SAGE: Sign-Adaptive Gradient for Memory-Efficient LLM Optimization

Anonymous ACL submission

Abstract

The AdamW optimizer, while standard for LLM pretraining, is a critical memory bottleneck, consuming optimizer states equivalent to twice the model's size. Light-state optimizers like SinkGD attempt to solve this, but we identify the *embedding layer paradox*: these methods fail to handle the sparse, high-variance gradients of the embedding layer, forcing a hybrid design that reverts to AdamW and partially nullifies memory gains. We propose **SAGE** (Sign Adaptive GradiEnt), a novel optimizer that resolves this paradox by replacing AdamW in this hybrid structure. **SAGE** combines a Lion-style update direction with a new, memory-efficient $O(d)$ adaptive scale. This scale acts as a "safe damper," provably bounded by 1.0, which tames high-variance dimensions more effectively than existing methods. This superior stability allows **SAGE** to achieve better convergence. On Llama models up to 1.3B parameters, our **SAGE**-based hybrid achieves new state-of-the-art perplexity, outperforming all baselines, including SinkGD hybrid, while significantly reducing optimizer state memory.

1 Introduction

The training of Large Language Models (LLMs) (Brown et al., 2020; Touvron et al., 2023) is a resource-intensive endeavor, fundamentally constrained by the "memory wall." While significant strides have been made to reduce the memory footprint of model parameters—through techniques such as Low-Rank Adaptation (LoRA) (Hu et al., 2022) or low-precision quantization (Dettmers et al., 2022)—these approaches often act as compromises, trading off model capacity or training dynamics for memory efficiency. However, a distinct and equally critical memory bottleneck persists in the optimizer itself. The industry-standard AdamW optimizer (Loshchilov and Hutter, 2019), prized for its stability, demands two full-sized moment states for every parameter. For modern multi-billion parameter models, these $O(Vd)$ states can consume memory proportional to twice the model size, drastically limiting feasible batch sizes and model scaling.

To alleviate this, a lineage of light-state optimization research has emerged. Sign-based methods like Lion (Chen et al., 2023) demonstrated the viability of using a single moment state with sign-based updates. Methods like GaLore (Zhao et al., 2024) and APOLLO (Zhu et al., 2024) explored low-rank projections and block-wise updates to reduce state overhead. Most recently, SWAN (Ma et al., 2024) and SinkGD (Scetbon et al., 2025) proposed a stateless $O(1)$ normalization technique, representing the latest advancement in minimizing memory overhead while maintaining training stability.

Building upon these advances, we observed an intriguing phenomenon regarding the embedding layer. While recent light-state methods are highly effective for dense layers, we found that the sparse, high-variance gradients characteristic of the embedding layer often pose a unique stability challenge. This observation explains the design choice in recent works to retain a hybrid structure, falling back to the memory-intensive AdamW specifically for embeddings. Our work takes this observation as a starting point: rather than viewing this as a limitation of prior work, we see it as an opportunity to develop a specialized, memory-efficient solution that eliminates this final dependency on heavy optimizers (Section 2).

Contributions. Our core contribution is to solve this *embedding layer paradox*. We introduce **SAGE** (**S**ign **A**daptive **G**radi**E**nt), a novel, memory-efficient optimizer specifically designed to master the embedding layer's gradients. **SAGE** maintains only a single $O(Vd)$ moment state (like Lion) but replaces AdamW's $O(Vd)$ second-moment state with a novel, $O(d)$ dimension-wise

adaptive damper. This state tracks the mean absolute gradient (L_1 norm) and is theoretically bounded, providing a memory-efficient mechanism to stabilize high-variance gradients. This design results in a truly memory-efficient hybrid optimizer. We demonstrate that our **SAGE**-based hybrid, which allows for a more aggressive learning rate, outperforms AdamW, Lion, and the original SinkGD+Adam hybrid in both test perplexity and memory footprint across models up to 1.3B parameters (Section 4).

2 Background and Motivation

2.1 The Optimizer State Memory Bottleneck

The memory cost of an optimizer is dominated by its states. For a parameter θ, AdamW maintains a first moment $\mathbf{M}_t$ and a second moment $\mathbf{V}_t$. For an embedding layer of size $V \times d$, this requires two $O(Vd)$ states. As V often exceeds $100,000$, these states are the single largest memory cost outside of the model parameters themselves.

2.2 The Promise of Light-State Optimizers

Our work is built upon the foundations laid by several key paradigms in optimization. We view the following methods as the cornerstones of our approach, synthesizing their strengths to achieve a new level of efficiency:

- **AdamW** (Loshchilov and Hutter, 2019): The robust standard, providing the stability of second-moment normalization which we aim to approximate efficiently.
- **Lion** (Chen et al., 2023): The pioneer of the single-state, sign-based update, which demonstrates that memory can be halved without sacrificing convergence speed.
- **SinkGD** (Scetbon et al., 2025): The state-of-the-art framework for light-state training, which provides the effective hybrid architecture that we adopt and refine.

2.3 The Embedding Layer Paradox

While promising, we observe a curious escape hatch in the design of state-of-the-art light-state optimizers. The work by Scetbon et al. (2025) itself defaults to a hybrid structure, applying its stateless SinkGD to 2D weights but reverting to the 2-state AdamW for the embedding layer.

This raises a critical question: why does the light-state approach fail for embeddings? We hypothesize this is due to the unique properties of the embedding layer: its gradients are both **sparse** and **high-variance** due to the Zipfian distribution (Zipf, 1946) of token frequencies. Stateless methods, which apply uniform normalization, struggle to efficiently learn representations under these conditions.

To validate this hypothesis, we conduct a motivating experiment comparing two versions of SinkGD on a 270M parameter model:

1. **SinkGD-Pure:** The stateless normalizer is applied to **all** 2D parameters, including embeddings.
2. **SinkGD-Hybrid:** The hybrid design from Scetbon et al. (2025), using AdamW for the embedding layer.

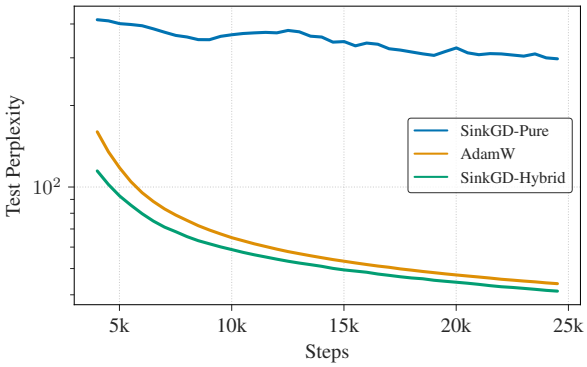


Figure 1: Test Perplexity for SinkGD-Pure vs. SinkGD-Hybrid. Even with tuned learning rates, the pure-stateless optimizer fails to learn effective embedding representations, resulting in significantly worse final perplexity compared to the hybrid design.

