TAP: EFFICIENT LONG-HORIZON AGENT PRUNING VIA TRAJECTORY-ANCHORED RECOVERY

Anonymous authors
Paper under double-blind review

ABSTRACT

Emerging long-horizon agentic tasks require repeated model calls, worsening the inference cost of already-costly language models. While narrow agentic tasks suggest potential for aggressive model pruning without performance drop, empirical results show existing methods proposed for question answering tasks severely degrade task performance when applied to agentic models. We trace this failure to two decisions: what to prune and how to recover. For pruning, one-shot importance estimates fail to track how the pruned model adapts. For recovery, offline distillation covers only teacher prefixes, while full-trajectory on-policy distillation causes student errors to compound across turns. In this work, we propose *Trajectory-Anchored Pruning (TAP)*, the first structural pruning framework for reinforcement learning (RL)-trained agents. TAP couples structural pruning with efficient on-policy recovery, anchoring interactions to teacher trajectories while allowing the student to generate each reasoning–action response. A frozen dense teacher supervises the student's response prefixes, addressing within-response training–inference mismatch while preventing student-induced deviations from propagating across training turns. Instead of one-shot pruning, TAP re-scores channels using gradients of the recovery objective on the recovered student, connecting iterative channel selection to the evolving policy. With 60% of FFN channels removed, TAP retains 99.2% and 88.0% of the dense 7B agents' task success rates on ALFWorld and WebShop, respectively, while reducing GPU time per successful task by approximately 22% and 17%. These results demonstrate effective structural compression of long-horizon agents under a limited recovery budget.

1 INTRODUCTION

Reinforcement learning (RL) has produced language agents with strong capabilities in multi-step tasks (Feng et al., 2025; He et al., 2026; Qi et al., 2026; Zhen et al., 2026; Li et al., 2026b; Qin et al., 2025). Each decision, however, conditions on environment observations, tool descriptions, and an ever-growing interaction history, and solving a long-horizon task requires many such model invocations, so inference cost accumulates over the trajectory (Kang et al., 2025; Wang et al., 2026b; Li et al., 2026d). This growing cost motivates model compression for efficient agent deployment.

Compressing an LLM tuned for a specific agentic task should, in principle, be unusually forgiving. Compression needs to preserve only the decision-making the task requires, not the pretrained model's broad knowledge, which suggests room for more aggressive pruning than general-purpose LLMs tolerate (Hu et al., 2025). However, existing structural pruning methods, developed and evaluated primarily on static tasks (Ma et al., 2023; Zhang et al., 2024; Muralidharan et al., 2024; Zhang et al., 2026), fail to exploit it. Our experiments confirm that a 7B WebShop agent with 80% of its FFN channels removed can retain the success rate attained at only 40% removal by previous methods (Figure 1b), indicating a significant gap between the compression potential of agentic LLMs and what existing methods achieve.

We attribute this gap to limitations in two key decisions: what to prune and how to recover afterwards. **Deciding what to prune.** Existing methods typically estimate channel importance once, on the dense model, and remove the selected channels in a single step. Two challenges arise when applying this recipe to RL-trained agents. First, what must be preserved is the agent's action-generating policy on interaction contexts, which criteria based on token likelihood only approximate; even on

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

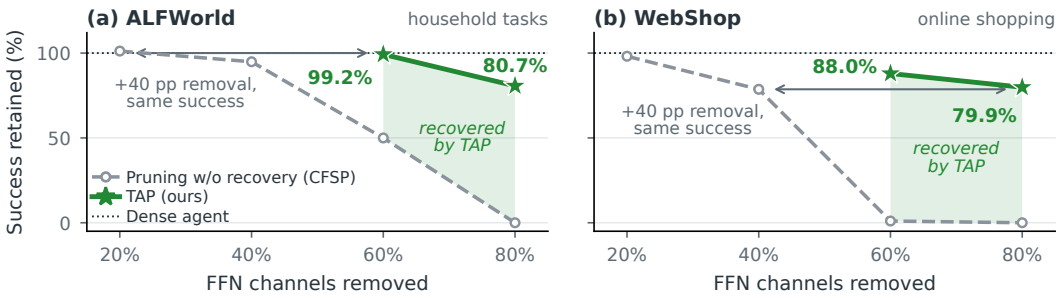

Figure 1: **RL-trained agents are highly compressible, but only with suitable pruning and recovery.** Share of the dense agent's task success retained by 7B agents as FFN channels are removed. Pruning without recovery collapses beyond 40% removal, whereas TAP retains about 80–99% of dense success up to 80% removal.

agent contexts, replacing a distribution-matching criterion with likelihood lowers out-of-distribution success by 5.5 points (Table 7). Second, a single cut commits to a channel set before observing how the pruned student adapts. After recovery, re-scoring changes which channels are retained relative to one-shot selection (Table 6). In each case, pruning moves the student away from the dense agent in ways the selection criterion does not see, leading to teacher–student policy drift.

Deciding how to recover. Knowledge distillation is the standard way to recover structurally pruned models (Muralidharan et al., 2024), but distillation on fixed teacher trajectories trains the student only on teacher prefixes, whereas at deployment the student conditions on its own outputs. In the agent setting this training–inference mismatch is severe: KD-recovered baselines steadily reduce their distillation loss, yet their steps produce missing or malformed action blocks. On-policy distillation (OPD) removes this mismatch by supervising the student's own generations (Gu et al., 2024; Agarwal et al., 2024). For long-horizon agents, however, full-trajectory OPD also lets the student's actions determine subsequent environment states. A heavily pruned student then trains on states increasingly unlike those the teacher visits, where teacher supervision becomes less reliable (Li et al., 2026c; Wang et al., 2026a; Li et al., 2026a), and its errors compound across turns (Wang et al., 2026a; Li et al., 2026a).

To address these two limitations, we propose **TAP**, an iterative pruning framework with *trajectory-anchored on-policy distillation*. TAP fixes interaction contexts to teacher trajectories while allowing the student to generate each complete reasoning–action response under token-level supervision from a frozen dense teacher. This retains on-policy supervision within each turn while preventing student actions from altering subsequent training contexts. We interleave structural pruning with anchored recovery and, before each subsequent cut, re-score channels using gradients of the same recovery objective on the recovered student, coupling policy recovery with structural selection.

Experiments show that with 60% of FFN channels removed, TAP retains 99.2% and 88.0% of the dense 7B agents' success rates on ALFWorld and WebShop, respectively, while reducing GPU time per successful task by approximately 22% and 17%; at 80% removal, it retains 80.7% and 79.9%. TAP also runs efficiently, using only 0.18–1.21 million supervised response tokens across all recovery stages for the 1.5B and 7B agents.

Our main contributions are summarized as follows:

- We study structural pruning of RL-trained long-horizon agents and trace existing methods' sharp degradation at aggressive ratios to limitations in *what to prune* and *how to recover*. We further show that decreasing distillation loss can coexist with malformed generation.
- We propose TAP, which couples trajectory-anchored OPD with iterative pruning and recovery-informed channel recalibration, so that the recovery objective also guides channel selection at each stage.
- Experiments on ALFWorld and WebShop at two model scales demonstrate improved task-success retention and reduced deployment cost, with ablations isolating the contributions of iterative pruning, recalibration, and anchoring.

2 Related Work

Structural pruning for LLMs. Structural pruning removes channels, attention heads, or layers so that the pruned model runs with standard dense operations. Prior work scores structures with gradient- or activation-based importance and adaptive budget allocation (Ma et al., 2023; Zhang et al., 2024; Wang et al., 2025; Guo et al., 2025b), compensates for pruning errors with curvature-aware updates or local reconstruction (van der Ouderaa et al., 2024; Wei et al., 2024; Ling et al., 2024; Guo et al., 2025b), and recovers the pruned model with parameter-efficient fine-tuning or distillation (Ma et al., 2023; Zhang et al., 2024; Muralidharan et al., 2024; Zhang et al., 2026), mostly on static language tasks. Gradual pruning interleaves pruning with training (Zhu & Gupta, 2018), and for LLMs, multi-step pruning with retraining improves on one-shot removal at high compression (Zhang et al., 2024; Pan et al., 2025). For agents, ACBench evaluates unstructured and 2:4 weight sparsity without recovery training (Dong et al., 2025), REAP prunes MoE experts for coding and tool use (Lasby et al., 2026), and pruning has been combined with RL for embodied control: Before Parc Fermé prunes driving LLMs during RL, and RLRC recovers compressed vision-language-action models with SFT followed by RL (Benfenati et al., 2026; Chen et al., 2026). We instead study structural pruning of dense RL-trained agents that interact with their environments over long horizons. Whereas prior criteria score channels by language-modeling gradients or activations, TAP scores them with gradients of the teacher–student divergence that its recovery minimizes on agent contexts, and re-scores them on the recovered student before each subsequent cut to limit drift that compounds over long horizons.

