TMS: Trajectory-Mixed Supervision for Reward-Free, On-Policy SFT

Anonymous Authors[1]

Abstract

Reinforcement Learning (RL) and Supervised Fine-Tuning (SFT) are the two dominant paradigms for enhancing Large Language Model (LLM) performance on downstream tasks. While RL generally preserves broader model capabilities (retention) better than SFT, it comes with significant costs: complex reward engineering, instability, and expensive on-policy sampling. In contrast, SFT is efficient but brittle, often suffering from catastrophic forgetting due to **Supervision Mismatch**: the divergence between the model's evolving policy and static training labels. We address this trade-off with **Trajectory-Mixed Supervision (TMS)**, a reward-free framework that approximates the on-policy benefits of RL by creating a dynamic curriculum from the model's own historical checkpoints. TMS minimizes *Policy-Label Divergence (PLD)*, preventing the mode collapse that drives forgetting in standard SFT. Experiments across reasoning (MATH, GSM8K) and instruction-following benchmarks demonstrate that TMS effectively shifts the accuracy–retention Pareto frontier. While RL remains the gold standard for retention, TMS significantly outperforms standard and iterative SFT, bridging the gap to RL without requiring reward models or verifiers. Mechanistic analysis confirms that PLD drift accurately predicts forgetting, and that TMS successfully mitigates this drift.

1. Introduction

The post-training of large language models (LLMs) has largely converged to a simple recipe: Supervised Fine-Tuning (SFT) on curated demonstrations, followed by preference or RL-based alignment when higher reliability is required (Ouyang et al., 2022; Christiano et al., 2023). SFT remains the default in practice due to its simplicity, stability, and efficiency in teaching task formats and high-quality behaviors. However, mounting evidence suggests that SFT often comes with a severe side effect: capability collapse or catastrophic forgetting (Kotha et al., 2024; Zhang et al., 2025). While improving performance on the target downstream distribution, SFT-tuned models can regress in general reasoning, linguistic diversity, and even pre-existing safety guardrails (Dong et al., 2023). As a result, these models become brittle under out-of-distribution (OOD) prompts, where small shifts in instruction style or reasoning demands can trigger disproportionate failures (Sclar et al., 2024; Perez et al., 2021; Wei et al., 2023).

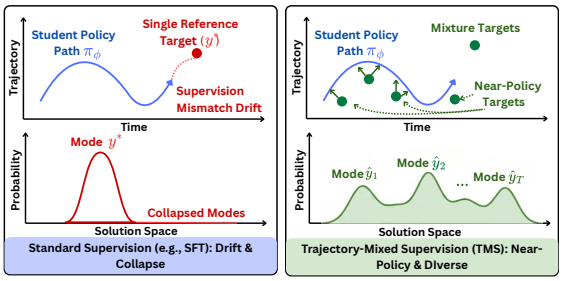

Figure 1. Single-reference supervision (e.g., SFT) induces supervision-mismatch drift and mode collapse; trajectory-mixed supervision (TMS) samples near-policy targets across training, preserving diverse solution modes.

In contrast, on-policy RL-style post-training, such as PPO (Schulman et al., 2017) and verifier-driven variants like GRPO (Shao et al., 2024), is often observed to better preserve broad generalization while still improving target-task performance (Ouyang et al., 2022; Bai et al., 2022a; Shao et al., 2024). A key distinction is *where supervision comes from*: RL continually induces training targets from the model's current policy, keeping learning coupled to the model's evolving behavior, whereas SFT relies on static labels that do not adapt as the policy shifts (Ross et al., 2011; Schulman et al., 2017). Despite these benefits, RL is often harder to adopt at scale because it typically requires reward modeling or verifiers for credit assignment and incurs substantial on-policy sampling overhead (Casper et al., 2023). This leaves practitioners with a persistent dilemma: SFT is easy and efficient but prone to over-specialization and forgetting, while RL is more robust but more complex.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

These observations raise a natural question:

> **Q.** *Can we recover the retention benefits of on-policy optimization without reward models, verifiers, or full RL training loops?*

Existing explanations (e.g., on-policy learning or KL regularization) (Chen et al., 2025; Chu et al., 2025; Jin et al., 2025a; Shenfeld et al., 2025) offer useful intuition, but they do not directly yield a simple, reward-free algorithm that can be dropped into standard SFT pipelines. We bridge this gap by re-framing what makes SFT fragile.

Key observation: SFT optimizes a moving policy against fixed targets. Although SFT is often framed as learning a fixed input–output mapping, the policy π_{θ_t} evolves throughout training while the supervision distribution $q(y|x)$ remains static. This induces *temporal supervision mismatch*: as the model shifts probability mass toward alternative (often valid) trajectories, single-reference cross-entropy can produce large corrective gradients that over-enforce the reference and suppress diversity, especially on tasks with non-unique solutions (Figure 1). In contrast, on-policy RL continually re-couples optimization targets to the current policy, which helps preserve broad capabilities.

Our approach: measure mismatch and fix it via trajectory supervision. We quantify mismatch with *Policy-Label Divergence (PLD)* and propose *Trajectory-Mixed Supervision (TMS)*: a reward-free post-training method that samples supervision from historical checkpoints along a single training trajectory. By mixing near-policy targets across time, TMS reduces mismatch drift and preserves historically plausible solution modes, approximating the stabilizing effect of on-policy optimization without reward models or verifiers.

In Summary, we contribute the following:

> ❶ **PLD framing.** We formalize *Policy-Label Divergence (PLD)* to characterize supervision mismatch in SFT, where a moving policy is optimized against a fixed label distribution, and connect PLD drift to catastrophic forgetting.
>
> ❷ **Reward-free, near-policy post-training.** We propose *Trajectory-Mixed Supervision (TMS)*, which mixes supervision targets from historical checkpoints to reduce PLD and approximate on-policy learning without reward models, verifiers, or RL loops.
>
> ❸ **Empirical and mechanistic validation.** Across model families and scales, TMS improves the accuracy–retention trade-off on reasoning and instruction-following benchmarks, and KL/PLD-based drift metrics reliably predict forgetting and explain why TMS shifts the Pareto frontier.

2. Related Work

Extended related works are listed in Appendix B.

Post-Training Dynamics and Forgetting. Post-training like SFT and RL has become a standard paradigm for adapting foundation models to downstream tasks, often yielding substantial gains in instruction-following, reasoning, and task-specific utility (Ouyang et al., 2022; DeepSeek-AI et al., 2024; Lu et al., 2025; Kirk et al., 2024; Luo et al., 2022; Xie et al., 2024). However, such post-training optimization can induce capability drift and catastrophic forgetting, especially under distribution shift or continual updates (Luo et al., 2025; Li et al., 2024b; Kotha et al., 2024). While prior work mitigates forgetting by constraining parameters or replaying data, these approaches typically assume a fixed supervision signal during post-training; in contrast, we study forgetting as a consequence of training a changing model under static supervision.

Preference Optimization and On-Policy Alignment. Recent work on post-training alignment spans both on-policy reinforcement learning and offline preference optimization objectives (Rafailov et al., 2023; Bhatia et al., 2025; Schulman et al., 2017; Annadani et al., 2025; Yan et al., 2025). A group of studies finds that RL-based post-training can exhibit substantially less forgetting and better generalization than supervised fine-tuning (SFT) under distribution shift (Chen et al., 2025; Chu et al., 2025; Lai et al., 2026; Jin et al., 2025b). Our work shows that a key source of this retention advantage lies in keeping supervision aligned with the model's evolving policy, and demonstrates that this effect can be approximated in a reward-free manner.

Trajectory Learning and Data Selection Curricula. Recent work increasingly leverages model-generated trajectories as training signals, enabling self-distillation and iterative refinement without additional labels (Wang et al., 2023; Zelikman et al., 2022; Guo et al., 2025; Yuan et al., 2025; Bai et al., 2022b). Empirical studies suggest that training on distributions induced by the evolving policy can substantially reduce forgetting relative to static supervision (Chu et al., 2025; Lai et al., 2026; Jin et al., 2025a). Unlike prior trajectory-based methods focused on performance or curricula, our work uses trajectories to align supervision with an evolving policy and reduce forgetting without rewards.

3. Preliminaries

3.1. Post-Training Formulation

We consider post-training of a language model parameterized by θ, initialized from a pre-trained base policy π_{θ_0}. The model maps inputs $x \in \mathcal{X}$ to outputs $y \in \mathcal{Y}$.

Supervised Fine-Tuning (SFT). SFT assumes a static dataset $\mathcal{D} = \{(x_i, y_i^*)\}_{i=1}^N$ drawn from a reference distri-

bution $q(y|x)$ (e.g., human experts, verifiers, or a teacher model). The objective minimizes the negative log-likelihood of the reference labels under the current policy π_θ (with θ updated over steps t):

$$\mathcal{L}_{\text{SFT}}(\theta) = -\mathbb{E}_{(x,y^*)\sim\hat{p}_\mathcal{D}}\left[\log \pi_\theta(y^*|x)\right]. \tag{1}$$

Crucially, in SFT the target distribution q remains fixed throughout training, while the policy π_θ evolves, inducing a temporal mismatch analyzed in subsequent sections.

Reinforcement Learning (RL). RL fine-tuning optimizes a reward signal $r(x, y)$ over model outputs y conditioned on prompts x. A common formulation maximizes expected reward while regularizing deviation from the base model via a KL penalty:

$$\max_\theta \ \mathbb{E}_{x\sim\mathcal{D}_{\text{prompt}}}\big[J(\theta; x)\big],$$
$$J(\theta; x) = \mathbb{E}_{y\sim\pi_\theta(\cdot|x)}[r(x, y)] - \beta\, D_{\text{KL}}\big(\pi_\theta(\cdot|x) \,\|\, \pi_{\theta_0}(\cdot|x)\big).$$

Here, $\mathcal{D}_{\text{prompt}}$ denotes the prompt distribution (which may overlap with inputs in $\mathcal{D}$). RL is *on-policy* with respect to $(y|x)$: training trajectories (responses) are sampled from the evolving policy itself, so the response distribution shifts with model capabilities.

3.2. Evaluation Framework: Target vs. Retention

We distinguish improvement on the trained task from preservation of existing capabilities.

❶ **Target Performance (S_{tgt}):** performance on downstream tasks $\mathcal{T}_{\text{tgt}}$ used for post-training.

❷**Retention Performance (S_{ret}):** performance on a held-out suite $\mathcal{T}_{\text{ret}}$ not seen during post-training, measuring retained capabilities and robustness to distribution shift.

3.3. The Forgetting Metric

Let $S_b(\pi)$ be the score of policy π on benchmark $b \in \mathcal{T}_{\text{ret}}$, and $\Delta_b = S_b(\pi_{\theta_t}) - S_b(\pi_{\theta_0})$. We define the Forgetting Score as average degradation across the retention suite, considering only negative changes:

$$\mathcal{F}(\pi_{\theta_t}) = \frac{1}{|\mathcal{T}_{\text{ret}}|} \sum_{b\in\mathcal{T}_{\text{ret}}} \min(\Delta_b, 0). \tag{2}$$

In this formulation, $\mathcal{F} \le 0$; more negative values indicate worse catastrophic forgetting.

4. What Exactly Fails in SFT?

Prior work (Chen et al., 2025; Chu et al., 2025; Jin et al., 2025a; Shenfeld et al., 2025) often attributes the superior retention of RL fine-tuning to broad factors such as KL regularization or the use of preference/negative samples. We argue that forgetting under SFT is not monolithic. Instead, it arises from two largely independent failure modes that emerge when optimizing an evolving policy against a static supervision distribution: *(A) temporal supervision mismatch* and *(B) mode collapse under single-reference cross-entropy*.

Table 1. **Failure Mode A: Mismatch drift during SFT.**

Step	Train NLL↓	Val NLL↓	MATH Acc↑	ARC-C Acc↑
0 (Base)	1.85	1.88	48.1%	66.9%
500	0.85	0.92	52.5%	59.1%
1000	0.42	0.44	56.5%	52.3%
2000	**0.15**	**0.67**	**61.2%**	**49.5%**
5000	0.04	0.98	62.4%	45.2%

4.1. Failure Mode A: Temporal Supervision Mismatch

The core structural property of SFT is that the supervision distribution $q(y|x)$ is fixed, while the policy π_{θ_t} evolves over optimization steps.

❶ **Mechanism.** As training progresses, π_{θ_t} often places mass on responses that are semantically correct but differ from the single reference y^* in surface form (e.g., phrasing, reasoning style, intermediate steps). When $\pi_{\theta_t}(y^*|x)$ becomes small, token-level cross-entropy induces large gradients that pull the policy back toward y^*, producing non-local updates that can interfere with features supporting unrelated capabilities. Intuitively, the optimizer is forced to spend capacity matching a rigid template rather than preserving general representations.

