# HASTE: HIERARCHICAL ACCUMULATION OF SKILLS FOR TRANSFER-EFFICIENT ML ENGINEERING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Agents evaluated on MLE-Bench solve every competition from scratch; whatever an agent learns while engineering one solution is discarded before the next task begins. We present HASTE, an ML-engineering agent that accumulates plain-text skills in a three-tier hierarchy of global, domain, and competition scope, promotes learnings upward between tasks through LLM-driven abstraction, and loads only the tiers relevant to the task at hand. Across three seeds on the 22 MLE-Bench Lite competitions, using Claude Sonnet 4.6 with a CLI backend under a 12-hour cap per attempt, HASTE reaches a medal rate of $62.1\% \pm 1.5$ (SEM); a replication on Claude Opus 4.8 reaches $66.7\% \pm 1.5$, with the same six competitions failing to medal under either model. A multi-pass campaign that keeps each competition's best graded attempt reaches 77.3%, in the leaderboard's top band on a mid-tier model at half the typical budget; because attempts are selected on test outcomes, this figure is an upper bound. The accumulation loop operates end to end: each seed grows a store of 121 to 143 skills from an empty start, the orchestrator promotes learnings across tiers, loaded skills are cited by name in refinement proposals, including 15 cross-competition citations in the logged runs, and revisiting competitions with a matured store recovers 7 of 12 first-attempt failures. A system-level ablation over 8 competitions and 3 seeds places the deployed configuration at $83.3\% \pm 4.2$ against $62.5\%$ with no skills and $50.0\% \pm 7.2$ with an unscoped frozen store; the arms differ in harness budget as well as loading policy, so we read the ablation as comparing configurations rather than isolating loading.

## 1 INTRODUCTION

Current ML-engineering agents solve every competition from scratch. MLE-Bench (Chan et al., 2025) evaluates agents on Kaggle competitions independently, and agents treat the competitions independently too, resetting all state between tasks. A feature-engineering trick that proved decisive in one tabular competition, or a model family that reliably performs best on small-image classification, must be rediscovered in the next, similar task. The cost of this redundancy is visible across the public leaderboard, where top entries rely on frontier models, 24-hour budgets, or both, in part because exploration repeats across tasks.

Recent agents do accumulate cross-task experience, but they almost universally store it in a single flat pool such as a vector store, an embedding-indexed library, or one growing prompt. A flat store leaves open the question that determines usefulness, namely which knowledge should enter the context for a given task. HASTE's answer is to make scope of applicability explicit. Every skill carries the scope at which it was judged to hold, global, domain, or competition, and a task loads exactly the tiers whose scope covers it. Scope is assigned once, when a learning is promoted, and loading is a directory read, which keeps both the store and the loading decision inspectable.

HASTE, short for Hierarchical Accumulation of Skills for Transfer-Efficient ML Engineering, organizes accumulated knowledge into three scope tiers. Global skills apply to any ML task, domain skills to tabular, NLP, or vision work, and competition skills to a single dataset. An orchestrator assigns competitions to domain specialists, and each specialist loads only the global tier, its own domain tier, and the current competition's tier. Between rounds, an LLM promotes learnings upward, abstracting a competition-specific observation into a domain rule or a domain rule into a global one. The store is plain text on a filesystem, so it can be read and diffed.



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

| System | Knowledge Form | Organization | Cross-Task? | Hierarchy? | Promotion? | MLE-Bench? |
|---|---|---|---|---|---|---|
| AIDE | None | — | ✗ | ✗ | ✗ | ✓ |
| MLE-STAR | Web retrieval | Flat (external) | ✗ | ✗ | ✗ | ✓ |
| R&D-Agent | RAG + traces | Flat (per-task) | ∼ | ✗ | ✗ | ✓ |
| MARS | Reflective memory | Flat (cross-branch) | ∼ | ✗ | ✗ | ✓ |
| Agent K v1.0 | Structured memory | Flat (shared) | ✓ | ✗ | ✗ | ✗ |
| MLCopilot | Past experiences | Flat (vector DB) | ✓ | ✗ | ✗ | ✗ |
| ExpeL | Text insights | Flat (vector store) | ✓ | ✗ | ✗ | ✗ |
| MLZero | Dual memory | Semantic + Episodic | ✓ | ✗ | ✗ | ∼ |
| Voyager | Code skills | Flat (embedding) | ✓ | ∼ | ✗ | ✗ |
| ADAS | Code archive | Flat (archive) | ✓ | ✗ | ✗ | ✗ |
| A-Mem | Linked notes | Graph (evolving) | ✓ | ✗ | ✗ | ✗ |
| G-Memory | Insights + traces | 3-tier graph | ✓ | ✓ | ∼ | ✗ |
| **HASTE (ours)** | **Text skills** | **3-tier hierarchy** | ✓ | ✓ | ✓ | ✓ |

Table 1: Comparison of cross-task knowledge mechanisms in ML engineering agents. ✓= present, ✗ = absent, ∼ = partial. G-Memory also organizes memory hierarchically; HASTE differs in that its tiers encode *scope of applicability* and directly gate what a task loads, and in being evaluated on MLE-Bench.

This paper makes three contributions. The first is a replicated benchmark result. Across three independent seeds on MLE-Bench Lite, HASTE medals at $62.1\% \pm 1.5$ (SEM) under a 12-hour cap per attempt on Claude Sonnet 4.6, a replication on Claude Opus 4.8 reaches $66.7\% \pm 1.5$ with the same failure set, and a multi-pass campaign's best attempts reach 77.3%, the top public band, on a mid-tier model at half the dominant budget. The second is evidence that the accumulate–promote–reload loop operates end to end in a deployed agent: stores grow from empty to 121–143 entries per seed, 24 promotions are recorded in the orchestrator logs, loaded skills are cited by name in refinement proposals with 15 cross-competition citations among the logged runs, and revisits with a matured store recover 7 of 12 first-attempt failures. The third is a system-level ablation with resource accounting: on 8 competitions and 3 seeds the deployed configuration medals in 83.3% of cells against 62.5% with no skills and 50.0% with an unscoped frozen store, and because the arms differ in harness budget as well as loading policy, we report it as a comparison of configurations.

## 2 Related Work

**MLE-Bench agents.** A first family of MLE-Bench agents advances the search axis. AIDE (Jiang et al., 2025) runs greedy tree search over solution code. MLE-STAR (Nam et al., 2025) performs ablation-guided targeted refinement with web-retrieved priors. AIRA-dojo (Toledo et al., 2025) formalizes MLE agents as search policies over operator sets, R&D-Agent (Yang et al., 2025) separates a researcher from a developer, and ML-Master (Liu et al., 2025) integrates exploration and reasoning. MARS (Chen et al., 2026) and LoongFlow (Wan et al., 2025) apply budget-aware and graph-aware evolutionary search, and the FM agent (Li et al., 2025) scales a similar recipe. HASTE is orthogonal to this axis. It contributes accumulated, scoped knowledge, and could in principle be combined with any of these search strategies.

**Cross-task knowledge in LLM agents.** Several systems accumulate experience across tasks but store it flat. Voyager (Wang et al., 2024) maintains an embedding-indexed library of code skills, and ExpeL (Zhao et al., 2024) extracts natural-language insights into a flat vector store. Reflexion (Shinn et al., 2023) introduced verbal self-reflection within a task, and ICAL (Sarch et al., 2024) extends reflection to vision-language agents. For data science specifically, Agent K (Grosnit et al., 2024) maintains shared structured memory across competitions, MLCopilot (Zhang et al., 2024) retrieves distilled knowledge by embedding, DS-Agent (Guo et al., 2024) applies case-based reasoning, MLZero (Fang et al., 2025) separates semantic from episodic memory, and ADAS (Hu et al., 2024) evolves agents over an archive of designs. SkillRL (Xia et al., 2026) reports a substantial performance drop when distilled skills are replaced with raw trajectories, which motivates heavy compression. Structured alternatives to flat pools exist outside ML engineering. A-Mem (Xu et al., 2025) maintains interlinked memory notes that evolve as new experience arrives, and G-Memory (Zhang et al., 2025) traces multi-agent experience through a three-level graph of insight, query, and

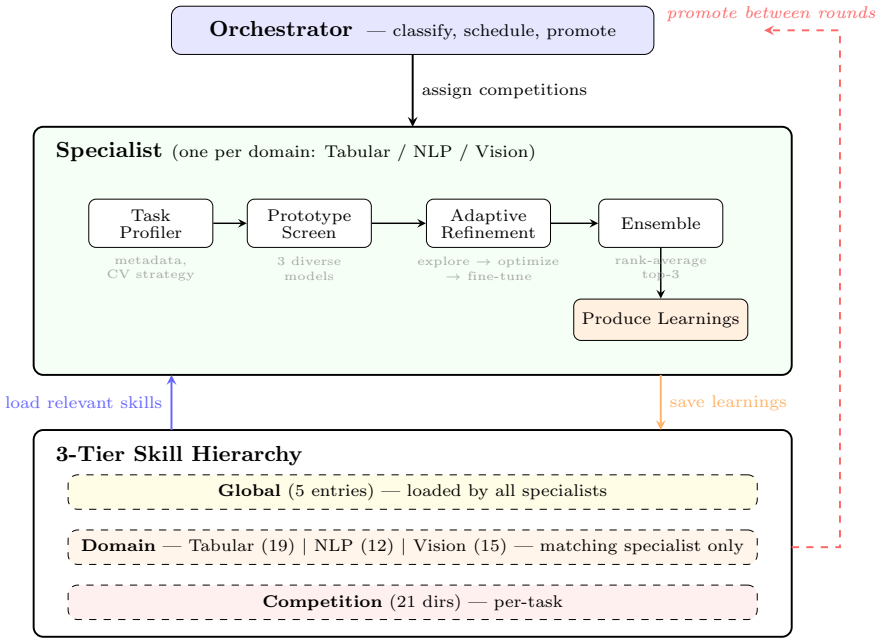


Figure 1: HASTE architecture. The Orchestrator assigns competitions to domain Specialists. Each Specialist loads relevant skills, executes the pipeline (profile → prototype → refine → ensemble), and produces learnings. Between rounds, the Orchestrator promotes generalizable learnings upward through the hierarchy via LLM-driven abstraction.

interaction memory, abstracting reusable insights upward much as our promotion step does. These are the closest precedents to HASTE; hierarchical organization of agent memory is not itself new.