Distillation and on-policy distillation. Distillation transfers capabilities, including those acquired through RL, from a teacher to a smaller student (Guo et al., 2025a; Yang et al., 2025). A student trained on fixed teacher demonstrations may reach sequences or states absent from them (Ross et al., 2011); MiniLLM and GKD address this by supervising the student's own generations (Gu et al., 2024; Agarwal et al., 2024). For multi-turn agents, TCOD and Guided-OPD report that student rollouts compound errors across turns and leave the teacher's reliable support, and mitigate this with a rollout-depth curriculum or decaying teacher intervention (Wang et al., 2026a; Li et al., 2026a); TurnOPD (Zhou et al., 2026) rebalances rollout and loss budgets across turns. ShortOPD recovers structurally pruned models with short-to-long rollouts on mathematics, code, and open-ended generation (Zhang et al., 2026). These methods collect student rollouts in the environment or target non-agentic tasks. TAP recovers a pruned agent by anchoring every training context to teacher trajectories while supervising the student's own responses, requires no environment interaction, and couples this recovery with iterative channel selection. Anchoring keeps training contexts where teacher supervision is reliable and stops student errors from propagating across turns.

3 Method

3.1 Preliminaries

Let x denote a task instruction and o_t the observation at step t. Let $h_{<t}$ contain the task instruction and retained observation–action pairs before step t. The interaction context is $c_t = (h_{<t}, o_t)$. We condition the policy on this context, writing $\pi_\theta(a_t \mid c_t)$ for brevity. This policy generates a response a_t containing reasoning and an executable action. The environment executes the parsed action and returns o_{t+1}. We start from a dense agent π_T after RL and freeze it as the teacher. We denote the student by $\pi_S = \pi_\theta$, where θ are its current parameters. Our objective is to preserve its complete-task performance under structural model reduction:

$$J(\pi_S) = \mathbb{E}_{x\sim\mathcal{D}_{\text{eval}},\, \tau\sim\pi_S(\cdot|x)}[R(\tau)], \tag{1}$$

where $\mathcal{D}_{\text{eval}}$ is the evaluation task distribution, τ is a complete student–environment trajectory, and $R(\tau)$ is its task return.

We prune intermediate channels in SwiGLU FFNs by removing the corresponding rows of the gate and up projections and the corresponding column of the down projection. Formal definitions are provided in Appendix C.2.

Overview. TAP progressively compresses the student over K stages, coupling structural pruning with teacher-guided recovery (Figure 2, Algorithm 1). Each stage follows a prune–recover–rescore

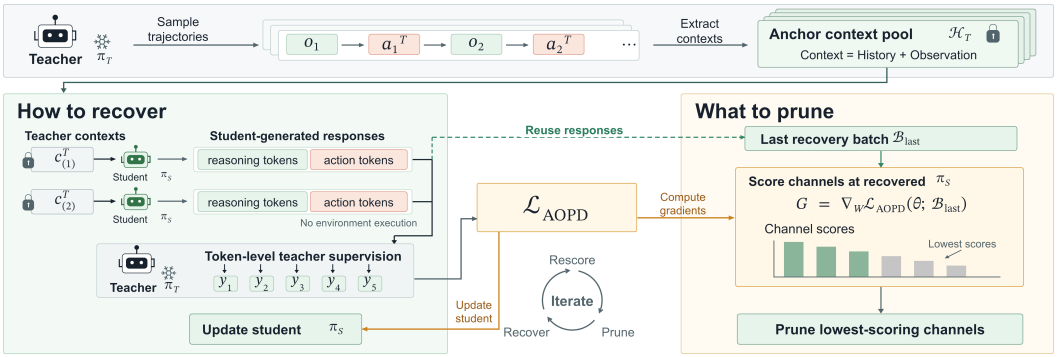

Figure 2: **TAP couples how to recover and what to prune through one objective.** Top: teacher trajectories provide a fixed pool of anchor contexts. Left (how to recover): within anchor contexts, the student generates reasoning and action tokens, the frozen teacher supervises every generated token, and minimizing $\mathcal{L}_{\text{AOPD}}$ updates the student. Right (what to prune): gradients of the same objective on the last recovery batch score channels at the recovered student, and remove the lowest-scoring channels. The prune–recover–rescore cycle repeats until the target removal ratio is reached.

Algorithm 1 TAP: Iterative Pruning with Trajectory-Anchored Recovery

Require: Teacher π_T, training tasks $\mathcal{D}_{\text{train}}$, target ratio $\rho_\star$, stages K, token budget Q, warm-up fraction α
Notation: $s_{\ell,j}$ is the importance score of FFN channel j in layer ℓ; $\mathcal{B}_{\text{last}}$ is the last training batch of anchored OPD
Construct anchor pool $\mathcal{H}_T$ from teacher trajectories on $\mathcal{D}_{\text{train}}$
Initialize $\pi_S \leftarrow \pi_T$; keep teacher π_T and anchor pool $\mathcal{H}_T$ fixed
$\{s_{\ell,j}\} \leftarrow \text{InitialScores}(\text{GaussianPerturb}(\pi_S), \pi_T, \mathcal{H}_T)$
for $k = 1, \ldots, K$ **do**
  Set cumulative removal target ρ_k using Eq. (5)
  Prune: $\pi_S \leftarrow \text{Prune}(\pi_S, \{s_{\ell,j}\}, \rho_k)$
  Recover: $\pi_S \leftarrow \text{KDWarmup}(\pi_S, \pi_T, \mathcal{H}_T;\ \alpha Q/K)$
    $(\pi_S, \mathcal{B}_{\text{last}}) \leftarrow \text{AnchoredOPD}(\pi_S, \pi_T, \mathcal{H}_T;\ (1-\alpha)Q/K)$
  if $k < K$ **then**
    Rescore: compute OPD gradients on $\mathcal{B}_{\text{last}}$ at the recovered π_S
      update $\{s_{\ell,j}\}$ using Eq. (4) for the next cut
  end if
end for
return π_S

cycle: it removes FFN channels to reach a cumulative ratio ρ_k, recovers the pruned student by distillation from the frozen teacher within anchored contexts, and re-scores the remaining channels with gradients of the same recovery objective on the recovered student. This shared objective links the two steps: recovery restores the student's policy after each cut, and its objective, evaluated on its last batch of responses, scores channels for the next cut, while pruning targets the channels whose removal least perturbs the recovery objective. Together, they keep teacher supervision reliable by limiting drift from both pruning and long-horizon interaction. Section 3.2 introduces the recovery objective, and Section 3.3 explains how it guides iterative channel selection.

3.2 Trajectory-Anchored On-Policy Distillation

TAP recovers the pruned student with on-policy distillation anchored to teacher trajectories: the student generates its own responses, but only within interaction contexts taken from the teacher's trajectories.

Anchoring interaction context. We collect teacher trajectories on training tasks $\mathcal{D}_{\text{train}}$ and extract a fixed pool $\mathcal{H}_T = \{(c_t^T, a_t^T)\}$ of context–response pairs. Each context $c_t^T = (h_{<t}^T, o_t^T)$ contains the retained teacher history and current observation; a_t^T is the corresponding teacher response. We denote the sampling distribution over these contexts by $q_{\mathcal{H}_T}$. Given an anchor context

formatted with the agent's prompt template, the current student generates a fresh response:

$$\widehat{a}_t \sim \pi_S(\cdot \mid c_t^T). \tag{2}$$

The generated action is not executed in the environment; the next training context is sampled from $\mathcal{H}_T$. This trajectory-anchored recovery requires no additional environment interaction during training.

Distillation on student responses. We construct each training batch $\mathcal{B}$ by sampling anchor contexts $c^T \sim q_{\mathcal{H}_T}$ and student responses $\widehat{a}$. For a response $\widehat{a} = (y_1, \ldots, y_L)$, let $p_j = (c^T, y_{<j})$ denote the context followed by the student-generated prefix. The frozen teacher scores the same prefixes, and we minimize the reverse KL averaged over the batch's $N_{\mathcal{B}}$ response tokens:

$$\mathcal{L}_{\mathrm{AOPD}}(\theta; \mathcal{B}) = \frac{1}{N_{\mathcal{B}}} \sum_{(c^T, \widehat{a}) \in \mathcal{B}} \sum_{j=1}^{|\widehat{a}|} D_{\mathrm{KL}}(\pi_\theta(\cdot \mid p_j) \,\|\, \pi_T(\cdot \mid p_j)). \tag{3}$$

Following GKD (Agarwal et al., 2024), sampled responses are held fixed during each update; following Guided-OPD (Li et al., 2026a), we use reverse KL with supervision restricted to response tokens.

