DIVOPD: SPREAD WIDE, LOOK CLOSE FOR ASYNCHRONOUS ON-POLICY DISTILLATION OF MULTI-TURN AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

On-policy distillation (OPD) trains student agents through teacher supervision on their own interactions with an environment. However, in asynchronous multi-turn training, arrival-order batching can allow a few early or long rollouts to dominate learner updates while other valid rollouts become stale before being used, wasting already-generated experience. To address this problem, we introduce **DivOPD**, a simple learner-side batch-selection method that spreads a fixed turn budget across more rollouts and, within each rollout, prioritizes turns with larger cumulative teacher–student disagreement. Turns without usable teacher feedback are excluded. The per-turn loss and optimizer remain fixed; selection only changes which student-visited turns receive training weight. For no-progress rollouts, an optional extension briefly hands control to the teacher before returning it to the student. Across six teacher–student settings on the simulated ALFWorld, ScienceWorld, and WebShop benchmarks, with 1.5B–7B students, DivOPD raises cross-setting mean peak success rate from 77.4 to 84.4 and mean success over the last five evaluations from 71.5 to 78.6. It reaches all reported setting-specific targets with geometric-mean speedups of $1.84\times$ in training tokens and $1.87\times$ in learner GPU time relative to vanilla OPD. Teacher intervention further raises this last-five mean to 82.4 while retaining about $1.7\times$ learner-GPU speedup over vanilla OPD. Anonymized code and configuration files are included in the supplementary material.

1 INTRODUCTION

OPD trains student agents through teacher supervision on their own interactions with an environment. It is a natural fit for agents because the student acts while a frozen teacher scores the states the student actually visits (Lin et al., 2020; Agarwal et al., 2023). In contrast, distillation on fixed teacher-generated sequences (Hinton et al., 2015; Kim & Rush, 2016) does not directly train on deviations that emerge during interaction. OPD follows the online imitation-learning principle of supervising learner-visited states (Ross et al., 2011) and reduces the exposure mismatch between training and inference (Bengio et al., 2015).

However, asynchronous multi-turn OPD creates a learner-side allocation problem: generating valid experience does not ensure that the learner will use it. Explorers produce variable-length rollouts while the learner trains on queued turns; model weights synchronize periodically, and turns expire after a bounded policy age. If learner batches are filled in arrival order, a few early or long rollouts can occupy many slots while other valid rollouts become stale before contributing a gradient. The system then wastes interactions and teacher scores that it has already paid to generate.

This asynchronous execution is increasingly common as foundation-model post-training targets agents that reason, call tools, and interact with environments over long, variable-duration trajectories (Xiao et al., 2026; Ma et al., 2026; Kimi Team, 2026; Gao et al., 2025). Strict rollout–update barriers are inefficient because an entire update can wait for slow generation, tool execution, or teacher scoring. Recent large-scale systems therefore span fully asynchronous rollout and learning (Fu et al., 2025; Hu et al., 2026), asynchronous teacher prefill overlapped with rollout generation (Ma et al., 2026), and partial rollouts that continue across training iterations (Xiao et al., 2026; Kimi Team, 2026; Zhou et al., 2025). These designs differ in synchronization, but they make experience freshness a key constraint. AsyncOPD exposes the same trade-off directly for distillation:

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.

separating rollout generation from learning improves throughput but creates experience from older policies (Kang et al., 2026).

Unlike single-turn OPD, where each prompt produces one self-contained completion, agentic OPD produces a multi-turn rollout: every action changes the environment state and therefore the observations and decisions available at later turns. Rollouts also vary in the number and latency of model calls, tool executions, and environment steps. As a result, their turns arrive as correlated, variable-length groups tied to the same policy version, rather than as unrelated single-turn examples.

Vanilla OPD does not account for this grouped structure: it flattens each multi-turn rollout into individual rows and fills learner batches in arrival order, discarding the rollout as the natural allocation unit. Under a bounded staleness window, batch construction therefore becomes an online allocation decision that determines which rollouts receive any gradient before their turns expire. It can also spend scarce slots on rows without a usable teacher score.

Recent agentic OPD methods primarily improve experience before it reaches the learner. They control rollout depth with staged schedules, early stopping, or probes; rebalance supervision across turns; or alter visited states through scheduled teacher intervention (Wang et al., 2026c; Ziheng et al., 2026; Zhou et al., 2026; Li et al., 2026; Chen et al., 2026a). These mechanisms address what experience is generated or how each turn is supervised, often through rollout- or task-dependent schedules. They do not directly address the separate learner-side question: how should a fixed batch budget be allocated across already-generated, correlated rollouts before they become stale? Better supervision cannot help a rollout that never reaches an optimizer update.

Other asynchronous OPD methods correct, filter, or assign less weight to stale samples (Kang et al., 2026; Rang et al., 2026; Chen et al., 2026b). Open-MOPD identifies a related token-budget imbalance across domains and refreshes stale rewards across repeated inner updates (Gao et al., 2026). These methods control domain balance, sample freshness, or policy drift; they do not model how correlated turns from multiple agent rollouts within one learner queue compete for a fixed batch. Rollout-level allocation therefore has not been addressed even when staleness is bounded.

Our training logs show that this allocation failure is large. Vanilla OPD batches draw only 3.9–8.3 effective rollouts, their largest rollout supplies 24–40% of batch tokens, and shared-buffer replays find that 49–59% of valid rollouts never enter an optimizer batch (Figure 2, Appendix Table 11). We therefore introduce **DivOPD**, which treats learner batch construction as online allocation over correlated, expiring rollouts rather than flat sampling over turns (Figure 1). It removes turns without usable supervision, limits each rollout's first-pass contribution, and selects high-disagreement turns within each rollout. The last two steps serve different roles. Coverage determines how many slots each rollout receives, while focus chooses which turns fill those slots. The selector reuses cached scores without an extra model pass and leaves vanilla OPD's rollout generation, per-turn loss, and optimizer unchanged.

When a rollout repeatedly makes no progress, selecting among its turns does not expose the student to new states. We therefore extend DivOPD with bounded teacher recovery (+R): the teacher briefly intervenes, then the student's subsequent turns return to the same batch composer.

Contributions. **(1)** We measure how arrival-order batching uses student experience in asynchronous multi-turn OPD: it concentrates batches on a few rollouts, spends slots on unscored rows, and lets many rollouts expire untrained (Section 2.2, Appendix Table 11). **(2)** We introduce DivOPD, a rollout-aware batch composer that retains the per-turn distillation loss and optimizer while changing which scored turns receive training weight (Section 3). Bounded teacher recovery extends the same composer to help students resume after repeated no-progress turns. **(3)** Across six teacher–student settings, we show success-rate gains and geometric-mean trained-token and trainer-GPU speedups of $1.84\times$ and $1.87\times$ over vanilla OPD. Controlled training runs and shared-buffer replays separate the effects of coverage and focus (Section 4).

2 Asynchronous Multi-turn OPD and Batch Allocation

2.1 Learning from queued student turns

Asynchronous multi-turn OPD. Rollout workers generate student interactions and frozen-teacher scores independently of learner updates, adding turns to a queue with rollout IDs and policy versions. The learner forms fixed-size batches, discarding turns whose generating policy is too old. A rollout is $\rho = (o_1, a_1, \ldots, o_{T_\rho}, a_{T_\rho})$, whose horizon T_ρ depends on the outcome. We write π_θ for the

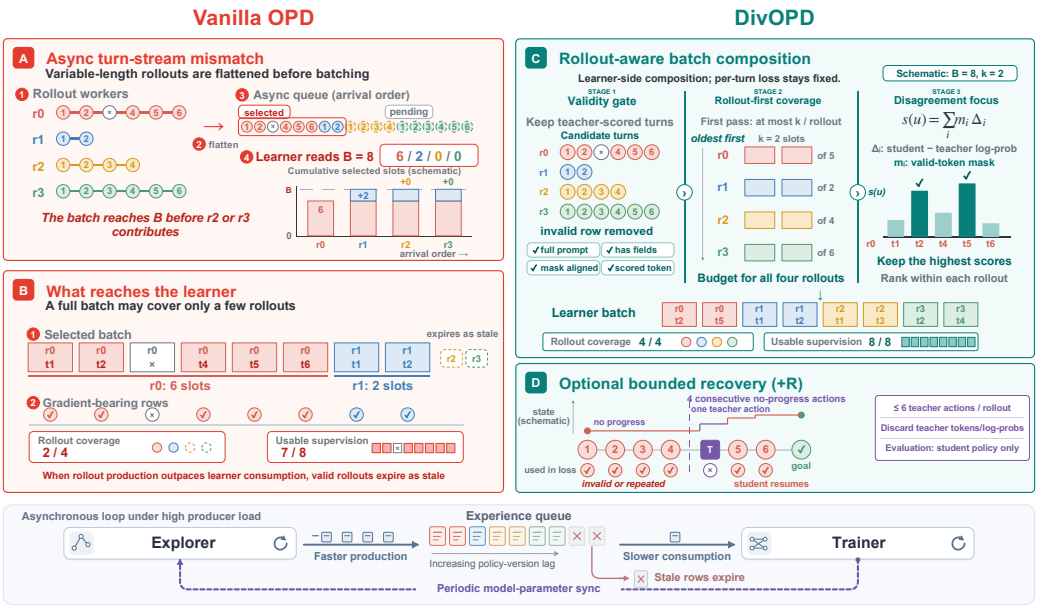

Figure 1: **Batch composition in asynchronous multi-turn OPD.** (A) Rollout workers enqueue variable-length interactions in arrival order. (B) When production outpaces learner consumption, early or long rollouts can fill the batch, include unscored rows, and leave others to expire as stale. (C) DivOPD composes batches on the learner side through the three stages in Section 3: validity gate, rollout-first coverage, and within-rollout disagreement focus using $s(u)$ (Equation 2). (D) When rollouts repeatedly make no progress, bounded teacher recovery helps the student continue; teacher actions are excluded from the loss, and recovery is disabled at evaluation. Bottom: independent exploration and training, with periodic model synchronization and stale-row expiry.

student and π_ϕ for the frozen teacher, reserving τ for the target success rate. A batch contains $N = B$ rows, one per student response turn; B is the learner's per-update budget. For row t, let $y_{t,i}$ be the i-th response token and $h_{t,<i}$ the prompt and tokens preceding it. The effective mask is $m_{t,i} = m^{\text{resp}}_{t,i} \wedge m^{\text{valid}}_{t,i}$ with valid-token set $V_t = \{i : m_{t,i} = 1\}$: tokens that belong to the response and have valid teacher scores. Rows with $|V_t| = 0$ are *dead*. With distillation coefficient β, the stopped per-token advantage is

$$A_{t,i} = \beta \, \text{sg}[\log \pi_\phi(y_{t,i} \mid h_{t,<i}) - \log \pi_{\theta_{\text{old}}}(y_{t,i} \mid h_{t,<i})], \qquad i \in V_t. \tag{1}$$

Here θ_{old} is the learner snapshot immediately before the update; its log probabilities are recomputed on queued tokens, rather than taken from their generating policy. sg stops gradients through the score difference. The loss averages valid tokens within each row, then averages all N rows (seq-mean-token-mean). A dead row contributes zero but stays in the denominator. Each batch receives one optimizer update, so the learner-snapshot ratio is one at gradient evaluation and PPO clipping is inactive; there is no entropy term, KL penalty, critic, or reference model (Appendix E).

2.2 Where the batch budget goes

Each queued turn is identified by rollout ID and turn index (rid, t). We ask two questions: how many rollouts are represented in each batch, and how many are ever used before they expire? For a batch, let p_r be the fraction of tokens from rollout r. The effective rollout count $(\sum_r p_r^2)^{-1}$ measures token-weighted coverage, and $\max_r p_r$ measures the largest rollout's share.

Concentration within an update. At the setting level, vanilla OPD batches contain only 3.9–8.3 effective rollouts, and the largest rollout supplies 24–40% of batch tokens. In addition, 5.0–27.8% of selected rows on ALFWorld and WebShop have no teacher-scored token in the final training window. All ScienceWorld rows pass the validity checks, allowing us to examine allocation without dead-slot removal.

