READAM: REFORMULATING ADAM FOR LOW-BIT QUANTIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Adaptive optimizers like Adam are very effective for model training, adaptively adjusting the learning rate for each learnable parameter using first- and second-order gradient statistics. However, storing these moments substantially increases Adam's memory footprint. Existing memory-efficient optimizers aim at preserving Adam's optimizer state representation and quantizing the first- and second-order moments independently. However, these raw moments are unbounded and prone to outliers, thus requiring more elaborate and memory-intensive quantization schemes. We pose a different question: *instead of improving how Adam's existing optimizer state can be quantized, can the optimizer state itself be reparameterized to be more amenable to quantization?* To this end, we derive a new scale-normalized reformulation of Adam in READAM by separating the optimizer state into (i) a gradient scale and (ii) a normalized quantity corresponding directly to Adam's effective update. We establish that the square root of the second moment is a particularly effective choice for the scale. READAM is an equivalent reformulation of Adam using these two new optimizer coordinates, that are much more compression-friendly compared to the raw moments of Adam. READAM performs on par with or better than existing low-bit optimizers across multiple benchmarks and modalities (text, speech and vision), while also reducing optimizer-state memory usage by 6% (relative).

1 INTRODUCTION

Adam (Kingma & Ba, 2015; Loshchilov & Hutter, 2019) is arguably the most popular adaptive optimizer in modern machine learning. Adam maintains moving averages of gradients and squared gradients (i.e., first- and second-order moments) for every learnable parameter, which are subsequently used to make per-parameter adjustments. However, storing these two moment tensors as the optimizer state incurs substantial memory costs, thus limiting the batch sizes that can be used to train models. This makes optimizer state compression an important problem to tackle for memory-efficient learning.

A natural approach towards reducing optimizer memory is to compress the raw moment tensors. The most well-known prior work in this direction uses blockwise quantization to design 8-bit optimizers that perform at par with unquantized 32-bit optimizers (Dettmers et al., 2022b). More recent work (Li et al., 2023) has further reduced optimizer state precision to 4 bits through careful outlier handling and partitioning of the moment tensors into blocks.

Quantizing Adam's first- and second-order moments directly is challenging given the nature of the moments. Their absolute magnitudes depend on the gradient scales, which can vary widely across parameters. The quantizer must therefore account for outliers through scaling factors, fine-grained moment partitioning or other auxiliary information, which adds to memory and computation costs. This additional overhead raises the question of whether the raw moments need to be reconstructed accurately at all. Adam's parameter update involves the first moment normalized by the square root of the second moment. Hence, an accurate reconstruction of the raw moments themselves may not be necessary as long as the effective update can be reconstructed.

This motivates reconsidering the representation of Adam's optimizer state itself. Instead of storing two scale-dependent moment tensors, can we reformulate the optimizer state to store a gradient scale separately from Adam's effective update? To this end, we introduce READAM, a scale-normalized reformulation of Adam well suited to quantization. READAM redefines Adam's optimizer state in

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.

terms of (i) a coordinate-wise gradient scale and (ii) Adam's effective update. We choose the scale to be the square root of the bias-corrected second-order moment. This quantity appears in Adam's original formulation as a denominator term normalizing the first-order moment. For this equivalent reformulation of Adam, we derive recursive expressions for both the scale and the effective update as functions of the incoming gradient relative to the previous scale (§ 3.2).

The two new optimizer state coordinates in READAM– the gradient scale and Adam's effective update – exhibit desirable compression properties. By design, the scale retains gradient-magnitude information but it can be compressed effectively using existing quantization approaches such as Adafactor (Shazeer & Stern, 2018) (§ 3.2.1). For standard values of the first- and second-moment decay rates, Adam's effective update admits bounds independent of the gradient magnitude and remains within a range suitable for low-bit quantization (Kingma & Ba, 2015) (§ 3.2.2). Beyond compression, we also show how scale normalization clarifies the role of Adam's historical gradients on its effective update (§ 3.3).

We empirically demonstrate that the new optimizer state coordinates of READAM can be represented using 4 bits with further memory cost reductions (compared to prior work on 4-bit compression Li et al. (2023)). READAM matches baseline Adam performance on a diverse set of tasks and multiple modalities (text, speech and vision), while being substantially more memory-efficient (§ 4).

2 RELATED WORK

Memory-efficient optimizers. A long line of prior work targets Adam's memory costs by storing a reduced set of statistics. Techniques like Adafactor (Shazeer & Stern, 2018), SM3 (Anil et al., 2019), extreme tensoring (Chen et al., 2020), CAME (Luo et al., 2023) and Adam-mini (Zhang et al., 2025) explore memory reductions through sharing or factorizing second-moment estimates. Existing techniques such as SMMF (Park & Lee, 2025) and AdaLomo (Lv et al., 2024) extend factorization to both moments and couple factorized statistics with memory-efficient parameter updates, respectively. Rather than focusing on moment sharing or factorization, other prior work aims to explicitly compress gradient history via low-rank projections (Zhao et al., 2024; Nguyen & Nguyen, 2025) or compression schemes with error feedback (Modoranu et al., 2024). A common motivation behind these techniques is identifying what information to retain from gradient histories for effective optimization. Other compression techniques reduce storage costs through pruning (Frantar et al., 2022; Yao et al., 2022; Dettmers et al., 2022a), quantized low-rank adaptation (Dettmers et al., 2023; Xu et al., 2024); systems-driven compression approaches reduce storage costs via state sharding and recomputing activations (Rajbhandari et al., 2020; Chen et al., 2016) or reducing communication costs as in 1-bit Adam (Tang et al., 2021). READAM adopts a different line of attack, namely reformulating Adam's optimizer states so as to obviate the need for complex moment sharing or factorization schemes.

Quantization of optimizer states. The line of work most closely related to READAM explores the quantization of optimizer states. A key challenge here is to accurately represent gradient statistics, despite their outliers and varying ranges, using only a few bits. Dettmers et al. (2022b) uses block-wise quantization to design an 8-bit optimizer, while Li et al. (2023) pushes it down to 4 bits by carefully handling outliers and small second moments. SOLO Xu et al. (2025) targets even lower precision of 2–3 bits using logarithmic quantization and momentum hyperparameters. Combinations of quantization and factorization have also been explored Han et al. (2025). Quantization could affect convergence and optimizer dynamics, which have been theoretically characterized in recent work Tang et al. (2026); Topollai & Choromanska (2026). Unlike these prior works, READAM rethinks which quantities to store rather than how to quantize raw moments and introduces a new basis for low-bit quantization.

Revisiting Adam's updates. A rich body of work (including AdamW (Loshchilov & Hutter, 2019), Adan (Xie et al., 2024), AMSGrad (Reddi et al., 2018), AdaBelief (Zhuang et al., 2020), etc.) aims to revise Adam's update rule to improve training stability and generalization. More recent work, especially for LLM training, proposes substantial changes to the optimizer through matrix preconditioning or orthogonalized momentum (e.g., Shampoo (Gupta et al., 2018), Muon (Jordan et al., 2024), etc.). Simplified sign-based optimization techniques such as SignSGD (Bernstein et al., 2018) and Lion (Chen et al., 2023) are also popular. Balles & Hennig (2018) examines Adam's optimizer design space by disentangling gradient signs from their relative variance. READAM, in contrast, preserves Adam's original update rule while expressing its optimizer states using new coordinates.

3 OUR METHOD: READAM

3.1 A QUICK OVERVIEW OF ADAM

Adam (Kingma & Ba, 2015) uses historical gradient estimates, namely exponentially weighted moving averages (EMAs) of past gradients and their element-wise squares, to adaptively determine parameter updates. Consider a parameter matrix $\mathbf{\Theta} \in \mathbb{R}^{n \times m}$ optimized with respect to a loss function $\mathcal{L}$ over T optimization steps. At any step $t \in \{1,2,...,T\}$, let $\mathbf{\Theta}_t$ denote the current parameters and $\mathbf{G}_t = \nabla_{\mathbf{\Theta}_t} \mathcal{L}$ denote the instantaneous mini-batch gradient. Adam maintains the EMAs of the gradient $\mathbf{G}_t$ and its element-wise square $\mathbf{G}_t^2$:

$$\mathbf{M}_t = \beta_1 \mathbf{M}_{t-1} + (1-\beta_1)\mathbf{G}_t \qquad \mathbf{V}_t = \beta_2 \mathbf{V}_{t-1} + (1-\beta_2)\mathbf{G}_t^2 \tag{1}$$

where $\beta_1, \beta_2 \in [0,1)$ control the decay rates of the respective averages. Since both moment estimates are initialized to zero, Adam applies a bias correction to offset this effect as follows:

$$\hat{\mathbf{M}}_t = \mathbf{M}_t / (1-\beta_1^t) \qquad \hat{\mathbf{V}}_t = \mathbf{V}_t / (1-\beta_2^t) \tag{2}$$

Finally, $\mathbf{\Theta}_{t+1}$ is computed as:

$$\mathbf{\Theta}_{t+1} = \mathbf{\Theta}_t - \eta_t \mathbf{U}_t \qquad \text{where } \mathbf{U}_t = \frac{\hat{\mathbf{M}}_t}{\sqrt{\hat{\mathbf{V}}_t} + \epsilon} \tag{3}$$

where η_t is the learning rate at step t and $\epsilon > 0$ is a small constant added for numerical stability. We refer to $\mathbf{U}_t$ as the *effective update* induced by Adam in the parameter space at step t. For $\beta_1^2 < \beta_2$, $\mathbf{U}_t$ is coordinate-wise upper-bounded by a constant that is solely a function of the decay rates. This guarantees a fixed numerical range, which is further established empirically in Section 3.2.2.

