Training Continuous Chain of Thought Models: A Tale of Two Regimes

Varun Yerram [1] **He He** [1] **Eunsol Choi** [1]

Abstract

Continuous Chain-of-Thought methods replace verbose reasoning traces with a short sequence of dense latent representations. Earlier continuous CoT methods indirectly supervise the latent representations such that its final state match that of verbose reasoning traces, requiring autoregressive, slow generation during training. We introduce C-MTP, a simpler, faster direct supervision approach that models each latent as an average of the embeddings in the CoT traces to be compressed. Our approach outperforms a prior direct supervision method that approximates the distribution of compressed tokens, and performs competitively to slower indirect supervision approaches in existing evaluation setup with simplified CoT traces (less than 100 tokens). Lastly, we extend the evaluation of Continuous CoT methods to complex tasks with longer reasoning traces ($\geq$ few hundreds reasoning tokens). We find both direct and indirect supervision training methods perform poorly (roughly 65% performance drop) in this setting, revealing the limitations of current continuous CoT methods. The code and checkpoints are released at https://github.com/Varun221/cmtp_research.

1. Introduction

Recent LLMs achieve remarkable performance on complex reasoning tasks via generating lengthy sequence of tokens that describe their reasoning process, often called chain-of-thought (CoT) traces (Wei et al., 2022b; Kojima et al., 2022), expanding the expressive power of LLMs (Merrill & Sabharwal, 2024). However, autoregressively generating long CoT traces increases the computational cost significantly.

Continuous Chain of Thought (ContinuousCoT), pioneered by Hao et al. (2025), allows an LLM to reason entirely in a short sequence of dense latent representations without discrete "thinking" tokens, significantly reducing reasoning length and, therefore, inference compute. Current approaches exceed the performance of discrete CoT on structured graph search tasks like ProsQA (Hao et al., 2025) and ProntoQA (Saparov & He, 2023) and show competitive performance (about 10% performance degradation) on math reasoning tasks (Cobbe et al., 2021; Deng et al., 2024) These methods (Shen et al., 2025; Wei et al., 2026) train a model to autoregressively predict intermediate latents and use a CoT teacher to supervise the final output. This allows the model to compress varying length CoT trajectories into a fixed number of latents. The sequential autoregressive passes required during training make it computationally expensive. As these methods utilize chain-of-thought traces *indirectly*, we refer to them as using **indirect supervision**. These methods start from a pretrained discrete token LLM and fine-tune it to predict latents that are fed back into the model for a fixed number of steps.

Figure 1. Head-to-Head comparison of standard CoT finetuning with two ContinuousCoT training methods: **Direct Supervision** (C-MTP, our proposed method) and **Indirect Supervision** (CODI) on three datasets of different CoT traces. *Structured* consists of compact mathematical expressions of $\sim$25 tokens, *Semi-Natural* contains sentence per-step explanations of $\sim$62 tokens while *Realistic* traces captures traces generated from off-the-shelf LLM ($\sim$350 tokens on average). Both methods remain competitive with CoT-SFT on structured/semi-natural traces but collapse to $\sim$35% of its accuracy on realistic traces.

Recent work, CoLaR (Tan et al., 2026), mitigates this by modeling latents as parameterized Gaussian distributions conditioned on token embeddings aggregated from a CoT teacher. This enables a parallelized training objective and

[1]Center for Data Science, New York University. Correspondence to: Varun Yerram <y.varun@nyu.edu>.

Published at the ICML 2026 Workshop, AdaptFM: Resource-Adaptive Foundation Model Inference, Seoul, South Korea. PMLR 306, 2026. Copyright 2026 by the author(s).

brings the training distribution closer to pretraining. However, the model must predict a distribution over the latent space, which poses optimization challenges. Since this method utilizes CoT traces directly for training and prediction we term such methods as using **direct supervision**. Unlike indirect supervision method, direct supervision methods enable parallel training.

In this work, we introduce a new direct supervision approach: **C-MTP**, that models Multi-Token Prediction (MTP) to simulate latent reasoning. We formalize intermediate latents as aggregated token embeddings within a chain-of-thought, training the model to predict the tokens composing each latent rather than a continuous vector. This aligns with the pretraining objective while enabling fully parallel training via teacher forcing over CoT traces, eliminating the need for sequential autoregressive passes. Our evaluations show that C-MTP significantly outperforms existing direct supervision methods due to its simplicity and alignment with the pretraining objective. With small structured reasoning traces, C-MTP is faster, sample-efficient and generalizes better to out-of-distribution problems (Sections 3.2 and 3.2.1) than indirect supervision methods.

On verbose semi-natural CoTs, we find that indirect supervision outperforms direct supervision, primarily because its recurrent training compresses long reasoning traces into few latents, while direct supervision is bounded by the original trace length and suffers from error accumulation under teacher-forcing at inference. Finally, on moving to realistic CoT traces from a strong teacher, both regimes fall well short of CoT-SFT. This suggests that structured CoT evaluations, which dominate current benchmarks, overstate the practical utility of ContinuousCoT methods.

We state our contributions as follows:

- We introduce **C-MTP**, a novel direct supervision framework that uses Multi-Token Prediction to simulate continuous latent reasoning.
- On structured reasoning tasks, C-MTP outperforms existing direct supervision methods and matches or exceeds indirect supervision methods while being faster, more sample-efficient, and generalizing better to out-of-distribution problems.
- We conduct a systematic empirical comparison across structured, semi-natural and realistic CoT trace regimes, showing that indirect supervision overtakes direct on verbose semi-natural traces through its compression objective, and that both regimes fall short of CoT-SFT on realistic traces, exposing the limitations of ContinuousCoT approaches.

2. Methodology

2.1. Notation and Problem Setup

Consider an LLM M_θ. Let $\mathcal{V}$ be its Vocabulary and $E \in \mathbb{R}^{|\mathcal{V}| \times d}$ be the Embedding matrix. Given an input prompt $Q = (q_1, q_2, \ldots, q_m) \in \mathcal{V}^m$ we denote the embedded prompt as $E(Q) = (e(q_1), e(q_2), \ldots, e(q_m))$, where $e(q_i) \in \mathbb{R}^d$. Given a prompt, we can obtain the final layer hidden state at the final position from a student model M_θ with a function denoted by $H_\theta(.) \in \mathbb{R}^d$.

Let's consider that we have access to K reasoning steps for the given prompt. Denote them as $y := (y_1, y_2, \ldots, y_K)$. Each $y_i = (y_{i1}, y_{i2}, \ldots, y_{iL_i}) \in \mathcal{V}^{L_i}$ and L_i is the token length of the i-th textual reasoning step. Finally, we denote an answer as $a = (a_1, a_2, \ldots, a_n) \in \mathcal{V}^n$.

Standard CoT supervised finetuning optimizes the following objective:

$$L_{CoT} = -\log P_\theta(a \mid Q, y_1, y_2, \ldots, y_K) \tag{1}$$

Where $P_\theta(.)$ is the probability distribution defined by the student model M_θ. (Through $H_\theta(.)$ and a softmax over the vocabulary). Continuous CoT aims to replace textual reasoning vectors with a sequence of dense latents $z := (z_1, z_2, \ldots, z_K)$* obtained by

$$z_k = H_\theta(I^{(k-1)}) \quad I^{(k)} = I^{(k-1)} \oplus z_k \tag{2}$$

Where, $I^{(0)} = E(Q)$ is the embedded input prompt and each $z_i \in \mathbb{R}^d$. $\oplus$ denotes concatenation along the sequence axis. The Continuous Chain of Thought is therefore defined as the sequence of continuous vectors $z := z_1, z_2, \ldots, z_K$. The primary goal of all ContinuousCoT methods is to design an effective training recipe to learn z such that the final answer a can be generated conditioned only on z and Q.

2.2. Prior Work: Indirect Supervision Methods

Coconut (Hao et al., 2025) uses a curriculum-based strategy where at each iteration, one of the textual reasoning steps y_i is replaced with the corresponding latent z_i starting from $i = 0$. The following objective is optimized at each step i.

$$L_i = -\log P_\theta(a \mid Q, z_1, \ldots, z_i, y_{i+1}, \ldots, y_K) \tag{3}$$

This progressively teaches M_θ to generate semantically useful latents and eventually answer based on these latents alone. CODI (Shen et al., 2025) improves training efficiency by replacing curriculum with distillation. The student, M_θ autoregressively generates K latents during training. Using these latents, it is trained to generate the answer.

$$L_{Student} = -\log P_\theta(a \mid Q, z_1, z_2, \ldots, z_K) \tag{4}$$

*Note that number of latents need not equal reasoning steps of a problem. Here we consider equal case for simplicity.

Table 1. **Comparison of indirect and direct supervision regimes for ContinuousCoT training.** Indirect methods reuse the model's own final-layer activation via recurrence, producing continuous latents. Direct methods instead map predictions through the embedding table, constraining outputs to the token space. For indirect methods, the reasoning length (number of recurrent latent steps) is a fixed hyperparameter set before training, whereas direct methods operate over variable-length CoT traces via teacher forcing.

	Indirect Supervision (e.g., CODI, Sim-CoT, COCONUT)	**Direct Supervision** (e.g., CoLaR, C-MTP)
Input	Final Layer Activation	Aggregate of Token Embeddings
Output	Continuous	Discrete
Training Efficiency	Slow (Multiple Forward passes)	Fast (1 Forward pass)
Reasoning Length	Fixed	Variable
Training Method	Recurrence	Teacher Forcing

The teacher M_ϕ (warm started with θ) is trained using standard CoT traces using Equation (1)[†]. The intermediate layer activations of M_ϕ are distilled to M_θ to guide student training substantially improving training efficiency and downstream performance.

$$L_{KD} = \frac{1}{M}\sum_{l=1}^{M} |sg[h_\phi^l] - h_\theta^l| \tag{5}$$

Where M is the number of layers in the model, sg denotes stop-gradient, and $h_\cdot^l$ denotes the l-th layer activation at the final token before answer generation, indexed by the corresponding model (ϕ for teacher, θ for student). Recently, Wei et al. (2026) argued that when K is increased to larger values, the latents degenerate to similar representations. To stabilize this effect they introduce step-level supervision using intermediate reasoning steps y. A Decoder LLM N_ψ is trained to autoregressively generate the k-th reasoning step conditioned only on the latent z_k. Authors show that adding this step-level supervision to CODI or Coconut improves performance significantly on downstream tasks.

