TRIBOS: TRAINING BEFORE SUPERVISION FOR CONTINUAL LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Applying supervised fine-tuning to sequential tasks is a standard practice in continual learning. However, direct SFT relies heavily on the independent and identically distributed assumption. Consequently, adaptation to newly encountered input distributions becomes entangled with the core objective of learning task semantics. As a result, the model may expend substantial parameter capacity on fitting superficial distributional variations, such as writing style and structural patterns, when processing continual text streams. This substantially dilutes the supervisory signal for core task semantics and leads to fragmented representations, thereby degrading continual-learning performance and exacerbating catastrophic forgetting. To address this problem, we introduce **TR**a**I**ning **B**ef**O**re **S**upervision (TRIBOS), a two-stage optimization framework that explicitly decouples distribution adaptation from semantic alignment. For each incoming task, TRIBOS first performs prompt-only self-supervised adaptation using a causal language-modeling objective to absorb distributional shifts without label leakage. Subsequently, it applies answer-only SFT to learn the target mapping. Extensive experiments on the TRACE benchmark demonstrate that TRIBOS consistently outperforms existing parameter-efficient continual learning methods. Across three backbone models, TRIBOS significantly improves final average performance by 11.06 to 17.22 percentage points and reduces forgetting by 12.90 to 17.61 percentage points compared with Sequential LoRA. Furthermore, TRIBOS effectively preserves general knowledge and reasoning capabilities on multiple mainstream datasets after long task sequences.

1 INTRODUCTION

In the field of natural language processing (NLP), developing intelligent agent systems that continuously accumulate new knowledge and flexibly adapt to dynamic environments has become a central research objective in embodied intelligence and large language models. However, continual learning in NLP remains a complex and challenging task. Unlike one-time training on a static dataset, continual learning requires models to learn incrementally from sequential tasks or non-stationary data streams(Shi et al., 2025; Sun et al., 2019). Throughout this process, the high-dimensional, sparse, and semantically rich nature of language makes models particularly susceptible to catastrophic forgetting and representation interference: acquiring knowledge from new tasks can rapidly disrupt the representations accumulated from previous tasks(Kirkpatrick et al., 2017; Sun et al., 2019).

The strong reliance of conventional supervised learning on the independent and identically distributed assumption further exacerbates this challenge. In real-world text streams, substantial domain shifts, stylistic changes, and variations in syntactic structure are inevitable(Gururangan et al., 2020). When confronted with these distributional changes, conventional supervised learning must devote considerable parameter and representational capacity to modeling domain-specific characteristics, writing styles, and complex textual structures. Consequently, supervision intended to capture the core semantic logic of a task is diluted and disrupted. Models are forced to make substantial compromises between adapting to distributional and stylistic variations and mastering task-specific semantics, reducing the effectiveness of supervised learning. This diversion of representational capacity places additional strain on an already fragile continual learning process.

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

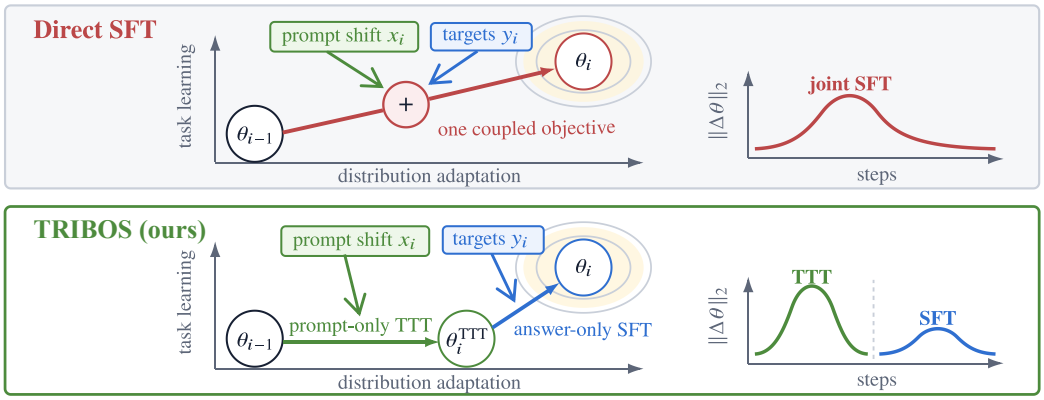

Figure 1: Motivation for TRIBOS. Top: Direct SFT couples prompt-distribution adaptation and answer-supervised learning in a single joint optimization trajectory. Bottom: TRIBOS first performs prompt-only TTT to obtain the intermediate state θ_i^{TTT} and then applies answer-only SFT, separating prompt-distribution adaptation from supervised task alignment.

Recent advances in Test-Time Training (TTT)(Sun et al., 2020), an emerging paradigm for unsupervised adaptation, offer a new perspective on this bottleneck(Hardt & Sun, 2024). By introducing lightweight self-supervised or unsupervised auxiliary objectives during inference or adaptation, such as masked language modeling, autoregressive reconstruction, or contrastive learning, TTT has demonstrated a particular strength in capturing and mitigating distribution shifts. Without requiring target-domain labels, it enables immediate adaptation at the instance or domain level to the distributional characteristics, writing styles, and structures of incoming inputs, effectively accommodating noise and stylistic drift.

Building on these insights, we argue that introducing TTT-based distribution alignment before supervised training offers a way to address representation interference in continual learning. By using TTT to adapt to distributional differences, stylistic changes, and structural variations in text without supervision, the system can fully decouple distribution adaptation from task-semantic learning. This separation frees subsequent supervised training from the burden of distribution fitting, allowing it to focus on extracting task-relevant information from supervision. Furthermore, label-free adaptation and immediate parameter updates enable TTT to improve robustness to previously unseen out-of-domain data, thereby mitigating representation conflicts at task transitions and substantially improving knowledge accumulation and resistance to forgetting.

To achieve this goal, we propose **TRa**I**n**ing **B**ef**O**re **S**upervision (TRIBOS), a decoupled, two-stage optimization framework for continual learning. As a general framework that is independent of the underlying architecture, TRIBOS inserts a prompt-only, self-supervised TTT stage before answer-supervised learning to fully decouple prompt-distribution adaptation from supervised task alignment. As illustrated in Figure 1, it explicitly restructures conventional single-stage supervised learning into two complementary stages: 1) **Prompt-Only TTT Adaptation.** Before label-driven supervised learning begins, TRIBOS performs TTT-style self-supervised adaptation on the prompt text of incoming examples using a causal language-modeling objective. Because this stage does not access answer tokens or task labels, it allows the model to absorb the vocabulary, writing style, and textual structure of the new task distribution without label leakage, smoothly removing interference caused by distribution shifts. 2) **Answer-Only Supervised Fine-Tuning.** Starting from the intermediate model state obtained through distribution adaptation, TRIBOS performs supervised fine-tuning on the same task, computing the training loss only over answer tokens. With environmental and distributional noise already absorbed in the preceding stage, the model can devote its full representational capacity to learning the mapping from prompts to answers.

To balance capacity and computational cost in parameter-efficient continual learning (PECL)(Qiao & Mahdavi, 2024), we instantiate the framework with a fixed-capacity Routed LoRA(Hu et al., 2021) module that separates stage roles. Within each adapted layer, all experts share one low-rank down-projection matrix, while a token-level top-k router selects and combines expert-specific up-projection matrices from a fixed pool(Fedus et al., 2022). Because the pool is fixed before continual training, adapter parameters do not grow with new tasks. This design enables efficient continual learning with resistance to forgetting and representational isolation.

We conduct extensive experiments on the TRACE(Wang et al., 2023b) continual learning benchmark using backbone models spanning multiple widely used architectures and scales. Our baselines include full-model sequential fine-tuning, sequential and independent LoRA, conventional regularization and replay methods, and recent state-of-the-art PECL approaches. The results demonstrate substantial advantages for TRIBOS across a range of settings. Across three backbone models, TRIBOS improves final average performance by 11.06–17.22 percentage points and reduces forgetting by 12.90–17.61 percentage points compared with Sequential LoRA. After long-sequence continual training, TRIBOS preserves general knowledge and reasoning on MMLU(Hendrycks et al., 2020), GSM8K(Cobbe et al., 2021), and BBH(Suzgun et al., 2023), optimally balancing continual learning performance and general capability retention.

Our main contributions are as follows:

- We pioneer the integration of test-time training into continual learning to explicitly decouple distribution adaptation from semantic alignment. This separation frees supervised learning from distributional and stylistic variations, allowing it to focus primarily on task-specific semantics and thereby improving continual learning.
- We propose TRIBOS, a two-stage optimization framework comprising prompt-only TTT and answer-only SFT, and instantiate it with a fixed-capacity Routed LoRA module to achieve stage-wise specialization without parameter growth.
- Extensive experiments demonstrate that TRIBOS consistently delivers substantial performance gains and marked reductions in catastrophic forgetting across multiple backbone models while effectively preserving general capabilities.

2 RELATED WORK

Training Before Supervision (TRIBOS) is inspired by two lines of research: continual learning with parameter-efficient adaptation and self-supervised adaptation before task-specific supervision.

Continual Learning and Parameter-Efficient Adaptation. Continual learning aims to learn sequential tasks while retaining prior capabilities (Parisi et al., 2019; De Lange et al., 2021; Sun et al., 2019). Existing methods mainly differ in how they preserve prior knowledge during later updates (Li & Hoiem, 2017; Zenke et al., 2017; Lopez-Paz & Ranzato, 2017; Rolnick et al., 2019). EWC (Kirkpatrick et al., 2017) regularizes parameters important to previous tasks, whereas SEEKR (He et al., 2024) combines limited replay with selective attention distillation. These regularization- and replay-based approaches mitigate forgetting by constraining updates or restoring prior information, but do not explicitly separate adaptation to input-distribution shifts from learning the corresponding supervised objective (Kirkpatrick et al., 2017; He et al., 2024).

A separate line of work addresses continual learning through parameter-efficient adaptation, restricting updates to compact modules while freezing the pretrained backbone (Razdaibiedina et al., 2023; Wang et al., 2022d;c). LoRA (Hu et al., 2021) represents task-specific updates with low-rank matrices. O-LoRA (Wang et al., 2023a) assigns tasks to approximately orthogonal subspaces, GainLoRA (Liang et al., 2026) adds gated task-specific branches, SAPT (Zhao et al., 2024) jointly learns and selects modules through shared attention, and SLoRA (Yang et al., 2026) filters low-rank directions to balance acquisition and retention. These approaches focus on controlling where or how strongly model parameters are updated, whereas TRIBOS studies a complementary question: whether the optimization transition toward each new task can be made less destructive by adapting to its input distribution before applying label-driven supervision.

