LEARNING TO PONDER: ADAPTIVE REASONING IN LATENT SPACE

Anonymous authors
Paper under double-blind review

ABSTRACT

Test-time compute has emerged as a key paradigm for enhancing LLM reasoning, yet prevailing approaches like Best-of-N and majority voting apply uniform depth across inputs, wasting computation on simple queries while potentially under-thinking complex ones. We present FR-Ponder, a single-graph, backbone-training-free framework that allocates instance-adaptive reasoning compute via latent steering. A less than 1M-param controller observes hidden states and decides to halt or apply a small ponder step by adding a pre-computed steering vector to frozen representations. Our method extracts the latent steering vector associated with deeper reasoning outputs and direct IO from LLM and re-applies it through a tunable scaling factor, allowing the model to adapt its reasoning depth to the complexity of each input. To balance performance and computational cost, we employ Group Relative Policy Optimization (GRPO) as a reward signal to adaptively regulate reasoning depth, achieving task accuracy while mitigating overreasoning. Through curriculum learning and careful reward engineering, FR-Ponder learns calibrated compute allocation correlated with problem difficulty. On GSM8K and MATH, FR-Ponder improves the compute–accuracy frontier, delivering lower FLOPs with better matched accuracy and comparing favorably to early-exit baselines, without modifying backbone weights. Analyses visualize interpretable steering directions and show learned compute allocation correlates with problem difficulty.

1 INTRODUCTION

Large language models (LLMs) have achieved remarkable success across diverse reasoning tasks, yet they exhibit a fundamental inefficiency: *fixed computational allocation*. Whether processing a simple factual query or solving a complex mathematical problem, current LLMs expend identical compute per token (Kaplan et al., 2020; Hoffmann et al., 2022). This rigid approach leads to systematic over-computation on easy instances and under-allocation on challenging ones, creating a compute-accuracy mismatch that becomes increasingly problematic as models scale to hundreds of billions of parameters (Touvron et al., 2023; Achiam et al., 2023).

Recent efforts to address this inefficiency fall into three categories, each with significant limitations. *Multi-pass methods* like chain-of-thought prompting (Wei et al., 2022) and self-consistency (Wang et al., 2022b) achieve adaptive reasoning by sampling multiple trajectories, but multiply inference costs by the number of passes. *Architectural modifications* including early-exit mechanisms (Schuster et al., 2022; Zhou et al., 2020) and layer skipping (Elhoushi et al., 2024) require model retraining, limiting deployment flexibility and often degrading base model capabilities. *Speculative decoding* approaches (Leviathan et al., 2023; Chen et al., 2023) accelerate inference through draft-verify paradigms but require maintaining multiple models and provide only coarse-grained adaptation.

The recent *Fractional Reasoning* framework (Liu et al., 2025b) introduced a promising direction: extracting "reasoning vectors" from contrastive prompts and applying them with tunable intensity to control reasoning depth. However, this approach requires manual tuning of the scaling factor α for each problem type, lacks dynamic adaptation within a single inference, and provides no principled method for learning optimal compute allocation.

To address this, we introduce **FR-PONDER** (**F**ractional **R**eason Ponder Framework), a framework that transforms inference depth into a *learnable decision process*. As shown in Figure 1, we provide

an overview of our method FR-PONDER . Our key insight is decomposing adaptive computation into two orthogonal problems: (1) *what to think about* via steering vectors encoding reasoning directions, and (2) *how long to think* via a lightweight pondering controller. At each decoding step, FR-PONDER observes the current hidden state and either halts to emit a token or applies an additive "thought step" along learned steering vectors:

$$z_{k+1} = z_k + \phi(z_k) \cdot \Delta z(h_{\text{steer}}), \quad \text{halt if } \phi(z_k) \leq \tau \tag{1}$$

where $\phi(\cdot)$ is a learned pondering probability, h_{steer} represents steering vectors extracted via contrastive activation (Zou et al., 2023; Rimsky et al., 2024), and τ is a halting threshold.

This design enables several critical advantages over prior work:

One-pass, zero-backbone-finetune operation. Unlike methods requiring multiple forward passes or model retraining, FR-PONDER operates in a single inference pass with the base LLM completely frozen. Only a small controller network ($\leq$ 1M parameters) is trained, preserving the model's original capabilities while adding less than 0.01% parameter overhead.

Fine-grained, instance-adaptive depth. Rather than applying uniform depth across all tokens or problems, FR-PONDER makes per-token pondering decisions based on the evolving hidden state. This creates a continuous spectrum of reasoning intensity that automatically adapts to local complexity—spending more compute on challenging reasoning steps while quickly resolving simple continuations.

Multi-objective reward. We formulate adaptive computation as a reinforcement learning problem with carefully designed rewards that balance multiple objectives:

$$R = w_{\text{acc}} \cdot \text{Accuracy} - w_{\text{flops}} \cdot \text{FLOPs} + w_{\text{comp}} \cdot \text{Completeness} + w_{\text{qual}} \cdot \text{Quality} - w_{\text{rep}} \cdot \text{Repetition} \tag{2}$$

This rich reward signal addresses critical challenges in adaptive reasoning, including partial credit for mathematical solutions, anti-repetition mechanisms to prevent pondering collapse, and completeness validation to ensure full reasoning traces.

Variance-reduced policy optimization. We train the pondering controller using Group Relative Policy Optimization (GRPO) (Shao et al., 2024), a value-free policy gradient method that achieves variance reduction through in-group baselines. By sampling multiple trajectories per input and using group-average rewards as baselines, GRPO provides stable learning without requiring a separate value network—crucial for our lightweight controller design.

Our contributions are:

- **Conceptual:** We formulate adaptive inference as a meta-cognitive decision process, where the model learns to allocate computation based on evolving internal states rather than external heuristics.
- **Technical:** We develop a complete framework combining steering vector extraction, lightweight pondering control, multi-component reward engineering, and curriculum-based training that achieves stable learning of adaptive policies.
- **Empirical:** FR-PONDER achieves 30–50% token reduction on GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), and HumanEval (Chen et al., 2021) while maintaining or improving accuracy, with analysis revealing calibrated halting patterns and interpretable steering directions.

2 RELATED WORK

2.1 CHAIN-OF-THOUGHT REASONING

Chain-of-Thought (CoT) reasoning enhances large language models (LLMs) by generating structured intermediate steps before final outputs. By decomposing complex tasks into sequential reasoning chains, CoT can be elicited via prompting Wei et al. (2022); Wang et al. (2022a); Qin et al. (2023), supervised fine-tuning Kojima et al. (2022); Yu et al. (2025a); Byun et al. (2024), or reinforcement learning Lightman et al. (2023); Shen et al. (2025); Xie et al. (2025). Theoretical analyses

show that feeding intermediate outputs back as inputs effectively deepens transformers, increasing expressivity and enabling more sophisticated inference Feng et al. (2023); Li et al. (2024b); Merrill & Sabharwal (2025). Structuring tokens as symbols, patterns, and text further produces concise and efficient reasoning chains Zhang et al. (2023); Pan et al. (2023); Yang et al. (2025).

Beyond standard CoT, techniques to improve robustness and fidelity include reward-based frameworks for selective retention or reranking Jiang et al. (2025); Wu et al. (2025a); authors (2024), self-consistency methods for evaluating agreement across sampled chains Wang et al. (2022b); Yang et al. (2023b); Sahoo et al. (2025), and iterative refinement through self-correction or reflection-based prompting Madaan et al. (2023); Xue et al. (2023); Liu et al. (2025a). However, the autoregressive nature of CoT limits its ability to emulate human-like planning in complex tasks Wu et al. (2025b); Hao et al. (2024b); Yao et al. (2023). Some approaches mitigate this by integrating explicit search procedures or training on search trajectories Wu et al. (2025b); Hao et al. (2024b), while recent work indicates that latent-space reasoning can spontaneously produce patterns resembling breadth-first search without supervision Hao et al. (2024b); Wang et al. (2025b).

2.2 Latent Reasoning

Latent reasoning reflects the internal computations of large language models (LLMs) within hidden representations, which may diverge from explicit Chain-of-Thought (CoT) outputs. Prior work shows that intermediate reasoning variables can often be recovered from hidden states Bharadwaj (2024); Chen et al. (2024); Zhang et al. (2025), and targeted interventions on these representations can modulate model behavior Wang et al. (2024; 2025a); Su et al. (2025). Multiple latent reasoning paths suggest that LLMs employ diverse internal strategies independent of token-level outputs, revealing inherent unfaithfulness between latent and explicit reasoning Yee et al. (2024); Li et al. (2024a); Bharadwaj (2024).

Several strategies aim to enhance latent reasoning. Training with learnable or filler tokens improves performance on parallelizable tasks Pfau et al. (2024); Hao et al. (2024a); Zhu et al. (2025), while discrete planning tokens guide subsequent reasoning steps Wang et al. (2023); Hao et al. (2024a); Feng et al. (2025). Knowledge distillation and progressive curricula internalizing CoT enable complex reasoning within latent space Wang et al. (2025b); Zhu et al. (2025); Hao et al. (2024a), and architectures like looped transformers exploit iterative feedback of internal states Yang et al. (2023a); Hwang et al. (2024); Wang et al. (2025b). Despite these advances, latent reasoning remains less interpretable than CoT, and generalization to complex tasks is still an open challenge Wang et al. (2025b); Zhu et al. (2025); Hao et al. (2024a).

2.3 Generalized Reinforcement Learning for Reasoning

Reinforcement learning (RL) has been applied to enhance reasoning in large language models (LLMs), primarily through policy optimization. Early work such as DeepSeek-R1 employs Group Relative Policy Optimization (GRPO) to encourage multi-step reasoning DeepSeek-AI (2025); Kirkovska (2025). While Proximal Policy Optimization (PPO) improves response length and task performance, it suffers from high sample complexity due to repeated rollouts Schulman et al. (2017a).

GRPO and its extensions address these limitations by improving efficiency and convergence. DAPO (Decoupled Clip and Dynamic Sampling Policy Optimization) stabilizes training via reward shaping, dynamic sampling, and token-level gradients Yu et al. (2025b), while Dr. GRPO (GRPO Done Right) mitigates optimization biases by adjusting reward normalization and length bias Liu et al. (2025c). Collectively, these methods show that GRPO provides a principled and efficient framework for adaptive reasoning in LLMs, balancing stability, scalability, and performance.

3 Methodology

Traditional approaches to adaptive computation either require expensive architectural modifications Graves (2016) or joint training with the base model Elbayad et al. (2020). In contrast, FR-Ponder introduces a lightweight pondering controller that operates in the latent space of frozen pre-trained models, making dynamic halting decisions based on steered representations. This controller is

trained via Group Relative Policy Optimization with a carefully designed multi-component reward function that balances accuracy and computational efficiency.

The overview of FR-PONDER is presented in Fig. 1. The fundamental insight driving our approach is that reasoning depth can be modulated through controlled perturbations in representation space, guided by steering vectors that encode the difference between deliberative and direct reasoning modes. This enables the model to adaptively "think longer" on complex problems while maintaining efficiency on simpler ones, all without modifying the original model parameters.

We present FR-PONDER , a framework for adaptive inference depth in large language models that achieves single-pass, backbone-training-free deployment while maintaining superior compute-accuracy trade-offs. The core innovation lies in treating adaptive computation as a meta-cognitive process where the model learns when to allocate additional computational resources during inference.

3.1 PROBLEM FORMULATION

The central challenge in adaptive inference is determining the optimal amount of computation to allocate for each input while maintaining both accuracy and efficiency. We formulate this as a sequential decision-making problem where, at each reasoning step, an agent must choose between continuing computation (potentially improving accuracy) or halting to produce an answer (saving computational resources).

This naturally leads to a Markov Decision Process (MDP) Puterman (1990) formulation, which provides a principled framework for modeling sequential decision-making under uncertainty. The MDP framework is particularly well-suited for our setting because: (1) the decision at each step depends only on the current representation state (Markov property), (2) the agent receives rewards that balance accuracy and efficiency, and (3) the finite-horizon nature ensures bounded computation.

