Training Dynamics Underlying Language Model Scaling Laws: Loss Deceleration and Zero-Sum Learning

Andrei Mircea [1,2,3], Supriyo Chakraborty [3], Nima Chitsazan[3], Irina Rish[1,2], Ekaterina Lobacheva[1,2]
[1]Mila – Quebec AI Institute, [2]Université de Montréal, [3]Capital One
Correspondence: mirceara@mila.quebec

Abstract

This work aims to understand how scaling improves language models, specifically in terms of training dynamics. We find that language models undergo *loss deceleration* early in training—an abrupt slowdown in the rate of loss improvement, resulting in piecewise linear behaviour of the loss curve in log-log space. Scaling up the model mitigates this transition by (1) decreasing the loss at which deceleration occurs, and (2) improving the log-log rate of loss improvement after deceleration. We attribute loss deceleration to a type of degenerate training dynamics we term *zero-sum learning* (ZSL). In ZSL, per-example gradients become systematically opposed, leading to destructive interference in per-example changes in loss. As a result, improving loss on one subset of examples degrades it on another, bottlenecking overall progress. Loss deceleration and ZSL provide new insights into the training dynamics underlying language model scaling laws, and could potentially be targeted directly to improve language models independent of scale. We make our code and artefacts available at: https://github.com/mirandrom/zsl

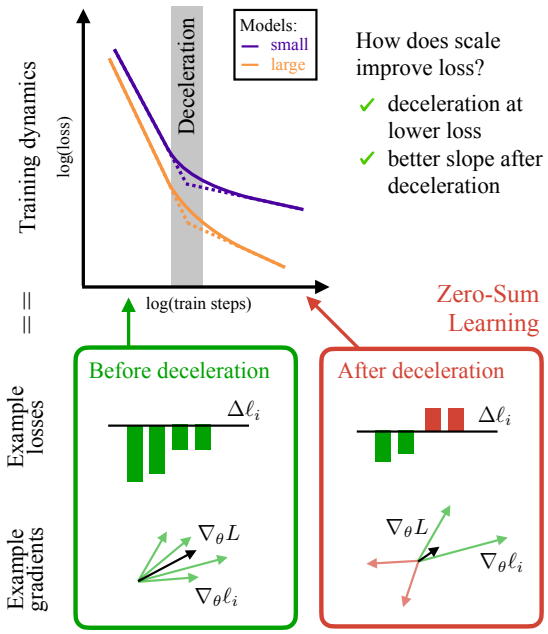


Fig. 1: Paper overview. Top: Language model loss curves exhibit deceleration early in training. Scaling model size affects at which loss level this transition happens and how severe it is. Bottom: Loss deceleration can be attributed to *zero-sum learning* dynamics: per-example gradients $\nabla_\theta \ell_i$ become systematically opposed, leading to competing changes in per-example losses $\Delta\ell_i$ and an overall slowdown in model loss improvement.

1 Introduction

What mechanisms underlie scaling laws?
Increasing language model (LM) size empirically improves cross-entropy loss with power-law behaviour, which can be accurately described with scaling laws (Kaplan et al., 2020). Despite their predictive capabilities, scaling laws offer limited insight into the underlying mechanism (Stumpf and Porter, 2012); i.e. they do not explain *how* scaling improves loss. This question is particularly interesting because, by identifying and understanding such a mechanism (Glennan and Illari, 2017), we may become able to target it directly and improve models independent of scale.

While several recent works have sought to explain scaling laws based on notions of e.g. intrinsic model capacity (Sharma and Kaplan, 2022), data distribution properties (Michaud et al., 2023), or asymptotic behaviour (Bahri et al., 2024), mechanistic explanations that can inform new approaches and drive principled progress (beyond resource-intensive scaling) remain under-explored. In particular, little is known about the changes in training dynamics that underlie scaling improvements, which our work addresses.

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 28154–28188
July 27 - August 1, 2025

Loss deceleration underlies scaling laws. We find that scaling improvements can be explained in terms of training dynamics via *loss deceleration*, a phenomenon where rates of loss improvement slow down abruptly, resulting in loss curves that are piecewise-linear in log-log space (see Fig. 1, top). Importantly, scaling improvements can be decomposed in terms of deceleration measurements. Specifically, we show that scaling up the model size **(1)** decreases the loss at which deceleration occurs, and **(2)** increases the log-log rate of loss improvement after deceleration. This connection suggests that the mechanism behind deceleration (and the mitigating effects of scale) can help understand how scaling improves final model loss.

The piecewise-linear nature of deceleration suggests a qualitative transition in training dynamics. To the best of our knowledge, deceleration and the underlying transition in training dynamics has not been addressed in relevant prior works on e.g. loss plateaus (Yoshida and Okada, 2020), learning curve shapes (Hutter, 2021; Viering and Loog, 2022), or LM saturation (Godey et al., 2024; Mircea et al., 2024). Our work fills this gap by proposing a mechanistic explanation for loss deceleration and showing how it underlies scaling laws.

A mechanistic explanation of deceleration. We hypothesize that deceleration occurs as a result of degenerate zero-sum learning dynamics (see Fig. 1, bottom). In ZSL, per-example gradients become systematically opposed, leading to destructive interference in loss improvements. In other words, loss can not be improved on one set of examples without degrading on another, thus bottlenecking the rate at which overall loss can improve. We verify this hypothesis against alternative explanations, characterizing and validating the proposed mechanism with a complementary empirical and theoretical results.

As a mechanistic explanation (Kaplan and Craver, 2011), zero-sum learning describes how the training dynamics of individual examples (i.e. their loss and gradients) behave and interact with one another to produce loss deceleration. This approach of understanding learning dynamics from the perspective of per-example gradient alignment and opposition is similar to Mircea et al. (2024), but otherwise under-explored outside of tangential areas of research on e.g. improving multi-task learning (Liu et al., 2021), or characterizing outliers in SGD (Rosenfeld and Risteski, 2023).

Importantly, we believe zero-sum learning can potentially be mitigated directly to improve loss independent of scale. Our findings offer new insights into how scaling improves loss by mitigating deceleration, and can provide a foundation for future work in this direction.

Summary of findings and contributions In Section 2 we identify loss deceleration as a novel qualitative transition in LM training dynamics. In particular, we show how scaling improvements can be explained in terms of mitigating deceleration. In Section 3, we propose and validate a mechanistic explanation of deceleration based on destructive interference in per-example gradients and loss improvements. Lastly, in Section 4, we connect these mechanisms to scaling improvements, showing how they are mitigated in ways that could be targeted directly and independent of scale.

Methodology We adapt the training setup of Groeneveld et al. (2024) and scaling experiments of Kaplan et al. (2020), training and analyzing models between 14M and 472M parameters. Details on training and model analyses are in Appendix A. We also provide ablation experiments with different model architectures, datasets, optimizers and training hyperparameters in Appendix C.

2 Loss deceleration in language models

Characterizing loss deceleration. We find that LM loss curves typically exhibit an abrupt slow down in the rate of loss improvements early during training, in a transition we refer to as *loss deceleration*. Notably, we see in Fig. 2 that loss deceleration is characterized by piecewise-linear behaviour in log-log space, consistent across different model scales and training setups, suggesting a qualitative transition in training dynamics.

An important observation from Fig. 2 is that loss improvements from scaling can be framed in terms of mitigating this transition, i.e. by improving:

(1) the loss at which deceleration occurs; and

(2) the log-log loss slope after deceleration.

This suggests that, by understanding the mechanism underlying loss deceleration (and the mitigating effects of scale), we can shed light on *how* scaling improves loss in terms of training dynamics. Such an understanding could in turn inform methods that directly target and mitigate deceleration independent of scale. However, to study how scale mitigates deceleration, we must first measure it.

Measuring loss deceleration with BNSL.
In measuring loss deceleration, we want to capture the log-log piecewise-linear behaviour observed in Fig. 2 and quantify how it changes with scaling. Luckily, this type of function can be parametrically described and fit with smoothly broken power laws such as BNSL (Caballero et al., 2023), particularly in the simplified one-break form:

$$L(t) - a = \left(bt^{-c_0}\right)\left(1 + (t/d_1)^{1/f_1}\right)^{-c_1 f_1}, \quad (1)$$

where $L(t)$ is the loss at step t, and the remaining variables are the parameters being fit: a represents the irreducible loss; b the y-axis intercept $L(0)$; c_0 the log-log slope of the first linear segment; c_1 the difference between the slope of the second segment and the first; d_1 the step at which the break occurs; and f_1 the smoothness of the transition between segments. However, these parameters provide limited insight into how deceleration relates to loss.

Quantifying the effect of deceleration on loss.
For a more interpretable but nevertheless quantifiable connection between deceleration and loss, we can tease out the linear segments underlying Eqn. 1. Concretely, an estimate $\hat{L}_T$ of the loss L_T at step $T > d_1$ can be expressed in terms of three measurements grounded in BNSL parameters[1]:

$$\log(\hat{\boldsymbol{L}}_{\boldsymbol{T}}) = \log(L_d) - r_d \log(T/t_d) \quad (2)$$
$$\hat{\boldsymbol{L}}_{\boldsymbol{T}} = L_d\,(t_d/T)^{r_d}$$

- $\boldsymbol{t_d}$ $= d_1$, the step where deceleration occurs, or where the two segments intersect;
- $\boldsymbol{L_d}$ $= b{d_1}^{-c_0}$, the loss where deceleration occurs, or where the two segments intersect;
- $\boldsymbol{r_d}$ $= c_0 + c_1$, the log-log rate of loss improvement after deceleration, or the negative log-log slope of the second segment.

Intuitively, we see that final loss is fundamentally a function of L_d, r_d, t_d; such that scaling improvements can be explained solely in terms of increased r_d and decreased L_d, t_d. These measurements are reported in Table 1, where we indeed observe monotonic improvements in L_d, r_d and t_d with increased model size[2]. We also confirm that $\hat{L}_T$ is a valid approximation, typically within 1% of L_T.

Crucially, these are interpretable measurements of loss deceleration, allowing us to naturally describe and reason about scaling improvements in terms of training dynamics. For example, Eqn. 2 forms the basis of a novel scaling law functional form, with improved explanatory power as a result of being grounded in these interpretable quantities. While beyond the scope of this paper, we include preliminary results in Appendix B.6.

More generally, this means we can shift our goal from understanding how scaling improves loss to understanding the mechanism underlying deceleration and how scaling improves L_d and r_d. In the next section, we focus on the first question of understanding deceleration.

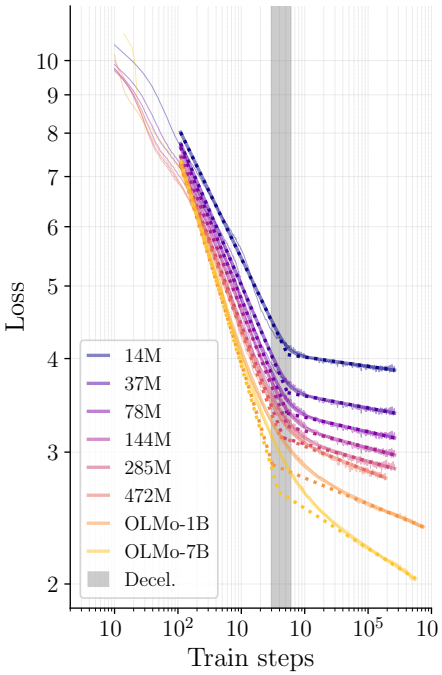


Fig. 2: Loss curves exhibit deceleration early in training (grey fill), and can be parametrically described with a one-break BNSL (Eqn. 1). The resulting BNSL fits are shown in bold, with the underlying piecewise-linear components shown as dashed lines. We also include the OLMo 1B and 7B models (Groeneveld et al., 2024), showcasing similar behaviour at larger scales.

Table 1: Loss deceleration measurements from Eqn. 2: larger models have lower L_d, t_d and higher r_d.

Model	$\downarrow L_d$	$\downarrow t_d$	$\uparrow r_d$	$\hat{L}_T$	L_T
14M	4.05	5900	0.013	3.86	3.88
37M	3.60	5900	0.016	3.39	3.40
78M	3.38	5900	0.020	3.14	3.15
144M	3.25	6000	0.023	2.98	2.99
285M	3.14	5300	0.025	2.85	2.87
472M	3.16	4600	0.035	2.77	2.80
OLMo-1B	2.86	3700	0.034	2.39	2.40
OLMo-7B	2.64	4600	0.053	2.04	2.03

[1] See Appendix A.2 in (Caballero et al., 2023).

[2] One notable outlier is t_d in OLMo-7B, likely attributable to OLMo-7B using a warmup of 5,000 rather than 2,000 steps.

3 Explaining Loss Deceleration

The log-log piecewise-linear behaviour of loss deceleration suggests that a qualitative change in training dynamics underlies the abrupt slowdown in loss improvements. Our goal in this section is to characterize this transition in training dynamics and provide a mechanistic explanation for loss deceleration. By "mechanistic explanation", we mean identifying and formalizing an underlying mechanism as defined in Glennan and Illari (2017). To this end, we propose and verify the hypothesis that loss deceleration is a transition in training dynamics characterized by *zero-sum learning*.

Zero-sum learning (ZSL) describes degenerate training dynamics where per-example gradients become systematically opposed, leading to significant destructive interference between per-example changes in loss. Put differently, ZSL corresponds to regions in parameter space where gradient descent cannot improve loss on one set of examples without commensurate loss degradation on another, effectively bottlenecking the overall rate of loss improvement. ZSL could therefore be a mechanistic explanation of how per-example gradients and loss changes interact to produce the abrupt slowdown in overall loss improvement seen in deceleration.

Verifying the ZSL hypothesis In the following sections, we break down the hypothesis that ZSL explains loss deceleration into atomic claims that we validate with empirical and theoretical results.

- **3.1** Introduces and defines measures of destructive interference used throughout our analysis.
- **3.2** Confirms deceleration *co-occurs* with increased destructive interference in per-example loss improvements and gradients.
- **3.3** Demonstrates deceleration is *primarily attributable* to destructive interference in per-example loss improvements.
- **3.4** Demonstrates destructive interference in loss improvements is *primarily attributable* to destructive interference in gradients.

Notation Let ℓ_i denote loss for token i in dataset $\mathcal{D}$, with overall loss $L = \sum_i \ell_i / |\mathcal{D}|$. Conversely, change in loss between steps t_1, t_2 is denoted as $\Delta_{t_1}^{t_2} L = \sum_i \Delta_{t_1}^{t_2} \ell_i / |\mathcal{D}|$. To reduce notation clutter, t_1, t_2 are sometimes omitted when evident from context or not relevant. Similarly, the overall gradient is denoted $\nabla_\theta L = \sum_i \nabla_\theta \ell_i / |\mathcal{D}|$ where $\nabla_\theta \ell_i$ is the gradient for token i.

3.1 Measuring Destructive Interference

To measure zero-sum learning, we define destructive interference in per-token loss improvements $\Delta\ell_i$ as the proportion with which they cancel out in overall loss improvements $\Delta L = \sum_i \Delta\ell_i / |\mathcal{D}|$, with respect to an ideal scenario where there is no interference $\Delta L^* = \sum_i |\Delta\ell_i| / |\mathcal{D}|$:

$$D(\Delta\ell) = \frac{\Delta L^* - |\Delta L|}{\Delta L^*} = 1 - \frac{|\sum_i \Delta\ell_i|}{\sum_i |\Delta\ell_i|} \quad (3)$$

Similarly, we use coordinate-level destructive interference to measure gradient opposition, typically reporting $D(\nabla_\theta \ell)$ as the average over coordinates:

$$\vec{D}\left(\nabla_\theta \ell\right) = 1 - \frac{|\sum_i \nabla_\theta \ell_i|}{\sum_i |\nabla_\theta \ell_i|} \quad (4)$$

Intuitively, $D(\Delta\ell)$ and $D(\nabla_\theta \ell)$ increase and approach 1 with ZSL. Conversely, $D(\Delta\ell)$ and $D(\nabla_\theta \ell)$ decrease and approach 0 with no ZSL.

