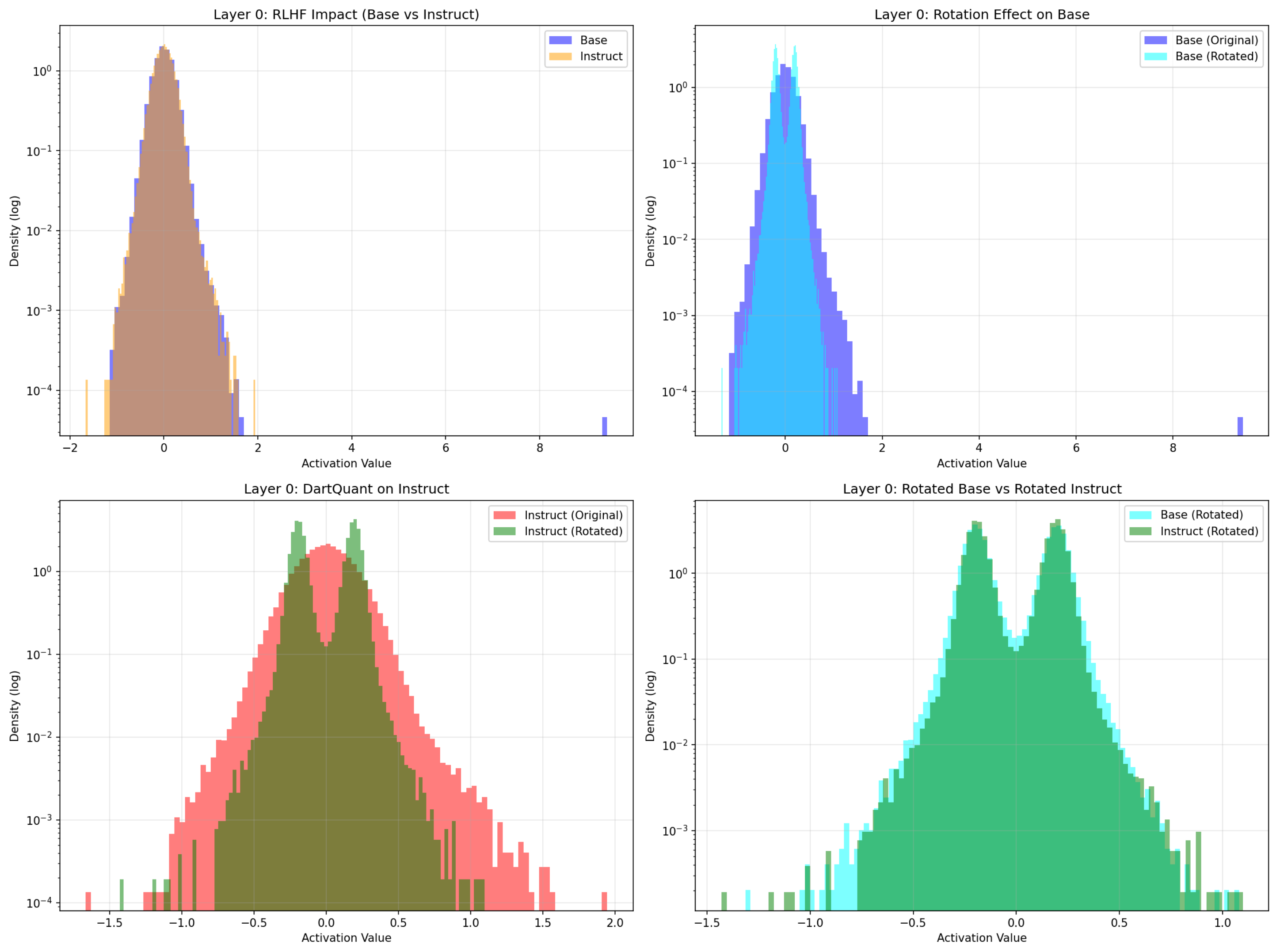
Chapter 1: Post-Training Quantization

Full-precision (FP16, FP32) large language models (LLMs) impose substantial demands on storage, memory, and GPU VRAM, rendering deployment on resource-constrained devices impractical. To address this challenge, *Post-Training Quantization (PTQ)* has emerged as an attractive solution. Unlike *Quantization-Aware Training (QAT)*, PTQ enables direct model compression without retraining (though calibration remains necessary), offering strong practical utility.

However, PTQ faces a fundamental challenge in practice: **activation outliers**. For any linear layer output Y in an LLM, we have:

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

where X denotes the activation matrix and W represents the fixed weight matrix. The figure below illustrates the activation distributions of Llama-3.2-1B-Base and Instruct variants at the first layer.



In the upper-left subplot, we observe a distinct outlier (~ 8) in the Base model's activations. **During quantization, accommodating both normal values and these extreme outliers forces the quantization range to expand significantly. Consequently, the majority of values concentrated in the central region are mapped to only a few quantization levels, resulting in severe precision degradation.** Two principal approaches exist for handling these anomalous activations:

1. Algorithmic handling of outliers during PTQ:
    - Examples in 4-bit quantization include GPTQ, AWQ, SmoothQuant, etc.
2. Pre-processing the activation matrix to address outliers prior to PTQ:
    - DartQuant, SpinQuant, etc.

Chapter 2: DartQuant

The current state-of-the-art (late 2025 onwards) favours the second approach, specifically through the introduction of a rotation matrix. The underlying principle involves applying an orthogonal matrix with unit norm to smooth outliers without altering model outputs or incurring additional inference costs:

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

Our objective is therefore to *find a matrix R such that the rotated activation matrix XR achieves optimal performance on a given dataset after quantization*.