These works share three characteristics: First, the number of latent steps is fixed prior to training, resulting in fixed-length reasoning steps. Second, the sequential, autoregressive generation of latents during training imposes significant overhead. Finally, these methods leverage textual reasoning steps **indirectly** for supervision; either through teacher distillation or by using latents to decode them back.

[†]Shen et al. (2025) uses multi-task training where student and teacher share same weights. We see the effect of this in more detail in Appendix H.1

2.3. Prior Work: Direct Supervision Methods

CoLaR (Tan et al., 2026) proposed a method to utilize reasoning steps directly by compressing embeddings of every s consecutive tokens in reasoning trace y into a latent z_i. During training, one token is sampled from each s-token group to form a subsampled reasoning trace. Model, M_θ is then trained via teacher forcing to predict this sampled trace and the final answer. This approach trains the model's predictions to approximate a multi-modal distribution representing all potential tokens within each compressed group. The aggregated embeddings are modeled as parameterized Gaussians, and a dedicated latent head is trained to predict the mean and standard deviation for each step. During inference, they use reparameterization trick to sample from this distribution, passing the resulting latent vector to the next autoregressive step.

2.4. C-MTP: Continuous Chain of Thought via Multi-Token Prediction

Similar to CoLaR, we view intermediate latent as aggregated token embeddings. However instead of modeling them as gaussians we provide a simpler recipe using Multi-Token Prediction (Gloeckle et al., 2024).[‡]

During training, we set a desired compression factor s, the number of tokens per span. Given the reasoning trace $y = (c_1, \ldots, c_L)$, we partition it into $r = \lceil L/s \rceil$ non-overlapping spans of length s:

$$C_i = (c_{(i-1)s+1}, \ldots, c_{is}), \quad i = 1, \ldots, r \tag{6}$$

Each span is compressed into a single latent by averaging its token embeddings[§]:

$$z_i = f(C_i) = \frac{1}{s}\sum_{j=1}^{s} E\big(c_{(i-1)s+j}\big) \tag{7}$$

This yields a sequence of r latents $z = (z_1, \ldots, z_r)$ that replaces the full reasoning trace y during the forward pass. At each token position, we want to model $P_\theta(z_i \mid Q, z_1 \ldots z_{i-1})$. Using Equation (7), we cast the problem of next latent prediction as predicting the tokens composing the latent. We train the student, M_θ to minimize

$$L_{multi} = -\sum_{i=1}^{r}\sum_{t \in C_i} \log P_\theta(t \mid E(Q), z_1, \ldots, z_{i-1}) \tag{8}$$

The answer supervision stays unchanged

$$L_{ans} = -\sum_{j=1}^{n} \log P_\theta(a_j \mid E(Q), z_{1:r}) \tag{9}$$

[‡]We describe our exact architecture in Section B (Appendix)

[§]Tan et al. (2026) divides by $\sqrt{s}$ instead of s. We did not find any difference in our experiments and hence use simple average for simplicity.

$$L_{MTP} = L_{multi} + L_{ans} \tag{10}$$

To guide training, we also employ a CoT Teacher for distillation using L_{KD} (Defined in Equation (5)). Overall, we employ two training configurations in this work: **Warmstart Init** where we first train a CoT teacher using L_{CoT}, use it to initialize the student, and then train the student using $L_{MTP} + L_{KD}$; and **Multitask Training** where we train the same backbone to optimize both L_{CoT} and $L_{MTP} + L_{KD}$ simultaneously. We found that Warmstart Init performs better in out-of-domain tasks and keep it for main experiments, we use multitask training for fair comparison with CODI in relevant experiments.

During inference, the trained M_θ first goes through a CoT phase and then answer phase. In the CoT phase, at each step i the model samples s tokens from $P_\theta(\cdot \mid E(Q), z_1, \ldots, z_{i-1})$, aggregates their embeddings to form the next latent z_i via Equation (7), and appends it to the latent sequence. This repeats until the end-of-thinking token is among the sampled tokens. In the answer phase, the model switches to standard autoregressive next-token prediction conditioned on $E(Q)$ and the accumulated latents $z_{1:r}$, continuing until the end-of-sequence token is produced.

We discuss the relevant literature and connection to other works in latent reasoning in Section A in the appendix.

3. Experimental Results and Analysis

In this section we discuss the setup, training details, experiments and detailed analysis of the proposed architecture.

3.1. Setup

Datasets and Tasks. We primarily use **GSM8k-Aug** (Deng et al., 2024), an augmented version of Grade-School level Math reasoning dataset GSM8k (Cobbe et al., 2021). It comprises ∼385k training samples of which we reserve 1% for validation and use rest for training. It is proven effective for training ContinuousCoT methods as it does not contain any natural language but only structured mathematical expressions leading to the final answer such as $\ll 18.5 - 8.5 = 10 \gg \ll 20 + 15 + 30 = 65 \gg$. We also use **GSM8k-Aug-NL**, a version that preserves natural language explanations. As each sentence perfectly corresponds to one reasoning step in structured trace, we term this dataset as Semi-Natural. For experiments with realistic CoT traces, we use challenging **MATH** (Hendrycks et al., 2021) problems and generate CoT traces from the target model being trained. For evaluation, we use the test splits of GSM8k to evaluate in-domain performance, and **GSM-Hard** (Gao et al., 2023), **SVAMP** (Patel et al., 2021) and **MultiArith** (Roy & Roth, 2015) to evaluate out-of-domain generalization on math tasks. We also use **MATH** test set for more complex natural language reasoning evaluation. See Section C in the appendix for example traces.

Baselines. We consider the following baselines: (1) **CoT** (Chain-of-Thought) finetuning, Two indirect supervision methods: (2) **CODI** (Shen et al., 2025) and (3) **Sim-CoT** (Wei et al., 2026). Among Direct supervision methods, we compare with (4) **CoLaR** (Tan et al., 2026). These methods are discussed extensively in Section 2.2.

Training and Implementation Details. We experiment with two backbones: **Llama-3.2-1B-Instruct** (Grattafiori et al., 2024) and **Qwen2.5-1.5B-Instruct** (Qwen et al., 2025). Following prior works, all experiments use a trainable LoRA module (Hu et al., 2021) and are initialized with a trained CoT checkpoint. Full training and implementation details are provided in Section I (Appendix). For C-MTP, we use Warmstart Init for the main experiments, and Multitask Training for analysis in Section 3.2.1 to ensure a fair comparison with CODI. We present a detailed comparison of the two training configurations in Section H.1 (Appendix). We also perform an ablation without L_{KD} in Section H.2 (Appendix) and find that C-MTP is able to learn effectively even without distilling from a frozen teacher and Warmstart-Init is sufficient to retain most of the accuracy.

3.2. Training on Structured CoT: GSM8k-Aug

Tables 2 and 10 compares C-MTP against the baselines from Sections 2.2 and 2.3 on GSM8k-Aug, reporting downstream accuracy and chain-of-thought length. **C-MTP outperforms CoLaR** across all compression ratios: on LLaMA-3.2-1B-Instruct, C-MTP at 4× compression matches CoLaR at 2×, while converging in ∼5 epochs versus CoLaR's ∼50, highlighting C-MTP's stronger alignment with the pretraining objective. Across models, neither approach consistently dominates. On Llama, indirect supervision outperforms direct on in-distribution data but underperforms on out-of-distribution; on Qwen (See Section I.2), this pattern reverses. The lack of a consistent winner across both models and distribution settings suggests the two approaches are broadly comparable, with relative performance being model-dependent. Following sections explore potential reasons why direct supervision can be preferable.

3.2.1. Simplicity, Efficiency, and Generalization

Fixed Wall-Clock Time Comparison. Our analysis in Section D in the appendix reveals that a single C-MTP training step is approximately 2× faster than CODI on LLaMA-3.2-1B. C-MTP outperforms CODI across all budget levels (Table 11). Full results are in Section I.4 (Appendix).

Sample Efficiency. Figure 4 in the appendix reports the performance of C-MTP (Direct) and CODI (Indirect) under

¶Values taken from Shen et al. (2025).

Table 2. **GSM8k-Aug results on LLaMA-3.2-1B-Instruct.** Accuracy is averaged over 5 runs (run-to-run deviation <1%). #L denotes the mean chain-of-thought length (in steps) with standard deviation, computed over the full evaluation set.

Training Method		GSM8k-Test		GSM-Hard		SVAMP		MultiArith	
		Acc. (↑)	# L (↓)	Acc. (↑)	# L (↓)	Acc. (↑)	# L (↓)	Acc. (↑)	# L (↓)
Standard	CoT	61.94	$27.17_{\pm 11.24}$	14.22	$34.43_{\pm 15.11}$	64.96	$13.15_{\pm 5.30}$	98.89	$15.46_{\pm 1.96}$
Indirect Supervision	Coconut[¶]	45.30	6.00	9.9	6.00	48.80	6.00	90.10	6.00
	CODI	53.03	6.00	12.40	6.00	60.26	6.00	**99.11**	6.00
	Sim-CoT	**53.95**	6.00	12.45	**6.00**	58.62	6.00	97.78	6.00
Direct Supervision	CoLaR-2	41.23	$12.99_{\pm 5.53}$	8.87	$16.30_{\pm 8.30}$	52.46	$6.31_{\pm 2.66}$	90.56	$7.39_{\pm 1.43}$
	CoLaR-3	32.65	$9.01_{\pm 3.75}$	6.79	$10.96_{\pm 6.30}$	48.36	$4.49_{\pm 1.91}$	87.33	$5.33_{\pm 1.93}$
	CoLaR-4	29.60	$6.80_{\pm 2.71}$	6.60	$7.66_{\pm 2.97}$	46.82	$3.47_{\pm 1.48}$	88.11	$4.13_{\pm 0.67}$
	C-MTP-2	53.30	$12.07_{\pm 5.53}$	**12.59**	$14.15_{\pm 6.44}$	**62.10**	$5.14_{\pm 2.58}$	95.33	$6.29_{\pm 1.20}$
	C-MTP-3	46.50	$7.94_{\pm 3.69}$	10.58	$9.02_{\pm 4.54}$	56.08	$3.43_{\pm 1.66}$	97.33	$4.16_{\pm 0.67}$
	C-MTP-4	40.70	$\mathbf{5.84_{\pm 2.71}}$	9.29	$6.85_{\pm 4.34}$	50.98	$\mathbf{2.39_{\pm 1.47}}$	94.67	$\mathbf{3.12_{\pm 0.56}}$

varying amounts of finetuning samples (12k, 25k, 50k, 100k and 200k) from GSM8k-Aug across various downstream tasks. On in-distribution tasks (GSM8k-Test, GSM-Hard), C-MTP's advantage is particularly clear, whereas on out-of-distribution tasks (SVAMP, MultiArith) the gap is smaller and more variable. Full details on experiments are provided in Section I.3 in the appendix.