3.2 Confirming Deceleration Occurs with ZSL

To show that loss deceleration co-occurs with ZSL, we analyse the behavior of destructive interference in both loss improvements and gradients during training. In Fig. 3, we measure destructive interference in loss improvements $D(\Delta_t^{2t}\ell)$. We observe that it exhibits a sharp increase, beginning just before deceleration, then converging towards its maximum. One important consideration is that these measurements are based on $\Delta_t^{2t}\ell$ to smooth out noise from loss oscillations on too-small timescales. In practice, we find that $D(\Delta_{t_1}^{t_2}\ell)$ is mitigated as the number of steps $t_2 - t_1$ increases, such that Fig. 3 is effectively under-reporting ZSL (Appendix B.2).

In Fig. 4, we measure gradient destructive interference[3] $D(\tilde{\nabla}_\theta \ell)$ and find that it also converges to a maximum at the same time as deceleration. Surprisingly, gradient opposition turns out to be quite high even at the start of training. Despite this high starting point, the increase in destructive interference leading up to deceleration still has a significant effect. In fact, increasing destructive interference in a sum beyond 0.9 rapidly decreases the magnitude of that sum by several orders of magnitude, as can be seen in the next section and Eqn. 6 for $D(\Delta\ell)$ and ΔL.

These results confirm that ZSL indeed co-occurs with loss deceleration, but are not sufficient evidence that ZSL is the underlying mechanism. The

[3] A tractable proxy for per-example gradients, described and empirically validated in Appendix A.3.

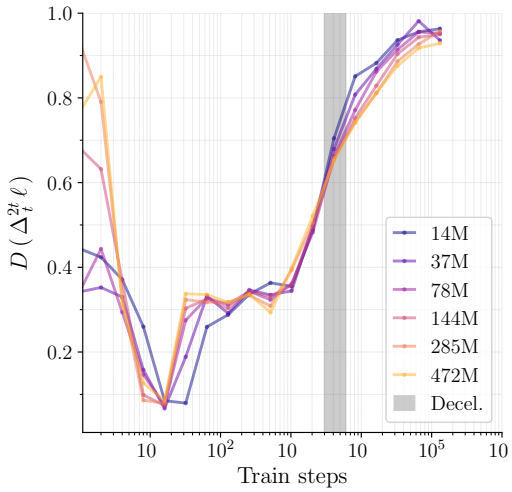


Fig. 3: ZSL throughout training, as measured by destructive interference in per-token loss improvements. Deceleration co-occurs with a sharp increase in ZSL.

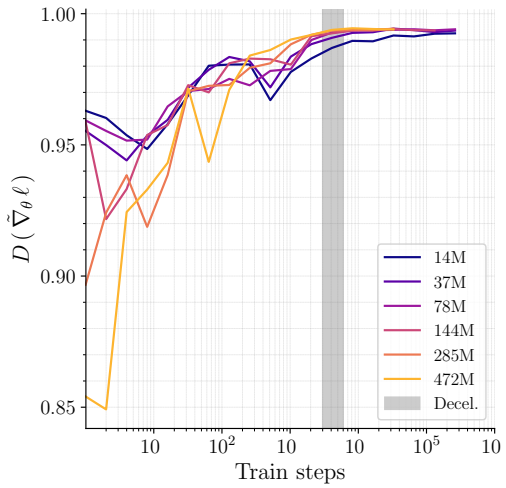


Fig. 4: Gradient destructive interference (averaged over parameters) converges to a maximum with deceleration.

following sections will demonstrate how, in terms of per-token loss behaviour, deceleration is driven by ZSL rather than the alternative explanation.

3.3 The Role of ZSL in Deceleration

Our framing of ZSL hypothesizes that deceleration is a result of destructive interference in loss improvements. In this section, we quantify the contribution of this destructive interference to overall loss improvements and show that it is indeed the main contributor to deceleration.

Quantifying the role of ZSL in deceleration. In terms of per-token loss improvements $\Delta\ell_i$, loss deceleration can occur for two (non mutually exclusive) reasons: (1) $\Delta\ell_i$ increasingly cancel one another out due to ZSL; or (2) $\Delta\ell_i$ increasingly shrink in magnitude across tokens. Destructive interference $D(\Delta\ell)$ in Eqn. 3 captures (1); while average magnitude $M(\Delta\ell)$ in Eqn. 5 captures (2):

$$M(\Delta\ell) = \frac{\sum_i |\Delta\ell_i|}{|\mathcal{D}|} \tag{5}$$

Importantly, we can express the absolute change in loss $|\Delta L|$ in terms of these two quantities:

$$|\Delta L| = \frac{|\sum_i \Delta\ell_i|}{|\mathcal{D}|} = M(\Delta\ell)(1 - D(\Delta\ell)) \tag{6}$$

If loss is monotonically decreasing, $|\Delta L|$ corresponds to overall loss improvements, such that we can effectively quantify and disentangle the relative contributions of increasing $D(\Delta\ell)$ from decreasing $M(\Delta\ell)$ in loss deceleration.

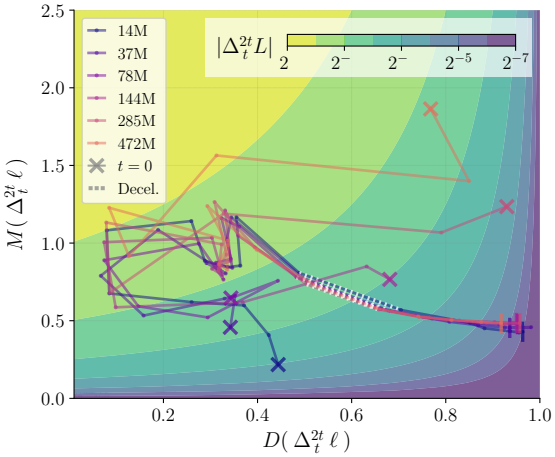


Fig. 5: Disentangling the relative contributions of increased ZSL ($D(\Delta\ell)$) and decreased token-level loss improvements ($M(\Delta\ell)$) towards decreased overall loss improvements ($|\Delta L|$). Model training trajectories, plotted with respect to these values, show that ZSL dominates decreases in $|\Delta L|$ after deceleration.

Showing ZSL is responsible for deceleration. In Fig. 5, we plot model training trajectories with respect to the terms in Eqn. 6. This allows us to visually determine and quantify how increases in $D(\Delta\ell)$ map to decreases in $|\Delta L|$; i.e. the contribution of ZSL to loss deceleration. Notably, we see that during and after deceleration, reductions in $|\Delta L|$ are largely attributable to changes in D rather than M. Concretely, we know from Eqn. 6 that the observed reduction in M during deceleration, from 0.75 to 0.5, corresponds to a 1.5x reduction in $|\Delta L|$. In contrast, the increase in D observed in that same period, from 0.5 to 0.95, corresponds to

a 10x reduction in loss improvements.

More generally, we see that as D increases and approaches 1.0, the required increase in M to maintain $|\Delta L|$ explodes such that ZSL effectively bottlenecks loss improvements and leads to deceleration. These results corroborate that, while decreasing magnitude across per-token loss improvements plays a role in deceleration, the effect of ZSL is almost an order of magnitude greater and effectively bottlenecks loss improvements.

3.4 The Role of Gradient Opposition in ZSL

Implicit to our framing of ZSL is the idea that destructive interference in loss improvements is a result of destructive interference in gradients. In this section, we make this assumption explicit and show how systematic gradient opposition, where $\vec{D}(\nabla_\theta \ell) \to \mathbf{1}$, fundamentally leads to ZSL under first-order training dynamics. We will then verify the validity of the first-order training dynamics assumption empirically. Lastly, we rule out an alternate explanation based on progressive sharpening.

Under first-order training dynamics

If weight updates $\Delta\theta$ are sufficiently small, first-order training dynamics apply where changes in overall or per-token losses are approximable via first-order Taylor expansion:

$$\begin{aligned} \tilde{\Delta}L &= \Delta\theta \cdot \nabla_\theta L = \textstyle\sum_i \tilde{\Delta}\ell_i \;/\; |\mathcal{D}| \\ \nabla_\theta L &= \textstyle\sum_i \nabla_\theta \ell_i \;/\; |\mathcal{D}| \\ \tilde{\Delta}\ell_i &= \Delta\theta \cdot \nabla_\theta \ell_i \end{aligned} \tag{7}$$

In such cases, ZSL is intrinsically a result of destructive interference in $\Delta\theta \cdot \nabla_\theta \ell_i$ across tokens:

$$\begin{aligned} D(\tilde{\Delta}\ell) &= 1 - \frac{|\sum_i \Delta\theta \cdot \nabla_\theta \ell_i|}{\sum_i |\Delta\theta \cdot \nabla_\theta \ell_i|} \\ &= 1 - \frac{|\Delta\theta \cdot \nabla_\theta L|}{\frac{1}{|\mathcal{D}|}\sum_i |\Delta\theta \cdot \nabla_\theta \ell_i|} \end{aligned} \tag{8}$$

Notably, $D(\tilde{\Delta}\ell)$ is a function of $\Delta\theta$ as well as $\nabla_\theta \ell_i$ and is not necessarily proportional to gradient destructive interference. In particular, we see in Fig. 6 that the effect of gradient destructive interference can be mitigated or exacerbated depending on its alignment with $\Delta\theta$. In some cases, $D(\tilde{\Delta}\ell)$ can be "induced" without destructive interference in $\nabla_\theta \ell$ if $\Delta\theta$ is aligned with differences in $\nabla_\theta \ell$.

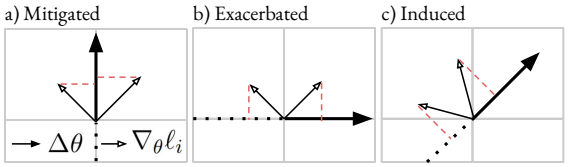


Fig. 6: $\tilde{\Delta}\ell_i$ is a projection of $\nabla_\theta \ell_i$ onto $\Delta\theta$, such that the effect of gradient destructive interference on ZSL can be **a) mitigated** or **b) exacerbated** depending on the alignment between $\vec{D}(\nabla_\theta \ell)$ and $\Delta\theta$. Moreover, destructive interference in loss improvements $D(\tilde{\Delta}\ell)$ can be **c) induced** even when gradients are not opposed, but their difference is aligned with $\Delta\theta$.

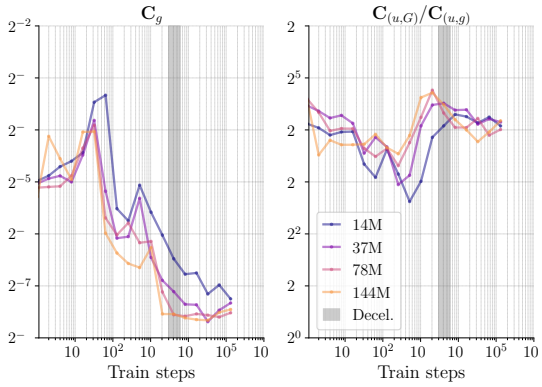


Fig. 7: $\mathbf{C}_g$ approaches 0 leading up to and during deceleration, while $\mathbf{C}_{(u,G)}/\mathbf{C}_{(u,g)}$ increases and remains relatively constant. As a result, we know from Eqn. 9 that $\mathbf{C}_g$ is the main contributor to $D(\tilde{\Delta}\ell)$.

In light of this, we attempt to disentangle the contribution of gradient opposition to $D(\tilde{\Delta}\ell)$ in Appendix B.1, decomposing Eqn. 8 into interpretable components :

$$D(\tilde{\Delta}\ell) = 1 - \mathbf{C}_g \cdot \frac{\mathbf{C}_{(u,G)}}{\mathbf{C}_{(u,g)}} \tag{9}$$

Each component captures "constructive" interference attributable to weight updates, per-example gradients and overall gradients (respectively denoted here as u, g and G for compactness). Constructive interference is simply one minus destructive interference. $\mathbf{C}_g \in [0, 1]$ measures (lack of) destructive interference in per-example gradients, taking into account the exacerbating or mitigating effects of $\Delta\theta$. In contrast, $\mathbf{C}_{(u,g)}, \mathbf{C}_{(u,G)} \in [0, 1]$ capture (lack of) destructive interference that is induced by projecting $\nabla_\theta \ell_i$ and $\nabla_\theta L$ onto $\Delta\theta$ independent of gradient opposition.

More specifically, $\mathbf{C}_g$ measures a weighted average of constructive interference in per-example gradients, across coordinates j:

$$\mathbf{C}_g = \sum_j W_j \cdot (1 - \vec{D}\,(\nabla_\theta \ell)_{[j]}) \qquad (10)$$
$$W_j \propto \sum_i |\Delta\theta_{[j]} \cdot \nabla_\theta \ell_{i,[j]}|$$

where $\sum_j W_j = 1$ and W_j captures the mitigating or exacerbating effect of $\Delta\theta$ on coordinate j. Notably, $\mathbf{C}_g \leq \max(\mathbf{1} - \vec{D}\,(\nabla_\theta \ell))$ by convexity, such that systematic gradient opposition where $\vec{D}\,(\nabla_\theta \ell) \to \mathbf{1}$ implies $\mathbf{C}_g \to 0$. We validate this empirically on a subset of models in Fig. 7 and confirm that $D(\tilde{\Delta}\ell) \to 1$ is indeed primarily attributable to $\mathbf{C}_g \to 0$; i.e. that destructive interference in loss improvements under first-order training dynamics is primarily attributable to destructive interference in gradients.

Testing the first-order dynamics assumption. Our hypothesis relies on the assumption that $\tilde{\Delta}\ell$ is a valid approximation of $\Delta\ell$ such that destructive interference in $\tilde{\Delta}\ell$ is reflective of destructive interference in $\Delta\ell$. To validate our hypothesis, we must therefore validate this assumption. However, computing $\nabla_\theta \ell_i$ and the corresponding $\tilde{\Delta}\ell_i = \Delta\theta \cdot \nabla_\theta \ell_i$ for each token is intractable.

Instead, to empirically measure $\tilde{\Delta}\ell$, we compute 1D cross-sections of per-token loss landscapes by evaluating model checkpoints along increments of their next weight update $\Delta\theta$, with $\theta(\alpha) = \theta + \alpha\Delta\theta/\|\Delta\theta\|$, $\alpha \in [-10, 10]$. This allows us to tractably measure $\tilde{\Delta}\ell$ as a linearization around $\alpha = 0$ where $\tilde{\Delta}\ell(\alpha) = \alpha\left(\frac{\ell_{\theta+\epsilon} - \ell_\theta}{\|\epsilon\|}\right)$. A sample of 1,000 such per-token loss landscapes is shown in Fig. 8, with the complete set in Appendix B.4. Generally, these appear linear in the vicinity of weight updates, suggesting that actual changes in per-token losses $\Delta\ell$ are well captured by their first-order approximation $\tilde{\Delta}\ell$.

However, to more quantifiably verify that this indeed is the case, we measure and plot the Pearson correlation coefficient between $\Delta\ell$ and $\tilde{\Delta}\ell$ throughout training in Fig. 9. We find strong correlation after deceleration where we observe destructive interference in loss improvements and gradients, validating our hypothesis by validating the underlying assumption of first-order dynamics on which our reasoning depends.

