GWT: SCALABLE OPTIMIZER STATE COMPRESSION FOR LARGE LANGUAGE MODEL TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Training large language models requires substantial memory, particularly when using memory-intensive optimizers such as Adam. Existing memory-efficient optimization methods often rely on low-rank projections to reduce the memory overhead of optimizer states, but such approaches may discard information outside the selected subspace or introduce computationally expensive operations such as singular value decomposition (SVD). To address these limitations, we propose **Gradient Wavelet Transform (GWT)**, an SVD-free framework that applies a multilevel Haar transform to matrix gradients. GWT maintains first- and second-moment states only for the compact approximation coefficients, while retaining all detail coefficients transiently during each update. We show that the wavelet transform preserves the complete current gradient and establish gradient alignment and a stationary-point guarantee for smooth objectives under bounded preconditioning and a directly verifiable coarse-momentum coherence condition. Experiments on language-model pre-training and downstream fine-tuning show that GWT reduces estimated model-and-optimizer-state memory by up to 52% while maintaining competitive or improved model quality. On LLaMA-3B pre-training, GWT achieves $1.9\times$ the throughput of 8-bit Adam and is compatible with multiple optimizer families, including Adam, Adam-mini, and Muon.

1 INTRODUCTION

Training large language models (LLMs) requires substantial GPU memory. Adam-style optimizers (Kingma & Ba, 2014) have become the standard choice for LLM training because of their strong empirical performance. This comes at the cost of storing separate first- and second-moment estimates for every trainable parameter, resulting in optimizer states that can require roughly twice the memory of the model parameters. For example, under the BF16 configuration, pre-training a 7B-parameter model requires approximately 28 GB solely for Adam states (Zhao et al., 2024). This overhead has become a major bottleneck in scaling LLM training.

Existing memory-efficient training methods largely rely on low-rank representations. LoRA (Hu et al., 2022) reduces trainable parameters by constraining weight updates to low-rank factors, but this constraint can limit its applicability to pre-training and other optimization settings (Xia et al., 2024; Zhang et al., 2023). GaLore (Zhao et al., 2024) instead projects full gradients into low-dimensional subspaces and maintains optimizer states in the space. Subsequent methods compensate for projection residuals (Chen et al., 2024; Robert et al., 2025), improve robustness to noisy gradients (He et al., 2024), or reduce the cost of subspace construction (Zhu et al., 2024). These methods still compress the gradient into a global low-dimensional subspace, leaving components outside that subspace unused. This motivates investigating whether a localized, multiresolution representation can provide a different structure for optimizer-state compression.

In particular, wavelet transforms separate a signal into coarse approximation coefficients and localized detail coefficients while preserving the complete signal through exact reconstruction. This raises the following question:

Can the multiresolution structure of wavelets be used to compress optimizer states while preserving the complete instantaneous gradient?

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.

Wavelet transforms are well suited to this objective. They represent signals through localized components at multiple resolutions and admit linear-complexity transforms in the number of matrix elements (Rioul & Vetterli, 1991). When applied to matrix gradients, the coarse approximation coefficients capture block-wise aggregate structure, whereas the detail coefficients retain variations at finer scales. This decomposition makes it possible to maintain persistent states only for the compact approximation coefficients, while using the current detail coefficients directly during each update.

Motivated by this observation, we introduce **Gradient Wavelet Transform (GWT)**, a plug-and-play framework that applies a multilevel Haar transform to matrix gradients. GWT stores optimizer states only for the coarsest approximation coefficients and reconstructs each update using both the stateful approximation coefficients and the current detail coefficients. Unlike methods that retain only a projected gradient, GWT preserves all instantaneous Haar coefficients and compresses only the temporally accumulated optimizer states. It requires neither repeated SVD nor persistent projection matrices and can be integrated with multiple optimizer families. Empirically, we find that the compact approximation band captures substantially more gradient energy than its dimensional fraction, with this concentration weakened by random column permutation, indicating exploitable local gradient structure. Moreover, the reconstructed GWT direction remains positively aligned with the instantaneous gradient across all sampled cases, while the retained detail coefficients provide a substantial positive contribution to the update. In our experiments, GWT reduces estimated model-and-optimizer-state memory by up to 52% and achieves up to $1.9\times$ the throughput of 8-bit Adam on LLaMA-3B pre-training while maintaining competitive model quality.

Our main contributions are summarized as follows:

- We propose **Gradient Wavelet Transform (GWT)**, a wavelet-domain optimizer-state compression method that maintains optimizer states only for the approximation coefficients while retaining the instantaneous detail coefficients during reconstruction. GWT requires neither repeated SVD nor persistent projection matrices and can be used with different stateful optimizers.
- We analyze the approximation introduced by GWT. The Haar transform preserves the current gradient exactly, and local column variation determines the energy in the detail coefficients. We also characterize the error caused by maintaining momentum only in the approximation band. Under bounded preconditioning and a verifiable coherence condition on the coarse-scale momentum, the reconstructed update remains aligned with the gradient and converges to a stationary point. Experiments further verify the local gradient structure and update alignment considered in our analysis.
- We extensively evaluate GWT across language-model pre-training and multiple downstream tasks. GWT substantially reduces optimizer-state memory and improves training throughput while maintaining competitive model quality. It also remains effective across different decomposition levels, model scales, and optimizer families.

2 Related Work

Memory-Efficient Training and Optimization. Parameter-efficient methods reduce training memory by restricting the number or structure of trainable parameters. LoRA (Hu et al., 2022), for example, freezes pretrained weights and represents updates with trainable low-rank factors. Numerous extensions improve its optimization dynamics, expressiveness, or applicability to continued training (Hao et al., 2024; Lialin et al., 2023; Xia et al., 2024; Zhang et al., 2023; Liu et al., 2024; Luo et al., 2024; Qiu et al., 2026). For full-parameter training, GaLore (Zhao et al., 2024) projects gradients into low-dimensional subspaces and maintains optimizer states in the compressed space. Subsequent methods compensate for discarded information, reduce subspace-update overhead, or improve compressed optimization (Chen et al., 2024; Zhu et al., 2024; Robert et al., 2025; Rajabi et al., 2025; Wang et al., 2026). Another line of work reduces optimizer states through matrix orthogonalization or gradient normalization, including Muon (Liu et al., 2025), SWAN (Ma et al., 2024), SinkGD (Scetbon et al., 2025), and SCALE (Glentis et al., 2025). GWT differs from these methods by using a fixed multilevel transform rather than a data-dependent low-rank factorization or repeated SVD.

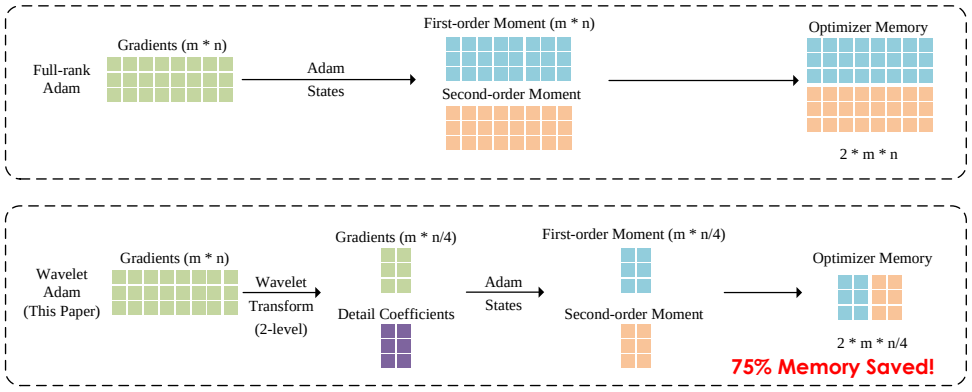


Figure 1: **Persistent optimizer-state memory under GWT.** For a matrix updated with GWT, a 2-level Haar transform stores Adam states for only 25% of the coefficients.

Transform-Domain Compression. Wavelet transforms provide localized multiresolution representations and have long been used for efficient signal and image compression (Porwik & Lisowska, 2004; Rioul & Vetterli, 1991). GWT transfers this principle to optimizer-state compression: the transform itself is lossless, but persistent adaptive states are allocated only to the approximation coefficients, while detail coefficients are retained transiently during reconstruction.

System-Level Memory Efficiency. System-level techniques target memory components beyond optimizer states. Activation checkpointing (Chen et al., 2016) reduces activation memory by recomputing intermediate values during the backward pass, whereas quantization (Dettmers et al., 2021; 2024) reduces storage or computation through lower-precision representations. These techniques address complementary memory sources and can therefore be combined with GWT.

3 Methodology

We first define the multilevel Haar representation used by GWT and then describe how adaptive optimizer states are maintained in the transformed domain. Throughout this section, t denotes the optimization iteration and j denotes the wavelet level. Implementation details, an eight-dimensional example, compact pseudocode, and the Norm-growth Limiter are provided in Appendix B.

3.1 Multilevel Haar Gradient Representation

Let $G_t \in \mathbb{R}^{m\times n}$ be the matrix gradient used at iteration t, where 2^l divides n, and set $A_t^{(0)} = G_t$. At level $j \in \{1,\dots,l\}$, a one-level orthonormal Haar transform is applied along the column dimension:

$$[A_t^{(j)}, D_t^{(j)}] = A_t^{(j-1)} H_{n/2^{j-1}},$$

where $A_t^{(j)}, D_t^{(j)} \in \mathbb{R}^{m\times n/2^j}$ are the approximation and detail coefficients, respectively. For an even dimension s, the Haar matrix $H_s \in \mathbb{R}^{s\times s}$ is orthogonal and satisfies

$$(H_s)_{2i-1,i} = (H_s)_{2i,i} = (H_s)_{2i-1,s/2+i} = \frac{1}{\sqrt{2}}, \qquad (H_s)_{2i,s/2+i} = -\frac{1}{\sqrt{2}}, \tag{1}$$

for $i = 1,\dots,s/2$, with all remaining entries equal to zero. The complete level-l transform is

$$\mathcal{H}_l(G_t) = \big(A_t^{(l)}, D_t^{(l)}, D_t^{(l-1)}, \dots, D_t^{(1)}\big), \tag{2}$$

and orthogonality gives the exact reconstruction

$$G_t = \mathcal{H}_l^{-1}\big(A_t^{(l)}, D_t^{(l)}, \dots, D_t^{(1)}\big). \tag{3}$$

It also gives the Parseval identity

$$\|G_t\|_F^2 = \|A_t^{(l)}\|_F^2 + \sum_{j=1}^{l} \|D_t^{(j)}\|_F^2. \tag{4}$$

Thus, the transform preserves all current gradient information, while the coarsest approximation contains only $mn/2^l$ entries. GWT uses this asymmetry to compress the persistent optimizer states rather than the current gradient.

3.2 Adam with GWT

For a matrix $W_t \in \mathbb{R}^{m\times n}$, let $\beta_1, \beta_2 \in [0, 1)$ and $\epsilon > 0$. Full Adam updates its first- and second-moment states using

$$M_{t+1} = \beta_1 M_t + (1-\beta_1)G_t, V_{t+1} = \beta_2 V_t + (1-\beta_2)G_t^2, \tag{5}$$

where the square is element-wise. These states contain $2mn$ persistent entries.

GWT instead maintains moments only for the coarsest approximation coefficients:

$$\begin{aligned} &M_{t+1}^A = \beta_1 M_t^A + (1-\beta_1)A_t^{(l)},\ V_{t+1}^A = \beta_2 V_t^A + (1-\beta_2)(A_t^{(l)})^2, \\ &\widehat{M}_{t+1}^A = \frac{M_{t+1}^A}{1-\beta_1^{t+1}}, \qquad \widehat{V}_{t+1}^A = \frac{V_{t+1}^A}{1-\beta_2^{t+1}}. \end{aligned} \tag{6}$$

Let $\mathcal{B}_{j,l}$ be the dyadic shape-alignment operator that expands a coarsest-level tensor to the shape of $D_t^{(j)}$ over the corresponding Haar supports. We define the positive element-wise preconditioners

$$S_{t+1}^A = \frac{1}{\sqrt{\widehat{V}_{t+1}^A} + \epsilon},\ S_{t+1}^{D,j} = \frac{1}{\sqrt{\mathcal{B}_{j,l}(\widehat{V}_{t+1}^A)} + \epsilon}, \qquad j = 1, \dots, l. \tag{7}$$

The transformed update coefficients are

$$\widetilde{A}_t^{(l)} = S_{t+1}^A \odot \widehat{M}_{t+1}^A,\ \widetilde{D}_t^{(j)} = S_{t+1}^{D,j} \odot D_t^{(j)}, \qquad j = 1, \dots, l, \tag{8}$$

where $\odot$ denotes element-wise multiplication. The GWT direction and parameter update are then

$$\widetilde{G}_t = \alpha\, \mathcal{H}_l^{-1}\big(\widetilde{A}_t^{(l)}, \widetilde{D}_t^{(l)}, \dots, \widetilde{D}_t^{(1)}\big),\ W_{t+1} = W_t - \eta \widetilde{G}_t, \tag{9}$$

where $\alpha > 0$ is the module-wise scale factor and $\eta > 0$ is the learning rate. The states M_t^A and V_t^A persist across iterations, whereas the detail coefficients are generated and consumed within the current iteration. All square roots, divisions, and powers in equation 6–equation 8 are element-wise.