Generalization to Problem Structures. GSM8k-Aug (Deng et al., 2024) was constructed by prompting GPT-4 to generate problems similar to GSM8k, thereby expanding the dataset $\sim 50\times$ via prompting and heuristic filtering. This process results in substantial structural redundancy: our analysis shows that over 50% of all training samples reduce to only 100 distinct problem structures. To evaluate out-of-domain generalization to problem structures not seen during training, we partition the data by held-out problem structures. We present more analysis in Section E (Appendix). Table 12 (Appendix) reports in-domain and out-of-domain accuracy for C-MTP and CODI across reasoning lengths $n \in \{2, 3, 4\}$. C-MTP generalizes better to held-out problem structures in all settings with the gap being particularly clear at $n = 3$ and $n = 4$ reasoning lengths, where CODI's out-of-domain accuracy drops substantially (e.g., 27.10 and 13.83 on $n = 3$ and $n = 4$ respectively) while C-MTP retains stronger performance (42.86 and 24.04). Full details on experiments are provided in Section I.5 (Appendix).

3.3. Training on Semi-Natural CoT: GSM8k-Aug-NL

Table 3 compares C-MTP and baselines on GSM8k-Aug-NL, which contains verbose natural language reasoning traces averaging ∼70 steps — roughly 2.5× longer than GSM8k-Aug. **C-MTP outperforms CoLaR** across all compression ratios, consistent with Section 3.2. More strikingly, the gap between Indirect and Direct Supervision widens considerably relative to the structured setting: CODI achieves 47.82% on GSM8k-Test with just 6 latent steps, while CoLaR-2 collapses to 23.46%, and even C-MTP-2, despite using ∼34 steps, reaches only 43.49%.

Two factors explain why verbose data disproportionately hurts parallel methods. First, indirect supervision is inherently suited to compression: a fixed latent budget forces the model to discard filler tokens and irrelevant content, whereas parallel methods must predict the full verbose trace without any such pressure. Second, teacher forcing causes a train-test mismatch: at inference, the model conditions on its own prior predictions rather than ground-truth CoT traces, with errors cascading particularly severely through math expressions. We provide more analysis of the C-MTP generations on structured and semi-natural data in Section F.

3.4. Training on Realistic CoT: MATH

To evaluate how direct and indirect supervision scale to realistic CoT traces, we compare the strongest representative from each regime: CODI for indirect supervision and C-MTP for direct supervision. Since our earlier experiments already identify the best method in each regime, and realistic traces are expensive to train on, we use a single representative per regime. We generate answers to all problems in the train set, filter out wrong answers and use the correct generations for training. We finetune CoT-SFT, C-MTP-2, CODI-6/10/20 to compare discrete CoT, direct and indirect supervision methods. In Table 4, we see that both methods drastically fail achieving only ∼12-15% accuracy compared to ∼53% for CoT-SFT for Qwen2.5-1.5B-Instruct. Experiments with Llama show similar trends. Although CODI's reasoning traces are not interpretable, analysis of traces generated by C-MTP underline similar issues of error accumulation seen in semi-natural tasks. The model generates the right natural language tokens but makes arithmetic errors, drops signs or numbers or produces malformed expressions, which leads to an irrecoverable error in the trace and an incorrect final answer, suggesting that future ability to im-

Table 3. **Training on Semi-Natural CoT: GSM8k-Aug-NL**

Training Method		GSM8k-Test		GSM-Hard		SVAMP		MultiArith	
		Acc. (↑)	# L (↓)	Acc. (↑)	# L (↓)	Acc. (↑)	# L (↓)	Acc. (↑)	# L (↓)
Standard	CoT	53.68	$70.04_{\pm 30.02}$	13.04	$78.38_{\pm 33.01}$	60.88	$36.77_{\pm 18.52}$	96.33	$41.64_{\pm 10.82}$
Indirect Supervision	CODI	47.82	6.00	10.99	6.00	56.82	6.00	95.56	6.00
	Sim-CoT	44.35	6.00	10.11	6.00	55.16	6.00	97.89	6.00
Direct Supervision	CoLaR-2	23.46	$35.63_{\pm 14.46}$	5.43	$38.99_{\pm 15.02}$	37.70	$19.15_{\pm 9.48}$	72.22	$20.69_{\pm 7.59}$
	CoLaR-3	19.88	$23.31_{\pm 10.01}$	4.61	$25.55_{\pm 10.65}$	33.86	$12.14_{\pm 5.56}$	67.67	$13.86_{\pm 3.82}$
	CoLaR-4	18.06	$18.15_{\pm 8.28}$	3.75	$20.36_{\pm 9.33}$	33.16	$9.43_{\pm 4.51}$	63.89	$10.96_{\pm 4.68}$
	C-MTP-2	43.49	$34.00_{\pm 15.04}$	10.13	$37.32_{\pm 15.87}$	56.60	$16.53_{\pm 8.83}$	91.22	$18.74_{\pm 5.04}$
	C-MTP-3	34.04	$22.74_{\pm 10.18}$	7.73	$24.64_{\pm 10.78}$	48.24	$10.57_{\pm 5.57}$	80.33	$12.42_{\pm 3.48}$
	C-MTP-4	25.60	$17.13_{\pm 7.79}$	6.20	$19.00_{\pm 9.25}$	38.78	$7.98_{\pm 4.16}$	67.11	$9.26_{\pm 2.89}$

Table 4. Results on MATH across two backbones. Models that perform best using greedy decoding report SD of 0.0.

Training		Acc. (↑)	# L (↓)
Llama-3.2-1B-Instruct			
Standard	Original	$24.00_{\pm 0.00}$	$826.60_{\pm 1464.90}$
	CoT-SFT	$23.80_{\pm 0.00}$	$350.10_{\pm 271.15}$
Direct	C-MTP-2	$7.40_{\pm 0.00}$	$163.62_{\pm 106.74}$
Indirect	CODI-6	$8.52_{\pm 0.20}$	6.00
	CODI-10	$9.92_{\pm 0.20}$	10.00
	CODI-20	$7.92_{\pm 0.16}$	20.00
Qwen2.5-1.5B-Instruct			
Standard	Original	$53.52_{\pm 0.70}$	$644.90_{\pm 460.30}$
	CoT-SFT	$52.80_{\pm 0.00}$	$547.37_{\pm 281.63}$
Direct	C-MTP-2	$14.20_{\pm 0.00}$	$272.11_{\pm 124.83}$
Indirect	CODI-6	$12.44_{\pm 0.15}$	6.00
	CODI-10	$14.52_{\pm 0.20}$	10.00
	CODI-20	$12.96_{\pm 0.50}$	20.00

prove such methods may require a combination of the two. Direct supervision may be essential for its efficiency and alignment with the token-based training objective and indirect supervision may be essential for its ability to compress long traces into fixed latent budgets and avoid train-test mismatch.

4. Limitations and Future Work

Our work centers on two popular LLMs in the 1-1.5B size range evaluated on mathematical reasoning tasks, which may limit the generality of our findings. We focus on math because prior work has shown that latent reasoning methods struggle to match CoT in this domain, making it a natural testbed. Extending the evaluation of C-MTP to tasks such as code, commonsense, and multi-hop question answering is a natural next step. More fundamentally, our realistic-trace results suggest that neither direct nor indirect supervision alone suffices: a promising direction is hybrid approaches that adaptively switch between token-based and continuous reasoning (Shi et al., 2026; Xu et al., 2026; Zhu et al., 2026), which may be essential for scaling ContinuousCoT to realistic settings.

5. Conclusion

We propose C-MTP, a direct supervision method using multi-token prediction to simulate ContinuousCoT, which substantially outperforms comparable approaches. Through systematic analysis of direct and indirect supervision across Structured, Semi-Natural, and Realistic CoT traces, we find: direct supervision (C-MTP) is preferable on structured tasks for its efficiency; indirect supervision dominates on verbose reasoning tasks, better compressing long traces into fixed latent budgets where direct methods fail and face train-test mismatch; and on realistic traces, both regimes fall far short of CoT-SFT (Figure 1). Our results suggest that structured and semi-natural training significantly overstates the effectiveness of Continuous CoT methods.

Acknowledgements

This work was supported in part through the NYU IT High Performance Computing resources, services, and staff expertise. The work is partially funded by NSF CAREER award 2443271 and NSF award RI-2521091.

References

Butt, N., Kwiatkowski, A., Labiad, I., Kempe, J., and Ollivier, Y. Soft tokens, hard truths, 2025. URL https://arxiv.org/abs/2509.19170.

Cai, T., Li, Y., Geng, Z., Peng, H., Lee, J. D., Chen, D., and Dao, T. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. In Salakhutdinov, R., Kolter, Z., Heller, K. A., Weller, A., Oliver, N., Scarlett, J., and Berkenkamp, F. (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, Proceedings of Machine Learning Research, pp. 5209–5235. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/cai24b.html.

