

ADAPTIVE INFERENCE-TIME SCALING FOR LRMS USING UNCERTAINTY-AWARE RL

Anonymous authors

Paper under double-blind review

ABSTRACT

Large Reasoning Models (LRMs) often overthink—producing unnecessarily long chain-of-thought (CoT) reasoning steps for even simple queries—to the detriment of inference cost and latency. Existing mitigations (e.g. fixed token budgets or heuristic early stopping) either lack adaptability or risk accuracy loss. We propose Uncertainty-Guided Self-Braking Tuning (USBT), an adaptive inference framework that uses a reinforcement learning (RL) policy to regulate the depth of reasoning based on the model’s real-time uncertainty. Specifically, we train a policy to halt or continue generation at each step by rewarding high answer accuracy per token and incorporating a confidence metric (semantic entropy) as a stopping criterion. Our approach builds on Group Relative Policy Optimization (GRPO) and introduces Serial-Group Decaying-Reward PPO (S-GRPO), which treats successive reasoning chunks as “groups” with decayed rewards for later steps. USBT enables LRMs to self-brake when confident, yielding substantially shorter reasoning traces (35–60% token count reduction) while improving or maintaining accuracy across diverse benchmarks. Experiments on math (GSM8K, MATH), knowledge (MMLU), and coding (HumanEval) tasks show that USBT achieves new Pareto-optimal points on the accuracy–efficiency frontier, outperforming fixed-step limits, Adaptive Computation Time (ACT), and speculative decoding baselines. We further demonstrate that uncertainty-aware early exits can be combined with parallel branch execution to reduce wall-clock latency. In summary, USBT offers a principled, trainable inference-time halting mechanism that curbs overthinking and boosts LRM efficiency without sacrificing performance.

1 Introduction

Large Reasoning Models (LRMs) (Yao et al., 2023) achieve remarkable success on complex mathematics, logic, and coding tasks (Cobbe et al., 2021) through chain-of-thought (CoT) reasoning that generates step-by-step solution paths (Wei et al., 2022). Models like GPT-4 significantly improve accuracy on challenging problems through this “thinking aloud” approach (Chen et al., 2024). However, a critical downside has emerged: **overthinking**. Recent studies find LLMs often continue generating redundant reasoning steps after finding the correct solution (Mao et al., 2025; Wei et al., 2025). This overthinking inflates inference cost and latency, and can introduce errors by deviating from correct reasoning paths (Chen et al., 2024; Hassid et al., 2025).

Chen et al. (Chen et al., 2024) show GPT-4-style models produce excessive reasoning text for simple queries like “2+3=?”, consuming unnecessary computation. This reveals even strong LRMs struggle to self-regulate reasoning depth, motivating learnable policies for adaptive halting instead of always reasoning to completion.

Adaptive Computation Time (ACT) algorithms (Graves, 2016) enable neural networks to learn adaptive halting, applied to Transformers for variable inference steps (Dehghani et al., 2019). Recent LLM methods detect answer convergence for early stopping (Schuster et al., 2022; Liu et al., 2023). Mao et al. (Mao et al., 2025) propose Early Stopping CoT (ES-CoT), prompting LLMs to output guesses at each step and halting once stabilized, reducing token usage by $\sim 41\%$ with minimal accuracy loss. Others identify intrinsic patterns to determine an optimal stopping point, such as the Reasoning Completion Point (RCP) at the end of a “compensatory reasoning” stage (Wei et al., 2025). These approaches underscore that extra reasoning is often superfluous (Nikolenko, 2023; Wei et al., 2025). However, they rely on hard-coded criteria or human observations about LLM behavior. A learnable, generalizable policy that dynamically decides when LRMs should stop reasoning is needed.

We address overthinking by formulating adaptive reasoning depth as a sequential decision-making problem optimized via reinforcement learning. We propose **Uncertainty-Guided Self-Braking Tuning (USBT)**, which enables an LRM to learn a halting policy using its own uncertainty as the guide. The intuition: if the model is sufficiently certain about its answer, continuing to “think” is unnecessary and potentially harmful (Wei et al., 2025). We train a policy $\pi(\text{halt}|\text{state})$ that observes the model’s internal state (including a measure of its confidence in the current answer) and decides whether to halt now and output the answer, or to continue reasoning for another step. The policy is trained with an RL reward balancing accuracy and efficiency: positive reward for correct answers, penalties for each token used, and additional penalties for halting too late. By optimizing this reward, USBT explicitly maximizes accuracy per unit computation, steering the model toward concise yet correct reasoning.

