Latent Thought Models with Variational Bayes Inference-Time Computation

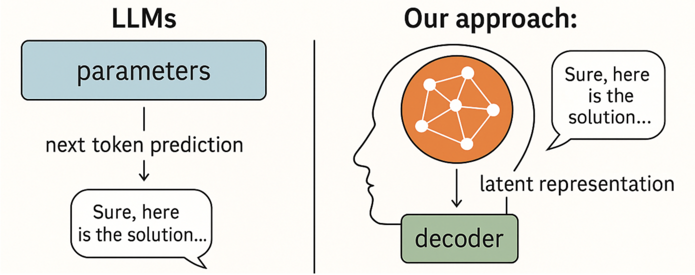
Jianwen Xie

Lambda

Motivation: The Language of Thought Hypothesis

The Problem: Standard LLMs have no explicit "thinking" step; "thinking" is an implicit, inseparable part of the token-generation process.

The Language of Thought [1] (Cognitive Science): Humans are hypothesized to use a structured, non-verbal "mentalese" to formulate thoughts before articulating them.

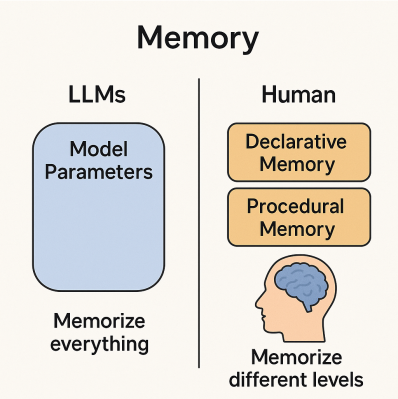


What if language models were given an **explicit internal space for thought**?

[1] Fodor, J. A. The Language of Thought. Harvard University Press, 1975.

Motivation: A Model with Two Memory Systems

The Problem: Standard LLMs use a single, monolithic memory, storing specific facts and general rules together in their parameters.



The declarative/procedural model [2]:

Declarative Memory:

1. Explicit knowledge of facts and events;
2. Characterized by **fast learning**, rapidly capturing unique, single-instance experiences.

Procedural Memory:

1. Implicit skills and rules, like grammar and syntax;
2. Characterized by **slow learning**, gradually internalizing general structures over time.

[2] Ullman, M. T. Contributions of memory circuits to language: The declarative/procedural model. Cognition, 2004.

Motivation: Beyond LLMs Scaling Laws

The Problem: The success of LLMs is driven by scaling laws that are now hitting a critical bottleneck: the scarcity of high-quality training data

Our Inspiration:

1. A "Language of Thought" that separates **thinking** from **verbalization**.
2. A dual-memory system that separates **fast-learning facts** from **slow-learning skills**.

The Central Question:

Can a model built on these principles unlock new scaling dimensions to achieve a new level of **sample and compute efficiency**?

Latent Thought Models (LTMs)

LTM: A model with explicit **latent thought vectors z** that guide text generation.

(1) **Structure**:
Latent thought vectors $\mathbf{z} = (\mathbf{z}_1, \ldots, \mathbf{z}_L) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

(2) **Generation**:
Transformer decoder generates tokens conditional on $\mathbf{z}$ as $p_\beta(\mathbf{x}|\mathbf{z})$, with a short context window ($k = 256$)

- ⋆ Layer l uses $\mathbf{z}_l$ via cross-attention.
- ⋆ Short context forces $\mathbf{z}$ to be an information carrier.
- ⋆ $\mathbf{z} \to$ declarative memory (local latent variable)
- ⋆ $\beta \to$ procedural memory (global parameters)

$$p_\beta(\mathbf{x}|\mathbf{z}) = \prod_{n=1}^{N} p_\beta(x^{(n)}|\mathbf{z}, \mathbf{x}^{(n-k:n-1)}).$$

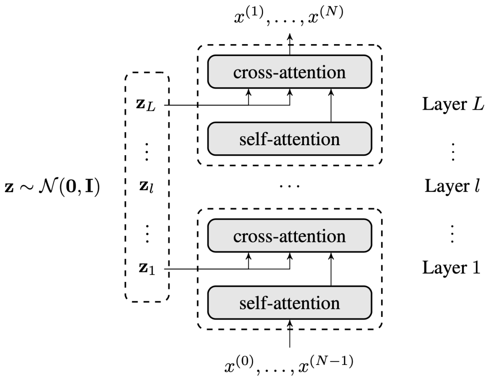


Figure: LTM Architecture ($L = 12$, $N = 1024$).

