CLoRE: Drift-Controlled Continual Learning of LLMs via Expert Expansion with Reservoir Routing

Anonymous authors
Paper under double-blind review

Abstract

Continual learning of large language models (LLMs) by mixture-of-experts (MoE) expansion freezes earlier experts and adds new ones, which in principle prevents forgetting. However, the router is shared, and new experts added to it can displace or down-weight an earlier input's experts. Most prior methods constrain routing during or after training. Yet what must be preserved is the representation, which can drift as soon as new experts are added. We propose CLoRE, which controls this representational drift without storing past data. *Concurrent router alignment* trains the router at every step with the language-modeling loss, matching no previous router. *Reservoir routing* initializes new experts and their router rows so that expansion preserves the softmax denominator in expectation. *Self-generated replay* steers each question toward an earlier task by routing only its first token through that task's experts, and replays the backbone. On TRACE with two LLMs (*e.g.*, Llama-3.1 and Qwen3) and on an MoE trained from scratch, CLoRE outperforms every baseline in average accuracy and forgetting, even one using original data, and largely preserves the backbone's general abilities.

1 Introduction

Once deployed, large language models (LLMs) face new domains and user needs that their training did not cover, so they have to keep learning new tasks (Shi et al., 2024). Retraining on all data seen so far is expensive, and it is often not even possible. Data from earlier fine-tuning tasks can be stored at low cost (Cho et al., 2026), but the pre-training data of a released model (*e.g.*, Llama-3 (Grattafiori et al., 2024)) is rarely made public (Shi et al., 2024). Training on the new task alone, however, makes the model forget earlier tasks and lose part of its general abilities (French, 1999; Luo et al., 2025). Continual learning methods aim to avoid this while training only on new data (Shi et al., 2024). Many of them limit how shared parameters change, either by penalizing updates to weights that matter for earlier tasks (Kirkpatrick et al., 2017) or by keeping updates in orthogonal subspaces (Wang et al., 2023a). Since all tasks share the same parameters, these methods must control the trade-off between stability and plasticity (Mermillod et al., 2013) within one parameter space. Expansion-based methods avoid this by adding new parameters for each task and freezing the old ones (Rusu et al., 2016). Many recent expansion methods build on mixture-of-experts (MoE) models, where a router picks a few experts for each token. New experts are added for each task and earlier experts are kept fixed (Chen et al., 2023; Zhou et al., 2025; Zhao et al., 2026). Because the parameters that store earlier tasks are never updated, such a model cannot forget in principle.

This holds for the experts but not for the router, which all inputs share. Each new expert adds a row to the router. For an input from an earlier task, the new row can enter the top-k selection and replace an expert the input was trained with. Even when it does not, it enters the softmax denominator and can lower the weights of the selected experts. As a result, the hidden representations of earlier inputs change even though every expert they use is frozen. We call this change *representational drift*. The underlying change in expert assignment has been called routing drift and identified as a cause of forgetting in MoE continual learners (Zhao et al., 2026; Guo et al., 2026; Hou et al., 2026). Existing methods counter it by keeping the routing of earlier inputs close to what it was, either by freezing or regularizing the router during training or by retraining it afterwards (Chen et al., 2023; Zhou et al., 2025; Zhao et al., 2026; Guo et al., 2026). However, what needs to be preserved is not

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.

the routing itself but the representation it produces. These methods also act only during or after training, even though drift already starts when the new rows are inserted. By the time they step in, the representations of earlier inputs have already drifted.

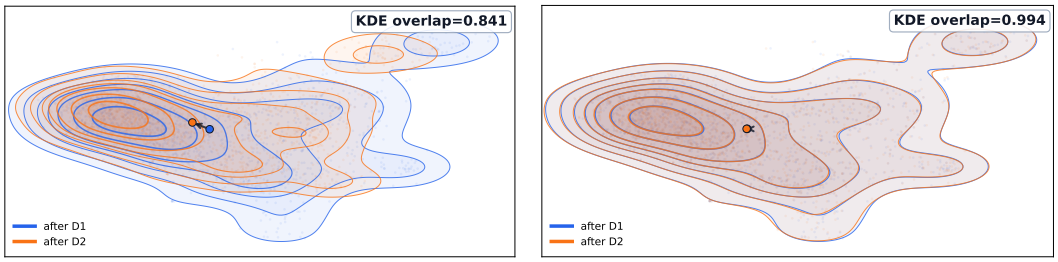
We propose CLoRE (**C**ontinual **L**earning **o**f LLMs via Expert Expansion with **RE**servoir Routing), which addresses both issues without storing past data. (i) During training, *concurrent router alignment* trains the router at every step together with the new experts, on new-task inputs and replayed inputs from earlier tasks, using the same language-modeling loss as the new task. Since there is no routing target, an earlier input may move to other experts as long as the model keeps its output. (ii) At expansion, *reservoir routing* initializes the new experts and their router rows so that adding them barely changes earlier outputs. A single router row reserves the softmax share of experts not yet added. Each new expert takes this row plus small noise as its router row and starts with zero output, so the softmax denominator is preserved in expectation. Since a new expert contributes nothing by itself, it can change an earlier output only by pushing an existing expert out of the top-k, and a margin loss makes this unlikely. (iii) *Self-generated replay* supplies the earlier-task inputs that router alignment needs. Following Magpie (Xu et al., 2025), the model writes a question from the chat template alone, and we route only its first token through the experts of the target task to steer the question toward that task. The pre-trained backbone is replayed as one more task, which protects its general abilities. Our contributions are summarized as follows:

- We show that freezing experts does not preserve what an MoE computes for earlier inputs. New router rows change their routing, and the resulting representational drift begins when the rows are inserted, before routing-level corrections can act.
- We propose CLoRE, which controls this drift at both points. Concurrent router alignment keeps earlier outputs during training with the language-modeling loss, and reservoir routing initializes new experts and router rows so that expansion preserves the softmax denominator in expectation. Both need only replay that the model generates itself.
- Without storing past data, CLoRE reaches the highest average accuracy and the lowest forgetting on TRACE benchmark with two backbones, keeping forgetting at 2.84 and 0.79 points against at least 6.55 and 7.50 for baselines that also store no past data. With Llama-3.1-8B-Instruct, it loses only 0.82 points on MMLU, GSM8K, and PIQA, against at least 7.28 for every baseline. It also outperforms MoE-LPR, which replays original data, on TRACE and on an MoE trained from scratch, where earlier inputs keep more of their original expert selections (90.31% vs. 86.00%) even though CLoRE never constrains routing.

2 Related Work

Continual learning and expert expansion. Most continual learning methods keep one set of parameters and limit how it changes, through regularization (Kirkpatrick et al., 2017; Zenke et al., 2017; Aljundi et al., 2018; Cha et al., 2021) or constrained gradients (Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2019; Farajtabar et al., 2020). For LLMs, O-LoRA learns a low-rank update for each task in a subspace orthogonal to earlier ones (Wang et al., 2023a), and SLoRA denoises each update after training (Yang et al., 2026). Since these updates are shared by all tasks rather than isolated to the new one, the representations of earlier tasks still drift as new tasks are learned, and this drift is closely tied to forgetting (Luo et al., 2023; Cha et al., 2025). Expansion-based methods avoid this by adding capacity for each task and freezing what was learned (Rusu et al., 2016; Yoon et al., 2017; Wang et al., 2025). In MoE-based expansion, however, the router is shared and keeps changing. Prior methods constrain its routing or distill it from a previous model (Chen et al., 2023; Zhou et al., 2025; Zhao et al., 2026), and a concurrent preprint does the same (Guo et al., 2026). Others remove the learned router (Lu et al., 2026; Hou et al., 2026). CLoRE instead updates the router at every training step with the language-modeling loss on new-task and replayed data, without matching a previous router or a fixed expert assignment.

Function-preserving expansion. Adding capacity without changing what a model computes is a recurring idea. Net2Net initializes wider or deeper networks to compute the same function as the original (Chen et al., 2015), LoRA initializes each update to zero (Hu et al., 2022). In continual learning, GainLoRA adds a LoRA branch for each task and initializes its gate to output zero on earlier inputs (Liang et al., 2026). All of these leave the output unchanged at insertion, but this does not carry over to a softmax router. Even with zero output, a new expert enters the router's softmax.



(a) $\mathcal{D}_1$ experts and router rows frozen (b) Ours

Figure 1: **Representational drift of earlier-domain inputs.** Hidden states of inputs from the first domain $\mathcal{D}_1$ at layer 5, right after $\mathcal{D}_1$ was learned (blue) and after a second domain $\mathcal{D}_2$ was learned on top of it (orange), projected onto the top two principal components of the two sets pooled; dots mark the means. Ideally the two would coincide, since the model has only been given new data from a different domain and nothing about $\mathcal{D}_1$ has been changed. (a) $\mathcal{D}_2$ is learned with the experts of $\mathcal{D}_1$ and their router rows frozen; the $\mathcal{D}_1$ distribution still moves. (b) With our method it stays in place. The number in each panel is the overlap of the two distributions in the projected space, $\int \min(p_{\text{before}}, p_{\text{after}})$ with densities from kernel density estimation, so that 1 means identical. Here $\mathcal{D}_1$ is Wiki and $\mathcal{D}_2$ is Code (§4.1).

It adds a term to the shared denominator, which can lower the weights of the experts an earlier input uses, and it can displace one of them from the top-k. CLoRE starts new experts at zero output as LoRA does, and its reservoir routing reserves the new experts' share of the denominator before they exist, so expansion preserves the denominator in expectation. A margin loss also makes top-k changes unlikely.

Generative replay. Generative replay replaces stored examples of earlier tasks with generated ones, produced by a separate generator (Shin et al., 2017) or by the learner itself. LAMOL trains the model to generate pseudo-samples of earlier tasks from task-specific tokens (Sun et al., 2020), and SSR generates them by in-context learning from a few retained or hand-written examples and refines their outputs with the latest model (Huang et al., 2024). Outside continual learning, Magpie prompts an aligned model with only the chat-template prefix up to the user turn to obtain diverse instructions, and steers their domain with a system prompt (Xu et al., 2025). For router alignment, CLoRE generates questions from the same prefix and steers them toward an earlier task by routing the first generated token through that task's experts. It therefore needs no stored examples, task tokens, or task description.

