The Illusion of Superposition? A Principled Analysis of Latent Thinking in Language Models

Michael Rizvi-Martel[1,2]* **Guillaume Rabusseau**[1,2] **Marius Mosbach**[1,3]

[1]Mila – Quebec AI Institute [2]Université de Montréal [3]McGill University

Abstract

Latent reasoning via continuous chain-of-thoughts (Latent CoT) has emerged as a promising alternative to discrete CoT reasoning. Operating in continuous space increases expressivity and has been hypothesized to enable *superposition*: the ability to maintain multiple candidate solutions simultaneously within a single representation. Despite theoretical arguments, it remains unclear whether language models actually leverage superposition when reasoning using latent CoTs. We investigate this question across three regimes: a *training-free* regime that constructs latent thoughts as convex combinations of token embeddings, a *fine-tuned* regime where a base model is adapted to produce latent thoughts, and a *from-scratch* regime where a model is trained entirely with latent thoughts to solve a given task. Using Logit Lens and entity-level probing to analyze internal representations, we find that only models trained *from scratch* exhibit signs of superposition. In training-free and fine-tuned regimes, superposition either collapses or is not used at all, with models discovering shortcut solutions instead. We propose mechanisms explaining both the failure of training-free approaches and the limitations of training latent reasoners from scratch, finding that: i) in the training-free case, pretraining biases models to commit to a single token, collapsing the superposition; ii) in the from-scratch regime, success is governed by embedding width rather than depth. Together, our results shed light on when and why superposition arises in continuous chain-of-thought reasoning, and propose hypotheses as to conditions which drive its collapse.

1 Introduction

Chain-of-thought (CoT) reasoning has become the standard approach for tackling complex problems with large language models (LLMs), enabling them to break down problems by reasoning "step-by-step" (Wei et al., 2022). Various works have tried to further improve LLM reasoning through methods like self-consistency (Wang et al., 2023), tree-of-thoughts (Yao et al., 2023), and stream-of-search (Gandhi et al., 2024). Post-training LLMs on CoT data is now a crucial part of the LLM training pipeline (OpenAI, 2025; Guo et al., 2025). Exploring an alternative approach for reasoning, recent work proposed to have LLMs reason *directly in latent space* (Hao et al., 2025; Butt et al., 2026), which has been shown theoretically to be more expressive than discrete CoT (Zhu et al., 2025). A compelling hypothesis for latent CoT's advantage is *superposition*: models could maintain multiple candidate solutions simultaneously, exploring several reasoning paths before committing to an answer (Zhu et al., 2025). This would represent a fundamental advantage over discrete CoT, which must commit to a single token at each step. However, there is little empirical evidence that LLMs leverage this capability, motivating the following research question:

Does superposition actually occur in latent CoT models?

We investigate this question across three complementary settings. First, we analyze Soft Thinking (Zhang et al., 2025), a training-free latent CoT method that creates superposition

*Corresponding author. Contact: michael.rizvi-martel@mila.quebec

by computing linear combinations of input embeddings. Second, we examine Coconut (Hao et al., 2025), a method that fine-tunes models to reason with continuous latent thoughts. Lastly, we analyze a from-scratch variant of Coconut, where a small GPT-2-style model is entirely trained with latent thoughts to solve a given task.

Empirically, we make the following contributions: First, using Logit Lens (Nostalgebraist, 2020) to probe internal representations, we find that **off-the-shelf LLMs collapse superposed inputs to a single interpretation** within the first few layers: when comparing with a discrete CoT baseline, entropy profiles are nearly identical. Moreover, replacing a soft token with a standard one yields nearly indistinguishable KL divergence and cosine similarity. Second, through entity-level probing, we show that **the Coconut model learns to extract answers directly from the question representations**, achieving comparable accuracy *without any latent tokens*. Our belief evolution analysis explains this: we find that models do not leverage step-by-step reasoning during latent computation. Third, when applying the same Coconut analysis to a model trained from scratch, we find that it **indeed shows signs of leveraging superposition**: the model encodes uncertainty between the correct next step and other possible next steps within its latent thoughts. Together, these results suggest that *training from scratch* is the most conducive mechanism to develop superposition in models.

Based on these results, we conduct an additional set of experiments aimed at better understanding this discrepancy. This analysis yields two distinct findings:

1. **Models trained on next-token prediction learn to commit to a token in the last layers.** We find that across all considered training-free and fine-tuned models, entropy of the logit distributions drops heavily at the last layers. This drop is much less significant in from-scratch models.
2. **Embedding size matters.** We find that from-scratch models suffer from a *width bottleneck*: larger embedding widths have a much more pronounced effect than depth on accuracy on tasks which require parallel exploration.

We believe our work highlights important caveats of current latent thinking methodologies and offers principled guidelines to design the next generation of latent reasoning models.

2 Related Work

Latent Reasoning. Many works investigate the use of continuous tokens and latent representations in LLMs. Hao et al. (2025) show that fine-tuning LLMs to output a reasoning trace of continuous tokens provides considerable gains on logical reasoning tasks that require search during planning. Zhu et al. (2025) popularized the notion of "superposition" by showing theoretically that superposition in the latent state allows transformer models to solve graph reachability tasks more efficiently. Follow-up work by Butt et al. (2026) proposes a novel method to train continuous CoTs via reinforcement learning which achieves comparable performance to discrete CoT on known math reasoning benchmarks. We base our analysis on the "Soft Thinking" approach by Zhang et al. (2025); a training-free method to generate latent CoTs based on convex combinations of embedding vectors. They report that their method offers a slight improvement on math benchmarks compared to discrete CoT baselines. Also close to our work, Deng et al. (2025) claim to have devised a training scheme which enables superposition in LLMs.

Alternative continuous thinking schemes have also been explored. Many works investigate the use of "filler" or "thinking" tokens: blank tokens which can be used to store intermediary computations (Pfau et al., 2024; Goyal et al., 2024; Herel & Mikolov, 2024). Moreover, there is a growing interest in "looped layers", a method which trains transformers with recurrent attention layers (Yang et al., 2024b; McLeish et al., 2025). These methods also show promise in increasing reasoning abilities through additional latent computation.

Interpretability of Reasoning Models. Understanding the internal representations of models has been a central research topic in NLP, even prior to LLMs (Adi et al., 2017; Linzen et al., 2016; Gulordava et al., 2018; Belinkov, 2022, *inter alia*). Recently, several works investigated how CoT reasoning changes internal computations (Yang et al., 2024c; Dutta et al.,

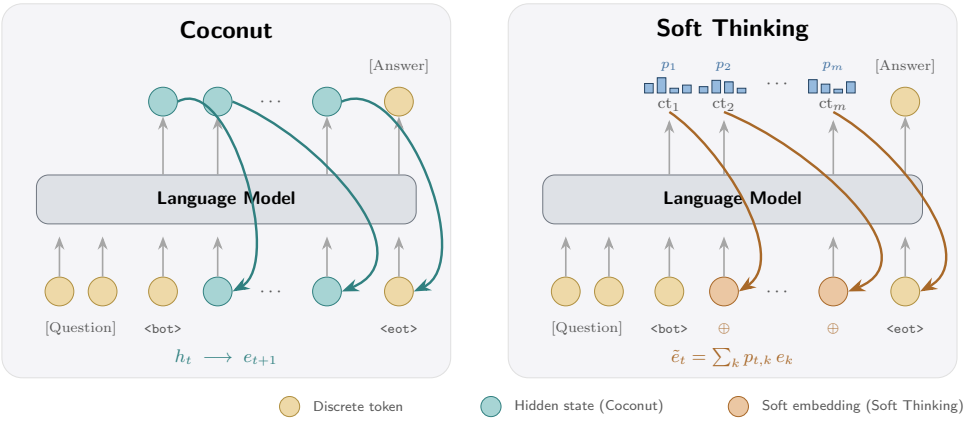


Figure 1: Two latent CoT approaches. **Left:** Coconut feeds the last hidden state directly back as the next input embedding, forming a recurrent loop in continuous space. **Right:** Soft Thinking computes a probability distribution over the vocabulary and forms the next input as a weighted sum of token embeddings. Both methods replace discrete reasoning tokens with continuous representations, but differ in how these representations are constructed.

2024; Cywiński et al., 2025). To the best of our knowledge, no other works have attempted to understand the inner workings of latent CoT models from an interpretability perspective.

3 Background

Let $\mathcal{M}$ be a transformer language model with L layers and vocabulary $\mathcal{V}$. We denote the *embedding matrix* by $E \in \mathbb{R}^{|\mathcal{V}| \times d}$, mapping discrete tokens to d-dimensional vectors, and the *unembedding matrix* by $U \in \mathbb{R}^{|\mathcal{V}| \times d}$, projecting hidden states back to the vocabulary space.For a token $k \in \mathcal{V}$, we write $e_k = E[k]$ for its embedding. Given an input sequence $\mathbf{x} = (x_1, \ldots, x_n)$, at each position i, $\mathcal{M}$ computes a hidden representation $h_i^{(\ell)} \in \mathbb{R}^d$ at each layer $\ell \in \{1, \ldots, L\}$. We start by defining the notion of superposition which is crucial to our analysis:

Definition 1 (Superposition)**.** A model reasons in *superposition* at position i if its hidden state $h_i^{(\ell)}$ encodes in token space a distribution over multiple candidate continuations.

We will consider two ways of obtaining superposition throughout this paper: *forced* superposition and *learned* superposition. We define both below:

Definition 2 (Forced superposition)**.** Forced superposition is superposition introduced at the input by explicitly constructing embeddings as convex combinations of token embeddings: $\tilde{e} = \sum_k \alpha_k e_k$ with $\alpha \in \Delta^{|\mathcal{V}|-1}$.

Definition 3 (Learned superposition)**.** Learned superposition is superposition that emerges from training a model to use latent thoughts on a task which rewards parallel exploration of reasoning paths.

CoT Generation. We consider a setting where, given an input query, a model generates a CoT followed by a final answer. A sequence consists of *input tokens* $\mathbf{x} = (x_1, \ldots, x_n)$, *reasoning tokens* $\mathbf{r} = (r_1, \ldots, r_T)$, and *answer tokens* $\mathbf{y} = (y_1, \ldots, y_m)$. Generation proceeds autoregressively: at step t, the model computes $p(r_t \mid \mathbf{x}, r_1, \ldots, r_{t-1})$ and selects a token according to some decoding strategy (we focus on greedy decoding). In *discrete* CoT, each $r_t \in \mathcal{V}$ is a vocabulary token with embedding e_{r_t} fed as input to the next step. The key difference with *latent* CoT is which point in embedding space is used: discrete CoT is constrained to the vocabulary manifold $\{e_k : k \in \mathcal{V}\}$, while latent CoT can use any point in embedding space.

Soft Thinking. Soft Thinking (Zhang et al., 2025) is a form of *forced* superposition which uses information from the logit distribution to craft the superposition. At each step t, instead of selecting a discrete token, the method computes a distribution over the vocabulary simplex $p_t \in \Delta^{|\mathcal{V}|-1}$ and forms the embedding

$$\tilde{e}_t = \sum_{k \in \mathcal{V}} p_{t,k}\, e_k, \tag{1}$$

which lies in the convex hull of vocabulary embeddings. According to Zhang et al. (2025), this method "naturally preserves a 'superposition' which retains the entire information in each step".

Coconut. Coconut (Hao et al., 2025) takes a different approach: instead of constructing soft embeddings from vocabulary distributions, it feeds the model's own last hidden representation back as the next input embedding, enabling recurrent "reasoning in continuous latent space." The model is trained via a staged curriculum, progressively replacing discrete CoT tokens with continuous latent thoughts. On ProsQA, a synthetic graph-traversal QA task, the authors report that latent tokens encode a breadth-first search (BFS) over the graph, citing Logit Lens probing that reveals intermediate entities at latent positions. In Section 5, we revisit these claims and conduct an interpretability analysis on trained Coconut models.

Logit Lens. To understand how Soft Thinking tokens are processed, we employ Logit Lens (Nostalgebraist, 2020), a technique for interpreting intermediate LLM computations. Normally, only the final layer representation $h_i^{(L)}$ is projected to vocabulary space via $p_i^{(L)} = \mathrm{softmax}(U\, h_i^{(L)})$. Logit Lens applies this projection to intermediate representations, yielding $p_i^{(\ell)} = \mathrm{softmax}(U\, h_i^{(\ell)})$ at any layer ℓ, revealing how predictions evolve across layers. For Soft Thinking, since soft tokens are linear combinations of embeddings, using Logit Lens is well motivated. For Coconut, we use GPT-2, whose tied embedding/unembedding matrices justify applying logit lens as a cosine-similarity measure between latent thoughts and vocabulary tokens.

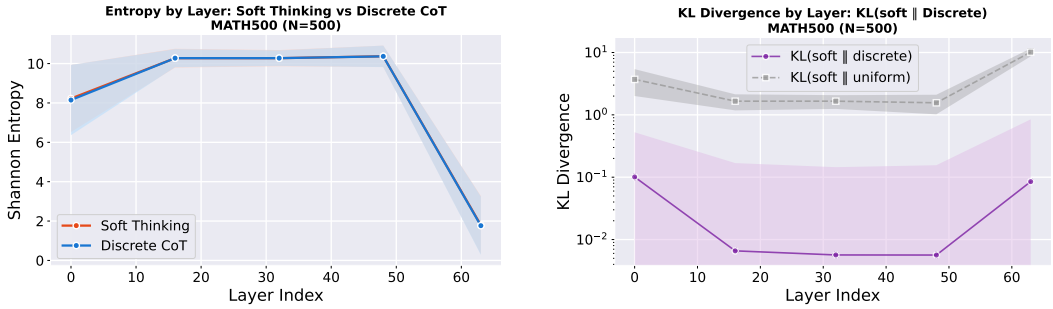
4 Do Off-the-Shelf Models Reason in Superposition?