Rollouts that expire unused. Turns from one rollout arrive together. With a finite staleness window, repeatedly selecting from early rollouts leaves less time to use later arrivals. Shared-buffer replays show that roughly half of valid rollouts are never selected by the arrival-order reader (Appendix

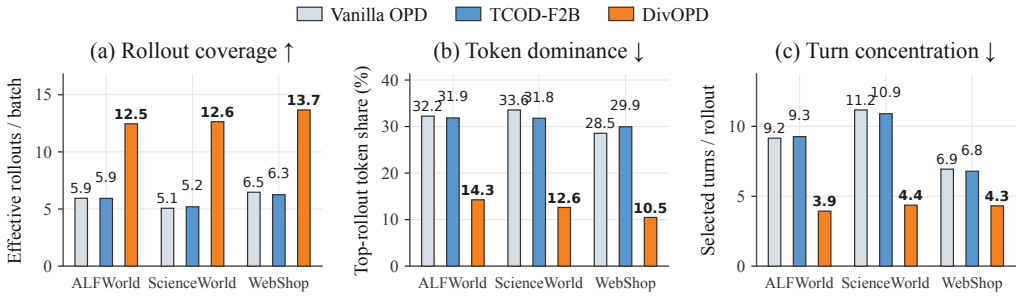

Figure 2: **DivOPD changes batch composition across environments.** Each bar averages the final-40% diagnostic over the two student sizes in that environment. DivOPD covers more rollouts and reduces both concentration measures in all three environments; TCOD-F2B remains close to vanilla OPD. The six setting-level results appear in Appendix Table 9.

Table 11). This motivates spreading batch slots across rollouts before ranking their turns. Section 4 tests whether changing this allocation improves learning.

3 DivOPD

Motivated by the batch composition in Figure 2, DivOPD composes learner batches from this asynchronous queue of already scored turns. It first combines turns retained from earlier updates (pending turns) with new arrivals in a pool of at most cB turns. From this pool, the three stages shown from left to right in Figure 1C remove invalid turns, allocate slots across rollouts, and rank turns within each rollout. The batch contains B valid turns whenever the pool has enough.

3.1 Stage 1: validity gate

A turn is valid only if its prompt was not truncated and it contains the tokens, teacher log probabilities, and action mask needed by the loss. If a teacher-valid mask is supplied, it must align with the action mask and share at least one valid position. Turns that fail these checks are discarded before batch selection.

In four ALFWorld replay buffers, every rejected row is a response turn with a truncated prompt from a failed rollout and zero scored tokens. Removing these rows frees slots for valid turns. Longer context windows or truncating the history before generation could also prevent such rows (Appendix E.4).

3.2 Stage 2: rollout-first coverage

To prevent one long rollout from filling the batch, the composer groups valid turns by rollout ID and visits the longest-waiting groups first. On the first pass through these groups, each rollout contributes at most k turns, which guarantees $\lceil B/k \rceil$ distinct rollouts whenever that many groups are available. If the batch is not full, each additional pass raises the cap by one. Thus k limits the first pass, not the final number of turns per rollout. Unlike shuffling, this changes which turns enter the batch, not just their order.

Unselected valid turns are kept for later updates until they expire. This allows more rollouts to enter each batch and gives waiting rollouts another chance to be used. The cap counts turns, not tokens, so it does not bound a rollout's token share when response lengths vary.

3.3 Stage 3: within-rollout disagreement focus

Once a rollout receives slots, focus chooses the turns that fill them. Let $v(u)$ be the student policy version that generated turn u. DivOPD ranks its valid turns by the cached score over its valid tokens V_u,

$$s(u) = \sum_{i \in V_u} \left[\log \pi_{\theta_{v(u)}}(y_i \mid h_{u,<i}) - \log \pi_\phi(y_i \mid h_{u,<i}) \right], \tag{2}$$

in descending order, breaking ties by (rid, t), on every pass. The score measures cumulative, not per-token, disagreement: it equals the valid-token count times the mean token score. With a complete response mask, its expectation is the sequence-level reverse KL at the generating policy (Gu et al.,

2024; Agarwal et al., 2023); a partial mask restricts it to scored tokens. A small signed sum may occur because positive and negative token scores cancel.

Focus uses this score only to rank turns within allocated rollout slots; it neither takes slots from other rollouts nor adds a length-based loss weight. The score reuses cached log probabilities without another model pass. It is a selection heuristic, not the current learner's row loss or gradient norm: the learner can be up to two versions ahead. Appendix E.2 audits ranking stability under this lag; Section 4.4 tests its added benefit at fixed allocation.

3.4 Optional bounded teacher recovery

Batch selection cannot help a rollout that repeatedly makes no progress. DivOPD+R therefore lets the teacher act for at most m turns after p no-progress turns, then returns control to the student. Teacher actions receive no distillation weight, subsequent student turns use the same composer, and recovery is disabled during evaluation. Unlike core DivOPD, +R changes the visited trajectory. Appendix G defines the trigger and gives the per-setting patience, takeover cap, and warmup values.

3.5 Training distribution and invariants

The per-turn loss is fixed, but selection still changes what the update averages over. Coverage limits the influence of rollout length on allocation, and focus shifts weight toward high-disagreement turns inside each rollout. We do not add importance weights to undo either change; Appendix E.1 states this shift precisely.

DivOPD-base uses the same gate and rollout allocations with uniform within-rollout sampling; DivOPD replaces that sampling with focus. Both use Equation 1 with unit per-turn weights, and the same reduction, optimizer, and rollout procedure as vanilla OPD. Appendix Table 3 collects the batch, pool, staleness, pending, cap, and loss settings. Algorithm 1 gives the full procedure; core DivOPD changes only queue consumption.

4 Experiments

4.1 Setup and metrics

We use the same asynchronous explorer–learner pipeline for all methods and evaluate two student scales per environment: ALFWorld (Shridhar et al., 2020) (Qwen3-1.7B/4B), WebShop (Yao et al., 2022) (Qwen2.5-3B/7B-Instruct), and ScienceWorld (Wang et al., 2022) (Qwen2.5-1.5B/3B-Instruct). Teachers are environment-specific GiGPO-Qwen2.5-7B models (Feng et al., 2026) on ALFWorld and WebShop, and the ScienceWorld checkpoint released with Embodied-Planner-R1 (Fei et al., 2025) on ScienceWorld. Baselines are vanilla OPD (Lin et al., 2020; Agarwal et al., 2023), TCOD-F2B (Wang et al., 2026c), and TurnOPD (Zhou et al., 2026), all with the same learner batch size. DivOPD-focus is an earlier focus-first composer reported for reference; it does not share the sampler of the other DivOPD variants, so it is not a controlled ablation of focus. DivOPD+R follows Section 3.4; exact settings appear in Appendix G, and evaluation remains student-only.

Vanilla reads B candidates per update; DivOPD can inspect cB pending and fresh turns. A no-cap control with uniform within-rollout sampling matches DivOPD-base's candidate access (Section 4.4). Appendix A gives all numerical training and evaluation settings.

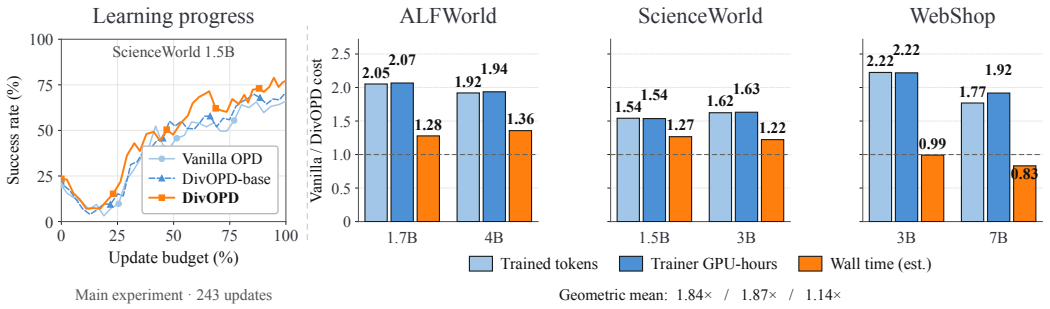

Figure 3: **Efficiency depends on which cost is measured.** Left: unsmoothed ScienceWorld 1.5B learning curves over a common 243-update budget. Right: vanilla-to-DivOPD cost ratios to the first evaluation at τ; values above one favor DivOPD. Wall time includes experience reads (Appendix Table 15).

Table 1: **Main results.** Peak/Final-5 are best-checkpoint/final-five mean SR (%); ± is the standard deviation over the last five checkpoints. Headers give teacher/student-initial SR. Tok./GPU are $C_{\text{vanilla}}(\tau)/C_{\text{method}}(\tau)$ for cumulative learner training tokens and trainer GPU-hours, respectively. Larger is better; a dash means that the method did not reach τ. Red/purple mark best/second-best; DivOPD-focus is an uncontrolled reference.

ALFWorld	**Qwen3-1.7B** (T/Init: 87.86/7.86; τ=70)				**Qwen3-4B** (T/Init: 87.86/26.43; τ=80)			
Method	**Peak↑**	**Final-5↑**	**Tok.↑**	**GPU↑**	**Peak↑**	**Final-5↑**	**Tok.↑**	**GPU↑**
Vanilla OPD	77.86	74.00 ± 3.18	1.00×	1.00×	86.43	84.57 ± 1.48	1.00×	1.00×
TCOD-F2B	72.86	68.43 ± 3.09	1.62×	1.63×	87.14	83.00 ± 1.28	1.49×	1.51×
TurnOPD	78.57	75.71 ± 2.72	1.77×	1.76×	87.14	86.43 ± 0.51	1.57×	1.59×
DivOPD-focus	80.00	71.00 ± 6.62	1.50×	1.46×	89.29	85.29 ± 3.10	1.69×	1.70×
DivOPD-base	74.29	72.86 ± 1.43	1.35×	1.35×	**92.14**	88.86 ± 2.29	1.27×	1.27×
DivOPD	79.29	71.86 ± 7.29	2.05×	2.07×	91.43	88.00 ± 2.74	**1.92**×	**1.94**×
DivOPD+R	**82.86**	**78.29 ± 1.20**	**2.17**×	**2.19**×	**92.14**	**89.00 ± 2.51**	1.90×	1.91×
WebShop	**Qwen2.5-3B** (T/Init: 85.94/1.56; τ=60)				**Qwen2.5-7B** (T/Init: 85.94/14.06; τ=85)			
Method	**Peak↑**	**Final-5↑**	**Tok.↑**	**GPU↑**	**Peak↑**	**Final-5↑**	**Tok.↑**	**GPU↑**
Vanilla OPD	60.94	43.44 ± 13.02	1.00×	1.00×	87.50	82.50 ± 3.11	1.00×	1.00×
TCOD-F2B	67.19	61.72 ± 4.42	1.64×	1.64×	89.06	83.59 ± 3.79	1.27×	1.40×
TurnOPD	68.75	58.59 ± 8.80	1.86×	1.84×	89.84	85.94 ± 2.47	1.30×	1.41×
DivOPD-focus	70.31	66.41 ± 3.45	2.37×	2.40×	89.84	86.88 ± 2.17	**1.91**×	**1.99**×
DivOPD-base	78.12	73.12 ± 4.64	1.88×	1.88×	88.28	78.44 ± 4.37	1.11×	1.22×
DivOPD	78.12	67.81 ± 3.96	2.22×	2.22×	**90.62**	82.66 ± 5.25	1.77×	1.92×
DivOPD+R	**79.69**	**76.41 ± 4.04**	**2.60**×	**2.61**×	89.84	**88.28 ± 2.71**	1.52×	1.67×
ScienceWorld	**Qwen2.5-1.5B** (T/Init: 93.75/21.48; τ=65)				**Qwen2.5-3B** (T/Init: 93.75/45.31; τ=85)			
Method	**Peak↑**	**Final-5↑**	**Tok.↑**	**GPU↑**	**Peak↑**	**Final-5↑**	**Tok.↑**	**GPU↑**
Vanilla OPD	66.02	63.75 ± 2.49	1.00×	1.00×	85.55	80.78 ± 4.09	1.00×	1.00×
TCOD-F2B	71.88	66.33 ± 3.77	1.21×	1.27×	87.89	83.98 ± 2.58	1.21×	1.20×
TurnOPD	71.48	68.75 ± 3.65	1.49×	**1.54**×	83.20	80.86 ± 2.72	–	–
DivOPD-focus	66.80	61.88 ± 4.71	0.94×	0.95×	77.34	74.69 ± 2.20	–	–
DivOPD-base	70.31	67.50 ± 2.20	1.00×	0.99×	86.72	82.66 ± 1.50	1.00×	0.99×
DivOPD	**78.91**	76.09 ± 2.23	**1.54**×	**1.54**×	88.28	**85.47 ± 2.49**	**1.62**×	**1.63**×
DivOPD+R	78.12	**77.42 ± 0.75**	1.29×	1.30×	**90.23**	85.08 ± 2.92	1.03×	1.07×

Protocol. All runs use two trainer GPUs. Full updates and GPU hours appear in Appendix Table 7.

