Consilience for Verifier-Free Test-Time Scaling

Lecheng Kong, Like Hui, Haitao Mao, Jun Huan
AWS AI Labs
{jkong,likehui,maoht,lukehuan}@amazon.com

Project page w/ interactive demo: https://consilience-tts.github.io/

Abstract

Test-time scaling often uses an external verifier, such as compilers and test cases in coding or trained value functions in robotics applications, to obtain high-quality rollouts. Verifier-free test-time scaling (or VF-TTS) is gaining extensive attention as a mechanism to enhance Large Language Model (LLM) reasoning, primarily because we do not have access to such high-quality verifiers in many real-world applications. Among existing VF-TTS methods, confidence-based VF-TTS methods, which compute and rank rollouts solely by confidence, are particularly promising. Such methods introduce near-zero overhead for sample evaluation and require minimal access to internal model states, making the methods highly flexible across models and tasks. In this paper, we demonstrate a critical limitation of existing confidence-based VF-TTS methods by showing that such methods catastrophically break down on complex tasks. We observe a very interesting phenomenon: uniformly high confidence frequently indicates a failure to explore, favoring confidently wrong answers. To address this, our core insight is that robust cognitive search requires a specific confidence trajectory pattern: such methods perform exploratory branching at the beginning, as manifested by low initial confidence, and converge to a high final confidence solution. To implement this insight, we introduce consilience, a novel selection framework that explicitly evaluates the temporal asymmetry of confidence in reasoning. We operationalize this via a combinatorial metric that actively penalizes high initial confidence while strictly demanding final certainty. Extensive experiments covering both graduate-level mathematics problems and free-form code generation demonstrate that consilience effectively outperforms existing baselines, validating our novel perspective on completion confidence.

1 Introduction

Reasoning models have demonstrated remarkable capabilities across diverse domains. In reasoning-heavy scenarios where accuracy is of utmost importance, parallel test-time scaling (TTS) has emerged as a highly effective method for improving model performance–generating multiple reasoning paths during inference and selecting the optimal output [34, 43, 3]. However, the prevailing mechanisms for selecting the optimal output from these parallel traces rely heavily on external verifiers [34, 33, 46, 21] or models [25, 43]. To scale test-time computation universally in unverifiable scenarios like free-form code generation and intermediate agentic steps, the research community recently started to explore selection mechanisms that rely entirely on the model's self-contained internal signals, such as token-level log-probabilities, to obtain high-quality rollouts.

Existing log-probability-based methods compute and select maximally confident completions [17, 11]. The key insight behind these approaches is: higher confidence implies a higher chance of correctness of the rollout. In Figure 1 (Left), we plot the histogram of the average sequence confidence, and found that the correct answer distribution has a higher mean than the incorrect ones, aligning with the insight aforementioned. However, we discovered that this simple maximization catastrophically breaks down

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

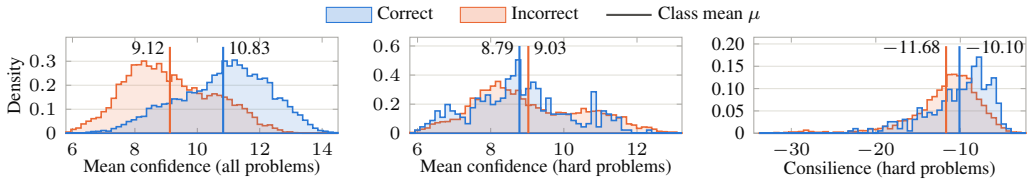


Figure 1: Left: Histogram of completion confidence on LiveCodeBench-V6 using GPT-OSS-120B model. Middle: Same setting as Left but only on hard problems with < 20% correct rate, where incorrect answers exhibit a higher mean confidence ($\mu = 9.03$) than correct ones ($\mu = 8.79$). Right: Consilience histogram on the hard problems. Confidence is effective on a macro level, but breaks on hard problems. However, our approach, consilience, can maintain the effectiveness.

on complex tasks (i.e., model achieving lower correct rate on these problems). It overlooks a critical cognitive factor necessary for solving complex problems: process diversity. To study this insight, in Figure 1 (Middle), we plot the same histogram except we only include questions where the model achieves $< 20\%$ correct rate. In these difficult tasks, the pattern is reversed, and the distributions become inseparable. This result indicates that confidence maximization is useful in solving problems where a model has good answers to such problems, but confidence maximization is less effective where the model has to explore several paths to identify the right solution.

In this paper, we have investigated the problem of effective TTS for hard problems through the lens of consilience—the principle that a **conclusion** is reliable only if the model **knows diverse paths** to solve the problem and all of the paths converge to the same solution [38, 6]. In confidence maximization, however, the confidence of both the answer and reasoning are maximized. That is, the selected completions not only converge at the answer, but also converge early in the reasoning phase. Such selections blindly commit to a single line of thought without the model internally acknowledging alternatives. Hence, "confidently wrong" selection happens because confidence maximization forces the system to focus solely on **quickly reaching a conclusion** or convergence while suppressing the importance of **knowing diverse paths**, breaking the principle of consilience. Fortunately, this observation naturally shed lights on how to enforce the principle: if high confidence represents convergence, we can use low confidence to identify essential process diversity. If a reasoning process shows a robust understanding of a problem, its awareness of alternative solutions forces the probability mass to split, naturally manifesting as a drop in early confidence. A consilient trace should therefore exhibit a rising trajectory: it begins with an exploratory phase characterized by lower confidence as the model recognizes various valid approaches, which then converges into a highly certain conclusion.

To implement this structural signature, we introduce the metric of consilience, a novel, verifier-free selection framework that explicitly quantifies this temporal confidence asymmetry of effective reasoning. Rather than uniformly maximizing sequence-level confidence, our approach evaluates the trajectory via a combined metric that penalizes high initial confidence while still encouraging high final confidence. Using our metric, we can re-plot the histogram of our metric on the hard problems in Figure 1 (right), and show that consilience can successfully recover the pattern to distinguish correct and incorrect responses. Extensive experiments across complex mathematics (HMMT), graduate-level knowledge (GPQA), and open-ended code generation (LiveCodeBench-V6, agentic Swe-Bench) demonstrate the efficacy of this approach. Most notably, consilience unlocks significant margins in free-form coding domains where voting-based technique is inapplicable; on LiveCodeBench, our metric elevates the GPT-OSS-120B model to an accuracy of 69.7%, substantially outperforming Pass@1 (65.7%). Similar systematic gains are observed in other reasoning-heavy datasets.

2 Related Work

Test-time scaling (TTS) devotes more test-time computation to achieve better performance. Distinct from sequential TTS signified by thinking models [28], this paper focuses on parallel TTS. Parallel TTS was initially effective but relied on strict environmental dependencies such as external verifiers [34, 33], causing serious latency and generalizability issues. To expand utility to broader tasks, methods such as Self-Consistency [38] take a majority vote over sampled answers, but this imposes a strict constraint on the answer format (multiple-choice, integers) and does not apply to

free-form generation. Another line of work uses the LLM itself to provide feedback—via LLM-based evaluators, process reward models, or self-refinement [6, 25, 26]—but these introduce significant aggregation overhead. Closest to us, several methods use token-level log-probabilities as a direct proxy for correctness to enable highly efficient, verifier-free selection [17, 11]. Our work builds on the computational advantages of these internal signals but diverges in usage: rather than assuming confidence should be globally maximized, we show that uniformly high confidence is a symptom of premature convergence, and introduce the consilience trajectory to explicitly reward early-stage exploration. A broader discussion of verifier-based TTS, search, and entropy/confidence in calibration and RL is in Appendix F.

Prefix and temporal confidence. A growing body of work—much of it concurrent—studies how confidence *evolves* across reasoning rather than its overall level, and our findings are consistent with this literature. On the "penalize early confidence" side, Gai et al. [12] show that premature confidence (early commitment followed by rationalization) predicts flawed reasoning and mitigate it with an RL reward for gradual confidence growth; Wang et al. [39] find correct traces exhibit rising confidence and use this "confidence gain" to reweight majority voting; Keramati et al. [18] find early-token confidence predicts reasoning quality in multi-agent debate; and related work shows the initial reasoning step disproportionately shapes the outcome [24], that per-step confidence carries a temporal signal [27], that reasoning models are better calibrated in expressing confidence [44], and that reasoning can nonetheless inflate confidence even when wrong [10]. Seemingly opposed, Otth et al. [30] report that *maximizing* prefix confidence improves mathematical reasoning; we reconcile this by noting the effect is difficulty-dependent—on easy problems, where Pass@1 is already high, high prefix confidence is a reliable signal and consilience is effectively neutral, whereas the initial penalty becomes beneficial on hard problems where uniform confidence marks premature convergence (Section 4.5, Figure 3). We regard this convergence of independent findings as validation of the underlying insight, and our contribution is distinct in both formulation and applicability. First, rather than only observing the pattern or using it to reweight votes, we operationalize it as an *active selection objective* that penalizes high initial confidence while rewarding final convergence. Second, being training-free and computed from logits alone, consilience applies directly to *free-form generation* where voting is inapplicable (code generation, agentic edits)—a regime none of the above addresses, since voting requires extractable answers and the RL approach requires training. Third, and uniquely, we isolate the reasoning phase from the final-answer phase before scoring, removing the artificially confident summarization tokens that dilute the signal.

3 Method

3.1 Preliminary: Token-Level Confidence

To understand and evaluate a generation's confidence, we follow the formulation in previous work [11, 17] and define token-level confidence. For a given prompt x and a generated sequence $y = (y_1, y_2, \ldots, y_L)$, the model outputs a probability distribution over the vocabulary at each time step t. Instead of relying solely on the probability of the single decoded token y_t, the token confidence is collected over the top-K candidate tokens. The token-level confidence $c_{y,t}$ as the negative mean log-probability of these top-K tokens:

$$c_{y,t} = -\frac{1}{K} \sum_{j=1}^{K} \log P_\theta(v_{t,j} | y_{<t}, x), \tag{1}$$

where $v_{t,j}$ represents the j-th most probable token in the vocabulary at step t. A centralized distribution causes the value to be high, indicating high confidence, whereas a flat distribution leads to low confidence.

