LEARNING TO ECHO CHAIN OF THOUGHT IN LATENT SPACE VIA SELF-DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Chain-of-thought (CoT) reasoning has driven much of the recent progress of language models on complex problems, but generating reasoning traces autoregressively makes inference costly. Latent reasoning instead carries out intermediate computation in continuous hidden states, which can compress reasoning into far fewer steps. Supervising these states is the open problem: a text trace does not say what each state should encode, so existing methods rely on sparse answer supervision or on traces with clear reasoning-step boundaries, which most domains lack. We introduce ECHO, which teaches a model to reason in latent space by "echoing" its own explicit reasoning. Our key insight is that as a model reads the CoT, its hidden states already trace the progression of its reasoning. Thus, randomly sampling and sorting them yields dense targets for latent states, without a separate teacher, a curriculum, and step annotations. ECHO has two branches: an explicit branch that sees the question and CoT and a latent branch that sees only the question and learns to reason with a small set of latent slots. At each training step, the slots are supervised with freshly sampled, chronologically ordered hidden states from the explicit branch. With GPT-2 and Llama-3.2-1B-Instruct, ECHO is the most competitive latent method in 9 of 12 settings across math, commonsense reasoning, and coding, even surpassing Explicit CoT in some cases. Since it refines a compact set of slots in parallel, it reduces inference latency by up to 63× relative to Explicit CoT. These results suggest that models can effectively self-distill from the CoT hidden states and reason compactly in the latent space during inference.

1 INTRODUCTION

Chain-of-thought (CoT) reasoning has been central to the remarkable capabilities of large language models (LLMs) across math, science, and programming, helping models tackle difficult problems by expressing intermediate computations in text (Wei et al., 2023; Kojima et al., 2022; OpenAI, 2026; DeepSeek-AI, 2025). This textual workspace is powerful, but costly. Under autoregressive decoding, both key-value cache size and decoding latency increase with generation length, making the cost of expressing intermediate computations in text an important bottleneck.

Humans, however, can reason about abstract problems without verbalizing every word (Constantinescu et al., 2016; Fedorenko et al., 2024). This motivates exploring computational workspaces beyond text. Latent reasoning does so by carrying out intermediate computation in continuous hidden representations before answering (Hao et al., 2024; Shen et al., 2025; Wu et al., 2025; Wei et al., 2026; Kuzina et al., 2026; Fan et al., 2026; Solgi et al., 2026). By avoiding a discrete token choice in intermediate steps, latent representations can preserve richer information about possible reasoning continuations and compress reasoning into fewer steps (Hao et al., 2024; Deng et al., 2025).

Yet training LLMs to reason in latent space remains challenging: it is unclear how to supervise the latent states. Explicit CoT models learn from dense token-level supervision on natural or synthetic reasoning data (Zelikman et al., 2022; DeepSeek-AI, 2025; Kimi-Team, 2026; Yang et al., 2025; Team et al., 2026; Abdin et al., 2024), but these traces do not specify what each latent state should encode. Existing methods (Wei et al., 2026; Shen et al., 2025; Kuzina et al., 2026; Solgi et al., 2026; Fan et al., 2026) provide different forms of supervision but suffer from drawbacks: supervising latent states only through the answer yields a sparse signal, while aligning each state with an explicit reasoning step in the text assumes clear step boundaries, which free-form reasoning traces often lack.

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.

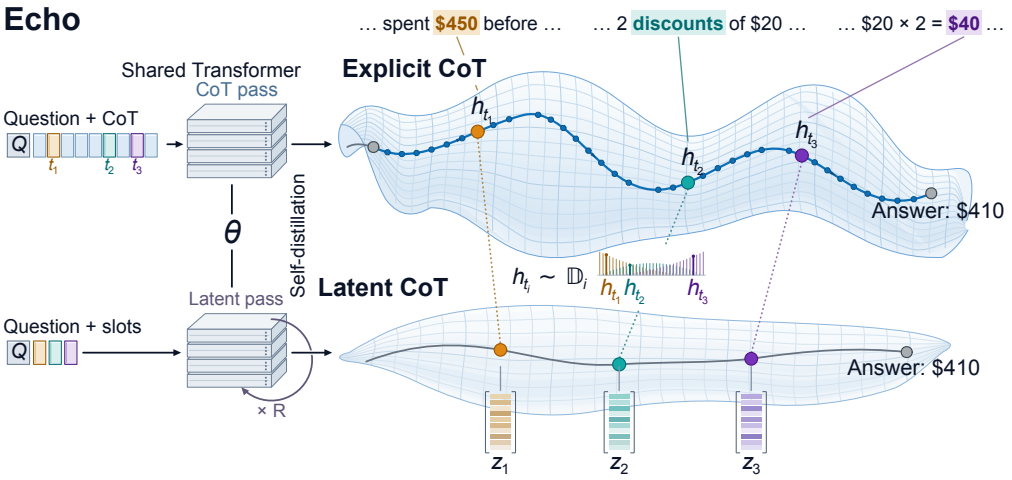


Figure 1: Overview of Echo. A shared Transformer runs one explicit CoT pass and R latent passes. Self-distillation aligns latent states $z_{1:3}$ with sampled explicit-CoT hidden states $h_{t_{1:3}}$.

Our key insight is that this supervision already exists in the model's own hidden states. As a model reads a CoT, the hidden state at each position encodes the reasoning up to that point, so any chronologically ordered subset of these states traces a compressed trajectory of that reasoning. This turns any free-form CoT into dense, per-slot supervision with no step boundaries: we sample CoT positions at random, keep them in chronological order, and use the hidden states at those positions as targets for the latent slots. Since the positions are resampled at every training step, each slot sees targets from many parts of the CoT rather than from one fixed segment. Since the model learns to "echo" its own explicit reasoning in latent space, we call the method Echo (Figure 1).

We introduce Echo, an online self-distillation method that turns free-form CoT into dense, ordered supervision for a fixed-size latent workspace (Figure 1). Echo consists of two branches with shared parameters. On the explicit-CoT branch, the model is trained with next-token prediction on the CoT and answer given the question. This branch also yields the trajectory of hidden states from which we sample the distillation targets. On the latent-CoT branch, the model sees only the question and learns to predict these chosen targets with its latent slots in parallel. To compensate for the loss in expressivity from parallel latent decoding (Li et al., 2024), we use a looped architecture (Fan et al., 2026) to refine the latent slots over multiple recurrent iterations. Jointly, the model is trained to predict the answer from its refined latent slots. Our contributions are as follows:

- We propose Echo, a single-stage training method for latent reasoning that needs no separate teacher, no curriculum, and no step boundaries, all of which prior work relies on. The key is to sample distillation targets at random from the model's own CoT hidden states. We propose four different target selection policies, of which `Random-Sorted` is simple and effective (Section 3).
- Across math, commonsense reasoning, and coding, Echo has the highest average accuracy among all baselines and is the most competitive latent method in 9 of 12 settings. Surprisingly, in commonsense reasoning and coding, Echo outperforms Explicit CoT by up to 5.16 and 6.56 percentage points, respectively (Section 4.2)
- Echo achieves this accuracy with lower inference latency than both Explicit CoT and LOTUS on all six backbone–dataset pairs. Its speedup over Explicit CoT scales with reasoning length, from 2.3× on GSM8K (~17 tokens) to 63× on PyX Forward (~460 tokens). Relative to LOTUS, it lowers latency by 5.2–16.2% and GFLOPs by up to 12× (Section 4.3).

2 Related Work

Supervision for latent reasoning. Latent reasoning methods differ in their intermediate supervision and training procedures. Coconut (Hao et al., 2024) and LOTUS (Fan et al., 2026) assume that reasoning data is decomposed into steps and train with a curriculum. Coconut gradually replaces

Table 1: Comparison of Echo with Coconut (Hao et al., 2024), CODI (Shen et al., 2025), LoRi (Solgi et al., 2026), and LOTUS (Fan et al., 2026). *Supervision*: intermediate targets for latent states. *One-stage*: no preparatory CoT or teacher training beyond pretraining. *Curriculum*: progressive CoT replacement. *CoT boundaries*: requires CoT segmentation for latent replacement or supervision. *Self-distilled*: teacher and student share a backbone.

Method	Supervision	One-stage	Curriculum	CoT boundaries	Self-distilled
Coconut	Textual CoT and answer	✗	✓	✓	✗
CODI	Pre-answer hidden state	✓	✗	✗	✓
LoRi	Low-rank trajectory statistics	✗	✗	✗	✗
LOTUS	Textual CoT and answer	✗	✓	✓	✗
Echo (ours)	Ordered sampled hidden states	✓	✗	✗	✓

reasoning steps with latent states and learns from the remaining CoT text and answer without direct latent targets. LOTUS first trains a teacher model on explicit CoT and uses a curriculum to gradually replace the reasoning steps with latent blocks, where each block predicts the corresponding CoT text. CODI jointly trains a student and teacher model to match their hidden states at a single position preceding the answer (Shen et al., 2025). LoRi also pursues a two-stage training, where they train an explicit CoT teacher and uses it to supervise the student by matching the first- and second-order trajectory statistics in a low-rank subspace (Solgi et al., 2026). In Echo, we jointly train the explicit and latent branches through self-distillation, supervising latent slots with repeatedly resampled, chronologically ordered hidden states from the explicit CoT trajectory. This provides dense per-slot supervision without a separate teacher, curriculum, or reasoning-step annotations. Table 1 summarizes these distinctions. Appendix A discusses additional methods.

Teacher-student distillation. Echo is closely related to distillation with privileged information (Lopez-Paz et al., 2015), where the teacher has access to information the student lacks at inference—in our case, the CoT. It also relates to context distillation (Snell et al., 2022), which internalizes instructions or hints so the model can operate without them. Recent self-distillation methods derive the teacher and student policies from the same base model, matching token distributions along student-generated trajectories (Zhao et al., 2026; Shenfeld et al., 2026) or over answer tokens (Kalwar et al., 2026). Unlike these token-level methods, Echo self-distills in representation space.

Looped computation. Looped transformers increase effective computation depth by repeatedly applying the model, without requiring additional generated reasoning tokens (Dehghani et al., 2018; Saunshi et al., 2025; Geiping et al., 2025; Prairie et al., 2026). PCCoT uses Jacobi iteration to update multiple continuous thoughts in parallel (Wu et al., 2025). LOTUS refines latent blocks with a looped transformer and aligns each latent block to a reasoning step in the CoT text (Fan et al., 2026). Echo similarly uses a looped transformer to refine question-conditioned latent slots in parallel over multiple recurrences, but does not rely on reasoning-step boundaries to supervise the slots.