For training stability, we additionally adopt the Norm-growth Limiter of Fira (Chen et al., 2024) with $\gamma = 1.01$. Its definition and ablation are deferred to Appendix B.3, since it is an auxiliary stabilization technique rather than a component specific to the wavelet transform.

3.3 Memory and Computational Complexity

Full Adam stores two persistent states of size mn. GWT stores two states of size $mn/2^l$, giving

$$\text{StateMem}_{\text{GWT}-l} = \frac{2mn}{2^l}, \qquad \text{Reduction} = 1 - \frac{1}{2^l}.$$

This is the reduction for a GWT-treated matrix; whole-model savings are smaller when embeddings, normalization parameters, or other modules continue to use full Adam. The detail coefficients do not contribute persistent optimizer-state memory, although the transform requires temporary coefficient buffers during the current iteration.

A multilevel Haar transform processes progressively shorter approximation bands. Its arithmetic cost is

$$\mathcal{O}\Big(mn + \frac{mn}{2} + \cdots + \frac{mn}{2^{l-1}}\Big) = \mathcal{O}(mn),$$

and the inverse transform has the same order. Updating the compressed moment states costs $\mathcal{O}(mn/2^l)$, while preconditioning the detail bands and reconstructing the update cost $\mathcal{O}(mn)$. Therefore, the overall GWT update remains linear in the number of matrix entries and requires neither SVD nor a persistent projection matrix.

Table 1: **Persistent optimizer-state memory and amortized per-step complexity for weight** $W \in \mathbb{R}^{m\times n}$, assuming $m \le n$. r is the projection rank.

Method	Optimizer states	Projection storage	Update complexity
Full Adam	$2mn$	0	$\mathcal{O}(mn)$
Muon (Liu et al., 2025)	mn	0	$\mathcal{O}(m^2n)$
GaLore (Zhao et al., 2024)	$2nr$	mr	$\mathcal{O}(m^2n)$
APOLLO (Zhu et al., 2024)	$2nr$	Seed $\mathcal{O}(1)$ or mr	$\mathcal{O}(mnr)$
GWT-l	$2mn/2^l$	0	$\mathcal{O}(mn)$

3.4 Theoretical Analysis

The analysis follows the structure of the algorithm. We first show that the Haar transform preserves the complete current gradient and quantify the energy of its detail coefficients. We then relate the approximation-state momentum error to temporal changes in the coarse coefficients. These two results identify the spatial and temporal structures used by GWT and lead to a descent and stationary-point convergence guarantee for the reconstructed direction.

Information preservation and detail energy. equation 3 shows that the Haar transform itself introduces no approximation error. GWT retains every current detail coefficient and reduces memory only by compressing the temporal optimizer states. Let $P_l(G)$ denote the reconstruction obtained by retaining $A^{(l)}$ and setting all detail bands to zero, and define

$$(\Delta G)_{:,k} = G_{:,k+1} - G_{:,k}, \qquad k = 1, \dots, n-1.$$

Lemma 3.1 (Detail-Energy Bound). *Let $b = 2^l$ divide n and define $\kappa_b = [2\sin(\pi/(2b))]^{-1}$. Then*

$$\sum_{j=1}^{l} \|D^{(j)}\|_F^2 = \|G - P_l(G)\|_F^2 \le \kappa_b^2 \|\Delta G\|_F^2. \tag{10}$$

The bound is not required for GWT to operate. It states that when adjacent columns vary slowly, a larger share of the gradient energy lies in the approximation coefficients. The remaining detail information is nevertheless retained in the reconstructed direction in equation 9.

Temporal coherence of the approximation state. The approximation coefficients are the only coefficients for which GWT stores a first-moment state. Define the coarse-momentum error

$$\Xi_t^A = \widehat{M}_{t+1}^A - A_t^{(l)}.$$

Lemma 3.2 (Temporal Form of the Coarse-Momentum Error). *Assume $M_0^A = 0$ and $0 \le \beta_1 < 1$. For $q = 0, \dots, t$ and $u = 0, \dots, t-1$, define*

$$\omega_{t,q} = \frac{(1-\beta_1)\beta_1^{t-q}}{1-\beta_1^{t+1}}, \qquad c_{t,u} = \frac{\beta_1^{t-u}(1-\beta_1^{u+1})}{1-\beta_1^{t+1}}.$$

Then

$$\widehat{M}_{t+1}^A = \sum_{q=0}^{t} \omega_{t,q} A_q^{(l)}, \; \Xi_t^A = -\sum_{u=0}^{t-1} c_{t,u}\big(A_{u+1}^{(l)} - A_u^{(l)}\big). \tag{11}$$

Consequently,

$$\left\|(S_{t+1}^A)^{1/2} \odot \Xi_t^A\right\|_F \le \sum_{u=0}^{t-1} c_{t,u} \left\|(S_{t+1}^A)^{1/2} \odot \big(A_{u+1}^{(l)} - A_u^{(l)}\big)\right\|_F. \tag{12}$$

For $t = 0$, both sides of equation 12 are zero.

Thus, the approximation-state error is small when the coarse coefficients vary gradually across nearby iterations. This relation also provides a direct empirical diagnostic for the assumption used below.

Assumption 3.3 (Bounded Preconditioning and Coarse-Momentum Coherence)**.** There exist constants $0 < s_{\min} \le s_{\max} < \infty$ and $0 \le \rho < 1$ such that, for every iteration $t \in \{0, \dots, T-1\}$ and level $j \in \{1, \dots, l\}$, each entry of S^A_{t+1} and $S^{D,j}_{t+1}$ lies in $[s_{\min}, s_{\max}]$, and

$$\left\| (S^A_{t+1})^{1/2} \odot \Xi^A_t \right\|_F \le \rho \left\| (S^A_{t+1})^{1/2} \odot A^{(l)}_t \right\|_F . \tag{13}$$

The preconditioner bounds control the effective coordinate-wise step sizes. Since $\epsilon > 0$, the entries are automatically bounded above by $1/\epsilon$; a positive lower bound follows whenever the second-moment estimates remain uniformly bounded. The coherence condition controls only the stateful approximation band and imposes no smallness condition on the detail coefficients.

Theorem 3.4 (Descent and Convergence of GWT)**.** *Let F be lower bounded by F_* and have an L-Lipschitz gradient with respect to the Frobenius norm. For $t = 0, \dots, T-1$, let $G_t = \nabla F(W_t)$ and construct $\widetilde{G}_t$ using equation 6–equation 9. Suppose Assumption 3.3 holds with the same constants over these iterations, and define*

$$\chi = \frac{s_{\max}}{s_{\min}}, \qquad \mu = \alpha(1-\rho)s_{\min}, \qquad C = \alpha s_{\max}(1 + \rho\sqrt{\chi}).$$

Then

$$\langle G_t, \widetilde{G}_t \rangle_F \ge \mu \|G_t\|_F^2, \qquad \|\widetilde{G}_t\|_F \le C \|G_t\|_F. \tag{14}$$

For a fixed learning rate satisfying $0 < \eta < 2\mu/(LC^2)$, the iterates in equation 9 further satisfy

$$\frac{1}{T} \sum_{t=0}^{T-1} \|\nabla F(W_t)\|_F^2 \le \frac{F(W_0) - F_*}{T\eta \left(\mu - \frac{L\eta C^2}{2}\right)}. \tag{15}$$

The theorem shows that state compression preserves a descent direction when the coarse momentum remains coherent. The current detail coefficients always contribute a positively preconditioned component, while the approximation band supplies the temporally accumulated state.

Scope of the analysis. Theorem 3.4 isolates the effect of GWT state compression under full-gradient updates. Extending the result to minibatch training would require controlling the conditional bias introduced by the interaction between stochastic gradients and the adaptive wavelet-domain preconditioner; unbiased minibatch gradients alone do not imply an unbiased GWT direction. We therefore do not impose an additional stochastic-direction assumption. Instead, the experiments can directly examine the coherence ratio in equation 13 during minibatch LLM training. The theorem analyzes the GWT direction before the optional Norm-growth Limiter. The limiter rescales the direction by a positive scalar and hence preserves its orientation, although its time-varying magnitude is not included in the stated rate. Complete proofs of Lemmas 3.1 and 3.2 and Theorem 3.4 are provided in Appendix C.

4 Experiments

In this section, we demonstrate that the Gradient Wavelet Transform (GWT), as a memory-efficient optimization technique, can achieve performance comparable to or even surpassing that of full-Adam optimizers, while significantly reducing memory usage and increasing training throughput. Specifically, we evaluate GWT in two contexts: pre-training LLaMA (Touvron et al., 2023a) models on the Colossal Clean Crawled Corpus (C4) (Raffel et al., 2019) English benchmark, fine-tuning various pre-trained models on the Multi-task Language Understanding (MMLU) (Hendrycks et al., 2020) and RoBERTa-base on the General Language Understanding Evaluation (GLUE) (Wang et al., 2018) benchmarks. We adopt the BF-16 format to reduce memory usage across all experiments and provide a detailed description of these datasets in the Appendix.

For GWT, we use the discrete Haar wavelet as the default filter. Additionally, we present an ablation study to examine the impact of key hyperparameters, including the scale factor (α), initial learning rate (lr), the GWT level (l), and generalization to models beyond LLaMA. A comprehensive description of the experimental setup can be found in the Appendix.

Table 2: Final validation perplexity (PPL, lower is better, 3 independent runs for 60M to 350M models) and estimated memory usage (BF16, model, and optimizer states) for pre-training LLaMA models on the C4 dataset.

Method	60M		130M		350M		1.3B	
	PPL	Mem.	PPL	Mem.	PPL	Mem.	PPL	Mem.
Full-Adam	$33.37_{\pm 0.04}$	0.34G	$25.08_{\pm 0.03}$	0.80G	$18.75_{\pm 0.00}$	2.20G	$16.10_{\pm 0.00}$	8.03G
Muon	$28.93_{\pm 0.03}$	0.30G	$23.05_{\pm 0.01}$	0.63G	$16.96_{\pm 0.00}$	1.60G	$14.28_{\pm 0.00}$	5.61G
Adam-Mini	$29.63_{\pm 0.05}$	0.22G	$23.73_{\pm 0.01}$	0.53G	$17.83_{\pm 0.01}$	1.46G	$15.10_{\pm 0.00}$	5.35G
LDAdamW	$29.27_{\pm 0.07}$	0.28G	$22.86_{\pm 0.02}$	0.57G	$17.53_{\pm 0.01}$	1.38G	$14.88_{\pm 0.00}$	4.76G
GaLore-1/4	$39.94_{\pm 0.07}$	0.28G	$26.47_{\pm 0.04}$	0.58G	$19.36_{\pm 0.02}$	1.38G	$15.66_{\pm 0.00}$	4.76G
APOLLO-1/4	$31.53_{\pm 0.12}$	0.28G	$23.35_{\pm 0.16}$	0.58G	$16.73_{\pm 0.05}$	1.38G	$14.10_{\pm 0.00}$	4.76G
GWT-2	$\mathbf{29.35}_{\pm 0.04}$	0.27G	$\mathbf{22.47}_{\pm 0.04}$	0.55G	$\mathbf{16.29}_{\pm 0.01}$	1.24G	$\mathbf{13.50}_{\pm 0.00}$	4.41G
GaLore-1/8	$48.48_{\pm 1.44}$	0.26G	$30.02_{\pm 0.79}$	0.53G	$21.59_{\pm 0.06}$	1.23G	$17.52_{\pm 0.00}$	4.15G
APOLLO-1/8	$32.50_{\pm 0.09}$	0.26G	$23.74_{\pm 0.17}$	0.53G	$16.98_{\pm 0.00}$	1.23G	$14.32_{\pm 0.00}$	4.15G
GWT-3	$\mathbf{29.81}_{\pm 0.09}$	0.26G	$\mathbf{22.63}_{\pm 0.04}$	0.51G	$\mathbf{16.35}_{\pm 0.02}$	1.09G	$\mathbf{13.48}_{\pm 0.00}$	3.80G
Training Tokens	1.3B		2.6B		7.8B		13.1B	

4.1 Memory-efficient Pre-Training

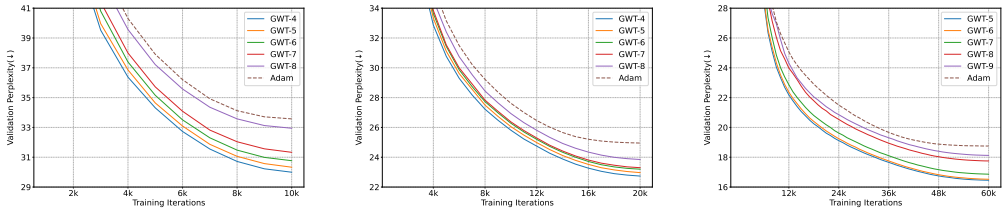
We integrate GWT with Adam (Kingma & Ba, 2014) and evaluate it on LLaMA (Touvron et al., 2023a) pre-training over C4. We compare against our reproduced results for full-Adam, Adam-mini (Zhang et al., 2024), LDAdamW (Robert et al., 2025), GaLore (Zhao et al., 2024), and APOLLO (Zhu et al., 2024). Unless otherwise specified, all experiments use cosine learning-rate decay, a sequence length of 256, and a batch size of 512, corresponding to 131K tokens per batch. GaLore and APOLLO are evaluated with ranks equal to one-fourth and one-eighth of the matrix dimension, denoted by GaLore/APOLLO-1/4 and GaLore/APOLLO-1/8. These compression ratios correspond to the state sizes of GWT-2 and GWT-3, respectively. Following prior work (Zhao et al., 2024; Zhu et al., 2024; Chen et al., 2024), compressed optimization is applied to the MLP and attention matrices, while the remaining parameters are updated with Adam.

