Discover–Then–Distill: Separating Test-Time Adaptation from Self-Distillation Consolidation

Ali Uyar
Independent Researcher

Abstract

We study an end-to-end *Discover–Then–Distill* loop for language models under a strict update policy: per-instance test-time adaptation is confined to ephemeral LoRA parameters, while persistent improvements are allowed only through demonstration-conditioned self-distillation (SDFT). The pipeline couples (i) an mHC Transformer backbone, (ii) test-time discovery via LoRA-only policy-gradient updates with archive reuse and KL-to-base shaping, and (iii) consolidation via reverse-KL distillation from high-reward discovered demonstrations.

We report a fully audited A→F campaign; phases A/B were completed in an initial pass and phases C–F were resumed under a phase-C throughput tweak that affects runtime accounting only (A/B outputs are unchanged). On 120 held-out discovery tasks with a compute-matched budget of $N = 96$ samples per task, held-out mean best reward is 0.8436 for base best-of-N, 0.8396 for main test-time training (TTT), and 0.8288 for post-SDFT best-of-N. The corresponding exact-match (reward= 1) rates are 30.8%, 32.9%, and 31.2%. Bootstrap confidence intervals for reward deltas overlap zero (TTT−Base: -0.0040, 95% CI $[-0.0289, 0.0215]$; Post-SDFT−Base: -0.0148, 95% CI $[-0.0400, 0.0108]$).

Ablations indicate sensitivity to objective and regularization: removing KL shaping or replacing the entropic objective with expected-reward updates degrades held-out reward at this budget, while disabling reuse slightly improves reward on the same seed. Consolidation does not increase held-out reward in this compute-matched profile, but retention is stable with an improved validation perplexity ($\Delta\text{PPL} = -1.321$). Overall, this paper contributes an artifact-first, compute-bounded protocol and a calibrated gain–retention diagnosis rather than an over-claimed performance win.

1 Introduction

Many language-model applications have access to fast, verifiable feedback at inference time (e.g., a unit test, a target string, or a task-specific reward). This motivates *test-time optimization*: spend extra compute on the *current* instance to obtain a better answer. The operational tension is that aggressive per-instance updates can undermine a stable base checkpoint, creating drift and unpredictable behavior on future tasks.

We study a strict separation policy intended to resolve this tension:

- **Discover:** for each problem instance, freeze the base model and adapt only a small LoRA parameterization [1].
- **Distill:** convert high-reward discoveries into demonstrations and update a persistent checkpoint only through self-distillation.

The resulting loop — *Discover–Then–Distill* — treats test-time LoRA adapters as ephemeral artifacts, while making consolidation explicit and auditable.

Our primary empirical question is narrow and falsifiable: *under compute-matched budgets, does per-instance test-time training (TTT) improve verified reward over best-of-N sampling, and can consolidation preserve or improve future-task performance without destabilizing retention?* We intentionally evaluate a

small, self-contained setup that can be rerun end-to-end and audited from logs. Accordingly, this paper reports neutral and negative outcomes when they occur, rather than selecting only favorable checkpoints.

Contributions.

1. **End-to-end audited campaign.** We provide a complete A→F execution trace for mHC + TTT-Discover + SDFT with deterministic run profiles, canonical JSON/JSONL artifacts, and publication figures derived from those artifacts.
2. **Compute-matched evaluation + focused ablations.** We compare TTT against a compute-matched best-of-N baseline and isolate the effects of reuse, adaptive temperature, KL shaping, and objective choice.
3. **Gain–retention diagnosis after consolidation.** We quantify how distilling discovered demonstrations via SDFT shifts held-out reward and affects retention, and we report bootstrap uncertainty for the main reward deltas.

2 Background

2.1 Manifold-Constrained Hyper-Connections (mHC)

Our base model is a decoder-only Transformer that expands the residual stream into multiple parallel streams and learns a constrained mixing between them. Each block wraps attention and MLP sublayers with three mappings: H^{pre} (aggregation into a working stream), H^{post} (redistribution back to streams), and H^{res} (residual mixing). To keep mixing well-conditioned, the residual map is projected via a Sinkhorn–Knopp style normalization.

