Section 9.2: Agent Model with Latent Space

2026 Spring

LLM Agents Foundation & Applications

Student Team / 20260421

Roadmap

- S1: LLM Basic Alignment
- S2: LLM Alignments for Reasoning
- S3: Agent applications
- S4: LLM Data synthesis
- S5: Agent Memory
- S6: LLM model serving
- S7: Agent Evaluation and Attack/Defense Landscape
- S8: Agent Planning / Testing Time Scaling
- S9: World Modeling for GenAI Agents ➔ This lecture!
- S10: Multi-Agents

The Latent Space

Foundation, Evolution, Mechanism, Ability, and Outlook —
on how continuous hidden states are becoming the native substrate of language-based intelligence.

A Comprehensive Survey of Latent Reasoning in Large Language Models

arXiv:2604.02029v1

Basics of latent space in LLM

THE FORMAL DEFINITION

What the paper means by *latent space*.

Definition.
The continuous hidden-state manifold $\mathscr{H}$ inside a language model, where contextual, semantic, syntactic, and cross-modal features are jointly encoded as high-dimensional vectors.

Formally,

Standard LLM: $y \sim \Phi_\theta(\cdot \mid x)$

$x \in \mathcal{V}$ — *generation conditioned only on input tokens*

Latent-space LLM: $y \sim \Phi_\theta(\cdot \mid x, z), \quad z \in \mathcal{H}$

generation conditioned on tokens AND a continuous latent variable z

- A sequence of z_i vectors replaces (or augments) the usual token sequence.
  - Each z_i is d-dimensional (typically d = 2048–4096)
  - z_i can encode a superposition of candidate reasoning paths.
- The latent variable z provides an additional channel for encoding information that is costly or impossible to express in tokens
  - global semantics
  - multimodal features
  - intermediate reasoning states
  - structural constraints.

WHERE THE INFORMATION GOES

A standard LLM inference lose a lot of information

STANDARD LLM STEP

$$h_t \rightarrow W_U \cdot h_t = \text{logits} \in \mathbb{R}^{32000} \rightarrow \text{argmax} \rightarrow \text{token} \rightarrow W_E \rightarrow h_{t+1}$$

BEFORE (h_t)

A 4,096-dim continuous vector. Can encode multiple candidate continuations simultaneously via linear subspaces.

MIDDLE (information loss)

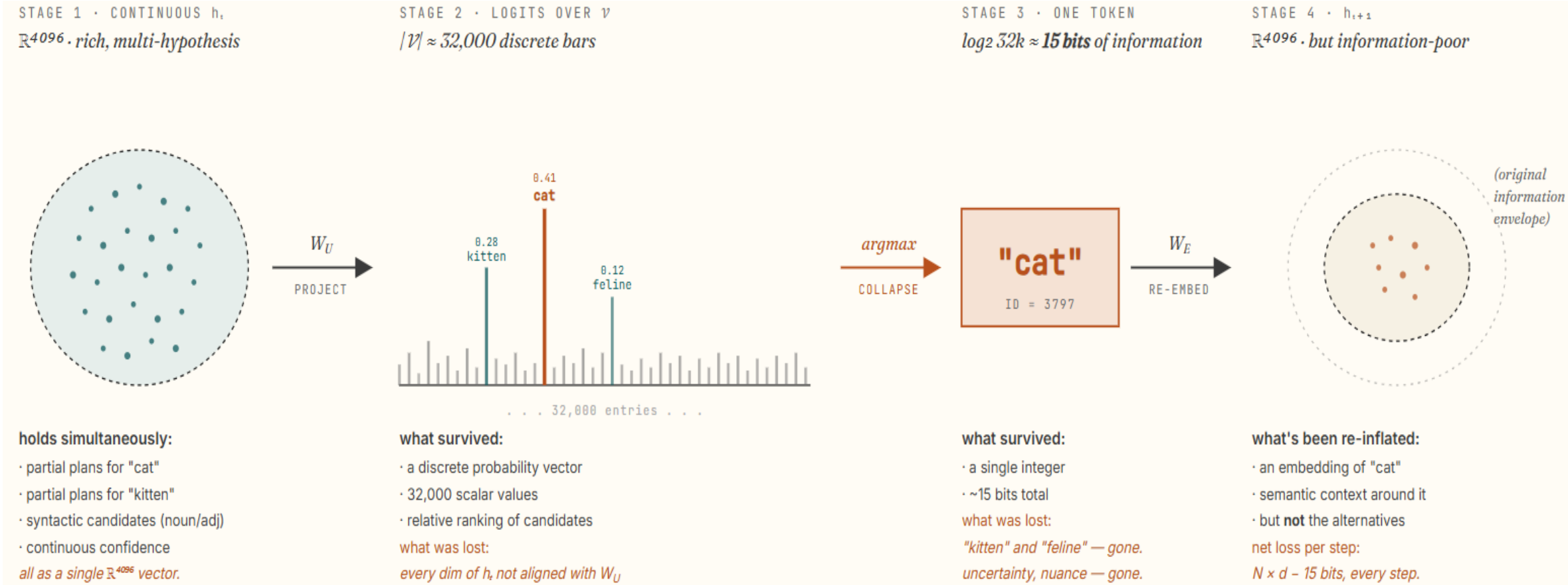
Collapsed to one of 32,000 discrete symbols. $\log_2(32k) \approx 15$ bits of information survive. Everything else is gone.

AFTER (h_{t+1})

Re-inflated from $W_E \cdot$ token. Same dimensionality, but now carrying only the information you could squeeze through 15 bits.

The token bottleneck, drawn out.

Every time a standard LLM takes one "prediction step," it runs a continuous → discrete → continuous round-trip through its vocabulary.



holds simultaneously:

· partial plans for "cat"
· partial plans for "kitten"
· syntactic candidates (noun/adj)
· continuous confidence

all as a single $\mathbb{R}^{4096}$ vector.

what survived:

· a discrete probability vector
· 32,000 scalar values
· relative ranking of candidates

what was lost:

every dim of h_t not aligned with W_U

what survived:

· a single integer
· ~15 bits total

what was lost:

"kitten" and "feline" — gone.
uncertainty, nuance — gone.

what's been re-inflated:

· an embedding of "cat"
· semantic context around it
· but **not** the alternatives

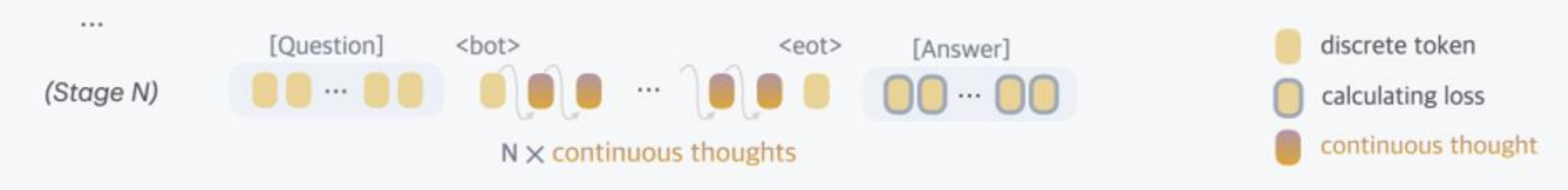
net loss per step:

N × d − 15 bits, every step.

Foundational Work of Latent Space- COCONUT

Two Modes

- In the proposed COCONUT method, the LLM switches between the “language mode” and “latent mode”.
- In language mode, the model operates as a standard language model, autoregressively generating the next token.
- In latent mode, it directly utilizes the last hidden state as the next input embedding. This last hidden state represents the current reasoning state, termed as a “continuous thought”.



Foundational work - COCONUT (Dec 2024).

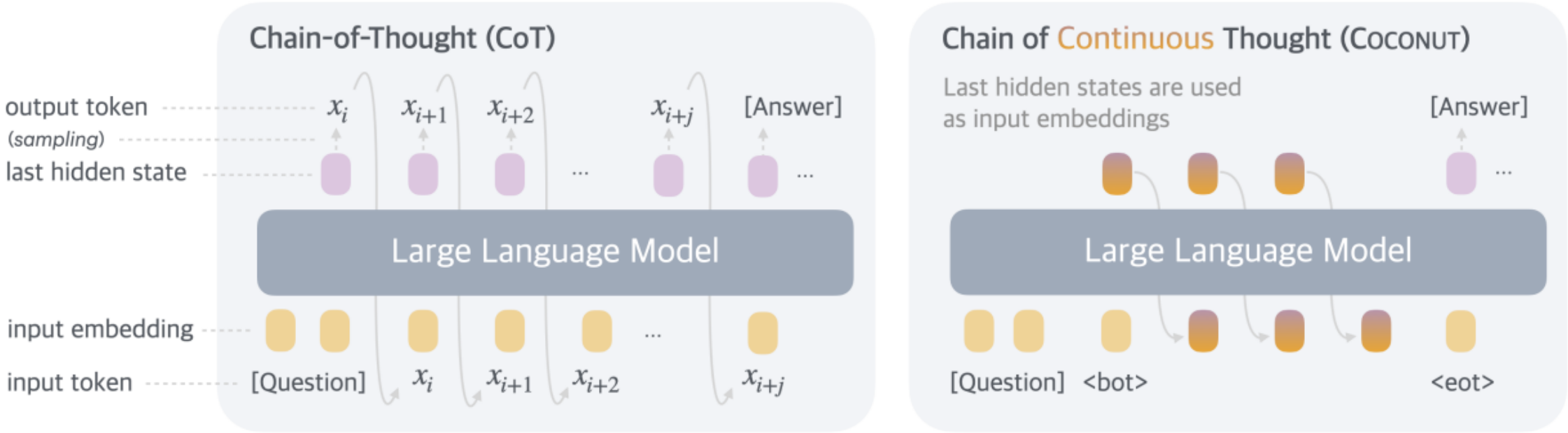
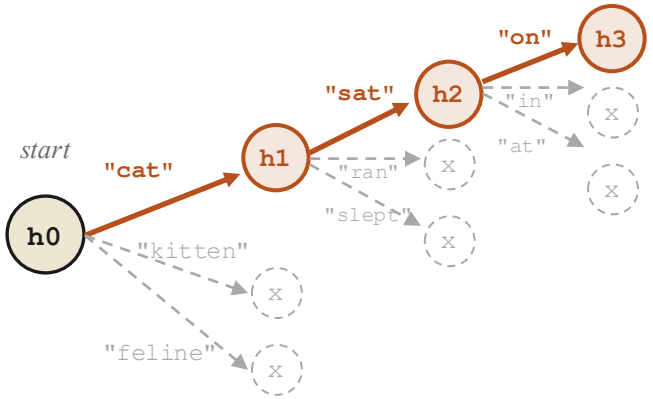


Figure 1 A comparison of Chain of Continuous Thought (COCONUT) with Chain-of-Thought (CoT). In CoT, the model generates the reasoning process as a word token sequence (e.g., $[x_i, x_{i+1}, ..., x_{i+j}]$ in the figure). COCONUT regards the last hidden state as a representation of the reasoning state (termed "continuous thought"), and directly uses it as the next input embedding. This allows the LLM to reason in an unrestricted latent space instead of a language space.

