MENTOR-SDFT: MAKING EXPLICIT KNOWLEDGE TRANSFERABLE VIA ON-POLICY REFERENCES FOR CONTINUAL SELF-DISTILLATION FINE-TUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Continual fine-tuning requires large language models (LLMs) to acquire new capabilities while retaining existing knowledge. Self-Distillation Fine-Tuning (SDFT) offers a practical approach to this problem by using a reference-conditioned self-teacher to provide on-policy soft targets. However, we find that its supervision is often weak: reference conditioning alone does not guarantee that the self-teacher can transform the reference into transferable supervision, thereby undermining the plasticity–stability trade-off in continual learning. To address this limitation, we propose **MENTOR-SDFT** (**M**aking **E**xplicit k**N**owledge **T**ransferable via **O**n-policy **R**eferences for SDFT), a framework that strengthens self-teacher guidance by improving both supervision construction and utilization. MENTOR-SDFT introduces Agent-Guided Prompt Evolution to optimize teacher conditioning and Confidence-guided Soft Trajectory Distillation to adaptively regulate supervision according to teacher confidence and trajectory position. Experiments across continual-learning sequences and multiple LLM backbones show that MENTOR-SDFT improves final average accuracy and reduces forgetting compared with sequential SDFT baselines.

1 INTRODUCTION

In ever-changing environments, large language models (LLMs) are expected to continually acquire new knowledge while avoiding the need for costly retraining. Continual fine-tuning provides a practical approach to such adaptation by sequentially updating the model on newly available task data (Ke et al., 2023). However, in practice, continual fine-tuning is inevitably subject to the plasticity–stability dilemma (Kirkpatrick et al., 2017; Parisi et al., 2019): while sufficient plasticity is required to acquire new capabilities, sequential updates may overwrite prior knowledge and cause catastrophic forgetting (McCloskey & Cohen, 1989; French, 1999).

To improve this plasticity–stability trade-off, recent work has explored Self-Distillation Fine-Tuning (SDFT) (Shenfeld et al., 2026). In SDFT, the model itself, conditioned on the reference answer, acts as its own teacher and provides on-policy soft targets as the learning signal. Compared with supervised fine-tuning (SFT), which uses hard one-hot targets, the self-derived signals in SDFT are softer and closer to the model's own distribution, thereby reducing the risk of large gradients that destabilize continual updates (Shenfeld et al., 2026). SDFT also inherits the benefits of on-policy distillation (OPD) while requiring no external teacher, making it better suited to continual-learning settings where such a teacher may be unavailable or impractical (Agarwal et al., 2024).

Despite these advantages, we find that the supervision provided by SDFT remains weak. As shown in Figure 1, compared with OPD, SDFT attains lower post-distillation accuracy and yields a higher rate of erroneous predictions. This gap is intuitive. Firstly, although the self-teacher is conditioned on the reference answer, it is instantiated from the same underlying model and thus cannot provide supervision beyond the model's own knowledge; reference conditioning may sharpen existing behavior, but it cannot introduce genuinely new capabilities. Secondly, prior work suggests that reference-conditioned self-distillation often encourages over-reliance on the reference, suppressing exploratory and reasoning-related tokens and curtailing the model's own deliberation (Kim et al., 2026). These observations expose a central limitation of SDFT: reference conditioning alone does

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.

not guarantee strong supervision. It caps the teacher's effective capability at that of the student and can suppress the student's own reasoning, thereby undermining the plasticity–stability trade-off.

The above analysis suggests that *the effectiveness of SDFT hinges not on whether a reference is provided, but on whether the self-teacher can transform that reference into transferable supervision*. We contend that access to the reference answer alone is insufficient: it provides only a static target, whereas effective supervision requires modeling the reasoning process through which that target is obtained. Without such process-level understanding, the teacher can only reinforce superficial imitation of the answer, rather than the reasoning that would allow the student to internalize it. This gap motivates us to optimize the teacher's conditioning context, rather than merely providing the reference answer. To test this hypothesis, we conduct a controlled experiment in which the teacher receives not only the reference but also detailed, manually designed task-specific prompts that guide it to interpret the problem and reconstruct the derivation of the reference. Figure 1 shows that such prompts substantially improve SDFT performance, indicating that the quality of self-distillation depends on making the reference intelligible to the teacher, not merely accessible.

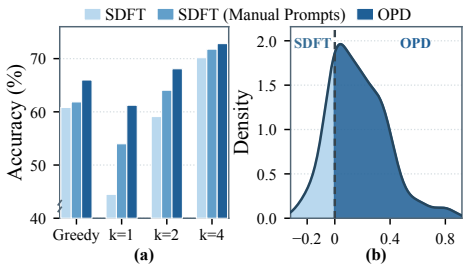


Figure 1: ToolUse performance. (a) Greedy accuracy and pass@k, $k \in \{1, 2, 4\}$, for SDFT, Manual-Prompt SDFT, and OPD. (b) Per-question success-rate gap density (OPD−SDFT); positive values favor OPD.

Motivated by these findings, we propose **MENTOR-SDFT** (**M**aking **E**xplicit k**N**owledge **T**ransferable via **O**n-policy **R**eferences for SDFT), a framework that transforms reference information into more transferable self-teacher supervision. MENTOR-SDFT employs an agent-based prompt evolution strategy to automatically construct task-specific prompts that condition the self-teacher more effectively during distillation, yielding more stable and informative supervision. We further introduce a token-level weighting scheme that modulates supervision according to the teacher's relative confidence and trajectory position, thereby reducing the influence of uncertain predictions before the final-answer region. Experiments on a four-task continual-learning sequence with Qwen2.5 and Qwen3 show that MENTOR-SDFT achieves the highest final average accuracy among the evaluated sequential methods. Our main contributions are summarized as follows:

- We identify a fundamental limitation of SDFT: reference conditioning alone is insufficient, as the self-teacher cannot provide supervision beyond the model's own knowledge and may suppress the student's reasoning, thereby undermining the plasticity–stability trade-off in continual learning.
- We propose **MENTOR-SDFT**, which realizes this insight through an agent-based prompt evolution strategy that automatically constructs task-specific prompts for teacher conditioning, and a token-level weighting scheme that modulates supervision according to teacher confidence and trajectory position.
- We demonstrate on a four-task continual-learning sequence using Qwen2.5 and Qwen3 that MENTOR-SDFT consistently improves over SDFT and achieves the highest final average accuracy among the evaluated sequential methods.

2 Related Work

Continual Fine-Tuning of Language Models. Continual fine-tuning learns new capabilities while retaining prior ones (Parisi et al., 2019; Wu et al., 2024; Shi et al., 2025). Replay-based methods retain prior examples to mitigate forgetting (Rebuffi et al., 2017); Continual-T0 (CT0) applies rehearsal to sequential instruction tuning (Scialom et al., 2022), and InsCL improves replay efficiency via instruction similarity and informativeness (Wang et al., 2024b). Yet storing and revisiting past data can be impractical under privacy, storage, or access constraints. Rehearsal-free alternatives constrain updates that interfere with existing capabilities: spurious-forgetting analysis shows early updates may disrupt prior-task alignment without erasing knowledge and proposes freezing lower layers (Zheng et al., 2025); O-LoRA (Wang et al., 2023) assigns task updates to orthogonal low-rank

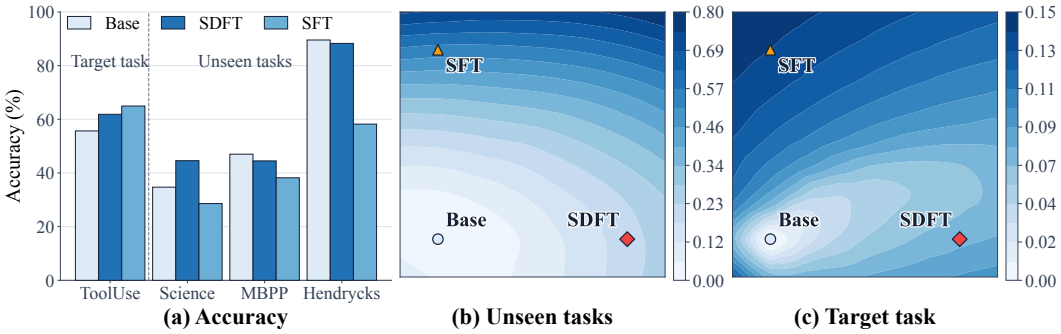


Figure 2: Task adaptation of Qwen3-4B via SFT and SDFT. (a) Accuracy before and after fine-tuning. (b–c) Token-level KL divergence from the base model on unseen tasks and the target task, respectively. Darker colors indicate larger divergence; color scales differ between panels.

subspaces; OSFT uses adaptive SVD to protect critical parameter directions (Nayak et al., 2026); MoCL composes task-specific modules for cross-task knowledge transfer (Wang et al., 2024a).

