LoREAL: Low-Rank Expert-Input Adaptation for Mixture-of-Experts Language Models

Anonymous authors
Paper under double-blind review

Abstract

Mixture-of-Experts (MoE) language models use the same token representation to select experts and to compute their outputs. Downstream adaptation can therefore act on two distinct pathways, routing and expert computation. We propose Low-Rank Expert-Input Adaptation (LoREAL), which separates these pathways and adapts only the inputs delivered to the experts. At each MoE layer, the frozen routing pathway uses the layer input to select experts and compute mixture weights. In parallel, the adaptive expert-input pathway passes the same input through a shared low-rank residual adapter before sending it to the selected frozen experts. The local adapter thus changes the expert inputs without changing the input used for routing at that layer. Earlier adapted layers may still affect later routing decisions. The shared adapter introduces $2dr$ trainable parameters per layer (hidden size d, rank r), with optional per-expert scaling adding one scalar for each of the N experts. On OLMoE-1B-7B, LoREAL outperforms router tuning across continued pretraining and supervised fine-tuning, reduces measured forgetting relative to router tuning, and improves on expert LoRA in most comparisons at comparable parameter budgets. Ablations support adapting the expert-input pathway as an effective alternative to increasing router capacity. Code is available at `https://anonymous.4open.science/r/anonymous-code-1FD5/`.

1 Introduction

Mixture-of-Experts (MoE) language models have emerged as an effective approach to scaling model capacity without proportionally increasing the computation required for each token (Shazeer et al., 2017; Lepikhin et al., 2021; Fedus et al., 2022; Jiang et al., 2024; DeepSeek-AI, 2024). Instead of activating all model parameters, an MoE layer consists of an expert pool together with a router that dynamically selects a small subset of experts for each input token and determines how their outputs are combined. Through large-scale pretraining, the expert pool acquires diverse capabilities distributed across individual experts. Effectively harnessing these pretrained expert capabilities is therefore key to realizing the full potential of MoE models.

However, base MoE models often exhibit limited performance on long-tail domains that are underrepresented during pretraining, such as mathematics and code. Recent studies attribute this limitation to suboptimal routing policies on downstream tasks, motivating router-tuning (RT) approaches (Yoon et al., 2026; Bai et al., 2025) that adapt routing policies while largely keeping the expert parameters fixed. In particular, Routing Manifold Alignment (RoMA) (Li et al., 2026) fine-tunes the router using correctly predicted samples to regularize task-specific routing patterns, while Re-Routing in Test-Time (R2-T2) (Li et al., 2025) performs per-sample test-time re-routing by leveraging nearby successful samples as references.

However, RT at the token level remains challenging, as different tokens can exhibit conflicting optimization directions, making it difficult for a single router to accommodate their heterogeneous adaptation needs (Yang et al., 2025), even with larger router parameters and more extensive training as shown in Table 4. Moreover, the router and experts are jointly optimized during pretraining. The router determines the input distribution of each expert, and each expert specializes under that distribution (Chi et al., 2022; Lv et al., 2026). Altering routing decisions separately may therefore disrupt this pretrained coupling, leads to greater forgetting on continued pretraining (CPT) tasks

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.

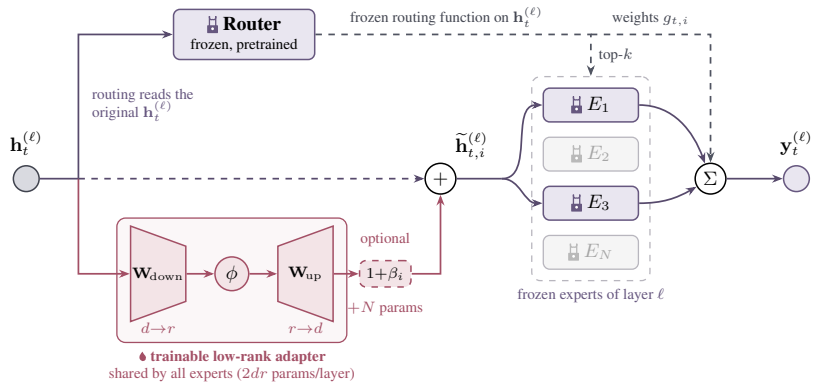


Figure 1: LoREAL separates router and expert within each MoE layer. The frozen router reads the layer input, while a shared trainable adapter adds a residual correction to the inputs delivered to frozen experts. The dashed block applies expert-specific scaling to this correction (Eq. 8).

illustrated in Table 2. Thus, improving downstream performance under a limited training budget while maximally preserving pretrained knowledge remains challenging.

To discuss this problem, instead of focusing on misrouting of input tokens, we first examine the inputs that experts actually receive on downstream data (Section 3.1). According to principal component analysis (PCA), most expert inputs remain in the same geometric region as the pretraining data along the leading principal components, suggesting that the pretrained experts can already accommodate most downstream inputs. The remaining distribution shift is concentrated along only a few specific directions, indicating a localized, subspace-specific drift of the representation space.

Motivated by this observation, we propose Low-Rank Expert-Input Adaptation (LoREAL), as illustrated in Figure 1. LoREAL separates each MoE layer into two input pathways. The routing pathway preserves the pretrained routing policy with the original hidden states and a frozen router. The expert pathway applies a low-rank residual transformation to the expert inputs for downstream adaptation. A single adapter is shared across all experts within each layer. An optional per-expert scalar scales the correction for each expert.

Our main contributions are threefold.

- We analyze expert inputs on downstream data and find that they largely remain within the pretrained representation geometry, with distribution shifts concentrated along only a few specific directions. This suggests that much of the pretrained expert capability remains reusable for downstream adaptation.
- We propose **Low-Rank Expert-Input Adaptation (LoREAL)**. To our knowledge, LoREAL is among the first training-based adaptation methods to explicitly separate the routing and expert-input pathways within a pretrained MoE layer, where the frozen router operates on the original hidden states while only the inputs to the selected experts are adaptively transformed.
- LoREAL consistently outperforms RT across both CPT and supervised fine-tuning (SFT) tasks, achieving lower CPT perplexity (6.47 vs. 7.00) and substantially less forgetting (0.60% vs. 3.03%), while also outperforming expert LoRA in most settings.

2 Related Work

Mixture-of-Experts language models. Sparse MoE layers scale the parameter count while keeping per-token computation nearly constant by routing each token to a small subset of experts (Shazeer et al., 2017; Lepikhin et al., 2021; Fedus et al., 2022; Zoph et al., 2022), and now underlie many open models (Jiang et al., 2024; Dai et al., 2024; DeepSeek-AI, 2024). Because the router determines which inputs are routed to each expert during pretraining, the two components co-adapt. Experts specialise on the input distribution the router assigns to them (Dai et al., 2024; Chen et al., 2022), routing representations collapse onto a few directions of the hidden space (Chi et al., 2022), and auxiliary losses have been proposed to strengthen the alignment between experts and

routers (Lv et al., 2026). Token-level gradient conflicts inside an expert (Yang et al., 2025) further show that *what* an expert receives, and not only *which* expert is chosen, shapes its behaviour. We build on this view and treat the pretrained router–expert coupling as an asset to be preserved rather than a constraint to be relaxed.