3.2 Why Confidence is Insufficient: The Long-Tail Trap of Hard Problems

To scale test-time compute without relying on external verifiers or exact-match consensus, recent methods maximize internal confidence metrics across a set of sampled sequences. These are typically implemented as the average token-level confidence across the generated sequence [17, 11]:

$$y = \arg\max_{y} \frac{1}{L} \sum_{t=1}^{L} c_{y,t}. \tag{2}$$

These approaches operate on a straightforward principle: maximizing a model's confidence maximizes correctness. This direction is particularly valuable for two primary reasons. **Flexibility.** It requires only token-level logits and no training, making it readily applicable to open-weight models and closed-weight enterprise LLMs (e.g., Google Gemini) that expose log-probabilities via API. **Efficiency and Latency.** It introduces near-zero computational overhead for evaluating each completion. All completions can be sampled in parallel, providing better latency than sequential scaling.

To validate the efficacy and limitations of confidence maximization, we analyzed the mean confidence distributions of GPT-OSS-120B completions on the LiveCodeBench-V6 (LCB) dataset, a free-form code generation task. In Figure 1 left, across the entire dataset, higher average confidence strongly correlates with correctness. This confirms the foundational intuition driving existing confidence-based metrics: on a macro level, confident generations are generally more accurate, and existing methods correctly interpret this global signal.

However, this approach becomes counterproductive on complex tasks. When isolated to "hard" problems (correctness rate $< 20\%$, Figure 1 middle), the confidence dynamic inverts: incorrect answers exhibit a higher mean confidence than correct ones. Crucially, the incorrect distribution on hard problems is not merely shifted—it becomes highly **over-dispersed**, forming a heavy "long tail" at the highest confidence percentiles. In free-form generation where exact-match majority voting is inapplicable, verifier-free test-time scaling often relies on argmax selection (picking the single highest-scoring completion from the pool). Because of this right-skewed tail, simply maximizing average confidence guarantees that the algorithm will systematically sample "confidently wrong" hallucinations.

3.3 Understanding Confidence through Consilience

To understand such contradictory behavior under different task difficulty, we introduce the concept of consilience. In scientific epistemology, the principle of consilience dictates that a conclusion is profoundly more reliable when it emerges from multiple and diverse lines of inquiry rather than a single, isolated path. Within LLM research, this foundational idea has been widely and successfully adopted—most notably through methods like Self-Consistency [38, 6]. When we evaluate the failure of global confidence maximization through this lens, its drawback becomes apparent. By rewarding traces that exhibit uniformly high confidence from the very first tokens, these metrics optimize for mere convergence and favor completion converging to an unambiguous single path in the quickest manner. Simultaneously, this strategy suppresses the completions that acknowledge diverse paths. This structurally leads to the selection of completion with an overconfident reasoning process.

To ground this insight in the actual mechanics of an LLM, we examine how confidence is computed. In an autoregressive language model, the confidence of generating the t-th token y_t given a prompt x and the preceding generated sequence $y_{<t}$ is coupled with its conditional probability: $P_\theta(y_t|y_{<t}, x)$. Crucially, a token's confidence does not measure the objective correctness of the future sequence; rather, it measures how strictly the past reasoning effort ($y_{<t}$) entails the current token. Because of this, the implications of confidence are phase-dependent:

Final Stage (Large t): The intuition behind existing confidence maximization is rooted in the final stages of reasoning. As a model successfully navigates its chosen path, the logical constraints should tighten. If confidence is **high at the end** of the reasoning trace, it indicates that the preceding, lengthy chain of thought has successfully and systematically converged, and the exhaustive reasoning steps inevitably entail the final conclusion.

Initial Stage (Small t): In the early steps of generation, high confidence is still a symptom of convergence. However, it implies that a very short prefix has already collapsed the probability space to a singular path. For complex reasoning tasks, this is highly detrimental; it signifies that the model has failed to recognize the difficulty or nuance of the problem, bypassed necessary exploration, and overfitted to a local, often flawed, solution. This blind commitment leads directly to the "confidently wrong" hallucinations observed in the long tail. Conversely, if a model possesses a robust understanding of a complex problem (one requiring exploration of many possibilities), it will naturally acknowledge that multiple viable routes exist to reach a solution, aligning with the consilience principle. This internal awareness of alternative strategies forces the autoregressive probability mass to split, reflecting a necessary cognitive divergence. Consequently, this essential exploratory phase inherently manifests as **low initial confidence**.

Table 1: The Four Categories of LLM Reasoning Trajectories

	High Final Confidence	**Low Final Confidence**
High Initial Confidence	**Premature Convergence (Category 1):** The model immediately collapses onto a singular path. Often occurs on trivially easy queries or when the model overfits to a flawed premise, leading to confidently wrong answers.	**Degeneration (Category 2):** The model confidently begins a path but encounters logical contradictions or out-of-distribution states later in the sequence, causing the reasoning to break down.
Low Initial Confidence	**Consilience (Category 3):** The model actively sees diverse paths (low initial confidence) before logically converging on a well-supported, highly certain conclusion.	**Total Confusion (Category 4):** The prompt is too difficult for the model. It fails to explore meaningful paths initially and fails to reach a logical consensus at the end.

Therefore, genuine reasoning on complex problems is not a flat line of certainty; it is a dynamic trajectory. It demands a temporal asymmetry: low initial confidence resolving into high final confidence. By dissecting a reasoning trace into its initial and final confidence phases, we can categorize the model's generation into four distinct categories as shown in Table 1.

Prior research correctly recognized the predictive power of confidence, effectively utilizing it to filter out trajectories that fail to reach logical convergence (Categories 2 and 4). However, their application of this signal remains coarse. By relying on simple maximization, existing methods inadvertently optimize for the Category 1 while actively ignoring—or mathematically penalizing—the essential initial uncertainty required for the Low-High Category 3, representing consilience. This structural blind spot is why existing mechanisms systematically underperform on complex reasoning tasks.

In the following sections, we formalize how to implement consilience as a selection metric designed specifically to target and identify generations within this optimal consilient Category.

3.4 The Consilience Score Formulation

With the token-level metric $c_{y,t}$ defined, we quantify the consilience trajectory for a generated sequence y of length L. From Section 3.2, a correct, well-explored reasoning path should exhibit low initial confidence (reflecting active path exploration) that eventually converges into high final confidence (reflecting logical consensus). To capture this principle, we use a boundary window of size W to capture the initial and final confidence:

$$C_{initial} = \frac{1}{W} \sum_{t=P+1}^{W+P} c_t, \quad C_{final} = \frac{1}{W} \sum_{t=L-W+1}^{L} c_t \tag{3}$$

The initial and final confidence is quantified by the average token confidence of the first and last W tokens. Notice that we also skip the first P tokens, because the initial tokens will have very similar confidence (e.g., the first token will have the same confidence given the same prompt), which will dilute the confidence signal. Finally, we define the Consilience Score (S) for a given reasoning trace to explicitly reward high final confidence while penalizing high initial confidence.

$$S = C_{final} - \alpha C_{initial}, \tag{4}$$

where $\alpha \geq 0$ is a balancing factor. During test-time scaling, we sample multiple completions for a given prompt, compute S for each sequence, and select the completion that maximizes S. By minimizing initial confidence and maximizing final confidence, this objective actively selects for the diverse exploration and robust convergence that characterizes true consilience. We plot the consilience histogram on the same hard problems set in Figure 1, the consilience mean of correct responses is significantly higher than that of incorrect ones. More importantly, the incorrect distribution does not exhibit an overly dispersed long-tail distribution, making consilience maximization a stable metric for free-form generation problems.

3.5 Reasoning-Phase Isolation

A critical nuance in applying confidence metrics to models trained with Chain-of-Thought (CoT) or explicit `<think>` tokens lies in the structural composition of the generated sequence. Autoregressive outputs under this paradigm typically consist of two distinct semantic phases: the active reasoning phase (Y_{reason}) and the final answer formulation (Y_{answer}). Specifically, the tokens within Y_{answer} do not reflect active problem-solving or cognitive search. These tokens mostly serve to summarize or mimic the logical conclusion already established within Y_{reason}. Consequently, the log-probabilities of Y_{answer} tokens can exhibit artificially high confidence, independent of the actual quality of reasoning. Including these final tokens in sequence-level evaluations hence acts as a noise factor, masking the model's true cognitive trajectory.

To extract a purer signal of the underlying thinking dynamics, we propose a reasoning-phase-isolated variant to full-sequence consilience. For any generated completion y, we parse the sequence to identify the semantic transition boundary and only take the reasoning portion of the generation. This isolation carries negligible cost and poses no practical obstacle: all mainstream thinking models expose an easily identifiable delimiter—e.g., `</think>` for DeepSeek-R1, Qwen3, Mistral, and Llama, and `<|channel|>final` for GPT-OSS—so the split is a single string operation. Even absent a dedicated delimiter, the `-full` variant remains a valid instantiation of the principle (Table 2), and any non-thinking model can be prompted to emit a "Final Answer" marker to recover a unified boundary. All subsequent boundary window calculations (W) and confidence aggregations are then strictly constrained to the isolated reasoning trace Y_{reason}, ensuring that the consilience score reflects only the genuine cognitive search and resolution process. For the non-thinking model, full sequence or segments before the "Final Answer" part are both applicable, as they enclose the actual reasoning process.

4 Experiment

In this section, we strive to answer the following research questions. **Q1**: How does consilience compare to other confidence-based TTS methods? **Q2**: How does consilience scale w.r.t. the number of completions? **Q3**: Does the sequence confidence follow the consilience principle? **Q4**: Is consilience robust against hyperparameters? What are the recommended hyperparameters?

4.1 Experimental Setup

Model. We utilize a diverse suite of open-weight large language models: DeepSeek-R1-Qwen-7B-Distill (DS-R1) [14], Qwen3-80B-Next-Instruct-FP8 (Qwen) [41], Qwen3-80B-Coder-Next-Instruct-FP8 [4] (Qwen3-Coder-Next), and GPT-OSS-120B/20B (OSS-120B/20B) [29]. We use high-thinking mode for GPT-OSS-20B, and medium-thinking mode for the 120B model. This selection is intentionally designed to cover a broad spectrum of parameter sizes. Furthermore, it encompasses both explicit reasoning architectures (Deepseek and GPT-OSS) and standard instruction-tuned models, ensuring our findings generalize across different generative paradigms.

