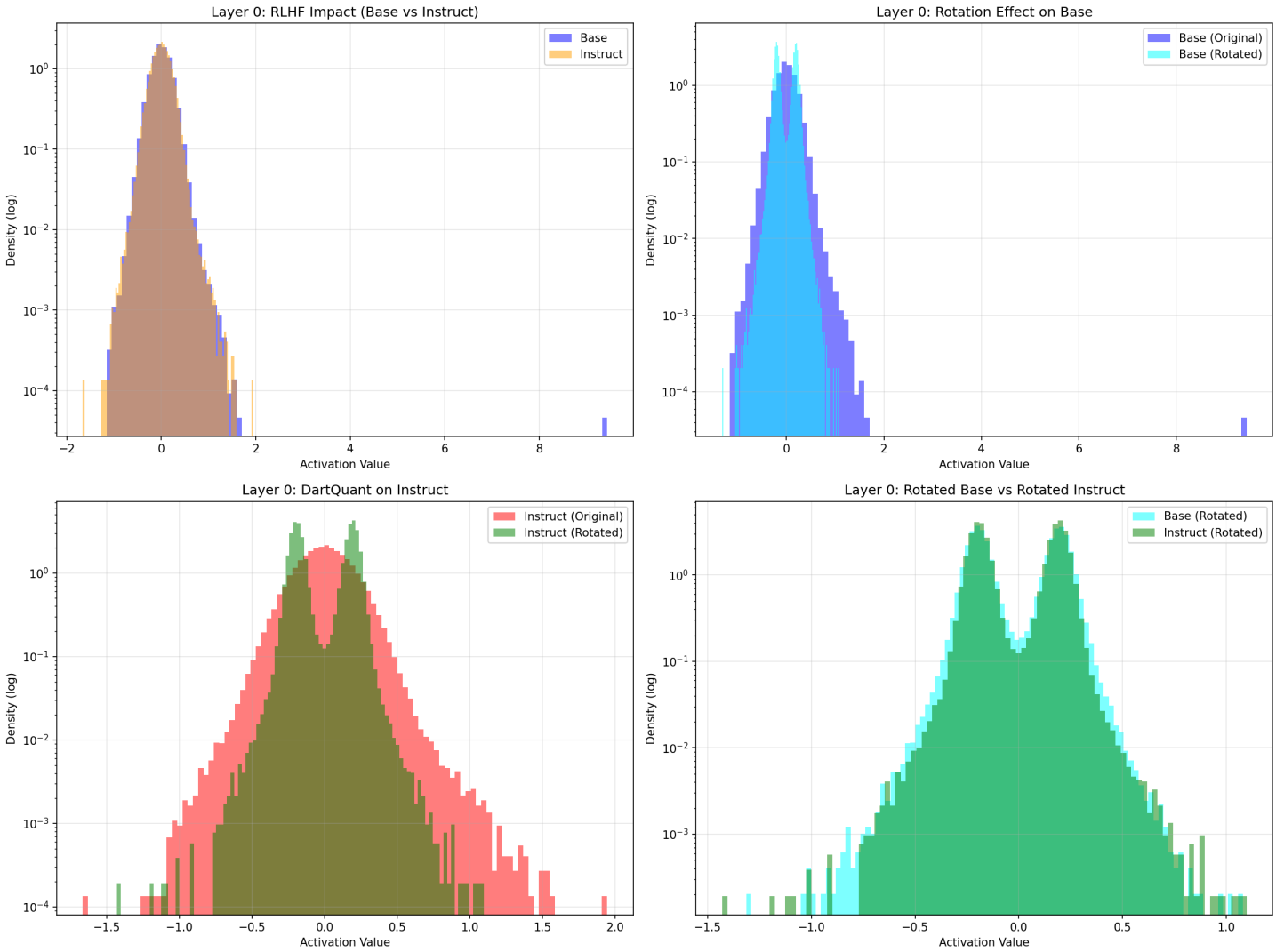
Chapter 1: Post-train Quantization

全精度（FP16, FP32）的大语言模型(LLM) 占用的存储，内存，显存消耗极大，使得很难部署在算力受限的设备。为了解决这个问题，*Post-Training Quantization (PTQ)* 成为一个极具吸引力的方案。因为对比 *Quantization-Aware-Training (QAT)*， PTQ能够直接对模型进行压缩，无需重新训练（但依然需要calibration），具有强实用性。

但是，PTQ在实践中面临一个核心难题：激活值离群点 (Activation Outliers)。对于任何一个LLM的线性层的输出 (Y)，应该满足：

$$Y = XW^\top \tag{1}$$

其中X是激活值的矩阵，W是大模型里固定的权重。下图是Llama-3.2-1B-Base 和 Instruct在第一层的激活分布图



在左上子图中我们可以清晰的看到Base模型的激活值有一个明显的outliers(~ 8)。**在量化过程中，为了同时容纳这些正常值和极端值，量化范围被迫拉得非常宽。这导致了绝大多数集中在中心区域的数值，只能被映射到极少数几个量化级别上，从而严重损失了模型的精度**。 目前有两种路径来处理这些异常的激活值：

1. 使用一些算法在PTQ的过程中自动的处理异常值。

- 例如在4-bit 量化中就有GPTQ, AWQ, SmoothQuant 等等方法

2. 对激活矩阵进行预处理，使得异常激活值在PTQ前就能被处理

- DartQuant, SpinQuant, ...

