How Much to Expand, and How to Route: Dynamic-Capacity Mixture-of-Experts for Continual Learning

Anonymous authors
Paper under double-blind review

Abstract

Continual learning (CL) adapts large pre-trained models to a stream of new tasks without revisiting earlier data. A common architecture adds a group of LoRA experts per task, freezes old experts, and routes each token through a learned Mixture-of-Experts (MoE) router. Freezing prevents weight overwriting, but forgetting persists through routing drift as old-task tokens shift toward newer experts. Two decisions govern this framework: *how much* capacity each task adds, and *how* the accumulated experts are routed at inference. Existing methods fix the first by hand on both the expert-count and rank axes, and address the second with per-token routing alone or with trained task predictors. We propose DyCapMoE, which makes both decisions data-driven. Dynamic Expert Capacity (ECap) attaches a Hard-Concrete ℓ_0 gate to each new expert with a sign-aware coupling that keeps gate closure monotone under softmax routing, and Dynamic Rank Capacity (RCap) applies the same mechanism to each expert's rank. Gates that reach zero are pruned at task end, so each task retains only the capacity it uses. A prototype task router provides the task-level inference prior from by-products of the forward pass without trainable parameters. Across four CL benchmarks on LVLMs and LLMs, the learned budget retains under a third of the allocated rank at equal or better final accuracy, the task router brings forgetting to near zero, and the full model attains the best final accuracy on all four benchmarks.

1 Introduction

Continual learning (CL) adapts a model to a stream of new tasks without revisiting earlier data, supplying *plasticity* for new tasks and *stability* against catastrophic forgetting (French, 1999; Li & Hoiem, 2017; Kirkpatrick et al., 2017) of old ones. For Large Vision-Language Models (LVLMs) (Liu et al., 2023; Wang et al., 2024b; Bai et al., 2025; Comanici et al., 2025) and Large Language Models (LLMs), trained on static data but facing new tasks after deployment, this goal is formalized as Multimodal Continual Instruction Tuning (MCIT) (He et al., 2023; Cao et al., 2024; Chen et al., 2024a) and CL of LLMs (Wang et al., 2023c).

A common approach for CL of pre-trained models adds a group of LoRA experts per task, freezes all previous experts, and routes each token to a sparse subset through a learned router (Huai et al., 2025; Zhao et al., 2025a; 2026); we refer to this as Incremental MoE-LoRA (IncMoE-LoRA). Parameter isolation through freezing prevents direct weight overwriting, but forgetting persists because old-task tokens can be routed to experts from newer tasks. This failure, termed *routing drift* (Zhao et al., 2026), occurs when training new experts and router columns shifts the routing of old-task tokens toward experts that were never trained on them.

Prior work mitigates routing drift during training or at inference. DyMoE constrains token assignment during training to reduce drift (Zhao et al., 2026), but does not guide inference. Task-level routing infers the task of each input and restricts the expert pool accordingly; it reduces forgetting even when combined with train-time regularization (Zhao et al., 2026). Existing designs buy task-level guidance with heavy machinery: an additional encoding of the full input through the language backbone (Zhao et al., 2025a), or trained task predictors that rely on an external encoder (Huai et al.,

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

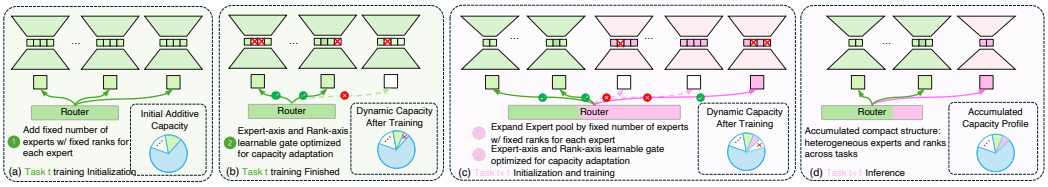

Figure 1: DyCapMoE training lifecycle across consecutive tasks. (a) At Task t, a fixed budget of new LoRA experts is added at a fixed rank. (b) ECap gates on the expert axis and RCap gates on the rank axis jointly prune redundant experts and rank directions during training, and surviving experts with their compact ranks are frozen at task end. (c) Task $t{+}1$ inherits the frozen pool and adds a fresh group that undergoes the same gate-driven pruning. (d) At inference, the accumulated pool contains heterogeneous expert counts and per-expert ranks, and the router selects from it.

2025; Zhao et al., 2025b; Guo et al., 2025a). A task prior built from by-products of the forward pass adds no parameters and no extra forward pass (Sec. 4.3).

The amount of capacity each task adds is also an open problem. Existing methods fix both the number of new experts and the rank of each, and in CL these allocations are irreversible: redundant experts and rank directions freeze permanently and remain routing candidates for all later tasks. A larger fixed budget stores more redundancy and also forgets more in our experiments (Sec. 4.3), yet the capacity a task requires varies widely across tasks and module types (Sec. 4.3). The few methods that adapt capacity decide it outside the task's own training, through pre-sampling detectors (Zhang et al., 2026), probing stages (Zhao et al., 2025a), hand-set compression schedules with distillation (Meng et al., 2026), or proxies distributing a fixed total across layers (Ge et al., 2025a).

We propose **DyCapMoE** (**Dy**namic **Cap**acity **MoE**), which addresses both decisions (Fig. 1). For *how much to expand*, DyCapMoE over-provisions each task and prunes what training does not use. **Dynamic Expert Capacity (ECap)** attaches a Hard-Concrete ℓ_0 gate to every new expert, so that the language-modeling loss and a sparsity penalty jointly decide which experts survive. Closing a gate should not raise the expert's routing probability, yet the naive multiplicative coupling violates this monotonicity under softmax; a *sign-aware* gate–router interface restores it by gating only the positive part of the routing logit. **Dynamic Rank Capacity (RCap)** applies the same ℓ_0 machinery within each surviving expert to prune rank directions. Because saturated gates reach exactly zero, the corresponding experts and rank directions are removed from the cumulative tensors at task end. Every later task inherits a compact pool with heterogeneous expert counts and ranks, and the per-task budget is determined by training. For *how to route*, a **prototype task router** provides the task-level prior without trainable parameters and without an extra forward pass: during training each task accumulates two anchors as running means of the vision tower's CLS token and the LLM's input-token embeddings; at inference a sample is matched to the nearest anchors by cosine similarity, and experts of later tasks are hard-masked from the top-k softmax. Under parameter isolation a correctly routed sample reproduces its end-of-task outputs exactly, which bounds forgetting by the routing error rate (Prop. A.2). The two decisions are complementary: the learned budget determines what each task stores and keeps the expert pool small, while the task prior determines which stored experts a sample uses. Sec. 4 evaluates them separately.

In summary, our contributions are:

- We frame dynamic-MoE continual learning by two coupled decisions: how much capacity each task adds, grounded in the irreversibility of frozen allocations, and how the accumulated experts are routed, grounded in routing drift (Zhao et al., 2026).
- We make per-task capacity end-to-end learnable on both axes. **ECap** gates each new expert through a sign-aware coupling that keeps gate closure monotone under softmax routing; **RCap** applies the same ℓ_0 machinery to each expert's rank. Saturated gates are pruned at task end, so each task stores under a third of the rank it is allocated, and every fixed budget we compare with stores more rank and reaches a lower final accuracy (Sec. 4.3). (Secs. 3.4, 3.5)
- We provide the task-level routing prior through a lightweight prototype task router that reuses by-products of the forward pass and adds no trainable parameters. Later-task experts are masked, and under parameter isolation the routing error rate bounds backward transfer (Prop. A.2). (Sec. 3.6)

- Across four CL benchmarks on multiple LVLM and LLM backbones, ECap and RCap keep under a third of the allocated rank budget at equal or better final accuracy with a much smaller deployed adapter. The task router brings backward transfer to near zero, and the full model attains the best final accuracy on all four benchmarks ahead of every baseline using task-level inference. (Sec. 4)

2 Related Work

Continual learning with pre-trained models. Continual learning methods mitigate forgetting by regularization, rehearsal or task-specific parameters (Kirkpatrick et al., 2017; Li & Hoiem, 2017; Rebuffi et al., 2017; Chaudhry et al., 2019; Serra et al., 2018; Yoon et al., 2018; Yan et al., 2021). For pre-trained models, CL is carried out through PEFT (Wang et al., 2022c; Liang & Li, 2024) on benchmarks for MCIT (He et al., 2023; Chen et al., 2024a; Guo et al., 2025a; Zhao et al., 2025b) and for LLMs (Wang et al., 2023c), most often with a pool of LoRA experts that grows by one group per task (Chen et al., 2024a; Yu et al., 2025; Huai et al., 2025; Zhao et al., 2025a; 2026). DyMoE regularizes routing drift in this pool during training under a fixed per-task budget (Zhao et al., 2026); DyCapMoE instead learns how much to expand. See App. M for a fuller discussion.

MoE with LoRA and adaptive capacity. MoE-LoRA activates a sparse subset of low-rank experts per input and has been adopted for both fine-tuning and CL (Shazeer et al., 2017; Fedus et al., 2022; Chen et al., 2023; Li et al., 2024a; Dou et al., 2024; Chen et al., 2024a). Outside CL, the rank or the number of experts is adapted within a single training run, since earlier data remains available (Zhang et al., 2023b; Ding et al., 2023; Guo et al., 2024; Louizos et al., 2018; Xia et al., 2022). In CL, capacity is decided outside the task's own training, by scheduled compression with distillation (Meng et al., 2026), pre-training detectors (Zhang et al., 2026), layer-allocation proxies under a fixed total (Ge et al., 2025a) or probe stages (Zhao et al., 2025a). DyCapMoE learns the per-task expert count and per-expert rank end-to-end with the task loss.

Task-level routing and task-identity inference. Inferring the task of a test input is an established sub-problem of CL (Wang et al., 2023a). Training-free rules match frozen features to stored prototypes (Rebuffi et al., 2017; Wang et al., 2022a; Zhou et al., 2024), whereas MoE-based CL relies on trained task predictors or extra forward passes (Zhao et al., 2025a; Huai et al., 2025; Zhao et al., 2025b; 2024). Our prototype task router applies the training-free rule to the incremental expert pool and hard-masks later-task experts, which bounds backward transfer by the routing error rate under parameter isolation (Prop. A.2).

3 Methodology

3.1 Problem Setup

We study *continual instruction tuning* under a unified formulation covering both MCIT on LVLMs and CL of LLMs. The model encounters a stream of T tasks $\{\mathcal{D}_1, \ldots, \mathcal{D}_T\}$ arriving sequentially, with access only to $\mathcal{D}_t$ during training on task t. Each $\mathcal{D}_t = \{(\mathbf{x}_v^{(j)}, \mathbf{x}_q^{(j)}, \mathbf{x}_a^{(j)})\}_{j=1}^{M_t}$ contains M_t instruction-response samples with visual, instruction, and answer token sequences $\mathbf{x}_v, \mathbf{x}_q, \mathbf{x}_a$ respectively. For LLMs, $\mathbf{x}_v$ is empty.

For each task t, the trainable parameters θ are optimized with the standard next-token-prediction loss on the answer tokens conditioned on the (multimodal) prompt:

$$\theta_t^* \;=\; \arg\min_{\theta} \; \mathbb{E}_{(\mathbf{x}_v, \mathbf{x}_q, \mathbf{x}_a) \sim \mathcal{D}_t} \left[- \sum\nolimits_{n=1}^{|\mathbf{x}_a|} \log p_\theta\big(\mathbf{x}_a^{(n)} \,\big|\, \mathbf{x}_v, \mathbf{x}_q, \mathbf{x}_a^{(<n)}\big) \right], \tag{1}$$

where $\mathbf{x}_a^{(n)}$ is the n-th answer token, $\mathbf{x}_a^{(<n)}$ its left context, and $|\mathbf{x}_a|$ the answer length. Two CL constraints distinguish this from standard instruction tuning: past-task data cannot be revisited, and task identity is *not* provided at inference. The goal is to sequentially update θ across T tasks while retaining performance on every previously seen task.

3.2 Preliminary: Incremental MoE with LoRA

MoE-LoRA (Hu et al., 2022; Chen et al., 2023; Li et al., 2024a; Dou et al., 2024) replaces each linear layer with N low-rank LoRA experts coordinated by a sparse router, activating only a sparse subset

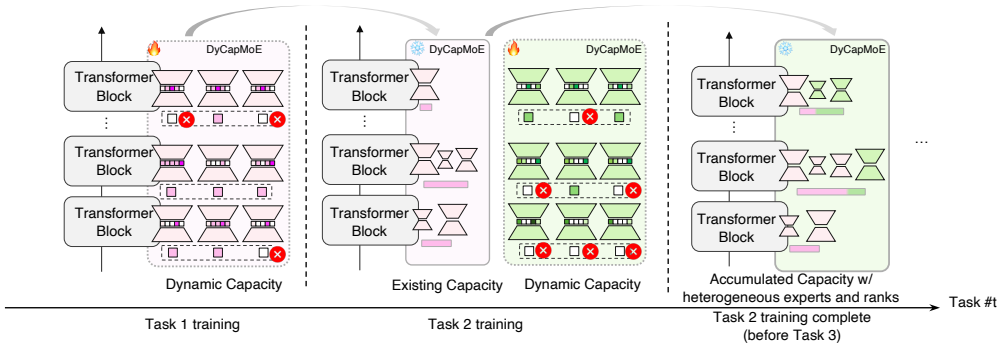

Figure 2: **DyCapMoE lifecycle.** At task t, every linear layer holds the frozen inherited pool plus an over-provisioned group of new experts; ECap gates each new expert (through the sign-aware coupling to the router) and RCap gates each rank direction, trained jointly with the task loss under ℓ_0 penalties, while the task anchor $\mathbf{a}_t$ accumulates as a running feature mean at no extra cost. At task end, saturated gates are pruned and survivors are frozen, so task $t{+}1$ inherits a compact pool with heterogeneous expert counts and ranks. At inference, the prototype task router matches each query to the nearest anchors and hard-masks experts of later tasks; the per-token router then selects top-k among the allowed set.

of experts per token. Adding one new expert group per task yields Incremental (Inc) MoE-LoRA, the prevailing template for parameter-isolation continual learning of large pre-trained models (Chen et al., 2024a; Huai et al., 2025; Zhao et al., 2025a; Qian et al., 2025).

MoE Layers with LoRA. At each linear layer of the backbone Transformer, a MoE-LoRA layer replaces a single LoRA adapter with N low-rank expert adapters $\{(\mathbf{A}_i^l, \mathbf{B}_i^l)\}_{i=1}^N$ coordinated by a router $R^l(\cdot) : \mathbb{R}^d \to \mathbb{R}^N$, where $\mathbf{A}_i^l \in \mathbb{R}^{r\times d}$, $\mathbf{B}_i^l \in \mathbb{R}^{d\times r}$, and the rank $r \ll d$. Given an input token representation $\mathbf{h}^l \in \mathbb{R}^d$, the router produces routing logits $\mathbf{s} = R^l(\mathbf{h}^l)$ and applies top-k gating to select a sparse subset of experts. The layer output combines the frozen pre-trained projection $\mathbf{W}_0^l\mathbf{h}^l$ with a weighted sum of the selected expert updates:

$$\mathbf{h}_{\text{out}}^l \;=\; \mathbf{W}_0^l\mathbf{h}^l \;+\; \sum\nolimits_{i=1}^{N} w_i\, \mathbf{B}_i^l\mathbf{A}_i^l\mathbf{h}^l, \qquad w_i = \begin{cases} \mathrm{Softmax}(\mathbf{s})_i, & i \in \mathrm{TopK}(\mathbf{s}, k), \\ 0, & \text{otherwise}, \end{cases} \tag{2}$$

where $\mathrm{TopK}(\mathbf{s}, k)$ denotes the indices of the k largest entries of $\mathbf{s}$, so per-token compute stays controlled as the pool grows.

Incremental expansion across tasks. At task t, the expert pool is expanded with N_t new experts and the active pool contains $N = N_t^{\text{old}} + N_t$ experts, where $N_t^{\text{old}} = \sum_{t'<t} N_{t'}$. The router R^l is expanded with N_t additional output columns accordingly. With $\mathcal{E}^{\text{new}} = \{N_t^{\text{old}}{+}1, \ldots, N\}$, training on $\mathcal{D}_t$ updates only $\{(\mathbf{A}_e^l, \mathbf{B}_e^l) : e \in \mathcal{E}^{\text{new}}\}$ and the new router columns, and all old experts and router columns remain frozen. The forward pass runs over the full N experts, so the router can mix old and new capacity end-to-end.

