A Scalable Measure of Loss Landscape Curvature for Analyzing the Training Dynamics of LLMs

Dayal Singh Kalra[1,2*] **Jean-Christophe Gagnon-Audet**[1] **Andrey Gromov**[1]

Ishita Mediratta[1] **Kelvin Niu**[1] **Alexander H Miller**[1] **Michael Shvartsman**[1]

[1]Meta Superintelligence Labs [2]University of Maryland, College Park

Abstract

Understanding the curvature evolution of the loss landscape is fundamental to analyzing the training dynamics of neural networks. The most commonly studied measure, Hessian sharpness ($\lambda^H_{\max}$) —the largest eigenvalue of the loss Hessian —determines local training stability and interacts with the learning rate throughout training. Despite its significance in analyzing training dynamics, direct measurement of Hessian sharpness remains prohibitive for Large Language Models (LLMs) due to high computational cost. We analyze *critical sharpness* (λ_c), a computationally efficient measure requiring fewer than 10 forward passes given the update direction $\Delta\boldsymbol{\theta}$. Critically, this measure captures well-documented Hessian sharpness phenomena, including progressive sharpening and Edge of Stability. Using this measure, we demonstrate progressive sharpening throughout pre-training and mid-training of OLMo-2 models up to 7B parameters. We further introduce *relative critical sharpness* ($\lambda_c^{1\to 2}$), which quantifies the curvature of one loss landscape while optimizing another, to analyze the transition from pre-training to fine-tuning and guide data mixing strategies. Critical sharpness provides practitioners with a practical tool for diagnosing curvature dynamics and informing data composition choices at scale. More broadly, our work shows that scalable curvature measures can provide actionable insights for large-scale training.

1 Introduction

Understanding the evolution of the loss landscape $L(\boldsymbol{\theta})$ over the high-dimensional parameter space $\boldsymbol{\theta}$ is fundamental to analyzing the training dynamics of neural networks. The loss landscape describes how the objective function changes with model parameters, and its geometry directly influences optimization, generalization, and stability throughout training [19, 27]. Intuitively, gradient-based optimization methods can swiftly navigate to minima in smooth landscapes, whereas in rough landscapes they can get trapped, hindering convergence or potentially converging to suboptimal solutions [33].

The local geometry of the loss landscape is commonly examined through the eigenvalues and eigenvectors of its Hessian matrix $H(\boldsymbol{\theta}) := \nabla^2_{\theta} L(\boldsymbol{\theta})$ [51, 32, 9]. These eigenvalues provide insights into the local curvature of the loss landscape —large eigenvalues correspond to sharp, steep directions, while small eigenvalues indicate flat, smooth regions. The largest eigenvalue of the Hessian, which is often referred to as *Hessian sharpness* $\lambda^H_{\max}$, quantifies the worst-case curvature of the landscape and is a fundamental metric in optimization. Its reciprocal, *flatness* $1/\lambda^H_{\max}$, is a complementary measure used to describe the curvature. In neural network optimization, Hessian sharpness relates to the local

*Work done during an internship at Meta.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

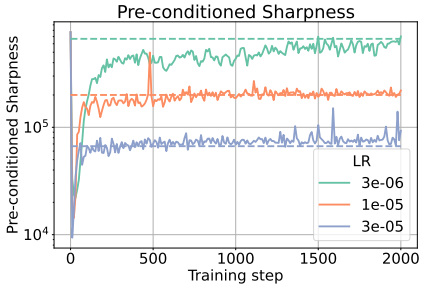


Figure 1: Pre-conditioned Hessian sharpness ($\lambda_{\max}^{PH}$) exhibits progressive sharpening and Edge of Stability (EoS) under constant learning rate. Pre-conditioned sharpness is normalized by the factor $(1+\beta_1)/(1-\beta_1)$ so that it oscillates around the threshold $2/\eta$ (marked by the dashed lines) at EoS.

training stability [51, 32]. For vanilla Gradient Descent (GD), if the learning rate exceeds the threshold $\sim 2/\lambda_{\max}^{H}$, loss increases as training 'catapults' out of the local basin and eventually converges to a flatter region where the stability conditions are satisfied [32, 26]. This observation generalizes to more complex optimization problems, including mini-batch settings and adaptive optimizers, albeit governed by different notions of sharpness [51, 1, 4, 10, 27]. For adaptive optimizers (e.g., Adam), the relevant measure is the pre-conditioned sharpness $\lambda_{\max}^{PH}$ [10], which we formalize in Section 2.

As illustrated in Figure 1, Hessian sharpness exhibits several robust trends throughout neural network training, particularly with constant learning rates. Training typically begins with an early reduction in Hessian sharpness [26, 28] followed by a continuous increase until it reaches the stability threshold [25, 9]. Once this threshold is reached, the training stabilizes through a self-stabilization mechanism [12, 8]—instead of diverging as classical optimization would predict, Hessian sharpness begins to oscillate around this critical value. The continual increase in Hessian sharpness is referred to as *progressive sharpening* while the subsequent oscillations around the critical threshold are termed the *Edge of Stability* (EoS) [9]. For the more realistic setting of learning rate schedules involving warmup and decay, sharpness closely follows the learning rate schedule [19, 27, 9]. Due to its particularly close relationship with learning rate, sharpness can be used as a diagnostic tool for identifying training instabilities.

Beyond training dynamics, sharpness of the final solutions is linked to generalization properties, motivated by the intuition that flatter minima generalize better [22]. However, this relationship has been called into question [29], as empirical analyses show mixed results —while flatter solutions generalize better in some settings, they can hurt performance in others. Sharpness also influences the early phase of training, with flatter initializations typically achieving better performance [15, 27].

Despite its value for examining model initialization, training dynamics, and generalization properties, analyzing sharpness at scale is challenging. Computing Hessian sharpness relies on iterative eigenvalue solvers (e.g., Power iteration, Lanczos), which require repeated Hessian-vector products (HVPs). However, HVP computation via automatic differentiation is often incompatible with modern training efficiency tools; kernels like Flash Attention [14] typically lack second-derivative implementations required for double backpropagation. Furthermore, iterative solvers can require hundreds of iterations with potential convergence failures. As a result, most existing studies are restricted to small-scale experiments (typically $\sim$ 10M parameters), leaving open questions about how sharpness evolves in Large Language Models (LLMs) at scale, and how it relates to optimization and downstream performance. In this work, we address this challenge by analyzing *critical sharpness*, a computationally efficient measure for studying curvature dynamics in LLM training, and extending it to guide practical data-mixing decisions.

Contributions. We make the following contributions:

- We analyze critical sharpness, defined as $\lambda_c = 2/\eta_c$, where η_c is the *critical learning rate*—the smallest learning rate that causes the training loss to increase in the next training step. To estimate the critical learning rate η_c, we perform an efficient line search along the update direction $\Delta\boldsymbol{\theta}$. This procedure only requires forward passes, making it fully compatible with modern large-scale distributed training infrastructure, while avoiding the convergence issues of iterative eigenvalue

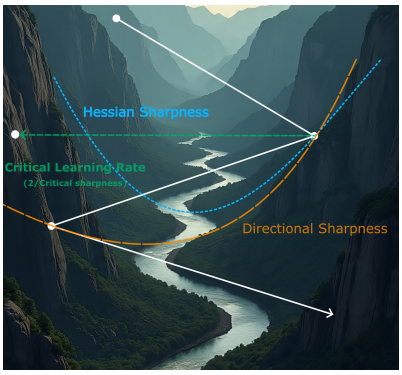


Figure 2: *Comparison of different sharpness measures on an illustrative landscape featuring a sharp valley direction and a flat river direction* [50]. Hessian sharpness quantifies the curvature along the sharpest direction of the landscape (the valley). Directional sharpness measures the quadratic curvature along $\Delta\boldsymbol{\theta}$, while critical sharpness, its empirical counterpart, quantifies how far one can step along $\Delta\boldsymbol{\theta}$ before the loss increases.

solvers. In practice, we find that this procedure reliably estimates critical sharpness in only 5-6 forward passes[2], making it 200–500$\times$ faster to compute than Hessian sharpness.

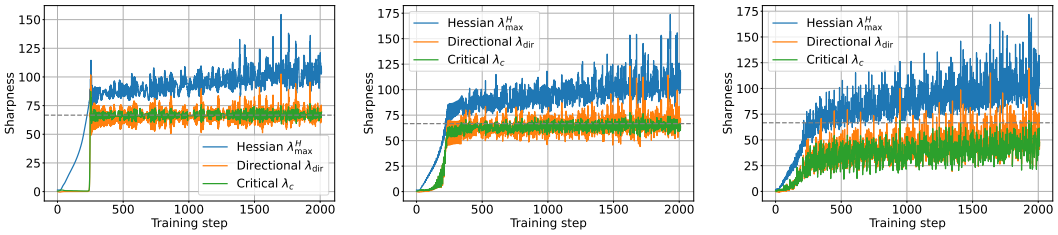
- We then examine the relationship between critical sharpness and Hessian sharpness. Under the quadratic loss approximation, we show that critical sharpness can be written as a weighted sum of Hessian eigenvalues, and the two measures coincide when the gradient aligns with the largest eigenvector of the Hessian. We also generalize this result to adaptive optimizers.
- We demonstrate that critical sharpness exhibits the same well-documented training phenomena as Hessian sharpness, *progressive sharpening*, and the *Edge of Stability* [9], while being tractable at scale. Leveraging OLMo-2 models up to 7B parameters [49] across pre-training and mid-training, we show that progressive sharpening persists across model scales throughout training.
- We introduce a relative measure of critical sharpness $\lambda_c^{1\to 2}$ to quantify the curvature of one loss relative to another, with application to analyzing the pre-training loss landscape during mid-training/fine-tuning. By varying the mix ratio of pre-training and fine-tuning (e.g., math), we show that increasing the proportion of pre-training data lowers relative critical sharpness, helping the model stay near the "pre-trained basin" during fine-tuning. We further show that downstream performance in OLMo-2 math fine-tuning depends on basin retention: GSM8K [7] benefits from leaving the pre-trained basin ($\eta > 2/\lambda_c^{1\to 2}$), while MMLU [21] is better preserved when the basin is retained ($\eta < 2/\lambda_c^{1\to 2}$). This enables us to identify candidate data mixtures that balance specialization and retention.

2 Characterizing Critical Sharpness: Theory and Empirics

In this section, we characterize the dynamics of critical sharpness both theoretically and empirically. We first establish the theoretical relationship between critical sharpness and Hessian sharpness under a quadratic loss approximation, then validate these insights empirically on MLPs trained on CIFAR-10 using SGD across different batch sizes (Figure 3).

Setup. Consider a model with trainable parameters $\boldsymbol{\theta} \in \mathbb{R}^n$, and let $L(\boldsymbol{\theta})$ denote the loss function with gradient $g(\boldsymbol{\theta}) := \nabla_\theta L(\boldsymbol{\theta})$ and Hessian $H(\boldsymbol{\theta}) := \nabla_\theta^2 L(\boldsymbol{\theta})$. Let $\{\lambda_i^H\}_{i=1}^n$ and $\{u_i\}_{i=1}^n$ denote the eigenvalues and eigenvectors of the Hessian, respectively. The largest eigenvalue $\lambda_{\max}^H = \max_i \lambda_i^H$, termed *Hessian sharpness*, quantifies the worst-case local curvature. For adaptive optimizers with pre-conditioner $P(\boldsymbol{\theta})$ (e.g., Adam), the training dynamics is governed by the pre-conditioned Hessian $P(\boldsymbol{\theta})^{-1/2} H(\boldsymbol{\theta}) P(\boldsymbol{\theta})^{-1/2}$, with eigenvalues $\{\lambda_i^{PH}\}_{i=1}^n$ and eigenvectors $\{v_i\}_{i=1}^n$. We omit explicit $\boldsymbol{\theta}$ dependence when clear from context.

[2]except for the first iteration, which depends on the initial guess η_0 for line search.