Cheng, J. and Durme, B. V. Compressed chain of thought: Efficient reasoning through dense representations. *CoRR*, abs/2412.13171, 2024. doi: 10.48550/ARXIV.2412.13171. URL https://doi.org/10.48550/arXiv.2412.13171.

Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, Y., Wang, X., Dehghani, M., Brahma, S., Webson, A., Gu, S. S., Dai, Z., Suzgun, M., Chen, X., Chowdhery, A., Castro-Ros, A., Pellat, M., Robinson, K., Valter, D., Narang, S., Mishra, G., Yu, A., Zhao, V. Y., Huang, Y., Dai, A. M., Yu, H., Petrov, S., Chi, E. H., Dean, J., Devlin, J., Roberts, A., Zhou, D., Le, Q. V., and Wei, J. Scaling instruction-finetuned language models. *J. Mach. Learn. Res.*, 25:70:1–70:53, 2024. URL https://jmlr.org/papers/v25/23-0870.html.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168, 2021. URL https://arxiv.org/abs/2110.14168.

DeepSeek-AI and team. Deepseek-v3 technical report, 2025. URL https://arxiv.org/abs/2412.19437.

Deng, Y., Choi, Y., and Shieber, S. M. From explicit cot to implicit cot: Learning to internalize cot step by step. *CoRR*, abs/2405.14838, 2024. doi: 10.48550/ARXIV.2405.14838. URL https://doi.org/10.48550/arXiv.2405.14838.

Fu, T., You, Y., Chen, Z., Dai, G., Yang, H., and Wang, Y. Think-at-hard: Selective latent iterations to improve reasoning language models. *CoRR*, abs/2511.08577, 2025. doi: 10.48550/ARXIV.2511.08577. URL https://doi.org/10.48550/arXiv.2511.08577.

Gao, L., Madaan, A., Zhou, S., Alon, U., Liu, P., Yang, Y., Callan, J., and Neubig, G. PAL: Program-aided language models. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10764–10799. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/gao23f.html.

Geiping, J., McLeish, S., Jain, N., Kirchenbauer, J., Singh, S., Bartoldson, B. R., Kailkhura, B., Bhatele, A., and Goldstein, T. Scaling up test-time compute with latent reasoning: A recurrent depth approach. *CoRR*, abs/2502.05171, 2025. doi: 10.48550/ARXIV.2502.05171. URL https://doi.org/10.48550/arXiv.2502.05171.

Gloeckle, F., Idrissi, B. Y., Rozière, B., Lopez-Paz, D., and Synnaeve, G. Better & faster large language models via multi-token prediction. In Salakhutdinov, R., Kolter, Z., Heller, K. A., Weller, A., Oliver, N., Scarlett, J., and Berkenkamp, F. (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, Proceedings of Machine Learning Research, pp. 15706–15734. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/gloeckle24a.html.

Goyal, S., Ji, Z., Rawat, A. S., Menon, A. K., Kumar, S., and Nagarajan, V. Think before you speak: Training language models with pause tokens. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=ph04CRkPdC.

Gozeten, H. A., Ildiz, M. E., Zhang, X., Harutyunyan, H., Rawat, A. S., and Oymak, S. Continuous chain of thought enables parallel exploration and reasoning. *CoRR*, abs/2505.23648, 2025. doi: 10.48550/ARXIV.2505.23648. URL https://doi.org/10.48550/arXiv.2505.23648.

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A., Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., and et al, A. R. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv e-prints*, pp. arXiv–2501, 2025. URL https://arxiv.org/abs/2501.12948.

Hao, S., Sukhbaatar, S., Su, D., Li, X., Hu, Z., Weston, J. E., and Tian, Y. 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.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2021.

Hsieh, C., Li, C., Yeh, C., Nakhost, H., Fujii, Y., Ratner, A., Krishna, R., Lee, C., and Pfister, T. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In Rogers, A., Boyd-Graber, J. L., and Okazaki, N. (eds.), *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, Findings of ACL, pp. 8003–8017. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.FINDINGS-ACL.507. URL https://doi.org/10.18653/v1/2023.findings-acl.507.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., and Iwasawa, Y. Large language models are zero-shot reasoners. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *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*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/8bb0d291acd4acf06ef112099c16f326-Abstract-Conference.html.

Merrill, W. and Sabharwal, A. The expressive power of transformers with chain of thought. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=NjNGlPh8Wh.

OpenAI. Openai o1 system card, 2026. URL https://arxiv.org/abs/2412.16720.

Patel, A., Bhattamishra, S., and Goyal, N. Are NLP models really able to solve simple math word problems? In Toutanova, K., Rumshisky, A., Zettlemoyer, L., Hakkani-Tur, D., Beltagy, I., Bethard, S., Cotterell, R., Chakraborty, T., and Zhou, Y. (eds.), *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/.

Pfau, J., Merrill, W., and Bowman, S. R. Let's think dot by dot: Hidden computation in transformer language models. *CoRR*, abs/2404.15758, 2024. doi: 10.48550/ARXIV.2404.15758. URL https://doi.org/10.48550/arXiv.2404.15758.

Qwen, :, Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., Lin, H., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Lin, J., Dang, K., Lu, K., Bao, K., Yang, K., Yu, L., Li, M., Xue, M., Zhang, P., Zhu, Q., Men, R., Lin, R., Li, T., Tang, T., Xia, T., Ren, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Wan, Y., Liu, Y., Cui, Z., Zhang, Z., and Qiu, Z. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Roy, S. and Roth, D. Solving general arithmetic word problems. In Màrquez, L., Callison-Burch, C., and Su, J. (eds.), *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1743–1752, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1202. URL https://aclanthology.org/D15-1202/.

Saparov, A. and He, H. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=qFVVBzXxR2V.

Saunshi, N., Dikkala, N., Li, Z., Kumar, S., and Reddi, S. J. Reasoning with latent thoughts: On the power of looped transformers. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=din0lGfZFd.

Shao, C., Li, D., Meng, F., and Zhou, J. Continuous autoregressive language models, 2025. URL https://arxiv.org/abs/2510.27688.

Shen, Z., Yan, H., Zhang, L., Hu, Z., Du, Y., and He, Y. CODI: compressing chain-of-thought into continuous space via self-distillation. In Christodoulopoulos, C., Chakraborty, T., Rose, C., and Peng, V. (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025, Suzhou, China, November 4-9, 2025*, pp. 677–693. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.EMNLP-MAIN.36. URL https://doi.org/10.18653/v1/2025.emnlp-main.36.

Shi, D., Asi, A., Li, K., Yuan, X., Pan, L., Lee, W., and Xiao, W. Swireasoning: Switch-thinking in latent and explicit for pareto-superior reasoning llms, 2026. URL https://arxiv.org/abs/2510.05069.

Tan, W., Li, J., Ju, J., Luo, Z., Song, R., and Luan, J. Think silently, think fast: Dynamic latent compression of LLM reasoning chains. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=AQsko3PPUe.

Tang, Y., Dong, L., Hao, Y., Dong, Q., Wei, F., and Gu, J. Multiplex thinking: Reasoning via token-wise branch-and-merge. *CoRR*, abs/2601.08808, 2026. doi: 10.48550/ARXIV.2601.08808. URL https://doi.org/10.48550/arXiv.2601.08808.

Wang, L., Xu, W., Lan, Y., Hu, Z., Lan, Y., Lee, R. K.-W., and Lim, E.-P. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In Rogers, A., Boyd-Graber, J., and Okazaki, N. (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2609–2634, Toronto, Canada, July 2023a. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.147. URL https://aclanthology.org/2023.acl-long.147/.

Wang, X., Wei, J., Schuurmans, D., Le, Q. V., Chi, E. H., Narang, S., Chowdhery, A., and Zhou, D. 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, 2023b. URL https://openreview.net/forum?id=1PL1NIMMrw.

Wei, J., Bosma, M., Zhao, V. Y., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., and Le, Q. V. Finetuned language models are zero-shot learners. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022a. URL https://openreview.net/forum?id=gEZrGCozdqR.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E. H., Le, Q. V., and Zhou, D. Chain-of-thought prompting elicits reasoning in large language models. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *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*, 2022b. URL http://papers.nips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Confere html.

Wei, X., Liu, X., Zang, Y., Dong, X., Cao, Y., Wang, J., Qiu, X., and Lin, D. SIM-cot: Supervised implicit chain-of-thought. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=6YRJ4jmVQl.

Wu, H., Teng, Z., and Tu, K. Parallel continuous chain-of-thought with jacobi iteration. In Christodoulopoulos, C., Chakraborty, T., Rose, C., and Peng, V. (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 914–926, Suzhou, China, November 2025a. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.47. URL https://aclanthology.org/2025.emnlp-main.47/.

Wu, J., Lu, J., Ren, Z., Hu, G., Wu, Z., Dai, D., and Wu, H. Llms are single-threaded reasoners: Demystifying the working mechanism of soft thinking, 2025b. URL https://arxiv.org/abs/2508.03440.

Xu, K. and Sato, I. A formal comparison between chain-of-thought and latent thought. *CoRR*, abs/2509.25239, 2025. doi: 10.48550/ARXIV.2509.25239. URL https://doi.org/10.48550/arXiv.2509.25239.

Xu, X., Yu, T., Chen, X., Wang, H., McAuley, J., and Mitra, S. Thinkrouter: Efficient reasoning via routing thinking between latent and discrete spaces, 2026. URL https://arxiv.org/abs/2602.11683.

Yue, Z., Jin, B., Zeng, H., Zhuang, H., Qin, Z., Yoon, J., Shang, L., Han, J., and Wang, D. Hybrid latent reasoning via reinforcement learning, 2025. URL https://arxiv.org/abs/2505.18454.

Zelikman, E., Wu, Y., Mu, J., and Goodman, N. D. Star: Bootstrapping reasoning with reasoning. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *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*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/639a9a172c044fbb64175b5fad42e9a5-Abstract-Confere html.

Zeng, B., Song, S., Huang, S., Wang, Y., Li, H., He, Z., Wang, X., Li, Z., and Lin, Z. Ponderlm: Pretraining language models to ponder in continuous space, 2026. URL https://arxiv.org/abs/2505.20674.

