RELATIVE GENERALIZATION INVARIANCE OF LLM PRETRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Large Language Model (LLM) pretraining performance is jointly shaped by three components of the training triplet: the optimizer, model architecture, and training data stream. However, how these components influence performance in distinct ways remains unclear. We take a first step toward isolating their effects by studying relative generalization. We introduce Relative Generalization Invariance (RGI), the invariance of the validation-loss difference between any two tokens across models. We show that RGI approximately holds across a wide range of optimizers and moderate architectural variations, suggesting that these choices induce an approximately uniform shift in token-wise losses. In contrast, changing the training data stream can substantially alter relative generalization. We further show that RGI cannot be explained by the neural tangent kernel or mean-field regimes alone and prove that it can emerge in an overparameterized quadratic model. Overall, our work identifies RGI as a new phenomenon in LLM pretraining that helps distinguish the effects of optimizers and architectures from those of training data.

1 INTRODUCTION

Pretraining equips Large Language Models (LLMs) with broad knowledge and reasoning capabilities and provides the foundation for subsequent instruction tuning and reinforcement learning. Its performance is jointly shaped by the three components of the *training triplet*: the optimizer, model architecture, and training data stream. Prior work has improved pretraining performance and efficiency by advancing each component, including new optimizers such as Soap [1] and Muon [2], architectural designs such as Sliding Window Attention (SWA) [3] and SiLU [4], and data pipelines such as dynamic data selection [5].

Despite this progress, much less is known about how the optimizer, model architecture, and training data stream affect performance in distinct ways. We take a step toward isolating their effects by studying a fundamental structure of generalization, which we call *relative generalization*: the relative loss structure across validation tokens. We show that optimizers and architectures affect token-wise losses in an approximately uniform way, preserving relative generalization across a wide range of optimizers and moderate architectural variations, a phenomenon we term Relative Generalization Invariance (RGI). In contrast, changing the training data stream can substantially alter relative generalization. Our primary contributions are as follows:

First, we define RGI between models as the invariance of the validation-loss difference between any two tokens across models. We show that RGI is equivalent to the two models having a token-independent constant gap in validation loss, i.e., a uniform shift across tokens. We then extend RGI to training triplets through the models they produce.

Second, we show that RGI holds across a wide range of optimizers and moderate architectural variations, but depends strongly on the training data stream. Specifically, we consider coordinate-wise optimizers, such as Adam, matrix-based optimizers, such as Muon, and variations in attention and Feed-Forward Networks (FFN) designs. We show that RGI persists under joint variations in optimizer and architecture and is robust to optimizer hyperparameters and moderate shifts in the validation distribution. However, RGI can break when models are trained on data streams from different sources. In other words, optimizer and architecture choices induce approximately uniform shifts in token-wise losses, whereas changes in the training data stream can alter the relative loss structure across tokens.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Third, we show that RGI cannot be explained by the Neural Tangent Kernel (NTK) or Mean-Field (MF) regimes alone. When RGI emerges, models deviate substantially from initialization in both parameters and hidden-state representations, and different optimizers lead to low similarity in both quantities. These findings rule out an NTK-style lazy-learning explanation. We further reduce the model width and show that RGI persists even when the model departs from the MF regime.

Finally, we study a stylized quadratic model and theoretically characterize the emergence of RGI. Beyond the NTK and MF regimes, we identify a setting where RGI holds across GD, Adam, and Muon, different initializations, and learning rates, despite low parameter similarity.

Together, these results take a first step toward distinguishing the effects of optimizers, architectures, and data through the definition, empirical demonstration, and theoretical characterization of RGI.

2 Preliminaries

Pretraining of LLMs optimizes model parameters over a stream of data sampled from a corpus. Let the initial parameters $\theta_0 \in \Theta$ be drawn from an initialization distribution $P_{\text{init}} \in \Delta(\Theta)$. Starting from θ_0, the model is trained on a data stream $D_{1:\infty} = (D_t)_{t=1}^{\infty}$, where each batch $D_t \subseteq \mathcal{D}$ is sampled from the corpus $\mathcal{D}$ according to a possibly time-dependent distribution $P_t(\cdot \mid D_{1:t-1})$, independent of θ_0. Each data point $d \in D_t$ consists of a prefix and a target token, e.g., $d =$ ("LLM pretraining is", "important"). Given an optimizer $\mathsf{A} \in \mathcal{A}$, we define the *training triplet* as $\mathfrak{T} = (\mathsf{A}, \theta_0, D_{1:\infty})$, where θ_0 specifies the initialization and its associated model architecture. The parameters at step t can then be written as $\theta_t^{\mathfrak{T}} = F(\mathfrak{T}, t)$, where F is deterministic given the training triplet.[1] The map F is induced by the optimizer's update rule. For a batch $D \subseteq \mathcal{D}$, let $L(\theta, D) = |D|^{-1} \sum_{d \in D} L(\theta, d)$ denote the average cross-entropy loss, and define the batch gradient as $\xi_t^{\mathfrak{T}} = \nabla_\theta L(\theta_{t-1}^{\mathfrak{T}}, D_t)$. The optimizer A recursively updates its internal state $S_t^{\mathfrak{T}} \in \mathcal{S}$ and the model parameters according to

$$S_t^{\mathfrak{T}} = G(S_{t-1}^{\mathfrak{T}}, \xi_t^{\mathfrak{T}}, \mathsf{A}), \quad \theta_t^{\mathfrak{T}} = H(S_t^{\mathfrak{T}}, \theta_{t-1}^{\mathfrak{T}}, \mathsf{A}),$$

where G and H denote the state-update and parameter-update maps, respectively.

Optimizers instantiate the update maps G and H, and hence the training map F, differently. Coordinate-wise optimizers, such as Adam [8], Lion [9], and Adam-mini [10], use entry-wise gradient statistics. For the training triplet $\mathfrak{T}_{\mathsf{Adam}} = (\mathsf{Adam}, \theta_0, D_{1:\infty})$, Adam maintains first- and second-moment states $S_t^{\mathsf{Adam}} = S_t^{\mathfrak{T}_{\mathsf{Adam}}} = (m_t, v_t) \in \mathcal{S}_{\mathsf{Adam}}$ and updates

$$m_t = \gamma_1 m_{t-1} + (1-\gamma_1)\xi_t, v_t = \gamma_2 v_{t-1} + (1-\gamma_2)\xi_t \odot \xi_t, \theta_t^{\mathsf{Adam}} = \theta_{t-1}^{\mathsf{Adam}} - \eta\, m_t \odot (\sqrt{v_t} + \varepsilon)^{-1},$$

where $\odot$ denotes the Hadamard product and $\gamma_1, \gamma_2, \eta, \varepsilon$ are hyperparameters; bias correction is omitted for simplicity. Here and below, we abbreviate $S_t^{\mathfrak{T}_{\mathsf{Adam}}}$ and $\theta_t^{\mathfrak{T}_{\mathsf{Adam}}}$ as S_t^{Adam} and θ_t^{Adam}. In contrast, matrix-based optimizers, such as Muon [2], Soap [1], Scion [11], and Shampoo [12], exploit the matrix structure of parameter blocks. For example, an idealized Muon update for each matrix-shaped block is

$$m_t = \gamma m_{t-1} + (1-\gamma)\xi_t, \quad \theta_t^{\mathsf{Muon}} = \theta_{t-1}^{\mathsf{Muon}} - \eta U_t V_t^\top,$$

where $S_t^{\mathsf{Muon}} = m_t$, $U_t \Sigma_t V_t^\top$ is the Singular-Value Decomposition (SVD) of m_t, and γ, η are hyperparameters. In practice, Muon approximates $U_t V_t^\top$ using Newton–Schulz iterations. See Appendix A for further discussion of these optimizers.

3 Relative Generalization Invariance

In this section, we define a new relationship between the validation losses of LLMs pretrained with different optimizers, model architectures, and data streams. Specifically, consider two training triplets $\mathfrak{T} = (\mathsf{A}, \theta_0, D_{1:\infty})$ and $\mathfrak{T}' = (\mathsf{A}', \theta_0', D_{1:\infty}')$, which produce parameters $\theta_t^{\mathfrak{T}}$ and $\theta_t^{\mathfrak{T}'}$ at step t. We allow θ_0 and θ_0' to correspond to different model architectures and, in particular, models of different sizes. In our experiments, $(\theta_0, D_{1:\infty})$ and $(\theta_0', D_{1:\infty}')$ are either identical or independently sampled. We compare the resulting models on subsets $D_{\text{val}} \subseteq \mathcal{D}_{\text{val}}$, with

[1] Our work focuses on deterministic optimizers. Randomized optimizers can be incorporated by treating the algorithmic randomness as an additional input to F [6, 7].

$L(\theta_t^{\mathfrak{T}}, D_{\text{val}}) = |D_{\text{val}}|^{-1} \sum_{d \in D_{\text{val}}} L(\theta_t^{\mathfrak{T}}, d)$, and analogously for $L(\theta_t^{\mathfrak{T}'}, D_{\text{val}})$. For generality, we define RGI under an arbitrary joint distribution over the two initializations and training data streams.

Definition 1 (Relative Generalization Invariance) *Relative Generalization Invariance (RGI) holds between two parameters $\theta, \theta' \in \Theta$ over $\mathcal{D}_{\text{val}}$ if*

$$L(\theta, d) - L(\theta, d') = L(\theta', d) - L(\theta', d') \text{ for any } d, d' \in \mathcal{D}_{\text{val}}. \tag{1}$$

For two training triplets $\mathfrak{T} = (\mathsf{A}, \theta_0, D_{1:\infty})$ and $\mathfrak{T}' = (\mathsf{A}', \theta_0', D_{1:\infty}')$ and their parameters at step t, $\theta_t^{\mathfrak{T}} = F(\mathfrak{T}, t)$ and $\theta_t^{\mathfrak{T}'} = F(\mathfrak{T}', t)$, we say that RGI holds across $\mathfrak{T}$ and $\mathfrak{T}'$ over $\mathcal{D}_{\text{val}}$ at time step t if

$$L\big(\theta_t^{\mathfrak{T}}, d\big) - L\big(\theta_t^{\mathfrak{T}}, d'\big) = L\big(\theta_t^{\mathfrak{T}'}, d\big) - L\big(\theta_t^{\mathfrak{T}'}, d'\big) \tag{2}$$

for any $d, d' \in \mathcal{D}_{\text{val}}$, almost surely with respect to the joint distribution of two initializations and training data streams. We say that Asymptotic RGI (ARGI) holds between training triplets $\mathfrak{T}$ and $\mathfrak{T}'$ if there exists $t^* > 0$ such that RGI holds between them for any $t \geq t^*$.

The notion of RGI between parameters requires that the validation-loss difference between any two tokens d and d' is the same under θ and θ'. Thus, θ and θ' preserve the same relative token-wise loss structure; in particular, if token d' incurs a higher loss than token d under θ, then the same holds under θ'. At the level of training triplets, RGI between $\mathfrak{T}$ and $\mathfrak{T}'$ at step t requires this property to hold between the resulting parameters almost surely with respect to the joint randomness in their initializations and training data streams. Equivalently, their token-wise validation losses differ only by a token-independent constant. This characterization is formalized in the following result.

Proposition 1 (Equivalent Statements of RGI) *The following statements are equivalent.*

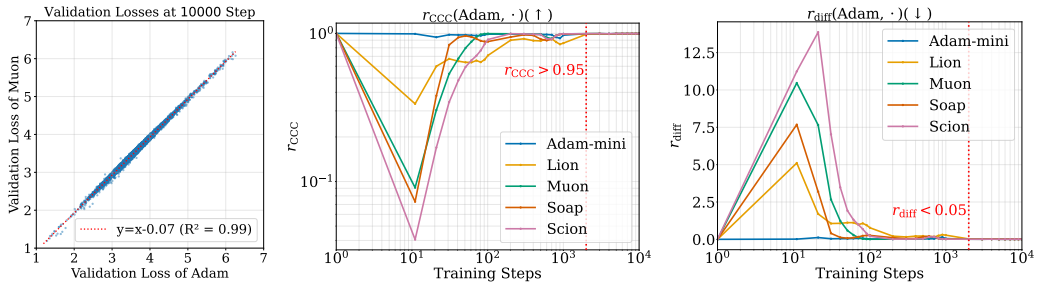
- *(RGI) RGI holds between training triplets $\mathfrak{T}$ and $\mathfrak{T}'$ at time step t.*
- *(Constant Generalization Gap) The training triplets $\mathfrak{T}$ and $\mathfrak{T}'$ have a constant generalization gap at time t, i.e., $L(\theta_t^{\mathfrak{T}}, d) - L(\theta_t^{\mathfrak{T}'}, d) = C(\mathfrak{T}, \mathfrak{T}', t)$ for any $d \in \mathcal{D}_{\text{val}}$ almost surely.*

This result shows that RGI is equivalent to a token-independent constant gap in validation loss, motivating two complementary metrics for evaluating ARGI. First, motivated by the constant-gap characterization, we evenly partition $\mathcal{D}_{\text{val}}$ into K non-overlapping subsets $(D_{\text{val},k})_{k=1}^K$ and compute the Concordance Correlation Coefficient (CCC) between the centered values of $L(\theta_t^{\mathfrak{T}}, D_{\text{val},k})$ and $L(\theta_t^{\mathfrak{T}'}, D_{\text{val},k})$ across k [13], denoted by $r_{\text{CCC}}(\mathfrak{T}, \mathfrak{T}') \in [-1, 1]$. Larger r_{CCC} indicates closer agreement with RGI, with $r_{\text{CCC}} = 1$ implying that the two sets of validation losses differ only by a constant shift. Second, we define the loss difference $\Delta(\mathfrak{T}, D_{\text{val}}, D_{\text{val}}') = L(\theta_t^{\mathfrak{T}}, D_{\text{val}}) - L(\theta_t^{\mathfrak{T}}, D_{\text{val}}')$. When $|D_{\text{val}}| = |D_{\text{val}}'| = 1$, RGI is equivalent to $\Delta(\mathfrak{T}, D_{\text{val}}, D_{\text{val}}') = \Delta(\mathfrak{T}', D_{\text{val}}, D_{\text{val}}')$. We define $r_{\text{diff}}(\mathfrak{T}, \mathfrak{T}')$ as the normalized discrepancy between the corresponding loss differences under $\mathfrak{T}$ and $\mathfrak{T}'$. Smaller r_{diff} indicates closer alignment with RGI; when each subset contains a single token, $r_{\text{diff}} = 0$ implies exact RGI. These metrics are normalized to facilitate the evaluation of RGI; see Appendix E for further discussion. Their explicit definitions are provided in Appendix B.

4 Empirical Studies of Relative Generalization Invariance

In this section, we investigate when RGI holds between two training triplets $\mathfrak{T}$ and $\mathfrak{T}'$. Specifically, we examine four sources of variation in $L(\theta_t^{\mathfrak{T}}, D_{\text{val}})$, where $\mathfrak{T} = (\mathsf{A}, \theta_0, D_{1:t})$: the optimizer A, the model architecture and initialization associated with θ_0, the training data stream $D_{1:t}$, and the validation data D_{val}.

Experimental setup. We first describe the baseline experimental setting, from which subsequent experiments vary the optimizer, model architecture, and training data. We consider three coordinate-wise optimizers, Adam [8], Lion [9], and Adam-mini [10], and three matrix-based optimizers, Soap [1], Muon [2], and Scion [11]. We use them to train 124M, 0.7B, and 4.5B Nano-GPT models on FineWeb [14], with RoPE [15] in attention and GELU [16] in the FFN. The validation set $\mathcal{D}_{\text{val}}$ is drawn from FineWeb but disjoint from the training set, so these experiments evaluate in-distribution generalization; the Out-Of-Distribution (OOD) setting is studied in Section 4.4. Within each model setting, all optimizers share the same initialization and training data stream. Following Hu et al.



(a) Validation losses of Adam and Muon

(b) Values of r_{CCC} across training

(c) Values of r_{diff} across training

Figure 1: Empirical evidence of RGI among optimizers. Panel (a) plots the validation losses of the 124M GPT models trained with Adam and Muon at the 10,000-step checkpoint, evaluated on 20,480 validation batches. Panel (b) reports $r_{\mathrm{CCC}}(\mathrm{Adam}, \cdot)$ ($\uparrow$) for the other five optimizers over training, all of which rise above 0.95 after 2,000 steps. Panel (c) reports $r_{\mathrm{diff}}(\mathrm{Adam}, \cdot)$ ($\downarrow$) for the other five optimizers over training, all of which drop below 0.05 after 2,000 steps. These results show that RGI holds across all considered optimizers after sufficient training when they are trained from the same initialization and on the same training data stream.

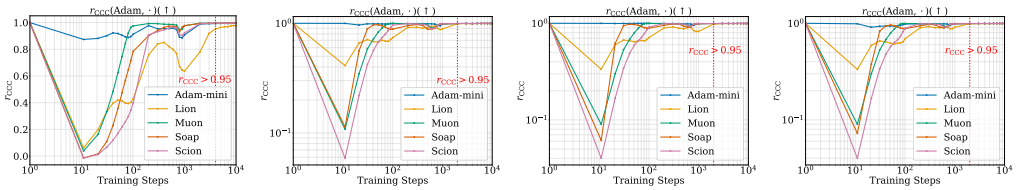
[17], Wen et al. [18], we use the Warmup-Stable-Decay learning-rate scheduler and tune the learning rate separately for each optimizer. We report the 124M results in the main text and defer the 0.7B and 4.5B results, which lead to the same conclusions, to Appendix C. Across 10 random seeds, we report the worst-case maximum r_{diff} and minimum r_{CCC}, providing evidence that RGI is robust to training randomness and consistent with the almost-sure requirement in Definition 1. Additional experimental details are provided in Appendix B.

4.1 RGI Holds Across Optimizers

In this section, we vary the optimizer and its hyperparameters while fixing the model initialization and training data stream, and show that ARGI approximately holds across optimizers. For simplicity, we abbreviate $r_{\mathrm{diff}}(\mathfrak{T}_{\mathsf{Adam}}, \cdot)$ and $r_{\mathrm{CCC}}(\mathfrak{T}_{\mathsf{Adam}}, \cdot)$ as $r_{\mathrm{diff}}(\mathsf{Adam}, \cdot)$ and $r_{\mathrm{CCC}}(\mathsf{Adam}, \cdot)$, respectively.

Figure 1 provides empirical evidence for approximate RGI across optimizers with identical initial parameters and the same training data stream. Figure 1(a) shows that, after 10,000 training steps, the validation losses of models trained with Muon and Adam across validation batches are well fitted by $y = x - 0.07$, indicating a nearly constant generalization gap. Results for the other optimizers are deferred to Appendix C and yield the same conclusion. Taking Adam as the baseline, Figures 1(c) and 1(b) further show that approximate RGI emerges across optimizers after 2,000 training steps, with $r_{\mathrm{diff}} < 0.05$ and $r_{\mathrm{CCC}} > 0.95$, as marked by the red dotted lines. Results using other optimizers as the baseline are reported in Appendix C and are consistent with this finding. Overall, these results suggest that ARGI emerges across optimizers on the in-distribution validation set after the early stage of training when each optimizer uses its best learning rate, while the remaining hyperparameters are set to their defaults and the initialization and training data stream are shared.

We next examine whether RGI continues to hold under variations in optimizer hyperparameters, including learning rates, momentum coefficients, learning-rate schedulers, and weight decay. To avoid confounding changes in both models being compared, we keep Adam fixed under the setting in Figure 1 and vary only the hyperparameters of the other optimizers. We first vary the learning rate of each optimizer A over $0.25\eta^{\mathsf{A}}$, $0.5\eta^{\mathsf{A}}$, $0.75\eta^{\mathsf{A}}$, and $1.25\eta^{\mathsf{A}}$, where η^{A} denotes its best learning rate. We exclude larger multipliers, such as $1.5\eta^{\mathsf{A}}$, because some optimizers diverge under these settings. Figures 2(a) and 2(b) report the r_{CCC} results for $0.25\eta^{\mathsf{A}}$ and $1.25\eta^{\mathsf{A}}$, while the remaining results are deferred to Appendix C. The different times at which r_{CCC} crosses the threshold 0.95 show that the learning rate affects when approximate RGI emerges. Nevertheless, r_{CCC} eventually grows above 0.95 for all tested learning rates. We next vary the momentum coefficient γ over 0.85, 0.9, and 0.95. Figures 2(c) and 2(d) show that ARGI persists across these momentum values, with the $\gamma = 0.9$ and corresponding r_{diff} results deferred to Appendix C. We further ablate the learning-rate scheduler and weight decay in Appendix C, obtaining the same conclusion. Overall, ARGI is robust across the tested optimizer hyperparameters, although these hyperparameters can affect when it emerges.