Self-Supervised Adaptation Before Supervision. Self-supervised adaptation uses unlabeled target-distribution data to prepare a model for downstream prediction. Domain-adaptive pretraining and task-adaptive pretraining (Gururangan et al., 2020) continue language-model training on unlabeled domain or task data before supervised fine-tuning, demonstrating that distribution-specific adaptation can improve subsequent task learning. Test-time training (Sun et al., 2020) similarly transforms an unlabeled test sample into a self-supervised learning problem and updates the model before prediction under distribution shift. Subsequent work strengthens this paradigm through feature alignment and contrastive self-supervision (Liu et al., 2021) and masked reconstruction (Gandelsman et al., 2022), while existing test-time adaptation methods explore entropy minimization, continual adaptation, and augmentation-based objectives (Wang et al., 2020; 2022a; Zhang et al., 2022). In language modeling, TTT-NN further shows that brief language-model updates on retrieved text related to a test input can improve language-modeling performance across diverse domains (Hardt & Sun, 2024). These studies demonstrate that unlabeled target-side or target-relevant data can provide a useful adaptation signal before task-specific supervision or prediction (Sun et al., 2020; Wang et al., 2020; 2022a; Zhang et al., 2022).

TRIBOS extends these ideas to continual learning. Specifically, its self-supervised stage first adapts the model to each incoming task's input distribution, allowing subsequent supervised learning to focus on task semantics while reducing interference with prior knowledge.

3 Background and Problem Setup

3.1 Continual Learning Setting

We consider a sequence of tasks $\mathcal{T}_1, \ldots, \mathcal{T}_N$. Task $\mathcal{T}_i$ provides a dataset $\mathcal{D}_i = \{(x_i^{(n)}, y_i^{(n)})\}_{n=1}^{n_i}$, where $x_i^{(n)}$ is a prompt and $y_i^{(n)}$ is its target answer. The language-model backbone, parameterized by ϕ, remains frozen throughout the sequence. Only adapter parameters θ are updated. After learning $\mathcal{T}_i$, the model is evaluated on the current task and all previously observed tasks. The objective is to acquire each new task while retaining earlier task performance and general capabilities. We focus on parameter-efficient adaptation, where the adapter capacity is fixed before training and remains constant throughout the task sequence.

3.2 Direct Supervised Adaptation with LoRA

A conventional continual-learning baseline applies supervised fine-tuning as soon as each task arrives. For an answer $y = (y_1, \ldots, y_{|y|})$, it minimizes

$$\mathcal{L}_{\mathrm{SFT}}(\theta; \mathcal{D}_i) = -\mathbb{E}_{(x,y)\sim\mathcal{D}_i}\left[\sum_{t=1}^{|y|} \log p_{\phi,\theta}(y_t \mid x, y_{<t})\right]. \tag{1}$$

We instantiate this baseline with LoRA. Let $\mathbf{u} \in \mathbb{R}^{d_{\mathrm{in}}}$ and $\mathbf{h} \in \mathbb{R}^{d_{\mathrm{out}}}$ denote the input and output hidden representations of a linear transformation, respectively. LoRA augments a frozen weight matrix $W_0 \in \mathbb{R}^{d_{\mathrm{out}} \times d_{\mathrm{in}}}$ as

$$\mathbf{h} = W_0\mathbf{u} + \frac{\alpha}{r} BA\mathbf{u}, \tag{2}$$

where $A \in \mathbb{R}^{r \times d_{\mathrm{in}}}$ projects the input into a low-rank space and $B \in \mathbb{R}^{d_{\mathrm{out}} \times r}$ maps it back to the output space. Since r is much smaller than the input and output dimensions, only the low-rank matrices are trained while W_0 remains frozen.

Sequential LoRA reuses the same low-rank adapter throughout the task stream. Task $\mathcal{T}_i$ starts from the adapter state θ_{i-1} produced after learning $\mathcal{T}_{i-1}$ and obtains

$$\theta_i^{\mathrm{Direct}} = \mathrm{Update}(\theta_{i-1}; \mathcal{D}_i, \mathcal{L}_{\mathrm{SFT}}). \tag{3}$$

This baseline keeps the adapter capacity constant, but every task writes into the same low-rank parameters. It also requires the supervised trajectory to accommodate the new prompt distribution $P_i(x)$ while learning the task-specific mapping $P_i(y \mid x)$.

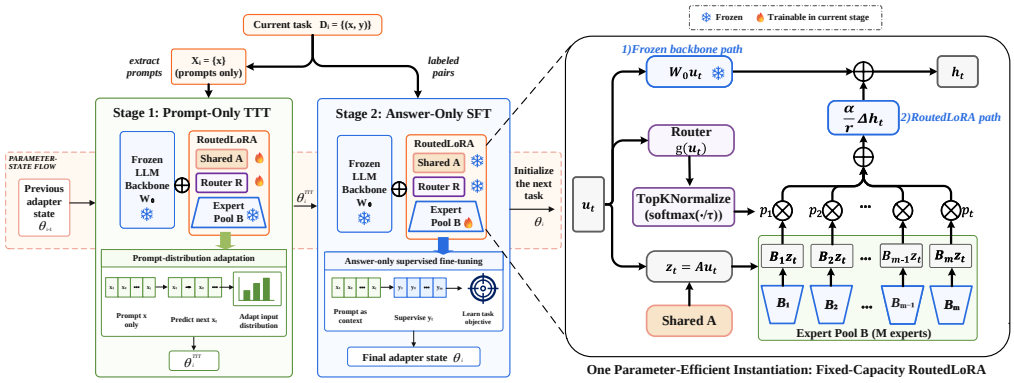

Figure 2: TRIBOS overview. Prompt-only TTT adapts the shared projection A and router R; answer-only SFT then updates the fixed expert pool $\mathcal{B}$ under token-level top-k routing.

The two forms of adaptation are therefore coupled from the first update on each task. We consider that this coupling can induce an abrupt or interfering transition at task boundaries. As a result, TRIBOS investigates whether the model can first adapt to the incoming prompt distribution before learning the corresponding supervised objective.

4 Training Before Supervision

4.1 Running Example and Method Overview

Consider MeetingBank after C-STANCE and FOMC. Its prompt contains an instruction and meeting transcript, and its answer is the summary, requiring adaptation to both a new input distribution and output objective. Direct SFT initializes from the FOMC adapter and immediately optimizes MeetingBank summaries with Eq. 1, coupling input adaptation and summary learning in the same answer-supervised updates. TRIBOS instead changes the optimization path: it first adapts to meeting-style inputs using prompts, then learns the summary mapping from labels.

For each task $\mathcal{T}_i$, TRIBOS starts from the previous adapter state θ_{i-1}. Its first stage uses only the prompts $\mathcal{X}_i = \{x : (x, y) \in \mathcal{D}_i\}$:

$$\theta_i^{\mathrm{TTT}} = \mathrm{Update}(\theta_{i-1}; \mathcal{X}_i, \mathcal{L}_{\mathrm{TTT}}) \,. \tag{4}$$

The second stage starts from this intermediate state and optimizes the supervised objective over $\mathcal{D}_i = \{(x_i^{(n)}, y_i^{(n)})\}_{n=1}^{n_i}$, the labeled prompt–answer dataset for task i:

$$\theta_i = \mathrm{Update}\big(\theta_i^{\mathrm{TTT}}; \mathcal{D}_i, \mathcal{L}_{\mathrm{SFT}}\big) \,. \tag{5}$$

Here, Update denotes optimization under the corresponding stage objective. TRIBOS specifies the objective order; the trainable parameter subsets depend on its architectural instantiation.

The two stages transition from distribution adaptation to supervised task learning. We examine whether this ordering improves new-task learning while reducing interference with prior and general capabilities. Figure 2 summarizes the workflow and fixed-capacity routed LoRA used in both stages.

4.2 Prompt-Only TTT-style Adaptation

The first stage uses no answer tokens or task labels. For a tokenized prompt $x = (x_1, \ldots, x_{|x|})$, TRIBOS minimizes the causal language-modeling objective

$$\mathcal{L}_{\mathrm{TTT}}(\theta; \mathcal{D}_i) = -\mathbb{E}_{(x,y)\sim\mathcal{D}_i} \left[\sum_{t=1}^{|x|} \log p_{\phi,\theta}(x_t \mid x_{<t}) \right] . \tag{6}$$

This objective adapts the model to the vocabulary, style, and structure of incoming prompts without optimizing target answers, providing a self-supervised signal before label-driven learning. We call it TTT-style because it follows the self-supervised adaptation principle of test-time training, but in TRIBOS it is applied at each task boundary before SFT rather than during inference.

4.3 Answer-Only Supervised Task Learning

The second stage initializes the model with θ_i^{TTT} and minimizes the answer-only objective in Eq. 1 on the same task's labeled examples. It therefore learns the conditional task objective after prompt-only adaptation. TRIBOS uses the same answer-only objective as direct SFT. Its defining difference is the optimization path: supervised learning begins from θ_i^{TTT} rather than directly from θ_{i-1}.

4.4 Parameter-Efficient Instantiation

TRIBOS is architecture-agnostic; we instantiate it with fixed-capacity routed LoRA, avoiding per-task adapters. Each adapted layer replaces the LoRA pair (A, B) (Hu et al., 2021) with a shared down-projection A and M expert-specific up-projections $B = \{B_j\}_{j=1}^{M}$ (Wang et al., 2022b; Tian et al., 2024). M is fixed before continual training and does not grow with tasks. Each layer maintains its own A and expert pool, with A shared across experts.

For token t, let $\mathbf{u}_t \in \mathbb{R}^{d_{\mathrm{in}}}$ denote the input hidden representation of an adapted linear layer. The shared projection first maps it into a low-rank space:

$$\mathbf{z}_t = A\mathbf{u}_t, \tag{7}$$

where $A \in \mathbb{R}^{r \times d_{\mathrm{in}}}$ is shared by the experts within that layer.

A lightweight router computes a sparse expert distribution independently for each token from its input hidden representation $\mathbf{u}_t$:

$$\mathbf{p}_t = \mathrm{TopKNormalize}(\mathrm{softmax}(g_R(\mathbf{u}_t)/\tau)), \tag{8}$$

where $g_R(\mathbf{u}_t) = W_R\mathbf{u}_t + \mathbf{b}_R$ is a learned linear router with $R = \{W_R, \mathbf{b}_R\}$, $W_R \in \mathbb{R}^{M \times d_{\mathrm{in}}}$, and $\mathbf{b}_R \in \mathbb{R}^M$. Here, τ is the router temperature and $\mathbf{p}_t \in \mathbb{R}^M$ the routing vector over the fixed expert pool, where $p_{t,j}$ is token t's weight for expert B_j and only the top-k entries are nonzero (Lepikhin et al., 2020). Since routing depends on each token's hidden state, no task ID is required.

