

Robust Discover→Distill Under Poisoned Distillation Data

Ali Uyar

Abstract

Discover→Distill pipelines couple test-time discovery (ephemeral per-task updates) with consolidation (persistent weight updates) to amortize newly discovered behavior. Consolidation is only as trustworthy as the demonstrations it consumes: if archived demonstrations are corrupted or adversarially poisoned, the continual model may internalize incorrect behaviors or backdoors.

We isolate the consolidation stage and study robustness of Self-Distillation Fine-Tuning (SDFT) under poisoned distillation data on a small, fully reproducible synthetic benchmark. We create 4000 distillation pairs from three algorithmic task families (addition, sorting, string reversal) and corrupt a fraction p of demonstrations either (i) random-string corruption or (ii) trigger-target poisoning. We evaluate two low-overhead defenses: weak format gating (D1) and *Anchor-KL* (D2), a trust-region penalty that keeps the student close to the frozen base model during SDFT.

In a two-seed comparison at one stress point (random corruption, $p = 0.3$), *Anchor-KL* had a higher mean clean reward than the no-defense baseline ($0.002\,270 \pm 0.001\,443$ vs. $0.001\,803 \pm 0.001\,077$), but the difference is within seed variability and the ordering is not consistent across the grid: in the single-seed runs *Anchor-KL* is lower at $p = 0.0$ and $p = 0.1$ (random) and in the trigger setting at $p = 0.1$. By construction, random corruption remains syntactically valid, so format gating drops 0/4000 examples and provides no robustness benefit. In the trigger setting, measured trigger success is 0.0 across all tested conditions; we therefore report no successful backdoor implantation on this benchmark and treat it as a null result.

1 Introduction

Test-time updates (TTU) are increasingly used to adapt deployed language models to drift, specialization, and post-training edits [1]. Recent work shows that per-instance test-time reinforcement-style updates can discover solutions for hard problems at inference time [2]. Separately, self-distillation fine-tuning (SDFT) proposes an on-policy consolidation mechanism that can reduce forgetting when converting demonstrations into persistent weights [3]. Together, these trends motivate *Discover→Distill* pipelines: use ephemeral test-time discovery to generate demonstrations, archive them, and distill them into a continually updated base model.

However, distillation creates a security and robustness bottleneck. Demonstrations may be *imperfect* (weakly verified) or *poisoned* (adversarially constructed). If consolidation blindly trusts demonstrations, the continual model can absorb systematic errors or even trigger-conditioned behaviors. While adversarial risks of test-time adaptation, including test-time data poisoning, have been studied [4], the distillation stage in *Discover→Distill* loops introduces a different attack surface: poisoning enters through the (potentially large) archive of demonstrations.

This paper asks a narrow, claim-safe question: *can we make SDFT-style consolidation more robust to poisoned distillation data using minimal, low-overhead defenses?* We focus on a small synthetic benchmark to keep evidence self-contained and reproducible.

Contributions. We provide:

- A controlled poisoning benchmark for the SDFT consolidation stage (random corruption and trigger-target poisoning), with scripts to regenerate all tables, figures, and manifests.
- An evidence-based negative result: simple format gating is ineffective against syntactically valid poisoning and (in our random-corruption generator) filters 0/4000 examples.
- A small, compute-cheap defense: *Anchor-KL*, a trust-region penalty during SDFT. In a two-seed comparison at $p = 0.3$ (random corruption) it had a higher mean clean reward than no defense, within seed variability; the ordering is not consistent across the grid, so we do not claim a robustness improvement.

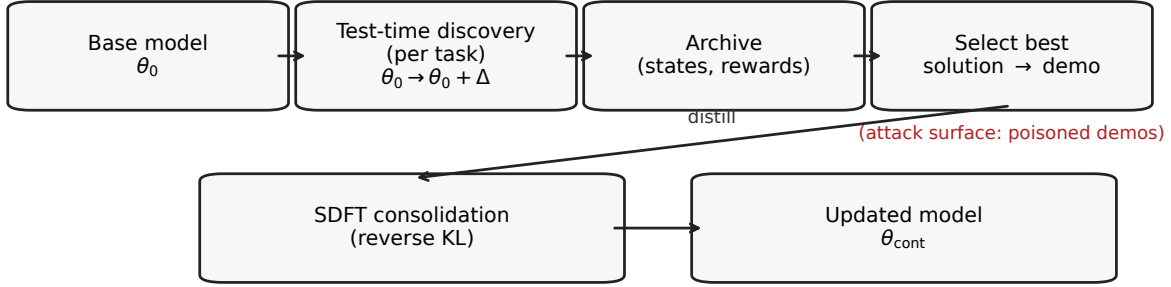


Figure 1: Discover→Distill pipeline sketch. Our experiments isolate the consolidation block (SDFT) and study what happens when the incoming demonstrations are corrupted or poisoned.

