REDESIGN MIXTURE-OF-EXPERTS ROUTERS WITH MANIFOLD POWER ITERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

The router is a cornerstone component of Mixture-of-Experts models. Each row of the router matrix represents an expert, computing its similarity to the MoE input to determine which subset of experts is activated. Ideally, each router row should condense the salient characteristics of its associated expert matrix into a representative vector, such that its dot product can better indicate whether the token should be routed to that expert. However, there is no explicit design principle in existing routers to enforce such a condensation. In this paper, we posit that each router row should be aligned with the principal singular direction of its associated expert matrix, which provides the most expressive one-dimensional representation of the expert matrix. Building on this insight, we propose a router redesign with Manifold Power Iteration (MPI). Specifically, it adopts a "Power-then-Retract" paradigm, in which the power iteration step steers the router weights toward the desired directions, while the retraction enforces a norm constraint to ensure efficient and stable optimization. Our theoretical analysis further confirms that the proposed router design indeed converges toward the target directions. We pretrain MoE models ranging from 1B to 11B parameters and conduct comprehensive experiments to demonstrate the effectiveness of our router design.

1 INTRODUCTION

Mixture-of-Experts (MoE, Muennighoff et al., 2025; OpenAI, 2025; DeepSeek-AI, 2026; GLM-5-Team et al., 2026) stands as a pivotal model architecture in LLMs for scaling model capacity with constrained computational budgets. Specifically, it replaces standard Transformer Feed Forward Networks (FFNs) with an ensemble of expert modules, using a router to select experts per token for sparse activation. This enables more efficient training under fixed computation budget, which paves the way for LLM training with trillions of parameters (DeepSeek-AI, 2026; Team et al., 2026).

At the heart of MoE models lies the router, which is typically parameterized as a simple linear matrix. For each input token, the router calculates its similarity scores against the matrix rows and dispatches the token to the top-scoring experts. While this design is straightforward and has long been accepted as a matter of course, we challenge this conventional wisdom. Ideally, each row of the router matrix should capture the intrinsic characteristics of the corresponding expert. Such a representation would make the resulting similarity scores more indicative of whether a token should be assigned to that expert. However, there lacks a constraint that enforces the encoding of expert features into router rows with limited expressivity. This absence may lead to suboptimal router design, compromising both training convergence and the overall capability of MoE.

This design gap motivates us to revisit MoE router and explore a more principled way to improve. To this end, we propose a new router design that seeks to couple each router row with the principal singular direction of its corresponding expert weight matrix. The design choice stems from a well-known property in linear algebra (Golub & Van Loan, 1996): the principal singular direction retains the dominant component of a matrix, making it an optimal representation under one-dimensional compression. Given that each expert is parameterized by weight matrices, its most informative direction naturally provides a principled basis to construct the router vector. However, it is computationally expensive to repeatedly obtain this direction throughout training. To reduce this overhead, we leverage power iteration (Halko et al., 2010) as a lightweight alternative to exact singular value

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

decomposition (SVD). Specifically, power iteration relies only on standard matrix-vector products to estimate the principal singular vector, thereby avoiding the need for costly full matrix factorization.

In practice, we restrict each training step to a single power iteration on the router weights, keeping this overhead modest. The resulting weights are then retracted to a fixed L_2 norm to prevent potential explosion or collapse of their magnitude. This "Power-then-Retract" paradigm gives rise to the name of our design: Routers with Manifold Power-Iteration (MPI). To better understand the behavior of our design, we also provide a steepest-ascent interpretation of this one-step power iteration update. Specifically, it maximizes the projection of the router weights onto the corresponding expert weights under the principle of minimal updates. From a theoretical perspective, this interpretation establishes MPI as an expert–router co-design scheme in which router design is explicitly coupled with the expert weights, a design principle long overlooked in traditional MoE routers.

We conduct large-scale pretraining experiments using trillions of training tokens in total. These experiments span a wide range of model scales and MoE designs, providing a comprehensive empirical evaluation of our router design. Empirical observations on models scaling up to 11B parameters confirm that MoE models equipped with our router design consistently achieve faster convergence, stronger downstream performance, and improved load balancing. We contend that this constitutes a substantial shift in how MoE routers are conventionally designed and brings intrinsic improvements to the overall MoE architecture. The observed improvements remain consistent across variations in training regimes and MoE setups, suggesting that these gains arise entirely from our principled treatment of router design. We hope these findings shed light on the fundamental role of router design in MoE training and inspire future efforts toward more principled expert–router co-design.

2 Background: Mixture-of-Experts (MoE)

In this paper, we center our discussion on MoE-based LLMs, where token dispatch is mediated by a router. To perform this dispatch, MoE router typically adopts a 2D linear weight matrix $\boldsymbol{R} \in \mathbb{R}^{N \times D}$ to project input $\boldsymbol{x} \in \mathbb{R}^D$ into gating weight vector $\boldsymbol{w}$ over the N experts:

$$\boldsymbol{w} = \texttt{Softmax}\left(\texttt{TopK}\left(\boldsymbol{x}\boldsymbol{R}^\top\right)\right). \tag{1}$$

where the experts with the top-K largest gating weights are selected for computation. The MoE layer output is then computed as a weighted sum of the selected expert outputs:

$$\text{MoE}\left(\boldsymbol{x}\right) \;=\; \sum_{k=1}^{K} \boldsymbol{w}_k \cdot \boldsymbol{E}_k(\boldsymbol{x}). \tag{2}$$

Consistent with standard practice in modern MoEs, we implement each expert as a Gated Linear Unit (GLU, Shazeer, 2020) with Swish activation (Ramachandran et al., 2017):

$$\boldsymbol{E}_k(\boldsymbol{x}) = \left(\texttt{SiLU}(\boldsymbol{x}\boldsymbol{W}_g^k) \odot (\boldsymbol{x}\boldsymbol{W}_p^k)\right) \boldsymbol{W}_o^k \;.$$

While this router design is straightforward and sufficient in most cases, certain limitations persist at the design stage and hinder its optimal performance. For instance, no explicit constraint is imposed to ensure that routers can faithfully reflect the experts' intrinsic features. Ideally, $\boldsymbol{R}_{[i]}$ should maximally preserve the geometry of the i-th expert weights to better act as a feature vector; however, such a constraint is absent and may leave $\boldsymbol{R}_{[i]}$ poorly representative of its associated expert and lead to suboptimal token–expert dispatch as a consequence. Leveraging this insight, we propose a redesign of MoE router and empirically confirm its effectiveness in the following sections.

3 Methodology

In this section, we first elucidate our motivation, introduce Manifold Power Iteration in detail, and discuss several key principles underlying its design. We then revisit the essence of our method from an optimization perspective and provide accessible insights into how it works.

3.1 Motivation

The preceding analysis suggests that steering each router row $\boldsymbol{R}_{[i]}$ toward the dominant spectral direction of the expert weights $\boldsymbol{W}_*^i$ could serve as a potential principle for MoE co-design. To

make this principle concrete, we seek a criterion that characterizes the degree of such router-expert coupling and reflects how strongly $\boldsymbol{R}_{[i]}$ aligns with the dominant structure of $\boldsymbol{W}_*^i$. Specifically, for each $\boldsymbol{R}_{[i]}$ and its associated $\boldsymbol{W}_*^i$, we construct this criterion based on their normalized response:

$$\max_{\boldsymbol{R}_{[i]}} \quad \phi(\boldsymbol{W}_*^i, \boldsymbol{R}_{[i]}) \;=\; \frac{\|\boldsymbol{R}_{[i]}\boldsymbol{W}_*^i\|_2^2}{\|\boldsymbol{R}_{[i]}\|_2^2}, \tag{3}$$

where $\phi(\cdot)$ is also known as the Rayleigh quotient of $\boldsymbol{W}_*^i\boldsymbol{W}_*^{i\top}$ with respect to $\boldsymbol{R}_{[i]}$[1]. The criterion $\phi(\cdot)$ is maximized when $\boldsymbol{R}_{[i]}$ lies in the span of the principal left singular vector of $\boldsymbol{W}_*^i$, which corresponds exactly to the desired router–expert spectral coupling.

