Skill-CDPO: Evolving Agent Tool-Use via Critical Step Preference Optimization

Anonymous ACL submission

Abstract

Compact open-source language models lag behind their larger counterparts in agentic tool-use reliability, yet standard remedies face fundamental obstacles: supervised fine-tuning suffers from exposure bias, while reinforcement learning is hampered by sparse credit assignment over long tool-interaction trajectories. We introduce **Skill-CDPO**, a progressive framework that first acquires tool-use skills at inference time through static tool analysis and dynamic strategy refinement, then distills the resulting error-correction signals into parameter updates via **Critical Step DPO (CDPO)**. CDPO identifies the specific trajectory steps where model capability is the bottleneck—through rollout divergence between a local policy and an expert model—and constructs distributional preference pairs from all cross-group rollouts at those steps, weighted by both step-level criticality and pair-level score gaps. This provides dense, fine-grained supervision without requiring a process reward model. We evaluate Skill-CDPO on three medical agent benchmarks—PubMed Search (a new PubMed-based deep research benchmark we contribute), CureBench, and MedBrowseComp—using an 8B-parameter deep research model. Skill-CDPO substantially outperforms SFT and trajectory-level DPO baselines and achieves competitive or superior performance compared to GPT-5.2 on retrieval-intensive tasks. Our code and data are available at https://github.com/Adam135792468/CDPO.

1 Introduction

Large Language Models (LLMs) are undergoing a paradigm shift from passive chatbots to autonomous agents capable of interacting with external environments to solve complex problems (Yao et al., 2023; Schick et al., 2023; Qin et al., 2024). As highlighted by recent advancements (Anthropic, 2024), the core competency of these agents lies in their *Skills*—the ability to robustly select, configure, and execute external tools. While large-scale models (e.g., >70B parameters) have demonstrated remarkable proficiency, their compact, deployable counterparts continue to struggle with reliability, particularly in specialized domains such as medical research where precision is paramount. Closing this gap through standard training approaches faces fundamental obstacles: Supervised Fine-Tuning (SFT) suffers from the *exposure bias* problem (Ranzato et al., 2016; Ouyang et al., 2022)—it teaches the model what to do without explicitly teaching what not to do, leaving agents vulnerable to error propagation in multi-step reasoning. Reinforcement Learning (RL) methods (Schulman et al., 2017; Shao et al., 2024) are hampered by the *credit assignment problem* (Minsky, 1961): in a long chain of tool interactions, a final failure reward provides sparse supervision, making it difficult to identify which specific step caused the error. Neither approach can effectively enable compact models to match their larger counterparts.

In this work, we present two complementary findings that together address this challenge. First, we observe that compact models can **substantially improve tool-use reliability through inference-time skill synthesis**. By analyzing tool implementations to build accurate mental models of tool capabilities (*static optimization*) and by learning refined usage strategies from runtime execution logs (*dynamic optimization*), an 8B-parameter model achieves significant performance gains without any parameter updates. This training-free skill acquisition alone closes a notable portion of the gap, yet deeper deficiencies in the model's decision-making persist and cannot be resolved without parameter updates.

Furthermore, we discover that a lightweight preference optimization approach can **further unlock the potential of newly acquired skills, enabling the compact model to surpass even larger**

closed-source systems. We propose **Critical Step DPO (CDPO)**, which identifies the specific trajectory steps where model capability remains the bottleneck—through rollout divergence between the local policy and an expert model—and constructs distributional preference pairs from all $n \times m$ cross-group rollouts at those steps, weighted by both step-level criticality and pair-level score gaps. This provides dense, fine-grained supervision without requiring a process reward model, resolving the credit assignment problem that plagues trajectory-level methods.

Together, these two insights form **Skill-CDPO**, a progressive framework that first acquires tool-use skills at inference time, then distills the resulting error-correction signals into parameter updates via CDPO. We evaluate Skill-CDPO on three medical agent benchmarks—**PubMed Search** (a new PubMed-based deep research benchmark we contribute), **CureBench**, and **MedBrowseComp**—using an 8B-parameter deep research model. Our results reveal a clear progression: skill acquisition provides a strong foundation, but standard training methods (SFT, trajectory-level DPO) yield limited additional gains on top of it; CDPO, by contrast, fully unleashes the potential of skill-augmented agents, substantially outperforming all training baselines and achieving competitive or superior performance compared to GPT-5.2 on retrieval-intensive tasks.

Our contributions are: (1) **Skill-CDPO**, a progressive framework that bridges inference-time skill acquisition with critical step preference optimization for tool-use agents; (2) **Critical Step DPO (CDPO)**, which resolves the credit assignment problem through distributional preference pairs at rollout-divergence-verified bottleneck steps, offering an efficient alternative to RL; and (3) **PubMed Search**, a new PubMed-based deep research benchmark, on which our 8B model achieves state-of-the-art open-source performance across all three medical agent tasks.

2 Related Work

LLM Agents and Tool-Use Training. ReAct (Yao et al., 2023) and Toolformer (Schick et al., 2023) establish foundational paradigms for interleaving reasoning with tool invocations. On the training side, ETO (Song et al., 2024) constructs trajectory-level preference pairs from exploration rollouts, while Agent Q (Putta et al., 2024) integrates tree search with offline preference learning. In biomedical applications, retrieval-augmented agents must chain multiple search, reading, and synthesis steps (Singhal et al., 2023), where a single poorly formulated query can derail downstream reasoning. Our work targets this setting with a two-stage pipeline tailored to the compounding-error dynamics of retrieval-intensive agent tasks.

Preference Optimization at Different Granularities. DPO (Rafailov et al., 2023) casts preference alignment as a binary classification problem, spawning variants that modify the loss geometry (Azar et al., 2024), relax paired data requirements (Ethayarajh et al., 2024), or drop the reference model (Meng et al., 2024)—all operating at the response level with uniform token weighting. Token-level DPO (Zeng et al., 2024) reweights tokens by marginal contribution, and Step-DPO (Lai et al., 2024) segments proofs into discrete steps. For agent trajectories, CSO (Li et al., 2026) trains a process reward model to score candidate steps, while IPR (Xiong et al., 2025) estimates per-step value through Monte Carlo rollouts at every position—yielding unbiased signals but scaling poorly with trajectory length. CDPO departs from both in three ways: it requires no process reward model, it leverages the full nm rollout cross-product rather than isolated pairs, and it weights steps by a normalized criticality score with provable variance reduction.

3 Method

In this section, we present **Skill-CDPO**, a progressive framework for evolving agent tool-use skills. As illustrated in Figure 1, our approach consists of two stages: (1) a *training-free* skill acquisition phase (§3.1), where the agent refines its tool-use behavior through static and dynamic optimization without parameter updates; and (2) a *training-based* preference optimization phase (§3.2), where critical step preference pairs are constructed from error correction logs and used for direct preference optimization.

3.1 Training-Free Skill Acquisition

Before any parameter optimization, the agent first acquires tool-use skills through two complementary mechanisms that operate at inference time: *static optimization*, which deepens the agent's understanding of tool specifications, and *dynamic optimization*, which refines tool-use strategies based

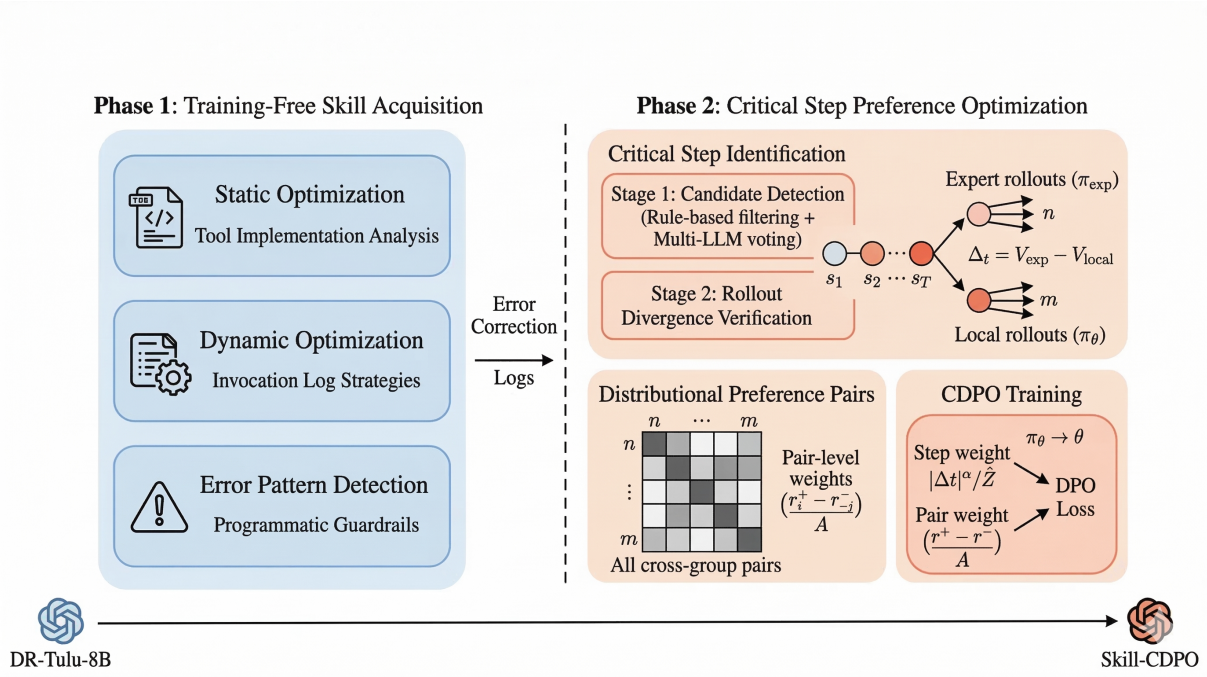

Figure 1: Overview of the Skill-CDPO framework. **Phase 1** (left): Training-free skill acquisition refines tool-use behavior through static tool analysis, dynamic strategy learning from invocation logs, and automated error pattern detection. **Phase 2** (right): Critical Step DPO identifies bottleneck steps via rollout divergence between expert and local policies, constructs distributional preference pairs from all $n \times m$ cross-group rollouts, and trains the local model with criticality-weighted, pair-weighted DPO.

on runtime feedback.

