# HyperSkill: Training-Free Omnimodal GRPO via Hypergraph-Indexed Skill-Library Evolution

**Haoran Luo**[1†], **Shangkai Lin**[1], **Jinyang Wu**[2], **Xinliang Zhou**[3*], **Luu Anh Tuan**[1,4*]
[1]Nanyang Technological University [2]Tsinghua University [3]Stanford University [4]VinUniversity
haoran.luo@ntu.edu.sg, xinliang.zhou@stanford.edu, anhtuan.luu@ntu.edu.sg

## Abstract

Reinforcement learning sharpens LLM reasoning, but parameter updates are expensive, forgetful, and opaque; training-free libraries swap gradients for a textual store but still use trajectory-level, correlation-based admission and flat structure. To address both, we propose **HyperSkill**, a *Training-Free Omnimodal GRPO* framework that keeps the backbone frozen and replaces gradient updates with auditable edits to a *Hypergraph-Indexed Skill-Library*, where skills are advantage-bearing nodes and compositions are hyperedges. It has three components: training-time distillation admits counterfactually checked skills and learns a protocol skill from failures; gated hypergraph retrieval injects advantage-ranked skills where a per-dataset gate admits them; and an execution protocol makes the frozen backbone commit to an answer. On 11 text, vision-language, and omnimodal benchmarks, HyperSkill is best or tied on every dataset and beats the best baseline by 13.0 average F1 points while keeping the backbone frozen. Our software and data are publicly available[1].

## 1 Introduction

Reinforcement learning is now the main way to improve the reasoning of large language models (LLMs), with group-relative policy optimisation (GRPO) [39, 52, 55] used to train systems in math, search, and tool use [16, 49, 20, 12]. However, it is still hard to keep improving an agent as it moves to omnimodal tasks such as web search, long-document reasoning, and audio-video question answering [45, 48, 58, 25], without a per-domain cost of updating its backbone.

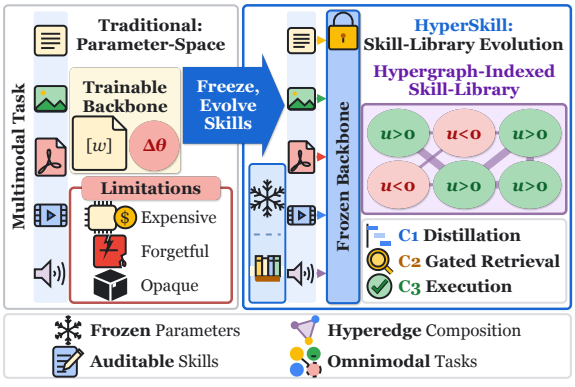


Figure 1: Parameter-space GRPO is costly, forgetful and opaque; HyperSkill instead keeps the backbone frozen and evolves a hypergraph-indexed skill library.

Existing methods follow two main lines. The mainstream parameter-space recipe applies gradient updates to the backbone, yielding capable but costly systems [16, 49, 20, 12]. A growing training-free line keeps the backbone frozen and grows an external textual experience store across rollouts, retrieved in-context at inference [3, 2, 4, 50, 41, 44].

Both lines have limits. On the parameter side (Figure 1), gradient updates remain expensive at frontier scale [16], every new task risks catastrophic forgetting [54], and improvement is buried in opaque weight deltas with no auditable trace of *which* patterns the agent has internalised. On the training-free side, three failure modes recur (Figure 2): (i) comparison is at the trajectory level [3, 2], so every

[†]Project Lead. [*]Corresponding Authors. [1]https://github.com/LHRLAB/HyperSkill

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

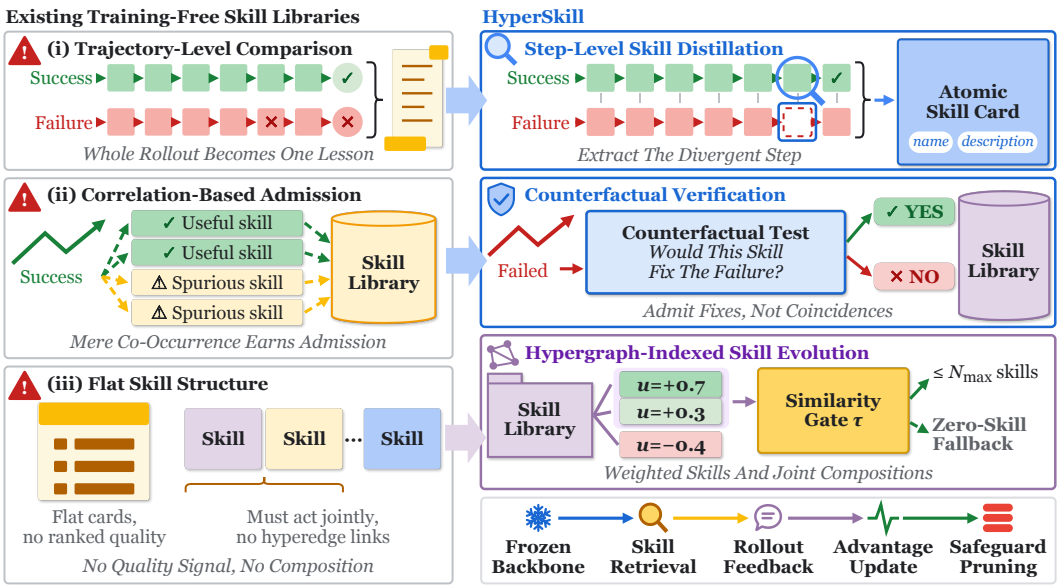


Figure 2: Three failure modes (i)–(iii) of training-free libraries (trajectory-level, correlation-based, flat skills) vs. HyperSkill's step distillation, counterfactual verification and hypergraph evolution.

divergent step is fused into one lesson; (ii) candidates are admitted by co-occurrence with successful rollouts [3, 4, 50], so the library keeps patterns that track reward but do not fix the failure; (iii) skills are stored as dictionaries or uniform-quality lists, which cannot express that *several skills must act jointly* or give a per-skill quality signal. Hypergraph retrieval [28, 13, 53] links several entities in one hyperedge, but it targets static factual knowledge bases rather than self-evolving skill libraries.

To close all three gaps, we propose HyperSkill, a *Training-Free Omnimodal GRPO* framework whose policy is an external *Hypergraph-Indexed Skill-Library*: atomic skills are advantage-bearing nodes, joint compositions are hyperedges, and the library is the optimisation target. **C1** (training-time distillation) handles (i) and (ii): it admits a step-level candidate only after a counterfactual test ("would the failed trajectory have succeeded had the agent known this skill?") and learns a protocol skill from failures. **C2** (gated hypergraph retrieval) handles (iii): an advantage-weighted retriever [22, 15, 17] ranks hyperedge-linked skills [28, 13, 27] behind a gate with zero-skill fallback. **C3** (execution protocol) makes the frozen backbone call its tools correctly and then commit to a final answer.

We validate this design across 11 text, vision-language, and omnimodal benchmarks [45, 5, 29, 18]: HyperSkill is best or tied on all 11 without updating a parameter, and lifts the average F1 from 29.0 for the strongest baseline, XSkill [19], to 42.0 (+13.0, 44.9% relative). Each component matters: removing C1, C2 or C3 alone lowers the average F1 by 2.5, 4.2 and 10.4 points, respectively.

## 2 Related Work

**Omnimodal LLMs and reinforcement learning.** Modern backbones fuse text, image, video, and audio into a single policy [35, 26, 36, 1, 48, 58, 25], and group-relative policy optimisation [39, 52, 55] trains reasoners [16, 11] and search and tool agents [20, 12, 27], while reflection-style agents [51, 40, 30] act through prompting on web and OS benchmarks [57, 45, 47]. The RL-trained ones still pay for backbone gradient updates and inherit the forgetting and opacity problems motivating this work. *Training-Free Omnimodal GRPO* keeps the backbone frozen and redirects a normalised reward advantage from the gradient onto an external *Hypergraph-Indexed Skill-Library* that acts as the policy, so each improvement is a readable library edit rather than an opaque weight delta.

**Training-free experience self-evolution.** TrainingFreeGRPO [3] replaces the gradient with an LLM-curated experience pool, skill-centric agents [2, 4, 50, 41, 44] grow flat dictionaries or frontier sets, and memory [33, 7, 34] with graph and hypergraph RAG [10, 17, 28, 13] supplies the retrievable side. Most of them compare *whole trajectories* and keep flat skills with no per-item quality for ranking; HyperSkill instead distils individual steps and ranks skills by advantage in a hypergraph.

Table 1: Comparison of training-free skill methods. TrainingFreeGRPO: token-prior experience pool; EvoSkill: Pareto frontier of skill folders; SkillCraft/AutoSkill: composed-tool code or interaction-trace cards; Memento-Skills/XSkill: markdown router memory or dual experience-skill stream; HyperSkill (Ours): weighted hypergraph with counterfactual distillation and gated RRF.

| **Method** | **Skill Organisation** | **Retrieval & Update** |
|---|---|---|
| TrainingFreeGRPO [3] | $\mathcal{L} = \{x_i\}_{i=1}^{\lvert\mathcal{L}\rvert}$<br>Experiential token-prior pool from rollouts. | $\mathcal{R}(q) = \text{Top-}k_{x_i}\ \cos(f(q), f(x_i))$<br>LLM CRUD via group-relative semantic adv. |
| EvoSkill [2] | $\mathcal{L}_{\text{frt}} = \text{Pareto}\{\text{folder}_i\}$<br>Pareto frontier of structured skill folders. | $\mathcal{R}(q) = \text{Top-}k_{s_i \in \mathcal{L}_{\text{frt}}}\ \cos$<br>Failure-driven propose, holdout-validate. |
| SkillCraft [4] | $\mathcal{L} = \{\texttt{def}\ s_i(\cdot)\}$<br>Auto-composed tool-Skill code cache. | $\mathcal{R}(q) = \text{Top-}k_{s_i}\ \cos$<br>Cache-hit reuse, no quality signal. |
| AutoSkill [50] | $\mathcal{L} = \{\text{card}_i\}$<br>Standardised cards from interaction traces. | $\mathcal{R}(q) = \text{Top-}k_{s_i}\ \cos$<br>Dynamic injection, self-evolving lifelong. |
| Memento-Skills [56] | $\mathcal{L} = \{m_i^{\text{md}}\}$<br>Markdown skills as stateful memory. | $\mathcal{R}(q) = \text{Router}(q, \mathcal{L})$<br>Read/write reflective skill router. |
| XSkill [19] | $\{\mathcal{X}_{\text{exp}}, \mathcal{L}_{\text{skill}}\}$<br>Dual experience and skill streams. | $\mathcal{R}(q, I) = \text{Top-}k_{(\mathcal{X},\mathcal{L})}\ \cos$<br>Vision-grounded retrieve + cross-critique. |
| **HyperSkill (Ours)** | $\mathcal{H} = (\mathcal{V}, \mathcal{E}, w),\ u_i \in [-1, 1]$<br>Weighted hypergraph with hyperedges. | $\mathcal{R}(q) = \text{RRF}_{u_i}(\mathcal{V}, \mathcal{E}) \circ \text{Gate}_{\tau, N_{\max}}$<br>Counterfactual admit + 0-skill fallback. |

## 3 Methodology: HyperSkill

**Problem Formulation.** Unlike the methods of Table 1, HyperSkill couples a per-skill quality signal with explicit multi-skill compositions in a weighted skill hypergraph. Formally, let $\pi_\theta$ be a frozen omnimodal backbone whose parameters $\theta$ are never updated, and let $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$ be train and test sets of triples $(q, a^*, m)$, in which each query $q$ carries a gold answer $a^*$ and a modality tag $m \in \mathcal{M} = \{\text{text}, \text{image}, \text{pdf}, \text{video}, \text{audio}\}$. We seek a weighted skill hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E}, w)$ in which every node $v_i \in \mathcal{V}$ carries a name, a description and an advantage $u_i \in [-1, 1]$, and every hyperedge $e_j \in \mathcal{E}$ groups $|e_j| \geq 2$ nodes into a joint composition with weight $w(e_j)$. We also seek a retriever $\mathcal{R} : (q, m, \mathcal{H}) \to \mathcal{S}_q \subset \mathcal{V}$ under which the skill-augmented policy improves on its skill-free version, while a hypergraph update $\mathcal{U}$ replaces the gradient step in every round $t$:

$$\big(\mathcal{H}^\star,\, \mathcal{R}^\star\big) \;=\; \underset{\mathcal{H},\,\mathcal{R}}{\text{argmax}}\ \mathbb{E}_{(q,a^*,m)\sim\mathcal{D}}\Big[\, r\big(\,\pi_\theta(\cdot \mid q, m, \mathcal{R}(q, m, \mathcal{H}))\,;\, a^*\,\big)\,\Big], \quad \mathcal{H}_{t+1} = \mathcal{U}(\mathcal{H}_t; \mathcal{B}_t)\,, \tag{1}$$

where $\mathcal{D}$ is the task distribution ($\mathcal{D}_{\text{train}}$ in training, $\mathcal{D}_{\text{test}}$ at evaluation), $\mathcal{B}_t$ the round-$t$ training batch and $r$ the token-level $F_1$ reward against $a^*$. Unlike GRPO, the optimisation variable is the pair $(\mathcal{H}, \mathcal{R})$ rather than $\theta$: gradient updates are replaced by an iterative, auditable rewrite of the library. C1 (§3.1) handles gaps (i) and (ii), C2 (§3.2) handles gap (iii), and C3 (§3.3) fixes execution failures of the frozen backbone; Figure 3 gives an overview of how C1 and C2 build, evolve and query the library.

### 3.1 C1: Training-Time Skill Distillation

To address (i) and (ii), we move admission from *trajectories* to *steps* and route every candidate through a counterfactual probe before insertion, so that a skill enters $\mathcal{V}$ only if the frozen backbone judges it to salvage a failure of its own question rather than merely co-occurring with a success.

**Contrastive Step Alignment.** For every question $q \in \mathcal{B}_t$ we draw a group of $n$ skill-augmented rollouts $\{o^{(\ell)}\}_{\ell=1}^n \sim \pi_\theta(\cdot \mid q, m, \mathcal{R}(q, m, \mathcal{H}_t))$ under the frozen backbone, score each by the token-level $F_1$ reward, and split the group at threshold $\rho = 0.5$ into a success bucket $\mathcal{O}_q^+$ and a failure bucket $\mathcal{O}_q^-$. A regex decomposer $\Psi$ (zero LLM cost) slices every rollout into a step sequence $\Psi(o) = (s_1, \ldots, s_{T_o})$, after which an LLM aligner $\Phi$ matches steps that solve the same sub-goal across the two buckets and harvests each divergent atomic skill that successes show but failures miss:

$$\widetilde{\mathcal{V}}_q \;=\; \Phi\Big(\,\big\{\,\Psi(o^{(\ell)}) : r(o^{(\ell)}; a^*) \geq \rho\,\big\}_{\ell=1}^{n},\ \ \big\{\,\Psi(o^{(\ell)}) : r(o^{(\ell)}; a^*) < \rho\,\big\}_{\ell=1}^{n}\,\Big), \tag{2}$$

where each candidate $\widetilde{v} \in \widetilde{\mathcal{V}}_q$ is a (name, description) pair, and the aligner may also name a *recipe* $\widetilde{e} \subset \widetilde{\mathcal{V}}_q$ of candidates that the successes used jointly, which becomes a hyperedge once at least two of

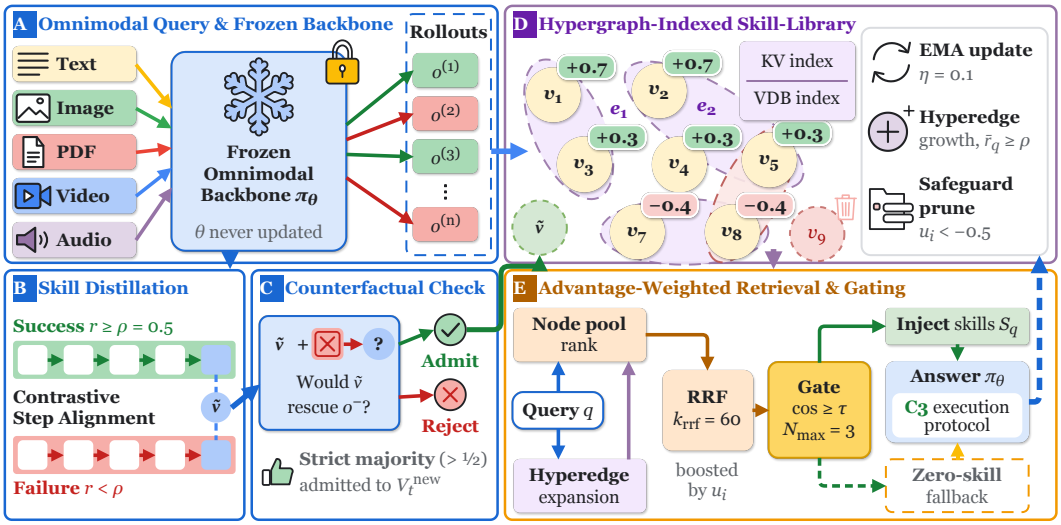


Figure 3: Methodology of HyperSkill: a frozen backbone consults a weighted skill hypergraph built by C1 (panels B–C) and evolved and retrieved by C2 (panels D–E); C3 (§3.3) governs execution.

its members are admitted. The contrast is thus localised to the divergence point of the two buckets, so noisy steps of successful rollouts are less likely to enter the harvested skill or recipe.

