LEGO: Expert Expansion for Continual Learning of Mixture-of-Experts Language Models

Jiawei Wu [1] **Junlin Chen** [1] **Daize Dong** [1] **Haolong Jia** [1] **Souvik Kundu** [2] **Muhao Chen** [3] **Hongyi Wang** [1]

Abstract

Mixture-of-experts (MoE)-based large language models (LLMs) scale model capacity without proportional growth in compute, yet adapting them to new domains after pretraining remains challenging. Retraining from scratch is prohibitively expensive, while naive continual learning can cause catastrophic forgetting; i.e., new-domain gradients overwrite previously trained router models in the MoE. To this end, we propose LEGO, a novel expert expansion strategy that adds new experts with target-side routing pressure, steering new-domain tokens toward the expanded capacity, and anchor-side stabilization to prevent weight drift in existing experts relative to the previous checkpoint. The key insight of LEGO is that new-domain learning should be localized in dedicated capacity, leaving existing expert routing undisturbed. Extensive experimental results demonstrate that LEGO consistently improves the trade-off between acquiring new knowledge and preserving prior capabilities.

1. Introduction

Training large language models (LLMs) from scratch requires immense computational resources (Hoffmann et al., 2022), which makes full pretraining feasible for only a small number of well-resourced organizations (OpenAI, 2025; NVIDIA, 2025; Muennighoff et al., 2024b; DeepSeek-AI, 2024; Team et al., 2025). Consequently, progress in LLM development increasingly depends not on repeated end-to-end pretraining but on building on pretrained checkpoints through additional training (Parmar et al., 2024; Ibrahim et al., 2024). This shift is more than a practical compromise, i.e., it reflects a structural reality of modern LLM research.

[1]Rutgers University [2]Intel [3]UC Davis. Correspondence to: Jiawei Wu <jw2070@rutgers.edu>, Hongyi Wang <hw689@cs.rutgers.edu>.

Proceedings of the 43rd International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026. Copyright 2026 by the author(s).

Recent works (Zeng et al., 2026; Yang et al., 2025; Tu et al., 2025) have increasingly treated continual learning as a distinct stage between broad pretraining and downstream post-training, often referred to as *mid-training*. Rather than merely appending additional tokens to pretraining, continual learning uses higher-quality or domain-focused data to reshape a pretrained model into a representation that is better suited for later specialization (Wang et al., 2024a). This stage is particularly important in reasoning-intensive domains such as mathematics and code, where the distribution mismatch between broad web-scale pretraining and downstream objectives can be substantial (Liu et al., 2026; Runwal et al., 2026; OLMo et al., 2025). Under this view, continual learning serves as a bridge between general-purpose pretraining and task-oriented post-training.

For Mixture-of-Experts (MoE) models, however, this stage introduces a distinct challenge. MoE architectures route each token to only a small subset of experts, allowing the total parameter count to grow without a proportional increase in computation (Shazeer et al., 2017; Lepikhin et al., 2020). Importantly, pretraining does not leave these experts interchangeable. Prior work shows that experts often develop non-uniform specialization across tasks, semantic patterns, and data distributions (Wang et al., 2024b; Lo et al., 2025; Bai et al., 2025). As a result, a pretrained MoE model may encode part of its capabilities within a structured expert space rather than solely within a shared dense backbone.

This structural property makes continual learning in MoE models fundamentally different from that in dense models. In a dense model, continued training primarily modifies a shared parameter space. In an MoE model, by contrast, standard adaptation can overwrite, blur, or repurpose the behaviors of experts that emerged during pretraining. Prior work finds that expert usage is often highly concentrated and task-dependent: examples from the same task tend to rely on similar experts, whereas different tasks may depend on different subsets (Wang et al., 2024b). At the same time, recent analyses suggest that expert modularization in current MoE language models remains imperfect (Lo et al., 2025), and that excessive expert overlap or overly uniform routing can weaken specialization, particularly when domain-specific inputs are forced to spread across experts (Guo et al., 2026;

Qiu et al., 2025). Multi-domain adaptation studies further show that MoE models remain vulnerable to cross-domain interference and catastrophic forgetting during continued training (Li et al., 2025b).

Taken together, these findings suggest that continual learning for MoE models should not be treated as a straightforward extension of continued pretraining in dense models. The challenge is not only to absorb new domain knowledge but also to do so without unnecessarily disrupting the expert organization acquired during pretraining.

To address this challenge, we introduce LEGO, a continual learning framework for pretrained MoE models. At a high level, LEGO expands the expert pool to provide new sparse capacity for incoming domains, encourages target-domain examples to route toward the newly added experts, and uses preservation objectives on anchor data to retain previously learned functionality. In this way, LEGO treats continual learning as the stage at which new knowledge is integrated into a pretrained MoE model while disruption to the existing expert structure is minimized.