3.3 Method Overview

IncMoE-LoRA leaves both of its governing decisions unaddressed. It fixes per-task capacity on two axes simultaneously, the expert count N_t and the per-expert rank r, and both allocations are *irreversible*: parameters committed to one task cannot be reclaimed, so over-allocation accumulates permanently and every redundant expert remains a routing candidate for all later tasks. It also routes purely per token, so the routing of old-task inputs can drift toward newly added experts (Zhao et al., 2026). DyCapMoE addresses the first decision with two parallel Hard-Concrete ℓ_0 gates learned with the task: ECap adapts N_t by pruning whole experts the task does not need (Sec. 3.4), and RCap adapts r by pruning rank directions within each surviving expert (Sec. 3.5). It addresses the second decision at inference with a prototype task router that activates only the task-relevant expert subset, without trainable parameters (Sec. 3.6). Sec. 3.7 states the training objective.

3.4 DYNAMIC EXPERT CAPACITY (ECAP)

Different tasks require different amounts of capacity beyond the previously learned experts, so a fixed per-task expert count N_t leaves some newly added experts redundant. ECap makes N_t end-to-end learnable by attaching a Hard-Concrete ℓ_0 gate to each new expert. For brevity, Secs. 3.4 and 3.5 omit the layer index l; the per-layer structure is that of Eq. (2).

Hard-Concrete gate. For task t, each new expert $e \in \mathcal{E}^{\text{new}}$ carries a learnable location parameter $\alpha_e \in \mathbb{R}$ that parameterizes a Hard-Concrete gate $g_e \in [0, 1]$ (Louizos et al., 2018), with the standard stretch interval (γ, ζ) and temperature β. Training samples the gate stochastically:

$$u \sim \mathcal{U}(\epsilon, 1-\epsilon),\ s = \sigma((\log u - \log(1-u) + \alpha_e)/\beta)\,,\ g_e = \min\big(1,\ \max\big(0,\ s(\zeta-\gamma)+\gamma\big)\big), \quad (3)$$

where ϵ is a small constant for numerical stability, $\mathcal{U}$ is the uniform distribution and $\sigma(\cdot)$ is the sigmoid. At inference, the noise is removed and the deterministic $s = \sigma(\alpha_e)$ is passed through the same stretch-and-clip. The clipping step assigns g_e positive probability mass at exact values of 0 and 1, so the gate can be exactly closed or open, and an expert whose gate saturates at 0 can be deleted instead of only being down-weighted. Under the deterministic gate, $\alpha_e \le \log(-\gamma/\zeta)$ saturates g_e to 0 and $\alpha_e \ge \log((1-\gamma)/(\zeta-1))$ saturates it to 1. We initialize each α_e just below the upper saturation threshold so every new expert starts almost fully open (α_e=2.2, g_e≈0.98; all gate settings are listed in App. E and their sensitivity in App. H), allowing the language-modeling gradient to assess usefulness from the start of training. After each backward pass, we clamp α_e to a bounded range so that an expert flagged for pruning cannot drift back into the active region within the task.

Sign-aware coupling. A useful capacity gate is monotone: lowering g_e never raises e's routing probability $p_e = \mathrm{Softmax}(\mathbf{s})_e$. The naive multiplicative form $s_e \to g_e s_e$ is not monotone. Differentiating with respect to the gate yields

$$\frac{\partial p_e}{\partial g_e} = s_e\, p_e(1-p_e). \quad (4)$$

When $s_e < 0$, i.e., when the router *rejects* expert e, decreasing g_e increases p_e, which counteracts the ℓ_0 penalty driving the expert toward pruning. We restore monotonicity by gating only the positive part of the logit:

$$s'_e \;=\; s_e \;-\; (1-g_e)\,[s_e]_+, \qquad [s]_+ := \max(s, 0). \quad (5)$$

By construction $\partial p_e/\partial g_e \ge 0$ on all of $\mathbb{R}^N$. In addition, $\partial s'_e/\partial g_e = 0$ when $s_e \le 0$, so a router-rejected expert receives no language-modeling gradient through the gate. Eq. (5) leaves $s'_e \le 0$ but finite at $g_e = 0$, so saturated-zero experts are additionally hard-masked to $s'_e = -\infty$ before TopK and are therefore not selected at training or inference. Monotonicity is preserved under any global logit shift (App. A.5).

Adaptive expert budget. The ℓ_0 penalty on the new-expert gates drives gates to exactly zero, so the corresponding experts can be removed at task end. The Hard-Concrete relaxation (Louizos et al., 2018) gives the closed-form penalty

$$\mathcal{L}_{\ell_0}^{\text{expert}} \;=\; \frac{1}{N_t} \sum\nolimits_{e \in \mathcal{E}^{\text{new}}} \sigma(\alpha_e - \beta \log(-\gamma/\zeta))\,, \quad (6)$$

where the sum runs over new experts only since old α's are frozen.

3.5 DYNAMIC RANK CAPACITY (RCAP)

ECap removes whole experts but cannot remove rank directions within an expert that the router selects. The rank an expert requires is determined by the gap between what the new task demands and what the model can already do, and this gap varies across tasks. Grounding, for example, requires localization ability that the base model largely lacks, whereas ImageNet classification posed as a VQA task relies on recognition ability the base model already has; the former is expected to require a higher rank than the latter. Consistent with this view, the number of rank directions retained by RCap differs across experts and tasks (App. G, Figs. 7–9). RCap applies the same Hard-Concrete ℓ_0 gates to the per-expert rank axis, so that each surviving expert learns how many of its r rank directions to keep.

Adaptive rank budget. For each new expert e and rank direction $j \in \{1, \ldots, r\}$, RCap places a Hard-Concrete gate $\tilde{g}_{e,j} \in [0, 1]$ (Eq. (3), with parameter $\tilde{\alpha}_{e,j}$) on the LoRA bottleneck $\mathbf{A}_i\mathbf{h}$.

Because the gate sits on the bottleneck rather than on a softmax logit, plain multiplication suffices and the sign-aware coupling of Eq. (5) does not apply. With both ECap and RCap enabled, Eq. (2) becomes

$$\mathbf{h}_{\text{out}} \;=\; \mathbf{W}_0\,\mathbf{h} \;+\; \sum\nolimits_{i=1}^{N} w_i\,\mathbf{B}_i\,\big(\tilde{\mathbf{g}}_i \odot \mathbf{A}_i\,\mathbf{h}\big), \tag{7}$$

with $\tilde{\mathbf{g}}_i = (\tilde{g}_{i,1}, \ldots, \tilde{g}_{i,r}) \in [0,1]^r$ ($= \mathbf{1}$ for old experts, whose surviving rank gates are folded into the factors at the end of their task), and w_i already absorbing the ECap gates. The rank-axis penalty mirrors Eq. (6):

$$\mathcal{L}_{\ell_0}^{\text{rank}} \;=\; \frac{1}{N_t\,r} \sum\nolimits_{e\in\mathcal{E}^{\text{new}}} \sum\nolimits_{j=1}^{r} \sigma(\tilde{\alpha}_{e,j} - \beta \log(-\gamma/\zeta))\,. \tag{8}$$

Post-task pruning (ECap and RCap). At task end, both gates apply a deterministic prune. Experts whose ECap gate has saturated to zero are removed from the cumulative LoRA factors, router columns and gate parameters. Rank directions whose RCap gate has saturated to zero are removed by dropping the corresponding rows of $\mathbf{A}_i$ and columns of $\mathbf{B}_i$ for each surviving expert i. Once training on the task ends, each surviving rank gate is a fixed scalar and is folded into the corresponding row of $\mathbf{A}_i$; old experts thus carry no rank gate at inference, and Eq. (7) holds for them with $\tilde{\mathbf{g}}_i = \mathbf{1}$. The ECap gate value of a surviving expert enters Eq. (5) through a rectified term, cannot be folded into its router column, and remains a fixed constant. Task $t{+}1$ therefore inherits a compact pool of surviving experts, whose ranks may differ, and adds a fresh group of trainable experts on top. Together, ECap and RCap make per-task capacity adaptive on *both* axes that drive parameter growth, so the cumulative footprint grows more slowly than under a fixed budget (Sec. 4). Beyond parameter efficiency, pruning on both axes restricts the subspace in which new experts can perturb old-task outputs (Lemmas A.1–A.2, App. A.2), providing a directional account of the BWT improvement observed in the capacity ablation (Table 4; conditional argument in App. A.3, Prop. A.1).

3.6 Prototype Task Routing

Oracle task labels are unavailable in deployment, and the per-token router alone is exposed to routing drift over a growing pool (Zhao et al., 2026). We use a *prototype task router* that reuses by-products of the forward pass.

Anchor maintenance. For each task i we maintain two per-task anchors: an image anchor $\mathbf{a}_i^I$ from the vision-tower output and a text anchor $\mathbf{a}_i^T$ from a text encoder ϕ^T. The text encoder ϕ^T is either the mean-pooled LLM input-token embedding, which reuses weights already in the model, or an external text encoder such as CLIP (Radford et al., 2021). Each anchor is the per-task feature mean $\mathbf{a}_i = \frac{1}{|\mathcal{D}_i|}\sum_{\mathbf{x}\in\mathcal{D}_i}\phi(\mathbf{x})$, accumulated online during expert training so it adds no new trainable parameters and no separate extraction pass. The input-token embedding is already computed in the forward pass and reaches the same accuracy as the CLIP text encoder (Table 5), so we use it as the default ϕ^T throughout.

Task scoring and expert masking. At inference, we predict the task of an incoming sample as the one whose anchors best match its features. The match score to a previous task $i \le t$ is the sum of per-modality cosine similarities to its anchors,

$$\text{score}_i \;=\; \cos\big(\phi^I(\mathbf{x}),\,\mathbf{a}_i^I\big) \;+\; \cos\big(\phi^T(\mathbf{x}),\,\mathbf{a}_i^T\big), \tag{9}$$

which reduces to the text term alone for LLMs. With $\hat{t} = \arg\max_i \text{score}_i$ as the predicted task, experts of tasks $> \hat{t}$ are hard-masked from the top-k softmax. Per-sample overhead is one mean-pool and a few cosine evaluations against the stored anchors, negligible compared to a single LMM forward pass. Under IncMoE-LoRA's parameter isolation, hard-masking experts of later tasks guarantees that correctly-routed samples produce layer outputs identical to those at the end of their task's training, bounding $|\text{BWT}|$ by the routing error rate (Prop. A.2, App. A).

3.7 Training Objective

The per-task objective couples next-token-prediction with the Hard-Concrete sparsity penalties of Eqs. (6) and (8):

$$\mathcal{L} \;=\; \mathcal{L}_{\text{NTP}} \;+\; \lambda_{\ell_0}^{\text{expert}}\,\mathcal{L}_{\ell_0}^{\text{expert}} \;+\; \lambda_{\ell_0}^{\text{rank}}\,\mathcal{L}_{\ell_0}^{\text{rank}}, \tag{10}$$

where $\lambda_{\ell_0}^{\text{expert}}$ and $\lambda_{\ell_0}^{\text{rank}}$ are scalar hyperparameters, and $\mathcal{L}_{\text{NTP}}$ is computed over output tokens only.

Table 1: Comparison on the CoIN benchmark.

Method	Accuracy on Each Task (%)								Aggregate Results (%)		
	SQA	VQA^{Text}	ImgNet	GQA	VizWiz	REF	VQA^{v2}	VQA^{OCR}	MFN↑	MAA↑	BWT↑
Without task-level prior at inference (w/o TR)											
LoRA	52.56	48.12	39.27	44.47	37.46	1.22	56.10	55.11	41.79	43.99	-23.12
MoELoRA	72.01	46.89	44.75	42.79	28.22	3.31	55.74	57.72	43.93	43.92	-22.18
EWC	59.11	47.21	39.88	45.12	35.33	2.72	56.29	41.21	40.86	43.75	-21.76
LWF	62.32	48.66	51.45	45.84	43.76	0.24	54.96	44.63	43.98	44.89	-19.69
O-LoRA	75.61	49.98	78.24	44.18	30.70	4.66	55.51	57.37	49.53	46.65	-17.54
DyMoE	**76.25**	**53.86**	**95.80**	48.40	52.35	**9.25**	58.30	62.00	57.03	57.70	**-4.67**
DyCapMoE w/o TR	74.05	52.22	94.30	**55.95**	**54.65**	**9.25**	**62.70**	**62.60**	**58.22**	**61.22**	-9.26
With task-level prior at inference (TR)											
ProgLoRA	74.84	51.83	83.90	49.93	53.87	31.19	62.71	**64.44**	59.09	62.38	-6.59
CMoE	77.55	58.17	94.50	48.91	55.45	23.40	56.40	59.44	59.23	62.81	-3.58
DyMoE	78.18	53.36	95.63	54.63	53.92	24.46	59.54	60.40	60.02	60.78	-1.73
PCLR	78.33	**58.24**	86.08	**58.14**	57.61	33.04	**64.17**	61.92	62.19	–	-3.39
DyCapMoE (ours)	**79.75**	54.51	**97.55**	56.95	**59.20**	**52.10**	61.55	61.60	**65.40**	**70.60**	**-0.30**

4 Experiments

4.1 Experimental Setup

We evaluate on four continual instruction-tuning benchmarks (App. B): CoIN (Chen et al., 2024a), UCIT (Guo et al., 2025a) and DCL (Zhao et al., 2025b) for LVLMs, where DCL is domain-incremental, and TRACE (Wang et al., 2023c) for LLMs. On each benchmark we compare with representative CL methods (App. C). In Tables 1 and 2, methods are grouped by whether they use task-level information at inference (TR): the upper block compares capacity mechanisms with no task prior on either side, and the lower block compares full systems that all use one. Metrics follow the benchmark protocols (App. D): mean fine-tune, final and average accuracy (MFT, MFN, MAA) and backward transfer (BWT) on the LVLM benchmarks (Chen et al., 2024a; 2025; Guo et al., 2025c), and overall performance (OP) and BWT on TRACE (Wang et al., 2023c; Qian et al., 2025).

Implementation details. DyCapMoE is inserted into every linear layer of the language backbone except the output head, and only the new experts, the corresponding router weights and the ECap/RCap gates are trained. We use the instruction-untuned LLaVA-1.5-7B (Liu et al., 2024) on CoIN, as in Chen et al. (2024a; 2025), both LLaVA-1.5-7B and InternVL-Chat-7B (Chen et al., 2024b) on UCIT and DCL, as in Guo et al. (2025c), and the three LLMs of Qian et al. (2025) on TRACE (Table 3). Hyperparameters and other details are in App. E.

4.2 Main Results

Multimodal benchmarks. In the upper blocks of Tables 1 and 2, where no method uses a task prior at inference, DyCapMoE has the best MFN and MAA on all three benchmarks and the best BWT on UCIT and DCL. With the task router (lower blocks), DyCapMoE obtains the best MFN, the best MAA and near-zero forgetting on all three benchmarks. On UCIT, several baselines reach a higher MFT, but all of them have a lower MFN. Additional results and comparisons are in Apps. F and L.

Continual learning of LLMs. In Table 3, we evaluate DyCapMoE on three LLM backbones using the TRACE benchmark (over three runs). Regularization- and prompt-based methods often fall below FIX(ICL), and plain LoRA shows large forgetting. LoRA-based baselines that reduce forgetting rely on orthogonality constraints (Wang et al., 2023b; 2025b) or per-task structural expansion (Qian et al., 2025). DyCapMoE obtains the best OP and the lowest BWT on all three backbones.

4.3 Discussion

Learned vs. fixed capacity. Table 4 isolates the capacity mechanism on UCIT. The task router is disabled on both sides, so accuracy differences reflect capacity alone. ECap prunes the expert pool to 27.63% of the allocated rank budget; RCap further prunes rank directions within the surviving experts, lowering the retained fraction to 23.48%. Together they raise MFN from 66.27 to 68.03.

Table 2: Comparison on the UCIT and DCL benchmarks.