GWT improves optimization efficiency. Table 2 shows that GWT achieves lower final PPL than full Adam and the matched-compression GaLore variants across the evaluated model scales. Figure 2 further shows that this behavior remains robust across different decomposition levels. Meanwhile, Table 3 shows that GWT retains high training throughput: GWT-2 achieves 0.532K tokens per second per GPU on LLaMA-3B, corresponding to a $1.9\times$ speedup over 8-bit Adam (Dettmers et al., 2021), about 6% higher throughput than GaLore-1/4, and performance comparable to the SVD-free APOLLO-1/4. These results show that GWT improves optimization while introducing limited computational overhead.

Table 3: **Pre-training LLaMA 3B on C4.** We report the validation PPL on different iterations and the token throughput per GPU.

	20K	**40K**	**60K**	**80K**	**100K**	**120K**	**Tokens/s**	**Time**	**Mem.**
8bit-Adam	21.79	17.44	15.70	14.99	14.52	14.31	0.274K	541.9h	10.7G
GaLore-1/4	20.42	17.36	15.98	15.12	14.79	14.73	0.526K	335.4h	9.28G
APOLLO-1/4	20.78	17.19	15.40	14.40	13.90	13.75	0.541K	274.4h	9.28G
GWT-2	19.74	16.41	14.80	13.77	13.46	**13.21**	0.532K	279.1h	8.54G
Tokens (B)	2.6	5.2	7.8	10.4	13.1	15.7			

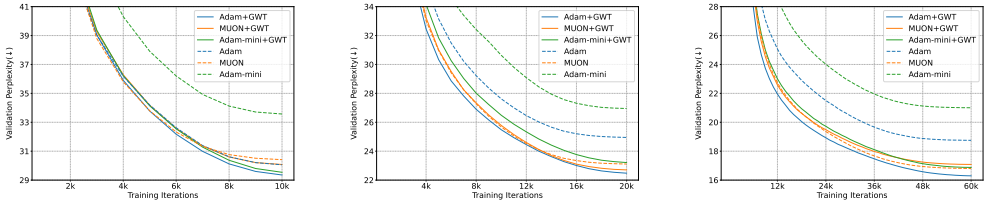
GWT remains efficient under SGD-level memory constraints. We assess the performance of GWT at higher decomposition levels, as illustrated in Figure 2. Notably, even at extreme levels (e.g., $l = 7, 8, 9$), where the memory footprint of the optimizer approaches that of SGD, GWT consistently outperforms full-Adam in terms of validation PPL, demonstrating its robustness and effectiveness in low-memory training regimes.



(a) LLaMA 60M (b) LLaMA 130M (c) LLaMA 350M

Figure 2: **Applying GWT to Adam for pre-training LLaMA models with reduced memory usage (higher l).** GWT maintains superior performance over Adam under high transform l levels.

GWT is optimizer-agnostic. We extend our method to the Adam-mini (Zhang et al., 2024) and Muon (Liu et al., 2025) optimizers and present the corresponding experimental results in Figure 3. Similar to its performance with Adam, GWT continues to achieve results comparable to or better than the tested methods. This highlights the versatility and effectiveness of our method when integrated with other optimizers, showcasing its broad applicability in memory-efficient optimization.



(a) LLaMA 60M (b) LLaMA 130M (c) LLaMA 350M

Figure 3: **Applying GWT to Adam/Adam-mini/Muon optimizers for pre-training various LLaMA models.** GWT is scalable to optimizers beyond Adam and consistently achieves lower or comparable PPL than full-Adam methods across all tested optimizers.

Pre-training with longer sequences. We evaluate GWT in handling longer sequences (512, 1024, 2048) while keeping the total number of training tokens per batch constant. Table 4 reports the final validation PPL across different methods. In contrast to GaLore, which exhibits noticeable performance degradation as sequence length increases, GWT maintains stable performance, showing only a slight increase in PPL. This resilience highlights GWT as a promising candidate for partial LLM pre-training scenarios, i.e., long-context windows and trillions of training tokens.

Table 4: **Final validation PPL on pre-training LLaMA models with different sequence length.** GWT consistently achieves the lowest validation perplexity across all model sizes and sequence lengths, demonstrating robust performance under longer-context training.

Model	60M			130M			350M		
Sequence length	512	1024	2048	512	1024	2048	512	1024	2048
full-Adam	34.55	37.52	39.77	25.95	28.12	32.01	19.95	22.02	24.71
GaLore-1/4	40.25	42.02	46.31	27.19	33.97	37.29	19.92	21.73	24.10
APOLLO-1/4	32.29	34.64	37.12	24.02	25.93	28.22	17.26	18.77	19.90
GWT-2	**30.12**	**32.55**	**35.84**	**23.01**	**24.89**	**26.91**	**16.68**	**18.21**	**19.30**

4.2 Memory-Efficient Fine-Tuning

In this section, we further evaluate the effectiveness of our method by fine-tuning the LLaMA-3.2-3B (Grattafiori et al., 2024), Gemma3-4B (Team et al., 2025), and Qwen2.5-7B (Yang et al., 2024)

Table 5: **Comparison of different fine-tuning methods on MMLU.** GWT achieves competitive performance across all three models.

Model	Method	STEM	Social Sciences	Humanities	Other	Avg.
LLaMA3.2-3B	Adam	49.50	68.25	56.28	64.74	**59.40**
	LoRA	49.83	67.63	56.56	63.88	59.23
	GaLore	49.73	68.70	55.94	64.47	59.37
	APOLLO	49.17	68.18	56.32	64.71	59.32
	GWT	49.50	68.61	56.56	64.62	59.34
Gemma3-4B	Adam	46.59	64.74	49.71	58.91	54.46
	LoRA	45.36	63.41	48.63	58.36	53.41
	GaLore	46.36	64.51	49.42	58.91	54.26
	APOLLO	46.49	64.28	49.39	58.67	54.17
	GWT	46.39	64.87	49.73	59.38	**54.56**
Qwen2.5-7B	Adam	-	-	-	-	OOM
	LoRA	70.44	83.23	67.80	76.90	73.85
	GaLore	70.21	83.49	67.52	76.87	73.76
	APOLLO	70.51	83.43	67.35	76.96	73.77
	GWT	70.38	83.88	68.06	77.14	**74.12**

models, and report their performance on the MMLU benchmark (Hendrycks et al., 2020). For a fair comparison, LoRA, GaLore, and APOLLO are configured with a rank of 8, while GWT is set to a decomposition level of 8 to ensure memory alignment. We report the best accuracy achieved by sweeping the learning rate among [1.0e-6, 2.5e-6, 5.0e-6, 1.0e-5, 2.5e-5, 5.0e-5, 1.0e-4], which ensures a fair evaluation for optimizers with different sensitivities to learning rates. Additionally, to provide a more comprehensive assessment, we include results from fine-tuning the RoBERTa-base model (Liu, 2019) on the GLUE (Wang et al., 2018) benchmarks.

We report the MMLU results in Table 5 and the GLUE results in Table 6. As shown, GWT consistently matches or exceeds the performance of the baselines. These results demonstrate that GWT is not only effective during pre-training but also excels in fine-tuning scenarios, achieving competitive performance across diverse tasks while maintaining memory efficiency.

Table 6: **Fine-tuning RoBERTa-Base on GLUE benchmarks.**

	CoLA	STS-B	MRPC	RTE	SST2	MNLI	QNLI	QQP	Avg.
Adam	**63.81**	**91.26**	92.25	79.06	**94.57**	87.29	92.18	**92.28**	**86.58**
GaLore	61.32	91.13	92.41	77.25	94.03	86.77	**92.56**	91.77	85.90
APOLLO	61.07	90.70	92.00	78.33	93.57	87.21	92.27	91.85	85.87
LoRA* (Zhao et al., 2024)	61.83	90.80	91.90	79.06	93.46	86.94	92.25	91.22	85.93
GWT	62.57	91.16	**93.26**	**79.42**	94.26	**87.37**	92.53	91.94	86.56

5 Conclusion

In this paper, we introduce GWT, an SVD-free framework that compresses optimizer states in a multilevel Haar domain while retaining all current gradient coefficients in the reconstructed update. Our theoretical and empirical analyses characterize why this design remains effective: the Haar representation captures structured gradient information, while the retained detail coefficients preserve complementary instantaneous information and help maintain gradient-aligned updates. Under bounded preconditioning and coarse-momentum coherence, GWT further admits a convergence guarantee. Across language-model pre-training and fine-tuning, GWT reduces memory and improves training throughput while maintaining competitive model quality. Its compatibility with Adam, Adam-mini, and Muon further demonstrates the potential of wavelet-domain state compression as a general mechanism for memory-efficient optimization.

AI USE STATEMENT

In this work, we used generative AI tools to edit the manuscript for clarity, conciseness, grammar, and English-language presentation. We have not used generative AI tools for other research tasks requiring disclosure, and the remaining disclosure categories are not applicable to this work. We have reviewed all AI-assisted edits and manually verified the final manuscript. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

REPRODUCIBILITY STATEMENT

We provide the information required to reproduce the proposed method and experiments throughout the paper and appendix. The assumptions and complete derivations supporting the theoretical results are also provided in the appendix. We will include the implementation and configuration files in the anonymized supplementary material to facilitate reproduction of the reported results.

REFERENCES

Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016.

Xi Chen, Kaituo Feng, Changsheng Li, Xunhao Lai, Xiangyu Yue, Ye Yuan, and Guoren Wang. Fira: Can we achieve full-rank training of llms under low-rank constraint? *ArXiv*, abs/2410.01623, 2024.

Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 8-bit optimizers via block-wise quantization. *arXiv preprint arXiv:2110.02861*, 2021.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36, 2024.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.

Katie Everett, Lechao Xiao, Mitchell Wortsman, Alex Alemi, Roman Novak, Peter J. Liu, Izzeddin Gur, Jascha Narain Sohl-Dickstein, Leslie Pack Kaelbling, Jaehoon Lee, and Jeffrey Pennington. Scaling exponents across parameterizations and optimizers. *ArXiv*, abs/2407.05872, 2024.

R'emi Genet and Hugo Inzirillo. Caadam: Improving adam optimizer using connection aware methods. *ArXiv*, abs/2410.24216, 2024.

Athanasios Glentis, Jiaxiang Li, Andi Han, and Mingyi Hong. A minimalist optimizer design for LLM pretraining. In *ES-FoMo III: 3rd Workshop on Efficient Systems for Foundation Models*, 2025.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Yongchang Hao, Yanshuai Cao, and Lili Mou. Flora: Low-rank adapters are secretly gradient compressors. *arXiv preprint arXiv:2402.03293*, 2024.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention, 2021.

Yutong He, Pengrui Li, Yipeng Hu, Chuyan Chen, and Kun Yuan. Subspace optimization for large language models with convergence guarantees. *arXiv preprint arXiv:2410.11289*, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

Vladislav Lialin, Namrata Shivagunde, Sherin Muckatira, and Anna Rumshisky. Relora: High-rank training through low-rank updates, 2023. *URL https://arxiv. org/abs/2307.05695*, 2023.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, et al. Muon is scalable for llm training. *arXiv preprint arXiv:2502.16982*, 2025.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation. In *Forty-first International Conference on Machine Learning*, 2024.

Yinhan Liu. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 364, 2019.

Qijun Luo, Hengxu Yu, and Xiao Li. Badam: A memory efficient full parameter optimization method for large language models. *Advances in Neural Information Processing Systems*, 37: 24926–24958, 2024.

Chao Ma, Wenbo Gong, Meyer Scetbon, and Edward Meeds. Swan: Sgd with normalization and whitening enables stateless llm training. *ArXiv*, abs/2412.13148, 2024.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Piotr Porwik and Agnieszka Lisowska. The haar-wavelet transform in digital image processing: Its status and achievements. *Machine graphics & vision*, 13:79–98, 2004.

Zeju Qiu, Lixin Liu, Adrian Weller, Han Shi, and Weiyang Liu. Poet-x: Memory-efficient llm training by scaling orthogonal transformation. *arXiv preprint arXiv:2603.05500*, 2026.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Colin Raffel, Noam M. Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2019.

Sahar Rajabi, Nayeema Nonta, and Sirisha Rambhatla. Subtrack++ : Gradient subspace tracking for scalable llm training. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, pp. 31584–31613. Curran Associates, Inc., 2025. doi: 10.52202/085713-1060. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/2d62cb71e87ae340e3ab0e874befcbc2-Paper-Conference.pdf.

O. Rioul and M. Vetterli. Wavelets and signal processing. *IEEE Signal Processing Magazine*, 8(4): 14–38, 1991. doi: 10.1109/79.91217.

Thomas Robert, Mher Safaryan, Ionut-Vlad Modoranu, and Dan Alistarh. LDAdam: Adaptive optimization from low-dimensional gradient statistics. In *The Thirteenth International Conference on Learning Representations*, 2025.

