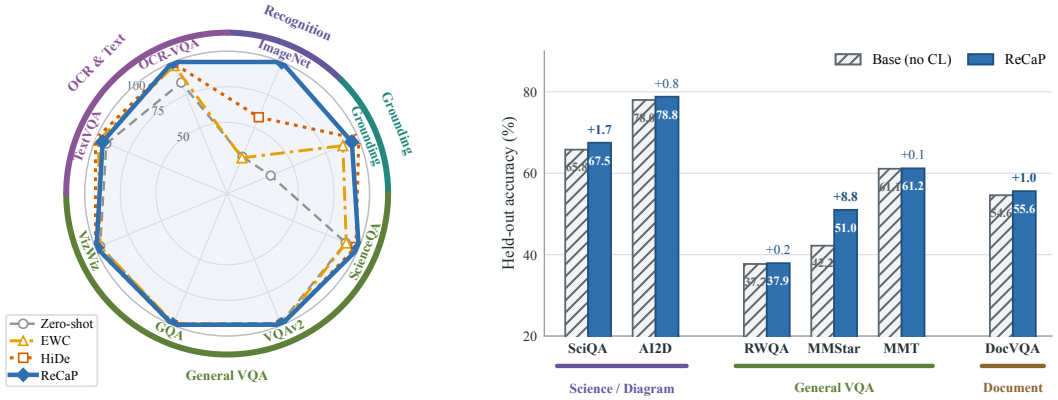
RECURSIVE CONTINUAL POST-TRAINING OF FOUNDATION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Foundation models should be able to learn new tasks after deployment without forgetting old ones or becoming larger over time. Current approaches tend to preserve one at the expense of the other: self-distillation keeps the model size fixed, but is limited by the model's own capabilities; expert routing protects past tasks with dedicated frozen experts, but every new task makes the deployed model larger and more expensive to run. We present Recursive Continual Post-Training (ReCaP), which captures the benefits of frozen experts without deploying them. At each stage, ReCaP trains an expert on the new task, then distills it back into a single base-sized checkpoint. For prior tasks, a small per-task generator reconstructs input embeddings, which the corresponding frozen expert relabels to recover its behavior in the student. These frozen experts provide stable distillation targets across later stages. On CoIN and UCIT-O, ReCaP reaches 78.0% and 75.5% average accuracy, respectively, matching or outperforming the strongest expert-routing baselines. It also improves average performance across six held-out benchmarks by 2.1 points over the base model, with no benchmark dropping. Similar results hold across Qwen and Llama models from 1B to 7B, and across both vision-language and text-only task sequences.

1 INTRODUCTION



(a) Learns the stream (b) Preserves general capability

Figure 1: **ReCaP learns a task sequence in one checkpoint while preserving general ability. Left:** Final accuracy on the eight CoIN tasks. Each axis is normalized to the best of the four shown methods, so results closer to the outer ring are better. ReCaP averages 78.0 and reaches 92.5 on ImageNet, compared with 53.7 for HiDe and 25.1 for EWC. **Right:** Across six held-out benchmarks, ReCaP improves the base-model average from 56.6 to 58.7 (+2.1), with no decrease on any benchmark.

Foundation models often need new skills or domain knowledge after deployment, but retraining the full model for every update is costly. Continual post-training instead updates a single model over a sequence of tasks. However, gains on new tasks can come at the cost of performance on earlier ones (Kirkpatrick et al., 2017; Li & Hoiem, 2017), as well as the broader instruction-following

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.

ability acquired during the model's original post-training (Qwen Team, 2025; Bai et al., 2025). A practical solution should protect both while keeping model size and inference cost fixed.

Existing approaches satisfy only part of this goal. Regularization keeps a single checkpoint by limiting parameter changes, but does not directly preserve the outputs of earlier tasks. Self-distillation also keeps a single checkpoint (Shenfeld et al., 2026), but relies on the current model as its teacher; once an earlier behavior is lost, the teacher cannot recover it. Expert routing provides a fixed reference instead. Each task gets a frozen adapter that preserves its behavior (Wang et al., 2023; Dou et al., 2024; Guo et al., 2025; Xu et al., 2026), but these adapters accumulate over time and require routing at inference. This leads to our central question: can frozen task experts guide continual training without being carried into deployment?

We answer this question with Recursive Continual Post-Training (ReCaP). For each task, ReCaP trains a LoRA expert while keeping the base model frozen. The knowledge learned for the task is therefore captured in the expert's LoRA weights. This frozen expert provides a natural and stable teacher. Instead of deploying the expert, ReCaP distills its behavior into one shared student. At each stage, the current expert teaches the new task, while selected earlier experts review past tasks. Their targets do not change as the student learns, and each expert receives inputs only from its own task. The experts therefore do not need to agree on the same input. This multi-teacher distillation transfers new and earlier task knowledge into one model.

The student learns from the current and earlier tasks in different ways. On the current task, it uses real inputs and learns from both the current expert and the ground-truth responses. For future replay, ReCaP also fits a small flow-matching model to the current task's input embeddings. At later stages, this flow model generates only input embeddings. The frozen expert produces target outputs for these embeddings, and the student learns to match them. Thus, replay does not require storing raw images or text from earlier tasks. After training, the student is merged into the next checkpoint. Only this checkpoint is deployed; the experts and flow models are kept only for future training. This design uses stable task targets during training without requiring a router or per-task adapter at inference. Its retention depends on how well the generated embeddings represent real past inputs. We measure this difference as a replay-fidelity gap.