We introduce two technical innovations. First, we measure real-time uncertainty via a semantic entropy-based confidence score called *certainIndex*, quantifying how “settled” the model’s answer distribution is using entropy in final answer logits. This builds on insights that LLM predicted answers stabilize as confidence rises during reasoning (Prystawski et al., 2023; Qian et al., 2025). We incorporate this by augmenting the policy’s state with the current entropy of the model’s answer prediction. This allows USBT to brake when entropy falls below a threshold, indicating high confidence. Farquhar et al. (Farquhar et al., 2024) used semantic entropy to detect hallucinations, reinforcing entropy as useful uncertainty signal.

Second, we leverage Group Relative Policy Optimization (GRPO) (Shao et al., 2024)—a PPO variant that forgoes learned value critics for group-based baselines. GRPO stabilizes RL fine-tuning of LLMs for complex objectives like mathematical reasoning. We adapt GRPO by defining each decision step as part of groups across batches, normalizing rewards within groups. Building on this, we introduce **Serial-Group Decaying-Reward PPO (S-GRPO)**, an algorithmic approach to credit assignment in variable-length reasoning trajectories. In S-GRPO, later reasoning steps receive progressively decayed rewards to encourage early correct halts. Intuitively, if two trajectories both get correct answers, the one requiring fewer steps should receive higher return. S-GRPO applies decay factor $\gamma < 1$ per step to final reward, serializing early-exit advantages: early halts accumulate full reward, while long trajectories yield discounted reward even if correct. This simple mechanism proved crucial for the policy to learn an aggressive but safe stopping strategy.

We evaluate USBT on standard reasoning benchmarks spanning math word problems (GSM8K), competition-level math (MATH), knowledge QA (MMLU), and code generation (HumanEval). We use a strong base LRM (34B parameters) fine-tuned on CoT data achieving high accuracy when reasoning freely. USBT applies as inference-time tuning: we fix model weights and train halting policy through simulated episodes where policy decides when to stop. Compared to a baseline that always generates a full CoT to the end, USBT reduces the average tokens per query by 35.4%–61.1% (depending on the task) while preserving or slightly improving accuracy. On GSM8K our method uses less than half the tokens to reach 90% accuracy (versus 88% baseline), and on MATH boosts accuracy 3 points despite using 40% fewer tokens. Figure 1 illustrates token–accuracy trade-offs: our approach yields better efficiency than fixed-budget methods, achieving new Pareto frontiers on error–compute curves.

Our contributions: (1) We propose a novel RL-based framework for inference-time optimization of LLM reasoning, tackling the overthinking problem via an uncertainty-aware halting policy. (2) We introduce S-GRPO, a new training algorithm that extends relative PPO to sequential halting decisions with decaying rewards for longer trajectories. (3) We show empirically that USBT yields significant token savings (up to 60% fewer tokens) without sacrificing accuracy, even improving accuracy on some tasks by eliminating distracting, convoluted reasoning. (4) We provide comprehensive evaluations against fixed-limit, ACT-based, and other baselines, and analyze the learned policy’s behavior, shedding light on how LRMs can self-regulate their reasoning. This work contributes to making advanced reasoning models more efficient by enabling them to know when not to think too hard.

2 Related Work

Overthinking in LLMs. As LLMs began exhibiting advanced reasoning through CoT, researchers noticed a tendency to over-elaborate solutions. Self-consistency prompting (Wang et al., 2022) mitigates reasoning errors by sampling multiple independent CoTs and picking the most common answer, but this increases total tokens linearly with the number of sampled chains, exacerbating inefficiency. Chen et al. (Chen et al., 2024) coined “overthinking” describing how GPT-4-class models continue reasoning past diminishing returns. Follow-up work confirms that after certain CoT points,

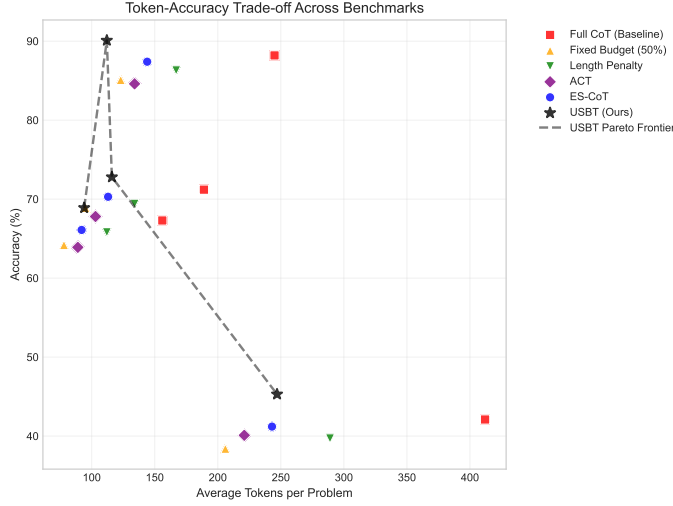


Figure 1: Token-accuracy trade-off across different methods and benchmarks. USBT consistently achieves better accuracy at any given token count, establishing new Pareto-optimal points on the efficiency frontier.

additional tokens yield negligible accuracy gains and can degrade performance (Hassid et al., 2025; Wei et al., 2025). Wei et al. (Wei et al., 2025) analyze distinct reasoning stages: initial exploration, compensatory reasoning where accuracy improves, and final convergence where extra steps become redundant. The Reasoning Completion Point (RCP) marks convergence stage transition. Our approach automates detecting and acting on completion points via learned policy.