**Counterfactual Verification.** Step alignment alone still admits skills that merely correlate with success; we therefore pair each candidate with a list $\mathcal{P}_q$ of at most $K$ failure probes $o^- \in \mathcal{O}_q^-$ of its own question and ask the same frozen $\pi_\theta$ a binary counterfactual probe [42, 30, 40]: "had the agent known $\tilde{v}$, would $o^-$ likely have produced $a^*$?". Only when a strict majority of the probes return YES is the candidate admitted into the round-$t$ skill increment, and a deterministic identifier $\text{sid}(\tilde{v}) = \text{md5}(\text{name}(\tilde{v}))$ deduplicates it against the current library $\mathcal{V}_t$ by skill name:

$$\mathcal{V}_t^{\text{new}} = \left\{ \tilde{v} \in \tilde{\mathcal{V}}_q \;:\; \frac{1}{|\mathcal{P}_q|} \sum_{o^- \in \mathcal{P}_q} \mathbf{1}\big[\, \pi_\theta(\cdot \mid \tilde{v},\, q,\, a^*,\, o^-) = \text{YES} \,\big] \;>\; \tfrac{1}{2} \right\}, \qquad |\mathcal{P}_q| \leq K = 2. \quad (3)$$

The admitted increment $\mathcal{V}_t^{\text{new}}$ and its recipes $\mathcal{E}_t^{\text{rec}}$ feed the hypergraph evolution of C2 (§3.2).

**Protocol Skill and Answer Form.** A frozen backbone often breaks the tool or answer format, not the task. C1 therefore sorts every training rollout that ends without an answer into four failure types: malformed tool call, call beyond the budget, call without tools, and no `<answer>` tag. Each tool-related type seen in at least 2 rollouts adds fixed instruction lines, filled with the dataset's tool tags and budget, to a *protocol skill*. Its last line fixes the answer form: the integer without `\boxed{}` on math, else a short answer or, per dataset, a form derived from the training gold answers (option letter, phrase or sentence, plus an abstention line). The protocol skill is not a node of $\mathcal{H}$ (App. A).

**Proposition 1.** *Counterfactual distillation favours failure-rescuing skills over co-occurring ones.*

*Proof.* We provide experimental results in §4.3 and §4.4 and proofs in Appendix B.1. □

### 3.2 C2: Hypergraph Evolution and Gated Retrieval

To address (iii), the survivors of C1 form a weighted hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E}, w)$. During training, rollout rewards are recycled, with no extra LLM call, into a per-node advantage and per-edge weights; at test time, with the library frozen, C2 ranks, gates and injects skills for every question.

**Hypergraph Index and Advantage EMA Update.** Each admitted skill $v_i$ is embedded by a frozen text encoder $f$ and stored with its advantage $u_i$ in a dual key-value/vector-database (KV/VDB) index; admitted recipes and skill sets retrieved together for high-reward questions become hyperedges $e_j \subset \mathcal{V}$ with $|e_j| \geq 2$. After every batch, the batch-normalised mean reward of each question $q$ is broadcast to its retrieved skills $\mathcal{S}_q$, and an exponential moving average with rate $\eta$ updates their advantages $u_i$:

$$\hat{A}_q = \tanh\!\left( \frac{\bar{r}_q - \bar{r}_{\mathcal{B}_t}}{\sigma_{r,\mathcal{B}_t} + \epsilon} \right), \quad u_i \leftarrow \text{clip}_{[-1,1]}\big( (1-\eta)\, u_i + \eta\, \hat{A}_q \big) \;\; \forall\, v_i \in \mathcal{S}_q\,, \quad (4)$$

where $\bar{r}_q$ is the mean reward of the $n$ rollouts of $q$ and $(\bar{r}_{\mathcal{B}_t}, \sigma_{r,\mathcal{B}_t})$ are statistics over all rollouts of the batch, with $(\eta, \epsilon) = (0.1, 10^{-8})$. Hyperedges grow whenever a high-reward question retrieves two or more skills jointly, and a periodic pruning step removes every node whose EMA advantage falls below a threshold $u_{\min}$; together, the node and hyperedge updates of round $t$ read:

$$\mathcal{V}_{t+1} \;=\; \left\{\, v_i \in \mathcal{V}_t \cup \mathcal{V}_t^{\text{new}} \;:\; u_i \geq u_{\min} \,\right\}, \qquad \mathcal{E}_{t+1} \;=\; \mathcal{E}_t \cup \mathcal{E}_t^{\text{rec}} \cup \left\{\, \mathcal{S}_q \;:\; \bar{r}_q \geq \rho,\; |\mathcal{S}_q| \geq 2 \,\right\}, \quad (5)$$

with $u_{\min} = -0.5$ and $\mathcal{E}_t^{\text{rec}}$ the round-$t$ recipes. Each hyperedge records its own EMA advantage $w(e_j)$, updated by Eq. (4) when its members are again retrieved together for a high-reward question.

**Advantage-Weighted Reciprocal Rank Fusion.** A query is embedded by $f$ and matched against the node and hyperedge banks; a reciprocal-rank-fusion scorer [9] boosts each node by its advantage, so high-quality skills overtake more similar but low-utility ones in the ranking [22, 15, 17]:

$$\text{score}(v_i; q) \;=\; \frac{1 \;+\; \frac{1}{2}(u_i + 1)}{k_{\text{rrf}} \;+\; \text{rank}_f(v_i; q)}, \qquad \mathcal{S}_q^{\text{pool}} \;=\; \text{Top-}k_{\cos}(\mathcal{V}) \;\cup\; \text{expand}_q(\mathcal{E}), \quad (6)$$

where $\text{rank}_f(v_i; q)$ is the rank of $v_i$ under cosine similarity $\langle f(q), f(v_i) \rangle$, $k_{\text{rrf}} = 60$, $k = 100$ nearest nodes form the candidate pool, and $\text{expand}_q(\mathcal{E})$ adds the members of hyperedges matching $q$, so that in libraries larger than the pool, skills composed with a relevant skill still enter it. Flat dictionaries [3, 4, 50] and frontier filters [2] lack this per-item quality prior and joint composition.

**Proposition 2.** *Advantage-weighted ranking places high-utility skills above merely similar ones.*

*Proof.* We provide experimental results in §4.3 and §4.5 and proofs in Appendix B.2. □

**Similarity Floor, Capacity Cap, and Zero-Skill Fallback.** Injecting marginal skills from a small, noisy library can be worse than injecting none. A floor $\tau$ discards candidates whose cosine to $q$ is below it, a cap $N_{\max}$ bounds their number, and an empty result falls back to the skill-free prompt:

$$\mathcal{S}_q \;=\; \text{Top-}N_{\max}\Big(\, \text{score}(\,\cdot\,; q)\,,\;\; \left\{\, v_i \in \mathcal{S}_q^{\text{pool}} \;:\; \langle f(q), f(v_i) \rangle \geq \tau \,\right\} \Big), \qquad N_{\max} = 3\,, \quad (7)$$

where $|\mathcal{S}_q| = 0$ reduces $\pi_\theta(\cdot \,|\, q, m, \mathcal{S}_q)$ to $\pi_\theta(\cdot \,|\, q, m)$; $\tau$ is set per dataset (App. E.7). Retrieved skills enter as a strategy hint (App. A) or a terse list. The protocol skill of C1, when enabled, is added for every question without gating and re-injected after each tool turn with the remaining budget.

**Proposition 3.** *Gating with zero-skill fallback never harms the skill-free pipeline in expectation.*

*Proof.* We provide experimental results in §4.3 and §4.6 and proofs in Appendix B.3. □

### 3.3 C3: Execution Protocol

**Tool Calls and Final Answer.** C3 decides *how* the frozen backbone executes a rollout; it never edits $\mathcal{H}$, and its options are set per dataset (App. E.7). A *concrete tool prompt* names the real tool tag (e.g. `<text_search>`) instead of a `TOOL_TAG` placeholder, and a *lenient parser* also executes copied placeholder forms such as `<TOOL_TAG>text_search>q`. If a rollout ends without a closed `<answer>` tag (e.g. once the tool budget is spent), a *final-answer turn* asks once more. An *answer suffix* requests the final integer without `\boxed{}` on math and the option letter on multiple choice.

**Evidence and Voting.** *Question pre-search* sends the question verbatim to the `text_search` tool once, outside the tool budget; *locate pages* adds the text of the pages the question names and of the top BM25 [38] pages of the question's own PDF, with no API call; on Omni, *time grounding* lists the keyframe times and, on multiple choice, an *option check* asks the model to weigh every option against frames and audio. *Self-consistency* [43] adds 5 samples at temperature 0.7 and keeps their most frequent answer, or the greedy one on a tie or when no sample answers. C3 also sets per-dataset token budgets, and on five datasets its options also apply to the training rollouts that build the library, so the library is learned under the same protocol that serves its test questions (App. E.7).

**Closing the Loop.** A training iteration on $\mathcal{B}_t$ retrieves $\mathcal{S}_q \leftarrow \mathcal{R}(q, m, \mathcal{H}_t)$, generates $n$ rollouts under frozen $\theta$, records their failure types, computes advantages $\{\hat{A}_q\}$, admits the probe survivors of $\Phi \circ \Psi$, and updates $\mathcal{H}_t \to \mathcal{H}_{t+1}$ by EMA, hyperedge growth and pruning, so an auditable delta $\Delta\mathcal{H}$ replaces the GRPO gradient step $\Delta\theta$. At evaluation the library is frozen, and C2 and C3 serve each test question. Defaults are $(n, K, \eta, k_{\text{rrf}}, N_{\max}, \rho, u_{\min}) = (2, 2, 0.1, 60, 3, 0.5, -0.5)$.

Table 2: Main results on the 11 datasets of HyperSkill-Bench (§4.1; token-level F1 %, CJK-aware scorer, full-set mean of one run). Subscripts: change vs. Zero-shot on the same dataset (green +, red −, grey ±0.0); **bold**: best, underline: second best per row. All columns share the frozen `gemini-2.5-flash-lite` backbone, the same tools and scoring rule; HyperSkill is ours.

| Modality | Dataset | Zero-shot | T.F.GRPO | SkillCraft | AutoSkill | EvoSkill | M.-Skills | XSkill | HyperSkill |
|---|---|---|---|---|---|---|---|---|---|
| Text | AIME2024 | 10.0 | 10.0$_{\pm 0.0}$ | 13.3$_{+3.3}$ | 13.3$_{+3.3}$ | 10.0$_{\pm 0.0}$ | 13.3$_{+3.3}$ | 6.7$_{-3.3}$ | **40.0**$_{+30.0}$ |
| | AIME2025 | 33.3 | 26.7$_{-6.6}$ | 30.0$_{-3.3}$ | 33.3$_{\pm 0.0}$ | 33.3$_{\pm 0.0}$ | 26.7$_{-6.6}$ | 30.0$_{-3.3}$ | **53.3**$_{+20.0}$ |
| | WebWalkerQA | 20.7 | 16.9$_{-3.8}$ | 16.5$_{-4.2}$ | 20.7$_{\pm 0.0}$ | 18.6$_{-2.1}$ | 16.3$_{-4.4}$ | 14.8$_{-5.9}$ | **23.3**$_{+2.6}$ |
| VL-Web | FVQA | 21.1 | 27.0$_{+5.9}$ | 17.8$_{-3.3}$ | 31.3$_{+10.2}$ | 23.9$_{+2.8}$ | 19.5$_{-1.6}$ | 26.6$_{+5.5}$ | **41.7**$_{+20.6}$ |
| | InfoSeek | 22.8 | 27.4$_{+4.6}$ | 15.2$_{-7.6}$ | 22.3$_{-0.5}$ | 16.8$_{-6.0}$ | 22.7$_{-0.1}$ | 33.4$_{+10.6}$ | **44.8**$_{+22.0}$ |
| | SimpleVQA | 26.5 | 19.3$_{-7.2}$ | 19.8$_{-6.7}$ | 27.7$_{+1.2}$ | 33.7$_{+7.2}$ | 28.6$_{+2.1}$ | 33.8$_{+7.3}$ | **37.4**$_{+10.9}$ |
| | LiveVQA | 55.0 | 56.3$_{+1.3}$ | 59.9$_{+4.9}$ | 64.4$_{+9.4}$ | 59.4$_{+4.4}$ | 58.5$_{+3.5}$ | 59.6$_{+4.6}$ | **65.7**$_{+10.7}$ |
| VL-Doc | DocBench | 32.8 | 26.1$_{-6.7}$ | 16.0$_{-16.8}$ | 32.9$_{+0.1}$ | 32.8$_{\pm 0.0}$ | 35.1$_{+2.3}$ | 33.1$_{+0.3}$ | **39.2**$_{+6.4}$ |
| | MMLongB. | 19.4 | 22.6$_{+3.2}$ | 9.8$_{-9.6}$ | 12.4$_{-7.0}$ | 15.4$_{-4.0}$ | 10.1$_{-9.3}$ | 23.0$_{+3.6}$ | **32.1**$_{+12.7}$ |
| Omni | OmniVideoB. | 16.0 | 6.4$_{-9.6}$ | 14.9$_{-1.1}$ | 12.2$_{-3.8}$ | 16.0$_{\pm 0.0}$ | 5.1$_{-10.9}$ | 17.2$_{+1.2}$ | **31.5**$_{+15.5}$ |
| | WorldSense | 43.8 | 47.5$_{+3.7}$ | 43.8$_{\pm 0.0}$ | 43.8$_{\pm 0.0}$ | 43.8$_{\pm 0.0}$ | 49.1$_{+5.3}$ | 40.6$_{-3.2}$ | **53.1**$_{+9.3}$ |
| | **Avg.** | 27.4 | 26.0$_{-1.4}$ | 23.4$_{-4.0}$ | 28.6$_{+1.2}$ | 27.6$_{+0.2}$ | 25.9$_{-1.5}$ | 29.0$_{+1.6}$ | **42.0**$_{+14.6}$ |

# 4 Experiments

We answer six RQs: **RQ1:** Does HyperSkill rank first on every dataset? **RQ2:** What does each component add? **RQ3:** Does counterfactual distillation keep a compact library of useful skills? **RQ4:** How does the hypergraph evolve under retrieval feedback? **RQ5:** How sensitive is gating to $\tau$, $N_{\max}$ and library size? **RQ6:** What are HyperSkill's training cost, library size and latency?

## 4.1 Experimental Setup

**Datasets and Evaluation Metrics.** We evaluate on *HyperSkill-Bench*, a collection of 11 public datasets in four regimes (Appendix D.1): Text reasoning (AIME2024/2025, WebWalkerQA), VL-Web (FVQA, InfoSeek, SimpleVQA, LiveVQA), VL-Doc (DocBench, MMLongBench-Doc) and Omni (OmniVideoBench, WorldSense). The metric is token-level $F_1$ [37] with answer-shape awareness and a CJK-aware tokeniser that splits Chinese into characters; every cell of Table 2 and Figure 4 is the full-set mean over the dataset's test questions (30 for AIME2024/2025, 64 otherwise; 636 in total), from one evaluation run per cell, scored by the same rule for every column (Appendix D.2).