Relation to KD and full-trajectory OPD. Recovery objectives differ in where the supervised prefixes $p_j = (c, y_{<j})$ come from (definitions in Appendix C.5). Logits KD supervises teacher prefixes, pairing a teacher context with a teacher response prefix, so the student never learns to continue from its own tokens. Full-trajectory OPD supervises prefixes generated entirely by the student, so its contexts shift with the student's own actions. Anchored OPD takes the context from teacher trajectories and the response prefix from the student: supervision covers the prefixes the student produces within each turn, while its contexts stay where teacher supervision is reliable, and recovery requires no environment interaction.

3.3 Recovery-Guided Iterative Pruning

Channel importance from the recovery objective. We select channels by minimal disturbance: a channel is unimportant if removing it barely changes the anchored OPD objective, the teacher–student divergence that recovery minimizes, so pruning aims to keep the student close to the teacher. Following loss-based pruning (Ma et al., 2023), we approximate this change with a first-order Taylor criterion. For layer ℓ, let $W_\ell^g, W_\ell^u, W_\ell^d$ denote the student's gate, up, and down weights, and let $G_\ell^p = \partial \mathcal{L}_{\mathrm{AOPD}}(\theta; \mathcal{B}) / \partial W_\ell^p$ for $p \in \{g, u, d\}$ be the gradient of Eq. (3) on a calibration batch $\mathcal{B}$ of anchor contexts and student responses. The importance of intermediate channel j combines the corresponding gate/up rows and down column:

$$s_{\ell,j} = \sum_i \left| W_{\ell,ji}^g G_{\ell,ji}^g \right| + \sum_i \left| W_{\ell,ji}^u G_{\ell,ji}^u \right| + \sum_i \left| W_{\ell,ij}^d G_{\ell,ij}^d \right|, \tag{4}$$

where i indexes the hidden dimension and weights use output-by-input indexing. We remove the lowest-scoring channels within each layer. Because the scores reuse recovery responses, they require no additional environment interaction.

At the first cut, the student equals the teacher and the KL gradients vanish. We therefore compute the initial scores on a perturbed student, obtained by adding small Gaussian noise to its FFN weights, and apply the resulting selection to the unperturbed student. Details of channel scoring and initial calibration are provided in Appendix C.3.

Iterative pruning. Pruning and recovery change the student's parameters and policy, so a ranking estimated once before compression no longer reflects the adapted student. We therefore prune in K stages and, before each subsequent cut, re-score the remaining channels on the recovered student using the last batch of recovery responses. Recovery between cuts also restores teacher–student alignment before further structural reduction. Let ρ_k be the cumulative fraction of original FFN channels removed after cut k, and $\rho_\star$ the final target. Following the cubic schedule of Zhu & Gupta (2018), we increase the cumulative removal ratio with larger cuts early and smaller cuts later:

$$\rho_k = \rho_\star \left[1 - \left(1 - \frac{k}{K} \right)^3 \right], \qquad k = 1, \ldots, K. \tag{5}$$

We use $K = 3$ in our experiments; for an 80% final target, the cumulative removal ratios are approximately 56%, 77%, and 80%. The teacher and anchor pool remain fixed throughout. We divide the total supervised response-token budget Q equally across stages; each stage allocates a fraction α of its budget to logits KD warm-up on teacher response prefixes, followed by anchored OPD. We set $\alpha = 0.2$ in our experiments.

4 EXPERIMENTS

We organize our experiments around four research questions (RQs). **RQ1** (Section 4.2): Does TAP preserve task success under aggressive structural pruning? RQ2 and RQ3 address the two decisions identified in Section 1. **RQ2** (Section 4.3, *what to prune*): Does re-scoring channels with the recovery objective on the adapted student select better channels than a fixed ranking? **RQ3** (Section 4.4, *how to recover*): How does trajectory anchoring compare with offline KD and full-trajectory OPD? Finally, we verify **RQ4** (Section 4.5): Does structural compression reduce the cost of completing agent tasks?

4.1 EXPERIMENTAL SETUP

Benchmarks and models. We conduct experiments on ALFWorld and WebShop, two representative benchmarks for long-horizon language agents (Shridhar et al., 2021; Yao et al., 2022). We use RL-trained agents at two model scales, **1.5B** and **7B**, from SPEAR and GiGPO (Qin et al., 2025; Feng et al., 2025), with model and checkpoint details provided in Appendix C.1. The FFN channel-pruning ratios are **40%** and **60%** for the **1.5B** models, and **60%** and **80%** for the **7B** models.

Baselines. We compare TAP with LLM-Pruner (Ma et al., 2023), CFSP (Wang et al., 2025), and SlimGPT (Ling et al., 2024), grouped by how they recover the pruned agent. LLM-Pruner and CFSP select channels once and recover with logits KD. We evaluate SlimGPT with its native local reconstruction and no additional recovery training to assess the effectiveness of reconstruction alone in our agent setting. We additionally continue GiGPO training of the 1.5B agents while removing one percentage point of FFN channels per optimizer step (online RL, denoted RL + Pruning; Appendix A). All methods use uniform layer-wise FFN pruning ratios. For offline recovery comparisons, we match the total supervised token budget and batch size. Other baseline adaptations and budget construction are provided in Appendix C.

Evaluation settings. We evaluate on both the in-distribution (In) and out-of-distribution (Out) splits of ALFWorld and the full 500-task test set of WebShop. We report task success rate over three seeds and performance retention; for ALFWorld, retention averages the two splits. The ablations in Sections 4.3 and 4.4 use the 7B agents.

4.2 MAIN RESULTS: TASK SUCCESS UNDER AGGRESSIVE PRUNING

RQ1. *Does TAP preserve task success under aggressive structural pruning?*

Comparison with pruning baselines. Table 1 shows that the baselines remain competitive only at the milder ratio of each model: at 60% removal of the 7B agents, CFSP + KD retains 96.34% of dense ALFWorld success, versus 99.24% for TAP. The gap opens at the aggressive ratio. At 60% removal of the 1.5B agents, no baseline retains more than 9.07% of dense WebShop success, whereas TAP retains 69.38%; at 80% removal of the 7B agents, TAP exceeds CFSP + KD by 10.28 and 8.12 retention points on ALFWorld and WebShop. The other recovery routes are brittle. Pruning during RL retains at most 41.96% of dense success already at 40% removal. SlimGPT's reconstruction alone collapses on 1.5B ALFWorld and on the 7B agents at 80%, despite near-dense success at 20% (Appendix B.1).

Pruning sensitivity. Figure 1 (Section 1) shows how much of this capacity is recoverable. Without recovery, CFSP success declines sharply with channel removal: WebShop SR falls to 0.8% at 60% removal, and both benchmarks reach zero at 80%. TAP retains a similar share of dense success at substantially higher removal: 99.2% on ALFWorld at 60% removal, comparable to prune-only CFSP at 20%, and 79.9% on WebShop at 80%, comparable to prune-only CFSP at 40%.

Table 1: **TAP retains the most task success, and its lead grows with the removal ratio.** We test whether TAP preserves the success of RL-trained agents under aggressive FFN pruning, compared with pipelines that recover by online RL (RL + Pruning), offline logits KD (+ KD), or local reconstruction alone (SlimGPT). Entries are success rates over three seeds and retention relative to the dense agent; ALFWorld retention averages the In and Out splits. **Bold**/underline: best/second best per setting. At the milder ratio of each model, the strongest baselines stay within 3.5 retention points of TAP; at the aggressive ratio, TAP's retention exceeds the best baseline's by 8–60 points. The redundancy of task-specific agents is therefore exploitable, but only with suitable channel selection and recovery.