Method	UCIT				DCL			
	MFT↑	MFN↑	MAA↑	BWT↑	MFT↑	MFN↑	MAA↑	BWT↑
Without task-level prior at inference (w/o TR)								
LoRA	**76.89**	61.24	76.55	-18.78	64.98	53.00	61.13	-14.97
O-LoRA	76.01	64.35	78.02	-13.99	65.16	55.54	62.12	-12.53
MoELoRA	71.17	58.98	75.08	-14.63	64.94	54.78	61.76	-12.71
SEFE	75.98	66.54	78.76	-11.33	65.01	58.51	63.63	-8.13
DyCapMoE w/o TR	74.93	**68.03**	**79.83**	**-8.28**	**67.38**	**67.72**	**68.64**	**0.43**
With task-level prior at inference (TR)								
ModalPrompt	60.76	60.71	70.57	-0.05	61.84	61.78	64.05	-0.07
CL-MoE	71.46	58.49	74.19	-15.56	66.06	54.88	61.79	-13.97
HiDE	69.96	62.29	77.32	-9.20	60.77	55.31	60.68	-6.82
DISCO	**75.87**	69.66	81.60	-7.45	64.78	60.33	63.93	-5.57
DyCapMoE (ours)	74.88	**74.91**	**82.86**	**0.03**	**67.38**	**67.40**	**67.36**	**0.03**

Table 3: Comparison on the TRACE benchmark.

	FIX(ICL)	LoRA	OGD	GEM	EWC	L2P	DualPrompt	HiDeLoRA	O-LoRA	TreeLoRA	**DyCapMoE**
meta-llama / LLaMA-2-7B-Chat											
OP↑	38.94 ± 0.3	34.30 ± 1.2	42.09 ± 1.6	40.08 ± 1.6	42.36 ± 1.2	36.23 ± 0.8	37.69 ± 1.2	41.60 ± 0.8	42.78 ± 0.8	43.52 ± 1.0	**44.27 ± 0.9**
BWT↓	–	18.50 ± 0.8	8.06 ± 1.2	6.77 ± 1.2	5.97 ± 0.8	8.25 ± 0.8	8.03 ± 0.8	7.12 ± 0.4	7.16 ± 0.4	3.46 ± 0.4	**0.92 ± 0.4**
google / Gemma-2B-it											
OP↑	32.30 ± 0.2	31.89 ± 0.8	32.85 ± 1.4	26.48 ± 1.5	28.35 ± 1.6	31.14 ± 1.2	32.42 ± 1.0	33.25 ± 0.9	33.73 ± 0.8	33.41 ± 0.9	**39.14 ± 0.7**
BWT↓	–	15.28 ± 0.4	12.27 ± 0.9	18.25 ± 0.9	16.96 ± 1.2	15.77 ± 0.7	14.25 ± 0.5	13.66 ± 0.5	12.36 ± 0.4	8.50 ± 0.5	**0.84 ± 0.3**
meta-llama / Llama-3.2-1B-Instruct											
OP↑	31.16 ± 0.4	29.73 ± 1.6	30.12 ± 2.0	32.19 ± 2.0	31.96 ± 1.6	29.38 ± 1.2	30.76 ± 1.2	33.73 ± 1.2	32.94 ± 0.8	36.14 ± 0.7	**40.36 ± 1.1**
BWT↓	–	17.03 ± 1.2	15.20 ± 1.6	10.74 ± 1.6	11.62 ± 1.2	13.57 ± 0.8	11.34 ± 0.8	12.36 ± 0.8	12.89 ± 1.2	7.36 ± 0.8	**1.13 ± 0.3**

DyCapMoE stores a rank of 7.51 per task, less than half of the fixed r=16 budget, and still reaches higher MFN and less forgetting (-8.28 vs. -9.91) than the fixed r=32 budget with over four times the capacity. Every fixed budget in Table 4, including r=8 which stores a comparable amount, reaches a lower MFN. The training overhead of the gates and the size of the deployed adapter on CoIN are reported in App. J. The pruned model therefore keeps enough capacity to learn each new task, and the removed units were redundant under the fixed budget. The same pattern holds when RCap alone is added to O-LoRA, which has no MoE components, so the gain comes from RCap itself and not from our architecture (App. H.4). BWT also declines as the fixed budget grows, from -8.97 at r=8 to -9.91 at r=32 (single runs), consistent with redundant capacity enlarging the space in which old-task outputs can be perturbed (App. A.2; conditional argument in App. A.3).

Table 4: Learned vs. fixed budget on UCIT.

Method	Stored rank↓	MFN↑	BWT↑
Fixed budget (IncMoE-LoRA)			
r=8	8	64.70	-8.97
r=16	16	65.36	-9.23
r=32	32	66.27	-9.91
Learned budget (DyCapMoE, allocated r=32)			
+ ECap	8.84 (27.63%)	67.16	-8.76
+ RCap (ours)	**7.51** (23.48%)	**68.03**	**-8.28**

The task router. Enabling the task router on the pruned model (Table 5) brings BWT to near zero on CoIN and UCIT and raises MFN by 6–7 points. DCL is the exception: its domains are separable enough that BWT is already ≈0 without routing, and the task router does not change MFN. The task router adds no trainable parameters. On CoIN its routing accuracy is 89.9%, and the cost relative to ground-truth task IDs is less than one MFN point (App. K). Under parameter isolation and hard masking, $|\mathrm{BWT}|$ is bounded by the mean routing error rate (Prop. A.2), and only routing to an earlier task removes capacity the sample uses (Eq. 9). The LLM input-token embedding and the CLIP text encoder give comparable accuracy (Table 5); we use the former as it adds no extra model.

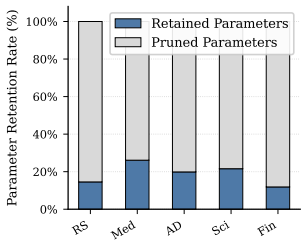

Figure 3: Per-task parameter retention on **DCL**. Retention spans $\sim$12% (Fin) to $\sim$26% (Med).

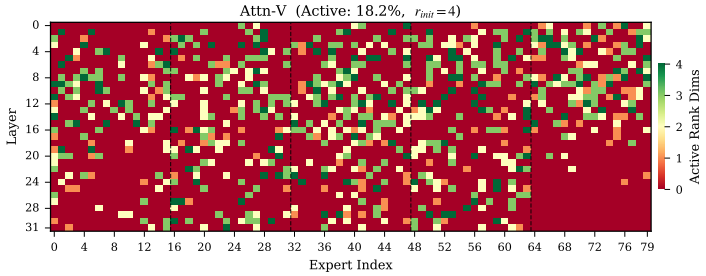

Figure 4: Per-(layer, expert) active rank dimensions for Attn-V on **DCL** (r_{init}=4, 80 experts across 5 tasks). Red: fully pruned; green: all r directions retained. The 18.2% overall active ratio reflects joint ECap-and-RCap pruning.

Where the capacity goes. Capacity retention varies substantially across tasks and module types, so no single fixed budget suits all of them. On UCIT, per-task retention ranges from $\sim$15% (CLEVR, Flickr30k) to $\sim$35% (ImgNet-R) (App. G, Fig. 6), showing that tasks differ widely in capacity demand. DCL shows a narrower range (Fig. 3), consistent with its more uniform domain-incremental structure. Fig. 4 shows the joint effect at the per-(layer, expert) level on Attn-V: ECap prunes most experts entirely (dark-red columns), and RCap further reduces the rank of the survivors (orange-to-green gradient). Across module types, FFN-Gate and FFN-Up are consistently the least pruned, and the attention projections are the most pruned (Attn-O on DCL, Attn-Q on CoIN and UCIT). Active ratios span 12.7% (Attn-O, DCL) to 38.4% (FFN-Up, CoIN); full heatmaps are in App. G. Neither layer depth nor task order explains per-task retention. The capacity a task retains is determined by the gap between what the task requires and what the model can already do when the task arrives (Sec. 3.5), and task order enters only through this gap. On UCIT, ImgNet-R is learned first, cannot reuse any earlier expert and retains the most capacity, whereas CLEVR and Flickr30k are learned fifth and sixth and retain the least. Across three CoIN task orders, retention varies between 28.0% and 31.2% while MFN changes by 0.4 points (App. H.3).

Table 5: Task-router ablation on UCIT.

Variant	MFN↑	BWT↑
DyCapMoE w/o TR	68.03	-8.28
w/ TR (LLM emb.)	74.91	0.03
w/ TR (CLIP)	74.97	0.06

5 Conclusion

We presented DyCapMoE, which addresses both governing decisions of incremental MoE-LoRA continual learning. ECap and RCap learn per-task capacity on the expert-count and rank axes through Hard-Concrete ℓ_0 gates pruned at task end, and a prototype task router provides the task-level inference prior from forward-pass by-products without trainable parameters. The two mechanisms are complementary and evaluated separately. Across four benchmarks on LVLMs and LLMs, the learned budget retains under a third of the allocated rank while matching or exceeding the fixed budget in final accuracy, and the task router brings forgetting to near zero. DyCapMoE attains the best final accuracy in every comparison block that uses task-level inference, with a lower MFT than some baselines on UCIT.

Limitations and future work. ECap and RCap optimize the two capacity axes independently; a joint formulation is left for future work. The inference-time router assumes separable tasks and its benefit decreases as task overlap grows (App. K).

LLM USAGE

We use LLMs solely for polishing and editing the manuscript. Our experiments involve training and evaluating LLMs as part of the experimental setup.

BROADER IMPACT

DyCapMoE reduces the parameter overhead of continual learning along both the expert-count and the rank axis, which makes it more practical to adapt large pre-trained models to new tasks on modest hardware. Slower parameter growth lowers the storage and serving cost of long-lived deployed models. Parameter isolation does not retain or replay earlier training data, which benefits settings where data privacy or storage is a constraint.

As with any parameter-efficient fine-tuning method, the dual-use concern inherent to large pre-trained models applies here as well. This work does not introduce new capabilities or attack surfaces beyond those of the underlying base models. The risks are those common to continual fine-tuning of large language and vision-language models.

REFERENCES

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 139–154, 2018.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020.

Meng Cao, Yuyang Liu, Yingfei Liu, Tiancai Wang, Jiahua Dong, Henghui Ding, Xiangyu Zhang, Ian Reid, and Xiaodan Liang. Continual llava: Continual instruction tuning in large vision-language models. *arXiv preprint arXiv:2411.02564*, 2024.

Shuaichen Chang, David Palzer, Jialin Li, Eric Fosler-Lussier, and Ningchuan Xiao. Mapqa: A dataset for question answering on choropleth maps. *arXiv preprint arXiv:2211.08545*, 2022.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019.

Cheng Chen, Junchen Zhu, Xu Luo, Heng T Shen, Jingkuan Song, and Lianli Gao. Coin: A benchmark of continual instruction tuning for multimodel large language models. *Advances in Neural Information Processing Systems*, 37:57817–57840, 2024a.

Jinpeng Chen, Runmin Cong, Yuzhi Zhao, Hongzheng Yang, Guangneng Hu, Horace Ho Shing Ip, and Sam Kwong. Sefe: Superficial and essential forgetting eliminator for multimodal continual instruction tuning. In *ICML*, 2025.

Zeren Chen, Ziqin Wang, Zhen Wang, Huayang Liu, Zhenfei Yin, Si Liu, Lu Sheng, Wanli Ouyang, Yu Qiao, and Jing Shao. Octavius: Mitigating task interference in mllms via lora-moe. *arXiv preprint arXiv:2311.02684*, 2023.

Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, et al. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 24185–24198, 2024b.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Yu Wu, et al. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*, 2024.

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385, 2021.

Ning Ding, Xingtai Lv, Qiaosen Wang, Yulin Chen, Bowen Zhou, Zhiyuan Liu, and Maosong Sun. Sparse low-rank adaptation of pre-trained language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Wei Shen, Limao Xiong, Yuhao Zhou, Xiao Wang, Zhiheng Xi, Xiaoran Fan, et al. Loramoe: Alleviating world knowledge forgetting in large language models via moe-style plugin. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1932–1945, 2024.

Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. In *International conference on artificial intelligence and statistics*, pp. 3762–3773. PMLR, 2020.

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

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

Chongyang Gao, Kezhen Chen, Jinmeng Rao, Baochen Sun, Ruibo Liu, Daiyi Peng, Yawen Zhang, Xiaoyuan Guo, Jie Yang, and VS Subrahmanian. Higher layers need more lora experts. *arXiv preprint arXiv:2402.08562*, 2024.

Chendi Ge, Xin Wang, Zeyang Zhang, Hong Chen, Jiapei Fan, Longtao Huang, Hui Xue, and Wenwu Zhu. Dynamic mixture of curriculum lora experts for continual multimodal instruction tuning. In *International Conference on Machine Learning*, 2025a.

Chendi Ge, Xin Wang, Zeyang Zhang, Hong Chen, Jiapei Fan, Longtao Huang, Hui Xue, and Wenwu Zhu. Dynamic mixture of curriculum lora experts for continual multimodal instruction tuning. In *International conference on machine learning*, 2025b.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6904–6913, 2017.

Haiyang Guo, Fanhu Zeng, Ziwei Xiang, Fei Zhu, Da-Han Wang, Xu-Yao Zhang, and Cheng-Lin Liu. Hide-llava: Hierarchical decoupling for continual instruction tuning of multimodal large language model. *arXiv preprint arXiv:2503.12941*, 2025a.

Haiyang Guo, Fanhu Zeng, Fei Zhu, Wenzhuo Liu, Da-Han Wang, Jian Xu, Xu-Yao Zhang, and Cheng-Lin Liu. Federated continual instruction tuning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1325–1335, 2025b.

Haiyang Guo, Fei Zhu, Hongbo Zhao, Fanhu Zeng, Wenzhuo Liu, Shijie Ma, Da-Han Wang, and Xu-Yao Zhang. Mcitlib: Multimodal continual instruction tuning library and benchmark. *arXiv preprint arXiv:2508.07307*, 2025c.

Yongxin Guo, Zhenglin Cheng, Xiaoying Tang, Zhaopeng Tu, and Tao Lin. Dynamic mixture of experts: An auto-tuning approach for efficient transformer models. *arXiv preprint arXiv:2405.14297*, 2024.

Ziyu Guo, Renrui Zhang, Hao Chen, Jialin Gao, Dongzhi Jiang, Jiaze Wang, and Pheng-Ann Heng. Sciverse: Unveiling the knowledge comprehension and visual reasoning of lmms on multi-modal scientific problems. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 19683–19704, 2025d.

Danna Gurari, Qing Li, Abigale J Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo, and Jeffrey P Bigham. Vizwiz grand challenge: Answering visual questions from blind people. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3608–3617, 2018.

Danna Gurari, Yinan Zhao, Meng Zhang, and Nilavra Bhattacharya. Captioning images taken by people who are blind. In *European Conference on Computer Vision*, pp. 417–434. Springer, 2020.

Jinghan He, Haiyun Guo, Ming Tang, and Jinqiao Wang. Continual instruction tuning for large multimodal models. *arXiv preprint arXiv:2311.16206*, 2023.

Xuehai He, Yichen Zhang, Luntian Mou, Eric Xing, and Pengtao Xie. Pathvqa: 30000+ questions for medical visual question answering. *arXiv preprint arXiv:2003.10286*, 2020.

Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 8320–8329. IEEE, 2021.

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.

Yebowen Hu, Timothy Ganter, Hanieh Deilamsalehy, Franck Dernoncourt, Hassan Foroosh, and Fei Liu. Meetingbank: A benchmark dataset for meeting summarization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16409–16423, 2023.

Tianyu Huai, Jie Zhou, Xingjiao Wu, Qin Chen, Qingchun Bai, Ze Zhou, and Liang He. Cl-moe: Enhancing multimodal large language model with dual momentum mixture-of-experts for continual visual question answering. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 19608–19617, 2025.

Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6700–6709, 2019.

Ching-Yi Hung, Cheng-Hao Tu, Cheng-En Wu, Chien-Hung Chen, Yi-Ming Chan, and Chu-Song Chen. Compacting, picking and growing for unforgetting continual learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. ReferItGame: Referring to objects in photographs of natural scenes. In Alessandro Moschitti, Bo Pang, and Walter Daelemans (eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 787–798, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1086. URL https://aclanthology.org/D14-1086.

Zixuan Ke, Bing Liu, Nianzu Ma, Hu Xu, and Lei Shu. Achieving forgetting prevention and knowledge transfer in continual learning. *Advances in Neural Information Processing Systems*, 34: 22443–22456, 2021.

Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Minjoon Seo, Hannaneh Hajishirzi, and Ali Farhadi. A diagram is worth a dozen images. In *European conference on computer vision*, pp. 235–251. Springer, 2016.

