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

Huanran Chen[1,2,‡] **Huaqing Zhang**[1] **Xiao Li**[2,†]
Yinpeng Dong[1,*] **Ke Shen**[2] **Jun Zhu**[1,*]
[1]Tsinghua University [2]ByteDance Seed
[‡]Work done at ByteDance Seed, [†]Project Lead, [*]Corresponding authors

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. 2b), or merely a minimizer of the averaged loss (e.g., Fig. 2a)? 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. Extensive experiments across models ranging from 130M to 3B parameters demonstrate that Nexus significantly boosts downstream performance, despite achieving nearly *the same pretraining loss* (see Fig. 1). Notably, 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.

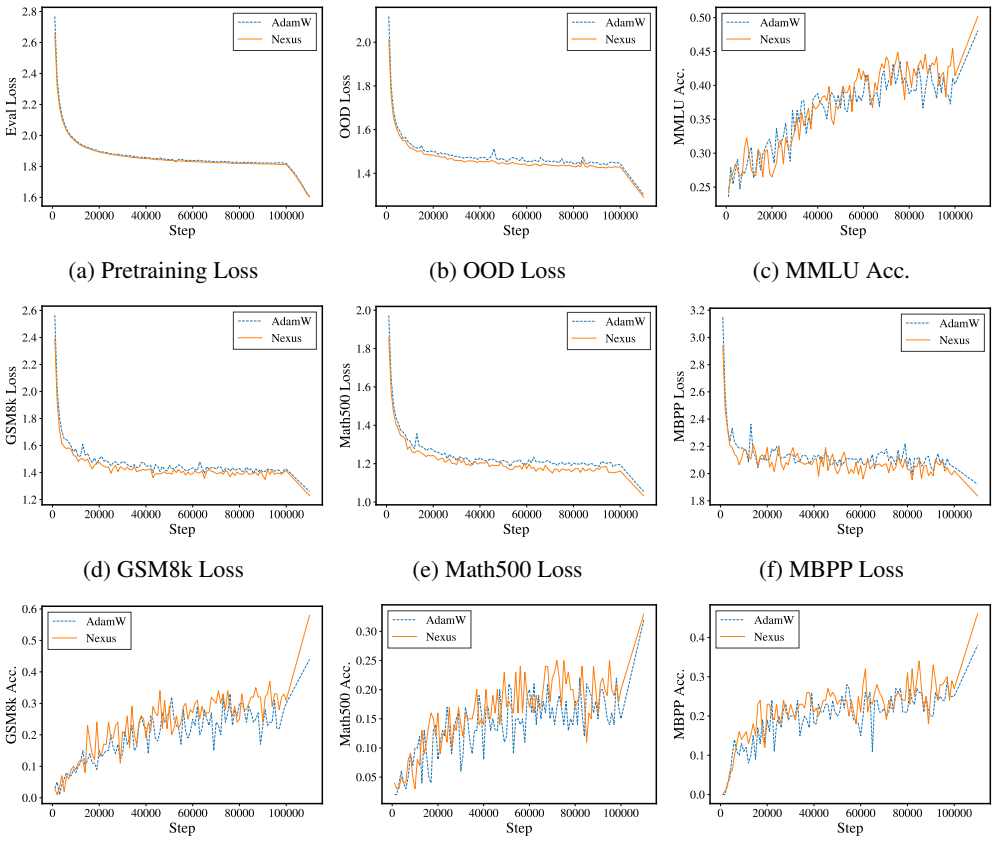
1 Introduction

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 [1, 2, 3]. During pretraining, LLMs acquire foundational knowledge from unprecedentedly large and diverse data sources, encompassing a vast array of domains such as general language, mathematics, code, and complex reasoning [4, 5, 6]. 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** (Fig. 2a), 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** (Fig. 2b), 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 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

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



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

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

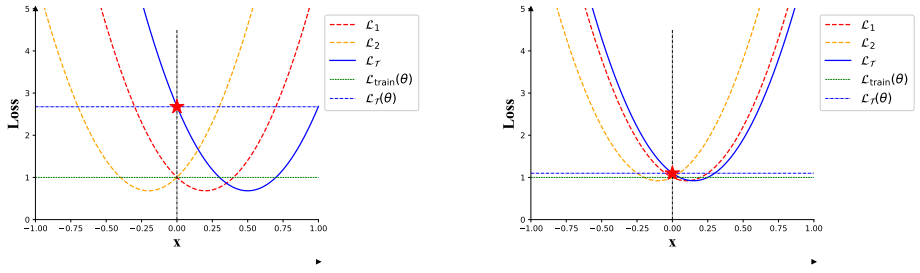
(g) GSM8k Acc. (h) Math500 Acc. (i) MBPP Acc.

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

quadratic and i.i.d. distributed, the Intersection-type minimizer (Fig. 2b) will strictly outperform the Average-type minimizer (Fig. 2a) on the downstream task $\mathcal{L}_{\mathcal{T}}$, given the same pretraining loss (see Theorem 2.1). 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, $\text{CosSim}(\nabla\mathcal{L}_i, \nabla\mathcal{L}_j) \triangleq \frac{\nabla\mathcal{L}_i^\top \nabla\mathcal{L}_j}{\|\nabla\mathcal{L}_i\|\|\nabla\mathcal{L}_j\|}$, 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 $\nabla\text{CosSim}(\nabla\mathcal{L}_i, \nabla\mathcal{L}_j)$. Nexus is compatible with the existing optimization techniques [7, 8, 9] to effectively maximize $\text{CosSim}(\nabla\mathcal{L}_i, \nabla\mathcal{L}_j)$. In Sec. 5.1, we show that both gradient similarity and geometric closeness generalize to downstream tasks, thus 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 [2, 7, 10], diverse pretraining data and mixtures [11, 12], learning rate schedules [13, 14, 15], and training compute [16]. Experimental results demonstrate that, across nearly all settings, *Nexus reduces the downstream loss by over 0.02 compared to the base optimizers—an improvement comparable to that of Muon over AdamW [7] (see Tab. 11)—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 [17], particularly as the current pretraining paradigm transitions from being compute-bound to data-bound [18, 19, 20, 21].



(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* $\mathcal{L}_\mathcal{T}$.

2 Closeness and Downstream Generalization

2.1 Problem Formulation

Formally, let the pretraining corpus be the union of K distinct data sources, denoted as $\mathcal{D}_{\text{train}} = \cup_{k=1}^{K} \mathcal{D}_k$. Let α_k represent the sampling probability (data mixing ratio) for the k-th source. We define the *weighted* empirical loss function for the k-th source as: $\mathcal{L}_k(\boldsymbol{\theta}) = -\frac{\alpha_k}{|\mathcal{D}_k|} \sum_{j=1}^{|\mathcal{D}_k|} \log p(x_j|\boldsymbol{\theta})$. Consequently, the total pretraining objective is simply the average of these weighted losses: $\mathcal{L}_{\text{train}}(\boldsymbol{\theta}) = \frac{1}{K} \sum_{k=1}^{K} \mathcal{L}_k(\boldsymbol{\theta})$.

2.2 Flatness and Closeness are both Second Order Generalization Biases

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 minimizer of the downstream task, $\boldsymbol{\theta}^*_\mathcal{T}$, we can characterize the downstream loss of pretraining minimizer:

$$\mathcal{L}_\mathcal{T}(\boldsymbol{\theta}^*_{\text{train}}) = \mathcal{L}_\mathcal{T}(\boldsymbol{\theta}^*_\mathcal{T}) + \frac{1}{2} \underbrace{\|\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\theta}^*_\mathcal{T}\|_2^2}_{\text{Closeness}} \cdot \underbrace{\boldsymbol{u}^\top \nabla^2 \mathcal{L}_\mathcal{T}(\boldsymbol{\xi}) \boldsymbol{u}}_{\text{Directional Sharpness}}, \tag{1}$$

where $\boldsymbol{\xi} \in [\boldsymbol{\theta}^*_\mathcal{T}, \boldsymbol{\theta}^*_{\text{train}}]$, $\boldsymbol{u} = \frac{\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\theta}^*_\mathcal{T}}{\|\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\theta}^*_\mathcal{T}\|_2}$. Note that the first-order term vanishes because $\boldsymbol{\theta}^*_\mathcal{T}$ is a local minimizer (i.e., $\nabla \mathcal{L}_\mathcal{T}(\boldsymbol{\theta}^*_\mathcal{T}) = \mathbf{0}$). The remaining term is controlled by two scalars: the directional *Flatness* $\boldsymbol{u}^\top \nabla^2 \mathcal{L}_\mathcal{T}(\boldsymbol{\xi}) \boldsymbol{u} \leq \|\nabla^2 \mathcal{L}_\mathcal{T}(\boldsymbol{\xi})\|_2 \leq \|\nabla^2 \mathcal{L}_\mathcal{T}(\boldsymbol{\xi})\|_F$, and the *Closeness* $\|\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\theta}^*_\mathcal{T}\|_2^2$.

Therefore, the flatter the local loss landscape of $\mathcal{L}_\mathcal{T}$ and the closer the converged parameter $\boldsymbol{\theta}^*_{\text{train}}$ is to the task minimizer $\boldsymbol{\theta}^*_\mathcal{T}$, the better the generalization. Together, flatness and closeness encapsulate all second-order information for downstream generalization. While flatness has been well-studied in prior literature [22, 23, 24, 25, 26], in this work, we focus solely on our new implicit bias: closeness.

2.3 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 (Fig. 2b) yields better downstream generalization compared to a "distant" minimizer (Fig. 2a).

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 B.1). We substantiate this intuition under a simplified quadratic setting in the main text:

Theorem 2.1 (Generalization of Closeness in the Quadratic Case, Proof in Appendix B.1.1)**.** *To model the non-convex landscape, assume the parameter space* $\mathbb{R}^d$ *is partitioned into a set of disjoint basins of attraction* $\{\mathcal{B}\}$*. Within any specific basin* $\mathcal{B}$*, assume that any task* $\mathcal{L}$ *sampled from a distribution* $\mathcal{P}$ *is locally a quadratic function:* $\mathcal{L}(\boldsymbol{\theta}) = \frac{a}{2}\|\boldsymbol{\theta} - \boldsymbol{\theta}^*_\mathcal{L}\|_2^2 + c_\mathcal{B}$*, where* $a > 0$ *is shared across tasks and basins,* $\boldsymbol{\theta}^*_\mathcal{L}$ *is the task-specific minimizer in basin* $\mathcal{B}$*, and* $c_\mathcal{B}$ *is the intrinsic loss*

(depth) of the basin. We assume the quadratic representation holds at the empirical minimizer within each basin. Let $\boldsymbol{\mu}_{\mathcal{B}} = \mathbb{E}_{\mathcal{L}\sim\mathcal{P}}[\boldsymbol{\theta}^*_{\mathcal{L}}]$ *and define the variance of task-specific minimizers as* $\sigma^2_{\mathcal{B}} = \mathbb{E}_{\mathcal{L}\sim\mathcal{P}}\left[\|\boldsymbol{\theta}^*_{\mathcal{L}} - \boldsymbol{\mu}_{\mathcal{B}}\|_2^2\right] < \infty$*. Let the pretraining tasks* $\{\mathcal{L}_k\}_{k=1}^K$ *and the downstream task* $\mathcal{L}_{\mathcal{T}}$ *be i.i.d. samples from* $\mathcal{P}$*. Let* $\Theta = \left\{\boldsymbol{\theta}^*_{\textit{train},\mathcal{B}} \,\middle|\, \mathbb{E}_{\{\mathcal{L}_k\}}\left[\mathcal{L}_{\textit{train}}(\boldsymbol{\theta}^*_{\textit{train},\mathcal{B}})\right] = C_{\textit{train}}\right\}$ *be the set of converged minimizers across different basins that achieve the same expected pretraining loss* $C_{\textit{train}}$*.*

For any fixed basin $\mathcal{B}$ *whose converged minimizer belongs to* Θ*, the expected downstream loss on an unseen task* $\mathcal{T} \sim \mathcal{P}$ *satisfies*

$$\mathbb{E}_{\{\mathcal{L}_k\},\mathcal{T}}\left[\mathcal{L}_{\mathcal{T}}(\boldsymbol{\theta}^*_{\textit{train},\mathcal{B}})\right] = C_{\textit{train}} + \frac{a}{K}\sigma^2_{\mathcal{B}}. \tag{2}$$

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 must coincide. Theoretically, both the gradient dot product and cosine similarity serve as tight bounds for closeness:

Theorem 3.1 (Gradient Similarity Upper Bounds Closeness)**.** *Let* $\boldsymbol{\theta}$ *be a converged parameter satisfying* $\nabla\mathcal{L}_{train}(\boldsymbol{\theta}) = \frac{1}{K}\sum_{k=1}^K \nabla\mathcal{L}_k(\boldsymbol{\theta}) = \mathbf{0}$*. Let* $\mathcal{S}_k = \{\boldsymbol{\vartheta} \mid \exists\epsilon > 0, \forall\boldsymbol{\vartheta}' \in B_\epsilon(\boldsymbol{\vartheta}), \mathcal{L}_k(\boldsymbol{\vartheta}) \leq \mathcal{L}_k(\boldsymbol{\vartheta}')\}$ *be the set of local minimizers for task k, and let* $\boldsymbol{\theta}^*_k = \arg\min_{\boldsymbol{\vartheta}\in\mathcal{S}_k} \|\boldsymbol{\vartheta} - \boldsymbol{\theta}\|_2$ *be a closest local minimizer, assumed to exist. Assume that each* $\mathcal{L}_k$ *is twice continuously differentiable along* $[\boldsymbol{\theta}^*_k, \boldsymbol{\theta}]$ *and that* $\nabla\mathcal{L}_k(\boldsymbol{\theta}) \neq \mathbf{0}$ *for all k, so that cosine similarity is well-defined. Define* $\boldsymbol{u}_k = \frac{\boldsymbol{\theta}-\boldsymbol{\theta}^*_k}{\|\boldsymbol{\theta}-\boldsymbol{\theta}^*_k\|_2}$*,* $\lambda_{\min} = \min_k \inf_{\boldsymbol{\xi}\in[\boldsymbol{\theta}^*_k,\boldsymbol{\theta}]} \boldsymbol{u}_k^\top \nabla^2\mathcal{L}_k(\boldsymbol{\xi})\boldsymbol{u}_k > 0$*, and let* $G = \sup_k \|\nabla\mathcal{L}_k(\boldsymbol{\theta})\|_2$*. Then, the closeness between the minimizers is bounded by:*

$$\begin{aligned}\frac{1}{K}\sum_{k=1}^K \|\boldsymbol{\theta} - \boldsymbol{\theta}^*_k\|_2^2 &\leq \frac{1}{K\lambda^2_{\min}} \sum_{i\neq j} \left(-\nabla\mathcal{L}_i(\boldsymbol{\theta})^\top \nabla\mathcal{L}_j(\boldsymbol{\theta})\right) \\ &\leq \frac{G^2}{K\lambda^2_{\min}} \sum_{i\neq j} \left(1 - \textit{CosSim}(\nabla\mathcal{L}_i(\boldsymbol{\theta}), \nabla\mathcal{L}_j(\boldsymbol{\theta}))\right).\end{aligned} \tag{3}$$