3 Method

Expansion-based continual learning adds capacity per domain and freezes what was learned before, on the premise that frozen parameters cannot forget. In a mixture-of-experts this premise fails at the router: earlier-domain inputs are rerouted as experts are added, and their representations move even though every expert is frozen. We treat this rerouting, not parameter change, as the object to control. We first introduce the architecture and characterize its representation drift in §3.1 and §3.2. We then address drift during training and at expansion in §3.3 and §3.4, respectively, using self-generated earlier-domain data from §3.5. §3.6 combines these components into the full procedure.

3.1 Problem Setting and LoRA-MoE with Skip Experts

In continual learning, a pre-trained causal language model is trained on domains $\mathcal{D}_1, \ldots, \mathcal{D}_T$ arriving one at a time; during stage t only $\mathcal{D}_t$ is available, and earlier data is not stored. The pre-trained weights are frozen and the model may grow by a bounded number of parameters per domain. After stage t it is evaluated on every $\mathcal{D}_\tau$ with $\tau \leq t$ and on the backbone's own distribution, indexed $\tau = 0$, with no domain identity given at evaluation. The goal is to fit each new domain while leaving behavior on all earlier ones, backbone included, unchanged. In place of stored data we use a replay set $\mathcal{R}_{<t}$ generated by the model itself (§3.5).

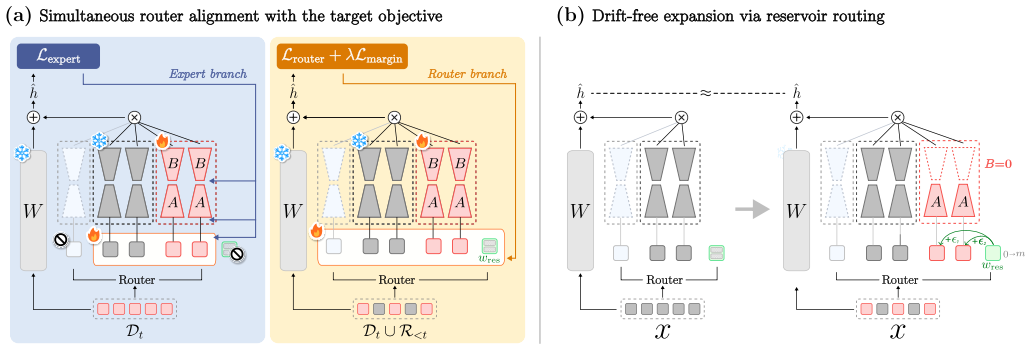


Figure 2: **Overview of CLoRE (a)** Two weight-sharing branches per step; colored arrows show gradient flow. The expert branch excludes the skip row and the reservoir ($\oslash$); the router branch updates only the router, including the reservoir w_{res}, which enters the softmax denominator but is never selected. **(b)** Expansion draws the new rows from the reservoir ($w_{\text{res}} + \epsilon_j$, $B = \mathbf{0}$), transferring its share of the denominator so that earlier-domain outputs are preserved in expectation.

Let W be the frozen weight of a dense layer with output $h = Wx$. We augment it with LoRA experts $E_i(x) = B_i A_i x$; each domain τ adds m experts $\mathcal{E}_\tau$, frozen once the domain is learned, so the pool after domain t is $\mathcal{E}^{(t)}_{\text{LoRA}} = \bigcup_{\tau \le t} \mathcal{E}_\tau$. A router W_r holds a row w_i per expert with logit $z_i(x) = w_i x$ and selects the k experts with the highest softmax probabilities, $\mathcal{S}_k(x)$, using their probabilities as weights without renormalization:

$$p_i(x) = \frac{e^{z_i(x)}}{\sum_{i'} e^{z_{i'}(x)}}, \qquad \hat{h} = h + \sum_{i \in \mathcal{S}_k(x)} p_i(x)\, E_i(x). \tag{1}$$

The routing space also contains k *skip experts* $\mathcal{E}_{\text{skip}}$ that output $\mathbf{0}$, one per selection slot, so that if all k selections fall on them, $\hat{h} = h$: the router has an explicit option to leave the backbone untouched. We take $m \ge k$ so that a domain's own experts can fill every slot. Whether an earlier-domain input is actually routed to skip depends on the router, which is exactly what changes as domains arrive.

3.2 Representational Drift

Freezing the backbone, the earlier experts, and even their rows in W_r does not fix what the model computes for earlier-domain inputs. The rows added for $\mathcal{D}_t$ alter their routing in two ways: a new expert can enter the top-k and displace one the input was learned with, and even when it does not, the new logits enlarge the softmax denominator and shrink the weights $p_i(x)$ of the selected experts. Either way $\hat{h}$ moves away from the value it had when the domain was learned. We call this *representational drift*. It is caused entirely by the added rows, as they change during training on $\mathcal{D}_t$ (§3.3) and at the moment they are inserted (§3.4).

Fig. 1 shows it on the hidden states of $\mathcal{D}_1$ inputs before and after $\mathcal{D}_2$ is learned. With the $\mathcal{D}_1$ experts and their router rows frozen (Fig. 1a), the distribution still moves and the two overlap by only 0.84; with our method (Fig. 1b) it stays in place at 0.99.

3.3 Simultaneous Router Alignment with the Target Objective

Training on $\mathcal{D}_t$ must fit the new domain and keep earlier-domain outputs where they were. We run two branches in every step (Fig. 2a). The *expert branch* trains the new experts and their router rows on $\mathcal{D}_t$. The *router branch* trains the router on $\mathcal{D}_t \cup \mathcal{R}_{<t}$ so that earlier-domain inputs keep their routing while the new domain is learned. It has no teacher: it uses the same language modeling loss $\mathcal{L}_{\text{LM}}$ as the expert branch, so earlier-domain routing is held in place by the task itself. The two run concurrently because drift begins as soon as the new rows start moving; aligning the router afterwards would let earlier-domain routing be pushed aside first and then try to restore it.

Fresh LoRA experts are worse than the backbone, so under $\mathcal{L}_{\text{LM}}$ the router would send new-domain inputs to skip, and the new experts, rarely selected, would never catch up; we call this *skip collapse*.

The expert branch therefore masks out the skip rows, so the new experts get selected and trained. The mask cannot be used at inference, so the router branch routes new-domain data over the full pool; since the expert branch improves the new experts at the same time, by the time the router must choose between a new expert and skip, the expert is already the better choice. The two branches are

$$\mathcal{L}_{\text{expert}} = \mathcal{L}_{\text{LM}}\Big(\mathcal{D}_t \;\Big|\; \mathcal{E}^{(t)}_{\text{LoRA}}\Big), \qquad \mathcal{L}_{\text{router}} = \mathcal{L}_{\text{LM}}\Big(\mathcal{D}_t \cup \mathcal{R}_{<t} \;\Big|\; \mathcal{E}^{(t)}_{\text{LoRA}} \cup \mathcal{E}_{\text{skip}}\Big), \tag{2}$$

where $\mathcal{L}_{\text{LM}}(\mathcal{D} \mid \mathcal{E})$ restricts routing to $\mathcal{E}$; the first updates the new experts and their rows, the second updates W_r only.

Existing methods align the router by distilling it from its previous state, zero-padded to the new pool. That target can only be approximated, and the approximation error is itself drift. The target objective admits the same error but is not indifferent to it: among updates that move earlier-domain representations by a similar amount, $\mathcal{L}_{\text{LM}}$ prefers those that lower the loss, so the residual drift is the kind that costs the task least.

3.4 Drift-Controlled Expansion via Reservoir Routing

Drift has a second source that precedes any training step. Adding m rows to W_r adds m terms to the denominator $Z(x) = \sum_i e^{z_i(x)}$, so every existing $p_i(x)$ shrinks the moment they appear, and a large new logit changes $\mathcal{S}_k(x)$ outright. Earlier-domain outputs are already displaced when the router branch starts.

We allocate the new rows' share of $Z(x)$ before they exist (Fig. 2b). A *reservoir* is a single row w_{res} with a count n, standing for n experts not yet instantiated whose rows are modeled as draws from $\mathcal{N}(w_{\text{res}}, \sigma^2 I)$; it enters the denominator as their expected mass,

$$Z(x) = \sum_{i \in \mathcal{E}^{(t)}_{\text{LoRA}} \cup \mathcal{E}_{\text{skip}}} e^{z_i(x)} \;+\; n\, e^{z_{\text{res}}(x) + \sigma^2 \|x\|^2/2}, \qquad z_{\text{res}}(x) = w_{\text{res}}\, x, \tag{3}$$

and is never a selection candidate. Expansion draws the m new experts from this prior, $w_j = w_{\text{res}} + \epsilon_j$ with $\epsilon_j \sim \mathcal{N}(0, \sigma^2 I)$, random A_j, $B_j = \mathbf{0}$, and reduces the count to $n - m$. Then a new expert outputs $\mathbf{0}$ even if selected, and the denominator is preserved in expectation because the drawn rows carry exactly the mass removed from the reservoir (§H). $\mathcal{S}_k(x)$ is preserved provided no drawn logit exceeds the k-th largest existing logit $\bar{z}_{(k)}(x)$, which holds with probability $1 - \eta$ if $z_{\text{res}}(x)$ lies at least $\delta(x) = \sigma\|x\|\sqrt{2\ln(m/\eta)}$ below it. This last condition is not given by the construction; we enforce it in the router branch with a hinge loss,

$$\mathcal{L}_{\text{margin}} = \sum_{x \in \mathcal{D}_t \cup \mathcal{R}_{<t}} \big[\, z_{\text{res}}(x) - \bar{z}_{(k)}(x) + \delta(x) \,\big]_+, \tag{4}$$

added to $\mathcal{L}_{\text{router}}$ with weight λ. The noise ϵ_j is what lets the m new experts, which start identical, diverge. We set $n = m$, one expansion's worth; after expansion the count is 0 and is restored to m during training, so the number of expansions is unbounded at the cost of one extra logit per layer.

