Exploring System 1 and 2 communication for latent reasoning in LLMs

Julian Coda-Forno[1,2*] **Zhuokai Zhao**[3] **Qiang Zhang**[3] **Dipesh Tamboli**[3] **Weiwei Li**[3]

Xiangjun Fan[3] **Lizhu Zhang**[3] **Eric Schulz**[2] **Hsiao-Ping Tseng**[3]

[1]Technical University of Munich (TUM) [2]Helmholtz Munich [3]Meta

Abstract

Should LLM reasoning live in a separate module, or within a single model's forward pass and representational space? We study dual-architecture latent reasoning, where a fluent Base exchanges latent messages with a Coprocessor, and test two hypotheses aimed at improving *latent communication* over Liu et al. [2024b]: (H1) increase channel capacity; (H2) learn communication via joint finetuning. Under matched latent-token budgets on GPT-2 and Qwen-3, H2 is consistently strongest while H1 yields modest gains. A unified soft-embedding baseline—a single model with the same forward pass and shared representations, using the same latent-token budget—nearly matches H2 and surpasses H1, suggesting current dual designs mostly add compute rather than qualitatively improving reasoning. Across GSM8K, ProsQA, and a Countdown stress test with increasing branching factor, scaling the latent-token budget beyond small values fails to improve robustness. Latent analyses show overlapping subspaces with limited specialization, consistent with weak reasoning gains. We conclude dual-model latent reasoning remains promising in principle, but likely requires objectives and communication mechanisms that explicitly shape latent spaces for algorithmic planning.

1 Introduction

Large language models (LLMs) trained with web-scale pre-training and alignment have achieved impressive zero-shot reasoning capabilities across diverse tasks [Dubey et al., 2024, Hurst et al., 2024, Yang et al., 2025, Liu et al., 2024a]. Despite their strong performance, LLMs are often seen as "fast and fluent" rather than genuinely deliberative — resembling the intuitive, System-1 side of dual-process theories of cognition [Kahneman, 2011]. Indeed, recent surveys explicitly describe progress in reasoning LLMs as a shift from System-1-like heuristics to System-2-style deliberation [Li et al., 2025b], highlighting the need for architectures that support more structured reasoning.

The dominant approach today is chain-of-thought (CoT) reasoning [Wei et al., 2022, Liu et al., 2024a], where intermediate steps are verbalized in natural language. While effective, CoT incurs substantial token-level overhead, limits abstraction bandwidth, and constrains inference unnecessarily to the sequential, symbolic space of text [Chen et al., 2025, Qu et al., 2025]. As model sizes and context window grow, these inefficiencies become increasingly prohibitive, motivating the search for more compact and expressive reasoning representations.

Latent reasoning [Hao et al., 2024, Liu et al., 2024c, Geiping et al., 2025] offers an alternative that enables the model to perform multi-step inference internally within its continuous hidden states, surfacing only the final answer. This paradigm promises two key advantages. First, reasoning in

[*]Work done during an internship at Meta.

high-dimensional embeddings provides vastly greater expressive bandwidth than token sequences, potentially allowing richer intermediate computations [Zhang et al., 2025]. Second, for combinatorial problems, operating over structured latent abstractions can dramatically reduce the effective search space [Geiping et al., 2025]. Such ideas find parallels in cognitive science, where humans are believed to reason in internal “mentalese" before translating thoughts into language [Fodor, 1975].

Most latent-reasoning methods still ask a single network to do both fast association and slow deliberation; e.g., Coconut [Hao et al., 2024] feeds the model’s last hidden state back as input, forcing the same space to support next-token prediction and a putative “language of thought,” creating a representational tug-of-war. Neuroscience evidence instead points to partially distinct substrates (PFC for deliberative control; striatal circuits for habitual responses [Miller and Cohen, 2001, O’Reilly and Frank, 2006, Dolan and Dayan, 2013]), suggesting that separating roles could help. KV-cache coprocessors [Liu et al., 2024c] fit this separation, but reported gains are limited; we argue the bottleneck is *latent communication* between modules.

We therefore revisit the KV-coprocessor design with two changes aimed at strengthening communication: (i) *frozen-Base cache augmentation*, where the coprocessor writes cache edits that reach all layers of the Base, and (ii) *co-finetuning*, where the Base and coprocessor are trained jointly to make the Base “listen” to latent messages. Under matched token budgets and latent counts N_L, we evaluate on two model families, standard pre-training benchmarks, and reasoning tasks (GSM8K, ProsQA, a controlled Countdown stress test), and we analyze latent representations for specialization. This lets us test whether these architectures deliver genuine “latent reasoning” rather than merely adding compute.

In brief, our KV-coprocessor variants lower perplexity but yield limited reasoning gains over a unified single model baseline; within the Liu et al. [2024c] framework, our results seem to suggest that scaling N_L mostly adds compute rather than reasoning.

2 Related work

Reasoning in latent space. Latent-space methods shift multi-step inference from tokenized CoT into a small set of latent variables, decoupling compute from text length. Representative approaches include Coconut, which feeds continuous “thoughts” back into the model [Hao et al., 2024]; differentiable KV-cache augmentation, which separates a Base token predictor from a coprocessor that edits the cache [Liu et al., 2024b]; and compressed/implicit CoT that learns dense contemplation tokens or plans without emitting text [Cheng et al., 2024, Kurtz et al., 2024]. Recent surveys synthesize this trend and its claimed benefits [Li et al., 2025a, Zhu et al., 2025]. Despite clear token-efficiency gains, reported improvements on *reasoning* are mixed, motivating our study of how to train dual-model systems for effective latent communication.

Communication and coordination between models. Our focus on “latent communication” between a Base and a coprocessor connects to two strands. First, teacher–student transfer suggests that sharing an initialization can facilitate implicit trait transfer, even without explicit supervision [Cloud et al., 2025]; this supports initializing the coprocessor from the Base to align internal representations. Second, multi-agent prompting frameworks use inter-model dialogue to improve reliability [Du et al., 2023, Liang et al., 2023], though recent analyses find gains can be brittle or dataset-dependent [Wang et al., 2024]. Finally, our unified *soft-embedding* baseline is grounded in continuous-prompt methods that endow a single network with extra latent capacity via trainable prefix/prompt vectors [Lester et al., 2021, Li and Liang, 2021, Goyal et al.], providing a strong, parameter-efficient alternative to dual-model designs.

3 Methods

3.1 Problem setting

Setup and notation. Let B_θ be a frozen, pretrained LLM (*Base*) with parameters θ. Given a prompt x and target y, a forward pass of B_θ produces per-layer key–value caches $\{(K_\ell(x;\theta), V_\ell(x;\theta))\}_{\ell=1}^{L}$, abbreviated $KV_\theta(x)$. A coprocessor C_ϕ with parameters ϕ reads $KV_\theta(x)$ together with N_L learnable

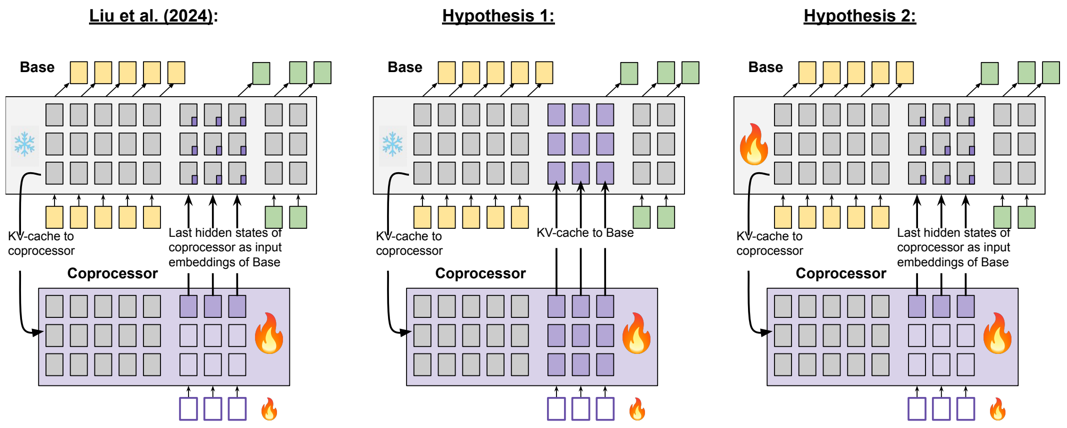


Figure 1: Overview of the deliberation–in–KV-cache architecture of Liu et al. [2024b] and our two variants designed to strengthen cross-module communication.

soft tokens and emits a latent sequence $Z \in \mathbb{R}^{N_L \times d}$. At decoding time, Z is injected back into B_θ (in a manner that depends on the variant below), after which the Base generates y.

