RESQADAM: RESIDUAL-PRESERVING ADAPTIVE OPTIMIZATION FOR MEMORY-EFFICIENT QUANTIZATION AWARE TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Quantization-aware training (QAT) is the most reliable way to obtain accurate low-bit large language models (LLMs), but training with AdamW keeps two dense FP32 states for every weight matrix. We find that low-rank memory-efficient optimizers such as GaLore and LDAdam degrade substantially in QAT. A considerable part of the QAT momentum lies outside its top-r singular subspace, and projection-based methods discard this part at every step. We further observe that this residual remains useful when only its sign is kept. Based on these observations, we propose ResQAdam. It keeps a single BF16 momentum, applies Adam only to the $r \times r$ coordinates of the principal subspace, and adds a normalized sign update computed from the residual. We evaluate ResQAdam with two from-scratch QAT methods and three QAT fine-tuning pipelines, with weights quantized down to 2 bits. It reduces optimizer-state memory by 71% to 74% relative to AdamW, stays close to AdamW in most settings, and lowers training throughput by less than 2%.

1 INTRODUCTION

Quantization-aware training (QAT) inserts the quantizer into the forward pass and updates latent full-precision weights through the straight-through estimator (STE) (Bengio et al., 2013). At low bit widths it recovers much of the accuracy lost by post-training quantization (PTQ) (Frantar et al., 2022; Xiao et al., 2023), both when fine-tuning a pretrained checkpoint (Liu et al., 2024; Chen et al., 2025; Liu et al., 2026) and when training from scratch (Panferov et al., 2025; Tabesh et al., 2025). This accuracy comes at a memory cost. QAT keeps full-precision weights and gradients, and most recipes rely on AdamW (Loshchilov & Hutter, 2017), which stores two FP32 moments and adds $8nm$ bytes for each $n \times m$ weight matrix.

A natural remedy is to borrow memory-efficient optimizers from full-precision training. GaLore (Zhao et al., 2024) and LDAdam (Robert et al., 2024) project the gradient onto a rank-r subspace, keep Adam states only in that subspace, and map the update back to the full space. This works well when the optimization signal is concentrated in a few directions. Figure 1 shows that the same strategy does not carry over to QAT. With rank 64 and tuned learning rates, both methods are clearly worse than AdamW, in from-scratch training as well as in fine-tuning.

We hypothesize that this is a property of the QAT update itself. Under the STE, the gradient has to compensate for element-wise quantization errors, which are spread over individual entries rather than aligned with a few directions of the weight matrix. If these corrections are less concentrated in the top-r subspace than the rest of the update, a projection-based optimizer discards them at every step, and no amount of tuning inside the subspace recovers them.

To address this limitation, we decompose the QAT optimization signal according to its geometric structure. We apply Singular Value Decomposition (SVD) to the dense global momentum, retaining the top-r singular vectors to form the orthonormal bases $U_t \in \mathbb{R}^{n\times r}$ and $V_t \in \mathbb{R}^{m\times r}$. These bases induce a principal matrix subspace $\mathcal{S}_t = \{U_t A V_t^\top : A \in \mathbb{R}^{r\times r}\}$. The momentum can then be exactly decomposed into its projection onto $\mathcal{S}_t$ and an orthogonal residual lying in $\mathcal{S}_t^\perp$. The principal projection captures the dominant low-rank structure of the optimization trajectory, whereas the orthogonal residual retains dispersed signals outside the principal subspace. Our empirical results

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.

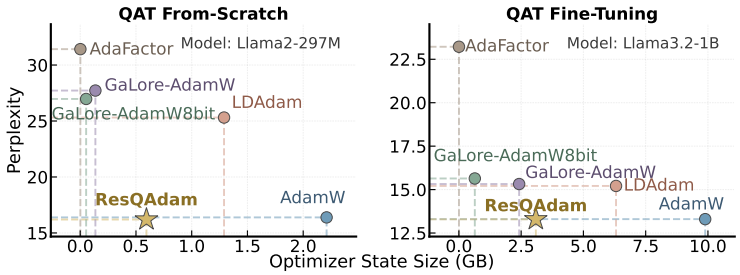


Figure 1: WikiText-2 perplexity versus optimizer state memory for different optimizers under two QAT paradigms. The x-axis reports the total optimizer state size of the full model (in GB). All experiments use W2A16 quantization with group size 128, and projection-based optimizers use rank 64. For a fair comparison, the learning rate of each optimizer is tuned via grid search and the best result is reported.

show that these residual signals become particularly important under QAT. "Their magnitude matters little: keeping only their sign, with no second-moment state, recovers most of the lost quality (Section 5.1).

Motivated by this insight, we propose **ResQAdam** (**Res**idual-Preserving **Q**uantization-Aware **Adam**), a geometrically decoupled optimizer that applies distinct update rules based on subspace characteristics. It stores one dense momentum in BF16 and refreshes its top-r singular vectors every τ steps. Adam runs on the $r \times r$ coordinates of the gradient in this subspace, and the residual of the momentum contributes a sign update whose norm matches the Adam update. Since no dense second moment is kept, persistent optimizer memory falls to roughly a quarter of AdamW, and the SVD cost is spread over τ steps.

By eliminating dense second-moment tracking, ResQAdam limits the persistent optimizer memory footprint to only the BF16 momentum and negligible low-rank components. For a weight matrix $W \in \mathbb{R}^{n\times m}$ with projection rank r, the persistent memory ratio compared to standard AdamW is: $2nm + \mathcal{O}((n+m)r + r^2)/8nm \approx 25\%$. The measured saving is 71% to 74%. Furthermore, while the required SVD introduces computational overhead, refreshing the bases only once every few hundred steps effectively amortizes this cost, resulting in minimal training-speed degradation.

To evaluate our approach, we conduct extensive experiments across both from-scratch QAT and QAT fine-tuning paradigms. Across multiple LLM architectures, quantization configurations, and QAT methods, our empirical results demonstrate that ResQAdam closely tracks the optimization trajectory and largely preserves AdamW-level final performance across the evaluated settings. It achieves this by reducing around 75% of AdamW's average memory footprint, while introducing negligible computational overhead for training.

2 Related Work

From-scratch QAT. Learnable quantizers such as PACT (Choi et al., 2018) and LSQ (Esser et al., 2019) were later scaled to LLMs by BitNet (Wang et al., 2023) and BitNet b1.58 (Ma et al., 2024). Recent work improves the gradient estimate. QuEST (Panferov et al., 2025) applies Hadamard transforms and down-weights gradients of entries with large quantization error, and CAGE (Tabesh et al., 2025) adds a curvature-aware correction to the STE. Both quantize only the forward pass and keep gradients and optimizer states in high precision, which is the setting we target.