We report peak success rate (SR) and final-five SR, the mean over the last five evaluation checkpoints; ± denotes the standard deviation across those checkpoints. For efficiency, the target success rate τ is a multiple of 5 at or below vanilla OPD's peak SR. For either cumulative learner-side resource C (training tokens or trainer GPU-hours), we report $C_{\text{vanilla}}(\tau)/C_{\text{method}}(\tau)$, where each cost is accumulated through the first evaluation reaching τ. Values above one favor the method; a dash means that the method did not reach τ. We also report normalized area under the success-rate curve over a common learner-update budget (nAUC), equivalently the mean SR over that budget, as a target-free comparison (Section 4.3). The trainer-loop wall-time estimate, which includes experience reads, improves by $1.14\times$ on geometric average, although WebShop 3B and 7B show no wall-time gain ($0.99\times$ and $0.83\times$); Appendix F gives the full breakdown.

4.2 Success rates across six settings

Table 1 compares accuracy and learner-side efficiency to target.

Across the six settings, DivOPD raises mean peak SR from 77.4 to 84.4 and final-five SR from 71.5 to 78.6. Peak SR improves in all six settings and final-five SR in five. On ALFWorld 1.7B, where late checkpoints fluctuate, DivOPD+R reaches the highest final-five SR (78.29 ± 1.20).

4.3 Learner efficiency and elapsed time

Our primary efficiency claim concerns learner-side resources. We report learner training tokens, trainer GPU-hours, and elapsed trainer-loop time separately (Figure 3, right) because reducing learner work need not accelerate the whole pipeline equally.

DivOPD reaches every target with fewer learner training tokens and trainer GPU-hours than vanilla OPD: the geometric-mean speedups are $1.84\times$ and $1.87\times$, respectively. On ALFWorld and WebShop, the gain combines fewer updates with fewer tokens per update; on ScienceWorld, where all rows are valid, it comes primarily from fewer updates. Appendix F gives the cost breakdown.

The trainer-loop wall-time estimate improves by $1.14\times$ on geometric average, but the ratio is $0.99\times$ on WebShop 3B and $0.83\times$ on WebShop 7B. Once learner updates become cheaper, rollout generation and data reads account for a larger share of elapsed time (Appendix F).

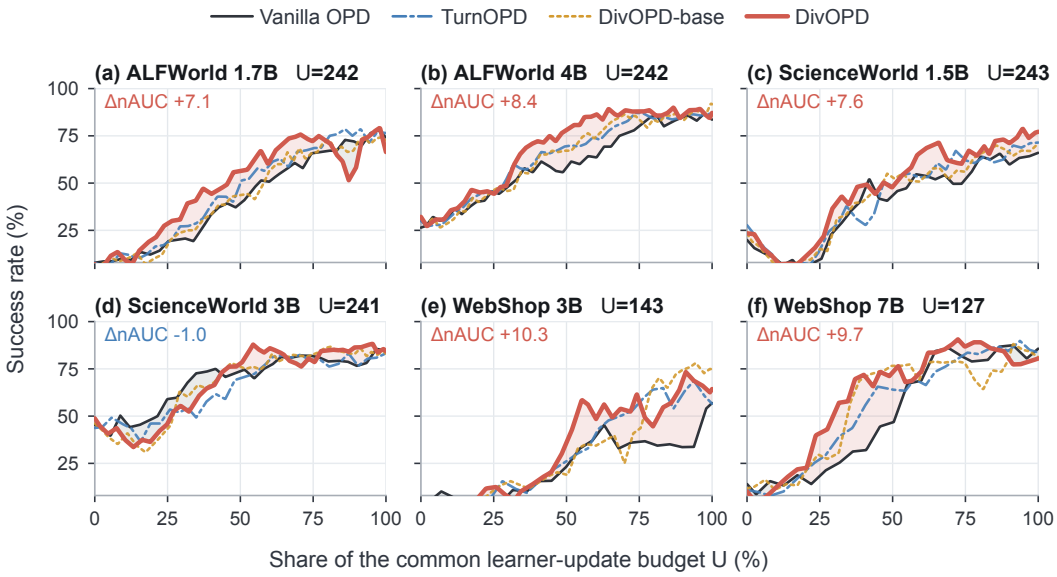

Figure 4: **Learning curves at a common update budget.** Each panel aligns the archived runs on the largest learner-update budget U they share. Curves are unsmoothed evaluation success rates. The normalized area under each curve (nAUC) is its mean SR over this budget; shading shows the area between DivOPD and vanilla OPD. Appendix B gives the protocol.

A target-free comparison. We also align the archived runs on a common learner-update budget U (Figure 4). DivOPD has the higher nAUC in five of six settings, providing evidence of improvement without selecting a target τ. Figure 3 (left) enlarges the ScienceWorld 1.5B comparison.

Sensitivity to asynchronous producer load. Table 2 varies the explorer rollout batch on ALFWorld 1.7B over 150 learner updates, with sampler settings fixed. The two readers share the producer configuration. To show the pressure that actually accumulates, we report the measured queue arrival rate divided by the learner drain rate. A value above one means experience enters the queue faster than the learner removes it; this ratio is an outcome of the reader, not a matched control. Vanilla OPD and DivOPD-base are nearly tied in final SR at light load, where both measured ratios are below one. At medium and high load, DivOPD-base raises final SR by 2.9 and 10.0 points and greatly reduces the raw stale-row count. Relative to vanilla, its nAUC is higher by 0.4, 0.7, and 0.9 points from light to high load, so the advantage is visible over the fixed training budget rather than only at the final checkpoint. This pattern is consistent with the intended use case: allocating limited learner slots when production exceeds consumption.

Table 2: **Stress test under increasing asynchronous producer load.** ALFWorld Qwen3-1.7B over 150 learner updates. Arrival/drain is the measured queue arrival rate divided by the learner drain rate; values above one indicate a growing backlog. SR and nAUC are percentages; stale rows are raw counts. Bold marks the better value within each load level.

Load	Reader	Rollout bs	Arrival/drain	Final SR↑	nAUC@150↑	Stale rows↓
Low	Vanilla OPD	2	0.77	**52.1**	25.0	240
	DivOPD-base	2	0.55	52.0	**25.4**	**53**
Medium	Vanilla OPD	6	1.35	53.6	25.3	8,035
	DivOPD-base	6	0.88	**56.5**	**26.0**	**4,237**
High	Vanilla OPD	16	2.27	49.3	26.9	30,175
	DivOPD-base	16	0.89	**59.3**	**27.8**	**9,766**

4.4 SEPARATING COVERAGE FROM FOCUS

DivOPD differs from vanilla OPD in three ways: it filters invalid turns, reads more candidates, and changes turn selection. We therefore use separate controls to test the cap and focus (Figure 5).

Cap with the same candidate access. We compare DivOPD-base with an uncapped ($k = \infty$) control. Both use the same cB candidate-pool size, validity gate, oldest-first order, pending mechanism, and uniform within-rollout sampling. Across the six settings, finite k has higher peak SR throughout (mean $+2.1$ points) and higher final-five SR in five (mean $+3.4$), with all other selection rules fixed (Table 18).

Selection with the same candidate access. We give vanilla OPD the same 256-turn candidate stream on one setting per environment and compare it with rollout-first selection (Table 17). Rollout-first selection yields peak SR 1.4–4.3 points higher and Best-5 SR 1.7–5.6 points higher in all three runs. At the common cutoff, final SR is higher on ScienceWorld, tied on ALFWorld, and lower on WebShop. Because these are separate runs truncated at a common learner-version cutoff, their absolute SR is not comparable to Table 1.

Focus with the same rollout allocation. DivOPD and DivOPD-base assign slots in the same way but choose different turns within each rollout. Focus raises mean peak SR by 2.8 points and reduces trainer-GPU time-to-target in all six settings, giving a further 1.38–1.60× dataset-mean speedup over DivOPD-base. Final-five SR rises in only three settings (mean $+1.4$ points).

Cap sensitivity. Peak SR varies by up to 7.9 points within a setting over $k \in \{3, \ldots, 6\}$ (Appendix Table 19). The best cap varies by setting, so choosing k matters.

4.5 WHICH ROLLOUTS AND TURNS ARE SELECTED?

Coverage brings more rollouts into training. We replay the selection rules on shared buffers so that every method sees the same experience. Appendix Table 11 tracks whether each valid rollout contributes any turn over 100 updates. Rollout-first selection lowers the unused fraction from 49–59% to 7–14% and reduces the concentration of selected turns. DivOPD-base and DivOPD allocate rollouts identically here; focus changes which turns are taken, not how many rollouts are reached.

Focus selects different gradients. Holding the candidate pool and the rollout allocation fixed, we compare selected gradients with the mean gradient of all valid candidate turns. Focus increases the measured projection onto this pool gradient in all 24 audited updates (Appendix E.3). This is a diagnostic before the optimizer update, not a guarantee of better learning.

In the ALFWorld 1.7B replay, focus selects earlier turns from failed rollouts without changing the success/failure mix (Appendix Table 14). Pairwise gradient cosines stay near zero for every composer: this diagnostic does not show a clear change in within-batch gradient similarity.

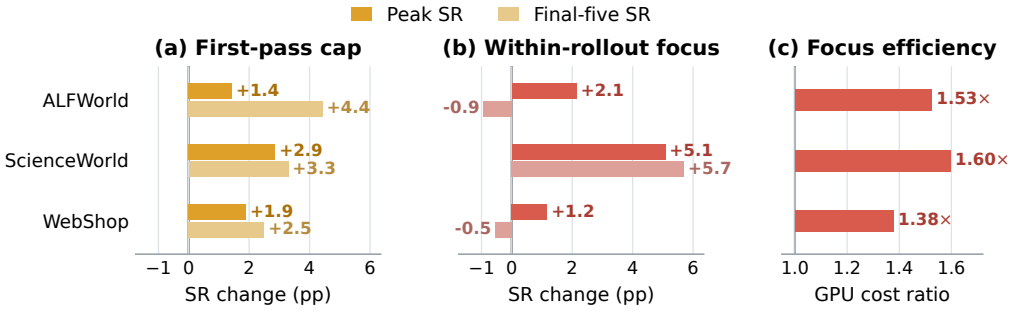

Figure 5: **The cap improves both metrics; focus improves peak SR and learner efficiency.** Each bar averages the two model sizes within one dataset. (a) DivOPD-base minus the matched-access $k = \infty$ control isolates the cap. (b) DivOPD minus DivOPD-base isolates focus at fixed rollout allocation; positive values favor focus. (c) $C_{\text{base}}(\tau)/C_{\text{DivOPD}}(\tau)$ for trainer GPU-hours; values above one mean that focus reaches τ with less trainer compute. Setting-level results are in Tables 18 and 1.

Disagreement is more evenly spread across selected rollouts. Figure 6 tracks the standard deviation of the total disagreement from each rollout in a learner batch. It is lower under DivOPD than vanilla OPD through most of training on ALFWorld and ScienceWorld.

Extending selection to no-progress rollouts. Adding recovery to DivOPD raises final-five mean SR in five of six settings (cross-setting mean 78.6 → 82.4) and peak SR in four, while retaining

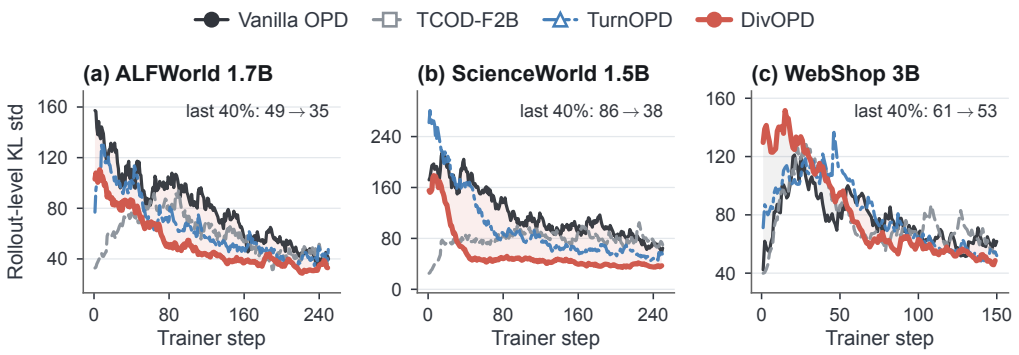

Figure 6: **Cross-rollout disagreement, one setting per environment.** Curves show the EMA ($\alpha = 0.12$) of the within-batch standard deviation of rollout-summed disagreement; labels give the final-40% mean (vanilla → DivOPD). Red/gray shading marks where DivOPD is lower/higher. All settings appear in Figure 7.

a geometric-mean trainer-GPU speedup of about $1.7\times$ over vanilla OPD. These learner-side ratios exclude teacher-generation cost (Appendix Figure 9).