3.2 REFORMULATING ADAM FOR QUANTIZATION (READAM)

An important observation from Section 3.1 is the mismatch between numerical ranges of Adam's raw moments ($\mathbf{M}_t, \mathbf{V}_t$) and its effective parameter update ($\mathbf{U}_t$). The raw moments (and their bias-corrected counterparts) are unbounded, exhibit wide ranges and are prone to outliers (Dettmers et al., 2022b; Li et al., 2023), whereas $\mathbf{U}_t$ is theoretically bounded and more well-behaved. This disparity arises because $\mathbf{U}_t$ is computed as the coordinate-wise ratio of $\hat{\mathbf{M}}_t$ to $\sqrt{\hat{\mathbf{V}}_t}$ (ignoring ϵ). Both $\hat{\mathbf{M}}_t$ and $\sqrt{\hat{\mathbf{V}}_t}$ scale by the same factor under a positive rescaling of the gradient history, which cancels out in their ratio. This makes the resulting normalized update $\mathbf{U}_t$ invariant to the absolute gradient scale (Kingma & Ba, 2015; Balles & Hennig, 2018). We refer to this as Adam's inherent *scale normalization*. Standard Adam maintains this normalization via the effective update in eq. (3). In READAM, we make this explicit in the optimizer state's representation: a scale transforms incoming quantities into normalized counterparts that are subsequently used to calculate $\mathbf{U}_t$. By maintaining the scale separately from the effective update in this manner, we isolate the absolute gradient magnitudes from the update recurrence, thus yielding a new optimizer state that is more amenable to low-bit quantization.

We begin by defining the scale. In Adam's effective update $\mathbf{U}_t = \hat{\mathbf{M}}_t / \sqrt{\hat{\mathbf{V}}_t}$ (ignoring ϵ), $\sqrt{\hat{\mathbf{V}}_t}$ is already used to normalize momentum. Thus, we choose the scale $\mathbf{S}_{t-1}$ to be $\sqrt{\hat{\mathbf{V}}_{t-1}}$ to align well with Adam's effective update. We can now define an equivalent reformulation of Adam in terms of the new optimizer state coordinates $\mathbf{U}_t$ and $\mathbf{S}_t$, instead of $\hat{\mathbf{M}}_t$ and $\hat{\mathbf{V}}_t$, through the following recurrence relations:

$$\mathbf{U}_t = \frac{(1-a_t)\mathbf{U}_{t-1} + a_t \mathbf{Z}_t}{\sqrt{(1-b_t) + b_t \mathbf{Z}_t^2} + \epsilon} \qquad \mathbf{S}_t = \mathbf{S}_{t-1} \odot \sqrt{(1-b_t) + b_t \mathbf{Z}_t^2}$$
$$\text{where} \quad \mathbf{Z}_t = \frac{\mathbf{G}_t}{\mathbf{S}_{t-1} + \epsilon},\ a_t = \frac{(1-\beta_1)}{(1-\beta_1^t)} \text{and}\ \ b_t = \frac{(1-\beta_2)}{(1-\beta_2^t)} \tag{4}$$

Substituting $\hat{\mathbf{M}}_t = \mathbf{S}_t \odot \mathbf{U}_t$ and $\hat{\mathbf{V}}_t = \mathbf{S}_t^2$ in the original moment recurrences results in recurrences in these new coordinates. The complete derivation is provided in Appendix A. For the initial step ($t = 1$),

we set $\mathbf{S}_1 = |\,\mathbf{G}_1\,|$ and $\mathbf{U}_1 = \texttt{sign}(\mathbf{G}_1)$, consistent with Adam's first-step calculations. During training, since $\mathbf{S}_t$ is computed through a multiplicative recurrence, if an entry of $\mathbf{S}_{t-1}$ becomes zero, then it remains zero indefinitely. Fortunately, for such cases, we can simply bypass the recurrence and directly update the scale using $\mathbf{S}_t = \sqrt{b_t} \cdot |\,\mathbf{G}_t\,|$. This follows because $\mathbf{S}_{t-1} = 0$ implies $\hat{\mathbf{V}}_{t-1} = 0$, causing the recursive formulation of the second moment to reduce to $\hat{\mathbf{V}}_t = b_t \cdot \mathbf{G}_t^2$.

Intuitively, Equation (4) can be interpreted as operating within a scale-normalized coordinate system, with $\mathbf{S}$ serving as the reference unit. The incoming raw gradient $\mathbf{G}_t$ is normalized by $\mathbf{S}_{t-1}$ to operate within these coordinates, where $\mathbf{U}_t$ evolves according to its own update dynamics. Adam's history can also be analyzed through this scale-normalized lens, which we explore in Section 3.3. More importantly, ignoring ϵ[1], Equation (4) is an exact reparameterization of Adam: it produces the same parameter updates while maintaining optimizer states $\mathbf{U}_t$ and $\mathbf{S}_t$ instead of $\mathbf{M}_t$ and $\mathbf{V}_t$. These new states are particularly attractive for quantization because $\mathbf{S}_t$ is directly derived from the second moment $\mathbf{V}_t$, for which several well-established compression techniques exist, while $\mathbf{U}_t$ is well-bounded (Kingma & Ba, 2015) thus permitting a simple quantization scheme. Our complete optimizer with quantization READAM is presented in algorithm 1; details of the quantization mechanisms are presented next.

Algorithm 1 READAM for a matrix parameter $\mathbf{\Theta}$ with factored scale and quantized update

1: **Inputs:** Initial parameters $\mathbf{\Theta}_0 \in \mathbb{R}^{n \times m}$, step sizes $\{\eta_t\}_{t=1}^T$, decay rates β_1,β_2, quantization function $\mathcal{Q}$, constant ϵ
2: Initialize $\mathbf{U}_0 = 0$, $\boldsymbol{r}_0 = 0$, $\boldsymbol{c}_0 = 0$
3: **for** $t = 1$ **to** T **do**
4: $\quad \mathbf{G}_t \leftarrow \nabla_{\mathbf{\Theta}_t} \mathcal{L}$
5: $\quad$ **if** $t = 1$ **then**
6: $\quad\quad \mathbf{U}_1 \leftarrow \texttt{sign}(\mathbf{G}_1)$
7: $\quad$ **else**
8: $\quad\quad \hat{\boldsymbol{r}}_{t-1} \leftarrow \boldsymbol{r}_{t-1} / (1 - \beta_2^{t-1})$
9: $\quad\quad \hat{\boldsymbol{c}}_{t-1} \leftarrow \boldsymbol{c}_{t-1} / (1 - \beta_2^{t-1})$
10: $\quad\quad \tilde{\mathbf{S}}_{t-1} \leftarrow \sqrt{\dfrac{(\hat{\boldsymbol{r}}_{t-1} \hat{\boldsymbol{c}}_{t-1}^T)}{(\mathbf{1}_n^T \hat{\boldsymbol{r}}_{t-1})}}$
11: $\quad\quad \mathbf{Z}_t \leftarrow \mathbf{G}_t / (\tilde{\mathbf{S}}_{t-1} + \epsilon)$
12: $\quad\quad a_t \leftarrow (1 - \beta_1) / (1 - \beta_1^t)$
13: $\quad\quad b_t \leftarrow (1 - \beta_2) / (1 - \beta_2^t)$
14: $\quad\quad \mathbf{U}_t \leftarrow \dfrac{(1 - a_t) \cdot \mathbf{U}_{t-1} + a_t \cdot \mathbf{Z}_t}{\sqrt{(1 - b_t) + b_t \cdot \mathbf{Z}_t^2} + \epsilon}$
15: $\quad$ **end if**
16: $\quad \boldsymbol{r}_t \leftarrow \beta_2 \boldsymbol{r}_{t-1} + (1 - \beta_2)(\mathbf{G}_t^2)\mathbf{1}_m$
17: $\quad \boldsymbol{c}_t \leftarrow \beta_2 \boldsymbol{c}_{t-1} + (1 - \beta_2)\mathbf{1}_n^T(\mathbf{G}_t^2)$
18: $\quad \mathbf{\Theta}_t \leftarrow \mathbf{\Theta}_{t-1} - \eta_t \mathbf{U}_t$
19: $\quad \mathbf{U}_t \leftarrow \mathcal{Q}(\mathbf{U}_t; \mathbf{Z}_t)$ $\quad \triangleright$ quantize to b bits
20: **end for**

3.2.1 COMPRESSING THE SCALE $\mathbf{S}_{t-1}$

Since the scale $\mathbf{S}_{t-1}$ is directly related to the second moment ($\mathbf{S}_{t-1} = \sqrt{\hat{\mathbf{V}}_{t-1}}$), compression techniques like Adafactor (Shazeer & Stern, 2018), originally designed for $\mathbf{V}_t$, can naturally be extended to compress $\mathbf{S}_{t-1}$. The core idea of Adafactor is to maintain a low-rank, specifically rank-1, approximation of the full second moment $\mathbf{V}_t$. Shazeer & Stern (2018) analytically show that this optimal rank-1 reconstruction is uniquely determined by its row and column sums (see Lemma 1 of Shazeer & Stern (2018) for details). Since these sums propagate linearly through the EMA equation, approximating the full second-moment EMA simplifies to maintaining two separate EMAs over the row and column sums of the squared gradients:

$$\boldsymbol{r}_t = \beta_2 \boldsymbol{r}_{t-1} + (1 - \beta_2)(\mathbf{G}_t^2)\mathbf{1}_m \qquad \boldsymbol{c}_t = \beta_2 \boldsymbol{c}_{t-1} + (1 - \beta_2)\mathbf{1}_n^T(\mathbf{G}_t^2) \tag{5}$$

where $\mathbf{1}_\ell = (1, \ldots, 1)^T \in \mathbb{R}^\ell$ denotes a column vector of ℓ ones. During the update calculation, these statistics are bias-corrected, $\hat{\boldsymbol{r}}_t = \boldsymbol{r}_t / (1 - \beta_2^t)$ and $\hat{\boldsymbol{c}}_t = \boldsymbol{c}_t / (1 - \beta_2^t)$, and then used to reconstruct $\mathbf{V}_t$: $\tilde{\mathbf{V}}_t = (\hat{\boldsymbol{r}}_t \hat{\boldsymbol{c}}_t{}^T) / (\mathbf{1}_n^T \hat{\boldsymbol{r}}_t)$. However, because the recurrence for $\mathbf{S}_t$ involves an element-wise square root, the row and column sums cannot propagate through it. As a result, Adafactor's mechanism is not directly applicable to compressing $\mathbf{S}_t$. But since we know $\mathbf{S}_t^2 = \hat{\mathbf{V}}_t$, we can instead compress $\mathbf{S}_t^2$, for which the factorization holds. The exact calculations are given in lines 8-10 and 16-17 of Algorithm 1. By compressing the scale this way, we reduce its memory footprint from $n \cdot m$ floating point values down to $n + m$.