If LLMs are indeed capable of leveraging forced superposition, their internal representations when processing Soft Thinking tokens should differ meaningfully from those for discrete tokens, maintaining uncertainty and showing higher entropy at intermediate layers. We test this hypothesis with two experiments: (1) a *side-by-side comparison* examining entropy profiles across layers when using latent CoT vs. discrete CoT, and (2) an *embedding-level intervention* measuring how changing a single token from Soft Thinking to discrete affects representations.

Experimental setup. We use QwQ-32B (Qwen Team, 2025) (reasoning model) and Qwen2-1.5B (Yang et al., 2024a) (base; results in Appendix B). We perform our analysis on MATH500 (Lightman et al., 2023), AIME2024 (AMC, 2025) and a 500-example subset of the test set from GSM8K (Cobbe et al., 2021). We apply Logit Lens at five evenly spaced layers (see Appendix B.3) and focus on presenting QwQ-32B results on MATH500 in the main text. Results on Qwen2-1.5B and additional results concerning AIME2024 and GSM8K are reported in Appendix C.

4.1 Comparing entropy profiles in latent vs. discrete CoT

We start by comparing the internal computations of a model using Soft Thinking to a discrete CoT baseline via Logit Lens. We prompt both models with each problem, have them generate their respective CoTs, and at every 50 decoding steps apply Logit Lens to compute the Shannon entropy of the distribution over $\mathcal{V}$ at selected layers. If entropy profiles differ significantly, this would be evidence that the soft tokens meaningfully alter the internal computations.



(a) Entropy comparison (MATH500, N=500)

(b) KL divergence (MATH500, N=500)

Figure 2: **Superposition collapses early in the forward pass of QwQ-32B. (a)** Shannon entropy shows identical patterns for Soft Thinking (orange) and discrete CoT (blue), both converging to near-zero entropy at the same rate. **(b)** KL divergence drops to $\sim 10^{-4}$ in middle layers, showing that Soft Thinking tokens become functionally identical to discrete tokens within the first few layers. The uncertainty in Soft Thinking token embeddings does not propagate through the network. The gray line represents the KL with a uniform distribution over the vocabulary as baseline.

Figure 2a shows entropy averaged over all CoT steps and problems. The entropy across layers is nearly *identical* for both approaches, with the same pattern: high entropy in early-to-middle layers collapsing to near-zero at the final layer. This is inconsistent with superposition. If the model truly maintained multiple solutions in parallel, Soft Thinking should exhibit higher entropy throughout; the indistinguishable profiles instead suggest that Soft Thinking tokens are processed like discrete CoT tokens, collapsing to a single interpretation early in the forward pass.

However, this compares independently generated CoTs that may differ in content. To control for this, we next investigate changing only a *single* token from soft to discrete.

4.2 Intervening with discrete tokens during Soft Thinking

We perform an intervention experiment to test more directly whether internal representations differ when processing soft vs. discrete tokens. At every 50 steps of a Soft Thinking generation, we run two independent forward passes: one using the usual Soft Thinking token $\tilde{e}_t$, and one replacing it with the discrete argmax embedding $e_{\text{argmax}\, p_t}$. For both, we apply Logit Lens at selected layers and compute KL divergence and cosine similarity between the resulting hidden representations. Note that only the intervened token changes; previous tokens in the KV cache remain Soft Thinking tokens.

Table 1: Summary of Soft Thinking token intervention metrics. Cosine similarity is averaged across all layers, steps, and problems. Mixing entropy is the Shannon entropy of the convex combination weights p_t used to form each soft token.

Metric	Qwen2-1.5B	QwQ-32B
Cosine similarity	0.998 ± 0.013	0.996 ± 0.025
Mixing weight entropy (nats)	0.10 ± 0.26	0.18 ± 0.34

As can be seen in Figure 2b, the KL divergence remains small (relative to the baseline) across thinking steps and layers, achieving at most values of 10^{-1}. The KL is largest at the first and last layers. We hypothesize that this is due to embedding differences in the first layers and to minor logit differences in the final layer. Table 1 corroborates this: the average cosine similarity between argmax and soft tokens is consistently high. Finally, Figure 6 (in Appendix) shows that top predicted tokens are typically very similar. Moreover, token predictions with high entropy do not seem to encode "hesitation" between key entities, but

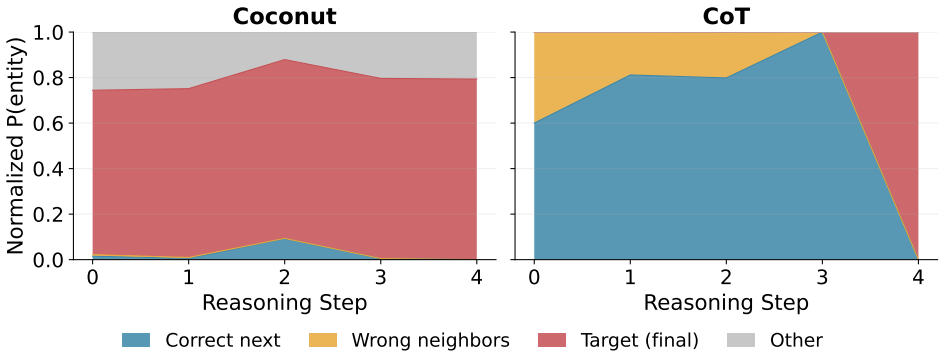


Figure 3: Step-aware entity belief for fine-tuned GPT-2 on 5-step ProsQA examples (normalized among all entities). **Coconut (left)**: target entity dominates from step 0 onward with no progression. **CoT (right)**: correct-next entity dominates early steps, target takes over at the final step, the expected pattern for genuine multi-hop reasoning.

rather show the model hesitating between prepositions or punctuation. Taken together, these results suggest that even at the token level, Soft Thinking methods produce computations that do not significantly differ from the discrete baseline.

5 Do Trained Models Reason in Superposition?

The previous section showed that off-the-shelf LLMs do not leverage superposition when given superposed inputs. A natural follow-up question is whether models *trained* for latent reasoning behave differently. In order to answer this question, we perform experiments using Coconut, widely regarded as one of the canonical frameworks for latent CoT training. We test both fine-tuned and from-scratch Coconut variants (Hao et al., 2025).

5.1 Fine-Tuned Models

We start by studying how fine-tuning pretrained language models to use latent thinking impacts their ability to reason in superposition. Here, we investigate if the model is able to *learn* to encode such a superposition into its latent thoughts.

Experimental setup. Following the methodology of Hao et al. (2025), we evaluate GPT-2 (124M) (Radford et al., 2019) as well as three models from the SmolLM2 suite (allal et al., 2025) (135M, 360M and 1.7B) on ProsQA, a synthetic graph-traversal QA task requiring multi-hop logical inference over defined relationships (e.g., *"Every dax is a wug. Every dax is a zug. Every wug is a blicket. Rex is a dax. Is Rex a blicket or a gorple? Blicket."*). The CoT baseline is trained with standard CoT supervised fine-tuning. The Coconut model is trained with the staged curriculum of Hao et al. (2025), which progressively replaces CoT steps with continuous latent tokens.

Latent tokens are unnecessary for performance. We perform a causal intervention to measure the effect of the latent tokens on the model's predictions. Since ProsQA poses a binary query (of the form *"Is Rex a blicket or a gorple?"*), removing the latent thoughts amounts to a *do* operation on the input; we then counterfactually evaluate the log probabilities the model assigns to both outcomes, thereby measuring how it would have scored each completion without any latent computation. As can be seen in Table 2 (left), the maximum observed drop is **1.0%** across all considered models; this suggests that the latent thinking makes a negligible contribution to the performance of the Coconut model. We hypothesize that the small increase in performance given by latent thinking is due to a positive feedback loop phenomenon. For cases where $P(\text{target})$ is lower at step 0, it is possible that the "latent

Table 2: ProsQA accuracy for fine-tuned and from-scratch models. (a) reports the no-latent evaluation across models. (b) reports it across model depth.

(a) Fine-tuned, counterfactual no-latent evaluation.

Model	CoT	Coconut	No lat.	Drop
GPT-2 (124M)	85.3	99.0	99.0	−0.0
SmolLM2-135M	72.7	93.3	92.3	−1.0
SmolLM2-360M	85.0	98.7	98.0	−0.7
SmolLM2-1.7B	98.3	100.0	100.0	−0.0

(b) Trained from scratch, 8 heads, 768-dim.

Layers	w/ / w/o latent
2	94.5 / 13.8
4	96.2 / 16.0
8	95.2 / 53.0
12	91.2 / 34.1

thinking" procedure increases the probability. Next, we use probing to investigate the structure of the latent thoughts to better understand what motivates this behavior.

Probing reveals no step-by-step reasoning. We probe by projecting hidden states at each reasoning position through the LM head. To better understand how belief evolves, we track how the normalized entity distribution evolves across reasoning steps. At each step, graph entities are categorized into four groups: *correct next* (the right answer for the given step), *wrong neighbors* (nodes that are adjacent but do not lead to the correct path), *target* (the final answer), and *other* (entities that are part of the graph but neither the correct answer nor reachable from the current node). For instance in the example above, the *correct next* would be "wug", the *wrong neighbor* would be "zug", and the target entity would be "blicket". For Coconut, Figure 3 (left) shows the target entity dominating the distribution throughout the entire reasoning process. For CoT, (right) the correct-next entity dominates early steps and the target takes over only at the final step: the expected signature of step-by-step multi-hop reasoning. This behavior suggests that the model learned a *shortcut solution*: it first obtains the correct answer in a single forward pass, then copies it over through the latent tokens.

Notably, this synthetic task was explicitly designed by Hao et al. (2025) to require parallel exploration of multiple reasoning paths, making the failure to learn superposition particularly striking. Despite this favorable setting, the model converges to a shortcut solution, suggesting that superposition is not a naturally preferred strategy under standard fine-tuning. This casts doubt on whether scaling model size or applying similar training procedures on more complex benchmarks would, by themselves, induce superposition-based reasoning. We observe similar behavior on ProntoQA (Saparov & He, 2023); see Appendix E.1 for details.

5.2 From-Scratch Models

Next, we investigate the use of superposition in models trained from scratch. We train small GPT-2 style transformers on a simplified variant of ProsQA with a symbolic tokenizer (40 tokens) (Zhu et al., 2025). For these models, we ablate over layer count (2, 4, 8 and 12 layers) but leave the number of heads fixed at 8 and the embedding dimension fixed to 768. We compare two training regimes: i) from scratch with latent thinking (Coconut); ii) from scratch using cross entropy on the gold CoT *then* training with Coconut (CoT + Coconut). Note that the variant of Coconut used here is different from that of Hao et al. (2025): here, at every step in the problem models are trained to use i latent thoughts to predict the value of the ith node. Crucially, this methodology never trains the model on the entire gold CoT. As in Section 5.1, we employ probing to understand how belief evolves through the model's latent CoT.

Superposition occurs in from-scratch models. Figure 4 shows belief evolution across thinking steps for from-scratch Coconut variants. For both the Coconut and CoT+Coconut training methods, the models show evidence of leveraging superposition. The state is dominated by the correct next entity probability but still leaves significant probability to the other potential neighbors. This remains true even when the model is first trained to perform the task with a discrete CoT, suggesting that next-token pretraining is not the sole factor leading to superposition collapse.

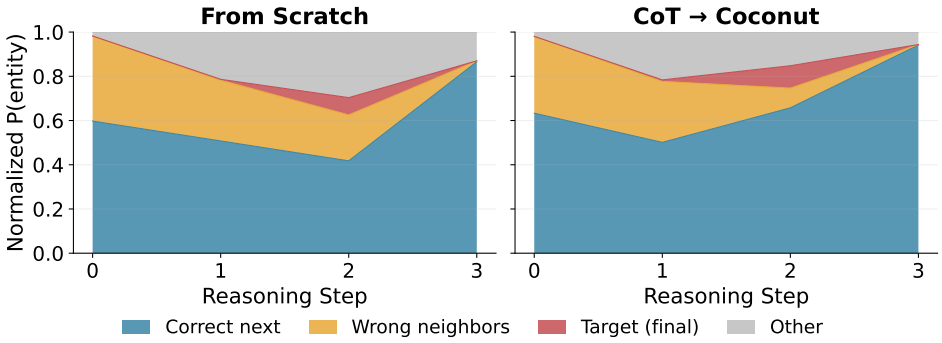


Figure 4: Step-aware entity belief for from-scratch 2-layer models on 4-step ProsQA examples. **From Scratch (left)**: correct-next and wrong-neighbor entities dominate intermediate steps, with belief evolving across the reasoning trace. **CoT → Coconut (right)**: similar pattern with slightly more concentrated belief on correct-next entities. We show 4-step examples here and 5-step examples for the fine-tuned model in Figure 3; the qualitative pattern is consistent across hop counts (see Appendix E.1).

Latent tokens are necessary for performance. As can be seen in Table 2 (right), models trained from scratch do indeed leverage their latent thoughts contrary to the fine-tuned case. Removing access to latent steps produces significant performance drops across all depths (for instance, from 94.5% to 13.8% for the 2-layer model), thus corroborating the entity belief results. This remains true even when training larger and larger models, all the way up to the size and depth of the pretrained GPT-2 model (12 layers, 768 embedding dim and 8 heads), which is surprising given the theoretical construction proposed by Zhu et al. (2025) needed only 2 layers. This suggests that shortcut learning in the fine-tuned case is not solely explained by the capacity of the model relative to the task; there may be biases from the pretraining or fine-tuning procedures which incentivize shortcut learning.

6 Exploring the Limitations of Latent Thinking

The previous two sections provide evidence that superposition only occurs in a very limited set of scenarios: only from-scratch models manage to leverage superposition in their reasoning process. In this section, we further investigate this discrepancy through two distinct lenses. First, we propose an explanation as to why superposition collapses in the training-free regime: next-token pretraining biases models to commit to a single token in its final layers, projecting superposed inputs onto a near-discrete representation. Second, we characterize *when* learned superposition emerges in from-scratch models. We show empirically that width has a far greater effect than depth on performance and models' propensity to leverage latents.