Objective. The training goal is to learn latents that improve conditional likelihood:

$$\max_{\phi \ \text{(and possibly } \theta)} \ \mathbb{E}_{(x,y)\sim\mathcal{D}}\big[\log p_\theta\big(y \mid x, \text{inject}(Z; KV_\theta(x))\big)\big],$$

where $Z = C_\phi\big(KV_\theta(x), N_L\text{-soft}\big)$ and $\text{inject}(\cdot)$ denotes the chosen mechanism for feeding Z (or the coprocessor's cache) back to the Base.

Pipeline (as in Liu et al. [2024b]). We follow the standard three-stage process (cf. Fig. 1, left):

(i) *KV-cache generation.* Run B_θ on x once to obtain $KV_\theta(x)$.

(ii) *Latent augmentation.* Run C_ϕ on $KV_\theta(x)$ plus N_L learnable soft tokens to produce a latent sequence Z. In Liu et al. [2024b], the final hidden states of C_ϕ are used as input embeddings to B_θ.

(iii) *Decoding.* Inject the augmentation into B_θ and decode y. Unless otherwise noted, only ϕ and the soft-token embeddings receive gradients.

3.2 Experimental variants

We test two *hypotheses* aimed at strengthening communication between modules and isolating where the gains arise. Each hypothesis states a falsifiable prediction at matched latent-token budgets.

Hypothesis 1 — Frozen-Base KV augmentation. *Change:* Instead of converting the coprocessor's output into input embeddings, we concatenate its per-layer cache to the Base cache at injection time. Let $KV_\phi(x)$ denote the coprocessor's cache produced from $KV_\theta(x)$ and the N_L soft tokens. Decoding uses

$$[K_\ell(x;\theta)\,;\,K_\ell(x;\phi)] \quad \text{and} \quad [V_\ell(x;\theta)\,;\,V_\ell(x;\phi)] \qquad \forall\, \ell = 1,\ldots,L,$$

i.e., concatenation along the sequence (cache) dimension.[2] *Optimization:* θ remains frozen; only ϕ and the soft tokens are trained. *Motivation:* With a frozen Base, steering only via input embeddings gives the coprocessor influence primarily at early layers. Cache-level augmentation propagates the latent signal through *all* layers, potentially enabling richer, layer-wise "latent communication". *Prediction:* With θ frozen, cache augmentation will outperform embedding-only feedback Liu et al. [2024b] at matched latent-token budgets N_L.

[2]We denote sequence-axis concatenation by $[\cdot\,;\,\cdot]$.

Hypothesis 2 — Co-finetuned dual model. *Change:* Same injection mechanism as the reference system in Liu et al. [2024b] (latents as input embeddings to the Base), but we unfreeze the Base. *Optimization:* Update both θ and ϕ jointly (plus soft tokens) under the log-likelihood objective above. *Motivation:* Joint training allows B_θ to learn to "listen" to the coprocessor's messages rather than treating them as exogenous noise. Although this increases the number of trainable parameters, it isolates whether improved cross-module interaction—rather than mere extra compute—drives downstream gains. *Prediction:* For the same latent injection as Liu et al. [2024b], co-finetuning will outperform the frozen-Base counterpart at matched N_L, reflecting learned communication rather than added compute.

Implementation note. Both variants are realized within the same three-pass schedule used throughout the paper (§4.1): Base pass to form $KV_\theta(x) \to$ coprocessor pass to form Z (and, for Hyp. 1, $KV_\phi(x)) \to$ Base decode with injection. This preserves a clean separation between next-token prediction and latent computation.

4 Experimental analysis:

4.1 Large scale data training

We replicate the deliberation KV-cache pipeline of Liu et al. [2024b], but adapt it to a tighter compute budget and to two model families.

Base models: GPT-2 (124 M) Radford et al. [2019] and Qwen-3 (0.6 B) Yang et al. [2025] replace the 2-B-parameter Gemma used in the Liu et al. [2024c] paper. This choice allows us to complete all runs within a 3-day window on an 8 × A100 (80 GB) node.

Token budget: Liu et al. train for $\approx 2 \times 10^{11}$ tokens (sequence 2048, batch 1024, 100 k steps). We scale this down to 40 B tokens for GPT-2 and 8 B tokens for Qwen-3, which we found sufficient to reproduce their qualitative trends without exceeding our hardware envelope.

Dataset. All large-scale training uses *FineWeb-Edu-100BT* Lozhkov et al. [2024].

Sequence & latent parameters: We use sequence length S=1024, latent augmentations per sequence M=64, ahead tokens N_A=16 for back-propagating the loss into the coprocessor (Liu et al. [2024b] use 2048/128/16), and N_L=16 latents. Figure 2**C/D** show that validation perplexity decreases almost linearly as N_L grows under Hypothesis 2. Training cost scales with the effective per-example context $S + M \cdot (N_L + N_A)$, so larger N_L substantially raises wall-clock. Within our budget, N_L=16 provided a practical operating point, while Liu et al. [2024b] report their strongest results around N_L=32 on larger bases.

Three-pass training loop implementation (Fig. 5): Liu et al. [2024b] describe "an efficient training framework ... in one forward pass", enabled by a custom attention mask that scales to large datasets. Such a mask must support multiple augmentations (M) per sequence; otherwise only one augmentation is trained, yielding two orders of magnitude less signal per token. However, a single forward pass could allow the Base model to shortcut the intended latent computation—performing both next-token prediction and latent augmentation itself, thereby collapsing the two representational spaces and defeating the purpose of the technique. We cannot verify whether Liu et al. [2024b]. avoid this, as their code is not public; the attention masks they describe imply that only one model processes the pass, but the exact implementation remains unclear. To keep our dataset large while preventing this shortcut, we adopt a strict three-pass loop (Figure 5) whose attention masks still allow M augmentations of each sequence.

Only the coprocessor (and, in Hypothesis 2, the Base model) receives gradients. Although this three-pass schedule incurs a 3x wall-clock penalty compared with the unknown "single-pass" implementation, it preserves a clean separation between next-token prediction and latent reasoning—the property we wish to evaluate.

After convergence, models are evaluated greedily on standard pre-training benchmark for small LLMs: HellaSwag, ARC-Easy, SocialIQA, PIQA, and Winogrande.