Meyer Scetbon, Chao Ma, Wenbo Gong, and Edward Meeds. Gradient multi-normalization for stateless and scalable llm training. *arXiv preprint arXiv:2502.06742*, 2025.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

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

Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, and et al Amjad Almahairi. Llama 2: Open foundation and fine-tuned chat models. *ArXiv*, abs/2307.09288, 2023b.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP workshop BlackboxNLP: Analyzing and interpreting neural networks for NLP*, pp. 353–355, 2018.

Zhengbo Wang, Jian Liang, Ran He, Zilei Wang, and Tieniu Tan. Taming momentum: Rethinking optimizer states through low-rank approximation. *arXiv preprint arXiv:2602.24283*, 2026.

Moritz Wolter, Felix Blanke, Jochen Garcke, and Charles Tapley Hoyt. ptwt - the pytorch wavelet toolbox. *Journal of Machine Learning Research*, 25(80):1–7, 2024.

Wenhan Xia, Chengwei Qin, and Elad Hazan. Chain of lora: Efficient fine-tuning of language models via residual learning. *arXiv preprint arXiv:2401.04151*, 2024.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Longteng Zhang, Lin Zhang, Shaohuai Shi, Xiaowen Chu, and Bo Li. Lora-fa: Memory-efficient low-rank adaptation for large language models fine-tuning. *arXiv preprint arXiv:2308.03303*, 2023.

Yushun Zhang, Congliang Chen, Ziniu Li, Tian Ding, Chenwei Wu, Diederik P Kingma, Yinyu Ye, Zhi-Quan Luo, and Ruoyu Sun. Adam-mini: Use fewer learning rates to gain more. *arXiv preprint arXiv:2406.16793*, 2024.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. Galore: Memory-efficient llm training by gradient low-rank projection. *arXiv preprint arXiv:2403.03507*, 2024.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. Llamafactory: Unified efficient fine-tuning of 100+ language models. *arXiv preprint arXiv:2403.13372*, 2024.

Hanqing Zhu, Zhenyu Zhang, Wenyan Cong, Xi Liu, Sem Park, Vikas Chandra, Bo Long, David Z. Pan, Zhangyang Wang, and Jinwon Lee. Apollo: Sgd-like memory, adamw-level performance, 2024.

APPENDIX CONTENTS

Appendix Contents **14**

A Notation **15**

B Haar Transform and Stabilization Details **16**

B.1 Eight-Dimensional Haar Example 16

B.2 Compact GWT Pseudocode 16

B.3 Norm-Growth Limiter 17

C Proofs for GWT Analysis **17**

C.1 Haar Detail Energy 17

C.2 Temporal Coherence of the Approximation State 19

C.3 Deterministic Descent and Convergence 19

D Experiment Details **21**

D.1 Network Architecture 21

D.2 Experiment Hyperparameters 21

D.3 Experimental Enviroments 23

E Additional Experimental Results **23**

E.1 Empirical Validation of Coarse-Momentum Coherence 23

E.2 Understanding GWT: Energy Concentration and Directional Alignment 26

E.3 Study the Effect of α 28

E.4 Extension to Other Model Families 28

E.5 Memory and Throughput Estimates 29

E.6 Adam with Module-wise Learning Rate 30

F Discussion **31**

A NOTATION

Symbol	Definition
t, T	Optimization iteration and total number of analyzed iterations; W_t is the parameter matrix before processing G_t.
j, l	Haar level and total decomposition level, respectively, with $j \in \{1, \dots, l\}$.
m, n	Row and column dimensions of a matrix parameter and its gradient.
$F(W)$	Deterministic objective used in the convergence analysis; F_* is a finite lower bound of F.
L	Lipschitz constant of ∇F with respect to the Frobenius norm.
G_t	Matrix gradient used at iteration t; in Theorem 3.4, $G_t = \nabla F(W_t)$.
H_s	One-level orthonormal Haar matrix for an even dimension s.
$\mathcal{H}_l, \mathcal{H}_l^{-1}$	Level-l Haar transform and its inverse.
$A_t^{(j)}, D_t^{(j)}$	Approximation and detail coefficients at Haar level j obtained from G_t.
M_t^A, V_t^A	Persistent first- and second-moment states for the coarsest approximation coefficients before iteration t.
$\widehat{M}_{t+1}^A, \widehat{V}_{t+1}^A$	Bias-corrected moment states after incorporating $A_t^{(l)}$.
$\mathcal{B}_{j,l}$	Dyadic shape-alignment operator that expands a coarsest-level tensor to the shape of $D_t^{(j)}$.
$S_{t+1}^A, S_{t+1}^{D,j}$	Positive element-wise preconditioners for the approximation and the level-j detail coefficients.
$\widetilde{A}_t^{(l)}, \widetilde{D}_t^{(j)}$	Preconditioned approximation and detail update coefficients at iteration t.
Ξ_t^A	Coarse-momentum error, $\Xi_t^A = \widehat{M}_{t+1}^A - A_t^{(l)}$.
$\omega_{t,q}, c_{t,u}$	Nonnegative temporal weights in Lemma 3.2.
$\widetilde{G}_t$	Reconstructed GWT update direction before the optional Norm-growth Limiter.
α, η	Module-wise scale factor and fixed learning rate.
$\beta_1, \beta_2, \epsilon$	Moment decay rates and numerical-stability constant.
$P_l(G)$	Haar low-pass reconstruction obtained by retaining the level-l approximation and setting all detail coefficients to zero.
ΔG	Adjacent-column difference matrix, $(\Delta G)_{:,k} = G_{:,k+1} - G_{:,k}$.
b, κ_b	Haar block length $b = 2^l$ and the discrete Poincaré constant $\kappa_b = [2\sin(\pi/(2b))]^{-1}$.
$s_{\min}, s_{\max}$	Uniform lower and upper bounds on the entries of the GWT preconditioners.
ρ	Coarse-momentum mismatch parameter in Assumption 3.3.
χ	Preconditioner condition ratio, $\chi = s_{\max}/s_{\min}$.
μ, C	Uniform alignment and direction-norm constants in Theorem 3.4.
$\langle \cdot, \cdot \rangle_F, \|\cdot\|_F$	Frobenius inner product and norm.

Algorithm 1 Adam with Gradient Wavelet Transform

Input: Initial weight W_0, learning rate η, decay rates β_1, β_2, scale α, level l, and $\epsilon > 0$
Initialize $M_0^A \leftarrow 0$ and $V_0^A \leftarrow 0$
for $t = 0, \dots, T-1$ **do**
  $G_t \leftarrow \nabla_W f(W_t)$
  $(A_t^{(l)}, D_t^{(l)}, \dots, D_t^{(1)}) \leftarrow \mathcal{H}_l(G_t)$
  Update M_{t+1}^A, V_{t+1}^A using equation 6
  Compute $\widehat{M}_{t+1}^A, \widehat{V}_{t+1}^A$ and the preconditioners in equation 7
  $\widetilde{A}_t^{(l)} \leftarrow S_{t+1}^A \odot \widehat{M}_{t+1}^A$
  for $j = 1, \dots, l$ **do**
    $\widetilde{D}_t^{(j)} \leftarrow S_{t+1}^{D,j} \odot D_t^{(j)}$
  end for
  $\widetilde{G}_t \leftarrow \alpha \mathcal{H}_l^{-1}(\widetilde{A}_t^{(l)}, \widetilde{D}_t^{(l)}, \dots, \widetilde{D}_t^{(1)})$
  If $t \geq 1$, optionally apply the Norm-growth Limiter in equation 17
  $W_{t+1} \leftarrow W_t - \eta \widetilde{G}_t$
end for
return W_T

B Haar Transform and Stabilization Details

B.1 Eight-Dimensional Haar Example

For an input vector $x = [x_1, \dots, x_8]$, the first-level approximation and detail coefficients are

$$\begin{aligned} A^{(1)} &= \frac{1}{\sqrt{2}}[x_1 + x_2,\ x_3 + x_4,\ x_5 + x_6,\ x_7 + x_8], \\ D^{(1)} &= \frac{1}{\sqrt{2}}[x_1 - x_2,\ x_3 - x_4,\ x_5 - x_6,\ x_7 - x_8]. \end{aligned} \tag{16}$$

Applying the transform recursively to $A^{(1)}$ gives

$$\begin{aligned} A^{(2)} &= \frac{1}{2}[x_1 + x_2 + x_3 + x_4,\ x_5 + x_6 + x_7 + x_8], \\ D^{(2)} &= \frac{1}{2}[x_1 + x_2 - x_3 - x_4,\ x_5 + x_6 - x_7 - x_8]. \end{aligned}$$

The complete coefficient set $(A^{(2)}, D^{(2)}, D^{(1)})$ contains the same number of entries as x and reconstructs it exactly. GWT stores persistent optimizer states only for $A^{(2)}$, which contains one quarter of the original entries; the detail coefficients remain temporary.

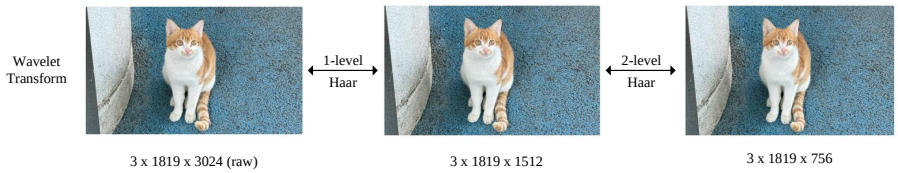


Figure 4: **Illustration of 2-level Haar approximation coefficients on an image.** The example is included only to visualize the multiresolution decomposition; the empirical gradient-structure diagnostic is discussed separately.

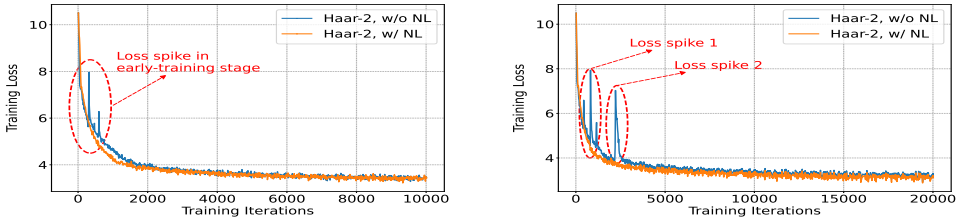
B.2 Compact GWT Pseudocode

B.3 Norm-Growth Limiter

The Norm-growth Limiter (NL) (Chen et al., 2024) controls abrupt changes in the norm of the reconstructed update. For $t \geq 1$ and threshold $\gamma > 1$, it applies

$$\text{if} \quad \frac{\|\widetilde{G}_t\|_F}{\|\widetilde{G}_{t-1}\|_F} > \gamma, \quad \text{then} \quad \widetilde{G}_t \leftarrow \frac{\widetilde{G}_t}{\|\widetilde{G}_t\|_F}\gamma\|\widetilde{G}_{t-1}\|_F. \tag{17}$$

We use $\gamma = 1.01$ by default. As shown in Figure 5, NL suppresses the loss spikes observed in early training. Because NL acts after gradient reconstruction and is not specific to the Haar transform, we treat it as an auxiliary stabilization technique.



(a) LLaMA 60M (b) LLaMA 130M

Figure 5: **Training loss of GWT-2 with and without the Norm-growth Limiter on LLaMA 60M and 130M.**

C Proofs for GWT Analysis

This section proves the detail-energy bound, the temporal identity for the approximation-state momentum, and the descent and convergence result stated in Section 3.4. The proofs follow the same order as the main text: spatial structure of the Haar coefficients, temporal structure of the compressed state, and finally descent and stationary-point convergence of the reconstructed GWT direction.

C.1 Haar Detail Energy

We begin with a discrete Poincaré inequality for a single block.

Lemma C.1 (Discrete Poincaré Inequality)**.** *Let* $x = (x_1, \ldots, x_b) \in \mathbb{R}^b$ *and let*

$$\bar{x} = \frac{1}{b}\sum_{k=1}^{b} x_k.$$

Define $R_b \in \mathbb{R}^{(b-1)\times b}$ *by* $(R_b x)_k = x_{k+1} - x_k$*. Then*

$$\|x - \bar{x}\mathbf{1}\|_2 \leq \kappa_b \|R_b x\|_2, \qquad \kappa_b = \frac{1}{2\sin(\pi/(2b))}. \tag{18}$$

Proof. Let $L_b = R_b^\top R_b$ be the path-graph Laplacian. Its quadratic form is

$$x^\top L_b x = \|R_b x\|_2^2 = \sum_{k=1}^{b-1}(x_{k+1} - x_k)^2.$$

For $h = 0, \ldots, b-1$, define $v^{(h)} \in \mathbb{R}^b$ by

$$v_k^{(h)} = \cos\left(\left(k - \frac{1}{2}\right)\frac{h\pi}{b}\right), \qquad k = 1, \ldots, b.$$

For an interior index, the cosine addition identity gives

$$\begin{aligned}(L_b v^{(h)})_k &= 2v_k^{(h)} - v_{k-1}^{(h)} - v_{k+1}^{(h)} \\ &= 2\left(1 - \cos\left(\frac{h\pi}{b}\right)\right) v_k^{(h)}.\end{aligned}$$

The same equality holds at the two endpoints by the first and last rows of L_b. Hence $v^{(h)}$ is an eigenvector with eigenvalue

$$\lambda_h = 2\left(1 - \cos\left(\frac{h\pi}{b}\right)\right), \qquad h = 0, \dots, b-1.$$

The eigenvalue $\lambda_0 = 0$ corresponds to the constant vector $\mathbf{1}$, and the smallest positive eigenvalue is

$$\lambda_1 = 2\left(1 - \cos\left(\frac{\pi}{b}\right)\right) = 4\sin^2\left(\frac{\pi}{2b}\right).$$

Set $y = x - \bar{x}\mathbf{1}$. By construction, y is orthogonal to $\mathbf{1}$ and $R_b y = R_b x$. The Rayleigh quotient on the subspace orthogonal to $\mathbf{1}$ therefore gives

$$y^\top L_b y \geq \lambda_1 \|y\|_2^2.$$

Using $y^\top L_b y = \|R_b x\|_2^2$ and rearranging yields

$$\|x - \bar{x}\mathbf{1}\|_2 \leq \frac{1}{\sqrt{\lambda_1}}\|R_b x\|_2 = \frac{1}{2\sin(\pi/(2b))}\|R_b x\|_2,$$

which proves the claim. □

The vector inequality extends row-wise to a matrix gradient.