Table 1 compares these mechanisms. HASTE's tiers encode *scope of applicability*, global, domain, or competition, and that scope deterministically gates which entries enter a task's context, rather than indexing them for retrieval. To our knowledge, HASTE is the first MLE-Bench agent to organize reusable knowledge this way and to evaluate that organization against no-skill and unscoped-store configurations. We do not claim that scoped tiers outperform graph-structured or retrieval-based memory; that comparison requires a matched baseline we do not run.

**Hierarchy in agent systems and AutoML.** Hierarchy is well motivated outside ML engineering. Options (Sutton et al., 1999) and feudal reinforcement learning (Dayan & Hinton, 1993; Vezhnevets et al., 2017) organize actions at multiple temporal resolutions, GITM (Zhu et al., 2023) uses a three-level task decomposition within Minecraft, CoALA (Sumers et al., 2024) taxonomizes agent memory by type, and Talebirad et al. (2026) argue that bounded-capacity agents benefit from multi-level knowledge organization. HASTE applies this principle to a transferable knowledge store rather than to actions or goals. Classical AutoML systems such as AutoGluon (Erickson et al., 2020), TPOT (Olson & Moore, 2016), Auto-WEKA (Thornton et al., 2013), and Auto-Sklearn (Feurer et al., 2015; 2022) amortize per-task cost by searching predefined configuration spaces, with meta-learning transferring numerical meta-features across datasets. LLM-era variants tune hyperparameters (Liu et al., 2024), generate features (Hollmann et al., 2023), search architectures (Chen et al., 2023), or orchestrate training (Zhang et al., 2023). HASTE pursues the same amortization goal, but its search space is unbounded Python code, made tractable by reloading qualitative lessons whose declared scope matches the current task.

## 3 HASTE

Figure 1 shows the two hierarchies that co-evolve during a multi-competition run. The skill hierarchy holds what has been learned, and the agent hierarchy determines which tiers each agent loads.

**Example: Global Skill File**

```
name: GLOBAL-dogs-vs-001  domain: global
type: technique  evidence_count: 1
source: dogs-vs-cats-redux  created: 2026-04-27
# Ensembling a strong model with a weaker model can degrade
performance
FAILURE: Combining ConvNeXt-Large (CV 0.0122) with EfficientNetV2-L
(CV 0.0211) via geometric mean produced 0.0211, matching the weaker
model rather than improving on the stronger one.  The geometric mean
emphasizes lower probabilities, so the weaker model's miscalibrated
predictions dominated.
Ensemble gains require components of similar strength that make
uncorrelated errors.  When one model is clearly dominant, ensembling
adds noise.  Before ensembling, verify that (a) individual scores are
within ~20% of each other and (b) error correlation is low.
```

Figure 2: A skill file from the global tier, shown in condensed form. Skills carry structured metadata and a plain-text body. This entry was learned from dogs-vs-cats-redux and promoted to the global tier; whether it changed later decisions is not established, and §7 shows that prompts embedded only the first 2,000 characters of the assembled block.

### 3.1 Three-Tier Skill Store

The store is a plain-text filesystem of markdown entries organized by scope of applicability. The global tier holds rules that apply to any ML task and contained 5 entries after one benchmark pass. The domain tier holds tabular, NLP, and vision knowledge, with 19, 12, and 15 entries respectively. The competition tier holds dataset-specific notes, 108 entries across 21 competition directories, and is loaded only when its competition is revisited. Formally, the store is $\mathcal{S} = \mathcal{S}^G \cup \mathcal{S}^D \cup \mathcal{S}^C$, and a specialist assigned task $t$ loads $\Lambda(t) = \mathcal{S}^G \cup \mathcal{S}^D_{d(t)} \cup \mathcal{S}^C_t$, where $d(t)$ is the task's domain. Figure 2 shows a representative global skill, and Table 7 in the appendix lists examples across tiers, each traceable to the experiment that produced it.

Within each tier, HASTE distinguishes three kinds of entries that are injected at different stages of the pipeline. Technique entries record what worked or failed and feed the proposal prompts during refinement. Commitment priors record design choices whose outcomes varied strongly across tasks, and they feed the prototype screen so that such choices are tested before the agent commits to one. Refinement hints record which adjustments tend to improve results for each model family and feed the optimizing and fine-tuning stages. Other systems keep all of these in a single flat pool. We separate them because they answer different questions and enter the loop at different points.

Loading is simple. A specialist reads the relevant directories and concatenates their entries into its context, subject to a character cap per prompt slot. No embedding index is used at the store's current scale, a decision consistent with recent results on the limits of single-vector retrieval under multi-field conditions (Weller et al., 2026). Skills carry their scope in their filesystem location, so selection is a directory read.

Most skills originate from an LLM reflection step at the end of each competition, as in Reflexion and ExpeL, with success and failure analyzed in pairs so that failure modes are recorded explicitly. Commitment priors and refinement hints are additionally mined from experiment logs by a prior extractor, which identifies high-variance design choices and high-acceptance refinement moves algorithmically rather than by summarization.

### 3.2 Agent Hierarchy and Execution Loop

An orchestrator has three responsibilities. It classifies each competition into a domain, it schedules rounds of competitions, and it promotes learnings between rounds. Competitions are executed by per-domain specialists for tabular, NLP, and vision tasks, each of which sees only the skill tiers matched to its scope.

Given a competition, a specialist runs a five-stage pipeline, formalized in Algorithms 1 and 2 in the appendix. Task profiling parses the metadata, selects a cross-validation strategy, and probes the available hardware without executing any model. The prototype screen then asks the LLM for several fundamentally diverse modeling approaches and executes each cheaply on a single validation fold. Score spreads between prototypes are frequently wide, which justifies screening several approaches before committing to one. Adaptive refinement iterates on the winning prototype, with a smaller runner-up branch kept as a fallback. The refinement loop passes through exploring, optimizing, and fine-tuning phases, escalates to the next phase after two consecutive non-improvements, and reverts any change that hurts the validation score before taking the next step. A six-mode failure taxonomy structures each proposal. The LLM first diagnoses the current state as underfitting, overfitting, feature gap, noise ceiling, distribution mismatch, or diminishing returns, and then proposes one atomic change. An ensemble stage rank-averages top checkpoints from both branches and keeps the ensemble only if it beats the best single member. Finally, the specialist reflects on the full experiment history and emits two to five plain-text learnings, each with a proposed tier.

The executor is a pluggable backend with file-editing and shell tools. It validates every candidate submission against the competition's sample submission for column names, row count, and value types before scoring. All experiments in this paper use Claude Sonnet 4.6 with a CLI backend, except the Claude Opus 4.8 replication reported in §8.

**Promotion.** Between rounds, the orchestrator evaluates each new learning with an LLM and decides among four outcomes: leave it at the competition tier, abstract and promote it to its domain, promote it to global, or mark it as conflicting with an existing skill, in which case both are kept and annotated with the conditions under which each held. Promotion turns a dataset-specific observation into a reusable rule, and, apart from the refinement hints that the prior extractor writes to the domain tier under the same scope rule, it is the only mechanism by which knowledge crosses competitions. In the multi-pass campaign, specialists produced 104 learnings and the orchestrator evaluated 83 of them; the retained logs record 24 promotion events, 22 to a domain tier and 2 to global (23 distinct skills, one promoted twice), plus one conflict kept with both entries annotated. These counts are lower bounds, because four hard-killed competitions had their learnings salvaged out of band and not every phase retained a parseable log; the supplementary archive reproduces the recount from the verbatim log lines. Promotion events also do not account for every upper-tier entry: 13 of the 46 domain entries are refinement hints written by the prior extractor rather than by promotion.

## 4 Experimental Setup

**Benchmark.** We evaluate on MLE-Bench Lite (Chan et al., 2025), the 22-competition subset used for cross-agent comparison. Submissions are scored on held-out test sets with the official grader, and medal thresholds are set per competition. The budget is 12 hours per competition, half the dominant 24-hour budget on the public leaderboard. Medal rate is computed over all 22 competitions.