Datasets. We assess test-time scaling performance across four challenging, reasoning-heavy benchmarks: LiveCodeBench-v6 (LCB) [15] for code generation, SWE-bench [16] for agentic application, HMMT25 [2] for advanced mathematical reasoning, and GPQA-Diamond [31] for graduate-level question answering. This selection provides comprehensive coverage across diverse cognitive domains. Crucially, LCB-v6 and SWE-bench require the generation of expressive, free-form programmatic solutions. In these environments, methods relying on internal signals like consilience are readily applicable, whereas methods relying on majority voting have limited application.

Experimental Setup. To assess the model's ability to scale in the general setting, we adopt the Best-of-n selection paradigm, where we sample n completions, and use a test-time scaling method to rank and select the Top-1 completion as the final answer. For one-turn datasets (LCB, GPQA, HMMT), we employ a shared-pool evaluation strategy. For every query, we first generate a common pool of 256 independent completions. During evaluation, we randomly draw a subset of size n from this common pool and apply each top-1 selection method. We repeat the experiment 10 times. For the agentic case (SWE-bench), we apply each method to individual steps, and we only apply test-time scaling to edit steps to save computation costs. Detailed agentic setup is in Appendix I. We also

Table 2: Accuracy (%) across models and datasets. All TTS methods have a sample size of $n = 64$. Bold = best per column.

	LCB				HMMT		GPQA	
Method	GPT-OSS-120B	GPT-OSS-20B	DS-R1	Qwen	GPT-OSS-20B	Qwen	GPT-OSS-20B	Qwen
Pass@1	65.7±0.2	80.2±0.1	47.8±0.1	55.3±0.2	76.5±0.3	52.6±0.5	71.8±0.2	60.5±0.2
Self-Certainty	64.5±1.3	79.0±1.3	54.9±1.5	53.2±1.7	68.0±3.1	45.3±3.1	71.7±1.4	57.8±1.8
DeepConf-20%	61.9±1.3	79.1±0.9	53.7±1.6	52.0±1.6	69.3±3.3	51.3±4.3	71.0±1.2	60.4±1.3
DeepConf-2K	62.5±0.8	76.7±1.2	53.7±1.4	55.4±1.8	73.3±4.9	47.3±3.6	71.3±1.3	57.6±2.4
Cns-full	62.2±1.0	80.2±0.4	52.0±1.9	59.5±2.0	78.3±2.7	49.7±3.5	72.5±1.1	**65.2±1.6**
Cns-2K-full	64.6±1.7	79.4±0.8	56.0±0.3	**60.9±1.6**	77.0±4.6	51.0±3.3	73.8±1.4	58.1±1.5
Cns-think	**69.7±0.9**	**81.1±0.7**	54.0±0.5	58.7±0.9	78.0±2.7	52.0±2.7	72.0±1.0	-
Cns-2K-think	67.8±1.5	80.7±0.6	**57.1±0.9**	60.4±1.9	**80.7±2.5**	**53.7±2.3**	**73.8±1.5**	-

Table 3: SWE-bench Verified issue resolve rate (%) across models and agents. ($n = 32$) Bold = best per column.

Tool	Model	% Resolved
mini-SWE-agent	GPT-OSS-120B	23.0±0.8
mini-SWE-agent + Consilience	GPT-OSS-120B	**26.9±1.1**
mini-SWE-agent	Qwen3-Coder-Next	65.3±0.8
mini-SWE-agent + Consilience	Qwen3-Coder-Next	**67.3±1.2**

note that consilience is applicable to a voting-based scheme, and present voting-based results in Appendix C, where we show the robustness of consilience as a correctness signal.

Baselines and Consilience Implementation. Our primary baselines are Self-Certainty [17], which calculates the mean confidence across the entire generated sequence and selects the completion that maximizes the confidence, and Deep-Conf [11], which proposes to use only the last-2048 tokens' confidence (DeepConf-2K). We additionally add in Deep-conf-20%, which uses the mean confidence of the last 20% of tokens. To align with the baselines, we also use two variants, absolute tokens and dynamic percentages, to determine the initial and final window sizes (W) for consilience computation. We always skip the first 5% of tokens as mentioned in Section 3.4 to focus on confidence computation. To compute token-level confidence, we use $K = 5$ top log-probabilities. We choose a small K for two reasons: (i) commercial inference APIs typically expose only the top-20 log-probabilities, so a small-K estimator makes consilience deployable on such models; and (ii) the top candidates already carry the bulk of the probability mass—on GPT-OSS-20B/LCB the top-5 tokens capture 93.9% of mass per step—so $K = 5$ suffices while avoiding the large log-prob payload of full-vocabulary scoring.

4.2 Consilience Performance

To answer **Q1**, Table 2 presents the top-1 selection accuracy across the LiveCodeBench, HMMT, and GPQA datasets. We calculate the consilience score from Equation 4 using a fixed penalty multiplier of $\alpha = 3$. The boundary window itself is evaluated either dynamically as 20% of the sequence length (Cns) or as an absolute threshold of 2048 tokens (Cns-2K), aligning with the baselines. The -think variants apply this calculation exclusively to the isolated reasoning trace, whereas -full evaluates the entire generated output as an ablation study, thinking isolation details can be found in Appendix B. For Qwen on GPQA, because the Y_{answer} is just a boxed character, we only report the full version.

The Failure of Simple Maximization. The most surprising results empirically validate our analysis regarding the "long-tail trap" of maximization-only confidence metrics. Across multiple models and datasets, applying these metrics without voting frequently results in catastrophic performance degradation compared to random selection, Pass@1. For instance, on the HMMT benchmark, applying Self-Certainty to OSS-20B drops performance from 76.5% down to 68.0%. Similarly, on LCB for OSS-120B, Pass@1 achieves 65.7%, while DeepConf-20% degrades to 61.9%. Without the guardrails of external verifiers, simply maximizing sequence-level confidence makes selecting confidently hallucinated, overfitted trajectories inevitable.

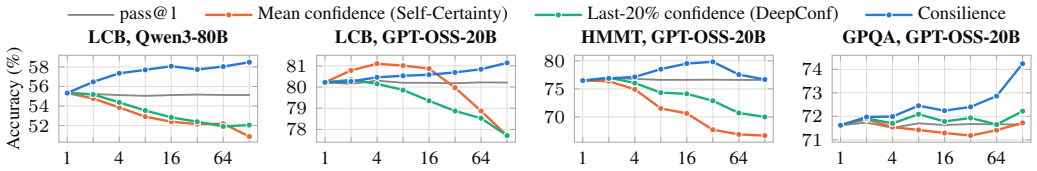


Figure 2: Scaling pattern on different models and datasets (x-axis: number of samples n). Consilience can scale in a robust fashion, while other metrics sometimes diminish.

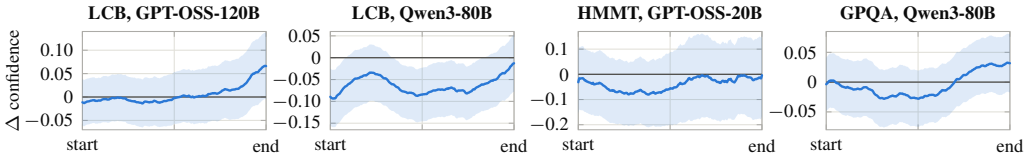


Figure 3: Difference between mean confidence trajectory of correct and incorrect completions on hard problems (<50% Pass@1). Completions show a consistent converging pattern, where the initial confidence of correct completions is lower.

Consistent Outperformance of Consilience. In stark contrast, the consilience framework successfully neutralizes this degradation, consistently outperforming both Pass@1 and all confidence baselines across all datasets. Notably, consilience excels on the open-ended LCB code generation benchmark—a critical domain where traditional exact-match majority voting is inapplicable. On LCB, our Cns-think elevates OSS-120B to 69.7%, Qwen from 55.3% (Pass@1) to 60.4%, showing necessity of early low confidence. We present in Appendix G a case study comparing consilience and over-confidence examples, providing intuitve understanding of the impact of low initial confidence.

Ablation: The Necessity of Reasoning Isolation. The comparison between the `-full` and `-think` variants illustrates how to maximally concentrate the consilience signal. Both variants rely on the same underlying logic, evaluating the temporal asymmetry of confidence, and both are effective implementations of our framework. For instance, the full-sequence variant is remarkably strong for Qwen on LiveCodeBench. Qualitative investigation reveals that Qwen frequently continues its active problem-solving by writing reasoning steps as comments directly within the final code blocks, so the `-full` trajectory still captures rich information on model confidence. The core takeaway is that isolating the reasoning phase (`-think`) reliably elevates performance across the broader spectrum of models: by stripping away the often deterministic summarization of the final answer, isolation distills the true reasoning pattern from the trajectory. We therefore recommend `-think` as the general best practice for test-time selection.

4.3 Agentic Application

To evaluate consilience in complex, multi-turn environments, we integrate it into the mini-SWE-agent framework on SWE-Bench-Verified and target TTS at critical action steps: when the model attempts a large edit step (identified via bash command matching), consilience selects the best action from $n = 32$ completions sampled in parallel. Details are in Appendix I.

This targeted integration yields a strong improvement over the base model (Table 3), showing that our temporal confidence signal reliably identifies high-quality reasoning and tool use in open-ended software engineering loops where programmatic verification is unavailable. It is also efficient: applying consilience with $n = 32$ at every step would incur roughly $32\times$ the computation, whereas with our harness Qwen3 takes 18 hours versus 8 hours for the plain mini-SWE-agent on our hardware. Smarter harnesses that trigger consilience only at the highest-value steps could reduce this cost further.

4.4 Investigation on Consilience

Scaling Robustness. To answer **Q2**, Figure 2 illustrates the accuracy scaling behavior as the sample size (n) increases up to 128. Consilience generally exhibits superior and more consistent scaling

Table 4: Difficulty-stratified selection accuracy (%) on LiveCodeBench, split into equally-sized tiers by Pass@1 rate. Consilience is neutral on easy problems and improves on medium/hard tiers.

Model	Method	Easy	Medium	Hard
Qwen	Mean-conf	99.9	58.1	3.7
	Consilience	99.9	**70.3**	**8.1**
GPT-OSS-120B	Mean-conf	100.0	74.9	13.4
	Consilience	99.1	**88.6**	**17.2**

Table 5: Hyperparameter transfer. *Suggested* uses the frozen $\alpha = 3, k = 20\%$; *Selected* is the per-set grid-search optimum under 5-fold cross-validation. The last row transfers parameters chosen on LCB to HMMT.

