TACIT: INTERNALIZING COLLABORATION VIA TRAINING-TIME EXPERIENCE TRANSFER

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-agent reasoning with large language models (LLMs) has unleashed strong potential for complex tasks. However, existing paradigms, such as multi-round debate, rely heavily on verbose and transient inference-time communication, incurring substantial token overhead and sometimes redundant interactions that undermine performance. Consequently, communication remains a recurring runtime patch rather than internalized competence among agents. To this end, we introduce **TACIT** (**T**raining-time multi-**A**gent **C**ollaboration via **I**mplicit experience **T**ransfer), a framework that enables LLMs to *learn from the exemplary when confronted with the flawed* through two-stage preference optimization. Specifically, LLM-based agents first develop *self-verification* by directly assessing their own answers during generation. This endogenous confidence signal then *gates implicit collaboration* in the second stage: uncertain agents acquire superior peer trajectories as confirmed experience, thereby constructing agent-specific preference data for co-training. At inference, TACIT requires *no explicit communication*, relying only on lightweight confidence-driven routing and answer aggregation. Experiments across diverse reasoning benchmarks show that TACIT elicits reliable and general self-verification for genuine collaboration, surpassing representative baselines in task performance. Instead of test-time indiscriminate dialogue, targeted experience transfer makes TACIT a truly *tacit* paradigm — *collaboration, once learned, need not be spoken*.

1 INTRODUCTION

Large language models (LLMs) have achieved remarkable success across a broad spectrum of real-world applications. A major driver of recent progress is test-time scaling (Snell et al., 2024), which improves performance by allocating additional computation during inference rather than increasing the model capacity. Beyond extending individual reasoning through chain-of-thought prompting (Wei et al., 2022) and repeated sampling with self-consistency (Wang et al., 2023; Brown et al., 2024), recent work scales reasoning along a collaborative dimension. These multi-agent systems enable LLMs to exchange perspectives through debate (Du et al., 2024; Liang et al., 2024), jointly evaluate candidates (Chan et al., 2024), or assume specialized roles in structured workflows (Li et al., 2023; Wu et al., 2023; Hong et al., 2024). By leveraging diverse reasoning trajectories and exposing solutions to external scrutiny, such systems often outperform their individual members.

However, most existing multi-agent systems realize collaboration through explicit communication at inference time. This design brings three recurring limitations. First, communication is *verbose*: agents repeatedly read and respond to peer-generated contexts, increasing token consumption and latency while often retransmitting information that has already been produced (Wang et al., 2024; Smit et al., 2024). Second, more interaction is *not always beneficial*. Unnecessary interaction can overturn initially correct answers, as agents defer to articulate but wrong peers. Empirical studies find that carefully prompted answer ensembling frequently matches elaborate debate protocols at a fraction of their cost (Wang et al., 2024; Smit et al., 2024; Huang et al., 2024). Third, communication is *transient*: any correction remains confined to the current context window, leaving the underlying model unchanged and forcing the same weaknesses to be addressed again on future queries. In effect, indiscriminate inference-time dialogue remains a runtime patch: it repeatedly exchanges superficial *text* without consolidating into reusable *competence*.

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.

Recent efficiency-oriented methods make interaction sparser or cheaper, yet still incur communication on new queries. We therefore argue that *the essence of collaboration lies in efficient, targeted experience transfer*, from those who have already mastered a problem to those who have not, at their own failures. In this way, agents should learn *when to trust*, learning from the exemplary when confronted with the flawed.

Hence, we recast collaboration from a recurring test-time expense into a one-off training investment. Rather than repeatedly exchanging messages at deployment, agents may benefit from internalizing stronger reasoning trajectories precisely where their own solutions are unreliable. This view reveals a key prerequisite: an agent must recognize both its competence and its need for assistance. Self-verification is therefore not merely an auxiliary capability, but the global control signal that determines when cross-agent experience should be acquired.

In this paper, we propose **TACIT** (**T**raining-time multi-**A**gent **C**ollaboration via **I**mplicit experience **T**ransfer), a two-stage preference-learning framework that converts complementary peer experience into competence. TACIT trains a population of agents with direct preference optimization (DPO) (Rafailov et al., 2023). In **Stage I**, each agent learns self-verification by appending a reserved marker, [`confirm`] or [`uncertain`], after its completion, eliciting an endogenous confidence signal that requires neither an external verifier nor an auxiliary confidence head at inference. In **Stage II**, the generative signal gates cross-agent experience transfer. When an agent is uncertain about a query, superior trajectories from peers that solved the problem and confirmed their answers are incorporated into preferred continuations. The preference data are agent-specific: each model evolves primarily on its own failure modes rather than indiscriminately imitating a globally strongest teacher. This targeted construction both concentrates supervision where it is needed and preserves heterogeneous strengths across the population. In effect, peer guidance that would otherwise occur through runtime dialogue is internalized as persistent competence during training.

At inference, TACIT requires no inter-agent message passing. A lightweight router selects the top agents according to competence statistics during training, and the selected agents generate independently without shared context or debate rounds. Each agent emits its verification marker after an initial solution; when uncertain, it can continue within the same generation with a revised answer, reproducing the correction behavior acquired from peer experience. Finally, TACIT aggregates the resulting outputs using confidence-weighted voting.

We evaluate TACIT with Llama-3.2-3B-Instruct, Llama-3.1-8B-Instruct (Grattafiori et al., 2024), and Qwen3-4B (Yang et al., 2025) across mathematical reasoning benchmarks, and further assess whether its learned self-verification generalizes to unseen code and symbolic reasoning tasks. Under matched inference budgets, TACIT outperforms representative test-time scaling baselines, achieving the best accuracy in five of eight benchmark settings and the second-best in the remaining three. Training Stage I improves self-verification performance from chance level to 73.9–91.2% across datasets and backbones, while cross-domain results show that this signal remains reliable and general beyond the training distribution. Overall, TACIT internalizes genuine collaboration into reusable competence without relying on inference-time dialogue.

To summarize, our contributions are threefold: **(1)** We introduce an *internalized* view of multi-agent collaboration, where complementary peer experience is transformed into persistent, intrinsic competence. **(2)** We propose TACIT, a two-stage preference-learning framework that first enables *endogenous self-verification* among agents and then *gates cross-agent correction learning* for co-evolution. **(3)** We demonstrate across multiple backbones and reasoning benchmarks that TACIT learns *reliable* and *general* self-verification while achieving a strong accuracy-efficiency trade-off *without explicit message passing* at inference.

2 Related Work

Test-Time Scaling and Multi-Agent Reasoning. Test-time scaling improves reasoning by allocating more inference compute per query, most commonly through repeated sampling with answer aggregation (Wang et al., 2023; Brown et al., 2024) and compute-optimal inference strategies (Snell et al., 2024). Multi-agent reasoning is its collaborative instantiation: agents read and respond to one another in debate (Du et al., 2024; Liang et al., 2024), evaluate candidates through discussion (Chan et al., 2024), or assume specialized roles within structured workflows (Li et al., 2023; Wu et al., 2023;

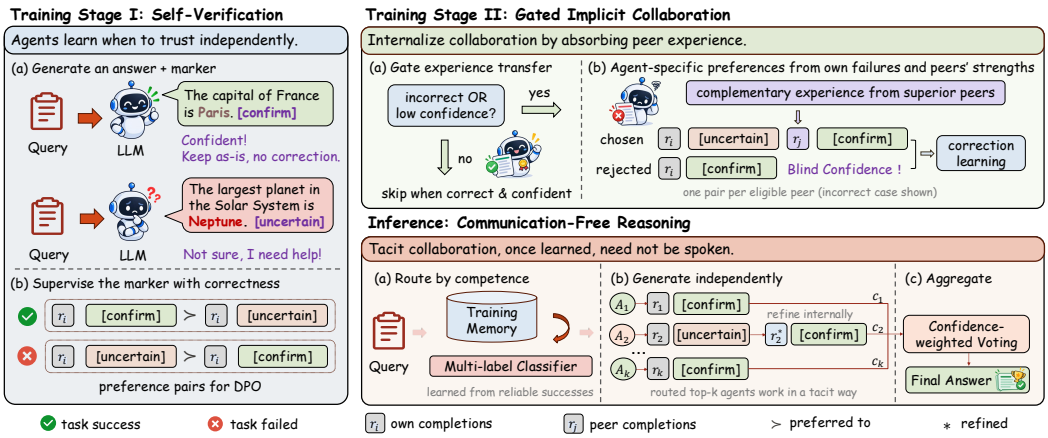


Figure 1: **Overview of TACIT**. In the Training Stage I, each agent learns self-verification via DPO, appending [confirm] or [uncertain] to its own answer. In the Training Stage II, experience transfer is triggered when the target agent is incorrect or insufficiently confident. Uncertain answers are refined by appending a higher-quality peer answer for correction learning, yielding agent-specific preference data on which agents co-evolve. At inference, a learnable router selects the top-k agents and then candidate answers are fused into the final outcome.

Hong et al., 2024; Chen et al., 2024). Subsequent analyses, however, question whether discussion justifies its cost: strong single-agent prompting with answer ensembling often matches elaborate debate protocols, and interaction can overturn initially correct answers (Wang et al., 2024; Smit et al., 2024). Efficiency-oriented follow-ups reduce runtime interaction rather than remove it, for example by sparsifying the communication topology (Li et al., 2024) or pruning redundant messages (Zhang et al., 2025), so the cost of collaboration is still incurred at every query. TACIT differs in that inference-time communication is removed altogether, with the collaborative mechanism relocated to training, where its cost is incurred once rather than repeatedly.

Train-Time Optimization of Multi-Agent Systems. A recent line of work optimizes the agents themselves rather than only their prompts or interaction topology. Multi-agent finetuning specializes a society of models on data generated through their own debates, sustaining self-improvement across more rounds than single-model training, but still deploys multi-round debate at inference (Subramaniam et al., 2025). MALT trains a sequential generator–verifier–refiner pipeline with outcome-driven credit assignment, again requiring a multi-model pipeline at test time (Motwani et al., 2024). MA-PoRL co-trains multiple LLMs with reinforcement learning so that their multi-turn discussions become more corrective and persuasive (Park et al., 2025); Optima iteratively trains agents to communicate more effectively and economically (Chen et al., 2025); and SiriuS fine-tunes agents on an experience library distilled from successful interaction trajectories (Zhao et al., 2025). These methods make inference-time interaction better, cheaper, or more instructive, but interaction remains the vehicle of collaboration at deployment. TACIT differs in kind: peer experience is transferred entirely during training, so no message passing occurs at inference. Moreover, whereas prior training signals are typically trajectory-level and applied uniformly, our approach is *confidence-gated* and *agent-specific*, which repairs individual weaknesses while preserving the population diversity on which ensembling relies.