❷ **Evidence and observable signatures.** We observe a characteristic *knee point*: training NLL decreases monotonically, while a held-out estimate of the same objective on $\mathcal{D}_{\text{val}}$ (from the same task distribution) improves early and then degrades at later checkpoints. Table 1 illustrates the pattern on QWEN2.5-1.5B (Yang et al., 2024) fine-tuned with SFT on MATH (Hendrycks et al., 2021c). While training NLL continues to fall, held-out NLL reaches a minimum at step 1000 and then increases. In the same interval, retention on ARC-C (Clark et al., 2018) collapses, even though target MATH accuracy continues to improve slightly. This decoupling indicates that SFT can "converge" on the training objective while drifting away from generalizable behavior.

Hypothesis A (H-A). *A major driver of SFT forgetting is supervision mismatch drift: as the policy evolves away from the fixed supervision distribution, held-out PLD (proxied by validation NLL) increases and retention collapses even while training loss improves.*

4.2. Failure Mode B: Mode Collapse Under Single-Reference CE Targets

Standard SFT applies token-level cross-entropy to a single reference trajectory y^*. For many reasoning and code tasks,

Table 2. **Failure Mode B: Mode collapse under SFT.**

Method	Pass@1↑	Pass@100↑	SC-Acc↑	AnsEnt↑
Base Model	48.1%	62.0%	56.0%	**2.45**
Standard SFT	62.4%	66.0%	68.0%	0.85
RL (GRPO)	**63.5%**	73.9%	**75.2%**	1.35
TMS (Ours)	62.8%	**74.0%**	70.9%	1.28

the solution set is non-unique: there exist many valid outputs y' that reach the correct answer.

❶ **Mechanism.** By concentrating likelihood mass on y^*, SFT implicitly suppresses alternative valid trajectories y', encouraging collapse onto a narrow mode. This is particularly harmful when robustness requires maintaining support over multiple valid reasoning paths (*e.g.*, paraphrases).

❷ **Evidence and diagnostics.** We quantify mode collapse using coverage and diversity proxies computed from K sampled rollouts per prompt under fixed decoding across methods: (i) *Pass@K* (coverage); (ii) *Self-consistency accuracy* (SC-Acc; majority-vote accuracy over K samples); and (iii) *Answer entropy* (AnsEnt; entropy of the empirical answer distribution; higher implies broader support). On MATH, SFT improves Pass@1 but sharply reduces answer entropy, consistent with collapsing to a narrow solution mode (Table 2). In contrast, on-policy RL (GRPO) and our TMS method preserve substantially higher entropy while maintaining strong coverage (Pass@100). Throughout, we compute Pass@K/SC-Acc/AnsEnt with the same sampling protocol for all methods, so the observed differences reflect changes in model support rather than evaluation artifacts.

Hypothesis B (H-B). *On tasks with non-unique solutions, single-reference SFT induces mode collapse, which harms robustness and contributes to downstream degradation. Methods that preserve multi-modal support improve coverage and retention.*

4.3. Policy-Label Divergence (PLD)

To formalize and detect these failures, we introduce **Policy-Label Divergence (PLD)**. While prior analyses often focus on drift from the base model (π_{θ_0}), PLD measures the tension between the model's current *beliefs* (π_{θ_t}) and the *supervision* distribution (q) it is trained to match.

Formal definition (forward form aligned with SFT). We define PLD as the expected forward KL (equivalently cross-entropy up to a constant) between the supervision distribution and the policy:

$$\mathrm{PLD}(\theta_t; q) = \mathbb{E}_{x\sim\mathcal{D}_x}\left[D_{\mathrm{KL}}(q(\cdot|x) \,\|\, \pi_{\theta_t}(\cdot|x))\right]. \quad (3)$$

In the ideal case, $q(y|x)$ would represent the full ground-truth posterior. In standard single-reference SFT, q is typically a Dirac delta δ_{y^*} concentrated on a single reference solution y^*. In this setting, minimizing PLD is equivalent to minimizing the usual token-level cross-entropy objective.

The generalization gap (PLD drift). The key distinction arises when we estimate PLD on *held-out* inputs drawn from the same task distribution rather than the training samples themselves. Let $\mathcal{D}_{\text{train}}$ be the training set and $\mathcal{D}_{\text{val}}$ a held-out set from the same distribution:

- ❒ **Memorization phase:** On $\mathcal{D}_{\text{train}}$, PLD decreases monotonically as the model fits the reference trajectories.
- ❒ **Mismatch drift phase:** On $\mathcal{D}_{\text{val}}$, PLD may reach a minimum early in training and then increase. We interpret this rising tail as **supervision mismatch drift**: the policy is being pulled toward increasingly brittle reference-matching rather than maintaining generalizable representations.

For single-reference supervision, we approximate held-out PLD using validation negative log-likelihood (NLL):

$$\widehat{\mathrm{PLD}}_{\text{val}}(\theta_t) = -\frac{1}{|\mathcal{D}_{\text{val}}|}\sum_{(x,y^*)\in\mathcal{D}_{\text{val}}} \log \pi_{\theta_t}(y^*|x). \quad (4)$$

A model can maintain high target accuracy (e.g., correct final answers) while $\widehat{\mathrm{PLD}}_{\text{val}}$ increases, indicating growing disagreement with the supervision template.

5. Method: Trajectory-Mixed Supervision

We introduce **Trajectory-Mixed Supervision (TMS)**, a reward-free post-training procedure that addresses both failure modes in Section 4. TMS replaces static single-reference supervision with a *trajectory mixture* of the model's own intermediate policies, turning the optimization path into a curriculum. Intuitively, intermediate checkpoints provide (i) *near-policy* targets that reduce supervision mismatch drift, and (ii) broader support over valid solution modes that mitigate mode collapse. We formalize this perspective and relate TMS to mismatch and mode preservation in Appendix D.

5.1. Core Algorithm

Stage 1: Trajectory harvesting. Given a training set $\mathcal{D} = \{(x_i, y_i^*)\}_{i=1}^N$, we run a standard post-training procedure for a fixed budget and record T intermediate checkpoints $\{\theta_1, \ldots, \theta_T\}$ at uniformly spaced steps.[1] For each checkpoint t and input x, we generate a response $\hat{y}^{(t)}(x) \sim \pi_{\theta_t}(\cdot|x)$ and store the mapping $(x, t) \mapsto \hat{y}^{(t)}(x)$. We denote the resulting trajectory buffer by

$$\mathcal{H} = \{\hat{y}^{(t)}(x) \;:\; x \in \mathcal{D}_x,\; t \in \{1, \ldots, T\}\},$$

where $\mathcal{D}_x$ denotes the empirical distribution over inputs $\mathcal{D}$.

Stage 2: Trajectory-mixed supervision distribution. For each input x, TMS defines an empirical mixture supervision

[1] In all experiments, we use $T = 10$ unless otherwise stated.

distribution supported on the trajectory outputs:

$$m(\cdot|x) = \frac{1}{T}\sum_{t=1}^{T} \delta_{\hat{y}^{(t)}(x)}(\cdot). \tag{5}$$

Optionally, we interpolate with the original reference distribution $q(\cdot|x)$ (e.g., δ_{y^*} for single-reference SFT) using a mixing weight $\alpha \in [0, 1]$:

$$q_\alpha(\cdot|x) = \alpha\, q(\cdot|x) + (1-\alpha)\, m(\cdot|x). \tag{6}$$

In the main setting, we use the trajectory mixture ($\alpha = 0.25$) unless stated otherwise.

Stage 3: Student training. We train a student policy π_ϕ from the same initialization as the baseline (e.g., $\phi_0 = \theta_0$) by minimizing the forward KL (equivalently cross-entropy) to the mixture supervision:

$$\min_{\phi} \; \mathbb{E}_{x\sim\mathcal{D}_x}\Big[D_{\mathrm{KL}}\big(q_\alpha(\cdot|x) \,\|\, \pi_\phi(\cdot|x)\big)\Big]. \tag{7}$$

Operationally, this corresponds to sampling $t \sim \mathrm{Uniform}\{1, \ldots, T\}$ (or sampling from q with probability α) and applying standard token-level NLL on it. The full algorithm can be found in Appendix C.

6. Theoretical Analysis

This section provides a formal lens for the empirical failure modes in Section 4. We first clarify why single-reference SFT can be structurally misaligned with robustness on non-unique tasks. We then show that changes in expected task performance are controlled by distributional drift from the base model, yielding a principled justification for KL-to-base as a predictor of forgetting.

6.1. Motivation: Why static single-reference supervision can be misaligned

SFT optimizes a forward-KL (cross-entropy) projection onto a fixed supervision distribution $q(y|x)$. When q is *sparse*, e.g., a Dirac delta δ_{y^*} representing a single reference trajectory, the projection can encourage concentration of probability mass on that trajectory, suppressing alternative valid solutions. This aligns with Failure Mode B (Section 4) and motivates relaxing q into a broader supervision distribution that preserves multi-modal support.

Proposition 6.1 (Single-reference projection encourages concentration). *For a fixed input x, minimizing the cross-entropy $\mathbb{E}_{y\sim q(\cdot|x)}[-\log \pi_\theta(y|x)]$ with $q(\cdot|x) = \delta_{y^*}$ is equivalent to maximizing $\log \pi_\theta(y^*|x)$. This objective provides no positive training signal for alternative valid outputs $y' \neq y^*$, and therefore can reduce support over the set of valid solutions when capacity is limited or optimization is continued beyond the point of generalization.*

Proposition 6.1 is not a claim that SFT must always collapse modes, but that its *training signal* is inherently one-sided under single-reference supervision. TMS replaces the fixed q by a trajectory mixture whose support includes multiple historically plausible outputs, directly addressing this.

6.2. Bounding behavior change via distribution drift

We now show that distributional drift from the base model upper-bounds changes in expected performance on held-out tasks. This formalizes why KL-to-base is a meaningful predictor of forgetting (and complements PLD, which diagnoses mismatch with supervision).

- **Assumption 1 (Bounded score).** For a given held-out task, let $f_x : \mathcal{Y} \to [0, 1]$ be a bounded scoring function for prompt x (e.g., indicator of correctness, normalized reward).
- **Assumption 2 (Prompt distribution).** Prompts are drawn from a fixed evaluation distribution $x \sim \mathcal{D}_{\text{eval}}$ (e.g., the held-out suite prompts).

Theorem 6.2 (Forgetting is controlled by KL drift). *Let $\pi_0(\cdot|x)$ be the base policy and $\pi_\theta(\cdot|x)$ a fine-tuned policy. Then the change in expected score on $\mathcal{D}_{eval}$ is bounded by the KL divergence from the base:*

$$\Big|\mathbb{E}_{x\sim\mathcal{D}_{eval}}\big[\mathbb{E}_{y\sim\pi_\theta(\cdot|x)} f_x(y)\big] - \mathbb{E}_{x\sim\mathcal{D}_{eval}}\big[\mathbb{E}_{y\sim\pi_0(\cdot|x)} f_x(y)\big]\Big| \le \mathbb{E}_{x\sim\mathcal{D}_{eval}}\Big[\sqrt{2\, D_{\mathrm{KL}}(\pi_\theta(\cdot|x) \,\|\, \pi_0(\cdot|x))}\Big].$$

Proof sketch. Fix a prompt x. The difference in expected score between π_θ and π_0 can be written as an inner product between the score function f_x (bounded in $[0, 1]$) and the signed measure $(\pi_\theta - \pi_0)$. By L_1/L_∞ duality, this difference is at most twice the total variation distance between $\pi_\theta(\cdot|x)$ and $\pi_0(\cdot|x)$. Pinsker's inequality upper-bounds total variation by $\sqrt{\frac{1}{2}\mathrm{KL}}$, yielding the stated bound. The full proof is in Appendix E.

7. Experiments

7.1. Experimental Setup

Task groups and evaluation protocol. We evaluate post-training along three axes: **(1) In-domain target performance** on the task used for training, **(2) cross-task transfer** to other downstream tasks not seen during training, and **(3) held-out retention** on capability benchmarks never used for post-training. For details, please refer to Appendix F.

Datasets. We consider the following task groups:

(1) Target tasks (used for training). We fine-tune on **Math-500** (Lightman et al., 2023), **MATH** (Hendrycks et al.,

Table 3. **Main results across models.** Target-task performance S_{tgt} (higher is better), cross-task transfer to other target tasks S_{xfer} (Avg Δ, lower is better), and held-out retention. Parentheses denote change vs. the base model; **best** / second-best are highlighted.

