GROUPROUTE: INSTRUCTION-GUIDED GROUPING AND ROUTING FOR CONTINUAL LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

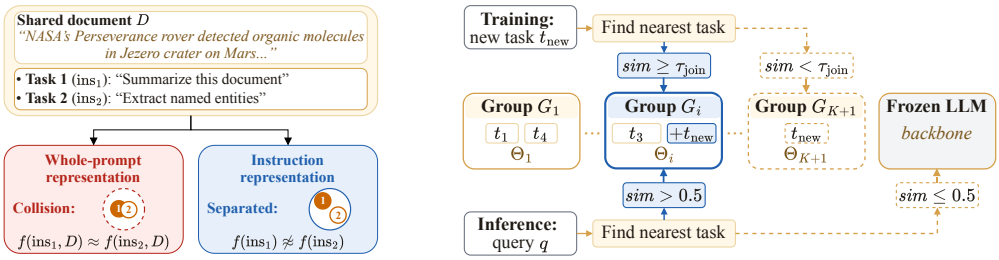
Continual instruction tuning requires language models to learn a sequence of tasks specified by natural-language instructions while maintaining performance on earlier tasks. When explicit task identifiers are unavailable at inference, the model must determine the requested operation from the prompt. Whole-prompt representations, however, can entangle the requested operation with input content, confusing tasks with similar inputs or shared source material. We show that instruction representations extracted from the earliest Transformer layers (layer indices 1–3) of frozen language models are sufficient for reliable task identification. On a shared-corpus benchmark, routing from the instruction alone achieves 99.4–99.7% accuracy even when instructions are reworded. Building on these insights, we introduce GroupRoute, an instruction-guided framework that groups related tasks as they arrive, guides training within each group, and routes queries to the corresponding group at inference. Task assignment and inference routing use frozen backbone representations and require no trainable routing parameters or additional training. GroupRoute routes queries that do not sufficiently match an observed task to the frozen backbone, preserving pretrained capabilities on unseen tasks, while routing related queries to learned groups enables positive transfer. We further introduce Instruction-Keyed Continual Learning (InK), which combines instruction-conditioned IA^3 scaling with a block-diagonal Fisher penalty to learn successive tasks with fixed adapter and consolidation storage within each group. Across the long-sequence benchmark, SuperNI-42, and TRACE, GroupRoute consistently improves performance and reduces forgetting for a range of continual learning methods, including InfLoRA, O-LoRA, DualPrompt, and InK. On SuperNI-42, a single InK adapter already outperforms all compared methods without GroupRoute on two of three backbones, and InK combined with GroupRoute exceeds all evaluated whole-prompt routing baselines on all three.

1 INTRODUCTION

Continual instruction tuning adapts a language model to a sequence of tasks specified by natural-language instructions (Yin et al., 2022; Scialom et al., 2022) while retaining previously acquired capabilities. Explicit task identifiers are often unavailable at inference time, so the model must determine the requested operation from the input prompt. One line of work mitigates interference during sequential adaptation without relying on explicit task routing. EWC (Kirkpatrick et al., 2017) regularizes parameter updates, while InfLoRA (Liang & Li, 2024) and O-LoRA (Wang et al., 2023a) combine task-specific updates learned under null-space or orthogonality constraints. However, accumulating task penalties over time degrades plasticity. Regularization methods restrict updates along previously learned directions (Schwarz et al., 2018), while subspace methods face subspace saturation as successive projections consume available dimensions.

A complementary line of work addresses interference through task routing or gating. DualPrompt (Wang et al., 2022b) and EPI (Wang et al., 2023c) select prompts or prefixes, while SAPT (Zhao et al., 2024) and GainLoRA (Liang et al., 2025) use attention or gating to weight adapter contributions. However, these methods derive their routing representations from the whole prompt, combining the instruction with the input text, whose domain, topic, and vocabulary may also influence routing. This becomes problematic when different tasks share source material. In Super-NaturalInstructions (Wang et al., 2022a), 78.5% of tasks share a source corpus with another task.

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.



(a) Shared content ambiguity.

(b) GroupRoute: instruction-guided grouping and routing.

Figure 1: **Overview of GroupRoute** (detailed in Section 3.1). (a) Tasks that share a document but request different operations collide under whole-prompt representations, whereas instruction representations separate them. (b) During training, a new task joins the group of its most similar task if $sim \geq \tau_{\text{join}}$ and otherwise starts a new group; at inference, a query is routed to the group of its most similar task if $sim > 0.5$ and otherwise to the frozen backbone.

The same document may require summarization, classification, or information extraction under different instructions. Whole-prompt routing can therefore confuse tasks with similar source content but different requested operations.

We address this ambiguity by extracting representations from the instruction only, which explicitly describes the requested operation (Figure 1a). Reading the instruction independently allows task identification before self-attention integrates it with the input text. Although some whole-prompt routing methods, such as EPI (Wang et al., 2023c), use final-layer representations, our analysis across LLaMA-2-7B (Touvron et al., 2023), Gemma-2B-it (Gemma Team, 2024), and T5-large (Raffel et al., 2020) shows that instruction representations extracted from the earliest Transformer layers (layer indices 1–3) of frozen language models are sufficient for reliable task identification. On 40 tasks from Super-NaturalInstructions that share source corpora, instruction routing achieves 100% accuracy with original instructions and 99.4–99.7% under rewording across all three backbones, outperforming the whole-prompt routing baselines.

Building on these insights, we introduce *GroupRoute, an online framework for instruction-guided grouping and routing* (Figure 1b). Each task is represented by a prototype formed from its instruction representation. An incoming task joins an existing group when its prototype is sufficiently similar to that of a previously observed task; otherwise, it starts a new group. Tasks assigned to the same group are trained sequentially within a shared group-specific parameter set using the chosen continual learning method, while different groups maintain separate parameter sets that isolate updates across groups. At inference, we route the query to the group containing the task prototype most similar to its instruction representation. Confusion between prototypes within the same group therefore does not change the selected group parameters. Task assignment and inference routing use frozen backbone representations and require no trainable routing parameters or additional training.

We further introduce *Instruction-Keyed Continual Learning (InK)*, which learns successive tasks through a single shared adapter whose activation scaling is conditioned on the instruction. InK extends IA^3 (Liu et al., 2022) by making its activation scaling dependent on the instruction. Tasks share a single adapter, whose scaling vectors are adjusted by the instruction representation of each task, and a block-diagonal Fisher penalty consolidates the shared parameters to protect earlier tasks. Adapter and consolidation storage therefore remain fixed as tasks accumulate. Without GroupRoute, a single InK adapter outperforms all compared methods that do not use GroupRoute on SuperNI-42 with Gemma-2B-it and LLaMA-2-7B, including whole-prompt routing methods that store parameters for each task. With GroupRoute, each group maintains its own InK adapter.

Our main contributions are the following:

- **Early-Layer Instruction Representations.** Across three Transformer backbones, we show that instruction representations extracted from the earliest layers (layer indices 1–3) reliably distinguish tasks sharing source corpora and remain robust to instruction rewording.

- **Instruction-Guided Grouping and Routing.** We introduce GroupRoute, an online framework that uses frozen instruction representations to group tasks during training and route queries at inference. Task assignment and inference routing require no trainable routing parameters or additional training. GroupRoute consistently improves existing continual learning methods, including InfLoRA, O-LoRA, and DualPrompt.
- **Instruction-Keyed Continual Learning.** InK combines instruction-conditioned IA^3 scaling with a block-diagonal Fisher penalty to learn successive tasks in a single fixed-size adapter. As a single adapter, InK outperforms all compared methods without GroupRoute on SuperNI-42 for Gemma-2B-it and LLaMA-2-7B; with GroupRoute, it achieves the highest average performance on all three benchmarks with Gemma-2B-it: the long-sequence benchmark (Razdaibiedina et al., 2023; Wang et al., 2023a), SuperNI-42, and TRACE (Wang et al., 2023b).

2 Instruction Representations for Task Identification

2.1 Early-Layer Instruction Representations

Instructions specify the requested operation, such as summarization, classification, or extraction, together with requirements for the output format and structure. The input text provides the material on which that operation is performed, so the instruction itself describes the task independently of the particular input instance. Text representations are commonly extracted from final hidden states, both for sentence embeddings (Reimers & Gurevych, 2019; Jiang et al., 2024) and classification (Devlin et al., 2019), and EPI likewise uses final-layer representations for task identification (Wang et al., 2023c). However, analyses of LLaMA-2 suggest that earlier layers preserve word meaning more directly, whereas later layers increasingly support next-token generation (Liu et al., 2024). We therefore examine whether early-layer instruction representations can identify the requested task.

Each instruction is processed separately from its input text using the frozen backbone. At a given projection site and layer ℓ, its representation $h_\ell(I)$ is obtained by averaging the activations over the instruction tokens. To discount generic framing and formatting shared across tasks, we center the representation by subtracting a reference mean μ_ℓ, estimated from an independent pool of SuperNI instructions that excludes all evaluation tasks. For task t, let $\mathcal{I}_t$ denote the instructions that occur with that task. We represent the task by the prototype $k_{t,\ell} = \frac{1}{|\mathcal{I}_t|} \sum_{I \in \mathcal{I}_t} (h_\ell(I) - \mu_\ell)$.