Fast-Slow Variational Bayes Learning

Variational Bayes: Optimize the Evidence Lower Bound (ELBO) with posterior $q(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mu, \sigma^2)$:

$$\mathcal{L}(\beta, \mu, \sigma^2) = \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\log p_\beta(\mathbf{x}|\mathbf{z})] - \mathrm{KL}(q(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z})).$$

1. Fast Learning (Posterior Inference)

Aim: Infer local parameters (μ, σ^2) for each $\mathbf{x}$.

How: T_{fast} gradient steps on (μ, σ^2) per input as $\frac{\partial \mathcal{L}}{\partial \mu}, \frac{\partial \mathcal{L}}{\partial \sigma}$ and $\mathbf{z} = \mu + \sigma \cdot \epsilon$.

— Rapid, on-the-fly "thinking" or parsing

— Inference-time (test-time) computation

— declarative memory

2. Slow Learning (Decoder Training)

Aim: Update the shared decoder parameters β.

How: A single gradient step on β per batch as $\frac{\partial \mathcal{L}}{\partial \beta}$.

— Gradual accumulation of grammar/syntax

— procedural memory.

The Fast-Slow Learning Algorithm

Algorithm 1 Dual-rate learning of LTM

Training data $\{\mathbf{x}_i\}_{i=1}^N$, generator $p_\beta(\mathbf{x}|\mathbf{z})$, learning rates η_{fast} and η_{slow}, fast learning steps T_{fast}.

while not converged **do**

  Sample mini-batch $\{\mathbf{x}_i\}_{i=1}^B$

  for each $\mathbf{x}_i$ in the mini-batch **do**

    `// fast learning`

    Initialize $\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i^2$

    for $t = 1$ to T_{fast} **do**

      Sample $\mathbf{z} \sim q_{\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i^2}(\mathbf{z}|\mathbf{x}_i)$

      Compute $\mathcal{L}_i = \mathbb{E}_q[\log p_\beta(\mathbf{x}_i|\mathbf{z})] - D_{\mathrm{KL}}(q(\mathbf{z}|\mathbf{x}_i)||p(\mathbf{z}))$.

      Update $\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i^2$ using AdamW with η_{fast}.

    end for

  end for

  `// slow learning`

  Compute batch loss $\mathcal{L}_{\text{batch}} = \frac{1}{B}\sum_{i=1}^B \mathcal{L}_i$

  Update β using AdamW with η_{slow}.

end while

Generation

Conditional: $\mathbf{x}$ question/instruction, $\mathbf{y}$ answer/completion.

$$p_\beta(\mathbf{y}|\mathbf{x}) \approx \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[p_\beta(\mathbf{y}|\mathbf{x},\mathbf{z})].$$

Uses variational inference for $\mathbf{z}$, then autoregressive sampling.

Unconditional:

$$p_\beta(\mathbf{x}) = \mathbb{E}_{p(\mathbf{z})}[p_\beta(\mathbf{x}|\mathbf{z})].$$

Experimental Setup

Dataset:

— Training: OpenWebText (8B tokens) [3];

— Validation: Penn Tree Bank (PTB) [4], WikiText [5], etc.

Baselines:

— Autoregressive models: GPT-2 [6], AR [8]

— Discrete diffusion models: SEDD [7], MDLM [8], MD4 [9].

LTMs:

— Small (38M, 3 layers), Medium (51M, 6 layers), Large (76M, 12 layers).

