Nexus: Same Pretraining Loss, Better Downstream Generalization via Common Minima

author names withheld

Under Review for the Workshop on High-dimensional Learning Dynamics, 2025

Abstract

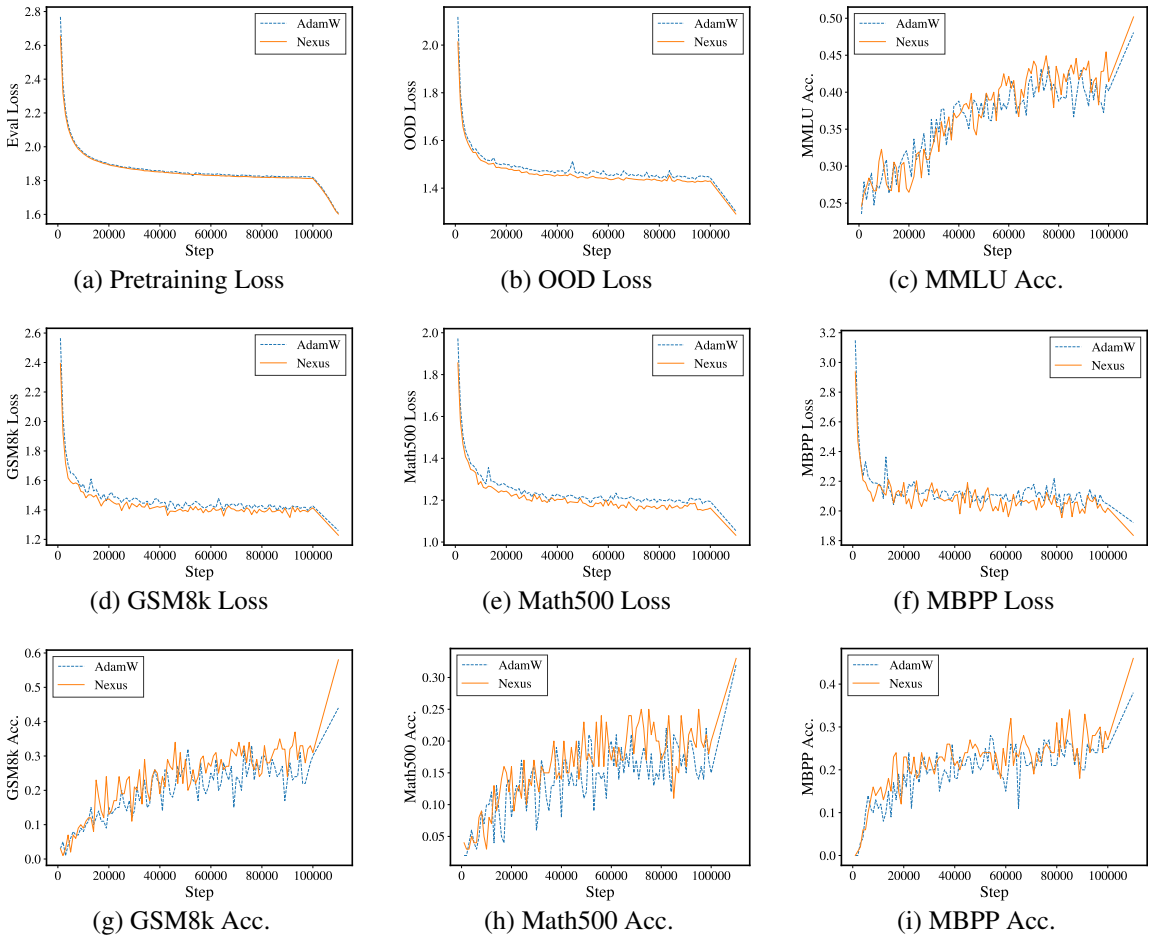
The foundational capabilities of large language models are acquired during pretraining on internet-scale, highly heterogeneous data mixtures. In this work, we investigate a geometric question regarding the converged state of this pretraining process: Does the model converge to a common minimizer across all data sources (e.g., Fig. 2 (b)), or merely a minimizer of the averaged loss (e.g., Fig. 2 (a))? We hypothesize that the geometric "closeness" of task-specific minima is intrinsically linked to downstream generalization. However, we reveal that standard optimizers (e.g., AdamW) often converge to points where task-specific minima are distant from each other. To address this, we propose the *Nexus* optimizer, which encourages the closeness of these minima by maximizing gradient similarity during optimization. Experiments across models ranging from 130M to 3B parameters demonstrate that Nexus boosts downstream performance, despite achieving nearly *the same pretraining loss* (see Fig. 1). On the 3B model, Nexus reduces the out-of-distribution loss by 0.012 and yields up to a 15.0% accuracy improvement on complex reasoning tasks (e.g., GSM8k). Our findings challenge the reliance on pretraining loss as the sole proxy for model evaluation and highlight the critical role of optimizer implicit bias in unlocking downstream generalization.

1. Introduction

Pretraining is the cornerstone of Large Language Models (LLMs). Accounting for over 95% of the total computational budget and data, it serves as the indispensable engine for their capabilities [17, 41, 79]. During pretraining, LLMs acquire foundational knowledge from an unprecedentedly massive and diverse data sources, encompassing a vast array of domains such as general language, mathematics, code, and complex reasoning [2, 54, 71]. To learn from such a heterogeneous corpus of K distinct sources, the standard practice is to average the loss of each data source $\mathcal{L}_k(\boldsymbol{\theta})$ and minimize the averaged loss $\mathcal{L}_{\text{train}}(\boldsymbol{\theta}) = \frac{1}{K}\sum_{k=1}^{K}\mathcal{L}_k(\boldsymbol{\theta})$.

In this work, we investigate an interesting geometric question: *Does the model converge to a common minimizer across all data sources $\mathcal{L}_k$, or does it merely find a minimizer of the averaged loss $\mathcal{L}_{train}$?* To illustrate this, consider a simplified setting composed of two data sources ($\mathcal{L}_1$ and $\mathcal{L}_2$), yielding a training loss of $\mathcal{L}_{\text{train}}(\boldsymbol{\theta}) = \frac{1}{2}(\mathcal{L}_1(\boldsymbol{\theta}) + \mathcal{L}_2(\boldsymbol{\theta}))$. As depicted in Fig. 2, there exist two distinct types of minimizers that achieve the exact same training loss $\mathcal{L}_{\text{train}}$. The first corresponds to the **Average-type Minimizer** (Sec. 1), where the converged parameter $\boldsymbol{\theta}^*_{\text{train}}$ successfully minimizes the total training loss $\mathcal{L}_{\text{train}}$ yet may remain geometrically distant from the minimizers of individual tasks $\mathcal{L}_k$. The second approaches the **Intersection-type Minimizer** (Sec. 1), where $\boldsymbol{\theta}^*_{\text{train}}$ is not only a minimizer of $\mathcal{L}_{\text{train}}$, but is also geometrically close to the minimizer of each individual task $\mathcal{L}_k$.

We hypothesize that this geometric "**closeness**"—the distance between task-specific minima—is strongly correlated with **downstream generalization**. Even when achieving the exact *same*



(a) Pretraining Loss (b) OOD Loss (c) MMLU Acc.

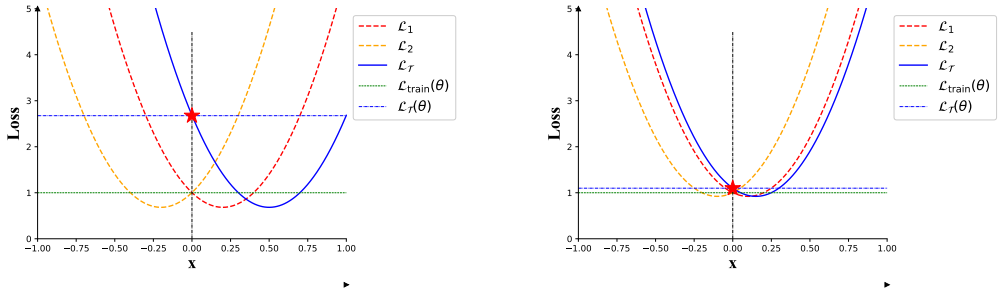
(d) GSM8k Loss (e) Math500 Loss (f) MBPP Loss

(g) GSM8k Acc. (h) Math500 Acc. (i) MBPP Acc.

Figure 1: Illustration of "same pretraining loss, better downstream task". The training loss of baseline and our Nexus are nearly the same. However, our methods achieves better downstream generalization.

pretraining loss, these two types of minimizers yield drastically *different downstream* losses $\mathcal{L}_{\mathcal{T}}$ (see the blue curve $\mathcal{L}_{\mathcal{T}}(\boldsymbol{\theta})$ in Fig. 2). Intuitively, if the training losses $\mathcal{L}_k$ and downstream task $\mathcal{L}_{\mathcal{T}}$ are quadratic and i.i.d. distributed, the Intersection-type minimizer (Sec. 1) will strictly outperform the Average-type minimizer (Sec. 1) on the downstream task $\mathcal{L}_{\mathcal{T}}$, given the same pretraining loss (see Theorem 2). Therefore, we posit that this intuition may generalize beyond quadratics to LLM pretraining, and steering the optimization toward the Intersection-type minimizer would achieve the "same pretraining loss, better downstream task".

However, directly optimizing for this geometric "closeness" is computationally intractable, as it requires knowing the exact minimizer of each $\mathcal{L}_k$ at every training step. To overcome this, we prove that the gradient similarity between tasks, $\mathrm{CosSim}(\nabla\mathcal{L}_i, \nabla\mathcal{L}_j) \triangleq \frac{\nabla\mathcal{L}_i^\top \nabla\mathcal{L}_j}{\|\nabla\mathcal{L}_i\|\|\nabla\mathcal{L}_j\|}$, upper bounds the geometric closeness. The rationale is straightforward: if the gradient directions of each loss $\nabla\mathcal{L}_k$ are always exactly the same throughout optimization, their respective minimizers $\boldsymbol{\theta}_k^*$ must be exactly the same. Based on this insight, we propose the Nexus algorithm, which approximates the gradient of gradient similarity $\nabla\mathrm{CosSim}(\nabla\mathcal{L}_i, \nabla\mathcal{L}_j)$. Nexus is compatible with the existing optimization techniques [30, 34, 77] to effectively maximize $\mathrm{CosSim}(\nabla\mathcal{L}_i, \nabla\mathcal{L}_j)$. In Appendix B.5, we show that



(a) Distant (Average-type Minimizer) (b) Close (Intersection-type Minimizer)

Figure 2: Illustration of two types of minimizers. (a) Distant: Minimizers of each source are distant from each other. (b) Close: Minimizers are geometrically close to each other. Although both configurations achieve the *same pretraining loss*, they perform *differently on downstream task* $\mathcal{L}_\mathcal{T}$.

both gradient and geometric closeness generalize to downstream tasks, leading to lower downstream loss and better downstream performance, even when achieving the same pretraining loss.

We empirically validate Nexus across various settings, including model scales ranging from 130M to 3B parameters [73, 77, 79], diverse pretraining data and mixtures [5, 70], learning rate schedules [27, 46, 78], and training compute [31]. Experimental results demonstrate that, across nearly all settings, *Nexus reduces the downstream loss by over 0.02 compared to the base optimizers—a substantial margin that typically requires doubling the pretraining compute [31]—while achieving the exact same pretraining loss*. For instance, on the 3B model, Nexus improves GSM8K accuracy by 15%, MATH500 by 8% and HumanEval by 4%. These consistent downstream gains demonstrate the importance of implicit biases in unlocking downstream generalization [43], particularly as the current pretraining paradigm transitions from being compute-bound to data-bound [33, 51, 58, 67].

2. Closeness and Downstream Generalization

Formally, let the pretraining corpus be the union of K distinct data sources, and the total pretraining objective be the average of these source losses $\mathcal{L}_{\text{train}}(\boldsymbol{\theta}) = \frac{1}{K}\sum_{k=1}^{K}\mathcal{L}_k(\boldsymbol{\theta})$.

Our primary interest lies in how well the pretraining minimizer $\boldsymbol{\theta}^*_{\text{train}} \in \arg\min_{\boldsymbol{\theta}} \mathcal{L}_{\text{train}}(\boldsymbol{\theta})$ performs on an unseen downstream task $\mathcal{T}$. By applying a second-order Taylor expansion around the local minimizer of the downstream task, $\boldsymbol{\theta}^*_\mathcal{T}$, the downstream loss $\mathcal{L}_\mathcal{T}(\boldsymbol{\theta}^*_{\text{train}})$ can be bounded by:

$$\mathcal{L}_\mathcal{T}(\boldsymbol{\theta}^*_{\text{train}}) \le \mathcal{L}_\mathcal{T}(\boldsymbol{\theta}^*_\mathcal{T}) + \frac{1}{2}\underbrace{\|\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\theta}^*_\mathcal{T}\|_2^2}_{\text{Closeness}} \cdot \underbrace{\max_{\boldsymbol{\xi}\in[\boldsymbol{\theta}^*_\mathcal{T},\boldsymbol{\theta}^*_{\text{train}}]} \|\nabla^2\mathcal{L}_\mathcal{T}(\boldsymbol{\xi})\|_2}_{\text{Flatness}}. \tag{1}$$

The first-order term vanishes as $\boldsymbol{\theta}^*_\mathcal{T}$ is a local minimizer. Eq. (1) reveals that the generalization gap is controlled by two factors: the *Flatness* of the loss landscape and the *Closeness* between the converged parameter and the task optimal. Therefore, a *flatter* local landscape and a *closer* converged parameter naturally yield better downstream generalization. This bound may serve as an highly accurate proxy rather than a loose upper estimation, since empirical loss landscapes often exhibit high quadraticity along typical optimization directions [8, 35, 76], under which Eq. (1) becomes tight.

Appendix C.1 demonstrates that closeness will improve downstream generalization, even when achieving the same pre-training loss, as long as the loss landscape is quadratic-like. Therefore,

explicitly optimizing the closeness σ^2 to reduce the task variance may be beneficial for downstream performance.

3. Nexus Optimizer: Enhancing Closeness via Second-Order Approximation

3.1. Gradient Similarity Upper Bounds Closeness

However, directly optimizing the closeness metric $\|\boldsymbol{\theta}-\boldsymbol{\theta}_k^*\|$ involves finding the minimizer $\boldsymbol{\theta}_k^*$ for each task, which is itself a minimization problem and computationally prohibitive. Fortunately, we observe that the gradient similarity between different source tasks, given by $\sum_{i\neq j} -\nabla\mathcal{L}_i(\boldsymbol{\theta})^\top\nabla\mathcal{L}_j(\boldsymbol{\theta})$, provides a tractable upper bound for closeness. Intuitively, if the gradients of distinct tasks $\mathcal{L}_k$ consistently align in direction, their respective minimizers be exactly the same. Theoretically, both the gradient dot product and cosine similarity serve as tight bounds for closeness:

Theorem 1 (Gradient Similarity Upper Bounds Closeness) *Let $\boldsymbol{\theta}$ be a converged parameter satisfying $\nabla\mathcal{L}_{train}(\boldsymbol{\theta}) = \mathbf{0}$. For each task k, let $\boldsymbol{\theta}_k^*$ denote its closest local minimizer to $\boldsymbol{\theta}$. Assume the loss is directionally strongly convex along the path $[\boldsymbol{\theta}, \boldsymbol{\theta}_k^*]$ with a minimum curvature $\lambda_{\min} > 0$, and let $G = \sup_k \|\nabla\mathcal{L}_k(\boldsymbol{\theta})\|_2$. Then, the closeness is bounded by:*

$$\frac{1}{K}\sum_{k=1}^{K}\|\boldsymbol{\theta}-\boldsymbol{\theta}_k^*\|_2^2 \leq \frac{1}{K\lambda_{\min}^2}\sum_{i\neq j}\left(-\nabla\mathcal{L}_i(\boldsymbol{\theta})^\top\nabla\mathcal{L}_j(\boldsymbol{\theta})\right) \leq \frac{G^2}{K\lambda_{\min}^2}\sum_{i\neq j}\left(1-\mathit{CosSim}(\nabla\mathcal{L}_i(\boldsymbol{\theta}),\nabla\mathcal{L}_j(\boldsymbol{\theta}))\right).$$

In other words, optimizing the training trajectory towards a regime where $\text{CosSim}(\nabla\mathcal{L}_i(\boldsymbol{\theta}),\nabla\mathcal{L}_j(\boldsymbol{\theta}))$ are consistently high guarantees high closeness (i.e., small $\|\boldsymbol{\theta}-\boldsymbol{\theta}_k^*\|_2$). This "gradient similarity upper bound" also provides an intuitive understanding of why closeness improves downstream generalization. Suppose the high gradient similarity achieved among training tasks $\text{Sim}(\nabla\mathcal{L}_i(\boldsymbol{\theta}),\nabla\mathcal{L}_j(\boldsymbol{\theta}))$ successfully generalizes to the similarity between the training objective and the downstream task $\text{Sim}(\nabla\mathcal{L}_{\text{train}}(\boldsymbol{\theta}),\nabla\mathcal{L}_{\mathcal{T}}(\boldsymbol{\theta}))$. This similarity directly represents the reduction in downstream loss after a single gradient descent step on the training set (in the first-order sense):

$$\underbrace{\mathcal{L}_{\mathcal{T}}(\boldsymbol{\theta}) - \mathcal{L}_{\mathcal{T}}(\boldsymbol{\theta}-\gamma\nabla\mathcal{L}_{\text{train}}(\boldsymbol{\theta}))}_{\text{decrease of downstream loss after one GD step on training set}} = \gamma\nabla\mathcal{L}_{\text{train}}(\boldsymbol{\theta})^\top\nabla\mathcal{L}_{\mathcal{T}}(\boldsymbol{\theta}) + O(\gamma^2). \tag{2}$$

Therefore, we view gradient similarity $\text{CosSim}(\nabla\mathcal{L}_i(\boldsymbol{\theta}),\nabla\mathcal{L}_j(\boldsymbol{\theta}))$ as a strong proxy for parameter closeness: it not only provides an upper bound, but also leads to the same beneficial effects on downstream generalization. Given this strong connection, in the remainder of this paper, we use the term "closeness" to refer to both parameter closeness and gradient closeness.

3.2. Optimizing Gradient Similarity via Nexus

However, directly optimizing gradient similarity involves the Jacobian-vector product:

$$\nabla_{\boldsymbol{\theta}}\text{CosSim}(\nabla_{\boldsymbol{\theta}}\mathcal{L}_i(\boldsymbol{\theta}),\nabla_{\boldsymbol{\theta}}\mathcal{L}_j(\boldsymbol{\theta})) = \left(\nabla_{\boldsymbol{\theta}}\frac{\nabla_{\boldsymbol{\theta}}\mathcal{L}_i(\boldsymbol{\theta})}{\|\nabla_{\boldsymbol{\theta}}\mathcal{L}_i(\boldsymbol{\theta})\|_2}\right)^\top\frac{\nabla_{\boldsymbol{\theta}}\mathcal{L}_j(\boldsymbol{\theta})}{\|\nabla_{\boldsymbol{\theta}}\mathcal{L}_j(\boldsymbol{\theta})\|_2} + \left(\nabla_{\boldsymbol{\theta}}\frac{\nabla_{\boldsymbol{\theta}}\mathcal{L}_j(\boldsymbol{\theta})}{\|\nabla_{\boldsymbol{\theta}}\mathcal{L}_j(\boldsymbol{\theta})\|_2}\right)^\top\frac{\nabla_{\boldsymbol{\theta}}\mathcal{L}_i(\boldsymbol{\theta})}{\|\nabla_{\boldsymbol{\theta}}\mathcal{L}_i(\boldsymbol{\theta})\|_2}. \tag{3}$$

To address this challenge, we propose the **Nexus optimizer**, which approximates the gradient in Eq. (3) through a dual-loop mechanism. The complete procedure is outlined in Algorithm 1. Conceptually, one should view each step in the outer loop as a standard parameter update, while the K steps in the inner loop serve as a *gradient approximator* for Eq. (3). Specifically, for each outer

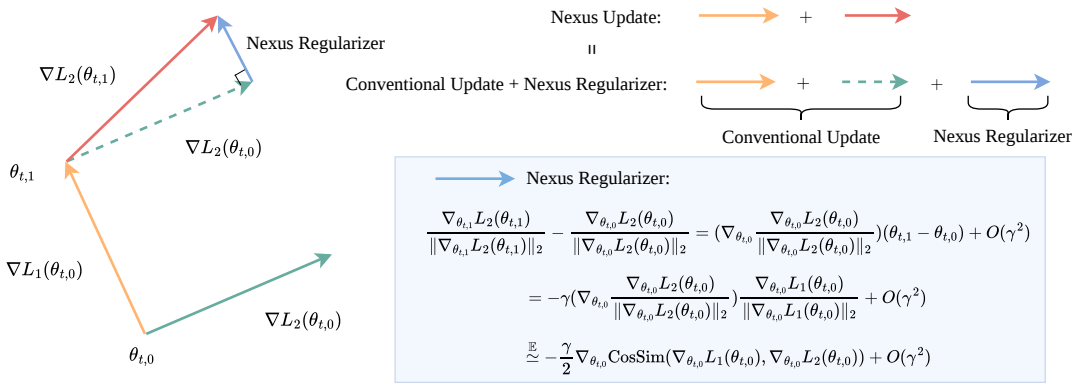


Figure 3: Illustration of Nexus Algorithm. $\overset{\mathbb{E}}{\simeq}$ denotes expected equality over task permutations.

iteration, we perform K normalized SGD steps (inner loop) to accumulate the approximated gradient $\hat{\boldsymbol{g}}_t$. This $\hat{\boldsymbol{g}}_t$ is then passed to the outer optimizer (e.g., AdamW [34, 45], SGDM [49]) to perform the actual update. Theorem 7 demonstrates that Nexus maximizes gradient similarity.

Intuitive Understanding of the Inner Loop. Consider a simplified scenario with two loss functions, $\mathcal{L}_1$ and $\mathcal{L}_2$, as illustrated in Fig. 3. At the current parameter state $\boldsymbol{\theta}_{t,0}$, a conventional optimizer (e.g., AdamW, Muon) would simply aggregate the gradients as $\nabla\mathcal{L}_1(\boldsymbol{\theta}_{t,0}) + \nabla\mathcal{L}_2(\boldsymbol{\theta}_{t,0})$ for the update. In contrast, the Nexus inner loop operates sequentially: it first takes a step using $\nabla\mathcal{L}_1(\boldsymbol{\theta}_{t,0})$ to reach an intermediate point $\boldsymbol{\theta}_{t,1}$, and subsequently evaluates the next gradient $\nabla\mathcal{L}_2(\boldsymbol{\theta}_{t,1})$ at this displaced location. As shown in the figure's equations, this sequential trajectory is mathematically equivalent to a conventional update plus a "Nexus regularizer". Crucially, this regularizer naturally yields a Jacobian-vector product $\left(\nabla_{\boldsymbol{\theta}_{t,0}} \frac{\nabla\mathcal{L}_2(\boldsymbol{\theta}_{t,0})}{\|\nabla\mathcal{L}_2(\boldsymbol{\theta}_{t,0})\|_2}\right) \frac{\nabla\mathcal{L}_1(\boldsymbol{\theta}_{t,0})}{\|\nabla\mathcal{L}_1(\boldsymbol{\theta}_{t,0})\|_2}$, which equals the gradient of the gradient similarity (in the first-order and expectation sense). Consequently, the nexus pseudo-gradient $\hat{\boldsymbol{g}}_t$ produced by the inner loop effectively serves as the sum of the gradient of the pretraining loss and the gradient of the gradient similarity defined in Eq. (3).

We adapt Nexus to practical pretraining in Appendix B.1 such that it introduces no additional computational overhead. Refer to Appendix B for the experimental results.

4. Conclusion and Limitation

In this work, we investigate the geometric closeness of task-specific minimizers in LLM pretraining and establish its strong correlation with downstream generalization. To optimize this closeness, we propose the Nexus algorithm, which maximizes gradient similarity across tasks. We demonstrate that this gradient consensus generalizes to unseen tasks, thus translating into reduced downstream losses and superior benchmark performance. As the LLM scaling paradigm transitions from compute-bound to data-bound, we argue that explicitly engineering the implicit biases of optimizers to unlock generalization serves as a critical frontier for developing more capable language models.

Limitations. Despite its empirical success with AdamW, Nexus is currently incompatible with the Muon optimizer. Specifically, Muon-Nexus underperforms AdamW-Nexus on downstream tasks due to optimization deceleration (in contrast to the slight acceleration observed with AdamW, see Appendix C.2). We hypothesize this stems from numerical sensitivities involving the pseudo-gradient coefficient γ (Eq. (57)) or complex interactions with Muon's Newton-Schulz iterations. Resolving this challenge remains an active direction for future work.

References

[1] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. *Advances in neural information processing systems*, 29, 2016.

[2] Anthropic. The claude 3 model family: Opus, sonnet, haiku. 2024.