Aniruddha Kembhavi, Minjoon Seo, Dustin Schwenk, Jonghyun Choi, Ali Farhadi, and Hannaneh Hajishirzi. Are you smarter than a sixth grader? textbook question answering for multimodal machine comprehension. In *2017 IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 5376–5384. IEEE, 2017.

Tannon Kew, Marek Kostrzewa, and Sarah Ebling. 20 minuten: A multi-task news summarisation dataset for german. In *Proceedings of the 8th edition of the Swiss Text Analytics Conference*, pp. 1–13, 2023.

Muhammad Gul Zain Ali Khan, Muhammad Ferjad Naeem, Luc Van Gool, Didier Stricker, Federico Tombari, and Muhammad Zeshan Afzal. Introducing language guidance in prompt-based continual learning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 11463–11473, 2023.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Hyunseo Koh, Dahyun Kim, Jung-Woo Ha, and Jonghyun Choi. Online continual learning on class incremental blurry task configuration with anytime inference. *arXiv preprint arXiv:2110.10031*, 2021.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020.

Dengchun Li, Yingzi Ma, Naizheng Wang, Zhengmao Ye, Zhiyuan Cheng, Yinghao Tang, Yan Zhang, Lei Duan, Jie Zuo, Cal Yang, et al. Mixlora: Enhancing large language models fine-tuning with lora-based mixture of experts. *arXiv preprint arXiv:2404.15159*, 2024a.

Hongbo Li, Sen Lin, Lingjie Duan, Yingbin Liang, and Ness Shroff. Theory on mixture-of-experts in continual learning. In *International Conference on Learning Representations*, 2025.

Lei Li, Yuqi Wang, Runxin Xu, Peiyi Wang, Xiachong Feng, Lingpeng Kong, and Qi Liu. Multimodal arxiv: A dataset for improving scientific comprehension of large vision-language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14369–14387, 2024b.

Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In *International conference on machine learning*, pp. 3925–3934. PMLR, 2019.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

Yan-Shuo Liang and Wu-Jun Li. Inflora: Interference-free low-rank adaptation for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 23638–23647, 2024.

Adam Dahlgren Lindström and Savitha Sam Abraham. Clevr-math: A dataset for compositional language, visual and mathematical reasoning. *arXiv preprint arXiv:2208.05358*, 2022.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 26296–26306, 2024.

Sylvain Lobry, Diego Marcos, Jesse Murray, and Devis Tuia. Rsvqa: Visual question answering for remote sensing data. *IEEE Transactions on Geoscience and Remote Sensing*, 58(12):8555–8566, 2020.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

Christos Louizos, Max Welling, and Diederik P Kingma. Learning sparse neural networks through l_0 regularization. In *International Conference on Learning Representations*, 2018.

Haodong Lu, Chongyang Zhao, Jason Xue, Lina Yao, Kristen Moore, and Dong Gong. Little by little: Continual learning via incremental mixture of rank-1 associative memory experts. *arXiv preprint arXiv:2506.21035*, 2025.

Pan Lu, Liang Qiu, Jiaqi Chen, Tony Xia, Yizhou Zhao, Wei Zhang, Zhou Yu, Xiaodan Liang, and Song-Chun Zhu. Iconqa: A new benchmark for abstract diagram understanding and visual language reasoning. *arXiv preprint arXiv:2110.13214*, 2021.

Pan Lu, Swaroop Mishra, Tony Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *The 36th Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 67–82, 2018.

Muhammad Anwar Ma'sum, Mahardhika Pratama, Savitha Ramasamy, Lin Liu, Habibullah Habibullah, and Ryszard Kowalczyk. Vision and language synergy for rehearsal free continual learning. In *The Thirteenth International Conference on Learning Representations*, 2025.

Weicheng Meng, Jingyang Qiao, Zhizhong Zhang, Shaohui Liu, and Yuan Xie. Pclr: Progressively compressed lora for multimodal continual instruction tuning. In *The Fourteenth International Conference on Learning Representations*, 2026.

Anand Mishra, Shashank Shekhar, Ajeet Kumar Singh, and Anirban Chakraborty. Ocr-vqa: Visual question answering by reading text in images. In *2019 international conference on document analysis and recognition (ICDAR)*, pp. 947–952. IEEE, 2019.

Swaroop Mishra, Arindam Mitra, Neeraj Varshney, Bhavdeep Sachdeva, Peter Clark, Chitta Baral, and Ashwin Kalyan. Numglue: A suite of fundamental yet challenging mathematical reasoning tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3505–3523, 2022.

Cuong V Nguyen, Yingzhen Li, Thang D Bui, and Richard E Turner. Variational continual learning. *arXiv preprint arXiv:1710.10628*, 2017.

Yu-Yang Qian, Yuan-Ze Xu, Zhen-Yu Zhang, Peng Zhao, and Zhi-Hua Zhou. Treelora: Efficient continual learning via layer-wise loras guided by a hierarchical gradient-similarity tree. In *International Conference on Machine Learning*, pp. 50066–50085. PMLR, 2025.

Jingyang Qiao, Zhizhong Zhang, Xin Tan, Yanyun Qu, Shouhong Ding, and Yuan Xie. Large continual instruction assistant. In *International conference on machine learning*, 2025.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Veselin Raychev, Pavol Bielik, and Martin Vechev. Probabilistic model for code with decision trees. *ACM SIGPLAN Notices*, 51(10):731–747, 2016.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 2001–2010, 2017.

Matthew Riemer, Tim Klinger, Djallel Bouneffouf, and Michele Franceschini. Scalable recollections for continual lifelong learning. *Proceedings of the AAAI conference on artificial intelligence*, 33 (01):1352–1359, 2019.

Mohammad Rostami, Soheil Kolouri, and Praveen K Pilly. Complementary learning for overcoming catastrophic forgetting using experience replay. *arXiv preprint arXiv:1903.04566*, 2019.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *International conference on machine learning*, pp. 4528–4537. PMLR, 2018.

Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International conference on machine learning*, pp. 4548–4557. PMLR, 2018.

Agam Shah, Suvan Paturi, and Sudheer Chava. Trillion dollar words: A new financial dataset, task & market analysis. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6664–6679, 2023.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. *Advances in neural information processing systems*, 30, 2017.

Chonghao Sima, Katrin Renz, Kashyap Chitta, Li Chen, Hanxue Zhang, Chengen Xie, Jens Beißwenger, Ping Luo, Andreas Geiger, and Hongyang Li. Drivelm: Driving with graph visual question answering. In *European conference on computer vision*, pp. 256–274. Springer, 2024.

Amanpreet Singh, Vivek Natarjan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8317–8326, 2019.

Qwen Team. Qwen2.5: A party of foundation models, September 2024. URL `https://qwenlm.github.io/blog/qwen2.5/`.

Ruilin Tong, Yuhang Liu, Javen Qinfeng Shi, and Dong Gong. Coreset selection via reducible loss in continual learning. In *The Thirteenth International Conference on Learning Representations*, 2025.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Zhiyi Wan, Wanrou Du, Liang Li, Miao Pan, and Xiaoqi Qin. Adaptive budget allocation for orthogonal-subspace adapter tuning in llms continual learning. *arXiv preprint arXiv:2505.22358*, 2025.

Huiyi Wang, Haodong Lu, Lina Yao, and Dong Gong. Self-expansion of pre-trained models with mixture of adapters for continual learning. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 10087–10098, 2025a.

Liyuan Wang, Jingyi Xie, Xingxing Zhang, Mingyi Huang, Hang Su, and Jun Zhu. Hierarchical decomposition of prompt-based continual learning: Rethinking obscured sub-optimality. In *Advances in Neural Information Processing Systems*, 2023a.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024a.

Liyuan Wang, Jingyi Xie, Xingxing Zhang, Hang Su, and Jun Zhu. Hide-pet: continual learning via hierarchical decomposition of parameter-efficient tuning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025b.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuan-Jing Huang. Orthogonal subspace learning for language model continual learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 10658–10671, 2023b.

Xiao Wang, Yuansen Zhang, Tianze Chen, Songyang Gao, Senjie Jin, Xianjun Yang, Zhiheng Xi, Rui Zheng, Yicheng Zou, Tao Gui, et al. Trace: A comprehensive benchmark for continual learning in large language models. *arXiv preprint arXiv:2310.06762*, 2023c.

Xinlong Wang, Xiaosong Zhang, Zhengxiong Luo, Quan Sun, Yufeng Cui, Jinsheng Wang, Fan Zhang, Yueze Wang, Zhen Li, Qiying Yu, et al. Emu3: Next-token prediction is all you need. *arXiv preprint arXiv:2409.18869*, 2024b.

Yabin Wang, Zhiwu Huang, and Xiaopeng Hong. S-prompts learning with pre-trained transformers: An occam's razor for domain incremental learning. In *Advances in Neural Information Processing Systems*, 2022a.

Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, et al. Dualprompt: Complementary prompting for rehearsal-free continual learning. In *European conference on computer vision*, pp. 631–648. Springer, 2022b.

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 139–149, 2022c.

Ziqi Wang, Chang Che, Qi Wang, Yangyang Li, Zenglin Shi, and Meng Wang. Smolora: Exploring and defying dual catastrophic forgetting in continual visual instruction tuning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 177–186, 2025c.

Xiwen Wei, Mustafa Munir, and Radu Marculescu. Mitigating intra-and inter-modal forgetting in continual learning of unified multimodal models. *Advances in Neural Information Processing Systems*, 2025.

Mitchell Wortsman, Vivek Ramanujan, Rosanne Liu, Aniruddha Kembhavi, Mohammad Rastegari, Jason Yosinski, and Ali Farhadi. Supermasks in superposition. *Advances in Neural Information Processing Systems*, 33:15173–15184, 2020.

Mengzhou Xia, Zexuan Zhong, and Danqi Chen. Structured pruning learns compact and accurate models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, 2022.

Ju Xu and Zhanxing Zhu. Reinforced continual learning. In *Advances in Neural Information Processing Systems*, 2018.

Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3014–3023, 2021.

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

Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. In *International Conference on Learning Representations*, 2018.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the association for computational linguistics*, 2:67–78, 2014.

Jiazuo Yu, Yunzhi Zhuge, Lu Zhang, Ping Hu, Dong Wang, Huchuan Lu, and You He. Boosting continual learning of vision-language models via mixture-of-experts adapters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 23219–23230, 2024.

Yahan Yu, Duzhen Zhang, Yong Ren, Xuanle Zhao, Xiuyi Chen, and Chenhui Chu. Progressive lora for multimodal continual instruction tuning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 2779–2796, 2025.

Fanhu Zeng, Fei Zhu, Haiyang Guo, Xu-Yao Zhang, and Cheng-Lin Liu. Modalprompt: Towards efficient multimodal continual instruction tuning with dual-modality guided prompt. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 12137–12152, 2025.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International conference on machine learning*, pp. 3987–3995. PMLR, 2017.

Duzhen Zhang, Yong Ren, Zhong-Zhi Li, Yahan Yu, Jiahua Dong, Chenxing Li, Zhilong Ji, and Jinfeng Bai. Enhancing multimodal continual instruction tuning with branchlora. *arXiv preprint arXiv:2506.02041*, 2025.

Gengwei Zhang, Liyuan Wang, Guoliang Kang, Ling Chen, and Yunchao Wei. Slca: Slow learner with classifier alignment for continual learning on a pre-trained model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 19148–19158, 2023a.

Junting Zhang, Jie Zhang, Shalini Ghosh, Dawei Li, Serafettin Tasci, Larry Heck, Heming Zhang, and C-C Jay Kuo. Class-incremental learning via deep model consolidation. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 1131–1140, 2020.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. In *International Conference on Learning Representations*, 2023b.

Ying Zhang, Xingyue Guo, Yu Zhao, Xuhui Sui, Baohang Zhou, Xinying Qian, and Xiaojie Yuan. Grow-on-demand: Sparse and adaptive expert expansion for continual instruction tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026.

Chenye Zhao, Yingjie Li, and Cornelia Caragea. C-stance: A large dataset for chinese zero-shot stance detection. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13369–13385, 2023.

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. *arXiv preprint arXiv:2603.27481*, 2026.

Hengyuan Zhao, Ziqin Wang, Qixin Sun, Kaiyou Song, Yilin Li, Xiaolin Hu, Qingpei Guo, and Si Liu. Llava-cmoe: Towards continual mixture of experts for large vision-language models. *arXiv preprint arXiv:2503.21227*, 2025a.

Hongbo Zhao, Fei Zhu, Haiyang Guo, Meng Wang, Rundong Wang, Gaofeng Meng, and Zhaoxiang Zhang. Mllm-cl: Continual learning for multimodal large language models. *arXiv preprint arXiv:2506.05453*, 2025b.

Weixiang Zhao, Shilong Wang, Yulin Hu, Yanyan Zhao, Bing Qin, Xuanyu Zhang, Qing Yang, Dongliang Xu, and Wanxiang Che. Sapt: A shared attention framework for parameter-efficient continual learning of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024.

Da-Wei Zhou, Zi-Wen Cai, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Revisiting class-incremental learning with pre-trained models: Generalizability and adaptivity are all you need. *International Journal of Computer Vision*, 2024.

Didi Zhu, Zhongyi Sun, Zexi Li, Tao Shen, Ke Yan, Shouhong Ding, Kun Kuang, and Chao Wu. Model tailor: Mitigating catastrophic forgetting in multi-modal large language models. In *International conference on machine learning*. PMLR, 2024.

Heming Zou, Yunliang Zang, Wutong Xu, Yao Zhu, and Xiangyang Ji. Flylora: Boosting task decoupling and parameter efficiency via implicit rank-wise mixture-of-experts. *Advances in Neural Information Processing Systems*, 2025.

A THEORETICAL IMPLICATIONS

Section A.2 proves the subspace-inclusion property cited in Sec. 3.5, and Sec. A.3 gives a conditional account of the BWT improvement attributable to ECap and RCap. Section A.4 proves the structural BWT bound of Sec. 3.6, under which backward transfer is controlled by the routing error rate alone.

A.1 SETUP AND NOTATION

Fix a linear layer of the backbone Transformer and an old task $t' < T$. After training on task T, the layer's new-expert contribution to its output on input $\mathbf{h}$ is

$$\Delta(\mathbf{h}) \;=\; \sum_{e \in \mathcal{E}^{\text{new}}_{>t'}} w_e(\mathbf{h}) \cdot \mathbf{B}_e\big(\tilde{\mathbf{g}}_e \odot \mathbf{A}_e \mathbf{h}\big), \tag{11}$$

where $\mathcal{E}^{\text{new}}_{>t'} = \bigcup_{t=t'+1}^{T} \mathcal{E}^{\text{new}}_t$ is the cumulative new-expert set added after task t', $w_e(\mathbf{h})$ is the routing weight on input $\mathbf{h}$, and $\tilde{\mathbf{g}}_e \in [0,1]^r$ are the RCap rank gates ($\tilde{\mathbf{g}}_e = \mathbf{1}$ when RCap is disabled). Old experts and old router columns are frozen, so their parameters do not change between the end of task t' training and the end of task T training. Their routing weights can still change when the task router is off, because the new router columns enter the top-k selection and the softmax normalization; Eq. (11) covers the new-expert term only, and this renormalization term is not treated in Secs. A.2–A.3. Under the task router with correct routing the new experts are masked, the softmax runs over the old expert set only, and both terms vanish (Sec. A.4). The *perturbation subspace* is

$$\mathcal{H}_T \;:=\; \mathrm{Col}\Big(\big\{\mathbf{B}_e \,\mathrm{diag}(\tilde{\mathbf{g}}_e)\,\mathbf{A}_e \;:\; e \in \mathcal{E}^{\text{new}}_{>t'}\big\}\Big) \;\subseteq\; \mathbb{R}^d, \tag{12}$$

and for any input $\Delta(\mathbf{h}) \in \mathcal{H}_T$.