Each $B_j \in \mathbb{R}^{d_{\mathrm{out}} \times r}$ is the up-projection of expert j and maps the shared low-rank representation $\mathbf{z}_t$ back to the layer's output space. Combining these expert outputs using the routing weights gives the routed LoRA update:

$$\Delta\mathbf{h}_t = \sum_{j=1}^{M} p_{t,j} B_j \mathbf{z}_t. \tag{9}$$

The adapted layer output is

$$\mathbf{h}_t = W_0\mathbf{u}_t + \frac{\alpha}{r}\Delta\mathbf{h}_t, \tag{10}$$

where M is fixed for the entire task sequence. The shared projection A, router R, and expert pool $\mathcal{B}$ are reused rather than expanded when a new task arrives.

Stage-Specific Parameter Optimization. TRIBOS strictly partitions trainable parameters between stages. For prompt-only adaptation, $\left(\eta_A^{\mathrm{TTT}}, \eta_B^{\mathrm{TTT}}, \eta_R^{\mathrm{TTT}}\right) = (\eta_{\mathrm{TTT}}, 0, \eta_{\mathrm{TTT}})$; this updates the shared projection A and router R while freezing every expert matrix $B_j \in \mathcal{B}$.

Answer-only SFT instead uses $\left(\eta_A^{\mathrm{SFT}}, \eta_B^{\mathrm{SFT}}, \eta_R^{\mathrm{SFT}}\right) = (0, \eta_{\mathrm{SFT}}, 0)$, freezing A and R and updating only the expert pool $\mathcal{B}$. This partition assigns prompt-distribution adaptation to the shared low-rank subspace and router, and answer mapping to the expert outputs. Accordingly, the generic updates in Eqs. 4 and 5 specialize to $\mathrm{Update}_{A,R}$ and $\mathrm{Update}_{\mathcal{B}}$, respectively, for the routed LoRA model used in our main experiments.

5 Experiments and Analysis

5.1 Experimental Setup

Backbone Models. We conduct the full comparison on Qwen3-8B-Base (Yang et al., 2025) and use Llama-3.1-8B Base (Grattafiori et al., 2024) and Qwen3-14B-Base (Yang et al., 2025) for cross-family and scaling validation with a focused set of representative baselines.

Benchmarks. We evaluate continual learning on TRACE (Wang et al., 2023b), an eight-task sequence comprising C-STANCE, FOMC, MeetingBank, Py150, ScienceQA, NumGLUE-cm, NumGLUE-ds, and 20Minuten. We follow the prescribed order and evaluate all tasks seen so far. To assess broader capability changes beyond TRACE, we evaluate the pretrained backbone and final checkpoint on MMLU (Hendrycks et al., 2020), GSM8K (Cobbe et al., 2021), and BBH (Suzgun et al., 2023), measuring general knowledge, mathematical problem solving, and multi-step reasoning, respectively. These benchmarks provide a complementary measure of general capability retention under continual adaptation.

Evaluation Protocol. For $T = 8$, we report final average performance $\mathrm{Final\,Avg.} = \frac{1}{T}\sum_{j=1}^{T} R_{T,j}$, learning accuracy $\mathrm{LA} = \frac{1}{T}\sum_{j=1}^{T} R_{j,j}$, backward transfer $\mathrm{BWT} = \frac{1}{T-1}\sum_{j=1}^{T-1}(R_{T,j} - R_{j,j})$, and forgetting $\mathrm{F} = \frac{1}{T-1}\sum_{j=1}^{T-1}\left(\max_{i\in\{j,\ldots,T\}} R_{i,j} - R_{T,j}\right)$. Final Avg. measures final retained performance, LA initial task acquisition, and BWT/F changes on earlier tasks. We assess general capabilities on MMLU, GSM8K, and BBH and report their average.

Baselines. We compare TRIBOS with complementary baselines: Sequential Full FT and Sequential LoRA (Hu et al., 2021) update the full model or a shared adapter; Independent LoRA uses task-specific adapters and inference-time task identity; Joint LoRA trains on pooled data as a non-continual reference. We also include EWC-LoRA (Kirkpatrick et al., 2017) (regularization), O-LoRA (Wang et al., 2023a) and SLoRA (Yang et al., 2026) (subspace control), GainLoRA (Liang et al., 2026) and SAPT-LoRA (Zhao et al., 2024) (gated/shared PEFT), and SEEKR (He et al., 2024) (replay and distillation).

Implementation Details. All main experiments use 5,000 training examples per task. TRIBOS performs one prompt-only TTT epoch per task, followed by $(5, 3, 7, 5, 3, 5, 5, 7)$ SFT epochs. Training uses FusedAdam with a base learning rate of 10^{-4}, bfloat16 precision, an effective batch size of 32, and prompt/answer lengths of 1,024/512 tokens. The primary experiments use four 48GB RTX 4090 GPUs with ZeRO-2. Additional settings are provided in Appendix A.

Table 1: Main TRACE results on Qwen3-8B-Base. TRIBOS-Std is compact ($\gamma = .25$ at inference); TRIBOS-CM is capacity-matched. TRIBOS results are mean $\pm$ standard deviation over three seeds; baselines are single runs. Best/second-best means are bold/underlined.

Method	Final Avg. ↑	LA ↑	BWT ↑	Forget. ↓
Sequential Full FT	48.47	<u>63.14</u>	-16.77	16.77
Sequential LoRA	36.29	49.14	-14.68	16.90
Independent LoRA	46.61	46.61	0.00	**0.00**
Joint LoRA	47.33	–	–	–
EWC-LoRA	51.54	51.34	<u>0.24</u>	0.61
O-LoRA	35.83	47.80	-13.68	18.99
GainLoRA	38.86	48.97	-11.55	15.02
SAPT-LoRA	44.87	44.96	-0.10	<u>0.27</u>
SEEKR	<u>53.62</u>	51.63	**2.26**	8.77
SLoRA	46.74	55.16	-9.61	14.11
TRIBOS-Std (ours)	52.50±0.43	52.40±0.57	0.12±0.66	4.00±0.61
TRIBOS-CM (ours)	**58.92**±0.51	**65.03**±0.46	-6.99±0.75	7.26±0.70

5.2 CONTINUAL-LEARNING PERFORMANCE ON TRACE

As shown in Table 1, TRIBOS-Std raises Final Avg. by 16.21 points and LA by 3.26, while cutting forgetting by 12.90 points to only 4.00. Its near-zero BWT (0.12) confirms strong retention. TRIBOS-CM delivers the strongest top-line performance, reaching 58.92 Final Avg. and 65.03 LA. It leads the respective runners-up by 5.30 and 1.89 points and outperforms Sequential LoRA by 22.62 and 15.89 points. Together, the two variants form a compelling trade-off: TRIBOS-CM maximizes acquisition and final performance, while TRIBOS-Std retains more strongly. Both use a fixed expert pool without replay or task identity at inference.

Table 2: Cross-backbone TRACE results. TRIBOS values are mean ± standard deviation over three seeds; baselines are single runs. Within each backbone, best/second-best means are bold/underlined.

Backbone	Method	Final Avg. ↑	LA ↑	BWT ↑	Forget. ↓
Llama-3.1-8B Base	Sequential Full FT	32.31	45.72	-15.32	<u>15.32</u>
	Sequential LoRA	32.11	<u>53.88</u>	-24.88	30.21
	O-LoRA	38.12	51.04	<u>-14.76</u>	20.03
	SLoRA	<u>38.48</u>	48.95	**-11.96**	**12.21**
	TRIBOS-Std (ours, $r = 8$**)**	**43.17**±0.69	**56.19**±0.76	-14.88±0.97	16.70±0.91
Qwen3-14B-Base	Sequential Full FT	**60.04**	**65.76**	<u>-6.55</u>	<u>6.91</u>
	Sequential LoRA	41.95	62.25	-23.20	23.58
	O-LoRA	43.50	59.26	-18.02	22.01
	SLoRA	53.27	63.01	-11.13	11.19
	TRIBOS-Std (ours)	<u>59.17</u>±0.34	<u>63.92</u>±0.41	**-5.43**±0.52	**5.98**±0.48

Table 2 shows that the gains transfer across families and scales. On Llama-3.1-8B, TRIBOS-Std ranks first in Final Avg. and LA, leading the runners-up by 4.69 and 2.31 points. Relative to Sequential LoRA on Llama-3.1-8B, TRIBOS-Std improves Final Avg. by 11.06 points and reduces forgetting by 13.51 points. On Qwen3-14B, TRIBOS-Std comes within 0.87 Final Avg. points of full-model fine-tuning while achieving the best BWT and forgetting. It also beats SLoRA by 5.90 Final Avg. points and reduces forgetting by 5.21 points. Across all three backbones, TRIBOS-Std improves Final Avg. by 11.06–17.22 points and cuts forgetting by 12.90–17.61 points over Sequential LoRA, demonstrating large and consistent gains across model families and scales.

Table 3: General-capability preservation after TRACE. Avg. is the mean of MMLU/GSM8K/BBH, and Δ is the change from pretrained. Bold/underline mark the best/second-best continual-learning result; pretrained is unranked.

Backbone	Method	MMLU ↑	GSM8K ↑	BBH ↑	Avg. ↑	Δ ↑
Qwen3-8B-Base	*Pretrained*	74.66	84.76	77.41	78.94	0.00
	Sequential LoRA	70.59	70.81	64.98	68.79	-10.15
	O-LoRA	72.72	<u>79.45</u>	69.50	73.89	-5.05
	SLoRA	<u>74.52</u>	77.79	<u>71.19</u>	<u>74.50</u>	<u>-4.45</u>
	Sequential Full FT	72.33	39.58	69.42	60.44	-18.50
	TRIBOS-Std (ours)	**74.63**	**86.96**	**77.15**	**79.58**	**+0.64**
Llama-3.1-8B Base	*Pretrained*	63.82	50.49	64.81	59.71	0.00
	Sequential LoRA	58.39	31.46	52.59	47.48	-12.23
	SLoRA	<u>62.68</u>	**48.67**	<u>62.74</u>	<u>58.03</u>	<u>-1.68</u>
	Sequential Full FT	46.32	3.03	36.52	28.62	-31.09
	TRIBOS-Std (ours)	**64.26**	<u>48.45</u>	**62.94**	**58.55**	**-1.16**
Qwen3-14B-Base	*Pretrained*	78.04	87.11	78.44	81.20	0.00
	Sequential LoRA	75.94	75.66	64.81	72.14	-9.06
	SLoRA	<u>78.17</u>	<u>85.37</u>	<u>78.76</u>	<u>80.77</u>	<u>-0.43</u>
	Sequential Full FT	74.63	45.11	68.28	62.68	-18.52
	TRIBOS-Std (ours)	**78.24**	**88.32**	**80.00**	**82.19**	**+0.99**

5.3 Near-Lossless Preservation of General Capabilities

Table 3 reveals a strikingly consistent preservation advantage. Excluding pretrained references, TRIBOS-Std ranks first in 14 of the 15 backbone–metric comparisons and second in the remaining case. It achieves the best average score on every backbone, exceeding SLoRA by 0.52–5.08 points, Sequential LoRA by 10.05–11.07 points, and Sequential Full FT by 19.14–29.92 points. Thus, TRIBOS improves continual learning without sacrificing broad capabilities and can even outperform its pretrained backbone after a long task sequence.

