The Physics of Multi-Turn Long-Horizon Planning: From Pre-training to Post-training via Single- and Multi-Teacher On-Policy Agentic Distillation

Tianyi Men[1,2], **Zhuoran Jin**[1,2], **Kang Liu**[1,2], **and Jun Zhao**[1,2,†]

[1]The Key Laboratory of Cognition and Decision Intelligence for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, China

[2]School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China

{tianyi.men, zhuoran.jin, kliu, jzhao}@nlpr.ia.ac.cn

Multi-turn long-horizon planning is a critical capability for foundation model agents, yet how to fundamentally improve it across different training stages remains unclear. Existing models are trained on uncontrollable and opaque Internet data, making it difficult to identify how planning ability is acquired, shaped, and integrated. To address this challenge, we introduce a unified and controlled multi-turn environment that enables precise control over task length, data quality, planning knowledge, and planning patterns, and systematically study long-horizon planning across three stages. **(1) Planning ability acquisition during pre-training.** We study data format, distribution, and quality. Explicit world model construction through chain-of-thought state transition modeling yields stronger long-horizon generalization than direct action prediction. Atomic skills alone are insufficient for compositional generalization, whereas a small amount of long-horizon data substantially improves planning ability. Moreover, suboptimal trajectories severely impair performance because decision errors accumulate and amplify over long horizons. **(2) Planning ability shaping via GRPO and OPD post-training.** From the perspective of mutual information, we distinguish general planning patterns from task-specific planning knowledge. For planning patterns, we identify three applicability regions of post-training: unnecessary, effective, and unsupported. Compared with group relative policy optimization (GRPO), on-policy distillation (OPD) has a broader effective region under low-quality pre-training data and long planning horizons, as it provides more consistent update directions than sparse credit assignment when the teacher is ideal. For planning knowledge, however, distilling unseen multi-path procedures from a teacher with different underlying knowledge may impair the student's existing world modeling without fully establishing the new knowledge. **(3) Planning ability integration through MOPD post-training.** We show that multi-teacher on-policy distillation (MOPD) integrates capabilities by converging to shared planning-pattern distributions across teachers. Compatible patterns enable cross-environment generalization, partially shared patterns support continual learning, while completely conflicting patterns cause severe catastrophic forgetting and cross-environment interference. Overall, our study provides a unified understanding of how long-horizon planning ability is acquired, shaped, and integrated across training stages, offering practical insights for developing stronger agentic foundation models.

Keywords: Multi-Turn, Long-Horizon, Agentic, Planning, Pre-Training, RL, GRPO, OPD, MOPD

Homepage Link: https://quester-one.github.io/PlanPhysWebsite/

Github Link: https://github.com/Quester-one/PlanPhysCode

Huggingface Model: https://huggingface.co/MultimodalAgent/TianyiMen_PlanPhys_Models

Huggingface Dataset: https://huggingface.co/datasets/MultimodalAgent/TianyiMen_PlanPhys_Datasets

[†]Corresponding author.

1. Introduction

As foundation models advance toward agentic systems, **multi-turn long-horizon planning** ability becomes a critical capability for building real-world general intelligent agents (Yuan et al., 2026, Zhu et al., 2026a, Li et al., 2026f, Zhang et al., 2026b). Different from single-turn question answering or short-sequence decision making, long-horizon planning requires agents to decompose complex tasks, track environment states, model state transitions, and compose atomic skills through long-horizon interactions (Li et al., 2026b, Cao et al., 2025). Recently, benchmarks for multi-turn long-horizon planning, including OSWorld 2.0 (Yuan et al., 2026), EdgeBench (Zhu et al., 2026a), Long-Horizon-Terminal-Bench (Li et al., 2026f), and DeepPlanning (Zhang et al., 2026b), demonstrate that while foundation models achieve basic short-horizon planning capabilities, they still struggle significantly with long-horizon planning tasks.

Currently, improving the long-horizon planning ability of agents mainly relies on three stages: (1) **Large-scale pre-training** to learn general planning knowledge and planning patterns; (2) **RL-based post-training** to further optimize long-horizon planning strategies; and (3) **Multi-teacher model consolidation** in post-training to combine long-horizon planning abilities from multiple teachers. Despite these efforts, one core question remains unclear:

A Core Question for Multi-Turn Long-Horizon Planning.
*How to **fundamentally** improve **long-horizon planning** ability of foundation models **at different training stages**?*

However, understanding and improving the long-horizon planning ability of foundation models still faces a fundamental challenge. The gap is that existing foundation models are trained on **uncontrollable and opaque Internet data**, making it difficult to systematically analyze where planning ability comes from and how it can be further strengthened. For example, it remains unclear whether the long-horizon planning ability exhibited by models already exists during pre-training, or it is a new ability activated during post-training. Therefore, when we seek to further strengthen long-horizon planning, as shown in Figure 1, we cannot answer the following three basic sub-questions from the perspective of the training stages:

Three Basic Sub-questions for Multi-Turn Long-Horizon Planning.
(1) Planning Ability Acquisition during Pre-training. *How does the model acquire long-horizon planning through pre-training, via world model internalization, atomic skill composition, and trajectory quality?*
(2) Planning Ability Shaping via GRPO and OPD Post-training. *How does RL shape long-horizon planning, and what are the applicable boundaries of RL algorithms in shaping planning knowledge and patterns?*
(3) Planning Ability Integration through MOPD Post-training. *How does model consolidation integrate long-horizon planning abilities to enable cross-environment generalization, continual learning, and conflict resolution?*

To answer above questions, we propose **a unified and controlled multi-turn environment to study long-horizon planning ability (Section 3)**. Unlike existing physics frameworks that are mainly based on single-turn mathematical reasoning (Ye et al., 2025, Zhang et al., 2025a), our environment supports multi-turn long-horizon planning, as shown in Figure 2. It enables precise control over environment numbers, levels and categories, task lengths, data quality, planning knowledge, and planning patterns. Moreover, it is compatible with both SFT and RL paradigms. Based on this framework, we draw the following conclusions through three progressive stages:

Planning ability acquisition during pre-training (Section 4). At this stage, we analyze the acquisition of long-horizon planning ability from **three perspectives: data format, data distribution, and data quality**. (i) Data format: Models that explicitly construct an internal world model through chain-of-thought for state transition modeling exhibit stronger generalization ability in long-horizon planning than direct action prediction. (ii) Data distribution: Although models struggle to achieve compositional generalization only through atomic skills, incorporating a small amount of long-horizon data during pre-training can enhance long-horizon planning ability. (iii) Data quality: Real-world pre-training data inevitably contains suboptimal trajectories alongside optimal ones. Since errors in long-sequence decision-making accumulate and amplify over time, such imperfect data severely impair long-horizon planning ability.

Planning ability shaping via GRPO and OPD post-training (Section 5). At this stage, we analyze planning ability through **the lens of mutual information and divide it into two aspects: planning patterns and planning knowledge**. The key difference is that planning patterns capture general planning behaviors shared across tasks, while planning knowledge contains task-specific procedures and solutions. (i) Planning patterns: Along the two dimensions of planning horizon and data quality mentioned in pre-training, the application space can be divided into three regions based on

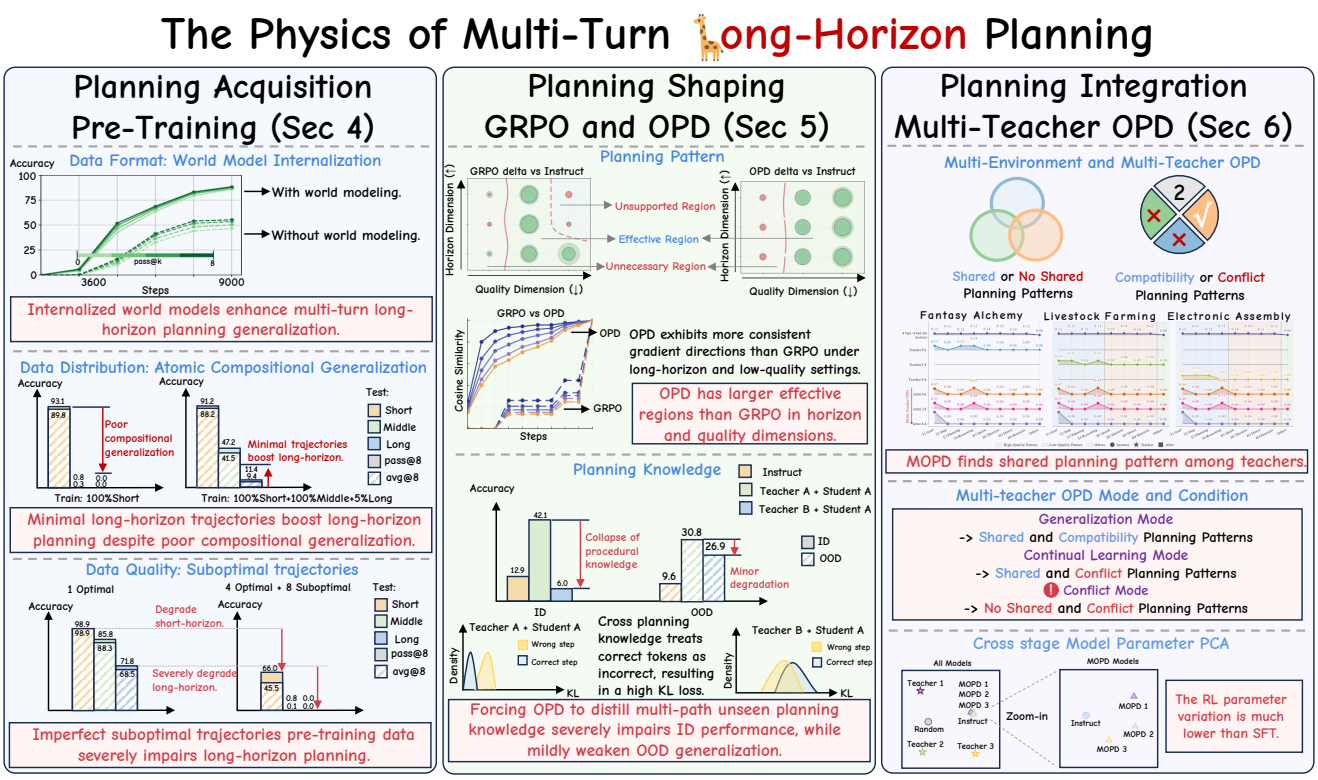


Figure 1: An overview of the studying into physics of multi-turn long-horizon planning. It studies the long-horizon planning ability across three training stages: **Large-scale pre-training**, **RL-based post-training (OPD and GRPO)**, and **Multi-teacher model consolidation post-training (MOPD)**. The giraffe icon is used to represent the "L" in "Long". Its long neck also reflects that the agents need to look far ahead in long-horizon planning.

the applicability boundaries of RL: *unnecessary region, effective region, unsupported region*. We compare mainstream post-training approaches including group relative policy optimization (GRPO) and on-policy distillation (OPD). GRPO has a narrower effective region. When pre-training data quality is low and the task requires long-horizon planning, OPD exhibits a broader effective boundary for shaping general planning patterns. This advantage stems from the fact that sparse credit assignment in RL introduces both correct and incorrect gradient update directions, whereas OPD provides more consistent and reliable update directions when the teacher is ideal. (ii) Planning knowledge: The situation is different. When the teacher and student models have different underlying procedural planning knowledge, directly applying OPD to distill unseen multi-path knowledge cannot guarantee effective transfer, even if the teacher achieves high accuracy. Instead, it may impair the student's existing in-domain world modeling and weaken its out-of-domain planning ability. This is because specific planning knowledge contains higher mutual information and requires precise, case-level updates. However, RL provides only small parameter updates, which are insufficient to fully replace old knowledge with new unseen knowledge. Therefore, the model may enter an intermediate state where prior knowledge is impaired while new planning knowledge remains not fully established.

Planning ability integration through MOPD post-training (Section 6). At this stage, we analyze **cross-environment generalization, continual learning, and cross-environment conflicts from two aspects: whether different planning patterns are shared or compatible across different environments**. The experiments show that the key mechanism of multi-teacher on-policy distillation (MOPD) for cascaded consolidation is to find and converge to the shared distribution among multiple teachers. (i) Generalization mode: When different environments share compatible planning patterns, MOPD can break domain boundaries and successfully transfer abilities activated in one environment to unseen environments. (ii) Continual learning mode: When planning patterns conflict across environments, as long as some shared and useful pattern structures remain across environments, the model can adapt while mitigating severe forgetting, thereby supporting continual learning. (iii) Conflict mode: However, when different environments share no common planning patterns and exhibit completely conflict behavior, the student model may suffer from severe catastrophic forgetting as it overfits to the new teacher, leading to interference of cross-environment planning abilities.

2. Preliminaries

2.1. Agent Planning

Planning Definition. In this work, we formulate agent planning as a sequential decision-making problem. Given an initial environmental state $s_0 \in \mathcal{S}$ and a specific target goal $g \in \mathcal{G}$, the objective of the agent is to generate a sequence of executable actions $A = \{a_1, a_2, \ldots, a_T\}$. The execution of this sequence transitions the environment from the initial state s_0 to a final state s_T that successfully satisfies the goal g.

Agent Skill/Experience/Procedural Knowledge/Script Knowledge. We define an agent skill (or experience or procedural knowledge or script knowledge) as a natural language mapping from a task description to a sequential list of sub-steps. Formally, a skill k in the skill library $\mathcal{K}$ is represented as a tuple $k = (d, P)$. Here, d represents the natural language description of the target task (e.g., "make a cup of coffee"), and $P = \langle p_1, p_2, \ldots, p_m \rangle$ is an ordered sequence of sub-steps, where each step p_i is also explicitly described in natural language (e.g., "find a mug", "pour water"). This format provides the agent with a clear recipe detailing exactly how a specific sub-goal should be systematically achieved. In practice, these skills can be utilized in two primary paradigms: either explicitly provided as external context, or implicitly internalized into the model's chain-of-thought (CoT) reasoning process. In this paper, we primarily focus on investigating the latter approach.

World Model. The world model aims to learn the rules of state transitions. Specifically, it models the state transition function $T(s_t, a_t) \to s_{t+1}$, which determines the next state given the current state and a specific action. In this process, procedural skills are not treated as separate, external modules; instead, they are directly internalized into the state transition function itself. By doing so, the model embeds these skills as the underlying rules driving state changes, enabling it to predict future states based on given actions.

2.2. On-Policy Agentic Distillation

On-Policy Agentic Distillation (OPAD) optimizes the student policy π_θ directly on its self-explored interactive trajectories. Formally, we define a K-turn trajectory $\tau = (g, s_0, \hat{Y}_1, s_1, \ldots, \hat{Y}_K, s_K) \sim \pi_\theta$, where $g \in \mathcal{G}$ and $s_0 \in \mathcal{S}$ are the goal and initial state. At each turn k, conditioned on the accumulated context H_k, the student generates a unified sequence $\hat{Y}_k = (\hat{y}_{k,1}, \ldots, \hat{y}_{k,T_k})$ that sequentially interleaves intermediate planning with an executable action, subsequently transitioning the environment to s_k.

At each token $t \in \{1, \ldots, T_k\}$ within turn k, both the student and a teacher policy π_{teacher} evaluate the sequence prefix to yield next-token distributions:

$$p_{k,t} \triangleq \pi_\theta(\cdot \mid H_k, \hat{Y}_{k,<t}) \quad \text{and} \quad q_{k,t} \triangleq \pi_{\text{teacher}}(\cdot \mid H_k, \hat{Y}_{k,<t}). \tag{1}$$

By autoregressively factorizing the reverse Kullback-Leibler (KL) divergence over the entire multi-turn rollout, the Agent-OPD objective admits an exact token-level decomposition. This provides dense, step-by-step supervision on both the reasoning chain and the decision-making process:

$$\mathcal{L}_{\text{Agent-OPD}}(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{k=1}^{K} \sum_{t=1}^{T_k} D_{\text{KL}}(p_{k,t} \parallel q_{k,t}) \right]. \tag{2}$$

PG-Style OPD vs GKD-Style OPD. On-Policy Agentic Distillation primarily diverges into two paradigms based on how teacher feedback is utilized. Policy Gradient (PG) style OPD treats the per-token reverse KL divergence as a dense reward signal. Specifically, it computes the token-level advantage using a single-sample log-ratio approximation: $A_{k,t} \approx \log q_{k,t}(\hat{y}_{k,t}) - \log p_{k,t}(\hat{y}_{k,t})$. The student is then updated via RL objectives to re-weight trajectory sampling probabilities: $\mathcal{L}_{\text{PG}} = \mathbb{E}_{\tau \sim \pi_\theta}[-\sum_{k,t} \text{sg}(A_{k,t}) \log p_{k,t}(\hat{y}_{k,t})]$, where $\text{sg}(\cdot)$ denotes the stop-gradient operator. In contrast, Generalized Knowledge Distillation (GKD) style OPD does not rely on sampled scalar advantages. Instead, it explicitly minimizes the statistical divergence between the full predictive distributions: $\mathcal{L}_{\text{GKD}} = \mathbb{E}_{\tau \sim \pi_\theta}[\sum_{k,t} D_{\text{KL}}(p_{k,t} \parallel q_{k,t})]$. Consequently, while PG-style indirectly optimizes the policy by increasing the likelihood of trajectories with lower reverse KL, GKD-style provides explicit supervision to directly reshape the token-level distributions. Crucially, GKD-style ignores the global trajectory-sampling probability. Instead, it leverages direct gradient propagation to minimize the local divergence between the student and teacher distributions exclusively on the already visited prefixes. In practice, Sampled-Token OPD predominantly adopts the PG and GKD paradigm, whereas Top-k and Full-Vocabulary OPD generally employ the GKD approach.

Sampled-Token OPD. Sampled-Token OPD evaluates the objective exclusively on the discrete tokens $\hat{y}_{k,t}$ sampled during the student's rollout, thereby avoiding the computation of the full vocabulary distribution. This approach is typically implemented in two ways. The **PG-Style** formulation utilizes a policy gradient estimator, treating the log-ratio as a token-level reward: $r_{k,t} = \log q_{k,t}(\hat{y}_{k,t}) - \log p_{k,t}(\hat{y}_{k,t})$. Alternatively, the **GKD-Style** formulation directly minimizes a local distance metric on the sampled tokens. Two common mathematical forms for the GKD-Style objective are based on Schulman's approximations: the **K1 approximation**, formulated as $\log p_{k,t}(\hat{y}_{k,t}) - \log q_{k,t}(\hat{y}_{k,t})$, and the **K2 approximation**, formulated as $\frac{1}{2}(\log p_{k,t}(\hat{y}_{k,t}) - \log q_{k,t}(\hat{y}_{k,t}))^2$.

Top-k OPD. Top-k OPD limits the divergence penalty to a truncated vocabulary space. Specifically, at each decoding step t within turn k, we isolate the subset of k tokens that are most probable according to the student policy $p_{k,t}$. Let us denote this high-confidence token subset as $\mathcal{V}_{k,t} \subset \mathcal{V}$. We then construct truncated and re-normalized probability mass functions for both the student and the teacher, restricted strictly to $\mathcal{V}_{k,t}$:

$$\tilde{p}_{k,t}(v) = \frac{p_{k,t}(v)}{\sum_{w \in \mathcal{V}_{k,t}} p_{k,t}(w)}, \quad \tilde{q}_{k,t}(v) = \frac{q_{k,t}(v)}{\sum_{w \in \mathcal{V}_{k,t}} q_{k,t}(w)}, \quad \forall v \in \mathcal{V}_{k,t}. \tag{3}$$

The distillation objective is subsequently formulated by evaluating the local KL divergence exclusively over these adjusted distributions. Integrating this into our agent's multi-turn trajectory framework yields the following formulation:

$$\mathcal{L}_{\text{Top-}k}(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{k=1}^{K} \sum_{t=1}^{T_k} D_{\text{KL}}(\tilde{p}_{k,t} \parallel \tilde{q}_{k,t}) \right]. \tag{4}$$

Full-Vocabulary OPD. In contrast to sampled or truncated approximations, Full-Vocabulary OPD evaluates the divergence penalty across the entire vocabulary $\mathcal{V}$ without any masking. At each decoding step t within turn k, it directly aligns the complete predictive distribution of the student with that of the teacher. The distillation objective function computes the exact KL divergence over all possible tokens:

$$\mathcal{L}_{\text{Full-Vocab}}(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{k=1}^{K} \sum_{t=1}^{T_k} D_{\text{KL}}(p_{k,t} \parallel q_{k,t}) \right]. \tag{5}$$

3. Task Formulation

3.1. Controllable Planning Gym

Skill Graph Construction. As shown in Figure 2, we construct hierarchical skill graphs across three distinct domains (e.g., *Fantasy Alchemy, Livestock Farming, Electronic Assembly*). To ensure rich semantic relationships, we leverage a large language model to generate category trees and concrete item entities. Each domain is structured as a tree with a height of H layers, where each layer encompasses W categories, and each category contains N unique concrete items.

The abstract graph transitions systematically connect nodes from lower layers to higher layers. **Crucially, the synthesis rules governing these transitions are formulated as a logical combination of AND/OR operations over prerequisite items.** For a target node v, its valid synthesis pathways are defined by an **OR** relation over multiple independent recipes, while each individual recipe requires the simultaneous availability (**AND**) of its constituent ingredients. Formally, the synthesis logic for node v is expressed as:

$$v \iff (u_{1,1} \wedge \cdots \wedge u_{1,k_1}) \vee \cdots \vee (u_{m,1} \wedge \cdots \wedge u_{m,k_m}) \tag{6}$$

where each conjunction block $(u_{i,1} \wedge \cdots \wedge u_{i,k_i})$ represents a valid abstract recipe. To prevent high-step-count explosion, we employ a greedy minimization algorithm that prioritizes combining items with fewer preceding synthesis steps. For each domain, we generate disjoint abstract transition graphs (e.g., Graph A and Graph B). The transitions require varying numbers of prerequisite nodes governed by a fixed probability p, ensuring structurally sound and non-overlapping rule sets. Furthermore, to improve scalability, counterfactual synthesis is used for the state transition rules, since this is done from the pre-training stage, it does not affect the experimental conclusions.

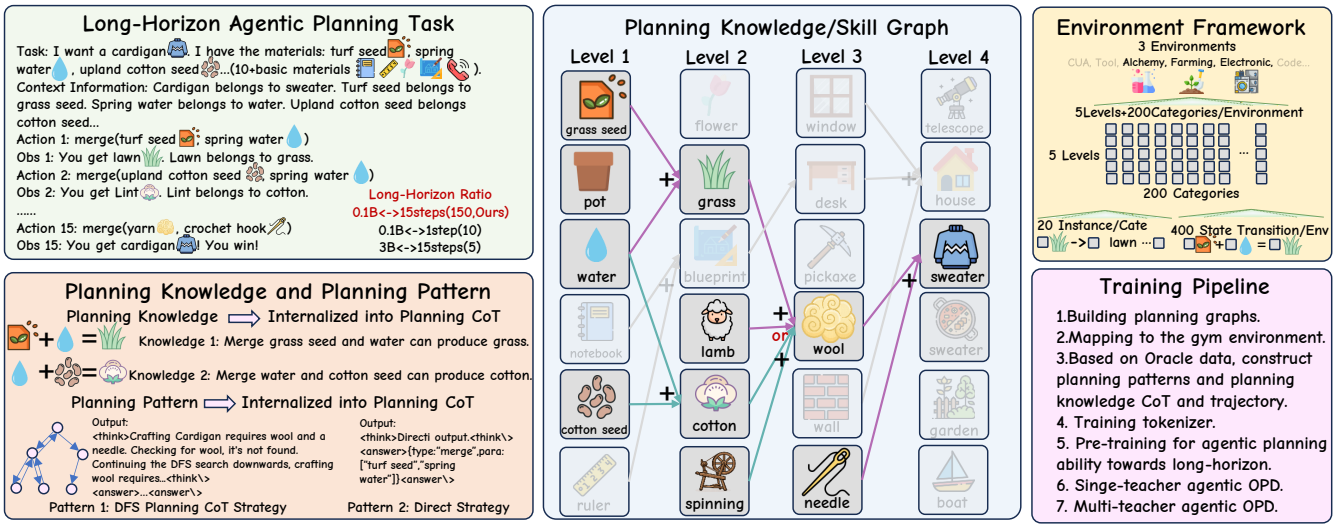


Figure 2: Overview of the proposed long-horizon agentic planning framework.