A single continuous hidden state can carry several candidate next-thoughts at once. A single token cannot.

=

Explicit: one path at a time



6 branches killed
after 3 tokens.

COST STRUCTURE

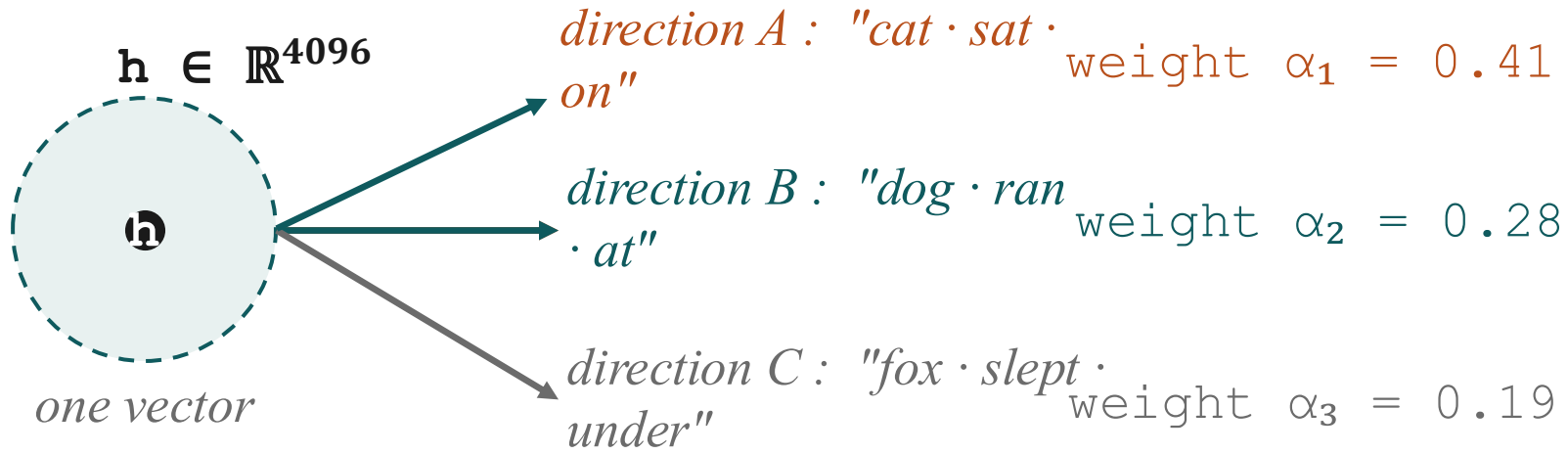
To keep k branches alive for n steps, you need *k forward passes*. Beam search is *linear in k* in compute, memory, and wall-clock.

Because each token commits to one discrete symbol, divergent branches cannot share a representation — they must be separate trajectories.

This is the geometric reason COCONUT can do breadth-first search in one forward pass, while explicit CoT can only do greedy depth-first.

Latent reasoning: all paths at once

ONE h · MULTIPLE BRANCHES ALIVE SIMULTANEOUSLY



DECOMPOSITION

$$h = \alpha_1 \cdot v_A + \alpha_2 \cdot v_B + \alpha_3 \cdot v_C + \text{noise}$$

The same h carries A, B, and C because v_A, v_B, v_C *are nearly orthogonal in* $\mathbb{R}^{4096}$*. Three probes read three answers from the same h.*

Training Procedure - COCONUT (Dec 2024).

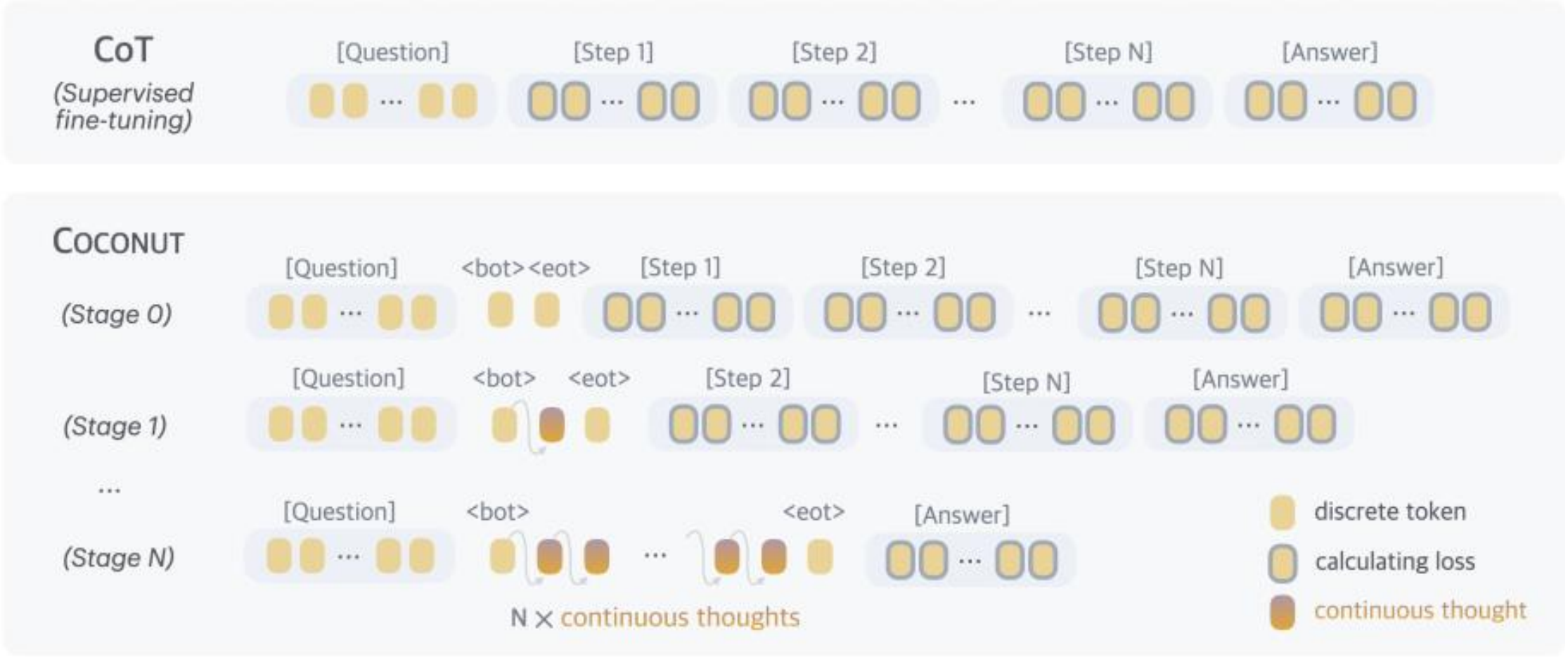
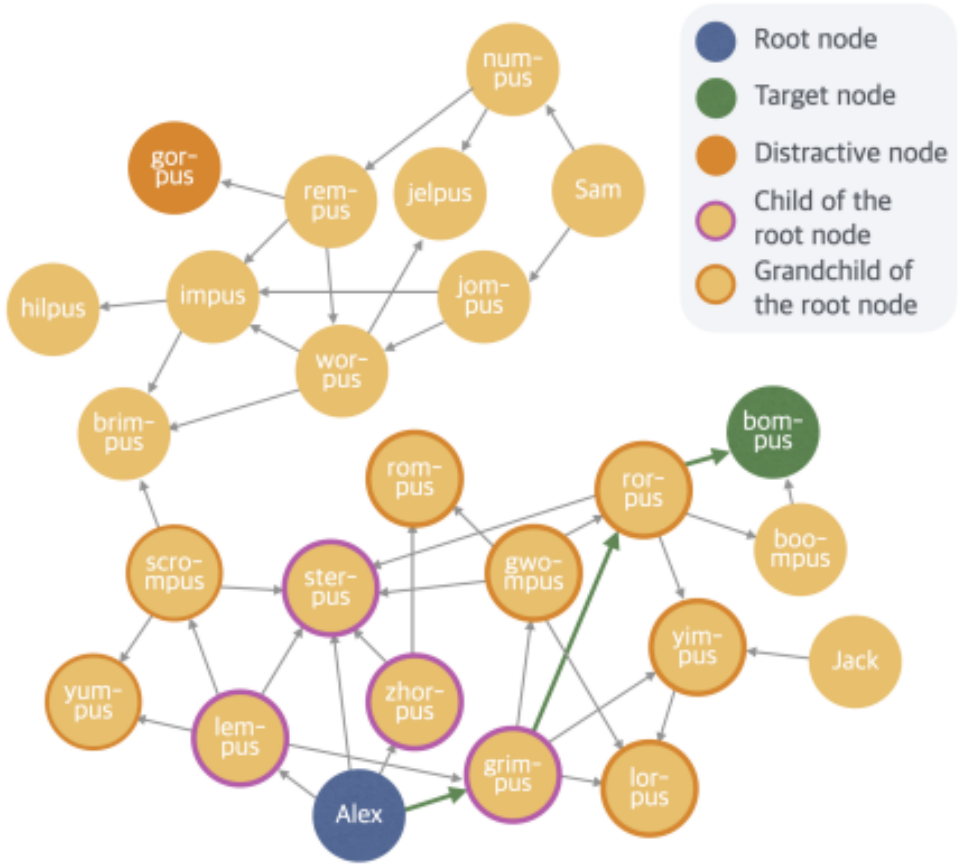


Figure 2 Training procedure of Chain of Continuous Thought (COCONUT). Given training data with language reasoning steps, at each training stage we integrate c additional continuous thoughts ($c = 1$ in this example), and remove one language reasoning step. The cross-entropy loss is then used on the remaining tokens after continuous thoughts.