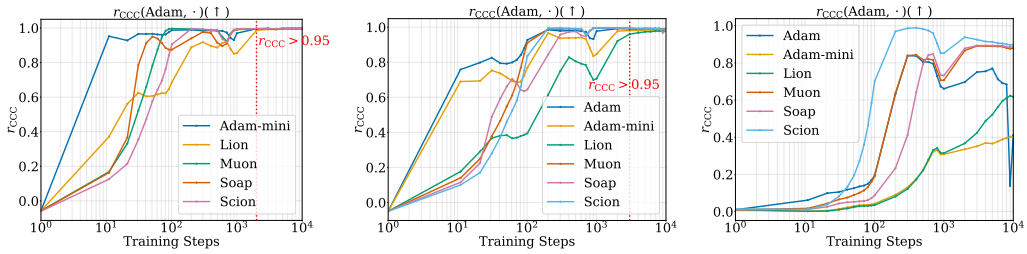
(a) r_{CCC} for $0.25\eta^{\mathsf{A}}$. (b) r_{CCC} for $1.25\eta^{\mathsf{A}}$. (c) r_{CCC} for $\gamma = 0.85$. (d) r_{CCC} for $\gamma = 0.95$.

Figure 2: RGI across optimizers under different hyperparameter settings. Panels (a) and (b) plot r_{CCC} under different learning-rate multipliers, while panels (c) and (d) plot r_{CCC} under different momentum values. RGI persists across all tested learning-rate and momentum settings.

4.2 RGI Holds Across Moderately Different Architectures

In this section, we study whether ARGI holds across model architectures. Consistent with Section 4.1, we train models with different architectures using different optimizers, thereby examining ARGI under joint variations in optimizer and architecture. Specifically, we consider two settings: (i) models with the same architectural components but different initializations and model sizes, and (ii) models with different architectural components.

We now examine whether RGI persists across different initializations and model sizes while keeping the architectural components fixed. First, we train LLMs from independent initializations $\theta_0, \theta_0' \in \Theta$, drawn from the same initialization distribution, using $\mathsf{A} = \mathrm{Adam}$ and A' chosen from Adam-mini, Lion, Muon, Soap, and Scion. Figure 3(a) shows that, unlike the shared-initialization setting in Figure 1, r_{CCC} initially deviates substantially from RGI; nevertheless, approximate RGI emerges after about 2,000 training steps. Second, we vary the model size from the baseline hidden-state dimension 768 to 384 and 64. Figures 3(b) and 3(c) show that RGI continues to hold between models with dimensions 768 and 384, but fails when the dimension is reduced to 64. The corresponding r_{diff} results in Appendix C lead to the same conclusion.

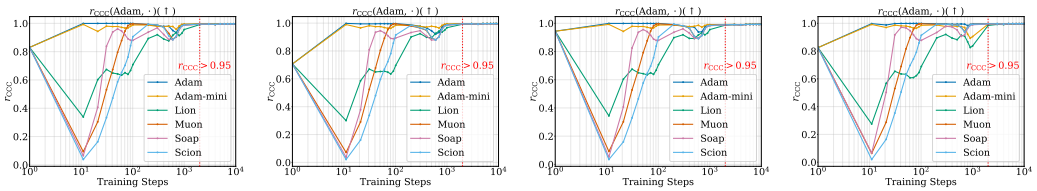


(a) r_{CCC} under different initial parameters. (b) r_{CCC} for hidden state dimensions 768 and 384. (c) r_{CCC} for hidden state dimensions 768 and 64.

Figure 3: RGI across optimizers, initializations, and model sizes. Panel (a) plots r_{CCC} across optimizers with different initializations. Panels (b) and (c) plot r_{CCC} across models with hidden-state dimensions 768, 384, and 64. RGI continues to hold across different initializations and moderate changes in model size.

We next examine whether RGI holds between models with different architectural components. First, we ablate the attention design. While our previous experiments use Full Attention (FA) with RoPE [15], we additionally consider FA with NoPE [19], ALiBi [20], and SWA with RoPE. We train each variant with different optimizers and compare it with the baseline RoPE FA model trained with Adam. Figures 4(a)–4(c) report the corresponding r_{CCC} values and show that RGI continues to hold across different attention designs and optimizers. Second, we ablate the FFN design. While the baseline uses GELU [16], we additionally consider Squared-ReLU (SReLU) [21] and SiLU [4]. We train both variants with all optimizers and compare them with the baseline GELU model trained with Adam. Figure 4(d) reports the SReLU results, while the SiLU results and additional comparisons between gated and non-gated FFNs are deferred to Appendix C; all exhibit ARGI. Together, these results show that RGI persists across different attention and FFN designs as well as optimizers.

In conclusion, ARGI persists under joint variations in optimizers and moderate architectural changes, though these choices can affect its emergence time. Appendix D further shows that RGI explains the shared training-loss curve shapes across optimizers and architectures.



(a) Values of r_{CCC} between FA models with RoPE and NoPE.

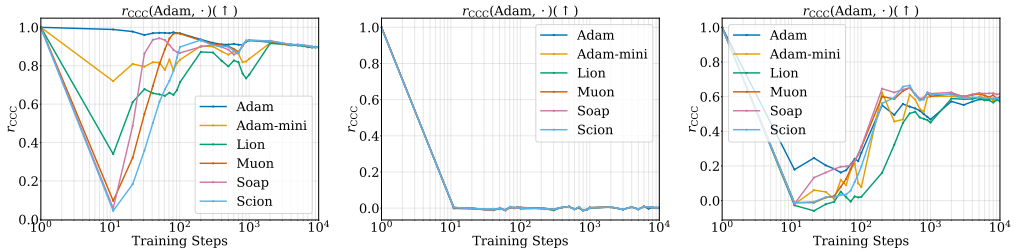
(b) Values of r_{CCC} between FA models with RoPE and ALiBi.

(c) Values of r_{CCC} between FA and SWA models.

(d) Values of r_{CCC} between models with GELU and SReLU.

Figure 4: RGI across optimizers and models with different attention and FFN modules. Panels (a)–(c) plot r_{CCC} between models using RoPE FA trained with Adam and models using NoPE FA, ALiBi FA, and RoPE SWA, each trained with all optimizers, respectively. Panel (d) plots r_{CCC} between models using GELU and SReLU activations. RGI continues to hold across different attention and FFN designs.

4.3 RGI is Strongly Influenced by Pretraining Data Streams



(a) Values of r_{CCC} with D_l' from C4.

(b) Values of r_{CCC} with D_l' from Arxiv.

(c) Values of r_{CCC} with D_l' from CodeParrot.

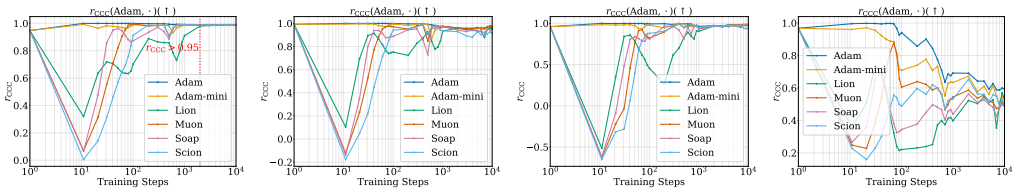
Figure 5: RGI across optimizers trained on different data streams. The panels plot r_{CCC} when Adam is trained on FineWeb and the compared optimizer is trained on C4 (Panel (a)), ArXiv (Panel (b)), or CodeParrot (Panel (c)). The results show that RGI is strongly affected by the training data stream: changing the data stream can substantially weaken, or even eliminate, RGI across optimizers.

We next explore whether ARGI emerges when the training data stream varies, jointly or independently of the optimizer, and show that it can fail even when the optimizer and architecture are held fixed. Specifically, we examine whether RGI emerges between $\theta_t^{\mathfrak{T}_{\mathsf{Adam}}} = F(\mathfrak{T}_{\mathsf{Adam}}, t)$, where $\mathfrak{T}_{\mathsf{Adam}} = (\mathsf{Adam}, \theta_0, D_1)$, and $\theta_t^{\mathfrak{T}'} = F(\mathfrak{T}', t)$, where $\mathfrak{T}' = (\mathsf{A}', \theta_0, D_1')$. Here, D_1 is drawn from FineWeb, while D_1' is drawn from one of three alternative datasets: C4 [22], the ArXiv subset of RedPajama [23], and CodeParrot [24]. FineWeb and C4 are both large-scale web-text datasets covering diverse topics and are derived, at least in part, from Common Crawl [25], suggesting relatively high distributional similarity. By contrast, ArXiv consists of academic papers from arXiv, while CodeParrot consists of source code collected from GitHub, making their distributions more distinct from FineWeb.

Figure 5 shows that the training data stream strongly affects the emergence of RGI. For C4, Figure 5(a) shows that r_{CCC} first decreases, then rises to a relatively large value, and finally declines again; the intermediate increase may reflect the greater distributional similarity between FineWeb and C4. In contrast, for ArXiv and CodeParrot, Figures 5(b) and 5(c) show that r_{CCC} remains below 0.95 throughout training even when the same optimizer Adam is used, indicating that RGI does not emerge across these training data streams. The corresponding r_{diff} results in Appendix C show the same trend. Overall, changes in the training data distribution can substantially weaken or even prevent the emergence of RGI.

4.4 RGI Persists under Moderate Validation Shifts

In this section, we ablate the effect of the validation data D_{val} on RGI across optimizers and architectures. For clarity, we focus on FA and SWA as representative architectural variants; additional results are provided in Appendix C. All models are trained on FineWeb, while r_{diff} and r_{CCC} are evaluated on validation data from other sources. As in Section 4.3, we consider C4, ArXiv, and CodeParrot, and additionally construct a Random dataset by sampling each token i.i.d. uniformly



(a) Values of r_{CCC} with D_{val} from C4.
(b) Values of r_{CCC} with D_{val} from ArXiv.
(c) Values of r_{CCC} with D_{val} from CodeParrot.
(d) Values of r_{CCC} with D_{val} from Random.

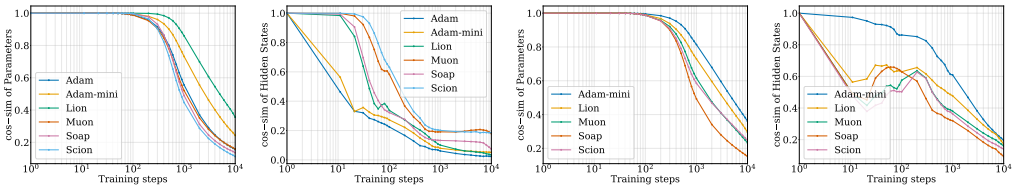
Figure 6: RGI across optimizers and architectures evaluated on different validation datasets. The panels plot r_{CCC} evaluated on C4, ArXiv, CodeParrot, and Random. These results show that RGI can transfer to related validation distributions, but becomes weaker on more distributionally distinct validation datasets.

from the vocabulary, yielding an extreme OOD setting with no meaningful semantic structure. Figure 6 reports the corresponding r_{CCC} values, comparing SWA models trained with various optimizers against the FA baseline trained with Adam. r_{CCC} remains large on C4, while on ArXiv and CodeParrot it is smaller but generally increases as training progresses. In contrast, the Random dataset shows no clear emergence of RGI. These results suggest that validation-distribution shifts weaken ARGI: moderate shifts preserve the trend, whereas sufficiently extreme shifts can prevent its emergence. The corresponding r_{diff} results in Appendix C support the same conclusion.

5 Relationships to the NTK and MF Regimes

Section 4 shows that approximate RGI holds across a wide range of optimizers and architectures. We now examine whether this phenomenon can be explained by two well-known theoretical regimes of training dynamics: the NTK and MF regimes. We focus on RGI across optimizers, since different architectures have parameterizations and intermediate representations that are not directly comparable.

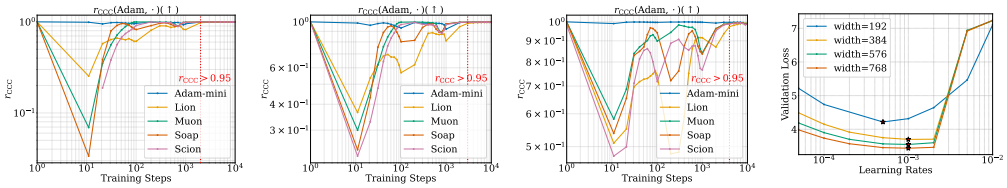
5.1 RGI Emerges Beyond the NTK Regime



(a) cos-sim between parameters within optimizers.
(b) cos-sim between hidden states within optimizers.
(c) cos-sim between parameters cross optimizers.
(d) cos-sim between hidden states cross optimizers.

Figure 7: Within- and cross-optimizer $\mathrm{cos\text{-}sim}(\uparrow)$ for parameters and hidden states. Panels (a)–(b) compare $\theta_0^{\mathfrak{T}}$ with $\theta_t^{\mathfrak{T}}$, while Panels (c)–(d) compare $\theta_t^{\mathfrak{T}}$ with $\theta_t^{\mathfrak{T}'}$. Hidden-state similarities are averaged over layers $l \in [L]$. Within-optimizer comparisons show that all optimizers learn nontrivial features throughout training, whereas cross-optimizer comparisons show that different optimizers learn distinct features.

In this section, we examine whether approximate RGI across optimizers sharing the same initialization and training data stream can be explained by the NTK regime. In this regime, sufficiently wide models under suitable Gaussian initialization are well approximated by their linearization around initialization, with the induced kernel remaining nearly constant [26, 27]. This is also known as the lazy-learning regime [28], whose characteristic signature is that parameters and representations remain close to initialization: $\|\theta_t - \theta_0\|_2 \ll \|\theta_0\|_2$ and $\|h(l,\theta_t,d) - h(l,\theta_0,d)\|_2 \ll \|h(l,\theta_0,d)\|_2$ for relevant t, $l \in [L]$, and $d \in \mathcal{D}_{\mathrm{val}}$ [29]. We test these signatures using cosine similarity and relative ℓ_2 difference, $\mathrm{cos\text{-}sim}(x,x') = x^\top x'/(\|x\|_2\|x'\|_2)$ and $r_{\ell_2}(x,x') = 2|x-x'|_2/(\|x\|_2 + \|x'\|_2)$, for both parameters and hidden states, comparing them across training steps within each optimizer and



(a) Values of r_{CCC} at width 576. (b) Values of r_{CCC} at width 384. (c) Values of r_{CCC} at width 192. (d) Validation losses of different learning rates of Adam with various widths.

Figure 8: RGI across models of different widths. Panels (a)–(c) plot r_{CCC} for models with different widths, showing that RGI holds across all tested widths. Panel (d) plots the validation loss of Adam at 10,000 steps across learning rates and widths, showing that the width-192 model exhibits behavior inconsistent with the MF regime.

across optimizers at the same step. We report cosine-similarity results in the main text and defer the relative-ℓ_2 results, which show consistent trends, to Appendix C.

Figures 7(a)–7(b) report within-optimizer cosine similarities between initialization and step t for both parameters, $\cos\text{-sim}(\theta_0^{\mathfrak{T}}, \theta_t^{\mathfrak{T}})$, and hidden states, $\cos\text{-sim}(h(l, \theta_0^{\mathfrak{T}}, d), h(l, \theta_t^{\mathfrak{T}}, d))$, averaged over layers $l \in [L]$, validation examples $d \in \mathcal{D}_{\mathrm{val}}$, and random seeds. After 5,000 training steps, both parameters and hidden states deviate substantially from initialization, suggesting that the observed RGI cannot be explained by the NTK lazy-learning regime alone. We further compare models trained by different optimizers at the same step t. Figures 7(c)–7(d) report the cosine similarities of parameters and hidden states between Adam and the other optimizers, both of which decrease as training proceeds. Thus, approximate RGI cannot be attributed either to NTK-style fixed representations or to different optimizers converging to shared parameters or hidden representations.

5.2 RGI Emerges Beyond the MF Regime

In this section, we examine whether approximate RGI across optimizers with identical initializations and training data streams can be explained by the MF regime. In this regime, the parameter distribution evolves nontrivially in the infinite-width limit, allowing both parameters and features to change substantially during training [30, 31, 32]. Starting from the 124M GPT model with hidden dimension 768 used in Section 4, we progressively reduce the width to 576, 384, and 192, where 192 is one quarter of the baseline width, and test whether approximate RGI persists. We also grid search the Adam learning rate at each width, using the width dependence of the optimal learning rate as an empirical diagnostic of consistency with the MF regime.

Figures 8(a)–8(c) evaluate RGI across optimizers at different fixed widths, unlike Figures 3(b) and 3(c), which compare models across widths. Approximate RGI persists even at width 192, one quarter of the standard GPT-2 width; the corresponding r_{diff} results in Appendix C lead to the same conclusion. To assess whether this narrow model remains consistent with the MF regime, Figure 8(d) plots Adam's validation loss after 10,000 steps across widths and learning rates, with red stars marking the optimal learning rates. The width-192 model exhibits a qualitatively different validation-loss–learning-rate profile from the width-768, 576, and 384 models, suggesting that it is no longer well described by the same MF-style approximation. Thus, approximate RGI can persist even beyond the regime where the MF approximation appears adequate.

6 Case Study of Over-Parameterized Quadratic Models

Section 5 shows that RGI cannot be explained by NTK or MF regimes alone. We fill this theoretical gap in this section by studying the behavior of GD, Adam, and Muon on a quadratic problem. Given a linear target response $y(x) = Hx$ with $H \in \mathbb{R}^{q\times r}, x \in \mathbb{R}^r$ and $q \geq 3r$, we adopt a two-layer overparameterized model $f_{A,B}(x) = ABx$ to learn it, where $A \in \mathbb{R}^{q\times s}, B \in \mathbb{R}^{s\times r}$ with $s \geq 3r$. With the dataset $\{(y_i, x_i)\}_{i=1}^n$, we train the parameters A, B via the loss $L(A, B) = 1/2n \cdot \sum_{i=1}^n \|y_i - ABx_i\|_2^2$. We require the target matrix H to have the following structure.

Assumption 1 *The target matrix $H \in \mathbb{R}^{q\times r}$ satisfies $H^\top H = \mu^2 I_r, \mathrm{sign}(H) = \kappa H$ for some $\mu > 0$ and $\kappa > 0$, where the sign map is applied entrywise with $\mathrm{sign}(0) = 0$.*

This assumption imposes two requirements. First, the columns of H are orthogonal and have the same norm. Second, all nonzero entries of H must have the same magnitude. This assumption is satisfied by a broad class of matrices; for example, the columns of H may form either a one-hot basis or a Hadamard basis.

Assumption 2 *The covariates $\{x_i\}_{i=1}^n$ are isotropic, i.e., $\sum_{i=1}^n x_i x_i^\top / n = \sigma_x^2 I_r$ for some $\sigma_x > 0$.*

This assumption requires the training covariates to cover the feature space approximately isotropically. We emphasize that Assumptions 1 and 2 define a stylized setting under which RGI can hold across optimizers. In Appendix F, we provide empirical evidence that these assumptions are relevant to practical LLM training; relaxing them is left for future work.

We then optimize the parameters A and B of our model with three optimizers, GD, Adam, and Muon. Given any parameter $W \in \{A, B\}$, these optimizers update the parameter W_t at step t as follows.

- GD updates the parameter according to the gradient, i.e., $W_{t+1}^{\mathsf{GD}} = W_t^{\mathsf{GD}} - \eta_t \nabla_W L(W_t^{\mathsf{GD}})$. Here, the notation $L(W_t^{\mathsf{GD}})$ suppresses its dependency on other parameters for conciseness.
- With $\gamma_1 = \gamma_2 = 0$, Adam updates the parameter with the coordinate-wise normalized gradient: $W_{t+1}^{\mathsf{Adam}} = W_t^{\mathsf{Adam}} - \gamma_t \mathtt{sgn}(\nabla_W L(W_t^{\mathsf{Adam}}))$, where $\mathtt{sgn}(\cdot)$ denotes the entrywise sign operator.
- With momentum set to 0, Muon updates the parameters using the spectrally normalized gradient: $W_{t+1}^{\mathsf{Muon}} = W_t^{\mathsf{Muon}} - \tau_t \mathtt{spec}(\nabla_W L(W_t^{\mathsf{Muon}}))$, where $\mathtt{spec}(G) = UV^\top$ for $G = U\Sigma V^\top$, which normalizes all nonzero singular values Σ of G to 1.

