QUANTIZATION-AWARE PRE-TRAINING WITH CONSTRAINED EMPIRICAL WEIGHT DISTRIBUTION

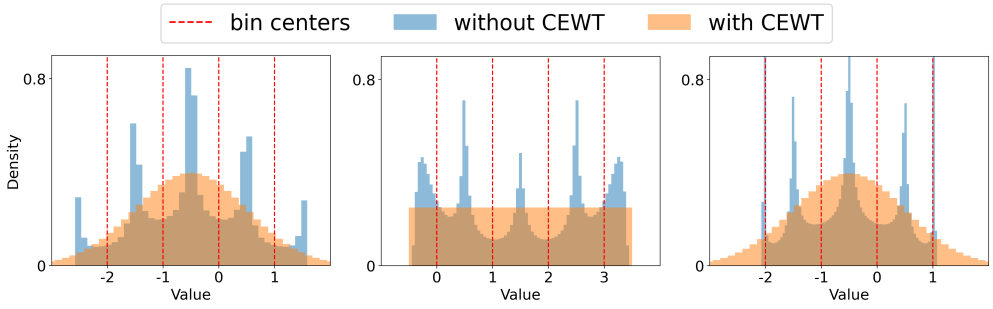
Anonymous authors
Paper under double-blind review

ABSTRACT

Quantization-Aware Pre-Training (QAPT) can increase the inference efficiency of DNNs, but a problematic behaviour known as rounding boundary weight oscillation can introduce detrimental noise into the training process and significantly reduce convergence speed. While existing methods can reduce this detrimental noise, they either introduce additional hyperparameters or memory overhead, or cannot consistently improve model accuracy. In this work, we propose optimization with **C**onstrained **E**mpirical **W**eight dis**T**ribution (CEWT), the first hyperparameter-free memory-overhead-free oscillation suppression method that consistently improves QAPT performance: an optimizer post-update step that projects weights to the nearest point in weight space whose empirical distribution (histogram) matches a zero-mean Gaussian. Our key insight is many quantizers are designed with the implicit assumption that the to-be-quantized data are permutations of samples from a zero-mean Gaussian, and this assumption is not true during QAPT. By enforcing the zero-mean Gaussian prior as a hard constraint, CEWT can suppress this detrimental noise. Empirical results on various combinations of SOTA quantizers and hypersphere optimizers suggest, that with a geomean increase of 4% in training time, CEWT can consistently reduce the pre-training perplexity (by an average of 2.5 and up to 21 points) of low-precision (down to 1-bit activations and weights and up to 610M parameters) LLaMA/GPT models without introducing any hyperparameters or storage overhead.

1 INTRODUCTION

Quantization can reduce the inference complexity of Deep Neural Networks (DNNs). Compared to full-precision DNNs, quantized DNNs consume lower memory capacity and communication bandwidth in data centers, while simultaneously achieving higher inference throughput and energy efficiency when deployed to edge devices. However, naively quantizing DNNs often degrades their quality. While Post-Training Quantization (PTQ) methods (Liu et al., 2025a; Lin et al., 2024; Shao



(a) QuEST (Panferov et al., 2025) (b) BBQ (Yang & Aamodt, 2026) (c) LSQ (Esser et al., 2020)

Figure 1: Effects of CEWT on pre-round (BBQ) and pre-clip (QuEST/LSQ) weight distributions of a LLaMA-95M with 2-bit weights and activations pre-trained on 3 billion C4 tokens. See Algorithms 8, 9, and 10 from definitions of pre-round and pre-clip weights.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

et al., 2024; Ma et al., 2024b; Liu et al., 2024a; Kim et al., 2024) can mitigate quality degradation at higher precisions without re-training, PTQ methods generally result in a large accuracy gap when both activations and weights are quantized to 4-bit and below (Panferov et al., 2025; Kumar et al., 2025; Ma et al., 2024a; Esser et al., 2020). In contrast, Quantization-Aware Training (QAT) methods (Liu et al., 2025b; Panferov et al., 2025; Ma et al., 2024a; Kumar et al., 2025; Shen et al., 2024; Yamamoto, 2021; Esser et al., 2020) can narrow this accuracy gap (Du et al., 2024; Panferov et al., 2025; Liu et al., 2024b) by introducing quantization into the optimization process. QAT can be further categorized into Quantization-Aware Pre-Training (Panferov et al., 2025; Yang & Aamodt, 2025) (QAPT) and Quantization-Aware Fine-Tuning (Malinovskii et al., 2024; Du et al., 2024) (QAFT). QAFT starts from a full-precision pre-trained checkpoint and briefly trains a low-precision model on a comparatively small (Du et al., 2024) downstream dataset. In contrast, QAPT trains a low-precision model from scratch on a much larger dataset and for substantially longer. Compared with full-precision pre-training followed by PTQ or QAFT, QAPT may accelerate pre-training (Kumar et al., 2025; Xi et al., 2023; Castro et al., 2025) by performing the forward pass in low precision. This work focuses on improving the accuracy of QAPT when both weights and activations are quantized to 2-bits and below.

In QAT, a problem known as rounding boundary weight oscillation (Nagel et al., 2022; Liu et al., 2023; Wenshøj et al., 2025), often attributed (Nagel et al., 2022) to the gradient bias of the Straight-Through Estimator (Bengio et al., 2013) and other rounding Jacobian estimators, can introduce detrimental noise into the optimization process and slow down convergence, especially at the later stages of training. A significant portion of parameters gather at the rounding boundaries of the quantization grid, and constantly oscillate from one side of the rounding boundaries to the other. Existing methods attempt to address the slow convergence problem by freezing oscillating parameters (Nagel et al., 2022), by reducing the number of parameters sufficiently close to rounding boundaries (Nagel et al., 2022; Liu et al., 2023; Li et al., 2026), or by simulating deterministic quantization error with random noise (Shin et al., 2023). However, these methods either introduce additional hyperparameters, which when under-tuned can reduce the effectiveness, while proper tuning can increase training budget, or they cannot consistently improve the performance of QAPT due to the use of biased optimization objectives.

Inspired by Hypersphere Optimization (Wen et al., 2026; Xie et al., 2026; Ren et al., 2026), which removes weight decay as a hyperparameter by enforcing what weight decay *does* (i.e. controlling the Frobenius norm of weights) as a hard constraint, we propose optimization with **C**onstrained **E**mpirical **W**eight Dis**T**ribution (CEWT, pronounced *cute*), a hyperparameter-free extension to existing Hypersphere optimizers (e.g. AdamH (Kingma & Ba, 2017; Wen et al., 2026), MuonH (Jordan, 2024; Wen et al., 2026), LionH (Chen et al., 2023; Wen et al., 2026), and SSO (Xie et al., 2026)) that projects weights to the nearest point in weight space whose empirical distribution (histogram) matches a Gaussian. CEWT is inspired by our observation that many quantizers (Yang & Aamodt, 2026; Panferov et al., 2025; Dettmers et al., 2023; Tu et al., 2022) are designed based on the assumption that the empirical distribution of raw weights are, or can be approximately converted to, a bell-shaped distribution approximately centered around zero — an assumption that is *not* enforced, and is generally *false*, in practice. To address this gap between assumption and reality, CEWT enforces the distribution shape to match a zero-mean Gaussian as a hard constraint. CEWT can suppress rounding boundary weight oscillation as it regulates, *by design*, the number of parameters near rounding boundaries and can significantly accelerate convergence without introducing any hyperparameters.

2 Background and Motivation

In this section, we discuss the rounding boundary weight oscillation problem and existing solutions, and Hypersphere Optimization which removes the weight decay hyperparameter by adding a hard constraint to the optimization problem. As we will discuss later in Section 3, CEWT also removes a hyperparameter by adding a hard constraint.

2.1 Rounding boundary Weight Oscillation

Rounding boundary weight oscillation (Nagel et al., 2022) is the phenomenon that during QAT, a significant portion of parameters gather at a rounding boundary, and constantly move from one side

of the rounding boundary to the other, generating significant noise which prevents convergence. Weight oscillation manifests as irregularly high density spikes near rounding boundaries (Nagel et al., 2022; Liu et al., 2023), and can be detected by plotting the histogram of pre-rounding weights. Unless explicitly addressed, weight oscillation happens regardless of the quantizer used. For example, Figure 1 shows the pre-rounding weight distribution of three SOTA quantizers and their density spikes near rounding boundaries. Weight oscillation tends to happen when a parameter is close to a locally optimal value (Nagel et al., 2022). Weight oscillation happens regardless (Nagel et al., 2022) of the rounding Jacobian estimation method[1]. Section 4.2 empirically demonstrates rounding boundary weight oscillation cannot be effectively suppressed with the use of optimizers equipped with momentum (e.g. MuonH/LionH) and/or preconditioning (e.g. AdamH/MuonH). In DNN pre-training, where parameters are typically interdependent[2], a subset of oscillating parameters can introduce noise into the gradient of other non-oscillating parameters. The gradient noise becomes more severe as precision (number of bits) reduces and the distance between two quantization bins increases. Section A shows two simple example problems that demonstrate the root cause of weight oscillation.

Existing methods are effective in oscillation suppression but may degrade model accuracy and introduce additional hyperparameters and memory overhead. Nagel et al. (2022) proposes to detect oscillating weights and freeze them until the end of training. Weight freezing requires additional storage to record the history of weights for oscillation detection and an oscillating frequency threshold hyperparameter which when under-tuned, as Nagel et al. (2022) noted, can freeze too many parameters too early, degrading the model's final accuracy. Nagel et al. (2022) also proposed an oscillation dampening loss to regularize the squared distance between quantized weights and clipped weights, effectively pulling unclipped parameters away from rounding boundaries, and can therefore prevent oscillation. However, Nagel et al. (2022) noted that overly strong regularization can inhibit beneficial weight movement between quantization bins and harm final accuracy. Liu et al. (2023) proposes Confidence-Guided Annealing (CGA), which trains only parameters sufficiently close to rounding boundaries while freezing parameters far from boundaries. CGA requires the schedule of a hyperparameter that defines "sufficiently close" to rounding boundaries. Liu et al. (2023) stated that when the threshold hyperparameter is overly small, too many parameter are frozen, inhibiting beneficial weight movement which leads to lower final accuracy. Li et al. (2026) proposed to periodically re-initialize weights to a linear interpolation between weights and quantized weights and showed this interpolation is equivalent to regularizing the squared distance between quantized and raw weights. While not originally proposed to suppress weight oscillation, empirically we show Periodic Interpolation also suppresses oscillation (Section 4.1) and we hypothesize this is due to the theoretical connection to dampening loss—both periodic interpolation and dampening loss explicitly pulls weights away from rounding boundaries. Periodic interpolation requires an interpolation ratio hyperparameter. Li et al. (2026) noted that too large or too low interpolation ratio can lead to suboptimal performance. Pseudo-Quantization Noise (PQN) (Shin et al., 2023) simulates the behaviour of rounding with addition of noise from $U(-0.5, 0.5)$, effectively replacing the original optimization objective (e.g. $L(s \cdot \lfloor w/s \rceil), w \in \mathbb{R}^d, s \in \mathbb{R}$) with a **biased** objective (e.g. $E_{\xi_i \sim U(-0.5,0.5),\forall i}[L(s \cdot (w/s + \xi))], w \in \mathbb{R}^d, s \in \mathbb{R}, \xi \in \mathbb{R}^d$). Sections 4.1 and I empirically show that PQN cannot consistently outperform baseline QAT methods like the STE, and we hypothesize this is due to the biasedness of the optimization objective. Sections 4.1 and D.3 empirically and theoretically demonstrate this biasedness for a toy problem. Sections 4.1 and I show that CEWT consistently outperforms both PQN and the STE.

Existing methods demonstrates that reducing the rate of weight oscillation leads to significant accuracy improvement, but tuning hyperparameter is non-trivial and a tuned hyperparameter for a model size/architecture might not transfer to another. On the other hand, existing hyperparameter-free methods like PQN cannot consistently outperform the STE. We therefore ask the research question: *can we regulate the rate of rounding boundary weight oscillation and consistently achieve higher-than-STE accuracy without introducing any hyperparameters*? As a preview, CEWT regulates the number of parameters near rounding boundaries, without introducing any hyperparameters, by enforcing the empirical weight distribution to match a zero-mean Gaussian, an assumption embedded

[1]Section D.8 and Table 6 empirically shows that some rounding Jacobian estimators may be less vulnerable, although not immune, to rounding boundary weight oscillation, and that CEWT can suppress rounding boundary weight oscillation and improve accuracy **regardless** of the rounding Jacobian estimator used.

[2]Sections A.2 and D.1 discuss parameter interdependency and its impact on QAT.

in the design of many quantizers (Yang & Aamodt, 2026; Panferov et al., 2025; Dettmers et al., 2023) but is not enforced in practice.

2.2 HYPERSPHERE OPTIMIZATION

Hypersphere Optimization (HO) (Xie et al., 2026; Wen et al., 2026; Ren et al., 2026) is a family of optimizers that *removes the hyperparameter weight decay* by constraining weights to lie on the radius-R Frobenius norm hypersphere (Wen et al., 2026) or the radius-R Spectral norm hypersphere (Xie et al., 2026). HO targets the following problem:

$$\underset{W \in \mathbb{R}^{m \times n}}{\text{min.}} L(W) \text{ s.t. } ||W|| = R \tag{1}$$

where L denotes the loss as a function of the parameters, $W \in \mathbb{R}^{m \times n}$ is the weight matrix of a linear layer[3] with m output features and n input features, $||\cdot||$ is either the Frobenius norm operator $||\cdot||_F$ or the spectral norm operator $||\cdot||_2$, and $R \in \mathbb{R}$ is typically chosen to be $||W^{(0)}||$, W's norm at random initialization time (Wen et al., 2026; Ren et al., 2026). The algorithm of HO is as follows:

$$\bar{W}^{(t)} = W^{(t-1)} - \eta^{(t)} R U^{(t)} \tag{2a}$$

$$\hat{W}^{(t)} = \bar{W}^{(t)} \tag{2b}$$

$$W^{(t)} = \frac{R}{||\hat{W}^{(t)}||} \hat{W}^{(t)} \tag{2c}$$

where $\cdot^{(t)}$ is the value of a variable at iteration t, $U \in \mathbb{R}^{m \times n}$ is a unit-norm (i.e. $||U|| = 1$) update whose definition is optimizer-dependent (e.g. see Algorithms 4, 5, 6, and 7), and $\bar{W}, \hat{W} \in \mathbb{R}^{m \times n}$ are variables introduced for notational convenience. In Equation 2a, HO scales the update U to have norm R and performs the update with a learning rate η, ensuring the ratio $||\Delta W||/||W||$ is exactly η, a key property that enables hyperparameter (learning rate) transfer (Wen et al., 2026; Ren et al., 2026; Yang et al., 2024). Equation 2b is a redundant equation that we intentionally introduced, and we will return to discuss Equation 2b in Section 3.1. In Equation 2c, HO projects the updated weight $\hat{W}$ back to the radius-R hypersphere. Hypersphere optimizers (e.g. AdamH/MuonH/SSO) empirically outperform (Xie et al., 2026; Wen et al., 2026; Ren et al., 2026) non-hypersphere optimizers (e.g. Adam/Muon).

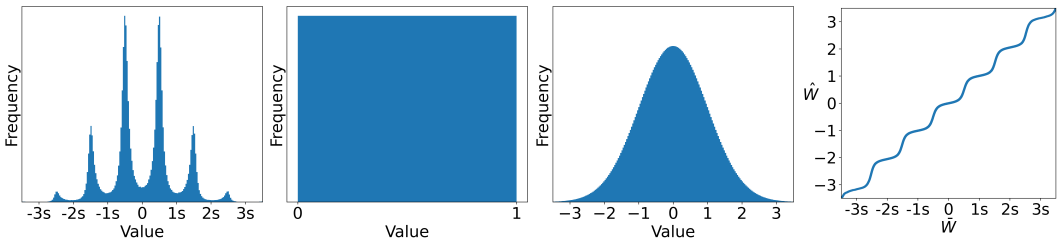
By converting a **soft constraint** (e.g. penalize the Frobenius norm of weights as much as possible) into a **hard constraint** (e.g. the Frobenius norm of weights must be exactly R), HO successfully removes the need to tune hyperparameters, as there exist plenty of theoretically-grounded methods (Yang et al., 2024; Yang & Hu, 2021; Yang et al., 2021) to choose[4] R, whereas choosing the weight decay hyperparameter generally relies on empirical observations and expensive grid searches. Therefore, if it is easier to describe **what a "good" solution looks like**, compared to **what regularization term should be added to the loss such that the model is "good" at the end of training**, hard constraints are an effective tool to remove the less expressive and harder-to-tune soft constraint hyperparameters. Returning to the problem of hyperparameter-free oscillation suppression in QAPT, we then ask the question: *what does the histogram of the weights of a "good" , or oscillation-free, model look like?*

3 PROPOSED METHOD

In this section, we present CEWT. CEWT is designed based on the observation that the weight distribution of round-boundary-oscillation-free models, specifically those trained *without any quantization* (rounding boundary oscillation is a QAT-specific phenomenon), tend to look like a zero-mean Gaussian (Dettmers et al., 2023). Therefore, many quantizer design decisions, such as choosing clipping functions (Yang & Aamodt, 2026), scaling factors (Panferov et al., 2025), and bin centers (Dettmers et al., 2023), are made based on the zero-mean Gaussian assumption. However,

[3]Without loss of generality, we assume there is only one layer in the model and it is a linear one.