5 Related Work

Agentic OPD. Generative distillation changes the sequence source or objective (Gu et al., 2024; Ko et al., 2024); TurnOPD balances turns (Zhou et al., 2026); TCOD and early stopping control rollout length (Wang et al., 2026c; Ziheng et al., 2026); and Guided-OPD and FutureBridge-OPD change visited states through teacher intervention (Li et al., 2026; Chen et al., 2026a). DivOPD instead selects which scored turns enter each asynchronous learner batch (Appendix Table 8).

Multi-teacher and asynchronous OPD. Multi-teacher OPD and Open-MOPD route supervision or balance token budgets across domains (Ma et al., 2026; Gao et al., 2026). AsyncOPD, Near-Policy, and f-OPD control policy lag through correction, filtering, or freshness (Kang et al., 2026; Rang et al., 2026; Chen et al., 2026b). These operate at the domain or sample level; DivOPD allocates a turn budget across correlated rollouts in one asynchronous learner queue.

Data selection and asynchronous learning. Prioritized selection favors informative examples (Loshchilov & Hutter, 2015; Schaul et al., 2015; Katharopoulos & Fleuret, 2018; Mindermann et al., 2022), DAPO drops zero-gradient groups (Yu et al., 2026), and asynchronous actor–learner systems control policy lag (Espeholt et al., 2018; Fu et al., 2025; Hu et al., 2026). DivOPD instead separates rollout allocation from turn selection when queued, variable-length rollouts compete for a fixed learner budget.

6 Conclusion and Future Work

Arrival-order batching in asynchronous multi-turn OPD can let early or long rollouts dominate a fixed learner budget, leaving valid experience unused. DivOPD treats this as a batch-composition problem: it filters unusable turns, caps each rollout's first-pass contribution, and focuses allocated slots on high-disagreement turns. The per-turn loss and optimizer stay unchanged; selection changes the data-weighted training objective.

Across six Qwen-family settings on three benchmarks, DivOPD raises mean peak SR from 77.4 to 84.4 and final-five SR from 71.5 to 78.6, while achieving geometric-mean speedups of $1.84\times$ in learner training tokens and $1.87\times$ in trainer GPU-hours. Controls support the cap and broader coverage; focus improves peak SR and time-to-target, while the stress test shows that gains grow with producer load. Recovery further raises mean final-five SR to 82.4 while retaining about $1.7\times$ trainer-GPU speedup. These gains require neither a learned selector nor a new per-turn loss.

Batch construction is therefore part of the learning algorithm in asynchronous multi-turn OPD, not neutral systems plumbing. Evidence remains limited to simulated benchmarks and Qwen-family models; total pipeline gains also depend on rollout and teacher costs. Future work should broaden these settings, measure end-to-end resources, and adapt allocation to queue pressure.

AI USE STATEMENT

Generative AI tools were used to draft portions of the manuscript and to assist with language polishing and organization. The authors supplied the scientific content and evidence, reviewed and revised all AI-assisted text, and verified quantitative statements against experiment logs and source artifacts. The authors take full responsibility for the final manuscript.

ETHICS STATEMENT

This work uses public simulated benchmarks and involves no human participants or personal data. It studies training efficiency rather than deployment in open-world systems. Teacher intervention can make an agent more capable, so deployment in settings with real users or high-impact actions would require task-specific safety evaluation and human review beyond the experiments reported here.

REPRODUCIBILITY STATEMENT

Section 3 specifies the composer and all of its default hyperparameters; Section 4 specifies the environments, student–teacher pairs, and the budget-to-target protocol. DivOPD-base, DivOPD, and DivOPD+R share one sampler implementation; DivOPD-focus is documented separately. Every method row of Appendix Table 7 corresponds to a single configuration file. An anonymized code snapshot and these configuration files are included in the supplementary material.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. Gkd: Generalized knowledge distillation for auto-regressive sequence models. *arXiv preprint arXiv:2306.13649*, 12, 2023.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in neural information processing systems*, 28, 2015.

Chishui Chen, Yaoyou Fan, Te Sun, Yi Yang, Chenghao Sun, Delin Mao, Hongbo Qiao, Zuowei Zhang, Junxi Wang, Chenxing Sun, et al. Look ahead before you distill: Future trajectory validation of teacher guidance for agentic on-policy distillation. *arXiv preprint arXiv:2608.01953*, 2026a.

Xianwei Chen, Shimin Zhang, and Jibin Wu. f-opd: Stabilizing long-horizon on-policy distillation with freshness-aware control. *arXiv preprint arXiv:2605.17862*, 2026b.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International conference on machine learning*, pp. 1407–1416. PMLR, 2018.

Zhaoye Fei, Li Ji, Siyin Wang, Junhao Shi, Jingjing Gong, and Xipeng Qiu. Unleashing embodied task planning ability in llms via reinforcement learning. *arXiv preprint arXiv:2506.23127*, 2025.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training. *Advances in Neural Information Processing Systems*, 38:46375–46408, 2026.

Wei Fu, Jiaxuan Gao, Xujie Shen, Chen Zhu, Zhiyu Mei, Chuyi He, Shusheng Xu, Guo Wei, Jun Mei, Jiashu Wang, Tongkai Yang, Binhang Yuan, and Yi Wu. AReaL: A large-scale asynchronous reinforcement learning system for language reasoning. *arXiv preprint arXiv:2505.24298*, 2025.

Huan-ang Gao, Haohan Chi, Yong Yan, Shiyuan Feng, Hanlin Wu, Zheng Jiang, Bingxiang He, Wei-Ying Ma, Ya-Qin Zhang, and Hao Zhou. Open-MOPD: Diagnosing and fixing capability imbalance in multi-teacher on-policy distillation. *arXiv preprint arXiv:2608.19098*, 2026.

Jiaxuan Gao, Wei Fu, Minyang Xie, Shusheng Xu, Chuyi He, Zhiyu Mei, Banghua Zhu, and Yi Wu. Beyond ten turns: Unlocking long-horizon agentic search with large-scale asynchronous RL. *arXiv preprint arXiv:2508.07976*, 2025.

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

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Tianhao Hu, Xiangcheng Liu, Yuchun Miao, Youshao Xiao, Hongyu Zang, Yang Zheng, Xuan Huang, Jinrui Ding, Yufei Zhang, Yu Yang, et al. DORA: A scalable asynchronous reinforcement learning system for language model training. *arXiv preprint arXiv:2604.26256*, 2026.

Wonjun Kang, Kevin Galim, Seunghyuk Oh, Minjun Kang, Sanghyun Park, Donghoon Kim, Minjae Lee, Minseo Kim, Rishabh Tiwari, Yuchen Zeng, Hyung Il Koo, and Kangwook Lee. AsyncOPD: How stale can on-policy distillation be? *arXiv preprint arXiv:2606.24143*, 2026.

Angelos Katharopoulos and François Fleuret. Not all samples are created equal: Deep learning with importance sampling. In *International conference on machine learning*, pp. 2525–2534. PMLR, 2018.

Yoon Kim and Alexander M Rush. Sequence-level knowledge distillation. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pp. 1317–1327, 2016.

Kimi Team. Kimi K3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653*, 2026.

Jongwoo Ko, Sungnyun Kim, Tianyi Chen, and Se-Young Yun. Distillm: Towards streamlined distillation for large language models. *arXiv preprint arXiv:2402.03898*, 2024.

Gengsheng Li, Mao Zheng, Mingyang Song, Ruiqi Liu, Tianyu Yang, Jie Sun, Qiyong Zhong, Haiyun Guo, Junfeng Fang, Dan Zhang, et al. On-policy distillation with curriculum turn-level guidance for multi-turn agents. *arXiv preprint arXiv:2606.15912*, 2026.

Alexander Lin, Jeremy Wohlwend, Howard Chen, and Tao Lei. Autoregressive knowledge distillation through imitation learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6121–6133, 2020.

Ilya Loshchilov and Frank Hutter. Online batch selection for faster training of neural networks. *arXiv preprint arXiv:1511.06343*, 2015.

Wenhan Ma, Jianyu Wei, Liang Zhao, Hailin Zhang, Bangjun Xiao, Lei Li, Qibin Yang, Bofei Gao, Yudong Wang, Rang Li, Jinhao Dong, Zhifang Sui, and Fuli Luo. MOPD: Multi-teacher on-policy distillation for capability integration in LLM post-training. *arXiv preprint arXiv:2606.30406*, 2026.

Sören Mindermann, Jan M Brauner, Muhammed T Razzak, Mrinank Sharma, Andreas Kirsch, Winnie Xu, Benedikt Höltgen, Aidan N Gomez, Adrien Morisot, Sebastian Farquhar, et al. Prioritized training on points that are learnable, worth learning, and not yet learnt. In *International Conference on Machine Learning*, pp. 15630–15649. PMLR, 2022.

Miao Rang, Zhenni Bi, Hang Zhou, Kai Han, Xuechun Wang, An Xiao, Xinghao Chen, Yunhe Wang, and Hanting Chen. Near-policy: Accelerating on-policy distillation via asynchronous generation and selective packing. *arXiv preprint arXiv:2605.05940*, 2026.

Stéphane Ross, Geoffrey 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*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020.

Hanyang Wang, Weijieying Ren, Yuxiang Zhang, Ding Cao, Zhizhao Zeng, Ke Zeng, and Tianxiang Zhao. Bipace: Bisimulation-guided policy optimization with action counterfactual estimation for llm agents. *arXiv preprint arXiv:2606.25556*, 2026a.

Hanyang Wang, Lu Wang, Chaoyun Zhang, Tianjun Mao, Si Qin, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. Text2grad: Reinforcement learning from natural language feedback. In *International Conference on Learning Representations*, volume 2026, pp. 50627–50664, 2026b.

Jiaqi Wang, Wenhao Zhang, Weijie Shi, Yaliang Li, and James Cheng. Tcod: Exploring temporal curriculum in on-policy distillation for multi-turn autonomous agents. *arXiv preprint arXiv:2604.24005*, 2026c.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. Scienceworld: Is your agent smarter than a 5th grader?, 2022. *URL https://arxiv. org/abs/2203.07540*, 2022.

Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, et al. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning. *arXiv preprint arXiv:2602.08234*, 2026.

Bangjun Xiao, Bingquan Xia, Bo Yang, Bofei Gao, Bowen Shen, Chen Zhang, Chenhong He, Chiheng Lou, Fuli Luo, Gang Wang, et al. MiMo-V2-Flash technical report. *arXiv preprint arXiv:2601.02780*, 2026.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35: 20744–20757, 2022.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38:113222–113244, 2026.

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. *arXiv preprint arXiv:2607.05804*, 2026.

Yuzhen Zhou, Jiajun Li, Yusheng Su, Gowtham Ramesh, Zilin Zhu, Xiang Long, Chenyang Zhao, Jin Pan, Xiaodong Yu, Ze Wang, et al. APRIL: Active partial rollouts in reinforcement learning to tame long-tail generation. *arXiv preprint arXiv:2509.18521*, 2025.

Zhou Ziheng, Jiaqi Li, Huacong Tang, Ying Nian Wu, and Demetri Terzopoulos. Less is more: Early stopping rollout for on-policy distillation. *arXiv preprint arXiv:2605.27028*, 2026.

Appendix Contents

A Sampler-side batch composition 12
B Fixed-budget results 14
C Full main results 14
D Batch-composition diagnostics 14
E OPD objective and analysis 17
F Time and compute analysis 21
G Ablation studies 22

A Sampler-Side Batch Composition Procedure

Algorithm 1 gives the reported procedure. After each student turn, the frozen teacher scores the same response and the workflow stores $s(u)$ from Equation 2; the learner still receives an ordinary OPD batch with unit per-turn weights.

$\textsc{Valid}(u)$ requires a complete prompt, tokens, teacher log probabilities, and a nonempty action mask. When present, the teacher-valid mask must align with and overlap the action mask; a missing mask is treated as all-valid. Turns with inference or mask-alignment failures are discarded because waiting cannot make their supervision usable.

Input: update index q; pending turns P; rollout buffer $\mathcal{E}$; batch size B; pool multiplier c; initial per-rollout cap k; maximum staleness δ; maximum pending age W.