Graph-to-Gym Mapping. The abstract skill graphs are instantiated into concrete simulated environments or gyms by mapping abstract nodes to specific items via an *index map*. To foster environment diversity, we generate a combination of fixed and randomized configurations. Crucially, a subset of K item indices combination per category is reserved for test set configurations. The remaining indices are randomly sampled to construct diverse training environments, preventing data leakage and ensuring evaluation on unseen entity mappings.

Task Difficulty Control. Task difficulty is determined by the minimum number of actions (i.e., synthesis steps) required to produce a target item from a given initial inventory. By sampling the target item from higher layers and starting materials from lower layers, we dynamically compute the exact step count required. Based on this, tasks are categorized into three difficulty levels: *Short*, *Middle*, and *Long* step thresholds. To simulate realistic planning noise, we inject distractor items from the same starting layers into the initial inventory.

Dataset Splitting. To evaluate generalization, we split the instantiated configurations into datasets for pre-training, post-training, and testing. The test set exclusively employs novel combinations of familiar items. The training set is divided: the majority of configurations are utilized for pre-training to encourage exploratory learning across all trajectories, while a smaller subset is reserved for post-training, providing high-quality trajectories.

3.2. Task Setup

We formulate the planning problem as a sequential decision-making task. The agent is initialized with a starting inventory (prerequisites and distractors) and must synthesize a target item in limited steps. The state observation includes natural language descriptions of the category each item belongs to, facilitating category-level reasoning without exposing underlying IDs. At each step, the model must select the correct sequence of materials from the inventory to execute valid synthesis actions until the target item is successfully created.

3.3. Evaluation

For evaluation, we uniformly sample M tasks across different step counts to ensure a balanced assessment spanning the difficulty spectrum. The evaluation is conducted on the reserved test configurations of Graph A for each difficulty level (*Short*, *Middle*, and *Long*). The primary evaluation metric is the *final success rate*, defined as the agent's ability to successfully synthesize the target item by strictly adhering to the unseen rules. For each test task, we independently run the agent K times in a multi-turn setting. *Avg@*K is computed by averaging *Pass@1* over the K independent runs. *Pass@*K is the fraction of tasks solved in at least one of the K runs.

4. Multi-Turn Long-Horizon Planning Ability Acquisition during Pre-training

Current large language model agents remain weak at multi-turn long-horizon planning. We investigate this problem by progressively relaxing three assumptions about pre-training data. (1) **Data format: world-model internalization.**

Given correct expert trajectories, does explicitly modeling intermediate state transitions via CoT to improve planning beyond only action-sequence imitation? (2) **Data distribution: atomic compositional generalization.** When long-horizon trajectories are long-tail distribution, can models compose abundant atomic skills to generalize to compositional long-horizon planning tasks? (3) **Data quality: suboptimal trajectories.** When expert trajectories are mixed with redundant, suboptimal, or erroneous solutions, how does this heterogeneity affect the long-horizon planning ability? Throughout this section, pre-training data includes both pre-training and mid-training corpora, which are combined into a unified training mixture. Accordingly, this section studies **Planning with Internalized World Model**, **Atomic Skill Compositional Generalization**, and **Impact of Suboptimal Trajectories**.

4.1. Planning with Internalized World Model

World Modeling. Fundamentally, a world model represents the underlying dynamics of an environment, serving as an internal simulator of how the environment transitions between states over time. It captures the relationship between current states, actions, and future states. The world model is defined by a state transition function:

$$\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S} \tag{7}$$

where $\mathcal{S}$ denotes the state space and $\mathcal{A}$ denotes the action space. Given a state $s_t \in \mathcal{S}$ and an action $a_t \in \mathcal{A}$, the transition function predicts the next state $s_{t+1} = \mathcal{T}(s_t, a_t)$. This predictive capability allows an agent to simulate the future consequences of its decisions through internal reasoning, without interacting with the external environment.

Skill Internalized World Modeling. We model planning as depth-first search over an internal simulated state space $\mathcal{S}$. Let $\mathcal{K}$ denote the set of atomic skills, where each skill $k \in \mathcal{K}$ is defined as a state transition function $f_k : \mathcal{S} \to \mathcal{S}$. We define a planning step t as one internal DFS node expansion. At each step, the model selects an ordered skill sequence $\boldsymbol{k}_t = (k_{t,1}, \ldots, k_{t,m_t}) \in \mathcal{K}^*$ and applies the corresponding transition operators sequentially on the internal state:

$$s_{t+1} = f_{k_{t,m_t}} \circ \cdots \circ f_{k_{t,1}}(s_t), \tag{8}$$

which represents the cumulative effect of executing the selected skills within the internal world model.

The resulting transitions define a search tree over simulated states, and the model performs depth-first search by recursively expanding one branch and exploring alternative skill sequences until a valid plan is found. All transitions and search operations are conducted within the internal CoT rather than through external environment interaction.

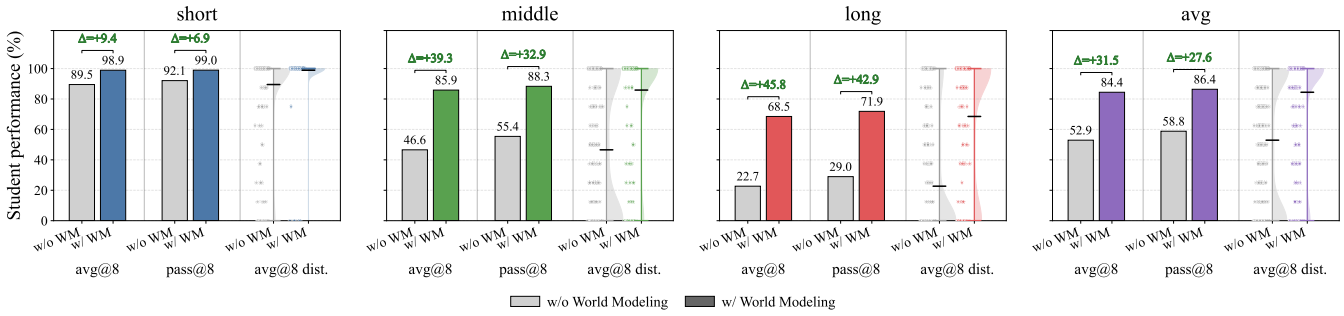


Figure 3: Final performance comparison between students with and without world modeling across three difficulty levels. For 3 domain and 3 difficulty level, each test subset contains 160 instances, resulting in 1,440 test instances for each checkpoint. During inference, we set the sampling temperature to 0.4 and independently sample 8 trajectories for each instance, with a limitation of 20 steps per trajectory. Therefore, evaluating one checkpoint requires a most total of 11,520 inference runs. We report $\mathrm{avg@8}$ as the average success rate across the 8 sampled trajectories, and $\mathrm{pass@8}$ as the proportion of instances for which at least one of the 8 trajectories successfully completes the task. Unless otherwise specified, we use the same evaluation protocol in all subsequent experiments.

Task Setup. (1) **Model configuration.** Since the pre-training corpora of existing models are proprietary and inaccessible, we instead construct a controlled experimental setting by randomly initializing a model under the Qwen2.5 architecture. To ensure consistent tokenization in the synthetic pre-training corpus, we train a byte-level BPE tokenizer from scratch. The full model configuration is detailed in Appendix A.1. (2) **Agent planning task configuration.** The corpus covers 3 domains. Each domain contains 5 levels, with 40 item categories per level and 20 instances per

category. These item categories have composition rules, meaning an item can be made from other items using AND and OR logic. The task requires the model to create a target item from basic materials, while extra noise items are added to the mix. You can find the details in Appendix A.2. (3) **Agent planning corpus configuration.** We compare the performance of direct answering and world modeling planning. For direct answering, the model predicts the action for each step directly. For world modeling planning, the model must first output a state transition function grounded in planning knowledge before acting. Specifically, it builds a chain of thought using post order traversal, state transition rules for item composition, and grounding from categories to exact instances. We train 1.2B corpus for the student model. You can find the exact corpus setup in Appendix A.4. And training details you can find in Appendix A.3.

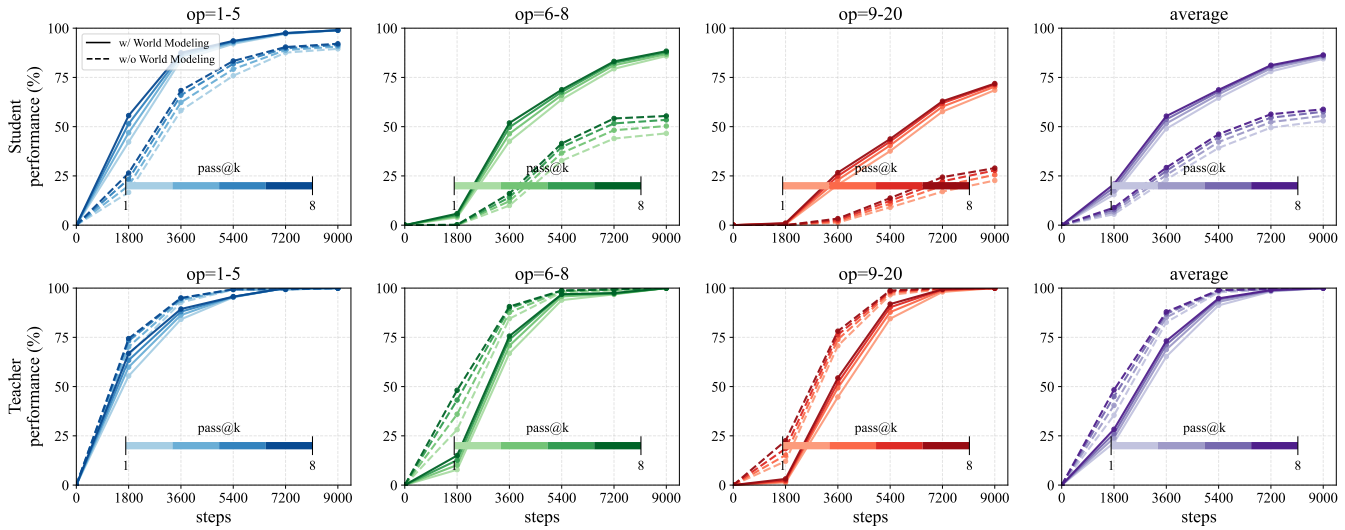


Figure 4: Performance scaling with pre-training corpus size for agent planning, comparing students with and without world modeling across three difficulty levels. The teacher is demonstrating the performance of the post-training set with the same distribution.

Results. (1) **Internalizing a world model leads to stronger generalization ability.** As shown in Figure 3, the student model equipped with a world model (w/ WM) consistently outperforms the direct answering baseline (w/o WM) across all difficulty levels. Specifically, the model shows improvements in the short-horizon setting of +9.4% (from 89.5% to 98.9%) and +6.9% (from 92.1% to 99.0%) for avg@8 and pass@8, respectively. Similar enhancements are observed in the middle ($\Delta = +39.3\%$ for avg@8, $\Delta = +32.9\%$ for pass@8) and high ($\Delta = +45.8\%$ for avg@8, $\Delta = +42.9\%$ for pass@8) long-horizon planning, proving that planning with an internal world model improves the agent's ability to generalize to new and complex planning tasks. (2) **Direct answering achieves higher efficiency with a lower token budget, but suffers from a lower performance upper bound.** As shown in Figure 4, analyzing the performance scaling curves over training steps reveals a distinct trade-off. In terms of learning speed, direct prediction (w/o WM, dashed lines) exhibits a faster initial trajectory in the teacher performance curves, where the dashed lines rise more rapidly and reach high performance at earlier stages (e.g., between 1800 and 5400 steps) compared to the solid lines (w/ WM). However, direct answering is limited by a strictly lower performance ceiling. As demonstrated by the student performance curves across all operation ranges (op=1-5, op=6-8, op=9-20) and their average, the solid lines (w/ WM) eventually surpass the dashed lines and stabilize at a significantly higher final accuracy. This demonstrates that using direct answering leads to faster convergence in training memory capabilities and requires a lower token budget, but it has much poorer generalization ability.

Takeaway

Internalized world models enhance multi-turn long-horizon planning generalization. While direct answering converges faster, requires a lower token budget, and is more efficient for tasks relying on memorized answers, it generalizes poorly to multi-turn long-horizon planning. In contrast, planning with a world model achieves stronger generalization and higher final accuracy on multi-turn long-horizon planning tasks.

4.2. Atomic Skill Compositional Generalization

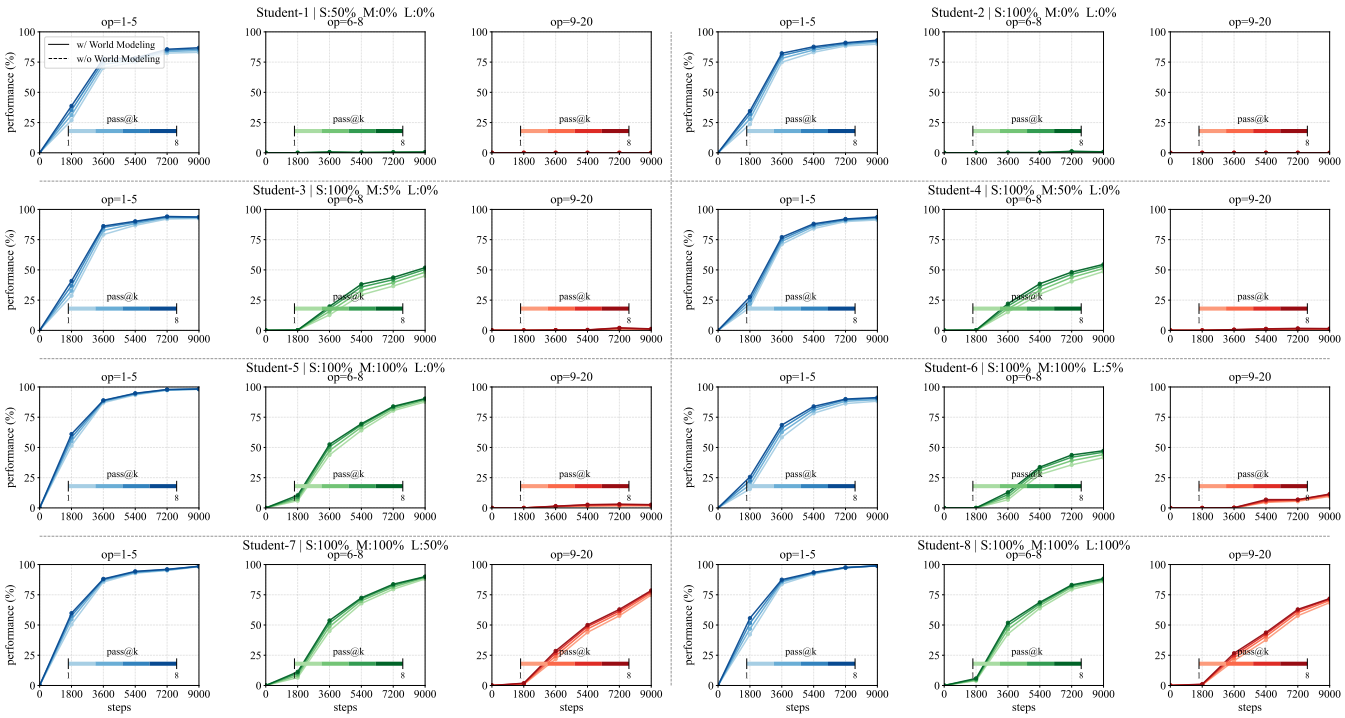


Figure 5: Performance scaling across different pretraining distributions.

Atomic Skill Composition. Atomic skill compositional generalization refers to the ability of an agent to combine known **atomic skills** into new **compositional skills** for solving unseen complex tasks (Yuan et al., 2025). In this work, such compositional ability is acquired through pre-training and encoded in the model parameters, rather than elicited through test-time demonstrations, prompt engineering, or in-context learning. In our environment, an atomic skill is the ability to complete a single valid synthesis step. We represent such a step as a basic function $f(x)$ that produces a new item from a set of input materials. Suppose the model learns two recipes, f and g, from the training corpus. Compositional generalization occurs when, without having observed their complete composition during training, the model can correctly execute the novel nested pathway

$$h(x) = g(f(x)) \tag{9}$$

at test time. In our hierarchical graph tasks, this requires the model to synthesize a target node at a higher level by chaining independently learned lower-level recipes. This setting therefore evaluates whether pre-training enables the model to internalize reusable synthesis rules and systematically compose them, rather than merely memorizing complete trajectories or relying on test-time prompting.

Task Setup. (1) **Difficulty definition.** We divide the tasks into short, middle, and long levels based on the number of operation steps. Specifically, the short-horizon requires 1 to 5 steps, the middle-horizon requires 6 to 8 steps, and the long-horizon requires 9 or more steps. (2) **Data allocation.** We organize the training data into three difficulty intervals. The long-horizon interval includes all data from the short-horizon intervals. The short-horizon intervals do not contain complete tasks from the higher level, but only partial task components. Within each interval, we set different data ratios including 0%, 5%, 50%, and 100%. (3) **Experiment and evaluation.** All experiments use a unified number of training steps for 9000 steps. We then evaluate the models across the three horizon levels to test the compositional generalization ability of their internalized skills.

Results. As shown in Figure 5 and Table 1. (1) **Atomic skills struggle to compose automatically.** When the model is pretrained exclusively on short tasks (Region S at 100%), it achieves strong performance on short-horizon tasks (pass@8 of 93.12%), but accuracy on longer horizons drops to near zero (0.83% for middle and 0.00% for long-horizon

Table 1: Pretrain distribution table with two global blocks (top: avg@8, bottom: pass@8). Ratio is split into S(Short)/M(Middle)/L(Long) columns. Each difficulty includes domain columns and a mean column. Two extra columns report difficulty gaps: Gap(L-S) and Gap(M-S). Domain abbreviations: FA = Fantasy Alchemy; LF = Livestock Farming; EA = Electronic Assembly.

Region	Ratio			Short				Middle				Long				Gap	
	S	M	L	FA	LF	EA	Mean	FA	LF	EA	Mean	FA	LF	EA	Mean	L-S	M-S
avg@8																	
S	50%	0%	0%	90.62	74.84	83.75	83.07	0.23	0.00	0.55	0.26	0.00	0.00	0.00	0.00	-83.07	-82.81
	100%	0%	0%	95.55	80.70	93.28	89.84	0.62	0.16	0.39	0.39	0.00	0.00	0.00	0.00	-89.84	-89.45
S+M	100%	5%	0%	96.88	84.53	96.09	92.50	56.95	37.19	41.25	45.13	0.70	0.62	1.02	0.78	-91.72	-47.37
	100%	50%	0%	94.14	88.83	90.86	91.28	59.69	38.52	47.73	48.65	0.55	0.00	1.95	0.83	-90.44	-42.63
	100%	100%	0%	99.77	97.42	96.72	97.97	89.38	87.50	86.09	87.66	0.70	1.72	2.81	1.74	-96.22	-10.31
S+M+L	100%	100%	5%	93.52	80.62	90.62	88.26	58.98	31.02	34.77	41.59	13.52	3.20	11.72	9.48	-78.78	-46.67
	100%	100%	50%	99.92	96.56	98.12	98.20	88.12	85.55	90.55	88.07	78.05	72.50	73.98	74.84	-23.36	-10.13
	100%	100%	100%	100.00	99.22	97.50	98.91	91.80	82.19	83.59	85.86	69.61	65.55	70.39	68.52	-30.39	-13.05
pass@8																	
S	50%	0%	0%	92.50	81.25	86.88	86.88	1.25	0.00	0.62	0.62	0.00	0.00	0.00	0.00	-86.88	-86.25
	100%	0%	0%	97.50	86.25	95.62	93.12	0.62	0.62	1.25	0.83	0.00	0.00	0.00	0.00	-93.12	-92.29
S+M	100%	5%	0%	98.12	86.25	96.88	93.75	61.88	43.12	50.62	51.88	1.25	0.62	1.88	1.25	-92.50	-41.88
	100%	50%	0%	95.00	93.12	93.12	93.75	63.12	45.00	55.62	54.58	0.62	0.00	3.75	1.46	-92.29	-39.17
	100%	100%	0%	100.00	98.12	96.88	98.33	91.25	90.00	90.00	90.42	1.25	2.50	4.38	2.71	-95.62	-7.92
S+M+L	100%	100%	5%	95.62	85.00	93.12	91.25	63.12	35.00	43.75	47.29	16.88	3.75	13.75	11.46	-79.79	-43.96
	100%	100%	50%	100.00	97.50	98.12	98.54	89.38	88.75	91.88	90.00	81.88	76.25	77.50	78.54	-20.00	-8.54
	100%	100%	100%	100.00	99.38	97.50	98.96	93.75	85.00	86.25	88.33	71.88	71.25	72.50	71.88	-27.08	-10.62

subsets). For multi-turn agents, simply knowing individual actions is insufficient. They fail to connect atomic skills into long-horizon planning ability without explicitly observing extended state transitions during training. (2) **Minimal exposure to long-horizon trajectories activate planning generalization.** Adding just a small fraction of long-horizon trajectories causes a sudden jump in performance on longer levels. For instance, introducing merely 5% of middle-horizon trajectories boosts middle pass@8 from 0.83% to 51.88%, while 5% of long-horizon trajectories increases long-horizon subset in pass@8 from 0.00% to 11.46%.

Takeaway

Models struggle with automatic compositional generalization, but minimal long-horizon trajectories activates long-horizon planning. Learning atomic skills exclusively from short tasks is insufficient for solving multi-turn long-horizon planning tasks. However, introducing just a small fraction of long-horizon trajectories teaches the agent how to connect these skills sequentially, bridging the performance gap on longer tasks.

4.3. Impact of Suboptimal Trajectories

Planning Pattern. We define the planning pattern as the structural and algorithmic skeleton of the CoT. It dictates the general thinking approach required to solve the task, independent of the specific planning knowledge involved. As demonstrated in Table 2, the planning pattern involves a series of steps, including identifying the target category, recalling relevant planning knowledge or skills, linking abstract categories with specific inventory items, and generating the final action. This template uses standardized textual structures with consistent phrasing to explicitly define the target objective, analyze the required prerequisites, and finally generate the corresponding action. However, pre-training corpora may also contain other planning patterns, such as direct-answer patterns that bypass explicit planning processes, or low-quality planning patterns that involve incomplete, noisy, or irrelevant planning information.

