# LEARNING TO STOP WITHOUT LEARNING TO STOP: SELF-SUPERVISED CONFIDENCE TRAINING IMPROVES REASONING EFFICIENCY

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Reasoning models often generate very long reasoning traces, making inference computationally expensive. Existing approaches typically improve efficiency either through inference-time early-stopping mechanisms or by explicitly encouraging shorter reasoning during training, for example through reinforcement learning with length penalties. We show that substantial efficiency gains can instead emerge from a different kind of supervision: *confidence*. Using a self-supervised procedure, we fine-tune reasoning models to predict their confidence in the answer at intermediate points along their own reasoning trajectories using only 600 training problems. Confidence is used only as a training target: the loss contains no objective for reasoning length, efficiency, or stopping. At inference, the fine-tuned models use the standard generation procedure, with no confidence elicitation or early-stopping mechanism. Despite this, self-supervised confidence fine-tuning makes reasoning more efficient, reducing generated tokens by up to 25% at matched accuracy across Gemma, Qwen, Nemotron, and GPT-OSS models on mathematical, scientific, and coding reasoning benchmarks, with efficiency gains comparable to methods that explicitly optimize for shorter reasoning. Analysis of reasoning episodes further shows that confidence supervision largely preserves the base models' high-level reasoning composition rather than selectively suppressing particular behaviors. Our results suggest that efficient reasoning may emerge as a downstream consequence of learning metacognitive signals, without being directly optimized.

## 1 INTRODUCTION

Reasoning language models have achieved strong performance across a wide range of challenging tasks, but often at substantial inference cost. Modern reasoning models can generate tens of thousands of tokens before producing an answer, motivating growing interest in making reasoning more efficient.

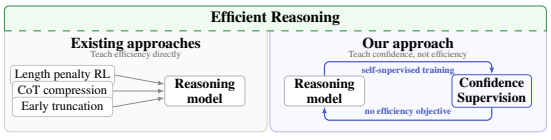



Figure 1: **Teaching confidence instead of efficiency.** Existing methods explicitly optimize for shorter reasoning traces; we supervise only confidence.

Several lines of work have sought to improve the efficiency of reasoning in LLMs. Despite their differences, these approaches all make efficiency an explicit part of either training or inference. At inference time, early-stopping methods monitor signals such as confidence (Yang et al., 2025; Hosseini et al., 2026), uncertainty (Wang et al., 2025), or answer stability (Liu & Wang, 2025) and terminate reasoning when further computation appears unnecessary. At training time, other approaches directly encourage shorter reasoning, for example through reinforcement learning with length penalties (Liu et al., 2025; Arora & Zanette, 2025) or fine-tuning on selected concise reasoning trajectories (Zhao et al., 2026; Munkhbat et al., 2025). These approaches either directly control when reasoning stops or explicitly teach the model to produce shorter reasoning.

In this work, we study a different question: *Can efficient reasoning emerge without directly training the model to produce shorter reasoning traces?* We introduce **ConfSFT**, a self-supervised fine-tuning



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

procedure that trains a reasoning model to predict its confidence at intermediate points along its own reasoning trajectories. The confidence targets are derived entirely from the model's own token probabilities and require neither gold answers nor external judges. Crucially, confidence is the only supervised signal: reasoning tokens are masked from the loss, and the objective contains no term for reasoning length, efficiency, or stopping. At inference, the fine-tuned model uses standard generation, with no confidence elicitation or stopping mechanism. Figure 1 illustrates this.

The motivation for this supervision comes from a simple observation: confidence provides a meaningful signal about the model's intermediate reasoning state. Across intermediate states, higher confidence is associated with more reliable and stable answers, while the expected benefit of further reasoning decreases. This makes confidence a natural signal to learn: it reflects both the reliability of the current answer and the potential value of further reasoning.

Learning this signal alone substantially shortens reasoning: ConfSFT reduces generated tokens by up to 25% at matched accuracy across four model families and transferring from mathematical training problems to scientific and coding tasks. Its efficiency gains are comparable to training methods that explicitly optimize for shorter reasoning.

Our analyses further show that confidence prediction improves over training while generated tokens decrease and accuracy remains stable. A decomposition of reasoning traces into high-level reasoning behaviors shows that ConfSFT largely preserves the base models' reasoning composition, rather than achieving efficiency by selectively suppressing a particular type of reasoning. Controlled ablations further show that the supervision signal matters: alternative targets, including randomly shuffled confidence labels, produce weaker or no efficiency gains. Together, these results provide evidence that learning confidence itself can reshape reasoning toward greater efficiency.

## 2 Related Work

**Training LLMs for efficient reasoning.** A growing line of work directly trains reasoning models to use less computation. Several approaches use reinforcement learning objectives that explicitly favor shorter correct reasoning traces (Liu et al., 2025; Arora & Zanette, 2025; Yi et al., 2025). Others fine-tune on self-generated concise trajectories, either by selecting short reasoning paths or filtering on-policy generations for correctness and conciseness (Munkhbat et al., 2025; Zhao et al., 2026). Other methods explicitly train models to adapt or terminate their reasoning through truncated rollouts, selective reasoning, or confidence-guided post-training (Chen et al., 2026a; Huang et al., 2026; Jiang et al., 2026; Han et al., 2026; Qiao et al., 2025). While these methods differ in how efficiency is encouraged, they all make shorter or more efficient reasoning an explicit part of the training procedure. In contrast, our approach supervises only confidence, without any explicit training signal that prefers shorter reasoning.

**Inference-time adaptive reasoning and early stopping.** A complementary line of work reduces reasoning cost by adapting computation at inference time. Several methods use confidence or uncertainty signals to determine when further reasoning is unnecessary (Yang et al., 2025; Hosseini et al., 2026; Wang et al., 2025; Kim et al., 2026; Yong et al., 2025; Fu et al., 2025). Others rely on the convergence or stability of intermediate answers to terminate reasoning (Liu & Wang, 2025; Mao et al., 2026), while Zhang et al. (2025) train probes on internal representations to detect intermediate answer correctness and enable early exit. Although these approaches use different signals, they all explicitly use an inference-time criterion to control how much reasoning is performed. In contrast, ConfSFT does not use confidence or any other stopping criterion during inference.

**Confidence and metacognitive learning.** A growing body of work incorporates confidence and metacognitive signals into training. Jang et al. (2025) show that fine-tuning on verbalized confidence labels alone can induce changes in reasoning behavior, including emergent self-verification. Other work explicitly trains models to improve their confidence calibration, for example by optimizing verbalized confidence calibration through reinforcement learning (Bani-Harouni et al., 2026; Damani et al., 2026), shaping confidence dynamics to avoid premature commitment (Gai et al., 2026), or using metacognitive self-evaluation as feedback during reinforcement learning (Liu et al., 2026). Confidence has also been incorporated into reward and alignment objectives to improve reasoning quality and reliability (He et al., 2025a; Chen et al., 2026b). ConfSFT differs in both the source and role of the confidence signal. We derive confidence targets directly from the model's own token

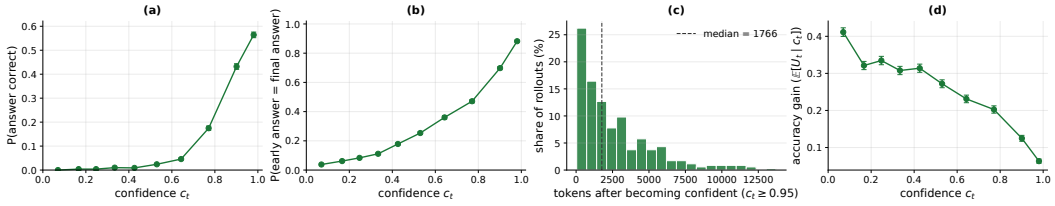



Figure 2: **(a)** Trial-answer correctness and **(b)** agreement with the final answer increase with confidence. **(c)** The model often continues generating after first reaching $c_t \geq 0.95$. **(d)** The expected accuracy gain from further reasoning decreases with confidence.

probabilities at intermediate reasoning states and train only to predict these targets, without gold correctness labels or external feedback. Rather than optimizing efficiency, correctness, or a desired confidence trajectory, we study whether this self-supervised confidence learning alone can induce more efficient reasoning.

## 3 Confidence and the Value of Further Reasoning

**What do we mean by confidence?** At an intermediate point in reasoning, we probe the model for its current answer. Given the reasoning generated so far, we append a fixed answer-elicitation prompt (e.g., `The final answer is \boxed{`) and greedily generate the model's current trial answer, as illustrated in Figure 3a (Step 2). Let $t$ index such an intermediate state and $a_t$ denote its trial answer. We define the corresponding *confidence* $c_t$ as the geometric mean of the token probabilities assigned to $a_t$, i.e., its length-normalized likelihood. Importantly, this score is computed entirely from the model's own distribution and does not require knowing whether the trial answer is correct. The exact answer-elicitation prompts are provided in Appendix A.3.

We do not interpret this score as a calibrated probability of correctness: a confidence of $0.8$, for example, does not necessarily imply that the answer is correct $80\%$ of the time. Instead, we ask whether it provides a meaningful signal about the model's intermediate reasoning state and the value of continuing to reason.

**Confidence is informative about the reasoning state.** We evaluate confidence at intermediate reasoning states immediately preceding each `Wait` in 480 Nemotron traces on AIME 2024 (16 samples per problem) and group the resulting intermediate states into equal-sized confidence quantiles, with roughly 1,700 states per bin.

As shown in Figure 2, higher confidence is strongly associated with a higher probability that the trial answer is correct. The relationship is even stronger with answer stability: as confidence increases, the trial answer becomes increasingly likely to already match the model's eventual final answer. We observe the same trends for Gemma in Appendix D.

**Confidence tracks the value of further reasoning.** Models nevertheless often continue reasoning after reaching a high-confidence state. After first reaching $c_t \geq 0.95$, Nemotron generates a median of 1,766 additional tokens, with many trajectories continuing for more than 10K tokens (Figure 2).

To quantify whether this additional computation is useful, we define the accuracy gain from continuing at an intermediate state $t$ as

$$U_t = \mathbb{1}[a_{\text{final}} \text{ is correct}] - \mathbb{1}[a_t \text{ is correct}], \tag{1}$$

where $a_t$ is the trial answer at $t$ and $a_{\text{final}}$ is the answer after completing the reasoning trajectory. Thus, $U_t > 0$ when further reasoning corrects the current answer and $U_t < 0$ when it changes a correct intermediate answer into an incorrect final answer. As shown in Figure 2, the expected gain from continued reasoning decreases sharply with confidence: further reasoning is most beneficial at low-confidence states and provides little average improvement once confidence is high.

Together, these observations suggest that confidence provides a self-supervised signal about both the model's current reasoning state and the value of additional computation. Prior work has used the same confidence signals to explicitly control reasoning, either through inference-time early stopping (Yang

et al., 2025; Hosseini et al., 2026) or by truncating rollouts during training (Chen et al., 2026a). We instead ask a different question: what happens if the model is simply trained to predict this signal at intermediate reasoning states? In the following sections, we show that confidence supervision alone can substantially reduce generated tokens, without truncating trajectories or explicitly encouraging shorter reasoning.

## 4 METHOD

Motivated by the observations in Section 3, we introduce **ConfSFT**, a self-supervised procedure for fine-tuning reasoning models with confidence supervision at intermediate reasoning states. As illustrated in Figure 3, ConfSFT repeatedly generates reasoning trajectories from the current policy, constructs confidence labels from intermediate states, and fine-tunes the model to predict these labels. We describe each step below.

**Generating reasoning rollouts.** We begin with a small subset $\mathcal{D}_{\text{train}}$ of the training problems. For each problem $x \in \mathcal{D}_{\text{train}}$, we use the current policy $\pi_\theta$ to generate $T$ reasoning trajectories $\tau = (y_1, \ldots, y_T)$. These are standard reasoning rollouts, generated without confidence elicitation or any instruction to reason efficiently. The prompts are provided in Appendix A.1.