We evaluate whether this design can learn task sequences while deploying one checkpoint. The experiments cover three vision-language sequences and one text-only sequence. On CoIN with Qwen2.5-VL-3B, ReCaP reaches 78.0% average accuracy, matching or exceeding the strongest expert-routing baseline. On UCIT-O, it reaches 75.5%, which is 4.8 points above the strongest prior method. Across six held-out benchmarks, the average rises from 56.6% to 58.7%, with no benchmark drop. The same procedure also works with other backbones and model sizes. With Qwen3.5-4B, ReCaP reaches 77.8% on CoIN and 79.5% on UCIT-O. On the text-only sequence, Qwen2.5-7B reaches 59.8% skill and 66.4% retention averages. With Llama-3.2-1B, the final Science, Tool, and Medical scores are 48.6%, 61.1%, and 43.1%, respectively.

2 Related Work

Continual learning and replay. Continual learning methods reduce forgetting through regularisation, task-specific parameters, or replay (Wang et al., 2024). EWC, SI, LwF, and L2-SP limit how much a model changes when it learns a new task (Kirkpatrick et al., 2017; Zenke et al., 2017; Li & Hoiem, 2017; Li et al., 2018). Parameter-efficient methods use adapters or structured subspaces to reduce interference (Wang et al., 2023; Li et al., 2025; Zhao et al., 2024). Replay methods such as GEM, A-GEM, iCaRL, and DER keep examples from earlier tasks (Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2018; Rebuffi et al., 2017; Buzzega et al., 2020). For a multimodal model, each example may include an image, an instruction, and a response. Generative replay replaces this buffer with a model of past data (Shin et al., 2017). Reconstructing a complete multimodal example is unnecessary when a frozen expert can provide the target response. ReCaP therefore generates only input embeddings and obtains target outputs from the corresponding task expert.

Distillation for knowledge preservation. Knowledge distillation trains a student to match a teacher (Hinton et al., 2015). Self-distillation uses the model or an earlier version of it as the teacher. Born-again networks train a new model from a previous model's predictions (Furlanello et al., 2018), which can act as regularisation (Mobahi et al., 2020). SDFT conditions the teacher

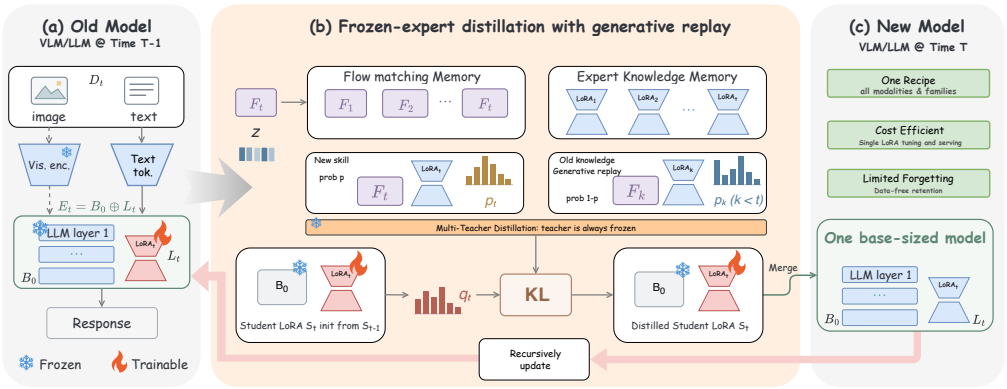


Figure 2: **ReCaP transfers a task sequence into one shared model.** **(a)** For task t, ReCaP trains LoRA L_t on the frozen base B_0 to form expert $E_t = B_0 \oplus L_t$. **(b)** It freezes L_t and fits flow memory F_t to the task input embeddings. Student S_t, initialized from S_{t-1}, learns from the current expert E_t and selected earlier experts E_k. Earlier inputs are sampled from F_k, so replay does not require raw images or text from prior tasks. KL matches the student's token distribution to that of the selected frozen expert. **(c)** ReCaP merges S_t into B_0 to produce checkpoint B_t. S_t initializes the next stage; expert adapters and flow memories are retained only for replay. Only the base-sized B_t is deployed.

on demonstrations and learns from the student's own outputs (Shenfeld et al., 2026). Because its teacher is derived from the current model, it changes as the model changes. If an earlier behavior has already been forgotten, it is also missing from this teacher. ReCaP instead freezes a separate expert for each task. The current expert and selected earlier experts form a fixed set of teachers for one shared model. Each expert teaches only on inputs from its own task, so the teachers do not need to agree on the same example.

Adapter routing and model merging. Routing methods keep task-specific adapters and select among them at inference (Dou et al., 2024; Luo et al., 2024; Guo et al., 2025; Huai et al., 2025). This can preserve earlier skills, but inference memory usage grows with each task and inference requires a router. SOCiaL (Xu et al., 2026) also keeps frozen experts, but serves them through a router; ReCaP uses experts only during training. Model-merging methods combine task updates through task arithmetic, conflict resolution, or scaling (Ilharco et al., 2022; Yadav et al., 2023; Yu et al., 2024; Wang et al., 2025). These methods combine parameter updates without explicitly matching the outputs of the original task experts. ReCaP first distills the experts into one shared model through multi-teacher training and generative replay. Merging is then only the final step that produces a base-sized checkpoint.

3 Recursive Continual Post-Training

We study continual post-training over a stream of T tasks. At stage t, raw data are available only for the current dataset $\mathcal{D}_t$; earlier tasks must be retained without replaying their raw images or text. The goal is to learn each new task while preserving earlier skills. After every stage, the method must produce a single checkpoint with the same parameter count as the foundation model. Inference cannot depend on a task router or task-specific adapter. Compact task memories may be retained for future training, but they are not deployed.