However, it is prohibitive to perform an exact singular value decomposition for all expert matrices to obtain the principal singular vectors, especially when repeated at every training step. To address this issue, we instead leverage power iteration to $\boldsymbol{R}_{[i]}$ as a lightweight alternative to SVD, built upon the power method theory Golub & Van Loan (1996). This design requires only standard matrix-vector multiplications, which are well suited to modern hardware and incur only marginal overhead. While power iteration steers each router row toward the target direction, repeated updates may cause uncontrolled growth of its norm. We therefore impose a row-wise norm constraint and retract the updated router rows onto the corresponding manifold. We next detail this "Power-then-Retract" paradigm and defer a deeper discussion of its design principles to a later section.

3.2 Routers with Manifold Power-Iteration

3.2.1 Manifold Power Iteration via Power-then-Retract.

Based on the preceding discussion, our router design adopts a "Power-then-Retract" paradigm, consisting of (1) a power iteration step that leverages expert weights to update router and (2) an L_2 retraction step for norm control and numerical stability. For any row $\boldsymbol{R}_{[i]}$ of the router weights, we first fetch its associated expert weights $\boldsymbol{W}_g^i$ and perform a single step of power iteration:

$$\hat{\boldsymbol{R}_{[i]}} \;=\; \boldsymbol{R}_{[i]}\, \boldsymbol{W}_g^i\, \boldsymbol{W}_g^{i\top}. \tag{4}$$

Notably, repeated power-iteration updates without norm control on the expert weights, may drive $|\hat{\mathbf{R}}_{[i]}|_2$ toward explosion or vanishing over training. To avoid such instability and preserve a controlled router scale, we constrain the L_2 norm of $\hat{\mathbf{R}}_{[i]}$ to a hyperparameter C after each iteration:

$$\boldsymbol{R}'_{[i]} = C \cdot \frac{\hat{\boldsymbol{R}_{[i]}}}{\|\hat{\boldsymbol{R}_{[i]}}\|_2}, \tag{5}$$

while designed to avoid instability, this retraction provides additional benefits. Conceptually, it rectifies the expert-wise bias induced by scale disparities in router norms, where an amplified norm can easily inflate gating weights and overload the corresponding expert. Combining the two steps above, we reconstruct the router matrix $\boldsymbol{R}'$ by concatenating the transformed rows:

$$\boldsymbol{R}' = \left[\boldsymbol{R}'^{\top}_{[1]}\; \boldsymbol{R}'^{\top}_{[2]}\; \cdots\; \boldsymbol{R}'^{\top}_{[N]}\right]^{\top}. \tag{6}$$

We then replace the original router with $\boldsymbol{R}'$ and the final gating weights $\boldsymbol{w}'$ is computed as:

$$\boldsymbol{w}' = \texttt{Softmax}\left(\texttt{TopK}\left(\boldsymbol{x}\boldsymbol{R}'^{\top}\right)\right). \tag{7}$$

Importantly, our redesign is confined to the router weights themselves while leaving the original routing-logit computation unchanged, making it readily compatible with existing MoE designs. We designate this refined router $\boldsymbol{R}'$ as **Routers with Manifold Power-Iteration (MPI)**.

Figure 1 (a) provides a Pytorch-style pseudocode to help understand our implementation.

3.2.2 Scaling Principle for the Norm Constraint.

Having introduced the row-wise norm constraint above, we next consider how its target scale C should be chosen. To this end, we establish a principle for setting C. Inspired by (Pethick et al., 2025), we introduce a constant-scale assumption on the routing logits to ensure stability:

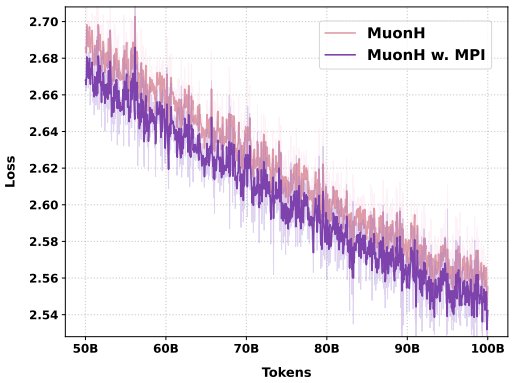
$$\|\,\boldsymbol{x}\boldsymbol{R}'^{\top}\,\|_\infty \;\sim\; O(1),$$

[1]Unless specified otherwise, we use $\mathbf{W}_g^i$ for $\mathbf{W}_*^i \in \{\mathbf{W}_g^i, \mathbf{W}_o^i, \mathbf{W}_p^i\}$ hereafter for the sake of simplicity.

```python
from torch.nn.functional import normalize
from megablocks.layers.moe import MoE

class MoE_MPI(MoE):
    def foward(self, x, C_prime=1):
        # R: [N, D], wg: [N, D, d]
        # PowerIter: ([N, 1, D] @ [N, D, d])
        #   @ [N, d, D] -> [N, D]
        R_hat = (self.R.unsqueeze(1) @ wg.
            transpose(1, 2) @ wg).squeeze()
        # Retraction: [N, D] -> [N, D]
        R_prime = normalize(R_hat, p=2, dim=-1)
        C = C_prime ** (N ** -0.5)
        logits = C ** (x @ R_prime.T)
        s_prime = logits.softmax(dim=-1)
        w_prime, _ = torch.topk(s_prime, dim=-1)
        w_prime /= w_prime.sum(-1, keepdim=True)
        return self.experts(x, s_prime, w_prime)
```

(a) Pseudocode of Manifold Power Iteration.



(b) Pretraining convergence on MuonH-1B.

Figure 1: (a) Pytorch-style pseudocode for Manifold Power Iteration. (b) Convergence comparison for MoE with MPI, exemplified by MuonH-1B. Our router design achieves a 0.013 reduction in pretraining loss. Similar observations for other optimizers are provided in Appendix.

Given a scale-invariant $\mathbf{x}$, this upper bound implies that C should scale as $\Theta(1/\sqrt{N})$ with the number of experts N. This scaling relationship is evidenced by the following derivation:

$$\| \boldsymbol{x}\boldsymbol{R}'^{\top} \|_\infty \leq \sqrt{\sum_{i=1}^{N} (\boldsymbol{x}\boldsymbol{R}_{[i]}^{\top})^2} \sim O(C\sqrt{N}),$$
$$\texttt{where} \quad \boldsymbol{x}\boldsymbol{R}_{[i]}^{\top} \sim O(C) \quad \text{for each expert.} \tag{8}$$

Therefore, to keep the routing-logit scale bounded as N varies, we choose C as $C'/\sqrt{N}$, with C' denoting a scale-invariant global hyperparameter. This scaling rule adapts the norm constraint to the number of experts while keeping the routing logits on a consistent scale.

3.3 From Rayleigh-Quotient Maximization to Manifold Power Iteration

Section 3.2 formulates the desired router-expert spectral coupling through a Rayleigh-quotient criterion and realizes it with the proposed "Power-then-Retract" update. We further examine this design through a steepest-ascent interpretation to provide additional insight into the underlying behavior of our router update. To this end, we cast the same object as a locally constrained ascent problem, where each update seeks to increase the Rayleigh quotient within a bounded neighborhood while satisfying the norm-preserving condition:

$$\max_{\Delta \boldsymbol{r}} \quad \boldsymbol{\Phi}(\boldsymbol{W}_g, \boldsymbol{R}'_{[i]} + \Delta \boldsymbol{r}) \tag{9}$$
$$\textbf{s.t.} \ \|\boldsymbol{R}'_{[i]}\|_2 = \|\boldsymbol{R}'_{[i]} + \Delta \boldsymbol{r}\|_2 = C, \ \ \|\Delta \boldsymbol{r}\|_2 \leq \eta.$$

Since the normalization ensures a constant denominator, the original objective reduces to:

$$\boldsymbol{\Phi}(\boldsymbol{W}_*^i, \boldsymbol{R}'_{[i]}) = \|\boldsymbol{R}'_{[i]}\boldsymbol{W}_*^i\|_2^2 = \boldsymbol{R}'_{[i]}\boldsymbol{W}_*^i\boldsymbol{W}_*^{i\top}\boldsymbol{R}'^{\top}_{[i]},$$

$\Delta \boldsymbol{r}$ represents the update, and η constrains the update within a small bounded region.