Masked reconstruction. Masked modeling learns representations by reconstructing masked portions of an input. The closest related work are data2vec, Context Autoencoders, and I-JEPA (Baevski et al., 2022; Chen et al., 2024; Assran et al., 2023), which use this paradigm to use contextualized representations rather than raw inputs (Devlin et al., 2019; He et al., 2022) as prediction targets. Echo can be viewed as a special case of conditional masked modeling (see Section 3.2).

3 Method

3.1 Echo

We propose Echo, which trains a latent reasoning model by self-distilling explicit CoT into a compact sequence of continuous states. Let a training example be $(\mathbf{x}, \mathbf{c}, \mathbf{y})$, where $\mathbf{x} = (x_1, \ldots, x_m)$ is the question, $\mathbf{c} = (c_1, \ldots, c_n)$ is the observed CoT trace, and $\mathbf{y} = (y_1, \ldots, y_T)$ is the answer. Echo runs a single causal Transformer decoder f_θ in two branches (Figure 2). The *explicit-CoT branch* reads $[\mathbf{x}; \mathbf{c}; \mathbf{y}]$ and produces hidden states at each CoT position. Given this hidden state trajectory, the *latent-CoT branch* predicts a random ordered subset K of the hidden states of $\mathbf{c}$ from its latent slots in parallel and answers from them. We now describe each branch and then the distillation that links the two.

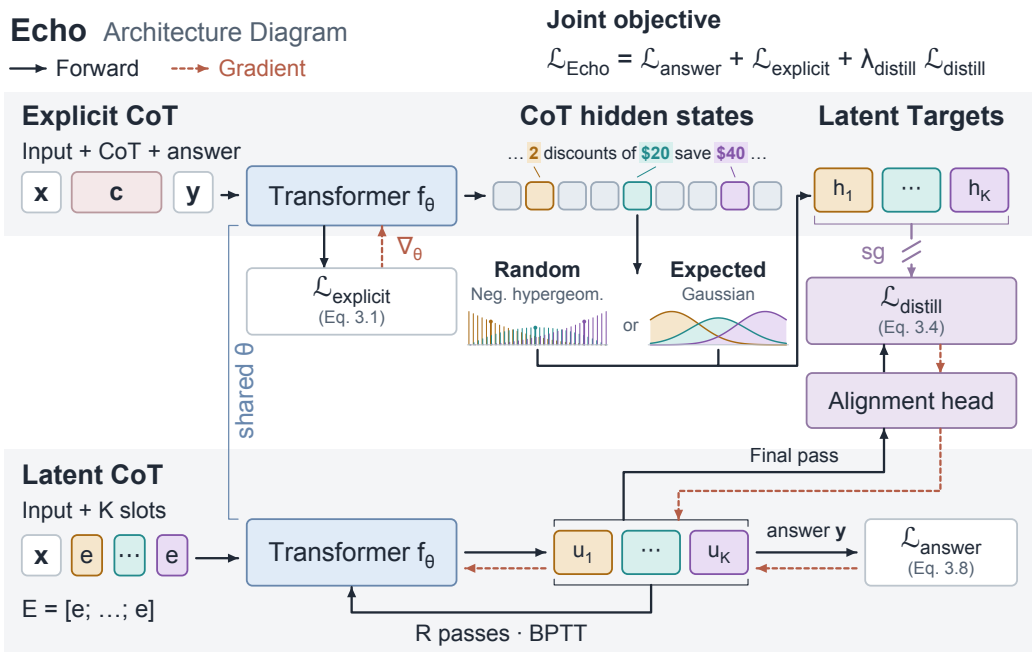


Figure 2: A detailed diagram of Echo architecture. At inference, only the bottom latent path is used.

Explicit-CoT branch. The explicit branch is trained with next-token prediction:

$$\mathcal{L}_{\text{explicit}} = -\frac{1}{n+T} \log p_\theta(\mathbf{c}, \mathbf{y} \mid \mathbf{x}). \tag{3.1}$$

This forward pass computes hidden states at each CoT position. We write $h_{i,t}^{(\ell)}$ for the hidden state at Transformer layer ℓ and CoT position t of example i; these later serve as distillation targets.

Latent-CoT branch: parallel and looped reasoning. Autoregressive latent-reasoning methods (Hao et al., 2024; Shen et al., 2025; Wei et al., 2026; Tan et al., 2026) generate a latent state one at a time, requiring K sequential decoder calls for K latent states. We instead follow the parallel, looped structure of LOTUS (Fan et al., 2026): K latent slots are processed jointly and refined over R weight-tied passes, decoupling the number of slots K from the number of recurrent passes R.

We first encode $[\mathbf{x}; \langle \texttt{BoT} \rangle]$ and keep its key–value cache C_{prefix}. Let $e \in \mathbb{R}^d$ be a learned slot embedding shared across examples and latent slot positions, and let $E = [e; \ldots; e] \in \mathbb{R}^{K \times d}$. Writing $U^{(r,\ell)} \in \mathbb{R}^{K \times d}$ for the latent slot states at layer ℓ of pass r, and starting from $U^{(0,L)} = \mathbf{0}$, we apply

$$\left(\{U^{(r,\ell)}\}_{\ell=1}^{L},\, C^{(r)}\right) = f_\theta\left(C_{\text{prefix}},\, E + U^{(r-1,L)}\right), \qquad r = 1, \ldots, R, \tag{3.2}$$

where $C^{(r)}$ is the resulting question-and-latent cache. Every pass re-injects E, uses the same parameters, and starts from the same C_{prefix}; the cache $C^{(r-1)}$ is not carried over, so only the final-layer slot states recur across passes. Within a pass, all K slots are computed in parallel under a causal mask: slot j attends to the prefix and to slots $1{:}j$ only. Although the slots share the embedding e, their positions make them ordered via RoPE (Su et al., 2023).

To answer, we append $\langle \texttt{EoT} \rangle$ to the final cache $C^{(R)}$ and decode autoregressively:

$$p_{\text{answer},\theta}(\mathbf{y} \mid \mathbf{x}) = \prod_{t=1}^{T} p_\theta\left(y_t \mid C^{(R)}, \langle \texttt{EoT} \rangle, \mathbf{y}_{<t}\right), \qquad \mathcal{L}_{\text{answer}} = -\frac{1}{T} \log p_{\text{answer},\theta}(\mathbf{y} \mid \mathbf{x}). \tag{3.3}$$

Importantly, $\mathcal{L}_{\text{answer}}$ conditions on the latent branch's *own* refined slots rather than the ground-truth CoT. This student-forcing makes answer supervision on-policy with respect to the latent trajectory, mirroring inference and avoiding the train–test mismatch of teacher-forcing. Since $\mathcal{L}_{\text{answer}}$ backpropagates through all R recurrent passes, it trains the slots to be useful for answering. However, it is a sparse signal: a single answer loss must credit K slots across R passes, and nothing specifies *what* intermediate reasoning each slot should carry.

Distillation target selection policy. To provide this intermediate signal, we supervise the latent slots with hidden states from the explicit-CoT trajectory. For a trace of length $n_i \geq K$, we select K positions in increasing order, $1 \leq t_{i1} < \cdots < t_{iK} \leq n_i$, and use the explicit-CoT hidden state at position t_{ij} as the target for slot j. Earlier slots are thus matched to earlier positions along the trace, consistent with the slots' causal mask. We consider target-selection policies with different slot-specific position distributions, including `Random-Sorted`, `Gaussian`, `Triangular`, and `Block` (Figure S1 in Appendix B). We also explore both *sampled* and *expected targets*. For sampled targets, we resample positions at every optimizer step, varying supervision while preserving temporal order. Resampling exposes the same K slots to different parts of the trace without requiring step annotations, while sorting preserves chronological assignment. *Expected targets* average the explicit-CoT hidden states using each slot's position probabilities as weights. Appendix B covers the policies, ordered sampling, expected-target construction procedures, and treatment of traces shorter than K.

Self-distillation from explicit to latent reasoning. Given the selected positions $\mathbf{t}_i$, we align each latent slot with its corresponding explicit-CoT hidden state. Let $u_{ij}^{(\ell)} = [U_i^{(R,\ell)}]_j$ be the layer-ℓ state of slot j after the final recurrent pass. Its distillation target is $h_{i,t_{ij}}^{(\ell)}$, which depends only on the question and the CoT up to t_{ij} under a causal mask. We compute the distillation loss with the hidden states from the last M of the L Transformer layers, each with its own learned alignment head q_ℓ:

$$\mathcal{L}_{\text{distill}} = \frac{1}{M|\mathcal{B}|K} \sum_{i \in \mathcal{B}} \sum_{j=1}^{K} \sum_{\ell=L-M+1}^{L} \mathcal{D}_\ell\Big(q_\ell(u_{ij}^{(\ell)}),\ \text{sg}\Big(h_{i,t_{ij}}^{(\ell)}\Big)\Big), \tag{3.4}$$

where sg is a stop-gradient, $\mathcal{B}$ is the mini-batch, and $\mathcal{D}_\ell$ is the Smooth L1 loss averaged over hidden dimensions and divided by the standard deviation of the sampled targets at layer ℓ in the current step. Resampling positions at each training step exposes each slot to different points along the CoT trajectory over training. Figure 4b shows that both resampling and preserving temporal order matter for distillation. Appendix C describes the alignment-head architecture and the loss objective.

Joint objective. We jointly optimize the shared decoder, the slot embedding, and the alignment heads with

$$\mathcal{L}_{\text{Echo}} = \mathcal{L}_{\text{answer}} + \mathcal{L}_{\text{explicit}} + \lambda_{\text{distill}} \mathcal{L}_{\text{distill}}. \tag{3.5}$$

Echo thus learns explicit reasoning while distilling it into a compact latent workspace.

Inference. During inference, we encode $[\mathbf{x}; \langle \texttt{BoT} \rangle]$, run R parallel recurrent passes of Eq. (3.2), append $\langle \texttt{EoT} \rangle$, and decode the answer as in Eq. (3.3). No explicit CoT tokens are generated.

3.2 Echo as Masked Reconstruction

Echo's distillation objective can be viewed as reconstructing some heldout reasoning content in representation space. This view connects Echo to masked modeling, where a model predicts withheld content from visible context. Echo directly reconstructs in the representation space, instead of the original content (Devlin et al., 2019; He et al., 2022; Baevski et al., 2022; Assran et al., 2023).

A distinctive feature is that Echo separates *what is withheld* from *what is supervised*. In standard masked reconstruction, the model predicts the masked input. In Echo, the entire CoT is withheld from the latent branch and the model predicts a randomly resampled subset of its hidden states. Each target is a causal hidden state, so reconstructing it requires recovering the reasoning accumulated up to its position. This encourages the model to learn a compressed, ordered representation.