Methods such as SpinQuant and OSTQuant typically employ "end-to-end fine-tuning" to identify the optimal rotation matrix—treating this matrix as a learnable parameter and optimising via gradient descent on a given dataset. However, this end-to-end fine-tuning approach suffers from the following limitations, conflicting with PTQ's goal of rapid deployment:

1. **High computational cost**: According to the SpinQuant paper, optimising a 70B parameter model requires hundreds of GiB of VRAM and tens of GPU-hours (necessitating high-end GPUs such as the A100).
2. **Overfitting risk**: Fine-tuning on a specific dataset can lead to overfitting to particular distributional characteristics.
3. **Optimisation difficulty**: Related to point 1, but specifically concerning R—maintaining orthogonality constraints requires specialised optimisers with prohibitively high time complexity.

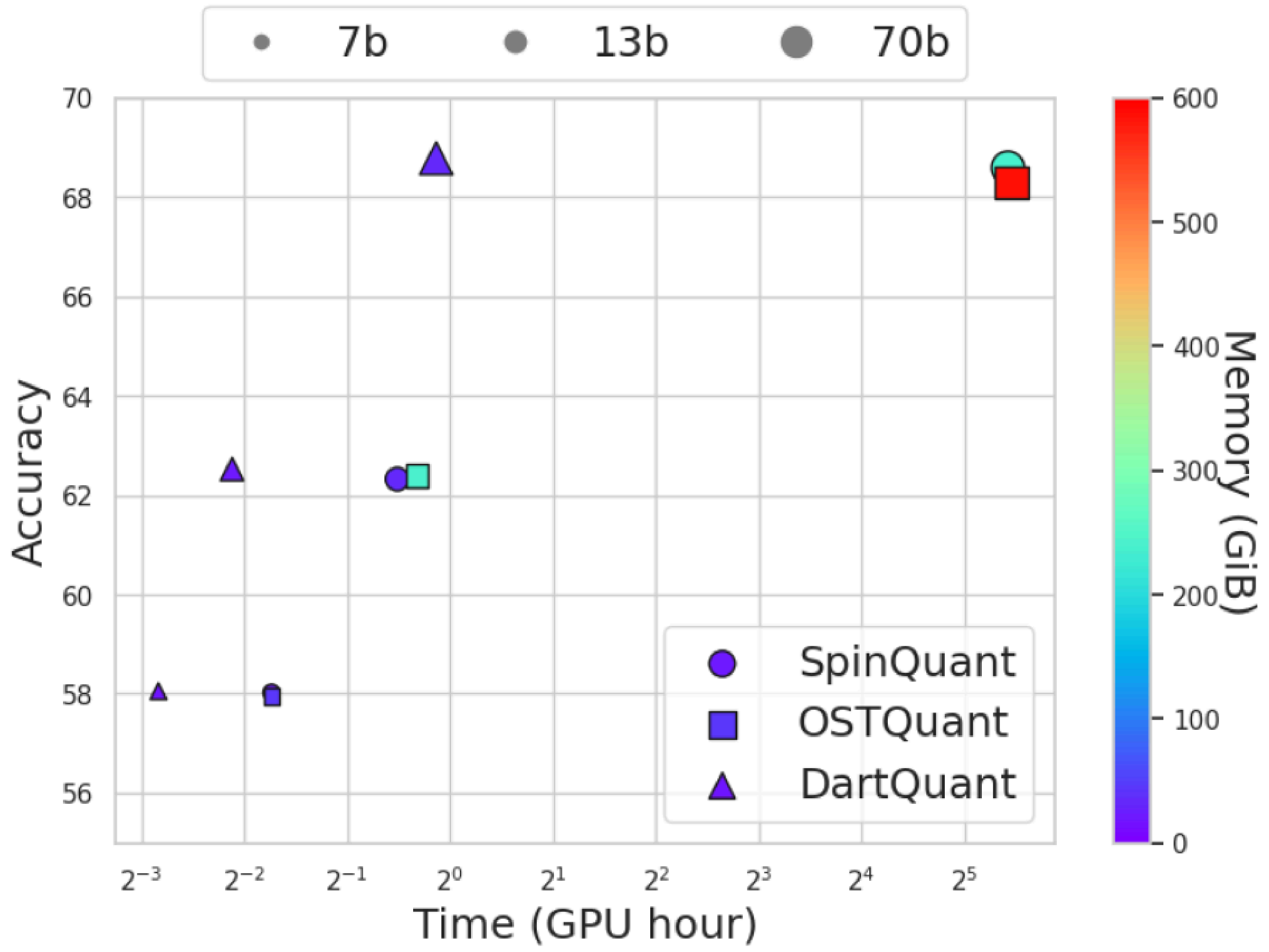


Figure 1: Comparison of computational costs across different rotation optimization methods.

2.1 DartQuant Innovations

To address these issues, DartQuant reformulates the objective. The new goal is: **find a matrix R such that the rotated activation matrix XR approximates a uniform distribution, thereby reducing the number of outliers**. In other words, DartQuant abandons task-specific fine-tuning in favour of a more direct objective: **find an optimal rotation matrix that transforms the activation distribution into a form better suited for quantization (uniform)**.

Question: Why target a uniform distribution?

In 4-bit quantization, only 16 discrete integer grid points are available. Information theory dictates that entropy is maximised when all quantization levels are utilised with equal probability.

