LATENT CHAIN-OF-THOUGHT AS PLANNING: DECOUPLING REASONING FROM VERBALIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Chain-of-Thought (CoT) empowers Large Language Models (LLMs) to tackle complex problems, but remains constrained by the computational cost and early token commitments in discrete reasoning traces. Recent latent reasoning approaches attempt to optimize efficiency by performing reasoning within continuous hidden states. However, many such methods optimize latent states end to end without a trained interface for intermediate textual readout, and several representative configurations use a pre-defined number of latent steps during inference. In this work, we introduce **PLaT** (**P**lanning with **La**tent **T**houghts), a framework that decouples latent planning from verbalization. The Planner deterministically evolves latent planning states, while an independent Decoder provides textual readouts when needed. Answer-aware textual stopping allows the latent rollout to use a problem-dependent number of groups rather than a pre-specified chain length. PLaT achieves competitive coverage at larger k in several mathematical settings, with lower Pass@1: on Llama-1B GSM8K, it reaches 80.59% Pass@128 versus CODI's 72.37%. These results support PLaT as a candidate-generation interface supplying multiple textual readouts for downstream verification or reranking.

1 INTRODUCTION

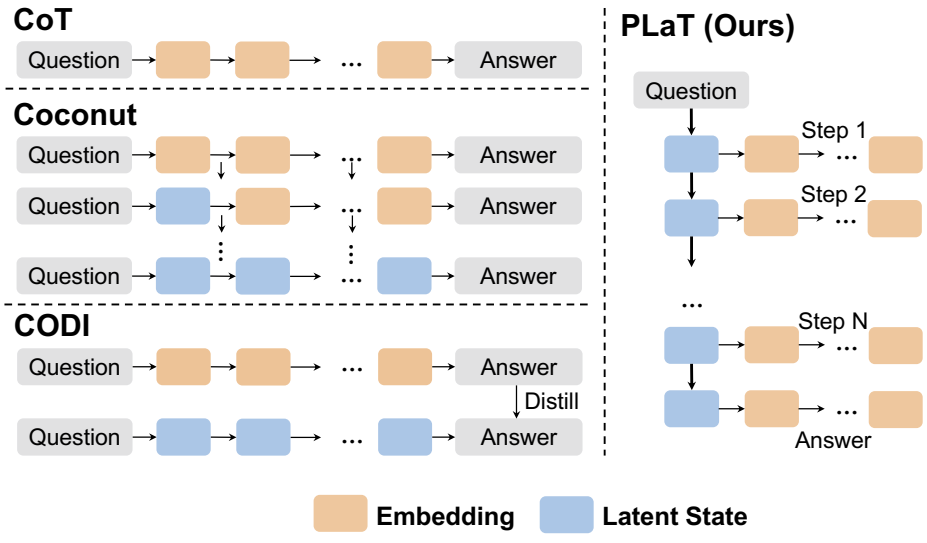


Figure 1: **Comparison of PLaT and other reasoning strategies.** Orange blocks denote token embeddings and blue blocks denote continuous latent states. CoT generates explicit reasoning tokens; Coconut progressively replaces textual reasoning with latent states during training; CODI distills explicit reasoning into a latent chain. PLaT separates latent-state evolution from textual readout: the vertical chain represents planning, and each horizontal branch represents a decoded reasoning segment or answer. Decoded text is not fed back into the Planner.

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

Chain-of-Thought (CoT) reasoning Wei et al. (2022); Kojima et al. (2022); Wang et al. (2023); Zhang et al. (2023) improves the reasoning performance of Large Language Models (LLMs) by decomposing problems into sequences of intermediate steps Zhou et al. (2023). This approach supports applications ranging from code generation Chen et al. (2021; 2023); Li et al. (2025a); Liu et al. (2025) to autonomous agents Yao et al. (2022); Shinn et al. (2023); Schick et al. (2023). However, each sampled trajectory commits to discrete textual choices. At every generation step, a sampled trajectory selects one discrete token, while alternative valid continuations remain available only across separate samples Yao et al. (2023a); Zhang et al. (2025); Chen et al. (2025). This can make recovery from an incorrect intermediate choice difficult within a single sampled trajectory. Additionally, generating long sequences of intermediate tokens incurs substantial inference cost Zhang et al. (2025); Sui et al. (2025); Wang et al. (2025); Feng et al. (2025).

To mitigate the inefficiency, recent works have explored latent reasoning, where the model evolves hidden states internally before outputting a final answer Zhang et al. (2025); Xu et al. (2025); Hao et al. (2025); Shen et al. (2025); Tan et al. (2025). While promising, they predominantly adopt an end-to-end implicit paradigm, optimizing latent states directly for the final generation. This leaves two practical gaps that motivate our design. First, intermediate states in many latent reasoning methods lack a trained interface for textual inspection. Second, several prominent methods, including the Coconut and CODI configurations evaluated here, use a prespecified number of latent steps during inference. This fixes their latent rollout budget regardless of problem difficulty, motivating an architecture that permits a problem-dependent chain length Li et al. (2025b).

We argue that a robust reasoning system should mirror the cognitive distinction between thought and language. From a cognitive perspective, language serves merely as a low-dimensional projection (interface) of high-dimensional thought; the core reasoning process often occurs implicitly without verbalization Varley & Siegal (2000); Fedorenko & Varley (2016); Coetzee et al. (2022); Fedorenko et al. (2024). Ideally, a continuous state can support multiple possible textual continuations until an interface with the external world is required. In PLaT, this is expressed as a distribution over Decoder readouts; the deterministic Planner itself does not sample multiple latent trajectories. Motivated to replicate this implicit process computationally, we draw inspiration from Multi-Token Prediction (MTP) Stern et al. (2018); Qi et al. (2020); Gloeckle et al. (2024); Nagarajan et al. (2025); Samragh et al. (2025) (that transformer hidden states are able to encode information about future tokens before they are generated) and propose to model reasoning as a sequence of latent planning states.

In this work, we introduce **PLaT** (**P**lanning with **La**tent **T**houghts), a framework that decouples the reasoning process from verbalization. As illustrated in Figure 1, our architecture comprises two distinct components: a latent Planner and a Decoder for verbalization. The Planner autoregressively evolves a trajectory of continuous states, while the Decoder provides text readouts when needed. The Decoder grounds these latent plans onto the language space via a reconstruction objective. This enables answer-aware adaptive termination and makes intermediate planning states available for textual readout, with a problem-dependent number of latent groups. Our empirical evaluations on mathematical benchmarks reveal a distinctive behavioral pattern. We identify a trade-off between single-sample precision and sampled coverage: PLaT-8 has lower Pass@1 but competitive or stronger Pass@k at larger k relative to the compared latent methods in several settings. Pass@k measures whether a correct candidate is present in a sampled pool, making it relevant to subsequent verification, reranking, voting, or search. Realizing this coverage as final-answer accuracy requires effective candidate selection. Furthermore, the latent states in PLaT can be decoded into text for intermediate inspection without disrupting the continuous reasoning flow. Our main contributions are summarized as follows:

- We formulate latent reasoning as an autoregressive sequence of continuous planning states with a textual Decoder that has separate parameters and is jointly optimized with the Planner, keeping generated text outside the Planner recurrence.
- We enable problem-dependent latent chain lengths through answer-aware stopping, while retaining textual access to intermediate planning states throughout the rollout.
- Experiments reveal a precision–coverage trade-off: PLaT can provide useful candidate coverage at larger k in several settings, accompanied by lower Pass@1, motivating future verifier- and search-based inference.

2 Related Work

2.1 Chain-of-thought Reasoning

Chain-of-Thought (CoT) prompting enables large language models to solve complex problems by explicitly decomposing them into intermediate reasoning steps Wei et al. (2022). Subsequent work shows that CoT can be activated with minimal prompting by a single instruction such as "Let's think step by step" Kojima et al. (2022). Beyond simple CoT strategies, self-consistency Wang et al. (2023) samples multiple CoT trajectories and marginalizes over final answers, improving robustness by approximating a distribution over reasoning paths. Tree-of-Thought (ToT) Yao et al. (2023a); Long (2023); Mo & Xin (2024) generalizes CoT into an explicit tree search, allowing the model to branch, evaluate, and backtrack over intermediate states. Graph-of-Thought (GoT) Yao et al. (2023b); Besta et al. (2024); Yao et al. (2024) further extends this paradigm to graph-structured reasoning, enabling the reuse and recombination of intermediate conclusions across different branches. Orthogonal to structured search, other works focus on refining intermediate reasoning steps through iterative revision or verification. These approaches improve correctness by critiquing, editing, or validating generated reasoning traces, often in a multi-pass manner Madaan et al. (2023); Lyu et al. (2023); Yang et al. (2023).

2.2 Latent Reasoning

Recent latent reasoning methods aim to reduce the cost and path collapse of explicit CoT by shifting part of the reasoning process into continuous hidden states, while retaining the final answer in natural language. Early approaches explore partial internalization of reasoning, where additional non-semantic structures are introduced during inference. For example, pause tokens or planning tokens allow models to suspend surface-level generation and perform internal processing before producing the next reasoning step Goyal et al. (2024); Wang et al. (2024). A second line of work focuses on compressing explicit CoT into latent representations, progressively removing or softening intermediate textual steps. Curriculum-based approaches such as Coconut gradually replace explicit reasoning tokens with continuous latent states, enabling the model to internalize multi-step reasoning Hao et al. (2025). Related techniques distill explicit CoT trajectories into latent spaces or employ continuous vectors for intermediate reasoning steps Zhang et al. (2025); Shen et al. (2025). Other approaches further explore latent compression objectives to stabilize and optimize implicit reasoning processes Tan et al. (2025). These approaches seek to reduce explicit reasoning generation, typically treating latent states as end-to-end optimized carriers of reasoning with limited interfaces for inspecting intermediate states.