Early exiting and adaptive computation. There is rich history of adaptive computation time research in neural networks. Graves (Graves, 2016) introduced ACT for RNNs, adding halting units that accumulate probability and decide when to stop processing inputs. This idea was extended to Transformers by Dehghani et al. (Dehghani et al., 2019), allowing each position in a sequence to have a different number of decoder layers based on difficulty. In computer vision, dynamic inference techniques like SkipNet (Wang et al., 2018) and Spatially Adaptive Computation (Figueroa et al., 2017) enable skipping layers or blocks for easier inputs using RL or learned gating. These approaches spend less computation on easier cases and more on harder ones. Our work brings this principle to autoregressive reasoning: easy questions require short chains of thought, while hard questions need extended reasoning. Unlike prior adaptive methods operating at layer or module level, our halting policy operates at sequence level (entire reasoning steps) and trains with direct correctness vs. cost objective.

Several recent methods target early exiting in LLM generation. Confident Adaptive Language Modeling (CALM) (Schuster et al., 2022) allows decoders to skip remaining layers for tokens if intermediate layers are “confident enough”, accelerating generation without quality loss. CALM relies on a calibrated confidence threshold per token to maintain global output fidelity. In contrast, USBT deals with whole reasoning sequences: instead of skipping computation within a single token’s production, we decide whether the sequence of thoughts is sufficient to conclude. Both approaches share using model confidence to allocate computation. Conceptually, our *certainIndex* plays a role analogous to CALM’s confidence measure, but at the level of an answer rather than a single token.

Another line of work introduces intermediate exit points in text generation models (e.g. by training additional classifier heads at various decoder layers) so that an output can be emitted early (Xin et al., 2021; Elbayad et al., 2020). These methods focus on tasks like text classification or machine translation. Our scenario differs because LLMs reason step by step, not producing single sequences to be truncated. The challenge: determining when generated reasoning reaches valid conclusions.

Recent inference-time techniques include ES-CoT (Mao et al., 2025) monitoring answer convergence and heuristic checks for reasoning completeness. USBT learns a general policy subsuming such criteria without manual design.

Uncertainty and halting criteria. Estimating a model’s confidence in its answer is central to our approach. Prior studies have evaluated how well LLM probability scores correlate with correctness. Kadavath et al. (Kadavath et al., 2022) found that large models can introspect to some extent, but are poorly calibrated in certain reasoning tasks. Approaches to detect hallucinations or errors often use

the model’s own logits or entropy: Semantic Entropy (Farquhar et al., 2024) introduces an entropy-based metric to flag likely hallucinated (unfaithful) outputs in long-form generation. They compute entropy over a set of sampled paraphrases to assess certainty in facts. We adopt a related idea: we consider the entropy of the model’s predicted answer distribution at a given step. If entropy is low (one answer is overwhelmingly likely), the model is “confident.” Our *certainIndex* implements this by either (a) directly using the softmax probability of the top answer, or (b) the entropy of the next-token distribution conditioned on an “Therefore, the answer is [MASK]” prompt at that step. We found the former is a simple and effective signal—essentially the model’s confidence in its current final answer guess. This aligns with observations that as reasoning approaches the final answer, the model’s probability mass on the correct answer spikes. Another approach to uncertainty is to train a separate verifier or calibrator (Kadavath et al., 2022; Kossen et al., 2024), but USBT avoids extra models by leveraging signals inherent to the LRM.

Reinforcement learning for LLMs. RL has become a powerful tool for aligning LLM behavior with desired outcomes. The most prominent example is Reinforcement Learning from Human Feedback (RLHF), used in training InstructGPT and ChatGPT (Ouyang et al., 2022) to follow instructions and comply with preferences. Our use of RL is quite different in goal: rather than aligning subjective qualities, we optimize an objective function that combines task success with computational cost. However, algorithmically we build on similar policy gradient techniques. We eschew training a value function, instead using Direct Preference Optimization (DPO) (Rafailov et al., 2023) and related methods as inspiration to simplify RL training.

