ReMi: Efficient Latent Reasoning with Content-Aware Stopping

Anonymous ACL submission

Abstract

Latent reasoning methods compress explicit Chain-of-Thought (CoT) into continuous hidden states for efficient inference, but predominantly assign a fixed number of latent steps to all inputs regardless of difficulty. This wastes computation on simple problems and, as we show, redundant latent positions can introduce interfering signals that corrupt the final answer—overthinking in latent space. Moreover, standard halting signals (e.g., token entropy) are inapplicable at latent positions where no discrete predictions are made. We propose ReMi, an adaptive latent reasoning framework that learns when to stop thinking through a dedicated exit head operating directly on hidden states, paired with per-sample adaptive allocation and difficulty-aware progressive training. Experiments on GSM8K and ProsQA across two model scales (GPT-2, LLaMA-1B) show that ReMi achieves **5.2**× compression over explicit CoT, improves OTC (ACC / Reasoning Tokens) by 72%, and outperforms its own best fixed-length configuration by +6.2% in ablation. We further show that coupling halting to the token prediction objective (as in CoLaR) degrades accuracy by 13%, while ReMi's decoupled exit head preserves the backbone's reasoning capacity. *The code will be open-sourced upon paper acceptance.*

1 Introduction

Chain-of-Thought (CoT) reasoning (Wei et al., 2022) has become the standard paradigm for enhancing the reasoning capabilities of large language models, enabling them to solve complex multi-step tasks by generating intermediate reasoning steps before the final answer. However, this effectiveness comes at a significant computational cost: reasoning chains are typically an order of magnitude longer than the answer itself, dominating inference latency and KV cache memory consumption. The problem is particularly acute for simple queries—a one-step arithmetic problem receives the same lengthy treatment as an 8-step multi-hop problem, wasting computation on verbose reasoning that contributes nothing to the final answer (Chen et al., 2024).

Latent reasoning methods (Hao et al., 2025; Wei et al., 2026; Shen et al., 2025) offer a compelling solution by replacing explicit CoT tokens with continuous hidden-state vectors, decoupling reasoning depth from token generation cost. In these approaches, the model performs iterative hidden-state refinement at designated latent positions, then decodes the answer conditioned on the accumulated latent context—achieving substantial compression without generating any intermediate text. However, a critical limitation persists in the dominant paradigm: methods such as Coconut, CODI, and SimCoT assign a fixed number of latent steps to every input, regardless of problem difficulty. A trivial problem and a complex multi-hop problem receive identical computational budgets. Our diagnostic experiments (§2.2) reveal two consequences of this rigidity: (1) accuracy saturates well before the allocated maximum, with 75% of solvable samples converging within half the fixed budget; and (2) redundant latent positions beyond the saturation point can actively introduce interfering signals that corrupt the final answer—overthinking manifests in latent space as well.

A natural question arises: can we teach the model when to stop thinking? In token-based CoT, halting methods rely on output-distribution signals such as entropy or top-1 probability to detect reasoning completion (Muennighoff et al., 2025; Laaouach, 2025). But in latent reasoning, no discrete tokens are produced at intermediate positions—there are no probabilities to threshold, no tokens to evaluate confidence over. We show experimentally that forcing token-level supervision onto latent positions degrades answer accuracy by

over 10 percentage points (Appendix A), confirming a fundamental tension between latent reasoning and discrete prediction objectives. The halting problem in latent space thus requires a fundamentally different approach.

We propose ReMi, which addresses this challenge through three interdependent design choices: (1) an **exit head**—a lightweight MLP classifier operating directly on hidden states—that learns to predict whether reasoning is complete at each latent position; (2) **per-sample adaptive latent allocation** during training, where each sample's latent count reflects its actual reasoning complexity rather than a uniform stage-wide constant; and (3) a **difficulty-aware progressive curriculum** that naturally creates diverse stopping positions within each batch, providing the exit head with meaningful contrastive supervision. The step decoder from SimCoT (Wei et al., 2026) is retained as an auxiliary training signal and post-hoc diagnostic tool. Together, these components enable ReMi to terminate reasoning early for simple problems while allocating more computation to complex ones—achieving 5.2× compression over explicit CoT on GSM8K and 2.2× over fixed latent allocation on ProsQA.

Our contributions are as follows:

- We identify and formalize the **halting problem in latent reasoning**: token-level confidence signals are inapplicable in latent space, and fixed allocation causes latent-space overthinking where redundant positions introduce interfering computations.
- We propose **ReMi**, an adaptive-length latent reasoning framework with difficulty-aware training. On GSM8K, ReMi improves OTC by 72% over fixed allocation, saves 47% of latent computation, and outperforms its own best fixed-length configuration by +6.2% in ablation. Results are consistent across GPT-2 and LLaMA-1B.
- We demonstrate through ablations and case studies that **token-level exit supervision degrades performance** (CoLaR-5: −13.1% vs ReMi on LLaMA-1B), while ReMi's decoupled exit head preserves the backbone's reasoning capacity.

2 Background and Related Works

2.1 Related Works

CoT reasoning. Chain-of-Thought prompting (Wei et al., 2022) improves reasoning by eliciting intermediate steps, with extensions including multi-path aggregation (Wang et al., 2023), tree search (Yao et al., 2023), and training-time optimization (Zelikman et al., 2022; Lightman et al., 2024). However, generating lengthy reasoning chains dominates inference latency, motivating latent reasoning approaches that decouple reasoning depth from token generation cost.

Latent reasoning. Coconut (Hao et al., 2025) replaces discrete CoT tokens with continuous thought vectors via progressive curriculum learning. CODI (Shen et al., 2025) applies hidden-state-level self-distillation to transfer reasoning from explicit CoT to implicit CoT. SimCoT (Wei et al., 2026) adds step-level supervision through an auxiliary decoder, stabilizing latent representations during scaling. CoLaR (Tan et al., 2025) introduces compressed latent reasoning with a variable compression factor and leverages reinforcement learning to encourage shorter reasoning paths. Soft Thinking (Zhang et al., 2025) reasons in a continuous concept space via probability-weighted embedding mixtures.

Halting in token-based reasoning. For explicit CoT, S1 (Muennighoff et al., 2025) uses budget forcing to control reasoning length, while HALT-CoT (Laaouach, 2025) triggers early termination based on token-level confidence signals (e.g., entropy or max probability of the next-token distribution).