We impose norm constraints on both $\boldsymbol{R}'_{[i]}$ and $\boldsymbol{R}'_{[i]} + \Delta \boldsymbol{r}$. To analyze this optimization landscape, we consider a first-order Taylor approximation of the objective:

$$\boldsymbol{\Phi}(\boldsymbol{W}_g, \boldsymbol{R}'_{[i]} + \Delta \boldsymbol{r}) \ = \ \boldsymbol{\Phi}(\boldsymbol{W}_g, \boldsymbol{R}'_{[i]}) + \langle \boldsymbol{G}, \Delta \boldsymbol{r} \rangle,$$

where $\boldsymbol{G} = 2\,\boldsymbol{R}'_{[i]}\boldsymbol{W}_g\boldsymbol{W}_g^{\top}$ represents the gradient of $\boldsymbol{R}_{[i]}$. This Taylor approximation reduces the objective to maximizing the inner product $\langle \boldsymbol{G}, \Delta \boldsymbol{r} \rangle$. To satisfy the norm constraint and ensure that

the updated router $\boldsymbol{R}'_{[i]}$ remains on the spherical manifold, we project the gradient $\boldsymbol{G}$ onto the tangent space of the sphere. Defining $\boldsymbol{M} \coloneqq \boldsymbol{W}_g \boldsymbol{W}_g^\top$ to simplify notation, and setting $C = 1$ without loss of generality, the gradient ascent update $\Delta \boldsymbol{r}_g$ on the manifold is formulated as:

$$\Delta \boldsymbol{r}_g = \eta \, \boldsymbol{G} \left(\boldsymbol{I} - \frac{\boldsymbol{R}'^\top_{[i]} \boldsymbol{R}'_{[i]}}{\boldsymbol{R}'_{[i]} \boldsymbol{R}'^\top_{[i]}} \right) = \eta \left(\boldsymbol{R}'_{[i]} \boldsymbol{M} - \boldsymbol{R}'_{[i]} \left(\boldsymbol{R}'_{[i]} \boldsymbol{M} \boldsymbol{R}'^\top_{[i]} \right) \right), \tag{10}$$

where the constant scaling factors are absorbed into the step size η for notational simplicity. We also derive an approximation for the exact update $\Delta \boldsymbol{r}_M$ introduced by Manifold Power-Iteration:

$$\Delta \boldsymbol{r}_M \approx \frac{1}{\boldsymbol{R}'_{[i]} \boldsymbol{M} \boldsymbol{R}'^\top_{[i]}} \Big(\boldsymbol{R}'_{[i]} \boldsymbol{M} - \boldsymbol{R}'_{[i]} (\boldsymbol{R}'_{[i]} \boldsymbol{M} \boldsymbol{R}'^\top_{[i]}) \Big). \tag{11}$$

By comparing steepest ascent ($\Delta \boldsymbol{r}_g$) with our router update ($\Delta \boldsymbol{r}_M$), we observe an unexpected structural alignment. In this light, our proposed router design constitutes an optimization tailored for maximum Rayleigh-quotient constraints with an adaptive step-size. Specifically, it drives a steady convergence of the router weights toward the principal singular vector, with the step size decreasing and updates becoming more careful as $\boldsymbol{R}'_{[i]}$ are mostly aligned with the principal direction of $\boldsymbol{W}_g^i$ (i.e. $\Delta \boldsymbol{r}_M$ is moderated because the denominator $\boldsymbol{R}'_{[i]} \boldsymbol{M} \boldsymbol{R}'^\top_{[i]}$ in Eq. 11 increases), and vice versa.

These interpretations help explain how the proposed method drives each router row toward the dominant spectral direction of its associated expert. We provide supplementary derivation in Appendix A and hope this perspective can inspire readers.

4 Experiment

4.1 MPI is an Optimizer-Agnostic Design

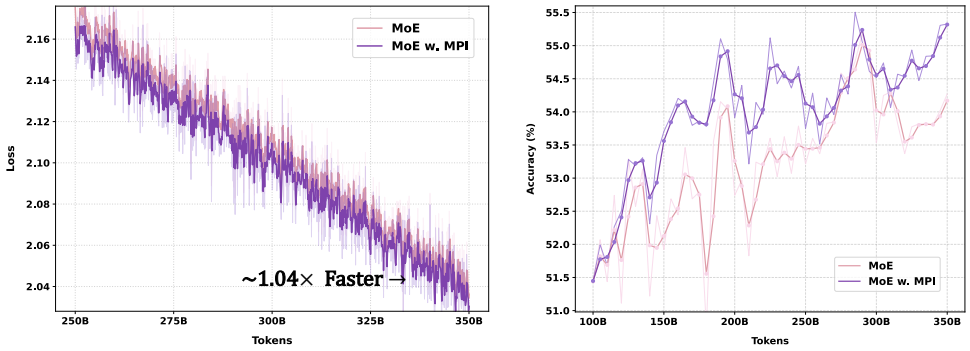
We first pretrain 1B MoE models with a diverse set of optimizers for two purposes: (1) to confirm that MPI remains robust to optimizer-induced variations in model features, providing evidence that its gains stem from the router design itself rather than optimizer-specific effects; and (2) to use these experiments as a pilot study to guide the setup of our subsequent large-scale experiments. To this end, we consider several optimizers with distinct update behaviors. Specifically, we pretrain 1B MoE models with AdamW (Loshchilov & Hutter, 2019) and Muon (Jordan et al., 2024), together with their Hyperball Optimization (Wen et al., 2026) variants, AdamH and MuonH.[2]

We pretrain all baselines on 100B tokens and evaluate both their convergence and downstream performance. Detailed model and optimizer configurations are provided in Appendix B. Figure 1b presents the convergence comparison, with MuonH shown as a representative example, while Table 1 summarizes the average downstream performance across optimizer settings. Crucially, MoE with MPI consistently achieves faster convergence and stronger downstream performance across all optimizer setups at the 1B scale. Motivated by these consistent gains, we scale up our experiments to examine whether the benefits persist at larger model capacities. For the large-scale experiments, we select MuonH owing to (1) its favorable hyperparameter transferability across scales and (2) its strongest convergence performance among the 1B MoE baselines.

Table 1: Downstream performance, reported as the average accuracy across 25 benchmarks. MPI shows consistent downstream performance improvement across all evaluated optimizers. Detailed results for individual tasks are provided in Table 8. Unless otherwise specified, we only report the average results across these tasks in the remainder of this paper.

	AdamW	AdamH	Muon	MuonH
MoE	42.26	42.59	43.01	42.78
+ MPI	43.56	43.93	43.55	43.98

[2]For readers unfamiliar with these advanced optimizers, we refer to the related work in Section 7.



(a) Pretraining Loss Comparison (b) Average Downstream Task Accuracy

Figure 2: Convergence and downstream performance comparison. MoE models with MPI achieve faster convergence and stronger downstream performance throughout 11B-scale pretraining.

4.2 Comparative Analysis with vanilla MoE

We train MoE models with MPI at two larger scales, 3B and 11B. All models are pretrained on 350B tokens sampled from the FineWeb-Edu dataset (Lozhkov et al., 2024), with an additional 1B tokens reserved for validation. We then continue midtraining on 100B tokens from (Olmo et al., 2025). Full architecture hyperparameters and training configurations are provided in Appendix B. By design, MPI preserves the standard router structure and is complementary to prior plug-in improvements built around the router. We therefore omit direct comparisons with these methods from the main experiments. Section 6 provides dedicated experiments to further verify this compatibility.

Convergence and Performance. Our pretraining results across two scales confirm that MoE with MPI achieves faster convergence and improved downstream performance. Figure 2 (a) and (b) present a comparison of pretraining loss and downstream performance evolution for 11B MoE.

We observe that MoE with MPI enables more effective training and maintains a loss advantage throughout pretraining. To assess whether this advantage generalizes to held-out data, we further report perplexity on the the held-out Math and Code sets from OLMo 3 (Olmo et al., 2025). Table 2 confirms that this perplexity advantage remains consistent across all evaluated domains and extends robustly to held-out data.

Table 2: Perplexity for MoE models with MPI.

PPL (↓)	**Validation**	**Math**	**Code**
MoE 3B	0.764	1.688	1.376
MoE *w.* MPI	**0.754**	**1.581**	**1.296**
MoE 11B	0.728	1.852	1.263
MoE *w.* MPI	**0.723**	**1.581**	**1.259**

Beyond these language-modeling improvements, we further evaluate downstream tasks to assess whether the gains translate into broader model capabilities. Specifically, we monitor a suite of nine core tasks, with Figure 2(b) showing the evolution of their average accuracy throughout pretraining. We observe that MPI maintains a consistent advantage on these downstream tasks. Following midtraining, we further extend the evaluation to a broader set of more challenging benchmarks, covering knowledge-intensive QA (Clark et al., 2018; Hendrycks et al., 2021), reading comprehension (Joshi

Table 3: Performance of MoE models with MPI on challenging benchmarks at 3B and 11B scales.