Chapter 2 DartQuant

目前（2025年末到现在）业界主推的是方法2，具体实现是引入一个旋转矩阵(Rotation Matrix)。其原理是引入一个正交(orthogonal)且范数(norm)为1的矩阵在不改变模型输出，不增加额外推理成本的情况下平滑掉离群点，

$$Y = (XR)(WR)^\top = XW^\top. \tag{2}$$

也就说我们的目标是 *找到一个矩阵R使得旋转过后的激活矩阵,RX 在量化后在给定的数据集上能有尽量好的表现*。

像SpinQuant和OSTQuant这类的方法通常采用 “端到端微调” 的方式来寻找最优的旋转矩阵。也就是说给定一个数据集，把这个矩阵当作可学的参数然后进行梯度下降寻找最优R。但是这种端到端的微调方式又以下局限性，并且和PTQ所追求的快速部署的目标背道而驰：

1. 计算成本高：根据SpinQuant的论文，优化这样一个70B参数的模型需要数百GiB的显存和运行数十个GPU小时来运行 （需要A100这样的高端GPU来运行）。
2. 容易过拟合：由于需要在一个数据集上微调，所以容易过拟合这些特定的分布特征，导致过拟合。
3. 优化难度高：其实还是第一点，但这里的优化难度主要是针对R。因为R必须要时刻保持其正交性，导致计算需要使用专用的优化器，其时间复杂度非常非常高。

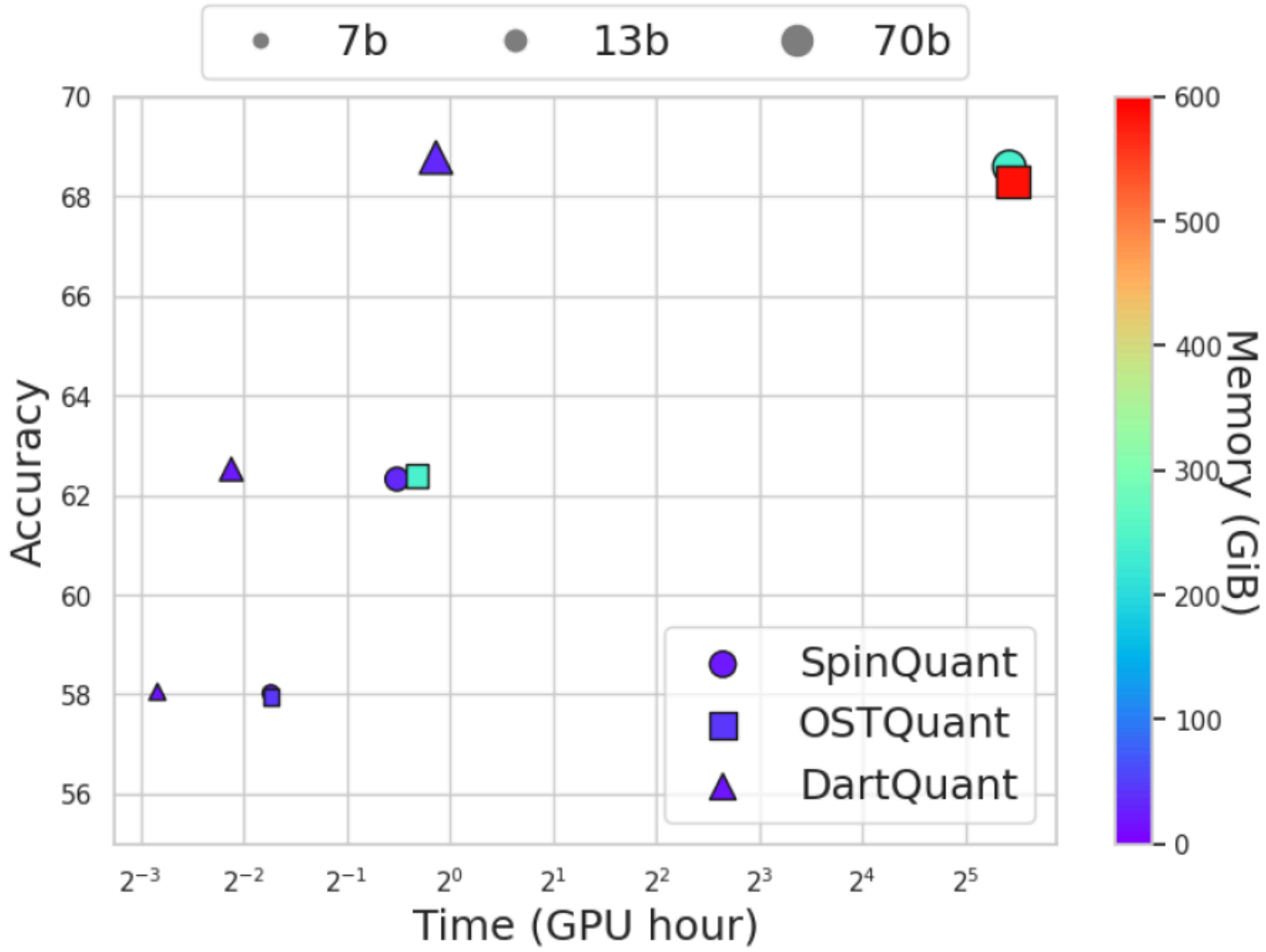


Figure 1: Comparison of computational costs across different rotation optimization methods.

2.1 DartQuant创新点

为了解决以上问题DartQuant重新整理了原来的目标，现在我们的目标是：**到一个矩阵R使得旋转过后的激活矩阵,RX尽量符合均匀分布，减少Outliers的数量**。换句话说，DartQuant不再试图通过微调来提升模型在某个具体任务上的分数。他的目标更直接和简单：**寻找一个最优的旋转矩阵，将激活值的分布形态变得更适合量化(Unif)。**