Zhang, J., Zhu, Y., Sun, M., Luo, Y., Qiao, S., Du, L., Zheng, D., Chen, H., and Zhang, N. Lightthinker: Thinking step-by-step compression, 2025a. URL https://arxiv.org/abs/2502.15589.

Zhang, Z., Zhang, A., Li, M., and Smola, A. Automatic chain of thought prompting in large language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=5NTt8GFjUHkr.

Zhang, Z., He, X., Yan, W., Shen, A., Zhao, C., Wang, S., Shen, Y., and Wang, X. E. Soft thinking: Unlocking the reasoning potential of llms in continuous concept space. *CoRR*, abs/2505.15778, 2025b. doi: 10.48550/ARXIV.2505.15778. URL https://doi.org/10.48550/arXiv.2505.15778.

Zhao, G., Xie, R., Wang, A., Li, S., Xie, H., and Sun, X. Self-distillation for multi-token prediction. *CoRR*, abs/2603.23911, 2026. doi: 10.48550/ARXIV.2603.23911. URL https://doi.org/10.48550/arXiv.2603.23911.

Zhou, D., Schärli, N., Hou, L., Wei, J., Scales, N., Wang, X., Schuurmans, D., Cui, C., Bousquet, O., Le, Q. V., and Chi, E. H. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=WZH7099tgfM.

Zhu, H., Hao, S., Hu, Z., Jiao, J., Russell, S., and Tian, Y. Reasoning by superposition: A theoretical perspective on chain of continuous thought. *CoRR*, abs/2505.12514, 2025a. doi: 10.48550/ARXIV.2505.12514. URL https://doi.org/10.48550/arXiv.2505.12514.

Zhu, R., Wang, Z., Hua, K., Zhang, T., Li, Z., Que, H., Wei, B., Wen, Z., Yin, F., Xing, H., Li, L., Shi, J., Ma, K., Li, S., Kergan, T., Smith, A., Qu, X., Hui, M., Wu, B., Min, Q., Huang, H., Zhou, X., Ye, W., Liu, J., Yang, J., Shi, Y., Lin, C., Zhao, E., Cai, T., Zhang, G., Huang, W., Bengio, Y., and Eshraghian, J. Scaling latent reasoning via looped language models. *CoRR*, abs/2510.25741, 2025b. doi: 10.48550/ARXIV.2510.25741. URL https://doi.org/10.48550/arXiv.2510.25741.

Zhu, Y., Zhang, J., Wan, Z., Luo, Y., Qiao, S., Gui, Z., Zheng, D., Liang, L., Chen, H., and Zhang, N. Lightthinker++: From reasoning compression to memory management, 2026. URL https://arxiv.org/abs/2604.03679.

Zou, J., Xiong, Y., and Liu, Y. Capabilities and fundamental limits of latent chain-of-thought. *CoRR*, abs/2602.01148, 2026. doi: 10.48550/ARXIV.2602.01148. URL https://doi.org/10.48550/arXiv.2602.01148.

Table of Contents

A. Related Works

B. MTP Architecture

C. Example of CoT Traces for Three Difficulties

D. Timing Analysis for Fixed Wallclock Time Experiments

E. Analysis of Problem Structures in GSM8k-Aug

F. Analysis of C-MTP Generations

G. Examples of Error Accumulation during MTP Inference

H. C-MTP Ablation Studies

- **H.1** Multi-Task Training vs Warmstart Initialization
- **H.2** Ablation on C-MTP: Training without L_{KD}

I. Extended Experimental Results

- **I.1** Setup Details
- **I.2** Training on Structured CoT: GSM8k-Aug
- **I.3** Sample Efficiency
- **I.4** Fixed WallClock Time Comparison
- **I.5** Generalization to Problem Structures
- **I.6** Full Example of MTP Error Accumulation

J. CoLaR - CODI Implementation Discrepancy

A. Related Works

Explicit Chain-of-Thought (CoT) (Wei et al., 2022b; Kojima et al., 2022) improves self-consistency (Wang et al., 2023b), simplifies complex problem solving through decomposition (Zhou et al., 2023), facilitates planning (Wang et al., 2023a), automates demonstration generation (Zhang et al., 2023), and enhances supervised fine-tuning (Wei et al., 2022a; Chung et al., 2024; Zelikman et al., 2022; Hsieh et al., 2023). These advancements have culminated in recent "thinking" models (Guo et al., 2025; OpenAI, 2026).

Continuous CoT, or latent reasoning, encompasses diverse approaches. Coconut (Hao et al., 2025) pioneered autoregressive generation with latents, inspiring subsequent research. Formal analyses (Zhu et al., 2025a; Zou et al., 2026; Xu & Sato, 2025) demonstrate its theoretical advantages over explicit CoT in specific scenarios. In this context, *latents* can encode multiple solution trajectories (Gozeten et al., 2025; Zhang et al., 2025b; Wu et al., 2025b; Tang et al., 2026; Butt et al., 2025), provide additional compute via filler tokens (Pfau et al., 2024; Goyal et al., 2024), or facilitate recurrent looping (Saunshi et al., 2025; Fu et al., 2025; Zeng et al., 2026; Zhu et al., 2025b; Geiping et al., 2025).

In this work, we utilize latents to encode intermediate reasoning steps through both indirect (Shen et al., 2025; Wei et al., 2026) and direct supervision (Tan et al., 2026; Cheng & Durme, 2024). Concurrently, the field is advancing novel training architectures (Wu et al., 2025a; Shao et al., 2025) and hybrid models that adaptively switch between latent reasoning and token-based generation (Shi et al., 2026; Yue et al., 2025; Xu et al., 2026; Zhu et al., 2026; Zhang et al., 2025a). Our work further underscores the importance of this adaptivity.

Finally, while Multi-Token Prediction (MTP) improves pretraining (DeepSeek-AI & team, 2025), accelerates speculative decoding (Cai et al., 2024), and can be learned via self-distillation (Zhao et al., 2026), it remains largely unexplored in the context of latent reasoning.

B. MTP Architecture

Standard LLMs generate a single probability distribution over $\mathcal{V}$. To generate multiple distributions from the same backbone, we utilize a small MLP to project the final layer activation into activations corresponding to multiple tokens. We then use the same output embedding matrix to generate logits for all tokens.

At each thinking step i, the hidden state is obtained by passing the prompt and all predicted latents through the model:

$$h_i = H_\theta(E(Q),\, z_1, \ldots, z_{i-1}) \in \mathbb{R}^d \tag{11}$$

A 2-layer MLP: $g \in \mathbb{R}^{d \times sd}$ maps this to s token representations simultaneously:

$$O_i = \text{reshape}(g(h_i)) \in \mathbb{R}^{s \times d} \tag{12}$$

Row $O_i[j]$ is the predicted representation for the j-th token of the next span C_{i+1}. Logits are obtained by projecting through the output embedding matrix $E \in \mathbb{R}^{|\mathcal{V}| \times d}$:

$$P_\theta(C_{i+1}[j] \mid E(Q), z_{1:i}) = \text{softmax}(O_i[j] \cdot E'\ ) \tag{13}$$

A single forward pass thus produces all s token distributions in parallel, with no sequential decoding within a span. We then sample one token from each distribution to form the next span C_{i+1}, and aggregate their embeddings to obtain the next latent z_{i+1} via Equation (7).

C. Example of CoT Traces for Three Difficulties

Bella bought 18 packs of chocolates each containing 12 chocolates. If Bella gives away 6 packs, how many dozens of chocolates will she have left?

Realistic Trace generated from Qwen2.5-1.5B-Instruct.

Structured	Semi-Natural	Realistic
≪18*12=216≫ ≪216-6*12=144≫ ≪144/12=12≫	Bella initially had 18 packs * 12 chocolates/pack = 216 chocolates. After giving away chocolates, Bella has 216 chocolates - 6 packs * 12 chocolates/pack = 144 chocolates left. Thus, Bella is left with 144 chocolates / 12 = 12 dozens of chocolates.	To determine how many dozens of chocolates Bella has left after giving some away, we can follow these steps: **1. Calculate the total number of chocolates Bella initially had:** - Each pack contains 12 chocolates. - Bella bought 18 packs. - Therefore, the total number of chocolates is: 18 packs $\times$ 12 chocolates per pack $=$ 216 chocolates **2. Determine the number of chocolates in the packs Bella gave away:** - Bella gave away 6 packs. - Each pack contains 12 chocolates. - Therefore, the number of chocolates given away is: 6 packs $\times$ 12 chocolates per pack $=$ 72 chocolates **3. Calculate the number of chocolates remaining:** - Subtract the number of chocolates given away from the initial total: 216 chocolates $-$ 72 chocolates $=$ 144 chocolates **4. Convert the remaining chocolates into dozens:** - There are 12 chocolates in a dozen. - To find out how many dozens of chocolates remain, divide the number of chocolates by 12: \frac{144 \text{ chocolates}}{12 \text{ chocolates per dozen}} = 12 \text{ dozens} Thus, the number of dozens of chocolates Bella has left is \(\boxed{12}\)

Table 5. CoT Trace Examples

D. Timing Analysis for Fixed Wallclock Time Experiments

To ensure a fair comparison for our fixed wallclock time experiments, we measure the latency of a single training step (one batch forward and backward) for both architectures. We benchmark our direct supervision method (MTP) using a span length of 4 against indirect supervision (CODI) using 6 latent steps. These configurations were selected as they result in similar compression ratios on the GSM8k-Aug dataset. All timing evaluations are conducted on an H200 GPU, with each sequence partitioned into a 60-30-10 ratio token split for the question, CoT, and answer, respectively. Model configurations are detailed in Table 6.