As shown in Figure 1, while SinkGD-Pure can be tuned to remain stable, it suffers from a catastrophic deficit in learning efficiency, failing to reach competitive perplexity. This confirms our hypothesis: the embedding layer requires specialized, adaptive optimization that naive light-state methods cannot provide.

This result provides strong evidence for our central hypothesis: the embedding layer is a critical point of instability that naive light-state optimizers cannot handle. Therefore, the key to a truly memory-efficient optimizer is to design a new, light-state algorithm that can successfully replace AdamW on the embedding layer.

3 Proposed Method: SAGE

3.1 A Hybrid, Efficient Optimizer

Our analysis in Section 2.3 establishes a clear design goal: an embedding optimizer that is both memory-light and stable against sparse, high-variance gradients.

The Lion optimizer (Chen et al., 2023) is a strong memory-light baseline, using only a single $O(Vd)$ momentum state $\mathbf{M}_t$. Its core update, $\hat{\mathbf{U}}_t = \text{sign}(\mathbf{M}_t)$, can be understood as having an implicit, static update scale of 1.0 in every dimension. While simple and fast, this static scale lacks any mechanism to control high-variance dimensions, which limits its maximum stabile learning rate and can be suboptimal.

We propose **SAGE** as a direct, adaptive generalization of Lion. SAGE replaces Lion's static 1.0 scale with a novel, $O(d)$ adaptive scale, $\mathbf{H}_t$. This adaptive scale is the core of our contribution. It is an element-wise relative damper derived from an $O(d)$ exponential moving average(EMA) of the mean absolute gradients (L_1 norm). As detailed in Algorithm 1, this scale compares the "loudness" of each dimension to the layer update's root mean sqaure(RMS) average. It is provably bounded ($||\mathbf{H}_t||_\infty \leq 1.0$), allowing it to selectively damp high-variance dimensions where the signal exceeds the layer average, while defaulting to Lion's 1.0 scale for quieter dimensions.

Our full optimizer follows the hybrid structure motivated in Section 2. While prior work (Scetbon et al., 2025) relied on AdamW for 1D parameters, we find that SAGE is equally effective for them. We therefore propose a configuration summarized in Table 1:

Model Component	Optimizer	State Size
Embedding Layer	**SAGE**	$O(Vd) + O(d)$
Biases/Norms (1D)	**SAGE**	$2 \times O(d)$
Dense Weights (2D)	SinkGD	$O(1)$

Table 1: Composition of the proposed **SAGE-Unified** optimizer. By using SAGE for embeddings instead of AdamW (which requires $2 \times O(Vd)$ states), we reduce the dominant memory cost by $\approx 50\%$ while maintaining stability.

3.2 The SAGE Algorithm

The SAGE update is detailed in Algorithm 1. While primarily designed for the embedding layer, SAGE generalizes naturally to 1D parameters (biases, layer norms) via simple branching logic (Equation 2), allowing it to act as a universal stateful optimizer.

Algorithm 1 Sign-Adaptive GradiEnt (SAGE)

Input: Learning rate schedule $(\eta_t)_{t\geq 1}$, decay rates $w, \beta_1, \beta_2, \epsilon$. **Initialize:** Params θ_0, $\mathbf{M}_0 \leftarrow \mathbf{0}$, $\mathbf{S}_0 \leftarrow \mathbf{0}$.

for $t = 1$ **to** T **do**
  $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
  $\theta_{t-1} \leftarrow \theta_{t-1} \cdot (1 - \eta_t w)$

  if θ *is* $2D$ *(Embedding)* **then**
    $(\mathbf{s}_t)_j \leftarrow \frac{1}{V}\sum_{i=1}^{V} |g_{t,ij}|$ for $j = 1...d$
  else
    $\mathbf{s}_t \leftarrow |g_t|$ {Element-wise Abs}
  end if
  $\mathbf{S}_t \leftarrow \beta_2 \mathbf{S}_{t-1} + (1 - \beta_2)\mathbf{s}_t$
  $\hat{\mathbf{S}}_t \leftarrow \mathbf{S}_t/(1 - \beta_2^t)$

  $\sigma_{\text{rms}} \leftarrow \sqrt{\frac{1}{d}\sum_{j=1}^{d}(\hat{\mathbf{S}}_t)_j^2}$

  $\gamma_{\text{rms}} \leftarrow \sqrt{\frac{1}{d}\sum_{j=1}^{d}(\mathbf{s}_t)_j^2}$

  for $j = 1$ **to** d **do**
    $(\mathbf{D}_t^{\text{ema}})_j \leftarrow \sigma_{\text{rms}}/((\hat{\mathbf{S}}_t)_j + \epsilon)$
    $(\mathbf{D}_t^{\text{inst}})_j \leftarrow \gamma_{\text{rms}}/((\mathbf{s}_t)_j + \epsilon)$
    $(\mathbf{H}_t)_j \leftarrow \min((\mathbf{D}_t^{\text{ema}})_j, (\mathbf{D}_t^{\text{inst}})_j, 1)$
  end for
  $\mathbf{C}_t \leftarrow \text{sign}(\beta_1 \mathbf{M}_{t-1} + (1 - \beta_1)g_t)$
  $\hat{\mathbf{U}}_t \leftarrow \mathbf{C}_t \odot \mathbf{H}_t$
  $\theta_t \leftarrow \theta_{t-1} - \eta_t \cdot \hat{\mathbf{U}}_t$
  $\mathbf{M}_t \leftarrow \beta_2 \mathbf{M}_{t-1} + (1 - \beta_2)g_t$
end for
return θ_T

Decoupled Weight Decay

We follow the AdamW (Loshchilov and Hutter, 2019) style of decoupled weight decay, which is applied directly to the parameters θ before any other operation:

$$\theta_{t-1} \leftarrow \theta_{t-1} \cdot (1 - \eta_t w) \tag{1}$$

Adaptive Scale Calculation

The core of SAGE's memory-efficiency is its $O(d)$ adaptive state, $\mathbf{S}_t$. For embedding layers (2D), we

compute $\mathbf{s}_t$ as the mean of absolute gradients across the vocabulary dimension ($O(d)$ reduction). For 1D parameters, we set $\mathbf{s}_t = |g_t|$, retaining per-element resolution.