4 Results

4.1 Experimental Setup

Models and baselines. We use GPT-2 (124M) and Llama-3.2-1B-Instruct for our experiments. We compare Echo against four latent reasoning methods: Coconut (Hao et al., 2024), CODI (Shen et al., 2025), LoRi (Solgi et al., 2026), and LOTUS (Fan et al., 2026). We also train an explicit CoT model and a model trained only on the answer given the questions (Answer-only SFT) as references.

Datasets. We train and evaluate on three datasets that demand different forms of reasoning: GSM8K-Aug (Deng et al., 2023) for math, CommonsenseQA (CSQA) (Talmor et al., 2019) with synthetic CoT generated by GPT-4o-mini (Shen et al., 2025) for commonsense reasoning, and PyX Forward (Ding

Table 2: GSM8K accuracy and out-of-distribution accuracy (%). LoRi† denotes our best-effort GPT-2 re-implementation, as the code has not been released. Bold denotes the best available score within each backbone and column, excluding Answer-only SFT and Explicit CoT baselines.

Model	Method	GSM8K	OOD		
			GSM-Hard	SVAMP	MultiArith
GPT-2	Answer-only SFT	27.75	6.07	25.30	54.44
	Explicit CoT	46.70	10.08	45.10	93.89
	Coconut	25.85	5.46	26.90	70.56
	CODI	38.36	8.72	40.30	**93.33**
	LoRi†	1.36	0.38	5.10	4.44
	LOTUS	44.35	10.08	41.10	90.56
	Echo (ours)	**47.08**	**10.99**	**42.80**	**93.33**
Llama-3.2-1B Instruct	Answer-only SFT	28.73	6.60	44.40	62.22
	Explicit CoT	58.30	13.27	56.90	98.33
	Coconut	34.57	8.87	44.10	89.44
	CODI	52.08	12.28	58.50	97.78
	LoRi†	30.71	6.60	42.40	38.89
	LOTUS	**56.94**	**12.74**	58.90	**98.33**
	Echo (ours)	55.12	12.21	**59.10**	97.22

et al., 2024) for predicting the output of short Python programs. We report test accuracy on GSM8K and PyX, and validation accuracy on CSQA since its test labels are unavailable. To test whether latent reasoning transfers beyond its training distribution, we also evaluate each GSM8K-Aug model on three unseen math benchmarks: GSM-Hard (Gao et al., 2023), SVAMP (Patel et al., 2021), and MultiArith (Roy & Roth, 2015). Appendix E reports training details and hyperparameters.

Evaluation. We evaluate across the six datasets using greedy decoding across all models. For inference compute analyses, we report the average GFLOPs and inference latency with batch size of 1 on a single H200 GPU over the dataset to remove the effect of padding in the inputs.

4.2 Main Results on Math, Commonsense Reasoning, and Coding

We present the main results across six datasets in Tables 2 and 3.

Echo is the strongest latent method overall. Averaged over the six benchmarks, Echo attains the highest accuracy among all latent methods and baselines, outperforming the Explicit CoT by 2.24 points with GPT-2 and competitive with Llama-3.2-1B Instruct.

The largest gains appear on free-form reasoning traces. CSQA and PyX, like most reasoning datasets (NVIDIA, 2025; Zhou et al., 2025), have rationales that are longer and free-form without step boundaries. We observe that Echo's advantage is largest on these tasks. It surpasses Explicit CoT in three of the four settings, including +8.3 percentage points (pp) on GPT-2 CSQA. In contrast, in GSM8K-Aug, where traces are sequences of intermediate calculations (e.g., $40 \times 2 = 80$) each corresponding to a reasoning step, methods relying on delineated reasoning steps perform competitively. With GPT-2, Echo is the best latent method on GSM8K (+2.73pp over LOTUS) and out-of-distribution tasks, and slightly exceeds Explicit CoT. With Llama, LOTUS is the strongest latent method on GSM8K, GSM-Hard, and MultiArith, while Echo is best on SVAMP.

Latent reasoning helps even where explicit CoT hurts. Training on explicit rationales does not always help small models (Shen et al., 2025; Hsieh et al., 2023; Sprague et al., 2025): explicit CoT underperforms Answer-only SFT on GPT-2 PyX and Llama CSQA. Echo still improves over Answer-only SFT in both cases (+4.23pp and +2.22pp), and on Llama CSQA, it is the only latent method to do so. Even when training on the CoT is detrimental for Explicit CoT, some latent reasoning models including Echo can learn to reason robustly via the distillation supervision.

Table 3: CommonsenseQA and PyX Forward Accuracy (%). LoRi† denotes our best-effort GPT-2 re-implementation, as the code has not been released. Bold indicates the best result for each model and dataset, excluding the Answer-only SFT and Explicit CoT baselines.

Model	Method	CommonsenseQA	PyX Forward
GPT-2	Answer-only SFT	20.39	39.96
	Explicit CoT	36.86	37.63
	Coconut	21.29	35.73
	CODI	37.51	42.71
	LoRi†	17.28	3.59
	LOTUS	41.85	41.44
	Echo (ours)	**45.29**	**44.19**
Llama-3.2-1B Instruct	Answer-only SFT	73.46	61.53
	Explicit CoT	70.52	67.90
	Coconut	66.34	54.52
	CODI	73.46	62.71
	LoRi†	67.98	39.65
	LOTUS	71.91	62.86
	Echo (ours)	**75.68**	**65.30**

4.3 Echo is Compute Efficient

A key motivation for latent reasoning is efficient inference. Echo replaces autoregressive CoT generation with R recurrent passes over K slots updated in parallel, setting the reasoning cost by (K, R) rather than generation length. We compare accuracy against measured latency and analytical GFLOPs in Figure 3 (GPT-2) and Figure S2 (Llama-3.2-1B) in Appendix D.1.

The inference speedup of Echo over Explicit CoT scales with generation length across both models (Table S2 of Appendix D.1): 2.3–2.5× on GSM8K (∼17 tokens), 11× on CSQA (∼67–85 tokens), and 61–63× on PyX (∼460 tokens). GFLOPs follow the same trend more gently, shrinking by 1.3–2.0× on CSQA and 3.3–3.7× on PyX; on GSM8K the traces are too short to amortize the fixed latent reasoning cost, so Echo uses more FLOPs yet remains 2.3–2.5× faster with parallel decoding.

With the compute efficiency gains, Echo improves by 5.2–8.3pp on CSQA and 6.9pp on GPT-2 PyX, trailing Explicit CoT by only 0.9pp on Llama PyX. Against LOTUS, our strongest latent baseline, it lowers latency by 5.2–16.2% on all six backbone–dataset pairs and GFLOPs by 3.2–12.0× on five.

4.4 Design Studies

We next study Echo's main design choices, specifically its training objectives, the ordering and resampling of distillation targets, the target selection policy, and the number of recurrences.

4.4.1 All three objectives are necessary

Our joint self-distillation objective consists of three losses: $\mathcal{L}_{\text{answer}}$, $\mathcal{L}_{\text{distill}}$, and $\mathcal{L}_{\text{explicit}}$. We remove each of the three losses with GPT-2 on GSM8K and find all three essential (Figure 4a). We observe the same phenomenon in Llama in Figure S3 (Appendix D.2). Without $\mathcal{L}_{\text{answer}}$, nothing trains the model to decode an answer from its own latent slots, so accuracy falls from 47.1% to 18.3%. Without $\mathcal{L}_{\text{distill}}$, the slots receive only the sparse answer signal, as with pause tokens (Goyal et al., 2024), and accuracy drops to 32.5% (−14.6pp). Dense supervision of the latents is therefore important for training latent reasoning models. Removing $\mathcal{L}_{\text{explicit}}$ causes the largest drop, to 9.6% (−37.5pp). Without it, the GSM8K-Aug reasoning traces are off-policy for the explicit-CoT branch, so its hidden states on the traces are less informative distillation targets. Moreover, the two branches share parameters, so without explicit-CoT training to anchor the hidden states, the model can minimize $\mathcal{L}_{\text{distill}}$ trivially by making its own targets predictable but uninformative.

4.4.2 Target Order and Resampling Both Matter

Although Echo updates latent slots in parallel, we assign sampled explicit-CoT hidden states to these slots in their original temporal order, aligning successive slots with earlier-to-later portions of

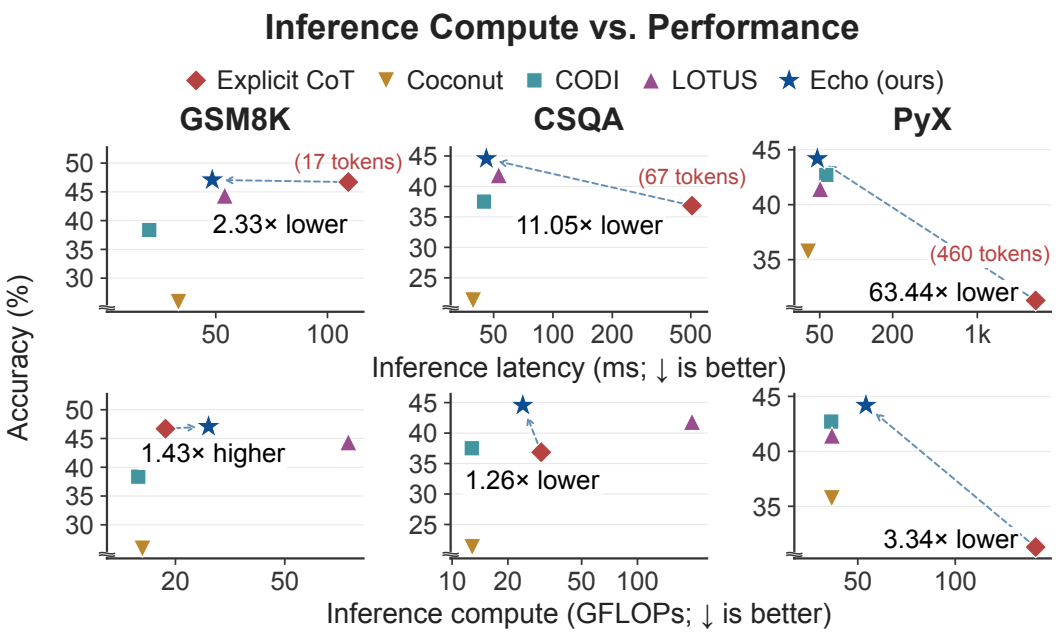


Figure 3: Accuracy vs. inference latency (top) and compute (bottom) for GPT-2. Echo lies on the accuracy–efficiency Pareto frontier (upper left) in every panel except GSM8K compute, running 2.3–63× faster than explicit CoT with 1.3–3.3× fewer GFLOPs on CSQA and PyX. On each of the Explicit CoT model markers, we annotate the mean reasoning generation length on each dataset. We observe that Echo compute efficiency scales with generation length.