**Constructing confidence labels.** For each rollout, we identify a set of intermediate *decision points* using a fixed textual marker. By default, we use occurrences of `Wait` and take the reasoning prefix immediately preceding each occurrence as an intermediate reasoning state. Let $t$ index such a decision point and $y_{<t}$ denote the corresponding reasoning prefix. When a trajectory contains many decision points, we retain at most $M$, selected approximately uniformly across the trajectory, to prevent a single rollout from contributing disproportionately to the training set.

To construct a confidence target for $y_{<t}$, we append a fixed answer-elicitation prompt $q$ and greedily generate a trial answer $a = (a_1, \ldots, a_n)$. We define its confidence as the geometric mean of the token probabilities assigned to the generated answer:

$$c_t = \exp\left(\frac{1}{n}\sum_{i=1}^{n} \log \pi_\theta\left(a_i \mid x, y_{<t}, q, a_{<i}\right)\right). \tag{2}$$

This corresponds to the length-normalized likelihood of the trial answer. Importantly, $c_t$ is computed entirely from the model's own distribution and does not require the gold answer.

We represent confidence as a textual percentage so that it can be learned using the model's standard next-token prediction objective. Specifically, we quantize $c_t \in [0, 1]$ onto a grid of $K = 50$ percentage levels, corresponding to $2\%$ increments, and round each score up to the smallest grid point greater than or equal to it. The resulting textual percentage $\ell_t$ serves as the supervision target for decision point $t$.

**Confidence-supervised fine-tuning.** For each decision point, we construct one training example by concatenating the problem prompt, the corresponding reasoning prefix $y_{<t}$, a fixed natural-language *priming prefix* $p$, and the quantized confidence label $\ell_t$:

$$s = \underbrace{\text{prompt}(x) \oplus y_{<t} \oplus p}_{\text{context, masked}} \oplus \underbrace{\ell_t}_{\text{supervised}} \tag{3}$$

where $\oplus$ denotes concatenation. We use the fixed priming prefix:

```
From 0% (very low) to 100% (very high), my confidence in the answer so far is
```

Let $S(s)$ denote the token positions corresponding to the confidence label $\ell_t$. We optimize the standard next-token cross-entropy only over these positions:

$$\mathcal{L}(\theta) = -\mathbb{E}_{(x,y,t)}\left[\sum_{j \in S(s)} \log \pi_\theta\left(s_j \mid s_{<j}\right)\right]. \tag{4}$$

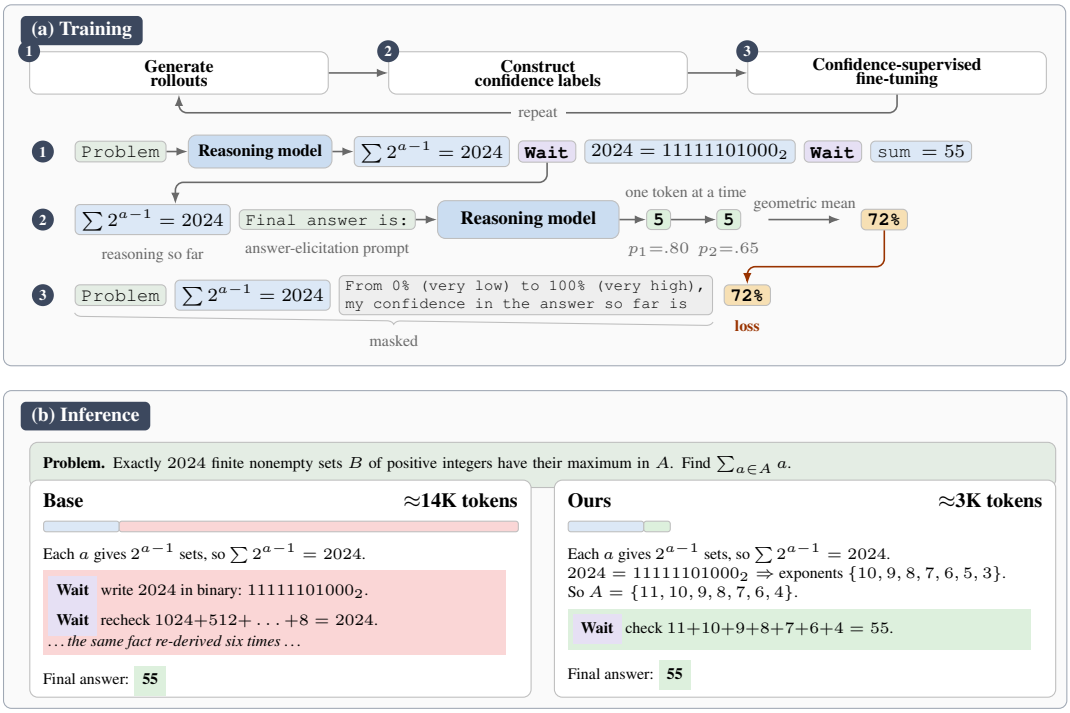



Figure 3: **Overview of ConfSFT. (a)** The policy generates reasoning rollouts, constructs self-supervised confidence labels at intermediate states, and is fine-tuned only on the confidence targets. **(b)** At inference, the fine-tuned policy uses standard generation but can produce substantially shorter reasoning while reaching the same correct answer.

All other tokens, including the problem prompt, reasoning prefix, and priming prefix, are masked from the loss. Thus, the model is trained only to predict the confidence label conditioned on an intermediate reasoning state; it is not trained to reproduce the sampled reasoning trajectory or to generate confidence text within its chain of thought. The objective contains no term for reasoning length, stopping, or efficiency.

**Iterative on-policy training.** Because both reasoning traces and confidence targets are policy-generated, we refresh them as the policy changes. At round $r$, we sample rollouts from a fresh subset of training problems using $\pi_{\theta_r}$, construct confidence labels at decision points, and fine-tune on the resulting examples to obtain $\pi_{\theta_{r+1}}$ (Figure 3a). Later rounds generally yield more efficient policies. We evaluate the policy after each round on a fixed held-out validation set.

**Inference.** Inference is unchanged from the base model. The fine-tuned model uses the same decoding procedure, with no confidence instruction, answer-elicitation cue, priming prefix, verifier, or early-exit mechanism. The fine-tuned models also do not spontaneously emit confidence values or the priming prefix within their reasoning traces. Thus, any change in reasoning length arises from the fine-tuned policy itself rather than an inference-time intervention. Figure 3b illustrates this behavior on an AIME problem, where the fine-tuned policy reaches the same correct answer with substantially shorter reasoning.

# 5 Experiments

## 5.1 Setup

**Models.** We evaluate ConfSFT on four reasoning model families of different scales: Gemma-4-E2B (Team, 2026), Qwen3-4B (Team, 2025), Nemotron-Nano-8B (Bercovich et al., 2025), and GPT-OSS-20B (OpenAI, 2025).

**Training Data.** We train on problems from AIME2000–2023 (Veeraboina, 2023). AIME2024 is held out for validation; no test benchmark is used during training or model selection.

Table 1: **Main results.** ConfSFT reduces generated tokens while maintaining base-model accuracy across model families and task domains.

| Method | MATH | | | | SCIENCE | | CODING | | | | AVERAGE | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | AIME2025 | | GSM8K | | GPQA-Diamond | | LiveCodeBench | | HumanEval | | | |
| | Acc↑ | Avg. Tokens↓ | Acc↑ | Avg. Tokens↓ | Acc↑ | Avg. Tokens↓ | Acc↑ | Avg. Tokens↓ | Acc↑ | Avg. Tokens↓ | Acc↑ | Token Red. |
| *Nemotron-Nano-8B* | | | | | | | | | | | | |
| Base | 42.5 | 11,325 | 91.8 | 1,192 | 52.5 | 7,338 | 44.2 | 11,761 | 89.7 | 3,600 | 64.1 | 0.0% |
| DEER(Yang et al., 2025) | 39.8 | 9,076 (-19.9%) | 89.2 | 765 (-35.8%) | 49.8 | 5,712 (-22.2%) | 17.4 | 1,731 (-85.3%) | 47.1 | 511 (-85.8%) | 48.7 | -49.8% |
| On-Policy SFT(Zhao et al., 2026) | 39.0 | 10,487 (-7.4%) | 91.1 | 999 (-16.2%) | 50.3 | 6,146 (-16.2%) | 41.8 | 11,364 (-3.4%) | 89.3 | 3,174 (-11.8%) | 62.3 | -11.0% |
| A&Z(Arora & Zanette, 2025) | 41.5 | 9,644 (-14.8%) | 91.0 | 1,218 (+2.2%) | 50.9 | 6,738 (-8.2%) | 44.5 | 11,346 (-3.5%) | 88.4 | 3,489 (-3.1%) | 63.3 | -5.5% |
| **ConfSFT (ours)** | 45.6 | 10,153 (-10.4%) | 91.6 | 1,010 (-15.3%) | 52.8 | 6,278 (-14.4%) | 45.1 | 11,151 (-5.2%) | 90.8 | 3,225 (-10.4%) | 65.2 | -11.1% |
| *Gemma-4-E2B* | | | | | | | | | | | | |
| Base | 35.0 | 7,284 | 91.2 | 956 | 42.6 | 3,294 | 42.1 | 7,543 | 93.2 | 2,210 | 60.8 | 0.0% |
| DEER(Yang et al., 2025) | 24.2 | 4,924 (-32.4%) | 90.4 | 912 (-4.6%) | 36.5 | 2,432 (-26.2%) | 31.3 | 5,333 (-29.3%) | 82.9 | 1,905 (-13.8%) | 53.1 | -21.3% |
| On-Policy SFT(Zhao et al., 2026) | 27.3 | 5,601 (-23.1%) | 90.9 | 1,049 (+9.7%) | 42.1 | 3,587 (+8.9%) | 41.0 | 7,711 (+2.2%) | 93.2 | 2,318 (+4.9%) | 58.9 | +0.5% |
| A&Z(Arora & Zanette, 2025) | 35.4 | 7,018 (-3.7%) | 91.3 | 963 (+0.7%) | 41.5 | 3,224 (-2.1%) | 41.2 | 7,291 (-3.3%) | 93.6 | 2,200 (-0.5%) | 60.6 | -1.8% |
| **ConfSFT (ours)** | 32.1 | 5,987 (-17.8%) | 91.4 | 938 (-1.9%) | 40.8 | 2,968 (-9.9%) | 42.4 | 6,403 (-15.1%) | 93.8 | 2,059 (-6.8%) | 60.1 | -10.3% |
| *Qwen3-4B* | | | | | | | | | | | | |
| Base | 59.2 | 13,268 | 94.8 | 2,292 | 53.0 | 9,075 | 46.8 | 14,954 | 93.3 | 3,495 | 69.4 | 0.0% |
| DEER(Yang et al., 2025) | 40.4 | 9,086 (-31.5%) | 84.8 | 580 (-74.7%) | 47.1 | 3,621 (-60.1%) | 25.8 | 10,374 (-30.6%) | 50.8 | 756 (-78.4%) | 49.8 | -55.0% |
| On-Policy SFT(Zhao et al., 2026) | 58.8 | 11,405 (-14.0%) | 95.1 | 1,414 (-38.3%) | 53.9 | 7,694 (-15.2%) | 45.9 | 14,518 (-2.9%) | 93.1 | 2,944 (-15.7%) | 69.3 | -17.2% |
| **ConfSFT (ours)** | 58.8 | 11,365 (-14.3%) | 94.5 | 1,710 (-25.4%) | 55.8 | 6,838 (-24.6%) | 45.0 | 12,122 (-18.9%) | 93.3 | 3,054 (-12.6%) | 69.5 | -19.2% |
| *gpt-oss-20b* | | | | | | | | | | | | |
| Base | 52.7 | 5,802 | 94.0 | 482 | 61.0 | 3,732 | 53.5 | 5,034 | 95.9 | 917 | 71.4 | 0.0% |
| DEER(Yang et al., 2025) | 52.7 | 5,807 (+0.1%) | 94.0 | 482 (+0.1%) | 61.0 | 3,753 (+0.6%) | 51.1 | 4,824 (-4.2%) | 90.6 | 832 (-9.2%) | 69.9 | -2.5% |
| **ConfSFT (ours)** | 53.3 | 5,235 (-9.8%) | 94.1 | 437 (-9.2%) | 61.0 | 3,112 (-16.6%) | 50.7 | 4,393 (-12.7%) | 95.0 | 850 (-7.2%) | 70.8 | -11.1% |

**Evaluation.** We evaluate on AIME2025 (Dekoninck et al., 2026), GSM8K (Cobbe et al., 2021), GPQA-Diamond (Rein et al., 2024), HumanEval (Chen et al., 2021), and LiveCodeBench (Jain et al., 2024). We sample 16 completions per problem and report accuracy and average generated tokens across them. The full evaluation protocol is provided in Appendix C.