To realise this objective, DartQuant introduces two technical innovations:

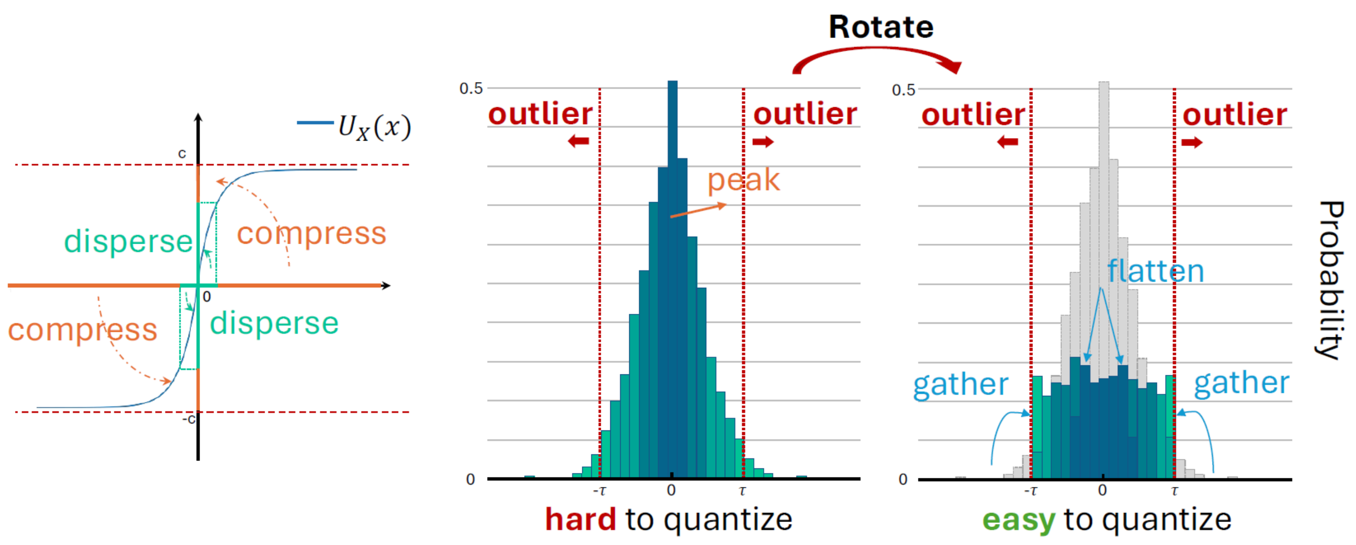
1. Whip loss function
2. QR-Orth optimiser

2.2 Whip Loss Function

The Whip loss function derives from observations about raw activation distributions. These typically exhibit a *Laplacian* form—concentrated at the centre with heavy tails. This is particularly detrimental to quantization, especially for smaller models ($\sim$1B–2B parameters). The Whip loss is based on the Cumulative Distribution Function (CDF) and transforms Laplacian distributions into uniform distributions.

Question: How can we be certain that all activations across all models follow a zero-mean Laplacian distribution?

1. The ReLU/SwiGLU activations and residual connections commonly used in LLMs accumulate to produce peaked distributions.
2. Experimental validation in the original paper's appendix (and my own experiments on Llama, Qwen) confirms that activations across all linear layers in all models exhibit peaked distributions with outliers. Only a few exceptional layers have non-zero means (e.g., one layer in Llama 3.2 1B had a mean of 0.2, but this minor deviation is not problematic).