**Baselines and Ablations.** We compare against bare *Zero-shot* [54] (same tools, no library or C3) and the 6 training-free skill methods of Table 1: TrainingFreeGRPO [3] (experience pool), EvoSkill [2] (frontier set), SkillCraft [4] and AutoSkill [50] (code, tip cards), Memento-Skills [56] and XSkill [19] (memory). Each baseline builds its library on the same 128 training questions by re-implementing its published procedure (Appendix E), likewise without C3. Three ablations reuse the trained library (Appendix E.5): *w/o C1* replaces all distilled content by a generic protocol line, *w/o C2* also drops that line, and *w/o C3*, applied on its own, only resets the C3 execution options to their defaults.

**Implementation Details.** Every method shares the frozen `gemini-2.5-flash-lite` [8] backbone with native vision/audio interfaces and the `gemini-embedding-2-preview` [21] encoder $f$, both via OpenRouter. Each HyperSkill library is trained for 1 epoch on 128 training questions per dataset (deduplicated, seed 42) with batch size 4, $n=2$ rollouts per question, seed 1 and the defaults of §3. Three library-run settings (R1–R3) differ in rollout temperature, early stopping, hint synthesis, probe sampling and training-time floor. Each dataset's run, floor $\tau$ and C3 options are selected on held-out training questions, and each library is then retrained on all 128 questions (Appendix E.7).

**Tool Stack and Decoding.** Every method uses the same regime tools: a `text_search` tool for WebWalkerQA and VL-Web, served by `gemini-3.8-flash`; images passed to the backbone; a BM25 index over the question's own PDF for VL-Doc; and uniformly sampled keyframes for Omni, 8 on OmniVideoBench and 32 plus the audio track on WorldSense. Each question gets at most 3 (Text), 5 (VL) or 8 (Omni) rounds and 2 tool calls. Decoding is greedy at temperature 0 with seed 1. The baselines use 1024 new tokens (2048 on AIME2025, 4096 on WorldSense) and retrieve without a similarity floor; HyperSkill's token budgets and votes belong to C3. Variance: Appendix D.2.

### 4.2 Main Results (RQ1)

**Overall.** HyperSkill is best or tied on all 11 datasets (Table 2) and lifts the average from 29.0 for the strongest baseline, XSkill, to 42.0: a +13.0 absolute / +44.9% relative gain, and +14.6 over Zero-shot (27.4). The six baselines lie within 23.4–29.0, −4.0 to +1.6 around Zero-shot, and each falls below Zero-shot on at least three datasets. HyperSkill is above Zero-shot on every row.

**Per-dataset.** The margin over the best other column is largest on AIME2024 (+26.7 over SkillCraft, AutoSkill and Memento-Skills), AIME2025 (+20.0 over three columns) and OmniVideoBench (+14.3 over XSkill), and is +11.4 on InfoSeek, +10.4 on FVQA and +9.1 on MMLongBench-Doc. The other five margins are +1.3 to +4.1, and their paired bootstrap 95% intervals include zero (Appendix D.2), so we conservatively count these five as ties. RQ2 then ablates C1, C1+C2, or C3 alone to measure how much each of the three components adds to the 11-dataset average.

**Per-regime.** By regime, HyperSkill's mean per-dataset lead over the best other column is +16.4 on Text, +9.2 on Omni, +6.7 on VL-Web and +6.6 on VL-Doc, so HyperSkill leads by more than 6 F1 points in every regime, with the largest lead on the Text regime.

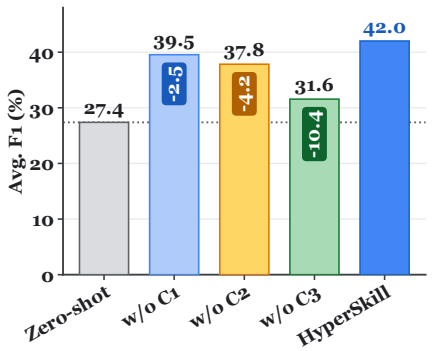


Figure 4: Average F1 over the 11 datasets; white numbers: change vs. HyperSkill.

### 4.3 Ablation Study (RQ2)

The three ablations (Figure 4) lower the 11-dataset average from 42.0 to 39.5 (w/o C1), 37.8 (w/o C2) and 31.6 (w/o C3), and every variant stays well above Zero-shot (27.4); since every one of the ablations lowers the average, each of the three components is effective. The components act at different stages: C1 distils verified skills and a protocol skill from training failures, C2 delivers them to each test question behind the gate, and C3 makes the frozen backbone call its tools and commit to an answer. Only the full HyperSkill, which combines all three components, reaches the best 11-dataset average, and every ablated variant still exceeds all six baselines (23.4–29.0), so HyperSkill's lead over the baselines does not depend on any single one of its components.

### 4.4 Counterfactual Distillation Effectiveness (RQ3)

Figure 5 examines the counterfactual step-level distillation of §3.1, asking whether it keeps a compact library, whether the probe admits useful skills, and whether the distilled skills help at test time.

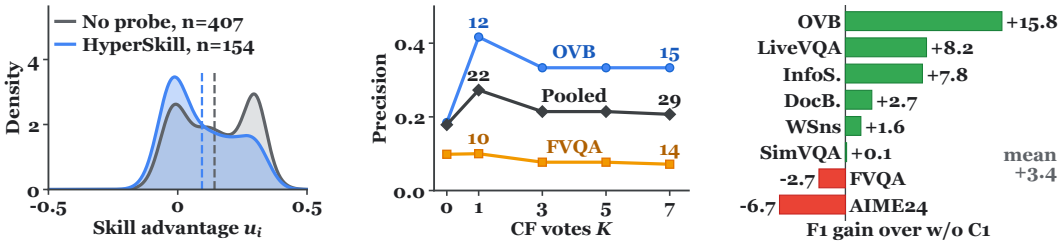


Figure 5: (a) $u_i$ KDEs of HyperSkill vs. libraries trained without the probe; (b) probe precision vs. $K$ (labels: #accepted; AIME24 none); (c) F1 gain over w/o C1 on the 8 differing datasets.

**Distilled skills are few and useful.** Pooled over the 11 libraries, HyperSkill admits 154 skills, 38% of the 407 skills of the libraries trained without the probe (panel a), so the probe keeps the library compact. They also help: against w/o C1 (panel c), which swaps in a generic protocol line, HyperSkill gains on 6 of the 8 datasets where the two configurations differ (mean +3.4 F1), led by OmniVideoBench, LiveVQA and InfoSeek; the other three share one configuration.

**A single verification vote sharpens skill admission.** On 96 of the 128 training questions each of AIME2024, FVQA and OmniVideoBench (panel b), we take the 140 (question, skill) candidates where the no-skill run fails (F1 $< 0.5$) and a skill is retrieved, and count a candidate as useful when injecting that skill flips the question to a success. Without the probe ($K = 0$) 25 of 140 candidates are useful (0.179). One counterfactual vote raises pooled precision to 0.273 (6/22) and,

on OmniVideoBench, from 0.185 to 0.417 (5/12). Votes at temperature 0.1 are nearly deterministic: a single vote already captures the filtering gain and every $K \geq 1$ keeps pooled precision above the no-probe rate; hence the libraries use one vote (R2/R3) or two votes on one failure (R1).

### 4.5 Hypergraph Evolution under Retrieval Feedback (RQ4)

Figure 6 traces the weighted hypergraph of §3.2 over training, asking how retrieval feedback grows the library and its connectivity and how the advantages $u_i$ of the admitted skills evolve.

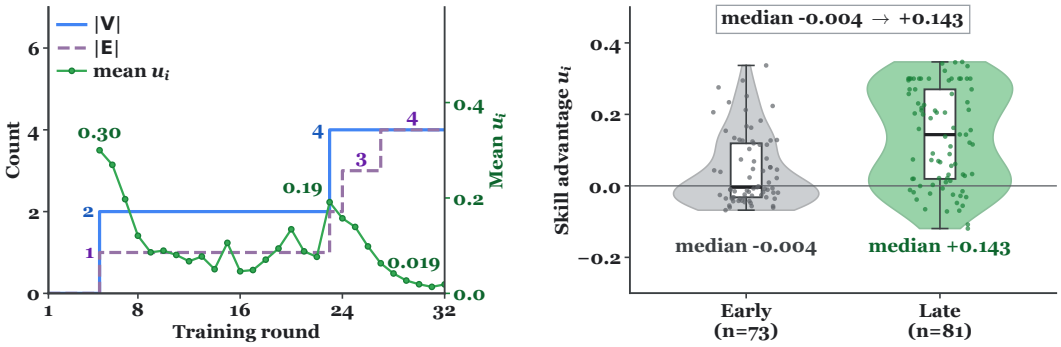


Figure 6: (a) Library size ($|\mathcal{V}|$, $|\mathcal{E}|$) and mean $u_i$ over the 32 rounds of an AIME2024 training re-run; (b) violins of $u_i$ for the early- and late-admitted halves pooled over the 11 final libraries.

**Skills accumulate early and stabilise.** In an AIME2024 training re-run with per-round logging (32 rounds of 4 questions; panel a), the library grows from 0 to 2 skills at round 5 and to 4 at round 23, where it stays through round 32; the hyperedges grow from 0 to 1 and reach 4 at round 27, when the library, taken as a skill–hyperedge graph, reaches 8 nodes and 10 links. The final 4 skills match the size of the AIME2024 library used in Table 2. Admitted skills start at $u_i = 0.3$ and the EMA of Eq. (4) pulls them toward the batch-normalised advantage, so the mean $u_i$ moves from 0.30 (round 5) to 0.19 (round 23) and 0.019 (round 32) and stays positive throughout the training re-run.

**Advantages keep being refined after admission.** In the 11 libraries (panel b), the earlier-admitted half of each library, updated most often, has median $u_i \approx 0$ (73 skills) and the later half $+0.143$ (81 skills): the gap ($\approx 0.15$) grows with the EMA updates that pull each $u_i$ from 0.3 toward $\hat{A}_q$.

### 4.6 Sensitivity of Gated Retrieval (RQ5)

Figure 7 varies one setting at a time around the frozen configuration of InfoSeek and OmniVideoBench, which cover the image-based VL-Web and the video-based Omni regimes (64 test questions per cell, single runs). The floor $\tau$ takes six values around its frozen value on each dataset (InfoSeek on the hub-corrected scale of Appendix E.7), $N_{\max} \in \{1, 3, 5\}$, and the library subsets are nested random subsets with 25, 50, 75 and 100% of the skills, where a hyperedge is kept only if all its member skills are kept; ungated retrieval sets $\tau = -9$. This is a post-hoc sensitivity analysis on the test split; it does not change the frozen per-dataset configuration or the reported results of Table 2.

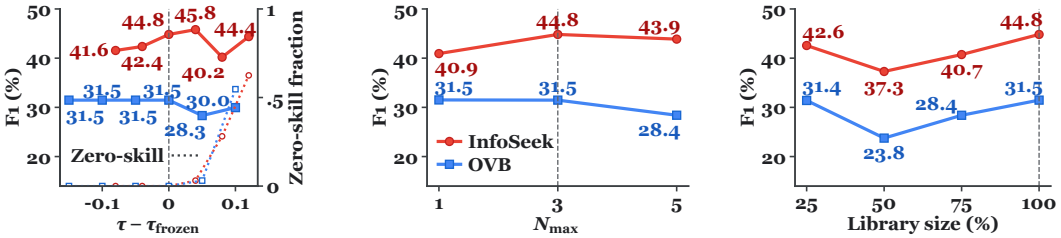


Figure 7: (a) F1 and zero-skill fraction vs. the offset of $\tau$ from its frozen value; (b) F1 vs. $N_{\max}$; (c) F1 of gated retrieval on nested library subsets. Grey lines: frozen settings.

**The frozen settings lie in a stable region.** Around the frozen floor (panel a), InfoSeek stays between 40.20 and 45.80 F1 (44.82 at its hub-corrected $\tilde{\tau} = -0.02$), and OmniVideoBench gives 31.47 at every $\tau \leq 0.55$, where injected skills are identical, and 28.33–29.95 at the two larger values.

**The floor controls injection.** The zero-skill fraction is 0 at the frozen $\tau$ on both datasets and rises to 0.63 and 0.55 at the largest $\tau$, where on average 0.70 and 0.78 skills are injected per question.

**A small capacity cap suffices.** $N_{\max} = 3$ is best or near-best on both datasets (panel b): 44.82 vs 40.93 at 1 and 43.87 at 5 on InfoSeek, and 31.47 vs 31.51 at 1 and 28.40 at 5 on OmniVideoBench, with 1, 3 and up to 5 skills injected per question at the three cap values, respectively.

**Gated, the full library is best.** With gating (panel c), the full library gives the highest F1 on both datasets (44.82 on InfoSeek, 31.47 on OmniVideoBench), +4.1 and +3.1 over the 75% subset; at full size, gated and ungated retrieval give 44.82 vs 44.64 on InfoSeek and 31.47 vs 31.47 on OmniVideoBench, where the injected skills coincide, so gating lowers F1 on neither dataset.

### 4.7 Cost Analysis (RQ6)

Figure 8 reports two deployment-relevant costs of HyperSkill on all 11 datasets: the training-API spend to build each library and the size of the learned library in skills and hyperedges; we also compare the per-question inference latency of HyperSkill with Zero-shot on three datasets.

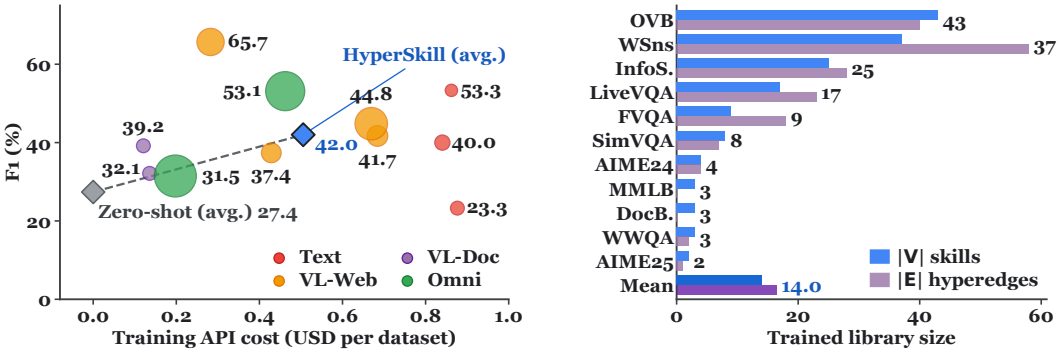


Figure 8: (a) Training API cost vs. F1 per dataset, coloured by regime, bubble area $\propto |\mathcal{V}|$; (b) library size per dataset: skills $|\mathcal{V}|$ (dark) and hyperedges $|\mathcal{E}|$ (light), with the mean in the last row.

**Cheap to train, compact to store.** HyperSkill's library costs \$0.12–\$0.88 per dataset to train (mean \$0.51, total \$5.56, of which \$2.48 is knowledge search) and holds 2–43 skills (mean 14.0) and 0–58 hyperedges; one test pass over all 11 datasets costs \$2.34. Panel (a) adds the Zero-shot (27.4) and HyperSkill (42.0) averages, and panel (b) sorts the datasets by library size, with the mean in the last row; the two Omni libraries are the largest (43 and 37 skills), and the 154 skills pooled over all 11 libraries are 38% of the 407 skills of the libraries trained without the probe (§4.4). Training needs no gradient step: each library is built from API calls alone, and per question the counterfactual check of C1 adds at most $2|\widetilde{\mathcal{V}}_q|$ verification calls at $K = 2$ (Eq. (8)), which take the place of the backward pass of standard GRPO (Appendix A) and are already included in the training costs above.

**Inference latency stays close to Zero-shot.** Measured for HyperSkill and Zero-shot on 16 test questions per timed dataset with the same harness, mean latency per question is 14.98 vs 11.75 s on AIME2024, 12.78 vs 14.56 s on FVQA and 3.56 vs 3.66 s on DocBench: HyperSkill adds at most about 3 s (on AIME2024) and is on par or slightly faster elsewhere, since API, tool and network time dominate and C1 and C3 add only short instruction texts to the prompt (Appendix A). The timed datasets cover three regimes (Text, VL-Web and VL-Doc), each with its own tools (§4.1).