Question: 为什么要符合Unif分布?
因为要4-bit量化中只有16个离散的整数格点可用。信息论指出，当所有量化格点被使用的概率相等时才能最大化熵。

为了实现这个思想，DartQuant引入了两个技术创新：

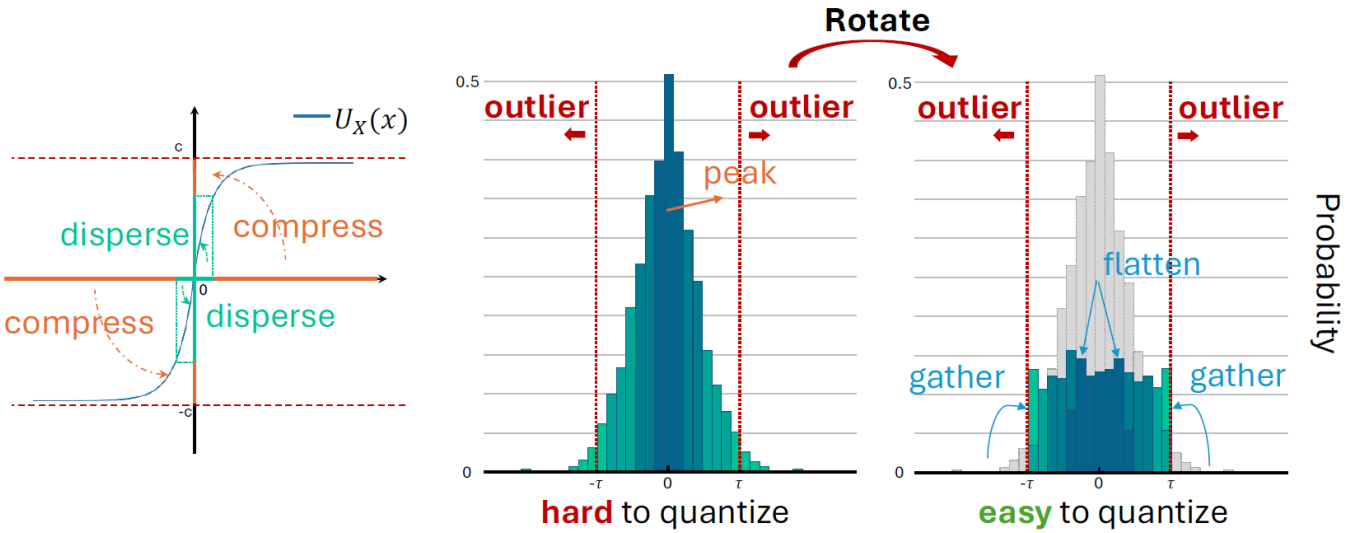
1. Whip损失函数
2. OR-Orth优化器

2.2 Whip损失函数

Whip损失函数源自于对原始激活值的洞察。原始的分布通常呈现出 *拉普拉斯分布* 的形态， 中心数据集中，heavy tail。这对量化极为不利，尤其是参数量小的模型(~1B - 2B)。Whip损失函数主要是源于Cumulative Distribution Function (CDF)，用于把原来是Laplace的分布转换成Unif分布。

Question: 你怎么知道所有模型的所有激活值是不是都是Mean 0的拉普拉斯分布的?

1. LLM常用的ReLU/SwiGLU激活函数和残差连接累积下来导致尖峰特性
2. 且经过原论文Appendix里面(还有我对Llama, Qwen)的实验验证，所有的模型的所有的线性层的激活值都是尖峰，有outliers的分布。除了有极个别层的激活值的mean不是0（Llama 3.2 1B 忘了那一层的mean是0.2，但不是很偏离问题不大）。



$$\mathcal{L}_{\text{whip}} := \sum_{i=1}^{c_{\text{in}}} \exp(-|x_i|)$$

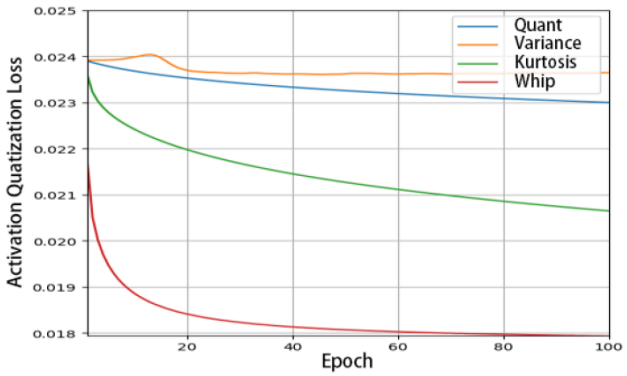
where, $\mathbf{x} = [x_1, ..., x_{c_{\text{in}}}] \in \mathbb{R}^{c_{\text{in}}}$ 是激活向量。具体推导请见DartQuant的文章。

2.3 OR-Orth

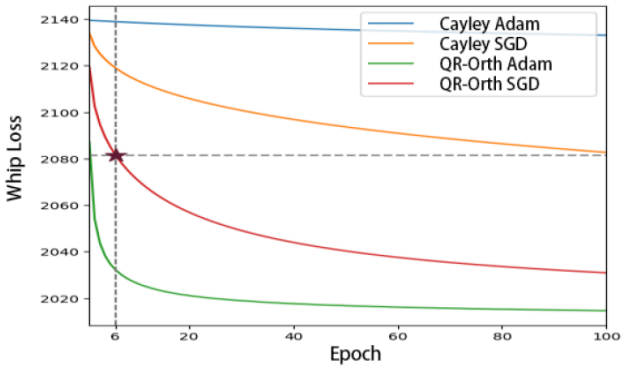
在传统方法中直接使用正交优化器(Cayley SGD)训练一个R时间复杂度特别高，OR-Orth里面我们可以定义一个隐变量矩阵（普通的，无约束的非正交矩阵），然后进行QR分解提取出一个正交的，范数不变的旋转矩阵。然后把分解出来的orthogonal matrix送到Whip loss再对隐变量矩阵SGD进行梯度下降。

Algorithm 1 Rotational Distribution Calibration with QR-Orth Optimizer

Input: LLM model LLM, calibration sequence S, initial latent parameter $Z_0 \in \mathbb{R}^{n \times n}$, max iterations T, learning rate η.
Output: Rotational matrix $R \in \mathbb{R}^{n \times n}$.
$X \leftarrow LLM(S)$
$X \leftarrow token\_sampling(X)$
$Z \leftarrow Z_0$
for $k = 0$ to T **do**
  $R \leftarrow qr\_decomposition(Z)$
  $O \leftarrow X@R$
  $\mathcal{L} \leftarrow Whip(O)$
  $Z \leftarrow Z - \eta \frac{\partial \mathcal{L}}{\partial Z}$
end for