Case Study- COCONUT (Dec 2024).



Question:

Every grimpus is a yimpus. Every worpus is a jelpus. Every zhorpus is a sterpus. Alex is a grimpus ⋯ Every lumps is a yumpus.
Question: **Is Alex a gorpus or bompus?**

Ground Truth Solution

Alex is a grimpus.
Every grimpus is a rorpus.
Every rorpus is a bompus.
Alex is a bompus

CoT

Alex is a lempus.
Every lempus is a scrompus.
Every scrompus is a yumpus.
Every yumpus is a rempus.
Every rempus is a gorpus.
Alex is a gorpus ✗

(Hallucination)

COCONUT (k=1)

<bot> <eot>
Every lempus is a scrompus.
Every scrompus is a brimpus.
Alex is a brimpus ✗

(Wrong Target)

COCONUT (k=2)

<bot> <eot>
Every rorpus is a bompus.
Alex is a bompus ✓

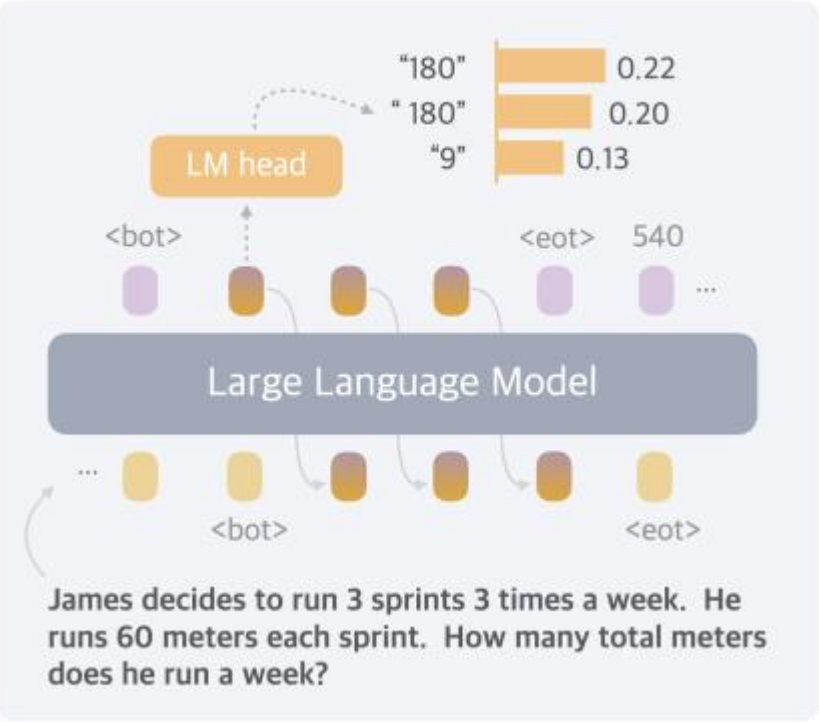
(Correct Path)

Figure 4 A case study of ProsQA. The model trained with *CoT* hallucinates an edge (*Every yumpus is a rempus*) after getting stuck in a dead end. COCONUT (k=1) outputs a path that ends with an irrelevant node. COCONUT (k=2) solves the problem correctly.

Experiment Results - COCONUT (Dec 2024).

Method	GSM8k		ProntoQA		ProsQA	
	Acc. (%)	# Tokens	Acc. (%)	# Tokens	Acc. (%)	# Tokens
CoT	42.9 ±0.2	25.0	98.8 ±0.8	92.5	77.5 ±1.9	49.4
No-CoT	16.5 ±0.5	2.2	93.8 ±0.7	3.0	76.7 ±1.0	8.2
iCoT	30.0*	2.2	99.8 ±0.3	3.0	98.2 ±0.3	8.2
Pause Token	16.4 ±1.8	2.2	77.7 ±21.0	3.0	75.9 ±0.7	8.2
COCONUT (Ours)	34.1 ±1.5	8.2	99.8 ±0.2	9.0	97.0 ±0.3	14.2
- *w/o curriculum*	14.4 ±0.8	8.2	52.4 ±0.4	9.0	76.1 ±0.2	14.2
- *w/o thought*	21.6 ±0.5	2.3	99.9 ±0.1	3.0	95.5 ±1.1	8.2
- *pause as thought*	24.1 ±0.7	2.2	100.0 ±0.1	3.0	96.6 ±0.8	8.2

Results on three datasets: GSM8k, ProntoQA and ProsQA. Higher accuracy indicates stronger reasoning ability, while generating fewer tokens indicates better efficiency.



Decoding a continuous thought into language tokens in a math word problem. The decoded tokens correspond to intermediate variables that help solve the problem.

THE METHOD THAT STARTED IT ALL

COCONUT (Dec 2024).

The trick

A standard LLM takes the last hidden state h_t, projects to vocab logits, samples a token, re-embeds it, and feeds it back.

COCONUT SKIPS the middle steps. The last hidden state is fed directly back as the next input embedding — no projection, no sampling, no re-embedding.

STANDARD LLM LOOP

$h_t \rightarrow$ logits $\rightarrow$ argmax $\rightarrow$ token $\rightarrow$ embed $\rightarrow h_{t+1}$

COCONUT LOOP

$h_t \rightarrow h_{t+1}$ (the hidden state IS the embedding)

The key finding

Continuous thoughts can encode a **SUPERPOSITION** of multiple next steps simultaneously. Because h is a continuous vector, not a discrete token, it can hold several candidate paths at once and let later computation disambiguate.

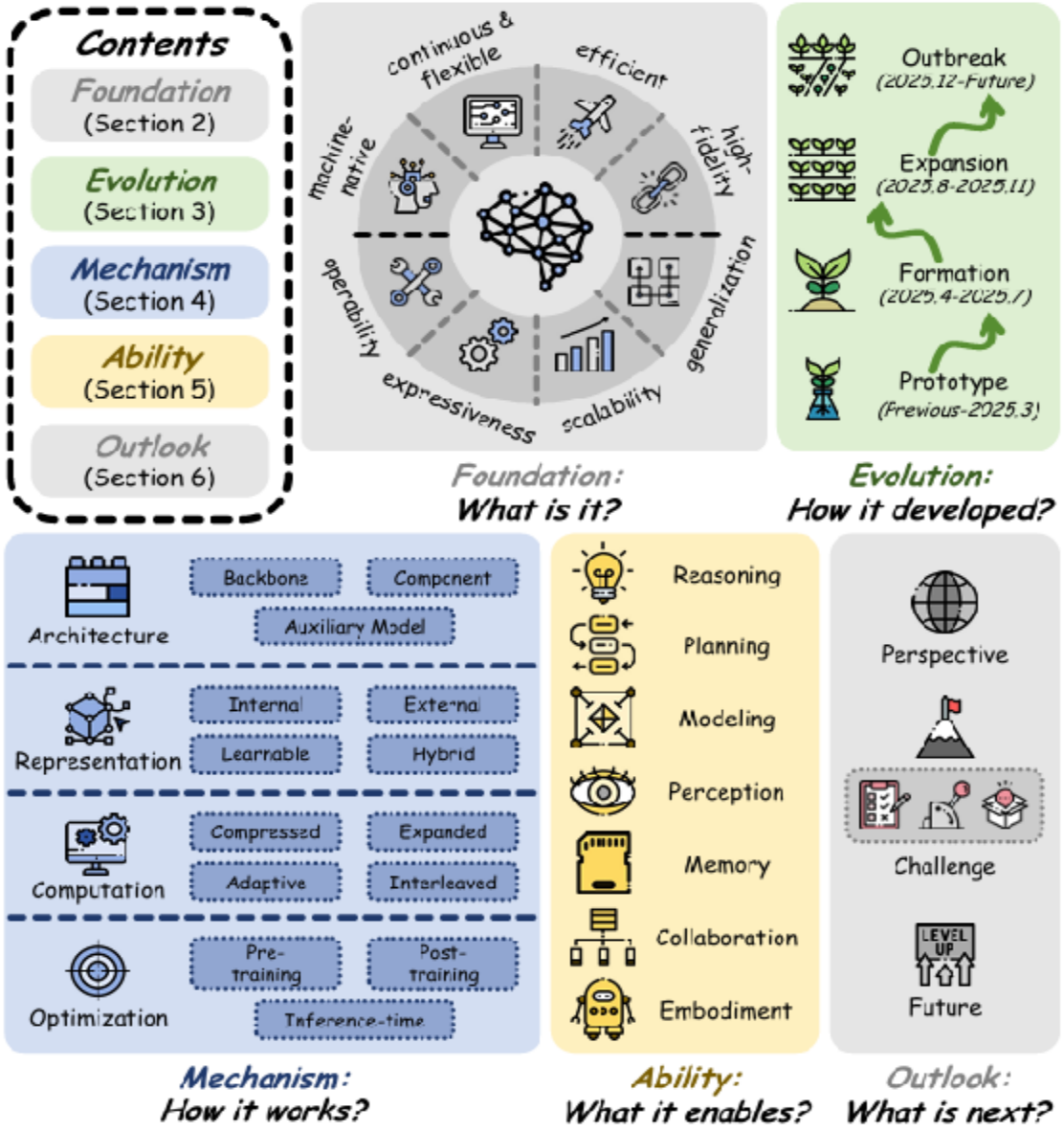
On ProntoQA and ProsQA, COCONUT effectively performs breadth-first search in latent space while standard CoT can only do greedy depth-first — and it does so with FEWER tokens.