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):

$$\underbrace{\mathcal{L}_{\mathcal{T}}(\boldsymbol{\theta}) - \mathcal{L}_{\mathcal{T}}(\boldsymbol{\theta} - \gamma\nabla\mathcal{L}_{\text{train}}(\boldsymbol{\theta}))}_{\text{decrease of downstream loss after one GD step on training set}} = \gamma\nabla\mathcal{L}_{\text{train}}(\boldsymbol{\theta})^\top \nabla\mathcal{L}_{\mathcal{T}}(\boldsymbol{\theta}) + O(\gamma^2). \tag{4}$$

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:

$$\nabla_{\boldsymbol{\theta}}\text{CosSim}(\nabla_{\boldsymbol{\theta}}\mathcal{L}_i(\boldsymbol{\theta}), \nabla_{\boldsymbol{\theta}}\mathcal{L}_j(\boldsymbol{\theta})) = \left(\nabla_{\boldsymbol{\theta}} \frac{\nabla_{\boldsymbol{\theta}}\mathcal{L}_i(\boldsymbol{\theta})}{\|\nabla_{\boldsymbol{\theta}}\mathcal{L}_i(\boldsymbol{\theta})\|_2}\right)^\top \frac{\nabla_{\boldsymbol{\theta}}\mathcal{L}_j(\boldsymbol{\theta})}{\|\nabla_{\boldsymbol{\theta}}\mathcal{L}_j(\boldsymbol{\theta})\|_2} + \left(\nabla_{\boldsymbol{\theta}} \frac{\nabla_{\boldsymbol{\theta}}\mathcal{L}_j(\boldsymbol{\theta})}{\|\nabla_{\boldsymbol{\theta}}\mathcal{L}_j(\boldsymbol{\theta})\|_2}\right)^\top \frac{\nabla_{\boldsymbol{\theta}}\mathcal{L}_i(\boldsymbol{\theta})}{\|\nabla_{\boldsymbol{\theta}}\mathcal{L}_i(\boldsymbol{\theta})\|_2}. \tag{5}$$

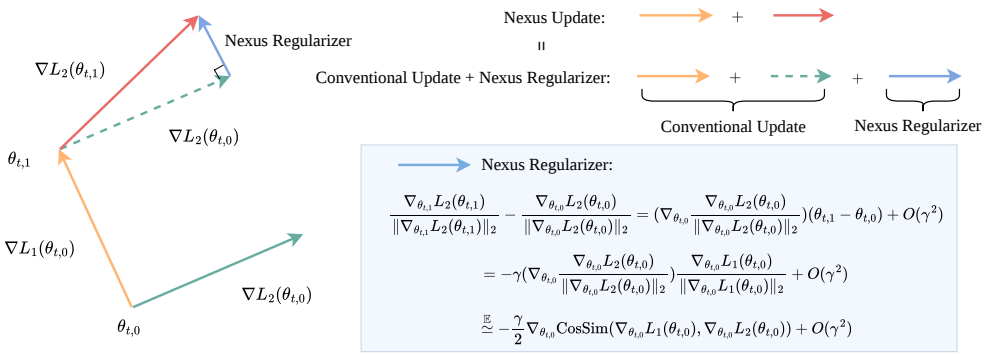


Figure 3: Illustration of Nexus Algorithm. $\overset{\mathbb{E}}{\simeq}$ denotes expected equality over task permutations.

To address this challenge, we propose the **Nexus optimizer**, which approximates the gradient in Eq. (5) through a dual-loop mechanism. The complete procedure is outlined in Algorithm 3. 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. (5). Specifically, for each outer 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 [8, 27], SGDM [28]) to perform the actual update. Theorem 3.2 demonstrates that Nexus maximizes gradient similarity.

Theorem 3.2 (Nexus Maximizes Gradient Similarity)**.** *Assume there exist constants $G_{\min}, L, \rho > 0$ such that for any $t \in [1, T]$ and $m \in [1, K]$:*

$$\|\nabla \mathcal{L}_i(\boldsymbol{\theta}_{t,m})\|_2 \geq G_{\min}; \quad \|\nabla^2 \mathcal{L}_i(\boldsymbol{\theta})\|_2 \leq L; \quad \|\nabla^2 \mathcal{L}_i(\boldsymbol{x}) - \nabla^2 \mathcal{L}_i(\boldsymbol{y})\|_2 \leq \rho \|\boldsymbol{x} - \boldsymbol{y}\|_2. \tag{6}$$

Then, the sequence $\{\boldsymbol{\theta}_t\}$ generated by Algorithm 3 minimizes the following second-order objective:

$$\mathcal{J}_{2nd}(\boldsymbol{\theta}) = \gamma \sum_{i=1}^{K} \mathcal{L}_i(\boldsymbol{\theta}) - \gamma^2 \frac{K-1}{4K} \sum_{i \neq j} \mathit{CosSim}\Big(\nabla \mathcal{L}_i(\boldsymbol{\theta}), \nabla \mathcal{L}_j(\boldsymbol{\theta})\Big). \tag{7}$$

This holds because the expected update direction satisfies:

$$\mathbb{E}[\hat{\boldsymbol{g}}_t] = \gamma \sum_{i=1}^{K} \frac{\nabla \mathcal{L}_i(\boldsymbol{\theta}_t)}{\|\nabla \mathcal{L}_i(\boldsymbol{\theta}_t)\|_2} - \gamma^2 \frac{K-1}{4K} \Big(\nabla_{\boldsymbol{\theta}} \sum_{i \neq j} \mathit{CosSim}(\nabla \mathcal{L}_i, \nabla \mathcal{L}_j) + \boldsymbol{\mathcal{E}}_{sym,i,j} \Big) + \boldsymbol{\mathcal{E}}_{2nd}, \tag{8}$$

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 [29, 30, 31, 32]. The Taylor approximation error is bounded by $\|\boldsymbol{\mathcal{E}}_{2nd}\|_2 \leq \frac{1}{6} \left(\frac{4L^2 + \rho G_{\min}}{G_{\min}^2} \right) K^3 \gamma^3 = \mathcal{O}(\gamma^3)$.

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. (5).

Strictly speaking, Nexus should be conceptualized as a *gradient approximator* rather than a standalone optimizer, for two reasons: (1) The inner optimization step must be a simple, memory-free update like NSGD or its generalized transformations (see Appendix C.5). Otherwise, as shown in Sec. 5.1, incorporating historical momentum would disrupt the accurate maximization of gradient similarity.

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()
```

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()
```

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 (Algorithm 3) to pretraining using an auxiliary `inner_model`, updated after each mini-batch. At each accumulation boundary, we use `inner_model` − `model` as the outer optimizer's pseudo-gradient $\hat{\boldsymbol{g}}$, then reset the inner model to the updated main model.

(2) The outer optimization step simply consumes the approximated gradient and can utilize standard optimization algorithms (see Appendix D.10). Consequently, Nexus acts as a modular plug-in that is broadly compatible with a wide range of outer base optimizers (e.g., AdamW, SGDM).

3.3 Adapting Nexus to Practical Pretraining

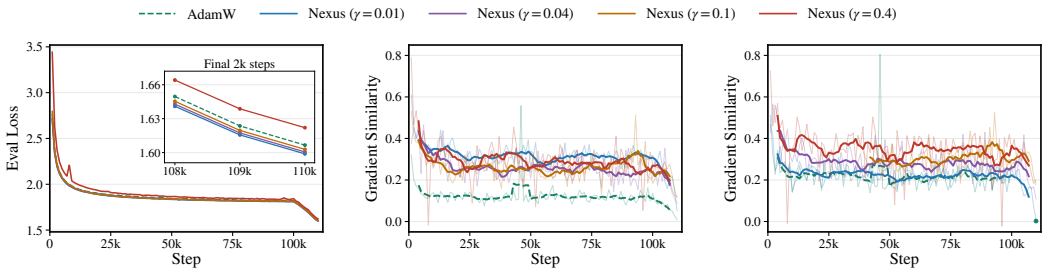
Motivation. While Nexus effectively maximizes gradient similarity, directly applying Algorithm 3 to pretraining requires computing gradients for every data source to perform a single effective outer update. In pretraining, the number of data sources is typically large (e.g., $K > 50$), which would result in an effective batch size that differs significantly from standard settings [16, 7]. 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 Algorithm 1, 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 Algorithm 2. 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 3.

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 $\sum_l \mathrm{CosSim}(\nabla_{\boldsymbol{W}^{(l)}}\mathcal{L}_i, \nabla_{\boldsymbol{W}^{(l)}}\mathcal{L}_j)$ (see Appendix C.5), rather than the global gradient similarity $\mathrm{CosSim}(\nabla_{\boldsymbol{\theta}}\mathcal{L}_i, \nabla_{\boldsymbol{\theta}}\mathcal{L}_j)$. This design choice is primarily motivated by the theoretical insights of μP [33, 34, 35] and recent practical wisdom [36] 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 [16]. 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 Algorithm 2 for all experiments, with the exception of specific ablation studies. Readers may proceed directly to Sec. 4. In Appendix C, we also provide a theoretical analysis of Nexus's convergence speed and discuss its implications for standard Normalized SGD.



(a) Pretraining Loss (b) Within-Pretraining Similarity (c) Train-to-Downstream Similarity

Figure 5: **Inner learning rate sweep.** We use the same 3B pretraining setting as in Sec. 4.3. As shown, when the inner LR (i.e., regularization coefficient) increases, while the pretraining loss increases, the gradient similarity within the pretraining set, $\mathrm{CosSim}(\nabla\mathcal{L}_i, \nabla\mathcal{L}_j)$, increases, and this gain carries over to train-to-downstream gradient similarity, $\mathrm{CosSim}(\nabla\mathcal{L}_{\mathrm{train}}, \nabla\mathcal{L}_{\mathcal{T}})$, thus improving downstream generalization (Tab. 5).

4 Experiments

In this section and Appendix D, 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 D.

4.1 Experimental Settings

Our experimental setup strictly follows the protocols in Wen et al. [7]. 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. [7] without any Nexus-specific tuning. See Appendix D.1 for detailed configurations.

4.2 Inner Learning Rate

Motivation. The inner learning rate γ is the only additional hyperparameter introduced by Nexus compared to its base optimizer. To determine its value, we sweep $\gamma \in \{0.01, 0.04, 0.10, 0.40\}$ using the 3B pretraining setting in Sec. 4.3, while keeping all other hyperparameters and the training budget unchanged.

Results. As shown in Eq. (7), the relative weights of the pretraining loss and the gradient similarity regularizer scale as $O(1 : \gamma)$. Therefore, a larger γ gives the Nexus regularizer more weight relative to the pretraining loss. As illustrated in Figs. 5b and 5c, increasing γ consistently improves gradient similarity both within the pretraining set and between the pretraining and OOD sets. Specifically, as γ increases from 0.01 to 0.10, the average within-pretraining gradient similarity increases from 0.4540 to 0.5465, while the pretraining-to-OOD gradient similarity increases from 0.2252 to 0.3042 (Tab. 5). This improvement in gradient alignment is accompanied by better downstream generalization: the average downstream accuracy increases from 38.2% to 40.2%, compared to 37.1% for the AdamW baseline. Meanwhile, Nexus maintains nearly the same pretraining validation loss as AdamW (1.602 vs. 1.606), as shown in Fig. 5a. However, further increasing γ to 0.40, while improving both gradient similarity metrics, increases the pretraining validation loss to 1.622 and reduces the average downstream accuracy to 37.7%.

Conclusion. Based on these observations, we choose $\gamma = 0.10$, the largest tested inner learning rate that preserves the baseline's pretraining loss. This choice allows us to strengthen the Nexus regularizer as much as possible within the tested range, thereby encouraging greater gradient similarity and better downstream generalization, while maintaining the same pretraining loss as the baseline.

4.3 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).

Table 1: **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**

Nexus achieves Same Pretraining Loss, Better Downstream Task. As detailed in Tab. 1, 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. 11). 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.

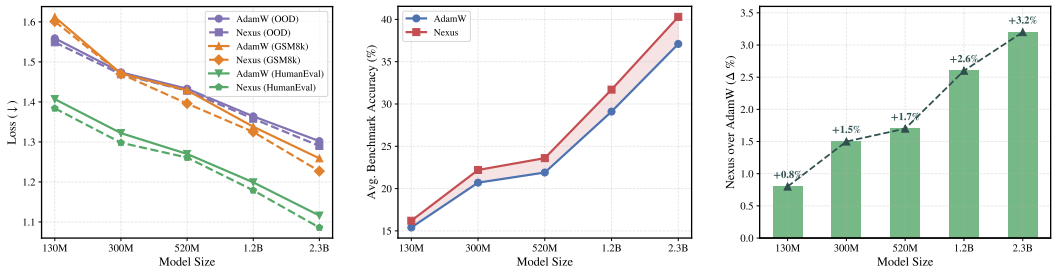
Output Analysis. Compared to the AdamW baseline, Nexus improves accuracy by 15.0% on GSM8k, 8.0% on MATH, and 4.0% on HumanEval. To investigate these improvements, we analyze the model outputs on these benchmarks. We observe that the set of correctly answered questions by Nexus is almost a strict superset 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.

4.4 Scaling Analysis on Model Size

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 [37, 38, 39, 40, 41, 42, 43]. Since Nexus operates via such implicit bias, we hypothesize that its downstream benefits will also amplify at larger scales.