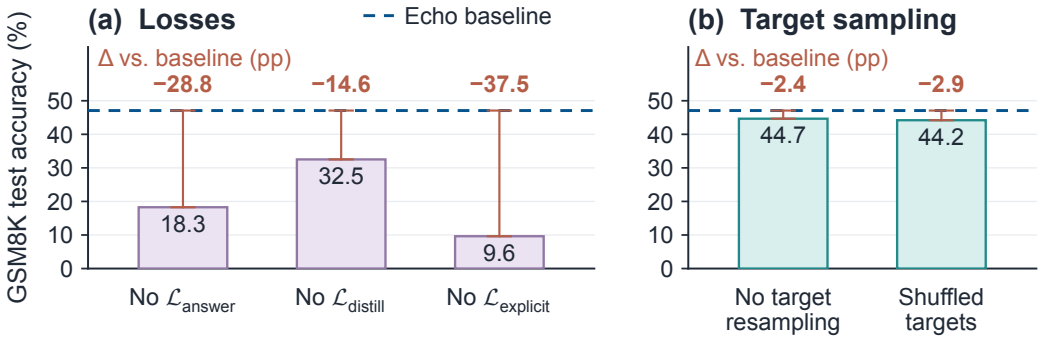


Figure 4: **Echo ablations on GSM8K** (GPT-2, $K = 10$ slots, $M = 5$ supervised layers). **(a)** Ablations of the answer, distillation, and explicit-CoT losses. Each component is critical to Echo. **(b)** Removing target resampling (fixing the sampled targets at the start of training) and shuffling targets to remove their temporal order. Both resampling and chronological order help. Dashed lines indicate the Echo baseline, and annotated deltas indicate the change in test accuracy.

the CoT. We test this design choice by randomly shuffling the assignment of distillation targets on GSM8K.

Chronological assignment improves GPT-2 accuracy from 44.2% to 47.1% (Figure 4b), and Llama shows the same pattern on CSQA and PyX (Figure S4 of Appendix D.3). Resampling targets at every step and exposing each slot to different parts of the CoT hidden state trajectory, rather than fixing them at the start of training ("No target resampling"), also improves by 2.4pp.

4.4.3 Target Selection for Latent Distillation

A key feature of Echo is that we sample distillation targets over a distribution we define over the CoT hidden state trajectory. We compare `Random-Sorted` with structured policies (`Triangular`,

Table 4: Target selection policy ablation with GPT-2. *Sampled* policies draw targets from the policy distribution in chronological order; *Expected* policies average the explicit-CoT hidden states weighted by each slot's position probabilities. Bold marks the best setting per dataset.

Targets	Policy	GSM8K-Aug	CommonsenseQA	PyX Forward
Sampled	Random-Sorted (Negative-hypergeometric)	**47.08**	44.55	**44.19**
	Triangular	46.93	**45.29**	41.86
	Gaussian	45.41	40.54	42.07
	Block	45.94	39.56	42.60
Expected	Triangular	45.56	43.82	43.02
	Gaussian	46.63	42.01	42.71
	Block	45.72	34.48	42.60

`Gaussian`, and `Block` detailed in Appendix B) that define different probability distributions over the CoT hidden-state trajectory. As shown in Table 4 for GPT-2 and Table S3 (Appendix D.4) for Llama, `Random-Sorted` performs best on GSM8K-Aug at 47.08% and PyX at 44.19%. On CSQA, `Triangular-Sampled` performs the best with GPT-2, albeit by 0.6% from `Random-Sorted`. On GSM8K, `Gaussian-Expected` performs the best with Llama by 5.76pp compared to `Random-Sorted`. Although the optimal policy varies by dataset, `Random-Sorted`, which lets each slot see a wide range of CoT hidden states across resamples, is the simplest and most consistently effective.

4.4.4 Fixed versus Dynamic Recurrence

Our main experiments fix the number of recurrences R during training and inference. We investigate whether we could train Echo with adaptive recurrence for test-time scaling. We train Echo with dynamic recurrence (Appendix C.1), sampling R during training so that it can be varied at inference (Prairie et al., 2026; Geiping et al., 2025). We compare this against fixed-R models separately trained for each recurrent depth. Both settings peak at ~47% GSM8K accuracy (Figure 5 of Appendix C.1). Fixed-R models are stronger at low R, since the training and inference match exactly, and the dynamic model holds steady through $R = 8$. However, we do not observe that performance scales beyond a certain recurrence depth.

5 Conclusion

Echo tackles the question of how to densely supervise latent reasoning models and proposes a simple method to self-distill its own CoT into the latent space, with no step annotations, curriculum, or separate teacher. Across GPT-2 and Llama, it is the strongest latent method on average and cuts latency by up to 63× relative to Explicit CoT.

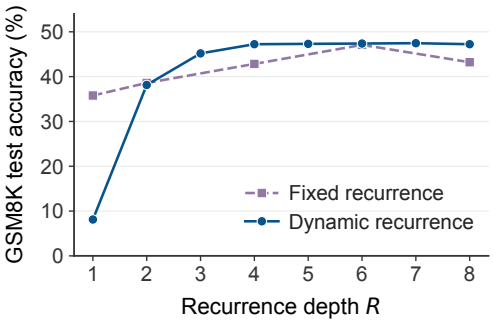


Figure 5: Fixed versus dynamic recurrence on GSM8K with GPT-2.

Limitations and Future Work. The compute benefit depends on generation length: the longer the trace, the more Echo saves over Explicit CoT. For short traces, however, this benefit may not be pronounced. Scaling and testing the robustness of Echo to larger models and larger latent workspaces are important future work. Also, our preliminary dynamic recurrence results motivate adaptive latent reasoning models that learn when to stop refining.

Safety. Replacing text reasoning with latent states removes a channel for CoT monitoring (Baker et al., 2025; Korbak et al., 2025). However, recent works have found CoT to be unreliable and unfaithful proxy for their intents (Turpin et al., 2023; Chen et al., 2025; Bogdan et al., 2026; METR, 2026). Though the loss of a readable trace is a real risk, the fragility of CoT monitoring underscores the need for safeguards that do not depend on a model's stated reasoning: more robust white-box monitors that inspect internal representations directly (Lindsey et al., 2025) and control mechanisms that direct audit and limit the model's actions (Greenblatt et al., 2024).

AI USE STATEMENT

In this work, we used generative AI tools (Claude Code and Codex) to implement methods. They helped debug the code for our proposed method and for the baselines, and helped launch and monitor training and evaluation runs. We also used generative AI tools to help interpret experimental results and verify our math and derivations. We did not use generative AI tools to propose or refine hypotheses or to design and refine the research direction. We did not use generative AI tools to generate datasets. The reasoning traces in GSM8K-Aug and CommonsenseQA chain-of-thought data were generated with language models by prior work. We used these datasets as released, and prior literature has already tested and studied them. Moreover, we used generative AI tools to write code to create figures and to edit the paper for readability. The authors found and formatted all references by hand, without AI assistance. We have reviewed all AI-assisted work. Two authors reviewed and tested the AI-written code. The authors verified all the results from the experiments. The authors read and revised all AI-edited text to make sure it accurately describes our methods, results, and the literature we cite. The authors take full responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

An important ethical consideration for ECHO is that latent inference removes textual CoT, reducing visibility into how the model reaches its answers. We recognize this as a safety concern: CoT monitoring can detect misbehavior that final outputs alone would miss (Baker et al., 2025; Korbak et al., 2025). At the same time, CoT can itself be an unreliable and fragile account of model behavior (Bogdan et al., 2026; METR, 2026). Models may omit hints and reward hacks they rely on (Turpin et al., 2023; Chen et al., 2025), produce post-hoc rationalizations (Arcuschin et al., 2026), or learn to conceal intent when trained against CoT monitors (Baker et al., 2025). Larger models can even reason less faithfully than smaller ones (Lanham et al., 2023), and degenerate or irrelevant text can further hinder interpretation (METR, 2025; Schoen et al., 2025). These limitations do not negate the risks of reduced transparency in latent reasoning, but reinforce the need for safeguards that do not rely solely on a model's stated reasoning. We therefore advocate developing and evaluating white-box methods that directly inspect internal representations (Lindsey et al., 2025) and, more importantly, control mechanisms that monitor and limit model actions (Greenblatt et al., 2024). Such safeguards would strengthen oversight and support responsible use of both explicit and latent reasoning models.

REPRODUCIBILITY STATEMENT

For the reproducibility of our work, we describe all the training details and hyperparameters used in our training, and we plan to release our code for reproducing not only our methods but also our baseline methods that we compare against in this work.

REFERENCES

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J. Hewett, Mojan Javaheripi, Piero Kauffmann, James R. Lee, Yin Tat Lee, Yuanzhi Li, Weishung Liu, Caio C. T. Mendes, Anh Nguyen, Eric Price, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Xin Wang, Rachel Ward, Yue Wu, Dingli Yu, Cyril Zhang, and Yi Zhang. Phi-4 technical report, 2024. URL https://arxiv.org/abs/2412.08905.

Iván Arcuschin, Jett Janiak, Robert Krzyzanowski, Senthooran Rajamanoharan, Neel Nanda, and Arthur Conmy. Chain-of-thought reasoning in the wild is not always faithful, 2026. URL https://arxiv.org/abs/2503.08679.

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15619–15629. IEEE, 2023.

Alexei Baevski, Wei-Ning Hsu, Qiantong Xu, Arun Babu, Jiatao Gu, and Michael Auli. Data2vec: A general framework for self-supervised learning in speech, vision and language. In *International conference on machine learning*, pp. 1298–1312. PMLR, 2022.

Bowen Baker, Joost Huizinga, Leo Gao, Zehao Dou, Melody Y. Guan, Aleksander Madry, Wojciech Zaremba, Jakub Pachocki, and David Farhi. Monitoring reasoning models for misbehavior and the risks of promoting obfuscation, 2025. URL https://arxiv.org/abs/2503.11926.