Halting in latent reasoning. For latent reasoning, Coconut briefly considers training a binary classifier for autonomous termination but adopts fixed padding for simplicity without further investigation (Hao et al., 2025). SimCoT (Wei et al., 2026) likewise fixes the step count. CoLaR (Tan et al., 2025) achieves variable-length latent chains where the Language Head implicitly signals termination, but this relies on token-level probability—a signal we show to be unreliable at pure latent positions (§2.2). Soft Thinking's Cold Stop (Zhang et al., 2025) similarly depends on output entropy. The core challenge remains: at latent positions where no discrete predictions are made, neither

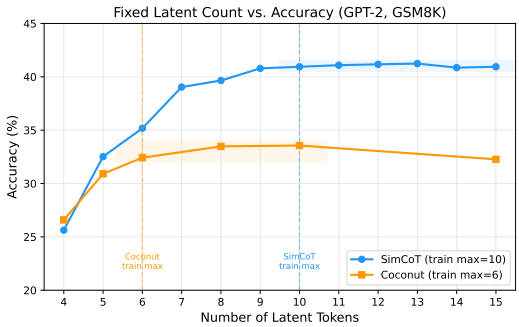

Figure 1: Accuracy vs. fixed latent token count at inference (GPT-2, GSM8K).

token probability nor entropy provides a reliable halting signal. ReMi addresses this by learning a dedicated exit head operating directly on hidden states, trained with per-sample variable-length supervision.

2.2 The Halting Problem in Latent Reasoning

We now analyze why the halting problem in latent reasoning is both practically important and fundamentally different from its token-based counterpart. Full experimental details and supporting evidence are provided in Appendix A.

2.2.1 Problems of Fixed Latent Allocation

Existing latent reasoning methods assign a uniform number of latent steps T to all inputs regardless of problem complexity. This leads to: (1) **Computational waste**—simple problems need only $\sim$50% of the allocated computation to reach the correct answer (Appendix A); (2) **Capacity mismatch**—accuracy plateaus or even degrades beyond a certain step count (Figure 1); and (3) **Overthinking**—redundant latent positions can introduce interfering signals that corrupt the final answer (Chen et al., 2024), as demonstrated in §4.5.

Figure 1 provides direct evidence for issues (1) and (2). We evaluate two trained models—SimCoT (train max=10 tokens) and Coconut (train max=6 tokens)—at varying latent counts during inference. Both exhibit a clear ceiling effect: accuracy plateaus near the training maximum and does not benefit from additional tokens. Coconut further shows degradation at 15 tokens (-1.3%), confirming that over-allocation actively harms performance.

2.2.2 Why Token-Based Halting Signals Do Not Fit

Token-based halting methods such as HALT-CoT (Laaouach, 2025) determine when to stop reasoning based on output token probability signals evaluated at each reasoning step. The two most commonly used indicators are:

$$H(\mathbf{p}_t) = -\sum_{v \in \mathcal{V}} p_t(v) \log p_t(v), \tag{1}$$

$$p_{\max}^{(t)} = \max_{v \in \mathcal{V}} p_t(v), \tag{2}$$

where $\mathbf{p}_t = \mathrm{softmax}(\mathbf{h}_t \mathbf{W}_{\mathrm{lm}})$ is the next-token probability distribution over vocabulary $\mathcal{V}$ at reasoning step t, obtained by projecting the hidden state $\mathbf{h}_t$ through the language model head $\mathbf{W}_{\mathrm{lm}}$. Halting is triggered when $H(\mathbf{p}_t)$ falls below a threshold (or equivalently, $p_{\max}^{(t)}$ exceeds a threshold), indicating the model is "confident" about its next output.

However, these signals are fundamentally **inapplicable in latent reasoning**. At latent positions, the model produces only continuous hidden states—no discrete token predictions are made, and no token-level supervision is applied during training. The distribution $\mathbf{p}_t$ obtained by projecting $\mathbf{h}_t$ through $\mathbf{W}_{\mathrm{lm}}$ therefore carries no meaningful signal about reasoning completeness. A natural remedy—introducing auxiliary token-level supervision at latent positions—severely degrades answer accuracy (see Appendix A), because latent reasoning and explicit token prediction serve conflicting optimization objectives: the former benefits from unconstrained continuous representations, while the latter demands alignment with a discrete distribution.

These findings motivate ReMi: instead of repurposing token-level signals, we learn a dedicated **exit head**—a lightweight classifier operating directly on hidden states $\mathbf{h}_t$—trained with per-sample variable-length latent allocation to acquire a content-aware stopping policy without interfering with the NTP objective.

3 Method

We present ReMi, an adaptive latent reasoning framework that learns when to stop thinking. Building on the latent reasoning formulation described in §2.2, ReMi introduces an **exit head** that determines per-sample stopping, paired with a difficulty-aware training strategy that enables the exit head to

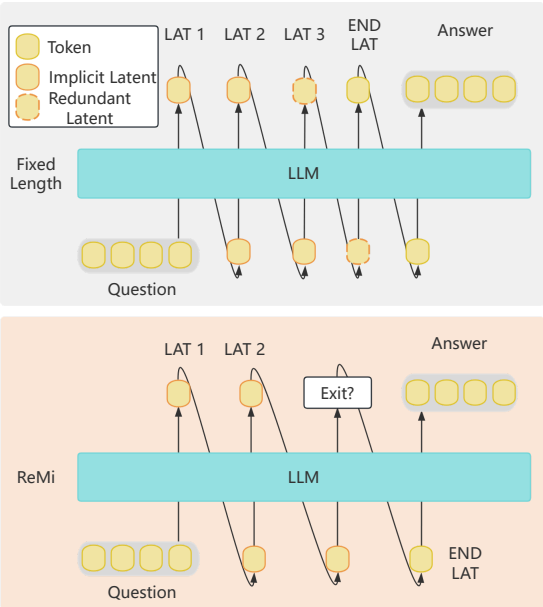

Figure 2: Fixed allocation vs. ReMi adaptive stopping. **Top**: for a sample requiring only 2 reasoning steps, fixed-length methods (Coconut, CODI, SimCoT) pad additional latent placeholders (LAT 3) to match a predetermined count, wasting computation on redundant positions. **Bottom**: ReMi evaluates the exit head after each latent step; once the stopping condition is met, the model immediately proceeds to answer generation.

learn content-aware decisions. Figure 2 illustrates the difference between fixed allocation and ReMi's adaptive stopping.

3.1 Adaptive Exit Head

At each latent position t, we extract the hidden state $\mathbf{h}_t \in \mathbb{R}^d$ from the backbone's final layer and pass it through a lightweight two-layer MLP:

$$p_{\text{exit}}(t) = \sigma\big(\mathbf{W}_2 \cdot \text{GELU}(\mathbf{W}_1 \mathbf{h}_t + \mathbf{b}_1) + \mathbf{b}_2\big), \quad (3)$$

where $\mathbf{W}_1 \in \mathbb{R}^{(d/4)\times d}$, $\mathbf{W}_2 \in \mathbb{R}^{2\times(d/4)}$, and σ denotes softmax. The exit head outputs a 2-class distribution: *continue* vs. *end*.