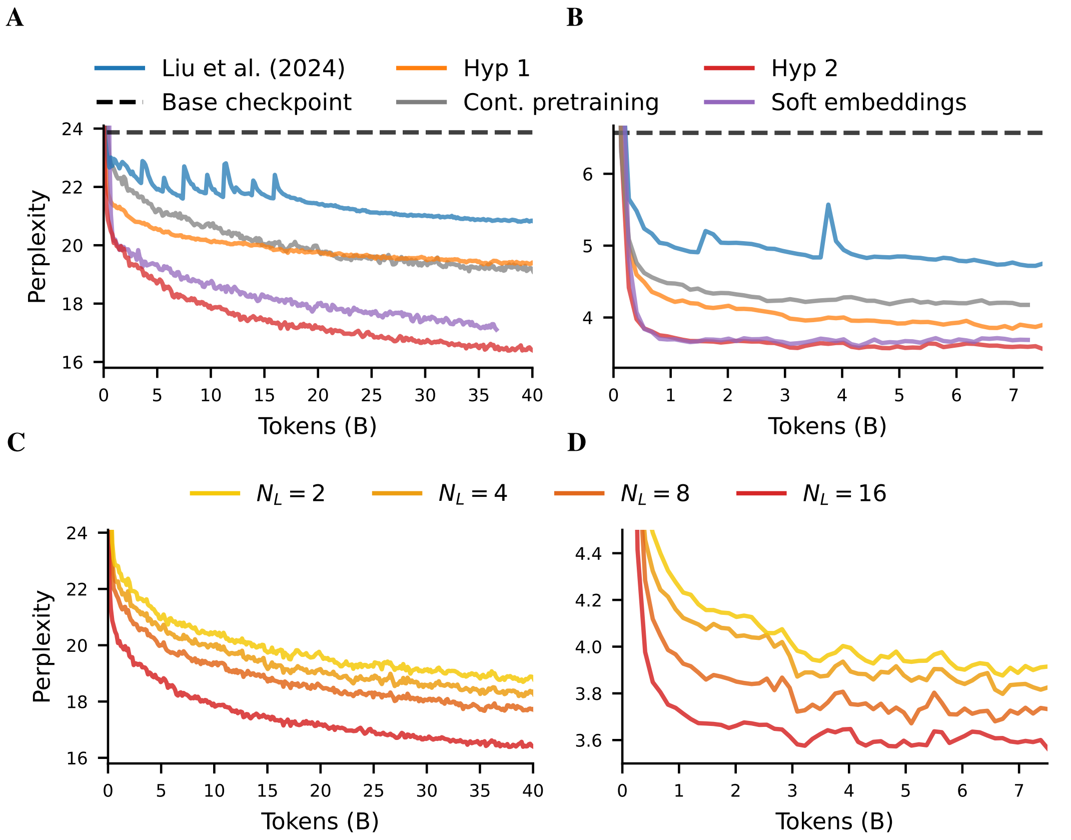


Figure 2: Validation perplexity whilst training on the FinWeb-Edu-100BT corpus. **A:** GPT-2 variants (using N_L=16 where applicable). **B:** Qwen-3 variants (using N_L=16 where applicable). **C:** Ablating number of latents N_L of the GPT-2 Coprocessor for Hypothesis 2. **D:** Ablating number of latents N_L of the Qwen-3 Coprocessor for Hypothesis 2.

4.1.1 Baselines

Baseline 1 - Base model + continued pre-training: Liu et al. [2024b], compare their dual-model to the initial base checkpoint, meaning the dual system has seen far more data. Instead, we keep the architecture unchanged and simply continue pre-training the base model on the **same number of tokens** used by our dual-model runs.

Baseline 2 - Base model + N_L soft embeddings: To test whether a single network can absorb the "System-2" role, we continue pre-training the base model alone while attaching the same number of new N_L learnable soft tokens used by the coprocessor setup (similar in spirit to Goyal et al., but with untied, slot-specific embeddings rather than a single shared token). This contrasts a unified versus dual-network design under identical data and latent embedding budgets. See App. A.1 for visualization.

Note. Neither baseline is strictly parameter-matched—our dual-model has twice as many weights—yet they provide useful sanity checks on data scaling and on the value of a separate coprocessor.

4.1.2 Results

Qualitative replication of Liu et al. [2024b]. Substituting the 2-B parameter Gemma with much smaller GPT-2 (124 M) and Qwen-3 (0.6 B) still produces the qualitative trends reported by Liu et al. [2024c](Fig. 2A/B). Perplexity (ppl) drops by 2 for GPT-2 and 2.5 for Qwen-3—an order-of-magnitude larger reduction than that reported in the original paper setup. On GPT-2 the dual-model also raises mean benchmark accuracy by +2.0 percentage point (pp) (Table 1). For Qwen-3, however, lower perplexity does not translate into higher benchmark scores; continued pre-training alone even

Model	Hellaswag	ARC-Easy	Social IQA	PIQA	Winogrande
GPT-2 variants					
GPT-2 continued pretraining	30.7 (+0.0)	54.2 (+0.0)	37.8 (+0.0)	64.5 (+0.0)	50.5 (+0.0)
GPT-2 + soft embeddings	30.7 (+0.0)	55.6 (+1.4)	37.7 (–0.1)	64.6 (+0.1)	51.9 (+1.4)
Liu et al. [2024b]	29.0 (–1.7)	45.0 (–9.2)	37.4 (–0.4)	62.1 (–2.4)	50.7 (+0.2)
Hypothesis 1	29.4 (–1.3)	48.2 (–6.0)	**38.7 (+0.9)**	62.9 (–1.6)	**52.3 (+1.8)**
Hypothesis 2	**31.2 (+0.5)**	**55.8 (+1.6)**	38.2 (+0.4)	**65.3 (+0.8)**	52.1 (+1.6)
Qwen variants					
Qwen continued pretraining	26.6 (+0.0)	34.5 (+0.0)	35.3 (+0.0)	55.1 (+0.0)	52.0 (+0.0)
Qwen + soft embeddings	25.9 (–0.7)	37.2 (+2.7)	34.6 (–0.7)	55.3 (+0.2)	50.4 (–1.6)
Liu et al. [2024b]	31.8 (+5.2)	36.5 (+2.0)	35.4 (+0.1)	58.1 (+3.0)	52.0 (+0.0)
Hypothesis 1	35.2 (+8.6)	45.0 (+10.5)	39.0 (+3.7)	61.0 (+5.9)	**55.7 (+3.7)**
Hypothesis 2	**37.3 (+10.7)**	**53.1 (+18.6)**	**41.9 (+6.6)**	**63.4 (+8.3)**	51.1 (–0.9)

Table 1: Model performance on standard small LLM benchmarks and Δ with continued pretraining baselines.

degrades accuracy. Given Qwen-3's heavy post-training, such brittleness under distribution shift is expected. We therefore report all numbers relative to the continued-pre-training baseline.

Both hypotheses outperform Liu et al. [2024b]. Relative to it's dual model, *Hypothesis 1* (frozen Base, cache concatenation) improves average accuracy by +1.5 pp on GPT-2 and +4.4 pp on Qwen-3. *Hypothesis 2* (co-finetuned) improves by +3.7 pp (GPT-2) and +6.6 pp (Qwen-3). In ppl, the variants are ≈ 2 and ≈ 4 lower, respectively (Fig. 2A/B).

Liu model versus data-matched continued pre-training. Relative to our *data-matched* Baseline 1 (same number of training tokens), the Liu dual model exhibits higher ppl ($\approx +1.5$) and mixed accuracy: -2.7 pp on GPT-2 but $+2.1$ pp on Qwen-3. Averaged across families the net change is ≈ -0.3 pp, underscoring the value of stricter baselines.