Static Optimization. When invoking external tools via the Model Context Protocol (MCP), models are typically provided with tool descriptions that specify function signatures and usage guidelines. However, such descriptions are often overly brief and may omit edge-case behaviors, undocumented parameters, or input formatting constraints. To address this, we enable the agent to construct tool-specific skills by examining the concrete implementation of each MCP tool, building a more accurate mental model of tool capabilities, input requirements, and expected outputs.

Dynamic Optimization. Static analysis alone cannot capture tool-specific usage strategies—for instance, Google benefits from verbose queries while PubMed requires concise, keyword-focused terms. Through learning from historical invocation logs, the model acquires fine-grained strategies for each tool and identifies recurring failure patterns, enabling programmatic guardrails that proactively intercept known error modes (e.g., deduplication checks to block redundant API calls). The training-free phase produces refined tool-use behavior and, critically, generates the error correction logs that serve as the foundation for the subsequent training-based optimization stage.

3.2 Critical Step Direct Preference Optimization

While the training-free phase effectively addresses surface-level issues such as malformed tool calls and incorrect API parameters, deeper deficiencies—such as suboptimal query formulation, inefficient search-retrieval loops, or poor information synthesis—persist. These deficiencies are rooted in the model's decision-making at specific trajectory steps, and cannot be resolved without parameter updates.

A natural approach is trajectory-level DPO (Rafailov et al., 2023), which treats entire trajectories as preference pairs. However, trajectory-level methods distribute a single binary outcome reward uniformly across dozens of steps, most of which the model already handles correctly. The result is noisy, slow training that wastes both data and compute. We instead propose **Critical Step DPO (CDPO)**, an offline preference optimization approach that identifies the specific steps where model capability is the bottleneck and provides fine-grained, distributional supervision at

those steps, without the high cost and instability of online RL methods.

3.2.1 Notation and Problem Setup

We consider an agent operating over a task with query q. Following the ReAct paradigm (Yao et al., 2023), the agent produces a trajectory:

$$\tau = (s_1, a_1, o_1,\ s_2, a_2, o_2,\ \ldots,\ s_T, a_T, o_T) \quad (1)$$

where $s_t = (q, a_1, o_1, \ldots, a_{t-1}, o_{t-1})$ is the full history up to step t, $a_t \sim \pi_\theta(\cdot \mid s_t)$ is the policy action, and o_t is the environment observation. The trajectory terminates with an outcome score $r \in [0, A]$ assigned by a task-specific evaluation rubric (with A being the maximum score).

We maintain two models throughout:

- π_θ: the *local policy* being trained (e.g., an 8B open-source model);
- π_{exp}: a fixed *expert model* used only for generating positive alternatives (e.g., a frontier closed-source model).

The central insight of CDPO is that a step t is *critical* if and only if model capability is the bottleneck at that point: the expert model succeeds from state s_t where the local model fails. Steps that are easy for both models, or hard for both, carry no discriminative learning signal and should be excluded from training.

3.2.2 Critical Step Identification

We adopt a two-stage pipeline to identify critical steps: a cheap candidate detection stage followed by a rigorous rollout-based verification stage.

Stage 1: Candidate Detection. To avoid the prohibitive cost of running rollouts at every step of every trajectory, we first apply lightweight filters to identify candidate critical steps. Identifying critical steps typically relies on a Process Reward Model (PRM) to judge step quality (Lightman et al., 2024). However, training a dedicated PRM for agentic tool-use trajectories is costly, and directly using an LLM as a PRM judge suffers from instability, as current LLMs have limited exposure to multi-step tool-use evaluation during training. We instead combine two complementary signals:

1. **Rule-based filtering.** Observable features such as tool call failures, empty search results, repeated queries, and timeout errors serve as high-recall indicators of potential critical steps. For tasks with definitive answers, we additionally apply term-frequency analysis over rollout outputs to pinpoint the step at which an incorrect answer is first introduced or where the frequency of incorrect answers begins to exceed that of the correct one. The detailed algorithm is provided in Appendix A.
2. **Multi-LLM voting.** Candidate steps from the rule-based filter are further assessed by multiple LLMs that independently evaluate whether the step represents a genuine capability bottleneck. This ensemble-based approach provides precision without requiring a dedicated PRM.

Stage 1 only needs to function as a rough filter: false negatives are acceptable since they merely reduce the training set, while false positives are caught by the subsequent verification stage.

Stage 2: Rollout Divergence Verification. At each candidate step t identified by Stage 1, we verify its criticality through *rollout divergence*—a direct, outcome-based measure that requires no learned reward model. From state s_t, we generate n expert rollouts and m local rollouts, each continuing to task completion:

$$\{a^+_{t,1}, \ldots, a^+_{t,n}\} \sim \pi_{\text{exp}}(\cdot \mid s_t) \quad (2)$$

$$\{a^-_{t,1}, \ldots, a^-_{t,m}\} \sim \pi_\theta(\cdot \mid s_t) \quad (3)$$

Each rollout yields an outcome score $r^+_{t,i}$ or $r^-_{t,j}$ from the task evaluation rubric. We define the empirical value estimates as:

$$\hat{V}_{\text{exp}}(s_t) = \frac{1}{n}\sum_{i=1}^{n} r^+_{t,i}, \quad \hat{V}_{\text{local}}(s_t) = \frac{1}{m}\sum_{j=1}^{m} r^-_{t,j} \quad (4)$$

These are unbiased Monte Carlo estimates of the true state values $V_{\pi_{\text{exp}}}(s_t)$ and $V_{\pi_\theta}(s_t)$.

Definition 1 (Criticality Score)**.** *The criticality score of step t is the empirical advantage of the expert policy over the local policy from state s_t:*

$$\Delta_t = \hat{V}_{exp}(s_t) - \hat{V}_{local}(s_t) \ \in\ [-A, +A] \quad (5)$$

This score has a natural interpretation: $\Delta_t \approx 0$ indicates both models handle this step similarly (not critical); $\Delta_t \gg 0$ indicates the expert consistently outperforms the local model (highly critical); $\Delta_t < 0$ indicates the local model happens to perform better (rare, ignored).

Definition 2 (Verified Critical Step)**.** *Step t is a verified critical step if $\Delta_t > \epsilon$ for a threshold $\epsilon > 0$. The set of critical steps in trajectory τ is:*

$$\mathcal{C}_\tau = \{t \in [T] : \Delta_t > \epsilon\} \tag{6}$$

Crucially, Δ_t is computed entirely from observed task outcomes with no reliance on a process reward model. This eliminates PRM approximation error as a noise source and ensures that criticality estimates converge to the true advantage gap as the number of rollouts increases.

Dual-Context Rollout Construction. Following a *proactive prevention and post-hoc remedy* paradigm, we construct two rollout contexts for each critical step: (1) a context truncated before the critical tool call, allowing the model to learn to avoid the error proactively by choosing a better action; and (2) a context that includes the critical tool call and its observation, allowing the model to learn corrective alternatives after encountering the error. Both the local and expert models generate rollouts from these contexts, yielding preference pairs that capture complementary aspects of the error—prevention and recovery.

3.2.3 From Standard DPO to Distributional CDPO

Standard DPO (Rafailov et al., 2023) operates on a single preference pair (s, a^+, a^-) and maximizes:

$$\mathcal{L}_{\text{DPO}}(\theta) = -\log \sigma\left(\beta \log \frac{\pi_\theta(a^+ \mid s)}{\pi_{\text{ref}}(a^+ \mid s)} - \beta \log \frac{\pi_\theta(a^- \mid s)}{\pi_{\text{ref}}(a^- \mid s)}\right) \tag{7}$$

This uses exactly one bit of preference information per step. However, at each critical step our rollout procedure yields $n \times m$ cross-group preference observations. We leverage all of these observations through a distributional formulation.