**Training Details.** We apply the ConfSFT procedure described in Section 4. We partition the training problems into eight groups, allowing up to eight training rounds, and sample eight rollouts per training problem and 16 per validation problem. Full training hyperparameters and implementation details are provided in Appendix A.

**Baselines.** We compare against the corresponding base model and DEER (Yang et al., 2025), an inference-time early-stopping method that uses confidence to terminate reasoning. As training-time baselines, we evaluate A&Z (Arora & Zanette, 2025), which explicitly incorporates response length into the RL objective, on Nemotron and Gemma, and On-Policy SFT (Zhao et al., 2026) on Nemotron, Gemma, and Qwen. Full baseline training details and more experiments are provided in Appendix F.

**Metrics.** We report accuracy (**Acc.**), average number of generated tokens (**Avg. Tokens**), and token reduction (**Token Red.**), defined as the percentage reduction in average generated tokens relative to the corresponding base model. We also report pass@8 (**P@8**) in Appendix C

### 5.2 MAIN RESULTS

Table 1 shows that ConfSFT consistently reduces generated tokens while maintaining base-model accuracy across all four model families. ConfSFT reduces tokens by 11.1%, 10.3%, 19.2%, and 11.1% on Nemotron, Gemma, Qwen, and GPT-OSS, respectively. In Appendix C.5, we report confidence intervals and show that the token reductions are statistically significant, while the changes in accuracy are not. Importantly, although ConfSFT is trained only on AIME problems, the efficiency gains transfer beyond mathematics to scientific reasoning and coding benchmarks.

**Comparison with explicit efficiency training.** We compare ConfSFT with two training-time approaches: A&Z (Arora & Zanette, 2025) and On-Policy SFT (Zhao et al., 2026). Across the model families where these baselines are evaluated, ConfSFT achieves comparable or larger efficiency gains despite receiving no supervision that favors shorter reasoning. For example, on Qwen, On-Policy SFT reduces tokens by 17.2% at 69.3 average accuracy, compared with 19.2% at 69.5 accuracy for ConfSFT. On Nemotron, ConfSFT similarly achieves substantial token reduction while maintaining base-model accuracy. These results show that confidence-only supervision can yield efficiency gains comparable to methods explicitly designed to shorten reasoning.

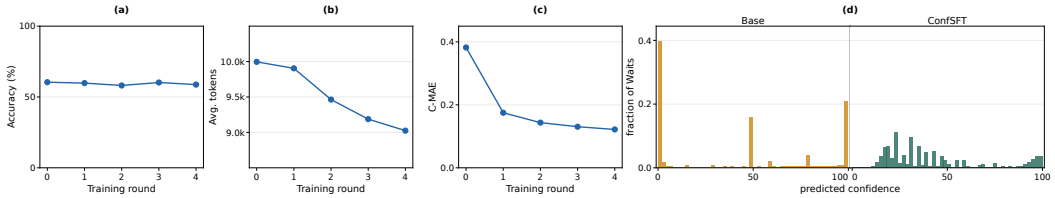



Figure 4: **Training Dynamics.** **(a)** Accuracy remain stable. **(b)** Generated tokens decrease. **(c)** Confidence predictions improved. **(d)** Confidence predictions become more graded.

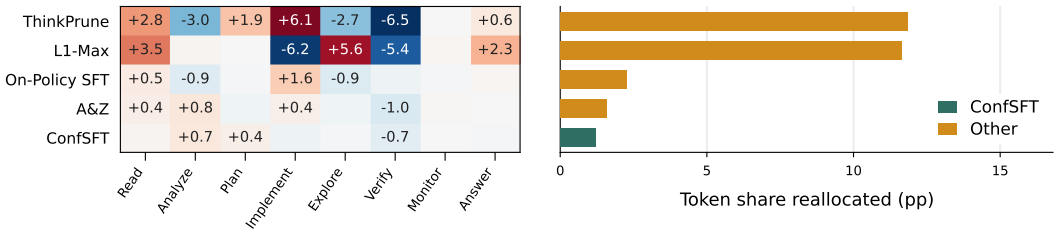



Figure 5: **Different efficiency methods reshape reasoning in different ways.** Following the Schoenfeld-style episode taxonomy, we compare each efficient policy with its corresponding base policy. **Left:** change in reasoning-token share for each episode, in percentage points. **Right:** total variation distance between the before- and after-training episode distributions.

**Comparison with inference-time early stopping.** We additionally compare against DEER (Yang et al., 2025), which uses confidence to explicitly terminate reasoning at inference time. DEER can produce larger token reductions, but its effectiveness varies considerably across tasks and can come with large accuracy losses. This is particularly pronounced on coding benchmarks: with Nemotron, LiveCodeBench accuracy drops from 44.2% to 17.4% and HumanEval accuracy from 89.7% to 47.1%. In contrast, ConfSFT reduces generated tokens without applying an inference-time stopping rule and preserves accuracy across domains. Additional baseline details are provided in Appendix F.

### 5.3 Confidence Learning and Efficiency Dynamics

We next examine whether the model actually learns the confidence-prediction task, and how this relates to the efficiency gains during training. We measure prediction quality using *confidence mean absolute error* (C-MAE), the mean absolute difference between the model's predicted confidence and the quantized self-supervised target on held-out reasoning states. Predictions are elicited using the same priming prefix as in Section 4; lower C-MAE indicates better prediction of the confidence targets.

Figure 4(a–c) shows the training dynamics for Nemotron on the validation set. As training progresses, C-MAE decreases substantially and average generated tokens fall, while accuracy remains approximately unchanged. Thus, the model learns to predict the confidence targets at the same time that its reasoning becomes more efficient.

Figure 4(d) further shows how the predicted confidence distribution changes. Before training, the model concentrates its predictions on only a few confidence levels. After ConfSFT, predictions span a much broader range, suggesting that the model learns to make more graded confidence judgments. We observe similar confidence-learning dynamics for Gemma in Appendix D.

### 5.4 Reasoning Composition under Efficiency Training

Prior work has used Schoenfeld's Episode Theory to decompose model reasoning into different cognitive steps (Li et al., 2025; 2026). In particular, ThinkARM (Li et al., 2026) annotates reasoning at the sentence level using eight categories—*Read*, *Analyze*, *Plan*, *Implement*, *Explore*, *Verify*, *Monitor*, and *Answer*—and shows that efficiency methods can alter reasoning in different ways, selectively changing the amount of computation allocated to different episodes rather than simply shortening all reasoning uniformly.

Following this framework, we annotate reasoning traces before and after efficiency training and compare the fraction of reasoning tokens assigned to each episode. We compare ConfSFT with several explicit efficiency-training methods, including L1 (Aggarwal & Welleck, 2025), ThinkPrune (Hou et al., 2025), A&Z (Arora & Zanette, 2025), and On-Policy SFT (Zhao et al., 2026). We perform this analysis on AIME2025 with 16 sampled completions per problem for each method, using publicly available checkpoints when available and our reproductions otherwise. Full annotation, checkpoint, and details are provided in Appendix G.

Figure 5 shows the change in episode composition relative to each method's corresponding base model. The heatmap reports the signed change in token share for each episode. We summarize the overall change as

$$D_{\mathrm{TV}} = \frac{1}{2} \sum_{k=1}^{8} \left| p_k^{\mathrm{after}} - p_k^{\mathrm{base}} \right|, \tag{5}$$

where $p_k$ is the fraction of reasoning tokens assigned to episode $k$. Equivalently, $D_{\mathrm{TV}}$ measures how much reasoning-token share is reallocated across episode categories after training.

Consistent with prior work (Li et al., 2026), different efficiency methods produce markedly different changes in reasoning composition. L1-Max, ThinkPrune, and On-Policy SFT substantially redistribute computation across episodes, with large shifts in categories such as *Implement*, *Explore*, and *Verify*. In contrast, ConfSFT induces relatively small changes in episode shares, largely preserving the reasoning composition of the corresponding base policies. This suggests that ConfSFT does not achieve efficiency by selectively suppressing a particular reasoning behavior, but rather reduces computation while preserving the relative allocation across high-level reasoning episodes.

## 6 Ablations

### 6.1 What supervision signal matters?

We test whether the efficiency gains of ConfSFT arise specifically from learning confidence, or more generally from fine-tuning on intermediate reasoning states. We keep the training procedure fixed and replace the confidence target with three alternative supervision signals. **Position** uses a target derived only from the position of the corresponding token in the reasoning trajectory, testing whether learning reasoning progress is sufficient. **Binary correctness** replaces graded confidence with a binary target: $100\%$ if the trial answer is correct and $2\%$ otherwise. Unlike ConfSFT, this target requires access to the gold answer. Finally, **Shuffled confidence** randomly permutes the original confidence targets across training examples, preserving the label distribution while breaking the correspondence between each reasoning state and its confidence.

Table 2 shows that the supervision signal matters. Shuffling the confidence targets largely removes or reverses the efficiency gains, while position and binary-correctness supervision yield weaker and less consistent improvements across the two models. These results suggest that the gains are not simply due to fine-tuning on intermediate reasoning states or providing a generic notion of progress; preserving the correspondence between each state and its graded confidence target appears important. Accuracy and token reduction across training rounds are shown in Appendix E.

Table 2: **Supervision-signal ablations.** Replacing graded confidence with position, binary correctness, or shuffled confidence yields weaker or less consistent efficiency gains.

| | Gemma-4-E2B | | Nemotron-Nano-8B | |
|---|---|---|---|---|
| Method | Acc.↑ | Avg. Tokens↓ | Acc.↑ | Avg. Tokens↓ |
| Base | 48.7 | 7,065 | 60.0 | 9,996 |
| Position | 46.5 | 6,550 (-7.3%) | 61.0 | 10,122 (+1.3%) |
| Binary correctness | 48.5 | 7,480 (+5.9%) | 58.3 | 9,703 (-2.9%) |
| Shuffled confidence | 48.7 | 7,057 (-0.1%) | 57.1 | 10,392 (+4.0%) |
| **ConfSFT** | 48.3 | 6,242 (-11.7%) | 58.3 | 9,025 (-9.7%) |

Table 3: **Choice of decision-point marker on Gemma-4-E2B.** ConfSFT remains effective when `Wait` is replaced with paragraph breaks (`\n\n`).

| Marker | Math | | | | Science | | Coding | | | | Average | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | AIME2025 | | GSM8K | | GPQA-Diamond | | LiveCodeBench | | HumanEval | | Acc.↑ | Token Red.↑ |
| | Acc.↑ | Avg. Tokens↓ | Acc.↑ | Avg. Tokens↓ | Acc.↑ | Avg. Tokens↓ | Acc.↑ | Avg. Tokens↓ | Acc.↑ | Avg. Tokens↓ | | |
| Base | 35.0 | 7,284 | 91.2 | 956 | 42.6 | 3,294 | 42.1 | 7,543 | 93.2 | 2,210 | 60.8 | 0.0% |
| `Wait` | 32.1 | 5,987 (-17.8%) | 91.4 | 938 (-1.9%) | 40.8 | 2,968 (-9.9%) | 42.4 | 6,403 (-15.1%) | 93.8 | 2,059 (-6.8%) | 60.1 | -10.3% |
| `\n\n` | 36.3 | 6,241 (-14.3%) | 91.2 | 935 (-2.2%) | 40.9 | 3,015 (-8.5%) | 41.0 | 6,741 (-10.6%) | 93.0 | 2,071 (-6.3%) | 60.5 | -8.4% |

### 6.2 Choice of Decision Points

By default, ConfSFT identifies decision points using occurrences of the textual marker `Wait`. However, the method does not require this marker to occur frequently: Gemma-4-E2B produces only about four `Wait` occurrences per trajectory on average, compared with more than 20 for the other model families, yet still shows substantial efficiency gains in Table 1. This raises a broader question of whether `Wait` itself is important, or whether it simply provides a convenient way to sample intermediate reasoning states. To test this, we use paragraph boundaries (`\n\n`) as a more general marker, which all evaluated models emit consistently and more than 100 times per trajectory on average.