Theorem 2 (A stylized realization of exact RGI with low parameter cosine) *For each optimizer $\mathsf{A} \in \{\mathsf{GD}, \mathsf{Adam}, \mathsf{Muon}\}$, set the initial parameters as $A_0^{\mathsf{A}} = 0, B_0^{\mathsf{A}} = [b_0^{\mathsf{A}} I_r, 0]^\top$ with any $b_0^{\mathsf{A}} > 0$ and set the step size to constants as*

$$\eta_t = \eta > 0,\ \gamma_t = \gamma < \min\{b_0^{\mathsf{Adam}}, (\kappa b_0^{\mathsf{Adam}} + \sqrt{\kappa})^{-1}\},\ \tau_t = \tau < \min\{b_0^{\mathsf{Muon}}, (b_0^{\mathsf{Muon}}/\mu + \mu^{-1/2})^{-1}\}.$$

Under Assumptions 1 and 2, there exist tie-breaking rules on zero-gradient, for any two optimizers $\mathsf{A}, \mathsf{A}' \in \{\mathsf{GD}, \mathsf{Adam}, \mathsf{Muon}\}$, any two inputs x, x' with $\|x\|_2 = \|x'\|_2 = \rho$, and any $t \geq 0$, we have

$$L_t^{\mathsf{A}}(x) - L_t^{\mathsf{A}}(x') = L_t^{\mathsf{A}'}(x) - L_t^{\mathsf{A}'}(x').$$

Moreover, let $\theta_t^{\mathsf{A}} = (\mathrm{vec}(A_t^{\mathsf{A}}), \mathrm{vec}(B_t^{\mathsf{A}}))$. At the same time, with cosine similarity defined to be zero whenever either vector is zero, the following holds as t increases.

$$|\text{cos-sim}(\theta_t^{\mathsf{GD}}, \theta_t^{\mathsf{Adam}})| = O(1/t), |\text{cos-sim}(\theta_t^{\mathsf{GD}}, \theta_t^{\mathsf{Muon}})| = O(1/t), |\text{cos-sim}(\theta_t^{\mathsf{Adam}}, \theta_t^{\mathsf{Muon}})| = O(1/t^2).$$

This theorem shows that RGI holds across GD, Muon, Adam, and different initializations under Assumptions 1 and 2, with the covariate x playing the same role as the token d in Definition 1. These results support the empirical findings in Sections 4.1 and 4.2. The theorem also permits a broad range of learning rates, imposing only upper-bound constraints on those of Muon and Adam, consistent with the robustness of RGI to optimizer hyperparameters observed in Section 4.1. Moreover, we show that the cosine similarity between parameters decreases during training, matching the low parameter similarity reported in Section 5.1. Overall, our analysis identifies a class of quadratic problems in which RGI is robust across optimizers, initializations, and learning rates.

7 Conclusion

We study how optimizers, architectures, and training data streams affect model performance through the lens of relative generalization. We show that RGI persists under joint variations in optimizers and architectures, which largely preserve relative generalization and induce approximately uniform shifts in token-wise losses, whereas changes in the training data stream can substantially alter it. Neither the NTK nor the MF regime alone explains RGI; instead, we identify a class of quadratic problems in which RGI emerges robustly across initializations and learning rates. Our analysis focuses on autoregressive LLMs, leaving other model classes, such as diffusion models, for future work.

AI USE STATEMENT

In this work, we used generative AI tools to assist with translation. We have not used generative AI to help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, implement methods, support qualitative and thematic data analysis, design or provide feedback on our research methodology or experiments, or interpret results. Additionally, we used generative AI tools to edit this research paper to improve readability. We have reviewed all AI-assisted work. All AI-assisted translations and textual edits to add readability were manually reviewed and verified by the authors to guarantee the accuracy of our technical contributions and to preserve the original meaning. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies numerical behaviors in large-scale neural network training and does not involve human subjects or sensitive information. All experiments are conducted on publicly available datasets and standard language modeling benchmarks. We do not release or analyze any data containing personally identifiable information. The proposed analyses and experiments are intended to improve the reliability and understanding of machine learning systems. We are not aware of any foreseeable ethical concerns arising from the methods or findings presented in this work.

REPRODUCIBILITY STATEMENT

The paper discloses the main experimental setup needed to reproduce the results: Sections 3–4 describe the evaluated optimizers, model sizes, datasets, validation setup, evaluation metrics, and seed-based reporting protocol. Appendix B provides further implementation details, including the tokenizer, sequence length, batch size, learning-rate search space, optimizer hyperparameters, model configurations, and validation-token counts.

REFERENCES

[1] Nikhil Vyas, Depen Morwani, Rosie Zhao, Mujin Kwun, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham Kakade. Soap: Improving and stabilizing shampoo using adam. *arXiv preprint arXiv:2409.11321*, 2024.

[2] Keller Jordan, Yuchen Jin, Vlado Boza, You Jiacheng, Franz Cecista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. *URL https://kellerjordan. github. io/posts/muon*, 6, 2024.

[3] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, volume 2024, pages 21875–21895, 2024.

[4] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural networks*, 107:3–11, 2018.

[5] Shaobo Wang, Xuan Ouyang, Tianyi Xu, Yuzheng Hu, Jialin Liu, Guo Chen, Tianyu Zhang, Junhao Zheng, Kexin Yang, Xingzhang Ren, et al. Opus: towards efficient and principled data selection in large language model pre-training in every iteration. *arXiv preprint arXiv:2602.05400*, 2026.

[6] Stephan Mandt, Matthew D Hoffman, and David M Blei. Stochastic gradient descent as approximate bayesian inference. *Journal of Machine Learning Research*, 18(134):1–35, 2017.

[7] Chi Jin, Rong Ge, Praneeth Netrapalli, Sham M Kakade, and Michael I Jordan. How to escape saddle points efficiently. In *International conference on machine learning*, pages 1724–1732. PMLR, 2017.

[8] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

[10] Yushun Zhang, Congliang Chen, Ziniu Li, Tian Ding, Chenwei Wu, Diederik P Kingma, Yinyu Ye, Zhi-Quan Luo, and Ruoyu Sun. Adam-mini: Use fewer learning rates to gain more. *arXiv preprint arXiv:2406.16793*, 2024.

[11] Thomas Pethick, Wanyun Xie, Kimon Antonakopoulos, Zhenyu Zhu, Antonio Silveti-Falls, and Volkan Cevher. Training deep learning models with norm-constrained lmos. *arXiv preprint arXiv:2502.07529*, 2025.

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

[13] Lawrence I-Kuei Lin. A concordance correlation coefficient to evaluate reproducibility. *Biometrics*, pages 255–268, 1989.

[14] Guilherme Penedo, Hynek Kydlícek, Loubna Ben Allal, and Thomas Wolf. Fineweb: decanting the web for the finest text data at scale. *HuggingFace. Accessed: Jul*, 12, 2024.

[15] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[16] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[17] Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, et al. Minicpm: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*, 2024.

[18] Kaiyue Wen, Zhiyuan Li, Jason Wang, David Hall, Percy Liang, and Tengyu Ma. Understanding warmup-stable-decay learning rates: A river valley loss landscape perspective. *arXiv preprint arXiv:2410.05192*, 2024.

[19] Amirhossein Kazemnejad, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Payel Das, and Siva Reddy. The impact of positional encoding on length generalization in transformers. *Advances in Neural Information Processing Systems*, 36:24892–24928, 2023.

[20] Ofir Press, Noah A Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. *arXiv preprint arXiv:2108.12409*, 2021.

[21] Jason M Klusowski and Andrew R Barron. Approximation by combinations of relu and squared relu ridge functions with ℓ^1 and ℓ^0 controls. *arXiv preprint arXiv:1607.07819*, 2016.

[22] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[23] Maurice Weber, Daniel Y Fu, Quentin Anthony, Yonatan Oren, Shane Adams, Anton Alexandrov, Xiaozhong Lyu, Huu Nguyen, Xiaozhe Yao, Virginia Adams, et al. Redpajama: an open dataset for training large language models. *Advances in neural information processing systems*, 37:116462–116492, 2024.

[24] CodeParrot. Github code dataset. https://huggingface.co/datasets/codeparrot/github-code, 2022.

[25] Stefan Baack. A critical analysis of the largest source for generative ai training data: Common crawl. In *Proceedings of the 2024 ACM conference on fairness, accountability, and transparency*, pages 2199–2208, 2024.

[26] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

[27] Eugene Golikov, Eduard Pokonechnyy, and Vladimir Korviakov. Neural tangent kernel: A survey. *arXiv preprint arXiv:2208.13614*, 2022.

[28] Mario Geiger, Stefano Spigler, Arthur Jacot, and Matthieu Wyart. Disentangling feature and lazy training in deep neural networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(11):113301, 2020.

[29] Simon S Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. *arXiv preprint arXiv:1810.02054*, 2018.

[30] Song Mei, Andrea Montanari, and Phan-Minh Nguyen. A mean field view of the landscape of two-layer neural networks. *Proceedings of the National Academy of Sciences*, 115(33): E7665–E7671, 2018.

[31] Song Mei, Theodor Misiakiewicz, and Andrea Montanari. Mean-field theory of two-layers neural networks: dimension-free bounds and kernel limit. In *Conference on learning theory*, pages 2388–2464. PMLR, 2019.

[32] Justin Sirignano and Konstantinos Spiliopoulos. Mean field analysis of neural networks: A central limit theorem. *Stochastic Processes and their Applications*, 130(3):1820–1852, 2020.

[33] Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *arXiv preprint arXiv:2504.13837*, 2025.

[34] Emmy Liu, Amanda Bertsch, Lintang Sutawika, Lindia Tjuatja, Patrick Fernandes, Lara Marinov, Michael Chen, Shreya Singhal, Carolin Lawrence, Aditi Raghunathan, et al. Not-just-scaling laws: Towards a better understanding of the downstream impact of language model design decisions. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 16407–16438, 2025.

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

[36] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[37] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5—RMSProp: Divide the gradient by a running average of its recent magnitude. 2012. URL https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.

[38] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[39] Timothy Dozat. Incorporating nesterov momentum into adam. 2016.

[40] Jerry Ma and Denis Yarats. Quasi-hyperbolic momentum and adam for deep learning. *arXiv preprint arXiv:1810.06801*, 2018.

[41] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*, 2019.

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

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

[44] Kwangjun Ahn, Byron Xu, Natalie Abreu, Ying Fan, Gagik Magakyan, Pratyusha Sharma, Zheng Zhan, and John Langford. Dion: Distributed orthonormalized updates. *arXiv preprint arXiv:2504.05295*, 2025.

[45] Zichong Li, Liming Liu, Chen Liang, Weizhu Chen, and Tuo Zhao. Normuon: Making muon more efficient and scalable. *arXiv preprint arXiv:2510.05491*, 2025.

[46] Peng Cheng, Jiucheng Zang, Qingnan Li, Liheng Ma, Yufei Cui, Yingxue Zhang, Boxing Chen, Ming Jian, and Wen Tong. Trasmuon: Trust-region adaptive scaling for orthogonalized momentum optimizers. *arXiv preprint arXiv:2602.13498*, 2026.

[47] Rong Shan, Aofan Yu, Bo Chen, Kuo Cai, Qiang Luo, Ruiming Tang, Han Li, Weiwen Liu, Weinan Zhang, and Jianghao Lin. Muonrec: Shifting the optimizer paradigm beyond adam in scalable generative recommendation. *arXiv preprint arXiv:2603.00416*, 2026.

[48] Tianyu Pang, Yujie Fang, Zihang Liu, Shenyang Deng, Lei Hsiung, Shuhua Yu, and Yaoqing Yang. Htmuon: Improving muon via heavy-tailed spectral correction. *arXiv preprint arXiv:2603.10067*, 2026.

[49] Shenyang Deng, Zhuoli Ouyang, Tianyu Pang, Zihang Liu, Ruochen Jin, Shuhua Yu, and Yaoqing Yang. Rmnp: Row-momentum normalized preconditioning for scalable matrix-based optimization. *arXiv preprint arXiv:2603.20527*, 2026.

[50] Yechen Zhang, Shuhao Xing, Junhao Huang, Kai Lv, Yunhua Zhou, Xipeng Qiu, Qipeng Guo, and Kai Chen. Mousse: Rectifying the geometry of muon with curvature-aware preconditioning. *arXiv preprint arXiv:2603.09697*, 2026.

[51] Ben S Southworth and Stephen Thomas. Beyond muon: Mud (momentum decorrelation) for faster transformer training. *arXiv preprint arXiv:2603.17970*, 2026.

[52] Ruijie Zhang, Yequan Zhao, Ziyue Liu, Zhengyang Wang, and Zheng Zhang. Muon+: Towards better muon via one additional normalization step. *arXiv preprint arXiv:2602.21545*, 2026.

[53] Ziyue Liu, Ruijie Zhang, Zhengyang Wang, Yequan Zhao, Yupeng Su, Zi Yang, and Zheng Zhang. Muon^2: Boosting muon via adaptive second-moment preconditioning. *arXiv preprint arXiv:2604.09967*, 2026.

[54] Minxin Zhang, Yuxuan Liu, and Hayden Scheaffer. Adam improves muon: Adaptive moment estimation with orthogonalized momentum. *arXiv preprint arXiv:2602.17080*, 2026.

[55] Hong Liu, Zhiyuan Li, David Hall, Percy Liang, and Tengyu Ma. Sophia: A scalable stochastic second-order optimizer for language model pre-training. *arXiv preprint arXiv:2305.14342*, 2023.

[56] Natalie Abreu, Nikhil Vyas, Sham Kakade, and Depen Morwani. The potential of second-order optimization for llms: A study with full gauss-newton. *arXiv preprint arXiv:2510.09378*, 2025.

[57] Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *International conference on machine learning*, pages 4596–4604. PMLR, 2018.

[58] Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. Galore: Memory-efficient llm training by gradient low-rank projection. *arXiv preprint arXiv:2403.03507*, 2024.

[59] Hanqing Zhu, Zhenyu Zhang, Wenyan Cong, Xi Liu, Sem Park, Vikas Chandra, Bo Long, David Z Pan, Zhangyang Wang, and Jinwon Lee. Apollo: Sgd-like memory, adamw-level performance. *Proceedings of Machine Learning and Systems*, 7, 2025.

[60] Ahmed Khaled, Kaan Ozkara, Tao Yu, Mingyi Hong, and Youngsuk Park. Muonbp: Faster muon via block-periodic orthogonalization. *arXiv preprint arXiv:2510.16981*, 2025.

[61] Xiaoxin He, Fuzhao Xue, Xiaozhe Ren, and Yang You. Large-scale deep learning optimizations: A comprehensive survey. *arXiv preprint arXiv:2111.00856*, 2021.

[62] Zhongwei Wan, Xin Wang, Che Liu, Samiul Alam, Yu Zheng, Jiachen Liu, Zhongnan Qu, Shen Yan, Yi Zhu, Quanlu Zhang, et al. Efficient large language models: A survey. *arXiv preprint arXiv:2312.03863*, 2023.

[63] Ruslan Abdulkadirov, Pavel Lyakhov, and Nikolay Nagornov. Survey of optimization algorithms in modern neural networks. *Mathematics*, 11(11):2466, 2023.

[64] Jiahang Zhou, Yanyu Chen, Zicong Hong, Wuhui Chen, Yue Yu, Tao Zhang, Hui Wang, Chuanfu Zhang, and Zibin Zheng. Training and serving system of foundation models: A comprehensive survey. *IEEE Open Journal of the Computer Society*, 5:107–119, 2024.

[65] Junyu Luo, Bohan Wu, Xiao Luo, Zhiping Xiao, Yiqiao Jin, Rong-Cheng Tu, Nan Yin, Yifan Wang, Jingyang Yuan, Wei Ju, et al. A survey on efficient large language model training: From data-centric perspectives. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 30904–30920, 2025.

[66] Kaiyuan Tian, Linbo Qiao, Baihui Liu, Gongqingjian Jiang, and Dongsheng Li. A survey on memory-efficient large-scale model training in ai for science. *arXiv e-prints*, pages arXiv–2501, 2025.

[67] Ziyang Xiao, Jingrong Xie, Lilin Xu, Shisi Guan, Jingyan Zhu, Xiongwei Han, Xiaojin Fu, WingYin Yu, Han Wu, Wei Shi, et al. A survey of optimization modeling meets llms: Progress and future directions. *arXiv preprint arXiv:2508.10047*, 2025.

[68] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[69] Jonathan S Rosenfeld, Amir Rosenfeld, Yonatan Belinkov, and Nir Shavit. A constructive prediction of the generalization error across scales. *arXiv preprint arXiv:1909.12673*, 2019.

[70] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, DDL Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 10, 2022.

[71] Chaofan Tao, Qian Liu, Longxu Dou, Niklas Muennighoff, Zhongwei Wan, Ping Luo, Min Lin, and Ngai Wong. Scaling laws with vocabulary: Larger models deserve larger vocabularies. *Advances in Neural Information Processing Systems*, 37:114147–114179, 2024.

[72] Song Bian, Tao Yu, Shivaram Venkataraman, and Youngsuk Park. Scaling laws meet model architecture: Toward inference-efficient llms. *arXiv preprint arXiv:2510.18245*, 2025.

[73] Rosie Zhao, Depen Morwani, David Brandfonbrener, Nikhil Vyas, and Sham Kakade. Deconstructing what makes a good optimizer for language models. *arXiv preprint arXiv:2407.07972*, 2024.

[74] Andrei Semenov, Matteo Pagliardini, and Martin Jaggi. Benchmarking optimizers for large language model pretraining. *arXiv preprint arXiv:2509.01440*, 2025.

[75] Kaiyue Wen, David Hall, Tengyu Ma, and Percy Liang. Fantastic pretraining optimizers and where to find them. *arXiv preprint arXiv:2509.02046*, 2025.

[76] Alexandra Volkova, Mher Safaryan, Christoph H Lampert, and Dan Alistarh. Towards robust scaling laws for optimizers. *arXiv preprint arXiv:2602.07712*, 2026.

[77] Danny Hernandez, Jared Kaplan, Tom Henighan, and Sam McCandlish. Scaling laws for transfer. *arXiv preprint arXiv:2102.01293*, 2021.

[78] Jiasheng Ye, Peiju Liu, Tianxiang Sun, Jun Zhan, Yunhua Zhou, and Xipeng Qiu. Data mixing laws: Optimizing data mixtures by predicting language modeling performance. *arXiv preprint arXiv:2403.16952*, 2024.

[79] David Mizrahi, Anders Boesen Lindbo Larsen, Jesse Allardice, Suzie Petryk, Yuri Gorokhov, Jeffrey Li, Alex Fang, Josh Gardner, Tom Gunter, and Afshin Dehghan. Language models improve when pretraining data matches target tasks. *arXiv preprint arXiv:2507.12466*, 2025.

[80] Mustafa Shukor, Louis Bethune, Dan Busbridge, David Grangier, Enrico Fini, Alaaeldin El-Nouby, and Pierre Ablin. Scaling laws for optimal data mixtures. *arXiv preprint arXiv:2507.09404*, 2025.

[81] Anirudh Subramanyam, Yuxin Chen, and Robert L Grossman. Scaling laws revisited: modeling the role of data quality in language model pretraining. *arXiv preprint arXiv:2510.03313*, 2025.

[82] Prasanna Mayilvahanan, Thaddäus Wiedemer, Sayak Mallick, Matthias Bethge, and Wieland Brendel. Llms on the line: Data determines loss-to-loss scaling laws. *arXiv preprint arXiv:2502.12120*, 2025.

[83] Feiyang Kang, Newsha Ardalani, Michael Kuchnik, Youssef Emad, Mostafa Elhoushi, Shubhabrata Sengupta, Shang-Wen Li, Ramya Raghavendra, Ruoxi Jia, and Carole-Jean Wu. Demystifying synthetic data in llm pre-training: A systematic study of scaling laws, benefits, and pitfalls. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 10750–10769, 2025.

[84] Yang Zhao, Li Du, Xiao Ding, Kai Xiong, Zhouhao Sun, Shi Jun, Ting Liu, and Bing Qin. Deciphering the impact of pretraining data on large language models through machine unlearning. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 9386–9406, 2024.