QAT Fine-Tuning. A complementary paradigm adapts a converged full-precision checkpoint to recover its compressed representation. LLM-QAT (Liu et al., 2024) performs data-free QAT by distilling from the pretrained model's own generations, jointly quantizing weights, activations, and the KV cache. EfficientQAT (Chen et al., 2025) reduces QAT cost via two stages: block-wise all-parameter training (Block-AP) for a strong low-bit initialization, followed by end-to-end training of only the quantization step sizes (E2E-QP). ParetoQ (Liu et al., 2026) unifies 1-to-4-bit QAT under a single framework and identifies a sharp representation transition between 2 and 3 bits. Unlike parameter-efficient alternatives such as PEQA (Kim et al., 2023), which shrinks optimizer memory by training only the quantization scales, or QLoRA (Dettmers et al., 2023), which fine-tunes high-precision adapters over a frozen quantized base, our method retains *full-parameter* updates while still reducing optimizer memory.

Low-rank Projection Optimization. Low-rank projection methods take a different route: GaLore (Zhao et al., 2024) periodically computes an SVD of the gradient, runs Adam within the resulting compact subspace, and projects the update back, while LDAdam (Robert et al., 2024) performs adaptive optimization inside dynamically shifting subspaces with a projection-aware transport rule and error feedback. These methods are most effective when optimization signals are strongly concentrated in a dominant low-rank subspace. Our experiments show that this assumption becomes substantially less effective under QAT: projection-only optimizers degrade markedly, while preserving the discarded residual restores much of the lost optimization quality. While recent works like Fira (Chen et al., 2026) and FRUGAL (Zmushko et al., 2024) also update the component outside the projected subspace with a state-free rule. ResQAdam differs in the residual is taken from the momentum rather than the current gradient, the bases are computed from the momentum, and the two updates are weighted by their momentum energy.

Algorithm 1 ResQAdam: Main Flow

Require: **Weights** $W_0 \in \mathbb{R}^{n\times m}$, **learning rate** η, **weight decay** λ, **rank** r, **interval** τ, **Adam parameters** $\beta_1, \beta_2, \epsilon$
Initialize: $M_0 \leftarrow 0, \quad m_0^{\|}, v_0^{\|} \leftarrow 0$
for $t = 1, 2, \dots$ **do**
  Obtain STE gradient G_t
  $M_t \leftarrow \beta_1 M_{t-1} + (1-\beta_1) G_t$
  if $t \bmod \tau = 1$ **then**
    $U_t, V_t \leftarrow \mathrm{SVD}(M_t, \mathrm{rank} = r)$
    Transport $m_{t-1}^{\|}, v_{t-1}^{\|}$ to new bases
  end if
  $M_t^{\|} \leftarrow U_t U_t^\top M_t V_t V_t^\top$
  $M_t^{\perp} \leftarrow M_t - M_t^{\|}$
  $\Delta_t^{\|} \leftarrow \textsc{ProjectedAdamUpdate}(G_t, U_t, V_t)$
  $\Delta_t^{\perp} \leftarrow \textsc{ResidualSignUpdate}(M_t^{\perp}, \Delta_t^{\|})$
  Compute energy ratio
  $\rho_t^{\|}, \rho_t^{\perp} \leftarrow \textsc{CalcEnergyWeights}(M_t, M_t^{\|}, M_t^{\perp})$
  Fuse principal and residual directions
  $\Delta_t \leftarrow \rho_t^{\|} \Delta_t^{\|} + \rho_t^{\perp} \Delta_t^{\perp}$
  $W_t \leftarrow (1 - \eta\lambda) W_{t-1} - \eta \Delta_t$
end for

3 ResQAdam

Based on the explicit orthogonal decomposition of optimization signals, we propose ResQAdam, a geometrically decoupled optimizer for QAT. ResQAdam applies different update rules to two parts of the optimization space: a compact coordinate space in $\mathbb{R}^{r\times r}$, where Adam-style preconditioning is performed, and the full parameter space $\mathbb{R}^{n\times m}$, where the orthogonal residual provides a sign-based correction for dispersed quantization-compensation signals. This decomposition substantially reduces the optimizer memory footprint while retaining the residual correction signals required for QAT. The overall training flow is summarized in Algorithm 1.

3.1 Principal Subspace and Residual

ResQAdam keeps no dense second moment. It keeps one dense momentum:

$$M_t = \beta_1 M_{t-1} + (1-\beta_1) G_t \in \mathbb{R}^{n\times m}. \tag{1}$$

which is used to estimate the subspace and the residual direction but never to normalize the update. We store M_t in BF16. Every τ steps, the top-r left and right singular vectors of M_t give $U_t \in \mathbb{R}^{n\times r}$ and $V_t \in \mathbb{R}^{m\times r}$. They span the principal subspace $\mathcal{S}_t = \{U_t A V_t^\top : A \in \mathbb{R}^{r\times r}\}$, which has dimension r^2. With the orthogonal projector $\Pi_t(X) = U_t U_t^\top X V_t V_t^\top$, the momentum splits into $M_t^{\|} = \Pi_t(M_t)$ and $M_t^{\perp} = M_t - M_t^{\|}$, which are orthogonal under the Frobenius inner product.

3.2 Space-Specific Optimization Strategies

Adam in the principal subspace. We project the gradient to $G_t^{\|} = U_t^\top G_t V_t$ and apply Adam to it, which gives preconditioned coordinates $\hat{A}_t^{\|}$ (Algorithm 2). The update in parameter space is:

$$\Delta_t^{\|} = \alpha_r U_t \hat{A}_t^{\|} V_t^\top \tag{2}$$

with $\alpha_r = \sqrt{nm}/r$. The back-projection preserves the Frobenius norm, so an $r \times r$ update with entries of order one has norm $O(r)$, whereas a dense update of the same entry scale has norm $O(\sqrt{nm})$. The factor α_r removes this dependence on r.

Sign update from the residual. A second moment for $M_t^\perp$ would bring back dense memory. We use only its sign and rescale it to the norm of the principal update:

$$\Delta_t^\perp = \|\Delta_t^\parallel\|_F \,\mathrm{sign}(M_t^\perp)/\|\mathrm{sign}(M_t^\perp)\|_F \tag{3}$$

Because the sign acts element-wise, $\Delta_t^\perp$ is in general not orthogonal to $\mathcal{S}_t$. We therefore view it as a direction derived from the residual rather than an update restricted to $\mathcal{S}_t^\perp$.