Table 3 shows that ConfSFT remains effective when decision points are identified using `\n\n`. On Gemma, this variant reduces generated tokens by 8.4% while maintaining average accuracy, compared with a 10.3% reduction using `Wait`. Because paragraph boundaries provide substantially more candidate decision points, we use fewer training problems per round and a smaller learning rate for this variant: 4 problems per round and $1 \times 10^{-6}$, compared with 80 problems and $2 \times 10^{-6}$ for `Wait`. Thus, the two rows should not be interpreted as a controlled comparison of the markers themselves; rather, they show that ConfSFT can remain effective with a generic structural marker under an appropriately tuned training configuration.

More generally, we view the textual marker as a mechanism for sampling intermediate reasoning states rather than as a semantically meaningful component of ConfSFT. The results with both `Wait` and `\n\n` are consistent with this view, suggesting that the method does not depend on a particular marker as long as it provides sufficient coverage of the reasoning trajectory.

## 7 Conclusion

We show that reasoning efficiency can emerge from self-supervised confidence training, even when the training objective does not explicitly encourage shorter reasoning or stopping and inference remains unchanged. By training models to predict confidence at intermediate reasoning states using only their own token probabilities, we obtain substantial reductions in generated tokens at matched accuracy across multiple model families and mathematical, scientific, and coding benchmarks, despite training on only 600 AIME problems. Confidence supervision achieves accuracy–efficiency tradeoffs comparable to methods that explicitly optimize for shorter reasoning. These results suggest that more efficient reasoning can emerge as a downstream consequence of learning confidence itself.

**Reproducibility.** We provide detailed descriptions of all components to facilitate reproducibility. Specifically, we provide our prompts in Appendix A, training configurations in Appendix B, evaluation protocol in Appendix C, baseline implementations in Appendix F, and details of our Schoenfeld analysis in Appendix G.

## References

Pranjal Aggarwal and Sean Welleck. L1: Controlling how long a reasoning model thinks with reinforcement learning, 2025. URL https://arxiv.org/abs/2503.04697.

Daman Arora and Andrea Zanette. Training language models to reason efficiently, 2025. URL https://arxiv.org/abs/2502.04463.

David Bani-Harouni, Chantal Pellegrini, Paul Stangel, Ege Özsoy, Kamilia Zaripova, Nassir Navab, and Matthias Keicher. Rewarding doubt: A reinforcement learning approach to calibrated confidence expression of large language models, 2026. URL https://arxiv.org/abs/2503.02623.

Akhiad Bercovich, Itay Levy, Izik Golan, Mohammad Dabbah, Ran El-Yaniv, Omri Puny, Ido Galil, Zach Moshe, Tomer Ronen, Najeeb Nabwani, Ido Shahaf, Oren Tropp, Ehud Karpas, Ran Zilberstein, Jiaqi Zeng, Soumye Singhal, Alexander Bukharin, Yian Zhang, Tugrul Konuk, Gerald Shen, Ameya Sunil Mahabaleshwarkar, Bilal Kartal, Yoshi Suhara, Olivier Delalleau, Zijia Chen, Zhilin Wang, David Mosallanezhad, Adi Renduchintala, Haifeng Qian, Dima Rekesh, Fei Jia, Somshubra Majumdar, Vahid Noroozi, Wasi Uddin Ahmad, Sean Narenthiran, Aleksander Ficek, Mehrzad Samadi, Jocelyn Huang, Siddhartha Jain, Igor Gitman, Ivan Moshkov, Wei Du, Shubham Toshniwal, George Armstrong, Branislav Kisacanin, Matvei Novikov, Daria Gitman, Evelina Bakhturina, Jane Polak Scowcroft, John Kamalu, Dan Su, Kezhi Kong, Markus Kliegl, Rabeeh Karimi, Ying Lin, Sanjeev Satheesh, Jupinder Parmar, Pritam Gundecha, Brandon Norick, Joseph Jennings, Shrimai Prabhumoye, Syeda Nahida Akter, Mostofa Patwary, Abhinav Khattar, Deepak Narayanan, Roger Waleffe, Jimmy Zhang, Bor-Yiing Su, Guyue Huang, Terry Kong, Parth Chadha, Sahil Jain, Christine Harvey, Elad Segal, Jining Huang, Sergey Kashirsky, Robert McQueen, Izzy Putterman, George Lam, Arun Venkatesan, Sherry Wu, Vinh Nguyen, Manoj Kilaru, Andrew Wang, Anna Warno, Abhilash Somasamudramath, Sandip Bhaskar, Maka Dong, Nave Assaf, Shahar Mor, Omer Ullman Argov, Scot Junkin, Oleksandr Romanenko, Pedro Larroy, Monika Katariya, Marco Rovinelli, Viji Balas, Nicholas Edelman, Anahita Bhiwandiwalla, Muthu Subramaniam, Smita Ithape, Karthik Ramamoorthy, Yuting Wu, Suguna Varshini Velury, Omri Almog, Joyjit Daw, Denys Fridman, Erick Galinkin, Michael Evans, Katherine Luna, Leon Derczynski, Nikki Pope, Eileen Long, Seth Schneider, Guillermo Siman, Tomasz Grzegorzek, Pablo Ribalta, Monika Katariya, Joey Conway, Trisha Saar, Ann Guan, Krzysztof Pawelec, Shyamala Prayaga, Oleksii Kuchaiev, Boris Ginsburg, Oluwatobi Olabiyi, Kari Briski, Jonathan Cohen, Bryan Catanzaro, Jonah Alben, Yonatan Geifman, Eric Chung, and Chris Alexiuk. Llama-nemotron: Efficient reasoning models, 2025. URL https://arxiv.org/abs/2505.00949.

Benteng Chen, Weida Wang, Shufei Zhang, Mingbao Lin, and Min Zhang. Step-grpo: Internalizing dynamic early exit for efficient reasoning, 2026a. URL https://arxiv.org/abs/2604.16890.

Kejia Chen, Jiawen Zhang, Yihong Wu, Kewei Gao, Jian Lou, Zunlei Feng, Mingli Song, and Ruoxi Jia. Confidence-aware alignment makes reasoning llms more reliable, 2026b. URL https://arxiv.org/abs/2605.07353.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Mehul Damani, Isha Puri, Stewart Slocum, Idan Shenfeld, Leshem Choshen, Yoon Kim, and Jacob Andreas. Beyond binary rewards: Training lms to reason about their uncertainty, 2026. URL https://arxiv.org/abs/2507.16806.

Jasper Dekoninck, Nikola Jovanović, Tim Gehrunger, Kári Rögnvaldsson, Ivo Petrov, Chenhao Sun, and Martin Vechev. Beyond benchmarks: Matharena as an evaluation platform for mathematics with llms. 2026. URL https://arxiv.org/abs/2605.00674.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. URL https://arxiv.org/abs/1810.04805.

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

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.

Jinyi Han, Ying Huang, Ying Liao, Zishang Jiang, Xikun Lu, Haiquan Zhao, Xinyi Wang, Guanghao Zhou, Sihang Jiang, Jiaqing Liang, Weikang Zhou, Zeye Sun, Fei Yu, and Yanghua Xiao. Your models have thought enough: Training large reasoning models to stop overthinking, 2026. URL https://arxiv.org/abs/2509.23392.

Qianxi He, Qingyu Ren, Shanzhe Lei, Xuhong Wang, and Yingchun Wang. Beyond correctness: Confidence-aware reward modeling for enhancing large language model reasoning. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 27227–27243, Suzhou, China, November 2025a. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1385. URL https://aclanthology.org/2025.emnlp-main.1385/.

Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen Liang, Wenxuan Wang, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Deepmath-103k: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning, 2025b. URL https://arxiv.org/abs/2504.11456.

Parsa Hosseini, Sumit Nawathe, Mahdi Salmani, Meisam Razaviyayn, and Soheil Feizi. Early stopping for large reasoning models via confidence dynamics, 2026. URL https://arxiv.org/abs/2604.04930.

Bairu Hou, Yang Zhang, Jiabao Ji, Yujian Liu, Kaizhi Qian, Jacob Andreas, and Shiyu Chang. Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2504.01296.

Zixuan Huang, Xin Xia, Yuxi Ren, Jianbin Zheng, Xuanda Wang, Zhixia Zhang, Hongyan Xie, Songshi Liang, Zehao Chen, Xuefeng Xiao, Fuzhen Zhuang, Jianxin Li, Deqing Wang, and Yikun Ban. Does your reasoning model implicitly know when to stop thinking?, 2026. URL https://arxiv.org/abs/2602.08354.

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. *arXiv preprint arXiv:2403.07974*, 2024.

Chaeyun Jang, Moonseok Choi, Yegon Kim, Hyungi Lee, and Juho Lee. Verbalized confidence triggers self-verification: Emergent behavior without explicit reasoning supervision, 2025. URL https://arxiv.org/abs/2506.03723.

Qizhi Jiang, Shuo Wang, Pei Ke, Yuhang Song, and Ke Qin. CAT: Confidence-adaptive thinking for efficient reasoning of large reasoning models. In Yunyao Li, Georg Rehm, and Mei Tu (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track)*, pp. 2271–2283, San Diego, California, USA, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-394-4. doi: 10.18653/v1/2026.acl-industry.152. URL https://aclanthology.org/2026.acl-industry.152/.

Junyeob Kim, Sang-goo Lee, and Taeuk Kim. Think just enough: Leveraging self-assessed confidence for adaptive reasoning in language models. In Vera Demberg, Kentaro Inui, and Lluís Marquez (eds.), *Findings of the Association for Computational Linguistics: EACL 2026*, pp. 5000–5006, Rabat, Morocco, March 2026. Association for Computational Linguistics. ISBN 979-8-89176-386-9. doi: 10.18653/v1/2026.findings-eacl.263. URL https://aclanthology.org/2026.findings-eacl.263/.

Ming Li, Nan Zhang, Chenrui Fan, Hong Jiao, Yanbin Fu, Sydney Peters, Qingshu Xu, Robert Lissitz, and Tianyi Zhou. Understanding the thinking process of reasoning models: A perspective from schoenfeld's episode theory, 2025. URL https://arxiv.org/abs/2509.14662.

Ming Li, Chenrui Fan, Yize Cheng, Soheil Feizi, and Tianyi Zhou. Schoenfeld's anatomy of mathematical reasoning by language models, 2026. URL https://arxiv.org/abs/2512.19995.

Gabrielle Kaili-May Liu, Avi Caciularu, Gal Yona, Idan Szpektor, and Arman Cohan. Reinforcement learning with metacognitive feedback elicits faithful uncertainty expression in llms, 2026. URL https://arxiv.org/abs/2606.32032.

