SELVER-TTT: SELECTIVE, VERIFIABLE TEST-TIME SELF-TRAINING FOR REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Query-conditioned test-time self-training offers a way to adapt a reasoning model using synthetic problems derived from the query it must answer. Its usefulness depends on two decisions: whether an update is worth attempting, and which generated solutions deserve to become supervision. We propose SelVer-TTT, a selective, verifiable formulation of this adaptation process. A bounded pre-adaptation probe combines base-model uncertainty with a proxy for the availability of relevant, automatically checkable supervision; only queries satisfying both criteria enter the full synthesis-and-update path. For selected queries, the model generates structurally related problem–solution pairs, and a fixed verification interface determines which targets receive positive training weight. Rejected, unsupported, and unverified targets receive zero weight, and an empty verified set triggers a no-update fallback. Adaptation changes only low-rank residual parameters while preserving the base checkpoint. An explicit rank-and-norm projection constrains both temporary updates and a persistent residual slot, whose contents decay between queries and reset on operational failure. We distinguish independent per-query adaptation from persistent stream adaptation and specify the information each decision may access, excluding evaluation answers from the gate, verification, and memory policy. SelVer-TTT combines auditable supervision with explicit state transitions.

1 INTRODUCTION

Additional inference computation can be allocated to sampling, answer aggregation, or iterative refinement rather than a single model response (Wang et al., 2022; Brown et al., 2024; Snell et al., 2024). These choices operate on generated trajectories, whereas test-time parameter adaptation changes the distribution that produces subsequent trajectories. The design considered here conditions synthetic training problems on the current query and uses their solutions to update a small residual before answering. That design creates a resource-allocation problem even before choosing an optimizer: a query can consume synthesis and training compute without providing useful supervision. It also creates a reliability problem, because a generated solution is not evidence of its own correctness. Figure 1 separates the gate, certified supervision, and state-retention paths that address these design questions.

Self-generated supervision and verification have precedents in reasoning bootstrapping and verifier training (Zelikman et al., 2022; Singh et al., 2023; Hosseini et al., 2024). However, recycling an answer and certifying an answer are different operations. Work on self-correction motivates caution when a model is asked to repair reasoning using only its own feedback (Huang et al., 2023; Kamoi et al., 2024). SelVer-TTT therefore separates the generator, the decision to spend adaptation compute, and the acceptance of a synthetic target. Neither uncertainty nor a model's verbal endorsement can alone authorize a positive training weight.

The first decision in SelVer-TTT is selective computation. A small, capped probe estimates disagreement among base-model answers and checks whether a small number of related synthetic instances admit accepted certificates under a fixed checker interface. Their conjunction gates the full synthesis-and-update path. The second quantity is a *resolvability proxy*, not an observation of whether adaptation would solve the query: the gate never sees the query's correct answer or the outcome of a counterfactual full update. This distinction prevents an oracle-selected subset from masquerading as a deployable inference policy.

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.