(a) Batch size = 50,000 (GD) (b) Batch size = 5000 (c) Batch size = 500

Figure 3: Comparison of different sharpness measures for MLPs trained on CIFAR-10 image classification task using SGD with learning rate η. Both critical and directional sharpness exhibit progressive sharpening and Edge of Stability, albeit some deviations from Hessian sharpness. The dashed line denotes the Edge of Stability threshold, given by $2/\eta$.

2.1 Critical Sharpness: A Scalable Measure of Curvature

Definition 2.1 (Critical learning rate η_c and Sharpness λ_c)**.** Given an update direction $\Delta\boldsymbol{\theta}$, we define *critical learning rate* as the smallest learning rate that causes loss to increase in the next training step:

$$\eta_c = \inf\{\eta > 0 \mid L(\boldsymbol{\theta} - \eta\Delta\boldsymbol{\theta}) > L(\boldsymbol{\theta})\}. \tag{1}$$

Correspondingly, we define *critical sharpness* as the scaled reciprocal of the critical learning rate $\lambda_c = 2/\eta_c$.

Geometric Interpretation. Critical sharpness provides an intuitive, optimizer-aware measure of the local curvature of the loss landscape. Geometrically, it quantifies the "natural length-scale" of the landscape by estimating how far one can move in the current update direction without leaving the local region. While Hessian sharpness quantifies the curvature along the steepest direction of the landscape, critical sharpness measures the curvature along the update direction $\Delta\boldsymbol{\theta}$. Figure 2 compares critical sharpness with Hessian sharpness on an illustrative landscape.

Efficient estimation of critical learning rate. To efficiently measure the critical learning rate, we build upon the line search method proposed by [27]. Given the update direction $\Delta\boldsymbol{\theta}$ from training, we compute the critical learning rate η_c using a two-phase line search procedure:

1. *Exponential Search:* Starting from an initial guess η_0, we iteratively double (halve) η until the loss $L(\boldsymbol{\theta} - \eta\Delta\boldsymbol{\theta})$ exceeds (or falls below) the current loss $L(\boldsymbol{\theta})$. This quickly identifies an interval $[\eta_{\text{lower}}, \eta_{\text{upper}}]$ containing η_c.
2. *Binary Search:* We then refine this interval using a binary search until the relative error $|1 - \eta_{\text{lower}}/\eta_{\text{upper}}|$ falls below a threshold ϵ. For $\epsilon = 1/2^k$, the binary search converges in k steps.

We approximate the critical learning rate using the mean of the final range $\eta_c \approx \frac{1}{2}(\eta_{\text{lower}} + \eta_{\text{upper}})$. We use the same mini-batch both to compute the update direction $\Delta\boldsymbol{\theta}$ and for all loss evaluations in the line search. In practice, the exponential search requires only 1–2 iterations per step (after an initial step whose cost depends on the starting guess η_0), since η_c tends to remain stable across consecutive training steps. For the binary search, we find that setting $\epsilon = 1/16$ (i.e., $k = 4$) yields a reliable estimate in approximately 5–6 forward passes total. We analyze the sensitivity of this choice in Section A.2, confirming that $k = 4$ introduces less than 2% relative error, which is an order of magnitude smaller than the step-to-step variation in critical sharpness. This two-step procedure relies solely on forward passes and is 200–500$\times$ faster than Hessian sharpness computation (see Section A.3) and is fully compatible with modern distributed training infrastructure, unlike Hessian-vector products, which are incompatible with memory-efficient kernels such as FlashAttention [14]. We provide the detailed algorithm and further discuss design choices in Section A.

2.2 The Relationship between Critical Sharpness and Hessian Sharpness

Before investigating the empirical behavior of critical sharpness, we establish its theoretical relationship with Hessian sharpness. This connection provides a principled foundation for understanding what critical sharpness captures. To this end, we consider the quadratic approximation

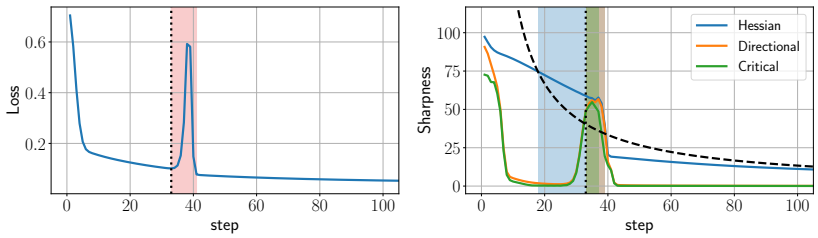


Figure 4: *Directional sharpness better tracks training instability than Hessian sharpness.* Loss and sharpness trajectories of an MLP trained on CIFAR-10 using GD with a linearly increasing learning rate. The dotted vertical line marks the onset of loss increase, and the shaded region in red marks the loss spike. The dashed line in the sharpness panel denotes the $2/\eta_t$ stability threshold, and the shaded regions mark steps where a measure exceeds it. Hessian sharpness exceeds $2/\eta_t$ at $t = 19$, yet the loss does not increase until $t = 33$, which coincides with directional sharpness crossing the $2/\eta_t$.

of the loss function along the update direction $\Delta\boldsymbol{\theta}$:

$$L(\boldsymbol{\theta} - \eta\Delta\boldsymbol{\theta}) \approx L(\boldsymbol{\theta}) - \eta\Delta\boldsymbol{\theta}^T g(\boldsymbol{\theta}) + \frac{1}{2}\eta^2\Delta\boldsymbol{\theta}^T H(\boldsymbol{\theta})\Delta\boldsymbol{\theta}.$$

Within this approximation, the loss will increase if the learning rate η exceeds $2/\lambda_{\text{dir}}$, where λ_{dir} is the *directional sharpness* [40, 43]:

Definition 2.2 (Directional Sharpness λ_{dir})**.** The directional sharpness along the direction $\Delta\boldsymbol{\theta}$ is:

$$\lambda_{\text{dir}} = \frac{\Delta\boldsymbol{\theta}^T H(\boldsymbol{\theta})\Delta\boldsymbol{\theta}}{\Delta\boldsymbol{\theta}^T g(\boldsymbol{\theta})}. \tag{2}$$

Directional sharpness λ_{dir} serves as an analytically tractable approximation to the empirically measured critical sharpness. Figure 3 shows that directional sharpness closely tracks critical sharpness throughout training. This alignment suggests that the local quadratic approximation well captures the complex dynamics in this simplistic setting. For Gradient Descent (GD), the directional sharpness can be expressed as a weighted sum of Hessian eigenvalues [43]:

Result 2.1 (Relationship between Directional and Hessian Sharpness for Gradient Descent)**.** For Gradient Descent (GD), the directional sharpness λ_{dir} can be expressed as a weighted sum of the Hessian eigenvalues $\{\lambda_i^H\}_{i=1}^n$, where the weights quantify the alignment of the gradient with Hessian eigendirections $\{u_i\}_{i=1}^n$:

$$\lambda_{\text{dir}} = \frac{\sum_{i=1}^n c_i^2\lambda_i^H}{\sum_{i=1}^n c_i^2} \leq \lambda_{\max}^H, \tag{3}$$

where $c_i = \boldsymbol{g}^T\boldsymbol{u}_i$ is the projection of the gradient onto the ith eigenvector.

It follows that if the gradient $\boldsymbol{g}$ is perfectly aligned with the largest eigenvector, i.e., $\boldsymbol{g} \propto \boldsymbol{u}_{\max}$, then directional sharpness coincides with Hessian sharpness. More generally, directional sharpness is bounded above by Hessian sharpness, and their gap depends on both gradient alignment and the Hessian spectrum. We now generalize this result to adaptive optimizers.

Result 2.2 (Relationship between Directional and Hessian Sharpness for Adaptive optimizers)**.** For adaptive optimizers with pre-conditioner P and update direction $P^{-1}\boldsymbol{g}$ (e.g., RMSProp), the directional sharpness λ_{dir} can be expressed as a weighted sum of the pre-conditioned Hessian eigenvalues $\{\lambda_i^{PH}\}_{i=1}^n$, where the weights quantify the alignment of the pre-conditioned gradient $P^{-1/2}\boldsymbol{g}$ with pre-conditioned Hessian eigendirections $\{\boldsymbol{v}_i\}_{i=1}^n$:

$$\lambda_{\text{dir}} = \frac{\sum_{i=1}^n c_i^2\lambda_i^{PH}}{\sum_{i=1}^n c_i^2} \leq \lambda_{\max}^{PH}, \tag{4}$$

where $c_i = P^{-1/2}\boldsymbol{g}^T\boldsymbol{v}_i$ is the projection of the pre-conditioned gradient $P^{-1/2}g$ onto the ith eigenvector $\boldsymbol{v}_i$ of the pre-conditioned Hessian. For momentum-based optimizers like Adam, this result serves as an approximation. We detail the derivations of the above results in Section D.1.

Together, these results establish a connection between directional and Hessian sharpness for both GD and adaptive optimizers. As critical sharpness coincides with directional sharpness under a local quadratic approximation, this connection in turn links critical sharpness to Hessian sharpness. In particular, for GD, all three measures coincide when the gradient aligns with the top Hessian eigendirection.

We examine the differences between the three sharpness measures in Figure 3. In the full-batch regime, Hessian sharpness progressively increases, eventually reaching the EoS threshold. In contrast, both directional sharpness and critical sharpness remain nearly constant during the early stages of training, followed by an abrupt increase to the EoS threshold. At smaller batch sizes, however, all three sharpness measures display a gradual increase from the onset of training. These observations are consistent with the gradient-alignment findings of [46]: under full-batch GD during the early phase, the gradient remains poorly aligned with the top eigendirection, while under SGD, stochastic noise causes the gradient to align with the top eigendirection early in training. Combined with Result 2.1, this explains the large gap between Hessian sharpness and critical/directional sharpness under full-batch GD. After crossing the EoS threshold, Hessian sharpness tends to oscillate above the threshold, whereas both critical sharpness and directional sharpness oscillate more closely around it. This is because the self-stabilization mechanism is triggered when the loss increases, which requires $\eta > 2/\lambda_{\text{dir}}$ rather than $\eta > 2/\lambda^H_{\max}$.

We emphasize that we do not propose critical sharpness as a proxy for Hessian sharpness. Rather, we introduce critical sharpness as a computationally efficient, optimizer-aware curvature measure. By definition, it tracks the learning-rate scale associated with one-step instability along the optimizer's update direction, without relying on a local quadratic approximation. Figure 4 illustrates the value of update-direction-aware curvature for diagnosing instability. Directional sharpness crosses the stability threshold at the onset of loss increase, whereas Hessian sharpness crosses it 14 steps earlier. Critical sharpness crosses at the onset by construction and is shown for reference.

In the following section, we extend our analysis to more realistic, large-scale training settings.

3 Critical Sharpness Dynamics at Scale

Modern large-scale models are typically trained using Adam with weight decay [36], which helps mitigate training instabilities [13]. To analyze sharpness dynamics in this context, we first analyze the stability threshold for common optimizers with weight decay:

Result 3.1 (Stability threshold for optimizers with weight decay)**.** For gradient descent and adaptive optimizers such as Adam, adding weight decay shifts the EoS threshold by a constant that depends on the decay strength γ:

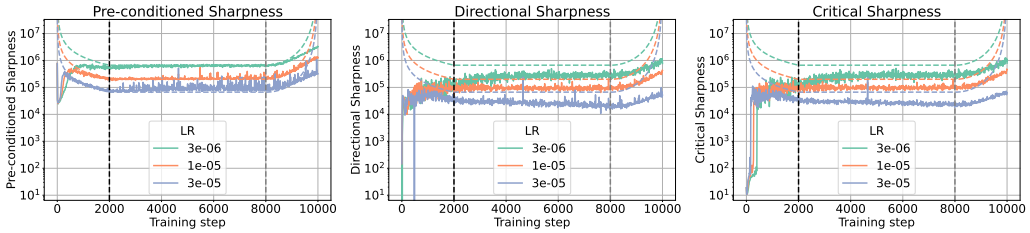
$$\lambda^H_{\max} = \frac{2}{\eta} - \gamma \qquad \text{(GD)}$$

$$\lambda^{PH}_{\max} = \left(\frac{2}{\eta} - \gamma\right)\left(\frac{1+\beta_1}{1-\beta_1}\right) \qquad \text{(Adam)} \tag{5}$$

where η is the learning rate, β_1 is Adam's momentum parameter, and γ is the weight decay strength. This result holds locally under a quadratic approximation and full-batch dynamics and, for Adam, it further assumes a slowly varying pre-conditioner. We provide the detailed proofs in Section D.2.