Models trained on next-token prediction commit to a token in the last layers. Figure 5 (left) shows, for Pythia-1B, the entropy across layers of a forced superposition where all tokens in the combination have uniform weight. Uniform weights are chosen to avoid the confound of soft tokens computed from peaky logit distributions. Moreover, we compare the pretrained model to a model with weights reinitialized at random to isolate the confound of the learning process and the architecture. Across different numbers of tokens in the mixture, the trend is clear: entropy drops rapidly when reaching the final layers. This contrasts sharply with the random weights baseline: here, the entropy remains high throughout. That this phenomenon only occurs in the pretrained model suggests that the pretraining is the driving factor to the final-layer entropy collapse. We also note that fine-tuning does not seem to be enough to fix this issue; the last-layer entropy collapse persists when fine-tuning GPT-2 using the Coconut methodology (see Appendix E.2).

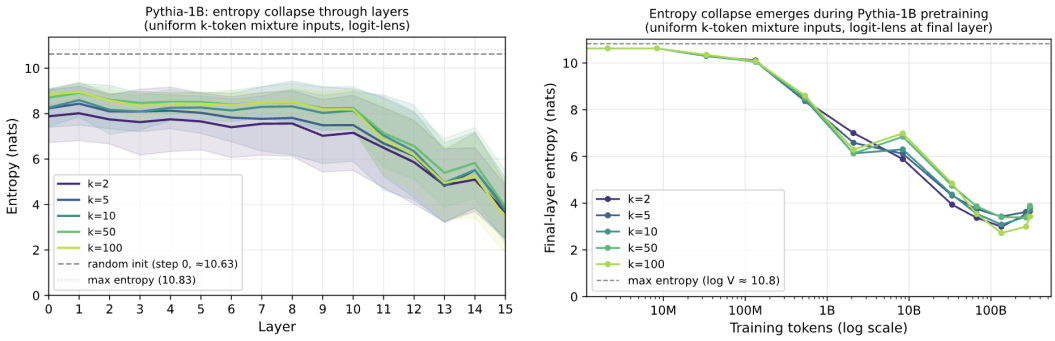


Figure 5: Entropy of synthetic uniform superpositions for Pythia-1B, where k tokens are sampled uniformly at random and combined with equal weights ($1/k$). **Left**: entropy across layers of the fully pretrained model collapses at the final layers regardless of k; the dashed line shows a randomly initialized model, whose entropy stays near the maximum. **Right**: final-layer entropy across pretraining checkpoints, showing that the collapse emerges gradually as the number of training tokens increases. Shaded regions show ± 1 std.

Table 3: Parameter-matched from-scratch Coconut on ProsQA ($\sim$15M parameters). Starting from the 2-layer model, depth is increased while the embedding width is shrunk to keep the parameter count fixed. Accuracies (%) are reported with and without latent tokens as mean $\pm$ std over 3 seeds.

Model	**Params**	**w/ latents**	**w/o latents**
2L-768d	14.97M	96.0 ± 0.9	30.9 ± 8.4
4L-544d	14.78M	91.8 ± 4.9	39.7 ± 20.8
8L-384d	14.61M	81.6 ± 4.3	73.2 ± 1.4
12L-320d	15.14M	72.4 ± 7.1	70.0 ± 6.3

Entropy collapse appears gradually during training. To determine if the final-layer entropy collapse is actually a consequence of the pretraining process, we evaluate the final-layer entropy over uniform superpositions at successive Pythia-1B pretraining checkpoints (Biderman et al., 2023). As shown in Figure 5 (right), for all combinations of soft tokens the entropy starts near its maximum and decreases gradually as the number of training tokens increases. This gradual emergence suggests that the collapse is tied to the pretraining process, and not merely to the output geometry of a fully trained model.

Embedding size matters when training models from scratch. Our previous results indicate that latent reasoners trained from scratch learn to leverage their latent CoT to solve the task, even as depth increases. In this section, we turn our attention to the effect of *width* on latent reasoning. We train GPT-2 style models using the same setup as described in Section 5.2. We start by performing a *parameter-matched* experiment: keeping the overall size of the model constant, we vary the depth to width ratio. The 2-layer model configuration by Zhu et al. (2025) serves as a baseline for parameter count. Table 3 shows that at fixed parameter count, shallower, wider models outperform deeper, narrower ones: the 2- and 4-layer variants exceed 90% accuracy, while the deeper 8- and 12-layer models lag behind by 10 to 20 points.

To confirm this behavior, we perform an additional ablation. First, we find the critical width at which performance degrades in 2-layer and 4-layer models (see Table 11). Then, for a set of embedding sizes below the critical width, we increase gradually the depth of the model, going all the way up to 12 layers. Table 4 shows that increasing capacity along the depth axis yields poor performance; the best model obtains 68.3% accuracy whereas wider models easily saturate the task, obtaining accuracy above 90%.

Table 4: Test accuracy (%) with and without latent tokens across depths, for sub-threshold embedding sizes ($d_{\text{embd}} \leq 64$).

	2L		4L		8L		12L	
d_{embd}	w/	w/o	w/	w/o	w/	w/o	w/	w/o
8	40.8	40.6	35.3	35.1	24.6	24.8	34.8	33.9
16	57.5	54.9	55.6	55.4	55.4	55.4	58.9	58.7
32	58.9	57.8	53.9	53.2	56.8	58.2	61.6	60.1
64	64.9	62.3	63.2	62.5	67.3	64.4	68.3	65.4

7 Discussion and Conclusion

In conclusion, our experiments provide evidence as to when reasoning in superposition occurs and when it does not across three complementary settings. For Soft Thinking, off-the-shelf LLMs process superposed inputs nearly identically to discrete tokens: entropy profiles match, KL divergences approach zero, and cosine similarities exceed 0.99. For fine-tuned Coconut, counterfactually removing the latent tokens from a fine-tuned model changes its predictions by at most 1.0% across all considered model families, and entity-level probing reveals no step-by-step reasoning during latent computation. Finally, superposition only appears in from-scratch Coconut models: removing latents sharply hurts performance, and increasing depth does not induce shortcut learning, even at model sizes matching the pretrained GPT-2 used for fine-tuning.

Why does superposition collapse in pretrained models? As we argue above, this is not merely an inductive bias but a consequence of the pretraining objective: autoregressive training optimizes for discrete next-token prediction, creating representations that separate token identities. When presented with a superposed input, the model projects it onto the nearest discrete interpretation, precisely what the training objective rewards. Moreover, the Soft Thinking distribution p_t is itself very peaky across steps (see Figure 6 and the entropy heatmaps in Appendix B), meaning the input is already near-discrete and there is little superposition to exploit in the first place.

Is token-level superposition desirable? Beyond finding that models do not reason in superposition, we question whether *token-level* superposition is a desirable property in the first place. Many Soft Thinking tokens combine semantically unrelated items (see Figure 6): formatting characters, punctuation, or tokens with similar logits but unrelated meanings. A superposition of "(" and "{" represents syntactic uncertainty, not exploration of alternative reasoning paths. Meaningful parallel exploration likely requires superposition at a higher level of abstraction (over entire reasoning strategies, not individual tokens), which we consider an interesting avenue for future work.

Latent reasoning as flexibility. Despite our negative findings for current methods, latent reasoning remains a promising direction. The advantage of continuous embeddings may not be superposition but *flexibility*: the ability to express intermediate computations that do not correspond to natural language tokens, avoiding the discretization bottleneck. Future work should investigate whether latent reasoning provides benefits through other mechanisms, such as smoother optimization landscapes or more expressive intermediate representations.

Limitations. It would be valuable to run similar experiments on other latent reasoning approaches, such as the RL-trained continuous CoTs of Butt et al. (2026) or the recurrent layer frameworks proposed by Giannou et al. (2023); McLeish et al. (2025). Moreover, our results only pertain to external latent CoTs. It is possible that models may have learned to internalize the latent thinking steps (Deng et al., 2024); our results do not investigate this. Analyzing model internals through probing or circuit discovery to understand their learned reasoning schemes could be an exciting direction for future work. Finally, we hope in future work to apply the findings from this paper to design novel latent thinking methods.

Acknowledgments

M. Rizvi-Martel's research is supported by NSERC (CGS-D Scholarship), and G. Rabusseau's by NSERC and the CIFAR AI Chair program. We also acknowledge NVIDIA for providing computational resources.

References

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. In *International Conference on Learning Representations*, 2017. URL https://sls.csail.mit.edu/publications/2017/ICLR17_Belinkov.pdf.

Loubna Ben allal, Anton Lozhkov, Elie Bakouch, Gabriel Martin Blazquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Agustín Piqueres Lajarín, Hynek Kydlíček, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan Son NGUYEN, Ben Burtenshaw, Clémentine Fourrier, Haojun Zhao, Hugo Larcher, Mathieu Morlon, Cyril Zakka, Colin Raffel, Leandro Von Werra, and Thomas Wolf. SmolLM2: When smol goes big — data-centric training of a fully open small language model. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=3JiCl2A14H.

AMC. American invitational mathematics examination. https://artofproblemsolving.com/wiki/index.php/American_Invitational_Mathematics_Examination, 2025.

Yonatan Belinkov. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219, March 2022. doi: 10.1162/coli_a_00422. URL https://aclanthology.org/2022.cl-1.7/.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar Van Der Wal. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pp. 2397–2430. PMLR, 2023. URL https://proceedings.mlr.press/v202/biderman23a/biderman23a.pdf.

Natasha Butt, Ariel Kwiatkowski, Ismail Labiad, Julia Kempe, and Yann Ollivier. Soft tokens, hard truths. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=9JjKTp8Jmy.

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

Bartosz Cywiński, Emil Ryd, Senthooran Rajamanoharan, and Neel Nanda. Towards eliciting latent knowledge from llms with mechanistic interpretability. *arXiv preprint arXiv:2505.14352*, 2025. URL https://arxiv.org/abs/2505.14352.

Jingcheng Deng, Liang Pang, Zihao Wei, Shichen Xu, Zenghao Duan, Kun Xu, Yang Song, Huawei Shen, and Xueqi Cheng. Latent reasoning in llms as a vocabulary-space superposition. *arXiv preprint arXiv:2510.15522*, 2025. URL https://arxiv.org/abs/2510.15522v2.

Yuntian Deng, Yejin Choi, and Stuart Shieber. From explicit cot to implicit cot: Learning to internalize cot step by step. *arXiv preprint arXiv:2405.14838*, 2024. URL https://arxiv.org/abs/2405.14838.

Subhabrata Dutta, Joykirat Singh, Soumen Chakrabarti, and Tanmoy Chakraborty. How to think step-by-step: A mechanistic understanding of chain-of-thought reasoning. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=uHLDkQVtyC.

Kanishk Gandhi, Denise H J Lee, Gabriel Grand, Muxin Liu, Winson Cheng, Archit Sharma, and Noah Goodman. Stream of search (sos): Learning to search in language. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=2cop2jmQVL.

Angeliki Giannou, Shashank Rajput, Jy-Yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 11398–11442. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/giannou23a.html.

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

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. Colorless green recurrent networks dream hierarchically. In Marilyn Walker, Heng Ji, and Amanda Stent (eds.), *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1195–1205, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1108. URL https://aclanthology.org/N18-1108/.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. URL https://arxiv.org/abs/2501.12948.

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

David Herel and Tomas Mikolov. Thinking tokens for language modeling. *arXiv preprint arXiv:2405.08644*, 2024. URL https://arxiv.org/abs/2405.08644.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=v8L0pN6EOi.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. Assessing the ability of lstms to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535, 2016. URL https://aclanthology.org/Q16-1037/.

Sean Michael McLeish, Ang Li, John Kirchenbauer, Dayal Singh Kalra, Brian R. Bartoldson, Bhavya Kailkhura, Avi Schwarzschild, Jonas Geiping, Micah Goldblum, and Tom Goldstein. Teaching pretrained language models to think deeper with retrofitted recurrence. In *NeurIPS 2025 Workshop on Efficient Reasoning*, 2025. URL https://openreview.net/forum?id=Oq3Xblt0x1.

Nostalgebraist. interpreting gpt: the logit lens, 2020. URL https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens.

OpenAI. OpenAI o3 and o4-mini System Card. Technical report, OpenAI, San Francisco, CA, April 2025. URL https://cdn.openai.com/pdf/2221c875-02dc-4789-800b-e7758f3722c1/o3-and-o4-mini-system-card.pdf. PDF available online.

Jacob Pfau, William Merrill, and Samuel R. Bowman. Let's think dot by dot: Hidden computation in transformer language models. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=NikbrdtYvG.

Qwen Team. QwQ-32B: Embracing the power of reinforcement learning. https://qwenlm.github.io/blog/qwq-32b/, 2025.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8): 9, 2019. URL https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

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

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

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

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024a. URL https://arxiv.org/abs/2407.10671.

Liu Yang, Kangwook Lee, Robert D Nowak, and Dimitris Papailiopoulos. Looped transformers are better at learning learning algorithms. In *The Twelfth International Conference on Learning Representations*, 2024b. URL https://openreview.net/forum?id=HHbRxoDTxE.

Sohee Yang, Elena Gribovskaya, Nora Kassner, Mor Geva, and Sebastian Riedel. Do large language models latently perform multi-hop reasoning? In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10210–10229, Bangkok, Thailand, August 2024c. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.550. URL https://aclanthology.org/2024.acl-long.550/.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik R Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=5Xc1ecxO1h.

Zhen Zhang, Xuehai He, Weixiang Yan, Ao Shen, Chenyang Zhao, and Xin Eric Wang. Soft thinking: Unlocking the reasoning potential of LLMs in continuous concept space. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=ByQdHPGKgU.