Setting	Pass@1	Suggested	Selected
GPT-OSS-120B / LCB	65.3	70.2	72.8
Qwen / LCB	54.9	58.1	61.3
GPT-OSS-20B / LCB	80.2	81.3	83.7
GPT-OSS-20B / LCB → HMMT	75.9	78.1	80.2

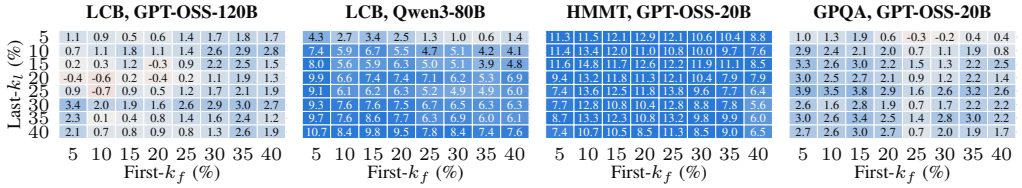


Figure 4: Difference between Consilience Accuracy (last window=$k_l\%$ and first window = $k_f\%$) and last-$k_l\%$ accuracy. Penalizing initial confidence is consistently beneficial.

properties compared to mean-conf (Self-Certainty) or last-20% (DeepConf-20%). On GPQA (OSS-20B), while both consilience and the last-20% scale positively, consilience scales at a visibly steeper rate. The advantage is most pronounced on free-form generation tasks (LCB). Using Qwen, only consilience scales positively. With OSS-20B, while the mean-conf baseline scales well initially, it quickly falls into the long-tail trap, suffering severe degradation as n increases, a similar trend observed in Chen et al. [5]. In contrast, consilience scales steadily through $n = 128$. While consilience occasionally exhibits minor degradation on HMMT for OSS-20B, it consistently preserves performance above the Pass@1 floor.

Temporal Confidence Trajectories. To answer **Q3** and validate the principle of consilience, Figure 3 normalizes the confidence within each problem, resamples confidence trajectories to the same length, plots the mean difference trajectory (correct - incorrect) on hard problems with Pass@1 rate < 50%. Across all evaluated models and datasets, the difference is negative or lowest at the beginning of the sequence and steadily rises toward the end. This empirically confirms that correct reasoning trajectories inherently possess lower initial confidence and higher relative final confidence. In some environments, the correct trajectories maintain a lower absolute confidence, further proving that an unpenalized high-confidence state actively correlates with over-confidence rather than reasoning.

Statistical Significance. This trajectory difference is statistically reliable rather than an artifact of the mean. On hard problems (Pass@1 $< 20\%$), consilience separates correct from incorrect completions with AUROC 0.61–0.62 (non-overlapping 95% CIs), whereas mean confidence is at chance (0.47–0.50). A paired test of the within-problem rise $\Delta = C_{final} - C_{initial}$ is significant on both LiveCodeBench panels ($p = 0.011, 0.018$; $d_z \approx 0.35$) and positive on GPQA (Wilcoxon $p = 0.042$). Full tables and test details are in Appendix D.

Difficulty Stratification. The trajectory analysis predicts that consilience should help precisely where mean-confidence maximization fails. We verify this by splitting each dataset into three equally-sized difficulty tiers by Pass@1 rate and reporting mean-confidence versus consilience selection accuracy in Table 4. On easy problems, Pass@1 is already near-perfect and consilience is neutral—matching the confidence baseline (99.9 vs. 99.9 for Qwen; 99.1 vs. 100.0 for GPT-OSS-120B) because all completions in the pool exhibit similarly high initial confidence, leaving no signal to penalize. The gains concentrate on medium and hard problems (up to +13.7 on medium), precisely where premature convergence is the dominant failure mode. This confirms that the initial-confidence penalty is harmless where confidence maximization already succeeds, and beneficial where it does not, directly addressing when high prefix confidence should be rewarded rather than penalized.

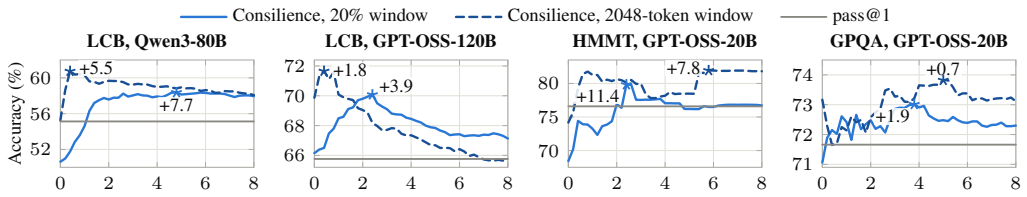


Figure 5: Consilience performance with different α. Stars mark the maximum performance improvement over $\alpha = 0$, and $\alpha > 0$ systematically brings performance boosts.

4.5 Robustness of Consilience

We answer **Q4** by analyzing sensitivity to the two primary hyperparameters: the boundary window sizes and the penalty multiplier α.

Window Size Robustness. While we use the same window size in Section 3.4, consilience can naturally be extended to different initial and last window sizes. We evaluated the performance delta between using our consilience score and using the corresponding last-k% across varying initial and final window percentages in Figure 4. The resulting heatmaps reveal that incorporating the initial confidence penalty consistently yields a positive performance boost across a massive combinatorial range of percentages. A generic robust range across model and data is 20% to 30%.

The Penalty Multiplier (α). We sweep α from 0 to 8 in Figure 5. A proper $\alpha > 0$ yields a positive gain that consistently outperforms the pass@1 baseline, and we advise $\alpha \in [2, 5]$. A weight above 1 is needed because autoregressive confidence naturally starts low and ends high, so the initial-exploration penalty must be scaled up to balance the final convergence score. Performance degrades when α becomes excessively large, which illustrates the dual nature of our theory: early exploration is necessary, but final convergence is equally critical. Over-penalizing initial confidence ignores convergence and selects from both Category 3 (consilience) and Category 4 (total confusion) in Table 1.

Selection Protocol and Transfer. To make the selection protocol explicit and rule out per-benchmark tuning, we fixed $\alpha = 3$, window $k = 20\%$, and the 5% skipped prefix on a single development setting (Qwen on LiveCodeBench) *before* evaluating any other benchmark; these frozen values are the reported `Cns` configuration in Table 2. To confirm these are not over-tuned, we additionally run a grid search ($\alpha \in \{1, \ldots, 5\}$, $k \in \{5\%, \ldots, 30\%\}$) under 5-fold cross-validation—selecting hyperparameters on the training fold and evaluating on the held-out fold—and a cross-dataset transfer where parameters chosen on GPT-OSS-20B/LCB are applied unchanged to HMMT (Table 5). The frozen configuration captures 72–78% of the per-set optimal gain, and transfers across datasets without re-tuning (+2.2 over Pass@1 on HMMT), suggesting a practical recipe: curate a small seed set per model to fix hyperparameters, then apply them to unseen data.

5 Conclusion, Limitations, and Future Work

We introduced consilience, a verifier-free test-time scaling framework that replaces confidence maximization with a temporal signal: low confidence while exploring, high confidence at convergence. It neutralizes confidently wrong selections and achieves significant margins over existing baselines, particularly on open-ended code generation, where exact-match majority voting is not applicable. More broadly, consilience instantiates a general test-time-scaling principle, that exploration should be valued rather than penalized. Our score is one efficient, zero-overhead instantiation of it, and a fixed heuristic rather than a universal rule: some settings may admit multiple valid confidence trajectories. Our agentic integration is also a coarse, keyword-triggered harness, which limits efficiency. Future directions include learned or judge-based selectors that evaluate the exploratory diversity of the reasoning itself; using consilience to select among multiple shallow trees under a fixed budget, mitigating the outsized influence of early sampling in tree search; smarter agentic harnesses; and, since consilience is an entirely internal measure of reasoning quality, an intrinsic reward for verifier-free agentic reinforcement learning.

Acknowledgments

The authors would like to thank Narayanan Sadagopan and Vijay Lingam for helpful feedback and discussion.

References

[1] Shivam Agarwal, Zimin Zhang, Lifan Yuan, Jiawei Han, and Hao Peng. The unreasonable effectiveness of entropy minimization in llm reasoning, 2025. URL https://arxiv.org/abs/2505.15134.

[2] Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. Matharena: Evaluating llms on uncontaminated math competitions. *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmark*, 2025.

[3] Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of thoughts: Solving elaborate problems with large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17682–17690, March 2024. ISSN 2159-5399. doi: 10.1609/aaai.v38i16.29720. URL http://dx.doi.org/10.1609/aaai.v38i16.29720.

[4] Ruisheng Cao, Mouxiang Chen, Jiawei Chen, Zeyu Cui, Yunlong Feng, Binyuan Hui, Yuheng Jing, Kaixin Li, Mingze Li, Junyang Lin, Zeyao Ma, Kashun Shum, Xuwu Wang, Jinxi Wei, Jiaxi Yang, Jiajun Zhang, Lei Zhang, Zongmeng Zhang, Wenting Zhao, and Fan Zhou. Qwen3-coder-next technical report, 2026. URL https://arxiv.org/abs/2603.00729.

[5] Lingjiao Chen, Jared Quincy Davis, Boris Hanin, Peter Bailis, Ion Stoica, Matei Zaharia, and James Zou. Are more llm calls all you need? towards scaling laws of compound inference systems, 2024. URL https://arxiv.org/abs/2403.02419.

[6] Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. Universal self-consistency for large language model generation, 2023. URL https://arxiv.org/abs/2311.17311.

[7] Daixuan Cheng, Shaohan Huang, Xuekai Zhu, Bo Dai, Wayne Xin Zhao, Zhenliang Zhang, and Furu Wei. Reasoning with exploration: An entropy perspective, 2025. URL https://arxiv.org/abs/2506.14758.

[8] Jinhao Duan, Hao Cheng, Shiqi Wang, Alex Zavalny, Chenan Wang, Renjing Xu, Bhavya Kailkhura, and Kaidi Xu. Shifting attention to relevance: Towards the predictive uncertainty quantification of free-form large language models, 2024. URL https://arxiv.org/abs/2307.01379.

[9] Ekaterina Fadeeva, Aleksandr Rubashevskii, Artem Shelmanov, Sergey Petrakov, Haonan Li, Hamdy Mubarak, Evgenii Tsymbalov, Gleb Kuzmin, Alexander Panchenko, Timothy Baldwin, Preslav Nakov, and Maxim Panov. Fact-checking the output of large language models via token-level uncertainty quantification, 2024. URL https://arxiv.org/abs/2403.04696.