Lemma C.2 (Block-Averaging Error)**.** *Let $b = 2^l$ divide n, and let $P_l(G)$ replace every consecutive block of b columns of $G \in \mathbb{R}^{m\times n}$ by its block mean. Then*

$$\|G - P_l(G)\|_F \leq \kappa_b \|\Delta G\|_F. \tag{19}$$

Proof. Fix a row of G and one consecutive block of b columns. Applying Lemma C.1 to the entries in this block bounds the squared deviation from the block mean by κ_b^2 times the sum of squared adjacent differences within the same block. Summing this inequality over all rows and all blocks gives

$$\|G - P_l(G)\|_F^2 \leq \kappa_b^2 \sum_{i=1}^{m}\sum_{q=1}^{n/b}\sum_{k=1}^{b-1}\left(G_{i,(q-1)b+k+1} - G_{i,(q-1)b+k}\right)^2.$$

The right-hand side contains only adjacent differences that remain inside a Haar block. These terms form a subset of all adjacent-column differences in ΔG, and hence

$$\|G - P_l(G)\|_F^2 \leq \kappa_b^2 \|\Delta G\|_F^2.$$

Taking square roots proves the result. □

We can now prove Lemma 3.1.

Proof of Lemma 3.1. Suppressing the optimization-iteration subscript, the low-pass reconstruction has the coefficient representation

$$P_l(G) = \mathcal{H}_l^{-1}(A^{(l)}, 0, \dots, 0).$$

Subtracting it from the exact reconstruction in equation 3 gives

$$G - P_l(G) = \mathcal{H}_l^{-1}(0, D^{(l)}, \dots, D^{(1)}).$$

Because the Haar transform is orthogonal, it preserves the Frobenius norm. Therefore,

$$\|G - P_l(G)\|_F^2 = \sum_{j=1}^{l}\|D^{(j)}\|_F^2.$$

Applying Lemma C.2 to the left-hand side yields

$$\sum_{j=1}^{l}\|D^{(j)}\|_F^2 = \|G - P_l(G)\|_F^2 \leq \kappa_b^2\|\Delta G\|_F^2,$$

which is equation 10. □

C.2 Temporal Coherence of the Approximation State

We next prove the representation of the coarse-momentum error used to interpret Assumption 3.3.

Proof of Lemma 3.2. Starting from $M_0^A = 0$, repeated substitution into the first recurrence in equation 6 gives

$$M_{t+1}^A = (1-\beta_1)\sum_{q=0}^{t}\beta_1^{t-q}A_q^{(l)}.$$

The geometric sum of the scalar coefficients is $1-\beta_1^{t+1}$. Dividing by the same factor in the bias correction therefore yields

$$\widehat{M}_{t+1}^A = \sum_{q=0}^{t}\omega_{t,q}A_q^{(l)}, \qquad \sum_{q=0}^{t}\omega_{t,q} = 1.$$

Thus, the bias-corrected first moment is a convex combination of the approximation coefficients observed from iteration 0 through iteration t.

Subtracting the current coefficient $A_t^{(l)}$ and using the fact that the weights sum to one gives

$$\Xi_t^A = \sum_{q=0}^{t}\omega_{t,q}\big(A_q^{(l)} - A_t^{(l)}\big).$$

The term with $q = t$ is zero. For every $q < t$, the difference can be expanded as a telescoping sum,

$$A_q^{(l)} - A_t^{(l)} = -\sum_{u=q}^{t-1}\big(A_{u+1}^{(l)} - A_u^{(l)}\big).$$

Substituting this expansion and exchanging the order of the two finite sums gives

$$\Xi_t^A = -\sum_{u=0}^{t-1}\left(\sum_{q=0}^{u}\omega_{t,q}\right)\big(A_{u+1}^{(l)} - A_u^{(l)}\big).$$

The cumulative coefficient is

$$\begin{aligned}\sum_{q=0}^{u}\omega_{t,q} &= \frac{1-\beta_1}{1-\beta_1^{t+1}}\sum_{q=0}^{u}\beta_1^{t-q}\\ &= \frac{\beta_1^{t-u}(1-\beta_1^{u+1})}{1-\beta_1^{t+1}} = c_{t,u}.\end{aligned}$$

This proves the second identity in equation 11.

Finally, multiply the identity element-wise by $(S_{t+1}^A)^{1/2}$ and apply the triangle inequality. Since every $c_{t,u}$ is nonnegative,

$$\left\|(S_{t+1}^A)^{1/2}\odot\Xi_t^A\right\|_F \le \sum_{u=0}^{t-1}c_{t,u}\left\|(S_{t+1}^A)^{1/2}\odot\big(A_{u+1}^{(l)} - A_u^{(l)}\big)\right\|_F,$$

which is equation 12. At $t = 0$, the bias-corrected first moment equals $A_0^{(l)}$, so $\Xi_0^A = 0$ and the empty-sum convention gives the same conclusion. □

C.3 Deterministic Descent and Convergence

The proof uses the fact that an orthogonal transform preserves both norms and inner products. For two coefficient tuples

$$\mathcal{C} = (A, D^{(l)}, \dots, D^{(1)}), \qquad \mathcal{C}' = (A', D'^{(l)}, \dots, D'^{(1)}),$$

we have

$$\begin{aligned}\left\langle \mathcal{H}_l^{-1}(\mathcal{C}), \mathcal{H}_l^{-1}(\mathcal{C}')\right\rangle_F &= \langle A, A'\rangle_F + \sum_{j=1}^{l}\langle D^{(j)}, D'^{(j)}\rangle_F,\\ \left\|\mathcal{H}_l^{-1}(\mathcal{C})\right\|_F^2 &= \|A\|_F^2 + \sum_{j=1}^{l}\|D^{(j)}\|_F^2.\end{aligned} \tag{20}$$

We now prove the alignment and convergence claims. Assumption 3.3 provides the same constants $s_{\min}$, $s_{\max}$, and ρ at every iteration considered by the theorem, so the resulting constants μ and C are uniform in t.

Proof of Theorem 3.4. For brevity, define the coarse-momentum error

$$\Xi_t^A = \widehat{M}_{t+1}^A - A_t^{(l)}.$$

Using equation 20 and the definition of the GWT coefficients, the inner product between the current gradient and the reconstructed direction is

$$\begin{aligned}\langle G_t, \widetilde{G}_t\rangle_F = \alpha\Bigg(&\left\langle A_t^{(l)}, S_{t+1}^A \odot \widehat{M}_{t+1}^A\right\rangle_F\\ &+ \sum_{j=1}^{l}\left\langle D_t^{(j)}, S_{t+1}^{D,j} \odot D_t^{(j)}\right\rangle_F\Bigg).\end{aligned}$$

The approximation term can be separated into the contribution of the current coefficient and the momentum error:

$$\begin{aligned}\left\langle A_t^{(l)}, S_{t+1}^A \odot \widehat{M}_{t+1}^A\right\rangle_F &= \left\|(S_{t+1}^A)^{1/2} \odot A_t^{(l)}\right\|_F^2\\ &\quad + \left\langle (S_{t+1}^A)^{1/2} \odot A_t^{(l)}, (S_{t+1}^A)^{1/2} \odot \Xi_t^A\right\rangle_F.\end{aligned}$$

By the Cauchy–Schwarz inequality and equation 13, the second term is bounded below by

$$\begin{aligned}&- \left\|(S_{t+1}^A)^{1/2} \odot A_t^{(l)}\right\|_F \left\|(S_{t+1}^A)^{1/2} \odot \Xi_t^A\right\|_F\\ &\geq -\rho \left\|(S_{t+1}^A)^{1/2} \odot A_t^{(l)}\right\|_F^2.\end{aligned}$$

It follows that

$$\left\langle A_t^{(l)}, S_{t+1}^A \odot \widehat{M}_{t+1}^A\right\rangle_F \geq (1-\rho)\left\|(S_{t+1}^A)^{1/2} \odot A_t^{(l)}\right\|_F^2.$$

Each detail term is nonnegative and has the exact form

$$\left\langle D_t^{(j)}, S_{t+1}^{D,j} \odot D_t^{(j)}\right\rangle_F = \left\|(S_{t+1}^{D,j})^{1/2} \odot D_t^{(j)}\right\|_F^2.$$

Since every preconditioner entry is at least $s_{\min}$ and $1-\rho \leq 1$, combining these inequalities gives

$$\begin{aligned}\langle G_t, \widetilde{G}_t\rangle_F &\geq \alpha(1-\rho)s_{\min}\left(\|A_t^{(l)}\|_F^2 + \sum_{j=1}^{l}\|D_t^{(j)}\|_F^2\right)\\ &= \alpha(1-\rho)s_{\min}\|G_t\|_F^2,\end{aligned}$$

where the last equality follows from equation 4. This proves the first inequality in equation 14 with $\mu = \alpha(1-\rho)s_{\min}$.

We next bound the norm of the GWT direction. The lower preconditioner bound and equation 13 imply

$$\begin{aligned}\|\Xi_t^A\|_F &\leq s_{\min}^{-1/2}\left\|(S_{t+1}^A)^{1/2} \odot \Xi_t^A\right\|_F\\ &\leq \rho s_{\min}^{-1/2}\left\|(S_{t+1}^A)^{1/2} \odot A_t^{(l)}\right\|_F\\ &\leq \rho\sqrt{\frac{s_{\max}}{s_{\min}}}\|A_t^{(l)}\|_F = \rho\sqrt{\chi}\|A_t^{(l)}\|_F.\end{aligned}$$

Hence the triangle inequality gives

$$\|\widehat{M}^A_{t+1}\|_F \le (1+\rho\sqrt{\chi})\|A^{(l)}_t\|_F.$$

Using equation 20 and the upper preconditioner bound, we obtain

$$\begin{aligned}\|\widetilde{G}_t\|_F^2 &= \alpha^2\left(\|S^A_{t+1}\odot\widehat{M}^A_{t+1}\|_F^2+\sum_{j=1}^{l}\|S^{D,j}_{t+1}\odot D^{(j)}_t\|_F^2\right)\\ &\le \alpha^2 s_{\max}^2\left((1+\rho\sqrt{\chi})^2\|A^{(l)}_t\|_F^2+\sum_{j=1}^{l}\|D^{(j)}_t\|_F^2\right)\\ &\le \alpha^2 s_{\max}^2(1+\rho\sqrt{\chi})^2\left(\|A^{(l)}_t\|_F^2+\sum_{j=1}^{l}\|D^{(j)}_t\|_F^2\right)\\ &= C^2\|G_t\|_F^2.\end{aligned}$$

Taking square roots proves the second inequality in equation 14.

It remains to show convergence. Since F has an L-Lipschitz gradient, the descent lemma applied to $W_{t+1}=W_t-\eta\widetilde{G}_t$ yields

$$F(W_{t+1})\le F(W_t)-\eta\langle\nabla F(W_t),\widetilde{G}_t\rangle_F+\frac{L\eta^2}{2}\|\widetilde{G}_t\|_F^2.$$

Substituting $G_t=\nabla F(W_t)$ and the two bounds established above gives

$$F(W_{t+1})\le F(W_t)-\eta\left(\mu-\frac{L\eta C^2}{2}\right)\|\nabla F(W_t)\|_F^2.$$

The learning-rate condition ensures that the coefficient in parentheses is positive. Summing from $t=0$ to $T-1$ gives

$$\begin{aligned}\eta\left(\mu-\frac{L\eta C^2}{2}\right)\sum_{t=0}^{T-1}\|\nabla F(W_t)\|_F^2 &\le F(W_0)-F(W_T)\\ &\le F(W_0)-F_*.\end{aligned}$$

Dividing by $T\eta(\mu-L\eta C^2/2)$ proves equation 15.