Paul C. Bogdan, Richard Qi, Jake Eaton, Sam Kennedy, Fabien Roger, Alex Glynn, Runjin Chen, Ben Wright, Otto Stegmaier, Jon Kutasov, Dan Foreman-Mackey, Trenton Bricken, Sylvie Carr, Shan Carter, Monte MacDiarmid, Samuel Marks, Adam Pearce, Elana Simon, Nicholas Carlini, Collin Burns, Jack Lindsey, Sara Price, and Subhash Kantamneni. An alignment assessment of recent cybersecurity incidents. Anthropic, September 2026. URL https://www.anthropic.com/research/alignment-assessment-cybersecurity-incidents. Sara Price and Subhash Kantamneni share senior authorship. Correspondence to subhash@anthropic.com.

Cristina Butucea, Jean-François Delmas, Anne Dutfoy, and Richard Fischer. Maximum entropy distribution of order statistics with given marginals, 2015. URL https://arxiv.org/abs/1509.02019.

Xiaokang Chen, Mingyu Ding, Xiaodi Wang, Ying Xin, Shentong Mo, Yunhao Wang, Shumin Han, Ping Luo, Gang Zeng, and Jingdong Wang. Context autoencoder for self-supervised representation learning. *International Journal of Computer Vision*, 132(1):208–223, 2024.

Yanda Chen, Joe Benton, Ansh Radhakrishnan, Jonathan Uesato, Carson Denison, John Schulman, Arushi Somani, Peter Hase, Misha Wagner, Fabien Roger, Vlad Mikulik, Samuel R. Bowman, Jan Leike, Jared Kaplan, and Ethan Perez. Reasoning models don't always say what they think, 2025. URL https://arxiv.org/abs/2505.05410.

Jeffrey Cheng and Benjamin Van Durme. Compressed chain of thought: Efficient reasoning through dense representations. *arXiv preprint arXiv:2412.13171*, 2024.

Alexandra O Constantinescu, Jill X O'Reilly, and Timothy EJ Behrens. Organizing conceptual knowledge in humans with a gridlike code. *Science*, 352(6292):1464–1468, 2016.

DeepSeek-AI. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, Sept 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.

Jingcheng Deng, Liang Pang, Zihao Wei, Shicheng Xu, Zenghao Duan, Kun Xu, Yang Song, Huawei Shen, and Xueqi Cheng. Llm latent reasoning as chain of superposition. *arXiv preprint arXiv:2510.15522*, 2025.

Yuntian Deng, Kiran Prasad, Roland Fernandez, Paul Smolensky, Vishrav Chaudhary, and Stuart Shieber. Implicit chain of thought reasoning via knowledge distillation, 2023. URL https://arxiv.org/abs/2311.01460.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

Yangruibo Ding, Jinjun Peng, Marcus J. Min, Gail Kaiser, Junfeng Yang, and Baishakhi Ray. Semcoder: Training code language models with comprehensive semantics reasoning, 2024. URL https://arxiv.org/abs/2406.01006.

Ying Fan, Anej Svete, and Kangwook Lee. Bridging the gap between latent and explicit reasoning with looped transformers. *arXiv preprint arXiv:2606.31779*, 2026.

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.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. In *International conference on machine learning*, pp. 10764–10799. PMLR, 2023.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach, 2025. URL https://arxiv.org/abs/2502.05171.

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, 2024. URL https://arxiv.org/abs/2310.02226.

Ryan Greenblatt, Buck Shlegeris, Kshitij Sachan, and Fabien Roger. Ai control: Improving safety despite intentional subversion, 2024. URL https://arxiv.org/abs/2312.06942.

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

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 15979–15988. IEEE, 2022.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Namgyu Ho, Laura Schmid, and Se-Young Yun. Large language models are reasoning teachers. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 14852–14882, 2023.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes, 2023. URL https://arxiv.org/abs/2305.02301.

Durgesh Kalwar, Vardhan Palod, and Subbarao Kambhampati. Masked distillation: Internalizing the chain-of-thought in language models. *arXiv preprint arXiv:2607.22629*, 2026.

Kimi-Team. Kimi k3: Open frontier intelligence, 2026. URL https://arxiv.org/abs/2607.24653.

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.

Tomek Korbak, Mikita Balesni, Elizabeth Barnes, Yoshua Bengio, Joe Benton, Joseph Bloom, Mark Chen, Alan Cooney, Allan Dafoe, Anca Dragan, Scott Emmons, Owain Evans, David Farhi, Ryan Greenblatt, Dan Hendrycks, Marius Hobbhahn, Evan Hubinger, Geoffrey Irving, Erik Jenner, Daniel Kokotajlo, Victoria Krakovna, Shane Legg, David Lindner, David Luan, Aleksander Mądry, Julian Michael, Neel Nanda, Dave Orr, Jakub Pachocki, Ethan Perez, Mary Phuong, Fabien Roger, Joshua Saxe, Buck Shlegeris, Martín Soto, Eric Steinberger, Jasmine Wang, Wojciech Zaremba, Bowen Baker, Rohin Shah, and Vlad Mikulik. Chain of thought monitorability: A new and fragile opportunity for ai safety, 2025. URL https://arxiv.org/abs/2507.11473.

Anna Kuzina, Maciej Pioro, Paul N. Whatmough, and Babak Ehteshami Bejnordi. Kava: Latent reasoning via compressed kv-cache distillation, 2026. URL https://arxiv.org/abs/2510.02312.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, Kamilė Lukošiūtė, Karina Nguyen, Newton Cheng, Nicholas Joseph, Nicholas Schiefer, Oliver Rausch, Robin Larson, Sam McCandlish, Sandipan Kundu, Saurav Kadavath, Shannon Yang, Thomas Henighan, Timothy Maxwell, Timothy Telleen-Lawton, Tristan Hume, Zac Hatfield-Dodds, Jared Kaplan, Jan Brauner, Samuel R. Bowman, and Ethan Perez. Measuring faithfulness in chain-of-thought reasoning, 2023. URL https://arxiv.org/abs/2307.13702.

Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. Chain of thought empowers transformers to solve inherently serial problems, 2024. URL https://arxiv.org/abs/2402.12875.

Jack Lindsey, Wes Gurnee, Emmanuel Ameisen, Brian Chen, Adam Pearce, Nicholas L. Turner, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. On the biology of a large language model. *Transformer Circuits Thread*, 2025. URL https://transformer-circuits.pub/2025/attribution-graphs/biology.html.

Llama-Team. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

David Lopez-Paz, Léon Bottou, Bernhard Schölkopf, and Vladimir Vapnik. Unifying distillation and privileged information. *arXiv preprint arXiv:1511.03643*, 2015.

METR. Details about metr's evaluation of openai gpt-5. https://metr.org/evaluations/gpt-5-report/, 08 2025.

METR. Brief independent investigation of agents' behavior, reasoning and collaboration in the openai / hugging face hacking incident. https://metr.org/blog/2026-08-26-openai-hugging-face-incident-investigation/, 08 2026.

NVIDIA. Llama-nemotron: Efficient reasoning models, 2025. URL https://arxiv.org/abs/2505.00949.

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

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, 2021.

Hayden Prairie, Zachary Novack, Taylor Berg-Kirkpatrick, and Daniel Y. Fu. Parcae: Scaling laws for stable looped language models, 2026. URL https://arxiv.org/abs/2604.12946.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets, 2015. URL https://arxiv.org/abs/1412.6550.

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.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with latent thoughts: On the power of looped transformers, 2025. URL https://arxiv.org/abs/2502.17416.

Bronson Schoen, Evgenia Nitishinskaya, Mikita Balesni, Axel Højmark, Felix Hofstätter, Jérémy Scheurer, Alexander Meinke, Jason Wolfe, Teun van der Weij, Alex Lloyd, Nicholas Goldowsky-Dill, Angela Fan, Andrei Matveiakin, Rusheb Shah, Marcus Williams, Amelia Glaese, Boaz Barak, Wojciech Zaremba, and Marius Hobbhahn. Stress testing deliberative alignment for anti-scheming training, 2025. URL https://arxiv.org/abs/2509.15541.

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. Codi: Compressing chain-of-thought into continuous space via self-distillation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 677–693, 2025.

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning. *arXiv preprint arXiv:2601.19897*, 2026.

Charlie Snell, Dan Klein, and Ruiqi Zhong. Learning by distilling context. *arXiv preprint arXiv:2209.15189*, 2022.

Ryan Solgi, Jiayi Tian, and Zheng Zhang. Lori: Low-rank distillation for implicit reasoning. *arXiv preprint arXiv:2606.05315*, 2026.

Zayne Sprague, Fangcong Yin, Juan Diego Rodriguez, Dongwei Jiang, Manya Wadhwa, Prasann Singhal, Xinyu Zhao, Xi Ye, Kyle Mahowald, and Greg Durrett. To cot or not to cot? chain-of-thought helps mainly on math and symbolic reasoning, 2025. URL https://arxiv.org/abs/2409.12183.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2023. URL https://arxiv.org/abs/2104.09864.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge, 2019. URL https://arxiv.org/abs/1811.00937.

Wenhui Tan, Jiaze Li, Jianzhong Ju, Zhenbo Luo, Ruihua Song, and Jian Luan. Think silently, think fast: Dynamic latent compression of llm reasoning chains. *Advances in Neural Information Processing Systems*, 38:4646–4668, 2026.

K2 Team, Zhengzhong Liu, Liping Tang, Linghao Jin, Haonan Li, Nikhil Ranjan, Desai Fan, Shaurya Rohatgi, Richard Fan, Omkar Pangarkar, Huijuan Wang, Zhoujun Cheng, Suqi Sun, Seungwook Han, Bowen Tan, Gurpreet Gosal, Xudong Han, Varad Pimpalkhute, Shibo Hao, Ming Shan Hee, Joel Hestness, Haolong Jia, Liqun Ma, Aaryamonvikram Singh, Daria Soboleva, Natalia Vassilieva, Renxi Wang, Yingquan Wu, Yuekai Sun, Taylor Killian, Alexander Moreno, John Maggs, Hector Ren, Guowei He, Hongyi Wang, Xuezhe Ma, Yuqi Wang, Mikhail Yurochkin, and Eric P. Xing. K2-v2: A 360-open, reasoning-enhanced llm, 2026. URL https://arxiv.org/abs/2512.06201.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting, 2023. URL https://arxiv.org/abs/2305.04388.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

Xilin Wei, Xiaoran Liu, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Jiaqi Wang, Xipeng Qiu, and Dahua Lin. Sim-cot: Supervised implicit chain-of-thought. In *International Conference on Learning Representations*, volume 2026, pp. 56721–56742, 2026.

Ronald J. Williams and Jing Peng. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural Computation*, 2:490–501, 1990. URL https://api.semanticscholar.org/CorpusID:12979634.