[3] Sanjeev Arora, Nadav Cohen, Wei Hu, and Yuping Luo. Implicit regularization in deep matrix factorization. *Advances in neural information processing systems*, 2019.

[4] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

[5] Aarti Basant, Abhijit Khairnar, Abhijit Paithankar, Abhinav Khattar, Adithya Renduchintala, Aditya Malte, Akhiad Bercovich, Akshay Hazare, Alejandra Rico, Aleksander Ficek, et al. Nvidia nemotron nano 2: An accurate and efficient hybrid mamba-transformer reasoning model. *arXiv preprint arXiv:2508.14444*, 2025.

[6] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, pages 15849–15854, 2019.

[7] Huanran Chen, Yichi Zhang, Yinpeng Dong, Xiao Yang, Hang Su, and Jun Zhu. Rethinking model ensemble in transfer-based adversarial attacks. In *The Twelfth International Conference on Learning Representations*, 2024.

[8] Huanran Chen, Yinpeng Dong, Zeming Wei, Yao Huang, Yichi Zhang, Hang Su, and Jun Zhu. Understanding pre-training and fine-tuning from loss landscape perspectives. *arXiv preprint arXiv:2505.17646*, 2025.

[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*, 2023.

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

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

[12] Jeremy M Cohen, Simran Kaur, Yuanzhi Li, J Zico Kolter, and Ameet Talwalkar. Gradient descent on neural networks typically occurs at the edge of stability. *arXiv preprint arXiv:2103.00065*, 2021.

[13] Alex Damian, Tengyu Ma, and Jason D Lee. Label noise sgd provably prefers flat global minimizers. *Advances in Neural Information Processing Systems*, pages 27449–27461, 2021.

[14] Alex Damian, Eshaan Nichani, and Jason D Lee. Self-stabilization: The implicit bias of gradient descent at the edge of stability. *arXiv preprint arXiv:2209.15594*, 2022.

[15] Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024.

[16] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[17] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[18] Mark Chen et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[19] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. On the convergence theory of gradient-based model-agnostic meta-learning algorithms. In *International conference on artificial intelligence and statistics*, pages 1082–1092, 2020.

[20] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, 2017.

[21] Suriya Gunasekar, Jason D Lee, Daniel Soudry, and Nati Srebro. Implicit bias of gradient descent on linear convolutional networks. *Advances in Neural Information Processing Systems*, 2018.

[22] Guy Gur-Ari, Daniel A Roberts, and Ethan Dyer. Gradient descent happens in a tiny subspace. *arXiv preprint arXiv:1812.04754*, 2018.

[23] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

[24] Jeff Z HaoChen, Colin Wei, Jason Lee, and Tengyu Ma. Shape matters: Understanding the implicit bias of the noise covariance. In *Conference on Learning Theory*, 2021.

[25] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

[26] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

[27] 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.

[28] Ziwei Ji and Matus Telgarsky. Gradient descent aligns the layers of deep linear networks. In *International Conference on Learning Representations*, 2018.

[29] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in Neural Information Processing Systems*, 2013.

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

[31] 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.

[32] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International conference on machine learning*, pages 5132–5143, 2020.

[33] Konwoo Kim, Suhas Kotha, Percy Liang, and Tatsunori Hashimoto. Pre-training under infinite compute. *arXiv preprint arXiv:2509.14786*, 2025.

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

[35] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *Advances in Neural Information Processing Systems*, 31, 2018.

[36] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, pages 429–450, 2020.

[37] Yuanzhi Li, Tengyu Ma, and Hongyang Zhang. Algorithmic regularization in over-parameterized matrix sensing and neural networks with quadratic activations. In *Conference on learning theory*, 2018.

[38] Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-sgd: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*, 2017.

[39] Zhiyuan Li, Yuping Luo, and Kaifeng Lyu. Towards resolving the implicit bias of gradient descent for matrix factorization: Greedy low-rank learning. *arXiv preprint arXiv:2012.09839*, 2020.

[40] Kaizhao Liang, Lizhang Chen, Bo Liu, and Qiang Liu. Cautious optimizers: Improving training with one line of code. *arXiv preprint arXiv:2411.16085*, 2024.

[41] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[42] Hong Liu, Zhiyuan Li, David Leo Wright Hall, Percy Liang, and Tengyu Ma. Sophia: A scalable stochastic second-order optimizer for language model pre-training. In *The Twelfth International Conference on Learning Representations*, 2023.

[43] Hong Liu, Sang Michael Xie, Zhiyuan Li, and Tengyu Ma. Same pre-training loss, better downstream: Implicit bias matters for language models. In *International Conference on Machine Learning*, pages 22188–22214. PMLR, 2023.

[44] Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, et al. Muon is scalable for llm training. *arXiv preprint arXiv:2502.16982*, 2025.

[45] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[46] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017.

[47] Kaifeng Lyu and Jian Li. Gradient descent maximizes the margin of homogeneous neural networks. In *International Conference on Learning Representations*, 2019.

[48] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282, 2017.

[49] Yurii Nesterov et al. *Lectures on convex optimization*, volume 137. Springer, 2018.

[50] Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. In search of the real inductive bias: On the role of implicit regularization in deep learning. *arXiv preprint arXiv:1412.6614*, 2014.

[51] Jinjie Ni, Qian Liu, Longxu Dou, Chao Du, Zili Wang, Hang Yan, Tianyu Pang, and Michael Qizhe Shieh. Diffusion language models are super data learners. *arXiv preprint arXiv:2511.03276*, 2025.

[52] Alex Nichol and John Schulman. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999*, 2018.

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

[54] OpenAI. Gpt-4 technical report. *arXiv*, 2023.

[55] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[56] Thomas Pethick, Wanyun Xie, Kimon Antonakopoulos, Zhenyu Zhu, Antonio Silveti-Falls, and Volkan Cevher. Training deep learning models with norm-constrained lmos. In *International Conference on Machine Learning*, 2025.

[57] Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*, 2022.

[58] Mihir Prabhudesai, Mengning Wu, Amir Zadeh, Katerina Fragkiadaki, and Deepak Pathak. Diffusion beats autoregressive in data-constrained settings. *arXiv preprint arXiv:2507.15857*, 2025.

[59] Shikai Qiu, Zixi Chen, Hoang Phan, Qi Lei, and Andrew Gordon Wilson. Hyperparameter transfer enables consistent gains of matrix-preconditioned optimizers across scales. *arXiv preprint arXiv:2512.05620*, 2025.

[60] Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid learning or feature reuse? towards understanding the effectiveness of maml. *arXiv preprint arXiv:1909.09157*, 2019.

[61] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *International conference on learning representations*, 2017.

[62] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

[63] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *International conference on machine learning*, pages 4334–4343. PMLR, 2018.

[64] Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng. Meta-weight-net: Learning an explicit mapping for sample weighting. *Advances in neural information processing systems*, 32, 2019.

[65] Minhak Song, Kwangjun Ahn, and Chulhee Yun. Does sgd really happen in tiny subspaces? *arXiv preprint arXiv:2405.16002*, 2024.

[66] Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *Journal of Machine Learning Research*, 2018.

[67] Jacob Mitchell Springer, Sachin Goyal, Kaiyue Wen, Tanishq Kumar, Xiang Yue, Sadhika Malladi, Graham Neubig, and Aditi Raghunathan. Overtrained language models are harder to fine-tune. *arXiv preprint arXiv:2503.19206*, 2025.

[68] Sebastian U Stich. Local sgd converges fast and communicates little. *arXiv preprint arXiv:1805.09767*, 2018.

[69] Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, , and Jason Wei. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*, 2022.

[70] ByteDance Seed Team. Seed-oss open-source models. https://github.com/ByteDance-Seed/seed-oss, 2025.

[71] Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

[72] Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

[73] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[74] Nikhil Vyas, Depen Morwani, Rosie Zhao, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham M Kakade. Soap: Improving and stabilizing shampoo using adam for language modeling. In *The Thirteenth International Conference on Learning Representations*, 2024.

[75] Ishaan Watts, Catherine Li, Sachin Goyal, Jacob Mitchell Springer, and Aditi Raghunathan. Sharpness-aware pretraining mitigates catastrophic forgetting.

[76] Kaiyue Wen, Tengyu Ma, and Zhiyuan Li. How does sharpness-aware minimization minimize sharpness? *arXiv preprint arXiv:2211.05729*, 2022.

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

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

[79] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[80] 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*, 2021.

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

[82] Greg Yang, James B Simon, and Jeremy Bernstein. A spectral condition for feature learning. *arXiv preprint arXiv:2310.17813*, 2023.

[83] Huizhuo Yuan, Yifeng Liu, Shuang Wu, Zhou Xun, and Quanquan Gu. Mars: Unleashing the power of variance reduction for training large models. In *International Conference on Machine Learning*, 2025.

[84] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

[85] Yushun Zhang, Congliang Chen, Tian Ding, Ziniu Li, Ruoyu Sun, and Zhiquan Luo. Why transformers need adam: A hessian perspective. *Advances in neural information processing systems*, 37:131786–131823, 2024.

[86] 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. In *The Thirteenth International Conference on Learning Representations*, 2024.

Appendix A. Extended Background and Setup

A.1. Related Work

Pretraining Optimizers. Various optimizers have been proposed for LLM pretraining, such as AdamW [34, 45], Muon [30], Lion [9], SOAP [74], Sophia [42], MARS [83], Adam-mini [86], Cautious [40], and Scion [56]. Most of these optimizers primarily focus on achieving faster optimization and accelerating the convergence of the pretraining loss. In contrast, our work focuses on the generalization aspect, aiming to achieve better downstream task performance under the same pretraining loss. Since Nexus serves as a gradient approximator prior to the optimizer update, our method is largely orthogonal to the choice of the base optimizer and may be used in conjunction with these existing optimization algorithms.

Implicit Biases. Implicit bias is a well-studied topic in deep learning theory, as it is closely related to downstream generalization, especially under the zero training loss or same pretraining loss [3, 21, 24, 28, 37, 39, 43, 47, 66]. Our work differs from previous studies in two main aspects. First, we focus on gradient and parameter closeness, whereas previous studies primarily investigate sharpness and margins. Second, we evaluate our method in modern auto-regressive LLM pretraining settings, as opposed to standard CNN, DNN, BERT or adversarial attack [7]. A concurrent work [75] also demonstrates improved downstream performance at the same pretraining loss in auto-regressive LLMs, but their approach relies on flatness rather than closeness.

Meta-Learning and Decentralized Learning. The nested inner-outer loop optimization structure utilized by Nexus has been extensively explored in fields like meta-learning [19, 20, 38, 52, 60] and decentralized learning [32, 36, 48, 68]. However, rather than aiming for few-shot fast adaptation during SFT or communication efficiency, our work repurposes this structure with two distinct contributions. First, we provide a rigorous mathematical characterization demonstrating that this inner-outer loop mechanism inherently optimizes gradient similarity (Theorem 7 and Appendix C.5). Second, we apply this mechanism to LLM pretraining, demonstrating its empirical value in achieving better downstream task performance under the same pretraining loss. Due to these different objectives, the practical algorithmic designs also diverge. For instance, while some meta-learning methods can flexibly incorporate momentum in the inner loop, the inner update in Nexus must strictly remain a memory-free step (such as vanilla or normalized SGD, see Appendix C.5); otherwise, historical momentum would unequally weight the first-order loss gradients from different steps. We believe other structural algorithms from the meta-learning literature, such as meta-optimizers [1, 61], hypernetworks [23], and meta-learned data weighting schemes [63, 64], might also yield unexpected implicit regularization benefits when applied to large-scale pretraining, which we leave as an open direction for future work.

A.2. Notations

The primary mathematical notations for data mixtures, loss functions, geometries, and optimization dynamics are summarized in Tab. 1.

A.3. Assumptions

The main assumptions used in our analysis are outlined below [11, 78]. Additional assumptions required for specific analyses will be stated in the respective theorems.

Table 1: Summary of key notations used in this paper.

Notation	Description
	Data and Loss Functions
K	Total number of distinct pretraining data sources (tasks).
α_k	Sampling probability (data mixing ratio) for the k-th data source.
$\mathcal{L}_k(\boldsymbol{\theta})$	The expected / empirical loss on the k-th source task.
$\mathcal{L}_{\text{train}}(\boldsymbol{\theta})$	The averaged pretraining loss: $\frac{1}{K}\sum_{k=1}^{K}\mathcal{L}_k(\boldsymbol{\theta})$.
$\mathcal{L}_{\mathcal{T}}(\boldsymbol{\theta})$	The loss on an unseen downstream evaluation task $\mathcal{T}$.
	Geometric and Statistical Variables
$\boldsymbol{\theta}^*_{\text{train}}$	The converged parameter state that minimizes $\mathcal{L}_{\text{train}}$.
$\mathcal{S}_k, \mathcal{S}_{\mathcal{T}}$	The set of local minimizers for task k and downstream task $\mathcal{T}$, respectively.
$\boldsymbol{\theta}^*_k, \boldsymbol{\theta}^*_{\mathcal{T}}$	The specific local minimizer in $\mathcal{S}_k$ or $\mathcal{S}_{\mathcal{T}}$ closest to the current parameter.
$\boldsymbol{\mu}$	The statistical center of task-specific minimizers: $\mathbb{E}[\boldsymbol{\theta}^*_k]$.
σ^2	The intrinsic variance (Closeness) of task-specific minimizers: $\mathbb{E}[\|\boldsymbol{\theta}^*_k - \boldsymbol{\mu}\|_2^2]$.
	Optimization and Nexus Variables
γ	The inner learning rate (step size) used in the Nexus gradient approximator.
$\hat{\boldsymbol{g}}_t$	The Nexus pseudo-gradient (displacement) passed to the outer optimizer.
$\text{CosSim}(\boldsymbol{x}, \boldsymbol{y})$	The cosine similarity between two vectors: $\frac{\boldsymbol{x}^\top \boldsymbol{y}}{\|\boldsymbol{x}\|_2 \|\boldsymbol{y}\|_2}$.
$S_{ij}(\boldsymbol{\theta})$	Shorthand for gradient similarity: $\text{CosSim}(\nabla\mathcal{L}_i(\boldsymbol{\theta}), \nabla\mathcal{L}_j(\boldsymbol{\theta}))$.

Assumption 1 (Bounded Gradients) *For all tasks $i \in [1, K]$ and parameters $\boldsymbol{\theta}$ along the optimization trajectory, the gradient norm is strictly bounded from below and above:*

$$0 < G_{\min} \leq \|\nabla\mathcal{L}_i(\boldsymbol{\theta})\|_2 \leq G$$

This ensures that the Normalized SGD step in Nexus is always well-defined and numerically stable.

Assumption 2 (Smoothness and Bounded Curvature) *The loss function $\mathcal{L}_i$ is L-smooth, meaning its Hessian spectral norm is bounded from above. Furthermore, within the local basin of attraction $[\boldsymbol{\theta}, \boldsymbol{\theta}^*_k]$, the curvature is strictly lower-bounded by $\lambda_{min} > 0$:*

$$\lambda_{min} \leq \inf_{\boldsymbol{\xi} \in [\boldsymbol{\theta}, \boldsymbol{\theta}^*_k]} \left(\boldsymbol{u}^\top \nabla^2 \mathcal{L}_i(\boldsymbol{\xi}) \boldsymbol{u} \right) \leq \|\nabla^2 \mathcal{L}_i(\boldsymbol{\theta})\|_2 \leq L$$

where $\boldsymbol{u}$ is any unit vector. Prior literature extensively characterizes the local loss landscape of deep neural networks as exhibiting high quadraticity, particularly along meaningful optimization trajectories [8, 35, 76]. Consequently, under the standard premise that the loss landscape can be locally and directionally approximated by a quadratic function, this bounded curvature condition should not be viewed as a restrictive assumption.

Assumption 3 (Hessian Lipschitz Continuous) *The Hessian matrix is ρ-Lipschitz continuous. For any parameters $\boldsymbol{x}, \boldsymbol{y}$:*

$$\|\nabla^2 \mathcal{L}_i(\boldsymbol{x}) - \nabla^2 \mathcal{L}_i(\boldsymbol{y})\|_2 \leq \rho \|\boldsymbol{x} - \boldsymbol{y}\|_2$$

Algorithm 1 Standard Pretraining

```
Require: model, loader
Require: opt_outer (e.g., AdamW)
Require: accum_steps
for i, batch in loader do
    {Mini-batch Step}
    ℒ ← model(batch)
    ℒ.backward()
    if i % accum_steps == 0 then
        {Accumulation Step}
        opt_outer.step()
        opt_outer.zero_grad()
    end if
end for
```

Algorithm 2 Nexus (Engineering Adaptation)

```
Require: model, loader, opt_outer, accum_steps
inner_model ← model.clone()
opt_inner ← NSGD(inner_model)
for i, batch in loader do
    ℒ ← inner_model(batch)
    ℒ.backward()
    opt_inner.step()
    if i % accum_steps == 0 then
        ĝ ← inner_model − model
        opt_outer.step(grad=−ĝ)
        inner_model ← model.clone()
    end if
end for
```

Figure 4: **Comparison of Standard Pretraining and Nexus Engineering Adaptation. Left:** Standard training accumulates gradients over multiple mini-batches before performing a single optimizer update (at the micro-batch/accumulation step). **Right:** We adapt Nexus from Algorithm 1 to pretraining by keeping an auxiliary `inner_model`. It performs immediate updates on the `inner_model` at every mini-batch step. At the accumulation boundary, the total displacement (`inner_model` − `model`) serves as the pseudo-gradient $\hat{\boldsymbol{g}}$ for the outer optimizer.

Appendix B. Experiments

In this section and Appendix I, we validate that Nexus achieves nearly the *same pretraining loss* while delivering *better downstream performance* through comprehensive experiments across various datasets, learning rate schedules, model scales, token scales, and SFT. Due to page limits, extensive additional results and ablation studies are presented in Appendix I.

B.1. Adapting Nexus to Practical Pretraining

Motivation. While Nexus effectively maximizes gradient similarity, directly applying Algorithm 1 to pretraining requires computing gradients for every data source to perform a single effective outer update. In pretraining, the number of data sources is typically large (e.g., $K > 50$), which would result in an effective batch size that differs significantly from standard settings [31, 77]. This prevents us from leveraging established hyperparameters, thereby increasing tuning costs and preventing the wide application. To address this, we propose an engineering adaptation. As shown in **??**, standard pretraining can be viewed as a gradient accumulation workflow: it computes gradients in every mini-batch and performs an optimizer update in every accumulation step.

Adapted Nexus. Leveraging this structure, we adapt Nexus as illustrated in Fig. 4. Specifically, we introduce an auxiliary `inner_model`. For each mini-batch, we perform an immediate Normalized SGD (NSGD) update on this inner model to approximate the hessian-gradient product. Upon completing the accumulation steps, we compute the displacement between the inner model and the frozen main model, using this displacement as the pseudo-gradient $\hat{\boldsymbol{g}}$ for the outer optimizer. Therefore, our adapted Nexus actually *maximizes the cosine similarity between mini-batches within a single accumulation step*. Since the pretraining corpus is typically vast and the mixing ratio for each source is typically low, two consecutive mini-batches are highly likely to be sampled from different sources. Thus, this approach effectively achieves the objective of Algorithm 1.

Table 2: **Main Results.** Nexus achieves nearly *identical pretraining losses* compared to the base optimizers, yet demonstrates *superior performance across downstream losses and benchmarks*.

Model	Optim.	Metric	Loss Metrics (↓)		Gen.	Reasoning			Math		Code		Avg.
			Pretrain.	OOD	MMLU	GPQA	GPQA-D	BBH	GSM8k	MATH	HumanEval	MBPP	All
1B	AdamW	Acc. (↑)	1.826	1.433	32.1	25.0	21.8	29.6	18.0	13.0	19.0	17.0	21.9
		Loss (↓)			2.363	2.221	2.124	1.640	1.429	1.204	1.270	2.035	1.786
	Nexus	Acc. (↑)	1.826	1.428	33.5	**30.4**	21.8	29.3	20.0	13.0	19.0	**22.0**	23.6
		Loss (↓)			**2.316**	**2.201**	**2.102**	1.638	**1.396**	**1.176**	1.261	**1.977**	**1.758**
	Improv.	Acc. (↑)	-	-	+1.4	**+5.4**	0.0	-0.3	+2.0	0.0	0.0	**+5.0**	+1.7
		Loss (↑)	0.000	+0.005	**+0.047**	**+0.020**	**+0.022**	+0.002	**+0.033**	**+0.028**	+0.009	**+0.058**	**+0.027**
3B	AdamW	Acc. (↑)	1.606	1.302	47.8	**32.8**	22.6	36.6	44.0	32.0	43.0	38.0	37.1
		Loss (↓)			2.265	2.005	1.910	1.534	1.259	1.054	1.116	1.922	1.633
	Nexus	Acc. (↑)	1.602	**1.290**	48.9	29.6	**23.4**	36.6	**59.0**	**40.0**	**47.0**	38.0	**40.3**
		Loss (↓)			**2.179**	**1.981**	**1.881**	**1.504**	**1.227**	**1.026**	**1.086**	1.921	**1.601**
	Improv.	Acc. (↑)	-	-	+1.1	-3.2	+0.8	0.0	**+15.0**	**+8.0**	**+4.0**	0.0	**+3.2**
		Loss (↑)	+0.004	**+0.012**	**+0.086**	**+0.024**	**+0.029**	**+0.030**	**+0.032**	**+0.028**	**+0.030**	+0.001	**+0.032**

Clarification on NSGD Normalization. Note that we apply *per-param* normalization rather than normalizing the globally flattened gradient vector. Mathematically, this adapted Nexus maximizes the sum of per-matrix gradient similarities $\sum_l \text{CosSim}(\nabla_{\boldsymbol{W}^{(l)}}\mathcal{L}_i, \nabla_{\boldsymbol{W}^{(l)}}\mathcal{L}_j)$ (see Appendix C.5), rather than the global gradient similarity $\text{CosSim}(\nabla_{\boldsymbol{\theta}}\mathcal{L}_i, \nabla_{\boldsymbol{\theta}}\mathcal{L}_j)$. This design choice is primarily motivated by the theoretical insights of μP [80–82] and recent practical wisdom [44] on controlling the RMS norm of each update, which potentially enables more stable optimization and better hyper-parameter transfer across different model scales.

Remark. Our adapted Nexus incurs *almost no extra computational cost*. The total number of forward and backward passes remains exactly the same as standard pretraining. The only computational overhead comes from the copy and update of the inner model, but this is negligible compared to the forward-backward pass [31]. The only memory overhead comes from the inner model, but this can be reduced to nearly zero through techniques like CPU offloading and asynchronous processing.

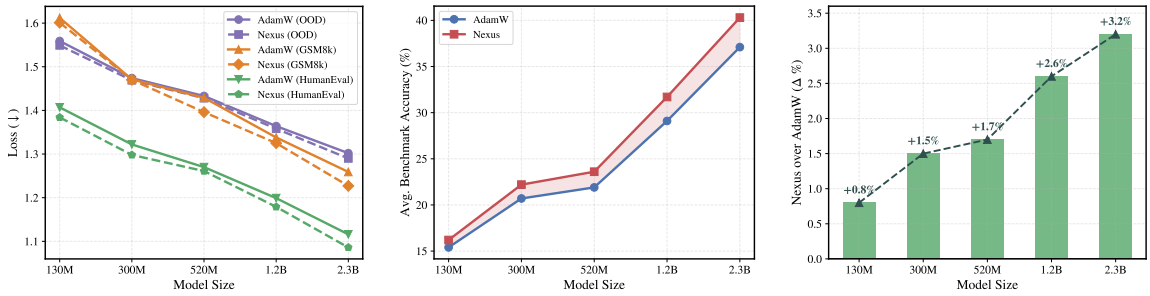
We employ Fig. 4 for all experiments, with the exception of specific ablation studies. Readers may proceed directly to Appendix B. In Appendix C, we also provide a theoretical analysis of Nexus's convergence speed and discuss its implications for standard Normalized SGD.

B.2. Experimental Settings

Our experimental setup strictly follows the protocols in Wen et al. [77]. We train Llama-architecture models ranging from 130M to 3B parameters. We mainly pretrain on a strictly cleaned, high-quality in-house corpus for experiments on other datasets). For evaluation, we track the OOD validation loss alongside 8 benchmarks. To ensure fair comparison, Nexus and all base optimizers strictly use the exact same optimal hyperparameters identified in Wen et al. [77] without any Nexus-specific tuning. See Appendix I.1 for detailed configurations.

B.3. Main Experimental Results

Settings. We train 1B models by 4× Chinchilla and 3B models for 2× Chinchilla tokens using the standard AdamW baseline and AdamW equipped with our Nexus regularizer (Nexus).



(a) Downstream Loss (b) Downstream Benchmark (c) Average Benchmark Gain

Figure 5: **Performance across Model Scales.** The relative gains of Nexus amplify as model capacity increases, growing from +0.8% on the 130M model to +3.2% on the 2.3B model.