5.4 Mechanistic Analysis of Stage Specialization

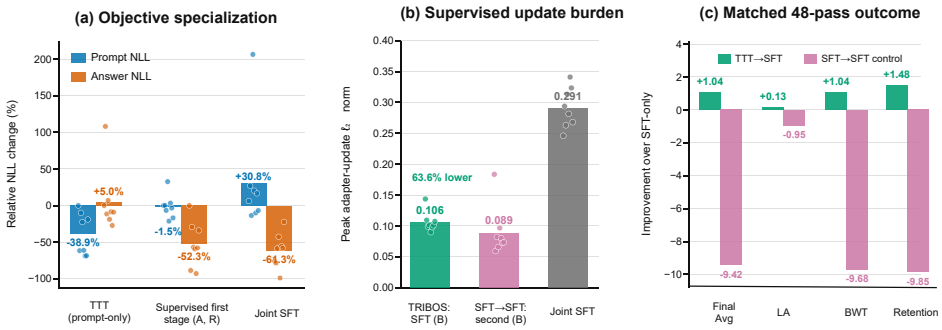

Figure 3: Stage specialization on Qwen3-8B-Base: (a) task-level prompt/answer NLL changes, (b) peak supervised update norms, and (c) gains over matched SFT-only. Retention denotes reduced forgetting (48 passes, $\gamma = 1$); Table 1 uses $\gamma = .25$.

Figure 3(a) shows objective specialization: prompt-only TTT reduces prompt NLL by 38.94% with only $+4.99\%$ answer-NLL change; the answer-supervised control reduces answer NLL by 52.30% but prompt NLL by only 1.52%; joint SFT reduces answer NLL by 61.29% while increasing prompt NLL by 30.75%. This supports separating prompt-distribution adaptation from answer alignment. In (b), peak update norms are 0.106 for TRIBOS's B-only SFT, 0.291 for joint SFT, and 0.089 for the worse-performing supervised control, ruling out update size alone. In (c), versus SFT only, TTT $\rightarrow$ SFT improves Final Avg., LA, BWT, and retention by 1.04, 0.13, 1.04, and 1.48 points, while the answer-supervised control degrades all four. Overall, the results support objective and parameter-role specialization. Appendix D reports full gradient, task-boundary NLL, and routing statistics, with all four experts active.

6 Conclusion

This paper proposes TRIBOS, a framework that leverages a prompt-only self-supervised adaptation stage prior to label-driven supervised fine-tuning. Instantiated with a fixed-capacity Routed LoRA architecture, TRIBOS separates prompt-distribution adaptation from task-specific alignment, allowing supervised learning to focus more directly on task-relevant semantic information rather than compensating for shifts in the input distribution. Evaluations on the TRACE benchmark establish that TRIBOS yields substantial gains in final task performance and marked reductions in forgetting across diverse backbone models, while concurrently preserving general reasoning capabilities. Future work could further investigate how continual learning is shaped not only by which parameters are updated, but also by the optimization path taken to acquire new tasks, including the interaction between training order, self-supervised adaptation, and parameter-efficient architectures.

AI USE STATEMENT

In this work, generative AI tools were used solely to aid and polish the writing, including improving grammar, clarity, organization, and presentation. All AI-assisted text was reviewed and verified by the authors, who take full responsibility for the final content of the paper.

REPRODUCIBILITY STATEMENT

Appendices A–E document the task order, metrics, training settings, score matrices, controlled ablations, mechanism probes, baseline implementations, and computing environment needed to reproduce our reported results. We additionally provide the implementation code in the supplementary material.

REFERENCES

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

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

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

Yossi Gandelsman, Yu Sun, Xinlei Chen, and Alexei Efros. Test-time training with masked autoencoders. *Advances in Neural Information Processing Systems*, 35:29374–29385, 2022.

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.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pp. 8342–8360, 2020.

Moritz Hardt and Yu Sun. Test-time training on nearest neighbors for large language models. In *International Conference on Learning Representations*, volume 2024, pp. 54625–54640, 2024.

Jinghan He, Haiyun Guo, Kuan Zhu, Zihan Zhao, Ming Tang, and Jinqiao Wang. Seekr: Selective attention-guided knowledge retention for continual learning of large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 3254–3266, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

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.

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.

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

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

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

Yuejiang Liu, Parth Kothari, Bastien Van Delft, Baptiste Bellot-Gurlet, Taylor Mordan, and Alexandre Alahi. Ttt++: When does self-supervised test-time training fail or thrive? *Advances in Neural Information Processing Systems*, 34:21808–21820, 2021.

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

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural networks*, 113:54–71, 2019.

Fuli Qiao and Mehrdad Mahdavi. Learn more, but bother less: parameter efficient continual learning. *Advances in Neural Information Processing Systems*, 37:97476–97498, 2024.

Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, and Amjad Almahairi. Progressive prompts: Continual learning for language models. *arXiv preprint arXiv:2301.12314*, 2023.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. *Advances in neural information processing systems*, 32, 2019.

Haizhou Shi, Zihao Xu, Hengyi Wang, Weiyi Qin, Wenyuan Wang, Yibin Wang, Zifeng Wang, Sayna Ebrahimi, and Hao Wang. Continual learning of large language models: A comprehensive survey. *ACM Computing Surveys*, 58(5):1–42, 2025.

Fan-Keng Sun, Cheng-Hao Ho, and Hung-Yi Lee. Lamol: Language modeling for lifelong language learning. *arXiv preprint arXiv:1909.03329*, 2019.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International conference on machine learning*, pp. 9229–9248. PMLR, 2020.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed H Chi, Denny Zhou, et al. 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.

Chunlin Tian, Zhan Shi, Zhijiang Guo, Li Li, and Chengzhong Xu. Hydralora: An asymmetric lora architecture for efficient fine-tuning. *Advances in Neural Information Processing Systems*, 37: 9565–9584, 2024.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726*, 2020.

Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7191–7201. IEEE, 2022a.

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, 2023a.

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

Yaqing Wang, Sahaj Agarwal, Subhabrata Mukherjee, Xiaodong Liu, Jing Gao, Ahmed Hassan, and Jianfeng Gao. Adamix: Mixture-of-adaptations for parameter-efficient model tuning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 5744–5760, 2022b.

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

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 *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 139–149. IEEE, 2022d.

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

Lina Yang, Yusheng Liao, Yanfeng Wang, and Yu Wang. Slora: Balancing plasticity and forgetting in large language models for continual learning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5437–5454, 2026.

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

Marvin Zhang, Sergey Levine, and Chelsea Finn. Memo: Test time robustness via adaptation and augmentation. *Advances in neural information processing systems*, 35:38629–38642, 2022.

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.

A Complete Experimental Setup

A.1 TRIBOS Task-Stream Procedure

Algorithm 1 summarizes the end-to-end procedure applied to the continual task stream.

Algorithm 1 Training Before Supervision over a Task Stream

Require: Frozen backbone ϕ, initial adapters θ_0, task datasets $\{\mathcal{D}_i\}_{i=1}^{N}$

for $i = 1, \ldots, N$ **do**
  $\mathcal{X}_i \leftarrow \{x : (x, y) \in \mathcal{D}_i\}$
  $\theta_i^{\mathrm{TTT}} \leftarrow \mathrm{Update}(\theta_{i-1}; \mathcal{X}_i, \mathcal{L}_{\mathrm{TTT}})$
  $\theta_i \leftarrow \mathrm{Update}(\theta_i^{\mathrm{TTT}}; \mathcal{D}_i, \mathcal{L}_{\mathrm{SFT}})$
end for
return θ_N

A.2 TRACE Benchmark

We use the balanced TRACE release Wang et al. (2023b) and keep one task order in every main comparison:

$$\text{C-STANCE} \rightarrow \text{FOMC} \rightarrow \text{MeetingBank} \rightarrow \text{Py150} \rightarrow \text{ScienceQA} \rightarrow \text{NumGLUE-cm} \rightarrow \text{NumGLUE-ds} \rightarrow \text{20Minuten}.$$

Each task provides 5,000 training examples. Evaluation and test sizes follow the source datasets and vary substantially across tasks. Table 4 reports the counts in the local `LLM-CL-Benchmark_5000` release used by all experiments.

Table 4: TRACE tasks in their continual-learning order. Scores are converted to percentages before they are combined into the continual-learning summaries.

j	Task	Task form	Domain	Language	Train	Eval	Test	Primary metric
1	C-STANCE	Stance classification	Social media	Chinese	5,000	2,000	2,000	Accuracy
2	FOMC	Policy classification	Finance	English	5,000	496	496	Accuracy
3	MeetingBank	Summarization	Meetings	English	5,000	687	692	ROUGE-L
4	Py150	Code completion	GitHub	Python	5,000	2,000	2,000	Edit similarity
5	ScienceQA	Multi-hop question answering	Science	English	5,000	2,000	2,000	Accuracy
6	NumGLUE-cm	Numerical reasoning	Mathematics	English	5,000	41	81	Accuracy
7	NumGLUE-ds	Numerical reasoning	Mathematics	English	5,000	164	325	Accuracy
8	20Minuten	Text simplification	News	German	5,000	200	200	SARI

All results in the main tables use `LLM-CL-Benchmark_5000`. The separate reasoning-augmented TRACE release is used only by our RCL reproduction runs, which are not included in the main comparison. TRIBOS and the reported baselines do not receive reasoning-augmented targets.

A.3 Evaluation Metrics and Output Parsing

Let $R_{i,j}$ be the score on task j after the model has learned tasks $1, \ldots, i$, and let $T = 8$. We compute

$$\text{FinalAvg} = \frac{1}{T} \sum_{j=1}^{T} R_{T,j}, \tag{11}$$

$$\text{LA} = \frac{1}{T} \sum_{j=1}^{T} R_{j,j}, \tag{12}$$

$$\text{BWT} = \frac{1}{T-1} \sum_{j=1}^{T-1} \left(R_{T,j} - R_{j,j} \right), \tag{13}$$

$$\text{Forgetting} = \frac{1}{T-1} \sum_{j=1}^{T-1} \left(\max_{i \in \{j,\ldots,T\}} R_{i,j} - R_{T,j} \right). \tag{14}$$

Higher Final Avg., LA, and BWT are better; lower forgetting is better. BWT compares the final score with the acquisition score, whereas forgetting compares the final score with the best score observed from task acquisition through the end of the stream. Including the final checkpoint ensures non-negative per-task forgetting.