On-Policy Self-Distillation. On-policy distillation trains students on their own generated trajectories under teacher supervision, thereby reducing the training-inference mismatch (Agarwal et al., 2024; Gu et al., 2024). SDFT employs a demonstration-conditioned copy of the student model as the teacher to improve capability retention during sequential learning (Shenfeld et al., 2026); similarly, OPSD conditions a self-teacher on privileged solutions to provide token-level supervision for mathematical reasoning (Zhao et al., 2026). Recent work further examines the quality and reliability of such supervision: EGRSD modulates token-level updates using teacher entropy (Ke et al., 2026), and PW-OPSD treats trajectory position as a signal of teacher-token reliability (Liu et al., 2026). Nevertheless, privileged teacher context can suppress self-correction on long reasoning trajectories (Kaur et al., 2026), and dense supervision from self-distillation policy optimization may induce greater drift and forgetting than GRPO during continual post-training (Wang et al., 2026). These findings underscore the importance of both teacher quality and the selective use of its guidance.

3 Understanding SDFT for Continual Learning

3.1 Problem Setup

We consider rehearsal-free continual fine-tuning over a sequence of tasks $\{\mathcal{T}_k\}_{k=1}^{K}$ that arrive sequentially. Each task $\mathcal{T}_k$ provides a dataset $\mathcal{D}_k = \{(x_i^{(k)}, y_i^{(k)})\}_{i=1}^{N_k}$, where $x_i^{(k)}$ and $y_i^{(k)}$ denote the input and target response, respectively. Starting from an initial model π_{θ_0}, the model is updated from θ_{k-1} to θ_k using only the current-task data $\mathcal{D}_k$; previous-task training data $\{\mathcal{D}_j\}_{j<k}$ are unavailable. The goal is to obtain π_{θ_K} that performs well on all seen tasks without catastrophic forgetting.

3.2 SDFT is a Stable Continual Learner

Continual fine-tuning must reconcile two competing requirements: adapting to new tasks and preserving previously acquired knowledge. A key factor in this trade-off is how the training signal is constructed. In this section, we first contrast the optimization signals of SFT and SDFT, and then examine how this difference affects continual adaptation.

SFT learns from a fixed reference trajectory. Given an input x and answer y, its objective is

$$\mathcal{L}_{\mathrm{SFT}}(x, y) = -\sum_{t=1}^{|y|} \log \pi_\theta(y_t \mid x, y_{<t}), \tag{1}$$

whereas SDFT performs on-policy distillation from a reference-conditioned self-teacher:

$$\mathcal{L}_{\mathrm{SDFT}}(x, c, \hat{y}) = \sum_{t=1}^{|\hat{y}|} \mathrm{KL}(\pi_\theta(\cdot \mid x, \hat{y}_{<t}) \,\|\, \pi_\phi(\cdot \mid x, c, \hat{y}_{<t})), \tag{2}$$

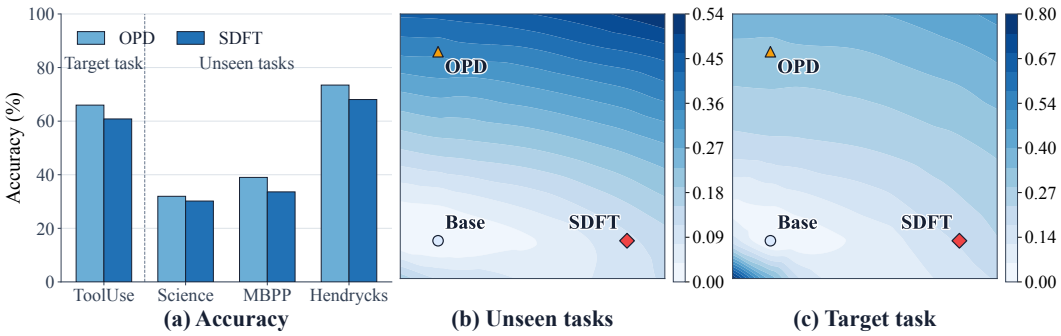


Figure 3: Comparison of external-teacher distillation and self-distillation. Both methods use Qwen2.5-3B as the student. OPD uses Qwen2.5-7B as an external teacher without access to reference answers, whereas SDFT uses a reference-conditioned Qwen2.5-3B self-teacher. (a) Student accuracy after distillation. (b,c) Token-level KL divergence relative to Base.

Table 1: Effect of teacher prompting on generation accuracy.

Teacher prompt	ToolUse	Science	MBPP	Hendrycks	Average
Original	53.73	81.30	62.83	90.65	72.13
Revised	**62.28**	**84.37**	**75.94**	**91.11**	**78.43**

where $\hat{y} \sim \pi_\theta(\cdot|x)$ represents the student-generated response, c denotes the additional reference information, and π_ϕ is the reference-conditioned copy of π_θ.

To isolate the effect of the supervision form, we compare SFT and SDFT under the same fine-tuning setup on ToolUse. Figure 2 summarizes the results. Figure 2(a) shows that SFT achieves higher target-task accuracy but causes larger degradation on unseen tasks, whereas SDFT better preserves previous capabilities while still adapting to the target task. The token-level KL divergence in Figure 2(b,c) provides a consistent picture: SFT induces larger distributional shifts from the base model, while SDFT yields more conservative updates on both target and unseen tasks.

The observed difference is reasonable, stemming from the distinct supervision signals used by the two methods. SFT uses a one-hot target that assigns all probability mass to the reference token, regardless of the student's current predictive distribution. Such a target drives the model toward an exact token-level match, producing large gradients whenever the reference token is not already dominant. In contrast, SDFT obtains supervision from a reference-conditioned copy of the model itself. These targets are soft and close to the student's predictive distribution. Since the self-teacher shares the student's parameters, the divergence between the distributions is substantially smaller, yielding updates that are less disruptive and more localized. Thus, SDFT refines the student's existing distribution rather than replacing it, rendering SDFT inherently stable for continual learning.

3.3 The Role of Self-Teacher Supervision in SDFT

Although SDFT yields more stable adaptation, its effectiveness hinges on the quality of supervision provided by the self-teacher. This section examines whether directly conditioning the model on the reference answer yields sufficiently informative supervision. Specifically, we compare SDFT with on-policy distillation (OPD), which distills directly from a stronger external teacher and serves as a performance upper bound:

$$\mathcal{L}_{\text{OPD}}(x, \hat{y}) = \sum_{t=1}^{|\hat{y}|} \text{KL}(\pi_\theta(\cdot \mid x, \hat{y}_{<t}) \,\|\, \pi_\psi(\cdot \mid x, \hat{y}_{<t})), \tag{3}$$

where π_ψ denotes the external teacher.