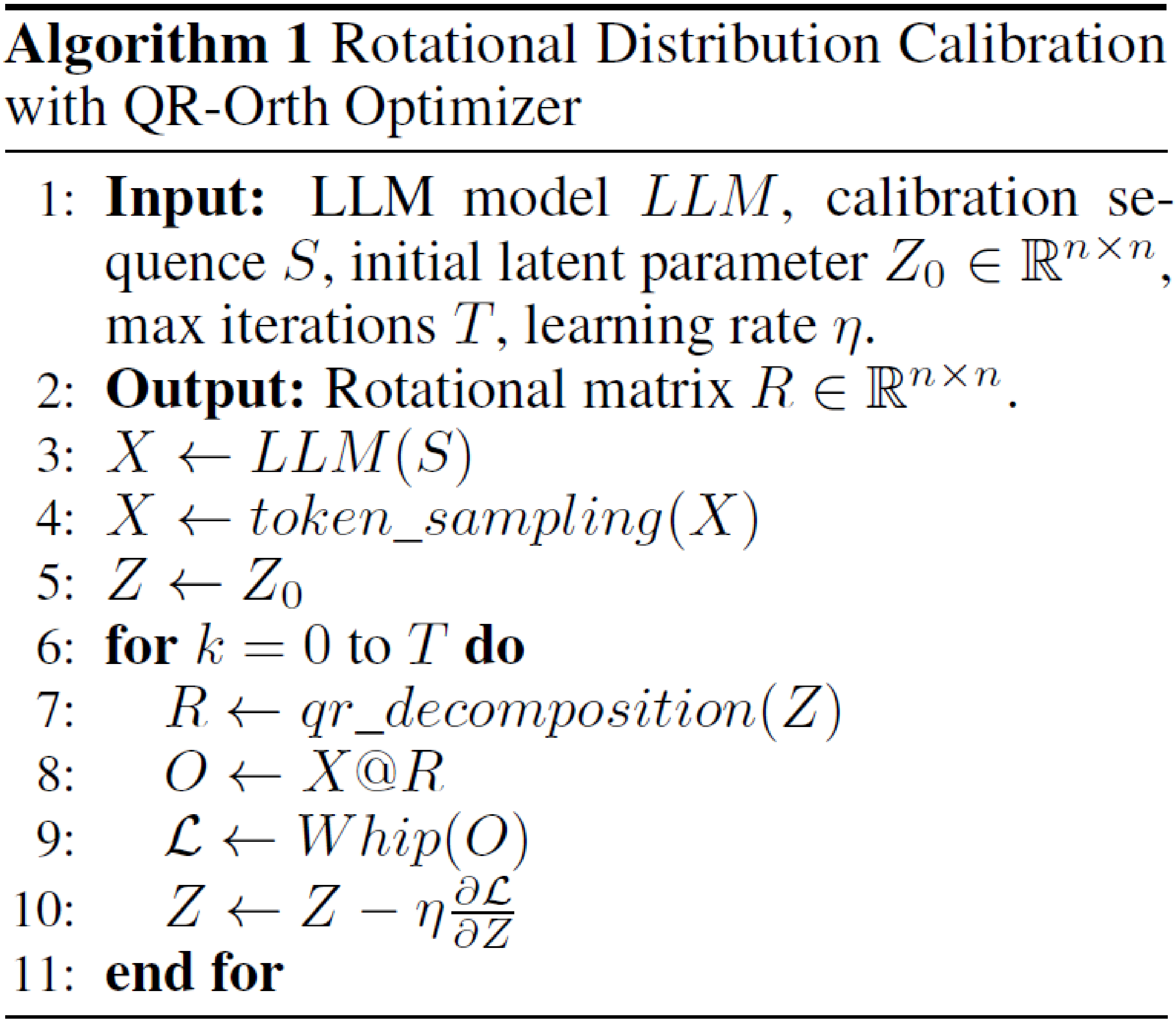


$$\mathcal{L}_{\text{whip}} := \sum_{i=1}^{c_{\text{in}}} \exp(-|x_i|)$$

where $\mathbf{x} = [x_1, \ldots, x_{c_{\text{in}}}] \in \mathbb{R}^{c_{\text{in}}}$ is the activation vector. For the detailed derivation, refer to the DartQuant paper.

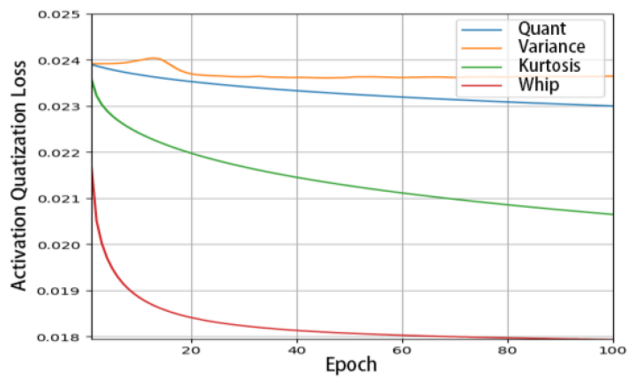
2.3 QR-Orth

Traditional approaches using orthogonal optimisers (e.g., Cayley SGD) to train R directly incur prohibitive time complexity. QR-Orth instead defines a latent variable matrix (an ordinary, unconstrained, non-orthogonal matrix), then extracts an orthogonal, norm-preserving rotation matrix via QR decomposition. The decomposed orthogonal matrix is passed to the Whip loss, and gradients are backpropagated to the latent variable matrix via standard SGD.

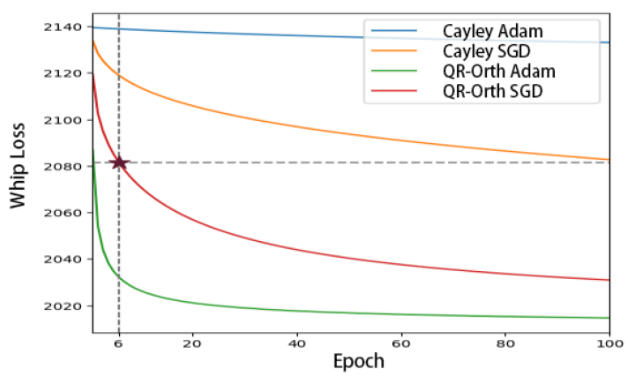


Algorithm 1 Rotational Distribution Calibration with QR-Orth Optimizer

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

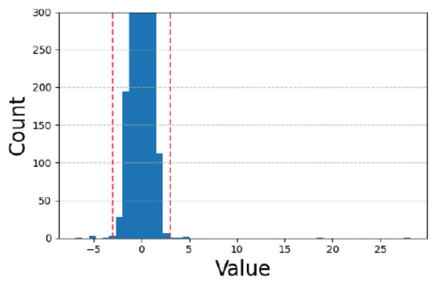


(a) Loss comparison.

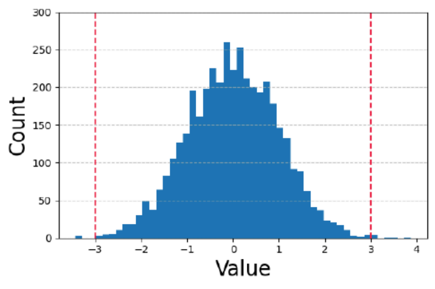


(b) Cayley vs QR-Orth.