ρ	Method	ALFWorld			WebShop	
		In SR (%)	Out SR (%)	Retention (%)	SR (%)	Retention (%)
Qwen2.5-1.5B						
0%	Dense RL	$87.86_{\pm 2.86}$	$83.83_{\pm 1.55}$	100	$77.93_{\pm 0.12}$	100
40%	RL + Pruning	$36.19_{\pm 2.30}$	$35.82_{\pm 2.69}$	41.96	$4.07_{\pm 1.22}$	5.22
	LLM-Pruner + KD	$73.81_{\pm 4.65}$	$61.94_{\pm 4.54}$	78.95	$14.00_{\pm 0.72}$	17.96
	CFSP + KD	$\mathbf{82.62}_{\pm 3.67}$	$\underline{75.12}_{\pm 0.86}$	$\underline{91.82}$	$39.73_{\pm 16.01}$	50.98
	SlimGPT (recon.)	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	0.00	$\underline{60.60}_{\pm 0.80}$	$\underline{77.76}$
	TAP (ours)	$\underline{82.38}_{\pm 0.41}$	$\mathbf{77.36}_{\pm 1.88}$	**93.02**	$\mathbf{63.33}_{\pm 0.90}$	**81.26**
60%	RL + Pruning	$6.90_{\pm 2.30}$	$1.74_{\pm 1.55}$	4.97	$0.00_{\pm 0.00}$	0.00
	LLM-Pruner + KD	$30.00_{\pm 0.71}$	$25.12_{\pm 1.55}$	32.06	$0.07_{\pm 0.12}$	0.09
	CFSP + KD	$\underline{67.38}_{\pm 1.09}$	$\underline{63.93}_{\pm 2.15}$	$\underline{76.48}$	$4.20_{\pm 2.60}$	5.39
	SlimGPT (recon.)	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	0.00	$\underline{7.07}_{\pm 1.33}$	$\underline{9.07}$
	TAP (ours)	$\mathbf{78.81}_{\pm 4.59}$	$\mathbf{67.66}_{\pm 2.83}$	**85.21**	$\mathbf{54.07}_{\pm 1.53}$	**69.38**
Qwen2.5-7B						
0%	Dense RL	$94.29_{\pm 1.89}$	$89.55_{\pm 0.75}$	100	$78.07_{\pm 0.50}$	100
60%	LLM-Pruner + KD	$84.05_{\pm 0.41}$	$75.87_{\pm 1.72}$	86.92	$52.47_{\pm 0.50}$	67.21
	CFSP + KD	$\mathbf{92.62}_{\pm 1.49}$	$\underline{84.58}_{\pm 1.88}$	$\underline{96.34}$	$\underline{66.27}_{\pm 0.76}$	$\underline{84.88}$
	SlimGPT (recon.)	$\underline{86.43}_{\pm 1.24}$	$83.08_{\pm 3.53}$	92.22	$51.60_{\pm 1.22}$	66.10
	TAP (ours)	$\mathbf{92.62}_{\pm 1.65}$	$\mathbf{89.80}_{\pm 1.55}$	**99.24**	$\mathbf{68.67}_{\pm 0.95}$	**87.96**
80%	LLM-Pruner + KD	$26.43_{\pm 1.89}$	$21.64_{\pm 1.49}$	26.06	$25.60_{\pm 0.69}$	32.79
	CFSP + KD	$\underline{66.19}_{\pm 2.18}$	$\underline{63.18}_{\pm 3.11}$	$\underline{70.38}$	$\underline{56.00}_{\pm 1.40}$	$\underline{71.73}$
	SlimGPT (recon.)	$6.43_{\pm 3.78}$	$2.24_{\pm 1.49}$	4.62	$0.00_{\pm 0.00}$	0.00
	TAP (ours)	$\mathbf{76.19}_{\pm 1.80}$	$\mathbf{72.14}_{\pm 3.11}$	**80.66**	$\mathbf{62.33}_{\pm 1.21}$	**79.85**

Table 2: **Iterative pruning helps when channels are re-scored with the recovery objective.** To isolate what to prune as the student adapts, we fix the recovery objective and vary how channels are removed on Qwen2.5-7B. Cubic and uniform schedules use three pruning stages. A and B delete the same final channels, so B vs. A isolates interleaving recovery between cuts; B vs. C isolates recalibration; U vs. C compares uniform and cubic schedules. **Bold**/underline: best/second best. Iterating with a fixed ranking gives mixed changes, whereas recalibration improves five of six metrics, and C improves on one-shot pruning in all six. The gain therefore comes from coupling channel selection with recovery.

			60% **FFN removal**			80% **FFN removal**		
Variant	Schedule	Ranking	ALF In	ALF Out	WebShop	ALF In	ALF Out	WebShop
A	One-shot	Fixed	$\underline{92.38}_{\pm 2.51}$	$86.32_{\pm 2.40}$	$\underline{66.87}_{\pm 0.23}$	$73.81_{\pm 2.06}$	$61.69_{\pm 2.28}$	$57.40_{\pm 1.22}$
B	Cubic	Fixed	$90.48_{\pm 1.65}$	$87.81_{\pm 0.86}$	$63.27_{\pm 0.12}$	$\mathbf{77.38}_{\pm 2.70}$	$\underline{63.93}_{\pm 1.14}$	$\underline{57.93}_{\pm 1.01}$
C (TAP)	Cubic	Recalibrated	$\mathbf{92.62}_{\pm 1.65}$	$\mathbf{89.80}_{\pm 1.55}$	$\mathbf{68.67}_{\pm 0.95}$	$\underline{76.19}_{\pm 1.80}$	$\mathbf{72.14}_{\pm 3.11}$	$\mathbf{62.33}_{\pm 1.21}$
U	Uniform	Recalibrated	$90.71_{\pm 1.24}$	$\underline{89.55}_{\pm 0.75}$	$65.80_{\pm 1.11}$	$50.00_{\pm 1.89}$	$43.03_{\pm 0.86}$	$51.93_{\pm 1.92}$

4.3 What to Prune: Recovery-Guided Channel Recalibration

RQ2. *Does re-scoring channels with the recovery objective on the adapted student select better channels than a fixed ranking?*

Iterative pruning and channel recalibration. In Table 2 (protocol in Appendix B.4), iterative pruning with a fixed ranking (B) gives mixed changes relative to one-shot pruning (A), lowering two of three metrics at 60% removal. Recalibration makes iteration pay off: C improves on B in five of six metrics, by up to 8.21 points on ALFWorld Out at 80% removal, and improves on one-shot pruning in all six. Recalibration also changes the surviving subnetwork, replacing 14–22% of the channels that one-shot selection retains (Appendix B.2). With recalibration, a uniform schedule (U) trails the cubic one on all six metrics.

The objective used for recalibration. Replacing TAP's reverse-KL scoring (Section 3.3) with token NLL leaves ALFWorld In SR similar but lowers Out SR from 89.80% to 84.33% at 60% removal (Table 7), so scoring channels by the teacher–student divergence better preserves the agent's policy. Overall, iterative pruning pays off consistently only when channel importance is re-estimated with the recovery objective on the adapted student.

4.4 How to Recover: Trajectory-Anchored Distillation

RQ3. *How does trajectory anchoring compare with offline KD and full-trajectory OPD?*

Table 3: **Recovery works best when the student generates its own responses within teacher interaction contexts.** To isolate how a pruned agent should be recovered, we fix the pruning path to TAP's and vary only the recovery objective on Qwen2.5-7B. *Context* and *Resp.* (response) indicate whose interaction contexts the student conditions on and whose responses it is supervised on: logits KD and SeqKD use teacher responses with soft and hard targets, full-trajectory OPD uses the student's own rollouts, and anchored OPD uses student responses within teacher contexts. Entries are SR; **bold**/underline: best/second best. Anchored OPD is the best or within one standard deviation of the best in every column, whereas full-trajectory OPD collapses on ALFWorld at 80% removal. Comparing rows isolates the two design choices: supervising the student's own responses improves on KD, and anchoring its contexts avoids the collapse of full-trajectory OPD.