Planning Knowledge. In contrast, we define planning knowledge as task-specific procedural knowledge that specifies how actions interact, combine, and transform to produce valid outcomes. It provides the concrete rules that instantiate abstract planning patterns into executable plans, such as synthesis recipes or state-action transitions. As illustrated in Table 2, the planning knowledge encompasses the explicit synthesis rules defining exactly which materials combine to create a new item. This refers to the factual recipes within the skill graph, such as the specific knowledge that combining Crystalgrain and Glowfern synthesizes a Quartz Solution. To support scalable dataset construction, we introduce counterfactual synthesis rules to expand the skill graph with systematically generated yet logically consistent synthesis knowledge. As these rules are consistently adopted throughout both pre-training and all downstream experiments, the experimental settings remain aligned, and our conclusions are not affected by the use of counterfactual synthesis.

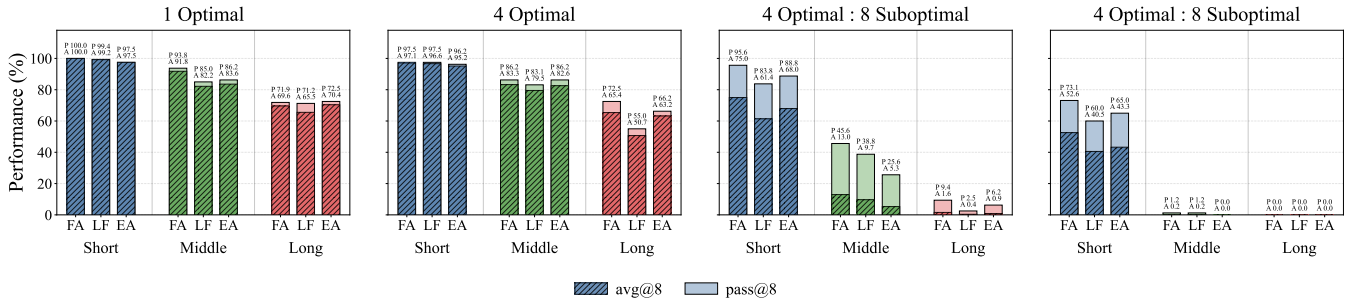


Figure 6: Effect of pretraining data composition on downstream performance. The figure compares avg@8 and pass@8 across three task domains and three difficulty levels under different pretraining configurations. FA, LF, and EA denote Fantasy Alchemy, Livestock Farming, and Electronic Assembly.

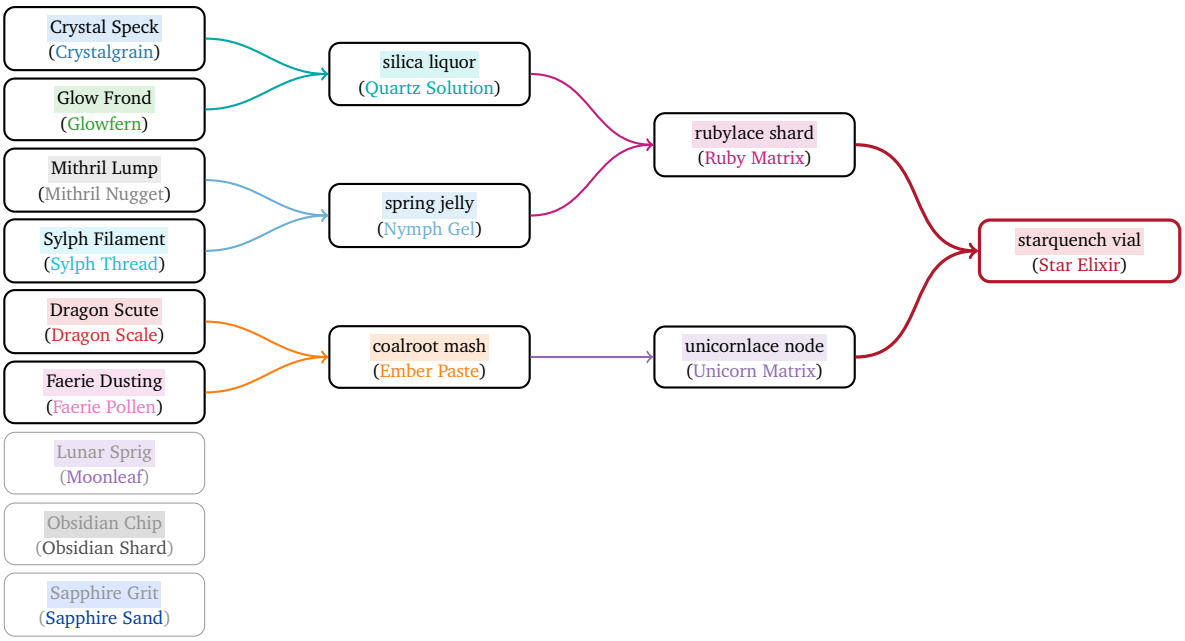
Task Setup. (1) **Planning pattern.** We primarily establish 8 planning patterns, which can be classified into two categories. The first thinking pattern features complex templates and optimal decisions. The second thinking pattern features direct and suboptimal decisions. Each template corresponds directly to whether it represents an optimal decision. (2) **Planning quality distribution.** We configure 8 quality distributions. For example, these distributions include 4 optimal modes, 4 to 4 ratio between optimal and suboptimal modes, and 4 to 8 ratio between optimal and suboptimal modes. In addition, all data utilize the same metadata except for different rendering, while the data scale and training steps remain consistent.

Results. As shown in Figure 6. (1) **Multiple optimal patterns maintain similar generalization.** Increasing the diversity of optimal reasoning templates from 1 Optimal to 4 Optimal does not significantly degrade or improve performance. The model maintains similar success rates across all three domains (FA, LF, EA). This indicates that agents can effectively internalize and utilize multiple high-quality structural skeletons without detrimental interference. (2) **Suboptimal trajectories cause catastrophic collapse due to compounding errors for long-horizon planning.** The introduction of suboptimal decisions (4 optimal : 8 suboptimal) severely undermines the model's long-horizon planning capabilities. While the model retains partial capability on short-horizon tasks, its performance plummets to near-zero on middle and long-horizon tasks. In sequential decision-making, a suboptimal step early in the trajectory does not merely cause a localized failure; rather, the error accumulates and amplifies step-by-step. While agents might tolerate minor deviations in short-horizon tasks, these cascading errors in extended plans steer the agent irrecoverably off course, preventing it from completing the objective within the required step limits.

Takeaway

Imperfect pre-training data severely degrades multi-turn long-horizon planning. In real-world environments, pre-training data naturally contains non-optimal trajectories instead of perfect expert demonstrations. These suboptimal paths introduce reasoning errors that accumulate heavily over long horizons. When trained on this mixed data, the agent fails to complete long tasks. Trying to overcome these learned flaws by simply increasing sampling is ineffective, as it fails to improve the success rate in long-horizon tasks. Furthermore, in multi-turn scenarios, relying on massive sampling causes unacceptable computational costs, making it impractical for real-world deployments.

Table 2: Planning pattern and planning knowledge for the first fantasy alchemy test sample. The state transition and synthesis rules of the Skill Graph represent the underlying dynamics of the environment, and they must be learned through interaction with the environment, as they are not provided as part of the context during multi-turn interactions.

Part	JSON Content
Input	<question> {"task": "Synthesize starquench vial", "target_category": "Star Elixir", "inventory": ["Crystal Speck", "Dragon Scute", "Faerie Dusting", "Glow Frond", "Lunar Sprig", "Mithril Lump", "Obsidian Chip", "Sapphire Grit", "Sylph Filament"], "inventory_category_description": ["Crystal Speck belongs to category Crystalgrain.", "Dragon Scute belongs to category Dragon Scale.", "Faerie Dusting belongs to category Faerie Pollen.", "Glow Frond belongs to category Glowfern.", "Lunar Sprig belongs to category Moonleaf.", "Mithril Lump belongs to category Mithril Nugget.", "Obsidian Chip belongs to category Obsidian Shard.", "Sapphire Grit belongs to category Sapphire Sand.", "Sylph Filament belongs to category Sylph Thread."], "history": []} </question>
Skill Graph	
Pattern 1 Output (Turn 1)	<solution>Goal: synthesize Star Elixir. Category synthesis rule: combining Ruby Matrix, Unicorn Matrix can produce Star Elixir. Ruby Matrix is currently not represented in inventory. Unicorn Matrix is currently not represented in inventory. Status: need to acquire Ruby Matrix, Unicorn Matrix first. Goal: synthesize Ruby Matrix. Category synthesis rule: combining Nymph Gel, Quartz Solution can produce Ruby Matrix. Nymph Gel is currently not represented in inventory. Quartz Solution is currently not represented in inventory. Status: need to acquire Nymph Gel, Quartz Solution first. Goal: synthesize Nymph Gel. Category synthesis rule: combining Mithril Nugget, Sylph Thread can produce Nymph Gel. Mithril Nugget is represented by items like Mithril Lump. Sylph Thread is represented by items like Sylph Filament. Action: use Mithril Lump (Mithril Nugget), Sylph Filament (Sylph Thread) to fulfill this category requirement.</solution> <answer>{"type": "process", "materials": ["Mithril Lump", "Sylph Filament"]}</answer>
Pattern 1 Obs (Turn 1)	You get spring jelly. It belongs to Nymph Gel. .....
Pattern 1 Output (Turn 15)	<solution>Goal: synthesize ... to fulfill this category requirement.</solution> <answer>{"type": "process", "materials": ["rubylace shard", "unicornlace node"]}</answer>
Final Obs	You get starquench vial!
Pattern 2 Output (Turn 1)	<solution>Direct selection from current inventory.</solution> <answer>{"type": "process", "materials": ["Lunar Sprig", "Obsidian Chip"]}</answer>
Pattern 2 Obs (Turn 1)	You get nothing. .....
Pattern 2 Output (Turn 20)	<solution>Direct selection from current inventory.</solution> <answer>{"type": "process", "materials": ["Sapphire Grit", "Dragon Scute"]}</answer>
Final Obs	The number of steps exceeds the maximum step budget.

5. Planning Ability Shaping via GRPO and On-Policy Agentic Distillation

5.1. Mutual Information for Planning Pattern and Planning Knowledge

Mutual Information. Inspired by Wang et al. (2026e), we use mutual information to measure the relevance between the input task (T) and the two distinct outputs: planning pattern (P) and planning knowledge (K). To reflect their different properties, we define their mutual information separately as:

$$I(T;P) = H(P) - H(P|T), \quad I(T;K) = H(K) - H(K|T) \tag{10}$$

where $I(T;P)$ and $I(T;K)$ quantify the reduction in uncertainty about P and K, respectively, when the task T is given. Due to the abstract and reusable characteristics of the planning pattern P, it is less dependent on the input task T and can be shared across different samples. Therefore, knowing T provides limited information about P, resulting in a lower mutual information $I(T;P)$. In contrast, the planning knowledge K contains task-specific information and is closely related to the input task T. Thus, knowing T can greatly reduce the uncertainty of K, leading to a higher mutual information $I(T;K)$.

To illustrate, general planning patterns P, such as reflection or backtracking, often span across diverse samples. Conversely, the planning knowledge K (e.g., specific skills and procedural knowledge) is highly specific and tightly bound to the current state of a specific sample. While different tasks may certainly necessitate different planning patterns, the input task T imposes far stricter constraints on the specific planning knowledge required than on the broad strategy used. This relative difference implies that knowing T reduces significantly more uncertainty about K than about P, establishing that $I(T;P) < I(T;K)$.

5.2. Impact of Planning Pattern

5.2.1. Accuracy Analysis

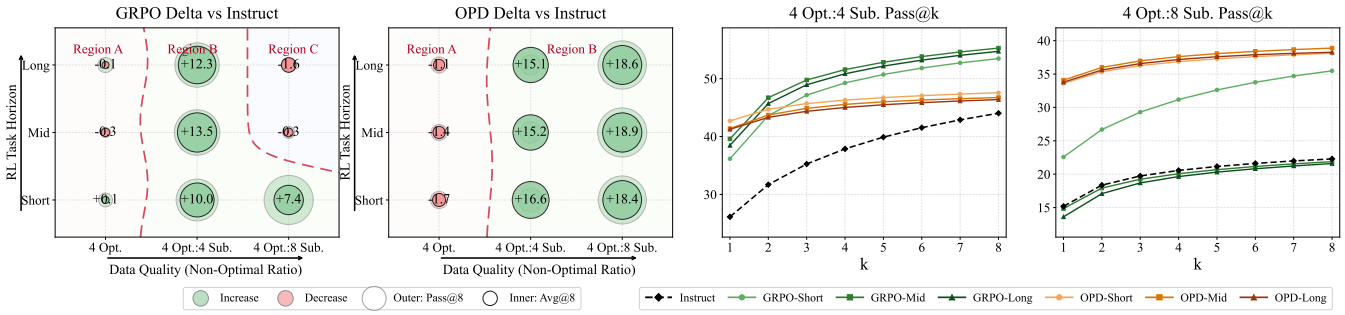


Figure 7: Applicability boundaries and post-training capability upper bounds of RL training. The first two panels show the applicability boundaries of multi-turn GRPO and OPD across RL task horizon and non-optimal action ratios (pre-training data quality). The last two panels compare the post-training capability upper bounds of different models.

RL Applicability Regions. We define three regions to characterize when RL can improve planning. The regions are determined by the performance gap among planning patterns and the ability of RL to discover better patterns.

- **Region A: Unnecessary Region.** Different planning patterns achieve similar performance in this region. RL-based selection is unnecessary because choosing different patterns leads to comparable results.
- **Region B: Effective Region.** Different planning patterns show clear performance differences in this region. RL can discover better patterns through optimization and achieve consistent performance improvements.
- **Region C: Unsupported Region.** Better planning patterns exist in this region, but discovering them depends on the RL algorithm design. An inappropriate RL algorithm may fail to activate the best planning pattern.

Training Setup. (1) **Base models.** As shown in Section 4.3, we use three pre-trained base models: 4 Opt, 4 Opt:4 Sub, and 4 Opt:8 Sub. (2) **RL data.** We use the post-training data from Table 11. To test cross-task generalization, we entirely use the Fantasy Alchemy subset as the post-training data. To test cross-horizon generalization, we use data from short, middle, and long horizons. We downsample 3000 data items during post-training, but training process may not use all of them limited in training steps and batch size. (3) **RL algorithms.** We use multi-turn GRPO and

multi-turn OPD algorithms. Multi-turn GRPO uses the binary final outcome reward. Multi-turn OPD uses Top-k OPD, where K is 100.

Table 3: Pattern-base comparison table. Rows are grouped by the pattern mixture: 4 Opt., 4 Opt.:4 Sub., and 4 Opt.:8 Sub. Each group contains the instruct checkpoint and multi-turn OPD/GRPO models trained on short, middle, and long levels. Two global blocks are reported (top: avg@8, bottom: pass@8). The last column reports the mean over short/mid/long evaluation difficulty means and its triangle delta vs. the instruct baseline in the same group. Domain abbreviations: FA = Fantasy Alchemy; LF = Livestock Farming; EA = Electronic Assembly.

Pattern Mix	Model	Short				Mid				Long				Avg
		FA	LF	EA	Mean	FA	LF	EA	Mean	FA	LF	EA	Mean	S/M/L Mean (△)
avg@8														
4 Opt.	Instruct	97.11	96.64	95.16	96.30	83.28	79.53	82.58	81.80	65.39	50.70	63.20	59.77	79.29 △+0.00
	Short OPD	97.27	94.77	95.55	95.86	81.95	76.64	78.52	79.04	63.12	48.98	61.88	57.99	77.63 △-1.66
	Short GRPO	97.03	96.88	95.16	96.35	83.28	79.14	82.89	81.77	65.47	50.86	63.75	60.03	79.38 △+0.10
	Mid OPD	97.81	95.47	94.92	96.07	81.33	78.83	78.67	79.61	64.61	48.44	61.09	58.05	77.91 △-1.38
	Mid GRPO	96.95	96.64	95.08	96.22	82.34	78.20	82.73	81.09	64.45	51.09	63.52	59.69	79.00 △-0.29
	Long OPD	96.88	95.55	95.31	95.91	81.17	78.83	82.66	80.89	61.88	50.08	61.48	57.81	78.20 △-1.09
	Long GRPO	96.80	96.48	95.62	96.30	82.11	77.66	82.66	80.81	65.47	52.50	63.52	60.49	79.20 △-0.09
4 Opt.:4 Sub.	Instruct	75.00	61.41	67.97	68.12	12.97	9.69	5.31	9.32	1.56	0.39	0.86	0.94	26.13 △+0.00
	Short OPD	93.52	71.02	80.08	81.54	45.78	32.97	25.39	34.71	15.47	6.02	14.06	11.85	42.70 △+16.57
	Short GRPO	85.78	69.84	75.39	77.01	34.14	26.80	15.47	25.47	9.30	2.42	6.33	6.02	36.16 △+10.03
	Mid OPD	91.25	71.56	79.77	80.86	42.19	30.86	26.48	33.18	12.89	6.41	10.94	10.08	41.37 △+15.24
	Mid GRPO	87.73	70.00	77.58	78.44	42.42	31.25	20.62	31.43	13.91	4.38	8.52	8.93	39.60 △+13.47
	Long OPD	92.03	70.78	79.92	80.91	40.39	33.12	24.61	32.71	13.28	4.53	12.58	10.13	41.25 △+15.12
	Long GRPO	86.95	69.30	77.66	77.97	37.66	29.38	19.77	28.93	14.22	2.97	8.36	8.52	38.47 △+12.34
4 Opt.:8 Sub.	Instruct	52.58	40.55	43.28	45.47	0.16	0.16	0.00	0.10	0.00	0.00	0.00	0.00	15.19 △+0.00
	Short OPD	85.86	62.58	72.66	73.70	29.22	22.34	16.02	22.53	6.88	3.52	3.44	4.61	33.61 △+18.42
	Short GRPO	71.48	55.08	60.55	62.37	6.64	6.80	1.88	5.10	0.23	0.39	0.16	0.26	22.58 △+7.39
	Mid OPD	86.72	64.06	71.95	74.24	33.44	20.47	14.84	22.92	8.75	3.91	2.66	5.10	34.09 △+18.90
	Mid GRPO	52.11	38.98	42.42	44.51	0.08	0.08	0.00	0.05	0.00	0.00	0.00	0.00	14.85 △-0.34
	Long OPD	85.00	62.89	71.80	73.23	31.64	20.78	16.09	22.84	7.81	3.67	4.38	5.29	33.78 △+18.59
	Long GRPO	45.55	37.81	38.91	40.76	0.08	0.16	0.00	0.08	0.00	0.00	0.00	0.00	13.61 △-1.58
pass@8														
4 Opt.	Instruct	97.50	97.50	96.25	97.08	86.25	83.12	86.25	85.21	72.50	55.00	66.25	64.58	82.29 △+0.00
	Short OPD	98.12	96.88	95.62	96.88	85.00	80.00	82.50	82.50	68.75	54.38	65.00	62.71	80.69 △-1.60
	Short GRPO	97.50	97.50	96.25	97.08	88.12	83.12	88.12	86.46	72.50	56.25	67.50	65.42	82.99 △+0.69
	Mid OPD	98.12	96.25	95.62	96.67	85.62	79.38	83.12	82.71	68.75	55.00	66.88	63.54	80.97 △-1.32
	Mid GRPO	97.50	97.50	96.25	97.08	86.25	83.12	86.25	85.21	68.12	57.50	66.88	64.17	82.15 △-0.14
	Long OPD	97.50	96.88	95.62	96.67	85.00	80.00	86.88	83.96	68.12	54.38	64.38	62.29	80.97 △-1.32
	Long GRPO	97.50	97.50	95.62	96.88	86.88	81.88	89.38	86.04	71.88	58.75	69.38	66.67	83.19 △+0.90
4 Opt.:4 Sub.	Instruct	95.62	83.75	88.75	89.38	45.62	38.75	25.62	36.67	9.38	2.50	6.25	6.04	44.03 △+0.00
	Short OPD	95.62	78.12	84.38	86.04	51.25	40.00	31.25	40.83	20.00	8.75	18.75	15.83	47.57 △+3.54
	Short GRPO	96.88	86.25	92.50	91.88	63.12	47.50	40.00	50.21	25.00	9.38	20.62	18.33	53.47 △+9.44
	Mid OPD	92.50	77.50	85.00	85.00	51.25	36.88	34.38	40.83	17.50	10.00	15.62	14.38	46.74 △+2.71
	Mid GRPO	96.88	86.25	91.88	91.67	66.25	46.88	45.62	52.92	27.50	13.12	23.12	21.25	55.28 △+11.25
	Long OPD	93.75	75.62	85.00	84.79	47.50	39.38	34.38	40.42	17.50	7.50	16.88	13.96	46.39 △+2.36
	Long GRPO	96.88	86.88	90.62	91.46	61.88	46.88	43.75	50.83	32.50	11.88	21.25	21.88	54.72 △+10.69
4 Opt.:8 Sub.	Instruct	73.12	60.00	65.00	66.04	1.25	1.25	0.00	0.83	0.00	0.00	0.00	0.00	22.29 △+0.00
	Short OPD	87.50	68.12	78.12	77.92	35.00	27.50	21.88	28.12	12.50	5.00	7.50	8.33	38.12 △+15.83
	Short GRPO	90.62	78.12	80.62	83.12	28.12	25.00	11.25	21.46	1.88	2.50	1.25	1.88	35.49 △+13.19
	Mid OPD	88.75	71.25	78.75	79.58	41.25	26.25	20.62	29.38	12.50	5.62	5.00	7.71	38.89 △+16.60
	Mid GRPO	71.25	58.12	66.25	65.21	0.62	0.62	0.00	0.42	0.00	0.00	0.00	0.00	21.88 △-0.42
	Long OPD	89.38	70.00	76.25	78.54	36.25	27.50	21.88	28.54	11.25	5.62	6.25	7.71	38.26 △+15.97
	Long GRPO	70.62	59.38	62.50	64.17	0.62	1.25	0.00	0.62	0.00	0.00	0.00	0.00	21.60 △-0.69

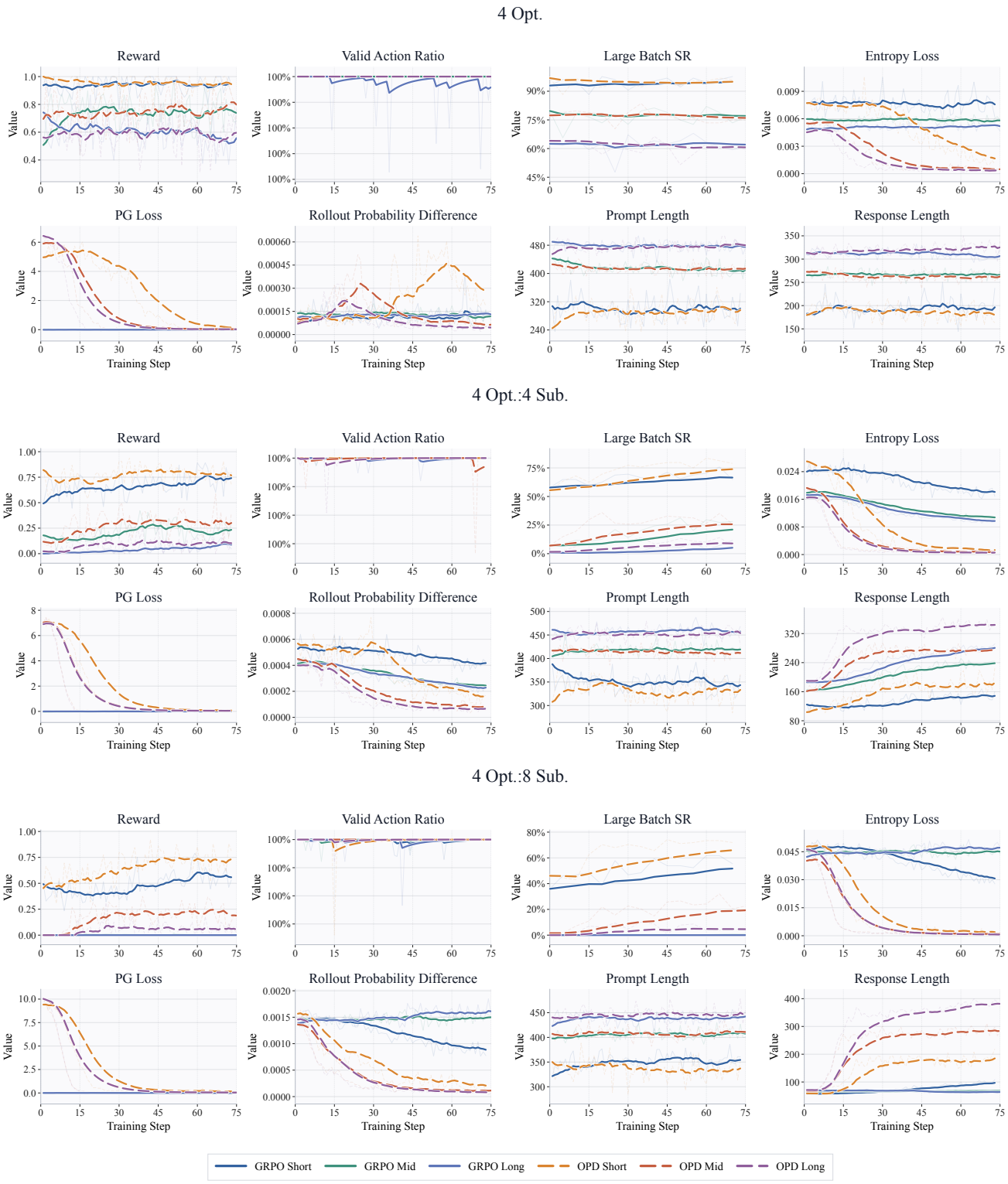


Figure 8: Training dynamics across optimization settings. Comparison of multi-turn GRPO and OPD under 4 Opt., 4 Opt.:4 Sub., and 4 Opt.:8 Sub. settings.