All-Pairs Preference Aggregation. At each verified critical step t, the n expert and m local rollouts provide $n \times m$ pairwise comparisons. Rather than selecting a single "best" pair, we aggregate over all pairs to form a distributional training signal.

For rubric-scored tasks, each pair (i, j) carries a continuous score gap $r^+_{t,i} - r^-_{t,j}$ that naturally encodes preference strength: a pair where the expert scores 38 and the local model scores 12 provides a much stronger signal than a pair with scores 32 and 29. We normalize this gap by the rubric maximum to obtain a dimensionless pair-level weight:

$$\tilde{r}_{t,ij} = \frac{r^+_{t,i} - r^-_{t,j}}{A} \in [-1, +1] \tag{8}$$

The sign of $\tilde{r}_{t,ij}$ naturally handles all cases: positive weights push the preferred action above the dispreferred one, zero weights contribute no gradient, and negative weights (when the local rollout outscores the expert) flip the preference direction. All nm pairs contribute without binary filtering (see Appendix D.4 for a detailed example).

Criticality-Weighted Step Importance. Not all critical steps are equally informative. A step where the expert scores 30 points higher on average than the local model should receive more gradient update than one with only a 5-point gap. We encode this through a *criticality weight* $w(\Delta_t)$.

The most obvious choice, using $|\Delta_t|$ directly as the weight, fails for two reasons. First, $|\Delta_t|$ is measured in rubric points and is therefore not dimensionless—multiplying a log-probability loss by a rubric-point value creates a unit mismatch that the optimizer simply absorbs via the learning rate, rendering the weighting ineffective. Second, without knowing what a typical divergence looks like across the dataset, the raw value provides no way to rank steps relative to each other.

We resolve both issues through a normalization constant $\hat{Z}$, computed once before training over all candidate steps in the dataset $\mathcal{D}$:

$$\hat{Z} = \frac{1}{|\mathcal{D}|} \sum_{t \in \mathcal{D}} |\Delta_t|^\alpha \tag{9}$$

The normalized step weight is then:

$$w(\Delta_t) = \frac{|\Delta_t|^\alpha}{\hat{Z}} \tag{10}$$

By construction, the weights satisfy a *mean-1 property*: $\mathbb{E}_{\mathcal{D}}[w(\Delta_t)] = 1$. Since standard DPO implicitly assigns weight 1 to every step, CDPO with mean-1 normalization preserves the same total gradient budget but redistributes it—steps with $|\Delta_t|^\alpha > \hat{Z}$ are upweighted (more critical than average), steps with $|\Delta_t|^\alpha < \hat{Z}$ are downweighted, and steps with $|\Delta_t| \approx 0$ receive weight ≈ 0 (effectively skipped). Crucially, existing hyperparameters β and learning rate transfer directly because the average loss magnitude is unchanged. The normalization also provides *scale invariance*—changing the rubric scale leaves $w(\Delta_t)$ unchanged (see Appendix D.2)—and the exponent α controls how aggressively near-zero divergences are suppressed ($\alpha = 1$ linear is our default; see Appendix D.3 for design choices).

The Complete CDPO Loss. Combining the step-level criticality weight, pair-level score gap, and standard DPO log-likelihood, we arrive at the full CDPO objective:

$$\mathcal{L}_{\text{CDPO}}(\theta) = -\,\mathbb{E}_{\{t:\,|\Delta_t|>\epsilon\}}\Bigg[\underbrace{\frac{|\Delta_t|^{\alpha}}{\hat{Z}}}_{\text{step weight}} \cdot \frac{1}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m} \underbrace{\frac{r^{+}_{t,i}-r^{-}_{t,j}}{A}}_{\text{pair weight}} \cdot \log\sigma\Bigg(\beta\log\frac{\pi_\theta(a^{+}_{t,i}\,|\,s_t)}{\pi_{\text{ref}}(a^{+}_{t,i}\,|\,s_t)} - \beta\log\frac{\pi_\theta(a^{-}_{t,j}\,|\,s_t)}{\pi_{\text{ref}}(a^{-}_{t,j}\,|\,s_t)}\Bigg)\Bigg] \tag{11}$$

The loss decomposes into three cleanly separated, dimensionless components: (1) $|\Delta_t|^{\alpha}/\hat{Z}$—the step-level importance weight that is mean-1 normalized and scale-invariant, concentrating learning on steps where model capability is the bottleneck; (2) $(r^{+}_{t,i} - r^{-}_{t,j})/A$—the pair-level score gap that always lies in $[-1, +1]$, handles partial credit naturally, and replaces the binary valid-pair filter from standard formulations; and (3) $\log\sigma(\cdot)$—the standard DPO log-likelihood, unchanged from the original objective so that all existing convergence theory applies.

Under the Bradley-Terry preference model, the inner log-sigmoid term corresponds to the probability that the expert action is preferred over the local action. The pair-averaged sum therefore maximizes the expected log-likelihood of all nm observed preference events, making CDPO a maximum likelihood estimator over the full set of rollout comparisons.

3.2.4 Theoretical Properties

We establish three theoretical properties of CDPO that justify its design choices. All proofs are provided in Appendix B.2.

Proposition 1 (Gradient Variance Reduction)**.** *Let g_1 denote the gradient of standard DPO using a single preference pair, and let $\hat{g}_{n,m}$ denote the gradient of the normalized all-pairs average with n expert and m local rollouts. Then* $\text{Var}[\hat{g}_{n,m}] = \frac{1}{nm}\text{Var}[g_1]$*, assuming preference pairs are drawn i.i.d. from the rollout distribution.*

Proposition 2 (Unbiased Criticality Estimation)**.** *The empirical criticality score $\hat{\Delta}_t$ is an unbiased estimator of the true advantage gap $\Delta^*_t = V_{\pi_{exp}}(s_t) - V_{\pi_\theta}(s_t)$, with standard error* $\text{SE}(\hat{\Delta}_t) = \sqrt{\text{Var}[r^{+}_t]/n + \text{Var}[r^{-}_t]/m}$.

Proposition 3 (Sample Efficiency)**.** *Let $\rho \in (0, 1)$ be the fraction of steps that are truly critical ($\Delta^*_t > \epsilon$), and let K be the number of preference pairs needed for standard DPO to reach expected loss $\mathcal{L}^*$. Then CDPO reaches the same loss using at most $K_{CDPO} \le \frac{\rho}{nm} \cdot K$ preference pairs—a $\frac{nm}{\rho}$-fold reduction.*

Together, these results show that CDPO achieves nm-fold variance reduction per step (e.g., $16\times$ with $n = m = 4$), unbiased criticality estimation that converges to the true advantage gap, and $\frac{nm}{\rho}$-fold sample efficiency gains over standard DPO.

4 Experiments

4.1 Experimental Setup

Policy Model and Tools. We adopt DR-Tulu-8B (Shao et al., 2025) as the local policy model π_θ, an open-source deep research model built on Qwen3-8B that has been trained for multi-step information retrieval and long-form answer synthesis. The model interacts with external tools through the MCP-based agent infrastructure provided by DR-Tulu, which includes a suite of search and web-reading tools (e.g., web search, PubMed API search, page reading, and content extraction) implemented via FastMCP. We use the default `dr-agent` tool configuration throughout all experiments.

Benchmarks. We evaluate on three medical benchmarks: **CureBench**, a multiple-choice benchmark evaluated by exact string matching with LLM judge fallback; **MedBrowseComp**, a closed-form QA benchmark with the same evaluation protocol; and **PubMed Search**, a new PubMed-based deep research benchmark we introduce for biomedical literature retrieval and open-ended QA, evaluated via rubric-based scoring on factual accuracy, completeness, and attribution. All benchmarks use GPT-4.1-mini as the judge model.

Training Data. We construct task-specific data synthesis pipelines for each benchmark, generating training trajectories that capture the diversity of tool-use patterns and failure modes encountered in medical research tasks. The resulting dataset contains 16,951 trajectory samples: 5,672 from CureBench, 3,124 from MedBrowseComp, and 8,155 from PubMed Search. Critical step identification and preference pair construction follow the pipeline described in §3.2.2, with $n = m = 4$ rollouts per group and criticality threshold $\epsilon = 0.3$.

Table 1: Main results on three medical benchmarks. PubMed Search reports rubric-based scores; CureBench and MedBrowseComp report accuracy (%). Best results are **bolded**; second best are underlined. †Closed-source model with equivalent tool access.

Method	PubMed Search ↑	Cure Bench ↑	MedBrowse Comp ↑
GPT-5.2†	19.08	**71.05**	27.93
DR-Tulu-8B	15.48	59.30	25.12
+ Skill	17.75	62.41	26.61
+ SFT	17.93	63.42	27.44
+ Traj. DPO	18.47	63.05	27.19
+ Step DPO	18.83	64.25	28.06
Skill-CDPO	**19.58**	66.63	**29.09**

Training Details. For DPO training, we set the KL penalty coefficient $\beta = 0.5$ and use the $\alpha = 1$ (linear) weighting scheme as default. We train for 3 epochs with a learning rate of 5×10^{-6} and batch size of 32. The reference model π_{ref} is initialized as the base DR-Tulu-8B checkpoint. All experiments are conducted on 8 NVIDIA A100 80GB GPUs.