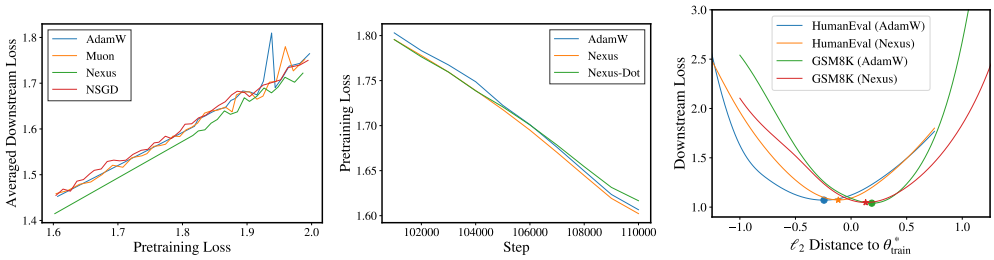
Settings. We evaluate models across five distinct sizes ranging from 130M to 2.3B as outlined in Sec. 4.1. To evaluate scalability with respect to tokens, we extend the training duration of the 3B model from the standard 2× Chinchilla optimal token count to 4× Chinchilla optimal. Please refer to Appendix D.2 for the detailed hyperparameters of each experiment.

Universal "Same Pretraining Loss, Better Downstream". As shown in Tab. 10 and Fig. 6, Nexus consistently maintains the pretraining loss within a negligible margin (defined as $\Delta < 0.01$ in Sec. 4.1) 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.



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

Figure 6: **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.



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

Figure 7: **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.

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.

4.5 Scaling Analysis on Training Tokens

The advantage of Nexus does not diminish with more training tokens. As shown in Tab. 9, the overall performance gap between Nexus and AdamW does not shrink with more tokens; Nexus at 4× Chinchilla achieves an average accuracy of 41.7, effectively maintaining and even slightly widening its substantial lead over the baseline.

5 Discussions

In this section, we conduct several interesting ablation studies of Nexus.

5.1 Training Dynamic of Nexus

Settings. We analyze the training trajectories of the 3B AdamW and 3B Nexus models from Sec. 4.3. 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. 2, 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 3.2.

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, $\text{CosSim}(\nabla \mathcal{L}_{\text{train}}, \nabla \mathcal{L}_{\mathcal{T}})$.

Table 2: **Gradient Similarity, Loss, and Benchmarks.** Nexus achieves higher gradient similarity between the pretraining and downstream corpus. Consistent with Eq. (4), 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

Downstream Closeness Yields Smaller Downstream Loss and Better Performance. Since this gradient similarity $\mathrm{CosSim}(\nabla\mathcal{L}_{\text{train}}, \nabla\mathcal{L}_{\mathcal{T}})$ generalizes, optimizing the pretraining objective inherently optimizes the downstream tasks, as indicated by the first-order approximation in Eq. (4). This gradient closeness translates into lower downstream losses and better benchmark performance.

Empirical Landscapes Align with Fig. 2. As shown in Fig. 7c, 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 2.1 and B.1. While Nexus reduces this distance, it does not cause all minima to nearly intersect as depicted in Fig. 2b—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.

5.2 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 Sec. 4.3. We plot the averaged downstream loss (y-axis) against the pretraining validation loss (x-axis) at corresponding checkpoints. The results are presented in Fig. 7a.

Muon does not possess a superior implicit bias. As illustrated in Fig. 7a, 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. [7].

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 Fig. 7a. This variant still fails to introduce any favorable implicit bias, which closely overlaps with the standard Adam baseline.

6 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. (7)) or complex interactions with Muon's Newton-Schulz iterations. Resolving this challenge remains an active direction for future work.

Acknowledgement

This work was supported by the National Natural Science Foundation of China (NSFC) under Grant No. 625B2104. Additional support was provided by the NSFC Special Fund and Major Program under Grants 62550004 and 62595773, and the Fundamental and Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China (No. JYB2025XDXM101).

The authors declare no other competing interests.

This work was conducted for research and validation purposes only. The algorithms and methodologies described herein are experimental prototypes and have not been integrated into any commercial products or services of the affiliated organizations.

We thank Kaiyue Wen, Jiacheng You, Haodong Wen, Yan Wu, Jianhui Duan, and Chengyin Xu for their insightful comments and helpful discussions.

References

[1] 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.

[2] 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.

[3] 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.

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

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

[6] 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.

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

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

[9] 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/.

[10] 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.

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

[12] 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.

[13] 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.

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

[15] 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.

[16] 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.

[17] 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.

[18] 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.

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

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

[21] 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.

[22] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.

[23] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[24] 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.

[25] Jungmin Kwon, Jeongseop Kim, Hyunseo Park, and In Kwon Choi. Asam: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks. In *International Conference on Machine Learning*, pages 5905–5914, 2021.

[26] Yihao Zhang, Hangzhou He, Jingyu Zhu, Huanran Chen, Yifei Wang, and Zeming Wei. On the duality between sharpness-aware minimization and adversarial training. *arXiv preprint arXiv:2402.15152*, 2024.

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

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

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

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

[31] 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.

[32] 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.

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

[34] 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.

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

[36] 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.

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

[38] 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.

[39] 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.

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

[41] 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.

[42] 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.

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

[44] 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.

[45] 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.

[46] 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.

[47] 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.

[48] 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.

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

[50] 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.

[51] 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.

[52] 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.

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

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

[55] 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.

[56] 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.

[57] Lei Wu, Chao Ma, and Weinan E. How sgd selects the global minima in over-parameterized learning: A dynamical stability perspective. *Advances in Neural Information Processing Systems*, 2018.

[58] 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.

[59] Ishaan Watts, Catherine Li, Sachin Goyal, Jacob Mitchell Springer, and Aditi Raghunathan. Sharpness-aware pretraining mitigates catastrophic forgetting. *arXiv preprint arXiv:2605.02105*, 2026.

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

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

[62] 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.

[63] 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.

[64] 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.

[65] 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.

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

[67] 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.

[68] 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.

[69] 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.

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

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

[72] 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.

[73] 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.

[74] 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.

[75] 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.

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

[77] 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.

[78] 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.

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

[80] 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.

[81] 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.

[82] 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.

[83] 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.

[84] 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.

[85] 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.

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

[87] 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.

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

[89] 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.

[90] 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.

[91] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

[92] Stanisław Jastrzębski, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. Three factors influencing minima in sgd. *arXiv preprint arXiv:1711.04623*, 2017.

[93] Samuel L Smith and Quoc V Le. A bayesian perspective on generalization and stochastic gradient descent. In *International Conference on Learning Representations*, 2018.

[94] Dominic Masters and Carlo Luschi. Revisiting small batch training for deep neural networks. *arXiv preprint arXiv:1804.07612*, 2018.

A Preliminaries

A.1 Related Work

Pretraining Optimizers. Various optimizers have been proposed for LLM pretraining, such as AdamW [8, 27], Muon [9], Lion [44], SOAP [45], Sophia [46], MARS [47], Adam-mini [48], Cautious [49], and Scion [50]. 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 [42, 43, 51, 52, 53, 54, 55, 56, 17]. 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 [57, 58]. A concurrent work [59] 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 [60, 61, 62, 63, 64] and decentralized learning [65, 66, 67, 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 3.2 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 algorithms from the meta-learning literature, such as meta-optimizers [69, 70], hypernetworks [71], and meta-learned data weighting schemes [72, 73], 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. 3.

A.3 Assumptions

The main assumptions used in our analysis are outlined below [74, 13]. Additional assumptions required for specific analyses will be stated in the respective theorems.

Assumption A.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 A.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

Table 3: 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: $\text{CosSim}(\nabla\mathcal{L}_i(\boldsymbol{\theta}), \nabla\mathcal{L}_j(\boldsymbol{\theta}))$.

trajectories [24, 75, 37]. 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 A.3 (Hessian Lipschitz Continuous)**.** The Hessian matrix is ρ-Lipschitz continuous. For any parameters $\boldsymbol{x}, \boldsymbol{y}$:

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

This assumption is necessary to bound the Jacobian of the normalized gradient during the second-order Taylor expansion in Nexus's inner loop.

B Proofs of Main Theorems

B.1 Proofs for Closeness Improving Generalization

B.1.1 Proof for Theorem 2.1

Proof. Fix a basin $\mathcal{B}$. By the stationarity condition of the quadratic pretraining loss, the converged parameter is the mean of the task-specific minimizers:

$$\boldsymbol{\theta}^*_{\text{train},\mathcal{B}} = \frac{1}{K}\sum_{k=1}^{K}\boldsymbol{\theta}^*_k. \tag{9}$$

Since the task-specific minimizers are i.i.d. with mean $\boldsymbol{\mu}_{\mathcal{B}}$ and variance $\sigma^2_{\mathcal{B}}$, we have

$$\mathbb{E}_{\{\mathcal{L}_k\}}\left[\|\boldsymbol{\theta}^*_{\text{train},\mathcal{B}} - \boldsymbol{\mu}_{\mathcal{B}}\|_2^2\right] = \frac{\sigma^2_{\mathcal{B}}}{K}. \tag{10}$$

Using the variance decomposition,

$$\frac{1}{K}\sum_{k=1}^{K}\|\boldsymbol{\theta}^*_{\text{train},\mathcal{B}} - \boldsymbol{\theta}^*_k\|_2^2 = \frac{1}{K}\sum_{k=1}^{K}\|\boldsymbol{\theta}^*_k - \boldsymbol{\mu}_{\mathcal{B}}\|_2^2 - \|\boldsymbol{\theta}^*_{\text{train},\mathcal{B}} - \boldsymbol{\mu}_{\mathcal{B}}\|_2^2. \tag{11}$$

Therefore, the expected pretraining loss is

$$C_{\text{train}} = \mathbb{E}_{\{\mathcal{L}_k\}}\left[\mathcal{L}_{\text{train}}(\boldsymbol{\theta}^*_{\text{train},\mathcal{B}})\right] = c_{\mathcal{B}} + \frac{a}{2}\left(1 - \frac{1}{K}\right)\sigma^2_{\mathcal{B}}. \tag{12}$$

For an independent downstream task $\mathcal{T} \sim \mathcal{P}$, we have

$$\mathbb{E}_{\{\mathcal{L}_k\},\mathcal{T}}\left[\mathcal{L}_{\mathcal{T}}(\boldsymbol{\theta}^*_{\text{train},\mathcal{B}})\right] = c_{\mathcal{B}} + \frac{a}{2}\mathbb{E}_{\{\mathcal{L}_k\},\mathcal{T}}\left[\|\boldsymbol{\theta}^*_{\text{train},\mathcal{B}} - \boldsymbol{\theta}^*_{\mathcal{T}}\|_2^2\right] = c_{\mathcal{B}} + \frac{a}{2}\left(1 + \frac{1}{K}\right)\sigma^2_{\mathcal{B}}. \tag{13}$$

Substituting Eq. (12) yields

$$\mathbb{E}_{\{\mathcal{L}_k\},\mathcal{T}}\left[\mathcal{L}_{\mathcal{T}}(\boldsymbol{\theta}^*_{\text{train},\mathcal{B}})\right] = C_{\text{train}} + \frac{a}{2}\left[\left(1 + \frac{1}{K}\right) - \left(1 - \frac{1}{K}\right)\right]\sigma^2_{\mathcal{B}} = C_{\text{train}} + \frac{a}{K}\sigma^2_{\mathcal{B}}. \tag{14}$$

Thus, for basins achieving the same expected pretraining loss, the expected downstream loss increases linearly with the variance of task-specific minimizers. □

B.1.2 Proof for Theorem B.1

Theorem B.1 (Generalization of Closeness beyond Quadratics, Proof in Appendix B.1.2)**.** *Let $\{\mathcal{L}_k\}_{k=1}^K$ and $\mathcal{L}_{\mathcal{T}}$ be i.i.d. samples from a task distribution $\mathcal{P}$. Let $\boldsymbol{\theta}^*$ be a fixed local minimizer of the population loss $\mathbb{E}_{\mathcal{L}\sim\mathcal{P}}[\mathcal{L}(\boldsymbol{\theta})]$, and let $\boldsymbol{\theta}^*_{\mathcal{L}} = \arg\min_{\boldsymbol{\theta}\in\mathcal{S}_{\mathcal{L}}}\|\boldsymbol{\theta} - \boldsymbol{\theta}^*\|_2$ be the corresponding task-specific local minimizer, assumed to exist with a measurable selection. Let $\boldsymbol{\theta}^*_{\text{train}}$ be a stationary local minimizer of the empirical pretraining loss.*

Assume that the losses are twice continuously differentiable along the relevant segments, and satisfy the following local curvature conditions, uniformly over tasks and almost surely over training samples:

- *Along $[\boldsymbol{\theta}^*_{\mathcal{L}}, \boldsymbol{\theta}^*]$, we have $\|\nabla^2\mathcal{L}(\boldsymbol{\xi})\|_2 \le \lambda_{\max}$ and $\boldsymbol{u}^\top\nabla^2\mathcal{L}(\boldsymbol{\xi})\boldsymbol{u} \ge \lambda_{\min} > 0$ for every unit vector $\boldsymbol{u} \in \text{span}\{\boldsymbol{\theta}^* - \boldsymbol{\theta}^*_{\mathcal{L}} \mid \mathcal{L} \sim \mathcal{P}\}$.*
- *Along $[\boldsymbol{\theta}^*, \boldsymbol{\theta}^*_{\text{train}}]$, the same spectral norm bound holds for every task, while the directional curvature lower bound holds along $\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\theta}^*$ whenever this displacement is nonzero.*

Let $\boldsymbol{\mu} = \mathbb{E}[\boldsymbol{\theta}^*_{\mathcal{L}}]$, $\sigma^2 = \mathbb{E}[\|\boldsymbol{\theta}^*_{\mathcal{L}} - \boldsymbol{\mu}\|_2^2]$, and $\kappa = \lambda_{\max}/\lambda_{\min}$. Assume the required expectations are finite and differentiation commutes with expectation at $\boldsymbol{\theta}^*$.

Defining $C_{\text{train}} = \mathbb{E}_{\{\mathcal{L}_k\}}[\mathcal{L}_{\text{train}}(\boldsymbol{\theta}^*_{\text{train}})]$, the expected downstream generalization gap satisfies

$$\mathbb{E}_{\{\mathcal{L}_k\},\mathcal{T}}[\mathcal{L}_{\mathcal{T}}(\boldsymbol{\theta}^*_{\text{train}})] - C_{\text{train}} \le \frac{\lambda_{\max}\kappa(\kappa+1)}{2K}\left(1 + \frac{(\kappa-1)^2}{4}\right)\sigma^2. \tag{15}$$

We show that the expected generalization gap is controlled by the variance of task-specific minimizers.