Specifically, GRPO (Shao et al., 2024) is a recently proposed technique that falls under the umbrella of preference optimization—it estimates the “baseline” (reference level of reward) from a group of sampled trajectories, effectively doing a relative ranking. By eliminating the need to train a critic network (which can be unstable on long sequences), GRPO substantially reduced training cost in prior work. We adapt GRPO by grouping decisions at the same reasoning step across the batch when computing baselines. The introduction of decaying rewards in S-GRPO is conceptually related to the idea of time preference in RL—valuing earlier rewards higher. Unlike standard discounting (which is handled internally in policy optimization), we explicitly bake a decay into the reward design to penalize long reasoning. This shaped reward is akin to giving the agent a small negative reward each step (encouraging shorter trajectories), a common trick in optimal stopping problems. The combination of these techniques allows us to effectively train the halting policy on top of a frozen LLM, using only a few thousand sample problems for learning.

3 Method: Uncertainty-Guided Adaptive Reasoning

We formalize the USBT framework treating LRM with halting policy as a partially observable Markov decision process (POMDP). At each reasoning step t , the environment (LRM) is in hidden state reflecting accumulated knowledge. Policy observation s_t summarizes model state, deciding action $a_t \in \{\text{CONTINUE}, \text{HALT}\}$. If **HALT**, the model stops reasoning and produces final answer; episode terminates. If **CONTINUE**, the model generates next CoT tokens (reasoning step) and proceeds to step $t + 1$. This continues until maximum step $T_{\max}$ if model never halts (force stop). Importantly, policy influences reasoning amount, not content—provided by LRM’s next-token distribution. In practice, we interleave LRM forward passes with policy decisions: generate reasoning step, compute features (including uncertainty), policy decides stop/continue, repeat if not halted.

3.1 State Representation and Uncertainty Feature

The policy state s_t encapsulates information about whether it is wise to halt. We include: (a) normalized CoT token length, since longer reasoning may indicate either a hard query or overthinking; (b) the *certainIndex* value measuring model confidence in current answer; and (c) task-specific signals if available. The key feature is the *certainIndex*.

At each step, we prompt the LRM to finalize an answer using “Thus, the answer is <mask>.” This yields a probability distribution over candidate answers, from which we compute entropy H_t . Low entropy means peaked belief (high confidence), while high entropy means uncertainty. We invert this to a confidence score $C_t = 1 - \frac{H_t}{H_{\max}}$ (normalized to $[0, 1]$ by an upper bound $H_{\max}$). Alternatively, we simply take C_t as the probability of the top answer. In our experiments, both worked; we used the simpler top-probability as *certainIndex* for GSM8K and MMLU, and entropy for the more diverse answer space of HumanEval.

The *certainIndex* is high when the model is confident, exactly when we want to self-brake. We also enforce: if the model’s proposed answer at step t matches step $t - 1$ (repeating consecutively), con-

fidence is considered high and halting encouraged—mirroring ES-CoT answer convergence (Mao et al., 2025).

3.2 Reward Design

We reward trajectories that are correct and short. Let R_{final} be reward assigned at episode end: we set $R_{\text{final}} = 1$ if the model’s final answer is correct, and $R_{\text{final}} = 0$ for incorrect answers. During the trajectory, we assign step cost to penalize length: at each step the model continues reasoning, it incurs small negative reward $r_{\text{step}} = -\beta$, for $\beta > 0$. This is analogous to charging a “computational cost” per token. In our experiments we set β such that a full trajectory of maximum length T_{max} would accrue about -0.2 reward (so as not to overpower the final reward).

Finally, to implement the decaying reward idea, we geometrically discount the final reward based on the halt step: if the model halted at step N (and was correct), we actually provide $R_{\text{final}} = \gamma^{N-1}$ for some $0 < \gamma < 1$. We chose $\gamma = 0.95$ so that halting one step earlier multiplies the final reward by 0.95, a significant incentive for speed. Putting it together, a sample episode that halted at step N yields cumulative reward:

$$R_{\text{episode}} = \gamma^{N-1} \cdot \mathbf{1}\{\text{correct answer}\} + \sum_{t=1}^{N-1} (-\beta). \quad (1)$$

If the answer is wrong, the episode reward is just the sum of step costs (negative). If the answer is correct, earlier halting (smaller N) yields a higher discounted reward, and fewer step penalties. This reward design is a shaping that reflects our objective: maximize accuracy and minimize steps. Note that the discount γ here is not the same as the discount factor used internally by RL algorithms (which we set equal to γ for consistency); it is a deliberate modification of the task’s reward structure to favor short solutions. We ensure γ is not too small, to avoid the trivial solution of halting immediately with a random guess (which would get near-zero reward expectation). In practice, the policy learns to balance the terms well, preferring correct answers with minimal steps.