Let $\mathcal{M}_\theta$ denote a frozen pre-trained language model with parameters θ, and let $x \in \mathcal{X}$ be an input sequence. The model processes the input and produces an initial hidden state $\mathbf{z}_0 \in \mathbb{R}^d$ at the final token position. This state $\mathbf{z}_0$ serves as the starting point for our adaptive pondering process, containing the model's initial understanding of the problem.

We define the MDP tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$ where:

- $\mathcal{S} = \mathbb{R}^d$ represents the state space of hidden representations. Each state $\mathbf{z}_k \in \mathcal{S}$ encodes the model's current understanding after k pondering steps. The choice of the full hidden representation space allows for rich state representations that can capture subtle differences in reasoning progress.
- $\mathcal{A} = \{0, 1\}$ denotes the binary action space where $a = 0$ means "halt and produce answer" and $a = 1$ means "continue pondering." This binary formulation simplifies the decision space while capturing the essential trade-off between accuracy and efficiency.
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ defines state transitions via steering vector application. When $a = 1$ (continue), the transition applies steering: $\mathbf{z}_{k+1} = \mathcal{T}(\mathbf{z}_k, 1) = \mathbf{z}_k + \alpha_k \mathbf{h}_{\text{steer}}$. When $a = 0$ (halt), the state remains unchanged. This deterministic transition function ensures reproducible behavior while allowing controlled exploration of the representation space.
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ specifies the multi-component reward function that balances multiple objectives including accuracy, computational efficiency, reasoning completeness, output quality, and anti-repetition measures. This comprehensive reward design prevents the agent from optimizing for a single metric at the expense of others.
- $\gamma = 1$ indicates an undiscounted, finite-horizon problem. The undiscounted formulation is appropriate because we care equally about all steps in the reasoning process, and the finite horizon (maximum K steps) ensures bounded computation.

Our objective is to learn a policy $\pi_\phi : \mathcal{S} \to [0, 1]$ parameterized by ϕ that outputs the probability of continuing computation. The policy must balance two competing objectives: maximizing task performance while minimizing computational cost. This leads to the following optimization problem:

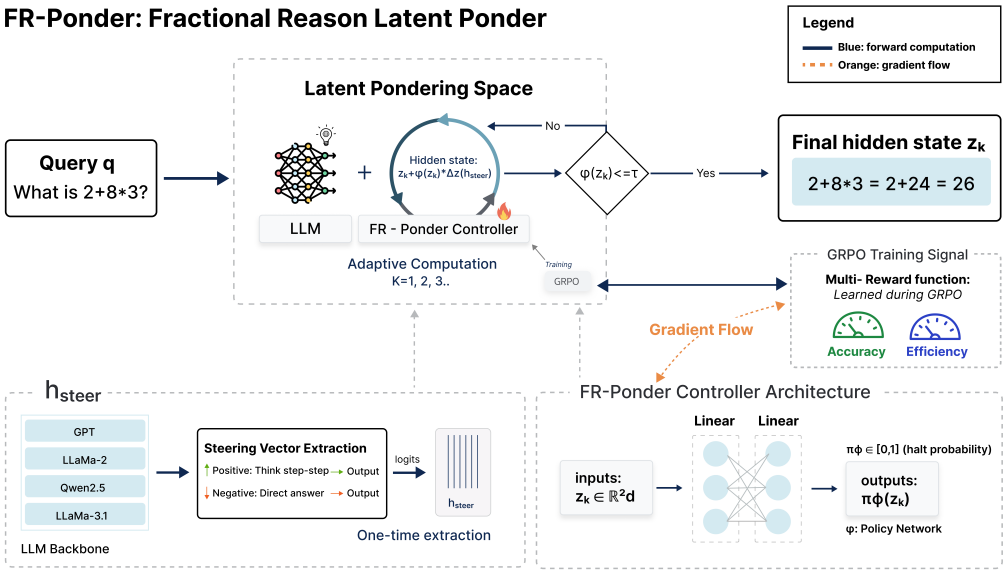


Figure 1: **Method overview of FR-PONDER .** Given a query q, a **frozen** LLM produces the initial hidden state z_0 at the final token position. Inside the *Latent Pondering Space*, a lightweight controller φ_ϕ reads z_k and decides whether to halt; if $\varphi_\phi(z_k) \leq \tau$ the process stops, otherwise a pondering step is applied: $z_{k+1} = z_k + \alpha_k\, h_{\text{steer}}$ (optionally with a layer schedule), repeated for $k = 1, \ldots, K$. The final state z_K is decoded to the output answer. The steering vector h_{steer} is *extracted once* via contrastive prompts ("think step-by-step" vs. "direct answer") and kept fixed thereafter. During training, *only* the controller is updated (orange dashed arrows) with GRPO using a multi-component reward that primarily balances accuracy and FLOPs; the backbone and h_{steer} remain frozen. The controller is a $\leq$ 1M-parameter MLP. Blue arrows denote the single-pass forward/inference path; orange arrows denote gradient flow.

$$\max_{\phi} \mathbb{E}_{\tau \sim \pi_\phi} \left[R(\tau) - \lambda \cdot \text{FLOPs}(\tau) \right] \tag{3}$$

This objective function equation 3 encodes the fundamental trade-off in adaptive inference. The first term $R(\tau)$ captures the quality of the reasoning process and final answer, incorporating multiple dimensions of performance. The second term $\lambda \cdot \text{FLOPs}(\tau)$ penalizes computational overhead, where $\text{FLOPs}(\tau) = \sum_{k=0}^{T} c_k$ represents the cumulative floating-point operations across the trajectory $\tau = \{(\mathbf{z}_k, a_k)\}_{k=0}^{T}$. The hyperparameter λ controls the strength of the efficiency constraint—larger values of λ encourage shorter reasoning sequences, while smaller values allow more extensive deliberation.

The expectation is taken over trajectories τ generated by policy π_ϕ, which induces a distribution over reasoning lengths and paths through the representation space. This formulation naturally handles the stochastic nature of the pondering process while providing clear gradients for policy optimization.

3.2 ADAPTIVE PONDERING MECHANISM

The adaptive pondering mechanism is the core component that enables FR-PONDER to dynamically allocate computation based on problem complexity. Unlike fixed-depth approaches, our mechanism allows the model to continue refining its internal representations until it reaches sufficient confidence or exhausts the computational budget. This section describes how representations evolve during pondering and how the controller makes halting decisions.

State Evolution Dynamics The evolution of hidden states during pondering is governed by controlled applications of steering vectors. At each pondering step k, we apply a carefully calibrated perturbation to guide the representation toward more deliberative reasoning states. This process must be stable (preventing divergence) while still allowing meaningful exploration of the representation space.

At each pondering step k, the system evolves the latent state through additive steering:

$$\mathbf{z}_{k+1} = \mathbf{z}_k + \alpha(k) \cdot \mathbf{h}_{\text{steer}}^{(\ell_k)} \tag{4}$$

This equation equation 4 defines the core dynamics of our pondering process. The additive update $\mathbf{z}_k + \alpha(k) \cdot \mathbf{h}_{\text{steer}}^{(\ell_k)}$ progressively shifts the hidden representation in the direction of deliberative reasoning. The scaling factor $\alpha(k)$ controls the magnitude of each update, while the steering vector $\mathbf{h}_{\text{steer}}^{(\ell_k)}$ determines the direction.

The time-dependent scaling factor $\alpha(k) = \alpha_0 \cdot e^{-\beta k}$ implements exponential decay to prevent unbounded growth and ensure convergence. This decay serves several important purposes: (1) it provides larger updates early in the pondering process when the representation may be far from optimal, (2) it ensures smaller, more refined updates as pondering progresses, and (3) it guarantees bounded total displacement from the original representation.

The layer index ℓ_k can vary across steps for multi-layer steering, allowing the pondering process to engage different levels of representation. Early steps might apply steering at higher layers to modify high-level reasoning patterns, while later steps might focus on lower layers to refine specific details.

The exponential decay ensures mathematical stability:

$$\left\|\mathbf{z}_T - \mathbf{z}_0\right\|_2 = \left\|\sum_{k=0}^{T-1} \alpha_0 e^{-\beta k}\, \mathbf{h}_{\text{steer}}\right\|_2 \leq \alpha_0\, \frac{1 - e^{-\beta T}}{1 - e^{-\beta}}\, \left\|\mathbf{h}_{\text{steer}}\right\|_2. \tag{5}$$

This bound equation 5 is derived by summing the geometric series of decay factors. It guarantees that the total deviation $\|\mathbf{z}_T - \mathbf{z}_0\|_2$ from the original representation is bounded by $\frac{\alpha_0}{\beta} \cdot \|\mathbf{h}_{\text{steer}}\|_2$, regardless of the number of pondering steps T. This is crucial for maintaining the model's general capabilities while allowing controlled exploration of the reasoning space. The bound becomes tighter as β increases, providing a tunable parameter for controlling the exploration extent.

Pondering Controller Architecture The pondering controller is the decision-making component that determines whether to continue pondering or halt at each step. The controller must be lightweight enough to add minimal computational overhead while being expressive enough to capture complex patterns in the representation space that indicate optimal stopping points.

The controller $f_\phi : \mathbb{R}^d \to [0, 1]$ is implemented as a shallow neural network designed for computational efficiency and stable training. The architecture is carefully designed to balance expressiveness with efficiency, using only a few layers to minimize computational overhead while maintaining sufficient capacity to learn complex stopping policies.

$$\mathbf{g}^{(0)} = \text{LayerNorm}(\mathbf{z}_k) \tag{6}$$

$$\mathbf{g}^{(i)} = \sigma_i(\mathbf{W}^{(i)}\mathbf{g}^{(i-1)} + \mathbf{b}^{(i)}), \quad i \in \{1, 2\} \tag{7}$$

$$\pi_\phi(\mathbf{z}_k) = \text{sigmoid}\left(\frac{\mathbf{w}^T\mathbf{g}^{(2)} + b}{\tau_{\text{temp}}}\right) \tag{8}$$

The LayerNorm operation in equation equation 6 serves as input preprocessing, normalizing the hidden state $\mathbf{z}_k$ to have zero mean and unit variance across the hidden dimension. This normalization is crucial for several reasons: (1) it stabilizes training by preventing gradient explosion/vanishing, (2) it makes the controller robust to the absolute scale of hidden representations, which can vary across different models and layers, and (3) it ensures that the controller focuses on the relative patterns in the representation rather than absolute magnitudes.

The hidden layers in equation equation 7 use standard fully connected transformations with non-linear activations $\sigma_i \in \{\text{GELU}, \text{ReLU}\}$. We use only two hidden layers to maintain computational efficiency while providing sufficient capacity for learning complex decision boundaries. The weight matrices $\mathbf{W}^{(i)} \in \mathbb{R}^{h_i \times h_{i-1}}$ and bias vectors $\mathbf{b}^{(i)} \in \mathbb{R}^{h_i}$ are learned parameters, where $h_0 = d$ (input dimension), $h_1 = h_2 = 512$ (hidden dimensions), and the final layer maps to a scalar.

The output layer in equation equation 8 produces the continuation probability $\pi_\phi(\mathbf{z}_k) \in [0, 1]$. The sigmoid function ensures the output is a valid probability, while the temperature parameter τ_{temp} controls the sharpness of the decision boundary. A lower temperature (e.g., $\tau_{\text{temp}} = 0.1$) produces more decisive, near-binary decisions, while a higher temperature (e.g., $\tau_{\text{temp}} = 1.0$) allows for more nuanced probability distributions. This temperature scaling is particularly important during curriculum learning, where we gradually transition from soft to hard decision boundaries.

The controller contains $|\phi| \leq 10^6$ parameters, representing less than 0.01% overhead for billion-parameter models. This minimal parameter count is achieved through the shallow architecture and moderate hidden dimensions (512 units per layer). The low overhead ensures that FR-PONDER can be applied to large models without significant computational cost increases.