Our contributions are threefold. First, we formulate continual learning for pretrained MoE language models as a sequential domain adaptation problem in which forgetting arises from interference over sparse expert capacity. Second, we propose LEGO, which combines expert expansion with asymmetric adaptation and preservation objectives to better absorb knowledge from domains with large distribution shifts while retaining prior capabilities. Third, we systematically study parameter update policies for the shared trunk, existing experts, newly added experts, and gating parameters, thereby providing an empirical view of the adaptation–retention trade-off in sparse language models.

2. Related Work

Adaptive pretraining and mid-training. Adaptive pretraining has long been known to improve downstream performance when the training distribution is moved closer to the target task or domain (Gururangan et al., 2020). More recent work reframes this stage as *mid-training*, a deliberate bridge between the original pretraining mixture and the downstream post-training distribution (Liu et al., 2025). Surveys now treat mid-training as a broad design space spanning data curation, synthetic data generation, optimization, long-context extension, and architectural intervention (Mo et al., 2025; Tu et al., 2025). Our work sits inside this line, but focuses specifically on *sequential* mid-training and on the structural role of MoE capacity under repeated domain shift.

Continual learning and retention. The preservation side of our method is related to classical continual learning (Zhu et al., 2024). Learning without Forgetting uses an older model as a behavioral target during new-task adaptation (Li & Hoiem, 2016). Elastic Weight Consolidation instead constrains parameters that appear important for earlier tasks (Kirkpatrick et al., 2017). Proximal formulations such as FedProx also limit drift from a reference solution (Li et al., 2018). Our setting differs in two respects. First, we are not solving a task-incremental classification problem, but large-corpus language modeling under sequential domain shift. Second, the model is sparse, so the question is not only *how much* to preserve, but also *where* new knowledge should be stored.

Mixture-of-experts and sparse specialization. Sparse MoE layers provide high total capacity without proportional growth in per-token computation (Shazeer et al., 2017; Fedus et al., 2022). Open MoE releases such as OLMoE make it possible to inspect routing structure and to study expert specialization in realistic language models (Muennighoff et al., 2024a). Recent work has also studied dynamic expert allocation and layer conversion as a route to stronger specialization under adaptation (Li et al., 2025a). Our contribution differs in regime and objective. We begin from an already pretrained open MoE checkpoint and ask how to conduct continual learning without overwriting earlier sparse behavior.

MoE adaptation under domain shift. Closer to our setting, recent studies examine how MoE models adapt across multiple domains while avoiding catastrophic forgetting. These works highlight the roles of expert expansion, modular expert composition, and specialized routing/adaptation strategies, but do not study continual mid-training of a pretrained MoE model as the bridge stage where new domain knowledge must be incorporated without degrading pretraining-acquired expert structure.

Positioning of this paper. Compared with standard continued pretraining, our emphasis is on *capacity isolation*. Compared with continual-learning regularizers, we use sparse structure to decide where new knowledge should go. Compared with prior open-model release efforts, our contribution is not a new foundation model release per se, but a reproducible benchmark and training framework for studying sequential MoE adaptation.

3. Methods

Continual learning of mixture-of-experts (MoE) models raises a central question: *how can new knowledge be incorporated without overwriting prior capabilities?* In many practical settings, one may not know in advance which downstream capabilities will need to be strengthened after pretraining.

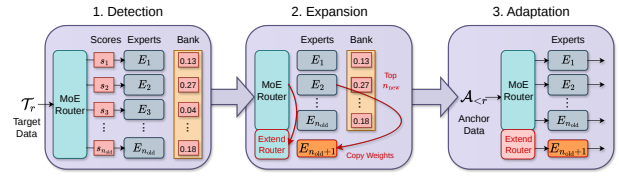

Figure 1. Overview of the LEGO pipeline. We first identify the most frequently activated experts, then duplicate them to expand model capacity, and finally adapt the expanded model through training.

A natural strategy is therefore to begin with a strong base model and introduce additional knowledge capacity only when new domains or capabilities need to be acquired. Compared with forcing all subsequent learning into a fixed expert set, this can reduce interference between new and previously learned knowledge.

Motivated by this view, our method, LEGO, combines three components: (1) expert expansion, which introduces additional knowledge capacity; (2) routing pressure, which encourages tokens from new domains to utilize this capacity; and (3) selective stabilization on anchor data, which preserves previously learned knowledge and capabilities. Together, these components reduce interference between new and existing knowledge, improving the trade-off between adaptation and retention.

3.1. Expert expansion in continual learning

We consider continual learning of a pretrained mixture-of-experts (MoE) language model over a sequence of rounds $r = 1, \ldots, R$. At round r, training uses two corpora: a target corpus $\mathcal{T}_r$ and an anchor corpus $\mathcal{A}_{<r}$. $\mathcal{T}_r$ contains data from the new domain to be learned in the current round. $\mathcal{A}_{<r}$ contains data used to preserve previously learned capabilities and mitigate forgetting.

These two corpora play different roles during training. $\mathcal{T}_r$ supports adaptation to new knowledge, while $\mathcal{A}_{<r}$ supports retention of prior knowledge. For later rounds, $\mathcal{A}_{<r}$ can include data retained from earlier rounds. For $r = 1$, $\mathcal{A}_{<r}$ is built from general-domain data that reflects the pretraining distribution.