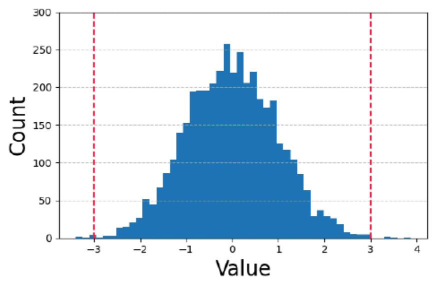
Figure 7: Comparison of activation quantization loss and convergence performance using different optimization methods.



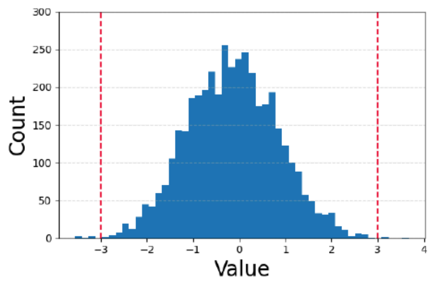
(a) Original.



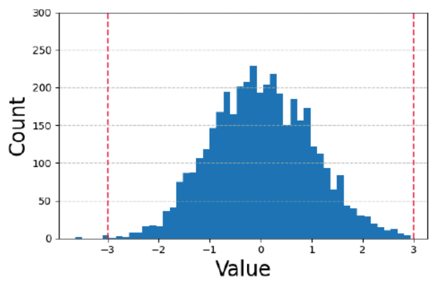
(b) Hadamard.



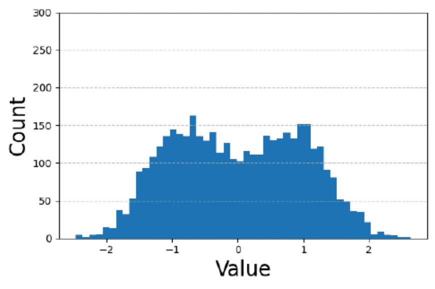
(c) Quant.



(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 Application of Rotation Matrices to Different Transformer Components

Modern LLM Transformer architectures comprise:

1. General linear layers
2. Attention layers
3. Feed-forward layers
4. MoE routing components

In the DartQuant framework, we define four orthogonal matrices designed to rotate the activation space and eliminate outliers whilst preserving computational invariance (i.e., unchanged full-precision mathematical outputs).

Let d_{model} denote the hidden dimension, d_{head} the attention head dimension, and d_{inter} the FFN intermediate dimension.

- $R_1 \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$ **(Residual Rotation)**: A **learnable** orthogonal matrix optimised by DartQuant using QR-Orth. It acts on the Transformer block's input activations X and propagates through the entire residual stream.
- $R_2 \in \mathbb{R}^{d_{\text{head}} \times d_{\text{head}}}$ **(Value-Head Rotation)**: A **learnable** orthogonal matrix applied specifically within the self-attention mechanism to the Value heads. Since attention heads operate in the subspace d_{head}, an independent rotation matrix R_2 is required.
- $R_3 \in \mathbb{R}^{d_{\text{head}} \times d_{\text{head}}}$ **(Key/Query Rotation)**: A **fixed** random Hadamard matrix that is not trained. It is applied online after RoPE to smooth Q and K distributions, particularly beneficial for KV-cache quantization.
- $R_4 \in \mathbb{R}^{d_{\text{inter}} \times d_{\text{inter}}}$ **(FFN Activation Rotation)**: A **fixed** random Hadamard matrix applied online to the intermediate states after the non-linear activation in FFN (or MoE experts), smoothing the input to the down-projection layer.

2.4.1 A. Global Residual Stream (R_1)

For any linear layer $Y = XW^\top$, if the input X is rotated to $\tilde{X} = XR_1$, the weight matrix W must absorb the inverse rotation $R_1^\top$:

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

Fusion strategy: For input-side weights ($W_q, W_k, W_v, W_{\text{up}}, W_{\text{gate}}$), R_1 is right-multiplied into the weight matrix:

$$\tilde{W} = WR_1$$

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

2.4.2 B. RoPE and Attention Mechanism (R_2, R_3)

1. **Input Projection** (W_q, W_k, W_v): All input weights first absorb the residual stream rotation 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 and R_3 (Online Computation)**: Rotary Position Embedding (RoPE) is rotation-sensitive, precluding direct fusion (as RoPE itself employs rotation matrices; see the original paper for details). R_3 is designed for online application after RoPE to smooth Q and K. Let $\mathcal{P}(\cdot)$ denote the RoPE function:

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

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

Since R_3 is orthogonal, it cancels in the dot-product attention computation:

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

where $Q' = \mathcal{P}(\tilde{X}\tilde{W}_q^\top)$, and similarly for K'.

3. **Value Head and R_2 (Fusion)**: To smooth the Value head output, R_2 is fused into W_v's output side:

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

(Note: The input side absorbs R_1; the output side produces activations rotated by R_2.)

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

Let X_{val} denote the intermediate output vector after multi-head attention concatenation, and W_{out} the output linear layer weight mapping back to residual stream dimensions. The input to W_{out} (i.e., X_{val}) now resides in the R_2 basis (from W_v output), whilst its output must return to the R_1 basis of the residual stream. Therefore, W_{out} must cancel R_2 at its input and introduce R_1 at its output:

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

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

The final output is rotated by R_1, consistent with the residual stream.

2.4.3 C. FFN and Gating Mechanism (R_4)

FFN (or MoE expert) structures typically contain non-linear activations (e.g., SwiGLU), which break the linear rotation fusion property. Hence, online rotation R_4 must be introduced.

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

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

Up/Gate Projections: Weights absorb the residual stream's R_1:

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

Activation and R_4 (Online Computation): Let σ denote the non-linear activation function (e.g., SiLU). After computing the intermediate state H, explicitly multiply by R_4 to smooth the distribution:

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

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

($\tilde{X}$ is R_1-rotated.)