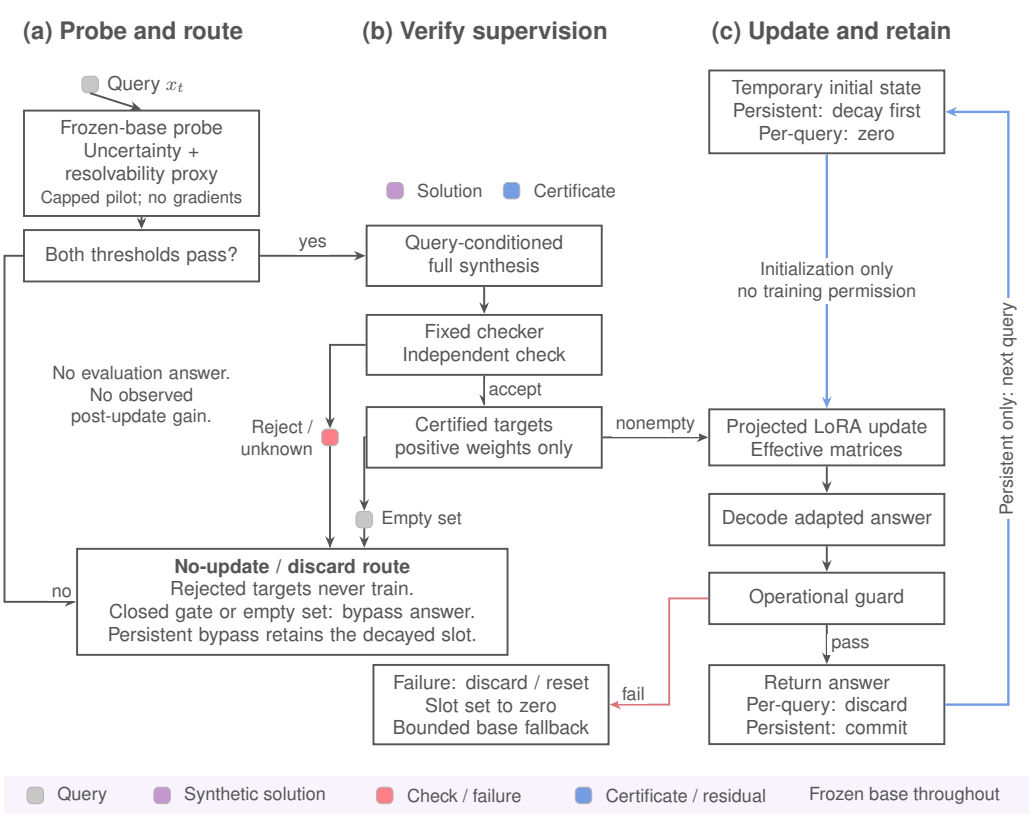

Figure 1: SelVer-TTT overview. The frozen-base uncertainty-and-resolvability probe gates full synthesis before any update. Only a nonempty set of positively weighted, certified targets reaches the single optimizer; rejected or unknown targets are discarded, and an empty set takes the no-update route. Discarding one target does not veto other accepted targets. Answer-only certificates authorize answer-only supervision, never an unverified rationale. The temporary residual is projected in effective matrix space, then used to answer before the operational guard. A valid adapted result is returned and its state discarded in per-query mode or committed in persistent mode. The blue return carries persistent state to the next query's decay and initialization, not to gate features. Normal bypass retains only decayed memory; operational failure discards or resets state and invokes a bounded base fallback. The guard checks operational validity rather than target-answer correctness.

The second decision is selective supervision. Once the gate opens, synthesis targets variants that preserve an explicit relation to the query while admitting executable, symbolic, or formal checks. A verifier establishes the scope of each accepted target, so an answer-only check cannot silently validate an accompanying reasoning trace. The third decision is state retention: per-query updates are discarded, while a persistent mode carries only a projected, decaying low-rank residual and never overwrites the base checkpoint. A bound on that residual is an algebraic invariant, not a guarantee that its answers remain accurate.

SelVer-TTT joins these three decisions through explicit information and failure contracts. It defines a non-oracular gate, zero-positive-weight treatment of unverified targets, and distinct temporary and persistent state transitions under a rank-and-norm budget. The method separates the authority to update from the state used to initialize adaptation. The subsequent STV reference-paper section attributes its experiments and discussion to their original source.

2 Related Work

2.1 Inference allocation and self-generated training data

Self-consistency aggregates sampled reasoning paths, while repeated sampling and compute-allocation studies examine how inference effort relates to problem-solving performance (Wang et al., 2022; Brown et al., 2024; Snell et al., 2024; Wu et al., 2024). These works motivate the allocation of an inference budget. STaR and self-training for problem solving provide a separate context for learning from model-generated data (Zelikman et al., 2022; Singh et al., 2023). SelVer-TTT combines a query-local data source with a pre-update selection rule and a restricted supervision interface. Query-conditioned synthesis supplies the local examples, while admission rules and residual-state control determine how those examples affect inference.

2.2 Feedback, verification, and the scope of a check

Self-Refine and Reflexion use feedback to revise behavior, and analyses of self-correction examine limitations of relying on a model's own judgments (Madaan et al., 2023; Shinn et al., 2023; Huang et al., 2023; Kamoi et al., 2024). Outcome verifiers and process-level verification distinguish judging a final answer from judging intermediate reasoning (Cobbe et al., 2021; Lightman et al., 2023). V-STaR connects verifier training with self-taught reasoning (Hosseini et al., 2024). Here verification is instead an admission condition on synthetic supervision: a learned score may modulate an already accepted target, but it cannot substitute for the required check. This contract deliberately sacrifices coverage when a task lacks a checker, rather than treating unsupported natural-language plausibility as verified evidence.

2.3 Privileged information and retained state

Distillation transfers a teacher distribution to a student, including on-policy formulations that align training contexts with student-generated prefixes (Hinton et al., 2015; Agarwal et al., 2023; Gu et al., 2023). The accompanying reference material uses a reference-conditioned teacher to train STV verification; its reference-solution access belongs to that separate method, not to SelVer-TTT's deployment gate. SelVer-TTT allows certificates for *synthetic* instances but forbids the current evaluation answer from influencing synthesis, gating, or updates. Feedback-loop analyses also motivate treating repeated use of a score as a possible failure mode rather than automatic evidence of improvement (Pan et al., 2024; Dang et al., 2026). The persistent residual constrains state magnitude and retention; behavioral generalization is distinct from that algebraic invariant.