A central challenge in this setting is interference: learning on $\mathcal{T}_r$ may overwrite knowledge useful for the distributions represented by $\mathcal{A}_{<r}$, especially when all updates must be absorbed by a fixed set of experts. To mitigate this, at the beginning of round r we expand each MoE layer from n_{old} experts to

$$n_{\text{total}} = n_{\text{old}} + n_{\text{new}},$$

where n_{new} denotes the number of newly added experts. The purpose of expansion is therefore not merely to increase parameter count, but to introduce additional sparse capacity that can absorb new-domain knowledge with less interference to previously learned capabilities.

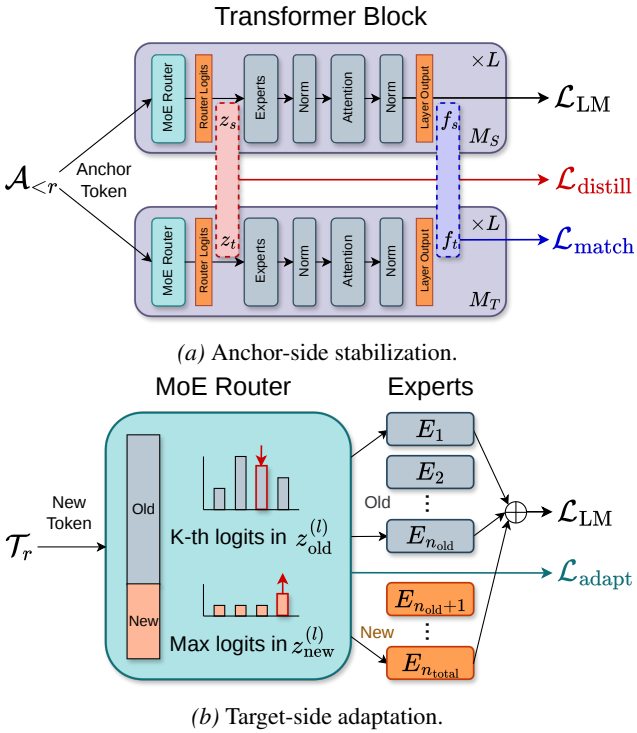

Figure 2. Overview of LEGO for continual learning of MoE language models.

This expansion affects two components of each MoE layer: the expert parameters and the router parameters that produce logits over experts. All other shared parameters keep their original shapes and are copied directly from the checkpoint of round $r - 1$ (or from the pretrained base model when $r = 1$).

As illustrated in Figure 1, we first run the router on target-domain data $\mathcal{T}_r$ and collect expert activation statistics. Based on these scores, we identify the most frequently activated experts in each MoE layer. We then expand the expert bank by adding new experts initialized as copies of the selected source experts. At the same time, we extend the router so that it can produce logits for the newly added experts. The newly introduced router parameters are initialized by copying from the corresponding original entries in the same round-robin manner. This initialization gives the added capacity a warm start from already functional model components rather than random weights.

3.2. Batch-wise training with selective stabilization

A key difficulty in continual learning is that adaptation and retention impose different, and potentially conflicting, update signals. We coordinate them through a batch-wise training scheme. At each optimization step, we sample either a target-domain batch from $\mathcal{T}_r$ or an anchor batch from $\mathcal{A}_{<r}$, with anchor batches drawn with probability ρ.

The standard language modeling loss $\mathcal{L}_{\mathrm{LM}}$, defined as the next-token cross-entropy, is applied on all batches. This keeps continual learning consistent with pretraining, where next-token prediction is the basic objective for acquiring knowledge. The auxiliary losses introduced below do not replace $\mathcal{L}_{\mathrm{LM}}$; instead, they shape how the resulting updates are allocated, by encouraging new knowledge to be absorbed by the added capacity and by preserving previously learned behavior on the anchor corpus.

Adaptation on the target corpus. After expert expansion, newly added experts are not guaranteed to be utilized. Without additional guidance, the router may continue to favor the existing experts, limiting the benefit of the added capacity. To address this, we introduce a training-time routing preference that encourages target-domain tokens to activate the newly added experts.

Let $z^{(l)}(t) \in \mathbb{R}^E$ denote the router logits for token t at layer l. We partition the logits into old and new expert subsets:

$$z^{(l)}(t) = \left[z^{(l)}_{\mathrm{old}}(t),\ z^{(l)}_{\mathrm{new}}(t)\right].$$

We define a margin-based adaptation loss

$$\mathcal{L}_{\mathrm{adapt}} = \frac{1}{L}\sum_{l=1}^{L}\mathbb{E}_t\Big[\max\Big(0,\ \mathrm{kth}\big(z^{(l)}_{\mathrm{old}}(t), K\big) + \delta - \max z^{(l)}_{\mathrm{new}}(t)\Big)\Big].$$

Here, $\mathrm{kth}(\cdot, K)$ denotes the K-th largest logit among the old experts, where K matches the router top-K setting. This loss encourages the strongest new-expert logits to exceed the routing threshold induced by the old experts, thereby increasing the probability that target-domain tokens activate the newly added experts during training.

