VQ-CoT: Discretising Latent Chain-of-Thought

Maria Pilligua | 414423, Chengheng Li Chen | 415680,
Nil Biescas | 416078, Pau Amargant Alvarez | 408221
RateLimit Achieved

Abstract

Latent chain-of-thought (CoT) replaces reasoning tokens with continuous "thought" vectors, promising CoT-level accuracy at lower cost, yet whether these latents truly reason is largely untested. We probe what they encode in two ways: by *discretising* them with vector quantisation (VQ-VAE, FSQ), and by mechanistically dissecting the latents of Coconut and CODI with logit lenses, probes, ablations, and content swaps. Jointly Training Coconut and VQ-VAE collapses, instead training with FSQ and two-stage VQ-VAE is able to encode genuine signal in the codebooks, The latents themselves are highly structured: Coconut assigns each position a fixed role (numbers, operators, connectors), while CODI's separator latents are content-free placeholders that transcribe the data's delimiter format. Our analysis cautions against over-attributing reasoning to latent thoughts and points toward leaner latent-CoT designs.

1 Introduction

Latent chain-of-thought (CoT) is a promising direction for efficient reasoning: instead of spelling out every step in text, models such as Coconut (Hao et al., 2024) reason in continuous "thought" vectors that reportedly encode rich, even parallel, reasoning. This promise rests on a strong assumption, namely that the latents genuinely *compute* intermediate reasoning. Because the latents are opaque, the assumption is rarely tested, and on GSM8K (Cobbe et al., 2021) Coconut still trails explicit CoT (34% vs. 43%).

We supply the missing evidence from two complementary angles. First, we ask whether the thoughts can be made human-readable by discretising them with vector quantisation into a learned codebook of reasoning units. Second, we mechanistically dissect CODI (Shen et al., 2025), the strongest latent-CoT model to date: it matches explicit CoT via self-distillation, so its latents provably carry CoT-level information and are an ideal testbed for asking *what* that information is. Our contributions are (i) three VQ/FSQ discretisation schemes with a Δ_{info} diagnostic of codebook reliance, (ii) a mechanistic analysis showing CODI's separator latents are dispensable, content-free filler, and (iii) a Qwen3 scaling study (up to 4B) of how latent CoT and discretisation behave on larger backbones. Together these probes return one answer: the latents carry some content, chiefly numbers and running results, but much of their structure transcribes dataset format, and what remains is low-dimensional.

2 Approach

2.1 Latent CoT: Coconut and CODI

Explicit CoT generates a textual rationale before the answer. Latent CoT replaces it with L continuous thought vectors $z_1, \dots, z_L$: the model runs L extra forward passes, feeding each step's last-layer hidden state back as the next input embedding, then decodes the answer from the question and $z_{1:L}$. We use $L{=}6$. The models differ only in training. **Coconut** (Hao et al., 2024) uses a multi-stage curriculum that replaces one more textual reasoning step with a latent at each stage. **CODI** (Shen et al., 2025) trains in a single stage by self-distillation, aligning the latent student's answer-token hidden state to a CoT-reading teacher, which avoids curriculum forgetting and lets CODI match explicit-CoT accuracy.

2.2 Discretising Latent Thoughts

VQ-VAE (van den Oord et al., 2017) replaces each latent $z_i \in \mathbb{R}^d$ with the nearest entry of a learned codebook; gradients bypass the argmin via a straight-through estimator (Bengio et al., 2013) and the codebook is updated with exponential moving averages. **FSQ** (Mentzer et al., 2024) instead rounds each coordinate of a low-dimensional projection to a fixed integer grid, which prevents codebook collapse by construction at the cost of a lossier encoder. In both cases the quantised vector is reconstructed to $\mathbb{R}^d$ before re-entering the latent loop, leaving the LLM interface unchanged.

We train three variants. VQ-JOINT trains the VQ-VAE and Coconut together, letting codebook and reasoning co-adapt. VQ-PTR is two-stage: the VQ-VAE is first trained alone, then frozen while Coconut learns to use it. FSQ follows the joint setup with the FSQ quantiser.

2.3 Qwen3 Scaling

We run the Coconut curriculum on Qwen3 0.6B, 1.7B, and 4B Instruct 2507 (hidden sizes 1024, 2048, 2560) with the original schedule (c_thought=2, max_latent_stage=3, hence six latents), evaluating on the standard GSM8K splits. To probe the latents at scale we use the VQ-PTR recipe storing the six latent vectors of every example, then fit (a) a VQ-VAE (dimension 64, K=4096, EMA updates, k-means initialisation, short autoencoder warm-up) and (b) a TopK sparse autoencoder (dictionary 8× hidden width, k=32 active features). Both bottlenecks are fit on train split latents only; all metrics use the held out test split (1319 examples), and continuous and VQ accuracies share the same Stage 3 checkpoint and generation settings, so any change is attributable to the bottleneck alone.