[3] Gokaslan, A. and Cohen, V. Openwebtext corpus. http://Skylion007.github.io/ OpenWebTextCorpus, 2019.

[4] Marcus, M. et al. Building a large annotated corpus of english: The penn treebank. Computational linguistics, 19(2):313–330, 1993.

[5] Merity, S., et al. Pointer sentinel mixture models. arXiv preprint arXiv:1609.07843, 2016.

[6] Radford, A., et al. Language models are unsupervised multitask learners. OpenAI blog, 2019.

[7] Lou, A. et al. Discrete Diffusion Modeling by Estimating the Ratios of the Data Distribution. ICML, 2024.

[8] Sahoo, S. S. et al. Simple and effective masked diffusion language models. arXiv preprint arXiv:2406.07524, 2024.

[9] Shi, J. et al. Simplified and generalized masked diffusion for discrete data. arXiv preprint arXiv:2406.04329, 2024.

Latent Thought Model (LTM)s introduce new ways to trade compute for performance, beyond just model and data size. Inference-time Computation $= T_{\text{fast}}$ steps of gradient descent to get $q(\mathbf{z}|\mathbf{x})$.

The total training cost, training FLOPs per token (trFLOPs/tok), is dominated by **inference steps** (T_{fast}).

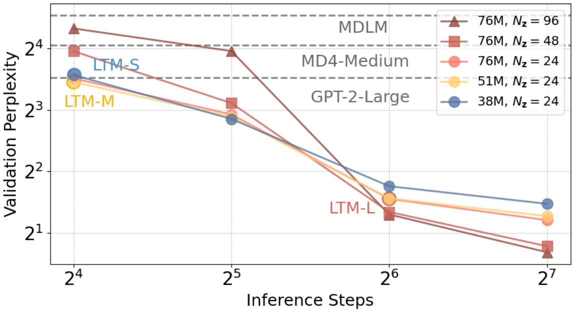


Figure: Validation perplexity across configurations.

Key Insights:

1. Increasing inference steps consistently improves performance.
2. Adding more latent thought vectors (N_z) provides additional gains, especially for larger models.

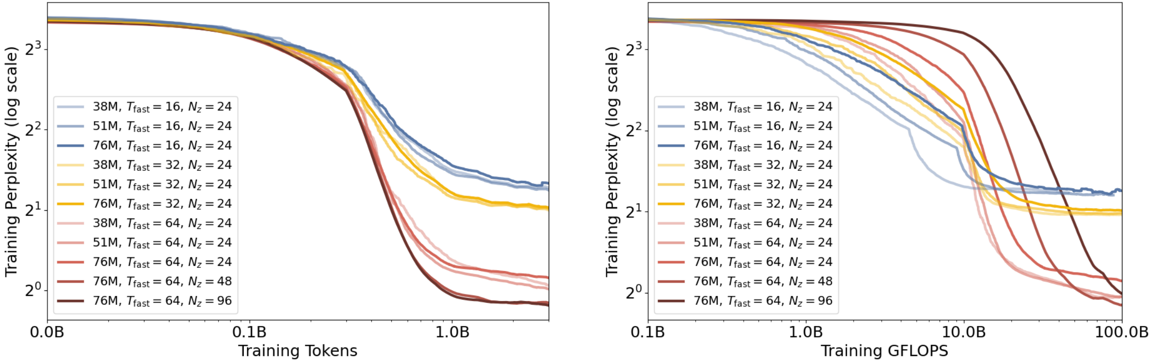
Zero-Shot Perplexity: Dominating Baselines

LTMs achieve significantly better perplexity with far fewer parameters.