Hanlin Zhu, Shibo Hao, Zhiting Hu, Jiantao Jiao, Stuart J Russell, and Yuandong Tian. Reasoning by superposition: A theoretical perspective on chain of continuous thought. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, pp. 79931–79963. Curran Associates, Inc., 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/72c363c2a573ca2128bd176d3317696b-Paper-Conference.pdf.

Appendix Contents

A Disclosure of LLM usage **14**

B Soft Thinking: Experimental Details **14**

B.1 Models 14

B.2 Soft Thinking Configuration 15

B.3 Logit Lens Setup 15

B.4 Evaluation Problems 16

B.5 Compute 16

C Soft Thinking: Additional Results **17**

C.1 Token-level comparisons 17

C.2 Full-Dataset Logit Lens Results 17

C.3 Uniform Superposition: Entropy and KL Divergence 17

C.4 Benchmark Results 23

D Coconut: Experimental Details **24**

D.1 ProsQA Task 24

D.2 Training Setup 24

D.3 Entity Probing: Sample Sizes 26

E Coconut: Additional Results **27**

E.1 Entity Distributions 27

E.2 Coconut Other Results 27

E.3 Coconut experiments on different models 27

E.4 Ablation over embedding width 28

E.5 From-Scratch Coconut: Multi-Seed Results 28

A Disclosure of LLM usage

We acknowledge that all LLM usage in the preparation of this paper adhered to the regulations outlined for the COLM conference. We used `Claude Opus 4.6` only to assist in the implementation, data visualization, and for shortening of text originally written by the authors.

B Soft Thinking: Experimental Details

B.1 Models

We use three models spanning two model families:

- **QwQ-32B**: A 32.5B-parameter reasoning model based on the Qwen2.5 architecture with 64 transformer layers and a hidden dimension of 5120. We use this as our primary model since it was trained for chain-of-thought reasoning.
- **Qwen2-1.5B**: A 1.5B-parameter base language model with 28 transformer layers and a hidden dimension of 1536. We use this smaller model to test whether our findings generalize across model scales.
- **DeepSeek-R1-Distill-Llama-70B**: A 70B-parameter reasoning model distilled from DeepSeek-R1 (Guo et al., 2025) into the Llama architecture, with 80 transformer layers and a hidden dimension of 8192. We include this model to verify that our findings extend beyond the Qwen family to a different architecture and scale.

The Qwen models use a shared tokenizer with a vocabulary of 151,643 tokens. DeepSeek-R1-Distill-Llama-70B uses the Llama tokenizer with a vocabulary of 128,256 tokens. All experiments use the models in `bfloat16` precision.

B.2 Soft Thinking Configuration

We use the following decoding hyperparameters for all Logit Lens experiments, consistent across both models:

Table 5: Soft Thinking decoding hyperparameters.

Parameter	Symbol	Value
Temperature	τ	0.6
Top-k (sampling)	k	30
Soft top-k (embedding mix)	k_{soft}	15
Weighting scheme		Softmax
Max new tokens	$T_{\max}$	2048

At each reasoning step t, the model computes logits over the vocabulary and selects the top-k_{soft} tokens. Their logits are normalized via softmax to obtain the mixing weights $p_t \in \Delta^{k_{\text{soft}}-1}$, and the Soft Thinking embedding is formed as $\tilde{e}_t = \sum_{i=1}^{k_{\text{soft}}} p_{t,i}\, e_{v_i}$ where $v_1, \ldots, v_{k_{\text{soft}}}$ are the top-k_{soft} tokens by logit value.

For the benchmark evaluation runs (Tables 7 and 8), we additionally vary the cold stop threshold $c \in \{0.0, 0.01, 0.1, 0.2\}$, which terminates Soft Thinking when the top token probability exceeds $1 - c$.

B.3 Logit Lens Setup

We apply Logit Lens at 5 evenly spaced probe layers: $\{0, \lfloor L/4 \rfloor, \lfloor L/2 \rfloor, \lfloor 3L/4 \rfloor, L-1\}$, where L is the number of transformer layers. This corresponds to layers $\{0, 7, 14, 21, 27\}$ for Qwen2-1.5B, $\{0, 16, 32, 48, 63\}$ for QwQ-32B, and $\{0, 20, 40, 60, 79\}$ for DeepSeek-R1-Distill-Llama-70B.

Entropy profile comparison (Section 4.1). For each problem, we run two independent generations: one using Soft Thinking and one using standard discrete decoding (greedy argmax). Every 50 decoding steps, we apply Logit Lens at each probe layer and record the Shannon entropy of the resulting distribution over V. We also store the top-10 predicted tokens at each checkpoint for qualitative comparison.

Token-level intervention (Section 4.2). During Soft Thinking generation, we intervene every 50 steps by performing two fresh forward passes over the full sequence: one using the Soft Thinking embedding and one replacing it with the argmax embedding. Crucially, only the current token's embedding differs; the KV cache from previous (soft) tokens is shared. At each probe layer, we compute:

- KL divergence: $D_{\mathrm{KL}}(p^{(\ell)}_{\mathrm{soft}} \| p^{(\ell)}_{\mathrm{argmax}})$ between the Logit Lens distributions.
- Cosine similarity: $\cos(h^{(\ell)}_{\mathrm{soft}}, h^{(\ell)}_{\mathrm{argmax}})$ between hidden representations.
- Entropy difference: $H(p^{(\ell)}_{\mathrm{soft}}) - H(p^{(\ell)}_{\mathrm{argmax}})$.
- Top-k token overlap: $|S^{(\ell)}_{\mathrm{soft}} \cap S^{(\ell)}_{\mathrm{argmax}}|/k$ for $k = 10$.

B.4 Evaluation Problems

We select 5 problems of varying difficulty from three standard math reasoning benchmarks. The same problems are used across all experiments and both models to enable direct comparison.

GSM8K, Problem 42 (Easy)

Grandma Jones baked 5 apple pies for the fireman's luncheon. She cut each pie into 8 pieces and set the five pies out on the buffet table for the guests to serve themselves. At the end of the evening, after the guests had taken and eaten their pieces of pie, there were 14 pieces of pie remaining. How many pieces were taken by the guests?

Answer: 26

MATH500, Problem 10 (Medium)

What is the least positive integer multiple of 30 that can be written with only the digits 0 and 2?

Answer: 2220

MATH500, Problem 55 (Medium)

Suppose that f is a polynomial such that $(x-1) \cdot f(x) = 3x^4 + x^3 - 25x^2 + 38x - 17$. What is the degree of f?

Answer: 3

AIME 2024, Problem 3 (Hard)

Let x, y and z be positive real numbers that satisfy the following system of equations:

$$\log_2\left(\frac{x}{yz}\right) = \frac{1}{2}, \quad \log_2\left(\frac{y}{xz}\right) = \frac{1}{3}, \quad \log_2\left(\frac{z}{xy}\right) = \frac{1}{4}.$$

Then the value of $\left|\log_2(x^4y^3z^2)\right|$ is $\frac{m}{n}$ where m and n are relatively prime positive integers. Find $m+n$.

Answer: 33

AIME 2024, Problem 5 (Hard)

Alice chooses a set A of positive integers. Then Bob lists all finite nonempty sets B of positive integers with the property that the maximum element of B belongs to A. Bob's list has 2024 sets. Find the sum of the elements of A.

Answer: 55

B.5 Compute

All experiments were run on a SLURM cluster using NVIDIA L40S GPUs (48 GB each). Table 6 summarizes the resource allocation for each experiment.

Table 6: Compute resources per experiment.

Experiment	Model	GPUs	RAM	Wall Time
Comparative (Logit Lens)	Qwen2-1.5B	1× L40S	32 GB	< 1 h
Comparative (Logit Lens)	QwQ-32B	4× L40S	128 GB	< 2 h
Convergence	Qwen2-1.5B	1× L40S	32 GB	< 1 h
Convergence	QwQ-32B	4× L40S	128 GB	< 3 h
Benchmark (MATH500)	QwQ-32B	4× L40S	128 GB	varies
Benchmark (AIME2024)	QwQ-32B	4× L40S	128 GB	varies
Coconut (ProsQA)	GPT-2	4× L40S	128 GB	∼ 3 h

For QwQ-32B, we distribute the model across 4 GPUs using HuggingFace Accelerate's `device_map="balanced"` strategy with a per-GPU memory cap of 75% to avoid out-of-memory errors from uneven shard allocation.

C Soft Thinking: Additional Results

In this section, we present additional results for the Soft Thinking experiments. We present token-level comparisons, full dataset Logit Lens visualizations (for models and metrics which are not shown in the main text) and additional benchmark results showing accuracy of models for different hyperparameter choices.

Curated set vs. full datasets. The entropy profiles and KL divergence curves in Section 4 (Figures 2 and 6) are computed on the *curated set of 5 problems* described in Appendix B.4. All figures in Appendix C.2 use the corresponding *full datasets* (MATH500: $N = 500$; AIME 2024: $N = 30$; GSM8K: $N = 500$), with sample sizes labeled in each caption. The Soft Thinking analysis involves no stochastic training, so reproducibility across runs is exact given fixed model weights and the deterministic decoding procedure.

C.1 Token-level comparisons

In this section, we include visualizations of the top-3 tokens for both the Soft Thinking and argmax decoding methods. We visualize the top-3 across 5 chosen problem instances (see Appendix B.4 for more details) for both the step with highest and lowest KL divergence. Results show that superposition is often being performed on tokens which do not have a meaningful relationship to the problem at hand.

C.2 Full-Dataset Logit Lens Results

In this section we present visualizations of cosine similarity, KL divergence and entropy difference not shown in the main paper. This section also includes figures for Qwen2-1.5B and DeepSeek-R1-Distill-Llama-70B which are not included in the main text.

C.3 Uniform Superposition: Entropy and KL Divergence

To test whether the collapse observed with softmax-weighted Soft Thinking in Section 4.1 is an artifact of the probability-weighted embedding concentrating mass near the argmax token, we repeat the entropy profile and KL divergence experiments with *uniform* weighting: each of the top-k tokens receives equal weight $1/k$, giving the maximally diverse input embedding.

For each combination of model (QwQ-32B, Qwen2-1.5B) and $k \in \{3, 10, 15\}$, we show entropy profiles and KL divergence side by side for AIME 2024 (N=30) and MATH500 (N=500) in Figures 22 to 24 (32B) and Figures 25 to 27 (1.5B).

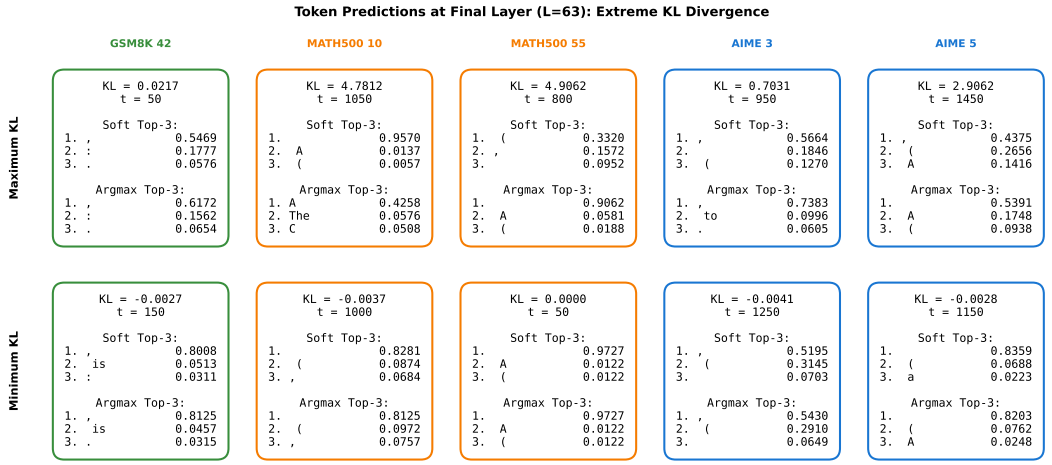


Figure 6: Top 3 tokens at the output layer for time steps with largest (top) and smallest (bottom) KL divergence between soft and argmax representations in **QwQ-32B**. Each column represents a problem instance.

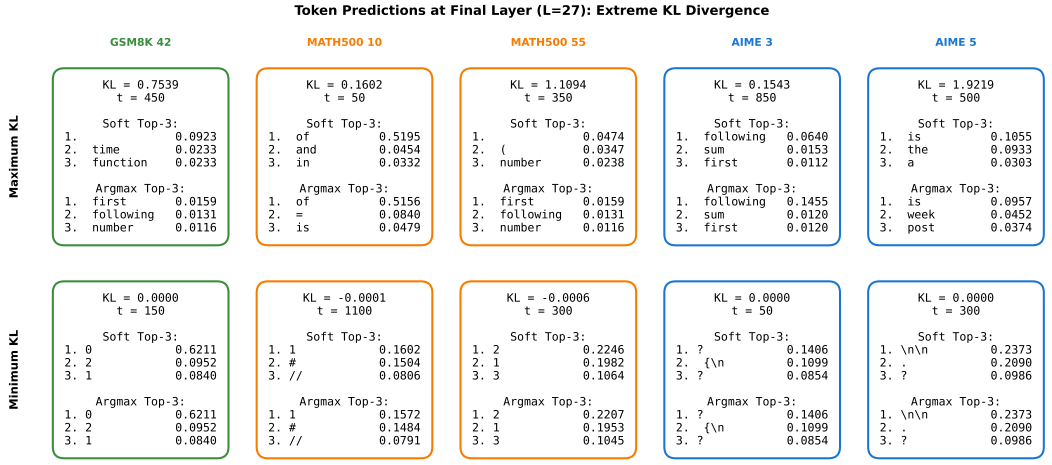


Figure 7: Top 3 predicted tokens at the output layer for time steps with largest (top) and smallest (bottom) KL divergence between soft and argmax representations in **Qwen2-1.5B**. Each column represents a problem instance.