WHY THIS WAS A BIG DEAL

Before COCONUT, everyone assumed reasoning HAD to be verbalized. COCONUT showed (a) LLMs can reason without text, (b) removing the tokenization constraint actually IMPROVES on search-heavy tasks.

Connecting latent space to LLM

Outline



- Foundation: What is Latent Space? (Section 2),
- Evolution: How Did Latent Space Develop? (Section 3),
- Mechanism: How Does Latent Space Work? (Section 4),
- Ability: What Does Latent Space Enable? (Section 5),
- Outlook: What is Next? (Section 6)

FOUR WORDS THAT DO ALL THE WORK

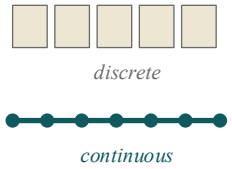
What makes latent space latent?

The previous definition had four adjectives: continuous, high-dimensional, jointly encoded, manifold. Each one is load-bearing — removing any of them breaks a different capability the paper depends on.

1. Continuous

vs discrete tokens

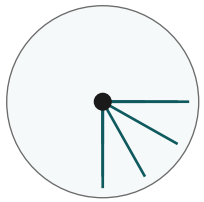
h ∈ ℝ^d supports addition, scaling, interpolation, gradients. You can take $\alpha \cdot \mathbf{h_A} + \beta \cdot \mathbf{h_B}$ and get a meaningful blend — impossible with tokens. This is what lets RL, steering, and gradient descent operate on thoughts directly.



2. High-dimensional

d = 2048 – 8192

In $\mathbb{R}^{4096}$ you can fit exponentially many near-orthogonal directions (Johnson–Lindenstrauss). That's the geometric room in which superposition lives — many features coexist in one vector with low mutual interference.

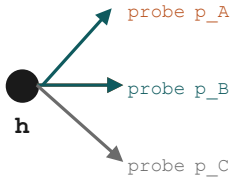


12 near-orthogonal directions

3. Jointly encoded

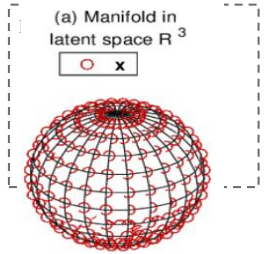
many features, one vector

The SAME h simultaneously carries token identity, syntactic role, sentiment, discourse reference, even cross-modal alignment. Not layered — superposed. Different linear probes read different features from the same state.



4. Manifold

Only a thin structured subset of $\mathbb{R}^{4096}$ corresponds to valid language states — training carves out $\mathcal{H} \subset \mathbb{R}^{D}$. Random vectors in $\mathbb{R}^{4096}$ are mostly garbage to the model; meaningful h lies on the learned submanifold.



A QUESTION THAT HAS TWO ANSWERS

Is ChatGPT or Claude latent-conditioned?

no, they are "conditioned on tokens" in the sense this paper means. But the word "latent" is used at two different levels — and the distinction matters for what counts as a latent-reasoning model.

THREE PRODUCTION-MODEL FEATURES THAT LOOK LATENT BUT ARE NOT

Extended thinking — *still tokens*

`o1 · o3 · R1 · Claude thinking`

The 'thinking' is a hidden token stream between <think> tags. Each step still goes through argmax/sampling. Architecturally identical to standard autoregressive generation — just with more tokens.

KV cache — *still tokens*

`every Transformer since 2017`

KV cache stores continuous key/value tensors across generation, but only WITHIN a single forward pass's context. The inter-step interface is still the emitted token that gets re-embedded. Cache is a speedup, not a new mode.

Multimodal inputs — *latent INPUT only*

`GPT-4o · Claude · Gemini`

Images enter as continuous vision-encoder embeddings — genuinely latent on the INPUT side. But the output is still strictly token-by-token. The z in $\Phi(\cdot|x,z)$ refers to latents PRODUCED DURING REASONING, not input-side embeddings.

PROPERTY BY PROPERTY

The differences, in detail.

Human-readable vs machine-native	Explicit states are natural-language tokens from vocab $\mathcal{V}$ — readable. Latent states are $\mathbb{R}$^d vectors (d=1024–8192); dimensions don't map to concepts, but the model processes them natively without encode/decode overhead.
Discrete & symbolic vs continuous & flexible	Tokens are discrete symbols grounded in grammar. Latent states form smooth differentiable manifolds — supporting linear combinations, gradient optimization, and interpolation. This is what enables RL, steering, and gradient descent ON thoughts.
Inefficient vs efficient	Explicit CoT stacks three overheads: linguistic redundancy, hidden→vocab→hidden round-trip, strictly sequential decoding. Latent sidesteps all three. Recurrent-depth or parallel-width can do many 'steps' per forward pass.
Semantically lossy vs high-fidelity	Projecting rich internal state onto $\|\mathcal{V}\|$≈32,000 symbols is quantization. Uncertainty, fine-grained gradation, cross-modal alignment flatten. Latent preserves continuous confidence, multimodal alignment, spatial geometry.
Evaluable vs opaque	The trade-off. Explicit is auditable — every step is readable. Latent gains efficiency and expressiveness but loses interpretability. §6 of the paper takes this up directly as the central open problem.

Three geometric consequences that drive the whole summary.

1 · Superposition is native	A single h can encode a mixture of several candidate "**next thoughts**" simultaneously — the cloud is dense enough to carry multiple probes at once. In the VAE, each z decodes to exactly one image. In the LLM, each h is interrogated by many linear probes, each reading a different feature. This is why *latent reasoning can branch without beam search*.
2 · No "valid latent" constraint	You can **inject arbitrary hidden states** — from external tools, memory modules, other models — and the LLM will try to continue from them. A VAE decoder requires z to lie on the manifold or reconstructions degenerate. An LLM has no such prior; it treats off-distribution *h* as input to continue processing.
3 · Asymmetric decode	The one-way nature of W_U is what makes latent reasoning interesting *and* hard. Interesting: computation can stay in $\mathbb{R}^{4096}$ for many steps before anyone pays the verbalization tax. Hard: there is no inverse — you can't "decode" the latent reasoning into a faithful text explanation the way you can decode a VAE z into a pixel image. This is §6's interpretability problem in a single geometric fact.
Takeaway	When the paper says "latent space," it does *not* mean a low-dim reconstructable manifold. It means the full **ambient $\mathbb{R}^d$ residual stream** and everything you can do inside it — with all the freedom and all the opacity that geometry implies.

From proof of concept to outbreak.

Monthly publication volume went from ~50 prototypes in late 2024 to 400+ papers per month by early 2026 — and the curve is still climbing.

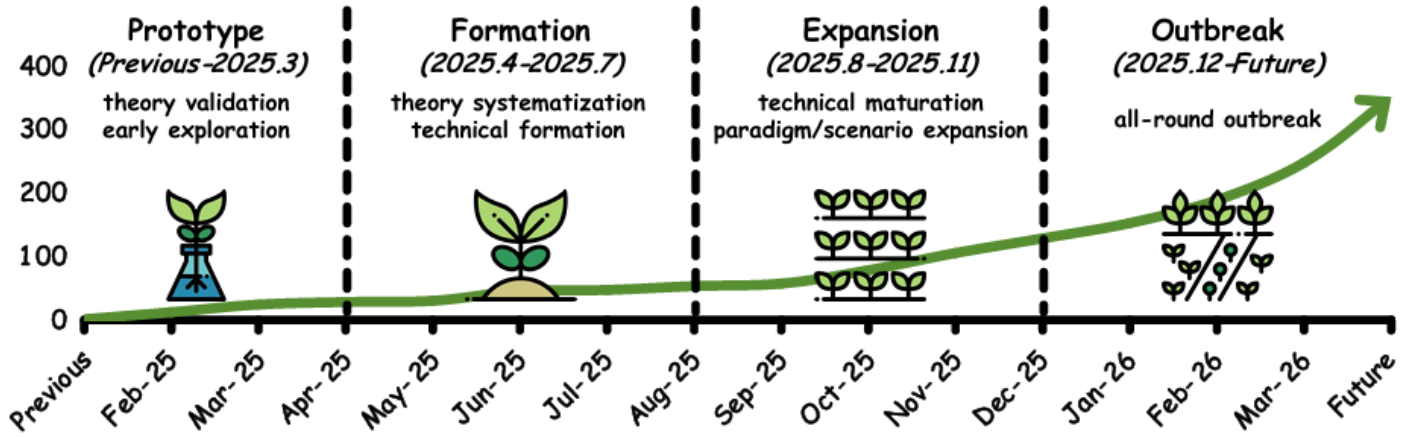


Figure 4 Timeline of representative works in the evolution of latent space research, organized into four developmental stages: **Prototype** (Section 3.1), **Formation** (Section 3.2), **Expansion** (Section 3.3), and **Outbreak** (Section 3.4) stages, where the horizontal axis denotes the month, and vertical axis indicates the number of the latent-level works.