Figure 3 shows that OPD achieves stronger target-task adaptation while maintaining stability. As the two methods differ only in the teacher that supplies the distillation target, this gap isolates the

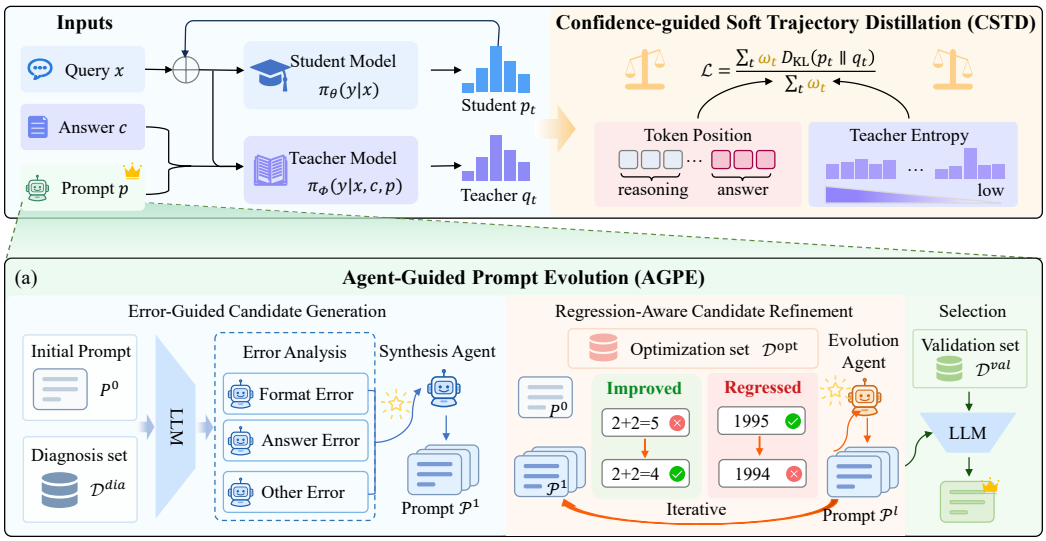


Figure 4: Overview of MENTOR-SDFT. The upper panel illustrates self-distillation with the evolved teacher prompt and confidence-guided token weighting. (a) AGPE evolves teacher prompts from student errors and refines them by balancing corrections against regressions, yielding the final prompt used for self-teacher conditioning.

effect of teacher quality. The weaker supervision in SDFT is therefore not surprising, as it does not rely on an external strong teacher. This, however, also reveals a central limitation for SDFT: *the effectiveness of SDFT hinges not on whether a reference is provided, but on whether the self-teacher can transform that reference into transferable supervision.*

To validate this claim, we conduct a controlled experiment in which the teacher parameters, task inputs, and privileged information are fixed, and only the teacher prompt is revised. The revised prompt is manually designed to provide the knowledge necessary for the teacher to interpret the reference answer. As shown in Table 1, it consistently improves teacher generation accuracy across all evaluated tasks, raising the average accuracy from 72.13% to 78.43%. These results validate our claim and establish a design principle for SDFT: the self-teacher must be conditioned to interpret the reference answer and convert it into transferable supervision.

4 Method

Section 3 shows that SDFT's effectiveness depends on whether the self-teacher transforms reference information into transferable supervision. Motivated by this, we propose **MENTOR-SDFT** (Figure 4), comprising two complementary components:

- **Agent-Guided Prompt Evolution (AGPE)** refines task-specific prompts to improve teacher conditioning, enabling the self-teacher to produce more informative supervision.
- **Confidence-Guided Soft Trajectory Distillation (CSTD)** adaptively modulates token-level distillation according to teacher confidence and trajectory position, allowing the student to better exploit the generated supervision.

4.1 Agent-Guided Prompt Evolution

Section 3.3 shows that manually designed prompts supply the necessary task knowledge, thereby enabling the self-teacher to interpret the reference answer. However, manually designing such prompts for each downstream task is laborious and costly. To address this, we introduce Agent-Guided Prompt Evolution (AGPE), which automatically constructs optimized prompts from each task training trajectory.

AGPE realizes this automatic evolution using three disjoint subsets of the current-task dataset $\mathcal{D}_k$: $\mathcal{D}_k^{\text{dia}}$ is used to diagnose errors under the initial prompt and generate candidates, $\mathcal{D}_k^{\text{opt}}$ to refine candidates by balancing corrections against regressions, and $\mathcal{D}_k^{\text{val}}$ to select the final prompt. Since AGPE is performed independently for each task, we omit the task index k in the remainder of this section.

Step 1. Error-Guided Candidate Generation. The first stage aims to produce a diverse set of candidate prompts that address the failure patterns of the initial prompt. Given an initial prompt $p^{(0)}$, which is typically the task-provided instruction (e.g., specifying a multiple-choice format), the frozen student generates responses on $\mathcal{D}^{\text{dia}}$ using task inputs alone. We collect the resulting reasoning traces and use an agent serving as an error diagnoser to group responses by failure pattern, while analysis agents identify potential improvements to the teacher conditioning. A synthesis agent then aggregates the resulting feedback $\mathcal{R}^{\text{dia}}$ and generates M candidate prompts:

$$\mathcal{P}^{(1)} = \left\{ p_m^{(1)} \right\}_{m=1}^{M} = \mathcal{A}_{\text{syn}} \left(p^{(0)}, \mathcal{R}^{\text{dia}} \right). \tag{4}$$

Step 2. Regression-Aware Candidate Refinement. We next evaluate each candidate prompt on $\mathcal{D}^{\text{opt}}$ by comparing its behavior with the initial prompt. Let $c_i(p) \in \{0, 1\}$ indicate whether the frozen student's response to example i is correct under prompt p. We define the corrected and regressed example sets as:

$$\begin{aligned} \mathcal{I}^{+}(p) &= \left\{ i : c_i(p^{(0)}) = 0,\ c_i(p) = 1 \right\}, \\ \mathcal{I}^{-}(p) &= \left\{ i : c_i(p^{(0)}) = 1,\ c_i(p) = 0 \right\}, \end{aligned} \tag{5}$$

where $\mathcal{I}^{+}(p)$ and $\mathcal{I}^{-}(p)$ represent newly corrected and degraded examples, respectively. Candidates are ranked according to their net gain:

$$G(p) = \left| \mathcal{I}^{+}(p) \right| - \left| \mathcal{I}^{-}(p) \right|. \tag{6}$$

This criterion balances newly corrected examples against newly regressed ones, helping prevent prompt evolution from improving individual examples at the expense of overall adaptation stability.

The refinement process is repeated for $L - 1$ rounds, yielding multiple generations of candidate prompts. At each round ℓ, a reflection agent analyzes how the top-B candidates alter predictions relative to the initial prompt, focusing on newly corrected and newly regressed cases, as well as those that remain consistently correct or incorrect. An evolution agent then uses the resulting feedback $\mathcal{R}^{(\ell)}$ to generate the next prompt population:

$$\mathcal{P}^{(\ell+1)} = \left\{ p_m^{(\ell+1)} \right\}_{m=1}^{M} = \mathcal{A}_{\text{evo}} \left(\text{TopB} \left(\mathcal{P}^{(\ell)}, G \right), \mathcal{R}^{(\ell)} \right). \tag{7}$$

Step 3. Validation-Based Prompt Selection. Finally, we select the prompt that provides the most reliable teacher conditioning on the validation subset:

$$p^{*} \in \arg \max_{p \in \{p^{(0)}\} \cup \bigcup_{\ell=1}^{L} \mathcal{P}^{(\ell)}} \text{Acc}^{\text{val}}(p), \tag{8}$$

where $\text{Acc}^{\text{val}}(p)$ measures the frozen model's accuracy using only the task input and prompt p. Reference answers are used only for evaluation. Including the original prompt $p^{(0)}$ prevents selection of a degraded conditioning strategy. The selected prompt p^{*} is then used to condition the self-teacher during SDFT.

4.2 Confidence-guided Soft Trajectory Distillation

AGPE improves teacher supervision by optimizing the conditioning teacher prompts. However, the effectiveness of teacher guidance also depends on how supervision signals are weighted during distillation. Prior work shows that weighted objectives improve distillation by emphasizing informative signals (Ke et al., 2026; Liu et al., 2026). Motivated by this, we propose CSTD, which dynamically adjusts distillation strength according to teacher uncertainty and trajectory position.