Results. (1) **GRPO and OPD have applicable boundaries based on RL task horizon and data quality (non-optimal action ratio),** as shown in Figure 7, Table 3, and Figure 8. There are three specific regions. **Region A is the unnecessary region, where planning knowledge tokens are less suitable to RL and are better acquired through SFT. Different planning patterns achieve comparable performance, making RL unnecessary.** For GRPO and OPD, even if the teacher performance is 100%, high mutual information planning knowledge tokens are difficult to improve via RL and hard to generalize across samples. These tokens need to be injected via SFT. Moreover, the planning knowledge in this region is highly sample-specific and difficult to transfer across instances. The parameter updates induced by RL optimization are insufficient to fully encode such fine-grained planning knowledge into the model, even with high-quality teacher supervision, making SFT a more effective approach for injecting these patterns. **Region B is the effective region. Low mutual information patterns generalize well through RL.** Low mutual information planning pattern tokens are easy to stimulate via RL and can generalize across samples. This is essentially a process of selecting planning patterns from low-quality pretraining data that are statistically more likely to achieve higher final rewards. **Region C is the unsupported region for long horizon tasks. Long horizon tasks suffer from incorrect credit assignment due to sparse outcome rewards and therefore require fine-grained reward methods.** In long-horizon tasks, when both non-optimal action ratio and RL task horizon increase, correct and incorrect actions are mixed in very long trajectories. ORM based GRPO will assign rewards incorrectly, failing to optimize long horizon agent tasks. In contrast, fine grained reward methods like OPD provide a promising approach. (2) **In the effective region, GRPO and OPD show similar performance improvements. However, OPD reduces entropy and performs worse than the approximate upper bound of GRPO's pass@8 performance,** as shown in Figure 7. In 4 Opt.:4 Sub., OPD leads in pass@1, while GRPO achieves a higher ceiling with larger k. In 4 Opt.:8 Sub., GRPO helps short-horizon but offers little gain on long-horizon tasks. Meanwhile, OPD shows better pass@1 and pass@8. This indicates that GRPO generates the best planning pattern based on the environment, showing stronger potential. However, OPD suppresses diversity and directly follows the planning pattern of the teacher. Therefore, it is clear that the correctness of the teacher planning pattern is extremely important. (3) **Pass@k may not always provide an accurate characterization of the upper bound of model capability in multi-turn long-horizon planning scenarios,** as shown in Figure 7 and Table 3. This observation is motivated by the fact that the pass@k performance before RL training is nearly zero in these long-horizon settings, while RL training can lead to substantial improvements. Such a near-zero pass@k score does not necessarily imply that the base model lacks the ability to generate successful trajectories, as the necessary knowledge or solution patterns may exist in the pretrained model but are rarely sampled. Instead, it suggests that pass@k may gradually become a less accurate estimator of the model capability upper bound as the planning horizon becomes increasingly long. (4) **RL shift CoT length toward high-reward patterns in pre-training data.** After training, the output length of model will change, which is related to the data distribution in pretraining. If the better planning pattern is longer, the trained output length will increase. If the better planning pattern is shorter, the trained output length may decrease.

Takeaway

Planning knowledge is better suited for SFT (unnecessary region). When planning knowledge has high mutual information, different samples may require different optimization directions. The limited update magnitude of RL makes it difficult to inject such knowledge effectively.
Planning patterns are better suited for RL (effective region). When planning patterns have low mutual information, different samples share similar optimization directions. RL can stimulate these patterns and improve generalization across samples that contain similar patterns in the pretraining corpus with high expected rewards.
Multi-turn long-horizon tasks need fine grained credit assignment like OPD (GRPO unsupported region). Outcome based rewards struggle when correct and incorrect actions mix in long trajectories, making fine grained reward based methods necessary. OPD demonstrates a larger effective region than GRPO with an ideal teacher.
OPD follows teacher patterns to improve performance, but it reduces entropy and limits its upper bound. In the effective region, OPD performs similarly to GRPO but scales worse with more sampling times.
Although pass@k exceeds the base model with large k in long-horizon tasks, we do not believe that RL improves the model's capability ceiling. Instead, we argue that pass@k is distorted in this setting because pre-training data may already contain the correct trajectories.
RL shifts CoT length toward high-reward patterns in the pretraining data. The output length increases depending on whether the better planning patterns are longer.

5.2.2. *Gradient Direction Analysis*

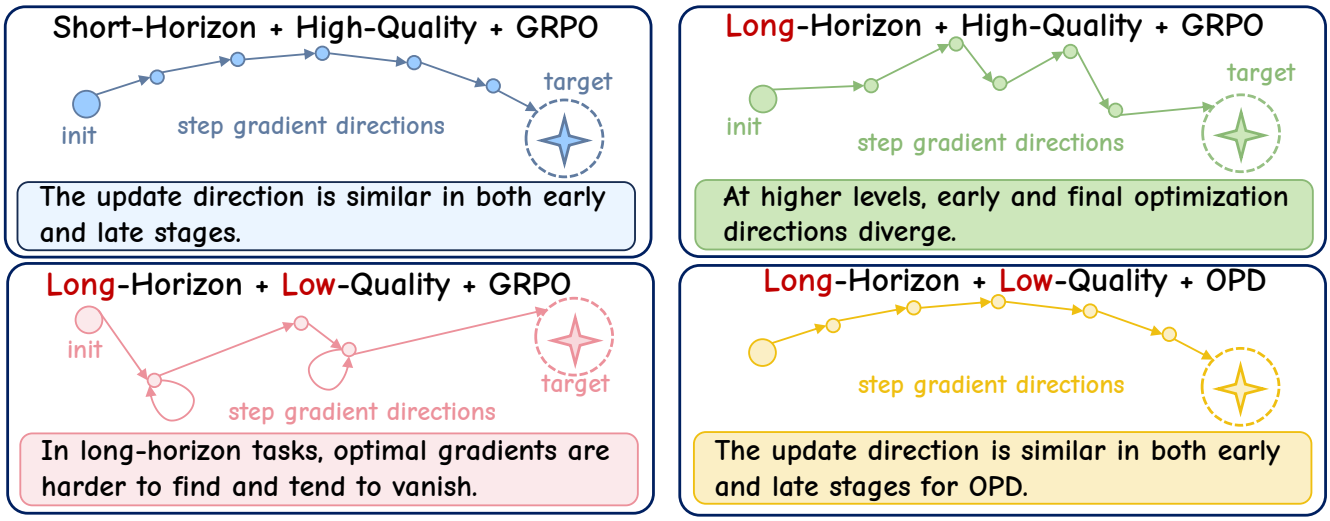


Figure 9: Comparison of step gradient directions between GRPO and OPD under varying optimization conditions.

Top-k Subspace for SVD. Beyond measuring task performance, we analyze whether the model already follows the final parameter-update direction in the early stages of training. Cai et al. (2026) analyzes the gradient direction. We further show how the update direction differs under two dimensions: pre-training data quality and post-training horizon. Specifically, we compare the dominant update directions of early and intermediate checkpoints with those of the final checkpoint. Let $W_\ell^{(t)}$ be the weight matrix of layer ℓ at checkpoint t, and let b be the base checkpoint. We first define the update matrix

$$\Delta W_\ell^{(t)} = W_\ell^{(t)} - W_\ell^{(b)}. \tag{11}$$

For each $\Delta W_\ell^{(t)}$, we compute SVD:

$$\Delta W_\ell^{(t)} = U_\ell^{(t)} \Sigma_\ell^{(t)} \left(V_\ell^{(t)}\right)^\top. \tag{12}$$

We keep the first k left singular vectors,

$$U_{\ell,k}^{(t)} = [u_{\ell,1}^{(t)}, u_{\ell,2}^{(t)}, \ldots, u_{\ell,k}^{(t)}]. \tag{13}$$

Given a target final checkpoint f, we compare checkpoint t with f by cosine similarity of matched singular vectors:

$$s_{\ell,i}(t,f) = \left| \frac{\left(u_{\ell,i}^{(t)}\right)^\top u_{\ell,i}^{(f)}}{\left\|u_{\ell,i}^{(t)}\right\|_2 \left\|u_{\ell,i}^{(f)}\right\|_2} \right|, \quad i = 1, \ldots, k. \tag{14}$$

We use absolute value because SVD vectors can flip sign. The layer-level Top-k score is

$$s_\ell^{(k)}(t,f) = \frac{1}{k} \sum_{i=1}^{k} s_{\ell,i}(t,f). \tag{15}$$

Finally, we average over all valid layers $\mathcal{L}$:

$$S^{(k)}(t,f) = \frac{1}{|\mathcal{L}|} \sum_{\ell \in \mathcal{L}} s_\ell^{(k)}(t,f), \quad S^{(k)}(t,f) \in [0,1]. \tag{16}$$

A larger value means the update direction at checkpoint t is closer to the final update direction.

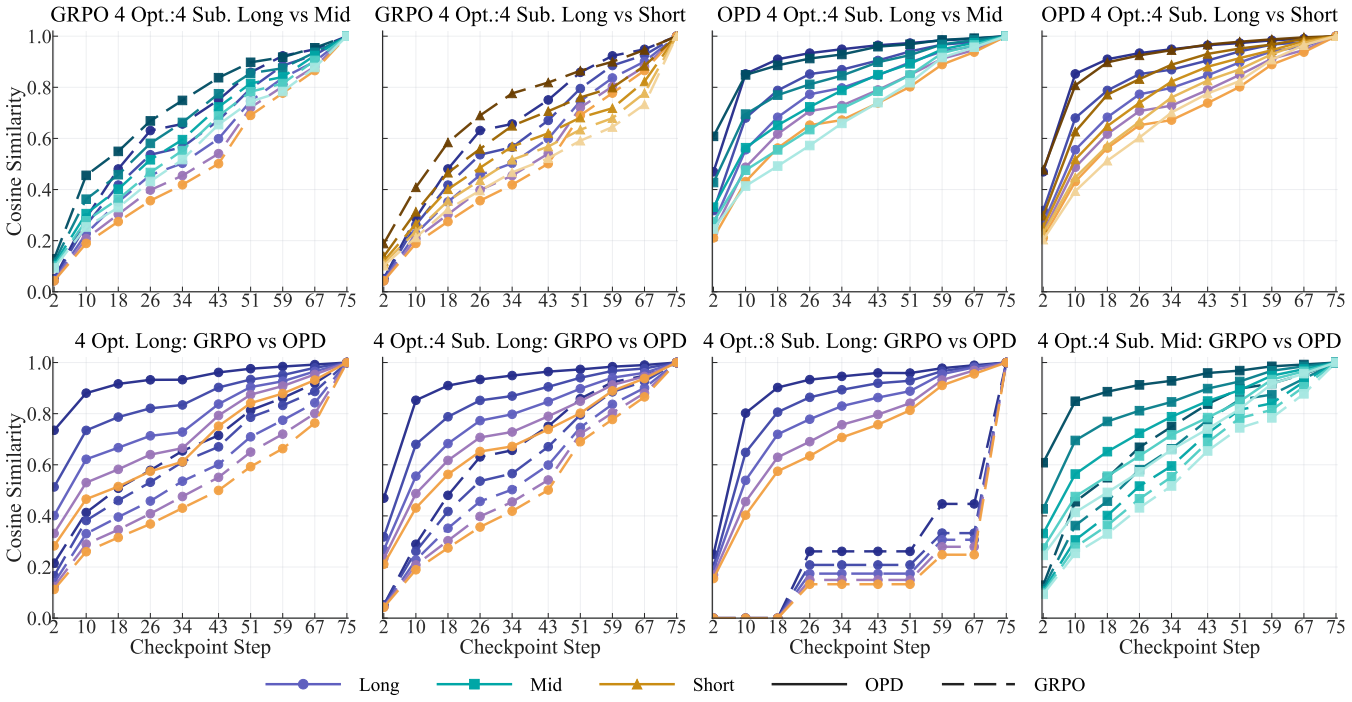


Figure 10: Evolution of gradient directions. Top-5 SVD direction similarity between intermediate and final checkpoints.

Experimental Setup. (1) **Horizon length.** Under the 4 Opt.:4 Sub setting, we evaluate both the GRPO and OPD algorithms. For each configuration, we conduct pairwise comparisons across the short, middle, and long levels. (2) **Algorithm comparison.** We fix the difficulty at the long-horizon and utilize three settings: 4 Opt., 4 Opt.:4 Sub, and 4 Opt.:8 Sub. This part primarily compares the parameter update directions between the GRPO and OPD algorithms under long-horizon conditions. (3) **Top-k Subspace.** We select $k = 5$ for the top-k subspace analysis. For every layer, we compute the parameters of both the FFN and Attention modules. The maximum number of training steps is set to 75. We select step 1 as the initial checkpoint and uniformly sample 10 intermediate checkpoints for evaluation.

Results. As shown in Figure 9 and Figure 10. (1) **Coarse credit assignment leads to noisy GRPO gradient direction in long horizons.** The first row display the performance of the GRPO algorithm across different horizon levels. The plotted curves visually demonstrate that as the task horizon increases, the numerical values remain lower. This directly indicates that as the horizon becomes longer, the coarse granularity of credit assignment causes the parameter update direction of GRPO to become inconsistent. (2) **Vanishing gradients with increasing planning horizon and non-optimal actions.** Observing the subfigures in the second row, specific GRPO curves begin exactly at zero and remain completely flat along the x-axis during the initial checkpoint steps. This distinct flat trajectory at zero provides evidence that continuously increasing the number of non-optimal actions triggers a vanishing gradient issue. Because of this issue, the GRPO gradients fail to undergo any updates. (3) **Superior directional consistency of OPD.** The second row of subfigures directly compares the GRPO and OPD algorithms across multiple configurations. The charts illustrate that the GRPO curves only reach higher values under restricted simple conditions. This proves that the update direction of GRPO is not universally consistent, relies heavily on specific applicability conditions, and is unsuitable for real long horizon scenarios. In contrast, the OPD curves maintain a relatively high level across all evaluated settings. This confirms that OPD maintains a much more consistent parameter update direction.

Takeaway

OPD exhibits stronger directional stability than GRPO under long-horizon and low-quality pre-training corpus settings. While GRPO shows inconsistent or even vanishing update directions as task horizon and non-optimal actions increase, OPD maintains consistently high subspace alignment, suggesting more reliable credit assignment in ideal teacher condition.

5.3. Impact of Planning Knowledge

5.3.1. Accuracy Analysis

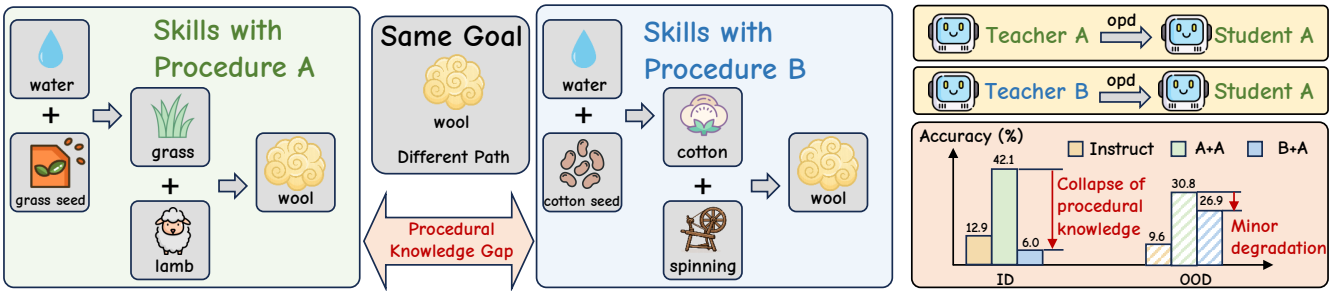


Figure 11: The procedural knowledge gap during teacher-student distillation. Distilling a different procedural path for the same goal via the OPD method (teacher B to student A) leads to a substantial collapse of procedural knowledge and severely damages ID performance, while mildly affecting OOD generalization.

Knowledge Gap in OPD. In real-world environments, there are multiple reasonable and effective paths to complete a task (e.g., Recipe A and Recipe B). This multi-path characteristic exposes a core issue in knowledge distillation: when the student and teacher models form divergent procedural knowledge based on different pre-training corpora, can this knowledge be effectively distilled? To this end, by comparing the mismatched setting (student with recipe A + teacher with recipe B) with the aligned setting (student with recipe A + teacher with recipe A), we investigate whether two completely different sets of procedural knowledge can successfully bridge this significant gap and be distilled. Specifically, we explore the scenario where the student model conducts procedural knowledge world modeling based on Recipe A, while the teacher model provides procedural knowledge world modeling based on Recipe B. As shown in Figure 11, we illustrate this gap using a specific crafting example where both procedures share the same final goal of producing wool. Procedure A creates wool using water, grass seed, and a lamb. Procedure B achieves the identical goal using water, cotton seed, and spinning. When Student A, who is pre-trained on Procedure A, receives distillation from Teacher B via the OPD method, a severe collapse of procedural knowledge occurs. This structural mismatch leads to a significant drop in in-distribution accuracy compared to the aligned setting where Teacher A is used.

Task Setup. (1) **Models.** For the base model, we use 4 Opt.:4 Sub setting. For the teacher models, we train the teacher models separately using Recipe A and Recipe B, respectively. Although these two recipes use different raw materials and input materials for the same target, both can successfully make the target. (2) **Training.** We train the models on the middle-horizon RL training set. The subset uses FA domain.

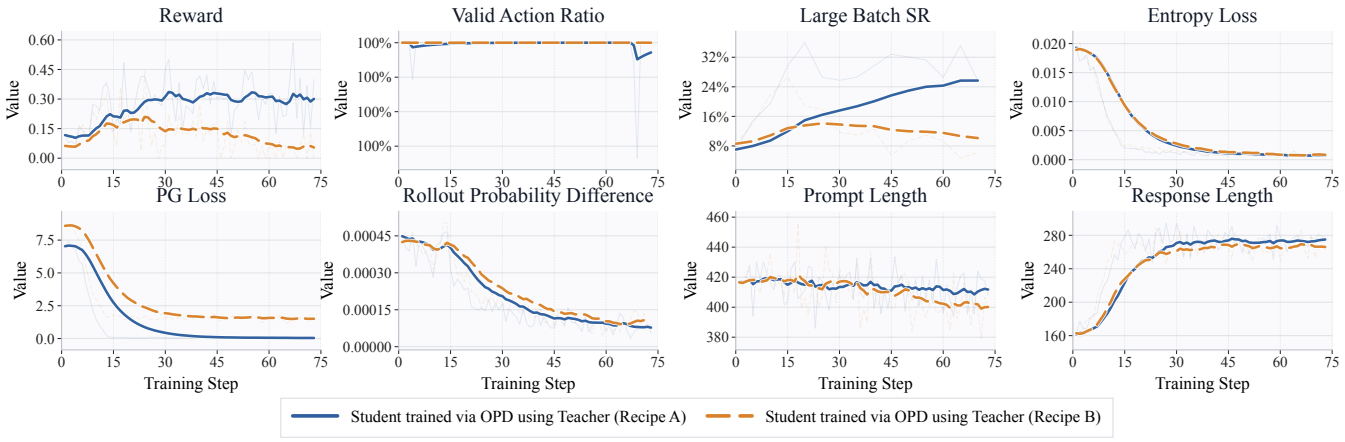


Figure 12: Comparison of training dynamics under two OPD teacher recipes.

Table 4: Knowledge gap comparison for the 4 Opt.:4 Sub pattern mixture in RL. The final $\Delta(B-A)$ row reports Student + Teacher B minus Student + Teacher A.