Since READAM relies on $\mathbf{S}_{t-1}$ for normalization at step t, we reconstruct $\mathbf{S}_{t-1}$ at the beginning of the iteration (line 10 of Algorithm 1) using the existing EMAs ($\boldsymbol{r}_{t-1}$ and $\boldsymbol{c}_{t-1}$). Once $\mathbf{G}_t$ is normalized, the scale is no longer needed, and the row and column EMA updates can be scheduled independently

[1]The role of ϵ in both Adam and READAM is to prevent division-by-zero errors. Following prior literature (Balles & Hennig, 2018), we omit ϵ from theoretical derivations and reintroduce it in actual implementations.

of the parameter update; we do it towards the end of the iteration (lines 16-17). In contrast, Adafactor requires the EMA updates prior to normalization.

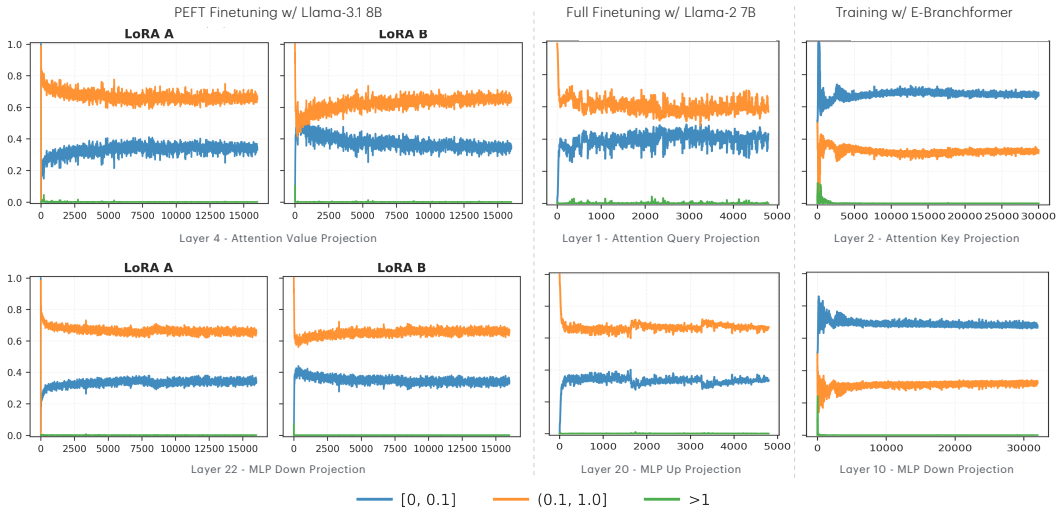

Figure 1: Plots illustrating the distribution of $|U_t|$ over training iterations, across three buckets: $[0,0.1]$ (–), $(0.1,1]$ (–) and >1 (–). The x-axis indicates training steps, and the y-axis shows the fraction of parameters falling within that bucket. These plots span three distinct training setups, capturing the behaviour of $\mathbf{U}_t$ across different layers and diverse architectural blocks.

3.2.2 Quantizing the effective update $\mathbf{U}_t$

Kingma & Ba (2015) claim that $\mathbf{U}_t$'s values are commonly within the interval $[-1,1]$. Empirically, we also observe that only a very small fraction of parameters satisfy $|\mathbf{U}_t| > 1$ (see bucket analysis in Figure 1). We leverage this observation and design our quantization scheme with representation levels spanning only the range $[-1,1]$. Although this clips per-coordinate updates to $[-\eta_t,\eta_t]$, we empirically find that such a clamping does not adversely affect training. With $[-1,1]$ as our target range and given a budget of b bits for each coordinate of $\mathbf{\Theta}$, we use the following simple quantization scheme. We reserve one bit for the sign, while the remaining $b-1$ bits provide 2^{b-1} possible magnitude levels, which are distributed across $[0,1]$ as follows: half of the levels are spaced uniformly in log-space between ϵ and 0.1, while the remaining levels are distributed linearly between 0.1 and 1, excluding 0.1. This simple scheme combines the finer resolution needed for the small fractional values with uniform linear resolution for larger magnitudes. Formally, for b bits with $N = 2^{b-2}$, we use the following magnitude levels:

$$\mathcal{Y}^+ = \left\{ \epsilon \left(\frac{0.1}{\epsilon} \right)^{\frac{k}{N-1}} : k = 0,\ldots,N-1 \right\} \cup \left\{ 0.1 + \frac{0.9m}{N} : m = 1,\ldots,N \right\} \tag{6}$$

These non-negative levels are mirrored to include both signs and sorted to form the full quantization map: $\mathcal{Y} = \texttt{Sort}(\{-y \,|\, y \in \mathcal{Y}^+\} \cup \mathcal{Y}^+)$.

There is a failure case that arises as a side effect of quantizing $\mathbf{U}_t$. When the gradient for a coordinate drops sharply towards zero (for example, near a local minima), Adam's moments cause the effective update to enter a decay regime, in which, ignoring ϵ, its magnitude decays by approximately a factor of $\beta_1/\sqrt{\beta_2}$ at each step. Under very low-bit quantization (for example, 4 bits), adjacent quantization levels are separated by relatively large gaps, so these small decay steps are insufficient to move the quantized representation of $\mathbf{U}_t$ to the next level closer to zero. As a result, the quantized update remains stuck at its current level, producing undesirable updates to the coordinate and potentially destabilizing training. To address this, we introduce a decay ratio γ. If the absolute ratio between $\mathbf{U}_t$ and $\mathbf{Z}_t$ exceeds γ for a coordinate, we identify it as being in the decay regime and explicitly nudge its quantization state to the next level toward zero. This simple correction prevents update stagnation and substantially stabilizes training under low-bit quantization. Thus, for any entry $u \in \mathbf{U}_t$ and its corresponding normalized

gradient $z \in \mathbf{Z}_t$, let $q(u) \in \operatorname{argmin}_{y \in \mathcal{Y}} |u - y|$ be the nearest quantization level and $\mathrm{next}_0(y)$ refer to the adjacent level toward zero (with $\mathrm{next}_0(0) = 0$). Then, our quantization scheme can be written as:

$$\mathcal{Q}(u; z) = \begin{cases} \mathrm{next}_0\big(q(u)\big), & |u| > \gamma |z| \\ q(u), & \text{otherwise} \end{cases} \tag{7}$$

3.3 A DRIFT-BASED PERSPECTIVE OF SCALE-NORMALIZED ADAM

In this section, we reinterpret the history encoded in Adam's scale-normalized state using new quantities that we call *drifts*. This interpretation is not relevant to quantization, but it highlights two distinct effects of Adam's history: correction of its effective update and adaptation of its reference scale. We first unroll the recursive forms in Equation (1) and rewrite Adam's bias-corrected moments as:

$$\hat{\mathbf{M}}_t = \mathbf{G}_t - \mathbf{D}_t^m \qquad \hat{\mathbf{V}}_t = \mathbf{G}_t^2 - \mathbf{D}_t^v \qquad \mathbf{U}_t = \frac{\mathbf{G}_t - \mathbf{D}_t^m}{\sqrt{\mathbf{G}_t^2 - \mathbf{D}_t^v} + \epsilon}, \tag{8}$$

where the *gradient drift* $\mathbf{D}_t^m$ and *squared-gradient drift* $\mathbf{D}_t^v$ are:

$$\begin{aligned} \mathbf{D}_t^m &= \sum_{k=1}^{t-1} \omega_{k,t}(\beta_1)(\mathbf{G}_{t-k+1} - \mathbf{G}_{t-k}) \\ \mathbf{D}_t^v &= \sum_{k=1}^{t-1} \omega_{k,t}(\beta_2)\Big(\mathbf{G}_{t-k+1}^2 - \mathbf{G}_{t-k}^2\Big) \end{aligned} \tag{9}$$

with $\omega_{k,t}(x) = \frac{(x^k - x^t)}{(1-x^t)}$ acting as a temporal weighting coefficient.[2] Detailed derivations for the expressions in Equation (8) and Equation (9) are in Appendix B. From this reformulation, we observe that Adam's historical corrections to instantaneous gradient statistics are weighted sums of their first-order temporal differences. This decomposition reveals an implicit lag in Adam's moment estimates introduced by temporal averaging, thereby providing additional intuition for predictive or look-ahead optimization methods such as Adan (Xie et al., 2024).

Using the scale $\mathbf{S}_t$, let us define normalized first- and second-moment drifts $\mathbf{d}_t^m$ and $\mathbf{d}_t^v$:

$$\mathbf{d}_t^m = \frac{\mathbf{D}_t^m}{\mathbf{S}_t} \qquad \mathbf{d}_t^v = \frac{\mathbf{D}_t^v}{\mathbf{S}_t^2} \tag{10}$$

Normalized first-moment drift. Using the definitions of $\hat{\mathbf{M}}_t$ and $\mathbf{U}_t$ from 8 and ignoring ϵ, the normalized first-moment drift $\mathbf{d}_t^m$ in terms of the scale $\mathbf{S}_t$ and update $\mathbf{U}_t$ becomes:

$$\mathbf{d}_t^m = \frac{\mathbf{G}_t}{\mathbf{S}_t} - \mathbf{U}_t \tag{11}$$

Since $\mathbf{U}_t = \hat{\mathbf{M}}_t / \mathbf{S}_t$, this normalized drift $\mathbf{d}_t^m$ measures how strongly first-moment averaging modifies the current gradient after rescaling, thus making corrections comparable across coordinates with different gradient magnitudes.