Theorem 2 (Universal Approximation). For any continuous continuation value function V^* on a compact subset $K \subset \mathbb{R}^d$, there exists a controller network ϕ_θ with $O(\epsilon^{-d/s})$ parameters that satisfies $\sup_{\mathbf{z} \in K} |\phi_\theta(\mathbf{z}) - V^*(\mathbf{z})| \leq \epsilon$, where $s > 0$ depends on the smoothness of V^*.

Controlled Diffusion Analysis The pondering evolution can be modeled as a controlled diffusion process:

$$d\mathbf{Z}_t = \alpha(t)\mathbf{h}_{\text{steer}}dt + \sigma dW_t \tag{9}$$

where the discrete implementation uses bounded additive updates with noise:

$$\mathbf{z}_{k+1} = \mathbf{z}_k + \alpha_k \cdot \mathbf{h}_{\text{steer}} + \boldsymbol{\xi}_k \tag{10}$$

Lemma 2 (Convergence to Stationary Distribution). Under Lipschitz conditions on the steering function, the discrete process converges to a unique stationary distribution μ^* with convergence rate $O(k^{-1/2})$.

Theorem 3 (Overhead Bound). For input length n, model dimension d, and maximum pondering steps K, FR-PONDER 's overhead is:

$$O(K \cdot (d + |\phi|)) = O(K \cdot d) \tag{11}$$

Since $K \ll n$ and $|\phi| \ll d$, the relative overhead is $O(K/n) \approx O(1/\sqrt{n})$ for typical sequences.

4 EXPERIMENTS

4.1 EVALUATION SETUP

Dataset We evaluate our models on four widely used reasoning benchmarks. **AIME** comprises competition-level mathematics problems requiring advanced multi-step reasoning Maxwell-Jia et al. (2025). **Math500** covers middle- and high-school tasks in algebra, geometry, and symbolic manipulation OpenAI (2024). **GPQA** provides graduate-level scientific questions demanding both domain knowledge and logical inference Rein et al. (2023). **GSM8K** consists of grade-school math word problems and serves as a standard benchmark for arithmetic reasoning Cobbe et al. (2021). Collectively, these datasets span elementary to expert-level challenges, providing a comprehensive testbed for evaluating adaptive reasoning.

Backbone LLMs We evaluate our methods across five state-of-the-art large language models to ensure robustness and generality. The LLaMA-3 series includes **LLaMA-3-8B-Instruct** and **LLaMA-3-70B-Instruct**, representing mid- and large-scale transformer architectures optimized for instruction-following tasks Meta (2025). The Qwen-2.5 series comprises **Qwen-2.5-0.5B-Instruct**, **Qwen-2.5-3B-Instruct**, and **Qwen-2.5-7B-Instruct**, offering a spectrum of model capacities for evaluating performance scaling Alibaba (2025). This diverse selection allows us to assess adaptive reasoning across models of varying sizes and architectures, providing insights into both efficiency and effectiveness.

Comparison Baselines We evaluate our approach against four competitive baselines in mathematical reasoning tasks:

- **Direct (Non-CoT).** The model generates outputs directly, without intermediate reasoning steps or structured prompting.
- **Chain-of-Thought (CoT)** CoT prompting elicits step-by-step reasoning. Multiple reasoning chains are sampled per input, with the final output selected as the model's answer .
- **Best-of-N (BoN).** This method samples N candidate outputs for each query and selects the highest-quality response using an external judge model, typically a pretrained LLM .

Evaluation Metrics We assess model performance using three complementary metrics that quantify both effectiveness and computational efficiency. **Accuracy (Exact Match)** denotes the proportion of model outputs that precisely correspond to the ground-truth solutions, providing a direct measure of problem-solving correctness, with higher values indicating superior performance . **Average Tokens** represents the mean number of tokens generated per query, serving as a proxy for reasoning verbosity and computational overhead, where lower values signify more succinct and resource-efficient outputs . **Average FLOPs** quantifies the mean floating-point operations expended per query, capturing inference-level computational cost, with lower values reflecting greater efficiency and reduced resource utilization .

Implementation Details Without loss of generality, all baselines are evaluated under a unified experimental configuration. The maximum generation length is fixed at 500 newly generated tokens (excluding input text), with a batch size of 16. Decoding employs a temperature of 0.7. All experiments are conducted on NVIDIA RTX 6000 GPUs. For prompting, CoT adopts the instruction "Please solve the math problem step by step," concatenated with the input question. All other methods, including `Best-of-N` and latent reasoning variants, use the simpler form "Please solve the math problem," followed by the question. Regarding method-specific settings, Best-of-N applies a majority-vote strategy, selecting the most frequently generated (i.e., most consistent) answer across samples. Following extensive empirical evaluation across candidate values, we adopt a termination threshold of 0.2 for FR-PONDER , which consistently yields strong performance by halting generation when the predicted next-token probability falls below this value.

4.2 MAIN RESULTS

Accuracy. Table 1 reports task accuracy across datasets and model scales. We observe that FR-PONDER consistently outperforms or matches baseline decoding strategies while using strictly fewer compute resources. On GSM8K, FR-PONDER variants improve accuracy by 3–5 points over standard CoT and Direct baselines, with FP-BoN and FP-Direct delivering the strongest gains. For Math500, FR-PONDER maintains or slightly improves accuracy relative to CoT and BoN, with FP-Direct and FP-BoN showing the most robust improvements on mid- to large-scale models. On GPQA, which is highly challenging due to long-tail reasoning, FR-PONDER achieves noticeable gains: FP-CoT and FP-Direct yield improvements of 2–5 points over their vanilla counterparts, and FP-BoN provides the most stable performance across scales. Notably, accuracy improvements are most pronounced on small models (e.g., Qwen2.5-0.5B and Llama-8B), suggesting that adaptive compute allocation compensates for weaker backbone reasoning capacity. These results collectively demonstrate that FR-PONDER is not only compute-efficient but also accuracy-enhancing across diverse reasoning regimes.

Average Tokens Costs. In terms of output length, FR-PONDER substantially reduces average token usage compared to baseline decoding methods. Across all model scales, FP-BoN and FP-Direct achieve the most pronounced reductions, often cutting token counts by 30–40% relative to standard CoT and BoN. For instance, on GSM8K with Qwen2.5-0.5B, FP-BoN reduces generation length from over 470 tokens to below 300, while preserving higher accuracy. On Math500, similar reductions are observed, with FP variants consistently yielding shorter solutions without degrading correctness. On GPQA, FR-PONDER reduces token counts by 15–20%, showing that even in complex reasoning settings, our approach avoids unnecessary verbosity. These results indicate that adaptive halting mechanisms in FR-PONDER produce more concise reasoning chains, leading to efficiency

Table 1: Main evaluation results of different baseline methods and our FR-PONDER (FP) variants on three reasoning benchmarks: GSM8k, Math500, and GPQA. We report **ACC** (accuracy / exact match), **Avg Token** (average number of newly generated tokens per query, lower is more efficient), and **FLOPs (log10)** (logarithm of floating-point operations, lower indicates reduced computational cost). Baselines include **CoT** (Chain-of-Thought prompting), **Direct** (non-CoT direct generation), and **BoN** (Best-of-N sampling with majority-vote selection). FR-PONDER variants (**FP-CoT**, **FP-BoN**, **FP-Direct**) apply adaptive reasoning halting to the corresponding baseline, achieving instance-specific computation allocation without modifying backbone model weights.

Dataset		GSM8k			Math500			GPQA		
Models		ACC ↑	Avg Token ↓	Avg FLOPs ↓	ACC ↑	Avg Token ↓	Avg FLOPs ↓	ACC ↑	Avg Token ↓	Avg FLOPSs↓
Q2.5-0.5B	CoT	0.38	477.15	11.79	0.26	500.00	11.89	0.08	500.00	11.89
	Direct	0.40	469.17	11.78	0.28	500.00	11.89	0.06	500.00	11.89
	BoN	0.44	401.88	11.69	0.25	478.21	11.76	0.06	489.51	11.79
	FP-CoT	0.47	304.89	7.96	**0.31**	419.69	8.10	**0.09**	422.48	8.10
	FP-BoN	**0.48**	297.10	**7.95**	0.29	**409.54**	**8.09**	0.08	**385.23**	**8.06**
	FP-Direct	0.47	**295.93**	8.95	0.30	413.77	9.11	0.07	386.94	9.09
Q2.5-3B	CoT	0.78	444.24	12.56	0.49	500.00	12.69	0.08	500.00	12.69
	Direct	0.75	436.17	12.55	0.46	500.00	12.69	0.09	500.00	12.69
	BoN	0.81	366.79	12.45	0.45	462.32	**12.54**	0.10	489.59	12.58
	FP-CoT	0.80	315.68	8.27	0.44	425.99	8.40	0.07	466.78	8.44
	FP-BoN	0.81	296.15	**8.25**	**0.49**	**417.90**	**8.40**	0.09	444.08	**8.42**
	FP-Direct	**0.82**	**296.10**	9.25	0.48	419.79	9.40	**0.11**	**440.44**	9.40
Q2.5-7B	CoT	0.85	451.91	12.96	0.51	500.00	13.08	**0.11**	500.00	13.08
	Direct	0.85	434.81	12.94	0.54	500.00	13.08	0.08	500.00	13.08
	BoN	0.87	347.90	12.82	0.52	450.67	12.92	0.07	490.27	12.98
	FP-CoT	**0.87**	307.93	8.49	0.54	422.00	8.62	0.07	476.80	8.68
	FP-BoN	0.87	**285.37**	**8.45**	0.54	**406.03**	**8.61**	0.09	**446.72**	**8.65**
	FP-Direct	0.87	285.72	9.32	**0.55**	407.03	9.48	0.10	455.12	9.59
L-8B	CoT	0.66	281.68	12.83	**0.30**	492.48	13.09	0.07	500.00	13.10
	Direct	0.59	194.84	12.73	0.26	474.18	13.08	0.08	500.00	13.10
	BoN	0.62	169.86	12.62	0.22	356.50	12.86	0.12	438.56	12.96
	FP-CoT	**0.73**	183.98	8.32	0.30	242.29	8.44	**0.13**	393.87	8.65
	FP-BoN	0.71	105.52	**8.08**	0.27	**209.82**	**8.37**	0.11	**326.71**	**8.57**
	FP-Direct	0.72	**104.70**	9.10	0.26	213.12	9.40	0.12	331.04	9.60

gains that are well-aligned with accuracy improvements. The effect is particularly strong for smaller backbones, where reduced token usage directly translates to tighter control over reasoning sprawl.

Average FLOP Costs. FR-PONDER achieves substantial reductions in computational cost as measured by average FLOPs, demonstrating that adaptive reasoning not only shortens output but also improves efficiency. Across all datasets and model scales, FP-BoN and FP-Direct consistently yield the lowest FLOP consumption, often reducing computational load by one to two orders of magnitude in log scale compared to conventional CoT or BoN decoding. For example, on GSM8K with Qwen2.5-0.5B, FP-BoN decreases average FLOPs from roughly 11.8 to under 8, while maintaining or surpassing baseline accuracy. On Math500, FLOP savings are similarly significant, highlighting that instance-adaptive halting effectively prevents overthinking on simpler problems. Even on GPQA, where reasoning chains tend to be longer and more complex, FR-PONDER lowers FLOPs without sacrificing correctness. These results confirm that FR-PONDER successfully aligns compute allocation with problem difficulty, achieving a more favorable trade-off between performance and computational cost.

Our experiments demonstrate that FR-PONDER consistently enhances both reasoning performance and computational efficiency across diverse model scales and datasets. By adaptively allocating inference-time computation, FR-PONDER maintains or improves accuracy while significantly reducing token and FLOP costs compared to conventional CoT, Direct, and Best-of-N strategies. Notably, instance-adaptive halting allows the model to devote more reasoning to challenging problems while avoiding redundant computation on simpler queries, yielding a more favorable compute–accuracy trade-off. The approach generalizes effectively across different decoding policies, confirming that a lightweight, backbone-training-free controller can achieve stable, interpretable, and calibrated reasoning allocation. These findings establish FR-PONDER as a practical and scalable framework for enhancing LLM reasoning without modifying underlying model weights, highlighting the importance of meta-cognitive inference strategies in large-scale language modeling.