Down Projection (W_{down}): The input to W_{down} now resides in the R_4 basis, whilst the output must return to the R_1 basis:

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

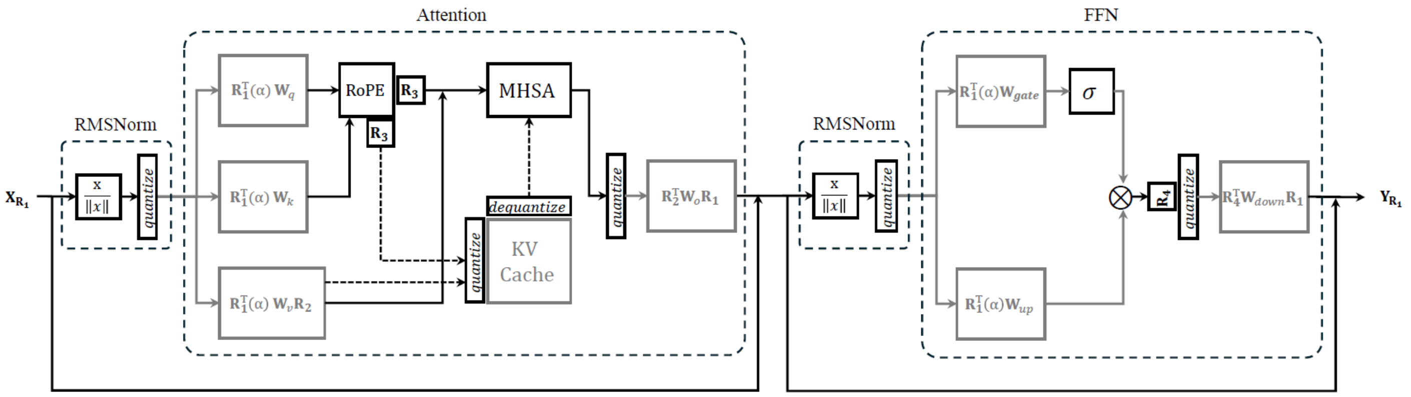
(Note: Although R_4 is a Hadamard matrix (typically symmetric), mathematically it should be R_4 to cancel the input rotation.)

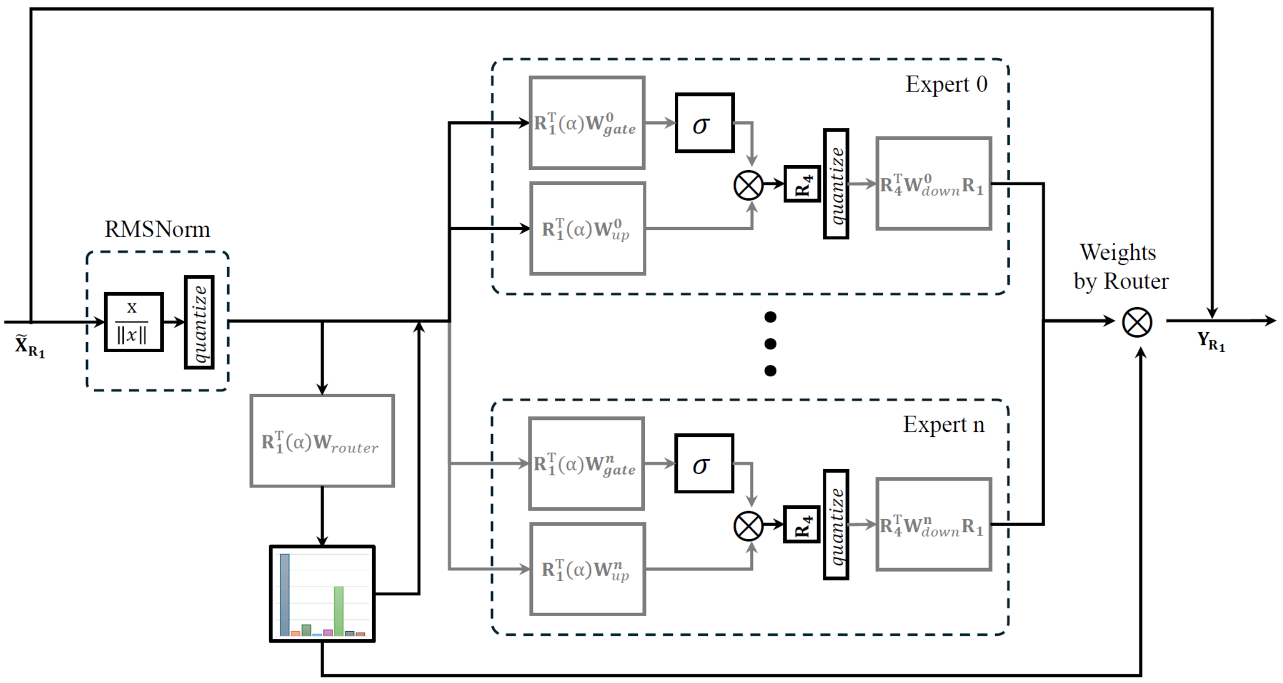
2.4.4 D. Mixture of Experts (MoE)

In MoE models (e.g., Mixtral), rotation matrix application follows similar logic to FFN but operates within each expert:

- **Router**: Receives the global residual stream input (already R_1-rotated) and computes routing weights in the R_1 basis.
- **Inside Experts**: Each expert's $W^i_{\text{gate}}, W^i_{\text{up}}$ absorbs R_1. **Online** R_4: Applied after the activation function within each expert. Typically, all experts share the same R_4 to reduce overhead. Each expert's W^i_{down} absorbs R_4 (input side) and $R_1^\top$ (output side).