Shih-Yang Liu, Xin Dong, Ximing Lu, Shizhe Diao, Mingjie Liu, Min-Hung Chen, Hongxu Yin, Yu-Chiang Frank Wang, Kwang-Ting Cheng, Yejin Choi, Jan Kautz, and Pavlo Molchanov. Dler: Doing length penalty right - incentivizing more intelligence per token via reinforcement learning, 2025. URL https://arxiv.org/abs/2510.15110.

Xin Liu and Lu Wang. Answer convergence as a signal for early stopping in reasoning. *arXiv preprint arXiv:2506.02536*, 2025.

Minjia Mao, Bowen Yin, Yu Zhu, and Xiao Fang. Early stopping chain-of-thoughts in large language models, 2026. URL https://arxiv.org/abs/2509.14004.

Tergel Munkhbat, Namgyu Ho, Seo Hyun Kim, Yongjin Yang, Yujin Kim, and Se-Young Yun. Self-training elicits concise reasoning in large language models, 2025. URL https://arxiv.org/abs/2502.20122.

OpenAI. gpt-oss-120b & gpt-oss-20b model card, 2025. URL https://arxiv.org/abs/2508.10925.

Ziqing Qiao, Yongheng Deng, Jiali Zeng, Dong Wang, Lai Wei, Guanbo Wang, Fandong Meng, Jie Zhou, Ju Ren, and Yaoxue Zhang. ConCISE: Confidence-guided compression in step-by-step efficient reasoning. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 8010–8029, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.405. URL https://aclanthology.org/2025.emnlp-main.405/.

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. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=Ti67584b98.

Alan H. Schoenfeld. *Mathematical Problem Solving*. Academic Press, 1985.

Gemma Team. Gemma 4 technical report, 2026. URL https://arxiv.org/abs/2607.02770.

Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Hemish Veeraboina. Aime problem set 1983-2024, 2023. URL https://www.kaggle.com/datasets/hemishveeraboina/aime-problem-set-1983-2024.

Xi Wang, James McInerney, Lequn Wang, and Nathan Kallus. Entropy after ⟨/Think⟩ for reasoning model early exiting. *arXiv preprint arXiv:2509.26522*, 2025.

Chenxu Yang, Qingyi Si, Yongjie Duan, Zheliang Zhu, Chenyu Zhu, Zheng Lin, Li Cao, and Weiping Wang. Dynamic early exit in reasoning models. *arXiv preprint arXiv:2504.15895*, 2025.

Jingyang Yi, Jiazheng Wang, and Sida Li. Shorterbetter: Guiding reasoning models to find optimal inference length for efficient reasoning, 2025. URL https://arxiv.org/abs/2504.21370.

Xixian Yong, Xiao Zhou, Yingying Zhang, Jinlin Li, Yefeng Zheng, and Xian Wu. Think or not? exploring thinking efficiency in large reasoning models via an information-theoretic lens, 2025. URL https://arxiv.org/abs/2505.18237.

Anqi Zhang, Yulin Chen, Jane Pan, Chen Zhao, Aurojit Panda, Jinyang Li, and He He. Reasoning models know when they're right: Probing hidden states for self-verification, 2025. URL https://arxiv.org/abs/2504.05419.

Anhao Zhao, Ziyang Chen, Junlong Tong, Yingqi Fan, Fanghua Ye, Shuhao Li, Yunpu Ma, Wenjie Li, and Xiaoyu Shen. On-policy supervised fine-tuning for efficient reasoning. *arXiv preprint arXiv:2602.13407*, 2026.

# A Prompts and training sequences

This section gives the exact prompts and training-example construction used by ConfSFT. We use the notation of Section 4.

## A.1 Reasoning prompt (rollouts and evaluation)

Training rollouts and the mathematical evaluations use the same user prompt, applied through each model's chat template with the generation prompt appended. No confidence instruction is added at training or at test time. GPQA-Diamond, LiveCodeBench, and HumanEval replace the boxed line with the benchmark-specific instructions in Appendix C.3.

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

The system prompt is `detailed thinking on` for Nemotron, which is how that family enables chain-of-thought, and empty for Gemma and Qwen. Gemma and Qwen are rendered with `enable_thinking=True` so the chat template opens a thinking block. GPT-OSS-20B uses the Harmony chat template with reasoning effort `medium` and no additional system prompt.

## A.2 Decision-point markers

By default, decision points are occurrences of `Wait` within the model's reasoning. We identify them with the regular expression `\bWait\b` and retain only matches before the end-of-thinking marker (`</think>` for Nemotron and Qwen; `<channel|>` for Gemma; `<|channel|>final` for GPT-OSS-20B). When a trajectory contains more than $M = 32$ decision points, we retain 32 approximately uniformly spaced points in their original order. For Gemma, special tokens are retained while decoding so that `<channel|>` remains available as the end-of-thinking marker. GPT-OSS-20B does the same for `<|channel|>final`.

The paragraph-boundary variant in Section 6 uses the same rule with the marker `\n\n` in place of `Wait`.

## A.3 Trial-answer probe

At a decision point, we append the following answer-elicitation prompt $q$ to the problem prompt and reasoning prefix $y_{<t}$. The decision-point marker itself is excluded from the prefix.

```
\n**Final Answer**\n\nThe final answer is \boxed{
```

We greedily generate at most 16 tokens and stop at a closing brace, newline, end-of-thinking marker, or end-of-sequence token. A terminating stop token is excluded from the score. The trial answer $a$ is the brace-balanced span following the open brace in $q$, and $c_t$ is the geometric mean of its token probabilities, as defined in Equation 2. Constructing the ConfSFT target never uses the gold answer.

## A.4 Priming prefix and label

We use $K = 50$ percentage levels in $2\%$ increments. Each $c_t \in [0, 1]$ is rounded up to the smallest grid point $\ell_t \in \{2\%, 4\%, \ldots, 100\%\}$ that is at least $c_t$. The priming prefix $p$ is the sentence used in Section 4:

```
From 0% (very low) to 100% (very high), my confidence in the answer
so far is
```

### A.5 Training sequence and loss mask

A training example concatenates the chat prompt, the reasoning prefix $y_{<t}$, the priming prefix $p$ and the label $\ell_t$ (Equation 3). A single space separates the reasoning from $p$ when the prefix does not already end in whitespace, and a single space separates $p$ from $\ell_t$. The tail of such a sequence reads

```
...  so the exponents are 10,9,8,7,6,5,3.  From 0% (very low) to
100% (very high), my confidence in the answer so far is 72%
```

and the loss mask over its segments is

| segment | supervised |
|---|---|
| chat prompt (system + problem + instruction) | no |
| reasoning prefix y_<t> | no |
| priming prefix p | no |
| separating space | no |
| "72%" | yes |

Only the tokens of $\ell_t$ receive a loss; every other position, including padding, is assigned the ignore index and does not contribute to Equation 4. We use a maximum training context of $16{,}384$ tokens for Nemotron, Gemma, and Qwen. For GPT-OSS-20B the limit is $8{,}192$ tokens, for both the supervised sequence and the generated rollout, because a $16{,}384$-token update does not fit our compute budget on this model (Appendix B.3).

## B Training hyperparameters

### B.1 Shared recipe

We use full-parameter fine-tuning in `bfloat16` with FSDP for Nemotron, Gemma, and Qwen. GPT-OSS-20B is trained with LoRA instead (Appendix B.3). We use Adafactor with learning rates given in Table 4, no weight decay, gradient clipping at $1.0$, and linear warmup over the first $3\%$ of optimizer steps in each round. The per-device batch size is $1$ with $4$ gradient-accumulation steps. We make one pass over the confidence examples constructed in each round.

Training uses the AIME problems from 2000–2023 (Veeraboina, 2023), while the $30$ AIME 2024 problems are held out for validation and checkpoint selection. The training set contains $695$ problems. We shuffle it once with seed $42$ and divide it into eight disjoint groups: seven groups of $87$ problems and one group of $86$. At round $r$, the current policy $\pi_{\theta_r}$ generates $8$ rollouts for each problem in group $r$. We construct one confidence-supervision example for each retained decision point and fine-tune on the resulting examples to obtain $\pi_{\theta_{r+1}}$. After each round, we generate $16$ validation rollouts per AIME 2024 problem. Training and validation rollouts use the boxed prompt and model-specific decoding parameters in Appendix C.1, with at most $16{,}384$ new tokens ($8{,}192$ for GPT-OSS-20B).

### B.2 Per-model settings

Table 4 lists the settings that differ across the three ConfSFT models in Table 1. The Hugging Face checkpoints are `nvidia/Llama-3.1-Nemotron-Nano-8B-v1`, `google/gemma-4-E2B-it`, and `Qwen/Qwen3-4B`. Attention is SDPA for Nemotron and Qwen and eager attention for Gemma. Gemma additionally keeps special tokens at decode time, as noted above.

Table 4: **ConfSFT hyperparameters** for the three models in Table 1. Shared settings are in the text above.

| | Nemotron-Nano-8B | Gemma-4-E2B | Qwen3-4B |
|---|---|---|---|
| System prompt | `detailed thinking on` | (none) | (none) |
| Think-end marker | `</think>` | `<channel\|>` | `</think>` |
| $T$ / top-$p$ / top-$k$ | 0.6 / 0.95 / 20 | 1.0 / 0.95 / 64 | 0.6 / 0.95 / 20 |
| Learning rate | $1 \times 10^{-6}$ | $2 \times 10^{-6}$ | $1 \times 10^{-6}$ |
| Groups / problems per round | 8 / 87 | 8 / 87 | 8 / 87 |
| Rollouts / training problem | 8 | 8 | 8 |
| Validation rollouts / problem | 16 | 16 | 16 |
| Decision-point marker | `Wait` | `Wait` | `Wait` |
| Selected round | 4 | 3 | 4 |

### B.3 GPT-OSS-20B

GPT-OSS-20B (`openai/gpt-oss-20b`) is a mixture-of-experts model. Its expert weights are stored in MXFP4, and a full-parameter update does not fit our training setup, so we fine-tune a LoRA adapter and leave the base weights frozen, including every expert layer. The adapter has rank $r = 16$, scaling $\alpha = 32$, and dropout 0, and is applied only to the attention projections `q_proj`, `k_proj`, `v_proj`, and `o_proj`. For the supervised step we dequantize the MXFP4 weights to `bfloat16`. Generation keeps the native MXFP4 weights and applies the adapter on top.

The optimizer, warmup, clipping, and loss mask match Section B.1. The learning rate is $2 \times 10^{-5}$, the per-device batch size is 1, and we do not accumulate gradients. We train for one round on the first group of 87 problems, with 2 rollouts per problem, and report that round-1 adapter. Decision points are `Wait`, and the end-of-thinking marker is `<|channel|>final`. Reasoning effort is `medium`.

### B.4 Ablation and decision-point variants

For the supervision-signal ablations in Table 2, we keep the training construction fixed and replace only the target that is expressed as $\ell_t$. **Position** is the fraction of thinking tokens preceding the decision point. **Binary correctness** is $100\%$ when the trial answer is correct and $2\%$ otherwise. **Shuffled confidence** randomly permutes the ConfSFT labels across training examples, preserving their marginal distribution.

The Nemotron ablations use the main Nemotron configuration and learning rate $1 \times 10^{-6}$. We report round 4 for ConfSFT, Position, and Binary correctness, and round 2 for Shuffled confidence. The Gemma ablations use paragraph boundaries as decision points, four training problems per round, learning rate $1 \times 10^{-6}$, and the round-5 checkpoint. This is also the paragraph-boundary configuration reported in Table 3.

## C Evaluation protocol

Table 1 uses standard generation with no confidence instruction, answer cue, or priming prefix. For each problem, we sample 16 independent completions. We report accuracy as mean correctness across all samples (**pass@1**) and **pass@8** using the unbiased estimator

$$\text{pass@}k = 1 - \frac{\binom{n-c}{k}}{\binom{n}{k}},$$

where $n = 16$ is the number of sampled completions and $c$ is the number of correct completions for the problem. This corresponds to the probability that at least one correct answer appears among $k$ samples.