Dual models versus the "soft-embedding" baseline. Our second baseline—*Base model* + N_L *soft embeddings*—keeps the architecture unified yet endows the network with the same latent budget. Against this control the dual-model results are underwhelming:

- Perplexity: *Hyp. 1* has higher ppl than the soft-embedding model for both families; *Hyp. 2* lowers it only marginally.
- GPT-2 benchmarks: Averaged over the five tasks in Table 1, *Hyp. 1* trails by -1.8 pp (46.3 vs. 48.1%), while *Hyp. 2* is only $+0.4$ pp higher (48.5%).
- Qwen-3 benchmarks: Dual models fare better: *Hyp. 1* $+6.5$ pp (47.2 vs. 40.7%); *Hyp. 2* $+8.7$ pp (49.4%).

Summary and caveat. At first glance both dual-model variants look successful: they outperform the Liu et al. [2024c] baseline and, on Qwen-3, yield sizeable benchmark gains. But the picture shifts once we introduce the *soft-embedding* control. That unified model matches the dual systems in token budget and latent capacity while using only half as many trainable weights, yet it equals or exceeds *Hyp. 1* and comes within a hair of *Hyp. 2*. This pattern indicates that the coprocessor is not just "adding compute"—it is adding it *inefficiently*. A single LLM with the same aggregate parameter count would almost certainly do better. Accordingly, the next section turns to reasoning-specific benchmarks to see whether the dual architecture offers any benefit that a parameter-matched, soft-prompted model cannot already provide.

4.2 Reasoning evaluation: benchmarks and a controlled stress test

In this section we wanted to test if additional latent tokens improve *reasoning* or do they mostly add compute? We first compare our systems on GSM8K Cobbe et al. [2021] and ProsQA Hao et al. [2024], then use a controlled *Countdown* stress test Pan et al. [2025] to probe robustness as combinatorial difficulty increases. We evaluate four systems: our reproduction of Liu et al., Hyp. 1 (frozen Base, cache-concat), Hyp. 2 (co-finetuned), and a single-model soft-embedding baseline with the same total latent budget N_L. For reference, we also compare against Coconut (GPT-2) Hao et al. [2024]. More implementation details can be found in App. A.2.

Benchmarks. On GSM8K/ProsQA we follow the staged curriculum of Hao et al. [2024] and report accuracy after fine-tuning. Table 2 shows the best results ($N_L = 12$ for this task). Figure 3A plots *GSM8K* accuracy versus total latents N_L; the corresponding *ProsQA* curves are deferred to App. A.2, as this benchmark is near-saturated for competitive systems and adds limited insight.

Stress test: Countdown. Countdown lets us scale difficulty in a controlled, task-homogeneous way by increasing the operand count (branching factor grows rapidly with each operand). Unlike the benchmarks, we train without curriculum and sweep both $N_L \in \{1, 2, 4, 8\}$ and operands $\in \{3, 4, 5\}$. An illustrative instance appears in the box below. Figure 3B plots accuracy vs. operands, combining GPT-2 and Qwen curves for clarity.

Countdown example with operands = 4

User: Using the numbers $[19,\ 36,\ 55,\ 7]$, create an equation that equals 65.
Assistant: Let me solve this step by step.
`<latent thinking>`
`<answer>` $55 + 36 - 7 - 19$ `</answer>`

Findings. (i) *Rank order on GSM8K/ProsQA:* Hyp. 2 $\geq$ soft-embedding $\gg$ Hyp. 1 $\approx$ Liu (Table 2). (ii) *Scaling latents:* Fig. 3A shows accuracy is largely *flat* as N_L increases and can dip at larger N_L. This contrasts with Section 4.1, where perplexity decreased as N_L grew, suggesting that extra latents help next-token prediction but do not translate into more reliable reasoning. Interestingly, Coconut (GPT-2) exhibits the same dipping trend, indicating the difficulty is not specific to dual-model designs. (iii) *Stress test:* In Fig. 3B, increasing operands sharply reduces accuracy for all systems after $N_L = 8$; larger N_L helps slightly at low difficulty but yields diminishing or negative returns thereafter, and remains very close to the single-model soft-embedding baseline.

Table 2: Accuracy (%) on GSM8K and ProsQA after curriculum fine-tuning with $N_L = 12$.

	GPT-2		Qwen-3	
Model	GSM8K	ProsQA	GSM8K	ProsQA
Coconut (B0)	**34.1**	97.0	–	–
Soft emb. (B1)	26.5	97.7	38.5	**99.5**
Liu et al. (B2)	16.5	52.6	22.4	81.0
Hyp. 1 (frozen, concat)	12.0	54.1	24.0	79.0
Hyp. 2 (co-finetuned)	31.5	**99.0**	**38.6**	**99.5**

Efficiency note (Hyp. 2 vs. Coconut). Coconut generates continuous thoughts sequentially with the same network; for a given input this requires approximately $N_L{+}1$ full forward passes (each time appending a new latent and re-processing), so compute and latency scale linearly with N_L Hao et al. [2024]. Our Hyp. 2 uses a strict three-pass schedule independent of N_L (Base cache $\rightarrow$ Coprocessor latents $\rightarrow$ Base decode). Assuming similar model sizes, this yields an approximate forward-pass/FLOPs reduction of

$$\text{speedup} \approx \frac{N_L + 1}{3},$$

e.g., $\sim 5.7\times$ fewer full passes at $N_L{=}16$, while remaining batch-parallel (multiple augmentations per sequence can still be handled in one coprocessor pass). In practice this removes Coconut's serial bottleneck—while achieving comparable task accuracy on GPT-2 (GSM8K: 31.5 vs. 34.1; ProsQA: 99.0 vs. 97.0; Table 2, Fig. 3a)—enabling larger models and datasets. However, it is worth noting that Hyp. 2 also introduces roughly $2\times$ as many trainable parameters as Coconut.

Summary. Across reasoning benchmarks and the controlled stress test, adding latents mostly buys FLOPs, not robust reasoning. A unified soft-embedding model closely matches the best dual

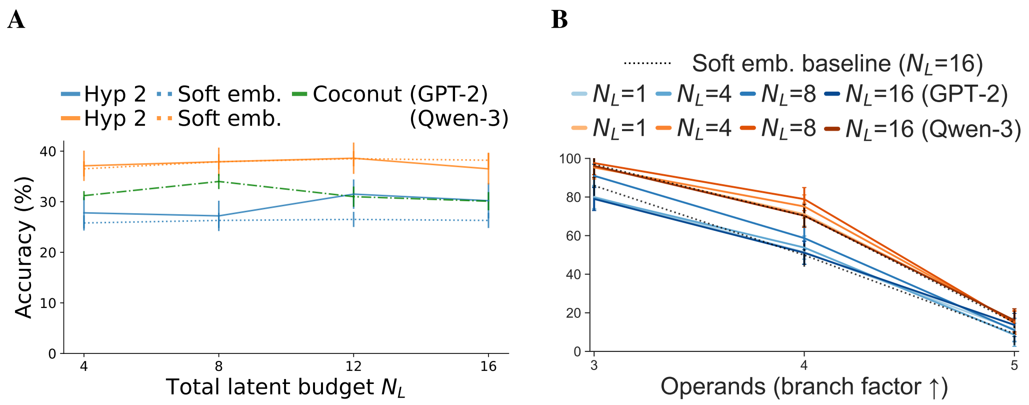


Figure 3: Ablating the latent budget. **A:** GSM8K accuracy vs. total latents N_L (GPT-2 and Qwen; Coconut shown for GPT-2). **B:** Countdown accuracy vs. operands (3, 4, 5) with lines for $N_L \in \{1, 4, 8, 16\}$, merged across model families.

model at equal N_L, and further increasing N_L rarely helps—sometimes hurts. This motivates the interpretability analysis in §4.3.