## 5 Conclusion

HyperSkill is a *Training-Free Omnimodal GRPO* framework that evolves a *Hypergraph-Indexed Skill-Library* through training-time distillation (C1), gated hypergraph retrieval (C2) and an execution protocol (C3). Without parameter updates, it is best or tied on all 11 datasets (avg. 42.0, +13.0 over the strongest baseline), and every skill stays readable and editable. Each component raises the average, counterfactual distillation keeps a compact library (154 vs 407 skills), and the gate settings lie in a stable region. A library costs at most \$0.88 to train, a full test pass costs \$2.34, and latency stays close to Zero-shot. Future work includes pairing the library with lightweight parameter updates, sharing it across related datasets and domains, and extending C3 to new tool types and modalities.

## Acknowledgments

This work is supported by the National Research Foundation, Singapore, under its Frontier Competitive Research Programme (NRF-F-CRP-2024-0005). Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not reflect the views of National Research Foundation, Singapore. The authors declare no competing interests.

## References

[1] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, et al. Gpt-4 technical report, 2024.

[2] Salaheddin Alzubi, Noah Provenzano, Jaydon Bingham, Christian Alexander Calvo, Weiyuan Chen, and Tu Vu. Evoskill: Automated skill discovery for multi-agent systems. In *Third Conference on Language Modeling*, 2026.

[3] Yuzheng Cai, Siqi Cai, Yuchen Shi, Zihan Xu, Lichao Chen, Yulei Qin, Xiaoyu Tan, Gang Li, Zongyi Li, Haojia Lin, Yong Mao, Ke Li, and Xing Sun. Training-free group relative policy optimization, 2025.

[4] Shiqi Chen, Jingze Gai, Ruochen Zhou, Jinghan Zhang, Tongyao Zhu, Junlong Li, Kangrui Wang, Zihan Wang, Zhengyu Chen, Klara Kaleb, Ning Miao, Siyang Gao, Cong Lu, Manling Li, Junxian He, and Yee Whye Teh. Skillcraft: Can llm agents learn to use tools skillfully?, 2026.

[5] Yang Chen, Hexiang Hu, Yi Luan, Haitian Sun, Soravit Changpinyo, Alan Ritter, and Ming-Wei Chang. Can pre-trained vision and language models answer visual information-seeking questions? In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14948–14968, Singapore, December 2023. Association for Computational Linguistics.

[6] Xianfu Cheng, Wei Zhang, Shiwei Zhang, Jian Yang, Xiangyuan Guan, Xianjie Wu, Xiang Li, Ge Zhang, Jiaheng Liu, Yuying Mai, Yutao Zeng, Zhoufutu Wen, Ke Jin, Baorui Wang, Weixiao Zhou, Yunhong Lu, Tongliang Li, Wenhao Huang, and Zhoujun Li. Simplevqa: Multimodal factuality evaluation for multimodal large language models, 2025.

[7] Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory. In *ECAI 2025*, pages 2993–3000. IOS Press, October 2025.

[8] Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, Luke Marris, Sam Petulla, Colin Gaffney, Asaf Aharoni, Nathan Lintz, Tiago Cardal Pais, Henrik Jacobsson, Idan Szpektor, Nan-Jiang Jiang, Krishna Haridasan, Ahmed Omran, Nikunj Saunshi, Dara Bahri, Gaurav Mishra, Eric Chu, Toby Boyd, Brad Hekman, Aaron Parisi, Chaoyi Zhang, Kornraphop Kawintiranon, Tania Bedrax-Weiss, Oliver Wang, Ya Xu, Ollie Purkiss, Uri Mendlovic, Ilaï Deutel, Nam Nguyen, Adam Langley, Flip Korn, Lucia Rossazza, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities, 2025.

[9] Gordon V. Cormack, Charles L A Clarke, and Stefan Buettcher. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 758–759. ACM, July 2009.

[10] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. From local to global: A graph rag approach to query-focused summarization, 2025.

[11] OpenAI, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, Ally Bennett, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrew Duberstein, Andrew Kondrich, Andrey Mishchenko, Andy Applebaum, Angela Jiang, Ashvin Nair, Barret Zoph, Behrooz Ghorbani, Bohan Zhang, Ben Rossen, Benjamin Sokolowsky, Boaz Barak, Bob McGrew, Borys Minaiev, Botao Hao, Bowen Baker, Brandon Houghton, Brandon McKinzie, et al. Openai o1 system card, 2026.

[12] Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. Retool: Reinforcement learning for strategic tool use in llms. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust, editors, *International Conference on Learning Representations*, volume 2026, pages 37909–37926, 2026.

[13] Yifan Feng, Hao Hu, Shihui Ying, Xingliang Hou, Shiquan Liu, Mingyuan Yang, Junchang Li, Shaoyi Du, Nanning Zheng, Han Hu, and Yue Gao. Hyper-rag: combating llm hallucinations using hypergraph-driven retrieval-augmented generation. *Nature Communications*, 17(1), April 2026.

[14] Mingyang Fu, Yuyang Peng, Dongping Chen, Zetong Zhou, Benlin Liu, Yao Wan, Zhou Zhao, Philip S. Yu, and Ranjay Krishna. Seeking and updating with live visual knowledge, 2025.

[15] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey, 2024.

[16] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

[17] Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. LightRAG: Simple and fast retrieval-augmented generation. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng, editors, *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 10746–10761, Suzhou, China, November 2025. Association for Computational Linguistics.

[18] Jack Hong, Shilin Yan, Jiayin Cai, Xiaolong Jiang, Yao Hu, and Weidi Xie. Worldsense: Evaluating real-world omnimodal understanding for multimodal llms. In *International Conference on Learning Representations*, 2026.

[19] Guanyu Jiang, Zhaochen Su, Xiaoye Qu, and Yi R. Fung. XSkill: Continual learning from experience and skills in multimodal agents. In Tong Zhang, Miroslav Dudik, Martin Jaggi, Alekh Agarwal, Sharon Li, Dale Schuurmans, Jerry Zhu, Felix Berkenkamp, Hanze Dong, and Alberto Bietti, editors, *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, pages 53078–53104. PMLR, 06–11 Jul 2026.

[20] Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training LLMs to reason and leverage search engines with reinforcement learning. In *Second Conference on Language Modeling*, 2025.

[21] Jinhyuk Lee, Feiyang Chen, Sahil Dua, Daniel Cer, Madhuri Shanbhogue, Iftekhar Naim, Gustavo Hernández Ábrego, Zhe Li, Kaifeng Chen, Henrique Schechter Vera, Xiaoqi Ren, Shanfeng Zhang, Daniel Salz, Michael Boratko, Jay Han, Blair Chen, Shuo Huang, Vikram Rao, Paul Suganthan, Feng Han, Andreas Doumanoglou, Nithi Gupta, Fedor Moiseev, Cathy Yip, Aashi Jain, Simon Baumgartner, Shahrokh Shahi, Frank Palma Gomez, Sandeep Mariserla, Min Choi, Parashar Shah, Sonam Goenka, Ke Chen, Ye Xia, Koert Chen, Sai Meher Karthik Duddu, Yichang Chen, Trevor Walker, Wenlei Zhou, Rakesh Ghiya, et al. Gemini embedding: Generalizable embeddings from gemini, 2025.

[22] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

[23] Caorui Li, Yu Chen, Yiyan Ji, Jin Xu, Zhenyu Cui, Shihao Li, Yuanxing Zhang, Wentao Wang, Zhenghao Song, Dingling Zhang, Ying He, Haoxiang Liu, Yuxuan Wang, Qiufeng Wang, Jiafu Tang, Zhenhe Wu, Jiehui Luo, Zhiyu Pan, Weihao Xie, Chenchen Zhang, Zhaohui Wang, Jiayi Tian, Yanghai Wang, Zhe Cao, Minxin Dai, Ke Wang, Runzhe Wen, Yinghao Ma, Yaning Pan, Sungkyun Chang, Termeh Taheri, Haiwen Xia, Christos Plachouras, Emmanouil Benetos, Yizhi Li, Ge Zhang, Jian Yang, Tianhao Peng, Zili Wang, Minghao Liu, et al. Omnivideobench: Towards audio-visual understanding evaluation for omni mllms. In *International Conference on Learning Representations*, 2026.

[24] Weizhen Li, Jianbo Lin, Zhuo Jiang, Jingyi Cao, Xinpeng Liu, Jiayu Zhang, Zhen Huang, Qianben Chen, Weichen Sun, Qiexiang Wang, Hongxuan Lu, Tianrui Qin, Chenghao Zhu, Yi Yao, Shuying Fan, Xiaowan Li, Tiannan Wang, Pai Liu, King Zhu, He Zhu, Dingfeng Shi, Piaohong Wang, Yeyi Guan, Xiangru Tang, Minghao Liu, Y. Jiang, Jian Yang, Jiaheng Liu, Ge Zhang, and Wangchunshu Zhou. Chain-of-agents: End-to-end agent foundation models via multi-agent distillation and agentic rl, 2025.

[25] Xiaoxi Li, Wenxiang Jiao, Jiarui Jin, Haoxuan Li, Hao Wang, Shijian Wang, Guanting Dong, Jiajie Jin, Yinuo Wang, Yuan Lu, Ji-Rong Wen, Zhicheng Dou, and Zhouchen Lin. Omnigaia: Towards native omni-modal ai agents, 2026.

[26] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.

[27] Haoran Luo, Haihong E, Guanting Chen, Qika Lin, Yikai Guo, Fangzhi Xu, Zemin Kuang, Meina Song, Xiaobao Wu, Yifan Zhu, and Anh Tuan Luu. Graph-r1: Towards agentic GraphRAG framework via end-to-end reinforcement learning. In Tong Zhang, Miroslav Dudik, Martin Jaggi, Alekh Agarwal, Sharon Li, Dale Schuurmans, Jerry Zhu, Felix Berkenkamp, Hanze Dong, and Alberto Bietti, editors, *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, pages 82220–82240. PMLR, 06–11 Jul 2026.

[28] Haoran Luo, Haihong E, Guanting Chen, Yandan Zheng, Xiaobao Wu, Yikai Guo, Qika Lin, Yu Feng, Zemin Kuang, Meina Song, Yifan Zhu, and Anh Tuan Luu. Hypergraphrag: Retrieval-augmented generation via hypergraph-structured knowledge representation. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen, editors, *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 152206–152234. Curran Associates, Inc., 2025.

[29] Yubo Ma, Yuhang Zang, Liangyu Chen, Meiqi Chen, Yizhu Jiao, Xinze Li, Xinyuan Lu, Ziyu Liu, Yan Ma, Xiaoyi Dong, Pan Zhang, Liangming Pan, Yu-Gang Jiang, Jiaqi Wang, Yixin Cao, and Aixin Sun. Mmlongbench-doc: Benchmarking long-context document understanding with visualizations. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 95963–96010. Curran Associates, Inc., 2024.

[30] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594, 2023.

[31] Mathematical Association of America. 2024 AIME I and AIME II problems. AoPS Wiki, `https://artofproblemsolving.com/wiki/index.php/2024_AIME_I_Problems` and `https://artofproblemsolving.com/wiki/index.php/2024_AIME_II_Problems`, 2024.

[32] Mathematical Association of America. 2025 AIME I and AIME II problems. AoPS Wiki, `https://artofproblemsolving.com/wiki/index.php/2025_AIME_I_Problems` and `https://artofproblemsolving.com/wiki/index.php/2025_AIME_II_Problems`, 2025.

[33] Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems, 2024.

[34] Hongjin Qian, Zheng Liu, Peitian Zhang, Kelong Mao, Defu Lian, Zhicheng Dou, and Tiejun Huang. Memorag: Boosting long context processing with global memory-enhanced retrieval augmentation. In *Proceedings of the ACM on Web Conference 2025*, pages 2366–2377, 2025.

[35] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[36] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. In *International conference on machine learning*, pages 28492–28518. PMLR, 2023.

[37] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In Jian Su, Kevin Duh, and Xavier Carreras, editors, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics.

[38] Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.

[39] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024.

[40] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

[41] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024.

[42] Nengbo Wang, Xiaotian Han, Jagdip Singh, Jing Ma, and Vipin Chaudhary. Causalrag: Integrating causal graphs into retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 22680–22693, 2025.

[43] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023.

[44] Zihao Wang, Shaofei Cai, Anji Liu, Yonggang Jin, Jinbing Hou, Bowei Zhang, Haowei Lin, Zhaofeng He, Zilong Zheng, Yaodong Yang, Xiaojian Ma, and Yitao Liang. JARVIS-1: Open-world multi-task agents with memory-augmented multimodal language models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(3):1894–1907, March 2025.

[45] Jialong Wu, Wenbiao Yin, Yong Jiang, Zhenglin Wang, Zekun Xi, Runnan Fang, Linhai Zhang, Yulan He, Deyu Zhou, Pengjun Xie, et al. Webwalker: Benchmarking llms in web traversal. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10290–10305, 2025.

[46] Jinming Wu, Zihao Deng, Wei Li, Yiding Liu, Bo You, Bo Li, Zejun Ma, and Ziwei Liu. Mmsearch-r1: Incentivizing lmms to search, 2025.

[47] Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094, 2024.

[48] Jin Xu, Zhifang Guo, Hangrui Hu, Yunfei Chu, Xiong Wang, Jinzheng He, Yuxuan Wang, Xian Shi, Ting He, Xinfa Zhu, Yuanjun Lv, Yongqi Wang, Dake Guo, He Wang, Linhan Ma,

Pei Zhang, Xinyu Zhang, Hongkun Hao, Zishan Guo, Baosong Yang, Bin Zhang, Ziyang Ma, Xipin Wei, Shuai Bai, Keqin Chen, Xuejing Liu, Peng Wang, Mingkun Yang, Dayiheng Liu, Xingzhang Ren, Bo Zheng, Rui Men, Fan Zhou, Bowen Yu, Jianxin Yang, Le Yu, Jingren Zhou, and Junyang Lin. Qwen3-omni technical report, 2025.

[49] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, et al. Qwen3 technical report, 2025.

[50] Yutao Yang, Junsong Li, Qianjun Pan, Bihao Zhan, Yuxuan Cai, Lin Du, Jie Zhou, Kai Chen, Qin Chen, Xin Li, Bo Zhang, and Liang He. Autoskill: Experience-driven lifelong learning via skill self-evolution, 2026.

[51] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *11th International Conference on Learning Representations, ICLR 2023*, 2023.

[52] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, juncai liu, LingJun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen, editors, *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 113222–113244. Curran Associates, Inc., 2025.

[53] Juwei Yue, Chuanrui Hu, Jiawei Sheng, Zuyi Zhou, Wenyuan Zhang, Tingwen Liu, Li Guo, and Yafeng Deng. HyperMem: Hypergraph memory for long-term conversations. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens, editors, *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 35237–35254, San Diego, California, United States, July 2026. Association for Computational Linguistics.

[54] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Zican Dong, Yupeng Hou, Beichen Zhang, Yingqian Min, Junjie Zhang, Peiyu Liu, Xiaolei Wang, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Yiwen Hu, Jian-Yun Nie, and Ji-Rong Wen. A survey of large language models. *Frontiers of Computer Science*, 20(12), May 2026.

[55] Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization, 2025.

[56] Huichi Zhou, Siyuan Guo, Anjie Liu, Zhongwei Yu, Ziqin Gong, Bowen Zhao, Zhixun Chen, Menglong Zhang, Yihang Chen, Jinsong Li, Runyu Yang, Qiangbin Liu, Xinlei Yu, Jianmin Zhou, Na Wang, Chunyang Sun, and Jun Wang. Memento-skills: Let agents design agents, 2026.

[57] Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations*, 2024.

[58] Yifan Zhu, Xinyu Mu, Tao Feng, Zhonghong Ou, Yuning Gong, and Haoran Luo. Omnirag-agent: Agentic omnimodal reasoning for low-resource long audio-video question answering, 2026.

[59] Anni Zou, Wenhao Yu, Hongming Zhang, Kaixin Ma, Deng Cai, Zhuosheng Zhang, Hai Zhao, and Dong Yu. Docbench: A benchmark for evaluating llm-based document reading systems, 2024.