$$\mathbf{s}_t \leftarrow \begin{cases} \frac{1}{V}\sum_{i=1}^{V}|g_{t,ij}| & \text{if 2D (Embedding)} \\ |g_t| & \text{if 1D (Bias/Norm)} \end{cases} \quad (2)$$

This vector $\mathbf{s}_t$ is used to update the exponential moving average (EMA) $\mathbf{S}_t$. To counteract bias towards zero during the initial training steps, we apply a standard bias correction (Kingma and Ba, 2015) to obtain the final state $\hat{\mathbf{S}}_t$

The adaptive scale $\mathbf{H}_t$ is calculated using a *Relative RMS* strategy. We first calculate the Root Mean Square (σ_{rms}) of the state vector $\hat{\mathbf{S}}_t$ to establish a reference "loudness" for the layer's update. We then define the relative adaptive scale $\mathbf{H}_t$ as the ratio of this reference to the j-th dimension's value.

This design of $\mathbf{H}_t$, formalized in Algorithm1, ensures scale invariance. For "quiet" dimensions (where $(\hat{\mathbf{S}}_t)_j < \sigma_{\text{rms}}$), the ratio exceeds 1, and we clip to 1, defaulting to Lion's behavior. For "loud" dimensions (where $(\hat{\mathbf{S}}_t)_j > \sigma_{\text{rms}}$), the ratio is < 1, safely damping the high-variance signal.

Finally, to prevent catastrophic instability from sudden spikes that the EMA may lag behind, we enforce an additional instantaneous stability constraint derived from the current batch statistics (analogous to element-wise Adaptive Gradient Clipping (Brock et al., 2021)); full details are provided in Appendix A.5.

Direction and Final Update

The final parameter update is the element-wise product of the direction $\mathbf{C}_t$ and our new adaptive scale $\mathbf{H}_t$, scaled by the learning rate η_t. While the update direction $\mathbf{C}_t$ and momentum state $\mathbf{M}_t$ are updated identically to Lion (Chen et al., 2023).

$$\hat{\mathbf{U}}_t \leftarrow \mathbf{C}_t \odot \mathbf{H}_t, \quad \theta_t \leftarrow \theta_{t-1} - \eta_t \cdot \hat{\mathbf{U}}_t \quad (3)$$

3.3 Theoretical Analysis

We now provide a theoretical analysis to ground our approach. The convergence of our hybrid optimizer relies on its components; we refer to the established proofs for AdamW (Loshchilov and Hutter, 2019) and Sinkhorn GD (Scetbon et al., 2025). Our analysis focuses on proving the convergence of our novel **SAGE** update.

Following standard practice in non-convex optimization (Nesterov, 2004; Robbins and Monro, 1951; Bottou et al., 2018), we make the following assumptions:

- **L-smoothness:** The loss function f is L-smooth, ensuring the gradient does not change arbitrarily fast.
- **Bounded Variance:** The stochastic gradients g_t have bounded variance, $\mathbb{E}[||g_t - \nabla f(\theta_t)||^2] \leq \sigma^2$.
- **Bounded Gradients:** The stochastic gradients are bounded, $||g_t||_\infty \leq G$.

Justification for the Adaptive State. Our use of the L_1-based state $\mathbf{S}_t$ as a proxy for per-dimension gradient scale is grounded in the established principles of deep network training. Modern architectures, through techniques like Xavier initialization (Glorot and Bengio, 2010) and Layer Normalization (Ba et al., 2016), are designed to maintain activation and gradient means near zero. Under this zero-mean assumption, the mean absolute value ($\mathbf{S}_t \approx \mathbb{E}[|g_t|]$) becomes a robust estimator of the gradient's standard deviation (scale), similar to the $\sqrt{\mathbb{E}[g_t^2]}$ estimator used by Adam.

Lemma 3.1 (Bounded $\mathbf{S}_t$)**.** *Given the Bounded Gradients assumption, the stochastic gradient g_t is bounded. Therefore, the per-dimension L1-mean $\mathbf{s}_t$ (an average of bounded values) is also bounded. As $\mathbf{S}_t$ is an EMA of this bounded signal, $\mathbf{S}_t$ is also bounded, i.e., $||\mathbf{S}_t||_\infty \leq S_{max}$.*

Lemma 3.2 (Bounded Adaptive Scale $\mathbf{H}_t$)**.** *The SAGE adaptive scale $\mathbf{H}_t$ is, by design, element-wise bounded, with an L_∞ norm $||\mathbf{H}_t||_\infty \leq 1.0$.*

Proof. Let $\hat{\mathbf{S}}_t$ be the bias-corrected EMA state. From Lemma 3.1, $\hat{\mathbf{S}}_t$ is bounded and non-negative. We calculate the layer-wise RMS, $\sigma_{\text{rms}} = \sqrt{\frac{1}{d}\sum(\hat{\mathbf{S}}_t)_j^2}$, which serves as a strictly positive scalar reference (assuming non-zero gradients). The raw damper $\mathbf{D}_t$ is calculated as $(\mathbf{D}_t)_j = \sigma_{\text{rms}}/((\hat{\mathbf{S}}_t)_j + \epsilon)$.

The final adaptive scale $\mathbf{H}_t$ is computed element-wise as:

$$(\mathbf{H}_t)_j = \min((\mathbf{D}_t)_j, 1) \quad (4)$$

By the definition of the $\min$ operation, for any dimension j, the value $(\mathbf{H}_t)_j$ is non-negative and guaranteed to be $(\mathbf{H}_t)_j \leq 1.0$. Therefore, the L_∞

norm (the maximum absolute element) of the adaptive scale is also bounded:

$$||\mathbf{H}_t||_\infty = \max_j |(\mathbf{H}_t)_j| \leq 1 \tag{5}$$

This formally proves that our adaptive scale $\mathbf{H}_t$ provides a provably safe, non-amplifying update. □

Theorem 3.3 (Convergence of SAGE)**.** *SAGE converges to a stationary point (i.e., $\lim_{T\to\infty} \mathbb{E}[||\nabla f(\theta_t)||^2] = 0$) under the standard assumptions and a suitable learning rate decay schedule.*

Proof Sketch. The **SAGE** update is $\hat{\mathbf{U}}_t = \mathbf{C}_t \odot \mathbf{H}_t$.