We now analyze GPT-style Transformers pre-trained for next-token prediction on the FineWebEdu dataset [41] using AdamW with Warmup-Stable-Decay (WSD) schedule [23]. Figure 5 compares the dynamics of critical and pre-conditioned sharpness for three different learning rates. The pre-conditioned sharpness $\lambda^{PH}_{\max}$ exhibits progressive sharpening while following the learning rate schedule throughout training —it is pushed down during the warmup phase, stays constant at the stability threshold during the stable phase, and increases again when the learning rate is decayed. Critical sharpness and directional sharpness follow similar trends throughout training —they exhibit clear progressive sharpening and EoS behavior, while oscillating below the EoS threshold during the stable phase and capturing the increase in pre-conditioned sharpness during the decay phase.

While progressive sharpening has been consistently documented at small scales, especially in full-batch settings, its manifestation and relevance at large scales, particularly in the context of online LLM training, remain largely unexplored. Having established that critical sharpness exhibits



(a) (b) (c)

Figure 5: *Critical sharpness tracks pre-conditioned sharpness trends throughout training.* Dynamics of pre-conditioned, directional, and critical sharpness during GPT pre-training on FineWebEdu with AdamW using a Warmup-Stable-Decay (WSD) schedule. Pre-conditioned sharpness is normalized by $(1+\beta_1)/(1-\beta_1)$ and shifted by γ so that all three measures share the same $2/\eta$ stability threshold (colored dashed lines). The two black vertical lines mark the end of the warmup and stable phases.

progressive sharpening and Edge of Stability phenomena, we are now in the position to investigate whether progressive sharpening persists in large-scale LLM training. To this end, we analyze OLMo-2 1B and 7B checkpoints [49]. OLMo-2 provides checkpoints throughout both pre-training and mid-training stages, enabling us to study sharpness dynamics at scale without the computational cost of training such models from scratch. Within each training stage, we measure critical sharpness at every checkpoint using the same fixed evaluation subset drawn from that stage's data distribution.

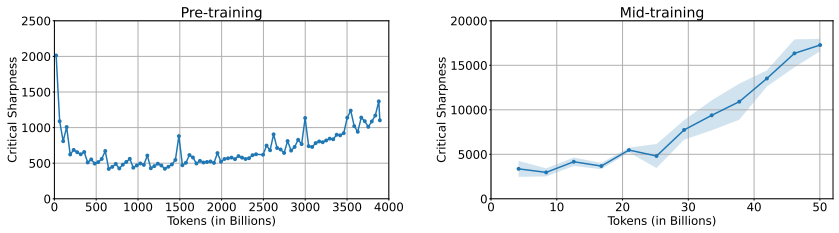
The OLMo-2 models are trained using a two-stage curriculum approach. During the pre-training stage ($> 90\%$ of compute), the model is trained for 4T tokens on the DCLM dataset [34], a diverse web-sourced corpus. Following pre-training, the model undergoes a mid-training stage, where it is trained for 50B tokens on a curated mix. This mix includes academic papers, Wikipedia articles, instruction-following data, StackExchange documents, and 47.2% of the original pre-training dataset (DCLM). During the pre-training stage, the model is trained using a learning rate schedule consisting of $2{,}000$ steps of warmup, followed by a cosine decay down to one-tenth of its peak value[3]. By comparison, during the mid-training stage, the learning rate is linearly decayed to zero. As the learning rate is continuously decreased throughout training after the warmup stage, the EoS threshold increases, and consequently we expect sharpness to progressively increase for the remainder of training. To confirm this, we examine how the critical sharpness evolves across pre-training and mid-training using the publicly available checkpoints (see Section B for the details). Figure 6 shows that critical sharpness rapidly decreases during early training, but then continually increases (progressive sharpening) throughout both the pre-training and mid-training stages. In Section B, we show that these observations are robust to the choice of evaluation batch: variation across independent batches is substantially smaller than the changes across checkpoints. We observe the same qualitative behavior for OLMo-2 1B (Figure 15), showing that this trend is consistent across both model scales.

The Edge of Stability boundary $2/\eta_t$ lies substantially above the measured critical sharpness for these OLMo-2 trajectories (Figure 14). This gap may reflect OLMo-2's conservative learning-rate schedule or the fact that our estimates use a fixed evaluation batch rather than the sequence of batches encountered during training. More fundamentally, large-scale training may not operate at the Edge of Stability. We leave the detailed analysis of this gap for future work.

4 How much Pre-training data is needed to avoid Catastrophic Forgetting?

To go beyond demonstrating pre-existing sharpness phenomena at scale, we next provide a new practical application of critical sharpness, focused on data mixing. We are motivated by the fact that during fine-tuning, neural networks are prone to *catastrophic forgetting*, where the model performance degrades on the pretraining dataset and benchmarks as the model adapts to the new task [38, 37, 39]. To mitigate this, several strategies have been proposed [16], with mixing samples from the pre-training data being the most effective [42]. This practice is reflected in large-scale training: for example, [49] uses a mid-training mix consisting of 47.2% pre-training data. Intuitively, we want to use as much new data as possible in fine-tuning, while using as little pre-training rehearsal data as possible to

[3]For OLMo-2 7B, cosine decay schedule is designed to reach $1/10$ of its peak value at 5T tokens, but is truncated at 4T.



(a) (b)

Figure 6: Critical sharpness of OLMo-2 7B throughout (a) pre-training and (b) mid-training. We evaluate checkpoints using a fixed subset of the corresponding dataset: DCLM for pre-training and the Dolmino mix for mid-training. We use a batch size of $1{,}024$ and a context length of $4{,}096$. The OLMo-2 training procedure runs three mid-training trajectories and combines their final checkpoints into a model soup. The mid-training curve shows their mean, with the shaded band denoting variation across them.

retain base capabilities. However, it remains unclear what fraction of pre-training data is sufficient to effectively prevent catastrophic forgetting. To this end, we leverage critical sharpness to systematically examine the effect of adding pre-training data to the training mix during OLMo mid-training and provide actionable guidance for selecting the pre-training data fraction without exhaustive grid search.

The goal of mid-training or fine-tuning is to improve the performance on specialized domains or tasks while preserving generic capabilities acquired during pre-training. Intuitively, this requires the model to adapt to the new task, while staying within the "pre-training basin" —a region of the parameter space where the pre-training loss remains low. Leaving this basin would be marked by an increase in the pre-training loss, and the critical learning rate quantifies exactly how far we can step before this occurs. To formalize this intuition, we define relative critical learning rate and sharpness, as follows:

Definition 4.1 (Relative Critical learning rate $\eta_c^{1\to 2}$ and Sharpness $\lambda_c^{1\to 2}$)**.** Consider a model with parameters $\boldsymbol{\theta}$, two loss functions $L_1(\boldsymbol{\theta})$ and $L_2(\boldsymbol{\theta})$, and an update direction $\Delta\boldsymbol{\theta}$ derived from L_2. The *relative critical learning rate* is the smallest learning rate for which taking a step in the direction $\Delta\boldsymbol{\theta}$ increases the loss $L_1(\boldsymbol{\theta})$:

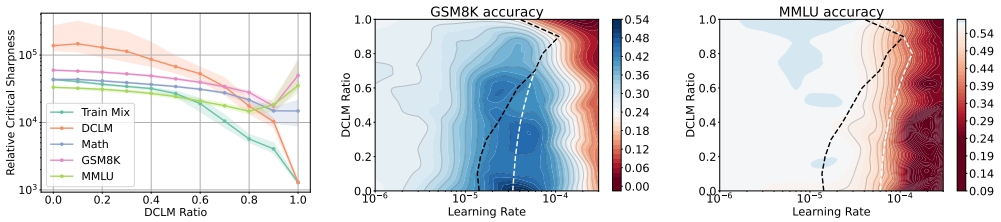
$$\eta_c^{1\to 2} = \inf\{\eta > 0 \mid L_1(\boldsymbol{\theta} - \eta\Delta\boldsymbol{\theta}) > L_1(\boldsymbol{\theta})\}. \tag{6}$$

The corresponding *relative critical sharpness* is $\lambda_c^{1\to 2} = 2/\eta_c^{1\to 2}$.

This definition is general and applies whenever optimization on one objective may affect performance on another. The two losses can correspond to different tasks (e.g., pre-training vs. fine-tuning), different loss functions (e.g., next-token prediction vs. reinforcement learning objectives), or different data characteristics (e.g., short vs. long context). In this work, we focus on the fine-tuning setting, where $L_1(\boldsymbol{\theta})$ is the pre-training loss on a general text corpus and $L_2(\boldsymbol{\theta})$ is the fine-tuning loss, which often is a mixture of pre-training and specialized data (e.g., math).

To assess the impact of pre-training data fraction in the training mix during fine-tuning, we consider the OLMo-2 7B pre-trained checkpoint as our starting point. We define $L_2(\boldsymbol{\theta})$ as the loss on a mixture composed of DCLM (pre-training data) and the math subset of Dolmino mix [49], and compute the update direction $\Delta\boldsymbol{\theta}$ from this mixture. We then examine how the relative critical sharpness varies with the DCLM fraction in this mixture[4]. Figure 7(a) shows how the relative critical sharpness varies with the DCLM ratio in the training mix for several evaluation tasks. When the mix contains mostly math data (low DCLM ratio), the sharpness for DCLM is an order of magnitude higher than for other tasks, indicating that the pre-training loss landscape is particularly sharp and sensitive to fine-tuning updates in this regime. As more pre-training data is added, the relative critical sharpness for most tasks decreases, suggesting that the landscapes align. On the other hand, when the DCLM ratio approaches one, the sharpness for downstream tasks such as Math, GSM8K, and MMLU increases, meaning that these tasks become the limiting factor for the maximum stable learning rate. Notably, the sharpness curves intersect in an intermediate DCLM-ratio region around 0.7, suggesting a favorable region in which no single task strongly limits the learning rate.

[4]When measuring the relative critical sharpness, we do not update the model parameters.



(a) (b) (c)

Figure 7: (a) Relative critical sharpness (Definition 4.1) for various evaluation tasks (L_1). The shaded region around the mean trends denotes the variation across batches. (b, c) GSM8K and MMLU accuracy as a function of pre-training (DCLM) mix ratio and learning rate. Red indicates a decrease in performance relative to the checkpoint, white indicates no change, and blue indicates an improvement. The black dashed line denotes the smallest critical learning rate at the pre-trained checkpoint across tasks from (a), and the white dashed line denotes the critical learning rate for the corresponding task.

