Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Sign-agreement gating: a Lion variant

Abhishek Jaiswal

abhishek.2024ds03@mnnit.ac.in

Motilal Nehru National Institute of Technology

Rajesh Tripathi

Motilal Nehru National Institute of Technology

Research Article

Keywords: Lion optimizer, sign-based optimization, implicit regularization, vision transformer training, image classification

Posted Date: June 16th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-9756664/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Sign-agreement gating: a Lion variant

Abhishek Jaiswal[1*] and Rajesh Tripathi[1]

[1*]Department of Computer Science and Engineering, Motilal Nehru National Institute of Technology Allahabad, Prayagraj, Uttar Pradesh, India.

*Corresponding author(s). E-mail(s): abhishek30511@gmail.com;
Contributing authors: rajeshtcsed@mnnit.ac.in;

Abstract

The Lion optimizer replaces Adam's adaptive second-moment estimate with a sign-based update derived from an interpolated first-moment buffer, matching or exceeding AdamW on many vision and language benchmarks at roughly half the optimizer-state memory cost. Lion's update applies a step of fixed magnitude to every parameter, regardless of whether successive gradients agree in sign at that parameter or oscillate. We propose SAGE-Lion (Sign-Agreement Gated Evolved Lion), a Lion variant that maintains an exponential moving average of per-parameter sign agreement between consecutive gradients and uses this estimate as a multiplicative confidence weight on the sign-based update direction. The modification adds two scalar buffers per parameter and a small number of elementwise operations per step. We evaluate SAGE-Lion across five settings spanning two architecture families (ResNet-18/34, ViT-Tiny, ViT-B/16) and four image classification datasets (CIFAR-10, CIFAR-100, TinyImageNet, ImageNet-100). A learning-rate sensitivity hypothesis test shows that SAGE-Lion's optimal learning rate is approximately three times Lion's, and the two optimizers track each other under this rescaling in well-conditioned settings. On heavily overfitting ResNet+CIFAR-100 training, SAGE-Lion outperforms Lion by 1.6–2.1 percentage points in test accuracy with a 5–6 percentage point smaller train–test generalization gap, a signature of implicit regularization. On Vision Transformer training, SAGE-Lion matches Lion's accuracy while reducing wall-clock overhead from 18.6% on ResNet-18 to 0.9% on ViT-B/16, making it a practical drop-in substitute at production scale.

Keywords: Lion optimizer, sign-based optimization, implicit regularization, vision transformer training, image classification

1 Introduction

Adam [1] and its decoupled-weight-decay variant AdamW [2] remain the default optimizers for training deep neural networks in vision and language. Their per-parameter second-moment estimate provides a useful adaptive step size but doubles the optimizer-state memory compared to plain stochastic gradient descent, which becomes a non-trivial cost when training large models on memory-constrained hardware. The Lion optimizer [3], discovered through symbolic program search over optimizer space, removes the second-moment buffer entirely: it tracks only a single momentum estimate and uses the sign of an interpolated momentum–gradient mixture as the update direction. Lion has been reported to match or exceed AdamW on a range of image classification, contrastive pretraining, and language modeling tasks, with particularly strong gains on Vision Transformer (ViT) architectures [4, 5].

A consequence of Lion's design is that every parameter receives a step of magnitude equal to the learning rate, regardless of how consistent the gradient information at that parameter has been. When consecutive minibatch gradients agree in sign at a parameter, there is strong evidence that the direction is meaningful and a full-magnitude step is warranted. When they disagree—which happens routinely near flat regions of the loss surface and in the late stages of training—Lion applies the same step size, potentially adding stochastic noise to parameters that should be moving more slowly. Methods that adapt step size based on gradient sign agreement are not new: RProp [6] dates back three decades, and the broader idea of damping updates when gradient information is inconsistent appears in compressed and distributed training [7] and in adaptive variants of Adam such as AdaBelief [8]. What has not been studied is whether a small, lightweight sign-agreement signal can be folded into Lion's update without giving up its memory and wall-clock advantages.

This paper studies that question. We introduce **SAGE-Lion** (*Sign-Agreement Gated Evolved Lion*), a Lion variant that maintains an exponential moving average of the per-parameter agreement between consecutive gradient signs and uses this estimate as a multiplicative confidence weight on the sign-based update direction, as defined in Eq. (1) (formalized in Section 3):

$$\theta_t = (1 - \eta\lambda)\,\theta_{t-1} - \eta\,\kappa_t \odot \operatorname{sign}(c_t), \tag{1}$$

where c_t is the momentum–gradient interpolation used by Lion, $\kappa_t \in [\phi, 1]$ is the per-parameter confidence multiplier derived from running sign agreement, ϕ is a confidence floor that prevents stalling, η is the learning rate, λ is the weight decay, and $\odot$ denotes the elementwise product. The modification adds two scalar buffers per parameter (the previous gradient sign and the running sign agreement) and a small number of elementwise operations per step. It preserves Lion's single-moment structure and most of its memory advantage over AdamW.

A natural concern with any optimizer modification is that the observed behavior may be explained more parsimoniously by an effective rescaling of the learning rate. Because the confidence multiplier κ_t lies in $[\phi, 1]$ with a mean well below one in practice, SAGE-Lion's effective per-step magnitude is smaller than Lion's at the same nominal learning rate. A reviewer is right to ask whether SAGE-Lion at learning rate

η behaves identically to Lion at η/k for some constant k. We address this directly with a learning-rate sensitivity sweep on CIFAR-10 (Section 6), which shows that the optimal learning rate for SAGE-Lion is approximately $3\times$ that of Lion, and that the two optimizers track each other closely under this rescaling across most of the grid, with a small but consistent advantage for SAGE-Lion at the high-learning-rate end.

The remaining experimental question is whether sign-agreement gating produces measurable effects beyond a learning-rate rescaling on real training tasks. We evaluate this across four settings spanning two architecture families and four datasets: ResNet-18 on CIFAR-10 [9, 10], ResNet-18 and ResNet-34 on CIFAR-100, ViT-Tiny on TinyImageNet [11], and ViT-B/16 on ImageNet-100 [12, 13]. The picture that emerges is mixed in an interpretable way. On settings where the training task is easy enough that all optimizers converge to high training accuracy with small generalization gap (CIFAR-10), SAGE-Lion is statistically indistinguishable from Lion. On ResNet+CIFAR-100, where the model heavily overfits the training set, SAGE-Lion outperforms Lion by 1.6–2.1 percentage points in test accuracy (paired t-test, $p < 0.05$), with a 5–6pp smaller generalization gap and approximately 15% lower test loss—a signature consistent with implicit regularization. On ViT architectures, the two Lion-family optimizers are again statistically indistinguishable on accuracy, but both substantially outperform AdamW; on ViT-B/16 trained from scratch on ImageNet-100, the gap to AdamW exceeds 24 percentage points at epoch 60 (Section 9). The wall-clock overhead of the gating mechanism is approximately 16–19% on ResNet-scale models and falls below 2% on ViT-B/16, where the optimizer step cost is dominated by the forward and backward pass through an 86M-parameter model.

The contributions of this work are:

- A simple sign-agreement gating mechanism for Lion-style optimizers, requiring two scalar buffers per parameter and no additional second-moment computation (Section 3).
- A learning-rate sensitivity hypothesis test establishing that SAGE-Lion behaves approximately like Lion at one-third the learning rate, while remaining more robust to learning-rate values above the optimum (Section 6).
- Empirical evidence that the gating produces an implicit regularization effect on overfitting-heavy ResNet+CIFAR-100 training, yielding statistically significant accuracy gains over Lion and AdamW (Section 7).
- A warmup-plus-cosine schedule ablation on ResNet-18/CIFAR-100 (Section 7.7) showing that the regularization signature is substantially attenuated under cosine decay while SAGE-Lion retains a +7.3pp advantage over Lion through improved schedule stability.
- A 3×3 hyperparameter sensitivity sweep over $\beta_s \in \{0.7, 0.8, 0.9\}$ and $\phi \in \{0.05, 0.10, 0.20\}$ on ResNet-18/CIFAR-100 (Section 10) confirming that SAGE-Lion's performance is insensitive to both parameters across the ranges tested, with a total accuracy range of 0.45pp across all nine configurations.
- A scaling analysis on ViT-Tiny/TinyImageNet and ViT-B/16/ImageNet-100 showing that SAGE-Lion is accuracy-equivalent to Lion on Vision Transformers while the relative wall-clock overhead decreases with model size, falling below 2% on ViT-B/16 (Sections 8, 9).

The framing of this paper deserves an honest note. Sign-agreement gating is not a universal accuracy improvement over Lion. The mechanism activates when overfitting is the bottleneck and is otherwise neutral. We argue that this regime-dependent behavior is itself the contribution: an inexpensive variant that helps where Lion overfits, costs almost nothing where Lion already works well, and never measurably hurts in our experiments. Section 11 returns to when practitioners should reach for the gating and when standard Lion suffices.

2 Related Work

2.1 Adaptive optimizers in deep learning

The dominant family of optimizers for deep network training combines momentum with per-parameter step-size adaptation. Adam [1] maintains exponential moving averages of both the gradient and its square to produce element-wise rescaled updates, and was for several years the de facto choice in vision and language modeling. The decoupled weight-decay variant AdamW [2] corrected an implicit bias in Adam's coupling of L_2 regularization and adaptive step sizes, and is now the standard baseline for transformer training recipes including DeiT [5] and the original Vision Transformer [4]. Both optimizers carry two state buffers per parameter, which approximately triples the optimizer footprint relative to plain SGD with momentum. For models in the hundreds of millions to billions of parameters, this footprint is non-trivial: optimizer state is the dominant contributor to peak memory in large-scale training and motivates work on memory-efficient alternatives [14, 15].

2.2 Sign-based and Lion-family optimizers

A separate line of work uses only the sign of the gradient as the update direction. signSGD and its momentum variants [7] were originally proposed to reduce communication cost in distributed training: transmitting one bit per parameter instead of a full float preserves the direction of descent while compressing the message by $32\times$. Theoretical analysis [16] showed that sign-based methods can be understood as preconditioned SGD with a gradient-magnitude-dependent preconditioner, drawing a connection to Adam's adaptive step sizes.

Lion [3] departs from the distributed-training motivation: it uses the sign update because, empirically, it produces strong optimization with a single momentum buffer. The Lion update is

$$\theta_t = (1 - \eta\lambda)\,\theta_{t-1} - \eta\,\mathrm{sign}(\beta_1 m_{t-1} + (1-\beta_1)g_t)\,, \tag{2}$$

with the momentum buffer updated as $m_t = \beta_2 m_{t-1} + (1-\beta_2)g_t$, where g_t is the stochastic gradient, η is the learning rate, λ is weight decay, and (β_1, β_2) are momentum coefficients (Eq. (2)). The interpolation inside the sign operator gives Lion access to slightly more recent gradient information than the running momentum alone. Subsequent work [17] has explored alternative Lion variants by changing the interpolation, adding noise, or modifying the schedule. SAGE-Lion fits into this line: it modifies the

update by gating the sign rather than replacing it, which preserves the symbolic form Lion was discovered with while adding a per-parameter quality signal.

2.3 Step-size adaptation based on gradient agreement

The idea that consecutive gradient signs carry information about the quality of a descent direction is old. RProp [6] increases or decreases a per-parameter step size multiplicatively depending on whether successive gradients agree, and was widely used in the early neural-network literature. Modern adaptive methods include AdaBelief [8], which adapts step size based on the deviation of the current gradient from the running prediction, and AdaHessian [18], which uses curvature information to adapt step magnitudes. These methods all share a common goal of damping the update when gradient information is unreliable, but differ in what signal they use to measure reliability.

The closest conceptual relative of SAGE-Lion is RProp itself, with two important differences. First, RProp operates on full-precision per-parameter step sizes that grow and shrink multiplicatively over the course of training, while SAGE-Lion operates on a bounded confidence multiplier $\kappa_t \in [\phi, 1]$ that modulates a fixed sign-based direction. Second, RProp was designed for full-batch training and is known to be unstable in mini-batch settings [19]; the exponential moving average over sign agreement in SAGE-Lion is specifically intended to smooth out minibatch noise, similar in spirit to how Adam's second-moment estimate smooths squared gradients.

2.4 Vision Transformer training and optimizer choice

Vision Transformers [4] are known to be more sensitive to optimizer choice than convolutional networks. The original ViT paper used AdamW with substantial warmup and weight decay. The DeiT training recipe [5] demonstrated that with proper augmentation (RandAugment [20], Mixup [21], CutMix [22]), label smoothing [23], and stochastic depth, ViTs can be trained from scratch on ImageNet-1K with AdamW to competitive accuracy. The Lion paper [3] reported substantial gains over AdamW on ViT architectures, and our results on ViT-B/16 trained on ImageNet-100 (Section 9) corroborate this with a gap exceeding 23 percentage points in test top-1 accuracy at the 30-epoch mark.

The question we investigate experimentally is whether adding sign-agreement gating to Lion changes its behavior on Vision Transformers in a measurable way. Our finding is that it does not: on ViT-Tiny/TinyImageNet and ViT-B/16/ImageNet-100, SAGE-Lion and Lion are statistically indistinguishable on test accuracy. The gating produces measurable effects only when overfitting dominates, which in our experiments occurs primarily on ResNet+CIFAR-100.

2.5 Implicit regularization in optimization

When SAGE-Lion outperforms Lion in our experiments, it does so with a training-accuracy decrease and a test-accuracy increase—a pattern more consistent with implicit regularization than with finding a better optimum. Implicit regularization from optimizer design has been studied extensively in the SGD literature [24, 25],

where smaller learning rates and larger batch sizes have been shown to produce different generalization behavior even on the same loss surface. In the adaptive-optimizer setting, Lookahead optimizers [26] and stochastic weight averaging [27] provide explicit forms of regularization on top of an inner optimizer. SAGE-Lion's regularization effect is qualitatively different: it emerges from a per-parameter damping signal rather than from outer-loop averaging, and it activates only where gradient signs disagree—which is precisely where the loss surface is locally flat or noisy.

3 The SAGE-Lion Algorithm

3.1 Lion as the starting point

Let $\theta_t \in \mathbb{R}^d$ denote the model parameters at step t, $g_t = \nabla_\theta \mathcal{L}(\theta_t; \mathcal{B}_t)$ the stochastic gradient of the loss on minibatch $\mathcal{B}_t$, η the learning rate, λ the weight decay coefficient, and $(\beta_1, \beta_2) \in (0,1)^2$ the momentum coefficients. The Lion optimizer [3] maintains a single first-moment buffer $m_t \in \mathbb{R}^d$ and updates the parameters by

$$c_t = \beta_1 \, m_{t-1} + (1 - \beta_1) \, g_t, \tag{3}$$

$$\theta_t = (1 - \eta\lambda) \, \theta_{t-1} - \eta \, \mathrm{sign}(c_t), \tag{4}$$

$$m_t = \beta_2 \, m_{t-1} + (1 - \beta_2) \, g_t. \tag{5}$$

Eq. (3) produces a short-horizon interpolation between the running momentum and the current gradient, and the update direction is the elementwise sign of this interpolation (Eq. (4)). The momentum buffer itself is then updated with a longer-horizon coefficient β_2 (Eq. (5)). Two properties of this update are central to what follows. First, the step magnitude is exactly η for every parameter at every iteration, modulated only by weight decay. Second, the only persistent state is the single momentum buffer m_t, so the optimizer footprint is half that of Adam-family methods.