2.4.5 Complete Architecture Diagrams (Classic Transformer and MoE Transformer)





Chapter 3: Proposed Innovation

3.1 Sliced Wasserstein Distance (SWD)

3.1.1 SWD Derivation

Let $X \in \mathbb{R}^{B \times D}$ be the original input activation matrix (where B is the batch size and D is the feature dimension). We introduce an orthogonal rotation matrix $R \in \mathbb{R}^{D \times D}$ (i.e., $R^\top R = I$), yielding rotated activations $\tilde{X} = XR$.

Our objective is to optimise R such that the empirical distribution $P_{\tilde{X}}$ of $\tilde{X}$ closely approximates a predefined target distribution Q. To quantify the discrepancy between two distributions, we employ the 1-Dimensional 2-Wasserstein Distance (the core component of Sliced Wasserstein Distance).

We flatten all rotated activation values into a vector $\mathbf{x} \in \mathbb{R}^N$ (where $N = B \times D$) as input to the loss function.

For two one-dimensional probability distributions P and Q with cumulative distribution functions (CDFs) F_P, F_Q and quantile functions (i.e., F^{-1}) F_P^{-1}, F_Q^{-1} respectively, the 1D W_2 distance admits a closed-form solution:

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

For discrete observations, the analytical form of $F_{P_Y}^{-1}$ is unavailable; we must estimate it using the empirical quantile function.

Definition 1 (Order Statistics). Let $\mathbf{y} = \{y_1, \ldots, y_N\}$ be the set of rotated activation values. Define permutation σ such that $\mathbf{y}$ is sorted in non-decreasing order. Denote the sorted vector as $\mathbf{y}_{\text{sorted}}$, with the i-th element being the order statistic:

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

Approximation 1 (Empirical Quantile). The quantile function $F_{P_Y}^{-1}(u)$ of the empirical distribution is a piecewise constant function. At discrete sample points $u_i = \frac{i-0.5}{N}$ (or approximately $\frac{i}{N}$), its value equals the order statistic:

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

Approximation 2 (Target Quantile): For target distribution $Q = \mathrm{Uniform}[-b, b]$, the quantile function is linear: $F_Q^{-1}(u) = 2b \cdot u - b$. At discrete points u_i, the target value is:

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

Lemma 1 (Calculation of b): If the target distribution is $\mathrm{Uniform}[-b, b]$, then the boundary is $b = \sqrt{3} \cdot \mathrm{RMS} = \sqrt{3}\sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}$.

Proof: For the rotated activations to map well to $\mathrm{Uniform}[-b, b]$, the target distribution's second moment must equal the input's second moment. Otherwise, regardless of how R rotates the original activations, perfect matching is unattainable.

- **Input second moment**: Let the RMS of input data x be RMS; then its second moment is $\mathbb{E}[x^2] = \mathrm{RMS}^2$.
- **Target uniform distribution $U[-b, b]$ second moment**: Let random variable $T \sim U[-b, b]$ with PDF $f(t) = \frac{1}{2b}$. Its second moment is:

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

Thus:

$$\mathrm{RMS}^2 = \frac{b^2}{3} \implies b = \sqrt{3} \cdot \mathrm{RMS} \quad \square$$

Discretising the continuous integral $\int_0^1$ as a summation, we convert the W_2 distance into an MSE-form loss function:

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

3.1.2 Gradient Analysis

Gradient Derivation:

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

where t_r is the fixed target quantile value.

Case Study: Extreme Outlier

Consider an extremely large positive outlier $y_k \to +\infty$.

- **Ranking**: Since y_k is extremely large, it is necessarily sorted last, i.e., rank $r \approx N$.
- **Target**: The corresponding target value is the uniform distribution's upper bound $t_N \approx b$.

Gradient Magnitude:

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

Conclusion: The gradient scales linearly with outlier magnitude. The more extreme the outlier, the stronger the gradient force pulling it back to the target interval $[-b, b]$.