3.3 Energy-Aware Geometric Fusion

The principal and residual components capture complementary structures of the optimization signal. To combine them in the full parameter space, we adapt their contributions according to the relative momentum energy carried by the two components. Because $M_t^\parallel$ and $M_t^\perp$ form an orthogonal decomposition of the momentum M_t, their squared Frobenius norms satisfy:

$$\|M_t\|_F^2 = \|M_t^\parallel\|_F^2 + \|M_t^\perp\|_F^2. \tag{4}$$

We therefore define the energy ratio as:

$$\rho_t^\parallel = \frac{\|M_t^\parallel\|_F}{\|M_t\|_F + \epsilon}, \qquad \rho_t^\perp = \frac{\|M_t^\perp\|_F}{\|M_t\|_F + \epsilon}. \tag{5}$$

As described in Section 3.2, the residual update is normalized to match the Frobenius norm of the principal update. We then combine the two directions using the momentum-derived energy ratio:

$$\Delta_t = \rho_t^\parallel \Delta_t^\parallel + \rho_t^\perp \Delta_t^\perp \tag{6}$$

Although $M_t^\parallel$ and $M_t^\perp$ are orthogonal by construction, the element-wise sign transformation does not preserve this orthogonality. Therefore, the resulting update directions $\Delta_t^\parallel$ and $\Delta_t^\perp$ are generally not orthogonal. We consequently interpret $\rho_t^\parallel$ and $\rho_t^\perp$ as energy-derived balancing coefficients rather than as weights that preserve an exact orthogonal norm decomposition.

Algorithm 2 Projected Adam Update

Require: $G_t, U_t, V_t, m_{t-1}^\parallel, v_{t-1}^\parallel$
Project gradient to the $r \times r$ coordinate space
$G_t^\parallel \leftarrow U_t^\top G_t V_t$
$m_t^\parallel \leftarrow \beta_1 m_{t-1}^\parallel + (1-\beta_1) G_t^\parallel$
$v_t^\parallel \leftarrow \beta_2 v_{t-1}^\parallel + (1-\beta_2) G_t^\parallel \odot G_t^\parallel$
$\hat{m}_t^\parallel, \hat{v}_t^\parallel \leftarrow \mathrm{BiasCorrect}(m_t^\parallel, v_t^\parallel)$
Adam preconditioning in coordinate space
$\hat{A}_t^\parallel \leftarrow \hat{m}_t^\parallel / (\sqrt{\hat{v}_t^\parallel} + \epsilon)$
$\alpha_r \leftarrow \sqrt{nm}/r$
Map update back to full space
$\Delta_t^\parallel = \alpha_r U_t \hat{A}_t^\parallel V_t^\top$
Return: $\Delta_t^\parallel$

Algorithm 3 residual-sign Update

Require: $M_t^\perp,\ \Delta_t^\parallel$
Use only the residual direction
$D_t^\perp \leftarrow \mathrm{sign}(M_t^\perp)$
Align residual magnitude with the principal update
$\Delta_t^\perp \leftarrow \|\Delta_t^\parallel\|_F \cdot \frac{D_t^\perp}{\|D_t^\perp\|_F + \epsilon}$
Return: $\Delta_t^\perp$

Finally, applying weight decay λ and learning rate η, the parameter update at step t is:

$$W_t = (1 - \eta\lambda) W_{t-1} - \eta \Delta_t \tag{7}$$

3.4 Subspace Refresh and Memory

The SVD is computed only every τ steps to amortize the computational cost. After each refresh, the principal states $m^\parallel$ and $v^\parallel$ are rewritten in the new basis with the transport rule of LDAdam (Robert et al., 2024) (Appendix A). Only these $r \times r$ states need transport, because M_t and its residual are defined in the original parameter space.

By storing the momentum M_t in BF16 and confining adaptive states to the compact coordinate representation, the dominant persistent-state cost of ResQAdam is $2nm$ bytes, with an additional $O((n+m)r + r^2)$ overhead for the bases and principal states. For $r \ll \min(n, m)$ this approaches 25% of AdamW. The measured reduction is 71% to 74% (Section 4), mainly because the bases are not negligible for the smaller matrices.

4 Experiments

We compare ResQAdam with AdamW in from-scratch QAT and QAT fine-tuning, in terms of training loss, final perplexity and accuracy, and optimizer-state memory.

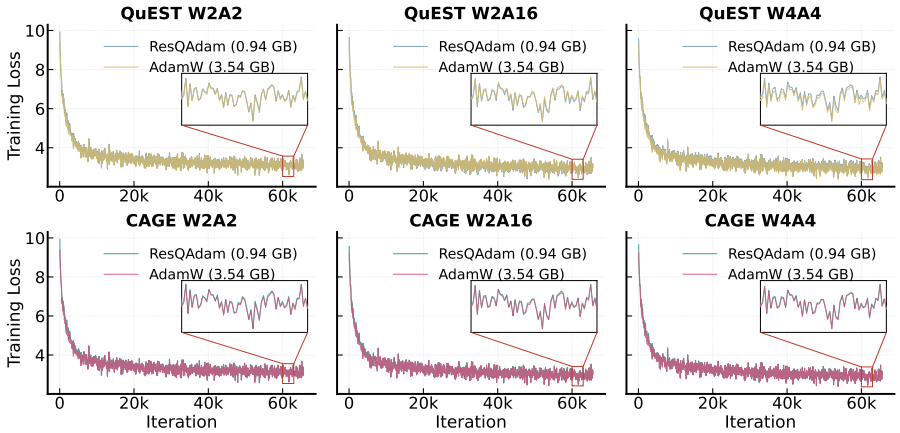


Figure 2: Training loss trajectories of ResQAdam and AdamW during from-scratch QAT under various quantization configurations on the 430M-parameter LLM. The values following the optimizer names in the legend indicate the average memory footprint of the optimizer states.

4.1 Experimental Setup

From-scratch QAT. We consider QuEST (Panferov et al., 2025) and CAGE (Tabesh et al., 2025), which represent two recent low-precision training frameworks with distinct gradient-estimation and optimization strategies. We pretrain Llama-style models at the 200M and 430M scales under W2A2, W2A16, and W4A4 settings.

QAT fine-tuning. We use Basic QAT (end-to-end next-token loss with LSQ scales on 2B RedPajama tokens), LLM-QAT (Liu et al., 2024) (data-free distillation), and the Block-AP stage of EfficientQAT (Chen et al., 2025), on Llama 3.2-1B, Qwen3 4B-Base and Llama 2-7B under multiple quantization configurations.. Appendix C gives the details.

Hyperparameters and evaluation. For from-scratch QAT, ResQAdam uses rank $r = 32$ and refresh interval $\tau = 400$. As an implementation detail of Algorithm 1, the bases are initialized from the first gradient and the periodic refresh is suspended during the first 10% of training steps, since the momentum of a randomly initialized model changes rapidly and early refreshes only add transport noise; after this warm-up phase, refreshes follow the schedule of Algorithm 1. For QAT fine-tuning, we set $r = 64$ and $\tau = 200$ and apply no suspension, because the pretrained checkpoint already provides a stable momentum. For QAT fine-tuning, we set $r = 64$ and $\tau = 200$ to accommodate larger model capacities. All other training hyperparameters are kept strictly consistent with the experimental codebases of the respective QAT methods.