**Replication protocol.** We run three independent seeds. Each seed starts from an empty skill store and works through the full roster, accumulating and promoting skills as it goes. Seeds share nothing and index independent replicate runs; LLM sampling is not seeded. We report the mean and standard error over seeds.

**Multi-pass campaign.** Separately, we report a campaign that made several passes over the roster with a single accumulating store. Every pass after the first starts warm, since it reloads the global and domain skills learned from the competitions solved so far, and we take the best attempt per competition. This is the accumulation setting the system is designed for, so the resulting figure is a single-campaign upper bound that complements the seed mean; Appendix A compares first attempts with campaign best and states why that does not isolate the warm starts.

**Loading ablation.** On 8 competitions spanning NLP, vision, and tabular tasks, one of them a bioacoustic competition that MLE-Bench presents as pre-extracted tabular features, we compare three configurations with three seeds each: tiered, the deployed system with scoped loading over its accumulating store; flat, the frozen 159-skill store loaded without regard to scope; and empty, no skills. §7 states what the arms hold fixed.

**Transfer evaluation.** Within the multi-pass campaign, we compare each competition's first attempt against its campaign-best attempt (§6).

| Model / seed | Medals | G/S/B | >med | Valid | Rate |
|---|---|---|---|---|---|
| Sonnet 4.6, seed 0 | 14 | 8/2/4 | 19 | 22 | 63.6% |
| Sonnet 4.6, seed 1 | 14 | 8/3/3 | 16 | 21 | 63.6% |
| Sonnet 4.6, seed 2 | 13 | 9/1/3 | 19 | 21 | 59.1% |
| **Sonnet 4.6, mean ± SEM** | | | | | **62.1% ± 1.5** |
| Opus 4.8, seed 0 | 15 | 9/3/3 | 18 | 22 | 68.2% |
| Opus 4.8, seed 1 | 15 | 8/4/3 | 19 | 21 | 68.2% |
| Opus 4.8, seed 2 | 14 | 8/2/4 | 18 | 22 | 63.6% |
| **Opus 4.8, mean ± SEM** | | | | | **66.7% ± 1.5** |

Table 2: Three-seed replications on MLE-Bench Lite at 12h per attempt, on Claude Sonnet 4.6 and Claude Opus 4.8. Medals, above-median counts (>med), and valid submissions are out of 22. The Opus campaign ran with harness hardening the Sonnet seeds lacked (§8), so the two means are not a controlled model comparison.

| # | Agent | LLM | Lite (%) | Budget |
|---|---|---|---|---|
| 1 | Famou-Agent 2.0 | Gemini-3-Pro | 80.30±1.52 | 24h |
| 2 | MLEvolve | Gemini-3-Pro | 80.30±1.52 | 12h |
| 3 | PiEvolve | Gemini-3-Pro | 80.30±1.52 | 24h |
| 4 | CAIR MARS+ | Gemini-3-Pro | 78.79±1.52 | 24h |
| – | **HASTE (ours, campaign best)** | **Sonnet 4.6** | **77.3**† | **12h** |
| 6 | AIBuildAI | Claude-Opus-4.6 | 77.27±0.00 | 24h |
| 7 | Famou-Agent 2.0 | Gemini-2.5-Pro | 75.76±1.52 | 24h |
| 8 | ML-Master 2.0 | DeepSeek-V3.2 | 75.76±1.51 | 24h |
| 9 | Leeroo | Gemini-3-Pro | 68.18±2.62 | 24h |
| 10 | MLE-STAR-Pro | Gemini-2.5-Pro | 68.18±2.62 | 24h |
| 11 | R&D-Agent | gpt-5 | 68.18±2.62 | 12h |
| – | **HASTE (ours, 3-seed mean)** | **Opus 4.8** | **66.67±1.52** | **12h** |
| 12 | Thesis | gpt-5-codex | 65.15±1.52 | 24h |
| 13 | Operand ens. | gpt-5-low | 63.64±0.00 | 24h |
| – | **HASTE (ours, 3-seed mean)** | **Sonnet 4.6** | **62.12±1.52** | **12h** |
| 14 | InternAgent | deepseek-r1 | 62.12±3.03 | 12h |
| 15 | AIRA-dojo | o3 | 55.00±1.47 | 24h |
| 16 | ML-Master | deepseek-r1 | 48.48±1.52 | 12h |
| 17 | AIDE | o1-preview | 35.91±1.86 | 24h |

Table 3: MLE-Bench Lite leaderboard comparison (public leaderboard, frozen 2026-04-24; lower-scoring duplicate entries of the same agent family and one further sub-50% entry omitted). Our rows are unranked because they are not directly comparable: †the campaign-best figure selects each competition's best graded attempt across a multi-pass campaign (250 cumulative hours), whereas other rows report per-attempt caps; the 3-seed means are single-pass, selection-free estimates and are the figures we consider comparable.

## 5 Main Results

Table 2 reports the replication. HASTE medals at $\mathbf{62.1\% \pm 1.5}$ (SEM) over three seeds, which corresponds to 14, 14, and 13 medals out of 22 with 8, 8, and 9 golds respectively. The gold-medal set is stable across seeds. Seven competitions earn gold at every seed, and eleven earn a medal at every seed. Above-median counts range from 16 to 19 of 22.

A separate multi-pass campaign that keeps, for each competition, the best of its several graded attempts reaches **77.3%**, or 17 of 22 competitions, with 10 gold, 3 silver, and 4 bronze medals, and 86.4% of competitions above the leaderboard median. Because attempts are selected on test outcomes, this is an upper bound rather than the expected medal rate of a single run, and it consumes more total compute than the per-attempt caps other entries report; Table 3 places both figures on the public MLE-Bench Lite board with that caveat attached. At the benchmark's resolution 17 of 22 ties the adjacent entry, so we place the campaign in the top band rather than at a specific rank. Unlike the other entries in that band, it uses a mid-tier model at half the typical hour budget. The three-seed

mean is in the tier of Gemini-2.5-Pro agents, above AIRA-dojo and ML-Master. §6 reports how the store grew during that campaign.

## 6 Accumulation in the Deployed Loop

The replication seeds and the multi-pass campaign exercise the full loop, so they show whether skills are produced, promoted, loaded, and used, independently of any ablation.

**The store grows and learnings are promoted.** Each replication seed starts empty and ends with 121, 129, or 143 skills (1–2 global, 27–36 domain, 92–106 competition-tier); the campaign store holds 159. In the campaign the orchestrator evaluated 83 learnings and promoted 24 (§3), and Table 8 records the store each competition's best attempt ran with, from 2 global and 6 domain entries at the earliest attempt to 5 and 46 at the latest.

**Loaded skills are cited across competitions.** The refinement prompt asks the specialist to name any past skill that influenced its proposal. Across the 35 seed runs with retained refinement logs, 317 proposals were logged; 24 (7.6%) cite a loaded skill by name, 27 citations in all, and 15 of those cite a skill produced by a *different* competition. Dog-breed and dogs-vs-cats both begin by citing a vision skill learned on aerial-cactus, random-acts-of-pizza cites a global skill learned on mlsp-2013-birds, and mlsp-2013-birds cites a tabular skill learned on leaf-classification. Proposals that cite a skill were kept at 62% (15 of 24) against 52% (151 of 293) for the rest. These are self-reported citations from a small sample and a lower bound on use; each prompt exposes at most 2,000 characters of the assembled block, so only a few skills are visible per proposal.

| Metric | First attempt | Campaign best |
|---|---|---|
| Medal rate | 45.5% (10/22) | **77.3%** (17/22) |
| Composition (G/S/B) | 6/3/1 | 10/3/4 |
| Above median | 15/22 (68.2%) | 19/22 (86.4%) |
| Improvement | +7 medals (+70%) | |

Table 4: Campaign first attempt versus campaign best, per competition, within the multi-pass campaign. Later attempts differ in accumulated skills, attempt count, compute, and test-outcome selection at once, so the gain is descriptive and is not a transfer estimate (Appendix A).

**Revisits with a matured store recover failures.** Within the campaign, first attempts medal on 10 of 22 competitions and campaign-best attempts on 17 (Table 4). Seven of the 12 first-attempt failures flipped to medals on revisits made after the store had matured, four of them to gold, and the five that medaled on neither attempt are five of the six competitions that never medal in the replication seeds either. Revisits were triggered by failure and add attempts and compute, so seven medals is an upper bound on the store's contribution rather than a transfer estimate; Appendix A gives the per-competition detail.

## 7 System-Level Ablation

We compare the three configurations of §4 on 8 competitions and 3 seeds.

**Results.** Table 5 gives the ordering over 24 cells per arm: tiered $83.3\% \pm 4.2$, empty $62.5\% \pm 0.0$, and flat $50.0\% \pm 7.2$. An exact paired sign-flip permutation test on per-competition medal fractions puts tiered above flat at $p = 0.031$; tiered above empty ($p = 0.19$) and empty above flat ($p = 0.25$) do not reach significance, and with 8 competitions the smallest attainable $p$ is $1/256$. Figure 3 shows which competitions account for the margin. Tiered is the only arm to medal mlsp-2013-birds (3 of 3 seeds) and random-acts-of-pizza (2 of 3), and it adds aptos2019; on the easier five the arms tie or nearly so. Seeds index replicate runs rather than sampler seeds, so empty's $\pm 0.0$ records identical outcomes. One flat cell is a mechanical artifact, a constant-class submission (Appendix F); flat is $47.6\%$ without it.