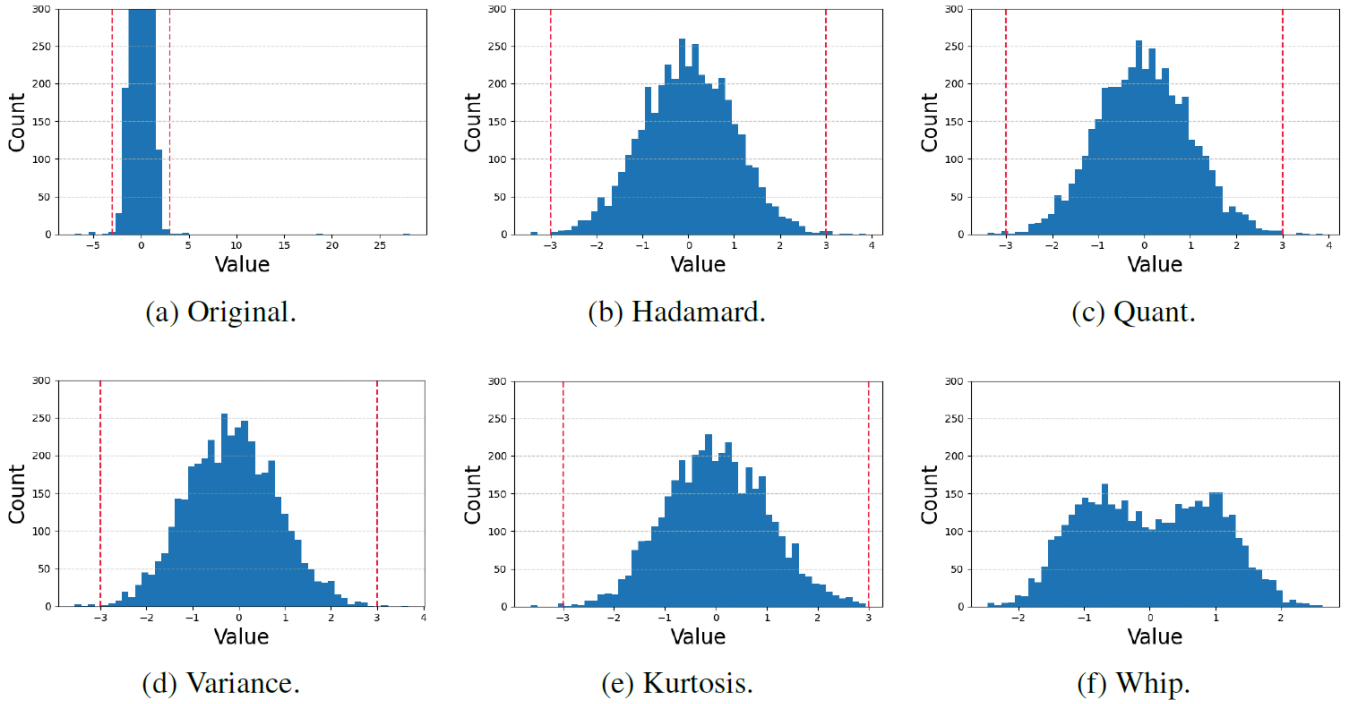


(a) Loss comparison.



(b) Cayley vs QR-Orth.

Figure 7: Comparison of activation quantization loss and convergence performance using different optimization methods.



(a) Original. (b) Hadamard. (c) Quant.

(d) Variance. (e) Kurtosis. (f) Whip.

Figure 6: Histograms of Activation Distributions After Rotation by Different Rotation Matrices. The region outside the red dashed line represents the outliers.

2.4 如何把旋转矩阵应用到Transformer架构模型的不同组件?

我们知道现代的LLM里面的Transformer架构包含：

1. 通义线性层
2. 注意力层
3. 前馈层
4. MoE架构等路由组件

   在 DartQuant 框架中，我们定义了四个正交矩阵，目的是在不改变模型全精度数学输出的前提下（即计算不变性，Computational Invariance），旋转激活空间以消除离群值（Outliers）。

设 d_{model} 为隐藏层维度，d_{head} 为注意力头维度，d_{inter} 为 FFN 中间层维度。

- $R_1 \in \mathbb{R}^{d_{model} \times d_{model}}$（残差流旋转矩阵 / The Residual Rotation）：这是一个**可学习的**正交矩阵，由 DartQuant 使用 QR-Orth 方法优化得到 。它作用于 Transformer 模块的输入激活 X，并贯穿整个残差连接（Residual Stream）。
- $R_2 \in \mathbb{R}^{d_{head} \times d_{head}}$（Value-Head 旋转矩阵 / The Value-Head Rotation）：这是一个**可学习的**正交矩阵 。它专门应用于自注意力机制（Self-Attention）内部的 Value 头。由于注意力头工作在子空间 d_{head} 中，因此需要独立的旋转矩阵 R_2
- $R_3 \in \mathbb{R}^{d_{head} \times d_{head}}$（Key/Query 旋转矩阵 / The Key/Query Rotation）：这是一个固定的随机阿达玛矩阵（Fixed Random Hadamard Matrix），不参与训练 。它通过**在线计算**的方式作用于 Query (Q) 和 Key (K)，用于平滑分布，特别有利KV-Cache 的量化。

- $R_4 \in \mathbb{R}^{d_{inter} \times d_{inter}}$（FFN 激活旋转矩阵 / The FFN Activation Rotation）：
  这是一个固定的随机阿达玛矩阵（Fixed Random Hadamard Matrix） 。它通过在线计算的方式作用于 FFN（或 MoE 专家）非线性激活函数之后的中间状态，用于平滑下投影层（Down-projection）的输入。