For a student response $\hat{y} \sim \pi_\theta(\cdot \mid x)$, let c denote the privileged teacher-side information, including the reference answer, and let p^* denote the task-specific teacher prompt selected by AGPE. Conditioned on both p^* and c along the student trajectory, the self-teacher produces the token-level distribution:

$$q_t(\cdot) = \pi_\phi(\cdot \mid x, p^*, c, \hat{y}_{<t}). \tag{9}$$

We quantify the relative uncertainty of teacher predictions along the trajectory using entropy:

$$H_t = -\sum_{v \in \mathcal{V}} q_t(v) \log q_t(v), \tag{10}$$

where $\mathcal{V}$ denotes the vocabulary. Normalizing entropy within each trajectory yields a relative confidence score:

$$\kappa_t = 1 - \frac{H_t - H_{\min}}{H_{\max} - H_{\min} + \epsilon}, \tag{11}$$

where $H_{\min} = \min_{1 \le j \le |\hat{y}|} H_j$, $H_{\max} = \max_{1 \le j \le |\hat{y}|} H_j$, and $\epsilon > 0$ ensures numerical stability. This confidence captures relative uncertainty variation within a trajectory rather than absolute prediction correctness.

Beyond uncertainty, the reliability of teacher supervision also varies across trajectory regions. Reasoning tokens admit diverse generation paths, so uncertain teacher signals should be down-weighted. In contrast, final-answer tokens directly determine task correctness and thus warrant stronger supervision. Let τ denote the start of the final-answer region, determined according to the task-specific output format. We define a position-aware gate:

$$r_t = \sigma\left(\frac{\tau - \frac{1}{2} - t}{\gamma}\right), \qquad \omega_t = r_t \kappa_t + (1 - r_t), \tag{12}$$

where $\gamma > 0$ controls the transition between trajectory regions. The resulting weight tracks relative confidence before the answer region and approaches one within it, so uncertainty-based modulation primarily affects reasoning tokens while preserving strong supervision for answer tokens.

The CSTD objective is defined as:

$$\mathcal{L}_{\mathrm{CSTD}}(x, p^*, c, \hat{y}) = \sum_{t=1}^{|\hat{y}|} \tilde{\omega}_t \, \mathrm{KL}(\pi_\theta(\cdot \mid x, \hat{y}_{<t}) \,\|\, \pi_\phi(\cdot \mid x, p^*, c, \hat{y}_{<t})), \tag{13}$$

where $\tilde{\omega}_t = \omega_t / \sum_{j=1}^{|\hat{y}|} \omega_j$ denotes the normalized supervision weight. Teacher distributions and weights are detached during student updates.

5 Experiments

5.1 Experimental Setup

Tasks and Protocol. We evaluate replay-free continual fine-tuning on four heterogeneous tasks: ToolUse, Science, MBPP, and Hendrycks. For AGPE, the current-task training set is partitioned into disjoint subsets for diagnosis, optimization, and validation, while test examples remain strictly excluded from prompt evolution.

Evaluation Metrics. Our primary metric is final average accuracy, which measures performance across all tasks after the full training sequence. We also report average forgetting to assess retention of previously learned tasks (Chaudhry et al., 2018):

$$\bar{A}_K = \frac{1}{K} \sum_{i=1}^{K} A_{K,i}, \qquad F = \frac{1}{K-1} \sum_{i=1}^{K-1} \left(\max_{i \le k \le K} A_{k,i} - A_{K,i} \right). \tag{14}$$

Here, K is the number of tasks and $A_{k,i}$ is the accuracy on task i after training through task k. Higher $\bar{A}_K$ and lower F indicate better overall performance.

Models and Baselines. We evaluate MENTOR-SDFT on Qwen2.5-3B (Hui et al., 2024), Qwen3-1.7B, and Qwen3-4B (Yang et al., 2025). We compare against replay-free sequential adaptation

Table 2: Performance across model backbones (%). Avg. is final average accuracy; Forgetting is average forgetting. SDFT is fine-tuned independently on each task. Bold indicates the best result among sequential methods for each metric and backbone.

Model	Method	ToolUse	Science	MBPP	Hendrycks	Avg. ↑	Forgetting ↓
Qwen2.5-3B	Base	32.99	30.37	37.80	73.46	43.66	–
	SDFT	60.82	66.47	41.00	78.61	61.73	–
	Sequential SFT	30.93	63.31	35.60	62.34	48.05	10.47
	Sequential SDFT	48.45	63.91	37.80	78.45	57.15	4.12
	SDFT+Layer Freezing	46.39	63.91	**41.20**	78.26	57.44	4.29
	SDFT+OSFT	50.52	65.09	36.00	**79.36**	57.74	4.23
	MENTOR-SDFT (Ours)	**64.95**	**66.07**	40.00	78.85	**62.46**	**3.33**
Qwen3-1.7B	Base	44.33	35.50	30.80	66.13	44.19	–
	SDFT	61.86	65.48	40.80	78.26	61.60	–
	Sequential SFT	52.58	48.52	34.20	76.92	53.06	**1.09**
	Sequential SDFT	52.58	62.52	36.00	**83.49**	58.65	1.61
	SDFT+Layer Freezing	49.48	64.30	37.80	82.14	58.43	1.46
	SDFT+OSFT	54.64	61.34	**39.00**	82.73	59.43	2.38
	MENTOR-SDFT (Ours)	**58.76**	**69.43**	37.80	83.40	**62.35**	1.33
Qwen3-4B	Base	55.67	34.71	47.00	89.55	56.73	–
	SDFT	62.89	63.12	48.00	92.00	66.50	–
	Sequential SFT	51.55	61.54	39.60	58.72	52.85	6.64
	Sequential SDFT	58.76	60.36	47.80	92.17	64.77	2.69
	SDFT+Layer Freezing	53.61	58.19	48.00	92.08	62.97	**1.18**
	SDFT+OSFT	56.70	61.34	**48.80**	**92.25**	64.77	4.09
	MENTOR-SDFT (Ours)	**62.89**	**65.48**	47.80	91.66	**66.96**	1.96

baselines. Sequential SFT and Sequential SDFT represent hard-target and self-distillation objectives, respectively. We further include SDFT+Layer Freezing (Zheng et al., 2025) and SDFT+OSFT (Nayak et al., 2026), which impose additional update constraints on sequential SDFT.

Implementation Details. Models are trained for four epochs per task using Adam (Kingma & Ba, 2014) on NVIDIA H100 GPUs. AGPE uses gpt-5.4-mini only for offline prompt evolution and never provides distillation targets. For each task, AGPE generates $M = 3$ candidate prompts per round over $L = 2$ evolution rounds. The selected prompt is used only to condition the self-teacher during training. For CSTD, we set $\gamma = 16$ and determine the final-answer boundary τ according to each dataset's output format.

5.2 Main Results

MENTOR-SDFT achieves the highest final average accuracy among all sequential methods across the three backbones (Table 2), improving over Sequential SDFT by 5.31, 3.70, and 2.19 on Qwen2.5-3B, Qwen3-1.7B, and Qwen3-4B, respectively. The independently fine-tuned SDFT results provide task-specific references without sequential interference, whereas all sequential methods are evaluated using a single model after the full task sequence. The consistent gains across backbones show that strengthening self-teacher supervision improves continual adaptation.

MENTOR-SDFT also reduces average forgetting relative to Sequential SDFT, from 4.12 to 3.33, 1.61 to 1.33, and 2.69 to 1.96 across the three backbones, respectively. Together, the improvements in final accuracy and retention indicate a more favorable balance between plasticity and stability during continual fine-tuning.

5.3 Ablation Studies

Table 3 evaluates the incremental contributions of AGPE and CSTD by progressively extending Sequential SDFT. Adding AGPE improves the average accuracy from 57.15% to 60.83% on Qwen2.5-3B and from 58.65% to 61.89% on Qwen3-1.7B, demonstrating that optimizing the teacher conditioning context provides more effective self-teacher supervision. Building upon AGPE, CSTD further increases the average accuracy to 62.46% and 62.35%, respectively, showing that adaptive

Table 3: Progressive ablation of AGPE and CSTD on two backbones. Bold indicates the best result.