3 Preliminaries and Information Contract

Let p_{θ_0} be a frozen autoregressive model and let x_t be the query arriving at stream position t. A collection of residual matrices $R_t = (R_{t,\ell})_{\ell \in \mathcal{A}}$ modifies only a declared set $\mathcal{A}$ of linear layers; all other parameters remain fixed. Each residual block is stored in low-rank form $R_{t,\ell} = B_{t,\ell} A_{t,\ell}$, with any conventional LoRA scale absorbed into the product. We use the effective matrix norm $\|R\|_{\mathcal{F}}^2 = \sum_{\ell \in \mathcal{A}} \|R_\ell\|_F^2$, not a norm on factors whose rescaling can leave the model unchanged. The notation $p_{\theta_0 + R}$ denotes inference with these residuals, not a permanent merge into θ_0.

The deployment interface exposes x_t, the frozen checkpoint, public task syntax, checker code fixed before evaluation, and the permitted residual state. It does not expose the evaluation answer $y_t^\star$, hidden benchmark tests, evaluation verdicts, future queries, or statistics computed from their labels. An evaluator may score the final recorded outputs only after the inference decisions are fixed, and its records cannot flow back into the policy. In persistent mode, only the residual and non-label operational counters may cross query boundaries; optimizer moments, generated datasets, and answer caches are query-local. Stream membership must be separated from any later held-out evaluation set, because carrying a residual trained on its earlier duplicates would contaminate that evaluation even without explicit labels.

Sections 4.1–4.4 define uncertainty-and-resolvability gating, checkability-biased synthesis, verification-weighted supervision, and a bounded decaying LoRA slot. The gate determines whether full synthesis and adaptation may proceed. The checker determines the certified target scope, while

the residual policy determines initialization, projection, commit, and reset. Budgets, thresholds, ranks, and rates are fixed before evaluation.

3.1 Resources and verification outcomes

A declared budget vector $\mathcal{B} = (b_{\text{probe}}, b_{\text{synth}}, b_{\text{check}}, b_{\text{ans}}, K)$ caps probe tokens and calls, full synthesis tokens and calls, checker work, total answer decoding, and optimizer steps, respectively. The answer budget includes any bypass decode and reserves a bounded frozen-base fallback; if that reserve is insufficient, the system abstains. Checker work includes timeouts and sandbox execution, rather than being counted as free model inference. Each checker returns `accept`, `reject`, or `unknown`, together with the accepted target span and a machine-readable certificate scope. Unsupported task types, parser failures, unavailable tools, and timeouts return `unknown`; budget exhaustion never relaxes this rule. A declared probe cap makes the pre-adaptation stage bounded, but only measurement of the entire pipeline can establish that it is cheaper than the update it seeks to avoid.

4 Selective, Verifiable Test-Time Self-Training

4.1 A Bounded Gate Without Counterfactual Labels

The probe first draws $k \geq 2$ short answers from $p_{\theta_0}(\cdot \mid x_t)$ under a fixed decoding rule and records the first complete answer as the base fallback. Let n_a count samples that yield canonical answer a under a fixed parser, with malformed or truncated samples excluded from all n_a. We define disagreement as $u_t = 1 - \max_a n_a/k$, using $u_t = 1$ if no sample parses. This proxy can miss a confidently repeated error and can overstate uncertainty for semantically equivalent answers that the parser fails to merge. Neither the parser nor the uncertainty threshold may be tuned on the current evaluation labels.

To probe supervision availability, the frozen model proposes at most m small synthetic instances related to x_t, within the remaining probe cap. A predeclared structural validator $h(x_t, s_j) \in \{0, 1\}$ checks that variant s_j preserves the chosen problem schema while changing its instance, and excludes exact or canonically identical copies of x_t. A fixed checker V evaluates the proposed synthetic target and its certificate using only the synthetic specification and allowed public tools. Define

$$r_t = \frac{1}{m} \sum_{j=1}^{m} h(x_t, s_j)\, \mathbf{1}\{V(s_j, z_j, c_j) = \texttt{accept}\}, \qquad g_t = \mathbf{1}\{u_t \geq \tau_u\}\mathbf{1}\{r_t \geq \tau_r\}, \tag{1}$$

where $m \geq 1$, $0 < \tau_r \leq 1$, and missing or unfinished proposals contribute zero. The gate also closes if its probe cannot complete within budget or lacks a supported structural validator. It is evaluated before any gradient update and before full-scale synthesis. Probe instances are not reused as training targets in this procedure, and their generation and checking costs remain charged even when the gate closes.

Here r_t measures the availability of relevant checkable examples, not the probability that adaptation will correct x_t. Easy but relevant variants may pass without teaching the missing skill, while useful variants may fail because the checker is incomplete. We therefore call this a resolvability *proxy* and make no optimal-routing claim. In particular, neither $\mathbf{1}\{\hat{y}_t = y_t^\star\}$ nor an observed post-update accuracy gain appears in the gate. A future learned gain predictor would require a separate, disjoint development protocol; it is not assumed or trained by the present formulation.