1. The update direction $\mathbf{C}_t = \text{sign}(\dots)$ is bounded, $||\mathbf{C}_t||_\infty \leq 1$.
2. From Lemma 3.2, the adaptive scale $\mathbf{H}_t$ is also bounded, $||\mathbf{H}_t||_\infty \leq 1$.
3. Therefore, the full update step $\hat{\mathbf{U}}_t$ is element-wise bounded, $||\hat{\mathbf{U}}_t||_\infty = ||\mathbf{C}_t \odot \mathbf{H}_t||_\infty \leq 1$.

SAGE applies an update that is provably no larger in magnitude than that of Lion. As **SAGE** is a strictly safer (i.e., more cautious) sign-based method, its convergence is guaranteed by the established frameworks for sign-based optimizers (Chen et al., 2023). □

3.4 Relationship to Lion

SAGE is a direct, adaptive generalization of the Lion(Chen et al., 2023) optimizer. The core Lion update can be expressed as:

$$\hat{\mathbf{U}}_t^{\text{Lion}} = \mathbf{C}_t \odot \mathbf{1} \tag{6}$$

where $\mathbf{C}_t = \text{sign}(\beta_1 \mathbf{M}_{t-1} + (1-\beta_1) g_t)$ and the update magnitude is a static scale of $\mathbf{1}$. The **SAGE** update simply replaces this static scale with our dynamic, adaptive scale $\mathbf{H}_t$:

$$\hat{\mathbf{U}}_t^{\text{SAGE}} = \mathbf{C}_t \odot \mathbf{H}_t \tag{7}$$

Therefore, Lion is a special case of **SAGE** where $\mathbf{H}_t$ is fixed to $\mathbf{1}$ for all dimensions and all steps t.

As our theoretical analysis in Lemma 3.2 proves, $||\mathbf{H}_t||_\infty \leq 1.0$. This means **SAGE** is a strictly safer generalization, as its per-dimension update magnitude is provably less than or equal to Lion's. Our experiments confirm that this bounded, adaptive damping is the key to providing the stability needed to outperform Lion by using a higher learning rate.

4 Experiments

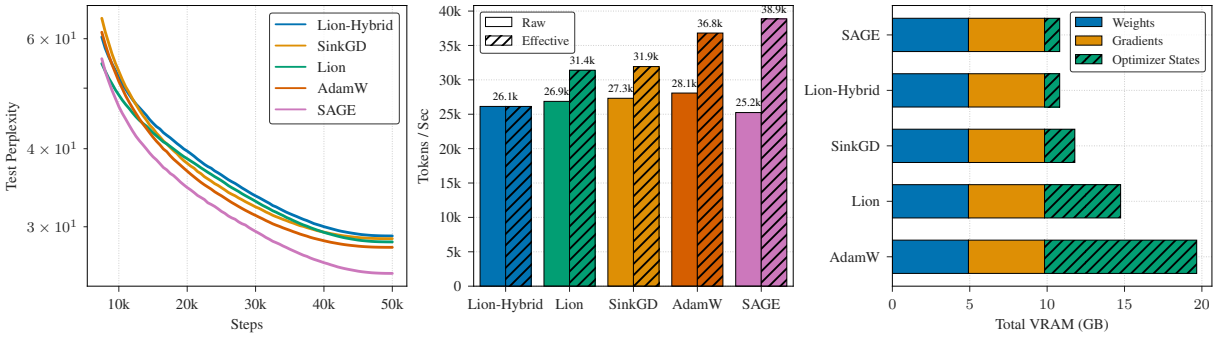
We conduct a series of experiments to validate SAGE and answer three key questions: 1) Does our hybrid SAGE optimizer outperform state-of-the-art baselines in final perplexity and convergence speed? 2) Is SAGE's adaptive "safe-damping" mechanism the key to its stability and performance? 3) Does SAGE deliver on its promise of significantly reducing memory consumption compared to AdamW used in SinkGD?

4.1 Experimental Setup

Training Details All models were implemented in PyTorch (Paszke et al., 2019) using the Transformers library (Wolf et al., 2019) and trained from scratch on a single NVIDIA H200 GPU. We used a global batch size of 130k tokens (sequence length 512) for all runs. All runs use a cosine learning rate schedule with 10% warmup. For baseline optimizers, we adopt the learning rates *favored* by their original work. For our proposed SAGE and other hybrid configurations not covered in prior work, we conduct a grid search to find the reasonable learning rate. This approach ensures a fair comparison where each optimizer operates in a reasonably tuned regime, while acknowledging that further fine-grained tuning could potentially yield marginal improvements for all methods. Other hyperparameters were kept constant ($\beta_1 = 0.9, \beta_2 = 0.99, \text{wd} = 0.01$ for all optimizers).

Models and Data We pretrain a series of LLaMA-architecture (Touvron et al., 2023) models from scratch, with parameter counts of 270M, 0.6B, and 1.3B. We train all models on a 6.6B token subset of The Pile (Gao et al., 2020), using **bfloat16** precision for efficiency. We choose The Pile over simpler datasets like C4 (Raffel et al., 2019) as recent work suggests its greater diversity provides a more rigorous benchmark for evaluating model capabilities (Bandari et al., 2024). As for the subset of The Pile, we used the data selection framework based on importance resampling introduced by Xie et al. (2023). We used an 8:2 train/test split ratio.

Optimizers and Baselines We compare following list of optimizer configurations. Our primary method is the **SAGE**, which uses SAGE for the embedding layer and follow SinkGD for all other weights. For brevity, we hereinafter refer to the hybrid configurations simply as SAGE and SinkGD,



(a) Perplexity (b) Training Throughput (c) Memory Usage

Figure 2: **Convergence and efficiency on the 1.3B model.** (a) Test perplexity - lower the better; SAGE achieves the lowest final perplexity and faster convergence. (b) Throughput - higher the better; *Effective* throughput normalizes processing speed by the steps required to match the lowest (Lion-Hybrid) performance. (c) Peak memory usage breakdown by component.