[4]Section B discusses that R is **not** introduced by HO, but is inherited from weight initialization methods and **reused** by HO.



(a) Example $\bar{W}$ (b) $\frac{1}{mn} \cdot \left(\text{ranks}(\bar{W}) + \frac{1}{2}\right)$ (c) $\hat{W}$ (d) CEWT's mapping

Figure 2: Visualization of CEWT (Equation 4). Notations like $\text{vec}(\cdot)$ and $\cdot^{(t)}$ are omitted. The example $\bar{W}$ we use here is the weights of the STE experiment in Table 2 at iteration 60. The s in sub-figures 2a and 2d is defined in Section D.

during QAT, the histogram of pre-clip[5] weights do *not* look like zero-mean Gaussians due to weight density spikes that appear at rounding boundaries (see Figure 1 for an example) . To address this gap between assumption and reality, we propose to encode the zero-mean Gaussian prior as a hard constraint (Equation 3d), thereby constraining the entire optimization trajectory in density-spike-free regions of the weight space. CEWT targets the following optimization problem:

$$\min_{W \in \mathbb{R}^{m \times n}} L(Q) \tag{3a}$$

$$\text{s.t. } Q = \textit{weight_quantizer}(W) \tag{3b}$$

$$W = \frac{R}{||\hat{W}||} \hat{W} \tag{3c}$$

$$\text{sort}(\text{vec}(\hat{W}))_i = \Phi^{-1}\left(\frac{i + 0.5}{mn}\right), \forall i \in \{0, 1, \cdots, mn - 1\} \tag{3d}$$

where $Q \in D_Q \subsetneq \mathbb{R}^{m \times n}$ denotes the quantized parameters of a linear layer[6]; $L : D_Q \to \mathbb{R}$ denotes the loss as a function of quantized parameters and typically contains, as part of L's definition, the model architecture, dataset, and **activation quantizers**; *weight_quantizer*: $\mathbb{R}^{m \times n} \to D_Q$ describes the behaviour of a generic quantizer like BBQ, QuEST, or LSQ; $\Phi^{-1} : (0, 1) \to \mathbb{R}$ is the inverse function of the standard Gaussian CDF; and sort: $\mathbb{R}^{mn} \to \mathbb{R}^{mn}$ treats its input vector as an array of numbers and returns the sorted list. Equation 3b is a standard constraint for QAT. Equation 3c states that CEWT is an extension to HO (Section 2.2), as a W that satisfy Equation 3c must lie on the radius-R (Frobenius/spectral norm) hypersphere. Equation 3d is the formal description of the zero-mean Gaussian assumption: for a sufficiently large mn, a random permutation of $\text{vec}(\hat{W})$ must look indistinguishable from a list of mn samples i.i.d. sampled from $N(0, 1)$. In essence, Equation 3d constrains the **center** (zero) and **shape** (Gaussian) of the weight distribution, while Equation 3c constrains the **width** of the weight distribution.

We discuss the algorithm of CEWT in Section 3.1. In Sections 3.2 and 3.3, we discuss how the function *weight_quantizer* in Equation 3b can be customized to maximally utilize the oscillation suppression effect of CEWT,

3.1 Algorithm

The algorithm of CEWT is Equation 4, which is meant to **replace Equation 2b** as CEWT is an extension to hypersphere optimizers. Figure 2 visualizes Equation 4.

$$\hat{W}^{(t)} = \text{mat}\left(\Phi^{-1}\left(\frac{1}{mn} \cdot \left(\text{ranks}(\text{vec}(\bar{W}^{(t)})) + \frac{1}{2}\right)\right)\right) \tag{4}$$

where Φ^{-1} is the element-wise inverse standard Gaussian CDF operation, vec: $\mathbb{R}^{m \times n} \to \mathbb{R}^{mn}$ flattens a matrix into a vector, mat: $\mathbb{R}^{mn} \to \mathbb{R}^{m \times n}$ is the reverse operation (un-flattening), and

[5]The definition of pre-clip weights vary across quantizers. See Algorithms 8, 9, and 10 for a few examples.

[6]Without loss of generality, we assume there is only one layer in the model and it is a linear one.

Algorithm 1 PyTorch-like Implementation of CEWT (Equation 4)

```
function CEWT(wbar)
    indices = argsort(wbar.flatten())
    ranks = zeros_like(indices)
    ranks[indices] = arange(indices.numel()) # scatter
    ranks = (ranks.to(float64) + 0.5) / ranks.numel() # use float64 for large mn
    what = erfinv(2 * ranks - 1) * (2 ** 0.5) # Φ^{-1}(x) = √2 · erf^{-1}(2x − 1)
    return what.to(float32).view_as(wbar)
end function
```

ranks: $\mathbb{R}^{mn} \to \{0, 1, \cdots mn-1\}^{mn}$ is defined as:

$$\text{ranks}(w)_i = \sum_{k=0}^{mn-1} \mathbf{1}_{(w_k < w_i) \vee (w_k = w_i \wedge k < i)}, \forall i \in \{0, 1, \cdots, mn-1\} \tag{5}$$

where $\mathbf{1}_{\text{cond}}$ returns a one if cond is true and a zero if cond is false. The function ranks counts the number of elements (w_k) in the input list (w) that is smaller than the current element (w_i). If multiple elements in w are equivalent ($w_k = w_i$), we use which element comes first in the list ($k < i$) to break ties, ensuring **each element is assigned a unique rank**. After ranking, the ranks are normalized to the range $(0, 1)$. Note the histogram of normalized ranks (visualized in Figure 2b) is guaranteed to be uniform as each rank shows up only once. Next, the normalized ranks are projected to Gaussian quantiles using the inverse standard Gaussian CDF. Since the normalized ranks are uniform, the histogram of projected quantiles (visualized Figure 2c) is guaranteed to be standard Gaussian (known as inverse transform sampling). Algorithm 1 shows a PyTorch-like implementation of Equation 4. Note that ranks can be efficiently implemented with argsort and scatter.

Equation 4 has a few properties. First, relative order is preserved ($\bar{W}_{ij} \leq \bar{W}_{kl} \implies \hat{W}_{ij} \leq \hat{W}_{kl}$) because each operation (ranks, multiplication by positive scalar, addition by positive scalar, and inverse standard Gaussian CDF) preserves relative order. For example, Figure 2d shows the mapping from an example $\bar{W}$ to its corresponding $\hat{W}$ is a non-decreasing function. Second, the histogram of $\hat{W}$ is always the standard Gaussian (as noted above) and therefore $\hat{W}$ satisfies Constraint 3d. This zero-mean Gaussian property can be exploited by quantizers to achieve desirable properties such as quantization error minimization (Panferov et al., 2025) and entropy maximization (Yang & Aamodt, 2026). Third, if the histogram of $\bar{W}$ contains rounding boundary density spikes (e.g. Figure 2a), CEWT's mapping effectively pulls weights too close to rounding boundary away from rounding boundary (e.g. Figure 2d). As such, CEWT can be interpreted as a hyperparameter-free hard-constraint variant of Dampening Loss (Nagel et al., 2022) or Periodic Interpolation (Li et al., 2026).

3.2 Removing Hadamard Transforms

Many implementations of *weight_quantizer* include orthogonal transforms such as the Hadamard Transform (Yang & Aamodt, 2026; Panferov et al., 2025; Lin et al., 2024), and we discuss two options to integrate such implementations with CEWT and why we choose the second option. The first option is to keep the original implementation of *weight_quantizer* and enforce the zero-mean Gaussian constraint in the Hadamard domain by converting $\bar{W}^{(t)}$ to the Hadamard domain, applying vanilla CEWT (Equation 4), and subsequently projecting the results back to the original domain, resulting in Equation 6:

$$\hat{W}^{(t)} = \mathbf{HT}\left(\text{mat}\left(\Phi^{-1}\left(\frac{1}{mn} \cdot \left(\text{ranks}(\text{vec}(\mathbf{HT}(\bar{W}^{(t)})) + \frac{1}{2}\right)\right)\right)\right) \tag{6}$$

The second option is to **remove** Hadamard Transform (HT) from the implementation of *weight_quantizer* and use **vanilla** CEWT (Equation 4). In terms of computational cost, the second option saves four Hadamard Transforms per layer per training iteration: one less HT in the forward pass of *weight_quantizer*, one less HT in the backward pass, and two less HTs in the optimizer (Equation 4 compared to 6). In terms of empirical performance, Section 4.2 shows that the second option consistently achieves lower loss than the first one. Therefore, when we use CEWT, we **remove the Hadamard Transform from *weight_quantizer***.

Per-Tensor	Per-Channel	TGCS (Ours)
$s_q = \text{quant_scale}(\text{vec}(W))$	$s_{qi} = \text{quant_scale}(W_i)$	$s_q = \text{quant_scale}(\text{vec}(W))$
$s_d = \text{dequant_scale}(\text{vec}(W))$	$s_{di} = \text{dequant_scale}(W_i)$	$s_{di} = \text{dequant_scale}(W_i)$
$Q = s_d \cdot \text{cliproud}(1/s_q \cdot W)$	$\forall i, Q_i = s_{di} \cdot \text{cliproud}(1/s_{qi} \cdot W_i)$	$\forall i, Q_i = s_{di} \cdot \text{cliproud}(1/s_q \cdot W_i)$

Table 1: Comparison between per-tensor, per-channel, and TGCS quantization. Example definitions of quant_scale, dequant_scale, and cliproud are found in Algorithms 8, 9, and 10.

3.3 Tensor-Wise Quantization Grid with Channel-wise Scaling (TGCS)

Many implementations of *weight_quantizer* uses **per-channel** quantization as opposed to **per-tensor** quantization (see Table 1 for the difference), and we discuss how to integrate CEWT with such per-channel weight quantizers. The underlying assumption of per-channel quantizers like BBQ (without HT) and QuEST (without HT) is that every **individual channel** (e.g. $W_0 \in \mathbb{R}^n$, $W_1 \in \mathbb{R}^n$, etc.) is a permutation of zero-mean Gaussian noise, whereas Equation 4 only enforce the zero-mean Gaussian constraint at the **tensor** level. To address this mismatch, we propose TGCS (see Table 1) which decouples the granularity between **quantization** and **dequantization**. Specifically, TGCS uses a **per-tensor quantization** scaling factor to construct a tensor-wise quantization grid whose underlying assumption is enforced by Equation 4, and **per-channel de-quantization** scaling factors to achieve the performance benefit of per-channel quantization. Section 4.2 visually (Figure 6) and quantitatively (Table 10) show that TGCS improves the oscillation suppression effect of CEWT relative to per-channel quantization. To summarize, when we use CEWT, we **modify *weight_quantizer* to use TGCS if the original implementation is per-channel**.

4 Evaluation

In this section, we systematically evaluate the effectiveness of CEWT. In Section 4.1, we compare CEWT against prior oscillation suppression methods, evaluated on a simple but important toy problem. In Section 4.2, we then evaluate the effectiveness of CEWT as a hyperparameter-free extension to existing hypersphere optimizers, by pre-training deeper networks under quantization. In Section 4.2, we also perform ablation studies to verify the insights in Sections 3.2 and 3.3.

4.1 Toy Problem

In this section, we evaluate the effectiveness of CEWT on a representative toy problem and confirm CEWT can, without introducing any hyperparameters, reach performance comparable to prior oscillation suppression methods. Our toy problem is as follows:

$$\min_W \frac{1}{2}||XQ - XW^*||_F^2 \text{ s.t. } Q = s \cdot \lfloor 1/s \cdot W \rceil \text{ and } ||W||_F = ||W^*||_F \tag{7}$$

where $X \in \mathbb{R}^{n \times d}$ is a dataset with n data points each with d features, $W \in \mathbb{R}^{d \times d}$ is the raw weights, $W^* \in \mathbb{R}^{d \times d}$ is one (not necessarily unique) optimal solution when no quantization (rounding) is involved, $XW^* \in \mathbb{R}^{n \times d}$ is the regression targets (expected output), $\lfloor \cdot \rceil$ is the element-wise rounding operation, and $s \in \mathbb{R}$ is the quantization step size. Section D.1 discusses that Problem 7 is a representative problem because it simulates multiple phenomena in the QAT of deep networks.

We compare CEWT against prior oscillation suppression methods: Pseudo-Quantization Noise (PQN) (Shin et al., 2023), Dampening Loss (DL) (Nagel et al., 2022), Iterative Freezing (IF) (Nagel et al., 2022), Confidence-Guided Annealing (CGA) (Liu et al., 2023), and Periodic Interpolation (PI) (Li et al., 2026). Section D shows details on how these algorithms are applied to Problem 7. If an oscillation suppression method comes with a hyperparameter, we choose a good hyperparameter and a bad hyperparameter, and report the mean-squared error between XQ and XW^*, and between XW and XW^*. To quantify weight oscillation, we report EMA Oscillation Frequency (Nagel et al., 2022) and Rounding Boundary Mass (RBM) which is inspired by Liu et al. (2023). Section C discusses the details of these two metrics and their tradeoffs. Table 2 suggests while existing non-PQN oscillation-suppression methods can consistently suppress rounding boundary spikes, the MSE they can achieve is highly dependent on the chosen hyperparameter. In contrast, CEWT can, without the

		STE	PQN	CEWT	DL	IF	CGA	PI
MSE(XQ, XW^*) (unit: 1e-3)	Good HP	81±0	87±0	**45±0**	**30±0**	59±0	53±0	54±0
	Bad HP				375±1	2080±12	**332±1**	1847±10
MSE(XW, XW^*) (unit: 1e-3)	Good HP	67±0	**4±0**	36±0	**34±0**	60±0	55±0	48±0
	Bad HP				**295±1**	2080±12	386±1	1807±9
EMA Oscillation Freq (unit: 1e-4)	Good HP	701±2	**9±0**	17±0	5±0	20±0	264±5	**0±0**
	Bad HP				**0±0**	**0±0**	1±0	**0±0**
RBM(vec($1/s \cdot W$)) (unit: 1e-3)	Good HP	573±1	**10±0**	**10±0**	6±0	13±0	133±3	**0±0**
	Bad HP				**0±0**	**0±0**	**0±0**	**0±0**

Table 2: Comparison between weight oscillation suppression methods. We report mean plus minus 2 standard deviations over 5 random seeds. Section C defines EMA Oscillation Frequency and RBM.

need for tuning, achieve performance comparable to non-PQN techniques with well-tuned hyperparameters. While PQN is hyperparameter-free and effective in oscillation suppression, it underperforms compared to the STE and CEWT due to the biasedness[7] in the optimization objective—PQN excels in minimizing MSE(XW, XW^*), but the STE and CEWT can find W that performs better in terms of the true objective MSE(XQ, XW^*). Since oscillation happens regardless (Nagel et al., 2022) of the rounding Jacobian estimation method, we also evaluate the effectiveness of CEWT when applied to HTGE (Pei et al., 2024), EWGS (Lee et al., 2021), and JacQuant-Probe (Yi et al., 2026) and show our results in Table 6. Results suggest CEWT can improve performance and suppress oscillation **regardless of the rounding Jacobian estimator**.