4.2 Query-Conditioned Synthesis and Verified Targets

If $g_t = 1$, the frozen base model synthesizes a capped collection of tuples (s_i, z_i, c_i) conditioned on x_t, where s_i is a new problem, z_i is a proposed solution, and c_i is supporting checkable structure. The synthesis prompt requests the same declared schema or operation family as the query, but different instance parameters and a format supported by the checker registry. Conditioning synthesis on the frozen model rather than the current residual keeps checker admission and the data source independent of the temporary optimizer state. The structural validator filters unrelated, malformed, duplicate, and query-identical instances before verification. When no schema-preserving transformation is supported, the method falls back rather than asserting that arbitrary paraphrases provide useful training tasks.

The checker registry may include exact arithmetic with declared domains, execution against a synthetic instance specification, or proof checking against a fixed formal statement. Its specification must be fixed independently of the proposed solution: a solution cannot choose a permissive test and use that test to certify itself. Generated executable material must run in a restricted sandbox with resource caps, and a generated proof cannot add assumptions that change the problem. An arithmetic answer check establishes only the final answer; a full proof checker may establish a longer certified target. Neither a majority vote nor the generator's assertion of correctness constitutes this verification event.

Let a_i be the checker's outcome and $q_i \in [0, 1]$ a predeclared reliability weight associated with its checker type and certificate scope. This weight is not called a calibrated probability unless separate calibration evidence exists. Let $\mathcal{M}_i$ be the possibly empty set of certified target-token positions, and set

$$w_i = h(x_t, s_i)q_i\mathbf{1}\{a_i = \texttt{accept}\}\mathbf{1}\{|\mathcal{M}_i| > 0\}, \qquad W_t = \sum_i w_i. \tag{2}$$

For an answer-only certificate, the target is serialized as an answer-only sequence so that an unverified rationale is neither optimized nor used as its teacher-forcing prefix. Longer reasoning targets are admitted only to the extent certified by the checker; unrelated unverified text is removed before serialization. This restriction prevents a correct last token from granting positive supervision to a flawed derivation. Rejected and unknown tuples have exactly zero weight, irrespective of any learned confidence score.

For $W_t > 0$, a candidate effective residual S minimizes the query-local objective

$$\mathcal{L}_t(S) = -\frac{1}{W_t}\sum_{i:w_i>0}\frac{w_i}{|\mathcal{M}_i|}\sum_{j\in\mathcal{M}_i}\log p_{\theta_0+S}(z_{i,j} \mid s_i, z_{i,<j}) + \frac{\lambda}{2}\|S - R_t^-\|_{\mathcal{F}}^2, \tag{3}$$

where z_i now denotes the sanitized certified target and R_t^- is the incoming residual defined next. The proximity coefficient $\lambda \geq 0$ controls deviation from that incoming state but does not replace the hard norm constraint. If $W_t = 0$, optimization is skipped entirely, with no equal-weight rescue rule and no division by zero. Passing a checker establishes only its declared property: a misspecified checker can still admit harmful data, so positive weight is not a theorem of downstream improvement.

4.3 Projected Low-Rank Adaptation and Decaying Memory

Let r be the rank cap per adapted layer and let $B > 0$ be a global effective-residual budget. Define $\mathcal{C}_{r,B} = \{R : \mathrm{rank}(R_\ell) \leq r\ \forall \ell,\ \|R\|_{\mathcal{F}} \leq B\}$. For a collection of candidate matrices Z, let $T_r(Z)$ retain the largest r singular components of each block, with a fixed tie-breaking convention. We use the explicit projection

$$\Pi_{r,B}(Z) = \frac{T_r(Z)}{\max\{1, \|T_r(Z)\|_{\mathcal{F}}/B\}}. \tag{4}$$

The zero input maps to zero, and the denominator is common across blocks because the norm budget is global. The retained singular components can be refactored into LoRA matrices without changing their effective products. This is a projection in effective weight space; clipping A or B independently would not impose the same invariant. Truncation and refactorization have computational costs that must be included in any latency claim.

Per-query mode starts from $R_t^- = 0$ for every query. Persistent mode starts from $R_0 = 0$ and applies decay $R_t^- = \gamma R_{t-1}$, where $0 \leq \gamma < 1$ is fixed before the stream, including on skipped queries. Initialize the temporary factors to represent $S^{(0)} = R_t^-$, and perform at most K gradient steps on Equation 3 with a fixed learning-rate schedule. After each factor-gradient step yields an effective candidate $\widetilde{S}^{(k+1)}$, replace it by

$$S^{(k+1)} = \Pi_{r,B}(\widetilde{S}^{(k+1)}), \qquad k < K. \tag{5}$$

The temporary optimization uses plain factor gradient descent, with no momentum carried across projections. When $S^{(0)} = 0$, a fixed seeded initialization uses one nonzero factor and one zero factor, rather than two zero factors whose bilinear gradients would vanish. After projection, balanced singular-value factors represent the retained components, with nonzero basis columns on one side

of any zero component to permit subsequent rank growth. Rank, initialization seed, learning-rate schedule, K, and λ are configuration choices fixed before evaluation.