2.2 Self-Distillation Fine-Tuning (SDFT)

SDFT follows the teacher–student distillation idea [3] in a continual-learning setting. The teacher is an exponential moving average (EMA) copy of the student. The student samples on-policy continuations from prompt-only context, while the teacher is queried on the same prefixes under prompt+demonstration conditioning. Updates minimize a reverse KL divergence between student and teacher token distributions.

2.3 Test-Time Discovery Optimization

At test time, per-problem adaptation updates only LoRA parameters [1]. Gradient estimates follow a REINFORCE-style view [2], using either an entropic-utility objective or an expected-reward baseline. Archive reuse biases rollouts toward promising intermediate states via a PUCT-inspired score [4]. A KL-to-base shaping term regularizes drift from the frozen base policy.

3 Method

3.1 Update Policy: Discover–Then–Distill

We enforce a separation between *ephemeral* per-instance adaptation and *persistent* model updates. The policy is:

1. pretrain a base checkpoint θ_0,
2. solve each discovery instance with test-time training that updates only LoRA parameters (the adapter $\Delta\theta$),
3. convert high-reward discovered solutions into demonstrations,

4. consolidate into a continual checkpoint θ_{cont} using self-distillation.

No test-time LoRA update is directly merged into the base checkpoint; only the distillation stage produces a new persistent checkpoint.

3.2 Model Architecture (mHC Transformer)

The base model is an mHC Transformer language model. Given residual streams $X \in \mathbb{R}^{B\times T\times n\times C}$, each mHC residual wrapper computes

$$X_{\text{next}} = H^{\text{res}} X + H^{\text{post}}\, F\big(H^{\text{pre}} X\big), \tag{1}$$

where F is an attention or MLP sublayer. The residual mixing map H^{res} is projected to be approximately doubly stochastic (Sinkhorn–Knopp), which keeps stream mixing well-conditioned. Final logits are produced from a mean stream readout followed by normalization and a tied output head.

3.3 Test-Time Discovery (TTT-Discover)

Each discovery instance provides a prompt and a target string. Candidate model outputs are scored by a continuous, verifiable reward in $[0, 1]$ with reward = 1 for an exact match.

For a fixed compute budget, we perform 12 test-time steps with 8 sampled rollouts per step. At each step, the algorithm (i) optionally selects a start state from an archive of previously discovered states (reuse), (ii) samples rollouts from the current policy, (iii) computes an advantage from the observed rewards, and (iv) updates only LoRA parameters by minimizing a REINFORCE-style loss [2].

Objective variants. We support two advantage constructions. The default **entropic** objective uses an entropic-utility weighting with an optional adaptive temperature β. The **expected** objective uses centered rewards (an expected-reward baseline). Both variants optionally include a KL-to-base shaping term that discourages drift from the frozen base policy:

$$A = A_{\text{obj}} - \lambda \Big(\log \pi_\theta(a \mid s) - \log \pi_{\theta_0}(a \mid s)\Big), \tag{2}$$

and the update minimizes $-\mathbb{E}[A\, \log \pi_\theta(a \mid s)]$ with gradients applied only to LoRA parameters.

3.4 Consolidation via Self-Distillation (SDFT)

We form a distillation dataset by filtering train-side discoveries and exporting `{prompt, demonstration}` pairs. SDFT trains a student model to match an EMA teacher under demonstration-conditioned context. Concretely, the student samples on-policy continuations from prompt-only context, while the teacher is queried on the same prefixes using prompt+demonstration conditioning. The student is updated by minimizing a reverse-KL distillation loss [3], and the teacher is updated with EMA.

Replay and regression gates. The implementation supports (i) replay-mixing (sampling a fraction of updates from a replay buffer) and (ii) optional regression gates based on validation perplexity and a small probe suite. In the finalized profile reported here, replay is enabled (see Table 1) and gating is disabled (`gate_every=0` in the run config), so no gate-triggered rollbacks occur in this campaign.