The main task metric follows the TRACE task definition in Table 4. We do not average heterogeneous auxiliary metrics within a task. In particular, ScienceQA uses answer accuracy in the continual-learning summaries; its BLEU and ROUGE values are diagnostic only.

For C-STANCE, FOMC, NumGLUE-cm, and NumGLUE-ds, the parser removes surrounding whitespace and an optional leading colon before exact comparison. Empty strings and outputs containing extra text are incorrect. This normalization is applied uniformly to every method.

ScienceQA accepts an initial label in $\{$A,B,C,D$\}$ only when it is followed by the end of the string, whitespace, or punctuation. Thus a word such as "Dr." is not interpreted as label D. Empty reasoning strings remain valid inputs to the accuracy calculation and do not cause ROUGE failures.

General-capability evaluation uses MMLU accuracy, GSM8K flexible numeric-answer extraction, and the macro-average exact match over BBH tasks. Their reported average is the unweighted mean of the three benchmark scores.

A.4 Training Protocol

Table 5 gives the primary TRIBOS schedule. The TTT objective is causal language modeling over prompt tokens only. The SFT objective masks prompt tokens and learns from answer tokens. EOS loss is masked in TTT.

Table 5: Primary TRIBOS training protocol. Learning-rate multipliers are applied to the base learning rate of 10^{-4}. A zero multiplier freezes that role.

Stage	Objective	Data target	Epochs per task	A mult.	B mult.	Router mult.	Total epochs
TTT	Prompt causal LM	Prompt tokens	$1, 1, 1, 1, 1, 1, 1, 1$	1.0	0.0	1.0	8
SFT	Answer likelihood	Answer tokens	$5, 3, 7, 5, 3, 5, 5, 7$	0.0	1.0	0.0	40

We use DeepSpeed FusedAdam with $\beta_1 = 0.9$, $\beta_2 = 0.95$, zero weight decay, a cosine learning rate schedule, and no warmup. Training uses bfloat16, a per-device batch size of 1, four devices, and eight-step gradient accumulation, giving an effective batch size of 32.

Prompts and answers are truncated to 1,024 and 512 tokens, respectively. Phase boundaries reset gradient accumulation so that gradients from TTT and SFT are not mixed. The principal TRIBOS results in Tables 1 and 2 aggregate three random seeds. Unless stated otherwise, the remaining single-run experiments use seed 1234.

Table 6: TRIBOS adapter configurations. "All" denotes q, k, v, and o attention projections plus gate, up, and down MLP projections. All configurations use token-level expert routing.

Name	Backbone use	Targets	Rank / α	Experts	Top-k	α/r	Params	Capacity
TRIBOS-Std	All three backbones	q, v	$8/32$	4	2	4	9.437M	Fixed
TRIBOS-CM	Qwen3-8B	All	$4/8$	4	2	2	33.031M	Fixed
Enhanced	Qwen3-8B	All	$64/128$	4	2	2	–	Fixed

At evaluation, we load every round checkpoint and evaluate all tasks observed by that round. We use greedy decoding with at most 512 new tokens. The primary Qwen3-8B evaluation uses batch size 16 and single-process data loading for the finalized runs.

A.5 Model and Adapter Configuration

We study Qwen3-8B-Base, Llama-3.1-8B Base, and Qwen3-14B-Base. All are base rather than instruction-tuned checkpoints. Table 6 defines each reported TRIBOS configuration.

For an adapted linear layer, TRIBOS uses one shared input projection A and four expert output projections B_j. For token representation h, its update is

$$\Delta h = \frac{\alpha}{r} \sum_{j \in \mathrm{TopK}(h)} g_j(h) B_j A h. \tag{15}$$

The router selects two experts per token. The expert pool and router are shared across tasks, so a new task does not allocate a new adapter and task identity is not needed at inference.

TRIBOS-Std is the compact, rank-8 q/v configuration used for the main method. TRIBOS-CM redistributes a similar final budget to all attention and MLP projections. The enhanced configuration is reported only where explicitly named and is not used as TRIBOS-Std.

For Llama-3.1-8B and Qwen3-14B, the reported TRIBOS-Std checkpoints use an SFT-stage B learning rate of 5×10^{-5}. Qwen3-8B and Qwen3-14B use $\gamma = 0.25$ for both reported evaluations; Llama uses $\gamma = 1$ on TRACE and $\gamma = 0.25$ for general capabilities.

Each $\gamma = 0.25$ setting applies a post-hoc multiplier to the adapter output at inference. These values were identified through sensitivity sweeps on the reported benchmarks rather than selected on a held-out validation set.

A.6 Software and Hardware

Experiments run on Linux with Python 3.10, PyTorch 2.0.1 with CUDA 11.8, DeepSpeed 0.9.5, Transformers 4.51.3, Datasets 2.14.7, and Accelerate 0.30.1. The server has eight NVIDIA GeForce RTX 4090 GPUs with 48GB memory each.

Training jobs in the main comparison use four selected GPUs and DeepSpeed ZeRO-2. Full-model 14B training uses ZeRO-3 where stated by its launcher. Continual-learning inference and general-capability evaluation use one GPU.

B Complete Continual-Learning Results

B.1 Full Performance Matrices

Tables 20–22 provide the complete lower-triangular score matrices. Rows are checkpoints after task i and columns are evaluation tasks $j \leq i$. Entries are percentages rounded to two decimals. The matrices and task-level analyses report the seed-1234 runs; Tables 1 and 2 aggregate three random seeds for TRIBOS.

Table 7: Full TRACE matrices for TRIBOS-Std and TRIBOS-CM. C-ST denotes C-STANCE, N-cm denotes NumGLUE-cm, and N-ds denotes NumGLUE-ds.

After task i	C-ST	FOMC	Meeting	Py150	ScienceQA	N-cm	N-ds	20Min.
TRIBOS-Std ($\gamma = .25$)								
1	58.90	–	–	–	–	–	–	–
2	56.85	64.31	–	–	–	–	–	–
3	55.75	53.43	21.62	–	–	–	–	–
4	53.95	58.67	21.98	10.64	–	–	–	–
5	54.45	60.48	21.45	8.78	90.75	–	–	–
6	53.70	59.68	22.06	9.91	89.95	71.60	–	–
7	53.55	58.67	21.85	8.98	90.15	70.37	63.38	–
8	52.85	58.27	21.04	38.98	90.05	66.67	54.15	38.01
TRIBOS-CM								
1	57.85	–	–	–	–	–	–	–
2	58.00	70.16	–	–	–	–	–	–
3	58.80	69.35	41.15	–	–	–	–	–
4	58.50	69.35	39.26	60.32	–	–	–	–
5	59.75	65.32	37.25	37.25	93.00	–	–	–
6	58.45	68.15	36.72	45.46	78.05	81.48	–	–
7	57.80	67.34	34.11	47.44	80.35	72.84	75.69	–
8	57.90	61.90	31.74	49.79	87.95	69.14	72.31	40.60

Table 8: Full TRACE matrices for the RoutedLoRA SFT-only compute control and Sequential LoRA.

After task i	C-ST	FOMC	Meeting	Py150	ScienceQA	N-cm	N-ds	20Min.
RoutedLoRA SFT-only compute control								
1	53.80	–	–	–	–	–	–	–
2	52.25	66.53	–	–	–	–	–	–
3	54.05	59.48	45.38	–	–	–	–	–
4	53.45	58.27	35.43	64.42	–	–	–	–
5	55.60	12.30	33.18	62.04	91.60	–	–	–
6	53.35	56.65	29.37	59.47	50.35	71.60	–	–
7	49.75	56.85	22.32	57.63	47.25	71.60	73.23	–
8	51.00	33.67	22.93	47.78	37.35	72.84	68.31	41.11
Sequential LoRA								
1	54.90	–	–	–	–	–	–	–
2	49.60	20.16	–	–	–	–	–	–
3	37.90	34.48	42.79	–	–	–	–	–
4	39.45	6.65	12.96	63.44	–	–	–	–
5	0.05	0.00	16.55	46.49	21.10	–	–	–
6	43.00	14.52	11.01	57.31	0.20	77.78	–	–
7	41.85	21.77	6.41	58.65	7.10	79.01	72.00	–
8	43.15	0.00	9.56	54.33	0.00	74.07	68.31	40.93

Table 9: Full TRACE matrices for SLoRA, EWC-LoRA, and SEEKR.

After task i	C-ST	FOMC	Meeting	Py150	ScienceQA	N-cm	N-ds	20Min.
SLoRA								
1	58.10	–	–	–	–	–	–	–
2	57.95	23.39	–	–	–	–	–	–
3	58.70	51.01	35.81	–	–	–	–	–
4	58.25	42.34	36.50	61.58	–	–	–	–
5	58.05	33.06	36.37	59.65	87.80	–	–	–
6	58.20	14.92	36.25	59.70	83.90	69.14	–	–
7	58.05	6.85	36.58	59.96	68.90	69.14	64.31	–
8	55.40	0.81	35.57	58.01	47.75	71.60	63.69	41.13
EWC-LoRA								
1	57.15	–	–	–	–	–	–	–
2	57.10	56.45	–	–	–	–	–	–
3	57.10	56.65	22.90	–	–	–	–	–
4	56.85	54.64	22.31	48.86	–	–	–	–
5	57.00	54.84	22.44	49.14	87.10	–	–	–
6	56.95	54.23	22.28	49.48	87.05	54.32	–	–
7	56.75	55.04	21.98	49.30	87.30	59.26	46.15	–
8	56.90	55.44	22.21	49.27	87.20	58.02	45.54	37.76
SEEKR								
1	54.95	–	–	–	–	–	–	–
2	57.75	0.00	–	–	–	–	–	–
3	52.55	5.24	47.99	–	–	–	–	–
4	57.55	64.11	36.82	64.24	–	–	–	–
5	57.10	43.35	37.40	64.56	48.80	–	–	–
6	57.15	43.95	36.49	59.43	25.25	77.78	–	–
7	57.30	55.04	37.85	60.87	43.40	74.07	78.15	–
8	56.55	20.56	38.50	60.19	58.80	75.31	77.85	41.17

B.2 Per-Task Final Results

Table 23 exposes the task composition hidden by Final Avg. Inference-scaled TRIBOS-Std is strongest on ScienceQA, whereas TRIBOS-CM is substantially stronger than TRIBOS-Std on Py150.