2.4.1 A. 全局残差流（The Global Residual Stream - R_1）：

对于任意线性层 $Y = XW^\top$，若输入 X 被旋转为 $\tilde{X} = XR_1$，则权重矩阵 W 需要吸收逆旋转 $R_1^\top$。

$$Y = (XR_1)(R_1^\top W^\top) = X(R_1 R_1^\top)W^\top = XW^\top$$

融合方式：输入侧权重（$W_q, W_k, W_v, W_{up}, W_{gate}$）：$R_1$ 右乘到权重矩阵中。

$$\tilde{W} = WR_1$$

$$\tilde{X}\tilde{W}^\top = (XR_1)(WR_1)^\top = XR_1R_1^\top W^\top = XW^\top$$

2.4.2 B. RoPE 与注意力机制（Attention - R_2, R_3）：

1. 输入投影（Input Projection - W_q, W_k, W_v）：所有输入权重首先吸收残差流的旋转 R_1：

$$\tilde{W}_q = W_q R_1, \quad \tilde{W}_k = W_k R_1, \quad \tilde{W}_v = W_v R_1$$

2. RoPE 与 R_3（在线计算）：旋转位置编码（RoPE）对旋转敏感，因此不能直接融合(因为RoPE本身就是用旋转矩阵，对RoPE感兴趣可以查看原始论文)。R_3 被设计为在 RoPE 之后在线应用，以平滑 Q 和 K。令 $\mathcal{P}(\cdot)$ 为 RoPE 函数：

$$Q_{rot} = \mathcal{P}(\tilde{X}\tilde{W}_q^\top)R_3$$

$$K_{rot} = \mathcal{P}(\tilde{X}\tilde{W}_k^\top)R_3$$

由于 R_3 是正交矩阵，它在点积注意力计算中会自动抵消：

$$\text{Attention Score} \propto (Q_{rot}K_{rot}^\top) = (Q'R_3)(K'R_3)^\top = Q'(R_3R_3^\top)K'^\top = Q'K'^\top$$

这里$Q' = \mathcal{P}(\tilde{X}\tilde{W}_q^\top)$, K'同理。

3. Value 头与 R_2（融合）：为了平滑 Value 头的输出，R_2 被融合进 W_v 的输出端。

$$\tilde{W}_v = R_2^\top W_v R_1$$

(注：输入端吸收 R_1，输出端产生被 R_2 旋转的激活值。)

$$\begin{aligned}\tilde{X}_{in}\tilde{W}_v^\top &= (XR_1)(R_2^\top W_v R_1)^\top \\ &= (XR_1)(R_1^\top W_v^\top R_2) \\ &= X(R_1R_1^\top)W_v^\top R_2 \\ &= (XW_v^\top)R_2 \\ &= VR_2\end{aligned}$$

定义 X_{val} 为多头注意力机制拼接后的中间输出向量，W_{out}为将其映射回残差流维度的输出线性层权重矩阵。W_{out} 的输入（即 X_{val}）现在处于 R_2 基底（由 W_v 输出），而其输出必须回到残差流的 R_1 基底。因此，W_{out} 需要在其输入端消除 R_2，并在输出端引入 R_1 。

$$\tilde{W}_{out} = R_1^\top W_{out} R_2$$

$$\begin{aligned}\text{Output} &= (X_{\text{val}}R_2)\tilde{W}_{out}^\top \\ &= (X_{\text{val}}R_2)(R_1^\top W_{out}R_2)^\top \\ &= X_{\text{val}}R_2(R_2^\top W_{out}^\top R_1) \\ &= X_{\text{val}}(R_2R_2^\top)W_{out}^\top R_1 \\ &= (X_{\text{val}}W_{out}^\top)R_1\end{aligned}$$

最终输出被 R_1 旋转，与残差流一致。

2.4.3 C. FFN 与 门控机制（Gate - R_4）

FFN（或 MoE 专家）结构通常包含非线性激活（如 SwiGLU），这破坏了线性旋转的融合性质，因此必须引入在线旋转 R_4 。

$$(\text{SwiGLU}(x, W, V, b, c) = \text{Swish}_\beta(xW + b) \otimes (xV + c)),$$

$$\begin{aligned}\text{Gate (Gate Proj):} \quad & G = \text{SiLU}(XW_{gate}^\top) \\ \text{Up (Up Proj):} \quad & U = XW_{up}^\top \\ \text{中间激活 (Intermediate):} \quad & H = G \odot U \quad (\text{element-wise multiplication}) \\ \text{最终输出 (Down Proj):} \quad & Y = HW_{down}^\top\end{aligned}$$

Up/Gate 投影：权重吸收残差流的 R_1：

$$\tilde{W}_{gate} = W_{gate}R_1, \quad \tilde{W}_{up} = W_{up}R_1$$

激活与 R_4（在线计算）：令 σ 为非线性激活函数（如 SiLU）。计算中间状态 H 后，显式乘以 R_4 以平滑分布：

$$H = \sigma(\tilde{X}\tilde{W}_{gate}^{\top}) \odot (\tilde{X}\tilde{W}_{up}^{\top})$$

$$H_{\text{rot}} = HR_4$$

($\tilde{X}$是R1旋转的)