1 Set $v_{\min} \leftarrow \max(q-\delta, 0)$ and remove every $u \in P$ with $\text{version}(u) < v_{\min}$.
2 Read F from $\mathcal{E}$ with version at least $v_{\min}$ until $|P| + |F| = cB$ (or the read returns no more turns); set $C \leftarrow P \cup F$.
3 Set $V \leftarrow \{u \in C : \textsc{Valid}(u)\}$ and permanently discard $C \setminus V$. Stamp a fresh turn's first-seen update as q.
4 Group V by rollout identifier r. Order rollout groups by oldest first-seen update, breaking ties by r.
5 Within every group, sort turns by descending $s(u)$, breaking ties by (r, t). Initialize the selected set $S \leftarrow \varnothing$ and cap $h \leftarrow k$.
6 Sweep the ordered rollout groups once. From each group append its highest ranked unselected turns until that group contributes k turns or $|S| = B$.
7 **while** $|S| < B$ and an unselected turn remains in V: set $h \leftarrow h + 1$ and sweep the groups again, adding the next-ranked turns up to cap h.
8 Set $P' \leftarrow \{u \in V \setminus S : q - \text{firstseen}(u) \le W\}$; discard older unselected turns.
9 **return** S to the unchanged OPD learner and retain P' for update $q + 1$.

Algorithm 1: **DivOPD sampler (`trajectory_first` with `turn_signal=raw_kl_sum`).** Table 3 lists the reported numerical settings.

With $B = 64$ and $k = 4$, the first sweep covers at least $\lceil B/k \rceil = 16$ rollouts whenever the valid pool contains that many groups. If the first sweep does not fill the batch, later passes raise the cap and select more turns. The batch is short only when the pool contains fewer than B valid turns. Unselected valid turns remain pending for at most $W = 8$ updates, while turns more than $\delta = 2$ model versions old are removed before composition.

The reported configuration sets `weight_mode=none` and `task_balance=false`. Within each rollout, selection is deterministic given the scores and allocated slots. We do not use importance weights to recover a uniform-turn objective. Selection changes the data-weighted objective, while the per-turn OPD loss, reduction, and optimizer stay the same.

Table 3: **Shared sampler configuration.** All methods use the same learner batch size. DivOPD-base and DivOPD share these sampler values; only their within-rollout selection differs.

Setting	Value
Learner batch size B	64 turn rows
Candidate-pool multiplier c	4 (at most 256 pending and fresh turns)
Maximum policy staleness δ	2 model versions
Maximum pending age W	8 learner updates
Initial per-rollout cap k	ALFWorld 1.7B/4B: 4/4; WebShop 3B/7B: 5/4; ScienceWorld 1.5B/3B: 4/5
Per-turn loss settings	$\beta = 1$; unit row weights
Task balancing	Disabled

Table 4: **Shared optimizer configuration.** These settings are fixed across all methods.

Setting	Value
Optimizer	AdamW
Learning rate	10^{-6}; constant schedule; no warmup
Adam moments	$\beta_1 = 0.9,\ \beta_2 = 0.999$
Weight decay	0.01
Gradient-norm clipping	1.0
Optimizer batch	$N = 64$ turn rows; seq-mean-token-mean (Equation 3)

Table 5: **Student initialization, frozen teachers, and evaluation protocol.** Initial SR is measured before the first learner update under the reported evaluation protocol; teacher SR uses the same fixed test episodes. Training responses are sampled at temperature 1.0, and evaluation uses temperature 0.4.

Environment	Student	Initial SR (%)
ALFWorld	Qwen3-1.7B	7.86
ALFWorld	Qwen3-4B	26.43
WebShop	Qwen2.5-3B-Instruct	1.56
WebShop	Qwen2.5-7B-Instruct	14.06
ScienceWorld	Qwen2.5-1.5B-Instruct	21.48
ScienceWorld	Qwen2.5-3B-Instruct	45.31

Environment	Frozen teacher	Teacher SR (%)	Episodes
ALFWorld	GiGPO-Qwen2.5-7B-Instruct-ALFWorld	87.86	140
WebShop	GiGPO-Qwen2.5-7B-Instruct-WebShop	85.94	128
ScienceWorld	Embodied-Planner-R1 (Fei et al., 2025)	93.75	256

Table 6: **ScienceWorld task-type split.** Training and evaluation use disjoint task types. Within each type, we retain the first half of the official variation IDs before applying a fixed shuffle.

Training task types (17)	Held-out evaluation task types (13)
boil; melt; change-the-state-of-matter-of; use-thermometer; measure-melting-point-known-substance; power-component; test-conductivity; find-living-thing; find-plant; grow-plant; chemistry-mix; chemistry-mix-paint-secondary-color; lifespan-shortest-lived; identify-life-stages-2; inclined-plane-determine-angle; inclined-plane-friction-named-surfaces; mendelian-genetics-known-plant	freeze; measure-melting-point-unknown-substance; power-component-renewable-vs-nonrenewable-energy; test-conductivity-of-unknown-substances; find-non-living-thing; find-animal; grow-fruit; chemistry-mix-paint-tertiary-color; lifespan-longest-lived; lifespan-longest-lived-then-shortest-lived; identify-life-stages-1; inclined-plane-friction-unnamed-surfaces; mendelian-genetics-unknown-plant

ScienceWorld subset construction. For a task type with N official variations, we retain variation IDs $0, \ldots, \lfloor N/2 \rfloor - 1$ and shuffle each split once with seed 42. This produces 2,294 training instances and 1,308 instances from held-out task types. Training cycles sequentially through the fixed shuffled training file. Every evaluation uses the same first 256 instances of the fixed shuffled held-out file; the frozen-teacher and initial-student SRs use these episodes as well. Thus, no ScienceWorld task type is shared between training and evaluation.

Evaluation processes each test set in order every five explorer steps (every 20 for the ALFWorld focus-only runs), giving 24–49 evaluations per run. The maximum interaction horizon is 30 turns on ALFWorld and ScienceWorld and 15 on WebShop. Runs are configured for 250 trainer updates on ALFWorld and ScienceWorld and 150 on WebShop; exceptions are focus-only on ScienceWorld 3B (150), WebShop 3B (130), and WebShop 7B (111), and DivOPD-base (138), DivOPD (134), and TurnOPD (132) on WebShop 7B. The matched-access cap comparison in Table 18 uses seed 42 for both arms in each setting.

A.1 Reference variant and comparison scope

DivOPD-base applies the validity gate and rollout-first coverage with uniform within-rollout sampling. DivOPD uses the same allocation rules and adds focus; DivOPD+R additionally changes rollout generation through recovery. These variants share the numerical sampler settings in Table 3, unit per-turn weights, and no task balancing.

DivOPD-focus is an earlier implementation. It has no validity gate, visits task groups in seeded random order, ranks turns by disagreement, caps a group at three turns and a task's token share at 25% on the first pass, removes the turn cap when relaxing, and enables per-rollout task-balance loss weights. Some runs are shorter and the ALFWorld evaluation frequency differs, as noted above. Its comparison with DivOPD is therefore descriptive.

Focus-only has the highest per-batch effective rollout count in five settings, while DivOPD has higher peak SR in five (Table 9). DivOPD with $k = 3$ also exceeds focus-only in peak SR in all six settings (mean $+4.9$ points), but gating, ordering, relaxation, and weighting differ. These comparisons do not isolate the cap or focus; the matched controls in Section 4.4 do.

B Fixed-Budget Efficiency

We reconstruct evaluation curves from the archived explorer logs, map each evaluation to its learner policy version through `rollout/model_version`, and linearly interpolate the unsmoothed success rates. Explorer log steps are not learner updates and are not used as the budget axis. For each setting, U is the smallest final evaluated learner version among vanilla OPD, TCOD-F2B, TurnOPD, DivOPD-base, DivOPD, and DivOPD+R. The normalized area is $\mathrm{nAUC} = U^{-1} \int_0^U \mathrm{SR}(q)\, dq$ over these interpolated curves; Figure 4 (Section 4.3) plots four of the methods over the shared budget. The aligned audit is reproduced by `plot_final_narrative.py` from the supplied logs.

C Full Per-Setting Main Results

Appendix Table 7 expands Table 1 with update counts and absolute trainer GPU-hours for every method and setting.

D Full Batch-Composition Diagnostics

Table 8 distinguishes batch selection from changes to the learning signal or rollout policy. Agent-RL methods such as Text2Grad, GiGPO, BiPACE, and SkillRL change the feedback extracted from interaction (Wang et al., 2026b; Feng et al., 2026; Wang et al., 2026a; Xia et al., 2026). Table 9 reports the complete six-setting audit behind the diagnostics summarized in Figure 2; the online cross-rollout disagreement diagnostic appears in Figure 6.

Table 7: **Full results across six teacher–student settings.** Peak SR is the best checkpoint; Final-5 is the mean ± standard deviation over the last five evaluation checkpoints. Best values are highlighted in bold; second-best are underlined.

Setting	Method	Accuracy		Efficiency to τ		
		Peak↑	Final-5↑	Updates↓	Trainer GPU h↓	GPU speedup↑
ALFWorld Qwen3-4B τ=80	Vanilla OPD	86.43	84.57 ± 1.48	184	13.51	1.00×
	TCOD-F2B	87.14	83.00 ± 1.28	182	8.95	1.51×
	TurnOPD	87.14	86.43 ± 0.51	161	8.50	1.59×
	DivOPD-focus	89.29	85.29 ± 3.10	148	7.97	1.70×
	DivOPD-base	**92.14**	88.86 ± 2.29	154	10.61	1.27×
	DivOPD	91.43	88.00 ± 2.74	127	**6.98**	**1.94**×
	DivOPD+R	**92.14**	**89.00 ± 2.51**	**126**	7.06	1.91×
ALFWorld Qwen3-1.7B τ=70	Vanilla OPD	77.86	74.00 ± 3.18	208	8.21	1.00×
	TCOD-F2B	72.86	68.43 ± 3.09	211	5.04	1.63×
	TurnOPD	78.57	75.71 ± 2.72	195	4.66	1.76×
	DivOPD-focus	80.00	71.00 ± 6.62	214	5.62	1.46×
	DivOPD-base	74.29	72.86 ± 1.43	189	6.08	1.35×
	DivOPD	79.29	71.86 ± 7.29	155	3.97	2.07×
	DivOPD+R	**82.86**	**78.29 ± 1.20**	**150**	**3.75**	**2.19**×
WebShop Qwen2.5-3B τ=60	Vanilla OPD	60.94	43.44 ± 13.02	147	10.02	1.00×
	TCOD-F2B	67.19	61.72 ± 4.42	104	6.11	1.64×
	TurnOPD	68.75	58.59 ± 8.80	114	5.44	1.84×
	DivOPD-focus	70.31	66.41 ± 3.45	99	4.17	2.40×
	DivOPD-base	78.12	73.12 ± 4.64	113	5.33	1.88×
	DivOPD	78.12	67.81 ± 3.96	106	4.52	2.22×
	DivOPD+R	**79.69**	**76.41 ± 4.04**	**87**	**3.84**	**2.61**×
WebShop Qwen2.5-7B τ=85	Vanilla OPD	87.50	82.50 ± 3.11	88	8.69	1.00×
	TCOD-F2B	89.06	83.59 ± 3.79	97	6.23	1.40×
	TurnOPD	89.84	85.94 ± 2.47	106	6.19	1.41×
	DivOPD-focus	89.84	86.88 ± 2.17	**82**	**4.36**	**1.99**×
	DivOPD-base	88.28	78.44 ± 4.37	116	7.14	1.22×
	DivOPD	**90.62**	82.66 ± 5.25	86	4.53	1.92×
	DivOPD+R	89.84	**88.28 ± 2.71**	101	5.20	1.67×
ScienceWorld Qwen2.5-3B τ=85	Vanilla OPD	85.55	80.78 ± 4.09	233	2.71	1.00×
	TCOD-F2B	87.89	83.98 ± 2.58	228	2.26	1.20×
	TurnOPD	83.20	80.86 ± 2.72	–	–	–
	DivOPD-focus	77.34	74.69 ± 2.20	–	–	–
	DivOPD-base	86.72	82.66 ± 1.50	190	2.73	0.99×
	DivOPD	88.28	**85.47 ± 2.49**	**131**	**1.66**	**1.63**×
	DivOPD+R	**90.23**	85.08 ± 2.92	214	2.52	1.07×
ScienceWorld Qwen2.5-1.5B τ=65	Vanilla OPD	66.02	63.75 ± 2.49	211	1.97	1.00×
	TCOD-F2B	71.88	66.33 ± 3.77	221	1.55	1.27×
	TurnOPD	71.48	68.75 ± 3.65	174	**1.28**	**1.54**×
	DivOPD-focus	66.80	61.88 ± 4.71	242	2.07	0.95×
	DivOPD-base	70.31	67.50 ± 2.20	195	1.98	0.99×
	DivOPD	**78.91**	76.09 ± 2.23	**149**	**1.28**	**1.54**×
	DivOPD+R	78.12	**77.42 ± 0.75**	179	1.52	1.30×

Protocol. Every run assigns two GPUs to the trainer. GPU hours accumulate trainer-step time through the first evaluation reaching τ; speedups are relative to vanilla OPD. Cap settings appear in Section 4.1.