**Avg. Tokens** is the mean number of generated completion tokens. **Token Red.** is computed relative to the corresponding base model as

$$100 \times \left(1 - \frac{t_{\text{method}}}{t_{\text{base}}}\right).$$

### C.1 MODELS AND DECODING

Decoding follows each model family's recommended sampler:

| Model | Mode | $T$ | top-$p$ | top-$k$ |
|---|---|---|---|---|
| Gemma-4-E2B | `enable_thinking=True` | 1.0 | 0.95 | 64 |
| Nemotron-Nano-8B | `detailed thinking on` | 0.6 | 0.95 | 20 |
| Qwen3-4B | `enable_thinking=True` | 0.6 | 0.95 | 20 |
| GPT-OSS-20B | `Reasoning: medium` | 1.0 | 1.0 | 40 |

We use a maximum of $8{,}192$ generated tokens for GPT-OSS-20B, matching the context used during confidence training, and $16{,}384$ for the other models. For LiveCodeBench, we increase the limit to $32{,}768$.

### C.2 BENCHMARKS AND SAMPLE COUNTS

We use the Hugging Face versions listed below. For LiveCodeBench, we use the lite code-generation split from releases v5 and v6 and keep problems with `contest_date` $\geq$ 2025-01-01, resulting in 182 problems from contests between 2025-01-04 and 2025-04-06. For GPQA-Diamond, we shuffle the answer choices once per question and use the same ordering for all methods.

| Benchmark | Hugging Face source | Problems |
|---|---|---|
| AIME 2025 | `opencompass/AIME2025` (parts I+II) | 30 |
| GSM8K | `openai/gsm8k` (`main` test) | 1,319 |
| GPQA-Diamond | `Idavidrein/gpqa` (`gpqa_diamond`) | 198 |
| LiveCodeBench | `livecodebench/code_generation_lite` (v5+v6) | 182 |
| HumanEval | `openai/openai_humaneval` (test) | 164 |

### C.3 EVALUATION PROMPTS

AIME 2025 and GSM8K use the boxed instruction of Appendix A.1. GPQA-Diamond asks for a letter in a box. Code benchmarks ask for a markdown fence rather than a box, and LiveCodeBench keeps the official stdin versus starter-code wording.

```
GPQA-Diamond:
{question}
(A) ...
(B) ...
(C) ...
(D) ...
Reason step by step and put your final answer letter (A, B, C, or D)
in \boxed{}.
```

```
LiveCodeBench (no starter code):
{question}
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 delimiters as follows.
Please reason step by step.  Put your final Python solution in a
markdown code block:  "'python ...  "'.
```

```
LiveCodeBench (with starter code):
{question}
You will use the following starter code to write the solution to
the problem and enclose your code within delimiters.
```python {starter_code} ```
Please reason step by step.  Put your final Python solution in a
markdown code block:  ```python ...  ```.
```

```
HumanEval:  Complete the following Python function.  Read the
docstring carefully.
```python {function_prompt} ```
Please reason step by step.  Put your final Python solution in a
markdown code block:  ```python ...  ```.  The block must define the
function described above (including the signature).
```

### C.4 Answer extraction and grading

We use the same fixed extraction and grading procedure for the base model and all evaluated methods. Although the prompts specify a final-answer format, models occasionally produce the correct answer without following it exactly. We therefore use simple fallback rules to recover answers or code already present in the completion rather than marking these cases incorrect.

**AIME 2025 and GSM8K.** We first look for an answer in `\boxed{...}` and take the first number inside the last box. If no boxed answer is present, we take the last number appearing in the final 160 characters of the completion. The extracted number is compared with the first number in the gold answer, so, for example, a prediction of 336 is credited against gold 336°.

**GPQA-Diamond.** We first extract the final answer letter A–D from the last `\boxed{...}`. If no boxed answer is present, we take the last standalone letter A–D in the final 160 characters. The extracted letter is compared with the gold answer after applying the fixed answer-choice ordering used for that question.

**LiveCodeBench.** We extract Python code from the completion and evaluate it with the official LiveCodeBench tests using a 6s timeout. We first use the last non-empty closed markdown code block. If the model opens a final code block but does not close it, we also consider the text following that fence. When the benchmark expects a `Solution` method but the generated code instead defines the corresponding standalone function, we additionally wrap that function in the required interface. A sample is marked correct if one of these extracted versions passes all tests. These fallback rules only recover code already present in the completion or adapt its interface; we do not repair or otherwise modify the generated solution.

**HumanEval.** We use the last markdown code block, dropping a leading language tag when present. If no usable code block is found, we use the code produced after the model's reasoning section. If the extracted program already defines the target function, we execute it directly; otherwise, we treat it as a continuation of the dataset prompt and prepend the original function prompt. We evaluate with the official HumanEval tests using a 3s timeout.

### C.5 More evaluation results

Table 5 reports paired question-level 95% confidence intervals for the results in Table 1. Each method is compared with its corresponding base model on the same set of questions. Accuracy values are differences in percentage points, while token values are differences in average generated tokens. The intervals show that the token reductions from ConfSFT are consistent across model families, while accuracy remains close to the corresponding base models.

Table 6 additionally reports pass@8, computed with the unbiased estimator described in Appendix C. The overall pattern is consistent with pass@1: ConfSFT substantially reduces generated tokens while preserving pass@8 close to the corresponding base model across all four model families.

Table 5: **Main results relative to the base model,** with paired 95% CIs. Acc and tokens are differences from the corresponding base row of Table 1.

| Method | Math | | | | Science | | Coding | | | | Average | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | AIME2025 | | GSM8K | | GPQA-Diamond | | LiveCodeBench | | HumanEval | | | |
| | Acc↑ | Avg. Tokens↓ | Acc↑ | Avg. Tokens↓ | Acc↑ | Avg. Tokens↓ | Acc↑ | Avg. Tokens↓ | Acc↑ | Avg. Tokens↓ | Acc↑ | Token Red. |
| *Nemotron-Nano-8B* | | | | | | | | | | | | |
| DEER(Yang et al., 2025) | −2.7±2.3 | −2,249±593 | −2.6±0.5 | −427±29 | −2.7±1.5 | −1,626±220 | −26.8±4.4 | −10,030±944 | −42.6±4.7 | −3,089±701 | −15.5±1.4 | −49.8±1.6% |
| On-Policy SFT(Zhao et al., 2026) | −3.5±4.9 | −838±351 | −0.7±0.5 | −193±25 | −2.2±2.1 | −1,192±248 | −2.3±1.4 | −397±189 | −0.4±1.2 | −426±234 | −1.8±1.1 | −11.0±1.6% |
| A&Z(Arora & Zanette, 2025) | −1.0±5.4 | −1,681±458 | −0.8±0.3 | +26±16 | −1.6±1.8 | −600±151 | +0.3±1.3 | −415±224 | −1.3±1.2 | −112±253 | −0.9±1.2 | −5.5±1.6% |
| **ConfSFT (ours)** | +3.1±4.6 | −1,173±385 | −0.2±0.3 | −182±17 | +0.3±1.7 | −1,060±128 | +1.0±1.1 | −610±191 | +1.1±1.3 | −376±275 | +1.1±1.0 | −11.1±1.6% |
| *Gemma-4-E2B* | | | | | | | | | | | | |
| DEER(Yang et al., 2025) | −10.8±5.6 | −2,360±630 | −0.8±0.2 | −44±6 | −6.1±1.9 | −862±125 | −10.8±2.4 | −2,210±310 | −10.3±2.2 | −305±89 | −7.8±1.4 | −21.3±1.9% |
| On-Policy SFT(Zhao et al., 2026) | −7.7±5.6 | −1,684±538 | −0.3±0.4 | +93±6 | −0.5±1.9 | +293±54 | −1.1±1.4 | +168±103 | 0.0±1.5 | +108±32 | −1.9±1.3 | +0.5±1.4% |
| A&Z(Arora & Zanette, 2025) | +0.4±3.9 | −266±171 | +0.1±0.3 | +7±5 | −1.1±1.5 | −70±31 | −0.9±1.6 | −252±96 | +0.3±1.2 | −10±34 | −0.2±0.9 | −1.8±0.6% |
| **ConfSFT (ours)** | −2.9±3.0 | −1,297±244 | +0.2±0.3 | −18±4 | −1.8±1.7 | −326±44 | +0.3±1.3 | −1,139±129 | +0.6±1.0 | −151±39 | −0.7±0.8 | −10.3±0.7% |
| *Qwen3-4B* | | | | | | | | | | | | |
| DEER(Yang et al., 2025) | −18.8±9.8 | −4,182±1121 | −9.9±1.8 | −1,711±84 | −6.0±4.0 | −5,455±427 | −21.1±3.9 | −4,580±580 | −42.5±4.8 | −2,738±524 | −19.6±2.5 | −55.0±2.4% |
| On-Policy SFT(Zhao et al., 2026) | −0.4±7.3 | −1,863±665 | +0.2±0.6 | −877±56 | +0.9±3.6 | −1,381±180 | −0.9±1.5 | −436±322 | −0.1±0.8 | −550±169 | −0.1±1.7 | −17.2±1.7% |
| **ConfSFT (ours)** | −0.4±5.2 | −1,903±583 | −0.3±0.8 | −583±63 | +2.7±2.9 | −2,237±215 | −1.9±1.3 | −2,832±383 | 0.0±1.1 | −440±207 | 0.0±1.2 | −19.2±1.6% |
| *gpt-oss-20b* | | | | | | | | | | | | |
| DEER(Yang et al., 2025) | 0.0±0.0 | +5±24 | 0.0±0.0 | +1±0 | 0.0±0.0 | +21±4 | −2.5±0.8 | −209±91 | −5.3±1.9 | −84±45 | −1.6±0.4 | −2.5±0.9% |
| **ConfSFT (ours)** | +0.6±4.3 | −567±187 | +0.1±0.3 | −44±9 | −0.1±1.6 | −620±89 | −2.9±2.3 | −641±110 | −0.9±1.4 | −66±25 | −0.6±1.1 | −11.1±1.0% |

Table 6: **Pass@8.** Same methods and benchmarks as Table 1.