A.2 SUBSPACE INCLUSION UNDER CAPACITY PRUNING

Lemma A.1 (ECap subspace inclusion)**.** *For any subset $\mathcal{E}^{\text{surv}} \subseteq \mathcal{E}^{\text{new}}_{>t'}$, the perturbation subspace under restriction to $\mathcal{E}^{\text{surv}}$ satisfies $\mathcal{H}_T^{\text{ECap}} \subseteq \mathcal{H}_T^{\text{full}}$. The inclusion is strict whenever some pruned expert has a column outside the span of the retained experts' columns, which is the generic case while the retained rank is below d.*

Proof. For any collection $\{\mathbf{X}_e\}_{e\in\mathcal{E}}$ and subset $\mathcal{S} \subseteq \mathcal{E}$, $\mathrm{Col}(\bigcup_{e\in\mathcal{S}} \mathbf{X}_e) \subseteq \mathrm{Col}(\bigcup_{e\in\mathcal{E}} \mathbf{X}_e)$. Setting $\mathbf{X}_e = \mathbf{B}_e \,\mathrm{diag}(\tilde{\mathbf{g}}_e)\,\mathbf{A}_e$, $\mathcal{S} = \mathcal{E}^{\text{surv}}$, $\mathcal{E} = \mathcal{E}^{\text{new}}_{>t'}$ yields the inclusion. If every column of every pruned expert lies in the span of the retained columns the two spans coincide; otherwise the inclusion is strict. □

Lemma A.2 (RCap subspace inclusion)**.** *Assume that each new expert's down-projection has full row rank, $\mathrm{rank}(\mathbf{A}_e) = r$. For each new expert e with rank gates $\tilde{\mathbf{g}}_e \in [0,1]^r$, $\mathrm{Col}\big(\mathbf{B}_e \,\mathrm{diag}(\tilde{\mathbf{g}}_e)\,\mathbf{A}_e\big) \subseteq \mathrm{Col}(\mathbf{B}_e\mathbf{A}_e)$, with strict inclusion whenever some $\tilde{g}_{e,j} = 0$ and the columns of $\mathbf{B}_e$ are linearly independent. Aggregating, $\mathcal{H}_T^{\text{RCap}} \subseteq \mathcal{H}_T^{\text{full}}$.*

Proof. Write $\mathbf{B}_e = [\mathbf{b}_1, \ldots, \mathbf{b}_r]$ and let $\mathcal{S}_e = \{j : \tilde{g}_{e,j} > 0\}$. Since $\mathbf{A}_e$ has full row rank, $\mathrm{Col}(\mathbf{A}_e) = \mathbb{R}^r$ and hence $\mathrm{Col}(\mathbf{B}_e\mathbf{A}_e) = \mathrm{Col}(\mathbf{B}_e) = \mathrm{span}\{\mathbf{b}_j\}_{j=1}^r$; for the same reason the rows of $\mathrm{diag}(\tilde{\mathbf{g}}_e)\mathbf{A}_e$ indexed by $\mathcal{S}_e$ are linearly independent, so $\mathrm{Col}\big(\mathbf{B}_e \,\mathrm{diag}(\tilde{\mathbf{g}}_e)\,\mathbf{A}_e\big) = \mathrm{span}\{\mathbf{b}_j\}_{j\in\mathcal{S}_e}$. The inclusion follows, and it is strict when some $\mathbf{b}_j$ with $j \notin \mathcal{S}_e$ lies outside $\mathrm{span}\{\mathbf{b}_j\}_{j\in\mathcal{S}_e}$, which holds when the columns of $\mathbf{B}_e$ are linearly independent. Without the rank assumption the inclusion can fail: for $r = 2$ and $\mathbf{A}_e$ with two equal rows, $\tilde{\mathbf{g}}_e = (1, 0)$ gives $\mathrm{span}\{\mathbf{b}_1\}$, which is not contained in $\mathrm{Col}(\mathbf{B}_e\mathbf{A}_e) = \mathrm{span}\{\mathbf{b}_1 + \mathbf{b}_2\}$. Aggregating across experts, $\mathcal{H}_T^{\text{RCap}} \subseteq \mathcal{H}_T^{\text{full}}$. □

Corollary A.1 (Joint inclusion)**.** $\mathcal{H}_T^{\text{ECap+RCap}} \subseteq \mathcal{H}_T^{\text{full}}$.

Lemmas A.1–A.2 and Corollary A.1 are elementary, but they establish that capacity pruning along either axis restricts the column span within which new experts can perturb the layer output. The inclusions hold without any assumption on data, optimization or routing. Strictness holds under the rank conditions stated in the lemmas.

A.3 CONDITIONAL INTERFERENCE REDUCTION

The inclusions constrain the subspace in which the new-expert perturbation can lie and say nothing about its norm, since removing components may also remove cancellation among them. The following condition is sufficient for pruning to lower the perturbation norm. It concerns a single set of trained experts, evaluated once with its gates closed and once with its gates open.

Fix the experts $\{\mathbf{B}_e, \mathbf{A}_e\}_{e \in \mathcal{E}^{\text{new}}_{>t'}}$, router columns and gates obtained by training with ECap and RCap, and fix a layer input $\mathbf{h}$ produced by an old-task sample. Let $\Delta^{(t')}_{\text{kept}}(\mathbf{h})$ denote the perturbation of Eq. (11) at $\mathbf{h}$ with the gates as trained, that is, with the experts at $g_e = 0$ masked and the rank directions at $\tilde{g}_{e,j} = 0$ removed, and let $\Delta^{(t')}_{\text{full}}(\mathbf{h})$ denote the perturbation of the same parameters at the same $\mathbf{h}$ with all gates open. Their difference,

$$\Delta^{(t')}_{\text{pruned}}(\mathbf{h}) \;:=\; \Delta^{(t')}_{\text{full}}(\mathbf{h}) - \Delta^{(t')}_{\text{kept}}(\mathbf{h}),$$

consists of the contribution of the pruned experts and rank directions and of the change in the routing weights of the retained experts caused by opening the gates. Expectations below are over the layer inputs $\mathbf{h}$ that the samples of $\mathcal{D}_{t'}$ induce, written $\mathbf{h} \sim \mathcal{D}_{t'}$.

Assumption A.1 (Non-cancelling pruned capacity)**.** *On old-task layer inputs the pruned part does not cancel the retained perturbation in expectation,*

$$\mathbb{E}_{\mathbf{h} \sim \mathcal{D}_{t'}} \big\langle \Delta^{(t')}_{\text{kept}}(\mathbf{h}),\, \Delta^{(t')}_{\text{pruned}}(\mathbf{h}) \big\rangle \;\geq\; 0.$$

The gates close the experts and rank directions along which the task-T gradient carries no useful signal, since a direction that lowered the language-modeling loss would be kept in spite of the ℓ_0 penalty. These directions therefore hold task-T noise absorbed during training, and the assumption says that this noise is not systematically anti-correlated with the retained perturbation on old-task layer inputs. The lemmas do not imply the assumption, which is an empirical condition.

Proposition A.1 (Conditional interference reduction)**.** *Under Assumption A.1,*

$$\mathbb{E}_{\mathbf{h} \sim \mathcal{D}_{t'}} \big\| \Delta^{(t')}_{\text{kept}}(\mathbf{h}) \big\|^2 \;\leq\; \mathbb{E}_{\mathbf{h} \sim \mathcal{D}_{t'}} \big\| \Delta^{(t')}_{\text{full}}(\mathbf{h}) \big\|^2.$$

Proof. Expanding $\|\Delta_{\text{full}}\|^2 = \|\Delta_{\text{kept}} + \Delta_{\text{pruned}}\|^2 = \|\Delta_{\text{kept}}\|^2 + 2\langle \Delta_{\text{kept}}, \Delta_{\text{pruned}} \rangle + \|\Delta_{\text{pruned}}\|^2$ and taking expectations, the cross term is non-negative by Assumption A.1 and the last term is non-negative. □

Assumption A.2 (Plasticity matching)**.** *The ECap+RCap run and the separately trained fixed-budget run reach the same accuracy on task T, $A^{\text{adaptive}}_{T,T} \approx A^{\text{full}}_{T,T}$.*

Proposition A.1 compares one model with its gates closed and open, whereas the capacity ablation of Table 4 compares the ECap+RCap run with a fixed-budget run trained from scratch, whose experts are different. The proposition carries over to that comparison only under Assumption A.2, which identifies the fixed-budget run with the open-gate model on task T. The ablation supports the assumption: the fixed r=32 and ECap + RCap runs reach MFT 74.53 and 74.93 on UCIT (obtained from the MFN and BWT in Table 4 through the identity in App. D). Proposition A.1 bounds the interference *magnitude* of the new-expert term on old-task inputs and not the BWT itself. Translating $\|\Delta^{(t')}\|$ into $|A_{T,t'} - A_{t',t'}|$ requires a Lipschitz model of accuracy in feature space, which we do not provide, and the renormalization term of Sec. A.1 is not covered. Both assumptions are empirically motivated and not proven. Tighter results would require statistical-learning-theoretic bounds on the generalization gap, an open problem for over-parameterized neural networks beyond the scope of this work.

The interference-reduction argument applies to the router-off ablation, where it gives a directional explanation for the BWT improvement attributable to ECap and RCap alone (Table 4: BWT improves from -9.91 to -8.28 with ECap+RCap and without the task router). With the router enabled and routing correct, $w_e(\mathbf{h}) = 0$ for all new experts, so $\Delta^{(t')}(\mathbf{h}) = \mathbf{0}$ exactly and the BWT bound of Prop. A.2 subsumes Prop. A.1.

A.4 STRUCTURAL BWT BOUND UNDER PROTOTYPE ROUTING

The prototype-based task router of Sec. 3.6 hard-masks the experts of tasks beyond the predicted one, and this step yields the following bound on forgetting.

Let $\hat{t}(\mathbf{x})$ denote the prototype router's task prediction on inference sample $\mathbf{x}$ (Eq. (9)), and let

$$\epsilon_{t'} \;=\; \Pr_{\mathbf{x}\sim\mathcal{D}_{t'}}\big[\hat{t}(\mathbf{x}) \neq t'\big] \tag{13}$$

be the routing error rate on task t'. Let $c_{t'}(\mathbf{x}) \in \{0,1\}$ indicate whether $\mathbf{x}$ is answered correctly at the end of task t', let $c_T(\mathbf{x})$ be the same indicator at the end of task T, and define

$$\Delta_{t'}^{\mathrm{mis}} \;=\; \mathbb{E}_{\mathbf{x}\sim\mathcal{D}_{t'}}\big[\,|c_T(\mathbf{x}) - c_{t'}(\mathbf{x})| \;\big|\; \hat{t}(\mathbf{x}) \neq t'\big] \;\in\; [0,1], \tag{14}$$

the mean absolute change of this indicator over the mis-routed samples of task t'.

Proposition A.2 (Structural BWT bound under prototype routing)**.** *Under IncMoE-LoRA's strict parameter isolation (every expert, router column and gate value from tasks $t \le t'$ frozen at the end of task t') and prototype-router hard masking, the backward transfer satisfies*

$$|\mathrm{BWT}| \;\le\; \frac{1}{T-1}\sum\nolimits_{t'=1}^{T-1} \epsilon_{t'}\,\Delta_{t'}^{\mathrm{mis}}. \tag{15}$$

As a corollary, $\mathrm{BWT} = 0$ *whenever* $\epsilon_{t'} = 0$ *for all* $t' < T$.

Proof. We prove the bound in two steps.

Step 1 (correctly-routed samples contribute zero). Under correct routing $\hat{t}(\mathbf{x}) = t'$, the prototype router excludes experts of tasks $> t'$ from the top-k softmax (Sec. 3.6), so the active expert set for $\mathbf{x}$ is exactly the surviving experts of tasks $\le t'$, $\bigcup_{t\le t'} \mathcal{E}_t^{\mathrm{new}}$. The routing decision is also the same at both times. The features $\phi(\mathbf{x})$ come from frozen modules and the anchors of tasks $\le t'$ are fixed once task t' ends, so the scores of these tasks do not change. If t' attains the maximum score over all tasks $\le T$, it also attains the maximum over the tasks $\le t'$, which are the only ones present at the end of task t'. By parameter isolation, every expert in this set, its router column and its gate values are frozen from the end of task t' onward; $A_{t',t'}$ is evaluated on the model after the prune that closes task t', so both evaluations use the same pool. The softmax routing weights over this set therefore depend only on these frozen parameters and on $\mathbf{x}$, and are identical to the weights computed at the end of task-t' training. The layer output $\mathbf{h}_{\mathrm{out}} = \mathbf{W}_0\mathbf{h} + \sum_e w_e \mathbf{B}_e(\tilde{\mathbf{g}}_e \odot \mathbf{A}_e\mathbf{h})$ is therefore identical at both times. Inducting over layers, the final model output is identical, so a correctly-routed sample contributes zero to $A_{t',t'} - A_{T,t'}$.

Step 2 (aggregation). Decompose the test set of task t' into correctly-routed samples (fraction $1-\epsilon_{t'}$) and mis-routed samples (fraction $\epsilon_{t'}$). By Step 1, correctly-routed samples contribute zero to $A_{T,t'} - A_{t',t'} = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}_{t'}}[c_T(\mathbf{x}) - c_{t'}(\mathbf{x})]$. Over the mis-routed samples the mean of $|c_T - c_{t'}|$ is $\Delta_{t'}^{\mathrm{mis}}$ by definition, so $|A_{t',t'} - A_{T,t'}| \le \epsilon_{t'}\,\Delta_{t'}^{\mathrm{mis}}$. Substituting into $\mathrm{BWT} = \frac{1}{T-1}\sum_{t'=1}^{T-1}(A_{T,t'} - A_{t',t'})$ and applying the triangle inequality yields Eq. (15). The corollary follows immediately from $\epsilon_{t'} = 0 \Rightarrow A_{T,t'} = A_{t',t'}$. □

The bound is a structural property of the architecture class. It holds whenever (a) old experts and router columns are frozen after their task and (b) experts of later tasks are hard-masked. Property (a) is shared by any IncMoE-LoRA system, and property (b) is supplied by the prototype router without trainable parameters or oracle task labels. Both assumptions concern the architecture and not the training data, so the routing error rate $\epsilon_{t'}$ is the only measured quantity in the bound. ECap and RCap complement this bound by shrinking the perturbation subspace on both axes (Sec. A.2), which reduces interference on mis-routed samples under Assumption A.1.

The two results describe forgetting from different angles. Proposition A.2 bounds BWT under correct prototype routing and uses only parameter isolation and hard masking. Proposition A.1 bounds the interference of the new-expert term under capacity pruning and is conditional on Assumption A.1. The task router removes forgetting on correctly routed samples, and ECap and RCap reduce interference on the remaining ones.

A.5 Logit reference of the sign-aware coupling

Eq. (5) subtracts $(1-g_e)[s_e]_+$ from the logit of expert e only and leaves every other logit unchanged. Unlike a global shift, it therefore changes the softmax output, and this is how the gate affects routing. Under a global shift $\mathbf{s} \to \mathbf{s} + c$ the TopK set and the selected experts are unchanged, and for any c

$$\frac{\partial p_e}{\partial g_e} \;=\; [s_e + c]_+ \, p_e(1 - p_e) \;\geq\; 0, \tag{16}$$

so closing a gate never raises an expert's routing probability, whereas the naive form $g_e s_e$ has derivative $s_e \, p_e(1-p_e)$ whose sign flips with s_e (Eq. (4)). A shift changes only how strongly the gate acts and which experts receive language-modeling gradient through the gate. For $s_e + c \leq 0$ the gate has no effect, and only the ℓ_0 term closes the expert. The reference point is therefore set by the router's parameterization and not by the softmax. At $s_e = 0$ the linear router has no preference for expert e.

B Benchmark composition

B.1 CoIN

CoIN (Chen et al., 2024a) comprises eight MCIT tasks: ScienceQA (SQA) (Lu et al., 2022), TextVQA (Singh et al., 2019), ImageNet (Russakovsky et al., 2015), GQA (Hudson & Manning, 2019), VizWiz (Gurari et al., 2018), RefCOCO (REF) (Kazemzadeh et al., 2014), VQAv2 (Goyal et al., 2017), and VQA-OCR (Mishra et al., 2019), covering question answering, grounding, and classification with substantial variation in dataset scale, instruction format, and visual domain.

B.2 UCIT and DCL

UCIT (Guo et al., 2025a) comprises six vision-language tasks that the base LVLMs have not seen during instruction tuning: ImageNet-R (Hendrycks et al., 2021) (ImgNet-R, image classification), ArxivQA (Li et al., 2024b) (multiple-choice QA on scientific figures), VizWiz-Caption (Gurari et al., 2020) (VizWiz, image captioning), IconQA (Lu et al., 2021) (multiple-choice QA on icon images), CLEVR-Math (Lindström & Abraham, 2022) (CLEVR, visual math reasoning), and Flickr30k (Young et al., 2014) (image captioning). **DCL** is the domain continual learning benchmark of MLLM-CL (Zhao et al., 2025b) and comprises five domain-incremental VQA tasks: remote sensing (RS, RSVQA (Lobry et al., 2020)), medical (Med, PathVQA (He et al., 2020)), autonomous driving (AD, DriveLM (Sima et al., 2024)), science (Sci, trained on AI2D (Kembhavi et al., 2016) and SciVerse (Guo et al., 2025d) and tested on MapQA (Chang et al., 2022) and TQA (Kembhavi et al., 2017)), and finance (Fin, StockQA, introduced by MLLM-CL).