Group	Method	270M		0.6B		1.3B	
		PPL	Mem (GB)	PPL	Mem (GB)	PPL	Mem (GB)
Baselines	AdamW	37.35	2.1	31.74	4.4	27.81	9.8
	APOLLO	52.83	1.2	44.84	2.1	39.54	3.6
	Lion	30.24	1.0	**26.58**	2.2	28.37	4.9
	SinkGD(Hybrid)	34.30	0.9	32.85	1.5	28.71	1.9
Ablations	SinkGD-Pure	192.7	0.0	117.8	0.0	98.84	0.0
	SAGE-Pure	116.0	1.0	179.0	2.2	216.0	4.9
	Lion-Hybrid	32.10	**0.5**	28.73	**0.7**	28.40	**0.9**
Our Methods	**SAGE(Unified)**	**29.95**	**0.5**	26.71	**0.7**	**24.33**	**0.9**

Table 2: Main results comparing final test perplexity (PPL) and peak optimizer memory usage (Mem) across three model scales. SAGE consistently achieves the best perplexity while maintaining a low memory footprint, outperforming both pure and hybrid baselines. Lower PPL is better.

unless the distinction with the pure ablation is necessary. We compare against:

- **AdamW**: The standard 2-state (M, V) baseline from Loshchilov and Hutter (2019).
- **Lion**: The sign-based baseline from Chen et al. (2023) with 1-state (M) applied to all parameters.
- **APOLLO**: The baseline from Zhu et al. (2024), which uses an auxiliary low-rank optimizer state based on pure random projection.
- **SinkGD-Pure**: An ablation applying SinkGD from Scetbon et al. (2025) to all weights, including the embeddings.
- **SinkGD**: The baseline from Scetbon et al. (2025), where SinkGD is applied to all 2D weights except for the embeddings and for all other weights AdamW is used.
- **Lion-Hybrid**: A strong baseline with the same setup as SinkGD, but using Lion for embeddings.
- **SAGE-Pure**: An ablation where SAGE is applied to all parameters, including embeddings and 1D weights, to validate our hybrid design.
- **SAGE-Unified**: Our proposed method, using SinkGD for all 2D dense weights except for the embeddings and SAGE is applied to all other parameters. We also evaluated a variant using AdamW for 1D parameters as did Scetbon et al. (2025), but found no practical difference in performance, so we report results for the simpler unified configuration.

To validate our design, we conduct ablation studies with **pure** and other **hybrid** configurations, confirming that the SAGE structure is the optimal com-

bination. All 1D parameters (biases, normalization) of the hybrid optimizers use AdamW as did in Scetbon et al. (2025). All experiments are run with 3 different random seeds, and we report the average of the final test perplexity. Further details on data preprocessing and hyperparameter tuning are provided in Appendix A.

Unit-Norm SinkGD for Dense Layers. We employ a modified **Unit-Norm** version of Sinkhorn Gradient Descent (SinkGD) for dense 2D layers. Let $W \in \mathbb{R}^{m \times n}$ denote a weight matrix where we assume $m \leq n$ without loss of generality. The original SinkGD (Scetbon et al., 2025) explicitly scales the normalized gradient matrix by $\sqrt{nm}$ to match the expected Frobenius norm of Adam updates ($\approx \sqrt{nm}$).

However, our SAGE optimizer employs a damping mechanism that reduces the effective update magnitude for stability. Pairing SAGE with the original variance-preserving SinkGD creates a massive magnitude mismatch: the dense layers receive updates scaled by $\sqrt{nm}$ while the embedding layer receives damped updates ≤ 1.0.

To resolve this, we remove the $\sqrt{nm}$ scaling factor from SinkGD, targeting a unit row norm (1.0). This aligns the update magnitude of the dense layers with the "safe-damped" regime of SAGE. We find this magnitude alignment is critical for stability at scale, allowing us to control the effective learning speed via a single scalar hyperparameter α (typically $\alpha \approx 10$) rather than implicit dimensions-dependent scaling.

4.2 Main Results

Performance at Scale As shown in Table 2, our SAGE optimizer achieves the lowest test perplexity across all 3 model scales. We observe that while the Lion-Hybrid is a strong baseline, our SAGE consistently outperforms it. This confirms our hypothesis that Lion's static 1.0 scale is suboptimal, and our adaptive, "safe-damping" scale provides a significant performance gain. Furthermore, Figure 2 visualizes these results, showing that the performance gap between SAGE and its baselines widens as the model size increases. This suggests that SAGE's superior stability and adaptivity are even more critical for larger, more complex models. The performance advantage of SAGE becomes increasingly pronounced at larger scales. For the 1.3B model, it achieves a perplexity of 24.33, surpassing the next-best baseline (AdamW at 27.81) by a substantial margin. This widening gap is particularly noteworthy because as models grow, the embedding layer's relative parameter share decreases (from 48% to 20% in our experiments). We hypothesize this trend arises because larger models are deeper and have higher-dimensional embeddings, making the gradient dynamics more complex. In this regime, SAGE's fine-grained, adaptive control over embedding updates becomes even more critical for overall model performance.

4.3 Memory Footprint

As shown in Table 2, our SAGE optimizer offers significant memory savings compared to the SinkGD baseline, which uses the memory-intensive AdamW for its embedding layer. By replacing AdamW's $O(Vd)$ second-moment state with SAGE's negligible $O(d)$ state, our hybrid matches the low memory footprint of Lion-Hybrid. This reduction in optimizer state translates to a significant 10-45% decrease in total peak memory, at the model size of 1.3B, enabling researchers to train larger models or use larger batch sizes on the same hardware. A detailed breakdown across all model scales can be found in Appendix A.2.

4.4 Analysis and Ablation Studies

Learning Rate Tuning and Stability We investigate the optimal learning rate for SAGE through a targeted grid search, motivated by its relationship to Lion. Chen et al. (2023) recommends a learning rate of 1×10^{-4} for language modeling. This relationship (discussed in 3.4) allows us to directly adapt the learning rates from Lion via a *lazy-tuning* approach (tuned without extensive search), for SAGE. This can be thought of as a variation of LR grafting (Agarwal et al., 2022). We therefore conducted a grid search over the range $[1\times10^{-4}, 2\times10^{-4}, 5\times10^{-4}, 1\times10^{-3}, 2\times10^{-3}]$. While Lion-Hybrid's performance peaks at 1×10^{-4} and collapses shortly after, SAGE remains stable and achieves its best performance at a much higher learning rate of 1×10^{-3}. This demonstrates that SAGE's "safe-damping" mechanism successfully tames gradient variance, unlocking a more potent learning rate. Additionally, SAGE's adaptive nature also renders it robust to batch size variations, unlike stateless methods which behave like SGD and often require strict adherence to linear scaling rules when gradient noise levels change (Goyal et al., 2017; Krizhevsky, 2014).