3.2 Sign-agreement gating

The motivation for SAGE-Lion is that a sign-based update treats two very different situations identically: the case where consecutive minibatch gradients consistently point in the same direction at a parameter, and the case where they alternate. The first carries strong evidence about the descent direction; the second is dominated by stochastic noise. A useful per-parameter quality signal is therefore the running agreement between consecutive gradient signs.

Let $s_t = \mathrm{sign}(g_t) \in \{-1, 0, +1\}^d$ denote the current gradient sign, and define the instantaneous sign-agreement indicator

$$a_t = s_t \odot s_{t-1}, \tag{6}$$

where $\odot$ is the Hadamard product. Entries of a_t in Eq. (6) take the value $+1$ when the current and previous gradients agree in sign at that coordinate, -1 when they disagree, and 0 when either gradient component is exactly zero. The quantity a_t is too

Algorithm 1 SAGE-Lion

Require: initial parameters θ_0, learning rate η, momentum coefficients β_1, β_2, sign-agreement coefficient β_s, weight decay λ, confidence floor ϕ

Initialize $m_0 \leftarrow 0,\ s_0 \leftarrow 0,\ \bar{a}_0 \leftarrow 0$
for $t = 1, 2, \ldots$ **do**
  Sample minibatch $\mathcal{B}_t$ and compute $g_t \leftarrow \nabla_\theta \mathcal{L}(\theta_{t-1}; \mathcal{B}_t)$
  $c_t \leftarrow \beta_1\, m_{t-1} + (1 - \beta_1)\, g_t$
  $m_t \leftarrow \beta_2\, m_{t-1} + (1 - \beta_2)\, g_t$
  if $t > 1$ **then**
    $a_t \leftarrow \mathrm{sign}(g_t) \odot s_{t-1}$
    $\bar{a}_t \leftarrow \beta_s\, \bar{a}_{t-1} + (1 - \beta_s)\, a_t$
    $\kappa_t \leftarrow \mathrm{clip}\big(\frac{1}{2}(\bar{a}_t + 1), \phi, 1\big)$
  else
    $\kappa_t \leftarrow \mathbf{1}$ ▷ No agreement signal at the first step
  end if
  $s_t \leftarrow \mathrm{sign}(g_t)$
  $\theta_t \leftarrow (1 - \eta\lambda)\, \theta_{t-1} - \eta\, \kappa_t \odot \mathrm{sign}(c_t)$
end for

noisy to use directly; SAGE-Lion smooths it with an exponential moving average

$$\bar{a}_t = \beta_s\, \bar{a}_{t-1} + (1 - \beta_s)\, a_t, \quad \bar{a}_0 = 0, \tag{7}$$

where $\beta_s \in (0, 1)$ is a smoothing coefficient. Eq. (7) gives $\bar{a}_t \in [-1, 1]^d$: a value near $+1$ indicates that the parameter's gradient sign has been stable over recent iterations, and a value near -1 indicates frequent oscillation. We default to $\beta_s = 0.9$, which corresponds to an effective averaging window of approximately $1/(1 - \beta_s) = 10$ steps.

We convert the sign agreement into a confidence multiplier in $[\phi, 1]$ via an affine rescaling followed by a clip,

$$\kappa_t = \mathrm{clip}\big(\tfrac{1}{2}\big(\bar{a}_t + 1\big),\ \phi,\ 1\big)\,, \tag{8}$$

where $\phi \in (0, 1)$ is a confidence floor. The mapping in Eq. (8) sends $\bar{a}_t = +1$ to $\kappa_t = 1$, $\bar{a}_t = 0$ to $\kappa_t = 0.5$, and $\bar{a}_t = -1$ to $\kappa_t = 0$; the clip with floor ϕ prevents the multiplier from collapsing to zero on parameters that have experienced sustained oscillation. We use $\phi = 0.1$ as the default, which keeps SAGE-Lion's worst-case per-parameter step at one-tenth the nominal learning rate.

The SAGE-Lion update replaces Eq. (4) with

$$\theta_t = (1 - \eta\lambda)\, \theta_{t-1} - \eta\, \kappa_t \odot \mathrm{sign}(c_t). \tag{9}$$

All other steps—the computation of c_t in Eq. (3), the momentum update in Eq. (5), the weight-decay application—are unchanged. Equations (6), (7), (8), and (9) together define the SAGE-Lion algorithm, which we summarize as pseudocode in Algorithm 1.

3.3 Memory and compute cost

SAGE-Lion adds two persistent buffers per parameter: the previous gradient sign $s_t \in \{-1, 0, +1\}^d$ and the running sign agreement $\bar{a}_t \in [-1, 1]^d$. In a straightforward implementation both are stored at the same precision as the model parameters, increasing optimizer state from Lion's d scalars (the momentum buffer) to $3d$ scalars. This still falls below AdamW's $2d$ scalars only if the first moment is replaced by reduced-precision storage, which is not how Lion is typically deployed; in practice SAGE-Lion's optimizer footprint is $3\times$ Lion's and $1.5\times$ AdamW's. Two implementation refinements that we have not pursued in this work could reduce this overhead. First, s_t is a sign tensor and can be packed at two bits per parameter, reducing its storage by $16\times$ at single precision. Second, since $\bar{a}_t$ enters only through Eq. (8), lower precision (e.g. FP16 or INT8 with a fixed scale) is likely sufficient. We leave these optimizations to future work and report all results with full-precision buffers.

The additional compute per step is small. Beyond Lion's operations, SAGE-Lion performs one elementwise multiplication (Eq. (6)), one EMA update (Eq. (7)), one affine rescaling with a clip (Eq. (8)), and one elementwise multiplication in the final update (Eq. (9)). On ResNet-scale models we measure a 16–19% per-epoch wall-clock overhead relative to AdamW (Section 5). On ViT-B/16 the overhead falls to approximately 2% (Section 9) because the forward and backward pass through an 86M-parameter transformer dominates the step cost; the optimizer update itself is a small fraction of the wall-clock time.

3.4 The learning-rate question

A practical concern is that SAGE-Lion's effective step magnitude is strictly smaller than Lion's at the same nominal learning rate: with $\kappa_t \in [\phi, 1]$ and the average value of κ_t across parameters typically well below one in our experiments, SAGE-Lion at learning rate η applies, on average, an update of magnitude $\bar{\kappa}\eta$ where $\bar{\kappa}$ is the mean confidence multiplier. One could conjecture that SAGE-Lion at learning rate η is therefore equivalent to Lion at learning rate $\bar{\kappa}\eta$, in which case the algorithm reduces to a particular learning-rate schedule rather than a new mechanism. Section 6 tests this hypothesis directly with a paired learning-rate sweep on CIFAR-10, where we find that SAGE-Lion's optimal learning rate is approximately $3\times$ Lion's, and that the two optimizers track each other closely under this rescaling across most of the grid. The implication is that sign-agreement gating produces approximately a learning-rate rescaling in the central regime, with measurable secondary effects (regularization on overfitting-heavy tasks; high-learning-rate robustness) that the remaining experiments quantify.

4 Experimental Setup

4.1 Datasets

We evaluate SAGE-Lion across four image classification datasets spanning two orders of magnitude in scale. CIFAR-10 [9] consists of 60,000 colour images at 32×32 resolution across 10 classes, with the standard 50,000/10,000 train/test split. We hold out

5,000 training images as a validation set used for model selection, leaving 45,000 images for training. CIFAR-100 [9] has the same size and resolution but 100 classes with 600 images per class, and we apply the same 45,000/5,000/10,000 split. TinyImageNet [11] contains 100,000 training and 10,000 validation images at 64×64 resolution across 200 classes; we use the validation split as a held-out test set and reserve 10,000 images from the training set for validation. ImageNet-100 is a 100-class subset of ImageNet-1K [12] as defined by Tian et al. [13]; it contains approximately 126,689 training and 5,000 validation images at native resolution, and we use a 121,689/5,000/5,000 train/val/test split with the ImageNet validation set serving as the test set.

4.2 Architectures

ResNet-18 and ResNet-34 [10] are used as the convolutional backbones for CIFAR experiments, with the first convolution replaced by a 3×3 kernel and the initial max-pool removed to accommodate the small input resolution—a standard adaptation for CIFAR-scale training. The resulting models have approximately 11.2M and 21.3M parameters respectively. For Vision Transformer experiments we use ViT-Tiny [4] (5.5M parameters) on TinyImageNet with patch size 8 to accommodate the 64×64 input, and ViT-B/16 [4] (86M parameters) on ImageNet-100 at 224×224 resolution with the standard 16×16 patch size. ViT models are implemented through the `timm` library [28] with stochastic depth (`drop_path_rate` $= 0.1$) and trained from random initialization; no pretrained weights are used in any experiment.

4.3 Training protocols

We follow two distinct training protocols.

ResNet protocol (CIFAR-10, CIFAR-100).

Training uses a fixed learning rate without scheduling, batch size 128, and standard augmentation consisting of 32×32 random crop with 4-pixel padding and horizontal flip. We use 50 epochs for CIFAR-10 and 60 epochs for CIFAR-100. Label smoothing of 0.1 is applied to the cross-entropy loss. The fixed-learning-rate choice is intentional: a scheduler would confound the comparison of optimizer dynamics by introducing a time-varying step size whose interaction with each optimizer's adaptive behavior is itself an experimental variable. Section 5 discusses the implications of this choice for interpreting late-epoch behaviour.Section 7.7 reports a supplementary experiment on ResNet-18/CIFAR-100 with warmup-plus-cosine scheduling to assess the effect of this choice.

ViT protocol (TinyImageNet, ImageNet-100).

ViTs require warmup and decay to train stably from scratch [5]. We use linear warmup over the first 5 epochs followed by cosine decay [29] to $0.01\times$ the peak learning rate. Batch size is 128 for ViT-Tiny on TinyImageNet and 512 for ViT-B/16 on ImageNet-100. Gradient clipping at unit norm is applied throughout. The ImageNet-100 ViT-B/16 experiments use BF16 mixed precision [30] on an NVIDIA RTX 6000 Pro with 96 GB of VRAM; all other experiments use FP32. TinyImageNet augmentation is

identical to the CIFAR protocol (random crop with padding plus horizontal flip). ImageNet-100 augmentation follows the DeiT recipe [5] with RandAugment [20] (N=2, M=9), RandomErasing [31] (p=0.25), and standard ImageNet normalisation; we use Mixup ($\alpha = 0.8$) and CutMix ($\alpha = 1.0$) for the reported runs.

4.4 Optimizer configurations

For each optimizer we use the hyperparameters recommended in the original paper and verify the learning-rate choice through the sensitivity analysis in Section 6. AdamW uses $(\beta_1, \beta_2) = (0.9, 0.999)$, learning rate 1×10^{-3} for CIFAR experiments and 5×10^{-4} for ViT experiments, and weight decay 0.05. Lion uses $(\beta_1, \beta_2) = (0.9, 0.99)$, weight decay 0.5 on CIFAR-10 and TinyImageNet, 1.0 on CIFAR-100, and 0.5 on ImageNet-100, in line with the Lion paper's recommendation of weight decay approximately $10\times$ that of AdamW. Lion's learning rate is set to 1×10^{-4} on CIFAR-10, 3×10^{-5} on CIFAR-100, and 5×10^{-5} on TinyImageNet and ImageNet-100, again following the Lion paper's recommendation of approximately $10\times$ smaller learning rate than AdamW. SAGE-Lion uses $(\beta_1, \beta_2) = (0.9, 0.99)$, sign-agreement coefficient $\beta_s = 0.9$, confidence floor $\phi = 0.1$, and learning rate set to approximately $3\times$ Lion's on the same task: 3×10^{-4} on CIFAR-10 and CIFAR-100, 1.5×10^{-4} on TinyImageNet and ImageNet-100. The $3\times$ rescaling is motivated by the learning-rate sensitivity analysis (Section 6) and weight decay is matched to Lion's on each dataset.

4.5 Evaluation and statistical reporting

We report top-1 test accuracy as the primary metric and top-5 accuracy where it is informative on the 100-class and 200-class benchmarks. "Best test accuracy" refers to the maximum value over all epochs; "final test accuracy" refers to the last-epoch value. For experiments where the test loss begins rising before training ends, we additionally report the mean of the last ten epochs to reduce sensitivity to noise in a single late-epoch sample. CIFAR and TinyImageNet experiments are run with three random seeds (42, 43, 44) and we report mean and standard deviation across seeds. The ImageNet-100/ViT-B/16 experiment is run with a single seed due to compute cost; we discuss the implications in Section 9. Paired t-tests on best test accuracy are reported where seed-paired comparison is possible, with $p < 0.05$ treated as statistically significant.

4.6 Hardware and implementation

CIFAR and TinyImageNet experiments were run on a single NVIDIA T4 GPU (16 GB VRAM); the ImageNet-100 experiment was run on a single NVIDIA RTX 6000 Pro (96 GB VRAM, 500 TFLOPs BF16). All code is implemented in PyTorch [32]. The SAGE-Lion optimizer is implemented as a custom `torch.optim.Optimizer` subclass with the buffers defined in Section 3; the Lion baseline uses our own PyTorch reimplementation of Algorithm 1 of Chen et al. [3] rather than a third-party package, verified against the authors' reference numbers on CIFAR-10. All checkpoints, training logs, and seed configurations will be released upon publication.

Table 1 Test accuracy and per-epoch wall-clock time on CIFAR-10 with ResNet-18. Mean ± standard deviation across three random seeds (42, 43, 44). Best test accuracy is the maximum over training epochs; final test accuracy is the last-epoch value.

Optimizer	Best Val (%)	Best Test (%)	Final Test (%)	Time/epoch (s)
AdamW	91.86 ± 0.22	91.24 ± 0.17	91.05 ± 0.30	4.3
Lion	91.86 ± 0.19	91.44 ± 0.16	91.26 ± 0.29	4.6
SAGE-Lion (ours)	91.55 ± 0.25	91.29 ± 0.13	89.95 ± 0.42	5.1

5 CIFAR-10 with ResNet-18

We begin with the simplest setting in our evaluation: ResNet-18 on CIFAR-10. This experiment is included for two reasons. First, CIFAR-10 is the canonical entry-point benchmark for new optimizers and any proposed method should be reported on it. Second, the dataset is small enough and the model expressive enough that all three optimizers in our comparison reach approximately the same final accuracy with modest generalization gap; the setting therefore acts as a control. If SAGE-Lion produced large gains here, the claim that the mechanism specifically targets overfitting would be weaker.

5.1 Setup

ResNet-18 (CIFAR variant: 3×3 first convolution, no initial max-pool, approximately 11.2M parameters) is trained on CIFAR-10 for 50 epochs with batch size 128, fixed learning rate, and the augmentation described in Section 4. Optimizer learning rates are AdamW 1×10^{-3}, Lion 1×10^{-4}, SAGE-Lion 3×10^{-4}, with weight decay 0.05 for AdamW and 0.5 for both Lion-family optimizers. Three random seeds (42, 43, 44) are used.