Method	Target Tasks S_{tgt} ↑						Cross-task S_{xfer} ↓	Held-out Retention ↑		
	Math500	MATH	GSM8K	Count.	IFEval	MMLU	Avg Δ other tgt	ARC-C	Hotpot	SafetyBench
Qwen-2.5-1.5B-Instruct										
Base	46.6	48.1	68.9	13.2	38.3	30.8	–	66.9	43.1	35.1
Standard SFT	58.4	62.4	77.3	25.2	54.1	**39.0**	↓ 26.2	42.3(↓ 24.6)	19.4(↓ 23.7)	31.4(↓ 3.7)
Self-SFT	54.9	57.1	73.1	29.4	47.3	34.7	↓ 18.4	51.4(↓ 15.5)	27.5(↓ 15.6)	34.3(↓ 0.8)
Final-SFT	57.1	61.9	**78.0**	25.1	53.1	36.8	↓ 24.3	48.3(↓ 18.6)	25.3(↓ 17.8)	34.0(↓ 1.1)
REINFORCE	56.9	60.1	75.2	26.1	49.3	34.7	↓ 3.4	63.8(↓ 3.1)	40.1(↓ 3.0)	35.0(↓ 0.1)
GRPO	58.9	**63.5**	76.3	**35.2**	53.6	35.2	↓ 1.2	**66.7**(↓ 0.2)	**42.0**(↓ 1.1)	**36.3**(↑ 1.3)
TMS (Ours)	**59.0**	62.8	76.8	32.4	**54.4**	38.6	↓ 2.9	65.4(↓ 1.4)	41.4(↓ 1.7)	34.8(↓ 0.3)
Qwen-2.5-3B-Instruct										
Base	60.2	62.0	85.2	29.6	60.6	43.8	–	80.5	53.6	35.7
Standard SFT	76.4	80.3	**90.2**	49.2	71.3	**54.0**	↓ 39.2	42.7(↓ 37.8)	24.1(↓ 29.5)	31.1(↓ 4.6)
Self-SFT	74.2	77.4	87.3	48.6	67.9	47.2	↓ 29.3	51.8(↓ 28.7)	33.1(↓ 20.5)	33.6(↓ 2.1)
Final-SFT	75.8	79.6	89.5	48.1	70.2	51.4	↓ 35.1	45.2(↓ 35.3)	28.4(↓ 25.2)	33.2(↓ 2.5)
REINFORCE	75.1	78.2	88.4	47.3	67.2	49.1	↓ 4.1	78.6(↓ 1.9)	51.4(↓ 2.2)	**35.5**(↓ 0.2)
GRPO	77.4	**81.5**	90.1	**54.2**	70.9	50.8	↓ 1.9	**80.2**(↓ 0.3)	**53.1**(↓ 0.5)	35.4(↓ 0.3)
TMS (Ours)	**77.8**	81.1	88.9	51.3	**72.0**	53.6	↓ 2.3	79.2(↓ 1.3)	51.7(↓ 1.9)	35.0(↓ 0.7)
LLaMA-3.1-8B-Instruct										
Base	47.2	46.0	84.1	18.6	67.0	46.2	–	84.1	52.6	33.8
Standard SFT	66.7	64.6	**89.6**	39.2	75.2	**55.1**	↓ 32.9	53.2(↓ 30.9)	28.2(↓ 24.4)	30.1(↓ 3.7)
Self-SFT	60.4	60.4	87.4	36.4	71.6	50.8	↓ 24.9	64.2(↓ 19.9)	39.1(↓ 13.5)	32.6(↓ 1.2)
Final-SFT	64.3	63.1	88.8	37.4	73.8	53.2	↓ 31.4	60.8(↓ 23.3)	36.2(↓ 16.4)	32.2(↓ 1.6)
REINFORCE	63.9	61.8	88.1	34.8	72.2	49.3	↓ 4.9	82.1(↓ 2.0)	50.6(↓ 2.0)	33.6(↓ 0.2)
GRPO	**70.5**	**65.4**	89.4	**41.2**	75.6	50.8	↓ 0.9	**83.8**(↓ 0.3)	**52.1**(↓ 0.5)	**34.1**(↑ 0.3)
TMS (Ours)	66.3	64.9	89.2	40.4	**76.1**	54.6	↓ 2.3	83.1(↓ 1.0)	51.4(↓ 1.2)	33.0(↓ 0.8)

2021c), **GSM8K** (Cobbe et al., 2021), and **Countdown** (Pan et al., 2025) (verifiable multi-step reasoning), and also on **IFEval** (Zhou et al., 2023) (instruction following) and **MMLU** (Hendrycks et al., 2021a;b) (knowledge). These tasks span both non-unique solution spaces (math) and more rigid evaluation formats (IFEval/MMLU), enabling targeted tests of our hypotheses. For MATH-500, MATH, GSM8K, and Countdown, the downstream performance is measured using the test split of the datasets. For IfEval, we evaluate on the training set, while for MMLU, the downstream performance is measured on MMLU-Pro (Wang et al., 2024).

(2) Cross-task transfer (held-out downstream tasks). For each run trained on a specific target task $\mathcal{T}_{\text{train}}$ (e.g., MATH), we additionally evaluate the post-trained model on the remaining target tasks $\mathcal{T}_{\text{other}} = \mathcal{T}_{\text{tgt}} \setminus \{\mathcal{T}_{\text{train}}\}$. We report the *average delta* relative to the base model on $\mathcal{T}_{\text{other}}$ as a cross-task generalization score, capturing whether improvements reflect transferable capability or brittle specialization.

(3) Held-out retention suite (never trained on). To measure catastrophic forgetting and robustness beyond the trained distribution, we evaluate on **ARC-Challenge** (Clark et al., 2018) (commonsense/science), **HotpotQA** (Yang et al., 2018) (multi-hop reading), and safety benchmarks **WildGuardTest** (Han et al., 2024), **SafetyBench** (Zhang et al., 2023), **WildJailbreak** (Jiang et al., 2024).

Models. We evaluate across instruction-tuned open-weight LLMs: *LLaMA-3.1-8B-Instruct* (Grattafiori et al., 2024), *Qwen-2.5-7B-Instruct* (Yang et al., 2024), *LLaMA-3.2-1B-Instruct*, *Qwen-2.5-1.5B-Instruct*, and *Qwen-2.5-3B-Instruct*. For all SFT-based models, we use *LLaMA-3.3-70B* as the teacher to generate our CoTs and filter them based on corrections. For RL algorithms, we use a simple binary verifier based on answer correctness.

Baselines. We compare against strong post-training baselines: **(1) Standard SFT** (ground-truth/teacher references), **(2) Self-SFT** (distill from π_{θ_0} outputs), **(3) Final-SFT** (distill from converged SFT outputs), and on-policy RL baselines **(5) GRPO** (Shao et al., 2024) and **(6) REINFORCE**. For all fine-tuning, we fine-tune the full model.

Metrics reported. For each training run on $\mathcal{T}_{\text{train}}$, we report: **(i) Target score** $S_{\text{tgt}}(\mathcal{T}_{\text{train}})$, **(ii) Cross-task transfer** $S_{\text{xfer}} = \frac{1}{|\mathcal{T}_{\text{other}}|} \sum_{b \in \mathcal{T}_{\text{other}}} \big(S_b(\pi_{\text{post}}) - S_b(\pi_{\theta_0})\big)$, and **(iii) Retention/forgetting** on $\mathcal{T}_{\text{ret}}$, reported both as absolute scores and the aggregate forgetting score $\mathcal{F}$ (defined in Section 3).

7.2. Main Results

We now answer RQ1–RQ4 using Table 3. Full results can be found in Appendix H. Across model families and scales, a consistent pattern emerges: **standard SFT reliably im-**

proves target-task accuracy but induces large cross-task interference and severe retention collapse, while **TMS achieves near-RL retention with SFT-level performance**.

RQ1 (Target performance). *Does TMS outperform standard SFT on the trained (in-domain) target task under matched compute?*

Target tasks: TMS preserves (and often improves) the SFT gains. Across all three models, TMS matches or exceeds SFT on most target benchmarks, indicating that trajectory mixing does not "wash out" learning signal. For *Qwen-2.5-1.5B*, TMS attains the best Math500 score and the best IFEval score, while remaining competitive on the remaining targets (Table 3). For *Qwen-2.5-3B*, TMS improves over SFT on Math500 and IFEval, and stays close on MATH/GSM8K/MMLU. For *LLaMA-3.1-8B*, TMS improves IFEval and MMLU, while matching SFT on the remaining targets. Overall, TMS consistently retains the primary benefit of post-training, *i.e.*, in-domain competence, while avoiding the instability.

RQ2 (Forgetting). *Does TMS reduce catastrophic forgetting on a held-out retention suite relative to standard SFT?*

Retention: TMS sharply reduces the collapse induced by SFT. The most striking effect in Table 3 is that SFT causes pronounced degradation on held-out capabilities, despite improving target-task accuracy. For *Qwen-2.5-1.5B*, standard SFT severely degrades ARC-C and HotpotQA (*e.g.*, ARC-C drops from 66.9 to 42.3), whereas TMS retains near-base performance. The same qualitative pattern appears at larger scales: on *Qwen-2.5-3B*, SFT collapses ARC-C from 80.5 to 42.7, while TMS preserves 79.2; on *LLaMA-3.1-8B*, SFT drops ARC-C from 84.1 to 53.2, while TMS retains 83.1. These large retention gaps demonstrate that trajectory mixing is an effective intervention against catastrophic forgetting, consistent with the mismatch and mode-collapse diagnoses in Section 4.

RQ3 (RL parity without rewards). *Can TMS match the accuracy–retention tradeoff of on-policy RL baselines* without *reward models or verifiers?*

Accuracy–retention frontier: TMS tracks on-policy RL while remaining reward-free. On-policy RL baselines (GRPO/REINFORCE) exhibit a characteristic signature across models: strong retention with low cross-task drift, but at non-trivial pipeline complexity. TMS attains a similar operating point without rewards, critics, or on-policy optimization. For *Qwen-2.5-3B*, TMS is close to GRPO on retention

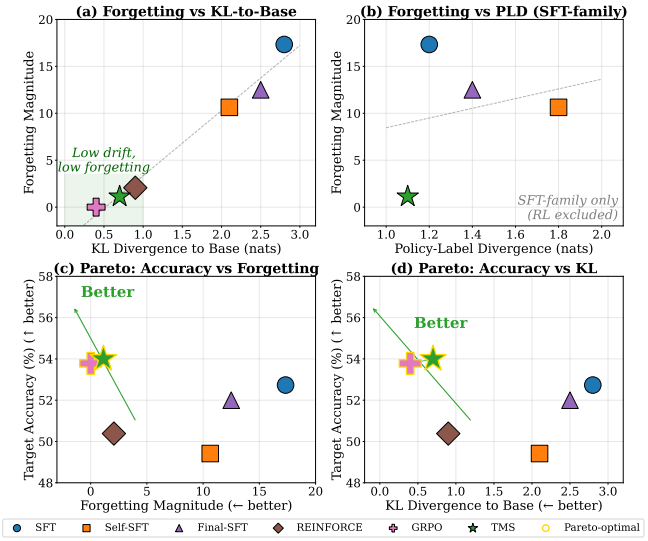

Figure 2. **RQ5 (Mechanistic law + Pareto).** (a) Forgetting magnitude increases with KL-to-base. (b) PLD predicts forgetting within the SFT-family (RL excluded). (c–d) Pareto frontiers show TMS moving closer to the accuracy–retention frontier relative to SFT/self-distillation.

(ARC-C: 79.2 vs. 80.2; HotpotQA: 51.7 vs. 53.1) while remaining competitive on target tasks (e.g., Math500: 77.8 vs. 77.4; IFEval: 72.0 vs. 70.9). For *LLaMA-3.1-8B*, TMS similarly tracks GRPO on retention, with comparable target accuracy. The gap is most visible in the cross-task transfer column: GRPO tends to minimize cross-task drop, while TMS incurs a modestly larger drop, yet still dramatically improves over SFT. Taken together, these results place TMS near the same Pareto region as on-policy RL, supporting the claim that *near-policy supervision* can be approximated through trajectory mixing alone.

RQ4 (Beyond self-distillation). *Are TMS gains explained by simple self-distillation (one checkpoint) or does the* trajectory mixture *provide additional benefit?*

Beyond single-checkpoint distillation: trajectory mixing is essential. Self-SFT and Final-SFT provide informative baselines: they replace oracle labels with a single policy snapshot, removing part of the supervision mismatch while keeping the training procedure identical. Across all models, these baselines improve retention relative to standard SFT but remain consistently below TMS. For *Qwen-2.5-1.5B*, Self-SFT improves ARC-C to 51.4, yet TMS achieves 65.4; similar gaps hold on HotpotQA. For *Qwen-2.5-3B*, Self-SFT recovers ARC-C to 51.8, while TMS preserves 79.2; and for *LLaMA-3.1-8B*, Self-SFT achieves 64.2 on ARC-C versus 83.1 for TMS. These patterns rule out an explanation based solely on "distill-to-self" and indicate that **mixing supervision across the training trajectory**, rather than selecting any single checkpoint, is the key ingredient.