3 Method

In this section, we first formalize reasoning as a latent autoregressive process. We then detail the PLaT architecture, the training via reconstruction, and inference and termination. All notations are listed in Appendix A.

3.1 Problem Formulation: Reasoning as Latent Planning

Standard CoT models the probability of a reasoning chain $y = (y_1, \ldots, y_T)$ autoregressively in the discrete token space:

$$p(y|x) = \prod_{k=1}^{T} p(y_k|y_{<k}, x) \tag{1}$$

Sampling from this formulation commits each trajectory to a specific token at each step, although other continuations remain available across samples.

PLaT introduces a sequence of continuous latent variables corresponding to textual segments. We tokenize the reasoning text and divide it into N segments of at most K tokens; segment boundaries need not coincide with complete equations. Let $\mathbf{S}_k = (\mathbf{s}_{k,1}, \ldots, \mathbf{s}_{k,L})$ denote the L latent vectors assigned to segment y_k. The Planner deterministically generates these vectors from the question and preceding latent vectors, without feeding decoded text back into the Planner. The latent history is

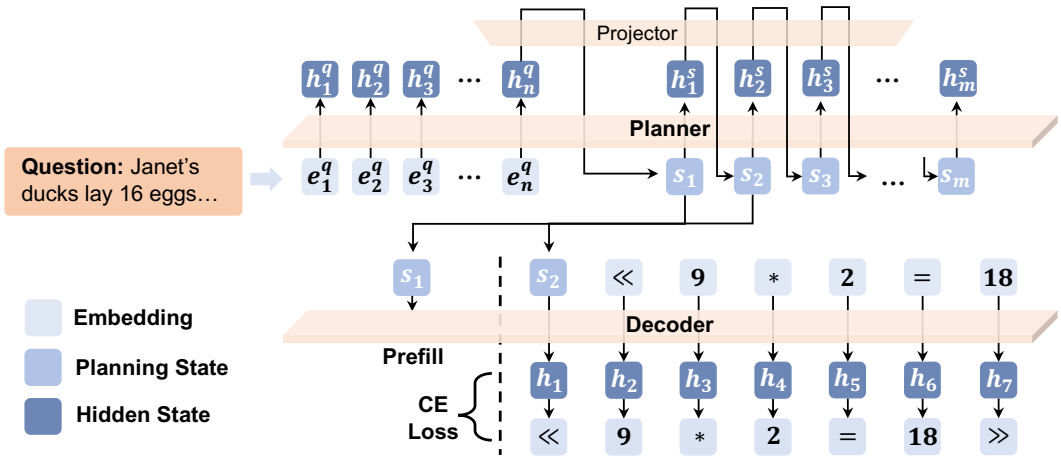


Figure 2: **Framework of the proposed PLaT paradigm.** The Planner autoregressively generates latent states conditioned on the question, and an independent Decoder uses each latent group as a soft prefix to generate a text segment. The illustration shows a general $L = 2$ example, with two latent vectors per decoded segment; the main experimental configuration uses $L = 1$.

maintained autoregressively by the Planner. For a given segmentation into N groups, the Decoder readout likelihood factorizes as follows:

$$p(y|x) = \prod_{k=1}^{N} p_{\mathcal{D}}(y_k|\mathbf{S}_k(x)). \tag{2}$$

The main configuration uses $K = 8$ and $L = 1$; the number of latent groups N is determined dynamically during inference.

3.2 Architecture

As shown in Figure 2, PLaT implements the above formulation through two distinct modules: the Planner and the Decoder. The Planner $\mathcal{M}$ and independent Decoder $\mathcal{D}$ have separate parameters. A latent-generation projection ϕ maps Planner hidden states back to the backbone embedding dimension d_m. It consists of a linear layer to width d_p, GELU, a linear layer back to d_m, and LayerNorm; projection dropout is zero in the verified configurations.

Planner. The Planner is responsible for evolving the reasoning trajectory autoregressively on the latent manifold. The final hidden state of the question initializes the first latent vector, without appending an additional start token:

$$\mathbf{s}_{1,1} = \phi(\mathcal{M}(x)_{-1}) . \tag{3}$$

Each subsequent latent vector is generated by feeding the previous vector as an input embedding while retaining the question and latent-history key-value cache:

$$\mathbf{s}_{k,i} = \phi(\mathcal{M}([x; \mathbf{S}_{<k}; \mathbf{s}_{k,<i}])_{-1}) , \qquad (k, i) \neq (1, 1). \tag{4}$$

It is important to note that the Planner generates deterministic vectors for a fixed question. Sampling diversity arises at the Decoder: each latent group defines a distribution over possible textual continuations.

Decoder. The Decoder maps each latent group to a distribution over textual segments. It takes the current latent group $\mathbf{S}_k$ directly as a soft prefix for generating the text segment y_k:

$$P(y_k|\mathbf{S}_k) = \prod_{j=1}^{|y_k|} P_{\mathcal{D}}(y_{k,j}|\mathbf{S}_k, y_{k,<j}). \tag{5}$$

The main configuration uses no EMA aggregation. The Decoder conditions only on the current latent group and its own within-segment tokens; the question and earlier decoded segments are not supplied as additional context. Decoder-context variants are evaluated separately in the ablation study.

3.3 Supervised Training via Reconstruction

During Supervised Fine-Tuning (SFT), we optimize the entire pipeline end-to-end using a latent-to-text reconstruction loss. The loss is calculated as the cross-entropy between the ground-truth text y_k and the model's prediction conditioned on the state $\mathbf{S}_k$:

$$\mathcal{L}_{\text{rec}} = -\sum_{k=1}^{N}\sum_{j=1}^{|y_k|}\log P(y_{k,j}|\mathbf{S}_k, y_{k,<j}). \tag{6}$$

This formulation treats intermediate reasoning steps and the final answer uniformly within the latent space, eliminating the need for mode-switching mechanisms and the constraint of a fixed number of latent steps. No latent-history noise is injected in the main configuration. The Planner LoRA parameters, latent projection, and independent Decoder are optimized through the reconstruction objective.

3.4 Inference and Termination

Since the Planner operates in the latent space, its trajectory can advance without feeding generated text back into the Planner. Each latent group is decoded for up to K tokens; K controls the local text budget rather than the total number of latent groups. We detect the tokenized prefix `The answer is:` in the accumulated output, allowing the prefix to span segment boundaries. After that prefix is detected, generation stops when EOS occurs in the answer content, or when the configured rollout limit is reached. This is an answer-aware rollout mechanism rather than a learned latent halting head. The number of latent groups is therefore not fixed by K.

4 Experiments

4.1 Experimental Setup.

Model. For the GSM8K-trained comparison, we use GPT-2 Radford et al. (2019) and the instruction-tuned Llama-3.2-1B and Llama-3.2-3B models Grattafiori et al. (2024). For the separate OpenR1-Math setting, all methods use only Llama-3.2-1B-Instruct. This provides a focused comparison of longer-form reasoning at a fixed backbone size while keeping long-sequence training and 128-sample evaluation computationally manageable. The OpenR1 experiments therefore assess performance on longer-form tasks at the 1B scale; the GSM8K experiments provide the comparison across backbone sizes.

Datasets. We train on **GSM8K-Aug** Deng et al. (2023), which augments **GSM8K** Cobbe et al. (2021) with equation-formatted reasoning generated by GPT-4 Achiam et al. (2023). GSM8K contains multi-step grade-school mathematics problems; we evaluate on its 1,319 test questions. To assess transfer beyond the training distribution, we also use **GSM-Hard** Gao et al. (2022), which increases numerical difficulty, **SVAMP** Patel et al. (2021), which tests robustness to variations in arithmetic word problems, and **MultiArith** Roy & Roth (2015), which requires combining multiple arithmetic operations.

For longer-form reasoning, we train separately on **OpenR1-Math-220k** Hugging Face (2025), whose extended mathematical solutions provide a less regularly segmented form of supervision. We evaluate on **MATH-500** Hendrycks et al. (2021), comprising 500 competition-level problems, and **AIME 2024** and **AIME 2025** AIME (2025), with 30 competition problems each. This setting tests whether the approach extends beyond short equation-formatted solutions and is reported separately from the GSM8K-trained comparison. Data preparation and grading details appear in Appendix B.3.

Baselines. We benchmark PLaT against three representative baselines: (1) **CoT-SFT** Wei et al. (2022): Fine-tunes the model with reasoning chains, and the trained models generate step-by-step during inference. (2) **Coconut** Hao et al. (2025): A curriculum-based latent reasoning method that progressively replaces explicit tokens with implicit hidden states. (3) **CODI** Shen et al. (2025): A latent reasoning framework that employs hidden-state distillation from an explicit reasoning teacher. All tables identify the backbone and use the corresponding fixed trained model. The Llama-1B PLaT

Table 1: **Main results on the GSM8K-trained models (%).** Pass@k is computed from the first k of 128 samples within each run, then averaged over the three runs. Standard deviations are given in Appendix B.5. Gray CoT-SFT rows are excluded from the latent-method ranking. Bold and underlined entries mark the highest and second-highest displayed latent-method scores per column, including ties. PLaT is evaluated with $K = 8$ (PLaT-8).