In Figure 7(b, c), we evaluate the impact of the pre-training mix by training the pre-trained checkpoint for 1B tokens and measuring downstream accuracy on GSM8K and MMLU benchmarks. In these plots, red indicates a decrease in performance relative to the pre-trained checkpoint, white indicates no change, and blue indicates an improvement. We observe a natural trade-off: GSM8K accuracy is typically maximized outside the pre-training basin (i.e., with low DCLM ratio and high learning rate), but this comes at the expense of MMLU performance, which is best preserved within the pre-training basin. Therefore, if the sole objective is to improve on the fine-tuning task (math), then training primarily on it with a large learning rate is effective, though it may lead to forgetting on other tasks (MMLU). In contrast, if the goal is to improve on the fine-tuning task (math) while maintaining pre-training performance (MMLU), including pre-training data in the mix is beneficial, with a favorable region around a DCLM ratio of 0.6 and a learning rate of $3 \cdot 10^{-5}$. This is close to the candidate region around 0.7 indicated by the critical sharpness analysis. This result highlights the importance of balancing task-specific and general data in the training mix. Once the pre-trained loss L_1 increases ($\eta > 2/\lambda_c^{1\to 2}$), subsequent updates need not recover L_1 unless pre-training data is included in the mix. In Section C.4, we extend the relative critical sharpness analysis to the datamix used in OLMo-2 mid-training. The results are consistent: the curves intersect in a candidate region around a DCLM ratio of 0.6, close to the 47.2% ratio used in the original OLMo-2 training. We leave validation of this candidate region through downstream evaluation to future work.

5 Related Works

In neural network training, if the learning rate exceeds the threshold $\sim 2/\lambda_{\max}^H$, it causes the loss to increase in the next training step [51]. However, Kalra et al. [26] demonstrated that the empirical critical learning rate can be much higher than this theoretical threshold derived from convex analysis, with critical learning rate reaching up to $40/\lambda_{\max}^H$ when the sharpness naturally decreases during training. Our work extends theirs by showing that a similarly large critical learning rate can also arise when sharpness increases during training, due to contributions from other eigendirections (Figure 3). Beyond its role in characterizing training instabilities, Hessian sharpness also exhibits progressive sharpening during training. In particular, Cohen et al. [9] showed that sharpness increases when the learning rate is decayed, which we corroborate with additional analysis at large scales (Figures 5 and 6).

The works most closely related to ours are those by [27], [43], and [5]. Kalra et al. [27] used the critical learning rate to set the initial learning rate during warmup. By comparison, Roulet et al. [43] proposed setting the learning rate at 'edge' throughout training, i.e., $\eta_t = 2/\lambda_{\text{dir}}$, and show that it can outperform constant learning rate training, but not typical learning rate schedules, consisting of learning rate warmup followed by decay. Vaswani et al. [48] also study setting the learning rate as $\eta = 1/\lambda_{\text{dir}}$. Bu et al. [5] proposed a generalized Newton method that uses forward passes to estimate local curvature for automatic learning rate selection. In contrast, we use the critical learning rate to study the curvature dynamics of large-scale models and examine the effect of pre-training data on catastrophic forgetting. Dagréou et al. [11] benchmarked automatic-differentiation methods for

efficiently computing Hessian-vector products. Dremov et al. [17] analyzed the progression of LLM optimization from a river-valley structure toward the final basin during learning-rate cooldown.

Catastrophic forgetting is a long-standing problem in neural networks, first identified by [38], where model performance degrades on previously learned tasks as it adapts to the new task. Approaches to mitigate catastrophic forgetting can be categorized into (i) regularization methods [30, 2], (ii) ensembling and parameter isolation [44, 3], and (iii) rehearsal [42, 35]. Among these, rehearsal —mixing samples from the pre-training data —has become the most widely adopted strategy due to its simplicity and effectiveness. Recent works have demonstrated that catastrophic forgetting persists in LLM fine-tuning [37, 24, 45]. Particularly related to our work is that of [6], who show that if fine-tuning retains previously learned capabilities, the model remains in "most-case" and "worst-case" basins. Theoretical works have further elucidated these dynamics; Tahir et al. [47] argue that fine-tuning success relies on high feature overlap with the pre-trained model, while Graldi et al. [20] show that staying in the "lazy" training regime can reduce forgetting. In the light of these works, relative critical sharpness can be used to quantify the feature overlap between the pre-trained model and the fine-tuning dataset and the critical learning rate required to stay in the lazy regime.

6 Discussion and Conclusion

In this work, we analyzed critical sharpness, a computationally efficient measure for studying the training dynamics of LLMs. Our results demonstrate that critical sharpness reliably captures key Hessian sharpness phenomena such as progressive sharpening and Edge of Stability, requiring fewer than 10 forward passes, avoiding challenges associated with Hessian-based methods. Using this measure, we observed progressive sharpening throughout pre-training and mid-training in OLMo-2 models up to 7B parameters. We also introduced relative critical sharpness, which quantifies the curvature of one loss landscape along the update direction of another. Using this measure, we identified a candidate range of pre-training data fractions that balances specialization and retention during fine-tuning, helping practitioners evaluate data-composition choices without exhaustive ablations. Beyond fine-tuning, relative critical sharpness provides a general framework for analyzing changes in the loss landscape due to distribution shifts, changes in loss functions, or modifications to the training data mixture.

We believe critical sharpness can extend beyond the settings studied here. More broadly, our results demonstrate that scalable curvature measures can provide actionable insights for large-scale training, from understanding optimization dynamics to informing data composition decisions.

Limitations. Our large-scale analysis is limited to the OLMo-2 family up to 7B parameters. Although critical sharpness itself is cheap to compute, the baseline cost of working with large models remains substantial, which precluded analysis across additional models. Our data-mixing study is likewise limited to a single finetuning task, and 1B training tokens, leaving broader validation across tasks to future work. That said, we observe consistent behavior across all scales we study, from MLPs on CIFAR-10 to OLMo-2 1B and 7B, lending confidence to the generality of our findings.

Acknowledgments and Disclosure of Funding

We thank Darshil Doshi, Tianyu He, Maissam Barkeshli, Aaron Dafezio, Jessie Dodge, Sean McLeish, John Kirchenbauer, and Tom Goldstein for helpful discussions and feedback on the draft. DSK thanks the Simons Collaboration on Physics of Learning and Neural Computation (SFI-MPS-POL-00012574-09) for support.

References

[1] Atish Agarwala and Jeffrey Pennington. High dimensional analysis reveals conservative sharpening and a stochastic edge of stability, 2025.

[2] Hongjoon Ahn, Sungmin Cha, Donggyu Lee, and Taesup Moon. Uncertainty-based continual learning with adaptive regularization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[3] Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert Gate: Lifelong Learning with a Network of Experts . In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7120–7129, Los Alamitos, CA, USA, July 2017. IEEE Computer Society.

[4] Arseniy Andreyev and Pierfrancesco Beneventano. Edge of stochastic stability: Revisiting the edge of stability for sgd, 2025.

[5] Zhiqi Bu and Shiyun Xu. Gradient descent with generalized newton's method. In *The Thirteenth International Conference on Learning Representations*, 2025.

[6] Huanran Chen, Yinpeng Dong, Zeming Wei, Yao Huang, Yichi Zhang, Hang Su, and Jun Zhu. Understanding pre-training and fine-tuning from loss landscape perspectives, 2025.

[7] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[8] Jeremy Cohen, Alex Damian, Ameet Talwalkar, J Zico Kolter, and Jason D. Lee. Understanding optimization in deep learning with central flows. In *The Thirteenth International Conference on Learning Representations*, 2025.

[9] Jeremy Cohen, Simran Kaur, Yuanzhi Li, J Zico Kolter, and Ameet Talwalkar. Gradient descent on neural networks typically occurs at the edge of stability. In *International Conference on Learning Representations*, 2021.

[10] Jeremy M. Cohen, Behrooz Ghorbani, Shankar Krishnan, Naman Agarwal, Sourabh Medapati, Michal Badura, Daniel Suo, David Cardoze, Zachary Nado, George E. Dahl, and Justin Gilmer. Adaptive gradient methods at the edge of stability, 2024.

[11] Mathieu Dagréou, Pierre Ablin, Samuel Vaiter, and Thomas Moreau. How to compute hessian-vector products? In *ICLR Blogposts 2024*, 2024. https://iclr-blogposts.github.io/2024/blog/bench-hvp/.

[12] Alex Damian, Eshaan Nichani, and Jason D. Lee. Self-stabilization: The implicit bias of gradient descent at the edge of stability. In *The Eleventh International Conference on Learning Representations*, 2023.

[13] Francesco D'Angelo, Maksym Andriushchenko, Aditya Varre, and Nicolas Flammarion. Why do we need weight decay in modern deep learning? In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[14] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 16344–16359. Curran Associates, Inc., 2022.

[15] Yann N Dauphin and Samuel Schoenholz. Metainit: Initializing learning by learning to initialize. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[16] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3366–3385, 2022.

[17] Aleksandr Dremov, Alexander Hägele, Atli Kosson, and Martin Jaggi. Training dynamics of the cooldown stage in warmup-stable-decay learning rate scheduler. *Transactions on Machine Learning Research*, 2025.

[18] S. Elaydi. *An Introduction to Difference Equations*. Undergraduate Texts in Mathematics. Springer New York, 2005.

[19] Justin Gilmer, Behrooz Ghorbani, Ankush Garg, Sneha Kudugunta, Behnam Neyshabur, David Cardoze, George Edward Dahl, Zachary Nado, and Orhan Firat. A loss curvature perspective on training instabilities of deep learning models. In *International Conference on Learning Representations*, 2022.

[20] Jacopo Graldi, Alessandro Breccia, Giulia Lanzillotta, Thomas Hofmann, and Lorenzo Noci. The importance of being lazy: Scaling limits of continual learning. In *Forty-second International Conference on Machine Learning*, 2025.

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

[22] Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural Computation*, 9:1–42, 1997.

[23] Shengding Hu, Yuge Tu, Xu Han, Ganqu Cui, Chaoqun He, Weilin Zhao, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Xinrong Zhang, Zhen Leng Thai, Chongyi Wang, Yuan Yao, Chenyang Zhao, Jie Zhou, Jie Cai, Zhongwu Zhai, Ning Ding, Chao Jia, Guoyang Zeng, dahai li, Zhiyuan Liu, and Maosong Sun. MiniCPM: Unveiling the potential of small language models with scalable training strategies. In *First Conference on Language Modeling*, 2024.

[24] Jianheng Huang, Leyang Cui, Ante Wang, Chengyi Yang, Xinting Liao, Linfeng Song, Junfeng Yao, and Jinsong Su. Mitigating catastrophic forgetting in large language models with self-synthesized rehearsal. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1416–1428, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[25] Stanislaw Jastrzebski, Maciej Szymczak, Stanislav Fort, Devansh Arpit, Jacek Tabor, Kyunghyun Cho*, and Krzysztof Geras*. The break-even point on optimization trajectories of deep neural networks. In *International Conference on Learning Representations*, 2020.

[26] Dayal Singh Kalra and Maissam Barkeshli. Phase diagram of early training dynamics in deep neural networks: effect of the learning rate, depth, and width. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[27] Dayal Singh Kalra and Maissam Barkeshli. Why warmup the learning rate? underlying mechanisms and improvements. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[28] Dayal Singh Kalra, Tianyu He, and Maissam Barkeshli. Universal sharpness dynamics in neural network training: Fixed point analysis, edge of stability, and route to chaos. In *The Thirteenth International Conference on Learning Representations*, 2025.

[29] Simran Kaur, Jeremy Cohen, and Zachary Chase Lipton. On the maximum hessian eigenvalue and generalization. In Javier Antorán, Arno Blaas, Fan Feng, Sahra Ghalebikesabi, Ian Mason, Melanie F. Pradier, David Rohde, Francisco J. R. Ruiz, and Aaron Schein, editors, *Proceedings on "I Can't Believe It's Not Better! - Understanding Deep Learning Through Empirical Falsification" at NeurIPS 2022 Workshops*, volume 187 of *Proceedings of Machine Learning Research*, pages 51–65. PMLR, 03 Dec 2023.

[30] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

[31] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research).

[32] Aitor Lewkowycz, Yasaman Bahri, Ethan Dyer, Jascha Sohl-Dickstein, and Guy Gur-Ari. The large learning rate phase of deep learning: the catapult mechanism, 2020.