4.2 Deep Learning Experiments and Ablation Studies

In this section, we evaluate the effectiveness of CEWT (Equation 4, Section 3.2, and Section 3.3) as a hyperparameter-free extension to existing hypersphere optimizers. We perform Quantization-Aware Pre-training with various combinations of two model architectures (LLaMA (Touvron et al., 2023) and GPT (Radford et al., 2019)), various model scales (95M, 125M, 200M, 300M, 600M), four optimizers (AdamH (Kingma & Ba, 2017; Wen et al., 2026), MuonH (Jordan, 2024; Wen et al., 2026), LionH (Chen et al., 2023; Wen et al., 2026), and SSO (Xie et al., 2026)), two precisions (1-bit and 2-bit), and three quantizers (LSQ, QuEST, and BBQ). Sections F and G show detailed algorithms for optimizers and quantizers. Section J presents our learning rates and full-precision model perplexity. Our training recipe large follows Panferov et al. (2025), with 100 tokens per non-embedding parameter, significantly exceeding the Chinchilla scaling (Hoffmann et al., 2022). In total, our experiments (in Table 3) consumed **30 RTX-5090-days and 34 H100-days**. For every experiment, we quantize both activations and weights of linear layers. We present pre-training perplexity in Table 3, zero-shot evaluation in Section K, and training time in Section L. Results suggest that with a geomean increase of 4% in training time, **CEWT can consistently improve the performance of hypersphere optimizers in QAPT without introducing any hyperparameters**. Unlike the RBM of HO, the RBM of CEWT is **invariant** across optimizers and model sizes, and the RBM of CEWT depends **entirely** on the quantizer and precision[8]. This invariance is a *key* feature of CEWT—ensuring the number of parameters near rounding boundary is a constant. Figure 5 reports **both** EMA Oscillation Frequency and RBM of a 2-bit LLaMA-95M during training, suggesting that the perplexity reduction of CEWT comes from its oscillation suppression effect. Section M shows CEWT improves the accuracy of vision models. Section H shows CEWT can match/outperform the performance of Periodic Interpolation and Dampening Loss with the best hyperparameter tuned using grid search, and Section I shows CEWT consistently outperforms the hyperparameter-free method PQN.

We additionally perform ablation studies to test the effects of removing the HT and TGCS. We compare HO, CEWT without TGCS, CEWT, and CEWT-HT (first option in Section 3.2). Table 10 and Figure 6 suggest that TGCS can improve the density spike suppression effect of CEWT. Table 10 suggests that the combination of removing the HT from *weight_quantizer* and using vanilla CEWT outperforms the combination of keeping the HT and using CEWT-HT. Section B shows that the perplexity reduction of CEWT compared to HO is consistent regardless of R.

[7] Section D.3 discusses why MSE(XW, XW^*) is PQN's true, yet biased, objective.

[8] For BBQ, since pre-rounding weights are uniformly distributed, the RBM of CEWT is 0.010 regardless of precision. For QuEST and LSQ, the RBM of CEWT depends on the precision.

Model	Q-B	MuonH		Muon-CEWT		AdamH		Adam-CEWT		LionH		Lion-CEWT		SSO		SSO-CEWT	
		PPL	RBM	PPL	RBM	PPL	RBM	PPL	RBM	PPL	RBM	PPL	RBM	PPL	RBM	PPL	RBM
L-95	L-1	52.02	0.047	**50.63**	**0.006**	51.51	0.054	**50.27**	**0.006**	64.20	0.033	**54.02**	**0.006**	**51.23**	0.034	52.79	**0.007**
L-95	Q-1	51.97	0.215	**48.66**	**0.119**	52.85	0.240	**48.02**	**0.119**	51.33	0.244	**48.23**	**0.119**	53.61	0.182	**49.47**	**0.119**
L-95	B-1	50.89	0.031	**47.48**	**0.010**	51.06	0.034	**46.12**	**0.010**	50.70	0.034	**46.67**	**0.010**	52.15	0.023	**49.05**	**0.010**
L-95	L-2	33.73	0.030	**33.42**	**0.009**	34.64	0.032	**33.45**	**0.009**	41.21	0.035	**34.91**	**0.009**	34.72	0.024	**34.22**	**0.009**
L-95	Q-2	31.71	0.081	**30.66**	**0.056**	32.30	0.095	**30.67**	**0.056**	31.99	0.095	**30.68**	**0.056**	32.52	0.070	**31.22**	**0.056**
L-95	B-2	31.22	0.021	**30.60**	**0.010**	31.80	0.023	**30.18**	**0.010**	31.41	0.021	**30.29**	**0.010**	32.27	0.019	**31.35**	**0.010**
L-127	Q-1	44.82	0.205	**42.07**	**0.119**	45.48	0.234	**41.72**	**0.119**	45.38	0.232	**41.45**	**0.119**	46.21	0.175	**43.11**	**0.119**
L-127	B-1	44.08	0.025	**41.64**	**0.010**	44.51	0.029	**40.60**	**0.010**	43.35	0.027	**40.74**	**0.010**	45.07	0.019	**42.93**	**0.010**
L-127	Q-2	27.48	0.076	**26.59**	**0.056**	27.96	0.090	**26.52**	**0.056**	27.62	0.089	**26.49**	**0.056**	28.23	0.066	**27.18**	**0.056**
L-127	B-2	26.93	0.018	**26.71**	**0.010**	27.42	0.020	**26.13**	**0.010**	27.03	0.018	**26.09**	**0.010**	28.02	0.017	**27.28**	**0.010**
L-206	B-1	36.04	0.033	**33.43**	**0.010**	36.76	0.037	**33.20**	**0.010**	36.03	0.030	**33.00**	**0.010**	36.46	0.022	**34.67**	**0.010**
L-206	B-2	22.33	0.022	**21.88**	**0.010**	22.73	0.025	**21.77**	**0.010**	22.25	0.020	**21.49**	**0.010**	22.72	0.018	**22.16**	**0.010**
L-332	B-1	-	-	-	-	30.91	0.046	**28.77**	**0.010**	-	-	-	-	-	-	-	-
L-610	B-1	-	-	-	-	25.89	0.053	**24.49**	**0.010**	-	-	-	-	-	-	-	-
G-94	L-1	65.63	0.043	**60.03**	**0.008**	70.52	0.032	**61.65**	**0.010**	95.35	0.028	**74.04**	**0.008**	64.39	0.027	**61.46**	**0.009**
G-94	Q-1	62.24	0.204	**57.87**	**0.119**	61.64	0.238	**59.63**	**0.119**	61.69	0.240	**54.17**	**0.119**	58.75	0.171	**55.93**	**0.119**
G-94	B-1	59.26	0.033	**55.71**	**0.010**	60.70	0.033	**57.10**	**0.010**	60.66	0.033	**57.44**	**0.010**	57.18	0.023	**54.13**	**0.010**
G-94	L-2	35.63	0.033	**34.73**	**0.009**	39.09	0.031	**36.69**	**0.009**	50.20	0.034	**38.79**	**0.009**	36.39	0.025	**35.02**	**0.009**
G-94	Q-2	34.37	0.081	**33.56**	**0.056**	34.70	0.094	**34.32**	**0.056**	34.10	0.095	**32.71**	**0.056**	34.74	0.068	**33.53**	**0.056**
G-94	B-2	34.65	0.023	**33.82**	**0.010**	35.12	0.023	**34.11**	**0.010**	33.20	0.022	**32.17**	**0.010**	34.19	0.020	**33.46**	**0.010**
G-127	Q-1	49.13	0.202	**47.71**	**0.119**	52.00	0.230	**48.45**	**0.119**	48.57	0.230	**47.66**	**0.119**	49.37	0.168	**46.52**	**0.119**
G-127	B-1	48.82	0.026	**46.04**	**0.010**	50.39	0.027	**46.65**	**0.010**	47.76	0.026	**46.47**	**0.010**	47.77	0.021	**45.31**	**0.010**
G-127	Q-2	29.29	0.076	**28.40**	**0.056**	29.94	0.089	**28.66**	**0.056**	29.02	0.088	**28.10**	**0.056**	29.83	0.064	**28.57**	**0.056**
G-127	B-2	28.90	0.019	**28.59**	**0.010**	29.47	0.020	**28.57**	**0.010**	28.23	0.018	**27.42**	**0.010**	29.54	0.018	**28.72**	**0.010**
G-204	B-1	39.83	0.035	**37.41**	**0.010**	41.39	0.036	**38.52**	**0.010**	40.06	0.030	**36.48**	**0.010**	38.84	0.025	**36.99**	**0.010**
G-204	B-2	24.13	0.023	**23.43**	**0.010**	24.81	0.025	**24.02**	**0.010**	23.80	0.021	**23.01**	**0.010**	24.14	0.020	**23.34**	**0.010**
G-325	B-1	-	-	-	-	35.46	0.047	**32.74**	**0.010**	-	-	-	-	-	-	-	-

Table 3: Evaluation Perplexity of LLaMA and GPT models pre-trained on C4 with hypersphere optimizers, with and without CEWT. The first two columns are "Model" (L for LLaMA, G for GPT, with a dash followed by the number of parameters in millions) and "Q-B" which stands for Quantizer-Bits (L for LSQ, Q for QuEST, B for BBQ, with a dash followed by the **activation and weight precision** in number of bits). PPL stands for perplexity, and RBM stands for rounding boundary mass (Equation 15) of pre-round weights (Algorithms 8, 9, and 10).

5 LIMITATIONS

CEWT relies on the law of large numbers to guarantee that a random permutation of weights look indistinguishable from Gaussian noise. Therefore, for extremely small models with only a few hundred parameters in a single layer, CEWT does not work well as it forbids making small changes to parameters. Our ablation studies in Section D.9 shows that CEWT generally works well as long as the number of parameters in a layer exceed 2^{10}.

CEWT improves accuracy/reduces loss by suppressing rounding boundary weight oscillation, which is most severe during Quantization-Aware **Pre-Training** (QAPT), where dataset size is large and training time is long. Section D.9 shows that when training time is short and dataset size is much smaller compared to model size, weight oscillation is less severe, and the accuracy improvement of CEWT can be small and inconsistent. These insights from Section D.9 suggest CEWT might not be suitable for QAFT. However, Section 4.2 shows that CEWT consistently improves accuracy in large-scale QAPT.

CEWT maintains the zero-mean Gaussian constraint using argsort, which has an $O(mn\log(mn))$ time complexity. The runtime overhead can become more severe for wider models, but the overhead can potentially be reduced with engineering tweaks. For example, CEWT may be applied to each channel individually, reducing the time complexity to $O(mn\log(n))$. In practice we observe CEWT results in a geomean increase of 4% in training time, while matching the accuracy of existing methods with the best hyperparameters found using **computationally expensive grid searches**.

6 CONCLUSION

In this work, we introduce optimization with Constrained Empirical Weight disTributions (CEWT), an optimizer post-update step that projects weights to the nearest point in weight space whose empirical distribution matches a zero-mean Gaussian. CEWT enforces the zero-mean Gaussian assumption, which is built into the design of many existing quantizers. CEWT suppress the irregularly high density spikes that tend to occur due to the weight oscillation problem, thereby reducing gradient noise and improving accuracy.

AI USE STATEMENT

We used AI tools to paraphrase less than five sentences in this paper. We use AI tools to review our paper and manually address the reviews. We used in-IDE auto-complete features of AI tools for code writing. We used AI coding agents to review our code and check for bugs, and we manually addressed the reviews. We occasionally accept coding-agent-proposed code changes that are less than 10 lines after carefully reviewing. We take responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

We have uploaded our source code to OpenReview which allows all results (tables and figures) to be reproduced.

REFERENCES

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation, 2013. URL https://arxiv.org/abs/1308.3432.

Roberto L. Castro, Andrei Panferov, Soroush Tabesh, Oliver Sieberling, Jiale Chen, Mahdi Nikdan, Saleh Ashkboos, and Dan Alistarh. Quartet: Native fp4 training can be optimal for large language models. *CoRR*, abs/2505.14669, May 2025. URL https://doi.org/10.48550/arXiv.2505.14669.

Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Yao Liu, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V. Le. Symbolic discovery of optimization algorithms, 2023. URL https://arxiv.org/abs/2302.06675.

Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A downsampled variant of imagenet as an alternative to the cifar datasets, 2017. URL https://arxiv.org/abs/1707.08819.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: efficient finetuning of quantized llms. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

DaYou Du, Yijia Zhang, Shijie Cao, Jiaqi Guo, Ting Cao, Xiaowen Chu, and Ningyi Xu. BitDistiller: Unleashing the potential of sub-4-bit LLMs via self-distillation. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 102–116, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.7. URL https://aclanthology.org/2024.acl-long.7/.

Steven K. Esser, Jeffrey L. McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S. Modha. Learned step size quantization. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rkgO66VKDS.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models, 2022. URL https://arxiv.org/abs/2203.15556.

Keller Jordan. Muon: An optimizer for hidden layers in neural networks, December 2024. URL https://kellerjordan.github.io/posts/muon/.

Sehoon Kim, Coleman Hooper, Amir Gholami, Zhen Dong, Xiuyu Li, Sheng Shen, Michael W. Mahoney, and Kurt Keutzer. Squeezellm: dense-and-sparse quantization. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

Tanishq Kumar, Zachary Ankner, Benjamin Frederick Spector, Blake Bordelon, Niklas Muennighoff, Mansheej Paul, Cengiz Pehlevan, Christopher Re, and Aditi Raghunathan. Scaling laws for precision. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=wg1PCg3CUP.

Yann Le Cun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural Networks, Tricks of the Trade*, Lecture Notes in Computer Science LNCS 1524. Springer Verlag, 1998. URL http://leon.bottou.org/papers/lecun-98x.

Junghyup Lee, Dohyung Kim, and Bumsub Ham. Network quantization with element-wise gradient scaling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6448–6457, June 2021.

Dongyue Li, Zechun Liu, Kai Yi, Zhenshuo Zhang, Changsheng Zhao, Raghuraman Krishnamoorthi, Harshit Khaitan, Hongyang R. Zhang, and Steven Li. Winq: Accelerating quantization-aware training of language models around saddle points, 2026. URL https://arxiv.org/abs/2605.17471.

Haokun Lin, Haobo Xu, Yichen Wu, Jingzhi Cui, Yingtao Zhang, Linzhan Mou, Linqi Song, Zhenan Sun, and Ying Wei. Duquant: Distributing outliers via dual transformation makes stronger quantized LLMs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=mp8u2Pcmqz.

Jing Liu, Ruihao Gong, Xiuying Wei, Zhiwei Dong, Jianfei Cai, and Bohan Zhuang. QLLM: Accurate and efficient low-bitwidth quantization for large language models. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=FIplmUWdm3.

Shih-Yang Liu, Zechun Liu, and Kwang-Ting Cheng. Oscillation-free quantization for low-bit vision transformers. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. LLM-QAT: Data-free quantization aware training for large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 467–484, Bangkok, Thailand, August 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.26. URL https://aclanthology.org/2024.findings-acl.26/.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. Spinquant: LLM quantization with learned rotations. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=ogO6DGE6FZ.

Zechun Liu, Changsheng Zhao, Hanxian Huang, Sijia Chen, Jing Zhang, Jiawei Zhao, Scott Roy, Lisa Jin, Yunyang Xiong, Yangyang Shi, Lin Xiao, Yuandong Tian, Bilge Soran, Raghuraman Krishnamoorthi, Tijmen Blankevoort, and Vikas Chandra. Paretoq: Improving scaling laws in extremely low-bit LLM quantization. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b. URL https://openreview.net/forum?id=PMSNd8xTHp.

Shuming Ma, Hongyu Wang, Lingxiao Ma, Lei Wang, Wenhui Wang, Shaohan Huang, Li Dong, Ruiping Wang, Jilong Xue, and Furu Wei. The era of 1-bit llms: All large language models are in 1.58 bits, 2024a. URL https://arxiv.org/abs/2402.17764.

Yuexiao Ma, Huixia Li, Xiawu Zheng, Feng Ling, Xuefeng Xiao, Rui Wang, Shilei Wen, Fei Chao, and Rongrong Ji. Affinequant: Affine transformation quantization for large language models. In *ICLR*, 2024b. URL https://openreview.net/forum?id=of2rhALq81.

Vladimir Malinovskii, Denis Mazur, Ivan Ilin, Denis Kuznedelev, Konstantin Pavlovich Burlachenko, Kai Yi, Dan Alistarh, and Peter Richtárik. PV-tuning: Beyond straight-through estimation for extreme LLM compression. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=YvA8UF0I37.

Markus Nagel, Marios Fournarakis, Yelysei Bondarenko, and Tijmen Blankevoort. Overcoming oscillations in quantization-aware training. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 16318–16330. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/nagel22a.html.

Andrei Panferov, Jiale Chen, Soroush Tabesh, Roberto L. Castro, Mahdi Nikdan, and Dan Alistarh. QuEST: Training accurate LLMs over highly-compressed weights and activation. In *Sparsity in LLMs (SLLM): Deep Dive into Mixture of Experts, Quantization, Hardware, and Inference*, 2025. URL https://openreview.net/forum?id=ElgAzv9fNk.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf.