The proof is completed. □

D Experiment Details

D.1 Network Architecture

In this section, we describe the network architectures of the LLaMA (Large Language Model Meta AI) (Touvron et al., 2023a) and RoBERTa (Robustly Optimized BERT Approach) (Liu, 2019) models, which complement the experimental details provided in the main text. LLaMA is a family of foundational language models based on transformers (Vaswani et al., 2017). For this paper, we adopt the LLaMA models from previous work (Lialin et al., 2023), which use RMSNorm and SwiGLU activations (Touvron et al., 2023b;a). RoBERTa, on the other hand, is a variant of BERT (Bidirectional Encoder Representations from Transformers) (Liu, 2019), also built on the transformer architecture. Table 7 presents the architectural hyperparameters for the LLaMA and RoBERTa-base models, as well as the pre-training token amounts for LLaMA models across different sizes.

D.2 Experiment Hyperparameters

In this section, we provide detailed information on the hyperparameters and experimental setup used in the experiments discussed in the main text. For the LLaMA pre-training tasks, we follow

Table 7: Architecture hyperparameters of LLaMA for pre-training. Batch size and training data amount are specified in tokens.

Models	Params	Hidden	Intermediate	Heads	Layers	Iteration	Tokens
LLaMA	60M	512	1376	8	8	10K	1.3B
	130M	768	2048	12	12	20K	2.6B
	350M	1024	2736	16	24	60K	7.8B
	1.3B	2048	5461	24	32	100K	13.1B
	3B	2560	6848	32	32	120K	15.7B
RoBERTa	125M	768	3072	12	12	-	-

the experimental configuration used in previous work (Zhao et al., 2024), which employs the default Adam hyperparameters ($\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-6}$), a maximum sequence length of 256 tokens, and gradient accumulation with a batch size of 512. Additionally, we apply a learning rate warm-up for the first 10% of the iterations and use a cosine annealing schedule with the initial learning rate.

Table 8: Hyperparameter lr, α, training GPUs, and training time (hours) for different LLaMA models. We report the training time for Adam with our GWT.

	60M	130M	350M	1.3B
Full-Adam	0.005, -	0.001, -	0.001, -	0.0005, -
GWT	0.01, 0.25	0.01, 0.25	0.01, 0.25	0.01, 0.25
GaLore	0.01, 0.25	0.01, 0.25	0.01, 0.25	0.01, 0.25
APOLLO	0.01, 1.0	0.01, 1.0	0.01, 1.0	0.01, 1.0
GPUs (RTX 3090)	4	4	16	32
Time	0.99h	4.03h	11.83	40.18h

For the GWT method, we perform hyperparameter tuning on the scale parameter α from the set $[0.1, 0.15, 0.2, 0.25]$, selecting the value that yields the best performance across all LLaMA pre-training tasks. We found that our method is not sensitive to scale, and setting α to 0.25 yields good results in most pretraining tasks, which is consistent with the hyperparameters used in previous work (Zhao et al., 2024; Zhu et al., 2024). For the baseline methods, we use their default hyperparameters: $\alpha = 0.25$ and $lr = 0.01$ for GaLore, and $lr = 0.001$ for Adam (0.005 for 60M and 0.0005 for LLaMA 1.3B and 3B). Consistent with prior work (Zhao et al., 2024), we enable GWT and GaLore in the attention and MLP modules.

Finally, we summarize the hyperparameter α, the GPUs used (RTX 3090), and the training time (in hours) for our method required to reproduce the results shown of Table 2 in Table 8.

For the MMLU fine-tuning tasks, we adopt the implementation from (Zheng et al., 2024) for both the dataset identity and the Alpaca English (Taori et al., 2023). For our GWT method, we use a fixed decomposition level of $l = 8$ across all evaluated models. This setting aligns the memory footprint with low-rank-based baselines that employ a rank of 8. The gradient scaling coefficient α is empirically set to $\alpha = 0.25$. For other baselines, we follow the hyperparameter recommendations provided in their original papers: $\alpha = 2$ for GaLore (Zhao et al., 2024), $\alpha = 32$ for APOLLO (Zhu et al., 2024), and $\alpha = 16$ $(2 \times r)$ for LoRA (Hu et al., 2022). In contrast to pre-training, we apply the memory-efficient fine-tuning technique to all linear layers during this task. We report the best accuracy obtained by sweeping the learning rate over the range [1.0e-6, 2.5e-6, 5.0e-6, 1.0e-5, 2.5e-5, 5.0e-5, 1.0e-4] for all the tested models fine-tuning. A summary of all used hyperparameters is provided in Table 9.

Table 9: Hyperparameter of different models on the MMLU benchmark.

Hyperparameter	Adam	LoRA	GaLore	APOLLO	GWT
Rank r	-	8	8	8	-
Level l	-	-	-	-	8
Scale α	-	16	2	32	0.25
lr scheduler			Cosine		
Warmup steps			10%		
Epochs			3		
Batch size w. accumulation			32		
Where			All		
N-shots			5		
Cut-off length			2048		
lr (LLaMA3.2-3B)	1.0e-6	2.5e-5	1.0e-5	1.0e-6	1.0e-5
lr (Gemma3-4B)	1.0e-6	5.0e-6	2.5e-6	1.0e-6	1.0e-5
lr (Qwen2.5-7B)	-	2.5e-6	2.5e-6	1.0e-6	5.0e-5

D.3 Experimental Enviroments

We utilize the BF-16 format throughout all pre-training processes to optimize memory usage. All experiments were conducted on a setup of 1 to 8 server nodes, each equipped with four NVIDIA 3090 GPUs and an Intel(R) Xeon(R) Gold 6226R CPU @ 2.90GHz. The Python version used is 3.10, with PyTorch (Paszke et al., 2017) version 2.3.1 for pre-training and 2.6.0 for MMLU fine-tuning. For the implementation of GWT, we leverage the PyTorch Wavelet Toolbox (Wolter et al., 2024).

E Additional Experimental Results

E.1 Empirical Validation of Coarse-Momentum Coherence

The convergence analysis in Section 3.4 relies on the coarse-momentum coherence condition in Eq. equation 13. We empirically examine whether this sufficient condition captures the behavior of GWT during practical LLM training. Specifically, we train LLaMA-60M and LLaMA-130M for 10K optimization steps using GWT-2 and GWT-3. All other settings follow the corresponding pre-training experiments in the main text, including the optimizer hyperparameters, batch size, sequence length, warm-up, and learning-rate schedule. During training, we periodically record gradients from representative layers and all GWT-optimized attention and MLP linear matrices within these layers.

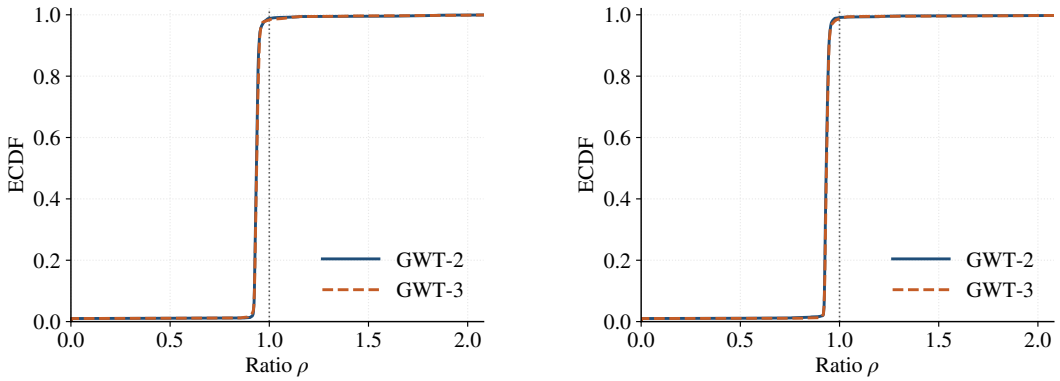
For each monitored matrix gradient G_t, we compute the empirical coherence ratio

$$\rho_t^{\text{emp}} = \frac{\left\| (S_{t+1}^A)^{1/2} \odot \left(\widehat{M}_{t+1}^A - A_t^{(l)} \right) \right\|_F}{\left\| (S_{t+1}^A)^{1/2} \odot A_t^{(l)} \right\|_F}. \tag{21}$$

Here, $A_t^{(l)}$ is the current coarsest Haar approximation coefficient, $\widehat{M}_{t+1}^A$ is its bias-corrected first-moment estimate, and S_{t+1}^A is the corresponding preconditioner. Thus, ρ_t^{emp} measures the magnitude of the preconditioned coarse-momentum error relative to the current approximation coefficient. The condition $\rho_t^{\text{emp}} < 1$ directly corresponds to the norm-based sufficient coherence condition in Assumption 3.3. Each observation in the following statistics corresponds to one monitored matrix at one sampled optimization step.

Figure 6 first reports the empirical cumulative distribution function (ECDF) of ρ_t^{emp} over all sampled matrix–step pairs. For a threshold x, the ECDF gives the fraction of samples satisfying $\rho_t^{\text{emp}} \leq x$; hence, its value around $x = 1$ directly reflects how frequently the sufficient coherence condition is satisfied. The distributions are strongly concentrated below 1 for both model sizes and both

decomposition levels. As summarized in Table 10, the median ratios lie in the narrow range 0.933–0.936, while the 95th percentiles remain between 0.952 and 0.955. Overall, 98.24%–99.05% of all sampled matrix–step pairs satisfy $\rho_t^{\text{emp}} < 1$.



(a) LLaMA-60M (b) LLaMA-130M

Figure 6: **Empirical distribution of coarse-momentum coherence.** The horizontal axis denotes the empirical coherence ratio ρ_t^{emp}, and the vertical axis is its ECDF, i.e., the fraction of sampled matrix–step pairs with coherence ratio no larger than the corresponding value. The vertical reference at $\rho = 1$ denotes the sufficient coherence threshold in Assumption 3.3. Both GWT levels exhibit closely matched distributions across LLaMA-60M and LLaMA-130M.

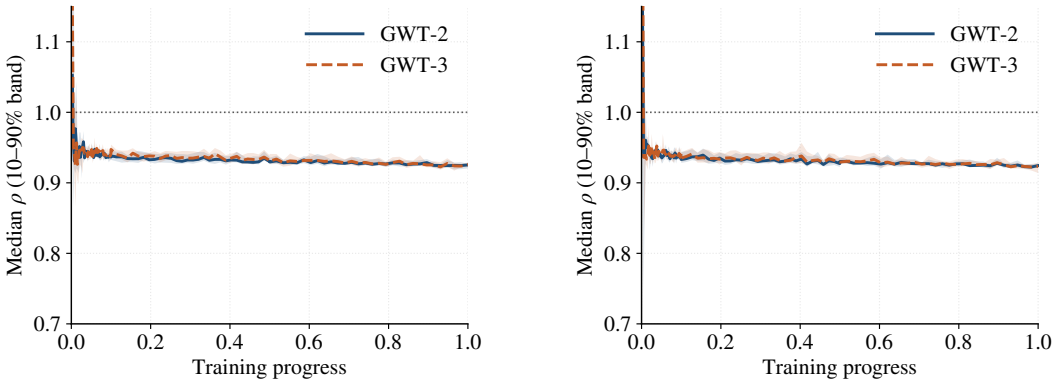
Figure 7 further preserves the temporal dimension and examines how the coherence ratio evolves throughout the 10K-step training trajectory. At each sampled optimization step, we collect ρ_t^{emp} from all monitored matrices. The solid curve shows the median across matrices, while the shaded region spans the 10th–90th percentiles (P10–P90) of the same cross-matrix distribution. The shaded band therefore characterizes variation across different layers and modules at a fixed optimization step; it is neither a confidence interval across random seeds nor a temporal moving window.

The median and the central P10–P90 range remain below $\rho = 1$ over nearly the entire training trajectory. The rare violations are strongly localized in time and occur almost exclusively during the first 50 optimization steps, when the optimizer states are still in their initial transient stage. These early samples are retained in all aggregate statistics. After this short initial period, the monitored matrices maintain stable coarse-momentum coherence throughout the remainder of training.

We further examine whether the aggregate coherence behavior is dominated by particular matrix types. Figure 8 reports the distribution of ρ_t^{emp} separately for the attention and MLP linear modules. Across both LLaMA-60M and LLaMA-130M, and under both GWT-2 and GWT-3, the medians and central box ranges of essentially all module types remain clearly below $\rho = 1$. Moreover, the module-wise distributions are broadly similar to one another. This indicates that the high overall satisfaction rate is not driven by a small subset of layers or matrix types, but is a common property across the monitored Transformer modules.