Model	Method	GSM8K				GSM-Hard				SVAMP				MultiArith			
		@1	@32	@64	@128	@1	@32	@64	@128	@1	@32	@64	@128	@1	@32	@64	@128
GPT-2	CoT-SFT	35.30	74.20	80.41	86.33	7.10	18.35	20.65	22.52	36.63	65.37	70.50	75.47	85.00	99.07	99.44	99.63
	Coconut	32.94	55.69	61.45	66.19	7.62	13.65	15.01	16.15	35.25	50.15	53.05	56.45	81.67	90.83	91.94	93.33
	CODI	**40.16**	**63.00**	67.12	69.90	**8.74**	**14.58**	15.52	16.53	**38.60**	53.67	55.80	58.13	**92.04**	**96.67**	**96.67**	**96.67**
	PLaT	18.35	59.51	**67.74**	**76.08**	3.75	14.22	**17.06**	**18.88**	21.80	**53.70**	**60.75**	**66.45**	33.61	79.44	87.50	91.94
Llama-1B	CoT-SFT	49.96	86.62	90.45	93.18	11.45	23.68	25.50	27.72	59.50	87.97	91.07	93.37	94.07	100.00	100.00	100.00
	Coconut	27.52	49.28	53.68	58.07	6.47	11.95	13.17	14.18	41.87	53.13	54.90	56.97	59.07	85.93	87.41	89.44
	CODI	**54.55**	**69.11**	71.08	72.37	**12.24**	**16.41**	17.13	17.55	**58.75**	**70.70**	**72.15**	73.00	**97.22**	**98.33**	**98.33**	98.33
	PLaT	36.20	68.73	**75.28**	**80.59**	8.26	16.30	**18.23**	**19.75**	43.00	66.87	70.63	**74.47**	84.63	96.85	97.41	**98.70**
Llama-3B	CoT-SFT	58.88	92.90	95.27	96.71	18.95	38.34	41.37	44.12	74.67	95.50	96.90	97.83	97.78	100.00	100.00	100.00
	Coconut	28.38	62.32	70.18	75.87	6.09	15.62	17.29	19.13	34.07	57.47	61.73	65.43	89.26	97.04	97.22	97.96
	CODI	**57.16**	**76.72**	**80.44**	83.24	**13.38**	**18.12**	**19.45**	**20.89**	**68.70**	**80.65**	**82.60**	**84.60**	**99.72**	**100.00**	**100.00**	**100.00**
	PLaT	39.85	73.82	80.09	**84.79**	9.48	17.66	19.23	20.87	54.77	76.00	79.43	82.87	95.37	98.89	99.63	99.63

row uses the PLaT-8 configuration. Coconut Llama-1B uses a stabilized learning rate of 3×10^{-5} rather than the original 10^{-4} setting. CODI Llama-3B uses the best available model supported by validation results; the optimum over the complete training history could not be verified.

Evaluation. We report **Pass@k** ($k = 1, 32, 64, 128$), the fraction of questions with at least one correct answer among the first k of 128 sampled responses. Short-form evaluations use temperature 0.9 and a 256-token generation cap. We average over available sampling runs and report their standard deviations in Appendix B.5; full evaluation settings appear in Appendix B.3. Pass@k measures candidate coverage; realizing this coverage as final-answer accuracy requires answer selection.

Implementation Details. Unless otherwise stated, PLaT training uses a learning rate of 8×10^{-4} with BF16 computation, LoRA adaptation Hu et al. (2022) (rank 128, $\alpha = 32$) on the Planner, and a full-parameter Decoder with separate parameters, jointly optimized with the Planner through the reconstruction objective. We denote the variant with K tokens per segment by PLaT-K; the suffix specifies segment length, not the number of latent vectors L or the total reasoning length. The main configuration, PLaT-8, uses one latent vector per eight-token segment ($L = 1$). Additional settings are provided in Appendix B.

4.2 Effectiveness

Main results. Among the compared latent methods, PLaT's main advantage is broader candidate coverage under repeated sampling, accompanied by lower single-sample accuracy (Table 1). On GSM8K, PLaT-8 reaches 80.59% Pass@128 with Llama-1B versus CODI's 72.37%, despite lower Pass@1 (36.20% versus 54.55%). The Llama-3B comparison shows the same trade-off, with a smaller coverage advantage. This pattern makes PLaT relevant when a system can examine several

Table 2: **Efficiency comparison.** The PLaT column reports lazy, greedy inference with the GPT-2 backbone on 200 GSM8K test questions. Forward counts include all backbone calls during planning and readout; times are per-question means $\pm$ across-question standard deviations. All methods are measured on the same hardware.

	CoT	Coconut	CODI	PLaT
Fwd.	25.55	6.00	6.00	12.83
Time (ms)	$349.6_{\pm 8.9}$	$100.6_{\pm 3.2}$	$240.0_{\pm 17.2}$	$237.5_{\pm 59.2}$

candidates, but does not establish an advantage when only one answer is generated. The benefit is also task-dependent: on Llama-3B SVAMP, CODI retains higher coverage (84.60% versus 82.87%). CoT-SFT achieves higher Pass@1 and Pass@128 than PLaT across all backbones and datasets in Table 1; the coverage gains therefore establish an advantage over particular latent baselines, not over explicit CoT.

Sampling behavior. The gap between Pass@1 and Pass@k indicates that correct answers often appear outside the first sampled response. In PLaT, these alternatives arise from repeated Decoder readouts of a deterministic latent trajectory. The results therefore motivate using this readout distribution to supply candidates for verification or search. The gains from additional samples make this distinction concrete. On Llama-1B GSM8K, increasing the pool from 32 to 128 responses raises PLaT's coverage from 68.73% to 80.59%, a gain of 11.86 percentage points, compared with 3.26 points for CODI (69.11% to 72.37%). PLaT is slightly behind at 32 samples and ahead at 64. Thus, within this sampling range, additional readouts continue to solve questions missed by the smaller pool. This is a benefit in candidate availability; converting it into final-answer accuracy still requires effective selection. The OOD results further indicate that the usefulness of this candidate pool depends on the task and backbone. Per-cell uncertainty is reported in Appendix B.5.

4.3 Efficiency

Table 2 shows that selective verbalization reduces GPT-2 latency relative to CoT on the same hardware: 237.5 versus 349.6 ms per question, a 32.1% reduction. Lazy readout skips intermediate text after a failed answer-prefix probe, while retaining Decoder calls for answer detection and generation. The textual interface thus remains available without fully verbalizing each intermediate state.

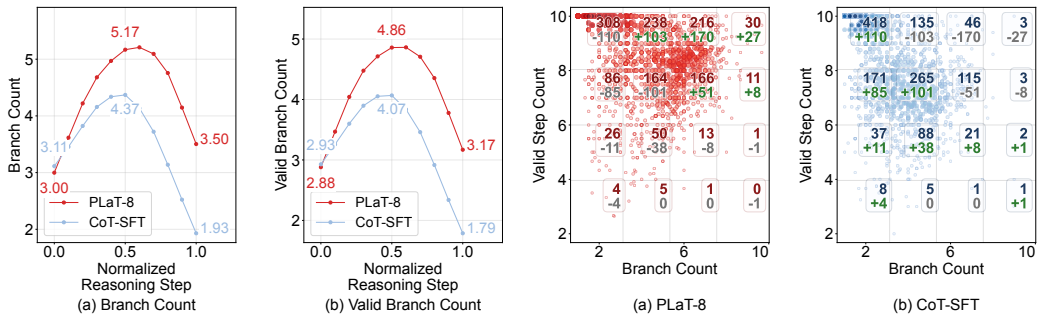
PLaT is comparable in measured latency to CODI (240.0 ms), but slower than Coconut (100.6 ms). Its 12.83 backbone calls fall between CoT's 25.55 and the latent baselines' 6.00, illustrating that call counts alone do not determine latency across architectures. These greedy measurements characterize selective readout cost; multi-sample efficiency remains to be evaluated.

4.4 Analysis of Latent States

Evolution of Reasoning Diversity. A deterministic planning state can support several distinct textual readouts. On all 1,319 GSM8K test questions, we sample 10 readouts per trajectory position and compare their diversity with explicit CoT. PLaT averages 4.648 distinct chunks and 4.347 reference-compatible chunks per nonempty state group. The proximity of these means suggests that much of the observed textual diversity remains compatible with the reference reasoning under the judgment protocol, rather than consisting only of invalid fragments.

Figure 3 shows that PLaT's additional readout diversity emerges as reasoning progresses. Both methods begin with similar counts, but PLaT retains more alternatives through the middle and later positions. At the final normalized position, PLaT still exposes 3.50 distinct readouts versus 1.93 for CoT-SFT; the corresponding valid-readout counts are 3.17 and 1.79. Thus, the separation persists after validity filtering and extends to the end of the trajectory. The scatter plots complement this aggregate trend: PLaT has more questions in regions combining higher branch counts with high valid-step counts. Together, these patterns suggest that a fixed Planner trajectory can sustain multiple locally compatible textual continuations across successive positions. This provides a readout-level perspective on the candidate diversity observed in the Pass@k results, without establishing distinct solution strategies or a causal explanation of the coverage gains. Counting and judgment details appear in Appendix B.2.

Distribution of Exploration vs. Exploitation. Figure 3b shows how readout diversity and local validity coexist across questions. CoT-SFT concentrates more questions in the upper-left region, where many sampled chunks are valid but yield relatively few distinct continuations. PLaT shifts more of this mass toward the upper-right: the two rightmost cells in the highest-validity row contain 246 questions for PLaT, compared with 49 for CoT-SFT. Conversely, the leftmost cell in that row contains 308 PLaT questions versus 418 for CoT-SFT. The contrast shows that high local validity can accompany a broader range of textual readouts from PLaT's fixed planning trajectory. The additional diversity is therefore visible across questions with consistently valid local readouts, supporting the



(a) Evolution of exploration during reasoning.

(b) Branching factor versus valid step count.

Figure 3: **Textual readout branching from latent states.** Results use PLaT and CoT-SFT on 1,319 questions with 10 sampled paths per question. For PLaT, the Planner trajectory is fixed for a question; the reported branches are distinct textual continuations sampled from intermediate latent states. In (a), curves show distinct and valid readout counts after normalizing each question's state sequence to 11 positions. In (b), samples without intermediate steps are excluded; the 16 zones annotate counts and between-method differences.

Table 3: **PLaT architectural and segmentation variants on Llama-1B GSM8K (%).** Entries report mean ± sample standard deviation over multiple verified sampling runs. Decoder-context variants use PLaT-8 with $L = 1$; fixed-K and latent-count sweeps are reported separately.