EWC-LoRA obtains its strong retention through consistently high classification scores.

Table 10: Per-task scores after the complete eight-task stream. Bold marks the best value in each column among the listed methods.

Method	C-ST	FOMC	Meeting	Py150	ScienceQA	N-cm	N-ds	20Min.
Sequential LoRA	43.15	0.00	9.56	54.33	0.00	74.07	68.31	40.93
RoutedLoRA SFT-only	51.00	33.67	22.93	47.78	37.35	72.84	68.31	41.11
SLoRA	55.40	0.81	35.57	58.01	47.75	71.60	63.69	41.13
EWC-LoRA	56.90	55.44	22.21	49.27	87.20	58.02	45.54	37.76
SEEKR	56.55	20.56	**38.50**	**60.19**	58.80	**75.31**	**77.85**	**41.17**
TRIBOS-Std ($\gamma = .25$)	52.85	58.27	21.04	38.98	**90.05**	66.67	54.15	38.01
TRIBOS-CM	**57.90**	**61.90**	31.74	49.79	87.95	69.14	72.31	40.60

The task-level view prevents a misleading global claim. No method dominates every task. TRIBOS-Std is particularly competitive on FOMC and ScienceQA, whereas TRIBOS-CM shifts capacity toward MeetingBank, Py150, and the two NumGLUE tasks.

B.3 Task-Wise Acquisition and Retention

Table 24 compares each task's acquisition score with its final score. The last task has zero change by construction and is included only for completeness.

Inference-scaled TRIBOS-Std retains FOMC and ScienceQA strongly, but its low Py150 acquisition exposes a task-specific cost of attenuation.

TRIBOS-CM learns Py150 more effectively than TRIBOS-Std but incurs a 5.05-point ScienceQA decline. The aggregate gains therefore do not imply uniform task-level dominance.

Table 11: Acquisition, final performance, and change for the two TRIBOS variants and the Routed-LoRA SFT-only compute control. $\Delta = R_{8,j} - R_{j,j}$.

	TRIBOS-Std ($\gamma = .25$)			TRIBOS-CM			SFT-only		
Task	Acquire	Final	Δ	Acquire	Final	Δ	Acquire	Final	Δ
C-STANCE	58.90	52.85	-6.05	57.85	57.90	+0.05	53.80	51.00	-2.80
FOMC	64.31	58.27	-6.05	70.16	61.90	-8.27	66.53	33.67	-32.86
MeetingBank	21.62	21.04	-0.58	41.15	31.74	-9.40	45.38	22.93	-22.45
Py150	10.64	38.98	+28.34	60.32	49.79	-10.53	64.42	47.78	-16.64
ScienceQA	90.75	90.05	-0.70	93.00	87.95	-5.05	91.60	37.35	-54.25
NumGLUE-cm	71.60	66.67	-4.94	81.48	69.14	-12.35	71.60	72.84	+1.24
NumGLUE-ds	63.38	54.15	-9.23	75.69	72.31	-3.38	73.23	68.31	-4.92
20Minuten	38.01	38.01	0.00	40.60	40.60	0.00	41.11	41.11	0.00

C Additional Controlled Studies and Sensitivity Analysis

C.1 Stage Composition and Order

The RoutedLoRA rows in Table 25 use the same architecture and 48 nominal stage epochs. Table 12 states which objective and parameter roles are active in each condition.

Table 12: Compute-controlled stage schedules. A, B, and R denote the shared projection, expert outputs, and router.

Setting	First stage	Updated roles	Second stage	Updated roles	TTT epochs	SFT epochs
SFT only	Answer SFT	A, B, R	–	–	0	48
TTT only	Prompt TTT	A, B, R	–	–	48	0
SFT $\rightarrow$ TTT	Answer SFT	B	Prompt TTT	A, R	8	40
TTT $\rightarrow$ SFT	Prompt TTT	A, R	Answer SFT	B	8	40

TTT-only updates B because B is zero initialized. If B were frozen, the initial adapter output would be zero and useful gradients to A and the router would be blocked. This control tests the prompt-only objective, not the strict stage partition.

Under the same nominal budget, TTT $\rightarrow$ SFT exceeds SFT-only by 1.0425 Final Avg. points and 0.1309 LA points. It also improves BWT by 1.0419 and reduces forgetting by 1.4754 points.

C.2 Answer-Supervised Stage-Wise Control

The stage-wise SFT control preserves the TRIBOS parameter partition but replaces the first-stage prompt objective with answer supervision. It performs one epoch of answer SFT on A and the router, then follows the usual per-task schedule while updating only B.

This control obtains Final Avg. 37.4515, LA 62.5110, BWT -28.6394, and forgetting 29.2421. TRIBOS reaches similar acquisition but improves Final Avg. by 10.4637 and forgetting by 11.3266 points.

The result rules out parameter freezing alone as the explanation. The target used before the B-only stage matters: prompt-only TTT is materially better than writing answer-specific signals into the shared projection and router.

C.3 Compute-Matching Details

For TRIBOS, SFT $\rightarrow$ TTT, and the stage-wise control, one first-stage pass is paired with the task-specific SFT schedule $5, 3, 7, 5, 3, 5, 5, 7$. The totals are 8 first-stage and 40 second-stage epochs.

The SFT-only and TTT-only controls use $6, 4, 8, 6, 4, 6, 6, 8$, totaling 48 epochs. This matches dataset passes and nominal optimizer-step budgets. It does not match token-level FLOPs because prompt-only TTT omits answer tokens.

No replay examples are added to these controls. All four schedules use the same task order, data, seed, optimizer family, batch construction, and RoutedLoRA configuration.

C.4 TTT/SFT Epoch Allocation

Table 13 tests whether assigning more of the fixed 48-epoch training budget to TTT improves retention. The 16+32 schedule uses two TTT epochs per task and SFT epochs $4, 2, 6, 4, 2, 4, 4, 6$.

Table 13: Allocation of a fixed 48-epoch budget under strict TTT $\rightarrow$ SFT. All rows use the default inference multiplier $\gamma = 1$.

TTT + SFT	Final Avg. ↑	LA ↑	BWT ↑	F ↓
8 + 40	**47.9152**	**63.5913**	**-17.9155**	**17.9155**
16 + 32	40.1616	62.4445	-25.4661	25.6661

Increasing TTT from 8 to 16 epochs does not improve retention. It lowers Final Avg. by 7.7536, worsens BWT by 7.5506, and raises forgetting by 7.7506 points. The main 8+40 allocation is therefore used throughout the primary comparison.

C.5 First-Task Initialization Controls

With random A and zero B, strict TTT cannot update A or the router on the first task. We test three direct remedies: reversing the initialization, allowing B a 0.1 learning-rate multiplier in the first TTT stage, and inserting one B-only SFT bootstrap epoch.

Table 14: First-task cold-start controls. Tasks 2–8 retain the strict TTT $\rightarrow$ SFT rule; all rows use the default inference multiplier $\gamma = 1$.

Initialization / repair	Final Avg. ↑	LA ↑	BWT ↑	F ↓
Random A, zero B	**47.9152**	**63.5913**	**-17.9155**	**17.9155**
Zero A, random B	24.3062	43.4027	-21.8246	37.0589
First-task TTT: B multiplier 0.1	36.4068	62.6547	-29.9976	30.5405
One-epoch B-only SFT bootstrap	37.6854	63.2835	-29.2551	29.5265

None of the remedies improves the complete stream. We retain random-A/zero-B initialization and report the first TTT stage as an explicit cold-start limitation rather than hiding it.

D Additional Mechanistic Results

D.1 Stage-Specific Gradient and Update Behavior

We log optimizer statistics densely for the first 20 steps of each stage and every 50 steps thereafter. For each task and role, we take the maximum observed norm within a stage, then average these task-level peaks over the eight-task stream.

Table 15: Mean task-level peak gradient and update ℓ_2 norms. Each cell is gradient/update. A zero update with a nonzero gradient indicates a zero learning-rate multiplier.

Run	Stage	A	B	Router	All trainable state
TRIBOS	Prompt TTT	1.040 / 0.362	28.811 / 0.000	6.938 / 0.054	29.991 / 0.366
TRIBOS	Answer SFT	0.000 / 0.000	179.054 / 0.106	0.000 / 0.000	179.054 / 0.106
Stage-wise SFT	SFT on A, R	2.601 / 0.303	121.393 / 0.000	18.556 / 0.047	124.672 / 0.307
Stage-wise SFT	SFT on B	0.000 / 0.000	95.008 / 0.089	0.000 / 0.000	95.008 / 0.089
SFT only	Joint SFT	1.703 / 0.275	81.205 / 0.101	6.237 / 0.044	82.061 / 0.291

The first TRIBOS stage produces a larger aggregate update peak than its B-only SFT stage. The logs therefore do not support the claim that TRIBOS merely splits one large update into two smaller peaks. They support a role-separation claim: the two stages move different parameter groups.

On task 1, TTT has zero A and router updates because B starts at zero. The nonzero B gradient is not applied. From task 2 onward, the learned B permits prompt loss to propagate to A and the router.

D.2 Optimization Objectives and Old-Task Interference

Table 16 summarizes fixed-probe NLL changes relative to the checkpoint before each stage. Old-task change is the ratio between the post-stage and pre-stage mean old-task NLL, averaged over task boundaries 2–8.

Table 16: Mean relative NLL changes across the eight-task mechanism run. Negative current-task values indicate improved fit. Positive old-task values indicate greater local interference.

Run	Stage	Current prompt NLL	Current answer NLL	Mean old-task NLL
TRIBOS	Prompt TTT	-38.94%	$+4.99\%$	-10.89%
TRIBOS	Answer SFT	$+15.28\%$	-62.42%	$+9.41\%$
Stage-wise SFT	SFT on A, R	-1.52%	-52.30%	-14.55%
Stage-wise SFT	SFT on B	$+27.81\%$	-65.47%	$+13.74\%$
SFT only	Joint SFT	$+30.75\%$	-61.29%	$+7.00\%$

Prompt-only TTT primarily reduces prompt NLL, whereas answer-supervised adaptation of A and the router primarily reduces answer NLL. This objective selectivity is consistent with the intended division between distribution adaptation and answer alignment.