| Method | Math | | | | Science | | Coding | | | | Average | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | AIME2025 | | GSM8K | | GPQA-Diamond | | LiveCodeBench | | HumanEval | | | |
| | P@8↑ | Avg. Tokens↓ | P@8↑ | Avg. Tokens↓ | P@8↑ | Avg. Tokens↓ | P@8↑ | Avg. Tokens↓ | P@8↑ | Avg. Tokens↓ | P@8↑ | Token Red. |
| *Nemotron-Nano-8B* | | | | | | | | | | | | |
| Base | 67.1 | 11,325 | 97.0 | 1,192 | 82.3 | 7,338 | 59.4 | 11,761 | 97.9 | 3,600 | 80.7 | 0.0% |
| DEER(Yang et al., 2025) | 66.9 | 9,076 (-19.9%) | 96.8 | 765 (-35.8%) | 79.7 | 5,712 (-22.2%) | 34.3 | 1,731 (-85.3%) | 80.5 | 511 (-85.8%) | 71.7 | -49.8% |
| On-Policy SFT(Zhao et al., 2026) | 67.7 | 10,487 (-7.4%) | 96.3 | 999 (-16.2%) | 81.9 | 6,146 (-16.2%) | 58.7 | 11,364 (-3.4%) | 97.5 | 3,174 (-11.8%) | 80.4 | -11.0% |
| A&Z(Arora & Zanette, 2025) | 71.6 | 9,644 (-14.8%) | 96.8 | 1,218 (+2.2%) | 80.1 | 6,738 (-8.2%) | 61.1 | 11,346 (-3.5%) | 97.3 | 3,489 (-3.1%) | 81.4 | -5.5% |
| **ConfSFT (ours)** | 69.7 | 10,153 (-10.4%) | 97.0 | 1,010 (-15.3%) | 81.9 | 6,278 (-14.4%) | 59.0 | 11,151 (-5.2%) | 98.3 | 3,225 (-10.4%) | 81.2 | -11.1% |
| *Gemma-4-E2B* | | | | | | | | | | | | |
| Base | 56.9 | 7,284 | 95.8 | 956 | 75.3 | 3,294 | 57.7 | 7,543 | 98.6 | 2,210 | 76.9 | 0.0% |
| DEER(Yang et al., 2025) | 46.5 | 4,924 (-32.4%) | 95.7 | 912 (-4.6%) | 70.9 | 2,432 (-26.2%) | 51.8 | 5,333 (-29.3%) | 97.7 | 1,905 (-13.8%) | 72.5 | -21.3% |
| On-Policy SFT(Zhao et al., 2026) | 45.5 | 5,601 (-23.1%) | 96.0 | 1,049 (+9.7%) | 75.0 | 3,587 (+8.9%) | 58.6 | 7,711 (+2.2%) | 99.4 | 2,318 (+4.9%) | 74.9 | +0.5% |
| A&Z(Arora & Zanette, 2025) | 53.5 | 7,018 (-3.7%) | 96.0 | 963 (+0.7%) | 73.4 | 3,224 (-2.1%) | 55.5 | 7,291 (-3.3%) | 99.0 | 2,200 (-0.5%) | 75.5 | -1.8% |
| **ConfSFT (ours)** | 57.3 | 5,987 (-17.8%) | 95.8 | 938 (-1.9%) | 73.6 | 2,968 (-9.9%) | 58.9 | 6,403 (-15.1%) | 99.1 | 2,059 (-6.8%) | 76.9 | -10.3% |
| *Qwen3-4B* | | | | | | | | | | | | |
| Base | 73.3 | 13,268 | 94.8 | 2,292 | 72.2 | 9,075 | 58.9 | 14,954 | 97.1 | 3,495 | 79.3 | 0.0% |
| DEER(Yang et al., 2025) | 73.3 | 9,086 (-31.5%) | 84.8 | 580 (-74.7%) | 64.1 | 3,621 (-60.1%) | 48.4 | 10,374 (-30.6%) | 85.4 | 756 (-78.4%) | 71.2 | -55.0% |
| On-Policy SFT(Zhao et al., 2026) | 76.7 | 11,405 (-14.0%) | 96.9 | 1,414 (-38.3%) | 77.1 | 7,694 (-15.2%) | 60.8 | 14,518 (-2.9%) | 97.2 | 2,944 (-15.7%) | 81.7 | -17.2% |
| **ConfSFT (ours)** | 76.7 | 11,365 (-14.3%) | 94.5 | 1,710 (-25.4%) | 70.7 | 6,838 (-24.6%) | 56.1 | 12,122 (-18.9%) | 97.0 | 3,054 (-12.6%) | 79.0 | -19.2% |
| *gpt-oss-20b* | | | | | | | | | | | | |
| Base | 78.4 | 5,802 | 97.7 | 482 | 85.8 | 3,732 | 67.4 | 5,034 | 99.3 | 917 | 85.7 | 0.0% |
| DEER(Yang et al., 2025) | 78.4 | 5,807 (+0.1%) | 97.7 | 482 (+0.1%) | 85.8 | 3,753 (+0.6%) | 67.0 | 4,824 (-4.2%) | 99.1 | 832 (-9.2%) | 85.6 | -2.5% |
| **ConfSFT (ours)** | 76.3 | 5,235 (-9.8%) | 97.8 | 437 (-9.2%) | 88.3 | 3,112 (-16.6%) | 66.1 | 4,393 (-12.7%) | 98.8 | 850 (-7.2%) | 85.5 | -11.1% |

## D Additional analyses on Gemma-4-E2B

Figure 6 repeats the four-panel analysis of Figure 2 on the Gemma-4-E2B base model.

We repeat the confidence-learning analyses from Section 5.2 on Gemma-4-E2B.

Figure 7 repeat the confidence-learning analyses from Section 5.2 on Gemma-4-E2B. It tracks C-MAE, tokens, and accuracy across training rounds. C-MAE and tokens falls over training, and accuracy stays within a few points of the base model.

## E Supervision-signal ablations over training rounds

Figure 8 tracks the four supervision variants of Table 2 on AIME 2024 validation ($30 \times 16$) over training rounds. Left panels show accuracy; right panels show token reduction relative to the pre-SFT baseline (positive is shorter). On both models, ConfSFT is the only variant whose token reduction grows steadily while accuracy remains near the base model.

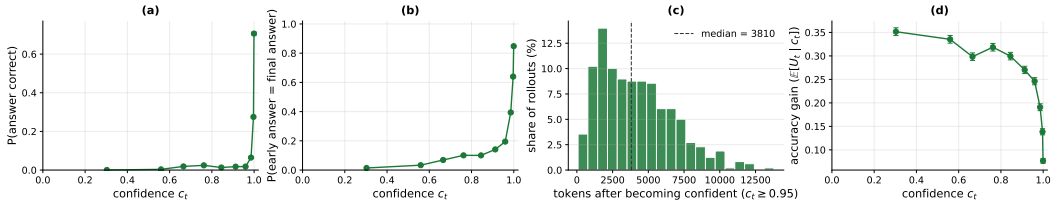



Figure 6: Gemma-4-E2B counterpart of Figure 2.

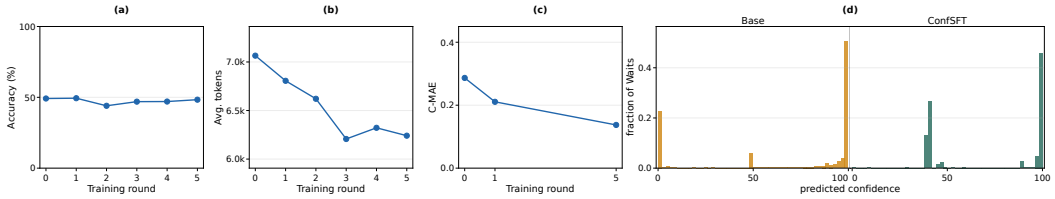



Figure 7: Gemma-4-E2B counterpart of Figure 4.

# F Baselines

## F.1 DEER

We apply the early-exit rule of Yang et al. (2025) with confidence threshold $0.95$ (following their setup) to stored reasoning traces and grade its output using the same benchmark-specific procedures described above. Decision points and the trial-answer probe match Appendices A.2 and A.3. DEER exits at the first decision point with $c_t \geq 0.95$ and returns the corresponding trial answer. If no decision point crosses the threshold, it returns the original completion.

The reported token count includes the reasoning prefix through the exit and the trial-answer tokens generated at each visited decision point. The fixed cue $q$ is not included in the token count.

On LiveCodeBench and HumanEval, the trial prompt replaces the boxed cue of Appendix A.3:

```
\n```python\n
```

Following the original DEER setup for coding benchmarks, confidence is computed using at most the first $50$ trial-answer tokens.

We additionally evaluate a more conservative threshold of $0.98$ in Table 7. Increasing the threshold reduces premature exits and improves accuracy, but correspondingly yields smaller token savings. The qualitative conclusion remains unchanged: DEER's accuracy–efficiency behavior is highly sensitive to the stopping threshold and varies substantially across tasks, whereas ConfSFT achieves more consistent reductions without an inference-time stopping rule.

## F.2 A&Z

We train Nemotron-8B and Gemma-4-E2B using the GRPO with the length penalty from Arora & Zanette (2025) on a random subset of size 10k from the DeepMath-103K dataset He et al. (2025b) for training. We use the Verl framework for training, with FSDP and colocated vLLM rollout, and TP=8 on $8\times$H100, using number of rollouts $K = 16$ for Nemotron and $K = 8$ for Gemma-4, with batch size 64, learning rate $1 \times 10^{-6}$, KL loss coefficient $0.04$, and no normalization of advantages by standard deviation in GRPO. To fit the model into a H100 node, we use maximum response length of $12{,}288$ for Nemotron and $9{,}216$ for Gemma-4, with GPU memory utilization of $0.4$ and $0.3$ respectively. This is because Gemma-4 has a much larger vocabulary size of $\sim$262k, which causes out-of-memory errors if the maximum response length is too high.

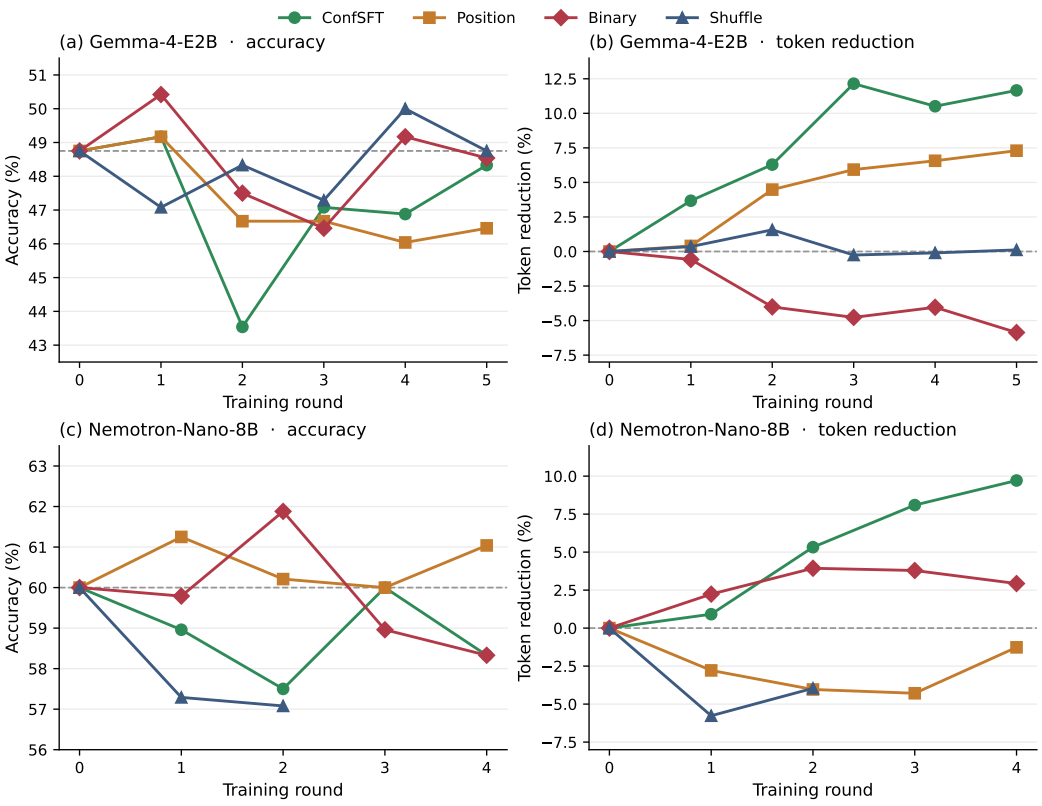



Figure 8: Accuracy (left) and token reduction (right) over training rounds for the four supervision signals in Table 2, AIME 2024 validation. Dashed lines mark the pre-SFT baseline.

We trained our RL length-penalty models using a DeepMath-103K subset rather than the AIME. We use $\alpha = 0.2$ for both Nemotron-8B and Gemma-4-E2B. Smaller values such as $\alpha = 0.05$ yielded only minor efficiency gains, while larger values such as $\alpha = 0.4$ caused accuracy degradation.

We provide the A&Z training curves in Figure 9. Compared with ConfSFT, this baseline uses a substantially larger training set (10K problems, roughly $15\times$ larger) and requires many more optimization steps. We evaluate multiple checkpoints from different training steps on AIME2025 and report the checkpoint with the greatest token reduction and accuracy close to the base model. Despite this additional training and checkpoint selection, A&Z yields smaller token reductions than ConfSFT on both Nemotron and Gemma in Table 1.