Baselines. We compare against: **DR-Tulu-8B** (base model), **+ Skill** (training-free skill acquisition only), **+ SFT** (behavioral cloning on expert trajectories), **+ Trajectory DPO** (trajectory-level preference pairs), and **+ Step DPO** (step-level DPO with uniform weighting and single pairs). All training-based methods include the Skill phase for fair comparison. We additionally report GPT-5.2 with equivalent tool access as a closed-source reference.

4.2 Main Results

Table 1 presents the main results. We highlight the following findings.

Skill synthesis is necessary and provides an essential foundation. The training-free Skill phase alone yields substantial gains over the base model (+2.27 on PubMed Search, +3.11% on CureBench, +1.49% on MedBrowseComp) without any parameter updates, confirming that inference-time skill synthesis provides a necessary foundation for subsequent optimization.

Standard training methods cannot fully exploit skill-augmented agents. On top of the Skill phase, SFT provides only marginal additional improvement (+0.18 on PubMed Search, +1.01% on CureBench, +0.83% on MedBrowseComp), confirming that behavioral cloning fails to teach the model *when and how* to avoid errors. Trajectory DPO shows similarly limited gains, as it distributes outcome rewards uniformly across all steps rather than targeting specific capability bottlenecks. Even Step DPO, which operates at the step level with uniform weighting and single pairs, offers only modest improvements over Trajectory DPO.

Table 2: Ablation study. Each row removes one component from the full Skill-CDPO model.

Variant	PubMed	Cure	MedBrowse
Skill-CDPO (Full)	19.58	66.63	29.09
– Training-free Skill	18.36	64.71	27.68
– Critical step select.	18.41	64.15	27.53
– Criticality weight	19.14	65.89	28.51
– Pair-level weight	18.71	66.25	28.82
– All-pairs (single pair)	19.07	65.62	28.38

CDPO fully unlocks the potential of skill-augmented agents. Skill-CDPO achieves substantial improvements over all training baselines (+0.75 over Step DPO on PubMed Search, +2.38% on CureBench, +1.03% on MedBrowseComp), confirming that criticality-weighted distributional supervision captures learning signals that other methods miss. Crucially, Skill-CDPO outperforms GPT-5.2 on PubMed Search (+0.50) and MedBrowseComp (+1.16%), demonstrating that targeted critical step optimization enables a compact 8B-parameter model to surpass a larger closed-source system on retrieval-intensive tasks. On CureBench, GPT-5.2 retains a lead (71.05% vs. 66.63%), likely due to broader parametric knowledge, but Skill-CDPO closes the gap from 11.75% to 4.42%.

4.3 Analysis

Ablation Study. To understand the contribution of each component, we remove one at a time from the full model. Results in Table 2 yield the following observations.

(1) **Critical step selection is the most impactful component.** Removing the criticality threshold and training on all steps produces the largest average degradation (-1.17 on PubMed Search, -2.48% on CureBench, -1.56% on MedBrowseComp), confirming that including non-critical steps dilutes the gradient with noise.

(2) **Training-free skill acquisition provides an essential foundation.** Removing the Skill phase leads to substantial drops, indicating that it not only improves inference-time behavior but also produces higher-quality trajectories for preference pair

Table 3: Sensitivity to the weighting exponent α. Default setting ($\alpha = 1.0$) shown in **bold**.

	PubMed	Cure	MedBrowse
$\alpha = 0.5$	19.32	65.91	28.87
$\boldsymbol{\alpha = 1.0}$	**19.58**	**66.63**	**29.09**
$\alpha = 2.0$	19.41	65.38	28.64

construction.

(3) **Pair-level score weighting disproportionately affects rubric-scored tasks.** Reverting to binary preference pairs causes the largest drop on PubMed Search (-0.87), where continuous score gaps are most informative, while the effect is minimal on binary-outcome CureBench (-0.38%).

(4) **The all-pairs formulation provides consistent variance reduction.** Using a single randomly sampled pair instead of all nm pairs degrades performance across all benchmarks (-0.51, -1.01%, -0.71%), empirically confirming Proposition 1.

(5) **Criticality weighting offers a moderate but consistent benefit**, indicating that the degree of criticality carries useful signal beyond the binary selection decision.

Critical Step Statistics. Only 14.3% of steps are verified critical ($\Delta_t > \epsilon$), confirming the sparsity assumption in Proposition 3. For CureBench and MedBrowseComp training sets—which have definitive answers—we further classify wrong-answer errors via the algorithm in Appendix A: **misled by evidence** (39.3%, correct reasoning derailed by a tool failure), **final reasoning error** (36.9%, correct reasoning but wrong extracted letter; 42% of these have the correct answer stated verbatim in the reasoning text), **evidence ignored** (13.1%, correct evidence retrieved but dismissed), and **insufficient search** (10.7%, correct evidence never retrieved). The dominant failure mode is *evidence integration under uncertainty*—abandoning correct conclusions after tool failures or failing to act on retrieved evidence—accounting for over half of all errors and explaining CDPO's largest gains at these steps. Representative cases are in Appendix H.

Sensitivity to α. Table 3 reports the sensitivity of CDPO to the weighting exponent α. The linear weighting ($\alpha = 1$) performs best on average across all three benchmarks. The concave choice $\alpha = 0.5$ is competitive, particularly on MedBrowseComp where noisier rollout estimates benefit from not over-emphasizing steps with large but potentially noisy $\hat{\Delta}_t$ values. The convex choice $\alpha = 2$ concentrates too aggressively on a few extreme steps, leading to slight underfitting on the remaining critical steps.

5 Conclusion

We presented Skill-CDPO, a progressive framework for evolving the tool-use capabilities of compact language models. Our experiments reveal two key findings. First, inference-time skill synthesis—through static tool analysis and dynamic strategy refinement—provides a necessary foundation that substantially narrows the gap between compact and large-scale models without any parameter updates. Second, and more importantly, Critical Step DPO (CDPO) further unlocks the potential of these newly acquired skills by concentrating preference learning on the specific trajectory steps where model capability is the bottleneck. Through rollout-divergence-based step identification, all-pairs distributional supervision, and criticality-weighted training, CDPO resolves the credit assignment problem without a process reward model or online RL. On three medical agent benchmarks, an 8B-parameter Skill-CDPO model not only substantially outperforms SFT and trajectory-level DPO baselines, but also surpasses GPT-5.2 on retrieval-intensive tasks—demonstrating that targeted critical step optimization can compensate for orders-of-magnitude fewer parameters. We additionally contribute PubMed Search, a new PubMed-based deep research benchmark for biomedical literature retrieval.

Limitations

Dependence on an expert model. CDPO requires access to a frontier closed-source model π_{exp} for generating positive rollouts at critical steps. This introduces both a cost overhead and a capability ceiling: the local model can only learn to match behaviors that the expert can demonstrate. In settings where no substantially stronger model is available, the rollout divergence signal may be weak or absent.

Domain and model coverage. Our experiments are conducted exclusively on medical benchmarks with a single 8B-parameter base model (DR-Tulu-8B). While the method is domain-agnostic in design, we have not verified its effectiveness on other specialized domains (e.g., legal, financial) or on models of different scales and architectures.

Data synthesis cost. Although CDPO achieves higher data utilization efficiency than trajectory-level DPO by concentrating learning on critical steps, the upstream cost of synthesizing the preference data is substantial: each candidate critical step requires $n + m$ complete trajectory rollouts from both the expert and local models, and the two-stage identification pipeline itself involves multi-LLM voting and rule-based filtering over the full trajectory set. These costs are incurred once offline before training, but may be prohibitive when the tool-interaction environment is slow or expensive to query.

Single-round training. Although we discuss iterative refinement in Appendix E, all reported results are from a single round of CDPO training. The benefits and potential instabilities of multi-round iterative CDPO remain empirically unvalidated.

Ethics Statement

All benchmarks used in this work are constructed from publicly available biomedical literature and do not involve private patient data or protected health information. PubMed Search, the benchmark we introduce, consists of open-ended research questions paired with information retrievable through the public PubMed API; no human subjects are involved in its construction or evaluation. Our framework is designed to improve tool-use reliability of language model agents and does not generate medical advice intended for clinical decision-making. We caution that outputs from any agent system—including those trained with Skill-CDPO—should not be used as a substitute for professional medical judgment. The expert model used for rollout generation is accessed through standard commercial APIs in compliance with the provider's terms of service.

References

Anthropic. 2024. Building effective agents. https://www.anthropic.com/research/building-effective-agents.

Mohammad Gheshlaghi Azar, Mark Rowland, Bilal Piot, Daniel Guo, Daniele Calandriello, Michal Valko, and Rémi Munos. 2024. A general theoretical paradigm to understand learning from human preferences. In *International Conference on Artificial Intelligence and Statistics*.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. KTO: Model alignment as prospect theoretic optimization. In *International Conference on Machine Learning*.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. 2019. PubMedQA: A dataset for biomedical research question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2567–2577. Association for Computational Linguistics.

