LEARNING AN ANCHORED PROMPT SPACE FOR CONTINUAL ADAPTATION OF LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Continually adapting large language models requires acquiring new knowledge while preserving previously learned capabilities. Jointly adapting model parameters and task-specific soft prompts offers a promising solution, but faces two key limitations: historical prompts may become less effective as the model evolves, while their transferable cross-task relationships are not explicitly learned. We propose Learning an Anchored Prompt Space (LAPS), which preserves historical prompt effectiveness and learns relationships among task-specific soft prompts to facilitate positive transfer. LAPS first aligns historical prompts with the updated model through self-distillation. LAPS then constructs an anchored prompt space whose vertices correspond to learned task-specific soft prompts and whose intermediate geometry is shaped by learnable Bézier control prompts. Once this anchored prompt space is learned, LAPS identifies the best-performing prompt for each task on its validation set, allowing the optimized prompt to draw on knowledge acquired from observed tasks. Experiments on the TRACE benchmark across three Qwen3 model scales show that LAPS consistently outperforms distillation-based, prompt-based, and joint prompt–parameter adaptation baselines, improving average performance while reducing forgetting. Our implementation is available at https://anonymous.4open.science/r/LAPS_.

1 INTRODUCTION

Biological learning achieves efficient adaptation by combining plastic mechanisms that incorporate new evidence with stabilizing mechanisms that protect useful prior structure, rather than rewriting an entire system for each new experience (Grossberg, 1980; McCloskey & Cohen, 1989; French, 1999). Complementary learning systems and the stability–plasticity trade-off provide classic accounts of how learning can balance acquisition and retention (McClelland et al., 1995; Kirkpatrick et al., 2017). A similar pressure arises in large language model (LLM) deployment: after pretraining and instruction tuning, LLMs must adapt to new tasks, domains, policies, and user requirements without full retraining (Wang et al., 2023a; Ibrahim et al., 2024; Yang et al., 2025b). Practical adaptation must incorporate task-specific evidence from limited data while retaining capabilities acquired from previous tasks, making continual learning central to LLM adaptation.

Recent studies on continual adaptation of LLMs have explored two complementary mechanisms: model parameter updates (Peng et al., 2024; Nayak et al., 2026) and task-specific soft prompts (Razdaibiedina et al., 2023; Qiu et al., 2025). Updating model parameters provides substantial capacity for acquiring task knowledge, but it can also interfere with prior knowledge. In contrast, prompt-based methods preserve task-specific behaviors without modifying the underlying model, although this limits adaptation capacity. To balance these strengths, later methods jointly adapt model parameters and prompts (Wang et al., 2024; Tiwari et al., 2026). Yet joint adaptation introduces an additional source of instability: historical prompts were optimized for earlier model parameters, and as model parameters evolve, these prompts may no longer induce their original behaviors.

In addition to maintaining prior task behaviors, continual adaptation also creates opportunities for positive transfer across related tasks. If knowledge acquired from previous tasks can be organized and reused, later tasks may benefit from earlier adaptations rather than treating each task in isolation. Existing approaches have explored prompt retrieval, sharing, and composition to reuse knowledge across tasks (Wang et al., 2022b; Smith et al., 2023; Gao et al., 2024; Han et al., 2025; Hong et al.,

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

2025). However, as the model evolves, these methods do not explicitly organize historical task-specific prompts into a learnable structure for discovering transferable cross-task relationships.

To address these limitations, we propose Learning an Anchored Prompt Space (LAPS), a continual adaptation method that preserves historical task adaptations under evolving model parameters while identifying prompts that exploit positive transfer across tasks. LAPS contains two key elements. First, LAPS aligns historical prompts with the evolving LLM through on-policy self-distillation (Shenfeld et al., 2026a; Zhao et al., 2026). When model parameters are updated for a new task, historical prompts are adjusted so that they remain close to their original task behaviors under the updated model. By restoring prior behaviors through the prompts rather than constraining the model update, this alignment leaves greater flexibility for new-task adaptation. Second, LAPS organizes task-specific knowledge encoded by soft prompts into an anchored prompt space. Learned task-specific soft prompts serve as vertex anchors for each observed task, and a degree-k Bézier simplex introduces learnable control prompts that shape the geometry between these anchors. Rather than relying on direct prompt combination, LAPS learns this anchored space from cross-task interactions and then searches it to identify high-performing prompts that exploit positive transfer.

Our contributions are summarized as follows:

- We propose a continual adaptation method called LAPS that introduces historical prompt alignment and an anchored prompt space. The alignment preserves historical task behaviors under continued model updates without constraining backbone adaptation to new tasks
- We introduce a two-stage optimization strategy for the anchored prompt space parameterized by a degree-k Bézier simplex: learning control prompts from cross-task interactions and selecting task-specific simplex coordinates by validation performance, enabling the discovery of positive transfer across observed tasks.
- We evaluate LAPS on the TRACE benchmark (Wang et al., 2023b) across Qwen3 models from 1.7B to 8B. LAPS consistently achieves a favorable adaptation–retention trade-off against distillation-based, prompt-based, and joint prompt–parameter adaptation methods.

2 Related Work

In this section, we review continual learning methods most closely related to LAPS: replay, knowledge distillation, and prompt-based adaptation. We focus on where reusable knowledge is stored, how prior behavior is maintained, and whether positive transfer is supported.

Replay-based Continual Learning. Replay-based methods mitigate catastrophic forgetting by retaining and revisiting previous data when learning new tasks. Experience Replay (Isele & Cosgun, 2018; Rolnick et al., 2019; Fedus et al., 2020) shows that even a small memory of historical samples can help preserve previous knowledge. Subsequent work improves replay by storing additional training signals, such as historical logits (Buzzega et al., 2020), or by learning generative models that reproduce previous task knowledge (Gao & Liu, 2023; Maekawa et al., 2023). For LLM adaptation, OPR (Chen et al., 2026) further studies on-policy replay data for sequential updates. These methods rely on stored or regenerated historical information, while directly mixing replay data into model updates may interfere with current task adaptation. LAPS instead uses limited replay data to supervise historical prompt alignment and anchored prompt space learning, supporting retention and cross-task transfer while avoiding direct interference with new task model updates.

Knowledge Distillation-based Continual Learning. Knowledge distillation mitigates forgetting by encouraging the updated model to match selected behaviors of an earlier model. LwF (Li & Hoiem, 2018) matches output distributions, while later continual learning methods distill soft predictions or intermediate representations (Buzzega et al., 2020; Douillard et al., 2020). Recent work further studies self-distillation for continual learning and restoration-style distillation for LLM behavior preservation (Dong et al., 2025; Shenfeld et al., 2026a). However, behavioral constraints that preserve earlier knowledge can also limit the acquisition of new task knowledge. LAPS instead applies self-distillation to historical prompts rather than constraining backbone updates. This leaves greater flexibility for new-task adaptation, while historical behaviors affected by model parameter updates are recovered through the aligned prompts.

Prompt-based Continual Learning. Prompt-based continual learning stores task knowledge in learnable prompts. One common setting introduces task-specific prompts while typically keeping

the pretrained backbone frozen. L2P (Wang et al., 2022b) learns a prompt pool for task-aware retrieval, enabling prompts to be reused across observed tasks. DualPrompt (Wang et al., 2022a) separates shared and task-specific prompts to capture common and task-dependent information. ProgPrompt (Razdaibiedina et al., 2023) learns a new prompt for each task and concatenates it with previous prompts, allowing later tasks to reuse earlier prompt knowledge. CODA-Prompt (Smith et al., 2023) decomposes prompts into reusable factors and learns attention-based combinations of these factors, providing a more flexible form of prompt composition. This prompt-only line of work reduces forgetting by storing task knowledge in learnable prompts, but keeping the backbone fixed limits the capacity for new-task adaptation through model parameter updates.

Another line of work combines prompt learning with model parameter updates. ProMoT (Wang et al., 2024) first learns task-specific prompts and then updates model parameters. Subsequent methods combine meta prompting with low-rank parameter updates (Yu et al., 2025) or use fast–slow adaptation mechanisms (Tiwari et al., 2026). Jointly adapting prompts and model parameters is difficult because historical prompts are learned under earlier model states and may become less compatible as model parameters evolve. Existing prompt pools, prompt composition methods, and meta prompting approaches do not explicitly align historical prompts under changing model parameters or learn a controlled prompt space for positive transfer. LAPS addresses this setting by aligning historical prompts under the current model, learning Bézier control prompts to shape an anchored prompt space, and selecting task-specific simplex coordinates according to validation performance.

3 Learning an Anchored Prompt Space for Continual Adaptation

In this section, we first formulate continual adaptation with joint prompt–parameter updates and identify the two difficulties that motivate LAPS. We then describe how LAPS aligns historical soft prompts, learns an anchored prompt space, and selects the final soft prompt for each observed task.

3.1 Problem Setup and Overview

We study continual adaptation over a sequence of tasks $\mathcal{T}_{1:T} = (\mathcal{T}_1, \ldots, \mathcal{T}_T)$. Each task $\mathcal{T}_t$ is specified by a training set $\mathcal{D}_t = \{(\boldsymbol{x}_t^{(i)}, y_t^{(i)})\}_{i=1}^{N_t}$, where $\boldsymbol{x}_t^{(i)}$ and $y_t^{(i)}$ denote the input and target output of the i-th example from task t. The tasks are learned sequentially, with the full training data of previous tasks unavailable when adapting to $\mathcal{T}_t$. After each learning stage, the model is evaluated on all observed tasks. We follow the task-incremental setting (Van de Ven et al., 2022), where the identity of the current task is known at inference time and can therefore be used to select the corresponding task-specific prompt.

Following recent studies (Wang et al., 2024; Tiwari et al., 2026), we consider two coupled adaptation components at stage t: the model parameters $\boldsymbol{\theta}_t$ and a task-specific soft prompt. We use $\mathbf{P}_{j,t}$ to denote the soft prompt for task $\mathcal{T}_j$ after adaptation stage t, with $\mathbf{P}_{t,t}$ denoting the prompt newly learned for the current task $\mathcal{T}_t$. In the standard joint adaptation setting, each task-specific soft prompt is frozen after it is learned, *i.e.*, $\mathbf{P}_{j,t} = \mathbf{P}_{j,j}$ for all $t \geq j$. A basic joint adaptation objective therefore optimizes the model parameters $\boldsymbol{\theta}_t$ and the current task prompt $\mathbf{P}_{t,t}$ on $\mathcal{D}_t$:

$$\boldsymbol{\theta}_t, \mathbf{P}_{t,t} = \arg\min_{\boldsymbol{\theta}, \mathbf{P}} \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}_t} \left[\ell_t \left(f(\boldsymbol{x}; \boldsymbol{\theta}, \mathbf{P}), y\right)\right], \quad t = 1, 2, ..., T. \tag{1}$$

where $f(\boldsymbol{x}; \boldsymbol{\theta}, \mathbf{P})$ denotes the model prediction for input $\boldsymbol{x}$ under parameters $\boldsymbol{\theta}$ and soft prompt $\mathbf{P}$, and ℓ_t denotes the task-specific supervised loss. LAPS instantiates this adaptation step through the alternating optimization strategy of ProMoT (Wang et al., 2024). For task $\mathcal{T}_t$, LAPS first learns the current task prompt $\mathbf{P}_{t,t}$ while keeping the previous model parameters $\boldsymbol{\theta}_{t-1}$ fixed:

$$\mathbf{P}_{t,t} = \arg\min_{\mathbf{P}} \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}_t} \left[\ell_t \left(f(\boldsymbol{x}; \boldsymbol{\theta}_{t-1}, \mathbf{P}), y\right)\right]. \tag{2}$$