We select the projection sites and layer ranges using an independent corpus built from PromptSource (Bach et al., 2022), which contains 928 independently authored instruction wordings from 165 task identities (see Appendix A.1). The multiple wordings associated with each task allow the analysis to measure both task discrimination and robustness to wording changes. For each backbone, we evaluate every candidate projection site and every contiguous layer range across the full network depth. We then evaluate each wording as a query against the task prototypes formed from the other wordings of that task. The query is assigned to the task prototype with the highest mean cosine similarity across the candidate locations. We select the configuration with the highest retrieval accuracy at rank one for each backbone.

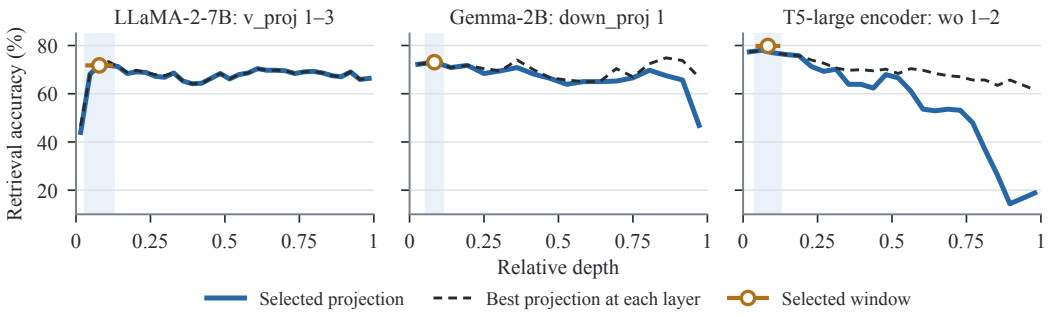


Figure 2: Retrieval accuracy (R@1) on the PromptSource corpus across relative depth. Solid: selected projection; dashed: best projection at each layer; circle and shading: selected window.

Table 1: Task-identification accuracy (%) on 40 Super-NaturalInstructions tasks using original instructions and eight alternative wordings per instruction.

	Gemma-2B-it		LLaMA-2-7B		T5-large	
Routing Method	**Orig.**	**Reword.**	**Orig.**	**Reword.**	**Orig.**	**Reword.**
Instruction, selected sites and layers	**100.00**	**99.38**	**100.00**	**99.69**	**100.00**	**99.38**
Whole prompt, same selected sites and layers	62.71	60.65	59.79	57.42	75.83	72.47
Whole prompt, all sites and layers	65.62	62.84	59.17	57.14	66.46	59.32
Whole prompt, final hidden layer	47.50	46.98	51.46	49.79	78.33	74.82
EPI, whole prompt	83.33	82.47	75.00	73.93	98.54	97.37

Figure 2 reports retrieval accuracy (R@1) across network depth and marks the selected configurations. The selected location is layer 1 at `down_proj` for Gemma-2B-it, layers 1–3 at `v_proj` for LLaMA-2-7B, and encoder layers 1–2 at `wo` for T5-large. Although the search covers the full depth of each backbone, all selected ranges lie within zero-based indices 1–3 and represent less than 12% of total network depth. These results show that the earliest layers already provide instruction representations that distinguish tasks across different wordings.

2.2 Task Identification Under Shared Content and Rewording

We evaluate task identification on 40 tasks from Super-NaturalInstructions (Wang et al., 2022a), each sharing its source corpus with at least one other task but having a distinct instruction (Appendix A.2). To test whether identification remains robust to instruction wording, we keep the input text unchanged and evaluate the original instruction together with eight alternative wordings that preserve the same operation.

At inference time, an incoming query instruction I is processed through the frozen backbone to obtain its centered representation $q_\ell(I) = h_\ell(I) - \mu_\ell$. We select the task whose stored prototype $k_{t,\ell}$ has the highest mean cosine similarity across the selected sites and layers $\mathcal{R}$ identified in Section 2.1:

$$\hat{t} = \arg\max_t \frac{1}{|\mathcal{R}|} \sum_{\ell \in \mathcal{R}} \cos\left(q_\ell(I), k_{t,\ell}\right). \tag{1}$$

We compare instruction routing with three whole-prompt baselines that apply this same decision rule to representations of the whole prompt (instruction and input text combined). These baselines extract representations from all candidate projection sites, from the identical sites and layers selected for instruction routing, or from the final hidden layer. Comparing instruction-only and whole-prompt representations at identical sites and layers isolates the confounding effect of the input text. We also compare against EPI (Wang et al., 2023c), which uses Mahalanobis distance over whole-prompt representations with a shared covariance estimate.

Table 1 shows that instruction routing achieves 100.00% accuracy on original instructions and 99.38%–99.69% under rewording across all three backbones. In contrast, whole-prompt routing performs substantially worse when tasks share source content. Final-layer whole-prompt representations achieve only 47.50% accuracy on Gemma-2B-it and 51.46% on LLaMA-2-7B. Even when whole-prompt representations are extracted from the same selected early layers, accuracy remains below 63% for Gemma-2B-it and LLaMA-2-7B, showing that early layer selection alone cannot overcome interference from the input text. While EPI improves whole-prompt performance, its accuracy remains 16.7–25.8 percentage points below instruction routing on Gemma-2B-it and LLaMA-2-7B, whereas the gap is only 1.5–2.0 points on T5-large. These results show that separating the instruction from the input text improves task identification on the shared-corpus benchmark, with the largest gains observed on Gemma-2B-it and LLaMA-2-7B, and that this advantage persists under alternative instruction wordings.

3 Instruction-Guided Continual Learning

Building on these findings, we introduce GroupRoute, which uses instruction representations to group arriving tasks and route queries to the appropriate group. The framework supports different

continual learning methods within each group; we further introduce InK to learn successive tasks without expanding the group's adapter.

3.1 GROUPROUTE: INSTRUCTION-GUIDED GROUPING AND ROUTING

GroupRoute uses instruction similarity to decide which group receives each arriving task. The process is depicted in Figure 1b. Each task is assigned before training begins. Each group has its own trainable parameters, and training a task updates only the parameters of its assigned group. Tasks assigned to different groups therefore use separate parameter sets, while tasks assigned to the same group are learned sequentially with the chosen continual learning method.

Tasks are assigned online to groups $G_1, \ldots, G_K$, with $g(t)$ denoting the group index of task t. The first task forms group G_1. When a subsequent task t arrives, we compare its prototype with the prototypes of previously observed tasks using the mean cosine similarity defined in Equation 1. If the most similar task, indexed by j_t^*, has similarity at least τ_{join}, task t joins group $G_{g(j_t^*)}$; otherwise, it forms a new group with newly initialized parameters. The number of groups is determined online.

We calibrate the joining threshold using the independent PromptSource corpus from Section 2.1. For each backbone, τ_{join} is the 99th percentile of cosine similarities between instructions from different tasks in that corpus. The resulting thresholds are 0.7485 for Gemma-2B-it, 0.7731 for LLaMA-2-7B, and 0.8052 for T5-large. Appendix B presents the task groups formed on Gemma-2B-it and examines the similarities between their instructions.

At inference, we identify the task prototype most similar to the query instruction using the decision rule in Section 2.2. The query is then routed to the group containing that task. Confusion between task prototypes within the same group therefore does not change the selected group.

To preserve the pretrained model's behavior on queries that do not match an observed task, GroupRoute can process such queries with the frozen backbone without using an adapter. After centering, the reference mean μ_ℓ corresponds to the origin, which provides the comparison point for deciding whether to use an existing group. Under normalized representations, for an instruction query q and task prototype k_t,

$$\|q - k_t\|^2 < \|q - \mathbf{0}\|^2 \iff 2 - 2\cos(q, k_t) < 1 \iff \cos(q, k_t) > \frac{1}{2}. \tag{2}$$

We apply this comparison to the average cosine similarity across the selected sites and layers. If no task prototype exceeds one half, the query is processed by the frozen backbone.

GroupRoute applies directly to different parameter-efficient continual learning methods, including InfLoRA, O-LoRA, and DualPrompt. These methods can continue adding parameters within each group as tasks accumulate. We next introduce InK, which learns successive tasks using a fixed-size adapter and consolidation state.

3.2 INSTRUCTION-KEYED CONTINUAL LEARNING (INK)

Instruction-Keyed Continual Learning (InK) extends IA^3 (Liu et al., 2022) by making its activation scales depend on the instruction. InK uses a single adapter for successive tasks, with instruction representations determining the scales applied to each task. When combined with GroupRoute, each group maintains its own InK adapter and consolidation storage. For a linear projection Wx, InK scales activations as

$$u_t = a + b \odot \hat{k}_t, \qquad y = \exp(u_t) \odot (Wx). \tag{3}$$

Here, $a, b \in \mathbb{R}^m$ are trainable vectors shared across tasks, where m is the dimension of the IA^3 scaling vector at the adapted site, and the instruction key $\hat{k}_t \in \mathbb{R}^m$ is the centered, normalized instruction representation extracted at that site. The vector a provides a shared log-scale, while $b \odot \hat{k}_t$ adjusts it according to the instruction. At inference, InK selects the stored task key most similar to the query instruction to set the activation scales.