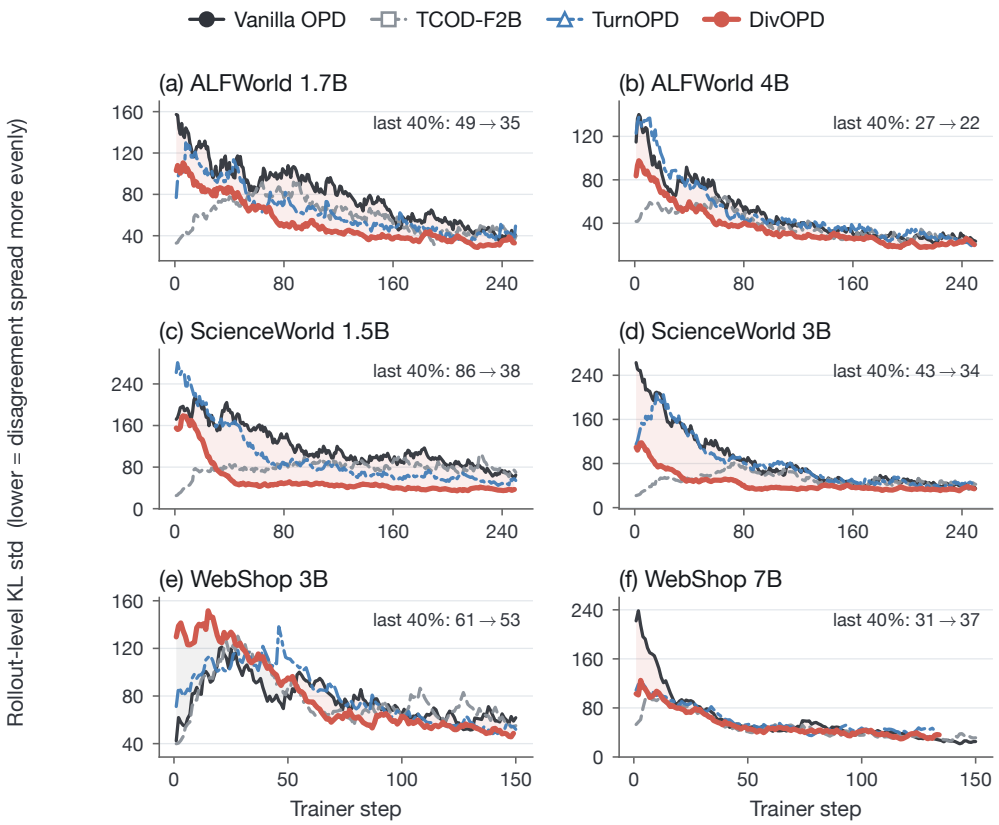

Figure 7: **Cross-rollout disagreement in all six settings** (main-text Figure 6 shows panels a, c, and e). Curves are the standard deviation of rollout-summed student-minus-teacher log probabilities (EMA, $\alpha = 0.12$); each panel gives the mean over the last 40% of training (vanilla $\rightarrow$ DivOPD). Red shading marks where DivOPD is lower and gray where it is higher. Variation falls on ALFWorld and ScienceWorld.

Table 8: **What DivOPD changes.** The core method changes batch selection; only the optional recovery extension changes rollout generation.

Design question	**Component**	**DivOPD**
How is each turn scored?	Learning signal	Unchanged (Eq. 1)
How long can a rollout continue?	Rollout horizon	Unchanged
Who acts during the rollout?	Rollout policy	Changed only by DivOPD+R (§3.4)
Which turns enter a learner batch?	Batch selection	Changed by DivOPD (§3)

Table 9: **Full batch-composition diagnostics.** Values are averaged over the final 40% of training; peak SR is repeated from Appendix Table 7. Focus-only maximizes coverage in five settings, while DivOPD has the higher peak SR in five. Bold marks the best peak SR among the five methods shown, which exclude DivOPD+R. Figure 2 averages the first three diagnostics over the two student sizes per environment.

Setting	Method	Eff. rollouts / batch ↑	Top-1 tok. share ↓	Turns per rollout	KL / token ↓	Step time (s) ↓	Peak SR (%) ↑
ALFWorld 4B	Vanilla OPD	6.7	0.281	8.1	0.121	94.9	86.43
	TCOD-F2B	6.5	0.291	8.6	0.114	99.6	87.14
	DivOPD-focus	18.6	0.110	2.4	0.174	85.2	89.29
	DivOPD-base	12.5	0.136	3.9	0.111	97.8	**92.14**
	DivOPD	12.5	0.145	3.9	0.176	82.1	91.43
ALFWorld 1.7B	Vanilla OPD	5.2	0.364	10.2	0.222	51.2	77.86
	TCOD-F2B	5.4	0.346	10.0	0.200	49.0	72.86
	DivOPD-focus	16.1	0.120	3.0	0.280	40.3	**80.00**
	DivOPD-base	12.4	0.132	3.9	0.201	44.4	74.29
	DivOPD	12.4	0.140	3.9	0.289	39.5	79.29
WebShop 3B	Vanilla OPD	4.7	0.334	7.8	0.179	106.3	60.94
	TCOD-F2B	5.0	0.328	7.4	0.167	101.9	67.19
	DivOPD-focus	25.7	0.069	2.0	0.181	64.6	70.31
	DivOPD-base	12.4	0.114	4.7	0.151	76.7	**78.12**
	DivOPD	12.3	0.114	4.7	0.156	70.9	**78.12**
WebShop 7B	Vanilla OPD	8.3	0.237	6.1	0.035	126.7	87.50
	TCOD-F2B	7.5	0.271	6.2	0.029	120.1	89.06
	DivOPD-focus	29.2	0.067	1.5	0.052	90.2	89.84
	DivOPD-base	14.8	0.101	3.9	0.027	104.5	88.28
	DivOPD	15.0	0.096	3.9	0.036	96.3	**90.62**
ScienceWorld 3B	Vanilla OPD	6.2	0.272	9.5	0.241	18.9	85.55
	TCOD-F2B	6.2	0.253	9.3	0.242	19.3	87.89
	DivOPD-focus	14.6	0.115	3.9	0.487	22.7	77.34
	DivOPD-base	11.3	0.146	4.7	0.257	21.1	86.72
	DivOPD	11.8	0.137	4.7	0.336	19.1	**88.28**
ScienceWorld 1.5B	Vanilla OPD	3.9	0.400	12.9	0.349	15.2	66.02
	TCOD-F2B	4.1	0.383	12.5	0.341	14.2	71.88
	DivOPD-focus	12.8	0.121	4.3	0.513	14.1	66.80
	DivOPD-base	12.1	0.130	4.4	0.341	16.3	70.31
	DivOPD	13.5	0.115	4.0	0.483	13.8	**78.91**

E EXECUTED OPD OBJECTIVE

Vanilla OPD, DivOPD-base, DivOPD, and the matched-cap controls use unit row weights with the same seq-mean-token-mean reduction. Its scalar value is

$$\mathcal{L}_{\mathrm{OPD}}^{\mathrm{value}} = \frac{1}{N}\sum_{t=1}^{N} \ell_t, \qquad \ell_t = \begin{cases} -\dfrac{1}{|V_t|}\displaystyle\sum_{i\in V_t} A_{t,i}, & |V_t| > 0, \\ 0, & |V_t| = 0. \end{cases} \tag{3}$$

Stored scores and recomputed learner probabilities. The workflow stores student log probabilities at the generating policy $\theta_{v(u)}$ and frozen-teacher log probabilities on the same response. Equation 2 uses these cached values for selection. Before update q, the learner recomputes student log probabilities at its current snapshot $\bar{\theta}_q$, which is θ_{old} in Equation 1; teacher scores remain fixed. The PPO surrogate uses

$$r_{t,i}(\theta) = \exp\Big[\log \pi_\theta(y_{t,i} \mid h_{t,<i}) - \mathrm{sg}\Big[\log \pi_{\bar{\theta}_q}(y_{t,i} \mid h_{t,<i})\Big]\Big].$$

At gradient evaluation, $\theta = \bar{\theta}_q$, so $r_{t,i} = 1$ and clipping is inactive, but $\nabla_\theta r_{t,i}$ is not zero. Each batch receives one optimizer update. There is no additional behavior-to-learner importance correction: bounded staleness limits the sampling mismatch but does not remove it. Vanilla OPD and the controlled composer variants share this update rule. The outer average includes dead rows, and the gradient is

$$\nabla_\theta \mathcal{L}_{\mathrm{OPD}} = \frac{1}{N}\sum_{t:\,|V_t|>0} \frac{1}{|V_t|}\sum_{i\in V_t} -A_{t,i}\,\nabla_\theta \log \pi_\theta(y_{t,i} \mid h_{t,<i}). \tag{4}$$

Thus valid tokens are averaged within each turn row, rows are equally weighted, and a dead row contributes zero while remaining in the N-row denominator. The formula describes the executed update on selected, potentially lagged samples; it is not an unbiased current-policy reverse-KL gradient.

E.1 GRADIENT EFFECT OF BATCH COMPOSITION

Because Equation 4 is a plain average of per-row terms, its selection-induced change can be written exactly for a fixed pool. These equations describe reweighting, not a guarantee of better policy performance. Let the candidate pool hold N_c turns and write the contribution of turn t as $g_t = \frac{1}{|V_t|}\sum_{i\in V_t} -A_{t,i}\nabla_\theta \log \pi_\theta(y_{t,i} \mid h_{t,<i})$, with $g_t = 0$ for a dead row. The pool gradient is $G = \frac{1}{N_c}\sum_t g_t$. A composer selects B rows through indicators $m_t \in \{0,1\}$ with $\sum_t m_t = B$; let $\pi_t = \mathbb{E}[m_t]$ be the inclusion probability, so $\sum_t \pi_t = B$. The optimizer sees $\hat{G} = \frac{1}{B}\sum_t m_t g_t$.

Lemma 1 (Selection-induced shift identity)**.** *For any full composer batch, $\mathbb{E}[\hat{G}] - G = \frac{N_c}{B}\operatorname{Cov}_t(\pi_t, g_t)$, where this is the scalar–vector covariance over a uniformly drawn pool row.*

Proof. $\mathbb{E}[\hat{G}] = \frac{1}{B}\sum_t \pi_t g_t = \frac{N_c}{B}\mathbb{E}_t[\pi_t g_t]$. Since $\mathbb{E}_t[\pi_t] = B/N_c$, also $G = \mathbb{E}_t[g_t] = \frac{N_c}{B}\mathbb{E}_t[\pi_t]\,\mathbb{E}_t[g_t]$. Subtracting gives $\frac{N_c}{B}\big(\mathbb{E}_t[\pi_t g_t] - \mathbb{E}_t[\pi_t]\mathbb{E}_t[g_t]\big)$. □

Score–loss relation at a common policy version. At the same student policy version v, the common valid mask gives

$$s_v(u) = -\frac{1}{\beta}\sum_{i\in V_u} A_{u,i}^{(v)} = \frac{|V_u|}{\beta}\,\ell_u^{\mathrm{value},(v)}. \tag{5}$$

With a complete response mask, its conditional expectation is the sequence-level reverse KL, $D_{\mathrm{KL}}(P_{\theta_v}(\cdot \mid h)\|P_\phi(\cdot \mid h))$. Thus raw s_v ranks token count times row-mean OPD loss, while $s_v/|V_u|$ ranks the row loss. Neither is the gradient norm because score-gradient vectors can vary and cancel. Cached scores use rollout version v; learner-side recomputation can be up to δ versions later. The equality therefore does not identify the cached score with the row loss evaluated at the later learner snapshot.