B.3 TRACE

TRACE (Wang et al., 2023c) comprises eight LLM tasks in four categories: domain-specific (ScienceQA (Lu et al., 2022), FOMC (Shah et al., 2023), MeetingBank (Hu et al., 2023)), multilingual (C-STANCE (Zhao et al., 2023), 20Minuten (Kew et al., 2023)), code completion (Py150 (Raychev et al., 2016)), and mathematical reasoning (NumGLUE-cm and NumGLUE-ds (Mishra et al., 2022)). We follow the TreeLoRA (Qian et al., 2025) setting.

C Baselines

On the multimodal benchmarks we compare against LoRA (Hu et al., 2022), MoELoRA (Chen et al., 2024a), EWC (Kirkpatrick et al., 2017; Schwarz et al., 2018), LWF (Li & Hoiem, 2017), O-LoRA (Wang et al., 2023b), DyMoE (Zhao et al., 2026), SEFE (Chen et al., 2025), PCLR (Meng et al., 2026), ProgLoRA (Yu et al., 2025), CMoE (Zhao et al., 2025a), ModalPrompt (Zeng et al., 2025), CL-MoE (Huai et al., 2025), HiDE (Guo et al., 2025a) and DISCO (Guo et al., 2025b). ProgLoRA, CMoE, PCLR, ModalPrompt, CL-MoE, HiDE, DISCO and the TR variant of DyMoE use task-level information at inference and are listed in the lower blocks of Tables 1 and 2. On TRACE we follow the protocol of Qian et al. (2025) and compare against FIX (in-context learning) (Brown et al., 2020), LoRA, OGD (Farajtabar et al., 2020), GEM (Lopez-Paz & Ranzato,

2017), EWC, L2P (Wang et al., 2022c), DualPrompt (Wang et al., 2022b), HiDeLoRA (Wang et al., 2025b), O-LoRA and TreeLoRA (Qian et al., 2025).

D EVALUATION METRICS

Let $A_{j,i}$ denote the accuracy on task i evaluated after the model has been trained on task j, with T total tasks in the sequence.

Mean Fine-tune Accuracy (MFT) captures the model's learning capacity by averaging per-task accuracy at the moment each task is first acquired, before any subsequent tasks can cause interference:

$$\text{MFT} = \frac{1}{T}\sum_{i=1}^{T} A_{i,i}.$$

Mean Final Accuracy (MFN) measures overall retention by averaging per-task accuracy after the entire training sequence has concluded:

$$\text{MFN} = \frac{1}{T}\sum_{i=1}^{T} A_{T,i}.$$

Mean Average Accuracy (MAA) tracks performance throughout training by computing the mean of the average accuracies over all tasks seen so far, aggregated across every incremental step:

$$\text{MAA} = \frac{1}{T}\sum_{j=1}^{T}\frac{1}{j}\sum_{i=1}^{j} A_{j,i}.$$

Backward Transfer (BWT) quantifies catastrophic forgetting as the average drop in accuracy on previously learned tasks between their acquisition and the end of training. The final task is excluded since it cannot be subject to subsequent forgetting:

$$\text{BWT} = \frac{1}{T-1}\sum_{i=1}^{T-1}(A_{T,i} - A_{i,i}).$$

A negative BWT indicates forgetting, and values closer to zero are better. Because $A_{T,T}$ enters both MFT and MFN, the three metrics satisfy $\text{MFT} = \text{MFN} - \frac{T-1}{T}\,\text{BWT}$, so MFT is determined by MFN and BWT; the ablation tables in the main text therefore report only MFN and BWT.

TRACE metrics. For the TRACE benchmark (Wang et al., 2023c), we follow Wang et al. (2023c); Qian et al. (2025) and report Overall Performance (OP) and BWT. OP is equivalent to MFN and BWT follows the same definition as above.

E ADDITIONAL IMPLEMENTATION DETAILS

Each task adds $N_t = 16$ new experts with top-$k = 8$ across all benchmarks. The per-expert LoRA rank is $r = 4$ on CoIN and DCL, and $r = 2$ on UCIT. We follow the experimental protocol of Chen et al. (2024a; 2025) for CoIN and Guo et al. (2025c) for UCIT and DCL. The linear router uses a per-benchmark temperature, with the new-router cold-start bias held for the first few hundred steps before release. All models are trained with bf16 mixed precision and DeepSpeed ZeRO-2 on four NVIDIA H100 GPUs.

We follow the standard Hard-Concrete settings of Louizos et al. (2018): stretch interval $(\gamma, \zeta) = (-0.1,\ 1.1)$ and temperature $\beta = 2/3$, giving saturation thresholds $\approx \pm 2.398$ and a sigmoid offset of ≈ 1.60 in Eqs. (6)–(8). Gate parameters are initialized at $\alpha_e \sim \mathcal{N}(2.2,\ 0.01^2)$, which places each gate just below the upper saturation threshold ($g_e \approx 0.98$) with a small perturbation that breaks symmetry across the N_t gates. After each backward pass, α_e (and $\tilde{\alpha}_{e,j}$ when RCap is enabled) is clamped to $[-4.605,\ 4.605]$ so that saturated gates do not drift back within a task. Gate parameters form a separate optimizer group with a higher learning rate than the LoRA weights. The ℓ_0 penalty is held at zero for the first 50% of training, so that the language-modeling gradient establishes useful expert representations before pruning pressure is applied.

Table 6: Performance across different model sizes on CoIN.

Size	Method	Accuracy on Each Task (%)								Aggregate Results (%)			
		SQA	VQA^{Text}	ImgNet	GQA	VizWiz	REF	VQA^{v2}	VQA^{OCR}	MFT↑	MFN↑	MAA↑	BWT↑
7B	DyCapMoE	79.75	54.51	97.55	56.95	59.20	52.10	61.55	61.60	65.67	65.40	70.60	-0.30
13B	DyCapMoE	82.45	56.26	97.45	60.30	61.55	55.45	65.90	66.20	68.52	68.20	72.23	-0.36

Table 7: Comparison on the UCIT benchmark with LLaVA-1.5-7B.

Method	Accuracy on Each Task (%)						Aggregate Results (%)			
	ImgNet-R	ArxivQA	VizWiz	IconQA	CLEVR	Flickr30k	MFT↑	MFN↑	MAA↑	BWT↑
Without task-level prior at inference (w/o TR)										
LoRA-FT	58.03	77.63	44.39	67.40	61.77	58.22	76.89	61.24	76.55	-18.78
O-LoRA	77.50	78.07	44.50	63.13	64.73	58.16	76.01	64.35	78.02	-13.99
MoELoRA	70.07	77.70	44.69	50.03	54.03	57.34	71.17	58.98	75.08	-14.63
SEFE	80.83	78.00	47.01	69.63	65.83	57.92	75.98	66.54	78.76	-11.33
DyCapMoE	78.43	87.30	49.61	74.53	61.00	57.33	74.93	68.03	79.83	-8.28
With task-level prior at inference (TR)										
ModalPrompt	74.43	92.00	55.92	44.27	53.97	43.67	60.76	60.71	70.57	-0.05
CL-MoE	66.33	77.00	44.78	51.87	53.53	57.42	71.46	58.49	74.19	-15.56
DISCO	87.43	93.07	46.96	68.13	65.70	56.69	75.87	69.66	81.60	-7.45
HiDE	84.03	90.73	44.43	58.93	41.37	54.25	69.96	62.29	77.32	-9.20
DyCapMoE	90.10	92.70	61.65	79.10	68.57	57.31	74.88	74.91	82.86	0.03

Table 8: Comparison on the DCL benchmark with LLaVA-1.5-7B.

Method	Accuracy on Each Task (%)					Aggregate Results (%)			
	RS	Med	AD	Sci	Fin	MFT↑	MFN↑	MAA↑	BWT↑
Without task-level prior at inference (w/o TR)									
LoRA-FT	69.65	41.59	25.43	40.88	87.45	64.98	53.00	61.13	-14.97
O-LoRA	74.64	44.42	30.02	41.47	87.15	65.16	55.54	62.12	-12.53
MoELoRA	77.54	41.85	27.62	40.13	86.75	64.94	54.78	61.76	-12.71
SEFE	77.26	50.37	37.21	40.87	86.82	65.01	58.51	63.63	-8.13
DyCapMoE	80.30	62.60	53.25	51.70	90.75	67.38	67.72	68.64	0.43
With task-level prior at inference (TR)									
ModalPrompt	78.67	51.37	47.80	43.29	87.78	61.84	61.78	64.05	-0.07
CL-MoE	71.34	46.84	26.33	41.17	88.74	66.06	54.88	61.79	-13.97
HiDE	74.31	48.95	33.21	38.54	81.55	60.77	55.31	60.68	-6.82
DISCO	76.49	44.48	44.84	46.61	89.22	64.78	60.33	63.93	-5.57
DyCapMoE	73.90	65.10	54.30	52.95	90.75	67.38	67.40	67.36	0.03

Attribution protocol. The rows of Table 4 share the backbone, the optimizer and the step schedule and differ only in the capacity mechanism. The fixed budgets use N_t=16 experts at rank 1 (r=16) or rank 2 (r=32), and N_t=8 experts at rank 1 (r=8). For O-LoRA (App. H.4) we keep its released training configuration and attach RCap gates to the LoRA bottleneck of each task's adapter (Eq. 7) with the same $\lambda_{\ell_0}^{\text{rank}}$=0.5 and gate schedule; ECap does not apply because O-LoRA has no experts. For the data-matched comparison with SEFE (App. L) we train DyCapMoE with the router disabled on the data released by SEFE (Chen et al., 2025) and change nothing else.

F Full per-task results on UCIT and DCL

Tables 7 and 8 give the per-task results behind Table 2 (LLaVA-1.5-7B), and Tables 9 and 10 give the corresponding results under InternVL-Chat-7B. The five DCL domains are remote sensing, medicine, autonomous driving, science and finance.

Table 9: Comparison on the UCIT benchmark with InternVL-Chat-7B.

Method	Accuracy on Each Task (%)						Aggregate Results (%)			
	ImgNet-R	ArxivQA	VizWiz	IconQA	CLEVR	Flickr30k	MFT↑	MFN↑	MAA↑	BWT↑
Without task-level prior at inference (w/o TR)										
LoRA-FT	76.50	78.60	45.57	71.13	61.63	58.82	79.20	65.38	80.66	-16.59
O-LoRA	90.30	93.53	46.26	68.40	65.17	56.06	70.97	69.95	81.23	-1.22
MoELoRA	68.57	80.47	45.43	69.23	65.93	58.57	79.25	64.70	80.07	-17.46
SEFE	85.93	76.90	47.42	69.77	63.37	58.19	73.55	66.93	79.44	-7.94
DyCapMoE	83.03	82.97	51.44	75.43	69.23	57.12	75.74	69.87	81.27	-7.05
With task-level prior at inference (TR)										
ModalPrompt	57.10	63.90	42.11	37.40	47.50	45.76	48.98	48.96	53.52	-0.02
CL-MoE	74.23	78.77	44.77	53.90	72.73	58.23	71.39	63.77	77.44	-9.15
HiDE	90.00	93.77	50.49	71.03	61.10	55.08	76.29	70.25	82.97	-7.25
DISCO	92.13	94.13	48.11	73.90	67.53	58.06	78.92	72.31	84.24	-7.93
DyCapMoE	94.00	92.63	61.16	75.97	71.13	56.93	75.62	75.30	85.43	-0.38

Table 10: Comparison on the DCL benchmark with InternVL-Chat-7B.

Method	Accuracy on Each Task (%)					Aggregate Results (%)			
	RS	Med	AD	Sci	Fin	MFT↑	MFN↑	MAA↑	BWT↑
Without task-level prior at inference (w/o TR)									
LoRA-FT	69.93	52.17	33.04	42.67	91.07	69.06	57.78	65.22	-14.11
O-LoRA	74.48	54.16	39.60	48.30	88.54	65.51	61.02	65.83	-5.62
MoELoRA	69.90	52.08	33.17	42.19	90.58	68.83	57.58	65.97	-14.06
SEFE	78.21	57.59	51.45	44.65	91.37	69.55	64.65	68.84	-6.12
DyCapMoE	79.90	56.10	53.80	48.95	91.15	66.81	65.98	69.32	-1.04
With task-level prior at inference (TR)									
ModalPrompt	50.20	30.95	14.11	33.91	65.45	38.93	38.92	38.70	-0.01
CL-MoE	78.12	52.51	35.53	42.69	91.24	69.22	60.02	67.60	-11.51
HiDE	75.40	57.66	36.73	41.48	88.59	65.26	59.97	65.94	-6.61
DISCO	77.90	47.50	49.13	49.37	90.92	68.55	62.96	67.81	-6.98
DyCapMoE	81.05	58.30	54.05	49.45	91.10	67.01	66.79	70.05	-0.27

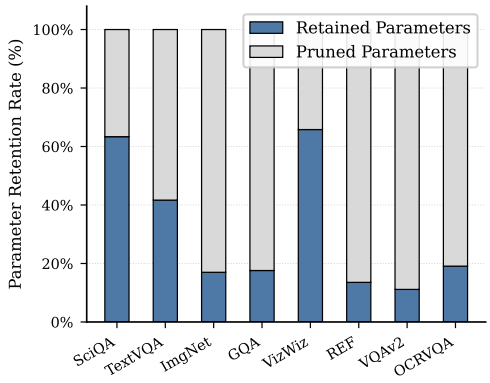

Figure 5: Per-task parameter retention rate on **CoIN** (blue: retained, gray: pruned). Retention spans $\sim$11% (VQAv2) to $\sim$66% (VizWiz), the widest range across all benchmarks, reflecting the high task diversity of CoIN.

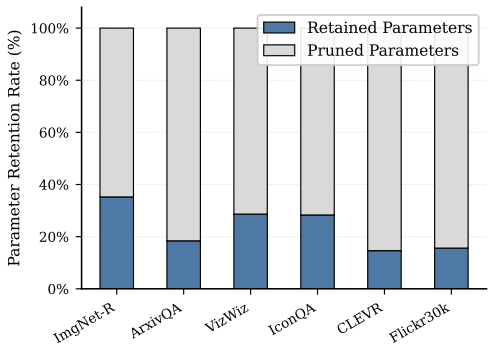

Figure 6: Per-task parameter retention rate on **UCIT**. Retention spans $\sim$15% (CLEVR, Flickr30k) to $\sim$35% (ImgNet-R), showing that tasks differ substantially in how much new capacity they require.

G Per-task surviving capacity on both axes

Figs. 5 and 6 show the per-task parameter retention rate on CoIN and UCIT, complementing the DCL counterpart (Fig. 3) in the main text. Figs. 7, 8, and 9 provide the per-(layer, expert) rank heatmaps for all three benchmarks.

H Sensitivity and robustness analyses

H.1 ℓ_0 weight sweep

Table 11 sweeps $\lambda_{\ell_0}^{\text{expert}}$ and $\lambda_{\ell_0}^{\text{rank}}$ jointly over a $10\times$ range, with a default of 0.5 on each (task router disabled). Raising both weights from 0.1 to 1.0 lowers the retained fraction monotonically ($25.12\% \to 21.38\%$) while MFN stays between 67.02 and 68.03 and BWT improves monotonically ($-8.57 \to -7.18$). The weight therefore sets the overall pruning strength, whereas the per-task and per-module pattern of retention is set by the data (Figs. 5 and 6, App. G). We adopt $\lambda_{\ell_0}^{\text{expert}} = \lambda_{\ell_0}^{\text{rank}} = 0.5$ as the default.

Table 11: ℓ_0 weight sweep on UCIT (task router disabled). MFT/MFN/MAA/BWT (%) and the retained fraction of the allocated rank budget (Rate, the fraction in parentheses in Table 4).