Pattern Mix	Model	Short				Middle				Long				Avg
		FA	LF	EA	Mean	FA	LF	EA	Mean	FA	LF	EA	Mean	S/M/L Mean (△)
avg@8														
4 Opt.:4 Sub.	Instruct Model	75.00	61.41	67.97	68.12	12.97	9.69	5.31	9.32	1.56	0.39	0.86	0.94	26.13 △+0.00
	Teacher A	100.00	–	–	100.00	100.00	–	–	100.00	100.00	–	–	100.00	100.00 △+73.87
	Teacher B	100.00	–	–	100.00	100.00	–	–	100.00	100.00	–	–	100.00	100.00 △+73.87
	Student + Teacher A	91.25	71.56	79.77	80.86	42.19	30.86	26.48	33.18	12.89	6.41	10.94	10.08	41.37 △+15.24
	Student + Teacher B	72.19	63.91	78.75	71.61	6.09	26.95	20.78	17.94	2.27	5.31	7.27	4.95	31.50 △+5.37
	Δ (B - A)	-19.06	-7.66	-1.02	-9.24	-36.09	-3.91	-5.70	-15.23	-10.62	-1.09	-3.67	-5.13	-9.87
pass@8														
4 Opt.:4 Sub.	Instruct Model	95.62	83.75	88.75	89.38	45.62	38.75	25.62	36.67	9.38	2.50	6.25	6.04	44.03 △+0.00
	Teacher A	100.00	–	–	100.00	100.00	–	–	100.00	100.00	–	–	100.00	100.00 △+55.97
	Teacher B	100.00	–	–	100.00	100.00	–	–	100.00	100.00	–	–	100.00	100.00 △+55.97
	Student + Teacher A	92.50	77.50	85.00	85.00	51.25	36.88	34.38	40.83	17.50	10.00	15.62	14.38	46.74 △+2.71
	Student + Teacher B	75.62	73.75	85.00	78.12	7.50	34.38	29.38	23.75	3.75	8.75	11.25	7.92	36.60 △-7.43
	Δ (B - A)	-16.88	-3.75	+0.00	-6.88	-43.75	-2.50	-5.00	-17.08	-13.75	-1.25	-4.38	-6.46	-10.14

Results. As shown in Figure 12 and Table 4. (1) **Knowledge mismatch blocks distillation even with perfect performance teachers.** Teacher B achieves a perfect score of 100.00 in all tasks. However, training a student with a different recipe fails completely. The Student + Teacher B model gets an overall pass@8 score of only 36.60. This is worse than the base instruct model score of 44.03. This shows that a highly capable teacher cannot improve the student if their training recipes do not match. (2) **Mismatched procedural distillation collapses ID planning knowledge while mildly affecting OOD generalization.** When the teacher and student acquire different but both valid procedural paths for the same goal, forcing distillation of the mismatched path causes a substantial collapse of the student's original ID planning knowledge. This disrupts the planning knowledge tokens for FA world modeling, leading to severe ID performance degradation, while LF and EA are less affected because their corresponding world modeling is not directly overwritten. For example, in the Middle difficulty FA task, Student + Teacher B reduces avg@8 and pass@8 by 36.09 and 43.75 points compared with Student + Teacher A. (3) **Large knowledge gap causes training collapse in later stages.** Figure 12 shows that the performance goes up in the early stage because the student aligns the planning pattern. In the later stage, the main optimization direction is to align the planning knowledge. However, the knowledge gap between the two recipes is too big, which finally leads to a complete training collapse.

Takeaway

OPD requires aligned procedural knowledge between student and teacher. OPD cannot effectively distill knowledge from a perfect teacher when the student and teacher internalize different procedural paths for the same goal. Successful distillation requires similar internal world modeling between the teacher and student.
Mismatched procedural distillation collapses ID planning knowledge while mildly affecting OOD generalization. When teacher and student follow different valid procedures, distilling the mismatched path overwrites the student's existing planning knowledge tokens.
OPD distills patterns early but fails on mismatched knowledge late. In the early stage, OPD successfully distills better planning patterns to improve performance. In the later stage, forcing OPD to distill completely different procedural knowledge leads to a training collapse.

5.3.2. KL Dynamic Analysis

KL Dynamic and Overlap. We use KL divergence to measure how close a trained student is to one of the two teacher models at the token distribution level. Let T_A and T_B denote the two teacher models. All student checkpoints start from the same A recipe. We use $\sigma \in \{A, B\}$ to denote which teacher is used to train the student, and $\gamma \in \{A, B\}$ to denote which teacher is used for KL comparison. The student trained with teacher T_σ is denoted as π_{θ_σ}. A trajectory $\tau \sim \pi_{\theta_\sigma}$ is rolled out by this student. On this fixed trajectory, the student policy $p^{\sigma}_{k,t}$ and the comparison teacher policy

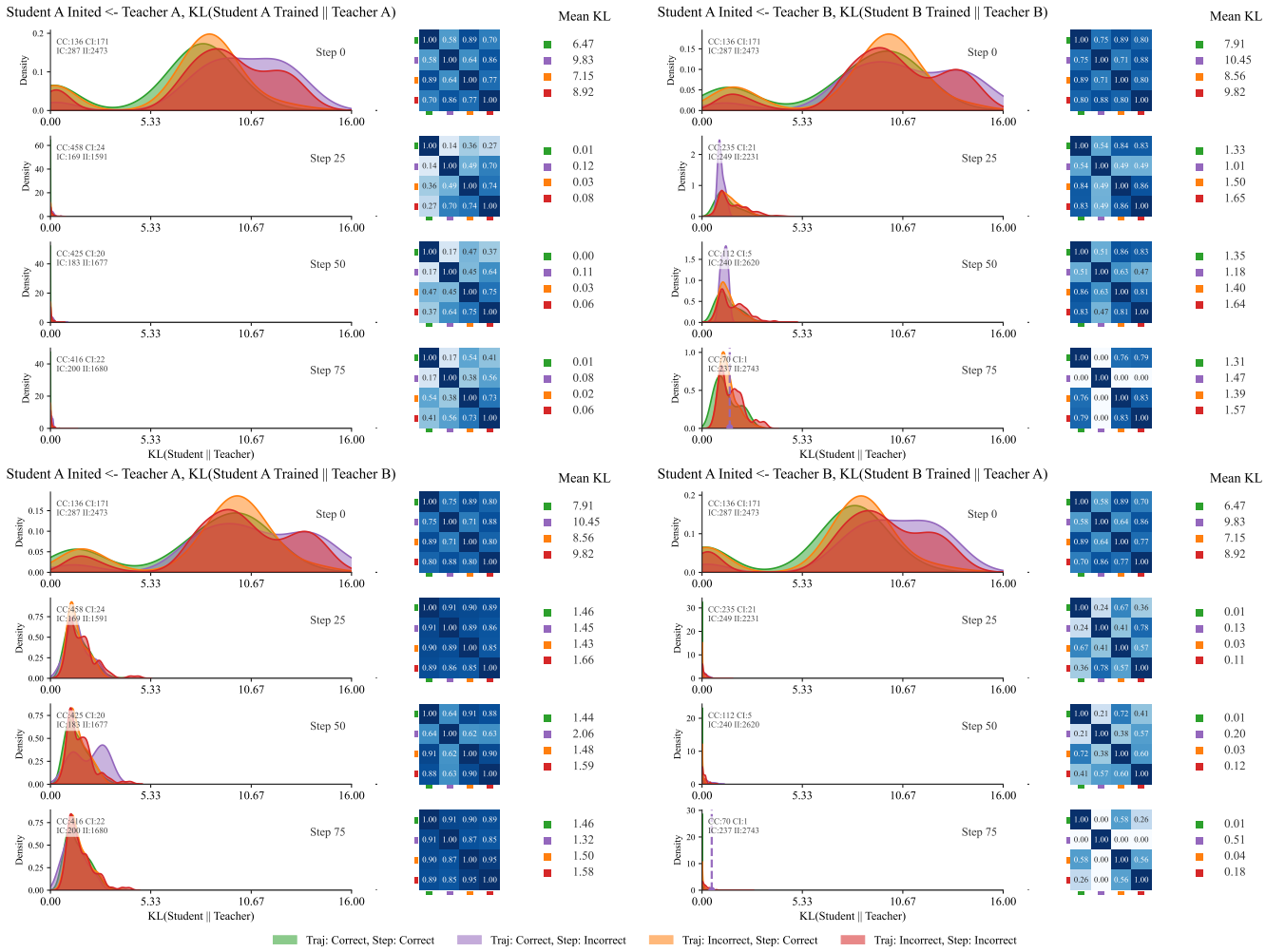


Figure 13: Token-level KL divergence distributions between students and matched or crossed teachers across training steps. Samples are grouped by trajectory-level and step-level correctness.

$q^{\gamma}_{k,t}$ evaluate next-token probabilities at decoding step t within turn k, conditioned on the same prefix. Let $\mathcal{V}_{k,t}$ be the top-k token subset, and let $\ell^{p,\sigma}_{k,t}(v)$ and $\ell^{q,\gamma}_{k,t}(v)$ be the stored log probabilities for token v.

$$
\begin{aligned}
p^{\sigma}_{k,t}(v) &= \exp\left(\ell^{p,\sigma}_{k,t}(v)\right), \quad q^{\gamma}_{k,t}(v) = \exp\left(\ell^{q,\gamma}_{k,t}(v)\right), \\
p^{\sigma}_{k,t,\text{tail}} &= 1 - \sum_{v\in\mathcal{V}_{k,t}} p^{\sigma}_{k,t}(v), \quad q^{\gamma}_{k,t,\text{tail}} = 1 - \sum_{v\in\mathcal{V}_{k,t}} q^{\gamma}_{k,t}(v).
\end{aligned}
\tag{17}
$$

For each token position, we compute the KL from the student distribution to the selected teacher distribution, and average it over the whole trajectory. We use four comparison settings: setting 1 uses the student trained with T_A and compares it with T_A; setting 2 uses the student trained with T_B and compares it with T_B; setting 3 uses the student trained with T_A and compares it with T_B; setting 4 uses the student trained with T_B and compares it with T_A.

$$
\mathrm{KL}^{\sigma,\gamma}(\tau) = \frac{1}{\sum_{k=1}^{K} T_k} \sum_{k=1}^{K} \sum_{t=1}^{T_k} \sum_{v\in\mathcal{V}_{k,t}\cup\{\text{tail}\}} P^{\sigma}_{k,t}(v) \log \frac{P^{\sigma}_{k,t}(v)}{Q^{\gamma}_{k,t}(v)}.
\tag{18}
$$

We further split trajectories into four data types according to trajectory correctness and step correctness: correct trajectory with correct step, correct trajectory with incorrect step, incorrect trajectory with correct step, and incorrect

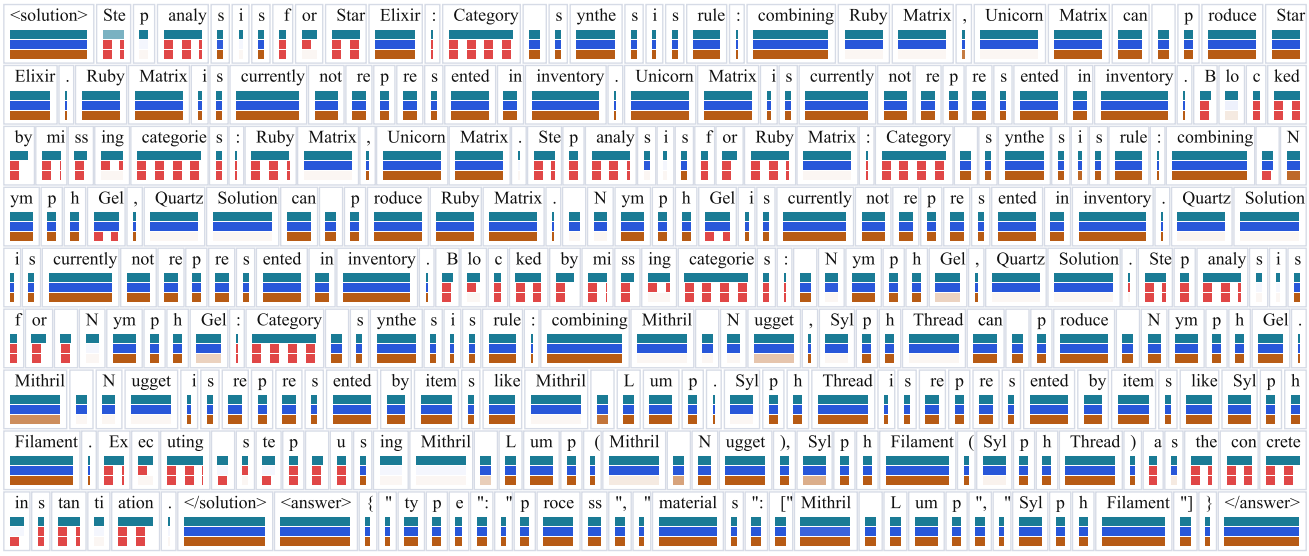


Figure 14: Token-level probability heatmap on a student A rollout. This figure shows an early-stage RL optimization, mainly focusing on optimizing the tokens corresponding to planning patterns that have high cross-sample generalization. For each token generated by student A, the top cyan, middle blue, and bottom orange bars show the probabilities assigned by student A, teacher A, and teacher B, respectively. Darker colors indicate higher probabilities. Red striped bars indicate that the generated token is absent from the corresponding teacher's top-k predictions.

trajectory with incorrect step. Let $\mathcal{R}_c$ be the set of trajectories in data type c. For each training step, comparison setting (σ, γ), and data type c, we report the mean KL. To compare two data types c and c', we estimate their KL density functions $f_c(x)$ and $f_{c'}(x)$ with Gaussian kernel density estimation, where x is a trajectory-level KL value, and compute their overlap.

$$\overline{\mathrm{KL}}_c^{\sigma,\gamma} = \frac{1}{|\mathcal{R}_c|} \sum_{\tau \in \mathcal{R}_c} \mathrm{KL}^{\sigma,\gamma}(\tau), \quad \mathrm{Overlap}(c, c') = \int \min\left(f_c(x), f_{c'}(x)\right)\, dx. \tag{19}$$

Task Setup. (1) **Data and model.** We use the two models trained in the Section 5.3 to perform inference on the FA test set with middle difficulty. The rollout process uses two models pretrained from recipe A and trained later by teacher A and teacher B respectively. Then we calculate the reverse KL divergence. We also divide the rollout data into four types based on trajectory correctness and single step correctness to obtain the distribution, overlap rate, and mean of the KL divergence. (2) **Distribution recording and divergence calculation.** During the KL divergence calculation stage, the models perform a rollout independently and record the probability distribution. Subsequently, teacher A and teacher B record the output probability distributions on these rollout trajectories and calculate the reverse KL divergence based on this. (3) **Data classification and statistical settings.** We divide the rollout data into four types based on trajectory correctness and single step correctness to obtain the distribution, overlap rate, and mean of the KL divergence. At the same time, we set the K value of top K to 100 and calculate the RL training dynamics at four points, which are 0, 25, 50, and 75.

Results. As shown in Figure 13 and Figure 14 and Figure 15. (1)**Need for error correction data.** The proportion of the four data types shows that the number of correct trajectories with incorrect steps is extremely small. This indicates a need to add more error correction data. Gold standard data alone fails to stimulate the recovery ability from error state prefixes. (2)**Optimization of planning knowledge.** As shown in the first group of figures, the KL divergence on the left side is successfully optimized down. In contrast, after optimizing the planning patterns, the KL divergence on the right side becomes difficult to decrease further. Planning knowledge has large token mutual information and requires point by point optimization. The reinforcement learning update magnitude is too small and is insufficient to transfer massive amounts of knowledge to the domain of teacher B. (3)**Reward correlation and knowledge disruption.** Based on the average KL values, the mean KL divergence for many correct steps is larger than that for incorrect steps. The correlation between the mean KL divergence and the environment reward is smaller for

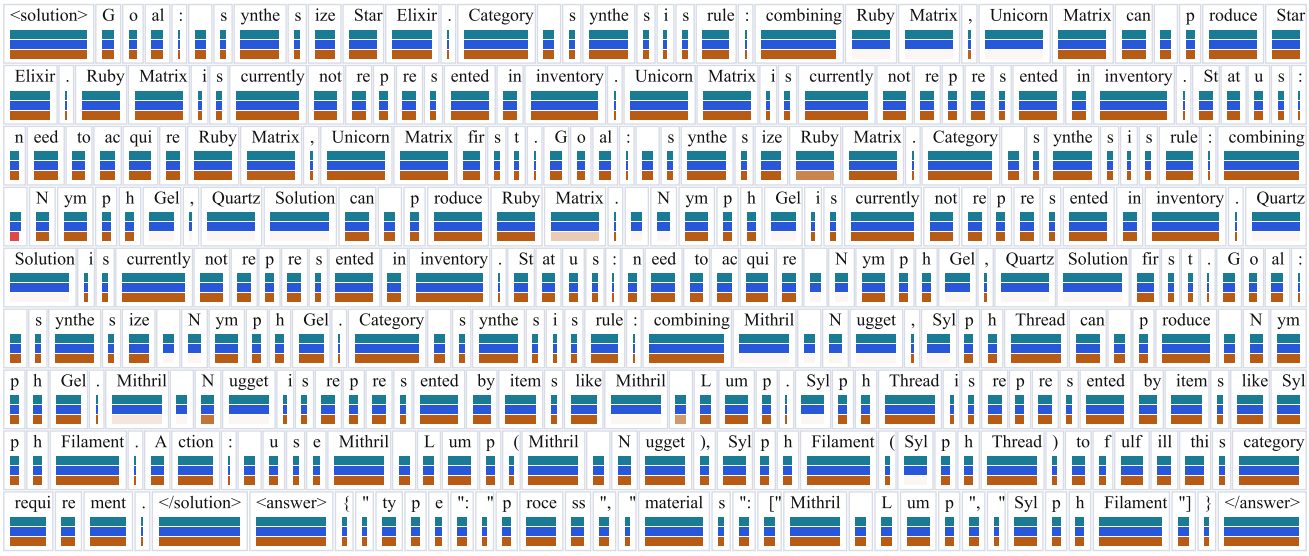


Figure 15: Token-level probability heatmap on a Student A rollout for step 75. This figure shows a later-stage RL optimization, which mainly focuses on optimizing sample-specific planning knowledge. However, the optimization capacity of RL is insufficient to inject highly specialized planning knowledge.

mismatched planning knowledge compared to matched planning knowledge. Completing a single sample involves multiple reasoning paths. If the teacher fails to cover these paths, the model treats other correct paths as incorrect ones. This process destroys the original knowledge of the model. (4)**Transfer failure in later stages.** As shown in Figure 14 and Figure 15, both models optimize the planning pattern in the early stages. In the later stages, teacher B primarily optimizes planning knowledge. However, planning knowledge varies greatly across different samples. The reinforcement learning update fails to transfer from knowledge A to knowledge B.

Takeaway

Incorporate error correction data. Relying solely on gold standard data is insufficient. Training pipelines need error state prefixes to teach models how to recover from mistakes.
RL updates are too small to support sample-specific large-scale knowledge distillation. Standard RL updates are too small for the point by point optimization required to distill massive planning knowledge.
The same target can be reached by multiple different but correct reasoning paths. If the teacher misses them, OPD may assign wrong probabilities to valid paths and hurt the student's existing knowledge via KL loss.
Early pattern learning improves RL rewards, but failure to learn task-specific tokens may cause later degradation. Models easily optimize general planning patterns early in training, but they struggle to distill high variance planning knowledge in later stages.

6. Planning Ability Integration through Multi-Teacher On-Policy Agentic Distillation

6.1. Definitions and Research Problems

Multi-Teacher On-Policy Agentic Distillation (MOPD) is a method that distills and combines the abilities of multiple teacher models into one student model. Existing technical route mainly follow two paradigms, as shown in Table 5. **(1) Cross domain capability integration.** This paradigm trains the student model in different environments based on the student model. After training, teacher models from different environments are merged to integrate the abilities of multiple experts (Xiao et al., 2026, Xu et al., 2026). **(2) Cross stage capability retention.** This paradigm uses curriculum learning to train the student model. The model learns from reasoning ability to agentic ability gradually. In the final stage, the final model from each stage is used as a teacher model and then merged to recover abilities that are forgotten in earlier stages (Zeng et al., 2026, Yang et al., 2026c).

Table 5: Comparison of MOPD paradigms across different foundation models.

Foundation Models	Training Paradigm	Student	Teacher	Primary Motivation
MiMo-V2-Flash (Xiao et al., 2026)	Pre-training → Domain training → MOPD	SFT Model	Domain-specialized Model	Cross domain capability integration
GLM-5 (Zeng et al., 2026)	Pre-training → Mid-training → Cascading RL → MOPD	Last RL checkpoint	Previous RL checkpoint	Cross stage capability retention
Nemotron-Cascade 2 (Yang et al., 2026c)	Pre-training → Multi-domain RL → MOPD → Cascading RL	Middle RL checkpoint	Previous RL checkpoint	Cross stage capability retention
DeepSeek-V4 (Xu et al., 2026)	Pre-training → Domain training → MOPD	SFT Model	Domain-specialized Model	Cross domain capability integration

Multi-teacher agent-OPD can generally be categorized into two paradigms: mixture and cascaded. Given that the mixture approach incurs infrastructure overhead (e.g., simultaneous deployment and inference of multiple teacher models), we focus on the cascaded form. Under this sequential training paradigm, the student policy learns from one teacher at a time. For the m-th teacher ($m \in \{1, \ldots, M\}$), the objective provides dense, step-by-step supervision by aligning the student's predictions with those of the specific teacher:

$$\mathcal{L}^{(m)}_{\text{Agent-OPD}}(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\left[\sum_{k=1}^{K}\sum_{t=1}^{T_k} D_{\text{KL}}(p_{k,t} \,\|\, q^{(m)}_{k,t})\right], \tag{20}$$

where $q^{(m)}_{k,t} \triangleq \pi^{(m)}_{\text{teacher}}(\cdot \mid H_k, \hat{Y}_{k,<t})$ denotes the next-token distribution provided by the m-th teacher. The overall training proceeds by iteratively minimizing $\mathcal{L}^{(m)}_{\text{Agent-OPD}}(\theta)$ in sequence from $m = 1$ to M.

Although MOPD is widely used in existing foundation models, the training mechanisms and conditions for MOPD remain unclear. Therefore, we urgently need to analyze the MOPD algorithm deeply to explore its working conditions and boundaries. Thus, we propose three progressive research questions:

- **Q1: Generalization Mode.** Under what conditions can MOPD effectively achieve cross-environment generalization?
- **Q2: Continual Learning Mode.** Under what conditions can MOPD address limited cross-environment generalization through continual learning without conflicts?
- **Q3: Conflict Mode.** Under what conditions does MOPD lose cross-environment generalization and suffer severe conflicts among environments?

To analyze the three problems above, we propose a unified framework. As shown in Figure 16, we use two dimensions for our analysis: (1) whether there is a shared area of planning patterns, and (2) whether the planning patterns are compatible with different environments. Based on these two dimensions, we classify the relationship between planning patterns and environments into three typical coverage patterns: Type I (Shared and Compatibility Planning Patterns), Type II (Shared and Conflict Planning Patterns), and Type III (No Shared and Conflict Planning Patterns). These structural relationships determine how the planning patterns distribute between the student and multiple teachers across different environments.

6.2. Shared and Compatibility Multi Teachers On-Policy Agentic Distillation

To answer the **Q1**, we focus on the **Type I** category: **Shared and Compatibility Planning Patterns**. In this scenario, different environments share common planning patterns that are entirely compatible and free of cross-domain conflicts.

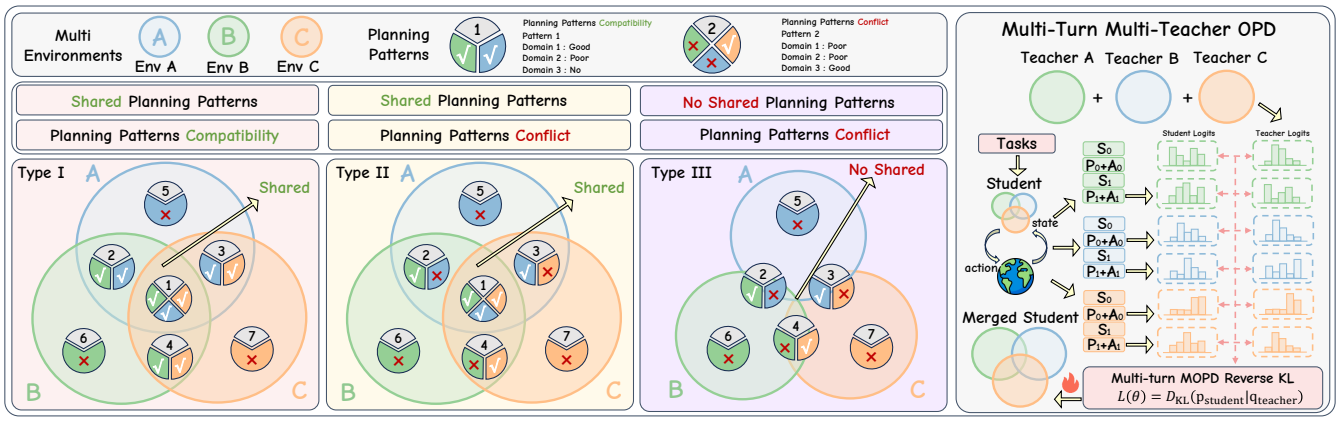


Figure 16: Multi-Turn MOPD. Analysis framework of non-shared and conflicting planning patterns in different environments for multi-teacher on-policy distillation.