[85] Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy S Liang, Quoc V Le, Tengyu Ma, and Adams Wei Yu. Doremi: Optimizing data mixtures speeds up language model pretraining. *Advances in Neural Information Processing Systems*, 36: 69798–69818, 2023.

[86] Qian Liu, Xiaosen Zheng, Niklas Muennighoff, Guangtao Zeng, Longxu Dou, Tianyu Pang, Jing Jiang, and Min Lin. Regmix: Data mixture as regression for language model pre-training. *arXiv preprint arXiv:2407.01492*, 2024.

[87] Wanyun Xie, Francesco Tonin, and Volkan Cevher. Chameleon: A flexible data-mixing framework for language model pretraining and finetuning. *arXiv preprint arXiv:2505.24844*, 2025.

[88] Margaret Li, Sneha Kudugunta, and Luke Zettlemoyer. (mis) fitting: A survey of scaling laws. *arXiv preprint arXiv:2502.18969*, 2025.

[89] Simon Du and Jason Lee. On the power of over-parametrization in neural networks with quadratic activation. In *International conference on machine learning*, pages 1329–1338. PMLR, 2018.

[90] Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. Learning and generalization in overparameterized neural networks, going beyond two layers. *Advances in neural information processing systems*, 32, 2019.

[91] Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. *Advances in neural information processing systems*, 32, 2019.

[92] Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *Advances in neural information processing systems*, 32, 2019.

[93] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Russ R Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. *Advances in neural information processing systems*, 32, 2019.

[94] Roman Novak, Lechao Xiao, Jiri Hron, Jaehoon Lee, Alexander A Alemi, Jascha Sohl-Dickstein, and Samuel S Schoenholz. Neural tangents: Fast and easy infinite neural networks in python. *arXiv preprint arXiv:1912.02803*, 2019.

[95] Chaoyue Liu, Libin Zhu, and Misha Belkin. On the linearity of large non-linear models: when and why the tangent kernel is constant. *Advances in Neural Information Processing Systems*, 33:15954–15964, 2020.

[96] Sanjeev Arora, Simon Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *International conference on machine learning*, pages 322–332. PMLR, 2019.

[97] Shokichi Takakura and Taiji Suzuki. Mean-field analysis on two-layer neural networks from a kernel perspective. *arXiv preprint arXiv:2403.14917*, 2024.

[98] Justin Sirignano and Konstantinos Spiliopoulos. Mean field analysis of deep neural networks. *Mathematics of Operations Research*, 47(1):120–152, 2022.

[99] Juno Kim and Taiji Suzuki. Transformers learn nonlinear features in context: Nonconvex mean-field dynamics on the attention landscape. *arXiv preprint arXiv:2402.01258*, 2024.

[100] Philippe Rigollet. The mean-field dynamics of transformers. *arXiv preprint arXiv:2512.01868*, 2025.

[101] Ángel Poc-López and Miguel Aguilera. Dynamical mean-field theory of self-attention neural networks. *arXiv preprint arXiv:2406.07247*, 2024.

[102] Greg Yang and Edward J Hu. Tensor programs iv: Feature learning in infinite-width neural networks. In *International Conference on Machine Learning*, pages 11727–11737. PMLR, 2021.

[103] Ge Yang, Edward Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tuning large neural networks via zero-shot hyperparameter transfer. *Advances in Neural Information Processing Systems*, 34: 17084–17097, 2021.

[104] Lucas Lingle. An empirical study of μ p learning rate transfer. *arXiv preprint arXiv:2404.05728*, 2024.

[105] Charlie Blake, Constantin Eichenberg, Josef Dean, Lukas Balles, Luke Y Prince, Björn Deiseroth, Andres Felipe Cruz-Salinas, Carlo Luschi, Samuel Weinbach, and Douglas Orr. u-μ p: The unit-scaled maximal update parametrization. *arXiv preprint arXiv:2407.17465*, 2024.

[106] Soufiane Hayou. A proof of learning rate transfer under μ p. *arXiv preprint arXiv:2511.01734*, 2025.

[107] Chenyu Zheng, Rongzhen Wang, Xinyu Zhang, and Chongxuan Li. Spectral condition for μ p under width-depth scaling. *arXiv preprint arXiv:2603.00541*, 2026.

A Related Work

Optimizers of LLMs play a central role in pretraining and can substantially affect the resulting model capabilities, which are further shaped by post-training [33, 34]. Over the past decade, a large body of work has developed and refined optimizers to improve the performance, stability, and efficiency of large-scale training. Starting from (Stochastic Gradient Descent) SGD [35], adaptive coordinate-wise optimizers were introduced to better adapt updates to the optimization landscape, including AdaGrad [36], RMSProp [37], AdaDelta [38], and Adam [8]. Owing to its strong empirical performance in large-scale training, Adam became the default optimizer for many LLM pretraining pipelines. Subsequent works further improved Adam through techniques such as Nesterov momentum [39], modified momentum schemes [40], bias correction [41], adaptive step sizes [42], and decoupled weight decay [43]. To better exploit the matrix structure of LLM parameters, matrix-based optimizers were later proposed, including Muon [2], Shampoo [12], Soap [1], Dion [44], and Scion [11]. Following the introduction of Muon, several recent works further improved its performance through alternative normalization strategies and adaptive designs [45, 46, 47, 48, 49, 50, 51, 52, 53, 54]. Relatedly, another line of work seeks to optimize parameters by approximating Gauss–Newton updates [55, 56]. Orthogonal to these directions, some works focus primarily on reducing the memory and communication cost of training, including Adafactor [57], Adam-mini [10], GaLore [58], Apollo [59], and MuonBP [60]. Comprehensive surveys of LLM optimizers are provided by He et al. [61], Wan et al. [62], Abdulkadirov et al. [63], Zhou et al. [64], Luo et al. [65], Tian et al. [66], Xiao et al. [67]. In contrast to these works, which propose new optimizers, we study the shared characteristics of existing optimizers. Specifically, we show that many commonly used optimizers exhibit the same relative generalization behavior.

Influential Factors in LLM Pretraining include model architecture, datasets, and optimizers. Following the seminal works on scaling laws [68, 69], a growing body of work has studied how these three factors separately and jointly affect pretraining performance. From the perspective of model design, Hoffmann et al. [70], Tao et al. [71] study the optimal allocation of compute and the role of vocabulary size as model size scales. Bian et al. [72] further studies the trade-off between throughput and performance across different architectural configurations. From the perspective of optimization, Zhao et al. [73], Semenov et al. [74], Wen et al. [75] examine how different optimizers affect training efficiency and model performance across different data and model-size regimes. In addition, Volkova et al. [76] proposes a robust method for fitting scaling laws under different optimizers. A large body of work also studies how datasets influence pretraining performance. Hernandez et al. [77], Ye et al. [78], Mizrahi et al. [79], Liu et al. [34], Shukor et al. [80], Subramanyam et al. [81], Mayilvahanan et al. [82] investigate how pretraining data mixtures affect target-task and downstream performance through the lens of scaling laws. In contrast, Kang et al. [83] and Zhao et al. [84] study properties of data mixtures using human-controlled synthetic data and machine unlearning, respectively. Beyond these mechanism-oriented studies, another line of work investigates how to construct near-optimal data mixtures efficiently for large-scale pretraining, including DoReMi [85], RegMix [86], and Chameleon [87]. See Li et al. [88] for a survey of scaling laws across different pretraining factors and Luo et al. [65] for a survey of data-centric LLM pretraining. Different from these works, we study the distinct roles of optimizers and data in shaping relative generalization, which holds almost surely with respect to the data stream.

Pretraining Dynamics of LLMs have received considerable attention over the past decade, and these studies have shed light on the behavior of different optimizers throughout training. Systematic theoretical studies of pretraining dynamics typically focus on over-parameterized regimes, in which the number of trainable parameters is much larger than the number of training samples [89, 90]. Among these settings, two regimes have been studied most extensively: the NTK regime and the MF regime. The NTK regime is also known as the lazy-training regime [91], in which the kernel induced by the network remains effectively fixed throughout training [26, 92]. Under suitable scalings of model width, initialization, and learning rate, a line of work establishes convergence rates to global optima in the NTK regime [93, 94, 28, 95]. In addition, Arora et al. [96] explicitly derives the kernel on the training data and shows how convergence depends on its properties. See Golikov et al. [27] for a survey of the NTK regime. To relax the restrictive assumption that the kernel remains fixed during training, the MF regime enables theoretical analysis of an evolving kernel via

mean-field approximation [30, 32, 97]. Rather than tracking each neuron individually, the mean-field approximation represents neurons collectively through a distribution. This framework was developed rigorously by Mei et al. [31] and Sirignano and Spiliopoulos [32] for two-layer networks and later extended to multi-layer networks [98]. A line of work further extends mean-field analysis from fully connected networks to transformers [99, 100, 101]. Motivated by mean-field analysis, μP was proposed to preserve nontrivial parameter updates in the infinite-width limit [102, 103]. Its analysis supports hyperparameter transfer across models of different sizes [104, 105, 106, 107]. Different from these works, the relative-generalization invariance behavior of optimizers studied by our work is not explained by either the NTK or the MF perspective, as shown in Figure 9.

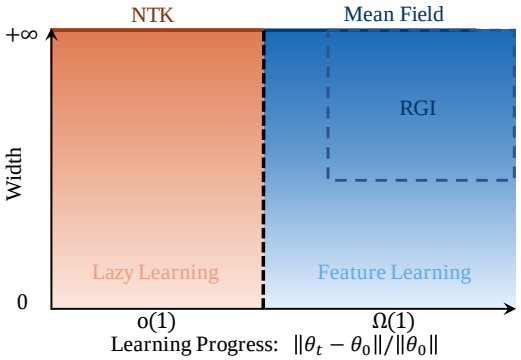


Figure 9: RGI cannot be explained by NTK and MF alone.

B Experimental Details

B.1 Concordance Correlation Coefficient

We present the detailed definition of CCC among centered losses $L(\theta_t^{\mathfrak{T}}, D_{\mathrm{val},k})$ and $L(\theta_t^{\mathfrak{T}'}, D_{\mathrm{val},k})$ as follows. The centered losses are defined as

$$\bar{L}\left(\theta_t^{\mathfrak{T}}, D_{\mathrm{val},k}\right) = L\left(\theta_t^{\mathfrak{T}}, D_{\mathrm{val},k}\right) - K^{-1}\sum_{k'=1}^{K} L\left(\theta_t^{\mathfrak{T}}, D_{\mathrm{val},k'}\right).$$

Then we have that

$$r_{\mathrm{CCC}}(\mathfrak{T},\mathfrak{T}') = \mathbb{E}\left[\frac{2\,\mathrm{Cov}_K\left(\bar{L}\left(\theta_t^{\mathfrak{T}}, D_{\mathrm{val},k}\right), \bar{L}\left(\theta_t^{\mathfrak{T}'}, D_{\mathrm{val},k}\right)\right)}{\mathrm{Var}_K\left(\bar{L}\left(\theta_t^{\mathfrak{T}}, D_{\mathrm{val},k}\right)\right) + \mathrm{Var}_K\left(\bar{L}\left(\theta_t^{\mathfrak{T}'}, D_{\mathrm{val},k}\right)\right)}\right],$$

where the expectation is taken with respect to the joint distribution of θ_0 and $(D_l)_{l=1}^t$, and Cov_K and Var_K are covariance and variance with respect to k. In other words, for any functions $f(D_{\mathrm{val},k})$ and $g(D_{\mathrm{val},k})$, we have

$$\mathrm{Var}_K\left(f(D_{\mathrm{val},k})\right) = \frac{1}{K}\sum_{k=1}^{K}\left(f(D_{\mathrm{val},k}) - \frac{1}{K}\sum_{k'=1}^{K} f(D_{\mathrm{val},k'})\right)^2$$

$$\mathrm{Cov}_K\left(f(D_{\mathrm{val},k}), g(D_{\mathrm{val},k})\right) = \frac{1}{K}\sum_{k=1}^{K} f(D_{\mathrm{val},k})g(D_{\mathrm{val},k}) - \frac{1}{K^2}\sum_{k'=1}^{K} f(D_{\mathrm{val},k'})\cdot\sum_{k'=1}^{K} g(D_{\mathrm{val},k'}).$$

B.2 Definition of r_{diff}

$$s(\mathfrak{T},\mathfrak{T}') = \sum_{k,k'\in[K]}\left(\Delta(\mathfrak{T}, D_{\mathrm{val},k}, D_{\mathrm{val},k'}) - \Delta(\mathfrak{T}', D_{\mathrm{val},k}, D_{\mathrm{val},k'})\right)^2$$

$$r_{\mathrm{diff}}(\mathfrak{T},\mathfrak{T}') = \frac{1}{2}\cdot\mathbb{E}\left[\frac{s(\mathfrak{T},\mathfrak{T}')}{\sum_{k,k'\in[K]}\Delta(\mathfrak{T}, D_{\mathrm{val},k}, D_{\mathrm{val},k'})^2} + \frac{s(\mathfrak{T},\mathfrak{T}')}{\sum_{k,k'\in[K]}\Delta(\mathfrak{T}', D_{\mathrm{val},k}, D_{\mathrm{val},k'})^2}\right],$$

where the expectation is taken with respect to the joint distribution of θ_0 and $D_{1:t}$.

B.3 Experimental Setups

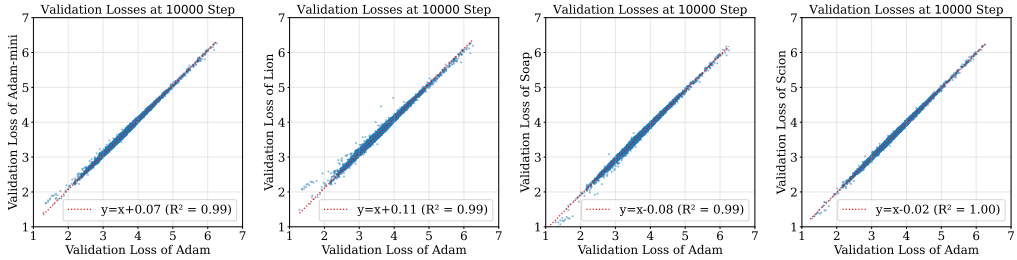
In all experiments, we train LLMs using the GPT-2 tokenizer with sequence length 1024 and batch size 524,288. For each optimizer, we disable weight decay and search the learning rate over

$\{1, 2, 5\} \times \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$. For Adam, Adam-mini, and Lion, we set $\beta_1 = 0.9$ and $\beta_2 = 0.95$; for SOAP, we set $\beta_1 = \beta_2 = 0.95$; for Muon and Scion, we use momentum values 0.95 and 0.9, respectively. We consider two GPT-style model sizes. The 124M model has 12 layers, 12 attention heads, and hidden dimension 768, while the 0.7B model has 36 layers, 20 attention heads, and hidden dimension 1280. For evaluations, we partition $\mathcal{D}_{\text{val}}$ into validation batches $(D_{\text{val},k})_{k=1}^{K}$ and evaluate two batch granularities: $|D_{\text{val},k}| = 1{,}024$ and $|D_{\text{val},k}| = 1$. In the main text, we report results with $|D_{\text{val},k}| = 1{,}024$ and $K = 20{,}480$, corresponding to approximately 21M validation tokens. The single-token results, i.e., $|D_{\text{val},k}| = 1$, are computed on 5M tokens due to computational cost and deferred to Appendix C; they yield conclusions consistent with the $|D_{\text{val},k}| = 1{,}024$ setting.

Compute Resources. All experiments are conducted on NVIDIA A40 GPUs. Each 124M model run uses 4 A40 GPUs and takes approximately 10 hours, while each 0.7B run uses 4 A40 GPUs and takes approximately 50 hours. Each 4.5B run uses 8 A40 GPUs and takes approximately three days.

C Additional Experimental Results

C.1 Additional Results of Validation Losses Between Optimizers



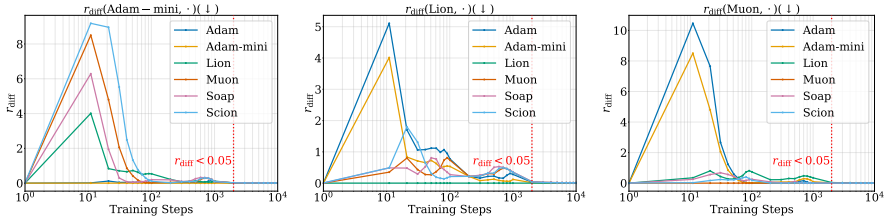
(a) Validation Losses of Adam and Adam-mini (b) Validation Losses of Adam and Lion (c) Validation Losses of Adam and Soap (d) Validation Losses of Adam and Scion

Figure 10: Validation-loss linearity across Adam and other optimizers. Panels (a)–(d) plot the validation-loss relationship between Adam and Adam-mini, Lion, Soap, and Scion, respectively, at the 10,000-step checkpoint. The near-perfect fits, with $R^2 \geq 0.99$, support approximate RGI across optimizers.

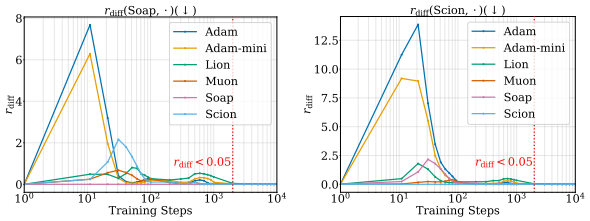
These figures show that the validation losses of models trained by different optimizers are highly linearly correlated at step 10,000. This supports the emergence of approximate RGI among optimizers.

C.2 Additional Results of r_{diff} and r_{CCC} with Other Optimizers as Baselines

In the following, we report r_{diff} and r_{CCC} using optimizers other than Adam as the baseline, while keeping the initial parameters and training data stream identical across all optimizers. The results show that changing the baseline optimizer does not qualitatively affect approximate RGI. This is intuitive: if exact RGI holds between θ and θ', and also between θ' and θ'', then it also holds between θ and θ''. In the approximate case, the same transitivity holds up to accumulated error.

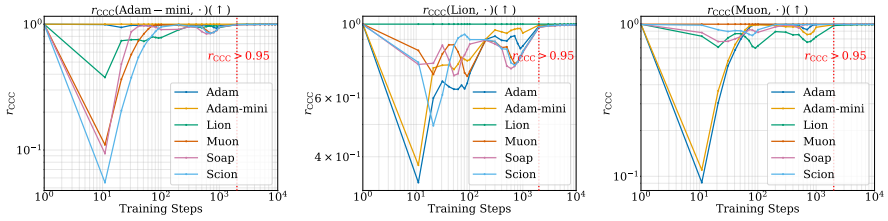


(a) Values of r_{diff} with Adam-mini as the baseline.
(b) Values of r_{diff} with Lion as the baseline.
(c) Values of r_{diff} with Muon as the baseline.

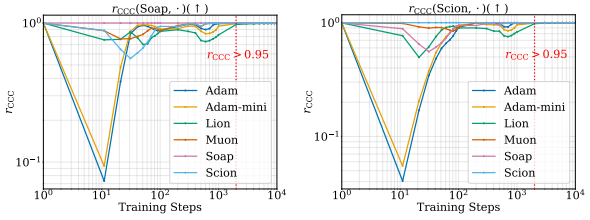


(d) Values of r_{diff} with Soap as the baseline.
(e) Values of r_{diff} with Scion as the baseline.

Figure 11: Baseline-optimizer ablation for r_{diff}. Panels (a)–(e) plot r_{diff} when the baseline optimizer is Adam-mini, Lion, Muon, Soap, and Scion, respectively. Approximate RGI is insensitive to which optimizer is used as the evaluation baseline.



(a) Values of r_{CCC} with Adam-mini as the baseline.
(b) Values of r_{CCC} with Lion as the baseline.
(c) Values of r_{CCC} with Muon as the baseline.

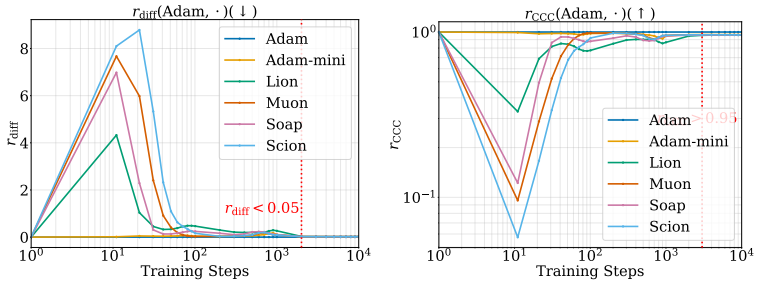


(d) Values of r_{CCC} with Soap as the baseline.
(e) Values of r_{CCC} with Scion as the baseline.