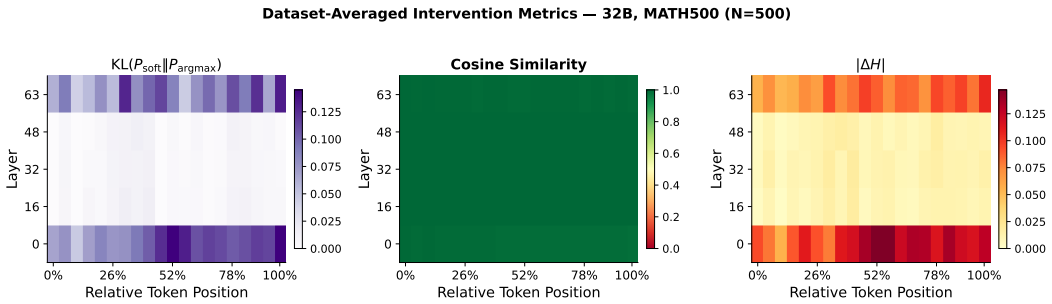


Figure 8: Dataset-averaged intervention metrics for **QwQ-32B on MATH500** (N=500). Left to right: KL divergence, cosine similarity, absolute entropy difference. Each cell is the mean over all problems at that (layer, relative position) bin.

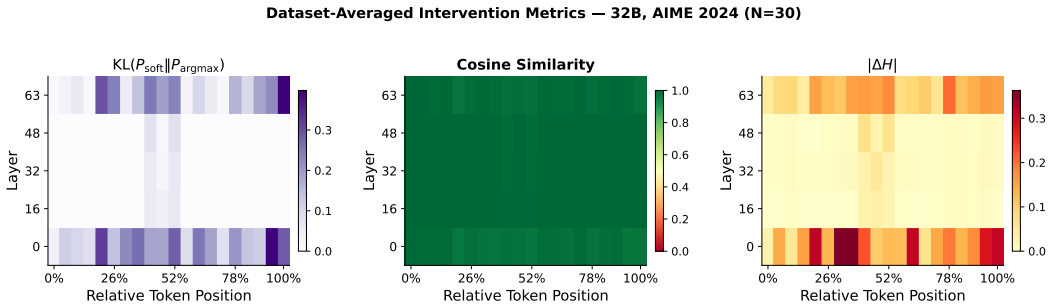


Figure 9: Dataset-averaged intervention metrics for **QwQ-32B on AIME 2024** (N=30).

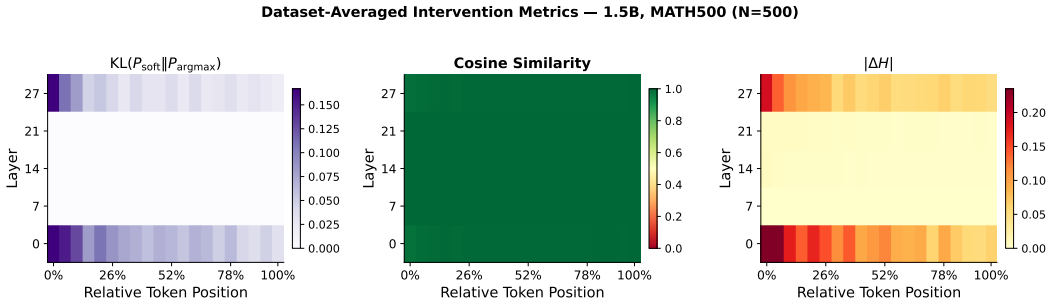


Figure 10: Dataset-averaged intervention metrics for **Qwen2-1.5B on MATH500** (N=500).

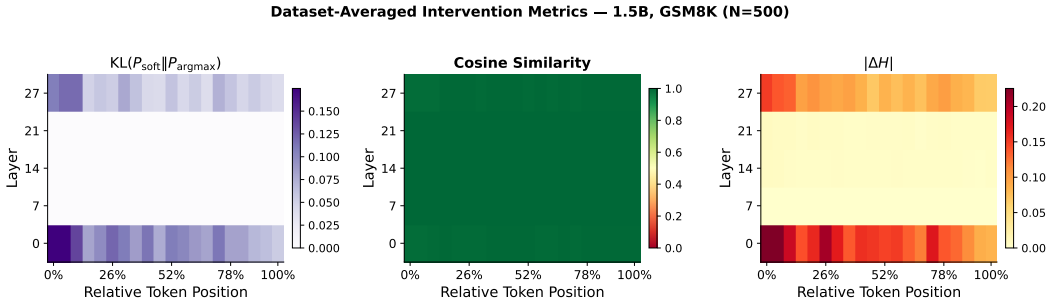


Figure 11: Dataset-averaged intervention metrics for **Qwen2-1.5B on GSM8K** (N=500).

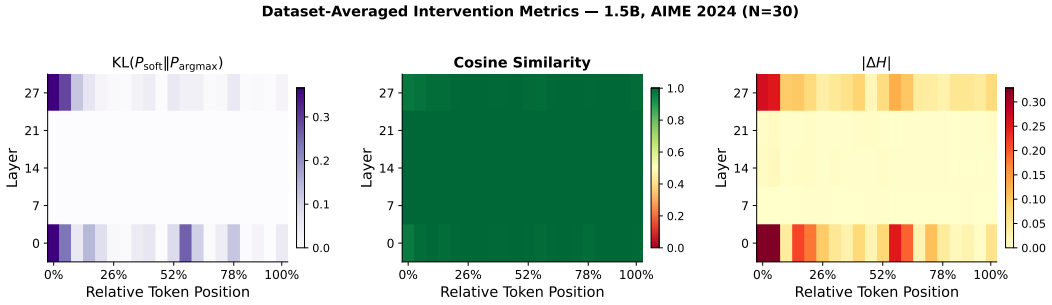


Figure 12: Dataset-averaged intervention metrics for **Qwen2-1.5B on AIME 2024** (N=30).

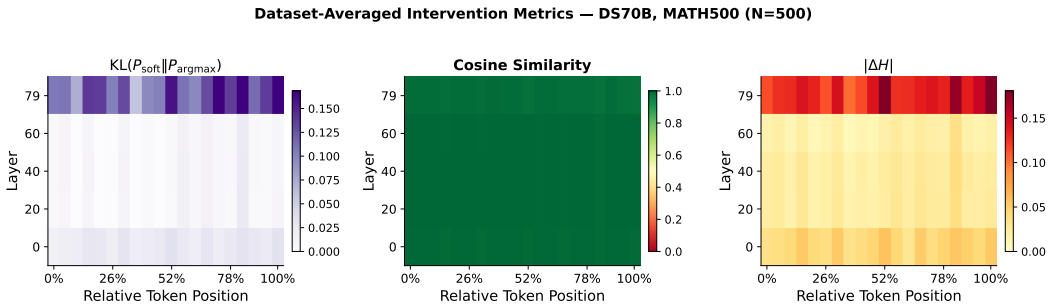


Figure 13: Dataset-averaged intervention metrics for **DeepSeek-R1-Distill-Llama-70B on MATH500** (N=500).

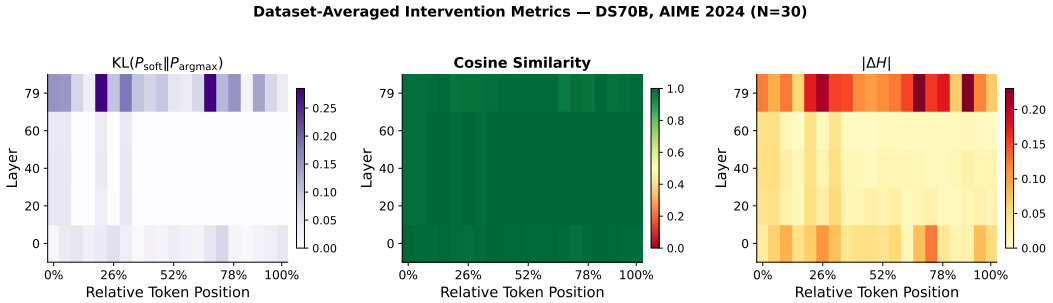
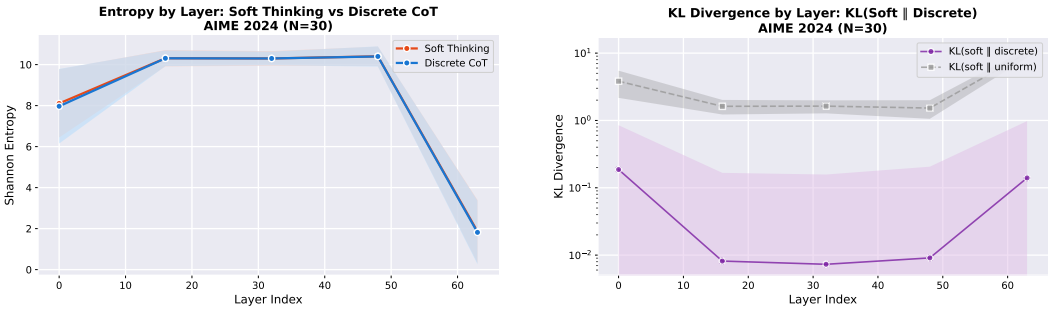


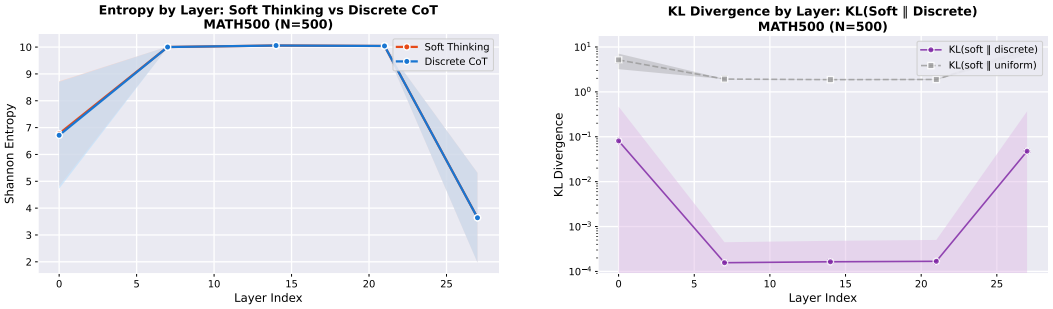
Figure 14: Dataset-averaged intervention metrics for **DeepSeek-R1-Distill-Llama-70B on AIME 2024** (N=30).



(a) Entropy comparison

(b) KL divergence by layer

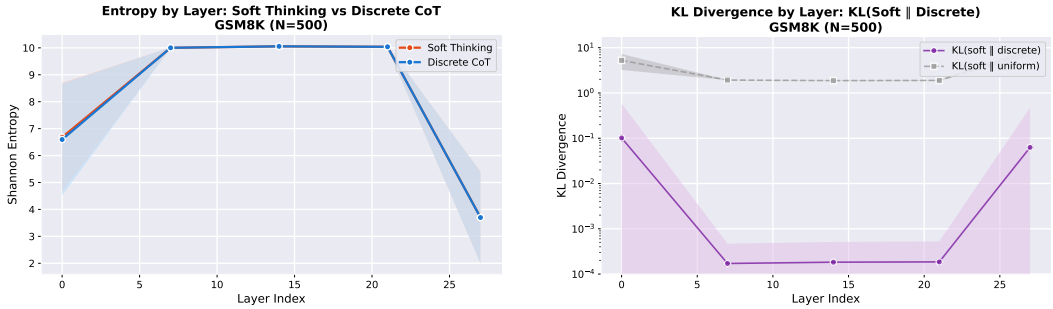
Figure 15: **QwQ-32B on AIME2024 (N=30).** Entropy and KL divergence by layer.



(a) Entropy comparison

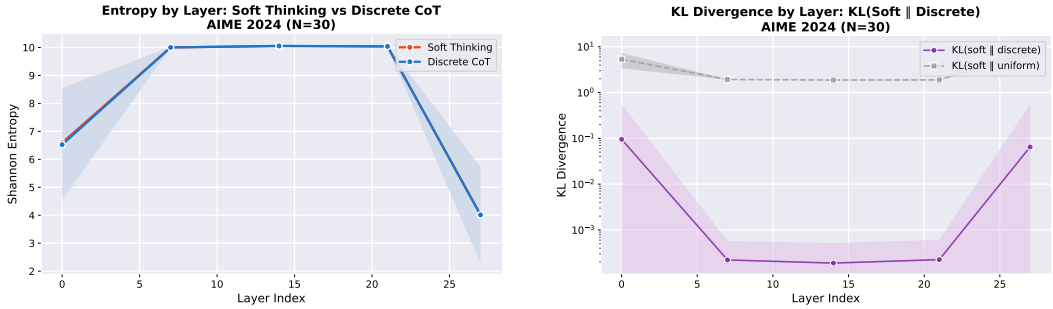
(b) KL divergence by layer

Figure 16: **Qwen2-1.5B on MATH500 (N=500).** Entropy and KL divergence by layer.



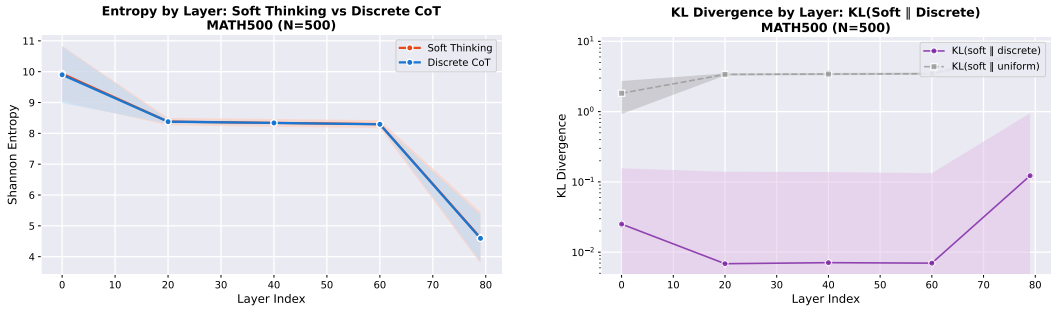
(a) Entropy comparison (b) KL divergence by layer

Figure 17: **Qwen2-1.5B on GSM8K (N=500).** Entropy and KL divergence by layer.