3.5 Self-Generated Replay

The router branch does not need the original data of earlier domains; it needs inputs the model currently handles as domain τ, with whatever it currently produces for them, so that the router preserves the model's own behavior. We obtain them by sampling from the model itself, immediately before each expansion, for every $\tau < t$ including the backbone as $\tau = 0$. Generation starts from a fixed prefix shared by all domains, so the only place the domain can be injected is the first generated token: there we override the router in every layer to select among $\mathcal{E}_\tau$ only,

$$p_i = \frac{e^{z_i}}{\sum_{i' \in \mathcal{E}_\tau} e^{z_{i'}}} \;\text{ for } i \in \mathcal{E}_\tau, \qquad \mathcal{S}_k = \text{TopK}_{i \in \mathcal{E}_\tau}\, p_i, \tag{5}$$

and route every later position freely. The experts of τ are biased toward τ, so the first token is of that domain, and we observe that the rest of the sequence stays there under the model's own routing. For $\tau = 0$ no override is applied, so the backbone's behavior is replayed as well and the router learns which inputs to send to skip. Replay is regenerated before every expansion so that it always reflects the current model (§E).

3.6 Overall Procedure

Starting from the frozen backbone with k skip experts and a full reservoir, each domain is processed by generating $\mathcal{R}_{<t}$ (§3.5), drawing m experts from the reservoir (§3.4), and training with the two branches of §3.3 while the reservoir refills; the new experts are then frozen. At inference the router runs over all experts and skip, with the reservoir in the denominator, and no domain identity is given. Algorithm 1 lists the procedure.

4 Experiments

4.1 Setup

Benchmarks. On TRACE (Wang et al., 2023b), eight tasks in the order of Yang et al. (2026), we use Llama-3.1-8B-Instruct (Grattafiori et al., 2024) and Qwen3-8B (Yang et al., 2025) as frozen backbones, adding one rank-64 LoRA expert per task with top-1 routing ($m = k = 1$). To compare with MoE methods on their own ground, we also train FLAME-MoE (Kang et al., 2025) from scratch on Wiki→Code→Conversation following Lifelong-MoE (Chen et al., 2023), adding eight FFN experts per task with top-4 routing ($m = 8$, $k = 4$); there is no frozen backbone and no skip experts, and the experts of the first task play that role (§K).

Baselines. Seq LoRA, EWC (Kirkpatrick et al., 2017), O-LoRA (Wang et al., 2023a), and S-LoRA (Yang et al., 2026) store no data from earlier tasks and isolate new knowledge by regularization, orthogonal subspaces, or task-specific adapters. Lifelong-MoE (Chen et al., 2023) adds experts per task, freezes earlier ones, and regularizes outputs toward the previous model without replay; MoE-LPR (Zhou et al., 2025) adds experts, then tunes the router on original replay with a prior routing every replayed token to the base expert. On TRACE we reimplement both on the LoRA-MoE of §3.1 with the same expert budget. MTL trains on all tasks at once and serves as the oracle. CLoREstores no data and replays its own generations, matching MoE-LPR's token budget.

Metrics. We report final average accuracy and forgetting, the mean drop of each task from its accuracy right after it was learned.[1] Because a backbone can also lose what it knew before continual learning, we evaluate the final model on MMLU (Hendrycks et al., 2021), GSM8K (Cobbe et al., 2021), and PIQA (Bisk et al., 2020) and report backbone forgetting, the mean drop from the backbone score. On Wiki→Code→Conversation, accuracy is next-token prediction accuracy on held-out text, following Lifelong-MoE. Experts, replay size, and reservoir hyperparameters are in §J.

4.2 Comparison with State-of-the-Art Methods

Comparison without stored data on TRACE. Tab. 1 groups methods by whether they store original data of earlier tasks. Both backbones show the same pattern, so we describe Llama-3.1-8B-Instruct. Among methods that store none, CLoREreaches the highest average accuracy, 59.44, and the lowest forgetting, 2.84, against 6.55 for the next best. The existing methods isolate new knowledge in separate parameters, whether orthogonal subspaces, importance-weighted weights, or task-specific adapters, but leave the representations of earlier tasks to move while the new one is learned; CLoREcontrols that movement, through the router branch on self-generated replay and the reservoir at expansion. Lifelong-MoE, reimplemented as a LoRA-MoE, freezes the experts and router rows of earlier tasks and nothing else, which is exactly the setting of §3.2, and it collapses to 35.11 with forgetting of 16.00.

Comparison with stored data. Given 10% of the original data of earlier tasks, CLoREreaches 63.09 with forgetting 0.28, within 3.1 points of MTL. MoE-LPR with the same data reaches 55.91 with forgetting 3.81, below CLoREwithout any stored data: its routing prior trains the router to reproduce a fixed expert assignment rather than to minimize the task loss. The 3.65 points between self-generated and original replay is the cost of not storing data (§4.4). On Qwen3-8B the two are close, 61.31 against 61.77, since self-generated replay already leaves little to forget there.

Retention of the backbone. Continual learning of a pre-trained LLM has two things to keep: the tasks learned so far and the backbone's general capability, which is what made the model worth

[1] TRACE reports the negated quantity; we follow the sign convention of most subsequent work.

Table 1: **TRACE results and backbone forgetting with Llama-3.1-8B-Instruct (top) and Qwen3-8B (bottom).** Rows are grouped by whether data of earlier tasks is available (Past data); Zero-shot, the backbone before continual learning, and MTL†, multi-task learning on all eight tasks at once, are not continual and serve as reference and upper bound. The second group has no earlier-task data; CLoREreplays 1.07M self-generated tokens instead. The third group replays original data. ‡: our reimplementation on the LoRA-MoE architecture of §3.1; the original architectures are compared in Tab. 2. Left: per-task accuracy after all eight tasks, average accuracy (AA), and forgetting (F). Right: accuracy on MMLU, GSM8K, and PIQA after all eight tasks, evaluated without further training, and backbone forgetting (F_0), the mean drop from Zero-shot. Bold: best within each past-data group of each backbone. All methods use LoRA rank 64 with their own target modules; O-LoRA adapts only q and v, hence a smaller budget, and CLoREexceeds the rest by the added router rows ($+0.08\%$). Task abbreviations: C-ST (C-STANCE), MTB (MeetingBank), SciQA (ScienceQA), NG-cm/ds (NumGLUE-cm/ds), 20Min (20Minuten).

		TRACE										Backbone forgetting			
Method	Past data	C-ST	FOMC	MTB	Py150	SciQA	NG-cm	NG-ds	20Min	AA↑	F↓	MMLU	GSM8K	PIQA	F_0 ↓
Llama-3.1-8B-Instruct															
Zero-shot	–	36.15	43.55	14.97	38.66	87.15	43.21	24.62	38.26	40.82	–	67.97	75.21	81.07	–
Seq LoRA	✗	48.35	**67.54**	37.54	50.88	85.95	**61.73**	**73.54**	**42.01**	58.44	8.35	59.93	46.10	78.84	13.13
EWC	✗	51.20	63.10	39.20	52.41	**90.90**	59.26	71.69	41.76	58.69	7.48	60.23	46.02	79.22	12.93
O-LoRA	✗	54.30	51.61	34.95	50.93	86.55	50.62	44.31	41.22	51.81	6.55	61.65	60.88	79.87	7.28
S-LoRA$_{Pre}$	✗	49.80	58.87	41.16	43.20	90.30	54.32	69.23	41.80	56.08	10.56	60.61	45.79	78.40	13.15
Lifelong-MoE‡	✗	38.55	41.53	21.63	22.50	62.90	24.69	30.15	38.94	35.11	16.00	52.41	0.00	76.39	31.82
CLoRE	✗	**58.35**	66.13	**59.71**	**56.24**	89.50	46.91	57.54	41.12	**59.44**	**2.84**	**66.25**	**74.53**	**81.01**	**0.82**
MoE-LPR‡	✓	55.95	68.35	61.37	48.05	88.10	37.04	48.00	40.46	55.91	3.81	63.05	52.54	80.14	9.51
CLoRE	✓	**56.50**	**70.77**	**63.52**	**58.39**	**90.65**	**64.20**	**59.08**	**41.62**	**63.09**	**0.28**	**66.66**	**73.69**	**81.07**	**0.94**
MTL†	–	56.45	71.57	65.69	71.10	93.20	60.49	68.92	42.05	66.18	–	62.75	48.82	78.89	11.26
Qwen3-8B															
Zero-shot	–	53.65	54.03	18.36	46.82	88.40	54.32	21.23	36.45	46.66	–	73.02	87.26	77.86	–
Seq LoRA	✗	50.65	61.49	23.19	54.37	84.95	66.67	74.46	41.55	57.17	12.18	70.37	82.03	78.24	2.50
EWC	✗	46.05	64.31	34.49	56.92	84.90	**72.84**	**76.31**	**41.71**	59.69	9.00	64.11	79.98	77.53	5.51
O-LoRA	✗	54.30	47.78	16.57	51.36	86.40	59.26	55.69	40.07	51.43	7.50	67.76	81.12	**78.51**	3.58
S-LoRA$_{Pre}$	✗	50.55	48.59	22.41	38.99	80.90	61.73	66.77	41.57	51.44	12.18	70.01	81.12	77.53	3.16
Lifelong-MoE‡	✗	35.90	45.56	23.42	19.39	78.80	53.09	50.15	39.25	43.20	9.77	62.18	50.72	77.80	15.81
CLoRE	✗	**58.50**	**65.52**	**58.68**	**58.28**	**88.00**	59.26	69.23	36.73	**61.77**	**0.79**	**72.72**	**87.11**	**78.51**	**−0.07**
MoE-LPR‡	✓	52.10	65.52	59.71	49.39	**88.55**	**71.60**	51.69	38.42	59.62	5.50	71.62	82.03	**78.13**	2.12
CLoRE	✓	**57.05**	**68.95**	**62.41**	**60.97**	88.10	56.91	**54.00**	**40.68**	**61.31**	**2.26**	**72.29**	**85.97**	77.86	**0.67**
MTL†	–	55.95	69.15	63.70	60.52	94.25	72.84	68.00	41.37	65.72	–	70.62	74.22	78.07	5.08

adapting. TRACE identified the second as the larger loss, especially on math and reasoning, and the right block confirms it: every baseline loses between 7 and 13 points of backbone accuracy on Llama-3.1-8B-Instruct, and MTL, with all of the data, loses 11.26, so a method can be competitive on the eight tasks while having lost much of the model. CLoREloses 0.82, and on Qwen3-8B nothing. MoE-LPR loses 9.51 even with stored data, because its replay covers earlier tasks but not the backbone; in CLoREthe backbone is the domain $\tau = 0$ of §3.1 and is replayed like any other. How its inputs are routed after continual learning is examined in §4.3.