Before 2025.3

- HCoT(Hierarchical Chain-of-Thought)
- COCONUT
- CCoT(Contemplation tokens)

2025.4-2025.7

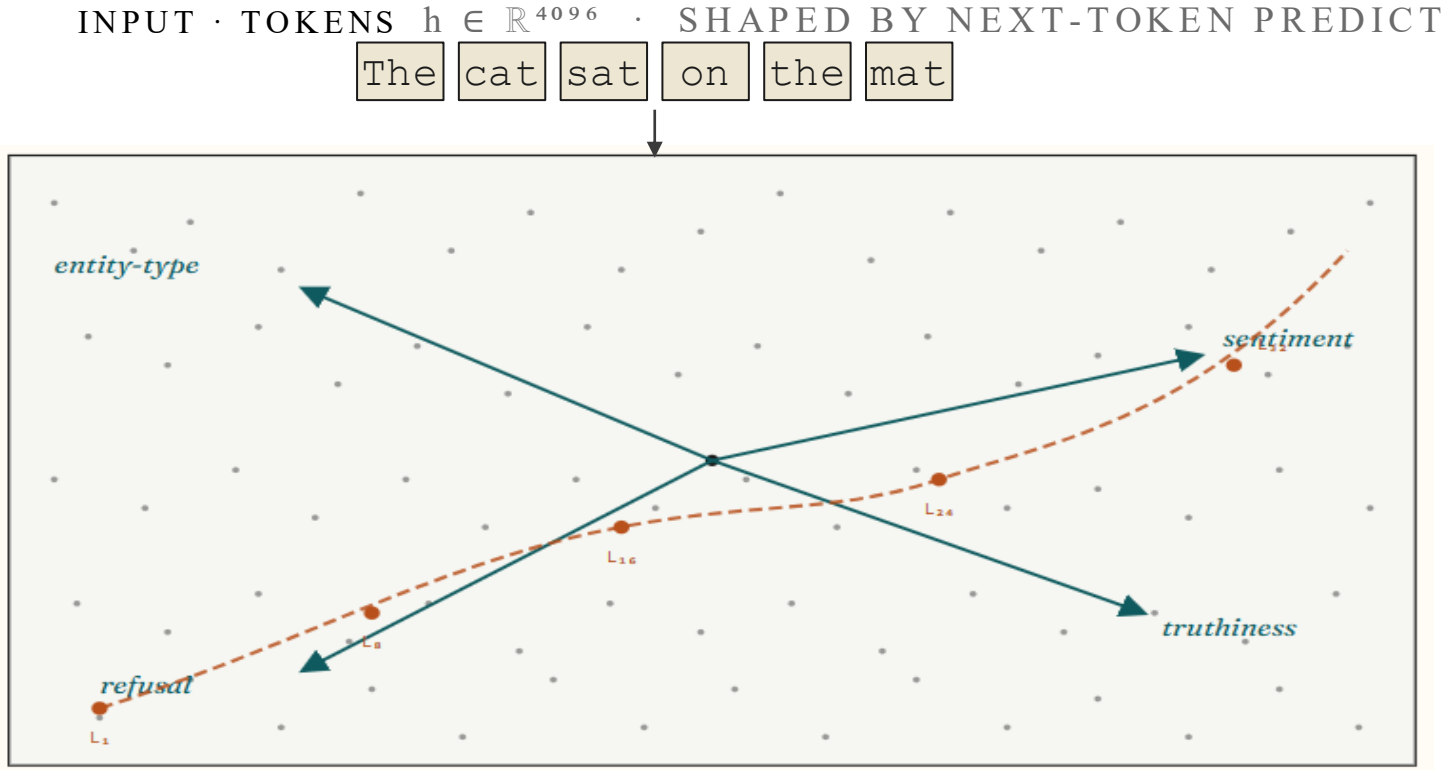
- Reasoning by Superposition
- CoT²
- CoLaR
- UniVLA(robotics)

2025.8-2025.11

- MemGen(Agent system
- LTPO
- 3DThinker
- C2C

Language-model hidden states

INPUT · TOKENS $h \in \mathbb{R}^{4096}$ · SHAPED BY NEXT-TOKEN PREDICTION



$|\mathcal{V}|$ = 32,000 DISCRETE SYMBOLS · NO INVERSE BACK TO h

high-dim · dispersed · directions carry features · projection → only · no decoder

LLM's hidden space is a high-dim ambient cloud, not a compact manifold. What gives it structure is not reconstruction pressure but ***directions*** *— linear subspaces that probes reveal correspond to features like sentiment or refusal.*

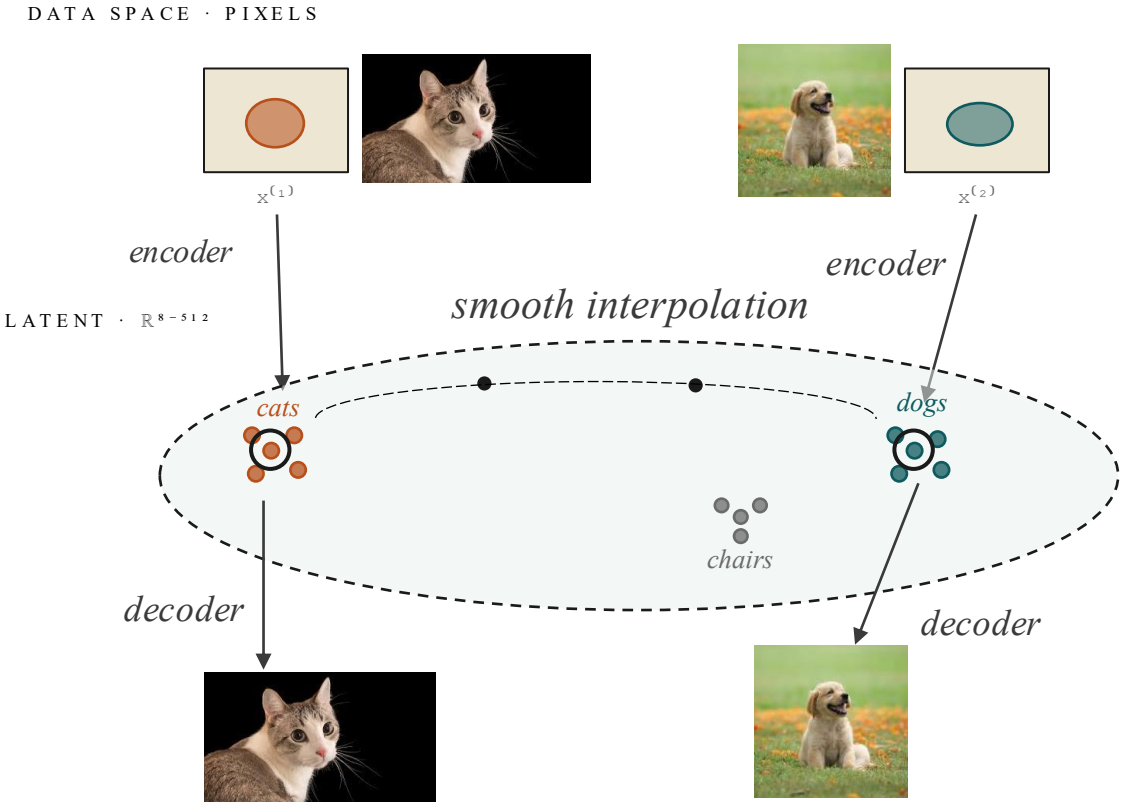
A single token's hidden state traces a trajectory through this cloud as it passes through the layers, and at the end a lossy linear map W_U projects it onto the 32k-symbol vocabulary. There is no inverse path from hidden state back to anything

Not the same as a *VAE* or *diffusion* latent.

If you've worked with Stable Diffusion or VQ-VAE, you already carry a mental picture of 'latent space' — a compact manifold to sample and interpolate on. The LLM version is geometrically a different animal.

VAE / Diffusion latent

$z \in \mathbb{R}^{8-512}$ · ANCHORED TO PIXEL RECONSTRUCTION



compact · clustered · every z → valid image · encoder ⇄ decoder

VAE latent is a compact, clustered manifold — every point decodes to a valid image, and walking between two points smoothly morphs content. The encoder/decoder pair enforces this geometry: if z doesn't look like pixels after decoding, training pushes it to.

THE TAXONOMY

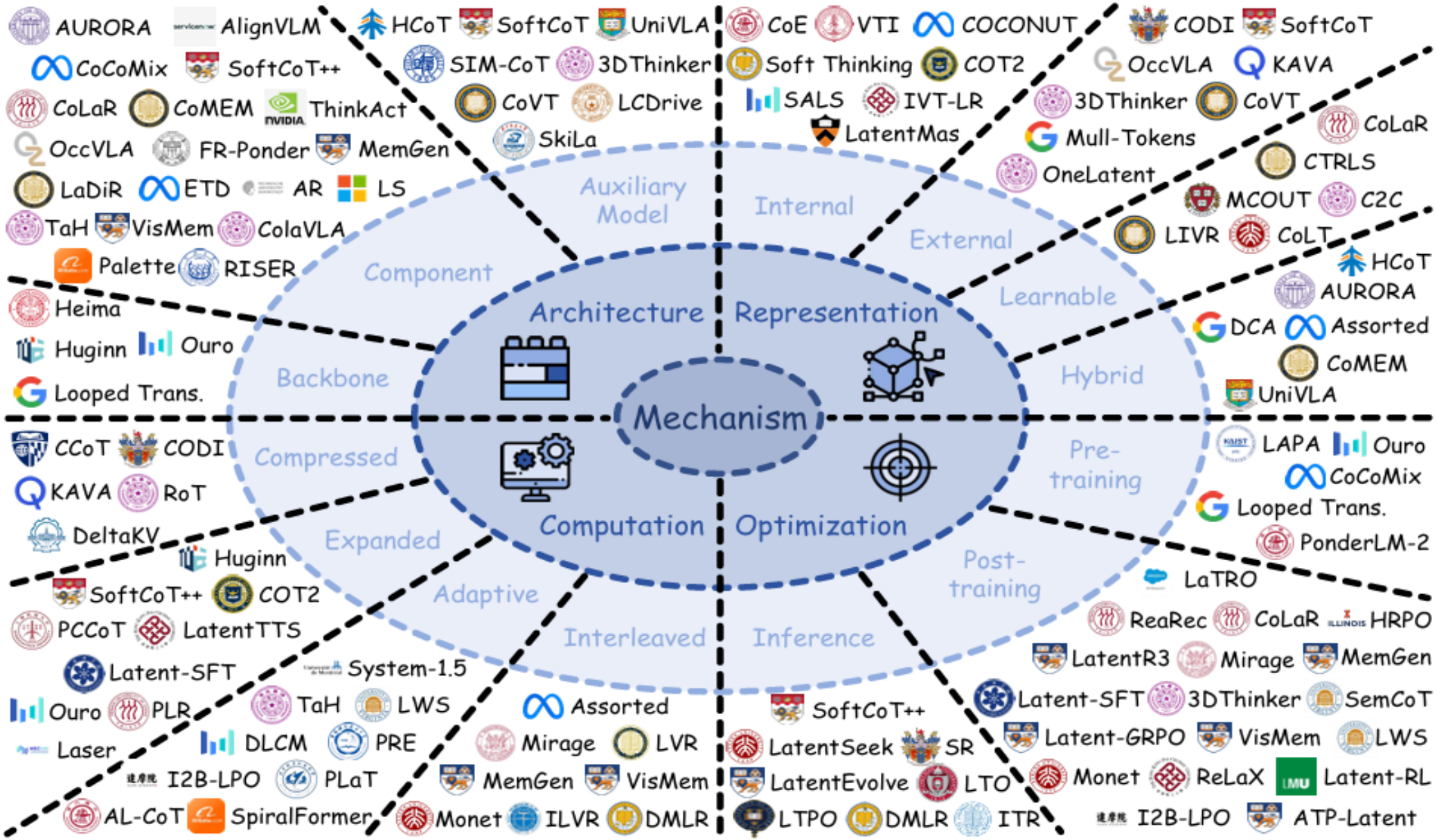


Figure 5 Representative works operate in accordance with latent space mechanisms. We classify all methods into four lines based on diverse ways of utilizing the latent space, including: **Architecture** (Section 4.1), **Representation** (Section 4.2), **Computation** (Section 4.3), and **Optimization** (Section 4.4).