Nexus achieves Same Pretraining Loss, Better Downstream Task. As detailed in Tab. 2, Nexus strictly satisfies the "same pretraining loss" condition, showing an immaterial difference of 0.004 compared to the baseline. Despite this parity in pretraining loss, Nexus demonstrates substantial improvements across nearly all evaluated out-of-distribution and downstream metrics. Specifically, it reduces the OOD validation loss by 0.012 and yields significant accuracy gains on complex reasoning benchmarks, including a +15.0% improvement on GSM8k, +8.0% on MATH, and +4.0% on HumanEval. These consistent gains across diverse domains validate our core hypothesis: optimizing closeness unlocks downstream generalization in the same pretraining loss regime.

Comparison of Muon and Nexus. Compared to the standard AdamW baseline, Muon reduces the pretraining loss by 0.029 and improves the average downstream accuracy by 2.3%. In contrast, Nexus achieves a negligible 0.004 reduction in pretraining loss yet yields a 3.2% improvement in average downstream accuracy, reaching a downstream performance level comparable to Muon (see Tab. 10). This observation indicates a divergence in their optimization pathways: while Muon's downstream improvements rely primarily on achieving a lower pretraining loss, the gains from Nexus stem from its implicit bias despite maintaining a nearly identical pretraining loss as AdamW.

Output Analysis. Compared to the AdamW baseline, Nexus improves accuracy by 15.0% on GSM8k, 8.0% on MATH, and 4.0% on HumanEval. To investigate these improvements, we analyze the model outputs on these benchmarks. We observe that the set of correctly answered questions by Nexus is almost a strict superset of those answered correctly by AdamW. Specifically, on GSM8k and HumanEval, Nexus retains a >95% retention rate on the questions already solved by AdamW, while the 15.0% net improvement stems entirely from exclusively solving previously failed questions. This behavior indicates that the performance gain provided by Nexus over the base optimizer is stable, expanding the capability boundaries without regressing on previously learned knowledge.

B.4. Scaling Analysis

Motivation. We investigate the scalability of Nexus on model size and training duration (tokens). Prevailing literature suggests that implicit regularization becomes more prominent with overparameterization and extended computational budgets, since sufficient expressive power and optimization steps grant the model the flexibility to satisfy the geometric implicit bias without compromising the minimization of the pretraining loss [6, 47, 50, 57, 66, 76, 84]. Since Nexus operates via such implicit bias, we hypothesize that its downstream benefits will also amplify at larger scales.

Table 3: **Gradient Similarity, Loss, and Benchmarks.** Nexus achieves higher gradient similarity between the pretraining and downstream corpus. Consistent with Eq. (2), this first-order gradient similarity translates to lower zero-th order downstream losses, yielding better downstream performance.

Metric	Optim.	Pretrain Set	OOD Set	GPQA-D	GSM8k	Math500	HumanEval	MBPP
Grad Sim. (↑)	AdamW	0.4499	0.2228	0.0824	**0.0374**	0.0422	0.0367	0.0091
	Nexus	**0.4661**	**0.2464**	**0.0924**	0.0325	**0.0427**	**0.0382**	**0.0092**
Param. Closeness. (↓)	AdamW	1.452	2.812	4.500	3.418	3.775	4.472	**3.645**
	Nexus	**1.441**	**2.806**	**4.482**	**3.326**	**3.766**	**4.444**	3.648
Loss (↓)	AdamW	1.606	1.302	1.910	1.259	1.054	1.116	1.922
	Nexus	1.602	**1.290**	**1.881**	**1.227**	**1.026**	**1.086**	1.921
Benchmark (↑)	AdamW	-	-	22.6	44.0	32.0	43.0	38.0
	Nexus	-	-	23.4	**59.0**	**40.0**	**47.0**	38.0

Settings. We evaluate models across five distinct sizes ranging from 130M to 2.3B as outlined in Appendix I.1. To evaluate scalability with respect to tokens, we extend the training duration of the 3B model from the standard $2\times$ Chinchilla optimal token count to $4\times$ Chinchilla optimal. Please refer to Appendix I.2 for the detailed hyperparameters of each experiment.

Universal "Same Pretraining Loss, Better Downstream". As shown in Tab. 9 and Fig. 5, Nexus consistently maintains the pretraining loss within a negligible margin (defined as $\Delta < 0.01$ in Appendix I.1) compared to the baseline, satisfying "same pretraining loss". Despite this parity in pretraining loss, Nexus achieves non-trivial loss reduction on nearly all downstream tasks. For instance, at the 1.2B scale, Nexus reduces MMLU loss by 0.086, and both BBH and HumanEval losses by 0.03, more than 7 times larger than the pretraining loss gap.

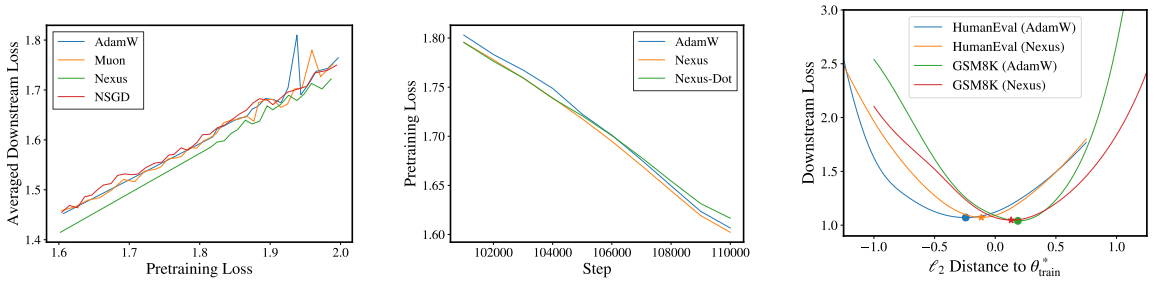
Performance Gains Amplify with Scale. We observe that the relative advantage of Nexus over the AdamW baseline expands monotonically as model capacity increases: the average benchmark improvements across the five scales are +0.8% (130M), +1.5% (300M), +1.7% (520M), +2.6% (1.2B), and +3.2% (2.3B). This demonstrate that Nexus scales favorably with model capacity, effectively leveraging the increased expressive power to enforce the geometric closeness bias.

The advantage of Nexus does not diminish with more training tokens. As shown in Tab. 8, the overall performance gap between Nexus and AdamW does not shrink with more tokens; Nexus at $4\times$ Chinchilla achieves an average accuracy of 41.7, effectively maintaining and even slightly widening its substantial lead over the baseline.

B.5. Training Dynamic of Nexus

Settings. We analyze the training trajectories of the 3B AdamW and 3B Nexus models from Appendix B.3. During pretraining, we record the gradient cosine similarity between test set and each downstream corpus every 1,000 steps and compute the average gradient similarity. After pretraining, we perform full batch GD using AdamW with learning rate 2×10^{-5} and weight decay 0 on each downstream task $\mathcal{L}_{\mathcal{T}}$ to locate the respective task-specific minimizer $\boldsymbol{\theta}^*_{\mathcal{T}}$.

Nexus Encourages Training Set Closeness. As demonstrated in Tab. 3, Nexus effectively increases the gradient similarity across the pretraining set, $\mathbb{E}_{i\neq j}[\text{CosSim}(\nabla\mathcal{L}_i, \nabla\mathcal{L}_j)]$, compared to the base optimizer, as analyzed in Theorem 5 and 7.



(a) Implicit Biases (b) Nexus-Dot (c) Loss Landscape

Figure 6: **Ablation Studies.** (a) Implicit biases of various optimizers, shown by pretraining vs. downstream loss correlation. (b) Pretraining loss of Nexus-Dot, showing the unnormalized dot product disrupts loss minimization. (c) Loss landscapes of Adam and Nexus.

Training Set Closeness Generalizes to Downstream Closeness. Fortunately, this gradient closeness generalizes beyond the pretraining corpus to unseen downstream tasks $\mathcal{T}$, effectively increasing the similarity between the training objective and the downstream task, $\text{CosSim}(\nabla\mathcal{L}_{\text{train}}, \nabla\mathcal{L}_{\mathcal{T}})$.

Downstream Closeness Yields Smaller Downstream Loss and Better Performance. Since this gradient similarity $\text{CosSim}(\nabla\mathcal{L}_{\text{train}}, \nabla\mathcal{L}_{\mathcal{T}})$ generalizes, optimizing the pretraining objective inherently optimizes the downstream tasks, as indicated by the first-order approximation in Eq. (2). This gradient closeness translates into lower downstream losses and better benchmark performance.

Empirical Landscapes Align with Fig. 2. As shown in Appendix B.6, Nexus reduces the downstream loss $\mathcal{L}_{\mathcal{T}}(\boldsymbol{\theta}^*_{\text{train}})$ by decreasing the geometric distance $\|\boldsymbol{\theta}^*_{\mathcal{T}} - \boldsymbol{\theta}^*_{\text{train}}\|_2$ to the task-specific minimizer. This observation matches the analyses in Theorem 2 and 6. While Nexus reduces this distance, it does not cause all minima to nearly intersect as depicted in Sec. 1—which would theoretically yield a nearly 0% OOD error. Instead, it achieves a moderate reduction, leading to a proportionally lower downstream loss. We hope future work can design stronger Nexus capable of approaching this extreme closeness without significant computational overhead.

B.6. Implicit Biases of Other Optimizers

Motivation. To explicitly demonstrate the "same pretraining loss, better downstream task", we visualize the correlation between pretraining and downstream losses, using the results of AdamW, Muon, and AdamW-Nexus from Appendix B.3. We plot the averaged downstream loss (y-axis) against the pretraining validation loss (x-axis) at corresponding checkpoints. The results are presented in Appendix B.6.

Muon does not possess a superior implicit bias. As illustrated in Appendix B.6, the curves for Adam and pure Muon almost completely overlap. This indicates that despite its orthogonalization mechanism, Muon seems not to introduce a favorable implicit bias for downstream generalization beyond what is explained by the pretraining loss itself, aligns with the findings in Wen et al. [77].

Implicit bias does not stem from gradient normalization. To further isolate the source of Nexus's generalization benefits, we conduct an ablation study where the normalized gradient $\boldsymbol{g}/\|\boldsymbol{g}\|_2$ is directly fed into the Adam optimizer instead of the raw gradient $\boldsymbol{g}$. The results are shown as the NSGD curve in Appendix B.6. This variant still fails to introduce any favorable implicit bias, which closely overlaps with the standard Adam baseline.

Appendix C. Additional Discussions

C.1. Closeness Improves Out-of-Distribution Generalization

Eq. (1) indicates that minimizing $\|\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\theta}^*_{\mathcal{T}}\|_2^2$ directly boosts downstream generalization; however, it would also achieve a lower pretraining loss if the intrinsic task losses $\mathcal{L}_k(\boldsymbol{\theta}^*_k)$ and flatness remain unchanged. In this section, we show that *even at the same pretraining loss* $\mathcal{L}_{\text{train}}$, a "close" minimizer (Sec. 1) yields better downstream generalization compared to a "distant" minimizer (Sec. 1).

The core intuition is that as long as the loss landscape is locally and directionally strongly convex along the directions of interest, improved closeness inherently lowers the generalization gap due to reduced task variance (see Theorem 6). We substantiate this intuition under a simplified quadratic setting in the main text:

Theorem 2 (Generalization of Closeness in the Quadratic Case) *Assume the pretraining tasks* $\{\mathcal{L}_k\}_{k=1}^K$ *and the downstream task* $\mathcal{L}_{\mathcal{T}}$ *are i.i.d. sampled from a task distribution* $\mathcal{P}$*. Locally, assume each task is a quadratic function* $\mathcal{L}(\boldsymbol{\theta}) = \frac{a}{2}\|\boldsymbol{\theta} - \boldsymbol{\theta}^*_{\mathcal{L}}\|_2^2 + c$*, where the task-specific minimizers have a spatial variance of* $\sigma^2 = \mathbb{E}[\|\boldsymbol{\theta}^*_{\mathcal{L}} - \boldsymbol{\mu}\|_2^2]$.

For any converged parameter $\boldsymbol{\theta}^*_{train}$ *that achieves a fixed total pretraining loss* $\mathcal{L}_{train}(\boldsymbol{\theta}^*_{train}) = C_{train}$*, the expected downstream loss on an unseen task* $\mathcal{T} \sim \mathcal{P}$ *is strictly decoupled into the pretraining loss and a variance-dependent gap:*

$$\mathbb{E}_{\mathcal{T}\sim\mathcal{P}}[\mathcal{L}_{\mathcal{T}}(\boldsymbol{\theta}^*_{train})] = C_{train} + \frac{a}{K}\sigma^2. \tag{4}$$

Therefore, explicitly optimizing the closeness σ^2 to reduce the task variance may be beneficial for downstream performance.

C.2. Convergence Rate of Nexus

All of our analyses are based on the assumption that Nexus should not be slower than its base optimizer. This ensures that both can achieve the "same training loss," allowing the implicit bias of Nexus to subsequently achieve "better downstream performance." One might concern that since Nexus optimizes two joint objectives (see Theorem 5 and 7), it may be slower than its base optimizer. Consequently, the downstream gains might not offset the speed loss, potentially leading to worse overall downstream performance.

Fortunately, this concern does not hold in practice. Empirically, across all experiments, Nexus is not slower, and sometimes even slightly faster, than its base optimizer (see Appendix B). Intuitively, Nexus makes the gradients of each $\mathcal{L}_i$ similar; thus, optimizing $\mathcal{L}_i$ effectively optimizes $\mathcal{L}_j$ simultaneously, as analyzed in Sec. 3.2. This "constructive interference" can lead to slightly faster convergence.

We can also adopt the framework of Johnson and Zhang [29] (assuming each $\mathcal{L}_i$ is L-smooth and μ-strongly convex) to obtain further theoretical intuition. In this setting, standard SGD typically achieves only an $O(1/T)$ convergence rate. However, if Nexus succeeds in finding a region where these tasks share common minimizers, it can achieve exponential convergence:

Theorem 3 *Suppose each* $\mathcal{L}_i$ *is* L*-smooth and* μ*-strongly convex. That is, for any* $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2$*, we have:*

$$\begin{aligned}
\mathcal{L}_i(\boldsymbol{\theta}_1) &\le \mathcal{L}_i(\boldsymbol{\theta}_2) + \nabla\mathcal{L}_i(\boldsymbol{\theta}_2)^\top(\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2) + \frac{L}{2}\|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|_2^2, \\
\mathcal{L}_i(\boldsymbol{\theta}_1) &\ge \mathcal{L}_i(\boldsymbol{\theta}_2) + \nabla\mathcal{L}_i(\boldsymbol{\theta}_2)^\top(\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2) + \frac{\mu}{2}\|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|_2^2.
\end{aligned} \tag{5}$$

Additionally, assume there exists a common minimizer $\boldsymbol{\theta}^*$ such that $\nabla \mathcal{L}_i(\boldsymbol{\theta}^*) = 0$ for all $i \in [K]$. Then, for the sequence $\{\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_T\}$ generated by Nexus with step size $\gamma \in (0, \frac{2}{L+\mu})$, we have:

$$\mathbb{E}[\|\boldsymbol{\theta}_T - \boldsymbol{\theta}^*\|^2] \leq \left(1 - \frac{2\gamma\mu L}{L+\mu}\right)^T \|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*\|^2. \tag{6}$$

Specifically, setting $\gamma = \frac{2}{L+\mu}$ and defining the condition number $\kappa = L/\mu$, we obtain the convergence rate:

$$\mathbb{E}[\|\boldsymbol{\theta}_T - \boldsymbol{\theta}^*\|^2] \leq \left(\frac{\kappa - 1}{\kappa + 1}\right)^{2T} \|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*\|^2. \tag{7}$$

Therefore, if Nexus guides the parameters into a locally convex and smooth regime where a common minimizer exists, it guarantees exponential convergence.

C.3. Implicit Bias of Normalized SGD

Interestingly, Nexus also offers a novel perspective on the success of Normalized SGD (NSGD). We observe that NSGD can be mathematically interpreted as a special case of Nexus, revealing that NSGD does not merely minimize the scalar loss but also implicitly optimizes gradient closeness. This implicit regularization provides a geometric explanation for why NSGD often generalizes better than standard Gradient Descent.

Theorem 4 (Implicit Bias of NSGD) *Let $\{\boldsymbol{\theta}_t\}$ be the sequence generated by Normalized SGD with learning rate γ. This sequence implicitly minimizes the following expected joint objective:*

$$\mathcal{J}_{NSGD}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[\mathcal{L}(\boldsymbol{x};\boldsymbol{\theta})] - \frac{\gamma}{8}\mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'\overset{i.i.d}{\sim}\mathcal{D}}[CosSim(\nabla\mathcal{L}(\boldsymbol{x};\boldsymbol{\theta}), \nabla\mathcal{L}(\boldsymbol{x}';\boldsymbol{\theta}))], \tag{8}$$

subject to a discretization error bounded by $\frac{4}{3}\left(\frac{4L^2+\rho G_{\min}}{G_{\min}^2}\right)\gamma^3$.

Proof A sequence of n updates of NSGD is algebraically equivalent to performing Nexus with k inner steps (using NSGD) and n/k outer steps (using SGD with step size 1). By Theorem 7, the magnitude of the gradient alignment signal scales as $S(k) \approx \gamma^2 \frac{k(k-1)}{4}$, while the residual error is bounded by $N(k) \leq \frac{1}{6}\left(\frac{4L^2+\rho G_{\min}}{G_{\min}^2}\right)k^3\gamma^3$. Defining the signal-to-noise ratio as $\rho(k) \triangleq S(k)/N(k)$ and maximizing it with respect to k yields $k = 2$. Thus, viewing the NSGD updates through the lens of Nexus with $k = 2$ yields the stated results.

■

C.4. Other Approximators for Hessian Gradient Product

While the Hessian-vector product can theoretically be implemented via the Jacobian-vector product (JVP) in PyTorch [55] with only a constant factor of computational overhead, implementing exact Hessian-gradient products in practical LLM pretraining remains prohibitive. First, standard Hessian-vector product implementations are often incompatible with memory-efficient kernels like FlashAttention [15, 16] (which typically do not support second-order differentiation efficiently), leading to significantly higher memory usage and computational costs. Second, the constant margin of memory overhead poses significant infrastructure challenges for large-scale distributed training.

Algorithm 1: Standard Nexus Algorithm

Require: Initial params $\boldsymbol{\theta}_0$, losses $\{\mathcal{L}_i\}_{i=1}^K$, total iterations T.
Require: Optimizers: $\text{Opt}_{\text{inner}}$ (Normalized SGD), $\text{Opt}_{\text{outer}}$ (e.g., AdamW). Inner learning rate γ.

for $t = 1$ **to** T **do**
  $\boldsymbol{\theta}_{t,0} \leftarrow \boldsymbol{\theta}_{t-1}$ {Initialize inner loop}
  for $m = 1$ **to** K **do**
    Sample task index $s_m \sim \text{Uniform}(\{1, \dots, K\})$
    $\boldsymbol{g} \leftarrow \nabla \mathcal{L}_{s_m}(\boldsymbol{\theta}_{t,m-1})$
    $\boldsymbol{\theta}_{t,m} \leftarrow \boldsymbol{\theta}_{t,m-1} - \gamma \cdot \frac{\boldsymbol{g}}{\|\boldsymbol{g}\|_2}$ {Update inner trajectory}
  end for
  $\hat{\boldsymbol{g}}_t \leftarrow \boldsymbol{\theta}_{t,0} - \boldsymbol{\theta}_{t,K}$ {Compute Nexus pseudo-gradient}
  $\boldsymbol{\theta}_t \leftarrow \text{Opt}_{\text{outer}}(\boldsymbol{\theta}_{t-1}, \hat{\boldsymbol{g}}_t)$ {Outer-update}
end forReturn: $\boldsymbol{\theta}_T$

Moreover, the Nexus algorithm exhibits a beneficial *third-order effect*. It actively seeks regions where gradients are not only aligned but also locally flat along the gradient dimension. This ensures that the gradient alignment property remains stable across a larger regime.

Theorem 5 (Nexus Maximizes Stability of Closeness) *Assume the existence of constants $G_{\min}, L, \rho > 0$ as in Theorem 7, and let M_3 be a constant such that $\nabla^3 \mathcal{L}_i(\boldsymbol{\theta})[\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}] \le M_3$ for any unit vectors $\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}$. Then, the sequence $\{\boldsymbol{\theta}_t\}$ generated by Algorithm 1 effectively minimizes the following third-order objective:*

$$\mathcal{J}_{3rd}(\boldsymbol{\theta}) = \mathcal{J}_{2nd}(\boldsymbol{\theta}) + \gamma^3 \frac{(K-1)(2K-1)}{12K^2} \sum_{i,j,p} \nabla \mathcal{L}_i(\boldsymbol{\theta})^\top \nabla^2 \mathcal{L}_j(\boldsymbol{\theta}) \nabla \mathcal{L}_p(\boldsymbol{\theta}). \tag{9}$$

The approximation error is bounded by:

$$\boldsymbol{\mathcal{E}}_{3rd} \triangleq \|\mathbb{E}[\hat{\boldsymbol{g}}_t] - \nabla \mathcal{J}_{3rd}\| \le \left(\frac{M_3}{24} + \frac{M_3 L}{8 G_{\min}} \right) K^4 \gamma^4 + \frac{M_3 L^2}{40 G_{\min}^2} K^5 \gamma^5 = O(\gamma^4). \tag{10}$$

Therefore, the third-order effect of Nexus works like a kind of "Multi-Task SAM": it minimize the directional sharpness along different tasks, leading to flatter landscape.

C.5. Extension to Generalized Similarity Metrics

In the main text, we primarily instantiate the Nexus algorithm using Normalized SGD (NSGD) in the inner loop, which implicitly maximizes the cosine similarity between task gradients. However, the theoretical framework of Nexus is highly adaptable and can be naturally extended to optimize a broader class of generalized similarity metrics.

Consider an arbitrary gradient transformation function $\Phi : \mathbb{R}^d \to \mathbb{R}^d$. We can define a generalized gradient similarity metric between two tasks as the inner product of their transformed gradients:

$$\text{Sim}_\Phi(\nabla \mathcal{L}_i(\boldsymbol{\theta}), \nabla \mathcal{L}_j(\boldsymbol{\theta})) \triangleq \langle \Phi(\nabla \mathcal{L}_i(\boldsymbol{\theta})), \Phi(\nabla \mathcal{L}_j(\boldsymbol{\theta})) \rangle. \tag{11}$$

To explicitly maximize this generalized similarity, we simply replace the normalized gradient in the Nexus inner loop with the generalized transformation Φ. The inner update rule becomes:

$$\boldsymbol{\theta}_{t,m} = \boldsymbol{\theta}_{t,m-1} - \gamma \Phi(\nabla \mathcal{L}_{s_m}(\boldsymbol{\theta}_{t,m-1})). \tag{12}$$

Following the exact same Taylor expansion logic derived in Eq. (3), when we evaluate the transformed gradient of task j at the displaced parameter state caused by task i, we obtain:

$$\Phi(\nabla\mathcal{L}_j(\boldsymbol{\theta}_{t,1})) - \Phi(\nabla\mathcal{L}_j(\boldsymbol{\theta}_{t,0})) = \left(\nabla_{\boldsymbol{\theta}_{t,0}}\Phi(\nabla\mathcal{L}_j(\boldsymbol{\theta}_{t,0}))\right)(\boldsymbol{\theta}_{t,1} - \boldsymbol{\theta}_{t,0}) + \mathcal{O}(\gamma^2) \tag{13}$$

$$= -\gamma\left(\nabla_{\boldsymbol{\theta}_{t,0}}\Phi(\nabla\mathcal{L}_j(\boldsymbol{\theta}_{t,0}))\right)\Phi(\nabla\mathcal{L}_i(\boldsymbol{\theta}_{t,0})) + \mathcal{O}(\gamma^2) \tag{14}$$

$$\overset{\mathbb{E}}{=} -\frac{\gamma}{2}\nabla_{\boldsymbol{\theta}}\text{Sim}_\Phi(\nabla\mathcal{L}_i(\boldsymbol{\theta}_{t,0}), \nabla\mathcal{L}_j(\boldsymbol{\theta}_{t,0})) + \mathcal{O}(\gamma^2). \tag{15}$$

This derivation demonstrates that for *any* choice of the transformation function Φ, the sequential evaluation mechanism of the Nexus inner loop inherently constructs a Jacobian-vector product that aligns with the gradient of the generalized similarity metric Sim_Φ.