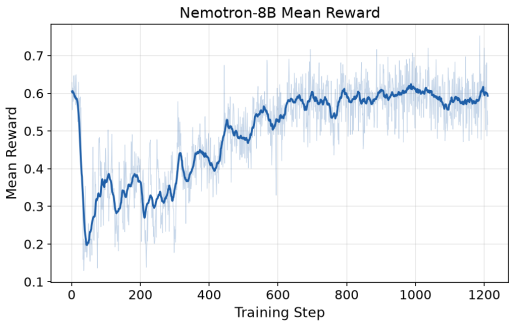
The RL optimization is also noticeably less stable. On Nemotron, response length collapses early in training together with reward, and both require several hundred steps to recover. Gemma is more stable, but its response length decreases only modestly over training. We also find the method sensitive to the length-penalty coefficient: small values provide little efficiency gain, while larger values substantially degrade performance.

### F.3 On-Policy SFT

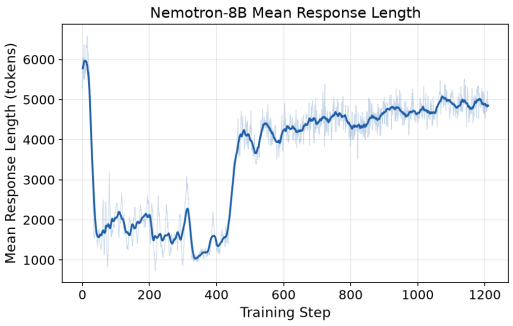
We evaluate On-Policy-SFT (Zhao et al., 2026) as an additional baseline training method. We utilize their released code, which is a modification of the Verl framework, with some minor changes for version compatibility and adding support for additional models. For the Qwen3-4B, Nemotron, and Gemma-4 models, we use AIME 2000-2023 as a training dataset, with $N = 2$ rollouts per prompt per epoch, a training batch size of 16, a maximum response length of 16384. As a validation set, we use AIME 2024 with 16 rollouts per problem and a maximum respone length of 16384. For the Qwen and Gemma models, training is performed on 16×RTXA5000 GPUs, while for Nemotron it is performed on 4×L40S GPUs; we use tenor parallelism TP=2 in all cases.

Table 7: **DEER confidence thresholds.** Early exit at 0.95 and 0.98 thresholds.

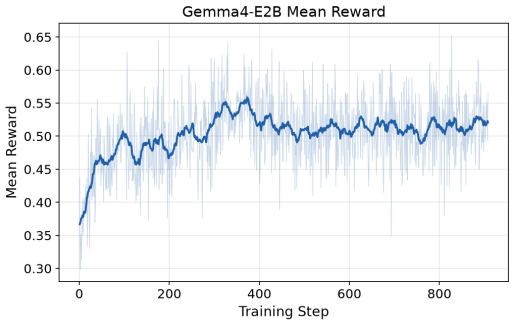
| Method | Math | | | | Science | | Coding | | | | Average | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | AIME2025 | | GSM8K | | GPQA-Diamond | | LiveCodeBench | | HumanEval | | | |
| | Acc↑ | Avg. Tokens↓ | Acc↑ | Avg. Tokens↓ | Acc↑ | Avg. Tokens↓ | Acc↑ | Avg. Tokens↓ | Acc↑ | Avg. Tokens↓ | Acc↑ | Token Red. |
| *Nemotron-Nano-8B* | | | | | | | | | | | | |
| Base | 42.5 | 11,325 | 91.8 | 1,192 | 52.5 | 7,338 | 44.2 | 11,761 | 89.7 | 3,600 | 64.1 | 0.0% |
| DEER (0.95) | 39.8 | 9,076 (-19.9%) | 89.2 | 765 (-35.8%) | 49.8 | 5,712 (-22.2%) | 17.4 | 1,731 (-85.3%) | 47.1 | 511 (-85.8%) | 48.7 | -49.8% |
| DEER (0.98) | 41.9 | 9,585 (-15.4%) | 90.4 | 916 (-23.2%) | 51.4 | 6,494 (-11.5%) | 28.1 | 6,727 (-42.8%) | 60.7 | 1,005 (-72.1%) | 54.5 | -33.0% |
| **ConfSFT** | 45.6 | 10,153 (-10.4%) | 91.6 | 1,010 (-15.3%) | 52.8 | 6,278 (-14.4%) | 45.1 | 11,151 (-5.2%) | 90.8 | 3,225 (-10.4%) | 65.2 | -11.1% |
| *Gemma-4-E2B* | | | | | | | | | | | | |
| Base | 35.0 | 7,284 | 91.2 | 956 | 42.6 | 3,294 | 42.1 | 7,543 | 93.2 | 2,210 | 60.8 | 0.0% |
| DEER (0.95) | 24.2 | 4,924 (-32.4%) | 90.4 | 912 (-4.6%) | 36.5 | 2,432 (-26.2%) | 31.3 | 5,333 (-29.3%) | 82.9 | 1,905 (-13.8%) | 53.1 | -21.3% |
| DEER (0.98) | 29.6 | 5,814 (-20.2%) | 90.7 | 925 (-3.2%) | 38.7 | 2,739 (-16.8%) | 38.9 | 6,910 (-8.4%) | 89.9 | 2,136 (-3.4%) | 57.6 | -10.4% |
| **ConfSFT** | 32.1 | 5,987 (-17.8%) | 91.4 | 938 (-1.9%) | 40.8 | 2,968 (-9.9%) | 42.4 | 6,403 (-15.1%) | 93.8 | 2,059 (-6.8%) | 60.1 | -10.3% |
| *Qwen3-4B* | | | | | | | | | | | | |
| Base | 59.2 | 13,268 | 94.8 | 2,292 | 53.0 | 9,075 | 46.8 | 14,954 | 93.3 | 3,495 | 69.4 | 0.0% |
| DEER (0.95) | 40.4 | 9,086 (-31.5%) | 84.8 | 580 (-74.7%) | 47.1 | 3,621 (-60.1%) | 25.8 | 10,374 (-30.6%) | 50.8 | 756 (-78.4%) | 49.8 | -55.0% |
| DEER (0.98) | 42.9 | 9,289 (-30.0%) | 87.4 | 689 (-69.9%) | 50.3 | 4,573 (-49.6%) | 39.2 | 13,986 (-6.5%) | 63.5 | 1,140 (-67.4%) | 56.7 | -44.7% |
| **ConfSFT** | 58.8 | 11,365 (-14.3%) | 94.5 | 1,710 (-25.4%) | 55.8 | 6,838 (-24.6%) | 45.0 | 12,122 (-18.9%) | 93.3 | 3,054 (-12.6%) | 69.5 | -19.2% |
| *gpt-oss-20b* | | | | | | | | | | | | |
| Base | 52.7 | 5,802 | 94.0 | 482 | 61.0 | 3,732 | 53.5 | 5,034 | 95.9 | 917 | 71.4 | 0.0% |
| DEER (0.95) | 52.7 | 5,807 (+0.1%) | 94.0 | 482 (+0.1%) | 61.0 | 3,753 (+0.6%) | 51.1 | 4,824 (-4.2%) | 90.6 | 832 (-9.2%) | 69.9 | -2.5% |
| DEER (0.98) | 52.7 | 5,833 (+0.5%) | 94.0 | 483 (+0.2%) | 61.0 | 3,756 (+0.7%) | 53.1 | 5,267 (+4.6%) | 94.4 | 908 (-1.0%) | 71.0 | +1.0% |
| **ConfSFT** | 53.3 | 5,235 (-9.8%) | 94.1 | 437 (-9.2%) | 61.0 | 3,112 (-16.6%) | 50.7 | 4,393 (-12.7%) | 95.0 | 850 (-7.2%) | 70.8 | -11.1% |



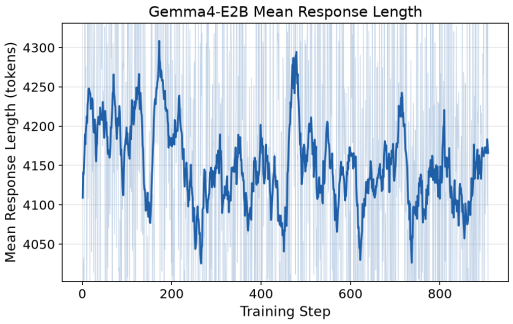

(a) Nemotron-8B reward.




(b) Nemotron-8B mean response length.




(c) Gemma-4-E2B reward.




(d) Gemma-4-E2B mean response length.

Figure 9: **Length-penalty GRPO training curves.**

For all three models, we perform a hyperparameter grid search over learning rate (ranging from $1 \times 10^{-6}$ to $5 \times 10^{-6}$). For all models, we use the recommended settings from each model's release for validation rollout generation; however, we set the generation training temperature at 1.0, following the guidelines in Zhao et al. (2026). For each model, we run a training loop for each learning rate, saving a checkpoint after every 10 batches up to 2 epochs; among these checkpoints, we choose the one that provides the best accuracy-efficiency tradeoff on our validation set.

## G SCHOENFELD'S ANATOMY

Schoenfeld's Episode Theory (Schoenfeld, 1985) is a framework originally designed for analyzing the cognitive states of human during mathematical problem solving. Li et al. (2025; 2026) adapt the theory for studying the thinking patterns of large reasoning models. By decomposing the reasoning process into 8 categories of cognitive episodes, namely *Read*, *Analyze*, *Plan*, *Implement*, *Explore*, *Verify*, *Answer*, and *Monitor*, the adapted framework provides a comprehensive characterization of reasoning in terms of structures and dynamics.

We follow their sentence-level annotation setting and train a classifier so that the same taxonomy can be applied to every trace in our efficiency comparison. Supervision comes from the sentence-level episode labels on the open-source traces of Li et al. (2026), covering $15$ models. We draw a uniform subsample of $120{,}000$ sentences with seed $42$, then hold out a stratified $15\%$ test set, leaving $102{,}000$ training sentences and $18{,}000$ test sentences. As in the original annotation guide, the category of a sentence depends on both the sentence and its immediate context. Each training example therefore uses the previous sentence and the current sentence as input, and the label is one of the eight episode categories. The first sentence of a trace has an empty previous sentence.

We fine-tune `google-bert/bert-base-uncased` (Devlin et al., 2019) with an eight-way classification head. Inputs are truncated at $160$ tokens. Training uses AdamW with learning rate $2 \times 10^{-5}$ and $\epsilon = 10^{-8}$, a linear decay schedule, and a warmup covering $6\%$ of the optimizer steps. Gradients are clipped to norm $1.0$, and the batch size is 32. We train for three epochs. On the $18{,}000$ held-out sentences the final checkpoint reaches Cohen's $\kappa = 0.754$, indicating strong agreement with the reference labels.

At annotation time we segment each completion with the same sentence splitter used to build the training labels, and classify every sentence from its preceding sentence together with itself. Figure 5 compares each trained policy with the base policy it starts from. Both sides use the same AIME 2025 problems and the same 16 sampled completions per problem. We use a public checkpoint when one is available (L1-Max checkpoint for L1 method), and our reproduction otherwise. The share of episode $k$ in a completion is the fraction of its reasoning tokens assigned to sentences labeled $k$. For a checkpoint, $p_k$ is the average of this share over its completions. The heatmap reports the signed change $100\big(p_k^{\text{after}} - p_k^{\text{base}}\big)$ in percentage points. The bar for each method is the total variation

$$D_{\text{TV}} = \frac{1}{2} \sum_{k=1}^{8} \left| p_k^{\text{after}} - p_k^{\text{base}} \right|, \tag{6}$$

the fraction of reasoning-token share reallocated across the eight episodes.

ConfSFT largely preserves the reasoning composition of the base policy it starts from. Its episode shares change only slightly, including on *Implement*, *Explore*, and *Verify*, so the shorter traces still allocate computation across episodes in nearly the same way. The same comparison shows why this is specific to ConfSFT: L1, ThinkPrune, and On-Policy SFT substantially redistribute computation across those episodes. ConfSFT reduces the amount of reasoning while keeping that allocation intact.

## AI USE DISCLOSURE

Generative AI tools were used to assist with writing polish and LaTeX formatting.