[33] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[34] Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, Kushal Arora, Saurabh Garg, Rui Xin, Niklas Muennighoff, Reinhard Heckel, Jean Mercat, Mayee Chen, Suchin Gururangan, Mitchell Wortsman, Alon Albalak, Yonatan Bitton, Marianna Nezhurina, Amro Abbas, Cheng-Yu Hsieh, Dhruba Ghosh, Josh Gardner, Maciej Kilian, Hanlin Zhang, Rulin Shao, Sarah Pratt, Sunny Sanyal, Gabriel Ilharco, Giannis Daras, Kalyani Marathe, Aaron Gokaslan, Jieyu Zhang, Khyathi Chandu, Thao Nguyen, Igor Vasiljevic, Sham Kakade, Shuran Song, Sujay Sanghavi, Fartash Faghri, Sewoong Oh, Luke Zettlemoyer, Kyle Lo, Alaaeldin El-Nouby, Hadi Pouransari, Alexander Toshev, Stephanie Wang, Dirk Groeneveld, Luca Soldaini, Pang Wei Koh, Jenia Jitsev, Thomas Kollar, Alexandros G. Dimakis, Yair Carmon, Achal Dave, Ludwig Schmidt, and Vaishaal Shankar. Datacomp-lm: In search of the next generation of training sets for language models, 2025.

[35] David Lopez-Paz and Marc' Aurelio Ranzato. Gradient episodic memory for continual learning. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[36] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[37] Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *IEEE Transactions on Audio, Speech and Language Processing*, 33:3776–3786, 2025.

[38] Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. volume 24 of *Psychology of Learning and Motivation*, pages 109–165. Academic Press, 1989.

[39] Sean McLeish, Ang Li, John Kirchenbauer, Dayal Singh Kalra, Brian R. Bartoldson, Bhavya Kailkhura, Avi Schwarzschild, Jonas Geiping, Tom Goldstein, and Micah Goldblum. Teaching pretrained language models to think deeper with retrofitted recurrence, 2025.

[40] Yan Pan and Yuanzhi Li. Toward understanding why adam converges faster than SGD for transformers. In *OPT 2022: Optimization for Machine Learning (NeurIPS 2022 Workshop)*, 2022.

[41] Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024.

[42] Anthony Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2):123–146, 1995.

[43] Vincent Roulet, Atish Agarwala, Jean-Bastien Grill, Grzegorz Michal Swirszcz, Mathieu Blondel, and Fabian Pedregosa. Stepping on the edge: Curvature aware learning rate tuners. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[44] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[45] Thomas Scialom, Tuhin Chakrabarty, and Smaranda Muresan. Fine-tuned language models are continual learners. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6107–6122, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics.

[46] Minhak Song, Kwangjun Ahn, and Chulhee Yun. Does SGD really happen in tiny subspaces? In *The Thirteenth International Conference on Learning Representations*, 2025.

[47] Javan Tahir, Surya Ganguli, and Grant M. Rotskoff. Features are fate: a theory of transfer learning in high-dimensional regression. In *Forty-second International Conference on Machine Learning*, 2025.

[48] Sharan Vaswani, Aaron Mishkin, Issam Laradji, Mark Schmidt, Gauthier Gidel, and Simon Lacoste-Julien. Painless stochastic gradient: Interpolation, line-search, and convergence rates. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[49] Evan Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, Allyson Ettinger, Michal Guerquin, David Heineman, Hamish Ivison, Pang Wei Koh, Jiacheng Liu, Saumya Malik, William Merrill, Lester James Validad Miranda, Jacob Morrison, Tyler Murray, Crystal Nam, Jake Poznanski, Valentina Pyatkin, Aman Rangapur, Michael Schmitz, Sam Skjonsberg, David Wadden, Christopher Wilhelm, Michael Wilson, Luke Zettlemoyer, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. 2 OLMo 2 furious (COLM's version). In *Second Conference on Language Modeling*, 2025.

[50] Kaiyue Wen, Zhiyuan Li, Jason S. Wang, David Leo Wright Hall, Percy Liang, and Tengyu Ma. Understanding warmup-stable-decay learning rates: A river valley loss landscape view. In *The Thirteenth International Conference on Learning Representations*, 2025.

[51] Lei Wu, Chao Ma, and Weinan E. How sgd selects the global minima in over-parameterized learning: A dynamical stability perspective. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

A Estimating Critical Learning Rate Using Forward Passes

This section provides additional details on how to measure the critical learning rate using only forward passes. We generalize the line search method proposed by [27] to accommodate a generic initial guess η_0 and modify the exit condition for the binary search to get a better estimate of the critical sharpness. Given the update direction $\Delta\boldsymbol{\theta}$ from training, we compute the critical learning rate η_c using a two-phase line search procedure:

A.1 Estimating Critical Learning Rate

Exponential Search: Starting from an initial guess η_0, we iteratively double (half) the learning rate η until the loss $L(\boldsymbol{\theta} - \eta\Delta\boldsymbol{\theta})$ exceeds (or falls below) the current loss $L(\boldsymbol{\theta})$. This quickly identifies an interval $[\eta_{\text{lower}}, \eta_{\text{upper}}]$ containing η_c, with $\eta_{\text{upper}} = 2\eta_{\text{lower}}$. Notably, each iteration requires only a single forward pass to evaluate the loss. The full algorithm is provided in Algorithm 1.

The number of exponential iterations depends on how close the initial guess η_0 is to the true critical learning rate η_c. After the first iteration, we update the initial guess to our current estimate of η_c, i.e., $\eta_0 = \eta_c$. In the subsequent steps, only $1 - 2$ exponential steps are needed, since η_c tends to remain stable. However, if the training exhibits a large instability, causing the landscape to drastically change, more steps may be necessary. To prevent indefinite iterations in such edge cases, we cap the exponential search to at most 40 iterations, which corresponds to increasing or decreasing the learning rate by a factor of 10^{12} relative to the initial guess.

Algorithm 1 Exponential Search

Input: Update direction $\Delta\boldsymbol{\theta}$, Initial loss $L(\boldsymbol{\theta})$, initial guess η_0, maximum iterations $N_{\max}$
Output: Interval $[\eta_{\text{lower}}, \eta_{\text{upper}}]$ containing the critical learning rate η_c
$\eta \leftarrow \eta_0$
$i \leftarrow 1$ // Iteration
$L(\boldsymbol{\theta} - \eta\Delta\boldsymbol{\theta}) \leftarrow \text{ComputeLoss}(\eta, \Delta\boldsymbol{\theta})$
if $L(\boldsymbol{\theta} - \eta\Delta\boldsymbol{\theta}) < L(\boldsymbol{\theta})$ **then**
  dir $\leftarrow +1$ // Increase learning rate
else
  dir $\leftarrow -1$ // Decrease learning rate
end if
while $i < N_{\max}$ **do**
  $\eta \leftarrow \eta \times 2^{\text{dir}}$
  $L(\boldsymbol{\theta} - \eta\Delta\boldsymbol{\theta}) \leftarrow \text{ComputeLoss}(\eta, \Delta\boldsymbol{\theta})$
  $i \leftarrow i + 1$
  if dir $= +1$ **and** $L(\boldsymbol{\theta} - \eta\Delta\boldsymbol{\theta}) > L(\boldsymbol{\theta})$ **then**
    $\eta_{\text{lower}} \leftarrow \eta/2$
    $\eta_{\text{upper}} \leftarrow \eta$
    return $[\eta_{\text{lower}}, \eta_{\text{upper}}]$
  end if
  if dir $= -1$ **and** $L(\boldsymbol{\theta} - \eta\Delta\boldsymbol{\theta}) < L(\boldsymbol{\theta})$ **then**
    $\eta_{\text{lower}} \leftarrow \eta$
    $\eta_{\text{upper}} \leftarrow 2\eta$
    return $[\eta_{\text{lower}}, \eta_{\text{upper}}]$
  end if
end while
return $[\eta, \eta]$

Binary Search: We then refine this interval using a binary search. At each iterate, we evaluate the loss at the midpoint $\eta_{\text{mid}} = \frac{1}{2}(\eta_{\text{lower}} + \eta_{\text{upper}})$ and shorten the interval accordingly. This process is continued until the relative error $|1 - \eta_{\text{lower}}/\eta_{\text{upper}}|$ falls below a specified threshold ϵ. For $\epsilon = \frac{1}{2^k}$, the binary search converges in k steps. In practice, we find that setting $\epsilon = 1/16$ (i.e., $k = 4$) provides a reliable and efficient estimate of the critical learning rate. The full algorithm is detailed in Algorithm 2.

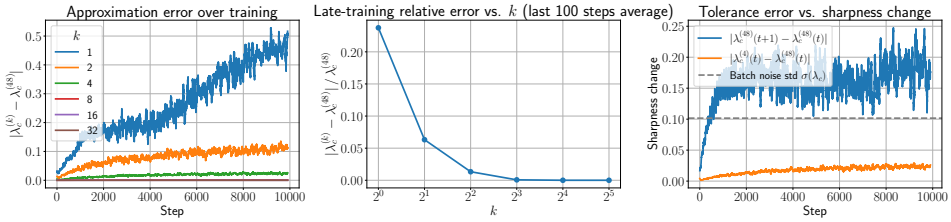


Figure 8: *Setting $k = 4$ ($\epsilon = 1/16$) provides a reliable estimate of critical sharpness.* Approximation error of critical sharpness $\lambda_c^{(k)}$ relative to the near-exact estimate $\lambda_c^{(k=48)}$, for varying binary search depth k (where $\epsilon = 1/2^k$). *Left:* absolute error throughout training. *Middle:* relative error over the last 100 steps (where error is largest). *Right:* $k = 4$ approximation error compared to step-to-step sharpness change and batch-to-batch standard deviation (dashed gray line) at step $t = 5000$.

Algorithm 2 Binary Search

Input: Update direction $\Delta\boldsymbol{\theta}$, Initial loss $L(\boldsymbol{\theta})$, initial interval $[\eta_{\text{lower}}, \eta_{\text{upper}}]$, tolerance ϵ
Output: Interval $[\eta_{\text{lower}}, \eta_{\text{upper}}]$ containing the critical learning rate η_c s.t. $\left|1 - \frac{\eta_{\text{lower}}}{\eta_{\text{upper}}}\right| < \epsilon$
i $\leftarrow 0$ // iteration
while $|1 - \eta_{\text{lower}}/\eta_{\text{upper}}| > \epsilon$ **do**
  $\eta_{\text{mid}} \leftarrow \frac{1}{2}(\eta_{\text{lower}} + \eta_{\text{upper}})$
  $L(\boldsymbol{\theta} - \eta_{\text{mid}}\Delta\boldsymbol{\theta}) \leftarrow \text{ComputeLoss}(\eta_{\text{mid}}, \Delta\boldsymbol{\theta})$
  $i \leftarrow i + 1$
  if $L(\boldsymbol{\theta} - \eta_{\text{mid}}\Delta\boldsymbol{\theta}) > L(\boldsymbol{\theta})$ **then**
    $\eta_{\text{upper}} \leftarrow \eta_{\text{mid}}$
  else
    $\eta_{\text{lower}} \leftarrow \eta_{\text{mid}}$
  end if
end while
return $[\eta_{\text{lower}}, \eta_{\text{upper}}]$

Finally, we approximate the critical learning rate using the mean of the final range $\eta_c \approx \frac{1}{2}(\eta_{\text{lower}} + \eta_{\text{upper}})$. Overall, this procedure reliably computes the critical learning rate in $5 - 6$ forward passes.

A.2 Effect of Binary Search Tolerance

We evaluate the effect of binary search depth k on the quality of the critical sharpness estimate, using the MLP CIFAR-10 experiment (batch size 500) as a testbed. We treat $k = 48$ ($\epsilon \sim 10^{-15}$) as the ground truth. The left panel of Figure 8 shows that $k = 4$ (used throughout the paper) achieves absolute error well below 0.05 across all training steps, while smaller k exhibit substantially larger and growing errors. The middle panel reports relative error over the final 100 steps, where it is largest: at $k = 4$ this stays below 2%, with diminishing returns for larger k. The right panel contextualizes the error: the $k = 4$ approximation error is $\sim 10\times$ smaller than the step-to-step sharpness change and $\sim 5\times$ smaller than the batch-to-batch standard deviation, confirming that $k = 4$ introduces negligible error relative to the dynamics of interest.