**Resources.** Table 6 shows resource use: the flat arm consumed 3.78M output tokens against empty's 1.86M and tiered's 2.27M, and 756K tokens per medal against tiered's 284K, a $2.7\times$ gap.

| Condition | Medals | Per seed | Rate ± SEM | G/S/B |
|---|---|---|---|---|
| **Tiered** | **20/24** | **7, 7, 6** | **83.3% ± 4.2** | **12/3/5** |
| Flat (all 159 skills) | 12/24 | 5, 4, 3 | 50.0% ± 7.2 | 10/2/0 |
| Empty (no skills) | 15/24 | 5, 5, 5 | 62.5% ± 0.0 | 14/0/1 |

Table 5: System-level ablation (8 competitions × 3 seeds). Tiered is the deployed configuration: scoped loading over an accumulating store under the production harness. Flat loads the frozen 159-skill store without regard to scope and empty loads no skills, both under the ablation harness. §7 states what the arms hold fixed. Medal rate is over all 24 cells.

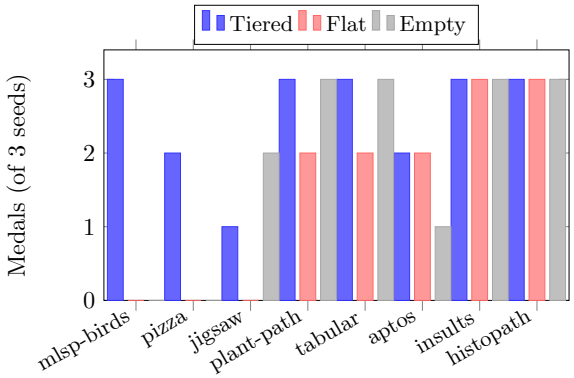


Figure 3: Medals won per competition (of 3 seeds) under each ablation arm, ordered by between-arm spread. The tiered margin is concentrated on mlsp-2013-birds, random-acts-of-pizza, and aptos2019, the tasks where search is most expensive. Because the arms differ in harness as well as loading (§7), the figure shows where the gap is, not what causes it.

**What the arms hold fixed.** The three arms share the roster, the grader, the model, and the pipeline, and they differ in loading policy as designed. They also differ in two respects the design did not intend. First, the tiered cells are the replication seeds under the production harness (12 hours, 20-iteration cap, learning production on, per-seed accumulating stores), whereas flat and empty ran an ablation harness (10.5 hours, 11-iteration ceiling, learning production off). Second, each prompt exposes at most the first 2,000 characters of the assembled skill block, and the block is assembled global tier first; the frozen store's five global entries fill that window on their own, so the flat arm's prompts exposed the head of the global tier rather than the unscoped store as a whole. The comparison therefore shows that the deployed configuration outperforms both ablated configurations on these tasks, with the gains concentrated where search is expensive; it does not isolate loading policy from budget or from what each prompt exposed. The harness-matched, cap-lifted rerun that would isolate it is the next experiment (§8).

## 8 Discussion and Limitations

**Isolating loading policy.** The follow-up we value most is the ablation rerun on one harness with the frozen store and a lifted per-prompt cap, together with an ExpeL-style top-$k$ retrieval arm and a matched-cardinality arm, so that scope can be separated from budget, entry count, and semantic relevance. We will also log the exact skill content of each prompt as a run artifact, so that what each memory arm exposed is recorded.

**Fixed domain taxonomy.** Routing assumes competitions fall into manually defined domains. We do not evaluate mixed-domain or unseen-domain tasks, taxonomy misspecification, or the cost of routing errors, all of which bear directly on whether scope-based loading generalizes beyond this roster.

**Model generality.** A three-seed replication with Claude Opus 4.8, holding the condition, iteration cap, per-competition budget, roster, and cold-start stores fixed, reaches $66.7\% \pm 1.5$ (15, 15, and 14

| Metric | Tiered | Flat | Empty |
|---|---|---|---|
| Medal rate | **83.3%** | 50.0% | 62.5% |
| Mean score | **0.923** | 0.875 | 0.900 |
| Output tokens | 2.27M | 3.78M | **1.86M** |
| Tokens / comp. | 284K | 472K | **232K** |
| Tokens / medal | **284K** | 756K | 371K |
| Skills loaded | tier-scoped | 159 | 0 |

Table 6: Resource comparison across ablation arms. Medal rate and mean score are from the multi-seed ablation (24 cells per arm); mean score is an unnormalized mean of raw per-cell test scores, comparable across arms on this fixed roster only. Output tokens are from single instrumented runs: flat and empty from the ablation harness, tiered from the May campaign's runs of the same competitions. Tokens per medal uses those runs' medal counts (8/5/5).

medals of 22). The six competitions that never medal across seeds are identical under both models, so the model swap moves the mean without changing which tasks the system fails on. Two caveats apply to the comparison. The Opus campaign ran with rate-limit hardening that the June Sonnet seeds lacked, namely staggered seed starts, hold-and-poll retries in place of fast failure, and explicit context caps, and with an unpinned reasoning-effort setting that resolved higher than the Sonnet runs' pinned value; both differences would be expected to favor Opus on harness grounds alone, so we do not read the 4.6-point gap as a model effect. Both are Claude models, so we have not tested a second model family.

**Scale.** The store holds 159 entries after one benchmark pass, and directory reads suffice at this scale. Whether plain-text scoping remains sufficient at thousands of skills, or requires retrieval within tiers, is an open question. We also do not audit the store for redundancy, stale advice, or erroneous promotions, which is the natural failure mode as it grows. The linear refinement loop is likewise simpler than the tree, evolutionary, and Monte Carlo searches used by other systems, and the loaded priors plausibly reduce the branching factor enough for linear search to compete. A controlled comparison at a fixed knowledge condition remains open.

## 9 Conclusion

HASTE turns ML-engineering experience into a three-tier plain-text skill store and loads only what is in scope for the task at hand. The result is a replicated $62.1\% \pm 1.5$ medal rate on MLE-Bench Lite under a 12-hour cap on a mid-tier model, $66.7\% \pm 1.5$ on a second model with the same failure set, and a campaign upper bound of 77.3% in the leaderboard's top band. The accumulation loop operates in the reported runs. Stores grow from empty to over a hundred entries per seed, learnings are promoted across tiers, skills are cited across competitions, and revisits with a matured store recover most first-attempt failures. On a system-level ablation the deployed configuration outperforms both a skill-free and an unscoped-store configuration; isolating the loading policy itself, on a single harness with a lifted per-prompt cap, is the next experiment.

### Reproducibility statement

The system is specified in §3, with the orchestrator and specialist procedures given as pseudocode in Appendix B and the four prompts that drive promotion and refinement reproduced in Appendix C. The experimental protocol, including seeds, the per-competition wall-clock budget, and the definition of each ablation condition, is in §4. The supplementary archive contains the complete source, the frozen 159-skill inventory used in the ablation, and the grading report and submission file behind every number we report, together with a script that recomputes the headline figures from those artifacts offline. The skill-citation statistics of §6 and the prompt-content check of §7 are each recomputed by a script in the archive from the run logs and loader.

### ETHICS STATEMENT

This work evaluates an automated ML-engineering agent on MLE-Bench Lite, a benchmark built from public Kaggle competitions; no human subjects, private data, or personally identifying information are involved. The agent writes and executes code in an isolated per-experiment workspace on a dedicated compute node, against fixed benchmark datasets. Agents that accumulate and reuse engineering knowledge lower the cost of automating ML development, with the attendant labor-displacement and misuse-of-capability concerns that apply to capable coding agents generally; our supplementary artifacts are text skills and grading records rather than a deployable autonomous system. Our experiments consume substantial compute, which we report in Table 6 partly so that the energy cost of this class of experiment is visible.

### AI USE STATEMENT

The system under study is itself an LLM-based agent; all such use is the object of the research and is described in the body of the paper. Separately, in preparing this manuscript the authors used an LLM-based assistant for copy-editing, LaTeX formatting, and checking the text against the code and result artifacts. All research contributions, including the system design, the experimental protocol, the analyses, and the claims made from them, are the authors' own, and the authors verified every number reported here against the grading artifacts and take full responsibility for the content.

## REFERENCES

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, et al. MLE-bench: Evaluating machine learning agents on machine learning engineering. In *ICLR*, 2025. URL https://arxiv.org/abs/2410.07095.

Angelica Chen, David M. Dohan, and David R. So. EvoPrompting: Language models for code-level neural architecture search. In *NeurIPS*, 2023. URL https://arxiv.org/abs/2302.14838.

Jiefeng Chen, Bhavana Dalvi Mishra, Jaehyun Nam, Rui Meng, Tomas Pfister, and Jinsung Yoon. MARS: Modular agent with reflective search for automated AI research, 2026. URL https://arxiv.org/abs/2602.02660.