Zehua Pei, Xufeng Yao, Wenqian Zhao, and Bei Yu. Quantization via distillation and contrastive learning. *IEEE Transactions on Neural Networks and Learning Systems*, 35(12):17164–17176, 2024. doi: 10.1109/TNNLS.2023.3300309.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019. URL https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

Liliang Ren, Yang Liu, Yelong Shen, and Weizhu Chen. Rethinking language model scaling under transferable hypersphere optimization, 2026. URL https://arxiv.org/abs/2603.28743.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vision*, 115(3):211–252, December 2015. ISSN 0920-5691. doi: 10.1007/s11263-015-0816-y. URL https://doi.org/10.1007/s11263-015-0816-y.

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. Omniquant: Omnidirectionally calibrated quantization for large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=8Wuvhh0LYW.

Xuan Shen, Zhenglun Kong, Changdi Yang, Zhaoyang Han, Lei Lu, Peiyan Dong, Cheng Lyu, Chih hsiang Li, Xuehang Guo, Zhihao Shu, Wei Niu, Miriam Leeser, Pu Zhao, and Yanzhi Wang. Edgeqat: Entropy and distribution guided quantization-aware training for the acceleration of lightweight llms on the edge. *CoRR*, abs/2402.10787, 2024. URL https://doi.org/10.48550/arXiv.2402.10787.

Juncheol Shin, Junhyuk So, Sein Park, Seungyeop Kang, Sungjoo Yoo, and Eunhyeok Park. Nipq: Noise proxy-based integrated pseudo-quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3852–3861, June 2023.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers & distillation through attention. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 10347–10357. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/touvron21a.html.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023. URL https://arxiv.org/abs/2302.13971.

Zhijun Tu, Xinghao Chen, Pengju Ren, and Yunhe Wang. Adabin: Improving binary neural networks with adaptive binary sets. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XI*, pp. 379–395, Berlin, Heidelberg, 2022. Springer-Verlag. ISBN 978-3-031-20082-3. doi: 10.1007/978-3-031-20083-0_23. URL https://doi.org/10.1007/978-3-031-20083-0_23.

Kaiyue Wen, Xingyu Dang, Kaifeng Lyu, Tengyu Ma, and Percy Liang. Fantastic pretraining optimizers and where to find them 2.1: Hyperball optimization, 2026. URL https://psychedelic-sunstone-851.notion.site/Fantastic-Pretraining-Optimizers-and-Where-to-Find-Them-2-1-Hyperball-Optimization

Jonathan Wenshøj, Bob Pepin, and Raghavendra Selvan. Oscillations make neural networks robust to quantization. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=bPwcJ0nkDC.

Haocheng Xi, ChangHao Li, Jianfei Chen, and Jun Zhu. Training transformers with 4-bit integers. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=H9hWlfMT6O.

Tian Xie, Haoming Luo, Haoyu Tang, Hu Yiwen, Jason Klein Liu, Qingnan Ren, Yang Wang, Xin Zhao, Rui Yan, Bing Su, Chong Luo, and Baining Guo. Controlled LLM training on spectral sphere. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=5kTn1c3vtt.

Kohei Yamamoto. Learnable Companding Quantization for Accurate Low-bit Neural Networks . In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5027–5036, Los Alamitos, CA, USA, June 2021. IEEE Computer Society. doi: 10.1109/CVPR46437.2021.00499. URL https://doi.ieeecomputersociety.org/10.1109/CVPR46437.2021.00499.

Ge Yang, Edward Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tuning large neural networks via zero-shot hyperparameter transfer. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 17084–17097. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/8df7c2e3c3c3be098ef7b382bd2c37ba-Paper.pdf.

Greg Yang and Edward J. Hu. Tensor programs iv: Feature learning in infinite-width neural networks. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 11727–11737. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/yang21c.html.

Greg Yang, James B. Simon, and Jeremy Bernstein. A spectral condition for feature learning, 2024. URL https://arxiv.org/abs/2310.17813.

Ningfeng Yang and Tor M. Aamodt. Improving the straight-through estimator with zeroth-order information. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=ceTeM2Xl1n.

Ningfeng Yang and Tor M. Aamodt. Boosting entropy with bell box quantization. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=tY9yPAT3PU.

Kai Yi, Vignesh Vivekraja, Harshit Khaitan, and Steven Li. Jacquant: Ste-free quantization-aware training via learned jacobian surrogates, 2026. URL https://arxiv.org/abs/2605.25469.

A EXAMPLES OF ROUNDING BOUNDARY WEIGHT OSCILLATION

In this section, we discuss two example problems in which rounding boundary weight oscillation occurs. The first example is a one-dimensional optimization problem that reveals some insight on the root cause of rounding boundary weight oscillation. The second example is a two-dimensional optimization problem that shows the oscillation of one quantized parameter at one of its rounding boundaries can introduce significant variance to the partial derivative of another parameter due to parameter interdependency.

A.1 A 1D EXAMPLE

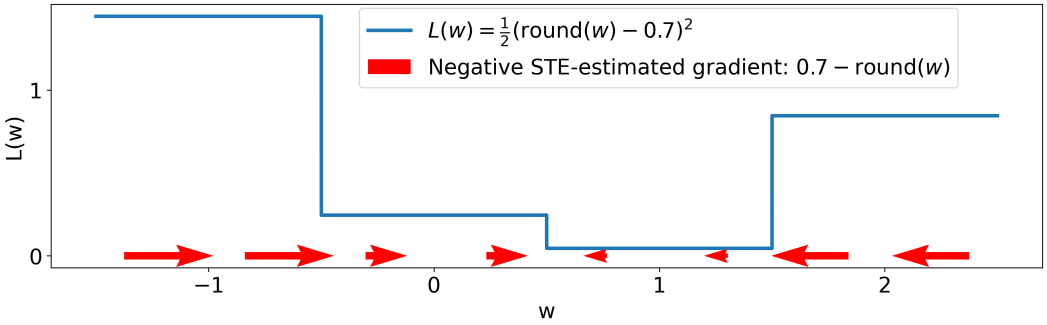


Figure 3: Visualization of $L(w) = \frac{1}{2}(\text{round}(w) - 0.7)^2$ as a function and its negative STE-estimated gradient $0.7 - \text{round}(w)$ as a one-dimensional vector field. The direction of an arrow represents its sign (left for negative and right for positive), and the size of an arrow represents its magnitude.

The root cause of rounding boundary weight oscillation can be explained with a simple example. Consider the problem of finding the closest integer to 0.7. To solve this problem with gradient descent, we reformulate it to the following alternative problem which has a continuous optimization variable $w \in \mathbb{R}$:

$$\begin{aligned} \min_{w \in \mathbb{R}} \ & \frac{1}{2}(q - 0.7)^2 \\ & q = \text{round}(w) \end{aligned} \tag{8}$$

Once a solution of this alternative problem is found, denoted as w^*, we can calculate $\text{round}(w^*)$ which is the solution to the original problem of finding the closest integer to 0.7. Problem 11 has more than one solution. Any w' that satisfies $0.5 \leq w' < 1.5$ is a valid solution as it suggests that the closest integer to 0.7 is $\text{round}(w') = 1$.

Let $L(w) = \frac{1}{2}(q - 0.7)^2$. We invoke the chain rule to attempt to calculate dL/dw. However, the round function is not differentiable, and we use the Straight-Through Estimator (STE) to estimate the derivative of the round function as 1, arriving at the following approximated gradient:

$$\frac{dL}{dw} = \frac{dL}{dq} \cdot \frac{dq}{dw} = (q - 0.7) \cdot \text{round}'(w) \underset{\text{STE}}{\approx} (q - 0.7) \cdot 1 = \text{round}(w) - 0.7 \tag{9}$$

With gradient descent, our update rule is therefore:

$$w \leftarrow w - \eta(\text{round}(w) - 0.7) \tag{10}$$

where η is a sufficiently small learning rate. Figure 3 visualizes $L(w)$ and its negative STE estimated gradient $0.7 - \text{round}(w)$ as a one-dimensional vector field. Figure 3 shows when $L(w)$ is *not* minimized, the negative STE-estimated gradient points towards the minimum solution. However, when $L(w)$ *is* minimized (i.e. $0.5 \leq w < 1.5$), the negative STE-estimated gradient evaluates to $0.7 - 1 = -0.3$, which is a negative number and *not* zero. This means even though an optimal solution has been found, w will continue to decrease. Eventually, w becomes slightly smaller than the rounding boundary 0.5, in which case the negative STE-estimated gradient evaluates to $0.7 - 0 = 0.7$, a positive number. This means w will increase until w becomes larger than 0.5 again. In other

words, w will continuously oscillate between the left and right side of 0.5 and never stop moving, and $q = \text{round}(w)$ will constantly oscillate between 0 and 1. This oscillation happens because q is not a continuous variable, and therefore q cannot encode $q = 0.7$, the stationary point that zeros the estimated gradient $q - 0.7$.

A.2 A 2D Example

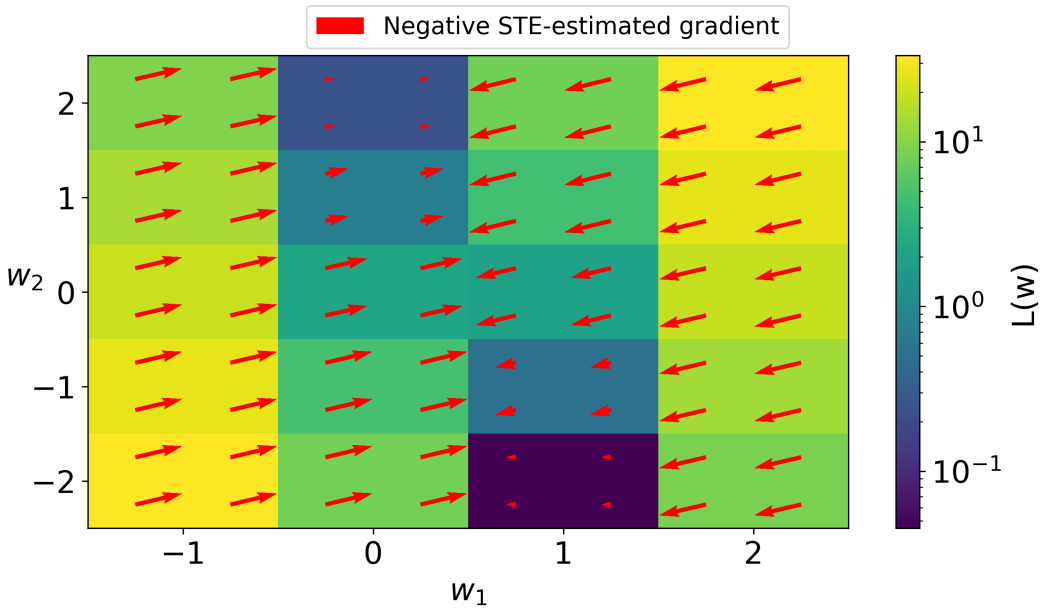


Figure 4: Visualization of $L(w) = \frac{1}{2}(q_1 - 0.7)^2 + \frac{1}{2}(4q_1 - q_2 - 2)^2$ as a function and its negative STE-estimated gradient $[-(q_1 - 0.7) - 4 \cdot (4q_1 + q_2 - 2), -(4q_1 + q_2 - 2)]^\top$ as a two-dimensional vector field. Note that $q_1 = \text{round}(w_1)$ and $q_2 = \text{round}(w_2)$. Darker colored regions have lower loss than brighter regions.

While rounding boundary weight oscillation appears harmless, we show it can significantly increase gradient variance in high-dimensional problems due to parameter interdependency. Consider the following 2D problem:

$$\begin{aligned} \min_{w \in \mathbb{R}^2} \; & \frac{1}{2}(q_1 - 0.7)^2 + \frac{1}{2}(4q_1 - q_2 - 2)^2 \\ \text{s.t. } & q_1 = \text{round}(w_1) \\ & q_2 = \text{round}(w_2) \end{aligned} \tag{11}$$

Let $L(w) = \frac{1}{2}(q_1 - 0.7)^2 + \frac{1}{2}(4q_1 - q_2 - 2)^2$. We invoke the chain rule to attempt to calculate the partial derivatives with respect to w_1 and w_2, but the round function is not differentiable. So, we use the Straight-Through Estimator (STE) to estimate the derivative of the round function as 1, arriving at the following expressions:

$$\begin{aligned} \frac{\partial L}{\partial w_1} &= \frac{\partial L}{\partial q_1} \cdot \frac{\partial q_1}{\partial w_1} \underset{\text{STE}}{\approx} (q_1 - 0.7) + 4 \cdot (4q_1 + q_2 - 2) \\ \frac{\partial L}{\partial w_2} &= \frac{\partial L}{\partial q_2} \cdot \frac{\partial q_2}{\partial w_2} \underset{\text{STE}}{\approx} (4q_1 + q_2 - 2) \end{aligned} \tag{12}$$

Therefore, we use the following STE-estimate gradient:

$$\nabla w = \left[\frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}\right]^\top \underset{\text{STE}}{\approx} \begin{bmatrix} (q_1 - 0.7) + 4 \cdot (4q_1 + q_2 - 2) \\ (4q_1 + q_2 - 2) \end{bmatrix} \tag{13}$$

Figure 4 visualizes $L(w)$ and its negative STE-estimated gradient. Figure 4 shows that when w_1 and w_2 are initialized close to zero, the negative estimated gradient has a strong tendency to immediately

push w_1 towards the line $w_1 = 0.5$, where w_1 will start to oscillate (as long as $-1.5 \leq w_2 < 1.5$). When $w_1 \approx 0.5$, the loss can no longer be significantly reduced by changing w_1, but changing w_2 (by pushing it upward towards the line $w_2 = 2$ or by pushing it downwards towards $w_2 = -2$) can reduce the loss significantly. However, changing w_2 can be challenging when $w_1 \approx 0.5$. When w_1 is close to, but less than 0.5, the negative estimated gradient has a tendency to increase w_2. When w_1 is close to, but greater than or equal to 0.5, the negative estimated gradient has a tendency to decrease w_2. Since w_1 constantly oscillates around $w_1 = 0.5$, the trajectory of w_2 will be noisy, with some iterations canceling the progress of previous iterations. The core reason is $\frac{\partial L}{\partial w_2}$ **depends** (see Equation 12) on the value of q_1. When w_1 oscillates around $w_1 = 0.5$, the value of q_1 jumps between 0 and 1, thereby increasing the variance of $\frac{\partial L}{\partial w_2}$. To summarize, **due to parameter interdependency, rounding boundary weight oscillation can introduce significant gradient variance in high-dimensional problems**.

B IS R A HYPERPARAMETER OF CEWT?

In this section, we discuss that R is **not** a hyperparameter of CEWT or HO, as **every** deep learning pre-training script choose a value of R by randomly initializing weight matrix $W \in \mathbb{R}^{m \times n}$ from a zero-mean distribution with a variance of σ^2. For a sufficiently large mn, the Frobenius norm radius is $R \approx \sigma\sqrt{mn}$ and the spectral norm radius is $R \approx \sigma(\sqrt{m} + \sqrt{n})$. Therefore, at random initialization time, every deep learning model lie on some radius-R hypersphere. Hypersphere optimizers (e.g. AdamH/MuonH) proposes to **keep** the model on this radius-R hypersphere, thereby **reusing** the hyperparameter R, bounding the Frobenius norm of W, and removing the need for weight decay. In this work, we use the Lecun initialization (Le Cun et al., 1998) which results in the Frobenius norm radius $R \approx \sqrt{m}$. We additionally perform ablation studies on other possible weight initialization strategies/values of R: PyTorch (Paszke et al., 2019) default initialization, Spectral μP(Yang et al., 2024), and the GPT-2 (Radford et al., 2019) initialization strategy. Table 4 shows that **the perplexity reduction of CEWT compared to HO is consistent across multiple values of** R. For clarity, Table 5 lists multiple combinations of regular/hypersphere optimizers and weight oscillation suppression methods and shows that CEWT has the **least** number (one to be exact) of hyperparameters among all combinations. In particular, Table 5 shows R is **not** a hyperparameter **introduced by CEWT** but is a part of **every** deep learning pre-training script that performs weight random initialization.