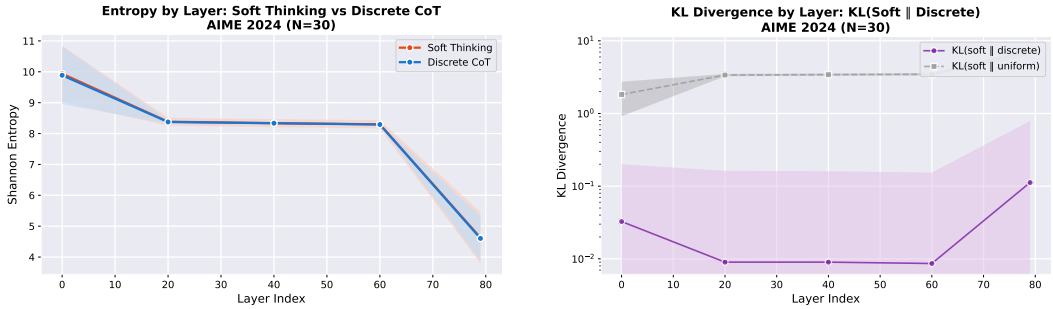
(a) Entropy comparison (b) KL divergence by layer

Figure 18: **Qwen2-1.5B on AIME2024 (N=30).** Entropy and KL divergence by layer.



(a) Entropy comparison (b) KL divergence by layer

Figure 19: **DeepSeek-R1-Distill-Llama-70B on MATH500 (N=500).** Entropy and KL divergence by layer. Logit Lens is applied at 5 evenly spaced layers $\{0, 20, 40, 60, 79\}$ of the 80-layer model.



(a) Entropy comparison

(b) KL divergence by layer

Figure 20: **DeepSeek-R1-Distill-Llama-70B on AIME2024 (N=30).** Entropy and KL divergence by layer. Same 5-layer probe as Figure 19; KL remains near zero from layer 20 onward.

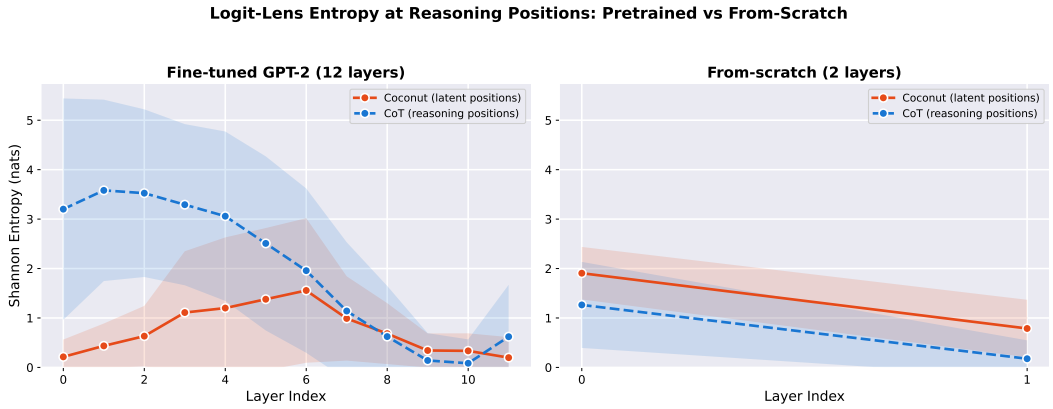
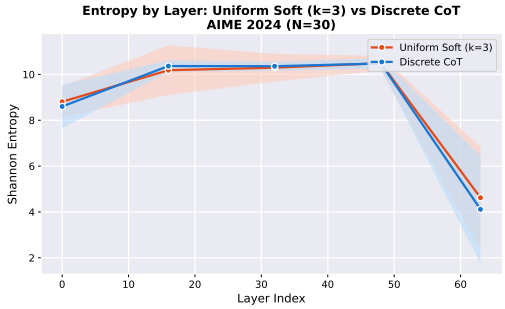


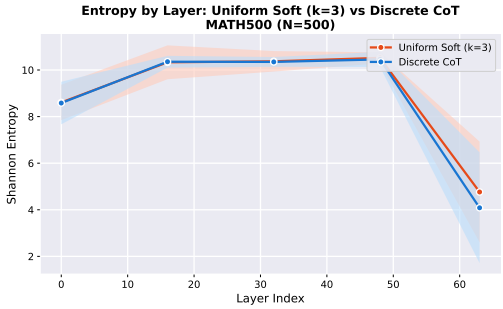
Figure 21: Logit Lens entropy at reasoning positions for fine-tuned GPT-2 (left, 12 layers) and a from-scratch 2-layer model (right). In the fine-tuned model, Coconut latent positions maintain uniformly low entropy across all layers, while CoT positions show high early-layer entropy that resolves by layers 8–9. In the from-scratch model, Coconut latent positions retain higher entropy than their CoT counterparts, consistent with the from-scratch model maintaining richer latent representations. Shaded regions denote ± 1 standard deviation across examples.

The finding is consistent across all conditions: uniform soft tokens collapse to near-argmax representations in middle layers, confirming that the collapse is not an artifact of the weighting scheme.

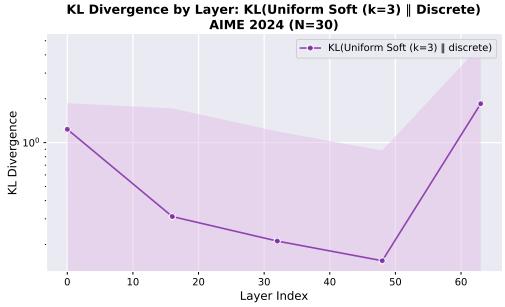
C.3.1 QwQ-32B



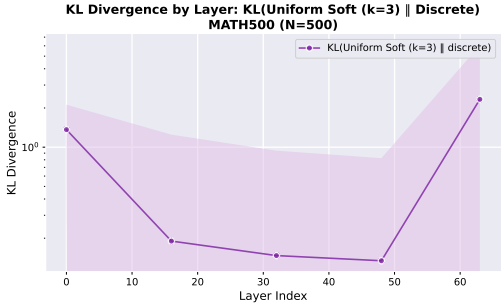
(a) Entropy — AIME 2024



(b) Entropy — MATH500



(c) KL divergence — AIME 2024



(d) KL divergence — MATH500

Figure 22: **QwQ-32B, uniform weighting** $k = 3$**.** Entropy profiles (top) and KL divergence (bottom) for AIME 2024 (left) and MATH500 (right). Despite uniform weights giving equal mass to 3 tokens, representations collapse to near-argmax in middle layers on both datasets.

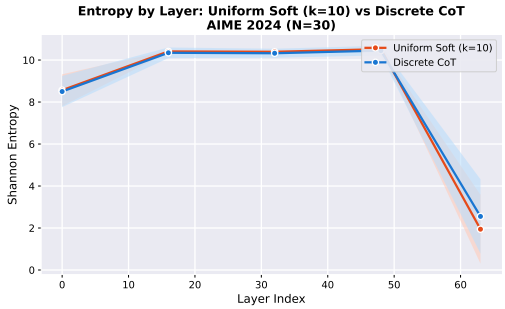
C.3.2 Qwen2-1.5B

C.4 Benchmark Results

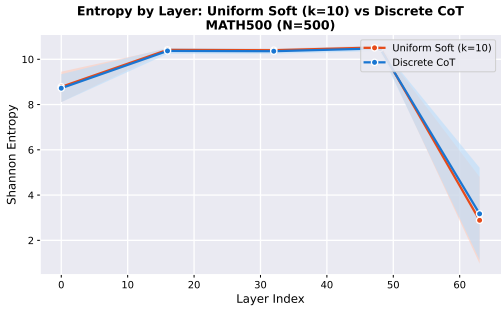
In this section we present a reproduction of accuracy results on MATH500 and AIME2024 for QwQ-32B. We find similar but not identical numbers to Zhang et al. (2025). We hypothesize this is due to numerical precision issues.

Table 7: MATH500 results with QwQ-32B (N=500). Discrete decoding uses standard greedy/sampling; Softmax and Uniform refer to the weighting scheme used to construct Soft Thinking embeddings.

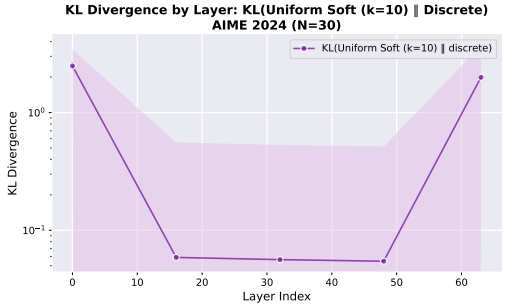
Run	Decoding	max_topk	Cold Stop	Accuracy (%)	Avg Tokens
1	Discrete	–	0.0 (none)	97.08	4,326
2	Discrete	–	0.1	97.08	4,326
3	Discrete	–	0.2	**97.19**	4,307
4	Softmax	10	0.0 (none)	96.47	4,222
5	Softmax	10	0.1	96.84	4,056
6	Softmax	15	0.01	96.84	4,044
7	Softmax	15	0.1	96.80	4,003
8	Uniform	3	0.1	93.97	5,388



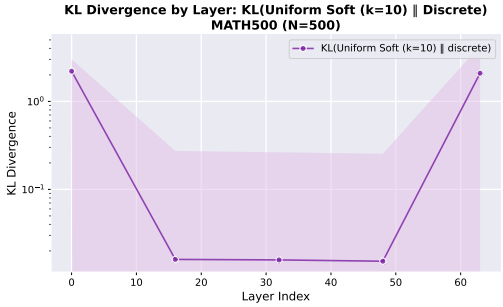
(a) Entropy — AIME 2024



(b) Entropy — MATH500



(c) KL divergence — AIME 2024



(d) KL divergence — MATH500

Figure 23: **QwQ-32B, uniform weighting** $k = 10$**.** Entropy profiles (top) and KL divergence (bottom) for AIME 2024 (left) and MATH500 (right).

Table 8: AIME2024 results with QwQ-32B (N=30).

Run	Decoding	max_topk	Cold Stop	Accuracy (%)	Avg Tokens
1	Discrete	–	0.0 (none)	**77.29**	13,445
2	Discrete	–	0.1	**77.29**	13,445
3	Discrete	–	0.2	**77.29**	13,445
4	Softmax	15	0.01	75.83	12,445
5	Softmax	15	0.1	76.25	11,818

D Coconut: Experimental Details

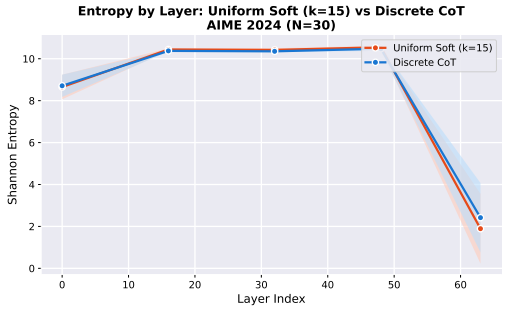
D.1 ProsQA Task

ProsQA (Hao et al., 2025) is a synthetic graph-traversal QA task designed to evaluate multi-hop reasoning. Each example consists of a randomly generated directed graph over named entities, a starting node, and a target node reachable via a sequence of directed edges. The question provides the graph structure (as a list of edges) and asks for the entity reachable from a given starting node after a specified number of hops. The ground-truth chain-of-thought consists of the sequence of intermediate entities visited during the traversal.

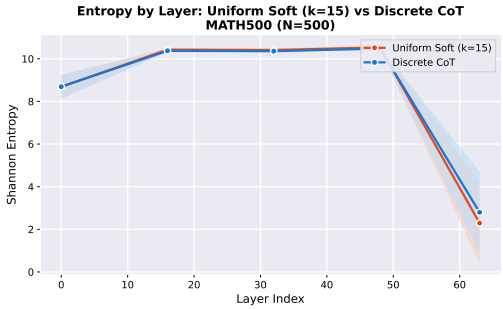
We use the ProsQA dataset provided with the original Coconut codebase, which contains 17,886 training examples and 300 validation examples. Graph depths range from 3 to 6 hops, with the distribution concentrated at 4 and 5 hops.

D.2 Training Setup

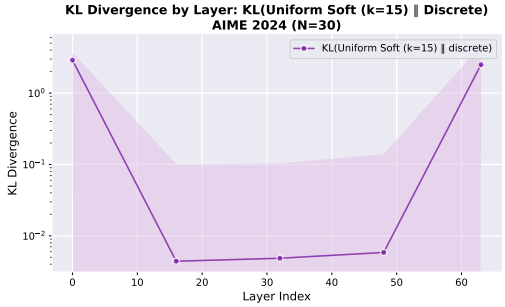
We train GPT-2 (124M parameters, 12 layers, hidden dimension 768) on ProsQA using the Coconut training procedure of Hao et al. (2025). Table 9 summarizes the key hyperparameters.