3 Methodology

3.1 Framework Overview

In this section, we present a **T**raining-time multi-**A**gent **C**ollaboration framework via **I**mplicit experience **T**ransfer (**TACIT**), enabling LLM agents to efficiently internalize peer experience through two-stage preference learning and independently infer without explicit communication. The overview of TACIT is depicted in Fig. 1.

TACIT comprises three core procedures: (i) **Training Stage I: Self-Verification**, in which every agent independently learns to judge its own answer by appending a specified token (i.e. [confirm] or [uncertain]) immediately after initial generation, trained with preference pairs derived from its own success and failure modes. (ii) **Training Stage II: Gated Implicit Collaboration**, in which the resulting marker then decides whether collaboration is warranted. When triggered, every consultant agent would absorb reasoning trajectories from superior peers (ones that solved the problem and confirmed their answers) as complementary experience, thereby preference pairs derived from its own failures and peers' strengths. (iii) **Inference: Communication-Free Reasoning**, in which a lightweight router dispatches each query to the most competent agents based on training memory. For each query, the selected agent iteratively refines the generation from an initial solution if followed by a [uncertain] marker, reproducing the correction procedure that previously required peer assistance. Finally, TACIT aggregates all confirmed answers through confidence-weighted voting. We elaborate on each procedure in the following subsections.

3.2 Stage I: Self-Verification

Choice of Verification Markers. A subtle but consequential design decision concerns the realization of verification markers. We instantiate [confirm] and [uncertain] using two reserved single-token IDs already present in the backbone tokenizer, rather than extending the vocabulary or repurposing semantically loaded tokens such as beginning or end-of-sequence markers. Adding new tokens would introduce randomly initialized embeddings and output-head parameters, perturb the pretrained distribution.

Confidence Formulation. Consider a multi-agent system,

$$\mathscr{A} = \{A_i\}_{i=1}^{N}, \tag{1}$$

where N is the population size, the agent A_i is parameterized by a policy π_{θ_i}. Given a query x, each agent generates a reasoning trajectory r_i (including its final answer y), followed by a verification marker $v_i \in \{[\texttt{confirm}], [\texttt{uncertain}]\}$. The marker [confirm] indicates that the agent considers its generation reliable, whereas [uncertain] indicates insufficient confidence. We emphasize that [confirm] expresses an agent's endorsement of its own answer rather than an instruction to halt generation. Generation always terminates with the tokenizer's native end-of-sequence token.

Rather than introducing an additional confidence head, TACIT derives confidence directly from the probabilities assigned to these two markers. For brevity, we denote by $p_i(v) = \pi_{\theta_i}(v|x, r_i)$ the probability that agent A_i emits marker v after reasoning trajectory r_i. Then the normalized confidence score is defined as

$$c_i(x, r_i) = \frac{p_i([\texttt{confirm}])}{p_i([\texttt{confirm}]) + p_i([\texttt{uncertain}])}. \tag{2}$$

However, both semantically vacuous reserved markers receive very little probability mass under the softmax over the full vocabulary, which is dominated by tokens the backbone was pretrained to produce. The absolute probabilities are therefore compressed into a narrow range, so thresholding on them is sensitive to numerical resolution and to the shifting scale of the remaining vocabulary rather than to the agent's actual preference. Instead, the *logit gap* sidesteps this issue by discarding the shared normalizer altogether: it measures the relative preference between the two markers directly, and consequently yields a markedly sharper ordering over questions. Hence, we adopt the underlying *logit gap* to be the more reliable quantity in practice,

$$g_i(x, r_i) = lg_i^{[\texttt{confirm}]}(x, r_i) - lg_i^{[\texttt{uncertain}]}(x, r_i) \tag{3}$$

where $lg_i^v(x, r_i)$ denotes the logit that agent A_i assigns to marker $v \in \{[\texttt{confirm}], [\texttt{uncertain}]\}$ at the position immediately following the trajectory r_i, i.e. the pre-softmax score produced by the language modeling head for that token in the next-token distribution conditioned on (x, r_i). Since preference learning is fundamentally a ranking decision, we report g_i as the main metric for evaluation, while treating confidence score c_i as its bounded counterpart for probabilistic analysis.

Preference Optimization. To instill this self-verification behavior, we adopt direct preference optimization (Rafailov et al., 2023), which aligns a policy with pairwise preferences without learning

an explicit reward model. Given a query x together with a preferred output y^+ and a rejected output y^-, the objective of agent A_i is

$$\mathscr{L}^i_{\text{DPO}} = -\mathbb{E}_{(x,y^+,y^-)\sim\mathscr{D}_i}\left[\log\sigma\Big(\beta\log\frac{\pi_{\theta_i}(y^+\mid x)}{\pi_{\text{ref},i}(y^+\mid x)} - \beta\log\frac{\pi_{\theta_i}(y^-\mid x)}{\pi_{\text{ref},i}(y^-\mid x)}\Big)\right], \tag{4}$$

where $\pi_{\text{ref},i}$ denotes the reference policy of agent A_i, β controls the strength of the preference signal, and $\sigma(\cdot)$ is the sigmoid function. Both training stages share this objective and differ only in the construction of $\mathscr{D}_i$, which we detail below.

Let $R(x, r_i) \in \{0, 1\}$ denote the correctness of trajectory r_i against the ground truth, where $R(x, r_i) = 1$ if the final answer is correct and 0 otherwise. Preference pairs are constructed accordingly: when the final answer is correct, the sequence ending in [`confirm`] is preferred over the one in [`uncertain`], and vice versa. Formally, for each sampled trajectory r_i we construct a preference pair (see Eq. 5), in which two candidates share an identical trajectory and differ only in the specified marker, producing a learned and interpretable confidence signal.

$$y^+_{i,\text{sv}} = r_i \oplus v^\star_i, \qquad y^-_{i,\text{sv}} = r_i \oplus \overline{v^\star_i}, \tag{5}$$

where $\oplus$ denotes sequence concatenation, $v^\star_i = $ [`confirm`] if $R(x, r_i) = 1$ and [`uncertain`] otherwise, and $\overline{v^\star_i}$ is the alternative marker.

We assume access to a training set $\mathscr{X} = \{x^{(m)}\}_{m=1}^M$ of queries, each paired with a ground-truth answer used solely to compute $R(\cdot)$ during training and never available at inference. Then Agent A_i is optimized over the self-verification dataset $\mathscr{D}^{\text{sv}}_i$ towards the following preference objective. All N agents are trained independently.

$$\mathscr{D}^{\text{sv}}_i = \big\{\big(x, y^+_{i,\text{sv}}, y^-_{i,\text{sv}}\big) : x \in \mathscr{X}\big\}, \tag{6}$$

$$\theta^{(1)}_i = \arg\min_{\theta_i} \mathscr{L}^i_{\text{DPO}}\big(\mathscr{D}^{\text{sv}}_i\big), \qquad i = 1, \ldots, N, \tag{7}$$

3.3 Stage II: Gated Implicit Collaboration

After self-verification, TACIT relies on the learned markers to decide when cross-agent experience transfer is warranted.

Gating Condition. Two conditions must hold jointly for an agent to be exempted from collaboration on a question: it must answer correctly, and it must be genuinely confident in doing so. We set the agent-specific threshold τ_i to the mean logit gap of A_i over the training set:

$$\tau_i = \frac{1}{|\mathscr{X}|}\sum_{x\in\mathscr{X}} g_i(x, r_i). \tag{8}$$

so that confidence is judged relative to each agent's own scale rather than by a shared constant. Collaboration is then triggered whenever either condition fails,

$$\text{gate}_i(x) = \begin{cases} 0, & g_i(x, r_i) > \tau_i \wedge R(x, r_i) = 1, \\ 1, & \text{otherwise}, \end{cases} \tag{9}$$

Intuitively, an agent that is right but hesitant still lacks consolidated competence, and an agent that is wrong yet assured is precisely where correction is most needed, so both cases warrant experience transfer; only the conjunction of correctness and confidence indicates that no collaboration is needed.

Peer Experience Transfer. Once the gate fires, we collect the superior peers whose trajectories are both correct and self-confirmed,

$$\mathscr{P}_i(x) = \big\{j \neq i \;\big|\; R(x, r_j) = 1,\ v_j = [\texttt{confirm}]\big\}, \tag{10}$$

and construct one preference pair per qualifying peer rather than selecting a single teacher. Taking the query x as the shared prompt, the two candidate responses are

$$\begin{aligned} y^+_{i,\text{gc}} &= r_i \oplus [\texttt{uncertain}] \oplus r_j \oplus [\texttt{confirm}], \\ y^-_{i,\text{gc}} &= r_i \oplus [\texttt{confirm}], \end{aligned} \tag{11}$$

and the gated-collaboration dataset for agent A_i is

$$\mathscr{D}_i^{\mathrm{gc}} = \left\{ \left(x, y_{i,\mathrm{gc}}^{+}, y_{i,\mathrm{gc}}^{-}\right) : \mathrm{gate}_i(x) = 1,\ j \in \mathscr{P}_i(x) \right\}. \tag{12}$$

The two candidates share the agent's own attempt r_i and diverge at the marker that follows it. The preferred response declares uncertainty, continues along a peer's verified route, and only then confirms; the rejected response instead confirms the flawed attempt on the spot. A single pair thus conveys two coupled lessons: an unreliable answer must not be endorsed, and declaring uncertainty is the entry point to a better continuation rather than a terminal admission of failure.

The gate also fires when an agent solves a question yet reports low confidence, in which case penalizing $r_i \oplus [\texttt{confirm}]$ would be counterproductive, as it would discourage an endorsement the agent is entitled to make. We therefore set the rejected response to $y_{i,\mathrm{gc}}^{-} = r_i \oplus [\texttt{uncertain}]$ for these instances, so that the preference contrasts a hesitant reading of a sound solution against a confident one. The rationale is that a correct answer produced without confidence often reflects a fortunate sample rather than a mastered solution mode.