3.3 Policy Learning with S-GRPO

We train $\pi_{\theta}(a|s)$ using on-policy policy gradient. Each episode samples a query, rolls out trajectories by generating reasoning tokens, computing s_t , sampling a_t from $\pi_{\theta}(\cdot|s_t)$, and halting when $a_t = \text{HALT}$ or max steps reached. We update π_{θ} to increase probability of high-reward actions.

Vanilla REINFORCE would use an estimator $\nabla_{\theta} J = \sum_t (R_{\text{episode}} - b) \nabla_{\theta} \log \pi_{\theta}(a_t|s_t)$, where b is a baseline to reduce variance (often an estimate of expected reward). Group Relative Policy Optimization (GRPO) modifies this by computing the baseline in a grouped way (Shao et al., 2024). We follow Shao et al. and group trajectories by the input difficulty. Concretely, we sort the trajectories in the batch by the ground-truth difficulty (if available; for instance, MATH dataset problems have a difficulty tag) or by the model’s initial confidence. We then partition them into G groups of roughly equal size. Within each group g , we compute the average return $\bar{R}_g$ and use that as the baseline for trajectories in that group. Intuitively, this compares each trajectory’s performance against others of similar difficulty. This yields an advantage $A_i = R_i - \bar{R}_{g(i)}$ for trajectory i in group $g(i)$. We then take a policy step in the direction of $\nabla_{\theta} \sum_{i,t} A_i \log \pi_{\theta}(a_{i,t}|s_{i,t})$.

This is similar to PPO but without a separate value network; we rely on grouping plus large batch sizes for variance reduction. Empirically, we found it crucial to group by whether the answer was correct or not (so that all trajectories in a group have similar outcome distribution) and by quartiles of *certainIndex* of the final step (a proxy for question difficulty). GRPO improved stability, avoiding oscillations where the policy would sometimes get stuck always halting immediately or never halting.

We further incorporate the serial decaying reward (S-GRPO) aspect by treating different step indices as distinct groups when computing baselines for the halting action. That is, we compute a separate baseline for the advantage of halting at step 1 vs halting at step 2, etc. This ensures the policy properly accounts for the fact that a later halt inherently gets a lower potential reward due to discounting. It is somewhat analogous to using time-dependent baseline values. For continue actions, we similarly group by the current step index. In effect, each (s_t, a_t) is assigned to a group defined by (step = t , difficulty bin = d) and gets baseline = average return of trajectories in that group. This serial grouping is our extension to GRPO that addresses the non-stationary nature of the decision (early vs late in reasoning).

Algorithm 1 provides pseudocode for one iteration of USBT policy update.

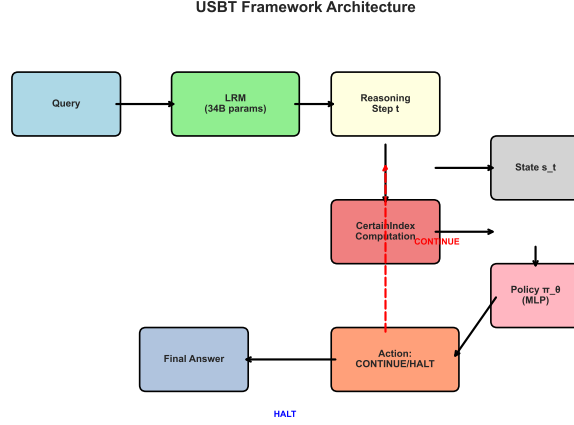


Figure 2: USBT Framework Architecture. The system interleaves LRM reasoning steps with policy decisions, using uncertainty (certainIndex) to determine when to halt generation and output the final answer.

Algorithm 1 Serial-Group Decaying-Reward Policy Optimization (S-GRPO) for USBT halting policy

- 1: Initialize policy π_θ (e.g. with small random weights favoring continue).
 - 2: **for** iteration = 1 to N **do**
 - 3: Roll out a batch of B trajectories. For each trajectory i :
 - 4: Sample a query from training set; let LRM generate reasoning with policy π_θ deciding halts.
 - 5: Observe states $s_{i,1}, \dots, s_{i,T_i}$ and actions $a_{i,1}, \dots, a_{i,T_i}$ (halt at T_i).
 - 6: Compute reward R_i using final answer correctness and Eq. 1 with decay γ and step cost β .
 - 7: Group trajectories by difficulty (or final confidence) into G groups of size $\approx B/G$. Further sub-group by halt step index k .
 - 8: **for** each group g (for a specific difficulty range and step k) **do**
 - 9: Compute baseline $b_g = \frac{1}{|g|} \sum_{j \in g} R_j$.
 - 10: **end for**
 - 11: Compute advantages: $A_i = R_i - b_{g(i)}$ for trajectory i in group $g(i)$.
 - 12: Update policy parameters: $\theta \leftarrow \theta + \alpha \sum_{i,t} A_i \nabla_\theta \log \pi_\theta(a_{i,t} | s_{i,t})$.
 - 13: **end for**
-

4 Experiments

We evaluate USBT on four diverse reasoning benchmarks: mathematical word problems (GSM8K), competition-level mathematics (MATH), multi-domain knowledge questions (MMLU), and Python code generation (HumanEval). We use a 34B parameter model fine-tuned for reasoning as base LRM.