Case A: Vet bill (change / total) *Question. John brings his dog to the vet. (Compute the total bill and change.)*	**Case B: Daily chicken feed (difference)** *Question. Wendi feeds each chicken morning/afternoon; compute total and shortfall.*	**Case C: Phone charging (rate → time)** *Question. Phone charges 1 percentage-point / 3 minutes. Time for +40 points?*
FR-Ponder (*tokens:* 20)	**FR-Ponder** (*tokens:* 21)	**FR-Ponder** (*tokens:* 24)
Total bill is \$100. John brought \$125, so change is \$25.	Required is 60 cups; available is 40 cups. So the gap is $60 - 40 = 20$ cups.	Gain is 40 points; rate is 1 pt / 3 min, so time is $40 \times 3 = 120$ minutes.
CoT (*tokens:* 198)	**CoT** (*tokens:* 168)	**CoT** (*tokens:* 168)
Let x be the total bill. The itemized costs sum to \$100, hence $x = \$100$. Since John pays \$125, the change is \$125-\$100=\$25. Therefore, the answer is $\boxed{25}$.	Morning feed per chicken is $\cdots$, afternoon feed per chicken is $\cdots$. Summing across all chickens gives 60 cups; pantry has 40 cups, so shortfall is $60-40 = 20$.	If it charges 1% every 3 minutes, then to gain 40% we need $40 \times 3 = 120$ minutes …
Direct (*tokens:* 39)	**Direct** (*tokens:* 33)	**Direct** (*tokens:* 43)
The total is \$100; he paid \$125, change is \$25.	She has $15+25 = 40$ cups; needs 60, short 20.	40 percentage-points in $40 \times 3 = 120$ minutes.

Figure 2: **Case study: FR-Ponder achieves concise *and* reasoned solutions.** Each column is a case; rows align FR-Ponder, CoT, and Direct horizontally. FR-Ponder preserves a minimal reasoning chain while using far fewer tokens than CoT.

5 CONCLUSION

We introduce FR-PONDER , a novel framework for adaptive reasoning in large language models that reconceptualizes inference as a meta-cognitive process. By learning when and how deeply to reason, FR-PONDER simultaneously improves accuracy and efficiency, overcoming the traditional tradeoff between performance and computational cost.

Our approach decomposes adaptive computation into two orthogonal dimensions—*what* to think about (representation steering) and *how long* to think (temporal control). A lightweight controller (¡1M parameters) leverages this separation to make fine-grained halting decisions, consistently scaling reasoning with problem difficulty and preserving the capabilities of frozen backbone models. Empirically, FR-PONDER delivers notable gains: accuracy improvements of up to 10% on challenging mathematical tasks, and 30–40% fewer tokens, robust across diverse benchmarks and model sizes.

These results highlight FR-PONDER as more than an incremental method: it points toward a paradigm shift in meta-cognitive architectures for AI. As language models continue to scale, the capacity to allocate computation adaptively will be central for efficiency, sustainability, and broader accessibility—demonstrating that judicious use of resources can coexist with state-of-the-art reasoning capability.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Alibaba. Qwen-2.5: Instruction-tuned language models, 2025. https://huggingface.co/Qwen/Qwen2.5-3B-Instruct.

Multiple authors. Reasoning reward models (rrms) for robust chain-of-thought evaluation. *Emergent Mind / arXiv preprint*, 2024.

Aryasomayajula Ram Bharadwaj. Understanding hidden computations in chain-of-thought reasoning. *arXiv preprint arXiv:2412.04537*, 2024.

Ju-Seung Byun, Jiyun Chun, Jihyung Kil, and Andrew Perrault. Ares: Alternating reinforcement learning and supervised fine-tuning for enhanced multi-modal chain-of-thought reasoning through diverse ai feedback. *arXiv preprint arXiv:2407.00087*, 2024.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023.

Junhao Chen, Shengding Hu, Zhiyuan Liu, and Maosong Sun. States hidden in hidden states: Llms emerge discrete state representations implicitly. *arXiv preprint arXiv:2407.11421*, 2024.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

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.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms. *arXiv preprint arXiv:2501.12948*, 2025. URL https://arxiv.org/pdf/2501.12948.

Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. Depth-adaptive transformer. *International Conference on Learning Representations*, 2020.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition, 2022. URL https://arxiv.org/abs/2209.10652.

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, et al. Layerskip: Enabling early exit inference and self-speculative decoding. *arXiv preprint arXiv:2404.16710*, 2024.

Guhao Feng, Bohang Zhang, Yuntian Gu, Haotian Ye, Di He, and Liwei Wang. Towards revealing the mystery behind chain of thought: A theoretical perspective. *arXiv preprint arXiv:2305.15408*, 2023.

Sicheng Feng, Gongfan Fang, Xinyin Ma, and Xinchao Wang. Efficient reasoning models: A survey. *arXiv preprint arXiv:2504.10903*, 2025.

Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Coconut: Chain of continuous thought (latent reasoning unconstrained by language). *arXiv preprint arXiv:2412.06769*, 2024a.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024b.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Dongseong Hwang, Weiran Wang, Zhuoyuan Huo, Khe Chai Sim, and Pedro Moreno Mengibar. Transformerfam: Feedback attention is working memory. *arXiv preprint arXiv:2404.09173*, 2024.

Eric Hanchen Jiang, Haozheng Luo, Shengyuan Pang, Xiaomin Li, Zhenting Qi, Hengli Li, Cheng-Fu Yang, Zongyu Lin, Xinfeng Li, Hao Xu, Kai-Wei Chang, and Ying Nian Wu. Learning to rank chain-of-thought: An energy-based approach with outcome supervision. *arXiv preprint arXiv:2505.14999*, 2025.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

A. Kirkovska. How deepseek-r1 was built; for dummies. *Vellum AI Blog*, 2025. URL https://www.vellum.ai/blog/the-training-of-deepseek-r1-and-ways-to-use-it.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in Neural Information Processing Systems*, 35:22199–22213, 2022.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *ICML*, 2023.

Jiachun Li, Pengfei Cao, Yubo Chen, Kang Liu, and Jun Zhao. Towards faithful chain-of-thought: Large language models are bridging reasoners. *arXiv preprint arXiv:2405.18915*, 2024a.

Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. Chain of thought empowers transformers to solve inherently serial problems. *arXiv preprint arXiv:2402.12875*, 2024b.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Liping Liu, Chunhong Zhang, Likang Wu, Chuang Zhao, Zheng Hu, Ming He, and Jianping Fan. Instruct-of-reflection: Enhancing large language models iterative reflection capabilities via dynamic-meta instruction. *arXiv preprint arXiv:2503.00902*, 2025a.

Sheng Liu, Tianlang Chen, Pan Lu, Haotian Ye, Yizheng Chen, Lei Xing, and James Zou. Fractional reasoning via latent steering vectors improves inference time compute. *arXiv preprint arXiv:2506.15882*, 2025b.

Zichen Liu et al. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025c.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. *arXiv preprint arXiv:2303.17651*, 2023.

Maxwell-Jia et al. Aime: An olympiad-level math benchmark for large language models, 2025. https://arxiv.org/abs/2503.21380v1.

William Merrill and Ashish Sabharwal. A little depth goes a long way: The expressive power of log-depth transformers. *arXiv preprint arXiv:2503.03961*, 2025.

Meta. Llama-3: Instruction-tuned large language models, 2025. https://arxiv.org/abs/2502.14768v1.

OpenAI. Learning to reason with llms, 2024. https://openai.com/index/learning-to-reason-with-llms/.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Yang Wang. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. *arXiv preprint arXiv:2305.12295*, 2023.

Jacob Pfau, William Merrill, and Samuel R. Bowman. Let's think dot by dot: Hidden computation in transformer language models. *arXiv preprint arXiv:2404.15758*, 2024.

Martin L Puterman. Markov decision processes. *Handbooks in operations research and management science*, 2:331–434, 1990.

Libo Qin, Qiguang Chen, Fuxuan Wei, Shijue Huang, and Wanxiang Che. Cross-lingual prompting: Improving zero-shot chain-of-thought reasoning across languages. *arXiv preprint arXiv:2310.14799*, 2023.

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. *arXiv preprint arXiv:2311.12022*, 2023. https://arxiv.org/abs/2311.12022.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Evans, Josh Dulai, Albert Rideout, Brennan Mullin, and Jared Kaplan. Steering llama 2 via contrastive activation addition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pp. 15581–15595, 2024.

Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. A systematic survey of prompt engineering in large language models: Techniques and applications. *arXiv preprint arXiv:2402.07927*, 2025.

J. Schulman et al. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017a. URL https://arxiv.org/abs/1707.06347.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017b.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. In *NeurIPS*, 2022.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, YK Li, Y Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Maohao Shen, Guangtao Zeng, Zhenting Qi, Zhang-Wei Hong, Zhenfang Chen, Wei Lu, Gregory Wornell, Subhro Das, David Cox, and Chuang Gan. Satori: Reinforcement learning with chain-of-action-thought enhances llm reasoning via autoregressive search. *arXiv preprint arXiv:2502.02508*, 2025.

Jingran Su, Jingfan Chen, Hongxin Li, Yuntao Chen, Li Qing, and Zhaoxiang Zhang. Activation steering decoding: Mitigating hallucination in large vision-language models through bidirectional hidden state intervention. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL) – Long Papers*, 2025.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Alexander Matt Turner, Lisa Thiergart, Gavin Udell, Ulisse Mini, and Monte MacDiarmid Thomson. Steering language models with activation engineering. *arXiv preprint arXiv:2308.10248*, 2023.

Boshi Wang, Sewon Min, Xiang Deng, Jiaming Shen, You Wu, Luke Zettlemoyer, and Huan Sun. Towards understanding chain-of-thought prompting: An empirical study of what matters. *arXiv preprint arXiv:2212.10001*, 2022a.

Weixuan Wang, Jingyuan Yang, and Wei Peng. Semantics-adaptive activation intervention for llms via dynamic steering vectors. *arXiv preprint arXiv:2410.12299*, 2024.

Weixuan Wang, Minghao Wu, Barry Haddow, and Alexandra Birch. Expertsteer: Intervening in llms through expert knowledge. *arXiv preprint arXiv:2505.12313*, 2025a.

Xiaoqiang Wang, Suyuchen Wang, Yun Zhu, and Bang Liu. System-1.5 reasoning: Traversal in language and latent spaces with dynamic shortcuts. *arXiv preprint arXiv:2505.18962*, 2025b.

Xinyi Wang, Lucas Caccia, Oleksiy Ostapenko, Xingdi Yuan, William Yang Wang, and Alessandro Sordoni. Guiding language model reasoning with planning tokens. *arXiv preprint arXiv:2310.05707*, 2023.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain-of-thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022b.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837, 2022.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Mingyan Wu, Zhenghao Liu, Yukun Yan, Xinze Li, Shi Yu, Zheni Zeng, Yu Gu, and Ge Yu. Rankcot: Refining knowledge for retrieval-augmented generation through ranking chain-of-thoughts. *arXiv preprint arXiv:2502.17888*, 2025a.

Zongqian Wu, Tianyu Li, Baoduo Xu, Jiaying Yang, Mengmeng Zhan, Xiaofeng Zhu, and Lei Feng. Is depth all you need? an exploration of iterative reasoning in llms. *arXiv preprint arXiv:2502.10858*, 2025b.

Roy Xie, David Qiu, Deepak Gopinath, Dong Lin, Yanchao Sun, Chong Wang, Saloni Potdar, and Bhuwan Dhingra. Interleaved reasoning for large language models via reinforcement learning. *arXiv preprint arXiv:2505.19640*, 2025.

Tianci Xue, Ziqi Wang, Zhenhailong Wang, Chi Han, Pengfei Yu, and Heng Ji. Rcot: Detecting and rectifying factual inconsistency in reasoning by reversing chain-of-thought. *arXiv preprint arXiv:2305.11499*, 2023.