Step 1: Closeness Bounds Gradient Variance. Since $\nabla\mathcal{L}(\boldsymbol{\theta}^*_{\mathcal{L}}) = \mathbf{0}$, the fundamental theorem of calculus gives

$$\nabla\mathcal{L}(\boldsymbol{\theta}^*) = \mathbf{H}_{\mathcal{L}}(\boldsymbol{\theta}^* - \boldsymbol{\theta}^*_{\mathcal{L}}), \qquad \mathbf{H}_{\mathcal{L}} := \int_0^1 \nabla^2\mathcal{L}\left(\boldsymbol{\theta}^*_{\mathcal{L}} + s(\boldsymbol{\theta}^* - \boldsymbol{\theta}^*_{\mathcal{L}})\right) ds. \tag{16}$$

By population stationarity, $\mathbb{E}_{\mathcal{L}}[\nabla\mathcal{L}(\boldsymbol{\theta}^*)] = \mathbf{0}$. Therefore,

$$\mathbb{E}_{\mathcal{L}}[\mathbf{H}_{\mathcal{L}}(\boldsymbol{\theta}^* - \boldsymbol{\mu})] = \mathbb{E}_{\mathcal{L}}[\mathbf{H}_{\mathcal{L}}(\boldsymbol{\theta}^*_{\mathcal{L}} - \boldsymbol{\mu})]. \tag{17}$$

The curvature bounds hold on the span containing both $\boldsymbol{\theta}^* - \boldsymbol{\mu}$ and $\boldsymbol{\theta}^*_{\mathcal{L}} - \boldsymbol{\mu}$. Using $\mathbb{E}[\boldsymbol{\theta}^*_{\mathcal{L}} - \boldsymbol{\mu}] = \mathbf{0}$, we obtain

$$\begin{aligned}
\lambda_{\min}\|\boldsymbol{\theta}^* - \boldsymbol{\mu}\|_2^2 &\le (\boldsymbol{\theta}^* - \boldsymbol{\mu})^\top \mathbb{E}_{\mathcal{L}}[\mathbf{H}_{\mathcal{L}}(\boldsymbol{\theta}^*_{\mathcal{L}} - \boldsymbol{\mu})] \\
&= (\boldsymbol{\theta}^* - \boldsymbol{\mu})^\top \mathbb{E}_{\mathcal{L}}\left[\left(\mathbf{H}_{\mathcal{L}} - \frac{\lambda_{\max} + \lambda_{\min}}{2}\mathbf{I}\right)(\boldsymbol{\theta}^*_{\mathcal{L}} - \boldsymbol{\mu})\right] \\
&\le \frac{\lambda_{\max} - \lambda_{\min}}{2}\|\boldsymbol{\theta}^* - \boldsymbol{\mu}\|_2 \mathbb{E}_{\mathcal{L}}[\|\boldsymbol{\theta}^*_{\mathcal{L}} - \boldsymbol{\mu}\|_2] \\
&\le \frac{\lambda_{\max} - \lambda_{\min}}{2}\|\boldsymbol{\theta}^* - \boldsymbol{\mu}\|_2\, \sigma.
\end{aligned} \tag{18}$$

Here the penultimate inequality uses the spectral bounds of the symmetric Hessian compressed to the displacement span. Hence,

$$\|\boldsymbol{\theta}^* - \boldsymbol{\mu}\|_2^2 \le \frac{(\kappa - 1)^2}{4}\sigma^2. \tag{19}$$

Using the integral representation and the Hessian spectral norm bound,

$$\mathbb{E}_{\mathcal{L}}[\|\nabla\mathcal{L}(\boldsymbol{\theta}^*)\|_2^2] \le \lambda_{\max}^2 \mathbb{E}_{\mathcal{L}}[\|\boldsymbol{\theta}^* - \boldsymbol{\theta}^*_{\mathcal{L}}\|_2^2] = \lambda_{\max}^2\left(\sigma^2 + \|\boldsymbol{\theta}^* - \boldsymbol{\mu}\|_2^2\right) \le \lambda_{\max}^2\left(1 + \frac{(\kappa - 1)^2}{4}\right)\sigma^2. \tag{20}$$

Step 2: Bounding the Estimation Error. By stationarity of the empirical minimizer and directional strong convexity along $[\boldsymbol{\theta}^*, \boldsymbol{\theta}^*_{\text{train}}]$,

$$\lambda_{\min}\|\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\theta}^*\|_2^2 \le -\left\langle \nabla\mathcal{L}_{\text{train}}(\boldsymbol{\theta}^*), \boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\theta}^*\right\rangle \le \|\nabla\mathcal{L}_{\text{train}}(\boldsymbol{\theta}^*)\|_2 \|\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\theta}^*\|_2. \tag{21}$$

Because the training tasks are i.i.d. and $\mathbb{E}_{\mathcal{L}}[\nabla\mathcal{L}(\boldsymbol{\theta}^*)] = \mathbf{0}$, their gradient cross-terms vanish in expectation. Consequently,

$$\mathbb{E}_{\{\mathcal{L}_k\}}[\|\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\theta}^*\|_2^2] \le \frac{1}{\lambda_{\min}^2}\mathbb{E}_{\{\mathcal{L}_k\}}[\|\nabla\mathcal{L}_{\text{train}}(\boldsymbol{\theta}^*)\|_2^2] = \frac{1}{K\lambda_{\min}^2}\mathbb{E}_{\mathcal{L}}[\|\nabla\mathcal{L}(\boldsymbol{\theta}^*)\|_2^2]. \tag{22}$$

Step 3: Bounding the Generalization Gap. By independence and identical distribution of the training and downstream tasks,

$$\begin{aligned}
\mathbb{E}_{\{\mathcal{L}_k\},\mathcal{T}}[\mathcal{L}_{\mathcal{T}}(\boldsymbol{\theta}^*_{\text{train}})] - C_{\text{train}} &= \mathbb{E}_{\{\mathcal{L}_k\},\mathcal{T}}\left[\mathcal{L}_{\mathcal{T}}(\boldsymbol{\theta}^*_{\text{train}}) - \mathcal{L}_{\mathcal{T}}(\boldsymbol{\theta}^*)\right] \\
&\quad + \mathbb{E}_{\{\mathcal{L}_k\}}\left[\mathcal{L}_{\text{train}}(\boldsymbol{\theta}^*) - \mathcal{L}_{\text{train}}(\boldsymbol{\theta}^*_{\text{train}})\right].
\end{aligned} \tag{23}$$

For the first term, local smoothness and population stationarity imply

$$\mathbb{E}_{\{\mathcal{L}_k\},\mathcal{T}}\left[\mathcal{L}_{\mathcal{T}}(\boldsymbol{\theta}^*_{\text{train}}) - \mathcal{L}_{\mathcal{T}}(\boldsymbol{\theta}^*)\right] \le \frac{\lambda_{\max}}{2}\mathbb{E}_{\{\mathcal{L}_k\}}[\|\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\theta}^*\|_2^2]. \tag{24}$$

For the second term, directional strong convexity gives

$$\begin{aligned}
\mathcal{L}_{\text{train}}(\boldsymbol{\theta}^*) - \mathcal{L}_{\text{train}}(\boldsymbol{\theta}^*_{\text{train}}) &\le -\left\langle \nabla\mathcal{L}_{\text{train}}(\boldsymbol{\theta}^*), \boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\theta}^*\right\rangle - \frac{\lambda_{\min}}{2}\|\boldsymbol{\theta}^*_{\text{train}} - \boldsymbol{\theta}^*\|_2^2 \\
&\le \frac{1}{2\lambda_{\min}}\|\nabla\mathcal{L}_{\text{train}}(\boldsymbol{\theta}^*)\|_2^2,
\end{aligned} \tag{25}$$

where the last inequality follows by completing the square.

Combining these bounds with Eqs. (20) and (22) yields

$$\begin{aligned}\mathbb{E}_{\{\mathcal{L}_k\},\mathcal{T}}[\mathcal{L}_{\mathcal{T}}(\boldsymbol{\theta}^*_{\text{train}})] - C_{\text{train}} &\leq \left(\frac{\lambda_{\max}}{2\lambda_{\min}^2} + \frac{1}{2\lambda_{\min}}\right)\frac{1}{K}\mathbb{E}_{\mathcal{L}}[\|\nabla\mathcal{L}(\boldsymbol{\theta}^*)\|_2^2] \\ &\leq \frac{\lambda_{\max}\kappa(\kappa+1)}{2K}\left(1 + \frac{(\kappa-1)^2}{4}\right)\sigma^2.\end{aligned} \tag{26}$$

Thus, for fixed local curvature parameters, the expected generalization gap is bounded by $O(\sigma^2/K)$.

B.2 Proof of Theorem 3.1

Proof. We prove the bound in three steps: relating parameter closeness to gradient norms, applying the stationarity condition, and bounding the resulting inner products using cosine similarity.

Step 1: Relating Closeness to Gradient Norm. Since $\boldsymbol{\theta}_k^*$ is a local minimizer, $\nabla \mathcal{L}_k(\boldsymbol{\theta}_k^*) = \mathbf{0}$. By the fundamental theorem of calculus,

$$\begin{aligned} \boldsymbol{u}_k^\top \nabla \mathcal{L}_k(\boldsymbol{\theta}) &= \boldsymbol{u}_k^\top \left(\nabla \mathcal{L}_k(\boldsymbol{\theta}) - \nabla \mathcal{L}_k(\boldsymbol{\theta}_k^*)\right) \\ &= \|\boldsymbol{\theta} - \boldsymbol{\theta}_k^*\|_2 \int_0^1 \boldsymbol{u}_k^\top \nabla^2 \mathcal{L}_k\left(\boldsymbol{\theta}_k^* + s(\boldsymbol{\theta} - \boldsymbol{\theta}_k^*)\right) \boldsymbol{u}_k \, ds \\ &\geq \lambda_{\min} \|\boldsymbol{\theta} - \boldsymbol{\theta}_k^*\|_2, \end{aligned} \tag{27}$$

where the last inequality follows from the directional curvature lower bound in the theorem.

By Cauchy–Schwarz,

$$\|\boldsymbol{\theta} - \boldsymbol{\theta}_k^*\|_2 \leq \frac{1}{\lambda_{\min}} \|\nabla \mathcal{L}_k(\boldsymbol{\theta})\|_2. \tag{28}$$

Squaring and averaging over all K tasks yields

$$\frac{1}{K} \sum_{k=1}^K \|\boldsymbol{\theta} - \boldsymbol{\theta}_k^*\|_2^2 \leq \frac{1}{K \lambda_{\min}^2} \sum_{k=1}^K \|\nabla \mathcal{L}_k(\boldsymbol{\theta})\|_2^2. \tag{29}$$

Step 2: Force Balance Decomposition. Since $\boldsymbol{\theta}$ satisfies the stationarity condition $\sum_{k=1}^K \nabla \mathcal{L}_k(\boldsymbol{\theta}) = \mathbf{0}$, we have

$$0 = \left\| \sum_{k=1}^K \nabla \mathcal{L}_k(\boldsymbol{\theta}) \right\|_2^2 = \sum_{k=1}^K \|\nabla \mathcal{L}_k(\boldsymbol{\theta})\|_2^2 + \sum_{i \neq j} \nabla \mathcal{L}_i(\boldsymbol{\theta})^\top \nabla \mathcal{L}_j(\boldsymbol{\theta}). \tag{30}$$

Therefore,

$$\sum_{k=1}^K \|\nabla \mathcal{L}_k(\boldsymbol{\theta})\|_2^2 = -\sum_{i \neq j} \nabla \mathcal{L}_i(\boldsymbol{\theta})^\top \nabla \mathcal{L}_j(\boldsymbol{\theta}). \tag{31}$$

Combining this identity with Eq. (29) gives the first inequality of the theorem.

Step 3: Bounding via Cosine Similarity. Let $\text{CosSim}_{ij} = \text{CosSim}(\nabla \mathcal{L}_i(\boldsymbol{\theta}), \nabla \mathcal{L}_j(\boldsymbol{\theta}))$. For any $i \neq j$, we have

$$\begin{aligned} -\nabla \mathcal{L}_i(\boldsymbol{\theta})^\top \nabla \mathcal{L}_j(\boldsymbol{\theta}) &= \|\nabla \mathcal{L}_i(\boldsymbol{\theta})\|_2 \|\nabla \mathcal{L}_j(\boldsymbol{\theta})\|_2 (1 - \text{CosSim}_{ij}) \\ &\quad - \|\nabla \mathcal{L}_i(\boldsymbol{\theta})\|_2 \|\nabla \mathcal{L}_j(\boldsymbol{\theta})\|_2 \\ &\leq G^2 (1 - \text{CosSim}_{ij}), \end{aligned} \tag{32}$$

where we use $1 - \text{CosSim}_{ij} \geq 0$ and $\|\nabla \mathcal{L}_k(\boldsymbol{\theta})\|_2 \leq G$.

Summing over all ordered pairs $i \neq j$ yields

$$\sum_{i \neq j} \left(-\nabla \mathcal{L}_i(\boldsymbol{\theta})^\top \nabla \mathcal{L}_j(\boldsymbol{\theta})\right) \leq G^2 \sum_{i \neq j} \left(1 - \text{CosSim}(\nabla \mathcal{L}_i(\boldsymbol{\theta}), \nabla \mathcal{L}_j(\boldsymbol{\theta}))\right). \tag{33}$$

Combining this with Step 2 completes the proof.

□

B.3 Implicit Bias of Nexus Optimizer

In this appendix, we provide the detailed proofs for Theorem 3.2. We rigorously analyze the update dynamics of Algorithm 3 using second-order Taylor expansions and derive the precise form of the implicit optimization objective with explicit non-asymptotic error bounds.