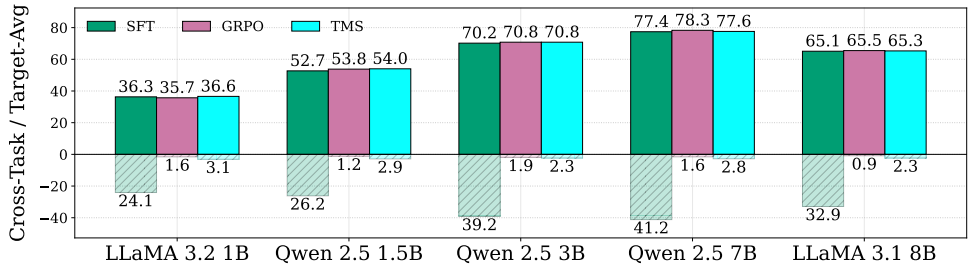

Figure 3. **RQ6 (Scaling):** Target Avg (higher is better) and Cross-task (lower is better) across model scales. TMS tracks GRPO-style low cross-task drift while matching SFT-level target accuracy.

RQ5 (Task dependence). *When does TMS help most: which task properties (e.g., non-unique solutions vs. rigid formats) predict larger gains?*

Task dependence: larger gains appear on flexible, multi-solution targets. Table 3 suggests a consistent pattern aligned with our dissection in Section 4: TMS yields its clearest advantages on targets where solution spaces are non-unique and supervision mismatch is common (*e.g.*, math and instruction-following). On such tasks, TMS typically matches SFT on target metrics while substantially improving held-out retention. In contrast, on more rigid or multiple-choice-style targets (*e.g.*, MMLU), TMS is generally competitive, but the gap to SFT is larger.

7.3. Mechanistic and Secondary Analyses

For further analyses, please refer to Appendix G.

RQ6 (Mechanistic law). *When we compute (i) KL-to-base on new-task prompts, (ii) PLD to the supervision distribution of each method, and (iii) forgetting on the held-out suite, do KL/PLD reliably predict forgetting, and does TMS shift the Pareto frontier relative to SFT?*

KL-to-base predicts forgetting, and TMS occupies the low-drift/low-forgetting regime. Figure 2a shows a strong monotonic relationship between KL-to-base and forgetting magnitude: methods that drift farther from the base model tend to forget more. In particular, SFT-family baselines exhibit both high KL drift and large forgetting, while GRPO and TMS cluster in the lower-left "low drift, low forgetting" region. This directly supports and provides a mechanistic explanation for why TMS retains capabilities despite matching target-task performance.

PLD is most informative within the SFT family. When PLD is computed w.r.t. the SFT-style supervision distribution, it is not directly comparable across RL methods. Accordingly, Figure 2b reports PLD–forgetting trends within the SFT-family only (RL excluded). In this regime, higher PLD aligns with higher forgetting, and TMS achieves the lowest PLD among SFT-style methods while also exhibiting the smallest forgetting magnitude.

Pareto analysis: TMS shifts the accuracy–retention frontier relative to SFT. Figure 2c–d visualize the core tradeoffs. In the accuracy vs. forgetting plane (c), and accuracy vs. KL plane (d), SFT and self-distillation baselines are dominated: they incur substantially higher forgetting and drift at comparable accuracy. In contrast, TMS lies on (or very near) the Pareto-optimal set alongside GRPO, indicating that it achieves a better accuracy–retention tradeoff than SFT.

RQ7 (Scaling). *How do TMS effects vary with model scale, in both target performance and retention?*

Scaling: TMS remains stable across 1B–8B and preserves the GRPO-like low cross-task drift. Figure 3 summarizes Target Avg and Cross-task transfer across five model sizes. Two trends are consistent: (i) Target Avg increases predictably with scale for all methods, and TMS matches SFT and GRPO closely; (ii) cross-task drift under SFT remains large across scales, while TMS stays close to GRPO with small cross-task drops. This indicates the TMS advantage is not a small-model artifact and persists across families and sizes.

8. Conclusion

We studied why post-training with static supervision (SFT) often improves target-task performance at the cost of broad capability retention. We introduced **Trajectory-Mixed Supervision (TMS)**, a reward-free alternative that turns the SFT optimization path into a supervision curriculum, reducing *policy–label mismatch* and mitigating mode collapse. Across model families and scales, TMS matches SFT-level target gains while approaching on-policy RL (GRPO/REINFORCE) in retention and cross-task stability. Mechanistically, KL-to-base reliably predicts forgetting, and TMS shifts the accuracy–retention Pareto frontier relative to SFT. These results suggest that *on-policy-like stability can be recovered without rewards* by designing supervision to track the model's evolving support, making TMS a simple drop-in primitive for robust post-training.

Impact Statement

This paper proposes *Trajectory-Mixed Supervision (TMS)*, a reward-free post-training method intended to improve the accuracy–retention tradeoff of large language models by reducing supervision mismatch and catastrophic forgetting. If adopted, TMS could yield practical benefits such as more reliable continual updates, lower dependence on reward models or complex RL pipelines, and improved stability of safety-related behaviors under post-training.

At the same time, improving retention and robustness can also increase the capability of models in ways that may be misused (e.g., retaining stronger general problem-solving ability while being adapted for a specialized domain). In addition, TMS relies on model-generated supervision from intermediate checkpoints; if the base model exhibits harmful or biased behaviors, mixing trajectory outputs may propagate such patterns unless mitigated through careful data curation and safety evaluation. To reduce these risks, we report held-out safety benchmarks (e.g., SafetyBench and jailbreak ASR) and encourage practitioners to pair TMS with standard deployment safeguards, auditing, and domain-appropriate oversight.

Overall, this work aims to advance the science of post-training dynamics and the design of safer, more stable adaptation procedures.

References

Agarwal, R., Vieillard, N., Zhou, Y., Stanczyk, P., Garea, S. R., Geist, M., and Bachem, O. On-policy distillation of language models: Learning from self-generated mistakes. In *The twelfth international conference on learning representations*, 2024.

Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., and Tuytelaars, T. Memory aware synapses: Learning what (not) to forget, 2018. URL https://arxiv.org/abs/1711.09601.

Annadani, Y., Belakaria, S., Ermon, S., Bauer, S., and Engelhardt, B. E. Preference-guided diffusion for multi-objective offline optimization, 2025. URL https://arxiv.org/abs/2503.17299.

Bai, Y., Kadavath, S., Kundu, S., Askell, A., Kernion, J., Jones, A., Chen, A., Goldie, A., Mirhoseini, A., McKinnon, C., Chen, C., Olsson, C., Olah, C., Hernandez, D., Drain, D., Ganguli, D., Li, D., Tran-Johnson, E., Perez, E., Kerr, J., Mueller, J., Ladish, J., Landau, J., Ndousse, K., Lukosuite, K., Lovitt, L., Sellitto, M., Elhage, N., Schiefer, N., Mercado, N., DasSarma, N., Lasenby, R., Larson, R., Ringer, S., Johnston, S., Kravec, S., Showk, S. E., Fort, S., Lanham, T., Telleen-Lawton, T., Conerly, T., Henighan, T., Hume, T., Bowman, S. R., Hatfield-Dodds, Z., Mann, B., Amodei, D., Joseph, N., McCandlish, S., Brown, T., and Kaplan, J. Constitutional ai: Harmlessness from ai feedback, 2022a. URL https://arxiv.org/abs/2212.08073.

Bai, Y., Kadavath, S., Kundu, S., Askell, A., Kernion, J., Jones, A., Chen, A., Goldie, A., Mirhoseini, A., McKinnon, C., et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022b.

Bhatia, M., Nayak, S., Kamath, G., Mosbach, M., Stańczak, K., Shwartz, V., and Reddy, S. Value drifts: Tracing value alignment during llm post-training, 2025. URL https://arxiv.org/abs/2510.26707.

Casper, S., Davies, X., Shi, C., Gilbert, T. K., Scheurer, J., Rando, J., Freedman, R., Korbak, T., Lindner, D., Freire, P., Wang, T., Marks, S., Segerie, C.-R., Carroll, M., Peng, A., Christoffersen, P., Damani, M., Slocum, S., Anwar, U., Siththaranjan, A., Nadeau, M., Michaud, E. J., Pfau, J., Krasheninnikov, D., Chen, X., Langosco, L., Hase, P., Bıyık, E., Dragan, A., Krueger, D., Sadigh, D., and Hadfield-Menell, D. Open problems and fundamental limitations of reinforcement learning from human feedback, 2023. URL https://arxiv.org/abs/2307.15217.

Chandra, A., Agrawal, A., Hosseini, A., Fischmeister, S., Agarwal, R., Goyal, N., and Courville, A. Shape of thought: When distribution matters more than correctness in reasoning tasks, 2025. URL https://arxiv.org/abs/2512.22255.

Chen, H., Razin, N., Narasimhan, K., and Chen, D. Retaining by doing: The role of on-policy data in mitigating forgetting, 2025. URL https://arxiv.org/abs/2510.18874.

Cheng, Z., Hao, S., Liu, T., Zhou, F., Xie, Y., Yao, F., Bian, Y., Zhuang, Y., Dey, N., Zha, Y., et al. Revisiting reinforcement learning for llm reasoning from a cross-domain perspective. *arXiv preprint arXiv:2506.14965*, 2025.

Christiano, P., Leike, J., Brown, T. B., Martic, M., Legg, S., and Amodei, D. Deep reinforcement learning from human preferences, 2023. URL https://arxiv.org/abs/1706.03741.

Chu, T., Zhai, Y., Yang, J., Tong, S., Xie, S., Schuurmans, D., Le, Q. V., Levine, S., and Ma, Y. Sft memorizes, rl generalizes: A comparative study of foundation model post-training. *arXiv preprint arXiv:2501.17161*, 2025.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*, 2018.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

DeepSeek-AI, Liu, A., Feng, B., Wang, B., Wang, B., Liu, B., Zhao, C., Dengr, C., Ruan, C., Dai, D., Guo, D., Yang, D., Chen, D., Ji, D., Li, E., Lin, F., Luo, F., Hao, G., Chen, G., Li, G., Zhang, H., Xu, H., Yang, H., Zhang, H., Ding, H., Xin, H., Gao, H., Li, H., Qu, H., Cai, J. L., Liang, J., Guo, J., Ni, J., Li, J., Chen, J., Yuan, J., Qiu, J., Song, J., Dong, K., Gao, K., Guan, K., Wang, L., Zhang, L., Xu, L., Xia, L., Zhao, L., Zhang, L., Li, M., Wang, M., Zhang, M., Zhang, M., Tang, M., Li, M., Tian, N., Huang, P., Wang, P., Zhang, P., Zhu, Q., Chen, Q., Du, Q., Chen, R. J., Jin, R. L., Ge, R., Pan, R., Xu, R., Chen, R., Li, S. S., Lu, S., Zhou, S., Chen, S., Wu, S., Ye, S., Ma, S., Wang, S., Zhou, S., Yu, S., Zhou, S., Zheng, S., Wang, T., Pei, T., Yuan, T., Sun, T., Xiao, W. L., Zeng, W., An, W., Liu, W., Liang, W., Gao, W., Zhang, W., Li, X. Q., Jin, X., Wang, X., Bi, X., Liu, X., Wang, X., Shen, X., Chen, X., Chen, X., Nie, X., Sun, X., Wang, X., Liu, X., Xie, X., Yu, X., Song, X., Zhou, X., Yang, X., Lu, X., Su, X., Wu, Y., Li, Y. K., Wei, Y. X., Zhu, Y. X., Xu, Y., Huang, Y., Li, Y., Zhao, Y., Sun, Y., Li, Y., Wang, Y., Zheng, Y., Zhang, Y., Xiong, Y., Zhao, Y., He, Y., Tang, Y., Piao, Y., Dong, Y., Tan, Y., Liu, Y., Wang, Y., Guo, Y., Zhu, Y., Wang, Y., Zou, Y., Zha, Y., Ma, Y., Yan, Y., You, Y., Liu, Y., Ren, Z. Z., Ren, Z., Sha, Z., Fu, Z., Huang, Z., Zhang, Z., Xie, Z., Hao, Z., Shao, Z., Wen, Z., Xu, Z., Zhang, Z., Li, Z., Wang, Z., Gu, Z., Li, Z., and Xie, Z. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model, 2024. URL https://arxiv.org/abs/2405.04434.

Dong, H., Xiong, W., Goyal, D., Zhang, Y., Chow, W., Pan, R., Diao, S., Zhang, J., Shum, K., and Zhang, T. Raft: Reward ranked finetuning for generative foundation model alignment, 2023. URL https://arxiv.org/abs/2304.06767.

Goodfellow, I. J., Mirza, M., Xiao, D., Courville, A., and Bengio, Y. An empirical investigation of catastrophic forgetting in gradient-based neural networks, 2015. URL https://arxiv.org/abs/1312.6211.