Haoyi Wu, Zhihao Teng, and Kewei Tu. Parallel continuous chain-of-thought with jacobi iteration. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 914–926, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. Star: Bootstrapping reasoning with reasoning, 2022. URL https://arxiv.org/abs/2203.14465.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. *arXiv preprint arXiv:2601.18734*, 2026.

Fan Zhou, Zengzhi Wang, Nikhil Ranjan, Zhoujun Cheng, Liping Tang, Guowei He, Zhengzhong Liu, and Eric P. Xing. Megamath: Pushing the limits of open math corpora, 2025. URL https://arxiv.org/abs/2504.02807.

Appendix

A Extended Related Work

Supervision for latent reasoning. Beyond the methods discussed in the main text, several approaches explore other ways to derive latent supervision from explicit reasoning. At one extreme, pause tokens append learnable dummy positions before the answer and train them through the answer loss alone, with no explicit reasoning involved (Goyal et al., 2024). Implicit CoT compresses reasoning into model depth by training an emulator whose successive layers match teacher activations at progressively later, evenly spaced positions in the textual reasoning trace (Deng et al., 2023). CCoT uses a pretrained scorer to select a small subset of activations from the full CoT and trains a model to autoregressively generate continuous states that approximate them (Cheng & Van Durme, 2024). Other methods supervise local reconstructions of the trace: SIM-CoT uses an auxiliary decoder to reconstruct the corresponding reasoning step from each latent state (Wei et al., 2026), while CoLaR predicts compressed embeddings formed by merging consecutive CoT tokens (Tan et al., 2026). KaVa provides trajectory-level supervision by distilling a compressed key–value cache of the explicit reasoning trace into the latent branch (Kuzina et al., 2026). Echo instead directly supervises latent slots with the shared model's explicit-CoT hidden states, resampling their positions throughout training while preserving chronological order.

Teacher-student distillation. Knowledge distillation traditionally trains a student to match a pretrained teacher's output distribution (Hinton et al., 2015), while feature-based variants align intermediate representations (Romero et al., 2015). Another common form of teacher-student distillation for language models is to generate synthetic data with reasoning from larger models and to train smaller models (Ho et al., 2023; Hsieh et al., 2023; Yang et al., 2025; Llama-Team, 2024).

B Distillation target selection policy

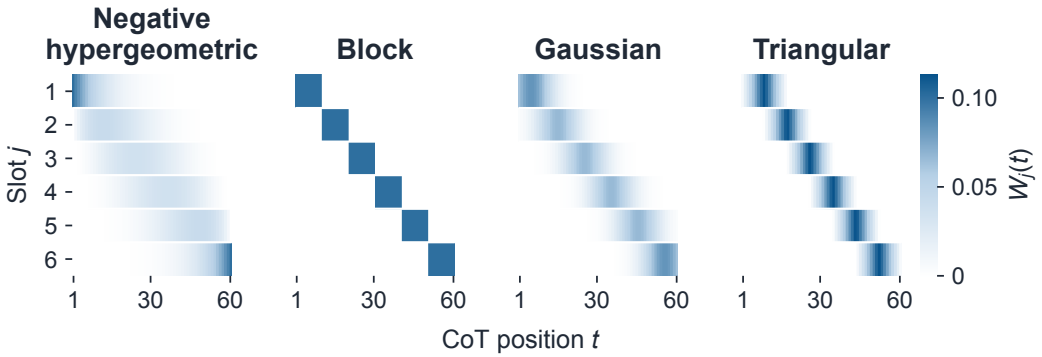


Figure S1: Per-slot distributions $W_j(t)$ of the four target selection policies for $K = 6$ slots and a CoT of $n = 60$ tokens. Each row gives the probability that slot j is supervised by position t; all panels share one color scale.

A target selection policy gives each latent slot a distribution over CoT positions. All four policies use the procedure below to draw positions jointly in increasing order, differing only in their per-slot distributions.

Notation. For the i-th example with a CoT of length n_i, we supervise the first $k_i = \min(K, n_i)$ latent slots. Slot j receives the target $z_{ij}^{(\ell)} = \mathrm{sg}(h_{i,t_{ij}}^{(\ell)})$ at each supervised layer ℓ, with ordered positions $1 \le t_{i1} < \cdots < t_{ik_i} \le n_i$ shared across layers. We omit the example index i below and write n, k, and t_j. A policy assigns slot j a distribution W_j over $\{1, \ldots, n\}$, with cumulative distribution function $F_j(t) = \sum_{p=1}^{t} W_j(p)$ and $F_j(0) = 0$.

Per-slot distributions. Each policy defines slot-specific probabilities $W_j(t)$ over CoT positions $t = 1, \ldots, n$. Table S1 lists these probability distributions, and Figure S1 illustrates them for $K = 6$ slots and $n = 60$ tokens. For the Gaussian and triangular policies, $W_j(t)$ is the mass of the

Table S1: Target-position probabilities for slot $j = 1, \ldots, k$, with $1 \le k \le n$ and $t = 1, \ldots, n$. We use block boundaries $e_j = \text{round}(jn/k)$ (ties to even), Gaussian centers $\mu_j = (j - \frac{1}{2})n/k$ and standard deviation $\sigma = 0.6n/k$, and triangular spacing $b = n/(k+1)$. Φ is the standard normal CDF, and H_j is the CDF of the triangular distribution on $[(j-1)b, (j+1)b]$ with peak at jb. Binomial coefficients are zero outside their valid range.

Policy	**Position probability** $W_j(t)$	**Description**
Negative-hypergeometric	$\frac{\binom{t-1}{j-1}\binom{n-t}{k-j}}{\binom{n}{k}}$	The j-th smallest position in a uniformly sampled size-k subset.
Block	$\frac{\mathbf{1}\{e_{j-1} < t \le e_j\}}{e_j - e_{j-1}}$	Equal probability within the j-th block; zero elsewhere.
Gaussian	$\frac{\Phi((t-\mu_j)/\sigma) - \Phi((t-1-\mu_j)/\sigma)}{\Phi((n-\mu_j)/\sigma) - \Phi(-\mu_j/\sigma)}$	Concentrated near the middle of the j-th block.
Triangular	$H_j(t) - H_j(t-1)$	Density peaks at jb and decreases linearly to zero at both endpoints.

corresponding continuous distribution over $(t-1, t]$, normalized over $(0, n]$. Positions are drawn jointly using the ordered sampling procedure below.

Drawing positions in order. Independent draws can violate slot order, while simply truncating and renormalizing W_j to positions after t_{j-1} generally fails to preserve the original marginals. Our sampler is inspired by Butucea et al. (2015, Eqs. (54)–(55), Theorem 5.4), which derives a joint density that maximizes entropy subject to ordering constraints and specified continuous marginals. Their result does not cover discrete positions; we establish marginal preservation separately, without claiming maximum entropy. We draw $t_1 \sim W_1$ and, for each slot $j \ge 2$, scan forward from $t_{j-1} + 1$, stopping at position p with probability

$$\lambda_j(p) = \frac{W_j(p)}{F_{j-1}(p-1) - F_j(p-1)}. \tag{B.1}$$

We need $\lambda_j(p) \le 1$, which is equivalent to $F_j(p) \le F_{j-1}(p-1)$ for all p and $j \ge 2$. Per-slot distributions produced by any ordered draw satisfy this: whenever slot j is at or before position p, slot $j-1$ is before p, so the first event is never more likely than the second. Under this condition, a zero denominator implies $W_j(p) = 0$ and an unreachable position, where we set $\lambda_j(p) = 0$.

Assume $n > K$, since every policy uses all CoT positions for $n \le K$. The negative hypergeometric, block, and triangular policies satisfy the condition, but the Gaussian violates it: its tails never reach zero, so it puts probability on positions a slot can never occupy, such as slot 2 on position 1. For the Gaussian, we let each slot also stop at its predecessor's position, replacing $F_{j-1}(p-1)$ with $F_{j-1}(p)$ in Eq. B.1. This reproduces every W_j, since it needs only $F_j(p) \le F_{j-1}(p)$, but can place two slots at the same position. We thus discard draws with ties, and the resulting per-slot distributions only approximate W_j.

When the condition holds, we verify marginal preservation by induction over slots and positions, starting from $t_1 \sim W_1$. For the induction step, suppose $\Pr(t_{j-1} = q) = W_{j-1}(q)$ for all q and $\Pr(t_j = q) = W_j(q)$ for $q < p$. The scan reaches p with probability $F_{j-1}(p-1) - F_j(p-1)$: the predecessor lies before p, excluding outcomes where slot j has already stopped. At reachable positions, selecting p requires reaching it and then stopping there:

$$\begin{aligned} \Pr(t_j = p) &= \Pr(\text{reach } p)\ \Pr(\text{stop at } p \mid \text{reach } p) \\ &= [F_{j-1}(p-1) - F_j(p-1)]\lambda_j(p) = W_j(p), \end{aligned} \tag{B.2}$$

and unreachable positions have $\Pr(t_j = p) = 0 = W_j(p)$. This establishes the induction step and hence the required marginals.

Resampling, short traces, and expected targets. Sampled positions are redrawn at every optimizer step from a generator seeded by the step and the example; the controls in Figure 4b instead fix the

draw (*no target resampling*) or randomly permute which slot receives each position (*shuffled targets*). When $n_i \le K$, sampled positions are $(1, \dots, n_i)$ under every policy, and the remaining $K - n_i$ slots receive no target and are left out of both the distillation loss and the target standard deviation that normalizes it. Expected targets instead replace the sampled state by its average under the slot's distribution, $z_j^{(\ell)} = \sum_{t=1}^{n} W_j(t)\,\mathrm{sg}(h_t^{(\ell)})$, which is the same at every step.

C Dense trajectory distillation

Here we describe the alignment-head architecture and the self-distillation objective from Section 3.

Alignment heads. Each supervised Transformer block has an independent alignment head consisting of layer normalization followed by a bias-free linear projection:

$$q_\ell(u) = W_\ell\,\mathrm{LN}_\ell(u), \qquad W_\ell \in \mathbb{R}^{d\times d}. \tag{C.1}$$

Here LN_ℓ denotes the layer-normalization operation for block ℓ. The alignment heads are jointly optimized with the shared decoder and learned slot embedding.

Normalized Smooth L1 loss. For vectors $a, b \in \mathbb{R}^d$, we define

$$\mathcal{D}_{\ell,s}(a,b) = \frac{1}{\sigma_{\ell,s} d}\sum_{h=1}^{d}\rho(a_h - b_h), \tag{C.2}$$

where ρ is the Smooth L1 penalty with unit transition:

$$\rho(\delta) = \begin{cases} \frac{1}{2}\delta^2, & |\delta| \le 1, \\ |\delta| - \frac{1}{2}, & |\delta| > 1. \end{cases} \tag{C.3}$$

The factor $1/d$ averages over hidden dimensions, while $\sigma_{\ell,s}$ normalizes the loss by the scale of the targets at the corresponding block and optimizer step. Normalization is applied to the loss after evaluating ρ.