The identity shows how each stage changes the expected gradient. The gate removes rows with $g_t = 0$ in the audited buffers. If the remaining rows are sampled uniformly, this increases the expected gradient magnitude without changing its direction, as shown below. Coverage limits how much rollout length affects π_t, moving from turn-uniform toward rollout-uniform averaging. Focus then favors turns with larger $s(u)$ within each selected rollout. Section E.3 measures the resulting gradient changes.

The following sampling calculation isolates dead-slot dilution under uniform sampling. Its variance result uses simplifying assumptions: rollout-correlated gradients and deterministic selection need not satisfy its independence assumptions.

Proposition 1 (Dead-slot dilution under uniform sampling)**.** *If a pool contains D dead rows and the gate samples uniformly from its $N_c - D$ live rows, then $\mathbb{E}[\hat{G}_{\mathrm{gate}}] = \frac{N_c}{N_c - D}G$: the gate preserves direction while undoing dead-slot shrinkage. Further, suppose live-row gradients are i.i.d. with mean μ and covariance Σ, and independent slot validity is $Z \sim \mathrm{Bernoulli}(1-d)$, where $d = D/N_c$ is the dead-row fraction. For the same valid-row target,*

$$\operatorname{Cov}(\hat{\mu}_{\mathrm{gate}}) = \frac{\Sigma}{B}, \qquad \operatorname{Cov}\left(\frac{\hat{G}_{\mathrm{ungated}}}{1-d}\right) = \frac{\Sigma + d\mu\mu^\top}{B(1-d)}. \tag{6}$$

Hence gating is a positive-semidefinite variance reduction; at $\mu = 0$, its variance is exactly $(1-d)$ times the validity-corrected ungated variance.

Proof. Dead rows have $g_t = 0$, so uniform live-row sampling gives $\mathbb{E}[\hat{G}_{\mathrm{gate}}] = (N_c - D)^{-1}\sum_{\mathrm{live}} g_t = N_c G/(N_c - D)$. For the stochastic statement, let X denote a live-row gradient. Then $\mathbb{E}[ZX] = (1-d)\mu$ and $\operatorname{Cov}(ZX) = (1-d)\Sigma + d(1-d)\mu\mu^\top$. Independence across the B slots yields Equation 6; subtracting Σ/B leaves $d(\Sigma + \mu\mu^\top)/[B(1-d)] \succeq 0$. □

At $\mu = 0$, validity-corrected ungated reading is variance-equivalent to a live-only batch of effective size $B_{\mathrm{eff}} = (1-d)B$; for $B = 64$ and $d = 0.05$–0.28, this is 46–61 rows, whereas the gate restores all 64. The scale factor $1/(1-d)$ is largely suppressed before optimization in our runs: gradient norms are clipped at 1.0 and logged pre-clip norms are 2–15. Under the simplifying assumption of clipped SGD, the first-order comparison then reduces to the direction of $\hat{G}$; AdamW's history-dependent preconditioning prevents this from being an exact equivalence. We therefore interpret B_{eff} as a sampling-efficiency statement rather than an exact claim about optimizer step length. Empirically, gate-only preserves mean $s(u)$ per valid ungated turn in all four replays; on the ALFWorld 1.7B DivOPD buffer, $14.59/(1 - 0.1471) = 17.11$. Focus is evaluated separately through the gradient-geometry audit and downstream results.

Corollary 1 (Coverage as first-sweep capped rollout averaging)**.** *Condition on the oldest-first selected rollout set $\mathcal{R}_S$ and allocations $\{n_r\}$, where $\sum_r n_r = B$. Uniform within-rollout sampling gives*

$$\mathbb{E}[\hat{G}_{\mathrm{base}} \mid \mathcal{R}_S, \{n_r\}] = \sum_{r\in\mathcal{R}_S}\frac{n_r}{B}\,\bar{g}_r. \tag{7}$$

When the batch fills in the first sweep, $n_r = \min(k, T_r)$ for each fully allocated group; the final group may receive only the remaining slots. If all selected $T_r \geq k$ and B is divisible by k, the result is the rollout-uniform mean over selected groups. If every rollout is selected in full, it is the original turn-uniform mean.

The first sweep therefore moves between turn-uniform and rollout-uniform averaging by capping length weights. Holding the selected set fixed separates that reweighting from oldest-first rollout selection and gives an exact description of the coverage stage.

Corollary 2 (Focus reweighting identity)**.** *For the same $\mathcal{R}_S$ and $\{n_r\}$, let K_r be the deterministic top-n_r set by $s(u)$ and D_r its complement. Then*

$$\hat{G}_{\text{focus}} - \mathbb{E}[\hat{G}_{\text{base}} \mid \mathcal{R}_S, \{n_r\}] = \frac{1}{B} \sum_{r:n_r<T_r} \frac{n_r(T_r - n_r)}{T_r} \left(\bar{g}_r^K - \bar{g}_r^D\right). \tag{8}$$

This follows from $\bar{g}_r = (n_r/T_r)\bar{g}_r^K + ((T_r - n_r)/T_r)\bar{g}_r^D$.

Thus the change caused by focus depends on the difference between the mean gradients of kept and dropped turns. Table 10 also compares their gradient norms: kept turns account for a larger share of total gradient norm than of turn count, and tend to be earlier, higher-disagreement decisions.

Table 10: **Focus retention audit** ($k = 5$, the initial cap used in Table 1 for these settings). Dropped gradient norm is the share of summed per-turn norms, $\sum_t \|g_t\|$, estimated by the sketch audit. The last four columns are means for kept and dropped turns.

	Dropped share		**Turn depth**		**Score/token**	
Setting	Turns	Grad. norm	Kept	Dropped	Kept	Dropped
ScienceWorld 3B	55%	40%	4.7	9.5	0.30	0.16
WebShop 3B	18%	12%	2.3	4.5	0.16	0.08

E.2 Rollout utilization under a staleness budget

Table 11: **Cumulative rollout utilization.** Never selected is the share of valid rollouts contributing no trained turn; Gini measures concentration of selected-turn counts. ALF/SciW/WS denote ALFWorld/ScienceWorld/WebShop. The separate-buffer result is grouped below the shared-buffer comparisons and is not a matched control.

	Never selected (%)			**Gini**		
Composer	ALF	SciW	WS	ALF	SciW	WS
Shared buffers: first 100 updates						
Vanilla OPD (arrival-order)	53.9	49.2	58.7	0.66	0.68	0.64
Top-B (no rollout limit)	–	–	–	0.44	0.45	0.27
DivOPD-focus	15.0	10.2	3.4	0.33	0.32	0.20
DivOPD-base / DivOPD	14.2	11.0	7.1	0.30	0.32	0.12
Separate buffers: full 250 updates						
DivOPD, $k = \infty$	56.6	67.0	39.9	0.70	0.79	0.51

Top-B ranks all valid turns by $s(u)$ without a per-rollout limit. DivOPD-base and DivOPD coincide because focus does not change rollout allocation. The separate 250-update buffers were generated by $k = \infty$ runs; this replay is a qualitative access diagnostic, not the uniform-within-rollout, matched-access training control in Table 18.

Consider a group of R rollouts that remains eligible for L updates. If each update touches at most M distinct rollouts, at most $\min(R, LM)$ members of that group can be selected. This is a group-level upper bound, not a steady-state utilization estimate: other arrival cohorts compete for the same slots, and a rollout can be selected repeatedly. An arrival-order reader consuming consecutive rollout blocks touches roughly $B/\bar{T}$ rollouts per update, where $\bar{T}$ is mean rollout length. The first-pass cap instead covers at least $\lceil B/k \rceil$ groups when enough valid groups are available. It therefore increases opportunities for rollout participation, while actual lifetime coverage must be measured across updates. Table 11 provides that measurement. Every full composer batch still contains $B = 64$ rows; the cap redistributes them across rollouts rather than increasing the row budget.

Effect of a wider staleness window. Relative to the main vanilla references with $\text{max_staleness} = 2$, two auxiliary runs with a limit of 4 leave row-level consumption unchanged

Table 12: **Vanilla OPD with a wider staleness window.** Mean age of consumed rows in policy versions (arrival-order replay, dead rows included), peak and final-checkpoint SR (%). The $\mathrm{max_staleness} = 2$ rows reuse the main vanilla runs in Table 1; the wider-window rows are separate auxiliary runs rather than paired reruns.

Setting	max_staleness	Mean age	Peak	Final ckpt.
ALFWorld 1.7B	2	1.69	77.86	73.57
	4	3.19	71.43	71.43
ScienceWorld 1.5B	2	1.68	66.02	66.02
	4	2.96	65.62	60.16

and consume rows 1.3–1.5 versions older (Table 12). Peak SR is lower by 6.4 points on ALFWorld and 0.4 points on ScienceWorld, while the mean current-versus-rollout probability gap is 10–15% larger.

Sensitivity to bounded policy lag. The usual reverse-KL gradient identity is exact when samples are drawn from the current learner policy. In our asynchronous implementation, trajectories may instead be generated by a behavior policy up to two versions behind the learner, so the executed update is a bounded-lag approximation. Across the six main settings, rollout and learner action probabilities remain highly correlated (0.956–0.989), with mean absolute differences of only 0.011–0.015. As an additional offline stress test, recomputing DivOPD scores at later checkpoints gives cached–current Spearman correlations of 0.984 on ScienceWorld and 0.997 on WebShop; the corresponding within-trajectory top-k selection overlaps are 93.5% ($k = 4$) and 99.6% ($k = 5$). In these tests, recomputing the scores within the two-version lag window rarely changes which turns are selected. This supports cached ranking in the tested setting, not an unbiased-gradient claim.

E.3 Gradient diagnostics

We replay every composer on the same candidate pools over the last eight updates of each buffer. To compare per-turn gradients without storing them in full, we use a Kronecker projection with unbiased inner products. Table 13 reports $\langle G, \hat{G}\rangle/\|G\|^2$ and $\cos(G, \hat{G})$, where G is the pool gradient and $\hat{G}$ is the selected batch gradient. Focus raises the first metric over uniform within-rollout sampling in all 24 audited updates; its mean cosine is also higher in all three environments.

The proxy measures alignment with the specified candidate pool, not an optimal policy-improvement direction. It is a pre-optimizer diagnostic; training outcomes are tested separately in Table 1. Mean pairwise gradient cosine is 0.003–0.02 for every composer; reduced gradient correlation is therefore not the main change observed here.

Selecting turns by alignment in the same sketch space gives an optimistic reference, with a proxy 1.3–2.1× that of the best $s(u)$ rule in each environment. Using a rollout-mean pool gradient leaves the rankings unchanged. All these comparisons use raw gradients, before optimizer preconditioning.

Table 13: **Selected gradients versus the candidate-pool gradient.** Means over 8 audited updates per environment. Projection is $\langle G, \hat{G}\rangle/\|G\|^2$; cosine similarity is $\cos(G, \hat{G})$. ALF/SciW/WS denote ALFWorld/ScienceWorld/WebShop. †Arrival order here draws from the finite-k composer's pool and is not vanilla OPD. Sketch oracle selects and evaluates the B turns with the largest sketched $\langle G, g_t\rangle$, so it is an optimistic reference rather than a full-gradient oracle.

	Gradient projection			Cosine similarity		
Composer	ALF	SciW	WS	ALF	SciW	WS
Arrival order†	1.74	0.75	1.32	0.97	0.49	0.61
Gate only	1.11	1.02	1.29	0.93	0.53	0.62
DivOPD-base	1.36	0.97	1.31	0.96	0.54	0.62
DivOPD	2.04	2.01	1.47	0.98	0.75	0.64
DivOPD-focus	3.12	1.66	1.50	0.99	0.67	0.65
Top-B ($k = \infty$)	2.97	2.30	1.61	0.99	0.79	0.68
Sketch oracle	4.19	3.70	3.30	1.00	0.91	0.84

E.4 WHICH TURNS EACH STAGE SELECTS

Table 14 follows the ALFWorld 1.7B DivOPD ($k = 4$) buffer through the selection chain. "Vanilla" reconstructs the arrival-order reader (one batch per update in queue order with the staleness filter), including invalid slots; "Gate" keeps that order but takes the first B valid rows; "Cover" and "Focus" add the remaining stages. Queue row ID provides a consistent arrival-order proxy for this controlled replay.

Table 14: **Where selected turns come from in the ALFWorld 1.7B DivOPD replay.** Share of selected turns (%), grouped by turn depth and rollout outcome. Each method column sums to approximately 100% after rounding; stages are added one at a time from left to right.