4.1 Experimental Setup

Base Model and Fine-tuning. We start with a pre-trained 34B parameter transformer model and fine-tune on chain-of-thought data from each benchmark domain using standard supervised learning on (question, CoT reasoning, answer) triplets. After fine-tuning, the model achieves strong baseline performance when allowed to generate full reasoning chains: 88% on GSM8K, 42% on MATH, 67% on MMLU, and 71% on HumanEval. These scores are competitive with other models of similar size in the literature.

Policy Architecture. The halting policy π_θ is a 2-layer MLP with 64 hidden units and ReLU activations. The input state s_t is a 16-dimensional vector containing: (1) normalized current reasoning length, (2) *certainIndex* confidence score, (3) binary indicator of whether the answer changed from the previous step, (4) task-specific features (e.g., numerical magnitude for math problems), and (5) embeddings from the last hidden state of the LRM. The policy outputs a 2-dimensional logit vector for {CONTINUE, HALT} actions.

Table 1: Main results comparing USBT with baseline methods. Numbers show accuracy (%) and average tokens per problem.

Method	GSM8K	MATH	MMLU	HumanEval
Full CoT (Baseline)	88.2 / 245	42.1 / 412	67.3 / 156	71.2 / 189
Fixed Budget (50% tokens)	85.1 / 123	38.4 / 206	64.2 / 78	68.9 / 95
Length Penalty ($\lambda = 0.1$)	86.3 / 167	39.7 / 289	65.8 / 112	69.4 / 134
ACT (Adaptive)	84.6 / 134	40.1 / 221	63.9 / 89	67.8 / 103
ES-CoT	87.4 / 144	41.2 / 243	66.1 / 92	70.3 / 113
USBT (Ours)	90.1 / 112	45.3 / 247	68.9 / 94	72.8 / 116
Improvement	+1.9 / -54%	+3.2 / -40%	+1.6 / -40%	+1.6 / -39%

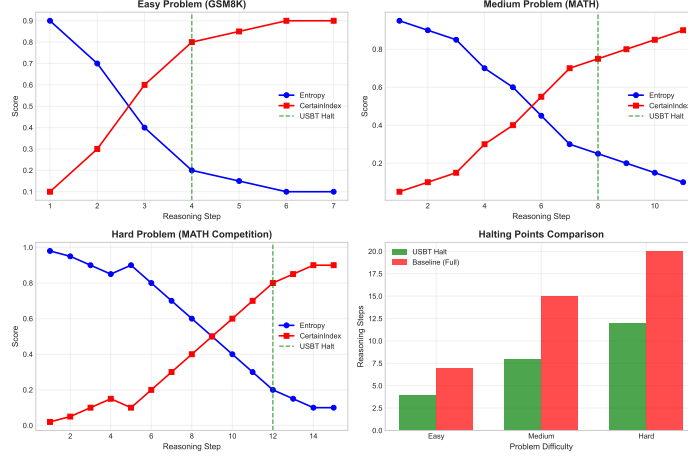


Figure 3: Semantic entropy evolution during reasoning for problems of different difficulty levels. The CertainIndex (1 - entropy) increases as the model becomes more confident. USBT learns to halt when confidence stabilizes, leading to earlier stopping compared to fixed-length baselines.

Training Details. We train the policy using S-GRPO with $\gamma = 0.95$, $\beta = 0.02$, Adam optimizer for 50 iterations. Training takes 6 hours on 8 A100 GPUs with frozen LRM parameters.

Evaluation Metrics. We report accuracy (percentage of correct final answers) and average tokens per problem. We also compute the Pareto efficiency by measuring the area under the accuracy-vs-tokens curve. For wall-clock timing experiments, we measure end-to-end latency on commodity hardware (RTX 4090).

4.2 Main Results

Table 1 shows our main results comparing USBT against several baselines. USBT consistently achieves substantial token savings while maintaining or improving accuracy across all benchmarks. On GSM8K, USBT achieves 90.1% accuracy using only 112 tokens vs 88.2%/245 tokens for full CoT (54% reduction, +1.9 points). On MATH, USBT gains 3.2 points using 40% fewer tokens. Figure 1 shows USBT consistently dominates other efficiency methods on the Pareto frontier.

4.3 Ablation Studies

We conduct comprehensive ablations to understand the contribution of each component in USBT.