THE FOUR DESIGN AXES, TOGETHER

How latent space is actually built.(4 pages)

Every concrete method makes a decision along four orthogonal axes — four successive questions a designer has to answer.

"Does latent reasoning need new network structure?" — This axis is about **where in the model** latent computation happens. Three sub-choices: **Backbone** redesigns the whole network (recurrent-depth Transformers, diffusion-as-reasoner). **Component** swaps one block — an extra reasoning head, a planning module. **Auxiliary** bolts on an external structure (memory bank, scratchpad tensor) without changing the backbone. The answer shapes your compute budget and what you can fine-tune.

Representation Four sub-choices. Internal: reuse residual stream (COCONUT). External: separate learned tensor as workspace (SoftCoT). Learnable tokens: insert <think> tokens trained to carry reasoning (Pause, Filler). Hybrid: continuous vectors + discrete tokens (CODI).

Computation

"How do extra steps of thought actually happen?"

Two regimes. Expanded depth: run same layers more times (loop forward pass N times — Huginn, CoTFormer). Expanded width: run many passes in parallel and combine (latent best-of-N). Depth trades wall-clock; width trades memory.

Optimization

"How do you train what you can't label?"

Three answers. Supervised: align latent states with teacher trajectories from a CoT model. Reinforcement: treat latent trajectory as policy, train on reward (HRPO, latent GRPO). Self-supervised: next-token prediction augmented with auxiliary objectives.

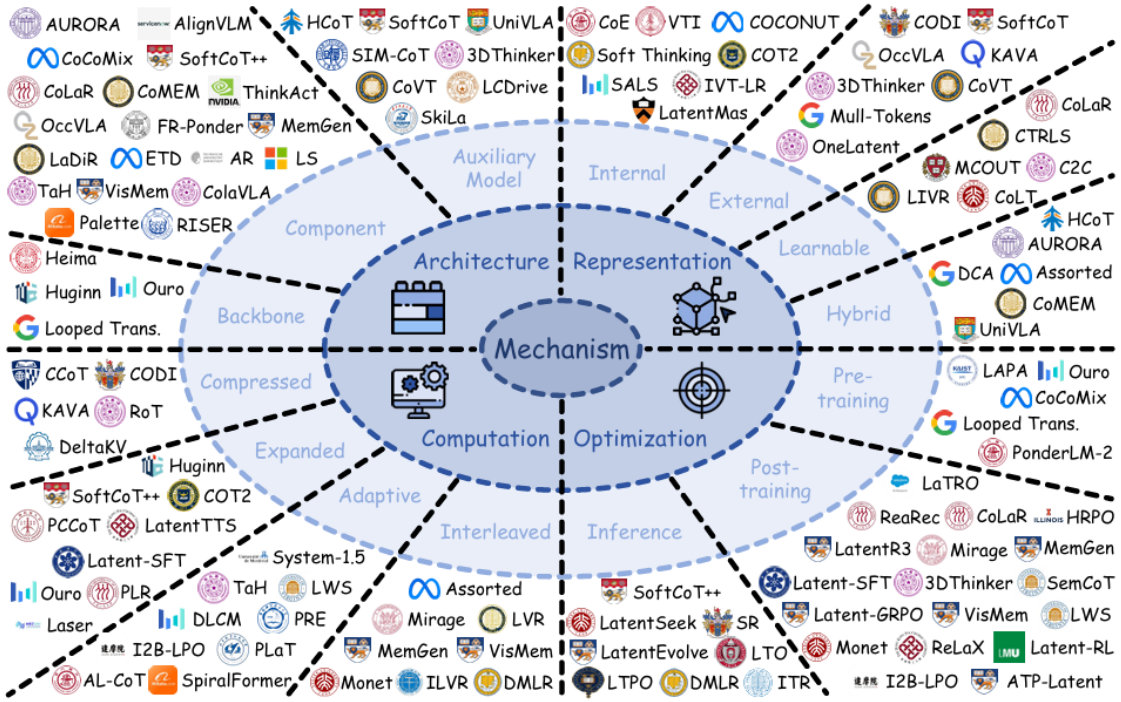


Figure 5 Representative works operate in accordance with latent space mechanisms. We classify all methods into four lines based on diverse ways of utilizing the latent space, including: **Architecture** (Section 4.1), **Representation** (Section 4.2), **Computation** (Section 4.3), and **Optimization** (Section 4.4).

WHY FOUR, NOT ONE

A paper like COCONUT makes decisions on *all four* axes at once — it reuses the backbone (Architecture = Backbone), uses internal hidden states as thoughts (Representation = Internal), expands depth by looping inference (Computation = Depth), and trains with distillation (Optimization = Supervised). Most methods can be summarized as a 4-tuple of choices. That's what makes the taxonomy predictive: once you know the 4-tuple, you know roughly what the system can and can't do.

Architecture

-Where is in the model latent computation happens

Table 2 Overview of the **Backbone** (Section 4.1.1) based architecture. We compare the hidden dimension, layer, size, and architectural feature of these backbones.

Date	Backbone	Paper	Code	Dimension	Layer	Size	Feature
01/25	Heima [173]	Paper	Code	4096	72	19B	encoder-decoder/progressive/adaptive decoding
02/25	Huginn [50]	Paper	Code	5280	8	3.5B	decoder-only/recurrent depth/shared recurrent block/test-time
02/25	Looped Trans. [167]	Paper	-	5120	24	1.5B	decoder-only/looped model/looping-based regularization
03/25	MoLAE [127]	Paper	-	512	12	0.1B	mixture of latent experts/shared projection/lower dimension
04/25	PHD-Trans. [223]	Paper	-	2048	16	1.2B	decoder-only/cache management/sliding window attention
09/25	PonderLM2 [267]	Paper	Code	2048	24	0.5B/1.4B	decoder-only/iterative refinement/Jacobi-style parallel updates
10/25	Ouro [296]	Paper	-	2048	24/48	1.4B/2.6B	decoder-only/recursive inference/parameter-shared loop
12/25	DLCM [158]	Paper	-	1536	32	2.3B	encoder-decoder/large concept model/hierarchical/heterogeneous
01/26	Dreamer [86]	Paper	-	1024	16/32	1B/2B	depth-recurrent/sequence-depth-sparse attention mixture
02/26	LoopFormer [72]	Paper	Code	2048	-	-	decoder-only/elastic-depth looped transformer
03/26	MLRA [121]	Paper	Code	3072	24	2.9B	multi-head low-rank attention/four-way tensor parallelism decoding

Backbone

The **main model itself** carries out iterative latent updates. Same transformer block is applied multiple times per forward pass, so depth scales without extra parameters.

Huginn recurrent-depth Transformer with shared-weight loops

PonderLM-2 Pondering with adaptive halting at each token

LCR-SER Latent Consistency Reasoner with sampling

Component

The **main model itself** carries out iterative latent updates. Same transformer block is applied multiple times per forward pass, so depth scales without extra parameters.

HCoT Hidden Chain-of-Thought module

COCONUT Loop the last hidden state back as the next input

SoftCoT Soft continuous CoT embedding

SIM-CoT Simulated continuous reasoning block

Auxiliary

A second model (often smaller or specialized) supplies either training signals or intermediate features. The host model remains the generator; the auxiliary model is the teacher or sensor.