Retaining every peer in $\mathscr{P}_i(x)$ exposes A_i to multiple distinct routes toward the same correct solution, which enriches the supervision and prevents the transfer from collapsing onto whichever agent happens to be strongest. Three properties distinguish this construction from indiscriminate distillation. It is *gated*, since transfer is activated by the target agent's own uncertainty rather than imposed on every example. It is *reciprocal*, since the source of supervision varies with the agents' relative expertise on each question, so that an agent teaches on one question and learns on another. And it is *agent-specific*, since each policy is optimized on a dataset indexed by its own failures, addressing individual weaknesses instead of driving the population toward a single homogeneous model.

3.4 Inference: Communication-Free Reasoning

At inference, TACIT requires no explicit communication, relying only on lightweight routing mechanism and answer aggregation.

Confidence-Driven Routing. Dispatching every query to all N agents wastes computation on those unlikely to contribute. We therefore populate a memory pool during training, with pairs $\langle \mathrm{embed}(x), i \rangle$, recorded only when A_i both solves x and confirms it, so that the pool encodes reliable rather than accidental competence. A frozen sentence encoder followed by a two-layer MLP maps a query to N routing scores $s(x) = (s_1(x), \ldots, s_N(x))$, where $s_i(x)$ is the predicted competence of agent A_i on x, supervised by per-agent binary cross-entropy against the label $m_i = \mathbb{I}\left[R(x, r_i) = 1 \wedge v_i = [\texttt{confirm}]\right]$, where $\mathbb{I}[\cdot]$ is the indicator function, so that $m_i = 1$ only when A_i both solves x and confirms its answer. We formulate routing as multi-label rather than multi-class classification, since several agents may be competent on the same query and should not compete for a shared probability mass. More fundamentally, selecting agents on the basis of what they have demonstrably solved amounts to a learned belief about where each agent's competence lies, so that the distributor complements the endogenous confidence of individual agents with a population-level view of who should be trusted on what.

$$\mathscr{S}_k(x) = \text{top-}k_{\,i \in [N]}\ s_i(x), \tag{13}$$

Answer Aggregation. The selected top-k agents (Eq. 13) generate a reasoning trajectory with a self-verification marker independently. If the emitted marker is $[\texttt{uncertain}]$, the agent continues within the same pass with a revised solution, and the generation terminates once it emits $[\texttt{confirm}]$ or a budget of revisions is reached. With agents reproducing internally the corrective continuation that peers once provided, the final answer is resolved by confidence-weighted voting.

$$\hat{a}(x) = \arg\max_{a} \sum_{i \in \mathscr{S}_k(x)} c_i(x, r_i)\, \mathbb{I}[a_i = a], \tag{14}$$

The weighting matters precisely as self-verification training renders c_i trustworthy, allowing a confident minority to outweigh a hesitant majority.

Table 1: Accuracy (%) on mathematical reasoning benchmarks with Llama backbones. The best and second-best scores are highlighted in *red* and *blue*, respectively.

Method	Llama-3.2-3B-Instruct		Llama-3.1-8B-Instruct	
	GSM8K	MATH-500	GSM8K	MATH-500
single-agent inference				
Chain-of-Thought	80.89 ± 0.16	35.20 ± 0.37	86.51 ± 0.26	45.20 ± 0.41
Self-Consistency	83.93 ± 0.25	36.48 ± 0.31	89.57 ± 0.37	47.34 ± 0.22
inference-time collaboration				
Multi-Agent Debate	88.71 ± 0.22	43.80 ± 0.36	90.15 ± 0.38	**53.64 ± 0.47**
iMAD	92.26 ± 0.27	44.12 ± 0.39	91.78 ± 0.43	51.76 ± 0.52
training-time optimization				
ReVISE	90.87 ± 0.13	36.69 ± 0.35	92.05 ± 0.32	49.32 ± 0.14
S2R	89.36 ± 0.29	37.25 ± 0.38	91.08 ± 0.25	49.83 ± 0.22
TACIT (Ours)	**93.47 ± 0.16**	**46.30 ± 0.22**	**94.78 ± 0.28**	53.08 ± 0.12

Table 2: Accuracy (%) on mathematical reasoning benchmarks with Qwen3-4B. The best and second-best scores are highlighted in *red* and *blue*, respectively.

Method	GSM8K	MATH-500	AIME 2024	AIME 2025
single-agent inference				
Chain-of-Thought	90.75 ± 0.19	52.40 ± 0.35	21.35 ± 0.57	17.14 ± 0.62
Self-Consistency	91.78 ± 0.24	56.05 ± 0.29	21.92 ± 0.56	18.26 ± 0.49
inference-time collaboration				
Multi-Agent Debate	93.27 ± 0.31	64.72 ± 0.26	**24.03 ± 0.47**	20.27 ± 0.51
iMAD	94.18 ± 0.24	65.31 ± 0.38	22.99 ± 0.61	20.46 ± 0.57
training-time optimization				
ReVISE	92.28 ± 0.21	57.98 ± 0.17	21.57 ± 0.42	21.05 ± 0.38
S2R	93.07 ± 0.23	63.68 ± 0.37	22.02 ± 0.56	**22.12 ± 0.67**
TACIT (Ours)	**96.61 ± 0.17**	**68.28 ± 0.27**	23.88 ± 0.31	21.88 ± 0.42

4 Evaluation

4.1 Experiment Setup

Base Models. We evaluate TACIT with three open-source instruction-tuned language models of different scales and families: Llama-3.2-3B-Instruct, Llama-3.1-8B-Instruct (Grattafiori et al., 2024), and Qwen3-4B (Yang et al., 2025). Due to computational constraints, we fine-tune quantized backbones with QLoRA (Dettmers et al., 2023), which also allows the agent population to be instantiated as lightweight adapters over a shared base model.

Datasets. To implement two-stage optimization, we construct preference pairs from the training splits of three mathematical reasoning datasets: GSM8K (Cobbe et al., 2021), competition-level MATH (Hendrycks et al., 2021), and DAPO-Math-17K (Yu et al., 2026). For in-domain evaluation, we report results on the GSM8K test split and MATH-500 (Lightman et al., 2024), a 500-problem subset of the MATH test split. We additionally evaluate on AIME 2024 (Mathematical Association of America, 2024) and AIME 2025 (Mathematical Association of America, 2025), which have substantially greater difficulty beyond the training distribution. To assess whether self-verification generalizes beyond the mathematical domain on which it is trained, we further evaluate on MBPP (Austin et al., 2021) for code generation and BBH (Suzgun et al., 2023) for diverse logical and symbolic reasoning, both unseen during training.

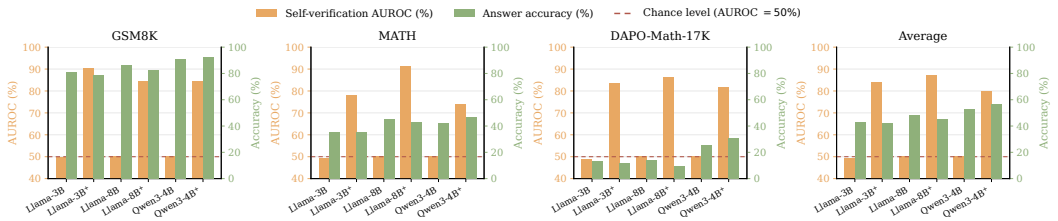


Figure 2: **Self-verification performance across backbones.** Orange bars show the self-verification AUROC of the confidence logit gap, while green bars report answer accuracy. Models marked with + are evaluated after Stage I; untrained backbones remain near the 50% chance level, whereas Stage I substantially improves self-verification across all datasets and backbones.

Baselines. We compare TACIT against three families of methods. (i) *Single-agent inference*: chain-of-thought prompting (Wei et al., 2022) and self-consistency (Wang et al., 2023), which scale test-time compute without any collaboration. (ii) *Inference-time collaboration*: multi-agent debate (Du et al., 2024), in which every agent reads and responds to all peers each round, and iMAD (Fan et al., 2026), which invokes debate selectively and therefore represents the closest inference-time baseline of our gating mechanism. (iii) *Training-time self-verification and self-correction*: ReVISE (Lee et al., 2025), which learns an intrinsic indicator to decide whether to refine, and S2R (Ma et al., 2025), which trains models to self-verify and self-correct through reinforcement learning; both improve an individual agent in isolation, disentangling the contribution of cross-agent experience transfer in TACIT.

Metrics. Answer accuracy is our primary metric. We report Pass@1 on all benchmarks to compare TACIT against test-time scaling baselines under a matched sampling budget; for TACIT, votes are aggregated using confidence-based weights as in Eq. 14 rather than counted uniformly. Given the small size of the AIME sets, we sample 32 generations per problem and report the average, whereas BBH is decoded greedily with a single generation and scored by macro-average accuracy over its constituent tasks, so that no sub-task dominates the aggregate. Beyond accuracy, we assess self-verification quality by the AUROC of the logit gap g_i (Eq. 22) in predicting whether a generated answer is correct, which measures the verification signal's discriminative power.

Implementation Details. Preference data for both stages are built from $n = 10$ trajectories per question sampled at temperature 0.7, which supplies the diversity needed to expose each agent's failures and populate the peer pool. At evaluation, single-generation results are decoded greedily and multi-sample protocols use temperature = 0.7 and top-$p = 0.9$; token budgets are matched across all methods, so that in-pass revision after an `[uncertain]` marker enjoys no additional allowance. All remaining experimental settings are provided in the appendix.

4.2 Main Results

We train each of the three backbones separately on GSM8K, DAPO-Math-17K, and Hendrycks-MATH, and report the resulting accuracy on four evaluation benchmarks in Table 1 and Table 2, without additional fine-tuning. As can be seen, TACIT attains the best accuracy in five of the eight columns and the second best in the remaining three, averaging 62.28% against 60.36% for the strongest baseline. We highlight three findings as follows. **First**, the ordering of the three families is consistent across backbones: single-agent inference is weakest (53.68% and 55.67%), training-time self-verification improves on it (57.73% and 58.55%), and inference-time collaboration is stronger still (59.82% and 60.36%).