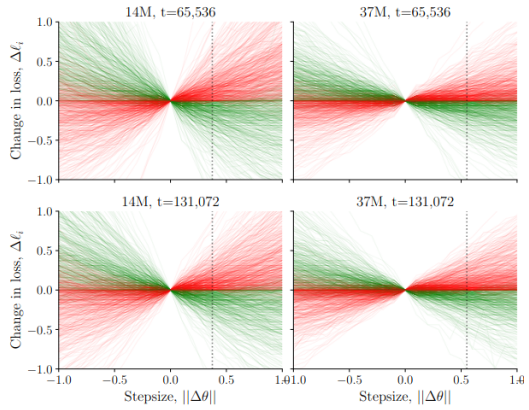


Fig. 8: Per-token loss landscapes at step t along $\Delta_t\theta$. Dashed vertical lines indicate $\theta_{t+1} = \theta_t + \Delta_t\theta$. Tokens which improve in loss after the update are indicated in green, and tokens which degrade are indicated in red

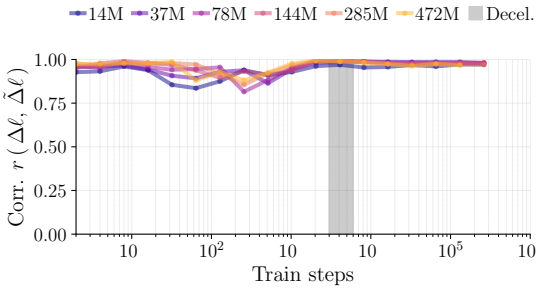


Fig. 9: Correlation between $\Delta\ell$ and it's first-order approximation $\tilde{\Delta}\ell$ is close to 1.0 after deceleration.

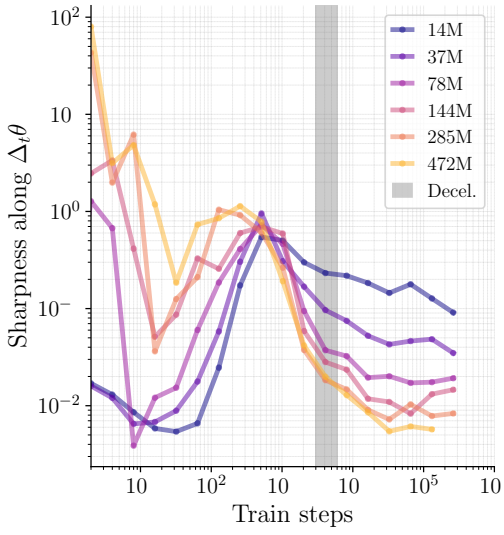


Fig. 10: Sharpness decreases with loss deceleration.

Ruling out the role of progressive sharpening. As an alternative explanation for ZSL, one might consider progressive sharpening (Cohen et al., 2022; Rosenfeld and Risteski, 2023) where $\Delta\theta$ might overshoot local minima for some examples but not others. Surprisingly, and perhaps counter to conventional wisdom, we observe in Appendix B.5 that loss landscapes instead become significantly flatter with deceleration; following an initial phase of high sharpness before deceleration.

To quantify this observation, we measure the sharpness of loss landscapes along update directions. Specifically, we fit a quadratic to the loss landscape cross-section, using the second order term as a measure of sharpness. In Fig. 10, we see the same trend, with sharpness peaking and immediately begin decreasing before deceleration. While the relationship between loss sharpness and zero-sum learning is outside the scope of this work, it appears there might be an interesting connection.

4 Explaining Scaling Improvements

In Section 2 we showed how scaling improves loss by mitigating loss deceleration, specifically by decreasing the loss L_d and step t_d at which it occurs, and increasing the subsequent log-log rate of loss improvement r_d (Table 1). Conversely, in Section 3 we proposed a mechanistic explanation of loss deceleration based on interactions at the levels of per-example loss improvements, and of per-example gradients. Specifically, we showed that loss deceleration is a transition in training dynamics characterized by the emergence of near-complete destructive interference in per-example gradients and loss improvements; i.e. ZSL. In this section, we will attempt to connect these findings, and shed light on *how* scaling mitigates deceleration based on the underlying mechanisms we identified.

Decomposing loss improvements. Similar to Section 3.4, we decompose the first-order Taylor expansion for changes in loss from Eqn. 7 into interpretable components that enable a finer-grained analysis of training dynamics, specifically the cosine similarity and L^2 norms of weight updates $\Delta\theta$ and gradients $\nabla_\theta L$:

$$\tilde{\Delta}L = \|\Delta\theta\|_2 \|\nabla_\theta L\|_2 \cos(\Delta\theta, \nabla_\theta L) \quad (11)$$

We show these values across training steps and model scales in Fig. 11, and will discuss their interpretation in the following sections.

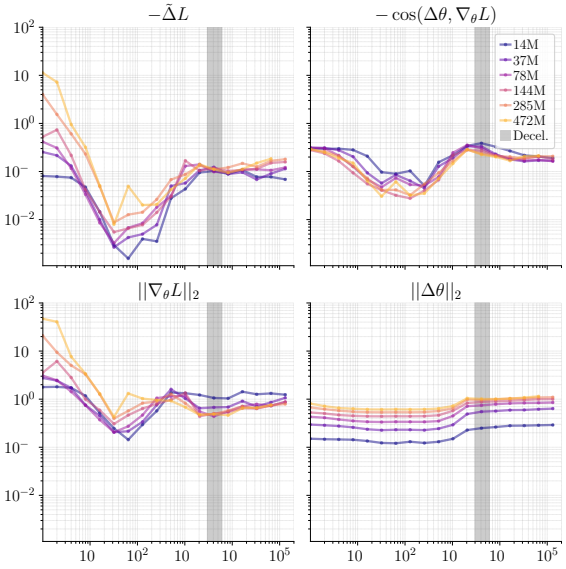


Fig. 11: First-order approximation of loss improvements with terms from Eqn. 11 plotted throughout training. Note that because $\log(\tilde{\Delta}L)$ is a sum of the log of its terms, the shared log-scale allows us to easily gauge how different terms contribute to changes in $\tilde{\Delta}L$.

Table 2: Scaling improvements in loss at deceleration L_d are established early during training.

Loss Improvement	$t = 32$	$t = 4096$	$t = 8192$
14M → 37M	0.76	0.43	0.45
37M → 78M	0.29	0.20	0.21
78M → 144M	0.15	0.12	0.12
144M → 285M	0.15	0.11	0.11
285M → 472M	0.05	0.06	0.07

4.1 Improving Loss Before Deceleration (L_d)

Surprisingly, we find in Table 2 that most of the scaling improvements in loss at deceleration L_d are already established by step $t = 32$.

From Eqn. 11 and Fig. 11, the underlying reason becomes apparent. Scaling models improves $\tilde{\Delta}L$ primarily by improving gradient norms $\|\nabla_\theta L\|_2$ in the beginning of training. Beyond $t = 32$, the effects of scaling become less significant, with improvements in $\tilde{\Delta}L$ and $\|\nabla_\theta L\|_2$ orders of magnitude smaller and eventually reversed leading up to deceleration. In contrast, scaling degrades gradient-update alignment $-\cos(\Delta\theta, \nabla_\theta L)$, and results in consistent but relatively insignificant improvements in $\|\Delta\theta\|_2$. These effects are trivially explained by an increased number of parameters and appear unrelated to deceleration, however it remains an open question how similar effects can be achieved independent of scale.

4.2 Improving Loss After Deceleration (r_d)

We see in Fig. 3 that post-deceleration ZSL is mitigated by scaling model size, which we know results in greater loss improvements from Eqn. 6 that can explain how scaling improves r_d. Unfortunately, the way in which scaling reduces ZSL after deceleration is not as immediately obvious.

We see in Fig. 4 that gradient destructive interference (averaged across parameters) actually becomes more pronounced with larger models. However, 99% destructive interference in a 14M-dimensional gradient does not have the same effect as in a 144M-dimensional gradient. In particular, the latter will have more degrees of freedom along which shared gradient directions can exist between tokens. Indeed, we find in Fig. 12 that, especially after deceleration, larger models have more parameters with lower destructive interference. This can explain why larger models have lower ZSL after deceleration, and thus improved r_d.

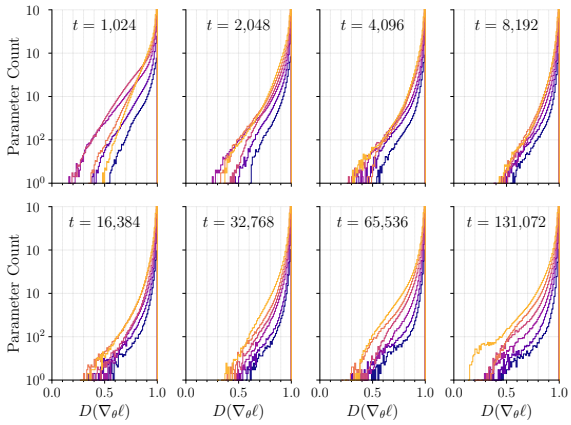


Fig. 12: Histograms of gradient destructive interference. Deceleration happens between steps 4096 and 8192. Note that model size legend is consistent with other plots but omitted for space.

5 Conclusion and outlook

In this work we proposed and validated a mechanistic explanation of scaling laws grounded in training dynamics. Specifically, we identified loss deceleration as a novel transition in training dynamics that can explain scaling improvements in quantifiable but interpretable terms, such that an explanation of deceleration becomes an explanation of scaling improvements. To this end, we proposed zero-sum learning as the mechanism underlying deceleration, validating these against alternate hypotheses with empirical and theoretical analyses. Lastly, we revisit scaling improvements from the perspective of these mechanisms and show how scaling improves loss by mitigating zero-sum learning and more specifically, systematic gradient opposition.

Our findings suggest that these could potentially be mitigated directly to improve loss independent of scale, laying a foundation for future research. Furthermore, studying per-example gradient dynamics as in our work is an under-explored area of research that can shed new light on learning dynamics, scaling, and generalization more broadly.

Limitations

Comprehensiveness of experimental settings. Scaling laws are a general phenomenon observed across tasks, model architectures, parameters, and evaluation measures. However, this work only considers the scaling of cross-entropy loss with model size in transformer-based language models on typical webscale text. While we replicate our experiments across several ablations in Appendix C, we do not generalize our findings to different settings beyond language modeling. While this lies beyond the scope of our original research question and the prior works on which we build, verifying how our findings generalize across different settings is an important area of future work.

Accounting for gradient opposition in both $M(\Delta\ell)$ and $D(\Delta\ell)$. In Section 3.3 and Eqn. 6 we showed that destructive interference in loss improvements $D(\Delta\ell)$ is primarily responsible for deceleration. In contrast, decreases in average per-token loss improvements $M(\Delta\ell)$ played an non-negligible but less significant role. However, our analysis of gradient opposition (Section 3.4) only considers its effect on $D(\Delta\ell)$, while it likely also has an effect on $M(\Delta\ell)$ via its effect on optimizer steps $\Delta\theta$. However, these effects are likely highly dependent on the optimizer and its configuration, and likely not generalizable in the scope of our research question; hence why we chose to abstract away $\Delta\theta$ in our analysis. Nevertheless, this a salient gap in our analysis that should be further explored.

Reconciling single step and multi step training dynamics. The connection between the behaviour of gradients and loss can be made more precise. In particular, our gradient analysis only reflects single-step training dynamics, while ZSL and loss improvements appear to depend on interactions across multiple optimization steps (see Ap-

pendix B.2). Understanding the effect of multi step interaction is a natural next step for this research.

Negative societal impacts or ethical concerns. Our work focuses on understanding existing and well-established methods, and does not meaningfully contribute to any negative societal impacts or ethical concerns beyond what is typically associated with language modeling research. In principle, by focusing our analysis on a single metric (cross-entropy loss), this could lead to over-optimizing that metric at the expense of other real-world concerns. While this work is at too early a stage for this to pose a meaningful risk, it is important to keep in mind as a limitation in interpreting our findings and building new methods on top of them.

Acknowledgments

We would like to acknowledge support from IVADO and the Canada First Research Excellence Fund [E.L.]; from the Canada CIFAR AI Chair Program and the Canada Excellence Research Chairs Program [I.R.]; and from the NSERC post-graduate doctoral (PGS D) scholarship as well as the FRQNT Doctoral (B2X) doctoral scholarship [A.M.]. We would also like to thank Mila (mila.quebec) and its IDT team for providing and supporting the computing resources used in this work; as well as the Capital One AGI Foundations Team and in particular Ashwinee Panda for the helpful discussions and feedback.

References

Zeyuan Allen-Zhu and Yuanzhi Li. 2023. Towards Understanding Ensemble, Knowledge Distillation and Self-Distillation in Deep Learning. *arXiv preprint*. ArXiv:2012.09816 [cs, math, stat].

Alexander B. Atanasov, Jacob A. Zavatone-Veth, and Cengiz Pehlevan. 2024. Scaling and renormalization in high-dimensional regression. *CoRR*. CoRR 2024.

Yasaman Bahri, Ethan Dyer, Jared Kaplan, Jaehoon Lee, and Utkarsh Sharma. 2024. Explaining neural scaling laws. *Proceedings of the National Academy of Sciences*, 121(27):e2311878121. PNAS 2024.

Blake Bordelon, Alexander Atanasov, and Cengiz Pehlevan. 2024. A Dynamical Model of Neural Scaling Laws. ICML 2024.

Ethan Caballero, Kshitij Gupta, Irina Rish, and David Krueger. 2023. Broken Neural Scaling Laws. *arXiv preprint*. ArXiv:2210.14891 [cs].

Jeremy M. Cohen, Simran Kaur, Yuanzhi Li, J. Zico Kolter, and Ameet Talwalkar. 2022. Gradient Descent on Neural Networks Typically Occurs at the Edge of Stability. *arXiv preprint*. ArXiv:2103.00065 [cs, stat].

Katie Everett, Lechao Xiao, Mitchell Wortsman, Alexander A. Alemi, Roman Novak, Peter J. Liu, Izzeddin Gur, Jascha Sohl-Dickstein, Leslie Pack Kaelbling, Jaehoon Lee, and Jeffrey Pennington. 2024. Scaling Exponents Across Parameterizations and Optimizers. *arXiv preprint*. ArXiv:2407.05872 [cs] version: 1.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. The Pile: An 800GB Dataset of Diverse Text for Language Modeling. *arXiv preprint arXiv:2101.00027*.

Stuart Glennan and Phyllis Illari, editors. 2017. *The Routledge Handbook of Mechanisms and Mechanical Philosophy*. Routledge, London.

Nathan Godey, Éric de la Clergerie, and Benoît Sagot. 2024. Why do small language models underperform? Studying Language Model Saturation via the Softmax Bottleneck. *arXiv preprint*. ArXiv:2404.07647 [cs].

Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Harsh Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, Shane Arora, David Atkinson, Russell Authur, Khyathi Raghavi Chandu, Arman Cohan, Jennifer Dumas, Yanai Elazar, Yuling Gu, Jack Hessel, Tushar Khot, William Merrill, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew E. Peters, Valentina Pyatkin, Abhilasha Ravichander, Dustin Schwenk, Saurabh Shah, Will Smith, Emma Strubell, Nishant Subramani, Mitchell Wortsman, Pradeep Dasigi, Nathan Lambert, Kyle Richardson, Luke Zettlemoyer, Jesse Dodge, Kyle Lo, Luca Soldaini, Noah A. Smith, and Hannaneh Hajishirzi. 2024. OLMo: Accelerating the Science of Language Models. *arXiv preprint*. ArXiv:2402.00838.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack W. Rae, and Laurent Sifre. 2022. Training compute-optimal large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, pages 30016–30030, Red Hook, NY, USA. Curran Associates Inc.

Marcus Hutter. 2021. Learning Curve Theory. *arXiv preprint*. ArXiv:2102.04074 [cs, stat].

Alexander Hägele, Elie Bakouch, Atli Kosson, Loubna Ben Allal, Leandro Von Werra, and Martin Jaggi. 2024. Scaling Laws and Compute-Optimal Training Beyond Fixed Training Durations.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. 2024. Muon: An optimizer for hidden layers in neural networks.

David Michael Kaplan and Carl F. Craver. 2011. The Explanatory Force of Dynamical and Mathematical Models in Neuroscience: A Mechanistic Perspective*. *Philosophy of Science*, 78(4):601–627. Publisher: [The University of Chicago Press, Philosophy of Science Association].

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling Laws for Neural Language Models. *arXiv preprint*. ArXiv:2001.08361 [cs, stat].

Andrej Karpathy. 2022. NanoGPT. Publication Title: GitHub repository.