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Guo, D., Yang, D., Zhang, H., Song, J., Wang, P., Zhu, Q., Xu, R., Zhang, R., Ma, S., Bi, X., Zhang, X., Yu, X., Wu, Y., Wu, Z. F., Gou, Z., Shao, Z., Li, Z., Gao, Z., Liu, A., Xue, B., Wang, B., Wu, B., Feng, B., Lu, C., Zhao, C., Deng, C., Ruan, C., Dai, D., Chen, D., Ji, D., Li, E., Lin, F., Dai, F., Luo, F., Hao, G., Chen, G., Li, G., Zhang, H., Xu, H., Ding, H., Gao, H., Qu, H., Li, H., Guo, J., Li, J., Chen, J., Yuan, J., Tu, J., Qiu, J., Li, J., Cai, J. L., Ni, J., Liang, J., Chen, J., Dong, K., Hu, K., You, K., Gao, K., Guan, K., Huang, K., Yu, K., Wang, L., Zhang, L., Zhao, L., Wang, L., Zhang, L., Xu, L., Xia, L., Zhang, M., Zhang, M., Tang, M., Zhou, M., Li, M., Wang, M., Li, M., Tian, N., Huang, P., Zhang, P., Wang, Q., Chen, Q., Du, Q., Ge, R., Zhang, R., Pan, R., Wang, R., Chen, R. J., Jin, R. L., Chen, R., Lu, S., Zhou, S., Chen, S., Ye, S., Wang, S., Yu, S., Zhou, S., Pan, S., Li, S. S., Zhou, S., Wu, S., Yun, T., Pei, T., Sun, T., Wang, T., Zeng, W., Liu, W., Liang, W., Gao, W., Yu, W., Zhang, W., Xiao, W. L., An, W., Liu, X., Wang, X., Chen, X., Nie, X., Cheng, X., Liu, X., Xie, X., Liu, X., Yang, X., Li, X., Su, X., Lin, X., Li, X. Q., Jin, X., Shen, X., Chen, X., Sun, X., Wang, X., Song, X., Zhou, X., Wang, X., Shan, X., Li, Y. K., Wang, Y. Q., Wei, Y. X., Zhang, Y., Xu, Y., Li, Y., Zhao, Y., Sun, Y., Wang, Y., Yu, Y., Zhang, Y., Shi, Y., Xiong, Y., He, Y., Piao, Y., Wang, Y., Tan, Y., Ma, Y., Liu, Y., Guo, Y., Ou, Y., Wang, Y., Gong, Y., Zou, Y., He, Y., Xiong, Y., Luo, Y., You, Y., Liu, Y., Zhou, Y., Zhu, Y. X., Huang, Y., Li, Y., Zheng, Y., Zhu, Y., Ma, Y., Tang, Y., Zha, Y., Yan, Y., Ren, Z. Z., Ren, Z., Sha, Z., Fu, Z., Xu, Z., Xie, Z., Zhang, Z., Hao, Z., Ma, Z., Yan, Z., Wu, Z., Gu, Z., Zhu, Z., Liu, Z., Li, Z., Xie, Z., Song, Z., Pan, Z., Huang, Z., Xu, Z., Zhang, Z., and Zhang, Z. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, September 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

Han, S., Rao, K., Ettinger, A., Jiang, L., Lin, B. Y., Lambert, N., Choi, Y., and Dziri, N. Wildguard: Open one-stop moderation tools for safety risks, jailbreaks, and refusals of llms, 2024. URL https://arxiv.org/abs/2406.18495.

Hendrycks, D., Burns, C., Basart, S., Critch, A., Li, J., Song, D., and Steinhardt, J. Aligning ai with shared human values. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021a.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021b.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021c.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., de Laroussilhe, Q., Gesmundo, A., Attariyan, M., and

Gelly, S. Parameter-efficient transfer learning for nlp, 2019. URL https://arxiv.org/abs/1902.00751.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Hu, Z., Wang, L., Lan, Y., Xu, W., Lim, E.-P., Bing, L., Xu, X., Poria, S., and Lee, R. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 5254–5276, 2023.

Jiang, L., Rao, K., Han, S., Ettinger, A., Brahman, F., Kumar, S., Mireshghallah, N., Lu, X., Sap, M., Choi, Y., and Dziri, N. Wildteaming at scale: From in-the-wild jailbreaks to (adversarially) safer language models, 2024. URL https://arxiv.org/abs/2406.18510.

Jin, H., Luan, S., Lyu, S., Rabusseau, G., Rabbany, R., Precup, D., and Hamdaqa, M. Rl fine-tuning heals ood forgetting in sft. *arXiv preprint arXiv:2509.12235*, 2025a.

Jin, H., Luan, S., Lyu, S., Rabusseau, G., Rabbany, R., Precup, D., and Hamdaqa, M. Rl fine-tuning heals ood forgetting in sft, 2025b. URL https://arxiv.org/abs/2509.12235.

Kirk, R., Mediratta, I., Nalmpantis, C., Luketina, J., Hambro, E., Grefenstette, E., and Raileanu, R. Understanding the effects of rlhf on llm generalisation and diversity, 2024. URL https://arxiv.org/abs/2310.06452.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Klasson, M., Kjellström, H., and Zhang, C. Learn the time to learn: Replay scheduling in continual learning. *arXiv preprint arXiv:2209.08660*, 2022.

Kotha, S., Springer, J. M., and Raghunathan, A. Understanding catastrophic forgetting in language models via implicit inference, 2024. URL https://arxiv.org/abs/2309.10105.

Krawczyk, A. and Gepperth, A. An analysis of best-practice strategies for replay and rehearsal in continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4196–4204, 2024.

Lai, S., Zhao, H., Feng, R., Ma, C., Liu, W., Zhao, H., Lin, X., Yi, D., Zhang, Q., Liu, H., Meng, G., and Zhu, F. Reinforcement fine-tuning naturally mitigates forgetting in continual post-training, 2026. URL https://arxiv.org/abs/2507.05386.

Li, H., Ding, L., Fang, M., and Tao, D. Revisiting catastrophic forgetting in large language model tuning. In Al-Onaizan, Y., Bansal, M., and Chen, Y.-N. (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 4297–4308, Miami, Florida, USA, November 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.249. URL https://aclanthology.org/2024.findings-emnlp.249/.

Li, H., Ding, L., Fang, M., and Tao, D. Revisiting catastrophic forgetting in large language model tuning, 2024b. URL https://arxiv.org/abs/2406.04836.

Li, X. L. and Liang, P. Prefix-tuning: Optimizing continuous prompts for generation, 2021. URL https://arxiv.org/abs/2101.00190.

Li, Z. and Hoiem, D. Learning without forgetting, 2017. URL https://arxiv.org/abs/1606.09282.

Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Liu, H., Tam, D., Muqeeth, M., Mohta, J., Huang, T., Bansal, M., and Raffel, C. A. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35: 1950–1965, 2022.

Lopez-Paz, D. and Ranzato, M. Gradient episodic memory for continual learning, 2022. URL https://arxiv.org/abs/1706.08840.

Lu, W., Luu, R. K., and Buehler, M. J. Fine-tuning large language models for domain adaptation: Exploration of training strategies, scaling, model merging and synergistic capabilities. *npj Computational Materials*, 11(1):84, 2025.

Luo, R., Sun, L., Xia, Y., Qin, T., Zhang, S., Poon, H., and Liu, T.-Y. Biogpt: generative pre-trained transformer for biomedical text generation and mining. *Briefings in Bioinformatics*, 23(6), September 2022. ISSN 1477-4054. doi: 10.1093/bib/bbac409. URL http://dx.doi.org/10.1093/bib/bbac409.

Luo, Y., Yang, Z., Meng, F., Li, Y., Zhou, J., and Zhang, Y. An empirical study of catastrophic forgetting in large language models during continual fine-tuning, 2025. URL https://arxiv.org/abs/2308.08747.

Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., Gupta, S., Majumder, B. P., Hermann, K., Welleck, S., Yazdanbakhsh, A., and Clark, P. Self-refine: Iterative refinement with self-feedback, 2023. URL https://arxiv.org/abs/2303.17651.

Mathew, M., Karatzas, D., and Jawahar, C. Docvqa: A dataset for vqa on document images. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 2200–2209, 2021.

McCloskey, M. and Cohen, N. J. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. Elsevier, 1989.

Mukherjee, S., Mitra, A., Jawahar, G., Agarwal, S., Palangi, H., and Awadallah, A. Orca: Progressive learning from complex explanation traces of gpt-4, 2023. URL https://arxiv.org/abs/2306.02707.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., and Lowe, R. Training language models to follow instructions with human feedback, 2022. URL https://arxiv.org/abs/2203.02155.

Pan, J., Zhang, J., Wang, X., Yuan, L., Peng, H., and Suhr, A. Tinyzero. https://github.com/Jiayi-Pan/TinyZero, 2025. Accessed: 2025-01-24.

Perez, E., Kiela, D., and Cho, K. True few-shot learning with language models, 2021. URL https://arxiv.org/abs/2105.11447.

Rafailov, R., Sharma, A., Mitchell, E., Manning, C. D., Ermon, S., and Finn, C. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36: 53728–53741, 2023.

Rebuffi, S.-A., Kolesnikov, A., Sperl, G., and Lampert, C. H. icarl: Incremental classifier and representation learning, 2017. URL https://arxiv.org/abs/1611.07725.

Ross, S., Gordon, G. J., and Bagnell, J. A. A reduction of imitation learning and structured prediction to no-regret online learning, 2011. URL https://arxiv.org/abs/1011.0686.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Sclar, M., Choi, Y., Tsvetkov, Y., and Suhr, A. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting, 2024. URL https://arxiv.org/abs/2310.11324.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y. K., Wu, Y., and Guo, D. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Shenfeld, I., Pari, J., and Agrawal, P. Rl's razor: Why online reinforcement learning forgets less. *arXiv preprint arXiv:2509.04259*, 2025.

Singh, A., Natarjan, V., Shah, M., Jiang, Y., Chen, X., Batra, D., Parikh, D., and Rohrbach, M. Towards vqa models that can read. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8317–8326, 2019.

Tang, Q., Deng, Z., Lin, H., Han, X., Liang, Q., Cao, B., and Sun, L. Toolalpaca: Generalized tool learning for language models with 3000 simulated cases. *arXiv preprint arXiv:2306.05301*, 2023.

Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., and Zhou, D. Self-consistency improves chain of thought reasoning in language models, 2023. URL https://arxiv.org/abs/2203.11171.

Wang, Y., Ma, X., Zhang, G., Ni, Y., Chandra, A., Guo, S., Ren, W., Arulraj, A., He, X., Jiang, Z., et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290, 2024.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., and Zhou, D. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

Xie, Q., Chen, Q., Chen, A., Peng, C., Hu, Y., Lin, F., Peng, X., Huang, J., Zhang, J., Keloth, V., et al. Medllama: Foundation large language models for medical applications. *Research square*, pp. rs–3, 2024.

Yan, J., Li, Y., Hu, Z., Wang, Z., Cui, G., Qu, X., Cheng, Y., and Zhang, Y. Learning to reason under off-policy guidance. *arXiv preprint arXiv:2504.14945*, 2025.

Yang, Q. A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Dong, G., Wei, H., Lin, H., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Lin, J., Dang, K., Lu, K., Bao,

K., Yang, K., Yu, L., Li, M., Xue, M., Zhang, P., Zhu, Q., Men, R., Lin, R., Li, T., Xia, T., Ren, X., Ren, X., Fan, Y., Su, Y., Zhang, Y.-C., Wan, Y., Liu, Y., Cui, Z., Zhang, Z., Qiu, Z., Quan, S., and Wang, Z. Qwen2.5 technical report. *ArXiv*, abs/2412.15115, 2024. URL https://api.semanticscholar.org/CorpusID:274859421.

Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W. W., Salakhutdinov, R., and Manning, C. D. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Yuan, W., Pang, R. Y., Cho, K., Li, X., Sukhbaatar, S., Xu, J., and Weston, J. Self-rewarding language models, 2025. URL https://arxiv.org/abs/2401.10020.

Zelikman, E., Wu, Y., Mu, J., and Goodman, N. D. Star: Bootstrapping reasoning with reasoning, 2022. URL https://arxiv.org/abs/2203.14465.

Zelikman, E., Harik, G., Shao, Y., Jayasiri, V., Haber, N., and Goodman, N. D. Quiet-star: Language models can teach themselves to think before speaking, 2024. URL https://arxiv.org/abs/2403.09629.

Zenke, F., Poole, B., and Ganguli, S. Continual learning through synaptic intelligence, 2017. URL https://arxiv.org/abs/1703.04200.

Zhang, H., Wu, Y., Li, D., Yang, S., Zhao, R., Jiang, Y., and Tan, F. Balancing speciality and versatility: A coarse to fine framework for mitigating catastrophic forgetting in large language models, 2025. URL https://arxiv.org/abs/2404.10306.

Zhang, Z., Lei, L., Wu, L., Sun, R., Huang, Y., Long, C., Liu, X., Lei, X., Tang, J., and Huang, M. Safetybench: Evaluating the safety of large language models with multiple choice questions. *arXiv preprint arXiv:2309.07045*, 2023.