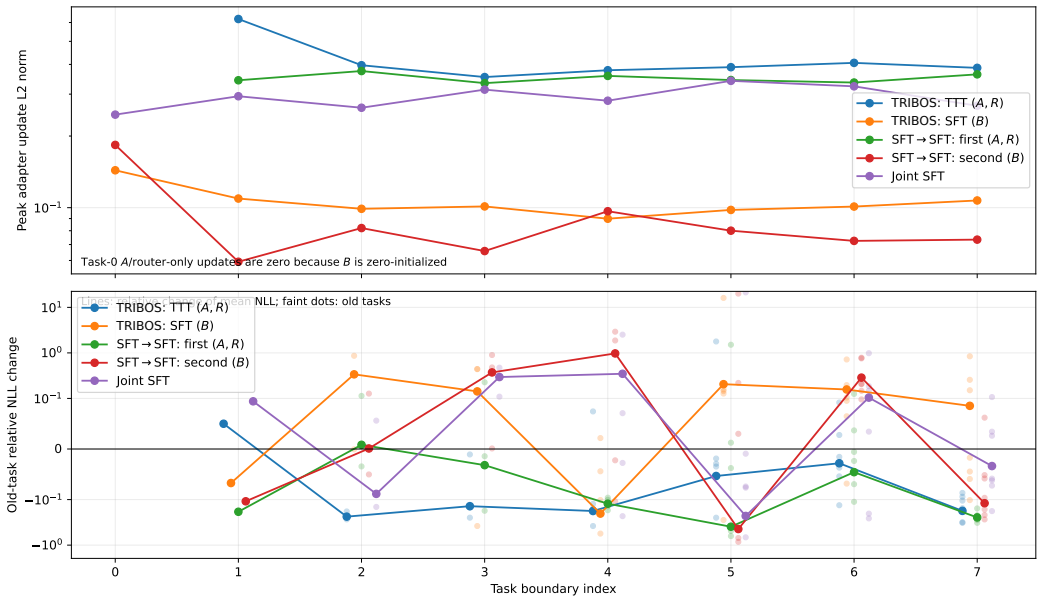

Figure 4: Detailed stage-wise optimization dynamics on the eight-task TRACE stream. The upper panel reports peak adapter-update ℓ_2 norms. The lower panel reports relative changes in mean old-task NLL from each pre-stage checkpoint; faint points denote individual old tasks. Positive values indicate greater local interference.

Figure 4 plots every task boundary. The faint task points show that old-task effects are heterogeneous; an average can hide both improvements and severe regressions. The figure is descriptive evidence and does not by itself identify a causal update direction.

The stage-wise SFT control also improves mean old-task NLL in its first stage, yet its final continual-learning metrics are much worse. Local NLL changes alone therefore do not explain long-horizon retention; the content stored in shared A and routing remains important.

D.3 Expert Utilization and Routing Dynamics

We aggregate router statistics across devices and routed layers every 50 micro-steps. The logged statistics include per-expert selection frequency, routing entropy, effective expert count, and the coefficient of variation (CV) of aggregate expert load.

Table 17: Task-wise router statistics for TRIBOS-Std on Qwen3-8B-Base. p_j is the selected fraction for expert j and sums to two because routing uses top-$k = 2$.

Task	p_0	p_1	p_2	p_3	H_{norm}	N_{eff}	CV
C-STANCE	0.620	0.551	0.427	0.403	0.499	3.938	0.178
FOMC	0.465	0.474	0.528	0.534	0.436	3.986	0.083
MeetingBank	0.437	0.519	0.498	0.546	0.471	3.984	0.088
Py150	0.482	0.544	0.457	0.517	0.466	3.985	0.086
ScienceQA	0.497	0.552	0.465	0.486	0.463	3.985	0.088
NumGLUE-cm	0.507	0.507	0.511	0.475	0.445	3.994	0.053
NumGLUE-ds	0.501	0.498	0.506	0.495	0.418	3.997	0.035
20Minuten	0.506	0.528	0.492	0.473	0.474	3.995	0.051

All four experts remain active. Effective expert count rises from 3.938 to 3.995 and load CV falls from 0.178 to 0.051, so the aggregate router does not collapse to a small subset.

Preferences still change with the task. C-STANCE favors experts 0 and 1, FOMC shifts mass toward experts 2 and 3, and later tasks become more balanced. These statistics show task-conditioned redistribution, not stable one-expert-per-task specialization.

The measurements use current-task training batches and aggregate layers and tokens. They do not track a fixed old-task probe across checkpoints, so they cannot establish cross-checkpoint routing stability or a semantic identity for each expert.

E Baseline Implementation and Fairness

E.1 Capacity, Growth, and Extra Resources

Table 18 separates final parameter capacity and task-dependent growth from additional resources. Parameter totals exclude replay examples, pseudo-data, Fisher statistics, and teacher targets; those resources are listed explicitly instead.

Table 18: Capacity and protocol comparison for Table 1. Capacity is the method-specific state retained after eight tasks. "Growth" asks whether effective task-specific state expands over the stream.

Method	Capacity / scale	Growth	Extra resource or protocol advantage
Sequential Full FT	$\sim$8B trainable	Fixed	Full-model plasticity
Sequential LoRA	3.834M adapter	Fixed	No auxiliary state
Independent LoRA	30.671M adapters	Task-growing	Task-ID inference
Joint LoRA	3.834M adapter	Fixed	Joint access to all tasks
EWC-LoRA	34.505M allocated	Task-growing	Full-space Fisher
O-LoRA	34.505M allocated	Task-growing	Orthogonal task subspaces
GainLoRA	$\sim$33.980M	Task-growing	Gated task branches
SAPT-LoRA	$\sim$16.196M	Task-growing	ARM and pseudo-data replay
SEEKR	$\sim$8B trainable	Fixed	Replay and knowledge distillation
SLoRA	$\leq$30.671M updates	Task-growing	Low-rank subspace denoising
TRIBOS-Std (ours)	9.437M adapter	Fixed	No task ID or replay
TRIBOS-CM (ours)	33.031M adapter	Fixed	Final-budget matched; no task ID or replay

TRIBOS-Std stores 9.437M adapter parameters and serves as the small common-rank comparison, while TRIBOS-CM stores 33.031M and approximately matches the final parameter budget of eight rank-8 task-growing adapters. Reporting both prevents a single capacity choice from favoring either compact fixed-adapter or task-growing methods.

Both TRIBOS variants keep a fixed four-expert pool. Adding a task neither creates an adapter nor enlarges the router, so stored capacity remains constant throughout the stream. Fixed top-k routing also keeps the number of selected expert branches per token independent of the task count. By contrast, Independent LoRA, O-LoRA, GainLoRA, SAPT-LoRA, and SLoRA retain task-dependent modules or updates. TRIBOS requires neither replay data nor task identity at inference, although these protocol differences are not included in the parameter totals.

E.2 Baseline Sources and Local Configurations

We first follow each official method's mechanism, then adapt its model and data interfaces to the same TRACE stream. Table 19 lists the settings that materially affect capacity or access to old information.

Table 19: Baseline implementations used for Qwen3-8B-Base. All methods use the task order in Appendix A.

Method	Source-aligned mechanism	Local configuration
Sequential Full FT	Sequentially update the complete backbone	Base LR 10^{-5}; full checkpoint per round
Sequential LoRA Hu et al. (2021)	Reuse one LoRA adapter through the stream	$r = 8$, $\alpha = 32$, q/v, LR 10^{-4}
Independent LoRA	Train and select one adapter per task	$r = 8$, $\alpha = 32$, q/v; task ID at inference
Joint LoRA	Train one adapter on the union of all tasks	$r = 8$, $\alpha = 32$, q/v; non-continual oracle
EWC-LoRA Kirkpatrick et al. (2017)	Fisher-weighted quadratic protection	$r = 8$, $\alpha = 32$, q/v; $\lambda = 10^7$; full Fisher
O-LoRA Wang et al. (2023a)	Task-specific orthogonal low-rank subspaces	$r = 8$, $\alpha = 32$, q/v per task
GainLoRA Liang et al. (2026)	Add a gated LoRA branch for each task	$r = 8$, $\alpha = 32$, q/v; gate hidden size 50
SAPT-LoRA Zhao et al. (2024)	Shared attentive learning/selection with ARM	$r = 4$, $\alpha = 32$, q/v; 2% pseudo inputs; KL weight 2
SEEKR He et al. (2024)	Replay with logit and selected-attention distillation	1% replay; logit weight 0.5; attention weight 1000; 24 layers/128 heads
SLoRA-Pre Yang et al. (2026)	Denoise and freeze each task's low-rank update	$r = 8$, $\alpha = 16$, q/v; official dense scoring; no replay
TRIBOS-Std	Fixed routed adapter with strict TTT $\rightarrow$ SFT	$r = 8$, $\alpha = 32$, q/v; four experts, top-$k = 2$
TRIBOS-CM	Fixed routed adapter with strict TTT $\rightarrow$ SFT	$r = 4$, $\alpha = 8$, all projections; four experts, top-$k = 2$

The GainLoRA integration isolates the published gated-branch mechanism rather than combining it with an additional InfLoRA or O-LoRA base method. SAPT includes both SALS and ARM; the ARM generator uses rank 8 and produces 2% pseudo inputs.

SEEKR updates the full model. Its replay and teacher targets are training-only and are not required at inference. SLoRA uses the online SLoRA-Pre variant; each denoised task update is materialized before the next task begins.

E.3 Protocol Differences

Independent LoRA is evaluated with task identity, and Joint LoRA sees all task data together. These are useful references but do not share the task-agnostic online protocol of TRIBOS.

EWC-LoRA retains Fisher information, SAPT and SEEKR use stored or generated examples, and SEEKR also stores teacher targets. TRIBOS, Sequential LoRA, and SLoRA do not replay old examples in the reported runs.

Task-growing methods can protect earlier updates by freezing task-specific state, but their stored capacity increases with the number of tasks. TRIBOS instead reuses a fixed expert pool, so its capacity does not grow after task 1.

E.4 Fairness Scope and Limitations

No single comparison matches parameters, trainable-state organization, replay access, full-model plasticity, and inference assumptions simultaneously. Table 1 should therefore be read together with Tables 18 and 19.

TRIBOS-Std matches the common rank-8 q/v setting but has four expert outputs. TRIBOS-CM addresses final capacity by matching task-growing PEFT budgets. Neither setting makes the optimization geometry identical to methods that freeze one adapter per task.

The principal TRIBOS results use one task order and three random seeds. The baseline, controlled-ablation, and mechanism-study results use one seed, with fixed probe batches for the mechanism study. The main tables therefore quantify random-seed variance for TRIBOS, but not variance across task orders or for the single-run comparisons.

The $\gamma = 0.25$ TRIBOS rows were chosen after inspecting inference-scale sweeps on the reported benchmarks. They are post-hoc sensitivity results, not values selected on a held-out validation set, and should be interpreted accordingly.

F Complete Continual-Learning Results

F.1 Full Performance Matrices

Tables 20–22 provide the complete lower-triangular score matrices. Rows are checkpoints after task i and columns are evaluation tasks $j \leq i$. Entries are percentages rounded to two decimals; all summary metrics use the unrounded JSON values.