Table 6. Latency of a single training pass (forward + backward) on NVIDIA H200

Model Config	Batch Size	CODI (ms)	MTP (ms)	Overhead
LLaMA-3.2-1B-Instruct	16	821.65 ± 24.25	343.10 ± 1.02	$2.39\times$
Seq. Len.: 128	32	1018.86 ± 23.48	630.26 ± 1.50	$1.62\times$
Qwen2.5-1.5B-Instruct	16	1125.74 ± 8.80	305.82 ± 1.22	$3.68\times$
Seq. Len.: 180	32	1219.28 ± 8.83	552.03 ± 0.60	$2.21\times$

At this scale, CODI requires approximately $\mathbf{2}\times$ the training time of MTP for a batch size of 16. While one might reasonably expect a $6\times$ overhead for CODI due to the requirement of 6 sequential forward passes for the latent steps, KV caching during training significantly accelerates this computation. As a result, the observed $2\times$ overhead is primarily bottlenecked by the initial forward pass for the question sequence and the final forward pass for the answer sequence. In contrast, our MTP approach computes the forward pass for all positions simultaneously, as the outputs at all positions are known during training.

MTP training demonstrates compute-bound characteristics (similar to standard NTP training), as the step latency scales almost linearly (nearly doubling from 343.10 ms to 630.26 ms) when the batch size is doubled from 16 to 32. Conversely, CODI exhibits memory-bound behavior, showing a distinctly sub-linear increase in latency across the same batch sizes. This indicates that a substantial portion of CODI's training time is consumed by memory overhead and the sequential memory accesses required for latent computations. Consequently, this leads to an under-utilization of GPU compute resources during training. The memory overhead is dominated by the need to update the KVCache states, consequently this is larger for Qwen2.5-1.5B due to larger hidden dimensions.

E. Analysis of Problem Structures in GSM8k-Aug

As the dataset consists of expressions in the CoT traces, we parse the expressions into templates to analyze the diversity of problem structures. Some examples of templates are given below:

Count in Train: 24353 **Template: a/b=r0**
Q: A school has 3 classrooms, each with the same number of students. If there are 36 students in total, how many students are there in each classroom?
CoT: ≪36/3=12≫

Count in Train: 22451 **Template: a*b=r0**
Q: Jacob is building a dog house. He uses 8 nails for each plank and needs 6 planks. How many nails will he need in total?
CoT: ≪8*6=48≫

Count in Train: 14989 **Template: a*b=r0, r0*c=r1**
Q: A basketball team consists of 12 players. Each player needs two basketball shoes. The cost of each shoe is $75. How much will it cost for the basketball team to buy shoes?
CoT: ≪12*2=24≫ ≪24*75=1800≫

Cumulative coverage Figure 2 shows that 80% of the dataset consists of 3k structures which is less than 1% the size of the dataset.

We find that top 10, 50, 100, 500 structures cover 28.3%, 46.5%, 53.3% and 66.5% of the 380k samples of GSM8k-Aug which is a considerable portion. We hence argue that this dataset does not reflect real world diversity of typical SFT finetuning.

Figure 2. Cumulative coverage of top-n problem structures in GSM8k-Aug

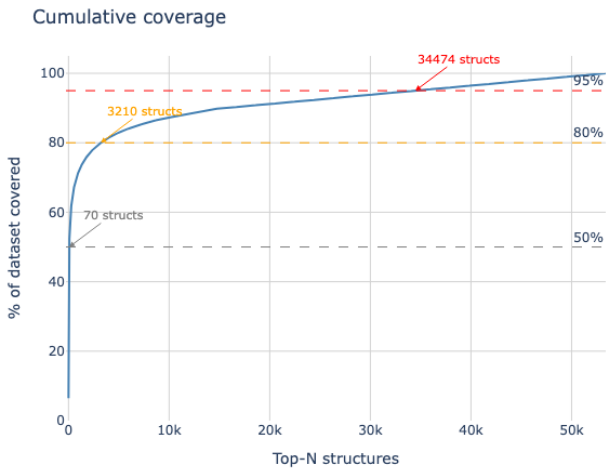
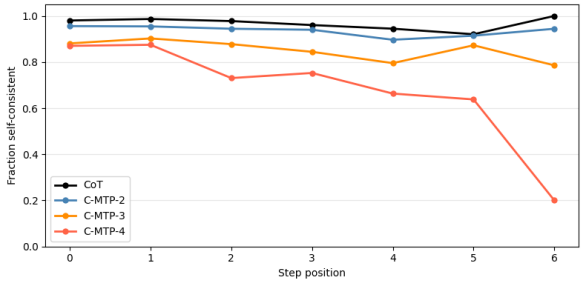


Figure 3. Fraction of self-consistent expressions by absolute position in C-MTP-X and CoT traces. We only plot for samples for which CoT arrives at the right answer. Trained on GSM8k-Aug, Eval: GSM8k-Test.



F. Analysis of C-MTP Generations

Training on GSM8k-Aug teaches models to reason only using expressions relevant to computing the final answer. We compute the mathematical validity of each produced expression for each sample and plot the average fraction of valid expressions at each position in the generated CoT trace for both C-MTP and CoT in Figure 3. We see that C-MTP starts with a high fraction at the beginning of the trace and falls off towards the end, especially for $s = 4$ where the model is forced to compress more. We further analyze generations upon training on GSM8k-Aug-NL, where the traces contain both math expressions and natural language. We find that the model gets things wrong primarily in math expressions (See examples in Section G) especially when a span boundary occurs in between. Upon an incorrect generation the model is forced to condition on its own erroneous output (error accumulation), which leads to an incorrect answer. We detail a few selected examples in Section I.6 (Appendix) to illustrate this.

G. Examples of Error Accumulation during MTP Inference

A Collection of all wrong arithmetic expressions on Semi-Natural CoT

500000*.150=150 ; **3*3=99** ; 75-15=15 ; 20-1=10 ; 1010*1=10 ; 16/3=3 ; 20+2=60 ; 4*4=160 ; 10.2*10=24 ; 24*45=108108 ; 30/100=54 ; 180+324=324 ; 180-0.3=120 ; 180+120=360 ; 2*8080=160 ; 6*55=330330 ; 90/4=9 ; 7*7=1 ; 90/31=2 ; 3*3=90 ; 25/100=4 ; 16-4=8 ; 20*20.2=4 ; 2020-4=16 ; 8000*1.2=960 ; **21/12=1** ; 1.1*75=4 ; 2+4=10 ; 6/44=1 ; 2/3=6 ; 3/5=9 ; .67+9=15 ; **3*2=6.67** ; 3*10=6 ; 5*15=9 ; .67+9=21 ; 23-25=8 ; 23-37=36 ; 2*2=8 ; 4+3=9 ; 55*2=10 ; 50/0.75=25 ; 75.50+126=292.5 ; 18*162=111 ; 11-18=5 ; 5*7=106 ; 18*162=108 ; 162-108=90 ; 2+20=40 ; 40+65=155 ; 155/3=51 ; 25*100=25 ; 33/3=23 ; 3/220=20.33 ; 40+85=225 ; 195-160=40 ; 2*8=26 ; 26/60=1 ; 3*3=1 ; 3+6=15 ; 3*6=2 ; 3*3=2 ; 3*6=4 ; 1/3=2 ; 2/3=3.3 ; .25+1.25=2.75 ; 0-100=1700 ; 400*3=120 ; 1200-1000=2000 ; 40-200=40 ; 400-10=290.00 ; 7+14=28 ; 5+7=19 ; 80+23=203 ; 203-23=80 ; 40/2=100 ; 200*40=80 ; 12/4=24 ; 15*15=120 ; 4*120=90 ; **4-1=33** ; 6*6=48 ; 18+12=25 ; 5000-3755=125 ; 3750-5000=3750 ; 25*40=10 ; 100*25=5 ; 4+4=9 ; 2525-9=16 ; 500+800=130 ;

Table 7. **Comparison of Warmstart Initialization (WS-Init) and Multitask Training (MultiTask-Train)**. with LLama3.2-1B-Instruct on GSM8k-Aug (Structured) CoT traces.

Method	GSM8k-Test		GSM-Hard		SVAMP		MultiArith	
	Acc. (↑)	# L (↓)	Acc. (↑)	# L (↓)	Acc. (↑)	# L (↓)	Acc. (↑)	# L (↓)
CoT	61.94	$27.17_{\pm 11.24}$	14.22	$34.43_{\pm 15.11}$	64.96	$13.15_{\pm 5.30}$	98.89	$15.46_{\pm 1.96}$
CODI	53.03	6.00	12.40	6.00	60.26	6.00	99.11	6.00
CoLaR-3	32.65	$9.01_{\pm 3.75}$	6.79	$10.96_{\pm 6.30}$	48.36	$4.49_{\pm 1.91}$	87.33	$5.33_{\pm 1.93}$
C-MTP-3 (WS-Init)	46.50	$7.94_{\pm 3.69}$	10.58	$9.02_{\pm 4.54}$	56.08	$3.43_{\pm 1.66}$	97.33	$4.16_{\pm 0.67}$
C-MTP-3 (MultiTask-Train)	48.29	$7.81_{\pm 3.56}$	11.11	$8.86_{\pm 4.05}$	55.02	$3.44_{\pm 1.68}$	96.11	$4.10_{\pm 0.58}$

1300+13=139 ; 1300+130=123 ; 000+25000=1 ; 126*60=756 ; 100*840=84 ; 840+756=149 ; 140+280=605 ; 140+340=625 ; 5*88=40 ; 2/100=15 ; 13.35+34.2=84 ; 128+256=512 ; 16/2=1 ; 1/2=8 ; 8+4=33 ; 4/3=8 ; 4+2=30 ; 740/110=6 ; 180/110=16 ; 4/4=100100 ; 2*100=50 ; 4+4=4 ; 24/66=4 ; 9+28=77 ; 125-2=510 ; 4-2=125 ; 23*23=92 ; 10/100=60 ; 10/100=66 ; 10/100=600 ; 600600+600=6600 ; 10/100=1320 ; 15400+6000=22000 ; 8800/88=1080 ; 75*4=3 ; 0*.110=22 ;

A lot of these mistakes, although they seem far from correct can be easily explained in the context of MTP.
3*2=6.67 has an extra prediction of ".67" due to 2-token prediction. The following CoT trace conditioned on this wrong output.
12/21=1 When 2-tokens are aggregated, both the numbers turned out to be the same, hence the result was 1.
3*3=99 has an extra prediction of "9" due to 2-token prediction.
and so on..