For the following derivation, let t index the tasks learned by one InK adapter in their order of arrival. Updating the shared vectors a and b can alter the scales applied to earlier tasks. We protect these scales using empirical Fisher information accumulated across tasks. Because two parameters jointly

determine each channel's log-scale, we use a channelwise 2×2 block-diagonal Fisher matrix. For an earlier task $s < t$, let $g_s = \nabla_u L_s$ denote the loss gradient with respect to the log-scale and $f_s = \mathbb{E}[g_s^2]$. The chain rule gives $\nabla_a L_s = g_s$ and $\nabla_b L_s = g_s \odot \hat{k}_s$, yielding the Fisher contribution

$$\widehat{\mathcal{F}}_s = \begin{bmatrix} \mathrm{diag}(f_s) & \mathrm{diag}(f_s \odot \hat{k}_s) \\ \mathrm{diag}(f_s \odot \hat{k}_s) & \mathrm{diag}(f_s \odot \hat{k}_s^2) \end{bmatrix}. \tag{4}$$

The off-diagonal block $\mathrm{diag}(f_s \odot \hat{k}_s)$ captures the coupled sensitivity between a and b under task s's instruction key. Penalizing a alone would leave parameter adjustments along b unprotected.

Following online EWC (Schwarz et al., 2018), we accumulate these contributions into a single running matrix, $\mathcal{F}_{t-1} = \sum_{s<t} \widehat{\mathcal{F}}_s$, with $\mathcal{F}_0 = \mathbf{0}$. The training objective for task t regularizes changes from the most recently learned parameters a_{t-1} and b_{t-1},

$$\mathcal{L}_t(a, b) = L_t(a, b) + \frac{\lambda}{2} \begin{bmatrix} a - a_{t-1} \\ b - b_{t-1} \end{bmatrix}^\top \mathcal{F}_{t-1} \begin{bmatrix} a - a_{t-1} \\ b - b_{t-1} \end{bmatrix}, \tag{5}$$

where λ controls the regularization strength, and the penalty is summed over adapter sites. This accumulation retains a fixed-size consolidation state alongside the shared adapter, so neither adapter nor consolidation storage grows with the number of tasks learned by that adapter. With GroupRoute, this storage remains fixed within each group, while total storage grows with the number of groups.

4 Experiments

We evaluate whether GroupRoute improves different continual learning methods, how InK compares with parameter-expansion methods in task performance and adapter storage, and whether GroupRoute preserves pretrained capabilities on unseen tasks.

4.1 Experimental Setup

Benchmarks. We evaluate three benchmarks. The long-sequence benchmark contains 15 NLP tasks introduced by Progressive Prompts (Razdaibiedina et al., 2023). Following O-LoRA (Wang et al., 2023a) and GainLoRA (Liang et al., 2025), we evaluate two task permutations (named O3 and O4 as in GainLoRA, Appendix C.1 lists both) to assess sensitivity to task order. We introduce SuperNI-42, a continual instruction tuning benchmark consisting of the 40 tasks studied in Section 2.2 and two additional tasks that share identical instructions (Appendix A.2). TRACE (Wang et al., 2023b) contains eight tasks covering domain-specific knowledge, multilingual capabilities, code generation, and mathematical reasoning.

Compared methods. We evaluate InfLoRA (Liang & Li, 2024), O-LoRA (Wang et al., 2023a), DualPrompt (Wang et al., 2022b), and InK with and without GroupRoute to assess its effect across different continual learning methods. We compare these combinations with the whole-prompt routing methods EPI (Wang et al., 2023c), SAPT (Zhao et al., 2024), and GainLoRA (Liang et al., 2025). EWC (IA^3) applies EWC (Kirkpatrick et al., 2017) to the IA^3 adapter used by InK and serves as its ablation without instruction conditioning. Sequential fine-tuning with IA^3 (Liu et al., 2022) and LoRA (Hu et al., 2022) serves as a baseline for learning without explicit protection against forgetting. We also report joint-training and task-ID oracle references. Joint training has access to all training tasks simultaneously, while the task-ID oracles receive task identifiers at inference.

Evaluation metrics. We use exact-match accuracy for the long-sequence benchmark and ROUGE-L for SuperNI-42. Following TRACE's evaluation protocol, we report accuracy on C-STANCE, FOMC, ScienceQA, and the two NumGLUE tasks, ROUGE-L on MeetingBank, SARI on 20Minuten, and edit similarity (ES) on Py150. Let $R_{t,i}$ denote the score on task i, expressed on a 0–100 scale, after learning task t. We report average performance (AP) and forgetting (F) after the final task, defined as

$$\mathrm{AP} = \frac{1}{T} \sum_{i=1}^{T} R_{T,i}, \qquad F = \frac{1}{T-1} \sum_{i=1}^{T-1} \left(\max_{t \in \{i, \ldots, T-1\}} R_{t,i} - R_{T,i} \right). \tag{6}$$

Forgetting measures the mean decline from earlier tasks' peak scores to their final scores (Chaudhry et al., 2018). Both metrics are reported in score points.

Table 2: Continual instruction tuning on Gemma-2B-it. Shaded rows report results obtained with GroupRoute. Bold marks the best results among continual learning methods, including ties; joint-training and task-ID references are excluded.

	15-Task (O3)		15-Task (O4)		SuperNI-42		TRACE	
Method	**AP** ↑	***F*** ↓	**AP** ↑	***F*** ↓	**AP** ↑	***F*** ↓	**AP** ↑	***F*** ↓
Joint training and task-ID references								
Multi-task learning (IA^3)	79.49	–	79.49	–	44.98	–	51.32	–
Multi-task learning (LoRA)	79.58	–	79.58	–	44.92	–	49.49	–
Task-ID oracle (IA^3)	81.56	–	81.56	–	47.90	–	54.99	–
Task-ID oracle (LoRA)	81.18	–	81.18	–	48.17	–	53.36	–
Sequential fine-tuning								
Seq-IA^3	13.30	73.86	10.51	77.57	29.75	18.98	29.35	25.13
Seq-LoRA	11.34	72.14	6.51	79.71	18.74	28.05	20.58	34.04
Whole-prompt routing								
EPI	79.35	**0.00**	79.23	**0.00**	40.70	2.80	53.09	**0.00**
SAPT‡	77.55	4.00	77.68	3.14	36.99	10.20	52.26	0.58
GainLoRA (InfLoRA)	37.22	27.71	43.62	36.57	30.81	9.29	43.08	4.98
Applying GroupRoute to existing methods								
InfLoRA	75.69	2.14	78.09	4.00	31.15	15.45	44.03	2.21
+ GroupRoute	80.33	**0.00**	80.34	**0.00**	46.74	0.42	48.00	**0.00**
O-LoRA	29.49	54.86	26.30	57.57	24.54	22.16	28.29	23.44
+ GroupRoute	80.87	0.14	81.41	0.86	45.37	2.57	52.04	1.41
DualPrompt	65.07	7.14	66.76	6.29	26.87	11.00	49.62	1.93
+ GroupRoute	79.80	0.14	79.68	**0.00**	46.12	1.31	53.01	0.35
InK and its ablation								
EWC (IA^3)	60.27	22.86	67.02	17.57	33.12	13.45	45.85	5.98
InK (single adapter)	75.19	4.14	77.45	4.29	43.50	4.60	47.89	3.82
+ GroupRoute (full method)	**81.43**	**0.00**	**81.47**	0.40	**47.37**	**0.41**	**53.29**	2.99

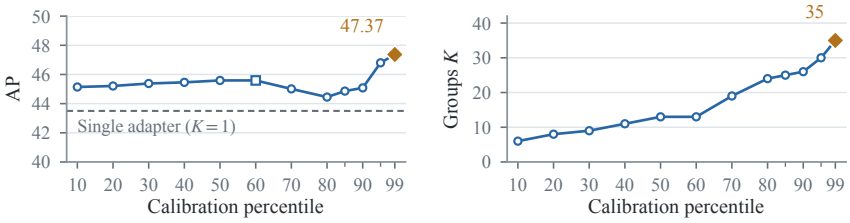
‡SAPT uses pseudo-sample replay; the other continual learning methods are replay-free. EWC (IA^3) and single-adapter InK store one fixed-size adapter (0.97M and 1.94M parameters); the other continual learning methods store parameters for each task (Appendix D.2). InK stores each task's instruction text rather than its key, which the frozen backbone recomputes deterministically at runtime.

Backbone and training protocol. Main experiments use Gemma-2B-it; results for LLaMA-2-7B and T5-large appear in Appendix D.1. GroupRoute uses the instruction representation sites and layers identified in Section 2.1 (Figure 2). We train for 10 epochs per task on the long-sequence benchmark and SuperNI-42, and five on TRACE. Learning rates and the InK regularization strength are selected on validation splits. Appendix C provides details of validation-set construction, together with optimizer and adapter settings, search grids, and selected hyperparameters.

4.2 Continual Instruction Tuning Performance

Table 2 shows that GroupRoute improves average performance and reduces forgetting for InfLoRA, O-LoRA, DualPrompt, and InK across all evaluated benchmarks on Gemma-2B-it. For example, on SuperNI-42, applying GroupRoute to InfLoRA raises AP from 31.15 to 46.74 while reducing forgetting from 15.45 to 0.42. These improvements extend across different adaptation paradigms, including low-rank adaptation with InfLoRA and O-LoRA, prompt tuning with DualPrompt, and activation scaling with InK, demonstrating the broad applicability of GroupRoute.