2.4 CODI: Mechanistic Analysis

We analyse the released CODI GPT-2 checkpoint without retraining. CODI trains six alternating latents (num/sep/num/sep/num/sep) with distillation at the answer token only, so per-position content is unsupervised. We run five analyses: **positional ablation** (keep only number latents z_1, z_3, z_5 or only separators, zeroing the rest); **content swap** (replace separator vectors with those of another example); **per-layer logit lens**; a **3-latent student** distilled to match the teacher's number-latent hidden states ($z_1, z_3, z_5 \rightarrow s_1, s_2, s_3$); and **CoT alignment**, testing whether z_{2k-1} decodes to the k-th running result r_k of the GSM8K-Aug CoT string.

3 Experiments

Data. All experiments use GSM8K with standard train-test splits; the CODI analysis additionally uses GSM8K-Aug for its explicit step delimiters. No preprocessing is applied beyond each model's native tokenisation and formatting.

Evaluation. We report GSM8K accuracy. To measure whether the LLM actually uses the discrete latents we introduce Δ_{info}, the accuracy drop when trained codebook embeddings are replaced by zeros or random vectors: a large drop means the model relies on the quantised pathway, a small one means it is ignored. We further analyse codebook utilisation, logit-lens projections, and UMAP structure of the latents (McInnes et al., 2018). **Interpretability Methods** We use logit lens (nostalgebraist, 2020) which projects a hidden state through the unembedding matrix to read off its most probable token, giving a human-readable label without training.

4 Results and Analysis

4.1 Discretising Coconut

All three variants reduce accuracy relative to the Coconut baseline (Table 1). VQ-JOINT loses 8.5 pp while collapsing to a single active code, and this collapse is consistent with the reconstruction and commitment losses. (Figures 2 and 3). The collapse originates in the bottleneck encoder, whose trained effective rank falls below that of a randomly initialised MLP of the same shape, so the cosine codebook quantises a near one-dimensional manifold and every position snaps to the same centroid. VQ-PTR shows slightly higher accuracy (-7.8 pp) and effectively learns to use 2523 codes from its codebook. FSQ, with K=65k, stays closest to the baseline (-5.6 pp) on a healthy 4280-code grid.

Similar accuracy hides opposite mechanisms. Table 1 shows that zeroing VQ-JOINT's codes costs almost nothing (Δ_{info}=$+0.3$ pp): the LLM ignores the collapsed codebook and answers from the question stem, so the codes act as pause tokens. VQ-PTR and FSQ codes carry real signal, with zeroing costing $+11.7$ and $+13.3$ pp; the best FSQ grid is the most content-dependent of the three.

A logit lens on the VQ-PTR latents shows each position specialising into a consistent `# NUM WORD # NUM #` template (Appendix Table 3); the # slots are GSM8K's own delimiters, a first sign of format transcription. The codes are not purely formatting, though: a minority are sharply tied to a single arithmetic operator (the most peaked giving $P(+\mid c)$=0.93), so some genuine reasoning content does survive the bottleneck. A UMAP of the raw latents reveals three clusters that align with the same partition into markers, numbers, and words (Appendix Figure 1). For FSQ, we observe for

Model	Variant	GSM8K accuracy (%)					Decomposition (pp)			Code-usage diagnostics		
		parent	child	zero	random	no-lat	Δ_{info}	Δ_{pause}	Δ_{total}	used/K	perp.	cos.
GPT-2	VQ-JOINT (K=1024, cos)	34.4	25.9	25.5	21.0	22.3	+0.3	+3.3	+3.6	1/1024	1.0	+0.95
GPT-2	VQ-PTR (K=16k, Eucl)	34.4	26.6	14.9	17.2	17.0	+11.7	−2.0	+9.6	2523/16384	752.2	+0.39
GPT-2	FSQ (K=1000, d=4)	34.4	24.8	18.9	18.8	14.9	+5.9	+4.0	+9.9	698/1000	506.7	+0.01
GPT-2	FSQ (K=65,625, d=7)	34.4	**28.8**	15.5	14.3	12.7	+**13.3**	+2.9	+**16.1**	4280/65625	2681	0.02
Qwen3-0.6B	VQ-PTR (K=4096, Eucl)	23.6	23.6	23.6	23.6	0.0†	0.0	—	—	3107/4096	2791	+0.01
Qwen3-1.7B	VQ-PTR (K=4096, Eucl)	31.5	31.5	31.5	31.5	0.0†	0.0	—	—	1519/4096	1639	+0.24
Qwen3-4B	VQ-PTR (K=4096, Eucl)	39.7	39.7	39.7	39.7	0.0†	0.0	—	—	1873/4096	1727	+0.46