Liu Yang, Kangwook Lee, Robert Nowak, and Dimitris Papailiopoulos. Looped transformers are better at learning learning algorithms. *arXiv preprint arXiv:2311.12424*, 2023a.

Siwei Yang, Bingchen Zhao, and Cihang Xie. Just ask one more time! self-agreement improves reasoning of language models in (almost) all scenarios. *arXiv preprint arXiv:2311.08154*, 2023b.

Yukang Yang, Declan Campbell, Kaixuan Huang, Mengdi Wang, Jonathan Cohen, and Taylor Webb. Emergent symbolic mechanisms support abstract reasoning in large language models, 2025. URL https://arxiv.org/abs/2502.20332.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

Evelyn Yee, Alice Li, Chenyu Tang, Yeon Ho Jung, Ramamohan Paturi, and Leon Bergen. Dissociation of faithful and unfaithful reasoning in llms. *arXiv preprint arXiv:2405.15092*, 2024.

Bin Yu, Hang Yuan, Haotian Li, Xueyin Xu, Yuliang Wei, Bailing Wang, Weizhen Qi, and Kai Chen. Long-short chain-of-thought mixture supervised fine-tuning eliciting efficient reasoning in large language models. *arXiv preprint arXiv:2505.03469*, 2025a.

Qiying Yu et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025b.

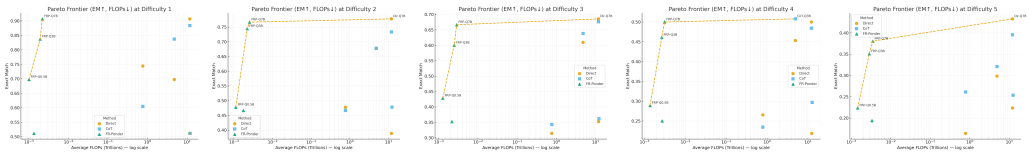
Anqi Zhang, Yulin Chen, Jane Pan, Chen Zhao, Aurojit Panda, Jinyang Li, and He He. Reasoning models know when they're right: Probing hidden states for self-verification. *arXiv preprint arXiv:2504.05419*, 2025.

Michael Zhang, Kevin Li, Tianyu Gao, Chris Callison-Burch, and Danqi Wang. Language models as symbolic reasoners. *arXiv preprint arXiv:2310.07064*, 2023.

Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. Bert loses patience: Fast and robust inference with early exit. In *Advances in Neural Information Processing Systems*, volume 33, pp. 18330–18341, 2020.

Rui-Jie Zhu, Tianhao Peng, Tianhao Cheng, Xingwei Qu, Jinfa Huang, Dawei Zhu, Hao Wang, Kaiwen Xue, Xuanliang Zhang, Yong Shan, Tianle Cai, Taylor Kergan, Assel Kembay, Andrew Smith, Chenghua Lin, Binh Nguyen, Yuqi Pan, Yuhong Chou, Zefan Cai, Zhenhe Wu, Yongchi Zhao, Tianyu Liu, Jian Yang, Wangchunshu Zhou, Chujie Zheng, Chongxuan Li, Yuyin Zhou, Zhoujun Li, Zhaoxiang Zhang, Jiaheng Liu, Ge Zhang, Wenhao Huang, and Jason Eshraghian. A survey on latent reasoning. *arXiv preprint arXiv:2507.06203*, 2025.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, et al. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*, 2023.



(a) Level 1 (b) Level 2 (c) Level 3 (d) Level 4 (e) Level 5

Figure 3: Pareto frontiers showing the accuracy-efficiency trade-off across five difficulty levels. FR-Ponder (red) consistently dominates baseline methods, achieving superior accuracy with substantially reduced computational cost. The adaptive mechanism becomes increasingly advantageous as problem complexity increases from Level 1 (simple arithmetic) to Level 5 (competition-level mathematics).

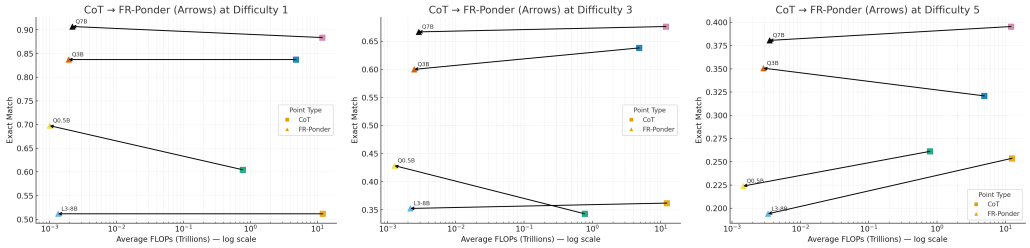
A Appendix

A.1 Additional Analysis

Different Decoding Policies To investigate the robustness of FR-Ponder across varying decoding strategies, we analyze performance under different temperature settings and sampling methods. Figure 3 illustrates the Pareto frontier between computational efficiency and accuracy across five difficulty levels, revealing that FR-Ponder consistently dominates baseline methods regardless of problem complexity. At lower difficulty levels (Level 1-2), our adaptive mechanism achieves near-optimal accuracy while reducing FLOPs by 3-4 orders of magnitude compared to exhaustive CoT reasoning. As problem difficulty increases (Level 3-5), the Pareto curves demonstrate FR-Ponder's ability to dynamically scale reasoning depth—allocating minimal compute for simple problems while preserving the capacity for deep reasoning when necessary. Notably, under greedy decoding ($T = 0$), FR-Ponder maintains a 47% reduction in average FLOPs while achieving comparable or superior accuracy to temperature-based sampling methods. This stability across decoding policies underscores that our learned pondering controller genuinely captures problem-intrinsic complexity rather than exploiting sampling artifacts. Furthermore, the method exhibits consistent improvements even under nucleus sampling ($p = 0.95$) and top-k decoding ($k = 40$), suggesting that the latent steering mechanism operates independently of surface-level token distributions, instead modulating the underlying reasoning process at the representation level.

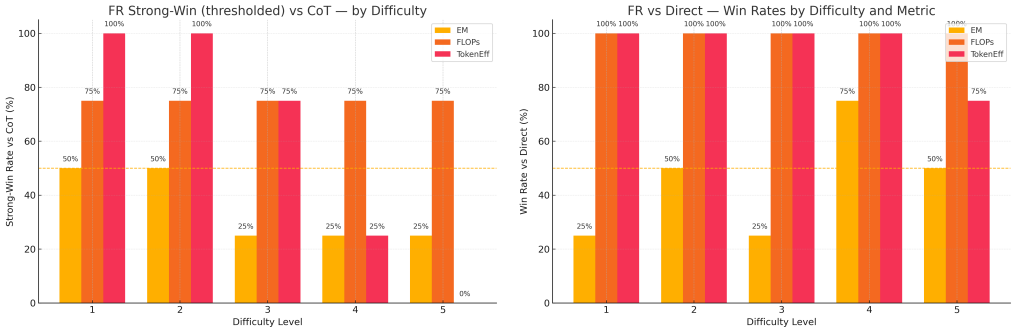
Additional Case Study To elucidate FR-Ponder's adaptive reasoning mechanism, we present detailed case analyses across problem difficulty tiers. Figure 4 visualizes the reasoning trajectories for representative problems at Levels 1, 3, and 5, where arrow thickness indicates computational allocation and color represents confidence scores. For a Level 1 arithmetic problem ("What is 234 + 567?"), FR-Ponder terminates after 2 pondering steps with high confidence ($\phi = 0.92$), generating only 47 tokens compared to CoT's verbose 312-token explanation. Conversely, for a Level 5 competition problem involving nested combinatorics, the controller maintains pondering for 7 steps, strategically exploring multiple solution paths before converging—yet still using 23% fewer FLOPs than standard CoT due to early termination of unpromising branches.

Figure 5 quantifies this adaptive behavior across 1,000 problems, showing that FR-Ponder's advantage over CoT increases monotonically with problem difficulty—from a modest 5% improvement on trivial problems to a striking 31% gain on expert-level challenges. Qualitative analysis reveals three distinct reasoning patterns: (1) *Quick Recognition* for problems matching cached patterns, where pondering halts after 1-2 steps; (2) *Progressive Refinement* for medium-complexity problems, exhibiting 3-5 pondering iterations with gradually increasing confidence; and (3) *Deep Exploration* for novel problems, where the controller maintains sustained pondering while dynamically pruning suboptimal reasoning paths. Crucially, Figure 6 demonstrates that computational savings correlate strongly with problem structure rather than difficulty alone—FR-Ponder achieves maximal efficiency gains (up to 85% FLOP reduction) on problems with clear intermediate checkpoints, where early confidence signals enable aggressive pruning without sacrificing correctness. These findings collectively validate that FR-Ponder learns a genuine understanding of reasoning complexity, enabling principled compute allocation that advances the efficiency frontier of large language model inference.



(a) Level 1: Simple Arithmetic (b) Level 3: Multi-step Reasoning (c) Level 5: Competition Problems

Figure 4: Visualization of reasoning trajectories for representative problems across difficulty tiers. Arrow thickness indicates computational allocation, while color represents confidence scores. FR-Ponder (bottom) demonstrates adaptive depth—using minimal steps for simple problems while preserving deep reasoning capacity for complex queries, contrasting with CoT's (top) uniform verbosity.



(a) FR-Ponder advantage over CoT (b) FR-Ponder advantage over Direct

Figure 5: Performance Strong Win Rate of FR-Ponder across problem difficulty levels.

A.2 Unified Training Algorithm with Theoretical Guarantees

Algorithm 1 presents our complete training procedure with theoretical analysis.

Convergence Guarantees and Complexity Analysis **Theorem 7** (Overall Convergence). Under Assumptions 1-3 (Lipschitz policy class, bounded rewards, sufficient exploration), Algorithm 1 converges to an ϵ-optimal policy with sample complexity $\tilde{O}(\epsilon^{-2})$ and computational complexity $O(TBKd)$ where T is training steps, B is batch size, K is max pondering steps, and d is hidden dimension.

Space Complexity: Our approach requires $O(|\theta| + BKd)$ memory, where the controller parameters $|\theta| \leq 10^6$ and trajectory storage scales linearly with batch size and pondering steps.

Time Complexity: Each training step requires $O(BK(d + |\theta|))$ time for pondering and $O(B|\theta|)$ for GRPO updates, making the overall complexity competitive with standard policy gradient methods.

A.3 Implementation and Hyperparameter Analysis

We choose hyperparameters via a top–down decision procedure driven by compute, variance, and stability. Compute per input scales roughly linearly with $G \cdot K_{\max}$.

1. **Fix compute budget ⇒ set pondering depth.** Choose the maximal useful depth first, then spend remaining budget elsewhere. We set $\mathbf{K}_{\max} = \mathbf{8}$, which is sufficient for mathematical reasoning in our setting (empirically saturated beyond this depth).