We next examine the self-verification mechanism, as it determines when collaboration is triggered. Figure 2 shows that this signal is learned rather than inherited. Every untrained backbone sits at chance (49.0–50.1% AUROC), confirming that the reserved marker tokens carry no informative prior, while a single stage of preference optimization lifts verification to 73.9–91.2% across all datasets and backbones. Answer accuracy meanwhile shifts by only a few points in either direction as preference optimization perturbs the generation distribution, a cost Stage II recovers and surpasses. Figure 3 shows that the signal also survives the domain shift: the verification stays above chance on MBPP and

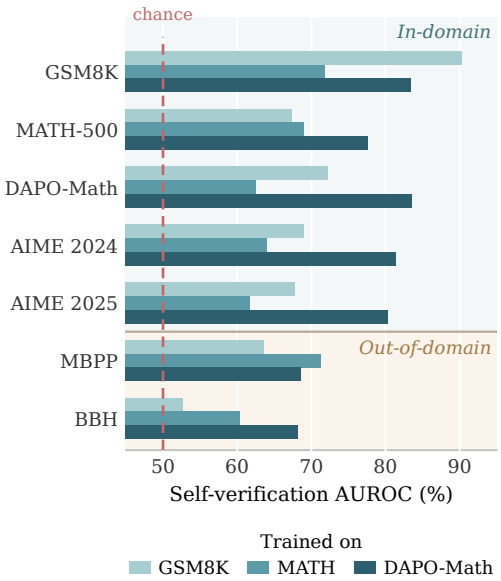


Figure 3: **Cross-domain generalization of self-verification.** Bars report AUROC of the logit gap g_i (Eq. 22) in predicting answer correctness.

Table 3: Factorial ablation on MATH-500 with Qwen3-4B.

	w/o Stage II	w/ Stage II
w/o Stage I	52.40	63.00
w/ Stage I	57.00	**68.28**

BBH, neither seen during training, so agents acquire a general awareness of their own competence rather than memorizing which training items they can solve. Its reach depends on the breadth of the training distribution: DAPO-Math keeps AUROC above 68% on all seven evaluation sets, whereas GSM8K alone verifies almost perfectly in domain (90.3%) yet falls to chance on BBH (52.7%).

4.3 Ablation Studies

We ablate the two training stages factorially on MATH-500 with Qwen3-4B (see Table 3). Removing Stage II withholds peer experience entirely, leaving agents that verify their own answers but never learn from one another. Removing Stage I is subtler and is the comparison we care about most: peer trajectories are still grafted, and on the same number of questions, but the confidence signal that selects those questions no longer exists, so the choice reduces to a random gate. This variant therefore holds the *amount* of transferred experience fixed and varies only *where* it is applied, isolating the contribution of gating from that of additional supervision.

5 Conclusion

We introduced TACIT, a training-time framework for internalizing multi-agent collaboration through targeted experience transfer. Discarding costly and indiscriminate exchanging messages at runtime, TACIT first equips each agent with self-verification and then identify where peer experience should be acquired for correction learning. This endogenous signal is what makes collaboration deployable: at inference, many problems admit no ground truth and thus no external arbiter of correctness, leaving an agent's own assessment valuable. Also, our results suggest a broader perspective: effective collaboration not only lies in interaction, but also in selectively routing experience among agents with complementary competencies. We hope TACIT offers a truly tacit paradigm for multi-agent reasoning: collaboration, once learned, need not be spoken.

AI USE STATEMENT

We used generative AI tools to revise wording for clarity and consistency of terminology. We did not use generative AI tools to generate experimental results, figures, or numerical claims. All AI-assisted text was reviewed by the authors, who take responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

We provide detailed implementation information, including training configuration, hyperparameters, as well as experimental setups such as base models, datasets and evaluation metrics in Section 4.1 and Appendix A. The executable code of TACIT is available in supplementary materials.

REFERENCES

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. ChatEval: Towards better LLM-based evaluators through multi-agent debate. In *International Conference on Learning Representations*, 2024.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, et al. AgentVerse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *International Conference on Learning Representations*, 2024.

Weize Chen, Jiarui Yuan, Chen Qian, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Optima: Optimizing effectiveness and efficiency for llm-based multi-agent system. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 11534–11557, 2025.

Karl Cobbe, Vineet Kosaraju, Mohammad 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 preprint arXiv:2110.14168*, 2021.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115, 2023.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *International Conference on Machine Learning*, 2024.

Wei Fan, JinYi Yoon, and Bo Ji. imad: intelligent multi-agent debate for efficient and accurate llm inference. In *Proceedings of the Fortieth AAAI Conference on Artificial Intelligence*, pp. 29403–29411. AAAI Press, 2026. ISBN 978-1-57735-906-7.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2021.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, et al. MetaGPT: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, 2024.

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. In *International Conference on Learning Representations*, 2024.

Hyunseok Lee, Seunghyuk Oh, Jaehyung Kim, Jinwoo Shin, and Jihoon Tack. Revise: Learning to refine at test-time via intrinsic self-verification. In *International Conference on Machine Learning*, pp. 33616–33634. PMLR, 2025.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. CAMEL: Communicative agents for "mind" exploration of large language model society. In *Advances in Neural Information Processing Systems*, 2023.

Yunxuan Li, Yibing Du, Jiageng Zhang, Le Hou, Peter Grabowski, Yeqing Li, and Eugene Ie. Improving multi-agent debate with sparse communication topology. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 7281–7294, 2024.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. Encouraging divergent thinking in large language models through multi-agent debate. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Ruotian Ma, Peisong Wang, Cheng Liu, Xingyan Liu, Jiaqi Chen, Bang Zhang, Xin Zhou, Nan Du, and Jia Li. S2r: Teaching llms to self-verify and self-correct via reinforcement learning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 22632–22654, 2025.

Mathematical Association of America. 2024 american invitational mathematics examination (AIME I and AIME II). `https://maa.org/maa-invitational-competitions/`, 2024. MAA American Mathematics Competitions.

Mathematical Association of America. 2025 american invitational mathematics examination (AIME I and AIME II). `https://maa.org/maa-invitational-competitions/`, 2025. MAA American Mathematics Competitions.

Sumeet Ramesh Motwani, Chandler Smith, Rocktim Jyoti Das, Rafael Rafailov, Ivan Laptev, Philip HS Torr, Fabio Pizzati, Ronald Clark, and Christian Schroeder de Witt. Malt: Improving reasoning with multi-agent llm training. *arXiv preprint arXiv:2412.01928*, 2024.

Chanwoo Park, Seungju Han, Xingzhi Guo, Asuman E. Ozdaglar, Kaiqing Zhang, and Joo-Kyung Kim. MAPoRL: Multi-agent post-co-training for collaborative large language models with reinforcement learning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pp. 30215–30248, 2025.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, 2023.

Andries P. Smit, Paul Duckworth, Nathan Grinsztajn, Thomas D. Barrett, and Arnu Pretorius. Should we be going MAD? a look at multi-agent debate strategies for LLMs. In *International Conference on Machine Learning*, 2024.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Vighnesh Subramaniam, Yilun Du, Joshua B Tenenbaum, Antonio Torralba, Shuang Li, and Igor Mordatch. Multiagent finetuning: Self improvement with diverse reasoning chains. In *International Conference on Learning Representations*, volume 2025, pp. 10840–10862, 2025.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed H Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13003–13051, 2023.

Qineng Wang, Zihao Wang, Ying Su, Hanghang Tong, and Yangqiu Song. Rethinking the bounds of LLM reasoning: Are multi-agent discussions the key? In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024.

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

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38:113222–113244, 2026.

Guibin Zhang, Yanwei Yue, Zhixun Li, Sukwon Yun, Guancheng Wan, Kun Wang, Dawei Cheng, Jeffrey Yu, and Tianlong Chen. Cut the crap: An economical communication pipeline for llm-based multi-agent systems. In *International Conference on Learning Representations*, volume 2025, pp. 75389–75428, 2025.

Wanjia Zhao, Mert Yuksekgonul, Shirley Wu, and James Zou. SiriuS: Self-improving multi-agent systems via bootstrapped reasoning. In *Advances in Neural Information Processing Systems*, 2025.

A Implementation Details

A.1 Dataset Description

We provide detailed descriptions of all datasets used for training and evaluation. All datasets are used in their official releases without additional filtering unless otherwise stated; statistics are summarized in Table 4.

Training Data. Preference pairs for two-stage optimization are constructed from the training splits of the following three mathematical reasoning datasets.

- **GSM8K** consists of 8.5K grade-school math word problems written by human annotators, split into 7,473 training and 1,319 test problems. Each problem requires two to eight steps of elementary arithmetic and is paired with a natural-language solution terminating in a single numeric answer. We use the training split as a source of *low-to-moderate difficulty* prompts.
- **MATH** contains 12,500 competition problems (7,500 training / 5,000 test), spanning seven subjects and five difficulty levels. Solutions are given as LaTeX-formatted step-by-step derivations with answers enclosed in `\boxed{}`. This dataset provides the primary source of *competition-level* training prompts.
- **DAPO-Math-17K** is a reinforcement-learning-oriented corpus of approximately 17K mathematical problems collected from online competition archives. All answers are normalized to integer form, which makes reward computation deterministic and eliminates the parsing ambiguity of free-form answer matching. We include it to increase prompt diversity beyond the GSM8K and MATH distributions.

Dataset	Domain	Split	#Problems	Answer Format	Metric
Training (preference-pair construction)					
GSM8K	Grade-school math	train	7,473	Integer	—
MATH	Competition math	train	7,500	`\boxed{}` expr.	—
DAPO-Math-17K	Competition math	train	17,398	Integer (normalized)	—
In-domain evaluation					
GSM8K	Grade-school math	test	1,319	Integer	EM Acc.
MATH-500	Competition math	subset of test	500	`\boxed{}` expr.	EM Acc.
AIME 2024	Olympiad math	full	30	Integer $\in [0, 999]$	avg@32
AIME 2025	Olympiad math	full	30	Integer $\in [0, 999]$	avg@32
Out-of-domain evaluation (unseen during training)					
MBPP	Code generation	test	500	Python function	Pass@1
BBH	Logical / symbolic	full	6,511	Free-form (23 tasks)	EM Acc.

Table 4: Statistics of all datasets used for training and evaluation. Training datasets are used only as prompt sources for preference-pair construction and are therefore not associated with an evaluation metric. EM Acc. denotes exact-match accuracy on the final answer. For AIME, we sample 32 responses per problem and report the mean accuracy (avg@32) to reduce variance on the small problem set.

In-Domain Evaluation. We evaluate on held-out benchmarks drawn from the same mathematical domain as training, ordered by increasing difficulty.