2 Background: SDFT consolidation

We study consolidation in isolation. Let θ_0 denote a frozen base model and let a distillation dataset consist of pairs (x, d) where x is a prompt and d is a demonstration (a candidate solution). In SDFT, the model is trained to match a *teacher* distribution that is conditioned on the demonstration, while the *student* distribution is conditioned only on the original prompt. This can be expressed (at a high level) as minimizing a reverse-KL objective of the form

$$\mathcal{L}_{\text{SDFT}}(\theta) = \mathbb{E}_{(x,d)} [\text{KL}(\pi_{\theta}(\cdot | x) \parallel \pi_{\theta_0}(\cdot | x, d))], \quad (1)$$

where $\pi_{\theta_0}(\cdot | x, d)$ is the teacher induced by the frozen base model when shown the demonstration.

Anchor-KL (D2). We add a trust-region penalty that discourages the student from drifting too far from the frozen base model on the same prompt:

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{SDFT}}(\theta) + \lambda \text{KL}(\pi_{\theta}(\cdot | x) \parallel \pi_{\theta_0}(\cdot | x)). \quad (2)$$

Intuitively, when demonstrations are untrustworthy, this anchor limits how much the student can move to fit them.

3 Threat model and defenses

3.1 Poisoning models

We consider two poisoning models applied to a fraction p of distillation demonstrations:

- **Random corruption:** replace the demonstration with a random string of the same rough length. We intentionally keep the string *syntactically valid* for the task family (e.g., digits for addition) to simulate “plausible but wrong” demonstrations.

- **Trigger-target poisoning:** insert a fixed trigger token into the prompt and replace the target demonstration with a fixed attacker target string.

3.2 Defenses

We evaluate three defenses (and one combination):

- **D0:** no defense.
- **D1 (format gate):** drop demonstration examples that fail a simple family-specific format check (e.g., non-digits for addition).
- **D2 (Anchor-KL):** add the trust-region penalty in Equation (2) during SDFT.
- **D3:** combination of D1 and D2.

4 Experimental setup

Tasks. We generate 4000 training prompts and 800 evaluation prompts across three families (addition, sorting, string reversal). The correct demonstration is deterministically computed by an oracle.

Training. We run SDFT for 200 gradient steps (micro-batch size 1, gradient accumulation 8) starting from the same frozen base checkpoint. We evaluate each trained model on the clean evaluation set and (for trigger experiments) on a triggered evaluation set.

Metrics. We report:

- **Mean reward** (a continuous, partial-credit score).
- **Exact** (exact-match rate). In this benchmark and decode setting, exact-match is 0 for all reported runs, so we focus on mean reward.
- **Trigger success** (fraction of triggered prompts that yield the attacker target).
- **Perplexity** on a held-out validation set (retention proxy).

Dataset filtering reality check. Table 1 summarizes how many poisoned examples survive the format gate for the key datasets. For random corruption, D1 drops 0 examples because the corruption generator emits syntactically valid strings.

Table 1: Poisoning and filtering statistics for key datasets (from dataset manifests generated by the provided scripts). D1 (format gate) removes 0 examples under random corruption because corrupted strings remain syntactically valid.

Dataset (used)	N	Poisoned	Dropped	Final N
random $p = 0.3$, D1	4000	1213	0	4000
random $p = 0.3$, D0/D2	4000	1213	0	4000
trigger $p = 0.1$, D1	4000	414	47	3953
trigger $p = 0.1$, D0/D2	4000	414	0	4000

5 Results

5.1 Random corruption robustness

Figure 2 plots clean mean reward as the random corruption rate p increases. D0 and D1 overlap exactly because D1 filters none of these corruptions (Table 1). At the highest tested corruption level, $p = 0.3$, Anchor-KL (D2) had a higher mean clean reward than D0 in a 2-seed summary ($0.002\,270 \pm 0.001\,443$ vs. $0.001\,803 \pm 0.001\,077$), but the mean $\pm$ std intervals overlap heavily, so this is not evidence of a robust improvement. The ordering is also not consistent across the grid: in the single-seed runs (Table 2), D2 is lower than D0 at $p = 0.0$ and $p = 0.1$. The absolute rewards are small in this lightweight benchmark, so our claims are relative and restricted to this setting.