We report perplexity on WikiText-2 and C4 to evaluate models pretrained with QAT. For QAT fine-tuning, in addition to perplexity, we evaluate zero-shot accuracy on several standard bench-

Table 1: Language modeling performance comparison across different from-scratch QAT methods. Values in each cell are formatted as AdamW / ResQAdam (Ours). *Q-Cfg* denotes the quantizer configuration, where *Wx* and *Ax* indicate x-bit weight and activation quantization, respectively. *Opt. State Size* reports average optimizer state size for each optimizer during the training , computed only over non-embedding and non-LM-head parameters.

Methods	Size	Q-Cfg	WikiText-2 PPL (AdamW / Ours)	C4 PPL (AdamW / Ours)	Avg. PPL (AdamW / Ours)	Opt. State Size (AdamW / Ours)
QuEST	200M	W2A2	50.56 / 50.98	26.99 / 27.41	38.78 / 39.20	1.63 / **0.44** GB ↓ 73.0%
		W2A16	44.04 / 42.80	22.92 / 23.41	33.48 / 33.11	
		W4A4	40.87 / 41.93	22.06 / 22.46	31.46 / 32.20	
CAGE		W2A2	51.15 / 52.32	27.19 / 27.65	39.17 / 39.99	
		W2A16	44.80 / 44.44	23.18 / 23.73	33.99 / 34.09	
		W4A4	41.22 / 42.68	22.21 / 22.61	31.71 / 32.66	
QuEST	430M	W2A2	39.12 / 41.48	21.69 / 22.11	30.40 / 31.80	3.54 / **0.94** GB ↓ 73.4%
		W2A16	34.74 / 34.66	18.79 / 19.18	26.77 / 26.92	
		W4A4	34.00 / 34.13	18.16 / 18.39	26.08 / 26.26	
CAGE		W2A2	40.49 / 41.24	21.81 / 22.25	31.15 / 31.75	
		W2A16	35.93 / 35.29	19.08 / 19.42	27.50 / 27.36	
		W4A4	34.55 / 34.13	18.30 / 18.57	26.43 / 26.35	

Table 2: Performance and memory footprint comparison between ResQAdam and standard AdamW across diverse QAT fine-tuning frameworks. Values in each cell are formatted as AdamW / ResQAdam (Ours). *Q-Cfg* denotes the quantizer configuration: Wx and Ax indicate x-bit weight and activation quantization, respectively; *G128* represents group-wise quantization with a group size of 128; and *C* denotes channel-wise quantization. *Opt. State Size* reports average optimizer state size of AdamW / ResQAdam during the training , computed only over non-embedding and non-LM-head parameters.

Method	Model	Q-Cfg	Avg. Task Acc. (AdamW / Ours)	Avg. PPL (AdamW / Ours)	Opt. State Size (AdamW / Ours)
Basic QAT	Llama 3.2-1B	W2A16G128	47.56 / 47.36	16.92 / 17.73	7.78 / **2.16** GB ↓ 72.2%
		W4A4G128	51.29 / 51.29	15.12 / 14.91	
LLM-QAT		W2A16C	38.26 / 39.37	43.73 / 35.05	7.78 / **2.16** GB ↓ 72.2%
		W4A16C	51.16 / 52.00	14.03 / 14.30	
		W4A4C	38.70 / 38.53	40.91 / 39.21	
EfficientQAT (Block-AP)		W2A16G128	35.14 / 35.43	351.78 / 379.52	494 / **141** MB/block ↓ 71.5%
		W3A16G128	49.93 / 49.31	16.73 / 16.79	
		W4A16G128	54.43 / 54.56	12.78 / 12.81	
Basic QAT	Qwen3 4B-Base	W2A16G128	61.29 / 61.94	12.76 / 12.88	29.07 / **7.78** GB ↓ 73.2%
		W4A4G128	68.21 / 67.91	12.27 / 12.20	
LLM-QAT		W2A16C	40.85 / 38.02	33.84 / 46.72	29.07 / **7.78** GB ↓ 73.2%
		W4A16C	68.07 / 66.81	11.68 / 11.93	
		W4A4C	45.70 / 45.19	35.46 / 31.32	
EfficientQAT (Block-AP)		W2A16G128	45.89 / 43.09	34.92 / 51.25	820 / **220** MB/block ↓ 72.0%
		W3A16G128	66.69 / 66.88	12.06 / 12.23	
		W4A16G128	69.29 / 69.49	10.88 / 10.94	
LLM-QAT	Llama 2-7B	W2A16C	43.03 / 42.31	12.25 / 13.93	51.81 / **13.28** GB ↓ 74.4%
		W4A16C	63.11 / 61.68	6.82 / 6.89	
		W4A4C	47.12 / 46.87	14.12 / 12.58	
EfficientQAT (Block-AP)		W2A16G128	49.10 / 47.95	24.61 / 12.91	1644 / **451** MB/block ↓ 72.6%
		W3A16G128	61.37 / 61.28	6.98 / 6.99	
		W4A16G128	63.77 / 63.76	6.49 / 6.49	

marks: HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020), ARC (Clark et al., 2018), WinoGrande (Sakaguchi et al., 2021), and MMLU (Hendrycks et al., 2021).

4.2 Main Results on From-Scratch QAT

Optimization Trajectory and Convergence Alignment. As illustrated in Figure 2, ResQAdam closely tracks the training-loss trajectory of the dense AdamW baseline. Across all evaluated bit-width configurations and both frameworks, the training loss curves of ResQAdam closely overlap with the baseline from initialization through to the final iterations. The zoomed-in callouts at the end of the training horizons further highlight that ResQAdam maintains stable convergence without divergence. This confirms that our geometry-decoupled and space-specific update strategies sufficiently capture the quantization errors to stabilize the low-bit pretraining process. Curves for the 200M model are in Appendix 6.

Language Modeling Quality. Table 1 reports the final validation perplexity across model scales and quantization settings. For both QuEST and CAGE, ResQAdam achieves perplexities close to full-state AdamW, with several cases where ResQAdam is slightly better. These results show that ResQAdam largely preserves AdamW-level language modeling quality in from-scratch QAT while substantially reducing optimizer-state memory.

4.3 Main Results on QAT Fine-Tuning

Table 2 compares ResQAdam with AdamW across the three fine-tuning recipes, which differ in data source, training objective and optimization granularity. Across these settings, ResQAdam generally achieves performance close to AdamW in both perplexity and zero-shot accuracy, while reducing optimizer-state memory by 71% to 74%. Although some aggressive W2A16 block-wise settings show larger metric fluctuations, the overall results indicate that ResQAdam can serve as a memory-efficient alternative to AdamW for diverse QAT fine-tuning pipelines.