The comparison between EWC (IA^3) and single-adapter InK examines the contribution of instruction-keyed units without GroupRoute. Both methods learn the task sequence with a single adapter, and InK achieves higher AP and lower forgetting in every evaluated setting. On the long-sequence benchmark, instruction conditioning increases AP by 14.92 and 10.43 points under orders O3 and O4, respectively, while reducing forgetting by more than 75% under both orders. On SuperNI-42, it raises AP from 33.12 to 43.50 and reduces forgetting from 13.45 to 4.60.



Percentile	10	20	30	40	50	60	70	80	85	90	95	99
τ_{join}	0.1879	0.2473	0.2888	0.3240	0.3582	0.3948	0.4360	0.4863	0.5110	0.5586	0.6255	0.7485

Figure 3: Sensitivity to τ_{join} for InK with GroupRoute on SuperNI-42 (Gemma-2B-it, $\lambda = 300$). Thresholds are listed below the panels. The dashed line shows single-adapter InK (43.50 AP). Gold marks the default p99 setting.

Single-adapter InK is also competitive with methods that store parameters for each task. On SuperNI-42, it reaches 43.50 AP with 1.94M adapter parameters, above every compared method without GroupRoute, including EPI (40.70 AP, 49.56M), SAPT (36.99 AP, 19.32M), and InfLoRA (31.15 AP, 38.64M). The same holds on LLaMA-2-7B, where InK reaches 47.78 AP against 40.45 for SAPT (Appendix D.1).

With GroupRoute, InK achieves the highest AP among the compared continual learning methods across all evaluated benchmarks on Gemma-2B-it. On SuperNI-42, where tasks share source corpora, it reaches 47.37 AP compared with 40.70 for EPI, the strongest whole-prompt routing baseline. Moreover, it retains *47.36* AP when every instruction is reworded at inference. Its long-sequence performance is consistent across orders O3 and O4 (81.43 and 81.47 AP).

GroupRoute also reduces routing computation: on Gemma-2B-it, it selects the group from instruction tokens at layer index 1, whereas EPI processes the whole prompt through the final layer.

Appendix D.2 reports the adapter storage for each method. Each InK group uses at most 1.94M adapter parameters and fixed-size consolidation storage, regardless of how many tasks join that group. Total adapter storage therefore scales with the number of groups rather than tasks, and the joining threshold controls this tradeoff. Figure 3 shows that lowering the calibration percentile from p99 to p50 on SuperNI-42 reduces the number of groups from 35 to 13 (a 63% reduction) while retaining 45.59 AP, well above all whole-prompt routing baselines.

4.3 Retention of Pretrained Capabilities

Following the TRACE protocol (Wang et al., 2023b), we evaluate retention of pretrained capabilities after sequential training on the eight TRACE tasks, using Gemma-2B-it as the backbone. We assess retention on BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), MMLU (Hendrycks et al., 2021), BBH (Suzgun et al., 2023), GSM8K (Cobbe et al., 2021), and TyDiQA (Clark et al., 2020), none of which is included in the continual training sequence. We report ΔGA, the mean score change across these six benchmarks relative to pretrained Gemma-2B-it.

Table 3 shows that GroupRoute improves retention of pretrained capabilities across InfLoRA, O-LoRA, DualPrompt, and InK. With GroupRoute, ΔGA ranges from $+0.14$ to $+0.35$, indicating that average performance remains close to the pretrained reference. For the comparison methods without GroupRoute, the average score is 1.80–14.59 points below the pretrained reference. For InK, instruction conditioning improves ΔGA from -3.26 for the EWC (IA^3) ablation to -1.80, the smallest loss without GroupRoute, and applying GroupRoute further raises it to $+0.17$.

GroupRoute preserves the pretrained scores exactly on BoolQ, PIQA, MMLU, and GSM8K by routing their queries to the frozen backbone under the rule in Section 3.1. Performance on BBH also remains close to the pretrained reference, with decreases of 0.38–0.44 points.

On TyDiQA, English instructions select the learned question-answering group, while non-English queries continue to use the frozen backbone. With GroupRoute, InfLoRA, O-LoRA, DualPrompt, and InK improve TyDiQA scores over the pretrained reference by 2.38, 1.27, 2.51, and 1.46 points, respectively, demonstrating positive transfer to an unseen task.

Table 3: Pretrained capability retention after TRACE training on Gemma-2B-it, with the scores of the pretrained model as reference. $\Delta\mathrm{GA}$ is the mean change from these scores, in score points. Shaded rows report results obtained with GroupRoute; bold marks the best $\Delta\mathrm{GA}$.

Method	ΔGA ↑	BoolQ	PIQA	MMLU	BBH	GSM8K	TyDiQA
Pretrained Gemma-2B-it	–	63.61	75.19	37.00	31.10	9.93	33.47
Whole-prompt routing							
EPI	−8.62	50.80	65.34	29.88	23.36	3.34	25.88
SAPT	−2.50	59.36	72.91	35.22	27.05	5.61	35.18
GainLoRA (InfLoRA)	−11.40	47.98	69.04	28.67	6.20	2.05	27.96
Applying GroupRoute to existing methods							
InfLoRA	−14.59	62.26	68.17	23.79	3.89	1.21	3.46
+ GroupRoute	+0.32	63.61	75.19	37.00	30.66	9.93	35.85
O-LoRA	−10.41	57.46	70.40	24.11	11.20	2.50	22.18
+ GroupRoute	+0.14	63.61	75.19	37.00	30.66	9.93	34.74
DualPrompt	−7.42	49.63	67.03	31.09	18.92	4.17	34.95
+ GroupRoute	**+0.35**	63.61	75.19	37.00	30.72	9.93	35.98
InK and its ablation							
EWC (IA^3)	−3.26	52.91	74.10	37.10	24.04	5.31	37.30
InK (single adapter)	−1.80	58.04	74.65	36.35	25.14	5.53	39.78
+ GroupRoute (full method)	+0.17	63.61	75.19	37.00	30.66	9.93	34.93

5 Related Work

Several continual learning methods use representations of the whole input to select prompts or task-specific parameters (Wang et al., 2022c;b). EPI (Wang et al., 2023c) retrieves prefixes using Mahalanobis distance, SAPT (Zhao et al., 2024) uses shared attention to combine task-specific parameters during training and inference, and GainLoRA (Liang et al., 2025) learns weights from the whole prompt to combine LoRA branches. Such representations couple the instruction with the input text. In representation learning, INSTRUCTOR (Su et al., 2023) shows that task descriptions can guide text representations, but it trains a dedicated embedding model for this purpose. While Li et al. (2024) derive instruction embeddings for task identification from last-token hidden states in the final layers, GroupRoute averages frozen early-layer activations over the instruction tokens.

Task similarity has also been used to organize sharing in continual learning (Ke et al., 2020). AdaPromptCL (Kim et al., 2024) forms and refines prompt-sharing groups using representations obtained through warm-up adaptation. In continual instruction tuning (Yin et al., 2022; Scialom et al., 2022), InsCL (Wang et al., 2024) uses instruction similarity to allocate replay across previously learned tasks. Qiao et al. (2025) use TF-IDF instruction matching to guide parameter reuse and selection alongside adaptive EMA updates (see Appendix A.3 for comparison under instruction rewording). GroupRoute uses instruction representations from the pretrained backbone to group tasks and route queries, independently of the continual learning method used within each group.

In multi-task learning, HyperFormer (Karimi Mahabadi et al., 2021) generates adapter parameters from learned task embeddings using shared hypernetworks. InK uses instruction representations extracted from the pretrained backbone to condition IA^3 (Liu et al., 2022) activation scaling, without a hypernetwork. To protect earlier tasks, InK extends online EWC (Kirkpatrick et al., 2017; Schwarz et al., 2018) with a Fisher penalty tailored to its instruction-conditioned activation scales.

6 Conclusion

We showed that instruction representations from the earliest layers of frozen language models reliably distinguish tasks and remain robust to instruction rewording. Building on this finding, we introduced GroupRoute, which uses these representations for online grouping and routing without trainable routing parameters. We further introduced InK, which combines instruction-conditioned IA^3 scaling with Fisher consolidation to learn successive tasks in a single fixed-size adapter. Experiments on the long-sequence benchmark, SuperNI-42, and TRACE show that GroupRoute improves performance and reduces forgetting across different continual learning methods. GroupRoute also preserves pretrained capabilities by retaining the frozen backbone's behavior when instruction similarity is low, while enabling positive transfer through relevant learned adaptations.

REPRODUCIBILITY STATEMENT

Section 2.1 and Figure 2 specify the instruction representation sites and layers for each backbone, and Section 3.1 gives the joining thresholds; Appendix A.1 describes the PromptSource corpus used to select them. Section 3.2 defines InK and its Fisher penalty. Appendix C lists the training protocol, task orders, validation-set construction, adapter configurations, search grids, and the selected hyperparameters for every method, backbone, and benchmark. The SuperNI-42 benchmark is constructed in this work from Super-NaturalInstructions tasks; Appendix A.2 lists its tasks and describes the instruction rewordings, and Appendix B reports the resulting task groups. We will release the source code, the complete SuperNI-42 benchmark, and all instruction rewordings upon publication.

AI USE STATEMENT

We used generative AI tools as coding assistants for implementing and debugging code, to draft the reworded instructions of SuperNI-42, which the authors checked by hand, to find related work, and to improve the writing of the manuscript (grammar, clarity, and readability). We did not use them for research ideation, method or experimental design, or interpretation of results. We reviewed all AI-assisted work and take full responsibility for the final content of this work.