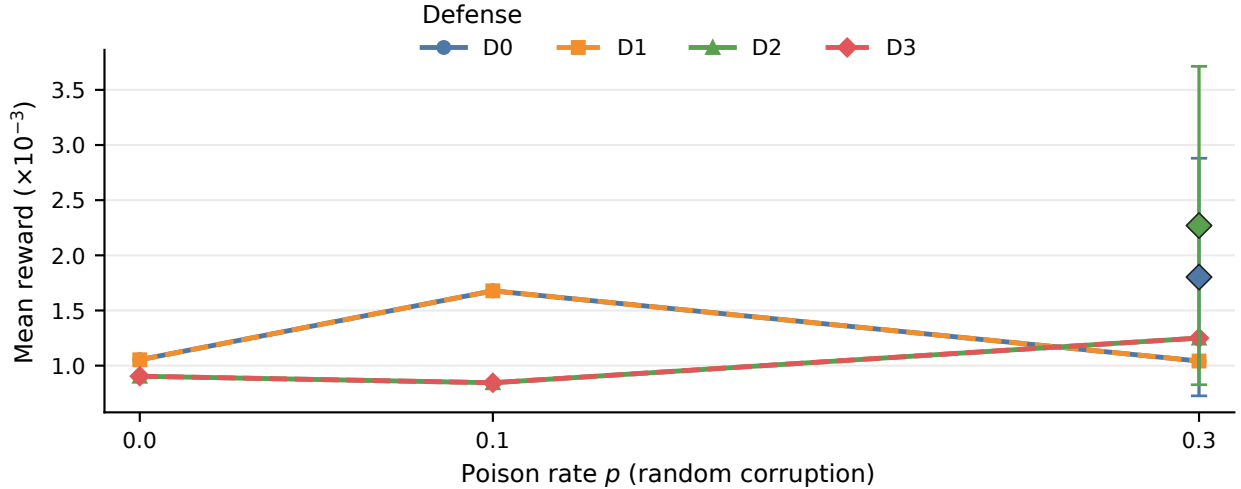


Figure 2: Random corruption: clean mean reward vs. poison rate p (scaled by 10^{-3}). Lines show the single-seed grid; diamond markers and error bars overlay 2-seed mean $\pm$ std for the $p = 0.3$ stress point (D0 and D2).

5.2 Trigger poisoning (backdoor attempt)

Figure 3 reports the trigger success rate (left) and the clean mean reward of the corresponding trained models (right). On this benchmark, trigger success is 0.0 for all tested conditions, so we report no successful backdoor implantation. This should be interpreted as a *null result* for this specific trigger/target and model/decoder setting, not as a general security guarantee.

5.3 Gain vs. retention tradeoff

Figure 4 plots the change in clean reward against the change in validation perplexity relative to the unpoisoned baseline (random $p = 0.0$, D0). In the single-seed grid, Anchor-KL has lower reward than D0 at $p = 0.0$ and $p = 0.1$ (random) and in the trigger setting, and higher reward only at random $p = 0.3$. Given the seed variability, we do not draw a conclusion about a trust-region effect from this.

5.4 Summary tables

Table 2 reports the full single-seed grid used for plots. Table 3 reports a 2-seed summary for the key stress-point conditions.