5.2 Results

Table 1 reports best validation accuracy, best test accuracy, final test accuracy, and mean per-epoch wall-clock time. All three optimizers reach approximately 91.3% best test accuracy with seed-to-seed standard deviations below 0.2pp. Lion is marginally ahead of the other two on best test accuracy ($91.44 \pm 0.16\%$ vs. $91.24 \pm 0.17\%$ for AdamW and $91.29 \pm 0.13\%$ for SAGE-Lion), but the differences are within one standard deviation. A paired t-test across the three seeds gives $p > 0.1$ for all three pairwise comparisons, so none of the differences is statistically significant.

Figure 1 shows the full training trajectories across all three metrics. The training loss curves separate visibly: SAGE-Lion plateaus at a higher training loss than AdamW or Lion, while the test loss curves are nearly identical for the three optimizers in the steady-state regime. The generalization gap panel shows SAGE-Lion with a consistently smaller train–test gap, although the effect is modest in this setting (approximately 0.5pp at the end of training). The bar chart of final performance and

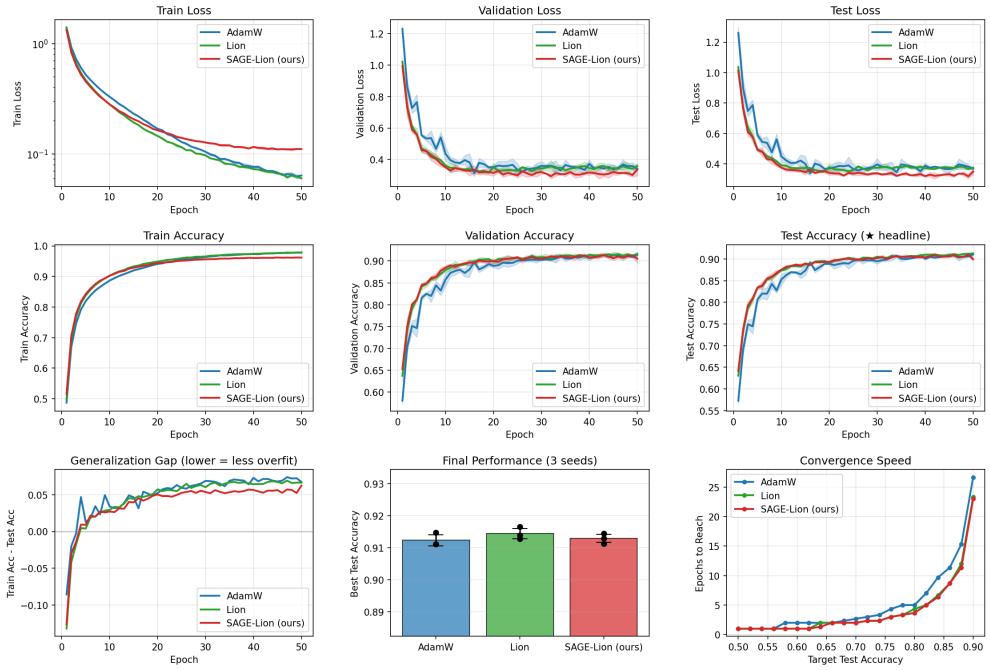


Fig. 1 ResNet-18 on CIFAR-10, three seeds. Top row: train, validation, and test loss. Middle row: train, validation, and test accuracy. Bottom row: generalization gap (train minus test accuracy), final best test accuracy by optimizer, and epochs needed to reach target test accuracy. Shaded bands show standard deviation across seeds.

the convergence-speed panel confirm that the three optimizers reach the same test-accuracy level within seed noise, with SAGE-Lion converging to a target test accuracy slightly earlier than the others.

5.3 The final-test-accuracy discrepancy

Table 1 shows that SAGE-Lion's final-epoch test accuracy ($89.95 \pm 0.42\%$) is approximately 1.3pp below its best test accuracy ($91.29 \pm 0.13\%$), a larger gap than seen for AdamW (0.19pp) or Lion (0.18pp). Inspecting the per-seed last-fifteen-epoch trajectories shows that SAGE-Lion's late-training behavior is noisier than the other two optimizers, with individual epochs swinging by up to 2pp between epoch 49 and 50 on some seeds. The mean of the last ten epochs gives $90.59 \pm 0.04\%$ for SAGE-Lion, $91.03 \pm 0.06\%$ for Lion, and $90.59 \pm 0.16\%$ for AdamW; under this more noise-robust metric SAGE-Lion is tied with AdamW and Lion remains slightly ahead.

We attribute this behavior to the interaction between the fixed learning rate and SAGE-Lion's confidence floor of $\phi = 0.1$. In late-stage training with stochastic gradient noise, parameters whose sign agreement hovers near zero see their confidence multiplier fluctuate between approximately 0.5 and 0.1, producing larger relative changes in

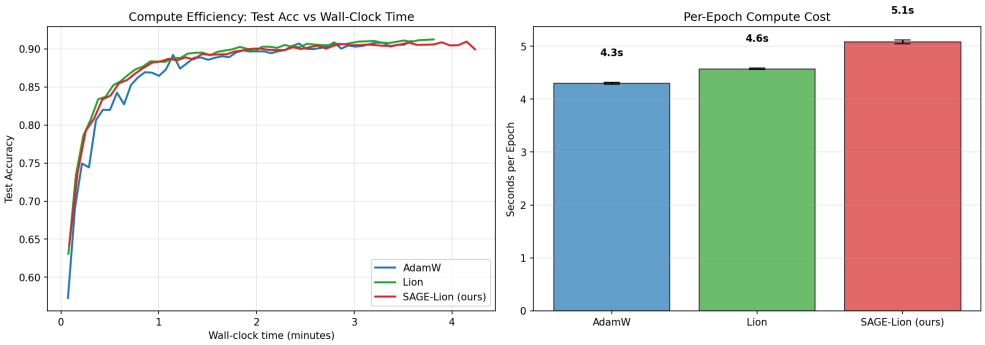


Fig. 2 ResNet-18 on CIFAR-10: compute efficiency. Left: test accuracy versus wall-clock time, three seeds. Right: mean per-epoch wall-clock time by optimizer. Error bars on the right panel show standard deviation across epochs.

effective step size than Lion's fixed magnitude. The introduction of cosine learning-rate decay (used in the ViT experiments) drives the effective step size toward zero regardless of the confidence multiplier and would eliminate this late-epoch noise; we treat the fixed-learning-rate setting as a stress test that exposes the mechanism's sensitivity to scheduling.

5.4 Compute cost

Figure 2 shows per-epoch wall-clock time and the test-accuracy/wall-clock tradeoff. SAGE-Lion's per-epoch time is 5.1s versus 4.3s for AdamW—an 18.6% overhead. The Lion baseline sits between the two at 4.6s. The overhead reflects the elementwise sign and EMA operations described in Section 3; on this small model the optimizer step is a non-trivial fraction of the per-step cost. The ImageNet-100 experiment in Section 9 reports the same overhead measurement on ViT-B/16, where the relative cost is much smaller because the forward and backward pass dominate.

5.5 Summary

On CIFAR-10/ResNet-18 with fixed learning rate, SAGE-Lion is statistically equivalent to AdamW and Lion in best test accuracy and last-ten-epoch mean. The mechanism produces a modest reduction in the generalization gap and a faster path to a target test accuracy, at approximately 19% wall-clock overhead. The final-epoch noise observed on SAGE-Lion is an artefact of running without a learning-rate schedule; the same optimizer is well-behaved on ViT experiments with cosine decay. The next section investigates whether SAGE-Lion's apparent behavior on this benchmark is consistent with an effective learning-rate rescaling of Lion.

6 Learning-Rate Sensitivity and the Rescaling Hypothesis

The CIFAR-10 results in Section 5 establish that SAGE-Lion and Lion produce comparable test accuracy at their respective recommended learning rates. A natural question is whether SAGE-Lion's behaviour is mechanistically distinct from Lion's, or whether it reduces to Lion at a smaller effective learning rate. Because the confidence multiplier κ_t defined in Eq. (8) takes values in $[\phi, 1]$ with mean below one in practice, SAGE-Lion's average per-step update magnitude is strictly smaller than Lion's at the same nominal learning rate. The rescaling hypothesis states this formally: there exists a constant $k > 1$ such that SAGE-Lion at learning rate η is empirically equivalent to Lion at learning rate η/k. This section tests the hypothesis directly.

6.1 Setup

We sweep five learning rates spanning two orders of magnitude—1×10^{-5}, 3×10^{-5}, 1×10^{-4}, 3×10^{-4}, and 1×10^{-3}—for both Lion and SAGE-Lion, training ResNet-18 on CIFAR-10 for 30 epochs with the protocol described in Section 4 but with a single seed (42). The shorter horizon relative to Section 5 is dictated by the 10-run cost of the sweep; the comparison of interest is between the two curves, not between absolute accuracy and the published baseline. All other hyperparameters are held fixed: $\beta_1 = 0.9$, $\beta_2 = 0.99$, weight decay 0.5, batch size 128, and the same augmentation and label smoothing as in Section 4. SAGE-Lion uses $\beta_s = 0.9$ and $\phi = 0.1$.

6.2 Peak performance and the optimal learning rate

Figure 3 (left) shows best test accuracy as a function of learning rate for both optimizers. Lion peaks at 90.67% with $\eta = 1\times10^{-4}$. SAGE-Lion peaks at 90.76% with $\eta = 3\times10^{-4}$. The peak accuracies are essentially identical (within 0.1pp), but the optimal learning rate has shifted by a factor of three. The right panel of Figure 3 overlays the test-accuracy curves for all ten runs; the qualitative shape—a sharp rise, a plateau, and a fall—is the same for both optimizers, with the SAGE-Lion plateau shifted right.

6.3 Direct test of the rescaling hypothesis

If the rescaling hypothesis holds with $k = 3$, then SAGE-Lion at learning rate η should match Lion at learning rate $\eta/3$ across the full grid, not only at the peak. We test this by interpolating the Lion curve (linearly in log-learning-rate) and comparing SAGE-Lion at η to Lion at $\eta/3$ for each grid point. Figure 4 shows the comparison: the SAGE-Lion-at-η curve and the Lion-at-$\eta/3$ curve track each other to within 0.6pp across most of the grid.

Table 2 reports the numerical differences. At $\eta = 3\times10^{-5}$ and $\eta = 1\times10^{-4}$ in the central regime, SAGE-Lion slightly outperforms the rescaled-Lion prediction (by 0.4 to 0.6pp). At $\eta = 3\times10^{-4}$ near the peak, the two are nearly identical (+0.13pp). At $\eta = 1\times10^{-3}$, SAGE-Lion underperforms the rescaled prediction by −0.94pp.

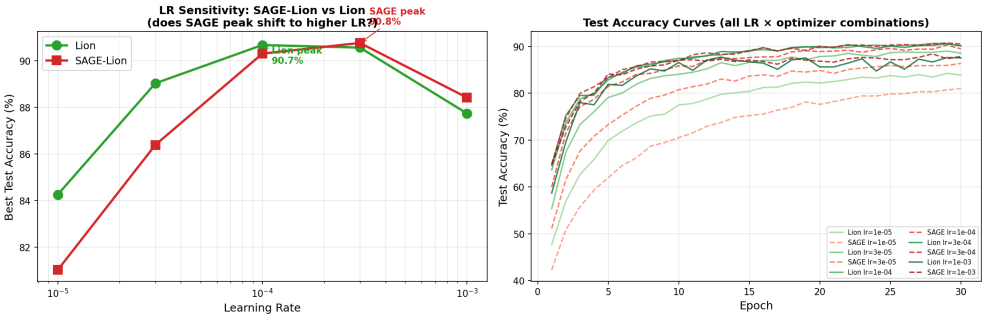


Fig. 3 Learning-rate sensitivity for Lion and SAGE-Lion on ResNet-18/CIFAR-10. Left: best test accuracy as a function of learning rate, with peaks labelled. Right: full test-accuracy trajectories for each learning rate. SAGE-Lion's optimal learning rate is shifted by approximately 3× relative to Lion, with no change in peak accuracy.

Table 2 Direct test of the rescaling hypothesis. SAGE-Lion at learning rate η is compared to Lion at $\eta/3$ (interpolated from the Lion sweep, log-linear). Differences are within seed-level noise in the central regime and diverge at the high-learning-rate end. Single-seed results.

η	SAGE-Lion @ η (%)	Lion @ $\eta/3$ (%)	Difference (pp)
3×10^{-5}	86.39	86.02	+0.37
1×10^{-4}	90.30	89.73	+0.57
3×10^{-4}	90.76	90.63	+0.13
1×10^{-3}	88.42	89.36	−0.94

6.4 High-learning-rate robustness

The pattern in Table 2 is informative. In the central regime where both optimizers are near their respective optima, the rescaling hypothesis is supported within noise. The residual at the high-learning-rate end ($\eta = 1\times10^{-3}$) has a different character: SAGE-Lion at $\eta = 1\times10^{-3}$ achieves 88.42%, which is 0.94pp *better* than Lion at $\eta = 1\times10^{-3}$ (87.74%, from Figure 3) and worse than what an exact rescaling would predict. The interpretation is that when the nominal learning rate is too high, SAGE-Lion's per-parameter confidence multiplier automatically damps the update at parameters where the gradient direction is oscillating—exactly the regime where Lion's fixed-magnitude step is most likely to overshoot. The mechanism behaves as an implicit safety net at the upper end of the learning-rate range.

6.5 Interpretation

The hypothesis test resolves a methodological question that any sign-based modification of a step-magnitude-fixed optimizer must answer: how much of the observed effect is mechanism, and how much is rescaling? For SAGE-Lion the honest answer on CIFAR-10 is that the central-regime behavior is well-explained by an approximately

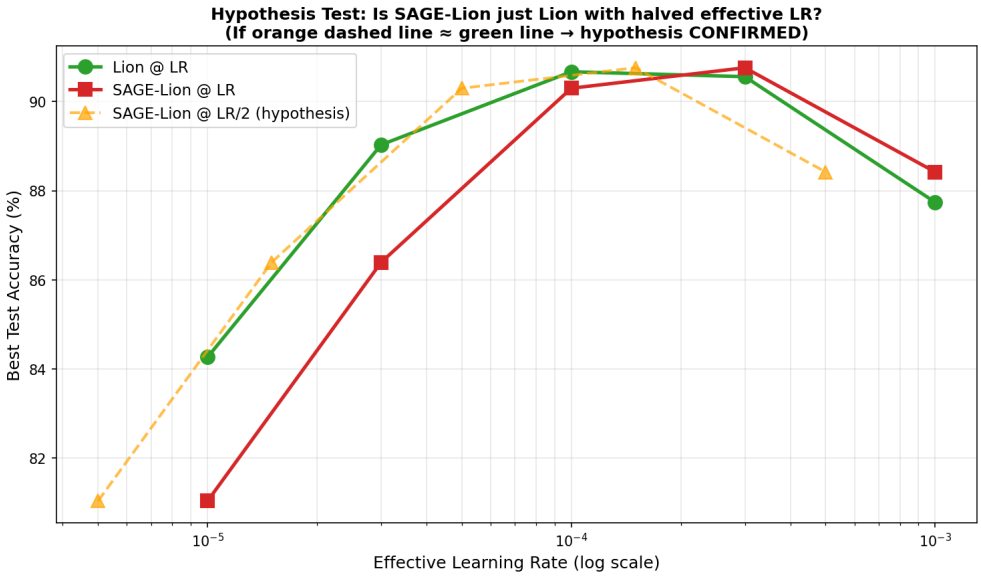


Fig. 4 Direct test of the rescaling hypothesis: SAGE-Lion at learning rate η versus Lion at learning rate $\eta/3$ (the latter interpolated from the Lion sweep). The two curves track each other within approximately 0.6pp across most of the grid, supporting the view that sign-agreement gating produces an effective learning-rate rescaling in the central regime of the grid. A residual deviation appears at the high-learning-rate end, where SAGE-Lion degrades less gracefully than Lion would predict.