4.3 Interpretability: do latents specialize or collapse?

Reasoning can be viewed as composing *distinct* intermediate computations or modules Andreas et al. [2016], Lake and Baroni [2017], Veličković and Blundell [2021]. If our Hypothesis 2 coprocessor truly supports such division of labor, different *latents* should occupy meaningfully different directions in representation space. If, instead, latents mostly scale confidence without new algorithmic structure, they will reuse the same span, yielding *redundant* computations Olah et al. [2020], Elhage et al. [2022], Kornblith et al. [2019]. We therefore analyze the **coprocessor's last hidden layer** (the signal fed back as input embeddings to the Base) and test whether latents specialize.

Diagnostic 1: cross-subspace capture heatmap. For each latent i, collect its centered occurrences $X_i \in \mathbb{R}^{n_i \times d}$ and compute the minimal PCA basis $U_i \in \mathbb{R}^{d \times k_i}$ that explains at least τ of variance (we use $\tau = 0.97$). Let $P_i = U_i U_i^\top$ be the orthogonal projector onto latent i's subspace. For any latent j we measure the fraction of X_j's variance captured by P_i:

$$H_{i,j} = \frac{\left\|X_j P_i\right\|_F^2}{\left\|X_j\right\|_F^2} \in [0, 1].$$

We plot the matrix H as a heatmap (diagonal entries are $\geq \tau$ by construction) and summarize it with a single scalar,

$$\text{Distinctiveness} = \tfrac{1}{N_L(N_L-1)} \textstyle\sum_{i \neq j} H_{i,j},$$

where *lower is better* (less cross-capture $\Rightarrow$ more subspace diversity across latents).

Diagnostic 2: silhouette (cluster separation by latent). We compute the silhouette score Kaufman and Rousseeuw [1990], $s \in [-1, 1]$, which for each point compares how close it is to its *own* latent cluster versus the *nearest other* latent cluster; higher s means tighter, better separated clusters. This complements diagnostic 1: Instead of looking at directional variance reuse across latents (orientation overlap), it looks at instance-level spatial separation/cohesion of latent-labeled clusters. We report the global average (formal definition and the mean per-latent scores in App. §A.3).

4.3.1 Results

Large-scale training. Figure 4a is nearly uniform and bright off the diagonal, and the quantitative scores confirm this collapse-like behavior: mean off-diagonal capture $\overline{M}_{\text{off}} = 0.9873$ (higher $\Rightarrow$ more redundancy), global silhouette $s = -0.1694$ with per-latent silhouettes mostly negative ($\{-0.26, \ldots, -0.07\}$; full vector in Appendix A.4). Together these indicate that occurrences labeled by different latents largely occupy the *same* subspaces and are not cluster-separated.

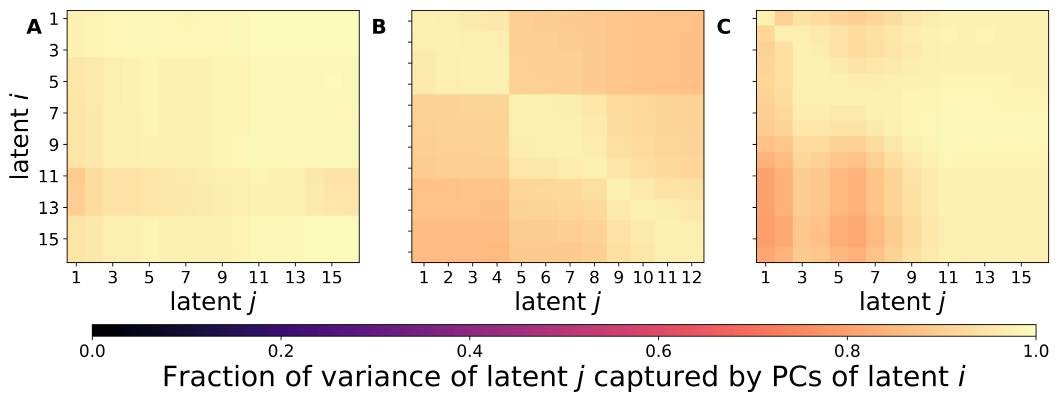


Figure 4: **Latent cross-subspace capture heatmaps** (last coprocessor layer). Each cell (i, j) shows the fraction of variance of latent j captured by the principal subspace of latent i. **A:** Large-scale training. **B:** Fine-tuning on GSM8k with curriculum. **C:** Finetuning on countdown with operands = 4.

GSM8K with curriculum. Figure 4b shows the curriculum–tuned model; the no–curriculum variant is in Appendix A.4. Relative to the large-scale pre-training task, fine-tuning reduces *directional redundancy* among latents: $\overline{M}_{\text{off}}$ drops to 0.962 without curriculum and to 0.914 with curriculum. However, cluster separation does not emerge: the global silhouette remains near zero in both task-tuned settings ($s \approx -0.031$). In short, curriculum nudges representations in the right direction—latents become less overlapping in orientation—but they still occupy essentially the same region of space (decorrelated directions without spatial separation).

Countdown (operands = 4). Countdown induces substantially better cluster separation (global silhouette $s = 0.4531$), yet the cross-subspace redundancy remains high with $\overline{M}_{\text{off}} = 0.9382$. This realizes the "*separated but redundant spans*" regime: clusters are distinct in Euclidean space (high silhouette) but their principal subspaces still explain most of each other's variance (high off-diagonal capture). Interestingly, Fig 4c shows distinctiveness degrading beyond latent 8, aligning with Fig 3B where accuracy improves from $N_L \in \{1, 4, 8\}$ but drops at N_L=16.

Across all three settings the latents tend toward redundant computations: their subspaces are highly overlapping (high $\overline{M}_{\text{off}}$), even when the data task encourages class separation (Countdown). Task-aligned fine-tuning helps (curriculum < pretraining in $\overline{M}_{\text{off}}$), but not enough to yield "separated and specialized" latents (high silhouette and low off-diagonals).

5 Discussion

Building on Liu et al. [2024b], we cast the coprocessor architecture as a principled attempt to disentangle "System-1" token prediction from "System-2" abstract reasoning. Treating latent embeddings as a private communication channel between the two modules clarifies the conceptual link to dual-process theories and motivates our two new training hypotheses. However, in our setting, simply providing the channel (Liu et al. [2024b]) or strengthening it (H1/H2) did not induce System-2-like computation.

Replacing last-layer hidden-state injection with full KV-cache concatenation yields modest perplexity and benchmark gains over the original design, yet falls short once stricter baselines are introduced. Its benefits appear sensitive to model family and vanish on harder reasoning tasks. Jointly updating both models produces the strongest results across all experiments, confirming that bidirectional adaptation facilitates cross-model communication. However, a *unified* network trained with the same soft-token budget narrows—sometimes erases—the gap, suggesting that current dual-system instantiations add compute inefficiently rather than unlocking qualitatively new reasoning abilities.

The soft-embedding baseline, parameter-matched continued pre-training, and Coconut all highlight scenarios where dual-model gains either disappear or can be matched closely by simpler means. Our Countdown experiments further show that scaling the latent budget beyond eight tokens fails to deliver systematic robustness as combinatorial complexity explodes. Our interpretability analysis seem to corroborate this by showing that learned latents largely occupy overlapping representational subspaces: extra latents mostly amplify confidence rather than add new algorithmic structure.