Name	Lecun	PyTorch	Spectral μP	GPT-2 (non-residual)	GPT-2 (residual)
Init Distribution	Normal	Uniform	Normal	Normal	Normal
Init Mean	0	0	0	0	0
Init Std	$\sqrt{\frac{1}{n}}$	$\sqrt{\frac{1}{3n}}$	$\frac{\sqrt{m/n}}{\sqrt{m}+\sqrt{n}}$	0.02	$\frac{0.02}{\sqrt{2l}}$
Frobenius norm R	$\sqrt{m}$	$\sqrt{m/3}$	$\frac{m}{\sqrt{m}+\sqrt{n}}$	$0.02\sqrt{mn}$	$0.02\sqrt{\frac{mn}{2l}}$
AdamH Perp	31.78	31.52	31.39	30.91	
Adam-CEWT Perp	**30.12**	**30.22**	**29.82**	**29.77**	

Table 4: Robustness of CEWT over different values of R. m is the number of output features of a linear layer. n is the number of input features of a linear layer. l is the number of layers in a model. Experiments are conducted using a LLaMA-95M with weights and activations quantized to 2-bit using BBQ. The model is pre-trained on 3 billion C4 tokens. We report perplexity (Perp).

	Weight Init HP (R)	Weight Decay HP	RB Weight Oscillation HP	# HPs
Adam	Yes	Yes	No	2
AdamH	Yes	No	No	**1**
Adam + Periodic Interpolation	Yes	Yes	Yes	3
AdamH + Periodic Interpolation	Yes	No	Yes	2
Adam + Dampening Loss	Yes	Yes	Yes	3
AdamH + Dampening Loss	Yes	No	Yes	2
Adam + PQN	Yes	Yes	No	2
AdamH + PQN	Yes	No	No	**1**
Adam-CEWT	Yes	No	No	**1**

Table 5: The number of hyperparameters among weight initialization radius R, weight decay, and rounding boundary weight oscillation, for multiple combinations of regular/hypersphere optimizers and oscillation suppression methods. A "yes" means the hyperparameter is used. A "no" means the hyperparameter is *not* used.

C Metrics To Quantify Rounding Boundary Weight Oscillation

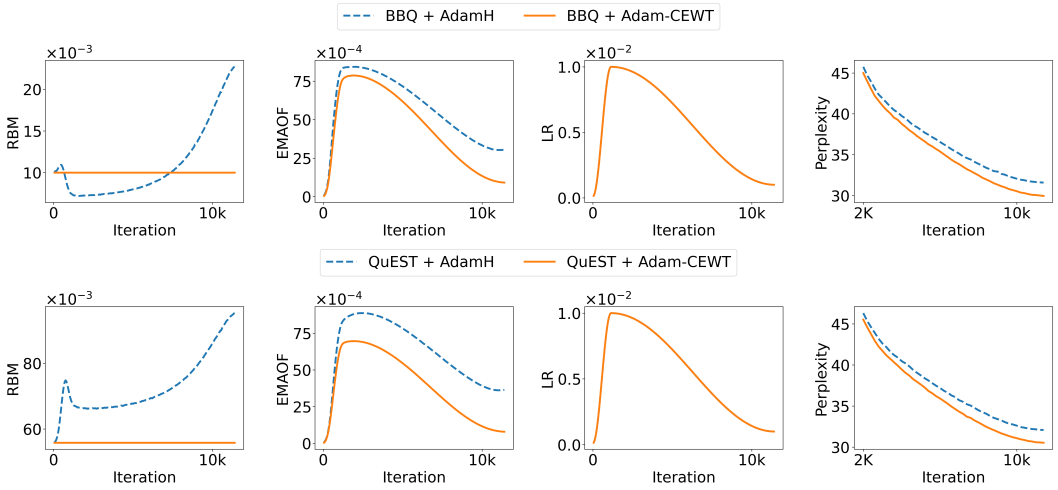


Figure 5: Rounding Boundary Mass (RBM), EMA Oscillation Frequency (EMAOF), learning rate (LR), and perplexity of a LLaMA-95M model with 2-bit weights and activations, pretrained on 3B C4 tokens. EMAOF is a more direct indicator of rounding boundary weight oscillation, but it correlates with LR and is not scale-invariant. RBM is scale-invariant, and RBM shows CEWT regulates the number of parameters close to rounding boundary *by design*.

In this work, we use two metrics to quantify weight oscillation: EMA Oscillation Frequency (Nagel et al., 2022) and Rounding Boundary Mass (RBM) which is inspired by Liu et al. (2023). EMA Oscillation Frequency (EMAOF) quantifies the frequency of rounding boundary weight oscillation, but calculating EMAOF introduces training storage overhead, and EMAOF correlates with the learning rate. RBM quantifies the amount of weights sufficiently close to a rounding boundary, has no training storage overhead, and is invariant of learning rate.

EMA Oscillation Frequency, denoted $f^{(t)}$, is calculated as follows. Denoted $m \in (0, 1)$ as the momentum of the exponential moving average. Denote $w_{\text{int}}^{(t)}$ as the integer value, or quantization bin number, of a parameter w at training iteration t. Denote $\Delta^{(t)} \in \mathbb{Z}$ as a storage variable initialized to zero. Note $f^{(t)}$ is also a storage variable initialized to zero. Variables $\Delta^{(t)}$ and $f^{(t)}$ are updated

every training iteration as follows:

$$\begin{aligned}\Delta_{\text{int}} &= w_{\text{int}}^{(t)} - w_{\text{int}}^{(t-1)} \\ o &= \mathbf{1}_{(\text{sign}(\Delta_{\text{int}}) \neq \text{sign}(\Delta_{\text{int}}^{\tau})) \wedge (\Delta_{int} \neq 0)} \\ f^{(t)} &= m \cdot o + (1-m) \cdot f^{(t-1)} \\ \Delta^{(t)} &= \begin{cases} \Delta_{\text{int}}, & \text{if } o = 1 \\ \Delta^{(t-1)}, & \text{otherwise} \end{cases}\end{aligned} \tag{14}$$

where $\mathbf{1}_{\text{cond}}$ returns one if cond is true and a zero if cond is false. We use $m = 0.1$ for our toy problem evaluation in Tables 2 and 6. We use $m = 0.01$ for our deep learning experiments in Figure 5. For an experiment with a total of T training iterations, we report $f^{(T)}$ averaged across all parameters in the model. EMAOF lies in the range $[0, 1]$. $f^{(t)} = 1$ indicates that w crosses a rounding boundary every iteration, and that the direction in which w crosses that rounding boundary flips every iteration. $f^{(t)} = 0$ indicates that there is no rounding boundary weight oscillation recently. EMAOF is a direct indicator of rounding boundary weight oscillation, but it introduces a significant training overhead of two storage variables per parameter: $\Delta^{(t)}$ and $f^{(t)}$. Additionally, Figure 5 shows EMAOF correlates with learning rate and is therefore not scale-invariant during training and across different learning rates or learning rate schedules. For example, a model may have extremely sharp density spikes near rounding boundaries, but if the learning rate is 0, oscillation *cannot* happen and therefore EMAOF is 0. Therefore, for large scale-experiments (e.g. Table 3), we prefer and use RBM as it introduces no training storage overhead and is scale-invariant.

Rounding Boundary Mass (RBM) quantifies the amount of weights which are sufficiently close to their closest rounding boundary. Given a list of pre-rounding[9] weights $w \in \mathbb{R}^n$, RBM: $\mathbb{R}^n \to [0, 1]$ is defined as:

$$\text{RBM}(w) = \frac{1}{n} \sum_{i=0}^{n-1} \mathbf{1}_{|w_i - (\text{floor}(w_i) + 0.5)| < \epsilon} \tag{15}$$

where ϵ is chosen to be 0.005 following Liu et al. (2023), and $\mathbf{1}_{\text{cond}}$ returns one if cond is true and a zero if cond is false. RBM is a estimation to the area under the rounding boundary spikes, and therefore also functions as a proxy to the severity of weight oscillation. **RBM does not introduce any training overhead**, and therefore we prefer RBM for large-scale experiments.

D Details of Evaluation on Toy Problem

In this section we discuss details of the rounding boundary weight oscillation suppression algorithms we used in Section 4.1. We reiterate Problem 7 here:

$$\begin{aligned}\min_{W} \ & \frac{1}{2} ||XQ - XW^*||_F^2 \\ \text{s.t. } & Q = s \cdot \lfloor 1/s \cdot W \rceil \\ & ||W||_F = ||W^*||_F\end{aligned} \tag{16}$$

where $s \in \mathbb{R}$, $X \in \mathbb{R}^{n \times d}$, $W, W^* \in \mathbb{R}^{d \times d}$, and $n = d = 2^{10}$. Elements of X are i.i.d. sampled from $N(0, 1)$; elements of W are initialized from $N(0, 1/d)$; elements of W^* are i.i.d. sampled from $N(0, 1/d)$; and s is chosen to be $1/\sqrt{d}$. Note that X and W^* are constants (already sampled and fixed), whereas W is randomly initialized and learnable. Note that Problem 11 is an instance of Problem 16 with $n = d = 1$, $s = 1$, $X = [1]$, and $W^* = [0.7]$. We use the AdamH optimizer, a learning rate of 0.1, and the Cosine Annealing scheduler. We train a total of 200 iterations.

D.1 Why this Problem?

While Problem 16 (or Problem 7) is simple, it simulates multiple phenomena observed in QAT of deep networks. First, Problem 16 recreates weight oscillation. Second, weight oscillation introduces

[9] The definition of pre-rounding weights varies across quantizers. See Algorithms 8, 9, and 10 for examples.

detrimental noise in the optimization process because of **parameter interdependency**. Specifically, the partial derivative with respect to the quantized parameter $Q_{i,j}$ is:

$$\frac{\partial L}{\partial Q_{i,j}} = [X^T(XQ - XW^*)]_{i,j} = \sum_{k=0}^{d-1} [X^T X]_{i,k} \cdot (Q_{k,j} - W^*_{k,j}) \tag{17}$$

In other words, how $Q_{i,j}$ should move **depends** on the value of $Q_{0,j}$, $Q_{1,j}$, $\cdots$, and $Q_{(d-1),j}$. If any of these parameters starts oscillating (constantly switches between two quantization bins), noise is introduced into the movement of $Q_{i,j}$. Third, the solution to Problem 16 need not be unique, which simulates the phenomenon that there can be many approximate solutions (local minima) in the loss landscapes of deeper networks. For example, the loss in Problem 16 is zeroed when $Q = W^* + E$ as long as $XE = 0$, which means there can be multiple non-zero options for E when the null space of X is non-trivial. If W^* cannot be perfectly represented/encoded by Q due to quantization constraint, the optimization process may find some $W^* + E$ (where $E \not\approx 0$) that can be better represented by Q. Fourth, the constraint $||W||_F = ||W^*||_F$ simulates the use of weight decay or hypersphere optimization, in that when multiple solutions exist, ones with reasonably small Frobenius norms are preferred.

D.2 STE

When we use the STE (Bengio et al., 2013), we estimate the gradient as the following:

$$G = X^T(XQ - XW^*) \tag{18}$$

The estimated gradient is further processed by AdamH to produced a weight update. AdamH also maintains the Frobenius norm constraint.

D.3 PQN

When we use PQN (Shin et al., 2023), the optimization objective changes to the following:

$$\begin{aligned} &\min_W E_{\Xi_{i,j} \sim U(-\frac{1}{2}, \frac{1}{2}), \forall i,j} \left[\frac{1}{2} ||XQ' - XW^*||_F^2 \right] \\ &\text{s.t. } Q' = s \cdot (1/s \cdot W + \Xi) = W + s\Xi \\ &\quad ||W||_F = ||W^*||_F \end{aligned} \tag{19}$$

In other words, the rounding error is simulated with addition of a noise matrix $\Xi \in \mathbb{R}^{m \times n}$ whose elements are i.i.d. sampled from $U(-\frac{1}{2}, \frac{1}{2})$. During training, we perform Monte Carlo sampling:

$$\begin{aligned} \Xi_{i,j} &\sim U(-\frac{1}{2}, \frac{1}{2}), \forall i, j \in \{0, 1, \cdots, d-1\} \\ L(W, \Xi) &= \frac{1}{2} ||X(W + s\Xi) - XW^*||_F^2 \\ \frac{\partial L(W, \Xi)}{\partial W} &= X^T(X(W + s\Xi) - XW^*) \end{aligned} \tag{20}$$

and we use a randomly sampled $\frac{\partial L(W,\Xi)}{\partial W}$ to perform weight update.

To understand the expected behaviour of training, we perform the following derivation on our loss function in Equation 19:

$$
\begin{aligned}
E\left[\frac{1}{2}||XQ' - XW^*||_F^2\right] &= \frac{1}{2}E[||XQ' - XW^*||_F^2] \\
&= \frac{1}{2}||E[XQ' - XW^*]||_F^2 + \frac{1}{2}\sum_{\forall i,j}\text{Var}([XQ' - XW^*]_{i,j}) \\
&= \frac{1}{2}||E[X(W + s\Xi - W^*)]||_F^2 + \frac{1}{2}\sum_{\forall i,j}\text{Var}([X(W + s\Xi - W^*)]_{i,j}) \\
&= \frac{1}{2}||X \cdot E[(W + s\Xi - W^*)]||_F^2 + \frac{1}{2}\sum_{\forall i,j}\text{Var}([sX\Xi]_{i,j}) \\
&= \frac{1}{2}||X \cdot (W + s \cdot E[\Xi] - W^*)||_F^2 + \frac{1}{2}\sum_{\forall i,j}\frac{s^2}{12}||X_i||_2^2 \\
&= \frac{1}{2}||X \cdot (W - W^*)||_F^2 + C \\
&= \frac{1}{2}||XW - XW^*||_F^2 + C \\
&= \frac{nd}{2} \cdot \text{MSE}(XW, XW^*) + C
\end{aligned}
\tag{21}
$$

which suggests that the modified loss function is simply $\text{MSE}(XW, XW^*)$ scaled by a constant and shifted by a constant. However, Table 2 shows PQN achieves suboptimal $\text{MSE}(XQ, XW^*)$ despite excelling in reducing $\text{MSE}(XW, XW^*)$. In contrast, the STE and CEWT are capable of finding a W that performs better in terms of the true objective $\text{MSE}(XQ, XW^*)$. This suggests that Problem 19 is a biased objective, or that a solution to Problem 19 is not a solution to Problem 16.

D.4 Dampening Loss (DL)

When we use DL (Nagel et al., 2022), the optimization objective changes to the following:

$$
\begin{aligned}
&\min_W \frac{1}{2}||XQ - XW^*||_F^2 + \frac{\alpha}{2}||W - Q||_F^2 \\
&\text{s.t. } Q = s \cdot \lfloor 1/s \cdot W \rceil \\
&\quad ||W||_F = ||W^*||_F
\end{aligned}
\tag{22}
$$

where α is a regularization hyperparameter. We estimate the gradient as follows:

$$
G = X^T(XQ - XW^*) + \alpha(W - Q) \tag{23}
$$

Note that the second gradient term $\alpha(W - Q)$ assumes (Nagel et al., 2022) Q is a constant in the second loss term $||W - Q||_F^2$.

Our findings agree with Nagel et al. (2022), in that α should be small at the beginning of training and should be gradually increased during training. An α that is too large too early generally prevents convergence.

D.5 Iterative Freezing (IF)

When we use IF (Nagel et al., 2022), our weight update algorithm is Algorithm 2. Our findings agree with Nagel et al. (2022), in that hyperparameter f_{th} should be large at the beginning of training and gradually decrease during training. An f_{th} that is too small too early prevents convergence.