	Source		60% **FFN removal**			80% **FFN removal**		
Recovery	Context	Resp.	ALF In	ALF Out	WebShop	ALF In	ALF Out	WebShop
D: Logits KD	Teacher	Teacher	$\mathbf{93.33}_{\pm 1.65}$	$\underline{88.06}_{\pm 1.29}$	$\underline{65.60}_{\pm 0.72}$	$\underline{75.95}_{\pm 2.18}$	$\underline{70.90}_{\pm 5.38}$	$59.80_{\pm 0.80}$
F: SeqKD	Teacher	Teacher	$\underline{92.62}_{\pm 1.09}$	$85.32_{\pm 0.86}$	$61.87_{\pm 0.58}$	$70.95_{\pm 4.86}$	$65.42_{\pm 3.76}$	$55.40_{\pm 1.60}$
E: Traj. OPD	Student	Student	$92.14_{\pm 0.71}$	$86.07_{\pm 1.55}$	$64.07_{\pm 0.23}$	$49.76_{\pm 2.89}$	$41.54_{\pm 5.80}$	$\mathbf{62.67}_{\pm 0.61}$
C: Anchored	Teacher	Student	$\underline{92.62}_{\pm 1.65}$	$\mathbf{89.80}_{\pm 1.55}$	$\mathbf{68.67}_{\pm 0.95}$	$\mathbf{76.19}_{\pm 1.80}$	$\mathbf{72.14}_{\pm 3.11}$	$\underline{62.33}_{\pm 1.21}$

Offline versus on-policy supervision. In Table 3, with teacher contexts fixed, supervising the student's own responses (anchored OPD) rather than teacher responses improves WebShop SR by 3.07 and 2.53 points over logits KD and by 6.80 and 6.93 points over SeqKD at 60% and 80% removal, and gives the highest ALFWorld Out SR at both ratios, while logits KD remains comparable on ALFWorld In, where the training–inference mismatch (Gu et al., 2024; Agarwal et al., 2024) is likely smaller. Offline KD can fail more severely after aggressive one-shot pruning: on 1.5B WebShop at 60% removal, the one-shot KD baselines reduce their KL by about 84% while 87–93% of their steps contain malformed action blocks (Appendix B.3). In Table 3, all recovery variants use the same pruning masks as TAP and differ only in the recovery objective. With these masks, logits KD is already much stronger than one-shot CFSP + KD; TAP's gains over the baselines thus combine a better pruning path with anchored recovery.

Anchored versus full-trajectory OPD. Full-trajectory OPD also supervises the student's own responses, but lets its executed actions determine the interaction contexts. It trails anchored OPD on all three metrics at 60% removal and collapses on ALFWorld at 80% (49.76% and 41.54% In and Out SR, versus 76.19% and 72.14%). These failures are not malformed outputs: only 1.32% of its steps contain a broken action block, and 95% of failed episodes run to the maximum step limit, so the student acts validly but fails to finish the task. The collapse arises for the weakest student on ALFWorld's much longer episodes, consistent with the cross-turn drift reported for multi-turn OPD (Wang et al., 2026a; Li et al., 2026a): the student's own rollouts steer supervision toward states unlike those in teacher trajectories, and errors compound over more turns. Anchoring interaction

contexts to teacher trajectories thus keeps on-policy recovery stable for heavily pruned students on long horizons.

4.5 Deployment Efficiency

RQ4. *Does structural compression reduce the cost of completing agent tasks?*

Table 4: **Compression lowers the cost per successful task when success is retained.** Agent cost depends on speed, steps per task, and success, so we report all three for dense and TAP-pruned Qwen2.5-7B agents. Steps: agent steps per task; E2E: end-to-end speedup per task; CPS: GPU time per success, including failed attempts, relative to dense (Appendix C.8). CPS falls by 17–29% except on ALFWorld at 80% removal, where 1.8× more steps per task outweigh faster inference.

		Speedup		**ALFWorld**			**WebShop**		
Model	Params	Prefill ↑	Decode ↑	Steps ↓	E2E ↑	CPS ↓	Steps ↓	E2E ↑	CPS ↓
Dense	7.62B	1.00×	1.00×	11.8	1.00×	1.00×	6.2	1.00×	1.00×
TAP, ρ=60%	4.19B	1.71×	1.73×	13.6	**1.30×**	**0.78×**	7.0	1.38×	0.83×
TAP, ρ=80%	3.05B	**1.94×**	**2.26×**	21.6	0.77×	1.61×	6.1	**1.76×**	**0.71×**

Model inference efficiency. Structural removal translates into practical acceleration: in Table 4, 60% and 80% FFN removal reduce the parameter count from 7.62B to 4.19B and 3.05B, and yield 1.71× and 1.94× prefill speedups and 1.73× and 2.26× decoding speedups, respectively. The pruned models also need 45% and 60% less memory for their weights.

Task-level efficiency. These gains extend to complete agent evaluations: at 60% removal, end-to-end speedups reach 1.30× on ALFWorld and 1.38× on WebShop; at 80% removal, WebShop achieves a 1.76× speedup. Thus, structural compression accelerates agent execution even with the additional costs of environment interaction. Because an agent calls the model at every step, per-call speedups accumulate over each trajectory. **Cost per success** (CPS), which also charges failed attempts, falls at 60% removal to 0.78× the dense baseline on ALFWorld and 0.83× on WebShop, reducing the resource cost per success by approximately 22% and 17%. WebShop at 80% removal achieves the largest saving, lowering CPS to 0.71×, or approximately 29% less GPU time per successful task. The exception is ALFWorld at 80% removal, where 1.8× more steps per task outweigh the faster inference.

5 Conclusion

We studied structural pruning of RL-trained long-horizon agents and traced the failure of existing pipelines at aggressive ratios to two decisions: what to prune and how to recover. TAP couples the two through a shared objective: it recovers each pruned student with trajectory-anchored on-policy distillation, supervising the student's own responses within teacher interaction contexts, and re-scores channels with gradients of this objective before each cut. With 60% of FFN channels removed, TAP retains 99.2% and 88.0% of the dense 7B agents' success on ALFWorld and WebShop while lowering cost per successful task; with 80% removed, it still retains 80.7% and 79.9%.

Our analyses separate the two decisions. For recovery, KD loss can decrease while generated actions remain malformed, and full-trajectory OPD collapses for heavily pruned students on the longer-horizon environment, whereas anchored recovery avoids both. For channel selection, iterative pruning helps consistently only when channels are re-scored on the recovered student.

TAP relies on informative teacher trajectories, and anchored recovery does not cover every state the deployed student may visit. We study FFN channel pruning of Qwen2.5 agents on two environments; extending TAP to other architectures, pruning granularities, and agentic domains, together with adaptive anchor selection and a gradual transition to unanchored recovery, are natural next steps.

Reproducibility Statement

All experiments start from publicly released RL-trained checkpoints (Appendix C.1). Appendix C specifies FFN channel removal (Appendix C.2), channel scoring and initial calibration (Appendix C.3), baseline adaptations (Appendix C.4), the definitions of all recovery objectives (Appendix C.5), recovery hyperparameters (Appendix C.6), recovery budgets for every setting (Appendix C.7), and the efficiency measurement protocol (Appendix C.8). The iterative-pruning ablation protocol is given in Appendix B.4.

AI Use Statement

In this work, we used generative AI tools to implement the proposed method, to support data analysis, and to assist with interpreting results. We have not used generative AI tools to design the research methodology or experiments, propose or refine hypotheses, develop theoretical models or conceptual frameworks, formulate mathematical claims, generate synthetic datasets, clean or reformat datasets, or assist with translation, and assistance with mathematical proofs is not applicable to this work. Additionally, we used generative AI tools to edit and proofread the manuscript, to write and debug code for experiment orchestration and figure plotting, and to search and summarize related literature. We have reviewed all AI-assisted work: the authors checked AI-edited text against the underlying results and references, and verified AI-assisted code, analyses, and interpretations against the logged experiment records. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=3zKtaqxLhW.

Luca Benfenati, Ali Azimi, Matteo Risso, Fabio Carapellese, Daniele Jahier Pagliari, and Alessio Burrello. Before Parc Fermé: RL-Time pruning for efficient embodied LLMs in autonomous driving. *CoRR*, abs/2605.31256, 2026. doi: 10.48550/ARXIV.2605.31256. URL https://doi.org/10.48550/arXiv.2605.31256.

Yuxuan Chen, Yixin Han, Yize Huang, and Xiao Li. RLRC: reinforcement learning-based recovery for compressed vision-language-action models. *IEEE Robotics Autom. Lett.*, 11(7):8864–8871, 2026. doi: 10.1109/LRA.2026.3700379. URL https://doi.org/10.1109/LRA.2026.3700379.

Peijie Dong, Zhenheng Tang, Xiang Liu, Lujun Li, Xiaowen Chu, and Bo Li. Can compressed LLMs truly act? an empirical evaluation of agentic capabilities in LLM compression. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/dong25k.html.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/420c9f777c0b4f78d515e53cf74d58b2-Abstract-Conference.html.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge distillation of large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=5h0qf7IBZZ.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nat.*, 645(8081):633–638, 2025a. doi: 10.1038/S41586-025-09422-Z. URL https://doi.org/10.1038/s41586-025-09422-z.

Jialong Guo, Xinghao Chen, Yehui Tang, and Yunhe Wang. SlimLLM: Accurate structured pruning for large language models. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025b. URL https://proceedings.mlr.press/v267/guo25a.html.