Let B_0 denote the fixed foundation model. Figure 2 summarizes the ReCaP workflow and notation. At stage t, task LoRA L_t forms expert $E_t = B_0 \oplus L_t$, and flow model F_t learns its input embeddings. Student LoRA S_t starts from S_{t-1}, with $S_0 = 0$, and forms the shared student when applied to B_0. For brevity, we also use S_t below to denote the composed student model $B_0 \oplus S_t$. Through multi-teacher distillation (Hinton et al., 2015), S_t learns from E_t on current data and from selected earlier experts E_k on embeddings generated by F_k. Finally, the student LoRA is merged into B_0 to form checkpoint B_t. Reusing S_t to initialize the next stage makes the update recursive. Algorithm 1 summarizes the complete training procedure.

3.1 Per-Task Experts for Knowledge Storage

At stage t, we train a fresh LoRA (Hu et al., 2021) L_t on $\mathcal{D}_t$ while keeping B_0 frozen. Since B_0 does not change, all trainable task-specific changes are stored in L_t, giving expert $E_t = B_0 \oplus L_t$. We freeze and retain L_t, so the fixed pair (B_0, L_t) can reconstruct the same stable teacher at any later stage. Each task therefore adds one LoRA, rather than one full checkpoint, to the training archive.

3.2 Flow Matching for Generative Replay

A frozen expert preserves task k's output behavior, but replay also needs task-appropriate inputs. Let μ_k be the empirical distribution of fixed-length input-embedding blocks extracted from $\mathcal{D}_k$ with E_k. We fit a small flow-matching model (Lipman et al., 2022) F_k to μ_k. Its velocity field is a lightweight transformer over these blocks.

Let $z \sim \mathcal{N}(0, I)$ be Gaussian noise and let $e \sim \mu_k$ be a real embedding block of the same shape. The straight path moves from noise at $\tau = 0$ to data at $\tau = 1$:

$$e_\tau = (1 - \tau)\, z + \tau e, \qquad \tau \sim \mathrm{Unif}[0, 1]. \tag{1}$$

We train v_{ϕ_k} to predict the path velocity $e - z$:

$$\mathcal{L}_{\mathrm{FM}}(\phi_k) = \mathbb{E}_{e\sim\mu_k,\, z\sim\mathcal{N}(0,I),\, \tau}\left[\left\| v_{\phi_k}(e_\tau, \tau) - (e - z)\right\|^2\right]. \tag{2}$$

To generate an input, F_k starts from $\tilde{e}_0 \sim \mathcal{N}(0, I)$ and follows the learned field to $\tau = 1$. With N Euler steps and $\Delta\tau = 1/N$,

$$\tilde{e}_{\tau+\Delta\tau} = \tilde{e}_\tau + v_{\phi_k}(\tilde{e}_\tau, \tau)\Delta\tau, \qquad \tilde{x} := \tilde{e}_1. \tag{3}$$

At a later stage $t > k$, the generated endpoint $\tilde{x}$ is passed to the remaining model layers. F_k therefore reconstructs the replay input, while the frozen expert E_k supplies the distillation target. For compact answer sets, the flow is class-conditioned with uniform class sampling. Replay stores L_k and ϕ_k instead of raw images or text.

Replay fidelity. Training uses generated embeddings from F_k, while evaluation uses the real input-embedding distribution P_k. For a fixed stage-t student, we define the replay-fidelity gap as $\zeta_k = |\mathbb{E}_{x\sim P_k}\mathrm{KL}(E_k\|S_t) - \mathbb{E}_{\tilde{x}\sim F_k}\mathrm{KL}(E_k\|S_t)|$. A smaller gap means that generated replay better estimates the distillation loss on real inputs. It does not measure a general distance between the two input distributions.

3.3 Multi-Teacher Distillation into a Shared Student

When task t arrives, we initialize S_t from S_{t-1} instead of starting again from B_0. The student therefore begins with the knowledge consolidated in earlier stages. It then learns from the current expert E_t and selected earlier experts $\{E_k : k \in \mathcal{R}_t\}$, where $\mathcal{R}_t \subseteq \{1, \ldots, t-1\}$. These experts remain frozen throughout the stage.

Each expert teaches on inputs from its own task. The current expert E_t uses real pairs $(x, y) \sim \mathcal{D}_t$. Under teacher forcing on y, the student matches the expert's token distribution at the response positions. It also learns from the labeled response through cross-entropy. Together, these losses transfer the expert's behavior while keeping the prediction tied to the training target.

Earlier experts use inputs reconstructed by their flow models. For a prior-task update, we select $k \in \mathcal{R}_t$ in round-robin order and draw $\tilde{x} \sim F_k$. Because earlier responses are not stored, distillation needs a continuation $\tilde{y}$ on which to compare the expert and student. The frozen expert E_k generates this continuation and supplies the target distribution. The student then learns to match the expert on the reconstructed input.

Training uses a current-task update with probability ρ and a prior-task update otherwise. For $t > 1$, the expected objective is

$$\begin{aligned}\mathcal{L}(\theta) = {} & \underbrace{\rho\, \mathbb{E}_{(x,y)\sim\mathcal{D}_t}[\mathrm{KL}(E_t\|S_t)]}_{\text{current teacher}} + \underbrace{\rho\beta\, \mathbb{E}_{(x,y)\sim\mathcal{D}_t}[-\log S_t(y \mid x)]}_{\text{target-response supervision}} \\ & + \underbrace{(1-\rho)\, \mathbb{E}_{k\sim\mathrm{Unif}(\mathcal{R}_t)}\mathbb{E}_{\tilde{x}\sim F_k}[\mathrm{KL}(E_k\|S_t)]}_{\text{prior teachers (generated replay)}}.\end{aligned} \tag{4}$$

Here θ parameterizes S_t. The first term matches the current expert, the second trains on the labeled response, and the third matches an earlier expert on a generated input. The weights ρ and β control the update mix and the labeled-response loss, respectively. Round-robin selection makes the expectation over $\mathcal{R}_t$ uniform. At stage one, there is no earlier expert, so we set $\rho = 1$. In the VLM streams, we retain recognition and captioning experts because these tasks gain the most from replay. Setting $\beta = 0$ removes labeled-response supervision, while $\rho = 1$ removes prior-task distillation. After training, we merge the student LoRA into B_0 to produce B_t. Its LoRA weights initialize the next student, while the task experts stay frozen. This is the recursive step in ReCaP.