Exit supervision. During training, each sample has a natural number of reasoning steps N_i derived from its CoT annotation. We construct binary exit labels:

$$y_t^{\text{exit}} = \begin{cases} 0 \text{ (continue)}, & \text{if } t < N_i, \\ 1 \text{ (end)}, & \text{if } t = N_i. \end{cases} \quad (4)$$

The exit loss is the cross-entropy over all latent positions:

$$\mathcal{L}_{\text{exit}} = -\frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \sum_{t=1}^{N_i} \log p_\phi(y_t^{\text{exit}} \mid \mathbf{h}_t^{(i)}), \quad (5)$$

where ϕ denotes the exit head parameters and $\mathcal{B}$ is the training batch.

Adaptive inference. At test time, the model iteratively performs latent steps until the exit head predicts *end* with probability exceeding a threshold τ:

$$T^* = \min\{t : p_{\text{exit}}(t)[1] > \tau\}, \quad (6)$$

with a safety bound $T_{\max}$. This enables the model to allocate more computation to harder problems while terminating early for simpler ones.

3.2 Difficulty-Aware Adaptive Training

A naive exit head trained on uniform-length latent sequences degenerates into a trivial position predictor ("always output *end* at position T"). To learn a meaningful, content-aware stopping policy, the exit head must observe different samples stopping at different positions. We achieve this through two design choices.

Per-sample adaptive latent count. Unlike Coconut and SimCoT which assign a uniform latent count to all samples within a training stage, we tie each sample's latent count to its actual reasoning complexity:

$$T_i = \min(N_i,\ T_{\max}), \quad (7)$$

where N_i is the number of CoT steps for sample i. Within the same batch, a 1-step problem uses 1 latent position while an 8-step problem uses 8, providing the exit head with contrastive supervision signals.

Progressive curriculum. Following Hao et al. (2025), we adopt a stage-wise curriculum where latent capacity increases across epochs. At stage s, the latent count for sample i is:

$$T_i^{(s)} = \min(N_i,\ s \cdot c), \quad (8)$$

where c is the number of latent tokens per reasoning step. This creates a per-sample curriculum: simple samples reach their natural capacity early and remain stable, while complex samples progressively gain more reasoning depth. This stands in contrast to the uniform curriculum of prior work where all samples transition simultaneously.

3.3 Training Objective

The complete training objective combines three loss components:

$$\mathcal{L} = \mathcal{L}_{\text{NTP}} + \lambda_{\text{exit}} \cdot \mathcal{L}_{\text{exit}} + \lambda_{\text{step}} \cdot \mathcal{L}_{\text{step}}, \quad (9)$$

where $\mathcal{L}_{\text{NTP}}$ is the standard next-token prediction loss on the answer tokens, and λ_{exit}, λ_{step} are weighting coefficients.

Step-level reconstruction loss. Following SimCoT (Wei et al., 2026), we employ a step decoder—a small language model that takes $\mathbf{h}_t$ as a conditioning prefix and autoregressively reconstructs the corresponding reasoning step $\mathbf{s}_t$:

$$\mathcal{L}_{\text{step}} = -\frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \sum_{t=1}^{N_i} \log p_\psi(\mathbf{s}_t^{(i)} \mid \mathbf{h}_t^{(i)}). \quad (10)$$

This loss ensures that latent representations capture meaningful reasoning content rather than degenerate, and enables post-hoc auditing of the latent reasoning process at inference time.

Design interdependence. The per-sample adaptive count, exit head, and progressive curriculum are mutually dependent: without per-sample variability the exit head cannot learn content-aware stopping; without the exit head the adaptive count cannot be exploited at inference; without progressive training, variable-length latents from the start cause instability.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate on **GSM8K** (Cobbe et al., 2021), a grade-school math reasoning dataset with 7,473 training and 1,319 test problems. The annotated CoT solutions span 1–8 reasoning steps (mean 3.26), providing natural variability in reasoning depth that is well-suited for evaluating adaptive computation allocation. We additionally evaluate on **ProsQA** (Hao et al., 2025), a synthetic propositional logic dataset requiring 5-hop graph traversal (304 validation samples), with fixed reasoning depth that complements GSM8K's variable-depth reasoning.

Models. We evaluate across two model scales: GPT-2 (124M parameters) for controlled experiments and detailed analysis, and LLaMA-1B for medium-scale validation of scalability.

Baselines. Our primary reference is **CoT SFT**, standard chain-of-thought fine-tuning that generates explicit reasoning text before the final answer. On GSM8K, CoT SFT produces an average of 27.3 reasoning tokens (GPT-2) and 25.4 tokens (LLaMA-1B) per problem. All latent reasoning methods aim to compress this explicit reasoning into fewer latent steps while retaining accuracy.

We compare against: **Coconut** (Hao et al., 2025): progressive latent reasoning with a fixed count of 6 latent tokens, achieving 4.6× compression over CoT SFT; **SimCoT**: an improved variant that adds step-level interpretability via a step decoder, using 10 fixed latent tokens (2.7× compression); **Training-free halting**: post-hoc adaptive stopping applied on top of SimCoT. We design two indicators: (i) **GeoMean**—the geometric mean of top-1 token probabilities over the last k latent steps, halting when it exceeds threshold τ; (ii) **EMA**—an exponential moving average of the same signal with smoothing factor α. Both operate without additional training by projecting latent hidden states through the language model head, achieving modest compression (8.88 tokens, 3.1×) with minimal accuracy loss; **CoLaR-5** (Tan et al., 2025): a token-level adaptive method that trains the model to emit an explicit halt token, supervising exit decisions directly within the language modeling objective. It uses 5.57 tokens on average (4.9× compression) but suffers from degraded generalization due to coupling halting with token prediction.

ReMi (ours) uses a learned exit head for adaptive stopping, averaging 5.27 latent tokens on GPT-2 (5.2× compression) and 5.2 tokens on LLaMA-1B (4.9× compression)—the highest compression among all methods.

Metrics. We report three metrics: (1) **Accuracy (ACC)**: exact-match accuracy on final answers; (2) **Avg Latent Steps**: mean number of latent reasoning steps used at inference; (3) **Output Token Contribution (OTC)** (Lv et al., 2025): defined as OTC $=$ ACC/Reasoning Tokens, measuring the accuracy gain per reasoning token. In latent reasoning, each token requires a full backbone forward pass, making this metric directly proportional to inference cost.

Implementation details. We follow the training setup of prior work (Hao et al., 2025; Wei et al., 2026). Full configuration details (hyperparameters, optimizer settings, stage schedule) are provided in Appendix C.

4.2 Main Results

Table 1 presents results on GSM8K. All methods marked with † are our reproductions trained under identical settings for fair comparison.