Consequently, the generalized Nexus algorithm guarantees convergence towards a region where Sim_Φ is maximized (in the first-order and expectation sense). This provides a unified theoretical framework:

- When $\Phi(\nabla\mathcal{L}) = \nabla\mathcal{L}$, Nexus recovers the maximization of the unnormalized dot product (as empirically analyzed in the Nexus-Dot ablation).
- When $\Phi(\nabla\mathcal{L}) = \boldsymbol{A}\nabla\mathcal{L}$ (where $\boldsymbol{A}$ is a full-rank matrix), Sim_Φ induces a valid Mahalanobis inner product, evaluating geometric consensus within a strictly defined Hilbert space. For non-linear Φ, Sim_Φ acts as a kernel function, implicitly mapping gradients to seek consensus within a Reproducing Kernel Hilbert Space (RKHS).
- When $\Phi(\nabla\mathcal{L}) = \frac{\nabla\mathcal{L}}{\|\nabla\mathcal{L}\|_2}$, Nexus recovers the standard cosine similarity maximization.
- Other choices of Φ (e.g., $\Phi(\nabla\mathcal{L}) = \text{sign}(\nabla\mathcal{L})$ or orthogonalized gradient in Muon [30]) could potentially be explored to enforce different geometric consensus biases across tasks, expanding the toolkit for implicit regularization in other settings.

C.6. Cosine Similarity Instead of Dot Product Similarity

Although as discussed in Sec. 3, the dot product similarity of gradients offers a more direct theoretical connection—yielding a tighter bound for parameter closeness (Theorem 1) and a more straightforward interpretation for downstream generalization (Eq. (2))—it proves practically challenging to optimize.

This difficulty primarily arises because the dot product objective introduces a pathological optimization shortcut. Specifically, the dot product is highly scale-dependent: if the overall loss magnitude scales by a factor of k, the gradient norm scales proportionally by k, causing the dot product similarity to artificially inflate by a factor of k^2. Consequently, directly maximizing the dot product severely disrupts the primary minimization of the pretraining loss, as the optimizer may exploit this shortcut by inadvertently increasing the gradient norms rather than discovering genuine task consensus.

As demonstrated in Appendix B.6, the optimization trajectory of Nexus-Dot lags significantly behind the standard Adam baseline. The resulting degradation in pretraining loss heavily outweighs any potential generalization benefits conferred by its implicit bias. Therefore, we adopt cosine similarity (via normalized gradients) as our primary regularization objective in this work. Note that the progressive deceleration of Nexus-Dot observed in Appendix B.6 is a persistent geometric

Algorithm 2: Generalized Nexus Algorithm

Require: Initial params $\boldsymbol{\theta}_0$, losses $\{\mathcal{L}_i\}_{i=1}^{K}$, total iterations T.
Require: Gradient transformation function $\Phi : \mathbb{R}^d \to \mathbb{R}^d$.
Require: Optimizers: $\text{Opt}_{\text{inner}}$ (Memory-free update via Φ), $\text{Opt}_{\text{outer}}$ (e.g., AdamW). Inner learning rate γ.
Implicit Objective: Maximize expected generalized similarity $\mathbb{E}_{i,j}[\langle \Phi(\nabla \mathcal{L}_i(\boldsymbol{\theta})), \Phi(\nabla \mathcal{L}_j(\boldsymbol{\theta}))\rangle]$.

for $t = 1$ **to** T **do**
  $\boldsymbol{\theta}_{t,0} \leftarrow \boldsymbol{\theta}_{t-1}$ {Initialize inner loop}
  for $m = 1$ **to** K **do**
    Sample task index $s_m \sim \text{Uniform}(\{1, \ldots, K\})$
    $\boldsymbol{g} \leftarrow \nabla \mathcal{L}_{s_m}(\boldsymbol{\theta}_{t,m-1})$
    $\boldsymbol{\theta}_{t,m} \leftarrow \boldsymbol{\theta}_{t,m-1} - \gamma \cdot \Phi(\boldsymbol{g})$ {Update inner trajectory with Φ}
  end for
  $\hat{\boldsymbol{g}}_t \leftarrow \boldsymbol{\theta}_{t,0} - \boldsymbol{\theta}_{t,K}$ {Compute generalized pseudo-gradient}
  $\boldsymbol{\theta}_t \leftarrow \text{Opt}_{\text{outer}}(\boldsymbol{\theta}_{t-1}, \hat{\boldsymbol{g}}_t)$ {Outer update}
end for
Return: $\boldsymbol{\theta}_T$

phenomenon, occurring consistently regardless of the learning rate scheduler or the choice of base optimizer (e.g., AdamW or Muon). Due to space constraints, we selectively present the ablation results for the 3B model with Adam, corresponding to the main setup in Appendix B.3.

Appendix D. Proofs for Closeness Improving Generalization

D.1. Proof for Theorem 2

Proof First, solving the stationarity condition $\nabla \sum \mathcal{L}_k(\boldsymbol{\theta}) = \mathbf{0}$, we obtain the closed-form solution for the converged parameter: $\boldsymbol{\theta}^*_{\text{train}} = \frac{1}{K}\sum_{k=1}^{K} \boldsymbol{\theta}^*_k$.

The training loss at this optimum is given by:

$$\mathcal{L}_{\text{train}}(\boldsymbol{\theta}^*_{\text{train}}) = \frac{1}{K}\sum_{k=1}^{K}\left(\frac{a}{2}\|\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\theta}^*_k\|_2^2 + c\right) = C_{\text{train}}. \tag{16}$$

From this, we can express the intrinsic loss constant c (which represents the "depth" of the minima) in terms of the fixed training loss C_{train}:

$$c = C_{\text{train}} - \frac{a}{2K}\sum_{k=1}^{K}\|\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\theta}^*_k\|_2^2. \tag{17}$$

Now, consider the loss on a new downstream task $\mathcal{T}$ with minimizer $\boldsymbol{\theta}^*_{\mathcal{T}} \sim \mathcal{P}$:

$$\mathcal{L}_{\mathcal{T}}(\boldsymbol{\theta}^*_{\text{train}}) = \frac{a}{2}\|\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\theta}^*_{\mathcal{T}}\|_2^2 + c. \tag{18}$$

Substituting c, the generalization gap becomes:

$$\mathcal{L}_{\mathcal{T}}(\boldsymbol{\theta}^*_{\text{train}}) - C_{\text{train}} = \frac{a}{2}\left(\|\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\theta}^*_{\mathcal{T}}\|_2^2 - \frac{1}{K}\sum_{k=1}^{K}\|\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\theta}^*_k\|_2^2\right). \tag{19}$$

Taking the expectation over the task distribution $\mathcal{P}$, and utilizing the property of variance for i.i.d. samples (where $\mathbb{E}[\|\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\theta}^*_{\mathcal{T}}\|^2] = (1+\frac{1}{K})\sigma^2$ and $\mathbb{E}[\frac{1}{K}\sum\|\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\theta}^*_k\|^2] = \frac{K-1}{K}\sigma^2$):

$$\mathbb{E}[\mathcal{L}_{\mathcal{T}}(\boldsymbol{\theta}^*_{\text{train}})] - C_{\text{train}} = \frac{a}{2}\left(\left(1+\frac{1}{K}\right)\sigma^2 - \frac{K-1}{K}\sigma^2\right) = \frac{a}{K}\sigma^2. \tag{20}$$

This concludes the proof. It explicitly shows that for a fixed training loss budget C_{train}, the generalization error scales linearly with the task variance σ^2. ■

D.2. Proof for Theorem 6

Theorem 6 (Generalization of Closeness beyond Quadratics, Proof in Appendix D.2) *Let $\boldsymbol{\theta}^*$ be a specific local minimizer of the population loss $\mathbb{E}_{\mathcal{L}\sim\mathcal{P}}[\mathcal{L}(\boldsymbol{\theta})]$. For any task $\mathcal{L}$ sampled from $\mathcal{P}$, let $\boldsymbol{\theta}^*_{\mathcal{L}} = \arg\min_{\boldsymbol{\theta}\in\mathcal{S}_{\mathcal{L}}}\|\boldsymbol{\theta}^* - \boldsymbol{\theta}\|_2$ be its corresponding local minimizer. Assume that for any task $\mathcal{L}\sim\mathcal{P}$, the loss function is locally and directionally strongly convex along the segments $[\boldsymbol{\theta}^*_{\mathcal{L}}, \boldsymbol{\theta}^*]$, i.e., $\lambda_{\max} \geq \boldsymbol{u}^\top\nabla^2\mathcal{L}(\boldsymbol{\xi})\boldsymbol{u} \geq \lambda_{\min} > 0$ for any $\boldsymbol{\xi}\in[\boldsymbol{\theta}^*_{\mathcal{L}}, \boldsymbol{\theta}^*]$ and any unit vector $\boldsymbol{u}\in \text{span}\{\boldsymbol{\theta}^* - \boldsymbol{\theta}^*_{\mathcal{L}} \mid \mathcal{L}\sim\mathcal{P}\}$. Let $\boldsymbol{\mu} = \mathbb{E}[\boldsymbol{\theta}^*_{\mathcal{L}}]$ and $\sigma^2 = \mathbb{E}[\|\boldsymbol{\theta}^*_{\mathcal{L}} - \boldsymbol{\mu}\|_2^2]$. Assuming the statistical independence between the task flatness $\nabla^2\mathcal{L}_{\mathcal{L}}(\boldsymbol{\xi})$ and the task closeness $\boldsymbol{\theta}^*_{\mathcal{L}}$ across the distribution $\mathcal{P}$. Conditioned on achieving a fixed training loss C_{train}, the expected out-of-distribution generalization error of the converged training parameter $\boldsymbol{\theta}^*_{train}$ is bounded by:*

$$\mathbb{E}_{\mathcal{T}\sim\mathcal{P}}[\mathcal{L}_{\mathcal{T}}(\boldsymbol{\theta}^*_{train})] - C_{train} \leq \frac{\lambda_{\max}\left(\left(\frac{\lambda_{\max}}{\lambda_{\min}}\right)^2 + 1\right)}{2K}\sigma^2. \tag{21}$$

We now generalize the previous result to the general case. Assume that the pretraining tasks $\{\mathcal{L}_k\}_{k=1}^K$ and the downstream task $\mathcal{L}_\mathcal{T}$ are sampled independently from a latent task distribution $\mathcal{P}$.

Due to the over-parameterized nature of LLMs, the minimizers are not unique. To rigorously analyze the closeness, we first define the set of local minimizers for the *expected population loss*:

$$\mathcal{S}_\mathcal{P} = \left\{ \boldsymbol{\vartheta} \mid \exists \epsilon > 0, \forall \boldsymbol{\vartheta}' \in B_\epsilon(\boldsymbol{\vartheta}), \mathbb{E}_{\mathcal{T}\sim\mathcal{P}}[\mathcal{L}_\mathcal{T}(\boldsymbol{\vartheta})] \leq \mathbb{E}_{\mathcal{T}\sim\mathcal{P}}[\mathcal{L}_\mathcal{T}(\boldsymbol{\vartheta}')] \right\}. \tag{22}$$

Let $\boldsymbol{\theta}^* \in \mathcal{S}_\mathcal{P}$ be one specific local minimizer of the population loss. This serves as the anchor point for the basin of attraction.

We then define the task-specific minimizer $\boldsymbol{\theta}_k^*$ as the projection of this population minimizer $\boldsymbol{\theta}^*$ onto the set of local minimizers of task k:

$$\boldsymbol{\theta}_k^* = \arg\min_{\boldsymbol{\vartheta}\in\mathcal{S}_k} \|\boldsymbol{\vartheta} - \boldsymbol{\theta}^*\|_2, \quad \text{where } \mathcal{S}_k \text{ denotes the set of local minimizers of } \mathcal{L}_k. \tag{23}$$

Given the distribution of these task-specific minimizers $\{\boldsymbol{\theta}_k^*\}$, we define their statistical center $\boldsymbol{\mu}$ and intrinsic covariance $\boldsymbol{\Sigma}$ as:

$$\boldsymbol{\mu} := \mathbb{E}_{\mathcal{T}\sim\mathcal{P}}[\boldsymbol{\theta}_\mathcal{T}^*], \quad \boldsymbol{\Sigma} := \mathbb{E}[(\boldsymbol{\theta}_\mathcal{T}^* - \boldsymbol{\mu})(\boldsymbol{\theta}_\mathcal{T}^* - \boldsymbol{\mu})^\top]. \tag{24}$$

We also define the scalar intrinsic variance $\sigma^2 = \text{Tr}(\boldsymbol{\Sigma}) = \mathbb{E}[\|\boldsymbol{\theta}_k^* - \boldsymbol{\mu}\|_2^2]$. From this point forward, our analysis focuses on the closeness to the statistical center $\boldsymbol{\mu}$, as $\mathbb{E}[\boldsymbol{\theta}_\mathcal{T}^* - \boldsymbol{\mu}] = \mathbf{0}$ holds by definition.

Step 1: Estimation Error. The converged parameter $\boldsymbol{\theta}_{\text{train}}^*$ satisfies the stationarity condition:

$$\nabla\mathcal{L}_{\text{train}}(\boldsymbol{\theta}_{\text{train}}^*) = 0 \iff \sum_{k=1}^K \nabla\mathcal{L}_k(\boldsymbol{\theta}_{\text{train}}^*) = 0. \tag{25}$$

Applying the Mean Value Theorem, there exists $\boldsymbol{\xi}_k \in [\boldsymbol{\theta}_{\text{train}}^*, \boldsymbol{\theta}_k^*]$ such that $\nabla\mathcal{L}_k(\boldsymbol{\theta}_{\text{train}}^*) = \nabla^2\mathcal{L}_k(\boldsymbol{\xi}_k)(\boldsymbol{\theta}_{\text{train}}^* - \boldsymbol{\theta}_k^*)$. Thus:

$$\sum_{k=1}^K \nabla^2\mathcal{L}_k(\boldsymbol{\xi}_k)(\boldsymbol{\theta}_{\text{train}}^* - \boldsymbol{\mu}) = \sum_{k=1}^K \nabla^2\mathcal{L}_k(\boldsymbol{\xi}_k)(\boldsymbol{\theta}_k^* - \boldsymbol{\mu}). \tag{26}$$

We assume the local curvature is bounded: for any k and vector $\boldsymbol{u}$, $\lambda_{\min}\|\boldsymbol{u}\|^2 \leq \boldsymbol{u}^\top\nabla^2\mathcal{L}_k(\boldsymbol{\xi}_k)\boldsymbol{u} \leq \lambda_{\max}\|\boldsymbol{u}\|^2$. Bounding the estimation error norm:

$$\|\boldsymbol{\theta}_{\text{train}}^* - \boldsymbol{\mu}\|_2 \leq \frac{1}{K\lambda_{\min}} \sum_{k=1}^K \lambda_{\max}\|\boldsymbol{\theta}_k^* - \boldsymbol{\mu}\|_2. \tag{27}$$

Taking the expectation (noting cross-terms vanish because $\mathbb{E}[\boldsymbol{\theta}_k^* - \boldsymbol{\mu}] = \mathbf{0}$) and defining $\kappa = \lambda_{\max}/\lambda_{\min}$:

$$\mathbb{E}[\|\boldsymbol{\theta}_{\text{train}}^* - \boldsymbol{\mu}\|_2^2] \leq \frac{\kappa^2}{K}\sigma^2. \tag{28}$$

Step 2: The Intrinsic Loss Trade-off. We condition on the training loss achieving a fixed value C_{train}. By exact Taylor expansion around the task minimizers, the training loss is:

$$C_{\text{train}} = \frac{1}{K}\sum_{k=1}^{K}\mathcal{L}_k(\boldsymbol{\theta}^*_{\text{train}}) = \frac{1}{K}\sum_{k=1}^{K}\left(\mathcal{L}_k(\boldsymbol{\theta}^*_k) + \frac{1}{2}(\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\theta}^*_k)^\top \nabla^2\mathcal{L}_k(\boldsymbol{\xi}_k)(\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\theta}^*_k)\right). \quad (29)$$

Taking the expectation over the task distribution, we can express the expected intrinsic loss exactly as:

$$\mathbb{E}[\mathcal{L}_k(\boldsymbol{\theta}^*_k)] = C_{\text{train}} - \underbrace{\frac{1}{2}\mathbb{E}\left[\frac{1}{K}\sum_{k=1}^{K}(\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\theta}^*_k)^\top \nabla^2\mathcal{L}_k(\boldsymbol{\xi}_k)(\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\theta}^*_k)\right]}_{\mathcal{Q}_{\text{train}}\text{ (Expected Empirical Closeness Penalty)}}. \quad (30)$$

We retain the term $\mathcal{Q}_{\text{train}}$ explicitly without approximation. This term represents the curvature-weighted variance of the minimizers around the converged point.

Step 3: Downstream Generalization (Rigorous Matrix Derivation). Finally, we analyze the expected performance on a downstream task $\mathcal{T}$ sampled from the same distribution $\mathcal{P}$. We perform a Taylor expansion of the test loss around the task-specific minimizer $\boldsymbol{\theta}^*_{\mathcal{T}}$. Since $\nabla\mathcal{L}_{\mathcal{T}}(\boldsymbol{\theta}^*_{\mathcal{T}}) = 0$, the first-order term vanishes:

$$\mathcal{L}_{\mathcal{T}}(\boldsymbol{\theta}^*_{\text{train}}) = \mathcal{L}_{\mathcal{T}}(\boldsymbol{\theta}^*_{\mathcal{T}}) + \frac{1}{2}(\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\theta}^*_{\mathcal{T}})^\top \nabla^2\mathcal{L}_{\mathcal{T}}(\boldsymbol{\xi}_{\mathcal{T}})(\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\theta}^*_{\mathcal{T}}). \quad (31)$$

Taking the expectation over the task distribution, we define the expected test closeness penalty $\mathcal{Q}_{\text{test}}$:

$$\mathbb{E}_{\mathcal{T}}[\mathcal{L}_{\mathcal{T}}(\boldsymbol{\theta}^*_{\text{train}})] = \mathbb{E}[\mathcal{L}_{\mathcal{T}}(\boldsymbol{\theta}^*_{\mathcal{T}})] + \underbrace{\frac{1}{2}\mathbb{E}\left[(\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\theta}^*_{\mathcal{T}})^\top \nabla^2\mathcal{L}_{\mathcal{T}}(\boldsymbol{\xi}_{\mathcal{T}})(\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\theta}^*_{\mathcal{T}})\right]}_{\mathcal{Q}_{\text{test}}}. \quad (32)$$

Recalling the intrinsic loss trade-off from Eq. (30), we have $\mathbb{E}[\mathcal{L}_{\mathcal{T}}(\boldsymbol{\theta}^*_{\mathcal{T}})] = C_{\text{train}} - \mathcal{Q}_{\text{train}}$. Substituting this into the equation above yields the generalization gap decomposition:

$$\mathbb{E}_{\mathcal{T}}[\mathcal{L}_{\mathcal{T}}(\boldsymbol{\theta}^*_{\text{train}})] = C_{\text{train}} + (\mathcal{Q}_{\text{test}} - \mathcal{Q}_{\text{train}}). \quad (33)$$

Let $\bar{\mathbf{H}} = \mathbb{E}_{\mathcal{P}}[\nabla^2\mathcal{L}(\boldsymbol{\xi})]$ denote the expected Hessian matrix over the task distribution. Since tasks are i.i.d., both training and test tasks share this expected geometry.

For the test term $\mathcal{Q}_{\text{test}}$, we use the identity $\boldsymbol{x}^\top\mathbf{A}\boldsymbol{x} = \text{Tr}(\mathbf{A}\boldsymbol{x}\boldsymbol{x}^\top)$. Replacing the specific task Hessian with the expected Hessian $\bar{\mathbf{H}} = \mathbb{E}_{\mathcal{P}}[\nabla^2\mathcal{L}(\boldsymbol{\xi})]$:

$$\mathcal{Q}_{\text{test}} = \frac{1}{2}\text{Tr}\left(\bar{\mathbf{H}} \cdot \mathbb{E}\left[(\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\theta}^*_{\mathcal{T}})(\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\theta}^*_{\mathcal{T}})^\top\right]\right). \quad (34)$$

We expand the covariance term fully around the statistical center $\boldsymbol{\mu}$:

$$\begin{aligned}\mathbb{E}\left[(\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\theta}^*_{\mathcal{T}})(\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\theta}^*_{\mathcal{T}})^\top\right] &= \mathbb{E}\left[((\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\mu}) - (\boldsymbol{\theta}^*_{\mathcal{T}} - \boldsymbol{\mu}))((\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\mu}) - (\boldsymbol{\theta}^*_{\mathcal{T}} - \boldsymbol{\mu}))^\top\right] \\ &= \mathbb{E}[(\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\mu})(\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\mu})^\top] + \mathbb{E}[(\boldsymbol{\theta}^*_{\mathcal{T}} - \boldsymbol{\mu})(\boldsymbol{\theta}^*_{\mathcal{T}} - \boldsymbol{\mu})^\top] \\ &\quad - \mathbb{E}[(\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\mu})(\boldsymbol{\theta}^*_{\mathcal{T}} - \boldsymbol{\mu})^\top] - \mathbb{E}[(\boldsymbol{\theta}^*_{\mathcal{T}} - \boldsymbol{\mu})(\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\mu})^\top].\end{aligned} \quad (35)$$

The cross-terms vanish strictly because $\boldsymbol{\theta}^*_{\mathcal{T}}$ is independent of $\boldsymbol{\theta}^*_{\text{train}}$ and is centered at $\boldsymbol{\mu}$ (i.e., $\mathbb{E}[\boldsymbol{\theta}^*_{\mathcal{T}} - \boldsymbol{\mu}] = \mathbf{0}$ by definition of $\boldsymbol{\mu}$). Substituting $\mathbb{E}[(\boldsymbol{\theta}^*_{\mathcal{T}} - \boldsymbol{\mu})(\boldsymbol{\theta}^*_{\mathcal{T}} - \boldsymbol{\mu})^\top] = \boldsymbol{\Sigma}$ back:

$$\mathcal{Q}_{\text{test}} = \frac{1}{2}\text{Tr}\left(\bar{\mathbf{H}} \cdot \mathbb{E}[(\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\mu})(\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\mu})^\top]\right) + \frac{1}{2}\text{Tr}(\bar{\mathbf{H}}\boldsymbol{\Sigma}). \tag{36}$$

For the training term $\mathcal{Q}_{\text{train}}$, we consider the expected quadratic penalty averaged over the training tasks. By linearity of expectation, we replace $\nabla^2\mathcal{L}_k$ with $\bar{\mathbf{H}}$ exactly:

$$\mathcal{Q}_{\text{train}} = \frac{1}{2K}\sum_{k=1}^{K}\mathbb{E}\left[(\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\theta}^*_k)^\top\bar{\mathbf{H}}(\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\theta}^*_k)\right]. \tag{37}$$

We apply the Generalized Centroid Property. For any positive semi-definite matrix $\bar{\mathbf{H}}$, the weighted sum of squared errors is minimized by the mean $\bar{\boldsymbol{\theta}}$. Thus, we have the rigorous lower bound:

$$\sum_{k=1}^{K}(\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\theta}^*_k)^\top\bar{\mathbf{H}}(\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\theta}^*_k) \geq \sum_{k=1}^{K}(\bar{\boldsymbol{\theta}} - \boldsymbol{\theta}^*_k)^\top\bar{\mathbf{H}}(\bar{\boldsymbol{\theta}} - \boldsymbol{\theta}^*_k). \tag{38}$$

We perform the matrix variance decomposition on the RHS by inserting $\boldsymbol{\mu}$:

$$\begin{aligned}
&\sum_{k=1}^{K}(\bar{\boldsymbol{\theta}} - \boldsymbol{\theta}^*_k)^\top\bar{\mathbf{H}}(\bar{\boldsymbol{\theta}} - \boldsymbol{\theta}^*_k) \\
&= \sum_{k=1}^{K}((\bar{\boldsymbol{\theta}} - \boldsymbol{\mu}) - (\boldsymbol{\theta}^*_k - \boldsymbol{\mu}))^\top\bar{\mathbf{H}}((\bar{\boldsymbol{\theta}} - \boldsymbol{\mu}) - (\boldsymbol{\theta}^*_k - \boldsymbol{\mu})) \\
&= \sum_{k=1}^{K}(\bar{\boldsymbol{\theta}} - \boldsymbol{\mu})^\top\bar{\mathbf{H}}(\bar{\boldsymbol{\theta}} - \boldsymbol{\mu}) + \sum_{k=1}^{K}(\boldsymbol{\theta}^*_k - \boldsymbol{\mu})^\top\bar{\mathbf{H}}(\boldsymbol{\theta}^*_k - \boldsymbol{\mu}) - 2(\bar{\boldsymbol{\theta}} - \boldsymbol{\mu})^\top\bar{\mathbf{H}}\underbrace{\sum_{k=1}^{K}(\boldsymbol{\theta}^*_k - \boldsymbol{\mu})}_{K(\bar{\boldsymbol{\theta}} - \boldsymbol{\mu})}.
\end{aligned} \tag{39}$$

Simplifying the cross-term and combining with the first term:

$$\begin{aligned}
&\sum_{k=1}^{K}(\bar{\boldsymbol{\theta}} - \boldsymbol{\theta}^*_k)^\top\bar{\mathbf{H}}(\bar{\boldsymbol{\theta}} - \boldsymbol{\theta}^*_k) \\
&= K(\bar{\boldsymbol{\theta}} - \boldsymbol{\mu})^\top\bar{\mathbf{H}}(\bar{\boldsymbol{\theta}} - \boldsymbol{\mu}) + \sum_{k=1}^{K}(\boldsymbol{\theta}^*_k - \boldsymbol{\mu})^\top\bar{\mathbf{H}}(\boldsymbol{\theta}^*_k - \boldsymbol{\mu}) - 2K(\bar{\boldsymbol{\theta}} - \boldsymbol{\mu})^\top\bar{\mathbf{H}}(\bar{\boldsymbol{\theta}} - \boldsymbol{\mu}) \\
&= \sum_{k=1}^{K}(\boldsymbol{\theta}^*_k - \boldsymbol{\mu})^\top\bar{\mathbf{H}}(\boldsymbol{\theta}^*_k - \boldsymbol{\mu}) - K(\bar{\boldsymbol{\theta}} - \boldsymbol{\mu})^\top\bar{\mathbf{H}}(\bar{\boldsymbol{\theta}} - \boldsymbol{\mu}).
\end{aligned} \tag{40}$$

Taking expectations and using the trace identity $\mathbb{E}[\boldsymbol{x}^\top\mathbf{A}\boldsymbol{x}] = \text{Tr}(\mathbf{A}\mathbb{E}[\boldsymbol{x}\boldsymbol{x}^\top])$:

- The first term: $\sum_{k=1}^{K}\text{Tr}(\bar{\mathbf{H}}\mathbb{E}[(\boldsymbol{\theta}^*_k - \boldsymbol{\mu})(\boldsymbol{\theta}^*_k - \boldsymbol{\mu})^\top]) = K\text{Tr}(\bar{\mathbf{H}}\boldsymbol{\Sigma})$.
- The second term (variance of the mean): $\mathbb{E}[(\bar{\boldsymbol{\theta}} - \boldsymbol{\mu})(\bar{\boldsymbol{\theta}} - \boldsymbol{\mu})^\top] = \frac{1}{K}\boldsymbol{\Sigma}$. Thus, $K\text{Tr}(\bar{\mathbf{H}} \cdot \frac{1}{K}\boldsymbol{\Sigma}) = \text{Tr}(\bar{\mathbf{H}}\boldsymbol{\Sigma})$.