Algorithm 1 ReCaP: Recursive Continual Post-Training

Require: Base B_0; stream $\{\mathcal{D}_t\}_{t=1}^{T}$; weights β, ρ
$\mathcal{A} \leftarrow \emptyset$; $S_0 \leftarrow 0$ ▷ replay archive; empty student
for $t = 1, \ldots, T$ **do**
  $L_t \leftarrow \mathrm{SFT}(\mathcal{D}_t \mid B_0)$; $E_t \leftarrow B_0 \oplus L_t$
  $F_t \leftarrow \mathrm{FitFlow}(E_t \text{ input embeddings on } \mathcal{D}_t)$
  Initialise student $S_t \leftarrow S_{t-1}$ on B_0
  for each optimisation step **do**
    With prob. ρ (or if $\mathcal{A} = \emptyset$): distil E_t and apply β target CE
    Else: choose $(L_k, F_k) \in \mathcal{A}$, sample $\tilde{x} \sim F_k$, and distil from E_k
    Update S_t using the corresponding term in Eq. 4
  end for
  $\mathcal{A} \leftarrow \mathcal{A} \cup \{(L_t, F_t)\}$; $B_t \leftarrow \mathrm{merge}(B_0, S_t)$
end for
return B_T ▷ one base-sized deployed checkpoint

4 Experiments

4.1 Experimental Setup

Models and Task Streams. We evaluate ReCaP on both vision-language and text-only task streams. The main VLM backbone is Qwen2.5-VL-3B-Instruct (Bai et al., 2025). We use three benchmarks with different stream lengths: six-task UCIT (Guo et al., 2025; Xu et al., 2026), eight-task CoIN (Chen et al., 2024a), and ten-task UCIT-O (Xu et al., 2026). For UCIT-O, we evaluate all three task orders. The text-only stream contains science, tool-use, and medical QA tasks in that order. We train Qwen2.5-7B-Instruct (Qwen Team, 2025) on SciKnowEval-L3 (Feng et al., 2024), ToolAlpaca (Tang et al., 2023), and HuatuoGPT-o1 (Chen et al., 2024b). Additional experiments use Qwen3.5-4B and Llama-3.2-1B-Instruct (Grattafiori et al., 2024) to check whether the results depend on one backbone.

Baselines. The VLM baselines differ in what they deploy. EWC (Kirkpatrick et al., 2017), LwF (Li & Hoiem, 2017), L2-SP (Li et al., 2018), and O-LoRA (Wang et al., 2023) end with one checkpoint. LoRAMoE (Dou et al., 2024), HiDe-LLaVA (Guo et al., 2025), and SOCiaL (Xu et al., 2026) instead retain several experts and use a router at inference. Seq-FFT and Seq-LoRA serve as direct sequential baselines. For CoIN, we also include the zero-shot model and joint multi-task training as reference points. Joint training is not a continual-learning method because it can access all tasks at once. The LLM comparison additionally includes SDFT (Shenfeld et al., 2026).

Training and Evaluation. All methods use the same backbone and follow the same task protocol. ReCaP uses rank-48 teacher experts and a rank-64 student. Increasing or decreasing the student rank by a total factor of eight changes the CoIN average by only 1.4 points, as shown in Table 1. We report ReCaP as the mean over three seeds and provide the standard deviations in the appendix. Baseline results use one seed. Each VLM task is scored with its standard benchmark metric. We use substring matching for question answering, the mean of the standard captioning metrics for caption generation, and a bounding-box IoU threshold of 0.5 for CoIN Grounding. We also test general VLM capability on six benchmarks that never appear in training: SciQA, RWQA, MMT, AI2D, DocVQA, and MMStar. For the LLM stream, each new skill is evaluated on 500 held-out examples, while general retention is measured on five lm-eval-harness benchmarks (Gao et al., 2021). The appendix lists the full training settings, evaluation splits, benchmark names, and deployment costs.

Table 1: CoIN (Chen et al., 2024a) 8-task last accuracy (%). Bold marks the best continual method in each column; reference and †rank-sweep rows are excluded. ReCaP uses one merged checkpoint with no router or per-task adapter at inference.