Table 1: Main results on GSM8K. All latent methods evaluated at max stage. Compression = $T_{\text{CoT}}/T_{\text{method}}$, measuring reasoning token reduction relative to CoT SFT. †Our reproduction; ‡from their report.

Method	Model	ACC(%)	Tokens	Compression	OTC
CoT SFT	GPT-2	43.7	27.3	1.0×	—
Coconut† (n=6)	GPT-2	32.42	6.0	4.6×	0.054
SimCoT† (n=10)	GPT-2	39.79	10.0	2.7×	0.040
SimCoT†+GeoMean	GPT-2	40.71	8.88	3.1×	0.046
ReMi	**GPT-2**	**36.14**	**5.27**	**5.2×**	**0.069**
CoT SFT	LLaMA-1B	58.1	25.4	1.0×	—
Coconut† (n=6)	LLaMA-1B	34.5	6.0	4.2×	0.058
SimCoT† (n=10)	LLaMA-1B	44.32	10.0	2.5×	0.044
CoLaR-5‡	LLaMA-1B	26.8	5.57	4.6×	0.048
ReMi	**LLaMA-1B**	**39.85**	**5.2**	**4.9×**	**0.077**

Observation 1: Superior OTC. ReMi achieves an OTC of 0.069 (GPT-2) and 0.077 (LLaMA-1B), representing a **72%** and **75%** improvement over SimCoT's fixed allocation respectively. This demonstrates that learned adaptive stopping extracts significantly more value from each reasoning token than uniform allocation, consistently across model scales.

Observation 2: Favorable compute-accuracy tradeoff. ReMi retains 90.8% of SimCoT's accuracy (36.14% vs 39.79%) while saving 47% of latent computation (5.27 vs 10.0 steps). Under a strict budget of ≤6 tokens, ReMi (36.14%) is the only method exceeding Coconut (32.42%). For simple problems (1–2 GT steps, 33% of the test set), ReMi exits in ∼3 steps on average, enabling near-instant inference on easy inputs.

Observation 3: Comparison with training-free methods. GeoMean halting (geometric mean of top-1 probabilities over a sliding window) achieves the best accuracy among training-free approaches, saving 11% computation while nearly matching fixed-allocation accuracy. ReMi offers a complementary operating point: **4× more computation savings** (47% vs 11%) through end-to-end learned stopping, at the cost of a modest accuracy reduction. Unlike post-hoc methods that require per-model threshold tuning, ReMi's exit head is trained jointly and requires no manual calibration.

Observation 4: Coupling halting to token prediction hurts generalization. CoLaR-5 (Tan et al., 2025), which couples the halting decision to the language modeling objective (the model learns via RL to transition from latent to answer tokens, with termination implicitly determined by the Language Head), achieves only 26.8% accuracy on LLaMA-1B despite using a comparable token budget (5.57 tokens). Under similar compression (4.6× vs ReMi's 4.9×), ReMi outperforms CoLaR-5 by **+13.1pp** (39.85% vs 26.8%). This is consistent with our analysis in §2.2: coupling halting to the token prediction objective constrains latent representations and damages generalization. In contrast, ReMi's auxiliary exit head decouples stopping from generation, preserving the backbone's reasoning capacity.

Table 2: Results on ProsQA (GPT-2, 5-hop reasoning, max stage). Compression is relative to fixed allocation (n=6).

Method	ACC(%)	Tokens	Compression	OTC
Coconut	91.4	6.0	1.0×	0.152
SimCoT	96.4	6.0	1.0×	0.161
ReMi	**93.4**	**2.67**	**2.2×**	**0.350**

Observation 5: Consistent across scales. On LLaMA-1B, ReMi maintains the same pattern: OTC of 0.077 (+75% vs SimCoT), 48% computation savings, and accuracy surpassing Coconut (39.85% vs 34.5%). The exit head's adaptive strategy generalizes across model capacities without task-specific tuning.

Results on ProsQA. Table 2 presents results on ProsQA, a structured 5-hop reasoning task. ReMi achieves 93.4% accuracy with only 2.67 average latent tokens—55% fewer than the fixed 6 tokens used by Coconut and SimCoT. Its OTC (0.350) is more than double that of SimCoT (0.161), confirming that adaptive stopping is particularly effective when reasoning structure is regular.

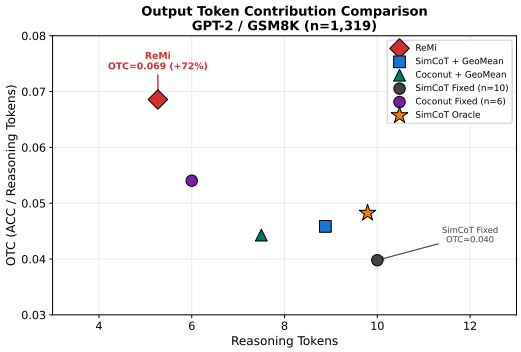

Figure 3: Output Token Contribution (OTC = ACC / Reasoning Tokens) comparison. ReMi achieves the highest OTC, indicating optimal utilization of latent computation.

Table 3: Exit head behavior: predicted latent steps vs. problem difficulty (GPT-2, GSM8K, full test set). The exit head allocates monotonically more computation to harder problems.

GT CoT Steps	Count	Avg Steps	Std
1–2 (easy)	440	3.29	1.46
3–4 (medium)	654	5.58	1.77
5–6 (hard)	195	7.52	1.68
7+ (very hard)	30	8.57	0.90
Overall	**1319**	**5.17**	—

4.3 Analysis

Efficiency landscape. Figure 3 visualizes the OTC landscape. ReMi occupies the Pareto-optimal region in the (Tokens, OTC) space, achieving the highest OTC among all methods. The oracle upper bound (per-sample optimal stopping on SimCoT, ACC=47.23%, OTC=0.048) indicates significant headroom remains for improving adaptive strategies.

Per-difficulty exit behavior. We analyze the exit head's stopping decisions grouped by problem difficulty (measured by ground-truth CoT step count) on the full test set (1,319 samples). Table 3 shows a strong monotonic correlation between problem difficulty and allocated computation: samples requiring 1–2 CoT steps receive an average of 3.29 latent steps, while 7+ step problems receive 8.57 steps. This confirms the exit head has learned a content-aware stopping policy rather than a trivial fixed-position rule.

Accuracy-efficiency tradeoff. We acknowledge that ReMi's absolute accuracy (36.14%) is lower than SimCoT fixed (39.79%) by 3.7 percentage points. This gap arises from gradient competition between the exit loss and NTP loss during joint training, and occasional premature stopping on borderline samples. However, ReMi's OTC (0.069) demonstrates that each latent step carries substantially more "reasoning value." In deployment with latency constraints, the 47% computation reduction directly translates to inference speedup while maintaining accuracy above Coconut.