Down 投影（W_{down}）：
W_{down} 的输入现在处于 R_4 基底，输出需回到 R_1 基底。

$$\tilde{W}_{down} = R_1^{\top} W_{down} R_4^{\top}$$

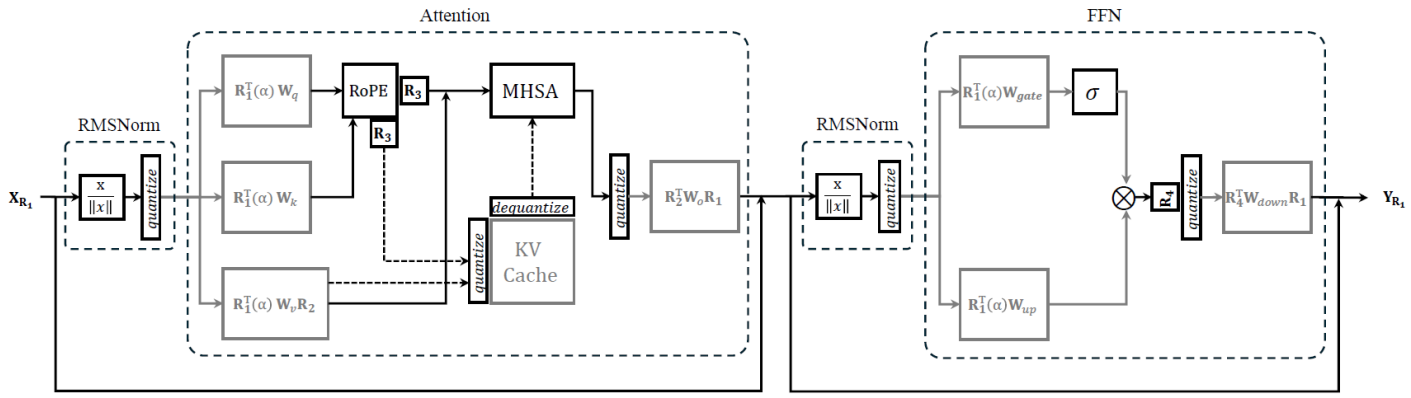
(注意：由于 R_4 是阿达玛矩阵，通常是对称的，但数学形式上应为 R_4 以抵消输入旋转。)

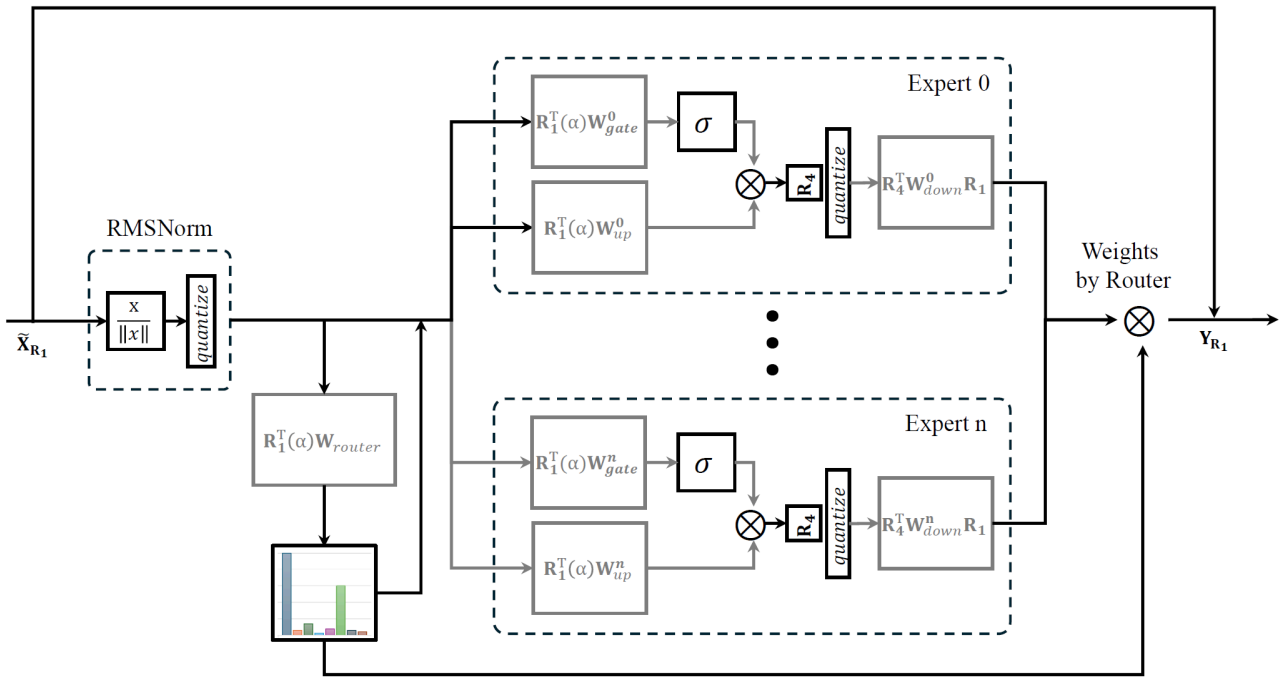
2.4.4 D. 混合专家模型（MoE - Mixture of Experts）

在 MoE 模型中（如 Mixtral），旋转矩阵的应用逻辑与 FFN 类似，但作用于每个专家（Expert）内部 。

- Router（路由）： Router 接收的是全局残差流输入（已被 R_1 旋转），直接在 R_1 基底上计算路由权重。
- 专家内部（Inside Experts）：每个专家的 W_{gate}^{i}, W_{up}^{i} 均吸收 R_1。在线 R_4： 在每个专家内部的激活函数后，应用在线 Hadamard 旋转 R_4。通常所有专家共享同一个 R_4 以减少开销。每个专家的 W_{down}^{i} 均吸收 R_4（输入侧）和 $R_1^{\top}$（输出侧）。

2.4.5 完整架构图如下（经典Transformer和MOE Transformer）





Chapter 3: 创新 - Goal

3.1 Sliced Wasserstein Distance (SWD)

3.1.1 SWD推导

设 $X \in \mathbb{R}^{B\times D}$ 为原始输入激活值矩阵（其中 B 为 batch size，D 为特征维度）。我们引入一个正交旋转矩阵 $R \in \mathbb{R}^{D\times D}$（即 $R^\top R = I$），得到旋转后的激活值 $\tilde{X} = XR$。

我们的目标是优化 R，使得 $\tilde{X}$ 的经验分布 $P_{\tilde{X}}$ 尽可能接近一个预定义的目标分布 Q。为了量化两个分布之间的差异，我们采用 1-Dimensional 2-Wasserstein Distance（即 Sliced Wasserstein Distance 的核心组件）。