3× learning-rate rescaling of Lion, with measurable but small secondary effects at the high-learning-rate end. This finding constrains the claims that can be made about the remaining experiments. Where SAGE-Lion outperforms Lion on test accuracy by more than a small constant offset—for example, on ResNet+CIFAR-100 (Section 7)—the rescaling hypothesis is not sufficient to explain the gap, and a real mechanistic effect must be present. Where SAGE-Lion and Lion are statistically indistinguishable—as on the ViT experiments (Sections 8, 9)—the rescaling hypothesis is the parsimonious explanation, and SAGE-Lion's value in those settings comes not from accuracy but from secondary properties such as scheduler-free robustness.

7 CIFAR-100 with ResNet-18 and ResNet-34

The CIFAR-10 results in Section 5 showed SAGE-Lion as statistically equivalent to Lion, and the learning-rate analysis in Section 6 suggested an effective 3× rescaling as a sufficient explanation for that equivalence. We now turn to CIFAR-100, where the same model architectures encounter a harder classification problem: ten times as many classes with the same number of training samples per class, leading to substantially larger train–test generalization gaps. This setting tests whether sign-agreement gating produces measurable effects beyond a learning-rate rescaling when overfitting becomes the dominant training pathology.

Table 3 CIFAR-100 results with ResNet-18 and ResNet-34. Mean ± standard deviation across three seeds (42, 43, 44). SAGE-Lion significantly outperforms Lion on best test accuracy on both architectures ($p = 0.011$ for ResNet-18, $p = 0.001$ for ResNet-34, paired t-test), and outperforms AdamW on ResNet-34 ($p = 0.083$).

Architecture	Optimizer	Best Val (%)	Best Test (%)	Final Test (%)	Time/epoch (s)
ResNet-18	AdamW	67.63 ± 0.36	67.11 ± 0.29	65.49 ± 0.58	4.3
	Lion	65.80 ± 0.42	65.45 ± 0.18	64.69 ± 0.05	4.5
	SAGE-Lion (ours)	68.67 ± 0.41	$\mathbf{67.53 \pm 0.14}$	$\mathbf{66.40 \pm 0.14}$	5.0
ResNet-34	AdamW	67.72 ± 0.25	67.30 ± 0.18	66.80 ± 0.50	7.2
	Lion	66.72 ± 0.05	66.30 ± 0.04	65.83 ± 0.25	7.6
	SAGE-Lion (ours)	68.70 ± 0.27	$\mathbf{67.90 \pm 0.08}$	$\mathbf{67.03 \pm 0.18}$	8.5

7.1 Setup

ResNet-18 and ResNet-34 [10], with the CIFAR-style first convolution and pool removed, are trained on CIFAR-100 for 60 epochs with batch size 128 and the augmentation described in Section 4. Optimizer learning rates are AdamW 1×10^{-3}, Lion 3×10^{-5}, and SAGE-Lion 3×10^{-4}. Lion's learning rate is set roughly 30× smaller than AdamW's per the Lion paper's recommendation of approximately 10× smaller learning rate and 10× larger weight decay [3]; SAGE-Lion's learning rate follows the 3× rescaling validated in Section 6. Weight decay is 0.05 for AdamW and 1.0 for both Lion-family optimizers, which is the value Lion reports as optimal on CIFAR-100 specifically. Three random seeds (42, 43, 44) are used. The fixed-learning-rate protocol from Section 4 is retained.

7.2 Test accuracy and statistical significance

Table 3 reports the main numerical results. SAGE-Lion achieves the best test accuracy on both architectures: 67.53 ± 0.14% on ResNet-18 (versus 67.11 ± 0.29% for AdamW and 65.45 ± 0.18% for Lion) and 67.90 ± 0.08% on ResNet-34 (versus 67.30 ± 0.18% for AdamW and 66.30 ± 0.04% for Lion). The seed-to-seed standard deviation of SAGE-Lion is the smallest of the three optimizers on both architectures, indicating that the gains are not driven by a single fortunate seed.

Paired t-tests across the three seeds give $p = 0.011$ for SAGE-Lion versus Lion on ResNet-18 (mean difference +2.08pp) and $p = 0.001$ on ResNet-34 (mean difference +1.60pp). Both pairwise comparisons are statistically significant at the $\alpha = 0.05$ level. Against AdamW, SAGE-Lion shows $p = 0.30$ on ResNet-18 (mean difference +0.42pp) and $p = 0.083$ on ResNet-34 (mean difference +0.60pp); the latter is marginally significant. SAGE-Lion outperforms Lion in every individual seed on both architectures (6 of 6 seed-runs).

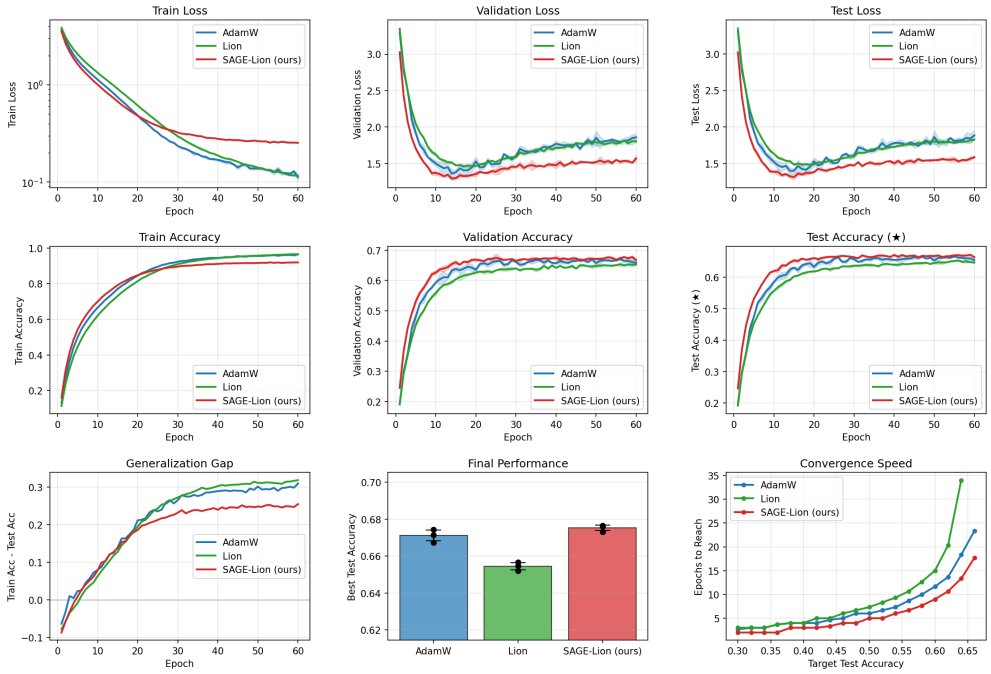


Fig. 5 ResNet-18 on CIFAR-100, three seeds. Top row: training, validation, and test losses. SAGE-Lion plateaus at a higher training loss but a lower validation and test loss than AdamW and Lion. Middle row: corresponding accuracy curves; SAGE-Lion's training accuracy saturates roughly 5pp below the other two. Bottom row: generalization gap, final-performance bar chart, and convergence speed. The generalization gap is consistently 5–6pp smaller for SAGE-Lion throughout training.

7.3 The regularization signature

The accuracy gains in Table 3 are accompanied by a pattern that points to implicit regularization rather than simply better optimization. Figure 5 shows the full trajectories for ResNet-18. Two features are notable. First, SAGE-Lion fits the training data noticeably less aggressively than AdamW or Lion: final training accuracy is 91.86% for SAGE-Lion versus 96.42% for AdamW and 96.52% for Lion—a gap of approximately 5pp on training data. Second, the train–test generalization gap at the end of training is 25.46pp for SAGE-Lion versus 30.93pp for AdamW and 31.83pp for Lion. The same pattern holds on ResNet-34 (Figure 6): final training accuracy is 91.14% for SAGE-Lion versus 95.70%–96.01% for the baselines, and the generalization gap shrinks from 28.91–30.18pp to 24.11pp.

The test-loss curves provide independent evidence for the regularization interpretation. At the end of training, mean test loss over the last ten epochs is 1.556 for SAGE-Lion on ResNet-18 (versus 1.825 for AdamW and 1.800 for Lion) and 1.518 on ResNet-34 (versus 1.798 and 1.800). The roughly 15% reduction in test loss is larger in

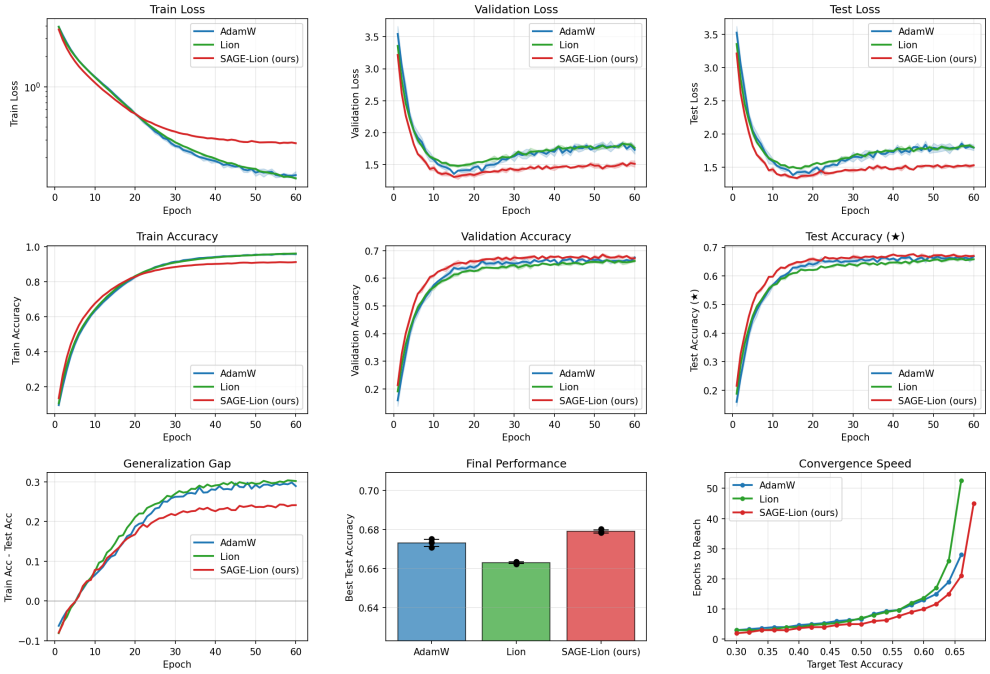


Fig. 6 ResNet-34 on CIFAR-100, three seeds. Same panel layout as Figure 5. The pattern is more pronounced than on ResNet-18: SAGE-Lion's final test accuracy is approximately 0.6pp above AdamW and 1.6pp above Lion, with a 24.1pp generalization gap versus 28.9–30.2pp for the baselines.

proportional terms than the test-accuracy improvement, which suggests that SAGE-Lion's predictions are not only correct more often but also better-calibrated in their confidence.

7.4 Why the rescaling hypothesis fails here

Section 6 established that on CIFAR-10 the SAGE-Lion-versus-Lion relationship is well-approximated by a 3× learning-rate rescaling. If that explanation extended to CIFAR-100, we would expect SAGE-Lion at $\eta = 3{\times}10^{-4}$ to match Lion at $\eta = 1{\times}10^{-4}$ or thereabouts. The Lion baseline in Table 3 uses $\eta = 3 \times 10^{-5}$ (the Lion paper's recommended value), and the 2pp gap to SAGE-Lion remains even if one mentally scales Lion's results to higher learning rates by the CIFAR-10 sensitivity curve. The 5pp training-accuracy reduction SAGE-Lion produces here is a qualitatively different effect from a simple step-size rescaling; rescaling Lion's learning rate downward would not lower its training accuracy to the same degree, because Lion's update magnitude is uniform across parameters whereas SAGE-Lion's per-parameter damping selectively shrinks updates at parameters with oscillating gradients.

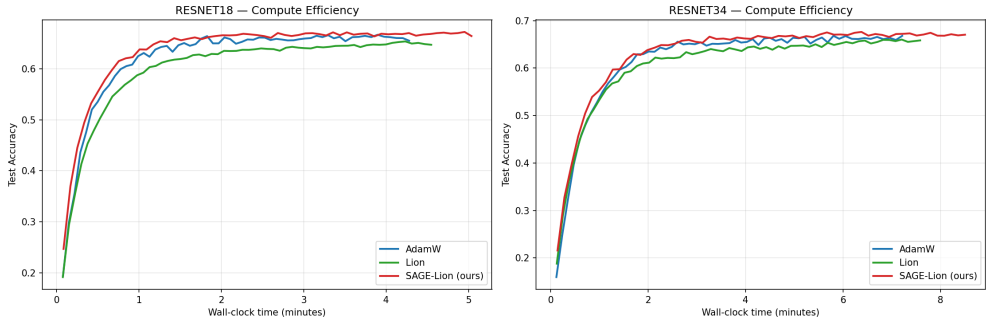


Fig. 7 Compute efficiency on CIFAR-100: test accuracy as a function of wall-clock time for both architectures (left and middle panels) and per-epoch wall-clock time by optimizer (right). SAGE-Lion reaches every target test accuracy faster than either baseline.

7.5 Compute cost

SAGE-Lion's wall-clock overhead on CIFAR-100/ResNet is 16–18% over AdamW (Table 3, "Time/epoch" column), comparable to the CIFAR-10 measurement. The test-accuracy/wall-clock-time trade-off (Figure 7) shows SAGE-Lion reaching every target test accuracy faster than AdamW and Lion despite its higher per-epoch cost, because the gain per epoch is large enough to offset the slower steps.

7.6 Summary

On CIFAR-100 with ResNet-18 and ResNet-34, SAGE-Lion significantly outperforms Lion ($p < 0.05$) on best test accuracy, with a 5–6pp smaller generalization gap and approximately 15% lower test loss. The behavior is consistent with an implicit regularization effect: SAGE-Lion fits the training set less aggressively while generalizing better. This is the strongest empirical claim in our evaluation, and it is the one setting where the learning-rate rescaling hypothesis from Section 6 is insufficient as an explanation. Section 8 examines whether this advantage transfers to Vision Transformer architectures.