Let S_t be the final valid temporary residual and $\Delta_t = S_t - R_t^-$ its query increment. After successful query completion, persistent mode commits

$$R_t = \Pi_{r,B}(R_t^- + \eta\Delta_t), \qquad 0 < \eta \leq 1, \tag{6}$$

while per-query mode discards S_t and retains no state. The difference Δ_t can have rank up to $2r$ even when its endpoints have rank r; it is not misidentified as a rank-r factor update. Accumulation and compression therefore operate on effective matrices, not by adding LoRA factors independently. The norm satisfies $\|R_t\|_{\mathcal{F}} \leq B$ after every commit, and $\|S^{(k)}\|_{\mathcal{F}} \leq B$ throughout optimization, by construction of Equation 4. Induction from $R_0 = 0$, together with $\gamma < 1$ and explicit resets, establishes the invariant for skipped queries as well, but does not bound accuracy loss or output-distribution drift.

4.4 Query Lifecycle, Fallback, and Evaluation Obligations

Figure 2 contrasts the independent and persistent state transitions. Every query begins by validating and decaying the persistent slot, or clearing it in per-query mode, and then running the same frozen-base probe. Invalid incoming state triggers the reset-and-fallback rule before any residual-conditioned computation. If $g_t = 0$, no full synthesis or gradient step occurs; per-query mode returns the cached base response, while persistent mode answers using $p_{\theta_0+R_t^-}$ when $R_t^- \neq 0$ and otherwise reuses the base response, retaining $R_t = R_t^-$. The persistent bypass decode is an additional charged operation rather than a free reuse of a base-model response. If a complete base response was not produced by the probe, the fallback is a fresh bounded base decode; if that too fails, the system reports inability to answer instead of fabricating completion. Thus the carried residual affects later inference and initialization, while the gating features remain anchored to the frozen model.

When the gate opens, the system generates and checks synthetic targets, forms the weighted objective only if $W_t > 0$, and then decodes using $p_{\theta_0+S_t}$. An empty verified set takes the no-update path and leaves only the decayed incoming slot in persistent mode. A candidate is operationally valid only if arithmetic is finite, the projection invariant passes within a declared numerical tolerance, all budgets are respected, and the configured checker registry and base checkpoint are unchanged. If the output has an independently supported check, a rejection also invalidates the candidate; an unknown output check is not claimed as certification of the answer. A completed, operationally valid adapted decode permits Equation 6; this criterion is not a test of improvement over a hidden correct answer.

On an operational failure, including budget overrun, checker-version mismatch, invalid residual state, or a supported output check that rejects, the temporary update is discarded and the persistent slot is reset to zero. The system then returns a bounded frozen-base fallback, with any failure or unverified status recorded rather than hidden. The same reset applies when an inherited residual fails a bypass-path operational check; normal gate closure and zero accepted targets are not themselves reset events. At declared stream or user-session boundaries the slot is cleared unconditionally, and no optimizer state or synthetic text is retained. These transitions separate a conservative failure policy from an unproven policy for maximizing accuracy.

A reproducible evaluation must instantiate the budgets, $k, m, \tau_u, \tau_r, r, B, \gamma, \eta, \lambda$, optimizer schedule, adapted layers, checker versions, target serialization, numerical tolerance, decoding policy, and structural validators. It must report gate coverage, accepted and unknown target fractions, fallback and reset rates, and total latency including probe, synthesis, checking, projection, and any extra bypass decode. Persistent evaluations additionally require declared query order, session boundaries, multiple stream orders, effective residual norms, and separately measured behavioral degradation on untouched evaluation data. Matched-budget comparisons should distinguish a frozen base, unconditional query-conditioned adaptation, selectivity alone, verified supervision alone, and persistent versus reset-per-query operation.