Algorithm 2 Iterative Freezing applied to Problem 16

function IF(f_{th}, m)
  Introduce four variables: $F \in \mathbb{R}^{d\times d}$, $B \in \{\text{False}, \text{True}\}^{d\times d}$, $\Delta \in \mathbb{R}^{d\times d}$, and $W_{\text{EMA}} \in \mathbb{R}^{d\times d}$
  $F^{(0)} = 0$
  $B^{(0)} = 0$
  $\Delta = 0$
  $W_{\text{EMA}} = \lfloor 1/s \cdot W^{(0)} \rceil$
  for $t = 1$ to T **do**
    $Q = s \cdot \lfloor 1/s \cdot W^{(t-1)} \rceil$
    $G = X^T(XQ - XW^*)$
    perform optimizer step with gradient G to obtain updated weight $W_{\text{curr}}^{(t)}$, but only update weights at index (i, j) s.t. $B_{i,j}^{(t-1)}$ is True
    $\Delta_{\text{int}} = \lfloor 1/s \cdot W^{(t)} \rceil - \lfloor 1/s \cdot W^{(t-1)} \rceil$
    $O = (\text{sign}(\Delta_{\text{int}}) \neq \text{sign}(\Delta^{(t-1)})) \wedge (\Delta_{\text{int}} \neq 0)$ ▷ Element-wise ops (sign, $\neq$, and $\wedge$)
    $F^{(t)} = m \cdot O + (1 - m) \cdot F^{(t-1)}$
    $B^{(t)} = B^{(t-1)} \vee (F^{(t)} > f_{th})$ ▷ Element-wise ops ($\vee$ and $>$)
    $W^{(t)} = \text{where}\left(F^{(t)} > f_{th}, s \cdot W_{\text{EMA}}^{(t-1)}, W_{\text{curr}}\right)$ ▷ "where" is similar to torch.where
    $W_{\text{EMA}}^{(t)} = m \cdot \lfloor 1/s \cdot W^{(t-1)} \rceil + (1 - m) \cdot W_{\text{EMA}}^{(t-1)}$
    $\Delta^{(t)} = \text{where}\left(O, \Delta_{\text{int}}, \Delta^{(t-1)}\right)$
  end for
end function

D.6 Confidence Guided Annealing

When we use CGA (Liu et al., 2023), we prune the STE gradient before passing it to the optimizer, as follows:

$$\begin{aligned} D &= |(1/s \cdot W) - (\lfloor 1/s \cdot W \rfloor + 0.5)| \\ G &= \left(X^T(XQ - XW^*)\right) \odot \underbrace{\mathbf{1}_{D \leq \text{BR}_x}}_{\text{pruning mask}} \end{aligned} \tag{24}$$

where $|\cdot|$, $\lfloor\cdot\rfloor$, $\odot$ are the element-wise absolute value operator, the element-wise floor operator, and the element-wise multiplication operator, respectively; $\mathbf{1}_{\text{cond}}$ returns a one if cond is true or a zero if cond is false; and $\text{BR}_x \in [0, 0.5]$ is a hyperparameter. In practice, we use $0.5/\text{BR}_x \cdot G$ (instead of G) as the actual gradient to compensate for the decrease in gradient norm due to gradient pruning.

We find that BR_x should be large at the beginning of training and should be gradually decreased during training. A BR_x that is too small too early prevents convergence.

D.7 Periodic Interpolation (PI)

When we use PI (Li et al., 2026), our weight update algorithm is Algorithm 3. PI introduces hyperparameters α and K, both of which need to be tuned. We find α does not need to be gradually increased, since the effects of interpolation naturally becomes stronger as learning rate decreases.

Algorithm 3 Periodic Interpolation applied to Problem 16

function PI(α, K)
  for $t = 1$ to T **do**
    $Q = s \cdot \lfloor 1/s \cdot W^{(t-1)} \rceil$
    $G = X^T(XQ - XW^*)$
    perform optimizer step with gradient G to obtain updated weight $W^{(t)}_{\text{curr}}$
    if $t \,\%\, K = 0$ **then**
      $W^{(t)} = (1-\alpha) \cdot W^{(t)}_{\text{curr}} + \alpha \cdot (s \cdot \lfloor 1/s \cdot W^{(t)} \rceil)$
    else
      $W^{(t)} = W^{(t)}_{\text{curr}}$
    end if
  end for
end function

D.8 Other rounding Jacobian Estimators

In Table 6, we additionally apply CEWT to multiple rounding Jacobian Estimators, demonstrating that CEWT can suppress weight oscillation and improve model performance regardless of the rounding Jacobian estimator used.

		STE	HTGE	JacQuant-Probe	EWGS
MSE(XQ, XW^*) (unit: 1e-3)	Default	81±0	53±0	63±0	61±0
	With CEWT	**45±0**	**46±0**	**52±0**	**51±0**
RBM(vec($1/s \cdot W$)) (unit: 1e-3)	Default	573±1	177±1	349±1	324±1
	With CEWT	**10±0**	**10±0**	**10±0**	**10±0**
EMA Oscillation Freq (unit: 1e-4)	Default	701±2	245±1	511±2	392±1
	With CEWT	**17±0**	**30±0**	**37±0**	**13±0**

Table 6: Effectiveness of CEWT on different rounding Jacobian estimators.

D.9 When is CEWT Less Effective?

In this section, we discuss when rounding boundary weight oscillation is less severe, and when CEWT is less effective.

In Table 7, we fix $d = 2^{10}$ and the number of training iterations (200), and we sweep over $n \in \{1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024\}$. Our results show that when the model size is significantly larger than the dataset size, rounding boundary weight oscillation is less severe, and the loss reduction effect of CEWT is small and almost negligible.

In Table 8, we fix $n = 2^{10}$ and $d = 2^{10}$, and we sweep over the number of training iterations in $\{1, 3, 6, 12, 25, 50, 100, 200\}$. Our results show that while CEWT can increase MSE loss when the number of training iterations is small (e.g. less than 100), as long as the model is trained to full convergence (i.e. STE loss no longer decreases significantly when the number of epochs doubles), CEWT can consistently achieve lower loss than the STE. Table 8 also suggests rounding boundary oscillation is less severe when the training time is not sufficiently long.

In Table 9, we fix $n = 2^{10}$ and the number of iterations (200), and sweep over $d \in \{1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024\}$. Our result suggest CEWT can consistently reduce loss as long as model size is sufficiently large (e.g. $d \geq 32$). CEWT is designed for large models, as it relies on the law of large numbers to guarantee that that a random permutation of weights look indistinguishable from Gaussian noise.

In conclusion, CEWT is most effective for large-scale Quantization-Aware Pre-Training (QAPT), or training models with **many low-precision parameters** for **a long duration** on a **large dataset**.

	$n =$	1	2	4	8	16	32	64	128	256	512	1024
MSE(XQ, XW^*) (unit: 1e-3)	STE	**1±0**	**1±0**	**1±0**	**2±0**	3±0	4±0	7±0	13±0	23±0	44±0	81±0
MSE(XQ, XW^*) (unit: 1e-3)	CEWT	**1±0**	**1±0**	**1±0**	**2±0**	**2±0**	**3±0**	**5±0**	**9±0**	**14±0**	**25±0**	**45±0**
RBM(vec($1/s \cdot W$)) (unit: 1e-3)	STE	11±0	12±0	15±0	19±0	27±0	40±0	64±0	108±1	186±0	332±1	573±1
RBM(vec($1/s \cdot W$)) (unit: 1e-3)	CEWT	**10±0**	**10±0**	**10±0**	**10±0**	**10±0**	**10±0**	**10±0**	**10±0**	**10±0**	**10±0**	**10±0**

Table 7: Rounding boundary mass and MSE loss for different values of n (dataset size).

	Iterations =	1	3	6	12	25	50	100	200
MSE(XQ, XW^*) (unit: 1e-3)	STE	1847±10	**1609±9**	**1277±7**	**747±5**	**244±1**	**90±0**	82±0	81±0
MSE(XQ, XW^*) (unit: 1e-3)	CEWT	**1839±9**	1625±9	1343±7	859±6	281±1	107±0	**63±0**	**45±0**
RBM(vec($1/s \cdot W$)) (unit: 1e-3)	STE	11±0	12±0	15±0	19±0	27±0	40±0	64±0	108±1
RBM(vec($1/s \cdot W$)) (unit: 1e-3)	CEWT	**10±0**	**10±0**	**10±0**	**10±0**	**10±0**	**10±0**	**10±0**	**10±0**

Table 8: Rounding boundary mass and MSE loss for different number of training iterations.

	$d =$	2	4	8	16	32	64	128	256	512	1024
MSE(XQ, XW^*) (unit: 1e-3)	STE	**398±452**	**258±352**	**130±47**	**137±38**	128±37	128±6	122±2	117±1	105±1	81±0
MSE(XQ, XW^*) (unit: 1e-3)	CEWT	1849±2115	1876±1619	313±281	117±41	**95±8**	**90±4**	**85±1**	**79±1**	**66±0**	**45±0**
RBM(vec($1/s \cdot W$)) (unit: 1e-3)	STE	600±671	775±244	756±114	753±52	741±40	733±21	712±5	693±2	661±1	573±1
RBM(vec($1/s \cdot W$)) (unit: 1e-3)	CEWT	**0±0**	**0±0**	**13±34**	**11±9**	**11±4**	**10±1**	**10±0**	**10±0**	**10±0**	**10±0**

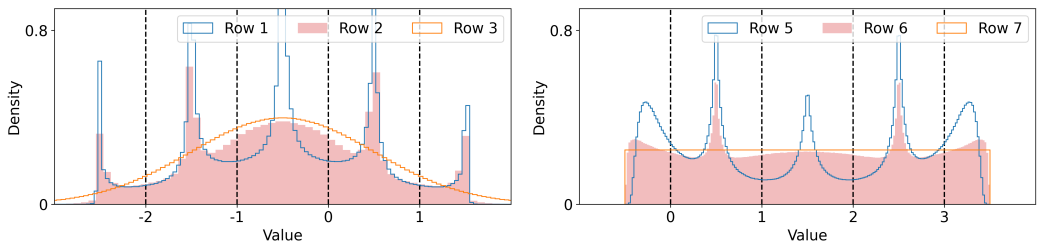
Table 9: Rounding boundary mass and MSE loss for different values of d (model size).

E Ablation Studies on Removing the Hadamard Transform and TGCS Quantization

In this section, we present our ablation studies in Table 10 and the effect of TGCS on weight distribution in Figure 6.

Row	Weight Quantizer			Optimizer	Perplexity	RBM	Notes
	Type	HT	TGCS				
1	QuEST	T	F	AdamH	32.42	0.079	Vanilla QuEST without CEWT
2	QuEST	F	F	Adam-CEWT	31.53	0.071	Introduce CEWT without TGCS and remove HT
3	QuEST	F	T	Adam-CEWT	**30.70**	**0.056**	Further Introduce TGCS (Intended usage of this work)
4	QuEST	T	T	Adam-CEWT-HT	30.81	**0.056**	Introduce HT to both weight quantizer and optimizer
5	BBQ	T	F	AdamH	31.61	0.029	Vanilla BBQ without CEWT
6	BBQ	F	F	Adam-CEWT	30.69	0.015	Introduce CEWT without TGCS and remove HT
7	BBQ	F	T	Adam-CEWT	**30.13**	**0.010**	Further Introduce TGCS (Intended usage of this work)
8	BBQ	T	T	Adam-CEWT-HT	30.29	**0.010**	Introduce HT to both weight quantizer and optimizer

Table 10: Evaluation Perplexity of HO, CEWT w.o TGCS, CEWT, and CEWT-HT.



(a) QuEST (b) BBQ

Figure 6: Pre-clip (QuEST) and pre-round (BBQ) weight distribution of models (rows) in Table 10.

F EXAMPLE HYPERSPHERE OPTIMIZERS

In this section, we show how a few example hypersphere optimizers can be integrated with CEWT. AdamH is a Frobenius-norm-hypersphere-optimization (Wen et al., 2026) variant of Adam (Kingma & Ba, 2017), and its algorithm (with optional CEWT integration) is in Algorithm 4. MuonH is a Frobenius-norm-hypersphere-optimization variant of Muon (Jordan, 2024), and its algorithm (with optional CEWT integration) is in Algorithm 5. LionH is a Frobenius-norm-hypersphere-optimization variant of Muon (Jordan, 2024), and its algorithm (with optional CEWT integration) is in Algorithm 5. SSO (Xie et al., 2026) is a spectral-norm hypersphere optimizer, and its algorithm (with optional CEWT integration) is shown in Algorithm 7. While conceptually SSO performs a **post-update** retraction back to the radius-R hypersphere, its implementation (Algorithm 7) uses a **pre-update** retraction step for efficiency purposes. To integrate CEWT with pre-update retraction, we add Line 17 to Algorithm 7. Line 17 normalizes $\hat{W}$ such that the Gaussianization does not drastically change the magnitude of weights.

Algorithm 4 AdamH with optional CEWT Integration for Matrix Weights $W^{(t-1)} \in \mathbb{R}^{m \times n}$

Adam (Lines 3 to 11) HO (Lines 2, 12, 13, 17, and 19) CEWT (Line 15)

1: **function** ADAMH($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, integrate_cewt = False)
2: $R = ||W^{(0)}||_F$
3: $M^{(0)} = 0$
4: $V^{(0)} = 0$
5: **for** $t = 1$ to T **do**
6: Denote $G^{(t)}$ as a stochastic gradient of loss with respect to $W^{(t-1)}$
7: $M^{(t)} = \beta_1 \cdot M^{(t-1)} + (1 - \beta_1) \cdot G^{(t)}$
8: $V^{(t)} = \beta_1 \cdot V^{(t-1)} + (1 - \beta_1) \cdot (G^{(t)} \odot G^{(t)})$ ▷ Element-wise multiplication ($\odot$)
9: $\hat{M}^{(t)} = 1/(1 - (\beta_1)^t) \cdot M^{(t)}$
10: $\hat{V}^{(t)} = 1/(1 - (\beta_2)^t) \cdot V^{(t)}$
11: $O^{(t)} = \hat{M}^{(t)} \oslash (\sqrt{\hat{V}^{(t)}} + \epsilon)$ ▷ Element-wise square root ($\sqrt{\cdot}$) and division ($\oslash$)
12: $U^{(t)} = 1/||O^{(t)}||_F \cdot O^{(t)}$ ▷ Ensures $||U^{(t)}||_F = 1$
13: $\bar{W}^{(t)} = W^{(t-1)} - \eta^{(t)} R U^{(t)}$ ▷ Equation 2a
14: **if** integrate_cewt **then**
15: $\hat{W}^{(t)} = \text{mat}\left(\Phi^{-1}\left(\frac{1}{mn} \cdot \left(\text{ranks}(\text{vec}(\bar{W}^{(t)})) + \frac{1}{2}\right)\right)\right)$ ▷ Equation 4
16: **else**
17: $\hat{W}^{(t)} = \bar{W}^{(t)}$ ▷ Equation 2b
18: **end if**
19: $W^{(t)} = \frac{R}{||\hat{W}^{(t)}||_F} \hat{W}^{(t)}$ ▷ Equation 2c
20: **end for**
21: **end function**

Algorithm 5 MuonH with optional CEWT Integration for Matrix Weights $W^{(t-1)} \in \mathbb{R}^{m \times n}$

Muon (Lines 3 to 11) HO (Lines 2, 12, 13, 17, and 19) CEWT (Line 15)

1: **function** MUONH($\mu = 0.95$, nesterov = True, integrate_cewt = False)
2: $R = ||W^{(0)}||_F$
3: $B^{(0)} = 0$
4: **for** $t = 1$ to T **do**
5: Denote $G^{(t)}$ as a stochastic gradient of loss with respect to $W^{(t-1)}$
6: $B^{(t)} = \mu \cdot B^{(t-1)} + G^{(t)}$
7: **if** nesterov **then**
8: $O^{(t)} = \text{NewtonShultz}(\mu \cdot B^{(t)} + G^{(t)})$
9: **else**
10: $O^{(t)} = \text{NewtonShultz}(B^{(t)})$
11: **end if**
12: $U^{(t)} = 1/||O^{(t)}||_F \cdot O^{(t)}$ ▷ Ensures $||U^{(t)}||_F = 1$
13: $\bar{W}^{(t)} = W^{(t-1)} - \eta^{(t)} R U^{(t)}$ ▷ Equation 2a
14: **if** integrate_cewt **then**
15: $\hat{W}^{(t)} = \text{mat}\left(\Phi^{-1}\left(\frac{1}{mn} \cdot \left(\text{ranks}(\text{vec}(\bar{W}^{(t)})) + \frac{1}{2}\right)\right)\right)$ ▷ Equation 4
16: **else**
17: $\hat{W}^{(t)} = \bar{W}^{(t)}$ ▷ Equation 2b
18: **end if**
19: $W^{(t)} = \frac{R}{||\hat{W}^{(t)}||_F} \hat{W}^{(t)}$ ▷ Equation 2c
20: **end for**
21: **end function**

Algorithm 6 LionH with optional CEWT Integration for Matrix Weights $W^{(t-1)} \in \mathbb{R}^{m \times n}$

Lion (Lines 3 to 7) HO (Lines 2, 8, 9, 13, and 15) CEWT (Line 11)