Kaizhao Liang, Lizhang Chen, Bo Liu, and Qiang Liu. 2025. Cautious Optimizers: Improving Training with One Line of Code. *arXiv preprint*. ArXiv:2411.16085 [cs].

Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. 2021. Conflict-Averse Gradient Descent for Multi-task learning. In *Advances in Neural Information Processing Systems*, volume 34, pages 18878–18890. Curran Associates, Inc.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*.

Ian Magnusson, Akshita Bhagia, Valentin Hofmann, Luca Soldaini, Ananya Harsh Jha, Oyvind Tafjord, Dustin Schwenk, Evan Pete Walsh, Yanai Elazar,

Kyle Lo, Dirk Groeneveld, Iz Beltagy, Hannaneh Hajishirzi, Noah A. Smith, Kyle Richardson, and Jesse Dodge. 2023. Paloma: A Benchmark for Evaluating Language Model Fit. *arXiv preprint*. ArXiv:2312.10523.

Max Marion, Ahmet Ustun, Luiza Pozzobon, Alex Wang, Marzieh Fadaee, and Sara Hooker. 2023. When Less is More: Investigating Data Pruning for Pretraining LLMs at Scale. *arXiv preprint*. ArXiv:2309.04564 [cs].

Eric Michaud, Ziming Liu, Uzay Girit, and Max Tegmark. 2023. The Quantization Model of Neural Scaling. *Advances in Neural Information Processing Systems*, 36:28699–28722. NeurIPS 2023.

Andrei Mircea, Ekaterina Lobacheva, and Irina Rish. 2024. Gradient Dissent in Language Model Training and Saturation.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. 2022. In-context Learning and Induction Heads. *arXiv preprint*. ArXiv:2209.11895 [cs].

Matteo Pagliardini, Pierre Ablin, and David Grangier. 2024. The AdEMAMix Optimizer: Better, Faster, Older. *arXiv preprint*. ArXiv:2409.03137 [cs].

Giambattista Parascandolo, Alexander Neitz, Antonio Orvieto, Luigi Gresele, and Bernhard Schölkopf. 2020. Learning explanations that are hard to vary.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners.

David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. 2024. Mixture-of-Depths: Dynamically allocating compute in transformer-based language models. *arXiv preprint*. ArXiv:2404.02258 [cs].

Elan Rosenfeld and Andrej Risteski. 2023. Outliers with Opposing Signals Have an Outsized Effect on Neural Network Optimization. *arXiv preprint*. ArXiv:2311.04163 [cs, stat].

Utkarsh Sharma and Jared Kaplan. 2022. Scaling Laws from the Data Manifold Dimension. *Journal of Machine Learning Research*, 23(9):1–34. JMLR 2022.

Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari Morcos. 2022. Beyond neural scaling laws: beating power law scaling via data pruning. In *Advances in Neural Information Processing Systems*, volume 35, pages 19523–19536. Curran Associates, Inc.

Michael P. H. Stumpf and Mason A. Porter. 2012. Critical Truths About Power Laws. *Science*, 335(6069):665–666. Publisher: American Association for the Advancement of Science.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, Anton Tsitsulin, Nino Vieillard, Piotr Stanczyk, Sertan Girgin, Nikola Momchev, Matt Hoffman, Shantanu Thakoor, Jean-Bastien Grill, Behnam Neyshabur, Olivier Bachem, Alanna Walton, Aliaksei Severyn, Alicia Parrish, Aliya Ahmad, Allen Hutchison, Alvin Abdagic, Amanda Carl, Amy Shen, Andy Brock, Andy Coenen, Anthony Laforge, Antonia Paterson, Ben Bastian, Bilal Piot, Bo Wu, Brandon Royal, Charlie Chen, Chintu Kumar, Chris Perry, Chris Welty, Christopher A. Choquette-Choo, Danila Sinopalnikov, David Weinberger, Dimple Vijaykumar, Dominika Rogozińska, Dustin Herbison, Elisa Bandy, Emma Wang, Eric Noland, Erica Moreira, Evan Senter, Evgenii Eltyshev, Francesco Visin, Gabriel Rasskin, Gary Wei, Glenn Cameron, Gus Martins, Hadi Hashemi, Hanna Klimczak-Plucińska, Harleen Batra, Harsh Dhand, Ivan Nardini, Jacinda Mein, Jack Zhou, James Svensson, Jeff Stanway, Jetha Chan, Jin Peng Zhou, Joana Carrasqueira, Joana Iljazi, Jocelyn Becker, Joe Fernandez, Joost van Amersfoort, Josh Gordon, Josh Lipschultz, Josh Newlan, Ju-yeong Ji, Kareem Mohamed, Kartikeya Badola, Kat Black, Katie Millican, Keelin McDonell, Kelvin Nguyen, Kiranbir Sodhia, Kish Greene, Lars Lowe Sjoesund, Lauren Usui, Laurent Sifre, Lena Heuermann, Leticia Lago, Lilly McNealus, Livio Baldini Soares, Logan Kilpatrick, Lucas Dixon, Luciano Martins, Machel Reid, Manvinder Singh, Mark Iverson, Martin Görner, Mat Velloso, Mateo Wirth, Matt Davidow, Matt Miller, Matthew Rahtz, Matthew Watson, Meg Risdal, Mehran Kazemi, Michael Moynihan, Ming Zhang, Minsuk Kahng, Minwoo Park, Mofi Rahman, Mohit Khatwani, Natalie Dao, Nenshad Bardoliwalla, Nesh Devanathan, Neta Dumai, Nilay Chauhan, Oscar Wahltinez, Pankil Botarda, Parker Barnes, Paul Barham, Paul Michel, Pengchong Jin, Petko Georgiev, Phil Culliton, Pradeep Kuppala, Ramona Comanescu, Ramona Merhej, Reena Jana, Reza Ardeshir Rokni, Rishabh Agarwal, Ryan Mullins, Samaneh Saadat, Sara Mc Carthy, Sarah Perrin, Sébastien M. R. Arnold, Sebastian Krause, Shengyang Dai, Shruti Garg, Shruti Sheth, Sue Ronstrom, Susan Chan, Timothy Jordan, Ting Yu, Tom Eccles, Tom Hennigan, Tomas Kocisky, Tulsee Doshi, Vihan Jain, Vikas Yadav, Vilobh Meshram, Vishal Dharmadhikari, Warren Barkley, Wei Wei, Wenming Ye, Woohyun Han, Woosuk Kwon, Xiang Xu, Zhe Shen, Zhitao Gong, Zichuan Wei, Victor Cotruta, Phoebe Kirk, Anand Rao, Minh Giang, Ludovic Peran, Tris Warkentin, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, D. Sculley, Jeanine Banks, Anca Dragan, Slav Petrov,

Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Sebastian Borgeaud, Noah Fiedel, Armand Joulin, Kathleen Kenealy, Robert Dadashi, and Alek Andreev. 2024. Gemma 2: Improving Open Language Models at a Practical Size. *arXiv preprint*. ArXiv:2408.00118 [cs].

Howe Tissue, Venus Wang, and Lu Wang. 2024. Scaling Law with Learning Rate Annealing. *arXiv preprint*. ArXiv:2408.11029 [cs].

Tom Viering and Marco Loog. 2022. The Shape of Learning Curves: a Review. *arXiv preprint*. ArXiv:2103.10948 [cs].

Kaiyue Wen, Zhiyuan Li, Jason Wang, David Hall, Percy Liang, and Tengyu Ma. 2024. Understanding Warmup-Stable-Decay Learning Rates: A River Valley Loss Landscape Perspective. ArXiv:2410.05192 [cs, stat].

Yuki Yoshida and Masato Okada. 2020. Data-Dependence of Plateau Phenomenon in Learning with Neural Network — Statistical Mechanical Analysis. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(12):124013. ArXiv:2001.03371 [cs, stat].

Ashkan Yousefpour, Igor Shilov, Alexandre Sablayrolles, Davide Testuggine, Karthik Prasad, Mani Malek, John Nguyen, Sayan Ghosh, Akash Bharadwaj, Jessica Zhao, Graham Cormode, and Ilya Mironov. 2021. Opacus: User-Friendly Differential Privacy Library in PyTorch.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient Surgery for Multi-Task Learning. *arXiv preprint*. ArXiv:2001.06782 [cs, stat].

A Methodology

A.1 Language model pretraining

For our experiments, we adapt the OLMo codebase (licensed under Apache-2.0) and train variants of OLMo with the publicly available training dataset of OLMo-7B-0724 (Groeneveld et al., 2024). Model dimensions and learning rates are based on (Kaplan et al., 2020) and shown in Table 3, labeled with (rounded) total parameter counts. For pretraining, we again adapt the experimental setup of (Kaplan et al., 2020), training with a batch size of 0.5M tokens for 2^{18} steps. However, instead of a cosine learning rate decay, we adopt the trapezoidal learning rate schedule from (Hägele et al., 2024) with a learning rate warmup to the values in Table 3 in the first 2,000 steps and no cooldown in the 2^{18} steps considered. Note that the OLMo-1B and OLMo-7B models are those trained by (Groeneveld et al., 2024) and could not included in our analysis of ZSL because of insufficient checkpointing frequency before deceleration.

Code and artefacts Model and optimizer checkpoints, and logs across training are available at `https://github.com/mirandrom/zsl` under an Apache-2.0 license to enable future research in this direction. In our experiments, we used a variety of computational resources which are recorded in the logs we make available. Generally, we performed distributed training 4-32 L40 GPUs or 4 H100 GPUs, with smaller models pretraining requiring on the order of 10 GPU hours, and the largest 472M requiring on the order of 1000 GPU hours.

Language model analyses During training, we checkpoint the model and optimizer every 2^i steps with $i \in [0, 18]$. Our analyses of ZSL and gradient opposition are done on these checkpoints after pretraining. Methodological details regarding e.g. precision or batch size are kept consistent with pretraining to obtain representative results. All of our evaluations are conducted on the C4 validation set from (Magnusson et al., 2023), using the tokenizer from (Groeneveld et al., 2024), consistent with pretraining.

Table 3: Model and Optimizer Parameters for Different Runs

Model size	**14M**	**37M**	**78M**	**144M**	**285M**	**472M**	**OLMo-1B**	**OLMo-7B**
`d_model`	256	512	768	1024	1536	2048	2048	4096
`mlp_dim`	256	512	768	1024	1536	2048	16384	22016
`n_heads`	4	8	12	16	16	16	16	32
`n_layers`	4	8	12	16	16	16	16	32
`peak_lr`	1.3E-3	9.7E-4	8.0E-4	6.8E-4	5.7E-4	4.9E-4	4.0E-4	3.0E-4
`warmup`	2,000	2,000	2,000	2,000	2,000	2,000	2,000	5,000

A.2 Additional Details on Fitting BNSL

Fitting We adapt the methodology for fitting Eqn. 1 published by Caballero et al. (2023) at https://github.com/ethancaballero/broken_neural_scaling_laws. We include the code implementation below. Empirically, we had to implement the following changes to improve stability:

- Assume $a = 0$ and remove it from the optimization procedure.
- Fit the function in log-log space instead of manually scaling b and d_1. Note that datapoints sampled uniformly along x will result in a data imbalance when fitting in log-log space; to mitigate this we also subsample datapoints uniformly in log space.
- Estimate initial parameters instead of running a bruteforce gridsearch.

```
import numpy as np
import scipy

def log_1b_bnsl(xlog, b, c0, c1, d1log, f1):
    ylog_pred = np.log(b) - c0*xlog - (c1*f1)*np.log(1+np.exp((xlog-d1log)/f1))
    return ylog_pred

def fit_1b_bnsl(x: np.ndarray, y: np.ndarray, d1_est: float = 6000):
    # initialize parameters with reasonable values (for stability)
    d1log = np.log(d1_est)
    xlog = np.log(x)
    ylog = np.log(y)
    d1_idx = np.argmin(np.abs(xlog - d1log))
    c0 = -np.mean((ylog[0:d1_idx] - ylog[1:d1_idx+1]) \
            / (xlog[0:d1_idx] - xlog[1:d1_idx+1]))
    c1 = -np.mean((ylog[d1_idx:-2] - ylog[d1_idx+1:-1]) \
            / (xlog[d1_idx:-2] - xlog[d1_idx+1:-1]))
    c1 = c1 - c0
    b = ylog[0] + c0*xlog[0]

    # fit parameters with scipy
    p0 = [b, c0, c1, d1log, 0.3]
    popt, pcov = scipy.optimize.curve_fit(
        log_1b_bnsl,
        xlog, ylog,
        p0=p0,
        method='dogbox',
    )

    return popt, pcov
```

Code 1: Code for fitting one-break BNSL.

Smoothing The loss curves we fit are batch losses logged at every step during training. Because training is single-epoch, i.e. online, these losses are effectively a noisy measurement of the true validation loss. However, we found that this noise (characterized by oscillations in loss at too-small timescales) leads to severe instability with the original methodology published by (Caballero et al., 2023). To smooth these curves, we use LSMA_k, a logarithmic variant of the simple moving average that we found to work well for fitting noisy log-log loss curves with high fidelity. Notably, LSMA naturally handles the increasing timescales at which loss oscillations occur as number of training steps increase. We found $k = 1.2$ to work sufficiently well as shown in Fig. 13.

$$\mathrm{LSMA}_k\left(L_t\right) = \frac{1}{t - p(t)} \sum_{s=p(t)}^{t} L_s \quad , \qquad p(t) = \mathrm{floor}(t/k)$$

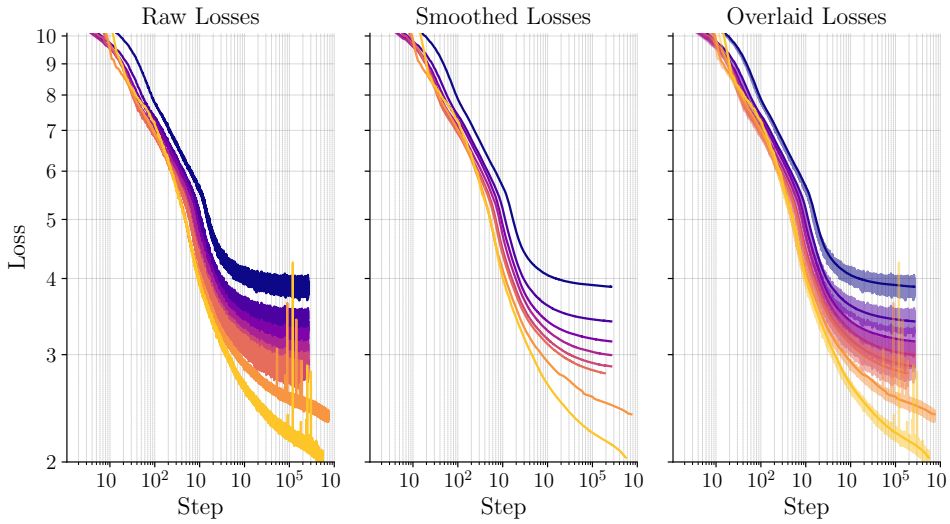


Fig. 13: $\mathrm{LMSA}_k\left(L_t\right)$ smoothing with $k = 1.2$.

Results and validation We report the resulting parameters and error measurements from fitting Eqn. 1 in Table 4, finding that parameter standard deviation is typically on the order of 1%, while root standard log error (RSLE) is on the order of 0.01, comparable with values reported by Caballero et al. (2023). These results suggest that loss deceleration is reliably measurable with BNSL.

Table 4: BNSL parameters and root-standerd log error if resulting fit (RSLE).