REFERENCES

Stephen H. Bach, Victor Sanh, Zheng-Xin Yong, Albert Webson, Colin Raffel, Nihal V. Nayak, Abheesht Sharma, Taewoon Kim, M Saiful Bari, Thibault Fevry, et al. PromptSource: An integrated development environment and repository for natural language prompts. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 93–104, 2022. doi: 10.18653/v1/2022.acl-demo.9. URL https://aclanthology.org/2022.acl-demo.9/.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7432–7439, 2020. doi: 10.1609/aaai.v34i05.6239. URL https://ojs.aaai.org/index.php/AAAI/article/view/6239.

Arslan Chaudhry, Puneet K. Dokania, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Riemannian Walk for incremental learning: Understanding forgetting and intransigence. In *European Conference on Computer Vision*, pp. 556–572, 2018. doi: 10.1007/978-3-030-01252-6_33. URL https://link.springer.com/chapter/10.1007/978-3-030-01252-6_33.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2924–2936, 2019. doi: 10.18653/v1/N19-1300. URL https://aclanthology.org/N19-1300/.

Jonathan H. Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. TyDi QA: A benchmark for information-seeking question answering in typologically diverse languages. *Transactions of the Association for Computational Linguistics*, 8:454–470, 2020. doi: 10.1162/tacl_a_00317. URL https://aclanthology.org/2020.tacl-1.30/.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/abs/2110.14168.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, 2019. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423/.

Gemma Team. Gemma: Open models based on Gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024. URL https://arxiv.org/abs/2403.08295.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Ting Jiang, Shaohan Huang, Zhongzhi Luan, Deqing Wang, and Fuzhen Zhuang. Scaling sentence embeddings with large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 3182–3196, 2024. doi: 10.18653/v1/2024.findings-emnlp.181. URL https://aclanthology.org/2024.findings-emnlp.181/.

Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 565–576, 2021. doi: 10.18653/v1/2021.acl-long.47. URL https://aclanthology.org/2021.acl-long.47/.

Zixuan Ke, Bing Liu, and Xingchang Huang. Continual learning of a mixed sequence of similar and dissimilar tasks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 18493–18504, 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/hash/d7488039246a405baf6a7cbc3613a56f-Abstract.html.

Doyoung Kim, Susik Yoon, Dongmin Park, Youngjun Lee, Hwanjun Song, Jihwan Bang, and Jae-Gil Lee. One size fits all for semantic shifts: Adaptive prompt tuning for continual learning. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 24658–24673, 2024. URL https://proceedings.mlr.press/v235/kim24ai.html.

Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. URL https://arxiv.org/abs/1412.6980.

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. doi: 10.1073/pnas.1611835114. URL https://doi.org/10.1073/pnas.1611835114.

Yiwei Li, Jiayi Shi, Shaoxiong Feng, Peiwen Yuan, Xinglin Wang, Boyuan Pan, Heda Wang, Yao Hu, and Kan Li. Instruction embedding: Latent representations of instructions towards task identification. In *Advances in Neural Information Processing Systems*, volume 37, pp. 87683–87711, 2024. doi: 10.52202/079017-2784. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/9f9e2982759f384495f4b75a33f3dd72-Abstract-Datasets_and_Benchmarks_Track.html.

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. doi: 10.1109/CVPR52733.2024.02231. URL https://openaccess.thecvf.com/content/CVPR2024/html/Liang_InfLoRA_Interference-Free_Low-Rank_Adaptation_for_Continual_Learning_CVPR_2024_paper.html.

Yan-Shuo Liang, Jia-Rui Chen, and Wu-Jun Li. Gated integration of low-rank adaptation for continual learning of large language models. In *Advances in Neural Information Processing Systems*, volume 38, pp. 76577–76607, 2025. doi: 10.52202/085713-2310. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/63692d8d567db671c700df5df912204a-Abstract-Conference.html.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 1950–1965, 2022. doi: 10.52202/068431-0142. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/0cde695b83bd186c1fd456302888454c-Abstract-Conference.html.

Zhu Liu, Cunliang Kong, Ying Liu, and Maosong Sun. Fantastic semantics and where to find them: Investigating which layers of generative LLMs reflect lexical semantics. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 14551–14558, 2024. doi: 10.18653/v1/2024.findings-acl.866. URL https://aclanthology.org/2024.findings-acl.866/.

Jingyang Qiao, Zhizhong Zhang, Xin Tan, Yanyun Qu, Shouhong Ding, and Yuan Xie. Large continual instruction assistant. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 50189–50213. PMLR, 2025. URL https://proceedings.mlr.press/v267/qiao25e.html.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL https://jmlr.org/papers/v21/20-074.html.

Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, and Amjad Almahairi. Progressive Prompts: Continual learning for language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=UJTgQBc91_.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3982–3992, 2019. doi: 10.18653/v1/D19-1410. URL https://aclanthology.org/D19-1410/.

Jonathan Schwarz, Wojciech M. Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & Compress: A scalable framework for continual learning. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4528–4537. PMLR, 2018. URL https://proceedings.mlr.press/v80/schwarz18a.html.

Thomas Scialom, Tuhin Chakrabarty, and Smaranda Muresan. Fine-tuned language models are continual learners. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 6107–6122, 2022. doi: 10.18653/v1/2022.emnlp-main.410. URL https://aclanthology.org/2022.emnlp-main.410/.

Hongjin Su, Weijia Shi, Jungo Kasai, Yizhong Wang, Yushi Hu, Mari Ostendorf, Wen-tau Yih, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. One embedder, any task: Instruction-finetuned text embeddings. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 1102–1121, 2023. doi: 10.18653/v1/2023.findings-acl.71. URL https://aclanthology.org/2023.findings-acl.71/.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13003–13051, 2023. doi: 10.18653/v1/2023.findings-acl.824. URL https://aclanthology.org/2023.findings-acl.824/.

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. URL https://arxiv.org/abs/2307.09288.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuanjing Huang. Orthogonal subspace learning for language model continual learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 10658–10671, 2023a. doi: 10.18653/v1/2023.findings-emnlp.715. URL https://aclanthology.org/2023.findings-emnlp.715/.

Xiao Wang, Yuansen Zhang, Tianze Chen, Songyang Gao, Senjie Jin, Xianjun Yang, Zhiheng Xi, Rui Zheng, Yicheng Zou, Tao Gui, Qi Zhang, and Xuanjing Huang. TRACE: A comprehensive benchmark for continual learning in large language models. *arXiv preprint arXiv:2310.06762*, 2023b. doi: 10.48550/arXiv.2310.06762. URL https://arxiv.org/abs/2310.06762.

Yifan Wang, Yafei Liu, Chufan Shi, Haoling Li, Chen Chen, Haonan Lu, and Yujiu Yang. InsCL: A data-efficient continual learning paradigm for fine-tuning large language models with instructions. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 663–677. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.naacl-long.37. URL https://aclanthology.org/2024.naacl-long.37/.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, et al. Super-NaturalInstructions: Generalization via declarative instructions on 1600+ NLP tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 5085–5109, 2022a. doi: 10.18653/v1/2022.emnlp-main.340. URL https://aclanthology.org/2022.emnlp-main.340/.

Zhicheng Wang, Yufang Liu, Tao Ji, Xiaoling Wang, Yuanbin Wu, Congcong Jiang, Ye Chao, Zhencong Han, Ling Wang, Xu Shao, and Wenqiu Zeng. Rehearsal-free continual language learning via efficient parameter isolation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10933–10946, 2023c. doi: 10.18653/v1/2023.acl-long.612. URL https://aclanthology.org/2023.acl-long.612/.

Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. DualPrompt: Complementary prompting for rehearsal-free continual learning. In *European Conference on Computer Vision*, pp. 631–648, 2022b. doi: 10.1007/978-3-031-19809-0_36. URL https://link.springer.com/chapter/10.1007/978-3-031-19809-0_36.

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. doi: 10.1109/CVPR52688.2022.00024. URL https://openaccess.thecvf.com/content/CVPR2022/html/Wang_Learning_To_Prompt_for_Continual_Learning_CVPR_2022_paper.html.

Wenpeng Yin, Jia Li, and Caiming Xiong. ConTinTin: Continual learning from task instructions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3062–3072, 2022. doi: 10.18653/v1/2022.acl-long.218. URL https://aclanthology.org/2022.acl-long.218/.

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. doi: 10.18653/v1/2024.acl-long.625. URL https://aclanthology.org/2024.acl-long.625/.

A Instruction Representations

This section provides details on the instruction representations studied in Section 2.

A.1 PromptSource Calibration Corpus

We build the calibration corpus from PromptSource (Bach et al., 2022). Each PromptSource dataset defines one task identity, and its prompt templates serve as independently authored wordings of that task. From each template, we keep the text before the target separator, replace answer choices with `[options]` and input fields with `[input]`, and discard templates containing control logic, templates not marked as the dataset's original task, and exact duplicates. Identities with fewer than two wordings are removed. To separate the corpus from the evaluation benchmarks, we remove all GLUE and SuperGLUE datasets and the datasets of the remaining long-sequence tasks (IMDB, Yelp, Amazon reviews, AG News, DBpedia, and Yahoo Answers topics), 31 datasets in total. The resulting corpus contains 928 wordings from 165 task identities. It contains no SuperNI-42 instruction; PromptSource datasets that share a source corpus with SuperNI-42 tasks, such as DuoRC, Quoref, and RACE, are retained with their own templates.