# Appendix

## A Prompts Used in HyperSkill

This appendix lists the LLM prompts invoked by HyperSkill, mirroring the prompt-card presentation of HyperGraphRAG [28]. Prompts are grouped by the role they play in the loop of §3: **(a) Frozen-Backbone Inference** (§3) consumed by every skill-augmented rollout, **(b) C1 Skill Distillation** (§3.1) that admits counterfactually checked skill nodes, **(c) C2 Gated Retrieval Injection** (§3.2) that surfaces a question-aware strategy hint, and **(d) the C1 protocol skill and the C3 execution texts** (§3.3). The weighted hypergraph evolution inside C2 is deliberately *prompt-free*: per-node advantage EMA, hyperedge growth and the periodic $u_{\min}$ safeguard prune are closed-form operators on the rollout reward and never call $\pi_\theta$, so no prompt card appears for that block. Every prompt below is reproduced from the released code (markdown emphasis, dashes and symbols normalised to plain text), with `{x}` denoting a runtime placeholder; the user-message wrapper around each system prompt is kept compact since the system prompt already constrains the output schema.

### A.1 Frozen-Backbone Inference Prompts

**Prompt P1: Base Reasoning** (`REASONING_PROMPT`, used on tool-free Text datasets (AIME)). The answer-only base prompt of tool-free datasets; like P2/P3, it is used without retrieved skills whenever Eq. (7) returns the empty skill set (the zero-skill fallback of §3.2), and with the P8 tip line otherwise.

```
Answer the following question. Output ONLY your final answer inside
<answer>...</answer> tags. Do NOT show your work.
```

**Prompt P2: Search Agent** (`SEARCH_AGENT_PROMPT`, used on Text/VL tool tasks with the `text_search` / BM25 channels of §4.1). Drives the per-question multi-round loop in which the agent either commits to an answer or emits exactly one tool tag per round under the regime cap (3 rounds on Text, 5 on VL; Omni datasets use P3) and the shared budget of 2 tool calls.

```
You are an expert research assistant equipped with search tools.
Analyze the question and decide whether to search for more information
or to provide a final answer directly.
{tool_desc}

You must respond in ONE of the following formats:

  Format A - Direct answer (when confident):
    <reason>...</reason>
    <answer>...</answer>

  Format B - Search for more information:
    <reason>...</reason>
    <TOOL_TAG>specific query</TOOL_TAG>

Replace TOOL_TAG with one of the available tools listed above.
```

On WebWalkerQA, for every method, the last two lines of P2 read `<text_search>specific query</text_search>` and "Put the search query itself between the `<text_search>` and `</text_search>` tags."

**Prompt P3: Omni Agent** (`OMNI_AGENT_PROMPT`, used on the Omni regime). The omnimodal counterpart of P2: keeps the same two-format response schema while explicitly instructing the backbone to fuse text, image, audio transcript, and video keyframes in one multimodal call.

```
You are an expert multimodal reasoning assistant. You can process text,
images, audio transcripts, and video descriptions. Analyze all provided
modalities together to derive a precise answer.
{tool_desc}

You must respond in ONE of the following formats:

  Format A - Direct answer (when confident):
    <reason>...</reason>
    <answer>...</answer>

  Format B - Search for more information:
    <reason>...</reason>
    <TOOL_TAG>specific query</TOOL_TAG>

Replace TOOL_TAG with one of the available tools listed above.
```

**Prompt P4: After-Search Continuation** (`AFTER_SEARCH_PROMPT`). Re-enters the agent into another reasoning round with the original question and the freshly retrieved evidence; iterated until the regime-specific round cap ($3$, $5$ or $8$) is reached or an `<answer>` tag closes the reasoning trace.

```
Based on the search results provided, continue your reasoning and either
search again (if needed) or provide the final answer.

Original question: {question}

{search_results}

Now provide your next step:
```

### A.2 C1: Skill Distillation Prompts (§3.1)

**Prompt P5: Contrastive Step Alignment** ($\Phi$ in Eq. (2), `SKILL_CONTRASTIVE_EXTRACT`). Operates on the success/failure split at threshold $\rho = 0.5$: aligns numbered reasoning steps across the two buckets, marks divergence points, and harvests both atomic skills and multi-skill recipes that successes exhibit but failures jointly miss; recipes feed downstream hyperedge construction in §3.2.

```
[system]
You are an expert at analyzing reasoning trajectories to extract reusable
skills. Your task: given a question with GOOD attempts (high reward) and
BAD attempts (low reward), perform step-level contrastive analysis to
find exactly WHERE and WHY the good attempts succeeded.
<Instructions>
1. Decompose each attempt into numbered reasoning steps.
2. Align steps across good and bad attempts - find corresponding steps
   that address the same sub-problem.
3. Identify divergence points - the specific steps where good and bad
   attempts DIFFER in strategy or correctness.
4. Extract atomic skills - from each divergence point, extract the
   generalizable skill that the good attempt used (NOT specific to
   this question).
5. Identify skill recipes - if multiple atomic skills worked TOGETHER
   in successful attempts but were missing together in failures, group
   them as a recipe.
</Instructions>
<Output Format>
<Step Alignment>
[Show the aligned steps of good vs bad attempts, marking divergence
```

```
 points with *]
</Step Alignment>
<Divergence Analysis>
[For each divergence point: what the good attempt did vs what the bad
 attempt did, and WHY the good approach worked]
</Divergence Analysis>
<Atomic Skills>
[One per line, extracted from divergence points:]
SKILL: <name (3-7 words)> | <what to do and when (1-2 sentences)>
</Atomic Skills>
<Skill Recipes>
[Combinations of atomic skills that must be used TOGETHER for success:]
RECIPE: <recipe name> | <skill_1 + skill_2 + ...> |
        <when to apply this combination>
</Skill Recipes>
</Output Format>

[user]
<Question>{question}</Question>
<Gold Answer>{gold_answer}</Gold Answer>
<Good Attempts>{good_attempts}</Good Attempts>
<Bad Attempts>{bad_attempts}</Bad Attempts>
Perform step-level contrastive analysis as instructed.
```

**Prompt P6: Counterfactual Verification** (`SKILL_COUNTERFACTUAL_VERIFY`; $K$-probe vote of Eq. (3)). Each candidate emitted by P5 is paired with up to $K = 2$ failure probes of its own question (Appendix E.7) and admitted only when a strict majority of probes return YES; the $\mathrm{md5}$ name identifier then deduplicates the increment $\mathcal{V}_t^{\text{new}}$ against $\mathcal{V}_t$, so a repeated skill name is skipped.

```
[system]
You are evaluating whether a proposed reasoning skill would help an
agent that FAILED a question.
Given:
- A question the agent got WRONG
- The agent's FAILED reasoning trajectory
- A proposed SKILL
Judge: If the agent had known and applied this skill, would it likely
have answered correctly?
Output EXACTLY one line:
VERDICT: YES | <confidence 0-1> | <brief reason>
or
VERDICT: NO | <confidence 0-1> | <brief reason>

[user]
<Question>{question}</Question>
<Gold Answer>{gold_answer}</Gold Answer>
<Failed Trajectory>{failed_trajectory}</Failed Trajectory>
<Proposed Skill>{skill}</Proposed Skill>
Would this skill have helped? Output VERDICT line only.
```

### A.3 C2: Gated Retrieval Injection Prompts (§3.2)

**Prompt P7: Strategy Hint Synthesis** (`SKILL_SUMMARY_PROMPT`). After the advantage-weighted RRF of Eq. (6) and the similarity/capacity gates of Eq. (7) yield a non-empty $\mathcal{S}_q$, this prompt distils the retrieved skills into a question-aware 2–3 sentence hint that emphasises *how to combine* them rather than enumerating them, so that the context injected into the prompt stays short.

```
Given a question and a set of retrieved reasoning skills, write a concise
```

```
strategy hint (2-3 sentences) that synthesizes these skills specifically
for this question. Focus on HOW to combine them, not just listing them.
Question: {question}
Retrieved Skills:
{skills_raw}
Strategy hint (2-3 sentences, actionable):
```

With the *schema-constrained hint* (SimpleVQA), the following constraint is inserted before the last line of P7; the bracketed clause is added only in tool regimes, with `{limit}` the tool budget.

```
Constraints for the hint: it must not change the required output format
(the agent ends with a short final answer inside <answer>...</answer>, as
short as the question allows, never a full sentence when a name, number
or phrase suffices). [It must not ask for more than {limit} tool calls in
total, and the agent must always finish with a final <answer> even if a
search did not find everything.]
```

**Prompt P8: Skill-Augmented Wrapper** (`SKILL_AUGMENTED_PROMPT`). The hint produced by P7 is prepended to the inference prompt of P1/P2/P3 as a single *tip* line; whenever $|\mathcal{S}_q| = 0$ no retrieved skill enters this wrapper, which then carries only the protocol skill or is skipped; this is the zero-skill fallback that Proposition 3 analyses, with no P7 call for such zero-skill questions.

```
Tip from prior experience: {skills_context}
```

### A.4 C1 Protocol-Skill and C3 Execution Texts (§3.1, §3.3)

These texts are reproduced verbatim from the released code (`strategies/protocol.py`, `prompting.py`, `presearch.py`, `locate_pages.py`, `video.py`); `{x}` is a runtime placeholder and `[...]` names the condition under which a line is used; all remaining text is fixed.

**Prompt P9: Protocol Skill** (C1, injected with the skills context). Instruction lines for the failure types kept in training (at least 2 occurrences; `no_answer` adds none of its own), followed by one answer-form line with word limit `{words}` $= \max(200, \lfloor 0.45 \cdot \text{max new tokens} \rfloor)$.

```
Output protocol (learned from training failures): <lines below>
[malformed_tool] To call a tool, put the query inside that tool's own
  tag, exactly like <{tag}>your query</{tag}>; the tag name is the tool
  name (never write TOOL_TAG and never nest tags).
[over_budget or malformed_tool] Use at most {budget} tool calls in
  total; after the last one, or as soon as you can answer, stop calling
  tools and give your best final answer.
[tool_without_tools] No tool is available for this question: do not
  write any tool call, search request or TOOL_TAG; answer directly from
  the given input.
[math] Reason concisely (the whole response must stay well under
  {words} words) and end with the final integer inside
  <answer>...</answer>; do not use \boxed{}.
[short / default] Always end with your final answer inside
  <answer>...</answer>, as short as the question allows (a name, number,
  short phrase or option letter, not a full sentence).
[data form, option letters] Always end with only the option letter of
  your choice inside <answer>...</answer> (e.g. <answer>B</answer>).
[data form, long golds] Always end with your final answer inside
  <answer>...</answer>, written as one complete sentence (about {n}
  words) that restates the key terms of the question together with the
```

```
  answer.
[data form, frequent abstention] If the input does not contain the
  answer, answer exactly <answer>{frequent}</answer>.
```

**Prompt P10: Protocol Re-injection** (C1/C2, after each tool turn, inside the "`--- Updated Skills ---`" block). Appended to every tool result while the protocol skill is on (line 7 of Algorithm 4), so that both the remaining tool budget and the required final-answer tag stay in view.

```
[calls left] Protocol: you may make {left} more tool call(s), written as
  <{tag}>your query</{tag}>; if the results above suffice, give the
  final answer inside <answer>...</answer> now.
[budget spent] Protocol: you have used all {budget} tool calls. Do not
  call a tool again; give your best final answer now inside
  <answer>...</answer>.
```

**Prompt P11: Final-Answer Turn** (C3). `{form}` is " (only the final integer)" on math, " (only the option letter)" on multiple choice, and empty otherwise; the turn is sent at most once per rollout.

```
[budget spent or repeated query] Tool budget exhausted: no further tool
  call will be executed. Based on the information above, give your final
  answer now inside <answer>...</answer>{form}.
[otherwise] You have not given your final answer yet. Give it now inside
  <answer>...</answer>{form}.
```

**Prompt P12: Answer Suffix and Concrete Tool Prompt** (C3). The suffix follows the question. The concrete tool prompt replaces the `TOOL_TAG` lines of P2/P3 by the real tag and one example; without tools (Omni), it drops the search format instead; the lenient parser accepts placeholders.

```
[math] Reason step by step, then end your response with the final
  integer inside <answer>...</answer> (for example <answer>N</answer>
  for the integer N); do not use \boxed{}.
[multiple choice] End your response with only the letter of the correct
  option inside <answer>...</answer> (for example <answer>X</answer> for
  option X).
[concrete tool prompt, Format B] <{tag}>specific query</{tag}>
  The tag is the tool's own name. Example of a correct tool call:
  <text_search>Eiffel Tower height in metres</text_search>
```

**Prompt P13: Evidence Texts** (C3: question pre-search, locate pages, time grounding, option check). Each text is added only when the dataset enables the matching C3 option (QP, PG$n$, TG, OC in Appendix E.7); the pre-search call is not counted against the per-question tool budget.

```
[pre-search] Results of a knowledge search for the question itself (run
  automatically before your first step):
  [text_search Results]
  {results}
  Use these results if they answer the question; otherwise search
  further.
[locate pages] Candidate pages of {pdf}, located by document search for
  this question (text extracted from the PDF): [page k] {text} ...
  Answer from these pages. If they do not contain the answer, search the
  document again with <doc_search>...</doc_search>. Answer "Not
  answerable" only if the information is still not found after
```

```
  searching. Put the final answer inside <answer>...</answer>.
[time grounding] Video timing: the video lasts {T}. The {n} images are
  keyframes in temporal order, taken at image 1 = {t1}, ... The audio
  track covers the whole video on the same timeline (0:00 to {T}). For
  questions about when something happens, compare its time with the
  total length: the beginning is the first third, the middle the second
  third and the end the last third. Between two keyframes, rely on the
  audio and the neighbouring frames.
[option check] Before answering, check every option against what the
  frames show and what the audio says: for each option write one short
  line 'X: supported / contradicted / no evidence' with the frame time
  or the audio it relies on, then choose the best-supported option.
  Choose an option such as 'not sure', 'unknown', 'it is unclear' or
  'none of the above' only if every other option is contradicted. Never
  decline to answer: always end with exactly one option letter inside
  <answer>...</answer>, for example <answer>B</answer>.
```

### A.5 Per-Baseline Injection Formats and Per-Question Call Budget

**Injection formats.** Each training-free baseline of §4.1 builds its *own* library (Appendix E) and injects its retrieved entries in its own surface form, prepended to the shared inference prompt. The forms, reproduced from the released baseline code with their retrieval depth in parentheses, are:

```
trainingfreegrpo : When solving problems, you MUST first carefully read
                   and understand the helpful instructions and
                   experiences: [<id>]. <experience> ...        (top-10)
skillcraft       : Reusable skills (procedures composed from your
                   operations; apply the most relevant one, then answer
                   in the required format): ‘‘‘python <code>‘‘‘  (top-3)
autoskill        : Relevant skill cards from prior experience:
                   * Skill: <name> -- <description>
                     * Tip: <tip> ...                            (top-3)
evoskill         : > [frontier-skill, val_gain=<gain>] <name>: <desc>
                      1. <instruction> ...                       (top-3)
mementoskills    : [Memory 1] Recall a similar past case:
                   <markdown skill file>                        (top-1)
xskill           : <> Cross-task experiences: - <experience> ...  (top-3)
                   <> Skill (<name>): 1. <step> ...             (top-1)
```

**Per-question LLM call budget.** For one round-$t$ batch question $q$ with $n = 2$ skill-augmented rollouts under regime cap $R \in \{3, 5, 8\}$, the prompts P1–P13 above together fire at most

$$\underbrace{\mathbf{1}\big[|\mathcal{S}_q| > 0\big] + n\,(R{+}1)}_{\text{inference}} \;+\; \underbrace{1}_{\text{distillation}} \;+\; \underbrace{K\,|\widetilde{\mathcal{V}}_q|}_{\text{verification}} \tag{8}$$

LLM calls: P7 once for hint synthesis when the gates of Eq. (7) admit at least one skill, then P1/P2/P3 followed by up to $R-1$ P4 continuations and at most one final-answer turn per rollout; P5 on the $n$-rollout pool whenever at least one success and one failure coexist; and P6 once per (candidate, failure-probe) pair, i.e. at most $2|\widetilde{\mathcal{V}}_q|$ calls at $K = 2$. P8 is a one-line wrapper that fires zero LLM calls. Compared to the backward pass of standard GRPO, this replaces $\Delta\theta$ with $\mathcal{O}(|\widetilde{\mathcal{V}}_q|)$ frozen-backbone calls per question, which benefit from prompt caching since the P5/P6 system messages are identical across calls and only the question, trajectories and candidate skill change per call.