On target batches, we optimize

$$\mathcal{L}_{\text{target}} = \mathcal{L}_{\mathrm{LM}} + \lambda_{\mathrm{gate}}\mathcal{L}_{\mathrm{adapt}}.$$

Here, $\mathcal{L}_{\mathrm{LM}}$ continues to drive knowledge acquisition through next-token prediction, while $\mathcal{L}_{\mathrm{adapt}}$ biases this adaptation toward the newly added experts. As a result, the added capacity receives a larger share of target-corpus updates and becomes the main carrier of new-domain knowledge.

Preservation on the anchor corpus. While routing pressure promotes adaptation on $\mathcal{T}_r$, it may also perturb representations and routing patterns learned in previous rounds. To mitigate this drift, we apply stabilization objectives on anchor data. We use the checkpoint from round $r-1$ as a frozen teacher and denote the current model as the student.

Feature matching. We first preserve hidden representations through feature matching. Let $h^{(l)}_s(x)$ and $h^{(l)}_t(x)$ denote the hidden representations at layer l of the student and teacher, respectively, for an anchor input $x \sim \mathcal{A}_{<r}$. We define

$$\mathcal{L}_{\mathrm{match}} = \frac{1}{L}\sum_{l=1}^{L}\mathbb{E}_{x\sim\mathcal{A}_{<r}}\left[\left\|h^{(l)}_s(x) - h^{(l)}_t(x)\right\|_2^2\right].$$

This term regularizes the student in function space, encouraging it to reproduce the representations of the previous model on earlier distributions.

Router distillation. We additionally preserve routing behavior over the original expert set. Let $z^{(l)}_s(x)$ and $z^{(l)}_t(x)$ denote the router logits at layer l for the student and teacher, respectively. Restricting both logits to the old-expert subset, we define

$$p^{(l)}_t(x) = \mathrm{softmax}\Big(z^{(l)}_{t,\mathrm{old}}(x)/\tau\Big),$$
$$p^{(l)}_s(x) = \mathrm{softmax}\Big(z^{(l)}_{s,\mathrm{old}}(x)/\tau\Big).$$

The router distillation loss is then

$$\mathcal{L}_{\mathrm{distill}} = \frac{1}{L}\sum_{l=1}^{L}\mathbb{E}_{x\sim\mathcal{A}_{<r}}\left[\tau^2\,\mathrm{KL}\Big(p^{(l)}_t(x)\ \Big\|\ p^{(l)}_s(x)\Big)\right].$$

Restricting distillation to the old-expert subset preserves previously learned routing patterns while leaving the newly added experts free to specialize. On anchor batches, we optimize

$$\mathcal{L}_{\text{anchor}} = \mathcal{L}_{\mathrm{LM}} + \lambda_{\mathrm{feat}}\mathcal{L}_{\mathrm{match}} + \lambda_{\mathrm{router}}\mathcal{L}_{\mathrm{distill}}.$$

Here, the feature-matching term stabilizes the model in representation space, while the router distillation term stabilizes how earlier inputs are assigned to the existing experts.

Putting the pieces together, the batch-wise training objective is

$$\mathcal{L}(B) = \begin{cases}\mathcal{L}_{\text{target}}(B), & B \sim \mathcal{T}_r,\\ \mathcal{L}_{\text{anchor}}(B), & B \sim \mathcal{A}_{<r}.\end{cases}$$

This formulation cleanly decouples adaptation from retention: target-domain batches drive the acquisition of new knowledge, whereas anchor batches provide a stabilization signal that helps preserve previously learned behavior and reduce forgetting.

We treat different parameter-update policies, such as updating all parameters, only MoE-related parameters, or only the newly added experts and their associated router parameters, as implementation variants and study them in the experiments.

3.3. Parameter update policies

We also consider different parameter update policies during continual learning. After expert expansion, updates may be applied to all parameters, limited to the MoE components while freezing shared non-expert components such as attention and normalization layers, or further restricted to the newly added experts and their associated router parameters. We treat these choices as implementation-level variants rather than core components of the method, and study them in the experiments.

4. Experiments

We evaluate LEGO in a math-domain continual-learning setup. Starting from OLMoE-1B-7B-0125, we adapt the model on math data and evaluate performance on math reasoning benchmarks. Our experiments study three questions: **RQ1:** Does vanilla continual learning improve math performance without forgetting previous behavior? **RQ2:** How strong is vanilla continual learning with direct data mixing, without expert expansion or our stabilization design? **RQ3:** Does LEGO outperform these vanilla continual-learning baselines on GSM8K-Flex and GSM8K-Strict?