	Pass@1	Pass@32	Pass@64	Pass@128
PLaT-8	36.20 ± 0.38	68.73 ± 0.05	75.28 ± 0.00	80.59 ± 0.32
PLaT semantic-step	37.86 ± 1.10	69.45 ± 0.46	75.16 ± 0.62	80.82 ± 0.62
CoT initialization	29.95 ± 0.43	67.48 ± 0.86	74.15 ± 0.43	78.01 ± 0.11
Decoder latent history	38.51 ± 0.11	67.29 ± 0.70	73.62 ± 0.21	78.85 ± 0.32
Decoder question + latent history	31.61 ± 0.95	67.73 ± 0.99	74.37 ± 0.23	80.29 ± 0.82

use of the Decoder as an interface for generating alternative continuations. These are local textual alternatives, not evidence that each continuation forms a distinct, correct complete solution. The judgment protocol and human-review results are provided in Appendices B.2 and B.7.

4.5 Ablation Study

Table 3 supports fixed-token segmentation as a practical alternative to semantic-step supervision: their Pass@128 values are close on GSM8K, although the training differences preclude attributing the comparison solely to segmentation. Our choice of fixed-token groups is motivated by portability. Equation-formatted GSM8K provides clear step boundaries, whereas longer solutions such as OpenR1-Math do not provide uniform step annotations. Controlling segment length through K allows the same interface to extend to such data without a task-specific semantic splitter.

Additional Decoder context changes the precision–coverage trade-off rather than improving both metrics consistently. Latent history raises Pass@1 but lowers Pass@128; adding the question as well does not improve either metric over PLaT-8. Both variants preserve decoupling because generated text remains outside the Planner recurrence. These results therefore do not establish that local conditioning alone explains PLaT's lower Pass@1. The CoT-initialized variant also falls below PLaT-8, but other training differences prevent isolating initialization as the cause. Full results across datasets appear in Appendix B.6.

4.6 Analysis of Hyperparameters

We analyze token-group size K and latent vectors per segment L separately from the architectural ablations. Here, K sets the text span assigned to each latent group, while L sets the number of vectors representing that span. Neither fixes the total number of groups N, which can vary across questions

under answer-aware stopping. These sweeps therefore examine local representation granularity, separately from the ability to extend the overall latent chain. The four-dataset tables in Appendix B.6 aggregate three verified sampling runs per configuration and report sample standard deviations. The sensitivity plots retain their original single-run visualizations and are intended to show trends rather than establish monotonic effects.

Token-group size (K). With one latent vector per segment, we compare $K \in \{2, 5, 8, 16, 32\}$. Across datasets, the intermediate settings $K = 5$ or $K = 8$ generally retain more sampled coverage than the extremes; the results are shown in Appendix Fig. 4.

Latent vectors per segment (L). Increasing the number of latent vectors per segment does not consistently improve coverage. At $K = 8$, the $L \in \{1, 2, 3\}$ sweep yields dataset-dependent changes that are smaller than those from changing K on several tasks (Appendix Fig. 5). These results do not support treating additional latent vectors as a uniformly beneficial increase in capacity.

4.7 Longer-form Mathematical Evaluation

Table 4: OpenR1-Math-trained Llama-1B results on longer-form benchmarks (Pass@1 and Pass@128, %). Each dataset reports the two prefix-coverage metrics separately. All entries report means ± sample standard deviations over three complete sampling runs. Gray CoT-SFT is excluded from ranking; bold and underlined scores denote the best and second-best latent-method results in each column, with ties sharing a mark. This table is a separate evaluation setting and is not directly comparable with the GSM8K-trained main table.

Method	MATH-500		AIME 2024		AIME 2025	
	Pass@1	Pass@128	Pass@1	Pass@128	Pass@1	Pass@128
CoT-SFT	16.20 ± 0.28	77.70 ± 0.99	1.67 ± 2.36	15.00 ± 2.36	1.67 ± 2.36	20.00 ± 4.71
Coconut	$\underline{5.20} \pm 0.85$	27.60 ± 0.57	$\underline{0.00} \pm 0.00$	3.33 ± 0.00	$\underline{0.00} \pm 0.00$	$\underline{10.00} \pm 0.00$
CODI	$\mathbf{7.70} \pm 0.71$	$\underline{38.70} \pm 0.71$	$\mathbf{1.67} \pm 2.36$	$\mathbf{18.33} \pm 2.36$	$\underline{0.00} \pm 0.00$	3.33 ± 4.71
PLaT-16	2.50 ± 0.14	$\mathbf{44.80} \pm 2.55$	$\underline{0.00} \pm 0.00$	$\underline{11.67} \pm 7.07$	$\mathbf{1.67} \pm 2.36$	$\mathbf{16.67} \pm 0.00$
PLaT-32	0.80 ± 0.00	35.60 ± 1.13	$\underline{0.00} \pm 0.00$	6.67 ± 0.00	$\underline{0.00} \pm 0.00$	8.33 ± 7.07

The longer-form results retain the coverage–precision trade-off but show its limits. On MATH-500, PLaT-16 has the highest Pass@128 among the evaluated latent methods (44.80%), while its Pass@1 remains below CODI and Coconut. CoT-SFT is substantially stronger on both metrics, so this comparison does not establish parity with explicit reasoning. Increasing segment length to PLaT-32 lowers coverage on all three benchmarks. The AIME results are mixed and based on only 30 questions per year, providing limited evidence for broader generalization to competition mathematics.

5 Conclusion

PLaT separates the autoregressive evolution of continuous planning states from textual readout through an independent Decoder. Generated text remains outside the Planner recurrence, and answer-aware stopping allows the number of latent groups to vary without fixing the total chain length through the local segment configuration.

The experiments identify a precision–coverage trade-off: PLaT provides stronger candidate coverage at larger k than the compared latent methods in several settings, while baseline methods often achieve higher Pass@1 and CoT-SFT remains stronger overall. This pattern motivates PLaT's potential role as a candidate generator, where additional readouts can expose correct answers missed by smaller sample pools.

The independent readout interface enables the study of planning-state content and textual diversity along a latent trajectory. Establishing causal faithfulness and translating candidate coverage into improved answer selection remain directions for further evaluation.

AI USE STATEMENT

Generative AI tools were used to assist with code development, figure preparation, and language polishing. All AI-assisted code, figures, and text were reviewed and verified by the authors, who take full responsibility for the final manuscript.

ETHICS STATEMENT

PLaT provides a textual interface for inspecting intermediate planning states, which may support the study of latent reasoning. The usefulness of this interface for auditing remains unverified: fluent readouts can be incorrect or fail to faithfully describe the computation that produced an answer, and should not be treated as verified explanations or guarantees of trustworthy decisions. High candidate coverage likewise requires reliable answer selection before it can improve deployed accuracy. The lazy-decoding timing result is limited to a shared-GPU greedy setting; broader end-to-end efficiency benefits remain to be established.

REPRODUCIBILITY STATEMENT

Section 3 specifies the latent recurrence, Decoder conditioning, reconstruction loss, and stopping mechanism. Appendix B records implementation details, model selection and repeated-sampling result summaries. Appendices B.7 and C describe the readout-judgment protocol and agreement analysis. The reported repetitions concern evaluation sampling rather than independent model training.

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.

AIME. AIME problems and solutions, 2025. URL `https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions`.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 17682–17690, 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.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856.

Xinghao Chen, Anhao Zhao, Heming Xia, Xuan Lu, Hanlin Wang, Yanjun Chen, Wei Zhang, Jian Wang, Wenjie Li, and Xiaoyu Shen. Reasoning beyond language: A comprehensive survey on latent chain-of-thought reasoning. *arXiv preprint arXiv:2505.16782*, 2025.

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

John P Coetzee, Micah A Johnson, Youngzie Lee, Allan D Wu, Marco Iacoboni, and Martin M Monti. Dissociating language and thought in human reasoning. *Brain Sciences*, 13(1):67, 2022.

Yuntian Deng, Kiran Prasad, Roland Fernandez, Paul Smolensky, Vishrav Chaudhary, and Stuart Shieber. Implicit chain of thought reasoning via knowledge distillation. *arXiv preprint arXiv:2311.01460*, 2023.

Evelina Fedorenko and Rosemary Varley. Language and thought are not the same thing: evidence from neuroimaging and neurological patients. *Annals of the New York Academy of Sciences*, 1369 (1):132–153, 2016.

Evelina Fedorenko, Steven T Piantadosi, and Edward AF Gibson. Language is primarily a tool for communication rather than thought. *Nature*, 630(8017):575–586, 2024.

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

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. *arXiv preprint arXiv:2211.10435*, 2022.

Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Roziere, David Lopez-Paz, and Gabriel Synnaeve. Better & faster large language models via multi-token prediction. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=pEWAcejiU2.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. Think before you speak: Training language models with pause tokens. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=ph04CRkPdC.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models, 2024.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason E Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=Itxz7S4Ip3.

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. In *Advances in Neural Information Processing Systems*, 2021.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Hugging Face. OpenR1-Math-220k. https://huggingface.co/datasets/open-r1/OpenR1-Math-220k, 2025. Dataset release.

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.

Jia Li, Ge Li, Yongmin Li, and Zhi Jin. Structured chain-of-thought prompting for code generation. *ACM Transactions on Software Engineering and Methodology*, 34(2):1–23, 2025a.

Zhong-Zhi Li, Duzhen Zhang, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu, Junhao Zheng, Pei-Jie Wang, Xiuyi Chen, et al. From system 1 to system 2: A survey of reasoning large language models. *arXiv preprint arXiv:2502.17419*, 2025b.

Ren-Biao Liu, Anqi Li, Chaoding Yang, Hui Sun, and Ming Li. Revisiting chain-of-thought in code generation: Do language models need to learn reasoning before coding? In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=wSZeQoJ1Vk.