Task (→)	ARC-C	MMLU	TriviaQA	NaturalQs	BBH	GSM8K	MBPP	AVG.
Setup (→)	5-shot	5-shot	5-shot	5-shot	3-shot CoT	8-shot CoT	3-shot	
MoE 3B	55.91	47.01	45.78	17.87	29.53	16.22	42.25	36.37
w. MPI	58.96	48.83	46.52	20.13	30.99	20.92	44.54	38.70
MoE 11B	61.54	50.00	55.41	25.30	31.17	17.89	45.12	40.92
w. MPI	62.24	50.93	56.89	25.36	31.45	27.60	44.87	42.76

et al., 2017; Kwiatkowski et al., 2019), language understanding and reasoning (Suzgun et al., 2022), and mathematical reasoning and code generation (Cobbe et al., 2021; Austin et al., 2021).

As shown in Table 3, MoE with MPI achieves consistent gains, further validating the effectiveness of our router design. Further details on the evaluation setup are provided in Appendix B.3.

Load Balance. As shown in Figure 3, a noticeable decrease in load balancing loss is observed for MoE models with MPI. Specifically, this loss drops sharply during the early stages and remains low thereafter. However, part of this reduction may result from the smaller router norm caused by retraction. Therefore, we report MaxVio on validation set as a more accurate load balance reflection.

Table in Figure 3 reports both MaxVio_{Batch} and MaxVio_{Global} across different models. The results indicate that the load-balancing loss remains effective with MPI, and MPI even achieves better expert load balance than the vanilla MoE. We tentatively attribute this improvement to the retraction design and leave a more thorough investigation for future work.

$\mathrm{MaxVio}_{\mathrm{Batch}}$ (↓)		$\mathrm{MaxVio}_{\mathrm{Global}}$ (↓)	
MoE	*w.* MPI	MoE	*w.* MPI
1.133	1.024	0.964	0.711

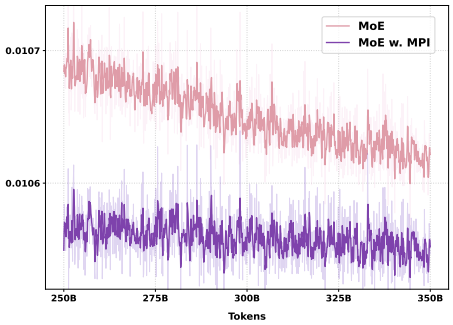


Figure 3: Load balanc analysis for MoE models with MPI, showing lower load-balancing loss and better expert balance than vanilla MoE.

Efficiency Analysis. We provide a detailed efficiency breakdown of MoE models with MPI to evaluate its additional overhead and scalability for large-scale pretraining.

(1) Our router design incurs negligible training overhead. In our 11B experiments, MPI introduces only a 0.2% throughput slowdown relative to vanilla MoE. This modest overhead can be attributed to the fact that MPI adds less computation than processing N tokens, making it marginal relative to the total number of tokens in a micro batch. As for communication overhead, MPI requires only a router-sized all-gather across expert ranks to assemble the complete refined router parameters under expert parallelism. Taken together, our analysis from both perspectives suggests that MPI is practical for larger-scale MoE pretraining .

(2) At inference time, once the model weights are fixed, the router weights can be precomputed when the model is loaded. Therefore, MPI incurs no additional inference overhead and remains fully compatible with standard inference engines out of the box.

Overall, these analyses support the scalability of MPI to larger-scale MoE training and deployment.

5 Method Analysis

5.1 Improved Router-Expert Alignment with Principal Singular Directions

We perform a post-hoc parameter analysis to verify that our design better aligns router rows with the principal singular vectors of their associated expert. We measure this router-expert alignment as:

$$\lambda = \frac{\|\boldsymbol{R}'_{[i]} \boldsymbol{W}_g^i\|_2}{\|\boldsymbol{R}'_{[i]}\|_2 \, \|\boldsymbol{W}_g^i\|_2}, \tag{12}$$

where λ is normalized by the spectral norm to constrain within $[0, 1]$. Table 4 compares the λ distributions, where the average values across experts per layer are reported. Compared to vanilla MoE, MoE with MPI achieves a tighter couple of router vectors with the principal directions of expert weights, manifest as a prominently higher λ.

The analysis in Section 3.3 explains why a single power iteration is sufficient to achieve effective router–expert alignment. A natural question is whether additional iterations could further improve

Table 4: Comparison of λ distributions. MoE with MPI exhibits an improved alignment between $\boldsymbol{R}'_{[i]}$ and the principal singular direction of expert weights, manifested by larger λ values.

Layer	1	2	3	4	5	6	7	8	9	10	11	12
MoE	0.37	0.31	0.31	0.28	0.28	0.25	0.24	0.23	0.24	0.24	0.22	0.26
w. MPI	0.67	0.66	0.69	0.70	0.68	0.67	0.64	0.63	0.62	0.62	0.62	0.69

performance by enforcing tighter alignment. To investigate this, we increase the number of power iterations to 10 and observe no further improvement in either convergence or downstream performance; instead, pretraining loss increases by 0.002–0.003 and downstream performance drops by 1.39 percentage. We argue that overly aggressive alignment may destabilize router optimization, making a single power iteration a more robust and efficient choice.

5.2 Ablation Studies

We conduct ablation studies to validate the design choices of routers with Manifold Power-Iteration.

5.2.1 Impact of the Key Design Choices

We pretrain 3B ablated models on 200B tokens to assess the individual contributions of these two core design components: (1) Power Iteration and (2) Router Retraction.

Ablation on Power Iteration Design. We introduce a baseline that simply performs row-wise normalization on router weights $\boldsymbol{R}$. This replaces the original $\boldsymbol{R}'_{[i]}$ with $\boldsymbol{R}^{\text{np}}_{[i]}$, which is given as:

$$\boldsymbol{R}^{np}_{[i]} = C \cdot \frac{\boldsymbol{R}_{[i]}}{\|\boldsymbol{R}_{[i]}\|_2}.$$

As shown in Figure 4, this ablated variant underperforms the full design, achieving nearly identical performance to the vanilla MoE. This confirms that the observed improvements cannot be attributed solely to router-weight retraction. However, we observe that it exhibits a similar balance loss distribution to that of our MPI. We leave it to future work to investigate whether this can lead to improved load balancing as a side benefit.

Router Retraction Enables Stable Training. To resolve the instability caused by power iteration, we adopt router retraction to mitigate the risk of L_2-Norm explosion or collapse. We replace $\boldsymbol{R}'_{[i]}$ with $\hat{\boldsymbol{R}}_{[i]}$ and first conduct ablation on 1B models.

Besides, we observe loss spikes and abnormal gradient behavior in 1B baselines pretrained with AdamW and Muon. While Hyperball Optimization alleviates this instability, it does not constrain expert spectral norms, leaving the router vulnerable to L_2-norm collapse as N increases. The ablated variant remains competitive on downstream tasks, yet incurs a 0.003 increase in pretraining loss. Our analysis, together with our empirical observations, highlights router retraction as a critical component for stable training.

Table 5: Validation perplexity across choices of C'.

C′	**1**	**2**	**4**	**8**	**MoE**
PPL	0.890	0.855	0.853	0.856	0.889

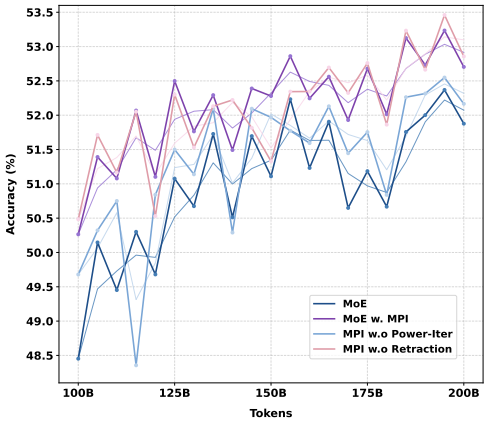


Figure 4: Ablation studies of our two key design components: (1) Power Iteration and (2) Router Retraction. Beyond these reported results, we observe training collapse when Retraction is removed from models trained with AdamW and Muon. This highlights its importance for stability, particularly for optimizers without explicit weight constraints.