This implies that the student and the multiple teachers possess an intersecting and consistent understanding of the tasks across different domains. To simulate this structural relationship and empirically evaluate how MOPD achieves effective cross-environment generalization under these conditions, we design the following experimental setup.

Task Setup. (1) **Teacher models.** The experiment includes three domains: fantasy alchemy, livestock farming, and electronic assembly. We train an expert teacher model for each domain, so we get three domain expert teachers in total. Each teacher model learns from the gold trajectories in the pretraining and posttraining datasets of its domain. To bring in different planning supervision signals, we design four planning patterns for internal world models. Three expert teachers share one planning template. We assign the other templates to only some teachers. We distribute three planning patterns equally for each teacher. Therefore, all three teachers learn the correct trajectory supervision, but they see different planning template distributions and have varied planning patterns. (2) **Student models.** The student models use 4 Opt.:4 Sub. and 4 Opt.:8 Sub. as base models. The 4 Opt.:4 Sub. setting has two versions: 4 Opt.:4 Sub.(a) and 4 Opt.:4 Sub.(b). For the student SFT data, we build two group settings on the same three domains: electronic assembly, fantasy alchemy, and livestock farming. Both settings build on pretraining trajectories and include two kinds of supervision signals: normal planning samples from gold actions, and shortcut style distractor samples with wrong actions. Both settings use four kinds of normal planning templates and four kinds of distractor planning templates. We make the sample size of each template type equal to keep an overall balance between normal samples and distractor samples. The main difference between the two settings is the domain and template distribution of normal planning samples. Form a uses a mixed distribution: the three domains share one normal planning template, and the other normal planning templates only appear in two domains. Therefore, the subsets of planning patterns seen by different domains are not exactly the same. Form b removes this difference: every normal planning template appears in all three domains with equal sampling. Thus, Form a focuses on the unequal planning pattern exposure among domain experts. Form b makes sure each domain covers all planning templates, while keeping the same domain scope and equal distractor construction. (3) **Training configurations.** To ensure a strict comparison, the three student models use the exact same amount of pretraining data and the same training parameters. The student models go through OPD one by one in the order of FA, LF, and EA. We train the OPD on the posttraining dataset with a middle difficulty level.

Table Results. As shown in Table 6. (1) **Shared and compatible planning patterns are a basic condition for cross environment generalization.** It shows that although teacher models only work in their own domains (for example, Teacher FA has an Avg@8 of 100.00 in the FA domain but 0.00 in LF and EA), student models achieve significant performance improvements in unseen domains after distillation from a single domain teacher. Take 4 Opt.:4 Sub.(a) as an example. In Middle difficulty, for in-domain environments, the avg@8 of the base Instruct model in the FA domain increases from 41.95 to 61.64. For out of domain environments, the avg@8 of the base Instruct model in the LF domain is 34.92. However, after it receives distillation only from the FA domain teacher, its performance in the LF domain jumps directly to 52.58. This indicates that when different environments shared and compatible planning patterns, MOPD breaks domain barriers and allows the model to transfer abilities learned in one environment to other environments.

Table 6: Ensemble main table. Rows are grouped by pattern mixture. The teacher group reports Teacher FA, Teacher LF, and Teacher EA. The MOPD order is FA->LF->EA. Two global blocks are reported: avg@8 and pass@8. Domain abbreviations: FA = Fantasy Alchemy; LF = Livestock Farming; EA = Electronic Assembly.

Pattern Mix	Model	Short				Mid				Long				Avg
		FA	LF	EA	Mean	FA	LF	EA	Mean	FA	LF	EA	Mean	S/M/L Mean
avg@8														
Teacher	Teacher FA	100.00	0.00	0.00	33.33	100.00	0.00	0.00	33.33	100.00	0.00	0.00	33.33	33.33
	Teacher LF	0.00	100.00	0.00	33.33	0.00	100.00	0.00	33.33	0.00	100.00	0.00	33.33	33.33
	Teacher EA	0.00	0.00	100.00	33.33	0.00	0.00	100.00	33.33	0.00	0.00	100.00	33.33	33.33
4 Opt.:4 Sub.(a)	Instruct	86.25	80.70	79.45	82.14	41.95	34.92	30.08	35.65	13.83	8.83	8.91	10.52	42.77
	After FA	92.50	88.59	86.41	89.17	61.64	52.58	48.59	54.27	31.64	24.38	22.19	26.07	56.50
	After LF	90.55	92.81	85.55	89.64	60.55	58.36	48.75	55.89	28.75	27.66	28.44	28.28	57.93
	After EA	88.59	92.73	85.08	88.80	51.95	47.42	47.81	49.06	22.50	18.83	27.81	23.05	53.64
4 Opt.:4 Sub.(b)	Instruct	77.19	62.66	68.44	69.43	14.38	9.53	5.23	9.71	1.17	0.23	0.70	0.70	26.61
	After FA	91.02	73.59	81.25	81.95	45.08	36.95	25.16	35.73	15.47	5.39	12.27	11.04	42.91
	After LF	90.00	73.05	80.62	81.22	45.16	33.36	26.02	34.84	15.23	6.88	13.44	11.85	42.64
	After EA	85.00	69.69	79.53	78.07	37.42	30.62	21.56	29.87	10.00	6.33	13.05	9.79	39.24
4 Opt.:8 Sub.	Instruct	51.64	40.55	42.27	44.82	0.00	0.08	0.00	0.03	0.00	0.00	0.00	0.00	14.95
	After FA	84.06	62.58	72.58	73.07	30.23	21.72	17.03	22.99	7.66	3.67	5.16	5.49	33.85
	After LF	83.83	61.09	71.48	72.14	28.83	21.17	18.12	22.71	7.81	3.36	5.00	5.39	33.41
	After EA	81.80	60.55	66.56	69.64	26.33	16.88	15.55	19.58	6.41	1.88	3.05	3.78	31.00
pass@8														
Teacher	Teacher FA	100.00	0.00	0.00	33.33	100.00	0.00	0.00	33.33	100.00	0.00	0.00	33.33	33.33
	Teacher LF	0.00	100.00	0.00	33.33	0.00	100.00	0.00	33.33	0.00	100.00	0.00	33.33	33.33
	Teacher EA	0.00	0.00	100.00	33.33	0.00	0.00	100.00	33.33	0.00	0.00	100.00	33.33	33.33
4 Opt.:4 Sub.(a)	Instruct	98.12	96.25	92.50	95.62	75.62	66.88	63.12	68.54	40.62	32.50	30.62	34.58	66.25
	After FA	95.00	95.62	93.12	94.58	68.12	66.88	65.62	66.88	41.25	40.62	40.62	40.83	67.43
	After LF	95.62	95.00	90.00	93.54	68.75	65.62	56.88	63.75	39.38	38.12	36.25	37.92	65.07
	After EA	91.88	95.62	90.62	92.71	60.62	58.75	58.12	59.17	32.50	28.75	41.88	34.38	62.08
4 Opt.:4 Sub.(b)	Instruct	95.62	86.88	89.38	90.62	51.25	38.12	28.12	39.17	8.12	1.88	5.00	5.00	44.93
	After FA	91.88	80.00	85.62	85.83	50.62	44.38	33.75	42.92	21.25	10.00	18.75	16.67	48.47
	After LF	91.88	78.75	88.12	86.25	51.88	41.88	36.25	43.33	21.88	10.00	21.25	17.71	49.10
	After EA	88.12	76.25	85.00	83.12	45.00	40.00	32.50	39.17	16.25	8.75	20.62	15.21	45.83
4 Opt.:8 Sub.	Instruct	70.62	61.25	67.50	66.46	0.00	0.62	0.00	0.21	0.00	0.00	0.00	0.00	22.22
	After FA	89.38	68.75	78.75	78.96	37.50	29.38	26.88	31.25	12.50	5.00	7.50	8.33	39.51
	After LF	88.12	68.75	78.75	78.54	34.38	28.75	26.25	29.79	10.62	6.88	8.12	8.54	38.96
	After EA	87.50	66.88	73.12	75.83	35.00	25.62	21.88	27.50	10.00	4.38	6.25	6.88	36.74

(2) **Cross environment generalization does not directly depend on the percentage of optimal actions in the data.** We compare the 4 Opt.:4 Sub. model with the 4 Opt.:8 Sub. model that introduces more interfering options. The Instruct model of the latter has very low base performance because a large number of low quality trajectories affect it. Its average avg@8 in Middle difficulty is only 0.03. However, after MOPD training, such as in the After FA setting, the performance of the 4 Opt.:8 Sub. model recovers greatly in all domains. Specifically, its average avg@8 in Middle difficulty successfully increases to 22.99. This shows that as long as shared planning patterns exist at the bottom level of the model, MOPD effectively extracts them and achieves cross environment generalization. This is true even if a large amount of low quality data covers these high quality patterns during the SFT stage. The generalization effect does not strictly depend on the initial percentage of high quality data.

Figure Results. As shown in Figure 17, Figure 18, and Figure 19. (1) **Consistent with previous studies, the process is mode seeking rather than mode covering.** As the distribution change graph in Figure 18 shows, the student model in the Instruct stage presents a relatively uniform distribution across various high quality planning patterns. However, after MOPD training, the probability distribution of the model does not cover all teacher preferences. Instead, it quickly shrinks to a portion of the shared patterns. For example, in Figure 18, the trained model strongly focuses on the "T1 Goal" and "T4 Reasoning" patterns in all three environments, and it does not cover the T3 pattern of the

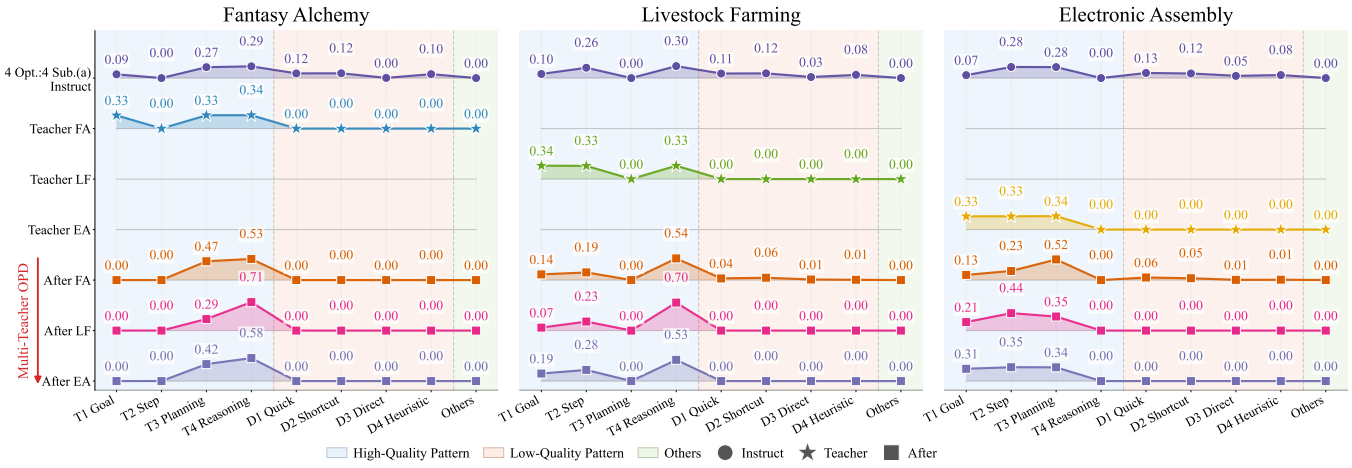


Figure 17: Planning pattern distributions for 4 Opt.:4 Sub. (a) before and after multi-teacher OPD.

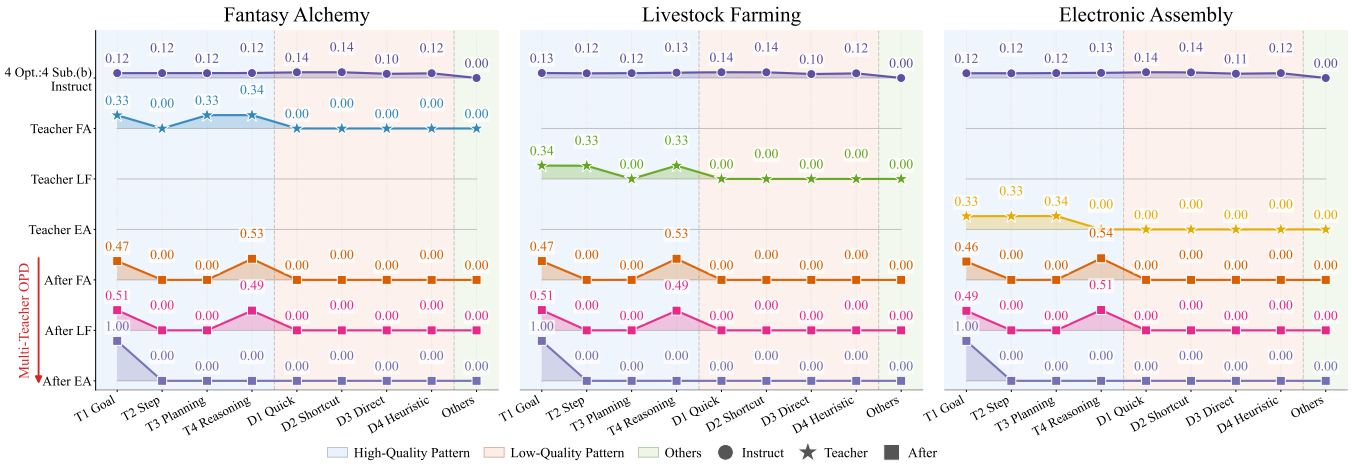


Figure 18: Planning pattern distributions for 4 Opt.:4 Sub. (b) before and after multi-teacher OPD.

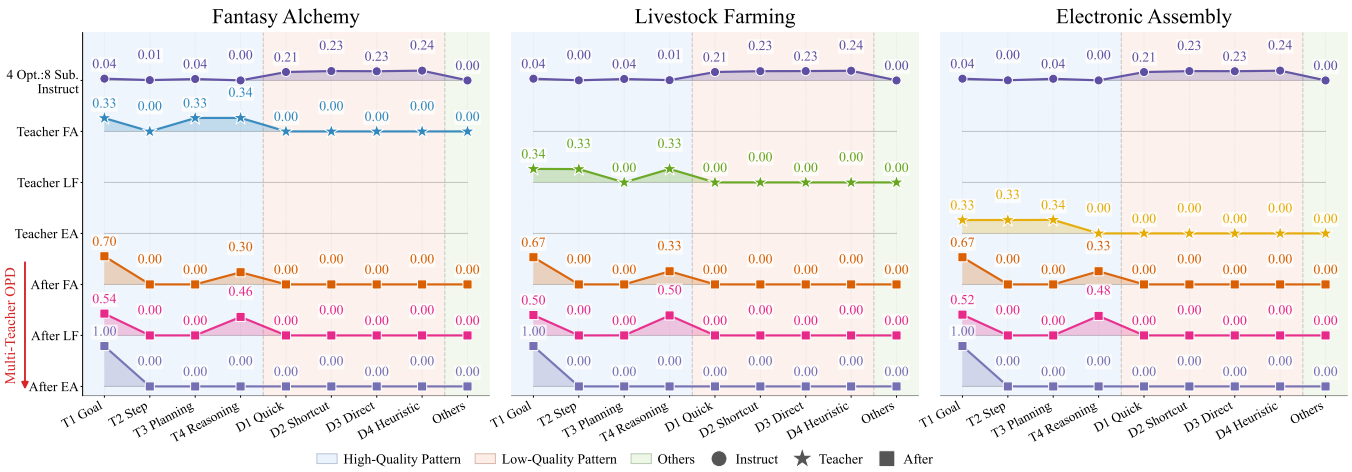


Figure 19: Planning pattern distributions for 4 Opt.:8 Sub. before and after multi-teacher OPD.

teacher. This matches the mode seeking phenomenon of the reverse KL divergence in existing OPD. (2) **MOPD is to capture the common distribution among multiple teachers as much as possible.** As shown in Figure 18 and Figure 19. Sequential distillation drives a consistent shift in the planning patterns of the student model across all domains. For example, when the student model completes the distillation from the last teacher (EA), its planning pattern distributions in Fantasy Alchemy, Livestock Farming, and Electronic Assembly all converge to the single "T1 Goal" template with a ratio of 1.00 at the same time. This shows that sequential MOPD does not cause separate local adaptations in different domains. Instead, it drives the model to find and lock onto a globally unified common distribution among the teacher strategies of each domain. (3) **Training on one environment affects the planning patterns of other environments that share the same distribution.** As shown in Figure 18, the experimental results in the first two columns reveal that the planning pattern trained in a single environment shows the same changes in other environments. For example, combining the data in the After FA row, when the model finishes training in the Fantasy Alchemy environment, the percentage of the T2 Step pattern in both the local FA environment and the unseen Livestock Farming (LF) environment shows the exact same decline. Specifically, both values drop from the baseline of 0.12 to 0.00. At the same time, the T4 Reasoning pattern in these two environments activates at the same time across domains, and both accurately reach 0.53. This indicates that as long as different environments share a consistent distribution basis in their underlying task structures, optimizing for a single environment directly causes cross domain generalization for the planning patterns in other environments. (4) **However, MOPD fails to trigger planning patterns that do not exist in the model's pre-training data for other environments.** This limitation is very clear in the T3 Planning changes in Figure 17. As the results in the After FA row show, when we introduce specific optimization in the Fantasy Alchemy environment, the model successfully increases the percentage of the T3 Planning pattern in the local FA environment from 0.27 in the Instruct stage to 0.47. However, when we apply this successful training signal from the FA environment to the Livestock Farming (LF) environment, it fails to trigger cross environment generalization for this planning pattern. This is because this planning pattern does not exist in the pretraining data of Livestock Farming. This proves that OPD cross environment generalization cannot artificially create or trigger specific planning patterns that are missing in the pretraining data of the target environment.

Takeaway

Shared and compatible planning patterns enable cross-environment generalization. When multiple environments share compatible planning patterns, MOPD can transfer the capability learned from one domain teacher to other domains, even if each teacher itself is only effective in its own environment.
MOPD performs mode seeking over the shared teacher distribution. Sequential multi-teacher OPD does not simply cover all planning patterns from all teachers. Instead, it tends to collapse toward a subset of shared high-quality planning modes, revealing that the core mechanism of MOPD is to identify and amplify the shared distribution among teachers.
Cross-environment generalization is bounded by the student's existing pattern support. MOPD can activate and strengthen planning patterns that already exist in the student's underlying distribution, even when they are diluted by low-quality SFT data. However, it cannot reliably create planning patterns that are absent from the target environment's pretraining or SFT support.

6.3. Non-Shared Conflicting Multi Teachers On-Policy Agentic Distillation

The previous section shows that MOPD generalizes across environments when teachers share compatible planning patterns. We now study the harder conflicting setting, where cross-environment generalization is no longer guaranteed. This section addresses both **Q2** and **Q3**: When non-shared teacher patterns remain compatible, MOPD can learn domain-specific abilities through continual learning without conflicts? When these patterns become incompatible, sequential distillation can overwrite earlier behaviors and induce severe cross-environment conflict.

Task Setup. (1) **Student models.** The student models use Form c and Form d as two SFT settings on the same three domains: electronic assembly, fantasy alchemy, and livestock farming. Both settings build on pretraining trajectories and contain three supervision signals: normal planning samples with gold actions, shortcut style distractor samples with wrong actions, and targeted wrong answer samples whose reasoning trace is correct but final action is wrong. (2) **Form c.** It keeps the Form b style coverage: all four normal planning templates appear in all three domains with balanced domain sampling, and all four distractor templates are also used. It chooses three domain and template

Table 7: Ensemble main table for No Shared Planning Patterns and Planning Patterns Conflict.

Pattern Mix	Model	Short				Mid				Long				Avg
		FA	LF	EA	Mean	FA	LF	EA	Mean	FA	LF	EA	Mean	S/M/L Mean
avg@8														
Teacher	Teacher FA	100.00	0.00	0.00	33.33	100.00	0.00	0.00	33.33	100.00	0.00	0.00	33.33	33.33
	Teacher LF	0.00	100.00	0.00	33.33	0.00	100.00	0.00	33.33	0.00	100.00	0.00	33.33	33.33
	Teacher EA	0.00	0.00	100.00	33.33	0.00	0.00	100.00	33.33	0.00	0.00	100.00	33.33	33.33
4 Opt.:4 Sub.(c)	Instruct	54.77	49.53	55.31	53.20	0.31	0.23	0.62	0.39	0.00	0.00	0.00	0.00	17.86
	After FA	95.23	64.61	35.47	65.10	70.62	2.89	0.00	24.51	55.16	0.00	0.00	18.39	36.00
	After LF	93.75	88.28	19.14	67.06	63.28	61.48	0.00	41.59	48.44	30.55	0.00	26.33	44.99
	After EA	91.88	88.98	89.77	90.21	58.83	57.27	66.56	60.89	47.34	29.14	42.97	39.82	63.64
4 Opt.:4 Sub.(d)	Instruct	48.52	46.09	51.48	48.70	0.08	0.31	0.00	0.13	0.00	0.00	0.00	0.00	16.28
	After FA	94.14	72.50	18.98	61.88	63.20	23.36	0.08	28.88	38.75	4.69	0.00	14.48	35.08
	After LF	90.00	84.69	18.59	64.43	55.55	59.77	0.00	38.44	33.59	28.52	0.00	20.70	41.19
	After EA	15.62	83.83	87.58	62.34	0.00	53.83	55.39	36.41	0.00	24.69	32.58	19.09	39.28
pass@8														
Teacher	Teacher FA	100.00	0.00	0.00	33.33	100.00	0.00	0.00	33.33	100.00	0.00	0.00	33.33	33.33
	Teacher LF	0.00	100.00	0.00	33.33	0.00	100.00	0.00	33.33	0.00	100.00	0.00	33.33	33.33
	Teacher EA	0.00	0.00	100.00	33.33	0.00	0.00	100.00	33.33	0.00	0.00	100.00	33.33	33.33
4 Opt.:4 Sub.(c)	Instruct	80.62	75.00	77.50	77.71	1.88	1.88	4.38	2.71	0.00	0.00	0.00	0.00	26.81
	After FA	96.25	87.50	55.00	79.58	75.62	19.38	0.00	31.67	65.00	0.00	0.00	21.67	44.31
	After LF	95.62	91.25	30.00	72.29	71.88	66.88	0.00	46.25	53.75	36.25	0.00	30.00	49.51
	After EA	94.38	92.50	91.25	92.71	68.12	64.38	72.50	68.33	54.38	35.00	48.12	45.83	68.96
4 Opt.:4 Sub.(d)	Instruct	65.62	64.38	66.88	65.62	0.62	1.88	0.00	0.83	0.00	0.00	0.00	0.00	22.15
	After FA	95.62	91.88	28.75	72.08	71.88	56.25	0.62	42.92	45.00	26.88	0.00	23.96	46.32
	After LF	93.12	88.12	28.75	70.00	67.50	63.75	0.00	43.75	40.62	35.00	0.00	25.21	46.32
	After EA	25.62	87.50	89.38	67.50	0.00	61.88	63.12	41.67	0.00	32.50	40.00	24.17	44.44

groups, template 2 in fantasy alchemy, template 3 in livestock farming, and template 4 in electronic assembly. For these groups, the answer is replaced by a wrong shortcut action while the reasoning trace remains correct. Each selected group uses three times the ordinary per-domain sample size. (3) **Form d.** It keeps the same three targeted wrong answer groups as Form c, with the same three times sampling. The main difference is that normal planning template 1 is removed. Only normal templates 2, 3, and 4 appear in all three domains. However, all four distractor templates are still kept. Thus, Form d tests missing normal template exposure while keeping full shortcut distractor coverage and the same targeted correct reasoning wrong answer signal.