A.2 SuperNI-42 Tasks and Rewordings

Table 4 lists the 42 Super-NaturalInstructions tasks (Wang et al., 2022a) in the SuperNI-42 continual instruction tuning benchmark, organized by source corpus and requested operation. The 40 unmarked tasks form the task-identification evaluation in Section 2.2.

Table 4: Tasks in SuperNI-42, listed by source corpus. Task IDs refer to Super-NaturalInstructions.

Source corpus	Task IDs and descriptions
CODA-19	1161 title generation; 1162 title verification
DuoRC	182 question-and-answer generation; 193 question generation; 194 answer generation
HealthFact	1368 claim generation; 1369 explanation generation
HotpotQA	170 answer generation; 191 question generation; 192 supporting-fact generation
IIRC	230 passage selection; 231 linked-term selection; 232 multiple-link requirement classification; 236 question generation
IMDB	284 sentiment classification; 285 sentiment verification
Jeopardy	305 answer generation (normal); 306* answer generation (double); 307* answer generation (final); 308 answer generation (all)
MRQA	469 answer extraction; 470 question generation
PerSent	418 title generation; 419 main-entity identification; 420 sentiment classification; 423 sentiment verification
Quoref	001 question generation; 002 answer generation
RACE	309 multiple-choice answering; 310 answer verification; 311 question generation; 649 fill-in-the-blank question generation
Reddit TIFU	510 title generation; 511 summarization
SciFact	1586 title generation; 1587 title verification
Scruples	498 blame classification; 500 title generation; 501 post-type verification; 502 blame verification
XLSum	1356 title generation; 1357 summarization

*Tasks 306 and 307 share the same original instruction as task 308 and are marked with an asterisk. They are the two additional tasks included in the 42-task continual instruction-tuning sequence in Section 4.

For task identification under rewording (Section 2.2), each task definition has eight alternative wordings that preserve the requested operation: two clause reorderings, three lexical substitutions, two changes of framing or voice, and one compression that removes framing words only. Option strings, label vocabulary, input references, answer format, and every stated constraint are kept verbatim. The word Jaccard similarity between each rewording and the original definition lies between 0.30 and

0.85: the upper bound requires every rewording to change a substantial part of the wording, and the lower bound excludes changes that go beyond rewording. Each rewording is 0.60 to 1.50 times the length of the original. Independently authored wordings, which differ more from each other, are evaluated on the PromptSource corpus in Section 2.1. Only the definition changes; the rest of the prompt and the input text are unchanged. Table 5 shows the eight rewordings of one definition.

Table 5: Original definition and its eight rewordings for task 1357 (XLSum summarization).

Type	Wording
Original	Generate an appropriate single-sentence summary for the given text such that it includes the main topic of the text.
Reorder	The summary must include the main topic of the text: produce an appropriate single-sentence summary for the given text.
Reorder	For the given text, write a suitable single-sentence summary such that it includes the main topic of the text.
Lexical	Generate a suitable single-sentence summary for the given text such that it covers the main topic of the text.
Lexical	Generate an appropriate one-sentence summary for the given passage such that it includes the main subject of the passage.
Lexical	Produce an appropriate single-sentence summary of the given text that captures the main topic of the text.
Frame	You are given a text. Your task is to generate an appropriate single-sentence summary for it such that it includes the main topic of the text.
Frame	The task is to generate an appropriate single-sentence summary for the given text; the summary has to include the main topic of the text.
Compress	Generate an appropriate single-sentence summary of the given text including its main topic.

A.3 Comparison with TF-IDF Instruction Matching

We compare early-layer instruction representations with the TF-IDF instruction matching of Qiao et al. (2025), which we refer to as LCIA. The evaluation follows Section 2.2, using 40 original instructions and 320 rewordings without task grouping (Table 6).

Table 6: Task-identification accuracy (%) on original and reworded instructions across the 40 Super-NaturalInstructions tasks.

Approach	Original (%)	Reworded (%)
Early-layer instruction representations, Gemma-2B-it	100.00	99.38
Early-layer instruction representations, LLaMA-2-7B	100.00	99.69
Early-layer instruction representations, T5-large	100.00	99.38
LCIA instruction matching (TF-IDF)	100.00	95.94

While both approaches achieve 100.00% accuracy on original instructions, LCIA's TF-IDF instruction matching drops to 95.94% under rewording. In contrast, early-layer instruction representations retain 99.38%–99.69% accuracy across all three backbones, demonstrating greater robustness to lexical variation in instruction wording.

B GroupRoute Task Groups on Gemma-2B-it

We report the task groups formed by GroupRoute on Gemma-2B-it (Table 7) using the default joining threshold $\tau_{\text{join}} = 0.7485$ from Section 3.1. For groups containing multiple tasks, we examine the similarities and differences between their instructions.

Table 7: Task groups formed by GroupRoute on Gemma-2B-it at the default joining threshold. The long-sequence counts apply to both orders O3 and O4.

Benchmark	Tasks	Groups	Groups containing multiple tasks	Single-task groups
Long-sequence benchmark	15	14	1	13
SuperNI-42	42	35	4	31
TRACE	8	7	1	6

Long-sequence benchmark. BoolQ and MultiRC are the only tasks grouped together on the long-sequence benchmark, producing 14 groups under both orders O3 and O4. Their instruction prototypes have cosine similarity 0.928, and Table 8 lists their instructions.

Table 8: Instructions for BoolQ and MultiRC, which GroupRoute assigns to the same group under both orders O3 and O4.

Task	Extracted instruction
BoolQ	According to the following passage, is the question true or false? Choose one from the option.
MultiRC	According to the following passage and question, is the candidate answer true or false? Choose one from the option.

Both instructions require a true-or-false judgment based on a passage and a question. MultiRC additionally provides a candidate answer whose correctness must be evaluated. The two instructions share nearly identical wording and identical output choices.

TRACE. On TRACE, the two NumGLUE tasks use the identical instruction, "Solve the following math problem." Their instruction prototypes have cosine similarity 1.0000, and GroupRoute assigns them to the same group. The remaining six tasks each form a separate group.

SuperNI-42. On SuperNI-42, GroupRoute forms four groups containing multiple tasks and 31 single-task groups. Table 9 lists these four groups, reports instruction-prototype similarities, and compares the instructions within each group.

Table 9: Groups containing multiple tasks on SuperNI-42. Each row lists one group, with tasks identified by their Super-NaturalInstructions IDs. Cosine similarities are reported for selected pairs of instruction prototypes.

Tasks in the same group	Cosine similarity	Instruction comparison
Jeopardy 305, 306, 307, 308	Three pairs at 1.000; three at 0.968	All four instructions ask for a lowercase answer from a trivia clue and category. Three are identical; the fourth replaces "For simplicity" with "For consistency."
HotpotQA 170, 191, 192	170–191: 0.9411; 170–192: 0.7679	All three instructions refer to context paragraphs, asking for an answer (170), a question (191), or a supporting fact (192). Tasks 191 and 192 join the group of task 170.
MRQA 469, 470	0.8070	Both instructions specify answers as continuous spans from a passage. Task 469 asks for the answer to a given question; task 470 asks for a question with such an answer.
DuoRC 182, 193	0.9087	Both instructions ask for factoid questions answerable from movie plots. Task 182 requests both the question and answer, while task 193 requests only the question.

C Experimental Setup

This section details the experimental setup of Section 4.1.

C.1 Benchmarks and Task Orders

For the long-sequence benchmark, validation examples are drawn from the source datasets, excluding examples in the benchmark's training and test sets. For SuperNI-42, each task's instances are split into 1,000 training, 200 validation, and 500 test examples using a fixed random seed. For TRACE, we remove duplicate prompts and training/test overlap from the official validation data, then supplement it with held-out training examples.

We use two task orders. O3 consists of Yelp, Amazon, MNLI, CB, COPA, QQP, RTE, IMDB, SST-2, DBpedia, AG News, Yahoo, MultiRC, BoolQ, and WiC (Order 6 of O-LoRA, Order 10 of Progressive Prompts). O4 consists of MNLI, CB, WiC, COPA, QQP, BoolQ, RTE, IMDB, Yelp, Amazon, SST-2, DBpedia, AG News, MultiRC, and Yahoo (Order 4 of O-LoRA, Order 8 of Progressive Prompts). SuperNI-42 tasks are presented in a fixed random order, given by their task IDs in Table 4: 1587, 1162, 170, 309, 501, 232, 510, 308, 1356, 469, 305, 182, 418, 285, 191, 307, 498, 500, 423, 192, 306, 284, 311, 502, 419, 236, 230, 649, 470, 420, 001, 1161, 310, 1357, 1368, 1586, 193, 194, 231, 002, 1369, 511. TRACE follows the benchmark's official order.

C.2 Training and Evaluation Protocol

All methods use Adam (Kingma & Ba, 2015) with $\beta_1 = 0.9$ and $\beta_2 = 0.95$. Table 10 summarizes the benchmark-specific training settings.

Table 10: Benchmark-specific training settings.

Setting	Long-sequence benchmark	SuperNI-42	TRACE
Epochs per task	10	10	5
Effective training batch size	32	32	8
Maximum sequence length (whole prompt / target)	512 / 50	512 / 50	1024 / 512
Validation examples per task	500	200	500

Within the stated epoch budget, we select the checkpoint with the highest score on the current task's full validation split, using exact match on the long-sequence benchmark, ROUGE-L on SuperNI-42, and each task's native metric on TRACE.