5.2.2 Sensitivity Analysis of Constant C

We benchmark small-scale MoE models with 256 experts, conducting a hyperparameter search over over $C' \in \{1, 2, 4, 8\}$. Each model variant is pretrained on 50B tokens and optimized with MuonH. Table 5 presents the validation perplexity (PPL) across different choices of C'.

Specifically, we make the following observations: (1) In most cases, MPI outperforms the vanilla MoE, suggesting that it is relatively insensitive to the choice of C'; (2) the optimal choice of C' is consistent with the scaling principles established in Section 3.2. Leveraging the hyperparameter transferability of Hyperball Optimization, we directly transfer the C' identified in the small-scale sweep to the 11B setting. As shown in Section 4.2, no performance collapse is observed. This suggests that C' is largely insensitive and transferable when paired with advanced optimizer designs.

6 Compatibility with Existing Router Designs

Auxiliary loss for MoE. In standard MoE practice, auxiliary losses are commonly introduced to regularize routing or address issues such as load imbalance and insufficient expert specialization. Section 4.2 confirms that our router design is compatible with the load-balancing loss. We further combine MPI with router z-loss using a coefficient of 0.001 Zoph et al. (2022). In small-scale experiments, we observe no loss or gradient anomalies, and the z-loss variant improves downstream performance by 0.68 points, providing further evidence of compatibility.

Routing Score Functions. By default, we use $\mathrm{Softmax}$ as the routing score function. Here, we also explore $\mathrm{Sigmoid}$ as an alternative. Compared with $\mathrm{Softmax}$, the pretraining-loss advantage becomes smaller, while downstream performance still improves from 41.64 to 42.05. We leave a more thorough exploration of $\mathrm{Sigmoid}$ and other activation functions to future work.

7 Background on Advanced Optimizers

We provide an overview of the advanced optimizers used in this paper, which have been shown to accelerate model convergence. Beyond convergence acceleration, we are also interested in their scalability, as it may allow the empirical insights obtained from models of up to 11B parameters trained on 350B tokens to extend to larger-scale settings.

We begin with an introduction of Muon (Jordan et al., 2024), which orthogonalizes momentum via Newton-Schulz iterations (Schulz, 1933) to construct parameter updates. Recent studies have demonstrated its effectiveness in pretraining models with up to trillions of parameters (Team et al., 2026; DeepSeek-AI, 2026). Further analyses interpret Muon as steepest descent under the spectral norm, inspiring other norm-constrained optimizer designs (Pethick et al., 2025). More recently, a line of work has explored imposing norm constraints on both weights and updates (Wen et al., 2026; Xie et al., 2026). The underlying intuition is that constraining weight norms promotes stable and scalable optimization, which can in turn accelerate convergence across scales and facilitate hyperparameter transfer. In this work, we provide an initial empirical study of these optimizers and further validate their effectiveness in large-scale MoE pretraining.

8 Conclusion

In this paper, we revisited MoE router design by treating each router row as a proxy for its corresponding expert. Based on this view, we proposed Manifold Power Iteration (MPI), which ties router weights to the parameter structure of its associated expert. To couple router weights with expert parameters, we adopt a "Power-then-Retract" paradigm, where power iteration steers each router row toward the principal singular direction of its associated expert, and retraction constrains its norm. We theoretically establish the convergence behavior of this router design and further show that MPI incurs negligible training overhead while remaining scalable. Experiments across different model sizes, architectures, and training settings consistently demonstrate its effectiveness. We hope this work can encourage further exploration of router design and inspire more principled router design.

AI USE STATEMENT

In this work, we used generative AI tools to assist with language editing. The core implementation of the proposed method was developed independently by the authors. All AI-assisted content, including identified literature and its interpretation, was independently reviewed and verified against the original sources. The authors take full responsibility for the final content of this paper.

ETHICS STATEMENT

This work studies an architectural improvement for Mixture-of-Experts (MoE) models. As the proposed method does not involve human subjects, sensitive data, or application-specific decision-making, we do not identify ethical concerns beyond those generally associated with the development and deployment of large language models.

REPRODUCIBILITY STATEMENT

We provide the model configurations, training hyperparameters, optimization settings, and evaluation protocols necessary to reproduce our experiments in the main text and Appendix. Additional implementation details are documented where relevant, and the corresponding code will be released when accepted.

REFERENCES

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.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*, 2018.

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.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. A dataset of information-seeking questions and answers anchored in research papers. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4599–4610, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.365. URL https://aclanthology.org/2021.naacl-main.365/.

DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2368–2378, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1246. URL https://aclanthology.org/N19-1246/.

Trevor Gale, Deepak Narayanan, Cliff Young, and Matei Zaharia. Megablocks: Efficient sparse training with mixture-of-experts. *Proceedings of Machine Learning and Systems*, 5:288–304, 2023.

GLM-5-Team, :, Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, Chenzheng Zhu, Congfeng Yin, Cunxiang Wang, Gengzheng Pan, Hao Zeng, Haoke Zhang, Haoran Wang, Huilong Chen, Jiajie Zhang, Jian Jiao, Jiaqi Guo, Jingsen Wang, Jingzhao Du, Jinzhu Wu, Kedong Wang, Lei Li, Lin Fan, Lucen Zhong, Mingdao Liu, Mingming Zhao, Pengfan Du, Qian Dong, Rui Lu, Shuang-Li, Shulin Cao, Song Liu, Ting Jiang, Xiaodong Chen, Xiaohan Zhang, Xuancheng Huang, Xuezhen Dong, Yabo Xu, Yao Wei, Yifan An, Yilin Niu, Yitong Zhu, Yuanhao Wen, Yukuo Cen, Yushi Bai, Zhongpei Qiao, Zihan Wang, Zikang Wang, Zilin Zhu, Ziqiang Liu, Zixuan Li, Bojie Wang, Bosi Wen, Can Huang, Changpeng Cai, Chao Yu, Chen Li, Chengwei Hu, Chenhui Zhang, Dan Zhang, Daoyan Lin, Dayong Yang, Di Wang, Ding Ai, Erle Zhu, Fangzhou Yi, Feiyu Chen, Guohong Wen, Hailong Sun, Haisha Zhao, Haiyi Hu, Hanchen Zhang, Hanrui Liu, Hanyu Zhang, Hao Peng, Hao Tai, Haobo Zhang, He Liu, Hongwei Wang, Hongxi Yan, Hongyu Ge, Huan Liu, Huanpeng Chu, Jia'ni Zhao, Jiachen Wang, Jiajing Zhao, Jiamin Ren, Jiapeng Wang, Jiaxin Zhang, Jiayi Gui, Jiayue Zhao, Jijie Li, Jing An, Jing Li, Jingwei Yuan, Jinhua Du, Jinxin Liu, Junkai Zhi, Junwen Duan, Kaiyue Zhou, Kangjian Wei, Ke Wang, Keyun Luo, Laiqiang Zhang, Leigang Sha, Liang Xu, Lindong Wu, Lintao Ding, Lu Chen, Minghao Li, Nianyi Lin, Pan Ta, Qiang Zou, Rongjun Song, Ruiqi Yang, Shangqing Tu, Shangtong Yang, Shaoxiang Wu, Shengyan Zhang, Shijie Li, Shuang Li, Shuyi Fan, Wei Qin, Wei Tian, Weining Zhang, Wenbo Yu, Wenjie Liang, Xiang Kuang, Xiangmeng Cheng, Xiangyang Li, Xiaoquan Yan, Xiaowei Hu, Xiaoying Ling, Xing Fan, Xingye Xia, Xinyuan Zhang, Xinze Zhang, Xirui Pan, Xu Zou, Xunkai Zhang, Yadi Liu, Yandong Wu, Yanfu Li, Yidong Wang, Yifan Zhu, Yijun Tan, Yilin Zhou, Yiming Pan, Ying Zhang, Yinpei Su, Yipeng Geng, Yong Yan, Yonglin Tan, Yuean Bi, Yuhan Shen, Yuhao Yang, Yujiang Li, Yunan Liu, Yunqing Wang, Yuntao Li, Yurong Wu, Yutao Zhang, Yuxi Duan, Yuxuan Zhang, Zezhen Liu, Zhengtao Jiang, Zhenhe Yan, Zheyu Zhang, Zhixiang Wei, Zhuo Chen, Zhuoer Feng, Zijun Yao, Ziwei Chai, Ziyuan Wang, Zuzhou Zhang, Bin Xu, Minlie Huang, Hongning Wang, Juanzi Li, Yuxiao Dong, and Jie Tang. Glm-5: from vibe coding to agentic engineering, 2026. URL https://arxiv.org/abs/2602.15763.