$\lambda_{\ell_0}^{\text{expert}}$	$\lambda_{\ell_0}^{\text{rank}}$	MFT↑	MFN↑	MAA↑	BWT↑	Rate↓
0.1	0.1	74.37	67.02	78.93	-8.57	25.12%
0.5	0.5	74.93	68.03	79.83	-8.28	23.48%
1.0	1.0	74.27	67.09	79.25	-7.18	21.38%

H.2 Gate initialization

The initialization trades off two effects. We start every new expert almost fully open ($g_e \approx 0.98$) so that removal is decided by the language-modeling gradient. However, the gradient reaching a gate weakens as α_e rises, since fewer Hard-Concrete draws of Eq. (3) leave the gate below its upper clip and the ℓ_0 term flattens. The default $\alpha_e = 2.2$, just under the deterministic saturation threshold $\log((1-\gamma)/(\zeta-1)) \approx 2.398$, satisfies both. Table 12 varies only α_e on UCIT with the router disabled and $\lambda_{\ell_0}^{\text{expert}} = \lambda_{\ell_0}^{\text{rank}} = 0.5$. A higher initialization keeps more capacity (20.56% to 25.73%), because the gate starts further from closure and receives less pruning pressure. MFN varies by 0.82 points across the three settings and is highest at the default, so the initialization mainly changes how much capacity is kept, not accuracy. At $\alpha_e = 3.0$ the deterministic gate is already saturated at 1 and only the stochastic draws can close it, yet 74% of the budget is still pruned. Together with the λ_{ℓ_0} sweep, this shows that the hand-set gate settings determine the overall level of retention, while its distribution across tasks and modules is determined by the data.

Table 12: Sensitivity to the gate initialization α_e on UCIT (LLaVA-1.5-7B, task router disabled; one run per setting).

α_e init	1.5	2.2 (default)	3.0
Retained capacity (%)↓	20.56	23.48	25.73
MFN↑	67.21	68.03	67.53

H.3 Task order

Because capacity demand depends on the accumulated expert pool (Sec. 4.3), the capacity a task retains can depend on its position in the sequence. Table 13 reruns CoIN under two further task orders (the reverse of the original order, and alphabetical order of the task names), holding the backbone, the schedule and every hyperparameter fixed and the task router disabled. Retained capacity changes with order (27.97%–31.15%), since a task learned later can reuse more of the existing experts and keeps less. Final accuracy changes little. MFN spans 0.39 points and BWT spans 0.72 points.

Table 13: Task-order robustness on CoIN (LLaVA-1.5-7B, task router disabled; one run per order). Retained capacity is the surviving fraction of the allocated rank budget.

Task order	MFN↑	MAA↑	BWT↑	Retained cap.↓
Original (Table 1)	58.22	61.22	-9.26	31.15%
Reverse	58.31	61.65	-8.63	27.97%
Alphabetical	57.92	61.13	-9.35	29.75%

H.4 RCap on a non-MoE adapter

RCap acts on the rank axis of a LoRA bottleneck and does not presuppose an expert pool. To check that its effect is not tied to the IncMoE-LoRA structure, we attach RCap gates to O-LoRA (Wang et al., 2023b), which trains one adapter per task under an orthogonality constraint and has no expert pool. Its released training configuration is kept and only the rank-level pruning is added (App. E); ECap does not apply because O-LoRA has no experts. Table 14 reports UCIT results with the task router disabled. With RCap, O-LoRA keeps 31.56% of the $r{=}32$ budget and improves on the $r{=}32$ model it starts from by 0.65 MFN and 1.14 BWT, and it exceeds the fixed $r{=}16$ model by 2.34 MFN at less capacity. This is the same pattern as for DyCapMoE in Table 4.

Table 14: RCap transferred to O-LoRA on UCIT (LLaVA-1.5-7B, task router disabled). A fixed budget stores the rank it allocates; O-LoRA + RCap allocates $r{=}32$ and prunes during training. Stored rank as in Table 4. Best per column in bold.

Method	Stored rank↓	MFN↑	BWT↑
Fixed budget (O-LoRA)			
$r{=}16$	16	64.55	-14.03
$r{=}32$	32	66.24	-14.31
Learned budget (O-LoRA + RCap, allocated $r{=}32$)			
O-LoRA + RCap	**10.10** (31.56%)	**66.89**	**-13.17**

I Run-to-run variance on the multimodal benchmarks

We reran DyCapMoE three times on each multimodal benchmark in both configurations (LLaVA-1.5-7B), with seeds differing only in initialization and data order. Baselines were not rerun. Table 15 reports mean ± std, and the DyCapMoE rows in Tables 1 and 2 show the first seed. The largest standard deviation in the table is 0.63 points, compared with differences of 2.05 MFN for

the capacity mechanism on UCIT (68.32 vs. 66.27 for the fixed-budget baseline) and 6.36 to 7.16 MFN for the task router. The effect on forgetting is larger and has a smaller spread: BWT moves from -9.15 ± 0.56 to -0.30 ± 0.01 on CoIN. Routing does not help on DCL, where MFN changes by -0.62 (seed standard deviation 0.37). DCL already has BWT ≈ 0.4 without routing, so there is little forgetting for the router to remove. TRACE results (Table 3) are averaged over three runs following Qian et al. (2025).

Table 15: Mean $\pm$ std over three seeds for DyCapMoE (LLaVA-1.5-7B). Baselines in Tables 1 and 2 are single runs.

	CoIN			UCIT			DCL		
	MFN	MAA	BWT	MFN	MAA	BWT	MFN	MAA	BWT
w/o task router	58.24 ± 0.62	61.31 ± 0.55	-9.15 ± 0.56	68.32 ± 0.37	79.95 ± 0.60	-8.36 ± 0.50	68.11 ± 0.37	68.75 ± 0.52	0.39 ± 0.05
w/ task router (ours)	65.40 ± 0.38	70.82 ± 0.48	-0.30 ± 0.01	74.68 ± 0.41	82.98 ± 0.63	0.03 ± 0.01	67.49 ± 0.35	67.86 ± 0.47	0.03 ± 0.01

J COMPUTATIONAL COST: PROTOCOL AND GATE-LEVEL BREAKDOWN

Protocol. Both runs in Table 16 are LLaVA-1.5-7B on CoIN, trained on the same node with $4\times$H100, DeepSpeed ZeRO-2 in bf16, and micro-batch 32 with gradient checkpointing. Both start from the same allocation of $N_t{=}16$ rank-4 experts per task and follow an identical 4,487-step schedule, so only the capacity mechanism differs. Peak memory is the maximum over the four ranks at task 8, where the expert pool is largest. FLOPs are per token of the LLM forward pass for the dense-materialised MoE branch we deploy, alone and with the frozen backbone added. They are computed as two FLOPs per stored parameter, so the MoE-branch reduction equals the stored-parameter reduction by construction. Inference speed is per sample on one H100 (Intel Xeon Platinum 8452Y host), batch size 1 with generation included, over 2,000 CoIN VQA^{OCR} test samples. Stored parameters count the LoRA factors and router columns of the deployed adapter and exclude the frozen backbone.

Table 16: Computational cost on CoIN (LLaVA-1.5-7B) against the fixed-budget IncMoE-LoRA baseline under an identical schedule. FLOPs: per LLM forward token, dense-materialised MoE branch (whole LLM in parentheses).

Method	Train time	Peak memory	Stored params↓	Latency↓	FLOPs/token, MoE (LLM)↓
IncMoE-LoRA (fixed budget)	7.65 h	33.05 GiB	1425.0 M	0.39 s	2.850 G (16.67 G)
DyCapMoE (ECap+RCap)	7.82 h (+2.2%)	33.05 GiB (+1 MiB)	**457.8 M** (3.11×)	**0.37 s**	**0.915 G** (14.74 G)

Where the training overhead comes from. Table 17 isolates the gates. The 2.2% wall-clock increment comes from three sources: the stochastic Hard-Concrete sampling in the forward pass, the ℓ_0 term, and the per-step clamp on the gate locations. The gate parameters add 0.14 M scalars, under 0.1% of the LoRA trainable parameters. The gates decide during training which units to close, so the step at task end only applies this decision with a single tensor slice, which takes negligible time compared with training a task. Peak memory does not change because the gates are learned jointly with the adapter: unused experts are masked during training and are removed only from the saved checkpoint. The ℓ_0 term is a function of the gate parameters rather than of the activations, so its memory cost stays at 0.14 M scalars regardless of batch size and sequence length. The one activation-sized term we add, the rank-gate broadcast, is confined to the gradient-checkpointing window and sits 2.09 GiB below the step's high-water mark. The task router adds no trained parameter and no second forward pass.

Savings at deployment. The deployed MoE branch keeps 31.15% of the rank budget on CoIN, 457.76 M stored parameters against 1425.01 M, a 3.11× reduction, and MoE-branch FLOPs drop by the same factor. Since the branch is only 17% of the LLM's per-token FLOPs, the saving is mostly in storage, and throughput improves only slightly, from 0.39 to 0.37 s per sample in single-sample generation. For comparison, PCLR's deployed adapter on CoIN stores 766.70 M parameters (their App. Q; both counts exclude the frozen backbone). Under the fixed budget the adapter grows by about 178 M parameters per task and reaches a fifth of the 7B backbone after eight tasks; DyCapMoE adds 57 M per task on average.

Table 17: Gate-level overhead on CoIN (LLaVA-1.5-7B), identical 4,487-step schedule.

	Total train time	Peak train memory	Added trainable params
Fixed-budget IncMoE-LoRA	$\sim$7.65 h	33.046 GiB	0
DyCapMoE (ECap+RCap)	$\sim$7.82 h ($+2.2\%$)	33.046 GiB ($+1$ MiB)	$+0.14$ M

K Routing accuracy and robustness under overlapping tasks

Routing accuracy. Table 18 reports the per-task routing accuracy of the prototype router on CoIN in the deployed configuration (LLM input-token text anchor, vision-tower image anchor, hard top-1 selection). Overall accuracy is 89.9% (unweighted mean over the eight tasks). GQA and VQA^{v2} have near-identical anchors (cosine similarity 0.996) and account for most misrouted samples. VQA^{v2} is also the task that retains the least new capacity ($\sim 11\%$, Fig. 5), that is, it adds little beyond the experts already present, so confusing the two costs little accuracy. Under correct routing the layer outputs match those at the end of that task's training, so a perfect router gives BWT exactly 0 (Prop. A.2). At 89.9% routing accuracy the router reaches MFN 65.40 against 66.32 with ground-truth task IDs, so a 10.1% error rate costs 0.92 points.

Table 18: Per-task routing accuracy (%) of the prototype router on CoIN (LLaVA-1.5-7B, deployed configuration). Tasks are listed in training order.

	SQA	VQA^{Text}	ImgNet	GQA	VizWiz	REF	VQA^{v2}	VQA^{OCR}	Overall
Routing acc.	100.0	87.2	100.0	58.8	97.1	100.0	76.4	99.5	89.9

Misrouting is one-sided. The mask suppresses only experts whose index is strictly greater than the predicted task (Eq. 9). Over-prediction therefore leaves intact every expert that existed when the task finished training, so its worst case is the no-router configuration of Tables 1 and 2. Only under-prediction (routing to a task with a lower index) removes capacity the sample uses. On CoIN the 10.1% total error splits into 7.0 points in the benign direction and 3.2 points in the harmful one. Which direction is harmful depends on the task order (GQA is task 4 and VQA^{v2} task 7 in the original order).

Training does not assume separability. A task is the batch of data that arrives at one time, and its anchor is the mean of that batch's embeddings. Training uses only the signal that a new batch has begun, which comes from the deployment schedule and not from the data. Only the inference-time router relies on separability (Table 19).

Blurry task boundaries. In the blurry-M setting (the $N{=}0$ case of i-Blurry (Koh et al., 2021), with each source dataset in place of a class), each task gives up a random $M\%$ of its own samples and receives the same number back from the other seven, so task sizes and the corpus are preserved and the realized blur is exactly M. Architecture, mask and inference settings are fixed, and only M varies (one run per M). Table 19 shows that routing accuracy falls from 89.9% to 85.6% at $M{=}30\%$ while the gap to oracle routing narrows from 0.92 to 0.42. At $M{=}30\%$ most VQA^{v2} samples are routed to lower-index tasks, yet the model stays within 0.42 points of the oracle. Two effects explain the smaller gap. Tasks that the router confuses tend to have experts that can replace each other, so a misroute costs less as overlap grows. The exchanged samples also expose every task to data from the others, which by itself raises the no-router accuracy (58.22 to 62.80 at $M{=}10\%$) and reduces the router's margin (7.18 to 0.74 points). The router therefore helps most when task boundaries are sharp, and its gap to the oracle is smallest when they are blurred.

L Comparison with SEFE and PCLR

Data-matched comparison with SEFE. SEFE trains a single LoRA with an importance-matrix constraint and has no task-allocated expert groups for a task-level prior to act on, so we compare it

Table 19: Blurry-M CoIN (LLaVA-1.5-7B): routing accuracy and MFN without the router, with the prototype router, and with ground-truth task IDs; one run per setting.

M	Routing acc. (%)	MFN w/o router	MFN w/ our router	MFN w/ oracle task ID
0% (original CoIN)	89.9	58.22	65.40	66.32
10%	89.0	62.80	63.71	64.16
30%	85.6	62.21	62.95	63.37

with DyCapMoE with the router disabled. To control the training data, we retrain DyCapMoE on the data released by SEFE and change nothing else. Table 20 summarizes the comparison. On the CoIN release, DyCapMoE is within 0.35 MFN of SEFE. On SEFE's release it leads by 3.16 MFN and its BWT improves from -9.26 to -4.38, so the gap in Table 1 comes from the training data and not from the method.

Table 20: Data-matched comparison with SEFE on CoIN (LLaVA-1.5-7B, task router disabled).

Method	Training data	MFN↑	MAA↑	BWT↑
SEFE	SEFE release	58.57	63.04	-10.45
DyCapMoE w/o TR	CoIN	58.22	61.22	-9.26
DyCapMoE w/o TR	SEFE release	61.73	62.16	-4.38

Comparison with PCLR. PCLR (Meng et al., 2026) splits each LoRA adapter into rank vectors and attaches a learnable key to every rank. At inference, a query is formed from frozen features, the mean-pooled LLM token embeddings and the visual [CLS] token, and the r ranks whose keys are most similar to the query are activated together with a set of shared ranks. The keys are trained with a query–key cosine loss, which is minimized when each key equals the mean query of its task (their Sec. 4.3). At convergence, the ranks of a task therefore share that task's prototype, and the top-r selection reduces to a prototype-based task prior inside every layer. Since our task router is built from the same frozen features, the two methods belong to the same family and can be compared directly, which is why PCLR appears in the TR block of Table 1. DyCapMoE realizes this prior by forming the prototype in closed form, sharing it across layers, learning no additional parameters, and applying it as a task-level mask, so that an input is served by the experts of one task and by none from later tasks. PCLR instead learns one key per rank in each layer and selects ranks one at a time, so the ranks serving an input may belong to several tasks, especially after compression has left a task with fewer than r ranks. The two methods also set capacity differently. PCLR fixes a total rank budget (256, of which 64 are activated and 32 shared); once the budget is full, a preset fraction of rank experts is pruned after every task (100% retention for the first three tasks, then 75%, then 87.5%, raised when a distillation loss exceeds a threshold; their Apps. B and J). Pruning involves no training, and an integration phase then distills the pool before compression into the pool after it, using the current task's data and updating the shared ranks and keys. In DyCapMoE, retention is learned end to end from a single λ_{ℓ_0}, decided per expert and per rank direction, and varies from 11% to 66% across CoIN tasks (Fig. 5); the parameters of earlier tasks remain unchanged. On CoIN, DyCapMoE is 3.21 MFN above PCLR, and its BWT of -0.30, against -3.39, follows from the task-level mask and the frozen earlier parameters. In storage, PCLR grows to 766.7 M adapter parameters by the fifth task and stays at that size (their App. Q), whereas DyCapMoE holds 457.8 M after all eight tasks, 40% fewer. Neither method replays past data; PCLR's integration uses only the data of the current task.

M Extended Related Work

This appendix gives the full discussion of related work summarized in Sec. 2.