5 ANALYSIS

5.1 COMPONENT ABLATION AND SUBSPACE DESIGN

We ablate the main components of ResQAdam on a 297M model under W2A2, W2A16, and W4A4 from-scratch QAT. To keep the ablation aligned with the goal of memory-efficient QAT, we restrict the comparison to lightweight variants that do not introduce substantial extra computation or optimizer-state memory. Variants whose added cost would eliminate the memory or throughput advantage over AdamW are excluded, since they would not be practical alternatives to AdamW.

Principal Update Only: Uses only the Adam-preconditioned update in the principal subspace, i.e., $\Delta_t = \Delta_t^{\parallel}$, and removes the sign correction from the orthogonal residual.

Residual-Sign Only: Uses only the normalized sign direction derived from the orthogonal residual $M_t^{\perp}$, and removes the Adam-preconditioned principal update.

Gradient-SVD Bases: Computes the projection bases U_t, V_t from the instantaneous gradient G_t instead of the momentum M_t, while keeping the remaining update rule unchanged.

Global-Momentum Sign: Applies a sign update directly to the momentum M_t without decomposing it into the principal component and the orthogonal residual.

ResQAdam: Uses the full design: Adam-style preconditioning in the $r \times r$ coordinate representation, residual-sign correction from $M_t^{\perp}$, and momentum energy-weighting to fuse the two updates.

Figure 3 reports the average perplexity on C4 and WikiText-2. The Principal Update Only variant performs poorly across all quantization settings, showing that updates restricted to the principal subspace are not sufficient for QAT. The Residual-Sign Only variant performs much better, especially under W2A2 and W2A16, which indicates that the orthogonal residual contains useful correction signals. However, it is not consistently better than the full method, suggesting that the Adam-preconditioned principal update still provides important low-rank structural information.

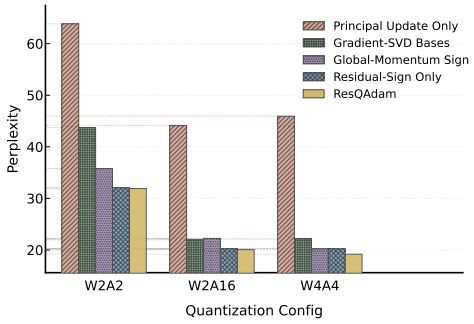


Figure 3: Component ablation of ResQAdam across different quantization configurations. The y-axis reports the average perplexity on C4 and WikiText-2.

The Gradient-SVD Bases variant underperforms ResQAdam, particularly under W2A2, validating the extraction of projection bases from the time-smoothed momentum rather than instantaneous gradients. Similarly, the Global-Momentum Sign variant falls short, demonstrating that applying a monolithic sign update negates the benefits of separating the principal component from the residual. Overall, the full ResQAdam consistently achieves the best perplexity across the evaluated settings, supporting the effectiveness of combining principal-subspace Adam preconditioning with a residual-derived sign correction.

5.2 HARDWARE EFFICIENCY AND OVERHEAD

To evaluate hardware efficiency, we compare ResQAdam against AdamW in terms of optimizer-state memory, total peak memory, and training throughput. Total peak memory is defined as the maximum GPU memory consumed during training, encompassing weights, activations, gradients, and optimizer states. All experiments utilize the Basic QAT framework with a W4A4G128 quanti-

Table 3: Comparison of optimizer-state memory, total peak memory, and training throughput across different models. Values in each cell are formatted as *AdamW / ResQAdam*, with relative changes in parentheses.

AdamW / ResQAdam	**Llama 3.2-1B**	**Qwen2.5-1.5B**	**Qwen3 4B-Base**
Opt. State Memory (GB)	7.78 / 2.16 (↓72.2%)	10.48 / 3.01(↓71.3%)	29.07 / 7.78 (↓73.2%)
Total Peak Memory (GB)	33.33 / 28.24 (↓15.3%)	43.93 / 37.19 (↓15.3%)	99.14 / 79.77 (↓19.5%)
Throughput (# token/s)	29.36k / 28.94k (↓1.4%)	18.31k / 18.19k (↓0.6%)	9.64k / 9.56k (↓0.8%)

Table 4: Perplexity comparison of memory-efficient optimizers under diverse QAT paradigms and quantization configurations. Projection-based optimizers use rank 64. For a fair comparison, all optimizers are tuned with a grid search over learning rates, and the best result is reported.

Training Paradigm	Q-Cfg	ResQAdam	GaLore-AdamW	GaLore-AdamW8bit	LDAdam	AdaFactor
From-Scratch QAT	W2A16G128	**20.05**	30.61	30.39	28.57	33.01
	W4A4G128	**18.89**	32.93	32.45	28.23	32.84
QAT Fine-Tuning	W2A16G128	**16.89**	19.25	20.32	19.59	27.79
	W4A4G128	15.05	15.30	15.64	15.42	**14.80**

zation configuration and a context length of 4096. Training is conducted on a single NVIDIA H200 GPU. The micro-batch size is set to 1 with 128 gradient accumulation steps. ResQAdam uses a uniform projection rank of $r = 64$. The reported optimizer-state memory strictly accounts for the transformer backbone and excludes the embedding and language modeling head layers.

As detailed in Table 3, ResQAdam consistently reduces the optimizer state footprint by over 71% across 3 different LLMs. This efficiency directly translates to a 15.3% to 19.5% reduction in total peak memory. This substantial memory reduction incurs minimal computational overhead. The training throughput decreases by less than 2% compared to standard AdamW in all evaluated cases. These results show that our amortized update strategy substantially alleviates the optimizer-state memory bottleneck of QAT while maintaining competitive training throughput.

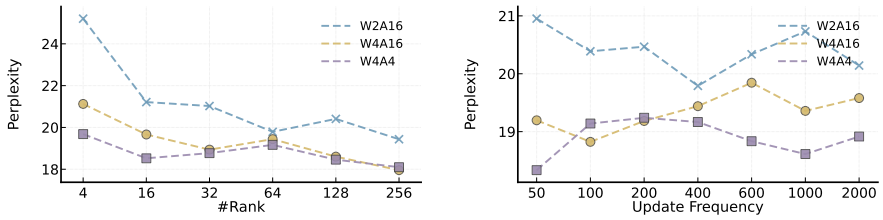
5.3 Comparison with Alternative Memory-Efficient Optimizers

We further compare ResQAdam with two families of memory-efficient optimizers. Low-rank projection methods such as GaLore (Zhao et al., 2024) and LDAdam (Robert et al., 2024) restrict adaptive states to a rank-r subspace obtained from the gradient, with LDAdam adding state transport across changing subspaces and error feedback. Structural factorization methods such as AdaFactor (Shazeer & Stern, 2018) keep the full parameter dimensionality but approximate the second moment by row and column statistics.