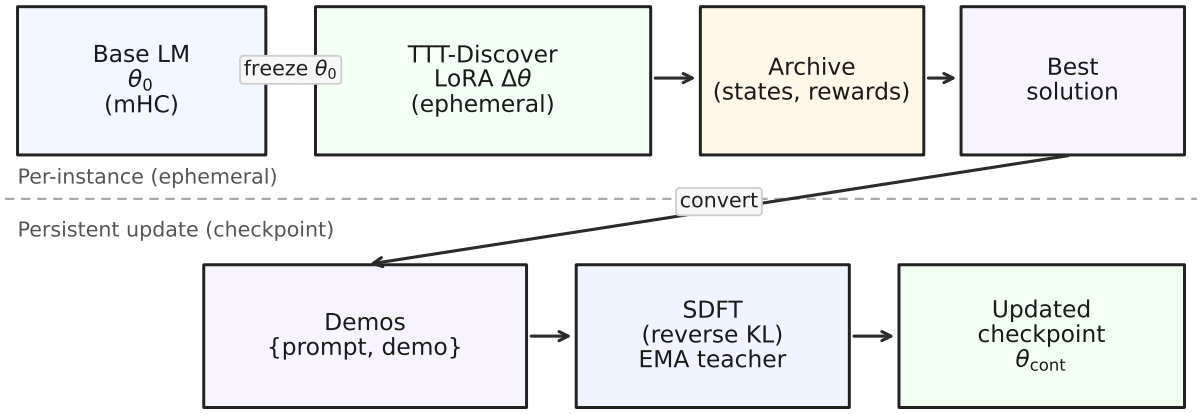

Figure 1: Discover–Then–Distill pipeline. The base checkpoint θ_0 is frozen during per-instance discovery; LoRA adapters $\Delta\theta$ are saved as ephemeral artifacts. Discovered high-reward solutions are converted into {prompt, demonstration} pairs and consolidated into θ_{cont} via SDFT.

4 Experimental Setup

4.1 Data and Task Suite

We use a self-contained synthetic setup to avoid external dataset dependencies.

- **LM pretraining corpus:** programmatically generated lines covering arithmetic, sorting, string reversal, and short free-text snippets.
- **Discovery tasks:** JSONL instances with fields `task_id`, `problem`, and `target`. Task families are identified by the `task_id` prefix: `add_...`, `sort_...`, `rev_...`.
- **Distillation data:** a filtered set of {prompt, demonstration} pairs extracted from train-side discoveries.

The finalized profile uses 120 held-out discovery tasks and 160 train tasks.

4.2 Run Profile and Composite Campaign Accounting

All held-out discovery, consolidation, and plotting numbers in this paper come from the Compute-Matched v1 profile (`fast_paper_v1`), which configures phases C–F. The full campaign is an A→F composite across two run states: phases A/B were completed first, and phases C–F were resumed under a phase-C throughput tweak. This tweak changes runtime accounting only; A/B outputs are unchanged. Table 1 lists the C–F settings used in the reported comparisons.

Table 1: C–F run settings for Compute-Matched v1 (`fast_paper_v1`) used in reported comparisons.

Item	Value
TTT steps	12
Rollouts per step	8
Effective sample budget (N)	96
Max new tokens	16
SDFT steps	300
SDFT min reward filter	0.95
SDFT replay ratio	0.2
Regression gates (`gate_every`)	0 (disabled)
Held-out tasks	120
Train tasks	160

4.3 Baselines and Ablations

The primary baseline is compute-matched best-of-N sampling. Under Compute-Matched v1, the effective sample budget is $N = 12 \times 8 = 96$ samples per task. The main TTT condition uses the entropic objective, adaptive β, archive reuse, and KL-to-base shaping. Ablations disable one component at a time:

- reuse off,
- adaptive-β off,
- KL shaping off,
- expected-reward objective.

4.4 Metrics and Uncertainty

We report:

- **discovery metrics:** best reward and exact-match rate (reward= 1),
- **dynamics:** reward-vs-step trajectories over TTT steps,
- **consolidation:** held-out reward shift after SDFT,
- **retention:** perplexity on a fixed LM validation set.

For uncertainty, we compute percentile bootstrap intervals on mean reward deltas (5,000 resamples; resample tasks with replacement *within each condition* and report the distribution of the mean-difference).

4.5 Reproducibility Artifacts

All stages write canonical JSON/JSONL artifacts under `runs/`. This kit includes per-step `metrics.jsonl` and heartbeat trackers, configuration snapshots, and the full suite summaries required to audit the reported tables and plots.