- **GSM8K (test)** comprises the 1,319 held-out problems of GSM8K and measures reliability on elementary multi-step arithmetic. We report exact-match accuracy on the final answer.
- **MATH-500** is a 500-problem subset of the MATH test split, sampled to preserve the subject and difficulty distribution of the full split. We adopt it instead of the full 5,000-problem split to keep evaluation cost tractable while remaining comparable to prior work.
- **AIME 2024** and **AIME 2025** each consist of the 30 problems from the corresponding year's American Invitational Mathematics Examination, with integer answers in $[0, 999]$. These problems lie substantially beyond the difficulty of the training distribution and serve as a stress test of generalization. Because each benchmark is small, we sample 32 responses per problem and report `avg@32` to reduce evaluation variance.

Out-of-Domain Evaluation. To assess whether the learned self-verification behavior transfers beyond mathematics, we evaluate on two benchmarks from domains unseen during training.

- **MBPP** contains 974 crowd-sourced Python programming problems, each specified by a short natural-language description and validated by three automated test cases. We evaluate on the standard test split and report Pass@1, where a generated program is correct only if it passes all associated test cases.
- **BBH** aggregates 23 challenging tasks from BIG-Bench on which language models had not surpassed the average human rater, totaling 6,511 examples spanning logical deduction, symbolic manipulation, and multi-step algorithmic reasoning. We report exact-match accuracy averaged uniformly over the constituent tasks.

A.2 Experimental Configuration

Implementation. We implement TACIT on four NVIDIA H200 GPUs, using vLLM for preference-data generation and distributed QLoRA training with Accelerate at both optimization stages. The frozen backbone is quantized to 4-bit NF4 with double quantization, and the trainable adapters are optimized in BF16 precision. Unless otherwise specified, all experiments disable the native thinking mode of the backbone and apply its original chat template. We repeat the complete two-stage pipeline separately for GSM8K, MATH, and DAPO-Math-17K; samples from different training corpora are never mixed.

Parameter	Stage I	Stage II
Training completions per question	10	10
Validation completions per question	3	3
Maximum new tokens	1,024	2,048
Temperature	0.7	0.7
Top-p	0.9	0.9
Generation seed	42	888
Bootstrap seed for A_i	$1000+i$	$2000+i$
Maximum context length	4,096	4,096
Returned log probabilities	20	20
Confidence suffix length	8	8
Thinking mode	Disabled	Disabled

Table 5: Preference Data Generation Parameters.

Preference Data Generation. For both stages, we use all available samples in the corresponding training and validation splits. Each agent generates 10 trajectories per training question and 3 trajectories per validation question using nucleus sampling with temperature 0.7 and $p = 0.9$. Stage I allows at most 1,024 newly generated tokens, whereas Stage II allows 2,048, with a maximum model context length of 4,096. We retain incomplete generations rather than discarding them, since such trajectories can still provide useful self-verification supervision. The full set of generation parameters is listed in Table 5.

Agent-specific datasets are constructed through sampling with replacement. Stage I bootstraps source questions using seed $1000+i$ for agent A_i, whereas Stage II first constructs cross-agent preference pairs on a shared question pool and then bootstraps the resulting question groups using seed $2000+i$. In both stages, the number of bootstrap draws equals the number of source questions. Candidate-generation seeds are set to 42 and 888 for Stages I and II, respectively.

Preference Optimization. We employ QLoRA for both optimization stages (Table 6). The frozen backbone weights are quantized to 4-bit NF4 with double quantization, while the trainable LoRA adapters and intermediate computations use BF16 precision. We set the LoRA rank to 16, the scaling factor to 32, and the dropout rate to 0.05. Adapters are applied to `q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, `down_proj`, and `lm_head`. Including `lm_head` is necessary for directly adapting the output probabilities of the two confidence markers. LoRA biases are disabled.

Both stages use the sigmoid DPO objective with $\beta = 0.1$, augmented with an RPO-style negative-log-likelihood term on the preferred response with weight $\alpha_{\text{RPO}} = 0.1$. We optimize the adapters using AdamW with a learning rate of 3×10^{-5}, a linear learning-rate schedule, and a warmup ratio of 0.1. Training uses Flash Attention 2, gradient checkpointing, and a maximum gradient norm of 1.0.

Stage I is trained for three epochs with a maximum sequence length of 2,048. Its per-device batch size is 2, with eight gradient-accumulation steps. Stage II is trained for five epochs with a maximum sequence length of 4,096, a per-device batch size of 1, and sixteen accumulation steps. The agents are assigned different optimization and data-order seeds. In the current configuration, the Stage I optimization seeds are $\{101,\ldots,105\}$, with data seeds $\{1001,\ldots,1005\}$; the corresponding Stage II seeds are $\{201,\ldots,205\}$ and $\{2001,\ldots,2005\}$. The Stage II policy and its frozen reference policy are initialized from the same Stage I adapter, with the former remaining trainable and the latter fixed throughout optimization.

Cross-Agent Correction and Routing. For Stage II data construction, an agent-specific confidence threshold is estimated as the mean confidence logit gap over its training trajectories. Collaboration is skipped only when the target answer is correct and its confidence gap exceeds this threshold. The minimum required quality improvement is set to zero, and non-collaborative self-verification examples are retained to preserve the capability acquired in Stage I.

The Router uses a Llama-3.2-1B encoder to obtain normalized, mean-pooled question embeddings. A memory is admitted only when the corresponding answer is correct, its normalized confidence is at least 0.7, and its confidence logit gap is non-negative. We train a linear multiclass classifier on

Parameter	Stage I	Stage II
Training epochs	3	5
Learning rate	3×10^{-5}	3×10^{-5}
DPO β	0.1	0.1
RPO coefficient	0.1	0.1
Warmup ratio	0.1	0.1
Maximum sequence length	2,048	4,096
Per-device batch size	2	1
Gradient accumulation	8	16
Number of processes	2	2
Effective batch size	32	32
LoRA rank / scaling	16/32	16/32
LoRA dropout	0.05	0.05
Backbone precision	4-bit NF4	4-bit NF4
Adapter precision	BF16	BF16
Double quantization	Yes	Yes
Gradient checkpointing	Yes	Yes
Attention implementation	Flash Attention 2	Flash Attention 2

Table 6: Optimization Hyperparameters for Training Stages.

Benchmark	Temp.	Top-p	Max tokens	Samples
GSM8K	0.7	0.9	2,048	8
MATH-500	0.7	0.9	2,048	8
AIME 2024	0.7	0.9	6,144	32
AIME 2025	0.7	0.9	6,144	32
MBPP	Greedy	–	2,048	1
BBH	Greedy	–	2,048	1

Table 7: Task Evaluation Configurations.

these memories for 200 epochs using AdamW, with learning rate 0.05, weight decay 10^{-4}, batch size 32, maximum input length 512, and random seed 66. Inverse-frequency class weights are applied to account for unequal numbers of memories across agents.

System-Level Inference. At inference, each question is routed to the top-3 agents from five trained agents. Each selected agent independently generates one response with temperature 0.2, top-$p = 0.9$, and at most 2,048 new tokens. No generated trajectory is communicated to another agent. Responses producing the same normalized answer are grouped, and each agent's vote is weighted by

$$w_i = \max(0.05, c_i), \tag{15}$$

where c_i is its generated confidence. Routing probabilities are not reused as aggregation weights. The answer group with the largest total confidence weight is selected, with the highest-confidence response in the group used as its representative output.

Task Evaluation. We perform zero-shot chain-of-thought evaluation using the backbone's chat template. GSM8K and MATH-500 use a maximum generation length of 2,048, both pass@1 and majority-vote (self-consistency) accuracy over eight trajectories are recorded. AIME 2024 and AIME 2025 use the same sampling parameters with a maximum of 6144 new tokens and 32 trajectories per problem. MBPP and BBH are evaluated greedily with one trajectory and a maximum length of 2,048. We report execution-based pass@1 for MBPP and macro-averaged accuracy across BBH subtasks. The full configuration is shown in Table 7.

A.3 Two-Stage Preference Learning

TACIT follows a strictly sequential optimization procedure in which agents first learn to verify their own solutions and then learn to correct their characteristic failures by internalizing complementary

peer experience. For each agent A_i, the two stages can be summarized as

$$\pi_i^{(0)} \xrightarrow[\mathscr{D}_{i,\mathrm{sv}}]{\mathrm{DPO}} \pi_i^{(1)} \xrightarrow[\mathscr{D}_{i,\mathrm{cc}}]{\mathrm{DPO}} \pi_i^{(2)}, \tag{16}$$

where $\pi_i^{(0)}$ is the initial instruction-tuned policy, $\pi_i^{(1)}$ is the self-verifying policy obtained after Stage I, and $\pi_i^{(2)}$ is the final policy after cross-agent correction. Importantly, Stage II is initialized from $\pi_i^{(1)}$, rather than restarting from the original backbone. The self-verification capability acquired in Stage I is therefore preserved and subsequently used to determine when complementary experience should be absorbed.

Stage I: Independent Self-Verification Learning. The first stage teaches each agent to assess the reliability of its own reasoning. For a generated trajectory r_i on question x, we construct a preference pair according to its correctness:

$$\left(y_{i,\mathrm{sv}}^{+}, y_{i,\mathrm{sv}}^{-}\right) = \begin{cases} \left(r_i \oplus [\texttt{confirm}],\ r_i \oplus [\texttt{uncertain}]\right), & R(x, r_i) = 1, \\ \left(r_i \oplus [\texttt{uncertain}],\ r_i \oplus [\texttt{confirm}]\right), & R(x, r_i) = 0. \end{cases} \tag{17}$$

Because the preferred and rejected responses contain the same reasoning trajectory and differ only in the terminal marker, the preference signal directly supervises whether the agent should confirm or question its answer, without conflating verification with changes in reasoning style.

Although all agents may share the same initial backbone, their Stage I optimization is fully independent. Each agent is trained on its own bootstrapped question set and maintains a separate parameter copy:

$$\theta_i^{(1)} = \arg\min_{\theta_i} \mathscr{L}_{\mathrm{DPO}}\left(\pi_{\theta_i}; \pi_i^{(0)}, \mathscr{D}_{i,\mathrm{sv}}\right). \tag{18}$$

No gradient or parameter update is shared across agents. This independent training preserves heterogeneous competence profiles and prevents agents from collapsing toward identical behaviors and failure modes.