Combining these, the expected training penalty is bounded by:

$$\mathcal{Q}_{\text{train}} \geq \frac{1}{2K}\left(K\text{Tr}(\bar{\mathbf{H}}\boldsymbol{\Sigma}) - \text{Tr}(\bar{\mathbf{H}}\boldsymbol{\Sigma})\right) = \frac{1}{2}\left(1 - \frac{1}{K}\right)\text{Tr}(\bar{\mathbf{H}}\boldsymbol{\Sigma}). \tag{41}$$

Subtracting the two terms ($\mathcal{Q}_{\text{test}} - \mathcal{Q}_{\text{train}}$), the dominant term $\frac{1}{2}\text{Tr}(\bar{\mathbf{H}}\boldsymbol{\Sigma})$ cancels out exactly. We then bound the remaining terms using the spectral norm $\lambda_{\max}$ and the estimation error bound derived in Eq. (28):

$$\begin{aligned}
\mathbb{E}_{\mathcal{T}}[\mathcal{L}_{\mathcal{T}}(\boldsymbol{\theta}^*_{\text{train}})] - C_{\text{train}} &\leq \left(\frac{1}{2}\text{Tr}(\bar{\mathbf{H}}\mathbb{E}[(\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\mu})(\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\mu})^\top]) + \frac{1}{2}\text{Tr}(\bar{\mathbf{H}}\boldsymbol{\Sigma})\right) - \frac{1}{2}\left(1 - \frac{1}{K}\right)\text{Tr}(\bar{\mathbf{H}}\boldsymbol{\Sigma}) \\
&= \frac{1}{2}\text{Tr}\left(\bar{\mathbf{H}} \cdot \mathbb{E}[(\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\mu})(\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\mu})^\top]\right) + \frac{1}{2K}\text{Tr}(\bar{\mathbf{H}}\boldsymbol{\Sigma}) \\
&\leq \frac{\lambda_{\max}}{2}\mathbb{E}[\|\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\mu}\|_2^2] + \frac{\lambda_{\max}}{2K}\text{Tr}(\boldsymbol{\Sigma}) \\
&\leq \frac{\lambda_{\max}}{2}\left(\frac{\kappa^2}{K}\sigma^2\right) + \frac{\lambda_{\max}}{2K}\sigma^2 \\
&= \frac{\lambda_{\max}(\kappa^2 + 1)}{2K}\sigma^2.
\end{aligned} \tag{42}$$

This confirms that the generalization gap scales with $O(\frac{\sigma^2}{K})$, driven by the intrinsic task variance and the number of pretraining tasks.

Appendix E. Proof of Theorem 1

In this section, we provide the detailed proof for Theorem 1, which bounds the closeness between minimizers using gradient similarity.

Proof The proof proceeds in three main steps: (1) relating the closeness to the gradient norm via the Mean Value Theorem; (2) exploiting the stationarity condition of the total loss to decompose the gradient norms; and (3) bounding the cross-terms using the gradient upper bound and cosine similarity.

Step 1: Relating Closeness to Gradient Norm. Recall that $\boldsymbol{\theta}_k^*$ is the projection of $\boldsymbol{\theta}$ onto the global optimal set $\mathcal{S}_k$. Since $\boldsymbol{\theta}_k^*$ is a minimizer, we have $\nabla\mathcal{L}_k(\boldsymbol{\theta}_k^*) = \mathbf{0}$. Applying the Mean Value Theorem to the vector-valued function $\boldsymbol{\vartheta} \mapsto \nabla\mathcal{L}_k(\boldsymbol{\vartheta})$, there exists a point $\boldsymbol{\xi}_k$ on the line segment connecting $\boldsymbol{\theta}_k^*$ and $\boldsymbol{\theta}$ such that:

$$\nabla\mathcal{L}_k(\boldsymbol{\theta}) - \nabla\mathcal{L}_k(\boldsymbol{\theta}_k^*) = \nabla^2\mathcal{L}_k(\boldsymbol{\xi}_k)(\boldsymbol{\theta} - \boldsymbol{\theta}_k^*). \tag{43}$$

Substituting $\nabla\mathcal{L}_k(\boldsymbol{\theta}_k^*) = \mathbf{0}$ and taking the norm:

$$\|\nabla\mathcal{L}_k(\boldsymbol{\theta})\|_2 = \|\nabla^2\mathcal{L}_k(\boldsymbol{\xi}_k)(\boldsymbol{\theta} - \boldsymbol{\theta}_k^*)\|_2. \tag{44}$$

We assume the curvature condition where the smallest eigenvalue of the Hessian along the displacement vector is bounded below by $\lambda > 0$. Specifically:

$$\boldsymbol{u}_k^\top \nabla^2\mathcal{L}_k(\boldsymbol{\xi}_k)\boldsymbol{u}_k \geq \lambda, \quad \text{where } \boldsymbol{u}_k = \frac{\boldsymbol{\theta} - \boldsymbol{\theta}_k^*}{\|\boldsymbol{\theta} - \boldsymbol{\theta}_k^*\|_2}. \tag{45}$$

This implies $\|\nabla^2\mathcal{L}_k(\boldsymbol{\xi}_k)(\boldsymbol{\theta} - \boldsymbol{\theta}_k^*)\|_2 \geq \lambda\|\boldsymbol{\theta} - \boldsymbol{\theta}_k^*\|_2$. Rearranging this inequality gives an upper bound on the closeness:

$$\|\boldsymbol{\theta} - \boldsymbol{\theta}_k^*\|_2 \leq \frac{1}{\lambda}\|\nabla\mathcal{L}_k(\boldsymbol{\theta})\|_2. \tag{46}$$

Squaring and averaging over all K tasks yields:

$$\frac{1}{K}\sum_{k=1}^{K}\|\boldsymbol{\theta} - \boldsymbol{\theta}_k^*\|_2^2 \leq \frac{1}{K\lambda^2}\sum_{k=1}^{K}\|\nabla\mathcal{L}_k(\boldsymbol{\theta})\|_2^2. \tag{47}$$

Step 2: Force Balance Decomposition. Since $\boldsymbol{\theta}$ is the converged parameter for the total loss, it satisfies the stationarity condition:

$$\sum_{k=1}^{K}\nabla\mathcal{L}_k(\boldsymbol{\theta}) = \mathbf{0}. \tag{48}$$

We analyze the squared norm of this sum, which must equal zero:

$$\left\|\sum_{k=1}^{K}\nabla\mathcal{L}_k(\boldsymbol{\theta})\right\|_2^2 = \sum_{k=1}^{K}\|\nabla\mathcal{L}_k(\boldsymbol{\theta})\|_2^2 + \sum_{i\neq j}\nabla\mathcal{L}_i(\boldsymbol{\theta})^\top\nabla\mathcal{L}_j(\boldsymbol{\theta}) = 0. \tag{49}$$

By rearranging terms, we obtain an exact identity relating the sum of squared gradient norms to the negative sum of cross-task inner products:

$$\sum_{k=1}^{K}\|\nabla\mathcal{L}_k(\boldsymbol{\theta})\|_2^2 = \sum_{i\neq j}\left(-\nabla\mathcal{L}_i(\boldsymbol{\theta})^\top\nabla\mathcal{L}_j(\boldsymbol{\theta})\right). \tag{50}$$

Substituting Eq. (50) into Eq. (47), we obtain the first inequality of the theorem:

$$\frac{1}{K}\sum_{k=1}^{K}\|\boldsymbol{\theta}-\boldsymbol{\theta}_k^*\|_2^2 \leq \frac{1}{K\lambda^2}\sum_{i\neq j}\left(-\nabla\mathcal{L}_i(\boldsymbol{\theta})^\top\nabla\mathcal{L}_j(\boldsymbol{\theta})\right). \tag{51}$$

Step 3: Bounding via Cosine Similarity. Finally, we bound the inner product term using the gradient magnitude upper bound $G = \sup_k \|\nabla\mathcal{L}_k(\boldsymbol{\theta})\|_2$. Recall that:

$$\nabla\mathcal{L}_i(\boldsymbol{\theta})^\top\nabla\mathcal{L}_j(\boldsymbol{\theta}) = \|\nabla\mathcal{L}_i(\boldsymbol{\theta})\|_2\|\nabla\mathcal{L}_j(\boldsymbol{\theta})\|_2\text{CosSim}(\nabla\mathcal{L}_i(\boldsymbol{\theta}),\nabla\mathcal{L}_j(\boldsymbol{\theta})). \tag{52}$$

We use the property that for any i, j, the following term is non-negative:

$$(G^2 - \|\nabla\mathcal{L}_i(\boldsymbol{\theta})\|_2\|\nabla\mathcal{L}_j(\boldsymbol{\theta})\|_2)(1-\text{CosSim}(\nabla\mathcal{L}_i(\boldsymbol{\theta}),\nabla\mathcal{L}_j(\boldsymbol{\theta}))) \geq 0, \tag{53}$$

since $\|\nabla\mathcal{L}_k(\boldsymbol{\theta})\|_2 \leq G$ and CosSim ≤ 1. Adding this non-negative term to the negative inner product allows us to derive the bound directly:

$$\begin{aligned}
-\nabla\mathcal{L}_i(\boldsymbol{\theta})^\top\nabla\mathcal{L}_j(\boldsymbol{\theta}) &= -\|\nabla\mathcal{L}_i(\boldsymbol{\theta})\|_2\|\nabla\mathcal{L}_j(\boldsymbol{\theta})\|_2\text{CosSim}(\nabla\mathcal{L}_i(\boldsymbol{\theta}),\nabla\mathcal{L}_j(\boldsymbol{\theta})) \\
&\leq -\|\nabla\mathcal{L}_i(\boldsymbol{\theta})\|_2\|\nabla\mathcal{L}_j(\boldsymbol{\theta})\|_2\text{CosSim}(\nabla\mathcal{L}_i(\boldsymbol{\theta}),\nabla\mathcal{L}_j(\boldsymbol{\theta})) \\
&\quad + (G^2 - \|\nabla\mathcal{L}_i(\boldsymbol{\theta})\|_2\|\nabla\mathcal{L}_j(\boldsymbol{\theta})\|_2)(1-\text{CosSim}(\nabla\mathcal{L}_i(\boldsymbol{\theta}),\nabla\mathcal{L}_j(\boldsymbol{\theta}))) \\
&= G^2(1-\text{CosSim}(\nabla\mathcal{L}_i(\boldsymbol{\theta}),\nabla\mathcal{L}_j(\boldsymbol{\theta}))) - \|\nabla\mathcal{L}_i(\boldsymbol{\theta})\|_2\|\nabla\mathcal{L}_j(\boldsymbol{\theta})\|_2 \\
&\leq G^2(1-\text{CosSim}(\nabla\mathcal{L}_i(\boldsymbol{\theta}),\nabla\mathcal{L}_j(\boldsymbol{\theta}))).
\end{aligned} \tag{54}$$

Summing this inequality over all $i \neq j$ yields:

$$\sum_{i\neq j}\left(-\nabla\mathcal{L}_i(\boldsymbol{\theta})^\top\nabla\mathcal{L}_j(\boldsymbol{\theta})\right) \leq G^2\sum_{i\neq j}(1-\text{CosSim}(\nabla\mathcal{L}_i(\boldsymbol{\theta}),\nabla\mathcal{L}_j(\boldsymbol{\theta}))). \tag{55}$$

Combining this with the result from Step 2 completes the proof. ■

Appendix F. Implicit Bias of Nexus Optimizer

Theorem 7 (Nexus Maximizes Gradient Similarity) *Assume there exist constants $G_{\min}, L, \rho > 0$ such that for any $t \in [1, T]$ and $m \in [1, K]$:*

$$\|\nabla \mathcal{L}_i(\boldsymbol{\theta}_{t,m})\|_2 \geq G_{\min}; \quad \|\nabla^2 \mathcal{L}_i(\boldsymbol{\theta})\|_2 \leq L; \quad \|\nabla^2 \mathcal{L}_i(\boldsymbol{x}) - \nabla^2 \mathcal{L}_i(\boldsymbol{y})\|_2 \leq \rho \|\boldsymbol{x} - \boldsymbol{y}\|_2. \tag{56}$$

Then, the sequence $\{\boldsymbol{\theta}_t\}$ generated by Algorithm 1 minimizes the following second-order objective:

$$\mathcal{J}_{2nd}(\boldsymbol{\theta}) = \gamma \sum_{i=1}^{K} \mathcal{L}_i(\boldsymbol{\theta}) - \gamma^2 \frac{K-1}{4K} \sum_{i \neq j} \mathit{CosSim}\Big(\nabla \mathcal{L}_i(\boldsymbol{\theta}), \nabla \mathcal{L}_j(\boldsymbol{\theta})\Big). \tag{57}$$

This holds because the expected update direction satisfies:

$$\mathbb{E}[\hat{\boldsymbol{g}}_t] = \gamma \sum_{i=1}^{K} \frac{\nabla \mathcal{L}_i(\boldsymbol{\theta}_t)}{\|\nabla \mathcal{L}_i(\boldsymbol{\theta}_t)\|_2} - \gamma^2 \frac{K-1}{4K} \Big(\nabla_{\boldsymbol{\theta}} \sum_{i \neq j} \mathit{CosSim}(\nabla \mathcal{L}_i, \nabla \mathcal{L}_j) + \boldsymbol{\mathcal{E}}_{sym,i,j} \Big) + \boldsymbol{\mathcal{E}}_{2nd}, \tag{58}$$

where $\boldsymbol{\mathcal{E}}_{sym,i,j} = \left(\nabla \frac{\nabla \mathcal{L}_i}{\|\nabla \mathcal{L}_i\|_2} - \left(\nabla \frac{\nabla \mathcal{L}_i}{\|\nabla \mathcal{L}_i\|_2} \right)^\top \right) \frac{\nabla \mathcal{L}_j}{\|\nabla \mathcal{L}_j\|_2} \perp \nabla \mathcal{L}_j$ is the non-symmetric error, which becomes zero when the gradients align with the top eigenvectors of their respective Hessians [12, 14, 22, 65]. The Taylor approximation error is bounded by $\|\boldsymbol{\mathcal{E}}_{2nd}\|_2 \leq \frac{1}{6} \left(\frac{4L^2 + \rho G_{\min}}{G_{\min}^2} \right) K^3 \gamma^3 = \mathcal{O}(\gamma^3)$.

In this appendix, we provide the detailed proofs for Theorem 7. We rigorously analyze the update dynamics of Algorithm 1 using second-order Taylor expansions and derive the precise form of the implicit optimization objective with explicit non-asymptotic error bounds.

F.1. Preliminaries and Notation

Let $\mathcal{L}_i : \mathbb{R}^d \to \mathbb{R}$ denote the loss function for the i-th task, where $i \in \{1, \ldots, k\}$. We denote the gradient and Hessian at parameters $\boldsymbol{\theta}$ as $\nabla \mathcal{L}_i(\boldsymbol{\theta})$ and $\nabla^2 \mathcal{L}_i(\boldsymbol{\theta})$, respectively. The cosine similarity between the gradients of task i and task j is defined as:

$$S_{ij}(\boldsymbol{\theta}) \triangleq \mathrm{CosSim}(\nabla \mathcal{L}_i(\boldsymbol{\theta}), \nabla \mathcal{L}_j(\boldsymbol{\theta})) = \frac{\nabla \mathcal{L}_i(\boldsymbol{\theta})^\top \nabla \mathcal{L}_j(\boldsymbol{\theta})}{\|\nabla \mathcal{L}_i(\boldsymbol{\theta})\|_2 \|\nabla \mathcal{L}_j(\boldsymbol{\theta})\|_2}. \tag{59}$$

Algorithm 1 performs k inner updates in each outer iteration t. Let $\boldsymbol{\theta}_{t,0}$ be the parameters at the start of the inner loop (i.e., $\boldsymbol{\theta}_{t,0} = \boldsymbol{\theta}_{t-1}$). At each inner step $m \in \{1, \ldots, k\}$, a task index s_m is sampled uniformly from $\{1, \ldots, k\}$. The update rule is:

$$\boldsymbol{\theta}_{t,m} = \boldsymbol{\theta}_{t,m-1} - \gamma \frac{\nabla \mathcal{L}_{s_m}(\boldsymbol{\theta}_{t,m-1})}{\|\nabla \mathcal{L}_{s_m}(\boldsymbol{\theta}_{t,m-1})\|_2}. \tag{60}$$

The Nexus pseudo-gradient passed to the outer optimizer is $\hat{\boldsymbol{g}}_t = \boldsymbol{\theta}_{t,0} - \boldsymbol{\theta}_{t,k} = \sum_{m=1}^{k} (\boldsymbol{\theta}_{t,m-1} - \boldsymbol{\theta}_{t,m})$.

F.2. Assumptions and Derived Constants

To derive explicit non-asymptotic bounds, we utilize the following standard assumptions regarding the loss landscape.

- **Assumption 1 (Bounded Gradients):** For all tasks i and parameters $\boldsymbol{\theta}$, the gradient norm is bounded from below: $0 < G_{\min} \leq \|\nabla \mathcal{L}_i(\boldsymbol{\theta})\|_2$.
- **Assumption 2 (Smoothness):** The loss $\mathcal{L}_i$ is L-smooth, i.e., $\|\nabla^2 \mathcal{L}_i(\boldsymbol{\theta})\|_2 \leq L$.
- **Assumption 3 (Hessian Lipschitz):** The Hessian is ρ-Lipschitz continuous, i.e., $\|\nabla^2 \mathcal{L}_i(\boldsymbol{x}) - \nabla^2 \mathcal{L}_i(\boldsymbol{y})\|_2 \leq \rho \|\boldsymbol{x} - \boldsymbol{y}\|_2$.

Based on the properties above, we further denote L_1 and L_2 as the Lipschitz constants for the normalized gradient and its Jacobian, respectively:

1. The normalized gradient is L_1-Lipschitz continuous:

$$\left\| \frac{\nabla \mathcal{L}_i(\boldsymbol{x})}{\|\nabla \mathcal{L}_i(\boldsymbol{x})\|_2} - \frac{\nabla \mathcal{L}_i(\boldsymbol{y})}{\|\nabla \mathcal{L}_i(\boldsymbol{y})\|_2} \right\|_2 \leq L_1 \|\boldsymbol{x} - \boldsymbol{y}\|_2. \tag{61}$$

2. The Jacobian of the normalized gradient is L_2-Lipschitz continuous:

$$\|\boldsymbol{J}_i(\boldsymbol{x}) - \boldsymbol{J}_i(\boldsymbol{y})\|_2 \leq L_2 \|\boldsymbol{x} - \boldsymbol{y}\|_2, \tag{62}$$

   where $\boldsymbol{J}_i(\boldsymbol{\theta}) = \frac{\partial}{\partial \boldsymbol{\theta}} \left(\frac{\nabla \mathcal{L}_i(\boldsymbol{\theta})}{\|\nabla \mathcal{L}_i(\boldsymbol{\theta})\|_2} \right)$.

Derivation of Constants. Here, we provide the detailed derivation of L_1 and L_2 based on Assumptions 1-3.

1. Derivation of L_1: By the Mean Value Theorem, L_1 is bounded by the supremum of the spectral norm of the Jacobian $\boldsymbol{J}_i(\boldsymbol{\theta})$. The Jacobian is explicitly given by:

$$\boldsymbol{J}_i(\boldsymbol{\theta}) = \frac{1}{\|\nabla \mathcal{L}_i\|_2} \left(\boldsymbol{I} - \frac{\nabla \mathcal{L}_i \nabla \mathcal{L}_i^\top}{\|\nabla \mathcal{L}_i\|_2^2} \right) \nabla^2 \mathcal{L}_i(\boldsymbol{\theta}). \tag{63}$$

The middle term is an orthogonal projection matrix with spectral norm 1. Using the bounds from Assumptions 1 and 2:

$$L_1 \leq \sup_{\boldsymbol{\theta}} \|\boldsymbol{J}_i(\boldsymbol{\theta})\|_2 \leq \frac{1}{G_{\min}} \cdot 1 \cdot L = \frac{L}{G_{\min}}. \tag{64}$$

2. Derivation of L_2: We decompose the Jacobian $\boldsymbol{J}_i(\boldsymbol{\theta})$ into three components: a scalar term $u(\boldsymbol{\theta})$, a projection term $\boldsymbol{\Pi}(\boldsymbol{\theta})$, and the Hessian $\boldsymbol{H}_i(\boldsymbol{\theta})$:

$$\boldsymbol{J}_i(\boldsymbol{\theta}) = \underbrace{\|\nabla \mathcal{L}_i(\boldsymbol{\theta})\|_2^{-1}}_{u(\boldsymbol{\theta})} \cdot \underbrace{\left(\boldsymbol{I} - \frac{\nabla \mathcal{L}_i \nabla \mathcal{L}_i^\top}{\|\nabla \mathcal{L}_i\|_2^2} \right)}_{\boldsymbol{\Pi}(\boldsymbol{\theta})} \cdot \underbrace{\nabla^2 \mathcal{L}_i(\boldsymbol{\theta})}_{\boldsymbol{H}_i(\boldsymbol{\theta})}. \tag{65}$$

We apply the product Lipschitz rule. For a product of three functions $f = abc$, the Lipschitz constant satisfies $L_f \leq L_a M_b M_c + M_a L_b M_c + M_a M_b L_c$, where $M_{(\cdot)}$ denotes the upper bound of the magnitude and $L_{(\cdot)}$ denotes the Lipschitz constant.

- **Part 1: Scalar $u(\boldsymbol{\theta}) = \|\nabla \mathcal{L}_i\|_2^{-1}$.**
  Magnitude (M_u): By Assumption 1, $|u| \leq \frac{1}{G_{\min}}$.
  Lipschitz (L_u): The gradient of u is $\nabla u = -\|\nabla \mathcal{L}_i\|_2^{-2} \frac{\nabla^2 \mathcal{L}_i \nabla \mathcal{L}_i}{\|\nabla \mathcal{L}_i\|_2} = -\frac{\boldsymbol{H}_i \nabla \mathcal{L}_i}{\|\nabla \mathcal{L}_i\|_2^3}$. Taking the norm, we have $\|\nabla u\|_2 \leq \frac{\|\boldsymbol{H}_i\|_2 \|\nabla \mathcal{L}_i\|_2}{\|\nabla \mathcal{L}_i\|_2^3} = \frac{\|\boldsymbol{H}_i\|_2}{\|\nabla \mathcal{L}_i\|_2^2}$. Using the bounds L and $G_{\min}$, we get $L_u = \frac{L}{G_{\min}^2}$.
- **Part 2: Projection $\boldsymbol{\Pi}(\boldsymbol{\theta}) = \boldsymbol{I} - \boldsymbol{h}_i \boldsymbol{h}_i^\top$.**
  Magnitude (M_Π): The spectral norm is $\|\boldsymbol{\Pi}\|_2 = 1$.
  Lipschitz (L_Π): $\boldsymbol{\Pi}$ depends on the normalized gradient $\boldsymbol{h}_i$, which is L_1-Lipschitz. For any unit vectors $\boldsymbol{x}, \boldsymbol{y}$, we have $\|\boldsymbol{x}\boldsymbol{x}^\top - \boldsymbol{y}\boldsymbol{y}^\top\|_2 \leq \|\boldsymbol{x}(\boldsymbol{x}-\boldsymbol{y})^\top\|_2 + \|(\boldsymbol{x}-\boldsymbol{y})\boldsymbol{y}^\top\|_2 = 2\|\boldsymbol{x}-\boldsymbol{y}\|_2$. By the chain rule, $L_\Pi = 2L_1 = \frac{2L}{G_{\min}}$.
- **Part 3: Hessian $\boldsymbol{H}_i(\boldsymbol{\theta}) = \nabla^2 \mathcal{L}_i$.**
  Magnitude (M_H): By Assumption 2, $\|\boldsymbol{H}_i\|_2 \leq L$.
  Lipschitz (L_H): By Assumption 3, $L_H = \rho$.