Zhou, J., Lu, T., Mishra, S., Brahma, S., Basu, S., Luan, Y., Zhou, D., and Hou, L. Instruction-following evaluation for large language models, 2023. URL https://arxiv.org/abs/2311.07911.

Appendix Contents

A. LLM Usage Disclosure

To enhance clarity and readability, we utilized LLMs (specifically OpenAI GPT-5) exclusively as a language polishing tool. Its role was confined to proofreading, grammatical correction, and stylistic refinement: functions analogous to those provided by traditional grammar checkers and dictionaries. This tool did not contribute to the generation of new scientific content or ideas, and its usage is consistent with standard practices for manuscript preparation.

B. Extended Related Work

Post-Training Dynamics and Forgetting (Extended). Prior work has analyzed how neural optimization trajectories reshape the loss landscape and contribute to forgetting in sequential training regimes (Goodfellow et al., 2015; McCloskey & Cohen, 1989). To mitigate drift and forgetting, existing approaches broadly fall into several categories. Regularization-based methods constrain parameter updates to preserve previously learned behaviors, e.g., by penalizing deviations from important weights or reference representations, trading off plasticity for stability (Kirkpatrick et al., 2017; Zenke et al., 2017; Aljundi et al., 2018; Li & Hoiem, 2017; Li et al., 2024a). Replay and rehearsal strategies maintain a small memory of past data (or synthetic samples) and interleave them during continued post-training to prevent collapse on earlier distributions (Lopez-Paz & Ranzato, 2022; Rebuffi et al., 2017; Krawczyk & Gepperth, 2024; Klasson et al., 2022). Parameter-efficient adaptation limits the effective degrees of freedom of post-training, reducing interference with pre-trained knowledge while still enabling task adaptation (Houlsby et al., 2019; Li & Liang, 2021; Hu et al., 2021; Liu et al., 2022; Hu et al., 2023). Together, these lines of work suggest that controlling which data drives post-training and how strongly updates reshape the model are both critical for maintaining capability retention under continual optimization.

Preference Optimization and On-Policy Alignment (Extended). Follow-up work further studies continual post-training and reports that reinforcement fine-tuning can mitigate forgetting over time, while RL fine-tuning can also alleviate OOD forgetting introduced by SFT and improve robustness under distribution shift (Lai et al., 2026; Jin et al., 2025b). Beyond RL and SFT, recent analyses emphasize distributional considerations in reasoning and alignment, showing that matching the training data distribution to a model's own generation distribution can be an important driver of stable improvements (Chandra et al., 2025; Cheng et al., 2025). While these findings highlight the benefits of on-policy training, existing approaches rely on explicit rewards, verifiers, or preference signals.

Trajectory Learning and Data Selection Curricula (Extended). Recent work increasingly leverages model-generated trajectories including rollouts (Wang et al., 2023), intermediate reasoning traces (Zelikman et al., 2022; Guo et al., 2025), and self-evaluations as training signals (Yuan et al., 2025; Bai et al., 2022b), enabling self-distillation (Mukherjee et al., 2023; Agarwal et al., 2024) and iterative refinement without additional labels (Madaan et al., 2023; Zelikman et al., 2024). Beyond RL, analyses of reasoning post-training emphasize distributional considerations, showing that matching the training distribution to a model's own generation distribution can be a key driver of stable improvements, sometimes even more influential than strict correctness of supervision (Chandra et al., 2025; Cheng et al., 2025). These findings naturally connect trajectory learning to dynamic data selection and curriculum learning, where trajectories provide feedback about the learner's evolving state and motivate adaptive sampling strategies that improve robustness and sample efficiency over static training sets (Cheng et al., 2025; Chu et al., 2025).

C. TMS Algorithm

We provide the full Trajectory-Mixed Supervision (TMS) procedure for transparency and reproducibility. TMS turns a *single* baseline post-training trajectory into a *distribution* over supervision targets. Concretely, instead of training a student against a fixed reference y^* (which can induce supervision-mismatch drift and mode collapse), we harvest intermediate policy outputs along the training path and sample targets from their mixture, yielding near-policy supervision that better preserves diverse solution modes (Figure 1).

Inputs and outputs. TMS takes as input a base policy π_{θ_0} and a supervised dataset $\mathcal{D} = \{(x_i, y_i^*)\}_{i=1}^N$. It outputs a post-trained student policy π_ϕ trained on *trajectory-mixed* targets. The hyperparameter T controls how many checkpoints are used (trajectory resolution), and $\alpha \in [0, 1]$ controls how often we fall back to oracle/teacher labels (quality anchoring).

Two-stage procedure. TMS is implemented in two stages. **Stage 1** harvests a trajectory buffer by running a baseline post-training procedure (typically SFT, but any deterministic update path can be used) and saving T checkpoints $\{\theta_1, \ldots, \theta_T\}$.

At each checkpoint t, we generate one (or a small number of) sampled outputs $\hat{y}^{(t)}(x)$ for each prompt x (or a representative subset) and store them in a buffer $\mathcal{H}[t, x]$. **Stage 2** trains a student initialized from θ_0 using standard token-level NLL, but with targets drawn from a mixture: with probability α we train on the reference label y^*; otherwise we sample a checkpoint index $t \sim p(t)$ (uniform in Algorithm 1 unless otherwise specified) and train on the corresponding trajectory target $\mathcal{H}[t, x]$.

Practical notes. (i) The checkpoint distribution $p(t)$ can be uniform (default) or non-uniform (e.g., early-/mid-/late-heavy) to test "window" effects; (ii) storing text sequences is sufficient for our experiments and is cheaper than storing logits; and (iii) to control compute, trajectory harvesting can be performed on a subset of prompts without changing the training objective.

Algorithm 1 Trajectory-Mixed Supervision

Input: Base policy π_{θ_0}; dataset $\mathcal{D} = \{(x_i, y_i^*)\}_{i=1}^N$; checkpoints T; mixing weight α.
Stage 1: Harvest trajectory outputs.
Run baseline post-training (e.g., SFT) and save checkpoints $\{\theta_1, \ldots, \theta_T\}$.
for $t = 1$ to T **do**
  for each $x \in \mathcal{D}_x$ (or subset) **do**
    Generate $\hat{y}^{(t)}(x) \sim \pi_{\theta_t}(\cdot \mid x)$; store in $\mathcal{H}[t, x]$.
  end for
end for
Stage 2: Train student on trajectory-mixed supervision.
Initialize student $\phi \leftarrow \theta_0$.
while training budget not exhausted **do**
  Sample minibatch x from $\mathcal{D}_x$.
  for each x in minibatch **do**
    With probability α, set $\tilde{y} \leftarrow y^*$.
      Else sample $t \sim \mathrm{Uniform}(\{1, \ldots, T\})$ and set $\tilde{y} \leftarrow \mathcal{H}[t, x]$.
    Add $(x, \tilde{y})$ to B_{mix}.
  end for
  Update ϕ by minimizing token-level NLL on B_{mix}.
end while
Output: Student policy π_ϕ.

D. A Unified View Through Mismatch and Modes

Our analysis in Section 4 suggests that post-training instability is not a single phenomenon. Rather, it arises from two orthogonal mechanisms: (i) *supervision mismatch drift* when a moving policy is trained against static supervision, and (ii) *mode collapse* when single-reference cross-entropy suppresses alternative valid solutions. This section provides a unified interpretation of TMS as an intervention that addresses both mechanisms using a single principle: **replace rigid supervision with near-policy, trajectory-supported supervision**.

(A) Reducing supervision mismatch drift via near-policy targets. Failure Mode A arises because SFT optimizes π_{θ_t} against a fixed supervision distribution $q(\cdot|x)$; as π_{θ_t} evolves, the mismatch between the current policy and the static targets can grow, which manifests as rising held-out PLD and increasingly corrective gradients. TMS replaces the static target distribution with a *trajectory mixture* $m(\cdot|x)$ whose support is formed by samples from intermediate policies along the same optimization path. Since each component policy π_{θ_t} is itself a feasible model distribution, targets drawn from m are *reachable* throughout training. Empirically, this reduces held-out PLD and mitigates the late-stage mismatch drift observed in Table 1.

(B) Preventing mode collapse by preserving trajectory support. Failure Mode B occurs when single-reference cross-entropy concentrates probability mass on one trajectory y^* and implicitly penalizes other valid trajectories. In contrast, the mixture $m(\cdot|x)$ aggregates outputs from checkpoints spanning early (higher-entropy, diverse) to late (more accurate) policies. Training on this mixture encourages the student to assign non-trivial probability to multiple historically plausible solution modes, improving coverage and diversity on non-unique tasks. Consistent with this view, TMS maintains higher Pass@K and answer entropy than single-reference SFT (Table 2), acting as a *mode-preserving regularizer* derived from the model's own learning dynamics.

(C) Relation to RLFT without rewards. On-policy RLFT is often more stable because training targets are generated from the current policy (on-policy in $y|x$), keeping updates near-policy. TMS achieves a similar effect *without* rewards or value estimation: the trajectory mixture m can be interpreted as a time-averaged approximation of on-policy supervision, where targets are sampled from policies that the model actually inhabited during training. This provides a reward-free route to near-policy learning and helps explain why TMS improves the accuracy–retention tradeoff relative to static-label SFT.

E. Proof of Theorem 6.2

Assumptions.

- **Assumption 1 (Bounded scoring function).** For each prompt x, the held-out task defines a scoring function $f_x : \mathcal{Y} \to [0, 1]$ (e.g., correctness indicator, normalized reward).
- **Assumption 2 (Evaluation prompt distribution).** Prompts are drawn from a fixed distribution $x \sim \mathcal{D}_{\text{eval}}$ (e.g., prompts from the retention suite).

Theorem 6.2 (Forgetting is controlled by KL drift). Let $\pi_0(\cdot|x)$ be the base policy and $\pi_\theta(\cdot|x)$ a fine-tuned policy. Then the change in expected score on $\mathcal{D}_{\text{eval}}$ satisfies

$$\left|\mathbb{E}_{x\sim\mathcal{D}_{\text{eval}}}\left[\mathbb{E}_{y\sim\pi_\theta(\cdot|x)} f_x(y)\right] - \mathbb{E}_{x\sim\mathcal{D}_{\text{eval}}}\left[\mathbb{E}_{y\sim\pi_0(\cdot|x)} f_x(y)\right]\right| \le \mathbb{E}_{x\sim\mathcal{D}_{\text{eval}}}\left[\sqrt{2\, D_{\text{KL}}(\pi_\theta(\cdot|x) \,\|\, \pi_0(\cdot|x))}\right].$$

Proof. Fix an arbitrary prompt x. Define the expected score under each policy:

$$\mu_\theta(x) := \mathbb{E}_{y\sim\pi_\theta(\cdot|x)}[f_x(y)], \qquad \mu_0(x) := \mathbb{E}_{y\sim\pi_0(\cdot|x)}[f_x(y)].$$

We bound $|\mu_\theta(x) - \mu_0(x)|$ in three steps.

Step 1: Expand the difference as an inner product.

$$\mu_\theta(x) - \mu_0(x) = \sum_{y\in\mathcal{Y}} f_x(y)\big(\pi_\theta(y|x) - \pi_0(y|x)\big).$$

Step 2: Apply L_1/L_∞ duality (Hölder). Taking absolute values and using $0 \le f_x(y) \le 1$ (Assumption 1),

$$|\mu_\theta(x) - \mu_0(x)| \le \sum_{y\in\mathcal{Y}} |f_x(y)|\,|\pi_\theta(y|x) - \pi_0(y|x)| \le \sum_{y\in\mathcal{Y}} |\pi_\theta(y|x) - \pi_0(y|x)|.$$

Recall total variation distance:

$$d_{\text{TV}}(P, Q) := \frac{1}{2}\sum_{y\in\mathcal{Y}} |P(y) - Q(y)|.$$

Therefore,

$$|\mu_\theta(x) - \mu_0(x)| \le 2\, d_{\text{TV}}(\pi_\theta(\cdot|x),\, \pi_0(\cdot|x))\,.$$

Step 3: Convert TV to KL via Pinsker's inequality. Pinsker's inequality states

$$d_{\text{TV}}(P, Q) \le \sqrt{\frac{1}{2} D_{\text{KL}}(P\|Q)}.$$

Applying this with $P = \pi_\theta(\cdot|x)$ and $Q = \pi_0(\cdot|x)$ yields

$$|\mu_\theta(x) - \mu_0(x)| \le 2\sqrt{\frac{1}{2} D_{\text{KL}}(\pi_\theta(\cdot|x) \,\|\, \pi_0(\cdot|x))} = \sqrt{2\, D_{\text{KL}}(\pi_\theta(\cdot|x) \,\|\, \pi_0(\cdot|x))}.$$

Step 4: Take expectation over prompts. Taking expectation over $x \sim \mathcal{D}_{\text{eval}}$ (Assumption 2) and using linearity of expectation,

$$\left|\mathbb{E}_x[\mu_\theta(x)] - \mathbb{E}_x[\mu_0(x)]\right| \le \mathbb{E}_x\left[|\mu_\theta(x) - \mu_0(x)|\right] \le \mathbb{E}_{x\sim\mathcal{D}_{\text{eval}}}\left[\sqrt{2\, D_{\text{KL}}(\pi_\theta(\cdot|x) \,\|\, \pi_0(\cdot|x))}\right].$$

This completes the proof. ■

Corollary E.1. *Under the assumptions of Theorem 6.2,*

$$\left|\mathbb{E}_x[\mu_\theta(x)] - \mathbb{E}_x[\mu_0(x)]\right| \le \sqrt{2\,\mathbb{E}_{x\sim\mathcal{D}_{\text{eval}}}[D_{\text{KL}}(\pi_\theta(\cdot|x)\,\|\,\pi_0(\cdot|x))]}.$$

Proof. Apply Jensen's inequality to $\sqrt{\cdot}$, which is concave. ■

F. Experimental Setup Details

This appendix provides dataset-level details, baseline implementation specifics, and training/inference hyperparameters for reproducibility. Definitions of the core evaluation axes (target, cross-task transfer, retention/forgetting) and aggregate metrics are given in the main paper (§7 and §3); here we describe how they are implemented for each benchmark.