**Reproducibility.** The prompts above, with Eqs. (2)–(3) and the closed-form operators of Eqs. (4)–(7), specify HyperSkill's LLM surface on the unified `gemini-2.5-flash-lite` backbone (§4.1). Evaluation decodes at temperature 0 (self-consistency samples at 0.7); training rollouts use temperature 0.4 in library runs R2/R3 and 0.7 (Text) / 1.0 (VL, Omni) in run R1, distillation 0.3, verification 0.1, and hint synthesis 0. The default hyperparameters are $(n, K, \eta, k_{\mathrm{rrf}}, N_{\max}, \rho, u_{\min}) = (2, 2, 0.1, 60, 3, 0.5, -0.5)$ with protocol threshold 2; the per-dataset floor $\tau$, injection variants and C3 options are listed in Appendix E.7, together with the library run of each dataset.

## B Proofs of Propositions

This appendix proves Propositions 1–3 stated in §3, formalising the three claims that ground counterfactual step-level skill distillation (C1, §3.1), the advantage-weighted hypergraph retrieval of C2 (§3.2), and its gate with zero-skill fallback (C2, §3.2). Notation follows §3: $\pi_\theta$ is the frozen omni-modal backbone, $\mathcal{H}=(\mathcal{V},\mathcal{E},w)$ the weighted skill hypergraph with per-node advantage $u_i\in[-1,1]$, $\mathcal{R}$ the retriever, $f$ the shared encoder, and $r\in[0,1]$ the token-level $F_1$ reward against $a^*$.

### B.1 Proof of Proposition 1

**Proposition 1.** *Counterfactual distillation favours failure-rescuing skills over co-occurring ones.*

*Proof.* Fix a question $q\in\mathcal{B}_t$ and let $\widetilde{\mathcal{V}}_q$ be the candidate set returned by the contrastive aligner of Eq. (2). Each candidate $\widetilde{v}$ has an unobserved *counterfactual success probability*

$$p_{\widetilde{v}} \;=\; \Pr\big[\,\pi_\theta(\cdot\,|\,\widetilde{v},q,a^*,o^-)=\mathrm{YES}\;\big|\;o^-\sim\mathrm{Unif}(\mathcal{O}_q^-)\big] \;\in\; [0,1], \tag{9}$$

which separates candidates into a *failure-rescuing* cohort $\mathcal{K}=\{\widetilde{v}:p_{\widetilde{v}}>\frac{1}{2}\}$ and a *correlation* cohort $\mathcal{N}=\widetilde{\mathcal{V}}_q\setminus\mathcal{K}$ with $p_{\widetilde{v}}\le\frac{1}{2}$; the YES verdict is the operational criterion for rescue, which RQ3(b) checks against observed flips. Modelling the $K$ probes of Eq. (3) as i.i.d. draws $o_1^-,\dots,o_K^-\in\mathcal{O}_q^-$ and writing $S_K(\widetilde{v})=\sum_\ell\mathbf{1}[\pi_\theta(\cdot\,|\,\widetilde{v},q,a^*,o_\ell^-)=\mathrm{YES}]\sim\mathrm{Binomial}(K,p_{\widetilde{v}})$, the admission rule of Eq. (3) selects the survivors of a strict majority vote over the $K$ probes of each candidate:

$$\mathcal{V}_t^{\mathrm{new}}=\big\{\widetilde{v}\in\widetilde{\mathcal{V}}_q:\;S_K(\widetilde{v})>K/2\big\},\;\;\alpha(p)=\sum_{j=\lfloor K/2\rfloor+1}^{K}\binom{K}{j}p^j(1-p)^{K-j}, \tag{10}$$

so the admission rate of any candidate is

$$\Pr\big[\widetilde{v}\in\mathcal{V}_t^{\mathrm{new}}\big] \;=\; \alpha(p_{\widetilde{v}}). \tag{11}$$

Because $\alpha$ is strictly increasing, the gap between cohorts is strict for every $K\ge1$, and for odd $K$ (including the single probe of runs R2/R3) $\alpha$ crosses $\frac{1}{2}$ exactly at $p=\frac{1}{2}$, so the two cohorts separate:

$$\min_{\widetilde{v}\in\mathcal{K}}\alpha(p_{\widetilde{v}}) \;>\; \max_{\widetilde{v}\in\mathcal{N}}\alpha(p_{\widetilde{v}}),\qquad \alpha(p_{\widetilde{v}})\to\mathbf{1}[p_{\widetilde{v}}>\tfrac{1}{2}]\text{ as }K\to\infty\;\;(p_{\widetilde{v}}\neq\tfrac{1}{2}), \tag{12}$$

where the limit follows from Hoeffding's inequality on $|S_K/K-p_{\widetilde{v}}|$. Letting $\beta=\Pr_{\widetilde{v}}[\widetilde{v}\in\mathcal{K}]\in(0,1)$ denote the prior failure-rescuing fraction of $\widetilde{\mathcal{V}}_q$, the library precision after admission obeys

$$\mathrm{Prec}(K) \;=\; \frac{\beta\,\mathbb{E}_{\widetilde{v}\in\mathcal{K}}[\alpha(p_{\widetilde{v}})]}{\beta\,\mathbb{E}_{\widetilde{v}\in\mathcal{K}}[\alpha(p_{\widetilde{v}})]\;+\;(1-\beta)\,\mathbb{E}_{\widetilde{v}\in\mathcal{N}}[\alpha(p_{\widetilde{v}})]} \;>\; \beta \;=\; \mathrm{Prec}(0), \tag{13}$$

since $\mathbb{E}_{\mathcal{K}}[\alpha]>\mathbb{E}_{\mathcal{N}}[\alpha]$ by Eq. (12), where $\mathrm{Prec}(0)$ is the precision without the probe. Eq. (13) further yields $\mathrm{Prec}(K)\to1$ as $K\to\infty$ if no candidate has $p_{\widetilde{v}}=\frac{1}{2}$, so the counterfactual gate raises admission precision, in this model, above the base rate of the candidate pool. Empirically (§4.4), a single vote raises pooled precision from $0.179$ to $0.273$; larger $K$ still keeps the precision above the no-probe base rate ($0.214$ at $K=3,5$ vs $0.179$, pooled over three datasets), as Proposition 1 requires. □

### B.2 Proof of Proposition 2

**Proposition 2.** *Advantage-weighted ranking places high-utility skills above merely similar ones.*

*Proof.* Fix a query $q$ and two skills $v_a,v_b\in\mathcal{V}$ with cosine similarities $s_a=\langle f(q),f(v_a)\rangle$ and $s_b=\langle f(q),f(v_b)\rangle$, and write $\kappa_i=\mathrm{rank}_f(v_i;q)$ for the rank under plain similarity. Under *plain* similarity, retrieval order depends only on the rank pair $(\kappa_a,\kappa_b)$ of the two skills:

$$\mathrm{score}_{\mathrm{plain}}(v_i;q) \;=\; -\kappa_i,\qquad \mathrm{score}_{\mathrm{plain}}(v_a;q)-\mathrm{score}_{\mathrm{plain}}(v_b;q) \;=\; \kappa_b-\kappa_a, \tag{14}$$

so any utility difference is ignored. Under the advantage-weighted RRF rule of Eq. (6),

$$\mathrm{score}(v_i;q) \;=\; \frac{1\;+\;\frac{1}{2}(u_i+1)}{k_{\mathrm{rrf}}\;+\;\kappa_i},\qquad u_i\in[-1,1],\;\;\kappa_i\ge1,\;\;k_{\mathrm{rrf}}=60, \tag{15}$$

the per-skill score factorises into a similarity-rank kernel $1/(k_{\mathrm{rrf}}+\kappa_i)$ and an advantage prior $1+\frac{u_i+1}{2}=\frac{3+u_i}{2}\in[1,2]$, so a less similar skill $v_b$ ($\kappa_b>\kappa_a$) is ranked above $v_a$ exactly when its advantage prior outweighs the rank gap between the two similarity kernels,

$$\mathrm{score}(v_b;q)\ >\ \mathrm{score}(v_a;q)\quad\Longleftrightarrow\quad\frac{3+u_b}{3+u_a}\ >\ \frac{k_{\mathrm{rrf}}+\kappa_b}{k_{\mathrm{rrf}}+\kappa_a},\tag{16}$$

which is the per-item quality signal that flat similarity cannot expose; with $k_{\mathrm{rrf}}=60$, a skill one rank lower overtakes its neighbour once its prior is larger by a factor above $1+1/(60+\kappa_a)$. To see that $u_i$ tracks utility, observe that the EMA update of Eq. (4) admits a contraction-mapping fixed-point analysis: let $\bar{r}_{i,t}$ be the mean reward of the question that retrieved $v_i$ at round $t$; with the advantage and recursion of Eq. (4),

$$\hat{A}_{i,t}\ =\ \tanh\Big(\frac{\bar{r}_{i,t}-\bar{r}_{\mathcal{B}_t}}{\sigma_{r,\mathcal{B}_t}+\epsilon}\Big)\in(-1,1),\qquad u_{i,t+1}\ =\ (1-\eta)\,u_{i,t}+\eta\,\hat{A}_{i,t},\tag{17}$$

the EMA recursion yields, by stationarity of the rollout distribution,

$$\lim_{t\to\infty}\mathbb{E}[u_{i,t}]\ =\ \mathbb{E}\big[\hat{A}_{i,\infty}\big],\qquad\mathbb{E}\big[\hat{A}_{i,\infty}\big]\ >\ 0\ \text{ if }\ \Delta_i\text{ is symmetric about a positive mean},\tag{18}$$

where $\Delta_i$ is the argument of $\tanh$ in Eq. (17) and the implication holds because $\tanh$ is odd and increasing; so $\mathbb{E}[u_{i,\infty}]>0$ when retrieving $v_i$ goes with above-average rewards. Combining Eq. (16) with Eq. (18), skills of nearby rank are ordered by their long-run utility, and the multiplicative prior of Eq. (15) cannot be replicated by any rank-only retriever, including the median-frontier filter $\mathcal{L}_{\mathrm{frt}}$ of EvoSkill, which exposes only a binary quality cut. Hence Proposition 2 is established. □

### B.3 Proof of Proposition 3

**Proposition 3.** *Gating with zero-skill fallback never harms the skill-free pipeline in expectation.*

*Proof.* Let $\mathrm{ZS}(q)=\pi_\theta(\cdot\,|\,q,m)$ denote the skill-free pipeline (no retrieved skill; protocol skill and C3 options unchanged) and $\mathrm{HS}(q,\mathcal{S}_q)=\pi_\theta(\cdot\,|\,q,m,\mathcal{S}_q)$ the gated skill-augmented policy of Eq. (7), with reward $r(\pi;q,a^*)\in[0,1]$. Partition the query distribution by the gate output,

$$\mathcal{Q}_0\ =\ \big\{q:|\mathcal{S}_q|=0\big\},\qquad\mathcal{Q}_1\ =\ \big\{q:|\mathcal{S}_q|\geq 1\big\},\qquad p_0=\Pr[\mathcal{Q}_0],\ \ p_1=1-p_0,\tag{19}$$

where $p_0$ is exactly the zero-skill fallback rate, reported as the zero-skill fraction in Figure 7(a) (0 at the frozen $\tau$ on both datasets, rising to $0.63$ and $0.55$ at the largest $\tau$). The fallback clause of Eq. (7) gives a pointwise identity on $\mathcal{Q}_0$,

$$\forall q\in\mathcal{Q}_0\colon\ \ \mathrm{HS}(q,\mathcal{S}_q)\ \equiv\ \mathrm{ZS}(q)\ \ \Longrightarrow\ \ \mathbb{E}_{q\sim\mathcal{Q}_0}\big[r(\mathrm{HS})-r(\mathrm{ZS})\big]\ =\ 0,\tag{20}$$

so the zero-skill branch contributes zero difference. On $\mathcal{Q}_1$ every injected skill obeys the two gates of Eq. (7),

$$\langle f(q),f(v_i)\rangle\ \geq\ \tau\quad\forall v_i\in\mathcal{S}_q,\qquad|\mathcal{S}_q|\ \leq\ N_{\max},\tag{21}$$

so the prompt receives at most $N_{\max}$ skills each carrying mutual information at least

$$I\big(v_i;a^*\,|\,q\big)\ \geq\ \iota_\tau\ :=\ \inf_{v:\langle f(q),f(v)\rangle\geq\tau}I(v;a^*\,|\,q)\ \geq\ 0,\tag{22}$$

where $a^*$ is the gold answer and $\iota_\tau$ is the similarity-floor information bound. Under the gate-admissibility assumption that, at budget $N_{\max}$, skills clearing both gates do not lower the expected reward of the frozen $\pi_\theta$ (formally, $\mathbb{E}[r(\mathrm{HS})]\geq\mathbb{E}[r(\mathrm{ZS})]$ whenever every injected skill clears the floor $\tau$; marginal skills below $\tau$ need not, §3.2), the gated branch never lowers the expected reward,

$$\mathbb{E}_{q\sim\mathcal{Q}_1}\big[r(\mathrm{HS})\big]\ \geq\ \mathbb{E}_{q\sim\mathcal{Q}_1}\big[r(\mathrm{ZS})\big],\quad\text{equality if }\pi_\theta\text{ ignores the injections}.\tag{23}$$

Combining Eq. (20) and Eq. (23) via the law of total expectation yields the unconditional safety bound

$$\mathbb{E}_q\big[r(\mathrm{HS})\big]\ -\ \mathbb{E}_q\big[r(\mathrm{ZS})\big]\ =\ p_1\cdot\big(\mathbb{E}_{q\sim\mathcal{Q}_1}\big[r(\mathrm{HS})-r(\mathrm{ZS})\big]\big)\ +\ p_0\cdot 0\ \geq\ 0,\tag{24}$$

so HS is no worse than the skill-free pipeline in expectation. The argument is independent of $|\mathcal{V}|$, so it applies uniformly across cold-start (high $p_0$, small $|\mathcal{V}|$) and mature regimes (low $p_0$, large $|\mathcal{V}|$); on a mature library the bound of Eq. (23) covers most queries, while on a cold-start library Eq. (20) dominates and HS recovers ZS exactly. The same decomposition further yields a quantitative envelope on the gain

$$0\ \leq\ \mathbb{E}_q\big[r(\mathrm{HS})-r(\mathrm{ZS})\big]\ =\ p_1\cdot\Big(\mathbb{E}_{q\sim\mathcal{Q}_1}\big[r(\mathrm{HS})\big]-\mathbb{E}_{q\sim\mathcal{Q}_1}\big[r(\mathrm{ZS})\big]\Big),\tag{25}$$

so the improvement scales with the non-fallback mass $p_1=1-p_0$, and the room for improvement grows as $p_0$ shrinks. Hence Proposition 3 is established for every library size $|\mathcal{V}|$. □

# C HyperSkill Algorithm Details

This appendix expands §3 into executable pseudocode in the algorithm-card style of HyperGraphRAG [28]. Algorithm 1 realises C1 (§3.1, Eqs. (2)–(3)) and records the failure types of the protocol skill; Algorithms 2 and 3 realise the training side (Eqs. (4)–(5)) and the retrieval side (Eqs. (6)–(7)) of C2 (§3.2); and Algorithm 4 realises the execution protocol of C3 (§3.3).

**C1: Training-Time Skill Distillation.** Each question $q \in \mathcal{B}_t$ receives $n$ skill-augmented rollouts, split at $\rho$ into $\mathcal{O}_q^+$ and $\mathcal{O}_q^-$. The regex decomposer $\Psi$ slices each rollout into steps at zero LLM cost, and the aligner $\Phi$ returns the candidate set $\widetilde{\mathcal{V}}_q$ of divergent atomic skills. Each candidate is probed against a list $\mathcal{P}_q$ of at most $K$ failure probes of its own question and is admitted only on a strict majority of YES verdicts; $\mathrm{sid}$ deduplicates it against $\mathcal{V}_t$, and every recipe with at least two admitted members joins $\mathcal{E}_t^{\mathrm{rec}}$. Every rollout that ends without an answer also increments the count $c_\phi$ of its failure type $\phi$; a type enters the protocol skill (Prompt P9) once $c_\phi \geq 2$, already for the remaining training rollouts when the skill is on, and the final counts define the protocol skill used at test time.