	Method	Sci	ImNet	VW	Grnd	TxtVQA	GQA	VQAv2	OCR	Avg
Ref.	Zero-shot	80.8	25.9	66.1	28.5	70.1	65.7	72.8	67.3	59.7
	Multi-task	95.9	95.7	69.9	82.7	73.6	67.7	72.8	79.9	79.8
MoE	LoRAMoE	83.4	35.1	68.2	83.5	76.0	66.1	74.0	79.0	70.7
	HiDe	87.8	53.7	68.5	**86.2**	76.4	66.4	74.0	79.2	74.0
	SOCiaL	**93.6**	75.7	**68.6**	81.9	**77.5**	**66.7**	**76.7**	78.8	77.4
$O(1)$	Seq-FFT	67.7	0.0	34.7	0.0	14.0	5.3	3.3	0.0	15.6
	Seq-LoRA	80.9	25.8	67.3	69.4	74.8	64.8	74.2	76.6	66.7
	EWC	81.2	25.1	66.2	75.8	74.7	65.4	74.0	77.6	67.5
	LwF	80.8	27.9	67.2	68.6	74.9	64.3	74.0	79.3	67.1
	L2-SP	81.2	26.0	66.7	54.8	74.6	64.7	74.3	69.5	64.0
	O-LoRA	80.6	26.0	65.8	67.9	74.7	65.5	73.9	77.4	66.5
	ReCaP	89.7	**92.5**	68.0	81.7	72.3	66.2	73.7	**79.8**	**78.0**
Sweep	ReCaP r16†	87.0	88.7	66.4	81.5	71.8	66.1	73.1	79.3	76.7
	ReCaP r48†	89.2	91.8	67.6	81.6	72.1	66.1	73.4	79.7	77.7
	ReCaP r128†	91.0	92.9	68.2	80.9	72.5	66.2	73.6	79.6	78.1

4.2 VLM on Independent Tasks: CoIN

CoIN is an independent eight-task benchmark spanning VQA, image recognition, visual grounding, and OCR (Chen et al., 2024a). Table 1 reports accuracy after the full eight-task stream. ReCaP reaches 78.0 using one merged checkpoint and no router. This is slightly higher than the 77.4 achieved by SOCiaL (Xu et al., 2026), even though SOCiaL retains a separate expert for every task and selects one at inference. HiDe-LLaVA (Guo et al., 2025) reaches 74.0, which is 4.0 points lower than ReCaP. The zero-shot model reaches 59.7, leaving an 18.3-point gap. Joint multi-task training reaches 79.8 with access to all eight tasks at once. Although ReCaP learns the tasks one at a time, it finishes only 1.8 points behind joint multi-task training.

The average score does not show where replay helps. The difference is concentrated on ImageNet: ReCaP reaches 92.5 with replay and 35.4 without it. Other single-checkpoint methods remain close to the zero-shot score of 25.9, with results between 25.1 and 27.9. HiDe performs better at 53.7, but still trails ReCaP. On the other seven tasks, removing replay has little effect. Replay is therefore preserving an earlier recognition task, not raising every score in the stream.

Grounding illustrates the other side of the method. It is learned from the current task data, so the training loss can directly teach the required bounding-box format. Its accuracy rises from 28.5 to 81.7 even though the task is not replayed. ImageNet faces a different problem. It is learned at stage 2 and then passes through six more updates. Figure 3 shows that both matched runs learn ImageNet well at stage 2. Without replay, however, the model gradually loses fine-grained distinctions and may replace "dalmatian" with the broader label "dog." Replaying ImageNet keeps those distinctions in the shared checkpoint. Current-task training teaches the new output format; replay keeps the earlier recognition skill.

4.3 VLM on Overlapping Tasks: UCIT-O

UCIT-O (Xu et al., 2026) contains ten stages with overlap built from ImageNet-R, ArxivQA, VizWiz, CLEVR-Math, IconQA, and Flickr30k. Several datasets return at later stages, so the stream contains more overlap than CoIN. We evaluate all three task orders. Table 2 reports the results on protocol 1, and the remaining two are presented in the App. C.

ReCaP reaches 75.5, exceeding SOCiaL (Xu et al., 2026) by 4.8 points. SOCiaL reaches 70.7 while retaining one expert per task and a router; ReCaP deploys one checkpoint. The task-level results resemble CoIN. ImageNet-R shows the largest replay effect and reaches 84.7. The replayed

Table 2: UCIT-O (Xu et al., 2026) Protocol 1 (10 tasks) last accuracy (%). Bold and markers follow Table 1. ReCaP uses one merged checkpoint with no inference-time router.

	Method	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	Avg
Ref.	Zero-shot	37.9	97.5	31.6	71.6	70.7	28.5	34.7	52.9	29.9	68.0	52.3
	Multi-task	87.6	99.5	52.0	89.3	93.4	29.1	90.5	87.4	39.0	89.4	75.7
MoE	LoRAMoE	44.7	93.1	10.3	81.0	83.1	**34.4**	48.8	75.5	35.3	78.6	58.5
	HiDe	68.7	92.3	10.4	88.7	86.8	33.3	70.7	84.8	37.2	84.0	65.7
	SOCiaL	82.8	**99.5**	39.3	79.5	92.2	29.7	83.8	81.0	33.3	85.7	70.7
$O(1)$	Seq-FFT	17.8	98.0	0.0	31.7	67.7	0.1	18.7	24.3	0.0	48.8	30.7
	Seq-LoRA	41.1	99.3	29.2	88.3	**92.6**	23.5	38.9	85.5	26.3	88.3	61.3
	EWC	40.5	99.4	30.1	88.1	91.3	25.8	38.5	82.3	27.8	86.8	61.1
	LwF	41.0	99.4	31.1	90.0	91.7	25.4	39.5	84.9	28.1	88.3	61.9
	L2-SP	38.7	98.8	31.5	90.8	89.5	29.6	37.1	83.8	30.4	86.8	61.7
	O-LoRA	41.4	99.3	30.4	89.9	91.3	25.5	38.7	84.5	27.8	88.5	61.7
	ReCaP	**84.7**	99.4	**50.2**	**95.2**	**92.6**	28.9	**86.1**	**88.8**	**37.4**	**91.8**	**75.5**

Table 3: LLM 3-stage stream on Qwen2.5-7B-Instruct after the final task. HS = HellaSwag, WG = Winogrande, and TQA = TruthfulQA-mc2. Comb. = (skill Avg + retention Avg)/2. Bold marks the best continual method; Base is excluded, and ties are both bold.