我们将旋转后的所有激活值展平为一个向量 $\mathbf{x} \in \mathbb{R}^N$（其中 $N = B \times D$），作为损失函数的输入。

对于两个一维概率分布 P 和 Q，其累积分布函数 (CDF) 分别为 F_P, F_Q，分位函数 (Quantile Function, 即 F^{-1}) 分别为 F_P^{-1}, F_Q^{-1}。根一维 W_2 距离具有闭式解：

$$W_2^2(P,Q) = \int_0^1 \left|F_P^{-1}(u) - F_Q^{-1}(u)\right|^2 du$$

在离散观测下，我们无法直接获取 $F_{P_Y}^{-1}$ 的解析形式，需使用经验分位函数 (Empirical Quantile Function) 进行估计。

Definition 1 (Order Statistics). 设 $\mathbf{y} = \{y_1, \ldots, y_N\}$ 为旋转后的激活值集合。定义置换 σ 使得 $\mathbf{y}$ 按非降序排列，记排序后的向量为 $\mathbf{y}_{\text{sorted}}$，其第 i 个元素为次序统计量：

$$y_{(i)} = y_{\sigma(i)}, \quad \text{s.t.} \quad y_{(1)} \leq y_{(2)} \leq \cdots \leq y_{(N)}$$

Approximation 1 (Empirical Quantile). 经验分布的分位函数 $F_{P_Y}^{-1}(u)$ 是分段常数函数。对于离散采样点 $u_i = \frac{i-0.5}{N}$ (或近似为 $\frac{i}{N}$)，其值精确等于次序统计量：

$$F_{P_Y}^{-1}(u_i) = y_{(i)}$$

Approximation 2 (Target Quantile): 目标分布 $Q = \text{Uniform}[-b, b]$ 的分位函数为线性函数 $F_Q^{-1}(u) = 2b \cdot u - b$。在离散点 u_i 处的目标值为：

$$t_i = F_Q^{-1}\left(\frac{i}{N}\right) = 2b \cdot \frac{i}{N} - b$$

Lemma 1 (Calculation of b): 如果目标分布是$Unif[-b, b]$，那边界$b = \sqrt{3} \cdot \text{RMS} = \sqrt{3}\sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}$

Proof: 为了让旋转后的激活值映射到均匀分布$Uni[-b, b]$是well-posed，目标分布的二阶矩必须等于输入的二阶矩。否则无论R如何旋转原始激活值，两者永远无法完全匹配。

- 输入数据的二阶矩：设输入数据 x 的均方根为 RMS，则其二阶矩为 $\mathbb{E}[x^2] = \text{RMS}^2$。
- 目标均匀分布 $U[-b, b]$ 的二阶矩：设随机变量 $T \sim U[-b, b]$，其概率密度函数为 $f(t) = \frac{1}{2b}$。其二阶矩为：

$$\mathbb{E}[T^2] = \int_{-b}^{b} t^2 \cdot \frac{1}{2b} dt = \frac{1}{2b}\left[\frac{t^3}{3}\right]_{-b}^{b} = \frac{1}{2b}\left(\frac{b^3}{3} - \frac{-b^3}{3}\right) = \frac{1}{2b} \cdot \frac{2b^3}{3} = \frac{b^2}{3}$$

由此可得

$$\begin{aligned} \text{RMS}^2 &= \frac{b^2}{3} \\ \implies b &= \sqrt{3} \cdot \text{RMS} \quad \square \end{aligned}$$

将连续积分 $\int_0^1$ 离散化为求和，我们将 W_2 距离转化为均方误差 (MSE) 形式的损失函数：

$$
\begin{aligned}
\mathcal{L}_{\text{SWD}}(\mathbf{y}) &= \int_0^1 \left| F_{P_Y}^{-1}(u) - F_Q^{-1}(u) \right|^2 du \\
&\approx \frac{1}{N} \sum_{i=1}^{N} \left(F_{P_Y}^{-1}\left(\frac{i}{N}\right) - F_Q^{-1}\left(\frac{i}{N}\right) \right)^2 \\
&= \frac{1}{N} \sum_{i=1}^{N} (y_{(i)} - t_i)^2 \\
&= \frac{1}{N} \sum_{i=1}^{N} (y_{\sigma(i)} - t_i)^2
\end{aligned}
$$

3.1.2 梯度分析

Gradient Derivation:

$$
\frac{\partial \mathcal{L}_{\text{SWD}}}{\partial y_k} = \frac{\partial}{\partial y_k} \left[\frac{1}{N} (y_k - t_r)^2 \right] = \frac{2}{N} (y_k - t_r)
$$

其中 t_r 是固定的目标分位数值。

Case Study:
Extreme Outlier考虑一个极大的正离群值 $y_k \to +\infty$。Ranking: 由于 y_k 极大，它必然被排序到最后，即秩 $r \approx N$。Target: 对应的目标值为均匀分布的上界 $t_N \approx b$。

Gradient Magnitude:

$$
\lim_{y_k \to +\infty} \frac{\partial \mathcal{L}_{\text{SWD}}}{\partial y_k} = \frac{2}{N} (y_k - b) \propto y_k
$$