Xin Lai, Zhuotao Tian, Yukang Chen, Senqiao Bian, Jing Wang, and Jiaya Liu. 2024. Step-DPO: Step-wise preference optimization for long-chain reasoning of LLMs. *arXiv preprint arXiv:2406.18629*.

Mukai Li, Qingcheng Zeng, Tianqing Fang, Zhenwen Liang, Linfeng Song, Qi Liu, Haitao Mi, and Dong Yu. 2026. Verified critical step optimization for LLM agents. *arXiv preprint arXiv:2602.03412*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *International Conference on Learning Representations*.

Yu Meng, Mengzhou Xia, and Danqi Chen. 2024. SimPO: Simple preference optimization with a reference-free reward. *Advances in Neural Information Processing Systems*.

Marvin Minsky. 1961. Steps toward artificial intelligence. *Proceedings of the IRE*, 49(1):8–30.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*.

Pranav Putta, Edmund Mills, Naman Garg, Sumit Motwani, Chelsea Finn, Divyansh Garg, and Rafael Rafailov. 2024. Agent Q: Advanced reasoning and learning for autonomous AI agents. *arXiv preprint arXiv:2408.07199*.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, and 1 others. 2024. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *International Conference on Learning Representations*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *International Conference on Learning Representations*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Rulin Shao, Akari Asai, Shannon Zejiang Shen, Hamish Ivison, Varsha Kishore, Jingming Zhuo, Xinran Zhao, Molly Park, Samuel G. Finlayson, David Sontag, and 1 others. 2025. DR Tulu: Reinforcement learning with evolving rubrics for deep research. *arXiv preprint arXiv:2511.19399*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y.K. Li, Y. Wu, and Daya Guo. 2024. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Karan Singhal, Shekoofeh Azizi, Tao Tu, S Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, and 1 others. 2023. Large language models encode clinical knowledge. *Nature*, 620:172–180.

Yifan Song, Pengjun Da, Amber Zhan, Zeyun Hu, Rui Nie, Zefan Xu, Caiwen Zhang, and Kyle Lo. 2024. Trial and error: Exploration-based trajectory optimization of LLM agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*.

Wei Xiong, Chengshuai Shi, Jiaming Shen, Aviv Rosbach, Tong Qin, Daniele Calandriello, Misha Khalman, Rishabh Joshi, Bilal Piot, Mohammad Saleh, and 1 others. 2025. Building math agents with multi-turn iterative preference learning. In *International Conference on Learning Representations*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*.

Yongcheng Zeng, Guoqing Liu, Weiyu Ma, Ning Yang, Haifeng Zhang, and Jun Wang. 2024. Token-level direct preference optimization. In *International Conference on Machine Learning*.

A Discriminative Critical Turn Localization Algorithm

For benchmarks with definitive ground-truth answers (CureBench and MedBrowseComp), we supplement the observable-feature heuristics with a *discriminative critical turn localization* procedure that identifies the specific reasoning turn where the agent's trajectory diverges from the correct answer, and classifies the error into one of four mechanistic subtypes. The entire pipeline is rule-based with no LLM calls.

Stage 1: Discriminative Keyword Extraction

Given the option set $\mathcal{O} = \{o_1, \ldots, o_K\}$ for a multiple-choice question, we extract a weighted keyword set for each option. For option o_i with text t_i, we tokenize t_i into lowercase alphanumeric tokens and remove a standard English stopword list. Crucially, we impose *no minimum token length*, since short but highly discriminative medical abbreviations (e.g., "HIV", "CT", "CRP") would otherwise be discarded.

We then partition each option's keyword set $\mathcal{K}_i$ into:

$$\mathcal{U}_i = \mathcal{K}_i \setminus \bigcup_{j \neq i} \mathcal{K}_j \quad \text{(unique keywords)} \tag{12}$$

$$\mathcal{S}_i = \mathcal{K}_i \setminus \mathcal{U}_i \quad \text{(shared keywords)} \tag{13}$$

Unique keywords appear only in option o_i and carry high discriminative value; shared keywords appear in multiple options and are down-weighted. Each keyword k is assigned weight $w(k) = 1.0$ if $k \in \mathcal{U}_i$ and $w(k) = 0.3$ if $k \in \mathcal{S}_i$.

Stage 2: Per-Turn Option Tendency Scoring

The agent trajectory is parsed into a sequence of turns $\mathcal{T} = [t_1, \ldots, t_N]$, where each turn contains a reasoning segment think_j (content within `<think>` tags), a tool call, and a tool output. For each turn t_j and option o_i, we compute a tendency score:

$$\text{score}(t_j, o_i) = \sum_{k \in \mathcal{U}_i} \text{count}(k,\, \text{think}_j) + 0.3 \sum_{k \in \mathcal{S}_i} \text{count}(k,\, \text{think}_j) \tag{14}$$

where $\text{count}(k, \cdot)$ counts case-insensitive occurrences of keyword k.

Stage 3: Trajectory-Based Critical Turn Localization

Rather than simply finding the first turn where the wrong option dominates, we analyze the full score trajectory via a *tendency gap*:

$$\text{gap}[j] = \text{score}(t_j, o_{\text{wrong}}) - \text{score}(t_j, o_{\text{correct}}) \tag{15}$$

where o_{wrong} is the model's (incorrect) answer and o_{correct} is the ground truth. A negative gap indicates the model leans toward the correct answer at turn j; a positive gap indicates a lean toward the wrong answer.

We compute two key trajectory anchors:

- **Last Correct Turn (LCT):** the last index j where $\text{gap}[j] < 0$, i.e., the correct answer was still favored. If no such turn exists, $\text{LCT} = \varnothing$.
- **Maximum Shift Turn (MST):** the index $j \geq 1$ maximizing the positive delta $\text{gap}[j] - \text{gap}[j-1]$, i.e., the turn where the model's tendency shifted most sharply toward the wrong answer. If all deltas ≤ 0, $\text{MST} = \varnothing$.

The critical turn interval is then:

- $\text{LCT} \neq \varnothing$ and $\text{MST} \neq \varnothing$: interval $[\text{LCT}, \text{MST}]$ (the full transition zone).
- Only $\text{MST} \neq \varnothing$: interval $[\text{MST} - 1, \text{MST}]$.
- Only $\text{LCT} \neq \varnothing$: interval $[\text{LCT}, \text{LCT} + 1]$.
- Neither: last 2 turns (fallback).

Stage 4: Error Sub-Classification

We additionally detect tool-output evidence signals: for each turn t_j, let $\text{cev}[j] = \mathbf{1}[\exists\, k \in \mathcal{U}_{\text{correct}} : k \in \text{tool\_output}_j]$ indicate whether the tool output contains a unique keyword of the correct option. Define $\text{AnyCorrectEv} = \bigvee_j \text{cev}[j]$ and $\text{CorrectEverFavored} = (\text{LCT} \neq \varnothing)$.

The four subtypes are determined by the following decision rule:

Correct Ever Favored	MST	Correct Ev.	Subtype
True	$\neq \varnothing$	—	MISLED-BY-EVIDENCE
True	$= \varnothing$	—	FINAL-REASONING-ERROR
False	—	True	EVIDENCE-IGNORED
False	—	False	INSUFFICIENT-SEARCH

MISLED-BY-EVIDENCE: the model initially favored the correct answer but was derailed at MST, typically by a failed or misleading tool call. Critical turns: $[\text{LCT}, \text{MST}]$.

FINAL-REASONING-ERROR: the model consistently favored the correct answer (no sharp shift) but output the wrong letter. Critical turns: last 2 turns.

EVIDENCE-IGNORED: correct-answer evidence appeared in tool outputs but was never incorporated into reasoning. Critical turns: the turn(s) where $\text{cev}[j] = 1$ and the immediately following turn.

INSUFFICIENT-SEARCH: no correct-answer evidence was retrieved at any point. Critical turns: first turn and the turn with the highest wrong-option evidence signal.

B Additional Theoretical Analysis

B.1 False Positive Bound for Critical Step Selection

A natural concern is whether the thresholding procedure $\Delta_t > \epsilon$ may incorrectly flag non-critical steps. We provide a formal bound on this false positive rate.