Table 4: Fixed vs. adaptive latent steps (GPT-2, GSM8K).

Tokens	ACC(%)	Note
n=1	4.47	Near random
n=2	16.14	
n=4	27.80	
n=8	29.92	Best fixed
n=10	26.06	Over-thinking
n=20	14.24	Degradation
Adaptive (5.27)	**36.14**	**+6.2pp**

4.4 Ablation Study

Adaptive vs. fixed latent count. To verify that the exit head learns a meaningful stopping policy (rather than a trivial strategy), we force the trained ReMi model to use fixed latent counts at inference, ignoring the exit head's predictions. Results are shown in Table 4.

The adaptive strategy (36.14%) outperforms ReMi's own best fixed-length configuration (n=8, 29.92%) by **+6.2%**, using fewer average steps (5.27 vs 8). This demonstrates that the exit head has learned to allocate computation where it matters most on a per-sample basis.

Two additional observations emerge: (1) Performance degrades significantly beyond n=8, indicating the model has not learned to utilize latent steps beyond its effective training range. (2) The optimal fixed point (n=8) differs from the training maximum (n=10), confirming that many samples do not benefit from maximum-depth reasoning.

Component ablation. Table 5 isolates the contribution of each design choice. Removing the step decoder (-4.9%) causes latent states to lose structured information, making "done" and "not done" indistinguishable for the exit head. Bypassing progressive training (-8.1%) exposes the exit head to sparse labels (mostly "continue") from the start, collapsing it to an always-continue strategy. Both confirm the mutual dependence described in §3.3.

Table 5: Component ablation (GPT-2, GSM8K). Each row removes one design choice from full ReMi.

Configuration	ACC (%)	Avg Tokens
Full ReMi	36.14	5.27
w/o Step Decoder	31.21 (-4.9)	4.82
w/o Progressive Training	28.03 (-8.1)	9.41

Table 6: Effect of exit threshold τ on ReMi (GPT-2, GSM8K). Lower τ exits earlier; higher τ is more conservative. Peak accuracy occurs at τ=0.7, not τ=0.9, due to overthinking at maximum steps.

τ	ACC (%)	Avg Tokens
0.3	33.97	4.88
0.5 (default)	36.14	5.20
0.7	**37.98**	5.54
0.9	37.53	6.02

Exit threshold sensitivity. We vary the exit threshold τ at inference to examine the accuracy-computation tradeoff controlled by this single hyperparameter (Table 6).

The threshold provides a smooth knob between aggressive compression (τ=0.3: 4.88 tokens, 33.97%) and conservative reasoning (τ=0.7: 5.54 tokens, 37.98%). Notably, τ=0.9 performs worse than τ=0.7, as overly conservative stopping forces more samples to reach $T_{\max}$=10, triggering the overthinking effect described in §2.2. At all operating points, ReMi outperforms its own best fixed-length configuration (29.92% at n=8), confirming that the exit head provides meaningful decisions regardless of threshold choice.

4.5 Interpretability and Overthinking

The step decoder enables post-hoc reconstruction of latent reasoning traces. We decode latent hidden states from both ReMi (adaptive exit) and SimCoT (fixed 10 steps) on GSM8K test samples. Three key findings emerge (full case studies in Appendix B):

Finding 1: Overthinking—redundant steps introduce interference. When SimCoT allocates more latent steps than needed, effective computation concentrates in the first 3–5 positions, while excess positions degenerate into repetitive patterns (`### 3`, `### -`) or, worse, introduce spurious computations that correlate with incorrect answers. In multiple cases, the model reaches the correct intermediate value early but a later position introduces an erroneous operation (e.g., an extra $\times 2$ multiplication) that corrupts the final answer. ReMi avoids this by exiting before redundant computation can accumulate.

Finding 2: Adaptive exit produces compact, meaningful traces. ReMi's exit head consistently allocates $\sim$3 steps for easy problems (1–2 GT steps). Decoded latents from these cases exhibit exact text matches with ground-truth reasoning steps, demonstrating that latent hidden states genuinely encode structured arithmetic operations retrievable by the step decoder. Each latent position carries meaningful computation rather than padding.

Finding 3: Failure modes of early exit. Adaptive stopping can fail in two characteristic ways: (a) *insufficient capacity*: for problems requiring 4+ GT steps, 3 latent steps lack the computational depth to complete the full reasoning chain; (b) *missing final conversion*: the model computes the correct intermediate value but exits before performing a required unit conversion or aggregation step (e.g., computing 84 eggs but failing to convert to 7 dozens). These patterns suggest that exit head training could benefit from harder negative mining at penultimate positions. A detailed error taxonomy with 4 failure types is provided in Appendix B.

5 Conclusion

We present ReMi, which addresses the overthinking problem in latent reasoning through a content-aware exit head with difficulty-aware progressive training. ReMi achieves 5.2$\times$ compression over explicit CoT and 72% OTC improvement, while avoiding a concrete failure mode of fixed allocation: redundant positions that introduce spurious computations correlated with incorrect answers. Remaining challenges—premature exit on multi-hop problems and gradient interference between exit and reasoning objectives—point to future work on decoupled training strategies and finer-grained exit supervision.

Limitations

Gradient interference between exit and reasoning objectives. Although ReMi uses a decoupled exit head (a separate MLP rather than token-level supervision), exit loss gradients still propagate through the shared hidden states back to the backbone. The backbone must simultaneously satisfy two demands: (1) encoding sufficient information in $\mathbf{h}_t$ for correct answer generation, and (2) producing a clear "done/not-done" signal for the exit

classifier. These objectives can conflict—the optimal reasoning representation is not necessarily the most discriminable for termination detection. We believe this tension is a fundamental open problem in adaptive latent reasoning. Future directions include stop-gradient strategies, adapter-based exit heads that fully decouple gradient flow, and pre-training-time integration of halting signals where the model has more capacity to reconcile both objectives.

Evaluation scope. We evaluate on GSM8K (math) and ProsQA (logic) with GPT-2 and LLaMA-1B. Whether ReMi generalizes to larger models (7B+), where richer hidden-state capacity may alleviate the gradient competition above, and to more diverse tasks (MATH, commonsense, multi-modal reasoning) remains open.

Exit granularity. The current exit head is trained with binary labels (continue/end). It struggles to distinguish "nearly complete but missing one critical step" from "fully complete"—leading to premature exit on problems requiring a final unit conversion or aggregation (§4.5). Multi-level confidence estimation or soft exit labels may provide finer-grained stopping signals.

References

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. 2024. Do NOT think that much for 2+3=? on the overthinking of o1-like llms. CoRR, abs/2412.21187.

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

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. 2025. Training large language models to reason in a continuous latent space. In COLM.