Figure 12: Baseline-optimizer ablation for r_{CCC}. Panels (a)–(e) plot r_{CCC} when the baseline optimizer is Adam-mini, Lion, Muon, Soap, and Scion, respectively. Approximate RGI is insensitive to which optimizer is used as the evaluation baseline.

C.3 Per-Token Results of Relative Generalization Invariance

In the following, we present the results of r_{diff} and r_{CCC} calculated with $|D_{\text{val},k}| = 1$ when the initial parameters and the training data stream are the same for all the optimizers. The results show that RGI among optimizers holds when we verify it in a token-wise manner.

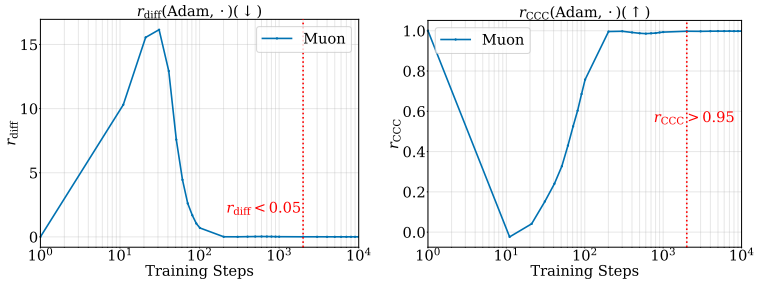


(a) Values of r_{diff} calculated per token. (b) Values of r_{CCC} calculated per token.

Figure 13: Per-token evaluation of RGI. Panels (a) and (b) plot r_{diff} and r_{CCC}, respectively, when each validation batch contains a single token, i.e., $|D_{\text{val},k}| = 1$. The small r_{diff} and large r_{CCC} values show that RGI also holds at token-level granularity.

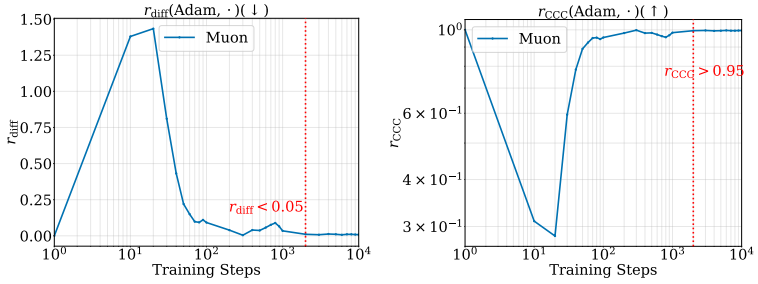
C.4 Results for 0.7B and 4.5B Models

Due to the high computational cost of training 0.7B and 4.5B models, we focus on Adam and Muon at this scale. We provide results for 0.7B and 4.5B models when Adam and Muon share the same initial parameters and training data stream. These results mirror those in Section 4.1, showing that RGI among optimizers also holds at larger scales.



(a) Values of r_{diff} for 0.7B models. (b) Values of r_{CCC} for 0.7B models.

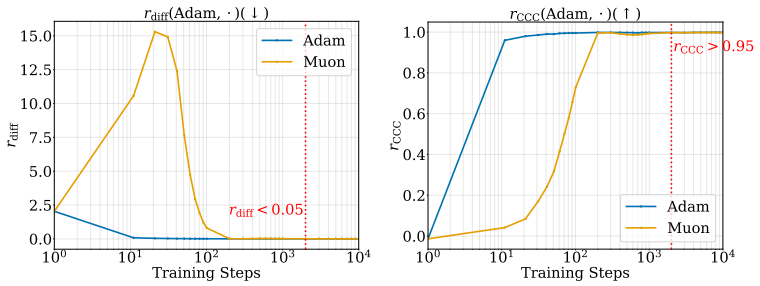
Figure 14: Baseline RGI for 0.7B models. Panels (a) and (b) plot r_{diff} and r_{CCC}, respectively, between Adam and Muon when they share the same initial parameters and training data stream. The results show that approximate RGI persists at the 0.7B scale.



(a) Values of r_{diff} for 4.5B models. (b) Values of r_{CCC} for 4.5B models.

Figure 15: Baseline RGI for 4.5B models. Panels (a) and (b) plot r_{diff} and r_{CCC}, respectively, between Adam and Muon when they share the same initial parameters and training data stream. The results show that approximate RGI persists at the 4.5B scale.

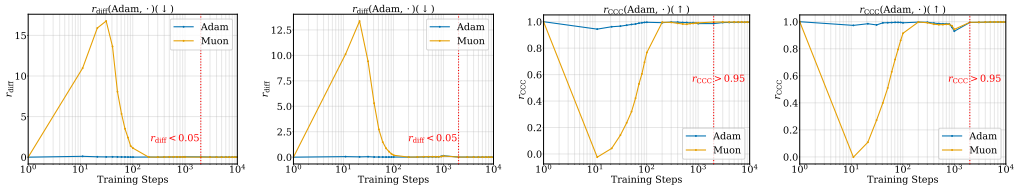
In the following experiments, we ablate how the hyperparameters influence RGI of 0.7B models. All results are compared against Adam under the baseline in Figure 14. Therefore, we also measure RGI between Adam runs with different settings.



(a) Values of r_{diff} for 0.7B models with different initial parameters.

(b) Values of r_{CCC} for 0.7B with different initial parameters.

Figure 16: Initialization ablation for 0.7B models. Panels (a) and (b) plot r_{diff} and r_{CCC}, respectively, for Adam and Muon runs started from different initial parameters. Approximate RGI still emerges at the 0.7B scale, showing that initialization differences do not prevent the phenomenon.



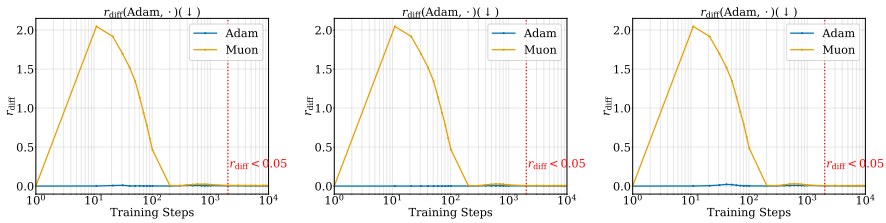
(a) Values of r_{diff} for 0.7B models with $0.25\eta^{\mathsf{A}}$.

(b) Values of r_{diff} for 0.7B models with $1.25\eta^{\mathsf{A}}$.

(c) Values of r_{CCC} for 0.7B with $0.25\eta^{\mathsf{A}}$.

(d) Values of r_{CCC} for 0.7B with $1.25\eta^{\mathsf{A}}$.

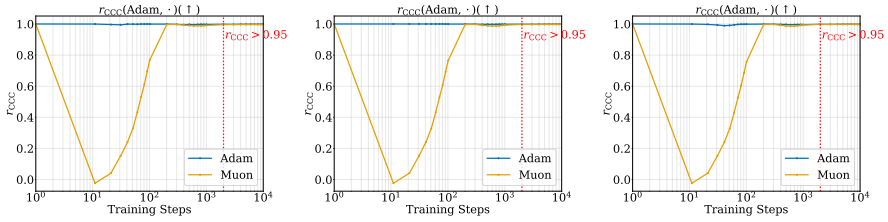
Figure 17: Learning-rate ablation for 0.7B models. Panels (a) and (b) plot r_{diff} for $0.25\eta^{\mathsf{A}}$ and $1.25\eta^{\mathsf{A}}$, while Panels (c) and (d) plot r_{CCC} for the same two learning-rate multipliers. Approximate RGI remains robust to these learning-rate changes at the 0.7B scale.



(a) Values of r_{diff} for 0.7B models with $\gamma = 0.85$.

(b) Values of r_{diff} for 0.7B models with $\gamma = 0.9$.

(c) Values of r_{diff} for 0.7B models with $\gamma = 0.95$.

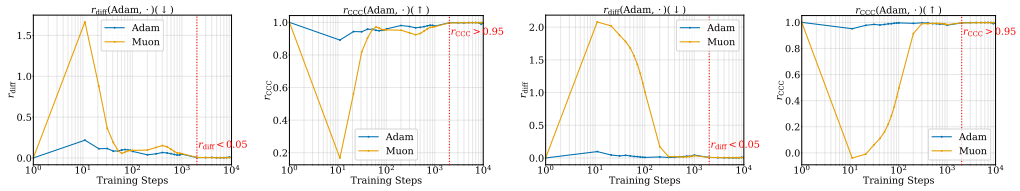


(d) Values of r_{CCC} for 0.7B with $\gamma = 0.85$.

(e) Values of r_{CCC} for 0.7B with $\gamma = 0.9$.

(f) Values of r_{CCC} for 0.7B with $\gamma = 0.95$.

Figure 18: Momentum ablation for 0.7B models. Panels (a)–(c) plot r_{diff} for $\gamma = 0.85$, 0.9, and 0.95, while Panels (d)–(f) plot r_{CCC} for the same momentum values. Approximate RGI remains robust to the tested momentum settings at the 0.7B scale.



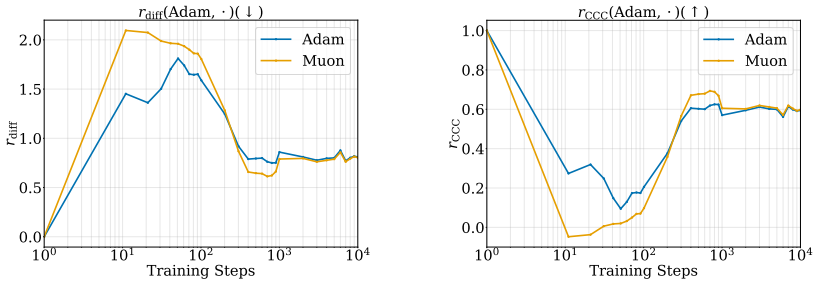
(a) Values of r_{diff} for 0.7B models with 100 warmup steps.

(b) Values of r_{CCC} for 0.7B models with 100 warmup steps.

(c) Values of r_{diff} for 0.7B models with 1400 warmup steps.

(d) Values of r_{CCC} for 0.7B models with 1400 warmup steps.

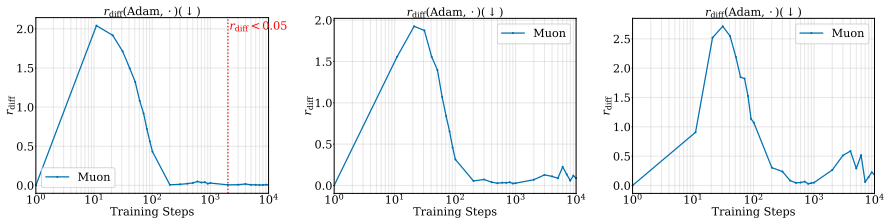
Figure 19: Warmup-step ablation for 0.7B models. Panels (a) and (c) plot r_{diff} for 100 and 1,400 warmup steps, while Panels (b) and (d) plot r_{CCC} for the same warmup settings. Approximate RGI remains robust to the tested scheduler warmup lengths at the 0.7B scale.



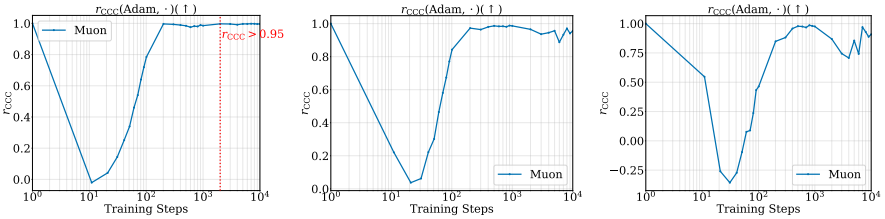
(a) Values of r_{diff} for 0.7B models when Adam is trained on FineWeb and Muon is trained on CodeParrot.

(b) Values of r_{CCC} for 0.7B models when Adam is trained on FineWeb and Muon is trained on CodeParrot.

Figure 20: Training data-stream ablation for 0.7B models. Panels (a) and (b) plot r_{diff} and r_{CCC}, respectively, when Adam is trained on FineWeb and Muon is trained on CodeParrot. The weak agreement shows that changing the training data stream strongly disrupts RGI at the 0.7B scale.



(a) Values of r_{diff} for 0.7B models with D_{val} from C4. (b) Values of r_{diff} for 0.7B models with D_{val} from ArXiv. (c) Values of r_{diff} for 0.7B models with D_{val} from CodeParrot.

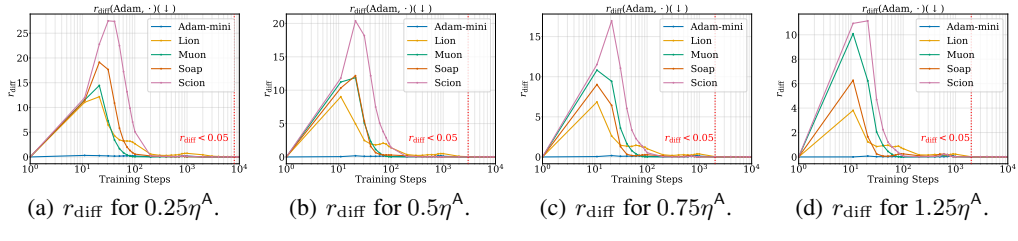


(d) Values of r_{CCC} for 0.7B models with D_{val} from C4. (e) Values of r_{CCC} for 0.7B models with D_{val} from ArXiv. (f) Values of r_{CCC} for 0.7B models with D_{val} from CodeParrot.

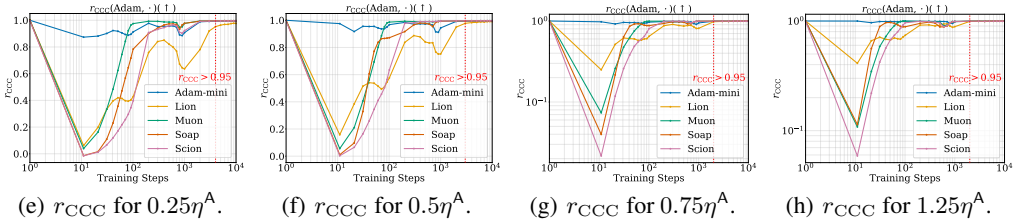
Figure 21: Validation-distribution ablation for 0.7B models. Panels (a)–(c) plot r_{diff} evaluated on C4, ArXiv, and CodeParrot, while Panels (d)–(f) plot r_{CCC} on the same validation datasets. RGI remains clearest on C4 and weakens on the more distributionally distinct ArXiv and CodeParrot validation sets.

C.5 Additional Results for Section 4

C.5.1 Additional Results for Section 4.1

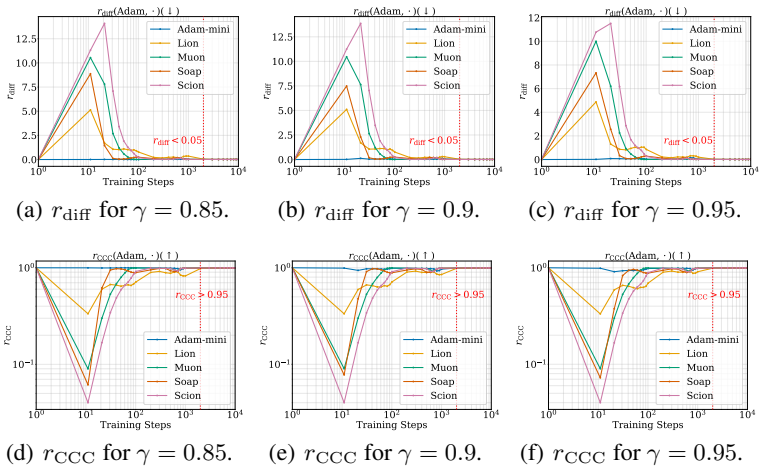


(a) r_{diff} for $0.25\eta^{\text{A}}$. (b) r_{diff} for $0.5\eta^{\text{A}}$. (c) r_{diff} for $0.75\eta^{\text{A}}$. (d) r_{diff} for $1.25\eta^{\text{A}}$.



(e) r_{CCC} for $0.25\eta^{\text{A}}$. (f) r_{CCC} for $0.5\eta^{\text{A}}$. (g) r_{CCC} for $0.75\eta^{\text{A}}$. (h) r_{CCC} for $1.25\eta^{\text{A}}$.

Figure 22: Learning-rate ablations for r_{diff} and r_{CCC}. Panels (a)–(d) plot r_{diff} for learning-rate multipliers $0.25\eta^{\text{A}}$, $0.5\eta^{\text{A}}$, $0.75\eta^{\text{A}}$, and $1.25\eta^{\text{A}}$, respectively. Panels (e)–(h) plot r_{CCC} for them. The results show that approximate RGI holds robustly across the tested learning rates.

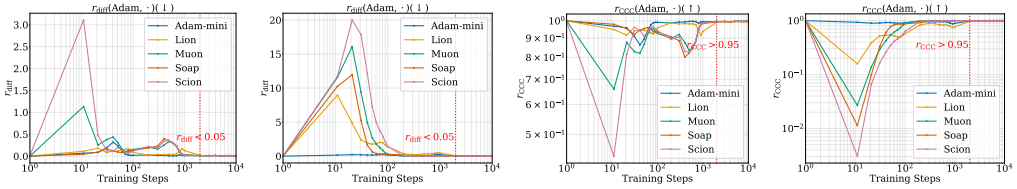


(a) r_{diff} for $\gamma = 0.85$. (b) r_{diff} for $\gamma = 0.9$. (c) r_{diff} for $\gamma = 0.95$.

(d) r_{CCC} for $\gamma = 0.85$. (e) r_{CCC} for $\gamma = 0.9$. (f) r_{CCC} for $\gamma = 0.95$.

Figure 23: Momentum ablations for r_{diff} and r_{CCC}. Panels (a)–(c) plot r_{diff} for momentumns $\gamma = 0.85.0.9, 0.95$, respectively. Panels (d)–(f) plot r_{CCC} for them. The results show that approximate RGI holds robustly across the tested momentum values.

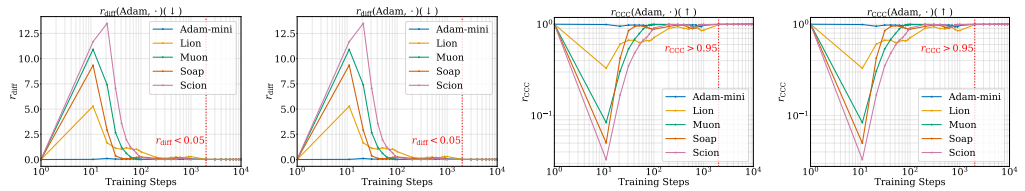
To evaluate the effect of the learning-rate scheduler, we ablate the number of warmup steps in the Warmup-Stable-Decay scheduler. The experiments in Section 4.1 use 700 warmup steps. We additionally evaluate all optimizers with 100 and 1,400 warmup steps. Figure 24 reports the corresponding values of r_{diff} and r_{CCC}. Across all tested warmup lengths, approximate RGI remains consistently observed.



(a) r_{diff} with 100 warmup steps. (b) r_{diff} with $1,400$ warmup steps. (c) r_{CCC} with 100 warmup steps. (d) r_{CCC} with $1,400$ warmup steps.

Figure 24: Warmup-step ablation. Panels (a) and (b) plot r_{diff} for 100 and 1,400 warmup steps, while Panels (c) and (d) plot r_{CCC} for the same warmup settings. Approximate RGI holds robustly across the tested scheduler warmup lengths.

The experiments in Section 4.1 use weight decay as 0. To evaluate the effect of the weight decay, we additionally evaluate all optimizers with weight decay 0.05 and 0.1. Figure 25 reports the corresponding values of r_{diff} and r_{CCC}. Across all tested weight-decay values, approximate RGI remains consistently observed.



(a) r_{diff} with weight decay value 0.05. (b) r_{diff} with weight decay value 0.1. (c) r_{CCC} with weight decay value 0.05. (d) r_{CCC} with weight decay value 0.1.