Substituting these values into the product rule formula:

$$\begin{aligned} L_2 &\leq L_u M_\Pi M_H + M_u L_\Pi M_H + M_u M_\Pi L_H \\ &\leq \left(\frac{L}{G_{\min}^2} \cdot 1 \cdot L\right) + \left(\frac{1}{G_{\min}} \cdot \frac{2L}{G_{\min}} \cdot L\right) + \left(\frac{1}{G_{\min}} \cdot 1 \cdot \rho\right) \\ &= \frac{L^2}{G_{\min}^2} + \frac{2L^2}{G_{\min}^2} + \frac{\rho}{G_{\min}}. \end{aligned} \tag{66}$$

Combining terms yields the final constant:

$$L_2 = \frac{3L^2 + \rho G_{\min}}{G_{\min}^2}. \tag{67}$$

F.3. Derivation of the Update Direction

We now derive the expansion of the total pseudo-gradient $\hat{\boldsymbol{g}}_t$ and bound the error terms.

F.3.1. STEP 1: EXPANSION OF THE NORMALIZED GRADIENT

We aim to expand the normalized gradient at the shifted parameters $\boldsymbol{\theta}_{t,m-1}$ around the initial point $\boldsymbol{\theta}_{t,0}$. Let $\Delta\boldsymbol{\theta}_{m-1} = \boldsymbol{\theta}_{t,m-1} - \boldsymbol{\theta}_{t,0}$.

The Jacobian of the normalized gradient is given explicitly by the projection of the Hessian:

$$\boldsymbol{J}_{s_m}(\boldsymbol{\theta}) = \frac{1}{\|\nabla \mathcal{L}_{s_m}(\boldsymbol{\theta})\|_2} \left(\boldsymbol{I} - \frac{\nabla \mathcal{L}_{s_m}(\boldsymbol{\theta}) \nabla \mathcal{L}_{s_m}(\boldsymbol{\theta})^\top}{\|\nabla \mathcal{L}_{s_m}(\boldsymbol{\theta})\|_2^2}\right) \nabla^2 \mathcal{L}_{s_m}(\boldsymbol{\theta}). \tag{68}$$

Applying Taylor's theorem with the Lagrange remainder form:

$$\frac{\nabla \mathcal{L}_{s_m}(\boldsymbol{\theta}_{t,m-1})}{\|\nabla \mathcal{L}_{s_m}(\boldsymbol{\theta}_{t,m-1})\|_2} = \frac{\nabla \mathcal{L}_{s_m}(\boldsymbol{\theta}_{t,0})}{\|\nabla \mathcal{L}_{s_m}(\boldsymbol{\theta}_{t,0})\|_2} + \boldsymbol{J}_{s_m}(\boldsymbol{\theta}_{t,0}) \Delta\boldsymbol{\theta}_{m-1} + \boldsymbol{r}_m. \tag{69}$$

Using the L_2-Lipschitz property of the Jacobian, the residual vector $\boldsymbol{r}_m$ is bounded by:

$$\|\boldsymbol{r}_m\|_2 \leq \frac{L_2}{2} \|\Delta\boldsymbol{\theta}_{m-1}\|_2^2. \tag{70}$$

F.3.2. STEP 2: RECURSIVE SUBSTITUTION

The displacement $\Delta\boldsymbol{\theta}_{m-1}$ is the sum of previous updates. Using the zeroth-order approximation:

$$\Delta\boldsymbol{\theta}_{m-1} = \sum_{l=1}^{m-1}(\boldsymbol{\theta}_{t,l} - \boldsymbol{\theta}_{t,l-1}) = -\gamma \sum_{l=1}^{m-1} \frac{\nabla\mathcal{L}_{s_l}(\boldsymbol{\theta}_{t,l-1})}{\|\nabla\mathcal{L}_{s_l}(\boldsymbol{\theta}_{t,l-1})\|_2}. \tag{71}$$

We approximate the terms in the sum using the zeroth-order expansion around $\boldsymbol{\theta}_{t,0}$. Using the L_1-Lipschitz property of the normalized gradient:

$$\left\| \frac{\nabla\mathcal{L}_{s_l}(\boldsymbol{\theta}_{t,l-1})}{\|\nabla\mathcal{L}_{s_l}(\boldsymbol{\theta}_{t,l-1})\|_2} - \frac{\nabla\mathcal{L}_{s_l}(\boldsymbol{\theta}_{t,0})}{\|\nabla\mathcal{L}_{s_l}(\boldsymbol{\theta}_{t,0})\|_2} \right\|_2 \leq L_1\|\boldsymbol{\theta}_{t,l-1} - \boldsymbol{\theta}_{t,0}\|_2 = L_1 \left\| -\sum_{j=1}^{l-1} \gamma \frac{\nabla\mathcal{L}_{s_l}(\boldsymbol{\theta}_{t,j})}{\|\nabla\mathcal{L}_{s_l}(\boldsymbol{\theta}_{t,j})\|_2} \right\| \leq L_1(l-1)\gamma. \tag{72}$$

Thus, we can write:

$$\Delta\boldsymbol{\theta}_{m-1} = -\gamma \sum_{l=1}^{m-1} \frac{\nabla\mathcal{L}_{s_l}(\boldsymbol{\theta}_{t,0})}{\|\nabla\mathcal{L}_{s_l}(\boldsymbol{\theta}_{t,0})\|_2} + \boldsymbol{\delta}_{m-1}, \tag{73}$$

where the accumulated error $\boldsymbol{\delta}_{m-1}$ is bounded by summing the individual errors:

$$\|\boldsymbol{\delta}_{m-1}\|_2 \leq \gamma \sum_{l=1}^{m-1} L_1(l-1)\gamma = L_1\gamma^2 \frac{(m-1)(m-2)}{2} \leq \frac{L_1}{2}(m-1)^2\gamma^2. \tag{74}$$

Substituting this expression for $\Delta\boldsymbol{\theta}_{m-1}$ back into Eq. (69):

$$\frac{\nabla\mathcal{L}_{s_m}(\boldsymbol{\theta}_{t,m-1})}{\|\nabla\mathcal{L}_{s_m}(\boldsymbol{\theta}_{t,m-1})\|_2} = \frac{\nabla\mathcal{L}_{s_m}(\boldsymbol{\theta}_{t,0})}{\|\nabla\mathcal{L}_{s_m}(\boldsymbol{\theta}_{t,0})\|_2} - \gamma \sum_{l=1}^{m-1} \boldsymbol{J}_{s_m}(\boldsymbol{\theta}_{t,0}) \frac{\nabla\mathcal{L}_{s_l}(\boldsymbol{\theta}_{t,0})}{\|\nabla\mathcal{L}_{s_l}(\boldsymbol{\theta}_{t,0})\|_2} + \boldsymbol{\mathcal{E}}_m. \tag{75}$$

Here, the total error at step m, denoted $\boldsymbol{\mathcal{E}}_m$, consists of the Taylor residual $\boldsymbol{r}_m$ and the propagation error from $\boldsymbol{\delta}_{m-1}$ scaled by the Jacobian. Using $\|\boldsymbol{J}_{s_m}\|_2 \leq L_1$ and $\|\Delta\boldsymbol{\theta}_{m-1}\|_2 \leq (m-1)\gamma$:

$$\|\boldsymbol{\mathcal{E}}_m\|_2 \leq \frac{L_2}{2}(m-1)^2\gamma^2 + L_1\left(\frac{L_1}{2}(m-1)^2\gamma^2\right) = \frac{L_2 + L_1^2}{2}(m-1)^2\gamma^2. \tag{76}$$

F.3.3. STEP 3: AGGREGATION OF THE PSEUDO-GRADIENT

The total pseudo-gradient is $\hat{\boldsymbol{g}}_t = \gamma\sum_{m=1}^{k} \frac{\nabla\mathcal{L}_{sm}(\boldsymbol{\theta}_{t,m-1})}{\|\nabla\mathcal{L}_{sm}(\boldsymbol{\theta}_{t,m-1})\|_2}$. Substituting the result from Step 2:

$$\hat{\boldsymbol{g}}_t = \gamma \sum_{m=1}^{k} \frac{\nabla\mathcal{L}_{s_m}}{\|\nabla\mathcal{L}_{s_m}\|_2} - \gamma^2 \sum_{m=1}^{k}\sum_{l=1}^{m-1} \boldsymbol{J}_{s_m} \frac{\nabla\mathcal{L}_{s_l}}{\|\nabla\mathcal{L}_{s_l}\|_2} + \boldsymbol{\mathcal{E}}_{total}. \tag{77}$$

(We omit the argument $\boldsymbol{\theta}_{t,0}$ for brevity; all terms are evaluated at $\boldsymbol{\theta}_{t,0}$). The total error vector $\boldsymbol{\mathcal{E}}_{total} = \gamma\sum_{m=1}^{k} \boldsymbol{\mathcal{E}}_m$ is explicitly bounded by summing the bounds from Step 2:

$$\|\boldsymbol{\mathcal{E}}_{total}\|_2 \leq \gamma \sum_{m=1}^{k} \frac{L_2 + L_1^2}{2}(m-1)^2\gamma^2 \leq \frac{L_2 + L_1^2}{6}k^3\gamma^3. \tag{78}$$

F.3.4. STEP 4: EXPECTATION ANALYSIS AND CONNECTION TO COSINE SIMILARITY

We now compute the expectation of $\hat{\boldsymbol{g}}_t$ over the independent uniform sampling of indices $s_1, \ldots, s_k$ and relate the second-order term to the gradient of the cosine similarity.

Linear Term. Let $\mathcal{T}_{linear} = \sum_{m=1}^{k} \frac{\nabla \mathcal{L}_{s_m}}{\|\nabla \mathcal{L}_{s_m}\|_2}$. By linearity of expectation:

$$\mathbb{E}[\mathcal{T}_{linear}] = k \cdot \frac{1}{k} \sum_{i=1}^{k} \frac{\nabla \mathcal{L}_i}{\|\nabla \mathcal{L}_i\|_2} = \sum_{i=1}^{k} \frac{\nabla \mathcal{L}_i}{\|\nabla \mathcal{L}_i\|_2}. \tag{79}$$

Interaction Term. Let $\mathcal{T}_{interact} = \sum_{m=1}^{k} \sum_{l=1}^{m-1} \boldsymbol{J}_{s_m} \frac{\nabla \mathcal{L}_{s_l}}{\|\nabla \mathcal{L}_{s_l}\|_2}$. The double summation contains $\frac{k(k-1)}{2}$ terms. Since $m > l$, s_m and s_l are independent. Summing over all pairs yields:

$$\mathbb{E}[\mathcal{T}_{interact}] = \frac{k-1}{2k} \sum_{i,j} \boldsymbol{J}_i \frac{\nabla \mathcal{L}_j}{\|\nabla \mathcal{L}_j\|_2} = \frac{k-1}{4k} \sum_{i \neq j} \left(\boldsymbol{J}_i \frac{\nabla \mathcal{L}_j}{\|\nabla \mathcal{L}_j\|_2} + \boldsymbol{J}_j \frac{\nabla \mathcal{L}_i}{\|\nabla \mathcal{L}_i\|_2} \right). \tag{80}$$

Note that the exact gradient of the cosine similarity between task i and task j is formulated using the transposed Jacobian (VJP):

$$\nabla_{\boldsymbol{\theta}} \text{CosSim}(\nabla \mathcal{L}_i, \nabla \mathcal{L}_j) = \boldsymbol{J}_i^\top \frac{\nabla \mathcal{L}_j}{\|\nabla \mathcal{L}_j\|_2} + \boldsymbol{J}_j^\top \frac{\nabla \mathcal{L}_i}{\|\nabla \mathcal{L}_i\|_2}. \tag{81}$$

To relate the interaction term to the cosine similarity gradient, we add and subtract the transposed Jacobian terms:

$$\begin{aligned} &\boldsymbol{J}_i \frac{\nabla \mathcal{L}_j}{\|\nabla \mathcal{L}_j\|_2} + \boldsymbol{J}_j \frac{\nabla \mathcal{L}_i}{\|\nabla \mathcal{L}_i\|_2} \\ &= \left(\boldsymbol{J}_i^\top \frac{\nabla \mathcal{L}_j}{\|\nabla \mathcal{L}_j\|_2} + \boldsymbol{J}_j^\top \frac{\nabla \mathcal{L}_i}{\|\nabla \mathcal{L}_i\|_2} \right) + (\boldsymbol{J}_i - \boldsymbol{J}_i^\top) \frac{\nabla \mathcal{L}_j}{\|\nabla \mathcal{L}_j\|_2} + (\boldsymbol{J}_j - \boldsymbol{J}_j^\top) \frac{\nabla \mathcal{L}_i}{\|\nabla \mathcal{L}_i\|_2} \\ &= \nabla_{\boldsymbol{\theta}} \text{CosSim}(\nabla \mathcal{L}_i, \nabla \mathcal{L}_j) + \boldsymbol{\mathcal{E}}_{\text{sym},i,j} + \boldsymbol{\mathcal{E}}_{\text{sym},j,i}, \end{aligned} \tag{82}$$

where $\boldsymbol{\mathcal{E}}_{\text{sym},i,j}$ is the Jacobian non-symmetric error defined as:

$$\boldsymbol{\mathcal{E}}_{\text{sym},i,j} = \left(\nabla \frac{\nabla \mathcal{L}_i}{\|\nabla \mathcal{L}_i\|_2} - \left(\nabla \frac{\nabla \mathcal{L}_i}{\|\nabla \mathcal{L}_i\|_2} \right)^\top \right) \frac{\nabla \mathcal{L}_j}{\|\nabla \mathcal{L}_j\|_2}. \tag{83}$$

The residual $\boldsymbol{\mathcal{E}}_{\text{sym},i,j}$ exhibits two properties:

1. **Orthogonality to the target gradient:** The matrix $\boldsymbol{A}_i = \nabla(\frac{\nabla \mathcal{L}_i}{\|\nabla \mathcal{L}_i\|_2}) - \nabla(\frac{\nabla \mathcal{L}_i}{\|\nabla \mathcal{L}_i\|_2})^\top$ is anti-symmetric ($\boldsymbol{A}_i^\top = -\boldsymbol{A}_i$). For any anti-symmetric matrix $\boldsymbol{A}$ and vector $\boldsymbol{x}$, the quadratic form is zero ($\boldsymbol{x}^\top \boldsymbol{A} \boldsymbol{x} = 0$). By setting $\boldsymbol{x} = \frac{\nabla \mathcal{L}_j}{\|\nabla \mathcal{L}_j\|_2}$, we have $\langle \boldsymbol{\mathcal{E}}_{\text{sym},i,j}, \nabla \mathcal{L}_j \rangle = 0$, which implies $\boldsymbol{\mathcal{E}}_{\text{sym},i,j} \perp \nabla \mathcal{L}_j$.

2. **Vanishing under gradient-Hessian alignment:** Recent literature on the Edge of Stability (EoS) and the "river valley" phenomenon observes that gradients tend to align with the top eigenvectors of the Hessian during training [12, 14, 22, 65]. When the gradient aligns with an eigenvector, the commutator between the orthogonal projection and the Hessian becomes zero, which implies $\boldsymbol{\mathcal{E}}_{\text{sym},i,j} = \boldsymbol{0}$.

Substituting this expansion into the interaction term expectation yields:

$$\mathbb{E}[\mathcal{T}_{interact}] = \frac{k-1}{4k} \sum_{i \neq j} \left(\nabla_{\boldsymbol{\theta}} \text{CosSim}(\nabla \mathcal{L}_i, \nabla \mathcal{L}_j) + \boldsymbol{\mathcal{E}}_{\text{sym},i,j} + \boldsymbol{\mathcal{E}}_{\text{sym},j,i} \right). \tag{84}$$

F.4. Proof Conclusion

Combining the linear term and the interaction term, the expected Nexus update direction is:

$$\mathbb{E}[\hat{\boldsymbol{g}}_t] = \gamma \sum_{i=1}^{k} \frac{\nabla \mathcal{L}_i(\boldsymbol{\theta}_{t,0})}{\|\nabla \mathcal{L}_i(\boldsymbol{\theta}_{t,0})\|_2} - \gamma^2 \frac{k-1}{4k} \sum_{i \neq j} \left(\nabla_{\boldsymbol{\theta}} \text{CosSim}(\nabla \mathcal{L}_i, \nabla \mathcal{L}_j) + \boldsymbol{\mathcal{E}}_{\text{sym},i,j} + \boldsymbol{\mathcal{E}}_{\text{sym},j,i} \right) + \boldsymbol{\mathcal{E}}_{total}. \tag{85}$$

Substituting the constants derived in Appendix F.2, the residual is bounded by:

$$\|\boldsymbol{\mathcal{E}}_{total}\|_2 \leq \frac{1}{6} \left(\frac{\rho G_{\min} + 4L^2}{G_{\min}^2} \right) k^3 \gamma^3 = \mathcal{O}(\gamma^3). \tag{86}$$

This confirms that the update direction follows the gradient of the loss plus the similarity alignment term, subject to the non-symmetric residual and a bounded cubic Taylor error. □

Appendix G. Proof of Convergence Rate (Theorem 3)

In this section, we provide the detailed proof for Theorem 3. Let $\boldsymbol{\theta}^*$ be the common minimizer such that $\nabla\mathcal{L}_i(\boldsymbol{\theta}^*) = 0$ for all $i \in [k]$. Consider the update at step m: $\boldsymbol{\theta}_m = \boldsymbol{\theta}_{m-1} - \gamma\nabla\mathcal{L}_{s_m}(\boldsymbol{\theta}_{m-1})$, where s_m is the task index sampled uniformly at random.

First, we expand the squared distance to the optimum for a specific realization of s_m:

$$\begin{aligned}\|\boldsymbol{\theta}_m - \boldsymbol{\theta}^*\|^2 &= \|\boldsymbol{\theta}_{m-1} - \gamma\nabla\mathcal{L}_{s_m}(\boldsymbol{\theta}_{m-1}) - \boldsymbol{\theta}^*\|^2 \\ &= \|\boldsymbol{\theta}_{m-1} - \boldsymbol{\theta}^*\|^2 - 2\gamma\langle\nabla\mathcal{L}_{s_m}(\boldsymbol{\theta}_{m-1}), \boldsymbol{\theta}_{m-1} - \boldsymbol{\theta}^*\rangle + \gamma^2\|\nabla\mathcal{L}_{s_m}(\boldsymbol{\theta}_{m-1})\|^2.\end{aligned} \tag{87}$$

To bound the inner product term, we utilize the property of smooth and strongly convex functions. Define the auxiliary function $\phi_i(\boldsymbol{\theta}) = \mathcal{L}_i(\boldsymbol{\theta}) - \frac{\mu}{2}\|\boldsymbol{\theta}\|^2$. Since each $\mathcal{L}_i$ is L-smooth and μ-strongly convex, $\phi_i(\boldsymbol{\theta})$ is convex and $(L-\mu)$-smooth. By the co-coercivity property of convex smooth functions, for any $\boldsymbol{\theta}$, we have:

$$\langle\nabla\phi_i(\boldsymbol{\theta}) - \nabla\phi_i(\boldsymbol{\theta}^*), \boldsymbol{\theta} - \boldsymbol{\theta}^*\rangle \geq \frac{1}{L-\mu}\|\nabla\phi_i(\boldsymbol{\theta}) - \nabla\phi_i(\boldsymbol{\theta}^*)\|^2. \tag{88}$$

Substituting $\nabla\phi_i(\boldsymbol{\theta}) = \nabla\mathcal{L}_i(\boldsymbol{\theta}) - \mu\boldsymbol{\theta}$ and noting that $\nabla\mathcal{L}_i(\boldsymbol{\theta}^*) = 0$, we substitute back:

$$\begin{aligned}\langle\nabla\mathcal{L}_i(\boldsymbol{\theta}) - \mu(\boldsymbol{\theta} - \boldsymbol{\theta}^*), \boldsymbol{\theta} - \boldsymbol{\theta}^*\rangle &\geq \frac{1}{L-\mu}\|\nabla\mathcal{L}_i(\boldsymbol{\theta}) - \mu(\boldsymbol{\theta} - \boldsymbol{\theta}^*)\|^2 \\ &= \frac{1}{L-\mu}\left(\|\nabla\mathcal{L}_i(\boldsymbol{\theta})\|^2 - 2\mu\langle\nabla\mathcal{L}_i(\boldsymbol{\theta}), \boldsymbol{\theta} - \boldsymbol{\theta}^*\rangle + \mu^2\|\boldsymbol{\theta} - \boldsymbol{\theta}^*\|^2\right).\end{aligned} \tag{89}$$

Rearranging the terms, we obtain the following inequality which holds for any task index i, and thus specifically for the sampled index s_m:

$$\langle\nabla\mathcal{L}_{s_m}(\boldsymbol{\theta}), \boldsymbol{\theta} - \boldsymbol{\theta}^*\rangle \geq \frac{1}{L+\mu}\|\nabla\mathcal{L}_{s_m}(\boldsymbol{\theta})\|^2 + \frac{\mu L}{L+\mu}\|\boldsymbol{\theta} - \boldsymbol{\theta}^*\|^2. \tag{90}$$

Substituting Eq. (90) back into Eq. (87) with $\boldsymbol{\theta} = \boldsymbol{\theta}_{m-1}$:

$$\begin{aligned}&\|\boldsymbol{\theta}_m - \boldsymbol{\theta}^*\|^2 \\ &\leq \|\boldsymbol{\theta}_{m-1} - \boldsymbol{\theta}^*\|^2 - 2\gamma\left(\frac{1}{L+\mu}\|\nabla\mathcal{L}_{s_m}(\boldsymbol{\theta}_{m-1})\|^2 + \frac{\mu L}{L+\mu}\|\boldsymbol{\theta}_{m-1} - \boldsymbol{\theta}^*\|^2\right) + \gamma^2\|\nabla\mathcal{L}_{s_m}(\boldsymbol{\theta}_{m-1})\|^2 \\ &= \left(1 - \frac{2\gamma\mu L}{L+\mu}\right)\|\boldsymbol{\theta}_{m-1} - \boldsymbol{\theta}^*\|^2 + \left(\gamma^2 - \frac{2\gamma}{L+\mu}\right)\|\nabla\mathcal{L}_{s_m}(\boldsymbol{\theta}_{m-1})\|^2.\end{aligned} \tag{91}$$

Provided that the step size satisfies $\gamma \in (0, \frac{2}{L+\mu}]$, the coefficient $\left(\gamma^2 - \frac{2\gamma}{L+\mu}\right)$ is non-positive. Since $\|\nabla\mathcal{L}_{s_m}(\boldsymbol{\theta}_{m-1})\|^2 \geq 0$, we can drop the gradient norm term to obtain an upper bound:

$$\|\boldsymbol{\theta}_m - \boldsymbol{\theta}^*\|^2 \leq \left(1 - \frac{2\gamma\mu L}{L+\mu}\right)\|\boldsymbol{\theta}_{m-1} - \boldsymbol{\theta}^*\|^2. \tag{92}$$

Since this inequality holds for any realization of the random sample s_m, we take the expectation over the sampling distribution. Let $\mathbb{E}[\cdot]$ denote the total expectation over the sequence of random indices $\{s_1, \ldots, s_m\}$. We have:

$$\mathbb{E}[\|\boldsymbol{\theta}_m - \boldsymbol{\theta}^*\|^2] \leq \left(1 - \frac{2\gamma\mu L}{L+\mu}\right)\mathbb{E}[\|\boldsymbol{\theta}_{m-1} - \boldsymbol{\theta}^*\|^2]. \tag{93}$$

Applying this recurrence relation recursively for T steps yields:

$$\mathbb{E}[\|\boldsymbol{\theta}_T - \boldsymbol{\theta}^*\|^2] \leq \left(1 - \frac{2\gamma\mu L}{L+\mu}\right)^T \|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*\|^2. \tag{94}$$

Specifically, when choosing the step size $\gamma = \frac{2}{L+\mu}$:

$$1 - \frac{2\gamma\mu L}{L+\mu} = 1 - \frac{4\mu L}{(L+\mu)^2} = \frac{(L-\mu)^2}{(L+\mu)^2} = \left(\frac{\kappa-1}{\kappa+1}\right)^2, \tag{95}$$

where $\kappa = L/\mu$ is the condition number. Thus, we obtain the convergence rate:

$$\mathbb{E}[\|\boldsymbol{\theta}_T - \boldsymbol{\theta}^*\|^2] \leq \left(\frac{\kappa-1}{\kappa+1}\right)^{2T} \|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*\|^2. \tag{96}$$

□

Appendix H. Third-Order Implicit Bias Analysis

In this section, we analyze the third-order implicit bias of Nexus, inspired by recent works [11, 13, 78]. While the second-order analysis reveals how Nexus aligns gradients, it does not fully explain the *stability* of this alignment in complex landscapes. Here, we demonstrate that the Nexus update direction implicitly minimizes a "Generalized Directional Sharpness" metric. This implies that Nexus actively seeks regions where the loss landscape is not only aligned but also locally flat along the alignment direction, thereby preventing the "de-alignment" caused by sharp curvature.