Yassir Laaouach. 2025. Halt-cot: Model-agnostic early stopping for chain-of-thought reasoning via answer entropy. ICML Workshop.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024. OpenReview.net.

Bo Lv, Yasheng Sun, Junjie Wang, and Haoxiang Shi. 2025. Onelatent: Distilling all intermediate reasoning into a single latent token. arXiv preprint arXiv:2506.03820.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel J. Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. In Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025, Suzhou, China, November 4-9, 2025, pages 20275–20321. Association for Computational Linguistics.

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. 2025. CODI: compressing chain-of-thought into continuous space via self-distillation. In Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025, Suzhou, China, November 4-9, 2025, pages 677–693. Association for Computational Linguistics.

Wenhui Tan, Jiaze Li, Jianzhong Ju, Zhenbo Luo, Jian Luan, and Ruihua Song. 2025. Think silently, think fast: Dynamic latent compression of LLM reasoning chains. CoRR, abs/2505.16552.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023. OpenReview.net.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022.

Xilin Wei, Xiaoran Liu, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Jiaqi Wang, Xipeng Qiu, and Dahua Lin. 2026. SIM-COT: Supervised implicit chain-of-thought. In International Conference on Learning Representations.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. 2022. Star: Bootstrapping reasoning with reasoning. In Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022.

Zhen Zhang, Xuehai He, Weixiang Yan, Ao Shen, Chenyang Zhao, Shuohang Wang, Yelong Shen, and Xin Eric Wang. 2025. Soft thinking: Unlocking the reasoning potential of llms in continuous concept space. Preprint, arXiv:2505.15778.

A Halting Problem Diagnostic Experiments

We provide full experimental details for the halting problem analysis in §2.2. We train a GPT-2 (124M) with SimCoT (Wei et al., 2026) on GSM8K (Cobbe et al., 2021) (max 10 latent steps, $c_{\text{thought}} = 2$, max latent stage = 5). At inference, we evaluate 1–15 latent steps on the full test set (1,319 samples), measuring answer accuracy at each step by early-exiting and greedily decoding.

A.1 Early-Exit Accuracy

Figure 1 visualizes the early-exit accuracy curve for both SimCoT and Coconut. The SimCoT model was trained at $\leq$ 10 latent tokens, yet accuracy peaks at step 13 (41.24%), beyond the training maximum, before showing mild degradation. This ceiling effect is consistent across architectures. We categorize all 1,319 test samples by their behavior across 15 latent steps:

- **Never Correct** (52.8%): 696 samples never produce the correct answer.
- **Eventually Stable** (40.9%): 540 samples converge to the correct answer and maintain it. 75% converge within $\leq$ 5 steps, 92% within $\leq$ 7 steps.
- **Mixed** (6.3%): 83 samples oscillate with no stable pattern.

A.2 AUROC of Halting Indicators

We evaluate 11 training-free halting indicators on their ability to discriminate between "should stop" and "should continue" at each latent step. We report AUROC (Area Under the Receiver Operating Characteristic curve), which measures the probability that a randomly chosen "should stop" sample scores higher than a randomly chosen "should continue" sample—0.5 indicates random chance, 1.0 indicates perfect discrimination. The oracle label is: a sample should stop at step t if its answer at step t matches its final converged answer (or the ground truth, for Never Correct samples).

Table 7: AUROC of halting indicators for latent reasoning. Output-based indicators (top) substantially outperform representation-based ones (bottom). Abbreviations: HS = hidden state, KL = KL divergence between consecutive steps.

Indicator	Mean	Best	Step
Logit entropy	0.648	0.906	14
Top-1 prob	0.639	0.898	14
Top-1/Top-2 gap	0.629	0.888	14
Logit variance	0.601	0.757	5
Answer stability	0.499	0.506	3
HS cosine sim	0.460	0.512	2
HS ℓ_2 delta	0.408	0.550	2
HS norm	0.432	0.573	3
End-latent prob	0.353	0.470	3
KL (step-to-step)	0.270	0.565	3

A.3 Token-Level Supervision at Latent Positions Harms NTP

To test whether token-level signals can serve as halting indicators at latent positions, we design a suppress-and-exit scheme that applies two auxiliary losses on the logits $\mathbf{p}_t = \text{softmax}(\mathbf{h}_t \mathbf{W}_{\text{lm}})$ at each latent position t:

- **Suppress loss** (non-final positions): penalize probability mass on termination tokens $\mathcal{C} = \{\texttt{<|end-latent|>}, \texttt{<eos>}\}$:

$$\mathcal{L}_{\text{suppress}} = -\frac{1}{|\mathcal{B}|} \sum_{t<N_i} \sum_{c \in \mathcal{C}} \log(1 - p_t(c)) \tag{11}$$

- **Exit NTP loss** (final position): force the last latent position to predict the end-latent token (<EL>):

$$\mathcal{L}_{\text{exit-ntp}} = -\frac{1}{|\mathcal{B}|} \sum_{i} \log p_{N_i}(\texttt{<EL>}) \tag{12}$$

The total loss is $\mathcal{L} = \mathcal{L}_{\text{NTP}} + \lambda_s \cdot \mathcal{L}_{\text{suppress}} + \lambda_e \cdot \mathcal{L}_{\text{exit-ntp}}$, where $\lambda_s = 0.3$ and $\lambda_e = 1.0$.

The intuition is to teach the model's own language model head to produce a clear termination signal at the final latent position, which could then serve as a training-free halting indicator at inference.

Table 8: Effect of suppress-and-exit training on answer accuracy (GPT-2, GSM8K). Adding token-level halting supervision ($\mathcal{L}_{\text{suppress}} + \mathcal{L}_{\text{exit-ntp}}$) at latent positions degrades NTP performance.

Configuration	ACC (%)	Δ
SimCoT (NTP + explain only)	39.79	—
$+ \mathcal{L}_{\text{suppress}} + \mathcal{L}_{\text{exit-ntp}}$	29.55	−10.2%

Results show that the suppress-and-exit scheme causes a **10.2 percentage point** drop in answer accuracy (29.55% vs 39.79%). The latent representations, when constrained to produce meaningful token-level signals, lose their capacity for flexible continuous reasoning—confirming the fundamental tension between latent reasoning and discrete token prediction discussed in §2.2. This negative result directly motivates ReMi's approach: a dedicated exit head that operates on hidden states without interfering with the NTP objective.

A.4 Qualitative Behavior Patterns

Early regression. At step 1, 65.4% of initially correct samples eventually converge to a wrong answer. The regression rate drops steadily: 27.5% at step 2, 5.6% at step 5, and <2% beyond step 9.

Late bloomers. 6 out of 1,319 samples (0.45%) only become correct after exceeding the training maximum of 10 latent steps. For instance, one sample outputs a consistent wrong answer (8) for steps 1–12 but produces the correct answer (25) at step 13.