Table 4 reports the average perplexity on WikiText-2 and C4. In from-scratch QAT, every baseline is far behind ResQAdam in both quantization settings, by a margin much larger than the differences we observe between ResQAdam and AdamW in Table 1. This is consistent with our hypothesis that confining the update to a low-rank subspace, or replacing the second moment by a factorized approximation, loses part of the correction that QAT relies on. In fine-tuning the picture depends on the bit width. At W2A16G128 the projection-based methods remain clearly behind, whereas at W4A4G128, where quantization error is small, all optimizers land within a narrow range and the choice of optimizer matters little.

5.4 Sensitivity to r and τ

The rank r sets the dimension of the principal subspace, and the refresh interval τ sets how many steps a subspace is used before the bases are recomputed. We vary one of them while fixing the other at $r = 64$ or $\tau = 400$, using the 297M model trained from scratch with Basic QAT under W2A16, W4A16 and W4A4 (Figure 4).



(a) Rank r ($\tau = 400$). (b) Refresh interval τ ($r = 64$).

Figure 4: Sensitivity of ResQAdam to the rank r and the SVD refresh interval τ on a 297M model trained from scratch. The y-axis is the average of WikiText-2 and C4 perplexity.

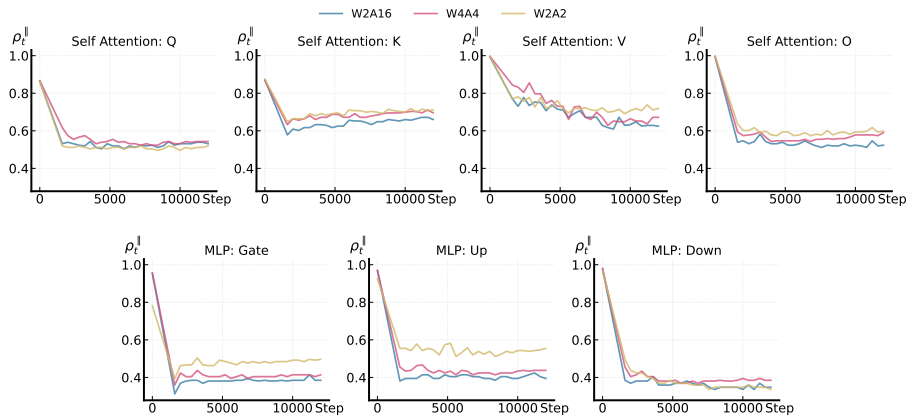


Figure 5: Energy ratio of the principal subspace ($\rho_t^{\|}$) during from-scratch QAT on a 297M Llama-style model. The curves represent different weight matrices within a middle transformer layer across W2A16, W4A4, and W2A2 quantization configurations.

Rank. A rank of 4 gives the highest perplexity in all three settings. The effect is largest at W2A16, where perplexity reaches 25.2 compared with 19.8 at $r = 64$. For $r \geq 16$, perplexity no longer decreases monotonically with r. Within each setting it varies by at most 1.8 over $r \in \{16, 32, 64, 128, 256\}$, and $r = 256$ gives the lowest value in all three settings. A very small subspace therefore hurts clearly, while above $r = 16$ the effect of r is small and not monotonic. The main from-scratch experiments use $r = 32$, which is 1.2 higher than $r = 64$ at W2A16 and 0.4 to 0.5 lower at W4A16 and W4A4.

Refresh interval. Training converges for every τ from 50 to 2000. The best interval differs across settings, namely $\tau = 400$ at W2A16 (19.8), $\tau = 100$ at W4A16 (18.8), and $\tau = 50$ at W4A4 (18.3). Within each setting, perplexity varies by about 1 over the whole range, and it shows no consistent trend with τ. Refreshing more often than every 400 steps therefore does not give a consistent gain, so the SVD cost can be spread over hundreds of steps. We use $\tau = 400$ for from-scratch QAT, which is within 0.9 of the best interval in every setting.

5.5 Dynamics of Momentum Energy Distribution

We track the energy ratio of the principal subspace, $\rho_t^{\|} = \|M_t^{\|}\|_F / (\|M_t\|_F + \epsilon)$, for the seven weight matrices of a middle layer in the 297M model pretrained with ResQAdam. We record it only at refresh steps, right after the bases are recomputed from M_t. At these points $\rho_t^{\|}$ is at its highest, so any residual energy we observe is a lower bound on the energy that a projection-based optimizer discards during the following τ steps.

The result is shown in Figure 5. $\rho_t^{\|}$ starts high, decreases, and settles at a level that depends on the matrix. It remains relatively high for attention K and V and is lowest for the MLP down projection, while the bit-width setting has a smaller effect. Even immediately after a refresh, a substantial share of the momentum lies outside $\mathcal{S}_t$, and projection-based optimizers discard this share.

Overall, the principal subspace consistently captures a meaningful but incomplete fraction of the momentum. This supports ResQAdam's selective-adaptivity design: Adam-style preconditioning is beneficial for the structured principal component, while a full-space residual correction remains necessary to preserve the substantial optimization signal outside the low-rank subspace.

6 Conclusion

In this paper, we proposed ResQAdam, a memory-efficient optimizer tailored for LLM QAT. We demonstrated that standard low-rank optimizers struggle in QAT because they discard important residual optimization signals outside the dominant principal subspace. ResQAdam leveraged Adam within this subspace and adds a sign update from the residual. Extensive experiments across from-scratch QAT and fine-tuning show that ResQAdam closely tracks AdamW's convergence behavior and largely preserves its final performance, while reducing persistent optimizer-state memory by 71% to 74%, respectively.

AI USE STATEMENT

In this work, we used generative AI tools for polishing the writing, correcting grammar, assisting with literature search and figure plotting, and locating code bugs. We have not used generative AI tools for developing the core ideas, methods, or contributions of this work. We have reviewed all AI-assisted work: polished text was checked for accuracy, retrieved literature was verified against the original sources, figures were checked against the experimental data, and suggested bug fixes were manually tested before adoption. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work focuses on methodological improvements and does not involve human subjects, personally identifiable information, or the collection of new data. All datasets and models used in our experiments are publicly available and were used in accordance with their respective licenses. We do not foresee any direct negative societal impact or ethical concerns arising from this work. The authors declare no conflicts of interest.

REPRODUCIBILITY STATEMENT

We have made efforts to ensure the reproducibility of this work. The details of proposed method are described in Section 3. Implementation details, including hyperparameters, training settings, and hardware configurations, are provided in Section 4.1 and Appendix C. The datasets and models used in our experiments are publicly available. We will release the source code publicly upon acceptance.