Adapting MoE models through routing. A line of work attributes the downstream gap of pretrained MoE models to routing and adapts the router while keeping the experts frozen. Router-only training elicits specialised experts (Bai et al., 2025) and has also been used to make depth dynamic (He et al., 2025). RA-MoE aligns routing with the downstream objective (Deng et al., 2026), RoMA aligns the routing manifold with task structure (Li et al., 2026), and R2-T2 re-routes tokens at test time (Li et al., 2025). Counterfactual analyses identify token-dependent routing failures (Yoon et al., 2026). Our aggregate coverage measurements do not rule out such failures. LoREAL instead keeps the routing function fixed and learns a low-rank expert-input adapter for CPT and SFT, changing expert inputs without directly modifying the local routing function. Its frozen routing function can still receive altered inputs from earlier adapted layers. RARE (Zhang et al., 2026) instead projects representation-steering perturbations into the router's null space and corrects propagated routing drift. LoREAL learns an expert-input adapter for CPT and SFT without a null-space constraint or a guarantee of end-to-end routing equality.

Parameter-efficient adaptation. Adapters (Houlsby et al., 2019) and LoRA (Hu et al., 2022) adapt frozen models through small trainable modules, and have been extended to MoE by attaching adaptation modules to experts or routing among a mixture of adapters (Liu et al., 2025; Wang et al., 2022), or by fine-tuning only the experts most relevant to a task (Wang et al., 2024). These approaches modify the computation performed by the experts. LoREAL instead modifies the input the experts receive, through a shared nonlinear bottleneck adapter whose correction lies in a learned subspace whose dimension is at most the adapter rank r (Section 3.2). The default adapter's parameter count is independent of the number of experts, enabling larger adapter ranks under a fixed budget. Combining it with expert-side adaptation remains future work.

3 METHOD

3.1 COVERAGE AND DRIFT OF DOWNSTREAM EXPERT INPUTS

A standard MoE layer with N experts maps an input representation $\mathbf{h} \in \mathbb{R}^d$ to

$$\mathbf{y} = \sum_{i=1}^{N} g_i(\mathbf{h}) E_i(\mathbf{h}), \tag{1}$$

where E_i is the i-th expert and $g_i(\mathbf{h})$ its routing weight. In a sparse MoE, only the top-k experts receive a nonzero weight for each input.

To understand a pretrained MoE model's suboptimal performance on downstream tasks, rather than attributing the performance gap directly to router drift or misrouting (Yoon et al., 2026; Bai et al., 2025), we examine the distributions of expert inputs. Specifically, we compare the pretraining-time input distribution under which each expert acquired its capabilities with the downstream input distribution, and characterize their overlap, coverage, and distributional shift.

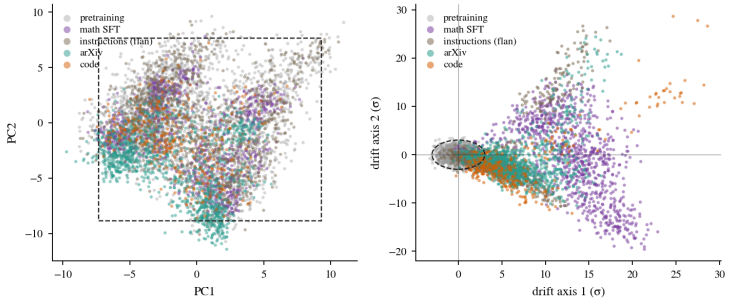
Definition (Distribution-Coverage Radius). Given a reference set $\mathcal{R} \subset \mathbb{R}^d$ and a query set $\mathcal{Q} \subset \mathbb{R}^d$, we define the coverage radius of an input $\mathbf{h}$ with respect to $\mathcal{R}$ as its mean Euclidean distance to the K nearest representations in $\mathcal{R}$:

$$r(\mathbf{h}; \mathcal{R}) = \frac{1}{K} \sum_{j=1}^{K} \left\| \mathbf{h} - \mathbf{h}_{\mathcal{R}}^{(j)} \right\|_2, \qquad \mathbf{h}_{\mathcal{R}}^{(j)} \in \mathrm{NN}_K(\mathbf{h}, \mathcal{R}), \quad K = 10. \tag{2}$$

Let $\widetilde{\mathcal{R}}$ be a held-out set sampled from the same distribution as $\mathcal{R}$. We define the coverage ratio of $\mathcal{Q}$ by $\mathcal{R}$ as

$$\rho(\mathcal{Q} \mid \mathcal{R}) = \frac{\mathrm{median}_{\mathbf{h} \in \mathcal{Q}}\ r(\mathbf{h}; \mathcal{R})}{\mathrm{median}_{\mathbf{h} \in \widetilde{\mathcal{R}}}\ r(\mathbf{h}; \mathcal{R})}. \tag{3}$$

A ratio $\rho(\mathcal{Q} \mid \mathcal{R}) \approx 1$ indicates that samples from $\mathcal{Q}$ are as close to $\mathcal{R}$ as held-out samples from the reference distribution itself, whereas $\rho(\mathcal{Q} \mid \mathcal{R}) > 1$ indicates weaker coverage of $\mathcal{Q}$ by $\mathcal{R}$.



(a) Principal components (b) Drift axes

Figure 2: Coverage of downstream expert inputs by pretraining inputs. (a) Coverage ratio ρ (Eq. 3) computed in the subspaces spanned by the leading principal components of the pretraining inputs at layer 8 ($\rho \approx 1$). (b) The same ratio restricted to the drift subspace(ρ rises to 1.35–4.40).

Observation 1: Coverage along Dominant Principal Components. For a corpus $\mathcal{C}$, the set of token representations routed to expert i at layer ℓ is

$$\mathcal{H}_i^{(\ell)}(\mathcal{C}) = \left\{ \mathbf{h}_t^{(\ell)}(\mathbf{s}) \;\middle|\; g_i^{(\ell)}\big(\mathbf{h}_t^{(\ell)}(\mathbf{s})\big) > 0,\ \mathbf{s} \in \mathcal{C},\ 1 \le t \le |\mathbf{s}| \right\}, \tag{4}$$

where $\mathbf{h}_t^{(\ell)}(\mathbf{s})$ denotes the representation of the t-th token in sequence $\mathbf{s}$ at the input to layer ℓ. Applying this definition to the pretraining corpus $\mathcal{C}_{\text{pre}}$ and the downstream corpus $\mathcal{C}_{\text{down}}$ yields the two expert-input sets

$$\mathcal{H}_{i,\text{pre}}^{(\ell)} = \mathcal{H}_i^{(\ell)}(\mathcal{C}_{\text{pre}}), \qquad \mathcal{H}_{i,\text{down}}^{(\ell)} = \mathcal{H}_i^{(\ell)}(\mathcal{C}_{\text{down}}). \tag{5}$$