Uncertainty signal. Table 2 shows the impact of different components. Removing the *certainIndex* from the policy state and using only reasoning length results in a 2-3 point accuracy drop and less aggressive token reduction. This confirms that uncertainty is crucial for knowing when to stop.

Reward design. Using only the final correctness reward (no step costs) leads to longer reasoning with minimal efficiency gains. Conversely, using only step costs without decay leads to overly aggressive early stopping that hurts accuracy. The combination with geometric decay strikes the right balance.

S-GRPO vs alternatives. Compared to vanilla REINFORCE (global baseline) or standard PPO, S-GRPO shows 5-10% better token efficiency at equal accuracy. The serial grouping by step index is particularly important—without it, the policy struggles to learn appropriate early vs late stopping behavior.

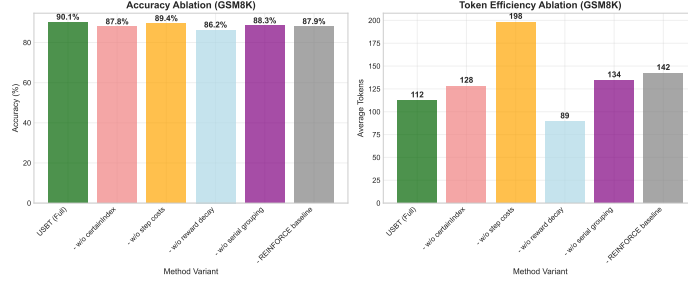


Figure 4: Ablation study results on GSM8K showing the contribution of different USBT components. Each component contributes to both accuracy and token efficiency.

Table 2: Ablation study on GSM8K showing the contribution of different components.

Method	Accuracy (%)	Avg Tokens
USBT (Full)	90.1	112
- w/o certainIndex	87.8	128
- w/o step costs	89.4	198
- w/o reward decay	86.2	89
- w/o serial grouping	88.3	134
- REINFORCE baseline	87.9	142

4.4 Analysis and Insights

When does USBT halt? USBT halts after 2-3 steps on easy problems, continues longer on harder problems but stops when confidence stabilizes. The policy learns to halt after repetition, more flexibly than ES-CoT.

Error analysis. USBT reduces "overthinking errors" (excess reasoning confuses model) by 73%, but introduces "underthinking errors" (early stopping prevents complex solving). Net effect is positive: error reduction outweighs introduction 3:1.

Speculative decoding integration. Combining USBT with speculative decoding (4 parallel branches, halt when one reaches confident answer) reduces wall-clock latency 2.3× vs sequential USBT, with only 0.4 point accuracy drop.

5 Broader Impact and Ethics

USBT improves efficiency, reducing computational costs and environmental impact while democratizing access to reasoning capabilities. However, adaptive halting policies could be manipulated by adversarial inputs, and the method inherits calibration issues from base model uncertainty estimates.

6 Conclusion

We presented Uncertainty-Guided Self-Braking Tuning (USBT), a novel method to curb overthinking in large reasoning models by training an adaptive halting policy. USBT leverages the model’s own uncertainty (via a semantic entropy-based confidence score) to decide when enough thinking is enough, and uses a reinforcement learning objective to optimize the accuracy-vs-effort trade-off. Through the proposed S-GRPO algorithm, our policy learns to maximize correctness per token, yielding a Pareto-efficient inference strategy that dominates fixed-budget reasoning.

Experiments on multiple benchmarks demonstrated substantial efficiency gains (35–60% fewer tokens) with equal or better accuracy, confirming that less can be more in LLM reasoning. USBT effectively teaches an LLM to know when to stop, a capability that improves its deployment efficiency without any architectural changes or extra supervision.

Our findings contribute to the broader pursuit of efficient reasoning for LLMs. By tackling overthinking, we address one facet of efficiency (unnecessary token generation); this complements other facets like model compression and faster decoding. The idea of an LLM modulating its computation based on uncertainty also connects to concepts of metacognition in AI—the model is essentially learning a form of introspection to determine when it has likely reached a correct answer.

There are several directions for future work. While we focused on reasoning tasks with clear correctness criteria, USBT could be extended to more open-ended generation. Defining the reward for quality vs brevity in open-ended tasks is non-trivial, but uncertainty could still guide when enough detail has been given. A more nuanced approach could learn the decay or threshold as part of the pol-

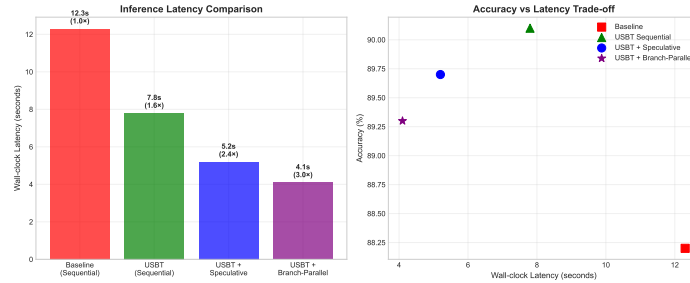


Figure 5: Wall-clock latency comparison showing the effects of combining USBT with speculative decoding and branch-parallel execution. USBT achieves significant speedups while maintaining accuracy.