A.3 Wall-Clock Time and Memory Comparison

Wall-Clock Time Comparison. In Figure 9, we compare the wall-clock time of computing the Hessian, directional, and critical sharpness as a function of model size, using MLPs with a batch size 500.

Hessian sharpness, estimated using the LOBPCG eigenvalue solver, requires multiple HVP iterations to converge, with each HVP roughly equivalent to 5–6 forward passes. While at most training steps eigensolvers converge in a few iterations, convergence can take up 100s of iterations when training navigates through ill-conditioned regions of the landscape. In contrast, critical sharpness always requires only 5–6 forward passes in total (equivalent to a *single* HVP iteration), and in practice is 200–500$\times$ faster than Hessian sharpness across model sizes, with the gap widening at scale.

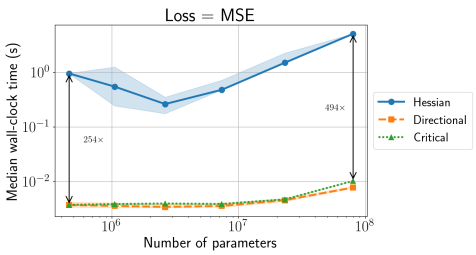


Figure 9: *Critical sharpness is* 200–500× *faster to compute than Hessian sharpness.* Median wall-clock time as a function of model size for the three sharpness measures, showing that the gap between Hessian and critical sharpness grows with model size. Hessian sharpness is computed using the LOBPCG eigenvalue solver.

Directional sharpness also requires a single HVP; therefore, its median wall-clock time is comparable to critical sharpness, as confirmed in Figure 9. However, critical sharpness has an important practical advantage: it relies solely on forward passes, making it fully compatible with standard first-order distributed training infrastructure, including parameter-sharded training. By comparison, directional sharpness requires double backpropagation to compute the HVP, which is incompatible with memory-efficient kernels such as FlashAttention [14] that lack second-derivative implementations.

Memory overhead. All three measures require storing one additional vector of size equal to the number of parameters: critical sharpness caches the original parameters $\boldsymbol{\theta}$ to evaluate $L(\boldsymbol{\theta} - \eta\Delta\boldsymbol{\theta})$, directional sharpness stores $\Delta\boldsymbol{\theta}$ for the HVP, and Hessian sharpness stores the current eigenvector iterate. The memory overhead is therefore equivalent across all three methods.

Critical sharpness estimation. In all experiments, we measure the critical learning rate on the same batch used to compute the update direction $\Delta\boldsymbol{\theta}$, making the line search deterministic given a batch.

B Experimental Details

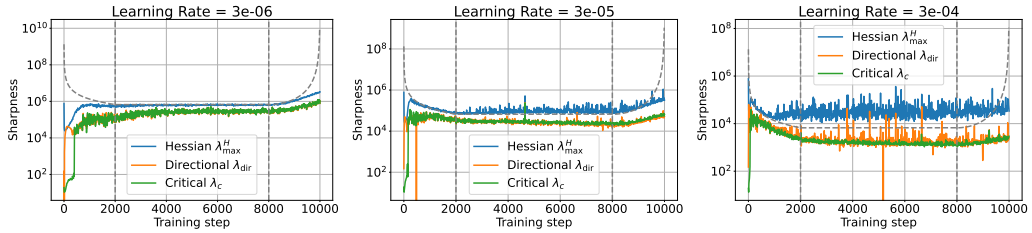
Figure 1: We considered GPT-style Pre-LN Transformers consisting of 12 layers, with an embedding dimension of $n_{\text{embd}} = 768$. The model in total has $\sim$ 100M parameters. We train the model on the 10B token subset of the FineWebEdu dataset with $\sim$ 1M tokens per step using AdamW with constant learning rate and hyperparameters $\beta_1 = 0.9$ and $\beta_2 = 0.95$. Pre-conditioned sharpness is normalized by $(1 + \beta_1)/(1 - \beta_1)$ such that it oscillates at $2/\eta$ at EoS.

Figure 4: Loss and sharpness trajectories for an MLP trained on CIFAR-10 with a linearly increasing learning rate. In the loss figure, the vertical dotted line marks the onset of loss increase, and the shaded region indicates the loss spike region. In the sharpness figure, the dashed line denotes the $2/\eta_t$ threshold, and the shaded regions in the sharpness panel mark steps where the sharpness measure exceeds $2/\eta_t$. Hessian sharpness exceeds $2/\eta_t$ from step $t = 19$, yet the loss does not increase until step $t = 33$, which coincides with directional sharpness crossing the threshold instead.

Figure 3: We consider Multi-Layer Perceptrons (MLPs) consisting of four layers, with width 512 and GeLU activation function. The models were trained on the CIFAR-10 image classification task using SGD with a constant learning rate of η =3e-02. We experimented with three different batch sizes: $B \in [500, 5000, 50000]$, with a batch size of $50,000$ corresponds to full-batch gradient descent.

Figure 5: We considered GPT-style Pre-LN Transformers consisting of 12 layers, with an embedding dimension of $n_{\text{embd}} = 768$. The model in total has $\sim$ 100M parameters. We train the model on the 10B token subset of FineWebEdu dataset with $\sim$ 1M tokens per step using AdamW and hyperparameters $\beta_1 = 0.9$ and $\beta_2 = 0.95$. Pre-conditioned sharpness is normalized by $(1 + \beta_1)/(1 - \beta_1)$ so that it oscillates at $^2/_\eta$ at EoS. This makes the comparison with critical and directional sharpness straightforward, which oscillate around $^2/_\eta$.

Figures 6 and 15: We evaluate OLMo-2 checkpoints throughout pre-training and mid-training using the optimizer states stored with each checkpoint. For each stage, we use one fixed evaluation subset drawn from the corresponding dataset: DCLM for pre-training and the Dolmino mix for mid-training. Matching the original OLMo-2 training configuration, we use a batch size of 1,024 and a context



(a) (b) (c)

Figure 10: Dynamics of pre-conditioned, directional, and critical sharpness during GPT-style Transformer training on FineWebEdu with AdamW. Critical sharpness tracks pre-conditioned sharpness trends throughout training. Pre-conditioned sharpness is normalized by $(1+\beta_1)/(1-\beta_1)$ and shifted by γ so that all three measures share the same $2/\eta$ stability threshold (colored dashed lines)

length of $4{,}096$ (4M tokens) at 7B, whereas at 1B we use a batch size of 512 with the same context length. The OLMo-2 training procedure runs three mid-training trajectories and combines their final checkpoints into a model soup. The mid-training curves show their mean sharpness and the variation across them.

Figure 7: We consider the last OLMo-2 7B as our starting point and further train on a mixture composed of DCLM and the math subset of Dolmino mix [49] with a context length of 4096. In Figure 7(a), we evaluate the relative critical sharpness for several evaluation tasks across DCLM ratios. (b, c) We further train the pre-trained checkpoint on the train mix for $16,000$ steps (1B tokens) using AdamW with a constant learning rate [5], batch size of 16 and hyperparameters $\beta_1 = 0.9$ and $\beta_2 = 0.99$

Compute Usage. All experiments were run on H100 GPUs. The data-mixing experiments on OLMo-2 7B (Figure 7) amount for most of the compute: each of the $\sim$ 80 configurations in the phase diagrams required $\sim$ 40 hours on a single node (8 GPUs), totaling approximately $25{,}000$ H100 GPU-hours. All other experiments (MLPs on CIFAR-10 and 100M GPT-style Transformers on FineWebEdu) together used under $1{,}000$ H100 GPU-hours.

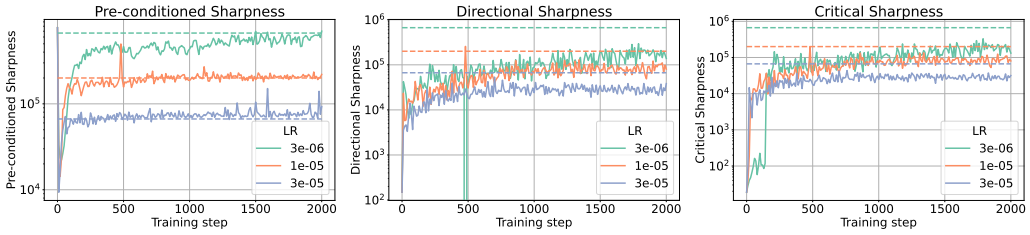
C Additional Results

C.1 GPT Pre-training

Figure 10 presents the same results as Figure 5, but directly compares different sharpness values at a fixed learning rate.

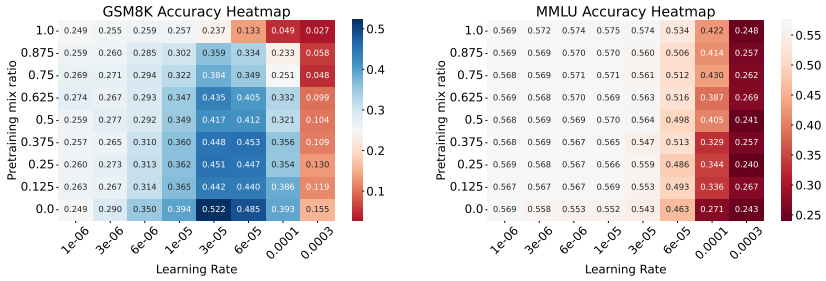
Figure 11 compares the training trajectories of pre-conditioned sharpness, directional sharpness and critical sharpness for GPT pre-training using AdamW with a fixed learning rate schedule. The pre-conditioned sharpness exhibits continually increases (progressive sharpening) until it reaches the stability threshold and oscillates around it (Edge of Stability). Both directional and critical sharpness also exhibit progressive sharpening and EoS behavior, while oscillating below the EoS threshold. The oscillation below the threshold is expected from Result 2.1, as critical sharpness includes contributions from other eigendirections as well.

[5] We consider a constant learning rate because we are only training the model for 1B tokens, as compared to the OLMo mid-training for 50B tokens.



(a) (b) (c)

Figure 11: Dynamics of pre-conditioned, directional, and critical sharpness during GPT-style Transformer training on FineWebEdu using AdamW with a constant learning rate. Pre-conditioned sharpness is normalized by $(1+\beta_1)/(1-\beta_1)$ and shifted by γ so that all three measures share the same $2/\eta$ stability threshold (colored dashed lines)



(a) (b)

Figure 12: GSM8K and MMLU accuracy as a function of pre-training (DCLM) mix ratio and learning rate. Red indicates a decrease in performance relative to the checkpoint, white indicates no change, and blue indicates an improvement.

C.2 OLMo trajectory analysis

Figure 13 shows the downstream performance of OLMo-2 7B checkpoints during both pre-training (left column) and mid-training (right column). We observe that MMLU shows consistent improvement throughout both phases, indicating steady improvement. GSM8K also improves during pre-training, but exhibits a dramatic improvement at a single checkpoint (5B tokens) during mid-training, where its accuracy jumps from 25% to 58%. As GSM8K exhibits such a quick improvement within a short training window, we specifically consider it as the candidate for further improvement during mid-training. This allows us to observe significant improvements without requiring extensive training. In contrast, Hellaswag and OpenBookQA exhibit rapid early improvements during pre-training, but do not show appreciable gains during mid-training.

Figure 14 shows the OLMo-2 7B pre-training and mid-training trajectories. The measured critical sharpness remains substantially below the Edge of Stability boundary $2/\eta_t$ throughout both stages. This gap may reflect OLMo-2's conservative learning-rate schedule, our use of a fixed evaluation subset rather than the sequence of batches encountered during training, or the possibility that large-scale training does not operate at the Edge of Stability. We repeat the trajectory analysis for OLMo-2 1B. Figure 15 shows the same qualitative pattern as at 7B: an early decrease followed by progressive sharpening throughout pre-training and mid-training.