MemGen Generative memory module

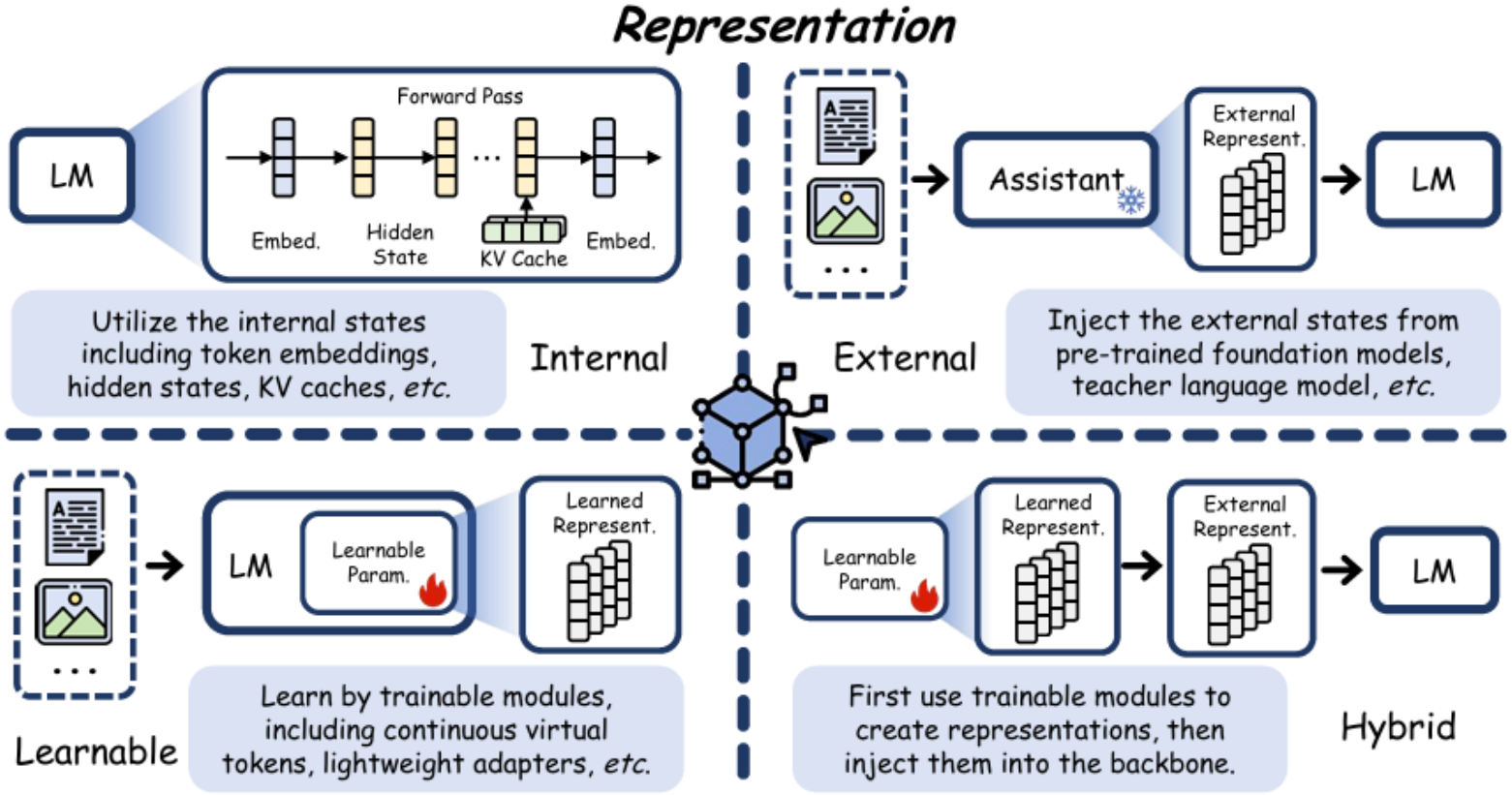
C2C Cache-to-cache agent communication

LatentMem Long-term latent memory bank

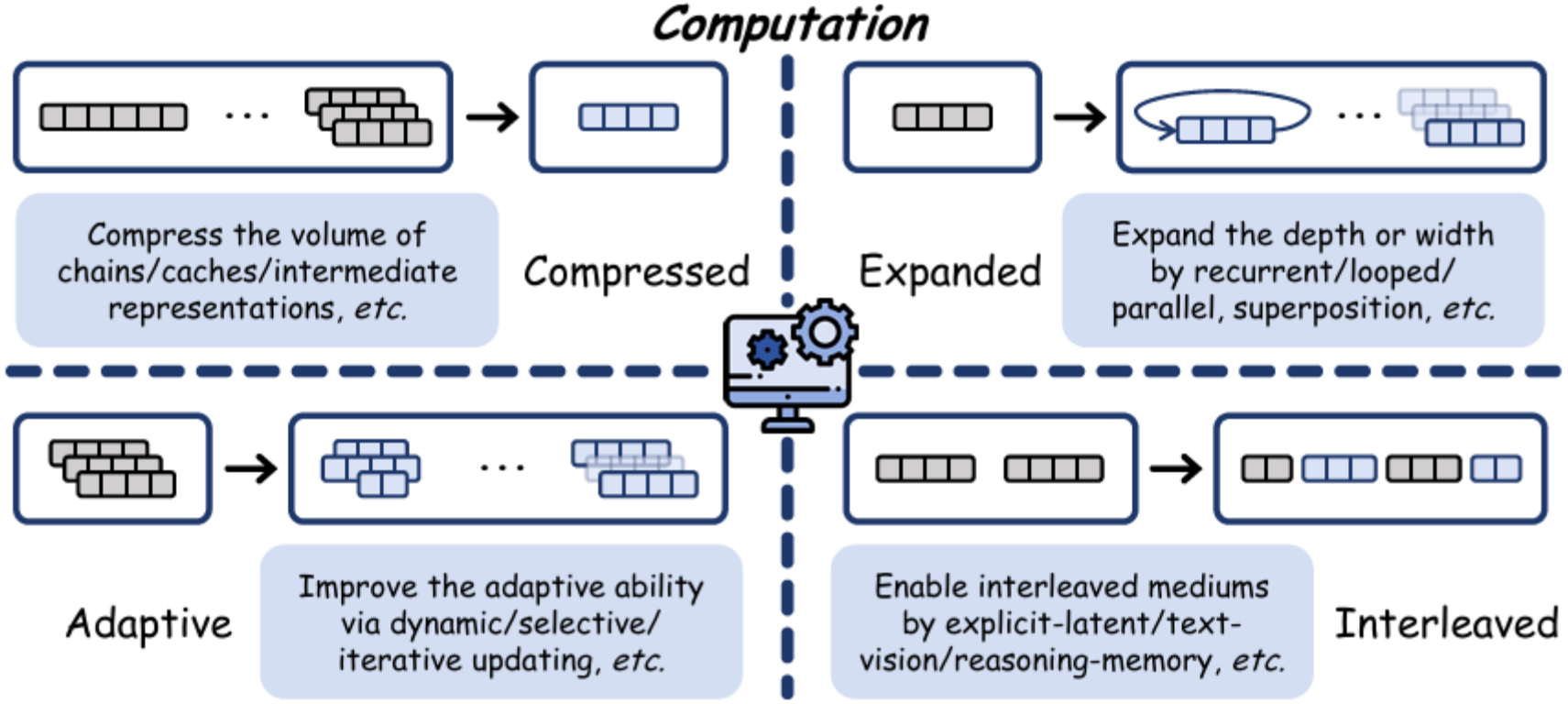
Scratchpad++ Auxiliary tensor as working memory

Representation

-"What kind of object is a 'thought'?"



Computation: expand depth or width?



Compressed TRADE LENGTH FOR DENSITY

Shrink long traces, caches, or cross-modal features into compact latents while preserving task-relevant content.

CCoT · CODI · KaVa · DeltaKV · HCoT

Expanded TRADE PARAMETERS FOR DEPTH/WIDTH

Increase effective computation via recurrence, looping, or parallel latent paths — more "thinking" per forward pass.

Huginn · Looped Trans. · SoftCoT++ · CoT² · PCCoT

Adaptive MATCH COMPUTE TO DIFFICULTY

Dynamic depth, halting, or routing — easy inputs get short paths, hard ones get more iterations.

TaH · FR-Ponder · System-1.5 · LWS · DLCM

Interleaved MIX DISCRETE & CONTINUOUS

Alternate tokens with latent states, or text with vision latents. Best of both: some steps verbalized, others not.

Mirage · LVR · SwiReasoning · SpiralThinker

AXIS 4 OF 4

Optimization: how do you train what you can't label?

You cannot supervise what you cannot label — and latent thoughts have no ground-truth string. The paper surveys three training strategies.

A Supervised (distillation)

Collect CoT trajectories from a teacher model. Project each teacher token back into the student's hidden space. Train student to match these continuous targets.

Distill-CoT	*token-to-hidden regression*
Latent-SFT	*SFT on continuous targets*
COT2Hidden	*teacher-student alignment*

B Reinforcement learning

Treat the latent trajectory as the policy. Roll out latent thoughts, evaluate the final answer, backprop reward. GRPO/PPO variants dominate in 2026.

HRPO	*hidden-state RPO*
Latent-GRPO	*GRPO on latent paths*
BoLT	*bootstrapped latent training*

C Self-supervised / auxiliary

Add auxiliary losses on hidden states during next-token pretraining — e.g. reconstruction at intermediate depths, consistency across loops, or contrastive objectives.

Soft-GRPO	*soft-RL auxiliary*
LatentTTS	*test-time self-training*
SelfConsist	*latent consistency*

Optimization

Mechanism: Optimization

- **Pre-training** (Section 4.4.1): starts with a randomly initialized model and trains it from the scratch, enabling native latent-level abilities.
- **Post-training** (Section 4.4.2): enhances the ability of pre-trained models, with diverse supervision signals and objectives, learning the latent space.
- **Inference** (Section 4.4.3): focuses on inference manipulation of latent states, allowing dynamic adjustment.

THE PAPER'S CENTRAL CONTRIBUTION

A map of a fragmented field.