Model	b	c_0	c_1	$\log(d_1)$	f_1	RSLE
14M	18.42 ± 0.16	0.17 ± 0.00	-0.16 ± 0.00	8.68 ± 0.02	0.20 ± 0.03	0.011
37M	19.64 ± 0.23	0.20 ± 0.00	-0.18 ± 0.00	8.68 ± 0.03	0.24 ± 0.03	0.015
78M	20.66 ± 0.25	0.21 ± 0.00	-0.19 ± 0.00	8.69 ± 0.03	0.29 ± 0.03	0.014
144M	20.31 ± 0.26	0.21 ± 0.00	-0.19 ± 0.00	8.71 ± 0.03	0.34 ± 0.03	0.015
285M	20.85 ± 0.30	0.22 ± 0.00	-0.20 ± 0.00	8.57 ± 0.03	0.44 ± 0.03	0.013
472M	21.16 ± 0.32	0.23 ± 0.00	-0.19 ± 0.00	8.44 ± 0.03	0.39 ± 0.04	0.014
OLMo-1B	25.97 ± 0.38	0.27 ± 0.00	-0.23 ± 0.00	8.22 ± 0.03	0.76 ± 0.02	0.008
OLMo-7B	27.49 ± 0.48	0.28 ± 0.00	-0.22 ± 0.00	8.44 ± 0.04	0.76 ± 0.03	0.008

A.3 Additional Details on Computing Gradient Opposition

Measuring a tractable proxy for per-token gradient destructive interference

While computing true per-token gradients is typically intractable, we can tractably compute destructive interference between gradients for each token *features* rather than for each token *loss*. Concretely, for any module $\mathcal{M}(x) = y,\ \mathcal{M} : S \times D_1 \mapsto S \times D_2$ with sequence length S and hidden dimensions D_1, D_2; we define $\nabla_\theta \tilde{\ell}_i = \sum_{\mathcal{M}} (\delta L / \delta y_i)(\delta y_i / \delta \theta_{\mathcal{M}})$ for the i^{th} token in a sequence. In the PyTorch code block below, we illustrate how the backward pass of a linear layer with weights W is modified to compute gradient destructive interference across samples $\frac{\delta L}{\delta W} = \sum_i \frac{\delta L}{\delta y_i} \frac{\delta y_i}{\delta W}$, similar to Yousefpour et al. (2021):

```
import torch
from torch import nn, autograd, functional as F

def compute_gdi(W: nn.Parameter):
  gdi = 1 - W.sum_grads.abs()/W.sum_abs_grads
  return gdi.mean()


class GDILinearFunction(autograd.Function):
  @staticmethod
  def forward(ctx, x, W):
    ctx.save_for_backward(x, W)
    y = F.linear(x,W)
    return y

  @staticmethod
  def backward(ctx, dLdy):
    x, W = ctx.saved_tensors
    if ctx.needs_input_grad[1]:

      # instantiate metrics if not present
      if not hasattr(W, 'sum_grads'):
        W.sum_grads = torch.zeros_like(W)
        W.sum_abs_grads = torch.zeros_like(W)

      # accumulate sum of gradients
      W.sum_grads.add_(
        torch.einsum(
          'B...d,B...p->pd', x, dLdy
        )
      )
      # accumulate sum of absolute gradients
      W.sum_abs_grads.add_(
        torch.einsum(
          'B...d,B...p->pd', x.abs(), dLdy.abs()
        )
      )
    # compute and return input gradient for backprop
    if ctx.needs_input_grad[0]:
      dLdx = torch.einsum(
        'B...p,pd->B...d', dLdy, W
      )
    else:
      dLdx = None

    return dLdx, None
```

Code 2: Illustrative example computing gradient destructive interference in PyTorch

Consistency of findings between proxy and true measure of gradient destructive interference Computing true per-token gradients is feasible, just intractable in most cases. In particular, it requires doing one backward pass per token in a sequence, effectively resulting in 1000x increase in costs. We were nevertheless able to measure true gradient destructive interference for models up to 144M so as to verify the validity of the results obtained in Fig. 4 with out proxy measure. We find that qualitatively, both our proxy and true measure behave consistently, converging towards complete destructive interference during deceleration for non-embedding parameters. However, we find that our proxy systematically *underestimates* gradient destructive interference. Understanding the discrepancies between the proxy and true measures of gradient destructive interference, especially in terms of how they relate to zero-sum learning, remains an open question.

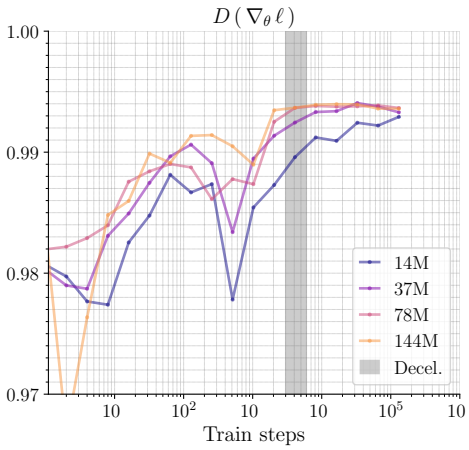


Fig. 14: Actual destructive interference in per-example gradient behaves consistently with our proxy in Fig. 4, but is systematically underestimated by it.

B Additional Results (Analyses)

B.1 Decomposing First-Order Training Dynamics

Notation and goal
Generalizing Eqns. 3,5,6 to any empirical average $X = \sum_i^N x_i/N$, we have:

$$\text{Destructive interference:} \qquad D_i^N(x_i) = 1 - \frac{|\sum_i^N x_i|}{\sum_i^N |x_i|} \in [0,1] \tag{12}$$

$$\text{Constructive interference:} \qquad C_i^N(x_i) = \frac{|\sum_i^N x_i|}{\sum_i^N |x_i|} = 1 - D_i^N(x_i) \in [0,1] \tag{13}$$

$$\text{Average magnitude:} \qquad M_i^N(x_i) = \frac{1}{N}\sum_i^N |x_i| \tag{14}$$

$$\text{Absolute empirical average:} \qquad |X| = M_i^N(x_i)\, C_i^N(x_i) \tag{15}$$

Our goal is to understand and quantify how destructive interference between per-example loss improvements $D_i^N(\Delta\ell_i)$ (Eqn. 16) is affected by gradient opposition (i.e. destructive interference between per-example gradients $D_i^N(g_i)$, Eqn. 17). Recall that the overall change in loss for examples $i \in [1 \dots N]$ is $\Delta L = \frac{1}{N}\sum_i^N \Delta\ell_i$. Similarly, the overall gradient is $\nabla_\theta L = \frac{1}{N}\sum_i \nabla_\theta \ell_i \in \mathbb{R}^M$ where M is number of parameters. For compactness, we denote $\nabla_\theta L$ as G and $\nabla_\theta \ell_i$ as g_i, using $G_{[j]}$ and $g_{i[j]}$ to indicate the scalar value of a gradient at coordinate j.

$$D_i^N(\nabla_\theta \ell_i) = 1 - |\sum_i^N \nabla_\theta \ell_i| \,/\, \sum_i^N |\nabla_\theta \ell_i| \tag{16}$$

$$D_i^N(g_i) = 1 - |\sum_i^N g_i| \,/\, \sum_i^N |g_i| \tag{17}$$

$$D_i^N(g_i)_{[j]} = D_i^N(g_{i[j]}) = 1 - |\sum_i^N g_{i[j]}| \,/\, \sum_i^N |g_{i[j]}| \tag{18}$$

Unless otherwise stated, moving forward i indexes the N examples used in computing the empirical average change in loss or gradient, and j indexes the M learnable model parameters flattened into a vector.

Change in loss under first-order training dynamics
Under first-order training dynamics, weight updates $\Delta\theta$ are sufficiently small such that changes in loss (per-example $\Delta\ell_i$, and overall ΔL) are approximable by first-order Taylor expansions $\tilde{\Delta}\ell_i$, $\tilde{\Delta}L$:

$$\tilde{\Delta}\ell_i = \langle \Delta\theta, g_i \rangle = \sum_j^M \Delta\theta_{[j]} \cdot g_{i[j]} \tag{19}$$

$$\begin{aligned} \tilde{\Delta}L &= \langle \Delta\theta, G \rangle = \textstyle\sum_j^M \Delta\theta_{[j]} \cdot G_{[j]} && (20) \\ &= \langle \Delta\theta, \tfrac{1}{N}\textstyle\sum_i^N g_i \rangle = \textstyle\sum_j^M \sum_i^N \tfrac{1}{N} \cdot \Delta\theta_{[j]} \cdot g_{i[j]} \\ &= \tfrac{1}{N}\textstyle\sum_i^N \tilde{\Delta}\ell_i = \sum_i^N \sum_j^M \tfrac{1}{N} \cdot \Delta\theta_{[j]} \cdot g_{i[j]} && (21) \end{aligned}$$

In such cases, ZSL is intrinsically a result of destructive interference in $\tilde{\Delta}\ell_i = \langle \Delta\theta, \nabla_\theta \ell_i \rangle$:

$$D_i^N(\tilde{\Delta}\ell_i) = 1 - \frac{\left|\sum_i^N \langle \Delta\theta, g_i \rangle\right|}{\sum_i^N |\langle \Delta\theta, g_i \rangle|} = 1 - \frac{\left|\sum_i^N \sum_j^M \Delta\theta_{[j]} \cdot g_{i[j]}\right|}{\sum_i^N \left|\sum_j^M \Delta\theta_{[j]} \cdot g_{i[j]}\right|} \tag{22}$$

Note that Eqn. 22 does not imply that gradient opposition necessarily results in ZSL. For instance, directions of high opposition in per-token gradients g_i may be orthogonal to a weight update $\Delta\theta$, such that they are nullified when g_i is projected onto $\Delta\theta$. Conversely, two gradients g_a, g_b with no destructive interference may result in ZSL if e.g. $\Delta\theta$ is aligned with $g_a - g_b$. In light of this, we want to disentangle ZSL in Eqn. 8 that is attributable to update-gradient alignment independent of gradient opposition, from ZSL attributable to gradient opposition specifically.

Isolating the role of gradient opposition

Instead of destructive interference (Eqn. 22), we can equivalently consider and isolate the effect of gradient opposition in constructive interference (Eqn. 23) based on its identity as $1 - D$ (Eqn. 13).

$$C_i^N(\tilde{\Delta}\ell_i) = 1 - D_i^N(\tilde{\Delta}\ell_i) = \frac{\left|\sum_i^N \sum_j^M \Delta\theta_{[j]} \cdot g_{i[j]}\right|}{\sum_i^N \left|\sum_j^M \Delta\theta_{[j]} \cdot g_{i[j]}\right|} \tag{23}$$

For compactness, we will denote $p_{i[j]} = \Delta\theta_{[j]} \cdot g_{i[j]}$ and $q_{[j]} = \Delta\theta_{[j]} \cdot G_{[j]}$. Intuitively, $p_{i[j]}$ captures the contribution of example i and parameter j to the first-order change in loss $\tilde{\Delta}L = \frac{1}{N}\sum_i^N \sum_j^M p_{i[j]}$. Note that g is not normalized by number of examples, i.e. $G_{[j]} = \frac{1}{N}\sum_i^N g_{i[j]}$ and $q_{[j]} = \frac{1}{N}\sum_i^N p_{i[j]}$.

Our goal is to isolate the effect of gradient opposition in Eqn. 23. To this end, we rewrite the numerator (Eqn. 24) and denominator (Eqn. 25) in terms of constructive interference in $p_{i[j]}$ and $q_{[j]}$, leveraging the fact that $|\sum_i^N x_i| = C_i^N(x_i) \cdot \sum_i^N |x_i|$. Crucially, $C_i^N(g_i) = C_i^N(p_i)$, allowing us to finally isolate the effect of gradient opposition in Eqn. 26, with all other terms being independent of gradient opposition.

$$\begin{aligned}
\left|\sum_i^N \sum_j^M \Delta\theta_{[j]} \cdot g_{i[j]}\right| &= \left|\sum_j^M \Delta\theta_{[j]} \cdot NG_{[j]}\right| \\
&= \sum_j^M N \cdot |q_{[j]}| \cdot C_{j'}^M(q_{[j']}) \\
&= \sum_i^N \sum_j^M |p_{i[j]}| \cdot C_{i'}^N(p_{i'[j]}) \cdot C_{j'}^M(q_{[j']}) \\
&= \sum_i^N \sum_j^M |p_{i[j]}| \cdot C_{i'}^N(g_{i'[j]}) \cdot C_{j'}^M(q_{[j']})
\end{aligned} \tag{24}$$

$$\begin{aligned}
\sum_i^N \left|\sum_j^M \Delta\theta_{[j]} \cdot g_{i[j]}\right| &= \sum_i^N \left|\sum_j^M p_{i[j]}\right| \\
&= \sum_i^N \sum_j^M |p_{i[j]}| \cdot C_{j'}^M(p_{i\,[j']})
\end{aligned} \tag{25}$$

$$C_i^N(\tilde{\Delta}\ell_i) = \frac{\sum_i^N \sum_j^M |p_{i[j]}| \cdot C_{i'}^N(g_{i'[j]})}{\sum_i^N \sum_j^M |p_{i[j]}| \cdot C_{j'}^M(p_{i\,[j']})} \cdot C_{j'}^M(q_{[j']}) \tag{26}$$

Intuitively, the constructive interference terms in Eqn. 26 can be interpreted as:

$C_{i'}^N(g_{i'[j]})$	(lack of) gradient opposition across examples i', independent of $\Delta\theta$
$C_{j'}^M(p_{i\,[j']})$	(mis) alignment between $\Delta\theta, g_i$, independent of gradient opposition
$C_{j'}^M(q_{[j']})$	(mis) alignment between $\Delta\theta, G$, independent of gradient opposition

Quantifying the role of gradient opposition

To disentangle and quantify the role of gradient opposition in $C_i^N(\tilde{\Delta}\ell_i)$, we can rewrite Eqn. 26 as:

$$C_i^N(\tilde{\Delta}\ell_i) = \mathbf{C}_g \cdot \frac{\mathbf{C}_{(u,G)}}{\mathbf{C}_{(u,g)}} \tag{27}$$

$\mathbf{C}_g \in [0,1]$ captures constructive interference in $\tilde{\Delta}L = \frac{1}{N}\sum_i^N \sum_j^M p_{i[j]}$ attributable to (lack of) gradient opposition, weighted by the relative magnitude of the update at coordinate j.

$$\mathbf{C}_g = \frac{\sum_i^N \sum_j^M |p_{i[j]}| \cdot C_{i'}^N(g_{i'[j]})}{\sum_i^N \sum_j^M |p_{i[j]}|} = \frac{\sum_j^M \left|\sum_i^N p_{i[j]}\right|}{\sum_i^N \sum_j^M |p_{i[j]}|} \tag{28}$$

$\mathbf{C}_{(u,g)} \in [0,1]$ captures constructive interference in $\tilde{\Delta}L = \frac{1}{N}\sum_i^N \sum_j^M p_{i[j]}$ attributable solely to update-gradient (mis) alignment, aggregated over all examples i and independent of gradient opposition.