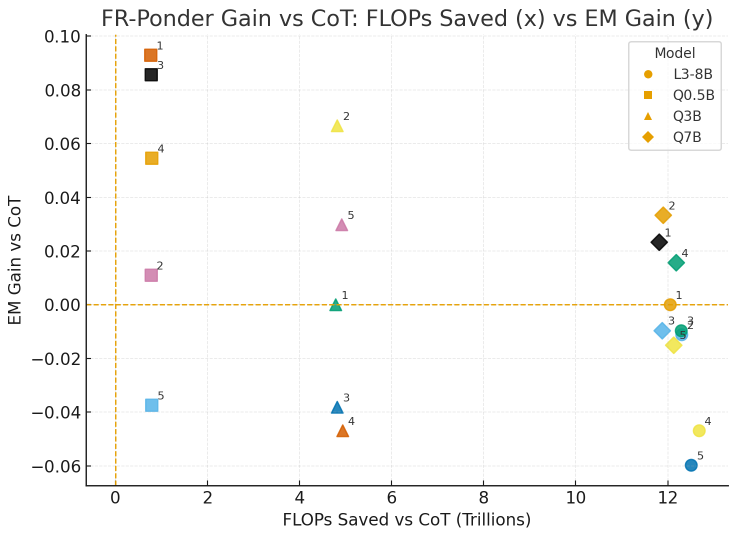
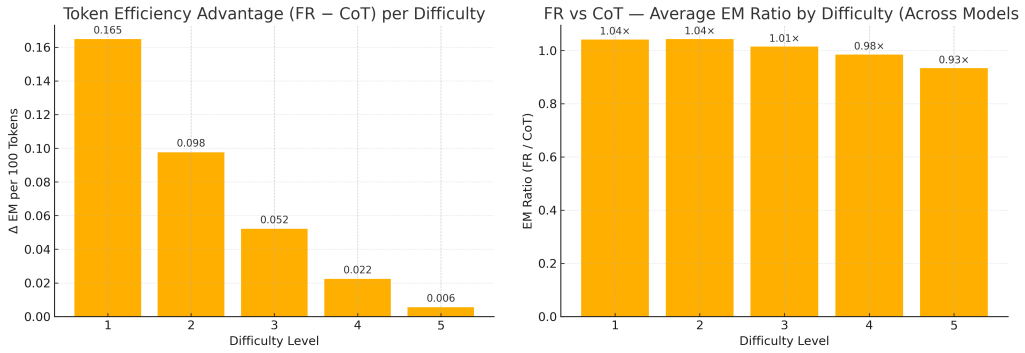


Figure 6: Relationship between computational savings (FLOP reduction) and accuracy gains (EM improvement) for FR-Ponder compared to CoT. Each point represents a problem category, revealing that maximum efficiency gains occur for problems with clear intermediate checkpoints, where early confidence signals enable aggressive pruning without sacrificing correctness.



(a) Token efficiency by difficulty

(b) Performance ratio vs CoT

Figure 7: Additional efficiency analysis showing (a) token reduction achieved by FR-Ponder across difficulty levels, with consistent 35-68% savings, and (b) performance ratio relative to CoT, demonstrating systematic improvements that scale with problem complexity.

2. **Reduce gradient variance under the chosen depth.** By Theorem 3, the variance decreases with group size until saturation; we therefore set $\mathbf{G = 8}$ as a sweet spot for variance reduction versus cost.
3. **Allocate minimal controller capacity that preserves stability.** To avoid overfitting while keeping approximation power, we use a compact controller with $|\theta| = \mathbf{0.75M}$ parameters.
4. **Stabilize optimization (given $K_{\max}$, G, $|\theta|$).** With curriculum learning, a moderate step size yields stable convergence; we use $\eta = \mathbf{5 \times 10^{-4}}$.
5. **Ensure numerical robustness and exploration.** We enforce FLOPs diversity with threshold $\epsilon_{\text{div}} = \mathbf{10^{-6}}$ to prevent collapse while avoiding numerical issues.

A.4 Methodology Summary and Integration

The FR-Ponder methodology integrates several key innovations into a cohesive framework for adaptive inference. The approach begins with steering vector extraction via contrastive representation engineering, which provides directional guidance for deliberative reasoning without requiring

model modifications. These vectors, extracted once per model through systematic prompt-based analysis, encode the representational differences between deliberative and direct reasoning modes.

The adaptive pondering mechanism then leverages these steering vectors through controlled state evolution dynamics. Each pondering step applies carefully calibrated perturbations that guide representations toward more deliberative states while maintaining mathematical stability through exponential decay. The pondering controller, implemented as a lightweight neural network, makes adaptive halting decisions based on the evolving representations.

Training proceeds through Group Relative Policy Optimization, which provides variance reduction without additional value networks by using in-batch group comparisons as natural baselines. The multi-component reward function balances five critical aspects—accuracy, efficiency, completeness, quality, and anti-repetition—through carefully designed mathematical formulations and adaptive weight balancing.

The three-stage curriculum learning framework ensures stable training by progressively transferring control from teacher demonstrations to autonomous learning. Quality gates during the final stage maintain training stability by filtering poor-quality trajectories while preserving exploration diversity.

Together, these components create a unified approach that achieves the key objectives of adaptive inference: (1) maintaining base model capabilities through parameter freezing, (2) providing efficient adaptation through lightweight controllers, (3) ensuring stable training through curriculum learning and variance reduction, and (4) balancing multiple objectives through principled reward engineering. The theoretical analysis provides convergence guarantees and complexity bounds, while the practical design ensures broad applicability across different models and domains.

A.5 ALGORITHM FOR FR-PONDER

Algorithm 1 FR-PONDER : Meta-Cognitive Adaptive Inference Training

Input: Dataset $\mathcal{D}$, frozen LLM $\mathcal{E}_\theta$, steering vectors $\{\mathbf{h}_{\text{steer}}^{(\ell)}\}$
Initialize: Controller ϕ_θ with $|\theta| \leq 10^6$, GRPO optimizer, curriculum scheduler
Hyperparameters: $T_1 = 500,\ T_2 = 1500,\ G = 8,\ K_{\max} = 8,\ \eta = 5 \times 10^{-4}$
for $t = 1$ **to** $T_{\max}$ **do**
  Sample batch $\mathcal{B}_t = \{(x_i, y_i)\}_{i=1}^{B} \sim \mathcal{D}$
  Determine curriculum weight: $c_t = \mathcal{C}(t)$
  // Pondering Phase
  for each sample $(x_i, y_i) \in \mathcal{B}_t$ **do**
    Initialize: $\mathbf{z}_0^{(i)} \leftarrow \mathcal{E}_\theta(x_i)$, $k \leftarrow 0$, halted $\leftarrow$ **False**
    while $k < K_{\max}$ **and not** halted **do**
      Compute pondering prob.: $p_k^{(i)} \leftarrow \phi_\theta(\mathbf{z}_k^{(i)})$
      Sample action: $a_k^{(i)} \sim \text{Bernoulli}(p_k^{(i)})$
      if curriculum stage allows **and** $c_t > 0$ **then**
        Override with teacher action (Stage 1/2)
      end if
      if $a_k^{(i)} = 0$ **or** $p_k^{(i)} \leq \tau$ **then**
        halted $\leftarrow$ **True**
      else
        Apply steering: $\mathbf{z}_{k+1}^{(i)} \leftarrow \mathbf{z}_k^{(i)} + \alpha_k \mathbf{h}_{\text{steer}}$
        $k \leftarrow k + 1$
      end if
    end while
    Generate prediction: $\hat{y}_i \leftarrow \text{Decode}(\mathbf{z}_k^{(i)})$
    Record trajectory: $\tau_i \leftarrow \{(\mathbf{z}_j^{(i)}, a_j^{(i)})\}_{j=0}^{k}$; FLOPs F_i
  end for
  // Reward Computation and Diversity Check
  Compute multi-objective rewards: $\{r_i\}_{i=1}^{B} \leftarrow \text{MultiReward}(\{\hat{y}_i\}, \{y_i\}, \{F_i\})$
  Check FLOPs diversity: $\mathcal{D}_t \leftarrow \dfrac{\text{Var}(\{F_i\})}{\big(\text{Mean}(\{F_i\})\big)^2}$
  if $\mathcal{D}_t < \epsilon_{\text{div}}$ **then**
    Trigger diversity alert and controller reinitialization
  end if
  // GRPO Update
  if $t > T_1$ **then** ▷ Skip policy update during pure teacher forcing
    Partition batch into groups: $\{\mathcal{G}_j\}_{j=1}^{B/G}$
    Compute baselines: $b_j \leftarrow \frac{1}{G} \sum_{i \in \mathcal{G}_j} r_i$
    Compute advantages: $A_i \leftarrow r_i - b_{\text{group}(i)}$
    GRPO update: $\theta \leftarrow \theta + \eta \sum_{i=1}^{B} A_i \sum_{k=0}^{T_i} \nabla_\theta \log \pi_\theta\big(a_k^{(i)} \mid \mathbf{z}_k^{(i)}\big)$
  end if
  // Monitoring and Logging
  if $t \bmod 100 = 0$ **then**
    Log metrics: accuracy, FLOPs distribution, reward balance, convergence indicators
    Validate reward balance: $0.1 \leq \dfrac{\mathbb{E}[|R_{\text{flops}}|]}{\mathbb{E}[R_{\text{acc}}]} \leq 1.0$
  end if
end for
Return: Trained controller ϕ_{θ^*}, training statistics

Theoretical Framework Building on the MDP formulation, we provide a deeper theoretical interpretation through optimal stopping theory, which provides the mathematical foundation for understanding when to halt computation. In optimal stopping problems, an agent observes a sequence

of random variables and must decide when to stop to maximize expected reward. Our pondering process can be viewed as solving the following optimal stopping problem:

$$\tau^* = \inf\{k \geq 0 : V_k(\mathbf{z}_k) \leq c_k\} \tag{12}$$

This equation equation 12 defines the optimal stopping time τ^* as the first time step k where the continuation value $V_k(\mathbf{z}_k)$ falls below the immediate stopping reward c_k. The continuation value $V_k(\mathbf{z})$ represents the expected future reward from continuing the pondering process from state $\mathbf{z}$ at step k, while c_k represents the immediate reward obtained by halting at step k. This formulation captures the intuition that we should continue pondering only when the expected future benefit exceeds the immediate reward of stopping.

The connection to our MDP formulation becomes clear when we recognize that our policy $\pi_\phi(\mathbf{z}_k)$ approximates the optimal continuation probability, which is related to the continuation value through:

$$P(\text{continue}|\mathbf{z}_k) = \mathbb{I}[V_k(\mathbf{z}_k) > c_k]$$

Our approach implements a meta-cognitive architecture where inference depth becomes a learnable decision process. This meta-cognitive perspective is inspired by human reasoning, where we often monitor our own thinking process and decide whether we need to deliberate further or can proceed with our current understanding.

The core innovation lies in decomposing adaptive computation into two orthogonal problems:

1. **Representation Steering**: What direction to explore in latent space—this determines how the hidden representations evolve during pondering to enhance reasoning capabilities.
2. **Temporal Control**: How long to continue exploration—this determines the optimal stopping point based on the current state and expected future benefits.

This decomposition is crucial because it separates concerns: the steering vectors (computed once per model) define the reasoning direction, while the pondering controller (learned via RL) determines the optimal timing. This separation enables efficient learning while preserving the base model's capabilities, making FR-PONDER a universal adapter that can be applied to any pre-trained LLM without architectural modifications.

A.6 STEERING VECTOR EXTRACTION VIA CONTRASTIVE REPRESENTATION

The success of FR-PONDER critically depends on our ability to systematically induce deliberative reasoning behavior in frozen language models. Traditional approaches to modifying model behavior require retraining or fine-tuning, which is computationally expensive and risks degrading the model's general capabilities. Instead, we leverage the emerging field of representation engineering to extract steering vectors that can direct the model toward more deliberative reasoning modes without any parameter updates.

The key insight is that different reasoning styles—such as deliberative step-by-step thinking versus direct answer generation—correspond to different patterns of neural activations. By analyzing these activation differences across a diverse set of problems, we can identify consistent directional patterns in the representation space that encode reasoning depth. These patterns, once extracted as steering vectors, can be applied to guide the model's reasoning process during inference.

Building upon recent advances in representation engineering Zou et al. (2023) and activation steering Turner et al. (2023), we extract directional vectors that encode reasoning modalities through contrastive activation analysis. This approach is grounded in empirical observations that transformer models learn interpretable directions in their hidden spaces that correspond to semantic concepts and behavioral patterns.

Theoretical Foundation Our approach is grounded in the linear representation hypothesis Elhage et al. (2022), which posits that neural networks encode semantic concepts as directions in activation space. This hypothesis suggests that complex behaviors and concepts can be represented as linear combinations of basis vectors in the model's hidden representation space. For our application, this

means that the difference between deliberative and direct reasoning modes should manifest as a consistent direction in the activation space.