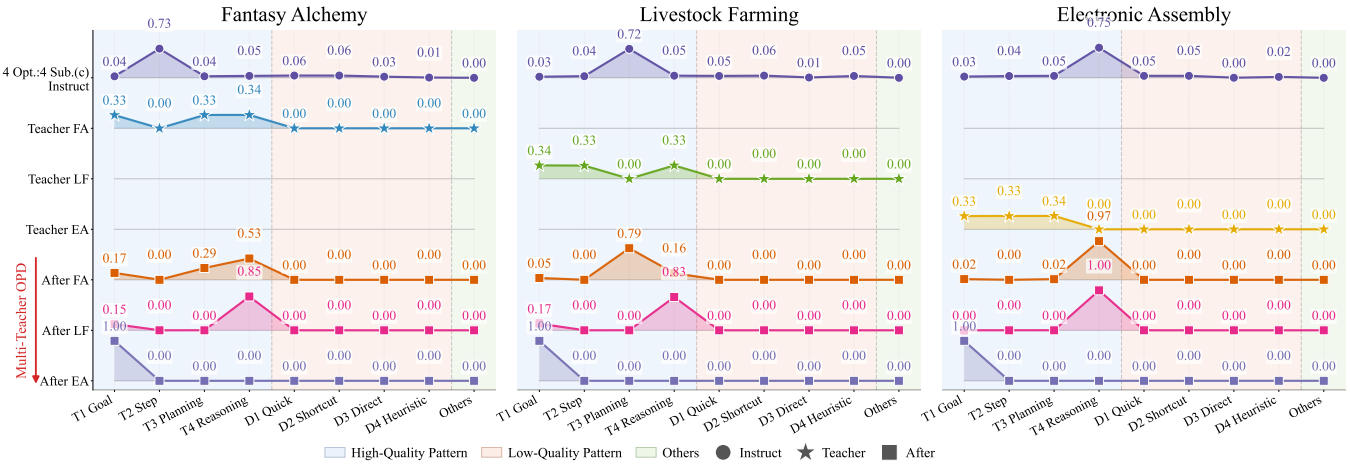


Figure 20: Planning pattern distributions for 4 Opt.:4 Sub. (c) before and after multi-teacher OPD.

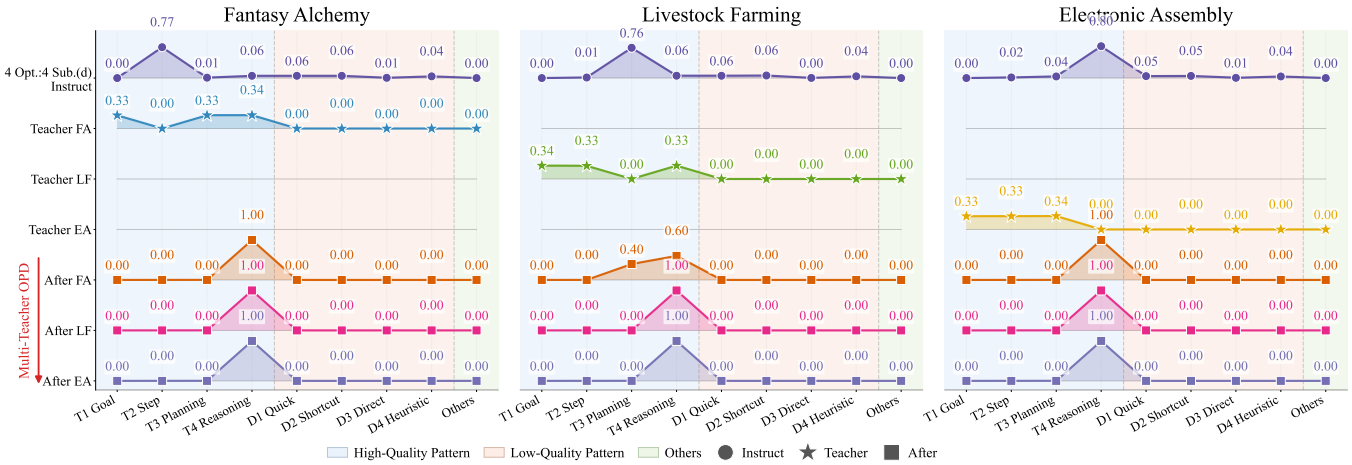


Figure 21: Planning pattern distributions for 4 Opt.:4 Sub. (d) before and after multi-teacher OPD.

Results. As shown in Table 7, Figure 20, and Figure 21. (1) **Shared and conflict patterns allow successful continual learning.** Under the Form c setting, the model successfully learns from different teachers over time without severe forgetting. Table 7 shows that after the model trains sequentially and finishes the Electronic Assembly environment, its Low difficulty avg@8 score remains high at 91.88 for the Fantasy Alchemy domain and 88.98 for the Livestock Farming domain, while it achieves 89.77 on the new Electronic Assembly domain. Figure 20 explains this stability. The agent learns to avoid domain specific traps and converges to the shared T1 Goal pattern with a 1.00 probability across all three domains. This shared compatible behavior preserves previous abilities and prevents conflicts between environments. (2) **Incompatible patterns cause severe conflicts.** Under the Form d setting, removing the shared T1 normal template forces the model into conflicts where new learning overwrites older behaviors. Table 7 highlights a massive performance drop. After the model learns from the Electronic Assembly teacher, its Low difficulty avg@8 score on the initial Livestock Farming domain crashes from 90.00 to just 15.62. Figure 21 shows the underlying reason. Without the shared pattern, the agent overfits to the new teacher and aggressively shifts to use a single conflicting pattern (T3 Planning) at a 1.00 probability everywhere. Because the remaining templates contain domain specific wrong shortcuts, applying one pattern universally triggers these traps and destroys the performance in previous domains like Fantasy Alchemy. The model has difficulty suppressing planning pattern 4, because the first two teachers already suppress patterns 2 and 3 to zero, while pattern 1 is not available during pretraining. This shows that the model learns cross-template knowledge transfer, but it also forgets much of its original knowledge.

Takeaway

Shared Planning Patterns with Planning Patterns Conflict still supports continual learning. MOPD learns from new teachers while preserving earlier domain skills when teacher patterns still share usable structure.
No Shared Planning Patterns with Planning Patterns Conflict causes severe forgetting. Experts from different environments conflict with each other, so the student tends to forget one expert after learning another.

6.4. Parameter Dynamics for MOPD

Task Setup. We study the parameter change of MOPD on three tasks, namely FA, LF, and EA. For each task, MOPD starts from the instruct model and then updates the model with the corresponding teacher signal. We compare Random, Instruct, three teacher models, and three MOPD stages. We use PCA to show the model positions in a low dimensional space. We also compute pairwise L2 distance between model parameters and cosine similarity between update directions.

Results. As shown in Figure 22. (1) **MOPD stays close to the instruct model.** In the PCA space of all models, Random and the three teacher models are far from Instruct. However, all MOPD stages stay near Instruct. The zoomed PCA plot further shows that MOPD moves in a small local region after FA, LF, and EA. This means that MOPD keeps the

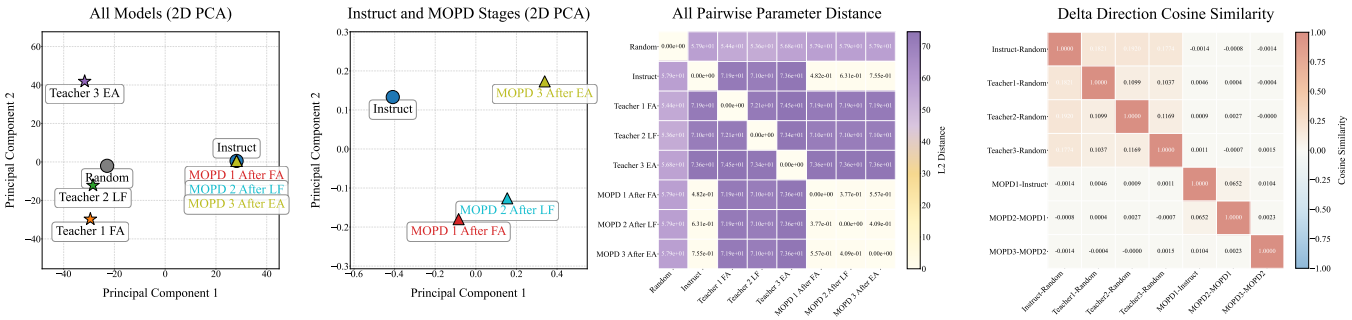


Figure 22: Parameter Dynamics for MOPD.

base model parameter stable while learning useful task signals. (2) **MOPD makes small and stable updates.** The pairwise distance matrix shows that the distance between Instruct and MOPD 1 is only 4.82×10^{-1}, and the distances between two neighboring MOPD stages are also small. By contrast, the distances from Instruct to the teacher models are around 7.10×10^{1} to 7.36×10^{1}. This gap shows that MOPD changes the model with a much smaller parameter shift. The cosine similarity matrix also shows that the update directions of different MOPD stages have low similarity. Thus, each stage adds new task information without strongly repeating the previous update direction.

Takeaway

MOPD keeps the base model parameter stable. The MOPD checkpoints remain close to the instruct model in both PCA space and parameter distance.
MOPD learns task signals with small updates. Each stage changes the parameters only slightly, and different stages bring different update directions, which explains why OPD has difficulty acquiring large-scale task-specific planning knowledge that is absent from pre-training.

7. Related Work

Long-Horizon Agentic Planning. As we are entering an excited era of long-horizon agents capable of pursuing goals over days and weeks (Li et al., 2026b, Cao et al., 2025), research in this field can be broadly categorized into two directions: benchmarks and applications. The benchmark direction includes representative efforts such as OSWorld 2.0 (Yuan et al., 2026), EdgeBench (Zhu et al., 2026a), DeepPlanning (Zhang et al., 2026b) and Long-Horizon-Terminal-Bench (Li et al., 2026f). For training, in terms of SFT data synthesis, some researchers study the synthesis of long-horizon trajectories, such as Agent-A1 (Bai et al., 2026) and Openresearcher (Li et al., 2026e). In terms of RL training algorithms, some researchers study training algorithms designed for long-horizon agents, such as SAO (Hou et al., 2026b) and TurnOPD (Zhou et al., 2026). Furthermore, agent process reward modeling serves as an important component in multi-turn long-horizon planning (Men et al., 2025, Jin et al., 2025, Fan et al., 2026, Lin et al., 2025). However, existing large language models are pretrained on large-scale and non-transparent Internet corpora, making it difficult to identify the factors behind their capability improvement. Therefore, understanding how long-horizon planning ability develops remains challenging. With controlled environments and training data splits, our study provides a more controlled analysis of the factors that enhance long-horizon planning ability.

Single-Turn and Multi-Turn OPD. OPD provides a denser supervision signal than standard GRPO and receives increasing attention (Shenfeld et al., 2025, Chen et al., 2026, Wang et al., 2026d). Current research mainly focuses on two types of OPD: single-turn OPD and multi-turn OPD. For single-turn OPD, early OPD is proposed as a knowledge distillation method (Gu et al., 2024, Agarwal et al., 2024, Lu and Lab, 2025). Recently, with the rise of RL and LLM capabilities (Song and Zheng, 2026), research directions are divided into three lines: strong-to-weak distillation (Li et al., 2026d, Fu et al., 2026, Jang et al., 2026, Li et al., 2026c, Yang et al., 2026b, Ko et al., 2026, Zhu et al., 2026b, Zhao et al., 2026a, Xing et al., 2026), self-distillation (Qu et al., 2026, Shenfeld et al., 2026, Hübotter et al., 2026, Zhao et al., 2026b, Zhang et al., 2026a, Li et al., 2026a, Yang et al., 2026a, He et al., 2026, Penaloza et al., 2026), and weak-to-strong distillation (Feng et al., 2026). Existing studies mainly focus on single-turn OPD, while multi-turn OPD

is rarely studied, and we are among the early works in this area. Existing studies mainly focus on post-training (Wang et al., 2026a, Zhou et al., 2026, Wang et al., 2026c), while opaque pre-training data limits the analysis of underlying mechanisms. We provide a unified study from pre-training to single-expert and multi-teacher distillation, covering data, algorithms, and parameter mechanisms.

Multi-Teacher OPD. Multi-Teacher OPD, as a model consolidation technique (or continual learning), plays an important role in the development of foundation models. The related research mainly focuses on two aspects: applications in different foundation models and algorithm optimization and improvement. In applications across different base models, there are mainly two uses. The first is cross-domain capability integration, including MiMo-V2-Flash (Xiao et al., 2026) and DeepSeek-V4 (Xu et al., 2026). The second is cross-stage capability retention, including GLM-5 (Zeng et al., 2026) and Nemotron-Cascade 2 (Yang et al., 2026c). Some other studies explore variants of multi-expert OPD, including CoPD (Gu et al., 2026), MAD-OPD (Wang et al., 2026b) and Uni-OPD (Hou et al., 2026a). However, existing training lacks a systematic analysis of the conditions that determine when model consolidation is effective or ineffective, limiting its practical applicability. Our work characterizes the effective boundaries of model consolidation through rigorous and controlled experimental settings with pre-trained models.

World Modeling and Compositional Generalization. For long-horizon planning, two abilities are especially important: world modeling and compositional generalization (Hao et al., 2023, Men et al., 2024, Wei et al., 2025, Gaven et al., 2025, Lyu et al., 2026). For world modeling, some works study whether internalizing a world model into existing language models can improve planning (Zhang et al., 2025b, Chen et al., 2025). In terms of compositional generalization, some works study whether language models can learn atomic abilities and spontaneously combine them into long-horizon planning abilities (Yuan et al., 2025, Sakai et al., 2025, Men et al., 2026, Zhang et al., 2025a). However, these works cannot clearly determine whether the world modeling ability and compositional generalization ability already exist in pre-training and are only activated by post-training, or whether post-training can spontaneously develop abilities that do not exist in pre-training. Our work studies these questions from the pre-training stage with clean data, which allows us to clearly analyze the origin of these abilities.

8. Conclusion

In this work, we systematically study how multi-turn long-horizon planning is acquired and improved across three training stages using a unified and controlled environment. At the pre-training stage, we show that explicit world model internalization, limited long-horizon data, and high-quality trajectories are critical for robust long-horizon planning. At the RL-based post-training stage, we distinguish between planning patterns and planning knowledge: OPD has a broader effective region than GRPO for shaping general planning patterns, whereas directly distilling unseen procedural knowledge may impair existing world modeling and out-of-domain planning. At the multi-teacher model consolidation stage, we show that MOPD integrates planning abilities by converging to shared distributions among teachers. Compatible planning patterns enable cross-environment generalization, partially shared structures support continual learning, and fully conflicting patterns lead to interference and catastrophic forgetting. Overall, our findings provide a controlled perspective on the mechanisms and boundaries of improving long-horizon planning across pre-training, RL-based post-training, and multi-teacher model consolidation.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (No.U24A20335), and the independent research project of the Key Laboratory of Cognition and Decision Intelligence for Complex Systems.

References

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

Lei Bai, Zongsheng Cao, Yang Chen, Zhiyao Cui, Shangheng Du, Yue Fan, Shiyang Feng, Zijie Guo, Haonan He, Liang He, et al. Scaling the horizon, not the parameters: Reaching trillion-parameter performance with a 35b agent. *arXiv preprint arXiv:2606.30616*, 2026.

Yuchen Cai, Ding Cao, Liang Lin, Chunxi Luo, Xin Xu, Kai Yang, Weijie Liu, Saiyong Yang, Tianxiang Zhao, Guangzhong Sun, et al. Learning to foresee: Unveiling the unlocking efficiency of on-policy distillation. *arXiv preprint arXiv:2605.11739*, 2026.

Pengfei Cao, Tianyi Men, Wencan Liu, Jingwen Zhang, Xuzhao Li, Xixun Lin, Dianbo Sui, Yanan Cao, Kang Liu, and Jun Zhao. Large language models for planning: A comprehensive and systematic survey. *arXiv preprint arXiv:2505.19683*, 2025.

Shiqi Chen, Tongyao Zhu, Zian Wang, Jinghan Zhang, Kangrui Wang, Siyang Gao, Teng Xiao, Yee Whye Teh, Junxian He, and Manling Li. Internalizing world models via self-play finetuning for agentic rl. *arXiv preprint arXiv:2510.15047*, 2025.

Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *Advances in Neural Information Processing Systems*, 38:57654–57689, 2026.

Shengda Fan, Xuyan Ye, Yupeng Huo, Zhi-Yuan Chen, Yiju Guo, Shenzhi Yang, Wenkai Yang, Shuqi Ye, Jingwen Chen, Haotian Chen, et al. Agentprocessbench: Diagnosing step-level process quality in tool-using agents. *arXiv preprint arXiv:2603.14465*, 2026.

Shiyuan Feng, Huan-ang Gao, Haohan Chi, Hanlin Wu, Zhilong Zhang, Zheng Jiang, Bingxiang He, Wei-Ying Ma, Ya-Qin Zhang, and Hao Zhou. Weak-to-strong generalization via direct on-policy distillation. *arXiv preprint arXiv:2607.05394*, 2026.

Yuqian Fu, Haohuan Huang, Kaiwen Jiang, Jiacai Liu, Zhuo Jiang, Yuanheng Zhu, and Dongbin Zhao. Revisiting on-policy distillation: Empirical failure modes and simple fixes. *arXiv preprint arXiv:2603.25562*, 2026.

Loris Gaven, Thomas Carta, Clément Romac, Cédric Colas, Sylvain Lamprier, Olivier Sigaud, and Pierre-Yves Oudeyer. Magellan: Metacognitive predictions of learning progress guide autotelic llm agents in large goal spaces. *arXiv preprint arXiv:2502.07709*, 2025.

Naibin Gu, Chenxu Yang, Qingyi Si, Chuanyu Qin, Dingyu Yao, Peng Fu, Zheng Lin, Weiping Wang, Nan Duan, and Jiaqi Wang. Co-evolving policy distillation. *arXiv preprint arXiv:2604.27083*, 2026.

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

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with language model is planning with world model. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

Yinghui He, Simran Kaur, Adithya Bhaskar, Yongjin Yang, Jiarui Liu, Narutatsu Ri, Liam Fowl, Abhishek Panigrahi, Danqi Chen, and Sanjeev Arora. Self-distillation zero: Self-revision turns binary rewards into dense supervision. *arXiv preprint arXiv:2604.12002*, 2026.

Wenjin Hou, Shangpin Peng, Weinong Wang, Zheng Ruan, Yue Zhang, Zhenglin Zhou, Mingqi Gao, Yifei Chen, Kaiqi Wang, Hongming Yang, et al. Uni-opd: Unifying on-policy distillation with a dual-perspective recipe. *arXiv preprint arXiv:2605.03677*, 2026a.

Zhenyu Hou, Yujiang Li, Jie Tang, and Yuxiao Dong. Single-rollout asynchronous optimization for agentic reinforcement learning. *arXiv preprint arXiv:2607.07508*, 2026b.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, et al. Reinforcement learning via self-distillation. *arXiv preprint arXiv:2601.20802*, 2026.

Ijun Jang, Jewon Yeom, Juan Yeo, Hyunggyu Lim, and Taesup Kim. Stable on-policy distillation through adaptive target reformulation. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 42217–42227, 2026.

Zhuoran Jin, Hongbang Yuan, Tianyi Men, Pengfei Cao, Yubo Chen, Jiexin Xu, Huaijun Li, Xiaojian Jiang, Kang Liu, and Jun Zhao. Rag-rewardbench: Benchmarking reward models in retrieval augmented generation for preference alignment. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 17061–17090, 2025.

Jongwoo Ko, Sara Abdali, Young Jin Kim, Tianyi Chen, and Pashmina Cameron. Scaling reasoning efficiently via relaxed on-policy distillation. *arXiv preprint arXiv:2603.11137*, 2026.

Gengsheng Li, Tianyu Yang, Junfeng Fang, Mingyang Song, Mao Zheng, Haiyun Guo, Dan Zhang, Jinqiao Wang, and Tat-Seng Chua. Unifying group-relative and self-distillation policy optimization via sample routing. *arXiv preprint arXiv:2604.02288*, 2026a.

Jiachun Li, Zhuoran Jin, Tianyi Men, Yupu Hao, Kejian Zhu, Lingshuai Wang, Dongqi Huang, Longxiang Wang, Shengjia Hua, Lu Wang, et al. Agentic environment engineering for large language models: A survey of environment modeling, synthesis, evaluation, and application. *arXiv preprint arXiv:2606.12191*, 2026b.

Jiaze Li, Hao Yin, Haoran Xu, Boshen Xu, Wenhui Tan, Zewen He, Jianzhong Ju, Zhenbo Luo, and Jian Luan. Video-opd: Efficient post-training of multimodal large language models for temporal video grounding via on-policy distillation. *arXiv preprint arXiv:2602.02994*, 2026c.

Yaxuan Li, Yuxin Zuo, Bingxiang He, Jinqian Zhang, Chaojun Xiao, Cheng Qian, Tianyu Yu, Huan-ang Gao, Wenkai Yang, Zhiyuan Liu, et al. Rethinking on-policy distillation of large language models: Phenomenology, mechanism, and recipe. *arXiv preprint arXiv:2604.13016*, 2026d.

Zhuofeng Li, Dongfu Jiang, Xueguang Ma, Haoxiang Zhang, Ping Nie, Yuyu Zhang, Kai Zou, Jianwen Xie, Yu Zhang, and Wenhu Chen. Openresearcher: A fully open pipeline for long-horizon deep research trajectory synthesis. *arXiv preprint arXiv:2603.20278*, 2026e.

Zongxia Li, Zhongzhi Li, Yucheng Shi, Ruhan Wang, Junyao Yang, Zhichao Liu, Xiyang Wu, Anhao Li, Yue Yu, Ninghao Liu, et al. Long-horizon-terminal-bench: Testing the limits of agents on long-horizon terminal tasks with dense reward-based grading. *arXiv preprint arXiv:2607.08964*, 2026f.