Comparison on Wiki→Code→Conversation. Tab. 2 compares CLoREwith Lifelong-MoE and MoE-LPR on their own architecture, an MoE trained from scratch with no frozen backbone and hence no skip experts. CLoREreaches the highest accuracy on all three corpora, 50.58 on average with forgetting of 0.26, against 49.80 and 0.49 for MoE-LPR, even though MoE-LPR replays original Wiki data and CLoREonly self-generated data of the same token budget; Lifelong-MoE reaches 37.28 with forgetting of 18.31. The routing columns show why. For a fixed set of held-out Wiki inputs we compare the top-k selections of the final model with those of the Wiki-trained checkpoint: retention is the fraction of original selections that remain, leakage the fraction of slots that go to experts added afterwards (§D). Lifelong-MoE, which freezes the Wiki experts and their gating dimensions but nothing else, retains only 34.55% and leaks 62.25%. MoE-LPR retains 86.00% by replaying original data with a routing prior that fixes where each replayed token should go; CLoREretains 90.31% and leaks 2.91% with no such target, from the language-modeling loss on generated replay alone. §4.3 examines where this drift arises.

Table 2: **Continual learning on Wiki→Code→Conv with FLAME-MoE.** Next-token prediction accuracy on the test set of each corpus after the full sequence, its average, and forgetting, following the protocol of Lifelong-MoE. Replay indicates the source of earlier-task data used during training; CLoREuses the same token budget as MoE-LPR. Retention and leakage compare the routing of held-out Wiki inputs in the final model with that of the Wiki-trained checkpoint. Bold: best.

		Token accuracy					Routing on Wiki	
Method	Replay	Wiki	Code	Conv	Average↑	Forgetting↓	Retention↑	Leakage↓
Lifelong-MoE	N/A	32.89	41.64	37.31	37.28	18.31	34.55	62.25
MoE-LPR	original	45.74	65.72	37.95	49.80	0.49	86.00	9.90
CLoRE	self-generated	**46.20**	**67.48**	**38.05**	**50.58**	**0.26**	**90.31**	**2.91**

Table 3: **Drift at the moment of expansion.** Change in routing and layer output for earlier-task inputs when new experts and router rows are added, before any training step, for three initializations of the new rows. Fraction of tokens whose top-k set changes, change in the weights of the originally selected experts, and relative change in layer output; averaged over expansions (§D).

	TRACE			Wiki→Code→Conv		
Initialization	$\mathcal{S}_k$ changed (%)	Δp	$\Delta\hat{h}$ (%)	$\mathcal{S}_k$ changed (%)	Δp	$\Delta\hat{h}$ (%)
Random	66.20	0.399	39.39	100.00	0.812	72.39
Zero	64.97	0.394	38.94	100.00	0.810	72.65
Reservoir (ours)	**0.37**	**0.002**	**1.07**	**1.08**	**0.003**	**0.85**

4.3 Representational Drift and Its Effect

Latent distributions. Fig. 1 shows the hidden states of Wiki inputs before and after Code is learned. With the Wiki experts and their router rows frozen (Fig. 1a), the distribution moves and the overlap drops to 0.84; with CLoRE(Fig. 1b) it stays at 0.99. This is the drift of §3.2 seen in the representations themselves; the routing statistics of Tab. 2 are its trace at the level of expert selection. The two paragraphs below measure its two sources, at the moment of expansion and during training, and what each leaves behind.

Drift at expansion. Tab. 3 isolates the moment of insertion. For each expansion we take the model just before and just after the new experts and router rows are added, with no training step in between, and pass a fixed batch of earlier-task inputs through both. Since $B_j = 0$, the new experts contribute nothing to the output, and any change comes from the router. Random and zero initialization behave alike: on TRACE they change the top-k set of about 65% of tokens and move layer outputs by 39%, and on Wiki→Code→Conv, where eight experts are added at once, every token changes its selection. The reservoir of §3.4 changes about 1% of selections and moves outputs by about 1% on both, because the denominator already accounted for these rows before they existed.

Drift during training. Tab. 4a separates when the router is aligned from how the rows are initialized: both use the reservoir, so both start from the same model after expansion, and only the timing differs. Concurrent alignment ends at 46.29 on Wiki, close to the 46.62 it started from, and 67.77 on Code. Under post-hoc alignment, Wiki accuracy falls to 34.98 while the Code experts are trained, because the router adapts to Code with nothing holding Wiki routing in place; the subsequent alignment on replay recovers it only to 45.61 and costs 0.75 on Code. This is what §3.3 predicted: once the representations of earlier inputs have moved, training the router cannot fully restore them.

4.4 Component Ablations

Router loss. Tab. 4b replaces the language-modeling loss of the router branch with distillation of the previous router, zero-padded to the new expert pool. Distillation matches the routing itself better, fewer changed selections ($\mathcal{S}_k$) and smaller weight changes (Δp), but the layer outputs ($\Delta\hat{h}$) no better, 11.67% against 11.59%, and forgets more, 0.64 against 0.40. Preserving the routing distribution is not the same as preserving the representation, and preserving the representation by the same amount is not the same as preserving the task; only the target objective optimizes the last, which is the argument of §3.3 and the pattern of MoE-LPR in Tab. 2.

Table 4: **Router alignment on Wiki→Code→Conv**, all rows with original-data replay. (a) Token accuracy on Wiki and Code after each training phase of learning Code; Wiki starts at 46.62 after expansion. Post-hoc trains experts and router on Code first, then the router on replay. (b) Language-modeling loss vs. distillation of the previous router in the router branch. Drift columns are those of Tab. 3, but measured after the next task has been trained rather than at expansion (Sec. D). ⋆: full CLoRE; other rows change only the one stated.

(a) Concurrent vs. post-hoc alignment

	Phase	Wiki	Code
Concurrent⋆	N/A	**46.29**	**67.77**
Post-hoc	expert	34.98	67.95
	router	45.61	67.20

(b) Router loss

	AA↑	F↓	$\mathcal{S}_k$ (%)↓	Δp ↓	$\Delta\hat{h}$ (%)↓
LM loss⋆	50.97	0.40	37.49	0.123	11.59
Router KL	50.28	0.64	31.60	0.090	11.67

Table 5: **Ablation on TRACE with Llama-3.1-8B-Instruct.** Average accuracy (AA), forgetting (F), and backbone forgetting (F_0). ⋆ marks the full CLoREwith every component of §3 in place and 10% of the original data of earlier tasks as replay, as in the stored-data group of Tab. 1; every other row changes only the one stated. w/o reservoir initializes the new router rows randomly, which also removes $\mathcal{L}_{\text{margin}}$. In (c), the rows below the dashed line replace original data with self-generated replay, produced with the routing override of §3.5 or with a learned per-task input embedding.

(a) Simultaneous alignment (§3.3)

	AA↑	F↓	F_0 ↓
Concurrent⋆	63.09	0.28	0.94
Post-hoc	59.22	3.00	5.89

(b) Reservoir routing (§3.4)

	AA↑	F↓	F_0 ↓
Reservoir⋆	63.09	0.28	0.94
w/o reservoir	62.79	0.58	2.41

(c) Self-generated replay (§3.5)

	AA↑	F↓	F_0 ↓
Original data⋆	63.09	0.28	0.94
Routing override	59.44	2.84	0.82
Learned prompt	55.94	6.69	2.92

Simultaneous alignment. Tab. 5a shows what the post-hoc schedule of Tab. 4a costs at the end of the eight TRACE tasks: forgetting rises from 0.28 to 3.00, average accuracy falls from 63.09 to 59.22, and backbone forgetting rises from 0.94 to 5.89, the largest changes of any row in Tab. 5. The partial recovery seen on one expansion in §4.3 compounds over seven.

Reservoir routing. Tab. 5b replaces the reservoir with random initialization of the new router rows. Forgetting rises from 0.28 to 0.58 and backbone forgetting from 0.94 to 2.41. Both rows align the router on replay from the first step, so this is what alignment alone does not undo: most of the displacement that random rows cause at expansion (Tab. 3) is repaired during training, and the remainder is what the reservoir removes.

Self-generated replay. Tab. 5c replaces the 10% original data with replay generated by the model itself. With nothing stored, the routing override of §3.5 keeps CLoREat 59.44 with forgetting of 2.84, 3.65 points below original data and, as Tab. 1 shows, above every method that stores none. The natural alternative, a learned task-specific BoS embedding as in soft prompting, drops to 55.94 with forgetting of 6.69. That the loss shows up as forgetting says the embedding was not under-trained but outlived: it was fitted to representations that then drift, and it stops eliciting the task. The routing override learns nothing: selecting a task's frozen experts at the first token is enough.

5 Conclusion

Continual learning of an LLM has to add domains without losing the earlier ones or the model itself. Isolating each domain in its own parameters is not enough: the router rows added for a new domain move the routing of earlier inputs, and their representations drift. CLoREholds the router in place with the target objective on self-generated replay and inserts new rows from a reservoir, so that neither training nor expansion moves them. Across two benchmarks it forgets least without storing any data, exceeds MoE methods that replay original data, and leaves the backbone's general capability nearly intact where every baseline loses it; expansion moves under 1% of earlier routing decisions, against two thirds with random initialization.

USE OF GENERATIVE AI

In this work, we used generative AI tools to assist with the translation of non-English dataset (Sec. E.2), drafting of parts of the manuscript and debugging LaTeX code. We reviewed all AI-assisted work. All AI-drafted text was revised by the authors, and every reference suggested by the tools was verified against the original publication. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *European conference on computer vision*, pp. 144–161. Springer, 2018.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

Sungmin Cha, Hsiang Hsu, Taebaek Hwang, Flavio Calmon, and Taesup Moon. {CPR}: Classifier-projection regularization for continual learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=F2v4aqEL6ze.

Sungmin Cha, Jihwan Kwak, Dongsub Shim, Hyunwoo Kim, Moontae Lee, Honglak Lee, and Taesup Moon. Towards more diverse evaluation of class incremental learning: Representation learning perspective. In Vincenzo Lomonaco, Stefano Melacci, Tinne Tuytelaars, Sarath Chandar, and Razvan Pascanu (eds.), *Proceedings of The 3rd Conference on Lifelong Learning Agents*, volume 274 of *Proceedings of Machine Learning Research*, pp. 86–101. PMLR, 29 Jul–01 Aug 2025. URL https://proceedings.mlr.press/v274/cha25a.html.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem, 2019. URL https://arxiv.org/abs/1812.00420.

Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer. *arXiv preprint arXiv:1511.05641*, 2015.

Wuyang Chen, Yanqi Zhou, Nan Du, Yanping Huang, James Laudon, Zhifeng Chen, and Claire Cui. Lifelong language pretraining with distribution-specialized experts. In *International Conference on Machine Learning*, pp. 5383–5395. PMLR, 2023.

Dongkyu Cho, Taesup Moon, Rumi Chunara, Kyunghyun Cho, and Sungmin Cha. Forget forgetting: Continual learning in a world of abundant memory. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=fvL8IIEPxG.

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, 2021. URL https://arxiv.org/abs/2110.14168.

Nan Du, Yanping Huang, Andrew M. Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, Barret Zoph, Liam Fedus, Maarten Bosma, Zongwei Zhou, Tao Wang, Yu Emma Wang, Kellie Webster, Marie Pellat, Kevin Robinson, Kathleen Meier-Hellstern, Toju Duke, Lucas Dixon, Kun Zhang, Quoc V Le, Yonghui Wu, Zhifeng Chen, and Claire Cui. Glam: Efficient scaling of language models with mixture-of-experts, 2022. URL https://arxiv.org/abs/2112.06905.

Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. In Silvia Chiappa and Roberto Calandra (eds.), *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 3762–3773. PMLR, 26–28 Aug 2020. URL https://proceedings.mlr.press/v108/farajtabar20a.html.

Robert M. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, 1999.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Zirui Guo, Quan Cheng, Da-Wei Zhou, and Lijun Zhang. Stable routing for mixture-of-experts in class-incremental learning. *arXiv preprint arXiv:2605.17571*, 2026.

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

Zhiyan Hou, Haiyun Guo, Haokai Ma, Yandu Sun, Yonghui Yang, and Jinqiao Wang. Pass-moe: Mitigating misaligned co-drift among router and experts via pathway activation subspaces for continual learning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 31959–31972, 2026.

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. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Jianheng Huang, Leyang Cui, Ante Wang, Chengyi Yang, Xinting Liao, Linfeng Song, Junfeng Yao, and Jinsong Su. Mitigating catastrophic forgetting in large language models with self-synthesized rehearsal. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1416–1428, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.77. URL https://aclanthology.org/2024.acl-long.77/.

Hao Kang, Zichun Yu, and Chenyan Xiong. Flame-moe: A transparent end-to-end research platform for mixture-of-experts language models, 2025. URL https://arxiv.org/abs/2505.20225.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, March 2017. ISSN 1091-6490. doi: 10.1073/pnas.1611835114. URL http://dx.doi.org/10.1073/pnas.1611835114.

Yan-Shuo Liang, Jia-Rui Chen, and Wu-Jun Li. Gated integration of low-rank adaptation for continual learning of large language models. *Advances in Neural Information Processing Systems*, 38:68659–68689, 2026.

Pierre Lison and Jörg Tiedemann. OpenSubtitles2016: Extracting large parallel corpora from movie and TV subtitles. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pp. 923–929, Portorož, Slovenia, 2016. European Language Resources Association (ELRA). URL https://aclanthology.org/L16-1147/.

Pierre Lison, Jörg Tiedemann, and Milen Kouylekov. OpenSubtitles2018: Statistical rescoring of sentence alignments in large, noisy parallel corpora. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, pp. 1742–1748, Miyazaki, Japan, 2018. European Language Resources Association (ELRA). URL https://aclanthology.org/L18-1275/.

David Lopez-Paz and Marc' Aurelio Ranzato. Gradient episodic memory for continual learning. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/f87522788a2be2d171666752f97ddebb-Paper.pdf.

Haodong Lu, Chongyang Zhao, Minhui Xue, Lina Yao, Kristen Moore, and Dong Gong. Little by little: Continual learning via incremental mixture of rank-1 associative memory experts, 2026. URL https://arxiv.org/abs/2506.21035.

Yun Luo, Zhen Yang, Xuefeng Bai, Fandong Meng, Jie Zhou, and Yue Zhang. Investigating forgetting in pre-trained representations through continual learning. *arXiv preprint arXiv:2305.05968*, 2023.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *IEEE Transactions on Audio, Speech and Language Processing*, 2025.

Martial Mermillod, Aurélia Bugaiska, and Patrick Bonin. The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects, 2013.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

Haizhou Shi, Zihao Xu, Hengyi Wang, Weiyi Qin, Wenyuan Wang, Yibin Wang, Zifeng Wang, Sayna Ebrahimi, and Hao Wang. Continual learning of large language models: A comprehensive survey. *arXiv preprint arXiv:2404.16789*, 2024.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, volume 30, pp. 2990–2999, 2017. URL https://arxiv.org/abs/1705.08690.

Fan-Keng Sun, Cheng-Hao Ho, and Hung-Yi Lee. LAMOL: LAnguage MOdeling for lifelong language learning. In *International Conference on Learning Representations*, 2020. URL https://arxiv.org/abs/1909.03329.

Lewis Tunstall, Leandro von Werra, and Thomas Wolf. *Natural Language Processing with Transformers*. O'Reilly Media, 2022.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuanjing Huang. Orthogonal subspace learning for language model continual learning, 2023a. URL https://arxiv.org/abs/2310.14152.

Xiao Wang, Yuansen Zhang, Tianze Chen, Songyang Gao, Senjie Jin, Xianjun Yang, Zhiheng Xi, Rui Zheng, Yicheng Zou, Tao Gui, Qi Zhang, and Xuanjing Huang. Trace: A comprehensive benchmark for continual learning in large language models, 2023b. URL https://arxiv.org/abs/2310.06762.

Zitao Wang, Xinyi Wang, and Wei Hu. Mixture of lora experts for continual information extraction with llms. In *EMNLP (Findings)*, pp. 13324–13339, 2025.

Wikimedia Foundation. Wikimedia downloads. https://dumps.wikimedia.org, 2023. Accessed via https://huggingface.co/datasets/wikimedia/wikipedia, config 20231101.en.

Zhangchen Xu, Fengqing Jiang, Luyao Niu, Yuntian Deng, Radha Poovendran, Yejin Choi, and Bill Yuchen Lin. Magpie: Alignment data synthesis from scratch by prompting aligned LLMs with nothing. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2406.08464.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Lina Yang, Yusheng Liao, Yanfeng Wang, and Yu Wang. SLoRA: Balancing plasticity and forgetting in large language models for continual learning. In Maria Liakata, Viviane P. Moreira,

Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5437–5454, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.247. URL https://aclanthology.org/2026.acl-long.247/.

Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*, 2017.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 3987–3995. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/zenke17a.html.

Chongyang Zhao, Mingsong Li, Haodong Lu, and Dong Gong. On token's dilemma: Dynamic moe with drift-aware token assignment for continual learning of large vision language models, 2026. URL https://arxiv.org/abs/2603.27481.

Hao Zhou, Zhijun Wang, Shujian Huang, Xin Huang, Xue Han, Junlan Feng, Chao Deng, Weihua Luo, and Jiajun Chen. Moe-lpr: Multilingual extension of large language models through mixture-of-experts with language priors routing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 26092–26100, 2025.

A DATASET

Wiki, Code, Conversation The analysis experiments use three domains learned in sequence, all obtained from Hugging Face. Wiki is drawn from English Wikipedia (Wikimedia Foundation, 2023) (`wikimedia/wikipedia`, config `20231101.en`), from which we randomly sample 3.20M of the 6.41M articles. Code is drawn from CodeParrot-clean (Tunstall et al., 2022) (`codeparrot/codeparrot-clean`), a corpus of Python files collected from GitHub. Conv consists of the English side of OpenSubtitles-v2018 (Lison & Tiedemann, 2016; Lison et al., 2018) (`Helsinki-NLP/open_subtitles`). Since this corpus is distributed as language pairs, English sentences repeated across pairs are removed. All corpora are exact-deduplicated, and their training sizes are listed in Tab. 6.

Table 6: Training splits of the Wiki–Code–Conv corpora, after exact deduplication.

Task	Source	Tokens	Documents
Wiki	Wikipedia (20231101.en, subset)	2.12B	2,922,937
Code	CodeParrot-clean (subset)	2.12B	703,256
Conv	OpenSubtitles-v2018 (English)	1.40B	828,534

Lifelong-MoE (Chen et al., 2023) pretrains on three distributions built from the GLaM corpus (Du et al., 2022): English webpages with Wikipedia, a non-English (i18n) corpus, and conversations from public-domain social media. None of the three is publicly released, so we reconstruct the setting with public corpora. We keep the property the original sequence was designed around—large gaps between consecutive distributions, so that forgetting is observable rather than masked by overlap—and substitute code for the non-English distribution, which places the second task outside the domain of the first while keeping every corpus in English.

The analysis experiments use three domains learned in sequence, all obtained from Hugging Face. Wiki is drawn from English Wikipedia (Wikimedia Foundation, 2023) (`wikimedia/wikipedia`, config `20231101.en`), from which we randomly sample 3.20M of the 6.41M articles. Code is drawn from CodeParrot-clean (Tunstall et al., 2022) (`codeparrot/codeparrot-clean`), a corpus of Python files collected from GitHub. Conv consists of the English side of OpenSubtitles-v2018 (Lison & Tiedemann, 2016; Lison et al., 2018) (`Helsinki-NLP/open_subtitles`). Since this corpus is distributed as language pairs, English sentences repeated across pairs are removed. All corpora are exact-deduplicated, and their training sizes are listed in Tab. 6.

For Wiki and Code, we randomly sample whole documents into a 2.33B-token pool (drawn from 8.11B tokens for Code) and split it into 2.12B training tokens and a 0.20B-token held-out split. Each task is trained on about 2.12B tokens, which is one epoch for Wiki and Code and about 1.51 epochs for Conv.

TRACE. The eight tasks, in the order used throughout, are C-STANCE (stance detection, Chinese), FOMC (hawkish/dovish classification of Federal Reserve statements), MeetingBank (meeting summarization), Py150 (Python code completion), ScienceQA (science question answering), NumGLUE-cm and NumGLUE-ds (arithmetic reasoning), and 20Minuten (German news summarization). Each is evaluated with its own metric: accuracy for C-STANCE, FOMC, ScienceQA, and the two NumGLUE tasks, ROUGE-L for MeetingBank, edit similarity for Py150, and SARI for 20Minuten; per-task scores are averaged into average accuracy. Training and test splits follow Wang et al. (2023b), with 5,000 training examples per task. The stored-data setting replays 10% of each earlier task's training set (500 examples), sampled uniformly once.