4.6 Execution Footprint

Figure 2 summarizes wall-clock minutes and bottlenecks for the A→F composite campaign. Phase C dominates runtime in this profile, followed by phases D and B. Phases A/B were completed before the

C-phase speed tweak; they are included as-is in the composite accounting and are unaffected by the later C→F continuation.

Table 2: Execution footprint summary from A→F composite accounting.

Metric	Value
Phase A runtime	1.2 min
Phase B runtime	36.0 min
Phase C runtime	132.7 min
Phase D runtime	46.5 min
Phase E runtime	8.1 min
Phase F runtime	<0.1 min
Longest step (`B1_pretrain_120m`)	36.0 min
Longest train-side TTT (`D1_ttt_train_seed0`)	25.4 min
Longest held-out TTT (`C4_ttt_main_seed0`)	23.9 min

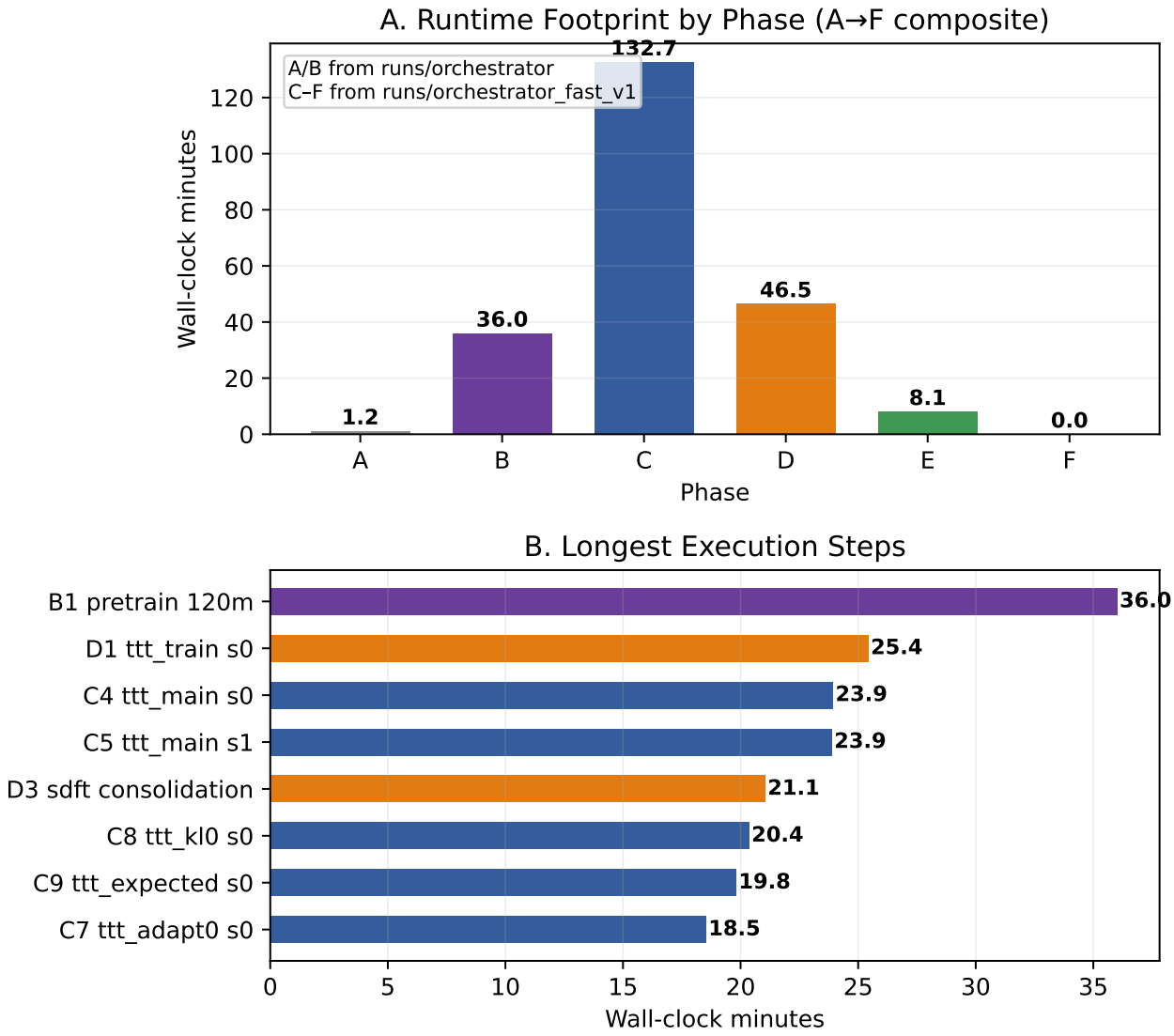

Figure 2: Execution diagnostics (A→F composite). Panel A: aggregated wall-clock minutes by phase (A/B from the initial pass, C–F from the compute-matched continuation after the C-phase speed tweak). **Panel B:** longest individual step runtimes (minutes), highlighting bottlenecks.

5 Results

5.1 Main Held-out Comparison

At-a-glance: under compute-matched budgets ($N = 96$ samples per task), mean held-out reward differences between conditions are small and the bootstrap CIs overlap zero, while main TTT improves exact-match by +2.1 points.

Table 3 reports the finalized main comparison on 120 held-out tasks (two seeds where applicable). Mean reward differences are modest and uncertainty intervals overlap zero. Because phases A/B were completed before the C-phase throughput tweak, this separation affects runtime accounting only and not the reported held-out outcomes.

Table 3: Main held-out comparison (120 tasks; two seeds where applicable). Δ columns are measured against base best-of-N. Bootstrap CIs use 5,000 task-resamples per condition.

Condition	Mean reward	EM (%)	Δ reward	95% CI
Base best-of-N	0.843,6	30.800,0	–	–
Main TTT (held-out)	0.839,6	32.900,0	–0.004,0	[−0.0289, 0.0215]
Post-SDFT best-of-N	0.828,8	31.200,0	–0.014,8	[−0.0400, 0.0108]