Peter Dayan and Geoffrey E. Hinton. Feudal reinforcement learning. In *NeurIPS*, 1993. URL https://proceedings.neurips.cc/paper/1992/hash/d14220ee66aeec73c49038385428ec4c-Abstract.html.

Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. AutoGluon-tabular: Robust and accurate AutoML for structured data, 2020. URL https://arxiv.org/abs/2003.06505.

Haoyang Fang, Boran Han, Nick Erickson, et al. MLZero: A multi-agent system for end-to-end machine learning automation, 2025. URL https://arxiv.org/abs/2505.13941.

Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In *NeurIPS*, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/11d0e6287202fced83f79975ec59a3a6-Abstract.html.

Matthias Feurer, Katharina Eggensperger, Stefan Falkner, Marius Lindauer, and Frank Hutter. Auto-Sklearn 2.0: Hands-free AutoML via meta-learning. *JMLR*, 23:1–61, 2022. URL https://www.jmlr.org/papers/v23/21-0992.html.

Antoine Grosnit, Alexandre Maraval, James Doran, et al. Kolb-based experiential learning for generalist agents with human-level kaggle data science performance, 2024. URL https://arxiv.org/abs/2411.03562.

Siyuan Guo, Cheng Deng, Ying Wen, Hechang Chen, Yi Chang, and Jun Wang. Ds-agent: Automated data science by empowering large language models with case-based reasoning. In *ICML*, 2024. URL https://arxiv.org/abs/2402.17453.

Noah Hollmann, Samuel Müller, and Frank Hutter. Large language models for automated data science: Introducing CAAFE for context-aware automated feature engineering. In *NeurIPS*, 2023. URL https://arxiv.org/abs/2305.03403.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *ICLR*, 2024. URL https://arxiv.org/abs/2408.08435.

Zhengyao Jiang, Dominik Schmidt, Dhruv Srikanth, Dixing Xu, Ian Kaplan, Deniss Jacenko, and Yuxiang Wu. AIDE: AI-driven exploration in the space of code, 2025. URL https://arxiv.org/abs/2502.13138.

Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pp. 85–103. Springer, 1972.

Annan Li, Chufan Wu, Zengle Ge, et al. The FM agent, 2025. URL https://arxiv.org/abs/2510.26144.

Siyi Liu, Chen Gao, and Yong Li. Large language model agent for hyper-parameter optimization. In *CPAL*, 2024. URL https://arxiv.org/abs/2402.01881.

Zexi Liu, Yuzhu Cai, Xinyu Zhu, et al. ML-master: Towards AI-for-AI via integration of exploration and reasoning, 2025. URL https://arxiv.org/abs/2506.16499.

Jaehyun Nam, Jinsung Yoon, Jiefeng Chen, Jinwoo Shin, Sercan O. Arik, and Tomas Pfister. MLE-STAR: Machine learning engineering agent via search and targeted refinement, 2025. URL https://arxiv.org/abs/2506.15692.

George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. An analysis of approximations for maximizing submodular set functions—I. *Mathematical Programming*, 14:265–294, 1978.

Randal S. Olson and Jason H. Moore. TPOT: A tree-based pipeline optimization tool for automating machine learning. In *Workshop on Automatic Machine Learning at ICML*, pp. 66–74, 2016. URL https://proceedings.mlr.press/v64/olson_tpot_2016.html.

Gabriel Sarch, Lawrence Jang, Michael J. Tarr, William W. Cohen, Kenneth Marino, and Katerina Fragkiadaki. VLM agents generate their own memories: Distilling experience into embodied programs of thought. In *NeurIPS*, 2024. URL https://arxiv.org/abs/2406.14596.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *NeurIPS*, 2023. URL https://arxiv.org/abs/2303.11366.

Theodore R. Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L. Griffiths. Cognitive architectures for language agents. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=1i6ZCvflQJ.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112:181–211, 1999. URL https://doi.org/10.1016/S0004-3702(99)00052-1.

Yashar Talebirad, Ali Parsaee, Csongor Y. Szepesvari, Amirhossein Nadiri, and Osmar Zaiane. Toward a theory of hierarchical memory for language agents. In *ICLR 2026 Workshop on Memory for LLM-Based Agentic Systems (MemAgents)*, 2026. URL https://arxiv.org/abs/2603.21564.

Chris Thornton, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In *KDD*, pp. 847–855, 2013. URL https://doi.org/10.1145/2487575.2487629.

Edan Toledo, Karen Hambardzumyan, Martin Josifoski, et al. AI research agents for machine learning: Search, exploration, and generalization in MLE-bench, 2025. URL https://arxiv.org/abs/2507.02554.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *ICML*, 2017. URL https://arxiv.org/abs/1703.01161.

Chunhui Wan, Xunan Dai, Zhuo Wang, Minglei Li, Yanpeng Wang, Yinan Mao, Yu Lan, and Zhiwen Xiao. Loongflow: Directed evolutionary search via a cognitive plan-execute-summarize paradigm, 2025. URL https://arxiv.org/abs/2512.24077.

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. *TMLR*, 2024. URL https://openreview.net/forum?id=ehfRiF0R3a.

Orion Weller, Michael Boratko, Iftekhar Naim, and Jinhyuk Lee. On the theoretical limitations of embedding-based retrieval. In *ICLR*, 2026. URL https://arxiv.org/abs/2508.21038.

Peng Xia, Jianwen Chen, Hanyang Wang, et al. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning, 2026. URL https://arxiv.org/abs/2602.08234.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-Mem: Agentic memory for LLM agents. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Xu Yang, Xiao Yang, Shikai Fang, et al. R&D-Agent: An LLM-agent framework towards autonomous data science, 2025. URL https://arxiv.org/abs/2505.14738.

Guibin Zhang, Muxin Fu, Kun Wang, Fanqi Wan, Ming Yu, and Shuicheng Yan. G-Memory: Tracing hierarchical memory for multi-agent systems. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Lei Zhang, Yuge Zhang, Kan Ren, Dongsheng Li, and Yuqing Yang. MLCopilot: Unleashing the power of large language models in solving machine learning tasks. In *EACL*, 2024. URL https://arxiv.org/abs/2304.14979.

Shujian Zhang, Chengyue Gong, Lemeng Wu, Xingchao Liu, and Mingyuan Zhou. AutoML-GPT: Automatic machine learning with GPT, 2023. URL https://arxiv.org/abs/2305.02499.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *AAAI*, 2024. URL https://doi.org/10.1609/aaai.v38i17.29936.

Xizhou Zhu, Yuntao Chen, Hao Tian, et al. Ghost in the minecraft: Generally capable agents for open-world environments via large language models with text-based knowledge and memory, 2023. URL https://arxiv.org/abs/2305.17144.

## A Campaign First Attempts versus Campaign Best: Detail

Table 4 in §6 compares each competition's campaign first attempt with its campaign best, where campaign best is the attempt selected by medal and then by test score. The two endpoints are consistent by construction: the 10 competitions that medaled on their first attempt retain a medal, and 7 that failed flipped to a medal, giving $10+7=17$. The flips are aerial-cactus, aptos2019, mlsp-2013-birds, and nomad2018 to gold, and jigsaw-toxic, siim-isic, and text-normalization-russian to bronze. Because campaign best includes the first attempt, no competition can lose a medal relative to its own first attempt; the single regression is a within-medal tier change on text-normalization-english. Five competitions medaled at neither endpoint: dog-breed, leaf-classification, NYC-taxi-fare, ranzcr-catheter, and tabular-playground-may. Table 9 lists every re-attempted competition with the store size at each attempt. Later attempts differ from first attempts in accumulated skills, attempt count, compute, and test-outcome selection at once, so the gain is descriptive. The 45.5% first-attempt figure is not comparable to the 62.1% single-pass mean of Table 2: the latter averages three complete independent campaigns, the former is the first attempt of each competition within one multi-pass campaign.

## B ALGORITHMS

Algorithm 1 gives the top-level multi-competition loop, and Algorithm 2 the per-competition specialist pipeline. The refinement budgets $B_w$ for the prototype winner and $B_r$ for the runner-up are configuration parameters, and each attempt runs under the 12-hour wall-clock budget, which in practice is the binding constraint. ADAPTIVEREFINE is a linear loop over its budget. Each step diagnoses the current state with the six-mode failure taxonomy, proposes one atomic change, escalates from exploring to optimizing to fine-tuning after two consecutive non-improvements, exits on stagnation in the final phase, and reverts any change that hurts the validation score before the next step.

Algorithm 1: HASTE orchestrator: multi-competition skill accumulation with LLM-driven promotion.