B.3.1 Preliminaries and Notation

Let $\mathcal{L}_i : \mathbb{R}^d \to \mathbb{R}$ denote the loss function for the i-th task, where $i \in \{1, \dots, k\}$. 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:

$$S_{ij}(\boldsymbol{\theta}) \triangleq \text{CosSim}(\nabla\mathcal{L}_i(\boldsymbol{\theta}), \nabla\mathcal{L}_j(\boldsymbol{\theta})) = \frac{\nabla\mathcal{L}_i(\boldsymbol{\theta})^\top \nabla\mathcal{L}_j(\boldsymbol{\theta})}{\|\nabla\mathcal{L}_i(\boldsymbol{\theta})\|_2 \|\nabla\mathcal{L}_j(\boldsymbol{\theta})\|_2}. \tag{34}$$

Algorithm 3 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 $m \in \{1, \dots, k\}$, a task index s_m is sampled uniformly from $\{1, \dots, k\}$. The update rule is:

$$\boldsymbol{\theta}_{t,m} = \boldsymbol{\theta}_{t,m-1} - \gamma \frac{\nabla\mathcal{L}_{s_m}(\boldsymbol{\theta}_{t,m-1})}{\|\nabla\mathcal{L}_{s_m}(\boldsymbol{\theta}_{t,m-1})\|_2}. \tag{35}$$

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

B.3.2 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: $0 < G_{\min} \le \|\nabla\mathcal{L}_i(\boldsymbol{\theta})\|_2$.
- **Assumption 2 (Smoothness):** The loss $\mathcal{L}_i$ is L-smooth, i.e., $\|\nabla^2\mathcal{L}_i(\boldsymbol{\theta})\|_2 \le L$.
- **Assumption 3 (Hessian Lipschitz):** The Hessian is ρ-Lipschitz continuous, i.e., $\|\nabla^2\mathcal{L}_i(\boldsymbol{x}) - \nabla^2\mathcal{L}_i(\boldsymbol{y})\|_2 \le \rho\|\boldsymbol{x} - \boldsymbol{y}\|_2$.

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:
$$\left\| \frac{\nabla\mathcal{L}_i(\boldsymbol{x})}{\|\nabla\mathcal{L}_i(\boldsymbol{x})\|_2} - \frac{\nabla\mathcal{L}_i(\boldsymbol{y})}{\|\nabla\mathcal{L}_i(\boldsymbol{y})\|_2} \right\|_2 \le L_1\|\boldsymbol{x} - \boldsymbol{y}\|_2. \tag{36}$$
2. The Jacobian of the normalized gradient is L_2-Lipschitz continuous:
$$\|\boldsymbol{J}_i(\boldsymbol{x}) - \boldsymbol{J}_i(\boldsymbol{y})\|_2 \le L_2\|\boldsymbol{x} - \boldsymbol{y}\|_2, \tag{37}$$
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 to 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:

$$\boldsymbol{J}_i(\boldsymbol{\theta}) = \frac{1}{\|\nabla\mathcal{L}_i\|_2} \left(\boldsymbol{I} - \frac{\nabla\mathcal{L}_i \nabla\mathcal{L}_i^\top}{\|\nabla\mathcal{L}_i\|_2^2} \right) \nabla^2\mathcal{L}_i(\boldsymbol{\theta}). \tag{38}$$

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

$$L_1 \le \sup_{\boldsymbol{\theta}} \|\boldsymbol{J}_i(\boldsymbol{\theta})\|_2 \le \frac{1}{G_{\min}} \cdot 1 \cdot L = \frac{L}{G_{\min}}. \tag{39}$$

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 $\boldsymbol{H}_i(\boldsymbol{\theta})$:

$$\boldsymbol{J}_i(\boldsymbol{\theta}) = \underbrace{\|\nabla\mathcal{L}_i(\boldsymbol{\theta})\|_2^{-1}}_{u(\boldsymbol{\theta})} \cdot \underbrace{\left(\boldsymbol{I} - \frac{\nabla\mathcal{L}_i \nabla\mathcal{L}_i^\top}{\|\nabla\mathcal{L}_i\|_2^2} \right)}_{\boldsymbol{\Pi}(\boldsymbol{\theta})} \cdot \underbrace{\nabla^2\mathcal{L}_i(\boldsymbol{\theta})}_{\boldsymbol{H}_i(\boldsymbol{\theta})}. \tag{40}$$

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:

$$\begin{aligned} L_2 &\leq L_u M_\Pi M_H + M_u L_\Pi M_H + M_u M_\Pi L_H \\ &\leq \left(\frac{L}{G_{\min}^2} \cdot 1 \cdot L\right) + \left(\frac{1}{G_{\min}} \cdot \frac{2L}{G_{\min}} \cdot L\right) + \left(\frac{1}{G_{\min}} \cdot 1 \cdot \rho\right) \\ &= \frac{L^2}{G_{\min}^2} + \frac{2L^2}{G_{\min}^2} + \frac{\rho}{G_{\min}}. \end{aligned} \tag{41}$$

Combining terms yields the final constant:

$$L_2 = \frac{3L^2 + \rho G_{\min}}{G_{\min}^2}. \tag{42}$$

B.3.3 Derivation of the Update Direction

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

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:

$$\boldsymbol{J}_{s_m}(\boldsymbol{\theta}) = \frac{1}{\|\nabla \mathcal{L}_{s_m}(\boldsymbol{\theta})\|_2} \left(\boldsymbol{I} - \frac{\nabla \mathcal{L}_{s_m}(\boldsymbol{\theta}) \nabla \mathcal{L}_{s_m}(\boldsymbol{\theta})^\top}{\|\nabla \mathcal{L}_{s_m}(\boldsymbol{\theta})\|_2^2}\right) \nabla^2 \mathcal{L}_{s_m}(\boldsymbol{\theta}). \tag{43}$$

Applying Taylor's theorem with the Lagrange remainder form:

$$\frac{\nabla \mathcal{L}_{s_m}(\boldsymbol{\theta}_{t,m-1})}{\|\nabla \mathcal{L}_{s_m}(\boldsymbol{\theta}_{t,m-1})\|_2} = \frac{\nabla \mathcal{L}_{s_m}(\boldsymbol{\theta}_{t,0})}{\|\nabla \mathcal{L}_{s_m}(\boldsymbol{\theta}_{t,0})\|_2} + \boldsymbol{J}_{s_m}(\boldsymbol{\theta}_{t,0}) \Delta\boldsymbol{\theta}_{m-1} + \boldsymbol{r}_m. \tag{44}$$

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

$$\|\boldsymbol{r}_m\|_2 \leq \frac{L_2}{2} \|\Delta\boldsymbol{\theta}_{m-1}\|_2^2. \tag{45}$$

Step 2: Recursive Substitution The displacement $\Delta\boldsymbol{\theta}_{m-1}$ is the sum of previous updates. Using the zeroth-order approximation:

$$\Delta\boldsymbol{\theta}_{m-1} = \sum_{l=1}^{m-1} (\boldsymbol{\theta}_{t,l} - \boldsymbol{\theta}_{t,l-1}) = -\gamma \sum_{l=1}^{m-1} \frac{\nabla \mathcal{L}_{s_l}(\boldsymbol{\theta}_{t,l-1})}{\|\nabla \mathcal{L}_{s_l}(\boldsymbol{\theta}_{t,l-1})\|_2}. \tag{46}$$

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:

$$\begin{aligned}\left\| \frac{\nabla \mathcal{L}_{s_l}(\boldsymbol{\theta}_{t,l-1})}{\|\nabla \mathcal{L}_{s_l}(\boldsymbol{\theta}_{t,l-1})\|_2} - \frac{\nabla \mathcal{L}_{s_l}(\boldsymbol{\theta}_{t,0})}{\|\nabla \mathcal{L}_{s_l}(\boldsymbol{\theta}_{t,0})\|_2} \right\|_2 &\leq L_1 \|\boldsymbol{\theta}_{t,l-1} - \boldsymbol{\theta}_{t,0}\|_2 \\ &= L_1 \left\| -\sum_{j=1}^{l-1} \gamma \frac{\nabla \mathcal{L}_{s_l}(\boldsymbol{\theta}_{t,j})}{\|\nabla \mathcal{L}_{s_l}(\boldsymbol{\theta}_{t,j})\|_2} \right\| \\ &\leq L_1 (l-1)\gamma.\end{aligned} \tag{47}$$

Thus, we can write:

$$\Delta \boldsymbol{\theta}_{m-1} = -\gamma \sum_{l=1}^{m-1} \frac{\nabla \mathcal{L}_{s_l}(\boldsymbol{\theta}_{t,0})}{\|\nabla \mathcal{L}_{s_l}(\boldsymbol{\theta}_{t,0})\|_2} + \boldsymbol{\delta}_{m-1}, \tag{48}$$

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

$$\|\boldsymbol{\delta}_{m-1}\|_2 \leq \gamma \sum_{l=1}^{m-1} L_1 (l-1)\gamma = L_1 \gamma^2 \frac{(m-1)(m-2)}{2} \leq \frac{L_1}{2}(m-1)^2\gamma^2. \tag{49}$$

Substituting this expression for $\Delta \boldsymbol{\theta}_{m-1}$ back into Eq. (44):

$$\frac{\nabla \mathcal{L}_{s_m}(\boldsymbol{\theta}_{t,m-1})}{\|\nabla \mathcal{L}_{s_m}(\boldsymbol{\theta}_{t,m-1})\|_2} = \frac{\nabla \mathcal{L}_{s_m}(\boldsymbol{\theta}_{t,0})}{\|\nabla \mathcal{L}_{s_m}(\boldsymbol{\theta}_{t,0})\|_2} - \gamma \sum_{l=1}^{m-1} \boldsymbol{J}_{s_m}(\boldsymbol{\theta}_{t,0}) \frac{\nabla \mathcal{L}_{s_l}(\boldsymbol{\theta}_{t,0})}{\|\nabla \mathcal{L}_{s_l}(\boldsymbol{\theta}_{t,0})\|_2} + \boldsymbol{\mathcal{E}}_m. \tag{50}$$

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 $\|\Delta \boldsymbol{\theta}_{m-1}\|_2 \leq (m-1)\gamma$:

$$\|\boldsymbol{\mathcal{E}}_m\|_2 \leq \frac{L_2}{2}(m-1)^2\gamma^2 + L_1 \left(\frac{L_1}{2}(m-1)^2\gamma^2 \right) = \frac{L_2 + L_1^2}{2}(m-1)^2\gamma^2. \tag{51}$$

Step 3: Aggregation of the Pseudo-Gradient The total pseudo-gradient is $\hat{\boldsymbol{g}}_t = \gamma \sum_{m=1}^{k} \frac{\nabla \mathcal{L}_{s_m}(\boldsymbol{\theta}_{t,m-1})}{\|\nabla \mathcal{L}_{s_m}(\boldsymbol{\theta}_{t,m-1})\|_2}$. Substituting the result from Step 2:

$$\hat{\boldsymbol{g}}_t = \gamma \sum_{m=1}^{k} \frac{\nabla \mathcal{L}_{s_m}}{\|\nabla \mathcal{L}_{s_m}\|_2} - \gamma^2 \sum_{m=1}^{k} \sum_{l=1}^{m-1} \boldsymbol{J}_{s_m} \frac{\nabla \mathcal{L}_{s_l}}{\|\nabla \mathcal{L}_{s_l}\|_2} + \boldsymbol{\mathcal{E}}_{total}. \tag{52}$$

(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:

$$\|\boldsymbol{\mathcal{E}}_{total}\|_2 \leq \gamma \sum_{m=1}^{k} \frac{L_2 + L_1^2}{2}(m-1)^2\gamma^2 \leq \frac{L_2 + L_1^2}{6} k^3 \gamma^3. \tag{53}$$

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:

$$\mathbb{E}[\mathcal{T}_{linear}] = k \cdot \frac{1}{k} \sum_{i=1}^{k} \frac{\nabla \mathcal{L}_i}{\|\nabla \mathcal{L}_i\|_2} = \sum_{i=1}^{k} \frac{\nabla \mathcal{L}_i}{\|\nabla \mathcal{L}_i\|_2}. \tag{54}$$

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:

$$\mathbb{E}[\mathcal{T}_{interact}] = \frac{k-1}{2k} \sum_{i,j} \boldsymbol{J}_i \frac{\nabla \mathcal{L}_j}{\|\nabla \mathcal{L}_j\|_2} = \frac{k-1}{4k} \sum_{i \neq j} \left(\boldsymbol{J}_i \frac{\nabla \mathcal{L}_j}{\|\nabla \mathcal{L}_j\|_2} + \boldsymbol{J}_j \frac{\nabla \mathcal{L}_i}{\|\nabla \mathcal{L}_i\|_2} \right). \tag{55}$$

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

$$\nabla_{\boldsymbol{\theta}} \text{CosSim}(\nabla \mathcal{L}_i, \nabla \mathcal{L}_j) = \boldsymbol{J}_i^\top \frac{\nabla \mathcal{L}_j}{\|\nabla \mathcal{L}_j\|_2} + \boldsymbol{J}_j^\top \frac{\nabla \mathcal{L}_i}{\|\nabla \mathcal{L}_i\|_2}. \tag{56}$$

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

$$\begin{aligned} \boldsymbol{J}_i \frac{\nabla \mathcal{L}_j}{\|\nabla \mathcal{L}_j\|_2} + \boldsymbol{J}_j \frac{\nabla \mathcal{L}_i}{\|\nabla \mathcal{L}_i\|_2} &= \left(\boldsymbol{J}_i^\top \frac{\nabla \mathcal{L}_j}{\|\nabla \mathcal{L}_j\|_2} + \boldsymbol{J}_j^\top \frac{\nabla \mathcal{L}_i}{\|\nabla \mathcal{L}_i\|_2} \right) \\ &\quad + (\boldsymbol{J}_i - \boldsymbol{J}_i^\top) \frac{\nabla \mathcal{L}_j}{\|\nabla \mathcal{L}_j\|_2} + (\boldsymbol{J}_j - \boldsymbol{J}_j^\top) \frac{\nabla \mathcal{L}_i}{\|\nabla \mathcal{L}_i\|_2} \\ &= \nabla_{\boldsymbol{\theta}} \text{CosSim}(\nabla \mathcal{L}_i, \nabla \mathcal{L}_j) + \boldsymbol{\mathcal{E}}_{\text{sym},i,j} + \boldsymbol{\mathcal{E}}_{\text{sym},j,i}, \end{aligned} \tag{57}$$

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

$$\boldsymbol{\mathcal{E}}_{\text{sym},i,j} = \left(\nabla \frac{\nabla \mathcal{L}_i}{\|\nabla \mathcal{L}_i\|_2} - \left(\nabla \frac{\nabla \mathcal{L}_i}{\|\nabla \mathcal{L}_i\|_2} \right)^\top \right) \frac{\nabla \mathcal{L}_j}{\|\nabla \mathcal{L}_j\|_2}. \tag{58}$$

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 [29, 30, 31, 32]. 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:

$$\mathbb{E}[\mathcal{T}_{interact}] = \frac{k-1}{4k} \sum_{i \neq j} \left(\nabla_{\boldsymbol{\theta}} \text{CosSim}(\nabla \mathcal{L}_i, \nabla \mathcal{L}_j) + \boldsymbol{\mathcal{E}}_{\text{sym},i,j} + \boldsymbol{\mathcal{E}}_{\text{sym},j,i} \right). \tag{59}$$