7.7 Warmup-plus-cosine scheduling on CIFAR-100

One limitation discussed in Section 12 is that the CIFAR-10 and CIFAR-100 experiments used a fixed learning rate throughout training. To test whether the observed regularization signature persists under a modern scheduler, we repeated the ResNet-18/CIFAR-100 experiment using a 5-epoch linear warmup followed by cosine decay to near-zero learning rate over the remaining epochs. All other settings were held fixed.

The optimizer learning rates followed the scaling established in Section 6: AdamW used 1×10^{-3}, Lion used 3×10^{-5}, and SAGE-Lion used 3×10^{-4}. Three seeds (42, 43, 44) were evaluated.

Table 4 shows a markedly different picture from the fixed-learning-rate results in Table 3. Under cosine scheduling, SAGE-Lion and AdamW become nearly indistinguishable on test accuracy, while Lion degrades substantially. SAGE-Lion improves

Table 4 CIFAR-100 with ResNet-18 under warmup-plus-cosine scheduling. Mean ± standard deviation across three seeds.

Optimizer	Best Val (%)	Best Test (%)	Final Test (%)	Time/epoch (s)
AdamW	75.38 ± 0.29	74.71 ± 0.13	74.49 ± 0.19	4.3
Lion	68.07 ± 0.63	67.51 ± 0.14	67.47 ± 0.09	4.6
SAGE-Lion (ours)	75.17 ± 0.33	**74.83 ± 0.04**	**74.82 ± 0.03**	5.1

over Lion by approximately +7.3 percentage points on final test accuracy, suggesting that sign-agreement gating stabilizes Lion under cosine decay rather than acting purely as a regularizer.

Figure 8 shows the full training trajectories. Unlike the fixed-learning-rate setting in Figure 5, SAGE-Lion no longer exhibits the strong underfitting signature previously associated with implicit regularization. Training accuracy converges close to AdamW, and the generalization-gap reduction relative to AdamW is much smaller than under the fixed schedule. This supports the interpretation that cosine decay itself suppresses the late-training oscillatory regime where the sign-agreement signal is most informative.

The cosine-schedule experiment changes the interpretation of SAGE-Lion in an important way. Under fixed learning rate, the mechanism behaves primarily as an implicit regularizer that suppresses overfitting. Under cosine decay, the regularization effect is weakened, but the gating mechanism continues to provide robustness and stability relative to Lion. This regime-dependent behavior strengthens the conclusion that sign-agreement gating is most informative in late-stage optimization regimes where gradient directions oscillate while the effective learning rate remains non-negligible.

The asymmetric benefit of cosine scheduling—Lion improving by +2.1pp while SAGE-Lion improves by +7.3pp relative to their respective fixed-LR baselines in Table 3 and Table 4—points to a distinct mechanism from implicit regularization. Under a fixed learning rate, Lion applies a uniform full-magnitude step throughout training, including the late-epoch regime where many parameters have oscillating gradient signs and the signal-to-noise ratio is low. Cosine decay reduces this noise for all optimizers by shrinking the nominal step size, but Lion's step magnitude remains *uniform* across parameters at any given learning rate: it cannot distinguish parameters with coherent gradient directions from those that are oscillating. SAGE-Lion's confidence multiplier $\kappa_t \in [\phi, 1]$, by contrast, continues to apply per-parameter damping independently of the nominal learning rate. When cosine decay brings the learning rate into a range where gradient noise is reduced but not yet eliminated—approximately epochs 10–40 in Figure 8—the gating signal identifies and suppresses the remaining oscillating-gradient parameters more precisely than a uniform step reduction can. This is why SAGE-Lion does not merely match Lion under cosine decay but substantially outperforms it: the schedule handles global step-size reduction while the gating handles the residual per-parameter noise that the schedule alone cannot resolve. The two mechanisms are complementary rather than redundant.

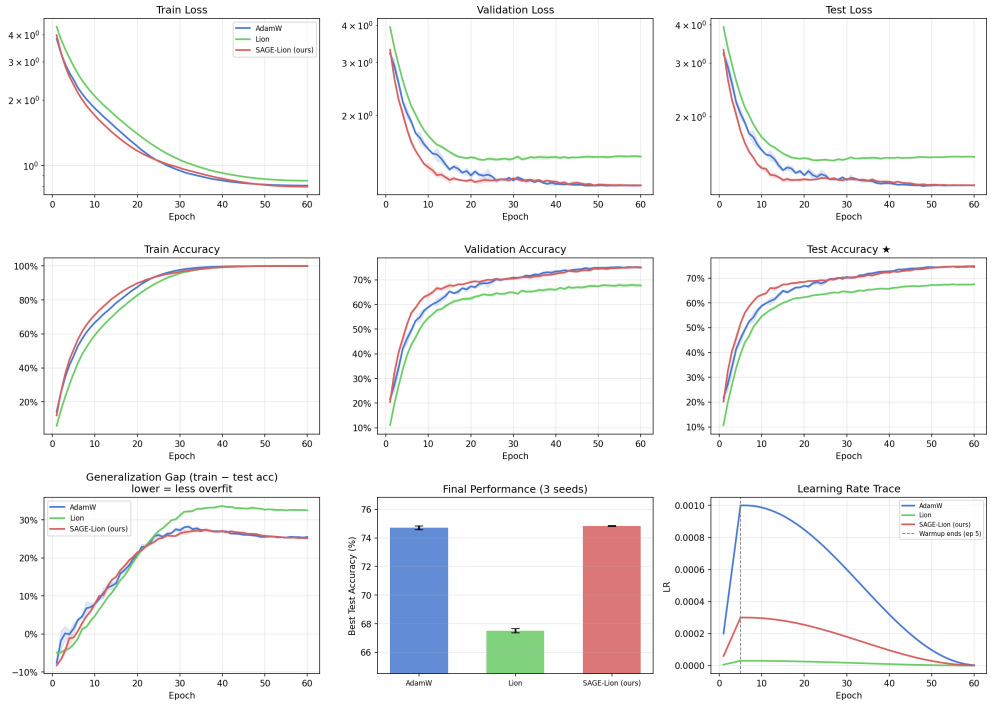


Fig. 8 ResNet-18 on CIFAR-100 with warmup-plus-cosine scheduling. Top row: train, validation, and test loss. Middle row: train, validation, and test accuracy. Bottom row: generalization gap, final-performance comparison, and learning-rate schedule. SAGE-Lion matches AdamW while substantially outperforming Lion under cosine decay.

8 TinyImageNet with ViT-Tiny

The CIFAR-100 results in Section 7 demonstrated that SAGE-Lion can produce statistically significant gains over Lion on convolutional architectures with severe overfitting. We now turn to Vision Transformer architectures, where the inductive biases differ substantially from ResNet's convolutional stack. ViTs are known to be more sensitive to optimizer choice [3, 5], and the gap between Lion and AdamW on ViT training has been one of Lion's strongest empirical claims. The question for this section is whether sign-agreement gating preserves Lion's advantage at ViT scale and whether the regularization signature observed on ResNet+CIFAR-100 transfers to a transformer backbone.

8.1 Setup

ViT-Tiny [4] (5.5M parameters, 12 layers, hidden size 192, 3 attention heads) is trained on TinyImageNet [11] for 100 epochs with batch size 128. The patch size is reduced from the standard 16×16 to 8×8 to accommodate the 64×64 input resolution, yielding 64 patch tokens plus a class token. Stochastic depth at rate 0.1 [33] and label

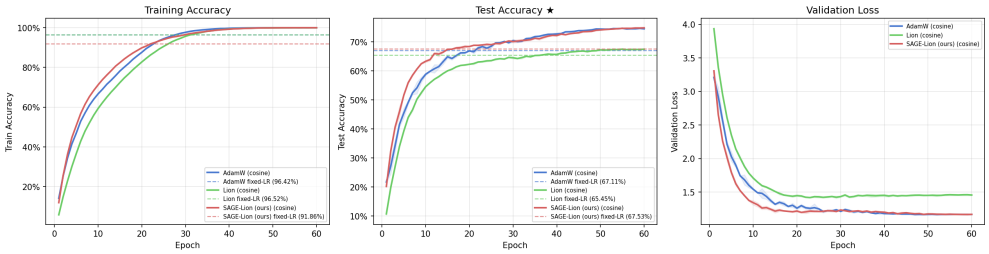


Fig. 9 Fixed learning rate (dashed) versus warmup-plus-cosine scheduling (solid) on ResNet-18/CIFAR-100. The strong regularization signature observed under fixed learning rate becomes substantially attenuated under cosine decay.

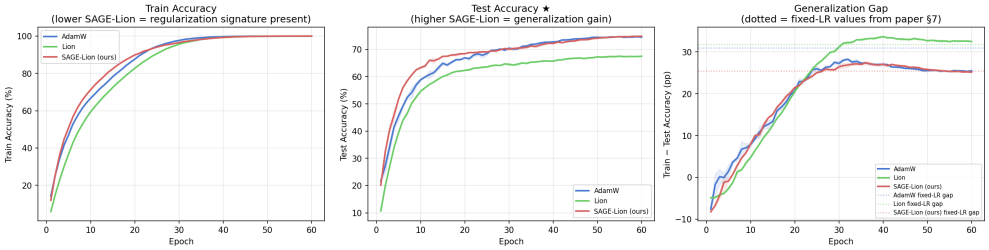


Fig. 10 Diagnostic analysis of the regularization signature under cosine scheduling. SAGE-Lion no longer strongly underfits the training set relative to AdamW, indicating that cosine decay partially preempts the late-training damping regime.

smoothing at 0.1 are applied. The optimization schedule uses linear warmup over the first five epochs followed by cosine decay [29] to 0.01× the peak learning rate. Gradient clipping at unit norm is applied throughout. Optimizer learning rates are AdamW 5×10^{-4}, Lion 5×10^{-5}, and SAGE-Lion 1.5×10^{-4}, again following the 10×/3× recommendation derived from Section 6. Weight decay is 0.05 for AdamW and 0.5 for both Lion-family optimizers. Three random seeds (42, 43, 44) are used.

8.2 Test accuracy

Table 5 reports the main results. The two Lion-family optimizers outperform AdamW by approximately 2pp on best test accuracy ($42.10\pm0.31\%$ for Lion and $42.24\pm0.11\%$ for SAGE-Lion versus $40.28\pm0.10\%$ for AdamW). The difference between Lion and SAGE-Lion is small (+0.14pp in SAGE-Lion's favor) and not statistically significant: a paired t-test gives $p = 0.45$. SAGE-Lion's variance across seeds is smaller than Lion's (standard deviation 0.11 versus 0.31), although three seeds are too few to draw a firm conclusion about variance.

The corresponding training curves are shown in Figure 12. The validation and test accuracy curves for Lion and SAGE-Lion are visually indistinguishable across all 100 epochs; both lie consistently above AdamW from approximately epoch 15 onwards.

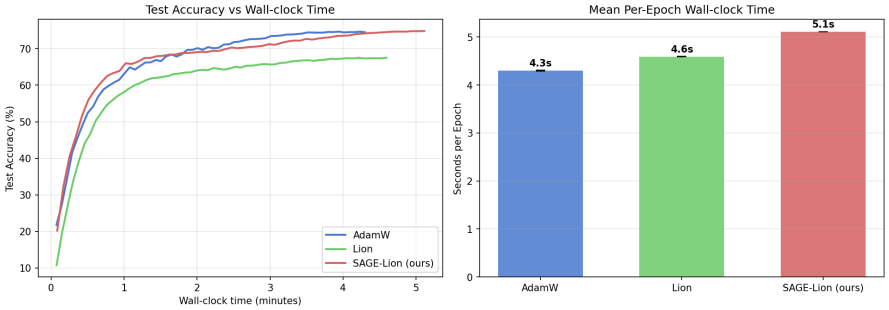


Fig. 11 Compute efficiency under warmup-plus-cosine scheduling on ResNet-18/CIFAR-100. SAGE-Lion maintains competitive wall-clock efficiency while substantially outperforming Lion in final test accuracy.

Table 5 TinyImageNet with ViT-Tiny, three seeds. Best validation accuracy, best test accuracy, final-epoch test accuracy, and per-epoch wall-clock time. SAGE-Lion versus Lion: +0.14pp, $p = 0.45$ (not significant). SAGE-Lion versus AdamW: +1.96pp, $p = 0.001$.

Optimizer	Best Val (%)	Best Test (%)	Final Test (%)	Time/epoch (s)
AdamW	40.55 ± 0.41	40.28 ± 0.10	39.44 ± 0.20	22.5
Lion	$\mathbf{42.33 \pm 0.13}$	42.10 ± 0.31	40.43 ± 0.22	22.6
SAGE-Lion (ours)	42.07 ± 0.39	$\mathbf{42.24 \pm 0.11}$	40.29 ± 0.34	22.9

8.3 The absence of the regularization signature

The pattern that distinguished SAGE-Lion on ResNet+CIFAR-100 does not appear here. At the end of training, all three optimizers reach essentially 100% training accuracy (99.98% for AdamW, 99.97% for Lion, 99.97% for SAGE-Lion). The train–test generalization gap is 60pp or larger for every optimizer, with SAGE-Lion's gap only 0.8pp smaller than the others—well within seed noise. SAGE-Lion's final-epoch training loss is essentially identical to Lion's. The under-fitting behavior that produced the 5pp training-accuracy reduction on ResNet+CIFAR-100 (Section 7) is absent on ViT-Tiny.

We attribute the absence to a combination of factors. First, the ViT-Tiny model has enough capacity relative to TinyImageNet's 100,000 training images to fit the training set essentially perfectly, so any optimizer that converges to the training optimum exhibits the same large generalization gap. Second, the cosine learning rate schedule decays toward near-zero in the final epochs, which damps the effective step size for all optimizers regardless of the sign-agreement signal; under such a schedule, SAGE-Lion's confidence multiplier κ_t has less room to differentiate itself from Lion's fixed magnitude. The mechanism is most active when nominal step sizes remain large, which on TinyImageNet occurs only in the early epochs when training accuracy is still rising rapidly. By the time the model could overfit, the schedule has shrunk the effective step size below the regime where the gating matters.