Model	AGPE	CSTD	ToolUse	Science	MBPP	Hendrycks	Average
Qwen2.5-3B	–	–	48.45	63.91	37.80	78.45	57.15
	✓	–	58.76	**66.07**	39.80	78.69	60.83
	✓	✓	**64.95**	**66.07**	**40.00**	**78.85**	**62.46**
Qwen3-1.7B	–	–	52.58	62.52	36.00	83.49	58.65
	✓	–	57.73	68.84	36.40	**84.58**	61.89
	✓	✓	**58.76**	**69.43**	**37.80**	83.40	**62.35**

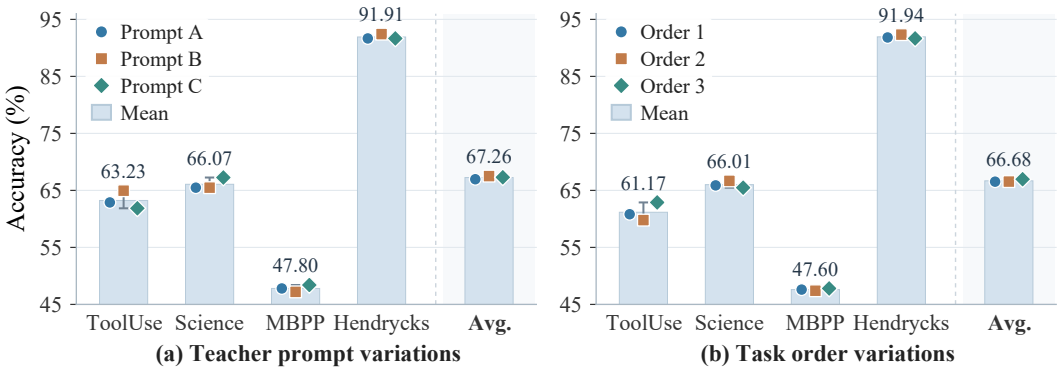


Figure 5: Sensitivity of MENTOR-SDFT to (a) teacher prompt variants and (b) task training orders.

modulation of teacher signals according to confidence and trajectory position provides additional benefits. These results verify the complementary roles of the two components: AGPE improves the quality of self-teacher supervision, while CSTD enhances its utilization during student optimization.

Robustness to Teacher Prompt Variations. We evaluate the robustness of MENTOR-SDFT to different teacher prompt formulations using three variants. As shown in Figure 5(a), the maximum difference in final average accuracy is only 0.55 across the three prompts. Although individual tasks exhibit minor variations, the overall performance remains highly consistent. These results suggest that MENTOR-SDFT does not rely on a specific teacher prompt formulation, but instead benefits from the proposed mechanism for improving self-teacher guidance.

Robustness to Task Ordering. We further evaluate MENTOR-SDFT under three different task training orders. As shown in Figure 5(b), the maximum difference in final average accuracy across the evaluated orders is only 0.43. While individual task performance varies due to differences in task exposure and subsequent updates, no single ordering consistently dominates across all tasks. The comparable aggregate performance across different orders indicates that MENTOR-SDFT remains robust under variations in task sequence.

6 Conclusion

In this paper, we presented MENTOR-SDFT, a framework that strengthens self-teacher guidance for continual fine-tuning by improving how the self-teacher generates and utilizes supervision signals. Through Agent-Guided Prompt Evolution and Confidence-guided Soft Trajectory Distillation, MENTOR-SDFT enhances teacher conditioning and adaptively regulates distillation according to supervision reliability, while preserving the stability advantages of SDFT. Extensive experiments across multiple continual-learning sequences and model backbones demonstrate that MENTOR-SDFT consistently improves final performance and reduces forgetting compared with sequential SDFT baselines. These results highlight that the effectiveness of self-distillation depends not only on providing reference information, but also on enabling the self-teacher to transform such information into more informative supervision.

AI USE STATEMENT

Generative AI tools were also used for limited assistance in improving the clarity and readability of the manuscript and for minor programming assistance during the development process. We did not use generative AI tools to generate the research methodology, experimental results, or scientific claims presented in this work. All AI-assisted suggestions were reviewed and verified by the authors. The authors take full responsibility for the final content of this paper.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we provide the complete source code as supplementary material, including implementations of AGPE and CSTD, training scripts, evaluation protocols, and hyperparameter settings. Detailed descriptions of the datasets, data splits, and model configurations are provided in Appendix F.These materials together are sufficient to reproduce the results reported in this work.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, volume 2024, pp. 21246–21263, 2024.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *European conference on computer vision*, pp. 556–572. Springer, 2018.

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

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

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models. In *International Conference on Learning Representations*, volume 2024, pp. 32694–32717, 2024.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

Simran Kaur, Narutatsu Ri, Yinghui He, Liam Fowl, and Sanjeev Arora. Rethinking on-policy self-distillation for thinking models. *arXiv preprint arXiv:2607.05184*, 2026.

Junlong Ke, Zichen Wen, Weijia Li, Conghui He, and Linfeng Zhang. Respecting self-uncertainty in on-policy self-distillation for efficient llm reasoning. *arXiv preprint arXiv:2605.13255*, 2026.

Zixuan Ke, Yijia Shao, Haowei Lin, Tatsuya Konishi, Gyuhak Kim, and Bing Liu. Continual pre-training of language models. *arXiv preprint arXiv:2302.03241*, 2023.

Jeonghye Kim, Xufang Luo, Minbeom Kim, Sangmook Lee, Dohyung Kim, Jiwon Jeon, Dongsheng Li, and Yuqing Yang. Why does self-distillation (sometimes) degrade the reasoning capability of llms? *arXiv preprint arXiv:2603.24472*, 2026.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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.

Xiaogeng Liu, Xinyan Wang, Yingzi Ma, Yechao Zhang, and Chaowei Xiao. When are teacher tokens reliable? position-weighted on-policy self-distillation for reasoning. *arXiv preprint arXiv:2605.21606*, 2026.

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. Elsevier, 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*, volume 2026, pp. 20212–20243, 2026.

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.

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

Thomas Scialom, Tuhin Chakrabarty, and Smaranda Muresan. Fine-tuned language models are continual learners. In *Proceedings of the 2022 conference on empirical methods in natural language processing*, pp. 6107–6122, 2022.

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

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.

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

Meng Wang, Haohan Zhao, Wenzhuo Liu, Lu Yang, Geng Liu, Haiyang Guo, Guo-Sen Xie, Gaofeng Meng, Hongbin Liu, and Fei Zhu. Denser $\neq$ better: Limits of on-policy self-distillation for continual post-training. *arXiv preprint arXiv:2607.01763*, 2026.

Mingyang Wang, Heike Adel, Lukas Lange, Jannik Strötgen, and Hinrich Schütze. Rehearsal-free modular and compositional continual learning for language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pp. 469–480, 2024a.

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

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

Tongtong Wu, Linhao Luo, Yuan-Fang Li, Shirui Pan, Thuy-Trang Vu, and Gholamreza Haffari. Continual learning for large language models: A survey. *arXiv preprint arXiv:2402.01364*, 2024. URL https://arxiv.org/abs/2402.01364.

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.

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. *arXiv preprint arXiv:2601.18734*, 2026.

Junhao Zheng, Xidi Cai, Shengjie Qiu, and Qianli Ma. Spurious forgetting in continual learning of language models. In *International Conference on Learning Representations*, volume 2025, pp. 67081–67146, 2025.

A Additional Experimental Results

Task acquisition and retention. Figure 6 compares task performance immediately after acquisition and at the end of the full training sequence, providing a direct view of both task learning and subsequent retention. MENTOR-SDFT maintains strong performance at both stages. On ToolUse, it achieves the highest accuracy upon acquisition and preserves most of this advantage after subsequent training, resulting in a noticeably smaller performance decline than Sequential SDFT. On Science, MENTOR-SDFT also attains the highest final accuracy, while remaining competitive on MBPP after the complete sequence. These results suggest that the gains of MENTOR-SDFT are not limited to stronger final-task performance, but arise from a better balance between acquiring new tasks and preserving previously learned capabilities throughout sequential adaptation.