C.3 OLMo mid-training on a mixture consisting of DCLM and Math

Figure 12 shows the same results as Figure 7(b, c), but shows them as a heatmap with accuracy values.

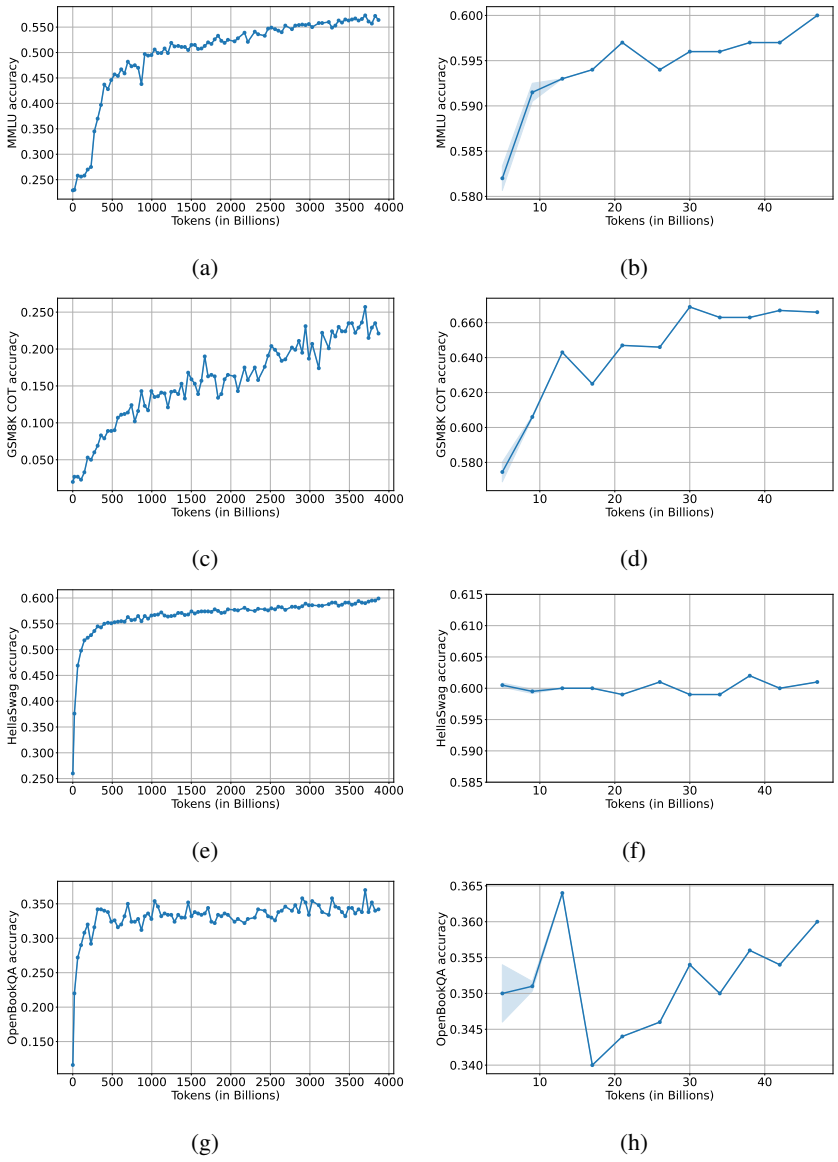
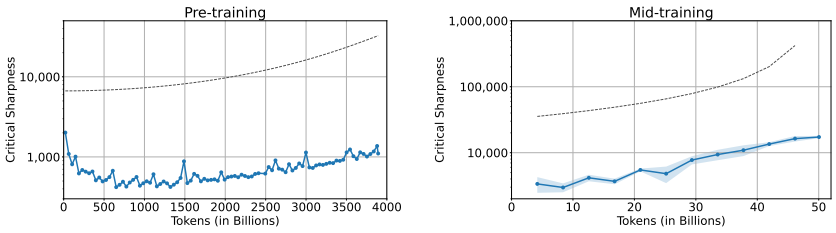


Figure 13: Downstream Performance of OLMo-2 7B checkpoints during pre-training (left column) and mid-training (right column).

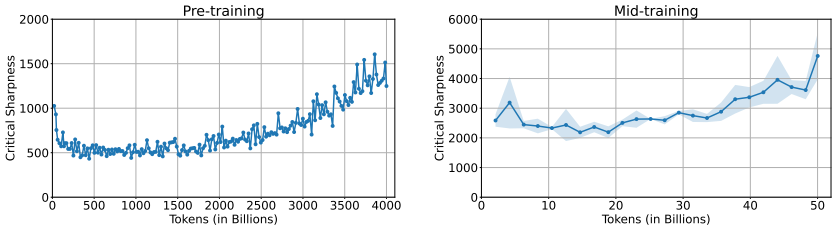
Source	*Tokens (B)*	Source %	Mix %
Filtered DCLM	752	3.23	47.2
Decontam. FLAN	17.0	50.0	16.6
StackExchange Q&A	1.26	100	2.45
peS2o	58.6	5.15	5.85
Wikipedia/Wikibooks	3.7	100	7.11
Dolmino Math	10.7	100	20.8

Table 1: Summary of Dolmino mix dataset used for mid-training OLMo-2 models.



(a) (b)

Figure 14: The OLMo-2 7B trajectories from Figure 6 with the Edge of Stability boundary $2/\eta_t$ overlaid (dashed). The measured critical sharpness remains below this boundary throughout both phases. The mid-training curve is averaged over the three model-soup runs, with the shaded band denoting variation across them.



(a) (b)

Figure 15: Critical sharpness of OLMo-2 1B throughout (a) pre-training and (b) mid-training. We evaluate checkpoints using a fixed subset of the corresponding dataset: DCLM for pre-training and the Dolmino mix for mid-training. We use a batch size of 512 and a context length of $4{,}096$. The mid-training curve is averaged over the three model-soup runs, with the shaded band denoting variation across them.

C.4 OLMo mid-training

Next, we consider OLMo mid-training with the full Dolmino mix dataset. Table 1 summarizes the composition of the 50B token Dolmino mix subset. To investigate the effect of the pre-training dataset, we keep the relative mix percentages of all components fixed except for the DCLM ratio, which we vary. Figure 16 presents the relative critical sharpness for different components of the Dolmino mix dataset. Consistent with our findings in Section 4, we observe a candidate region at a DCLM ratio around 0.6. At this ratio, the relative critical sharpness of DCLM decreases, and the sharpness curves for different tasks intersect. This intersection identifies a region that permits a large learning rate without being limited by any single task. The candidate ratio of 0.6 is close to the DCLM ratio of 47.2% used in OLMo-2 training. We leave further validation of this candidate region to future work.

C.5 Batch sensitivity

Critical sharpness depends on the sampled mini-batch. We therefore test whether batch-to-batch variability is smaller than the change in critical sharpness along the training trajectories. At 1B, measurements on 20 independent batches per checkpoint have standard errors of approximately 4–5% of the mean, while the mean increases by $2.8\times$ across the checkpoints in Table 2. At 7B, eight independent original-size batches at representative pre-training and mid-training anchors give coefficients of variation of 4.67% and 6.91%, respectively. Thus, batch variability is substantially smaller than the trajectory-scale changes.

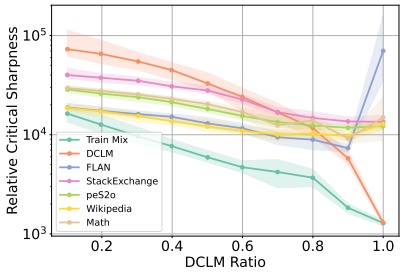


Figure 16: Relative critical sharpness for various subsets of the Dolmino mix dataset. The shaded region around the mean trends denotes the variation across batches.

Scale	Checkpoint	Batches	Mean λ_c	Batch std.	CV
1B	Pre-training, step 510,000	20	602.7	96.6	16.0%
1B	Pre-training, step 970,000	20	878.4	197.0	22.4%
1B	Pre-training, step 1,440,000	20	1045.3	197.2	18.9%
1B	Pre-training, step 1,907,359	20	1672.0	316.4	18.9%
7B	Pre-training, step 928,000	8	1113.95	52.07	4.67%
7B	Mid-training trajectory 1, step 11,931	8	16,688.27	1152.60	6.91%

Table 2: Batch-to-batch variability of OLMo-2 critical sharpness.

D Theoretical Results

In this section, we provide the proofs for the theoretical results presented in the main text.

D.1 The Relationship between Directional and Hessian sharpness

Result D.1 (Relationship between Directional and Hessian Sharpness for Gradient Descent)**.** For Gradient Descent (GD), the directional sharpness λ_{dir} can be expressed as a weighted sum of the Hessian eigenvalues $\{\lambda_i^H\}_{i=1}^n$, where the weights quantify the alignment of the gradient with Hessian eigendirections $\{u_i(\boldsymbol{\theta})\}_{i=1}^n$.

$$\lambda_{\text{dir}} = \frac{\sum_{i=1}^n c_i^2 \lambda_i^H}{\sum_{i=1}^n c_i^2} \leq \lambda_{\max}^H$$

where $c_i = g(\boldsymbol{\theta})^T u_i(\boldsymbol{\theta})$ is the projection of the gradient onto the ith eigenvector.

Proof: For gradient descent $\Delta\boldsymbol{\theta} = g(\boldsymbol{\theta})$, the directional sharpness is given by:

$$\lambda_{\text{dir}} = \frac{g(\boldsymbol{\theta})^T H(\boldsymbol{\theta}) g(\boldsymbol{\theta})}{g(\boldsymbol{\theta})^T g(\boldsymbol{\theta})}.$$

Next, writing $g(\boldsymbol{\theta}) = \sum_{i=1}^n c_i u_i(\boldsymbol{\theta})$, we have the required result:

$$\lambda_{\text{dir}} = \frac{\sum_{i=1}^n c_i^2 \lambda_i^H}{\sum_{i=1}^n c_i^2} \leq \lambda_{\max}^H$$

Result D.2 (Relationship between Directional and Hessian Sharpness for Adaptive optimizers)**.** For adaptive optimizers with pre-conditioner $P(\boldsymbol{\theta})$ and update $P^{-1}\boldsymbol{g}$ (e.g., RMSProp), the directional sharpness λ_{dir} can be expressed as a weighted sum of the pre-conditioned Hessian eigenvalues $\{\lambda_i^{PH}\}_{i=1}^n$, where the weights quantify the alignment of the pre-conditioned gradient $P^{-1/2}g(\boldsymbol{\theta})$ with pre-conditioned Hessian eigendirections $\{v_i(\boldsymbol{\theta})\}_{i=1}^n$.

$$\lambda_{\text{dir}} = \frac{\sum_{i=1}^n c_i^2 \lambda_i^{PH}}{\sum_{i=1}^n c_i^2} \leq \lambda_{\max}^{PH}$$

where $c_i = P^{-1/2} g(\boldsymbol{\theta})^T v_i(\boldsymbol{\theta})$ is the projection of the gradient $g(\boldsymbol{\theta})$ onto the ith eigenvector $v_i(\boldsymbol{\theta})$ of the pre-conditioned Hessian.