To compare these distributions, for each expert we project both sets onto subspaces spanned by the leading principal components of $\mathcal{H}_{i,\text{pre}}^{(\ell)}$ and compute $\rho(\mathcal{H}_{i,\text{down}}^{(\ell)} \mid \mathcal{H}_{i,\text{pre}}^{(\ell)})$. As illustrated in Figure 2(a), at layer $\ell = 8$, the coverage ratios remain between 0.99 and 1.01 in the subspaces spanned by the leading 256 and 1,024 principal components. The corresponding fractions of downstream inputs whose radii exceed the 95th percentile of the held-out pretraining distribution range from 2.4% to 8.2%, close to the 5% pretraining control. Additional results in Appendix A, covering seven downstream domains and principal-component subspaces of up to 1,024 dimensions, consistently yield $\rho \in [0.9, 1.1]$. Thus, along the dominant directions, downstream expert inputs are covered as well as held-out pretraining inputs, and the downstream gap is unlikely to stem solely from a broad loss of coverage, suggesting that the pretrained experts retain relevant capabilities that are not fully elicited downstream.

Low-Rank Drift in Expert Inputs. Despite the broad coverage along the dominant principal components, we find that the discrepancy between $\mathcal{H}_{i,\text{pre}}^{(\ell)}$ and $\mathcal{H}_{i,\text{down}}^{(\ell)}$ is concentrated along only a few directions. As illustrated in Figure 2(b), the coverage ratio within the identified drift subspace reaches 1.35–4.40 across seven downstream domains and multiple experts at layers 4, 8, 12, and 15. The largest shift occurs on Math SFT, where the ratio consistently ranges from 3.44 to 4.40. We defer the construction and additional analysis of the drift directions to Appendix B. This low-rank structure motivates adapting expert inputs with a low-rank adapter while preserving the router–expert coupling established during pretraining.

3.2 Low-Rank Expert-Input Adaptation

The analysis above shows broad coverage along the dominant principal components and a shift concentrated in a few directions. We therefore keep the pretrained routing fixed and adapt only the inputs to the selected experts. Figure 1 summarises the design.

Routing. Let $\mathbf{h}_t^{(\ell)} \in \mathbb{R}^d$ denote the representation of token t at the input of MoE layer ℓ. The frozen pretrained router computes the routing weights from this representation,

$$\mathbf{g}_t^{(\ell)} = \mathbf{g}^{(\ell)}\big(\mathbf{h}_t^{(\ell)}\big). \tag{6}$$

The i-th component $g_{t,i}^{(\ell)} = g_i^{(\ell)}(\mathbf{h}_t^{(\ell)})$ is the weight of expert i in Eq. 1. The router parameters remain frozen during adaptation.

Expert-input adapter. A single low-rank adapter is shared by all experts of layer ℓ,

$$\boldsymbol{\delta}^{(\ell)}(\mathbf{h}) = \mathbf{W}_{\text{up}}^{(\ell)}\, \phi\big(\mathbf{W}_{\text{down}}^{(\ell)}\mathbf{h}\big), \qquad \mathbf{W}_{\text{down}}^{(\ell)} \in \mathbb{R}^{r\times d}, \quad \mathbf{W}_{\text{up}}^{(\ell)} \in \mathbb{R}^{d\times r}, \quad r < d, \tag{7}$$

where $\phi(\cdot)$ is the SiLU activation. The input to expert i is

$$\widetilde{\mathbf{h}}_{t,i}^{(\ell)} = \mathbf{h}_t^{(\ell)} + \big(1+\beta_i^{(\ell)}\big)\, \boldsymbol{\delta}^{(\ell)}\big(\mathbf{h}_t^{(\ell)}\big), \tag{8}$$

where $\beta_i^{(\ell)}$ is an optional per-expert scalar initialised to zero. Without it, all selected experts take the same input $\mathbf{h}_t^{(\ell)} + \boldsymbol{\delta}^{(\ell)}(\mathbf{h}_t^{(\ell)})$. The experiments report both variants and mark the second one by $+\beta_i$. The output of the layer is

$$\text{MoE}_{\text{adapt}}^{(\ell)}\big(\mathbf{h}_t^{(\ell)}\big) = \sum_{i=1}^{N} g_{t,i}^{(\ell)}\, E_i^{(\ell)}\big(\widetilde{\mathbf{h}}_{t,i}^{(\ell)}\big). \tag{9}$$

The adapter is excluded from routing at layer ℓ. The frozen router selects the experts and computes their weights from $\mathbf{h}_t^{(\ell)}$, before the adapter is applied. The adapter modifies only the input to the selected experts.

Trainable parameters. The router and expert parameters Θ remain frozen. Only the adapter parameters are optimized,

$$\Phi = \big\{\mathbf{W}_{\text{down}}^{(\ell)},\ \mathbf{W}_{\text{up}}^{(\ell)},\ \boldsymbol{\beta}^{(\ell)}\big\}_{\ell\in\mathcal{L}_{\text{adapt}}}, \qquad \Phi^* = \arg\min_{\Phi}\ \mathcal{L}_{\text{task}}\big(\Theta, \Phi; \mathcal{C}_{\text{down}}\big), \tag{10}$$

where $\mathcal{L}_{\text{adapt}}$ is the set of adapted MoE layers. The vector $\boldsymbol{\beta}^{(\ell)} = (\beta_1^{(\ell)}, \ldots, \beta_N^{(\ell)})$ is trained only in the $+\beta_i$ variant. The shared adapter adds $2dr$ parameters per layer, independent of N. The optional scalars add N parameters per layer, which is 1,024 in total for 16 MoE layers with N=64. The correction lies in the column space of $\mathbf{W}_{\text{up}}^{(\ell)}$, of dimension at most r. This subspace is learned from the task loss and is not constrained to equal the diagnostic drift subspace.

4 Experiments

4.1 Experimental Setup

Tasks. We compare LoREAL with router tuning and LoRA under two adaptation settings, continued pretraining (CPT) and supervised fine-tuning (SFT). CPT evaluates adaptation on domain-mixed unlabeled text using the next-token prediction objective. For SFT, we consider a broad general-purpose instruction mixture, Tülu 3, together with two domain-specific settings. These are mathematical reasoning on OpenMathInstruct-2 (OMI2) and code generation on CODEMIX, our curated code instruction corpus. Within each comparison, methods use the same training corpus. Fixed-step and early-stopped protocols are distinguished below.