[10] Tairan Fu, Javier Conde, Gonzalo Martinez, Maria Grandury, and Pedro Reviriego. Multiple choice questions: Reasoning makes large language models (llms) more self-confident, specially when they are wrong. *IEEE Intelligent Systems*, page 1–10, 2026. ISSN 1941-1294. doi: 10.1109/mis.2026.3660833. URL http://dx.doi.org/10.1109/MIS.2026.3660833.

[11] Yichao Fu, Xuewei Wang, Yuandong Tian, and Jiawei Zhao. Deep think with confidence, 2025. URL https://arxiv.org/abs/2508.15260.

[12] Jingchu Gai, Guanning Zeng, Christina Baek, Chen Wu, J. Zico Kolter, Andrej Risteski, and Aditi Raghunathan. Understanding and mitigating premature confidence for better llm reasoning, 2026. URL https://arxiv.org/abs/2605.24396.

[13] Jiahui Geng, Fengyu Cai, Yuxia Wang, Heinz Koeppl, Preslav Nakov, and Iryna Gurevych. A survey of confidence estimation and calibration in large language models. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6577–6595, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.366. URL https://aclanthology.org/2024.naacl-long.366/.

[14] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, Sept 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

[15] Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code, 2024. URL https://arxiv.org/abs/2403.07974.

[16] Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues?, 2024. URL https://arxiv.org/abs/2310.06770.

[17] Zhewei Kang, Xuandong Zhao, and Dawn Song. Scalable best-of-n selection for large language models via self-certainty, 2025. URL https://arxiv.org/abs/2502.18581.

[18] Ali Keramati, Justin Cheok, Jacob Horne, and Mark Warschauer. Early-token confidence predicts reasoning quality in multi-agent llm debate, 2026. URL https://arxiv.org/abs/2606.10307.

[19] Joongwon Kim, Wannan Yang, Kelvin Niu, Hongming Zhang, Yun Zhu, Eryk Helenowski, Ruan Silva, Zhengxing Chen, Srinivasan Iyer, Manzil Zaheer, Daniel Fried, Hannaneh Hajishirzi, Sanjeev Arora, Gabriel Synnaeve, Ruslan Salakhutdinov, and Anirudh Goyal. Scaling test-time compute for agentic coding, 2026. URL https://arxiv.org/abs/2604.16529.

[20] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion

Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019.

[21] Hung Le, Yue Wang, Akhilesh Deepak Gotmare, Silvio Savarese, and Steven C. H. Hoi. Coderl: Mastering code generation through pretrained models and deep reinforcement learning, 2022. URL https://arxiv.org/abs/2207.01780.

[22] Ang Li, Zhihang Yuan, Yang Zhang, Shouda Liu, and Yisen Wang. Know when to explore: Difficulty-aware certainty as a guide for llm reinforcement learning, 2025. URL https://arxiv.org/abs/2509.00125.

[23] Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Xinglin Wang, Bin Sun, Heda Wang, and Kan Li. Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning, 2024. URL https://arxiv.org/abs/2401.10480.

[24] Baohao Liao, Xinyi Chen, Sara Rajaee, Yuhui Xu, Christian Herold, Anders Søgaard, Maarten de Rijke, and Christof Monz. Lost at the beginning of reasoning, 2025. URL https://arxiv.org/abs/2506.22058.

[25] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023. URL https://arxiv.org/abs/2305.20050.

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

[27] Zhenjiang Mao, Artem Bisliouk, Rohith Reddy Nama, and Ivan Ruchkin. Temporalizing confidence: Evaluation of chain-of-thought reasoning with signal temporal logic, 2025. URL https://arxiv.org/abs/2506.08243.

[28] OpenAI, :, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, Ally Bennett, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrew Duberstein, Andrew Kondrich, Andrey Mishchenko, Andy Applebaum, Angela Jiang, Ashvin Nair, Barret Zoph, Behrooz Ghorbani, Ben Rossen, Benjamin Sokolowsky, Boaz Barak, Bob McGrew, Borys Minaiev, Botao Hao, Bowen Baker, Brandon Houghton, Brandon McKinzie, Brydon Eastman, Camillo Lugaresi, Cary Bassin, Cary Hudson, Chak Ming Li, Charles de Bourcy, Chelsea Voss, Chen Shen, Chong Zhang, Chris Koch, Chris Orsinger, Christopher Hesse, Claudia Fischer, Clive Chan, Dan Roberts, Daniel Kappler, Daniel Levy, Daniel Selsam, David Dohan, David Farhi, David Mely, David Robinson, Dimitris Tsipras, Doug Li, Dragos Oprica, Eben Freeman, Eddie Zhang, Edmund Wong, Elizabeth Proehl, Enoch Cheung, Eric Mitchell, Eric Wallace, Erik Ritter, Evan Mays, Fan Wang, Felipe Petroski Such, Filippo Raso, Florencia Leoni, Foivos Tsimpourlas, Francis Song, Fred von Lohmann, Freddie Sulit, Geoff Salmon, Giambattista Parascandolo, Gildas Chabot, Grace Zhao, Greg Brockman, Guillaume Leclerc, Hadi Salman, Haiming Bao, Hao Sheng, Hart Andrin, Hessam Bagherinezhad, Hongyu Ren, Hunter Lightman, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian Osband, Ignasi Clavera Gilaberte, Ilge Akkaya, Ilya Kostrikov, Ilya Sutskever, Irina Kofman, Jakub Pachocki, James Lennon, Jason Wei, Jean Harb, Jerry Twore, Jiacheng Feng, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joaquin Quiñonero Candela, Joe Palermo, Joel Parish, Johannes Heidecke, John Hallman, John Rizzo, Jonathan Gordon, Jonathan Uesato, Jonathan Ward, Joost Huizinga, Julie Wang, Kai Chen, Kai Xiao, Karan Singhal, Karina Nguyen, Karl Cobbe, Katy Shi, Kayla Wood, Kendra Rimbach, Keren Gu-Lemberg, Kevin Liu, Kevin Lu, Kevin Stone, Kevin Yu, Lama Ahmad, Lauren Yang, Leo Liu, Leon Maksin, Leyton Ho, Liam Fedus, Lilian Weng, Linden Li, Lindsay McCallum, Lindsey Held, Lorenz Kuhn, Lukas Kondraciuk, Lukasz Kaiser, Luke Metz, Madelaine Boyd, Maja Trebacz, Manas Joglekar, Mark Chen, Marko Tintor, Mason Meyer, Matt Jones, Matt Kaufer, Max Schwarzer, Meghan Shah, Mehmet Yatbaz, Melody Y. Guan, Mengyuan Xu, Mengyuan Yan, Mia Glaese, Mianna Chen, Michael Lampe, Michael Malek, Michele Wang, Michelle Fradin, Mike McClay, Mikhail Pavlov, Miles Wang, Mingxuan

Wang, Mira Murati, Mo Bavarian, Mostafa Rohaninejad, Nat McAleese, Neil Chowdhury, Neil Chowdhury, Nick Ryder, Nikolas Tezak, Noam Brown, Ofir Nachum, Oleg Boiko, Oleg Murk, Olivia Watkins, Patrick Chao, Paul Ashbourne, Pavel Izmailov, Peter Zhokhov, Rachel Dias, Rahul Arora, Randall Lin, Rapha Gontijo Lopes, Raz Gaon, Reah Miyara, Reimar Leike, Renny Hwang, Rhythm Garg, Robin Brown, Roshan James, Rui Shu, Ryan Cheu, Ryan Greene, Saachi Jain, Sam Altman, Sam Toizer, Sam Toyer, Samuel Miserendino, Sandhini Agarwal, Santiago Hernandez, Sasha Baker, Scott McKinney, Scottie Yan, Shengjia Zhao, Shengli Hu, Shibani Santurkar, Shraman Ray Chaudhuri, Shuyuan Zhang, Siyuan Fu, Spencer Papay, Steph Lin, Suchir Balaji, Suvansh Sanjeev, Szymon Sidor, Tal Broda, Aidan Clark, Tao Wang, Taylor Gordon, Ted Sanders, Tejal Patwardhan, Thibault Sottiaux, Thomas Degry, Thomas Dimson, Tianhao Zheng, Timur Garipov, Tom Stasi, Trapit Bansal, Trevor Creech, Troy Peterson, Tyna Eloundou, Valerie Qi, Vineet Kosaraju, Vinnie Monaco, Vitchyr Pong, Vlad Fomenko, Weiyi Zheng, Wenda Zhou, Wes McCabe, Wojciech Zaremba, Yann Dubois, Yinghai Lu, Yining Chen, Young Cha, Yu Bai, Yuchen He, Yuchen Zhang, Yunyun Wang, Zheng Shao, and Zhuohan Li. Openai o1 system card, 2024. URL https://arxiv.org/abs/2412.16720.

[29] OpenAI, :, Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K. Arora, Yu Bai, Bowen Baker, Haiming Bao, Boaz Barak, Ally Bennett, Tyler Bertao, Nivedita Brett, Eugene Brevdo, Greg Brockman, Sebastien Bubeck, Che Chang, Kai Chen, Mark Chen, Enoch Cheung, Aidan Clark, Dan Cook, Marat Dukhan, Casey Dvorak, Kevin Fives, Vlad Fomenko, Timur Garipov, Kristian Georgiev, Mia Glaese, Tarun Gogineni, Adam Goucher, Lukas Gross, Katia Gil Guzman, John Hallman, Jackie Hehir, Johannes Heidecke, Alec Helyar, Haitang Hu, Romain Huet, Jacob Huh, Saachi Jain, Zach Johnson, Chris Koch, Irina Kofman, Dominik Kundel, Jason Kwon, Volodymyr Kyrylov, Elaine Ya Le, Guillaume Leclerc, James Park Lennon, Scott Lessans, Mario Lezcano-Casado, Yuanzhi Li, Zhuohan Li, Ji Lin, Jordan Liss, Lily, Liu, Jiancheng Liu, Kevin Lu, Chris Lu, Zoran Martinovic, Lindsay McCallum, Josh McGrath, Scott McKinney, Aidan McLaughlin, Song Mei, Steve Mostovoy, Tong Mu, Gideon Myles, Alexander Neitz, Alex Nichol, Jakub Pachocki, Alex Paino, Dana Palmie, Ashley Pantuliano, Giambattista Parascandolo, Jongsoo Park, Leher Pathak, Carolina Paz, Ludovic Peran, Dmitry Pimenov, Michelle Pokrass, Elizabeth Proehl, Huida Qiu, Gaby Raila, Filippo Raso, Hongyu Ren, Kimmy Richardson, David Robinson, Bob Rotsted, Hadi Salman, Suvansh Sanjeev, Max Schwarzer, D. Sculley, Harshit Sikchi, Kendal Simon, Karan Singhal, Yang Song, Dane Stuckey, Zhiqing Sun, Philippe Tillet, Sam Toizer, Foivos Tsimpourlas, Nikhil Vyas, Eric Wallace, Xin Wang, Miles Wang, Olivia Watkins, Kevin Weil, Amy Wendling, Kevin Whinnery, Cedric Whitney, Hannah Wong, Lin Yang, Yu Yang, Michihiro Yasunaga, Kristen Ying, Wojciech Zaremba, Wenting Zhan, Cyril Zhang, Brian Zhang, Eddie Zhang, and Shengjia Zhao. gpt-oss-120b & gpt-oss-20b model card, 2025. URL https://arxiv.org/abs/2508.10925.