Proof. For adaptive optimizers $\Delta\boldsymbol{\theta} = P^{-1} g(\boldsymbol{\theta})$, the directional sharpness is given by:

$$\lambda_{\text{dir}} = \frac{g(\boldsymbol{\theta})^T P^{-1} H(\boldsymbol{\theta}) P^{-1} g(\boldsymbol{\theta})}{g(\boldsymbol{\theta})^T P^{-1} g(\boldsymbol{\theta})}.$$

Next, define $\phi := P^{1/2}\boldsymbol{\theta}$, then the pre-conditioned gradient and Hessian are $P^{-1/2} g(\boldsymbol{\theta})$ and $P^{-1/2} H(\boldsymbol{\theta}) P^{-1/2}$. Finally, writing the pre-conditioned gradient as $P^{-1/2} g(\boldsymbol{\theta}) = \sum_{i=1}^{n} \alpha_i v_i(\boldsymbol{\theta})$, we have the required result:

$$\lambda_{dir} = \frac{\sum_{i=1}^{n} \alpha_i^2 \lambda_i^{PH}}{\sum_{i=1}^{n} \alpha_i^2} \leq \lambda_{\max}^{PH}.$$

Remark: The derivation above assumes that the update direction is aligned with the preconditioned gradient, i.e., $\Delta\boldsymbol{\theta} = P^{-1}\boldsymbol{g}$. For optimizers with momentum (e.g., Adam), this result serves as an approximation.

D.2 Stability Threshold for Optimizers with Weight Decay

In this section, we derive stability thresholds for common optimizers with Weight Decay (WD) with strength γ using a quadratic loss function defined as:

$$L(\boldsymbol{\theta}) = \frac{1}{2}\boldsymbol{\theta}^T H \boldsymbol{\theta} + \boldsymbol{g}^T \boldsymbol{\theta} + c.$$

Result D.3 (Stability threshold for GD with WD)**.** For gradient descent, adding weight decay shifts the EoS threshold by the decay strength γ:

$$\lambda_{\max}^{H} = \frac{2}{\eta} - \gamma$$

where η is the learning rate and γ is the weight decay strength.

Proof. The GD updates with weight decay are given by

$$\boldsymbol{\theta}_{t+1} = (1 - \eta\gamma)\boldsymbol{\theta}_t - \eta\left(H\boldsymbol{\theta}_t + \boldsymbol{g}\right).$$

The GD dynamics projected along the top eigenvector $\boldsymbol{u}$ is given by:

$$\boldsymbol{u}^T\boldsymbol{\theta}_{t+1} = \boldsymbol{u}^T(1 - \eta\gamma - \eta H)\boldsymbol{\theta}_t - \eta\boldsymbol{u}^T\boldsymbol{g},$$
$$\boldsymbol{u}^T\boldsymbol{\theta}_{t+1} = (1 - \eta\gamma - \eta\lambda)\boldsymbol{u}^T\boldsymbol{\theta}_t - \eta\boldsymbol{u}^T\boldsymbol{g}.$$

Furthermore, define $q = \boldsymbol{u}^T\boldsymbol{\theta} + \frac{1}{\lambda+\gamma}\boldsymbol{u}^T\boldsymbol{g}$, then

$$q_{t+1} = (1 - \eta\gamma - \eta\lambda)q_t.$$

The sequence $\{q_t\}_{t=0}^{T}$ diverges if $|1 - \eta(\lambda + \gamma)| > 1$, which is equivalent to $\lambda > \frac{2}{\eta} - \gamma$.

To prove results for Adam, we need the following result from [18].

Lemma D.1 (Stability Condition for Difference Equations)**.** *Consider non-homogeneous difference equations of the type:*

$$q_{t+1} + p_1 q_t + p_2 q_{t-1} - c = 0.$$

The solutions of the above equation are asymptotically stable iff:

$$1 + p_1 + p_2 > 0, \qquad 1 - p_1 + p_2 > 0, \qquad 1 - p_2 > 0$$

Result D.4 (Stability threshold for AdamW)**.** For Adam, adding weight decay shifts the EoS threshold a constant that depends on the decay strength γ:

$$\lambda_{\max}^{PH} = \left(\frac{2}{\eta} - \gamma\right)\left(\frac{1 + \beta_1}{1 - \beta_1}\right)$$

where η is the learning rate and γ is the weight decay strength.

Proof. The update equations of AdamW can be written as:

$$\boldsymbol{m}_{t+1} = \beta_1 \boldsymbol{m}_t + (1-\beta_1)\nabla_\theta L$$
$$\boldsymbol{\theta}_{t+1} = (1-\eta\gamma)\boldsymbol{\theta}_t - \eta P_{t+1}^{-1}\boldsymbol{m}_{t+1},$$

where $P_{t+1} = (1-\beta_1^{t+1})\left[\text{diag}(\sqrt{\frac{\boldsymbol{v}_{t+1}}{(1-\beta_2^{t+1})}}) + \epsilon \boldsymbol{I}\right]$ is Adam's pre-conditioner with $\boldsymbol{v}_{t+1} = \beta_2 \boldsymbol{v}_t + (1-\beta_2)(\nabla_\theta L)^2$ is the moving average of the squared gradients.

Next, we multiply the parameter update equation by P_t and rearrange the terms to obtain:

$$P_{t+1}\boldsymbol{\theta}_{t+1} = (1-\eta\gamma)P_{t+1}\boldsymbol{\theta}_t - \eta \boldsymbol{m}_{t+1}$$
$$\boldsymbol{m}_{t+1} = \frac{1}{\eta}P_{t+1}\left[(1-\eta\gamma)\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t+1}\right].$$

Next, shift the time index back by one to get $\boldsymbol{m_t}$:

$$\boldsymbol{m}_t = \frac{1}{\eta}P_t\left[(1-\eta\gamma)\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}_t\right].$$

Inserting this in the momentum update equation:

$$\boldsymbol{m}_{t+1} = \beta_1 \boldsymbol{m}_t + (1-\beta_1)(H\boldsymbol{\theta}_t + \boldsymbol{g}),$$

we obtain:

$$\frac{1}{\eta}P_{t+1}\left[(1-\eta\gamma)\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t+1}\right] = \beta_1\frac{1}{\eta}P_t\left[(1-\eta\gamma)\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}_t\right] + (1-\beta_1)(H\boldsymbol{\theta}_t + \boldsymbol{g}).$$

Next, we assume that the pre-conditioner is changing slowly and as a result $P = P_{t+1} \approx P_t$. On multiplying the above equation with ηP_{t+1}^{-1}, we get:

$$(1-\eta\gamma)\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t+1} = \beta_1(1-\eta\gamma)\boldsymbol{\theta}_{t-1} - \beta_1\boldsymbol{\theta}_t + \eta(1-\beta_1)P^{-1}H\boldsymbol{\theta}_t + \eta(1-\beta_1)P^{-1}\boldsymbol{g}.$$

Next, we rearrange the terms to group $\boldsymbol{\theta}_{t+1}, \boldsymbol{\theta}_t$ and $\boldsymbol{\theta}_{t-1}$:

$$\boldsymbol{\theta}_{t+1} = \left[(1-\eta\gamma+\beta_1-\eta(1-\beta_1)P^{-1}H)\right]\boldsymbol{\theta}_t - \beta_1(1-\eta\gamma)\boldsymbol{\theta}_{t-1} - \eta(1-\beta_1)P^{-1}\boldsymbol{g}_t.$$

Next, we multiply the equation by $P^{1/2}$ and define the transformed variable $\phi_t := P^{1/2}\boldsymbol{\theta}_t$:

$$\boldsymbol{\phi}_{t+1} = \left[(1-\eta\gamma+\beta_1-\eta(1-\beta_1)P^{-1/2}HP^{-1/2})\boldsymbol{\phi}_t - \beta_1(1-\eta\gamma)\boldsymbol{\phi}_{t-1} - \eta(1-\beta_1)P^{-1/2}\boldsymbol{g}_t\right] \tag{7}$$

Next, we multiply on the left with the top eigenvector of the pre-conditioned Hessian $\boldsymbol{v}_{\max}$ and define $q_t = \boldsymbol{v}_{\max}^T\boldsymbol{\phi}_t$ to obtain:

$$q_{t+1} = \left[(1-\eta\gamma+\beta_1-\eta(1-\beta_1)\lambda_{\max}^{PH})\right]q_t - \beta_1(1-\eta\gamma)q_{t-1} - \eta(1-\beta_1)\boldsymbol{v}_{\max}P^{-1/2}\boldsymbol{g}.$$

Comparing the equation with Lemma D.1, we have:

$$p_1 = -\left[1-\eta\gamma+\beta_1-\eta(1-\beta_1)\lambda_{\max}^{PH}\right]$$
$$p_2 = \beta_1(1-\eta\gamma).$$

Plugging the above expressions into the first stability condition $1-p_1+p_2 > 0$, we have:

$$1 + \left[1-\eta\gamma+\beta_1-\eta(1-\beta_1)\lambda_{\max}^{PH}\right] + \beta_1(1-\eta\gamma) > 0,$$

which yields the desired result:

$$\lambda_{\max}^{PH} < \frac{2+2\beta_1}{\eta(1-\beta_1)} - \gamma\left(\frac{1+\beta_1}{1-\beta_1}\right) = \left(\frac{2}{\eta}-\gamma\right)\left(\frac{1+\beta_1}{1-\beta_1}\right).$$

The other two stability conditions are trivially satisfied.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: In the abstract and introduction, we claim: (1) critical sharpness as a computationally efficient curvature measure (Section 2.1), (2) its theoretical relation to Hessian sharpness (Results 2.1 and 2.2), (3) empirical evidence of progressive sharpening at the 7B scale (Section 2), and (4) the data-mixing application through relative critical sharpness (Section 4). We justify these claims in their respective sections.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss limitations at the end of the discussion section, including the limitation of the large-scale analysis restricted to OLMo-2 7B due to the substantial baseline compute cost of working with large models. We also discuss the computational cost of estimating critical sharpness (5-6 forward passes) and perform wall-clock time analysis for models up to 80M parameters.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The theoretical relationship between directional and Hessian sharpness for GD and adaptive optimizers (Results 2.1 and 2.2) and the stability thresholds for GD and Adam with weight decay (Section D) are stated with their assumptions and proved in their respective sections in the Appendix to the best of our knowledge.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide extensive details in the main text and Section B, including model architectures, datasets, optimizer hyperparameters, learning rate schedules, and batch sizes for all experiments. The line-search algorithm for estimating critical sharpness is fully specified in Section A.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: We do not release code with the paper. However, we provide extensive experimental details (Section B) and a complete specification of the line-search algorithm for estimating critical sharpness (Section A), which can be used to reproduce our results. All datasets and pre-trained checkpoints used are publicly available.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: All training details, including model architectures, datasets, optimizers, hyperparameters, batch sizes, and learning rate schedules, are specified in the main text and in Section B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: Training trajectories are reported from single runs, as is standard for sharpness dynamics studies. For the OLMo-2 7B mid-training data-mixing experiments (Figure 7), the substantial cost of training OLMo-2 7B combined with the extensive sweep over pre-training mix ratios and learning rates precluded multiple seeds per configuration. However, we observe consistent qualitative behavior across all scales we study, from MLPs on CIFAR-10 to OLMo-2 7B, lending confidence to the generality of our findings.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Compute resource requirements are mentioned in Section B. The data-mixing experiments on OLMo-2 7B (Figure 7) required over 25,000 H100 GPU-hours. Smaller-scale experiments (MLPs on CIFAR-10, 100M GPT-style Transformers on FineWebEdu) took about 1000 H100 hours in total.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

Justification: We confirm that this research conforms with the NeurIPS Code of Ethics. This work studies optimization hyperparameters for neural network training and does not involve human subjects, sensitive data, or dual-use concerns.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [N/A]

Justification: This work studies the optimization of neural networks. We do not foresee any direct negative societal impacts beyond the general risks associated with advances in machine learning.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: This work does not release any models, datasets, or other assets that pose a high risk for misuse.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.

- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All existing assets used in the paper are properly credited and used under permissive research licenses: CIFAR-10 [31] (free for research use), FineWebEdu [41] (ODC-BY), DCLM [34] (MIT), Dolmino mix and OLMo-2 checkpoints [49] (Apache 2.0), GSM8K [7] (MIT), and MMLU [21] (MIT).

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: No new assets are introduced.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: Our work does not involve crowdsourcing or human subjects.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer:[N/A]

    Justification: Our work does not involve crowdsourcing or human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: We only used LLMs for pruning, editing, and formatting purposes.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.