Our negative results do not invalidate the dual-system framework; they indicate that *how* the two models exchange information remains an open problem, and that current instantiations do not create conditions for System-2-like computation to emerge. Promising directions include (i) designing objectives that explicitly reward diversity or orthogonality in latent representations to encourage broader search, and (ii) developing training schedules that preserve large-scale language competence while gradually shaping latent spaces for multi-step reasoning.

References

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models. *arXiv preprint arXiv:2503.09567*, 2025.

Yuxin Cheng, Yao Fu, Ruochen Xu, et al. Compressed chain-of-thought: Learning dense contemplation tokens for reasoning. *arXiv preprint arXiv:2412.13171*, 2024.

Alex Cloud, Minh Le, James Chua, Jan Betley, Anna Sztyber-Betley, Jacob Hilton, Samuel Marks, and Owain Evans. Subliminal learning: Language models transmit behavioral traits via hidden signals in data. *arXiv preprint arXiv:2507.14805*, 2025.

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

Ray J. Dolan and Peter Dayan. Goals and habits in the brain. *Neuron*, 80(2):312–325, 2013. doi: 10.1016/j.neuron.2013.09.007.

Yilun Du, Shizhe Li, Antonio Torralba, and Joshua B. Tenenbaum. Improving factuality and reasoning in language models through multiagent debate. In *NeurIPS 2023 Workshop on LLMs*, 2023. arXiv:2305.14325.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407, 2024.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, et al. Toy models of superposition. *arXiv preprint arXiv:2209.10652*, 2022. URL `https://arxiv.org/abs/2209.10652`.

Jerry A Fodor. *The language of thought*, volume 5. Harvard university press, 1975.

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. *arXiv preprint arXiv:2502.05171*, 2025.

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*.

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, 2024. URL `https://arxiv.org/abs/2412.06769`.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Daniel Kahneman. *Thinking, fast and slow*. macmillan, 2011.

Leonard Kaufman and Peter J Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley, 1990.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3519–3529, Long Beach, California, USA, Jun 2019. PMLR. URL `http://proceedings.mlr.press/v97/kornblith19a.html`.

Alexander Kurtz, Eric Zelikman, Samuel R. Bowman, Christopher D. Manning, Noah D. Goodman, Fei Sha, Joshua B. Tenenbaum, et al. Quiet-star: Language models can learn to think without externalized chain-of-thought. *arXiv preprint arXiv:2403.09629*, 2024.

Brenden M. Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. *arXiv preprint arXiv:1711.00350*, 2017. Version commonly cited in 2018; we reference the arXiv preprint for clarity.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *EMNLP 2021*, 2021. arXiv:2104.08691.

Jing Li, Rui Wang, Yiming Zhao, et al. A survey on latent reasoning in large language models. *arXiv preprint arXiv:2507.06203*, 2025a.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

Zhong-Zhi Li, Duzhen Zhang, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu, Junhao Zheng, Pei-Jie Wang, Xiuyi Chen, Yingying Zhang, Fei Yin, Jiahua Dong, Zhiwei Li, Bao-Long Bi, Ling-Rui Mei, Junfeng Fang, Xiao Liang, Zhijiang Guo, Le Song, and Cheng-Lin Liu. From system 1 to system 2: A survey of reasoning large language models, 2025b. URL `https://arxiv.org/abs/2502.17419`.

Yichen Liang, Yuxiang Gu, Jian Yang, et al. Language is not all you need: Aligning agents with multi-agent debate. *arXiv preprint arXiv:2305.12674*, 2023.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024a.

Luyang Liu, Jonas Pfeiffer, Jiaxing Wu, Jun Xie, and Arthur Szlam. Deliberation in latent space via differentiable cache augmentation, 2024b. URL `https://arxiv.org/abs/2412.17747`.

Luyang Liu, Jonas Pfeiffer, Jiaxing Wu, Jun Xie, and Arthur Szlam. Deliberation in Latent Space via Differentiable Cache Augmentation, December 2024c. URL `http://arxiv.org/abs/2412.17747`. arXiv:2412.17747 [cs].

Anton Lozhkov, Loubna Ben Allal, Leandro von Werra, and Thomas Wolf. Fineweb-edu: the finest collection of educational content, 2024. URL `https://huggingface.co/datasets/HuggingFaceFW/fineweb-edu`.

Earl K. Miller and Jonathan D. Cohen. An integrative theory of prefrontal cortex function. *Annual Review of Neuroscience*, 24(1):167–202, 2001. doi: 10.1146/annurev.neuro.24.1.167.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 2020. doi: 10.23915/distill.00024.001. URL `https://distill.pub/2020/circuits/zoom-in/`.

Randall C. O'Reilly and Michael J. Frank. Making working memory work: A computational model of learning in the prefrontal cortex and basal ganglia. *Neural Computation*, 18(2):283–328, 2006. doi: 10.1162/089976606775093909.

Jiayi Pan, Junjie Zhang, Xingyao Wang, Lifan Yuan, Hao Peng, and Alane Suhr. Tinyzero. https://github.com/Jiayi-Pan/TinyZero, 2025. Accessed: 2025-01-24.

Xiaoye Qu, Yafu Li, Zhaochen Su, Weigao Sun, Jianhao Yan, Dongrui Liu, Ganqu Cui, Daizong Liu, Shuxian Liang, Junxian He, et al. A survey of efficient reasoning for large reasoning models: Language, multimodality, and beyond. *arXiv preprint arXiv:2503.21614*, 2025.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Petar Veličković and Charles Blundell. Neural algorithmic reasoning. *Patterns*, 2(7):100273, 2021. doi: 10.1016/j.patter.2021.100273.

Xuezhi Wang, Jason Wei, Denny Zhou, et al. Rethinking the bounds of llm reasoning: Are multi-agent methods actually reasoning? In *ACL 2024*, 2024. arXiv:2402.15764.

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.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

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

Ziqi Zhu, Ziyang Chen, Zichen Li, et al. Reasoning beyond language: A comprehensive survey on latent chain-of-thought reasoning. *arXiv preprint arXiv:2502.04043*, 2025.