Datasets. For CPT, we use CPT1B, a one-billion-token corpus sampled from Dolmino-Mix-1124 (OLMo Team et al., 2025), the mid-training mixture of OLMo 2, with the sampling following the original domain proportions of the mixture. Its validation set is separated by a document-level hash before sampling, so that no document contributes to both splits. For mathematical SFT, we use the official two-million-example subset of OpenMathInstruct-2 rather than sampling examples ourselves. It contains 2,004,000 training and 4,000 validation examples, corresponding to approximately 14% of the complete dataset. For general instruction tuning, we use the OLMoE SFT mixture derived from Tülu 3 (Lambert et al., 2024; Muennighoff et al., 2025), containing 606,042 training and 2,000 validation conversations. For code SFT, we construct CODEMIX, a corpus drawn from six selected OpenCoder, CodeFeedback, and Magicoder sources. We use all examples within the selected sources and apply only deduplication and benchmark-contamination filtering, resulting in 554,464 training and 2,000 validation examples. OMI2 is formatted as a direct concatenation of the problem and solution, without an instruction template. Tülu 3 and CODEMIX use the same user–assistant conversation format. Table 1 summarizes the reported data splits.

Table 1: Adaptation corpora and their training/validation splits.

Setting	Corpus	Train	Validation	Format
CPT	CPT1B (sampled from Dolmino-Mix-1124)	1B tokens	3,072 chunks	raw text
SFT (math)	OpenMathInstruct-2 (OMI2)	2,004,000	4,000	problem ‖ solution
SFT (general)	Tülu 3 (OLMoE SFT mixture)	606,042	2,000	user–assistant
SFT (code)	CODEMIX (OpenCoder / CodeFeedback / Magicoder)	554,464	2,000	user–assistant

Baselines and parameter budgets. We compare against three baselines. The first is the *frozen base model*. The second is *router tuning*, which trains only the 16 routers ($16 \times 64 \times 2048 = 2.10$M parameters) and keeps all experts frozen. The third is *LoRA* (Hu et al., 2022) on the expert weight matrices with the routers frozen, where the rank is chosen so that the trainable parameter count matches ours. LoREAL adds a rank-r adapter to every MoE layer, i.e. $16\,(2dr)$ parameters. Budgets are matched as follows. Against router tuning (2.10M), we use $r = 32$ (2.10M) for both SFT and CPT. The CPT run at this budget adds the $1{,}024$ per-expert scalars. SFT runs use no per-expert scalars. Against LoRA, we use $r = 144$ (9.44M, matching rank-1 LoRA on the experts) for SFT, and for CPT additionally $r = 288$ (18.87M, rank-2 LoRA) and $r = 576$ (37.75M, rank-4 LoRA). At these CPT budgets, we also evaluate a variant that adds a per-expert scalar β_i ($1{,}024$ parameters in total). Within a budget methods use the same data order up to their respective stopping steps. Budgets count effective parameters. The LoRA implementation pads ranks to 8 for execution, so allocated tensors and actual memory costs are larger.

Training details. Adapters are inserted in all 16 MoE layers, $\mathbf{W}_{\mathrm{up}}^{(\ell)}$ is initialised to zero so training starts from the pretrained computation exactly. No auxiliary routing or load-balancing loss is used. Freezing router weights does not guarantee unchanged expert loads, since earlier adapters can alter later router inputs. Within each setting, optimiser, learning rate and schedule are shared across methods. CPT runs at the 2.10M budget train for $2{,}400$ steps with learning rate 5×10^{-4}. Larger CPT runs use validation-based early stopping and report the selected checkpoint, so their training steps can differ. SFT results are reported at the final checkpoint (step $1{,}721$ for the 9.44M runs).

Evaluation. For OMI2 we report accuracy on the full GSM8K (Cobbe et al., 2021) test set ($1{,}319$ problems) in the training format of the corpus, with boxed and flexible answer matching. For Tülu 3 we evaluate zero-shot with the Tülu chat template. We report GSM8K with flexible matching and IFEval (Zhou et al., 2023) prompt- and instruction-level accuracy under strict and loose matching. For CODEMIX we report execution-based pass@1 on HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021). For CPT we report held-out perplexity (PPL, denoted CPT PPL in the tables) on the CPT1B validation split and, to measure forgetting, the relative increase (%) in combined perplexity over four upstream pretraining domains (StarCoder, arXiv, AlgebraicStack and OpenWebMath) relative to the frozen base model (perplexity 4.3369 in the fixed-$2{,}400$-step upstream evaluation). Forgetting is $100(P_{\mathrm{adapt}}/P_{\mathrm{base}} - 1)$ for the same upstream evaluation within each comparison, not an absolute PPL difference. All numbers are single runs with a fixed seed.

4.2 Main Results

Continued pretraining. Table 2 reports continued pretraining on CPT1B. At the 2.10M budget, LoREAL lowers held-out perplexity from 7.843 (frozen base) to 6.467, against 6.997 for router tuning, while increasing upstream perplexity by only 0.60% compared with 3.03% for router tuning, a fivefold difference in forgetting. At the three larger budgets, LoREAL reaches lower held-out perplexity than expert-side LoRA with approximately the same number of trainable parameters at every scale (6.349 vs. 6.354 at 9.44M, 6.306 vs. 6.315 at 18.87M and 6.273 vs. 6.278 at 37.75M). Its forgetting is lower than that of LoRA at 9.44M (0.70% vs. 0.74%) and 18.87M (0.70% vs. 0.75%), and higher at 37.75M (0.94% vs. 0.83%). Both methods freeze the pretrained router parameters and increase upstream perplexity by roughly 1% or less, well below the 3.03% of router tuning, which trains only 2.10M parameters. These results are consistent with improved retention when router parameters remain frozen. They do not isolate routing as the main cause of forgetting, since parameterisation, optimisation and stopping steps also differ. Between LoREAL and LoRA, the forgetting difference is small and changes sign with scale. Adding the per-expert scalar β_i ($1{,}024$ additional parameters) further lowers perplexity at all three budgets (only marginally at 18.87M) at a small additional cost in forgetting. Figure 3 shows the full training curves behind these results. CPT is also the setting in which the coverage analysis of Section 3.1 applies most directly. The

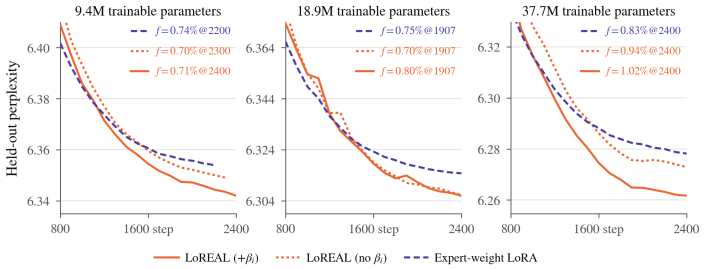


Figure 3: Held-out perplexity on CPT1B during continued pretraining at three matched budgets for LoREAL with and without the per-expert scalar β_i (orange solid and dotted) and expert-weight LoRA (blue dashed). Annotations give the forgetting f (relative upstream perplexity increase, %) and the step at which it is measured.