Gene H. Golub and Charles F. Van Loan. *Matrix computations (3rd ed.)*. Johns Hopkins University Press, USA, 1996. ISBN 0801854148.

Yuling Gu, Oyvind Tafjord, Bailey Kuehl, Dany Haddad, Jesse Dodge, and Hannaneh Hajishirzi. Olmes: A standard for language model evaluations, 2025. URL https://arxiv.org/abs/2406.08446.

Nathan Halko, Per-Gunnar Martinsson, and Joel A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions, 2010. URL https://arxiv.org/abs/0909.4061.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021. URL https://arxiv.org/abs/2009.03300.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. What disease does this patient have? a large-scale open domain question answering dataset from medical exams, 2020. URL https://arxiv.org/abs/2009.13081.

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

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In Regina Barzilay and Min-Yen Kan (eds.), *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1147. URL https://aclanthology.org/P17-1147/.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://aclanthology.org/Q19-1026/.

Jon M. Laurent, Joseph D. Janizek, Michael Ruzo, Michaela M. Hinks, Michael J. Hammerling, Siddharth Narayanan, Manvitha Ponnapati, Andrew D. White, and Samuel G. Rodriques. Lab-bench: Measuring capabilities of language models for biology research, 2024. URL https://arxiv.org/abs/2407.10362.

Wanchao Liang, Tianyu Liu, Less Wright, Will Constable, Andrew Gu, Chien-Chin Huang, Iris Zhang, Wei Feng, Howard Huang, Junjie Wang, Sanket Purandare, Gokul Nadathur, and Stratos Idreos. Torchtitan: One-stop pytorch native solution for production ready LLM pretraining. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=SFN6Wm7YBI.

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

Anton Lozhkov, Loubna Ben Allal, Leandro von Werra, and Thomas Wolf. Fineweb-edu: the finest collection of educational content, 2024. URL https://huggingface.co/datasets/HuggingFaceFW/fineweb-edu.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, Yuling Gu, Shane Arora, Akshita Bhagia, Dustin Schwenk, David Wadden, Alexander Wettig, Binyuan Hui, Tim Dettmers, Douwe Kiela, Ali Farhadi, Noah A. Smith, Pang Wei Koh, Amanpreet Singh, and Hannaneh Hajishirzi. Olmoe: Open mixture-of-experts language models, 2025. URL https://arxiv.org/abs/2409.02060.

Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, Jacob Morrison, Jake Poznanski, Kyle Lo, Luca Soldaini, Matt Jordan, Mayee Chen, Michael Noukhovitch, Nathan Lambert, Pete Walsh, Pradeep Dasigi, Robert Berry, Saumya Malik, Saurabh Shah, Scott Geng, Shane Arora, Shashank Gupta, Taira Anderson, Teng Xiao, Tyler Murray, Tyler Romero, Victoria Graf, Akari Asai, Akshita Bhagia, Alexander Wettig, Alisa Liu, Aman Rangapur, Chloe Anastasiades, Costa Huang, Dustin Schwenk, Harsh Trivedi, Ian Magnusson, Jaron Lochner, Jiacheng Liu, Lester James V. Miranda, Maarten Sap, Malia Morgan, Michael Schmitz, Michal Guerquin, Michael Wilson, Regan Huff, Ronan Le Bras, Rui Xin, Rulin Shao, Sam Skjonsberg, Shannon Zejiang Shen, Shuyue Stella Li, Tucker Wilde, Valentina Pyatkin, Will Merrill, Yapei Chang, Yuling Gu, Zhiyuan Zeng, Ashish Sabharwal, Luke Zettlemoyer, Pang Wei Koh, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. Olmo 3, 2025. URL https://arxiv.org/abs/2512.13961.

OpenAI. gpt-oss-120b & gpt-oss-20b model card, 2025. URL https://arxiv.org/abs/2508.10925.

Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. Medmcqa: A large-scale multi-subject multi-choice dataset for medical domain question answering. In Gerardo Flores, George H Chen, Tom Pollard, Joyce C Ho, and Tristan Naumann (eds.), *Proceedings of the Conference on Health, Inference, and Learning*, volume 174 of *Proceedings of Machine Learning Research*, pp. 248–260. PMLR, 07–08 Apr 2022. URL https://proceedings.mlr.press/v174/pal22a.html.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In Katrin Erk and Noah A. Smith (eds.),

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 1525–1534, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1144. URL https://aclanthology.org/P16-1144/.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf.

Thomas Pethick, Wanyun Xie, Kimon Antonakopoulos, Zhenyu Zhu, Antonio Silveti-Falls, and Volkan Cevher. Training deep learning models with norm-constrained LMOs. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=2Oqm2IzTy9.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models, 2020. URL https://arxiv.org/abs/1910.02054.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In Jian Su, Kevin Duh, and Xavier Carreras (eds.), *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL https://aclanthology.org/D16-1264/.

Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions, 2017. URL https://arxiv.org/abs/1710.05941.

Siva Reddy, Danqi Chen, and Christopher D. Manning. CoQA: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266, 2019. doi: 10.1162/tacl_a_00266. URL https://aclanthology.org/Q19-1016/.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *arXiv preprint arXiv:1907.10641*, 2019.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. Socialiqa: Commonsense reasoning about social interactions, 2019. URL https://arxiv.org/abs/1904.09728.

Günther Schulz. Iterative berechung der reziproken matrix. *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik*, 13(1):57–59, 1933. doi: https://doi.org/10.1002/zamm.19330130111. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/zamm.19330130111.

Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

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.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4149–4158, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1421. URL https://aclanthology.org/N19-1421/.

Kimi Team, Yifan Bai, Yiping Bao, Y. Charles, Cheng Chen, Guanduo Chen, Haiting Chen, Huarong Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, Zhuofu Chen, Jialei Cui, Hao Ding, Mengnan Dong, Angang Du, Chenzhuang Du, Dikang Du, Yulun Du, Yu Fan, Yichen Feng, Kelin Fu, Bofei Gao, Chenxiao Gao, Hongcheng Gao, Peizhong Gao, Tong Gao, Yuyao Ge, Shangyi Geng, Qizheng Gu, Xinran Gu, Longyu Guan, Haiqing Guo, Jianhang Guo, Xiaoru Hao, Tianhong He, Weiran He, Wenyang He, Yunjia He, Chao Hong, Hao Hu, Yangyang Hu, Zhenxing Hu, Weixiao Huang, Zhiqi Huang, Zihao Huang, Tao Jiang, Zhejun Jiang, Xinyi Jin, Yongsheng Kang, Guokun Lai, Cheng Li, Fang Li, Haoyang Li, Ming Li, Wentao Li, Yang Li, Yanhao Li, Yiwei Li, Zhaowei Li, Zheming Li, Hongzhan Lin, Xiaohan Lin, Zongyu Lin, Chengyin Liu, Chenyu Liu, Hongzhang Liu, Jingyuan Liu, Junqi Liu, Liang Liu, Shaowei Liu, T. Y. Liu, Tianwei Liu, Weizhou Liu, Yangyang Liu, Yibo Liu, Yiping Liu, Yue Liu, Zhengying Liu, Enzhe Lu, Haoyu Lu, Lijun Lu, Yashuo Luo, Shengling Ma, Xinyu Ma, Yingwei Ma, Shaoguang Mao, Jie Mei, Xin Men, Yibo Miao, Siyuan Pan, Yebo Peng, Ruoyu Qin, Zeyu Qin, Bowen Qu, Zeyu Shang, Lidong Shi, Shengyuan Shi, Feifan Song, Jianlin Su, Zhengyuan Su, Lin Sui, Xinjie Sun, Flood Sung, Yunpeng Tai, Heyi Tang, Jiawen Tao, Qifeng Teng, Chaoran Tian, Chensi Wang, Dinglu Wang, Feng Wang, Hailong Wang, Haiming Wang, Jianzhou Wang, Jiaxing Wang, Jinhong Wang, Shengjie Wang, Shuyi Wang, Si Wang, Xinyuan Wang, Yao Wang, Yejie Wang, Yiqin Wang, Yuxin Wang, Yuzhi Wang, Zhaoji Wang, Zhengtao Wang, Zhengtao Wang, Zhexu Wang, Chu Wei, Qianqian Wei, Haoning Wu, Wenhao Wu, Xingzhe Wu, Yuxin Wu, Chenjun Xiao, Jin Xie, Xiaotong Xie, Weimin Xiong, Boyu Xu, Jinjing Xu, L. H. Xu, Lin Xu, Suting Xu, Weixin Xu, Xinran Xu, Yangchuan Xu, Ziyao Xu, Jing Xu, Jing Xu, Junjie Yan, Yuzi Yan, Hao Yang, Xiaofei Yang, Yi Yang, Ying Yang, Zhen Yang, Zhilin Yang, Zonghan Yang, Haotian Yao, Xingcheng Yao, Wenjie Ye, Zhuorui Ye, Bohong Yin, Longhui Yu, Enming Yuan, Hongbang Yuan, Mengjie Yuan, Siyu Yuan, Haobing Zhan, Dehao Zhang, Hao Zhang, Wanlu Zhang, Xiaobin Zhang, Yadong Zhang, Yangkun Zhang, Yichi Zhang, Yizhi Zhang, Yongting Zhang, Yu Zhang, Yutao Zhang, Yutong Zhang, Zheng Zhang, Haotian Zhao, Yikai Zhao, Zijia Zhao, Huabin Zheng, Shaojie Zheng, Longguang Zhong, Jianren Zhou, Xinyu Zhou, Zaida Zhou, Jinguo Zhu, Zhen Zhu, Weiyu Zhuang, and Xinxing Zu. Kimi k2: Open agentic intelligence, 2026. URL https://arxiv.org/abs/2507.20534.