After solving Eq. (2), LAPS fixes $\mathbf{P}_{t,t}$ and updates the model parameters on the current task:

$$\boldsymbol{\theta}_t = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}_t} \left[\ell_t \left(f(\boldsymbol{x}; \boldsymbol{\theta}, \mathbf{P}_{t,t}), y\right)\right]. \tag{3}$$

After this alternating update, LAPS has the current prompt and model state. However, the adaptation process still faces two challenges. First, updating model parameters from $\boldsymbol{\theta}_{t-1}$ to $\boldsymbol{\theta}_t$ for task t can cause model drift, making prompts learned for earlier tasks, $\{\mathbf{P}_{j,t-1}\}_{j<t}$, less compatible with the updated model. Second, although task-specific soft prompts encode useful task knowledge, their

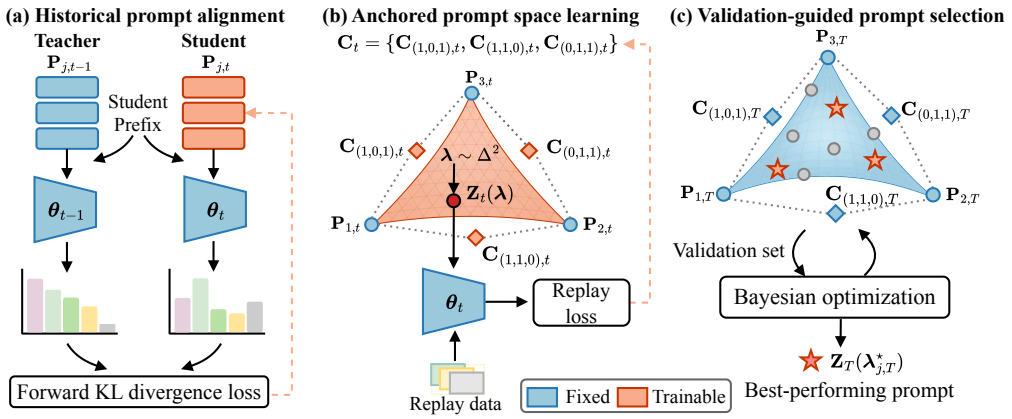


Figure 1: **Overview of LAPS. (a)** After adapting to the current task $\mathcal{T}_t$, LAPS aligns each historical prompt $\mathbf{P}_{j,t-1}$ to the updated model, producing $\mathbf{P}_{j,t}$ through on-policy self-distillation. **(b)** The aligned prompts $\{\mathbf{P}_{j,t}\}_{j\le t}$ are fixed as simplex anchors, while non-vertex Bézier control prompts $\mathbf{C}_{\boldsymbol{\alpha},t}$ are trained with cross-task replay losses to shape the space between anchors. **(c)** With the final model fixed, Bayesian optimization searches validation scores over simplex coordinates $\boldsymbol{\lambda}$ and returns the selected soft prompt $\mathbf{Z}_T(\boldsymbol{\lambda}^\star_{j,T})$ for each observed task.

transferable relationships are not explicitly organized as the model evolves. LAPS addresses these challenges through the three components illustrated in Fig. 1: historical prompt alignment, anchored prompt space learning, and validation-guided prompt selection.

3.2 Historical Prompt Alignment via Self-Distillation

As model parameters evolve, a prompt learned at an earlier stage may no longer induce the same behavior under the updated model. LAPS aligns historical prompts after each model update so that they remain compatible with the model, without constraining the model update in Eq. (3).

Direct supervised fine-tuning of historical prompts on limited previous task replay data is insufficient because the replay set sparsely covers previous tasks, and ground-truth targets do not capture the full behavior of the original model–prompt pair, such as output distributions or responses to alternative continuations. LAPS therefore uses the previous model–prompt pair as the teacher, providing a richer behavioral target for aligning each historical prompt with the updated model.

Let $\mathcal{S}_t \subset \mathcal{D}_t$ be a subset sampled from the current task. Prior work shows that the forward KL divergence from the original model to the updated model, evaluated on new task data, strongly predicts forgetting of previously learned capabilities (Shenfeld et al., 2026b). Motivated by this finding, we include $\mathcal{S}_t$ alongside the stage-t replay set $\mathcal{R}_t$ when aligning historical prompts. For each task $j = 2, \ldots, t$, the teacher is the previous pair $(\boldsymbol{\theta}_{t-1}, \mathbf{P}_{j,t-1})$, and the student is the pair $(\boldsymbol{\theta}_t, \mathbf{P}_{j,t})$. The teacher provides a behavioral reference for how the historical prompt functioned before the model update. Matching its predictive distribution allows the updated prompt to recover this behavior while keeping $\boldsymbol{\theta}_t$ fixed. LAPS aligns $\mathbf{P}_{j,t}$ by minimizing the forward KL divergence:

$$\min_{\mathbf{P}_{j,t}} \mathbb{E}_{\substack{\boldsymbol{x}\sim\mathcal{R}_t\cup\mathcal{S}_t \\ \boldsymbol{y}_{<\tau}\sim\pi_{\boldsymbol{\theta}_t,\mathbf{P}_{j,t}}(\cdot|\boldsymbol{x})}} \left[D_{\mathrm{KL}}\left(\pi_{\boldsymbol{\theta}_{t-1},\mathbf{P}_{j,t-1}}(\cdot \mid \boldsymbol{x}, \boldsymbol{y}_{<\tau}) \| \pi_{\boldsymbol{\theta}_t,\mathbf{P}_{j,t}}(\cdot \mid \boldsymbol{x}, \boldsymbol{y}_{<\tau}) \right) \right], \tag{4}$$

where $\pi_{\boldsymbol{\theta},\mathbf{P}}(\cdot \mid \boldsymbol{x}, \boldsymbol{y}_{<\tau})$ denotes the next-token predictive distribution induced by parameters $\boldsymbol{\theta}$ and prompt $\mathbf{P}$, and $\boldsymbol{y}_{<\tau}$ is sampled on-policy from the student and used as the shared prefix for both teacher and student distributions. Optimizing Eq. (4) realigns earlier prompts to the updated model, so new task adaptation need not preserve every previous behavior in the model parameters.

3.3 Anchored Prompt Space Learning

After alignment, LAPS has a set of task prompts $\{\mathbf{P}_{1,t}, \ldots, \mathbf{P}_{t,t}\}$ that remain usable under the current model $\boldsymbol{\theta}_t$. LAPS treats these task-specific soft prompts as anchors for their corresponding task knowledge. Using the anchors alone restricts inference to independently learned task solutions, leaving potential cross-task transfer unexplored. A natural alternative is convex interpolation, but its intermediate prompts are fixed linear combinations of the anchors and cannot be adapted through

cross-task supervision. LAPS therefore introduces learnable non-vertex Bézier control prompts to shape the space between the fixed anchors using replay losses across observed tasks.

At stage t, LAPS defines a simplex over the observed tasks:

$$\Delta^{t-1} = \left\{ \boldsymbol{\lambda} \in \mathbb{R}^t_{\geq 0} \;\middle|\; \sum_{j=1}^{t} \lambda_j = 1 \right\}. \tag{5}$$

Each component λ_j controls the relative emphasis on task $\mathcal{T}_j$. For any coordinate $\boldsymbol{\lambda} \in \Delta^{t-1}$, LAPS constructs a prompt $\mathbf{Z}_t(\boldsymbol{\lambda})$ with a degree-k Bézier simplex:

$$\mathbf{Z}_t(\boldsymbol{\lambda}) = \sum\nolimits_{j=1}^{t} \lambda_j^k \mathbf{P}_{j,t} + \sum\nolimits_{\boldsymbol{\alpha}} \frac{k!}{\prod_{j=1}^{t} \alpha_j!} \prod_{j=1}^{t} \lambda_j^{\alpha_j} \mathbf{C}_{\boldsymbol{\alpha},t}. \tag{6}$$

As shown in Fig. 1(b), the task prompts $\mathbf{P}_{j,t}$ form the simplex vertices, while the non-vertex control prompts $\mathbf{C}_{\boldsymbol{\alpha},t}$ shape the space between them. In the second sum, $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_t)$ ranges over the non-vertex degree-k multi-indices: each $\alpha_j \in \{0, 1, \ldots, k\}$, $\sum_{j=1}^{t} \alpha_j = k$, and $\boldsymbol{\alpha} \notin \{k\boldsymbol{e}_1, \ldots, k\boldsymbol{e}_t\}$, where $\boldsymbol{e}_j$ is the j-th unit vector. Each $\boldsymbol{\alpha}$ indexes one learnable control prompt $\mathbf{C}_{\boldsymbol{\alpha},t}$, so LAPS adds $\binom{t+k-1}{k} - t$ control prompts at stage t. For coordinate $\boldsymbol{\lambda}$, the Bernstein weight $\frac{k!}{\prod_{j=1}^{t} \alpha_j!} \prod_{j=1}^{t} \lambda_j^{\alpha_j}$ determines how strongly $\mathbf{C}_{\boldsymbol{\alpha},t}$ contributes to $\mathbf{Z}_t(\boldsymbol{\lambda})$. When $k = 1$, no non-vertex control prompts exist, and Eq. (6) reduces to convex interpolation among task anchors. When $k > 1$, the control prompts add learnable degrees of freedom between the task anchors. Notably, these additional control prompts account for less than 0.5% of the backbone parameters (see Appendix C).

To learn these degrees of freedom, LAPS keeps the model parameters $\boldsymbol{\theta}_t$ and the task anchors fixed and optimizes only the non-vertex control prompts, collected as $\mathbf{C}_t$. These prompts are learned from supervision across all observed tasks. For each observed task $j \leq t$, the loss induced by the coordinate $\boldsymbol{\lambda}$ and control prompts $\mathbf{C}_t$ is evaluated on examples from the stage-t replay set $\mathcal{R}_t$:

$$\ell_{j,t}(\boldsymbol{\lambda}, \mathbf{C}_t) = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{R}_t} \left[\ell_j \left(f(\boldsymbol{x}; \boldsymbol{\theta}_t, \mathbf{Z}_t(\boldsymbol{\lambda})), y \right) \right]. \tag{7}$$

Each $\boldsymbol{\lambda}$ specifies a desired trade-off among observed tasks. We compute the control prompt loss using a smooth Tchebycheff scalarization, which aggregates the $\boldsymbol{\lambda}$-weighted losses and smoothly emphasizes the larger weighted loss:

$$\min_{\mathbf{C}_t} \mathbb{E}_{\boldsymbol{\lambda}\sim\Delta^{t-1}} \left[\mu \log \sum\nolimits_{j=1}^{t} \exp\left(\frac{\lambda_j \ell_{j,t}(\boldsymbol{\lambda}, \mathbf{C}_t)}{\mu} \right) \right], \tag{8}$$

where $\mu > 0$ controls the smoothness of the scalarization. Optimizing Eq. (8) learns the non-vertex control prompts across sampled trade-offs $\boldsymbol{\lambda}$, allowing cross-task interactions to shape the anchored prompt space for subsequent task-specific search.