Table 2: Continued pretraining on CPT1B at matched trainable-parameter budgets. We report held-out perplexity and forgetting (lower is better). The $+\beta_i$ variant adds a per-expert scalar (1,024 parameters).

Trainable Params.	Method	CPT PPL	Forgetting (%)
–	Frozen base	7.8434	–
2.10M	Router tuning	6.9966	+3.03
	LoREAL (r=32, $+\beta_i$)	**6.4669**	**+0.60**
9.44M	Expert LoRA (r=1)	6.3540	+0.74
	LoREAL (r=144)	6.3490	**+0.70**
	LoREAL (r=144, $+\beta_i$)	**6.3422**	+0.71
18.87M	Expert LoRA (r=2)	6.3148	+0.75
	LoREAL (r=288)	6.3063	**+0.70**
	LoREAL (r=288, $+\beta_i$)	**6.3060**	+0.80
37.75M	Expert LoRA (r=4)	6.2783	**+0.83**
	LoREAL (r=576)	6.2729	+0.94
	LoREAL (r=576, $+\beta_i$)	**6.2616**	+1.02

Table 3: SFT results under matched parameter budgets (router tuning vs. LoREAL r=32 at ≈2.10M, and expert-side LoRA r=1 vs. LoREAL r=144 at ≈9.44M). LoREAL is used without β_i in all SFT runs. All scores are percentages. The best result within each budget is in bold.

			Budget ≈ 2.10M		Budget ≈ 9.44M	
SFT corpus	Metric	Frozen	Router tuning	LoREAL (r=32)	LoRA (r=1)	LoREAL (r=144)
OMI2 (math)	GSM8K, boxed	0.0	13.0	**33.1**	38.9	**43.5**
	GSM8K, flexible	2.0	12.6	**31.8**	36.5	**41.1**
Tülu 3 (general)	GSM8K, flexible	1.8	12.4	**24.9**	28.7	**30.6**
	IFEval prompt, strict	24.2	15.7	**25.7**	26.1	**29.0**
	IFEval prompt, loose	24.6	18.7	**28.3**	29.0	**31.1**
	IFEval inst., strict	35.9	26.5	**39.0**	39.2	**40.9**
	IFEval inst., loose	36.1	29.7	**41.4**	42.6	**43.1**
CODEMIX (code)	HumanEval pass@1	0.0	15.9	**17.7**	17.7	**21.3**
	MBPP pass@1	0.0	20.0	**24.0**	**21.2**	20.0

objective and domain mixture are close to pretraining, which makes expert-input adaptation a natural hypothesis to test. The performance gains support its practical utility, but do not establish that the learned correction reverses the measured drift.

Supervised fine-tuning. Table 3 reports SFT results on the three corpora under two matched parameter budgets. Three observations stand out. First, at the ≈2.10M budget, LoREAL improves over router tuning on all nine metrics, often by a wide margin (e.g., 33.1 vs. 13.0 boxed GSM8K accuracy after OMI2 SFT, and 24.9 vs. 12.4 GSM8K accuracy after Tülu 3 SFT). Moreover, router tuning falls below the frozen base model on IFEval (15.7 vs. 24.2 prompt-level strict accuracy) despite being trained on Tülu 3, whereas LoREAL improves every metric over the base model. Second, at the ≈9.44M budget, LoREAL with r=144 outperforms expert-side LoRA on eight of nine

Table 4: Router capacity at fixed 2,400 steps. Multipliers compare trainable parameters with router tuning, not total frozen-plus-trainable router size. Forgetting is the relative increase (%) in combined upstream PPL over the frozen base model.

Method	Params. (M)	Multiple	CPT PPL	Forgetting (%)
Router tuning	2.097	1.00×	6.9966	+3.03
SiLU routing residual, $r = 64$	2.163	1.031×	6.8616	+0.96
SiLU routing residual, $r = 128$	4.325	2.063×	6.7880	+0.96
SiLU routing residual, $r = 512$	17.302	8.25×	6.6955	+1.07
SiLU routing residual, $r = 1024$	34.603	16.5×	6.6501	+1.12
SiLU routing residual, $r = 2048$	69.206	33.0×	6.6173	+1.20
LoREAL, $r = 32$, $+\beta_i$	2.098	1.0005×	**6.4669**	**+0.60**

metrics, with the largest gains on mathematics (+4.6 boxed GSM8K) and HumanEval (+3.6). The only exception is MBPP (20.0 vs. 21.2). Third, improvements occur across mathematics, instruction following and HumanEval, but do not establish uniform gains across tasks or parameter scales. For example, MBPP decreases from 24.0 at $r = 32$ to 20.0 at $r = 144$. These single-run results motivate further seed and hyperparameter sweeps rather than a claim that increasing adapter rank always helps.

4.3 Ablation Study

Is the router simply too small? We freeze the backbone and original gate and add a residual to the routing logits,

$$\mathbf{z}^{(\ell)}(\mathbf{h}) = \mathbf{W}_g^{(\ell)}\mathbf{h} + \mathbf{U}^{(\ell)}\phi(\mathbf{D}^{(\ell)}\mathbf{h}), \quad \mathbf{D}^{(\ell)} \in \mathbb{R}^{r\times d}, \quad \mathbf{U}^{(\ell)} \in \mathbb{R}^{N\times r}. \tag{11}$$

The original sparse top-k selection and weighting are then applied to these logits. Only $\mathbf{D}, \mathbf{U}$ are trained, with $\mathbf{U}$ initialised to zero. The parameter count is $16r(d+N)$, excluding the frozen gate. All capacity and nonlinearity runs use CPT1B, the same 3,072 held-out chunks, learning rate 5×10^{-4}, and 2,400 steps. We report final evaluations, which can differ slightly from periodic evaluations of the same checkpoint.

Table 4 scales the SiLU routing residual from 2.16M to 69.21M parameters (32×). Perplexity improves by 0.2443, from 6.8616 to 6.6173, but remains 0.1504 above the 2.10M expert-input adapter. The largest residual uses approximately 33× as many trainable parameters as this adapter. Capacity helps routing, but does not close the gap over the tested range. Forgetting rises from 0.96% to 1.20% within the residual family, yet remains below router tuning (3.03%) and above LoREAL (0.60%). Thus parameter count alone does not explain the adaptation–retention trade-off.

How much does nonlinearity contribute? We replace SiLU by the identity while keeping the architecture, parameter count, learning rate and 2,400-step budget fixed within each pair (Table 5). The PPL penalty is 0.2094 and 0.3522 for the two routing residuals, versus 0.0649 for the expert-input adapter. Without SiLU, the $r = 64$ routing residual is 0.0744 worse than router tuning, and $r = 1024$ remains 0.0057 worse. Scaling the linear residual from $r = 64$ to 1024 improves PPL by 0.0687, but does not outperform router tuning. In contrast, the linear expert-input adapter reaches 6.5318, better by 0.1183 than the nonlinear 34.60M routing residual, and by 0.0855 than the 69.21M residual. These results indicate that expert-input adaptation does not depend primarily on the added SiLU.