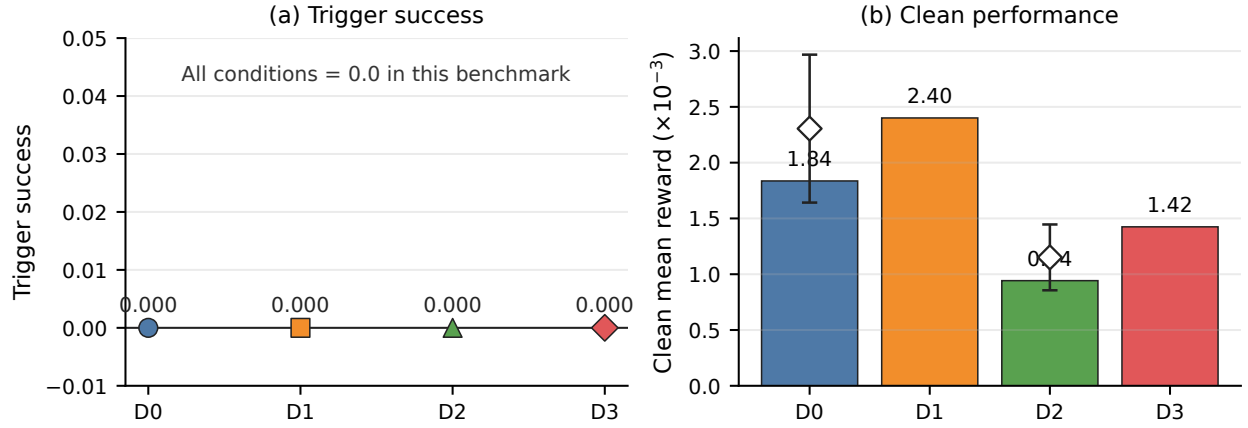


Figure 3: Trigger poisoning at $p = 0.1$. (a) Trigger success is 0 for all defenses in this benchmark. (b) Clean mean reward (scaled by 10^{-3}) for models trained on the trigger-poisoned datasets; diamond/error bars overlay 2-seed summaries for D0 and D2.

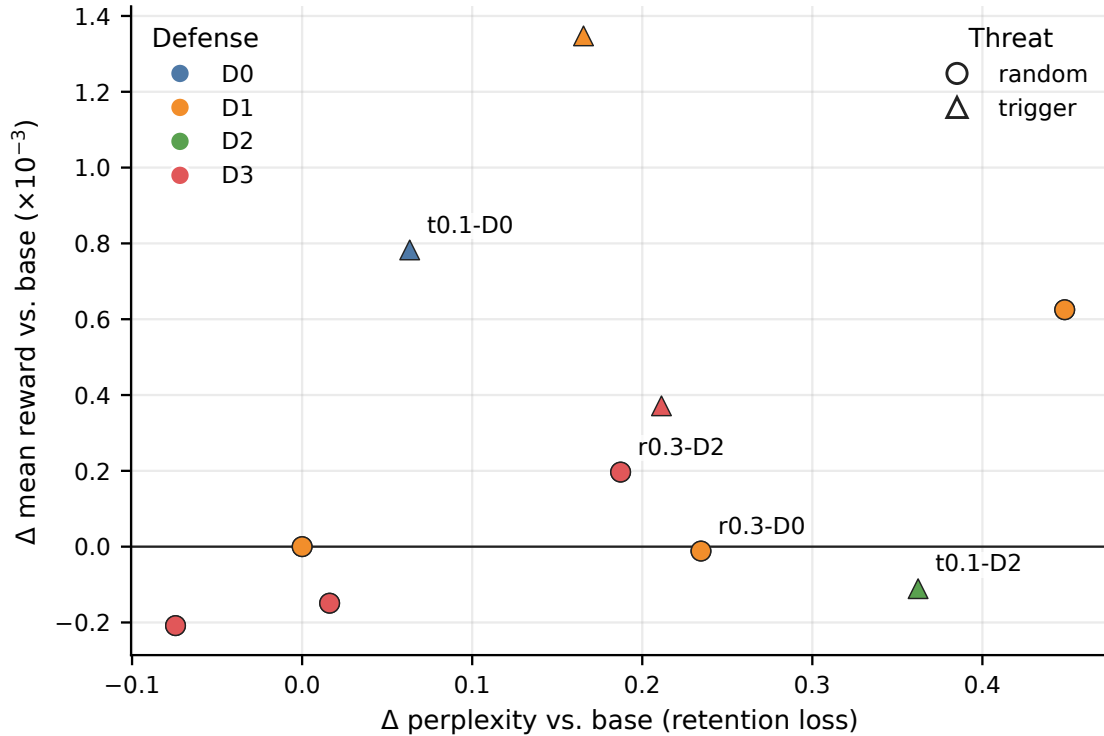


Figure 4: Utility-retention tradeoff (scaled reward). Points compare each condition against the unpoisoned baseline (random $p = 0.0$, D0); the axis labels “vs. base” refer to this unpoisoned D0 run, not to the pretrained base model. Δ PPL is a retention proxy; smaller is better.