3.4 Validation-Guided Prompt Selection

After anchored prompt space learning, LAPS fixes the final model $\boldsymbol{\theta}_T$, task anchors, and Bézier control prompts, and uses validation data to select a soft prompt from the learned anchored space for each observed task. For task $\mathcal{T}_j$, the prompt $\mathbf{Z}_T(\boldsymbol{\lambda})$, parameterized by the simplex coordinate $\boldsymbol{\lambda} \in \Delta^{T-1}$, is directly optimized with the fixed model on the task's validation set:

$$\boldsymbol{\lambda}^\star_{j,T} = \arg\max_{\boldsymbol{\lambda}\in\Delta^{T-1}} \mathrm{Val}_{j,T}(\boldsymbol{\lambda}), \qquad j = 1, \ldots, T, \tag{9}$$

where $\mathrm{Val}_{j,T}(\boldsymbol{\lambda})$ denotes the resulting task-specific validation score, which may be discrete or otherwise unsuitable for gradient-based optimization (*e.g.*, accuracy, F1 score, or exact match). LAPS therefore treats $\mathrm{Val}_{j,T}$ as a black-box objective and approximates Eq. (9) with Bayesian optimization (Snoek et al., 2012; 2014). For each task, a Gaussian process surrogate is fitted to the observed coordinate–score pairs, and Expected Improvement (Jones et al., 1998) proposes coordinates that balance high predicted validation scores with uncertainty. Each proposed coordinate $\boldsymbol{\lambda}$ is used to construct $\mathbf{Z}_T(\boldsymbol{\lambda})$, which is evaluated on the validation set to form a new coordinate–score pair for updating the surrogate. After the search, the leading candidate coordinates are reevaluated on the full validation set over repeated runs, with the task's own anchor added if it is not already among the finalists. The coordinate with the highest mean validation score is mapped through Eq. (6) to obtain the final soft prompt for that task. Appendix B provides further details on the search and evaluation.

4 EXPERIMENT

In this section, we evaluate LAPS on continual instruction tuning. We first give the TRACE setup and comparison methods, then analyze aggregate performance, task-wise adaptation dynamics, and the learned prompt space. Finally, we present ablations and diagnostic studies of LAPS components.

4.1 EXPERIMENTAL SETUP

Benchmark. We conduct experiments on TRACE (Wang et al., 2023b), an eight-task continual instruction tuning benchmark consisting of C-STANCE, FOMC, MeetingBank, Py150, ScienceQA, NumGLUE-cm, NumGLUE-ds, and 20Minuten. Following prior work (Chen et al., 2026), we subsample 5,000 training examples from each task and use the official evaluation set.

Training. We evaluate all methods on Qwen3-1.7B, Qwen3-4B, and Qwen3-8B (Yang et al., 2025a). Methods that update model parameters use full-parameter fine-tuning with AdamW, a learning rate of 1×10^{-5}, a global batch size of 128, a maximum sequence length of 2,048, and `bf16` precision. Unless otherwise stated, prompt-based methods use 32 prompt tokens. Following the task order in TRACE, we train the eight tasks for $[5, 3, 7, 5, 3, 5, 5, 7]$ epochs, respectively. For replay-based methods, the replay ratio is set to 1%. For LAPS, we use a cubic Bézier simplex with degree $k = 3$, resulting in 112 control prompts for eight tasks. The scalarization smoothing parameter is $\mu = 0.1$. We construct the alignment dataset by combining replay data from previous tasks and data from the current task at a $1:1$ ratio. Bayesian optimization is performed on the official validation set with a budget of 100 evaluations per task, except for MeetingBank, where we use 50 evaluations.

Baselines. We compare against baselines that stress different continual-adaptation mechanisms. SFT is a *parameter-only adaptation* baseline; vanilla replay (Rolnick et al., 2019) and on-policy replay (OPR-SC and OPR-RU, Chen et al., 2026) add stored or selected previous task examples; LwF (Li & Hoiem, 2018) uses *distillation-based adaptation*; ProgPrompt (Razdaibiedina et al., 2023) performs *prompt-only adaptation*; and ProMoT (Wang et al., 2024) implements *joint prompt–parameter adaptation*. Together, these baselines enable a systematic comparison of retention behavior across parameter-only adaptation, replay, distillation, frozen-backbone prompt tuning, and joint prompt–parameter adaptation under continual learning.

Metrics. We report average performance (**Average**) across all tasks and backward transfer (**BWT**, Lopez-Paz & Ranzato, 2017) as a measure of forgetting. Let $R_{j,t}$ denote the performance on task $\mathcal{T}_j$ after learning stage t. We define $\mathrm{BWT} = \frac{1}{T-1}\sum_{j=1}^{T-1}(R_{j,T} - R_{j,j})$. Higher Average indicates better final performance averaged across tasks, while higher BWT indicates stronger retention of earlier tasks. All results are averaged over eight inference runs with a sampling temperature of 0.1.

Method characteristics and trainable parameter counts are provided in Appendix C; dataset statistics and evaluation metrics in Appendix D; and additional experimental analyses in Appendix E.

4.2 MAIN RESULTS

Table 1 shows the adaptation–retention pattern across the three Qwen3 model scales. Parameter updating methods such as SFT, Replay, OPR-SC, OPR-RU, and ProMoT often improve new-task performance, but their BWT values show that earlier task behavior can still be overwritten. ProgPrompt has the opposite profile: freezing the backbone helps preserve performance on previous tasks, yet its Average remains lower than the best parameter-updating methods on several scales. LAPS attains the highest Average on all three model scales, with 60.5%, 66.2%, and 67.5% on Qwen3-1.7B, Qwen3-4B, and Qwen3-8B, respectively, while keeping BWT close to or above zero. The most directly comparable baseline is ProMoT, since it also combines soft prompts with model parameter updates. Relative to ProMoT, LAPS improves Average by 0.2%–2.8% and BWT by 1.4%–3.2%. This consistent pattern across model scales suggests a more favorable adaptation–retention trade-off that is not confined to a particular model size.

Continual Adaptation Dynamics. Beyond the aggregate table, the task-wise trajectories in Fig. 2 reveal when forgetting occurs during the continual sequence. On Qwen3-1.7B, SFT illustrates the failure mode of sequential fine-tuning: later updates are followed by degradation on several earlier tasks. ProgPrompt keeps the backbone and previous prompts fixed, which helps retention, but this constraint leaves less room for new-task adaptation. ProMoT adapts both prompts and model

Table 1: **Performance across Qwen3 model scales.** The best and second best results at each model scale are shown in **bold** and underlined, respectively.

Method	C-STANCE	FOMC	MeetingBank	Py150	ScienceQA	NG-cm	NG-ds	20Minuten	Average ↑	BWT ↑
Qwen3-1.7B										
Initial	36.3	32.7	18.4	2.5	63.4	70.2	30.2	35.8	36.2	N/A
SFT	53.1	58.5	20.1	52.6	76.7	46.3	53.8	38.8	50.0	−9.3
Replay	52.8	62.4	46.6	56.6	73.4	51.1	53.2	38.9	54.4	−4.1
OPR-RU	50.4	58.2	47.0	54.7	77.1	55.7	51.7	38.8	54.2	−4.9
OPR-SC	50.7	51.9	38.6	55.0	77.7	47.5	52.6	38.7	51.6	−7.6
LwF	45.9	28.4	38.2	43.4	41.6	43.1	53.5	38.0	41.5	−13.8
ProgPrompt	**53.5**	60.6	47.7	49.8	73.8	45.4	47.6	38.4	52.1	**0.0**
ProMoT	50.9	62.7	50.9	59.2	**84.2**	47.7	**67.7**	**40.5**	58.0	−4.4
LAPS (Ours)	52.0	**69.4**	**58.7**	**60.0**	83.4	**56.0**	64.8	40.0	**60.5**	−1.2
Qwen3-4B										
Initial	42.7	5.0	20.6	2.4	64.6	76.4	38.2	35.9	35.7	N/A
SFT	52.5	66.2	54.3	37.7	89.2	**66.5**	77.3	40.8	60.6	−6.7
Replay	55.3	71.4	59.0	62.3	91.9	64.0	75.0	40.5	64.9	−2.7
OPR-RU	53.0	**73.3**	57.7	62.2	91.1	63.9	76.9	**41.1**	64.9	−1.2
OPR-SC	52.8	61.9	58.0	60.4	82.7	62.8	**77.8**	40.4	62.1	−5.5
LwF	56.0	71.1	47.7	38.8	83.3	65.1	69.4	39.9	58.9	**+0.2**
ProgPrompt	57.8	62.2	51.4	57.9	79.7	55.4	52.4	39.0	57.0	0.0
ProMoT	51.9	59.3	**62.9**	**63.0**	**93.4**	63.1	72.7	41.0	63.4	−2.0
LAPS (Ours)	**59.5**	71.9	60.3	61.6	92.4	66.4	76.9	40.6	**66.2**	−0.3
Qwen3-8B										
Initial	52.6	21.0	20.8	2.4	68.0	77.8	38.4	36.4	39.6	N/A
SFT	47.3	63.5	50.0	50.0	73.4	67.9	77.7	41.5	58.9	−10.1
Replay	52.3	71.9	55.2	62.1	92.4	67.1	**79.5**	41.4	65.2	−2.9
OPR-RU	52.6	**74.5**	60.8	62.6	92.0	69.1	78.5	41.1	66.4	−2.7
OPR-SC	51.8	72.6	57.0	62.1	93.0	**69.8**	76.7	41.5	65.6	−2.5
LwF	49.5	72.0	46.3	30.8	84.7	52.0	66.1	39.6	55.1	**+4.3**
ProgPrompt	**58.6**	64.9	54.9	60.7	84.9	47.5	56.0	39.3	58.4	0.0
ProMoT	55.9	69.8	64.5	**64.1**	**94.6**	69.3	78.5	41.6	67.3	−0.2
LAPS (Ours)	57.6	73.4	**66.4**	**64.1**	92.4	66.2	78.3	**41.8**	**67.5**	+1.2

parameters and improves performance on several new tasks, yet earlier tasks such as Py150 and NumGLUE-cm still decline as the model parameters continue to change across the task sequence.

In contrast, LAPS exhibits a different trajectory. After sequential adaptation ends at $\mathcal{T}_8$, indicated by the vertical gray dashed line, the model is fixed and Bayesian optimization selects a prompt for each task from the learned anchored prompt space. The rightmost point in each subplot shows performance with the selected prompt, compared with the preceding pre-search point. Improvements on most tasks suggest that useful task-specific regions exist beyond the original task anchors.

Visualization of the Learned Prompt Space. Fig. 3 uses Principal Component Analysis (PCA) to examine where validation-selected prompts lie relative to the original per-task prompt anchors. The selected prompts are generally not located at the anchors and improve average performance by 0.78%, 0.48%, and 2.70% over using the anchors on Qwen3-1.7B, Qwen3-4B, and Qwen3-8B, respectively. This pattern suggests that the Bézier control prompts define a searchable prompt space in which useful task-specific prompts need not coincide with the original anchors.

4.3 Ablations and Diagnostic Analyses

Effect of LAPS Components. Table 2 examines the contributions of historical prompt alignment, anchored prompt space learning, and validation-guided prompt selection. The full model outperforms each removal variant, indicating that the gains do not arise from any single component alone. Alignment has the clearest effect on retention: removing it causes the largest degradation in BWT while also lowering Average. This pattern is consistent with the role of alignment, which keeps historical prompts compatible with the updated model for each new task.