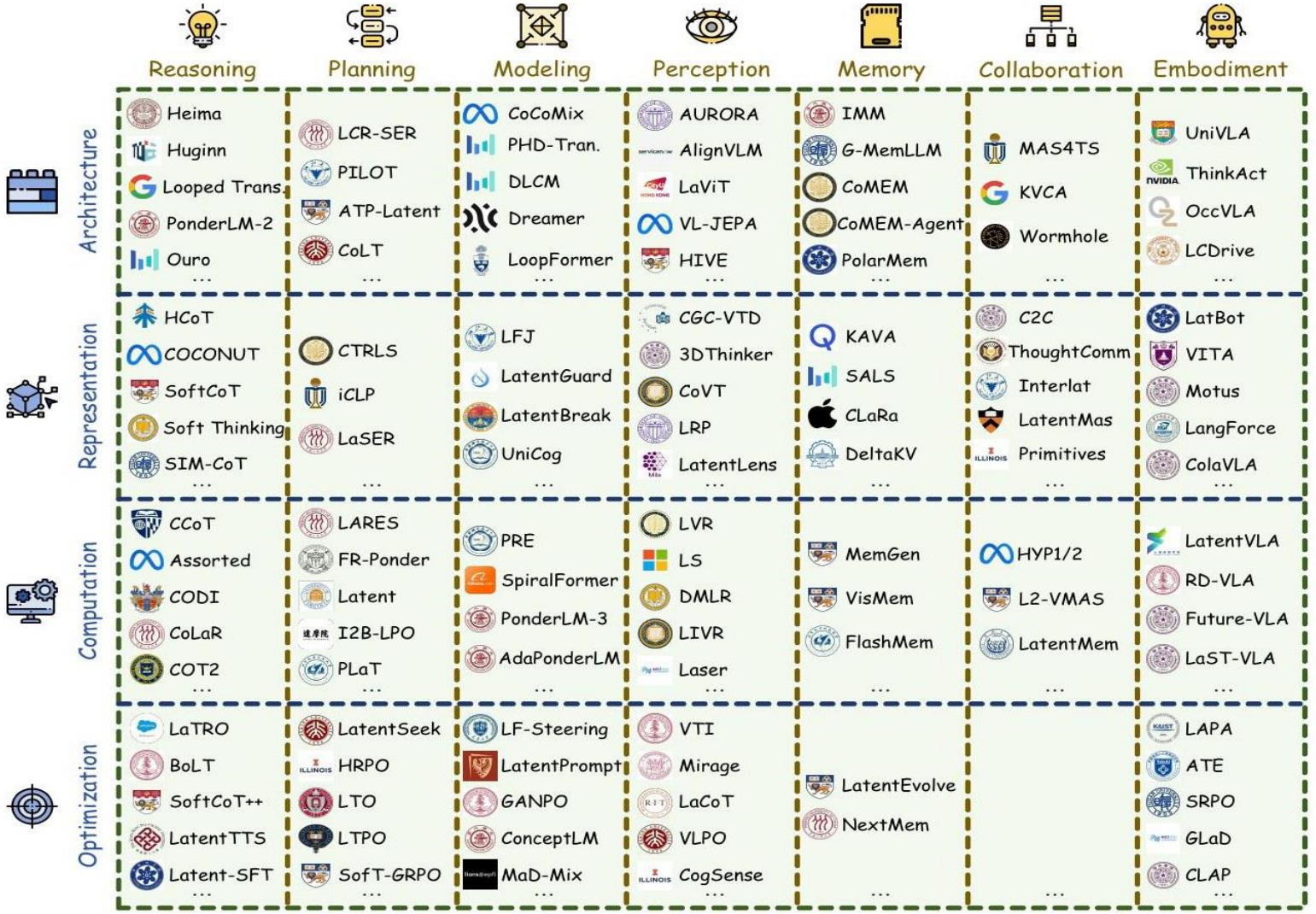


Figure 1 Overview of the latent space methods classified by two axes: four main **Mechanisms** (Section 4) and seven key

Capability

WHAT LATENT SPACE ACTUALLY UNLOCKS

Seven capabilities, one common thread.

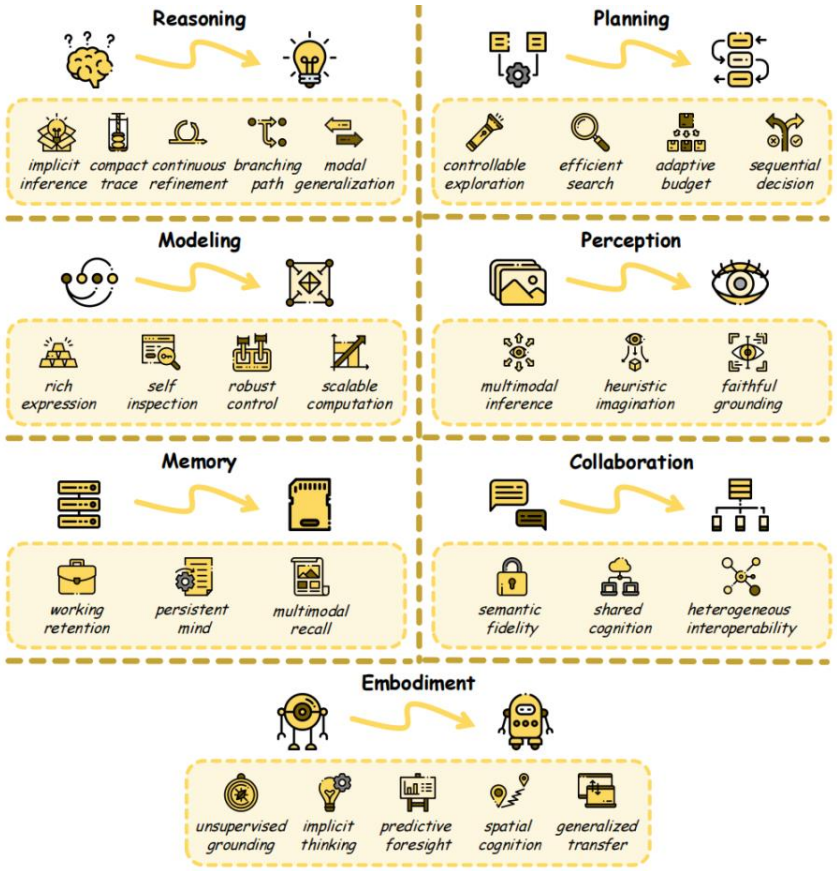


Figure 8 Core abilities brought by the latent space, including: **Reasoning** (Section 5.1), **Planning** (Section 5.2), **Modeling** (Section 5.3), **Perception** (Section 5.4), **Memory** (Section 5.5), **Collaboration** (Section 5.6), and **Embodiment** (Section 5.7).

Reasoning Branching paths without beam search. Continuous thoughts hold multiple half-developed hypotheses in superposition.

Planning Partial plans as continuous objects. Language forces commitment at every token; latent states hold the partial order.

World modeling State that isn't verbalizable. Physics, geometry, smooth dynamics — a ball's trajectory wants to be a vector.

Perception Pre-captioning visual reasoning. Vision models reason about an image without reducing it to a caption.

Memory Persistent traces. Compressed latent state summarizes long-horizon context more densely than any text summary.

Collaboration Agent-to-agent channel. Two LLMs pass hidden states directly — less lossy, much faster than English round-trips.

Embodiment Continuous motor control. Joint angles live in $\mathbb{R}^n$; tokens are the wrong alphabet for a robot.

The common thread: all seven involve information that is costly or lossy to externalize in language.

DEEP DIVE — CAPABILITY 1 OF 7

Reasoning in latent space: five sub-abilities.

Implicit inference	Skip verbalizing intermediate steps. The model performs logical chaining entirely in the residual stream, then outputs only the answer.	*COCONUT · Heima · Ouro*
Compact trace	Same reasoning as CoT, but with 5–10× fewer tokens because branching fits in a single continuous state.	*CoLT · SIM-CoT*
Continuous refinement	Iteratively polish a latent answer candidate over several inner loops — like beam search, but gradient-based.	*Looped Trans · PonderLM-2*
Branching paths	A single forward pass explores several logical branches in parallel via superposition. No beam search required.	*Huginn · LCR-SER*
Model generalization	Transfer latent reasoning skill learned on one domain (math, code) to a new one (science, law) without retraining the reasoning head.	*SoftCoT · SIM-CoT*

DEEP DIVE — CAPABILITY 7 OF 7

Embodiment: latent space meets the physical world.

The strongest case for latent reasoning: you are literally producing motor commands. Text is the wrong alphabet.

Unsupervised pretrain	Pretrain a latent action head on unlabeled video — learn the space of possible motor patterns before any reward signal.	*LAPA · Motus*
Implicit planning	Plan multi-step actions as latent trajectories, not token strings. Faster, differentiable, end-to-end trainable.	*ThinkAct · CoVT*
Predictive modeling	Learn a latent world model alongside the policy. Use it for look-ahead rollouts during action selection.	*Dreamer · LVR*
Spatial grounding	Tie latent hidden states to 3D spatial coordinates and object graphs — bridges perception and manipulation.	*3DThinker · VITA*
Generalized control	A single latent policy transfers across robot morphologies by operating in a shared embedding space.	*UniVLA · ColaVLA*

THE OPEN PROBLEMS

What's hard about this, actually.

01

Interpretability

The geometric cost of going latent. Explicit CoT is auditable; latent reasoning is not. No inverse from h back to readable text. Linear probes and SAEs help but are experimental.

02

Training signal

You can't supervise what you can't label. Most latent-reasoning systems rely on distillation from a CoT teacher, which caps their ceiling at the teacher's level.

03

Benchmark coverage

Current benchmarks were designed for token-based reasoning. Measuring latent advantage fairly is nontrivial — what counts as the same 'step'?

04

Scaling behavior

Does latent reasoning scale like CoT does? Early evidence is mixed — depth helps, but returns diminish faster than with explicit reasoning on some tasks.

IF YOU'RE LOOKING FOR A THESIS TOPIC

Five open directions worth pursuing.

1

Latent interpretability tools

Build probes, SAEs, or circuit-level tools that recover CoT-equivalent explanations from latent trajectories. The single most-cited gap in §6.

2

Latent RL without teachers

Most methods distill from a CoT teacher. Can you train latent reasoning from scratch with pure reward? Early HRPO/BoLT work suggests yes, but unstable.

3

Latent benchmarks

Design evaluations that reward latent advantage — branching, superposition, continuous confidence — not just final-answer accuracy.

4

Cross-modal latent spaces

Shared latent representations across text, vision, action. Most current systems glue modality-specific latents; a unified space is underexplored.

5

Safety of opaque reasoning

If the reasoning is unreadable, how do you verify alignment? Latent-space safety is a brand-new subfield with almost no published work.

IF YOU REMEMBER FIVE THINGS

Five takeaways.

01 LLMs reason in continuous hidden space. Tokens are the interface, not the workspace.

02 The token bottleneck is mechanical: a 15-bit round-trip every step that discards all but one candidate.

03 LLM 'latent space' is NOT a VAE manifold. It's a high-dim cloud with feature directions and one-way projection to vocab.

04 Every method = a 4-tuple of (Architecture, Representation, Computation, Optimization). Memorize this mental model.

05 The field went from 50 → 400+ papers/month in twelve months. Interpretability is the single biggest open problem.