Normalized second-moment drift. Rearranging the bias-corrected second-moment recurrence gives us the following definition of the normalized second-moment drift $\mathbf{d}_t^v$:

$$\mathbf{d}_t^v = \frac{\beta_2 - \beta_2^t}{1 - \beta_2} \left[1 - \left(\frac{\mathbf{S}_{t-1}}{\mathbf{S}_t} \right)^2 \right] \tag{12}$$

(The complete derivation is in Appendix B.) Thus, $\mathbf{d}_t^v$ measures just the relative change in scale.

Both the normalized drifts are invariant to a positive rescaling of the gradient history and together signify how Adam's history modifies its effective update in a scale-invariant coordinate system.

[2] Since $\beta_1, \beta_2 \in [0, 1)$, for any fixed t and $k \in \{1, \ldots t-1\}$, $\omega_{k,t}(\beta_1)$ and $\omega_{k,t}(\beta_2)$ are non-negative and non-increasing with lag k.

Table 1: Performance and optimizer-state memory overhead comparisons between READAM, full-precision AdamW (fp32) (Loshchilov & Hutter, 2019), 8-bit AdamW (Dettmers et al., 2022b), and various LPMM configurations (Li et al., 2023) across five datasets spanning speech, text, and image modalities. Models evaluated include (from left to right): E-Branchformer (38M), Llama3 with DoRA r=32 (54M), Llama3 with LoRA r=256 (453M), Llama2-7B (7B), and BLIP (224M). All numbers reported are averages across three random seeds. For each task, indicates the top-performing method, and subscripts denote uncertainties. Similarly, highlights the method with the maximum memory savings, and subscripts indicate percentage savings relative to the most efficient baseline (4-bit LPMM with Adafactor). For Commonsense Reasoning, the reported values are the averages across Arc-C, OBQA, and SIQA test splits and the detailed numbers are available in Table 4. DTQ and AF stand for Dynamic Tree Quantization and Adafactor, respectively.

Method	Librispeech-100h	Commonsense Reasoning	Metamath + GSM8k	Alpaca + MMLU	MS-COCO
(a) Performance					
	Test other (WER) ↓	**Average (Acc.) ↑**	**Test (Acc.) ↑**	**Test (Acc.) ↑**	**Test (CIDEr) ↑**
AdamW	$16.2_{\pm 0.11}$	$83.9_{\pm 0.40}$	$75.5_{\pm 1.34}$	$38.9_{\pm 1.89}$	$133.5_{\pm 0.4}$
8-bit AdamW	$16.4_{\pm 0.31}$	$84.3_{\pm 0.14}$	$74.5_{\pm 0.93}$	$38.2_{\pm 0.95}$	$133.4_{\pm 0.3}$
4-bit LPMM (DTQ)	$16.4_{\pm 0.15}$	$83.2_{\pm 0.54}$	$71.7_{\pm 0.53}$	$38.6_{\pm 4.30}$	$133.2_{\pm 0.7}$
4-bit LPMM (AF)	$16.5_{\pm 0.12}$	$83.7_{\pm 0.51}$	$75.2_{\pm 0.93}$	$38.3_{\pm 1.18}$	$133.4_{\pm 0.5}$
5-bit LPMM (DTQ)	$16.2_{\pm 0.08}$	$83.8_{\pm 0.60}$	$73.8_{\pm 0.38}$	$37.8_{\pm 1.37}$	$133.5_{\pm 0.3}$
5-bit LPMM (AF)	$16.3_{\pm 0.02}$	$84.3_{\pm 0.14}$	$75.0_{\pm 0.71}$	$38.9_{\pm 0.85}$	$\mathbf{133.6}_{\pm 0.4}$
4-bit READAM (ours)	$16.6_{\pm 0.05}$	$\mathbf{84.7}_{\pm 0.19}$	$\mathbf{76.5}_{\pm 0.40}$	$39.2_{\pm 1.27}$	$133.3_{\pm 0.1}$
5-bit READAM (ours)	$\mathbf{16.1}_{\pm 0.06}$	$84.5_{\pm 0.64}$	$76.1_{\pm 0.48}$	$\mathbf{42.1}_{\pm 1.09}$	$133.5_{\pm 0.1}$
(b) Optimizer-state memory ↓					
AdamW	293.5 MB	440.32 MB	3.38 GB	50.2 GB	1.67 GB
8-bit AdamW	75.88 MB	112.44 MB	878.1 MB	12.7 GB	435.9 MB
4-bit LPMM (DTQ)	40.14 MB	67.28 MB	452.6 MB	6.28 GB	233.9 MB
4-bit LPMM (AF)	24.47 MB	40.04 MB	236.6 MB	3.34 GB	117.6 MB
5-bit LPMM (DTQ)	76.49 MB	121.75 MB	884.7 MB	12.55 GB	437.4 MB
5-bit LPMM (AF)	42.80 MB	67.28 MB	452.6 MB	6.48 GB	224.4 MB
4-bit READAM (ours)	$\mathbf{23.35}$ MB$_{4.5\times}$	$\mathbf{38.30}$ MB$_{4.3\times}$	$\mathbf{222.2}$ MB$_{6.1\times}$	$\mathbf{3.15}$ GB$_{5.7\times}$	$\mathbf{110.9}$ MB$_{5.7\times}$
5-bit READAM (ours)	41.57 MB	65.54 MB	439.1 MB	6.29 GB	217.6 MB

4 EXPERIMENTS

Experimental Setup. We conduct experiments across a diverse range of tasks, modalities, model scales, and training paradigms: (i) from-scratch training of an E-Branchformer model (Kim et al., 2023) (38M trainable params) on the 100-hour LibriSpeech corpus (Panayotov et al., 2015) of English read audiobooks (speech) (ii) Parameter-Efficient Fine-Tuning (PEFT) of LLaMA-3-8B (Grattafiori et al., 2024) using DoRA (Liu et al., 2024) ($r = 32$, 54M trainable params) on the Commonsense Reasoning benchmark (Hu et al., 2023) for multiple-choice questions (text) (iii) PEFT of LLaMA-3-8B using LoRA (Hu et al., 2022) ($r = 256$, 453M trainable params) on the Metamath dataset (Yu et al., 2024), with evaluation on the GSM8k benchmark (Cobbe et al., 2021) for complex mathematical reasoning questions (text) (iv) full fine-tuning of Llama-2-7B (Touvron et al., 2023) (7B trainable params) on the Alpaca dataset (Taori et al., 2023), with evaluation on the MMLU benchmark (Hendrycks et al., 2021) (text) and (v) multimodal fine-tuning of BLIP (Li et al., 2022) (224M trainable paramets) on the MS-COCO image captioning task using the Karpathy splits (Karpathy & Fei-Fei, 2015) (text + image). All experiments are run on NVIDIA RTX A6000 (48GB) and NVIDIA B200 (180GB) GPUs using publicly available codebases and toolkits with their default hyperparameters [3]. Each experimental result reported is an average across three different seeds (with standard deviations). Upon publication,

[3]We ensure that all experiments for a particular setting are conducted on the same GPU and environment.

we will publicly release our code to ensure reproducibility and facilitate further research on optimizer compression.

Main Results. Table 1 compares the performance and optimizer memory footprint of READAM against full-precision AdamW, 8-bit AdamW, and various LPMM configurations across our evaluation settings. [4] Because LPMM is the strongest baseline for our work, we experiment with both 4- and 5- bit variants of their algorithm, using Dynamic Tree Quantization for both moments (denoted as DTQ) as well as a hybrid configuration combining DTQ for the first moment with Adafactor for the second moment. Across all settings, READAM consistently matches the performance of AdamW and other baseline methods while requiring substantially less memory. Minor performance gains are observed in specific settings. However, we attribute these to training stochasticity and do not consider them to be statistically significant. While achieving on-par performance, READAM requires only 7.2% of the optimizer memory required by full-precision AdamW and 28% of that required by 8-bit Adam. Compared with the most memory-efficient baseline, 4-bit LPMM with Adafactor, READAM achieves an additional 5.3% relative reduction in optimizer memory owing to our scale-normalized reformulation, which reduces the need for auxiliary tracking states. Finally, increasing the precision from 4-bit to 5-bit in both LPMM and READAM yields marginal to no performance gains while incurring a 49.4% increase in memory.

Ablations. To understand how quantizing and compression of our new states impact performance and memory savings, in Table 2, we show two ablation experiments on LibriSpeech-100h from-scratch training and Metamath+GSM8k PEFT finetuning by (i) compressing a subset of our states and (ii) varying the number of bits used in the quantization of $\mathbf{U}_t$. We observe that GSM8K performance improves at low bit-widths, which could be explained as follows. The quantization levels at low bit-widths are far apart, so the states remain at their current levels until the incoming gradient signal induces a sufficiently large change. This could produce larger-than-desired updates. For finetuning, this may not be very disruptive since training starts near a local minima with more stable gradients and scope for recovery. However, for training from scratch, its impact is more pronounced, leading to 2-bit quantization failing to converge and 3-bit quantization performing poorly. Thus, $b = 4$ provides a good balance between training stability and performance across varying tasks. Regarding memory usage, our implementation uses 2-, 4- or 8-bit compressions. So, the 3-bit experiment uses 4-bit storage, while the 5-bit, 6-bit experiments use 8-bit storage.

Table 2: Impact of compression and quantization of READAM's performance on Metamath + GSM8k and LibriSpeech 100h settings. These numbers are reported on a single seed.