Prompt space learning and selection provide additional benefits once alignment is present. Removing either component weakens the final trade-off, but the two ablations isolate different functions: prompt space learning tests whether the intermediate prompt geometry is learned, whereas prompt selection tests whether each task can choose a suitable prompt after training. The results suggest

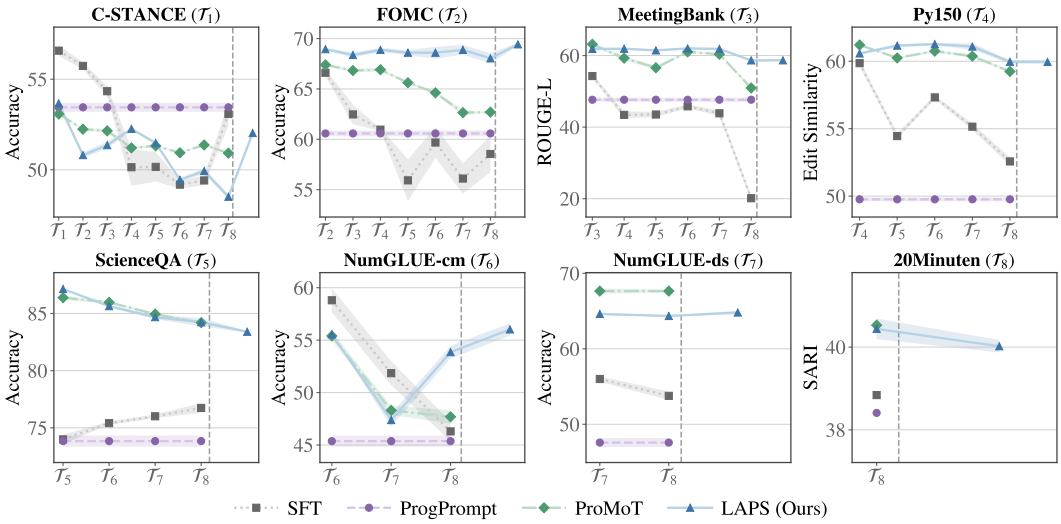


Figure 2: **Task-wise continual adaptation trajectories with Qwen3-1.7B.** Each panel tracks one task after it is learned, and later points show its performance after subsequent adaptation stages. Drops indicate forgetting as the model learns new tasks. The gray dashed line marks the transition from sequential adaptation to the validation-guided prompt selection introduced in LAPS.

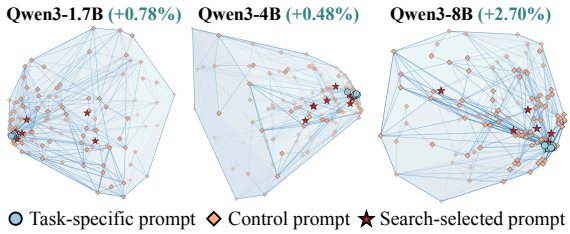


Figure 3: PCA projection of learned anchored prompt spaces across Qwen3 model scales.

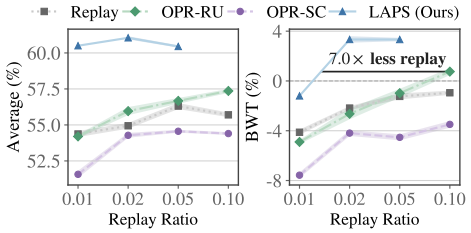


Figure 4: Average performance and BWT across replay ratios on Qwen3-1.7B.

Table 2: **Ablation of the three LAPS components:** historical prompt alignment, prompt space learning, and prompt selection.

Components			Qwen3-1.7B		Qwen3-4B	
Alignment	Space Learning	Selection	Average	BWT	Average	BWT
✓	✓	✓	**60.5**	**−1.2**	**66.2**	**−0.3**
✗	✓	✓	58.8	−4.0	65.6	−0.4
✓	✗	✓	59.6	−2.3	65.5	−0.5
✓	✓	✗	59.8	−2.1	65.8	−0.7
✗	✗	✗	57.3	−5.8	64.3	−1.9

Table 3: **Prompt-based baselines with replay and LAPS components.** † denotes the variant without prompt concatenation.

Method	Qwen3-1.7B		Qwen3-4B	
	Average	BWT	Average	BWT
ProgPrompt†	52.3	0.0	58.9	0.0
+ Replay	50.8	0.0	55.1	0.0
+ Space Learning & Selection	**53.2**	**+1.0**	**60.5**	**+1.8**
ProMoT	58.0	−4.4	63.4	−2.0
+ Replay	54.2	−6.9	63.9	−2.8
+ Space Learning & Selection	**58.8**	**−4.0**	**65.6**	**−0.4**

complementary roles: alignment is most closely tied to retaining historical behavior, while space learning and selection expand the learned prompt space and use it for task-specific prompt selection.

Transferability of LAPS Components. Table 3 compares replay augmentation with integrating the prompt space learning and selection components of LAPS into ProgPrompt and ProMoT. Adding replay alone provides limited or inconsistent benefits and can even reduce performance. Adding prompt space learning and selection improves both Average and BWT across the reported model scales. These comparisons suggest that organizing and searching the prompt space provides additional adaptation and retention benefits beyond replay alone.

Effect of the Replay Ratio. Fig. 4 compares LAPS with replay-based methods under different replay ratios, testing whether retention is mainly explained by the number of previous-task examples. Increasing the replay ratio does not lead to monotonic improvements, and LAPS remains competitive with a much smaller replay budget. These results suggest that retention depends not only on replay volume but also on how historical samples are used for alignment and prompt space learning.

Table 4: Effect of the soft prompt length. Results are averaged over eight evaluations.

Method	Qwen3-1.7B						Qwen3-4B					
	Length = 32		Length = 64		Length = 128		Length = 32		Length = 64		Length = 128	
	Average	BWT	Average	BWT	Average	BWT	Average	BWT	Average	BWT	Average	BWT
ProgPrompt	52.1	**0.0**	49.2	**0.0**	51.2	0.0	57.0	**0.0**	57.4	**0.0**	57.5	**0.0**
ProMoT	58.0	−4.4	55.8	−4.6	48.6	−6.9	63.4	−2.0	65.5	−0.2	62.6	−3.2
LAPS (Ours)	**60.5**	−1.2	**59.5**	−1.6	**58.5**	**0.7**	**66.2**	−0.3	**65.6**	−0.4	**64.7**	−1.3

Table 5: Effect of the Bézier degree k.

k	# Control Prompts	Qwen3-1.7B		Qwen3-4B	
		Average	BWT	Average	BWT
1	0	59.6	−2.3	65.5	−0.5
2	28	60.4	−2.1	65.7	−0.4
3	112	**60.5**	**−1.2**	**66.2**	**−0.3**
4	322	59.3	−2.4	65.2	−0.9

Table 6: LAPS with on-policy replay.

Method	Qwen3-1.7B		Qwen3-4B	
	Average	BWT	Average	BWT
OPR-SC	51.6	−7.6	62.1	−5.5
OPR-SC + LAPS	**60.0**	**−1.1**	**64.2**	**−1.5**
OPR-RU	54.2	−4.9	64.9	−1.2
OPR-RU + LAPS	**60.0**	**−2.0**	**65.5**	**−0.4**

Effect of the Soft Prompt Length. Table 4 varies the number of soft prompt tokens to test whether the observed gains depend on a particular prompt length. Increasing the prompt length does not consistently improve performance, and the baselines show greater sensitivity to this choice. Across all tested prompt lengths and model scales, LAPS maintains the highest Average, suggesting that its aggregate advantage does not depend on a narrowly tuned prompt length.

Effect of the Bézier Degree k. Table 5 studies the capacity of the learned prompt space by varying the Bézier degree. At $k = 1$, prompt search is limited to convex combinations of the task anchors. Increasing k introduces learnable control prompts and consistently improves both Average and BWT up to $k = 3$ across model scales. Further increasing k to 4 increases the number of control prompts but brings no additional gain, suggesting that a moderate degree is sufficient in this setting.

Compatibility of LAPS with On-Policy Replay. LAPS can be combined with on-policy replay methods. Table 6 shows that adding LAPS improves both Average and BWT for OPR-SC and OPR-RU across the reported model scales. These results indicate compatibility with on-policy replay: replay selects which historical examples are revisited, while LAPS provides an alignment and prompt selection mechanism for using the selected examples.

5 Conclusion and Discussion

We presented LAPS for continual LLM adaptation by jointly updating prompts and model parameters. LAPS aligns historical prompts with the evolving model through self-distillation, learns an anchored prompt space that captures cross-task relationships, and selects task-specific soft prompts through validation-guided search. Experiments on TRACE across three Qwen3 scales show that LAPS achieves the highest average performance among compared methods while reducing forgetting. These results suggest that organizing task-specific soft prompts into a structured, searchable space can facilitate cross-task knowledge transfer during continual adaptation.

The current formulation assumes known task identities and introduces additional control prompts as tasks accumulate. This becomes more challenging in open-ended learning, where task boundaries are unavailable and prompt count can grow continuously. A natural extension is to learn a task-free router that selects prompts from the input, without explicit task labels. Meanwhile, prompt capacity could be managed by adding new control prompts only when needed and consolidating redundant ones, thereby supporting flexible cross-task interactions while keeping memory overhead bounded.

The learned anchored prompt space also provides a way to study which previous tasks are useful for a new task. By evaluating how task losses change across the anchored prompt space, LAPS could identify regions associated with positive transfer or interference and use this information to guide prompt selection. The learned anchored space could also provide better initializations for new task prompts by reusing nearby anchors or control prompts, reducing the amount of data and optimization needed for adaptation. These directions would enable LAPS to reuse cross-task structure more explicitly and improve the efficiency of future adaptation.

Reproducibility Statement

We provide implementation details, training configurations, and additional experimental results in the supplementary material. An anonymous repository containing the source code and configuration files is also provided for reproducibility. The repository will be made public upon publication.

AI Assistance Statement

We used AI tools for limited assistance in improving writing clarity and formatting. All research ideas, methodology, experiments, and analyses were developed and verified by the authors.

References

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: A strong, simple baseline. In *Advances in Neural Information Processing Systems*, pp. 15920–15930, 2020.

Yan Chen, Taojie Zhu, Meng Zhang, Xin Chen, Jiaqi Huang, Dongyang Xu, and Yizhi Wang. On-policy replay for continual supervised fine-tuning. *arXiv preprint arXiv:2605.29495*, 2026.

Zican Dong, Junyi Li, Jinhao Jiang, Mingyu Xu, Wayne Xin Zhao, Bingning Wang, and Weipeng Chen. LongReD: Mitigating short-text degradation of long-context large language models via restoration distillation. In *Annual Meeting of the Association for Computational Linguistics (Long Papers)*, pp. 10687–10707, 2025.

Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. PODNet: Pooled outputs distillation for small-tasks incremental learning. In *European Conference on Computer Vision*, pp. 86–102, 2020.

William Fedus, Prajit Ramachandran, Rishabh Agarwal, Yoshua Bengio, Hugo Larochelle, Mark Rowland, and Will Dabney. Revisiting fundamentals of experience replay. In *International Conference on Machine Learning*, pp. 3061–3071, 2020.

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

Rui Gao and Weiwei Liu. DDGR: Continual learning with deep diffusion-based generative replay. In *International Conference on Machine Learning*, pp. 10744–10763, 2023.

Zhanxin Gao, Jun Cen, and Xiaobin Chang. Consistent prompting for rehearsal-free continual learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 28463–28473, 2024.