Stage II: Agent-Specific Cross-Agent Correction. Once self-verification has been acquired, the resulting Stage I policies $\{\pi_i^{(1)}\}_{i=1}^{M}$ are used to generate and evaluate trajectories on a shared question pool. Two conditions must hold jointly for a target agent A_i to be exempted from collaboration on a question: it must answer correctly, and it must be genuinely confident in doing so. Given an uncertain trajectory r_i and a better trajectory r_j produced by peer A_j, the corrected response is preferred over the target agent's uncorrected behavior:

$$\begin{aligned} y_{i,\mathrm{cc}}^{+} &= r_i \oplus [\texttt{uncertain}] \oplus r_j \oplus [\texttt{confirm}], \\ y_{i,\mathrm{cc}}^{-} &= r_i \oplus [\texttt{confirm}]. \end{aligned} \tag{19}$$

The resulting dataset $\mathscr{D}_{i,\mathrm{cc}}$ is inherently agent-specific: its training instances are determined by the particular failures of A_i and by which peers outperform it on those failures. Hence, even when agents encounter questions from the same source distribution, they receive different corrective supervision. Stage II then optimizes

$$\theta_i^{(2)} = \arg\min_{\theta_i} \mathscr{L}_{\mathrm{DPO}}\left(\pi_{\theta_i}; \pi_i^{(1)}, \mathscr{D}_{i,\mathrm{cc}}\right), \tag{20}$$

using the Stage I policy as both the initialization and the frozen reference policy. Agents are therefore co-trained through exchanged experience, while their parameters remain independently optimized.

Dataset-Specific Sequential Training. We instantiate the complete two-stage procedure separately on GSM8K, MATH, and DAPO-Math-17K, without mixing their training examples. For each instantiation, both stages construct preference data exclusively from the training split of the same source dataset. Stage I independently bootstraps source questions before trajectory generation. Stage II first evaluates all Stage I agents on a shared pool from the same training split, ensuring that their trajectories are comparable, and then independently bootstraps the resulting preference groups for each target agent. Keeping the three training corpora separate enables controlled cross-domain evaluation: any improvement on a benchmark outside the corresponding training corpus reflects generalization rather than exposure to a mixed training distribution.

This shared source distribution also explains the functional continuity between the two stages. Learning to emit calibrated verification markers can slightly perturb the original generation distribution and may cause a modest accuracy decrease for some backbones after Stage I. However, Stage II starts from the resulting self-verifying model and provides targeted preferences containing superior solutions to failures drawn from the same source distribution. It can therefore recover the small generation cost of learning verification while further strengthening the agent's task-solving and self-correction capabilities.

Why Direct Preference Optimization (DPO). Neither stage requires an additional supervised fine-tuning phase. First, the instruction-tuned backbones already possess the basic ability to generate complete reasoning trajectories; the goal of Stage I is not to teach a new solution format but to establish a relative preference between [`confirm`] and [`uncertain`] conditioned on the generated solution. Second, Stage II supervision is naturally comparative: the model must prefer a corrected continuation containing superior peer experience over its original uncorrected behavior. DPO directly represents both learning problems, whereas SFT would maximize the likelihood of preferred responses alone and discard the contrastive information carried by the rejected alternatives. Moreover, naively imitating all peer solutions through SFT could encourage indiscriminate distillation and erode the diversity among agents. DPO instead internalizes peer experience selectively while regularizing each policy against its preceding-stage reference.

Self-verification is thus not an auxiliary endpoint but the prerequisite for effective cross-agent learning. Although acquiring it may introduce a small short-term perturbation in task accuracy, it equips agents with the ability to identify where their reasoning is unreliable. Once this capability is established, subsequent preference optimization can target precisely those failures and convert complementary peer trajectories into lasting task improvements. In this sense, *an agent must first learn to recognize its own errors before it can learn effectively from others*.

B Engineering Challenges and Solutions

B.1 Verification Marker Design

A subtle but consequential design decision concerns the realization of [`confirm`] and [`uncertain`]. Rather than introducing new entries into the vocabulary, we repurpose two *reserved special tokens* already present in the backbone's tokenizer. This choice is deliberate on two counts.

First, resizing the vocabulary would perturb the pretrained output distribution. Newly added entries carry randomly initialized embeddings and output-head rows, so their logits are uncalibrated with respect to the pretrained token distribution; the softmax over the enlarged vocabulary must be renormalized, and gradients that fit these parameters propagate into the shared representation. Under preference optimization, this is especially harmful: the implicit reward of DPO is a log-probability ratio against a frozen reference policy, yet a token absent from that reference has no meaningful likelihood under it, leaving the corresponding ratio ill-defined and the resulting signal dominated by embedding initialization rather than by the quality of the reasoning trajectory. Reusing an existing token keeps the vocabulary, the embedding matrix, and the reference policy's support intact, so what the model learns is when to emit a marker, not what the marker's embedding should be.

Second, among existing tokens, we deliberately avoid semantically loaded ones such as beginning- or end-of-sequence markers, which the backbone already emits with high probability in ordinary completions. Anchoring confidence to such a token would entangle our signal with a strong pretrained prior: the model's prior tendency to terminate a sequence would masquerade as confidence, and preference gradients on the marker would interfere with its generation behavior. Reserved tokens that never occur in natural completions carry essentially no pretrained prior, providing a semantically neutral carrier whose probability mass is shaped exclusively by our self-verification objective.

Accordingly, we assign backbone-specific marker tokens as listed in Table 8. We repurpose reserved tokens that already encode to a single ID in the original vocabulary, which avoids vocabulary expansion and the associated embedding-resizing and initialization issues. For Qwen3, we instead repurpose the multimodal placeholders `<image_pad>` and `<video_pad>`; since these rarely appear in the text-only tasks we target, they introduce no interference with the pretrained token distribution.

Backbone	**Uncertain** (U)	**Confirm** (C)
Llama-3.2-3B-Instruct	<|reserved_special_token_1|>	<|reserved_special_token_0|>
Llama-3.1-8B-Instruct	<|reserved_special_token_1|>	<|reserved_special_token_0|>
Qwen3-4B	<image_pad>	<video_pad>

Table 8: Overview of Verification Marker Tokens.

Both verification markers therefore encode to independent single-token IDs, so the confidence in Eq. 21 is read off a single next-token distribution—no sequence-level marginalization, no additional forward pass, and no architectural modification.

$$c_i(x, r_i) = \frac{p_i([\texttt{confirm}])}{p_i([\texttt{confirm}]) + p_i([\texttt{uncertain}])}. \tag{21}$$

However, both semantically vacuous reserved markers receive very little probability mass under the softmax over the full vocabulary, which is dominated by tokens the backbone was pretrained to produce. The absolute probabilities are therefore compressed into a narrow range, so thresholding on them is sensitive to numerical resolution and to the shifting scale of the remaining vocabulary rather than to the agent's actual preference. Instead, the *logit gap* sidesteps this issue by discarding the shared normalizer altogether: it measures the relative preference between the two markers directly, and consequently yields a markedly sharper ordering over questions. Hence, we adopt the underlying *logit gap* to be the more reliable quantity in practice,

$$g_i(x, r_i) = lg_i^{[\texttt{confirm}]}(x, r_i) - lg_i^{[\texttt{uncertain}]}(x, r_i) \tag{22}$$

where $lg_i^v(x, r_i)$ denotes the logit that agent A_i assigns to marker $v \in \{[\texttt{confirm}], [\texttt{uncertain}]\}$ at the position immediately following the trajectory r_i, i.e. the pre-softmax score produced by the language modeling head for that token in the next-token distribution conditioned on (x, r_i). Since preference learning is fundamentally a ranking decision, we report g_i as the main metric for evaluation, while treating confidence score c_i as its bounded counterpart for probabilistic analysis.

B.2 Agent-Wise Bootstrapped Self-Verification Data

A central challenge is to develop self-verification without collapsing agent diversity. Because all agents share the same backbone, exposing them to identical training questions would produce correlated updates and increasingly homogeneous failure modes, thereby diminishing the complementary expertise required for cross-agent correction. Meanwhile, imbalanced correctness outcomes may induce superficial marker priors rather than trajectory-dependent verification. TACIT therefore combines agent-wise question bootstrap with symmetric preference construction, diversifying agents' learning experiences while ensuring equal exposure to both verification markers. Preserving such diversity is not merely an ensemble regularizer; it is a prerequisite for Stage II, where one agent's strength must remain available to correct another agent's characteristic weakness.

To diversify self-verification experience across agents without imposing an artificial correctness prior, we independently bootstrap the training questions for each agent. Given a training set of N questions, agent A_i draws N questions with replacement using an agent-specific random seed. Thus, writing $U_N = \sum_{j=1}^{N} \mathbb{I}[\text{question } j \text{ is drawn at least once}]$ for the number of distinct questions in samples, linearity of expectation gives

$$\mathbb{E}[U_N] = N\left[1 - \left(1 - \frac{1}{N}\right)^N\right], \tag{23}$$

so that

$$\frac{\mathbb{E}[U_N]}{N} \longrightarrow 1 - e^{-1} \approx 0.632. \tag{24}$$

This preserves the nominal dataset size while exposing different agents to overlapping but non-identical subsets, each containing approximately 63.2% unique source questions in expectation. For every sampled question x, we generate multiple reasoning trajectories and retain all of them, including incorrect generations. For a trajectory r_i, the self-verification preference pair is constructed as

$$\left(y^+_{i,\text{sv}}, y^-_{i,\text{sv}}\right) = \begin{cases} (r_i \oplus [\texttt{confirm}],\ r_i \oplus [\texttt{uncertain}]), & R(x, r_i) = 1, \\ (r_i \oplus [\texttt{uncertain}],\ r_i \oplus [\texttt{confirm}]), & R(x, r_i) = 0. \end{cases} \tag{25}$$

where $R(x, r_i) \in \{0, 1\}$ denote the correctness of trajectory r_i against the ground truth, i.e. $R(x, r_i) = 1$ if the final answer is correct and 0 otherwise.

Since the two responses differ only in their terminal marker, the preference signal directly supervises self-verification rather than the reasoning content. Moreover, every pair contains both markers exactly once, ensuring symmetric marker exposure, while the preference direction preserves each agent's empirical correctness distribution. Consequently, incorrect trajectories are not discarded but converted into valuable supervision for recognizing uncertainty. Without resampling, agents trained on identical data would converge to highly correlated failure modes, weakening the cross-agent transfer signal in Stage II.

C Additional Results

C.1 Ablation Studies

Breakdown of the decision. We ablate the two training stages factorially on MATH-500 with Qwen3-4B. Removing Stage II withholds peer experience entirely, leaving agents that verify their own answers but never learn from one another. Removing Stage I is subtler and is the comparison we care about most: peer trajectories are still grafted, and on the same number of questions, but the confidence signal that selects those questions no longer exists, so the choice reduces to a random gate. This variant therefore holds the *amount* of transferred experience fixed and varies only *where* it is applied, isolating the contribution of gating from that of additional supervision. The four cells of Table 3 correspond to the backbone alone, each stage in isolation, and the full framework.