**Require:** competitions $\mathcal{T}$, domain partition $\mathcal{D}$, initial store $\mathcal{S}$
tag each $t \in \mathcal{T}$ with domain $d(t)$; schedule rounds $R_1, R_2, \ldots$
**for** round $R_r$ in $R_1, R_2, \ldots$ **do**
  $\mathcal{L}_r \leftarrow \emptyset$
  **for** competition $t \in R_r$ **do**
    $\Lambda(t) \leftarrow \mathcal{S}^G \cup \mathcal{S}^D_{d(t)} \cup \mathcal{S}^C_t$
    $(\hat{y}_t, \ell_t) \leftarrow$ SPECIALIST$(t, \Lambda(t))$
    $\mathcal{S}^C_t \leftarrow \mathcal{S}^C_t \cup \ell_t$;  $\mathcal{L}_r \leftarrow \mathcal{L}_r \cup \ell_t$
  **end for**
  **for** learning $\ell \in \mathcal{L}_r$ **do**
    $decision \leftarrow$ PROMOTELLM$(\ell, \mathcal{S})$
    **if** $decision =$ GLOBAL **then**
      $\mathcal{S}^G \leftarrow \mathcal{S}^G \cup \{$ABSTRACT$(\ell)\}$
    **else if** $decision =$ DOMAIN **then**
      $\mathcal{S}^D_{d(t)} \leftarrow \mathcal{S}^D_{d(t)} \cup \{$ABSTRACT$(\ell)\}$
    **else if** $decision =$ CONFLICT **then**
      keep both entries, annotated with their conditions
    **else**
      leave $\ell$ at the competition tier
    **end if**
  **end for**
**end for**
**return** $\{\hat{y}_t\}_{t \in \mathcal{T}}, \mathcal{S}$

Algorithm 2: HASTE specialist: per-competition pipeline.

**Require:** competition $t$, loaded skills $\Lambda(t)$, budgets $B_w, B_r$
$profile \leftarrow$ TASKPROFILER$(t)$
$m_1, m_2, m_3 \leftarrow$ PROTOTYPESCREEN$(t, profile, \Lambda(t))$
run each $m_i$;  $win, ru \leftarrow$ top-2 prototypes by validation score
$win^\star \leftarrow$ ADAPTIVEREFINE$(win, \Lambda(t), B_w)$
$ru^\star \leftarrow$ ADAPTIVEREFINE$(ru, \Lambda(t), B_r)$
$ens \leftarrow$ RANKAVERAGE(top-3 checkpoints)
$\hat{y} \leftarrow \arg\max_{c \in \{ens, win^\star, ru^\star\}}$ score$(c)$
$\ell \leftarrow$ PRODUCELEARNINGS$(history)$
**return** $\hat{y}, \ell$

## C PROMPT TEMPLATES

Four prompts drive the specialist and orchestrator loops, and each is reproduced here in abbreviated form. Figure 4 is the prototype screen that begins every competition with three diverse model families. Figure 5 is the refinement proposal issued at each step of ADAPTIVEREFINE (Algorithm 2),

which asks for a failure-mode diagnosis and a single atomic change. Figure 6 is the learning production prompt issued at the end of each competition, and Figure 7 is the promotion prompt the orchestrator uses to move learnings up the hierarchy (Algorithm 1). Placeholders in braces are filled at run time, and italic bracketed lines stand for injected context such as the resource budget, the experiment history, and the loaded skills, which are omitted for space.

**Prototype Screen Prompt (abbreviated)**

```
You are an ML engineer starting a Kaggle competition.
Competition: {competition_id}
Domain: {domain}  Metric: {metric} ({direction})
Data: {rows} rows, {features} features
Target: {target_type}
```
*[Resource budget: GPU model, VRAM, CPU cores, RAM]*
*[Accumulated skills from past competitions, if any]*
```
Phase 0 is a diversity screen across model families --- the winner
will be refined later.
Propose 3 DIVERSE model approaches to try.  Each should be a
fundamentally different strategy (e.g. gradient boosting vs. neural
net vs. linear model, or different preprocessing philosophies).
Return JSON:
{"models": [{"name": ..., "description": ...,
"change_specification": ...}, ...]}
```

Figure 4: Prototype screen prompt (abbreviated), issued once at the start of each competition. The full prompt injects measured GPU, CPU, and RAM from a resource probe and up to 2,000 characters of accumulated skills. The LLM returns three model specifications, each executed with 1-fold validation on the full data.

**Refinement Proposal Prompt (abbreviated)**

```
You are an ML engineer working on a Kaggle competition.
Competition: {competition_id}
Metric: {metric} ({direction})
Data: {rows} rows, {features} features
Current best score: {best_score}
Phase: {tier_label}
{tier_guidance}
```
*[Resource budget, experiment history, accumulated skills,*
*current code (first 3000 chars)]*
```
## Failure Mode Diagnosis
First, diagnose which failure mode best describes
the current state:
1.  UNDERFITTING --- train score near baseline, small gap
2.  OVERFITTING --- large train-val gap, high train score
3.  FEATURE_GAP --- top features dominate, score plateaus
4.  NOISE_CEILING --- high CV variance, score fluctuates
5.  DISTRIBUTION_MISMATCH --- train-val-test disagreement
6.  DIMINISHING_RETURNS --- each iter improves <0.1%
Mention which mode applies and which past skill (if any) influenced
your decision.
Propose ONE atomic change to improve the score.
Return JSON: {"plan": ..., "change_specification": ...,
"decision": "CONTINUE" | "NEXT_TIER" | "STOP"}
```

Figure 5: Refinement proposal prompt (abbreviated), issued at every refinement step. The phase label advances through Exploring, Optimizing, and Fine-tuning; the six-mode failure taxonomy structures the diagnosis, and the `decision` field lets the LLM escalate the phase or stop early.

**Learning Production Prompt (abbreviated)**

```
You just finished working on "{competition_id}".
Domain: {domain} Metric: {metric} ({direction})
Dataset: {rows} rows, {features} features
## Your Experiments and Results
```
*[Full experiment history with scores, deltas, kept/reverted status]*
```
Write learnings that could help in future competitions.
Distinguish SUCCESS and FAILURE learnings:
- What worked --- "this technique improved the score because..."
- What failed --- "this technique hurt the score because..."
Failure learnings are equally valuable. Knowing what NOT to do saves
future compute.
For failure learnings, note:
- WHY it failed (overfitting? wrong model family?)
- WHEN it might fail again (conditions to watch for)
Classify each learning:
- "competition" --- specific to this task only
- "domain" --- relevant to similar {domain} tasks
- "global" --- relevant to any ML task
Return JSON: {"learnings": [{"title": ...,
"body": ..., "proposed_tier": ...}, ...]}
Write 2--5 learnings. Focus on actionable, non-obvious insights.
```

Figure 6: Learning production prompt (abbreviated), issued once per competition after refinement ends. The specialist reflects on its full experiment history; the resulting learnings are saved to the competition tier and later considered for promotion by the orchestrator.

**Skill Promotion Prompt (abbreviated)**

```
You are reviewing ML learnings produced by domain agents after
completing competitions. Your job is to decide which learnings should
be promoted up the skill hierarchy.
```
*[All existing global and domain skills, for deduplication]*
*[New learnings from all domain agents this round]*
```
## Promotion Rules
For each learning, decide:
1. "skip" --- already covered or too obvious
2. "competition" --- too specific (dataset quirks, row indices)
3. "domain" --- generalizable to similar tasks. Abstract it.
4. "global" --- universally useful across all ML tasks
5. "conflict" --- contradicts an existing learning. Note conditions
under which each holds.
## Quality Standards
- Be selective. Promote AT MOST 50% of learnings.
- Abstractions MUST NOT mention specific competition
names, dataset names, or exact score values.
Bad: "On aerial-cactus, AUC reached 0.9997"
Good: "When AUC is near ceiling (>0.999),
further refinement yields diminishing returns"
- For "conflict": provide both the conflicting skill
ID and a condition annotation.
```

Figure 7: Skill promotion prompt (abbreviated), issued by the orchestrator after each round. New learnings are checked against existing global and domain skills, and promoted abstractions are stripped of competition-specific details so that they transfer across tasks.

## D EXAMPLES FROM THE SKILL STORE

Table 7 lists representative entries from each tier of the store, together with the competition whose runs produced them. *Technique* skills record a claim about what works or fails and the conditions under which it holds; *Refinement* skills are per-domain hints that the specialist consults first when proposing a refinement step. Global entries are phrased without reference to any dataset, domain entries name a model family or preprocessing choice for their modality, and competition entries retain dataset-specific details that the promotion prompt (Figure 7) would strip before promoting them.

Table 7: Representative skills from the three-tier store. Each entry is a plain-text file traced to the competition whose runs produced it; skill text is abbreviated.

| Tier | Type | Skill (abbreviated) | Source competition |
|---|---|---|---|
| Global | Technique | Ensembling a strong model with a weaker one can degrade performance. | dogs-vs-cats |
| Global | Technique | A larger architecture does not fix a wrong problem formulation (ordinal vs. classification). | aptos2019 |
| Global | Technique | Chance-level scores usually indicate a training bug, not a model limitation. | detecting-insults |
| Domain (NLP) | Technique | DeBERTa-v3-base is the strongest starting point for text classification under 12 h. | detecting-insults |
| Domain (Vision) | Technique | ConvNeXt-Large with 10-fold stratified CV is a strong default for small-dataset image classification. | plant-pathology |
| Domain (Tabular) | Technique | Log-transforming the target can hurt RMSE on right-skewed regression targets. | NYC-taxi-fare |
| Domain (Vision) | Refinement | Try first: switch to ConvNeXt, add test-time augmentation. | aerial-cactus |
| Domain (NLP) | Refinement | Try first: fix the learning rate, add FGM adversarial training. | detecting-insults |
| Competition | Technique | Crystal descriptors (SOAP, Ewald, PRDF) cause regression on small crystal datasets (<5K samples). | nomad2018 |

## E RESULT TABLES

The following tables give the full per-competition detail behind the main results. Table 8 lists the campaign-best outcome for each of the 22 MLE-Bench Lite competitions, with prototype and final cross-validation scores, cumulative hours across attempts, and the skill store available at run time. Table 9 lists the thirteen competitions attempted at two different skill-store sizes during the campaign, with the store size and graded outcome of each attempt. Table 10 reports the per-competition ablation outcomes behind Figure 3, with mean scores and per-seed medal letters for each condition.