Method	Librispeech-100h		Metamath + GSM8k	
	Test Other (WER) ↓	Mem. ↓	Test (Acc.) ↓	Mem. ↓
Full Precision AdamW	16.2	293.5 MB	74.9	3.38 GB
Full-Precision READAM	16.1	293.5 MB	74.8	3.38 GB
+ compressing scale only	16.1	154.5 MB	74.9	1.69 GB
+ quantizing ($b=4$) update only	16.6	168.7 MB	76.4	1.90 GB
Fully compressed READAM				
− w/ $b=2$ bits	no convergence		**78.0**	**112.6 MB**
− w/ $b=3$ bits	16.9	**23.32 MB**	77.5	222.1 MB
− w/ $b=4$ bits	16.6	23.35 MB	76.5	222.2 MB
− w/ $b=5$ bits	16.1	41.57 MB	76.5	439.1 MB
− w/ $b=6$ bits	16.1	41.69 MB	75.7	439.1 MB
− w/ $b=8$ bits	16.0	42.19 MB	75.1	439.2 MB

Additional Memory Analysis. To understand the memory requirements of READAM and the baselines when training even larger models, we simulate the amount of GPU memory required to maintain the optimizer states when training these models by closely following the mechanisms employed in the original implementation with their default hyperparameters. As seen in Table 3, the memory saved using READAM compared to LPMM becomes more substantial with larger model sizes. We note these numbers are only indicative; training such large models (which is beyond our available compute budget) requires sophisticated pipelines that may result in the empirical gains being slightly different.

[4] We also experimented with SOLO, but found it to have convergence issues while also not being memory efficient in comparison to READAM.

Table 3: Comparison of optimizer memory requirements between READAM and the baseline optimizers for models spanning from 1B to 680B parameters. highlights the method with the maximum memory savings. The numbers reported here are analytical estimates computed using each model's parameter metadata and closely following the mechanisms and default hyperparameters of each baseline.

Model	Params	AdamW		LPMM (4 bit)		READAM	Memory Saved
		full precision	8 bit	w/ DTQ	w/ Adafactor	4 bit	
Llama family of models							
−Llama-3.2-1B	1.24 B	9.21 GB	2.31 GB	1.22 GB	0.62 GB	0.58 GB	0.04 GB (6.45%)
−Llama-3.1-8B	8.03 B	59.83 GB	14.99 GB	7.95 GB	3.98 GB	3.75 GB	0.23 GB (5.78%)
−Llama-3-70B	70.55 B	525.7 GB	131.7 GB	69.82 GB	34.96 GB	32.91 GB	2.05 GB (5.86%)
−Llama-3.1-405B-Instruct	405.8 B	3023.8 GB	757.4 GB	401.6 GB	200.9 GB	189.1 GB	11.8 GB (5.87%)
Gemma family of models							
−Gemma-4-E4B	8.00 B	59.58 GB	14.93 GB	7.92 GB	3.98 GB	3.74 GB	0.24 GB (6.03%)
−Gemma-4-31B	31.27 B	233.0 GB	58.37 GB	30.95 GB	15.51 GB	14.60 GB	0.91 GB (5.87%)
Miscelleneous models							
−Qwen2.5-72B-Instruct	72.71 B	541.7 GB	135.7 GB	71.95 GB	36.02 GB	33.92 GB	2.10 GB (5.83%)
−C4ai-command-r-plus	103.8 B	773.4 GB	193.7 GB	102.7 GB	51.41 GB	48.40 GB	3.01 GB (5.85%)
−Snowflake-arctic-instruct	478.6 B	3565.7 GB	893.2 GB	473.6 GB	237.4 GB	223.5 GB	13.92 GB (5.86%)
−DeepSeek-V3	684.5 B	5100.2 GB	1277.8 GB	677.6 GB	340.6 GB	320.6 GB	19.92 GB (5.85%)

5 LIMITATIONS AND FUTURE WORK

There are several aspects of READAM that could be improved upon further. First, our bucketing analysis also shows that $\mathbf{U}_t$'s values occupy the quantization levels non-uniformly, with a small number of levels accounting for a large fraction of the values. This motivates non-uniform encoding schemes such as Huffman coding for $\mathbf{U}_t$, which exploit such distributional skewness by assigning shorter codes to frequently occurring levels, thereby reducing the average number of bits required for storing these values. However, variable-length representations introduce implementation challenges, since reading and writing individual values becomes less straightforward and can limit parallelization. Such ideas have been explored in the post-training quantization regime and could potentially be useful for READAM (Han et al., 2015). Another observation from our bucketing analysis is that the distribution of $\mathbf{U}_t$ evolves dynamically, both across parameters and throughout training. Therefore, instead of using a fixed, predetermined quantization scheme, one could allow the quantization levels to evolve during training, thereby maintaining a quantization map that better reflects $\mathbf{U}_t$'s distribution at any given time. This could be achieved either through periodic calibration checkpoints, where clustering is invoked, or through adaptive quantization schemes that learn the representation levels during training. However, these approaches introduce additional computation and memory overhead. Since our primary goal in this work was to develop a minimal yet effective quantization technique that maintains performance while being highly memory-efficient, we leave the exploration of such sophisticated approaches to future work. This variation in optimization dynamics across layers and components also raises the possibility of allocating the memory budget non-uniformly across the model. In particular, components that are more sensitive to quantization error can be allocated higher bit precisions, while less sensitive components could use fewer bits. Such an allocation could provide a better trade-off between performance and memory usage. Finally, our core principle of reformulating an optimizer to make its state more amenable to quantization can also be applied to other popular optimizers (such as (Gupta et al., 2018),(Jordan et al., 2024)), which we plan to explore as future work.

READAM can be combined with other orthogonal compression techniques, such as GaLore (Zhao et al., 2024) and Adam-mini (Zhang et al., 2025), to achieve further memory reductions. However, preliminary experiments in this direction (combining GaLore and READAM) indicate that the maximum benefit from such integrations materialize when the auxiliary tensors, like the projection matrices from GaLore, are also efficiently quantized. More exploration in this direction is left for future work.

AI USE STATEMENT

In this work, we used generative AI tools for editing text, formatting tables in LaTeX, some research ideation and related work discovery. We also used AI tools to generate code, which was thoroughly

verified and tested by a human author. We have carefully reviewed all AI-assisted work. We take complete responsibility for the final content of this work, including text or code produced with the aid of generative AI tools.

REFERENCES

Rohan Anil, Vineet Gupta, Tomer Koren, and Yoram Singer. Memory efficient adaptive optimization. *Advances in Neural Information Processing Systems*, 32, 2019.

Lukas Balles and Philipp Hennig. Dissecting adam: The sign, magnitude and variance of stochastic gradients. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 404–413. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/balles18a.html.

Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. signsgd: Compressed optimisation for non-convex problems. In *International conference on machine learning*, pp. 560–569. PMLR, 2018.

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

Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, et al. Symbolic discovery of optimization algorithms. *Advances in neural information processing systems*, 36:49205–49233, 2023.

Xinyi Chen, Naman Agarwal, Elad Hazan, Cyril Zhang, and Yi Zhang. Extreme tensoring for low-memory preconditioning. 2020, 2020.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in neural information processing systems*, 35: 30318–30332, 2022a.

Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 8-bit optimizers via block-wise quantization. In *International Conference on Learning Representations*, 2022b. URL https://openreview.net/forum?id=shpkpVXzo3h.

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.

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.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning*, pp. 1842–1850. PMLR, 2018.

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

Yizhou Han, Chaohao Yang, Congliang Chen, Xingjian Wang, and Ruoyu Sun. Q-adam-mini: Memory-efficient 8-bit quantized optimizer for large language model training. In *ES-FoMo III: 3rd Workshop on Efficient Systems for Foundation Models, ICML*, 2025.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Lee. LLM-adapters: An adapter family for parameter-efficient fine-tuning of large language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5254–5276, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.319. URL https://aclanthology.org/2023.emnlp-main.319/.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL https://kellerjordan.github.io/posts/muon/.

Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3128–3137, 2015.

Kwangyoun Kim, Felix Wu, Yifan Peng, Jing Pan, Prashant Sridhar, Kyu J. Han, and Shinji Watanabe. E-branchformer: Branchformer with enhanced merging for speech recognition. In *2022 IEEE Spoken Language Technology Workshop (SLT)*, pp. 84–91, 2023. doi: 10.1109/SLT54892.2023.10022656.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *ICLR (Poster)*, 2015. URL http://dblp.uni-trier.de/db/conf/iclr/iclr2015.html#KingmaB14.

Bingrui Li, Jianfei Chen, and Jun Zhu. Memory efficient optimizers with 4-bit states. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=nN8TnHB5nw.

Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. BLIP: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 12888–12900. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/li22n.html.

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 Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32100–32121. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/liu24bn.html.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Yang Luo, Xiaozhe Ren, Zangwei Zheng, Zhuo Jiang, Xin Jiang, and Yang You. Came: Confidence-guided adaptive memory efficient optimization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4442–4453, 2023.

Kai Lv, Hang Yan, Qipeng Guo, Haijun Lv, and Xipeng Qiu. Adalomo: Low-memory optimization with adaptive learning rate. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 12486–12502, 2024.

Ionut-Vlad Modoranu, Mher Safaryan, Grigory Malinovsky, Eldar Kurtic, Thomas Robert, Peter Richtárik, and Dan Alistarh. Microadam: Accurate adaptive optimization with low space overhead and provable convergence. *Advances in Neural Information Processing Systems*, 37:1–43, 2024.

Thien Hang Nguyen and Huy Nguyen. Lean and mean adaptive optimization via subset-norm and subspace-momentum with convergence guarantees. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 46116–46161. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/nguyen25f.html.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: An asr corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5206–5210, 2015. doi: 10.1109/ICASSP.2015.7178964.

Kwangryeol Park and Seulki Lee. Smmf: Square-matricized momentum factorization for memory-efficient optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 19848–19856, 2025.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *SC20: international conference for high performance computing, networking, storage and analysis*, pp. 1–16. IEEE, 2020.

Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In *International Conference on Learning Representations*, 2018.

Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4596–4604. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/shazeer18a.html.

Hanlin Tang, Shaoduo Gan, Ammar Ahmad Awan, Samyam Rajbhandari, Conglong Li, Xiangru Lian, Ji Liu, Ce Zhang, and Yuxiong He. 1-bit adam: Communication efficient large-scale training with adam's convergence speed. In *International Conference on Machine Learning*, pp. 10118–10129. PMLR, 2021.

Xuan Tang, Jichu Li, and Difan Zou. A convergence analysis of adaptive optimizers under floating-point quantization. In *International Conference on Learning Representations*, volume 2026, pp. 4069–4127, 2026.

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.

Kristi Topollai and Anna Choromanska. Understanding quantization of optimizer states in llm pre-training: Dynamics of state staleness and effectiveness of state resets. *arXiv preprint arXiv:2603.16731*, 2026.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288.

Xingyu Xie, Pan Zhou, Huan Li, Zhouchen Lin, and Shuicheng Yan. Adan: Adaptive nesterov momentum algorithm for faster optimizing deep models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):9508–9520, 2024. doi: 10.1109/TPAMI.2024.3423382.

Cong Xu, Wenbin Liang, Mo Yu, Anan Liu, Ke-Yue Zhang, Shunli Wang, Lizhuang Ma, Jianyong Wang, Jun Wang, and Wei Zhang. Pushing the limits of low-bit optimizers: A focus on ema dynamics. *arXiv preprint arXiv:2505.00347*, 2025.

Yuhui Xu, Lingxi Xie, Xiaotao Gu, Xin Chen, Heng Chang, Hengheng Zhang, Zhengsu Chen, Xiaopeng Zhang, and Qi Tian. Qa-lora: Quantization-aware low-rank adaptation of large language models. In *International Conference on Learning Representations*, volume 2024, pp. 52401–52418, 2024.

Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *Advances in neural information processing systems*, 35:27168–27183, 2022.

Longhui Yu, Weisen JIANG, Han Shi, Jincheng YU, Zhengying Liu, Yu Zhang, James Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 45040–45061, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/c400474e8a36d0812fdee52739288b12-Paper-Conference.pdf.

Yushun Zhang, Congliang Chen, Ziniu Li, Tian Ding, Chenwei Wu, Diederik Durk Kingma, Yinyu Ye, Zhi-Quan Luo, and Ruoyu Sun. Adam-mini: Use fewer learning rates to gain more. In *International Conference on Learning Representations*, volume 2025, pp. 28033–28063, 2025.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. GaLore: Memory-efficient LLM training by gradient low-rank projection. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 61121–61143. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/zhao24s.html.

Juntang Zhuang, Tommy Tang, Yifan Ding, Sekhar C Tatikonda, Nicha Dvornek, Xenophon Papademetris, and James Duncan. Adabelief optimizer: Adapting stepsizes by the belief in observed gradients. *Advances in neural information processing systems*, 33:18795–18806, 2020.

A SCALE NORMALIZED REPARAMETERIZATION OF ADAM

Definitions: Consider $\mathbf{S}_{t-1} = \sqrt{\hat{\mathbf{V}}_{t-1}}$ as our scale. Assuming $\mathbf{S}_{t-1} > 0$ for the moment, we define the scale-normalized versions of $\mathbf{G}_t$, $\hat{\mathbf{M}}_t$, and $\hat{\mathbf{V}}_t$ as:

$$\mathbf{Z}_t = \frac{\mathbf{G}_t}{\mathbf{S}_{t-1}} \qquad \bar{\mathbf{M}}_t = \frac{\hat{\mathbf{M}}_t}{\mathbf{S}_{t-1}} \qquad \bar{\mathbf{V}}_t = \frac{\hat{\mathbf{V}}_t}{\mathbf{S}_{t-1}^2} \tag{13}$$

Deriving recurrence for scale: Using our scale definition, unrolling $\hat{V}_t$ for one step, and simplifying, we obtain:

$$\begin{aligned}
\mathbf{S}_t &= \sqrt{\hat{\mathbf{V}}_t} && (14)\\
&= \sqrt{\frac{\beta_2}{(1-\beta_2^t)} \cdot \mathbf{V}_{t-1} + \frac{(1-\beta_2)}{(1-\beta_2^t)} \cdot \mathbf{G}_t^2} \quad \text{(using eqs. (1) and (2))} && (15)\\
&= \sqrt{\frac{\beta_2(1-\beta_2^{t-1})}{(1-\beta_2^t)} \cdot \hat{\mathbf{V}}_{t-1} + \frac{(1-\beta_2)}{(1-\beta_2^t)} \cdot \mathbf{G}_t^2} \quad \text{(using } \hat{\mathbf{V}}_{t-1} = \mathbf{V}_{t-1}/(1-\beta_2^{t-1})\text{)} && (16)\\
&= \sqrt{\frac{\beta_2(1-\beta_2^{t-1})}{(1-\beta_2^t)} \cdot \mathbf{S}_{t-1}^2 + \frac{(1-\beta_2)}{(1-\beta_2^t)} \cdot \mathbf{G}_t^2} \quad \text{(using } \mathbf{S}_{t-1} = \sqrt{\hat{\mathbf{V}}_{t-1}}\text{)} && (17)\\
&= \sqrt{\frac{(\beta_2-\beta_2^t)}{(1-\beta_2^t)} \cdot \mathbf{S}_{t-1}^2 + \frac{(1-\beta_2)}{(1-\beta_2^t)} \cdot \mathbf{G}_t^2} && (18)\\
&= \sqrt{(1-b_t) \cdot \mathbf{S}_{t-1}^2 + b_t \cdot \mathbf{G}_t^2} \quad \text{where} \quad b_t = \frac{(1-\beta_2)}{(1-\beta_2^t)} && (19)\\
&= \mathbf{S}_{t-1} \odot \sqrt{(1-b_t) + b_t \cdot \frac{\mathbf{G}_t^2}{\mathbf{S}_{t-1}^2}} && (20)\\
&= \mathbf{S}_{t-1} \odot \sqrt{(1-b_t) + b_t \cdot \mathbf{Z}_t^2} \quad \text{(using defs in eq. (13))} && (21)
\end{aligned}$$

where $\odot$ denotes element-wise multiplication. Therefore, $\mathbf{S}_t$ can be recursively computed as:

$$\mathbf{S}_t = \mathbf{S}_{t-1} \odot \sqrt{(1-b_t) + b_t \cdot \mathbf{Z}_t^2} \quad \text{where} \quad b_t = \frac{(1-\beta_2)}{(1-\beta_2^t)} \tag{22}$$

Deriving recurrence for effective update: Ignoring ϵ, Adam's effective update in eq. (3) becomes:

$$\mathbf{U}_t = \frac{\bar{\mathbf{M}}_t}{\sqrt{\bar{\mathbf{V}}_t}} \tag{23}$$

Note that multiplying and dividing eq. (23) by $\mathbf{S}_{t-1}$ directly recovers Adam's original update equation. Now, unrolling $\bar{\mathbf{M}}_t$ for one step and simplifying, we get:

$$\mathbf{U}_t = \frac{\bar{\mathbf{M}}_t}{\sqrt{\bar{\mathbf{V}}_t}} \tag{24}$$

$$= \frac{\frac{\beta_1}{(1-\beta_1^t)} \cdot \frac{\mathbf{M}_{t-1}}{\mathbf{S}_{t-1}} + \frac{(1-\beta_1)}{(1-\beta_1^t)} \cdot \frac{\mathbf{G}_t}{\mathbf{S}_{t-1}}}{\sqrt{\bar{\mathbf{V}}_t}} \quad \text{(using eqs. (1) and (2))} \tag{25}$$

$$= \frac{\frac{\beta_1(1-\beta_1^{t-1})}{(1-\beta_1^t)} \cdot \frac{\hat{\mathbf{M}}_{t-1}}{\mathbf{S}_{t-1}} + \frac{(1-\beta_1)}{(1-\beta_1^t)} \cdot \frac{\mathbf{G}_t}{\mathbf{S}_{t-1}}}{\sqrt{\bar{\mathbf{V}}_t}} \text{ (using } \hat{\mathbf{M}}_{t-1} = \mathbf{M}_{t-1}/(1-\beta_1^{t-1})) \tag{26}$$

$$= \frac{\frac{\beta_1(1-\beta_1^{t-1})}{(1-\beta_1^t)} \cdot \mathbf{U}_{t-1} + \frac{(1-\beta_1)}{(1-\beta_1^t)} \cdot \mathbf{Z}_t}{\sqrt{\bar{\mathbf{V}}_t}} \quad \text{(using defs in eqs. (13) and (23))} \tag{27}$$

$$= \frac{(1-a_t) \cdot \mathbf{U}_{t-1} + a_t \cdot \mathbf{Z}_t}{\sqrt{\bar{\mathbf{V}}_t}} \quad \text{where} \quad a_t = \frac{(1-\beta_1)}{(1-\beta_1^t)} \tag{28}$$

$$= \frac{(1-a_t) \cdot \mathbf{U}_{t-1} + a_t \cdot \mathbf{Z}_t}{\frac{\sqrt{\hat{\mathbf{V}}_t}}{\mathbf{S}_{t-1}}} \quad \text{(using defs in eq. (13))} \tag{29}$$

$$= \frac{(1-a_t) \cdot \mathbf{U}_{t-1} + a_t \cdot \mathbf{Z}_t}{\frac{\mathbf{S}_t}{\mathbf{S}_{t-1}}} \quad \text{(using } \mathbf{S}_t = \sqrt{\hat{\mathbf{V}}_t}) \tag{30}$$

$$= \frac{(1-a_t) \cdot \mathbf{U}_{t-1} + a_t \cdot \mathbf{Z}_t}{\sqrt{(1-b_t) + b_t \cdot \mathbf{Z}_t^2}} \quad \text{(using defs in eq. (22))} \tag{31}$$