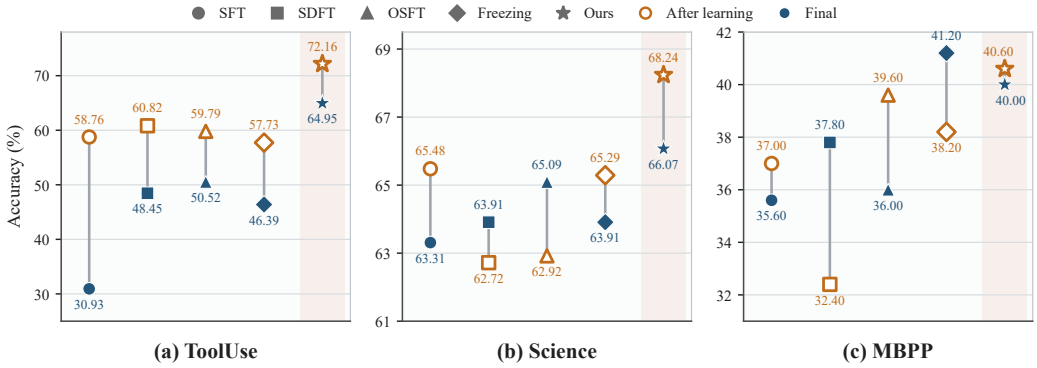


Figure 6: Task accuracy on Qwen2.5-3B measured immediately after each task is learned and again after completing the full four-task sequence. Open and filled markers indicate post-acquisition and final performance, respectively. Hendrycks is omitted because it is the final task and is therefore not followed by subsequent training.

Continual Training: SFT vs. SDFT. We further compare SFT and SDFT after sequential training on all four tasks using Qwen3-4B. As shown in Figure 7, the SDFT checkpoints exhibit consistently smaller distributional shifts from the base model than their SFT counterparts, whether KL divergence is measured across all tasks or on the final task, Hendrycks. Importantly, this more conservative distributional change is accompanied by stronger overall performance: SDFT achieves substantially higher accuracy on Hendrycks and a higher final average accuracy, while also exhibiting lower average forgetting (Table 2). Although SFT retains a slight advantage on Science, the overall results are consistent with the single-task analysis, showing that SDFT provides a more favorable stability–plasticity trade-off throughout the evaluated continual-learning sequence.

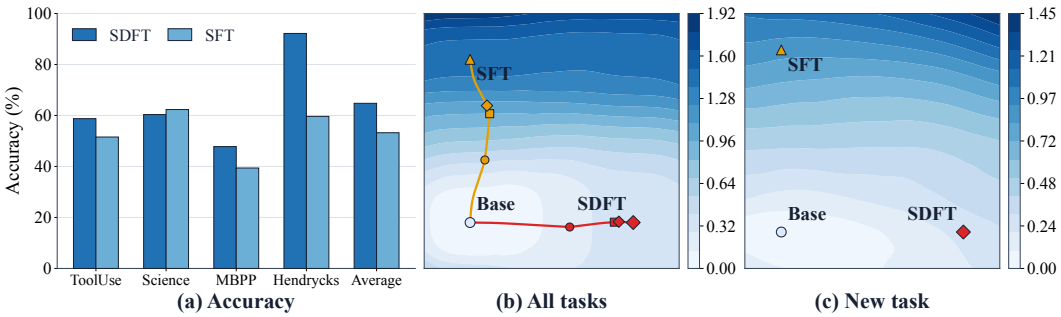


Figure 7: Sequential SFT and SDFT on Qwen3-4B after continual training. (a) Task accuracy after completing the full four-task sequence. (b,c) Token-level KL divergence from the base model, evaluated across all tasks and on the final task, hendrycks, respectively. Markers in (b) denote successive training stages, and darker regions indicate larger distributional shifts. The two panels use different color scales.

Numerical Projection in the KL Landscape. Figure 8 visualizes the mean token-level KL divergence from Base across the four tasks in a two-dimensional landscape constructed from the SDFT and SFT axes. MENTOR-SDFT and OPD are placed in this landscape through numerical projection of their corresponding measurements, rather than through projection of model parameters or update vectors. The resulting geometry reveals distinct behaviors across different training methods. SFT lies in a substantially higher-KL region, consistent with the larger distributional shift observed in our earlier analysis, whereas SDFT remains close to the low-KL region around Base. MENTOR-SDFT occupies an intermediate region between SDFT and OPD: relative to standard SDFT, its projection shifts toward the numerical profile associated with the stronger external-teacher supervision of OPD, while remaining far from the high-KL region occupied by SFT. The contour values characterize only the constructed SDFT–SFT landscape and should not be interpreted as the actual KL divergence of MENTOR-SDFT or OPD.

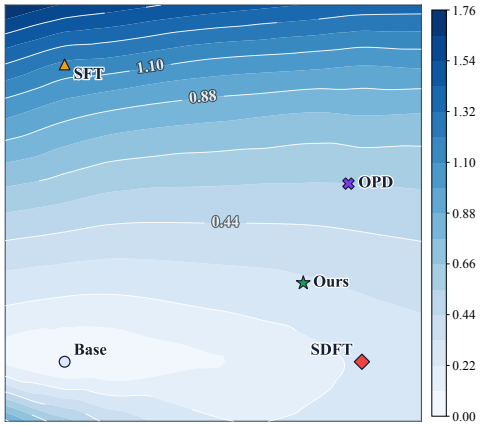


Figure 8: Four-task KL landscape defined by the SDFT and SFT axes. Contours show mean token-level KL divergence from Base; markers denote the evaluated methods.

B Training Details

Teacher Initialization and Updates. For each task $\mathcal{T}_k$, the student is initialized from θ_{k-1}, with θ_0 denoting the pretrained model for the first task. The self-teacher is initialized from the same parameters but does not receive gradient updates. Following SDFT, its parameters are updated after each optimization step using exponential moving average (EMA) of the student parameters with the same coefficient as the SDFT baseline. Teacher distributions and the token-level weights produced by CSTD are detached when computing the student objective.

Prompt Usage. AGPE is performed before distillation on each task, with model parameters fixed throughout prompt evolution. The selected task-specific prompt is used exclusively to condition the self-teacher during training, while the student retains the default task input. Neither the evolved teacher prompts nor the reference answers are required during evaluation or inference.

C ALGORITHMS

Algorithm 1 summarizes the continual training procedure of MENTOR-SDFT. For each task $\mathcal{T}_k$, AGPE first evolves and selects a task-specific teacher prompt while keeping the model parameters fixed. The student then generates on-policy trajectories from the default task input, while the EMA self-teacher is additionally conditioned on the selected prompt and privileged reference information. CSTD modulates the resulting token-level supervision according to teacher confidence and trajectory position. After completing each task, the updated student parameters are carried forward to the next task.

Algorithm 1 Continual Training with MENTOR-SDFT

Require: Initial model π_{θ_0}; task datasets $\{\mathcal{D}_k\}_{k=1}^{K}$; initial teacher prompts $\{p_k^{(0)}\}_{k=1}^{K}$; AGPE parameters M, B, L; CSTD parameters γ, ϵ; EMA coefficient μ