B.3.4 Proof Conclusion

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

$$\begin{aligned} \mathbb{E}[\hat{\boldsymbol{g}}_t] &= \gamma \sum_{i=1}^{k} \frac{\nabla \mathcal{L}_i(\boldsymbol{\theta}_{t,0})}{\|\nabla \mathcal{L}_i(\boldsymbol{\theta}_{t,0})\|_2} \\ &\quad - \gamma^2 \frac{k-1}{4k} \sum_{i \neq j} \left(\nabla_{\boldsymbol{\theta}} \text{CosSim}(\nabla \mathcal{L}_i, \nabla \mathcal{L}_j) + \boldsymbol{\mathcal{E}}_{\text{sym},i,j} + \boldsymbol{\mathcal{E}}_{\text{sym},j,i} \right) + \boldsymbol{\mathcal{E}}_{total}. \end{aligned} \tag{60}$$

Substituting the constants derived in Appendix B.3.2, the residual is bounded by:

$$\|\boldsymbol{\mathcal{E}}_{total}\|_2 \leq \frac{1}{6} \left(\frac{\rho G_{\min} + 4L^2}{G_{\min}^2} \right) k^3 \gamma^3 = \mathcal{O}(\gamma^3). \tag{61}$$

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. □

C Additional Theoretical Results

C.1 A Gradient-Flow Interpretation of Downstream Generalization

Motivation. In Eq. (4), we show that gradient similarity determines the first-order decrease in downstream loss after a single gradient descent step on the pretraining objective. A natural question is whether we can extend this intuition to the entire training trajectory: **if gradient similarity improves by δ, how much additional downstream loss reduction can we expect at the same pretraining loss?**

Gradient-flow interpretation. Consider the continuous-time gradient flow of the pretraining objective:

$$\dot{\theta}(t) = -\eta(t)\nabla L_{\text{train}}(\theta(t)). \tag{62}$$

By the chain rule, the changes in downstream and pretraining losses satisfy

$$\begin{aligned} L_{\mathcal{T}}(T) - L_{\mathcal{T}}(0) &= -\int_0^T \eta(t)\nabla L_{\mathcal{T}}(t)^\top \nabla L_{\text{train}}(t)\, dt, \\ L_{\text{train}}(T) - L_{\text{train}}(0) &= -\int_0^T \eta(t)\|\nabla L_{\text{train}}(t)\|_2^2\, dt. \end{aligned} \tag{63}$$

Since the pretraining loss decreases monotonically under gradient flow, we can use L_{train} instead of time t to parameterize the training trajectory. This gives

$$L_{\mathcal{T}}(T) - L_{\mathcal{T}}(0) = \int_{L_{\text{train}}(0)}^{L_{\text{train}}(T)} \frac{\nabla L_{\mathcal{T}}^\top \nabla L_{\text{train}}}{\|\nabla L_{\text{train}}\|_2^2}\, dL_{\text{train}}. \tag{64}$$

This equation provides a quantitative interpretation of gradient alignment: **the normalized gradient dot product determines how much downstream loss decreases per unit reduction in pretraining loss.**

Implication for downstream generalization. Suppose the normalized gradient dot product improves by a constant δ throughout training, when compared at the same pretraining loss. Assuming the same initial downstream loss and final pretraining loss, the additional downstream loss reduction is

$$\delta\big(L_{\text{train}}(0) - L_{\text{train}}(T)\big). \tag{65}$$

Therefore, even when two training trajectories achieve exactly the same pretraining loss reduction, better gradient alignment can lead to a lower downstream loss.

To connect this result with the cosine similarity optimized by Nexus, note that

$$\frac{\nabla L_{\mathcal{T}}^\top \nabla L_{\text{train}}}{\|\nabla L_{\text{train}}\|_2^2} = \text{CosSim}(\nabla L_{\text{train}}, \nabla L_{\mathcal{T}}) \cdot \frac{\|\nabla L_{\mathcal{T}}\|_2}{\|\nabla L_{\text{train}}\|_2}. \tag{66}$$

Thus, if cosine similarity improves by δ while the gradient norm ratio remains approximately unchanged, the additional downstream loss reduction is proportional to δ and the accumulated pretraining loss reduction, weighted by the gradient norm ratio.

Although practical optimizers such as AdamW and Nexus do not follow exact gradient flow, this analysis offers an intuitive explanation for our central observation: **models can achieve the same pretraining loss while exhibiting different downstream performance, because their optimization trajectories differ in how effectively pretraining progress transfers to downstream tasks.**

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 3.2), 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 Sec. 4). 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 [76] (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 C.1. *Suppose each $\mathcal{L}_i$ is L-smooth and μ-strongly convex. That is, for any $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2$, we have:*

$$\begin{aligned}\mathcal{L}_i(\boldsymbol{\theta}_1) &\leq \mathcal{L}_i(\boldsymbol{\theta}_2) + \nabla\mathcal{L}_i(\boldsymbol{\theta}_2)^\top(\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2) + \frac{L}{2}\|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|_2^2,\\ \mathcal{L}_i(\boldsymbol{\theta}_1) &\geq \mathcal{L}_i(\boldsymbol{\theta}_2) + \nabla\mathcal{L}_i(\boldsymbol{\theta}_2)^\top(\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2) + \frac{\mu}{2}\|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|_2^2.\end{aligned} \tag{67}$$

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:

$$\mathbb{E}[\|\boldsymbol{\theta}_T - \boldsymbol{\theta}^*\|^2] \leq \left(1 - \frac{2\gamma\mu L}{L+\mu}\right)^T \|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*\|^2. \tag{68}$$

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

$$\mathbb{E}[\|\boldsymbol{\theta}_T - \boldsymbol{\theta}^*\|^2] \leq \left(\frac{\kappa - 1}{\kappa + 1}\right)^{2T} \|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*\|^2. \tag{69}$$

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

C.2.1 Proof of Convergence Rate (Theorem C.1)

In this section, we provide the detailed proof for Theorem C.1. 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:

$$\begin{aligned}\|\boldsymbol{\theta}_m - \boldsymbol{\theta}^*\|^2 &= \|\boldsymbol{\theta}_{m-1} - \gamma\nabla\mathcal{L}_{s_m}(\boldsymbol{\theta}_{m-1}) - \boldsymbol{\theta}^*\|^2\\ &= \|\boldsymbol{\theta}_{m-1} - \boldsymbol{\theta}^*\|^2 - 2\gamma\langle\nabla\mathcal{L}_{s_m}(\boldsymbol{\theta}_{m-1}), \boldsymbol{\theta}_{m-1} - \boldsymbol{\theta}^*\rangle + \gamma^2\|\nabla\mathcal{L}_{s_m}(\boldsymbol{\theta}_{m-1})\|^2.\end{aligned} \tag{70}$$

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:

$$\langle\nabla\phi_i(\boldsymbol{\theta}) - \nabla\phi_i(\boldsymbol{\theta}^*), \boldsymbol{\theta} - \boldsymbol{\theta}^*\rangle \geq \frac{1}{L-\mu}\|\nabla\phi_i(\boldsymbol{\theta}) - \nabla\phi_i(\boldsymbol{\theta}^*)\|^2. \tag{71}$$

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:

$$\begin{aligned}\langle\nabla\mathcal{L}_i(\boldsymbol{\theta}) - \mu(\boldsymbol{\theta} - \boldsymbol{\theta}^*), \boldsymbol{\theta} - \boldsymbol{\theta}^*\rangle &\geq \frac{1}{L-\mu}\|\nabla\mathcal{L}_i(\boldsymbol{\theta}) - \mu(\boldsymbol{\theta} - \boldsymbol{\theta}^*)\|^2\\ &= \frac{1}{L-\mu}\left(\|\nabla\mathcal{L}_i(\boldsymbol{\theta})\|^2 - 2\mu\langle\nabla\mathcal{L}_i(\boldsymbol{\theta}), \boldsymbol{\theta} - \boldsymbol{\theta}^*\rangle + \mu^2\|\boldsymbol{\theta} - \boldsymbol{\theta}^*\|^2\right).\end{aligned} \tag{72}$$

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

$$\langle\nabla\mathcal{L}_{s_m}(\boldsymbol{\theta}), \boldsymbol{\theta} - \boldsymbol{\theta}^*\rangle \geq \frac{1}{L+\mu}\|\nabla\mathcal{L}_{s_m}(\boldsymbol{\theta})\|^2 + \frac{\mu L}{L+\mu}\|\boldsymbol{\theta} - \boldsymbol{\theta}^*\|^2. \tag{73}$$

Substituting Eq. (73) back into Eq. (70) with $\boldsymbol{\theta} = \boldsymbol{\theta}_{m-1}$:

$$\begin{aligned}\|\boldsymbol{\theta}_m - \boldsymbol{\theta}^*\|^2 \leq\ & \|\boldsymbol{\theta}_{m-1} - \boldsymbol{\theta}^*\|^2 \\ & - 2\gamma \left(\frac{1}{L+\mu} \|\nabla \mathcal{L}_{s_m}(\boldsymbol{\theta}_{m-1})\|^2 + \frac{\mu L}{L+\mu} \|\boldsymbol{\theta}_{m-1} - \boldsymbol{\theta}^*\|^2 \right) \\ & + \gamma^2 \|\nabla \mathcal{L}_{s_m}(\boldsymbol{\theta}_{m-1})\|^2 \\ =\ & \left(1 - \frac{2\gamma\mu L}{L+\mu}\right) \|\boldsymbol{\theta}_{m-1} - \boldsymbol{\theta}^*\|^2 + \left(\gamma^2 - \frac{2\gamma}{L+\mu}\right) \|\nabla \mathcal{L}_{s_m}(\boldsymbol{\theta}_{m-1})\|^2.\end{aligned} \tag{74}$$

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:

$$\|\boldsymbol{\theta}_m - \boldsymbol{\theta}^*\|^2 \leq \left(1 - \frac{2\gamma\mu L}{L+\mu}\right) \|\boldsymbol{\theta}_{m-1} - \boldsymbol{\theta}^*\|^2. \tag{75}$$

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:

$$\mathbb{E}[\|\boldsymbol{\theta}_m - \boldsymbol{\theta}^*\|^2] \leq \left(1 - \frac{2\gamma\mu L}{L+\mu}\right) \mathbb{E}[\|\boldsymbol{\theta}_{m-1} - \boldsymbol{\theta}^*\|^2]. \tag{76}$$

Applying this recurrence relation recursively for T steps yields:

$$\mathbb{E}[\|\boldsymbol{\theta}_T - \boldsymbol{\theta}^*\|^2] \leq \left(1 - \frac{2\gamma\mu L}{L+\mu}\right)^T \|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*\|^2. \tag{77}$$

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

$$1 - \frac{2\gamma\mu L}{L+\mu} = 1 - \frac{4\mu L}{(L+\mu)^2} = \frac{(L-\mu)^2}{(L+\mu)^2} = \left(\frac{\kappa - 1}{\kappa + 1}\right)^2, \tag{78}$$

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

$$\mathbb{E}[\|\boldsymbol{\theta}_T - \boldsymbol{\theta}^*\|^2] \leq \left(\frac{\kappa - 1}{\kappa + 1}\right)^{2T} \|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*\|^2. \tag{79}$$

□

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 C.2 (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:*

$$\mathcal{J}_{NSGD}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}}[\mathcal{L}(\boldsymbol{x}; \boldsymbol{\theta})] - \frac{\gamma}{8} \mathbb{E}_{\boldsymbol{x}, \boldsymbol{x}' \overset{i.i.d}{\sim} \mathcal{D}}[CosSim(\nabla \mathcal{L}(\boldsymbol{x}; \boldsymbol{\theta}), \nabla \mathcal{L}(\boldsymbol{x}'; \boldsymbol{\theta}))], \tag{80}$$

subject to a Jacobian non-symmetric error and 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 3.2, 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.

□

Algorithm 3 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 $s_m \sim \text{Uniform}(\{1, \ldots, K\})$
    $\boldsymbol{g} \leftarrow \nabla \mathcal{L}_{s_m}(\boldsymbol{\theta}_{t,m-1})$
    $\boldsymbol{\theta}_{t,m} \leftarrow \boldsymbol{\theta}_{t,m-1} - \gamma \cdot \frac{\boldsymbol{g}}{\|\boldsymbol{g}\|_2}$ {Update inner trajectory}
  $\hat{\boldsymbol{g}}_t \leftarrow \boldsymbol{\theta}_{t,0} - \boldsymbol{\theta}_{t,K}$ {Compute Nexus pseudo-gradient}
  $\boldsymbol{\theta}_t \leftarrow \text{Opt}_{\text{outer}}(\boldsymbol{\theta}_{t-1}, \hat{\boldsymbol{g}}_t)$ {Outer-update}
Return: $\boldsymbol{\theta}_T$

A related implicit bias on Hessian alignment is analyzed by Wu et al. [57] in over-parameterized regimes. In their setting, the individual loss functions share identical minimizers, causing first-order gradients to vanish and leaving second-order properties as the sole means to characterize closeness. In contrast, modern LLM pretraining is far from this over-parameterized state. Consequently, first-order gradients remain non-zero, enabling the use of first-order gradient similarity to characterize the implicit regularization effect.

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 [77] 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 [78, 79] (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.

C.5 Extension to Generalized Similarity Metrics

In the main text, we instantiate the Nexus algorithm using Normalized SGD (NSGD) in the inner loop to maximize the cosine similarity between task gradients. Nexus may also be extended to optimize a broader class of generalized similarity metrics.