Therefore, $\mathbf{U}_t$ can be recursively computed as:

$$\mathbf{U}_t = \frac{(1-a_t) \cdot \mathbf{U}_{t-1} + a_t \cdot \mathbf{Z}_t}{\sqrt{(1-b_t) + b_t \cdot \mathbf{Z}_t^2}} \quad \text{where} \quad a_t = \frac{(1-\beta_1)}{(1-\beta_1^t)} \tag{32}$$

Handling division-by-zero errors: Division-by-zero can originate in two places: when computing $\mathbf{Z}_t$ (since $\mathbf{S}_{t-1}$ could be zero), and in the denominator during the computation of $\mathbf{U}_t$. In both cases, similar to Adam, we introduce a small constant $\epsilon > 0$ for numerical stability. Therefore, the final equations become:

$$\mathbf{U}_t = \frac{(1-a_t) \cdot \mathbf{U}_{t-1} + a_t \cdot \mathbf{Z}_t}{\sqrt{(1-b_t) + b_t \cdot \mathbf{Z}_t^2} + \epsilon} \qquad \mathbf{S}_t = \mathbf{S}_{t-1} \odot \sqrt{(1-b_t) + b_t \cdot \mathbf{Z}_t^2}$$
$$\text{where} \quad \mathbf{Z}_t = \frac{\mathbf{G}_t}{\mathbf{S}_{t-1} + \epsilon}, \; a_t = \frac{(1-\beta_1)}{(1-\beta_1^t)} \text{and} \; \; b_t = \frac{(1-\beta_2)}{(1-\beta_2^t)} \tag{33}$$

B Reinterpretation of Adam's Moment Estimates

In Adam, the first- and second-order moments are computed as:

$$\mathbf{M}_t = \beta_1 \mathbf{M}_{t-1} + (1-\beta_1)\mathbf{G}_t \qquad \mathbf{V}_t = \beta_2 \mathbf{V}_{t-1} + (1-\beta_2)\mathbf{G}_t^2 \tag{34}$$

Unrolling these equations with $\mathbf{M}_0 = \mathbf{V}_0 = 0$ yields:

$$\mathbf{M}_t = (1-\beta_1)\sum_{k=0}^{t-1} \beta_1^k \mathbf{G}_{t-k} \qquad \mathbf{V}_t = (1-\beta_2)\sum_{k=0}^{t-1} \beta_2^k \mathbf{G}_{t-k}^2 \tag{35}$$

B.1 REWRITING THE FIRST-ORDER MOMENT

Consider $\mathbf{G}_{t-k} = \mathbf{G}_t + \delta_k$. Substituting this into eq. (35), we get:

$$\mathbf{M}_t = (1-\beta_1)\sum_{k=0}^{t-1}\beta_1^k(\mathbf{G}_t+\delta_k) \tag{36}$$

$$= \mathbf{G}_t\left((1-\beta_1)\sum_{k=0}^{t-1}\beta_1^k\right)+(1-\beta_1)\sum_{k=0}^{t-1}\beta_1^k\delta_k \tag{37}$$

$$= \mathbf{G}_t\left((1-\beta_1)\frac{(1-\beta_1^t)}{(1-\beta_1)}\right)+(1-\beta_1)\sum_{k=0}^{t-1}\beta_1^k\delta_k \quad \text{(sum of G.P. with } r=\beta_1<1) \tag{38}$$

$$= (1-\beta_1^t)\mathbf{G}_t+(1-\beta_1)\sum_{k=1}^{t-1}\beta_1^k\delta_k \quad \text{(since } \delta_k=0 \text{ when } k=0) \tag{39}$$

$$= (1-\beta_1^t)\mathbf{G}_t+\underbrace{(1-\beta_1)\sum_{k=1}^{t-1}\beta_1^k(\mathbf{G}_{t-k}-\mathbf{G}_t)}_{\mathbf{E}_t^m} \tag{40}$$

We can simplify $\mathbf{E}_t^m$ further by computing the quantity $\mathbf{E}_t^m - \beta_1\mathbf{E}_{t-1}^m$:

$$\mathbf{E}_t^m-\beta_1\mathbf{E}_{t-1}^m = (1-\beta_1)\sum_{k=1}^{t-1}\beta_1^k(\mathbf{G}_{t-k}-\mathbf{G}_t)-(1-\beta_1)\sum_{k=1}^{t-2}\beta_1^{k+1}(\mathbf{G}_{t-k-1}-\mathbf{G}_{t-1}) \tag{41}$$

$$= (1-\beta_1)\sum_{k=1}^{t-1}\beta_1^k(\mathbf{G}_{t-k}-\mathbf{G}_t)-(1-\beta_1)\sum_{j=2}^{t-1}\beta_1^j(\mathbf{G}_{t-j}-\mathbf{G}_{t-1}) \quad \text{(use } j=k+1) \tag{42}$$

$$= \beta_1(1-\beta_1)(\mathbf{G}_{t-1}-\mathbf{G}_t)+(1-\beta_1)\sum_{k=2}^{t-1}\beta_1^k[(\mathbf{G}_{t-k}-\mathbf{G}_t)-(\mathbf{G}_{t-k}-\mathbf{G}_{t-1})] \tag{43}$$

$$= \beta_1(1-\beta_1)(\mathbf{G}_{t-1}-\mathbf{G}_t)+(1-\beta_1)\sum_{k=2}^{t-1}\beta_1^k(\mathbf{G}_{t-1}-\mathbf{G}_t) \tag{44}$$

$$= (1-\beta_1)(\mathbf{G}_{t-1}-\mathbf{G}_t)\left(\beta_1+\sum_{k=2}^{t-1}\beta_1^k\right) \tag{45}$$

$$= (1-\beta_1)(\mathbf{G}_{t-1}-\mathbf{G}_t)\left(\sum_{k=1}^{t-1}\beta_1^k\right) \tag{46}$$

$$= (1-\beta_1)(\mathbf{G}_{t-1}-\mathbf{G}_t)\left(\frac{\beta_1(1-\beta_1^{t-1})}{(1-\beta_1)}\right) \tag{47}$$

$$= \beta_1(1-\beta_1^{t-1})(\mathbf{G}_{t-1}-\mathbf{G}_t) \tag{48}$$

Thus, $\mathbf{E}_t^m$ can be recursively computed as:

$$\mathbf{E}_t^m = \beta_1\mathbf{E}_{t-1}^m+\beta_1(1-\beta_1^{t-1})(\mathbf{G}_{t-1}-\mathbf{G}_t) \tag{49}$$

Unrolling this recursive relation with $\mathbf{E}_1^m = 0$, we get:

$$\mathbf{E}_t^m = \sum_{k=1}^{t-1}\beta_1^{k-1}\cdot\beta_1(1-\beta_1^{t-k})(\mathbf{G}_{t-k}-\mathbf{G}_{t-k+1}) \tag{50}$$

$$= \sum_{k=1}^{t-1}(\beta_1^k-\beta_1^t)(\mathbf{G}_{t-k}-\mathbf{G}_{t-k+1}) \tag{51}$$

$$= -\sum_{k=1}^{t-1}(\beta_1^k-\beta_1^t)(\mathbf{G}_{t-k+1}-\mathbf{G}_{t-k}) \tag{52}$$

Substituting this into eq. (40), we finally get:

$$\mathbf{M}_t = (1-\beta_1^t)\mathbf{G}_t - \sum_{k=1}^{t-1}(\beta_1^k - \beta_1^t)(\mathbf{G}_{t-k+1} - \mathbf{G}_{t-k}) \tag{53}$$

Consequently, the bias-corrected version looks like:

$$\hat{\mathbf{M}}_t = \frac{\mathbf{M}_t}{(1-\beta_1^t)} \tag{54}$$

$$= \mathbf{G}_t - \sum_{k=1}^{t-1}\frac{(\beta_1^k - \beta_1^t)}{(1-\beta_1^t)}(\mathbf{G}_{t-k+1} - \mathbf{G}_{t-k}) \tag{55}$$

$$= \mathbf{G}_t - \sum_{k=1}^{t-1}\omega_{k,t}(\beta_1)(\mathbf{G}_{t-k+1} - \mathbf{G}_{t-k}) \tag{56}$$

where $\omega_{k,t}(\beta_1) = \frac{(\beta_1^k - \beta_1^t)}{(1-\beta_1^t)}$. Since $\beta_1 < 1$, for a fixed t, the weighting coefficient $\omega_{k,t}(\beta_1) > 0$ and it decreases monotonically with k, thereby giving maximum weight to the most recent gradient changes.