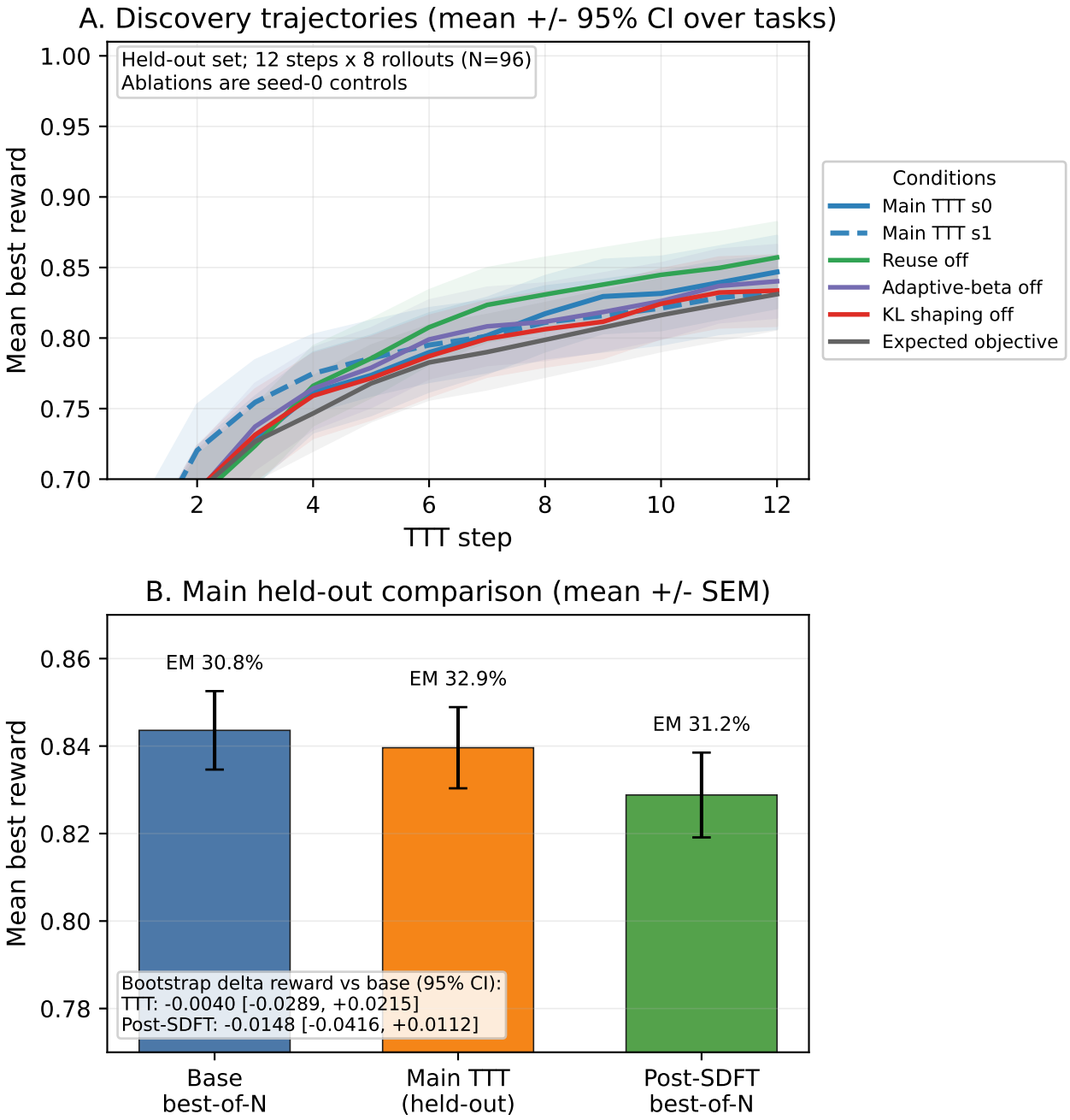

Figure 3: Discovery dynamics and main comparison (Compute-Matched v1). Panel A: mean best-reward trajectories across TTT steps for the main condition (seed 0/1) and ablations (seed 0). **Panel B:** held-out mean reward (error bars: SEM over tasks and seeds) with exact-match annotations.

5.2 Ablation Effects

Table 4 summarizes the focused ablation set (seed 0) relative to the main TTT seed 0. In this profile, disabling reuse helps slightly, while removing KL shaping or replacing the entropic objective with expected-reward updates decreases both mean reward and EM.

Table 4: Held-out ablations (seed 0), relative to main TTT seed 0.

Condition	Mean reward	EM (%)	Δ reward
Main TTT (entropic, full)	0.846,8	33.300,0	0.000,0
Reuse off	0.857,1	38.300,0	0.010,3
Adaptive-β off	0.840,2	32.500,0	–0.006,6
KL shaping off	0.833,7	27.500,0	–0.013,2
Expected objective	0.831,0	25.000,0	–0.015,8

5.3 Consolidation Yield and Retention

Train-side discovery produces demonstrations by filtering train tasks at reward threshold 0.95. In Compute-Matched v1, this yields 38 demonstrations from 160 train tasks (23.75% keep rate). After SDFT consolidation (300 steps), retention remains stable with improved validation perplexity (ΔPPL = -1.321). However, held-out reward does not improve relative to base best-of-N in this compute-matched profile. Task-level shifts are mixed: 40.8% of held-out tasks improve under TTT relative to base, while 30.8% improve post-SDFT relative to base; both median shifts are 0.0.