Target normalization. The normalizer $\sigma_{\ell,s}$ estimates the scale of the block-ℓ targets at optimizer step s. It is the sample standard deviation over all scalar coordinates of the selected, detached block-ℓ targets in the batch. Let $\alpha_{\ell,s} \in \mathbb{R}^{N_s}$ collect these scalar values. For $N_s > 1$, we compute

$$\mu_{\ell,s} = \frac{1}{N_s}\sum_{a=1}^{N_s}\alpha_{\ell,s,a}, \qquad \sigma_{\ell,s} = \left(\frac{1}{N_s - 1}\sum_{a=1}^{N_s}(\alpha_{\ell,s,a} - \mu_{\ell,s})^2\right)^{1/2}. \tag{C.4}$$

Because the target states are detached, gradients do not propagate through either the targets or their normalization statistics.

Expanded objective. Let $\Omega_s = \{(i,j) : i \in \mathcal{B}_s,\ 1 \le j \le k_i\}$ denote the supervised slot–target pairs in the batch $\mathcal{B}_s$ at step s, and let $\mathcal{J} = \{L - M + 1, \dots, L\}$ denote the M supervised blocks. Substituting Eq. C.2 into Eq. 3.4 gives

$$\mathcal{L}_{\text{distill}} = \frac{1}{M}\sum_{\ell\in\mathcal{J}}\frac{1}{\sigma_{\ell,s}|\Omega_s|d}\sum_{(i,j)\in\Omega_s}\sum_{h=1}^{d}\rho\Big(q_\ell(u_{ij}^{(\ell)})_h - z_{ij,h}^{(\ell)}\Big). \tag{C.5}$$

Thus, the objective averages over supervised blocks, slot–target pairs, and hidden dimensions. When $|\Omega_s| = 0$, we set $\mathcal{L}_{\text{distill}} = 0$ without evaluating the pairwise losses.

C.1 Dynamic Recurrence Experiments

In the dynamic recurrence setting in Section 4.4.4, we use GPT-2 with $K = 10$ latent slots and $M = 5$ supervised layers. For each training microbatch, we sample the total number of refinement passes as

$$Z \sim \mathrm{Poisson}(5), \qquad R = 1 + \min\{11, \max\{1, Z\}\}.$$

This gives approximately six passes on average, with a minimum of two and a maximum of twelve, including the first pass.

The first pass follows Eq. (3.2), producing $U^{(1,L)}$ from the learned slot embeddings E. For the remaining $R - 1$ passes, we replace the additive reinjection of E with a learned linear injection of

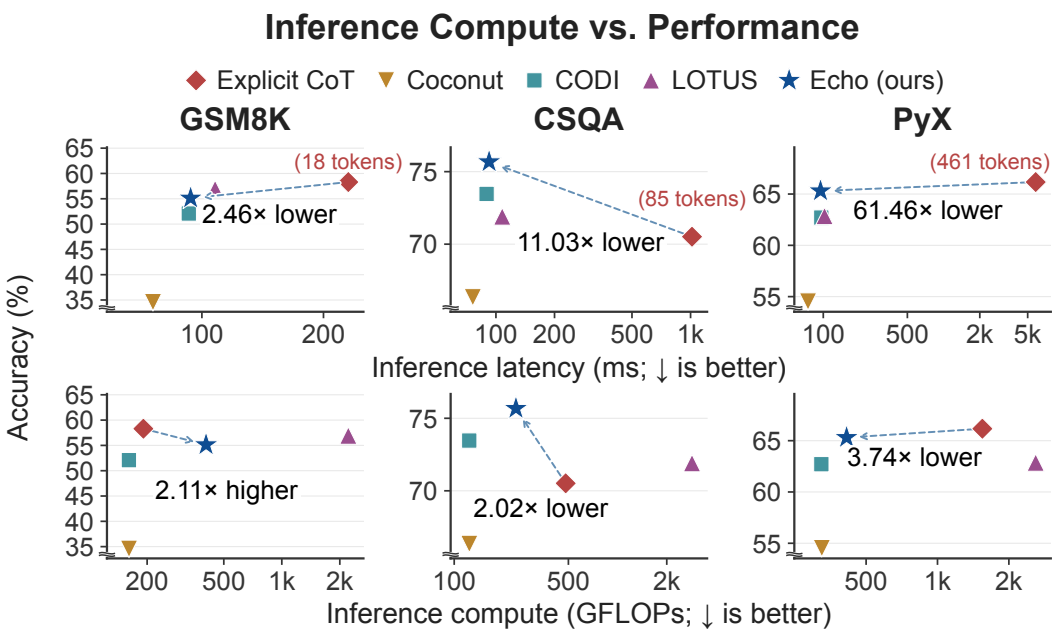


Figure S2: Accuracy vs. inference latency (top) and compute (bottom) for Llama-3.2-1B-Instruct. Echo lies on the accuracy–efficiency Pareto frontier (upper left) in every panel except GSM8K compute, running 2.5–61× faster than explicit CoT with 2.0–3.7× fewer GFLOPs on CSQA and PyX. On each of the Explicit CoT model markers, we annotate the mean reasoning generation length on each dataset. We observe that Echo compute efficiency scales with generation length.

$U^{(1,L)}$. For each slot, we concatenate its recurrent state with its corresponding first-pass state and apply a shared, bias-free linear layer $W \in \mathbb{R}^{d\times 2d}$, initialized as $[I; I]$.

Before the second pass, we initialize each slot's recurrent state with zero-mean Gaussian noise, scaled per hidden dimension by the standard deviation of $U^{(1,L)}$ across the batch and slot positions. The second pass combines this initial state with the corresponding first-pass state. Each subsequent pass combines the preceding pass's final-layer state with the same first-pass state, without adding further noise. These inputs are processed jointly by f_θ, always starting from C_{prefix} and retaining the same causal slot mask.

For optimization stability at higher R, we use truncated backpropagation through time (Williams & Peng, 1990; Hochreiter & Schmidhuber, 1997), backpropagating through the last $\min\{5, R-1\}$ passes after the first pass. At inference, we evaluate at $R \in \{1, \ldots, 12\}$ and compare against separately trained fixed-R models, with depths up to $R = 8$ shown in Figure 5.

D Additional Experimental Results

D.1 Llama-3.2-1B Instruct Inference Compute vs. Performance

We analyze the inference latency (in ms) and analytical compute for decoding (in GFLOPs) in Llama-3.2-1B models, as we have done for GPT-2 in Figure 3. We observe similar gains of up to about 61× speedup in latency and 3.7× lower GFLOPs. The difference in the speedup and compute efficiency stems from the varying reasoning generation lengths across the three datasets, as shown in Table S2. With longer reasoning traces from the Explicit CoT model, the better Echo scales in compute efficiency.

D.2 Llama-3.2-1B Loss Ablation

In Figure S3, we ablate the three losses in our Echo framework: $\mathcal{L}_{\text{answer}}$, $\mathcal{L}_{\text{distill}}$, and $\mathcal{L}_{\text{explicit}}$. As with GPT-2 shown in Figure 4a, we find each of the three losses to be important, and, of the three, $\mathcal{L}_{\text{explicit}}$ is the most important for anchoring the explicit CoT branch and the hidden states so that they are on-policy with respect to the model and meaningful targets to distill from.

Table S2: Mean reasoning generation length (in tokens) for each explicit CoT model and dataset.

Model	GSM8K	CommonsenseQA	PyX Forward
GPT-2	17.47	66.52	460.05
Llama-3.2-1B Instruct	17.84	85.49	460.77

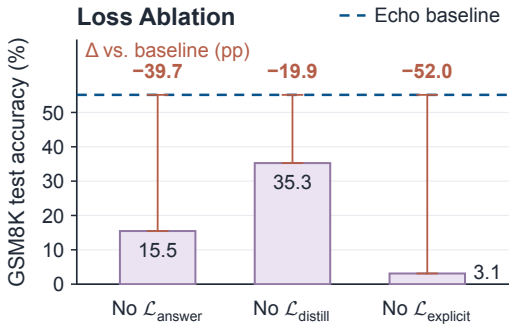


Figure S3: **Ablations of the answer loss, distillation loss, and explicit CoT loss on GSM8K with Llama-3.2-1B Instruct**. As with GPT-2, we observe that each of the three losses is crucial in our Echo framework. Dashed lines indicate the Echo baseline, and annotated deltas indicate the change in test accuracy.

D.3 Llama-3.2-1B Teacher-state Order

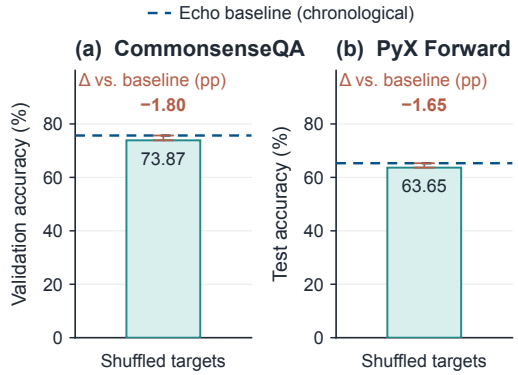


Figure S4: Effect of teacher-state assignment order with Llama-3.2-1B-Instruct on (a) CommonsenseQA validation and (b) PyX Forward test. Bars show shuffled assignment; dashed lines mark chronological baselines. Annotations indicate accuracy changes in percentage points.

We study the importance of preserving the temporality of the teacher hidden states during latent distillation in Llama-3.2-1B models in Figure S4, as we did for GPT-2 in Figure 4b. As with GPT-2, disrupting the temporality of the teacher hidden states by shuffling them hurts performance, which shows that aligning successive latent slots with earlier-to-later portions of the reasoning is important.

D.4 Llama-3.2-1B Target Selection Policy

We study the different target selection policies using both sampled and expected approaches described in Appendix B, as we did in Section 4.4.3 with the GPT-2 results in Table 4. Similarly, we observe that `Random-Sorted`, despite its simplicity, performs the best on CSQA and PyX datasets and that `Gaussian` performs the best in GSM8K.

Table S3: Target selection policy ablation with Llama-3.2-1B-Instruct. *Sampled* policies draw targets from the policy distribution in chronological order. *Expected* policies average the explicit-CoT hidden states weighted by each slot's position probabilities. Bold marks the best setting per dataset.