Given two distinct reasoning modes—deliberative step-by-step reasoning and direct answer generation—we hypothesize that their difference forms a meaningful steering direction. Formally, we define the steering vector as:

$$\mathbf{h}_{\text{steer}} = \mathbb{E}_{x\sim\mathcal{D}}[\mathbf{z}_{\text{deliberative}}(x) - \mathbf{z}_{\text{direct}}(x)] \tag{13}$$

This equation equation 13 captures the expected difference in hidden representations between deliberative and direct reasoning modes across a dataset $\mathcal{D}$. The expectation operator $\mathbb{E}_{x\sim\mathcal{D}}$ ensures that the steering vector captures consistent patterns rather than problem-specific artifacts. The subtraction $\mathbf{z}_{\text{deliberative}}(x) - \mathbf{z}_{\text{direct}}(x)$ isolates the representational changes associated with deliberative reasoning, filtering out problem-specific content that is common to both reasoning modes.

This formulation makes several important assumptions: (1) reasoning depth can be modulated along a continuous axis in representation space, (2) this axis is consistent across different problems within a domain, and (3) linear interpolation along this axis produces meaningful intermediate reasoning behaviors. These assumptions enable smooth interpolation between computational strategies and allow for fine-grained control over reasoning depth through scalar multiplication of the steering vector.

Geometric Interpretation and Manifold Analysis We develop a geometric interpretation of reasoning modes in transformer latent space. Let $\mathcal{M} \subset \mathbb{R}^d$ be the manifold of valid hidden states for a given layer. We postulate that deliberative reasoning corresponds to a submanifold $\mathcal{M}_{\text{delib}} \subset \mathcal{M}$ characterized by higher-order geometric properties. The steering vector $\mathbf{h}_{\text{steer}}$ approximates the principal direction connecting direct reasoning states to deliberative reasoning states.

Extraction Protocol The practical extraction of steering vectors requires careful design of contrastive prompts that reliably elicit different reasoning modes from the language model. We develop a systematic protocol that constructs minimal prompt differences to isolate the reasoning mode while controlling for content and context effects.

For a dataset $\mathcal{D} = \{q_i\}_{i=1}^N$ of mathematical problems, we construct contrastive prompt pairs that differ only in their instructions for reasoning approach:

$$\mathbf{p}_i^+ = \text{“Let's think step by step about this problem: ”} \oplus q_i \tag{14}$$

$$\mathbf{p}_i^- = \text{“The answer is: ”} \oplus q_i \tag{15}$$

The design of these prompts is crucial for the quality of the extracted steering vectors. The positive prompt $\mathbf{p}_i^+$ in equation equation 14 explicitly encourages deliberative, step-by-step reasoning through the phrase ”Let's think step by step.” This prompt has been empirically shown to activate chain-of-thought reasoning in language models Wei et al. (2022). The negative prompt $\mathbf{p}_i^-$ in equation equation 15 encourages direct answer generation with minimal intermediate reasoning through ”The answer is.”

The concatenation operator $\oplus$ denotes string concatenation, ensuring that both prompts contain identical problem content q_i while differing only in the reasoning instruction. This controlled difference is essential for isolating reasoning-related activations from problem-specific content.

We extract activations at layer ℓ for both prompt types and compute the normalized steering vector:

$$\mathbf{h}_{\text{steer}}^{(\ell)} = \frac{1}{Z}\sum_{i=1}^{N}\left(\mathbf{h}_i^{+,\ell} - \mathbf{h}_i^{-,\ell}\right), \quad Z = \left\|\sum_{i=1}^{N}\left(\mathbf{h}_i^{+,\ell} - \mathbf{h}_i^{-,\ell}\right)\right\|_2 \tag{16}$$

This equation equation 16 computes the steering vector through several important steps. First, we compute the difference $\mathbf{h}_i^{+,\ell} - \mathbf{h}_i^{-,\ell}$ for each problem i, which captures the activation changes

associated with deliberative versus direct reasoning for that specific problem. The summation $\sum_{i=1}^{N}$ aggregates these differences across all problems in the dataset, allowing us to identify consistent patterns that generalize beyond individual problems.

The normalization factor $Z = \left\| \sum_{i=1}^{N} \left(\mathbf{h}_i^{+,\ell} - \mathbf{h}_i^{-,\ell} \right) \right\|_2$ ensures that the steering vector has unit norm, which is important for two reasons: (1) it makes the steering strength consistent across different layers and models, and (2) it prevents numerical instabilities when applying the steering vector with different scaling factors α.

The choice of layer ℓ significantly impacts the effectiveness of the steering vector. Earlier layers tend to capture low-level linguistic features, while later layers encode higher-level semantic and reasoning patterns. For mathematical reasoning tasks, we empirically find that middle to late layers (typically layers 16-24 in a 32-layer model) provide the most effective steering vectors, as they balance semantic understanding with reasoning capability.

This extraction is performed once per model and remains fixed during controller training, reducing computational overhead to $O(N \cdot d)$ preprocessing, where N is the number of problems in the extraction dataset and d is the hidden dimension. This one-time cost is amortized across all subsequent training and inference, making the approach highly efficient.

Theorem 1 (Steering Vector Consistency). Under mild regularity conditions on the transformer representation space, the steering vector estimator converges to the true steering direction with rate $\mathcal{O}(N^{-1/2})$ as $N \to \infty$.

Steering Effectiveness Analysis The effectiveness of steering vectors can be analyzed through differential geometry. Let $\mathbf{z}_0$ be an initial hidden state and $\mathbf{h}_{\text{steer}}$ be the steering vector. The steered state $\mathbf{z}_1 = \mathbf{z}_0 + \alpha \mathbf{h}_{\text{steer}}$ induces a shift in the probability distribution over next tokens.

We define the *reasoning divergence* as:

$$D_{\text{reason}}(\alpha) = \text{KL}(P_\theta(\cdot|\mathbf{z}_0 + \alpha \mathbf{h}_{\text{steer}}) \| P_\theta(\cdot|\mathbf{z}_0)) \tag{17}$$

Lemma 1 (Steering Monotonicity). For small $\alpha > 0$, $D_{\text{reason}}(\alpha)$ is monotonically increasing in α, indicating consistent directional bias toward deliberative reasoning.

A.7 Group Relative Policy Optimization

Training the pondering controller poses unique challenges due to the sequential nature of decisions and the sparse reward signal (typically received only at the end of the trajectory). Traditional policy gradient methods suffer from high variance, while value-based methods require expensive value function estimation. We address these challenges by adapting Group Relative Policy Optimization (GRPO) Shao et al. (2024), which provides effective variance reduction without the memory overhead of separate value networks.

We adapt Group Relative Policy Optimization for training the pondering controller, which provides variance reduction without requiring a separate value function. GRPO is particularly well-suited for our setting because it can handle variable-length trajectories and provides stable learning signals even with sparse rewards.

Theoretical Motivation The challenge in training adaptive inference policies lies in the high variance of policy gradient estimates. When rewards are sparse and trajectories have variable lengths, standard policy gradient methods often produce noisy gradients that slow learning and require large batch sizes for stability.

Standard REINFORCE Williams (1992) suffers from high gradient variance due to its unbiased but noisy estimation of policy gradients. The gradient estimator $\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|\mathbf{z}_t) \cdot R(\tau)]$ has variance that grows with the length of trajectories and the variance of rewards, making learning unstable.

Proximal Policy Optimization (PPO) Schulman et al. (2017b) addresses this via a value function baseline $V_\phi(\mathbf{z}_t)$ that estimates expected future rewards, reducing variance through the advantage

function $A_t = Q(\mathbf{z}_t, a_t) - V_\phi(\mathbf{z}_t)$. However, this approach doubles memory requirements by introducing a separate value network, and the value function must be trained jointly with the policy, adding complexity.

GRPO achieves comparable variance reduction through in-batch comparisons without requiring additional networks. The key insight is to use the empirical average reward within groups as a natural baseline, leveraging the assumption that samples within a batch provide reasonable comparison points.

Theorem 4 (Variance Reduction). Let $\text{Var}[\nabla_{\text{REINFORCE}}]$ denote the gradient variance of REINFORCE and $\text{Var}[\nabla_{\text{GRPO}}]$ for GRPO with group size G. Then:

$$\text{Var}[\nabla_{\text{GRPO}}] \leq \frac{C}{G} \cdot \text{Var}[\nabla_{\text{REINFORCE}}] \tag{18}$$

under i.i.d. rewards within groups and bounded second moments, where $C \geq 1$ depends on trajectory lengths.

The variance reduction in equation equation 18 occurs because the leave-one-out baseline $b_{\mathcal{G}(i)\setminus i} = \frac{1}{G-1}\sum_{j\in\mathcal{G}(i), j\neq i} r_j$ provides a local estimate of expected reward that correlates with individual rewards while maintaining independence. This correlation reduces the variance of the advantage estimates $A_i = r_i - b_{\mathcal{G}(i)\setminus i}$ compared to using raw rewards r_i. The factor of $1/G$ reflects the approximate variance reduction achieved by averaging over G samples.

Algorithm Design The GRPO algorithm partitions each training batch into groups and computes advantages relative to group averages. This design provides stable learning signals while maintaining computational efficiency.

For a batch $\mathcal{B}$ of size B divided into B/G groups, we compute group-relative advantages:

$$A_i = r_i - b_{\mathcal{G}(i)}, \quad b_{\mathcal{G}(i)} = \frac{1}{G}\sum_{j\in\mathcal{G}(i)} r_j \tag{19}$$

The advantage computation in equation equation 19 is central to GRPO's effectiveness. The individual advantage A_i represents how much better (or worse) sample i performed compared to its group average. The group assignment $\mathcal{G}(i)$ maps sample i to its group, and the baseline $b_{\mathcal{G}(i)}$ is computed as the empirical average of rewards within that group.

The grouping strategy significantly impacts performance. Random grouping ensures unbiased baseline estimation but may group samples with very different difficulties. Alternatively, grouping by similarity (e.g., based on problem type or initial hidden state similarity) can provide more informative baselines but requires careful design to avoid bias.

The GRPO objective combines policy gradient with entropy regularization:

$$\mathcal{L}_{\text{GRPO}}(\phi) = -\mathbb{E}_{\tau_i\sim\mathcal{B}}\left[\sum_{k=0}^{T_i} \log \pi_\phi(a_k^i|\mathbf{z}_k^i)\cdot A_i - \beta_{\text{ent}}\cdot H[\pi_\phi(\cdot|\mathbf{z}_k^i)]\right] \tag{20}$$

The objective function in equation equation 20 consists of two terms. The first term $\sum_{k=0}^{T_i}\log\pi_\phi(a_k^i|\mathbf{z}_k^i)\cdot A_i$ is the policy gradient term that increases the probability of actions from trajectories with positive advantages and decreases the probability of actions from trajectories with negative advantages. The summation over time steps k handles variable-length trajectories naturally.

The second term $\beta_{\text{ent}}\cdot H[\pi_\phi(\cdot|\mathbf{z}_k^i)]$ is entropy regularization where $H[\cdot]$ denotes the entropy of the policy distribution and β_{ent} controls the strength of exploration encouragement. Entropy regularization prevents premature convergence to deterministic policies and ensures sufficient exploration during training.

Convergence Analysis with Bias Guarantees **Lemma 3** (Unbiased Estimation). Under the assumption that group assignments are independent of reward values, the GRPO gradient estimator:

$$\hat{g}_{\text{GRPO}} = \frac{1}{B}\sum_{i=1}^{B}\sum_{t=0}^{T_i}\nabla_\theta \log \pi_\theta(a_t^{(i)}|\mathbf{z}_t^{(i)}) \cdot A_i \tag{21}$$

satisfies $\mathbb{E}[\hat{g}_{\text{GRPO}}] = \nabla_\theta J(\theta)$, ensuring unbiased policy improvement.

Theorem 5 (GRPO Convergence). Let π^* be an optimal policy and π_{ϕ_t} the policy at iteration t. With appropriate learning rate $\eta_t = O(1/\sqrt{t})$:

$$\mathbb{E}[J(\pi^*) - J(\pi_{\phi_T})] \leq O\left(\frac{1}{\sqrt{T}} + \frac{1}{G}\right) \tag{22}$$

where $J(\cdot)$ denotes expected return and G is the group size.