Flip-flop behavior. ≈9% of samples exhibit two or more oscillations between correct and incorrect answers (e.g., correct → wrong → correct → wrong → correct). The wrong answers are typically plausible intermediate results rather than random noise, suggesting the model is navigating between competing reasoning paths.

A.5 Coconut Ceiling Verification

To confirm that the fixed-allocation problem is a general phenomenon of latent reasoning (not specific to SimCoT), we run the same early-exit analysis on a Coconut model (GPT-2, trained with $c_{\text{thought}} = 2$, max_latent_stage= 3, i.e., max 6 latent tokens).

Table 9: Coconut early-exit accuracy (GSM8K, 1,320 samples, train max=6).

Steps	ACC (%)	Note
1	8.48	< train min
2	14.92	Stage 1
4	26.59	Stage 2
6	**32.42**	**Train max**
8	33.48	+1.1%
10	33.56	Saturated
15	32.27	−1.3%

Key findings: (1) Accuracy plateaus near the training maximum (step 6), confirming that the ceiling phenomenon generalizes across latent reasoning architectures. (2) Unlike SimCoT's hard ceiling (bit-for-bit identical outputs beyond step 10), Coconut exhibits a soft ceiling with a small "grace zone" (steps 6–10) where marginal gains are possible, followed by degradation at step 15. (3) This degradation further motivates adaptive stopping: not only is fixed over-allocation wasteful, it can actively harm accuracy.

B Step Decoder Case Studies

We decode latent hidden states into readable text via the step decoder for both ReMi (adaptive exit, GPT-2, checkpoint 9) and SimCoT (fixed 10 latent steps, GPT-2, checkpoint 21) on GSM8K test samples. Below we present representative cases organized by finding category.

B.1 Overthinking in SimCoT

Fixed-allocation SimCoT assigns 10 latent steps to all problems regardless of difficulty. Our step decoder analysis reveals that effective computation concentrates in the first 3–5 positions, while later positions degenerate into repetitive patterns or introduce spurious computations.

Case A: Spurious computation causes wrong answer (Sample 3).

> **Q:** James decides to run 3 sprints 3 times a week. He runs 60 meters each sprint. How many total meters does he run a week?
> **GT:** «3*3=9» → «9*60=540» Answer: 540

ReMi (3 steps, pred=540 ✓):

> Latent 1: .75=0.75»
> Latent 2: 180180180...=180» (noisy but captures 180)
> Latent 3: .4=1.6»
> [Exit Head: STOP]

SimCoT (10 steps, pred=1080 ✗):

Latent 1: «60*3=60»
Latent 2: .6((36*36*100=3600»
Latent 3: «180=180» (correct intermediate)
Latent 4: ### 180
Latent 5: «2*180=540» (**spurious ×2 operation**)
Latent 6: ### 180
Latent 7: «540=540»
Latent 8–10: ###, «180=60», ### - - - (degenerated)

Analysis: SimCoT computes the correct intermediate 180 at position 3, but the excess computation at position 5 introduces an erroneous ×2 multiplication, yielding 1080. The model's internal state is corrupted by overthinking. ReMi avoids this entirely by exiting after 3 steps.

Case B: Degeneration causes drift (Sample 4).

Q: Wendi feeds each of her chickens three cups of mixed feed in three separate meals. She gives 15 cups in the morning and 25 cups in the afternoon. How many cups does she need in the evening if she has 20 chickens?
GT: «3*20=60» → «60-15-25=20» Answer: 20

ReMi (3 steps, pred=20 ✓):

Latent 1: «25*3=75»
Latent 2: 3*3=15»
Latent 3: «25+25=50»
[Exit Head: STOP]

SimCoT (10 steps, pred=8 ✗):

Latent 1: «20+.93=33.93»
Latent 2: «3+20+15=35»
Latent 3: «15=15»
Latent 4: −10+-8+-16=10»
Latent 5–10: ### 8, ### - -, ### 8, ... (locked onto wrong answer 8)

Analysis: SimCoT's later positions degenerate and lock onto the incorrect answer 8, overriding any partial correctness from earlier steps.

Case C: Both correct, but SimCoT wastes 70% computation (Sample 1).

Q: A robe takes 2 bolts of blue fiber and half that much white fiber. How many bolts in total?
GT: «2/2=1» → «2+1=3» Answer: 3

ReMi (3 steps, pred=3 ✓):

Latent 1: «1*2=2»
Latent 2: «2+1=2»
Latent 3: «3*1=3»
[Exit Head: STOP]

SimCoT (10 steps, pred=3 ✓):

Latent 1: «2=3»
Latent 2: .
Latent 3: «2+1=3» (effective computation ends here)
Latent 4–10: ### 6, ««3.00+2=5», ### 3, « 3, ###, 3, ### 3

Analysis: Both models arrive at the correct answer. However, SimCoT's effective reasoning finishes by position 3—positions 4–10 contribute only repetition and noise, representing 70% wasted computation.

B.2 Efficient Adaptive Exit in ReMi

When problem complexity matches ReMi's allocated steps, the exit head produces compact traces where decoded latents align well with ground-truth reasoning.

Case D: Perfect alignment with GT (Sample 6).

Q: Toulouse has twice as many sheep as Charleston. Charleston has 4× as many as Seattle. How many total if Seattle has 20?
GT: «20*4=80» → «2*80=160» → «20+160+80=260» Answer: 260

ReMi (3 steps, pred=260 ✓):

Latent 1: 8080=80» (captures intermediate 80)
Latent 2: «2*80=160» (**exact match with GT step 2**)
Latent 3: «160=160» (confirms, then EXIT)

SimCoT (10 steps, pred=260 ✓):

Latent 1: «80=80»
Latent 2: «2*20=40»
Latent 3: «160=160»
Latent 4–10: +, «140=140», ### 180, «20+80=140», ### 270, ... (degenerated)

Analysis: ReMi's Latent 2 is an exact text match with GT step 2, demonstrating that the latent space genuinely encodes structured arithmetic operations retrievable by the step decoder.

Case E: Compressing 5 GT steps into 3 latent steps (Sample 10).

Q: A program had 60 downloads month 1, 3× in month 2, then reduced by 30% in month 3. Total?
GT: «3*60=180» → «180+60=240» → «30/100*180=54» → «180-54=126» → «126+240=366» Answer: 366

ReMi (3 steps, pred=366 ✓):

Latent 1: «180*2=360» (captures 180)
Latent 2: 30-(30-(-(30-(... (noisy, but encodes 30% reduction)
Latent 3: «126*2=222» (captures 126)

SimCoT (10 steps, pred=318 ✗):

Latent 1: «180=180»
Latent 2: «180+180=270» (error: should be 180+60=240)
Latent 3–10: mixed noise, converges to wrong answer 318

Analysis: Despite the problem requiring 5 GT steps, ReMi compresses the reasoning into 3 latent steps and still arrives at the correct answer. The decoded latents show key intermediates (180, 30% reduction, 126) are encoded even though the text reconstructions are noisy. SimCoT introduces an error at position 2 ($180 + 180$ instead of $180 + 60$) and never recovers.