David Wadden, Kejian Shi, Jacob Morrison, Alan Li, Aakanksha Naik, Shruti Singh, Nitzan Barzilay, Kyle Lo, Tom Hope, Luca Soldaini, Shannon Zejiang Shen, Doug Downey, Hannaneh Hajishirzi, and Arman Cohan. SciRIFF: A resource to enhance language model instruction-following over scientific literature. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 6072–6109, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.310. URL https://aclanthology.org/2025.emnlp-main.310/.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. In Leon Derczynski, Wei Xu, Alan Ritter, and Tim Baldwin (eds.), *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pp. 94–106, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4413. URL https://aclanthology.org/W17-4413/.

Kaiyue Wen, David Leo Wright Hall, Tengyu Ma, and Percy Liang. Fantastic pretraining optimizers and where to find them. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=2J51qUZ0iG.

Tian Xie, Haoming Luo, Haoyu Tang, Yiwen Hu, Jason Klein Liu, Qingnan Ren, Yang Wang, Wayne Xin Zhao, Rui Yan, Bing Su, Chong Luo, and Baining Guo. Controlled llm training on spectral sphere, 2026. URL https://arxiv.org/abs/2601.08393.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472/.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. St-moe: Designing stable and transferable sparse expert models, 2022. URL https://arxiv.org/abs/2202.08906.

A Supplementary Derivations for Approximation in Equation 11

We provide detailed derivation for weights update $\Delta \boldsymbol{r}$ of router $\boldsymbol{R}'_{[i]}$ with Manifold Power Iteration:

$$\Delta \boldsymbol{r}_M = \frac{\boldsymbol{R}'_{[i]}\boldsymbol{M}}{\|\boldsymbol{R}'_{[i]}\boldsymbol{M}\|_2} - \boldsymbol{R}'_{[i]},$$

We project $\boldsymbol{R}'_{[i]}\boldsymbol{M}$ onto the subspace spanned by $\boldsymbol{R}'_{[i]}$ and its orthogonal complement is given as:

$$\begin{aligned} \boldsymbol{R}'_{[i]}\boldsymbol{M} \quad &= \quad \boldsymbol{R}'_{[i]}\left(\boldsymbol{R}'_{[i]}\boldsymbol{M}\boldsymbol{R}'^{\top}_{[i]}\right) \\ &+ \underbrace{\left(\boldsymbol{R}'_{[i]}\boldsymbol{M} - \boldsymbol{R}'_{[i]}\left(\boldsymbol{R}'_{[i]}\boldsymbol{M}\boldsymbol{R}'^{\top}_{[i]}\right)\right)}_{\text{orthogonal to } \boldsymbol{R}'_{[i]}}. \end{aligned}$$

As the power iteration proceeds, $\boldsymbol{R}'_{[i]}$ gradually converges toward the dominant subspace, while the orthogonal component becomes negligible. Consequently, we obtain the following approximation:

$$\frac{\boldsymbol{R}'_{[i]}\boldsymbol{M}}{\|\boldsymbol{R}'_{[i]}\boldsymbol{M}\|_2} \;\approx\; \boldsymbol{R}'_{[i]} + \frac{\boldsymbol{R}'_{[i]}\boldsymbol{M} - \boldsymbol{R}'_{[i]}(\boldsymbol{R}'_{[i]}\boldsymbol{M}\boldsymbol{R}'^{\top}_{[i]})}{\boldsymbol{R}'_{[i]}\boldsymbol{M}\boldsymbol{R}'^{\top}_{[i]}}.$$

Through a simple rearrangement of terms, we obtain

$$\Delta \boldsymbol{r}_M \approx \frac{1}{\boldsymbol{R}'_{[i]}\boldsymbol{M}\boldsymbol{R}'^{\top}_{[i]}}\Big(\boldsymbol{R}'_{[i]}\boldsymbol{M} - \boldsymbol{R}'_{[i]}(\boldsymbol{R}'_{[i]}\boldsymbol{M}\boldsymbol{R}'^{\top}_{[i]})\Big).$$

This completes the derivation and recovers the approximate update form presented in Eq. 11.

B Details for Pretraining Experiments

B.1 Implementation Details

Table 6 summarizes the architectural hyperparameters used across our experiments. To enable larger-scale evaluation, we scale the 3B model to 11B by increasing the number of routed experts from 64 to 256, resulting in a sparse MoE model with 11.36B total parameters and 823M activated parameters.

Our training pipeline is built upon the TorchTitan framework (Liang et al., 2025). For Transformer components, we use PyTorch's scaled dot-product attention (SDPA) (Paszke et al., 2019) for efficient attention computation and MegaBlocks MLP (Gale et al., 2023) for the efficient implementation of sparse MoE layers. For distributed training, we adopt Fully Sharded Data Parallel (FSDP) (Rajbhandari et al., 2020) across all pretraining experiments to shard model states and support training at larger model scales.

Table 6: Hyperparameters of model architectures.

Configuration	1 B	3 B	11 B
Dimension	1024	1536	1536
Layers	8	12	12
Attention Heads	8	16	16
Key-value Heads	8	8	8
RoPE Theta (θ)	500,000	500,000	500,000
FFN Dimension	512	768	768
Activated Experts	8	8	8
Routed Experts	64	64	256
Sequence Length	2048	4096	4096
Vocab Size	128,256	128,256	128,256
# Dense Params	296 M	479 M	479 M
# Sparse Params	806 M	2.72 B	10.88 B
# Activated Params	397 M	823 M	823 M
# Total Params	1.10 B	3.20 B	11.36 B

B.2 Optimizer Setup Details

Table 7 presents the hyperparameters used for pretraining the 1B MoE model with AdamW. This parameter set was identified through our hyperparameter search over key configurations. For pretraining 1B models with other optimizers, we simply align their update RMS with AdamW. This ensures fair convergence comparisons and avoid the cost for extensive hyperparameter search.

For Muon, matching the update RMS requires scaling the learning rate of Muon-optimized parameters by a factor of $0.2 \times \sqrt{\max(d_{in}, d_{out})}$ (Team et al., 2026). We follow this scaling rule throughout our experiments, with further implementation details provided in the released code.

Table 7: Pretraining hyperparameters.