Haojia Lin, Xiaoyu Tan, Yulei Qin, Zihan Xu, Yuchen Shi, Zongyi Li, Gang Li, Shaofei Cai, Siqi Cai, Chaoyou Fu, et al. Cuarewardbench: A benchmark for evaluating reward models on computer-using agent. *arXiv preprint arXiv:2510.18596*, 2025.

Kevin Lu and Thinking Machines Lab. On-policy distillation. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tml.20251026. https://thinkingmachines.ai/blog/on-policy-distillation.

Bohan Lyu, Yucheng Yang, Siqiao Huang, Jiaru Zhang, Qixin Xu, Xinghan Li, Xinyang Han, Yicheng Zhang, Huaqing Zhang, Runhan Huang, et al. Mls-bench: A holistic and rigorous assessment of ai systems on building better ai. *arXiv preprint arXiv:2605.08678*, 2026.

Tianyi Men, Pengfei Cao, Zhuoran Jin, Yubo Chen, Kang Liu, and Jun Zhao. Unlocking the future: Exploring look-ahead planning mechanistic interpretability in large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7713–7724, 2024.

Tianyi Men, Zhuoran Jin, Pengfei Cao, Yubo Chen, Kang Liu, and Jun Zhao. Agent-rewardbench: Towards a unified benchmark for reward modeling across perception, planning, and safety in real-world multimodal agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 17521–17541, 2025.

Tianyi Men, Zhuoran Jin, Pengfei Cao, Yubo Chen, Kang Liu, and Jun Zhao. Empowering gui agents via autonomous experience exploration and hindsight experience utilization for task planning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 36090–36108, 2026.

Emiliano Penaloza, Dheeraj Vattikonda, Nicolas Gontier, Alexandre Lacoste, Laurent Charlin, and Massimo Caccia. Privileged information distillation for language models. *arXiv preprint arXiv:2602.04942*, 2026.

Yuxiao Qu, Amrith Setlur, Virginia Smith, Ruslan Salakhutdinov, and Aviral Kumar. Pope: Learning to reason on hard problems via privileged on-policy exploration. *arXiv preprint arXiv:2601.18779*, 2026.

Yusuke Sakai, Hidetaka Kamigaito, and Taro Watanabe. Revisiting compositional generalization capability of large language models considering instruction following ability. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 31219–31238, 2025.

Idan Shenfeld, Jyothish Pari, and Pulkit Agrawal. Rl's razor: Why online reinforcement learning forgets less. *arXiv preprint arXiv:2509.04259*, 2025.

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

Mingyang Song and Mao Zheng. A survey of on-policy distillation for large language models. *arXiv preprint arXiv:2604.00626*, 2026.

Hao Wang, Guozhi Wang, Han Xiao, Yufeng Zhou, Yue Pan, Jichao Wang, Ke Xu, Yafei Wen, Xiaohu Ruan, Xiaoxin Chen, et al. Skill-sd: Skill-conditioned self-distillation for multi-turn llm agents. *arXiv preprint arXiv:2604.10674*, 2026a.

Jianze Wang, Ying Liu, Jinlong Chen, Xuchun Hu, Qilong Zhang, Yu Cao, Jun Wang, Hua Yang, Yong Xie, and Qianglong Chen. Mad-opd: Breaking the ceiling in on-policy distillation via multi-agent debate. *arXiv preprint arXiv:2605.01347*, 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.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xiong-Hui Chen, Jianxin Yang, Zhenru Zhang, et al. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *Advances in Neural Information Processing Systems*, 38:115452–115486, 2026d.

Zihan Wang, Chi Gui, Xing Jin, Qineng Wang, Licheng Liu, Kangrui Wang, Shiqi Chen, Linjie Li, Zhengyuan Yang, Pingyue Zhang, et al. Ragen-2: Reasoning collapse in agentic rl. *arXiv preprint arXiv:2604.06268*, 2026e.

Hui Wei, Zihao Zhang, Shenghua He, Tian Xia, Shijia Pan, and Fei Liu. Plangenllms: A modern survey of llm planning capabilities. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 19497–19521, 2025.

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.

Xingrun Xing, Haoqing Wang, Boyan Gao, Ziheng Li, and Yehui Tang. Trust region on-policy distillation. *arXiv preprint arXiv:2606.01249*, 2026.

Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

Chenxu Yang, Chuanyu Qin, Qingyi Si, Minghui Chen, Naibin Gu, Dingyu Yao, Zheng Lin, Weiping Wang, Jiaqi Wang, and Nan Duan. Self-distilled rlvr. *arXiv preprint arXiv:2604.03128*, 2026a.

Wenkai Yang, Weijie Liu, Ruobing Xie, Kai Yang, Saiyong Yang, and Yankai Lin. Learning beyond teacher: Generalized on-policy distillation with reward extrapolation. *arXiv preprint arXiv:2602.12125*, 2026b.

Zhuolin Yang, Zihan Liu, Yang Chen, Wenliang Dai, Boxin Wang, Sheng-Chieh Lin, Chankyu Lee, Yangyi Chen, Dongfu Jiang, Jiafan He, et al. Nemotron-cascade 2: Post-training llms with cascade rl and multi-domain on-policy distillation. *arXiv preprint arXiv:2603.19220*, 2026c.

Tian Ye, Zicheng Xu, Yuanzhi Li, and Zeyuan Allen-Zhu. Physics of language models: Part 2.1, grade-school math and the hidden reasoning process. In *International Conference on Learning Representations*, volume 2025, pages 97699–97709, 2025.

Lifan Yuan, Weize Chen, Yuchen Zhang, Ganqu Cui, Hanbin Wang, Ziming You, Ning Ding, Zhiyuan Liu, Maosong Sun, and Hao Peng. From $f(x)$ and $g(x)$ to $f(g(x))$: Llms learn new skills in rl by composing old ones. *arXiv preprint arXiv:2509.25123*, 2025.

Mengqi Yuan, Zilong Zhou, Xinzhuang Xiong, Weiming Wu, Jiayang Sun, Jiamin Song, Kaiqian Cui, Bowen Wang, Haoyuan Wu, Yitong Li, et al. Osworld2. 0: Benchmarking computer use agents on long-horizon real-world tasks. *arXiv preprint arXiv:2606.29537*, 2026.

Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, et al. Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026.

Charlie Zhang, Graham Neubig, and Xiang Yue. On the interplay of pre-training, mid-training, and rl on reasoning language models. *arXiv preprint arXiv:2512.07783*, 2025a.

Kai Zhang, Xiangchao Chen, Bo Liu, Tianci Xue, Zeyi Liao, Zhihan Liu, Xiyao Wang, Yuting Ning, Zhaorun Chen, Xiaohan Fu, et al. Agent learning via early experience. *arXiv preprint arXiv:2510.08558*, 2025b.

Ruixiang Zhang, Richard He Bai, Huangjie Zheng, Navdeep Jaitly, Ronan Collobert, and Yizhe Zhang. Embarrassingly simple self-distillation improves code generation. *arXiv preprint arXiv:2604.01193*, 2026a.

Yinger Zhang, Shutong Jiang, Renhao Li, Jianhong Tu, Yang Su, Lianghao Deng, Xudong Guo, Chenxu Lv, and Junyang Lin. Deepplanning: Benchmarking long-horizon agentic planning with verifiable constraints. *arXiv preprint arXiv:2601.18137*, 2026b.

Anhao Zhao, Haoran Xin, Yingqi Fan, Junlong Tong, Wenjie Li, and Xiaoyu Shen. Decoupling kl and trajectories: A unified perspective for sft, dagger, offline rl, and opd in llm distillation. *arXiv preprint arXiv:2605.16826*, 2026a.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. *arXiv preprint arXiv:2601.18734*, 2026b.

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.

Deyao Zhu, Xin Zhou, Shengling Qin, Xuekai Zhu, Hangliang Ding, Shu Zhong, Zixin Wen, Zhonglin Xie, Chenhui Gou, Linxuan Ren, et al. Edgebench: Unveiling scaling laws of learning from real-world environments. *arXiv preprint arXiv:2607.05155*, 2026a.

Siqi Zhu, Xuyan Ye, Hongyu Lu, Weiye Shi, and Ge Liu. The many faces of on-policy distillation: Pitfalls, mechanisms, and fixes. *arXiv preprint arXiv:2605.11182*, 2026b.

Appendix Table of Contents

A. Pre-training Details

A.1. Pre-trained Model Configuration

This section details the specific architectural design and tokenizer configuration for the randomly initialized language model utilized in our study. The model is built following the `Qwen2ForCausalLM` architecture specifications. In total, this configuration yields a compact model with approximately 100 million (100M) parameters. We call it Qwen2.5-100M. Table 8 lists the detailed network hyperparameters, and Table 9 summarizes the configuration properties of the accompanying byte-level BPE tokenizer. Finally, it should be noted that the tokenizer is trained on the corresponding full pre-training corpus. Because this corpus varies slightly across different chapters, we perform tokenization separately for each. This variation does not affect our overall experimental conclusions.

Table 8: Pre-trained model architectural hyperparameters.

Hyperparameter	Value
Model Type	`qwen2`
Architecture	`Qwen2ForCausalLM`
Number of Hidden Layers	12
Hidden Size (d_{model})	768
Intermediate Size (d_{ff})	3072
Number of Attention Heads (n_{head})	12
Number of Key-Value Heads ($n_{\text{kv\_head}}$)	2 (Grouped-Query Attention)
Attention Head Dimension	64
Hidden Activation Function	SiLU
Attention Dropout	0.0
RMS Norm Epsilon (ϵ)	1×10^{-6}
RoPE Theta (θ)	$1,000,000.0$
Max Position Embeddings	2048
Vocabulary Size (V)	3000
Tie Word Embeddings	True
Data Type (`dtype`)	`bfloat16`

A.2. Pre-training Schema Configuration

The pretraining corpus is constructed over three distinct domains, namely Fantasy Alchemy, Livestock Farming, and Electronic Assembly. As summarized in Table 10, each domain contains 200 item nodes and a total of 4000 item names. The task space is organized into 5 hierarchical levels, with each level comprising 40 item categories and 20 instances per category.

Each item category is defined by a compositional schema, where long-level items are recursively generated from shorter-level components through logical conjunction (AND) and disjunction (OR) operators. In particular, the input

Table 9: Tokenizer configuration and training details.

Tokenizer Property	Value
Tokenizer Type	`byte_level_bpe+regex`
Vocabulary Size	3000
Minimum Token Frequency	2
Limit Alphabet	1000
Maximum Token Length	100
Byte-level Add Prefix Space	False
BOS Token	`[BOS]`
EOS Token	`[EOS]`
Special Tokens	`[UNK]`, `[PAD]`, `[BOS]`, `[EOS]`, `<question>`, `</question>`, `<solution>`, `</solution>`, `<answer>`, `</answer>`

Table 10: Pretraining schema statistics across three domains.

Metric	Fantasy Alchemy	Livestock Farming	Electronic Assembly
Total nodes	200	200	200
Total item names	4000	4000	4000
One-input ratio	30.00%	30.00%	30.00%
Two-input ratio	70.00%	70.00%	70.00%
Avg atoms for target L2 from source L1	1.82	1.82	1.82
Avg atoms for target L3 from source L1	3.08	3.08	3.08
Avg atoms for target L3 from source L2	1.68	1.68	1.68
Avg atoms for target L4 from source L1	4.17	4.17	4.17
Avg atoms for target L4 from source L2	2.23	2.23	2.23
Avg atoms for target L4 from source L3	1.65	1.65	1.65
Avg atoms for target L5 from source L1	5.78	5.78	5.78
Avg atoms for target L5 from source L2	3.15	3.15	3.15
Avg atoms for target L5 from source L3	2.38	2.38	2.38
Avg atoms for target L5 from source L4	1.65	1.65	1.65

degree distribution is controlled to maintain a fixed structural prior, with 30% of items following a one-input dependency and 70% requiring two-input compositions. This induces a sparse but non-trivial dependency graph over items.

The compositional depth increases with level, leading to progressively more complex construction paths from base materials. As shown in Table 10, the expected number of atomic components required to construct target items grows with hierarchy depth, e.g., from Level 1 to Level 2 requiring on average 1.82 atoms, while Level 5 items require up to 5.78 atoms when traced back to Level 1 sources.

To further increase task difficulty and robustness, each planning instance is constructed by mixing target-relevant items with additional noise items, preventing trivial pattern matching and encouraging the model to recover correct compositional structure under partial observability. The resulting environment thus provides a controlled benchmark for evaluating long-horizon compositional planning under structured but noisy generation conditions.

A.3. Pre-training Configuration

A.4. Pre-training Corpus Configuration

We construct the datasets from three synthetic domains: electronic assembly, livestock farming, and fantasy alchemy. For each domain, the dataset builder reads a shared item configuration, a list of concrete instantiations, and two recipe schemas. The two schemas define two planning environments, denoted as split A and split B. Each instantiation maps every abstract item node to one concrete item name, so split A and split B share the same item names but use different recipe graphs. We adopt a counterfactual formulation of the state transition function for scalability. This reformulation follows the same environment transition rules and produces the same observable interaction outcomes, so it does not

affect the empirical results.

For each concrete instantiation, the builder enumerates every non base target item and every feasible starting layer below the target layer. It recursively collects the materials that are required by the A recipe graph and the B recipe graph at the selected starting layer. The initial inventory contains the union of these required materials, together with two to four distractor items sampled from the same starting layer. The target item, the input inventory, the required materials, the distractors, and the gold plan are then stored as one sample. The gold plan is obtained by recursively expanding the target item according to the corresponding recipe graph until all required intermediate items are produced. The number of actions in this gold plan defines the field total step nums.

The first 20 concrete instantiations form the test pool. Instantiations with indices from 20 to 219 form the post training pool. All remaining instantiations form the pretraining pool. The post training and test samples are further grouped by plan length. The short group contains samples with one to five steps, the mid group contains samples with six to eight steps, and the long group contains samples with at least nine steps. The teacher_all pretraining set contains all generated samples for a split and a domain, including samples from all instantiation ranges, and serves as the full teacher data source.

As shown in Table 11. Table rows follow the file naming convention used by the dataset builder. The row label Category gives the synthetic domain. The row label Split summarizes the recipe split and dataset subset. The token A or B identifies which recipe graph produces the gold trajectory. The token pretrain denotes samples from the pretraining pool, posttrain denotes samples from the post training pool, and test denotes held out evaluation samples. The token teacher_all denotes the full teacher data source for the corresponding split and domain with both train and test datasets. The tokens short, mid, and long denote the step based difficulty groups. The token final, when present, denotes the final evaluation copy of the corresponding test subset. The column Samples reports the number of samples in the row. The columns Avg, Min, and Max report the mean, minimum, and maximum gold plan length. The columns named Step report the number of samples with each exact gold plan length.

As summarized in Table 12, our SFT corpus consists of student and teacher datasets with distinct characteristics. The teacher datasets act as domain experts, encompassing both the training and post-training sets. Each teacher dataset contains exactly 622,960 samples, specializing exclusively in one of three fields (fantasy alchemy, livestock farming, and electronic assembly) while covering three difficulty levels. In contrast, the student datasets broadly encompass all three domains and are constructed using specific sampling ratios across the three difficulty tiers (short, mid, and long), as reflected in their dataset names, yielding sample sizes that scale from roughly 322K to 1.07M. Additionally, both student and teacher datasets include subsets with and without CoT reasoning. Token statistics highlight that CoT subsets produce significantly longer outputs (averaging 186-230 tokens) compared to the direct answers in non-CoT subsets (averaging $\sim$30 tokens). Meanwhile, the average input lengths remain relatively stable across all datasets, ranging from 264 to 340 tokens.

All experiments are conducted with 2 GPUs training and share the same optimization setup: a per-device training batch size of 128, 2 gradient accumulation steps, a learning rate of 1.0e-4 with a cosine scheduler, a training steps of 9000, a warmup ratio of 0.1, and bf16 precision enabled. We use full-parameter SFT on the same base model (*Qwen2.5-100M*).

Table 11: Statistics of pretrain, posttrain, and test datasets.

Category	Split	Samples	Avg	Min	Max	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6	Step 7	Step 8	Step 9	Step 10	Step 11	Step 12	Step 13
Pretrain																		
electronic assembly	A / pretrain	120000	3.00	1	12	48000	12300	26400	5700	8100	10200	1500	2100	2100	3000	300	300	0
	A / pretrain / teacher_all	208000	3.00	1	12	83200	21320	45760	9880	14040	17680	2600	3640	3640	5200	520	520	0
	B / pretrain	120000	3.01	1	13	48000	11700	27300	6600	6900	8100	4500	1200	2400	1500	900	600	300
	B / pretrain / teacher_all	208000	3.01	1	13	83200	20280	47320	11440	11960	14040	7800	2080	4160	2600	1560	1040	520
fantasy alchemy	A / pretrain	120000	3.00	1	12	48000	12300	26400	5700	8100	10200	1500	2100	2100	3000	300	300	0
	A / pretrain / teacher_all	208000	3.00	1	12	83200	21320	45760	9880	14040	17680	2600	3640	3640	5200	520	520	0
	B / pretrain	120000	3.01	1	13	48000	11700	27300	6600	6900	8100	4500	1200	2400	1500	900	600	300
	B / pretrain / teacher_all	208000	3.01	1	13	83200	20280	47320	11440	11960	14040	7800	2080	4160	2600	1560	1040	520
livestock farming	A / pretrain	120000	3.00	1	12	48000	12300	26400	5700	8100	10200	1500	2100	2100	3000	300	300	0
	A / pretrain / teacher_all	208000	3.00	1	12	83200	21320	45760	9880	14040	17680	2600	3640	3640	5200	520	520	0
	B / pretrain	120000	3.01	1	13	48000	11700	27300	6600	6900	8100	4500	1200	2400	1500	900	600	300
	B / pretrain / teacher_all	208000	3.01	1	13	83200	20280	47320	11440	11960	14040	7800	2080	4160	2600	1560	1040	520
Posttrain																		
electronic assembly	A / short	67000	2.14	1	5	32000	8200	17600	3800	5400	0	0	0	0	0	0	0	0
	B / short	67000	2.13	1	5	32000	7800	18200	4400	4600	0	0	0	0	0	0	0	0
	A / mid	9200	6.41	6	8	0	0	0	0	0	6800	1000	1400	0	0	0	0	0
	B / mid	9200	6.50	6	8	0	0	0	0	0	5400	3000	800	0	0	0	0	0
	A / long	3800	9.79	9	12	0	0	0	0	0	0	0	0	1400	2000	200	200	0
	B / long	3800	10.11	9	13	0	0	0	0	0	0	0	0	1600	1000	600	400	200
fantasy alchemy	A / short	67000	2.14	1	5	32000	8200	17600	3800	5400	0	0	0	0	0	0	0	0
	B / short	67000	2.13	1	5	32000	7800	18200	4400	4600	0	0	0	0	0	0	0	0
	A / mid	9200	6.41	6	8	0	0	0	0	0	6800	1000	1400	0	0	0	0	0
	B / mid	9200	6.50	6	8	0	0	0	0	0	5400	3000	800	0	0	0	0	0
	A / long	3800	9.79	9	12	0	0	0	0	0	0	0	0	1400	2000	200	200	0
	B / long	3800	10.11	9	13	0	0	0	0	0	0	0	0	1600	1000	600	400	200
livestock farming	A / short	67000	2.14	1	5	32000	8200	17600	3800	5400	0	0	0	0	0	0	0	0
	B / short	67000	2.13	1	5	32000	7800	18200	4400	4600	0	0	0	0	0	0	0	0
	A / mid	9200	6.41	6	8	0	0	0	0	0	6800	1000	1400	0	0	0	0	0
	B / mid	9200	6.50	6	8	0	0	0	0	0	5400	3000	800	0	0	0	0	0
	A / long	3800	9.79	9	12	0	0	0	0	0	0	0	0	1400	2000	200	200	0
	B / long	3800	10.11	9	13	0	0	0	0	0	0	0	0	1600	1000	600	400	200
Test																		
electronic assembly	A / short	6700	2.14	1	5	3200	820	1760	380	540	0	0	0	0	0	0	0	0
	B / short	6700	2.13	1	5	3200	780	1820	440	460	0	0	0	0	0	0	0	0
	A / mid	920	6.41	6	8	0	0	0	0	0	680	100	140	0	0	0	0	0
	B / mid	920	6.50	6	8	0	0	0	0	0	540	300	80	0	0	0	0	0
	A / long	380	9.79	9	12	0	0	0	0	0	0	0	0	140	200	20	20	0
	B / long	380	10.11	9	13	0	0	0	0	0	0	0	0	160	100	60	40	20
fantasy alchemy	A / short	6700	2.14	1	5	3200	820	1760	380	540	0	0	0	0	0	0	0	0
	B / short	6700	2.13	1	5	3200	780	1820	440	460	0	0	0	0	0	0	0	0
	A / mid	920	6.41	6	8	0	0	0	0	0	680	100	140	0	0	0	0	0
	B / mid	920	6.50	6	8	0	0	0	0	0	540	300	80	0	0	0	0	0
	A / long	380	9.79	9	12	0	0	0	0	0	0	0	0	140	200	20	20	0
	B / long	380	10.11	9	13	0	0	0	0	0	0	0	0	160	100	60	40	20
livestock farming	A / short	6700	2.14	1	5	3200	820	1760	380	540	0	0	0	0	0	0	0	0
	B / short	6700	2.13	1	5	3200	780	1820	440	460	0	0	0	0	0	0	0	0
	A / mid	920	6.41	6	8	0	0	0	0	0	680	100	140	0	0	0	0	0
	B / mid	920	6.50	6	8	0	0	0	0	0	540	300	80	0	0	0	0	0
	A / long	380	9.79	9	12	0	0	0	0	0	0	0	0	140	200	20	20	0
	B / long	380	10.11	9	13	0	0	0	0	0	0	0	0	160	100	60	40	20

Table 12: Estimated token statistics of SFT datasets.

Dataset	#Samples	Input Tokens	Output Tokens	Avg Input	Avg Output
Student CoT, short50% mid0% long0%	322,711	85,921,610	60,195,896	266.25	186.53
Student CoT, short100% mid0% long0%	645,300	170,851,435	120,079,424	264.76	186.08
Student CoT, short100% mid5% long0%	658,591	176,507,591	123,342,635	268.01	187.28
Student CoT, short100% mid50% long0%	778,129	220,910,979	153,718,139	283.90	197.55
Student CoT, short100% mid100% long0%	910,800	272,430,026	189,074,794	299.11	207.59
Student CoT, short100% mid100% long5%	919,146	277,076,378	191,666,574	301.45	208.53
Student CoT, short100% mid100% long50%	994,512	310,958,940	215,904,279	312.67	217.10
Student CoT, short100% mid100% long100%	1,078,200	349,875,900	240,656,936	324.50	223.20
Student w/o CoT, short100% mid100% long100%	1,078,200	350,036,228	32,182,761	324.65	29.85
Teacher CoT, Fantasy Alchemy	622,960	211,826,833	142,973,369	340.03	229.51
Teacher CoT, Livestock Farming	622,960	201,144,813	140,325,789	322.89	225.26
Teacher CoT, Electronic Assembly	622,960	192,900,810	133,524,250	309.65	214.34
Teacher w/o CoT, Fantasy Alchemy	622,960	211,628,047	18,799,998	339.71	30.18
Teacher w/o CoT, Livestock Farming	622,960	199,517,081	18,476,308	320.27	29.66
Teacher w/o CoT, Electronic Assembly	622,960	193,367,781	18,393,766	310.40	29.53