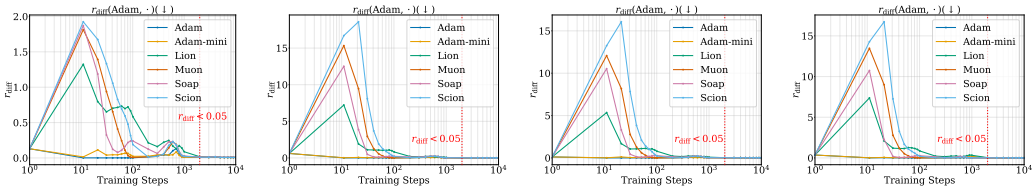
Figure 25: Weight-decay ablation. Panels (a) and (b) plot r_{diff} for 0.05 and 0.1 weight decay, while Panels (c) and (d) plot r_{CCC} for the same weight decay settings. Approximate RGI holds robustly across the tested weight decay values.

C.5.2 Additional Results for Section 4.2



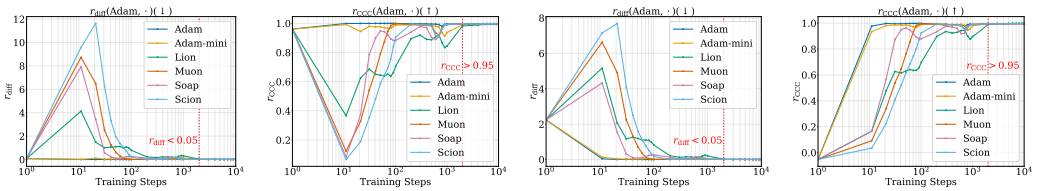
(a) r_{diff} under different initial parameters. (b) r_{diff} for hidden state dimensions 768 and 384. (c) r_{diff} for hidden state dimensions 768 and 64.

Figure 26: RGI across optimizers, initializations, and model sizes. Panel (a) plots r_{diff} across optimizers with different initializations. Panels (b) and (c) plot r_{diff} across models with hidden-state dimensions 768, 384, and 64. RGI continues to hold across different initializations and moderate changes in model size.



(a) Values of r_{diff} between FA models with RoPE and NoPE. (b) Values of r_{diff} between FA models with RoPE and ALiBi. (c) Values of r_{diff} between FA and SWA models. (d) Values of r_{diff} between models with GELU and SReLU.

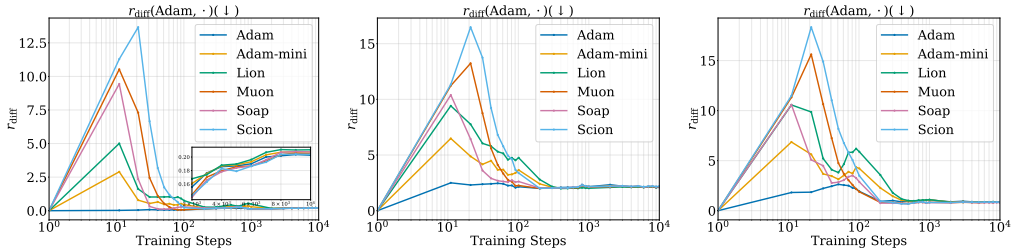
Figure 27: RGI across optimizers and models with different attention and FFN designs. Panels (a)–(c) plot r_{diff} between the RoPE FA baseline trained with Adam and NoPE FA, ALiBi FA, and RoPE SWA models, respectively, each trained with all optimizers. Panel (d) plots r_{diff} between models using GELU and SReLU activations. RGI persists across different attention and FFN designs.



(a) Values of r_{diff} between models with GELU and SiLU.
(b) Values of r_{CCC} between models with GELU and SiLU.
(c) Values of r_{diff} between models with gated and non-gated FFN.
(d) Values of r_{CCC} between models with gated and non-gated FFN.

Figure 28: RGI across optimizers and models with different FFN designs. Panels (a) and (b) plot r_{diff} and r_{CCC}, respectively, between the GELU baseline trained with Adam and SiLU models trained with other optimizers. Panels (c) and (d) plot r_{diff} and r_{CCC}, respectively, between the non-gated FFN baseline trained with Adam and gated FFN models trained with other optimizers. RGI persists across different FFN designs.

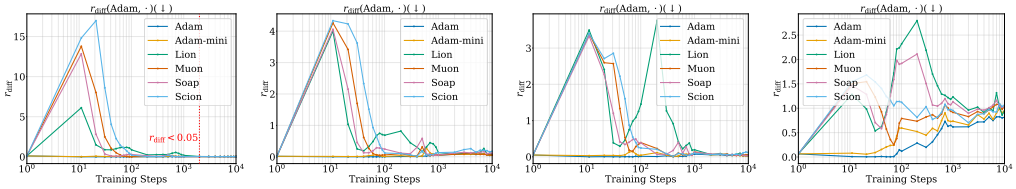
C.5.3 Additional Results for Section 4.3



(a) Values of r_{diff} with D_l' from C4.
(b) Values of r_{diff} with D_l' from ArXiv.
(c) Values of r_{diff} with D_l' from CodeParrot.

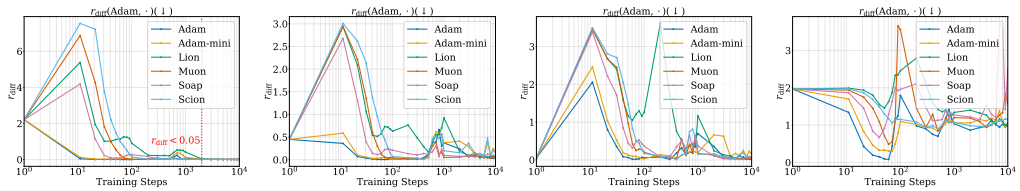
Figure 29: RGI across optimizers trained on different data streams. The panels plot r_{diff} when Adam is trained on FineWeb and the compared optimizer is trained on C4 (Panel (a)), ArXiv (Panel (b)), or CodeParrot (Panel (c)). The results show that RGI is strongly affected by the training data stream: changing the data stream can substantially weaken, or even eliminate, RGI across optimizers.

C.5.4 Additional Results for Section 4.4

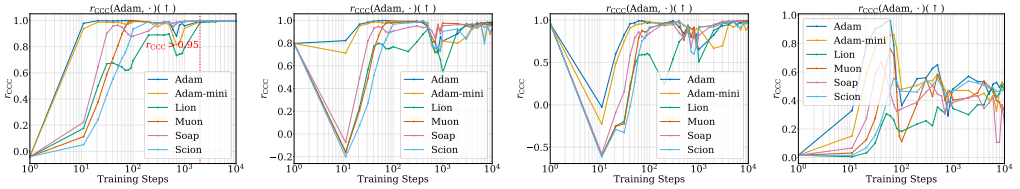


(a) Values of r_{diff} with D_{val} from C4.
(b) Values of r_{diff} with D_{val} from ArXiv.
(c) Values of r_{diff} with D_{val} from CodeParrot.
(d) Values of r_{diff} with D_{val} from Random.

Figure 30: RGI across optimizers and FA/SWA models evaluated on different validation datasets. The panels plot r_{diff} evaluated on C4, ArXiv, CodeParrot, and Random. These results show that RGI can transfer to related validation distributions, but becomes weaker on more distributionally distinct validation datasets.



(a) Values of r_{diff} with D_{val} from C4. (b) Values of r_{diff} with D_{val} from ArXiv. (c) Values of r_{diff} with D_{val} from CodeParrot. (d) Values of r_{diff} with D_{val} from Random.

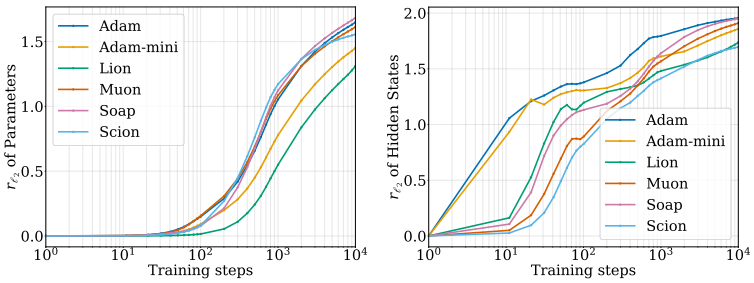


(e) Values of r_{CCC} with D_{val} from C4. (f) Values of r_{CCC} with D_{val} from ArXiv. (g) Values of r_{CCC} with D_{val} from CodeParrot. (h) Values of r_{CCC} with D_{val} from Random.

Figure 31: RGI across optimizers and gated/non-gated models evaluated on different validation datasets. The panels plot r_{diff} and r_{CCC} evaluated on C4, ArXiv, CodeParrot, and Random. These results show that RGI can transfer to related validation distributions, but becomes weaker on more distributionally distinct validation datasets.

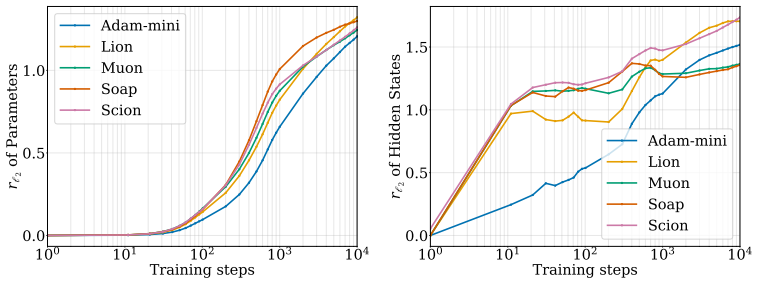
C.6 Additional Results for Section 5

C.6.1 Additional Results for Section 5.1



(a) Values of r_{ℓ_2} between parameters. (b) Values of r_{ℓ_2} between hidden states.

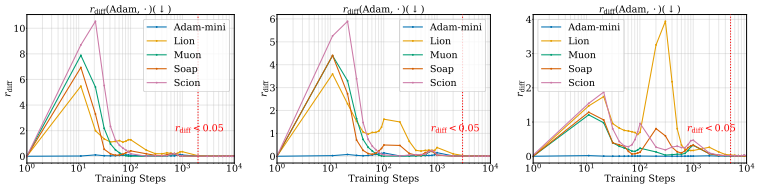
Figure 32: Movement from initialization in parameter and hidden-state space. Panels (a) and (b) plot $r_{\ell_2}(\downarrow)$ between initialization and later checkpoints for parameters and hidden states, respectively. The large relative distances show substantial movement during training, providing evidence against an NTK-style lazy-training explanation.



(a) Values of r_{ℓ_2} between parameters. (b) Values of r_{ℓ_2} between hidden states.

Figure 33: Cross-optimizer distances in parameter and hidden-state space. Panels (a) and (b) plot $r_{\ell_2}(\downarrow)$ between Adam and the other optimizers at matched training steps for parameters and hidden states, respectively. The large relative distances show that different optimizers need not share similar parameters or hidden states even when their validation-loss differences exhibit RGI.

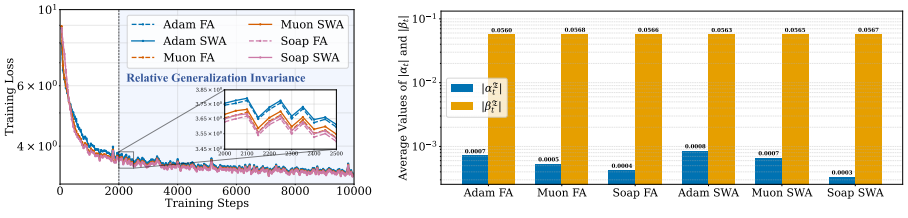
C.6.2 Additional Results for Section 5.2



(a) Values of r_{diff} at width 576. (b) Values of r_{diff} at width 384. (c) Values of r_{diff} at width 192.

Figure 34: Model-width ablations for r_{diff}. Panels (a)–(c) plot r_{diff} for model widths 576, 384, and 192, respectively. RGI holds across all tested widths, supporting the conclusion that the phenomenon is not restricted to the standard model width.

D Explaining Parallel Training-Loss Curves



(a) Training losses of optimizers and architectures

(b) Average values of $|\alpha_t^{\mathfrak{T}}|$ and $|\beta_t^{\mathfrak{T}}|$

Figure 35: Shared Training-Loss Curves. Panel (a) shows that training-loss curves across optimizers and architectures share a similar shape after 2,000 steps. Panel (b) shows that $|\beta_t^{\mathfrak{T}}|$ is much larger than $|\alpha_t^{\mathfrak{T}}|$, explaining Panel (a) with RGI.

Shared Training-Loss Shapes across Optimizers and Architectures Arise from RGI. Training LLMs with different optimizers and architectures on the same data stream often yields training-loss curves with similar shapes. As shown in Figure 35(a), after 2,000 steps, all optimizers exhibit nearly the same training-loss profile. We show that this phenomenon can be explained by approximate RGI.

The training loss at step t is $L(\theta^{\mathfrak{T}}_{t-1}, D_t)$, namely the loss of the parameters trained on $D_{1:t-1}$ and evaluated on the current batch D_t. Its change between adjacent steps can be decomposed as

$$L\big(\theta^{\mathfrak{T}}_t, D_{t+1}\big) - L\big(\theta^{\mathfrak{T}}_{t-1}, D_t\big) = \alpha^{\mathfrak{T}}_t + \beta^{\mathfrak{T}}_t,$$

where $\alpha^{\mathfrak{T}}_t = L(\theta^{\mathfrak{T}}_t, D_{t+1}) - L(\theta^{\mathfrak{T}}_{t-1}, D_{t+1})$ captures the loss change induced by one optimizer update, while $\beta^{\mathfrak{T}}_t = L(\theta^{\mathfrak{T}}_{t-1}, D_{t+1}) - L(\theta^{\mathfrak{T}}_{t-1}, D_t)$ captures the loss difference of the same parameters across adjacent batches. Figure 35(b) shows that $|\beta_t|$ is approximately 50 times larger than $|\alpha_t|$, so the local shape of the training-loss curve is dominated by β_t. Since $\theta^{\mathfrak{T}}_{t-1}$ has not been trained on either D_t or D_{t+1}, these batches act as held-out data for $\theta^{\mathfrak{T}}_{t-1}$. Approximate RGI therefore implies that $\beta^{\mathfrak{T}}_t$ is approximately invariant across optimizers and architectures. Consequently, the adjacent-step changes in training loss are nearly the same across optimizers and architectures, explaining their shared training-loss shapes.

E Discussions of Alternative Explanations for RGI

In this section, we test and falsify several hypotheses that could potentially explain RGI. Specifically, we consider two hypotheses:

- **Intrinsic data-difficulty hypothesis.** Suppose the loss decomposes as $L(\theta, d) = H(d) + \varepsilon(\theta, d)$, where $\varepsilon(\theta, d)$ is much smaller than $H(d)$. Then the relative loss structure is dominated by the intrinsic difficulty $H(d)$ of the data and depends only weakly on the learned parameters θ.
- **Different convergence-speed hypothesis.** RGI emerges because different optimizers generate the same learning trajectory but with different speed.

We first consider the intrinsic data-difficulty hypothesis. For a validation example $d = (x_{\text{prefix}}, x_{\text{pred}})$, our work focuses on the token-level loss as $L(\theta, d) = -\log P_\theta(x_{\text{pred}} \mid x_{\text{prefix}})$. Importantly, RGI does not average over x_{pred}; it holds pointwise for each $d \in \mathcal{D}_{\text{val}}$.

Consider any plausible loss decomposition $L(\theta, d) = H(d) + \varepsilon(\theta, d)$ such that $H(d), \varepsilon(\theta, d) \geq 0$ for every d and θ, where $H(d)$ is independent of the model. We have that

$$0 \leq H(d) = \inf_\theta \big[L(\theta, d) - \varepsilon(\theta, d)\big] \leq \inf_\theta L(\theta, d).$$

Therefore, the minimum achievable loss on d provides an upper bound on any model-independent component $H(d)$. To estimate this bound, we directly optimize the LLM parameters θ on each data d and define the achieved loss as $\hat{H}(d) = L(\hat{\theta}_d, d)$, where $\hat{\theta}_d$ denotes the optimized parameters. Since $\inf_\theta L(\theta, d) \leq \hat{H}(d)$, we obtain $H(d) \leq \hat{H}(d)$. We then report the average values of $\hat{H}(d)$ and $L(\theta_t, d)$, as well as their ratio, at $t = 10000$, when $L(\theta_t, d)$ attains its minimum along the training trajectory. Table 1 shows that the model-independent component $H(d)$ is negligible, i.e., about 0.4%, relative to the smallest observed loss $L(\theta, d)$ along the trajectory. Thus, this falsifies the intrinsic data-difficulty hypothesis.

Table 1: Comparison of $H(d)$ and $L(\theta_t, d)$ across optimizers. The results show that $H(d)$ is negligible relative to $L(\theta_t, d)$.

Metric	Adam-mini	Adam	Lion	Muon	Scion	Soap
Average $\hat{H}(d)$	0.001	0.001	0.001	0.001	0.001	0.001
Average $L(\theta_{10000}, d)$	3.500	3.457	3.554	3.409	3.459	3.403
Average $\hat{H}(d)/L(\theta_{10000}, d)$	0.003	0.004	0.003	0.004	0.004	0.004

We then consider the different convergence-speed hypothesis. We falsify this hypothesis from two perspectives.

First, we view this from the representation space. If the convergence-speed explanation were correct, models trained with different optimizers should reach similar parameters when matched at the same validation-loss level.

Table 2: Parameter cosine similarity across optimizers with aligned validation loss.

Adam step	Validation loss	Adam-mini	Lion	Muon	Scion	Shampoo	SOAP
0	11.023	1.000	1.000	1.000	1.000	1.000	1.000
1000	4.261	0.773	0.753	0.641	0.612	0.396	0.559
2000	3.815	0.658	0.604	0.475	0.469	0.186	0.400
3000	3.654	0.580	0.520	0.425	0.411	0.150	0.303
4000	3.588	0.526	0.465	0.377	0.383	0.130	0.257

Table 2 compares models trained with different optimizers after aligning them to the validation loss achieved by Adam at each training step. We report results only for validation-loss levels greater than or equal to 3.55, because some optimizers do not achieve a loss below 3.55 during training. The results show that the similarity between validation-loss-aligned models decreases as training progresses. This finding provides evidence against the convergence-speed explanation.

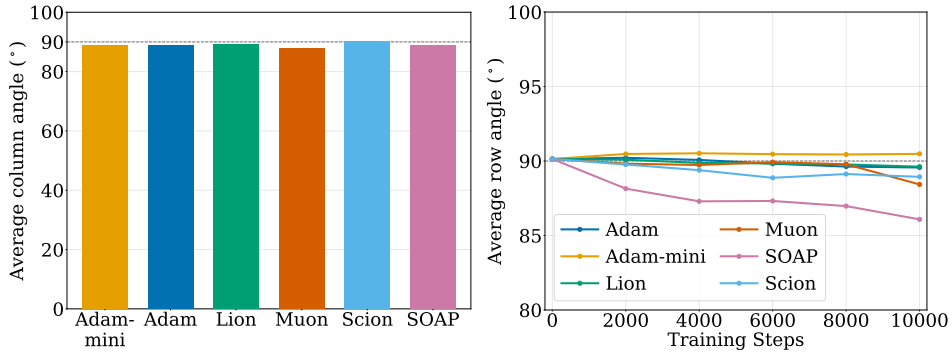
Second, we view the loss difference between optimizers. Table 3 reports the average loss difference between each optimizer A and Adam, defined as $C_{A,\mathrm{Adam}}(t) = L_A(t) - L_{\mathrm{Adam}}(t)$. We observe that $C_{A,\mathrm{Adam}}(t)$ at $t = 2000$, when RGI emerges, differs substantially from its value at convergence, i.e., at $t = 10000$. This finding provides evidence against the convergence-speed explanation.

Table 3: Loss difference relative to Adam, where $C_{A,\mathrm{Adam}}(t) = L_A(t) - L_{\mathrm{Adam}}(t)$.