Targets	Policy	GSM8K-Aug	CommonsenseQA	PyX Forward
Sampled	Random-Sorted (Negative-hypergeometric)	49.36	**75.68**	**65.30**
	Triangular	49.51	73.46	63.81
	Gaussian	50.34	73.71	62.08
	Block	49.73	73.96	64.67
Expected	Triangular	52.24	74.12	60.82
	Gaussian	**55.12**	75.02	64.44
	Block	47.38	73.63	63.10

E Additional Details on Experimental Setup

E.1 Datasets

E.1.1 GSM8K-Aug

We use GSM8K-Aug with structured arithmetic CoTs from Deng et al. (2023). We first discard examples with missing fields and answers that do not begin with a digit. We select for examples whose question, CoT, and answer are below 256 tokens. This retains 383,854 training examples for both GPT-2 and Llama.

E.1.2 CommonsenseQA

We use the training split augmented with GPT-4o-mini rationales from Shen et al. (2025). Prompts contain the question and five multiple choices; answers are the corresponding answer. We retain all 8,196 augmented training examples, which fit within the 512-token limit.

E.1.3 PyX Forward

From the full PyX dataset (Ding et al., 2024), we extract forward-execution examples, where the task is to simulate the output of the program given a function call. We use the supplied monologues as CoT and assertion outputs as answers. We reject malformed assertions, mismatched function calls, and unsupported answer literals, and keep program groups disjoint across splits. We exclude examples with CoTs longer than 576 tokens or complete references longer than 1,000 tokens, without truncation. Tokenizer-dependent filtering leaves 7,496 training examples for GPT-2 and 10,685 for Llama.

E.2 Training Details

Backbones and hyperparameter tuning. We conduct experiments with GPT-2 and Llama-3.2-1B Instruct. Preliminary tuning considered effective global batch sizes $B \in \{128, 256, 512\}$ and method and backbone-specific learning-rate grids, $LR_{\text{GPT-2}} \in \{3 \times 10^{-4}, 1 \times 10^{-3}, 3 \times 10^{-3}\}$ and $LR_{\text{Llama}} \in \{3 \times 10^{-4}, 8 \times 10^{-4}, 1 \times 10^{-3}\}$. We then fix the adopted learning rate and batch size for each method–backbone–dataset combination while varying its latent-reasoning configuration. We report the training hyperparameters selected after the sweep for each dataset and model in Tables S4 to S6.

Echo. We train the explicit and latent branches jointly, assigning unit weight to each answer/reference cross-entropy term. We use AdamW with cosine learning-rate decay, a 3% warmup ratio, weight decay 0.1, and gradient clipping at norm 2. LoRA adapters have rank 128, scaling parameter $\alpha = 32$, and dropout 0.1. The embedding layers are frozen except for the added <BoT>, <EoT>, and latent slot embeddings. Otherwise, all other weights are trained via LoRA. The base configuration sweep varies the number of latent slots $K \in \{2, 4, 6, 8, 10, 20\}$ and the number of supervised final transformer layers $L \in \{1, 2, 3, 4, 5, 6, \text{all}\}$, with six refinement passes ($R = 6$). Here, "all" denotes 12 layers for GPT-2 and 16 for Llama. Random CoT-token targets are resampled during training and ordered chronologically. Matching losses are averaged across supervised layers, with one alignment head per layer. The final trajectory-loss weights after a hyperparameter sweep for GSM8K-Aug, CommonsenseQA, and PyX are respectively $(1, 1, 10)$ for GPT-2 and $(20, 10, 15)$ for Llama. GPT-2 uses a mini-batch size of 512 across datasets. Llama uses mini-batch size of 128 on GSM8K-Aug, CommonSenseQA, and PyX.

Table S4: GSM8K-Aug training settings. Batch size is the effective global batch size. Epochs denote the full training budget; LoRi epochs are passes over its 64-example student-training subset. Llama denotes Llama-3.2-1B Instruct.

Backbone	Method	Learning rate	Batch size	Epochs
GPT-2	Answer-only SFT	3×10^{-3}	512	40
	Explicit CoT	3×10^{-3}	128	40
	Echo	3×10^{-3}	512	40
	CODI	3×10^{-3}	128	40
	LOTUS	10^{-4}	128	30
	Coconut	3×10^{-4}	128	50
	LoRi	3×10^{-4}	16	1
Llama	Answer-only SFT	10^{-4}	128	10
	Explicit CoT	8×10^{-4}	128	10
	Echo	8×10^{-4}	128	10
	CODI	8×10^{-4}	128	10
	LOTUS	10^{-4}	128	30
	Coconut	3×10^{-4}	128	10
	LoRi	2×10^{-5}	16	1

Table S5: CommonsenseQA training settings. Batch size is the effective global batch size. Epochs denote the full training budget; LoRi epochs are passes over its 64-example student-training subset. Llama denotes Llama-3.2-1B Instruct.

Backbone	Method	Learning rate	Batch size	Epochs
GPT-2	Answer-only SFT	3×10^{-3}	512	50
	Explicit CoT	3×10^{-3}	512	50
	Echo	3×10^{-3}	512	50
	CODI	3×10^{-3}	512	50
	LOTUS	10^{-4}	128	30
	Coconut	10^{-4}	128	50
	LoRi	3×10^{-4}	16	1
Llama	Answer-only SFT	10^{-4}	128	10
	Explicit CoT	3×10^{-4}	128	10
	Echo	8×10^{-4}	128	10
	CODI	8×10^{-4}	128	10
	LOTUS	10^{-4}	128	30
	Coconut	10^{-4}	128	9
	LoRi	2×10^{-5}	16	1

The final Echo configurations use $(K, L) = (10, 5), (10, 6), (6, 5)$ for GPT-2 and $(10, 1), (10, 6), (6, 2)$ for Llama, in GSM8K-Aug, CommonsenseQA, and PyX respectively. All use $R = 6$. GPT-2 GSM8K-Aug removes the final CoT step; the other five configurations retain it. Llama GSM8K-Aug uses Gaussian expected targets with width parameter 0.6; the other five use sampled random CoT-token targets.

Explicit CoT. Explicit CoT uses LoRA rank 128 and $\alpha = 32$, with recipe-specific dropout of 0.05 or 0.1, and the same AdamW, cosine decay, warmup, weight-decay, and clipping settings as in Echo.

CODI. We follow the recipe suggested by the original authors (Shen et al., 2025). We sweep $K \in \{2, 4, 6, 8, 10\}$ latent steps, varying distillation loss coefficients, and remove the whole final CoT step on both backbones as recommended by the original paper (Shen et al., 2025). CODI uses LoRA rank 128 and $\alpha = 32$, with recipe-specific dropout of 0.05 or 0.1, and the same AdamW, cosine decay, warmup, weight-decay, and clipping settings as in Echo. We retain its projected, teacher-standard-deviation-normalized Smooth-L1 distillation objective and unit reference-loss weight.

Table S6: PyX Forward training settings. Batch size is the effective global batch size. Epochs denote the full training budget; LoRi epochs are passes over its 64-example student-training subset. Llama denotes Llama-3.2-1B Instruct.

Backbone	Method	Learning rate	Batch size	Epochs
GPT-2	Answer-only SFT	3×10^{-3}	512	50
	Explicit CoT	5×10^{-3}	128	50
	Echo	3×10^{-3}	512	50
	CODI	3×10^{-3}	512	50
	LOTUS	10^{-4}	128	30
	Coconut	3×10^{-4}	128	50
	LoRi	3×10^{-3}	16	4
Llama	Answer-only SFT	8×10^{-4}	128	10
	Explicit CoT	10^{-3}	128	10
	Echo	8×10^{-4}	128	10
	CODI	10^{-3}	128	10
	LOTUS	3×10^{-4}	128	30
	Coconut	10^{-4}	128	9
	LoRi	10^{-4}	16	1

LOTUS. We follow the training recipe recommended by the original authors (Fan et al., 2026), initialize LOTUS from task-specific explicit-CoT checkpoints and train for 30 epochs with six latent blocks and six refinement passes. We sweep block width $c \in \{1, 5, 10, 25, 30\}$, yielding $6c$ latent positions, while keeping $R = 6$ for a fair comparison. Supervision truncates each reference block to its allocated width. LOTUS uses AdamW with a constant learning rate, weight decay 0.01, no warmup, and gradient clipping at norm 1. We use full fine-tuning for GPT-2 on all tasks and Llama on GSM8K-Aug; Llama CommonsenseQA and PyX use frozen-head LoRA with rank 128, $\alpha = 32$, and dropout 0.1.

Coconut. We use progressive replacement of explicit reasoning steps by continuous latent states. The learning-rate search uses $\{10^{-4}, 3 \times 10^{-4}, 10^{-3}\}$ at global batch size 128. Training uses constant-rate AdamW with weight decay 0.01 and LoRA rank 128 and $\alpha = 32$ for the transformer and vocabulary adapters. GPT-2 trains for 50 epochs with six final latent positions. For GSM8K-Aug, it uses six explicit-CoT warmup epochs and three epochs per replacement stage, with two latent positions per replaced step. Llama uses three warmup epochs and one epoch per replacement stage: the original GSM8K-Aug recipe has three latents and seven total epochs, while CommonsenseQA and PyX have six latents and nine epochs. The GSM8K-Aug recipe retains full CoTs, initializes latent marker embeddings after CoT warmup, and uses zero LoRA dropout; CommonsenseQA and PyX use dropout 0.1. Checkpoint selection is restricted to the terminal, fully latent curriculum stage. Additional Llama experiments vary $K \in \{2, 4, 6, 8, 10\}$, with $K + 4$ epochs on GSM8K-Aug and $K + 3$ on the other two tasks, preserving the curriculum pace.

LoRi. We follow the training recipe from Solgi et al. (2026). We initialize each student from its corresponding explicit-CoT teacher and use four latent steps. Following the adopted small-subset protocol, student distillation uses 64 training examples and an effective batch size of 16 (microbatch 1 with 16 gradient-accumulation steps). We also ablate with 128 training examples and find 64 to be generally better. This gives four optimizer updates, except for GPT-2 PyX, which uses 16 updates over four passes through the subset. Student learning rates on GSM8K-Aug, CommonsenseQA, and PyX are respectively $(3 \times 10^{-4}, 3 \times 10^{-4}, 3 \times 10^{-3})$ for GPT-2 and $(2 \times 10^{-5}, 2 \times 10^{-5}, 10^{-4})$ for Llama. Our LoRi students use LoRA rank 128, $\alpha = 32$, and dropout 0.1. Optimization uses AdamW with linear learning-rate decay, a 3% warmup ratio, zero weight decay, and gradient clipping at norm 1.