A Appendix

A.1 Large scale training implementation details

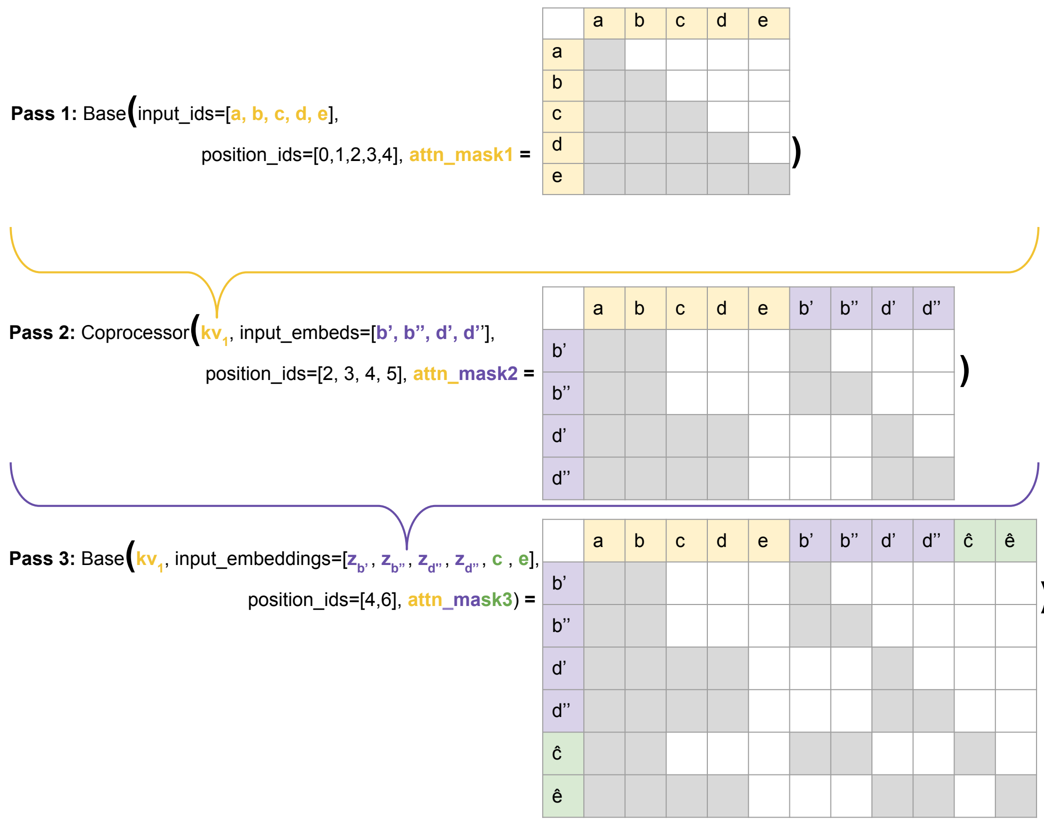

Pass 1: Base(input_ids=[a, b, c, d, e],
position_ids=[0,1,2,3,4], attn_mask1 =
Pass 2: Coprocessor(kv1, input_embeds=[b', b'', d', d''],
position_ids=[2, 3, 4, 5], attn_mask2 =
Pass 3: Base(kv1, input_embeddings=[zb', zb'', zd', zd'', c, e],
position_ids=[4,6], attn_mask3) =
a b c d e b' b'' d' d'' ĉ ê

Figure 5: **Three-pass schedule on a toy sequence.** Example with input “abcde” and two augmentation sites at b and d (M=2), N_L=2 latent slots per site (b$'$, b$''$, d$'$, d$''$), and N_A=1 ahead token per site. *Pass 1* (**Base**): run once on the raw sequence to form the cache $KV_\theta(x)$. *Pass 2* (**Coprocessor**): consume $KV_\theta(x)$ together with the *concatenated* $M \times N_L$ latent placeholders [b$'$, b$''$, d$'$, d$''$], producing per-slot latent embeddings $z_{b'}, z_{b''}, z_{d'}, z_{d''}$. *Pass 3* (**Base** decode): reuse the same $KV_\theta(x)$ and feed the *concatenated* outputs plus all $M \times N_A$ ahead tokens (one per site in this toy) as input embeddings (e.g., $[z_{b'}, z_{b''}, z_{d'}, z_{d''}, \hat{c}, \hat{e}]$). Concatenation in Passes 2–3 enables many augmentations per sequence to be trained in parallel while keeping latent computation (Pass 2) separated from next-token prediction (Pass 3). We are still unsure by the single-pass implementation sketched by Liu et al. [2024b] (no public code), but this schedule preserves disentanglement without sacrificing batch parallelism.

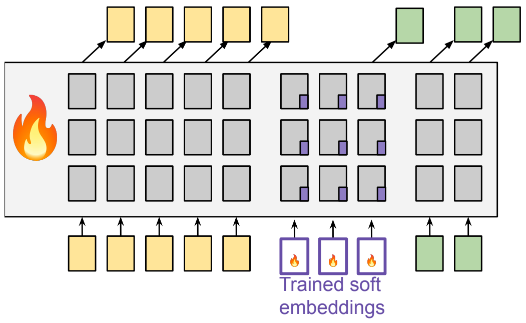
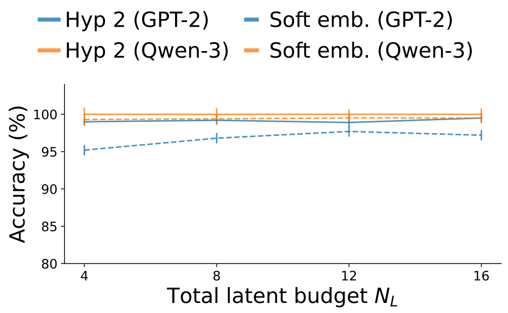


Figure 6: **Soft-embedding baseline (main control).** A single transformer (no coprocessor) plays both "System-1" and "System-2" roles. We attach N_L learnable soft tokens ("latent slots") to the Base and continue pre-training the same Base checkpoint LLM and these tokens on the same dataset and token budget as the dual-model runs. This matches the dual setup's latent capacity (N_L) while using half the trainable parameters, directly testing whether a separate coprocessor—and its cache-level latent communication with the Base—provides benefits beyond added compute. During decoding, the learned soft tokens serve as the latent context via input-embedding injection.

A.2 Reasoning implementation details

Benchmarks and curriculum. For GSM8K Cobbe et al. [2021] and ProsQA Hao et al. [2024] we follow the staged curriculum of Hao et al. [2024]. At stage k, the first k CoT steps are replaced by $k \times c$ continuous latents. Loss is masked on question tokens and on the latent spans; only the remaining language tokens contribute to the objective. We sweep $c \in \{1, 2, 3, 4\}$ with $k{=}4$, so the total latent budget is $N_L{=}k\,c \in \{4, 8, 12, 16\}$. All benchmark fine-tuning runs use a learning rate of $1{\times}10^{-4}$. To make the comparison to Hao et al. [2024] as faithful as possible, we deliberately minimize changes to their curriculum recipe; better schedules may exist for our dual-architecture, but exploring them is out of scope here.



Notes. ProsQA is near-saturated for both GPT-2 and Qwen-3, so curves are flat across N_L and the soft-embedding baseline tracks Hypothesis 2 closely. This complements Fig. 3A (GSM8K), where accuracy likewise fails to grow monotonically with N_L. Plotting conventions match the main text; dotted lines denote the single-model soft-embedding baseline.

Figure 7: ProsQA accuracy vs. total latents N_L.

Countdown setup. Countdown is trained *without* curriculum. Operands are sampled uniformly from $[1, 50]$ and the target from $[0, 100]$; expressions use $+$, $-$, $\div$ and $\times$ (as in TinyZero Pan et al. [2025]). For each $(\text{operands}, N_L)$ setting we train for 3 epochs over 262,144 generated training samples and evaluate on 1,024 held-out samples. We vary the operand count $\in \{3, 4, 5\}$ and latent budget $N_L \in \{1, 2, 4, 8\}$. All runs use a learning rate of $1{\times}10^{-4}$. The branching factor grows with operands as $\text{Catalan}(n{-}1)\, 2^{n-1}\, n!$, yielding the controlled difficulty axis reported in Fig. 3B.

Fine-tuning from large-scale pre-training. On GSM8K/ProsQA, resuming from large-scale pre-training *hurts* relative to starting from base checkpoints (Tables 3–4), suggesting a mismatch between next-token pre-training geometry and our curriculum-based supervision at this scale.

Comparison protocols (all end curriculum at $N_L{=}16$).

- *Hyp. 2 (scratch):* co-finetune Base and Coprocessor from the standard base checkpoints; staged curriculum with $k{=}4$ stages and $c{=}4$ latents per stage.
- *Resume + curriculum:* initialize Hyp. 2 from the large-scale training checkpoints (Sec. 4.1) and run the same curriculum ($k{=}4$, $c{=}4$).
- *Straight-to-16:* initialize from the large-scale checkpoints but skip the curriculum; a single stage with $k{=}1$, $c{=}16$.

Table 3: Effect of resuming from large-scale pre-training on **GSM8K**. Accuracy (%); Δ is the change vs. Hyp. 2 (scratch).