Shuo He, Lang Feng, Qi Wei, Xin Cheng, Lei Feng, and Bo An. Hierarchy-of-groups policy optimization for long-horizon agentic tasks. *CoRR*, abs/2602.22817, 2026. doi: 10.48550/ARXIV.2602.22817. URL https://doi.org/10.48550/arXiv.2602.22817.

Lanxiang Hu, Tajana Rosing, and Hao Zhang. TrimLLM: Progressive layer dropping for domain-specific LLMs. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pp. 667–681. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.ACL-LONG.33. URL https://doi.org/10.18653/v1/2025.acl-long.33.

Minki Kang, Wei-Ning Chen, Dongge Han, Huseyin A. Inan, Lukas Wutschitz, Yanzhi Chen, Robert Sim, and Saravan Rajmohan. ACON: optimizing context compression for long-horizon LLM agents. *CoRR*, abs/2510.00615, 2025. doi: 10.48550/ARXIV.2510.00615. URL https://doi.org/10.48550/arXiv.2510.00615.

Mike Lasby, Ivan Lazarevich, Nish Sinnadurai, Sean Lie, Yani Ioannou, and Vithursan Thangarasa. REAP the experts: Why pruning prevails for one-shot MoE compression. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=ukGxWd2aDG.

Gengsheng Li, Mao Zheng, Mingyang Song, Ruiqi Liu, Tianyu Yang, Jie Sun, Qiyong Zhong, Haiyun Guo, Junfeng Fang, Dan Zhang, and Jinqiao Wang. On-policy distillation with curriculum turn-level guidance for multi-turn agents. *CoRR*, abs/2606.15912, 2026a. doi: 10.48550/ARXIV.2606.15912. URL https://doi.org/10.48550/arXiv.2606.15912.

Meng Li, Lei Li, Xiting Wang, Yi Yuan, Zheng Wei, Jiang Bian, and Zang Li. SOAR: supervision from observation for agentic reinforcement learning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2026, San Diego, California, United States, July 2-7, 2026*, pp. 35175–35197. Association for Computational Linguistics, 2026b. doi: 10.18653/V1/2026.ACL-LONG.1624. URL https://doi.org/10.18653/v1/2026.acl-long.1624.

Yaxuan Li, Yuxin Zuo, Bingxiang He, Jinqian Zhang, Chaojun Xiao, Cheng Qian, Tianyu Yu, Huan-ang Gao, Wenkai Yang, Zhiyuan Liu, and Ning Ding. Rethinking on-policy distillation of large language models: Phenomenology, mechanism, and recipe. *CoRR*, abs/2604.13016, 2026c. doi: 10.48550/ARXIV.2604.13016. URL https://doi.org/10.48550/arXiv.2604.13016.

Yuanzhe Li, Jianing Deng, Jingtong Hu, Tianlong Chen, Song Wang, and Huanrui Yang. Dynamic mixed-precision routing for efficient multi-step LLM interaction. *arXiv preprint arXiv:2602.02711*, 2026d. URL https://arxiv.org/abs/2602.02711.

Gui Ling, Ziyang Wang, Yuliang Yan, and Qingwen Liu. SlimGPT: Layer-wise structured pruning for large language models. In *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/c1c44e46358e0fb94dc94ec495a7fb1a-Abstract-Conference.html.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. LLM-Pruner: On the structural pruning of large language models. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/44956951349095f74492a5471128a7e0-Abstract-Conference.html.

Saurav Muralidharan, Sharath Turuvekere Sreenivas, Raviraj Joshi, Marcin Chochowski, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, Jan Kautz, and Pavlo Molchanov. Compact language models via pruning and knowledge distillation. In *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/4822991365c962105b1b95b1107d30e5-Abstract-Conference.html.

Rui Pan, Shivanshu Shekhar, Boyao Wang, Shizhe Diao, Jipeng Zhang, Xingyuan Pan, Renjie Pi, and Tong Zhang. Adapt-Pruner: Adaptive structural pruning for efficient small language model training, 2025. URL https://arxiv.org/abs/2502.03460.

Qiuyi Qi, Tian Liang, Mutian Bao, Jinjian Zhang, Dongnan Liu, Wei Zhou, Linjian Mo, Ming Kong, Jie Liu, Feng Zhang, and Qiang Zhu. STAPO: selective trajectory-aware policy optimization for LLM agent training. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2026, San Diego, California, United States, July 2-7, 2026*, pp. 28371–28392. Association for Computational Linguistics, 2026. doi: 10.18653/V1/2026.ACL-LONG.1308. URL https://doi.org/10.18653/v1/2026.acl-long.1308.

Yulei Qin, Xiaoyu Tan, Zhengbao He, Gang Li, Haojia Lin, Zongyi Li, Zihan Xu, Yuchen Shi, Siqi Cai, Renting Rui, Shaofei Cai, Yuzheng Cai, Xuan Zhang, Sheng Ye, Ke Li, and Xing Sun.

Learn the ropes, then trust the wins: Self-imitation with progressive exploration for agentic reinforcement learning. *CoRR*, abs/2509.22601, 2025. doi: 10.48550/ARXIV.2509.22601. URL https://doi.org/10.48550/arXiv.2509.22601.

Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, volume 15 of *JMLR Proceedings*, pp. 627–635. JMLR.org, 2011. URL http://proceedings.mlr.press/v15/ross11a/ross11a.pdf.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew J. Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=0IOX0YcCdTn.

Tycho F. A. van der Ouderaa, Markus Nagel, Mart van Baalen, Yuki M. Asano, and Tijmen Blankevoort. The LLM surgeon. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=DYIIRgwg2i.

Jiaqi Wang, Wenhao Zhang, Weijie Shi, Yaliang Li, and James Cheng. TCOD: exploring temporal curriculum in on-policy distillation for multi-turn autonomous agents. *CoRR*, abs/2604.24005, 2026a. doi: 10.48550/ARXIV.2604.24005. URL https://doi.org/10.48550/arXiv.2604.24005.

Yuhang Wang, Yuling Shi, Mo Yang, Rongrui Zhang, Shilin He, Heng Lian, Yuting Chen, Siyu Ye, Kai Cai, and Xiaodong Gu. SWE-Pruner: Self-adaptive context pruning for coding agents. *CoRR*, abs/2601.16746, 2026b. doi: 10.48550/ARXIV.2601.16746. URL https://doi.org/10.48550/arXiv.2601.16746.

Yuxin Wang, Minghua Ma, Zekun Wang, Jingchang Chen, Liping Shan, Qing Yang, Dongliang Xu, Ming Liu, and Bing Qin. CFSP: an efficient structured pruning framework for LLMs with coarse-to-fine activation information. In *Proceedings of the 31st International Conference on Computational Linguistics, COLING 2025, Abu Dhabi, UAE, January 19-24, 2025*, pp. 9311–9328. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.coling-main.626/.

Jiateng Wei, Quan Lu, Ning Jiang, Siqi Li, Jingyang Xiang, Jun Chen, and Yong Liu. Structured optimal brain pruning for large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pp. 13991–14007. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.EMNLP-MAIN.775. URL https://doi.org/10.18653/v1/2024.emnlp-main.775.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/82ad13ec01f9fe44c01cb91814fd7b8c-Abstract-Conference.html.

Mingyang Zhang, Hao Chen, Chunhua Shen, Zhen Yang, Linlin Ou, Xinyi Yu, and Bohan Zhuang. LoRAPrune: Structured pruning meets low-rank parameter-efficient fine-tuning. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, volume ACL 2024 of *Findings of ACL*, pp. 3013–3026. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.FINDINGS-ACL.178. URL https://doi.org/10.18653/v1/2024.findings-acl.178.

Qingyu Zhang, Qianhao Yuan, Hongyu Lin, Yaojie Lu, Xianpei Han, Le Sun, Ming Xu, and Jiarui Li. ShortOPD: Recovering pruned LLMs with short-to-long on-policy distillation. *arXiv preprint arXiv:2607.13124*, 2026. URL https://arxiv.org/abs/2607.13124.

Shuai Zhen, Yanhua Yu, Ruopei Guo, Nan Cheng, and Yang Deng. Hierarchical reinforcement learning with augmented step-level transitions for LLM agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2026, San Diego, California, United States, July 2-7, 2026*, pp. 7035–7053. Association for Computational Linguistics, 2026. doi: 10.18653/V1/2026.ACL-LONG.318. URL https://doi.org/10.18653/v1/2026.acl-long.318.