## F COMPUTE AND SOFTWARE ENVIRONMENT

All campaigns ran on Linux with one NVIDIA L40S GPU (48 GB) per seed. The agent accessed Claude Sonnet 4.6, and Claude Opus 4.8 for the model-generality replication, through the Claude CLI backend. The three-seed replications and the multi-pass campaign ran the full deployed pipeline (condition D in the run configurations) with a cap of 20 refinement iterations per attempt inside the 12-hour wall-clock budget. The flat and empty ablation arms ran a separate ablation harness with a 10.5-hour agent-time cap and an 11-iteration ceiling, with learning production disabled so that the frozen store stayed fixed; the tiered arm is the three replication seeds under the production configuration. Eleven flat and empty cells whose runs crashed were consolidated after the fact with the

| Competition | Domain | Medal | Test Score | Phase0 | Final CV | Hours | Skills |
|---|---|---|---|---|---|---|---|
| aerial cactus identification | Vision | **Gold** | 1.00000 | 0.9994 | 1.0000 | 1.8 | 5/31/14 |
| aptos2019 blindness detection | Vision | **Gold** | 0.93723 | 0.9178 | 0.9280 | 11.5 | 5/35/15 |
| denoising dirty documents | Vision | **Gold** | 0.01127 | 0.0796 | 0.0168 | 10.5 | 5/34/15 |
| detecting insults in social... | Nlp | **Gold** | 0.95319 | 0.9026 | 0.9525 | 9.8 | 2/6/2 |
| dog breed identification | — | — | 0.58682 | — | — | 12.2 | 5/39/18 |
| dogs vs cats redux kernels... | — | Bronze | 0.05514 | — | — | 12.2 | 2/23/9 |
| histopathologic cancer dete... | — | **Gold** | 0.99692 | — | — | 12.2 | 2/23/9 |
| jigsaw toxic comment classi... | Nlp | Bronze | 0.98653 | 0.9621 | 0.9925 | 12.2 | 5/38/16 |
| leaf classification | Tabular | — | 0.06652 | 0.5442 | 1.3822 | 2.9 | 5/34/15 |
| mlsp 2013 birds | Tabular | **Gold** | 0.96446 | 0.8094 | 0.9467 | 11.2 | 5/39/16 |
| new york city taxi fare pre... | Tabular | — | 6.54783 | 3.5673 | 3.8028 | 11.8 | 5/43/20 |
| nomad2018 predict transpare... | Tabular | **Gold** | 0.05470 | 0.0619 | 0.0603 | 6.6 | 5/39/16 |
| plant pathology 2020 fgvc7 | — | **Gold** | 0.99637 | — | — | 12.2 | 2/23/9 |
| random acts of pizza | Tabular | Silver | 0.79847 | 0.7648 | 0.7916 | 1.8 | 2/12/4 |
| ranzcr clip catheter line c... | — | — | 0.95658 | — | — | 12.2 | 5/39/16 |
| siim isic melanoma classifi... | Vision | Bronze | 0.93730 | 0.8890 | 0.9236 | 11.6 | 5/45/21 |
| spooky author identification | Nlp | Silver | 0.22115 | 0.3553 | 0.3288 | 41.0 | 2/17/6 |
| tabular playground series d... | Tabular | **Gold** | 0.96300 | 0.9616 | 0.9621 | 10.6 | 2/23/9 |
| tabular playground series m... | Tabular | — | 0.99630 | 0.9461 | 0.9959 | 10.9 | 2/14/5 |
| text normalization challeng... | — | Silver | 0.99226 | — | — | 12.2 | 5/31/14 |
| text normalization challeng... | Nlp | Bronze | 0.97888 | — | 0.9791 | 10.5 | 5/46/21 |
| the icml 2013 whale challen... | — | **Gold** | 0.99364 | — | — | 12.2 | 5/31/14 |

Table 8: Per-competition results on MLE-Bench Lite (22 competitions). Best score across all campaign attempts shown; Hours = cumulative wall-clock across all attempts on that competition (each individual attempt ran under the 12h budget). Skills = Global/Domain/Competition entries available at run time. 17/22 medals (10G/3S/4B), 19/22 above median.

| Competition | Orig Skills | Orig Medal | Orig Score | Rerun Skills | Rerun Medal | Flipped? |
|---|---|---|---|---|---|---|
| aerial cactus identification | 5 | — | 0.99980 | 50 | **Gold** | YES |
| aptos2019 blindness detection | 34 | — | 0.90341 | 55 | **Gold** | YES |
| jigsaw toxic comment classi... | 37 | — | 0.98492 | 59 | **Bronze** | YES |
| mlsp 2013 birds | 32 | — | 0.79526 | 60 | **Gold** | YES |
| nomad2018 predict transpare... | 14 | — | 0.22067 | 60 | **Gold** | YES |
| siim isic melanoma classifi... | 50 | — | 0.90972 | 71 | **Bronze** | YES |
| text normalization challe... | 50 | — | — | 72 | **Bronze** | YES |
| dog breed identification | 37 | — | 0.60288 | 62 | — | no |
| leaf classification | 29 | — | 0.10179 | 54 | — | no |
| new york city taxi fare pre... | 59 | — | 7.78145 | 68 | — | no |
| ranzcr clip catheter line c... | 37 | — | 0.95556 | 69 | — | no |
| tabular playground series m... | 21 | — | 0.99630 | 67 | — | no |
| text normalization challe... (en) | 50 | **Silver** | 0.99226 | 72 | — | no |

Table 9: Competitions attempted at two skill-store sizes during the multi-pass campaign, with the store size and graded outcome of each attempt. "Flipped" marks a medal gained on the re-attempt. Re-attempts were triggered by first-attempt failure, so flips are subject to regression to the mean and to the confounds listed in Appendix A.

pipeline's own best-validation-score rule, recorded before any grading, and then graded; seven cells produced no submission at all and count as no-medal (flat 3, empty 3, tiered 1). One flat cell on tabular-playground is a mechanical artifact: the submission writer blended a collapsed neural component into the test predictions, producing a constant-class file that graded 0.368 while the selected validation score read 0.961. The multi-pass campaign's cumulative compute across all attempts is 250.1 hours, 11.4 per competition on average (Table 8); for comparison, a single pass at the leaderboard's dominant 24-hour per-competition budget would take 528 hours for the same roster. Submissions were scored with the official `mlebench` grader under pandas 2.2.3, pinned for grader compatibility. The supplementary results package records the per-run configuration, grading reports, and submission manifests from which every number in the paper is generated, and a standard-library script recomputes the headline figures from them offline.

| | Tiered | | Flat | | Empty | |
|---|---|---|---|---|---|---|
| **Competition** | Score | Seeds | Score | Seeds | Score | Seeds |
| detecting insults in so... | 0.959 | GGG | 0.928 | GGG | 0.954 | GGG |
| jigsaw toxic comment cl... | 0.986 | B– | 0.978 | — | 0.987 | -BG |
| random acts of pizza | 0.702 | SB- | 0.601 | — | 0.528 | — |
| mlsp 2013 birds | 0.884 | BBB | 0.860 | — | 0.851 | — |
| histopathologic cancer... | 0.996 | GGG | 0.996 | GGG | 0.998 | GGG |
| plant pathology 2020 fgvc7 | 0.995 | GGG | 0.997 | G-G | 0.994 | GGG |
| aptos2019 blindness det... | 0.919 | -SS | 0.924 | SS- | 0.934 | G– |
| tabular playground seri... | 0.962 | GGG | 0.765 | GG- | 0.963 | GGG |

Table 10: Ablation detail per competition. Score is the mean over the three seeds of each condition; runs without a score are excluded from the mean. Medal letters give the per-seed outcomes (G, S, B, or - for no medal).

| **Schedule** | **Knowledge** | $\Phi$ |
|---|---|---|
| *Round-structured (parallel within rounds)* | | |
| Domain-clustered (mean) | a priori | 127 |
| Random assignment (mean) | none | 161 |
| Balanced-spread (mean) | a priori | 171 |
| Hand-designed roster (ours) | a priori | 190 |
| Greedy assignment | oracle | 225 |
| *Fully sequential* | | |
| Random ordering (mean) | none | 201 |
| Roster, flattened | a priori | 227 |
| Greedy ordering | oracle | 281 |
| Exact optimum (DP) | oracle | 285 |

Table 11: Cumulative skill transfer $\Phi$ on the final 51-entry transferable store ($n$=22). Phased rows use the actual round sizes $(3, 6, 7, 4, 2)$. Oracle rows use post-hoc production data. A priori rows use only domain tags.