	Skill (%)				Retention (%)						
Method	Science	Tool	Medical	Avg	HS	MMLU	WG	IFEval	TQA	Avg	Comb.
Base (no CL)	71.1	52.6	40.7	54.8	80.5	71.7	70.7	60.8	64.7	69.7	62.3
SDFT	52.5	62.9	40.5	52.0	79.5	71.7	74.7	28.3	55.7	62.0	57.0
Seq-FFT	**78.7**	3.0	0.6	27.4	72.9	60.6	**75.4**	6.8	40.2	51.2	39.3
Seq-LoRA	55.0	65.1	**47.8**	56.0	79.7	70.3	72.4	44.2	57.9	64.9	60.4
EWC	76.6	59.2	47.7	**61.2**	79.9	71.5	71.2	**50.3**	62.0	**67.0**	**64.1**
LwF	78.1	**66.5**	27.0	57.2	79.9	**72.2**	72.0	41.6	57.1	64.6	60.9
L2-SP	76.9	10.4	18.1	35.1	**80.4**	71.5	70.4	45.3	**62.4**	66.0	50.5
ReCaP	76.0	61.8	41.6	59.8	79.7	71.4	72.1	49.4	59.2	66.4	63.1

captioning stages also improve, but by smaller amounts, while most tasks that are not replayed change little. We observe the same pattern with Qwen3.5-4B. Its CoIN result is 77.8, which is within the seed variation of the main backbone. On UCIT-O, it reaches 79.5, improving on the main-backbone result by 4.0 points. The largest replay gains again come from the recognition and captioning tasks selected for replay.

4.4 Cross-Modality Validation: LLM 3-Stage Stream

We next ask whether the same training procedure works for a text-only model. The stream contains science, tool-use, and medical QA tasks. We evaluate the result in two parts: how well the final model retains the three new skills, and how much of its general language ability remains.

On the three new skills, ReCaP finishes with an average of 59.8 in Table 3. This is 5.0 points above the untrained base model and 3.8 points above Seq-LoRA. The sequential baselines often retain one skill at the expense of another. Seq-FFT, for example, scores 78.7 on science but falls to 3.0 on tool use and 0.6 on medical QA.

For SDFT, we use its original tool→science→medical order and selected learning rate. Its final tool and science scores are 62.9 and 52.5, respectively, and its skill average is 52.0. EWC performs best on this short stream with an average of 61.2, placing ReCaP second. That ranking does not carry over to the longer VLM streams. On UCIT-O, EWC reaches 61.1 while ReCaP reaches 75.5. ReCaP also obtains 78.0 on CoIN and 81.0 on UCIT. These streams differ in modality, task composition, and length, so they do not form a controlled scaling study. They do show that the best continual-learning method can depend on the type of stream.

For general retention, ReCaP averages 66.4 across the five benchmarks. The base model reaches 69.7, so the difference is 3.3 points. ReCaP remains close to EWC at 67.0 and scores above L2-SP at 66.0, Seq-LoRA at 64.9, LwF at 64.6, and SDFT at 62.0. Most of this loss comes from IFEval. Its score falls from 60.8 to 49.4, a drop of 11.4 points. The other held-out evaluation gives a different view: across five additional benchmarks, the average rises from 69.2 to 69.9. The per-benchmark results are reported in Table 11. The retention loss is therefore concentrated in strict instruction following rather than spread evenly across all measured abilities.

When skill learning and general retention are weighted equally, ReCaP reaches 63.1, ahead of LwF at 60.9 and Seq-LoRA at 60.4, but below EWC at 64.1. The base model scores 62.3, but this is only a no-training reference because it never learns the three new tasks. On this short LLM stream, EWC gives the strongest overall result. ReCaP performs better on the longer VLM streams.

4.5 Ablation Studies and Discussion

Replay Ablation. To isolate the role of replay, we remove it while keeping the student, training schedule, and current-task loss unchanged. Table 4 shows that the CoIN average then falls from 78.0 to 70.9. On the UCIT-O order reported in the main table, it falls from 75.5 to 74.3. The averages hide a much larger change on the replayed recognition tasks. ImageNet drops from 92.5 to 35.4 on CoIN, while ImageNet-R drops from 84.7 to 76.0 on UCIT-O. Most other tasks move only slightly. Replay is therefore not a general accuracy boost. Its main role is to keep selected earlier skills from being overwritten. The appendix reports per-task replay results for the Qwen3.5 backbone.

Student Capacity. We also test whether the result comes from student capacity rather than the training method. Increasing the student rank from 16 to 128 changes the CoIN average from 76.7 to 78.1. This is a range of only 1.4 points across an eight-fold change in rank. The rank-64 student used in the main experiments lies within this range, as shown in Table 1. Additional controls compare four ways to merge the task experts directly in weight space (App. G).

Table 4: Component analysis and held-out retention (Qwen2.5-VL-3B, %). *Stream* reports last accuracy. The distillation-only and ReCaP stream results are 3-seed full-split means. *Retention* reports six benchmarks outside the training streams.