Evaluation uses a batch size of 32 across all benchmarks, with all sequence lengths measured in tokens. We use label smoothing of 0.1 for LLaMA-2-7B and cross-entropy loss without label smoothing for Gemma-2B-it and T5-large. No method receives task identity in its input; in particular, we do not prepend the task and dataset names that the official O-LoRA scripts add to every prompt.

C.3 Method Configurations

Table 11 summarizes the adapter and GroupRoute configurations. Layer indices are zero-based.

Table 11: Adapter and GroupRoute configurations.

Method	Configuration
IA^3-based methods	IA^3 adapters on all linear layers in every Transformer block (all attention and feed-forward projections).
LoRA-based methods	Rank $r = 8$, scaling coefficient $\alpha = 32$, and dropout rate 0, applied to the query and value projections. SAPT uses the configuration specified in the SAPT row.
DualPrompt	G-prompt length 5 on layers 0–2; E-prompt length 10 on every remaining layer (3–17 on Gemma-2B-it, 3–31 on LLaMA-2-7B, 3–23 of the T5-large encoder); top-$k = 2$; pull-constraint weight 1.0.
EPI	Prefix length 16 and at most 20 class prototypes.
GainLoRA (InfLoRA)	Rank-8 LoRA branches, gating-network hidden width 100, and a feature-energy threshold of 95%.
SAPT	LoRA rank from the official SAPT scripts ($r = 8$ for T5-large on the long-sequence benchmark, $r = 4$ otherwise, following its LLaMA scripts where none exists); $\alpha = 32$, query-projection width 100, and pseudo-samples (2% of each task's training set) generated by a per-task generator LoRA on the frozen backbone, as in the official implementation.

GroupRoute uses the instruction representation sites and layer ranges selected in Section 2.1 and the joining thresholds specified in Section 3.1. Appendix B reports the resulting task groups on Gemma-2B-it.

C.4 Hyperparameter Selection

For each method, learning rates are selected from

$$\{10^{-4},\ 3\times10^{-4},\ 10^{-3},\ 3\times10^{-3},\ 10^{-2},\ 3\times10^{-2}\}.$$

For EWC (IA^3) and InK, the regularization strength is selected from

$$\{30,\ 100,\ 300,\ 1000,\ 3000\}.$$

The learning rate and, for EWC (IA^3) and InK, the regularization strength are selected using validation performance across the full task sequence. The same values are then used throughout the task sequence.

Tables 12–14 report the selected hyperparameters for each backbone and benchmark. Entries give the learning rate, except for EWC (IA^3) and InK, for which they give the pair (learning rate, regularization coefficient λ). This convention also applies to InK with GroupRoute. The long-sequence settings apply to both orders O3 and O4.

Multi-task learning and task-ID oracle models use the same learning rate for a given adapter type.

Table 12: Selected hyperparameters for Gemma-2B-it. An indented "+ GroupRoute" row refers to the method immediately above it. Entries follow the convention described in this subsection.

Method	Long sequence, O3 and O4	SuperNI-42	TRACE
Sequential fine-tuning			
Seq-IA^3	3×10^{-3}	3×10^{-3}	10^{-3}
Seq-LoRA	3×10^{-4}	3×10^{-4}	3×10^{-4}
Whole-prompt routing			
EPI	10^{-2}	10^{-2}	10^{-2}
SAPT	3×10^{-4}	3×10^{-4}	10^{-4}
GainLoRA (InfLoRA)	3×10^{-4}	3×10^{-4}	3×10^{-4}
Applying GroupRoute to existing methods			
InfLoRA	3×10^{-4}	3×10^{-4}	3×10^{-4}
+ GroupRoute	3×10^{-4}	3×10^{-4}	3×10^{-4}
O-LoRA	3×10^{-4}	3×10^{-4}	3×10^{-4}
+ GroupRoute	3×10^{-4}	3×10^{-4}	3×10^{-4}
DualPrompt	10^{-2}	10^{-2}	10^{-2}
+ GroupRoute	3×10^{-2}	3×10^{-2}	10^{-2}
InK and its ablation			
EWC (IA^3)	$(3 \times 10^{-3}, 300)$	$(3 \times 10^{-3}, 300)$	$(10^{-3}, 100)$
InK (single adapter)	$(3 \times 10^{-3}, 300)$	$(3 \times 10^{-3}, 300)$	$(10^{-3}, 100)$
+ GroupRoute (full method)	$(3 \times 10^{-3}, 300)$	$(3 \times 10^{-3}, 300)$	$(10^{-3}, 100)$
Joint training and task-ID references			
Multi-task learning / task-ID oracle (IA^3)	3×10^{-3}	3×10^{-3}	10^{-3}
Multi-task learning / task-ID oracle (LoRA)	3×10^{-4}	3×10^{-4}	3×10^{-4}

Table 13: Selected hyperparameters for LLaMA-2-7B. An indented "+ GroupRoute" row refers to the method immediately above it. Entries follow the convention described in this subsection.

Method	Long sequence, O3 and O4	SuperNI-42	TRACE
Sequential fine-tuning			
Seq-IA^3	3×10^{-3}	3×10^{-3}	3×10^{-4}
Seq-LoRA	3×10^{-4}	3×10^{-4}	3×10^{-4}
Whole-prompt routing			
EPI	10^{-2}	10^{-2}	10^{-2}
SAPT	3×10^{-4}	3×10^{-4}	10^{-4}
GainLoRA (InfLoRA)	3×10^{-4}	3×10^{-4}	3×10^{-4}
Applying GroupRoute to existing methods			
InfLoRA	3×10^{-4}	3×10^{-4}	3×10^{-4}
+ GroupRoute	3×10^{-4}	3×10^{-4}	3×10^{-4}
O-LoRA	3×10^{-4}	3×10^{-4}	3×10^{-4}
+ GroupRoute	3×10^{-4}	3×10^{-4}	3×10^{-4}
DualPrompt	10^{-2}	10^{-2}	10^{-2}
+ GroupRoute	10^{-2}	3×10^{-2}	10^{-2}
InK and its ablation			
EWC (IA^3)	$(3 \times 10^{-3}, 300)$	$(3 \times 10^{-3}, 300)$	$(10^{-3}, 100)$
InK (single adapter)	$(10^{-3}, 300)$	$(3 \times 10^{-3}, 1000)$	$(10^{-3}, 100)$
+ GroupRoute (full method)	$(10^{-3}, 300)$	$(3 \times 10^{-3}, 3000)$	$(10^{-3}, 1000)$
Joint training and task-ID references			
Multi-task learning / task-ID oracle (IA^3)	3×10^{-3}	3×10^{-3}	10^{-3}
Multi-task learning / task-ID oracle (LoRA)	3×10^{-4}	3×10^{-4}	3×10^{-4}

Table 14: Selected hyperparameters for T5-large. An indented "+ GroupRoute" row refers to the method immediately above it. Entries follow the convention described in this subsection.

Method	Long sequence, O3 and O4	SuperNI-42
Sequential fine-tuning		
Seq-IA^3	3×10^{-3}	3×10^{-3}
Seq-LoRA	3×10^{-4}	3×10^{-4}
Whole-prompt routing		
EPI	10^{-2}	10^{-2}
SAPT	3×10^{-4}	3×10^{-4}
GainLoRA (InfLoRA)	3×10^{-4}	3×10^{-4}
Applying GroupRoute to existing methods		
InfLoRA	3×10^{-4}	3×10^{-4}
+ GroupRoute	3×10^{-4}	3×10^{-4}
O-LoRA	3×10^{-4}	3×10^{-4}
+ GroupRoute	3×10^{-4}	3×10^{-4}
DualPrompt	3×10^{-2}	10^{-2}
+ GroupRoute	10^{-2}	3×10^{-2}
InK and its ablation		
EWC (IA^3)	$(3 \times 10^{-3}, 1000)$	$(3 \times 10^{-3}, 300)$
InK (single adapter)	$(10^{-3}, 3000)$	$(3 \times 10^{-3}, 300)$
+ GroupRoute (full method)	$(3 \times 10^{-3}, 1000)$	$(3 \times 10^{-3}, 1000)$
Joint training and task-ID references		
Multi-task learning / task-ID oracle (IA^3)	3×10^{-3}	3×10^{-3}
Multi-task learning / task-ID oracle (LoRA)	3×10^{-4}	3×10^{-4}

D Additional Results

This section reports results that complement Section 4.2.

D.1 LLaMA-2-7B and T5-large

We report continual instruction tuning results on LLaMA-2-7B (Table 15) and T5-large (Table 16). Across both backbones, GroupRoute improves average performance and reduces forgetting for InfLoRA, O-LoRA, DualPrompt, and InK, consistent with the results on Gemma-2B-it. TRACE is not evaluated on T5-large because its tokenizer cannot represent the Chinese text required by tasks such as C-STANCE.

Table 15: Continual instruction tuning on LLaMA-2-7B. Shaded rows report results obtained with GroupRoute. Bold marks the best results among continual learning methods, including ties; joint-training and task-ID references are excluded.