Backbone evaluation. MMLU (0-shot, accuracy), GSM8K (5-shot, exact match on the extracted final answer), and PIQA (0-shot, length-normalized accuracy) are evaluated with lm-evaluation-harness 0.4.8 using its default prompts, without a chat template, on the final checkpoint of each method with the same protocol as the untouched backbone.

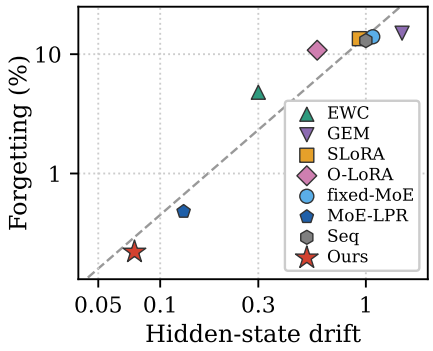


Figure 3: **Hidden-state drift and forgetting on Wiki→Code→Conv.** Each point is one method, trained in the controlled setting of Tab. 2. Drift is measured on the hidden states of held-out earlier-task inputs, and forgetting is defined as in Sec. D. Both axes are logarithmic. Methods that drift less also forget less, and CLoRE has the smallest drift and the smallest forgetting.

B Correlation between hidden-state drift and forgetting

To examine whether hidden-state drift accounts for forgetting across methods, free of any bias from a pretrained backbone, we move every method of Tab. 1 to the controlled setting of Tab. 2, in which the model is trained from scratch on Wiki→Code→Conv.

Implementation details. Methods defined on a mixture of experts, namely CLoRE, Lifelong-MoE, and MoE-LPR, are implemented on the MoE architecture. Methods that attach LoRA modules to a backbone, namely Seq LoRA, EWC, S-LoRA, and O-LoRA, etc have no pretrained backbone in this setting, so we first train the base model on Wiki and then attach LoRA modules to learn Code and Conv in sequence. All other training settings follow Tab. 2.

Results. As shown in Fig. 3, hidden-state drift and forgetting are closely related across all methods, with a Pearson correlation of 0.96. Methods whose earlier-task representations drift less also forget less, and CLoRE has both the smallest drift and the smallest forgetting. This supports the premise of Sec. 3.2 that forgetting in expansion-based methods can be traced to the drift of earlier-domain representations, even when the parameters encoding them are frozen.

C FLAME MoE model structure

Table 7: Configuration of the FLAME-MoE model used in Tab. 2.

Hyperparameter	Value
Attention heads	16
Expert FFN hidden size	352
Experts added per task (m)	8
Experts after all tasks	24
Routed experts per token (k)	4
Shared experts	none
Router score function	softmax before top-k, fp32
Normalization	RMSNorm ($\epsilon = 10^{-6}$)
Position embedding	RoPE
Activation	SwiGLU
Dropout	0.0
Initialization std	0.02
Load-balancing loss coefficient	0.01
Router z-loss coefficient	0.001
Tokenizer	Pythia (GPT-NeoX)
Framework	Megatron-LM

Base architecture. The experiments of Tab. 2 build on FLAME-MoE (Kang et al., 2025), a decoder-only transformer with SwiGLU feed-forward layers, RMSNorm, rotary position embeddings, and untied input and output embeddings, in which feed-forward sublayers are replaced by mixture-of-experts layers. Each MoE layer consists of feed-forward experts E_i and a linear router r that scores every expert for the input token representation x. The top-k experts by score are selected, and their outputs are combined with weights given by the softmax of the router scores over all experts,

$$\mathrm{MoE}(x) = \sum_{i \in \mathrm{TopK}(r(x))} \mathrm{softmax}\big(r(x)\big)_i \, E_i(x). \tag{6}$$

The softmax is applied before top-k selection and the selected weights are not renormalized, so adding experts enlarges the denominator exactly as in Eq. (1), and the analysis of §3.4 applies without modification. In its original configuration, each MoE layer also contains shared experts that process every token; we remove them, so that every expert is routed and can be assigned to a single task.

Configuration for continual learning. We train the model from scratch and replace the fixed expert pool with the expanding pool of §3.6: each task adds $m = 8$ experts, and the router selects the top $k = 4$, so the model holds 24 experts after the three tasks. When a new task begins, the experts of earlier tasks are frozen, and the router is trained as described in §3.3. Since there is no frozen backbone, the model has no skip experts. Each expert has a feed-forward hidden size of 352, so the active feed-forward width per token is 1,408 regardless of how many experts have been added. Tab. 7 lists the configuration.

D Evaluation Metrics

Let $a_{t,j}$ denote the evaluation score on task j after completing task t, where $j \le t$. Scores use a consistent scale within each experiment; task-specific metrics are specified in the evaluation protocol. We report average accuracy (AA), forgetting (F), and learning accuracy (LA):

$$\mathrm{AA}_t = \frac{1}{t} \sum_{j=1}^{t} a_{t,j}, \tag{7}$$

$$F_t = \frac{1}{t-1} \sum_{j=1}^{t-1} \left(\max_{j \le s < t} a_{s,j} - a_{t,j} \right), \tag{8}$$

$$\mathrm{LA}_t = \frac{1}{t} \sum_{j=1}^{t} a_{j,j}. \tag{9}$$

Unless otherwise stated, we report these metrics after the final task, at $t = T$.

AA measures performance across all learned tasks. F measures the decline on previous tasks relative to their best earlier scores. We retain signed forgetting: a task contributes a negative value when its current score exceeds all earlier scores, without clipping at zero. Forgetting is undefined for $t = 1$ and is not reported.

LA measures performance immediately after each task is learned, separating acquisition from subsequent forgetting. We also inspect individual $a_{j,j}$ values when comparing plasticity, since final average performance alone cannot distinguish weak acquisition from later forgetting.

E Implementation Details

E.1 Layer Output at Expansion

Suppressing layer and projection indices, expert j contributes

$$\Delta_j(u) = \frac{\alpha}{r} B_j A_j u, \qquad \Delta_{\mathrm{skip}}(u) \equiv 0. \tag{10}$$

An expert drawn from the reservoir starts with $B_j = \mathbf{0}$ (§3.4), so $\Delta_j(u) = 0$ at expansion. Let $g_j(h) = p_j(h)$ if $j \in \mathcal{S}_k(h)$ and 0 otherwise. For a frozen backbone projection W, the output after expansion is

$$y_{\text{after}} - y_{\text{before}} = \sum_{j \in \mathcal{I}_{<t}} \left(g_j^{\text{after}}(h) - g_j^{\text{before}}(h)\right) \Delta_j(u), \tag{11}$$

since the new experts contribute nothing whatever weight they receive. The layer output therefore changes only through the gates of earlier experts: an earlier expert leaving the top-k sets its gate to zero, and a larger denominator shrinks the gates of those that remain. By Sec. H, reservoir expansion leaves $\mathcal{S}_k(h)$ unchanged with probability at least $1 - \eta$ under the margin condition and preserves the denominator in expectation, so both terms are small.

Knowledge-distilled initialization for MoE models. When FFNs themselves consist of experts, bypassing expert computation does not preserve the pre-expansion FFN transformation. We instead initialize the added capacity by matching the pre-expansion model's output distribution.

Let f_{t-1} denote the frozen pre-expansion model and f the expanded model. Writing $\mathcal{M}_t = \bigcup_{j<t} \mathcal{G}_j$ for generated previous-task data, we optimize

$$\min_{\theta_t, \phi_{\text{new}}} \mathcal{L}_{\text{init}} = \mathbb{E}_{(x,i) \sim \mathcal{V}(\mathcal{M}_t)} \left[\text{KL}\big(p_{f_{t-1}}(\cdot \mid x_{<i}) \,\big\|\, p_f(\cdot \mid x_{<i})\big) \right], \tag{12}$$

where θ_t and ϕ_{new} denote new expert parameters and new router rows, respectively, and $\mathcal{V}(\mathcal{M}_t)$ contains valid prediction positions selected by the loss mask.

Both distributions are computed from the final LM-head logits over the vocabulary with temperature one. Generated sequences supply the contexts, while the teacher supplies distributions over possible continuations rather than only the generated next-token targets. Only new experts and new router rows are optimized; previous experts and router rows remain fixed. We use the KL objective alone, without incoming-task data, to approximately preserve the pre-expansion output distribution.

E.2 Generated data sample

C-STANCE

Question: 判以下文本指定象的度,一:A.支持,B.反,C.中立 出A,B或者C 。
文本:然我不看球,但我可以的是,中男足有希望,是的,希望他可以一比,便一比,就算是和台的也行 。
象:
中男足
度:

(Translation) Determine the attitude of the following text toward the given target, choosing one: A. support, B. oppose, C. neutral. Output A, B, or C.
Text: Although I don't like watching soccer, I can say this: the Chinese men's national team has hope, truly. I hope they can win a match, any match, even one against Taiwan.
Target:
Chinese men's national team
Attitude:

Answer: A

FOMC

Question: What is the monetary policy stance for the following text? A. dovish, B. hawkish, C. neutral. Choose one from A, B and C.
Text: Moreover, most members saw considerable downside risks to the economic outlook and judged that a rate reduction at this meeting would provide valuable additional insurance against an unexpectedly severe weakening in economic activity.
Stance:

Answer: A

Py150

Question:
<s> from distutils. core import setup, Extension <EOL> setup (name = '<STR_LIT>', <EOL> version = '<STR_LIT>', <EOL> description = '<STR_LIT>', <EOL> url = '<STR_LIT>', <EOL>

Answer:
author = '<STR_LIT>',

MeetingBank

Question:
Write a summary of the following meeting transcripts.
Meeting transcripts: Item 18 Report from Police Recommendation to authorize City Manager to execute a memorandum of understanding with the Los Angeles County Metropolitan Transportation Authority to accept and expend grant funding in the amount of $2,000,000 for the California Law Enforcement Telecommunications System CAD Project.Mr. Chief, you have any any comment?Just that the U.S. Department of Justice provided the initial funding for this and it was a one time grant. And this grant from Metro will be a recurring source of funding f [...]