4.1. Experimental Setup

Model and optimization setup. All experiments start from OLMoE-1B-7B-0125 (Muennighoff et al., 2024a), an open MoE language model with sparse top-8 routing. Table 1 summarizes the default optimization settings used in the main experiments. Most architectural and optimization choices follow the OLMoE continuation recipe, while a small number of systems-level settings are chosen for our continual-learning pipeline. Unless otherwise specified, all compared methods use the same configuration and the same 3000-step training budget. All headline comparisons are run on 8 nodes $\times$ 8 H200 GPUs, so every reported result is obtained under a matched 64-GPU budget.

Hyperparameter	Value	Source
Sequence length	4096	OLMoE-style mid-training recipe
Per-device batch size	1	Ours
Gradient accumulation	4	Ours
Learning rate	6×10^{-6}	OLMoE-style small LR continuation
Warmup steps	5	Ours
Weight decay	0.0	OLMoE-style continuation setting
Gradient clipping	1.0	OLMoE-style stability setting
Precision	BF16	OLMoE / hardware-compatible default
MoE top-k	8	OLMoE architecture
Student load mode	parallel / FSDP-rank0	Systems choice

Table 1. Default optimization settings for the main experiments. Most optimization and architectural settings follow the OLMoE configuration, while only a small number of systems-level choices are adjusted for our continual-learning pipeline. All headline runs use 8 nodes $\times$ 8 H200 GPUs (64 H200 GPUs total).

Corpus	Source split	Role	Samples	Tokens
Math Target Corpus	`stage4/math_oss`	$\mathcal{T}_r$	14.84M	26.27B
Anchor Corpus	`stage2/general`	$\mathcal{A}_{<r}$	7.30M	12.06B

Table 2. Summary of the target and anchor corpora used in the math-domain continual-learning. Both corpora are drawn from the TxT360-Midas dataset (LLM360, 2026).

Training corpora. Table 2 summarizes the corpora used in the math-domain experiments. Following Section 3, we denote the target corpus by $\mathcal{T}_r$ and the anchor corpus by $\mathcal{A}_{<r}$. In this setup, $\mathcal{T}_r$ is drawn from the `math_oss` split of TxT360-Midas (LLM360, 2026), while $\mathcal{A}_{<r}$ is drawn from the `stage2/general` split of TxT360-Midas. The anchor corpus is used only in methods that include replay-based stabilization, and its source is kept fixed across methods for a controlled comparison. Token counts are estimated with the OLMoE-1B-7B-0125 tokenizer.

Compared methods. We compare four checkpoints, including two baselines, the pretrained reference model, and our method. OLMoE-1B-7B-0125 is the untouched pretrained model. `Noexpand_CP` is a vanilla continual learning baseline, where the original MoE is directly trained on math data without expert expansion. `Noexpand_MT` is a vanilla continual learning baseline with direct data mixing: replay data is mixed with math data during training, but the model architecture remains unchanged and our method is not applied. LEGO is our full method, which adds new experts and combines target-side routing pressure with anchor-side stabilization. All adapted models are trained from the same base checkpoint.

Evaluation. We evaluate on GSM8K with 8-shot chain-of-thought prompting and report exact-match accuracy under two answer-extraction filters: `flexible-extract` and `strict-match`. For brevity, we denote them as GSM8K-Flex and GSM8K-Strict. Although both metrics are computed on the same GSM8K task, they serve different purposes. The `flexible-extract` uses a more permissive answer-extraction rule, so it mainly reflects whether the model can produce the correct answer somewhere in its response; in our setting, it is used to assess whether math

Method	GSM8K-Flex	GSM8K-Strict	Avg.
OLMoE-1B-7B-0125	52.08	44.05	48.07
Noexpand_CP	63.61	15.54	39.58
Noexpand_MT	63.91	40.03	51.97
LEGO	**63.23**	**52.01**	**57.62**

Table 3. Math benchmark comparison for the base checkpoint and the continual adaptation stage. Starting from OLMoE-1B-7B-0125, the model is adapted to math data. We report GSM8K-Flex, GSM8K-Strict, and their average.

capability improves after training on math data. By contrast, `strict-match` requires the predicted final answer to satisfy a stricter matching rule, making it more sensitive to whether the model preserves the answer behavior expected by the original evaluation. Therefore, strong performance on GSM8K-Flex indicates successful acquisition of math-domain capability, while degradation on GSM8K-Strict is more diagnostic of forgetting previously useful evaluation-compatible output behavior.

4.2. Main Results

RQ1: Vanilla continual learning causes severe degradation on strict GSM8K evaluation. Table 3 shows that Noexpand_CP, i.e., vanilla continual learning without our method, improves GSM8K under `flexible-extract` from 52.08 to 63.61, but its `strict-match` score collapses from 44.05 to 15.54. Since these two columns correspond to the same GSM8K task under two answer-extraction rules, this gap indicates a major failure mode of vanilla continual learning: after direct training on math data, the model appears better under permissive extraction, but performs much worse under strict evaluation. This strongly suggests that direct vanilla continual learning does not reliably preserve the answer behavior required by GSM8K `strict-match`, including stable final-answer formatting and other evaluation-sensitive output patterns. As a result, the average score also drops from 48.07 to 39.58. This failure motivates the need for a method that can acquire math-domain knowledge without overwriting previously useful output behavior.