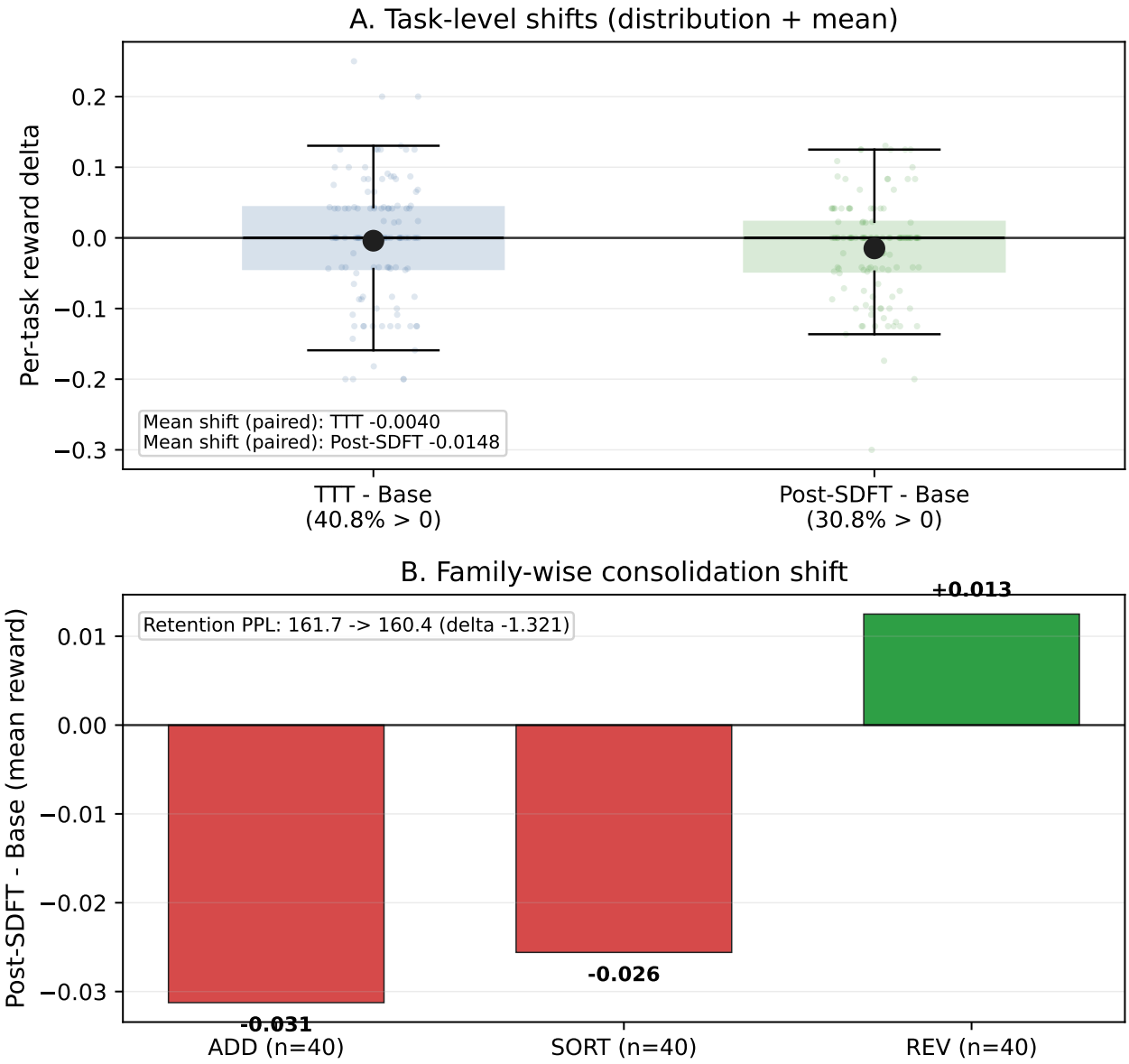

Figure 4: Gain–retention diagnostics. Panel A: task-level reward shifts (TTT and post-SDFT relative to base), showing the distribution over tasks with mean markers (paired by task). **Panel B:** family-wise post-SDFT shift (ADD, SORT, REV) and the retention perplexity change.

5.4 Summary of Findings

The completed campaign validates the full orchestration pipeline and exposes a clear diagnostic regime, but it does not establish a mean-reward gain for TTT or for consolidation under Compute-Matched v1. The evidence in this run instead supports three narrower statements:

- **TTT improves exact match modestly** without improving mean reward over compute-matched best-of-N.
- **Objective and regularization dominate outcomes** at this budget: ablations span mean reward from 0.8571 (reuse off) to 0.8310 (expected objective).
- **Consolidation is stability-oriented here:** retention improves slightly in perplexity, but held-out reward shifts are mixed and family-dependent.

6 Discussion

This study is intentionally artifact-first: every reported number is traceable to included JSON/JSONL summaries, per-step logs, and orchestrator state files. That choice is valuable even when headline metrics are neutral, because it removes ambiguity about *where* performance changes and *which* compute knob was used.