For the bias-free linear routing residual, the logits are $(\mathbf{W}_g + \mathbf{UD})\mathbf{h}$. Since $N = 64$ and both tested ranks satisfy $r \geq N$, $\mathbf{UD}$ can represent any $N \times d$ matrix. Its logit-function class therefore equals that of directly training the linear gate. This equivalence does not imply equal optimisation trajectories under factorisation, initialisation and weight decay. A linear expert-input adapter, by contrast, precedes nonlinear frozen experts and can change their computation even with fixed local routing.

Which component receives the adapted representation? A separate experiment uses the same rank-32 adapter without per-expert scales (2,097,152 parameters), learning rate 10^{-3} and one epoch (1,907 steps). Both arms use the same CPT1B data and validation set. We send $\widetilde{\mathbf{h}}$ either to both

Table 5: SiLU versus identity at fixed $2{,}400$ steps. Δ is PPL(identity)$-$PPL(SiLU). The expert-input pair includes the same per-expert scales in both arms.

Module	Params. (M)	SiLU	Identity	Δ
Routing residual, $r = 64$	2.163	6.8616	7.0710	0.2094
Routing residual, $r = 1024$	34.603	6.6501	7.0023	0.3522
Expert-input adapter, $r = 32$, $+\beta_i$	2.098	6.4669	6.5318	0.0649

Table 6: Insertion-position comparison with learning rate 10^{-3} and $1{,}907$ steps. These values should not be compared directly with the $2{,}400$-step protocol. Router and expert columns specify the local layer input.

Method	Router input	Expert input	CPT PPL
Adapter to both	$\widetilde{\mathbf{h}}$	$\widetilde{\mathbf{h}}$	6.4125
Expert-only adapter	$\mathbf{h}$	$\widetilde{\mathbf{h}}$	6.4208
Router tuning (no adapter)	$\mathbf{h}$	$\mathbf{h}$	6.9810

the router and experts, or only to the experts (Table 6). Allowing the router to read $\widetilde{\mathbf{h}}$ lowers PPL by only 0.0083, compared with the 0.5602 improvement of expert-only over router tuning under this protocol. This supports expert-input adaptation as the main contributor in this comparison, but does not constitute an additive causal decomposition or establish significance from a single run. We use expert-only to exclude the local adapter's direct influence on routing. This does not guarantee identical downstream routing to the frozen base model.

An exploratory output-side adapter reaches PPL 6.5905 at step $1{,}025$, versus approximately 6.45 for expert-only at the same step. This intermediate-checkpoint comparison is suggestive, but a matched completed run is needed before drawing a firm conclusion about input versus output placement.

Should the router be trained jointly with the adapter? Under the same 10^{-3} learning-rate protocol, jointly training routers and adapters doubles the trainable parameters from $2{,}097{,}152$ to $4{,}194{,}304$. In the both-input setting, joint training improves PPL at step 25 but is worse at all six subsequent evaluations through step 175, with a final observed gap of 0.0723. Expert-only shows the same initial crossover at steps 25 and 50. These incomplete trajectories support freezing routers under this shared learning rate, but do not establish final convergence or exclude gains from separately tuning the router learning rate. Appendix C reports all available paired checkpoints.

5 Conclusion

We asked whether the downstream gap of pretrained MoE language models is a routing problem. Measuring the inputs that experts actually receive, we found near-unit local coverage ratios in the tested principal-component subspaces, while exploratory drift diagnostics remain inconclusive about intrinsic dimensionality because of the reference-control limitations discussed in the appendix. Together with the ablations, these observations support expert-input adaptation as an effective alternative to routing changes in the tested setting. LoREAL acts on exactly this quantity. It freezes the pretrained router and experts and keeps routing on the input before the local adapter. It inserts a single low-rank residual adapter, shared across the experts of a layer, in front of expert computation. With a default adapter parameter count independent of the number of experts, it outperforms router tuning and improves on LoRA in most matched-budget comparisons across continued pretraining and supervised fine-tuning, and reduces measured forgetting relative to router tuning, while still incurring nonzero forgetting. Our analysis is limited to one model family and to adapters placed at the MoE layer input. Extending the coverage analysis to other architectures and combining input-side with expert-side adaptation are natural next steps.

REPRODUCIBILITY STATEMENT

All experiments use OLMoE-1B-7B. The inspected CPT logs specify the 0924 step-1,200,000 checkpoint (Muennighoff et al., 2025). Section 4.1 specifies the tasks, training corpora (Table 1), baselines, matched parameter budgets and evaluation protocols. Section 3.2 gives the exact form of the adapter and the set of trainable parameters (Eq. 10). Detailed measurements for Section 3.1 appear in Appendix A (per-expert values in Table 7) and Appendix B (identification of the drift axes, including the held-out split and control experiments). Training code, adapter checkpoints and evaluation scripts will be released upon publication.

AI USE STATEMENT

We used AI tools for five purposes. These are polishing the writing, drafting parts of some sections, finding related work, implementing and debugging experiments, and deriving and checking mathematical claims. AI tools were not used to generate datasets. The authors reproduced all experimental results and manually verified all mathematical derivations. The authors reviewed and revised all AI-assisted text and take full responsibility for the content of the paper.

REFERENCES

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021. URL https://arxiv.org/abs/2108.07732.

Jun Bai, Minghao Tong, Yang Liu, Zixia Jia, and Zilong Zheng. Understanding and leveraging the expert specialization of context faithfulness in mixture-of-experts LLMs. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. URL https://arxiv.org/abs/2107.03374.

Zixiang Chen, Yihe Deng, Yue Wu, Quanquan Gu, and Yuanzhi Li. Towards understanding mixture of experts in deep learning. *arXiv preprint arXiv:2208.02813*, 2022. URL https://arxiv.org/abs/2208.02813.

Zewen Chi, Li Dong, Shaohan Huang, et al. On the representation collapse of sparse mixture of experts. In *Advances in Neural Information Processing Systems*, 2022.

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. URL https://arxiv.org/abs/2110.14168.

Damai Dai, Chengqi Deng, Chenggang Zhao, et al. DeepSeekMoE: Towards ultimate expert specialization in mixture-of-experts language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024.

DeepSeek-AI. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Guanzhi Deng, Kuan Wu, Haibo Wang, Shing Yin Wong, Sichun Luo, and Linqi Song. Routing-aligned fine-tuning for multilingual downstream tasks in mixture-of-experts models. *arXiv preprint arXiv:2605.28306*, 2026.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Shwai He, Tao Ge, Guoheng Sun, Bowei Tian, Xiaoyang Wang, and Dong Yu. Router-tuning: A simple and effective approach for enabling dynamic-depth in transformers. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning*, 2019.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tülu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. GShard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*, 2021.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, Kushal Arora, et al. DataComp-LM: In search of the next generation of training sets for language models. In *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*, 2024. URL https://arxiv.org/abs/2406.11794.