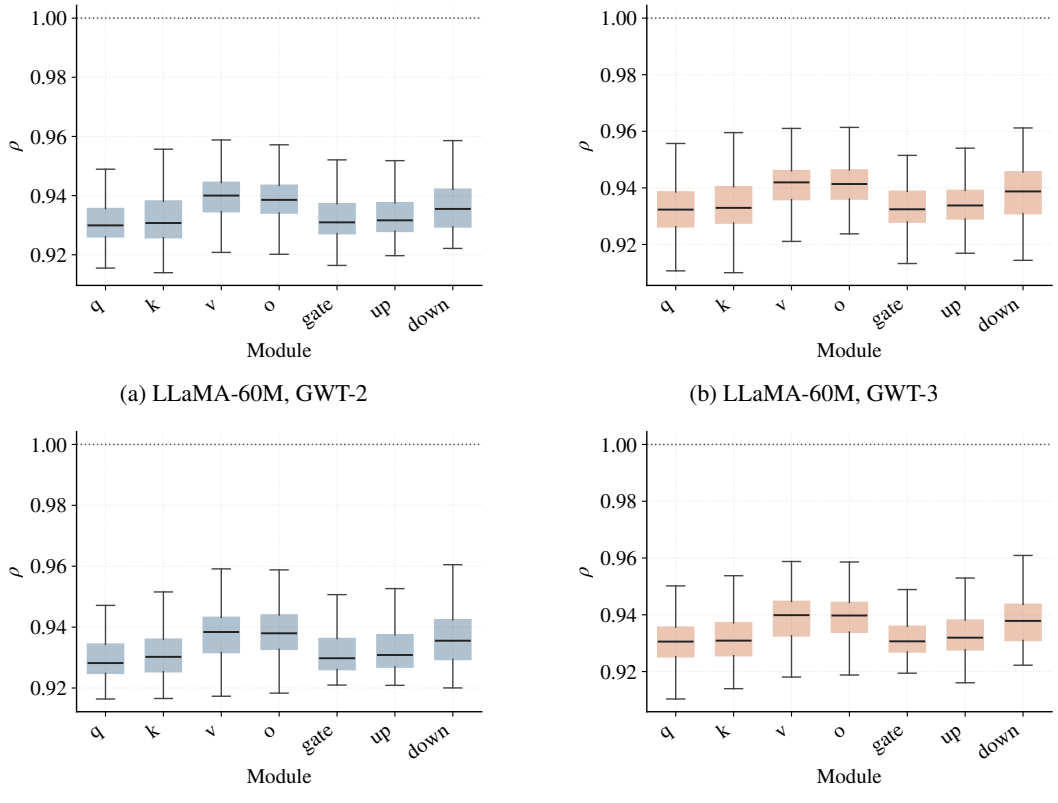
The empirical behavior is also highly consistent when increasing the model size from LLaMA-60M to LLaMA-130M. Both models have median ρ values of approximately 0.93, P95 values of approximately 0.95, and satisfaction rates above 98% for both GWT-2 and GWT-3. Together with the similar module-wise distributions, this suggests that the observed coarse-momentum coherence is not specific to the smallest model and remains stable under the model-scale increase considered here.

Taken together, the ECDF, temporal evolution, and module-wise distributions consistently support the empirical relevance of the coarse-momentum coherence condition. We emphasize that $\rho_t^{\text{emp}} < 1$ is a sufficient rather than necessary condition. In particular, the small number of early samples exceeding the threshold does not imply that the complete reconstructed GWT direction becomes negatively aligned with the instantaneous gradient. We examine this property next.



(a) LLaMA-60M (b) LLaMA-130M

Figure 7: **Evolution of coarse-momentum coherence during pre-training.** At each sampled optimization step, the solid curve denotes the median ρ_t^{emp} across all monitored matrices, and the shaded region denotes the corresponding P10–P90 range across matrices. Hence, the shaded band measures cross-layer/module dispersion at the same step rather than variation across random seeds or temporal smoothing. The horizontal reference at $\rho = 1$ marks the sufficient coherence threshold. The few violations are concentrated in the very early stage of training.



(a) LLaMA-60M, GWT-2 (b) LLaMA-60M, GWT-3

(c) LLaMA-130M, GWT-2 (d) LLaMA-130M, GWT-3

Figure 8: **Module-wise coarse-momentum coherence.** Boxplots show the empirical distributions of ρ_t^{emp} for different attention and MLP linear modules. The horizontal reference at $\rho = 1$ denotes the sufficient coherence threshold. Across both model sizes and decomposition levels, the central distributions of the monitored modules remain consistently below this threshold, showing that coarse-momentum coherence is not confined to a particular module type.

Table 10: **Empirical validation of coarse-momentum coherence.** Statistics are computed over all sampled matrix–step pairs, including the initial transient stage. The last column reports the fraction satisfying the sufficient condition $\rho_t^{\text{emp}} < 1$.

Model	Setting	Median ρ	P95 ρ	Max ρ	$P(\rho < 1)$
LLaMA-60M	GWT-2	0.934	0.955	2.219	98.76%
	GWT-3	0.936	0.955	2.129	98.24%
LLaMA-130M	GWT-2	0.933	0.952	4.283	99.05%
	GWT-3	0.934	0.953	4.250	98.71%

E.2 Understanding GWT: Energy Concentration and Directional Alignment

We next investigate why GWT remains effective despite maintaining persistent optimizer states only for the coarsest approximation coefficients. We examine three complementary properties: whether the compact approximation band captures a disproportionate fraction of the instantaneous gradient energy, whether this concentration depends on the original local column structure of the gradient matrix, and whether the complete reconstructed GWT direction remains aligned with the instantaneous gradient.

Energy concentration in the approximation band. For each sampled gradient, we define the approximation-energy ratio as

$$R_{A,t} = \frac{\|A_t^{(l)}\|_F^2}{\|G_t\|_F^2}. \tag{22}$$

Since the Haar transform is orthogonal, $R_{A,t}$ measures the exact fraction of instantaneous gradient energy contained in the coarsest approximation band, with the complementary fraction carried by the detail bands. In contrast, the approximation band contains only a fraction 2^{-l} of the transformed coefficients, which is also the fraction for which GWT maintains persistent optimizer states. We therefore define the energy enrichment as the median approximation-energy ratio divided by 2^{-l}.

The left panel of Figure 9 reports the ECDF of $R_{A,t}$ over all sampled matrix–step pairs. The horizontal axis is the approximation-energy ratio, while the vertical axis gives the cumulative fraction of samples with approximation energy no larger than the corresponding value. The distributions show that the compact approximation band captures a substantial fraction of the instantaneous gradient energy. As summarized in Table 11, GWT-2 maintains persistent states for only 25% of the transformed coefficients, yet the median approximation-energy ratios reach 51.5% and 48.9% on LLaMA-60M and LLaMA-130M, respectively. For GWT-3, only 12.5% of the coefficients retain persistent states, while the corresponding median energy ratios remain 36.4% and 37.1%. These values correspond to enrichment factors of $1.96\times$–$2.96\times$ across the four settings.

Dependence on the original column structure. The detail-energy analysis in Section 3.4 relates the approximation-band energy to local variation between neighboring gradient columns. We therefore perform a column-permutation control to determine whether the observed concentration depends on the original ordering of the matrix columns.

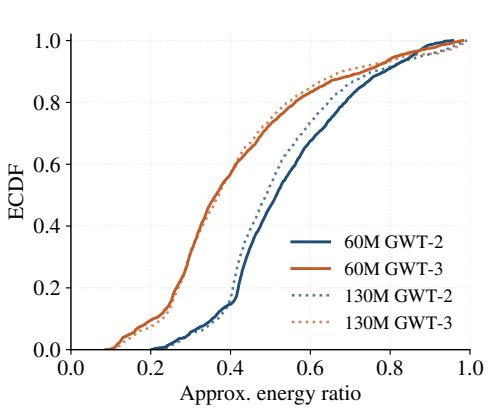
For each gradient selected for the permutation diagnostic, we compute the approximation-energy ratio $R_{A,t}^{\text{orig}}$ under its original column ordering. We then independently permute the matrix columns several times, apply the same Haar transform to each permutation, and average the resulting approximation-energy ratios. We define the permutation energy gap as

$$\Delta R_{A,t} = R_{A,t}^{\text{orig}} - \frac{1}{K_{\text{perm}}} \sum_{k=1}^{K_{\text{perm}}} R_{A,t}^{\text{perm},k}, \tag{23}$$

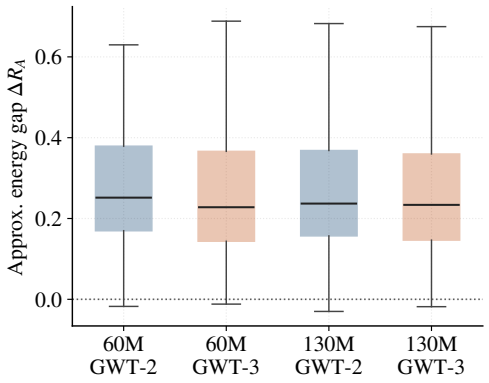
where K_{perm} denotes the number of random column permutations. A positive $\Delta R_{A,t}$ therefore means that the original column ordering places more gradient energy in the Haar approximation band than the average random permutation of the same gradient matrix.

The right panel of Figure 9 reports boxplots of $\Delta R_{A,t}$ for the four model/level configurations, using approximately 210 permutation-diagnostic samples per configuration. The distributions lie

predominantly above zero, showing that the original column ordering systematically yields higher approximation-band energy than the randomly permuted controls. This behavior is also consistent across the two model sizes and both GWT levels. Quantitatively, Table 11 reports $P(\Delta R_A > 0)$, which ranges from 94.8% to 98.1%. Since random column permutation preserves the entries and total Frobenius energy of each gradient while disrupting adjacency relationships, the positive energy gaps indicate that Haar exploits non-random local structure in the original gradient matrices.



(a) ECDF of approximation-energy ratio



(b) Approximation-energy gap under column permutation

Figure 9: **Gradient energy structure in the Haar domain.** Left: ECDF of the approximation-energy ratio $R_{A,t} = \|A_t^{(l)}\|_F^2 / \|G_t\|_F^2$. The horizontal axis denotes the fraction of instantaneous gradient energy contained in the coarsest approximation band, and the vertical axis denotes its empirical cumulative distribution. Right: boxplots of the permutation energy gap $\Delta R_{A,t}$, defined as the approximation-energy ratio under the original column ordering minus its mean over random column permutations. Positive values indicate that the original ordering concentrates more gradient energy in the Haar approximation band than the permutation control.

Table 11: **Energy concentration in the Haar approximation band.** State Frac. is the fraction 2^{-l} of transformed coefficients for which GWT maintains persistent optimizer states. Approx. Energy is the median approximation-energy ratio, and Enrichment is its ratio to the state fraction. The final column reports the fraction of permutation-diagnostic samples with a positive energy gap $\Delta R_A > 0$.

Model	Setting	State Frac.	Approx. Energy	Enrichment	$P(\Delta R_A > 0)$
LLaMA-60M	GWT-2	25.0%	51.5%	2.06×	96.7%
	GWT-3	12.5%	36.4%	2.91×	97.6%
LLaMA-130M	GWT-2	25.0%	48.9%	1.96×	98.1%
	GWT-3	12.5%	37.1%	2.96×	94.8%

Directional alignment of the reconstructed update. Energy concentration alone does not guarantee that state compression produces a useful optimization direction. We therefore directly measure the cosine similarity between the instantaneous gradient G_t and the reconstructed GWT direction $\widetilde{G}_t$ before the optional Norm-growth Limiter:

$$\mathrm{CosAlign}_t = \frac{\langle G_t, \widetilde{G}_t \rangle_F}{\|G_t\|_F \|\widetilde{G}_t\|_F}. \tag{24}$$

A positive value indicates that the reconstructed direction has a positive inner product with the current gradient.

We additionally decompose the gradient–update inner product into its approximation- and detail-band contributions. The approximation band contains the temporally accumulated first-moment state, whereas each current detail band contributes a nonnegative preconditioned term. We report

the normalized detail contribution to the complete gradient–update inner product; this quantity can exceed one when the approximation-band contribution is negative.

As shown in Table 12, the reconstructed GWT direction is positively aligned with the instantaneous gradient for every sampled matrix–step pair across both model sizes and decomposition levels. The median cosine similarities range from 0.649 to 0.724, and even the minimum observed similarities remain positive, between 0.195 and 0.217. This includes the rare samples for which $\rho_t^{\text{emp}} \geq 1$, empirically confirming that the coherence condition is sufficient rather than necessary for positive directional alignment.

The retained detail coefficients also make a substantial contribution to the complete update. Their median normalized contribution ranges from 93.7% to 96.7% across the four settings. Hence, while persistent optimizer states are compressed into the approximation band, the current detail coefficients preserve the complementary gradient information and continue to provide a positive contribution to the reconstructed direction.

Table 12: **Directional alignment of the reconstructed GWT update.** Cosine similarity is measured between the instantaneous gradient G_t and the reconstructed direction $\widetilde{G}_t$ before the optional Norm-growth Limiter. Detail Contribution denotes the median normalized contribution of the retained current detail coefficients to the gradient–update inner product.

Model	Setting	Median Cos.	Min Cos.	$P(\text{Cos} > 0)$	Detail Contribution
LLaMA-60M	GWT-2	0.649	0.215	100%	93.7%
	GWT-3	0.717	0.203	100%	96.6%
LLaMA-130M	GWT-2	0.653	0.217	100%	93.9%
	GWT-3	0.724	0.195	100%	96.7%

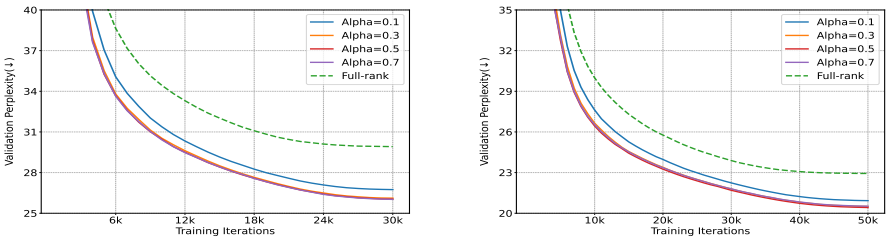
Taken together, these diagnostics provide complementary evidence for the empirical behavior of GWT. The compact approximation band captures substantially more gradient energy than its dimensional fraction, and the positive permutation energy gaps show that this concentration benefits from the original local ordering of gradient columns. At the same time, the complete reconstructed direction remains positively aligned with the instantaneous gradient in all sampled cases, with the retained detail coefficients providing a substantial positive contribution. These properties are consistent across LLaMA-60M and LLaMA-130M and across GWT-2 and GWT-3, suggesting that the observed structure is stable under the model-scale and compression-level variations considered here.