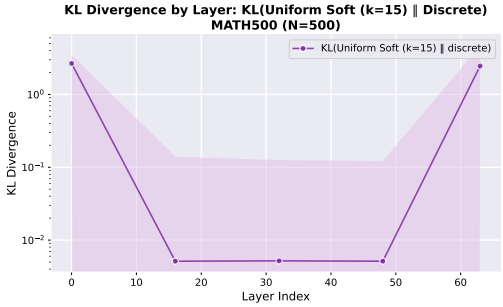
(a) Entropy — AIME 2024



(b) Entropy — MATH500



(c) KL divergence — AIME 2024



(d) KL divergence — MATH500

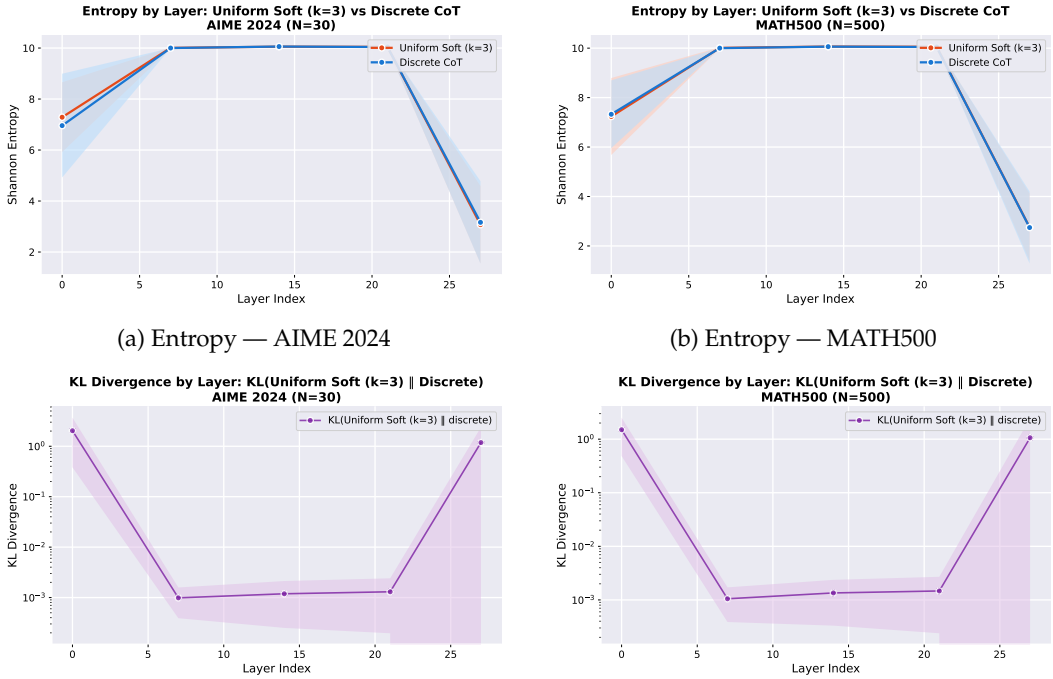
Figure 24: **QwQ-32B, uniform weighting $k = 15$.** Matching the top-k of the standard softmax experiments, uniform weighting still collapses in middle layers. The entropy profiles are indistinguishable from discrete CoT.

Table 9: Coconut training hyperparameters on ProsQA.

Parameter	**CoT baseline**	**Coconut**
Base model	GPT-2 (124M)	GPT-2 (124M)
Learning rate	10^{-4}	10^{-4}
Optimizer	AdamW	AdamW
Weight decay	0.01	0.01
Batch size (per device)	16	16
Gradient accumulation steps	2	2
Total epochs	50	50
Epochs per stage	–	5
Latent tokens per step (c_{thought})	–	1
Max latent stage	–	6
Precision	FP32	FP32
Random seed	0	0

Training follows a staged curriculum. At stage k (epochs $5k$ through $5(k+1)-1$), the first k chain-of-thought steps are replaced by k continuous latent tokens; the remaining steps are kept as discrete text. The model is trained to predict only the remaining CoT steps and the final answer (labels for question and latent positions are masked with -100). By stage 6 (epochs 30–34), all reasoning steps have been replaced by latent tokens, and the model must produce the answer using only continuous latent computation. The optimizer is reset at the start of each epoch (`reset_optimizer=True`).

Three special tokens are added to the vocabulary: `<|start-latent|>`, `<|latent|>`, and `<|end-latent|>`, all initialized from the embedding of the << token. During the forward pass, each `<|latent|>` token's embedding is replaced by the last hidden state from the



(a) Entropy — AIME 2024

(b) Entropy — MATH500

(c) KL divergence — AIME 2024

(d) KL divergence — MATH500

Figure 25: **Qwen2-1.5B, uniform weighting $k = 3$.** Entropy profiles (top) and KL divergence (bottom) for AIME 2024 (left) and MATH500 (right). Collapse is consistent with the 32B findings.

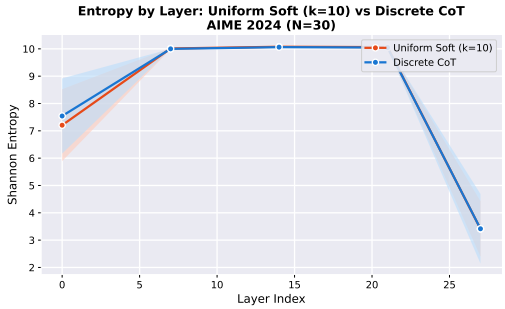
preceding forward pass, implementing the continuous thought recurrence. A custom collator left-pads batches to align latent token positions across examples, enabling KV cache reuse in the multi-pass forward.

We use `torchrun` with 4 GPUs; FSDP wraps the model but does not shard GPT-2's layers (effectively acting as DDP at this model scale). The best CoT checkpoint is at epoch 49 (85.3% validation accuracy); the best Coconut checkpoint is at epoch 50 (99% validation accuracy).

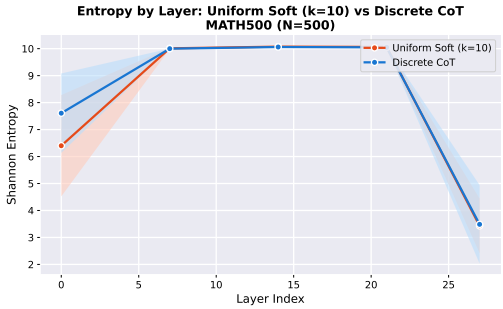
Reproducibility note. The fine-tuned Coconut experiments reported in Section 5 use a single training seed (seed = 0), consistent with the setup of the original Coconut paper (Hao et al., 2025). The core finding, that counterfactually removing the latent tokens changes the model's predictions by at most 1.0% across models, replicates under independently trained SmolLM2-135M, SmolLM2-360M and SmolLM2-1.7B checkpoints (see Table 10), providing evidence that the result is not seed-specific.

D.3 Entity Probing: Sample Sizes

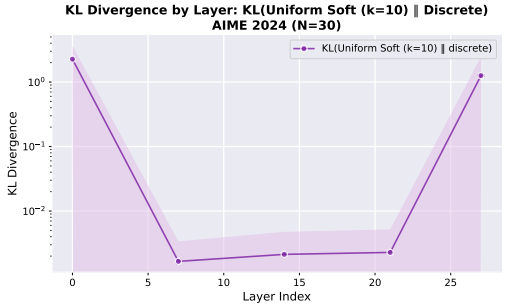
The entity probing results shown in Figures 3 and 28 use different sample sizes for the two conditions. The Coconut model is probed on all $N = 300$ ProsQA validation examples. The CoT model is probed on a $N = 50$ subset (the first 50 examples), due to the higher cost of teacher-forced probing over full CoT sequences. Both subsets are sampled from the same validation split in the same order, so the 50 CoT examples are a subset of the 300 Coconut examples. The qualitative pattern — target dominance from step 0 in Coconut vs. progressive correct-next tracking in CoT — is visually consistent and robust across both subsets, and holds on all step-count subgroups (3-step through 5-step) analyzed separately in Appendix E.1.



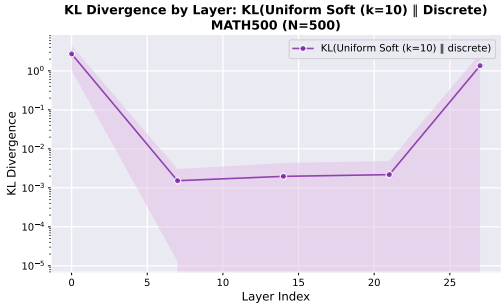
(a) Entropy — AIME 2024



(b) Entropy — MATH500



(c) KL divergence — AIME 2024



(d) KL divergence — MATH500

Figure 26: **Qwen2-1.5B, uniform weighting** $k = 10$**.** Entropy profiles (top) and KL divergence (bottom) for AIME 2024 (left) and MATH500 (right).

E Coconut: Additional Results

E.1 Entity Distributions

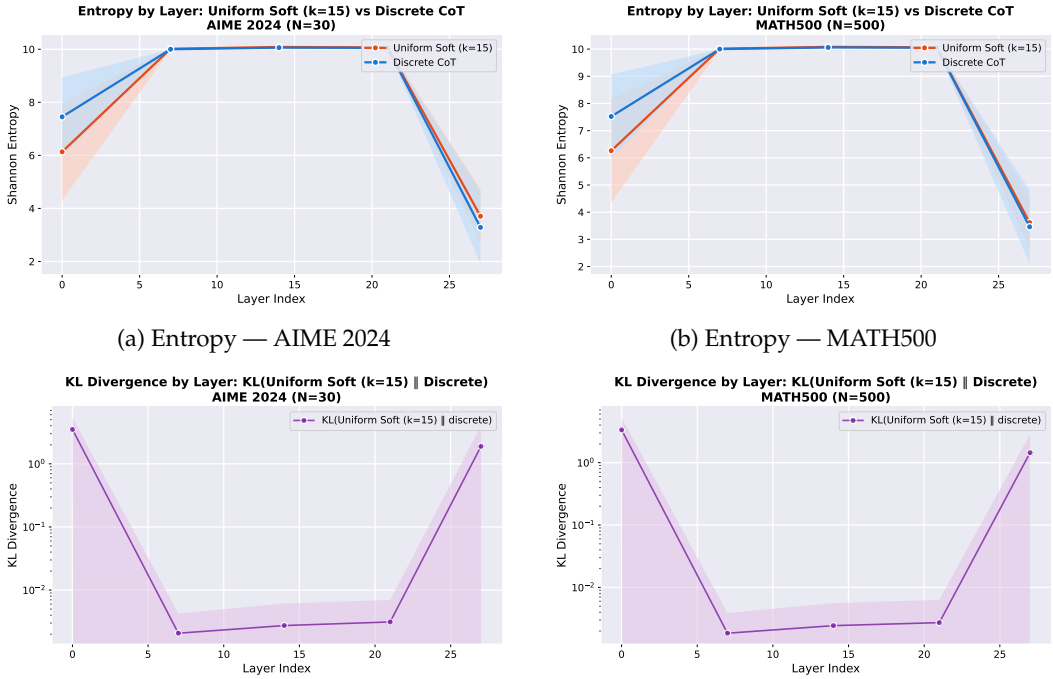
In this section, we present entity distribution probing results for different step counts or scenarios which are not presented in the main paper.

E.2 Coconut Other Results

In this section, we present additional experiments not presented in the paper. Mainly, we look at the entropy throughout layers for Coconut and CoT models on the ProsQA task, plot Coconut gradient norms throughout training and show entity belief plots for the ProntoQA (Saparov & He, 2023) task.

E.3 Coconut experiments on different models

We replicate the fine-tuned Coconut no-latent finding using independently trained SmolLM2-135M, SmolLM2-360M and SmolLM2-1.7B models on ProsQA. Each model was trained with a fresh random initialization and the same Coconut curriculum as the GPT-2 baseline; training seeds differ from the GPT-2 runs. To probe the effect of the latent tokens causally, we exploit the binary structure of ProsQA (queries of the form *"Is Rex a blicket or a gorple?"*): we remove the latent thoughts and score the log probabilities the model assigns to both candidate completions, thereby measuring counterfactually how it would have answered without any latent computation. Table 10 shows that this intervention changes the predictions by at most 1.0% across all four model families, confirming that the latent tokens are not load-bearing and that the behavior is not specific to GPT-2 or a single training seed.



(a) Entropy — AIME 2024

(b) Entropy — MATH500

(c) KL divergence — AIME 2024

(d) KL divergence — MATH500

Figure 27: **Qwen2-1.5B, uniform weighting** $k = 15$**.** Entropy profiles (top) and KL divergence (bottom) for AIME 2024 (left) and MATH500 (right). Results are consistent across both model scales: uniform superposition collapses regardless of k.

Table 10: Counterfactual no-latent evaluation on ProsQA across independently trained models. *Coconut (no latents)* removes all latent tokens and scores both candidate completions; Drop = Coconut (no latents) − Coconut.

Model	CoT	Coconut	Coconut (no latents)	Drop
GPT-2 (124M)	85.3	99.0	99.0	−0.0
SmolLM2-135M	72.7	93.3	92.3	−1.0
SmolLM2-360M	85.0	98.7	98.0	−0.7
SmolLM2-1.7B	98.3	100.0	100.0	−0.0

E.4 Ablation over embedding width

Table showing the importance of the width in models:

E.5 From-Scratch Coconut: Multi-Seed Results

To assess reproducibility of the from-scratch Coconut depth scaling results (Figures 21 and 29), we train three independently seeded runs (seeds 0, 1, 2) for each depth. Table 12 reports accuracy with and without latent tokens, and the resulting gap, as mean ± standard deviation across the three runs.

The 2L gap is large and tight across seeds (65.1 ± 9.0 pp). The 4L gap is similarly large on average but more variable (63.7 ± 23.7 pp), driven by one seed where the no-latent condition reaches 61.8% — suggesting that at this depth the model can sometimes learn shortcuts. At 8L and 12L the gap shrinks substantially, consistent with the main text argument that deeper models can extract the answer without latent tokens. These results confirm that the shallow-model reliance on latent tokens reported in the paper is not an artifact of a single training run.