1: **function** LIONH($\beta_1 = 0.9$, $\beta_2 = 0.99$, integrate_cewt = False)
2: $R = ||W^{(0)}||_F$
3: $M^{(0)} = 0$
4: **for** $t = 1$ to T **do**
5: Denote $G^{(t)}$ as a stochastic gradient of loss with respect to $W^{(t-1)}$
6: $M^{(t)} = \beta_2 \cdot M^{(t-1)} + (1 - \beta_2) \cdot G^{(t)}$
7: $O^{(t)} = \text{sign}(\beta_1 \cdot M^{(t-1)} + (1 - \beta_1) \cdot G^{(t)})$ ▷ Element-wise sign function
8: $U^{(t)} = 1/||O^{(t)}||_F \cdot O^{(t)}$ ▷ Ensures $||U^{(t)}||_F = 1$
9: $\bar{W}^{(t)} = W^{(t-1)} - \eta^{(t)} R U^{(t)}$ ▷ Equation 2a
10: **if** integrate_cewt **then**
11: $\hat{W}^{(t)} = \text{mat}\left(\Phi^{-1}\left(\frac{1}{mn} \cdot \left(\text{ranks}(\text{vec}(\bar{W}^{(t)})) + \frac{1}{2}\right)\right)\right)$ ▷ Equation 4
12: **else**
13: $\hat{W}^{(t)} = \bar{W}^{(t)}$ ▷ Equation 2b
14: **end if**
15: $W^{(t)} = \frac{R}{||\hat{W}^{(t)}||_F} \hat{W}^{(t)}$ ▷ Equation 2c
16: **end for**
17: **end function**

Algorithm 7 SSO with optional CEWT Integration for Matrix Weights $\hat{\mathbf{W}}^{(\mathbf{t-1})} \in \mathbb{R}^{m \times n}$

Update Computation (Lines 3 to 13) HO (Lines 2, 14, 19, and 10) CEWT (Line 16 and 17)

1: **function** SSO($\beta = 0.9$, $\epsilon = 10^{-6}$, integrate_cewt = False)
2: $R = \sqrt{m/n}$
3: $M^{(0)} = 0$
4: **for** $t = 1$ to T **do**
5: Denote $G^{(t)}$ as a stochastic gradient of loss with respect to $\hat{\mathbf{W}}^{(\mathbf{t-1})}$
6: $M^{(t)} = \beta \cdot M^{(t-1)} + (1 - \beta) \cdot G^{(t)}$
7: $\hat{M}^{(t)} = 1/||M^{(t)}||_F \cdot M^{(t)}$
8: $\sigma^{(t-1)}, u^{(t)}, v^{(t)} = \text{PowerIteration}(\hat{W}^{(t-1)})$ ▷ $\sigma^{(t-1)} \approx ||\hat{W}^{(t-1)}||_2$
9: $\Theta^{(t)} = u^{(t)} v^{(t)}$
10: $W^{(t-1)} = \frac{R}{\sigma^{(t-1)}} \hat{W}^{(t-1)}$ ▷ Equation 2c implemented as pre-update retraction
11: Define $h(\lambda) = \langle \Theta^{(t)}, \text{msign}(\hat{M}^{(t)} + \lambda \Theta^{(t)}) \rangle$
12: $(\lambda^*)^{(t)} = \text{Bisection}(h, \text{tolerance} = \epsilon)$
13: $U^{(t)} = \text{msign}\left(\hat{M}^{(t)} + (\lambda^*)^{(t)} \Theta^{(t)}\right)$ ▷ Note: $||U^{(t)}||_2 = 1$
14: $\bar{W}^{(t)} = W^{(t-1)} - \eta^{(t)} R U^{(t)}$ ▷ Equation 2a
15: **if** integrate_cewt **then**
16: $\hat{W}^{(t)}_{\text{temp}} = \text{mat}\left(\Phi^{-1}\left(\frac{1}{mn} \cdot \left(\text{ranks}(\text{vec}(\bar{W}^{(t)})) + \frac{1}{2}\right)\right)\right)$ ▷ Equation 4
17: $\hat{W}^{(t)} = \frac{||\bar{W}^{(t)}||_F}{||\hat{W}^{(t)}_{\text{temp}}||_F} \cdot \hat{W}^{(t)}_{\text{temp}}$ ▷ Ensure $||\hat{W}^{(t)}||_F = ||\bar{W}^{(t)}||_F$
18: **else**
19: $\hat{W}^{(t)} = \bar{W}^{(t)}$ ▷ Equation 2b
20: **end if**
21: **end for**
22: **end function**

G EXAMPLE QUANTIZERS

In this section we present the algorithm of a few quantizers and how they can be integrated with CEWT (Hadamard Transform Removal and TGCS). BBQ (Yang & Aamodt, 2026) is a per-channel quantizer with Hadamard Transform, and its algorithm (with optional CEWT integration) is shown in Algorithm 8. QuEST (Panferov et al., 2025) is a per-channel quantizer with Hadamard Transform, and its algorithm (with optional CEWT integration) is shown in Algorithm 9. LSQ (Esser et al., 2020) is a per-tensor quantizer, and its algorithm is shown in Algorithm 10. LSQ does not require customization when integrated with CEWT.

Algorithm 8 BBQ with optional CEWT Integration for Matrix Weights $W^{(t)} \in \mathbb{R}^{m \times n}$

HT Removal (Line 26) TGCS (Line 32) Pre-clip and Pre-round Weights (Lines 11 and 12)

1:
2: **function** quant_scale($v \in \mathbb{R}^d$)
3: return $\sqrt{\frac{1}{d}\sum_{i=0}^{d-1}(v_i)^2}$
4: **end function**
5:
6: **function** dequant_scale($v \in \mathbb{R}^d$)
7: return $\zeta^*/(2^b) \cdot \sqrt{\frac{1}{d}\sum_{i=0}^{d-1}(v_i)^2}$ ▷ ζ^* is a constant defined in the original paper.
8: **end function**
9:
10: **function** cliproround($M \in \mathbb{R}^{m \times n}$)
11: preclip = M
12: preround = $2^b \cdot \Phi(\text{preclip}) - \frac{1}{2}$ ▷ Element-wise standard Gaussian CDF (Φ)
13: return $\lfloor \text{preround} \rceil - 2^{b-1} + \frac{1}{2}$ ▷ Element-wise round-to-nearest-integer ($\lfloor \cdot \rceil$)
14: **end function**
15:
16: **function** HT($M \in \mathbb{R}^{m \times n}$)
17: Denote 2^k as the Hadamard block size
18: Denote H_k as the Hadamard matrix with shape $2^k \times 2^k$
19: assert n % 2^k = 0
20: $\bar{M}$ = M.reshape($m \cdot n/2^k$, 2^k)
21: $\hat{M} = \bar{M} H_k$
22: return $\hat{M}$.reshape(m, n)
23: **end function**
24:
25: **function** BBQ($W \in \mathbb{R}^{m \times n}$, integrate_cewt = False)
26: **if** not integrate_cewt **then** ▷ See Section 3.2
27: W = HT(W)
28: **end if**
29: Denote $Q \in \mathbb{R}^{m \times n}$ as quantized weights
30: **for** $i = 0$ to $m - 1$ **do**
31: **if** integrate_cewt **then**
32: s_q = quant_scale(vec(W)) ▷ See Section 3.3
33: **else**
34: s_q = quant_scale(W_i)
35: **end if**
36: s_{di} = dequant_scale(W_i)
37: $Q_i = s_{di} \cdot \text{cliproround}(1/s_q \cdot W_i)$
38: **end for**
39: return Q
40: **end function**

Algorithm 9 QuEST with optional CEWT Integration for Matrix Weights $W^{(t)} \in \mathbb{R}^{m \times n}$

HT Removal (Line 26) TGCS (Line 32) Pre-clip and Pre-round Weights (Lines 11 and 12)

1:
2: **function** quant_scale($v \in \mathbb{R}^d$)
3: return $2\alpha^*/(2^b-1) \cdot \sqrt{\frac{1}{d}\sum_{i=0}^{d-1}(v_i)^2}$ ▷ α^* is a constant defined in the original paper.
4: **end function**
5:
6: **function** dequant_scale($v \in \mathbb{R}^d$)
7: return $2\alpha^*/(2^b-1) \cdot \sqrt{\frac{1}{d}\sum_{i=0}^{d-1}(v_i)^2}$
8: **end function**
9:
10: **function** cliproud($M \in \mathbb{R}^{m \times n}$)
11: preclip = $M - \frac{1}{2}$
12: preround = clip(preclip, $-2^{b-1}, 2^{b-1}-1$) ▷ Element-wise clipping
13: return $\lfloor \text{preround} \rceil + \frac{1}{2}$ ▷ Element-wise round-to-nearest-integer ($\lfloor \cdot \rceil$)
14: **end function**
15:
16: **function** HT($M \in \mathbb{R}^{m \times n}$)
17: Denote 2^k as the Hadamard block size
18: Denote H_k as the Hadamard matrix with shape $2^k \times 2^k$
19: assert n % 2^k = 0
20: $\bar{M}$ = M.reshape($m \cdot n/2^k$, 2^k)
21: $\hat{M} = \bar{M} H_k$
22: return $\hat{M}$.reshape(m, n)
23: **end function**
24:
25: **function** QuEST($W \in \mathbb{R}^{m \times n}$, integrate_cewt = False)
26: **if** not integrate_cewt **then** ▷ See Section 3.2
27: W = HT(W)
28: **end if**
29: Denote $Q \in \mathbb{R}^{m \times n}$ as quantized weights
30: **for** $i = 0$ to $m-1$ **do**
31: **if** integrate_cewt **then**
32: s_q = quant_scale(vec(W)) ▷ See Section 3.3
33: **else**
34: s_q = quant_scale(W_i)
35: **end if**
36: s_{di} = dequant_scale(W_i)
37: $Q_i = s_{di} \cdot$ cliproud($1/s_q \cdot W_i$)
38: **end for**
39: return Q
40: **end function**

Algorithm 10 LSQ for Matrix Weights $W^{(t)} \in \mathbb{R}^{m \times n}$

Pre-clip and Pre-round Weights (Lines 11 and 12)

1:
2: **function** quant_scale($v \in \mathbb{R}^d$)
3: return s ▷ s is a learnable parameter independent of v
4: **end function**
5:
6: **function** dequant_scale($v \in \mathbb{R}^d$)
7: return s ▷ s is a learnable parameter independent of v
8: **end function**
9:
10: **function** cliproundd($M \in \mathbb{R}^{m \times n}$)
11: preclip = $M - \frac{1}{2}$
12: preround = clip(preclip, $-2^{b-1}, 2^{b-1} - 1$) ▷ Element-wise clipping
13: return $\lfloor \text{preround} \rceil + \frac{1}{2}$ ▷ Element-wise round-to-nearest-integer ($\lfloor \cdot \rceil$)
14: **end function**
15:
16: **function** LSQ($W \in \mathbb{R}^{m \times n}$, integrate_cewt = False)
17: Denote $Q \in \mathbb{R}^{m \times n}$ as quantized weights
18: **for** $i = 0$ to $m - 1$ **do**
19: s_q = quant_scale(vec(W))
20: s_d = dequant_scale(vec(W))
21: $Q_i = s_d \cdot \text{cliproundd}(1/s_q \cdot W_i)$
22: **end for**
23: return Q
24: **end function**

H PERIODIC INTERPOLATION VS. DAMPENING LOSS VS. CEWT (DEEP NETWORKS)

Section 3.1 noted CEWT can be interpreted as a hyperparameter-free variant of Periodic Interpolation or Dampening loss. In Figure 7, we compare the STE, CEWT, and Periodic Interpolation by setting the period to 1 iteration and sweeping through the interpolation ratio α. Results suggest CEWT can achieve lower perplexity than Periodic Interpolation on 7 of the 8 chosen α's, while matching the performance of Periodic Interpolation with the best α. In Figure 8, we compare the STE, CEWT, and Dampening Loss by sweeping through the dampening hyperparameter α. Results suggest CEWT can achieve lower perplexity than Dampening Loss on all 8 of the chosen hyperparameters.

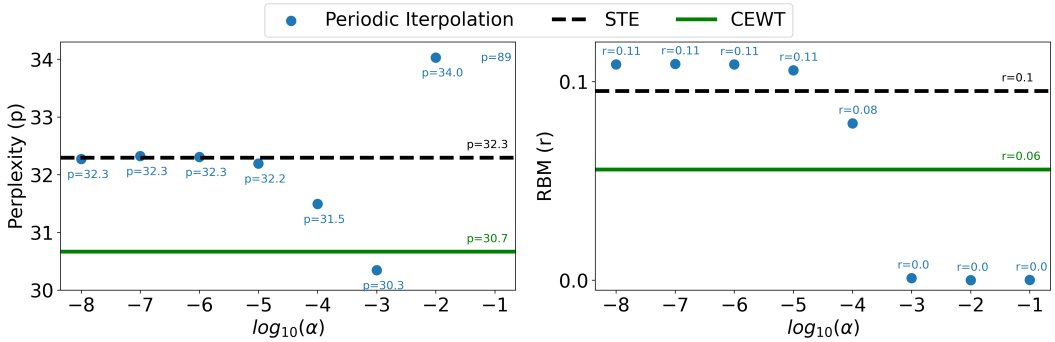


Figure 7: Comparing STE, CEWT, and Periodic Interpolation with a period of 1 and with grid search of the interpolation ratio hyperparameter α. Experiments are conducted on a LLaMA-95M with activations and weights quantized to 2-bit (with QuEST) and pre-trained on 3 billion C4 tokens with the AdamH/Adam-CEWT optimizer.

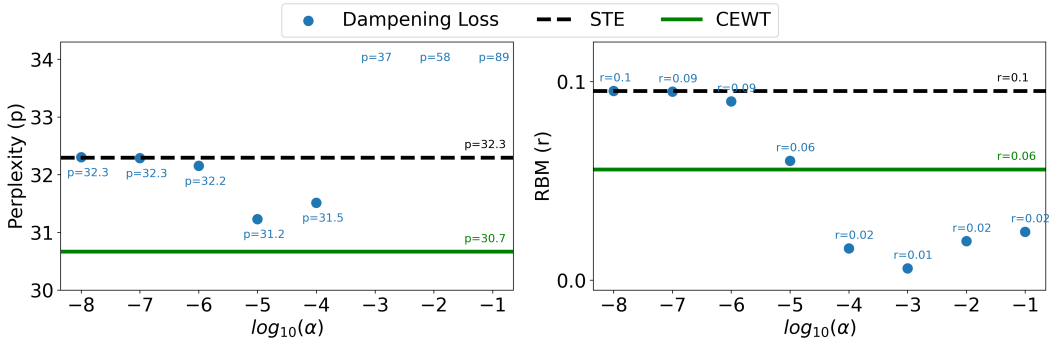


Figure 8: Comparing STE, CEWT, and Dampening Loss with a grid search of the dampening hyperparameter α. Experiments are conducted on a LLaMA-95M with activations and weights quantized to 2-bit (with QuEST) and pre-trained on 3 billion C4 tokens with the AdamH/Adam-CEWT optimizer.

I PSEUDO-QUANTIZATION NOISE VS. CEWT (DEEP NETWORKS)

In this section, we compare STE, PQN, and CEWT with a LLaMA-95M pre-trained on 3 billion C4 tokens. Table 11 presents our results. We find that while both PQN and CEWT are capable of suppressing rounding boundary weight oscillation, PQN cannot consistently outperform STE while CEWT can consistently outperform both the STE and PQN.

Quantizer	Bits W/A	AdamH + STE		AdamH + PQN		Adam-CEWT	
		PPL	RBM	PPL	RBM	PPL	RBM
QuEST	1	52.85	0.240	50.61	**0.002**	**48.02**	0.119
BBQ	1	51.06	0.034	49.54	**0.001**	**46.12**	0.010
QuEST	2	32.30	0.095	32.14	**0.007**	**30.67**	0.056
BBQ	2	31.80	0.023	31.75	**0.005**	**30.18**	0.010
QuEST	3	27.27	0.043	27.31	**0.010**	**26.49**	0.029
BBQ	3	27.35	0.017	27.64	**0.008**	**26.67**	0.010

Table 11: Comparing STE, PQN, and CEWT.

J FULL-PRECISION PERPLEXITY AND HYPERSPHERE LEARNING RATES

We tune our hypersphere optimizer learning rates by training a full-precision LLaMA-95M with the recipe used by Panferov et al. (2025), log the ratio $||\Delta W||/||W||$, and set the hypersphere learning rate to the largest observed ratio. For larger models, we half our hypersphere learning rate whenever Panferov et al. (2025) halves the learning rate. Table 12 presents the full-precision (without any quantization) perplexity of LLaMA models in Table 3 trained on AdamW and AdamH, along with the learning rate used for both full-precision and quantized models. For GPT model experiments, we reuse the learning rates in Table 12, which are tuned for LLaMA models.