Table 1: Mechanism decomposition per variant. The continuous *parent* is the shared GPT-2 Coconut baseline (454/1319 = 34.4%);Accuracy columns: continuous parent, quantised child, then the child with codebook embeddings zeroed, randomised, or the latent loop removed. Δ_{info} = child − zero, Δ_{pause} = zero − no-lat, Δ_{total} = child − no-lat. Diagnostics: unique codes used at inference over capacity K, inference perplexity, mean pairwise codebook cosine. Bold marks the best accuracy and the largest deltas.

a 1K codebook, 698 active codes. When embeddings are projected into vocabulary space, these collapse to 33 tokens, dominated by numbers (77.2%, 772 codes), followed by markers (14.4%), operators (4.5%), and words (3.9%). Increasing to 64K codes, this expands to 773 distinct tokens and shifts strongly toward words (91.8%), while numbers drop to 2.3% (1165 codes). In terms of representational efficiency, a 768-dimensional float32 latent (24,576 bits) is compressed into 9.97 bits (1K codes) or 16.00 bits (64K codes), respectively. At low capacity, the model prioritizes numerical encoding almost exclusively. Increasing capacity enables broader lexical coverage for GSM8K-style text, while the number of numerical codes remains nearly constant, suggesting a saturation point around ~ 1100 codes for numbers, after which additional capacity is mainly used to encode linguistic structure rather than improve numerical representation.

4.2 Scaling Coconut to Qwen3

Accuracy peaks at Stage 0 and degrades with every latent step (Appendix Table 4): the curriculum hurts stronger backbones. Do these latents carry anything the model uses? We apply the Δ_{info} ablation of Section 4.1 to the frozen Stage 3 latents.

Discretisation is free at scale because the latents are never read. Inserting the trained VQ-VAE into the frozen latent loop leaves GSM8K test accuracy unchanged at every scale (Table 1, bottom block). To find the cause, we evaluate the continuous parent, the quantised child, and the child with its codebook zeroed or replaced by random noise. If the model used the latent content, this corruption would lower accuracy. Instead, all variants answer the same test examples correctly (311, 415, and 524 of 1319 at 0.6B, 1.7B, and 4B), so Δ_{info}=0 pp at every scale: the frozen backbone does not use the content of its latent slots. This is not the codebook collapse seen on GPT-2. The Qwen3 codebooks stay in use, with perplexity in the thousands and 1.5k to 3.1k codes covering 90% of usage (Appendix Figure 5), and reconstruction error decreases with scale (test MSE 1.38, 0.50, 0.07). The quantiser therefore represents the latents accurately, but the model answers without them. We next examine what the unread slots contain.

The latents are position-specialised and low-dimensional. Although unread, the latents are structured. Each slot uses a separate part of the codebook: adjacent slots share almost no codes, and 97% of 4B codes serve a single position. A principal-component projection of the codebook shows the same separation, complete on 4B (Appendix Table 5, Figures 6 and 7). This structure may come from each slot's position in the iterated forward pass rather than from semantic content. The logit lens is consistent with this reading: only 1.7B has an interpretable slot (position 0, 97% number mass, as on GPT-2), and most codes decode to abstract subword fragments (Appendix Table 6, Figure 8). The latents are also low-dimensional: a TopK sparse autoencoder explains 89.9 to 99.5% of each thought, and its test-active features drop from about 6300 (0.6B, 1.7B) to 180 on 4B, since 95% of the 4B latent variance lies in two to three principal components (Appendix Table 7, Figure 4). Across scales the result is the same: a compact, position-indexed code that the model writes but never reads.

4.3 CODI: What Separator Latents Do

The logit lens on CODI's six latents reveals a clear pattern: odd positions (z_1, z_3, z_5) decode to numbers, even positions (z_2, z_4, z_6) decode to pause-like punctuation. We ask whether the even latents compute anything useful or merely hold structural placeholders.

Table 2 answers this. Removing separator positions drops accuracy by 10.6 pp, but swapping their content across examples leaves accuracy unchanged, a double dissociation: positions matter, content does not. To test whether the separator forward passes provide computation that cannot be absorbed by fewer latents, we train a 3-latent student: each of its three latents is supervised via SmoothL1 to match the hidden state of the corresponding number latent of a frozen 6-latent teacher ($z_1, z_3, z_5 \to s_1, s_2, s_3$). The student reaches 41.85%, matching the full 6-latent model with no separator positions at all. This shows the $\sim$10 pp positional drop when removing separators is a distribution-mismatch artefact, not lost computation. The per-layer lens (Appendix Figure 9) shows separators transiently relay a numerical value in early layers before collapsing to $>>$ at layers 6 to 9, consistent with register-like filler slots (Pfau et al., 2024).