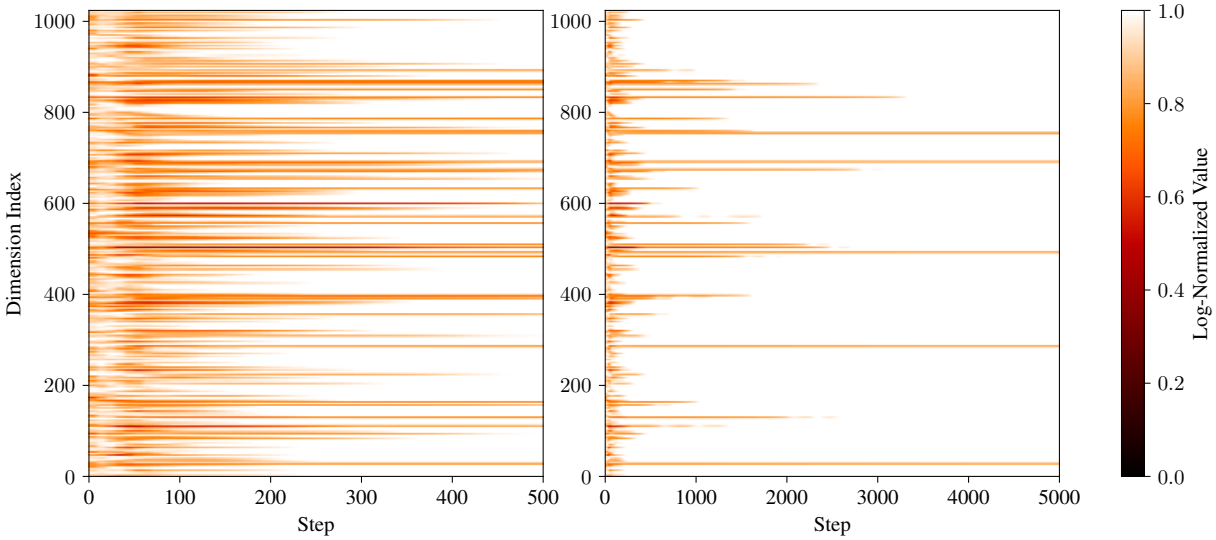


Figure 3: Visualization of the SAGE adaptive scale $\mathbf{H}_t$ (for $d = 1024$ dimensions) over the first 500 (left) and 5k (right) training steps. The raw $\mathbf{H}_t \in (0, 1.0]$ values are log-transformed and normalized. The **white background** (value 1.0) corresponds to the majority of "quiet" dimensions where the gradient variance is below the layer average, causing SAGE to clip the scale to 1.0 (Lion's default). The **colored stripes** (low values $\ll 1.0$) correspond to specific "loud" high-variance dimensions where SAGE actively applies strong damping. This confirms SAGE acts as a sparse damper, intervening only on specific unstable dimensions while letting the majority update at full scale.

Internal State Visualization Finally, in Figure 3, we provide direct evidence of SAGE's mechanism. We visualize the normalized adaptive scale $\mathbf{H}_t$ over training time. The visualization reveals a distinct "sparse damping" policy. The vast majority of the heatmap is white (value 1.0), identifying the low-variance dimensions where SAGE clips the scale to 1.0, defaulting to Lion's behavior. Conversely, distinct colored stripes appear horizontally, **representing specific high-variance dimensions where SAGE actively applies strong damping** ($\mathbf{H}_t \ll 1.0$). Crucially, the heatmap captures the temporal evolution of this mechanism. In the early training steps, Figure 3(left), we observe dense, widespread damping activity as the optimizer calibrates to the initial gradient noise. Over the long term, Figure3(right), the system stabilizes: the damping becomes more sparse and localized to a consistent subset of structurally high-variance dimensions, allowing the majority of the model to update at full magnitude. This proves SAGE is not just reacting to random spikes, but applying a consistent, fine-grained policy that evolves from global stabilization to surgical intervention.

This low-dimensional structure is consistent with findings that neural network optimization occurs in a low-dimensional subspace (Gur-Ari et al., 2018), and that adaptation signals often have a low-rank structure (Aghajanyan et al., 2021). Our analysis shows that although $\mathbf{H}_t$ lives in a d-dimensional space, its evolution is governed by a few smooth, coherent principal directions. This indicates that SAGE maintains a compact representation of the update geometry, enabling stable optimization without relying on traditional per-parameter moment estimates. Further PCA analysis of $\mathbf{H}_t$ can be found in A.3.

5 Conclusion

In this work, we identified the *embedding layer paradox* as an unaddressed bottleneck for light-state optimizers. We proposed SAGE, a novel optimizer that generalizes Lion with a memory-efficient $O(d)$ adaptive scale. By designing this scale as a "safe damper", SAGE effectively tames the high-variance embedding gradients that can limit the performance of the optimizer. Our experiments on models up to 1.3B parameters show that SAGE achieves state-of-the-art perplexity and convergence speed. We demonstrated SAGE's superior stability allows for more aggressive learning rates than Lion, achieving top-tier performance while matching the low memory footprint of Lion-based hybrids. By resolving the *embedding layer paradox*, SAGE delivers optimal performance without the high memory overhead of traditional methods, offering a more efficient path for researches involving LLM training.

Limitations

While SAGE demonstrates strong performance, our study has several limitations. First, our experiments were conducted on models up to 1.3B parameters. Although we observe a clear trend of a widening performance gap as model size increases, these models are still relatively small in the modern LLM landscape, and further validation on larger scales (e.g., 7B+) is necessary.

Second, the duration of our pre-training experiments was constrained by computational budgets. Since the *Effective Throughput* metric is a function of the final perplexity reached by the baseline, the reported efficiency ratios are tied to this specific training horizon. In practice, performance gaps between optimization algorithms often widen as training extends on larger datasets; thus, our reported efficiency gains may represent conservative estimates compared to full-scale, multi-trillion token pre-training.

Finally, our analysis was confined to language modeling on The Pile dataset. The effectiveness of SAGE on other modalities (e.g., vision) or fine-tuning tasks remains an open question.

References

Naman Agarwal, Rohan Anil, Elad Hazan, Tomer Koren, and Cyril Zhang. 2022. Learning rate grafting: Transferability of optimizer tuning.

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. 2021. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 7319–7328. Association for Computational Linguistics.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *Preprint*, arXiv:1607.06450.

Abhinav Bandari, Lu Yin, Cheng-Yu Hsieh, Ajay Jaiswal, Tianlong Chen, Li Shen, Ranjay Krishna, and Shiwei Liu. 2024. Is C4 dataset optimal for pruning? an investigation of calibration data for LLM pruning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 18089–18099. Association for Computational Linguistics.