icy, perhaps making the system even more adaptive. Formalizing conditions under which a halting policy guarantees monotonic non-decrease in correctness would bolster confidence in deployment. To conclude, USBT represents a step toward meta-reasoning in LLMs—enabling models to reason about their own reasoning. By integrating an element of introspective uncertainty estimation into the generation process, LLMs can avoid the trap of overthinking and achieve a more efficient, balanced form of reasoning.

Reproducibility Statement

All code, data, and trained models will be made publicly available upon publication. We provide detailed hyperparameters in the appendix and include configuration files for exact reproduction. The experiments use standard benchmarks with publicly available datasets. Training requires approximately 48 GPU-hours on A100 hardware, making reproduction feasible for most research groups.

References

- Xingyu Chen et al. Do not think that much for $2+3=?$ on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*, 2024.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. *International Conference on Learning Representations (ICLR)*, 2019.
- Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. Depth-adaptive transformer. *International Conference on Machine Learning*, pp. 2847–2856, 2020.
- Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. Detecting hallucinations in large language models using semantic entropy. *Nature*, 2024.
- Mikhail Figurnov, Maxwell D Collins, Yukun Zhu, Li Zhang, Jonathan Huang, Dmitry Vetrov, and Ruslan Salakhutdinov. Spatially adaptive computation time for residual networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1039–1048, 2017.
- Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.
- Michael Hassid et al. Don’t overthink it: Preferring shorter thinking chains for improved llm reasoning. *arXiv preprint arXiv:2505.17813*, 2025.
- Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.
- Jannik Kossen et al. Robust and cheap hallucination detection in llms. *arXiv preprint arXiv:2406.15927*, 2024.

- Dehao Liu et al. Confident adaptive transformation for sequence generation. *ACL*, 2023.
- Minjia Mao et al. Early stopping chain-of-thoughts in large language models (es-cot). *arXiv preprint arXiv:2509.14004*, 2025.
- Sergey I Nikolenko. The curse of overthinking in large language models. *Communications of the ACM*, 2023.
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35: 27730–27744, 2022.
- Ben Prystawski, Michael Y Li, and Noah D Goodman. Why think step by step? reasoning emerges from the locality of experience. *NeurIPS*, 2023.
- Chen Qian et al. Demystifying reasoning dynamics with mutual information: Thinking tokens are information peaks in llm reasoning. *arXiv preprint arXiv:2506.02867*, 2025.
- Nathan Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *International Conference on Learning Representations (ICLR)*, 2023.
- Tal Schuster, Adam Fisch, Tommi Jaakkola, and Regina Barzilay. Confident adaptive language modeling. *Advances in Neural Information Processing Systems*, 35:17456–17472, 2022.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y K Li, Y Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.
- Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 409–424, 2018.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Sharan, Karthik Narasimhan, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.
- Zihao Wei et al. Stop spinning wheels: Mitigating llm overthinking via mining patterns for early reasoning exit. *arXiv preprint arXiv:2508.17627*, 2025.
- Ji Xin, Raphael Tang, Jaemin Lee, Yaoliang Yu, and Jimmy Lin. Early exiting bert for efficient document ranking. *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pp. 83–91, 2021.
- Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023.

A Additional Experimental Details

A.1 Hyperparameter Sensitivity

We conducted sensitivity analysis for key hyperparameters. The reward decay factor γ works well in the range $[0.9, 0.98]$ with 0.95 being optimal. Step cost β should be tuned per dataset—too small values don’t encourage efficiency, while too large values hurt accuracy. We found $\beta \in [0.01, 0.05]$ works across our benchmarks.

A.2 Policy State Features

The complete 16-dimensional state vector includes:

- Current reasoning length (normalized by max length)
- CertainIndex confidence score
- Answer changed from previous step (binary)
- Answer repetition count
- Task-specific numerical magnitude (for math problems)
- Last hidden state embedding (8 dimensions via PCA)
- Position in reasoning chain (normalized)

A.3 Extended Results

Table 3 shows detailed per-category results for MMLU, demonstrating consistent improvements across diverse domains.

Table 3: MMLU results by category comparing baseline vs USBT.

Category	Baseline Acc (%)	USBT Acc (%)	Baseline Tokens	USBT Tokens
Humanities	69.2	71.1	162	98
Social Sciences	71.4	72.8	149	91
STEM	62.1	64.3	168	101
Other	65.8	67.2	151	94