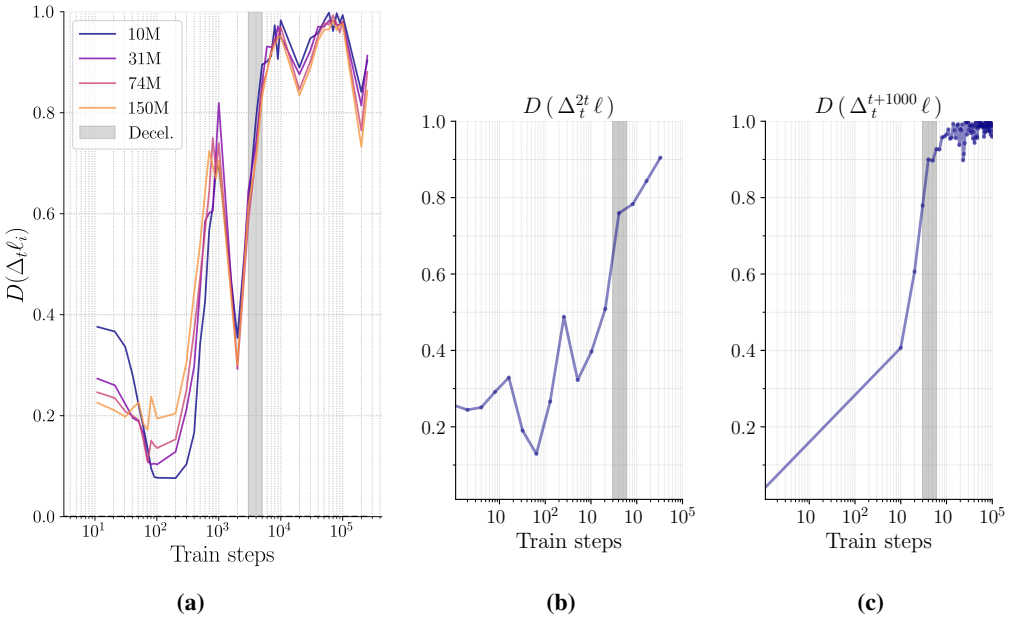
$$\mathbf{C}_{(u,g)} = \frac{\sum_i^N \sum_j^M |p_{i[j]}| \cdot C_{j'}^M(p_{i\,[j']})}{\sum_i^N \sum_j^M |p_{i[j]}|} = \frac{\sum_i^N \left|\sum_j^M p_{i[j]}\right|}{\sum_i^N \sum_j^M |p_{i[j]}|} \tag{29}$$

$\mathbf{C}_{(u,G)} \in [0,1]$ captures constructive interference in $\tilde{\Delta}L = \frac{1}{N}\sum_i^N \sum_j^M p_{i[j]}$ attributable solely to update-gradient (mis) alignment for the overall gradient, independent of gradient opposition.

$$\mathbf{C}_{(u,G)} = C_{j'}^M(q_{[j']}) = \frac{\left|\sum_j^M q_{[j]}\right|}{\sum_j^M |q_{[j]}|} = \frac{\left|\sum_i^N \sum_j^M p_{i[j]}\right|}{\sum_j^M \left|\sum_i^N p_{i[j]}\right|} \tag{30}$$

B.2 Effect of Increasing Steps on ZSL

Destructive interference is mitigated by increasing number of steps.
While the experiments and results in Section 3.2 consider the change in loss between steps t and $2t$, our initial experiments were based on checkpoints for steps $[1, 2, \ldots, 10, 20, \ldots, 100, 200, \ldots, 1000]$ and so on. When plotting $D(\Delta\ell)$ between these checkpoints in Fig. 15, we can see that $D(\Delta\ell)$ increases much more rapidly leading up to deceleration, when compared to Fig. 3. However, we also observe abrupt drops and subsequent rises in $D(\Delta\ell)$ after the number of steps between checkpoints is increased by a factor of 10. These results highlight that ZSL actually increases leading up to rather than after deceleration, but is mitigated by increasing number of steps.



(a) **(b)** **(c)**

Fig. 15: Effect of number of steps (between changes in loss) on destructive interference in per-example loss improvements, across several experiments. **(a)** Initial experiments based on checkpoints for steps $[1, 2, \ldots, 10, 20, \ldots, 100, 200, \ldots, 1000, 2000, \ldots]$, i.e. varying Δt. **(b)** The nanoGPT baseline from Appendix C.2 based on checkpoints for steps $[1, 2, 4, 8, \ldots]$ as in our main results. **(c)** The same nanoGPT baseline but for checkpoints $[0, 1000, 2000, \ldots]$, i.e. $\Delta t = 1000$. These different results point to the fact that increasing the number of steps for measuring loss improvements mitigates destructive interference in loss improvements.

Destructive interference in loss improvements after one step

In the extreme, we can consider loss improvements after only one optimizer step, although this can be quite noisy and vary significantly between consecutive steps. Nevertheless, this allows us to compare "Train" and "Eval" batches, i.e. the training batch for the given optimizer step and a withheld validation batch. We observe several surprising trends that are distinct from our main results (where we considered loss improvements over multiple steps to avoid issues with noise). Notably, we observe:

(1) In `Eval.` examples, destructive interference in loss improvements approaches it's maximum well before deceleration. Note that the dip before deceleration actually corresponds to a period where loss degrades after one step, suggesting overfitting of the training data that ends near deceleration, when destructive interference stabilizes around 1.0.

(2) In contrast, destructive interference in `Train.` examples increases until deceleration, but actually decreases after deceleration. In particular, there is a clear trend where increased model size correlates with decreased destructive interference after deceleration.

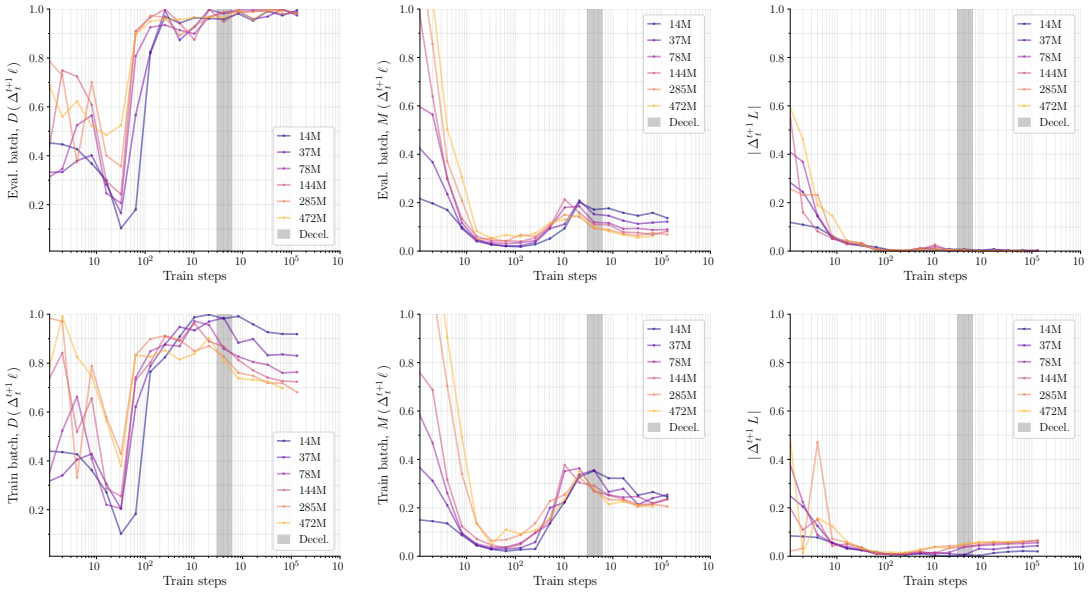


Fig. 16: Single-step ZSL in Train and Eval. batches.

Connecting these two observations and understanding how they relate to the multi-step behaviour seen in our main results is not clear. Counter-intuitively, it seems the only way in which larger models are markedly better is in their ability to overfit examples from a given training batch in one step, without generalization to other examples. However, despite this, our multi-step results suggest that larger models improve generalization, given the single epoch training on which our results are based. This implies that while overfitting occurs in any one step, there is generalization that occurs over many steps. One informal explanation for this phenomenon could be that larger models are better able to find common gradient directions for a given batch of data, leading to overfitting when considering one step, but better generalization when considering multiple steps. This interpretation is consistent with several other observations that learning after deceleration is more strongly associated with generalization (notably in-context learning ability (Olsson et al., 2022) as measured by improvements in later tokens, Appendix C.1; and greater improvements on downstream tasks compared to a model with the same loss but before deceleration Appendix B.3). Making this admittedly vague notion more precise and exploring it more rigorously is something we leave for future work.

B.3 Loss Deceleration and Downstream Performance

We compare the dowsntream performance of OLMo-1B and OLMo-7B on several downstream tasks as reported by Groeneveld et al. (2024). Surprisingly, we find that OLMo-1B checkpoints after deceleration typically outperform OLMo-7B checkpoints with the same loss before and during deceleration. This suggests that zero-sum learning plays an important role in generalization and that loss improvements after deceleration result in more significant generalization capabilities than equivalent loss improvements before deceleration.

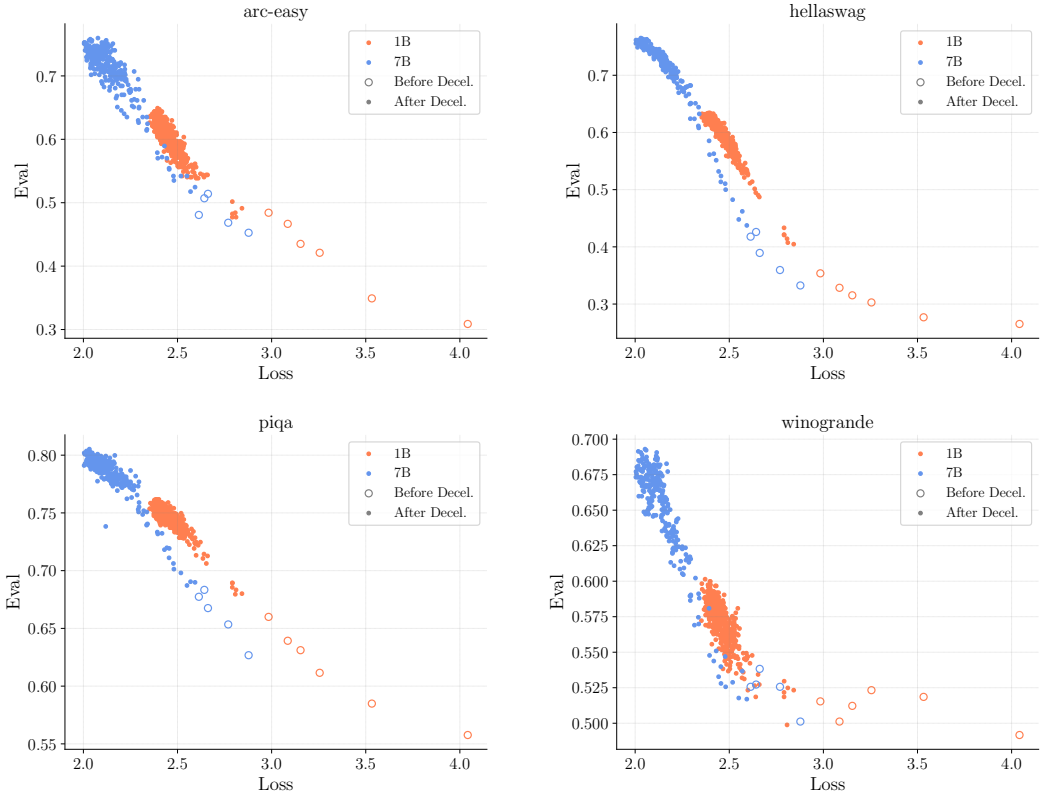


Fig. 17: Downstream performance of OLMo-1B and OLMo-7B on different tasks. Before and during deceleration, OLMo-7B checkpoints underperform OLMo-1B checkpoints with the same loss but after deceleration. This suggests zero-sum learning after deceleration plays an important role in generalization.

B.4 Per-token loss landscape cross-sections

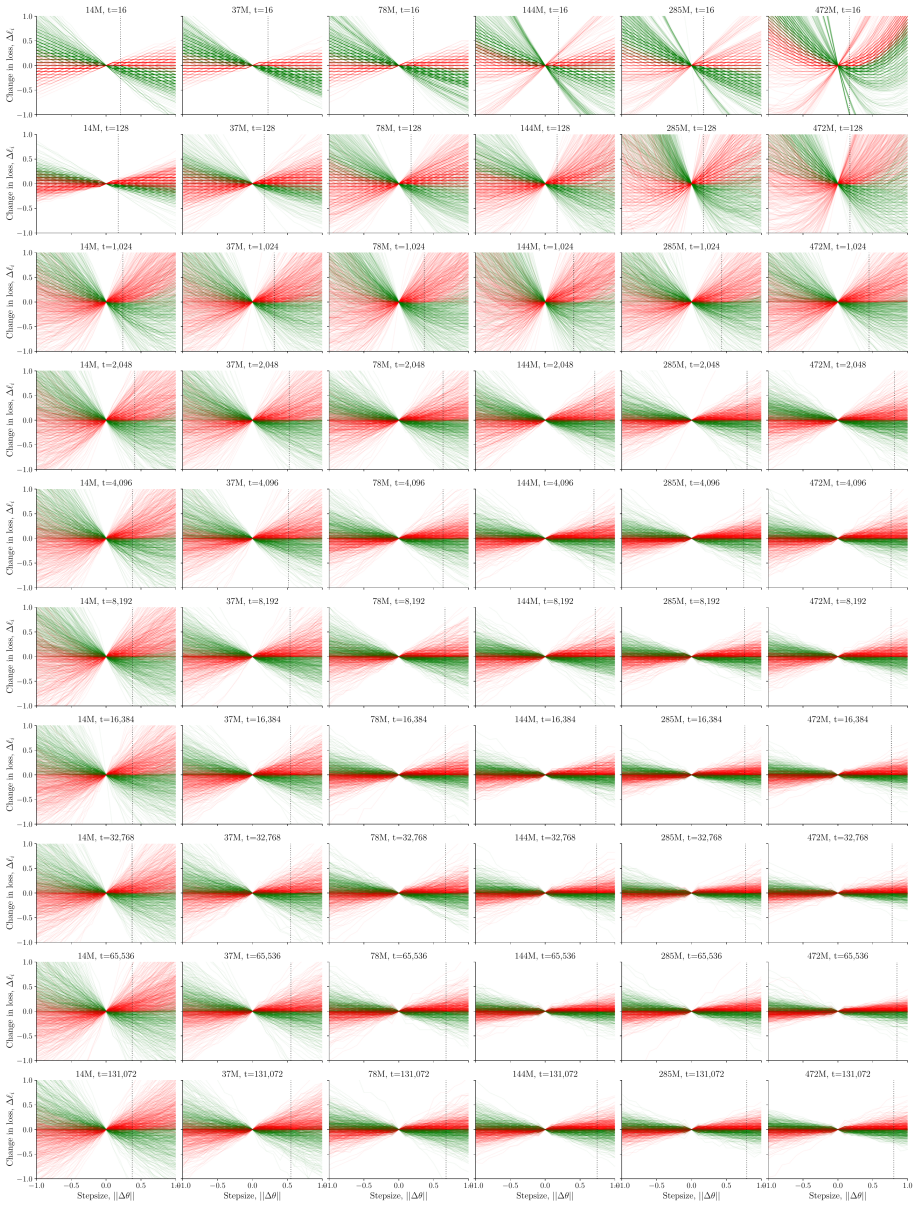


Fig. 18: Sampled per-token loss landscape cross-sections across model sizes and train steps
Across model sizes (columns) and train steps (rows), we plot loss landscape cross-sections along increments of the weight update $\Delta\theta$ at step t. The actual stepsize is indicated with a dotted vertical line. We plot ΔL rather than L, which has the same geometry but allows more easily distinguishing loss improvements from degradations. Lines are colored in green or red depending on whether the loss (respectively) improved or deteriorated at the actual stepsize.

Fig. 19: Sampled per-token loss landscape cross-sections across model sizes at the start of training We plot the same data as in Fig. 18, but focused on the beginning of training (before deceleration).