The conclusion is that pause tokens in CODI are not reasoning steps: they occupy forward-pass slots the model was trained to expect, and removing them cold hurts accuracy, but a student model taught to skip them entirely matches the teacher. The content they carry is data-format noise, not deliberate intermediate computation.

Finally, Table 8 traces where the number/separator pattern originates. GSM8K-Aug CoT reads $r_1 >> r_2 >> \ldots$ and CODI supervises only the final answer token, yet z_{2k-1} decodes to step result r_k at 55%/29%/13% top-1. The alternation is a transcription of the data's surface delimiter format, not an emergent reasoning structure.

Configuration	Acc (%)	Δ
full 6-latent (released)	41.8	
numbers only (z_1, z_3, z_5)	31.2	-10.6
content swap (separators)	$\approx$41.8	$\approx$0
no-distill 3-latent	$\sim$37.0	-4.8
step-distill 3-latent	**41.85**	$+0.05$

Table 2: CODI ablation on GSM8K. Separator positions are worth $\sim$10 pp but their content is fungible; a step-distilled 3-latent student matches the 6-latent model.

5 Ethical considerations

Multilingual adaptation. Our methods are language-agnostic: high-resource languages such as French or German need only translated CoT data, while low-resource languages such as Urdu or Swahili are harder: data is scarce and, since latents transcribe surface format, any format bias is inherited directly.

Benefits and harms. The work mainly benefits researchers building interpretable, efficient reasoning systems, but the probing tools are dual-use, and any such model can produce confidently wrong answers that mislead overtrusting users.

Mitigation. We release our analysis openly and frame it conservatively, so it reduces rather than inflates claims that these latents reason; we recommend auditing the reasoning corpus and validating per task before deployment.

Vulnerable groups. Our evaluation is English-centric and GSM8K-specific, so the reported structure reflects English arithmetic conventions and could disadvantage speakers of under-represented languages; multilingual data and per-language evaluation would mitigate this.

6 Conclusion

Three complementary probes give one consistent answer: latent thoughts are a thin channel whose apparent structure systematically overstates the computation it carries. First, discretising Coconut on GPT-2 shows that a jointly trained codebook collapses to pause-like codes the model ignores, while decoupled quantisers (VQ-PTR, FSQ) carry genuine signal (Δ_{info} up to $+13.3$ pp) whose codes nonetheless mirror GSM8K formatting. Second, on frozen Qwen3 backbones up to 4B the latents are position-specialised, low-dimensional, and faithfully discretisable, yet $\Delta_{\text{info}}{=}0$: zeroing or randomising the codes changes not a single answer, so the model writes this structure but never reads it. Third, CODI's separator latents are content-free placeholders: a 3-latent student that skips them matches the full teacher, so their alternation transcribes the data's delimiter format rather than emergent reasoning. These results caution against attributing reasoning to latent thoughts on the basis of structure alone and favour leaner latent-CoT designs.

References

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *Preprint*, arXiv:1308.3432.

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. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

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

Leland McInnes, John Healy, and James Melville. 2018. UMAP: Uniform manifold approximation and projection for dimension reduction. *Preprint*, arXiv:1802.03426.

Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. 2024. Finite scalar quantization: VQ-VAE made simple. In *The Twelfth International Conference on Learning Representations (ICLR)*.

nostalgebraist. 2020. interpreting GPT: the logit lens. LessWrong.

Jacob Pfau, William Merrill, and Samuel R. Bowman. 2024. Let's think dot by dot: Hidden computation in transformer language models. In *First Conference on Language Modeling (COLM)*.

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

Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. 2017. Neural discrete representation learning. In *Advances in Neural Information Processing Systems 30 (NeurIPS)*.

A Appendix

A.1 Coconut discretisation: positional roles and training dynamics

pos	# marker	number	word	op
0	435	65	0	0
1	6	**494**	0	0
2	4	7	**485**	4
3	**500**	0	0	0
4	34	**465**	0	1
5	483	13	3	1

Table 3: Logit-lens top-1 token category per latent position (counts over 500 validation examples). Each position specialises into one role, forming an implicit `# NUM WORD # NUM #` template.