Proposition 4 (False Positive Bound)**.** *Suppose the true criticality gap is $\Delta_t^* = 0$ (step t is not critical). With $n = m = k$ rollouts per group, the probability of a false positive detection $\hat{\Delta}_t > \epsilon$ is bounded by:*

$$P(\hat{\Delta}_t > \epsilon \mid \Delta_t^* = 0) \leq \exp\left(-\frac{k\epsilon^2}{2A^2}\right) \tag{16}$$

Proof. Under $\Delta_t^* = 0$, we have $\mathbb{E}[\hat{\Delta}_t] = 0$. Define paired differences $D_i = r_{t,i}^+ - r_{t,i}^- \in [-A, +A]$, so that $\hat{\Delta}_t = \frac{1}{k}\sum_{i=1}^k D_i$. By Hoeffding's inequality:

$$\begin{aligned} P(\hat{\Delta}_t > \epsilon) &\leq \exp\left(-\frac{2k^2\epsilon^2}{k \cdot (A - (-A))^2}\right) \\ &= \exp\left(-\frac{k\epsilon^2}{2A^2}\right) \quad \square \end{aligned} \tag{17}$$

Corollary 1. *For binary-outcome tasks ($A = 1$) with $k = 4$ rollouts per group and threshold $\epsilon = 0.5$: false positive rate $\leq e^{-0.5} \approx 0.61$. With $k = 8$: $\leq e^{-1.0} \approx 0.37$. With $k = 16$: $\leq e^{-2.0} \approx 0.14$.*

This bound confirms that increasing the number of rollouts k exponentially suppresses the false positive rate, and that even moderate values of k combined with a reasonable threshold ϵ provide meaningful control over spurious critical step detections.

B.2 Full Proofs

Proof of Proposition 1 (Gradient Variance Reduction). Let ℓ_{ij} denote the loss contribution of the (i, j) preference pair. The normalized all-pairs loss is $\hat{\mathcal{L}} = \frac{1}{nm}\sum_{i=1}^n \sum_{j=1}^m \ell_{ij}$. Taking the gradient:

$$\nabla_\theta \hat{\mathcal{L}} = \frac{1}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m} \nabla_\theta \ell_{ij} \tag{18}$$

Since the rollouts are i.i.d., the gradients $\nabla_\theta \ell_{ij}$ are pairwise independent with the same distribution as

$g_1 = \nabla_\theta \ell_{11}$. By independence:

$$\begin{aligned}\mathrm{Var}[\nabla_\theta \hat{\mathcal{L}}] &= \frac{1}{(nm)^2} \sum_{i,j} \mathrm{Var}[\nabla_\theta \ell_{ij}] \\ &= \frac{nm}{(nm)^2} \mathrm{Var}[g_1] = \frac{1}{nm} \mathrm{Var}[g_1] \quad \square\end{aligned} \tag{19}$$

Proof of Proposition 2 (Unbiased Criticality Estimation). Since $r_{t,i}^+$ are i.i.d. draws from the expert rollout distribution with mean $V_{\pi_{\text{exp}}}(s_t)$, the sample mean $\hat{V}_{\text{exp}}(s_t) = \frac{1}{n}\sum_{i=1}^n r_{t,i}^+$ is unbiased by linearity of expectation. The same holds for $\hat{V}_{\text{local}}(s_t)$. Since the two groups of rollouts are independent:

$$\begin{aligned}\mathbb{E}[\hat{\Delta}_t] &= \mathbb{E}[\hat{V}_{\text{exp}}(s_t)] - \mathbb{E}[\hat{V}_{\text{local}}(s_t)] \\ &= V_{\pi_{\text{exp}}}(s_t) - V_{\pi_\theta}(s_t) = \Delta_t^*\end{aligned} \tag{20}$$

The variance formula follows from independence and the standard variance of a sample mean: $\mathrm{Var}[\hat{\Delta}_t] = \mathrm{Var}[r_t^+]/n + \mathrm{Var}[r_t^-]/m$. $\square$

Proof of Proposition 3 (Sample Efficiency). Standard DPO requires K gradient steps, each using one preference pair with gradient variance $\mathrm{Var}[g_1]$. The total "variance budget" consumed is $K \cdot \mathrm{Var}[g_1]$. CDPO operates only on the fraction ρ of steps that are truly critical. At each critical step, it uses nm pairs with normalized variance $\mathrm{Var}[g_1]/(nm)$ (Proposition 1). To match the same total variance reduction:

$$\begin{aligned}K_{\text{CDPO}} \cdot \frac{\mathrm{Var}[g_1]}{nm} &= \rho \cdot K \cdot \mathrm{Var}[g_1] \\ \Longrightarrow \quad K_{\text{CDPO}} &= \frac{\rho}{nm} \cdot K\end{aligned} \tag{21}$$

Simplifying: $K_{\text{CDPO}} \leq \frac{\rho}{nm} \cdot K$, yielding a $\frac{nm}{\rho}$-fold reduction. $\square$

C Comparison with Step-Level Preference Methods

Table 4 provides a systematic comparison of CDPO with existing step-level preference optimization methods.

Key advantages of CDPO over prior step-level methods:

1. **No PRM required.** Criticality is measured directly from task outcomes via rollout divergence, eliminating the need to train or maintain a separate process reward model.
2. **Distributional supervision.** All nm rollout outcomes contribute to the training signal through the all-pairs formulation, rather than using a single "best" pair.
3. **Continuous weighting.** The criticality weight $w(\Delta_t)$ encodes the degree of criticality, unlike binary select/reject schemes.
4. **Formal guarantees.** Propositions 1–3 provide theoretical backing for variance reduction, unbiasedness, and sample efficiency.

D CDPO Weight Design: Detailed Examples

D.1 Step-Level Weight Allocation Example

To illustrate how the normalized step weight $w(\Delta_t) = |\Delta_t|^\alpha / \hat{Z}$ redistributes gradient budget, consider a concrete example with $\alpha = 1$ and rubric maximum $A = 40$. Suppose Stage 1 identifies 5 candidate steps with the following empirical criticality scores:

The sum of weights is $0.25 + 1.00 + 1.875 + 0.375 + 1.50 = 5.0$, confirming the mean-1 property ($5.0/5 = 1.0$). Step 3 has the largest divergence—the local model fails badly while the expert succeeds—and receives the strongest gradient update ($1.875\times$ standard DPO). Step 1 has nearly no divergence between models and is effectively downweighted ($0.25\times$).

Figure 2 visualizes this redistribution. The left panel shows raw criticality scores with the $\hat{Z} = 8.0$ threshold, while the right panel shows the resulting normalized weights centered around the mean-1 baseline.

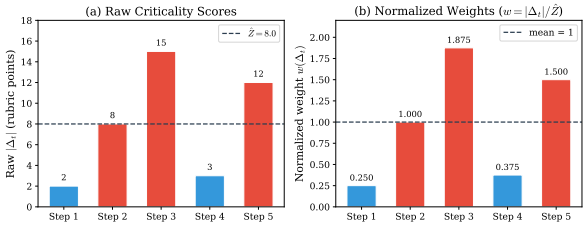

Figure 2: Step-level weight allocation. (a) Raw criticality scores $|\Delta_t|$ with the normalization constant $\hat{Z} = 8.0$ (dashed). (b) Normalized weights $w(\Delta_t) = |\Delta_t|/\hat{Z}$, averaging to 1 (dashed). Red bars indicate upweighted steps; blue bars indicate downweighted steps.

D.2 Scale Invariance of Normalized Weights

A key property of the $\hat{Z}$-normalized weights is that they are completely unaffected by the rubric scale

Table 4: Comparison of step-level preference learning methods. CDPO is the only method that combines outcome-verified step selection (no PRM), distributional supervision from multiple rollouts, and continuous criticality weighting.

Method	**Step Selection**	**Signal Source**	**Training Signal**	**Reward Model**
Trajectory DPO (ETO)	None (full trajectory)	Outcome	Coarse, uniform	Not required
Step-DPO	PRM score	PRM estimate	Dense, noisy	Required
IPR	Outcome-verified	MC rollout	Step-level, binary	Not required
CSO	PRM + verification	Outcome	Binary weight	Required
CDPO (Ours)	**Rollout divergence**	**Outcome (verified)**	**Distributional, weighted**	**Not required**

Table 5: Step-level weight allocation example ($\alpha = 1$, $A = 40$). The normalization constant $\hat{Z} = (2 + 8 + 15 + 3 + 12)/5 = 8.0$ ensures weights average to 1. Steps with larger divergence receive proportionally more gradient update.

Step	$\lvert\Delta_t\rvert$ **(pts)**	$\hat{Z}$	$w(\Delta_t)$	**Verdict**
Step 1	2	8.0	0.25×	Downweighted ↓
Step 2	8	8.0	1.00×	Average
Step 3	15	8.0	1.875×	Upweighted ↑↑
Step 4	3	8.0	0.375×	Downweighted ↓
Step 5	12	8.0	1.50×	Upweighted ↑

A. If the same task used $A' = 2A$ instead of A, every criticality score doubles, but so does $\hat{Z}$:

$$\Delta_t \to 2\Delta_t, \quad \hat{Z} \to 2^\alpha \hat{Z}$$
$$\implies \quad w(\Delta_t) = \frac{|\Delta_t|^\alpha}{\hat{Z}} \text{ unchanged} \qquad (22)$$

This means one can switch rubrics (e.g., from 40-point to 100-point grading) without retuning any hyperparameters. $\hat{Z}$ acts as an automatic unit converter, absorbing the rubric scale.

Figure 3 illustrates this property. Although the raw scores double when switching from $A = 40$ to $A = 80$, the normalized weights remain identical.