Jieyi Long. Large language model guided tree-of-thought. *arXiv preprint arXiv:2305.08291*, 2023.

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. Faithful chain-of-thought reasoning. In *The 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (IJCNLP-AACL 2023)*, 2023.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594, 2023.

Shentong Mo and Miao Xin. Tree of uncertain thoughts reasoning for large language models. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 12742–12746. IEEE, 2024.

Vaishnavh Nagarajan, Chen Henry Wu, Charles Ding, and Aditi Raghunathan. Roll the dice & look before you leap: Going beyond the creative limits of next-token prediction. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=Hi0SyHMmkd.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2080–2094, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.168. URL https://aclanthology.org/2021.naacl-main.168.

Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. Prophetnet: Predicting future n-gram for sequence-to-sequence pre-training. *arXiv preprint arXiv:2001.04063*, 2020.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners, 2019.

Subhro Roy and Dan Roth. Solving general arithmetic word problems. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pp. 1743–1752, 2015.

Mohammad Samragh, Arnav Kundu, David Harrison, Kumari Nishu, Devang Naik, Minsik Cho, and Mehrdad Farajtabar. Your llm knows the future: Uncovering its multi-token prediction potential. *arXiv preprint arXiv:2507.11851*, 2025.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551, 2023.

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. Codi: Compressing chain-of-thought into continuous space via self-distillation. *arXiv preprint arXiv:2502.21074*, 2025.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652, 2023.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. *Advances in Neural Information Processing Systems*, 31, 2018.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, et al. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*, 2025.

Wenhui Tan, Jiaze Li, Jianzhong Ju, Zhenbo Luo, Ruihua Song, and Jian Luan. Think silently, think fast: Dynamic latent compression of LLM reasoning chains. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=AQsko3PPUe.

Rosemary Varley and Michael Siegal. Evidence for cognition without grammar from causal reasoning and 'theory of mind'in an agrammatic aphasic patient. *Current Biology*, 10(12):723–726, 2000.

Xiaoqiang Wang, Suyuchen Wang, Yun Zhu, and Bang Liu. System-1.5 reasoning: Traversal in language and latent spaces with dynamic shortcuts, 2025. URL https://arxiv.org/abs/2505.18962.

Xinyi Wang, Lucas Caccia, Oleksiy Ostapenko, Xingdi Yuan, William Yang Wang, and Alessandro Sordoni. Guiding language model reasoning with planning tokens. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=wi9IffRhVM.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. Softcot: Soft chain-of-thought for efficient reasoning with llms. *arXiv preprint arXiv:2502.12134*, 2025.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations*, 2023.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2022.

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, 2023a.

Yao Yao, Zuchao Li, and Hai Zhao. Beyond chain-of-thought, effective graph-of-thought reasoning in language models. *arXiv preprint arXiv:2305.16582*, 2023b.

Yao Yao, Zuchao Li, and Hai Zhao. Got: Effective graph-of-thought reasoning in language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 2901–2921, 2024.

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

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. Automatic chain of thought prompting in large language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=5NTt8GFjUHkr.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, et al. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*, 2023.

A NOTATIONS

Table 1: Summary of notations used in PLaT.

Symbol	Definition	Note
x	Input question sequence	Token sequence
y	Complete reasoning chain	N segments, T tokens
y_k	The k-th textual segment	At most K tokens
k	Index of textual segments	$k \in \{1, \ldots, N\}$
i	Index within a latent group	$i \in \{1, \ldots, L\}$
$\mathcal{M}$	Planner backbone	LoRA-adapted
$\mathcal{D}$	Independent Decoder	Fully trained
d_m	Backbone hidden dimension	Latent vectors have this dimension
d_p	Projection intermediate width	Between the two linear layers
T	Total number of text tokens	Before segmentation
N	Number of textual segments / latent groups	Variable across questions
K	Tokens per segment	Main configuration: 8
L	Latent vectors per segment	Main configuration: 1
$\mathbf{h}$	Backbone hidden state	$\mathbf{h} \in \mathbb{R}^{d_m}$
$\mathbf{s}_{k,i}$	Latent vector	$\mathbf{s}_{k,i} \in \mathbb{R}^{d_m}$
$\mathbf{S}_k$	Current Decoder prefix	$(\mathbf{s}_{k,1}, \ldots, \mathbf{s}_{k,L})$
ϕ	Latent-generation projection	$\mathbb{R}^{d_m} \to \mathbb{R}^{d_m}$

B EXTRA INFORMATION OF EXPERIMENTS

B.1 MORE IMPLEMENTATION DETAILS

General Details. The GPT-2 and Llama-1B PLaT-8 configurations use LoRA dropout 0.05, one latent vector per eight-token segment, no state aggregation, and zero latent-history noise. Neither configuration initializes the Planner or Decoder from CoT-SFT weights. The projection dimension is 768 for GPT-2 and 2048 for Llama-1B, with projection dropout zero and LayerNorm enabled. The training seed is 11. The fixed-token and semantic-step variants are trained and evaluated separately. The sampling temperature for Pass@k is set to 0.9. The stored evaluation settings use at most 97 latent rollout steps and 256 generated tokens, with full decoding for the reported PLaT-8 results.

Readout settings. The reported accuracy results use full decoding. For the efficiency measurement, an online lazy variant probes each latent group for the answer prefix, skips the remaining intermediate readout after a prefix mismatch, and resumes full fixed-token decoding once the answer prefix is detected. Probe calls are included in its computational cost. This is an approximate detection rule: a prefix beginning inside a skipped segment or spanning segments can be missed, so equivalence is evaluated empirically rather than assumed.

B.2 READOUT COUNTING PROTOCOL

We compared textual readout branching for PLaT against explicit CoT on the GSM8k test set (we did not use the OOD datasets because they do not contain intermediate-step annotations for reference). For both methods, we sampled 10 reasoning paths per question (temperature=0.9) and analyzed them. We study intermediate readouts from the fixed-token PLaT model and its CoT-SFT reference on all 1,319 GSM8K test questions, using 10 stored sampled trajectories per question. The stored judgment pools contain 21,695 distinct PLaT chunks and 14,178 distinct CoT chunks. The original 100-question trajectories and their judgments are retained; the remaining CoT questions come from a separate stored sampling batch. GPT-5.4-mini receives the candidate and its stored previous/next readouts, together with reference reasoning. This context is needed because an apparent equality can be the suffix of an expression split across chunk boundaries. A truncated fragment is not invalid merely because it is incomplete.

The current pipeline distinguishes readout chunks by exact text, rather than semantic clustering. Repeated chunk text is associated with its first stored context; this is not an occurrence-level judgment of every trajectory. Consequently, these counts should be interpreted as diversity among sampled textual readouts, rather than as counts of independent Planner trajectories. Judgments cover all distinct chunks in this analysis. Existing labels are preserved, and the first stored successful judgment is used for each newly judged text. Mean branch counts are computed over 5,340 nonempty PLaT state groups and 5,979 nonempty CoT state groups, excluding empty readouts. The full-test PLaT mean branch count is 4.648 and its mean valid branch count is 4.347; the previously reported 4.225 is retained as the mean valid branch count on the original 100-question subset. The curves in Fig. 3(a) show distinct branch counts and distinct valid branch counts, normalize each question's state sequence to 11 positions, and then average across questions. Fig. 3(b) contains 1,319 questions; its valid-step count includes repeated sampled steps, unlike the distinct-branch count used in the curves.

B.3 Data Preparation and Evaluation Details

GSM8K-Aug supervision consists of a question, a sequence of equation-formatted reasoning steps, and an answer. PLaT divides the tokenized reasoning into segments of K tokens, independently of semantic step boundaries; the main comparison uses $K = 8$ and $L = 1$.

We also evaluate on MATH-500, the 500-problem test split of MATH Hendrycks et al. (2021), and the 30-problem AIME 2024 and AIME 2025 competition sets AIME (2025). For this longer-form setting, we train separate Llama-1B models on the OpenR1-Math-220k training release Hugging Face (2025). We use its cleaned `cot_answer` field as reasoning supervision, preserve the independent answer field, and apply protected-boundary tokenization with a target block length of approximately 32 tokens and a preferred upper bound of 64 tokens for the latent variants. We evaluate on all 500 MATH-500 questions and all 30 questions from each AIME year, excluding these frozen test questions from training-data preparation. Each method uses one fixed trained model and 128 ordered samples per question; Pass@k is the fraction of questions solved by at least one of the first k samples. We use the recorded mathematical-equivalence grader (LaTeX2Sympy2 1.9.1 with ANTLR 4.7.2). These results are kept separate from the GSM8K-trained comparison because the training data, trained models, and sampling protocols differ.

Within a run, the k samples form a candidate pool; they are not k independent experimental repeats. Each short-form evaluation run uses temperature 0.9, 128 ordered samples per question, and a 256-token generation cap. PLaT uses top-$k = 0$ and top-$p = 1$. Table 1 reports means over the available sampling runs for each fixed trained model; Appendix B.5 reports sample standard deviations where multiple runs are available. These repetitions measure variation across evaluation runs, not variation from retraining. Method-specific prompts and evaluation batching are retained.

B.4 Model Selection and Evaluation Protocol

For each reported configuration, we evaluate a fixed set of trained weights. PLaT Llama-3B is selected by validation performance. The fixed-token and semantic-step variants are trained and evaluated separately; models are never substituted across configurations.

CODI GPT-2 uses the public model. CODI Llama-1B uses an adapted public model, evaluated with batch size 1 and disjoint sampling-seed ranges for its three repetitions. CODI Llama-3B uses the best available model supported by validation results; the optimum over the complete training history could not be verified. Coconut Llama-1B uses learning rate 3×10^{-5}. The comparisons retain method-specific prompts, batching, and runtime implementations.