Model Family	Model Size	trFLOPs/tok	# Tokens	PTB	WikiText	LM1B	LAMBADA	AG News	PubMed	Arxiv
GPT-2-Medium	345M	2.42G	–	130.04	32.14	44.03	36.09	44.53	23.33	23.82
GPT-2-Large	762M	5.32G	–	161.33	30.09	45.61	34.26	39.93	68.15	21.01
AR (Sahoo et al., 2024)	110M	0.85G	524B	82.05	25.75	51.25	51.28	52.09	49.01	41.73
AR-Retrained	76M	0.46G	105B	258.95	52.49	107.37	61.55	110.31	60.61	55.35
SEDD (Sahoo et al., 2024)	110M	0.85G	524B	$\leq$ 100.09	$\leq$ 34.28	$\leq$ 68.20	$\leq$ 49.86	$\leq$ 62.09	$\leq$ 44.53	$\leq$ 38.48
SEDD (Lou et al., 2024)	345M	2.42G	–	$\leq$ 87.12	$\leq$ 29.98	$\leq$ 61.19	$\leq$ 42.66	–	–	–
MDLM (Sahoo et al., 2024)	110M	0.85G	524B	$\leq$ 95.26	$\leq$ 32.83	$\leq$ 67.01	$\leq$ 47.52	$\leq$ 61.15	$\leq$ 41.89	$\leq$ 37.37
MD4 (Shi et al., 2024)	345M	2.42G	–	$\leq$ 66.07	$\leq$ 25.84	$\leq$ 51.45	$\leq$ 44.12	–	–	–
LTM-Small ($T_{\text{fast}} = 16$)	38M	4.07G	7B	$\leq$ 34.71	$\leq$ 18.87	$\leq$ 23.59	$\leq$ 19.31	$\leq$ 34.76	$\leq$ 22.73	$\leq$ 21.67
LTM-Medium ($T_{\text{fast}} = 16$)	51M	5.52G	5.2B	$\leq$ 32.06	$\leq$ 17.39	$\leq$ 25.16	$\leq$ 17.32	$\leq$ 27.89	$\leq$ 20.45	$\leq$ 19.22
LTM-Large ($T_{\text{fast}} = 64$)	76M	32.2G	0.9B	$\leq$ **4.43**	$\leq$ **3.66**	$\leq$ **3.92**	$\leq$ **3.48**	$\leq$ **4.56**	$\leq$ **3.87**	$\leq$ **3.54**

LTM-Large, with only 76M parameters, achieves state-of-the-art perplexity.

Models with more inference steps achieve better sample efficiency and become more compute-efficient at larger scales.



(a) Sample Efficiency

(b) Compute Efficiency

Figure: Scaling behavior over training tokens (left) and GFLOPs (right).

Emergent Few-Shot Reasoning at Small Scale

LTMs demonstrate in-context learning capabilities for arithmetic reasoning (GSM8K) at significantly smaller scales.

$$\mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[p_\beta(\mathbf{y}|\mathbf{x},\mathbf{z})] \quad \mathbf{x}: \text{Few-shot Prompt} \quad \mathbf{y}: \text{Answer}$$

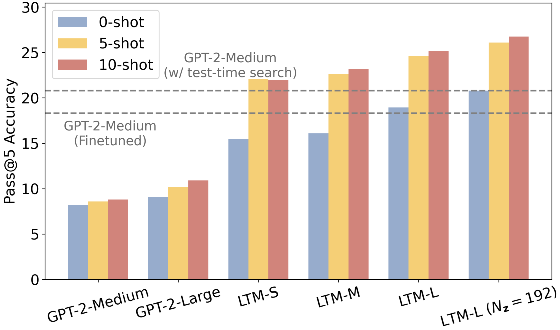


Figure: Pass@5 accuracy on GSM8K.

Conclusion

We present Latent Thought Models (LTMs), a new class of language models that leverage explicit latent vectors and inference-time computation.

Key Achievements:

- Established new scaling laws for inference steps.
- Achieved state-of-the-art perplexity and superior parameter efficiency.
- Unlocked few-shot reasoning at small model scales.

Acknowledgments

I thank Deqian Kong, Minglu Zhao, Dehong Xu, Bo Pang, Shu Wang, Edouardo Honig, Zhangzhang Si, Chuan Li, Sirui Xie, and Ying Nian Wu for their collaboration.

Thank you!