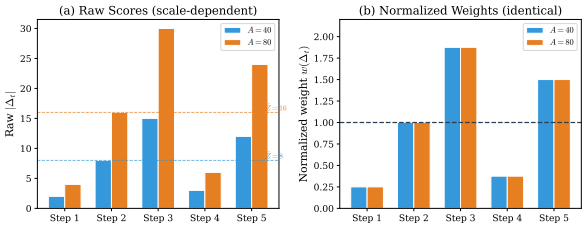

Figure 3: Scale invariance of $\hat{Z}$-normalized weights. (a) Raw criticality scores under $A = 40$ and $A = 80$: all values double, as do $\hat{Z}$. (b) Normalized weights are identical regardless of rubric scale, confirming that $\hat{Z}$ absorbs A automatically.

D.3 Weighting Exponent α: Design Choices

The exponent α in $|\Delta_t|^\alpha$ controls how aggressively the weight function discriminates between steps of different criticality. Table 6 summarizes the design space.

Table 6: Design choices for the weighting exponent α. All choices satisfy the mean-1 property; they differ only in how sharply they discriminate between steps.

α	**Shape**	**Effect**	**When to use**
0.5	Concave	Small divergences get significant weight	Noisy estimates; avoid missing signals
1	Linear	Proportional to normalized gap	Default — simple and interpretable
2	Convex	Near-zero gaps strongly suppressed	High-quality rollouts; sharp focus

We recommend starting with $\alpha = 1$. If rollout estimates are noisy (small n, m), $\alpha = 0.5$ avoids over-emphasizing steps with accidentally large $\hat{\Delta}_t$. If rollouts are plentiful and reliable, $\alpha = 2$ concentrates learning most aggressively on the hardest bottlenecks.

Figure 4 plots the weight function $w(|\Delta_t|/\hat{Z})$ for each α, showing the concave, linear, and convex shapes respectively.

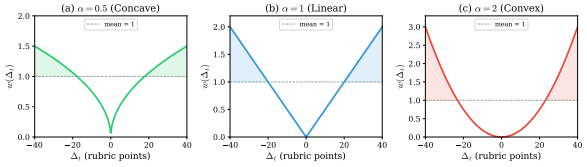

Figure 4: Weight function shapes for different α values. (a) $\alpha = 0.5$: concave—small divergences still receive significant weight. (b) $\alpha = 1$: linear—weight proportional to normalized gap. (c) $\alpha = 2$: convex—near-zero gaps are strongly suppressed. The dashed line marks $w = 1$ (average weight).

D.4 Pair-Level Score Weight Example

Within each critical step, individual rollout pairs also differ in informativeness. The pair-level weight $\tilde{r}_{t,ij} = (r^+_{t,i} - r^-_{t,j})/A$ handles three cases naturally, as shown in Table 7.

Table 8 shows a concrete 4×4 pair weight matrix for a critical step with $A = 40$.

Table 7: Three cases handled by the pair-level score weight $\tilde{r}_{t,ij}$. The sign of the weight determines the direction and magnitude of the preference gradient.

Condition	$\tilde{r}_{t,ij}$	Effect
$r^+_{t,i} > r^-_{t,j}$	Positive	Push a^+ over a^-
$r^+_{t,i} = r^-_{t,j}$	Zero	No gradient
$r^+_{t,i} < r^-_{t,j}$	Negative	Flip preference

Table 8: Pair-level score weights for $n = m = 4$ rollouts ($A = 40$). Expert scores: 38, 32, 35, 29. Local scores: 18, 25, 12, 22. Larger score gaps (higher values) push harder on the preference objective. All 16 pairs contribute with no binary filtering.

	Local 1 (18)	**Local 2** (25)	**Local 3** (12)	**Local 4** (22)
Expert 1 (38)	0.50	0.33	0.65	0.40
Expert 2 (32)	0.35	0.17	0.50	0.25
Expert 3 (35)	0.42	0.25	0.57	0.33
Expert 4 (29)	0.28	0.10	0.42	0.17

Pairs with larger score gaps (e.g., Expert 1 vs. Local 3: $\tilde{r} = 0.65$) provide stronger supervision, while near-tied pairs (e.g., Expert 4 vs. Local 2: $\tilde{r} = 0.10$) contribute minimally. This continuous weighting replaces the binary valid-pair filter of the standard formulation, where pairs are either fully included or fully excluded.

Figure 5 visualizes the full 4×4 pair weight matrix as a heatmap, where darker cells correspond to more informative pairs.

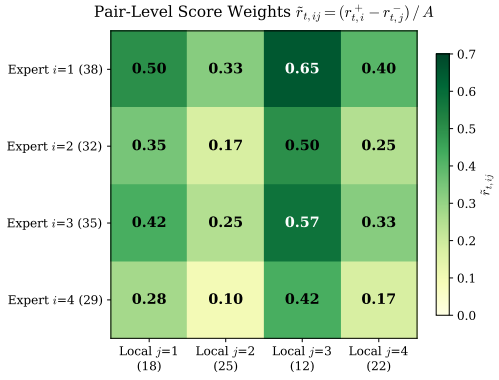

Figure 5: Pair-level score weight heatmap for $n = m = 4$ rollouts ($A = 40$). Each cell shows $\tilde{r}_{t,ij} = (r^+_{t,i} - r^-_{t,j})/A$. Darker green indicates larger score gaps and stronger preference supervision. All 16 pairs contribute with no binary filtering.

D.5 Binary QA vs. Rubric CDPO: Component-Level Comparison

Table 9 summarizes the component-level differences between the binary QA version of CDPO and the rubric-based extension presented in this work.

The rubric loss is strictly cleaner than the binary version: $\hat{Z}$ makes step weights scale-free and mean-1; replacing $|\mathcal{P}_t|$ with $1/nm$ removes the correlation between the normalizer and the pair count; and the pair-level score gap provides continuous, partial-credit supervision across all nm rollout combinations with no edge cases.

E Iterative Refinement

The CDPO framework naturally supports iterative refinement. After each training round, the updated policy π_{θ_i} can be deployed to collect fresh trajectories, and the full critical step identification and preference construction pipeline is repeated:

$$\pi_{\theta_0} \xrightarrow{\mathcal{D}^{(0)}_{\text{pref}}} \pi_{\theta_1} \xrightarrow{\mathcal{D}^{(1)}_{\text{pref}}} \pi_{\theta_2} \xrightarrow{\cdots} \tag{23}$$

At iteration i, the reference model is set to $\pi_{\text{ref}} = \pi_{\theta_{i-1}}$ for stable KL penalization. As the policy improves, previously critical steps become non-critical (the local model has learned to handle them), and new, harder decision points emerge as critical. This induces a natural curriculum learning effect: early iterations address frequent, basic errors (e.g., malformed queries), while later iterations target subtler deficiencies (e.g., suboptimal information synthesis strategies). We leave empirical validation of multi-round iterative CDPO to future work.

F Discussion on Weight Design Choices

Split Estimation for Selection and Weighting. The linear weighting $w(\Delta_t) = \Delta_t / \hat{Z}$ is theoretically motivated by advantage-weighted regression, but introduces a subtle correlation: the same rollouts are used both to select critical steps ($\Delta_t > \epsilon$) and to weight them ($w(\Delta_t)$). This can create a positive bias—steps selected because of a lucky high $\hat{\Delta}_t$ also receive inflated weight. A principled solution is split estimation: partition the rollout budget into two independent batches, using one for selection and the other for weighting. This eliminates the correlation at the cost of doubling the rollout budget per candidate step. In practice, we find that the bias is small relative to other sources of noise, and the simpler single-batch approach suffices.

Table 9: Component-level comparison between binary QA CDPO and rubric-based CDPO. The rubric formulation replaces discrete components with continuous alternatives, eliminating edge cases and providing richer supervision.

Component	Binary QA	Rubric (this work)
Criticality score Δ_t	$\hat{V}_{\text{exp}} - \hat{V}_{\text{local}} \in [0, 1]$	$\bar{r}_t^+ - \bar{r}_t^- \in [-A, +A]$
Step weight	$w(\Delta_t)$ raw or binary	$\lvert\Delta_t\rvert^\alpha / \hat{Z}$ (mean-1, scale-free)
Pair selection	$\lvert\mathcal{P}_t\rvert$ pairs where $y^+ = 1, y^- = 0$	All nm pairs
Pair normalizer	$1/\lvert\mathcal{P}_t\rvert$ (variable, correlated with Δ_t)	$1/nm$ (constant)
Pair weight	1 (binary)	$(r_{t,i}^+ - r_{t,j}^-)/A \in [-1, +1]$
Division-by-zero risk	Yes (when $\lvert\mathcal{P}_t\rvert = 0$)	No

Optimal Rollout Budget Allocation. Proposition 4 and Proposition 3 suggest a tradeoff: larger n, m reduces false positives and gradient variance, but increases compute per candidate step. An adaptive allocation strategy—running more rollouts at high-uncertainty candidates (where $\hat{\Delta}_t$ is close to ϵ) and fewer at clearly critical or clearly non-critical steps—could optimize this tradeoff automatically. We leave the exploration of such adaptive schemes to future work.