B.5 Overall loss landscape cross-sections throughout training

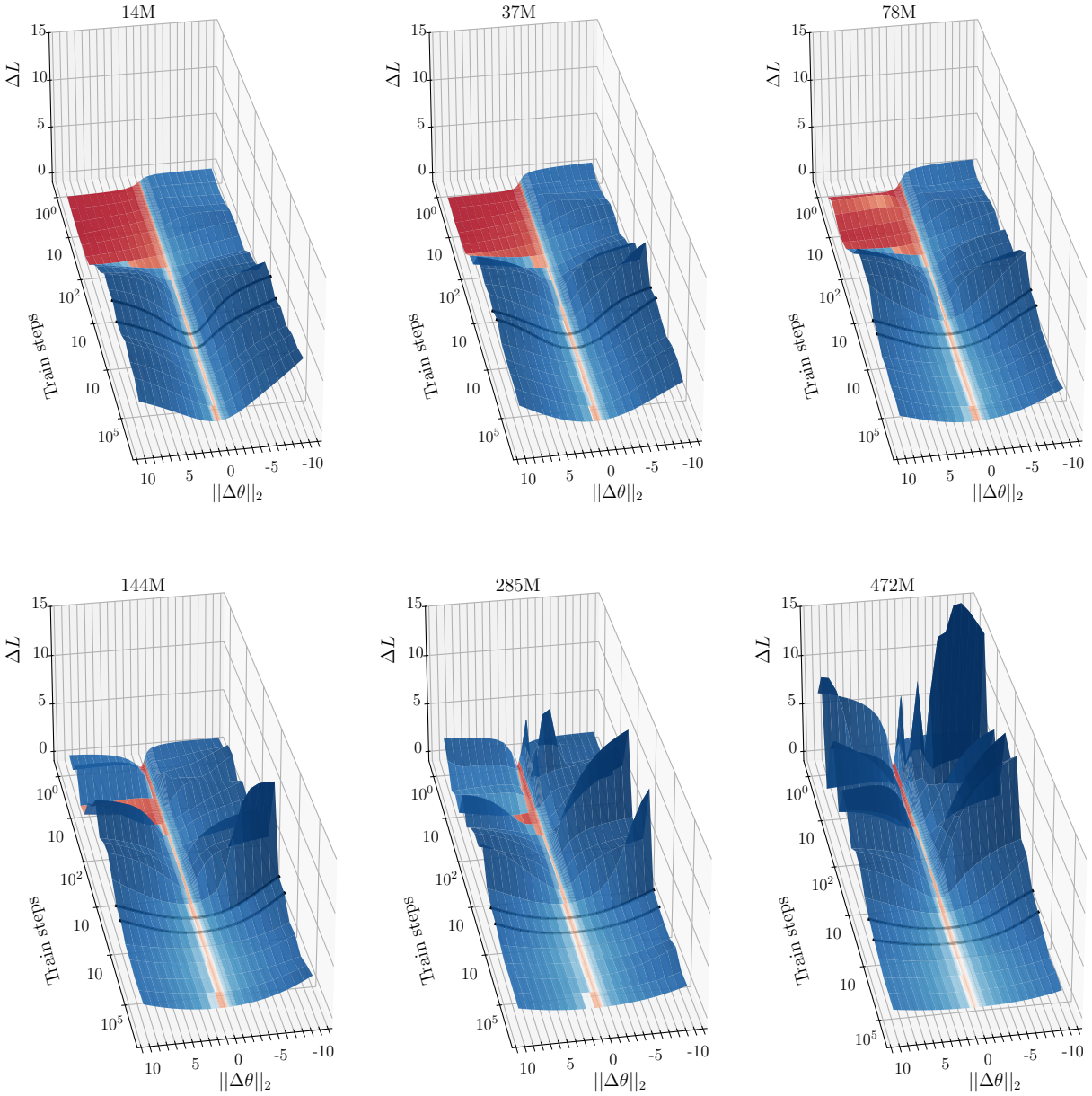


Fig. 20: Overall loss landscapes (cross section along $\Delta\theta$), visualized throughout training
We plot overall loss landscape cross sections across model sizes and train steps. Similar to Appendix B.4, we plot ΔL which has equivalent geometry to L but allows better distinguishing loss improvements from loss degradations. ΔL is additionally indicated with a symlog colorscale, with loss improvements being red. Loss deceleration is approximately indicated with two lines at $t = 4096$ and $t = 8192$. We observe that loss landscapes sharpen leading up to deceleration, but flatten significantly afterwards; with this trend being more pronounced in larger models. Furthermore, loss landscapes along $\Delta\theta$ appear much sharper in the beginning of training for larger models.

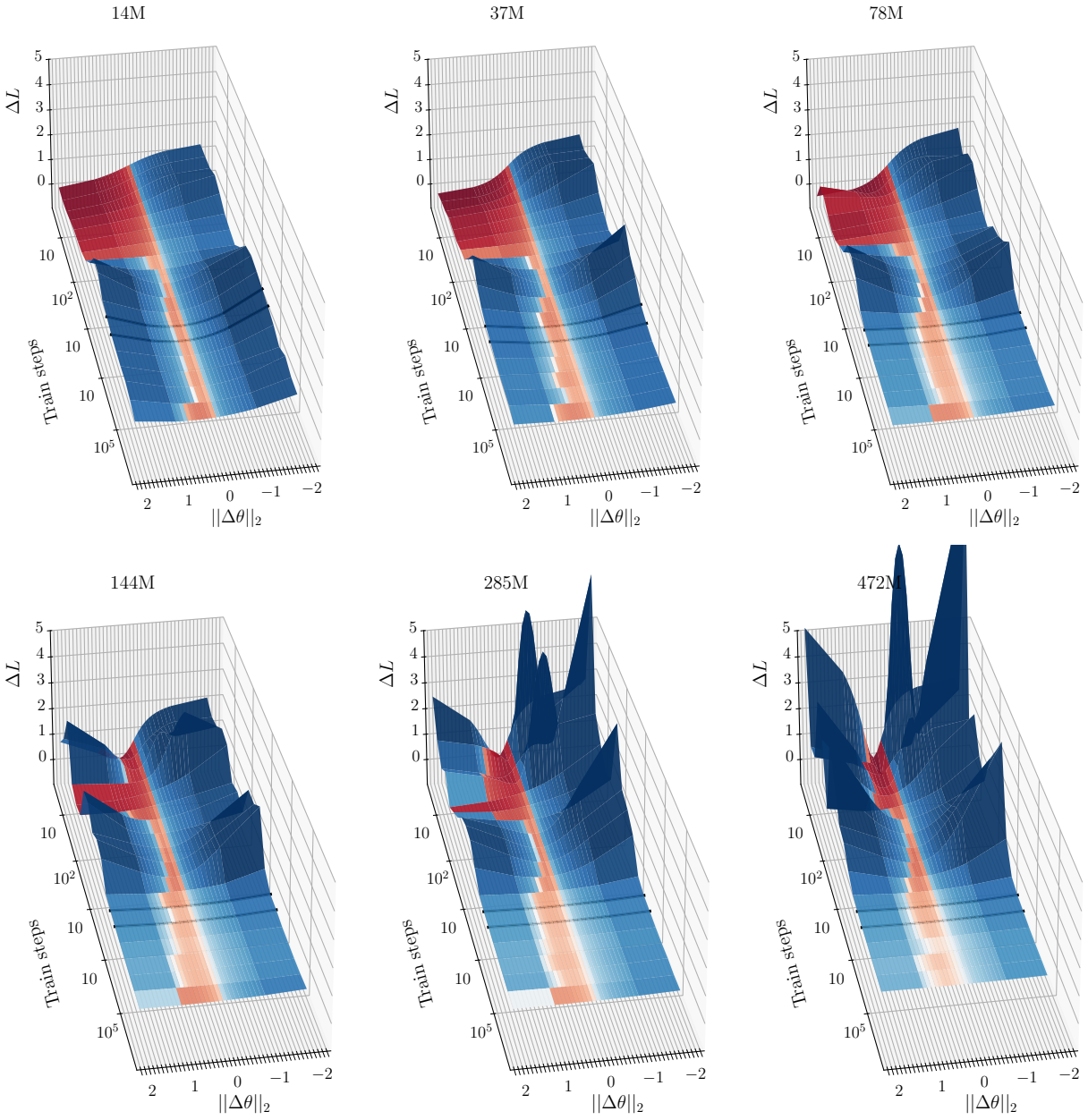


Fig. 21: Overall loss landscapes (cross section along $\Delta\theta$), visualized throughout training (zoomed in) We plot the same data as in Fig. 20, but zoomed into a narrower range.

B.6 Language Model Scaling Law Grounded in Loss Deceleration

Defining and fitting a scaling law grounded in loss deceleration

Let $L(N, T)$ be a scaling law for language model loss L, where N is the number of model parameters and T is number of training steps (with dataset size $D = T \cdot B$ for batch size B, i.e. single-epoch training). Recall from Eqn. 2 that an estimate of the loss L can be expressed in terms of the following parameters: **(1)** the number of steps at which deceleration occurs t_d; **(2)** the loss at which deceleration occurs L_d; and **(3)** the log-log rate of loss improvement after deceleration r_d. These parameters, shown in Table 1, are dependent on N, such that we can define a scaling law $L(N, T)$ grounded in loss deceleration as follows:

$$L(N, T) = L_d(N) \cdot t_d(N)^{r_d(N)} \cdot T^{-r_d(N)} \tag{31}$$

In Fig. 22, we observe, with the admittedly limited datapoints from our experiments, that L_d and r_d seem to exhibit power law scaling. In contrast, t_d appears to scale linearly if the outlier value for OLMo-7B, which is likely a result of being trained with 5,000 warmup steps instead of 2,000, is omitted. This suggests that warmup steps, among potentially other hyperparameters, have an important role not accounted for here. However, these results are preliminary and intended more as an exploratory proof of concept, included here for completeness, rather than a key result or claim of the paper. We leave the costly task of conducting sufficient training runs to more adequately validate this functional form for future work.

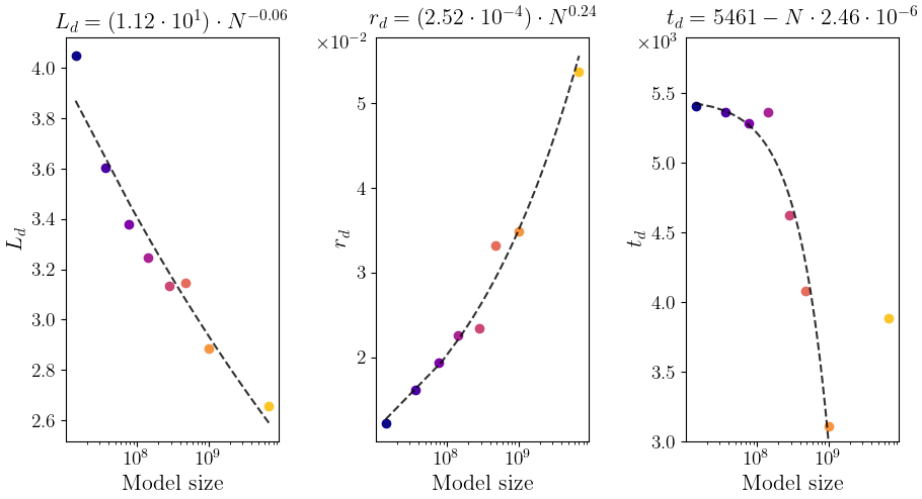


Fig. 22: Power law and linear scaling in deceleration parameters.

Table 5: BNSL parameters and error when training with cosine lr decay.

Model	b	c_0	c_1	$\log(d_1)$	f_1	RSLE
14M	18.32 ± 0.16	0.18 ± 0.00	-0.16 ± 0.00	8.56 ± 0.02	0.16 ± 0.03	0.012
37M	19.60 ± 0.22	0.20 ± 0.00	-0.17 ± 0.00	8.52 ± 0.02	0.18 ± 0.03	0.014
78M	20.67 ± 0.24	0.21 ± 0.00	-0.18 ± 0.00	8.48 ± 0.02	0.22 ± 0.03	0.014
144M	20.31 ± 0.25	0.21 ± 0.00	-0.18 ± 0.00	8.46 ± 0.03	0.24 ± 0.03	0.014
285M	20.87 ± 0.28	0.22 ± 0.00	-0.18 ± 0.00	8.27 ± 0.03	0.31 ± 0.03	0.013
472M	21.30 ± 0.29	0.23 ± 0.00	-0.18 ± 0.00	8.20 ± 0.03	0.31 ± 0.03	0.013
OLMo-1B	26.53 ± 0.42	0.28 ± 0.00	-0.24 ± 0.00	8.04 ± 0.03	0.76 ± 0.02	0.008
OLMo-7B	28.14 ± 0.54	0.29 ± 0.00	-0.23 ± 0.00	8.26 ± 0.04	0.78 ± 0.03	0.008

C Additional Results (Ablations)

C.1 Effect of Batch Size and Sequence Length

On deceleration In this set of experiments, we vary the batch size (1M and 4M tokens) and the sequence length (1024 and 2048 tokens) for our 144M model. The results are presented in Fig. 23 and Table 6. In terms of loss deceleration, increasing the batch results in improved loss at deceleration L_d but similar post-deceleration rate of loss improvement r_d. In contrast, increasing sequence length results in improved r_d and similar L_d.

Note that these results are obtained from training curves, i.e. losses are computed on training batches that differ in batch size and sequence length. Hence, the differences in results can be partially attributed to differences in the data used to compute the losses. Similarly, the results for OLMo-1B and OLMo-7B in Fig. 2 were obtained from training curves due to limited checkpoint availability. The apparent performance gap between them and smaller models can also be partially attributed to increases in batch size and sequence length.

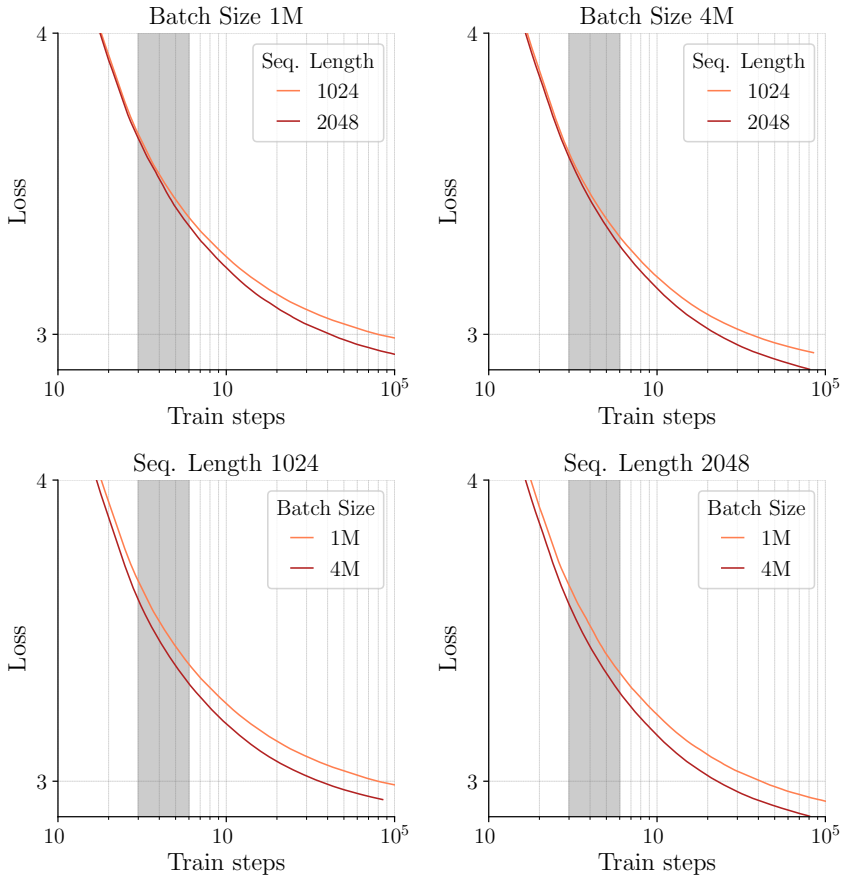


Fig. 23: Increasing batchsize improves loss at deceleration L_d, but shows similar post-deceleration rate of loss improvement r_d. In contrast, increasing sequence length results in improved r_d and similar L_d.

Batch Size	Seq. Len	L_d	t_d	r_d	b	c_0	c_1	$\log(d_1)$	f_1	RSLE
1M	1024	3.28	5120	0.033	21.39 ± 0.30	0.22 ± 0.00	-0.19 ± 0.00	8.54 ± 0.03	0.25 ± 0.04	0.016
	2048	3.24	5331	0.034	22.06 ± 0.34	0.22 ± 0.00	-0.19 ± 0.00	8.58 ± 0.03	0.25 ± 0.04	0.018
4M	1024	3.26	4893	0.038	21.56 ± 0.32	0.22 ± 0.00	-0.18 ± 0.00	8.50 ± 0.03	0.20 ± 0.05	0.017
	2048	3.23	4968	0.041	22.11 ± 0.35	0.23 ± 0.00	-0.18 ± 0.00	8.51 ± 0.04	0.21 ± 0.05	0.019

Table 6: Deceleration measurements, BNSL fit parameters and errors for different batch sizes and sequence lengths.