Answer:
Recommendation to authorize City Manager to execute a Memorandum of Understanding with the Los Angeles County Metropolitan Transportation Authority to accept and expend grant funding in the amount of $2,000,000 for the California Law Enforcement Telecommunications System (CLETS) CAD Project. (Citywide)

ScienceQA

Question:
Choose an answer for the following question and give your reasons.
Question: What do these two changes have in common?
a piece of metal rusting
a copper penny tarnishing
Choices:
A. Both are chemical changes.
B. Both are only physical changes.
C. Both are caused by heating.
D. Both are caused by cooling.
Answer:

Answer:
A
Chemical changes involve the formation of one or more new substances. When a piece of metal rusts, the metal combines with oxygen from the air to form iron oxide. When a copper penny tarnishes, the copper reacts with oxygen in the air to form copper oxide. In both cases, the original materials change into new materials, so bot [...]

NumGLUE-cm

Question:
Solve the following math problem.
Question: Ava decides to go to a nearby animal farm that has a total of 9 animals. He is standing in a queue near the entry gate, from the gate he could only see legs of the animals in the farm and he counted the total number of legs to be 26. He already knows that there are only chickens and buffalos in this farm. From this information find the number of chickens.
Answer:

Answer:
5

NumGLUE-ds

Question:

```
Solve the following math problem.
Question: Find the mass percentage of C in H2CO3
Answer:
```

Answer:

```
19.35
```

We show replay samples generated by the procedure of §3.5 with Llama-3.1-8B-Instruct, taken from the pool built before the eighth task by the checkpoint after the first seven TRACE tasks; 20Minuten, the last task, is never replayed. Each question is generated from the prefix, with the first token routed through the experts of the indicated task (Eq. (5)); the answer is generated with free routing over the full pool. The generated questions reproduce each task's instruction verbatim. We show one sample per task whose generated answer is correct; [...] marks truncation for space.

F ALGORITHM

Algorithm 1 Continual learning with CLoRE

Input: Frozen backbone f; domains $\mathcal{D}_1, \ldots, \mathcal{D}_T$; experts per domain m; top-k; replay size N per domain; noise scale σ; margin weight λ

Output: LoRA pool $\mathcal{E}^{(T)}_{\text{LoRA}}$ and router W_r

1: Initialize k skip experts $\mathcal{E}_{\text{skip}}$, $\mathcal{E}^{(0)}_{\text{LoRA}} \leftarrow \emptyset$, reservoir row w_{res}, count $n \leftarrow m$
2: **for** $t = 1, \ldots, T$ **do**
// Self-generated replay (§3.5)
3: $\mathcal{R}_{<t} \leftarrow \emptyset$
4: **for** $\tau = 0, \ldots, t-1$ **do**
5: Generate N question–answer pairs $\mathcal{R}_\tau$, routing the first question token over $\mathcal{E}_\tau$ only (Eq. (5); no override for $\tau = 0$)
6: $\mathcal{R}_{<t} \leftarrow \mathcal{R}_{<t} \cup \mathcal{R}_\tau$
7: **end for**
// Expansion from the reservoir (§3.4)
8: **for** $j = 1, \ldots, m$ **do**
9: $w_j \leftarrow w_{\text{res}} + \epsilon_j,\ \epsilon_j \sim \mathcal{N}(\mathbf{0}, \sigma^2 I);\ A_j$ random; $B_j \leftarrow \mathbf{0}$
10: **end for**
11: $\mathcal{E}^{(t)}_{\text{LoRA}} \leftarrow \mathcal{E}^{(t-1)}_{\text{LoRA}} \cup \mathcal{E}_t;\ n \leftarrow 0$
// Two-branch training (§3.3)
12: **for** each training step **do**
13: Sample batches $b \sim \mathcal{D}_t$ and $b' \sim \mathcal{D}_t \cup \mathcal{R}_{<t}$
14: **Expert branch:** update $\mathcal{E}_t$ and the LoRA rows of W_r with $\mathcal{L}_{\text{expert}}(b)$, routing over $\mathcal{E}^{(t)}_{\text{LoRA}}$ only
15: **Router branch:** update W_r, including w_{res}, with $\mathcal{L}_{\text{router}}(b') + \lambda \mathcal{L}_{\text{margin}}(b')$, routing over $\mathcal{E}^{(t)}_{\text{LoRA}} \cup \mathcal{E}_{\text{skip}}$ with the reservoir in the denominator
16: Increase n toward m ▷ restored by the end of training
17: **end for**
18: Freeze $\mathcal{E}_t$; discard $\mathcal{R}_{<t}$
19: **end for**

Algorithm 1 summarizes the procedure of §3.6. Each domain passes through three phases: the current model first generates replay for the backbone and all earlier domains (lines 3–7), the new experts are then drawn from the reservoir so that earlier-domain outputs are unchanged in expectation (lines 8–11), and the two branches are trained together on every step (lines 12–17). Only the new experts are trained in the expert branch, while the router branch updates the whole router, including the rows of earlier experts and the reservoir. At inference, no domain identity is given; the router selects over all experts and the skip experts, with the reservoir in the denominator.

G Details of the drift analysis

Latent distributions (Fig. 2). For each panel we sample 1,200 tokens from held-out Wiki inputs, take their hidden states at layer 5 from the checkpoint right after Wiki was learned and from the checkpoint after Code was learned, and project both sets onto the top two principal components fitted on the two sets pooled. Each projected set is smoothed separately by kernel density estimation with a two-dimensional Gaussian kernel of full covariance and Scott's bandwidth, $n^{-1/(d+4)} = 1200^{-1/6} \approx 0.307$ times the sample covariance of the points. The two densities are evaluated on a common 200×200 grid covering the 0.5–99.5 percentile range of the pooled points with a 12% margin on each side, normalized to sum to one, and the overlap is $\sum \min(p_{\text{before}}, p_{\text{after}})$ over the grid, so that identical distributions score 1. The same grid is used for both panels.

Drift at expansion (Tab. 3). For expansion t we compare the model after training on D_{t-1} with the same model right after the experts and router rows for D_t are added, on a fixed set of inputs from $D_1, \ldots, D_{t-1}$: 1,638,400 tokens on Wiki→Code→Conv, and on TRACE the first 32 replay sequences of each earlier task (32–224 sequences, 4.4K–61K tokens per expansion; padding and the guarded chat header are excluded). At every MoE layer and token we record whether the top-k set $\mathcal{S}_k(x)$ differs between the two models; $\|\hat{h}^{\text{after}} - \hat{h}^{\text{before}}\|/\|\hat{h}^{\text{before}}\|$, the relative change in the layer output; and $|p_i^{\text{after}}(x) - p_i^{\text{before}}(x)|$ for each $i \in \mathcal{S}_k^{\text{before}}(x)$, the change in the weight applied to an expert selected before expansion, with $p_i^{\text{after}}(x) = 0$ if i leaves the top-k. Each is averaged over tokens, layers, and expansions $t = 2, \ldots, 8$ on TRACE and $t = 2, 3$ on Wiki→Code→Conv. Random initialization draws the new rows from the default initialization of a linear layer (Kaiming uniform), $\mathcal{U}(-1/\sqrt{d},\, 1/\sqrt{d})$ with d the hidden size; zero sets them to 0; reservoir sets them to $w_{\text{res}} + \epsilon_j$ as in §3.4. For tokens whose top-k set is unchanged, the weight change is exactly $p_i^{\text{before}}(x)\,|Z^{\text{after}}(x) - Z^{\text{before}}(x)|/Z^{\text{after}}(x)$, where Z is the softmax denominator; for zero initialization $Z^{\text{after}} - Z^{\text{before}} = m$ up to the change in the reservoir term. We use this identity to check the measurement.

G.1 Routing of Backbone Inputs

CLoREtreats the backbone as the domain $\tau = 0$ of §3.1: its inputs are replayed like those of any task. Tab. 8 reports the measurement summarized in §4.3, which examines how the router handles these inputs as tasks are added. We fix 500 inputs from each of MMLU, GSM8K, and PIQA, formatted as in evaluation, and pass them through the checkpoint saved after each of the eight TRACE tasks on Llama-3.1-8B-Instruct. At every MoE layer and token we record whether the top-k selection is a skip expert, and report the fraction of routing slots so assigned, averaged over tokens and layers. The same inputs are used at every checkpoint, so differences across columns reflect only the router. No training takes place on these inputs; the router sees the backbone domain only through the self-generated replay of §3.5. At the same checkpoints we also report accuracy on the full benchmarks.

Tab. 8 shows that the skip fraction does not stay fixed: over all inputs it falls from 61.4% after the first task to 29.3% after the last, so by the end most routing slots of backbone inputs go to added experts. Backbone accuracy nonetheless holds at every checkpoint, between 73.4 and 75.1 on average against 74.8 for Zero-shot, while the baselines lose 7.28 to 31.82 points by the end (Tab. 1). The router thus preserves the backbone not by keeping its inputs on the skip experts, but by mixing in added experts in a way that keeps the backbone's predictions. This follows from the objective: the router branch is trained with the language-modeling loss on replay of $\tau = 0$ rather than toward a fixed routing target, so it is free to use any expert as long as those predictions are kept. GSM8K shows this most clearly: its skip fraction is the lowest throughout and falls to 16.8%, yet its accuracy ranges from 72.2 to 76.4 against 75.2 for Zero-shot.

Retention and leakage (Tab. 2). Let $\mathcal{X}$ be a fixed set of 1,638,400 held-out Wiki tokens. For a token x at a given MoE layer, let $\mathcal{S}_k^0(x)$ be its top-k set under the checkpoint right after Wiki was learned, and $\mathcal{S}_k^T(x)$ its top-k set under the final model. Let $\mathcal{E}_{>1}$ be the experts added after Wiki.