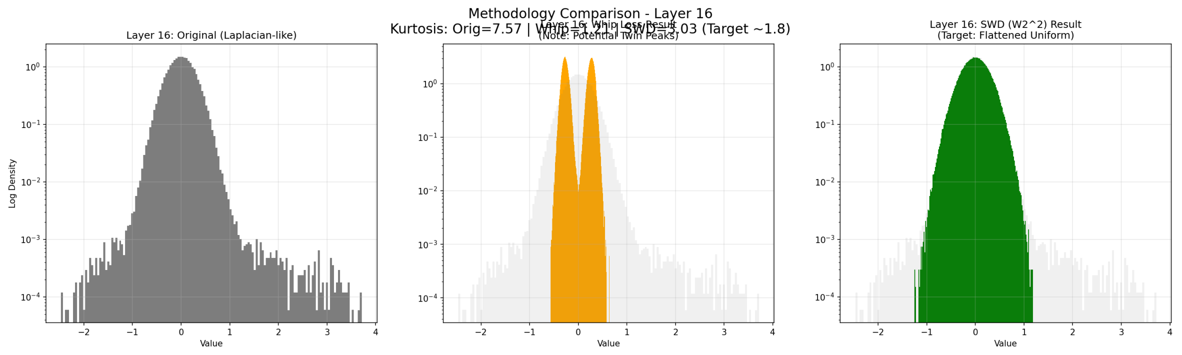
Comparison with Whip Loss

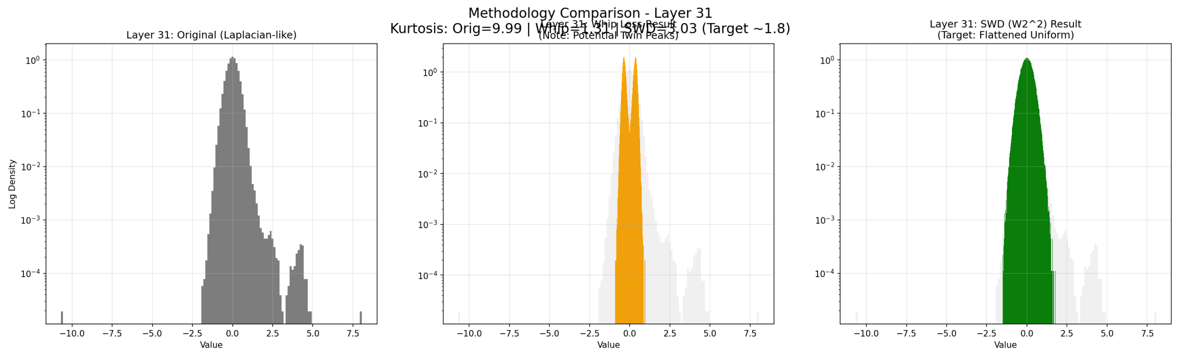
Recall the Whip Loss $\mathcal{L}_{\text{Whip}} = \sum e^{-|y_k|}$ gradient:

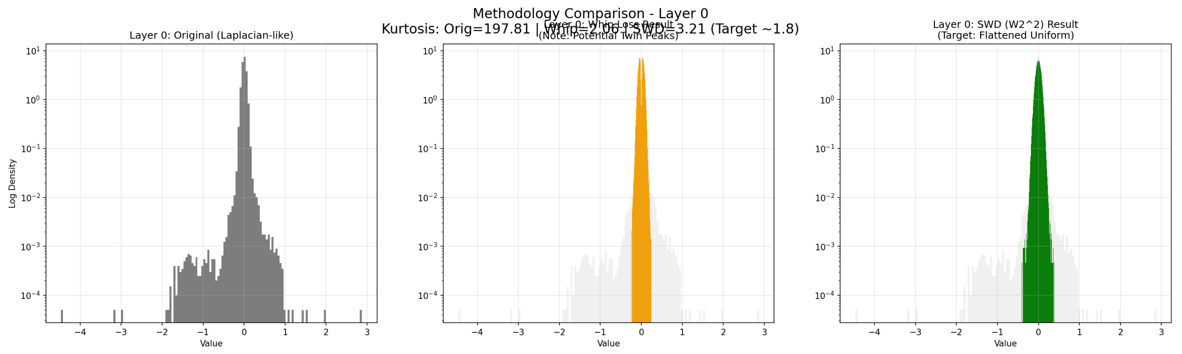
$$\left|\frac{\partial \mathcal{L}_{\text{Whip}}}{\partial y_k}\right| = e^{-|y_k|}$$

As $y_k \to +\infty$, the gradient vanishes. This mathematically explains why Whip Loss struggles with extreme outliers, whereas SWD provides robust constraint forces.

3.1.3 Activation Distribution Comparison Between Loss Functions







3.2 Current Experimental Results

I tested both loss functions on LLaMA-2-7B and found that perplexity on WikiText-2 is **virtually identical** (unquantized model: 5.4947, Whip: 5.8359, SWD: 5.8317). Although SWD may theoretically be superior, this suggests two possibilities:

1. Despite the bimodal distribution being suboptimal, the inter-peak distance is negligible relative to the overall variance, approximating a unimodal distribution—hence similar performance.
2. Modality (bimodal vs. unimodal) has minimal impact on model performance; what matters is outlier handling. Once outliers are adequately addressed, performance is comparable.

Another finding: theoretically, RLHF-tuned models should introduce substantial bias, potentially producing anomalous activations. However, my investigation reveals that at least for Llama and Qwen model series, differences are minimal, with no anomalous activations observed on WikiText-2.

3.3 Next-Stage Objectives

1. Compare Whip and SWD losses within the same quantization paradigm across multiple vendors, model series, and sizes. Conduct grouped quantization experiments locally (or on rented servers) and aggregate results to determine whether SWD or Whip is superior.
    - **Note**: Different vendors and model series may have distinct architectures. For instance, MoE architectures (as in DeepSeek) were not widespread before DeepSeek's release, and classic models did not use RoPE, etc. Follow the guidance in Section 2.4 according to specific model architectures.
    - **How to detect errors?** If implemented incorrectly, perplexity will be substantially elevated since the orthogonal property fails to cancel the added rotation matrices. I recommend first testing on smaller models within the same series (e.g., Llama 3.2 1B), checking whether perplexity is anomalously high compared to the unquantized baseline, and cross-referencing with DartQuant's original paper to verify Whip perplexity is not drastically different (though the original paper omits many models).
    - Students without GPU access should contact me or rent servers on AutoDL (approximately ¥2/hour even for 4090s).
2. Compare RLHF-tuned and non-RLHF models.
3. Evaluate on additional benchmark datasets.

3.4 Next Stage split of work:

Each one of us will be allocated a model and write the code for quantization, at the final stage we will rent a cloud GPU to run the entire quantization and evaluation process.