H. C-MTP Ablation Studies

In this section we present two ablation studies of C-MTP. The first study compares multi-task training with warmstart initialization, and the second ablates the necessity of L_{KD} by training without distillation loss.

H.1. Multi-Task Training vs Warmstart Initialization

We compare Multi-Task Training and Warmstart Initialization on full Structured CoT trace dataset Table 7. Warmstart Init and Multitask Training yield comparable accuracy (C-MTP-3: 46.50 vs. 48.29 on GSM8k-Test). Since C-MTP is a small addition on top of CoT, we view Warmstart Initialization as the more principled way but a fair comparison with CODI compelled us to experiment with Multi-task training setup. Our results show similar performance with both settings, hence we use Warmstart Init for our main table.

H.2. Ablation on C-MTP: Training without L_{KD}

In this section, we ablate the effect of using the knowledge distillation loss L_{KD} in our training setup. We compare the performance of our model with and without the L_{KD} term, while keeping all other training configurations constant. The results are summarized in Table 8. We observe that the performance difference is minimal, suggesting that the C-MTP training is able to learn effectively even without distilling from a frozen teacher. The warmstart initialization from the CoT checkpoint provides a strong starting point and the simplicity of the C-MTP training objective prevents catastrophic forgetting.

I. Extended Experimental Results

In this section we present additional details and results for the experiments described in the main paper.

I.1. Setup Details

Table 9 reports the training configuration for every run, following these conventions. *Data* denotes the training corpus: *Structured* (GSM8k-Aug, `icot` in code), *Semi-Nat.* (semi-natural language CoT, `icot-nl` in code), and *Realistic*

Table 8. **GSM8k-Aug results on LLaMA-3.2-1B-Instruct with no L_{KD} on C-MTP.** Accuracy is averaged over 5 runs (run-to-run deviation <1%). #L denotes the mean chain-of-thought length (in steps) with standard deviation, computed over the full evaluation set.

Training Method		GSM8k-Test		GSM-Hard		SVAMP		MultiArith	
		Acc. (↑)	# L (↓)	Acc. (↑)	# L (↓)	Acc. (↑)	# L (↓)	Acc. (↑)	# L (↓)
Standard	CoT	61.94	$27.17_{\pm 11.24}$	14.22	$34.43_{\pm 15.11}$	64.96	$13.15_{\pm 5.30}$	98.89	$15.46_{\pm 1.96}$
Supervision	CODI	53.03	6.00	12.40	6.00	60.26	6.00	**99.11**	6.00
	Sim-CoT	**53.95**	6.00	12.45	**6.00**	58.62	6.00	97.78	6.00
Direct Supervision	CoLaR-2	41.23	$12.99_{\pm 5.53}$	8.87	$16.30_{\pm 8.30}$	52.46	$6.31_{\pm 2.66}$	90.56	$7.39_{\pm 1.43}$
	CoLaR-3	32.65	$9.01_{\pm 3.75}$	6.79	$10.96_{\pm 6.30}$	48.36	$4.49_{\pm 1.91}$	87.33	$5.33_{\pm 1.93}$
	CoLaR-4	29.60	$6.80_{\pm 2.71}$	6.60	$7.66_{\pm 2.97}$	46.82	$3.47_{\pm 1.48}$	88.11	$4.13_{\pm 0.67}$
	C-MTP-2	53.30	$12.07_{\pm 5.53}$	**12.59**	$14.15_{\pm 6.44}$	**62.10**	$5.14_{\pm 2.58}$	95.33	$6.29_{\pm 1.20}$
	-L_{KD}	52.12	$12.10_{\pm 5.88}$	12.19	$14.39_{\pm 6.94}$	59.50	$5.27_{\pm 2.65}$	94.67	$6.23_{\pm 0.98}$
	C-MTP-3	46.50	$7.94_{\pm 3.69}$	10.58	$9.02_{\pm 4.54}$	56.08	$3.43_{\pm 1.66}$	97.33	$4.16_{\pm 0.67}$
	-L_{KD}	46.00	$7.92_{\pm 3.65}$	10.60	$8.99_{\pm 4.34}$	56.10	$3.43_{\pm 1.69}$	96.67	$4.17_{\pm 0.63}$
	C-MTP-4	40.70	$\mathbf{5.84_{\pm 2.71}}$	9.29	$6.85_{\pm 4.34}$	50.98	$\mathbf{2.39_{\pm 1.47}}$	94.67	$\mathbf{3.12_{\pm 0.56}}$
	-L_{KD}	36.60	$5.84_{\pm 2.66}$	8.79	$6.82_{\pm 3.22}$	48.70	$2.42_{\pm 1.51}$	94.78	$3.13_{\pm 0.51}$

(`mathllama`/`mathqwen` in code). The effective batch size is reported as (per-device batch $\times$ gradient-accumulation steps). All runs use a warmup ratio of 0.05, except the CODI runs, which use 0.03 following original implementation. All models are trained with LoRA adapters unless marked *Full* (full fine-tuning). The last column reports the C-MTP span length or the number of CODI latent tokens, where applicable; it is left blank otherwise.

All experiments were conducted on a single NVIDIA H200 GPU.

I.2. Training on Structured CoT: GSM8k-Aug

We present results on Qwen2.5-1.5B-Instruct in Table 10. Our experiments on training Sim-CoT with this backbone were rife with training instabilities leading to suboptimal downstream results. Therefore, we did not add them in the table.

I.3. Sample Efficiency

To understand how the two training paradigms compare under varying amounts of supervision, we train both methods on randomly sampled subsets of GSM8k-Aug (12k, 25k, 50k, 100k and 200k samples) and evaluate on downstream tasks. For fair comparison with CODI, we use Multitask-Train configuration where we train the same backbone for CoT and C-MTP and use the CoT inference output as the teacher. For each random subset size we run a grid search over $lr \in \{$5e-4, 7e-4, 8e-4$\}$ and $epochs \in \{6, 7, 10\}$ and select the configuration with final best validation loss. Figure 4 shows accuracy as a function of training set size. In the low-data regime (up to $\sim$100k samples), C-MTP consistently outperforms CODI, with the gap most pronounced at 12k samples. On in-distribution tasks (GSM8k-Test, GSM-Hard), C-MTP's advantage is particularly clear, whereas on out-of-distribution tasks (SVAMP, MultiArith) the gap is smaller and more variable.

I.4. Fixed WallClock Time Comparison

Our analysis (Table 6) reveals that a single C-MTP training step is approximately $2\times$ faster than CODI on LLaMA-3.2-1B, allowing C-MTP to process roughly twice as many samples within the same wall-clock budget. This $2\times$ factor represents a lower bound: for larger models like Qwen2.5-1.5B-Instruct, the sequential memory access overhead of CODI's latent forward passes grows more pronounced, widening the gap further.

In a finetuning setup, this increased compute can be allocated in several ways, such as additional samples, more epochs, a larger base model, or increased sequence length. Here, we isolate the effect of increased sample count under direct supervision (C-MTP) versus indirect supervision (CODI). To maintain $2\times$ wall-clock parity, we hold the number of epochs constant and set learning rates based on sample-efficiency grid sweeps. We note that this configuration is not necessarily optimal and serves primarily to illustrate the magnitude of gains achievable under a fixed time budget. Overall, Table 11 shows that C-MTP-4 consistently outperforms CODI across all budget levels, with particularly pronounced gains on GSM8k-Test.

Table 9. **Final hyperparameters for all experiments.** Learning rates use the shorthand 5e-4 $= 5 \times 10^{-4}$. The effective batch size is reported as (per-device batch $\times$ gradient-accumulation steps).

Backbone	Data	Setting	Batch (p×g)	Ep.	LR	WD	Max Tok.	Span/Lat.
Explicit CoT								
Llama	Structured	LoRA	128 (64×2)	4	5e-4	0.01	200	–
Llama	Structured	Full	128 (64×2)	3	2e-5	0.01	200	–
Llama	Semi-Nat.	LoRA	128 (32×4)	5	5e-4	0.01	300	–
Llama	Semi-Nat.	Full	128 (32×4)	4	2e-5	0.01	300	–
Qwen	Structured	LoRA	128 (32×4)	5	5e-4	0.1	200	–
Qwen	Structured	Full	128 (32×4)	3	2e-4	0.1	200	–
Llama	Realistic	LoRA	64 (32×2)	5	5e-4	0.1	1024	–
Qwen	Realistic	LoRA	64 (16×4)	5	5e-4	0.1	1024	–
C-MTP								
Llama	Structured	LoRA	128 (32×4)	5	5e-4	0.01	200	2 / 3 / 4
Llama	Structured	LoRA (multitask)	128 (32×4)	8	5e-4	0.01	200	3
Llama	Semi-Nat.	LoRA	128 (32×4)	5	5e-4	0.01	300	2 / 3 / 4
Qwen	Structured	LoRA	128 (32×4)	5	5e-4	0.1	200	2 / 3 / 4
Llama	Realistic	LoRA†	64 (8×8)	8	5e-4	0.1	1024	2
Qwen	Realistic	LoRA†	64 (8×8)	8	5e-4	0.1	1024	2
CODI								
Llama	Structured	LoRA	128 (64×2)	8	7e-4	0.1	200	6
Llama	Semi-Nat.	LoRA	128 (64×2)	8	7e-4	0.1	300	6
Llama	Realistic	LoRA	128 (16×8)	10	5e-4	0.1	1024	6 / 10 / 20
Qwen	Realistic	LoRA	128 (16×8)	10	5e-4	0.1	1024	6 / 10 / 20

† The C-MTP student and teacher are initialized from the merged LoRA checkpoint of the corresponding *Realistic* CoT run (rows above).