Step	Adam-mini	Lion	Muon	SOAP	Scion
0	0.000	0.000	0.000	0.000	0.000
1000	0.321	0.738	-0.275	-0.341	-0.226
2000	**0.129**	**0.276**	**-0.138**	**-0.164**	**-0.085**
3000	0.092	0.181	-0.099	-0.117	-0.046
4000	0.073	0.130	-0.089	-0.100	-0.033
5000	0.067	0.107	-0.072	-0.083	-0.018
6000	0.056	0.089	-0.064	-0.075	-0.010
7000	0.044	0.073	-0.062	-0.070	-0.008
8000	0.041	0.078	-0.061	-0.065	-0.007
9000	0.041	0.090	-0.048	-0.053	-0.007
10000	**0.041**	**0.081**	**-0.040**	**-0.041**	**-0.007**

F Empirical Verification of Assumptions 1 and 2

To empirically assess this Assumptions 1, we estimate the target transformation at each layer as the best linear mapping between the hidden states of two adjacent layers after training. We compute $\widehat{H}^l = \arg\min_A \sum_i \|f_i^{l+1} - A f_i^l\|^2$, where f_i^l denotes the normalized hidden state of token i at layer l. We then calculate the angle $\widehat{a}_{i,j}^l$ between columns $\widehat{H}_{:,i}^l$ and $\widehat{H}_{:,j}^l$. The average angle at layer l is $\widehat{a}^l = \sum_{i \neq j} \widehat{a}_{i,j}^l / (r^2 - r)$, and the layer-averaged angle is $\widehat{a} = \sum_{l \in [L]} \widehat{a}^l / L$. An average angle close to 90° will provide empirical evidence that the estimated transformation approximately satisfies Assumption 1. Figure 36(a) verifies this by showing that $\hat{a}$ for various optimizers is close to 90° at the end of training.



(a) Values of $\hat{a}$ for models trained with different optimizers after training.

(b) Values of $\hat{b}$ for different optimizers at different training steps.

Figure 36: Empirical assessment of Assumptions 1 and 2. Panel (a) shows that the columns of the targeted linear mappings are nearly orthogonal, while panel (b) shows that the hidden states of different tokens remain nearly orthogonal throughout training across all optimizers. These results provide empirical support for Assumptions 1 and 2 in practical LLM training.

Assumption 2 essentially requires the inputs to each layer to be isotropic within the subspace they span. Our proof continues to hold when $\sum_{i=1}^{n} x_i x_i^\top / n = Q\,\mathrm{diag}(\sigma_x^2 I_r, 0)Q^\top$, where r is the dimension of the subspace spanned by the inputs and $Q \in \mathbb{R}^{d\times d}$ is an orthogonal matrix. A sufficient condition is that, up to an orthogonal change of coordinates, the input vectors are mutually orthogonal and have equal norms.

Proposition 3 *Suppose $x_1, \ldots, x_n \in \mathbb{R}^d$ are nonzero, mutually orthogonal vectors with the same norm. Then $\sum_{i=1}^{n} x_i x_i^\top / n = Q\,\mathrm{diag}(\sigma_x^2 I_r, 0)Q^\top$ for some $\sigma_x > 0$ and orthogonal matrix $Q \in \mathbb{R}^{d\times d}$, where $r = n$.*

Since the input to each layer is normalized by RMSNorm, we focus on empirically assessing the directional orthogonality of the hidden states. Let f_i^l denote the normalized hidden state of token i at layer l. We measure the pairwise directional similarity by $\widehat{b}_{i,j}^l$, which is the angle between f_i^l and f_j^l, and average $\widehat{b}_{i,j}^l$ over distinct token pairs and layers to obtain the overall orthogonality measure $\widehat{b}$. An average angle close to $90°$ indicates greater directional orthogonality. Figure 36(b) confirms this along the entire training courses of various optimizers.

G Proof of Proposition 1

We first prove that RGI is sufficient for constant generalization gap. From Definition 1, we have that

$$L\big(\theta_t^{\mathfrak{T}}, d\big) - L\big(\theta_t^{\mathfrak{T}'}, d\big) = L\big(\theta_t^{\mathfrak{T}}, d^*\big) - L\big(\theta_t^{\mathfrak{T}'}, d^*\big), \tag{3}$$

where we fix d' as any d^*. Since the optimizers A, A' are deterministic, the right-hand side of Eqn. (3) is a deterministic function of $\theta_0, \theta_0', D_{1:t}, D_{1:t}'$. We write it as $C(\mathfrak{T}, \mathfrak{T}', t)$. Thus, we have that

$$L\big(\theta_t^{\mathfrak{T}}, d\big) - L\big(\theta_t^{\mathfrak{T}'}, d\big) = C(\mathfrak{T}, \mathfrak{T}', t).$$

Then we prove that the constant generalization gap is sufficient for RGI. This is direct, since the following holds

$$L\big(\theta_t^{\mathfrak{T}}, d\big) - L\big(\theta_t^{\mathfrak{T}'}, d\big) = C(\mathfrak{T}, \mathfrak{T}', t) = L\big(\theta_t^{\mathfrak{T}}, d'\big) - L\big(\theta_t^{\mathfrak{T}'}, d'\big).$$

Thus, we conclude the proof of Proposition 1.

H Proof of Theorem 2

We prove Theorem 2 in three steps.

- **Step 1**: Derive closed-form expressions for the optimizer trajectories.
- **Step 2**: Use these expressions to analyze the resulting losses.
- **Step 3**: Characterize the cosine similarity between the learned weights.

Step 1: Derive closed-form expressions for the optimizer trajectories

We start by calculating the gradients of A and B. With Assumption 2, the loss can be simplified as follows.

$$\frac{1}{2n}\sum_{i=1}^{n}\|(H-AB)x_i\|_2^2 = \frac{1}{2}\operatorname{tr}\left[(H-AB)\left(\frac{1}{n}\sum_{i=1}^{n}x_ix_i^\top\right)(H-AB)^\top\right] = \frac{\sigma_x^2}{2}\|H-AB\|_F^2.$$

Therefore, the gradients of A and B are

$$\nabla_A L = \sigma_x^2(AB-H)B^\top, \qquad \nabla_B L = \sigma_x^2 A^\top(AB-H). \tag{4}$$

To analyze the trajectories of GD, Adam, and Muon, we decompose the hidden dimension s of parameters $A \in \mathbb{R}^{q\times s}, B \in \mathbb{R}^{s\times r}$ as $\mathbb{R}^s = \mathbb{R}^r \oplus \mathbb{R}^r \oplus \mathbb{R}^{s-2r}$. Accordingly, the parameters are decomposed as

$$A = [A_1, A_2, A_3], \qquad B = \begin{bmatrix} B_1 \\ B_2 \\ B_3 \end{bmatrix},$$

where $A_1, A_2 \in \mathbb{R}^{q\times r}$, $A_3 \in \mathbb{R}^{q\times(s-2r)}$, $B_1, B_2 \in \mathbb{R}^{r\times r}$, and $B_3 \in \mathbb{R}^{(s-2r)\times r}$. To define the trajectories of different optimizers, we introduce two auxiliary matrices $G, J \in \mathbb{R}^{q\times r}$ with orthonormal columns in the orthogonal complement of the target column space $\operatorname{col}(H)$. Specifically, we choose G and J such that

$$G^\top G = J^\top J = I_r, \qquad H^\top G = H^\top J = 0, \qquad G^\top J = 0.$$

Such matrices exist provided that $q - \operatorname{rank}(H) \geq 2r$. In addition, since $s \geq 3r$, we can choose a matrix $E \in \mathbb{R}^{(s-2r)\times r}$ with orthonormal columns:

$$E^\top E = I_r.$$

Then we present the following proposition about the trajectories of different optimizers.

Proposition 4 *For each optimizer* $\mathsf{A} \in \{\mathsf{GD}, \mathsf{Adam}, \mathsf{Muon}\}$*, initialize the parameters by*

$$A_0^{\mathsf{A}} = 0, \qquad B_0^{\mathsf{A}} = \begin{bmatrix} b_0^{\mathsf{A}} I_r \\ 0 \\ 0 \end{bmatrix}, \qquad b_0^{\mathsf{A}} > 0.$$

For any positive step sizes $\eta_t, \gamma_t, \tau_t > 0$ *for* GD, Adam, *and* Muon, *respectively, under the specified tie-breaking rules, the iterates remain in the following low-dimensional invariant subspaces. Namely, for every* $t \geq 0$*, there exist scalars* $a_t^{\mathsf{GD}}, b_t^{\mathsf{GD}}, a_t^{\mathsf{Adam}}, b_t^{\mathsf{Adam}}, \lambda_t^{\mathsf{Adam}}, a_t^{\mathsf{Muon}}, b_t^{\mathsf{Muon}}, \lambda_t^{\mathsf{Muon}} \in \mathbb{R}$ *such that*

$$A_t^{\mathsf{GD}} = [a_t^{\mathsf{GD}} H, 0, 0], \qquad B_t^{\mathsf{GD}} = \begin{bmatrix} b_t^{\mathsf{GD}} I_r \\ 0 \\ 0 \end{bmatrix}, \tag{5}$$

$$A_t^{\mathsf{Adam}} = [a_t^{\mathsf{Adam}} H, -\lambda_t^{\mathsf{Adam}} G, 0], \qquad B_t^{\mathsf{Adam}} = \begin{bmatrix} b_t^{\mathsf{Adam}} I_r \\ 0 \\ 0 \end{bmatrix}, \tag{6}$$

$$A_t^{\mathsf{Muon}} = [a_t^{\mathsf{Muon}} H, 0, -\lambda_t^{\mathsf{Muon}} J E^\top], \qquad B_t^{\mathsf{Muon}} = \begin{bmatrix} b_t^{\mathsf{Muon}} I_r \\ 0 \\ 0 \end{bmatrix}. \tag{7}$$

Proof To prove Proposition 4, we introduce the structured family of parameters that will serve as an invariant class for the optimizers' trajectories.

$$A = [aH,\ -\lambda_G G,\ -\lambda_J J E^\top], \qquad B = \begin{bmatrix} bI_r \\ 0 \\ 0 \end{bmatrix}, \tag{8}$$

where $a, b, \lambda_G, \lambda_J \in \mathbb{R}$ are real numbers. Then, with $c = ab$, we have that $AB - H = (c-1)H$. Substituting this into Eqn. (4) gives

$$\nabla_A L = \sigma_x^2 (c-1) b\, [H, 0, 0]. \tag{9}$$

For the gradient of B, the orthogonality conditions on H, G, J derive that

$$A^\top H = \begin{bmatrix} aH^\top H \\ -\lambda_G G^\top H \\ -\lambda_J E J^\top H \end{bmatrix} = \begin{bmatrix} a\mu^2 I_r \\ 0 \\ 0 \end{bmatrix},$$

where the parameter μ results from Assumption 1. Thus, the gradient of B is

$$\nabla_B L = \sigma_x^2 (c-1) \begin{bmatrix} a\mu^2 I_r \\ 0 \\ 0 \end{bmatrix}. \tag{10}$$

In the following, we will prove that GD, Adam, and Muon maintain structured weights defined in Proposition 4 with induction. We start with GD. The initialization of GD satisfies Eqn. (5) with $a_t^{\mathsf{GD}} = 0$ and $b_t^{\mathsf{GD}} = b_0^{\mathsf{GD}}$ for $t = 0$. We assume that Eqn. (5) holds for $l = 0, \cdots, t$. The parameters at step $t+1$ is updated as

$$A_{t+1}^{\mathsf{GD}} = A_t^{\mathsf{GD}} - \eta_t \nabla_A \mathcal{L}(A_t^{\mathsf{GD}}, B_t^{\mathsf{GD}}), \qquad B_{t+1}^{\mathsf{GD}} = B_t^{\mathsf{GD}} - \eta_t \nabla_B \mathcal{L}(A_t^{\mathsf{GD}}, B_t^{\mathsf{GD}}).$$

Denote $c_t^{\mathsf{GD}} = a_t^{\mathsf{GD}} b_t^{\mathsf{GD}}$, by Eqns. (9) and (10), we have that A_{t+1}^{GD} and B_{t+1}^{GD} satisfy Eqn. (5) with

$$\begin{aligned} a_{t+1}^{\mathsf{GD}} &= a_t^{\mathsf{GD}} - \eta_t \sigma_x^2 (c_t^{\mathsf{GD}} - 1) b_t^{\mathsf{GD}}, \\ b_{t+1}^{\mathsf{GD}} &= b_t^{\mathsf{GD}} - \eta_t \sigma_x^2 \mu^2 (c_t^{\mathsf{GD}} - 1) a_t^{\mathsf{GD}}. \end{aligned}$$

Thus, we prove the desired result for GD. We then analyze Adam. Since the Adam optimizer takes the element-wise sign of gradient, we consider the following tie-breaking rule

$$\mathsf{Sgn}(M) = \{Z :\ Z_{ij} = \mathrm{sign}(M_{ij}) \text{ if } M_{ij} \neq 0, \quad Z_{ij} \in [-1, 1] \text{ if } M_{ij} = 0\},$$

where $\mathrm{sign}(x)$ denotes the sign of x, with $\mathrm{sign}(0) = 0$. Then the update of Adam is

$$A_{t+1}^{\mathsf{Adam}} = A_t^{\mathsf{Adam}} - \gamma_t \Xi_t^A, \qquad B_{t+1}^{\mathsf{Adam}} = B_t^{\mathsf{Adam}} - \gamma_t \Xi_t^B,$$

where Ξ_t^A and Ξ_t^B are plausible updates, i.e.,

$$\Xi_t^A \in \mathsf{Sgn}(\nabla_A \mathcal{L}), \qquad \Xi_t^B \in \mathsf{Sgn}(\nabla_B \mathcal{L}).$$

We then prove that this update rule satisfies Eqn. (6). We first note that the initialization satisfies Eqn. (6) with $a_t^{\mathsf{Adam}} = \lambda_t^{\mathsf{Adam}} = 0$, $b_t^{\mathsf{Adam}} = b_0^{\mathsf{Adam}}$ for $t = 0$. Then we assume that Eqn. (6) holds for $l = 0, \cdots, t$. Eqs. (9) and (10). Using $\mathrm{sign}(H) = \kappa H$ imply that we can choose

$$\Xi_t^A = \mathrm{sign}((c_t^{\mathsf{Adam}} - 1) b_t^{\mathsf{Adam}}) \kappa [H, 0, 0] + [0, G, 0], \text{ and } \Xi_t^B = \mathrm{sign}((c_t^{\mathsf{Adam}} - 1) a_t^{\mathsf{Adam}}) \begin{bmatrix} I_r \\ 0 \\ 0 \end{bmatrix}.$$

The update $[0, G, 0]$ term in Ξ_t^A is admissible because it is selected on a zero-gradient block and the entries of G lie in $[-1, 1]$. Denote $c_t^{\mathsf{Adam}} = a_t^{\mathsf{Adam}} b_t^{\mathsf{Adam}}$, we have that A_{t+1}^{Adam} and B_{t+1}^{Adam} satisfy Eqn. (6) with

$$a_{t+1}^{\mathsf{Adam}} = a_t^{\mathsf{Adam}} - \gamma_t \kappa\, \mathrm{sign}((c_t^{\mathsf{Adam}} - 1) b_t^{\mathsf{Adam}}), \tag{11}$$

$$b_{t+1}^{\mathsf{Adam}} = b_t^{\mathsf{Adam}} - \gamma_t\, \mathrm{sign}((c_t^{\mathsf{Adam}} - 1) a_t^{\mathsf{Adam}}), \tag{12}$$

$$\lambda_{t+1}^{\mathsf{Adam}} = \lambda_t^{\mathsf{Adam}} + \gamma_t. \tag{13}$$

Thus, we prove the desired result for Adam. Finally, we consider Muon. Given a matrix $M \in \mathbb{R}^{m\times n}$ with rank r, let $M = U\Sigma V^\top$ be its compact SVD over the nonzero singular values, where $\Sigma \in \mathbb{R}^{r\times r}$, $U \in \mathbb{R}^{m\times r}$, and $V \in \mathbb{R}^{n\times r}$. We define the spectral normalization operator by

$$\mathsf{Spec}(M) = \left\{ Z \in \mathbb{R}^{m\times n} : \; ZV = U, \quad Z^\top U = V, \quad \|Z\|_2 \le 1 \right\},$$

where $\|\cdot\|_2$ denotes the spectral norm of matrices. Specifically, when $M = 0$, this definition gives $\mathsf{Spec}(0) = \{Z : \|Z\|_2 \le 1\}$. The Muon optimizer then updates the parameters as

$$A_{t+1}^{\mathsf{Muon}} = A_t^{\mathsf{Muon}} - \tau_t T_t^A, \qquad B_{t+1}^{\mathsf{Muon}} = B_t^{\mathsf{Muon}} - \tau_t T_t^B,$$

where T_t^A and T_t^B are plausible spectrally normalized updates, i.e.,

$$T_t^A \in \mathsf{Spec}(\nabla_A \mathcal{L}), \qquad T_t^B \in \mathsf{Spec}(\nabla_B \mathcal{L}).$$

In the following, we will prove that the parameters updated by Muon satisfiy Eqn. (7) via induction. The intialization of Muon satifies Eqn. (7) with $a_t^{\mathsf{Muon}} = \lambda_t^{\mathsf{Muon}} = 0$, $b_t^{\mathsf{Muon}} = b_0^{\mathsf{Muon}}$ for $t = 0$. We assume that Eqn. (7) holds for $l = 0, \cdots, t$. Since $H^\top H = \mu^2 I_r$, the normalized active direction associated with H is H/μ. Then we can choose T_t^A and T_t^B as follows.

$$T_t^A = \operatorname{sign}((c_t^{\mathsf{Muon}} - 1) b_t^{\mathsf{Muon}}) \left[\frac{1}{\mu} H, 0, 0 \right] + [0, 0, JE^\top], \text{ and } T_t^B = \operatorname{sign}((c_t^{\mathsf{Muon}} - 1) a_t^{\mathsf{Muon}}) \begin{bmatrix} I_r \\ 0 \\ 0 \end{bmatrix}.$$

The completion $[0, 0, JE^\top]$ acts only on a zero singular subspace. Moreover, $H^\top J = 0$ and $E^\top E = I_r$ imply that $[H/\mu, 0, 0]$ and $[0, 0, JE^\top]$ have orthogonal left and right singular subspaces. Therefore the selected T_t^A has spectral norm one and is admissible. With $c_t^{\mathsf{Muon}} = a_t^{\mathsf{Muon}} b_t^{\mathsf{Muon}}$, A_{t+1}^{Muon} and B_{t+1}^{Muon} satisfy Eqn. (7) with

$$\begin{aligned} a_{t+1}^{\mathsf{Muon}} &= a_t^{\mathsf{Muon}} - \frac{\tau_t}{\mu} \operatorname{sign}((c_t^{\mathsf{Muon}} - 1) b_t^{\mathsf{Muon}}), \\ b_{t+1}^{\mathsf{Muon}} &= b_t^{\mathsf{Muon}} - \tau_t \operatorname{sign}((c_t^{\mathsf{Muon}} - 1) a_t^{\mathsf{Muon}}), \\ \lambda_{t+1}^{\mathsf{Muon}} &= \lambda_t^{\mathsf{Muon}} + \tau_t. \end{aligned}$$

Thus, we conclude the proof of Proposition 4 ■

Step 2: Use these expressions to analyze the resulting losses.

Proposition 4 shows that for any $\mathsf{A} \in \{\mathsf{GD}, \mathsf{Adam}, \mathsf{Muon}\}$, we have that

$$A_t^{\mathsf{A}} B_t^{\mathsf{A}} = a_t^{\mathsf{A}} b_t^{\mathsf{A}} H.$$

Thus, with $c_t^{\mathsf{A}} = a_t^{\mathsf{A}} b_t^{\mathsf{A}}$, we obtain

$$\mathcal{L}_t^{\mathsf{A}}(x) = \frac{1}{2}(1 - c_t^{\mathsf{A}})^2 \|Hx\|_2^2.$$

Since Assumption 1 requires that $H^\top H = \mu^2 I_r$, every input point satisfying $\|x\|_2 = \rho$. Hence

$$\mathcal{L}_t^{\mathsf{A}}(x) = \frac{\mu^2 \rho^2}{2}(1 - c_t^{\mathsf{A}})^2.$$

The right-hand side may depend on the optimizer A and on time t, but it is constant over all points on the sphere $\|x\|_2 = \rho$. Therefore, for any two points x, x' with $\|x\|_2 = \|x'\|_2 = \rho$, and any two optimizers $\mathsf{A}, \mathsf{A}' \in \{\mathsf{GD}, \mathsf{Adam}, \mathsf{Muon}\}$, we have

$$L_t^{\mathsf{A}}(x) - L_t^{\mathsf{A}}(x') = L_t^{\mathsf{A}'}(x) - L_t^{\mathsf{A}'}(x').$$

This proves exact RGI on the validation sphere.