Configuration	**1B**
Optimizer	AdamW
(β_1, β_2)	$(0.9, 0.95)$
Peak LR	1.0E-03
Minimum LR	0
LR scheduler	cosine
Warmup steps	1000
Weight decay	0.1
Gradient clip	1.0
LBL weight	0.01

For Hyperball Optimization, we constrain the Frobenius norm of each weight matrix $\boldsymbol{W} \in \mathbb{R}^{d_{in} \times d_{out}}$ to $\sqrt{d_{out}}$. Under this norm constraint, matching the update RMS yields a corresponding learning-rate scaling factor of $0.2 \times \sqrt{d_{in}}$. Rather than allowing this factor to vary with model width, we instantiate it using the d_{in} of the 1B model and keep the resulting value fixed across all model scales. This provides a scale-invariant reference for the lr adjustment and ensures a consistent optimization setup when scaling from smaller to larger models.

B.3 Evaluation Setup

We perform all downstream evaluations using OLMES (Gu et al., 2025). During pretraining, we evaluate model checkpoints on nine core tasks:ARC-Easy (Clark et al., 2018), ARC-Challenge (Clark et al., 2018), MMLU (Hendrycks et al., 2021), CommonsenseQA (Talmor et al., 2019), SocialIQA (Sap et al., 2019), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2019), PIQA (Bisk et al., 2020), and SciQ (Welbl et al., 2017). To efficiently track their general capabilities throughout training. Unless otherwise specified, we follow the evaluation setup of OLMo 3 (Olmo et al., 2025) and report results on a benchmark comprising 25 multiple-choice tasks. The complete task list is provided in Table 8. To save space, we omit the per-task results for these 25 benchmarks unless otherwise specified.

	AdamW		AdamH		Muon		MuonH	
	MoE	*w.* MPI	MoE	*w.* MPI	MoE	*w.* MPI	MoE	*w.* MPI
Arc-Easy Clark et al. (2018)	59.64	64.05	64.09	66.88	60.65	62.16	61.82	65.36
Arc-Challenge Clark et al. (2018)	32.51	36.26	35.15	37.37	34.90	35.75	35.24	33.62
MMLU-Stem Hendrycks et al. (2021)	28.13	27.80	28.19	28.66	28.26	27.60	27.87	28.33
MMLU-Humanities Hendrycks et al. (2021)	34.38	34.68	35.03	35.55	33.77	34.38	33.96	34.51
MMLU-Social Science Hendrycks et al. (2021)	29.12	29.12	28.20	29.08	28.48	29.50	29.25	29.07
MMLU-Other Hendrycks et al. (2021)	33.41	35.84	35.47	37.05	35.35	35.84	33.47	36.64
CSQA Talmor et al. (2019)	38.25	48.73	38.00	48.08	44.14	45.37	42.83	46.76
HellaSwag Zellers et al. (2019)	47.18	47.68	48.52	48.35	46.52	48.16	46.31	45.24
WinoGrande Sakaguchi et al. (2019)	52.25	51.93	51.70	51.93	51.78	52.01	52.25	51.93
SocialIQA Sap et al. (2019)	41.91	44.58	42.02	43.86	42.99	44.11	41.91	43.86
PiQA Bisk et al. (2020)	69.04	68.82	69.04	69.91	68.61	69.64	67.74	67.90
CoQA Reddy et al. (2019)	47.09	52.39	47.00	37.84	54.37	53.61	51.27	54.00
DROP Dua et al. (2019)	30.79	27.71	26.83	29.03	28.53	28.34	28.72	28.15
Jeopardy	46.92	43.82	50.02	54.51	49.02	49.64	52.98	53.36
NaturalQs Kwiatkowski et al. (2019)	24.31	25.76	23.71	25.57	25.31	23.51	24.21	26.11
SQuAD Rajpurkar et al. (2016)	49.91	48.96	50.47	51.04	49.86	48.63	53.65	52.70
SciQ Welbl et al. (2017)	70.90	76.70	74.50	76.80	73.20	73.60	72.30	72.60
QASPER Dasigi et al. (2021)	67.40	68.02	58.93	67.40	62.07	68.03	61.44	67.40
DBQA Laurent et al. (2024)	27.12	23.84	26.15	25.38	24.81	26.54	23.85	26.35
ProtocalQA Laurent et al. (2024)	25.00	27.78	26.85	27.78	30.56	27.78	26.85	29.63
Lambada Paperno et al. (2016)	39.34	39.39	41.57	40.15	39.76	39.24	38.95	40.99
MedMCQA Pal et al. (2022)	27.16	28.26	26.99	28.66	27.32	28.16	28.42	28.14
MedQA Jin et al. (2020)	23.17	23.80	24.04	24.90	24.90	23.57	21.84	23.96
SciRIFF Wadden et al. (2025)	72.44	72.44	71.93	72.82	71.11	71.87	72.44	72.00
Basic Skills Olmo et al. (2025)	39.23	40.73	40.27	39.73	39.03	39.54	39.90	40.78
Avg RC Acc.	42.26	43.93	42.59	43.93	43.01	43.55	42.78	43.98

Table 8: Task-specific performance comparisons for 1B MoE with different optimizers.

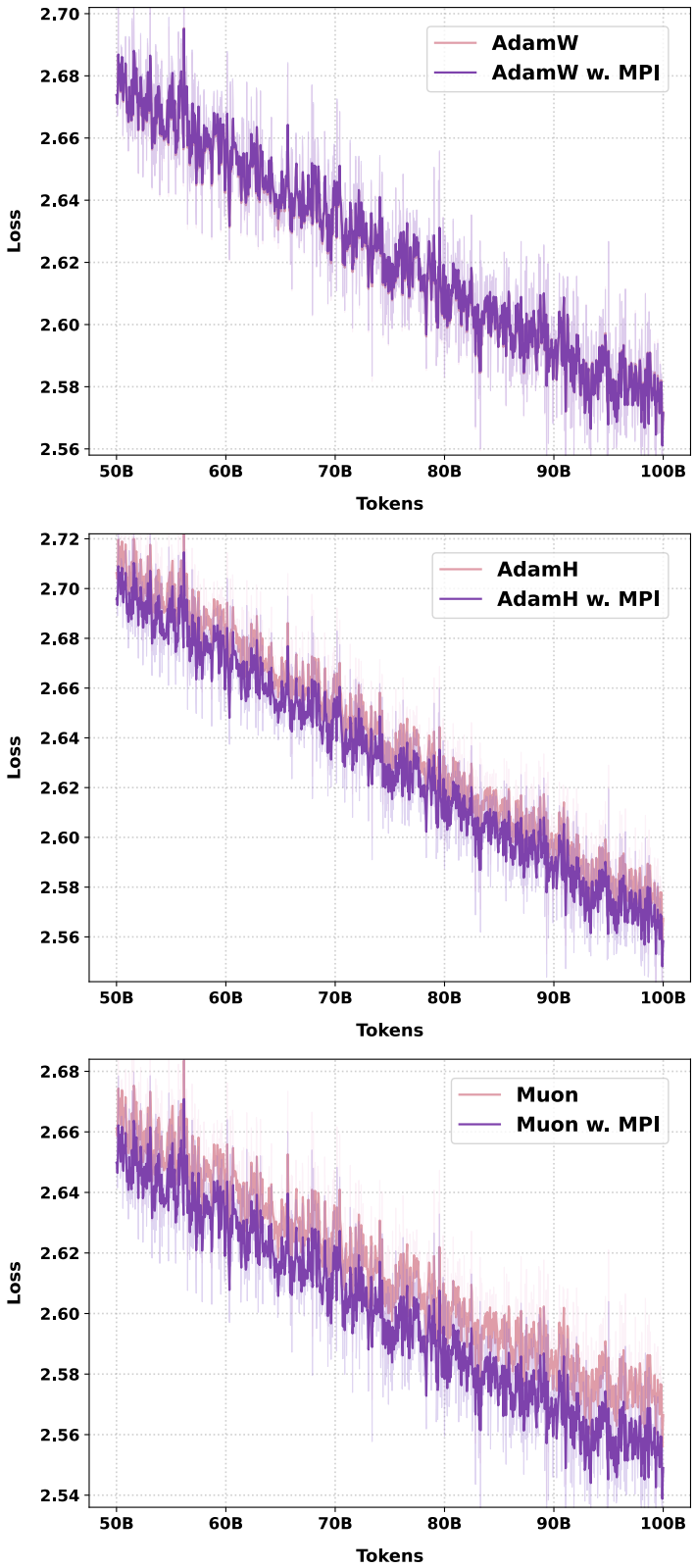


Figure 5: Pre-training loss comparison for a 1B MoE model across optimizers (AdamW, AdamH, Muon). MoE with MPI achieves a convergence advantages over all alternative setups.