Stephen Grossberg. How does a brain build a cognitive code? *Psychological Review*, 87(1):1–51, 1980.

Jisu Han, Jaemin Na, and Wonjun Hwang. Semantic prompting with image token for continual learning. In *IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 6987–6997, 2025.

Kiseong Hong, Gyeong-hyeon Kim, and Eunwoo Kim. RainbowPrompt: Diversity-enhanced prompt-evolving for continual learning. In *IEEE/CVF International Conference on Computer Vision*, pp. 1130–1140, 2025.

Adam Ibrahim, Benjamin Thérien, Kshitij Gupta, Mats Leon Richter, Quentin Gregory Anthony, Eugene Belilovsky, Timothée Lesort, and Irina Rish. Simple and scalable strategies to continually pre-train large language models. *Transactions on Machine Learning Research*, 2024.

David Isele and Akansel Cosgun. Selective experience replay for lifelong learning. In *AAAI Conference on Artificial Intelligence*, pp. 3302–3309, 2018.

Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, 1998.

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.

Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics-Doklady*, 10(8):707–710, 1966.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2018.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, pp. 6470–6479, 2017.

Aru Maekawa, Hidetaka Kamigaito, Kotaro Funakoshi, and Manabu Okumura. Generative replay inspired by hippocampal memory indexing for continual language learning. In *Conference of the European Chapter of the Association for Computational Linguistics*, pp. 930–942, 2023.

James L. McClelland, Bruce L. McNaughton, and Randall C. O'Reilly. Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. *Psychological Review*, 102(3):419–457, 1995.

Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of Learning and Motivation*, volume 24, pp. 109–165. Academic Press, 1989.

Nikhil Shivakumar Nayak, Krishnateja Killamsetty, Ligong Han, Abhishek Bhandwaldar, Prateek Chanda, Kai Xu, Oleg Silkin, Mustafa Eyceoz, Hao Wang, Aldo Pareja, et al. Sculpting subspaces: Constrained full fine-tuning in LLMs for continual learning. In *International Conference on Learning Representations*, pp. 20212–20243, 2026.

Bohao Peng, Zhuotao Tian, Shu Liu, Ming-Chang Yang, and Jiaya Jia. Scalable language model with generalized continual learning. In *International Conference on Learning Representations*, pp. 28412–28434, 2024.

Jiabao Qiu, Zixuan Ke, and Bing Liu. Continual learning using only large language model prompting. In *International Conference on Computational Linguistics*, pp. 6014–6023, 2025.

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.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. In *Advances in Neural Information Processing Systems*, pp. 348–358, 2019.

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning. In *International Conference on Machine Learning*, 2026a.

Idan Shenfeld, Jyothish Pari, and Pulkit Agrawal. RL's razor: Why online reinforcement learning forgets less. In *International Conference on Learning Representations*, pp. 59839–59864, 2026b.

James Seale Smith, Leonid Karlinsky, Vyshnavi Gutta, Paola Cascante-Bonilla, Donghyun Kim, Assaf Arbelle, Rameswar Panda, Rogerio Feris, and Zsolt Kira. CODA-Prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11909–11919, 2023.

Jasper Snoek, Hugo Larochelle, and Ryan Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, pp. 2951–2959, 2012.

Jasper Snoek, Kevin Swersky, Rich Zemel, and Ryan Adams. Input warping for Bayesian optimization of non-stationary functions. In *International Conference on Machine Learning*, pp. 1674–1682, 2014.

Rishabh Tiwari, Kusha Sareen, Lakshya A. Agrawal, Joseph E. Gonzalez, Matei Zaharia, Kurt Keutzer, Inderjit S. Dhillon, Rishabh Agarwal, and Fnu Devvrit. Learning, fast and slow: Towards LLMs that adapt continually. In *COLM Workshop on Context Beyond the Window: Persistent Knowledge in Language Models*, 2026.

Gido M. Van de Ven, Tinne Tuytelaars, and Andreas S. Tolias. Three types of incremental learning. *Nature Machine Intelligence*, 4(12):1185–1197, 2022.

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 Empirical Methods in Natural Language Processing*, 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.

Yihan Wang, Si Si, Daliang Li, Michal Lukasik, Felix Yu, Cho-Jui Hsieh, Inderjit Dhillon, and Sanjiv Kumar. Two-stage LLM fine-tuning with less specialization and more generalization. In *International Conference on Learning Representations*, pp. 20380–20398, 2024.

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, 2022a.

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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 139–149, 2022b.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415, 2016.

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*, 2025a.

Yutao Yang, Jie Zhou, Xuanwen Ding, Tianyu Huai, Shunyu Liu, Qin Chen, Yuan Xie, and Liang He. Recent advances of foundation language models-based continual learning: A survey. *ACM Computing Surveys*, 57(5):1–38, 2025b.

Xiaobing Yu, Jin Yang, Xiao Wu, Peijie Qiu, and Xiaofeng Liu. FM-LoRA: Factorized low-rank meta-prompting for continual learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 6399–6408, 2025.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. In *International Conference on Machine Learning*, 2026.

A Properties of the Anchored Prompt Space

We analyze the optimization of the shared Bézier parameters. During this stage, the LLM parameters and task-specific prompt anchors remain fixed, and only the non-vertex control prompts are updated. We adopt the following standard assumptions.

Assumption 1 (Regularity of Task Losses). *For every observed task i and simplex coordinate $\boldsymbol{\lambda} \in \Delta^{t-1}$, the replay loss $\ell_i(\boldsymbol{\lambda}, \mathbf{C})$ is continuously differentiable with respect to the non-vertex control prompts $\mathbf{C}$. Moreover, there exist constants $\beta > 0$ and $M > 0$ such that, for all feasible $\mathbf{C}$ and $\mathbf{C}'$,*

$$\|\nabla_{\mathbf{C}} \ell_i(\boldsymbol{\lambda}, \mathbf{C}) - \nabla_{\mathbf{C}} \ell_i(\boldsymbol{\lambda}, \mathbf{C}')\|_2 \leq \beta \|\mathbf{C} - \mathbf{C}'\|_2 , \tag{10}$$

$$\|\nabla_{\mathbf{C}} \ell_i(\boldsymbol{\lambda}, \mathbf{C})\|_2 \leq M. \tag{11}$$

We further assume that the expected smooth Tchebycheff objective is lower bounded, i.e., there exists a constant $G_{\inf}$ such that $G_t(\mathbf{C}) \geq G_{\inf}$ for all feasible $\mathbf{C}$.

The regularity of the task losses first implies smoothness of the scalarized objective.

Lemma 1 (Smoothness of the Smooth Tchebycheff objective). *Under Assumption 1, for every fixed $\boldsymbol{\lambda} \in \Delta^{t-1}$, the pointwise objective $g_\mu(\boldsymbol{\lambda}, \mathbf{C})$ has a Lipschitz-continuous gradient with respect to $\mathbf{C}$ for every $\mu > 0$. Consequently,*

$$G_t(\mathbf{C}) = \mathbb{E}_{\boldsymbol{\lambda} \sim \Delta^{t-1}} \left[g_\mu(\boldsymbol{\lambda}; \mathbf{C}) \right] \tag{12}$$

is L_G-smooth for some finite constant L_G.

Proof. For a fixed $\boldsymbol{\lambda}$, the gradient of the smooth Tchebycheff objective g_μ is

$$\nabla_{\mathbf{C}} g_\mu = \sum_{i=1}^{t} q_i \lambda_i \nabla_{\mathbf{C}} \ell_i. \tag{13}$$

The first condition in Assumption 1 ensures that the gradient of every task loss varies Lipschitz-continuously with $\mathbf{C}$. The second condition bounds the variation introduced through the softmax coefficients q_i, whose derivatives are scaled by $1/\mu$. For any fixed $\mu > 0$, both contributions are therefore bounded, implying that $\nabla_{\mathbf{C}} g_\mu$ is Lipschitz continuous.

The expectation over $\boldsymbol{\lambda} \sim \Delta^{t-1}$ preserves Lipschitz continuity of the gradient. Hence, $G_t(\mathbf{C})$ is L_G-smooth for some finite L_G. □

Lemma 1 establishes the smoothness of the smooth Tchebycheff objective. We next exploit the Bézier parameterization to make its smoothness constant explicit, which will be used in the subsequent convergence analysis.

Lemma 2 (Smoothness Bound for the Bézier–Smooth Tchebycheff Objective). *Under Assumption 1, if $\boldsymbol{\lambda}$ is sampled uniformly from Δ^{t-1}, the smoothness constant in Lemma 1 satisfies*

$$L_G \leq R_{t,k} \left(\beta + \frac{M^2}{\mu} \right), \tag{14}$$

where

$$R_{t,k} = \frac{(k!)^2 (t-1)!}{(t+2k-1)!} \left[\frac{4^k (t/2)_k}{k!} - t \binom{2k}{k} \right], \tag{15}$$

and $(a)_k = a(a+1)\cdots(a+k-1)$ denotes the rising factorial.

Proof. For fixed $\boldsymbol{\lambda}$, the Bézier prompt can be written as

$$\mathbf{Z}_t(\boldsymbol{\lambda}) = \sum_{j=1}^{t} \lambda_j^k \mathbf{P}_{j,t} + \sum_{\boldsymbol{\alpha}} B_{\boldsymbol{\alpha}}^k(\boldsymbol{\lambda}) \mathbf{C}_{\boldsymbol{\alpha},t}. \tag{16}$$

Since the task anchors $\{\mathbf{P}_{j,t}\}_{j=1}^t$ and $\boldsymbol{\lambda}$ are fixed during control prompt optimization, the first term is constant with respect to $\mathbf{C}_t$. Hence, $\mathbf{Z}_t(\boldsymbol{\lambda})$ is affine in the non-vertex control prompts. Define

$$\rho_{t,k}^2(\boldsymbol{\lambda}) = \sum_{\boldsymbol{\alpha}} \left(B_{\boldsymbol{\alpha}}^k(\boldsymbol{\lambda}) \right)^2, \tag{17}$$

where the sum is over the non-vertex degree-k multi-indices. Since the Bézier map is affine in the control prompts, its Jacobian satisfies

$$\|J_{\mathbf{Z},\mathbf{C}}(\boldsymbol{\lambda})\|_{op}^2 = \rho_{t,k}^2(\boldsymbol{\lambda}). \tag{18}$$

Therefore, the assumed gradient bound and β-smoothness of each replay loss imply

$$\left\|\nabla_{\mathbf{C}}\mathcal{L}_{i,t}^{\mathrm{r}}\right\|_{\mathrm{F}} \le M\rho_{t,k}(\boldsymbol{\lambda}), \tag{19}$$

$$\left\|\nabla_{\mathbf{C}}^2\mathcal{L}_{i,t}^{\mathrm{r}}\right\|_{op} \le \beta\rho_{t,k}^2(\boldsymbol{\lambda}). \tag{20}$$

For fixed $\boldsymbol{\lambda}$, let

$$g_\mu(\boldsymbol{\lambda},\mathbf{C}) = \mu\log\sum_{i=1}^{t}\exp\left(\frac{\lambda_i\ell_{i,t}}{\mu}\right), \tag{21}$$

and define the corresponding softmax weights

$$q_i = \frac{\exp(\lambda_i\ell_{i,t}/\mu)}{\sum_{j=1}^{t}\exp(\lambda_j\ell_{j,t}/\mu)}. \tag{22}$$