	Components		Stream (last acc.)			Retention (held-out)						
Configuration	Distil	Replay	CoIN	UCIT	UCIT-O	SciQA	RWQA	MMT	AI2D	DocVQA	MMStar	Avg
Base (no CL)	–	–	59.7	46.6	52.3	65.8	37.7	61.1	78.0	54.6	42.2	56.6
Weight merge	×	×	60.4	47.7	53.2	65.8	**39.9**	61.1	78.6	54.9	47.6	58.0
+distil	✓	×	70.9	76.5	74.3	66.8	39.2	**61.3**	78.7	**56.3**	**51.1**	**58.9**
ReCaP	✓	✓	**78.0**	**81.0**	**75.5**	**67.5**	37.9	61.2	**78.8**	55.6	51.0	58.7

Held-Out General Capability. We next ask whether learning the task stream damages unrelated VLM abilities. It does not. The held-out average rises from 56.6 for the base model to 58.7 for ReCaP, a gain of 2.1 points, and every individual benchmark remains at or above its base score. These results appear on the right side of Table 4. The intermediate rows show where this gain comes from. Distillation without replay reaches 58.9, while the full method reaches 58.7. Distillation accounts for the improvement on held-out benchmarks. Replay serves a different purpose: it protects earlier tasks within the training stream.

Replay Generator and Flow Path. We compare the replay generators using two properties. Fidelity asks whether generated embeddings produce the same distillation loss as real embeddings. We measure it with $\zeta_k = |\mathbb{E}_{P_k}\mathrm{KL} - \mathbb{E}_{F_k}\mathrm{KL}|$, reported as a percentage, where a smaller value is better. Second, coverage asks how much of the real embedding distribution the generator can represent, so a larger value is better. We estimate the fidelity gap from 150 generated and 150 real blocks using one seed. Coverage is measured with kNN density and coverage (Naeem et al., 2020) after projection to 64 PCA dimensions.

The comparison includes the five-component GMM used by SOCiaL (Xu et al., 2026), a diagonal Gaussian, a block-level VAE, a diffusion model, and flow matching. Diffusion and flow matching

Table 5: Replay-fidelity gap ζ_k (%) and mode coverage (%) on VLM tasks. Lower ζ_k is better; higher coverage is better. Coverage uses kNN after a PCA-64 projection. Bold marks the best result for each task.

	GMM (k=5)		Gaussian (diag)		VAE		Diffusion		Flow-matching (ours)	
Task	$\zeta_k\downarrow$	Cov.↑	$\zeta_k\downarrow$	Cov.↑	$\zeta_k\downarrow$	Cov.↑	$\zeta_k\downarrow$	Cov.↑	$\zeta_k\downarrow$	Cov.↑
ImageNet-R	1.1	57.7	4.6	43.7	30.0	37.0	6.3	0.4	**0.5**	**88.6**
VizWiz	5.0	51.8	4.7	35.3	41.4	27.8	4.1	1.0	**1.8**	**87.5**
Flickr30k	2.6	68.3	8.4	47.7	**0.8**	45.6	1.2	0.0	1.6	**88.5**

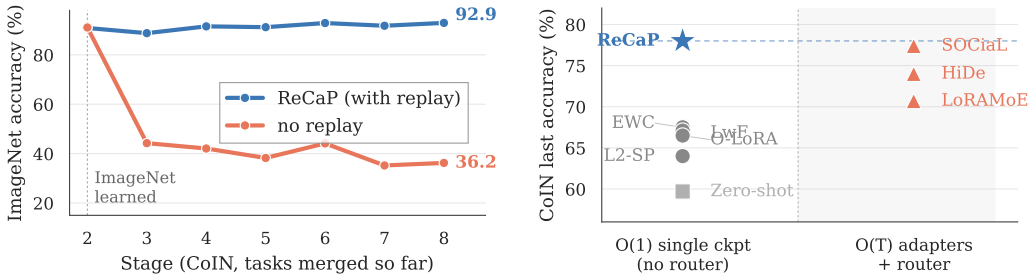


Figure 3: **Left:** Per-stage ImageNet accuracy on CoIN for two matched runs that differ only in replay (single seed 42). Both reach ≈91 when ImageNet is learned at stage 2. Later merges reduce it to 36.2 without replay (35.4 as a 3-seed full-split mean; Table 4), while replay keeps it at 89–93. **Right:** Accuracy against deployment cost. ReCaP uses one $O(1)$ checkpoint and no router.

use the same DiT backbone and the same 50 sampling steps. Their comparison changes the training objective while keeping model capacity and sampling cost fixed.

Flow matching covers the most modes on every task, with coverage between 87.5% and 88.6%. It also has the smallest fidelity gap on ImageNet-R and VizWiz. The VAE gives the smallest ζ_k on Flickr30k at 0.8%, but this value rises to between 30.0% and 41.4% on the other two tasks. Diffusion produces ζ_k values between 1.2% and 6.3%, similar to flow matching, yet its coverage stays between 0.0% and 1.0%. The GMM and diagonal Gaussian cover more modes than diffusion, but their coverage remains between 35.3% and 68.3%. This comparison also explains our use of a straight flow path. Each retained task needs its own generator, so the generator should remain simple as the stream grows. The straight path produces embeddings directly in the expert's input space, without the VAE's separate decoder or a diffusion noise schedule. As shown in Table 5, it is the only tested generator that maintains both low fidelity gaps and high coverage across all three tasks.

Task-Order Sensitivity. The two additional UCIT-O orders lead to the same conclusion. ReCaP reaches 73.9 and 74.1, making it the strongest single-checkpoint method on both orders. It remains 0.5 points below SOCiaL (Xu et al., 2026), which keeps an expert pool and a router. Together with the order reported in Table 2, these results show that the single-checkpoint result is not tied to one task order. Full per-task results for the two additional orders are provided in App. C.

5 Conclusion

ReCaP distils each new-task expert and selected earlier experts into one base-sized checkpoint. Flow-matching memories regenerate earlier-task inputs, so replay needs no buffer of raw images or text. The experts provide fixed targets only during training. On CoIN and UCIT-O Protocol 1, ReCaP reaches 78.0 and 75.5, respectively, matching or exceeding the strongest $O(T)$-expert baselines while deploying one checkpoint. The same procedure works on vision-language and text-only streams and on Qwen and Llama backbones from 1B to 7B. Fixed multi-teacher distillation and generated replay can therefore preserve earlier tasks without increasing deployed model size.