I.5. Generalization to Problem Structures

GSM8k-Aug (Deng et al., 2024) was constructed by prompting GPT-4 to generate problems similar to GSM8k, thereby expanding the dataset $\sim 50\times$ via prompting and heuristic filtering. This process results in substantial structural redundancy: our analysis shows that over 50% of all training samples reduce to only 100 distinct problem structures. To evaluate out-of-domain generalization to problem structures not seen during training, we partition the data by held-out problem structures. More analysis in Section E (Appendix)

Concretely, let $\mathcal{D}^n \subset \mathcal{D}$ denote the subset of samples containing n reasoning steps, for $n \in \{2, 3, 4\}$. We partition each $\mathcal{D}^n$ by problem structure into two disjoint, roughly equal-sized splits $\mathcal{D}^n_A$ and $\mathcal{D}^n_B$, ensuring no structure appears in both. Within each split, we hold out 2% of problems as a test set. For each $(n, X) \in \{2, 3, 4\} \times \{A, B\}$, we train on $\mathcal{D}^n_X$ and evaluate on the test sets of *both* $\mathcal{D}^n_A$ and $\mathcal{D}^n_B$, yielding in-domain accuracy (same split) and out-of-domain accuracy (held-out structures).

To ensure a fair comparison, we first train C-MTP-4 on all splits and measure the average generation length on each test set: $\mathcal{D}^2$=3.23 $\pm$ 0.55, $\mathcal{D}^3$=5.38 $\pm$ 0.75, $\mathcal{D}^4$=7.04 $\pm$ 1.13 tokens. Since CODI requires the number of latents to be specified

Table 10. **GSM8k-Aug results on Qwen2.5-1.5B-Instruct.** Accuracy is averaged over 5 runs (run-to-run deviation <1%). #L denotes the mean chain-of-thought length (in steps) with standard deviation, computed over the full evaluation set.

Training Method		GSM8k-Test		GSM-Hard		SVAMP		MultiArith	
		Acc. (↑)	# L (↓)	Acc. (↑)	# L (↓)	Acc. (↑)	# L (↓)	Acc. (↑)	# L (↓)
Standard	CoT	68.23	$36.46_{\pm 16.56}$	26.23	$57.37_{\pm 29.60}$	68.36	$18.30_{\pm 9.24}$	98.00	$19.18_{\pm 2.45}$
Sequential	CODI	44.50	**6.00**	10.46	**6.00**	**63.10**	6.00	**92.22**	6.00
Parallel	CoLaR-2	37.48	$17.86_{\pm 8.40}$	9.01	$27.55_{\pm 14.22}$	49.90	$8.36_{\pm 4.38}$	84.33	$9.30_{\pm 2.01}$
	CoLaR-3	30.69	$12.04_{\pm 5.42}$	7.19	$18.36_{\pm 10.05}$	43.64	$5.80_{\pm 2.89}$	68.44	$6.35_{\pm 1.03}$
	CoLaR-4	25.28	$9.29_{\pm 4.13}$	6.02	$14.14_{\pm 8.46}$	40.22	$4.57_{\pm 2.17}$	73.33	$4.88_{\pm 0.87}$
	C-MTP-2	**50.57**	$16.60_{\pm 8.20}$	**11.78**	$25.08_{\pm 12.56}$	54.20	$7.51_{\pm 4.42}$	90.44	$8.27_{\pm 1.47}$
	C-MTP-3	42.29	$10.84_{\pm 5.52}$	10.60	$16.01_{\pm 8.29}$	45.96	$4.79_{\pm 2.88}$	79.22	$5.53_{\pm 1.31}$
	C-MTP-4	34.42	$7.99_{\pm 4.03}$	8.02	$11.77_{\pm 6.29}$	44.24	$\mathbf{3.40_{\pm 2.08}}$	72.11	$\mathbf{3.94_{\pm 0.95}}$

Figure 4. Sample efficiency comparison on GSM8k-Aug.

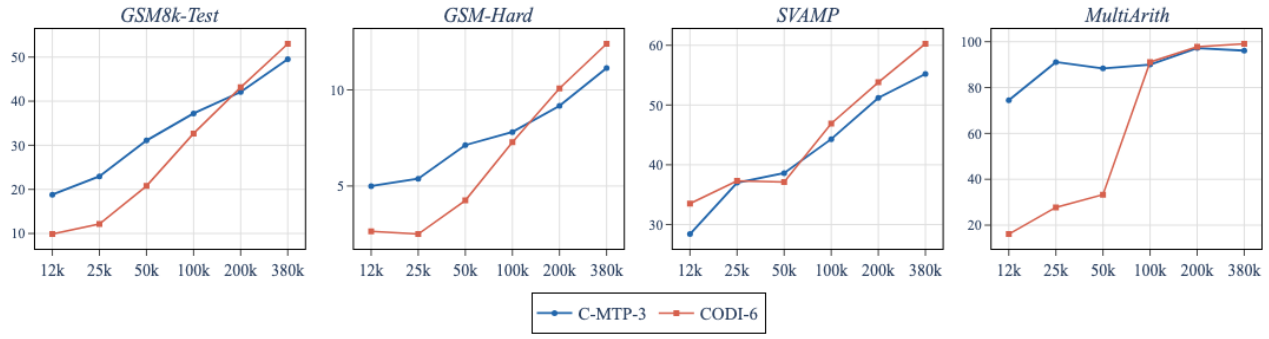


Table 11. Fixed Wall clock time Comparison on GSM8k-aug. We report downstream accuracy on each (# Samples, Epochs) configuration on CODI and C-MTP-4. C-MTP-4 consistently outperforms CODI across all budget levels, with gains on GSM8k-Test and MultiArith particularly pronounced.

WallClock Time		GSM8k Test	GSM Hard	SVAMP	MultiArith
(50k, 7)	CODI	20.85	4.17	36.80	33.33
(100k, 7)	C-MTP-4	29.19	6.29	43.10	86.11
(60k, 6)	CODI	22.44	5.00	40.90	31.67
(120k, 6)	C-MTP-4	30.63	6.82	41.90	87.22
(70k, 5)	CODI	22.14	4.62	42.70	37.22
(140k, 5)	C-MTP-4	32.37	7.96	41.90	86.67
(88k, 4)	CODI	26.00	5.46	42.90	51.11
(175k, 4)	C-MTP-4	33.59	8.42	45.20	91.67

before training, we use these lengths to guide the choice, setting the latent count to 4, 6, and 8 for $n \in \{2, 3, 4\}$ respectively.

Table 12 (Appendix) reports in-domain and out-of-domain accuracy for C-MTP and CODI across reasoning lengths $n \in \{2, 3, 4\}$. C-MTP generalizes better to held-out problem structures in all settings. The gap is particularly clear at $n = 3$ and $n = 4$, where CODI's out-of-domain accuracy drops substantially (e.g., 27.10 and 13.83 on $n = 3$ and $n = 4$ respectively) while C-MTP retains stronger performance (42.86 and 24.04). In-domain, the two methods are more competitive, with CODI performing slightly better at $n = 2$ while C-MTP leads at longer reasoning chains. Overall, these results suggest that the sequential training of indirect supervision methods leads to overfitting to the specific problem structures seen during training.

Table 12. in-domain and out-of-domain evaluation results for C-MTP-4 and CODI across reasoning lengths $n \in \{2, 3, 4\}$. Each matrix entry reports accuracy when trained on the row set and evaluated on the column set.

C-MTP

	$\mathcal{D}_A^2$	$\mathcal{D}_B^2$
$\mathcal{D}_A^2$	73.41	53.54
$\mathcal{D}_B^2$	60.40	72.42

	$\mathcal{D}_A^3$	$\mathcal{D}_B^3$
$\mathcal{D}_A^3$	56.46	42.86
$\mathcal{D}_B^3$	46.09	55.38

	$\mathcal{D}_A^4$	$\mathcal{D}_B^4$
$\mathcal{D}_A^4$	32.55	24.04
$\mathcal{D}_B^4$	24.68	33.40

CODI

	$\mathcal{D}_A^2$	$\mathcal{D}_B^2$
$\mathcal{D}_A^2$	75.11	51.56
$\mathcal{D}_B^2$	63.79	76.52

	$\mathcal{D}_A^3$	$\mathcal{D}_B^3$
$\mathcal{D}_A^3$	41.88	27.10
$\mathcal{D}_B^3$	26.42	40.41

	$\mathcal{D}_A^4$	$\mathcal{D}_B^4$
$\mathcal{D}_A^4$	17.87	13.83
$\mathcal{D}_B^4$	17.23	18.51

I.6. Full Example of MTP Error Accumulation

We present an example of how an error at the initial part of the sequence can cascade to the rest of the CoT. / represents span prediction boundary.

Question: Janet has 16 eggs. She uses 3 eggs to make breakfast and 4 eggs to bake muffins. If she sells the remaining eggs at the farmers' market for 2 each, how much money does she make at the farmers' market each day?
CoT: She has 16 eggs - 3 eggs - 4 eggs = 9 eggs left for the farmers' market. Thus, Janet makes 9 * 2 =18 in the farmers' market each day.
MTP Prediction: Janet/ has /16 -/ 3/ = /13 eggs/ left after/ her breakfast/. She/ bakes/ muffins/ with /13 */ 4/ = /52 eggs/. She/ sells /52 -/ 16/ = /36 eggs/ at the/ farmers'/ market daily/. So/, she/ makes /36 */ 2/ = /72 daily/ at the/ farmers'/ market.

Question: Emma can make and upload 72 vlogs per month. But she was only able to make 18 vlogs for the first week, 21 vlogs for the second week, and 15 vlogs for the third week. How many vlogs should she do to complete the 72 vlogs per month?
CoT: Emma was able to make 18 + 21 + 15 = 54 vlogs for the first three weeks. So, she needs to make 72 - 54 = 18 more vlogs to meet the 72 vlogs per month requirement.
MTP Prediction: Emma has first/ to /72 - /18 - /21 = /45 vib/logs to meet/ her quota./ Finally her third/ week, she/ has /45 - /15 = /30 moreib/ toib to/ meet. quota/ Thus, of/ she should /72 + /30 = /102 moreks/ to meet the/ monthly72.

J. CoLaR - CODI Implementation Discrepancy

We acknowledge that our reproduced results for CODI (Shen et al., 2025) differ from the baseline results reported by CoLaR (Tan et al., 2026). This discrepancy likely stems from implementation differences, as the official CODI codebase may not have been accessible during CoLaR's development. Our experiments use the official CODI repository, yielding results consistent with those reported in the original paper.