Its Hessian with respect to the control prompts can be written as

$$\nabla_{\mathbf{C}}^2 g_\mu = \sum_{i=1}^{t} q_i\lambda_i\nabla_{\mathbf{C}}^2\ell_{i,t} + \frac{1}{\mu}\operatorname{Cov}_q\left(\lambda_i\nabla_{\mathbf{C}}\ell_{i,t}\right). \tag{23}$$

Since $0 \le \lambda_i \le 1$ and $\sum_i q_i = 1$, applying Eq. (20) gives

$$\left\|\nabla_{\mathbf{C}}^2 g_\mu(\boldsymbol{\lambda},\mathbf{C})\right\|_{op} \le \rho_{t,k}^2(\boldsymbol{\lambda})\left(\beta + \frac{M^2}{\mu}\right). \tag{24}$$

Taking expectation over uniformly sampled $\boldsymbol{\lambda} \in \Delta^{t-1}$ therefore yields

$$L_G \le \mathbb{E}_{\boldsymbol{\lambda}}\left[\rho_{t,k}^2(\boldsymbol{\lambda})\right]\left(\beta + \frac{M^2}{\mu}\right). \tag{25}$$

It remains to evaluate the expectation. For uniform sampling over the simplex, the Dirichlet moment identity gives

$$\mathbb{E}_{\boldsymbol{\lambda}}\left[\prod_{j=1}^{t}\lambda_j^{2\alpha_j}\right] = \frac{(t-1)!}{(t+2k-1)!}\prod_{j=1}^{t}(2\alpha_j)!. \tag{26}$$

Using

$$B_{\boldsymbol{\alpha}}^k(\boldsymbol{\lambda}) = \frac{k!}{\prod_{j=1}^{t}\alpha_j!}\prod_{j=1}^{t}\lambda_j^{\alpha_j}, \tag{27}$$

we obtain

$$\mathbb{E}_{\boldsymbol{\lambda}}\left[\rho_{t,k}^2(\boldsymbol{\lambda})\right] = \frac{(k!)^2(t-1)!}{(t+2k-1)!}\sum_{\boldsymbol{\alpha}}\prod_{j=1}^{t}\binom{2\alpha_j}{\alpha_j}, \tag{28}$$

where the sum again excludes the vertex multi-indices.

The generating function identity

$$\sum_{\substack{\boldsymbol{\alpha}\ge 0\\ \sum_j\alpha_j=k}}\prod_{j=1}^{t}\binom{2\alpha_j}{\alpha_j} = \frac{4^k(t/2)_k}{k!} \tag{29}$$

sums over all degree-k multi-indices. Each of the t excluded vertex indices contributes $\binom{2k}{k}$. Hence,

$$\mathbb{E}_{\boldsymbol{\lambda}}\left[\rho_{t,k}^2(\boldsymbol{\lambda})\right] = \frac{(k!)^2(t-1)!}{(t+2k-1)!}\left[\frac{4^k(t/2)_k}{k!} - t\binom{2k}{k}\right] = R_{t,k}. \tag{30}$$

Substituting this expression into Eq. (25) proves Eq. (14). □

We can now establish convergence of the shared control prompt optimization. Since the induced objective is generally non-convex, the appropriate guarantee is convergence to a first-order stationary point rather than global optimality.

Theorem 1 (Convergence of Bézier–Smooth Tchebycheff Optimization)**.** *Given Lemma 2, suppose that $G_t(\mathbf{C})$ is lower bounded by $G_{\inf}$. Then, gradient descent with learning rate $\eta = 1/L_G$ satisfies*

$$\frac{1}{K}\sum_{s=0}^{K-1}\left\|\nabla G_t(\mathbf{C}^{(s)})\right\|_{\mathrm{F}}^2 \le \frac{2R_{t,k}\left(\beta+\frac{M^2}{\mu}\right)\left(G_t(\mathbf{C}^{(0)})-G_{\inf}\right)}{K}. \tag{31}$$

Proof. By Lemma 1, G_t is L_G-smooth. Therefore, for the gradient descent update

$$\mathbf{C}^{(s+1)} = \mathbf{C}^{(s)} - \eta\nabla G_t(\mathbf{C}^{(s)}), \tag{32}$$

the descent lemma gives

$$G_t(\mathbf{C}^{(s+1)}) \le G_t(\mathbf{C}^{(s)}) + \left\langle \nabla G_t(\mathbf{C}^{(s)}), \mathbf{C}^{(s+1)}-\mathbf{C}^{(s)}\right\rangle + \frac{L_G}{2}\left\|\mathbf{C}^{(s+1)}-\mathbf{C}^{(s)}\right\|_{\mathrm{F}}^2. \tag{33}$$

Substituting the update yields

$$G_t(\mathbf{C}^{(s+1)}) \le G_t(\mathbf{C}^{(s)}) - \eta\left(1-\frac{L_G\eta}{2}\right)\left\|\nabla G_t(\mathbf{C}^{(s)})\right\|_{\mathrm{F}}^2. \tag{34}$$

For $\eta \le 1/L_G$,

$$G_t(\mathbf{C}^{(s+1)}) \le G_t(\mathbf{C}^{(s)}) - \frac{\eta}{2}\left\|\nabla G_t(\mathbf{C}^{(s)})\right\|_{\mathrm{F}}^2. \tag{35}$$

Summing Eq. (35) over $s = 0, \dots, K-1$ gives

$$\frac{\eta}{2}\sum_{s=0}^{K-1}\left\|\nabla G_t(\mathbf{C}^{(s)})\right\|_{\mathrm{F}}^2 \le G_t(\mathbf{C}^{(0)}) - G_t(\mathbf{C}^{(K)}). \tag{36}$$

Since G_t is lower bounded by $G_{\inf}$,

$$\frac{1}{K}\sum_{s=0}^{K-1}\left\|\nabla G_t(\mathbf{C}^{(s)})\right\|_{\mathrm{F}}^2 \le \frac{2\left(G_t(\mathbf{C}^{(0)})-G_{\inf}\right)}{\eta K}. \tag{37}$$

Choosing $\eta = 1/L_G$ gives

$$\frac{1}{K}\sum_{s=0}^{K-1}\left\|\nabla G_t(\mathbf{C}^{(s)})\right\|_{\mathrm{F}}^2 \le \frac{2L_G\left(G_t(\mathbf{C}^{(0)})-G_{\inf}\right)}{K}. \tag{38}$$

Using Lemma 2,

$$L_G \le R_{t,k}\left(\beta+\frac{M^2}{\mu}\right), \tag{39}$$

we obtain

$$\frac{1}{K}\sum_{s=0}^{K-1}\left\|\nabla G_t(\mathbf{C}^{(s)})\right\|_{\mathrm{F}}^2 \le \frac{2R_{t,k}\left(\beta+\frac{M^2}{\mu}\right)\left(G_t(\mathbf{C}^{(0)})-G_{\inf}\right)}{K}, \tag{40}$$

which proves Eq. (31). □

Remark 1 (Scope of the Convergence Guarantee)**.** *Theorem 1 characterizes optimization of the shared non-vertex Bézier control prompts while the LLM parameters and task-specific anchors remain fixed. The guarantee concerns convergence to first-order stationarity of the expected smooth Tchebycheff objective rather than global optimality of the underlying non-convex problem. Importantly, the bound makes the LAPS-specific dependence explicit: $R_{t,k}$ captures the sensitivity induced by the Bézier parameterization, while M^2/μ reflects the additional curvature introduced by the smooth Tchebycheff objective.*

B Bayesian Optimization for Positive Transfer Discovery

After constructing the final anchored prompt space, LAPS performs Bayesian optimization (BO) independently for each task. On the TRACE benchmark, the search variable is

$$\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_8) \in \Delta^7, \qquad \lambda_j \geq 0, \qquad \sum_{j=1}^{8} \lambda_j = 1. \tag{41}$$

Given $\boldsymbol{\lambda}$, the cubic Bézier simplex deterministically constructs the corresponding prompt $\mathbf{Z}_T(\boldsymbol{\lambda})$.

Task-wise Search. We perform an independent procedure for each task $\mathcal{T}_j$. Thus, each task obtains its own coordinate $\boldsymbol{\lambda}^\star_{j,T}$ rather than sharing a single global coordinate. The task-specific search objective is

$$\boldsymbol{\lambda}^\star_{j,T} = \arg\max_{\boldsymbol{\lambda} \in \Delta^7} \mathrm{Val}_j(\boldsymbol{\lambda}), \tag{42}$$

where $\mathrm{Val}_j(\boldsymbol{\lambda})$ denotes the validation score of $\mathbf{Z}_T(\boldsymbol{\lambda})$ on task $\mathcal{T}_j$, measured by its task-specific evaluation metric; the metric for each task is reported in Table 8.

Initial Design. Each run starts from 16 simplex coordinates: all eight task vertices, the simplex centroid $\boldsymbol{\lambda}_{\text{center}} = (1/8, \ldots, 1/8)$, and seven interior coordinates generated using a Sobol sequence and mapped onto the simplex. Explicitly including all task vertices ensures that the original task-specific prompts are evaluated during the search.

Candidate Evaluation. For each candidate $\boldsymbol{\lambda}$, we construct $\mathbf{Z}_T(\boldsymbol{\lambda})$ and evaluate it using the task-specific metric on at most 500 validation examples with sampling temperature 0.1. Each candidate is evaluated over three independent inference runs, from which we record the empirical mean and variance. The empirical mean serves as the Bayesian optimization response, while the empirical variance estimates candidate-dependent observation noise.

Simplex-Aware Gaussian Process. We fit a Gaussian process (GP) to the validation score $\mathrm{Val}_j(\boldsymbol{\lambda})$. Before GP fitting, each simplex coordinate is transformed elementwise as

$$\widetilde{\boldsymbol{\lambda}} = \sqrt{\boldsymbol{\lambda}} = \left(\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_8}\right). \tag{43}$$

Since $\sum_j \lambda_j = 1$, the transformed coordinates satisfy $\|\widetilde{\boldsymbol{\lambda}}\|_2 = 1$, mapping the simplex to the positive orthant of the unit sphere. Euclidean distance between the square root coordinates is proportional to the Hellinger distance between the corresponding simplex coordinates.

We use a Constant Kernel multiplied by an RBF kernel:

$$K(\boldsymbol{\lambda}, \boldsymbol{\lambda}') = c \exp\left(-\frac{\left\|\sqrt{\boldsymbol{\lambda}} - \sqrt{\boldsymbol{\lambda}'}\right\|_2^2}{2r^2}\right), \tag{44}$$

where c and r denote the kernel amplitude and length scale, respectively. We initialize $c = 1.0$ and $r = 0.5$, with optimization ranges $c \in [0.1, 10]$ and $r \in [0.05, 5.0]$. Both hyperparameters are optimized by maximizing the GP log marginal likelihood. The empirical variance from repeated inference is provided to the GP as heteroscedastic observation noise.

Expected Improvement. At each Bayesian optimization iteration, we generate 4,096 candidate simplex coordinates. Half are sampled from a standard Dirichlet distribution for broad simplex coverage, one quarter from a sparse Dirichlet distribution to emphasize boundaries and vertices, and the remaining quarter around the current best coordinate for local refinement.