Note on sequence length We reproduce Fig. 23 with fixed batches of size 0.5M and sequence length 1024 on saved checkpoints and show the results in Fig. 24. We find that improvements from increasing sequence length do not transfer, suggesting they are limited to tokens from longer sequences. This is not particularly surprising given that tokens with more context are typically easier to predict, which would account for the loss improvements seen before. However, what is surprising is that these improvements only appear after deceleration, suggesting that the ability to leverage this extra context to better predict later tokens emerges with ZSL.

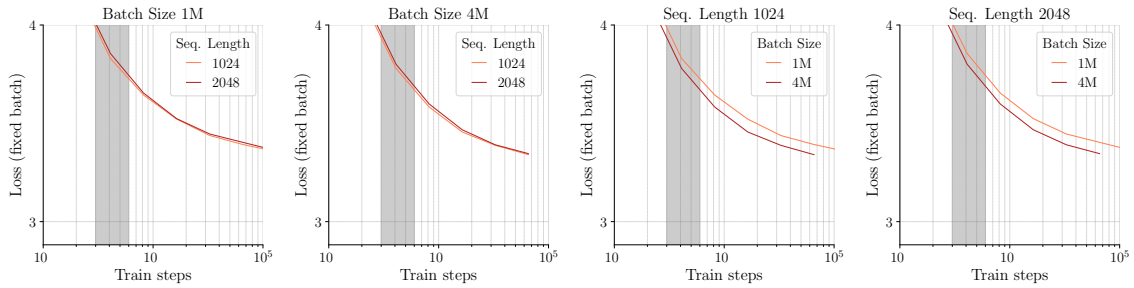


Fig. 24: When measuring loss on withheld validation batches with fixed sequence length and batch size, we find that the improvements from increased sequence length disappear (left) while those from increased batch size remain (right).

Effects on zero-sum learning We measure destructive interference in loss improvements as in Fig. 3 and show the results in Fig. 25. Results are consistent across several batch sizes and sequence lengths, and these hyperparameters, despite their different effects on train and validation loss curves, do not have an effect on zero-sum learning. In the case of sequence length, this is consistent with the observation that sequence length does not improve performance. In the case of batch sizes, this is consistent with the observation that improvements are established *before* deceleration and would therefore not involve mitigating zero-sum learning.

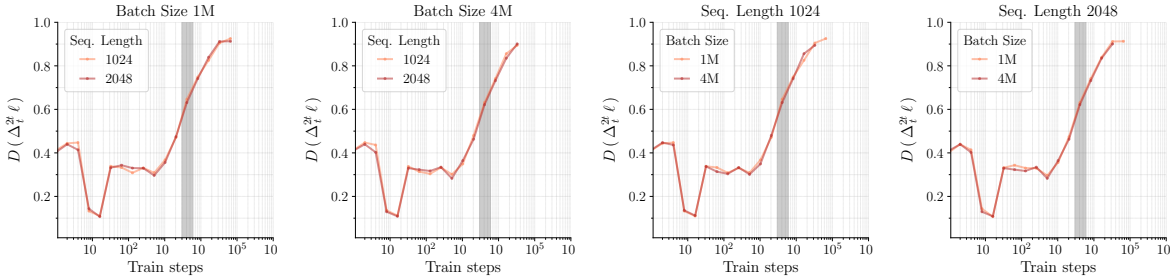


Fig. 25: Increasing batch size and sequence length both have no discernible effect on ZSL as measured by destructive interference in loss improvements.

C.2 Effect of Optimizer Variants

In this set of experiments, we compare different optimizers to the baseline AdamW (Loshchilov and Hutter, 2019) used in our main experiments, with the 144M model. The tested optimizers include AdEMAMix (Pagliardini et al., 2024), and Muon (Jordan et al., 2024). Furthermore, these results are obtained with NanoGPT (Karpathy, 2022) rather than OLMo as in our main results, primarily because modifying optimizers is less error-prone in this simpler framework and because it is the basis of the original Muon implementation by Jordan et al. (2024). This framework differs in several important aspects which help corroborate the generality of our findings. In particular, these experiments use a different dataset–OpenWebText2 (Gao et al., 2020)–and a different model architecture and vocabulary based on GPT-2 (Radford et al., 2019).

Effect on deceleration

Surprisingly, Muon and AdEMAMix have qualitatively different effects on loss deceleration. While Muon appears to improve early performance by accelerating deceleration with smaller t_d, AdEMAMix appears to improve performance by improving rate of loss improvement after deceleration with larger r_d.

Furthermore, despite faster convergence, Muon exhibits significant training instabilities after deceleration, coinciding with large spikes in gradient norms. While this has, to the best of our knowledge, not been reported elsewhere, these results were obtained using the original implementation of Jordan et al. (2024). Whether or not this instability can be avoided is not clear.

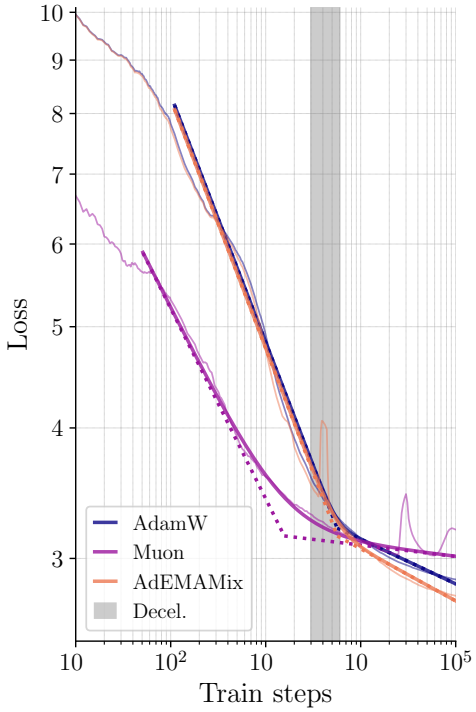


Fig. 26: Loss curves and BNSL fits.

Model	L_d	t_d	r_d	b	c_0	c_1	$\log(d_1)$	f_1	RSLE
AdamW	3.21	5650	0.043	24.77 ± 0.46	0.24 ± 0.00	-0.19 ± 0.00	8.64 ± 0.04	0.15 ± 0.06	0.026
Muon	3.15	1569	0.011	12.04 ± 0.15	0.18 ± 0.00	-0.17 ± 0.00	7.36 ± 0.04	0.71 ± 0.05	0.010
AdEMAMix	3.15	5673	0.050	24.73 ± 0.53	0.24 ± 0.00	-0.19 ± 0.01	8.64 ± 0.05	0.25 ± 0.07	0.025

Table 7: Deceleration measurements; BNSL fit parameters and errors for different optimizers.

Effect on zero-sum learning We find that the effects of AdEMAMix and Muon on zero-sum learning are consistent with the previously observed effect on loss deceleration. Specifically, destructive interferences rises notably faster in Muon, consistent with the faster onset of loss deceleration observed. Furthermore, we observe significant oscillations after deceleration which correspond to the training instabilities observed as loss degrades closer to the end of training. In contrast, AdEMAMix behaves very similarly to AdamW leading up to deceleration, consistent with the fact that they both reach loss deceleration at similar step and loss values. However, after deceleration, AdEMAMix typically exhibits lower destructive interference than AdamW. In contrast to Muon which has similar oscillations, these are not attributable to training instability and loss degradation, and more likely to account for the improved rate of loss improvements observed in AdEMAMix after deceleration. These ablations were intended simply to verify the generality of our findings across different optimizers, however the results suggest that the considered optimizers differ in important ways with respect to zero-sum learning, although more comprehensive experiments are required to validate our preliminary findings.

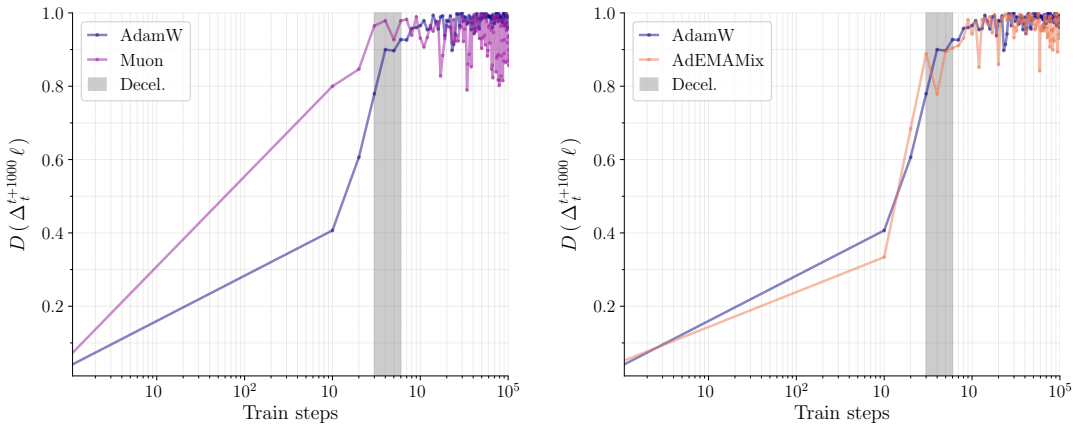


Fig. 27: Destructive interference in loss improvements for AdEMAMix and Muon, compared to baseline AdamW

Note on Cautious AdamW

We initially included Cautious or C-AdamW (Liang et al., 2025) in our analysis, however the results were effectively indistinguishable from our baseline. This is consistent with the original results reported by Liang et al. (2025) (Fig. 28), where C-AdamW only improves performance on the 1B model (which itself underperforms the 350M model AdamW baseline).

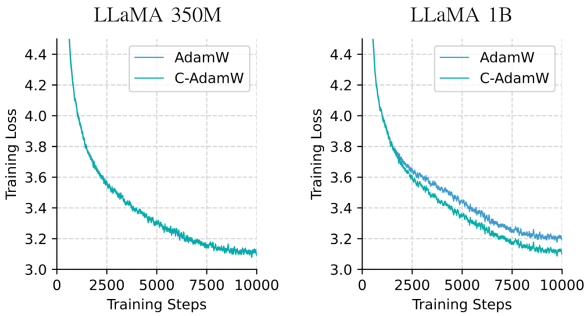


Fig. 28: Original C-AdamW results (Liang et al., 2025).

C.3 Effect of Learning Rate Decay

Our main results are for training runs where learning rate was warmed up and held constant, in line with Hägele et al. (2024) and Wen et al. (2024). However, typically scaling experiments have been conducted with learning rate decay. In particular, Hoffmann et al. (2022) note that consistently decaying to 0.1 of the peak learning rate as an important difference to Kaplan et al. (2020), leading to different compute-optimal scaling. To rule out this potential confound, we replicate our experiments with a cosine learning rate decay in line with Hoffmann et al. (2022) (and Groeneveld et al. (2024)), leaving all else equal.

Effect on loss deceleration

Fig. 2 is replicated in Fig. 29, with similar results and quality of fits. Table 4 is replicated in Table 5 with again similar results, and generally smaller values for c_1, $\log(d_1)$, and f_1. Lastly, Table 1 is replicated in Table 8, where we see that L_d resulting from the BNSL fit is increased, but this is offset by improved r_d and t_d, leading to better final loss. This improvement in final loss appears to increase with model size, suggesting a complementary mechanism by which scale improves loss under learning rate decay, which is not accounted for by our principal findings.

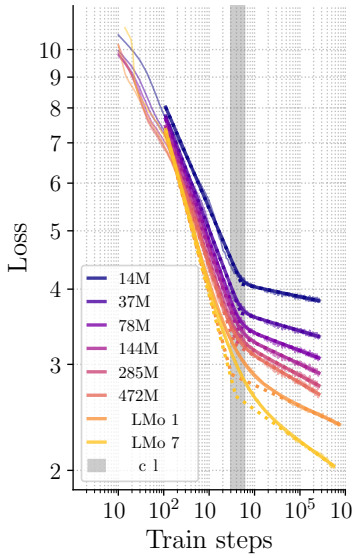


Fig. 29: Loss curves and BNSL fits when training with cosine lr decay.

Model	L_d	r_d	t_d	$\hat{L}_T$	L_T
14M	4.08	0.016	5198	3.83	3.86
37M	3.65	0.023	5029	3.34	3.36
78M	3.45	0.029	4808	3.07	3.09
144M	3.35	0.036	4712	2.90	2.92
285M	3.28	0.040	3921	2.77	2.78
472M	3.24	0.045	3653	2.68	2.69
OLMo-1B	2.89	0.035	3106	2.39	2.38
OLMo-7B	2.66	0.054	3885	2.03	2.02

Table 8: Deceleration measurements with lr decay.

D Related works

This work connects several existing areas of research. In particular, several recent works attempt to explain scaling laws, typically from the perspective of intrinsic model capacity, long-tailed data distributions, and asymptotic behaviour (e.g. Hutter, 2021; Sharma and Kaplan, 2022; Michaud et al., 2023; Bahri et al., 2024; Bordelon et al., 2024). In contrast, our goal is to identify a mechanism grounded in training dynamics that can be targeted independent of scale. The mechanism we identify, loss deceleration, is to the best of our knowledge not addressed in relevant prior works on e.g. loss plateaus (Yoshida and Okada, 2020), learning curve shapes (Viering and Loog, 2022), or LM saturation (Godey et al., 2024; Mircea et al., 2024). Lastly, the study of training dynamics based on per-example gradient interactions remains under-explored, with related tangential works on e.g. improving multi-task learning (Liu et al., 2021), or characterizing outliers in SGD (Rosenfeld and Risteski, 2023).

Explaining scaling laws Several works have proposed different explanations for neural scaling laws such as (Kaplan et al., 2020; Hoffmann et al., 2022; Caballero et al., 2023; Hägele et al., 2024; Tissue et al., 2024; Everett et al., 2024). Notably, Bahri et al. (2024) explain scaling laws in terms of asymptotic behaviour, identifying variance-limited regimes based on concentration around infinite limits, and resolution-limited regimes based on distances between train and test data points on their manifold (see also (Sharma and Kaplan, 2022)). Atanasov et al. (2024) analytically explain power-law scaling in high-dimensional ridge regression with tools from random matrix theory. Michaud et al. (2023) propose a "quantization model of neural scaling", whereby power law scaling is a result of (1) language models improving loss by learning discrete capabilities from their demonstration in data, (2) larger models being able to learn more capabilities, and (3) rarer capabilities improve loss by smaller and smaller amounts due to their vanishing frequency. Similarly, Hutter (2021) show how power law scaling with data can arise from long-tail feature distributions.

Improving language models independently of scaling Recent work on e.g. data pruning (Marion et al., 2023; Sorscher et al., 2022), model distillation (Allen-Zhu and Li, 2023; Team et al., 2024), and model pruning (Raposo et al., 2024) show that improvements predicted from scaling can (up to a point) be realized without scaling. This suggests that scaling may indirectly improve loss by its effect on training dynamics, and that similar effects/improvements can be obtained without necessarily scaling.

Gradient opposition From the perspective of training dynamics, Rosenfeld and Risteski, 2023 discuss the effect of outlier samples with opposing gradients. In the context of multi-task learning, several works have proposed approaches to mitigate gradient opposition between tasks, e.g. (Parascandolo et al., 2020; Yu et al., 2020; Liu et al., 2021). Gradient opposition between tokens in language modeling has, to the best of our knowledge, not been characterized. Related but distinct, is the work of Mircea et al. (2024) characterizes opposition within token gradients rather than between different tokens.

Loss deceleration and learning curves To the best of our knowledge, the loss deceleration transition we identify and characterize in this work has not been previously established or explained. We refer the reader to Viering and Loog (2022) for a comprehensive review of learning curve shapes, as well as Hutter (2021) and Yoshida and Okada (2020) as examples of attempting to explain features in a learning curve.