REFERENCES

AI@Meta. Llama 3 model card. 2024. URL `https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md`.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

Mengzhao Chen, Wenqi Shao, Peng Xu, Jiahao Wang, Peng Gao, Kaipeng Zhang, and Ping Luo. Efficientqat: Efficient quantization-aware training for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10081–10100, 2025.

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? *Advances in Neural Information Processing Systems*, 38:120680–120712, 2026.

Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. Pact: Parameterized clipping activation for quantized neural networks. *arXiv preprint arXiv:1805.06085*, 2018.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Together Computer. Redpajama: An open source recipe to reproduce llama training dataset, 2023. URL `https://github.com/togethercomputer/RedPajama-Data`.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115, 2023.

Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. *arXiv preprint arXiv:1902.08153*, 2019.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

Jeonghoon Kim, Jung Hyun Lee, Sungdong Kim, Joonsuk Park, Kang Min Yoo, Se Jung Kwon, and Dongsoo Lee. Memory-efficient fine-tuning of compressed large language models via sub-4-bit integer quantization. *Advances in Neural Information Processing Systems*, 36:36187–36207, 2023.

Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. Llm-qat: Data-free quantization aware training for large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 467–484, 2024.

Zechun Liu, Changsheng Zhao, Hanxian Huang, Sijia Chen, Jing Zhang, Jiawei Zhao, Scott Roy, Lisa Jin, Yunyang Xiong, Yangyang Shi, et al. Paretoq: Improving scaling laws in extremely low-bit llm quantization. *Advances in Neural Information Processing Systems*, 38:91311–91336, 2026.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Shuming Ma, Hongyu Wang, Lingxiao Ma, Lei Wang, Wenhui Wang, Shaohan Huang, Li Dong, Ruiping Wang, Jilong Xue, and Furu Wei. The era of 1-bit llms: All large language models are in 1.58 bits. *arXiv preprint arXiv:2402.17764*, 2024.

Andrei Panferov, Jiale Chen, Soroush Tabesh, Roberto L Castro, Mahdi Nikdan, and Dan Alistarh. Quest: Stable training of llms with 1-bit weights and activations. *arXiv preprint arXiv:2502.05003*, 2025.

Thomas Robert, Mher Safaryan, Ionut-Vlad Modoranu, and Dan Alistarh. Ldadam: Adaptive optimization from low-dimensional gradient statistics. *arXiv preprint arXiv:2410.16103*, 2024.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *International conference on machine learning*, pp. 4596–4604. PMLR, 2018.

Soroush Tabesh, Mher Safaryan, Andrei Panferov, Alexandra Volkova, and Dan Alistarh. Cage: Curvature-aware gradient estimation for accurate quantization-aware training. *arXiv preprint arXiv:2510.18784*, 2025.

Qwen Team. Qwen3 technical report, 2025. URL `https://arxiv.org/abs/2505.09388`.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Huaijie Wang, Lingxiao Ma, Fan Yang, Ruiping Wang, Yi Wu, and Furu Wei. Bitnet: Scaling 1-bit transformers for large language models. *arXiv preprint arXiv:2310.11453*, 2023.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International conference on machine learning*, pp. 38087–38099. PMLR, 2023.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 4791–4800, 2019.

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.

Philip Zmushko, Aleksandr Beznosikov, Martin Takáč, and Samuel Horváth. Frugal: Memory-efficient optimization by reducing state overhead for scalable training. *arXiv preprint arXiv:2411.07837*, 2024.

APPENDIX

A ESTIMATING THE STATISTICS OF PROJECTED GRADIENTS

When the projection bases are refreshed, the coordinate representation of the principal subspace changes from (U_{t-1}, V_{t-1}) to (U_t, V_t), where $t-1$ denotes the last step before the refresh. Since Adam-style preconditioning is anisotropic, directly reusing the old principal moments $m^{\|}_{t-1}$ and $v^{\|}_{t-1}$ in the new coordinate system would mix statistics computed under different bases. To address this, we adopt the statistical state transport mechanism introduced by LDAdam (Robert et al., 2024), which estimates the first- and second-order statistics of the newly projected gradients. As noted in Section 3.4, only the compact principal states are transported; we treat this transport as a lightweight heuristic that maintains the continuity of the Adam-preconditioned principal update across refreshes.

Recall that ResQAdam projects the full-space gradient $G_t \in \mathbb{R}^{n\times m}$ into the $r \times r$ coordinate representation of the principal subspace as

$$G_t^{\|} = U_t^\top G_t V_t \in \mathbb{R}^{r\times r}. \tag{8}$$

Let the old and new bases define the transition matrices

$$P_t = U_t^\top U_{t-1} \in \mathbb{R}^{r\times r}, \qquad Q_t = V_{t-1}^\top V_t \in \mathbb{R}^{r\times r}. \tag{9}$$

Under the rank-r truncation induced by the old bases, the projection of any past gradient G_k ($k < t$) into the new coordinate representation can be approximated by

$$U_t^\top G_k V_t \approx \left(U_t^\top U_{t-1}\right)\left(U_{t-1}^\top G_k V_{t-1}\right)\left(V_{t-1}^\top V_t\right) = P_t G_k^{\|} Q_t. \tag{10}$$

Because the optimizer states are essentially exponentially moving averages (EMAs) of these past gradients, this transformation forms the basis for transporting their statistics.

First-order statistic. The transport must be performed on the *bias-corrected* statistical moments rather than the raw EMAs. This ensures that the first and second moments are on the same true scale when estimating the variance. Because the first moment is linear in the gradient, the transported first-order statistic is obtained by applying the coordinate rotation to the bias-corrected old first moment:

$$\widehat{m}^{\|}_{t-\frac{1}{2}} = P_t \widehat{m}^{\|}_{t-1} Q_t. \tag{11}$$

Equivalently, the raw transported first moment is

$$m^{\|}_{t-\frac{1}{2}} = \left(1-\beta_1^{t-1}\right)\widehat{m}^{\|}_{t-\frac{1}{2}}. \tag{12}$$

Second-order statistic. The second moment $v^{\|}$ is an element-wise squared-gradient EMA rather than a true covariance matrix, so it is not linear under a change of coordinates. For each new coordinate (a, b), the corresponding projected gradient is approximated as

$$\left(G_k^{\|,\text{new}}\right)_{ab} \approx \sum_{i=1}^{r}\sum_{j=1}^{r}(P_t)_{ai}(Q_t)_{jb}\left(G_k^{\|,\text{old}}\right)_{ij}. \tag{13}$$