Step 3: Characterize the cosine similarity between the learned weights.

We denote the vectorized parameters as $\theta_t^{\mathsf{A}} = (\mathrm{vec}(A_t^{\mathsf{A}}), \mathrm{vec}(B_t^{\mathsf{A}}))$, where $\mathrm{vec}(\cdot)$ vectorizes the matrix parameter. Then the cosine similarity between parameters learned by different optimizers can be calculated as follows.

Proposition 5 *Under the same conditions of Proposition 4, the cosine similarities between parameters learned by* GD*,* Adam*, and* Muon *are*

$$\text{cos-sim}(\theta_t^{\mathsf{GD}}, \theta_t^{\mathsf{Adam}}) = \frac{\mu^2 a_t^{\mathsf{GD}} a_t^{\mathsf{Adam}} + b_t^{\mathsf{GD}} b_t^{\mathsf{Adam}}}{w_t^{\mathsf{GD}} \cdot w_t^{\mathsf{Adam}}},$$
$$\text{cos-sim}(\theta_t^{\mathsf{GD}}, \theta_t^{\mathsf{Muon}}) = \frac{\mu^2 a_t^{\mathsf{GD}} a_t^{\mathsf{Muon}} + b_t^{\mathsf{GD}} b_t^{\mathsf{Muon}}}{w_t^{\mathsf{GD}} \cdot w_t^{\mathsf{Muon}}},$$
$$\text{cos-sim}(\theta_t^{\mathsf{Adam}}, \theta_t^{\mathsf{Muon}}) = \frac{\mu^2 a_t^{\mathsf{Adam}} a_t^{\mathsf{Muon}} + b_t^{\mathsf{Adam}} b_t^{\mathsf{Muon}}}{w_t^{\mathsf{Adam}} \cdot w_t^{\mathsf{Muon}}},$$

where the denominators are defined as $w_t^{\mathsf{GD}} = \sqrt{\mu^2 (a_t^{\mathsf{GD}})^2 + (b_t^{\mathsf{GD}})^2}$, $w_t^{\mathsf{Adam}} = \sqrt{\mu^2 (a_t^{\mathsf{Adam}})^2 + (b_t^{\mathsf{Adam}})^2 + (\lambda_t^{\mathsf{Adam}})^2}$, *and* $w_t^{\mathsf{Muon}} = \sqrt{\mu^2 (a_t^{\mathsf{Muon}})^2 + (b_t^{\mathsf{Muon}})^2 + (\lambda_t^{\mathsf{Muon}})^2}$. *Each displayed ratio applies when its denominator is nonzero; otherwise, the corresponding similarity is zero by convention.*

Proof To compute the cosine similarity between two parameter vectors, we normalize their inner product by the product of their norms. We first record several basic identities for the corresponding inner products and norms. Specifically, we have

$$\langle [H, 0, 0], [0, G, 0] \rangle_F = 0, \quad \langle [H, 0, 0], [0, 0, JE^\top] \rangle_F = 0, \quad \langle [0, G, 0], [0, 0, JE^\top] \rangle_F = 0.$$

Moreover, since $H^\top H = \mu^2 I_r, G^\top G = I_r, J^\top J = I_r, E^\top E = I_r$, we have that

$$\|[H, 0, 0]\|_F^2 = \mu^2 r, \quad \|[0, G, 0]\|_F^2 = \|G\|_F^2 = r, \quad \|[0, 0, JE^\top]\|_F^2 = r.$$

Thus, for GD and Adam, the inner product between parameters are

$$\langle \theta_t^{\mathsf{GD}}, \theta_t^{\mathsf{Adam}} \rangle = r \left(\mu^2 a_t^{\mathsf{GD}} a_t^{\mathsf{Adam}} + b_t^{\mathsf{GD}} b_t^{\mathsf{Adam}} \right).$$

Moreover, the norms of the parameters are

$$\|\theta_t^{\mathsf{GD}}\|_2^2 = r \left(\mu^2 (a_t^{\mathsf{GD}})^2 + (b_t^{\mathsf{GD}})^2 \right), \quad \|\theta_t^{\mathsf{Adam}}\|_2^2 = r \left(\mu^2 (a_t^{\mathsf{Adam}})^2 + (b_t^{\mathsf{Adam}})^2 + (\lambda_t^{\mathsf{Adam}})^2 \right).$$

Combining these, we have that

$$\text{cos-sim}(\theta_t^{\mathsf{GD}}, \theta_t^{\mathsf{Adam}}) = \frac{\mu^2 a_t^{\mathsf{GD}} a_t^{\mathsf{Adam}} + b_t^{\mathsf{GD}} b_t^{\mathsf{Adam}}}{w_t^{\mathsf{GD}} \cdot w_t^{\mathsf{Adam}}}.$$

Similarly, the cosine similarity between the parameters of GD and Muon is

$$\text{cos-sim}(\theta_t^{\mathsf{GD}}, \theta_t^{\mathsf{Muon}}) = \frac{\mu^2 a_t^{\mathsf{GD}} a_t^{\mathsf{Muon}} + b_t^{\mathsf{GD}} b_t^{\mathsf{Muon}}}{w_t^{\mathsf{GD}} \cdot w_t^{\mathsf{Muon}}}.$$

For the cosine similarity between the parameters learned by Adam and Muon, we have that

$$\langle [0, G, 0], [0, 0, JE^\top] \rangle_F = 0.$$

Then we calculate the cosine similarity as

$$\text{cos-sim}(\theta_t^{\mathsf{Adam}}, \theta_t^{\mathsf{Muon}}) = \frac{\mu^2 a_t^{\mathsf{Adam}} a_t^{\mathsf{Muon}} + b_t^{\mathsf{Adam}} b_t^{\mathsf{Muon}}}{w_t^{\mathsf{Adam}} \cdot w_t^{\mathsf{Muon}}}.$$

Thus, we conclude the proof of Proposition 5. ■

In the following, we will prove that a_t and b_t of Adam and Muon are uniformly bounded, whereas λ_t of them grows linearly. We use constant step sizes for them, i.e., $\gamma_t = \gamma, \tau_t = \tau$.

First, we consider Adam. Recall that we initialize the parameter with $a_0^{\mathsf{Adam}} = 0, \lambda_0^{\mathsf{Adam}} = 0, b_0^{\mathsf{Adam}} > 0$. Since $c_0^{\mathsf{Adam}} = a_0^{\mathsf{Adam}} b_0^{\mathsf{Adam}} = 0$, the first update gives

$$a_1^{\mathsf{Adam}} = a_0^{\mathsf{Adam}} - \gamma\kappa \operatorname{sign}\left((c_0^{\mathsf{Adam}} - 1) b_0^{\mathsf{Adam}}\right) = \kappa\gamma,$$
$$b_1^{\mathsf{Adam}} = b_0^{\mathsf{Adam}} - \gamma \operatorname{sign}\left((c_0^{\mathsf{Adam}} - 1) a_0^{\mathsf{Adam}}\right) = b_0^{\mathsf{Adam}},$$

where the equalities result from Eqn. (11) and (12). Thus, the first update leads to $a_1^{\mathsf{Adam}} > 0, b_1^{\mathsf{Adam}} > 0$. We then define

$$\Delta_{\mathsf{Adam}} = a_1^{\mathsf{Adam}} - \kappa b_1^{\mathsf{Adam}} = \kappa(\gamma - b_0^{\mathsf{Adam}}), \quad D_{\mathsf{Adam}} = |\Delta_{\mathsf{Adam}}|, \quad r_{\mathsf{Adam}} = \frac{1}{D_{\mathsf{Adam}} + \sqrt{\kappa}}.$$

Then we have the following results about Adam.

Proposition 6 *With the step-size condition* $\gamma_t = \gamma < \min\left\{b_0^{\mathsf{Adam}}, (\kappa b_0^{\mathsf{Adam}} + \sqrt{\kappa})^{-1}\right\}$, *we have that* $a_t^{\mathsf{Adam}} > 0, b_t^{\mathsf{Adam}} > 0$ *and* $a_t^{\mathsf{Adam}} - \kappa b_t^{\mathsf{Adam}} = \Delta_{\mathsf{Adam}}$ *hold for all* $t \geq 1$.

Proof

We prove the desired results by induction. The results trivially hold for $t = 1$. Suppose the claims hold at time $t \geq 1$. Since $a_t^{\mathsf{Adam}} > 0$ and $b_t^{\mathsf{Adam}} > 0$, we have

$$\operatorname{sign}\left((c_t^{\mathsf{Adam}} - 1)b_t^{\mathsf{Adam}}\right) = \operatorname{sign}(c_t^{\mathsf{Adam}} - 1), \quad \operatorname{sign}\left((c_t^{\mathsf{Adam}} - 1)a_t^{\mathsf{Adam}}\right) = \operatorname{sign}(c_t^{\mathsf{Adam}} - 1).$$

Therefore, Eqn. (11) and (12) show that $a_{t+1}^{\mathsf{Adam}} = a_t^{\mathsf{Adam}} - \gamma\kappa \operatorname{sign}(c_t^{\mathsf{Adam}} - 1)$, and $b_{t+1}^{\mathsf{Adam}} = b_t^{\mathsf{Adam}} - \gamma \operatorname{sign}(c_t^{\mathsf{Adam}} - 1)$. It follows that

$$a_{t+1}^{\mathsf{Adam}} - \kappa b_{t+1}^{\mathsf{Adam}} = a_t^{\mathsf{Adam}} - \kappa b_t^{\mathsf{Adam}} = \Delta_{\mathsf{Adam}}.$$

It remains to show that the next iterate stays in the positive quadrant. Since $a_t^{\mathsf{Adam}} = \kappa b_t^{\mathsf{Adam}} + \Delta_{\mathsf{Adam}}$, if $0 < b_t^{\mathsf{Adam}} \leq r_{\mathsf{Adam}}$, then

$$c_t^{\mathsf{Adam}} = \left(\kappa b_t^{\mathsf{Adam}} + \Delta_{\mathsf{Adam}}\right) b_t^{\mathsf{Adam}} \leq \kappa (b_t^{\mathsf{Adam}})^2 + D_{\mathsf{Adam}} b_t^{\mathsf{Adam}} \leq \kappa r_{\mathsf{Adam}}^2 + D_{\mathsf{Adam}} r_{\mathsf{Adam}}.$$

By the definition of r_{Adam}, we have

$$\kappa r_{\mathsf{Adam}}^2 + D_{\mathsf{Adam}} r_{\mathsf{Adam}} = \frac{\kappa}{(D_{\mathsf{Adam}} + \sqrt{\kappa})^2} + \frac{D_{\mathsf{Adam}}}{D_{\mathsf{Adam}} + \sqrt{\kappa}} \leq 1.$$

Hence, we can show that if $c_t^{\mathsf{Adam}} > 1$, we have $b_t^{\mathsf{Adam}} > r_{\mathsf{Adam}}$. Similarly, since $b_t^{\mathsf{Adam}} = (a_t^{\mathsf{Adam}} - \Delta_{\mathsf{Adam}})/\kappa$, if $0 < a_t^{\mathsf{Adam}} \leq \kappa r_{\mathsf{Adam}}$, then

$$c_t^{\mathsf{Adam}} = \frac{a_t^{\mathsf{Adam}}\left(a_t^{\mathsf{Adam}} - \Delta_{\mathsf{Adam}}\right)}{\kappa} \leq \frac{a_t^{\mathsf{Adam}}\left(a_t^{\mathsf{Adam}} + D_{\mathsf{Adam}}\right)}{\kappa} \leq \kappa r_{\mathsf{Adam}}^2 + D_{\mathsf{Adam}} r_{\mathsf{Adam}} \leq 1.$$

Hence, we also show that if $c_t^{\mathsf{Adam}} > 1$, we have $a_t^{\mathsf{Adam}} > \kappa r_{\mathsf{Adam}}$.

Now consider two cases. If $c_t^{\mathsf{Adam}} \leq 1$, then $\operatorname{sign}(c_t^{\mathsf{Adam}} - 1) \leq 0$, by the update rule in Eqn. (11) and (12), we have

$$a_{t+1}^{\mathsf{Adam}} \geq a_t^{\mathsf{Adam}} > 0, \qquad b_{t+1}^{\mathsf{Adam}} \geq b_t^{\mathsf{Adam}} > 0.$$

If $c_t^{\mathsf{Adam}} > 1$, then we have that $b_t^{\mathsf{Adam}} > r_{\mathsf{Adam}}, a_t^{\mathsf{Adam}} > \kappa r_{\mathsf{Adam}}$. Since step-size condition $\gamma_t = \gamma < \min\left\{b_0^{\mathsf{Adam}}, (\kappa b_0^{\mathsf{Adam}} + \sqrt{\kappa})^{-1}\right\}$ implies $0 < \gamma < r_{\mathsf{Adam}}$, we have

$$b_{t+1}^{\mathsf{Adam}} = b_t^{\mathsf{Adam}} - \gamma > r_{\mathsf{Adam}} - \gamma > 0, \quad a_{t+1}^{\mathsf{Adam}} = a_t^{\mathsf{Adam}} - \kappa\gamma > \kappa r_{\mathsf{Adam}} - \kappa\gamma > 0.$$

Thus, the positive quadrant is invariant for Adam. We conclude the proof of Proposition 6. ■

In the following, we will show that a_t^{Adam} and b_t^{Adam} are bounded. Define

$$R_{\mathsf{Adam}} = \max\left\{\frac{2D_{\mathsf{Adam}}}{\kappa}, \sqrt{\frac{2}{\kappa}}\right\}, \quad M_{\mathsf{Adam}} = \max\left\{b_0^{\mathsf{Adam}}, R_{\mathsf{Adam}} + \gamma\right\}.$$

We prove by induction that $b_t^{\mathsf{Adam}} \leq M_{\mathsf{Adam}}$ for all t. The claim holds at $t = 0$. Suppose $b_t^{\mathsf{Adam}} \leq M_{\mathsf{Adam}}$. If $b_t^{\mathsf{Adam}} \geq R_{\mathsf{Adam}}$, then we have that

$$a_t^{\mathsf{Adam}} = \kappa b_t^{\mathsf{Adam}} + \Delta_{\mathsf{Adam}} \geq \kappa b_t^{\mathsf{Adam}} - D_{\mathsf{Adam}} \geq \frac{\kappa}{2} b_t^{\mathsf{Adam}},$$

where the first equality results from Proposition 6. Therefore, we have $c_t^{\mathsf{Adam}} = a_t^{\mathsf{Adam}} b_t^{\mathsf{Adam}} \geq \kappa/2 \cdot (b_t^{\mathsf{Adam}})^2 \geq 1$. Hence, we have if $b_t^{\mathsf{Adam}} \geq R_{\mathsf{Adam}}$, then $b_{t+1}^{\mathsf{Adam}} \leq b_t^{\mathsf{Adam}} \leq M_{\mathsf{Adam}}$. If $b_t^{\mathsf{Adam}} < R_{\mathsf{Adam}}$, then we have that

$$b_{t+1}^{\mathsf{Adam}} = b_t^{\mathsf{Adam}} - \gamma \operatorname{sign}(c_t^{\mathsf{Adam}} - 1) \leq b_t^{\mathsf{Adam}} + \gamma < R_{\mathsf{Adam}} + \gamma \leq M_{\mathsf{Adam}}.$$

Therefore, we show that $b_t^{\mathsf{Adam}} \le M_{\mathsf{Adam}}$ for all t. Furthermore, since $a_t^{\mathsf{Adam}} = \kappa b_t^{\mathsf{Adam}} + \Delta_{\mathsf{Adam}}$, we also have

$$0 < a_t^{\mathsf{Adam}} \le \kappa M_{\mathsf{Adam}} + D_{\mathsf{Adam}}.$$

Thus, we show that $a_t^{\mathsf{Adam}}, b_t^{\mathsf{Adam}} = O(1)$ with t increasing.

Second, we consider Muon. The parameters are initialized with $a_0^{\mathsf{Muon}} = 0, \lambda_t^{\mathsf{Muon}} = 0, b_0^{\mathsf{Muon}} > 0$. We then define

$$\Delta_{\mathsf{Muon}} = a_1^{\mathsf{Muon}} - \frac{1}{\mu} b_1^{\mathsf{Muon}}, \quad D_{\mathsf{Muon}} = |\Delta_{\mathsf{Muon}}|, \quad r_{\mathsf{Muon}} = \frac{1}{D_{\mathsf{Muon}} + \mu^{-1/2}}.$$

The step-size condition of $\tau_t = \tau < \min\left\{b_0^{\mathsf{Muon}}, (b_0^{\mathsf{Muon}}/\mu + \mu^{-1/2})^{-1}\right\}$ implies $0 < \tau < r_{\mathsf{Muon}}$. Similar to Proposition 6 for Adam, we can also show by induction that $a_t^{\mathsf{Muon}} > 0, b_t^{\mathsf{Muon}} > 0$ and $a_t^{\mathsf{Muon}} - \mu^{-1} b_t^{\mathsf{Muon}} = \Delta_{\mathsf{Muon}}$ for all $t \ge 1$. For the boundness, we define

$$R_{\mathsf{Muon}} = \max\left\{2\mu D_{\mathsf{Muon}}, \sqrt{2\mu}\right\}, \quad M_{\mathsf{Muon}} = \max\left\{b_0^{\mathsf{Muon}}, R_{\mathsf{Muon}} + \tau\right\}.$$

Similar to our proof for Adam, we can show by induction that $b_t^{\mathsf{Muon}} \le M_{\mathsf{Muon}}$ for all $t \ge 1$. Since $a_t^{\mathsf{Muon}} = \mu^{-1} b_t^{\mathsf{Muon}} + \Delta_{\mathsf{Muon}}$, we also have $0 < a_t^{\mathsf{Muon}} \le \mu^{-1} M_{\mathsf{Muon}} + D_{\mathsf{Muon}}$. Thus, we have $a_t^{\mathsf{Muon}}, b_t^{\mathsf{Muon}} = O(1)$ when t increases.

In the following, we then derive the bounds for the cosine similarities between parameters. Since $\lambda_t^{\mathsf{Adam}} = \lambda_0^{\mathsf{Adam}} + \gamma t$ and $\lambda_t^{\mathsf{Muon}} = \lambda_0^{\mathsf{Muon}} + \tau t$, we have that

$$|\lambda_t^{\mathsf{Adam}}| = \Theta(t), \quad |\lambda_t^{\mathsf{Muon}}| = \Theta(t).$$

From the boundedness proved above, there exist finite constants $C_{\mathsf{Adam}}, C_{\mathsf{Muon}}$ such that for all t,

$$\mu^2 (a_t^{\mathsf{Adam}})^2 + (b_t^{\mathsf{Adam}})^2 \le C_{\mathsf{Adam}}^2, \quad \mu^2 (a_t^{\mathsf{Muon}})^2 + (b_t^{\mathsf{Muon}})^2 \le C_{\mathsf{Muon}}^2.$$

Thus, for GD and Adam, we have

$$\left|\text{cos-sim}(\theta_t^{\mathsf{GD}}, \theta_t^{\mathsf{Adam}})\right| \le \frac{\sqrt{\mu^2 (a_t^{\mathsf{Adam}})^2 + (b_t^{\mathsf{Adam}})^2}}{\sqrt{\mu^2 (a_t^{\mathsf{Adam}})^2 + (b_t^{\mathsf{Adam}})^2 + (\lambda_t^{\mathsf{Adam}})^2}} \le \frac{C_{\mathsf{Adam}}}{|\lambda_t^{\mathsf{Adam}}|} = O(t^{-1}),$$

where the first inequality results from the Cauchy-Schwarz inequality, and the second inequality results from the boundedness of a_t^{Adam} and b_t^{Adam}. Similarly, we have that

$$\left|\text{cos-sim}(\theta_t^{\mathsf{GD}}, \theta_t^{\mathsf{Muon}})\right| \le \frac{C_{\mathsf{Muon}}}{|\lambda_t^{\mathsf{Muon}}|} = O(t^{-1}), \ \left|\text{cos-sim}(\theta_t^{\mathsf{Adam}}, \theta_t^{\mathsf{Muon}})\right| \le \frac{C_{\mathsf{Adam}} C_{\mathsf{Muon}}}{|\lambda_t^{\mathsf{Adam}}||\lambda_t^{\mathsf{Muon}}|} = O(t^{-2}).$$

Thus, we conclude the proof of Theorem 2.