Table 20: Full TRACE matrices for TRIBOS-Std and TRIBOS-CM. C-ST denotes C-STANCE, N-cm denotes NumGLUE-cm, and N-ds denotes NumGLUE-ds.

After task i	C-ST	FOMC	Meeting	Py150	ScienceQA	N-cm	N-ds	20Min.
TRIBOS-Std								
1	55.40	–	–	–	–	–	–	–
2	52.40	64.92	–	–	–	–	–	–
3	52.45	63.10	42.74	–	–	–	–	–
4	47.70	58.06	33.69	63.51	–	–	–	–
5	50.15	60.28	29.69	61.73	91.35	–	–	–
6	47.65	54.03	29.93	60.92	59.60	76.54	–	–
7	47.80	47.58	24.38	56.50	56.10	72.84	73.54	–
8	39.60	25.20	21.75	49.41	66.10	71.60	68.92	40.73
TRIBOS-CM								
1	54.65	–	–	–	–	–	–	–
2	54.60	69.15	–	–	–	–	–	–
3	55.70	68.75	46.68	–	–	–	–	–
4	55.10	69.56	44.58	63.91	–	–	–	–
5	55.75	0.00	40.75	64.69	90.80	–	–	–
6	55.85	69.76	39.03	63.05	58.15	76.54	–	–
7	52.15	70.36	35.62	61.53	57.00	74.07	75.69	–
8	53.50	55.65	33.40	61.51	17.40	74.07	73.54	41.77

Table 21: Full TRACE matrices for the RoutedLoRA SFT-only compute control and Sequential LoRA.

After task i	C-ST	FOMC	Meeting	Py150	ScienceQA	N-cm	N-ds	20Min.
RoutedLoRA SFT-only compute control								
1	53.80	–	–	–	–	–	–	–
2	52.25	66.53	–	–	–	–	–	–
3	54.05	59.48	45.38	–	–	–	–	–
4	53.45	58.27	35.43	64.42	–	–	–	–
5	55.60	12.30	33.18	62.04	91.60	–	–	–
6	53.35	56.65	29.37	59.47	50.35	71.60	–	–
7	49.75	56.85	22.32	57.63	47.25	71.60	73.23	–
8	51.00	33.67	22.93	47.78	37.35	72.84	68.31	41.11
Sequential LoRA								
1	54.90	–	–	–	–	–	–	–
2	49.60	20.16	–	–	–	–	–	–
3	37.90	34.48	42.79	–	–	–	–	–
4	39.45	6.65	12.96	63.44	–	–	–	–
5	0.05	0.00	16.55	46.49	21.10	–	–	–
6	43.00	14.52	11.01	57.31	0.20	77.78	–	–
7	41.85	21.77	6.41	58.65	7.10	79.01	72.00	–
8	43.15	0.00	9.56	54.33	0.00	74.07	68.31	40.93

Table 22: Full TRACE matrices for SLoRA, EWC-LoRA, and SEEKR.

After task i	C-ST	FOMC	Meeting	Py150	ScienceQA	N-cm	N-ds	20Min.
SLoRA								
1	58.10	–	–	–	–	–	–	–
2	57.95	23.39	–	–	–	–	–	–
3	58.70	51.01	35.81	–	–	–	–	–
4	58.25	42.34	36.50	61.58	–	–	–	–
5	58.05	33.06	36.37	59.65	87.80	–	–	–
6	58.20	14.92	36.25	59.70	83.90	69.14	–	–
7	58.05	6.85	36.58	59.96	68.90	69.14	64.31	–
8	55.40	0.81	35.57	58.01	47.75	71.60	63.69	41.13
EWC-LoRA								
1	57.15	–	–	–	–	–	–	–
2	57.10	56.45	–	–	–	–	–	–
3	57.10	56.65	22.90	–	–	–	–	–
4	56.85	54.64	22.31	48.86	–	–	–	–
5	57.00	54.84	22.44	49.14	87.10	–	–	–
6	56.95	54.23	22.28	49.48	87.05	54.32	–	–
7	56.75	55.04	21.98	49.30	87.30	59.26	46.15	–
8	56.90	55.44	22.21	49.27	87.20	58.02	45.54	37.76
SEEKR								
1	54.95	–	–	–	–	–	–	–
2	57.75	0.00	–	–	–	–	–	–
3	52.55	5.24	47.99	–	–	–	–	–
4	57.55	64.11	36.82	64.24	–	–	–	–
5	57.10	43.35	37.40	64.56	48.80	–	–	–
6	57.15	43.95	36.49	59.43	25.25	77.78	–	–
7	57.30	55.04	37.85	60.87	43.40	74.07	78.15	–
8	56.55	20.56	38.50	60.19	58.80	75.31	77.85	41.17

F.2 Per-Task Final Results

Table 23 exposes the task composition hidden by Final Avg. TRIBOS-CM improves the final C-STANCE and FOMC scores over TRIBOS-Std, but its final ScienceQA score is substantially lower. EWC-LoRA obtains its strong retention through consistently high classification scores.

The task-level view prevents a misleading global claim. No method dominates every task. TRIBOS-Std is particularly competitive on ScienceQA, whereas TRIBOS-CM shifts capacity toward C-STANCE, FOMC, MeetingBank, Py150, and the two NumGLUE tasks.

F.3 Task-Wise Acquisition and Retention

Table 24 compares each task's acquisition score with its final score. The last task has zero change by construction and is included only for completeness.

TRIBOS-Std improves final ScienceQA by 28.75 points over the SFT-only compute control, despite similar acquisition scores. Its largest losses instead occur on FOMC and MeetingBank. TRIBOS-CM reduces losses on most tasks but incurs a 73.40-point ScienceQA decline.

Table 23: Per-task scores after the complete eight-task stream. Bold marks the best value in each column among the listed methods.

Method	C-ST	FOMC	Meeting	Py150	ScienceQA	N-cm	N-ds	20Min.
Sequential LoRA	43.15	0.00	9.56	54.33	0.00	74.07	68.31	40.93
RoutedLoRA SFT-only	51.00	33.67	22.93	47.78	37.35	72.84	68.31	41.11
SLoRA	55.40	0.81	35.57	58.01	47.75	71.60	63.69	41.13
EWC-LoRA	**56.90**	55.44	22.21	49.27	**87.20**	58.02	45.54	37.76
SEEKR	56.55	20.56	**38.50**	60.19	58.80	**75.31**	**77.85**	41.17
TRIBOS-Std	39.60	25.20	21.75	49.41	66.10	71.60	68.92	40.73
TRIBOS-CM	53.50	**55.65**	33.40	**61.51**	17.40	74.07	73.54	**41.77**

Table 24: Acquisition, final performance, and change for the two TRIBOS variants and the Routed-LoRA SFT-only compute control. $\Delta = R_{8,j} - R_{j,j}$.

	TRIBOS-Std			TRIBOS-CM			SFT-only		
Task	Acquire	Final	Δ	Acquire	Final	Δ	Acquire	Final	Δ
C-STANCE	55.40	39.60	-15.80	54.65	53.50	-1.15	53.80	51.00	-2.80
FOMC	64.92	25.20	-39.72	69.15	55.65	-13.50	66.53	33.67	-32.86
MeetingBank	42.74	21.75	-20.99	46.68	33.40	-13.28	45.38	22.93	-22.45
Py150	63.51	49.41	-14.10	63.91	61.51	-2.40	64.42	47.78	-16.64
ScienceQA	91.35	66.10	-25.25	90.80	17.40	-73.40	91.60	37.35	-54.25
NumGLUE-cm	76.54	71.60	-4.94	76.54	74.07	-2.47	71.60	72.84	+1.24
NumGLUE-ds	73.54	68.92	-4.62	75.69	73.54	-2.15	73.23	68.31	-4.92
20Minuten	40.73	40.73	0.00	41.77	41.77	0.00	41.11	41.11	0.00

G Additional Ablation Studies

G.1 Controlled Ablations of the Two-Stage Training Design

Panel A of Table 25 fixes the architecture and a 48-pass budget. At similar LA, TTT→SFT raises Final Avg. from 46.87 to 47.92 and lowers forgetting from 19.39 to 17.92. Reversing the order gives 32.90 Final Avg. and 24.71 forgetting, while TTT alone reaches only 17.83 LA. An answer-supervised first stage retains similar LA (62.51 vs. 63.59) but gives 37.45 Final Avg. and 29.24 forgetting. Thus, the gain comes from prompt adaptation rather than extra passes, freezing, or staging (Section D).

G.2 Architecture Generality and Capacity-Controlled Comparison

Panel B of Table 25 tests whether adaptation before supervision transfers beyond RoutedLoRA.

With vanilla LoRA, TTT→SFT raises LA from 53.75 to 62.22, showing that the training principle improves initial task acquisition across adapter architectures. Final Avg. remains nearly unchanged (34.44 vs. 34.37), while forgetting increases from 23.91 to 32.26. Training order alone therefore improves plasticity but does not preserve old tasks throughout the stream. Table 26 compares TRIBOS-Std with rank-20 vanilla LoRA under nearly equal parameter budgets (9.437M vs. 9.585M). TRIBOS raises Final Avg. from 34.11 to 47.92 and LA from 48.20 to 63.59, while reducing forgetting from 24.23 to 17.92.

Table 25: Compute-controlled TRACE results (Qwen3-8B-Base; 48 passes, $\gamma = 1$). A/B: Routed-LoRA/vanilla LoRA. †TTT-only retention accompanies low LA.

Setting	TTT	SFT	Final Avg. ↑	LA ↑	BWT ↑	F ↓
Panel A: RoutedLoRA controls						
SFT only	0	48	46.87	63.46	-18.96	19.39
SFT$(A, R)\rightarrow$SFT(B)	0	48	37.45	62.51	-28.64	29.24
TTT only	48	0	21.96	17.83	+4.73†	6.17†
SFT→TTT	8	40	32.90	54.16	-24.29	24.71
TTT→SFT	**8**	**40**	**47.92**	**63.59**	**-17.92**	**17.92**
Panel B: Vanilla LoRA						
SFT only	0	48	34.37	53.75	-22.15	23.91
TTT only	48	0	17.79	21.71	-4.48†	5.35†
SFT→TTT	8	40	31.24	50.49	-21.99	24.12
TTT→SFT	8	40	34.44	62.22	-31.75	32.26

Table 26: Capacity-matched comparison on Qwen3-8B-Base.

Architecture	Params	Final Avg. ↑	LA ↑	BWT ↑	F ↓
Vanilla LoRA ($r = 20$)	9.585M	34.11	48.20	**-16.10**	24.23
TRIBOS-Std ($r = 8$)	9.437M	**47.92**	**63.59**	-17.92	**17.92**