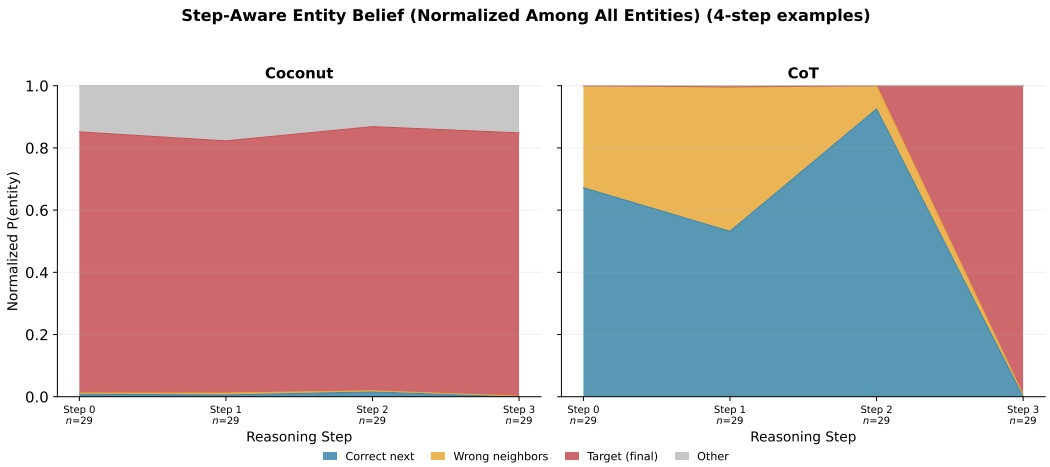


Figure 28: Normalized entity probability mass at each reasoning step for 4-step ProsQA examples (fine-tuned Coconut). Each stacked area shows the fraction of probability assigned to four entity categories: correct next hop (blue), wrong graph neighbors (orange), target/final answer (red), and other entities (gray). In the CoT model, probability mass shifts progressively toward the correct next entity at each step, consistent with step-by-step chain traversal. In the Coconut model, the target entity dominates from the first reasoning step onward, indicating that the model commits to the final answer without tracking intermediate hops.

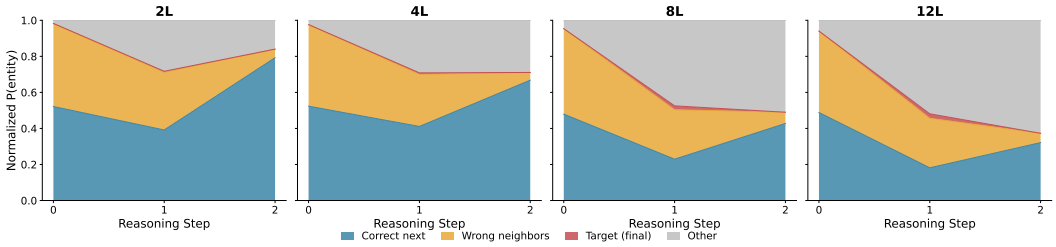
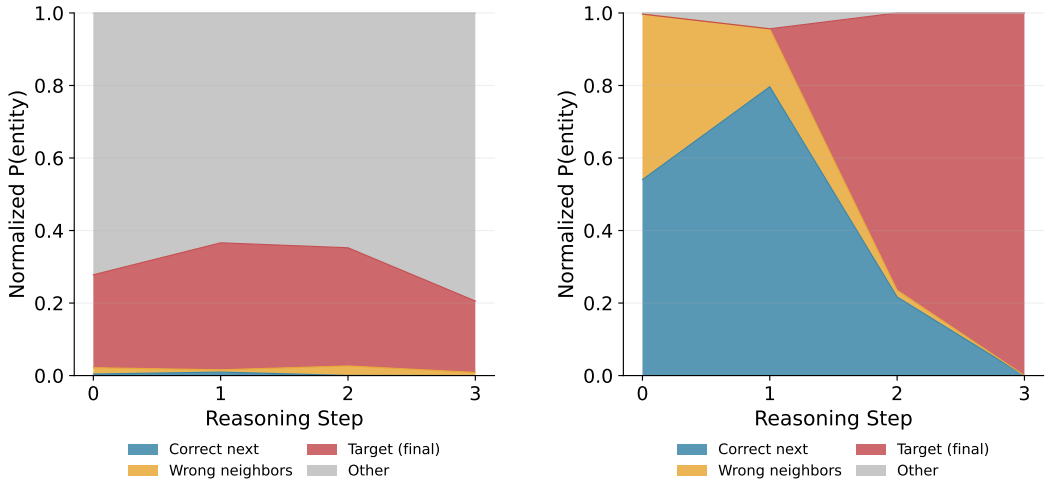
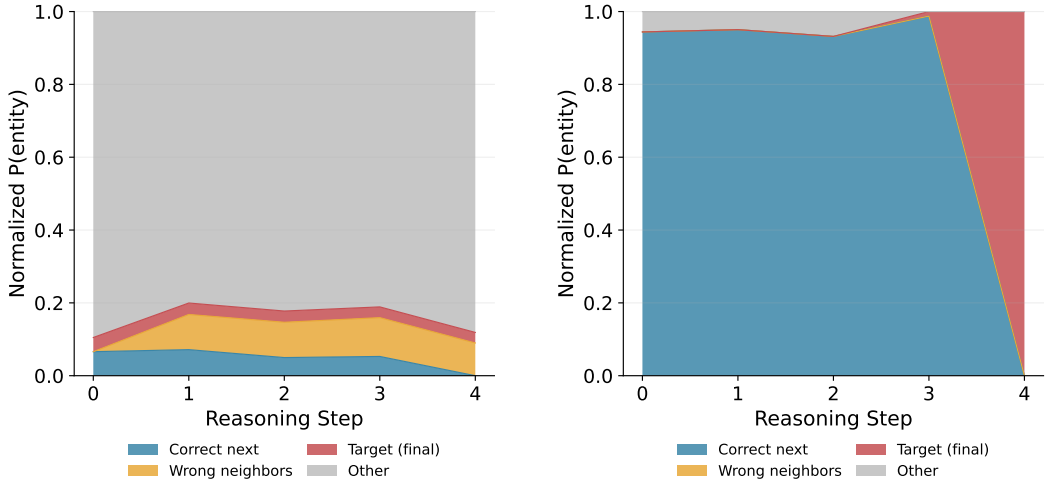


Figure 29: Step-aware entity belief across model depths for 3-step ProsQA examples (from-scratch Coconut). Each panel corresponds to a model trained from random initialization at a different depth (2L, 4L, 8L, 12L). Shallow models (2L, 4L) exhibit belief evolution across latent steps, with probability mass shifting toward the correct next entity at intermediate positions. Deeper models (8L, 12L) show no such progression; the target entity appears early without intermediate tracking.



(a) Coconut (latent positions) (b) CoT (reasoning positions)

Figure 30: Step-aware entity belief for the 3.4% of ProsQA test examples (17/500) that the Coconut model answers correctly with latent tokens but incorrectly without them. In the Coconut panel, the "other" category dominates (∼70%), with the target receiving only ∼25−35% of the probability mass; the model does not confidently commit to the final answer from the question embedding alone on these examples, unlike the general population (Figure 28). In contrast, the CoT model still performs step-by-step chain traversal on the same examples. This suggests that the latent tokens may be load-bearing for a small subset of examples where the question embedding does not suffice.



(a) Coconut (latent positions) (b) CoT (reasoning positions)

Figure 31: Normalized entity probability mass at each reasoning step on ProntoQA (fine-tuned Coconut). In the Coconut model, the "other" category dominates throughout (∼80%), with correct next, wrong neighbors, and target each receiving minimal probability mass. In contrast, the CoT model tracks the correct next entity at each intermediate step before transitioning to the target at the final step.

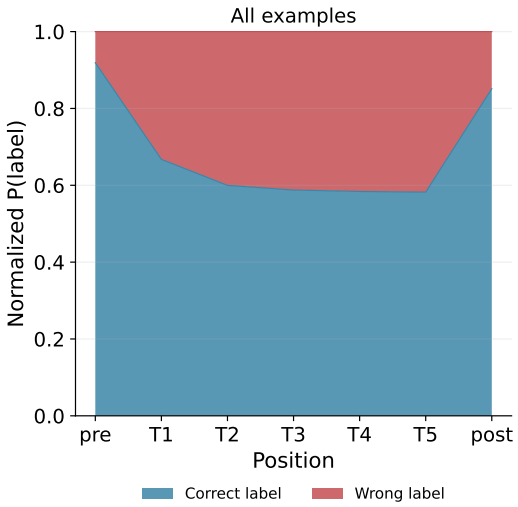


Figure 32: Normalized probability of the correct label ("True" or "False") across latent positions for the Coconut model on ProntoQA. At the pre-thinking position the model assigns ∼93% of the probability mass to the correct label; this drops to ∼60% during latent reasoning (T1–T5) before recovering at the post-thinking position. The dip during latent positions is consistent with the model redistributing probability mass without performing productive intermediate computation. Probing for these labels is more consistent with the ProntoQA task as the model must return a binary "True or False" response, as opposed to returning the entity name as is the case in ProsQA.

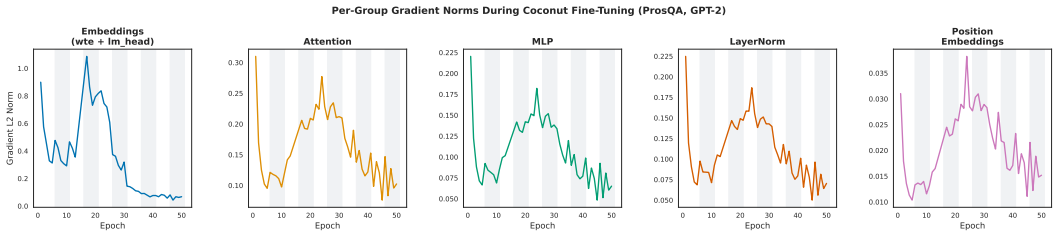


Figure 33: Mean L_2 gradient norms per parameter group during Coconut fine-tuning on ProsQA (GPT-2). Each panel shows one parameter group: word token embeddings (wte + lm_head), attention, MLP, LayerNorm, and positional embeddings (wpe). Alternating shaded bands denote the five-epoch training stages of the Coconut curriculum. Note that gradient magnitudes remain non-trivial across all groups and stages, indicating that the model parameters are being actively updated throughout training; however, as shown in Section 5, this training does not yield latent representations that participate in multi-step reasoning.

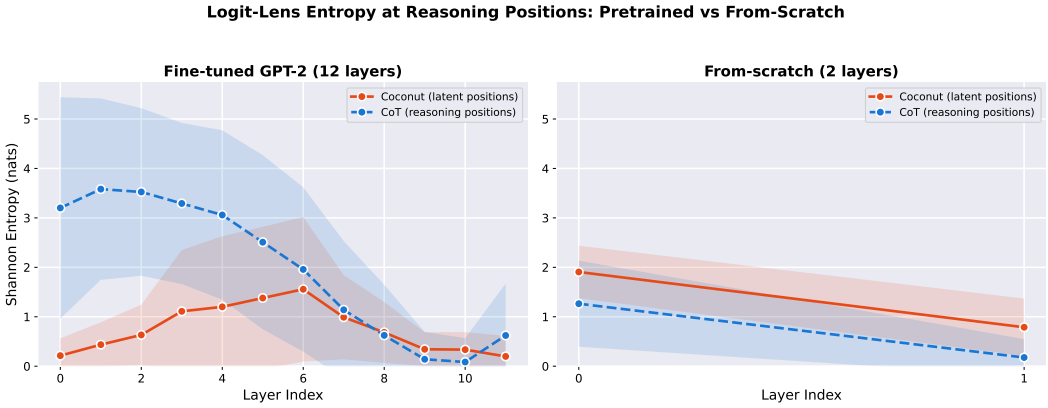


Figure 34: Logit Lens entropy at reasoning positions for fine-tuned GPT-2 (left) and a from-scratch 2-layer model (right). In the fine-tuned model, Coconut latent positions maintain near-zero entropy across all layers, consistent with early commitment to a single token. In the from-scratch model, this pattern reverses: Coconut latent positions retain higher entropy than CoT, suggesting richer latent representations. Shaded regions denote ± 1 standard deviation.

Table 11: Test accuracy (%) with and without latent tokens across embedding dimensions, for 2-layer and 4-layer from-scratch Coconut models on ProsQA. The gap (w/ $-$ w/o) measures reliance on latent reasoning; a large gap indicates the model genuinely uses latent tokens. Graph width ≈ 28 nodes; the theoretical lower bound is $\Omega(28)$.

	2-Layer			**4-Layer**		
d_{embd}	w/ latents	w/o latents	gap	w/ latents	w/o latents	gap
8	40.8	40.6	+0.2	35.3	35.1	+0.2
16	57.5	54.9	+2.6	55.6	55.4	+0.2
32	58.9	57.8	+1.1	53.9	53.2	+0.7
64	64.9	62.3	+2.6	63.2	62.5	+0.7
128	67.1	60.6	+6.5	61.3	60.4	+0.9
256	75.4	27.2	**+48.2**	66.6	60.6	+6.0
384	91.6	57.8	**+33.8**	90.5	32.7	**+57.8**
768	96.7	32.0	**+64.7**	96.2	16.0	**+80.2**

Table 12: From-scratch Coconut on ProsQA ($d_{\text{embd}} = 768$): accuracy (%) with and without latent tokens, across 3 independent seeds. Gap = w/ $-$ w/o. Shallow models (2L, 4L) show a large, consistent gap; deeper models show a smaller but more variable gap, consistent with the depth-dependent shortcut pattern discussed in the main text.

Depth	**w/ latents**	**w/o latents**	**gap**
2L	96.0 ± 0.9	30.9 ± 8.4	65.1 ± 9.0
4L	96.9 ± 1.2	33.2 ± 25.0	63.7 ± 23.7
8L	83.4 ± 2.5	67.9 ± 5.5	15.4 ± 3.1
12L	71.7 ± 8.8	54.4 ± 23.4	17.3 ± 26.8