Model	**Hyp. 2 (scratch)**	**Resume + curriculum**	**Straight-to-16**
GPT-2	31.5	26.0 ($\Delta - 5.5$)	28.2 ($\Delta - 3.3$)
Qwen-3	38.6	35.7 ($\Delta - 2.9$)	37.3 ($\Delta - 1.3$)

Table 4: Effect of resuming from large-scale pre-training on **ProsQA**. Accuracy (%); Δ is the change vs. Hyp. 2 (scratch).

Model	**Hyp. 2 (scratch)**	**Resume + curriculum**	**Straight-to-16**
GPT-2	99.0	97.1 ($\Delta - 1.9$)	97.3 ($\Delta - 1.7$)
Qwen-3	99.5	88.3 ($\Delta - 11.2$)	84.5 ($\Delta - 15.0$)

By contrast, on Countdown (no curriculum) resuming helps at $N_L{=}16$: Fig. 8 compares three curves per family (GPT-2 left, Qwen-3 right)—Hypothesis 2 (from scratch), Hypothesis 1, and Hypothesis 2 resumed from large-scale training—and the resumed Hypothesis 2 dominates. One plausible explanation is that Countdown's objective is closer to the pre-training signal (no curriculum, homogeneous task family), whereas GSM8K/ProsQA curriculum alters the supervision structure in a way that conflicts with the pre-trained latent geometry. A fuller exploration of how to transfer large-scale pre-training into reasoning-specific curricula is a promising direction for future work.

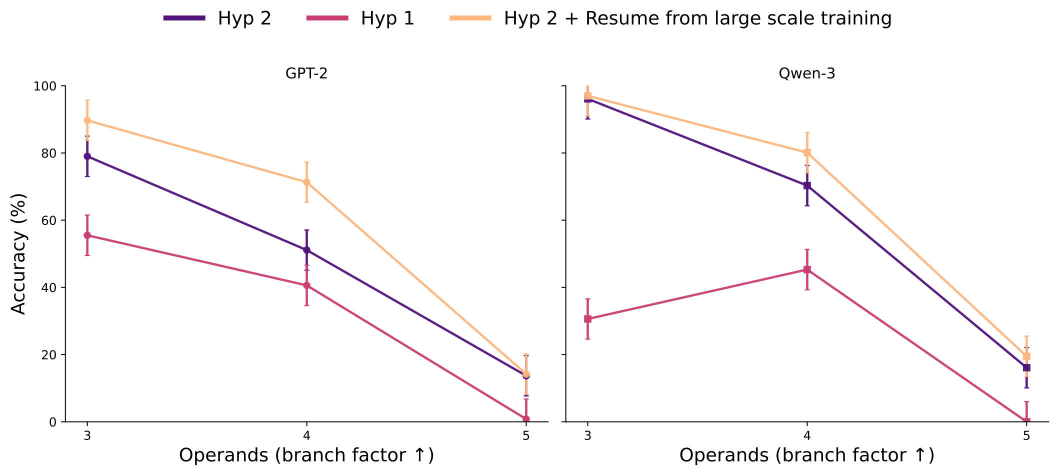


Figure 8: Countdown appendix: accuracy at $N_L{=}16$ for Hypothesis 2 (from scratch), Hypothesis 1, and Hypothesis 2 resumed from large-scale pre-training. Left: GPT-2. Right: Qwen-3. Resuming improves Countdown despite hurting GSM8K/ProsQA, hinting at a mismatch between curriculum-based supervision and pre-trained latent geometry.

A.3 Silhouette definition

Let $X = \{x_p\}_{p=1}^{N} \subset \mathbb{R}^d$ be all occurrences from the coprocessor's last hidden layer, labeled by latent indices $y_p \in \{1, \ldots, N_L\}$ with clusters $C_\ell = \{p : \ y_p = \ell\}$. For $p \in C_\ell$, define the mean intra-latent distance

$$a_p \;=\; \frac{1}{|C_\ell|-1}\sum_{\substack{q\in C_\ell\\ q\neq p}}\|x_p - x_q\|_2,$$

and, for $\ell' \neq \ell$, the mean distance

$$d(p, C_{\ell'}) \;=\; \frac{1}{|C_{\ell'}|}\sum_{q\in C_{\ell'}}\|x_p - x_q\|_2, \qquad b_p \;=\; \min_{\ell'\neq\ell} d(p, C_{\ell'}).$$

The silhouette of p is

$$s_p \;=\; \frac{b_p - a_p}{\max\{a_p,\, b_p\}} \in [-1, 1].$$

We report the global silhouette $s = \frac{1}{N}\sum_{p=1}^{N} s_p$ and the per-latent silhouette $s_\ell = \frac{1}{|C_\ell|}\sum_{p\in C_\ell} s_p$. For singleton clusters, we set $s_p = 0$.

A.4 Additional interpretability results

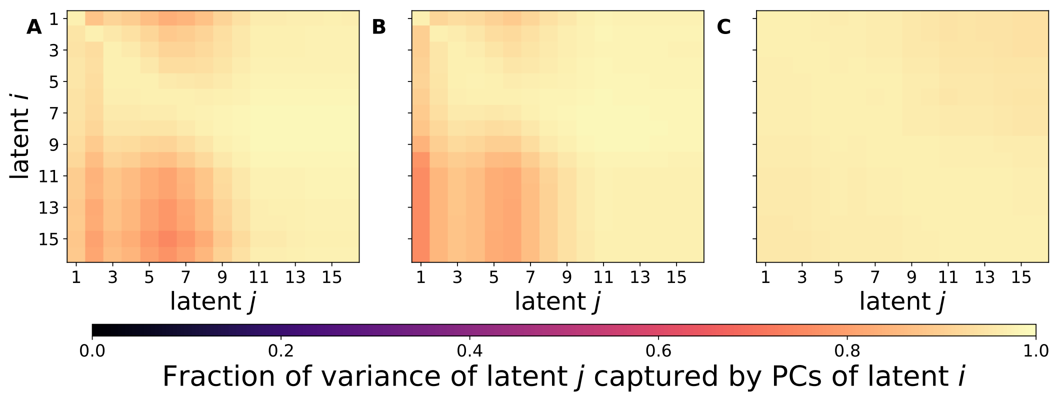


Figure 9: **Supplementary latent cross-subspace capture heatmaps** (coprocessor last layer). **A:** *Countdown* with operands = 3. **B:** *Countdown* with operands = 5. **C:** *GSM8K* without curriculum. Panels A/B are qualitatively similar to the main-paper operands = 4 plot; panel C shows stronger collapse than its curriculum counterpart.

Latent	Large-scale pretrain	GSM8K (curr.)	Countdown (op=4)
1	-0.158	-0.014	0.912
2	-0.089	-0.036	0.506
3	-0.176	-0.036	0.324
4	-0.173	-0.032	0.209
5	-0.259	-0.044	0.243
6	-0.113	-0.047	0.550
7	-0.067	-0.047	0.635
8	-0.143	-0.045	0.629
9	-0.117	-0.032	0.723
10	-0.170	-0.040	0.742
11	-0.202	-0.022	0.568
12	-0.252	0.019	0.357
13	-0.193	–	0.336
14	-0.133	–	0.461
15	-0.191	–	0.026
16	-0.211	–	0.030

Table 5: **Per-latent silhouette** s_ℓ . Dashes indicate unused latents in the GSM8K run (here N_L=12). Large-scale pretraining shows uniformly negative silhouettes (cluster intermixing); GSM8K with curriculum hovers near zero (weak separation); Countdown exhibits strong separation for most latents while still showing redundancy in cross-subspace capture (see main-text heatmaps).