B.5 Sampling Repeats and Prefix Results

Each run uses one fixed trained model and 128 ordered samples per question. We first compute prefix Pass@k within each run, then report the mean and sample standard deviation across sampling runs. These are sampling repeats, not independently trained models. Different evaluation runtimes are retained, so the standard deviations describe the available reruns rather than isolating seed effects.

Table 2: GPT-2 prefix results (%): mean ± sample standard deviation over repeated sampling runs.

Method	Dataset	Pass@1	Pass@32	Pass@64	Pass@128
CoT-SFT	GSM8K	35.30 ± 0.54	74.20 ± 0.23	80.41 ± 0.29	86.33 ± 0.56
CoT-SFT	GSM-Hard	7.10 ± 0.44	18.35 ± 0.23	20.65 ± 0.49	22.52 ± 0.60
CoT-SFT	SVAMP	36.63 ± 0.93	65.37 ± 0.84	70.50 ± 0.82	75.47 ± 0.55
CoT-SFT	MultiArith	85.00 ± 0.00	99.07 ± 0.32	99.44 ± 0.00	99.63 ± 0.32
Coconut	GSM8K	32.94 ± 0.59	55.69 ± 0.16	61.45 ± 0.27	66.19 ± 0.64
Coconut	GSM-Hard	7.62 ± 0.05	13.65 ± 0.32	15.01 ± 0.11	16.15 ± 0.11
Coconut	SVAMP	35.25 ± 0.21	50.15 ± 0.78	53.05 ± 0.64	56.45 ± 0.49
Coconut	MultiArith	81.67 ± 0.79	90.83 ± 0.39	91.94 ± 1.18	93.33 ± 0.00
CODI	GSM8K	40.16 ± 1.18	63.00 ± 0.50	67.12 ± 0.19	69.90 ± 0.15
CODI	GSM-Hard	8.74 ± 0.19	14.58 ± 0.09	15.52 ± 0.12	16.53 ± 0.08
CODI	SVAMP	38.60 ± 0.95	53.67 ± 0.40	55.80 ± 0.36	58.13 ± 0.29
CODI	MultiArith	92.04 ± 0.85	96.67 ± 0.00	96.67 ± 0.00	96.67 ± 0.00
PLaT-8	GSM8K	18.35 ± 0.11	59.51 ± 1.29	67.74 ± 0.80	76.08 ± 1.23
PLaT-8	GSM-Hard	3.75 ± 0.05	14.22 ± 0.16	17.06 ± 0.64	18.88 ± 0.11
PLaT-8	SVAMP	21.80 ± 0.71	53.70 ± 1.41	60.75 ± 1.91	66.45 ± 1.20
PLaT-8	MultiArith	33.61 ± 2.75	79.44 ± 0.79	87.50 ± 0.39	91.94 ± 0.39

Table 3: Llama-1B prefix results (%): mean ± sample standard deviation over repeated sampling runs.

Method	Dataset	Pass@1	Pass@32	Pass@64	Pass@128
CoT-SFT	GSM8K	49.96 ± 0.32	86.62 ± 0.48	90.45 ± 0.54	93.18 ± 0.21
CoT-SFT	GSM-Hard	11.45 ± 0.20	23.68 ± 0.64	25.50 ± 0.44	27.72 ± 0.19
CoT-SFT	SVAMP	59.50 ± 0.95	87.97 ± 0.31	91.07 ± 0.25	93.37 ± 0.40
CoT-SFT	MultiArith	94.07 ± 1.70	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
Coconut	GSM8K	27.52 ± 0.57	49.28 ± 0.50	53.68 ± 0.47	58.07 ± 0.67
Coconut	GSM-Hard	6.47 ± 0.29	11.95 ± 0.38	13.17 ± 0.36	14.18 ± 0.55
Coconut	SVAMP	41.87 ± 0.12	53.13 ± 0.29	54.90 ± 0.53	56.97 ± 0.75
Coconut	MultiArith	59.07 ± 1.95	85.93 ± 0.85	87.41 ± 0.64	89.44 ± 0.56
CODI	GSM8K	54.55 ± 0.05	69.11 ± 0.38	71.08 ± 0.48	72.37 ± 0.05
CODI	GSM-Hard	12.24 ± 0.05	16.41 ± 0.48	17.13 ± 0.43	17.55 ± 0.16
CODI	SVAMP	58.75 ± 0.07	70.70 ± 0.57	72.15 ± 0.07	73.00 ± 0.00
CODI	MultiArith	97.22 ± 0.79	98.33 ± 0.00	98.33 ± 0.00	98.33 ± 0.00
PLaT-8	GSM8K	36.20 ± 0.38	68.73 ± 0.05	75.28 ± 0.00	80.59 ± 0.32
PLaT-8	GSM-Hard	8.26 ± 0.00	16.30 ± 0.54	18.23 ± 0.05	19.75 ± 0.16
PLaT-8	SVAMP	43.00 ± 0.95	66.87 ± 0.42	70.63 ± 0.55	74.47 ± 0.32
PLaT-8	MultiArith	84.63 ± 1.16	96.85 ± 0.85	97.41 ± 1.28	98.70 ± 0.85

B.6 Ablation Results by Dataset

All displayed entries report means ± sample standard deviations over verified sampling runs. The semantic-step row uses one latent vector per complete training step, while fixed-K rows use one latent vector per K-token segment; the two rows therefore change both segmentation and training configuration. Semantic-step segmentation is practical when the supervision has reliable step boundaries, but is difficult to transfer to less regularly structured solutions; fixed-token segmentation avoids requiring such boundaries. The CoT-initialization row is a historical training variant and does not isolate the effect of initialization. Decoder-context variants use the PLaT-8 configuration. OpenR1-trained evaluations are separate.

B.7 Agreement of LLM Judgments

We drew 100 unique chunks per method without replacement using seed 20260915, independently of GPT tags. The eligible pools contained 1,725 PLaT records and 3,951 CoT records. We excluded 28 and 119 records, respectively, that lacked complete Boolean judgment fields or stored neighboring context. The earlier 31-item stratified pilot is separate from this 200-item review.

All 200 human labels are valid or invalid (181 valid, 19 invalid). The reviewer could see the GPT tags and explanations. The resulting agreement therefore characterizes assisted review and cannot establish independent inter-rater reliability. The mapping is invalid when `malformed=true`, valid when `valid_local=true` and `malformed=false`, and otherwise ambiguous. The `incomplete` tag alone does not imply invalidity.

With valid as the positive class, there are five false positives and nine false negatives, plus two GPT abstentions on human-valid items. Agreement is 184/200 (92.0%). Cohen's $\kappa = 0.5943$ is computed over valid, invalid and ambiguous categories; it is not a binary calculation that discards abstentions. PLaT has agreement 97.0% and $\kappa = 0.7845$; CoT-SFT has agreement 87.0% and $\kappa = 0.4821$. These labels do not establish correctness of every occurrence of a repeated chunk.

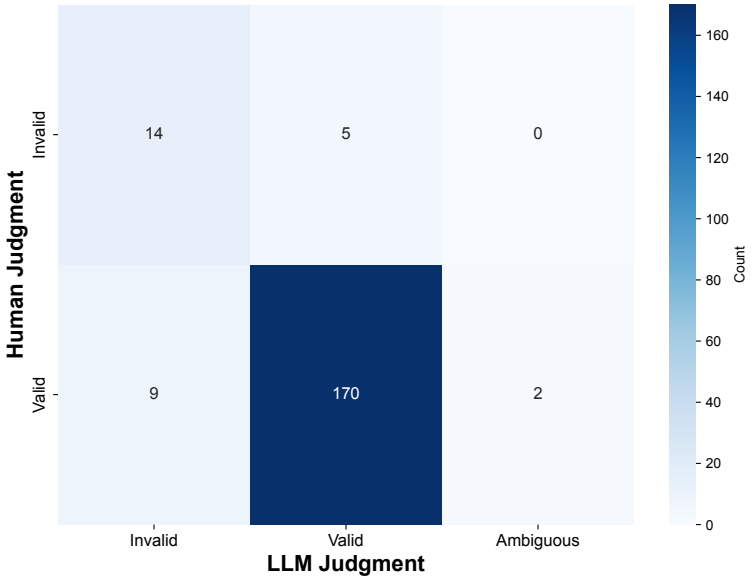


Figure 1: **Confusion matrix of Human-LLM agreement.** All 200 reviewed items are included. The two ambiguous LLM judgments count as disagreements.

B.8 Case Study

The deterministic answer is -10, while sample 0 gives the reference answer 10 at the same answer-start state. This illustrates variation in sampled readouts, without establishing the validity of every intermediate fragment or latent superposition.

Table 4: Llama-3B prefix results (%): mean ± sample standard deviation over repeated sampling runs.

Method	Dataset	Pass@1	Pass@32	Pass@64	Pass@128
CoT-SFT	GSM8K	58.88 ± 0.93	92.90 ± 0.53	95.27 ± 0.69	96.71 ± 0.32
CoT-SFT	GSM-Hard	18.95 ± 0.15	38.34 ± 0.27	41.37 ± 0.76	44.12 ± 1.28
CoT-SFT	SVAMP	74.67 ± 1.05	95.50 ± 0.17	96.90 ± 0.20	97.83 ± 0.21
CoT-SFT	MultiArith	97.78 ± 1.67	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
Coconut	GSM8K	28.38 ± 0.83	62.32 ± 0.13	70.18 ± 0.69	75.87 ± 0.34
Coconut	GSM-Hard	6.09 ± 0.37	15.62 ± 0.20	17.29 ± 0.13	19.13 ± 0.27
Coconut	SVAMP	34.07 ± 0.57	57.47 ± 0.35	61.73 ± 0.86	65.43 ± 0.55
Coconut	MultiArith	89.26 ± 2.51	97.04 ± 0.32	97.22 ± 0.00	97.96 ± 0.32
CODI	GSM8K	57.16 ± 0.43	76.72 ± 0.00	80.44 ± 0.43	83.24 ± 0.32
CODI	GSM-Hard	13.38 ± 0.27	18.12 ± 0.43	19.45 ± 0.16	20.89 ± 0.38
CODI	SVAMP	68.70 ± 0.71	80.65 ± 0.35	82.60 ± 0.71	84.60 ± 0.99
CODI	MultiArith	99.72 ± 0.39	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
PLaT-8	GSM8K	39.85 ± 0.85	73.82 ± 0.42	80.09 ± 0.35	84.79 ± 0.31
PLaT-8	GSM-Hard	9.48 ± 0.20	17.66 ± 0.15	19.23 ± 0.04	20.87 ± 0.12
PLaT-8	SVAMP	54.77 ± 1.12	76.00 ± 0.50	79.43 ± 0.64	82.87 ± 0.74
PLaT-8	MultiArith	95.37 ± 0.85	98.89 ± 0.00	99.63 ± 0.32	99.63 ± 0.32

Table 5: Llama-1B GSM8K ablations (%).