Assuming coordinate-wise independence as a heuristic, its second-order statistic can be estimated by transporting the variance via the squared transition matrices and re-adding the squared transported mean:

$$\left(\widehat{v}^{\|}_{t-\frac{1}{2}}\right)_{ab} \approx \sum_{i=1}^{r}\sum_{j=1}^{r}(P_t)_{ai}^2(Q_t)_{jb}^2\left[\left(\widehat{v}^{\|}_{t-1}\right)_{ij} - \left(\widehat{m}^{\|}_{t-1}\right)_{ij}^2\right] + \left(\widehat{m}^{\|}_{t-\frac{1}{2}}\right)_{ab}^2. \tag{14}$$

The first term estimates the transformed coordinate-wise variance, while the second term accounts for the squared transformed mean. In practice, to avoid numerical artifacts caused by this independence approximation, we clip the transported second-order estimate to be non-negative:

$$\widehat{v}^{\|}_{t-\frac{1}{2}} \leftarrow \max\left(\widehat{v}^{\|}_{t-\frac{1}{2}}, 0\right). \tag{15}$$

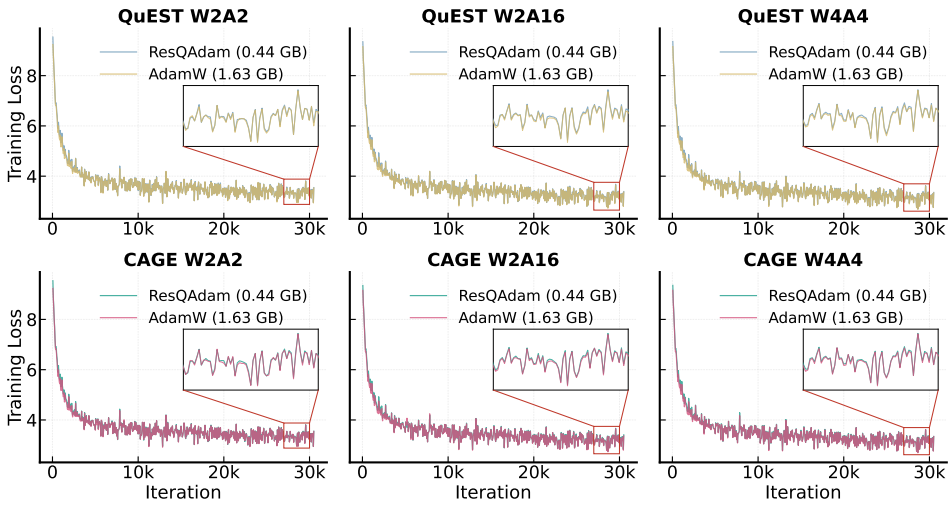


Figure 6: Training loss trajectories of ResQAdam and AdamW during from-scratch QAT under various quantization configurations on the 200M-parameter LLM. The values following the optimizer names in the legend indicate the average memory footprint of the optimizer states.

The raw transported second moment is then recovered by reapplying the bias term:

$$v^{\parallel}_{t-\frac{1}{2}} = \left(1-\beta_2^{t-1}\right)\widehat{v}^{\parallel}_{t-\frac{1}{2}}. \tag{16}$$

After this transport step, ResQAdam continues the principal Adam update in the refreshed coordinate representation:

$$G^{\parallel}_t = U_t^{\top} G_t V_t, \tag{17}$$

$$m^{\parallel}_t = \beta_1 m^{\parallel}_{t-\frac{1}{2}} + (1-\beta_1) G^{\parallel}_t, \tag{18}$$

$$v^{\parallel}_t = \beta_2 v^{\parallel}_{t-\frac{1}{2}} + (1-\beta_2)\left(G^{\parallel}_t \odot G^{\parallel}_t\right). \tag{19}$$

Difference in Application Scope. While the statistical transport equations strictly follow LDAdam (Robert et al., 2024), our application differs in scope. LDAdam relies on this transport to perform adaptive optimization entirely within the compressed principal subspace, using generalized error feedback to reintroduce projection errors. In contrast, ResQAdam restricts this transport solely to the compact principal states $m^{\parallel}_t, v^{\parallel}_t \in \mathbb{R}^{r\times r}$. We maintain a dense momentum M_t in the original parameter space (stored in BF16) and explicitly preserve the discarded orthogonal residual $M_t^{\perp}$ through a full-space sign correction. Thus, our use of the projected-gradient statistic estimation serves purely as a continuity mechanism for the principal update, deliberately avoiding the compression of residual QAT compensations into LDAdam-style dense second-moment tracking.

B Training loss trajectories on 200M-parameter LLM

Figure 6 shows the optimization trajectory and convergence alignment during training for a 200M-parameter LLM, comparing ResQAdam and AdamW under two from-scratch QAT methods, QuEST and CAGE. For the 200M-scale model, ResQAdam exhibits a training-loss trajectory that closely tracks that of AdamW.

C QAT Frameworks and Experimental Configurations

We evaluate ResQAdam across both from-scratch QAT and post-pretraining QAT fine-tuning, covering substantially different optimization objectives, training schemes, and quantization settings.

C.1 FROM-SCRATCH QAT

We consider two recent low-precision pretraining frameworks.

QuEST. QuEST (Panferov et al., 2025) applies Hadamard transformations to weights and activations before quantization and uses a trust-based gradient estimator to reduce gradient bias from large quantization errors. We replace its original optimizer with ResQAdam while keeping the remaining training pipeline unchanged.

CAGE. CAGE (Tabesh et al., 2025) augments the STE with a curvature-aware correction derived from a multi-objective formulation that jointly considers task loss and quantization error. ResQAdam is integrated without modifying its quantization or gradient-correction scheme. For both frameworks, we pretrain Llama-style models with 200M and 430M parameters on C4 under W2A2, W2A16, and W4A4 quantization. The training-token budget is set to $40\times$ the model size.

C.2 QAT FINE-TUNING

We further evaluate ResQAdam under three QAT fine-tuning frameworks with different optimization objectives and data sources.

Basic QAT. Our Basic QAT baseline performs end-to-end next-token training with learnable asymmetric quantization, where quantization scales are optimized following LSQ (Esser et al., 2019). All parameters are jointly fine-tuned on a 2B-token subset of RedPajama (Computer, 2023).

EfficientQAT (Block-AP). EfficientQAT (Chen et al., 2025) first performs block-wise all-parameter training (Block-AP), followed by an E2E-QP stage that updates only quantization parameters. Since ResQAdam operates on weight-matrix updates, we evaluate it only in the Block-AP stage.

LLM-QAT. LLM-QAT (Liu et al., 2024) uses data-free knowledge distillation, generating training sequences from the full-precision teacher and optimizing the quantized student to match it. It jointly quantizes weights, activations, and the KV cache using symmetric MinMax quantization. For fine-tuning experiments, we use Llama 3.2-1B (AI@Meta, 2024), Qwen 3-4B-Base (Team, 2025), and Llama 2-7B (Touvron et al., 2023) under multiple low-bit quantization configurations. Exact configurations are reported with the corresponding experimental results.