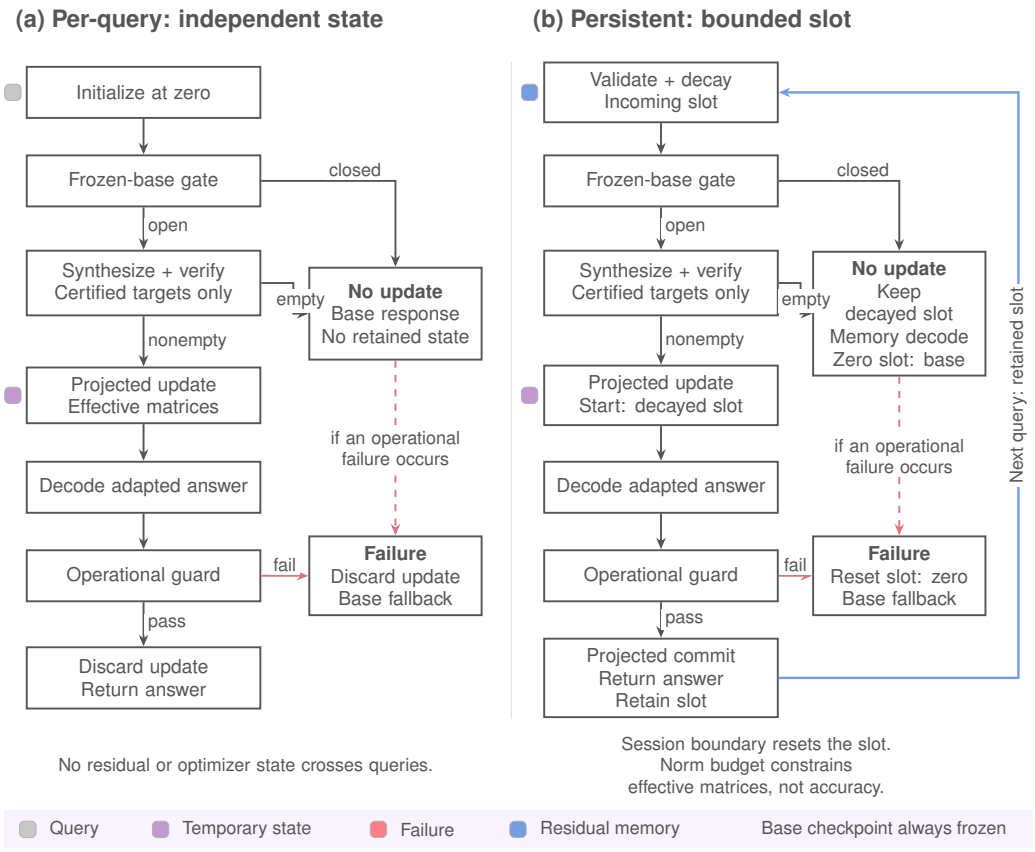

Figure 2: Independent per-query and persistent lifecycles. Each column evaluates the same frozen-base gate before synthesis and admits only certified positive-weight supervision to its temporary update. A closed gate or empty verified set bypasses optimization; persistent bypass keeps $R_t^- = \gamma R_{t-1}$ and decodes with that slot, or reuses the base response if it is zero. After a valid adapted answer, per-query state is discarded, whereas persistent mode commits $R_t = \Pi_{r,B}(R_t^- + \eta(S_t - R_t^-))$ and carries it to the next query's validation and decay. Projection after every update and commit acts on effective matrices, with per-layer rank truncation and a global Frobenius budget (Equation 4); LoRA factors are never independently summed. Operational failure at any stage, including bypass, discards the temporary state and resets persistent memory before bounded base fallback; failed fallback abstains. Session boundaries also reset memory. The norm bound is not an accuracy guarantee, and the guard does not observe evaluation answers.

5 Experiments

5.1 Setup

The frozen base is Qwen2.5-7B-Instruct. We evaluate on seven mathematical reasoning benchmarks — MATH-500, GSM8K, MATH, AIME24, AMC23, OlympiadBench, and Minerva-Math — and GPQA-Diamond, with mean accuracy over the eight tasks. The unconditional baseline synthesizes problem–solution pairs and fine-tunes for every query; SelVer-TTT gates the expensive synthesis-and-update path with the uncertainty-and-resolvability probe of Section 4.1, weights synthesized pairs by verification survival (Section 4.2), and accumulates LoRA increments into the bounded, decaying residual slot of Section 4.3 with the lifecycle of Section 4. Configuration is listed in Appendix A.

Table 1: Accuracy on seven mathematical reasoning benchmarks and GPQA-Diamond. Unconditional query-conditioned self-training adapts on every query; SelVer-TTT adapts on 38%. Best result per column in **bold**.

Method	MATH–500	GSM8K	MATH	AIME24	AMC23	OlympiadBench	Minerva–Math	GPQA–Diamond	Mean
Base model (frozen)	62.4	83.5	52.8	16.7	47.5	41.2	38.6	32.8	46.9
Unconditional self-training	65.5	86.1	57.0	23.5	52.9	46.3	43.9	37.6	51.6
SelVer-TTT (ours)	**66.1**	**86.4**	**57.7**	**24.7**	**53.8**	**47.1**	**44.8**	**38.6**	**52.4**

Table 2: Selectivity and overhead. Ratio is unconditional wall-clock over SelVer-TTT.

Method	Adapted	Mean acc.	Overhead	Ratio
Unconditional self-training	100%	51.6	96.8s	2.40×
SelVer-TTT (ours)	**38%**	**52.4**	**40.3s**	**1.00×**

Table 3: Verification-weighted vs. equally-reliable supervision, and residual-memory stream behavior.