	Pass@1	Pass@32	Pass@64	Pass@128
CoT initialization	29.95 ± 0.43	67.48 ± 0.86	74.15 ± 0.43	78.01 ± 0.11
semantic-step	37.86 ± 1.10	69.45 ± 0.46	75.16 ± 0.62	80.82 ± 0.62
PLaT-2	13.98 ± 0.95	49.43 ± 1.32	58.78 ± 0.86	68.54 ± 1.19
PLaT-5	34.72 ± 0.96	69.56 ± 0.27	75.93 ± 0.48	81.80 ± 0.21
PLaT-8	36.20 ± 0.38	68.73 ± 0.05	75.28 ± 0.00	80.59 ± 0.32
PLaT-8, $L = 2$	37.04 ± 0.27	68.69 ± 0.11	74.53 ± 0.21	79.49 ± 0.27
PLaT-8, $L = 3$	37.38 ± 1.61	68.80 ± 0.27	74.64 ± 0.48	79.80 ± 0.48
PLaT-16	37.45 ± 0.11	65.43 ± 0.54	69.71 ± 0.38	73.84 ± 0.96
PLaT-32	32.79 ± 0.05	62.85 ± 0.11	68.92 ± 0.75	73.12 ± 0.91
Decoder latent history	38.51 ± 0.11	67.29 ± 0.70	73.62 ± 0.21	78.85 ± 0.32
Decoder question + latent history	31.61 ± 0.95	67.73 ± 0.99	74.37 ± 0.23	80.29 ± 0.82

Table 6: Llama-1B GSM-Hard ablations (%).

	Pass@1	Pass@32	Pass@64	Pass@128
CoT initialization	7.25 ± 0.31	16.50 ± 0.12	18.52 ± 0.55	20.04 ± 0.42
semantic-step	9.00 ± 0.31	17.11 ± 0.32	18.65 ± 0.35	20.50 ± 0.37
PLaT-2	2.77 ± 0.27	10.80 ± 0.27	13.68 ± 0.16	16.72 ± 0.48
PLaT-5	8.04 ± 0.21	17.02 ± 0.27	18.95 ± 0.21	20.58 ± 0.38
PLaT-8	8.26 ± 0.00	16.30 ± 0.54	18.23 ± 0.05	19.75 ± 0.16
PLaT-8, $L = 2$	8.30 ± 0.59	16.91 ± 0.21	18.57 ± 0.32	19.94 ± 0.21
PLaT-8, $L = 3$	9.02 ± 0.21	16.76 ± 0.75	18.50 ± 0.43	19.94 ± 0.32
PLaT-16	8.53 ± 0.05	15.66 ± 0.05	16.87 ± 0.27	18.01 ± 0.16
PLaT-32	7.73 ± 0.00	15.09 ± 0.32	16.60 ± 0.43	18.01 ± 0.27
Decoder latent history	8.53 ± 0.05	16.64 ± 0.80	18.42 ± 0.54	19.71 ± 0.43
Decoder question + latent history	7.05 ± 0.26	16.27 ± 0.16	18.30 ± 0.31	19.86 ± 0.30

Table 7: Llama-1B SVAMP ablations (%).

	Pass@1	Pass@32	Pass@64	Pass@128
CoT initialization	36.70 ± 0.14	65.40 ± 1.70	69.55 ± 0.07	72.35 ± 0.07
semantic-step	44.20 ± 0.44	68.77 ± 0.49	72.37 ± 0.65	75.93 ± 0.84
PLaT-2	19.50 ± 0.14	50.80 ± 0.71	57.90 ± 0.85	65.15 ± 0.64
PLaT-5	45.05 ± 0.21	68.70 ± 0.42	73.15 ± 0.07	77.45 ± 0.21
PLaT-8	43.00 ± 0.95	66.87 ± 0.42	70.63 ± 0.55	74.47 ± 0.32
PLaT-8, $L = 2$	45.10 ± 0.28	67.45 ± 0.07	71.75 ± 0.21	75.50 ± 0.42
PLaT-8, $L = 3$	47.10 ± 0.00	67.35 ± 0.49	70.50 ± 1.27	73.70 ± 0.71
PLaT-16	42.70 ± 0.57	61.20 ± 0.00	64.35 ± 0.21	67.15 ± 0.07
PLaT-32	40.65 ± 0.49	56.65 ± 0.21	59.25 ± 0.07	62.40 ± 0.28
Decoder latent history	45.65 ± 0.07	66.80 ± 0.57	70.35 ± 0.07	73.45 ± 0.49
Decoder question + latent history	33.60 ± 0.95	61.73 ± 0.68	66.53 ± 0.55	71.23 ± 0.81

Table 8: Llama-1B MultiArith ablations (%).

	Pass@1	Pass@32	Pass@64	Pass@128
CoT initialization	72.50 ± 1.18	93.61 ± 0.39	95.56 ± 0.00	95.83 ± 0.39
semantic-step	88.89 ± 1.67	95.74 ± 0.64	96.67 ± 0.56	96.85 ± 0.64
PLaT-2	27.50 ± 0.39	86.11 ± 0.79	91.94 ± 1.96	95.00 ± 0.00
PLaT-5	78.61 ± 0.39	95.83 ± 0.39	97.50 ± 0.39	98.33 ± 0.00
PLaT-8	84.63 ± 1.16	96.85 ± 0.85	97.41 ± 1.28	98.70 ± 0.85
PLaT-8, $L = 2$	93.61 ± 0.40	98.89 ± 0.78	99.44 ± 0.00	99.44 ± 0.00
PLaT-8, $L = 3$	96.94 ± 1.96	99.44 ± 0.00	99.44 ± 0.00	99.72 ± 0.39
PLaT-16	84.17 ± 0.39	95.00 ± 0.79	96.11 ± 0.00	96.94 ± 0.39
PLaT-32	60.28 ± 4.32	82.22 ± 0.79	84.72 ± 0.39	87.78 ± 0.00
Decoder latent history	93.06 ± 0.39	98.06 ± 0.39	98.06 ± 0.39	99.17 ± 0.39
Decoder question + latent history	60.93 ± 1.16	91.67 ± 1.11	93.89 ± 1.11	96.30 ± 0.32

B.9 Entropy Analysis

We evaluate the fixed-token PLaT model on all 1,319 GSM8K test questions using deterministic generation for this entropy analysis. Only question text is provided to the model.

To investigate the internal decision-making process, we measure the Shannon entropy of the first-token distribution decoded from each generated latent state. For a given latent state $\mathbf{S}_k$ at reasoning step k, let $P_{\mathcal{D}}(v \mid \mathbf{S}_k)$ denote the probability of token v being the first token generated by the Decoder. The reasoning entropy of PLaT $H(\mathbf{S}_k)$ is defined as:

$$H(\mathbf{S}_k) = -\sum_{v \in \mathcal{V}} P_{\mathcal{D}}(v \mid \mathbf{S}_k) \log P_{\mathcal{D}}(v \mid \mathbf{S}_k) \tag{7}$$

where $\mathcal{V}$ is the vocabulary and $\mathcal{D}$ is the Decoder. In our analysis, we normalize the reasoning progress of each sample to $[0, 100\%]$ to aggregate samples with varying lengths.

Appendix Figure 3 reports the mean entropy in nats, computed before sampling transformations. We exclude the state in which the answer prefix is detected and all subsequent states. All 1,319 generations contain a detected answer prefix. One question has the answer prefix detected in its first state and therefore has no retained reasoning state; it is excluded from the entropy average. The remaining 1,318 sequences contain 1–8 states per question; each is linearly interpolated to 20 normalized progress points (constant for a single state). These points do not represent 20 independent reasoning steps. For reference, we retain the original CoT-SFT, Coconut, and CODI measurements (1,319 questions per method); these baselines have not been rerun. To reproduce the original baseline curves, we remove the final entropy value from each cached sequence and interpolate to 21 progress points. Their original method-specific probes are retained: CoT-SFT uses generated-token distributions, while Coconut and CODI use latent-step vocabulary distributions. Accordingly, the curves are descriptive references rather than a matched-model comparison.

B.10 Detailed Analysis of Hyperparameters

We report the single-run sensitivity plots for token-group size K and latent vectors per segment L.

Token-group Size (K). Figure 4 reports $K = 2, 5, 8, 16, 32$ with $L = 1$.

Table 9: Human versus GPT labels on all 200 reviewed items. Columns are GPT labels.

Human label	Valid	Invalid	Ambiguous
Valid	170	9	2
Invalid	5	14	0

Table 10: **Human-LLM Agreement Metrics.**

Metric	Cohen's Kappa (κ)	Accuracy
Value	0.5943	0.9200

Sensitivity to Latent Vectors per Segment (L). The sweep varies $L = 1, 2, 3$ at fixed $K = 8$.