Let $m_j(\boldsymbol{\lambda})$ and $s_j(\boldsymbol{\lambda})$ denote the GP predictive mean and standard deviation for task $\mathcal{T}_j$, and let u_j^+ denote the best validation score observed so far. Candidates are ranked using Expected Improvement (Jones et al., 1998):

$$\mathrm{EI}_j(\boldsymbol{\lambda}) = \left(m_j(\boldsymbol{\lambda}) - u_j^+ - \xi\right) \Phi(z) + s_j(\boldsymbol{\lambda}) \phi(z), \tag{45}$$

$$z = \frac{m_j(\boldsymbol{\lambda}) - u_j^+ - \xi}{s_j(\boldsymbol{\lambda})}, \tag{46}$$

Table 7: Comparison of trainable components and parameter counts.

Method	Replay	Backbone	Soft Prompt	# Trainable Parameters (M)		
				Qwen3-1.7B	Qwen3-4B	Qwen3-8B
SFT	✗	✓	✗	1,720.6	4,022.5	8,190.7
Replay	✓	✓	✗	1,720.6	4,022.5	8,190.7
LwF	✗	✓	✗	1,720.6	4,022.5	8,190.7
OPR-RU	✓	✓	✗	1,720.6	4,022.5	8,190.7
OPR-SC	✓	✓	✗	1,720.6	4,022.5	8,190.7
ProgPrompt	✗	✗	✓	0.5	0.7	1.0
ProMoT	✗	✓	✓	1,721.1	4,023.1	8,191.7
LAPS (Ours)	✓	✓	✓	1,727.9	4,031.7	8,205.5

Table 8: Statistics of the eight continual adaptation tasks used in our experiments.

Dataset	Source	Average Length	Metric	Language	Size
C-STANCE	Social Media	109	Accuracy	Chinese	5,000
FOMC	Finance	76	Accuracy	English	5,000
MeetingBank	Meeting	3,550	ROUGE-L	English	5,000
Py150	GitHub	814	Edit Similarity	English	5,000
ScienceQA	Science	285	Accuracy	English	5,000
NumGLUE-cm	Math	50	Accuracy	English	5,000
NumGLUE-ds	Math	39	Accuracy	English	5,000
20Minuten	News	739	SARI	German	5,000

where $\Phi(\cdot)$ and $\phi(\cdot)$ denote the cumulative distribution function and probability density function of the standard Gaussian, respectively. We set $\xi = 0.05$ and $\mathrm{EI}_j(\boldsymbol{\lambda}) = 0$ whenever $s_j(\boldsymbol{\lambda}) = 0$.

We select at most four candidates per iteration. Within each batch, selected coordinates must satisfy a minimum pairwise distance to avoid spending evaluations on nearly identical locations. Each task is allocated at most 100 candidate evaluations, except MeetingBank, whose budget is 50. Search terminates early if the maximum EI remains below 0.005 for three consecutive iterations.

Final Candidate. After Bayesian optimization terminates, we retain the three coordinates with the highest observed validation scores. If the vertex corresponding to the target task is not among them, it is added as an additional candidate, yielding three or four finalists. Each finalist is reevaluated on the complete validation set using eight fresh inference runs. The coordinate with the highest mean validation score is selected as $\boldsymbol{\lambda}^{\star}_{j,T}$, and $\mathbf{Z}_T(\boldsymbol{\lambda}^{\star}_{j,T})$ is used as the final soft prompt for task $\mathcal{T}_j$.

C Method Characteristics and Parameter Efficiency

Table 7 compares the trainable components and parameter counts of different methods. Although LAPS jointly optimizes the backbone and soft prompts, it introduces only 7.3M, 9.2M, and 14.8M additional parameters for Qwen3-1.7B, Qwen3-4B, and Qwen3-8B, respectively. These additions account for only approximately 0.43%, 0.23%, and 0.18% of the corresponding backbone sizes.

D Dataset Details

We conduct experiments on the eight tasks from TRACE, covering diverse domains, languages, and evaluation metrics. Samples exceeding the maximum sequence length of 2,048 tokens are removed. Table 8 summarizes the statistics of each task. We use the official task-specific evaluation metrics from TRACE, including Accuracy, ROUGE-L, Edit Similarity, and SARI (Xu et al., 2016). Edit Similarity is computed from normalized Levenshtein distance (Levenshtein, 1966).

The benchmark contains both classification and generation tasks and spans Chinese, English, and German. This diversity allows us to evaluate continual adaptation under heterogeneous task distributions rather than within a single domain or output format.

Table 9: Task-specific simplex coordinates selected by Bayesian optimization.

Model	Target Task	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6	λ_7	λ_8
Qwen3-1.7B	C-STANCE	0.3313	0.2977	0.0001	0.0134	0.0646	0.2884	0.0000	0.0045
	FOMC	0.2173	0.3009	0.0164	0.0002	0.4052	0.0170	0.0000	0.0430
	MeetingBank	0.0000	0.0011	0.9967	0.0000	0.0011	0.0000	0.0003	0.0006
	Py150	0.0000	0.0000	0.0000	0.9962	0.0004	0.0014	0.0000	0.0015
	ScienceQA	0.0112	0.0000	0.0000	0.0000	0.9445	0.0012	0.0007	0.0424
	NumGLUE-cm	0.0004	0.0260	0.0016	0.0059	0.0087	0.8868	0.0636	0.0070
	NumGLUE-ds	0.0002	0.0042	0.0000	0.0168	0.0317	0.0000	0.9471	0.0001
	20Minuten	0.1403	0.2323	0.0019	0.0368	0.0010	0.1004	0.0705	0.4168
Qwen3-4B	C-STANCE	0.0767	0.0797	0.0957	0.4814	0.0292	0.0118	0.1497	0.0759
	FOMC	0.0000	0.9310	0.0004	0.0000	0.0001	0.0084	0.0575	0.0027
	MeetingBank	0.0249	0.0000	0.9634	0.0000	0.0000	0.0000	0.0032	0.0085
	Py150	0.0607	0.0001	0.0379	0.8538	0.0054	0.0408	0.0000	0.0012
	ScienceQA	0.0220	0.0000	0.0002	0.0048	0.9268	0.0079	0.0043	0.0339
	NumGLUE-cm	0.1174	0.2825	0.0670	0.0016	0.0086	0.5036	0.0063	0.0129
	NumGLUE-ds	0.0002	0.0345	0.0000	0.0160	0.0001	0.0025	0.9466	0.0002
	20Minuten	0.1589	0.2461	0.1227	0.0508	0.0356	0.0511	0.1885	0.1464
Qwen3-8B	C-STANCE	0.0660	0.2528	0.1278	0.0516	0.4995	0.0006	0.0010	0.0006
	FOMC	0.0002	0.9137	0.0004	0.0002	0.0000	0.0465	0.0000	0.0389
	MeetingBank	0.0003	0.0002	0.9884	0.0000	0.0002	0.0000	0.0080	0.0029
	Py150	0.0372	0.0000	0.0034	0.9253	0.0198	0.0085	0.0041	0.0017
	ScienceQA	0.0001	0.0005	0.0000	0.0323	0.9030	0.0639	0.0002	0.0000
	NumGLUE-cm	0.0830	0.0588	0.0211	0.1582	0.0070	0.3813	0.2833	0.0073
	NumGLUE-ds	0.0156	0.0054	0.0223	0.0002	0.0003	0.0001	0.9208	0.0353
	20Minuten	0.0009	0.0254	0.0000	0.0012	0.1002	0.0000	0.1368	0.7355

E Additional Experiments

This section provides complementary analyses of LAPS, including search-selected task coordinates and convergence, robustness to task order, continual adaptation dynamics, and the effect of learnable control prompts on space geometry. Introducing learnable control prompts reshapes this fixed interpolation geometry and yields larger cumulative gains from space search on both model scales, supporting their role in creating more useful intermediate prompts.

E.1 Task-Specific Coordinates Selected by BO

As shown in Table 9, we further report the simplex coordinates selected by Bayesian optimization for every task. The selected coordinates reveal two characteristic behaviors. For several tasks, including MeetingBank, Py150, ScienceQA, and NumGLUE-ds, search-selected coordinates remains close to the corresponding task vertex across model scales. In contrast, C-STANCE, NumGLUE-cm, and 20Minuten often select more distributed interior coordinates, indicating that their high-performing prompts draw more strongly on other task directions. The selected coordinates also vary substantially across model scales, supporting task- and model-specific space search rather than a single fixed combination.

E.2 Effect of Learnable Control Prompts on Space Geometry

To further examine the role of non-vertex control prompts, we compare the learned anchored space with and without them on Qwen3-1.7B and Qwen3-4B, as shown in Fig. 5. Without control prompts, the space reduces to the convex simplex determined entirely by the task-specific prompts. In this case, anchored prompt space search is restricted to the fixed interpolation geometry spanned by the task anchors.

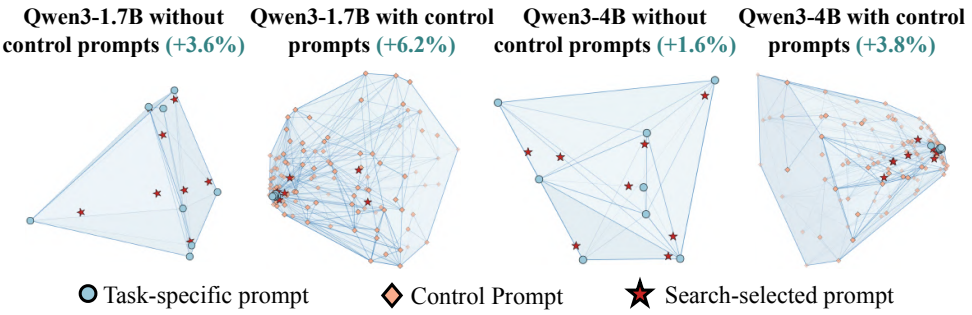


Figure 5: Visualization of the learned anchored prompt space with and without non-vertex control prompts on Qwen3-1.7B and Qwen3-4B. The values in green indicate the *cumulative performance improvement* over the eight tasks after space search.

Table 10: Continual learning performance under the reverse task order (20Minuten → NumGLUE-ds → NumGLUE-cm → ScienceQA → Py150 → MeetingBank → FOMC → C-STANCE).

Method	20Minuten	NG-ds	NG-cm	ScienceQA	Py150	MeetingBank	FOMC	C-STANCE	Average ↑	BWT ↑
Qwen3-1.7B										
SFT	39.1	**64.2**	54.2	18.2	58.6	43.8	52.3	50.8	47.6	−17.1
Replay	39.5	59.0	**59.3**	68.8	58.8	56.0	62.3	51.4	56.9	−7.7
ProgPrompt	38.5	49.2	37.7	70.9	50.5	47.7	60.6	50.1	50.6	**0.0**
ProMoT	39.7	61.4	30.6	75.6	59.4	**61.9**	**64.7**	**52.2**	55.7	−4.2
LAPS (Ours)	**39.8**	56.3	50.9	**76.5**	**60.3**	61.7	62.4	51.3	**57.4**	−2.7
Qwen3-4B										
SFT	38.6	71.2	65.7	9.8	63.1	61.4	53.9	56.3	52.5	−16.3
Replay	39.1	68.7	64.4	62.3	**64.3**	63.3	60.1	**56.8**	59.9	−7.7
ProgPrompt	39.4	52.3	56.5	**81.6**	57.4	49.4	61.4	54.8	56.6	**0.0**
ProMoT	40.2	73.2	**69.1**	40.7	63.6	**65.5**	67.4	55.3	59.4	−8.3
LAPS (Ours)	**40.6**	**74.6**	62.2	62.7	62.6	65.1	**71.7**	56.7	**62.0**	−5.7

E.3 Robustness to Task Order

We further evaluate the robustness of LAPS under the reverse TRACE task order. As shown in Table 10, reversing the task sequence changes the difficulty of individual tasks and leads to noticeable performance shifts across all methods. Nevertheless, LAPS achieves the highest Average performance on both model scales, reaching 57.4% on Qwen3-1.7B and 62.0% on Qwen3-4B. It also obtains BWT scores of -2.7% and -5.7%, respectively, substantially reducing forgetting compared with the parameter-updating baselines SFT, Replay, and ProMoT. Although ProgPrompt achieves a BWT of 0.0% by keeping the backbone and previously learned prompts fixed, its Average performance is considerably lower, at 50.6% on Qwen3-1.7B and 56.6% on Qwen3-4B. These results show that LAPS consistently provides a stronger balance between adaptation and retention across both model scales, indicating that its advantages are robust to changes in task order.