Then

$$\text{Retention} = \frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} \frac{|\mathcal{S}_k^0(x) \cap \mathcal{S}_k^T(x)|}{k}, \tag{13}$$

$$\text{Leakage} = \frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} \frac{|\mathcal{S}_k^T(x) \cap \mathcal{E}_{>1}|}{k}, \tag{14}$$

both computed at each MoE layer and averaged uniformly over layers. Retention counts, out of the k experts a token was routed to when Wiki was learned, how many it is still routed to; leakage counts how many of its k slots now go to experts that did not exist then. The two need not sum to one, since a slot can move to another Wiki expert. We also report the fraction of tokens whose top-k set is exactly preserved, $\frac{1}{|\mathcal{X}|} \sum_x \mathbb{1}[\mathcal{S}_k^0(x) = \mathcal{S}_k^T(x)]$: Lifelong-MoE 0.21, MoE-LPR 53.98, CLoRE 65.36. Under CLoRE the final model has the same expert set as the Wiki checkpoint plus $\mathcal{E}_{>1}$, so the comparison is well defined; for Lifelong-MoE and MoE-LPR we identify experts by position, which their expansion preserves.

Latent distributions (Fig. 1). For each panel we sample 1,200 tokens from held-out Wiki inputs, take their hidden states at layer 5 from the checkpoint right after Wiki was learned and from the checkpoint after Code was learned, and project both sets onto the top two principal components fitted on the two sets pooled. Each projected set is smoothed separately by kernel density estimation with a two-dimensional Gaussian kernel of full covariance and Scott's bandwidth, $n^{-1/(d+4)} = 1200^{-1/6} \approx 0.307$ times the sample covariance of the points. The two densities are evaluated on a common 200×200 grid covering the 0.5–99.5 percentile range of the pooled points with a 12% margin on each side, normalized to sum to one, and the overlap is $\sum \min(p_{\text{before}}, p_{\text{after}})$ over the grid, so that identical distributions score 1. The same grid is used for both panels.

Table 8: **Routing of backbone inputs across the TRACE sequence.** Top: fraction of routing slots (%) assigned to the skip experts for a fixed set of inputs from each benchmark; bottom: accuracy on the full benchmark (MMLU 0-shot, GSM8K 5-shot strict match, PIQA acc_norm), at the checkpoint after each task, on Llama-3.1-8B-Instruct with CLoRE.

	After task								
Inputs	1	2	3	4	5	6	7	8	Mean
Skip routing (%)									
MMLU	70.7	58.0	52.7	53.1	49.3	42.3	37.1	35.2	49.8
GSM8K	45.3	42.0	32.9	35.4	32.3	23.5	18.1	16.8	30.8
PIQA	68.3	61.6	55.2	53.9	50.9	44.5	38.0	36.0	51.0
All	61.4	53.9	46.9	47.5	44.2	36.8	31.1	29.3	43.9
Score (%)									
MMLU	67.8	68.0	67.9	68.0	66.9	67.1	66.4	66.3	67.3
GSM8K	74.8	75.6	76.4	76.4	73.0	72.2	72.8	74.5	74.5
PIQA	80.8	80.9	80.9	81.0	80.9	80.9	81.0	81.0	80.9
Avg.	74.5	74.8	75.1	75.1	73.6	73.4	73.4	73.9	74.2

Drift over training (Tab. 4b). Retention, leakage, and the three drift measures of Tab. 3 are here computed between checkpoints rather than at expansion: for Wiki inputs between the checkpoints after Wiki and after Code, and for Code inputs between those after Code and after Conv, each separately and the two averaged. They therefore include what training on the next task does to earlier-task routing, and are not comparable with the Wiki-only, end-of-sequence values of Tab. 2.

H Properties of Reservoir Expansion

Let $Z_{\text{before}}(x)$ be the denominator of Eq. (3) with count n, and $Z_{\text{after}}(x)$ the denominator after m experts are drawn and the count is reduced to $n - m$.

Denominator. For $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$, $\mathbb{E}[e^{(w_{\text{res}}+\epsilon)x}] = e^{z_{\text{res}}(x)+\sigma^2\|x\|^2/2}$, which is the factor carried by the reservoir term. The m drawn rows contribute $\sum_{j=1}^{m} e^{z_{\text{res}}(x)+\epsilon_j x}$, with expectation

$m\, e^{z_{\text{res}}(x)+\sigma^2\|x\|^2/2}$, exactly the mass removed from the reservoir, so $\mathbb{E}_\epsilon[Z_{\text{after}}(x)] = Z_{\text{before}}(x)$ for every x. The deviation is the sampling error of an m-term mean of i.i.d. log-normal terms and is of order $\sigma\|x\|/\sqrt{m}$ relative to the reservoir's share of $Z(x)$.

Selected set. Each drawn logit is $z_{\text{res}}(x) + \epsilon_j x$ with $\epsilon_j x \sim \mathcal{N}(0, \sigma^2\|x\|^2)$. By the Gaussian tail bound $\Pr[\epsilon x > s] \le e^{-s^2/(2\sigma^2\|x\|^2)}$ and a union bound over the m draws, the largest drawn logit exceeds $z_{\text{res}}(x) + \sigma\|x\|\sqrt{2\ln(m/\eta)}$ with probability at most η. Hence if $z_{\text{res}}(x) + \delta(x) \le \bar{z}_{(k)}(x)$ with $\delta(x) = \sigma\|x\|\sqrt{2\ln(m/\eta)}$, no new expert enters the top-k with probability at least $1-\eta$ and $\mathcal{S}_k(x)$ is unchanged. $\mathcal{L}_{\text{margin}}$ is the hinge form of this condition.

Count and restoration. The count enters only through the product $n\, e^{z_{\text{res}}(x)}$, so a different n is compensated by a constant shift of z_{res} once w_{res} is learned; $n = m$ is the smallest count that covers one expansion. After expansion the count is 0 and is increased linearly to m over training on the new domain. The denominator grows during this restoration, but the router branch is active throughout and absorbs it as ordinary training-time drift.

Where the reservoir acts. The reservoir term is present in the router branch, where $\mathcal{L}_{\text{router}}$ is computed with the denominator of Eq. (3), and at inference; it is absent from the expert branch, whose routing is restricted to the LoRA pool. w_{res} receives gradient only from the router branch.

I Replay Generation

Replay for domain τ is generated in two stages by the model obtained after domain $t-1$. In the first stage the input is the backbone's chat template up to the beginning of the user turn, which is identical for every domain, and the model generates the user turn with the routing override of Eq. (5) applied at the first generated token only; the output is kept as the question. In the second stage the question is wrapped in the template and the model generates the assistant turn without override, routing over the full pool as at inference; the output is the answer. Questions are sampled with temperature 1.0 and top-p 0.95 and answers are decoded greedily, both up to 1024 tokens or an end-of-sequence token. For the backbone ($\tau = 0$) both stages run without override. We generate $N = 500$ pairs per domain, use all of them, and discard them after training on domain t. Examples are in Sec. E.2.

Learned prompt (Tab. 5c). For each task we replace the embedding of the BoS token with a learned vector, trained on that task's training data with the language-modeling loss while all other parameters are frozen, and generate replay by sampling from the model with that embedding in place and no routing override.

J Implementation Details

Architecture. On TRACE, LoRA experts of rank 64 are attached to the q, k, v, o projections and the FFN of every layer, with one router per layer shared across the modules. Each task adds one expert ($m = 1$), top-1 routing ($k = 1$), and one skip expert. The router rows and the reservoir row are the only router parameters; the reservoir adds one logit per layer.

Training. Each task is trained for 3–7 epochs with AdamW ($\beta = (0.9, 0.999)$, no weight decay), learning rate 2×10^{-4} with a cosine schedule and 3% linear warmup, and a global batch size of 64, following Yang et al. (2026). The expert branch and the router branch each process one batch per step; the router branch batch mixes $\mathcal{D}_t$ and $\mathcal{R}_{<t}$ in equal shares, one share for $\mathcal{D}_t$ and one for each source in $\mathcal{R}_{<t}$ (each earlier domain and the backbone). The margin loss weight is $\lambda = 0.1$, the reservoir noise $\sigma = 0$. The reservoir count stays at 0 for the first 5% of steps and is then restored linearly from 0 to m over the remaining steps.

Replay. $N = 500$ sequences per domain, 1.07M tokens in total over the eight tasks on Llama-3.1-8B-Instruct; generation settings are in Sec. E. The backbone domain ($\tau = 0$) uses the same N.

Baselines. All LoRA baselines use rank 64 (α=128, dropout 0.05) with their own target modules; O-LoRA adapts only q and v as in its reference implementation. EWC uses online EWC with $\lambda_{\text{EWC}} = 400$ and a diagonal Fisher estimated from 1,000 training samples per task; O-LoRA uses orthogonality weight $\lambda_1 = 0.5$ and no ℓ_2 term ($\lambda_2 = 0$), with one rank-64 block per task merged at the end; S-LoRA adapts all seven projections and uses the `max` denoising mode. Lifelong-MoE‡ freezes earlier experts and their router rows, keeps a shared rank-64 attention adapter trainable across tasks, and applies the output KL to the previous model on new-task data only, with weight 1.5 (temperature 1). MoE-LPR‡ trains the new expert and its router row on new-task data with earlier experts and router rows frozen, then tunes only the router for 20% of that task's update count on an equal mix of new-task data and a fixed 10% subset (500 records) of each earlier task, with the language-modeling loss plus the routing prior with weight $\gamma = 0.1$. MTL trains one rank-64 LoRA on the union of all eight training sets, repeating each task for its own epoch count (5/3/7/5/3/5/5/7).

K FLAME-MoE Setting

FLAME-MoE (Kang et al., 2025) is a decoder with 9 layers and hidden size 1024, with a dense FFN in the first layer and an MoE block in each of the remaining eight. Each MoE block holds LoRA experts on the FFN and on the attention projections (q, k, v, o), selected together by one shared router with top-4 routing. We train it from scratch on Wiki, then expand and continue on Code and Conversation. Each task adds $m = 8$ experts per MoE layer, so the first task's experts are the only pool and there is no skip expert. CLoRE applies as in §3 with these substitutions: the expert branch trains the new experts and all router rows on the new corpus over the full pool, with earlier experts frozen; the router branch trains the router on replay over the full pool with the reservoir in the denominator; expansion draws the eight new rows from the reservoir, and the new experts start as a no-op (random FFN experts with a zero output projection, attention LoRA with $B = 0$). Replay is real data: a fixed 0.1% subset of every corpus seen so far, current one included, repeated 200 times (829,440 sequences per task), matching the token budget of MoE-LPR's original replay. Lifelong-MoE and MoE-LPR follow their published settings, with KL weight $\lambda_{\text{KL}} = 1.0$ for Lifelong-MoE and routing-prior weight $\gamma = 0.1$ with task labels for MoE-LPR.