C Prompts for LLM Judgments

The current evaluator receives the problem/reference reasoning, candidate chunk, and stored previous and next readouts. It first inspects the local window for equations split across boundaries. A plausible arithmetic fragment with no explicit complete arithmetic error may be marked locally valid even when truncated. Only genuinely nonsensical content or an explicitly incorrect complete equality is marked malformed. Reference reasoning is contextual information, not a requirement to follow the reference solution's ordering or path. The output fields are `coherent`, `valid_local`, `incomplete`, `malformed`, and `reason`. This permissive local criterion differs from proving the correctness of a complete reasoning step. No semantic clustering is used in the current exact-text branch counts.

Prompt for Validating Equations

You evaluate one candidate fixed−token readout from mathematical reasoning.
The candidate is a short token window, not necessarily a complete sentence or equation.
The symbols << and >> are formatting/chunk−boundary markers: ignore them completely.
Gold reasoning steps and gold fixed−token chunks are reference only. They explain the problem and typical operations, but do NOT require the candidate to match the gold solution, prefix, ordering, or arithmetic continuity.

The user message includes the previous and next readout from the SAME generated trajectory when available. These are context, not additional candidate text.
First inspect the concatenated local window to determine whether an apparent equation is split across chunk boundaries. Then judge the candidate chunk:
– If it contains a plausible arithmetic/semantic fragment and no explicit complete arithmetic error, set valid_local=true.
– A fragment may begin/end mid−number, operator, or equation; missing context is not evidence of invalidity. Set incomplete=true when visibly truncated, but keep valid_local=true if the visible operation is plausible.
– Do not mark a candidate malformed merely because it looks like 'a=b' in isolation when the neighboring readouts show it is the suffix/prefix of a larger expression such as 'c+a=b'.
– Set malformed=true only for genuinely nonsensical text or an explicitly incorrect complete equality (e.g. 3∗43=128). Do not call boundary markers, dangling operators, or missing operands malformed.
Return JSON only with keys coherent, valid_local, incomplete, malformed, reason.

D Limitations and Future Work

Candidate coverage and textual inspectability have distinct evidential limits. Higher Pass@k does not directly establish better final-answer selection, and locally valid readouts do not establish complete-chain consistency or causal faithfulness to the Planner's computation. Evaluating a common answer

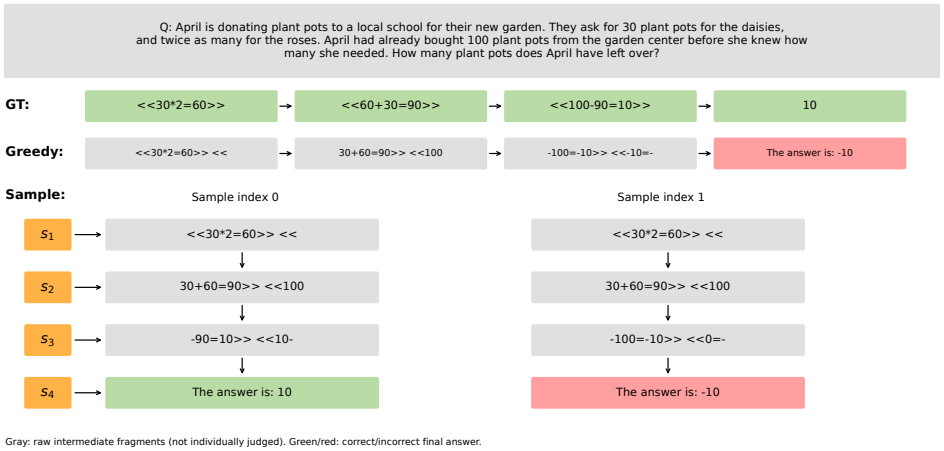


Figure 2: **Visualization of PLaT's reasoning process.** *GT*, *Deterministic*, and *Sample* are the ground-truth reasoning steps in the dataset, the deterministic readouts from each group of states, and sampled readouts from each group of states, respectively. This example is drawn from the GSM8K test set, with two stored sampled trajectories. State labels are one-based. Gray boxes preserve raw intermediate fragments without individual validity labels; green and red final-answer boxes indicate correctness and incorrectness. Arrows connect consecutive readouts within each stored trajectory.

selector and testing how latent-state interventions affect readouts and final answers would address these questions.

The architecture permits a variable number of latent groups, but the current experiments do not establish how performance scales with latent computation. The single-run sensitivity results vary with token-group size K and latent-vector count L, without establishing monotonic improvement or isolating the effects of optimization, representation capacity, and readout configuration. Controlled studies are needed to separate these factors. The efficiency measurements cover GPT-2 greedy inference; costs for other backbones and the multi-sample regime remain to be characterized.

Our evaluation covers mathematical reasoning; transfer to less structured tasks remains untested. The reported sampling repeats quantify variation in evaluation with fixed trained weights, rather than variation across independent training runs. Broader task coverage and independently trained repetitions would help assess the generality and stability of the observed trade-offs.

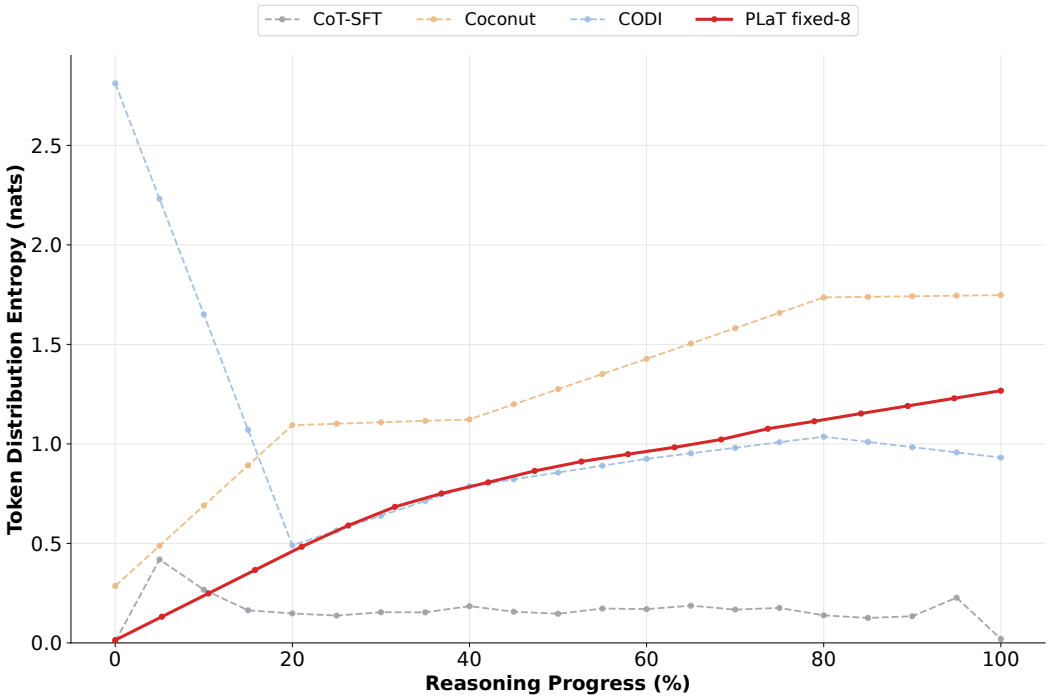


Figure 3: **Evolution of Token Distribution Entropy over Normalized Reasoning Progress.** Updated PLaT-8 first-token decoder entropy (1,318 of 1,319 questions with retained reasoning states, solid red) with original CoT-SFT, Coconut, and CODI measurements (1,319 questions each, dashed). Each question contributes equally after interpolation (20 progress points for PLaT; 21 for the original baselines). Entropy is measured in nats; baseline model outputs and method-specific probes are retained from the original analysis.

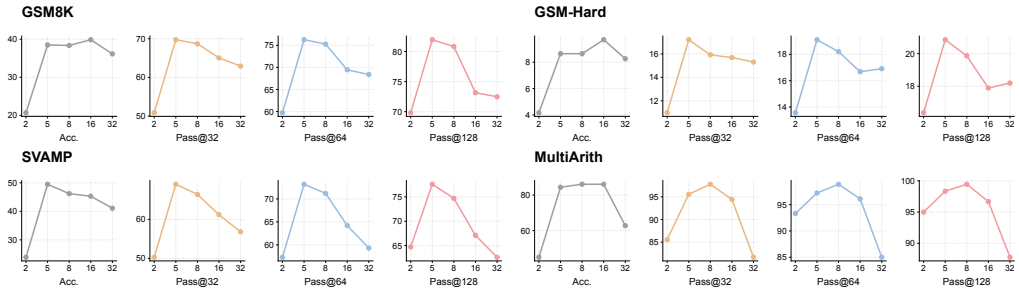


Figure 4: Sensitivity to token-group size K with one latent vector per segment. Single-run results on four datasets; no mean or standard deviation across repeated runs is reported.

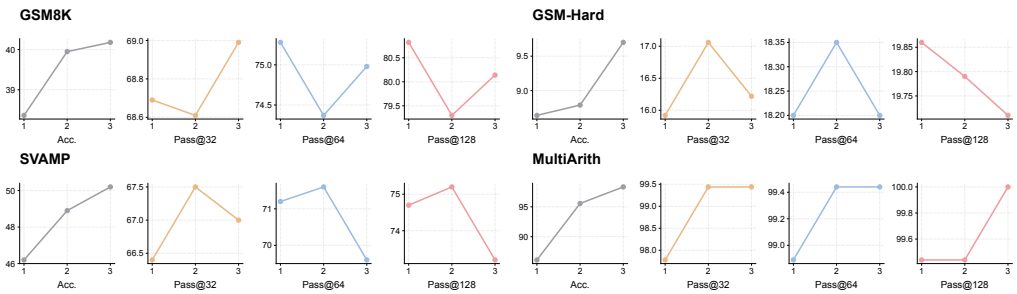


Figure 5: Impact of latent vectors per segment L at fixed token-group size $K = 8$. Single-run results on four datasets, without repeat-run error bars.