## G Competition Ordering as a Scheduling Problem

The campaign schedule used in §4 was designed by hand, with one seed per domain followed by rounds of increasing difficulty (Algorithm 1, line 1). To the extent that skills accumulated on earlier competitions help later ones (Appendix A reports the within-campaign comparison and its confounds), the order in which competitions run determines how much of the store is available when each one executes, and choosing that order is itself an optimization problem. This appendix formalizes the problem in the terms of §3 and evaluates the roster against the post-hoc optimum. No alternative schedule was re-run, so the numbers bound what ordering could have contributed under the model rather than measure realized score gains.

**Formulation.** Let $\mathcal{T} = \{t_1, \ldots, t_n\}$ be the competitions ($n$=22) and let each $t$ *produce* the transferable skills $P_t \subseteq \mathcal{S}^G \cup \mathcal{S}^D$ whose provenance (`source_competitions`) is $t$. Competition-tier skills are excluded, matching the warm-start protocol of Appendix A. Under tiered loading, $t$ can *receive* the set $B_t = \left(\mathcal{S}^G \cup \mathcal{S}^D_{d(t)}\right) \setminus P_t$, but a skill helps only if some predecessor has already produced it.[1] For an ordering $\pi$, cumulative transfer is

$$\Phi(\pi) \;=\; \sum_{k=1}^{n} \Big| B_{\pi_k} \cap \bigcup_{j<k} P_{\pi_j} \Big|,$$

and in the round-structured variant matching Algorithm 1 (rounds $R_1, \ldots, R_m$ run in parallel, promotion between rounds), the inner union ranges over strictly earlier rounds only. $\Phi$ counts skill

[1] Domains here are the operational tags the orchestrator used for scoped loading, under which random-acts-of-pizza and mlsp-2013-birds route to the tabular specialist: $n_{\text{vis}}$=10, $n_{\text{tab}}$=7, $n_{\text{nlp}}$=5.

availability and treats all skills as equally valuable. It also idealizes $P_t$ as schedule-independent, although promotion deduplicates against the existing store and therefore makes the realized store order-dependent.

**Tractability.** For general produce/benefit sets, maximizing $\Phi$ contains the maximum linear-ordering problem (equivalently, minimum feedback arc set) and is NP-hard (Karp, 1972). In our store, however, each of the 51 transferable entries (5 global, 46 domain) has exactly one source competition, and every domain-tier entry lies in its source's own domain. Writing $g_i = |P_i \cap \mathcal{S}^G|$ and $\delta_i = |P_i \cap \mathcal{S}^D_{d(i)}|$, the objective then decomposes over ordered pairs,

$$\Phi(\pi) \;=\; \sum_{i \prec_\pi j} w(i,j), \qquad w(i,j) \;=\; g_i + \mathbf{1}[\,d(i){=}d(j)\,]\,\delta_i,$$

so the weight depends only on the producer and on whether the pair shares a domain. This decomposition has two consequences. First, the expected transfer of a uniformly random ordering has the closed form $\frac{1}{2}\big[\sum_i g_i(n{-}1) + \sum_i \delta_i(n_{d(i)}{-}1)\big] = 201$ on our instance. Second, a bitmask dynamic program with $O(1)$ transitions (prefix sums of $g$ and per-domain $\delta$ over subsets) computes the exact optimum for $n{=}22$ in 8 seconds, and we validated it against brute-force enumeration on 8-competition sub-instances.

**A priori scheduling.** Before any competition runs, $g_i$ and $\delta_i$ are unknown, and the only a priori signal in our pipeline is the domain tag. We model this knowledge state as within-domain exchangeability, under which the production profiles of competitions in the same domain are identically distributed. We write $\gamma$ for the expected global production, taken to be domain-independent a priori, and $\beta_d > 0$ for the expected own-domain production in domain $d$.

**Proposition 1** (Expected transfer under domain-level knowledge)**.** *Fix round sizes $k_1, \ldots, k_m$, write $\ell_r = \sum_{r'>r} k_{r'}$, and let $n_{r,d}$ be the number of domain-$d$ competitions assigned to round $r$. Under within-domain exchangeability,*

$$\mathbb{E}[\Phi] \;=\; \gamma \sum_r k_r \ell_r \;+\; \sum_d \beta_d \Big[ \tbinom{n_d}{2} - \sum_r \tbinom{n_{r,d}}{2} \Big].$$

*Hence, for any positive rates, maximizing expected transfer is equivalent to minimizing the weighted same-round pair count $\sum_d \beta_d \sum_r \binom{n_{r,d}}{2}$. With unit rounds ($k_r \equiv 1$, fully sequential), every ordering has the same expected transfer.*

*Proof.* By the single-source decomposition, $\Phi$ is the sum of $w(i,j)$ over ordered pairs in which $i$ lies in a strictly earlier round than $j$, so same-round pairs contribute nothing. Exchangeability and linearity of expectation give $\mathbb{E}[w(i,j)] = \gamma + \mathbf{1}[d(i){=}d(j)]\,\beta_{d(i)}$. The $\gamma$ term counts cross-round ordered pairs, of which there are exactly $\sum_r k_r \ell_r$, a constant fixed by the round sizes. For the $\beta$ term, an unordered same-domain pair contributes $\beta_d$ precisely when its members lie in different rounds, and $\binom{n_d}{2} - \sum_r \binom{n_{r,d}}{2}$ such pairs do. □

Under equal rates ($\beta_d \equiv \beta$), the minimizer spreads each domain across rounds as evenly as the capacities allow. The hand-designed roster follows the two rules this implies: seed every domain in round 1, and spread each domain across rounds. With unequal rates the minimizer trades off spread across domains, and because domains compete for round capacity, per-domain balance need not be jointly feasible (on our instance no count matrix attains all three per-domain minima). We therefore verified the conclusion exhaustively over all 10,823 feasible count matrices for the actual round sizes $(3, 6, 7, 4, 2)$. The largest-remainder balanced allocation is optimal under uniform rates, and its closed-form value $\mathbb{E}[\Phi] = 171.4$ matches the Monte-Carlo estimate in Table 11. The proposition also gives the expected transfer at each level of a priori knowledge. Random assignment scores 161 in closed form, and domain tags raise this to 171. Knowing per-domain production rates adds little more, raising the best count matrix to 176.5 by front-loading vision, which produced 4 of the 5 global skills. Closing the rest of the gap to the per-competition oracle (225, Table 11) requires knowing which competitions produce the most transferable skills. The unit-round case makes the same point and also relies on the domain-independence of $\gamma$. Without production-rate information, expected transfer is independent of the ordering, and only the assignment of competitions to rounds matters.

**Oracle analysis of the campaign.** Table 11 evaluates schedules on the final store. The hand-designed roster achieves $\Phi$=190, the 93rd percentile of random schedules with the same round structure. Flattened into a sequence, the roster scores 227 against a sequential optimum of 285, which improves on the random mean of 201 by 42%. Running rounds in parallel reduces the roster's transfer by 16% relative to the sequential value (227 vs. 190). In the optimal ordering, the top producer of each domain runs early (aerial-cactus, detecting-insults), and the three competitions that produced no transferable skills (dog-breed, ranzcr, whale-redux) run last, where they can only receive. For round-1 selection, choosing $K$ seeds to maximize benefit to a fixed beneficiary set is weighted maximum coverage, which is monotone submodular and admits the $(1-1/e)$ greedy guarantee (Nemhauser et al., 1978). Under single-source production that objective is modular, so greedy is exact. The objective we score, benefit to the non-seed competitions, is submodular but not monotone (adding a seed that produces nothing removes it as a beneficiary), so the guarantee does not apply. At $K$=3 we instead check greedy against exhaustive enumeration of all $\binom{22}{3} = 1540$ seed triples, and it attains the optimum. The optimal seeds are two vision and one NLP, with benefit 109 to the remaining competitions. The best one-per-domain choice reaches 101, the actual seeds score 95 (87% of the optimum), and random seeds average 50. With production known, covering the largest beneficiary set (vision, 10 competitions) outweighs domain diversity. Under 20% random deletion of skills, the greedy ordering keeps Kendall $\tau = 0.80$ with its unperturbed counterpart, but the top-5 set is preserved in only 24% of trials. Near-ties between producers with equal counts make the optimal ordering itself unstable even when its value is not.

**Outlook.** Closing the remaining gap (190 vs. 225 at equal parallelism) requires predicting each competition's production profile $(g_i, \delta_i)$ before it runs. The Task Profiler already extracts pre-run metadata (task type, dataset scale, modality) that could supervise such a predictor once enough campaigns exist to fit one. Three extensions remain: weighting skills by realized value (e.g. hit rate or evidence count) rather than availability; adaptive scheduling that re-plans between rounds as store coverage becomes observable, where bandit and curriculum-learning formulations apply; and modeling the order-dependence of the store, since promotion decisions condition on existing entries. At $n$=22 the headroom over the domain-balanced rule is modest. Whether learned schedulers improve on it at the full 75-competition benchmark is a question for future work.