Model	Size	Optimizer	LR	Bits W/A	Perplexity
LLaMA	95M	AdamW (wd=0.1)	0.0012	32/16	24.76
LLaMA	95M	AdamH	0.01	32/16	23.95
LLaMA	127M	AdamW (wd=0.1)	0.0012	32/16	21.53
LLaMA	127M	AdamH	0.01	32/16	20.86
LLaMA	206M	AdamW (wd=0.1)	0.0006	32/16	17.93
LLaMA	206M	AdamH	0.005	32/16	17.52
LLaMA	332M	AdamW (wd=0.1)	0.0003	32/16	-
LLaMA	332M	AdamH	0.0025	32/16	15.17
LLaMA	610M	AdamW (wd=0.1)	0.00015	32/16	-
LLaMA	610M	AdamH	0.00125	32/16	-

Table 12: Full-Precision Perplexity and learning rate of LLaMA models in Table 3. A "-" indicates an experiment omitted due to resource constraint.

K ZERO-SHOT EVALUATION

Table 13 presents our zero-shot results corresponding to the models in Table 3 with more than 200M parameters.

M-S	Quant-Bits	Optimizer	HSWAG (↑)	ARC-e (↑)	ARC-c (↑)	PiQA (↑)	Winog (↑)	Wiki (↓)	Pile 10k (↓)
L-206	None-16	AdamW	30.12	41.29	17.92	65.29	50.67	48.90	539.96
L-206	None-16	AdamH	30.55	43.98	18.17	64.69	49.80	47.20	538.48
L-332	None-16	AdamH	32.69	46.09	19.45	67.14	51.38	37.92	407.21
L-206	BBQ-1	MuonH	27.13	**32.70**	17.66	56.64	49.57	128.73	**2277.11**
L-206	BBQ-1	Muon-CEWT	**27.43**	32.24	**18.77**	**57.89**	**53.35**	**115.87**	2363.58
L-206	BBQ-1	AdamH	27.12	**33.16**	17.41	56.80	**50.67**	133.04	2362.53
L-206	BBQ-1	Adam-CEWT	**27.18**	33.12	**19.45**	**58.32**	50.51	**115.07**	**2043.74**
L-206	BBQ-1	LionH	26.91	33.00	**18.00**	57.45	49.33	128.38	2369.09
L-206	BBQ-1	Lion-CEWT	**27.21**	**33.50**	17.49	**57.51**	**52.88**	**113.15**	**2148.31**
L-206	BBQ-1	SSO	26.94	33.04	**19.03**	57.40	**52.25**	130.55	2297.06
L-206	BBQ-1	SSO-CEWT	**27.29**	**33.67**	17.83	**57.40**	51.07	**122.31**	**1961.63**
L-206	BBQ-2	MuonH	28.71	38.13	17.06	**62.95**	**51.14**	64.83	816.27
L-206	BBQ-2	Muon-CEWT	**28.76**	**40.40**	**17.83**	61.92	50.43	**63.18**	**802.27**
L-206	BBQ-2	AdamH	**28.78**	39.10	**19.71**	**61.53**	50.20	66.78	**988.03**
L-206	BBQ-2	Adam-CEWT	28.72	**40.03**	18.43	60.94	**51.93**	**63.38**	1045.41
L-206	BBQ-2	LionH	28.71	37.29	18.52	61.15	51.22	65.09	**1133.95**
L-206	BBQ-2	Lion-CEWT	**28.92**	**42.00**	**19.37**	**62.68**	**51.22**	**62.27**	1158.92
L-206	BBQ-2	SSO	28.56	**40.28**	**18.43**	**62.24**	50.20	67.33	876.57
L-206	BBQ-2	SSO-CEWT	**28.58**	38.05	18.17	60.94	**51.46**	**64.47**	**799.21**
L-332	BBQ-1	AdamH	27.33	34.60	**19.71**	58.54	**50.43**	103.60	1512.79
L-332	BBQ-1	Adam-CEWT	**27.44**	**35.10**	18.43	**59.52**	50.36	**93.12**	**1445.00**
L-610	BBQ-1	AdamH	**28.17**	37.50	**20.22**	59.85	**51.22**	79.84	1041.07
L-610	BBQ-1	Adam-CEWT	27.90	**37.67**	18.09	**60.72**	49.57	**73.60**	**1015.54**
G-204	BBQ-1	MuonH	**27.03**	33.08	18.94	57.40	49.01	147.80	2428.02
G-204	BBQ-1	Muon-CEWT	26.91	**33.33**	**19.20**	**57.73**	**50.83**	**134.10**	**2020.69**
G-204	BBQ-1	AdamH	**26.94**	**31.94**	**18.34**	55.88	49.57	155.48	2561.49
G-204	BBQ-1	Adam-CEWT	26.80	31.65	18.17	**57.62**	**49.96**	**139.19**	**2223.63**
G-204	BBQ-1	LionH	26.77	32.53	17.58	**57.18**	50.75	149.03	2497.89
G-204	BBQ-1	Lion-CEWT	**26.89**	**33.04**	**18.60**	56.09	**52.72**	**128.73**	**1997.89**
G-204	BBQ-1	SSO	**27.02**	**33.50**	18.26	**57.45**	**52.25**	141.21	2250.85
G-204	BBQ-1	SSO-CEWT	26.90	31.94	**18.34**	57.24	50.67	**130.86**	**1905.77**
G-204	BBQ-2	MuonH	28.22	35.98	**19.03**	60.17	49.25	72.07	916.26
G-204	BBQ-2	Muon-CEWT	**28.51**	**38.43**	16.98	**60.55**	**52.09**	**69.59**	**822.85**
G-204	BBQ-2	AdamH	28.01	37.84	17.32	**61.10**	**51.85**	74.73	947.88
G-204	BBQ-2	Adam-CEWT	**28.35**	**38.72**	**19.62**	61.04	51.22	**71.73**	**866.34**
G-204	BBQ-2	LionH	28.08	37.92	**19.97**	60.39	49.96	70.75	906.43
G-204	BBQ-2	Lion-CEWT	**28.38**	**38.26**	17.92	**61.32**	**51.85**	**67.67**	**831.54**
G-204	BBQ-2	SSO	28.28	**38.43**	17.66	60.50	**52.01**	72.15	983.45
G-204	BBQ-2	SSO-CEWT	**28.45**	38.22	**18.69**	**61.37**	49.88	**68.67**	**843.55**
G-325	BBQ-1	AdamH	26.78	33.33	**19.11**	**57.34**	50.75	123.77	1850.39
G-325	BBQ-1	Adam-CEWT	**26.91**	**33.54**	16.98	56.64	**51.78**	**110.03**	**1500.09**

Table 13: Zero-shot Evaluation of Models (from Table 3) Pre-trained on the C4 Dataset.

L TRAINING TIME

Table 14 presents the training time of our experiments in Table 3. We note that our experiments were conducted on shared hardware, and the execution time has a non-zero variance. On average (geometric mean), we find that CEWT results in a 4% longer training time.

M-S	Q-B	GPU	MuonH	Muon-CEWT	AdamH	Adam-CEWT	LionH	Lion-CEWT	SSO	SSO-CEWT
L-95	L-1	RTX 5090	**2.44**	2.45	**2.42**	2.43	**2.42**	2.43	**3.42**	3.42
L-95	Q-1	RTX 5090	**2.74**	2.85	**2.73**	2.84	**2.72**	2.83	**3.76**	3.83
L-95	B-1	RTX 5090	**2.76**	2.91	**2.74**	2.90	**2.74**	2.90	**3.76**	3.90
L-95	L-2	RTX 5090	**2.44**	2.46	**2.43**	2.44	**2.42**	2.43	**3.42**	3.42
L-95	Q-2	RTX 5090	**2.75**	2.86	**2.74**	2.84	**2.73**	2.85	**3.75**	3.85
L-95	B-2	RTX 5090	**2.77**	2.92	**2.76**	2.91	**2.75**	2.90	**3.75**	3.90
L-127	Q-1	RTX 5090	**6.04**	6.43	**6.00**	6.38	**5.99**	6.36	**8.87**	9.20
L-127	B-1	RTX 5090	**6.06**	6.55	**6.01**	6.50	**6.01**	6.49	**8.84**	9.31
L-127	Q-2	RTX 5090	**6.06**	6.43	**6.02**	6.38	**6.02**	6.37	**8.81**	9.22
L-127	B-2	RTX 5090	**6.09**	6.57	**6.05**	6.51	**6.04**	6.51	**8.82**	9.31
L-206	B-1	H100 80GB	**10.28**	10.76	**9.09**	10.32	13.20	**10.58**	**15.07**	15.20
L-206	B-2	H100 80GB	13.15	**12.82**	13.04	**10.68**	**10.77**	14.12	**14.99**	15.36
L-332	B-1	H100 80GB	-	-	40.53	**38.08**	-	-	-	-
L-610	B-1	H100 80GB	-	-	**130.52**	140.41	-	-	-	-
G-94	L-1	RTX 5090	**2.29**	2.31	**2.28**	2.29	**2.27**	2.29	**3.20**	3.21
G-94	Q-1	RTX 5090	**2.67**	2.76	**2.64**	2.75	**2.64**	2.74	**3.60**	3.70
G-94	B-1	RTX 5090	**2.69**	2.79	**2.66**	2.77	**2.66**	2.77	**3.64**	4.03
G-94	L-2	RTX 5090	**2.30**	2.31	**2.28**	2.29	**2.28**	2.29	**3.22**	3.22
G-94	Q-2	RTX 5090	**2.67**	2.77	**2.65**	2.75	**2.65**	2.75	**3.64**	3.74
G-94	B-2	RTX 5090	**2.69**	2.80	**2.68**	2.79	**2.67**	2.78	**3.65**	3.76
G-127	Q-1	RTX 5090	**5.89**	6.31	**5.83**	6.26	**5.84**	6.25	**8.05**	8.45
G-127	B-1	RTX 5090	**5.94**	6.26	**5.88**	6.22	**5.87**	6.21	**8.09**	8.42
G-127	Q-2	RTX 5090	**5.91**	6.32	**5.86**	6.27	**5.86**	6.26	**8.06**	8.50
G-127	B-2	RTX 5090	**5.96**	6.29	**5.91**	6.25	**5.91**	6.23	**8.08**	8.43
G-204	B-1	H100 80GB	10.43	**9.73**	**9.98**	10.43	9.66	**9.25**	**13.59**	19.75
G-204	B-2	H100 80GB	**8.77**	10.40	**8.76**	8.93	**9.52**	10.05	**13.73**	18.66
G-325	B-1	H100 80GB	-	-	38.12	**35.41**	-	-	-	-

Table 14: Training Time in Hours of experiments in Table 3.

M EVALUATION ON VISION MODELS

While our main empirical claims are on LLMs, we additional evaluate CEWT with the vision model DEIT-Base (Touvron et al., 2021) pre-trained on Imagenet-64x64 (Russakovsky et al., 2015; Chrabaszcz et al., 2017). Following Touvron et al. (2021), we use 16x16 pixels as a token, and therefore each 64x64 image is converted to a flattened sequence of 16 tokens. We use a batch size of 1024, a hypersphere learning rate of 0.004, and AdamH/Adam-CEWT to conduct our experiments.

We first run weight-only quantization experiments with IDs 1-8 in Table 15. Results confirm that rounding boundary weight oscillation can decrease training efficiency, and that CEWT can suppress weight oscillation and achieve lower loss.

We next run weight-activation quantization experiments with IDs 9-16 in Table 15. Results suggest that rounding boundary weight oscillation is less severe (when CEWT is not used) and therefore the loss reduction of CEWT is less consistent. We believe weight oscillation is less severe because activation quantization can reduce training efficiency (especially in a training recipe with heavy data augmentation) and delay the behaviour of weight oscillation to a much later stage in training. Table 16 shows that the rounding boundary mass consistently increases (throughout training) in

Exp ID	Model	Epochs	Quantizer	A/W	Optimizer	RBM	Train Loss	Test Accuracy
1	DEIT-Base	300	BBQ	16/2	AdamH	0.018	4.25	60.22
2	DEIT-Base	300	BBQ	16/2	Adam-CEWT	**0.010**	**4.15**	**61.04**
3	DEIT-Base	300	BBQ	16/1	AdamH	0.020	4.54	56.41
4	DEIT-Base	300	BBQ	16/1	Adam-CEWT	**0.010**	**4.39**	**58.88**
5	DEIT-Base	300	QuEST	16/2	AdamH	0.088	4.19	**61.32**
6	DEIT-Base	300	QuEST	16/2	Adam-CEWT	**0.056**	**4.16**	61.18
7	DEIT-Base	300	QuEST	16/1	AdamH	0.213	4.55	57.08
8	DEIT-Base	300	QuEST	16/1	Adam-CEWT	**0.119**	**4.41**	**58.70**
9	DEIT-Base	300	BBQ	2/2	AdamH	0.011	4.56	55.85
10	DEIT-Base	300	BBQ	2/2	Adam-CEWT	**0.010**	**4.53**	**56.60**
11	DEIT-Base	300	BBQ	1/1	AdamH	0.013	**5.34**	**39.80**
12	DEIT-Base	300	BBQ	1/1	Adam-CEWT	**0.010**	**5.34**	39.24
13	DEIT-Base	300	QuEST	2/2	AdamH	0.081	**4.57**	**56.51**
14	DEIT-Base	300	QuEST	2/2	Adam-CEWT	**0.056**	4.58	56.06
15	DEIT-Base	300	QuEST	1/1	AdamH	0.215	5.37	38.71
16	DEIT-Base	300	QuEST	1/1	Adam-CEWT	**0.119**	**5.35**	**39.39**
17	DEIT-Base	600	BBQ	2/2	AdamH	0.014	**4.38**	**58.19**
18	DEIT-Base	600	BBQ	2/2	Adam-CEWT	**0.010**	**4.38**	58.16
19	DEIT-Base	600	BBQ	1/1	AdamH	0.017	5.25	42.10
20	DEIT-Base	600	BBQ	1/1	Adam-CEWT	**0.010**	**5.20**	**42.92**
21	DEIT-Base	600	QuEST	2/2	AdamH	0.084	4.44	58.21
22	DEIT-Base	600	QuEST	2/2	Adam-CEWT	**0.056**	**4.41**	**58.49**
23	DEIT-Base	600	QuEST	1/1	AdamH	0.219	5.29	40.54
24	DEIT-Base	600	QuEST	1/1	Adam-CEWT	**0.119**	**5.26**	**41.53**

Table 15: Evaluation on Vision Models pre-trained on Imagenet 64x64.

weight-activation quantization experiments, but the rate of RBM increase is much slower compared to weight-only quantization experiments. Table 16 suggests that 300 epochs are not enough to train activation-weight-quantized DEIT-Base models to full convergence.

Exp ID	Model	Quantizer	A/W	Optimizer	Ep. 50	Ep. 100	Ep. 150	Ep. 200	Ep. 250	Ep. 299
1	DEIT-Base	BBQ	16/2	AdamH	0.008	0.010	0.012	0.013	0.012	0.018
9	DEIT-Base	BBQ	2/2	AdamH	**0.006**	**0.006**	**0.006**	**0.006**	**0.007**	**0.011**
2	DEIT-Base	BBQ	16/1	AdamH	0.005	0.008	0.009	0.009	0.010	0.020
11	DEIT-Base	BBQ	1/1	AdamH	**0.003**	**0.003**	**0.003**	**0.004**	**0.005**	**0.013**
3	DEIT-Base	QuEST	16/2	AdamH	0.077	0.079	0.079	0.079	0.079	0.088
13	DEIT-Base	QuEST	2/2	AdamH	**0.067**	**0.067**	**0.068**	**0.069**	**0.071**	**0.081**
4	DEIT-Base	QuEST	16/1	AdamH	0.193	0.198	0.200	0.201	0.203	**0.213**
15	DEIT-Base	QuEST	1/1	AdamH	**0.186**	**0.187**	**0.189**	**0.192**	**0.198**	0.215

Table 16: RBM of Vision Models at different epochs (Ep.). The experiment IDs in this table refers to the IDs from Table 15.

Therefore, we double the number of epochs (from 300 to 600) and run weight-activation quantization experiments with IDs 17-24 in Table 15. Results suggest that rounding boundary weight oscillation is more severe (when CEWT is not used) compared to experiments with 300 epochs, and CEWT can now consistently achieve lower loss, confirming our insight from Section D.9 and Table 8.

Overall, our results suggest that rounding boundary weight oscillation becomes more severe as training time increases, and therefore the loss reduction effect of CEWT becomes more consistent with longer training.