	15-Task (O3)		15-Task (O4)		SuperNI-42		TRACE	
Method	**AP** ↑	F ↓	**AP** ↑	F ↓	**AP** ↑	F ↓	**AP** ↑	F ↓
Joint training and task-ID references								
Multi-task learning (IA^3)	83.32	–	83.32	–	49.63	–	55.40	–
Multi-task learning (LoRA)	84.03	–	84.03	–	51.23	–	53.29	–
Task-ID oracle (IA^3)	85.12	–	85.12	–	54.08	–	58.87	–
Task-ID oracle (LoRA)	85.04	–	85.04	–	54.92	–	59.84	–
Sequential fine-tuning								
Seq-IA^3	12.53	74.71	7.42	84.00	32.20	21.48	48.65	7.47
Seq-LoRA	12.85	75.14	6.78	84.71	30.80	23.59	29.68	31.70
Whole-prompt routing								
EPI	57.86	**0.00**	55.94	**0.00**	34.71	4.91	45.65	**0.00**
SAPT‡	81.08	4.29	78.60	7.43	40.45	14.66	56.77	1.82
GainLoRA (InfLoRA)	79.22	3.86	81.52	1.43	33.55	15.41	55.43	1.03
Applying GroupRoute to existing methods								
InfLoRA	80.34	2.29	79.48	3.86	40.13	14.71	44.94	2.31
+ GroupRoute	**85.66**	**0.00**	**85.44**	**0.00**	**54.76**	**0.45**	**58.81**	**0.00**
O-LoRA	14.63	75.29	9.45	82.00	27.21	26.78	29.06	30.54
+ GroupRoute	85.15	**0.00**	85.20	0.57	51.42	3.09	57.93	1.41
DualPrompt	67.11	6.43	68.10	9.57	33.38	12.75	46.27	7.44
+ GroupRoute	67.39	**0.00**	71.90	**0.00**	44.57	1.99	49.54	0.18
InK and its ablation								
EWC (IA^3)	48.42	39.57	71.14	15.71	28.11	25.83	52.20	3.43
InK (single adapter)	79.77	7.14	81.52	3.14	47.78	5.74	53.69	2.81
+ GroupRoute (full method)	84.34	**0.00**	85.32	0.14	53.92	0.56	58.70	0.53

‡SAPT uses pseudo-sample replay; the other continual learning methods are replay-free.

Table 16: Continual instruction tuning on T5-large. Shading, bold, and ‡ follow Table 15.

	15-Task (O3)		15-Task (O4)		SuperNI-42	
Method	**AP** ↑	F ↓	**AP** ↑	F ↓	**AP** ↑	F ↓
Joint training and task-ID references						
Multi-task learning (IA^3)	81.31	–	81.31	–	42.50	–
Multi-task learning (LoRA)	81.68	–	81.68	–	43.16	–
Task-ID oracle (IA^3)	82.24	–	82.24	–	44.07	–
Task-ID oracle (LoRA)	82.19	–	82.19	–	44.06	–
Sequential fine-tuning						
Seq-IA^3	18.76	69.71	20.14	67.00	18.78	26.52
Seq-LoRA	13.19	73.00	8.35	76.86	25.08	20.11
Whole-prompt routing						
EPI	81.54	**0.00**	81.69	**0.00**	43.09	0.05
SAPT‡	76.15	7.86	75.99	8.14	38.19	7.23
GainLoRA (InfLoRA)	78.87	1.71	79.51	3.29	33.41	5.90
Applying GroupRoute to existing methods						
InfLoRA	80.42	3.29	80.31	3.86	35.43	9.36
+ GroupRoute	81.84	0.29	82.08	**0.00**	**44.22**	0.05
O-LoRA	37.06	48.00	39.03	47.14	29.94	14.11
+ GroupRoute	**82.02**	**0.00**	81.71	0.43	42.49	1.03
DualPrompt	72.90	5.29	77.50	3.00	38.96	3.53
+ GroupRoute	80.76	**0.00**	81.47	**0.00**	43.75	0.12
InK and its ablation						
EWC (IA^3)	69.80	15.14	59.79	23.43	33.85	7.89
InK (single adapter)	79.67	2.29	75.53	6.71	39.52	3.96
+ GroupRoute (full method)	81.92	0.14	**82.09**	0.29	43.97	**0.01**

D.2 Adapter Storage

Table 17 reports adapter storage when placing IA^3 adapters on all linear layers in each Transformer block (attention: q, k, v, o; MLP: gate, up, down).

To compute the storage totals, let d denote the total number of IA^3 adapter parameters across the adapted layers. A single-task group stores one effective IA^3 adapter with d parameters. A multi-task group stores the shared log-scale vector a and the instruction coefficient vector b, with $2d$ parameters in total. SuperNI-42 contains 31 single-task groups and four multi-task groups, giving $31d + 4(2d) = 39d$. The 15-task long-sequence benchmark contains 13 single-task groups and one multi-task group, giving $13d + 1(2d) = 15d$.

Table 17: Stored adapter parameters after continual training on Gemma-2B-it. Shaded rows report configurations using GroupRoute. For IA^3-based methods, SuperNI-42 totals use 42 tasks and 35 groups when adapting all linear layers ($d = 0.97$M).

Method	Stored adapter parameters	SuperNI-42 total (approx.)
Sequential fine-tuning		
Seq-IA^3	0.97M (d)	0.97M
Seq-LoRA	0.92M	0.92M
Whole-prompt routing		
EPI	1.18M $\times T$	49.56M
SAPT	0.46M $\times T$	19.32M
GainLoRA (InfLoRA)	1.33M $\times T$	55.86M
Applying GroupRoute to existing methods		
InfLoRA	0.92M $\times T$	38.64M
+ GroupRoute	0.92M $\times T$	38.64M
O-LoRA	0.92M $\times T$	38.64M
+ GroupRoute	0.92M $\times T$	38.64M
DualPrompt	0.41M $\times T$	17.22M
+ GroupRoute	0.41M $\times T$	17.22M
InK and its ablation		
EWC (IA^3)	0.97M (d)	0.97M
InK (single adapter)	1.94M ($2d$)	1.94M
+ GroupRoute (full method)	$K_s \cdot d + (K - K_s) \cdot 2d$	37.74M
Task-ID references		
Task-ID oracle (IA^3)	0.97M $\times T$	40.74M
Task-ID oracle (LoRA)	0.92M $\times T$	38.64M

T denotes the number of tasks, K the number of groups, and K_s the number of single-task groups ($K_s = 13$ for long-sequence, $K_s = 31$ for SuperNI-42, and $K_s = 6$ for TRACE; see Table 7 in Appendix B). For GroupRoute + InK, a single-task group stores one effective IA^3 adapter ($d = 0.97$M), whereas a multi-task group stores both the shared log-scale vector a and instruction coefficient vector b ($2d = 1.94$M). On SuperNI-42 ($K_s = 31, K = 35$), the total is $31 \times 0.97\text{M} + 4 \times (2 \times 0.97\text{M}) = 39d \approx 37.74$M, which is below the $42d$ (40.74M) required by the task-ID oracle and the 38.64M required by LoRA. These totals include stored adapter parameters but exclude instruction prototypes, consolidation state, and optimizer state.

D.3 Sensitivity to Adapted Layers

Table 18 evaluates GroupRoute + InK when restricting IA^3 adapters to subsets of these layers on Gemma-2B-it across SuperNI-42 and the 15-task long-sequence benchmark.

On SuperNI-42, restricting adapters to subsets of layers achieves between 47.40 and 47.59 AP, comparable to adapting all linear layers (47.37 AP). Adapting only the five layers q, k, v, o, and down reduces stored InK parameters across the 35 groups from 37.74M to 14.74M (a 61% reduction). On the 15-task long-sequence benchmark, this same five-layer configuration achieves 80.82 AP on O3 and 80.54 AP on O4 (compared to 81.43 and 81.47 AP when adapting all layers), while reducing storage from 14.52M to 5.67M (also a 61% reduction). Thus, GroupRoute + InK does not require

modifying all linear layers; adapting a subset of layers substantially reduces stored parameters while preserving performance across both benchmarks.

Table 18: Performance and stored InK parameters under reduced adapter placement on Gemma-2B-it. The reference row adapts all linear layers in each Transformer block, while subsequent rows adapt the indicated subsets. Results are reported for SuperNI-42 and the 15-task long-sequence benchmark.

	Adapter size (d)	SuperNI-42 (35 groups)		15-Task Long-Sequence (14 groups)		
Adapted layers		**Stored InK params**	**AP**	**Stored InK params**	**O3 AP**	**O4 AP**
All linear layers	0.97M	37.74M	47.37	14.52M	**81.43**	**81.47**
`k, v, gate, down`	0.60M	23.36M	47.59	8.99M	80.92	80.45
`q, k, v, o, down`	0.38M	14.74M	47.40	5.67M	80.82	80.54
`q, k, v, o, gate`	0.38M	14.74M	47.40	5.67M	80.59	80.33

d denotes the total number of IA^3 adapter parameters across the active layers in one adapter. A single-task group stores one effective IA^3 adapter (d parameters), whereas a multi-task group stores the shared log-scale vector a and instruction coefficient vector b ($2d$ parameters). The SuperNI-42 totals use 31 single-task groups and four multi-task groups ($39d$), while the long-sequence totals use 13 single-task groups and one multi-task group ($15d$). Stored InK parameters exclude instruction prototypes, Fisher consolidation state, and optimizer state.