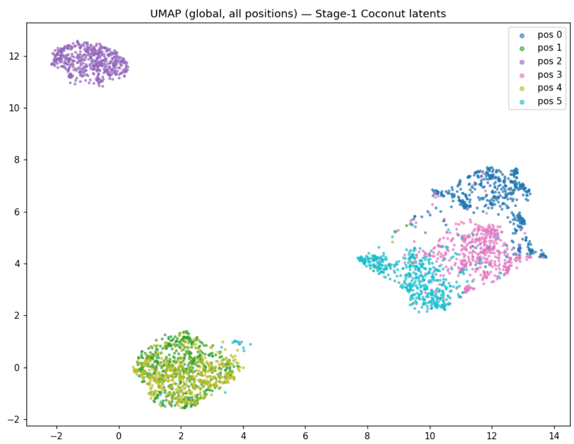

Figure 1: UMAP of the raw 768-dimensional latents coloured by position. Three clusters match the logit-lens roles in Table 3: connectors (pos 2), numbers (pos 1, 4), delimiters (pos 0, 3, 5), confirming the partition without the LM head.

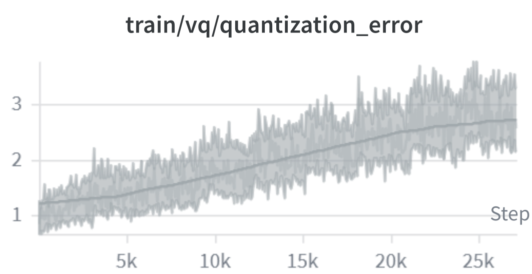

Figure 2: Quantisation error during training for VQ-JOINT, the error increases as the model collapses

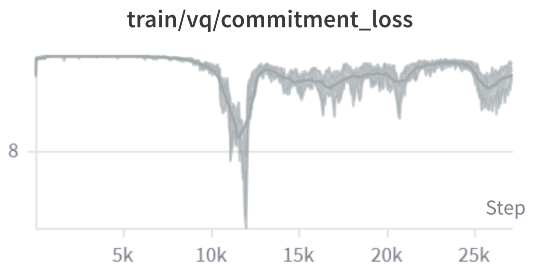

Figure 3: Commitment loss during training for VQ-JOINT, consistent with the encoder rank collapse described in Section 4.1.

A.2 Qwen3 scaling: detailed tables and figures

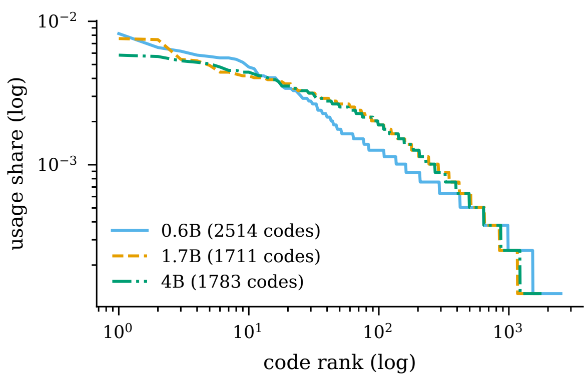

Figure 5: No codebook collapse at any scale: code usage follows a heavy-tailed rank-frequency curve with thousands of active codes (legend), in contrast to the single active code of the jointly trained GPT-2 codebook (Table 1).

Stage		**0.6B**	**1.7B**	**4B**
Stage 0 (0 latents)	val	**.582**	**.690**	**.786**
	test	**.538**	**.635**	**.736**
Stage 1 (2 latents)	val	.488	.566	.742
	test	.492	.541	.697
Stage 2 (4 latents)	val	.344	.468	.624
	test	.340	.436	.588
Stage 3 (6 latents)	val	.252	.320	.498
	test	.239	.302	.460

Table 4: GSM8K accuracy per Coconut curriculum stage on Qwen3. Bold marks the best stage per model.

Model	**p0**	**p1**	**p2**	**p3**	**p4**	**p5**	**Excl.**	**Jaccard (adj.)**
0.6B	551	**57**	457	428	575	637	93%	0.0, 0.0, 0.01, 0.07, 0.10
1.7B	666	386	243	371	364	299	77%	0.0, 0.0, 0.31, 0.43, 0.49
4B	156	110	522	265	468	309	97%	0.0, 0.0, 0.0, 0.0, 0.0

Table 5: Distinct VQ codes per latent slot (test split). Excl. is the share of codes exclusive to a single slot; the Jaccard column lists the five adjacent pairs p0p1 through p4p5. Early slots share no codes and all 4B adjacent pairs are disjoint; Figure 6 shows all pairs.

Model	**p0**	**p1**	**p2**	**p3**	**p4**	**p5**
0.6B	2	5	1	1	1	1
1.7B	**97**	7	7	7	7	8
4B	0	2	0	1	0	1

Table 6: Logit-lens number-token mass (%) per latent slot, probability-weighted over the VQ codes. Only 1.7B shows a dedicated number slot (position 0); the full category composition is in Figure 8.