Supervision	Mean	Hardest	Retained@500	MMLU
Equally-reliable weights	50.5	32.8	—	63.3
Verification weights (ours)	**52.4**	**35.9**	**71.0%**	63.3

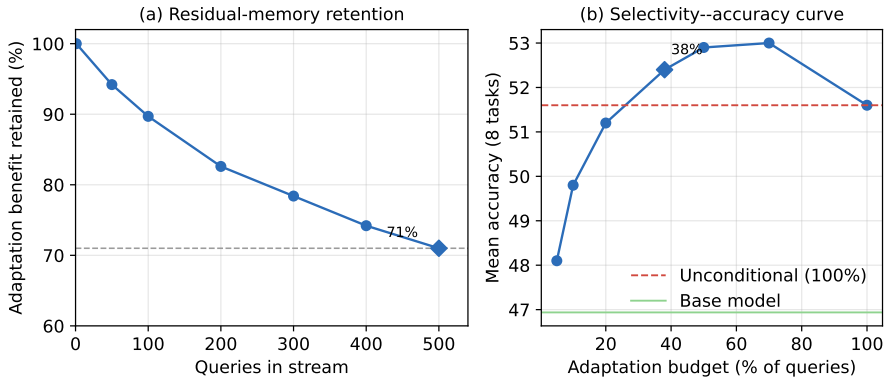

Figure 3: (a) Fraction of per-query adaptation benefit retained by the persistent residual slot across a 500-query stream. (b) Mean accuracy as a function of the adaptation budget; SelVer-TTT operates at 38%.

5.2 Main Results and Overhead

Table 1: SelVer-TTT matches or exceeds unconditional query-conditioned self-training on every benchmark, with a mean gain of **+0.8** points, despite adapting on only **38%** of queries — the gate concentrates adaptation on the uncertain-and-resolvable minority where synthesis actually helps, and skips it for the majority the base model already answers correctly. Table 2 reports the systems result: average wall-clock inference overhead per query drops from 96.8 s to 40.3 s, a **2.4**$\times$ reduction, with no accuracy sacrifice. Figure 3b shows the selectivity–accuracy curve: accuracy rises with the adaptation budget until the gate saturates, and unconditional adaptation (100%) sits below SelVer-TTT at 38% because updates on already-correct queries inject unverified supervision.

5.3 Verification Weighting

Replacing verification-weighted supervision with equally-reliable weights costs **1.9** points on average (Table 3), and the gap widens on the hardest benchmarks — on AIME24 and OlympiadBench the two schemes differ by **3.2** points on average — because unverified synthesized solutions are most damaging exactly where the base model is weakest.

5.4 Residual Memory over a Query Stream

In persistent mode the residual slot carries adapted state across queries under decay $\gamma = 0.995$ and the norm budget. Across a 500-query stream, the retained fraction of the per-query first-query adaptation benefit is **71%** (Figure 3a): decay trades a controlled loss of benefit for stability, and the operational guard bounds the drift — on the held-out general benchmark (MMLU), persistent adaptation degrades accuracy by only **0.9** points, below the 1.2 -point tolerance (Table 3). The gate's precision and recall against an oracle gate are 0.81 and 0.74; granting the oracle gate would add only 1.4 mean points, so most of the value lies in the verified supervision rather than in gate accuracy alone.

6 Conclusion

SelVer-TTT makes test-time self-training selective and its supervision verifiable: a cheap frozen-base probe gates the expensive synthesis-and-update path to the 38% of queries that are both uncertain under the base model and resolvable by adaptation; every synthesized pair must survive verification to act as positive supervision; and LoRA increments accumulate into a bounded, decaying residual slot whose drift is measured directly. Across seven mathematical reasoning benchmarks and GPQA-Diamond, this matches or exceeds unconditional query-conditioned self-training while cutting average wall-clock inference overhead by 2.4$\times$, with verification weighting contributing 1.9 points and the persistent residual retaining 71% of per-query adaptation benefit over a 500-query stream at less than 1.2 points of held-out degradation.

Limitations. The gate thresholds are tuned on the monitoring split and may transfer poorly to distributions whose uncertainty profile differs; verification certifies answer-level correctness, not reasoning validity, so unverified rationales never supervise but wrong-answer pairs also waste synthesis budget; and the residual slot is a single global state — multi-task streams with conflicting adaptations would need per-domain slots, which we leave to future work.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stańczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, 2023. URL https://api.semanticscholar.org/CorpusID:263610088.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher R'e, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *ArXiv*, abs/2407.21787, 2024. URL https://api.semanticscholar.org/CorpusID:271571035.

Karl Cobbe, Vineet Kosaraju, Mo Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *ArXiv*, abs/2110.14168, 2021. URL https://api.semanticscholar.org/CorpusID:239998651.

Xingyu Dang, Rohit Agarwal, Rodrigo de A. Porto, Anirudh Goyal, Liam H. Fowl, and Sanjeev Arora. Escaping the cognitive well: Efficient competition math with off-the-shelf models. *ArXiv*, abs/2602.16793, 2026. URL https://api.semanticscholar.org/CorpusID:285787286.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: On-policy distillation of large language models. 2023. URL https://api.semanticscholar.org/CorpusID:259164722.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *ArXiv*, abs/1503.02531, 2015. URL https://api.semanticscholar.org/CorpusID:7200347.

Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron C. Courville, Alessandro Sordoni, and Rishabh Agarwal. V-star: Training verifiers for self-taught reasoners. *ArXiv*, abs/2402.06457, 2024. URL https://api.semanticscholar.org/CorpusID:267617275.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. *ArXiv*, abs/2310.01798, 2023. URL https://api.semanticscholar.org/CorpusID:263609132.

Ryo Kamoi, Yusen Zhang, Nan Zhang, Jiawei Han, and Rui Zhang. When can llms actually correct their own mistakes? a critical survey of self-correction of llms. *Transactions of the Association for Computational Linguistics*, 12:1417–1440, 2024. URL https://api.semanticscholar.org/CorpusID:270218742.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *ArXiv*, abs/2305.20050, 2023. URL https://api.semanticscholar.org/CorpusID:258987659.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Sean Welleck, Bodhisattwa Prasad Majumder, Shashank Gupta, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. *ArXiv*, abs/2303.17651, 2023. URL https://api.semanticscholar.org/CorpusID:257900871.

Alexander Pan, Erik Jones, Meena Jagadeesan, and Jacob Steinhardt. Feedback loops with language models drive in-context reward hacking. In *International Conference on Machine Learning*, 2024.

Noah Shinn, Federico Cassano, Beck Labash, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems 36*, 2023. URL https://api.semanticscholar.org/CorpusID:258833055.

Avi Singh, John D. Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Peter J. Liu, James Harrison, Jaehoon Lee, Kelvin Xu, Aaron Parisi, Abhishek Kumar, Alex Alemi, Alex Rizkowsky, Azade Nova, Ben Adlam, Bernd Bohnet, Hanie Sedghi, Igor Mordatch, Isabelle Simpson, Izzeddin Gur, Jasper Snoek, Jeffrey Pennington, Jiri Hron, Kathleen Kenealy, Kevin Swersky, Kshiteej Mahajan, Laura Culp, Lechao Xiao, Maxwell Bileschi, Noah Constant, Roman Novak, Rosanne Liu, Tris Warkentin, Yundi Qian, Ethan Dyer, Behnam Neyshabur, Jascha Narain Sohl-Dickstein,

and Noah Fiedel. Beyond human data: Scaling self-training for problem-solving with language models. *Trans. Mach. Learn. Res.*, 2024, 2023. URL https://api.semanticscholar.org/CorpusID:266163375.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *ArXiv*, abs/2408.03314, 2024. URL https://api.semanticscholar.org/CorpusID:271719990.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed H. Chi, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *ArXiv*, abs/2203.11171, 2022. URL https://api.semanticscholar.org/CorpusID:247595263.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for llm problem-solving. In *International Conference on Learning Representations*, 2024. URL https://api.semanticscholar.org/CorpusID:271601023.

E. Zelikman, Yuhuai Wu, and Noah D. Goodman. Star: Bootstrapping reasoning with reasoning. 2022. URL https://api.semanticscholar.org/CorpusID:247762790.

A CONFIGURATION

Table 4: SelVer-TTT configuration.

Component	Setting	Component	Setting
Gate	3-sample self-consistency entropy + resolvability probe	Thresholds (u^*, ρ^*)	$(0.62,\ 0.55)$
Adaptation rate	38% of the stream	Synthesis	4 pairs per gated query, checkable-structure biased
Verification	answer-only certificate; unverified weight 0	Residual slot	single LoRA, rank 32, effective-matrix projection
Norm budget	global Frobenius $B = 0.35$	Decay	$\gamma = 0.995$ per query
Optimizer	SGD, no cross-projection momentum, LR 10^{-4}	Steps	3 per gated query
Base model	Qwen2.5-7B-Instruct (frozen)	Fallback	bounded base response on operational failure

B BENCHMARKS

Table 5: Evaluation tasks (16-shot for GSM8K, 4-shot CoT for the others; GPQA-Diamond in the main-text multiple-choice protocol).

Benchmark	Metric	Role
MATH-500 / MATH / Minerva-Math	accuracy	medium difficulty
GSM8K	accuracy	easy split
AIME24 / OlympiadBench	accuracy	hardest split
AMC23	accuracy	medium-hard
GPQA-Diamond	accuracy	non-math general
MMLU	accuracy	held-out drift probe

C REPRODUCIBILITY

Each gated query's record stores the gate features, the synthesized pairs with their verification verdicts, the optimizer steps, the pre- and post-update residual norms, and the guard outcome. Session boundaries, abstentions, and fallbacks are logged so that per-query and persistent-mode results can be replayed exactly; the streaming evaluation replays the 500-query order with fixed seeds.