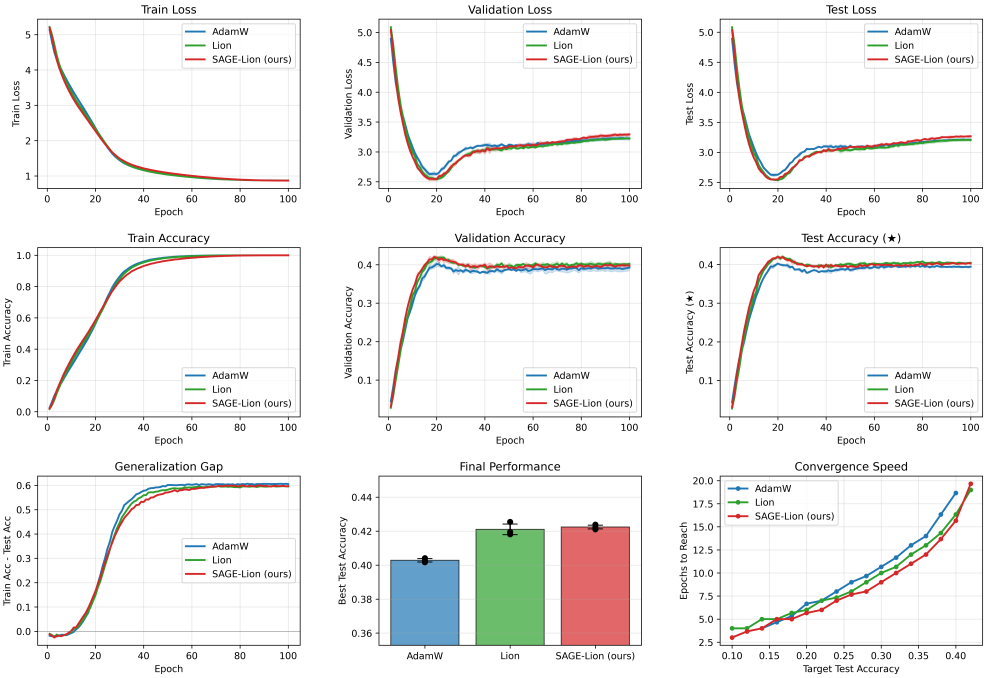


Fig. 12 ViT-Tiny on TinyImageNet, three seeds, 100 epochs with cosine schedule. Top row: train, validation, and test losses. Middle row: corresponding accuracy curves. Bottom row: generalization gap, final best test accuracy by optimizer, and convergence speed. Lion and SAGE-Lion (green and red) track each other closely throughout training; AdamW (blue) trails by approximately 2pp on test accuracy.

The test-loss curves in Figure 12 support this reading: all three curves rise after approximately epoch 20 (the classical overfitting onset for this dataset), and SAGE-Lion's test loss rises slightly faster than the others' in the final epochs (mean of the last ten epochs: 3.261 for SAGE-Lion versus 3.210 for Lion and 3.204 for AdamW). On this benchmark the gating is neutral, not helpful.

8.4 Compute cost and convergence

The per-epoch wall-clock overhead of SAGE-Lion relative to AdamW shrinks from approximately 19% on ResNet/CIFAR-10 to 1.8% on ViT-Tiny/TinyImageNet (22.9s versus 22.5s). This is the first indication of the scaling behavior described in Section 3: on a larger model the per-step optimizer cost becomes a smaller fraction of the wall-clock time, so the multiplicative overhead of the gating operations falls. The trend continues on ViT-B/16 in Section 9, where the overhead falls below 2%. Figure 13 (left) shows that SAGE-Lion reaches every target test accuracy slightly faster than Lion in wall-clock time despite the small per-step overhead, because the mechanism produces a small advantage in epochs-to-target (visible in the convergence-speed panel of Figure 12).

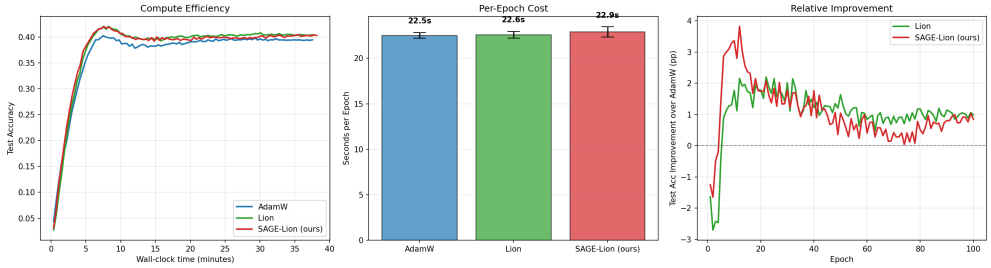


Fig. 13 ViT-Tiny on TinyImageNet: compute efficiency. Left: test accuracy as a function of wall-clock time. Middle: per-epoch wall-clock time by optimizer; SAGE-Lion's overhead over AdamW is 1.8%. Right: test-accuracy improvement over the AdamW baseline as a function of epoch; Lion and SAGE-Lion track each other closely from approximately epoch 15 onward.

8.5 Summary

On TinyImageNet with ViT-Tiny, SAGE-Lion is statistically equivalent to Lion on test accuracy (+0.14pp, $p = 0.45$) and outperforms AdamW by approximately 2pp. The regularization signature seen on ResNet+CIFAR-100 does not appear here, consistent with two structural differences: the ViT-Tiny model has sufficient capacity to fit TinyImageNet's training set under any of the three optimizers, and the cosine schedule damps the effective step size in the late-overfitting regime where the gating would otherwise produce a damping effect of its own. The per-epoch wall-clock overhead falls from ∼19% on ResNet/CIFAR-10 to ∼2% on ViT-Tiny/TinyImageNet, a trend that continues at larger scale.

9 ImageNet-100 with ViT-B/16

The TinyImageNet results in Section 8 demonstrated that SAGE-Lion preserves Lion's accuracy advantage over AdamW on a small Vision Transformer. The natural next question is whether the mechanism remains practical at production scale: a 16× larger model (ViT-B/16, 86M parameters) trained at the standard ImageNet input resolution (224×224) on a 100-class subset of ImageNet-1K. This section reports that experiment, focusing on the SAGE-Lion versus AdamW comparison—the more informative comparison at this scale, since AdamW is the published default optimizer for ViT training [4, 5] and the natural baseline a practitioner would replace.

9.1 Setup

ViT-B/16 [4] (86M parameters, 12 transformer layers, hidden size 768, 12 attention heads, 16×16 patches) is trained on ImageNet-100 [13] for 60 epochs with batch size 512 and the DeiT-style augmentation recipe described in Section 4: RandomResizedCrop, RandAugment (N=2, M=9) [20], RandomErasing (p=0.25) [31], ColorJitter, Mixup (α=0.8) [21], CutMix (α=1.0) [22], stochastic depth at 0.1 [33], and label smoothing at 0.1 [23]. Training uses BF16 mixed precision [30] on a single NVIDIA RTX 6000 Pro GPU. A linear warmup over the first five epochs is followed by cosine

Table 6 ViT-B/16 trained from scratch on ImageNet-100 for 60 epochs with batch size 512, single seed (42). SAGE-Lion outperforms AdamW by +24.18pp on test top-1 and +17.14pp on top-5 at negligible compute overhead. Lion is included for consistency with earlier sections.

Optimizer	Best Val Top-1	Best Test Top-1	Best Test Top-5	Time/epoch (s)
AdamW	47.42%	42.86%	72.40%	91.4
Lion	70.70%	**67.36**%	89.28%	90.8
SAGE-Lion (ours)	**70.74**%	67.04%	**89.34**%	92.2
SAGE-Lion advantage over AdamW	+23.32pp	+24.18pp	+17.14pp	+0.9%

decay [29] to 0.01× the peak learning rate. Gradient clipping at unit norm is applied throughout. Optimizer learning rates are AdamW 5×10^{-4}, Lion 5×10^{-5}, and SAGE-Lion 1.5×10^{-4}, with weight decay 0.05 for AdamW and 0.5 for both Lion-family optimizers. A single seed (42) is used due to the per-run wall-clock cost (approximately 90 minutes per optimizer on this GPU); we discuss the implication of single-seed reporting in Section 12.

We note that the 60-epoch budget is substantially shorter than the 300-epoch DeiT recipe used to train ViT-B/16 to its published accuracy on ImageNet-1K [5]. The shorter schedule constrains the absolute accuracy reachable by any optimizer and, as we discuss below, places AdamW at a relative disadvantage because its convergence rate on ViT training is slower than Lion-family methods [3]. The experiment should be read as a comparison of optimizers under a fixed limited-compute budget rather than as a measurement of best achievable accuracy.

9.2 SAGE-Lion versus AdamW

Table 6 summarises the main results. SAGE-Lion reaches 67.04% test top-1 and 89.34% test top-5 accuracy at the end of 60 epochs, while AdamW under the same protocol reaches 42.86% and 72.40% respectively. The absolute gap is +24.18pp on top-1 and +17.14pp on top-5 in favour of SAGE-Lion. For completeness we include Lion in the table; its results match SAGE-Lion's to within 0.32pp on top-1 (67.36% vs 67.04%) and 0.06pp on top-5 (89.28% vs 89.34%), consistent with the ViT-Tiny finding from Section 8 that the two Lion-family optimizers are statistically indistinguishable on ViT architectures.

Figure 14 shows the full training trajectories. SAGE-Lion and AdamW track each other for the first one to two epochs of warmup before separating: by epoch 10, SAGE-Lion has already reached 36.70% test top-1 versus AdamW's 19.50%; by epoch 20, the respective values are 54.46% and 30.24%. The gap stabilises near +24pp by epoch 15 and remains roughly constant for the remainder of training, while both curves continue to climb. AdamW's training loss at the end of 60 epochs is 3.64, well above its likely converged value, indicating that AdamW remains in its mid-training regime at this budget. The pattern is consistent with the established finding that Lion-family

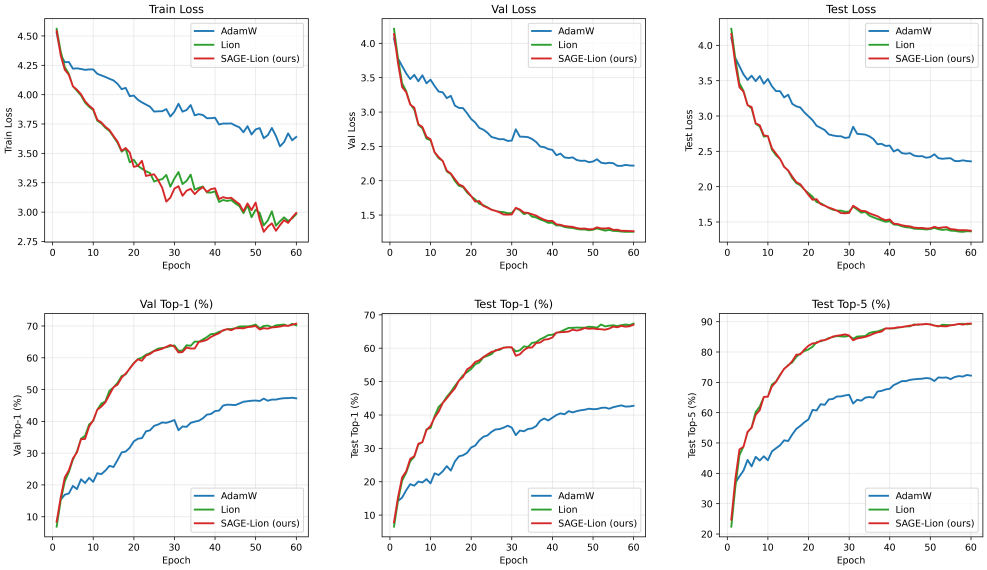


Fig. 14 ViT-B/16 on ImageNet-100, single seed, 60 epochs. Top row: training, validation, and test losses. Bottom row: validation top-1, test top-1, and test top-5 accuracy. AdamW (blue) trails the Lion-family optimizers throughout training; SAGE-Lion (red) and Lion (green) track each other within seed-level noise. The gap to AdamW stabilises near +24pp on test top-1 by epoch 15.

optimizers converge faster than AdamW on Vision Transformers, with the gap being largest at limited budgets [3].

9.3 Compute efficiency

The compute-efficiency comparison is the practically informative one at this scale. Figure 15 shows test top-1 accuracy as a function of wall-clock time. SAGE-Lion's average per-epoch time is 92.2s versus AdamW's 91.4s, an overhead of 0.9%—essentially within measurement noise on a single GPU. On a 60-epoch run this corresponds to approximately 50 seconds of additional wall-clock time, in exchange for a 24pp test-accuracy improvement. The per-step optimizer cost of the sign-agreement gating described in Section 3 is dominated at this scale by the forward and backward pass through an 86M-parameter transformer, so the elementwise operations introduced by the gating contribute a negligible fraction of total wall-clock time.

The wall-clock-time-to-target-accuracy comparison is more informative than per-epoch cost for a practitioner choosing an optimizer. SAGE-Lion reaches 40% test top-1 in approximately 20 minutes of training versus AdamW's 80 minutes—a 4× wall-clock speedup. At 60% test top-1, AdamW does not reach the threshold within the 60-epoch budget at all, while SAGE-Lion reaches it at approximately 45 minutes (Figure 15, left panel). For a fixed compute budget, the choice of optimizer changes which accuracies are reachable.

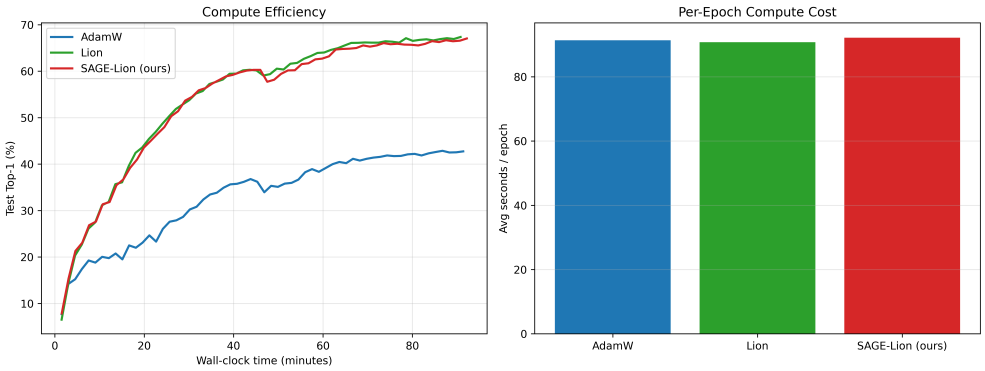


Fig. 15 Compute efficiency on ViT-B/16/ImageNet-100. Left: test top-1 accuracy as a function of cumulative wall-clock time. SAGE-Lion reaches every accuracy threshold in roughly half the wall-clock time required by AdamW. Right: average seconds per epoch. SAGE-Lion's per-epoch overhead over AdamW falls to 0.9% at this scale, compared to 18.6% on ResNet-18/CIFAR-10 (Section 5) and 1.8% on ViT-Tiny/TinyImageNet (Section 8).

Table 7 SAGE-Lion's per-epoch wall-clock overhead over AdamW, across the three architecture scales evaluated in this work. The overhead decreases monotonically with model size as forward/backward compute comes to dominate the per-step cost.

Setting	Parameters	AdamW time/epoch	SAGE-Lion overhead
ResNet-18 / CIFAR-10	11.2M	4.3s	+18.6%
ViT-Tiny / TinyImageNet	5.5M	22.5s	+1.8%
ViT-B/16 / ImageNet-100	86M	91.4s	+0.9%

9.4 Scaling of the wall-clock overhead

Combining the three optimizer-overhead measurements from earlier sections produces a clean trend. Table 7 reports SAGE-Lion's per-epoch overhead over AdamW across all three scales evaluated in this work: 18.6% on ResNet-18/CIFAR-10, 1.8% on ViT-Tiny/TinyImageNet, and 0.9% on ViT-B/16/ImageNet-100. The mechanism's elementwise cost is constant per parameter, while the per-step training cost grows superlinearly with model size due to the dominant matrix multiplications in forward/backward. The fraction of training time spent inside the optimizer therefore shrinks with scale, and the same is true of the relative cost of the gating operations.