[30] Matthias Otth, Jonas Hübotter, Ido Hakimi, and Andreas Krause. Maximizing prefix-confidence at test-time efficiently improves mathematical reasoning, 2025. URL https://arxiv.org/abs/2507.18122.

[31] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark, 2023. URL https://arxiv.org/abs/2311.12022.

[32] William Saunders, Catherine Yeh, Jeff Wu, Steven Bills, Long Ouyang, Jonathan Ward, and Jan Leike. Self-critiquing models for assisting human evaluators, 2022. URL https://arxiv.org/abs/2206.05802.

[33] Amrith Setlur, Nived Rajaraman, Sergey Levine, and Aviral Kumar. Scaling test-time compute without verification or rl is suboptimal, 2025. URL https://arxiv.org/abs/2502.12118.

[34] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters, 2024. URL https://arxiv.org/abs/2408.03314.

[35] Siddarth Venkatraman, Vineet Jain, Sarthak Mittal, Vedant Shah, Johan Obando-Ceron, Yoshua Bengio, Brian R. Bartoldson, Bhavya Kailkhura, Guillaume Lajoie, Glen Berseth, Nikolay Malkin, and Moksh Jain. Recursive self-aggregation unlocks deep thinking in large language models, 2026. URL https://arxiv.org/abs/2509.26626.

[36] Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.510. URL https://aclanthology.org/2024.acl-long.510/.

[37] Xinglin Wang, Yiwei Li, Shaoxiong Feng, Peiwen Yuan, Yueqi Zhang, Jiayi Shi, Chuyi Tan, Boyuan Pan, Yao Hu, and Kan Li. Every rollout counts: Optimal resource allocation for efficient test-time scaling, 2025. URL https://arxiv.org/abs/2506.15707.

[38] 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, 2023. URL https://arxiv.org/abs/2203.11171.

[39] Yu Wang, Minghao Liu, Jiayun Wang, Jinrui Huang, Ankit Shah, and Wei Wei. Inference time optimization with confidence dynamics, 2026. URL https://arxiv.org/abs/2605.25244.

[40] Can Xie, Ruotong Pan, Xiangyu Wu, Yunfei Zhang, Jiayi Fu, Tingting Gao, and Guorui Zhou. Unlocking exploration in rlvr: Uncertainty-aware advantage shaping for deeper reasoning, 2026. URL https://arxiv.org/abs/2510.10649.

[41] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

[42] John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik R Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2405.15793.

[43] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models, 2023. URL https://arxiv.org/abs/2305.10601.

[44] Dongkeun Yoon, Seungone Kim, Sohee Yang, Sunkyoung Kim, Soyeon Kim, Yongil Kim, Eunbi Choi, Yireun Kim, and Minjoon Seo. Reasoning models better express their confidence, 2025. URL https://arxiv.org/abs/2505.14489.

[45] Wenhao Zeng, Yaoning Wang, Chao Hu, Yuling Shi, Chengcheng Wan, Hongyu Zhang, and Xiaodong Gu. Pruning the unsurprising: Efficient llm reasoning via first-token surprisal, 2026. URL https://arxiv.org/abs/2508.05988.

[46] Tianyu Zheng, Ge Zhang, Tianhao Shen, Xueling Liu, Bill Yuchen Lin, Jie Fu, Wenhu Chen, and Xiang Yue. Opencodeinterpreter: Integrating code generation with execution and refinement, 2025. URL https://arxiv.org/abs/2402.14658.

[47] Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, Biqing Qi, Youbang Sun, Zhiyuan Ma, Lifan Yuan, Ning Ding, and Bowen Zhou. Ttrl: Test-time reinforcement learning, 2025. URL https://arxiv.org/abs/2504.16084.

Appendix

A Dataset Details

To rigorously evaluate the efficacy of the consilience metric, we selected a diverse suite of benchmarks encompassing advanced mathematics, graduate-level science, and open-ended software engineering. These datasets were specifically chosen for their high difficulty and, critically, their requirement for complex, multi-step reasoning where standard heuristics often fail. Our script and agentic implementation can be found here: https://github.com/LechengKong/consilience.

- **HMMT25-Feb** [2]: The Harvard-MIT Mathematics Tournament (February 2025) comprises exceptionally difficult competition-level mathematics problems. It demands rigorous, multi-step logical deductions and creative problem-solving, making it a robust testbed for advanced mathematical reasoning. We use the provided script in MathArena benchmark to compare and group generated Latex answers.
- **AIME25** [2]: The 2025 American Invitational Mathematics Examination consists of highly challenging, integer-answer mathematics problems. It rigorously evaluates a model's capacity for extended algebraic, geometric, and combinatorial reasoning under strict constraints. We use the provided script in MathArena benchmark to compare and group generated Latex answers.
- **GPQA-Diamond** [31]: GPQA is a highly challenging dataset of graduate-level questions in physics, biology, and chemistry. The "Diamond" subset is specifically curated to be difficult even for highly skilled human experts, providing a rigorous evaluation of deep domain knowledge and complex scientific reasoning.
- **LiveCodeBench-v6** [15]: A dynamic code generation benchmark utilizing recent contest problems (e.g., from LeetCode, AtCoder) to mitigate training data contamination. It requires models to generate expressive, free-form programmatic solutions, serving as our primary environment to evaluate selection metrics where exact-match majority voting is inapplicable. For this LiveCodeBench, we use only problem from v6 release, excluding problems from v1-v5 release to control the effect of data contamination and focus on the harder problems. We use the evalutaion script supplied with the benchmark to evaluate the results.
- **SWE-bench-Verified** [16]: A framework for evaluating large language models on real-world software engineering tasks. It requires models to act as autonomous agents, navigating complex code repositories and generating multi-step, open-ended edits to resolve actual GitHub issues. We use the minimal bash-only agent, mini-SWE-agent [42], provided along with the benchmark to focus on evaluating the power of consilience.
- **Natural Questions** [20]: An open-domain common-sense question answering benchmark of real Google search queries with answers grounded in Wikipedia. We use it (a 2K-question subsample) to evaluate consilience on a data distribution outside the mathematics and code domains; see Appendix E.

B Model Configuration

All model inference is conducted using the vLLM framework hosted on a node with 4$\times$ NVIDIA L40S (48GB) GPUs. Unless otherwise specified, we utilize the default generation configurations provided by the models' respective HuggingFace repositories. The specific sampling parameters and reasoning effort configurations are detailed in Table 6.

Due to computational resource constraints, we restrict the "high" reasoning effort mode to the GPT-OSS-20B model, while utilizing the "medium" setting for the larger 120B variant. This variation in configuration also serves to demonstrate the robust applicability of the consilience metric across distinctly different thinking patterns and generation lengths. We observe that under the "high" thinking mode, GPT-OSS-20B occasionally fails to complete its answer within the 130k maximum token budget. To ensure a strictly fair evaluation, any such truncated responses are filtered out of the generation pool prior to computing Pass@1 and evaluating all other selection baselines.

Implementation of Reasoning Isolation. To execute the reasoning-phase isolation, we employ model-specific structural delimiters to parse the reasoning trace (Y_{reason}) from the final answer

Table 6: Sampling configurations and thinking parameters for the evaluated models. Default Hugging Face generation parameters were used where applicable.

Model	Temperature	Top-p	Top-k	Max Len.	Reasoning Effort
GPT-OSS-120B	1.0	1.0	40	130k	Medium
GPT-OSS-20B	1.0	1.0	40	130k	High
Qwen3-Next-80B	0.7	0.8	20	130k	N/A
DeepSeek-R1-Qwen-7B	0.6	0.95	40	130k	N/A

Table 7: Accuracy (%) of confidence metrics integrated with Borda Count voting.

		HMMT		GPQA		AIME	
Method	**p**	20B	Qwen	20B	Qwen	20B	Qwen
Pass@1	—	76.6±0.3	52.5±0.2	71.7±0.2	60.5±0.1	93.0±0.0	69.4±0.6
Self-Consistency	0	92.7±1.3	57.0±1.0	90.3±0.2	68.0±0.9	96.7±0.0	76.0±1.3
Self-Certainty	0.5	**93.3±0.0**	57.3±1.3	90.2±0.3	67.9±1.5	96.7±0.0	73.7±1.0
	1	90.0±0.0	57.0±2.3	90.1±0.2	67.9±1.1	96.7±0.0	73.3±0.0
	2	90.0±0.0	54.7±3.4	89.7±0.2	68.1±1.2	93.3±0.0	73.7±1.0
	4	90.0±0.0	53.0±3.8	89.0±0.3	66.6±1.5	93.3±0.0	75.0±1.7
DeepConf-2K	0.5	89.7±1.0	58.0±1.6	**90.3±0.2**	67.9±1.5	96.7±0.0	74.3±1.5
	1	90.0±0.0	58.7±1.6	**90.3±0.2**	67.5±1.4	96.7±0.0	74.3±2.1
	2	90.0±0.0	59.0±1.5	90.2±0.2	67.6±1.4	96.7±0.0	77.0±2.3
	4	86.3±1.0	57.0±1.8	89.4±0.4	66.2±1.5	96.7±0.0	77.0±2.3
DeepConf-20%	0.5	90.0±0.0	57.7±1.5	**90.3±0.2**	68.5±1.2	96.7±0.0	75.3±1.6
	1	90.0±0.0	58.7±1.6	90.1±0.2	68.3±1.2	96.7±0.0	76.0±3.3
	2	90.0±0.0	**59.3±1.3**	89.1±0.5	67.9±1.0	96.7±0.0	79.0±4.0
	4	86.7±0.0	58.0±1.6	88.6±0.4	67.6±1.0	96.7±0.0	81.3±3.4
Cns-20%	0.5	**93.3±0.0**	57.7±1.5	**90.3±0.2**	**68.6±1.2**	96.7±0.0	75.7±1.5
	1	90.0±0.0	58.7±1.6	90.2±0.2	68.3±1.1	96.7±0.0	76.7±2.6
	2	86.7±0.0	**59.3±1.3**	90.1±0.3	68.1±1.3	96.7±0.0	80.0±3.7
	4	86.7±0.0	57.7±2.1	89.7±0.4	67.6±1.2	96.7±0.0	**82.7±2.5**
Cns-2048	0.5	89.3±1.3	58.0±1.6	**90.3±0.3**	67.3±1.1	96.7±0.0	76.3±1.0
	1	89.7±1.0	58.3±1.7	90.1±0.3	66.8±1.1	96.7±0.0	77.3±2.0
	2	89.7±1.0	**59.3±1.3**	89.7±0.5	65.9±1.0	96.7±0.0	80.0±1.5
	4	86.3±1.0	58.0±1.6	89.0±0.6	65.1±1.0	96.7±0.0	81.7±1.7