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

$$\text{Sim}_\Phi(\nabla \mathcal{L}_i(\boldsymbol{\theta}), \nabla \mathcal{L}_j(\boldsymbol{\theta})) \triangleq \langle \Phi(\nabla \mathcal{L}_i(\boldsymbol{\theta})), \Phi(\nabla \mathcal{L}_j(\boldsymbol{\theta})) \rangle. \tag{81}$$

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

$$\boldsymbol{\theta}_{t,m} = \boldsymbol{\theta}_{t,m-1} - \gamma \Phi(\nabla \mathcal{L}_{s_m}(\boldsymbol{\theta}_{t,m-1})). \tag{82}$$

Following the Taylor expansion logic in the main text, evaluating the transformed gradient of task j at the displaced parameter state from task i yields:

$$\Phi(\nabla \mathcal{L}_j(\boldsymbol{\theta}_{t,1})) - \Phi(\nabla \mathcal{L}_j(\boldsymbol{\theta}_{t,0})) = \left(\nabla_{\boldsymbol{\theta}_{t,0}} \Phi(\nabla \mathcal{L}_j(\boldsymbol{\theta}_{t,0}))\right)(\boldsymbol{\theta}_{t,1} - \boldsymbol{\theta}_{t,0}) + \mathcal{O}(\gamma^2) \tag{83}$$

$$= -\gamma \left(\nabla_{\boldsymbol{\theta}_{t,0}} \Phi(\nabla \mathcal{L}_j(\boldsymbol{\theta}_{t,0}))\right) \Phi(\nabla \mathcal{L}_i(\boldsymbol{\theta}_{t,0})) + \mathcal{O}(\gamma^2). \tag{84}$$

Taking the expectation over task permutations and pairing symmetric counterparts yields:

$$\overset{\mathbb{E}}{=} -\frac{\gamma}{2} \left(\nabla_{\boldsymbol{\theta}} \text{Sim}_\Phi(\nabla \mathcal{L}_i(\boldsymbol{\theta}_{t,0}), \nabla \mathcal{L}_j(\boldsymbol{\theta}_{t,0})) + \mathcal{E}_{\text{sym},\Phi,i,j} + \mathcal{E}_{\text{sym},\Phi,j,i}\right) + \mathcal{O}(\gamma^2), \tag{85}$$

where $\mathcal{E}_{\text{sym},\Phi,i,j} \triangleq \left(\nabla_{\boldsymbol{\theta}} \Phi(\nabla \mathcal{L}_i(\boldsymbol{\theta})) - (\nabla_{\boldsymbol{\theta}} \Phi(\nabla \mathcal{L}_i(\boldsymbol{\theta})))^\top\right) \Phi(\nabla \mathcal{L}_j(\boldsymbol{\theta}))$ represents the generalized Jacobian non-symmetric error.

Algorithm 4 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}
  $\hat{\boldsymbol{g}}_t \leftarrow \boldsymbol{\theta}_{t,0} - \boldsymbol{\theta}_{t,K}$ {Compute generalized pseudo-gradient}
  $\boldsymbol{\theta}_t \leftarrow \text{Opt}_{\text{outer}}(\boldsymbol{\theta}_{t-1}, \hat{\boldsymbol{g}}_t)$ {Outer update}
Return: $\boldsymbol{\theta}_T$

This derivation shows that for a given choice of the transformation function Φ, the sequential evaluation in the inner loop constructs a Jacobian-vector product that aligns with the gradient of Sim_Φ, subject to the non-symmetric error.

The generalized update covers several specific formulations:

- When $\Phi(\nabla \mathcal{L}) = \nabla \mathcal{L}$, Nexus optimizes the unnormalized dot product.
- When $\Phi(\nabla \mathcal{L}) = \boldsymbol{A} \nabla \mathcal{L}$ for a full-rank matrix $\boldsymbol{A}$, Sim_Φ induces a Mahalanobis inner product. For non-linear Φ, Sim_Φ acts as a kernel function mapping gradients into a Reproducing Kernel Hilbert Space (RKHS).
- When $\Phi(\nabla \mathcal{L}) = \frac{\nabla \mathcal{L}}{\|\nabla \mathcal{L}\|_2}$, the update recovers the standard cosine similarity maximization.
- Other choices, such as $\Phi(\nabla \mathcal{L}) = \text{sign}(\nabla \mathcal{L})$ or the orthogonalized gradient in Muon [9], enforce different geometric consensus choices across tasks.

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 3.1) and a more straightforward interpretation for downstream generalization (Eq. (4))—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 Fig. 7b, 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 Fig. 7b is a persistent geometric 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 Sec. 4.3.

D Additional Experiments

D.1 Detailed Experimental Settings

Our experimental setup largely follows the protocols established in Wen et al. [7] and OLMo et al. [80].

Pretraining Datasets. We utilize an in-house pretraining dataset similar to [11]. 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 [12] in Appendix D.5. 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. [7], 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 (Sec. 4.4) as required by Kaplan et al. [16].

Hyperparameters. Wen et al. [7] 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. [7]. 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. Compared with its base optimizer, Nexus has only one additional hyperparameter, the inner learning rate γ. See Sec. 4.2 for details.

Benchmarks. We evaluate on diverse benchmarks encompassing general knowledge (MMLU [81]), reasoning (GPQA, GPQA Diamond [82], BBH [83]), math (GSM8k [84], MATH500 [85]), and coding (HumanEval [86], MBPP [87]). 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. [7].

D.2 Detailed Hyper-parameters

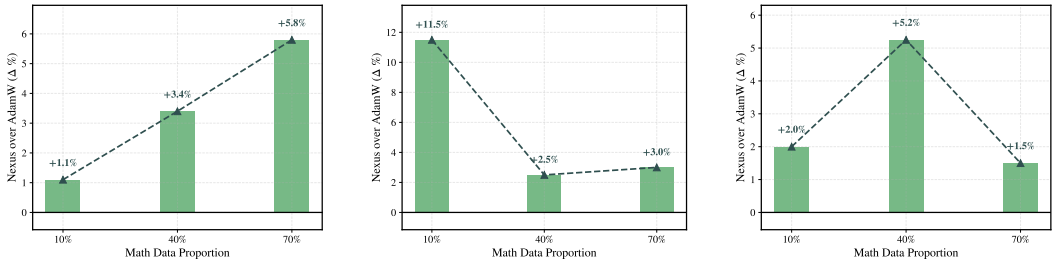
Our hyperparameter configurations strictly follow the baseline established in Wen et al. [7]. To verify the optimality of these settings on our specific pretraining corpus, we conduct a learning rate grid search with a multiplier of 2 (i.e., evaluating $0.5\times$ and $2.0\times$), which confirms that the original configurations remain optimal. For clarity and reproducibility, the key hyperparameters are summarized in Tab. 4. Regarding the learning rate schedule, the main experiments that utilize the Warmup-Stable-Decay (WSD) scheduler employ 1,000 warmup steps and 10,000 decay steps following Wen et al. [7], while the schedules for various ablation studies are clarified in their respective sections. Across all settings, we maintain a global batch size of 256, an Adam β of $(0.9, 0.95)$, an Adam ϵ of 10^{-10}, and a gradient clipping norm of 1.0.

Table 4: Summary of key hyperparameters for the main pretraining experiments.

Model Size	Optimizer	Outer LR	Inner LR (γ)	Chinchilla	Tokens (B)	Weight Decay	Reference
1B	Adam	0.002	-	4×	50	0.2	Sec. 4.4
	Nexus	0.002	0.04	4×	50	0.2	
3B	Adam	0.001	-	2×	110	0.2	Sec. 4.3 and Appendix D.3
	Muon	0.001	-	2×	110	0.1	
	Nexus	0.001	0.1	2×	110	0.2	Sec. 4.5 and Appendix D.5

Table 5: **Detailed numerical results for Fig. 5.**

Optim.	Inner LR	Pretraining	Gradient Similarity (↑)		Downstream	
		Val. Loss (↓)	Within Train Set	Train to OOD	Avg. Loss (↓)	Avg. Acc. (%, ↑)
AdamW	—	1.606	0.4499	0.2228	1.633	37.1
Nexus	0.01	1.598	0.4540	0.2252	1.623	38.2
	0.04	1.600	0.5416	0.2880	**1.612**	**39.5**
	0.10	1.602	0.5465	0.3042	**1.609**	**40.2**
	0.40	1.622	0.5656	0.3514	1.629	37.7



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

Figure 8: **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.

Compared to the base optimizers, Nexus introduces only one additional hyperparameter: the inner-loop learning rate γ. We tune γ based on a straightforward yet grounded empirical criterion: maximizing the regularization strength while strictly operating within the "same pretraining loss" regime. Specifically, we select the largest value of γ that incurs no discernible degradation in pretraining loss convergence, thereby maximizing the effect of the Nexus regularizer.

D.3 Robustness to Data Mixing

Motivation. In Sec. 3.2 and Eq. (4), we show that the gradient similarity implies the marginal gains on task i when optimizing on task j (in the first-order sense):

$$\underbrace{\mathcal{L}_i(\boldsymbol{\theta}) - \mathcal{L}_i(\boldsymbol{\theta} - \gamma \nabla \mathcal{L}_j(\boldsymbol{\theta}))}_{\text{decrease of task i after one GD step on task j}} = \gamma \nabla \mathcal{L}_i(\boldsymbol{\theta})^\top \nabla \mathcal{L}_j(\boldsymbol{\theta}) + O(\gamma^2). \tag{86}$$

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 Sec. 4.1.

Results. As shown in Fig. 8 and Tab. 6, 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.

D.4 Detailed Results for Data Mixture

Table 6: **Results on varying data mixtures** (3B models). Hyperparameters follow Secs. 4.1 and 4.3. 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

As shown in Tab. 6, 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.
- **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.

D.5 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. [12] 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 [12]. All other training configurations, including model architectures and base optimizer hyperparameters, are kept strictly identical to the main experiments detailed in Sec. 4.3.

Table 7: **Results on a public pretraining dataset [12].** 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. 7, 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.

D.6 Robustness to Learning Rate Schedule

Motivation. While the Warmup-Stable-Decay (WSD) scheduler [15] has become increasingly popular in recent LLM pretraining, the Cosine annealing schedule remains a widely adopted standard [13, 7]. 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 Sec. 4.3.

Results. As demonstrated in Tab. 8, 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.

D.7 Detailed Results for Model Size Scaling

This section provides the detailed experimental results corresponding to the model size scaling analysis discussed in Sec. 4.4.

As demonstrated above and analyzed in Sec. 4.4, 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).

Table 8: **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**

Table 9: **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**

D.8 Experiments on Muon Optimizers

This section provides the detailed experimental results discussed in Secs. 4.3 and 5.2.

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.

D.9 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 [11] and branch off from the 100,000-step checkpoints of the experiments in Appendix D.6. 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 [88, 2, 11].

Nexus still outperforms AdamW after SFT. As shown in Tab. 12, 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.

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

Table 10: **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**

Table 11: **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

(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

Table 12: **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**

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.

D.10 Experiment using SGDM

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

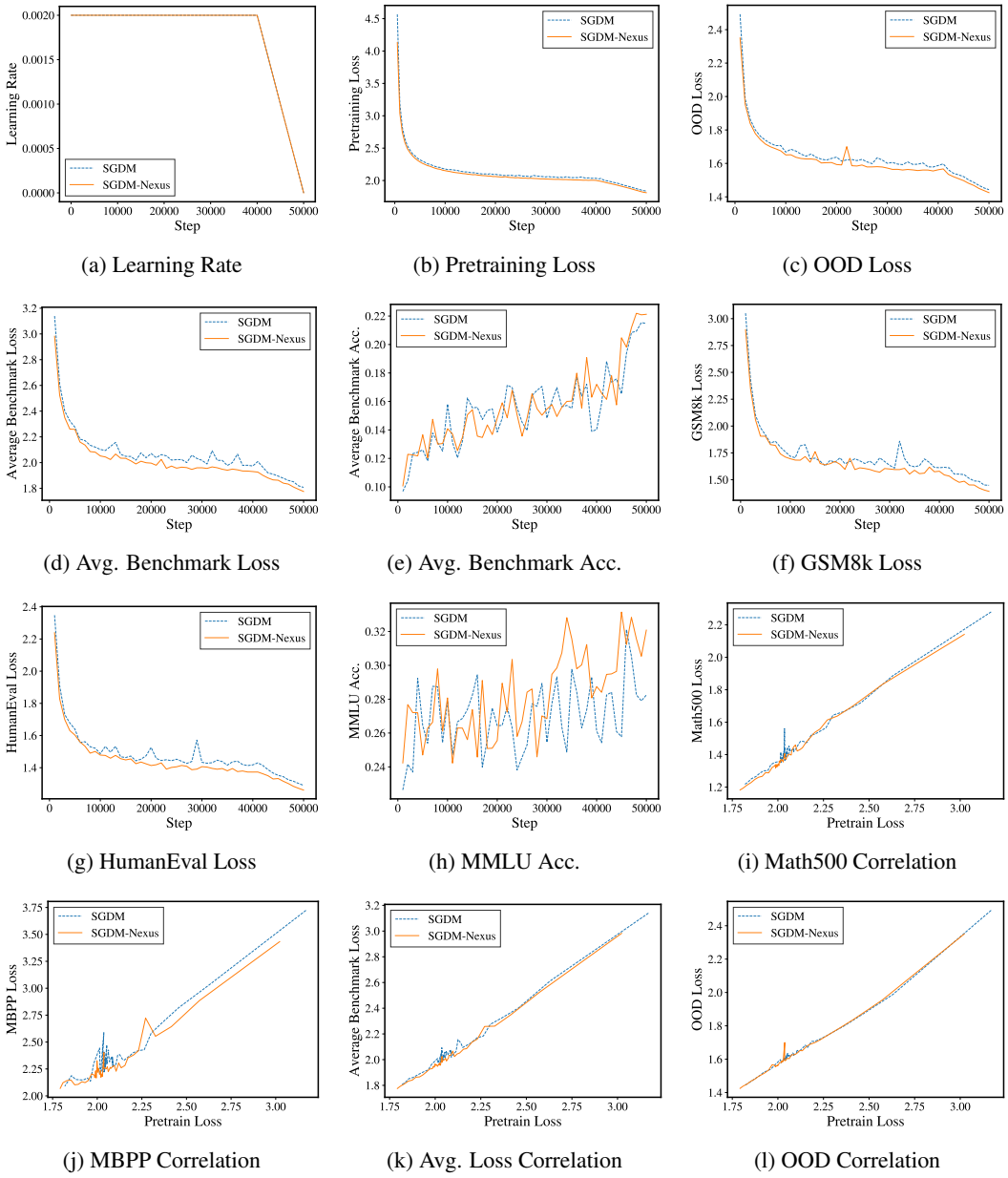
Motivation. We also conduct an experiment using SGD with Momentum (SGDM). However, SGDM typically converges slower than AdamW in LLM pretraining [89].

Muon-Inspired SGDM. To this end, we adapt the Muon optimizer [9] 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. 9. This design is inspired by the empirical findings in Liu et al. [36], 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 [90]. Surprisingly, in our LLM pretraining settings, this significantly accelerates SGDM, reaching speeds competitive with AdamW.

Settings. Due to computational constraints, we only conduct experiments on 1B models, strictly adhering to the experimental settings detailed in Sec. 4.1.