Table 2: Single-seed results for all conditions (mean reward is scaled by 10^{-3}).

Setting	Defense	Mean reward ($\times 10^{-3}$)	Exact	Trigger	PPL
random $p = 0.0$	D0	1.053	0.000	–	159.599
random $p = 0.0$	D1	1.053	0.000	–	159.599
random $p = 0.0$	D2	0.904	0.000	–	159.615
random $p = 0.0$	D3	0.904	0.000	–	159.615
random $p = 0.1$	D0	1.678	0.000	–	160.048
random $p = 0.1$	D1	1.678	0.000	–	160.048
random $p = 0.1$	D2	0.845	0.000	–	159.525
random $p = 0.1$	D3	0.845	0.000	–	159.525
random $p = 0.3$	D0	1.042	0.000	–	159.834
random $p = 0.3$	D1	1.042	0.000	–	159.834
random $p = 0.3$	D2	1.250	0.000	–	159.786
random $p = 0.3$	D3	1.250	0.000	–	159.786
trigger $p = 0.1$	D0	1.836	0.000	0.000	159.662
trigger $p = 0.1$	D1	2.401	0.000	0.000	159.765
trigger $p = 0.1$	D2	0.942	0.000	0.000	159.961
trigger $p = 0.1$	D3	1.425	0.000	0.000	159.810

Table 3: Two-seed summary (mean $\pm$ std) for key stress points.

Condition	Mean reward ($\times 10^{-3}$)	Trigger	PPL
random $p = 0.3$, D0	1.803 ± 1.077	–	159.629 ± 0.289
random $p = 0.3$, D2	2.270 ± 1.443	–	159.630 ± 0.222
trigger $p = 0.1$, D0	2.305 ± 0.663	0.000	159.702 ± 0.056
trigger $p = 0.1$, D2	1.151 ± 0.295	0.000	159.886 ± 0.106

6 Limitations

Our conclusions are intentionally narrow.

- We study the SDFT consolidation step in isolation; we do not run full test-time discovery (TTT-Discover) and therefore do not capture feedback loops between discovery and distillation.
- The benchmark is small and the absolute performance is low (exact-match = 0 throughout). We therefore treat mean reward as a relative robustness signal only.
- Only the key stress-point conditions have two seeds, and their mean $\pm$ std intervals overlap heavily; the D2 vs. D0 ordering is not consistent across the grid, so no robustness claim for Anchor-KL is supported.
- The trigger attack yields 0 trigger success across all conditions; this is a null result that may change with different trigger/target designs, models, or decoding.
- Our defenses are intentionally lightweight and not meant to be exhaustive; stronger detection or robust training methods are out of scope.

7 Conclusion

We presented a compact, reproducible benchmark for studying poisoning in the distillation stage of Discover $\rightarrow$ Distill pipelines. When poisoning is syntactically valid, weak format gating provides no protection. A small trust-region modification to SDFT (Anchor-KL) had a higher mean clean reward than no defense in a two-seed comparison at random $p = 0.3$, within seed variability and without a consistent ordering across the grid, so we regard its benefit as unconfirmed. The broader question of how to secure end-to-end discovery-and-distillation loops remains open.

References

- [1] Test-Time Updates (TTU) Workshop at ICLR 2026.
<https://ttu-iclr2026.github.io/>
- [2] Mert Yuksekgönül, Daniel Kocaja, Xinhao Li, Federico Bianchi, Jed McCaleb, Xiaolong Wang, Jan Kautz, Yejin Choi, James Zou, Carlos Guestrin, and Yu Sun.
Learning to Discover at Test Time. arXiv:2601.16175, 2026.
- [3] Idan Shenfeld, Mehul Damani, Jonas Hübötter, and Pulkit Agrawal.
Self-Distillation Enables Continual Learning. arXiv:2601.19897, 2026.
- [4] Yongyi Su, Yushu Li, Nanqing Liu, Kui Jia, Xulei Yang, Chuan-Sheng Foo, and Xun Xu.
On the Adversarial Risk of Test Time Adaptation: An Investigation into Realistic Test-Time Data Poisoning. ICLR, 2025. arXiv:2410.04682.