(Y_{answer}). For the GPT-OSS models, we extract tokens preceding the `<|channel|>final` delimiter. For DeepSeek-R1, we truncate the sequence at the standard `</think>` tag. For Qwen3-Next, which lacks a dedicated end-of-thought token in its instruction format, we apply task-specific structural heuristics: on LiveCodeBench, we extract the sequence prior to the first ```` ```python ```` block; on HMMT, we split the trace at the first occurrence of the `\boxed` macro, because the model generate large trunk of justification texts and reiterates on the answer; on GPQA, and AIME, we use the full trajectory, because the answer is just a single interger or character; on Swe-bench, we split the trace by ```` ```bash ````. We use these specialized delimiters because we are following the original prompt provided by the benchmark. In reality or in production, one can always prompt non-thinking model to output a "Final Answer" token before the answer, which makes thinking isolation simple and unified.

C Voting-based Results

While the primary focus of the consilience metric is to enable verifier-free selection in open-ended domains where exact-match consensus is impossible, it can also be seamlessly integrated with traditional majority voting techniques on structured tasks. To evaluate this, we combine the confidence

metrics with voting on the HMMT, GPQA, and AIME datasets, where answers can be strictly extracted and matched.

Because different confidence metrics (e.g., DeepConf vs. Consilience) operate on entirely different mathematical scales and distributions, directly aggregating their raw scores is unstable. To ensure a fair and robust comparison, we utilize **Borda Count voting**. Under this scheme, instead of aggregating raw scores, each metric ranks the N sampled completions. For a pool of N candidates, the top-ranked completion receives N points, the second receives $N-1$ points, and so on. A hyperparameter p controls the exponent of the rank weighting during aggregation.

Specifically, for a set of N sampled completions $\{y_1, y_2, \ldots, y_N\}$ for a given prompt, let $R(y_i) \in \{1, \ldots, N\}$ denote the rank of completion y_i based on the chosen selection metric (where 1 represents the highest metric score). Let $A(y_i)$ represent the extracted final answer from completion y_i. The total Borda score $B(a)$ for a unique answer candidate a is computed as:

$$B(a) = \sum_{i=1}^{N} \mathbb{I}(A(y_i) = a) \cdot (N - R(y_i) + 1)^p \tag{5}$$

where $\mathbb{I}$ is the indicator function. The final selected answer is simply the candidate that maximizes this score: $\arg\max_a B(a)$.

The results in Table 7 demonstrate that under a voting scheme, almost all test-time scaling methods successfully and heavily outperform the greedy Pass@1 baseline. Because the standard Self-Consistency baseline is already exceptionally high on these structured, closed-form tasks (e.g., 96.7% on AIME for OSS-20B), the margins between the various confidence metrics become naturally compressed.

Despite this ceiling effect, the consilience metric still manages to slightly outperform the baselines, achieving the peak scores across almost all model-dataset pairs. This indicates that even when exact-match consensus is available, the temporal trajectory evaluated by consilience provides a superior, additive signal of reasoning quality. However, we emphasize that while consilience is still competitive in these environments, its primary structural advantage remains its ability to unlock these exact types of scaling margins in free-form, un-votable domains like LiveCodeBench and SWE-bench.

D Statistical Significance of the Confidence Trajectory

We provide the full statistical analysis supporting the trajectory difference reported in Section 4.4. Both analyses are computed on the existing completion pools, requiring no additional generation.

AUROC. Table 8 reports the area under the ROC curve for separating correct from incorrect completions on hard problems (Pass@1 $< 20\%$), using either mean confidence or the consilience score as the ranking signal. Mean confidence is at chance (0.47–0.50), while consilience is a reliable discriminator (0.61–0.62); the 95% confidence intervals are non-overlapping, confirming that the separation is not attributable to sampling noise.

Table 8: AUROC (95% CI) for separating correct from incorrect completions on hard problems (Pass@1 $< 20\%$).

Metric	Qwen	GPT-OSS-120B
Mean confidence	0.50 [0.48, 0.52]	0.47 [0.44, 0.50]
Consilience	**0.61 [0.59, 0.64]**	**0.62 [0.59, 0.65]**

Paired test of the trajectory rise. The $\pm$SEM band in Figure 3 reflects *between*-problem variance, whereas the initial$\rightarrow$final rise is a *within*-problem effect. We therefore test the per-problem quantity $\Delta = C_{final} - C_{initial}$ (last-20% minus first-20%, the same window used by our selection method) with a paired test across hard problems, reported in Table 9. The rise is significant on both LiveCodeBench panels (paired t: $p = 0.011, 0.018$; Wilcoxon $p = 0.011, 0.018$; Cohen's $d_z = 0.37, 0.35$) and positive on GPQA (Wilcoxon $p = 0.042$). The HMMT hard subset ($n = 7$) is underpowered for a significance claim. Because the 20% window is fixed by the method design, this test is pre-specified rather than chosen post-hoc to maximize significance.

Table 9: Paired test of the per-problem rise $\Delta = C_{final} - C_{initial}$ (last-20% minus first-20%) on hard problems.

Panel	n	**mean** Δ	**95% CI**	**paired** t (p)	**Cohen's** d_z
LCB / GPT-OSS-120B	37	$+0.077$	$[+0.007, +0.148]$	$p = 0.011$	$+0.37$
LCB / Qwen	56	$+0.077$	$[+0.018, +0.136]$	$p = 0.018$	$+0.35$
GPQA / Qwen	72	$+0.036$	$[-0.002, +0.073]$	$p = 0.063$	$+0.22$
HMMT / GPT-OSS-20B	7	$+0.030$	$[-0.067, +0.127]$	$p = 0.48$	$+0.28$

E Practical Budget and Extended Baselines on Common-Sense QA

To evaluate consilience at a small, practical sample size, under extended baselines, and on a data distribution outside the mathematics/code domains of the main paper, we conduct an additional experiment on Natural Questions [20], a common-sense open-domain question answering benchmark. We use a 2K-question subsample and GPT-OSS-20B, with a budget of $n = 8$ generations. To ensure a fair comparison, every method is granted the same generation budget of eight total passes, allocated according to its design: consilience and DeepConf select from 8 parallel samples; Self-Refine [26] runs 8 sequential refinement iterations; and Self-Verify [32] runs 4 parallel threads each with one additional self-verification round.

Table 10: Natural Questions (2K subsample, GPT-OSS-20B) at a practical budget of $n = 8$. All methods use eight total generation passes. Wall-time is measured on identical hardware.

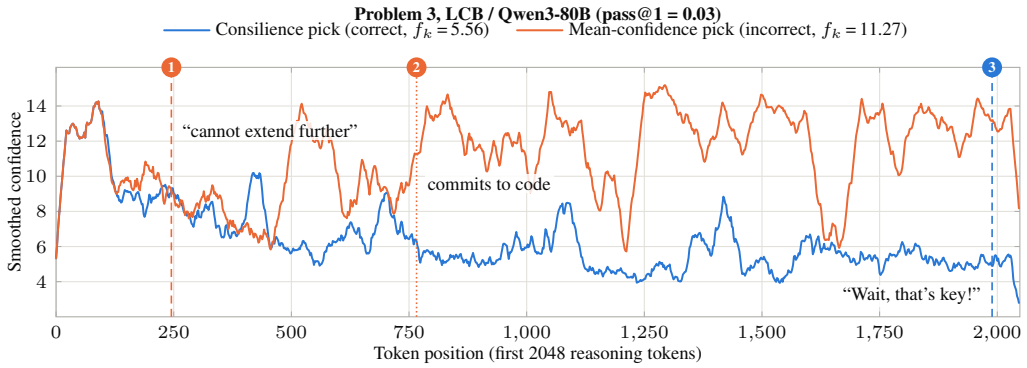
Method	**Accuracy (%)**	**Wall-time**
Pass@1	25.9	14 min
DeepConf ($n = 8$)	26.7	37 min
Consilience ($n = 8$)	**30.1**	36 min
Self-Refine (8 iter)	26.0	101 min
Self-Verify (4×2)	30.4	59 min

Table 10 reports the results. At the practical $n = 8$ budget, consilience improves over Pass@1 by $+4.2$ points and is competitive with the strongest baseline, Self-Verify (30.1 vs. 30.4), while being fully parallelizable—it completes in 36 minutes versus 59 minutes for Self-Verify and 101 minutes for the sequential Self-Refine. Self-Refine does not help on this dataset. Two points are worth noting. First, gains persist at a small budget on a distribution distinct from math and code, indicating that the consilience signal is not specific to the formal-reasoning domains of the main experiments. Second, judge- and refinement-based methods incur substantial wall-time because they cannot be fully parallelized, and they scale poorly for long free-form outputs (e.g., LiveCodeBench solutions), where feeding all candidates into a single verification call can quickly exhaust the context window. Consilience avoids this by scoring each completion independently from its own logits. We also note that consilience is complementary to judge-based selection: a judge could additionally assess the exploratory diversity of the reasoning process rather than only final-answer consensus, which we leave as future work.