Model	**Dict.**	**Var. expl.**	**Dead**	**Active feat.**
0.6B	8192	95.1%	0	6193
1.7B	16384	89.9%	0	6348
4B	20480	99.5%	0	**180**

Table 7: TopK sparse autoencoder (k=32) on Qwen3 latents: dictionary size, variance explained on the test split, features that never fire during training (Dead), and features firing on the test split (Active). Bold marks the 4B collapse to 180 features, reflecting its low intrinsic dimensionality (Figure 4).

A.3 CODI: supporting evidence

Latent	**Step**	**Top-1**	**Top-5**	N
z_1	r_1	55%	65%	492
z_3	r_2	29%	37%	468
z_5	r_3	13%	21%	312
sep. latents	>>	20%	n/a	1500

Table 8: CoT alignment (N=500). z_{2k-1} top-1/top-5 hit rate for step result r_k. Separators decode to >> in only 20% of cases.

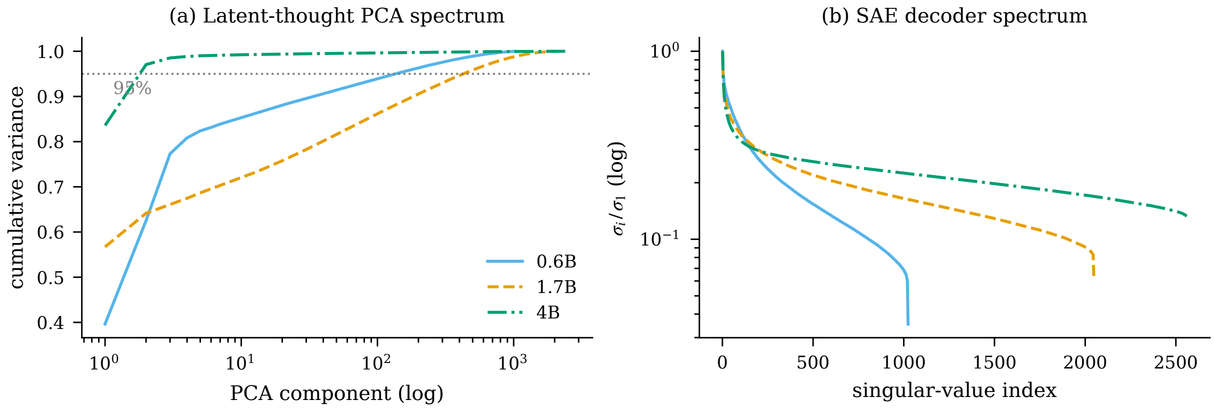

Figure 4: Latent dimensionality collapses with scale, measured by two independent lenses. (a) Cumulative PCA variance of the latent thoughts: 4B reaches 95% with 2 components, against 136 for 0.6B and 426 for 1.7B. (b) SAE decoder singular-value spectra show the same rank ordering. This low intrinsic dimensionality explains the 180 active 4B features (Table 7) and is consistent with discretisation costing no accuracy.

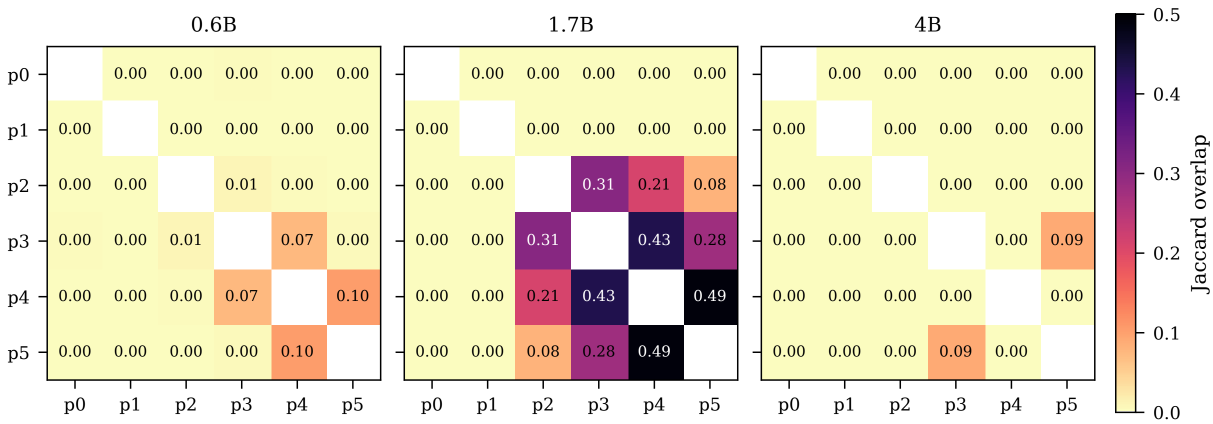

Figure 6: Latent positions use disjoint code vocabularies. Jaccard overlap between the code sets of every pair of latent slots (test split): early positions share no codes on any model, 1.7B mixes content codes across late slots, and 4B is disjoint at every pair except a small p3 to p5 overlap of 0.09, matching Table 5.