F.1. Dataset and Benchmark Details

Target tasks (used for training).

- **GSM8K.** (Cobbe et al., 2021) 7.5K train / 1K test grade-school math word problems. We fine-tune on the official train split and evaluate on the official test split.
- **MATH.** (Hendrycks et al., 2021c) Competition-level math problems. We fine-tune on the official training split and evaluate on the official test split.
- **Math-500.** (Lightman et al., 2023) A 500-problem subset used for fast evaluation and tuning (reported as a target task when used for training).
- **Countdown.** (Pan et al., 2025) Synthetic arithmetic task (reach target number with given numbers/operators). We generate 10K training examples and a disjoint held-out set using a fixed generator seed.
- **IFEval.** (Zhou et al., 2023) 541 prompts with verifiable instruction constraints; we report strict constraint satisfaction using the official checker.
- **MMLU.** (Hendrycks et al., 2021a;b) 57 subjects; we report standard evaluation accuracy and use an auxiliary training split when fine-tuning on MMLU.
- **ToolAlpaca.** (Tang et al., 2023) A tool-use instruction-following dataset where models must invoke external tools (e.g., calculator/search) and produce multi-step tool-augmented responses. We fine-tune on the provided training split and evaluate on the standard held-out prompts using the ToolAlpaca evaluation protocol.
- **TextVQA.** (Singh et al., 2019) A multimodal visual question answering benchmark focused on reading and reasoning over text in images. We fine-tune on the official training split and evaluate on the standard test split using exact-match accuracy.
- **DocVQA.** (Mathew et al., 2021) A document visual question answering benchmark requiring understanding of scanned documents (layout + OCR text). We fine-tune on the official validation split and evaluate on the standard test split using exact-match accuracy.

Held-out retention suite (never trained on).

- **ARC-Challenge.** (Clark et al., 2018) Commonsense/science multiple-choice benchmark.
- **HotpotQA.** (Yang et al., 2018) Multi-hop question answering; we use the standard evaluation setting (distraction).
- **WildGuardTest / SafetyBench / WildJailbreak.** (Han et al., 2024; Zhang et al., 2023; Jiang et al., 2024) Safety and jailbreak robustness benchmarks; we report refusal/violation rates using benchmark-provided evaluators or a fixed judge model (specified in Section F.5).

Table 4. **Hyperparameters for full fine-tuning.**

Model	LR	BS/dev	Grad Acc.	Eff. BS
LLaMA-3.2-1B	1×10^{-4}	4	8	256
Qwen-2.5-1.5B	1×10^{-4}	4	8	256
Qwen-2.5-3B	2×10^{-5}	2	16	256
Qwen-2.5-7B	5×10^{-6}	1	32	256
LLaMA-3.1-8B	5×10^{-6}	1	32	256

Table 5. **Inference tensor parallelism (TP) settings.**

Model	TP
LLaMA-3.2-1B	8
Qwen-2.5-1.5B	4
Qwen-2.5-3B	8
Qwen-2.5-7B	4
LLaMA-3.1-8B	8

F.2. Baseline Implementations

Standard SFT. We minimize token-level NLL on reference completions using the same optimizer, schedule, and training budget as TMS.

Self-SFT / Final-SFT. We generate one completion per prompt with a fixed decoding policy (temperature/top-p; Section F.3), then fine-tune on the resulting synthetic dataset using standard NLL. Self-SFT uses outputs from π_{θ_0}; Final-SFT uses outputs from the converged SFT model.

GRPO / REINFORCE. For verifiable tasks (math and Countdown), we use outcome rewards $r(x, y) \in \{0, 1\}$ based on final-answer correctness (exact match or symbolic equivalence). For others, we use LLama-3.3-70B to give a numerical score. GRPO uses group-normalized advantages without a value network; REINFORCE uses a moving-average baseline. All RL runs use the same prompt distribution as SFT/TMS and are matched for total training budget.

F.3. Decoding and Sampling Policies

Trajectory harvesting (TMS). For each checkpoint t and input x, we generate $\hat{y}^{(t)}(x) \sim \pi_{\theta_t}(\cdot|x)$ using a fixed sampling policy (temperature/top-p), max token limit, and stop rules. Unless otherwise stated, we store one sampled output per (x, t).

Pass@K and diversity proxies. To compute Pass@K, majority-vote self-consistency accuracy (SC-Acc), and answer entropy, we draw K independent samples per prompt under the same decoding policy for all methods. We report K and decoding hyperparameters alongside results to ensure comparability. This metric is used in Section 4.

F.4. Training Hyperparameters and Compute

We use AdamW with a cosine learning rate schedule (3% warmup) and BF16 precision. We train all methods for 2 epochs. Table 4 lists hyperparameters for full fine-tuning.

Compute environment. All experiments run on 8× NVIDIA A100 (80GB) GPUs. We use `accelerate` for distributed training and `vLLM` for high-throughput inference and trajectory harvesting.

F.5. Metric Implementation Details

Answer extraction and verification (math). For GSM8K/MATH/Math-500/Countdown, we extract a final answer using a deterministic parser (e.g., `\boxed{}` if present; otherwise the last numeric expression) and verify correctness via exact match or symbolic equivalence using `math-verify`[2]. For Countdown, we additionally verify that the expression uses only permitted numbers/operators.

[2]Math-verify: https://github.com/huggingface/Math-Verify

Table 6. **RQ8 (Alignment & Safety Retention).** Held-out safety evaluation. We report **SafetyBench** (↑) and attack success rates (ASR; ↓) on **WildGuardTest** and **WildJailBreak**. Best/second-best are computed *per model* for each metric.

Model	Method	SafetyBench↑	WildGuard ASR↓	WildJailBreak ASR↓
LLaMA-3.2-1B	Standard SFT	21.8	38.6	52.4
	GRPO	**25.0**	**32.0**	**41.4**
	TMS (Ours)	24.6	32.3	42.2
Qwen-2.5-3B	Standard SFT	31.1	61.3	89.4
	GRPO	**35.4**	54.8	83.7
	TMS (Ours)	35.0	**54.2**	**82.9**
Qwen-2.5-7B	Standard SFT	34.1	43.8	74.2
	GRPO	**38.6**	**36.6**	**65.2**
	TMS (Ours)	37.1	37.1	65.9

IFEval. We report strict prompt-level constraint satisfaction using the official IFEval checker.

Helpfulness and safety. For WildGuardTest/WildJailbreak we report refusal/violation rates using benchmark-provided evaluators. For SafetyBench, we calculate the Accuracy.

KL-to-base and PLD proxies. KL-to-base is computed as an expectation over evaluation prompts of the KL between $\pi_\theta(\cdot|x)$ and $\pi_{\theta_0}(\cdot|x)$ using token log-probabilities. Held-out PLD is computed as validation NLL on a held-out split of the training task, as described in Section 4.

G. Additional Experiments

We present targeted ablations to probe *which aspects of trajectory mixing matter* and whether TMS extends beyond standard single-turn text benchmarks.

RQ8 (Alignment and safety retention). *Does TMS better preserve helpfulness and safety behavior than SFT on held-out alignment/safety benchmarks?*

Setup (held-out safety retention). To isolate safety *retention* rather than safety *learning*, we evaluate all post-trained models on safety benchmarks that are **never used for post-training**. We report (i) **SafetyBench** (↑), a safety QA benchmark scored by accuracy-style criteria, and (ii) jailbreak **attack success rates** (ASR; ↓) on **WildGuardTest** and **WildJailBreak**. Lower ASR indicates stronger refusal/robustness to adversarial prompts. We compare **Standard SFT**, **GRPO**, and **TMS** under matched training budgets; best/second-best are computed per model and metric (Table 6).

Safety retention: SFT erodes safety, while TMS recovers most of the loss. Across all three models, standard SFT yields noticeably weaker safety behavior, increasing ASR and reducing SafetyBench relative to the stronger baselines. In contrast, TMS consistently reduces ASR and improves SafetyBench compared to SFT, closely tracking the on-policy RL baseline. For example, on **Qwen-2.5-3B**, TMS slightly improves over GRPO on both ASR metrics (WildGuard: 54.2 vs. 54.8; WildJailBreak: 82.9 vs. 83.7) while remaining within a small margin on SafetyBench (35.0 vs. 35.4). On **Qwen-2.5-7B**, TMS again stays close to GRPO (WildJailBreak ASR: 65.9 vs. 65.2), and on **LLaMA-3.2-1B** it recovers a substantial fraction of SFT-induced degradation (e.g., WildJailBreak ASR: 42.2 vs. 52.4 for SFT).

Interpretation. These results support the view that safety regressions under continued post-training are a form of *capability drift*: static-label SFT can over-specialize and inadvertently erode previously acquired guardrails. By keeping supervision closer to the model's evolving support, TMS reduces destructive updates and thereby preserves safety behavior that is otherwise fragile under standard SFT.

RQ9 (Window vs uniform). *Which parts of the training trajectory matter most: does concentrating TMS on an intermediate "window" outperform uniform sampling for retention at matched target accuracy?*

Table 7. **RQ9: Sampling distribution for TMS** (Qwen-2.5-3B, T=10).

Sampling	Avg Target ↑	Cross-task ↓
Early	64.7	4.5
Uniform	70.8	2.3
Late	71.9	2.0

Setup (sampling distributions over checkpoints). We instantiate TMS with T=10 equally-spaced checkpoints along the baseline post-training trajectory (SFT on the same task), saved at fixed step intervals from the start to the end of training.[3] For each input x, we store one sampled trajectory output $\hat{y}^{(t)}(x)$ per checkpoint $t \in \{1, \dots, T\}$. During student training, each synthetic target is selected by first sampling a checkpoint index $t \sim p(t)$, then setting $\tilde{y} \leftarrow \hat{y}^{(t)}(x)$ (with the same decoding settings across variants). We compare the following distributions $p(t)$:

- **Early-heavy:** $p(t) = \frac{1}{\lfloor T/3 \rfloor} \mathbb{1}\{t \le \lfloor T/3 \rfloor\}$ (uniform over the first $\approx 30\%$ of checkpoints).
- **Late-heavy:** $p(t) = \frac{1}{T - \lceil 2T/3 \rceil + 1} \mathbb{1}\{t \ge \lceil 2T/3 \rceil\}$ (uniform over the last $\approx 30\%$ of checkpoints).
- **Uniform:** $p(t) = \frac{1}{T}$ (uniform over all checkpoints).

These choices directly test the "window" hypothesis suggested by mismatch drift: early checkpoints emphasize diversity but are noisier, while late checkpoints emphasize correctness but may reflect over-committed solution modes.

Windowing matters: early checkpoints are too noisy, late checkpoints can over-specialize. Table 7 reports results on Qwen-2.5-3B. Sampling from *early* checkpoints substantially degrades target performance (64.7 vs. 70.8 for uniform), consistent with injecting supervision before the model has learned task format and basic competence. This variant also increases cross-task drop (4.5), suggesting that overly noisy supervision can act as an unstable training signal rather than a regularizer. In contrast, *late*-heavy sampling yields the highest target average (71.9) and slightly improves cross-task drop (2.0). However, in retention-focused runs (measured on the held-out suite), late-heavy mixtures can re-introduce the same brittleness we observe in late-stage SFT: the supervision concentrates around a narrow set of converged trajectories, weakening the mode-preservation benefit and partially reviving mismatch drift (Section 4). Uniform sampling provides a robust default (70.8 target avg, 2.3 cross-task) that balances competence (late checkpoints) and diversity (early/mid checkpoints).