B.2 Rewriting the second-order moment

Similarly, consider $\mathbf{G}_{t-k}^2 = \mathbf{G}_t^2 + \rho_k$. Substituting this into eq. (35), we get:

$$\mathbf{V}_t = (1-\beta_2)\sum_{k=0}^{t-1}\beta_2^k\left(\mathbf{G}_t^2 + \rho_k\right) \tag{57}$$

$$= \mathbf{G}_t^2\left((1-\beta_2)\sum_{k=0}^{t-1}\beta_2^k\right) + (1-\beta_2)\sum_{k=0}^{t-1}\beta_2^k\rho_k \tag{58}$$

$$= \mathbf{G}_t^2\left((1-\beta_2)\frac{(1-\beta_2^t)}{(1-\beta_2)}\right) + (1-\beta_2)\sum_{k=0}^{t-1}\beta_2^k\rho_k \quad \text{(sum of G.P. with } r=\beta_2<1\text{)} \tag{59}$$

$$= (1-\beta_2^t)\mathbf{G}_t^2 + (1-\beta_2)\sum_{k=1}^{t-1}\beta_2^k\rho_k \quad \text{(since } \rho_k = 0 \text{ when } k=0\text{)} \tag{60}$$

$$= (1-\beta_2^t)\mathbf{G}_t^2 + \underbrace{(1-\beta_2)\sum_{k=1}^{t-1}\beta_2^k\left(\mathbf{G}_{t-k}^2 - \mathbf{G}_t^2\right)}_{\mathbf{E}_t^v} \tag{61}$$

Just like before, we can simplify $\mathbf{E}_t^v$ further by computing the quantity $\mathbf{E}_t^v - \beta_2 \mathbf{E}_{t-1}^v$:

$$\mathbf{E}_t^v - \beta_2 \mathbf{E}_{t-1}^v = (1-\beta_2)\sum_{k=1}^{t-1}\beta_2^k\left(\mathbf{G}_{t-k}^2 - \mathbf{G}_t^2\right) - (1-\beta_2)\sum_{k=1}^{t-2}\beta_2^{k+1}\left(\mathbf{G}_{t-k-1}^2 - \mathbf{G}_{t-1}^2\right) \tag{62}$$

$$= (1-\beta_2)\sum_{k=1}^{t-1}\beta_2^k\left(\mathbf{G}_{t-k}^2 - \mathbf{G}_t^2\right) - (1-\beta_2)\sum_{j=2}^{t-1}\beta_2^j\left(\mathbf{G}_{t-j}^2 - \mathbf{G}_{t-1}^2\right)$$

$$\text{(use } j = k+1\text{)} \tag{63}$$

$$= \beta_2(1-\beta_2)\left(\mathbf{G}_{t-1}^2 - \mathbf{G}_t^2\right) + (1-\beta_2)\sum_{k=2}^{t-1}\beta_2^k\left[\left(\mathbf{G}_{t-k}^2 - \mathbf{G}_t^2\right) - \left(\mathbf{G}_{t-k}^2 - \mathbf{G}_{t-1}^2\right)\right] \tag{64}$$

$$= \beta_2(1-\beta_2)\left(\mathbf{G}_{t-1}^2 - \mathbf{G}_t^2\right) + (1-\beta_2)\sum_{k=2}^{t-1}\beta_2^k\left(\mathbf{G}_{t-1}^2 - \mathbf{G}_t^2\right) \tag{65}$$

$$= (1-\beta_2)\left(\mathbf{G}_{t-1}^2 - \mathbf{G}_t^2\right)\left(\beta_2 + \sum_{k=2}^{t-1}\beta_2^k\right) \tag{66}$$

$$= (1-\beta_2)\left(\mathbf{G}_{t-1}^2 - \mathbf{G}_t^2\right)\left(\sum_{k=1}^{t-1}\beta_2^k\right) \tag{67}$$

$$= (1-\beta_2)\left(\mathbf{G}_{t-1}^2 - \mathbf{G}_t^2\right)\left(\frac{\beta_2(1-\beta_2^{t-1})}{(1-\beta_2)}\right) \tag{68}$$

$$= \beta_2(1-\beta_2^{t-1})\left(\mathbf{G}_{t-1}^2 - \mathbf{G}_t^2\right) \tag{69}$$

This yields the recursive formulation for $\mathbf{E}_t^v$:

$$\mathbf{E}_t^v = \beta_2 \mathbf{E}_{t-1}^v + \beta_2(1-\beta_2^{t-1})\left(\mathbf{G}_{t-1}^2 - \mathbf{G}_t^2\right) \tag{70}$$

Unrolling this recursive relation with $\mathbf{E}_1^v = 0$, we get:

$$\mathbf{E}_t^v = \sum_{k=1}^{t-1}\beta_2^{k-1}\cdot\beta_2(1-\beta_2^{t-k})\left(\mathbf{G}_{t-k}^2 - \mathbf{G}_{t-k+1}^2\right) \tag{71}$$

$$= \sum_{k=1}^{t-1}(\beta_2^k - \beta_2^t)\left(\mathbf{G}_{t-k}^2 - \mathbf{G}_{t-k+1}^2\right) \tag{72}$$

$$= -\sum_{k=1}^{t-1}(\beta_2^k - \beta_2^t)\left(\mathbf{G}_{t-k+1}^2 - \mathbf{G}_{t-k}^2\right) \tag{73}$$

Substituting this into eq. (61), we get:

$$\mathbf{V}_t = (1-\beta_2^t)\mathbf{G}_t^2 - \sum_{k=1}^{t-1}(\beta_2^k - \beta_2^t)\left(\mathbf{G}_{t-k+1}^2 - \mathbf{G}_{t-k}^2\right) \tag{74}$$

Finally, the bias-corrected version looks like:

$$\hat{\mathbf{V}}_t = \frac{\mathbf{V}_t}{(1-\beta_2^t)} \tag{75}$$

$$= \mathbf{G}_t^2 - \sum_{k=1}^{t-1}\frac{(\beta_2^k - \beta_2^t)}{(1-\beta_2^t)}\left(\mathbf{G}_{t-k+1}^2 - \mathbf{G}_{t-k}^2\right) \tag{76}$$

$$= \mathbf{G}_t^2 - \sum_{k=1}^{t-1}\omega_{k,t}(\beta_2)\left(\mathbf{G}_{t-k+1}^2 - \mathbf{G}_{t-k}^2\right) \tag{77}$$

where $\omega_{k,t}(\beta_2) = \frac{(\beta_2^k - \beta_2^t)}{(1-\beta_2^t)}$. Since $\beta_2 < 1$, for a fixed t, $\omega_{k,t}(\beta_2) > 0$ and decreases monotonically with k, assigning maximum weight to the most recent changes in squared-gradient values.

B.3 Deriving the normalized second-moment drift

Recall the following definitions of $\hat{\mathbf{V}}_t$, the squared gradient drift $\mathbf{D}_t^v$, and its normalized counterpart $\mathbf{d}_t^v$:

$$\hat{\mathbf{V}}_t = \mathbf{S}_t^2, \qquad \mathbf{D}_t^v = \mathbf{G}_t^2 - \hat{\mathbf{V}}_t, \qquad \mathbf{d}_t^v = \frac{\mathbf{D}_t^v}{\mathbf{S}_t^2} \tag{78}$$

From the definition of $\mathbf{V}_t$ and $\hat{\mathbf{V}}_t = \frac{\mathbf{V}_t}{1-\beta_2^t}$, we can write:

$$\hat{\mathbf{V}}_t = \frac{\beta_2 - \beta_2^t}{1-\beta_2^t}\hat{\mathbf{V}}_{t-1} + \frac{1-\beta_2}{1-\beta_2^t}\mathbf{G}_t^2$$

Substituting $\hat{\mathbf{V}}_t = \mathbf{S}_t^2$, $\hat{\mathbf{V}}_{t-1} = \mathbf{S}_{t-1}^2$, and rearranging terms gives us:

$$(1-\beta_2)(\mathbf{G}_t^2 - \mathbf{S}_t^2) = (\beta_2 - \beta_2^t)(\mathbf{S}_t^2 - \mathbf{S}_{t-1}^2)$$

Given $\mathbf{D}_t^v = \mathbf{G}_t^2 - \hat{\mathbf{V}}_t$, we can write:

$$\mathbf{D}_t^v = \frac{\beta_2 - \beta_2^t}{1-\beta_2}(\mathbf{S}_t^2 - \mathbf{S}_{t-1}^2) \tag{79}$$

Thus, from the definition of $\mathbf{d}_t^v$ in Equation (78) and Equation (79), $\mathbf{d}_t^v$ can be further expanded and written as:

$$\begin{aligned} \mathbf{d}_t^v &= \frac{\beta_2 - \beta_2^t}{1-\beta_2}\frac{(\mathbf{S}_t^2 - \mathbf{S}_{t-1}^2)}{\mathbf{S}_t^2} \\ &= \frac{\beta_2 - \beta_2^t}{1-\beta_2}\left[1 - \left(\frac{\mathbf{S}_{t-1}}{\mathbf{S}_t}\right)^2\right] \end{aligned} \tag{80}$$

B.4 Detailed results on Commonsense Reasoning Task

Table 4: Performance comparison between READAM, full-precision AdamW (fp32) (Loshchilov & Hutter, 2019), 8-bit AdamW (Dettmers et al., 2022b), and various LPMM configurations (Li et al., 2023) on Commonsense Reasoning Task using DoRA with rank $r = 32$ on Llama3. DTQ and AF stand for Dynamic Tree Quantization and Adafactor, respectively.

Method	Arc-E (Acc.) ↑	Arc-C (Acc.) ↑	OBQA (Acc.) ↑	SIQA (Acc.) ↑
AdamW	$92.96_{\pm 0.87}$	$82.71_{\pm 0.65}$	$87.67_{\pm 1.07}$	$81.47_{\pm 0.72}$
8-bit AdamW	$92.61_{\pm 0.31}$	$83.16_{\pm 0.91}$	$87.67_{\pm 1.33}$	$\mathbf{82.14}_{\pm 0.66}$
4-bit LPMM (DTQ)	$91.25_{\pm 1.72}$	$81.17_{\pm 0.80}$	$87.80_{\pm 0.60}$	$80.59_{\pm 0.37}$
4-bit LPMM (AF)	$91.98_{\pm 0.32}$	$81.80_{\pm 1.08}$	$87.93_{\pm 0.53}$	$81.32_{\pm 1.02}$
5-bit LPMM (DTQ)	$92.58_{\pm 0.58}$	$82.42_{\pm 0.85}$	$88.07_{\pm 2.07}$	$80.91_{\pm 0.98}$
5-bit LPMM (AF)	$92.69_{\pm 0.20}$	$83.13_{\pm 0.91}$	$88.60_{\pm 1.00}$	$81.22_{\pm 0.15}$
4-bit READAM (ours)	$\mathbf{93.53}_{\pm 0.56}$	$\mathbf{84.13}_{\pm 0.43}$	$88.67_{\pm 0.33}$	$81.46_{\pm 0.45}$
5-bit READAM (ours)	$93.08_{\pm 0.53}$	$83.08_{\pm 0.40}$	$\mathbf{88.80}_{\pm 2.00}$	$81.63_{\pm 0.61}$