E.4 Convergence of Bayesian Optimization

Fig. 6 shows the convergence behavior of Bayesian optimization across the eight tasks and three model scales. In most cases, the best validation performance improves rapidly within the first few evaluations and then gradually stabilizes, indicating that effective regions of the learned anchored prompt space can be identified with a relatively small search budget.

E.5 Additional Continual Adaptation Dynamics

The main text reports task-wise continual adaptation trajectories on Qwen3-1.7B. Figures 7 and 8 extend this analysis to Qwen3-4B and Qwen3-8B. Across both model scales, methods that update model parameters continue to exhibit noticeable fluctuations on previously learned tasks, whereas LAPS generally maintains more stable historical performance while preserving strong adaptation to newly introduced tasks.

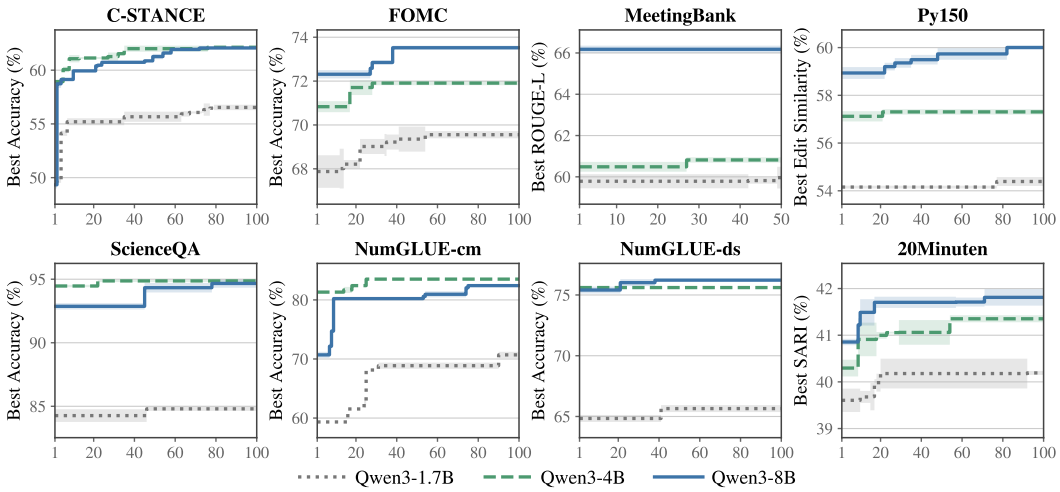


Figure 6: Bayesian optimization convergence across eight TRACE tasks and three Qwen3 model scales. Curves report the best validation performance observed so far over successive evaluations.

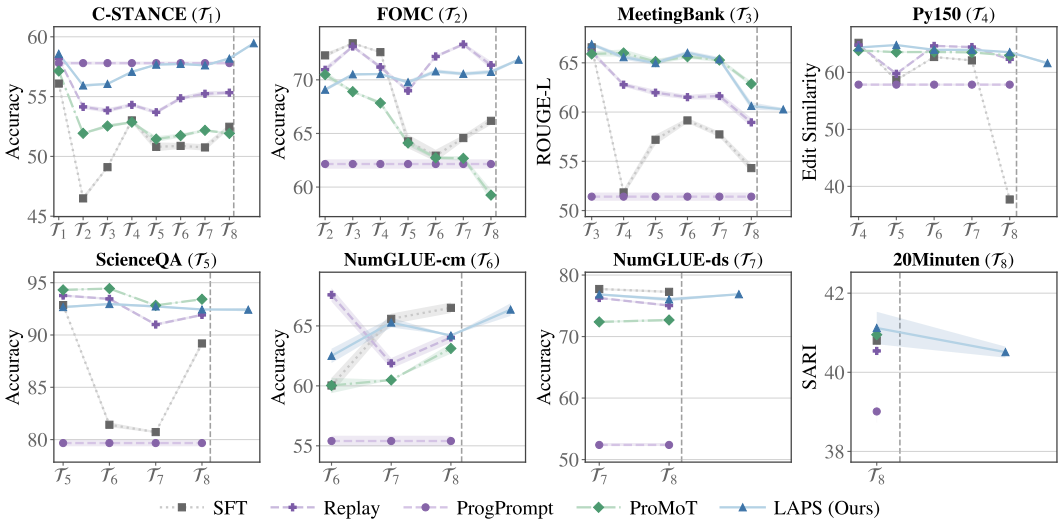


Figure 7: Task-wise continual adaptation trajectories on TRACE with Qwen3-4B.

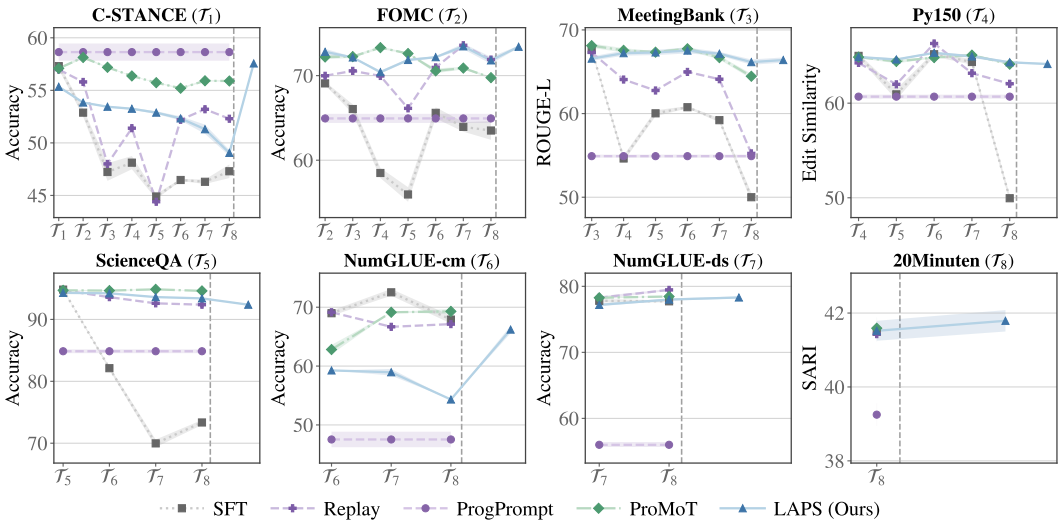


Figure 8: Task-wise continual adaptation trajectories on TRACE with Qwen3-8B.

E.6 Qualitative Analysis of Space Search

To further understand the effect of space search, we analyze qualitative examples on Qwen3-8B by comparing the outputs obtained from all eight task vertices with the search-selected prompt. The examples illustrate cases where searching the learned anchored prompt space yields behaviors that cannot be obtained from the task vertices alone. The MeetingBank example shows a different form of improvement. The original MeetingBank vertex already contains the relevant information, but substantially over-generates the transcript, resulting in a low ROUGE-L score. In contrast, the search-selected point remains close to the MeetingBank vertex while producing a much more concise summary, increasing the per-example ROUGE-L from 8.25 at the MeetingBank vertex to 45.28. This suggests that the anchored space learning can refine not only task correctness but also the form in which task knowledge is expressed.

Example for MeetingBank

Query: Write a summary of the meeting transcript concerning access to low cost veterinary care in Boston.

Reference: Order for a hearing regarding access to low cost veterinary care. Remains in the Committee on City Services and Innovation Technology.

Responses:

Prompt	Generated Summary	ROUGE-L
C-STANCE vertex	Councilor Braden proposed a hearing to address access to low cost veterinary care for residents in public and subsidized affordable housing. The response also discusses unequal access across neighborhoods and a One Health approach.	21.74
FOMC vertex	The transcript discusses difficulty accessing low cost veterinary care, particularly for residents in Brighton, and calls for collaboration between city agencies and veterinary organizations.	16.16
MeetingBank vertex	Order for a hearing regarding access to low cost veterinary care. The response then reproduces a long portion of the meeting discussion, including details on housing developments, veterinary providers, public health, and the One Health approach.	8.25
Py150 vertex	Councilor Braden called for a hearing on the difficulty faced by low income residents in accessing affordable veterinary care and proposed considering a holistic One Health approach.	20.95
ScienceQA vertex	The hearing will address the problem of low income Boston residents lacking access to affordable veterinary care and consider partnerships with veterinary services.	28.57
NumGLUE-cm vertex	A hearing was called to address difficulty accessing affordable veterinary care in Boston, especially in Brighton, and to consider a One Health approach linking human and animal health.	23.26
NumGLUE-ds vertex	The hearing addresses unequal access to low cost veterinary care for residents in public and subsidized housing and proposes a One Health framework for city services.	20.56
20Minuten vertex	A Boston City Council hearing was held on access to low cost veterinary care. The response emphasizes unequal access and the potential use of a One Health approach.	24.24
Search-selected	**Order for a hearing regarding Access to Low Cost Veterinary Care. The Chair recognized Councilor Breadon. Councilor Breadon offered the following: Order for a hearing regarding access to low cost veterinary care.**	**45.28**

BO coordinate:

$$\boldsymbol{\lambda}^{\star} = [0.0003,\ 0.0002,\ 0.9884,\ 0.0000,\ 0.0002,\ 0.0000,\ 0.0080,\ 0.0029].$$

Example for FOMC

Query: What is the monetary policy stance for the following text? A. dovish, B. hawkish, C. neutral.

Text: Of course, the basic problem in tackling the inflation of the 1970s is that we have one observation and many competing theories.

Reference: B (hawkish).

Predictions:

Prompt	Prediction	Correct
C-STANCE vertex	C (neutral)	No
FOMC vertex	C (neutral)	No
MeetingBank vertex	C (neutral)	No
Py150 vertex	C (neutral)	No
ScienceQA vertex	C (neutral)	No
NumGLUE-cm vertex	C (neutral)	No
NumGLUE-ds vertex	C (neutral)	No
20Minuten vertex	C (neutral)	No
Search-selected	**B (hawkish)**	**Yes**

BO coordinate:

$$\boldsymbol{\lambda}^{\star} = [0.0002,\ 0.9137,\ 0.0004,\ 0.0002,\ 0.0000,\ 0.0465,\ 0.0000,\ 0.0389].$$

For the FOMC task, all eight task vertices predict the FOMC example as neutral, whereas the search-selected interior point gives the correct hawkish prediction. Although the selected coordinate remains close to the FOMC vertex, its small contributions from other task directions are sufficient to induce a different model behavior. This illustrates that space search can identify useful prompts that are unavailable through vertex selection alone.