Léon Bottou, Frank E. Curtis, and Jorge Nocedal. 2018. Optimization methods for large-scale machine learning. *SIAM Rev.*, 60(2):223–311.

Andy Brock, Soham De, Samuel L. Smith, and Karen Simonyan. 2021. High-performance large-scale image recognition without normalization. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 1059–1071. PMLR.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler angpt3d Jeffrey Wu, Clemens Winter, Christopher Hesse, and 11 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V. Le. 2023. Symbolic discovery of optimization algorithms. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Gpt3.int8(): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. The Pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, volume 9 of *JMLR Proceedings*, pages 249–256. JMLR.org.

Priya Goyal, Piotr Dollár, Ross B. Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. 2017. Accurate, large minibatch SGD: training imagenet in 1 hour. *CoRR*, abs/1706.02677.

Guy Gur-Ari, Daniel A. Roberts, and Ethan Dyer. 2018. Gradient descent happens in a tiny subspace. *CoRR*, abs/1812.04754.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and

Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Alex Krizhevsky. 2014. One weird trick for parallelizing convolutional neural networks. *CoRR*, abs/1404.5997.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Chao Ma, Wenbo Gong, Meyer Scetbon, and Edward Meeds. 2024. SWAN: SGD with normalization and whitening enables stateless LLM training. *CoRR*, abs/2412.13148.

Yurii E. Nesterov. 2004. *Introductory Lectures on Convex Optimization - A Basic Course*, volume 87 of *Applied Optimization*. Springer.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, and 2 others. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv e-prints*.

Herbert Robbins and Sutton Monro. 1951. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407.

Meyer Scetbon, Chao Ma, Wenbo Gong, and Edward Meeds. 2025. Gradient multi-normalization for stateless and scalable LLM training. *CoRR*, abs/2502.06742.

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

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771.

Sang Michael Xie, Shibani Santurkar, Tengyu Ma, and Percy Liang. 2023. Data selection for language models via importance resampling. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Guodong Zhang, Chaoqi Wang, Bowen Xu, and Roger B. Grosse. 2019. Three mechanisms of weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. 2024. Galore: Memory-efficient LLM training by gradient low-rank projection. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Hanqing Zhu, Zhenyu Zhang, Wenyan Cong, Xi Liu, Sem Park, Vikas Chandra, Bo Long, David Z. Pan, Zhangyang Wang, and Jinwon Lee. 2024. APOLLO: sgd-like memory, adamw-level performance. *CoRR*, abs/2412.05270.

George Kingsley Zipf. 1946. The psychology of language. In *Encyclopedia of psychology*, pages 332–341. Philosophical Library.

A Appendix

A.1 Model Configurations

The table 3 provides a full configurations of each models used in the experiments, for readers to reproduce the results.

Configuration	270M	0.6B	1.3B
Hidden Size	1024	1536	2048
Intermediate Size	2816	4224	5632
Max Position Embeddings	8192	8192	8192
Num Attention Heads	16	24	32
Num Hidden Layers	13	16	24
Num Key-Value Heads	2	3	4
Vocab Size	128256	128256	128256

Table 3: Model configurations for the different scales used in the experiments.

A.2 Full Memory Footprint Comparison

Table 4 provides a comprehensive breakdown of memory consumption across all tested model scales.

	270M (Emb. Params: 48%)		0.6B (Emb. Params: 33%)		1.3B (Emb. Params: 20%)	
Optimizer	**Opt. (GB)**	**Total (GB)**	**Opt. (GB)**	**Total (GB)**	**Opt. (GB)**	**Total (GB)**
AdamW	2.045	4.091	4.421	8.843	9.833	19.67
APOLLO	1.208	3.254	2.102	6.524	3.648	13.48
Lion	1.023	3.067	2.211	6.631	4.916	14.75
SinkGD(Hybrid)	0.979	3.023	1.468	5.890	1.958	11.79
SinkGD-Pure	*0.0*	*2.046*	*0.0*	*4.422*	*0.0*	*9.832*
SAGE-Pure	1.023	3.069	2.212	6.634	4.92	14.75
Lion-Hybrid	**0.489**	**2.535**	**0.734**	**5.156**	**0.979**	**10.81**
SAGE(Unified)	**0.489**	**2.535**	**0.734**	**5.156**	**0.979**	**10.81**

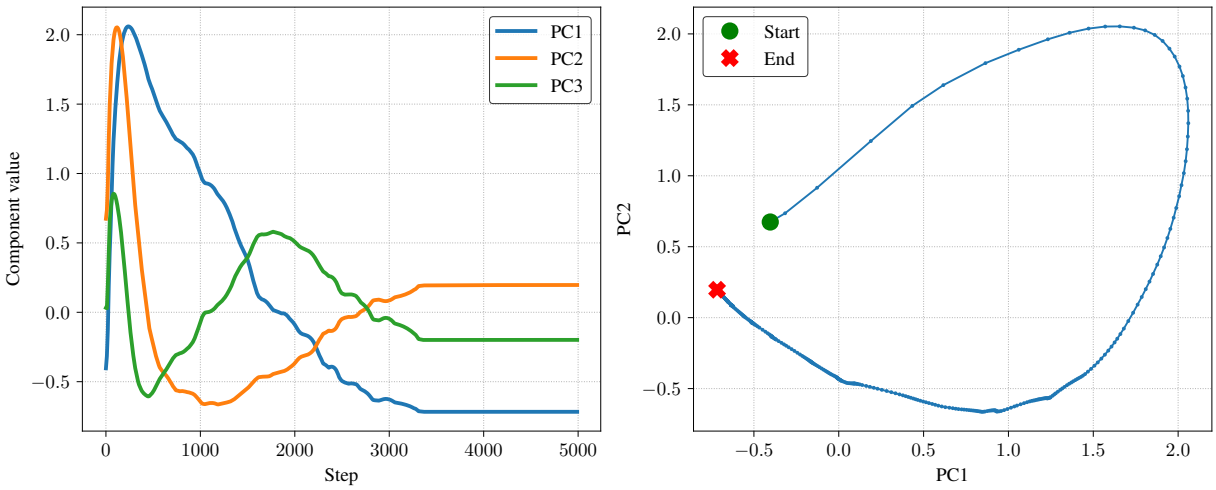
Table 4: Peak memory footprint comparison across all model scales. "Opt." refers to optimizer state memory, while "Total" is the sum of model weights, gradients and the optimizer states. SAGE is comparable to the low total memory of a pure stateless optimizer *SinkGD-Pure*, offering a substantial reduction compared to the AdamW-based hybrid, which translates to significant savings in total memory.