Yuhang Zhou, Kai Zheng, Haoling Li, Dengyun Peng, Can Xu, and Jingjing Chen. TurnOPD: Making on-policy distillation turn-aware for efficient long-horizon agent training. *CoRR*, abs/2607.05804, 2026. doi: 10.48550/ARXIV.2607.05804. URL https://doi.org/10.48550/arXiv.2607.05804.

Michael Zhu and Suyog Gupta. To prune, or not to prune: Exploring the efficacy of pruning for model compression. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=Sy1iIDkPM.

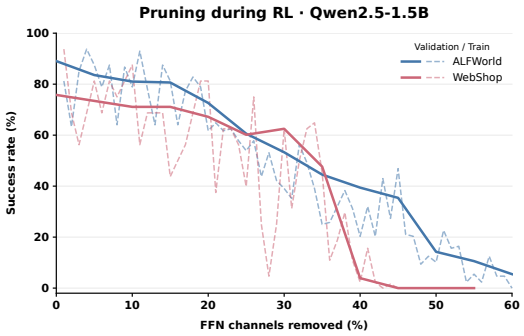

Figure 3: Pruning during GiGPO continuation on Qwen2.5-1.5B agents, which removes one additional percentage point of FFN channels per optimizer step. Solid lines show periodic validation and dashed lines training-batch success. ALFWorld validation pools the 140 In and 134 Out tasks; WebShop uses 128 fixed tasks. Validation uses temperature 0.4. Validation success falls as removal increases and drops sharply beyond about 30% removal on WebShop, so continued RL does not keep the pruned agent functional under this schedule.

Table 5: SlimGPT on SPEAR ALFWorld 1.5B at 20% FFN channel removal. Success rates are percentages.

ρ	FFN width	Split	Mean $\pm$ std
20%	8960 $\rightarrow$ 7168	In	87.62 ± 2.51
20%	8960 $\rightarrow$ 7168	Out	79.85 ± 2.59

A Pruning with Continued RL

We evaluate pruning during RL on the SPEAR Qwen2.5-1.5B checkpoints using GiGPO continuation with rank-8 LoRA and learning rate 10^{-5}. Starting from the dense checkpoint, each optimizer step removes an additional one percentage point of FFN channels using the baseline CFSP channel score. Channel masks are enforced during training. Online RL interacts with the environment and is not bound by the distillation-token budget. Figure 3 shows success during training, and Table 1 reports the evaluations at 40% and 60% removal.

B Supplementary Results

B.1 SlimGPT at Lower Pruning Ratios on SPEAR-1.5B

Table 5 reports the 20% FFN-removal result on the SPEAR ALFWorld 1.5B checkpoint, using SlimGPT local reconstruction. At 40% removal, the FFN width decreases from 8960 to 5376, and In and Out success rates are zero; these results appear in Table 1.

B.2 Channel-Mask Changes under Recalibration

We compare the final channel masks from three-stage cubic pruning with recalibration (C) against one-shot selection from the same initial perturbed-student Taylor ranking (A), at matched final removal ratios, using the Qwen2.5-7B runs of Table 2.

Let U be the set of all eligible FFN channels, indexed by layer and original dense-model channel index, and let $D_A, D_C \subset U$ denote the equally sized deleted-channel sets. We report their Jaccard similarity $J_D = |D_A \cap D_C|/|D_A \cup D_C|$, the deleted-channel replacement fraction $R_D = |D_A \setminus D_C|/|D_A|$, and the retained-channel replacement fraction $R_K = |K_A \setminus K_C|/|K_A|$, where $K_A = U \setminus D_A$ and $K_C = U \setminus D_C$. Counts are pooled across layers before taking ratios.

Table 6: Final channel masks of TAP (C) versus one-shot selection from the same initial ranking (A) on Qwen2.5-7B. J_D is the Jaccard similarity of the deleted sets; R_D and R_K are the fractions of A's deleted and retained channels that C replaces. All values are percentages. Although the deleted sets largely overlap, recalibration replaces 14–22% of the retained channels.

Environment	Removal ρ	Jaccard J_D	Deleted R_D	Retained R_K
ALFWorld	60%	82.66	9.49	14.24
WebShop	60%	82.03	9.87	14.81
ALFWorld	80%	89.44	5.58	22.31
WebShop	80%	90.13	5.19	20.76

Table 7: Effect of the calibration objective on Qwen2.5-7B at 60% FFN channel removal on ALFWorld. NLL replaces KL for channel-importance estimation while the rest of TAP is unchanged. KL yields similar in-distribution but higher out-of-distribution success.

Calibration objective	In SR (%)	Out SR (%)
KL (TAP)	$\mathbf{92.62}_{\pm 1.65}$	$\mathbf{89.80}_{\pm 1.55}$
NLL	$91.90_{\pm 1.80}$	$84.33_{\pm 2.99}$

B.3 Decreasing KD Loss Does Not Ensure Executable Agent Behavior

We inspect the WebShop Qwen2.5-1.5B agents at 60% FFN removal, where the one-shot KD baselines and TAP share a budget of 0.35M supervised response tokens (Table 9). For CFSP+KD, the response-token-weighted forward KL over the first and last 20 updates decreases from 1.580 to 0.250; for LLM-Pruner+KD, it decreases from 2.062 to 0.326.

Despite decreasing teacher-forced loss, the baselines frequently fail to produce complete, non-empty `<action>` blocks, and the environment marks most executed actions invalid (Table 8). These observations show that improved fitting under teacher prefixes can coexist with malformed free generation.

B.4 Iterative Pruning Ablation Protocol

In Table 2, all variants share the dense initialization, initial response-only reverse-KL Taylor scores, anchor-context pool, final removal ratio, and total recovery-token budget Q. A uses 20% of Q for KD and 80% for anchored OPD after its single cut. B, C, and U divide Q equally across three iterations. Each iteration uses 20% KD warm-up followed by 80% anchored OPD. U uses cumulative removal ratios $\rho/3, 2\rho/3, \rho$.

C Implementation Details

C.1 Dense Checkpoints and Evaluation Protocol

We use the following author-released, environment-specific checkpoints:

- Qwen2.5-1.5B-Instruct (ALFWorld): `yolay/SPEAR-ALFWorld-DrBoT-GiGPO-1.5B`.
- Qwen2.5-1.5B-Instruct (WebShop): `yolay/SPEAR-WebShop-DrBoT-GiGPO-1.5B`.
- Qwen2.5-7B-Instruct (ALFWorld): `langfeng01/GiGPO-Qwen2.5-7B-Instruct-ALFWorld`.
- Qwen2.5-7B-Instruct (WebShop): `langfeng01/GiGPO-Qwen2.5-7B-Instruct-WebShop`.

The 1.5B checkpoints use SPEAR + Dr.BoT (GiGPO), incorporating self-imitation and trajectory replay, whereas 7B uses GiGPO (Qin et al., 2025; Feng et al., 2025); cross-size differences also

Table 8: Free-generation failure diagnostics for the same WebShop 1.5B setting. Percentages are computed on the 500 WebShop test goals. Action-block and invalid-action rates are computed over steps; repeated-action and step-limit rates are computed over episodes. Repeated action denotes at least three consecutive identical executed actions within an episode. Flags can overlap.

	Steps (%)		Episodes (%)	
Method	Missing/broken action block	Invalid action	Repeated action	15-step limit
CFSP+KD	87	95	58	84
LLM-Pruner+KD	93	93	82	98
TAP	< 1	< 1	4	4

reflect different RL recipes. Evaluation episodes are capped at 50 steps on ALFWorld and 15 steps on WebShop, and decoding uses temperature 0.4.

C.2 FFN Channel Removal and Pruning Ratio

For a gated FFN at layer ℓ, write

$$u_\ell(z) = \sigma(W_\ell^g z) \odot (W_\ell^u z), \qquad \mathrm{FFN}_\ell(z) = W_\ell^d u_\ell(z), \tag{6}$$

where σ is the SiLU activation, $W_\ell^g, W_\ell^u \in \mathbb{R}^{m_\ell \times d}$ and $W_\ell^d \in \mathbb{R}^{d \times m_\ell}$. Removing intermediate channel j deletes row j of both input projections and column j of the output projection. For retained indices I_ℓ, the resulting module is

$$\mathrm{FFN}'_\ell(z) = W_\ell^d[:, I_\ell]\,[\sigma(W_\ell^g[I_\ell, :]z) \odot (W_\ell^u[I_\ell, :]z)]\,. \tag{7}$$

The tensors are physically resized, so the model runs with smaller dense operations. We report the cumulative channel-removal ratio

$$\rho = 1 - \frac{\sum_\ell |I_\ell|}{\sum_\ell m_\ell}. \tag{8}$$

Because only FFN channels are removed, ρ exceeds the total parameter reduction reported in Table 4.