Precision of the gate, not the volume of data. Random gating recovers a substantial part of the gain ($52.40 \rightarrow 63.00$), which is unsurprising: verified peer solutions are useful supervision wherever they are placed. But placing them by the agent's own uncertainty adds a further 5.28 points ($63.00 \rightarrow 68.28$) at identical data volume, and this margin is attributable to gating alone. Indiscriminate injection overwrites behaviour the agent had already mastered and spends its preference budget on questions it could solve unaided, whereas gated injection concentrates the signal on the failures that remain. The gate is thus not an efficiency measure layered on top of experience transfer; it is what makes the transfer targeted, and it accounts for roughly a third of the total improvement over the backbone.

The two stages reinforce each other. Each stage also helps more in the presence of the other. Stage I is worth 4.60 points alone and 5.28 once Stage II is in place; Stage II is worth 10.60 alone and 11.28 given Stage I. An additive model of the two effects predicts 67.60% against the observed 68.28%. The interaction follows from the fact that Stage I does not merely add accuracy but produces the signal on which the rest of the framework is defined: it selects where experience is injected during training, and at inference it supplies the criterion by which the distributor records competence and the aggregator weights votes. Removing it therefore degrades three mechanisms at once, which is why its marginal value exceeds what it contributes in isolation. Competence awareness and competence acquisition are two halves of one mechanism, and neither is expendable.

C.2 Self-Verification Behavior Distribution

Table 9 shows that the Stage-I checkpoint exhibits a highly asymmetric confidence-token distribution. The model emits the uncertainty token for 73.68% of the examples, while only 4.21% are assigned the confirmation token. A further 22.11% of the outputs contain no valid confidence marker, resulting in a confidence-marker coverage of 77.89%.

Despite the imbalanced emission frequency, the generated markers exhibit meaningful behavioral discrimination. Answers followed by the confirmation token achieve a final accuracy of 75.00%, whereas answers followed by the uncertainty token achieve only 6.51%. Equivalently, 93.49% of the uncertain-only outputs are incorrect. These results suggest that the model has learned to associate the two special tokens with substantially different answer-quality regimes.

Nevertheless, the checkpoint remains strongly biased toward uncertainty. Among the 68 correct answers with an explicit confidence marker, only 27 are confirmed, corresponding to a confirmation

Output behavior	Count	Proportion	Before-rethink correct	Final-answer correct	Accuracy change
Confirm only	36	4.21%	4 (11.11%)	27 (75.00%)	+63.89 pp
Uncertain only	630	73.68%	41 (6.51%)	41 (6.51%)	0.00 pp
No confidence marker	189	22.11%	4 (2.12%)	4 (2.12%)	0.00 pp
Uncertain followed by confirm	0	0.00%	–	–	–
Total	**855**	**100.00%**	**49 (5.73%)**	**72 (8.42%)**	**+2.69 pp**

Table 9: Distribution of self-verification behaviours at the Stage I checkpoint of Llama-3.2-3B-Instruct on DAPO-Math. Accuracy in the last two columns is computed within each behaviour category rather than over the full test set.

Behavior after uncertainty token	Count	Rate
Immediately terminated without a revised answer	616	97.78%
Generated a non-empty continuation	14	2.22%
Successful incorrect-to-correct revision	0	0.00%
Complete uncertainty–revision–confirmation trajectory	0	0.00%

Table 10: Continuation behavior after emitting the uncertainty token.

recall of 39.71%. Moreover, 22.11% of all generations do not emit either marker. Thus, the Stage-I model demonstrates useful confidence discrimination but incomplete marker compliance and substantial under-confidence.

As shown in Table 10, emitting the uncertainty token does not yet induce autonomous answer revision. Among 630 uncertain-only outputs, 616 terminate immediately after the marker, while only 14 produce a non-empty continuation. None of these continuations changes an incorrect answer into a correct one, and no complete uncertainty–revision–confirmation trajectory is observed. This behavior is consistent with the role of Stage I, which trains confidence assessment rather than correction. The ability to continue after uncertainty and incorporate a higher-quality trajectory is introduced only during Stage-II cross-Agent correction training.

C.3 Training Dynamics

Figure 4 traces how the verification ability develops during Stage I. All three backbones begin near the random baseline of 0.5 and rise sharply within the first few hundred steps, exceeding 0.9 preference accuracy well before the first epoch concludes, which indicates that self-assessment is a readily learnable behaviour rather than one demanding extensive optimization. The three models nevertheless separate clearly, and consistently in both panels. Qwen3-4B learns fastest and most decisively, crossing 0.9 within roughly 600 steps and driving the reward margin to 7.9; Llama-3.1-8B-Instruct follows, converging more gradually to a comparable margin of 7.7; and Llama-3.2-3B-Instruct is slowest throughout and plateaus at 6.3, the smallest separation of the three. The ordering is not merely one of speed. A wider margin means the two markers are pushed further apart in logit space, so the agent is not only ordering them correctly but growing more decisive about which one belongs, and this magnitude is precisely what gating, routing and aggregation consume downstream. The same ranking reappears at evaluation, where Qwen3-4B attains the highest verification AUROC, suggesting that a stronger backbone supplies a more reliable substrate for competence awareness rather than simply a better reasoner. Because the two candidates in each preference pair share an identical trajectory and differ only in the terminal marker, these differences are attributable to self-assessment alone.

D Algorithm Details

This section presents the complete pseudocode for our framework. Algorithm 1 describes the Stage I independent self-verification learning procedure, Algorithm 2 the Stage II confidence-guided cross-agent knowledge transfer, and Algorithm 3 the communication-free multi-agent inference procedure.

We denote the confidence-confirmation token and uncertainty-indication token as C and U, respectively. The model-native end-of-sequence token is denoted by E, and $\oplus$ represents string or token-sequence concatenation. We use M to denote the number of Agents, while K_1 and K_2 denote the numbers of trajectories sampled per question during Stage I and Stage II, respectively.

Algorithm 1 Stage I: Independent Self-Verification Learning

Input: Training set $\mathscr{D}_{\text{tr}}$; shared base policy π_{θ_0}; agents $\{A_i\}_{i=1}^{M}$; confidence tokens C, U; native EOS E; bootstrap seed b_1; generation seed s_1; number of samples K_1

Output: Stage I policies $\{\pi_{\theta_i^{(1)}}\}_{i=1}^{M}$

for all $i \in \{1,\dots,M\}$ **do** ▷ Agents are independent and can be trained in parallel
  $\widetilde{\mathscr{D}}_i \leftarrow \textsc{BootstrapWithReplacement}(\mathscr{D}_{\text{tr}}, b_1 + i)$
  $\mathscr{P}_i^{(1)} \leftarrow \emptyset$
  for all $(q, y^\star) \in \widetilde{\mathscr{D}}_i$ **do**
    for $k = 1$ to K_1 **do**
      $z_{ik} \sim \pi_{\theta_0}(\cdot \mid q; s_1 + i)$
      $a_{ik} \leftarrow \textsc{StripConfidenceMarker}(z_{ik})$
      $r_{ik} \leftarrow \textsc{Evaluate}(a_{ik}, y^\star)$
      if $r_{ik} = 1$ **then**
        $y^+ \leftarrow a_{ik} \oplus \texttt{C} \oplus \texttt{E}$
        $y^- \leftarrow a_{ik} \oplus \texttt{U} \oplus \texttt{E}$
      else
        $y^+ \leftarrow a_{ik} \oplus \texttt{U} \oplus \texttt{E}$
        $y^- \leftarrow a_{ik} \oplus \texttt{C} \oplus \texttt{E}$
      end if
      $\mathscr{P}_i^{(1)} \leftarrow \mathscr{P}_i^{(1)} \cup \{(q, y^+, y^-)\}$
    end for
  end for
  $\theta_i^{(1)} \leftarrow \text{DPO-RPO}(\theta_0, \theta_0, \mathscr{P}_i^{(1)})$
end for
return $\{\pi_{\theta_i^{(1)}}\}_{i=1}^{M}$

Algorithm 2 Stage II: Confidence-Guided Cross-Agent Knowledge Transfer

Input: Training set $\mathscr{D}_{\text{tr}}$; Stage I policies $\{\pi_{\theta_i^{(1)}}\}_{i=1}^{M}$; confidence tokens C, U and native EOS E; suffix length L; samples per question K_2; memory thresholds τ_s, τ_g; bootstrap seed b_2

Output: Stage II policies $\{\pi_{\theta_i^{(2)}}\}_{i=1}^{M}$; memory pool $\mathscr{M}$; Router f_ϕ

Phase 1: Shared candidate pool (unbootstrapped)

$\mathscr{C} \leftarrow \emptyset$

for all $(q, y^\star) \in \mathscr{D}_{\text{tr}}$, agent $i \in [M]$, sample $k \in [K_2]$ **do**

  $z_{ik} \sim \pi_{\theta_i^{(1)}}(\cdot \mid q)$; $a_{ik} \leftarrow \textsc{StripMarker}(z_{ik})$; $r_{ik} \leftarrow \textsc{Eval}(a_{ik}, y^\star)$

  $g_{ik} \leftarrow \textsc{ConfGap}(z_{ik}; \theta_i^{(1)})$; $s_{ik} \leftarrow \sigma(g_{ik})$ ▷ Eq. equation 22

  $\mathscr{C} \leftarrow \mathscr{C} \cup \{(q, y^\star, i, a_{ik}, r_{ik}, g_{ik}, s_{ik})\}$

end for

Phase 2: Per-agent confidence threshold

for all $i \in [M]$ **do**

  $\mu_i \leftarrow \text{mean}\{\, g_c : \; c \in \mathscr{C},\ \text{ag}(c) = i,\ |g_c| < \infty \,\}$

end for

Phase 3: Confidence-guided preference pairs

$\mathscr{P}_i^{(2)} \leftarrow \emptyset$ for all $i \in [M]$

for all question group $\mathscr{C}_q \subseteq \mathscr{C}$, candidate $c \in \mathscr{C}_q$ **do**