Ensure: Continually fine-tuned model π_{θ_K}

$\theta \leftarrow \theta_0$
for $k = 1, \ldots, K$ **do**
  Partition $\mathcal{D}_k$ into $\mathcal{D}_k^{\mathrm{dia}}$, $\mathcal{D}_k^{\mathrm{opt}}$, and $\mathcal{D}_k^{\mathrm{val}}$
  $p_k^{*} \leftarrow \mathrm{AGPE}(\pi_\theta, p_k^{(0)}, \mathcal{D}_k^{\mathrm{dia}}, \mathcal{D}_k^{\mathrm{opt}}, \mathcal{D}_k^{\mathrm{val}}, M, B, L)$
  Initialize the EMA self-teacher: $\phi \leftarrow \theta$
  for each training epoch **do**
    for each mini-batch $\mathcal{B} \subset \mathcal{D}_k$ **do**
      $\mathcal{L}_{\mathcal{B}} \leftarrow 0$
      for each $(x, y) \in \mathcal{B}$ **do**
        Construct privileged information c including the reference answer y
        Sample an on-policy trajectory $\hat{y} \sim \pi_\theta(\cdot \mid x)$
        Determine the final-answer boundary τ from the task-specific output format
        for $t = 1, \ldots, |\hat{y}|$ **do**
          $q_t(\cdot) \leftarrow \pi_\phi(\cdot \mid x, p_k^{*}, c, \hat{y}_{<t})$
          $H_t \leftarrow -\sum_{v \in \mathcal{V}} q_t(v) \log q_t(v)$
        end for
        $H_{\min} \leftarrow \min_t H_t$; $H_{\max} \leftarrow \max_t H_t$
        for $t = 1, \ldots, |\hat{y}|$ **do**
          $\kappa_t \leftarrow 1 - \dfrac{H_t - H_{\min}}{H_{\max} - H_{\min} + \epsilon}$
          $r_t \leftarrow \sigma\left(\dfrac{\tau - \frac{1}{2} - t}{\gamma}\right)$
          $\omega_t \leftarrow r_t \kappa_t + (1 - r_t)$
        end for
        $\tilde{\omega}_t \leftarrow \dfrac{\omega_t}{\sum_{j=1}^{|\hat{y}|} \omega_j}, \quad t = 1, \ldots, |\hat{y}|$
        Detach $\{q_t, \tilde{\omega}_t\}_{t=1}^{|\hat{y}|}$
        $\mathcal{L}_{\mathcal{B}} \leftarrow \mathcal{L}_{\mathcal{B}} + \sum_{t=1}^{|\hat{y}|} \tilde{\omega}_t \mathrm{KL}(\pi_\theta(\cdot \mid x, \hat{y}_{<t}) \,\|\, q_t)$
      end for
      Update θ with Adam using $\nabla_\theta\big(\mathcal{L}_{\mathcal{B}}/|\mathcal{B}|\big)$
      Update the EMA self-teacher: $\phi \leftarrow \mu\phi + (1 - \mu)\theta$
    end for
  end for
  $\theta_k \leftarrow \theta$
end for
return π_{θ_K}

C.1 AGENT-GUIDED PROMPT EVOLUTION

Algorithm 2 summarizes AGPE. With model parameters fixed, AGPE evolves task-specific teacher prompts through three stages: error-guided candidate generation, regression-aware refinement, and

Algorithm 2 Agent-Guided Prompt Evolution

Require: Current model π_θ; initial teacher prompt $p_k^{(0)}$; diagnosis set $\mathcal{D}_k^{\mathrm{dia}}$; optimization set $\mathcal{D}_k^{\mathrm{opt}}$; validation set $\mathcal{D}_k^{\mathrm{val}}$; candidates per generation M; retained candidates B; number of generations L

Ensure: Selected teacher prompt p_k^*

Freeze model parameters θ

Stage 1: Error-Guided Candidate Generation

$\mathcal{E}_k \leftarrow \textsc{CollectErrors}(\pi_\theta, p_k^{(0)}, \mathcal{D}_k^{\mathrm{dia}})$

$\mathcal{G}_k \leftarrow \textsc{ErrorDiagnoser}(\mathcal{E}_k)$

$\mathcal{R}_k^{\mathrm{dia}} \leftarrow \textsc{AnalysisAgents}(\mathcal{G}_k)$

$\mathcal{P}_k^{(1)} \leftarrow \textsc{SynthesisAgent}(p_k^{(0)}, \mathcal{R}_k^{\mathrm{dia}}, M)$

Stage 2: Regression-Aware Candidate Refinement

$\{c_i(p_k^{(0)})\}_i \leftarrow \textsc{EvaluatePrompt}(\pi_\theta, p_k^{(0)}, \mathcal{D}_k^{\mathrm{opt}})$

for $\ell = 1, \ldots, L$ **do**

  for all $p \in \mathcal{P}_k^{(\ell)}$ **do**

    $\{c_i(p)\}_i \leftarrow \textsc{EvaluatePrompt}(\pi_\theta, p, \mathcal{D}_k^{\mathrm{opt}})$

    $\mathcal{I}_k^+(p) \leftarrow \{i : c_i(p_k^{(0)}) = 0,\ c_i(p) = 1\}$

    $\mathcal{I}_k^-(p) \leftarrow \{i : c_i(p_k^{(0)}) = 1,\ c_i(p) = 0\}$

    $G_k(p) \leftarrow |\mathcal{I}_k^+(p)| - |\mathcal{I}_k^-(p)|$

  end for

  if $\ell < L$ **then**

    $\mathcal{S}_k^{(\ell)} \leftarrow \mathrm{TopB}\big(\mathcal{P}_k^{(\ell)}, G_k\big)$

    for all $p \in \mathcal{S}_k^{(\ell)}$ **do**

      $\mathcal{R}_k^{(\ell)}(p) \leftarrow \textsc{ReflectionAgent}(p, \{c_i(p_k^{(0)})\}_i, \{c_i(p)\}_i, \mathcal{D}_k^{\mathrm{opt}})$

    end for

    $\mathcal{R}_k^{(\ell)} \leftarrow \{\mathcal{R}_k^{(\ell)}(p) \mid p \in \mathcal{S}_k^{(\ell)}\}$

    $\mathcal{P}_k^{(\ell+1)} \leftarrow \textsc{EvolutionAgent}(\mathcal{S}_k^{(\ell)}, \mathcal{R}_k^{(\ell)}, M)$

  end if

end for

Stage 3: Validation-Based Prompt Selection

$\mathcal{C}_k \leftarrow \{p_k^{(0)}\} \cup \bigcup_{\ell=1}^{L} \mathcal{P}_k^{(\ell)}$

for all $p \in \mathcal{C}_k$ **do**

  $\mathrm{Acc}^{\mathrm{val}}(p) \leftarrow \textsc{PromptAccuracy}(\pi_\theta, p, \mathcal{D}_k^{\mathrm{val}})$

end for

$p_k^* \in \arg\max_{p \in \mathcal{C}_k} \mathrm{Acc}^{\mathrm{val}}(p)$

return p_k^*

validation-based selection. Candidate prompts are evaluated by their net gain over the initial prompt, defined as the number of corrected examples minus the number of regressed examples. The final prompt p_k^* is selected using held-out validation accuracy and is used only to condition the self-teacher during distillation. Here, $c_i(p) \in \{0, 1\}$ indicates whether example i is answered correctly under prompt p.

D Datasets and Evaluation Details

Tool Use. Following the experimental setup of SDFT (Shenfeld et al., 2026), we used the ToolAlpaca dataset (Tang et al., 2023) with its original train–test split. We adopted the expert demonstrations used in SDFT, which originate from the ToolAlpaca dataset. Accuracy was evaluated by regex matching the predicted API call against the ground-truth call while allowing for variations in argument ordering.

Science. Following SDFT (Shenfeld et al., 2026), we used the Chemistry L-3 subset of SciKnowEval (Feng et al., 2024) with an approximate 75%/5%/20% train/validation/test split. We adopted the expert demonstrations used in SDFT. As reported by Shenfeld et al. (2026), these demonstrations were constructed by sampling up to eight GPT-4o responses per prompt and retaining one response whose final answer matched the ground truth; this procedure produced a valid demonstration for every training example. Because Science is a multiple-choice task, accuracy was computed by exact matching of the model's extracted answer choice against the ground-truth option.

MBPP. We used the MBPP dataset (Austin et al., 2021), following the original train-test split provided by the authors. Reference Python solutions were included in the original dataset and used as expert demonstrations. Accuracy was evaluated by executing the generated code against the provided test cases, with a solution considered correct only if all test cases passed.

Hendrycks. We used the algebra subset of the MATH dataset (Hendrycks et al., 2021), following its original train–test split. The step-by-step reference solutions distributed with the dataset were used as expert demonstrations. Accuracy was evaluated by extracting the final answer from each model response and matching it against the ground-truth answer extracted from the corresponding reference solution.

Evaluation Protocol. We assess TOOLUSE by the selected tool and its arguments, SCIENCE by the extracted answer option, MBPP by whether the generated code passes all tests, and MATH (Algebra) by the extracted final answer. Invalid or unparsable outputs count as incorrect. Unless otherwise stated, all methods use the same evaluation prompts and decoding settings.