G PubMed Search Benchmark Construction

Existing PubMed-oriented benchmarks, such as PubMedQA (Jin et al., 2019), were developed under a closed-form question-answering paradigm: each instance has a definitive answer, and evaluation reduces to binary or categorical matching. While useful for measuring factual recall, this format cannot assess a model's ability to formulate effective search strategies, iteratively refine queries, synthesize information from multiple retrieved articles, and produce well-attributed, comprehensive responses—capabilities that are central to real-world biomedical literature research. To fill this gap, we introduce **PubMed Search**, a deep research benchmark built around the PubMed API that evaluates open-ended, multi-step retrieval and synthesis.

Question Generation. We sample seed topics from the Medical Subject Headings (MeSH) thesaurus to ensure broad coverage of biomedical sub-disciplines. For each seed topic, GPT-5.2 is prompted to generate an open-ended, complex research question that requires consulting multiple primary sources and synthesizing evidence across studies—for example, comparing treatment efficacy across populations, summarizing mechanistic controversies, or identifying gaps in the current literature.

Multi-Model Rollout. To avoid the systematic biases inherent in single-model data construction, we employ three heterogeneous frontier models—GPT-5.2, Gemini 2.5 Pro, and Qwen3-235B-A22B—each equipped with the same MCP tool suite. Every model independently performs two complete rollouts per question, yielding six trajectories that include the full sequence of tool invocations, intermediate reasoning, and a final synthesized answer.

Cross-Model Rubric Construction. A multi-agent deliberation is then conducted among the same three models. Each model reviews all six rollout answers, and the models engage in a structured discussion to (i) identify the factual claims and evidence threads that should be included in a reference answer, and (ii) construct a rubric consisting of weighted content criteria derived from these claims. This cross-model consensus procedure mitigates the risk that the rubric reflects the idiosyncratic strengths or blind spots of any single model.

Format Rubrics. In addition to the content-based rubric items, the evaluation protocol includes four format rubrics that assess process quality: (1) *tool utilization breadth*, requiring the model to make substantive use of the available search tools rather than relying on parametric knowledge alone; (2) *citation presence*, requiring the final answer to include specific literature references; (3) *citation format compliance*, requiring citations to follow a prescribed format (e.g., PMID-based references); and (4) *citation faithfulness*, penalizing fabricated or hallucinated references that do not correspond to real PubMed articles. These format criteria ensure that high scores reflect not only factual correctness but also methodological rigor in evidence retrieval and attribution.

H Representative Error Cases by Subtype

The following cases are drawn from real model trajectories on CureBench. Each illustrates one of the four error subtypes identified by the algorithm in Appendix A. Gap trajectory values are per-turn tendency gaps: $\text{gap}[j] = \text{score}(t_j, o_{\text{wrong}}) - \text{score}(t_j, o_{\text{correct}})$; negative values indicate the model leaning toward the correct answer.

MISLED-BY-EVIDENCE. *Question*: "Which requirement is part of the THALOMID REMS program to ensure safe prescribing and dispensing?" Correct: A (*Prescribers must be certified with the THALOMID REMS program*). Model chose: C (*Dispensing by any licensed community pharmacist*). Turns: 10. LCT = T9, MST = T4 ($\delta = +3.0$). Gap trajectory:

$$[+2.0, +2.0, +2.0, -2.0, +1.0, \\ -4.0, -1.0, -3.0, -3.0, -1.0]$$

The model correctly shifted toward option A at T3 (gap = -2.0) after retrieving relevant REMS information. However, the T3 FDA search returned an error (`No FDA data found for drug: Thalidomide THALOMID REMS`), causing the model to discard its intermediate conclusion and restart reasoning at T4, erasing the correct lean. Despite recovering and favoring A in T5–T9, the model ultimately selected C. *Takeaway*: a tool failure caused the model to abandon a correct intermediate conclusion and re-open all options as equally plausible.

FINAL-REASONING-ERROR. *Question*: "What are the approved indications for Dexmedetomidine Injection and Dexmedetomidine in 5% Dextrose Injection?" Correct: C (*Sedation of intubated/mechanically ventilated adult ICU patients ≤24h and sedation of non-intubated adult patients prior/during procedures*). Model chose: A (*Sedation of pediatric patients undergoing diagnostic procedures and treatment of acute pain*). Turns: 4. Gap trajectory: $[-2.0, -2.7, -4.7, -5.7]$ (monotonically decreasing, consistently favoring C).

The model's answer text reads: *"The FDA-labeled indications . . . are sedation of initially intubated, mechanically ventilated adult ICU patients (continuous infusion, not to exceed 24 hours) and sedation of non-intubated adult patients prior to/during procedures"*—a verbatim description of option C. Yet the extracted answer letter is A. *Takeaway*: the reasoning trajectory is entirely correct; the error is a failure to map the described content to the correct option letter during answer extraction.

EVIDENCE-IGNORED. *Question*: "Which drug was shown to block the tumorigenic effect of albuterol sulfate in Sprague-Dawley rats?" Correct: A (*Propranolol*). Model chose: C (*Atenolol*). Turns: 2. Gap trajectory: $[0.0, 0.0]$. The keyword *propranolol* appeared in both tool outputs: T0 returned a PubMed result contrasting propranolol (non-selective) with atenolol and metoprolol (selective β_1-blockers); T1 returned a Sprague-Dawley rat study involving propranolol. The model dismissed both as "general beta-blocker pharmacology" and continued searching for a more specific study, never connecting the retrieved evidence to the question. *Takeaway*: an overly strict relevance threshold caused the model to discard evidence that was in fact sufficient to answer the question.

INSUFFICIENT-SEARCH. *Question*: "For which type of transplant is Everolimus indicated, with the specification that it should be administered no earlier than 30 days posttransplant?" Correct: A (*Liver transplant in adult patients*). Model chose: D (*Kidney transplant in patients at high immunologic risk*). Turns: 2. Gap trajectory: $[+3.0, +3.0]$. The model confused everolimus with sirolimus from the first turn and issued 584 total queries (only 9 unique), all variants of "sirolimus extended-release capsules indication renal transplant". The correct answer required searching for "everolimus liver transplant", but no such query was ever issued. All retrieved evidence pointed to renal transplant indications for sirolimus, reinforcing the wrong answer throughout. *Takeaway*: an early drug-name confusion led to a misdirected query loop; the correct answer's key concept (*liver*) was never queried.

I Tool Suite Description

Our agent interacts with the environment through a suite of MCP (Model Context Protocol) tools. These tools are shared across all experimental conditions (base model, Skill, SFT, DPO variants, and CDPO) to ensure fair comparison. Table 10 summarizes the tool availability per benchmark, and the following paragraphs describe each tool.

General-Purpose Tools. Two tools are shared across all three benchmarks. **browse_webpage** fetches the content of a given URL and returns it in Markdown format, serving as a general-purpose reader for FDA drug label pages, PubMed articles,

and clinical trial registries. **google_search** performs web search via the Serper API and returns a ranked list of results (title, snippet, URL), primarily used as a fallback when specialized tools cannot satisfy the information need.

CureBench Tools. In addition to the general-purpose tools, CureBench uses **fda_drug_search**, which queries the FDA drug label database given a drug name and an aspect of interest (e.g., adverse reactions, indications, dosage, contraindications, drug interactions, pregnancy safety). It leverages LLM-assisted extraction to return the relevant section from the drug's prescribing information.

MedBrowseComp Tools. MedBrowseComp employs five specialized tools. **medbrowsecomp_search** is a unified medical information search tool with automatic routing: it dispatches NCT identifier queries to the clinical trial API and drug ingredient queries to the appropriate Orange Book endpoint based on the `reason` parameter (patent, approval, or exclusivity). **get_trial_info** queries ClinicalTrials.gov by NCT number and returns sponsor, recruitment status, intervention ingredients, and source links. **get_drug_patents** queries the FDA Orange Book for patent numbers, jurisdictions, and expiration dates of a given drug. **get_drug_approvals** returns product name, approval date, marketing authorization holder, and approval status (restricted to post-2000 records, sorted by date). **get_drug_exclusivities** returns exclusivity type (e.g., NCE, orphan drug, pediatric), start date, and end date (sorted by expiration).

PubMed Search Tools. The PubMed Search benchmark uses **pubmed_search**, which queries the PubMed database with medical or scientific keywords (recommended 3–6 terms) and returns matching articles with title, authors, abstract, PMID, publication year, and journal name. Pagination is supported via `limit` and `offset` parameters. This tool provides peer-reviewed biomedical literature as the primary evidence source for open-ended question answering.

Table 10: Tool availability across benchmarks. ✓ indicates the tool is available for the corresponding benchmark.

Tool	Cure	MedBrowse	PubMed
fda_drug_search	✓		
pubmed_search			✓
medbrowsecomp_search		✓	
get_trial_info		✓	
get_drug_patents		✓	
get_drug_approvals		✓	
get_drug_exclusivities		✓	
browse_webpage	✓	✓	✓
google_search	✓	✓	✓