Zhongyang Li, Ziyue Li, and Tianyi Zhou. R2-T2: Re-routing in test-time for multimodal mixture-of-experts. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 35292–35316. PMLR, 2025.

Zhongyang Li, Ziyue Li, and Tianyi Zhou. Routing manifold alignment improves generalization of mixture-of-experts LLMs. In *International Conference on Learning Representations*, 2026.

Yilun Liu, Yunpu Ma, Yuetian Lu, Shuo Chen, Zifeng Ding, and Volker Tresp. Parameter-efficient routed fine-tuning: Mixture-of-experts demands mixture of adaptation modules. *arXiv preprint arXiv:2508.02587*, 2025.

Ang Lv, Jin Ma, Yiyuan Ma, and Siyuan Qiao. Coupling experts and routers in mixture-of-experts via an auxiliary loss. In *International Conference on Learning Representations*, 2026.

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. In *International Conference on Learning Representations*, 2025.

OLMo Team, 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, Michal Guerquin, Hamish Ivison, Pang Wei Koh, Jiacheng Liu, Saumya Malik, William Merrill, Lester James V. Miranda, Jacob Morrison, Tyler Murray, Crystal Nam, 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. *arXiv preprint arXiv:2501.00656*, 2025.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017.

Yaqing Wang, Sahaj Agarwal, Subhabrata Mukherjee, Xiaodong Liu, Jing Gao, Ahmed Hassan Awadallah, and Jianfeng Gao. AdaMix: Mixture-of-adaptations for parameter-efficient model tuning. *arXiv preprint arXiv:2205.12410*, 2022. URL https://arxiv.org/abs/2205.12410.

Zihan Wang, Deli Chen, Damai Dai, Runxin Xu, Zhuoshu Li, and Yu Wu. Let the expert stick to his last: Expert-specialized fine-tuning for sparse architectural large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024.

Longrong Yang, Dong Shen, Chaoxiang Cai, Fan Yang, Tingting Gao, Di Zhang, and Xi Li. Solving token gradient conflict in mixture-of-experts for large vision-language model. In *International Conference on Learning Representations*, 2025.

Youngsik Yoon, Siwei Wang, Wei Chen, and Jungseul Ok. When are experts misrouted? counterfactual routing analysis in mixture-of-experts language models. *arXiv preprint arXiv:2605.07260*, 2026.

Zhibo Zhang, Zhen Ouyang, Ling Shi, and Kailong Wang. RARE: Decoupling representation steering from expert routing in mixture-of-experts language models. *arXiv preprint arXiv:2608.21236*, 2026.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023. URL https://arxiv.org/abs/2311.07911.

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. *arXiv preprint arXiv:2202.08906*, 2022. URL https://arxiv.org/abs/2202.08906.

A Estimating the Expert-Input Statistics

Coverage per expert. Table 7 reports the coverage ratio of Eq. 3 at the level of individual experts, taking $\mathcal{R} = \mathcal{H}^{(\ell)}_{i,\text{pre}}$ and $\mathcal{Q} = \mathcal{H}^{(\ell)}_{i,\text{down}}$, after projection onto each expert's leading 1,024 reference principal components, for four experts at each of four layers and seven downstream domains. Across all 112 expert–domain pairs the ratio stays within $[0.93,\ 1.03]$, i.e., within a few percent of the held-out pretraining reference and never more than 3% above it.

Table 7: Per-expert coverage ratio $\rho(\mathcal{H}^{(\ell)}_{i,\text{down}} \mid \mathcal{H}^{(\ell)}_{i,\text{pre}})$ (Eq. 3, K=10) in the leading 1,024 reference principal components, for four experts at layers 4, 8, 12 and 15 across seven downstream domains. A value of $\rho \approx 1$ means downstream inputs are covered as well as held-out pretraining inputs.

Layer	Expert	Math	Math SFT	FLAN	arXiv	Code	Wikipedia	peS2o
4	23	0.99	0.97	1.02	0.99	0.96	1.00	0.99
	59	1.00	0.98	1.03	0.99	0.93	1.00	0.99
	60	0.99	0.99	1.01	1.00	1.01	1.01	0.99
	62	0.99	0.99	1.01	1.00	1.00	1.01	1.01
8	10	1.00	1.00	1.01	0.98	0.98	0.99	0.98
	17	0.98	1.00	1.00	0.97	0.98	0.98	0.97
	18	1.00	1.01	1.03	0.98	0.98	1.00	0.99
	35	1.00	1.01	1.01	0.98	0.97	0.99	0.98
12	14	0.99	1.00	1.00	0.97	0.99	0.99	0.97
	50	0.99	0.99	0.99	0.97	0.98	0.99	0.97
	52	0.99	0.98	0.99	0.96	0.99	0.99	0.97
	61	0.96	0.95	0.99	0.93	0.98	0.99	0.95
15	7	0.98	0.98	0.98	0.96	0.98	1.00	0.98
	10	0.99	1.01	0.99	0.98	0.99	1.02	0.99
	13	0.99	1.01	0.99	0.98	0.99	1.02	0.99
	21	0.98	0.98	0.98	0.96	0.99	1.00	0.98

B Finding the drift axes

All quantities are computed separately for every layer ℓ, expert i and downstream corpus, in the full d-dimensional expert-input space ($d = 2048$ for OLMoE). Following Eq. 5, the reference set is $\mathcal{R} = \mathcal{H}^{(\ell)}_{i,\text{pre}}$, the representations routed to expert i (those with $g^{(\ell)}_i(\mathbf{h}) > 0$) on a DCLM (Li et al., 2024) reference sample (2,048 chunks of 1,024 tokens), used as a proxy for pretraining inputs, and the query set is $\mathcal{Q} = \mathcal{H}^{(\ell)}_{i,\text{down}}$. We write $\boldsymbol{\mu}$ and Σ for the mean and covariance of $\mathcal{R}$.

Step 1: put the pretraining inputs on a common scale. The implementation factorises $C_\epsilon = LL^\top = \Sigma + \epsilon I$ and maps each input to $\mathbf{z} = L^{-1}(\mathbf{h} - \boldsymbol{\mu})$, where $\epsilon = 0.05\ \text{tr}(\Sigma)/d$. The transformed reference covariance is $C_0 = L^{-1}\Sigma L^{-\top}$, which is generally not the identity. This regularised transform rescales the inputs but does not exactly whiten them.