RQ2: Direct data mixing partially mitigates this failure mode, but is still not enough. Noexpand_MT is still vanilla continual learning, but with replay/preservation data directly mixed into training. This substantially improves over Noexpand_CP, raising GSM8K `strict-match` from 15.54 to 40.03 and the average from 39.58 to 51.97. However, it still remains below the base model on `strict-match` (40.03 vs. 44.05), which means that direct data mixing only partially alleviates the forgetting problem. Compared with LEGO, it is lower by 11.98 points on `strict-match` and 5.65 points on the average. Therefore, simply mixing replay/preservation data helps vanilla continual learning, but it is not sufficient to match our method.

RQ3: LEGO achieves the best and most robust math performance. LEGO obtains the best overall result, with 63.23 on GSM8K-Flex, 52.01 on GSM8K-Strict, and 57.62 on average. Compared with the base model, LEGO improves both GSM8K-Flex and GSM8K-Strict simultaneously. Compared with vanilla continual learning with direct data mixing (Noexpand_MT), it keeps GSM8K-Flex at a comparable level (63.23 vs. 63.91) while delivering a much stronger GSM8K-Strict result (52.01 vs. 40.03). These results show that the gain cannot be explained by vanilla continual learning alone, even when replay/preservation data is directly mixed into training. The full method is necessary to acquire math-domain knowledge without overwriting previously useful routing behavior.

4.3. Ablation Study

We first analyze the core components of LEGO on the math-domain continual learning stage. Table 4 studies three ingredients: target-side routing pressure, anchor-side feature matching, and expert expansion. Then we study parameter update policies in Table 6, with the detailed definitions of the four policies summarized in Table 5.

Method	GSM8K-Flex	GSM8K-Strict	Avg.
LEGO	**63.23**	**52.01**	**57.62**
w/o $\mathcal{L}_{\text{adapt}}$	62.93	44.96	53.80
w/o $\mathcal{L}_{\text{match}}$	62.62	45.34	53.98
w/o expert expansion	61.11	43.90	52.51

Table 4. Ablation of the core ingredients of LEGO in the first continual learning stage on the math domain. All rows report absolute scores.

Update policy	Trainable parameters
Full update	All parameters are updated, including the dense trunk, all experts, and all router parameters.
MoE-only	The dense trunk is frozen. Only MoE parameters are updated, i.e., all experts and all router parameters, including both old and newly added ones.
Old-prefix-frozen	The dense trunk remains trainable. Within each MoE layer, the old expert subspace is frozen together with the corresponding router outputs, while the newly added experts and their router outputs are updated.
New-MoE-only	The dense trunk is frozen. Within each MoE layer, only the newly added experts and their corresponding router outputs are updated; the old experts and old router outputs remain frozen.

Table 5. Different parameter update policies.

Table 4 shows that all three components are important, with GSM8K-Strict being substantially more sensitive

than GSM8K-Flex. Removing target-side routing pressure causes only a minor drop on GSM8K-Flex (-0.30), but a much larger drop on GSM8K-Strict (-7.05). A similar pattern holds when feature matching is removed (-0.61 on Flex versus -6.67 on Strict), indicating that these terms mainly improve exact-answer reliability rather than loose extraction accuracy. Among all ablations, removing expert expansion is the most damaging, reducing the average by 5.11 points and GSM8K-Strict by 8.11 points. This result suggests that added sparse capacity is the central ingredient, while routing pressure and anchor-side stabilization make that capacity more effective and less interference-prone.

Table 6 further shows that the full-update policy performs best. Among the restricted variants, freezing only the old MoE prefix is the strongest alternative, remaining close on GSM8K-Flex (62.85 versus 63.23) while still trailing on GSM8K-Strict (46.55 versus 52.01). By contrast, freezing the dense trunk, as in the MoE-only policy, leads to a much larger degradation, especially on GSM8K-Strict (39.80). Restricting learning even further to only the newly added MoE capacity recovers slightly over the MoE-only policy, but still remains far below full updating. Taken together, these results indicate that expert expansion works best when the shared trunk is allowed to co-adapt with the MoE layers, rather than when adaptation is forced into a narrowly isolated subset of parameters. When stronger protection is desired, freezing only the old MoE prefix provides a substantially better trade-off than freezing the dense trunk.

Freeze mode	GSM8K-Flex	GSM8K-Strict	Avg.
LEGO (Full update)	**63.23**	**52.01**	**57.62**
MoE_only	59.67	39.80	49.74
Old_prefix	62.85	46.55	54.70
New_MoE_only	60.80	41.02	50.91

Table 6. Ablation of parameter update policies. We compare full updating with progressively more restrictive freezing rules.