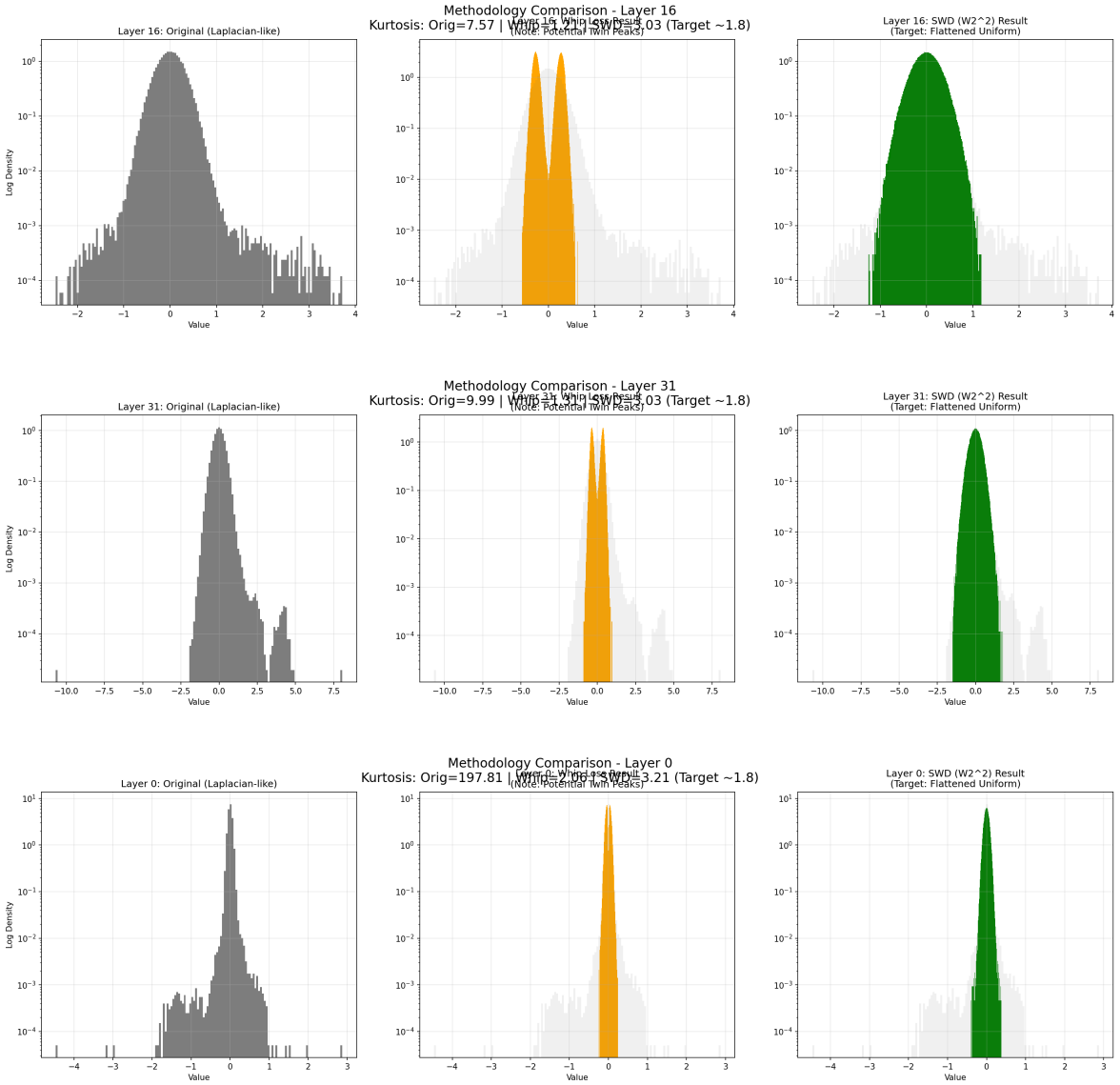
结论：梯度随离群值的幅度线性增长 (Linearly Scaling)。离群值越远，将其拉回目标区间 $[-b, b]$ 的梯度力越大。

Comparison with Whip Loss
回顾 Whip Loss $\mathcal{L}_{\text{Whip}} = \sum e^{-|y_k|}$ 的梯度：

$$
\left| \frac{\partial \mathcal{L}_{\text{Whip}}}{\partial y_k} \right| = e^{-|y_k|}
$$

当 $y_k \to +\infty$ 时，梯度趋向于 0 (Vanishing Gradient)。这在数学上解释了为何 Whip Loss 难以处理极端离群值，而 SWD 能提供强健的约束力。

3.1.3 两种损失函数的激活值对比



3.2 当前实验结果

当前我对LLama-2-7b进行了测试，发现两种损失函数在WikiTex2数据集上的perplexity **一模一样**（未量化模型: 5.4947, Whip: 5.8359, SWD: 5.8317）。尽管理论上SWD可能是更好的选择，但是这可能说明了以下两种情况：

1. 尽管双峰分布不好，但是由于中间峰值距离对比整体的variance太小，所以可以近似成单峰分布，所以两者表现相似
2. 双峰还是单峰对模型影响不大，重要的是要处理Outliers,只要Outliers都处理到位了那模型表现都会大差不大。

还有另外一个发现，在理论来说RLHF之后的模型应该会引入一个比较强的Bias，导致模型可能会出现异常值。但实际经过我调查后发现至少在Llama, Qwen两个系列的模型中差别不大，在Wikitex上没有发现异常激活值。

3.3 下阶段目标

1. 对Whip和SWD loss在同一量化范式内选择更多公司的，更多系列的，更多大小的模型。分组在自己的电脑（或者租服务器）进行量化，最后进行整合得到结论看是SWD好还是Whip好。

- **注意**：由于每个公司，每个系列的模型架构都有可能会不一样。例如在Deepseek模型出来前大家都没用MoE架构，经典的模型没有用RoPE等等。所以一定要根据2.4的指导下根据特定模型的架构进行操作。
- 如果操作错了怎么办？我怎么发现我的错误?
  - 如果操作错了那Perplexity肯定会很高，因为你本质上就是没有用正交的特性抵消掉你添加的旋转矩阵。所以我的建议是先在同一系列的小模型下进行操作，例如Llama 3.2版本下先跑1b的模型。然后看看Perplexity对比未量化的基线模型是不是出齐的高，同时可以对比DartQuant的原始论文，看看Whip下的Perplexity是不是和原文的相差极大（但是原始文章有很多模型都没测试）。
- 如果没有卡的同学请找我，或者可以自己在Autodl上租服务器，大概一个小时就算是4090也就不到2块。

2. 对比RLHF和未RLHF的模型
3. 在更多的数据集上进行测试