F More Related Work

Verifier-based and search-based TTS. Beyond the confidence-based methods discussed in the main text, a large body of parallel TTS work relies on external signals. Programmatic verifiers such as compilers [46, 21] and formal proof checkers provide ground-truth feedback, and search algorithms combine sequential and parallel rollouts to select promising partial traces [43, 37]. On the aggregation side, Self-Consistency with early stopping [23], Process Reward Models that verify intermediate steps [36], recursive/ensemble self-aggregation [35, 19], and self-verification [32] improve robustness at the cost of extractable answers, trained verifiers, or substantial aggregation overhead. Consilience instead requires only the model's own logits and applies to free-form generation.

Using Entropy and Confidence in LLMs. In proximal area of output caliberation, confidence and entropy are also widely acknowledged as an important signal to determine output quality [13, 9, 8].



1 'Stop on duplicate' framing, token ∼245

```
the current path 4. When visiting a node: - If its value
is already in the path, we cannot extend further (since
values must be unique) - Otherwise, we can extend the path 5.
Track the maximum path length found and the minimum number
of nodes that achieve this maximum 6. For each valid path,
calculate its total length (sum of edge weights)
## Algorithm Details
1. Create adjacency list: for
```

2 Commits to implementation, token ∼766 (f_k = 11.27)

```
Implementation
```python class Solution: def longestSpecialPath(self,
edges: List[List[int]], nums: List[int]) -> List[int]:
from collections import defaultdict n = len(nums) # Build
adjacency list graph
```

3 Consilience pick: reframing at token ∼1989 (f_k = 5.56)

```
...traverse from root to current node, we maintain a set
of values in the current path. When we reach a node u with
value nums[u], if this value already exists in the current
path, then we cannot have any path from any ancestor before
the previous occurrence of this value to u (because the path
would include two copies of nums[u]). But we can have paths
from ancestors after the previous occurrence of nums[u] to
u.
Wait, that's key!
Let me think: if we have a path from root to u: [v0, v1, ...,
vk=u] If nums[u] appears at position i and also at position
j (j < k), then any path that includes both vi and vk would
have duplicate nums[u]. So the only valid paths ending
at u are those that start from an ancestor after the last
occurrence of nums[u] in the path from root to u.
Actually, no: if we
```

Figure 6: Confidence in first 2048 tokens for two completions. Consilience pick, despite lower initial confidence, spent more time on critical design choice, and lead to final correct in this particularly hard problem.

Entropy use is also adopted in reinforcement learning research, as this signal frequently serve as effective RL optimization targets. Leveraging this, works like Test-Time Reinforcement Learning [47] utilize internal metrics to lift the reliance on external verifiers, typically minimizing sequence entropy [1] to drive high-confidence consensus. Within Reinforcement Learning with Verifiable Rewards, on the other hand, researchers utilize entropy maximization as a proactive global training perturbation; frameworks such as DACE [22], UCAS [40], and Reasoning with Exploration [7] explicitly integrate entropy bonuses to penalize convergence and force diverse search paths. Zeng et al. [45] uses entropy to quantify surprise in LLM generation to effectively curate data. Our consilience framework connects to these efforts by utilizing the same internal probability distributions, but diverges significantly in both application and conceptualization. Rather than acting as a global training perturbation that steers intermediate training procedure for exploration, consilience serves as a zero-shot inference signature leading to robust reasoning process. Moreover, we demonstrate that the value of exploration is phase-dependent: high entropy is profoundly beneficial during initial hypothesis branching, but uniformly high entropy throughout a sequence signifies total confusion rather than robust exploration.

G Qualitative Analysis

Figure 6 traces the per-token confidence of two completions produced by QWEN3-80B on the problem below (pass@1 = 0.03; only 4 of 128 completions are correct). The *consilience pick* (f_k=5.56, blue) is correct; the *mean-confidence pick* (f_k=11.27, red) is not.
```

Problem (LCB #3)

Given an undirected tree rooted at node 0 with n nodes, where each node has a value nums[i] and each edge has an integer length. A *special path* is a downward path (ancestor $\rightarrow$ descendant) in which all node values are distinct. Return [max_length, min_nodes]: the length of the longest special path, and the minimum number of nodes among all such longest paths. ($n \leq 5\times10^4$, nums[i]$\leq 5\times10^4$.)

Both completions open with the same sketch: DFS from the root, maintain a set of seen values, stop when a duplicate is encountered. The mean-confidence completion accepts this framing immediately (marker 1): *"if its value is already in the path, we cannot extend further"*—and never revisits it. This *stop-on-duplicate* assumption translates directly into a `return` statement in the final code, silently pruning entire subtrees: once a node with a repeated value is reached, none of its descendants are explored or scored. The bug is fatal on any input where a long valid path passes through a node whose value already appeared higher in the tree, because the stop prevents the DFS from ever discovering that valid path. Confident in its framing, the model commits to writing code at marker 2.

The consilience completion, by contrast, spends the entire first 2,048 tokens questioning the same framing. Recognising that the path may start at *any* ancestor—not just the root—it repeatedly asks: which ancestors are actually valid starting points for a path ending at the current node? At marker 3 it finds the answer: *"we cannot have any path from any ancestor **before** the previous occurrence of `nums[u]` to u ... but we can have paths from ancestors **after** the previous occurrence ... So the only valid paths ending at u are those that start from an ancestor after the last occurrence of `nums[u]`. Wait, that's key!"* The model continues to reason for thousands of tokens before arriving at the concrete algorithm for this problem. It identifies the right question to ask, a question the over-confident completion never posed because it had already committed to code.

H Prompt for Generation

For all evaluations, we utilized the minimal and default prompt templates supplied by their respective benchmarks. We introduced only slight modifications to the instructions to enforce strict structural formatting (such as `\boxed{}` or specific Markdown delimiters) to ensure reliable, automated answer extraction. The exact system and user prompts for each dataset are detailed below.

LiveCodeBench (LCB) Prompt

System Message:
You are an expert Python programmer. Your task is to solve a coding problem by first creating a plan and then writing clean, efficient code.
IMPORTANT INSTRUCTIONS: 1. **Show Thinking** In your response, first describe your detailed step-by-step thinking process, how you understand the problem, and how you arrive at your solution. You then provide the implementation plan of your code. 2. **Clean Code Block:** After the plan, provide the complete and final code solution in a single, clean markdown block. 3. **NO THINKING IN CODE:** The code block MUST NOT contain any comments that explain your thought process or step-by-step logic.

User Message (Without Starter Code):
You will be given a question (problem specification) and will generate a correct Python program that matches the specification and passes all tests beyond provided examples in the most optimal complexity.
Question: {input_text}
First, provide your plan. Then, write code to read the inputs from stdin, solve the problem, and write the answer to stdout (do not directly test on the sample inputs). Enclose your code within a single "'python block.
"'python # YOUR CODE HERE "'

For Qwen model, despite explicit instruction to contraint its thinking outside the code block, it still tends to put big trunk of thinking process as comments in the final code.

AIME Prompt

System Message:
You are an expert mathematician solving AIME (American Invitational Mathematics Examination) problems. Provide clear step-by-step reasoning and clearly mark your final answer using \boxed{} notation.

User Message:
{question}
Please reason step by step, and put your final answer within \boxed{}.

HMMT Prompt

System Message:
You are a helpful assistant.

User Message:
Please reason step by step, and put your final answer within \boxed{}.
{problem}

GPQA Prompt

System Message:
You are an expert at answering graduate-level science questions. Answer the following multiple-choice question by analyzing each option carefully.

User Message:
{Question}
(A) {A}
(B) {B}
(C) {C}
(D) {D}
Express your final answer as the corresponding option 'A', 'B', 'C', or 'D'. Put your answer within \boxed{}.

I Agentic Harness for Consilience

In multi-step, interactive agentic environments such as SWE-bench, applying test-time scaling (TTS) and consilience selection to every generation step is both computationally inefficient and theoretically unnecessary. Many steps in an agentic trajectory are rudimentary information-gathering actions—such as listing directories or reading file chunks—that do not necessitate deep cognitive search or complex reasoning. Therefore, we constrain the application of our metric strictly to critical editing steps: the junctures where the model actually formulates the logic to modify the codebase.

To integrate consilience into a sequential workflow without derailing the agent's overarching strategy, we design a targeted interception harness. The execution proceeds as follows:

1. **Base Sampling and Detection:** For any given step, we initially sample a single standard completion. We analyze this response via keyword matching to determine if it constitutes a file-editing action (specifically checking for: `sed -i`, `cat «`, `tee `, `> /`, `patch `, or `EOF`).
2. **Conditional Triggering:** If the step is merely exploratory (no keywords detected), it is executed normally. If an editing keyword is present, and the bash command is larger then L lines, the step is flagged as a critical reasoning node. (We use $L = 40$ for GPT-OSS-120B, and $L = 100$ for Qwen3-Coder-Next).
3. **Parallel Generation and Filtering:** At a flagged node, we sample K parallel completions. To ensure the agent adheres to the established workflow and intent of the base trajectory, we filter these K candidates, retaining only those that utilize the same editing keywords identified in the initial step.

4. **Consilience Selection:** Finally, we evaluate the isolated reasoning phases of the filtered pool and apply our consilience metric to select the most structurally robust editing command for execution.

We note that this keyword-triggered interception is an intentionally coarse harness. Its primary function is to serve as a minimal viable integration to demonstrate that consilience effectively identifies high-quality reasoning in open-ended, sequential agentic tasks where exact-match voting is impossible. Developing more sophisticated, dynamic TTS triggering mechanisms for autonomous agents remains a compelling avenue for future work.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claim (low initial confidence and high final confidence for better TTS), is explicited stated in introduction and abstract.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discussed limitation in Section 4.3, 4.5, 5.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: NA

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provided full details for reproduction, including hardware, open-sourced software, prompts, and implementation details.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The method, hyperparameters and harness are fully specified in Sections 3.4–4.5 and Appendices B and I. The project page (`https://consilience-tts.github.io`) provides the consilify skill and a reference implementation of the consilience score with known-answer tests.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: All hyperparameters, dataset evaluation are included in Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report 1-sigma error bar for baselines and our methods, in the drawing, we also show 1-sigma boundary.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The hardware information is included in the Appendix

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The paper uses open model and datasets.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: The model uses existing model and data.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: NA

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All datasets and model usesd are properly cited.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: NA

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: NA

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: NA

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: NA

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.