Depth	Outcome	Vanilla	+Gate	+Coverage	+Focus
< 2	Failure	3.2	3.8	6.1	10.6
	Success	17.6	20.7	14.3	15.8
$[2, 4)$	Failure	3.1	3.6	6.1	8.5
	Success	17.4	20.4	14.1	14.5
$[4, 6)$	Failure	2.9	3.4	5.9	7.4
	Success	13.1	15.3	9.7	9.7
$[6, 10)$	Failure	5.6	6.6	11.9	11.9
	Success	11.4	13.4	8.5	7.2
≥ 10	Failure	23.1	9.9	20.7	12.4
	Success	2.6	3.0	2.7	2.1

The gate removes unscored turns late in failed rollouts. Coverage retains other valid turns from those rollouts for later updates, while focus favors earlier, higher-disagreement turns within them. Coverage and focus select the same fraction of turns from successful rollouts (49.3%), so their difference is not explained by selecting more successes. The arrival-order reader, meanwhile, consumes 62.1% successful turns against 22.8% in the pool. This is consistent with shorter successful rollouts finishing and entering the queue earlier. On vanilla OPD's own ALFWorld 1.7B buffer, the same reader selects 48.1% successful turns, and its 19.5% dead-slot fraction matches the online trainer log.

F WHERE THE TIME GOES

We measure trained tokens, learner GPU time, and elapsed trainer-loop time separately. The first two quantify learner-side efficiency; the third captures elapsed time including experience reads, not total system GPU consumption. Teacher scoring is performed for generated turns whether or not they are selected. Recovery adds teacher generation, which is excluded from the reported +R learner-GPU speedup. These measurements do not provide a full accounting of explorer and teacher GPU time or queue memory.

Table 15: **Resource efficiency to τ.** Vanilla-to-DivOPD ratios; larger is better. Trainer GPU time uses the two GPUs assigned to the learner. The wall estimate sums trainer-step and experience-read time. Its speedup differs from timestamp-based end-to-end measurement by at most 1.2% on the three settings whose timestamps were retained.

Setting	Trained tokens	Trainer GPU	Wall time (est.)
ALFWorld 4B	1.92×	1.94×	1.36×
ALFWorld 1.7B	2.05×	2.07×	1.28×
WebShop 3B	2.22×	2.22×	0.99×
WebShop 7B	1.77×	1.92×	0.83×
ScienceWorld 3B	1.62×	1.63×	1.22×
ScienceWorld 1.5B	1.54×	1.54×	1.27×
Geometric mean	**1.84**×	**1.87**×	**1.14**×

All six settings improve in trained-token and trainer-GPU efficiency.

Fewer tokens make updates cheaper. Trainer time closely tracks batch tokens on ALFWorld and WebShop (Pearson $r = 0.96$), while model FLOP utilization is nearly unchanged. The gate removes long rows with truncated prompts, and the first-pass cap limits repeated turns from long rollouts, reducing time across trainer stages (Figure 8).

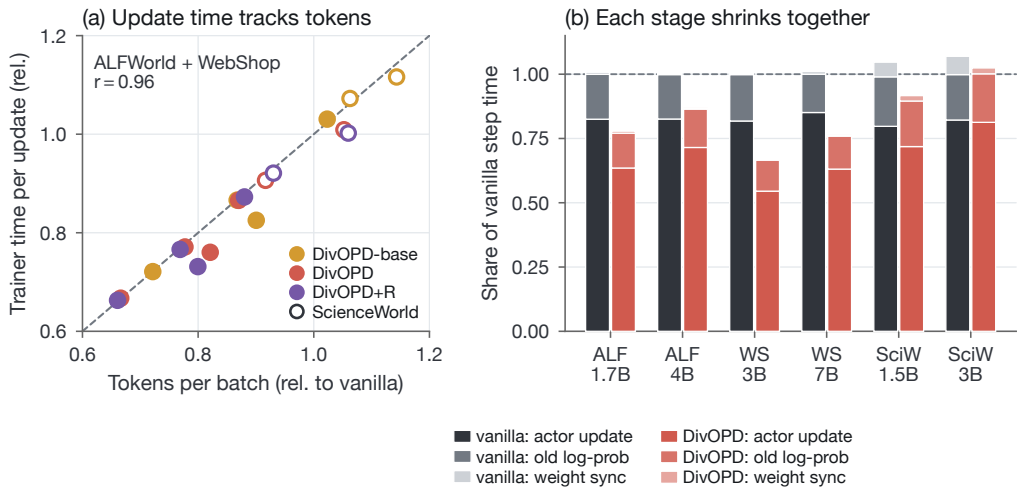

Figure 8: **Cheaper updates come from fewer tokens.** (a) Relative trainer time versus relative tokens per batch; hollow ScienceWorld points use the trainer clock only here. (b) The stage-wise time breakdown.

ScienceWorld gains come from fewer updates. Because its turns are valid and focus favors longer responses, its trainer-GPU improvement arises primarily from reaching the target in fewer updates rather than from shorter individual steps.

Teacher-side cost. The explorer scores every rollout turn, but many scored tokens never enter a learner batch (Table 16). The explorer produces turns faster than the learner consumes them, and the staleness rule limits how long excess turns can wait in the buffer.

Table 16: **Teacher tokens scored per token trained.** Effective tokens; "not selected" is the share never entering an optimizer batch. This audit uses the $k = 3$ runs from Table 19, not the per-setting caps of Table 1. *Two parallel 150-update trainer histories sharing one buffer, merged.

	DivOPD (k=3)		**DivOPD-focus**	
Setting	scored/trained	not selected	scored/trained	not selected
ALFWorld 1.7B	2.77×	63.9%	2.89×	65.4%
ALFWorld 4B	2.73×	63.4%	2.93×	65.9%
ScienceWorld 1.5B	3.66×	72.6%	3.79×	73.6%
ScienceWorld 3B	3.41×	70.6%	3.60×*	72.2%
WebShop 3B	1.89×	47.1%	1.86×	46.3%
WebShop 7B	1.95×	48.7%	1.97×	49.3%
Token-weighted	2.44×	59.1%	2.60×	61.6%

Selection overhead. The composer sorts at most $cB = 256$ rows and adds no GPU work. As learner updates become faster, rollout generation can limit further wall-time gains.

G Additional Ablations and Teacher Recovery

This appendix compares the method variants, tests the initial cap, and details how teacher recovery extends DivOPD to no-progress rollouts.

Recovery protocol. Following the use of teacher intervention in online imitation learning (Ross et al., 2011), DivOPD+R invokes the teacher after p consecutive no-progress turns: an invalid action, or a repeated action with an unchanged set of allowed actions. The teacher acts for at most four to six turns per rollout. The patience/cap/warmup triplets (warmup in learner updates) are ALFWorld 1.7B: $4/6/0$, 4B: $6/6/0$; WebShop 3B: $4/5/45$, 7B: $4/4/45$; and ScienceWorld 1.5B: $2/6/120$, 3B: $2/6/50$. Teacher actions are excluded from the distillation loss, and the student resumes interaction from the resulting state. Recovery is disabled at evaluation.

Table 18 isolates the first-pass cap under matched candidate access. Table 19 complements that comparison by varying the finite cap in DivOPD, with focus enabled.

Table 17: **Matched candidate access.** Both arms draw from the same 256-turn stream; vanilla otherwise reads in arrival order. Best-5 averages the five best checkpoints; Final is SR at the common learner-version cutoff. These separate runs are not numerically comparable to Table 1.

		Vanilla reader			Rollout-first selection		
Environment	**Model**	**Peak↑**	**Best-5↑**	**Final↑**	**Peak↑**	**Best-5↑**	**Final↑**
ALFWorld	1.7B	78.57	73.86	**78.57**	**80.00**	**77.71**	**78.57**
WebShop	3B	75.00	68.44	**71.09**	**78.12**	**74.06**	68.75
ScienceWorld	1.5B	73.44	72.66	73.05	**77.73**	**74.38**	**74.61**

Table 18: **Matched-access cap comparison.** Both arms use the same candidate-pool size, gate, ordering, pending rule, and within-rollout sampling; only the initial cap differs. SR is in percent.

		Finite k (DivOPD-base)		$k = \infty$ (no cap)	
Environment	**Model**	**Peak SR↑**	**Final-5 SR↑**	**Peak SR↑**	**Final-5 SR↑**
ALFWorld	1.7B	74.29	72.86	73.57	65.14
ALFWorld	4B	92.14	88.86	90.00	87.71
WebShop	3B	78.12	73.12	75.00	65.47
WebShop	7B	88.28	78.44	87.62	81.09
ScienceWorld	1.5B	70.31	67.50	67.97	62.89
ScienceWorld	3B	86.72	82.66	83.33	80.67

Table 19: **Sensitivity to the initial per-rollout cap k.** Peak SR (%); TCOD-F2B is a reference. Best DivOPD values are highlighted, and second-best values are underlined.

Method	**ALFWorld**		**WebShop**		**ScienceWorld**	
	1.7B	**4B**	**3B**	**7B**	**1.5B**	**3B**
TCOD-F2B	72.86	87.14	67.19	89.06	71.88	87.89
DivOPD ($k = 3$)	**87.14**	<u>90.00</u>	71.09	**91.41**	<u>77.73</u>	85.55
DivOPD ($k = 4$)	79.29	**91.43**	<u>73.44</u>	<u>90.62</u>	**78.91**	<u>86.33</u>
DivOPD ($k = 5$)	79.29	**91.43**	**78.12**	88.28	75.78	**88.28**
DivOPD ($k = 6$)	<u>80.71</u>	88.57	<u>73.44</u>	86.72	75.78	85.55

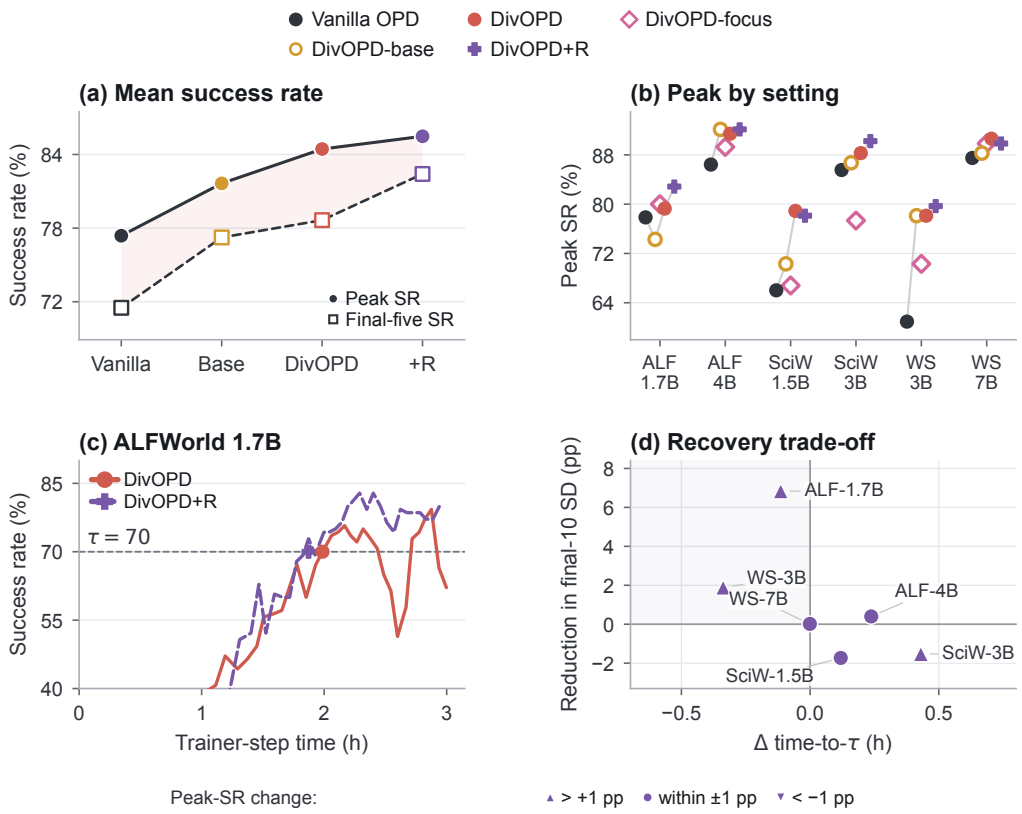

Figure 9: **Variant chain and optional recovery.** (a–b) Base combines gating, expanded access, and coverage; hollow diamonds show the separate focus-only reference. (c) Recovery on ALFWorld 1.7B. (d) Per-setting recovery effects, with marker shape encoding peak-SR change. Only the controls in Figure 5 isolate individual selection components.