Continual Learning (CL) investigates how a model can acquire knowledge from a stream of non-stationary tasks while mitigating catastrophic forgetting (Li & Hoiem, 2017; De Lange et al., 2021; Wang et al., 2024a). Existing CL methods are commonly grouped by their strategy: (i) rehearsal-based methods replay stored or generated samples of previous tasks (Rebuffi et al., 2017; Lopez-Paz

& Ranzato, 2017; Chaudhry et al., 2019; Buzzega et al., 2020; Shin et al., 2017; Rostami et al., 2019; Riemer et al., 2019; Tong et al., 2025); (ii) regularization-based methods penalize updates to parameters important for prior tasks (Kirkpatrick et al., 2017; Zenke et al., 2017; Nguyen et al., 2017; Li & Hoiem, 2017; Aljundi et al., 2018; Zhang et al., 2020; 2023a); (iii) architecture-based methods allocate task-specific parameter subsets, by masking (Serra et al., 2018; Ke et al., 2021; Wortsman et al., 2020; Mallya et al., 2018) or by expanding the network with dedicated modules (Yoon et al., 2018; Xu & Zhu, 2018; Hung et al., 2019; Li et al., 2019; Yan et al., 2021; Wang et al., 2025a). Among expansion methods, DEN shrinks each task's added neurons with group-Lasso regularization (Yoon et al., 2018), RCL sizes the growth with a reinforcement-learning controller (Xu & Zhu, 2018), CPG grows only when an accuracy target is missed (Hung et al., 2019), and DER prunes added backbones with learned channel masks (Yan et al., 2021); these adapt growth for CNN-era models trained from scratch, and none transfers to LoRA-expert pools over frozen pre-trained backbones, where a pruning mistake is permanent because the data cannot be revisited. Recent CL research increasingly exploits large pre-trained models (PTMs) with parameter-efficient fine-tuning (PEFT) (Wang et al., 2022c; Liang & Li, 2024; Zou et al., 2025; Yu et al., 2024; Wei et al., 2025), sometimes drawing on auxiliary pre-trained models for semantic priors (Khan et al., 2023; Ma'sum et al., 2025).

Continual Learning for LVLMs and LLMs. Retraining LLMs (Touvron et al., 2023; Jiang et al., 2023; Team, 2024) and LVLMs (Liu et al., 2023; Wang et al., 2024b; Bai et al., 2025; Comanici et al., 2025) from scratch is expensive, so extending them continually is of practical interest. MCIT formalizes this goal as incremental adaptation of LVLMs to a stream of instruction-tuning tasks, with several benchmarks (He et al., 2023; Chen et al., 2024a; Guo et al., 2025a; Zhao et al., 2025b). Existing approaches mirror the taxonomy above: replay- and regularization-style methods protect critical subnetworks or exploit task similarity (Zhu et al., 2024; He et al., 2023; Qiao et al., 2025; Chen et al., 2025; Meng et al., 2026), while architecture-based methods allocate fresh parameters per task, most commonly via LoRA (Hu et al., 2022). Within this family, MoELoRA (Chen et al., 2024a) and its descendants grow a pool of LoRA experts across tasks, and routing variants refine how the per-task expert subset is identified (Yu et al., 2025; Zhang et al., 2025; Zhao et al., 2025a; Ge et al., 2025b; Wang et al., 2025c; Huai et al., 2025; Zhao et al., 2026). Three prior works define the line we continue, and we state the differences explicitly. DyMoE diagnoses forgetting in the frozen IncMoE-LoRA pool as routing drift and regularizes token assignment during training (Zhao et al., 2026); it fixes the per-task budget and leaves inference unguided. SEMA decides *when and where* to expand through representation-shift detectors (Wang et al., 2025a). MoRAM fixes rank-1 expert granularity and removes the router through self-activating associative memories (Lu et al., 2025). DyCapMoE addresses what all three leave open: it learns *how much* to expand, on both axes, and supplies the inference-time task prior without training anything.

MoE with LoRA and adaptive capacity. MoE scales capacity by activating a sparse subset of experts per input (Shazeer et al., 2017; Lepikhin et al., 2020; Fedus et al., 2022; Dai et al., 2024; Yang et al., 2025); combined with LoRA (Hu et al., 2022), each expert is a low-rank adapter, widely used for fine-tuning (Chen et al., 2023; Li et al., 2024a; Gao et al., 2024; Dou et al., 2024) and for CL (Chen et al., 2024a; Huai et al., 2025; Yu et al., 2025; Zhang et al., 2025; Zhao et al., 2025a; Ge et al., 2025b); freezing old experts and gates is also supported theoretically (Li et al., 2025). Deciding the size of such modules has been studied outside CL: AdaLoRA prunes singular directions under a decaying global budget (Zhang et al., 2023b), SoRA sparsifies a gate vector on the LoRA bottleneck with ℓ_1 proximal steps (Ding et al., 2023), DynMoE auto-tunes the number of active experts during training (Guo et al., 2024), and ℓ_0/Hard-Concrete gating underlies structured pruning at multiple granularities (Louizos et al., 2018; Xia et al., 2022). These methods can revisit their data to re-estimate importance; CL cannot, so a budget mistake is permanent, and global budget schedulers assume one stationary run rather than a stream of task-end pruning events. Closest to us in CL, PCLR compresses rank-level experts after each task under hand-set retention rates and recovers accuracy by distillation (Meng et al., 2026); GoD-MoE makes a binary add-or-reuse decision from a pre-training detector on a data sample (Zhang et al., 2026); D-MoLE distributes a *fixed* per-task budget across layers with zero-cost proxies (Ge et al., 2025a); OA-Adapter adapts a serial adapter's bottleneck dimension for CL of LLMs (Wan et al., 2025); LLaVA-CMoE decides when and where to add experts with probe stages (Zhao et al., 2025a). DyCapMoE differs from all of these in learning the per-task expert count and per-expert rank end-to-end with the task loss, via ℓ_0 gates pruned at task end, with no detector, proxy, schedule, or distillation phase.

Task-level routing and task-identity inference. Inferring the task of a test input is an established sub-problem of CL (Wang et al., 2023a): nearest-prototype rules date to iCaRL (Rebuffi et al., 2017), S-Prompts selects per-domain prompts by nearest centroid of frozen features (Wang et al., 2022a), and prototype classifiers over frozen PTM features rival trained CL methods (Zhou et al., 2024). In MoE-based CL, however, existing task-level guidance is trained and heavy: extra forward passes over deep hidden states (Zhao et al., 2025a), learned task routers (Huai et al., 2025; Zhao et al., 2025b), or attentive selection modules (Zhao et al., 2024). Our prototype task router brings the training-free prototype rule to the incremental expert pool: dual multimodal anchors built from by-products of the forward pass, and a hard mask over later-task expert groups that, under parameter isolation, bounds backward transfer by the routing error rate (Prop. A.2).

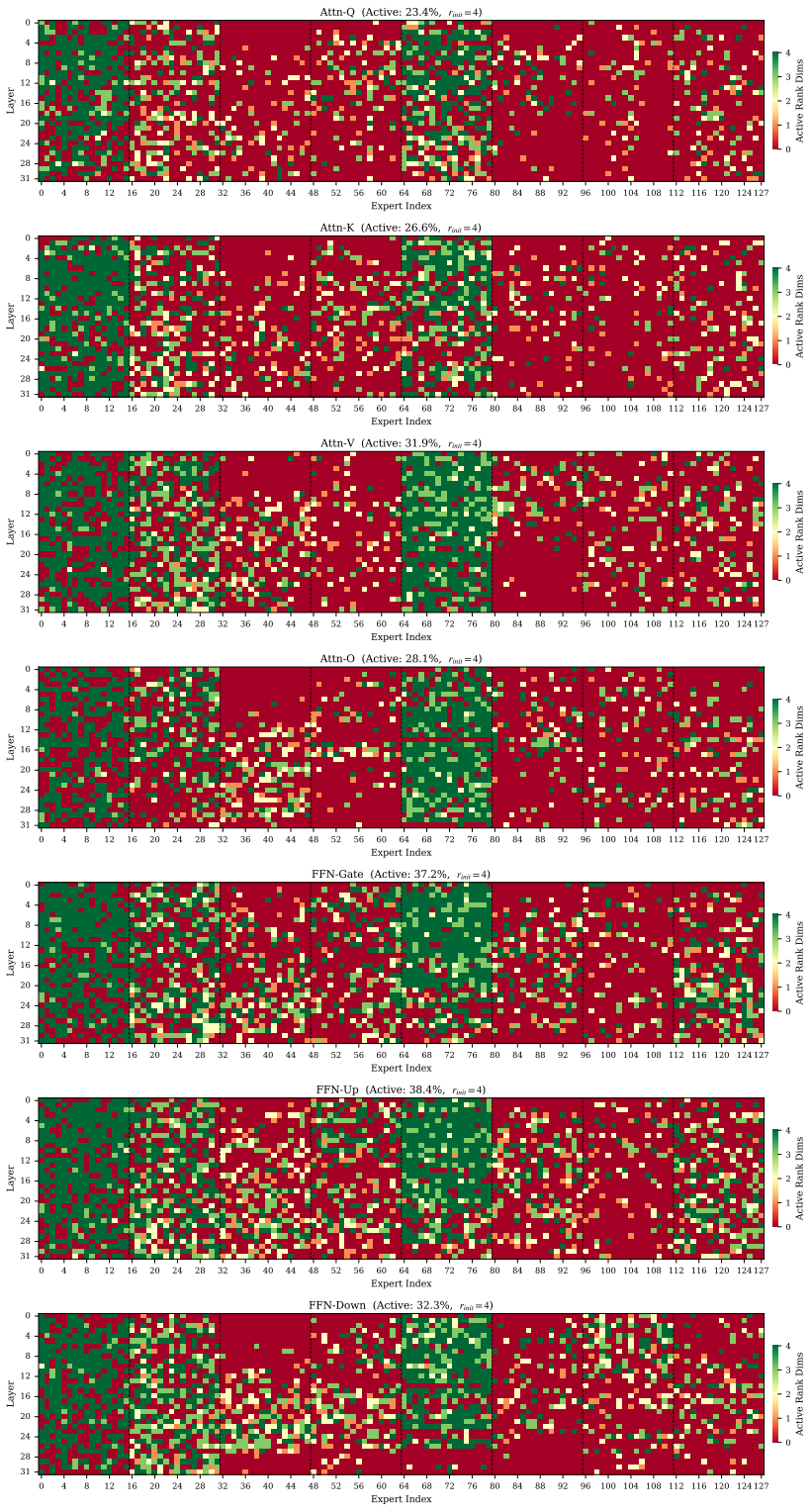

Figure 7: Active rank dimensions per (layer, expert) cell after sequential training on **CoIN** (r=4, 128 experts). Each cell reports the number of surviving rank directions for expert e at layer l, where red indicates full pruning and green indicates all r dimensions retained. Dashed vertical lines mark task boundaries. The active ratio in each subplot title aggregates over all (layer, expert) cells within that module. FFN-Gate and FFN-Up retain the most capacity (37.2% and 38.4%), while Attn-Q is the most aggressively pruned (23.4%).

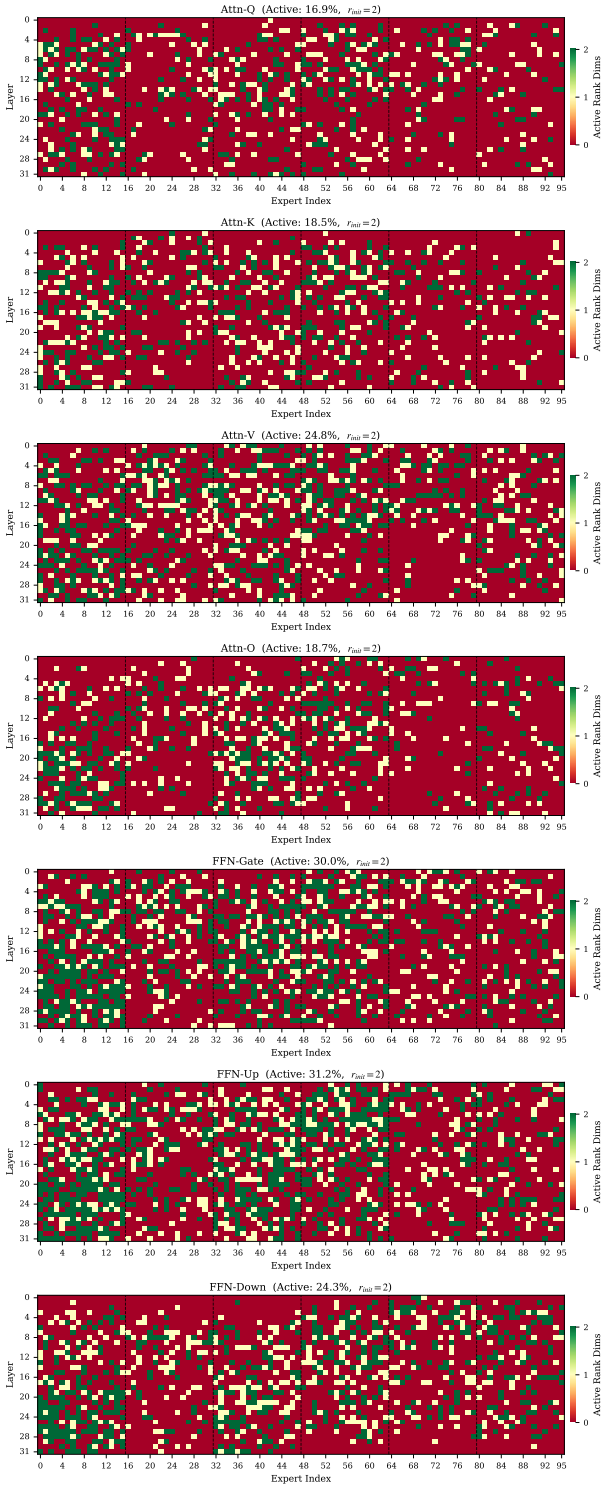

Figure 8: Same visualization as Fig. 7 on **UCIT** (r=2, 96 experts). With a smaller per-expert rank budget, FFN-Gate and FFN-Up again retain the most capacity (30.0% and 31.2%), while Attn-Q is the most aggressively pruned (16.9%).

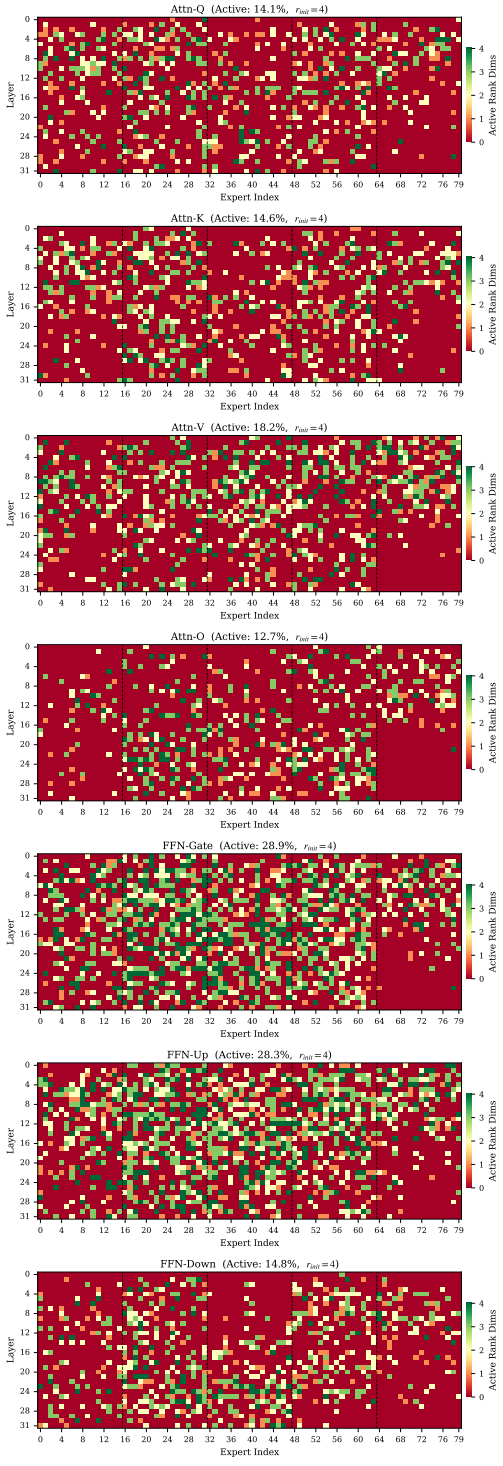

Figure 9: Same visualization as Fig. 7 on **DCL** (r=4, 80 experts). Pruning is more aggressive than CoIN overall, with Attn-O reaching 12.7% and FFN-Gate retaining up to 28.9%.