AI USE STATEMENT

We used generative AI tools in three roles. First, they aided writing by suggesting edits for grammar, clarity, concision, and document organisation; the authors selected and revised the final wording. Second, they supported retrieval and discovery by proposing search terms and candidate related work; the authors read the original sources and checked citation metadata before inclusion. Third, they assisted with research-code drafting, debugging, result analysis, and manuscript consistency checks. The authors chose the research directions, ran and evaluated the experiments, checked the reported claims against source artifacts, and reviewed all AI-assisted text and code.

REPRODUCIBILITY STATEMENT

The recipe is specified in Sec. 3, with the per-stage objective in Eq. 4 and the full procedure in Alg. 1. All backbones, streams, optimiser settings, LoRA ranks, step counts and skill/replay ratios are given in Sec. 4 and App. A. These sections also state the evaluation metrics, sample counts, and seeds. Every reported ReCaP number is a 3-seed mean, with per-task standard deviations in the appendix tables, unless marked single-seed. Deployment costs are compared in App. H. Per-task results for the shorter UCIT stream and for the Qwen3.5-4B backbone are in App. B and App. D.

REFERENCES

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-VL technical report. arXiv preprint arXiv:2502.13923, 2025.

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.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with A-GEM. *arXiv preprint arXiv:1812.00420*, 2018.

Cheng Chen, Junchen Zhu, Xu Luo, Heng Tao 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.

Junying Chen, Zhenyang Cai, Ke Ji, Xidong Wang, Wanlong Liu, Rongsheng Wang, Jianye Hou, and Benyou Wang. HuatuoGPT-o1, towards medical complex reasoning with LLMs. *arXiv preprint arXiv:2412.18925*, 2024b.

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.

Kehua Feng, Xinyi Shen, Weijie Wang, Xiang Zhuang, Yuqi Tang, Qiang Zhang, and Keyan Ding. SciKnowEval: Evaluating multi-level scientific knowledge of large language models. *arXiv preprint arXiv:2406.09098*, 2024.

Tommaso Furlanello, Zachary Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. In *International conference on machine learning*, pp. 1607–1616. PMLR, 2018.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, et al. A framework for few-shot language model evaluation. *Zenodo*, 2021.

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

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. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13572–13586, 2025.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

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. *arXiv preprint arXiv:2106.09685*, 2021.

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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19608–19617. IEEE, 2025.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

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.

Xinlong Li, Weijieying Ren, Wei Qin, Lei Wang, Tianxiang Zhao, and Richang Hong. Analyzing and reducing catastrophic forgetting in parameter efficient tuning. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2025.

Xuhong Li, Yves Grandvalet, and Franck Davoine. Explicit inductive bias for transfer learning with convolutional networks. In *International conference on machine learning*, pp. 2825–2834. PMLR, 2018.

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

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

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

Tongxu Luo, Jiahe Lei, Fangyu Lei, Weihao Liu, Shizhu He, Jun Zhao, and Kang Liu. MoELoRA: Contrastive learning guided mixture of experts on parameter-efficient fine-tuning for large language models. *arXiv preprint arXiv:2402.12851*, 2024.

Hossein Mobahi, Mehrdad Farajtabar, and Peter Bartlett. Self-distillation amplifies regularization in Hilbert space. *Advances in Neural Information Processing Systems*, 33:3351–3361, 2020.

Muhammad Ferjad Naeem, Seong Joon Oh, Youngjung Uh, Yunjey Choi, and Jaejun Yoo. Reliable fidelity and diversity metrics for generative models. In *International conference on machine learning*, pp. 7176–7185. PMLR, 2020.

Qwen Team. Qwen2.5 technical report. arXiv preprint arXiv:2412.15115, 2025.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. iCaRL: Incremental classifier and representation learning. In *IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 5533–5542. IEEE, 2017.

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning. *arXiv preprint arXiv:2601.19897*, 2026.

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

George Stoica, Pratik Ramesh, Boglarka Ecsedi, Leshem Choshen, and Judy Hoffman. Model merging with SVD to tie the knots. In *International Conference on Learning Representations*, volume 2025, pp. 4501–4519, 2025.

Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, Boxi Cao, and Le Sun. ToolAlpaca: Generalized tool learning for language models with 3000 simulated cases. *arXiv preprint arXiv:2306.05301*, 2023.

Ke Wang, Nikos Dimitriadis, Alessandro Favero, Guillermo Ortiz-Jimenez, Francois Fleuret, and Pascal Frossard. LiNeS: Post-training layer scaling prevents forgetting and enhances model merging. In *International Conference on Learning Representations*, volume 2025, pp. 27100–27133, 2025.

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*, 46(8):5362–5383, 2024.

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, 2023.

Xiang Xu, Yiyang Su, Tianchen Zhao, Zheng Zhang, Zhuowen Tu, Anil Jain, and Jon Wu. Beyond disjoint tasks: Towards more natural continual learning for vision-language models. In *European Conference on Computer Vision (ECCV)*, Malmö, Sweden, September 8–12, 2026.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. TIES-Merging: Resolving interference when merging models. *Advances in neural information processing systems*, 36:7093–7115, 2023.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language models are Super Mario: Absorbing abilities from homologous models as a free lunch. In *International Conference on Machine Learning (ICML)*, 2024.

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

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 (Volume 1: Long Papers)*, pp. 11641–11661, 2024.