C.3 Channel Importance Scoring

The calibration batch $\mathcal{B}$ in Eq. (4) is $\mathcal{B}_{\text{last}}$, the last training batch of anchored OPD, which contains anchor contexts c^T with sampled student responses $\widehat{a}$. Before each subsequent cut, we recompute the gradient of $\mathcal{L}_{\text{AOPD}}(\theta; \mathcal{B}_{\text{last}})$ at the recovered parameters with one forward–backward pass over the batch, which requires no additional generation or environment interaction. The loss averages the reverse KL over all supervised response tokens in the batch, including reasoning and action tokens and excluding prompt and padding tokens. Because recovery trains LoRA adapters, W and G in Eq. (4) refer to the effective FFN weights with the adapters merged; each cut removes the selected channels from both the base weights and the adapters, which continue training in the next stage.

Initial calibration. For the initial cut, we add Gaussian noise to the FFN weights of a separate student copy, with standard deviation 0.01 times each weight matrix's RMS. This perturbed copy generates the anchored calibration responses and supplies the gradients for initial channel scoring. The resulting channel selection is applied to the unperturbed student; later calibration stages use the recovered student without this perturbation.

C.4 Baseline Adaptations for Uniform Layer-wise Pruning

All methods, including TAP, use the same target FFN channel-removal ratio in every Transformer layer, so that comparisons isolate channel selection and recovery. For LLM-Pruner (Ma et al., 2023), we use first-order element-wise importance (Element^1) with Summation aggregation, followed by logits KD recovery. For CFSP, we replace its importance-based inter-block sparsity allocation with the uniform target while retaining its within-layer FFN channel-selection criterion (Wang et al., 2025). For SlimGPT, we replace its depth-dependent pruning allocation with the uniform target while retaining its FFN pruning and local reconstruction procedure (Ling et al., 2024). The baseline names in our tables refer to these adaptations under a shared FFN allocation.

C.5 Recovery Objectives for Ablation Studies

Let π_T be the frozen teacher and π_θ the student. A response $a = (y_1, \ldots, y_{|a|})$ is conditioned on an interaction context $c = (h_{<t}, o_t)$, with token prefix $p_j = (c, y_{<j})$ and prompt formatting implicit. For a batch $\mathcal{B}$ of context–response pairs, let $N_{\mathcal{B}} = \sum_{(c,a)\in\mathcal{B}} |a|$ count valid response tokens. All objectives exclude prompt, environment-observation, and padding tokens from supervision.

Logits KD. We use fixed teacher-generated pairs $\mathcal{B}_T = \{(c^T, a^T)\}$ and match the teacher's full-vocabulary distribution on teacher response prefixes $p_j^T = (c^T, y_{<j}^T)$ using forward KL:

$$\mathcal{L}_{\mathrm{KD}}(\theta) = \frac{1}{N_{\mathcal{B}_T}} \sum_{(c^T,a^T)\in\mathcal{B}_T} \sum_{j=1}^{|a^T|} D_{\mathrm{KL}}\big(\pi_T(\cdot \mid p_j^T) \,\|\, \pi_\theta(\cdot \mid p_j^T)\big)\,. \tag{9}$$

Sequence KD (SeqKD). We use the same fixed teacher-generated pairs, but supervise the teacher's response tokens as hard targets:

$$\mathcal{L}_{\mathrm{SeqKD}}(\theta) = -\frac{1}{N_{\mathcal{B}_T}} \sum_{(c^T,a^T)\in\mathcal{B}_T} \sum_{j=1}^{|a^T|} \log \pi_\theta(y_j^T \mid p_j^T). \tag{10}$$

Vanilla OPD (full-trajectory OPD). We sample complete student trajectories $\tau \sim P_{\pi_{\bar\theta},\mathcal{E}}(\cdot \mid x)$ on training tasks x, where $\bar\theta$ denotes the student at sampling time and $\mathcal{E}$ is the environment. Student actions are executed, so subsequent interaction contexts c_t^S depend on the student's preceding actions and resulting observations. We collect $\mathcal{B}_S = \{(c_t^S, a_t^S) : (c_t^S, a_t^S) \in \tau\}$ from both successful and failed rollouts and minimize reverse KL on student response prefixes $p_{t,j}^S = (c_t^S, y_{t,<j}^S)$:

$$\mathcal{L}_{\mathrm{OPD}}(\theta) = \frac{1}{N_{\mathcal{B}_S}} \sum_{(c_t^S,a_t^S)\in\mathcal{B}_S} \sum_{j=1}^{|a_t^S|} D_{\mathrm{KL}}\big(\pi_\theta(\cdot \mid p_{t,j}^S) \,\|\, \pi_T(\cdot \mid p_{t,j}^S)\big)\,. \tag{11}$$

C.6 Run Configuration and Data Splits

Calibration and recovery use training tasks only. The teacher pool is fixed across methods and stages. Response and episode caps, the action parser, and context handling are shared across comparisons.

TAP recovery and calibration settings. Anchored OPD samples 16 prompts per batch and generates one response per prompt with temperature 0.7 and top-$p = 0.9$. LoRA uses rank 8, alpha 16, and dropout 0. We optimize the LoRA parameters with AdamW at a constant learning rate of 10^{-5}. KL temperature is 1, independently of the generation temperature. Supervision covers valid response tokens and excludes prompt, environment-observation, and padding tokens. For Taylor calibration, we use 16 anchored responses at the first cut and the last batch of 16 recovery responses before each subsequent cut.

C.7 Recovery Budgets and Controlled Comparisons

Within each model–environment–removal setting, all distillation-based recovery methods in Tables 1 and 3 share the same supervised budget Q (Table 9), an effective batch size of 16 prompt–response pairs, and the same LoRA, optimizer, and KL-temperature settings as TAP (Appendix C.6). The pruning baselines compute channel importance and reconstruction statistics on the same teacher trajectories that form TAP's anchor pool, and full-trajectory OPD samples its rollouts with the same temperature and top-p as anchored OPD. Q is the total number of response tokens obtained by splitting the selected teacher episodes into per-turn prompt–response pairs. It includes reasoning, action, and naturally terminating EOS tokens, and excludes prompts and padding. Every method stops after Q supervised response tokens; for anchored OPD, these are student-response tokens, and the final batch is masked to the remaining budget. Iterative pruning divides Q equally across three stages, each with 20% KD warm-up and 80% anchored OPD.

Table 9: Recovery budget for each main-table setting. All distillation-based methods within a setting share the supervised response-token budget Q; total tokens also count the unsupervised prompts.

Model	Environment	Removal	Episodes	Response tokens Q (M)	Total tokens (M)
1.5B	ALFWorld	40%, 60%	500	0.86	5.65
1.5B	WebShop	40%	500	0.18	3.10
1.5B	WebShop	60%	1,000	0.35	6.18
7B	ALFWorld	60%, 80%	500	0.48	3.72
7B	WebShop	60%	500	0.60	3.96
7B	WebShop	80%	1,000	1.21	7.96

Table 3 compares recovery recipes under the same cubic schedule and total response-token budget. The OPD recipes use 20% KD warm-up followed by 80% reverse-KL OPD within each stage; the KD control uses the entire stage budget for forward-KL distillation. SeqKD allocates the full stage budget to response-only cross-entropy on fixed teacher-generated trajectories, using hard teacher tokens as targets.

C.8 Efficiency Measurement Protocol

Model inference. We benchmark dense and pruned models on the same NVIDIA RTX A6000 GPU using BF16, PyTorch 2.8.0, and Transformers 4.57.3 with SDPA. Each input contains 512 tokens, with batch size 1 and 128 generated tokens. We average three synchronized measurements after one warm-up, excluding model loading. All models receive identical inputs and generate the full output length. Prefill measures prompt processing; decoding averages the subsequent 127 cached generation steps.

Agent execution and cost per success. Agent evaluations use vLLM on the same A6000 host with matched evaluator settings and task sets, pooling evaluation time and success counts. End-to-end time T spans initialization, model loading, inference, environment execution, and output collection. Each evaluation uses one GPU. For N attempted tasks and S successes, we report amortized time per task T/N and cost per success T/S, the GPU time per successful task including failed attempts. End-to-end speedup is the dense-to-pruned ratio of T/N; relative CPS is the pruned-to-dense ratio of T/S. These are aggregate costs over each full evaluation.