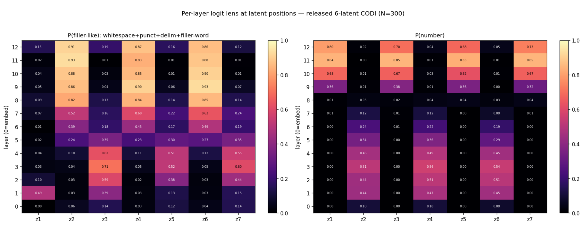

Figure 9: Per-layer logit-lens heatmap (6-latent CODI). Separator latents (z_2, z_4, z_6) compute numerical tokens at early layers and collapse to >>/comma at layers 6 to 9; number latents resolve at layers 10 to 12.

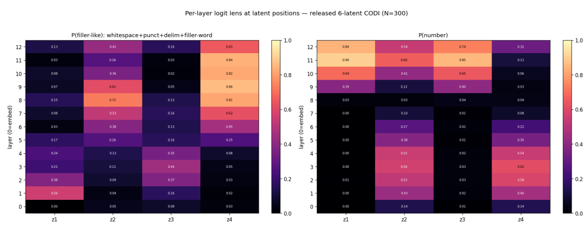

Figure 10: Per-layer logit-lens heatmap of our step-distilled 3-latent student. All three latents resolve to numerical tokens with no separator slots, confirming the separators' computation is absorbed.

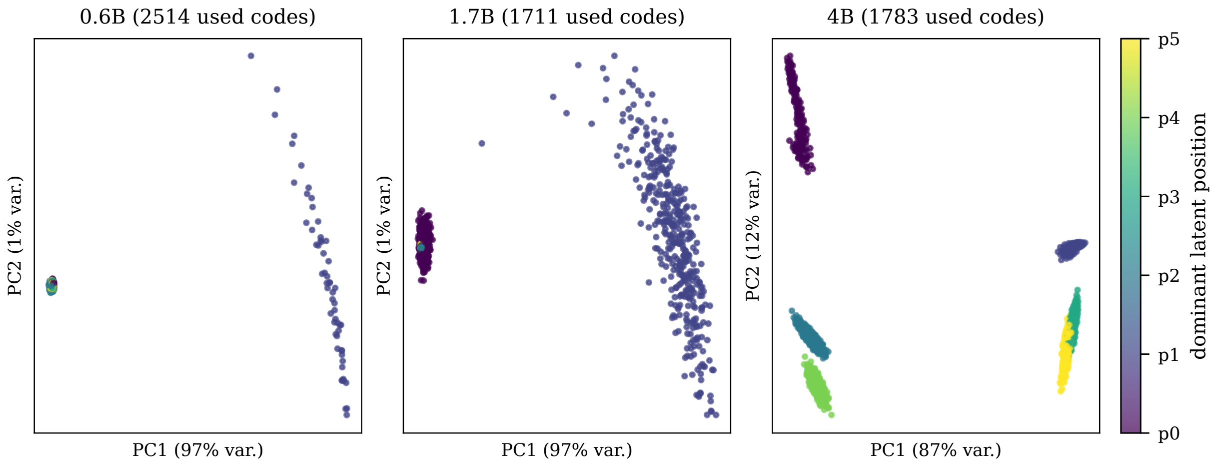

Figure 7: Two-component PCA of the VQ codebook, colouring each used code by the latent position that uses it most; axes report each component's explained variance. On 0.6B and 1.7B the first component alone carries 97% of codebook variance and separates the position-0 cluster from the rest, with finer position structure lying in higher dimensions. On 4B the six position clusters separate completely in two components (87% and 12%), the geometric counterpart of its fully disjoint code vocabularies (Figure 6).