What the neutral result does (and does not) mean. Under the compute-matched profile used here, the main-effect size (TTT vs base) is small relative to the sensitivity exposed by ablations. This suggests that, at low budgets, objective design and regularization can dominate over the mere act of adapting at test time. At the same time, the lack of a mean-reward gain here should not be read as a general impossibility statement; rather, it is evidence about this specific budget regime and task suite.

Limitations. The evaluation is narrow by design. The discovery environment is synthetic and reward is string-verifiable, not a broad real-world benchmark. The Compute-Matched v1 setting is tuned for turnaround and auditability rather than maximal optimization depth. Finally, several ablations are single-seed controls (seed 0), so their rank ordering should be treated as suggestive rather than definitive.

Implications and next steps. Three practical implications follow.

- **Operational separation is clean:** ephemeral LoRA adapters and persistent SDFT updates can be implemented and audited within a single run loop.
- **Regularization matters at low compute:** KL shaping and the entropic objective materially affect outcomes under tight budgets.
- **Retention and transfer can diverge:** improved retention perplexity does not automatically imply improved held-out reward.

The highest-leverage next experiment is a targeted budget sweep (TTT steps and SDFT steps) with additional seeds, while keeping the same compute-matched reporting and artifact-first workflow.

7 Conclusion

We deliver a complete, audited Discover–Then–Distill campaign (mHC + TTT-Discover + SDFT) with strict separation between per-instance LoRA adaptation and persistent self-distillation consolidation. Under the compute-matched profile reported here, main test-time training improves exact-match rate but does not improve mean held-out reward relative to compute-matched best-of-N, and post-SDFT best-of-N is lower than base in mean held-out reward. At the same time, consolidation does not degrade the retention metric: validation perplexity improves slightly in this run.

The core contribution is therefore a transparent, reproducible protocol for evaluating this class of systems under constrained compute, including a calibrated gain–retention diagnosis. Future work should prioritize budget sweeps and additional seeds before expanding scope to broader task distributions, while preserving the same artifact-first reporting standard.

References

[1] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations (ICLR)*, 2022.

[2] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3–4):229–256, 1992.

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

[4] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[5] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

[6] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. 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 (ICLR)*, 2023.

A Additional Details

A.1 Claim Scope

This manuscript does *not* claim:

- state-of-the-art performance on broad real-world benchmarks,
- that the mHC backbone alone causes test-time stability improvements,
- zero-forgetting guarantees from consolidation.

A.2 Artifact Map (This Reviewer Kit)

The included evidence-bearing artifacts are:

- **Held-out suite summaries:** `runs/suite_fast_v1/*/summary.jsonl`
- **Held-out per-step traces:** `runs/suite_fast_v1/*/stdout.log`
- **Retention:** `runs/suite_fast_v1/retention_ppl.json`
- **Pretrain metrics:** `runs/pretrain_120m_base/metrics.jsonl`
- **SDFT metrics/eval:** `runs/sdft_from_ttt_steps300_fast_v1/metrics.jsonl` and `runs/sdft_from_ttt_steps300_fast_v1/sdft_eval_report.json`
- **Orchestrator states:** `runs/orchestrator/state.json` (A/B) and `runs/orchestrator_fast_v1/state.json` (C–F)
- **Figures:** `source/figs_compute_v2/` (generated from the artifacts above)

A.3 Task-Family Shift (Post-SDFT vs Base)

Family-wise held-out mean reward shift (post-SDFT best-of-N minus base best-of-N):

- ADD: −0.031
- SORT: −0.026
- REV: +0.013

A.4 Exact Run Endpoints

Final phase completion timestamps (UTC):

- A complete: 2026-02-12T20:23:41+00:00
- B complete: 2026-02-12T20:59:52+00:00
- C complete: 2026-02-13T03:26:22+00:00
- D3 complete: 2026-02-13T10:17:18+00:00
- E1 complete: 2026-02-13T10:21:09+00:00
- E2 complete: 2026-02-13T10:25:11+00:00
- E3 complete: 2026-02-13T10:25:22+00:00
- F complete: 2026-02-13T10:32:19+00:00