**Algorithm 1** C1: Training-Time Skill Distillation (one batch)

**Require:** Batch $\mathcal{B}_t$, frozen $\pi_\theta$, hypergraph $\mathcal{H}_t$, threshold $\rho$, probe count $K$, current $\mathcal{V}_t$
**Ensure:** Round-$t$ admission set $\mathcal{V}_t^{\mathrm{new}} \subseteq \bigcup_{q \in \mathcal{B}_t} \widetilde{\mathcal{V}}_q$ and recipes $\mathcal{E}_t^{\mathrm{rec}}$

$\mathcal{V}_t^{\mathrm{new}} \leftarrow \emptyset$
**for** each question $q \in \mathcal{B}_t$ **do**
  Sample $\{o^{(\ell)}\}_{\ell=1}^{n} \sim \pi_\theta(\cdot \mid q, m, \mathcal{R}(q, m, \mathcal{H}_t))$ and score $r(o^{(\ell)}; a^*)$ {rollout group}
  For each $o^{(\ell)}$ without a final answer, $c_\phi \leftarrow c_\phi + 1$ for its failure type $\phi$ {protocol statistics}
  Split $\mathcal{O}_q^+ \leftarrow \{o^{(\ell)} : r \geq \rho\}$, $\mathcal{O}_q^- \leftarrow \{o^{(\ell)} : r < \rho\}$
  $\Psi(o^{(\ell)}) \leftarrow (s_1, \ldots, s_{T_o})$ via regex decomposition {zero LLM cost}
  $\widetilde{\mathcal{V}}_q \leftarrow \Phi(\{\Psi(o^+)\}_{\mathcal{O}_q^+}, \{\Psi(o^-)\}_{\mathcal{O}_q^-})$ {contrastive alignment → candidates}
  **for** each candidate $\widetilde{v} \in \widetilde{\mathcal{V}}_q$ **do**
    Take $\leq K$ probes $\mathcal{P}_q$ on own failures $\mathcal{O}_q^-$ and query $\pi_\theta(\cdot \mid \widetilde{v}, q, a^*, o^-)$ for each $o^- \in \mathcal{P}_q$
    **if** $\sum_{o^- \in \mathcal{P}_q} \mathbf{1}[\pi_\theta(\cdot \mid \widetilde{v}, q, a^*, o^-) = \mathrm{YES}] > |\mathcal{P}_q|/2$ **and** $\mathrm{sid}(\widetilde{v}) \notin \mathcal{V}_t$ **then**
      $\mathcal{V}_t^{\mathrm{new}} \leftarrow \mathcal{V}_t^{\mathrm{new}} \cup \{\widetilde{v}\}$ {counterfactual admission}
    **end if**
  **end for**
**end for**
**return** $\mathcal{V}_t^{\mathrm{new}}$ and $\mathcal{E}_t^{\mathrm{rec}}$ (recipes with at least two admitted members)

*Complexity Analysis.* Per batch, the rollouts cost $\mathcal{O}(|\mathcal{B}_t| \cdot n)$ backbone calls, $\Psi$ is free, $\Phi$ fires once per group ($\mathcal{O}(|\mathcal{B}_t|)$ calls) and the probe adds $K$ calls per candidate ($\mathcal{O}(|\mathcal{B}_t| \cdot |\widetilde{\mathcal{V}}_q| \cdot K)$); with $(n, K) = (2, 2)$ and an empirical $|\widetilde{\mathcal{V}}_q| \leq 3$, this is at most 6 probes per question, so rollout generation dominates.

**C2 (training side): Weighted Hypergraph Evolution.** Every admitted skill is embedded by $f$ and stored in the dual KV/VDB index with initial advantage $u_i = 0.3$. The batch-normalised advantage $\hat{A}_q$ of each question drives the EMA of Eq. (4) on its retrieved skills; a high-reward question ($\bar{r}_q \geq \rho$) that retrieves two or more skills adds them as a hyperedge (or EMA-updates its weight), and a periodic safeguard prune evicts every node with $u_i < u_{\min}$. None of these steps calls the LLM. Training retrieval differs from test retrieval in three ways: it uses an advantage floor of 0, so a skill whose advantage turns negative is no longer reinforced; for VL items its query embedding averages the question and image embeddings; and after each tool call the rollout refreshes its skills (query: question and tool query) in an "Updated Skills" block that receives no EMA credit.

*Complexity Analysis.* Each batch issues at most $|\mathcal{V}_t^{\mathrm{new}}|$ skill embedding calls and one per new hyperedge, $\mathcal{O}(\sum_{q \in \mathcal{B}_t} |\mathcal{S}_q|)$ scalar EMA updates, and an $\mathcal{O}(|\mathcal{V}_{t+1}|)$ periodic prune; no step calls $\pi_\theta$.

**C2 (test side): Gated Retrieval and Injection.** Every query $q$ is embedded by the same encoder $f$ as the skills. Its $k = 100$ nearest nodes form the candidate pool, and hyperedges that match $q$ add their member skills to this pool (scored at $0.8\times$ the hyperedge cosine). The similarity floor $\tau$ discards weak matches, the advantage-weighted RRF score ranks the survivors, and the capacity cap $N_{\max}$ keeps the top ones. If no candidate survives, no retrieved skill is injected. The protocol skill of C1, when enabled for the dataset, is injected for every question regardless of the gate.

**Algorithm 2** C2 (training side): Weighted Hypergraph Evolution under Retrieval Feedback

**Require:** $\mathcal{H}_t = (\mathcal{V}_t, \mathcal{E}_t, w)$, $\mathcal{V}_t^{\text{new}}$, rewards $\{\bar{r}_q\}$, retrieved sets $\{\mathcal{S}_q\}$, recipes $\mathcal{E}_t^{\text{rec}}$, $\eta$, $\rho$, $u_{\min}$
**Ensure:** Updated $\mathcal{H}_{t+1} = (\mathcal{V}_{t+1}, \mathcal{E}_{t+1}, w)$

Embed $f(v_i)$ for every $v_i \in \mathcal{V}_t^{\text{new}}$ and append to the dual KV/VDB index with initial $u_i = 0.3$
Compute statistics $\bar{r}$, $\sigma_r$ over all rollouts of the batch; $\mathcal{E}_{t+1} \leftarrow \mathcal{E}_t \cup \mathcal{E}_t^{\text{rec}}$ (recipe weight 0)
**for** each question $q \in \mathcal{B}_t$ with $\mathcal{S}_q \neq \emptyset$ **do**
    $\hat{A}_q \leftarrow \tanh\big((\bar{r}_q - \bar{r})/(\sigma_r + \epsilon)\big)$; $u_i \leftarrow \text{clip}_{[-1,1]}\big((1-\eta)u_i + \eta \hat{A}_q\big)$ for $v_i \in \mathcal{S}_q$ {EMA}
    **if** $\bar{r}_q \geq \rho$ **and** $|\mathcal{S}_q| \geq 2$ **then**
        Add $e = \mathcal{S}_q$ with $w(e) \leftarrow \max(0, \hat{A}_q)$, or EMA-update $w(e)$ if $e \in \mathcal{E}_{t+1}$ {co-occurrence}
    **end if**
**end for**
$\mathcal{V}_{t+1} \leftarrow \{v_i \in \mathcal{V}_t \cup \mathcal{V}_t^{\text{new}} : u_i \geq u_{\min}\}$ {safeguard prune, periodic}
Pruned members of a hyperedge are skipped whenever it is expanded at retrieval {edges kept}
**return** $\mathcal{H}_{t+1} = (\mathcal{V}_{t+1}, \mathcal{E}_{t+1}, w)$

**Algorithm 3** C2 (test side): Gated Retrieval, Injection and Zero-Skill Fallback

**Require:** Query $(q, m)$, hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E}, w)$, encoder $f$, constant $k_{\text{rrf}}$, floor $\tau$, cap $N_{\max}$
**Ensure:** Injected context $\mathcal{C}(q)$ for the C3 rollout

$\mathcal{S}_q^{\text{pool}} \leftarrow \text{Top-}k_{\cos}(\mathcal{V}) \cup \text{expand}_q(\mathcal{E})$ by cosine similarity $\langle f(q), f(v) \rangle$ {candidate pool}
$\mathcal{S}_q^{\tau} \leftarrow \{v \in \mathcal{S}_q^{\text{pool}} : \langle f(q), f(v) \rangle \geq \tau\}$ {similarity-floor gate}
$\text{score}(v_i; q) \leftarrow (1 + \frac{1}{2}(u_i + 1))/(k_{\text{rrf}} + \text{rank}_f(v_i; q))$ for $v_i \in \mathcal{S}_q^{\tau}$ {advantage-weighted RRF}
$\mathcal{S}_q \leftarrow \text{Top-}N_{\max}(\text{score}(\cdot\,; q),\ \mathcal{S}_q^{\tau})$ {capacity-cap gate}
**if** $|\mathcal{S}_q| = 0$ **then**
    $\mathcal{C}(q) \leftarrow \emptyset$ {zero-skill fallback (Eq. (7))}
**else**
    $\mathcal{C}(q) \leftarrow \text{P7}(q, \mathcal{S}_q)$ or the terse list of $\mathcal{S}_q$ {skill injection}
**end if**
Append the protocol skill (P9) to $\mathcal{C}(q)$ if enabled {ungated}
**return** $\mathcal{C}(q)$

*Complexity Analysis.* Per query, retrieval is one exact (brute-force) cosine search at $\mathcal{O}(|\mathcal{V}| + |\mathcal{E}|)$, both gates are linear in the candidate pool, and the prompt grows by at most $N_{\max}$ skills.

**C3: Execution Protocol.** Every rollout runs the agent loop of §3.3 with the dataset's C3 options: the prompt carries the injected context, the question, and the suffix and evidence texts of Prompts P12–P13; tool calls are parsed strictly or leniently; and a final-answer turn (P11) closes rollouts without an answer. With self-consistency, $k_{\text{sc}}$ samples vote by plurality (greedy answer on a tie).

**Algorithm 4** C3: Execution Protocol

**Require:** Query $(q, m)$, context $\mathcal{C}(q)$, tools $\mathcal{T}$, tool budget $B$, round cap $R$, C3 options and $k_{\text{sc}}$ (Appendix E.7)
**Ensure:** Final answer $\hat{a}(q)$

$x \leftarrow \mathcal{C}(q) \,\|$ prompt (concrete tool tag if enabled) $\| \, q \,\|$ suffix, located pages, video timing, option check
If pre-search: $x \leftarrow x \,\|$ `text_search`$(q)$ {not counted in $B$}
**for** rollout $j \in \{\text{greedy}\} \cup \{1, \ldots, k_{\text{sc}}\}$ (temperature 0 / 0.7) **do**
    **for** round $= 1, \ldots, R$ **do**
        $y \leftarrow \pi_\theta(x)$; $x \leftarrow x \,\|\, y$; **break** if $y$ has a closed `<answer>`
        Parse a tool call (strict, then lenient forms); **break** if none, repeated, or $B$ spent
        $x \leftarrow x \,\|$ tool result $\|$ protocol re-injection (P10) {if protocol skill on}
    **end for**
    If final turn and no closed answer: $x \leftarrow x \,\|$ P11; $y \leftarrow \pi_\theta(x)$
    $a_j \leftarrow$ last `<answer>` of the rollout
**end for**
$\hat{a}(q) \leftarrow$ most frequent answer of $\{a_1, \ldots, a_{k_{\text{sc}}}\}$ if unique, else $a_{\text{greedy}}$
**return** $\hat{a}(q)$

**End-to-end pipeline.** In every training batch, Algorithm 3 with the training settings feeds the rollouts of Algorithm 1, whose admissions and rewards drive Algorithm 2; at evaluation the library is frozen and Algorithms 3 and 4 serve each test question. The nested ablations of Figure 4 (w/o C2 contains w/o C1; w/o C3 stands alone) act at evaluation on the trained library (Appendix E.5).

# D Benchmark Details

This appendix expands §4.1 into a reference for *HyperSkill-Bench*, which curates 11 public datasets, with no new data, into one training-free skill-evolution benchmark with a unified JSONL schema. §D.1 covers the sourcing and §D.2 the evaluation; $q$, $a^*$ and $m \in \mathcal{M}$ are as in §3.

## D.1 Dataset Details

We collect 11 public datasets, group them into four regimes by dominant modality, and convert every sample into a unified JSONL record $\mathrm{rec}(s) = (q, a^*, m, \text{task_type}, \text{resource_type}, \text{assets})$ where `assets` stores the relative paths of any images, PDFs, video or audio referenced by $q$; we sample 128 training questions per dataset (deduplicated by exact question text, seed 42; for AIME2024/2025 from DAPO-Math [52] and for WebWalkerQA from AFM-RL [24]) and evaluate on 30 test questions for AIME2024/2025 and 64 for each of the other nine datasets (636 questions in total).

**Text reasoning.** *AIME2024* and *AIME2025* [31, 32] are the canonical math-reasoning probes used by RL-trained reasoners [16, 49]; resources are empty and answers are integers. *WebWalkerQA* [45] contributes open-web agent QA; as on VL-Web, every method answers it through the `text_search` tool (§4.1), which `gemini-3.8-flash` serves as a knowledge source in place of live web search.

**VL-Web (web-search VQA).** *FVQA* [46], *InfoSeek* [5], *SimpleVQA* [6] and *LiveVQA* [14] pair an image $I_q$ with a question whose answer requires entity-level web evidence; the four datasets stress different difficulty axes: FVQA on factual visual knowledge, InfoSeek on long-tail entities, SimpleVQA on everyday recognition, and LiveVQA on recent, time-sensitive events.

**VL-Doc (long-document QA).** *DocBench* [59] and *MMLongBench-Doc* [29] pair a multi-page PDF with a question grounded in a small number of pages; resources are PDF paths, and a per-document BM25 chunk index is built once and shared by every method; search is restricted to the question's own PDF, and image-only PDFs of MMLongBench-Doc are indexed through local OCR. Score differences on VL-Doc therefore reflect how each method uses this shared index.

**Omni (audio-video reasoning).** *OmniVideoBench* [23] pairs a video with a question whose answer requires temporal grounding across visual frames, while *WorldSense* [18] pairs a video clip and its audio track with a question that requires fusing visual and acoustic cues, delivered as 32 uniformly sampled keyframes plus the dataset's audio track (extracted with `ffmpeg` when missing) in one call; OmniVideoBench questions receive 8 keyframes. The same frame counts apply to every method.

## D.2 Evaluation Details

All metrics are percentages. Let $\mathrm{tok}(\cdot)$ denote whitespace tokenisation after lowercasing and removing punctuation and articles; the answer-shape branch follows the released evaluator.

**Token-level $F_1$ (primary).** The prediction $\hat{a}$ is the content of the last `<answer>` tag, or the whole response if there is none. Token-$F_1$ with answer-shape awareness applies option-letter matching when $a^*$ is an option letter, substring or token-set containment when $a^*$ is a short entity of at most 5 tokens, and SQuAD-style token-$F_1$ otherwise. The main table uses a CJK-aware tokeniser that splits Chinese, Japanese and Korean text into single characters and otherwise splits at whitespace: $F_1(\hat{a}, a^*) = 2|\mathrm{tok}(\hat{a}) \cap \mathrm{tok}(a^*)|/(|\mathrm{tok}(\hat{a})| + |\mathrm{tok}(a^*)|)$, reported as $\mathrm{R}_{F_1} = 100 \cdot F_1$.

**Exact Match and format score.** Neither is reported. On AIME the gold is a short integer, and the containment rule above applies to the answer tag, or to the full response without a tag, for every column. The format score (final answer inside an `<answer>` tag) is logged in training only.

**Reward and reported statistic.** The GRPO advantage of §3.2 is computed against the token-$F_1$ reward $r = F_1(\hat{a}, a^*)$ alone (whitespace tokeniser; Table 2 uses the CJK-aware variant). Every cell of Table 2 and Figure 4 is the full-set mean of the per-question $F_1$ from a single evaluation run; a rollout with a provider error scores 0, as for two OmniVideoBench questions in every column. Differences are tested with a paired bootstrap over questions (10k resamples). HyperSkill's margins over the best other column have 95% intervals of $[-4.1, +8.3]$ on WebWalkerQA, $[-2.3, +10.0]$ on SimpleVQA, $[-7.3, +9.8]$ on LiveVQA, $[-4.3, +13.0]$ on DocBench and $[-6.3, +15.0]$ on WorldSense. Re-running HyperSkill with a freshly trained library changed the 11-dataset average by $+0.08$ (95% interval $[-1.38, +1.54]$) and single datasets by up to 3.2 points (SimpleVQA).