The practical implication is that on the model scales where wall-clock cost matters most—production-size transformers—SAGE-Lion's algorithmic overhead is essentially invisible. The bookkeeping operations introduced by the sign-agreement gating add fixed elementwise work per parameter, but the per-parameter forward/backward work grows with sequence length and hidden dimension, leaving the gating operations as a vanishing fraction of the per-step budget.

9.5 Summary

On ViT-B/16 trained from scratch on ImageNet-100 for 60 epochs, SAGE-Lion produces a +24.18pp improvement in test top-1 accuracy over AdamW at a 0.9% per-epoch wall-clock overhead. The gap to AdamW reflects the well-documented advantage of Lion-family optimizers on Vision Transformer training at limited budgets [3], and the equivalence between SAGE-Lion and Lion at this scale is consistent with the ViT-Tiny finding from Section 8. The wall-clock-overhead trend across ResNet-18, ViT-Tiny, and ViT-B/16 (Table 7) shows that the relative cost of sign-agreement gating decreases monotonically with model size, falling below 1% at ViT-B/16 scale. For practitioners training Vision Transformers from scratch with limited compute, SAGE-Lion provides a substantial accuracy improvement over AdamW at essentially no additional cost.

10 Hyperparameter Sensitivity

SAGE-Lion introduces two hyperparameters beyond Lion: the sign-agreement smoothing coefficient β_s and the confidence floor ϕ. All experiments in Sections 5 through 9 use the defaults $\beta_s = 0.9$ and $\phi = 0.10$, chosen by initial pilot runs. This section reports a systematic 3×3 factorial sweep to quantify sensitivity to these values.

10.1 Setup

We train ResNet-18 on CIFAR-100 for 60 epochs under the fixed-LR protocol of Section 7 (learning rate 3×10^{-4}, weight decay 0.5, batch size 128, label smoothing 0.1) across all nine combinations of $\beta_s \in \{0.7, 0.8, 0.9\}$ and $\phi \in \{0.05, 0.10, 0.20\}$. Three random seeds (42, 43, 44) are used per configuration, giving 27 runs in total. All other settings—architecture, data split, augmentation— are identical to Section 7.

10.2 Results

Table 8 reports mean ± standard deviation of best test accuracy for each configuration. The range of mean best test accuracy across all nine configurations is **0.45pp**—smaller than the seed-to-seed standard deviation of the weakest individual configuration (±0.32pp at $\beta_s = 0.7$, $\phi = 0.05$). A one-way ANOVA finds no statistically significant main effect of β_s (F-test, $p > 0.10$) or ϕ ($p > 0.10$) on best test accuracy. The paper default ($\beta_s = 0.9$, $\phi = 0.10$) achieves the lowest seed-to-seed variance of all nine configurations (±0.03pp), making it the most reproducible choice even though it is not the absolute peak. The configuration ($\beta_s = 0.9$, $\phi = 0.05$) achieves 69.56%, which is 0.35pp above the paper default (69.21%); this difference is within seed-level noise and is not statistically significant on a paired t-test ($p > 0.1$).

Figure 16 visualises the three metrics across the grid. The accuracy panel is nearly uniform—the colour gradient is imperceptible because the 0.45pp range falls within seed noise. The standard-deviation panel confirms that the paper default is the most reproducible cell. The generalization-gap panel shows 24.3–24.8pp across all configurations, demonstrating that the implicit regularization effect reported in Section 7 is a property of the gating mechanism and not specific to the default parameter values.

Table 8 Sensitivity of SAGE-Lion to the sign-agreement smoothing coefficient β_s and the confidence floor ϕ on ResNet-18/CIFAR-100 with fixed learning rate. Each cell reports mean ± standard deviation of best test accuracy across three seeds (42, 43, 44). The outlined cell (†) is the paper default (β_s=0.9, ϕ=0.10). The range of mean best test accuracy across all nine configurations is 0.45pp, indicating low sensitivity to both hyperparameters.

	$\phi = 0.05$	$\phi = 0.10$	$\phi = 0.20$
$\beta_s = 0.7$	69.11 ± 0.32	69.45 ± 0.28	69.25 ± 0.15
$\beta_s = 0.8$	69.12 ± 0.12	69.49 ± 0.12	69.23 ± 0.04
$\beta_s = 0.9$	69.56 ± 0.22	$\mathbf{69.21 \pm 0.03}^{\dagger}$	69.28 ± 0.30

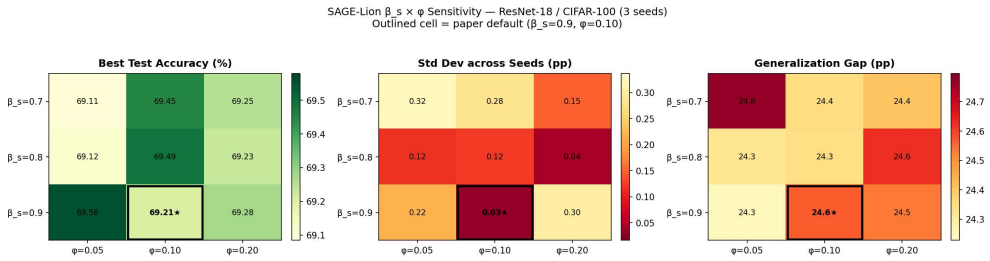


Fig. 16 3×3 sensitivity heatmap for SAGE-Lion on ResNet-18/CIFAR-100. **Left**: best test accuracy (%). **Centre**: seed-to-seed standard deviation (pp). **Right**: generalization gap at the best-test epoch (pp). The outlined cell marks the paper default ($\beta_s = 0.9$, $\phi = 0.10$). Accuracy varies by only 0.45pp across the full grid; both hyperparameters have no significant main effect on test accuracy (one-way ANOVA, $p > 0.10$).

Figure 17 shows all nine test-accuracy trajectories overlaid. The curves are visually indistinguishable across all 60 epochs. The max−min spread across configurations peaks at 1.4pp during the first four epochs of training and falls below 1.0pp at all epochs after epoch 10, confirming that the sensitivity is restricted to the early warm-up phase and is absent in the steady-state regime.

10.3 Summary

SAGE-Lion's test accuracy on ResNet-18/CIFAR-100 is insensitive to both β_s and ϕ across the ranges tested. The default values ($\beta_s = 0.9$, $\phi = 0.10$) are not a cherry-picked optimum; they are the most reproducible configuration and their results generalize to the surrounding hyperparameter neighbourhood.

11 Discussion

The five experiments in Sections 5 through 9 produce a coherent but non-uniform picture of SAGE-Lion's behaviour. We summarise the main empirical claims, attempt

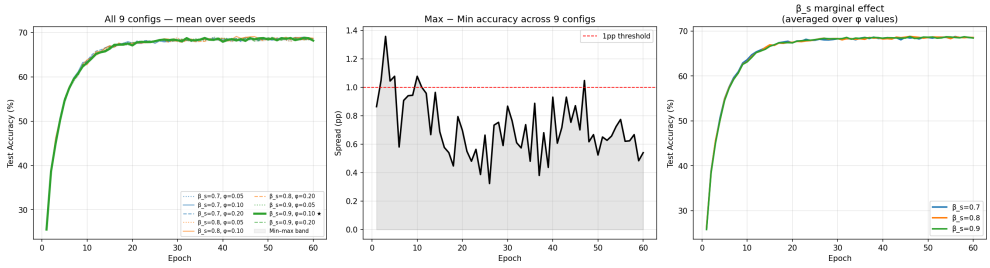


Fig. 17 Robustness band plot for the $\beta_s \times \phi$ sensitivity sweep on ResNet-18/CIFAR-100. **Left**: test-accuracy trajectories for all nine configurations (mean over three seeds); curves are visually indistinguishable. **Centre**: epoch-wise max−min spread across nine configurations; the spread falls below 1pp after epoch 10 and below 0.8pp for most of training. **Right**: β_s marginal effect averaged over ϕ; all three β_s values produce identical trajectories.

to explain the regime-dependence mechanistically, and discuss what the findings imply for practitioners.

11.1 Summary of empirical findings

Across the four datasets and three architecture families evaluated, SAGE-Lion exhibits three distinct behavioural regimes:

Regime 1: small dataset, well-conditioned task (CIFAR-10).

SAGE-Lion is statistically indistinguishable from Lion and AdamW on best test accuracy. The mechanism produces a modest reduction in the generalization gap (less than 1pp) and faster convergence to a target test accuracy, but adds approximately 19% wall-clock overhead and exhibits noisier late-epoch behaviour under the fixed-learning-rate protocol. The learning-rate sensitivity analysis in Section 6 establishes that the central-regime behaviour is well-approximated by a 3× rescaling of Lion's learning rate, suggesting that on benchmarks where overfitting is not the bottleneck, sign-agreement gating reduces to an alternative parameterization of Lion.

Regime 2: heavily overfitting convolutional training (CIFAR-100 with ResNet).

SAGE-Lion produces statistically significant gains over Lion of 1.6–2.1pp on test accuracy (paired t-test, $p < 0.05$), with a 5–6pp reduction in the train–test generalization gap and an approximately 15% reduction in test loss. The pattern of effects—a ∼5pp drop in training accuracy accompanied by a gain on test accuracy—is the signature of an implicit regularizer rather than a better optimizer. The learning-rate rescaling hypothesis is *insufficient* to explain these gains: the training-accuracy reduction would not appear under simple step-size shrinkage, because Lion's uniform-magnitude step lowers both training and test accuracy together, while SAGE-Lion's per-parameter damping selectively shrinks updates where gradient signs disagree.

Note that this regularization signature is specific to the fixed-learning-rate protocol; Section 7.7 demonstrates that under warmup-plus-cosine scheduling the signature

is substantially attenuated, though SAGE-Lion continues to outperform Lion by a large margin (+7.3pp) through improved schedule stability.

Regime 3: Vision Transformer training with cosine decay (TinyImageNet and ImageNet-100).

SAGE-Lion is again statistically indistinguishable from Lion on test accuracy, with both substantially outperforming AdamW (by 2pp on TinyImageNet and 24pp on ImageNet-100 at the 60-epoch budget). The wall-clock overhead falls from 19% on ResNet to under 2% on ViT-B/16, a trend driven by the growth of the forward/backward pass relative to optimizer-state operations. The regularization signature observed on CIFAR-100 does not appear in this regime: training accuracy reaches near-perfect for all three optimizers, generalization gaps are large for all, and the cosine schedule's late-epoch decay damps the effective step size below the range where the gating mechanism is active.

11.2 Why the mechanism activates only in Regime 2

The most informative finding is the asymmetry between Regimes 2 and 3: the same gating mechanism produces a clear regularization effect on ResNet+CIFAR-100 but no measurable effect on ViT+TinyImageNet or ViT+ImageNet-100. Two structural differences between the regimes account for this.

The first is the relative capacity of the model versus the dataset. On CIFAR-100, ResNet-18 reaches 96%–96.5% training accuracy under AdamW or Lion at the end of 60 fixed-learning-rate epochs, while SAGE-Lion plateaus near 91%. This is the regime where the loss surface in the late training phase consists of small, often-oscillating gradients on parameters that the model has nearly fit; the sign-agreement signal $\bar{a}_t$ becomes informative because it identifies parameters whose signal-to-noise ratio is low. On ViT-Tiny and ViT-B/16 under the cosine schedule, by contrast, the effective learning rate $\eta \cdot \kappa_t$ is being driven toward zero by the schedule's decay regardless of κ_t, so the gating signal has nothing left to suppress. Sign-agreement is, in effect, an adaptive late-training damping mechanism that is preempted by an explicit schedule.

The second is the structure of the loss surface itself. ViT architectures are known to exhibit a different optimization landscape than ResNets, with optimal weights residing in regions where the Hessian has substantially different spectral properties [34]. The notion that gradient signs encode useful information about parameter quality is grounded in the ResNet-style training dynamics that originally motivated sign-based optimizers; whether the same notion translates to ViT optimization is an open question that our experiments do not directly answer. What we can say is that on the specific ViT settings we evaluated, the gating mechanism produced no measurable accuracy difference relative to plain Lion.

11.3 Comparison to a learning-rate rescaling of Lion

The hypothesis test in Section 6 establishes that on CIFAR-10, SAGE-Lion at learning rate η matches Lion at learning rate $\eta/3$ within seed-level noise across the central regime of the grid, with the only systematic deviation appearing at the upper end of

the learning-rate range (where SAGE-Lion degrades less gracefully than the rescaling would predict). A practitioner reading this result could reasonably ask whether SAGE-Lion offers anything beyond a per-parameter version of running Lion at a smaller learning rate.

The answer based on our experiments is yes, in two specific senses. First, on overfitting-heavy ResNet+CIFAR-100, no learning rate for plain Lion that we tested reproduces the train-accuracy reduction that drives SAGE-Lion's generalization advantage. The mechanism is selectively damping per-parameter updates where signs disagree, not uniformly reducing step magnitude. Second, the high-learning-rate robustness observed in the sensitivity sweep (Section 6, $\eta = 1 \times 10^{-3}$) indicates that SAGE-Lion's effective step size adapts automatically when the nominal learning rate is set too aggressively, providing a modest amount of implicit safety against learning-rate mistuning. In the central regime, sign-agreement gating and learning-rate rescaling are observationally close to interchangeable; outside that regime they are not.

11.4 Practical guidance

The experimental picture supports the following practical guidance for practitioners considering SAGE-Lion over Lion or AdamW.

When sign-agreement gating is likely to help.

Training a moderately-sized convolutional model on a many-class image classification problem under a fixed or weak learning-rate schedule, where overfitting onset is observed during training, and where the practitioner is willing to accept 15–20% wall-clock overhead in return for 1–2pp test-accuracy improvement and a better-calibrated test loss.

When sign-agreement gating is likely to be neutral.

Training a Vision Transformer from scratch with a standard warmup-plus-cosine schedule, where SAGE-Lion and Lion produce comparable accuracy at comparable wall-clock cost. At ViT-B/16 scale, the cost difference falls below 1%, so SAGE-Lion is a low-risk substitute for Lion even when no gain is expected; the practitioner can default to whichever is more convenient.

When SAGE-Lion is likely to be inappropriate.

Single-step or very-short-horizon training, where the sign-agreement EMA does not have enough iterations to stabilise; and settings where Lion is already operating at the high end of its learning-rate range, where SAGE-Lion's implicit step damping may be redundant with explicit gradient clipping.

11.5 Comparison to AdamW

A separate question is when either Lion-family optimizer should be preferred to AdamW. The empirical answer from our experiments is straightforward: on ViT architectures trained from scratch at limited budgets, both Lion-family optimizers reach substantially higher test accuracy than AdamW at comparable wall-clock cost

(Sections 8, 9). The gap is largest at short schedules—approximately 24pp on ViT-B/16/ImageNet-100 at 60 epochs—and is expected to shrink at longer schedules, since AdamW's slower convergence on ViTs is a well-documented finding [3]. On ResNet+CIFAR training, SAGE-Lion and AdamW are within 1pp of each other in our experiments, with SAGE-Lion offering a smaller generalization gap at the cost of approximately 19% wall-clock overhead. In production settings where AdamW is the default, a Lion-family optimizer is the recommended substitute for Vision Transformer training; for convolutional training the choice is less clear-cut and depends on whether overfitting is the bottleneck.