H.1. Setup and Definitions

To perform this analysis, we verify the behavior of the third-order terms in the Taylor expansion. We introduce a standard assumption regarding the smoothness of the Hessian.

Assumption 4 (Bounded Third Derivative). Assume the third-order derivative tensor is bounded, i.e., for any unit vectors $\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}$ and any task i, there exists a constant M_3 such that $\|\nabla^3 \mathcal{L}_i(\boldsymbol{\theta})[\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}]\|_2 \leq M_3$. This implies that the third-order Taylor remainder satisfies $\|\boldsymbol{r}_{Taylor}(\boldsymbol{\delta})\|_2 \leq \frac{M_3}{6}\|\boldsymbol{\delta}\|_2^3$.

Definition (Generalized Directional Sharpness). We define the generalized sharpness term involving the Hessian of task j and the gradient directions of tasks i and p as:

$$\mathcal{R}_{i,j,p}(\boldsymbol{\theta}) \triangleq \frac{1}{2}\nabla \mathcal{L}_i(\boldsymbol{\theta})^\top \nabla^2 \mathcal{L}_j(\boldsymbol{\theta}) \nabla \mathcal{L}_p(\boldsymbol{\theta}). \tag{97}$$

This term measures the curvature of task j along the plane spanned by the gradients of tasks i and p. When $i = p$, this reduces to the standard directional sharpness, quantifying how fast the gradient changes along the update direction.

H.2. Proof of Theorem 5

1. Exact Expansion of the Gradient. Consider the m-th inner step with sampled task s_m. Let $\boldsymbol{\theta}_{m-1} = \boldsymbol{\theta}_0 + \boldsymbol{\Delta}_{m-1}$. The exact third-order Taylor expansion is:

$$\nabla \mathcal{L}_{s_m}(\boldsymbol{\theta}_{m-1}) = \nabla \mathcal{L}_{s_m}(\boldsymbol{\theta}_0) + \nabla^2 \mathcal{L}_{s_m}(\boldsymbol{\theta}_0)\boldsymbol{\Delta}_{m-1} + \frac{1}{2}\nabla^3 \mathcal{L}_{s_m}(\boldsymbol{\theta}_0)[\boldsymbol{\Delta}_{m-1}, \boldsymbol{\Delta}_{m-1}] + \boldsymbol{r}_{Taylor}^{(m)}. \tag{98}$$

The remainder is bounded by $\|\boldsymbol{r}_{Taylor}^{(m)}\|_2 \leq \frac{M_3}{6}\|\boldsymbol{\Delta}_{m-1}\|_2^3$. Using the bound on displacement magnitude $\|\boldsymbol{\Delta}_{m-1}\|_2 \leq (m-1)\gamma$:

$$\|\boldsymbol{r}_{Taylor}^{(m)}\|_2 \leq \frac{M_3}{6}(m-1)^3\gamma^3. \tag{99}$$

2. Displacement Decomposition. We define the ideal displacement using initial gradients as $\tilde{\boldsymbol{\Delta}}_{m-1} = \sum_{l=1}^{m-1} -\gamma \hat{\boldsymbol{d}}_{s_l}$. The true displacement is $\boldsymbol{\Delta}_{m-1} = \tilde{\boldsymbol{\Delta}}_{m-1} + \boldsymbol{\delta}_{m-1}$. Using the Lipschitz constant $L_1 = L/G_{\min}$ for the normalized gradient, the accumulated error is bounded by:

$$\|\boldsymbol{\delta}_{m-1}\|_2 \leq \gamma \sum_{l=1}^{m-1} L_1(l-1)\gamma \leq \frac{L_1}{2}(m-1)^2\gamma^2. \tag{100}$$

3. Substitution into Quadratic Term. We substitute $\boldsymbol{\Delta}_{m-1}$ into the third-order term. By multilinearity of the tensor:

$$\frac{1}{2}\nabla^3\mathcal{L}_{s_m}[\boldsymbol{\Delta}_{m-1},\boldsymbol{\Delta}_{m-1}] = \frac{1}{2}\nabla^3\mathcal{L}_{s_m}[\tilde{\boldsymbol{\Delta}}_{m-1},\tilde{\boldsymbol{\Delta}}_{m-1}] + \boldsymbol{r}_{sub}^{(m)}. \tag{101}$$

The residual $\boldsymbol{r}_{sub}^{(m)}$ accounts for the cross-terms and quadratic error terms. Its norm is strictly bounded by:

$$\|\boldsymbol{r}_{sub}^{(m)}\|_2 \le \frac{1}{2}\left(2\|\nabla^3\|\|\tilde{\boldsymbol{\Delta}}_{m-1}\|\|\boldsymbol{\delta}_{m-1}\| + \|\nabla^3\|\|\boldsymbol{\delta}_{m-1}\|^2\right). \tag{102}$$

Substituting the bounds for $\|\tilde{\boldsymbol{\Delta}}_{m-1}\|$ and $\|\boldsymbol{\delta}_{m-1}\|$:

$$\begin{aligned}\|\boldsymbol{r}_{sub}^{(m)}\|_2 &\le M_3((m-1)\gamma)\left(\frac{L_1}{2}(m-1)^2\gamma^2\right) + \frac{M_3}{2}\left(\frac{L_1}{2}(m-1)^2\gamma^2\right)^2 \\ &= \frac{M_3L_1}{2}(m-1)^3\gamma^3 + \frac{M_3L_1^2}{8}(m-1)^4\gamma^4.\end{aligned} \tag{103}$$

4. Derivation of the Expected Update Direction. The explicit third-order component of the update (excluding residuals) is:

$$\boldsymbol{v}_3 = \sum_{m=1}^{k} -\gamma\left(\frac{1}{2}\nabla^3\mathcal{L}_{s_m}[\tilde{\boldsymbol{\Delta}}_{m-1},\tilde{\boldsymbol{\Delta}}_{m-1}]\right). \tag{104}$$

Substituting $\tilde{\boldsymbol{\Delta}}_{m-1} = \sum_{l=1}^{m-1} -\gamma\hat{\boldsymbol{d}}_{s_l}$:

$$\boldsymbol{v}_3 = -\frac{\gamma^3}{2}\sum_{m=1}^{k}\sum_{l=1}^{m-1}\sum_{p=1}^{m-1}\nabla^3\mathcal{L}_{s_m}[\hat{\boldsymbol{d}}_{s_l},\hat{\boldsymbol{d}}_{s_p}]. \tag{105}$$

Taking the expectation over uniform sampling of indices s_m, s_l, s_p, each triplet (i,j,p) appears with probability $1/k^3$:

$$\begin{aligned}\mathbb{E}[\boldsymbol{v}_3] &= -\frac{\gamma^3}{2}\sum_{m=1}^{k}\sum_{l=1}^{m-1}\sum_{p=1}^{m-1}\mathbb{E}_{s_m,s_l,s_p}\left[\nabla^3\mathcal{L}_{s_m}[\hat{\boldsymbol{d}}_{s_l},\hat{\boldsymbol{d}}_{s_p}]\right] \\ &= -\frac{\gamma^3}{2}\left(\sum_{m=1}^{k}\sum_{l=1}^{m-1}\sum_{p=1}^{m-1}1\right)\left(\frac{1}{k^3}\sum_{j,i,p}\nabla^3\mathcal{L}_j[\hat{\boldsymbol{d}}_i,\hat{\boldsymbol{d}}_p]\right) \\ &= -\frac{\gamma^3}{2}\left(\sum_{m=1}^{k}(m-1)^2\right)\left(\frac{1}{k^3}\sum_{i,j,p}\nabla^3\mathcal{L}_j[\hat{\boldsymbol{d}}_i,\hat{\boldsymbol{d}}_p]\right).\end{aligned} \tag{106}$$

Using the summation formula $\sum_{m=1}^{k}(m-1)^2 = \frac{k(k-1)(2k-1)}{6}$:

$$\mathbb{E}[\boldsymbol{v}_3] = -\gamma^3\frac{(k-1)(2k-1)}{12k^2}\sum_{i,j,p}\nabla^3\mathcal{L}_j[\hat{\boldsymbol{d}}_i,\hat{\boldsymbol{d}}_p]. \tag{107}$$

Recognizing that $\nabla_{\boldsymbol{\theta}}\mathcal{R}_{i,j,p} = \frac{1}{2}\nabla^3\mathcal{L}_j[\hat{\boldsymbol{d}}_i, \hat{\boldsymbol{d}}_p]$ (treating the direction vectors as locally constant for the gradient of the surrogate), we can rewrite the update as a gradient descent step on the sharpness metric:

$$\mathbb{E}[\boldsymbol{v}_3] = -\gamma\nabla\left(\gamma^2\frac{(k-1)(2k-1)}{6k^2}\sum_{i,j,p}\mathcal{R}_{i,j,p}\right). \tag{108}$$

This confirms that Nexus implicitly minimizes the generalized directional sharpness.

5. Bounding the Total Residual. The total error vector is $\boldsymbol{\mathcal{E}}_{3rd} = \sum_{m=1}^{k} -\gamma(\boldsymbol{r}_{Taylor}^{(m)} + \boldsymbol{r}_{sub}^{(m)})$. Taking the norm:

$$\|\boldsymbol{\mathcal{E}}_{3rd}\|_2 \leq \gamma\sum_{m=1}^{k}\left(\left(\frac{M_3}{6} + \frac{M_3L_1}{2}\right)(m-1)^3\gamma^3 + \frac{M_3L_1^2}{8}(m-1)^4\gamma^4\right). \tag{109}$$

Using summation bounds $\sum_{j=1}^{k-1} j^3 \leq \frac{k^4}{4}$ and $\sum_{j=1}^{k-1} j^4 \leq \frac{k^5}{5}$, and substituting $L_1 = L/G_{\min}$:

$$\|\boldsymbol{\mathcal{E}}_{3rd}\|_2 \leq \left(\frac{M_3}{24} + \frac{M_3L}{8G_{\min}}\right)k^4\gamma^4 + \frac{M_3L^2}{40G_{\min}^2}k^5\gamma^5. \tag{110}$$

□

Appendix I. More Experiments Details

I.1. Detailed Experimental Settings

Our experimental setup largely follows the protocols in Wen et al. [77] and OLMo et al. [53].

Pretraining Datasets. We utilize an in-house pretraining dataset similar to [70]. This corpus is: (1) strictly cleaned to ensure no data contamination regarding the evaluated benchmarks or distillation data; and (2) of higher quality and stability than typical open-source datasets, allowing us to observe smooth and clear optimization trends. We also conduct experiments on public datasets [5] in Appendix I.4. However, these public datasets are not strictly decontaminated and contain training samples from our benchmarks. This leads to artificially inflated performance on certain tasks while underperforming on others. Consequently, we primarily rely on the strictly cleaned dataset for more stable analysis.

Model Architecture. Following Wen et al. [77], we train Llama-architecture models of 130M, 300M, 520M, 1.2B, and 2.3B parameters (excluding embeddings). We primarily analyze the 520M (1B total parameters) and 2.3B (3B total parameters) models, hereafter referred to by their total parameter counts for brevity, except in the scaling law analysis (Appendix B.4) as required by Kaplan et al. [31].

Hyperparameters. Wen et al. [77] have already conducted extensive parameter searches using grid search, coordinate descent, and fine-grained tuning. To ensure fairness, we always apply exact the same hyper-parameters to both Nexus and its corresponding base optimizers. For the base optimizers, we adopt the optimal hyperparameters identified in Wen et al. [77]. We further verified these settings by sweeping the learning rate with a multiplier of 2 (i.e., verifying $0.5\times$ and $2.0\times$), confirming that their configurations remain optimal for our dataset. See Appendix I.2 for the detailed hyperparameters in each experiment.

Benchmarks. We evaluate on diverse benchmarks encompassing general knowledge (MMLU [25]), reasoning (GPQA, GPQA Diamond [62], BBH [69]), math (GSM8k [10], MATH500 [26]), and coding (HumanEval [18], MBPP [4]). Beyond discrete accuracies, we also track downstream task losses and out-of-distribution (OOD) loss. The OOD loss is evaluated on a strictly cleaned proprietary in-house corpus, which exhibits a strong correlation with downstream benchmark capabilities.

Highlighting Strategy. We use **bold** to highlight non-trivial performance gaps, defined as a loss difference > 0.01 or a benchmark improvement $> 2\%$, following Wen et al. [77].

I.2. Detailed Hyper-parameters

Our hyperparameter configurations strictly follow the baselines established by Wen et al. [77]. To ensure these settings are optimal for our specific pretraining corpus, we conducted a grid search by scaling the learning rate with multipliers of $0.5\times$ and $2.0\times$. Empirical results confirmed that the original learning rate settings remain optimal for our setup. For clarity and reproducibility, we summarize the key hyperparameters in Tab. 4.

In experiments utilizing the Warmup-Stable-Decay (WSD) scheduler [78], we consistently employ 1,000 warmup steps and 10,000 decay steps. Across all experiments, we maintain a global batch size of 256, Adam β values of $(0.9, 0.95)$, an Adam ϵ of 10^{-10}, and a gradient clipping norm of 1.0.

Additionally, we provide training compute estimates for each experiment in Tab. 4. Please note that the reported "A100 days" serve strictly as a standardized estimation metric to facilitate comparison, rather than indicating the specific hardware utilized for our training runs.

Table 4: Summary of key hyperparameters. WD denotes Weight Decay.

Size	Optimizer	Outer LR	Inner LR	Chinchilla	Tokens (B)	WD	A100 Days	Reference
1B	Adam	0.002	-	4×	50	0.2	∼60	Appendix B.4
	Nexus	0.002	0.01	4×	50	0.2		
3B	Adam	0.001	-	2×	110	0.2	∼352	Sec B.3, I.3, B.4, I.4
	Muon	0.001	-	2×	110	0.1		
	Nexus	0.001	0.01	2×	110	0.2		

I.3. Robustness to Data Mixing

Motivation. In Sec. 3.2 and Eq. (2), we show that the gradient similarity implies the marginal gains on task i when optimizing on task j (in the first-order sense):

$$\underbrace{\mathcal{L}_i(\boldsymbol{\theta}) - \mathcal{L}_i(\boldsymbol{\theta} - \gamma \nabla \mathcal{L}_j(\boldsymbol{\theta}))}_{\text{decrease of task i after one GD step on task j}} = \gamma \nabla \mathcal{L}_i(\boldsymbol{\theta})^\top \nabla \mathcal{L}_j(\boldsymbol{\theta}) + O(\gamma^2). \tag{111}$$

Since Nexus encourages gradient similarity across the training set, optimizing a sample-dense domain implicitly optimizes sample-sparse domains. Therefore, we conjecture that Nexus acts like a dynamic data mixture, which boosts the sample-sparse or harder-to-learn domains within the mixture without manual re-weighting.

Setup. To validate our hypothesis, we construct three distinct data mixtures by explicitly anchoring the sampling weight of the mathematics domain to 10%, 40%, and 70% (denoted as Math10, Math40, and Math70). Accordingly, we downsample the remaining data sources to fulfill the complementary proportion (e.g., Math70 consists of 70% math and 30% downsampled other data). We train 3B models on each mixture using both AdamW and Nexus, strictly adhering to the hyperparameter settings detailed in Appendix I.1.

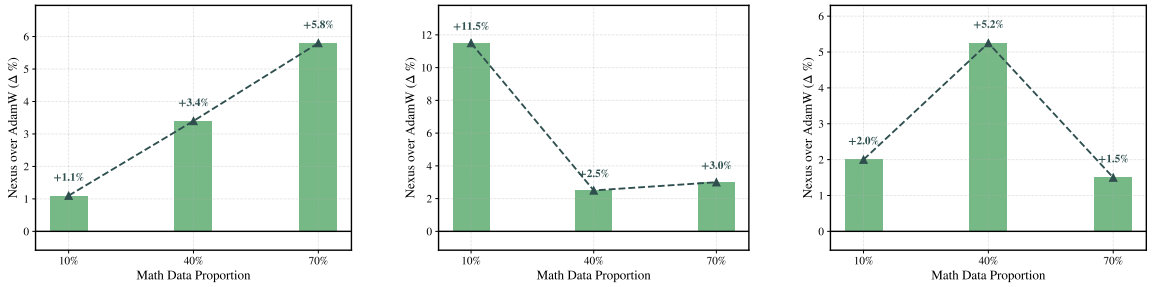
Results. As shown in Fig. 7 and Tab. 5, increasing the proportion of math data from 10% to 70% gradually reduces Nexus's relative gain on math reasoning. Conversely, as general data becomes the relative minority, Nexus yields a larger improvement in this domain, increasing its gain from **+1.1%** to **+5.8%**. Interestingly, the gain on coding tasks exhibits a non-monotonic trend, which we hypothesize is because code generation is a composite capability requiring a complex balance of both logical reasoning and domain knowledge. Furthermore, Nexus mitigates the performance fluctuations observed in the baseline across these mixture shifts. These results support our conjecture that Nexus acts as an implicit balancer, dynamically prioritizing under-optimized tasks without manual mixture tuning.

As shown in Tab. 5, we observe a dynamic trade-off mechanism:

- **In the sample-sparse regime (Math10):** Where math data is scarce, the baseline optimizer struggles to generalize on reasoning tasks. Nexus provides the most significant gains here (e.g., **+15.0** on GSM8k), effectively "mining" the rare training signals to build robust reasoning capabilities.

Table 5: **Results on varying data mixtures** (3B models). Hyperparameters follow Appendices B.3 and I.1. As the proportion of math data increases (10% → 70%), the relative performance gains of Nexus on math benchmarks gradually diminish, whereas its advantages on other domains (General, Reasoning) progressively expand. This suggests Nexus boosts the sample-sparse or harder-to-learn domains in the mixture.

Data	Optim.	Metric	Loss Metrics (↓)		Gen.	Reasoning			Math		Code		Avg.
			Pretrain.	OOD	MMLU	GPQA	GPQA-D	BBH	GSM8k	MATH	HumanEval	MBPP	All
Math10	AdamW	Acc. (↑)	1.606	1.302	47.8	**32.8**	22.6	36.6	44.0	32.0	43.0	38.0	37.1
		Loss (↓)			2.265	2.005	1.910	1.534	1.259	1.054	1.116	1.922	1.633
	Nexus	Acc. (↑)	1.602	**1.290**	48.9	29.6	23.4	36.6	**59.0**	**40.0**	**47.0**	38.0	**40.3**
		Loss (↓)			**2.179**	**1.981**	**1.881**	**1.504**	**1.227**	**1.026**	**1.086**	1.921	**1.601**
	Improv.	Loss (↑)	+0.004	**+0.012**	**+0.086**	**+0.024**	**+0.029**	**+0.030**	**+0.032**	**+0.028**	**+0.030**	+0.001	**+0.032**
Math40	AdamW	Acc. (↑)	1.336	1.330	47.8	29.6	22.6	38.1	64.0	44.0	41.0	38.0	40.6
		Loss (↓)			2.210	1.989	1.891	1.522	1.171	0.969	1.144	1.976	1.609
	Nexus	Acc. (↑)	1.339	1.331	**51.2**	**33.5**	**27.3**	**41.1**	**70.0**	43.0	**45.0**	**44.5**	**44.4**
		Loss (↓)			**2.182**	1.990	1.889	**1.511**	**1.117**	**0.929**	**1.132**	**1.876**	**1.578**
	Improv.	Loss (↑)	-0.003	-0.001	**+0.028**	-0.001	+0.002	**+0.011**	**+0.054**	**+0.040**	**+0.012**	**+0.100**	**+0.031**
Math70	AdamW	Acc. (↑)	1.033	**1.399**	44.0	27.3	23.4	41.1	77.0	45.0	38.0	38.0	41.7
		Loss (↓)			2.252	2.025	1.923	1.541	1.111	0.923	1.178	1.897	1.606
	Nexus	Acc. (↑)	1.040	1.409	**49.8**	**30.4**	23.4	42.5	76.0	**52.0**	**41.0**	38.0	**44.1**
		Loss (↓)			**2.221**	2.037	1.936	1.548	**1.082**	**0.911**	1.176	**1.872**	1.598
	Improv.	Loss (↑)	-0.007	-0.010	**+0.031**	-0.012	-0.013	-0.007	**+0.029**	**+0.012**	+0.002	**+0.025**	+0.008



(a) General Domain (MMLU) (b) Math Domain (Avg.) (c) Code Domain (Avg.)

Figure 7: **Results on varying data mixtures** (3B models). As the proportion of math data increases (10% → 70%), the relative performance gains of Nexus on math benchmarks gradually diminish, whereas its advantages on General domain progressively expand. This suggests Nexus boosts the sample-sparse or harder-to-learn domains in the mixture.

- **In the sample-dense regime (Math70):** As math data becomes abundant, the baseline catches up on math benchmarks. However, Nexus automatically shifts its advantage to the *now-relative-minority* domains. It significantly boosts General Knowledge (MMLU: **+5.8**) and broad Reasoning (GPQA: **+3.1**) compared to the baseline, which begins to suffer from domain dominance.

- **Lower sensitivity to mixture shifts:** Nexus also demonstrates higher stability against drastic changes in data mixture. When shifting from a math-heavy (Math70) to a math-sparse (Math10) mixture, the performance variance of Nexus is significantly smaller than that of the baseline. For instance, while the baseline's GSM8k score drops precipitously by **33.0 points** (from 77.0 to 44.0), Nexus mitigates this degradation, dropping only **17.0 points** (from 76.0 to 59.0). Similarly, on MMLU, while the baseline fluctuates by **3.8 points**, Nexus remains highly stable with a variation of less than **1.0 point** (49.8 vs. 48.9), demonstrating its stability against data mixture changes.

This suggests that Nexus reduces sensitivity to manual data mixing ratios, acting as an automatic balancer that prioritizes representations for the most under-optimized tasks in the mixture.

I.4. Experiments on a Public Dataset

Motivation. While our primary analyses utilize strictly cleaned data to avoid confounding factors, many popular open-source pretraining datasets inevitably suffer from data contamination, inadvertently including benchmark training sets (e.g., GSM8k). We evaluate Nexus on a public dataset from Basant et al. [5] to investigate whether its consensus-seeking mechanism remains robust and mitigates shortcut over-memorization in the presence of such noisy, contaminated signals.

Settings. We train the 1B and 3B models on a public dataset [5]. All other training configurations, including model architectures and base optimizer hyperparameters, are kept strictly identical to the main experiments detailed in Appendix B.3.

Table 6: **Results on a public pretraining dataset [5].** The Adam baseline exhibits artificial performance inflation on leaked benchmarks. In contrast, Nexus effectively resists shortcut over-memorization, successfully reallocating model capacity to uncontaminated tasks and achieving superior overall OOD generalization.