  $i \leftarrow \text{ag}(c)$; $\text{need}(c) \leftarrow \neg[r_c = 1 \wedge g_c > \mu_i]$ ▷ skip only if confidently correct

  if need(c) **then**

    $\mathscr{B}_c \leftarrow \textsc{SelectBetterPeers}(c, \mathscr{C}_q)$ ▷ ≤ 1 traj. per peer

    if $\mathscr{B}_c \neq \emptyset$ **then**

      for all $t \in \mathscr{B}_c$ **do**

        $y^+ \leftarrow a_c \oplus \texttt{U} \oplus a_t \oplus \texttt{C} \oplus \texttt{E}$; $y^- \leftarrow a_c \oplus \texttt{C} \oplus \texttt{E}$

        $\mathscr{P}_i^{(2)} \leftarrow \mathscr{P}_i^{(2)} \cup \{(q, y^+, y^-, \text{teacher} = A_t)\}$

      end for

    else

      $y^+ \leftarrow a_c \oplus \texttt{U} \oplus y^\star \oplus \texttt{C} \oplus \texttt{E}$; $y^- \leftarrow a_c \oplus \texttt{C} \oplus \texttt{E}$ ▷ fall back to GT

      $\mathscr{P}_i^{(2)} \leftarrow \mathscr{P}_i^{(2)} \cup \{(q, y^+, y^-, \text{src} = \text{GT})\}$

    end if

  else if IncludeNonCollab **then**

    $y^+ \leftarrow a_c \oplus \texttt{C} \oplus \texttt{E}$; $y^- \leftarrow a_c \oplus \texttt{U} \oplus \texttt{E}$

    $\mathscr{P}_i^{(2)} \leftarrow \mathscr{P}_i^{(2)} \cup \{(q, y^+, y^-)\}$

  end if

end for

Phase 4: Memory admission and Router

$\mathscr{M} \leftarrow \textsc{BestPerQuestionAgent}(\{c \in \mathscr{C} : r_c = 1 \wedge s_c \ge \tau_s \wedge g_c \ge \tau_g\})$

$e_q \leftarrow \textsc{Normalize}\big(\textsc{MeanPool}(\textsc{Embed}(q))\big)$ for all $(q, i, \cdot) \in \mathscr{M}$

$\phi \leftarrow \arg\min_\phi \sum_{(e_q, i) \in \mathscr{M}} w_i \, \text{CE}\big(f_\phi(e_q), i\big)$ ▷ class-balanced linear head

Phase 5: Bootstrapped Stage II optimization

for all $i \in [M]$ **do**

  $\widetilde{\mathscr{P}}_i^{(2)} \leftarrow \textsc{BootstrapGroups}(\mathscr{P}_i^{(2)}, b_2 + i)$

  $\theta_i^{(2)} \leftarrow \text{DPO-RPO}(\theta_i^{(1)}, \text{sg}[\theta_i^{(1)}], \widetilde{\mathscr{P}}_i^{(2)})$ ▷ reference frozen at Stage I

end for

return $\{\pi_{\theta_i^{(2)}}\}_{i=1}^{M}$, $\mathscr{M}$, f_ϕ

Algorithm 3 Communication-Free Multi-Agent Inference

Input: Question q; Stage II agents $\{\pi_{\theta_i^{(2)}}\}_{i=1}^{M}$; Router f_ϕ; number of routed agents K; suffix length L; confidence floor ε; aggregation switch $\gamma_{\text{agg}} \in \{0,1\}$

Output: Final answer $\hat{y}$

Phase 1: Routing

$e_q \leftarrow \textsc{Normalize}\big(\textsc{MeanPool}(\textsc{Embed}(q))\big)$

$\mathscr{I}_q \leftarrow \textsc{TopK}(f_\phi(e_q), K)$ ▷ $|\mathscr{I}_q| = K$ routed agents

Phase 2: Independent generation (no inter-agent communication)

for all $i \in \mathscr{I}_q$ **do**

  $z_i \sim \pi_{\theta_i^{(2)}}(\cdot \mid q)$; $a_i \leftarrow \textsc{StripMarker}(z_i)$

  $g_i \leftarrow \textsc{ConfGap}(z_i; \theta_i^{(2)})$; $s_i \leftarrow \sigma(g_i)$ ▷ Eq. equation 22

end for

Phase 3: Confidence-weighted aggregation

if $\gamma_{\text{agg}} = 1$ **then**

  $\mathscr{G} \leftarrow \textsc{GroupByAnswer}(\{a_i\}_{i \in \mathscr{I}_q})$

  $W_G \leftarrow \sum_{i \in G} \max(\varepsilon, s_i)$ for all $G \in \mathscr{G}$ ▷ floor ε retains unconfident votes

  $G^\star \leftarrow \arg\max_{G \in \mathscr{G}} W_G$; $\hat{y} \leftarrow a_{i^\star}$, $i^\star = \arg\max_{i \in G^\star} s_i$

else

  $\hat{y} \leftarrow a_{i_1}$, where i_1 is the top-1 routed agent ▷ single-agent ablation

end if

return $\hat{y}$

E Prompt Templates

For readability, we present the prompt template for each dataset as plain text. The final input format is determined by each model's own tokenizer chat template, which we apply without modification.

GSM8K

> {question}
>
> Let's think step by step. Put your final answer at the end with "The answer is:".

DAPO-Math

> Solve the following math problem step by step. The last line of your response should be of the form Answer: $Answer (without quotes), where $Answer is the answer to the problem.
>
> {question}
>
> Remember to put your answer on its own line after "Answer:".

Hendrycks-MATH

> {question}
>
> Solve the problem step by step and put only the final answer in `\boxed{}`.

AIME 2024 & 2025

> {question}
>
> Solve the problem step by step. End your response with exactly "The answer is: N", where N is the requested AIME integer from 000 through 999.

MBPP

{question}

Write a correct Python solution. Return only the implementation in one "python code block", with no example calls or explanation outside the block.

BBH

{question}

Reason carefully, then end your response with exactly "The answer is: <answer>". Preserve the answer text or option label shown by the task.

F DEMO CASES

To intuitively demonstrate the effectiveness of our proposed method TACIT, we show two generations that illustrate the behaviour each stage is responsible for. The first is a Stage I output in which the agent reaches an incorrect answer and flags it with the uncertainty marker but does not revise it; the second is a Stage II output in which the same uncertainty triggers the absorption of a peer trajectory that supplies the missing insight. Together they trace the division of labour in TACIT: Stage I decides *whether* an answer should be *trusted*, and Stage II decides *what to do* when it should not. **Red** marks the erroneous span in generation; **green** marks the correction supplied by the selected peer trajectory.

Uncertain Case on MATH-500 after Stage I

Question: Helena is a mad scientist testing her latest creation, the Lots-of-Limbs Serum. The serum will make the drinker grow an extra arm every three days and an extra leg every five days. After fifteen days, how many new limbs will Helena's serum cause a person to grow if they drink it?
Solve the problem step by step and put only the final answer in `\boxed{}`.

Reasoning: Since the serum causes an extra arm every three days, we divide fifteen days by three: $15/3 = 5$ extra arms. ... **Since a person cannot grow more than one extra limb of the same type, the maximum number of extra limbs in fifteen days is 2.**

Answer: $\boxed{2}$ [uncertain]

Correction Case on AIME 2024 after Stage II

Question: Let $\mathscr{B}$ be the set of rectangular boxes with surface area 54 and volume 23. Let r be the radius of the smallest sphere that can contain each box in $\mathscr{B}$. If $r^2 = \frac{p}{q}$, where $\gcd(p,q) = 1$, find $p+q$.
Solve the problem step by step. End your response with exactly "The answer is: N", where N is the requested AIME integer from 000 through 999.

Initial Reasoning: Let the side lengths be x, y, z. The constraints are

$$xy + yz + zx = 27, \qquad xyz = 23,$$

and the squared radius required by a box is

$$r^2 = \frac{x^2 + y^2 + z^2}{4}.$$

Therefore, we need to minimize $x^2 + y^2 + z^2$ under these constraints. Assuming two equal sides leads to an irrational candidate, and the trajectory fails to obtain a valid final answer.

Initial Answer: No valid final answer [uncertain]

Peer Correction: **Because one sphere must be large enough to contain every box in $\mathscr{B}$, the required radius is determined by the maximum, rather than the minimum, of the space**

diagonal. At an extremum, two side lengths are equal. Setting $x = y$ gives

$$x^2 + 2xz = 27, \qquad x^2 z = 23,$$

and hence

$$x^3 - 27x + 46 = 0.$$

The maximizing solution is

$$x = y = 2, \qquad z = \frac{23}{4}.$$

Therefore,

$$x^2 + y^2 + z^2 = 4 + 4 + \left(\frac{23}{4}\right)^2 = \frac{657}{16},$$

so

$$r^2 = \frac{657}{64} \quad \Longrightarrow \quad p + q = 657 + 64 = 721.$$

Refined Answer: The answer is: 721 [`confirm`]

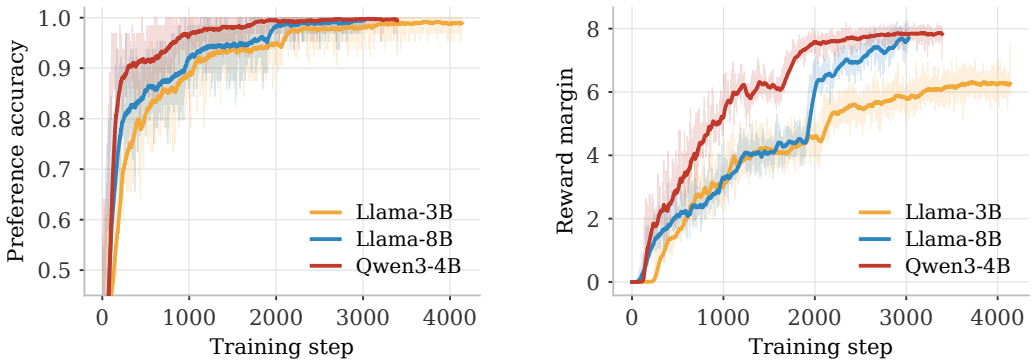


Figure 4: Stage I training dynamics across three backbones. **Left:** the fraction of preference pairs ranked correctly, which measures how reliably an agent distinguishes answers it should confirm from those it should flag as uncertain. **Right:** the implicit reward margin between the chosen and rejected sequences. Faint lines show per-step values and solid lines an exponential moving average. Since the two candidates in each pair share an identical trajectory and differ only in the terminal marker, both curves track the emergence of self-verification in isolation.