E PROMPT EVOLUTION EXAMPLES

This section presents the original and evolved teacher prompts for each task. These examples illustrate how AGPE adapts the teacher-side prompt to different task-specific training contexts.

E.1 TOOLUSE

Original prompt.

```
Your task is to answer the user's question using available
tools.

You have access to the following tools:

Use the following format:

Thought: you should always think about what to do
Action: the action to take, should be one of the tool names.
Action Input: the input to the action, must be in JSON format.
All of the action input must be realistic and from the user.
```

Evolved prompt.

```
You are a tool-using assistant. Your primary objective is to
match the reference tool actions and Action Input JSON exactly
.

Operating rules:
- Identify the minimal exact tool-call sequence before any
Action.
- Use only the required tool names; do not invent, substitute,
 or generalize tools.
```

```
- Copy every concrete parameter value exactly from the user
request or prior tool output.
- Do not normalize, abbreviate, convert, default, guess, or
placeholder any value unless the tool documentation explicitly
 requires it.
- When a tool is chosen, include only the documented fields
required for that call, with exact field names and exact value
 conventions from the tool description.
- Do not add extra fields, convenience fields, or speculative
follow-up calls.
- If a required value cannot be recovered exactly from the
user or previous tool output, ask for clarification instead of
 guessing.

Positive examples to preserve:
1) Action: sendHttpRequest
   Action Input: {"method":"POST","url":"https://httpbin.org/
post","headers":{"Content-Type":"application/json"},"data":{"
name":"John Doe","email":"john.doe@example.com"}}
2) Action: getGeographicData
   Action Input: {"location":"Rio de Janeiro"}
```

E.2 SCIENCE

Original prompt.

```
Given a question and four options, please select the right
answer.

Respond in the following format:
<reasoning>
...
</reasoning>
<answer>
...
</answer>

For the answer, only output the letter corresponding to the
correct
option (A, B, C, or D), and nothing else. Do not restate the
answer
text.

For example, if the answer is "A", just output:
<answer>
A
</answer>
```

Evolved prompt.

```
You are given a multiple-choice question with four options.

Reason carefully, then give your final choice in one terminal
<answer> block. The evaluator extracts only the text inside
that final block and compares it by exact string match, so the
```

```
 block must contain exactly one option letter and nothing else
.

Required output:
<answer>
A
</answer>

Inside <answer>...</answer>, include only one of A, B, C, or D
. Do not add any other text, punctuation, symbols, or line
contents inside the answer block. Do not place any text after
</answer>. You may include reasoning before the final answer
block, but the final answer must be a standalone terminal <
answer> block.
```

E.3 MBPP

Original prompt.

```
You are solving a Python programming problem. Return a correct
,
self-contained Python solution.

Respond in the following format:
<answer>
```python
...
```
</answer>

Put only valid Python code inside the code block in <answer>.
Do not include Markdown or explanations outside the required
tags.
```

Evolved prompt.

```
You are solving a Python programming problem for an evaluator
that extracts Python from the final answer and runs tests.

If the prompt is an executable coding task, return exactly
this wrapper and nothing else:
<answer>
```python
# complete Python program
```
</answer>

The code block must contain only valid, executable Python code
. Do not include explanations, prose, markdown, or any non-
code text inside the code block. Do not place any text outside
 the wrapper.

If the prompt explicitly specifies a function name, parameters
, or return behavior, match them exactly; otherwise, implement
 only what is stated and avoid guessing unstated behavior.
```

```
The solution must be self-contained, use only the Python
standard library, and run directly.

If the task has clear semantics, handle common edge cases and
include needed imports. If the prompt is ambiguous, stay as
close as possible to the stated requirement and do not over-
infer missing behavior.

Keep the final answer limited to the wrapper and the code
block.
```

E.4 HENDRYCKS

Original prompt.

```
Solve the following algebra problem.

Respond in the following format:
<reasoning>
...
</reasoning>
<answer>
...
</answer>

Put the solution inside <reasoning>.
Put only the final answer inside <answer>.
```

Evolved prompt.

```
Solve the problem. If you include reasoning, keep it brief and
 place it outside the final answer. Then provide the final
answer inside exactly one <answer>...</answer> block. The <
answer> block must contain only the final result, with no
explanation, wrappers, boxing, or extra text after the closing
 tag. Use canonical notation only when it is clearly needed
for notation-sensitive answers.
```

F IMPLEMENTATION DETAILS AND HYPERPARAMETERS

Continual-Learning Setup. We perform continual fine-tuning over four tasks in the fixed order: ToolUse, Science, MBPP, and Hendrycks. For each model backbone, the student at task $\mathcal{T}_k$ is initialized from the student obtained after the previous task, while the self-teacher is initialized from the same parameters and subsequently updated by exponential moving average (EMA). Only current-task data are used at each stage. Unless otherwise specified, training uses `bfloat16` precision, with vLLM for completion generation. The maximum prompt and completion lengths are both 1,024 tokens.

Optimization. Models are trained for four epochs per task using Adam with a cosine learning-rate schedule, a warmup ratio of 0.1, and a maximum gradient norm of 1.0. The per-device batch size is 1 with 32 gradient-accumulation steps, yielding an effective batch size of 32. We use a learning rate of 5×10^{-5} for ToolUse and Science and 2×10^{-5} for MBPP and Hendrycks. All data shuffling and subset construction use random seed 42. The common training settings are summarized in Table 4.

EMA Self-Teacher. At the beginning of each task, the self-teacher is initialized from the current student parameters. Only the student receives gradient updates. After each optimization step, the

Table 4: Common training hyperparameters.

Hyperparameter	Value
Numerical precision	`bfloat16`
Optimizer	Adam
Learning-rate scheduler	Cosine
Warmup ratio	0.1
Per-device batch size	1
Gradient accumulation	32
Effective batch size	32
Training epochs per task	4
Learning rate (ToolUse, Science)	5×10^{-5}
Learning rate (MBPP, Hendrycks)	2×10^{-5}
Maximum prompt length	1,024
Maximum completion length	1,024
Maximum gradient norm	1.0
Random seed	42

Table 5: Data used for AGPE prompt evolution. All subsets are mutually disjoint.

Task	$\lvert\mathcal{D}_k^{\mathrm{dia}}\rvert$	$\lvert\mathcal{D}_k^{\mathrm{opt}}\rvert$	$\lvert\mathcal{D}_k^{\mathrm{val}}\rvert$
ToolUse	512	512	512
Science	512	512	512
MBPP	200	74	100
Hendrycks	256	512	512

teacher parameters ϕ are updated from the student parameters θ using

$$\phi \leftarrow \mu\phi + (1-\mu)\theta, \tag{15}$$

where the EMA decay is $\mu = 0.99$. Teacher distributions are detached when computing the student objective.

AGPE Configuration. AGPE is performed before distillation for each task while model parameters remain fixed. The current-task data used for prompt evolution are divided into three disjoint subsets: $\mathcal{D}_k^{\mathrm{dia}}$ for error diagnosis and candidate generation, $\mathcal{D}_k^{\mathrm{opt}}$ for regression-aware refinement, and $\mathcal{D}_k^{\mathrm{val}}$ for final prompt selection. Test examples are never used during prompt evolution.

We use GPT-5.4-mini only for offline prompt evolution. AGPE generates $M = 3$ candidate prompts per generation, retains up to $B = 3$ candidates for reflection, and performs $L = 2$ generations. Prompt generation uses temperature 0.0, top-p of 0.95, and a maximum generation length of 2,048 tokens. The selected prompt p_k^* is used exclusively to condition the self-teacher during subsequent distillation; the student input remains unchanged.

CSTD Configuration. For CSTD, the self-teacher is conditioned on the evolved prompt p_k^* and the privileged reference information, while the student follows its on-policy trajectory. We set the position-transition parameter to $\gamma = 16$ and determine the final-answer boundary τ according to the task-specific output format. The resulting confidence- and position-aware weights are detached when computing the weighted reverse-KL objective.