Model	Optim.	Metric	Loss Metrics (↓)		Gen.	Reasoning			Math		Code		Avg.
			Pretrain.	OOD	MMLU	GPQA	GPQA-D	BBH	GSM8k	MATH	HumanEval	MBPP	All
1B	Adam	Acc. (↑)	1.331	1.863	**34.2**	25.8	18.8	24.8	18.0	14.0	38.0	1.0	21.8
		Loss (↓)			2.552	2.280	2.191	1.700	**1.708**	1.346	1.324	2.991	2.011
	Nexus	Acc. (↑)	1.338	**1.835**	31.4	25.0	18.8	23.0	**22.0**	12.0	**41.0**	**15.0**	23.5
		Loss (↓)			**2.446**	**2.261**	**2.172**	**1.689**	1.749	**1.325**	1.325	**2.908**	**1.984**
	Improv.	Loss (↑)	-0.007	**+0.028**	**+0.106**	**+0.019**	**+0.019**	**+0.011**	**-0.041**	**+0.021**	-0.001	**+0.083**	**+0.027**
3B	Adam	Acc. (↑)	1.330	1.623	55.1	25.0	**27.3**	**47.4**	44.0	31.0	59.0	23.0	39.0
		Loss (↓)			**2.356**	2.062	1.975	**1.529**	**1.519**	1.121	1.205	2.732	1.812
	Nexus	Acc. (↑)	1.338	**1.606**	56.2	25.8	23.4	44.4	**47.0**	32.0	**63.0**	**38.0**	**41.2**
		Loss (↓)			2.380	**2.047**	**1.957**	1.540	1.533	**1.106**	1.199	**2.530**	**1.786**
	Improv.	Loss (↑)	-0.008	**+0.017**	**-0.024**	**+0.015**	**+0.018**	**-0.011**	**-0.014**	**+0.015**	+0.006	**+0.202**	**+0.026**

Results. As shown in Tab. 6, the Adam baseline exhibits artificial performance inflation on potentially contaminated benchmarks like GSM8k. In contrast, Nexus resists overfitting to these leaked signals and effectively reallocates the model's capacity to uncontaminated, sparse domains. This dynamic balancing is evidenced by the striking improvements on coding tasks—such as MBPP accuracy increasing from 1.0% to 15.0% (1B) and 23.0% to 38.0% (3B)—ultimately leading to a consistently lower OOD loss across both scales.

Table 7: **Results under different learning rate schedulers.** We evaluate the 3B model trained with AdamW and Nexus using both WSD and Cosine schedulers. The results demonstrate that the "same pretraining loss, better downstream performance" phenomenon is highly robust regardless of the scheduler.

Schedule	Optim.	Metric	Loss Metrics (↓)		Gen.	Reasoning			Math		Code		Avg.
			Eval	OOD	MMLU	GPQA	GPQA-D	BBH	GSM8k	MATH	HumanEval	MBPP	All
WSD	AdamW	Acc. (↑)	1.606	1.302	47.8	**32.8**	22.6	36.6	44.0	32.0	43.0	38.0	37.1
		Loss (↓)			2.265	2.005	1.910	1.534	1.259	1.054	1.116	1.922	1.633
	Nexus	Acc. (↑)	1.602	**1.290**	48.9	29.6	**23.4**	36.6	**59.0**	**40.0**	**47.0**	38.0	**40.3**
		Loss (↓)			**2.179**	**1.981**	**1.881**	**1.504**	**1.227**	**1.026**	**1.086**	1.921	**1.601**
	Improv.	Loss (↑)	+0.004	**+0.012**	**+0.086**	**+0.024**	**+0.029**	**+0.030**	**+0.032**	**+0.028**	**+0.030**	+0.001	**+0.032**
Cosine	AdamW	Acc. (↑)	1.526	1.255	53.2	26.6	19.5	**41.5**	60.0	32.0	56.0	39.0	41.0
		Loss (↓)			2.195	1.924	1.829	1.480	1.212	1.022	1.045	1.867	1.572
	Nexus	Acc. (↑)	1.528	1.250	54.9	**30.5**	**27.3**	34.8	59.0	**41.0**	54.0	**46.0**	**43.4**
		Loss (↓)			**2.115**	1.917	1.826	1.479	**1.169**	**0.994**	**1.025**	**1.805**	**1.541**
	Improv.	Loss (↑)	-0.002	+0.005	**+0.080**	+0.007	+0.003	+0.001	**+0.043**	**+0.028**	**+0.020**	**+0.062**	**+0.030**

I.5. Robustness to Learning Rate Schedule

Motivation. While the Warmup-Stable-Decay (WSD) scheduler [27] has become increasingly popular in recent LLM pretraining, the Cosine annealing schedule remains a widely adopted standard [77, 78]. To ensure that our observed generalization benefits are not merely an artifact of a specific learning rate dynamic, we evaluate the robustness of Nexus across different schedulers.

Settings. We conduct an ablation study by replacing the default WSD scheduler with a standard Cosine learning rate scheduler. All other training configurations, including the 3B model architecture, data mixture, and base optimizer hyperparameters, remain strictly identical to the main setup detailed in Appendix B.3.

Results. As demonstrated in Tab. 7, the "same pretraining loss, better downstream performance" phenomenon persists consistently across both schedulers. Under the Cosine schedule, Nexus maintains a negligible pretraining loss difference compared to the AdamW baseline (1.528 vs. 1.526) while delivering substantial improvements on downstream metrics, such as a +0.03 loss gain on downstream benchmarks. This confirms that the implicit bias introduced by Nexus is highly robust and orthogonal to the choice of learning rate trajectory.

Table 8: **Scaling Analysis on Training Tokens.** The downstream advantage of Nexus over the AdamW baseline persists strictly when extending training to $4\times$ Chinchila.

Chinchila	Optim.	Metric	Loss Metrics (↓)		Gen.	Reasoning			Math		Code		Avg.
			Pretrain.	OOD	MMLU	GPQA	GPQA-D	BBH	GSM8k	MATH	HumanEval	MBPP	All
2	AdamW	Acc. (↑)	1.606	1.302	47.8	**32.8**	22.6	36.6	44.0	32.0	43.0	38.0	37.1
		Loss (↓)			2.265	2.005	1.910	1.534	1.259	1.054	1.116	1.922	1.633
	Nexus	Acc. (↑)	1.602	**1.290**	48.9	29.6	23.4	36.6	**59.0**	**40.0**	**47.0**	38.0	**40.3**
		Loss (↓)			**2.179**	**1.981**	**1.881**	**1.504**	**1.227**	**1.026**	**1.086**	1.921	**1.601**
	Improv.	Loss (↑)	+0.004	**+0.012**	**+0.086**	**+0.024**	**+0.029**	**+0.030**	**+0.032**	**+0.028**	**+0.030**	+0.001	**+0.032**
4	AdamW	Acc. (↑)	1.591	1.293	48.3	**23.4**	21.9	35.2	54.0	33.0	45.0	43.0	38.0
		Loss (↓)			2.240	1.975	1.880	1.513	1.245	1.038	1.119	1.976	1.623
	Nexus	Acc. (↑)	1.588	**1.281**	**52.8**	20.3	**25.0**	**44.1**	**62.0**	33.0	**49.0**	**47.0**	**41.7**
		Loss (↓)			**2.216**	**1.957**	**1.863**	**1.501**	**1.229**	**1.008**	**1.087**	**1.885**	**1.593**
	Improv.	Loss (↑)	+0.003	**+0.012**	**+0.024**	**+0.018**	**+0.017**	**+0.012**	**+0.016**	**+0.030**	**+0.032**	**+0.091**	**+0.030**

I.6. Detailed Results for Model Size Scaling

This section provides the detailed experimental results corresponding to the model size scaling analysis discussed in Appendix B.4.

As demonstrated above and analyzed in Appendix B.4, Nexus consistently outperforms the base optimizer across all evaluated model scales, with average benchmark accuracy improvements of +0.8% (130M), +1.5% (300M), +1.7% (520M), +2.6% (1.2B), and +3.2% (2.3B).

I.7. Experiments on Muon Optimizers

This section provides the detailed experimental results discussed in Appendices B.3 and B.6.

As demonstrated above, Nexus achieves comparable downstream performance to Muon, despite maintaining a pretraining loss that is nearly identical to the AdamW baseline. These results explicitly indicate that while Muon improves downstream performance primarily by reaching a significantly lower pretraining loss, the gains from Nexus stem directly from its favorable implicit bias.

I.8. Downstream SFT

Motivation and Settings. To verify whether the performance gains of Nexus are merely a result of "pre-consuming" the potential improvements of the SFT phase in advance, we evaluate the supervised fine-tuning (SFT) performance of our checkpoints. We use an SFT dataset similar to [70] and branch off from the 100,000-step checkpoints of the experiments in Appendix I.5. Training is conducted on the SFT data with a learning rate of 2×10^{-5} and a global batch size of 256, which matches the pretraining learning rate and batch size at the 100,000-step mark. This setup can be viewed as continuing the learning rate decay on the SFT dataset, consistent with standard practices [70, 72, 79].

Nexus still outperforms AdamW after SFT. As shown in Tab. 11, after supervised fine-tuning, Nexus achieves an average accuracy of 42.7%, surpassing the AdamW baseline by 2.0%. Specifically, Nexus outperforms AdamW by 7.0% on MATH, 6.0% on HumanEval, and 6.0% on MBPP. These results indicate that Nexus does not prematurely compromise the model's capacity for downstream alignment.

Table 9: **Benchmark Performance across Model Scales.** We compare downstream capabilities for models ranging from 130M to 2.3B parameters. Notably, the relative gains of Nexus over the base optimizer amplify as model capacity increases, with the average benchmark accuracy improvement growing from +0.8% on the 130M model to +3.2% on the 2.3B model.

Size	Optim.	Metric	Loss Metrics (↓)		Gen.	Reasoning			Math		Code		Avg.
			Pretrain.	OOD	MMLU	GPQA	GPQA-D	BBH	GSM8k	MATH	HumanEval	MBPP	All
130M	AdamW	Acc. (↑)	2.038	1.559	28.0	22.6	**24.2**	25.1	7.0	6.0	0.0	10.0	15.4
		Loss (↓)			2.555	2.438	2.338	1.793	1.612	1.360	1.407	2.230	1.967
	Nexus	Acc. (↑)	2.031	1.549	27.0	**25.7**	21.8	**28.8**	5.0	**11.0**	0.0	10.0	16.2
		Loss (↓)			**2.523**	**2.414**	**2.312**	1.792	**1.601**	**1.330**	**1.384**	**2.183**	**1.942**
	Improv.	Acc. (↑)	-	-	-1.0	**+3.1**	-2.4	**+3.7**	-2.0	**+5.0**	0.0	0.0	+0.8
		Loss (↑)	+0.007	+0.010	**+0.032**	**+0.024**	**+0.026**	+0.001	**+0.011**	**+0.030**	**+0.023**	**+0.047**	**+0.024**
300M	AdamW	Acc. (↑)	1.909	1.474	**33.3**	26.5	21.8	31.8	15.0	15.0	6.0	16.0	20.7
		Loss (↓)			2.495	2.296	2.196	1.704	1.471	1.255	1.322	2.141	1.860
	Nexus	Acc. (↑)	1.901	1.469	30.3	27.3	**25.0**	30.7	14.0	16.0	**13.0**	**21.0**	22.2
		Loss (↓)			**2.381**	**2.278**	**2.177**	1.707	1.470	**1.237**	**1.298**	**2.046**	**1.824**
	Improv.	Acc. (↑)	-	-	-3.0	+0.8	**+3.2**	-1.1	-1.0	+1.0	**+7.0**	**+5.0**	+1.5
		Loss (↑)	+0.008	+0.005	**+0.114**	**+0.018**	**+0.019**	-0.003	+0.001	**+0.018**	**+0.024**	**+0.095**	**+0.036**
520M	AdamW	Acc. (↑)	1.826	1.433	32.1	25.0	21.8	29.6	18.0	13.0	19.0	17.0	21.9
		Loss (↓)			2.363	2.221	2.124	1.640	1.429	1.204	1.270	2.035	1.786
	Nexus	Acc. (↑)	1.826	1.428	33.5	**30.4**	21.8	29.3	20.0	13.0	19.0	**22.0**	23.6
		Loss (↓)			**2.316**	**2.201**	**2.102**	1.638	**1.396**	**1.176**	1.261	**1.977**	**1.758**
	Improv.	Acc. (↑)	-	-	+1.4	**+5.4**	0.0	-0.3	+2.0	0.0	0.0	**+5.0**	+1.7
		Loss (↑)	0.000	+0.005	**+0.047**	**+0.020**	**+0.022**	+0.002	**+0.033**	**+0.028**	+0.009	**+0.058**	**+0.027**
1.2B	AdamW	Acc. (↑)	1.714	1.364	**44.4**	22.6	24.2	27.4	30.0	23.0	30.0	31.0	29.1
		Loss (↓)			2.626	2.410	2.000	1.799	1.338	1.112	1.199	2.023	1.813
	Nexus	Acc. (↑)	1.707	1.358	41.7	**25.7**	23.4	**32.9**	**37.0**	**28.0**	**35.0**	30.0	**31.7**
		Loss (↓)			**2.466**	**2.373**	**1.984**	1.792	**1.325**	**1.109**	**1.179**	**1.987**	**1.777**
	Improv.	Acc. (↑)	-	-	-2.7	**+3.1**	-0.8	**+5.5**	**+7.0**	**+5.0**	**+5.0**	-1.0	**+2.6**
		Loss (↑)	+0.007	+0.006	**+0.160**	**+0.037**	**+0.016**	+0.007	**+0.013**	+0.003	**+0.020**	**+0.036**	**+0.036**
2.3B	AdamW	Acc. (↑)	1.606	1.302	47.8	**32.8**	22.6	36.6	44.0	32.0	43.0	38.0	37.1
		Loss (↓)			2.265	2.005	1.910	1.534	1.259	1.054	1.116	1.922	1.633
	Nexus	Acc. (↑)	1.602	**1.290**	48.9	29.6	**23.4**	36.6	**59.0**	**40.0**	**47.0**	38.0	**40.3**
		Loss (↓)			**2.179**	**1.981**	**1.881**	**1.504**	**1.227**	**1.026**	**1.086**	1.921	**1.601**
	Improv.	Acc. (↑)	-	-	+1.1	-3.2	+0.8	0.0	**+15.0**	**+8.0**	**+4.0**	0.0	**+3.2**
		Loss (↑)	+0.004	**+0.012**	**+0.086**	**+0.024**	**+0.029**	**+0.030**	**+0.032**	**+0.028**	**+0.030**	+0.001	**+0.032**

Nexus maintains lower SFT loss than AdamW throughout training. We observe that for the pre-SFT checkpoints, Nexus already yields a lower loss on the SFT dataset compared to AdamW (1.647 vs. 1.655). This result demonstrates that the geometric properties optimized by Nexus during pretraining translate into better generalization even before any explicit fine-tuning. Furthermore, this lower SFT loss is consistently maintained throughout the entire training process, as evidenced by the post-SFT loss (1.028 for Nexus vs. 1.035 for AdamW). These observations indicate the potential of Nexus for continual training and extended optimization phases.

Table 10: **Comparison with Muon Optimizer on 3B Models.** As shown, Muon improves downstream performance by decreasing the pretraining loss. While Nexus achieves nearly the same pretraining loss as AdamW, it achieves comparable performance to Muon on downstream tasks.

Optim.	Metric	Loss Metrics (↓)		Gen.	Reasoning			Math		Code		Avg.
		Pretrain.	OOD	MMLU	GPQA	GPQA-D	BBH	GSM8k	MATH	HumanEval	MBPP	All
AdamW	Acc. (↑)	1.606	1.302	47.8	32.8	22.6	36.6	44.0	32.0	43.0	38.0	37.1
	Loss (↓)			2.265	2.005	1.910	1.534	1.259	1.054	1.116	1.922	1.633
Adam+Nexus	Acc. (↑)	1.602	**1.290**	48.9	29.6	23.4	36.6	**59.0**	**40.0**	**47.0**	38.0	**40.3**
	Loss (↓)			**2.179**	**1.981**	**1.881**	**1.504**	**1.227**	**1.026**	**1.086**	1.921	**1.601**
(- AdamW)	Acc. (↑)	-	-	+1.1	-3.2	+0.8	0.0	**+15.0**	**+8.0**	**+4.0**	0.0	**+3.2**
	Loss (↑)	+0.004	**+0.012**	**+0.086**	**+0.024**	**+0.029**	**+0.030**	**+0.032**	**+0.028**	**+0.030**	+0.001	**+0.032**
Muon	Acc. (↑)	1.577	1.285	49.8	32.0	24.2	41.9	46.0	38.0	40.0	43.0	39.4
	Loss (↓)			2.188	1.968	1.874	1.502	1.236	1.035	1.091	1.951	1.606
(- AdamW)	Acc. (↑)	-	-	+2.0	-0.8	+1.6	+5.3	+2.0	+6.0	-3.0	+5.0	+2.3
	Loss (↑)	+0.029	+0.017	+0.077	+0.037	+0.036	+0.032	+0.023	+0.019	+0.025	-0.029	+0.027

Table 11: **Downstream SFT Results.** As shown, Nexus does not prematurely compromise the model's SFT capabilities; on the contrary, it continues to outperform AdamW after SFT.

Phase	Optim.	Metric	Loss Metrics (↓)		Gen.	Reasoning			Math		Code		Avg.
			SFT	OOD	MMLU	GPQA	GPQA-D	BBH	GSM8k	MATH	HumanEval	MBPP	All
Pre-SFT	AdamW	Acc. (↑)	1.655	1.263	52.7	**28.9**	25.8	**41.1**	54.0	37.0	**50.0**	42.0	**41.4**
		Loss (↓)			2.221	1.938	1.842	1.484	1.233	1.031	1.053	1.859	1.583
	Nexus	Acc. (↑)	1.647	1.258	50.9	22.7	22.7	35.9	**57.0**	**40.0**	48.0	42.0	39.9
		Loss (↓)			**2.138**	1.929	1.838	1.489	**1.179**	**1.006**	**1.030**	**1.803**	**1.552**
	Improv.	Loss (↑)	+0.008	+0.005	**+0.083**	+0.009	+0.004	-0.005	**+0.054**	**+0.025**	**+0.023**	**+0.056**	**+0.031**
Post-SFT	AdamW	Acc. (↑)	1.035	1.278	51.4	28.9	**33.6**	**45.6**	58.0	28.0	40.0	40.0	40.7
		Loss (↓)			2.244	2.006	1.915	1.575	1.377	1.111	1.077	1.957	1.658
	Nexus	Acc. (↑)	1.028	1.274	**54.7**	28.9	29.7	39.3	**62.0**	**35.0**	**46.0**	**46.0**	**42.7**
		Loss (↓)			**2.220**	**1.990**	**1.901**	**1.551**	**1.299**	**1.076**	**1.060**	1.952	**1.631**
	Improv.	Acc. (↑)	-	-	**+3.3**	0.0	-3.9	-6.3	**+4.0**	**+7.0**	**+6.0**	**+6.0**	**+2.0**
		Loss (↑)	+0.007	+0.004	**+0.024**	**+0.016**	**+0.014**	**+0.024**	**+0.078**	**+0.035**	**+0.017**	+0.005	**+0.027**

I.9. Experiment using SGDM

Motivation. We also conduct an experiment using SGD with Momentum (SGDM). However, SGDM typically converges slower than AdamW in LLM pretraining [85].

Muon-Inspired SGDM. To this end, we adapt the Muon optimizer [30] to construct a faster SGDM baseline. Specifically, we control the RMS norm of SGDM updates to be the same as the orthogonalized updates in Muon, as shown in Fig. 8. This design is inspired by the empirical findings in Liu et al. [44], which suggest that maintaining a consistent per-tensor RMS norm aligns with the principles of Maximal Update Parameterization (μP). This may stabilize update dynamics and mitigate sensitivity to hyperparameter selection, and typically facilitate acceleration [59]. Surprisingly, in our LLM pretraining settings, this significantly accelerates SGDM, reaching speeds competitive with AdamW.

Algorithm Newton-Schulz Iteration

```
def zeropower_via_ns(G, steps=5):
  a, b, c = 3.4445, -4.7750, 2.0315
  X = G / (torch.norm(G) + 1e-7)
  for _ in range(steps):
      A = X @ X.mT
      X = a*X + (b*A + c*A@A) @ X
  return X
```

Algorithm RMS-Aligned Update

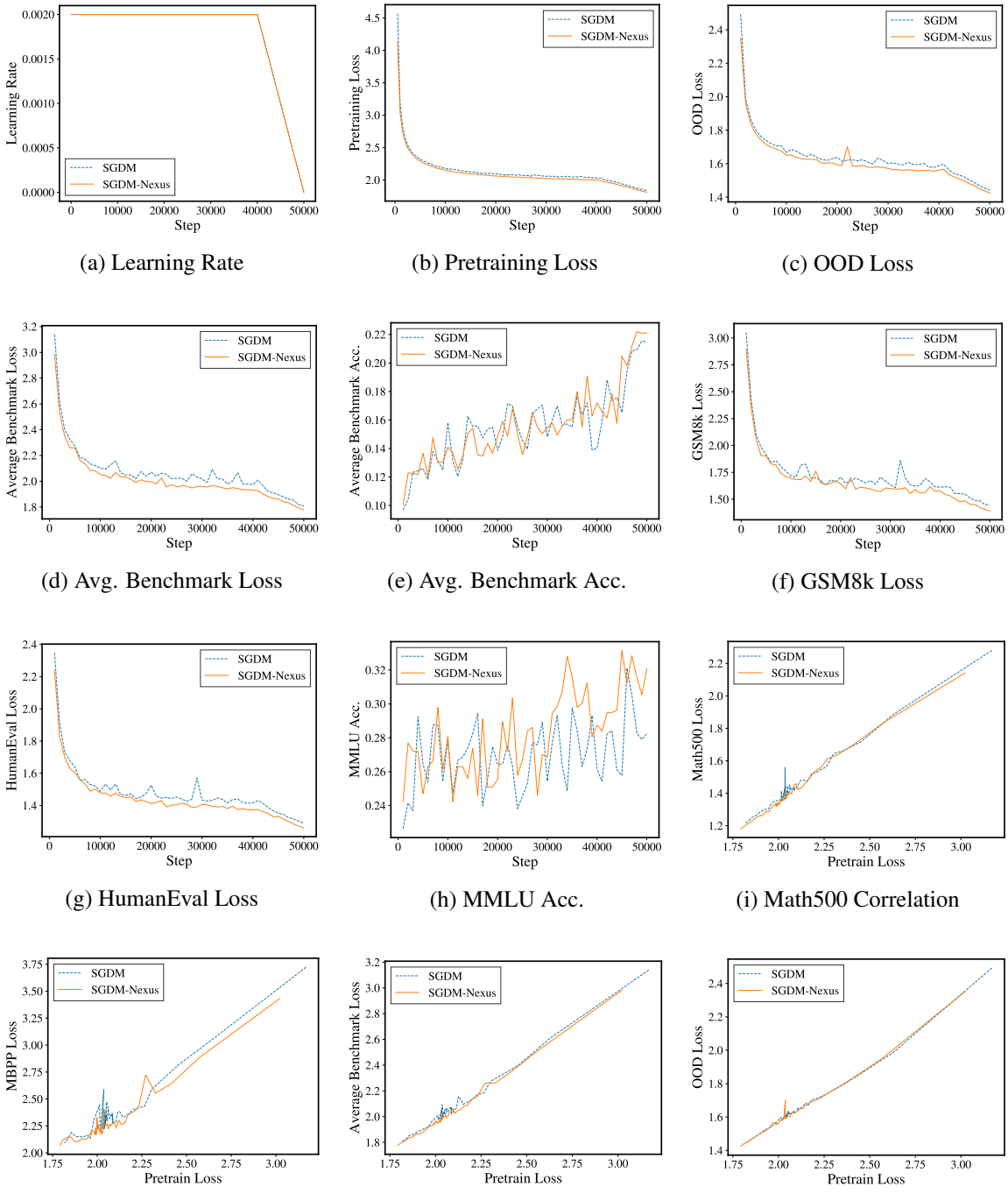
```
def sgd_alignRMS_update(G):
  # K = min(G.shape[-2:])
  # Normalize to match Muon scale
  norm = torch.norm(G) + 1e-7
  return G * (K**0.5) / norm
```

Figure 8: Muon-inspired SGDM. **Left:** Quintic Newton-Schulz iterations for orthogonalization. **Right:** Per-tensor RMS normalization ensuring consistent update scale as Muon.

Settings. Due to computational constraints, we only conduct experiments on 1B models, strictly adhering to the experimental settings detailed in Appendix I.1.

Results. On this SGDM baseline, we observe the following results (Fig. 9):

- **Pretraining Acceleration.** Surprisingly, Nexus significantly accelerates pretraining loss minimization on SGDM. While the baseline SGDM is notably slower than AdamW, SGDM equipped with Nexus even surpasses the AdamW baseline. This acceleration may be attributed to the mechanism described in Appendix G.
- **Robust Implicit Bias.** Since Nexus accelerates pretraining loss, direct comparisons of absolute downstream performance are confounded by the lower pretraining loss. Thus, we analyze the correlation between pretraining and downstream losses (Fig. 9). As shown in Fig. 9, the Nexus-SGDM curve lies consistently below the baseline SGDM, confirming that the favorable implicit bias persists even under different base optimizers.



(a) Learning Rate (b) Pretraining Loss (c) OOD Loss

(d) Avg. Benchmark Loss (e) Avg. Benchmark Acc. (f) GSM8k Loss

(g) HumanEval Loss (h) MMLU Acc. (i) Math500 Correlation

(j) MBPP Correlation (k) Avg. Loss Correlation (l) OOD Correlation

Figure 9: **Experimental Results using SGDM.** Nexus both gives acceleration and implicit bias.