Step 2: compute deviations from the identity reference. Split $\mathcal{Q}$ into the first and second halves of the stored token sequence, $\mathcal{Q}_{\text{fit}}$ and $\mathcal{Q}_{\text{test}}$. On the fitting half compute

$$M \;=\; \frac{1}{|\mathcal{Q}_{\text{fit}}|} \sum_{\mathbf{h} \in \mathcal{Q}_{\text{fit}}} \mathbf{z}_\mathbf{h} \mathbf{z}_\mathbf{h}^\top \;-\; I. \tag{12}$$

For a unit direction $\mathbf{u}$, $\mathbf{u}^\top M\mathbf{u}$ is the downstream mean squared projection minus one. Because regularisation gives reference covariance $C_0 \neq I$, this is not exactly downstream-minus-reference excess spread. That quantity is $\mathbf{u}^\top(M + I - C_0)\mathbf{u}$. The exact downstream-minus-reference second-moment matrix is $D = M + I - C_0$. The reported axes were computed from M, not D. Substituting D requires recomputing the axes and all dependent measurements. A transformed downstream mean shift $\boldsymbol{\delta}$ contributes $\boldsymbol{\delta}\boldsymbol{\delta}^\top$ to its second moment.

Step 3: order directions by deviation magnitude. Diagonalise $M = \sum_j \lambda_j \mathbf{v}_j \mathbf{v}_j^\top$ and sort the eigenpairs by $|\lambda_j|$. Ordering by absolute eigenvalue selects the largest-magnitude deviations from the identity reference, including negative deviations. It does not necessarily order increases in spread. We call $\mathbf{v}_1, \mathbf{v}_2, \ldots$ the drift axes of $(\ell, i, \mathcal{Q})$ and $S_k = \mathrm{span}(\mathbf{v}_1, \ldots, \mathbf{v}_k)$ the k-dimensional drift subspace. In practice $\mathbf{v}_1$ is closely aligned with the downstream mean-shift direction (cosine 0.96–0.99 for the math, math-SFT and FLAN sets). The following axes are directions of increased spread with little mean shift.

Step 4: evaluate on the held-out half only. Downstream axis measurements use $\mathcal{Q}_{\mathrm{test}}$. The implementation uses contiguous token halves, so document-level independence is not guaranteed. In the two-sample control, part of the reference used for covariance fitting is reused for classifier evaluation. This control is not fully independent. Per axis $\mathbf{v}_j$ we report (i) the spread ratio, the mean squared projection of $\mathcal{Q}_{\mathrm{test}}$ onto $\mathbf{v}_j$ divided by that of the pretraining inputs, and (ii) the share of $\mathcal{Q}_{\mathrm{test}}$ whose projection falls outside the interval holding 98% of the pretraining inputs, minus the 2% that pretraining itself has. Per subspace S_k we report the kNN radius ratio of Eq. 3 inside S_k and inside its orthogonal complement, and the accuracy of a 10-nearest-neighbour two-sample classifier (downstream vs. pretraining) after the S_k component has been removed.

Controls. The following controls accompany the exploratory drift analysis. (a) A held-out sample of pretraining text is processed exactly like a downstream set. The reported first-axis spread ratios are 4.5–7.5 (approximately 4.5 by axis 32 at layer 8). These values require a check of the split independence, denominator and reference sampling before they can be interpreted as a noise floor. Axis selection alone does not explain such large ratios on independent held-out samples. (b) For every k, a random k-dimensional subspace is removed instead of S_k. If the two-sample classifier stays as accurate as with nothing removed, the difference is not spread over arbitrary directions.

Exploratory axis counts. At layer 8, the reported math-SFT spread ratios are $129\times$, $53\times$, $40\times$ and $34\times$ on axes 1–4, $22\times$ on axis 8, $14\times$ on axis 16 and $9\times$ on axis 32. The reference-control ratios of 4.5–$7.5\times$ are not a validated noise floor. The previously used threshold of four times this control gives 3–9 axes, but this count cannot presently be interpreted as an estimate of drift rank. A separate diagnostic reports classifier accuracy ≤ 0.52 after removing 8–16 axes for math text, math SFT and FLAN, and more axes for arXiv and code. Random subspaces of the same size do not yield the same reduction.

Interpretation and limitations. The classifier result suggests that selected directions contain information useful to this particular discriminator. Near-chance accuracy does not establish distributional equality, and 8–16 directions (less than 1% of 2048 dimensions) should not be interpreted as the intrinsic dimension of the shift. The shared reference samples, contiguous token splits and unexpectedly large reference-control ratios limit this interpretation. A stronger claim requires document-disjoint fitting and evaluation sets, a verified reference denominator, and recomputation using the reference-corrected second moment D. The adaptation experiments remain empirical tests of the proposed architecture and do not demonstrate inversion of the measured shift.

C JOINT ROUTER AND ADAPTER TRAINING: EARLY CHECKPOINTS

We compare frozen-router adapters with joint router–adapter training in two settings. In *expert-only*, the router takes the current layer input $\mathbf{h}$ and the experts take $\widetilde{\mathbf{h}}$. In *both*, the router and the experts take $\widetilde{\mathbf{h}}$. Enabling `--adapter_train_gate` adds $2{,}097{,}152$ trainable router parameters across 16 layers to the $2{,}097{,}152$ adapter parameters, for a total of $4{,}194{,}304$. Expert weights remain frozen. Each pair uses rank 32, no per-expert scales, learning rate 10^{-3}, a planned one-epoch schedule ($1{,}907$ steps), the `ds_single_fp32` configuration and `dolmino_cpt_1b`, with validation every 25 steps. The remaining recipe is unchanged within each pair. This protocol differs from the $2{,}400$-step main comparison. The results below should be interpreted only against their paired controls.

Table 8: All available paired validation checkpoints for joint router–adapter training. Frozen-router adapters have 2.10M trainable parameters, and joint training has 4.19M. Δ is joint minus frozen-router PPL, so positive values favour freezing the router. These are partial trajectories, not final results.

Setting	Step	Frozen router	Joint training	Δ PPL
Expert-only	25	7.2782	7.1974	-0.0808
	50	6.9700	6.9877	$+0.0177$
Both	25	7.2910	7.1912	-0.0998
	50	6.9753	6.9923	$+0.0170$
	75	6.8633	6.9083	$+0.0450$
	100	6.8025	6.8593	$+0.0568$
	125	6.7558	6.8190	$+0.0632$
	150	6.7191	6.7867	$+0.0676$
	175	6.6919	6.7642	$+0.0723$

Joint training has lower PPL at step 25 in both settings, but higher PPL at step 50. In the both-input setting it is worse at six of seven observed checkpoints, and its disadvantage grows monotonically from step 50 to 175, even though both arms continue to improve in absolute PPL. The expert-only observations establish only the initial crossover, not a longer-term trend.

Scope and limitations. At the reporting snapshot, the expert-only joint run had reached step 56 (two evaluated checkpoints), while the both-input joint run had been interrupted at step 175. Neither trajectory reaches the planned $1{,}907$ steps. Router and adapter learning rates were both set to 10^{-3} without a separate router learning-rate search. These single-run observations show no sustained early benefit from the additional router parameters under this configuration. They do not establish final convergence, statistical significance, or a causal mechanism such as gradient interference. A smaller router learning rate could change the outcome. No forgetting measurements are available for these joint-training runs.