5. Conclusion

We study continual learning of mixture-of-experts language models under sequential domain shift and argue that the central challenge is not only how to update the model, but also where new knowledge should be stored. To address this, we propose LEGO, a simple framework that expands the expert pool, encourages target-domain tokens to use the added capacity, and stabilizes prior behavior on anchor data. In the math-domain setting, LEGO achieves a stronger adaptation–retention trade-off than vanilla continual learning baselines. Overall, our results suggest that explicit sparse capacity allocation is an effective principle for continually updating pretrained MoE models while preserving the expert organization acquired during pretraining.

6. Future Work.

A next step is to extend this capacity-allocation perspective beyond continual pretraining stage to reinforcement-learning-based adaptation. In settings such as RLHF, online agent training, and continual reinforcement learning, the training distribution is policy-dependent and evolves with the policy itself, which can make interference even more severe than in static corpora. A promising direction is to couple expert expansion with reward-aware routing or trajectory-level credit assignment, so that newly added experts absorb behaviors associated with new tasks, environments, or reward functions, while anchor-side stabilization preserves previously learned reasoning, safety, and policy behaviors. Studying when to expand, reuse, merge, or retire experts during long-horizon policy optimization may further improve the robustness and efficiency of sparse models in interactive settings.

LLM/Agent Usage Disclosure

We used LLM-based tools to assist with manuscript editing, formatting, and LaTeX debugging. The research ideas, experimental design, implementation, analysis, and final claims were developed and verified by the authors.

References

Bai, J., Tong, M., Liu, Y., Jia, Z., and Zheng, Z. Understanding and leveraging the expert specialization of context faithfulness in mixture-of-experts LLMs. In Christodoulopoulos, C., Chakraborty, T., Rose, C., and Peng, V. (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 21927–21942, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1114. URL https://aclanthology.org/2025.emnlp-main.1114/.

DeepSeek-AI. Deepseek-v3 technical report, 2024. URL https://arxiv.org/abs/2412.19437.

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

Guo, H., Lu, H., Nan, G., Chu, B., Zhuang, J., Yang, Y., Che, W., Cao, X., Leng, S., Cui, Q., and Jiang, X. Advancing expert specialization for better moe, 2026. URL https://arxiv.org/abs/2505.22323.

Gururangan, S., Marasovic, A., Swayamdipta, S., Lo, K., Beltagy, I., Downey, D., and Smith, N. A. Don't stop

pretraining: Adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*, 2020.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., de Las Casas, D., Hendricks, L. A., Welbl, J., Clark, A., Hennigan, T., Noland, E., Millican, K., van den Driessche, G., Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., Rae, J. W., Vinyals, O., and Sifre, L. Training compute-optimal large language models, 2022. URL https://arxiv.org/abs/2203.15556.

Ibrahim, A., Thérien, B., Gupta, K., Richter, M. L., Anthony, Q., Lesort, T., Belilovsky, E., and Rish, I. Simple and scalable strategies to continually pre-train large language models, 2024. URL https://arxiv.org/abs/2403.08763.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

Lepikhin, D., Lee, H., Xu, Y., Chen, D., Firat, O., Huang, Y., Krikun, M., Shazeer, N., and Chen, Z. Gshard: Scaling giant models with conditional computation and automatic sharding, 2020. URL https://arxiv.org/abs/2006.16668.

Li, C., Deng, Y., Zhang, J., and Zong, C. Group then scale: Dynamic mixture-of-experts multilingual language model. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 1730–1754, 2025a.

Li, J., Wang, B., Zhou, X., and Hu, X. Dynamic expert specialization: Towards catastrophic forgetting-free multi-domain MoE adaptation. In Christodoulopoulos, C., Chakraborty, T., Rose, C., and Peng, V. (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 18478–18493, Suzhou, China, November 2025b. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.932. URL https://aclanthology.org/2025.emnlp-main.932/.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. Federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 2018.

Li, Z. and Hoiem, D. Learning without forgetting. *arXiv preprint arXiv:1606.09282*, 2016.

Liu, E., Neubig, G., and Xiong, C. Midtraining bridges pretraining and posttraining distributions. *arXiv preprint arXiv:2510.14865*, 2025.

Liu, E., Neubig, G., and Xiong, C. Midtraining bridges pretraining and posttraining distributions, 2026. URL https://arxiv.org/abs/2510.14865.

LLM360. TxT360-MidAS: Mid-training Actual and Synthetic Data, 2026. URL https://huggingface.co/datasets/LLM360/TxT360-Midas.

Lo, K. M., Huang, Z., Qiu, Z., Wang, Z., and Fu, J. A closer look into mixture-of-experts in large language models. In Chiruzzo, L., Ritter, A., and Wang, L. (eds.), *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 4427–4447, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-195-7. doi: 10.18653/v1/2025.findings-naacl.251. URL https://aclanthology.org/2025.findings-naacl.251/.

Mo, K., Shi, Y., Weng, W., Zhou, Z., Liu, S., Zhang, H., and Zeng, A. Mid-training of large language models: A survey. *arXiv preprint arXiv:2510.06826*, 2025.