E.3 Study the Effect of α

Similar to other memory-efficient optimizers (Zhao et al., 2024; Zhu et al., 2024; Chen et al., 2024), GWT introduces an additional hyperparameter α to control the effective learning rate of the GWT module. In this section, we investigate the impact of this hyperparameter on the performance of GWT. We evaluate Adam with a 2-level Haar GWT across different values of α while fixing the learning rate at $lr = 0.01$. As shown in Figure 10, our results indicate that GWT performance is largely invariant to the choice of α. In particular, for $\alpha > 0.1$, the final performance remains stable with no significant differences. Under our module-wise optimizer strategy, most network modules operate with an effective learning rate greater than 0.001 ($lr \times \alpha$). In contrast, Adam typically requires learning rates smaller than 0.001 to prevent loss spikes. This highlights the superior robustness of GWT to higher learning rates. Furthermore, in fine-tuning tasks, tuning α effectively corresponds to adjusting the learning rate lr under our strategy. A more detailed study of the interaction between lr and α in fine-tuning settings is provided in the Appendix.

E.4 Extension to Other Model Families

In this section, we extend GWT to models beyond pre-training LLaMA, where we pre-train GPT-2 (Radford et al., 2019), DeBERTa (He et al., 2021), and Qwen2.5-0.5B (Yang et al., 2024) on the C4 dataset with 2.6B tokens. The final validation loss results are presented in Table 13. We can see that GWT still maintains superior performance in terms of validation loss across the tested models,



(a) LLaMA 60M (b) LLaMA 130M

Figure 10: **Study the effects of α on pre-training LLaMA 60M and 130M models.**

demonstrating its effectiveness on other LLM architectures and its strength as a choice for LLM pre-training.

Table 13: **Final validation loss on GPT, DeBERTa, and Qwen.**

Method	GPT	DeBERTa	Qwen
Full-rank Adam	3.31	2.16	2.85
GaLore-1/4	3.43	2.22	2.97
APOLLO	3.26	2.07	2.82
GWT-2	**3.22**	**2.02**	**2.70**

E.5 Memory and Throughput Estimates

In this section, we provide further clarification on the GPU memory usage results reported in the paper. First, Table 1 presents the theoretical memory consumption of various methods for comparison.

Compared to low-rank methods that reduce memory overhead by controlling the rank, our Gradient Wavelet Transform (GWT) method does not require additional storage for projection matrices. In our estimates, GWT requires no persistent projection matrix and can use less optimizer-state memory than the compared low-rank methods at matched compression settings.

In actual training scenarios, GPU memory usage is influenced by multiple factors, including model size, batch size, optimizer states, PyTorch's memory fragmentation management, and whether training is conducted on a single GPU or across multiple GPUs. To focus on core differences, we estimate GPU memory usage for the model weights and Adam optimizer states using bfloat16 precision. For example, the weights of the LLaMA 60M model occupy approximately 0.11 GB of GPU memory, while the Adam optimizer states require about 0.23 GB. Since GWT primarily targets reducing optimizer memory usage, the weight memory footprint under Adam with GWT remains unchanged at 0.12 GB.

As a representative example, the LLaMA-60M model contains approximately 58 million parameters. Using BF16 precision (2 bytes per parameter), this yields a model memory footprint of approximately 0.11 GB. The Adam optimizer maintains two auxiliary states—the first-order moment M and second-order moment V—each the same size as the model parameters, resulting in an additional 0.23 GB. The total memory usage with Adam is thus approximately 0.34 GB. For our GWT method, the computation formula depends on the number of parameters that use GWT and those that do not. Specifically, for LLaMA-60M, there are 25.3M parameters that use GWT to save memory, and the remaining 32.77M parameters are updated using Adam. In the case of using GWT with level $l = 2$, the optimizer states that the parameters are calculated as follows:

- The optimizer state for the GWT parameters:

$$25.3\text{ M} \times 2\text{ Bytes} \times 2/4 = 25.3\text{ MBytes}.$$

- The optimizer state for the Adam parameters:

$$32.77\text{ M} \times 2\text{ Bytes} \times 2 = 131.08\text{ MBytes}.$$

- The memory occupied by the model parameters:

$$116.14 \text{ MBytes}.$$

Therefore, the estimated training memory overhead with GWT-2 is

$$25.3 \text{ MBytes} + 131.08 \text{ MBytes} + 116.14 \text{ MBytes}$$
$$= 272.52 \text{ MBytes} s \approx 0.27 \text{ GBytes}$$

Besides the memory estimate, we also report the training throughput as a complement to the training speed report. For example, we pre-train LLaMA 60M with a total batch size of 512 and a max sequence length of 256 in 10,000 iterations. This is equivalent to a total token size of $512 \times 256 = 131072 \approx 131$K and a total of $131072 \times 10000 = 1310720000 \approx 1.3$B tokens during the whole training process. In Tables 16 and 8, we provide the training time and the number of GPUs used. Training LLaMA60M with our Haar-2 on four GPUs takes approximately 0.99 hours. Hence, we can compute the token throughput per GPU per second as $1310720000/(0.99 \times 3600 \times 4) \approx 91.9$Ktokens/s

Table 14: **Memory estimate of weight parameters/optimizer states.** * indicates results published in prior works.

Method	60M	130M	350M	1.3B
Full-Rank	0.11G/0.23G	0.26G/0.51G	0.68G/1.37G	2.60G/5.20G
MUON	0.11G/0.19G	0.26G/0.38G	0.68G/0.86G	2.60G/3.61G
GaLore-1/4	0.11G/0.17G	0.26G/0.32G	0.68G/0.70G	2.60G/2.16G
APOLLO-1/4	0.11G/0.17G	0.26G/0.32G	0.68G/0.70G	2.60G/2.16G
GWT-2	0.11G/0.16G	0.26G/0.29G	0.68G/0.56G	2.60G/1.81G
GaLore-1/8	0.11G/0.15G	0.26G/0.27G	0.68G/0.55G	2.60G/1.55G
APOLLO-1/8	0.11G/0.15G	0.26G/0.27G	0.68G/0.55G	2.60G/1.55G
GWT-3	0.11G/0.14G	0.26G/0.25G	0.68G/0.41G	2.60G/1.20G
Low-Rank*	0.08G/0.17G	0.18G/0.37G	0.36G/0.72G	1.19G/2.38G
LoRA*	0.20G/0.17G	0.44G/0.37G	1.04G/0.72G	3.79G/2.38G
ReLoRA*	0.20G/0.17G	0.44G/0.37G	1.04G/0.72G	3.79G/2.38G

Table 15: Recorded memory overhead for different methods on LLaMA-1.3B with a batch size of 32. We train the LLaMA-1.3B model on 32 NVIDIA RTX 3090 (24GB) GPUs.

Method	Memory	Method	Memory	Method	Memory
Full-Adam	20.61G	GaLore-1/4	17.55G	GaLore-1/8	16.74G
Muon	18.30G	APOLLO-1/4	17.56G	APOLLO-1/8	16.74G
Adam-mini	18.05G	GWT-2	17.25G	GWT-3	16.57G

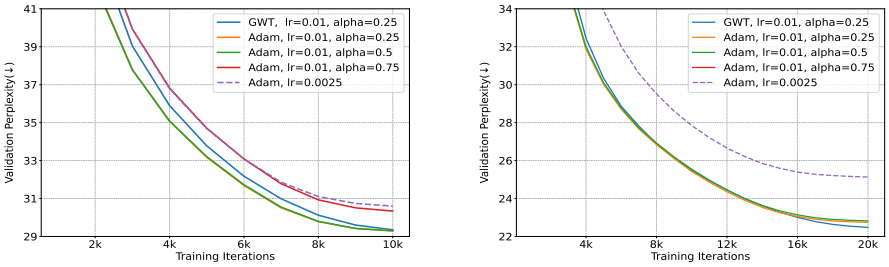
GWT level l. We investigate the effects of the GWT level hyperparameter l, with the corresponding validation perplexity curves shown in Figure 2. We found that the value of l has little impact on the final experimental results, and lower values of l are only slightly better than others. One question that arises from the experimental results is whether the approximation coefficients are not crucial when using wavelets to compress gradients in LLM training, but rather the detail coefficients. The main impact of l seems to be on memory usage and throughput. The throughput results and optimizer memory estimate for different GWT levels are summarized in Table 16, demonstrating the trade-off between memory efficiency and model performance. Higher wavelet levels improve memory usage, but this may come at the cost of slightly higher validation perplexity and slower training speed.

E.6 Adam with Module-wise Learning Rate

One key learning rate strategy employed in GWT is the assignment of different learning rates to distinct parameter modules. Specifically, the overall learning rate is partitioned such that the At-

Table 16: Estimated optimizer memory usage and token throughput (per GPU) with different GWT levels. Higher GWT levels lead to slower training token throughput but smaller memory usage.

Method	60M		130M		350M	
	Memory	Tokens/s	Memory	Tokens/s	Memory	Tokens/s
GWT-1	0.18G	95.8K	0.36G	46.0K	0.86G	12.7K
GWT-2	0.16G	91.9K	0.29G	45.1K	0.56G	12.1K
GWT-3	0.14G	90.1K	0.24G	44.8K	0.41G	11.5K
GWT-4	0.13G	84.2K	0.21G	43.8K	0.33G	10.8K
GWT-5	0.13G	83.5K	0.20G	41.5K	0.30G	9.25K



(a) LLaMA 60M (b) LLaMA 130M

Figure 11: **Applying module-wise learning rate to Adam optimizer.** We can observe that applying a module-wise split learning rate in Adam leads to better convergence speed and final results compared to using a uniform learning rate across all modules.

tention and MLP modules are updated using a scaled learning rate of $lr \times \alpha$, while the rest of the model uses the base learning rate lr. This modular learning rate strategy is commonly adopted in memory-efficient training algorithms, including LoRA (Hu et al., 2022), GaLore (Zhao et al., 2024), Fira (Chen et al., 2024), and APOLLO (Zhu et al., 2024). We incorporate this strategy into GWT as a standard configuration. The method of using different learning rates for different network modules in the Adam optimizer has been studied in related works (Everett et al., 2024; Genet & Inzirillo, 2024). Here, we provide a supplement as an additional ablation experiment.

In this section, we first apply the same modular learning rate scheme to the Adam optimizer to examine whether it also benefits from this design, and to evaluate whether GWT can still match or surpass Adam's performance, as demonstrated in the main text. Following standard practice in other memory-efficient methods, we use a learning rate of $lr \times \alpha$ for the Attention and MLP modules, while keeping lr for the remaining modules. To measure the benefit of this configuration, we compare it with Adam using a unified learning rate of 2.5e-3, selected from the range [1e-3, 2.5e-3, 5e-3, 7.5e-3, 1e-2] based on best performance. The learning curves are shown in Figure 11.

Experimental results show that, under our setup, Adam significantly benefits from the modular learning rate strategy, achieving clearly better performance than with a single global learning rate. Even with this enhanced setting, GWT achieves a final validation perplexity (PPL) close to that of Full-Adam. This partly explains why many memory-efficient training methods can outperform full-rank Adam—the modular learning rate strategy plays a key role. It also suggests that Attention and MLP modules may require smaller learning rates than other parts of the model. These findings raise an important question: Do different modules in LLMs inherently demand distinct learning rates? And could explicitly integrating modular learning rate strategies into Adam lead to further gains?

F Discussion

GWT and GaLore (Zhao et al., 2024) both reduce the persistent optimizer states associated with matrix gradients, but they do so in different ways. GaLore updates a low-dimensional projected gradient and discards the component outside the selected subspace. GWT instead applies an or-

thogonal transform, retains the full coefficient set $(A_t^{(l)}, D_t^{(l)}, \ldots, D_t^{(1)})$ in the current update, and stores temporal moment states only for $A_t^{(l)}$. The detail coefficients are temporary and therefore do not add persistent optimizer-state memory. Thus, GWT should be viewed as state compression in a fixed transform domain rather than as projected gradient descent.

Here, we outline several intriguing open problems related to GWT: (a) Wavelets are originally designed for time-series signals and images, raising the question: Can a specialized wavelet transform be developed to handle gradients, which are inherently more disordered and chaotic? Additionally, since the impact of l on the experimental results is minimal, the main effect of higher-order GWT lies in throughput, while offering higher memory efficiency. Therefore, the potential of GWT with higher l values is worth exploring. Given the linear properties of the Haar wavelet in GWT, is it possible to accelerate multi-level wavelet transforms to achieve even higher memory compression ratios, such as 1/64 or 1/128? (b) The applicability and effectiveness of GWT in other domains, such as vision models (Dosovitskiy et al., 2021) and diffusion models (Ho et al., 2020; Song et al., 2020), remain to be validated.