Results. On this SGDM baseline, we observe the following results (Fig. 10):

- **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 C.2.1.
- **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. 10). As shown in



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

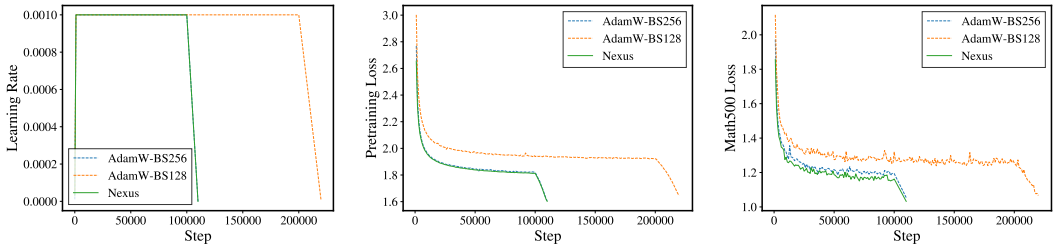
(d) Avg. Benchmark Loss (e) Avg. Benchmark Acc. (f) GSM8k Loss

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

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

Figure 10: **Experimental Results using SGDM.** Nexus both gives acceleration and implicit bias.

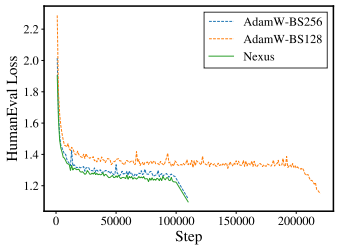
Fig. 10, the Nexus-SGDM curve lies consistently below the baseline SGDM, confirming that the favorable implicit bias persists even under different base optimizers.



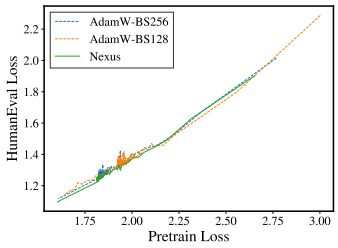
(a) Learning Rate

(b) Pretraining Loss

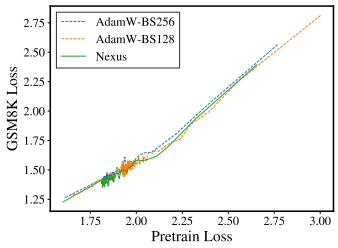
(c) Math500 Loss



(d) HumanEval Loss



(e) HumanEval Correlation



(f) GSM8K Correlation

Figure 11: **Experimental Results including the BS-128 Baseline.** The standard AdamW and Nexus trajectories are retained from the main trials in Sec. 4.3 for direct comparison.

Table 13: **Results of Small Batch Size Baseline.** Validation losses and downstream capabilities for AdamW trained with a reduced batch size of 128. Compared to the optimal BS-256 baseline reported in the main text (Tab. 1), reducing the batch size results in significant performance degradation.

Model	Configuration	Metric	Loss Metrics (↓)		Gen.	Reasoning			Math		Code		Avg.
			Pretrain.	OOD	MMLU	GPQA	GPQA-D	BBH	GSM8k	MATH	HumanEval	MBPP	All
1B	AdamW (BS=128)	Acc. (↑)	1.937	1.506	29.3	25.0	20.3	23.3	8.0	5.0	11.0	14.0	17.0
		Loss (↓)			2.396	2.333	2.231	1.728	1.510	1.286	1.340	2.091	1.864
3B	AdamW (BS=128)	Acc. (↑)	1.627	1.326	45.4	29.7	21.9	40.7	39.0	26.0	37.0	39.0	34.8
		Loss (↓)			2.262	2.033	1.937	1.556	1.281	1.067	1.148	1.951	1.654

D.11 Experiment using Smaller Batch Size

Motivation. It is well-established that smaller batch sizes can introduce an implicit bias that enhances generalization [91, 92, 93, 94]. To distinguish the geometric closeness bias of Nexus from this small-batch effect, we evaluate a baseline using AdamW with a reduced batch size of 128 on the 1B and 3B models. All hyperparameters remain identical to those in Sec. 4.1, except for the learning rate schedule, where we double the duration of both the stable and decay phases, as illustrated in Fig. 11(a).

On this baseline, we observe the following:

BS-128 is suboptimal compared to BS-256. As detailed in Sec. 4.1, our hyperparameter configurations are strictly aligned with the optimal settings established by Wen et al. [7]. Deviating from these settings compromises optimization efficiency. As demonstrated in Tab. 13, reducing the global batch size from 256 to 128 causes the training speed to severely lag behind. Consequently, the AdamW-BS128 baseline substantially underperforms the standard AdamW-BS256 across both pretraining convergence and downstream benchmark evaluations by a loss margin of more than 0.02.

BS-128 introduces implicit bias, but less significantly than Nexus. As illustrated in the correlation plots of Fig. 11, AdamW-BS128 indeed exhibits a favorable implicit bias compared to the standard AdamW, particularly during the early stages of training where the pretraining loss is high. However, in the later stages of training, the correlation curve of BS-128 lies between the standard AdamW and Nexus, indicating that its implicit bias is not as pronounced as that of Nexus. This confirms that the strong implicit bias of Nexus genuinely originates from its explicit geometric regularization, rather than merely mimicking the gradient noise effect of small batch sizes. Furthermore, while Nexus

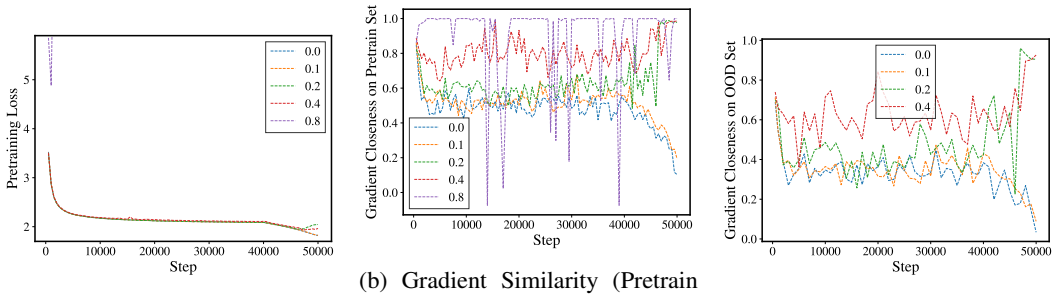
Algorithm 5 Controllable Nexus Variant for Ablation Study

Require: Control coefficient $\lambda \geq 0$

1: $\hat{\boldsymbol{g}}_t \leftarrow$ Algorithm 2 and 3$(\nabla\mathcal{L}(\boldsymbol{\theta}_t))$ {Standard Nexus pseudo gradient}

2: $\boldsymbol{u}_{\text{ablation}} \leftarrow \hat{\boldsymbol{g}}_t - \gamma\lambda\frac{\nabla\mathcal{L}(\boldsymbol{\theta}_t)}{\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|_2+\epsilon}$ {Decouple optimization and regularization}

3: **Return:** $\boldsymbol{u}_{\text{ablation}}$



(a) Pretraining Loss

(b) Gradient Similarity (Pretrain Set)

(c) Gradient Similarity (OOD Set)

Figure 12: **Ablation Results on Regularization Strength λ.** As the coefficient λ increases, there is a monotonic increase in gradient closeness across both training and OOD sets.

successfully induces this bias while maintaining an identical pretraining loss to the baseline, BS-128 fails to do so (suffering a pretraining loss degradation of approximately 0.02).

D.12 Ablation Studies on Regularization Term

Motivation. The standard formulations of Nexus (Algorithm 2 and 3) intrinsically couple the optimization of the primary loss and the regularization term at a fixed, implicit ratio of $1 : \gamma\frac{K-1}{4K}$. To systematically decouple and investigate the specific regularization effect of Nexus, we design a controllable variant. As demonstrated in Algorithm 5, after obtaining the Nexus pseudo gradient $\hat{\boldsymbol{g}}_t$, we subtract a scaled, L_2-normalized loss gradient $\gamma\lambda\frac{\nabla\mathcal{L}(\boldsymbol{\theta}_t)}{\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|_2}$. Consequently, the expectation of the actual modified update $\boldsymbol{u}_{\text{ablation}}$ applied to the parameters is given by:

$$\mathbb{E}[\boldsymbol{u}_{\text{ablation}}] = \gamma(1-\lambda)\sum_{i=1}^{K}\frac{\nabla\mathcal{L}_i(\boldsymbol{\theta}_t)}{\|\nabla\mathcal{L}_i(\boldsymbol{\theta}_t)\|_2} - \gamma^2\frac{K-1}{4K}\left(\nabla_{\boldsymbol{\theta}}\sum_{i\neq j}\text{CosSim}(\nabla\mathcal{L}_i, \nabla\mathcal{L}_j) + \boldsymbol{\mathcal{E}}_{\text{sym},i,j}\right) + \boldsymbol{\mathcal{E}}_{\text{2nd}}. \tag{87}$$

In this formulation, $\lambda \in [0, 1)$ serves as a direct dial to control the regularization strength. As λ approaches 1, the first-order loss-minimization component is explicitly suppressed, effectively amplifying the relative influence of the geometric regularization term.

However, this modified algorithm is intended exclusively for ablation purposes, as it incurs double the computational overhead per step (despite maintaining an identical token budget). Consequently, we evaluate the impact of the regularization-to-loss ratio using the 1B model configuration described in Sec. 4.1, sweeping across $\lambda \in \{0, 0.1, 0.2, 0.4, 0.8\}$, where $\lambda = 0$ corresponds to the original Nexus algorithm. The results are summarized as follows:

Results. As illustrated in Fig. 12, we observe a clear monotonic relationship: the higher the λ, the higher the gradient closeness on both pretraining and OOD sets. This empirical evidence confirms that the Nexus mechanism indeed explicitly maximizes gradient closeness by inner-loop outer-loop mechanism.

D.13 Multi-Seed Evaluation

We evaluate the stability of Nexus across multiple runs, including three runs for the 1B model and two runs for the 3B model, with the same data order across runs. Results are reported in Tab. 14.

Across both model sizes, Nexus achieves nearly the same pretraining loss as AdamW while consistently improving downstream performance. The improvements in downstream loss are substantially larger than the observed standard deviations.

Table 14: Multi-seed evaluation on 1B and 3B models. We report mean $\pm$ standard deviation across three and two runs, respectively.

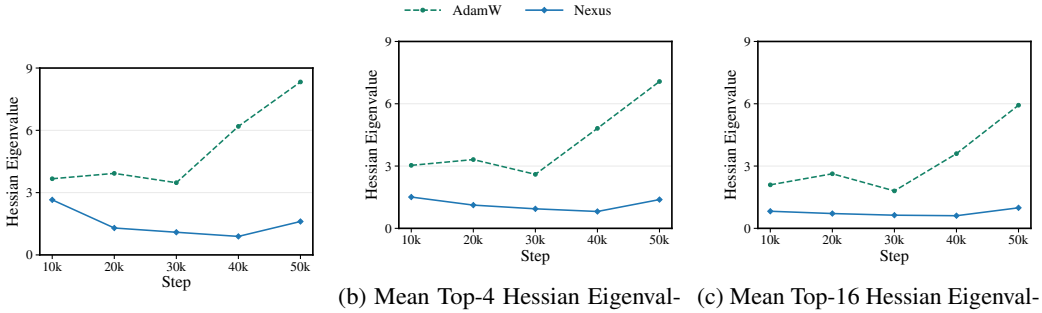
Model	Optim.	Pretrain. Loss (↓)	Avg. Downstream Loss (↓)	Avg. Acc. (%, ↑)
1B	AdamW	1.818 ± 0.0055	1.786 ± 0.0012	22.43 ± 0.41
1B	Nexus	1.818 ± 0.0055	$\mathbf{1.756 \pm 0.0017}$	24.07 ± 0.59
3B	AdamW	1.607 ± 0.0005	1.631 ± 0.0020	37.40 ± 0.30
3B	Nexus	1.602 ± 0.0000	$\mathbf{1.609 \pm 0.0000}$	$\mathbf{40.33 \pm 0.12}$

D.14 Closeness and Flatness

Does Nexus improve flatness in addition to closeness? As discussed in Sec. 2.2, downstream loss depends on both the distance to task-specific minimizers (closeness) and the directional curvature of the loss landscape (flatness).

To investigate this, we measure the leading Hessian eigenvalues throughout 1B pretraining. As shown in Fig. 13, Nexus consistently exhibits smaller top-1, average top-4, and average top-16 Hessian eigenvalues than AdamW. For example, at step 50k, the largest eigenvalue decreases from 8.330 to 1.609.

These results suggest that Nexus affects both closeness and flatness. While our theoretical analysis focuses on closeness under controlled curvature assumptions, the two properties may be coupled in practice. Their individual contributions to downstream generalization remain to be understood.



(a) Top-1 Hessian Eigenvalue

(b) Mean Top-4 Hessian Eigenvalues

(c) Mean Top-16 Hessian Eigenvalues

Figure 13: Leading Hessian eigenvalues throughout 1B pretraining. We report the top-1 eigenvalue and the averages of the top-4 and top-16 eigenvalues. Nexus consistently exhibits lower leading Hessian eigenvalues than AdamW.

D.15 Computational Overhead

Nexus introduces no additional forward or backward passes compared to standard pretraining. Its extra computation comes from updating and synchronizing the auxiliary inner model. Under $8\times$A100-equivalent accounting, the measured wall-clock overhead over AdamW is only 0.55% for the 1B model and 2.52% for the 3B model.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: We state our contribution clearly in the abstract and introduction.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: See Sec. 6.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

Justification: See Appendix B.3.2.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: We provide detailed hyper-parameters, training code on public dataset Appendix D.5.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: See Appendix D.5 and Sec. 4.1; https://github.com/huanranchen/NexusPretraining.

Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: See Sec. 4.1.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: We report mean $\pm$ standard deviation across three runs for the 1B model and two runs for the 3B model, with the same data order across runs (see Tab. 14).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: See Appendices D.2 and D.15.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: We follow the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [N/A]

    Justification: This is a technical pre-training paper.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [N/A]

    Justification: We follow Wen et al. [7].

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We cite them.

    Guidelines:

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

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [Yes]

    Justification: The training code and usage instructions are available at https://github.com/huanranchen/NexusPretraining.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: No relevance with human subjects.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects

    Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [N/A]

Justification: We only use LLM to assist writing papers (e.g., polishing grammar, rephrase some sentences).

Guidelines:

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