Muennighoff, N., Soldaini, L., Groeneveld, D., Lo, K., Morrison, J., Min, S., Shi, W., Walsh, P., Tafjord, O., Lambert, N., Gu, Y., Arora, S., Bhagia, A., Schwenk, D., Wadden, D., Wettig, A., Hui, B., Dettmers, T., Kiela, D., Farhadi, A., Smith, N. A., Koh, P. W., Singh, A., and Hajishirzi, H. Olmoe: Open mixture-of-experts language models. *arXiv preprint arXiv:2409.02060*, 2024a.

Muennighoff, N., Soldaini, L., Groeneveld, D., Lo, K., Morrison, J., Min, S., Shi, W., Walsh, P., Tafjord, O., Lambert, N., Gu, Y., Arora, S., Bhagia, A., Schwenk, D., Wadden, D., Wettig, A., Hui, B., Dettmers, T., Kiela, D., Farhadi, A., Smith, N. A., Koh, P. W., Singh, A., and Hajishirzi, H. Olmoe: Open mixture-of-experts language models. *arXiv preprint arXiv:2409.02060*, 2024b.

NVIDIA. Nvidia nemotron 3: Efficient and open intelligence, 2025. URL https://arxiv.org/abs/2512.20856. White Paper.

OLMo, T., Walsh, P., Soldaini, L., Groeneveld, D., Lo, K., Arora, S., Bhagia, A., Gu, Y., Huang, S., Jordan, M., Lambert, N., Schwenk, D., Tafjord, O., Anderson, T., Atkinson, D., Brahman, F., Clark, C., Dasigi, P., Dziri, N., Ettinger, A., Guerquin, M., Heineman, D., Ivison, H., Koh, P. W., Liu, J., Malik, S., Merrill, W., Miranda, L. J. V., Morrison, J., Murray, T., Nam, C., Poznanski, J., Pyatkin, V., Rangapur, A., Schmitz, M., Skjonsberg, S., Wadden, D., Wilhelm, C., Wilson, M., Zettlemoyer, L., Farhadi, A., Smith, N. A., and Hajishirzi, H. 2 olmo 2 furious, 2025. URL https://arxiv.org/abs/2501.00656.

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

Parmar, J., Satheesh, S., Patwary, M., Shoeybi, M., and Catanzaro, B. Reuse, don't retrain: A recipe for continued pretraining of language models, 2024. URL https://arxiv.org/abs/2407.07263.

Qiu, Z., Huang, Z., Zheng, B., Wen, K., Wang, Z., Men, R., Titov, I., Liu, D., Zhou, J., and Lin, J. Demons in the detail: On implementing load balancing loss for training specialized mixture-of-expert models, 2025. URL https://arxiv.org/abs/2501.11873.

Runwal, B., Agrawal, A., Roy, A., and Panda, R. Prism: Demystifying retention and interaction in mid-training, 2026. URL https://arxiv.org/abs/2603.17074.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer, 2017. URL https://arxiv.org/abs/1701.06538.

Team, K., Liu, Z., Tang, L., Jin, L., Li, H., Ranjan, N., Fan, D., Rohatgi, S., Fan, R., Pangarkar, O., Wang, H., Cheng, Z., Sun, S., Han, S., Tan, B., Gosal, G., Han, X., Pimpalkhute, V., Hao, S., Hee, M. S., Hestness, J., Jia, H., Ma, L., Singh, A., Soboleva, D., Vassilieva, N., Wang, R., Wu, Y., Sun, Y., Killian, T., Moreno, A., Maggs, J., Ren, H., He, G., Wang, H., Ma, X., Wang, Y., Yurochkin, M., and Xing, E. P. K2-v2: A 360-open, reasoning-enhanced llm, 2025. URL https://arxiv.org/abs/2512.06201.

Tu, C., Zhang, X., Weng, R., Li, R., Zhang, C., Bai, Y., Yan, H., Wang, J., and Cai, X. A survey on llm mid-training. *arXiv preprint arXiv:2510.23081*, 2025.

Wang, L., Zhang, X., Su, H., and Zhu, J. A comprehensive survey of continual learning: Theory, method and application, 2024a. URL https://arxiv.org/abs/2302.00487.

Wang, Z., Chen, D., Dai, D., Xu, R., Li, Z., and Wu, Y. Let the expert stick to his last: Expert-specialized fine-tuning for sparse architectural large language models. In Al-Onaizan, Y., Bansal, M., and Chen, Y.-N. (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 784–801, Miami, Florida, USA, November 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.46. URL https://aclanthology.org/2024.emnlp-main.46/.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., et al. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Zeng, A., Lv, X., Hou, Z., Du, Z., Zheng, Q., Chen, B., Yin, D., Ge, C., Xie, C., Wang, C., et al. Glm-5: from vibe coding to agentic engineering, 2026. URL https://arxiv.org/abs/2602.15763.

Zhu, T., Qu, X., Dong, D., Ruan, J., Tong, J., He, C., and Cheng, Y. Llama-moe: Building mixture-of-experts from llama with continual pre-training. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pp. 15913–15923, 2024.