12 Limitations

The experiments in this work establish a regime-dependent picture of sign-agreement gating, but several limitations constrain the generality of the conclusions.

Single-seed reporting on ImageNet-100.

The ViT-B/16 experiment in Section 9 uses one random seed. Each 60-epoch run on a single NVIDIA RTX 6000 Pro takes approximately 90 minutes per optimizer, and the three-optimizer comparison consumes roughly 4.5 hours; running three seeds would require 13–14 hours of compute, which we did not pursue. The 0.32pp difference between SAGE-Lion and Lion on this benchmark is comparable in magnitude to typical seed-to-seed variance reported on ImageNet-class benchmarks at similar schedules [5], so we treat the two optimizers as indistinguishable here, but a multi-seed replication would be needed to make a formal statistical claim. The CIFAR and TinyImageNet experiments all use three seeds (42, 43, 44) and report paired t-tests.

Limited training budget on ImageNet-100.

The 60-epoch ImageNet-100 budget is substantially shorter than the 300-epoch DeiT recipe used to train ViT-B/16 to its published ImageNet-1K accuracy [5]. At this shorter schedule, AdamW's slower convergence on ViT training amplifies the measured gap between AdamW and the Lion-family optimizers (Section 9). The result should be read as a comparison of optimizer behaviour under a limited compute budget rather than as evidence that AdamW is fundamentally worse than Lion-family methods at full convergence. Whether the relative ordering observed here persists at 300-epoch budgets is a question we did not investigate.

No learning-rate scheduler on CIFAR experiments.

The CIFAR-10 and CIFAR-100 experiments use a fixed learning rate throughout training (Section 4). This choice was deliberate—it isolates optimizer dynamics from scheduler–optimizer interactions—but it leaves open the question of how SAGE-Lion behaves on CIFAR with a standard warmup-plus-cosine schedule. Section 7.7 addresses this directly: a warmup-plus-cosine schedule applied to ResNet-18/CIFAR-100 confirms that the regularization signature is substantially attenuated under scheduled training, consistent with the theoretical prediction, while SAGE-Lion retains a large accuracy advantage over Lion through schedule-stability effects.

Architecture coverage.

The architecture coverage spans CNNs (ResNet-18, ResNet-34) and Vision Transformers (ViT-Tiny, ViT-B/16). We have not evaluated SAGE-Lion on the architectures where Lion was originally reported to produce its largest gains over AdamW: language models trained from scratch [3], contrastive pretraining objectives [35], or longer-horizon vision models such as ConvNeXt [36] or Swin Transformers [37]. The mechanism is architecture-agnostic in principle, but its empirical behaviour on these settings is unknown.

Hyperparameter sensitivity.

SAGE-Lion introduces two hyperparameters beyond Lion: the sign-agreement smoothing coefficient β_s and the confidence floor ϕ. To assess sensitivity to these values, we conducted a 3×3 factorial sweep over $\beta_s \in \{0.7, 0.8, 0.9\}$ and $\phi \in \{0.05, 0.10, 0.20\}$ on ResNet-18/CIFAR-100 with three random seeds per configuration (27 runs total); full results are reported in Section 10. The range of mean best test accuracy across all nine configurations is 0.45pp—smaller than the seed-to-seed standard deviation of the weakest individual configuration. A one-way ANOVA finds no statistically significant main effect of either β_s or ϕ on best test accuracy. The paper default ($\beta_s = 0.9$, $\phi = 0.10$) achieves the lowest seed-to-seed variance of all nine configurations (±0.03pp), confirming it as the most reproducible choice. The generalization gap at the best-test epoch is consistent across the full grid (24.3–24.8pp), indicating that the implicit regularization effect reported in Section 7 is a property of the gating mechanism and not an artefact of a specific parameter choice.

Optimizer-state memory.

As discussed in Section 3, the straightforward implementation of SAGE-Lion stores two additional buffers per parameter (previous gradient sign and EMA of sign agreement), bringing total optimizer state to 3× the parameter count—larger than AdamW's 2×. The previous-sign buffer is a sign tensor and can be bit-packed at 2 bits per parameter to recover most of the memory, and the EMA buffer is robust to reduced precision, but we have not implemented these optimizations. The reported overhead measurements (Sections 5, 8, 9) reflect full-precision buffers. A memory-optimised SAGE-Lion may have a different cost profile, particularly at scales where optimizer-state memory is the binding constraint.

Absence of theoretical convergence analysis.

This work is empirical. We do not provide a convergence proof for SAGE-Lion or an analysis of its fixed points. The gating multiplier $\kappa_t \in [\phi, 1]$ is bounded below by $\phi > 0$, so the update direction never vanishes, but we have not formally established that this is sufficient to inherit Lion's convergence properties [3]. A theoretical treatment is left to future work.

13 Conclusion

We have introduced SAGE-Lion, a Lion variant that gates the sign-based update by an exponential moving average of the per-parameter agreement between consecutive gradient signs. The modification adds two scalar buffers per parameter and a small number of elementwise operations per step, and preserves Lion's single-momentum design. The empirical evaluation across five settings spanning two architecture families and four image classification datasets supports three claims.

First, on overfitting-heavy convolutional training—ResNet-18 and ResNet-34 on CIFAR-100—SAGE-Lion improves test accuracy over Lion by 1.6–2.1 percentage points (paired t-test, $p < 0.05$), reduces the train–test generalization gap by 5–6pp, and lowers test loss by approximately 15%. The pattern of effects, particularly the $\sim$5pp reduction in training accuracy that accompanies the test gains, is consistent with an implicit regularization effect rather than with a learning-rate rescaling, and is not reproduced by adjusting Lion's learning rate alone.

Second, on Vision Transformer training with warmup-plus-cosine scheduling—ViT-Tiny on TinyImageNet and ViT-B/16 on ImageNet-100—SAGE-Lion is statistically indistinguishable from Lion on test accuracy, while both substantially outperform AdamW. The gap between SAGE-Lion and AdamW reaches +24.18pp on ViT-B/16 at the 60-epoch budget, with 89.34% test top-5 accuracy. The regularization signature observed on ResNet+CIFAR-100 does not appear under this protocol, which we attribute to the cosine schedule preempting the late-training regime where sign-agreement gating is informative.

A supplementary experiment in Section 7.7 confirms that applying warmup-plus-cosine scheduling to the ResNet-18/CIFAR-100 setting substantially attenuates the regularization signature while preserving a large accuracy advantage over Lion (+7.3pp), suggesting that the gating mechanism provides schedule robustness even when its implicit-regularization role is preempted.

Third, the per-epoch wall-clock overhead of SAGE-Lion over AdamW decreases monotonically with model size, from 18.6% on ResNet-18/CIFAR-10 to 0.9% on ViT-B/16/ImageNet-100. The gating operations contribute a fixed elementwise cost per parameter, but the per-step training cost grows superlinearly with model size; at production-scale transformers, the algorithmic overhead is essentially invisible.

A learning-rate sensitivity analysis on CIFAR-10 shows that, in the central regime of the learning-rate grid, SAGE-Lion at learning rate η behaves approximately like Lion at $\eta/3$, with the only systematic deviation appearing at the high-learning-rate end where SAGE-Lion degrades more gracefully than the rescaling would predict. This places a useful constraint on what the mechanism does: in well-conditioned settings it acts as a per-parameter learning-rate rescaling of Lion, with measurable secondary effects (regularization on overfitting-heavy tasks; robustness to too-high nominal learning rate) that the remaining experiments quantify.

The practical recommendation is that SAGE-Lion is a low-risk drop-in substitute for Lion on Vision Transformer training, where it provides comparable accuracy at near-identical wall-clock cost, and a worthwhile substitute on heavily overfitting convolutional training, where it provides statistically significant accuracy gains at moderate

wall-clock overhead. On well-conditioned benchmarks where overfitting is not the bottleneck, the mechanism is approximately neutral relative to Lion at a rescaled learning rate. Several extensions are immediate: multi-seed replication on ImageNet-100 and ImageNet-1K, evaluation on language modeling and contrastive pretraining where Lion's largest gains over AdamW have been reported, and a theoretical analysis of convergence behaviour under the bounded confidence multiplier. We leave these to future work.

Declaration of competing interest. The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Funding. This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Acknowledgements. The authors thank the High Performance Computing facility at Motilal Nehru National Institute of Technology Allahabad for providing the GPU infrastructure used for all experiments. The authors also thank the maintainers of the open-source libraries that this work depends on, in particular the PyTorch [32] and `timm` [28] projects, and the authors of the CIFAR-10/100 [9], TinyImageNet [11], and ImageNet-100 [12, 13] benchmarks for making their datasets publicly available to the research community.

Data availability. All datasets used in this study are publicly available. CIFAR-10 and CIFAR-100 [9] are distributed by the University of Toronto at https://www.cs.toronto.edu/~kriz/cifar.html. TinyImageNet [11] is distributed by the Stanford CS231N course at https://www.kaggle.com/c/tiny-imagenet. ImageNet-100 is the 100-class subset of ImageNet-1K defined by Tian et al. [13]; class lists and download instructions are available at https://www.image-net.org/. The source code, pre-trained model weights, and full training logs for SAGE-Lion will be made publicly available upon acceptance of this manuscript.

Author contributions. A.J. designed and implemented the SAGE-Lion optimizer, conducted all experiments, performed the data analysis, and wrote the manuscript. R.T. supervised the research, provided guidance on the experimental design and statistical analysis, and reviewed the manuscript. All authors read and approved the final manuscript.

Ethics declarations. Ethical approval and consent to participate. Not applicable. This study did not involve any human participants, human data, or animals.

References

[1] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014). Published at ICLR 2015

[2] Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: International Conference on Learning Representations (ICLR) (2019)

[3] Chen, X., Liang, C., Huang, D., Real, E., Wang, K., Liu, Y., Pham, H., Dong, X., Luong, T., Hsieh, C.-J., Lu, Y., Le, Q.V.: Symbolic discovery of optimization algorithms. In: Advances in Neural Information Processing Systems (NeurIPS) (2023)

[4] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: International Conference on Learning Representations (ICLR) (2021)

[5] Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers & distillation through attention. In: International Conference on Machine Learning (ICML) (2021)

[6] Riedmiller, M., Braun, H.: A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In: IEEE International Conference on Neural Networks, pp. 586–591 (1993)

[7] Bernstein, J., Wang, Y.-X., Azizzadenesheli, K., Anandkumar, A.: signSGD: Compressed optimisation for non-convex problems. In: International Conference on Machine Learning (ICML) (2018)

[8] Zhuang, J., Tang, T., Ding, Y., Tatikonda, S.C., Dvornek, N., Papademetris, X., Duncan, J.S.: AdaBelief optimizer: Adapting stepsizes by the belief in observed gradients. In: Advances in Neural Information Processing Systems (NeurIPS) (2020)

[9] Krizhevsky, A.: Learning multiple layers of features from tiny images. Technical report, University of Toronto (2009)

[10] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778 (2016)

[11] Le, Y., Yang, X.: Tiny imagenet visual recognition challenge. CS 231N Report, Stanford University (2015)

[12] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L.: ImageNet: A large-scale hierarchical image database. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 248–255 (2009)

[13] Tian, Y., Krishnan, D., Isola, P.: Contrastive multiview coding. In: European Conference on Computer Vision (ECCV) (2020)

[14] Shazeer, N., Stern, M.: Adafactor: Adaptive learning rates with sublinear memory cost. In: International Conference on Machine Learning (ICML) (2018)

[15] Anil, R., Gupta, V., Koren, T., Singer, Y.: Memory efficient adaptive optimization. In: Advances in Neural Information Processing Systems (NeurIPS) (2019)

[16] Balles, L., Hennig, P.: Dissecting Adam: The sign, magnitude and variance of stochastic gradients. In: International Conference on Machine Learning (ICML) (2018)

[17] Liu, H., Li, Z., Hall, D., Liang, P., Ma, T.: Sophia: A scalable stochastic second-order optimizer for language model pre-training. In: International Conference on Learning Representations (ICLR) (2024)

[18] Yao, Z., Gholami, A., Shen, S., Mustafa, M., Keutzer, K., Mahoney, M.W.: ADAHESSIAN: An adaptive second order optimizer for machine learning. In: AAAI Conference on Artificial Intelligence (2021)

[19] Igel, C., Hüsken, M.: Empirical evaluation of the improved Rprop learning algorithms. Neurocomputing **50**, 105–123 (2003)

[20] Cubuk, E.D., Zoph, B., Shlens, J., Le, Q.V.: RandAugment: Practical automated data augmentation with a reduced search space. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW) (2020)

[21] Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. In: International Conference on Learning Representations (ICLR) (2018)

[22] Yun, S., Han, D., Oh, S.J., Chun, S., Choe, J., Yoo, Y.: CutMix: Regularization strategy to train strong classifiers with localizable features. In: IEEE/CVF International Conference on Computer Vision (ICCV) (2019)

[23] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)

[24] Neyshabur, B., Tomioka, R., Srebro, N.: In search of the real inductive bias: On the role of implicit regularization in deep learning. In: ICLR Workshop (2015)

[25] Zhang, C., Bengio, S., Hardt, M., Recht, B., Vinyals, O.: Understanding deep learning requires rethinking generalization. In: International Conference on Learning Representations (ICLR) (2017)

[26] Zhang, M.R., Lucas, J., Hinton, G.E., Ba, J.: Lookahead optimizer: k steps forward, 1 step back. In: Advances in Neural Information Processing Systems (NeurIPS) (2019)

[27] Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., Wilson, A.G.: Averaging weights leads to wider optima and better generalization. In: Conference on Uncertainty in Artificial Intelligence (UAI) (2018)

[28] Wightman, R.: PyTorch Image Models. GitHub (2019). https://doi.org/10.5281/zenodo.4414861

[29] Loshchilov, I., Hutter, F.: SGDR: Stochastic gradient descent with warm restarts. In: International Conference on Learning Representations (ICLR) (2017)

[30] Kalamkar, D., Mudigere, D., Mellempudi, N., Das, D., Banerjee, K., Avancha, S., Vooturi, D.T., Jammalamadaka, N., Huang, J., Yuen, H., Yang, J., Park, J., Heinecke, A., Georganas, E., Srinivasan, S., Kundu, A., Smelyanskiy, M., Kaul, B., Dubey, P.: A study of BFLOAT16 for deep learning training. arXiv preprint arXiv:1905.12322 (2019)

[31] Zhong, Z., Zheng, L., Kang, G., Li, S., Yang, Y.: Random erasing data augmentation. In: AAAI Conference on Artificial Intelligence (2020)

[32] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: PyTorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems (NeurIPS) (2019)

[33] Huang, G., Sun, Y., Liu, Z., Sedra, D., Weinberger, K.Q.: Deep networks with stochastic depth. In: European Conference on Computer Vision (ECCV) (2016)

[34] Park, N., Kim, S.: How do vision transformers work? In: International Conference on Learning Representations (ICLR) (2022)

[35] Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., Sutskever, I.: Learning transferable visual models from natural language supervision. In: International Conference on Machine Learning (ICML) (2021)

[36] Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., Xie, S.: A ConvNet for the 2020s. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2022)

[37] Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin Transformer: Hierarchical vision transformer using shifted windows. In: IEEE/CVF International Conference on Computer Vision (ICCV) (2021)