A.3 Internal State Dynamics Analysis

Internal State Dynamics (PCA) To further understand how SAGE regulates update magnitudes, we perform Principal Component Analysis (PCA) on the sequence of the adaptive scale vectors, $\mathbf{H}_t$. The trajectories of the top principal components (PCs) reveal a distinct two-phase adaptation process.

As shown in Figure 4 (a), both PC1 and PC2 exhibit a sharp, significant rise in magnitude during the initial training steps ($t < 50$), followed by a gradual decay and stabilization. This indicates that SAGE performs **rapid initial calibration**, aggressively adjusting the damping factors as it first encounters the embedding layer's high-variance gradients. Following this discovery phase, the components settle into stable, non-zero values, confirming that the optimizer finds and maintains a consistent damping policy for specific dimensions rather than oscillating erratically.

The trajectory in the PC1-PC2 plane (Figure 4 (b)) visualizes this dynamic as a clear "excursion and return" path. The system quickly moves to a high-activity region to handle initial training instability and



(a) PCA Component Trajectories

(b) Trajectory in PC1-PC2 Space

Figure 4: PCA of the adaptive scale vector $\mathbf{H}_t$. (a) Trajectories of the top 3 principal components. PC1 and PC2 show a sharp, distinct spike in activity during the early training phase (initial 1k steps), reflecting rapid adaptation to the embedding layer's geometry. This is followed by a gradual stabilization phase where components settle into steady states. (b) The trajectory in PC1-PC2 space reveals a clear "hook" pattern: a rapid initial excursion away from the origin (adaptation) followed by a smooth return path that converges toward a stable operating point (stabilization).

then smoothly converges toward a fixed point. This confirms that SAGE's adaptive mechanism is both responsive to early-stage noise and stable during long-term training.

A.4 Learning Rate Tuning

We performed a comprehensive comparison runs of learning rates for SAGE on the 0.6B model, which resulted in $2e^{-3}$ as the best out of the range.

Furthermore, we observed a distinct advantage in stability regarding batch size variations. Stateless optimizers like SinkGD, lacking a history-based moment to normalize variance, behave similarly to standard SGD. Consequently, they are highly sensitive to the noise introduced by smaller batch sizes. To maintain convergence stability when the batch size is reduced, they typically require a proportional reduction in learning rate, following the Linear Scaling Rule (Goyal et al., 2017; Krizhevsky, 2014). In contrast, SAGE's adaptive damper $\mathbf{H}_t$ naturally adjusts to the changing gradient variance associated with different batch sizes. This behavior aligns with the properties of adaptive optimizers, which often follow a gentler square-root scaling or remain robust across a wider range of settings (Zhang et al., 2019). Empirically, we found that SAGE could maintain stability using the same learning rate even when the effective batch size was reduced to fewer than 10, whereas SinkGD required significant re-tuning.

A.5 Instantaneous Stability Constraint

While the EMA-based state $\mathbf{S}_t$ effectively handles long-term variance, it introduces a lag that can be vulnerable to sudden, extreme gradient spikes. To address this, we implement an *Instant Damper* based on the current batch's gradient $\mathbf{g}_t$.

We calculate the instantaneous relative scale $\mathbf{H}_t^{\text{inst}}$ using the same *Relative RMS* logic but applied to $|\mathbf{g}_t|$ instead of $\mathbf{S}_t$:

$$(\mathbf{H}_t^{\text{inst}})_j = \frac{\sqrt{\text{mean}(|\mathbf{g}_t|^2)}}{|(\mathbf{g}_t)_j| + \epsilon} \tag{8}$$

The final adaptive scale applied to the update is the conservative minimum of the EMA-based scale and this instantaneous scale:

$$\mathbf{H}_t \leftarrow \min(\mathbf{H}_t^{\text{EMA}}, \mathbf{H}_t^{\text{inst}}, 1.0) \tag{9}$$

This acts as a fast-acting *circuit breaker* that clips updates immediately if the current gradient direction is an outlier relative to the layer's current activation pattern.

A.6 Throughput Analysis and Metrics

To comprehensively evaluate training efficiency, we report performance on a single H200 GPU using two distinct metrics in Figure2(b):

Raw Throughput. This measures the standard computational speed of the optimizer, expressed in tokens processed per second (tokens/sec). It reflects the computational overhead of the optimizer's update step (e.g., SAGE vs. AdamW) independent of convergence quality.

Effective Throughput. Raw throughput does not account for algorithmic efficiency (i.e., how much the model learns per step). Following the methodology of Scetbon et al. (2025), we compute *Effective Throughput* to quantify wall-clock convergence speed.

We define the effective throughput $\mathcal{T}_{\text{eff}}$ of an optimizer $\mathcal{O}$ as:

$$\mathcal{T}_{\text{eff}}(\mathcal{O}) = \frac{N_{\text{base}}}{T_{\mathcal{O}}} \tag{10}$$

where N_{base} is the total number of training tokens required by the baseline optimizer Lion-Hybrid, which exhibited the slowest convergence in our 1.3b scale experiments, to reach its final test perplexity, and $T_{\mathcal{O}}$ is the wall-clock time in seconds required by optimizer $\mathcal{O}$ to reach that same perplexity. This metric effectively penalizes optimizers that process tokens quickly but converge slowly, while rewarding optimizers like SAGE that maximize learning efficiency per second.

Sensitivity to Experimental Setup. It is important to note that the absolute magnitudes of these throughput ratios may vary across different experimental environments. Raw throughput is sensitive to implementation details (e.g., kernel optimizations), while effective throughput depends heavily on the dataset size and total training budget. Specifically, the target perplexity defined by the baseline changes as training extends; typically, superior optimizers widen the performance gap on larger datasets or longer training horizons. Therefore, while computational constraints limited the scale of this study, we hypothesize that the reported efficiency gains are conservative estimates. We expect the relative performance ordering—with SAGE outperforming the baselines—to remain consistent or become more pronounced in larger-scale training regimes.

B Use of AI Assistants

We acknowledge the use of AI coding assistants to refine the Python scripts used for generating the plots in this paper. We emphasize that all data points presented in the figures were loaded directly from raw experimental logs, and the AI tools were strictly limited to formatting and visualization tasks.