Case F: Step-by-step alignment (Sample 22).

> **Q:** Billy has 8 customers: first 3 buy 1 DVD each, next 2 buy 2 each, last 3 buy none. Total DVDs?
> **GT:** «3*1=3» → «2*2=4» → «3+4+0=7» Answer: 7

ReMi (3 steps, pred=7 ✓):

> Latent 1: «3*1=3» (**exact match with GT step 1**)
> Latent 2: 22222=4» (captures 4, noisy format)
> Latent 3: «4*2=8»

Analysis: Latent 1 produces a perfect reconstruction of GT step 1. Latent 2 captures the correct intermediate value 4 despite format noise. This demonstrates that step-level alignment between latent thoughts and GT reasoning is achievable.

B.3 Failure Modes of Adaptive Exit

We categorize all 4 cases where ReMi produces incorrect answers (while SimCoT answers correctly) into distinct failure types.

Type 1: Reasoning path divergence (Sample 0).

> **Q:** Janet's ducks lay 16 eggs/day. She eats 3, bakes with 4, sells rest at $2 each. Daily income?
> **GT:** «16-3-4=9» → «9*2=18» Answer: 18

ReMi (3 steps, pred=14 ✗):

> Latent 1: «4*7=28» (unrelated computation)
> Latent 2: «16-7=11» (builds on wrong intermediate)
> Latent 3: «9*4=36»

SimCoT (10 steps, pred=18 ✓):

> Latent 1: «12=12»
> Latent 5: «18=18» (correct answer appears)
> Latent 6–10: ### 18, ... (repeats correct answer)

Analysis: ReMi's failure here is not due to premature exit—the reasoning path itself diverges from step 1. The 4*7=28 computation has no grounding in the problem. This is a latent reasoning quality issue unrelated to the stopping mechanism.

Type 2: Reverse reasoning difficulty (Sample 13).

> **Q:** Melanie sold $\frac{1}{3}$ at green house, 2 more to red house, half of remainder at orange house. If 5 left, how many started?
> **GT:** «5*2=10» → «10+2=12» Answer: 18

ReMi (3 steps, pred=30 ✗):

> Latent 1: .33=1.67» (attempts $\frac{1}{3}$ but fails)
> Latent 2: «10+5=15» (partial: captures 10)
> Latent 3: «5+2=7»

SimCoT (10 steps, pred=18 ✓):

> Latent 1: «2+5=10» (reverse: from remainder back)
> Latent 6: ### 18 (converges to correct answer)
> Latent 9: «5*2=10»

Analysis: Reverse reasoning (working backwards from a final state) requires multi-step logical unfolding. ReMi's Latent 1 attempts the $\frac{1}{3}$ relationship but fails to establish the correct backward chain within 3 steps. SimCoT's additional capacity allows the model to "explore" multiple paths until converging on the answer.

Type 3: Insufficient steps for multi-hop problems (Sample 16).

> **Q:** Two trains travel 80 miles west and 150 miles north. Distance covered by each in two days?
> **GT:** «2*80=160» → «150*2=300» → «300+160=460» → «460/2=230» Answer: 230

ReMi (3 steps, pred=115 ✗):

> Latent 1: «230*0.5=115» (attempts shortcut: answer/2)
> Latent 2: «230/4=115» (stuck on 115)
> Latent 3: «220-15=185»

SimCoT (10 steps, pred=230 ✓):

> Latent 5: «230=230» (correct answer appears at step 5)

Analysis: The problem requires 4 GT steps but ReMi exits after 3. Interestingly, the model appears to "know" the answer should relate to 230 (it appears in latents 1–2), but lacks sufficient depth to derive it properly and instead produces $230/2 = 115$. This is a clear capacity limitation: 3 steps cannot support a 4-step reasoning chain.

Type 4: Missing final conversion (Sample 18).

> **Q:** Claire makes a 3-egg omelet every morning. How many **dozens** of eggs in 4 weeks?
> **GT:** «3*7=21» → «4*21=84» → «84/12=7» Answer: 7

ReMi (3 steps, pred=1.75 ✗):

Latent 1: `.2=.4»`
Latent 2: `«4*21=84»` (**exact match with GT step 2**)
Latent 3: `«4*7=28»` (does NOT perform ÷12)

SimCoT (10 steps, pred=7 ✓):

Latent 1: `«21=21»`
Latent 3: `«84=84»`
Latent 8: `### 7` (converges to correct answer)

Analysis: ReMi successfully computes $4 \times 21 = 84$ (matching GT step 2 exactly), but exits before performing the critical unit conversion $84 \div 12 = 7$. The exit head incorrectly judges reasoning as complete when a key final transformation remains. This "last-mile" failure is the most actionable finding: it suggests exit head training should incorporate harder negatives at penultimate positions where intermediate values are available but the final answer requires one more transformation.

C Implementation Details

We build on the Coconut (Hao et al., 2025) and SimCoT (Wei et al., 2026) codebase. All models are trained on a single node with NVIDIA A100 GPUs.

Model and optimization. The backbone is GPT-2 (124M) or LLaMA-1B. We use AdamW with a constant learning rate of 2.5×10^{-4} for GPT-2 and 2×10^{-5} for LLaMA (no warmup or decay). For training efficiency, we use a larger batch size and learning rate compared to the original Coconut/SimCoT settings; all baselines are retrained under our configuration for fair comparison.

Exit head. A two-layer MLP ($d \to d/4 \to 2$) with GELU activation, where d is the backbone hidden dimension. Exit loss weight $\lambda_{\text{exit}} = 0.5$.

Step decoder. A separate GPT-2 model (original vocabulary size 50,257) that shares the backbone's tokenizer. It takes the latent hidden state $\mathbf{h}_t$ as a conditioning prefix and is trained with next-token prediction on ground-truth reasoning steps. Step reconstruction loss weight $\lambda_{\text{step}} = 1.0$.

Progressive training schedule. We use $c_{\text{thought}} = 2$ latent tokens per reasoning step, epochs_per_stage $= 3$, and max_latent_stage $= 5$, yielding a maximum of $5 \times 2 = 10$ latent tokens at the final stageance. Stage 0 trains with full explicit CoT (no latent tokens).

Inference. Exit threshold $\tau = 0.5$ with safety bound $T_{\max} = 10$. Decoding is greedy (temperature $= 0$). The step decoder is invoked post-hoc for interpretability analysis and does not affect answer generation.