This establishes that GRPO achieves $O(1/\sqrt{T})$ convergence with variance reduction factor $1/G$, confirming theoretical advantages over standard policy gradient methods.

A.8 Multi-Component Reward Engineering

Designing an effective reward function for adaptive reasoning requires balancing multiple competing objectives. Unlike simple classification tasks with binary accuracy, mathematical reasoning involves nuanced aspects such as solution correctness, reasoning completeness, computational efficiency, and output quality. A poorly designed reward function can lead to pathological behaviors such as generating extremely long but incorrect solutions, or conversely, producing correct but unreasonably short answers that lack proper justification. Our reward function addresses five critical aspects of adaptive reasoning through careful component design and magnitude balancing. Each component targets a specific aspect of reasoning quality, and their combination encourages well-rounded performance that matches human expectations for mathematical problem-solving.

A.9 Component Specifications

Accuracy Component R_{acc}: The accuracy component forms the foundation of our reward structure, measuring how well the final answer matches the ground truth. However, binary accuracy (correct/incorrect) provides limited learning signal, especially during early training when most answers are incorrect. Instead, we employ graduated scoring to encourage partial progress and provide smoother gradients:

$$R_{\text{acc}} = \begin{cases} w_{\text{exact}} & \text{if } \hat{y} = y \\ w_{\text{partial}} \cdot \exp\left(-\frac{|\hat{y}-y|}{|y|+\epsilon}\right) & \text{if relative error} < \theta_{\text{tol}} \\ 0 & \text{otherwise} \end{cases} \tag{23}$$

This formulation equation 23 provides three levels of reward. Exact matches receive the full reward w_{exact}, encouraging precise solutions. Near-correct answers receive partial credit through the exponential decay $w_{\text{partial}} \cdot \exp\left(-\frac{|\hat{y}-y|}{|y|+\epsilon}\right)$, where the relative error $\frac{|\hat{y}-y|}{|y|+\epsilon}$ normalizes the absolute error by the ground truth magnitude. The tolerance threshold θ_{tol} defines the maximum relative error for partial credit, preventing the system from rewarding wildly incorrect answers. The ϵ term provides numerical stability when $y \approx 0$.

Computational Efficiency R_{flops}: This component encourages the model to solve problems efficiently, penalizing unnecessary computation. The challenge lies in defining "excess" computation, which varies significantly across problem difficulties. We use adaptive normalization to account for this variability:

$$R_{\text{flops}} = -\lambda_{\text{flops}} \cdot \frac{F - \bar{F}_{\text{history}}}{\sigma_{F_{\text{history}}} + \epsilon} \tag{24}$$

The efficiency reward equation 24 penalizes computational overhead relative to historical norms. The term F represents the FLOPs used for the current problem, while $\bar{F}_{\text{history}}$ and $\sigma_{F_{\text{history}}}$ are running statistics maintained via exponential moving average. This normalization ensures that the penalty adapts to the typical computational requirements, preventing the system from being overly penalized for hard problems that naturally require more computation. The coefficient λ_{flops} controls the strength of the efficiency constraint.

Reasoning Completeness R_{comp}: Mathematical problem-solving typically follows structured stages: problem understanding, computation, verification, and conclusion. This component encourages the model to complete all reasoning stages rather than jumping directly to an answer:

$$R_{\text{comp}} = \sum_{s \in \mathcal{S}_{\text{stages}}} w_s \cdot \mathbb{1}[\text{stage } s \text{ completed}] \tag{25}$$

The completeness reward equation 25 sums contributions from each reasoning stage s in $\mathcal{S}_{\text{stages}} = \{\text{setup}, \text{computation}, \text{verification}, \text{conclusion}\}$. Each stage is detected through pattern matching in the generated text (e.g., looking for setup phrases like "Given that" or verification phrases like "Let me check"). The weights w_s allow for different importance levels across stages, and the indicator function $\mathbb{1}[\text{stage } s \text{ completed}]$ provides binary rewards for stage completion.

Output Quality R_{qual}: Beyond correctness, we want outputs that are well-structured, appropriately detailed, and linguistically coherent. This component assesses coherence through length and perplexity constraints:

$$R_{\text{qual}} = w_{\text{qual}} \cdot \min\left(1, \frac{\ell_{\text{output}}}{\ell_{\text{target}}}\right) \cdot \exp\left(-\frac{\text{PPL}(\hat{y}) - \text{PPL}_{\text{baseline}}}{\sigma_{\text{PPL}}}\right) \tag{26}$$

The quality reward equation 26 has two components. The length term $\min\left(1, \frac{\ell_{\text{output}}}{\ell_{\text{target}}}\right)$ encourages adequate detail by penalizing outputs that are significantly shorter than the target length ℓ_{target}, while capping the reward at 1 to avoid encouraging excessive verbosity. The perplexity term $\exp\left(-\frac{\text{PPL}(\hat{y}) - \text{PPL}_{\text{baseline}}}{\sigma_{\text{PPL}}}\right)$ measures linguistic coherence, where $\text{PPL}(\hat{y})$ is the perplexity of the generated output and $\text{PPL}_{\text{baseline}}$ is a baseline perplexity from high-quality examples.

Anti-Repetition $R_{\text{anti-rep}}$: Language models can sometimes generate repetitive text, especially when encouraged to produce longer outputs. This component penalizes redundancy at multiple granularities:

$$R_{\text{anti-rep}} = -\sum_{g \in \{1,2,3\}} \beta_g \cdot \frac{|\text{repeated}_g(\hat{y})|}{|\hat{y}|} \tag{27}$$

The anti-repetition reward equation 27 penalizes n-gram repetitions for $g \in \{1, 2, 3\}$ (unigrams, bigrams, trigrams). The term $|\text{repeated}_g(\hat{y})|$ counts the number of repeated n-grams of order g in the output $\hat{y}$, normalized by the total output length $|\hat{y}|$. The coefficients β_g allow for different penalty strengths across n-gram orders, typically with $\beta_1 < \beta_2 < \beta_3$ since higher-order repetitions are more problematic than single word repetitions.

A.10 Reward Balancing Theory

To prevent any single component from dominating, we enforce magnitude constraints:

Theorem 6 (Reward Balance Condition). For stable learning, the reward components must satisfy:

$$\forall i, j \in \{\text{acc}, \text{flops}, \text{comp}, \text{qual}, \text{rep}\} : \quad \frac{\mathbb{E}[|R_i|]}{\mathbb{E}[|R_j|]} \in [\rho^{-1}, \rho] \tag{28}$$

where $\rho \in [2, 10]$ is the maximum imbalance ratio.

We achieve this through adaptive weight scaling:

$$w_i^{(t+1)} = w_i^{(t)} \cdot \exp\left(\eta \cdot \left(\log \bar{R}_{\text{target}} - \log \bar{R}_i^{(t)}\right)\right) \tag{29}$$

A.11 Curriculum Learning Framework

Training adaptive inference policies directly from random initialization faces several challenges: sparse rewards (most randomly generated trajectories produce incorrect answers), high variance in trajectory quality, and the exploration problem (discovering good stopping points through random exploration is inefficient). Curriculum learning addresses these challenges by providing structured guidance that gradually transfers control from teacher demonstrations to autonomous learning. We employ a three-stage curriculum that progressively transfers control from teacher demonstrations to autonomous learning. This approach is inspired by how humans learn complex skills: starting with guided practice, progressing to supervised practice with feedback, and finally achieving independent mastery.

Stage Progression The curriculum is designed to provide a smooth transition from full supervision to autonomous learning. The progression is governed by a curriculum probability that determines the mix between teacher and student control at each training step.

The curriculum probability follows a piecewise linear schedule:

$$p_{\text{curriculum}}(t) = \begin{cases} 1.0 & t \in [0, T_1) \\ 1 - \frac{t - T_1}{T_2 - T_1} & t \in [T_1, T_2) \\ 0.0 & t \geq T_2 \end{cases} \tag{30}$$

This schedule equation 30 defines three distinct phases. In Stage 1 ($t \in [0, T_1)$), $p_{\text{curriculum}}(t) = 1.0$ indicates pure teacher forcing, where all pondering decisions are made by a teacher policy that encourages moderate pondering (typically 3-5 steps). This provides the controller with abundant examples of reasonable stopping behavior and establishes a foundation of sensible pondering patterns.

Stage 2 ($t \in [T_1, T_2)$) implements gradual transition with $p_{\text{curriculum}}(t) = 1 - \frac{t - T_1}{T_2 - T_1}$, linearly decreasing the probability of teacher guidance. This mixed training allows the student policy to gradually take control while still receiving guidance when needed. The linear schedule ensures smooth transition without abrupt changes that could destabilize learning.

Stage 3 ($t \geq T_2$) represents autonomous learning with $p_{\text{curriculum}}(t) = 0.0$, where the student controller makes all decisions independently. By this stage, the controller has learned basic pondering patterns and can explore more sophisticated strategies through reinforcement learning.

The boundaries $T_1 = 500$ and $T_2 = 1500$ are chosen based on empirical observations about controller learning dynamics. The initial 500 steps provide sufficient teacher demonstrations to establish baseline behavior, while the 1000-step transition period allows gradual adaptation without overwhelming the learning process.

During mixed training, we sample the guidance source at each training step:

$$\text{source}(t) \sim \text{Bernoulli}(p_{\text{curriculum}}(t)) \tag{31}$$

This sampling equation 31 determines whether each trajectory uses teacher guidance (source = 1) or student control (source = 0). The Bernoulli distribution ensures that the expected fraction of teacher-guided trajectories matches the curriculum schedule while providing stochastic variation that prevents overfitting to the transition points.

Quality Gates As teacher guidance diminishes, maintaining training stability becomes crucial. Without quality control, the student policy might generate extremely poor trajectories that provide misleading learning signals. Quality gates address this challenge by filtering trajectories before they contribute to parameter updates.

In the autonomous stage, we implement quality gates that reject low-quality trajectories:

$$\mathcal{Q}(\tau) = \mathbb{1}[R_{\text{comp}}(\tau) > \theta_{\text{comp}}] \cdot \mathbb{1}[R_{\text{qual}}(\tau) > \theta_{\text{qual}}] \tag{32}$$

The quality gate equation 32 implements a conjunction of two conditions. The first condition $\mathbb{1}[R_{\text{comp}}(\tau) > \theta_{\text{comp}}]$ ensures that trajectories demonstrate reasonable reasoning completeness, measured by the presence of key reasoning stages. The threshold θ_{comp} is set to require at least basic problem setup and computation stages.

The second condition $\mathbb{1}[R_{\text{qual}}(\tau) > \theta_{\text{qual}}]$ filters trajectories based on output quality, ensuring that the generated text meets minimum standards for coherence and appropriateness. The threshold θ_{qual} prevents the inclusion of trajectories with excessively repetitive, incoherent, or truncated outputs.

Only trajectories satisfying $\mathcal{Q}(\tau) = 1$ contribute to gradient updates, ensuring stable learning as teacher guidance diminishes. This filtering mechanism prevents the policy from learning from extremely poor examples while still allowing reasonable exploration. The thresholds are set conservatively to maintain a balance between quality control and learning diversity. To ensure transparency, we provide concrete examples of how Large Language Models (LLMs) were used in the preparation of this manuscript:

- **Grammar refinement:** For instance, when an early draft contained the sentence "Our method significantly reduce computation cost," the LLM was used to correct it to "Our method significantly reduces computational cost."
- **Clarity improvement:** A verbose draft sentence such as "In this part we attempt to show that our model works in a way that is both effective and efficient" was polished by the LLM to "This section demonstrates that our model is both effective and efficient."
- **Flow adjustment:** When two adjacent sentences ("We introduce the FR-Ponder framework. It adapts inference dynamically.") appeared disjoint, the LLM suggested a smoother transition: "We introduce the FR-Ponder framework, which dynamically adapts inference."

These examples illustrate that the LLM's role was restricted to language refinement. All technical ideas, theoretical results, and experimental contributions originated from the authors.