# E Baseline Details

This appendix expands Table 1 into a per-method specification: for Zero-shot and the 6 training-free baselines, the 3 HyperSkill ablations, and HyperSkill itself, we report *(i) Overview* (purpose and substrate), *(ii) Per-task implementation* (admission/retrieval/update) and *(iii) Hyperparameters*. Methods are grouped into Zero-shot and three paradigm classes (§E.1–§E.4), followed by the ablations (§E.5) and HyperSkill (§E.6). Throughout we reuse $\pi_\theta$ (frozen `gemini-2.5-flash-lite`), $f$ (shared query/skill encoder), Top-$k\langle\cdot,\cdot\rangle$ (cosine top-$k$), and the reward $r$ of §D.2; every method shares this backbone, embedder and tool stack. Each baseline builds its *own* library from the same 128 training questions with a re-implementation of its published procedure, using the backbone for every LLM call; the baselines retrieve by cosine without a similarity floor and do not receive C3.

## E.1 Zero-shot Baseline

**Zero-shot [54].** *(i) Overview.* Bare frozen LLM with task-specific tools and no skill library: every question is answered from $\pi_\theta(\cdot \mid q, m)$. Substrate: empty $\mathcal{H} = (\emptyset, \emptyset, w)$. *(ii) Per-task implementation.* The agent may call the regime's tool (`text_search` on WebWalkerQA and VL-Web, BM25 on VL-Doc; Omni receives keyframes, plus audio on WorldSense) and writes the final answer inside `<answer>...</answer>`; admission, advantage update, and gating are all unsupported. *(iii) Hyperparameters.* At most 3 (Text), 5 (VL) or 8 (Omni) rounds and 2 tool calls; greedy decoding at temperature 0 with 1024 new tokens (2048 on AIME2025, 4096 on WorldSense).

## E.2 Trajectory-Level Skill Pools

**TrainingFreeGRPO [3].** *(i) Overview.* Replaces gradient updates with an LLM-curated experience pool $\mathcal{L} = \{x_i\}$ that distils group-relative *semantic* advantage from each rollout group and injects retrieved experiences as token-prior hints. Substrate: flat pool indexed by $f$. *(ii) Implementation in our protocol.* Builds its own experience pool from on-policy groups of $n = 2$ rollouts under the evolving pool (1 epoch, batches of 4): for partially correct groups, each rollout is summarised, a group-advantage call extracts at most one semantic experience, and group and batch update calls apply ADD/UPDATE/DELETE operations. The top experiences by cosine are injected as `[<id>] <experience>` (Appendix A). *(iii) Hyperparameters.* top-$k = 10$; rollout temperature of run R1.

**EvoSkill [2].** *(i) Overview.* Maintains a Pareto frontier of structured skill folders, each created by failure-driven proposal and retained only if it beats the held-out median. Substrate: $\mathcal{L}_{\text{frt}} = \{s_i : g_i \geq \text{med}(g)\}$ with one validated gain $g_i$ per skill. *(ii) Implementation in our protocol.* Runs 3 failure-driven iterations: a proposer sees 8 failed training questions and the current skills and proposes one skill (name, description, at most 5 instructions); the candidate program is kept only if its greedy F1 on 16 held-out training questions does not drop. The final program's top skills are injected as `> [frontier-skill, val_gain=<gain>] <name>: <description>` with numbered instructions. *(iii) Hyperparameters.* top-$k = 3$; 3 iterations of 8 failed questions each.

## E.3 Code/Card Skill Caches

**SkillCraft [4].** *(i) Overview.* Auto-composes atomic tools into reusable Skills cached as Python-style `def` $s_i(\cdot)$ snippets, with cache-hit retrieval and no quality ranking. Substrate: code cache $\mathcal{L} = \{$ `def` $s_i(\cdot)\}$. *(ii) Implementation in our protocol.* Composes each question's best successful trajectory (reward $\geq 0.5$) into one Python-style function over the agent's atomic operations; a skill is cached only if it parses as one documented function, and a name or docstring match (cosine $\geq 0.90$) counts as a cache hit. The top skills are injected as a `python` code block. *(iii) Hyperparameters.* top-$k = 3$; success threshold 0.5, cache-hit cosine 0.90 and one composition call per success.

**AutoSkill [50].** *(i) Overview.* Distils skills from interaction traces into standardised cards (`name + tip`) and injects the top matches into future requests for lifelong personalisation. Substrate: tip-card store $\mathcal{L} = \{\text{card}_i\}$ with no per-card quality. *(ii) Implementation in our protocol.* Extracts a card (name, description, at most 3 tips) from each question's best successful trace and inserts it, merging it by an LLM call with the nearest card when their cosine is at least 0.85. The top cards are injected as `* Skill: <name> -- <description>` with `* Tip:` bullets. *(iii) Hyperparameters.* top-$k = 3$; merge threshold 0.85; at most 3 tips per card and one merge call per near-duplicate.

### E.4 Memory-Stream Skills

**Memento-Skills [56].** *(i) Overview.* Stores reusable skills as Markdown files in a stateful memory and selects them through a read/write reflective router conditioned on the current prompt. Substrate: Markdown memory $\mathcal{L} = \{m_i^{\mathrm{md}}\}$ plus a behaviour-trainable router. *(ii) Implementation in our protocol.* Processes the training cases in order: a top-1 cosine router reads the most similar Markdown skill file, and one reflective call rewrites it, adds a new file, or does nothing. At test time the routed file is injected after `[Memory 1] Recall a similar past case:`. *(iii) Hyperparameters.* top-$k = 1$; the behaviour-trained router is approximated by cosine; one reflective call per case.

**XSkill [19].** *(i) Overview.* Dual-stream framework that distils *experiences* for tool selection and *skills* for task orchestration, both grounded in visual observations and merged via cross-rollout critique. Substrate: dual stream $\{\mathcal{X}_{\mathrm{exp}}, \mathcal{L}_{\mathrm{skill}}\}$ jointly indexed by $f$. *(ii) Implementation in our protocol.* For every training question, one critique call compares the two rollouts, grounded by a backbone-generated image caption, and outputs at most 2 experiences ("When <condition>, <action>") and one workflow skill; near-duplicates are dropped. The query embedding $f(q, I)$ averages the text embedding with the embedding of the image caption; the top experiences and the top skill are injected after `<> Cross-task experiences:` and `<> Skill (<name>):`. *(iii) Hyperparameters.* top-3 experiences and top-1 skill; retrieval queries the whole question, not its sub-tasks.

### E.5 HyperSkill Ablations

The ablation w/o C2 contains w/o C1, and w/o C3 is applied on its own; all three act at evaluation on HyperSkill's trained library and keep every other setting of the dataset's configuration (Appendix E.7); none of them retrains the library or re-tunes a setting specifically for the ablated system.

**w/o C1.** *(i) Overview.* HyperSkill without the knowledge distilled in training. This row drops the retrieved skills and replaces the learned protocol skill by one generic text used for every dataset, without re-injection. *(ii) Per-task implementation.* The floor admits no skill ($\tau = 9$); hub correction, P7, the schema-constrained hint and the answer-form line are off. Where the configuration enables the protocol skill, the context is "Output protocol: Answer the question. If tools are available, call them using the tool formats described above. End your response with your final answer inside `<answer>...</answer>`." *(iii) Hyperparameters.* C3 options are left unchanged.

**w/o C2.** *(i) Overview.* HyperSkill without test-time delivery: nothing from the library is injected. *(ii) Per-task implementation.* As w/o C1, and the protocol skill is also off. On AIME2025, WebWalkerQA and MMLongBench-Doc, whose configurations use neither retrieved skills nor the protocol skill, w/o C1 and w/o C2 are identical. *(iii) Hyperparameters.* C3 options unchanged.

**w/o C3.** *(i) Overview.* HyperSkill without the execution protocol. *(ii) Per-task implementation.* The C3 options are reset to their defaults: no concrete tool prompt, final-answer turn, answer suffix, lenient parser, voting, located pages, question pre-search, time grounding or option check, and the default token budget (1024 new tokens; 4096 and 32 frames on WorldSense), prompt-length limit and round and tool limits. *(iii) Hyperparameters.* C1, C2 and the library unchanged.

### E.6 HyperSkill

**HyperSkill.** *(i) Overview.* Counterfactual step-level distillation (C1, §3.1) admits failure-rescuing skills and learns the protocol skill; retrieval feedback updates each skill's advantage and grows hyperedges (§3.2); an advantage-weighted RRF retriever with similarity/capacity gates and a zero-skill fallback (C2, §3.2) injects only confident matches; and the execution protocol (C3, §3.3) runs the test rollout. Substrate: weighted hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E}, w)$ with $u_i \in [-1, 1]$. *(ii) Per-task implementation.* Each batch applies Eqs. (2)–(3) for admission, Eqs. (4)–(5) for evolution, and Eqs. (6)–(7) for retrieval; evaluation adds the C3 options of the dataset. *(iii) Hyperparameters.* Defaults of §3, protocol threshold 2 and the per-dataset settings of Appendix E.7.

### E.7 Per-Dataset HyperSkill Configuration

**Library runs.** The three library runs all train for one epoch on the same 128 training questions with $n = 2$, batch size 4 and seed 1: **R1** uses temperature 0.7 (Text) / 1.0 (VL, Omni), early stopping, a training-time floor of 0.2, and two counterfactual votes on the candidate's single failed rollout that

must both return YES; **R2** uses temperature $0.4$, no early stopping, P7 hints during training rollouts, $K_{\text{eff}}$ probed own failures, and a training-time floor of $0.55$; **R3** equals R2 with a training-time floor of $0.2$. On AIME2024/2025, WebWalkerQA, FVQA and OmniVideoBench training rollouts also use the concrete tool prompt, final-answer turn, answer suffix and lenient parser, with $8192$ (AIME) or $4096$ tokens; the protocol statistics classify each rollout before its final-answer turn.

**Counterfactual probes.** Runs R2/R3 probe $K_{\text{eff}} = \min(K, |\mathcal{O}_q^-|)$ distinct failures of the candidate's own question and take the strict majority of Eq. (3) over $K_{\text{eff}}$; with $n = 2$, a question that yields candidates has exactly one failed rollout (alignment needs a success and a failure), so $K_{\text{eff}} = 1$. Admitted skills enter the library with $u_i = 0.3$, after which only the EMA of Eq. (4) changes $u_i$.

**Similarity floor.** The floor $\tau$ of Eq. (7) is set per dataset. Test question-skill cosines of $f$ lie in $[0.31, 0.80]$, so a floor of $0.2$ keeps every candidate and $\tau = 9$ admits none. Retrieved skills are injected on InfoSeek (hub-corrected $\tilde{\tau} = -0.02$), SimpleVQA ($\tau = 0.2$) and OmniVideoBench ($\tau = 0.55$). Elsewhere $\tau = 9$, and C1's protocol skill and C3 carry the library's knowledge.

**Injection variants.** Retrieved skills are injected through the strategy hint of prompt P7 (Appendix A) or, when P7 is off, as a terse "name: description" list. Three optional variants act only on this injection step: (i) a *hub-corrected similarity* $\tilde{s}(q, v_i) = \langle f(q), f(v_i) \rangle - \mu_i$, where $\mu_i$ is the mean similarity of $v_i$ to the training questions, used for ranking and for the floor of Eq. (7) so that skills uniformly similar to every query no longer pass the gate; (ii) a *schema-constrained hint*: P7 must preserve the output schema and, in tool regimes, the tool budget; and (iii) an *answer-form line*, derived for each skill from the gold answers of its $5$ nearest training questions (option letter, short phrase, or free form), followed in tool regimes by the tool budget (see the per-dataset configuration table below).

**Selected configuration per dataset** (chosen on held-out validation questions, $32$ or $64$ of the $128$ training questions, with each library then retrained on all $128$; used for Table 2 and Figure 4; Protocol: active failure types of the protocol skill, M = `malformed_tool`, B = `over_budget`, T = `tool_without_tools`, A = `no_answer`, "data": answer form from training golds, "off": not injected; C3: CT concrete tool prompt, FT final-answer turn, AS answer suffix, LP lenient parser, QP question pre-search, PG$n$ located pages, TG time grounding, OC option check, V$k_{\text{sc}}$ self-consistency over $k_{\text{sc}}$ samples, V$k_{\text{sc}}$r refusals do not vote; Tok.: maximum new tokens at evaluation):

| Dataset | Run | Floor | Injection | Protocol | C3 | Tok. |
|---|---|---|---|---|---|---|
| AIME2024 | R3 | $\tau = 9$ | – | A | AS | 8192 |
| AIME2025 | R3 | $\tau = 9$ | – | off | AS, V5 | 8192 |
| WebWalkerQA | R1 | $\tau = 9$ | – | off | FT, LP, QP | 4096 |
| FVQA | R3 | $\tau = 9$ | – | B | CT, FT, LP | 1024 |
| InfoSeek | R1 | $\tilde{\tau} = -0.02$ | P7 + answer-form | M, B, A | – | 1024 |
| SimpleVQA | R3 | $\tau = 0.2$ | P7 + schema | M, B, A | V5r | 1024 |
| LiveVQA | R1 | $\tau = 9$ | – | M, A | – | 1024 |
| DocBench | R1 | $\tau = 9$ | – | M, B, A (data) | – | 1024 |
| MMLongBench-Doc | R3 | $\tau = 9$ | – | off | CT, FT, LP, PG4 | 1024 |
| OmniVideoBench | R2 | $\tau = 0.55$ | terse | A (data) | CT, FT, AS | 4096 |
| WorldSense | R1 | $\tau = 9$ | – | T, A | TG, OC | 4096 |

All datasets: $N_{\max} = 3$, $k_{\text{rrf}} = 60$, seed 1; 8 keyframes on OmniVideoBench and 32 on WorldSense.

**Budgets.** The token budgets above are part of C3 (baselines: §4.1). The maximum prompt length is $16384$ words on AIME, WebWalkerQA and MMLongBench-Doc, $32384$ on OmniVideoBench, and the regime default elsewhere; FVQA training rollouts use a limit of $24576$ words.

# F Future Work

HyperSkill improves a frozen backbone by editing an auditable skill library instead of its weights, which leads to four future directions. First, the library can be paired with lightweight parameter updates, e.g. a low-rank adapter that complements the in-context skills. Second, because skills are plain text with an advantage, the libraries of related datasets, such as the four VL-Web datasets that share the `text_search` tool, can be merged and re-weighted by the EMA of Eq. (4) from new rollouts. Third, C3 refers to tools only through their tags, so extending it to code execution or GUI actions mainly requires new tool tags, budgets and evidence texts, while C1 and C2 stay unchanged. Finally, batched or importance-sampled counterfactual probes can lower the admission cost of C1.

## NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims in the abstract and §1 match Table 2, Figure 4 and §4.2.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss limitations here: each skill library incurs a one-time API cost before serving; counterfactual probes add verification calls at training time; and all experiments use one frozen backbone, with lightweight parameter updates left to future work.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Propositions 1–3 are in §3; their assumptions and full proofs are in App. B.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Settings: §4.1, §3.3, App. E.7; pseudocode: App. C; prompts: App. A.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Code and data are public at the URL in the title-page footnote.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Data splits, tools and decoding are in §4.1, App. D.1 and App. E.7.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Paired bootstrap 95% intervals (10k resamples) are in App. D.2.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: API calls only, no GPU (one CPU host); §4.7 reports the API costs.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We use only licensed public datasets and collect no human-subject data.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Positive: auditable skill libraries can reduce compute and make agent behaviour editable. Risks: wrong or biased skills could propagate at serving time. Mitigation: we use only public datasets and APIs, and recommend reviewing libraries before deployment.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Only a pipeline, a public-data benchmark and skill libraries are released.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All assets are cited (§4.1, App. D) and used under their licences.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: HyperSkill-Bench and the pipeline are documented in App. D, C and A.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper involves no crowdsourcing and no research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: No human subjects are involved, so no IRB approval was required.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: The frozen LLM backbone is a core component of the method (App. A).

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.