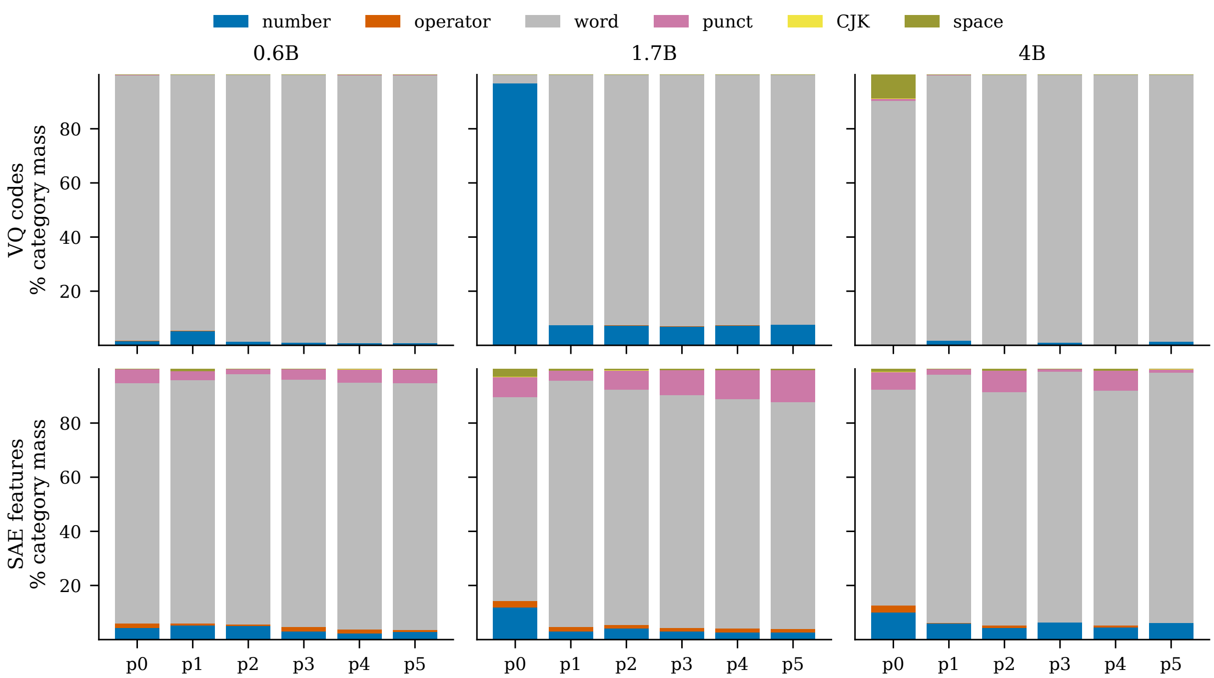

Figure 8: Logit-lens token-category composition per latent slot (probability-weighted), for VQ codes (top) and SAE features (bottom). The 1.7B number slot at position 0 stands out (97% number mass); elsewhere mass concentrates on abstract word-like fragments, with punctuation accruing toward later slots in the SAE view. VQ and SAE agree on the structure.

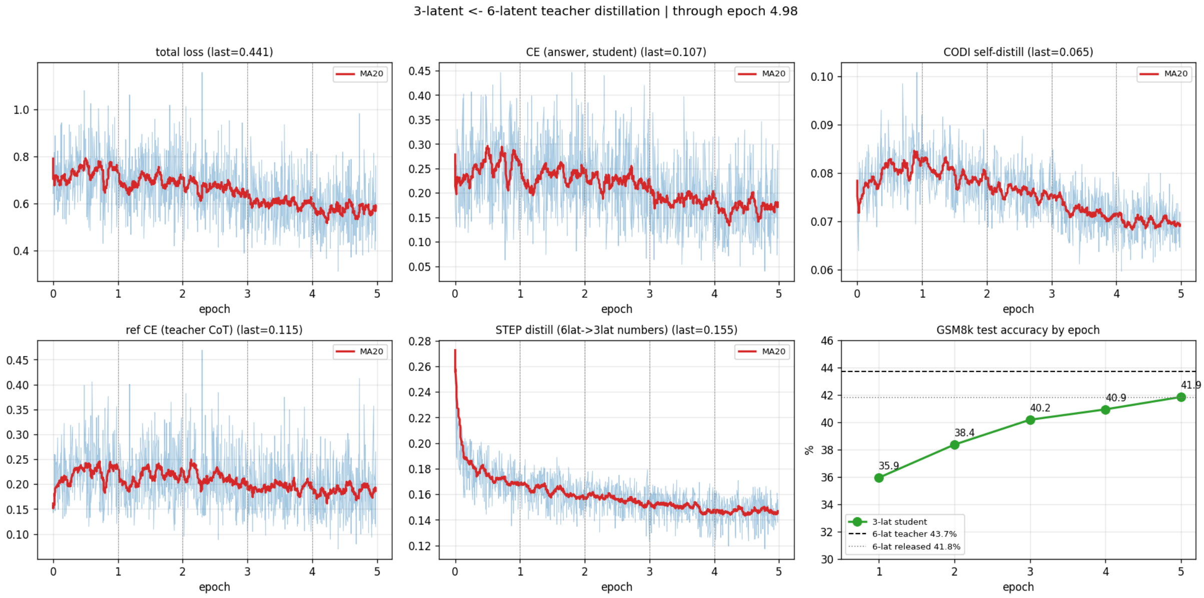

Figure 11: Training curves of the 6-to-3 latent step distillation. The student converges to teacher-level accuracy (41.85%, Table 2) within the distillation budget.