RQ10 (Effect of T on supervision quality). *Does increasing the number of checkpoints T improve TMS by capturing more diverse solution modes, or hurt by injecting early noisy supervision, and where is the optimal T?*

Setup (sweeping the number of checkpoints). We sweep the number of trajectory checkpoints T while holding the *total post-training compute* fixed. Concretely, for each $T \in \{2, 4, 6, 8, 10, 12\}$, we (i) save T equally-spaced checkpoints along the same underlying post-training trajectory (Qwen-2.5-3B on the same target task), (ii) harvest one sampled completion per training example per checkpoint using the same decoding policy, and (iii) train a TMS student under identical optimizer settings and total update steps. Unless otherwise stated, we use **uniform checkpoint sampling** $p(t) = 1/T$ and the same mixture procedure as in Section 5. The reported metrics are Avg Target (mean across the six target tasks) and Cross-task (Avg Δ on the other target tasks), matching the main-table definitions.

Moderate T is best: diminishing returns past the mid-range. Figure 4 summarizes the sweep. Increasing T from small values initially improves Avg Target and reduces cross-task interference, consistent with a richer supervision mixture that (i) better covers alternative valid modes (reducing mode collapse) and (ii) keeps supervision closer to the model's evolving support (reducing mismatch-driven updates). However, beyond a moderate range, gains saturate and variability increases: adding more checkpoints begins to incorporate (a) very early snapshots whose outputs are noisy and format-inconsistent, and (b) very late snapshots that may be over-specialized. Both effects reduce the marginal benefit of additional checkpoints.

In our setting, $T \in [8, 10]$ provides a stable sweet spot for the accuracy–transfer tradeoff. This range captures enough of the trajectory to preserve diversity while avoiding excessive exposure to the lowest-quality early supervision. We therefore use T=10 as the default throughout the paper unless otherwise noted.

[3] All variants use the same total training budget and the same trajectory buffer; only the checkpoint sampling distribution changes.

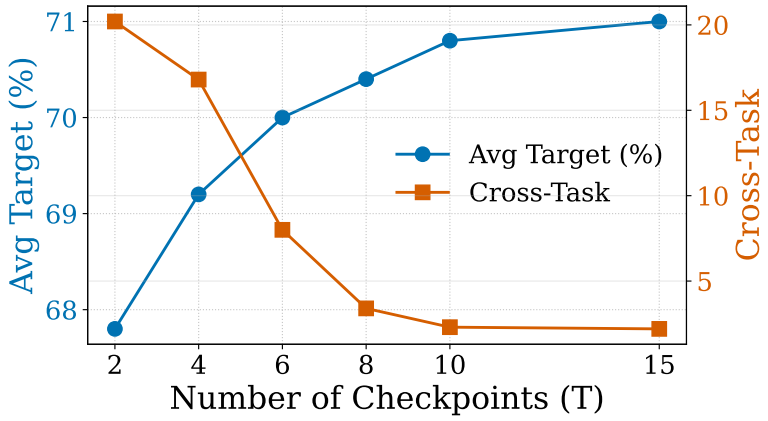

Figure 4. **RQ10: Effect of the number of checkpoints** T (Qwen-2.5-3B). We report Avg Target and Cross-task.

Table 8. **RQ11–12: Agentic and multimodal pilots** (ToolAlpaca on Qwen-2.5-7B; TextVQA/DocVQA on Qwen-2.5-Omni-3B).

Method	ToolAlpaca	Multimodal		Cross-task↓
		TextVQA	DocVQA	
Base	48.6	79.8	93.3	–
Standard SFT	66.2	85.1	96.2	29.4
GRPO	**68.4**	85.0	**96.3**	**1.8**
TMS (Ours)	67.9	**85.2**	96.2	2.6

RQ11 (Multimodal feasibility). *Does trajectory-mixed supervision extend to a multimodal setting, improving retention and reducing drift relative to standard SFT in a single-model pilot?*

RQ12 (Agentic feasibility). *Does TMS extend to a small multi-step tool-use setting, preserving the same KL/PLD–forgetting trends observed in single-turn tasks?*

Setup. To test feasibility beyond single-turn text benchmarks, we run two small pilots under the same evaluation lens used throughout the paper: target performance on the pilot task(s) and cross-task interference to the remaining downstream targets. **Agentic pilot:** we fine-tune **Qwen-2.5-7B** on **ToolAlpaca**, a tool-use dataset requiring multi-step tool-augmented responses. **Multimodal pilot:** we fine-tune **Qwen-2.5-Omni-3B** on **TextVQA** and **DocVQA**, which require answering questions about images containing text and scanned documents, respectively. For SFT and TMS, we use the same supervised objective as in Section 5; for GRPO, we use outcome-based rewards when verifiable signals exist and otherwise rely on the benchmark evaluation protocol. Cross-task interference is reported as the average Δ on the remaining target tasks (defined in Section 7), keeping the measurement consistent with the main results.

TMS extends to multimodal and tool-use pilots while remaining stable. Table 8 summarizes the pilot results. Standard SFT substantially improves the agentic and multimodal targets relative to zero-shot but incurs large cross-task interference (29.4), suggesting brittle specialization. In contrast, GRPO achieves minimal interference (1.8), consistent with the general trend that on-policy optimization tends to stay in a low-drift regime. TMS closely tracks this RL-like behavior: it achieves low cross-task interference (2.6) while remaining competitive on both tool-use and multimodal targets.

On the agentic benchmark, TMS nearly matches GRPO (67.9 vs. 68.4 on ToolAlpaca). On multimodal tasks, TMS matches or slightly improves over SFT (TextVQA: 85.2 vs. 85.1) while remaining comparable to GRPO. Taken together, these pilots suggest that trajectory-mixed supervision is not confined to single-turn text tasks; the same "near-policy via trajectory" principle transfers to multi-step tool-use and multimodal post-training, and continues to reduce cross-task interference in the same direction observed in Table 3.

H. Full Experimental Results

Table 9. **Full results across models.** Target-task performance S_{tgt} (higher is better), cross-task transfer to other target tasks S_{xfer} (Avg Δ, lower is better), and held-out retention on ARC-C/HotpotQA/SafetyBench. Parentheses denote change vs. the base model; **best** / second-best are highlighted.

Method	Target Tasks S_{tgt} ↑						Cross-task S_{xfer} ↓	Held-out Retention ↑		
	Math500	MATH	GSM8K	Count.	IFEval	MMLU	Avg Δ other tgt	ARC-C	Hotpot	SafetyBench
Qwen-2.5-1.5B-Instruct										
Base	46.6	48.1	68.9	13.2	38.3	30.8	–	66.9	43.1	35.1
Standard SFT	58.4	62.4	77.3	25.2	54.1	**39.0**	↓ 26.2	42.3(↓ 24.6)	19.4(↓ 23.7)	31.4(↓ 3.7)
Self-SFT	54.9	57.1	73.1	29.4	47.3	34.7	↓ 18.4	51.4(↓ 15.5)	27.5(↓ 15.6)	34.3(↓ 0.8)
Final-SFT	57.1	61.9	**78.0**	25.1	53.1	36.8	↓ 24.3	48.3(↓ 18.6)	25.3(↓ 17.8)	34.0(↓ 1.1)
REINFORCE	56.9	60.1	75.2	26.1	49.3	34.7	↓ 3.4	63.8(↓ 3.1)	40.1(↓ 3.0)	35.0(↓ 0.1)
GRPO	58.9	**63.5**	76.3	**35.2**	53.6	35.2	↓ 1.2	**66.7**(↓ 0.2)	**42.0**(↓ 1.1)	**36.3**(↑ 1.3)
TMS (Ours)	**59.0**	62.8	76.8	32.4	**54.4**	38.6	↓ 2.9	65.4(↓ 1.4)	41.4(↓ 1.7)	34.8(↓ 0.3)
Qwen-2.5-3B-Instruct										
Base	60.2	62.0	85.2	29.6	60.6	43.8	–	80.5	53.6	35.7
Standard SFT	76.4	80.3	**90.2**	49.2	71.3	**54.0**	↓ 39.2	42.7(↓ 37.8)	24.1(↓ 29.5)	31.1(↓ 4.6)
Self-SFT	74.2	77.4	87.3	48.6	67.9	47.2	↓ 29.3	51.8(↓ 28.7)	33.1(↓ 20.5)	33.6(↓ 2.1)
Final-SFT	75.8	79.6	89.5	48.1	70.2	51.4	↓ 35.1	45.2(↓ 35.3)	28.4(↓ 25.2)	33.2(↓ 2.5)
REINFORCE	75.1	78.2	88.4	47.3	67.2	49.1	↓ 4.1	78.6(↓ 1.9)	51.4(↓ 2.2)	**35.5**(↓ 0.2)
GRPO	77.4	**81.5**	90.1	**54.2**	70.9	50.8	↓ 1.9	**80.2**(↓ 0.3)	**53.1**(↓ 0.5)	35.4(↓ 0.3)
TMS (Ours)	**77.8**	81.1	88.9	51.3	**72.0**	53.6	↓ 2.3	79.2(↓ 1.3)	51.7(↓ 1.9)	35.0(↓ 0.7)
Qwen-2.5-7B-Instruct										
Base	72.2	70.5	89.9	41.0	69.5	55.8	–	88.8	63.9	38.3
Standard SFT	82.6	83.4	**94.8**	59.4	**79.7**	**64.2**	↓ 41.2	53.5(↓ 35.3)	36.2(↓ 27.7)	34.1(↓ 4.2)
Self-SFT	80.4	81.1	91.6	56.0	72.9	59.4	↓ 33.8	68.4(↓ 20.4)	47.1(↓ 16.8)	36.9(↓ 1.4)
Final-SFT	81.8	82.0	92.8	58.4	75.0	63.4	↓ 37.0	59.8(↓ 29.0)	43.6(↓ 20.3)	36.4(↓ 1.9)
REINFORCE	81.1	80.4	92.1	60.1	74.1	59.0	↓ 4.2	86.8(↓ 2.0)	62.1(↓ 1.8)	38.0(↓ 0.3)
GRPO	83.4	**85.8**	93.6	**69.0**	77.5	60.8	↓ 1.6	**88.4**(↓ 0.4)	**63.4**(↓ 0.5)	**38.6**(↑ 0.3)
TMS (Ours)	**83.6**	83.9	93.7	62.8	78.6	62.9	↓ 2.8	86.9(↓ 1.9)	62.8(↓ 1.1)	37.1(↓ 1.2)
LLaMA-3.1-8B-Instruct										
Base	47.2	46.0	84.1	18.6	67.0	46.2	–	84.1	52.6	33.8
Standard SFT	66.7	64.6	**89.6**	39.2	75.2	**55.1**	↓ 32.9	53.2(↓ 30.9)	28.2(↓ 24.4)	30.1(↓ 3.7)
Self-SFT	60.4	60.4	87.4	36.4	71.6	50.8	↓ 24.9	64.2(↓ 19.9)	39.1(↓ 13.5)	32.6(↓ 1.2)
Final-SFT	64.3	63.1	88.8	37.4	73.8	53.2	↓ 31.4	60.8(↓ 23.3)	36.2(↓ 16.4)	32.2(↓ 1.6)
REINFORCE	63.9	61.8	88.1	34.8	72.2	49.3	↓ 4.9	82.1(↓ 2.0)	50.6(↓ 2.0)	33.6(↓ 0.2)
GRPO	**70.5**	**65.4**	89.4	**41.2**	75.6	50.8	↓ 0.9	**83.8**(↓ 0.3)	**52.1**(↓ 0.5)	**34.1**(↑ 0.3)
TMS (Ours)	66.3	64.9	89.2	40.4	**76.1**	54.6	↓ 2.3	83.1(↓ 1.0)	51.4(↓ 1.2)	33.0(↓ 0.8)
LLaMA-3.2-1B-Instruct										
Base	20.4	22.9	36.3	3.6	45.1	16.7	–	38.9	21.9	24.8
Standard SFT	35.2	38.8	53.1	12.8	52.8	**25.2**	↓ 24.1	23.4(↓ 15.5)	10.8(↓ 11.1)	21.8(↓ 3.0)
Self-SFT	30.6	32.0	48.9	12.0	49.6	21.8	↓ 17.4	29.2(↓ 9.7)	14.8(↓ 7.1)	23.6(↓ 1.2)
Final-SFT	33.9	37.9	51.0	12.2	51.4	23.8	↓ 24.6	26.8(↓ 12.1)	13.1(↓ 8.8)	23.1(↓ 1.7)
REINFORCE	32.4	35.6	51.4	11.9	50.1	22.1	↓ 3.8	36.8(↓ 2.1)	20.1(↓ 1.8)	24.4(↓ 0.4)
GRPO	**38.5**	**39.5**	**54.3**	**15.8**	42.1	24.1	↓ 1.6	**38.2**(↓ 0.7)	21.2(↓ 0.7)	**25.0**(↑ 0.2)
TMS (Ours)	35.7	38.9	52.6	13.5	**53.7**	**25.2**	↓ 3.1	37.6(↓ 1.3)	**21.8**(↓ 0.1)	24.6(↓ 0.2)