Why We Think

June 21, 2025

Date: May 1, 2025 **Estimated Reading Time:** 40 min **Author:** Lilian Weng

Special thanks to John Schulman for a lot of super valuable feedback and direct edits on this post.

Test time compute (Graves et al. 2016, Ling, et al. 2017, Cobbe et al. 2021) and Chain-of-thought (CoT) (Wei et al. 2022, Nye et al. 2021), have led to significant improvements in model performance, while raising many research questions. This post aims to review recent developments in how to effectively use test-time compute (i.e. "thinking time") and why it helps.

1 Motivation

Enabling models to think for longer can be motivated in a few different ways.

1.1 Analogy to Psychology

The core idea is deeply connected to how humans think. We humans cannot immediately provide the answer for "What's 12345 times 56789?". Rather, it is natural to spend time pondering and analyzing before getting to the result, especially for complex problems. In *Thinking, Fast and Slow* (Kahneman, 2013), Daniel Kahneman characterizes human thinking into two modes, through the lens of the dual process theory:

- **Fast thinking (System 1)** operates quickly and automatically, driven by intuition and emotion while requiring little to no effort.
- **Slow thinking (System 2)** demands deliberate, logical thought and significant cognitive efforts. This mode of thinking consumes more mental energy and requires intentional engagement.

Because System 1 thinking is fast and easy, it often ends up being the main decision driver, at the cost of accuracy and logic. It naturally relies on our brain's mental shortcuts (i.e., heuristics) and can lead to errors and biases. By consciously slowing down and taking more time to reflect, improve and analyze, we can engage in System 2 thinking to challenge our instincts and make more rational choices.

1.2 Computation as a Resource

One view of deep learning is that neural networks can be characterized by the amount of computation and storage they can access in a forward pass, and if we optimize them to solve problems using gradient descent, the optimization process will figure out how to use these resources—they'll figure out how to organize these resources into circuits for calculation and information storage. From this view, if we design an architecture or system that can do more computation at test time, and we train it to effectively use this resource, it'll work better.

In Transformer models, the amount of computation (flops) that the model does for each generated token is roughly 2 times the number of parameters. For sparse models like mixture of experts (MoE), only a fraction of the parameters are used in each forward pass, so computation $= 2 \times$ parameters/sparsity, where sparsity is the fraction of experts active.

On the other hand, CoT enables the model to perform far more flops of computation for each token of the answer that it is trying to compute. In fact, CoT has a nice property that it allows the model to use a variable amount of compute depending on the hardness of the problem.

1.3 Latent Variable Modeling

A classic idea in machine learning is to define a probabilistic model with a latent (hidden) variable z and a visible variable y, where y is given to our learning algorithm. Marginalizing (summing) over the possible values of the latent variable allows us to express a rich distribution over the visible variables,

$$P(y) = \sum_{z \sim P(z)} P(y \mid z).$$

For example, we can model the distribution over math problems and solutions by letting x denote a problem statement, y be ground truth answer or proof, and z as a free-form thought process that leads to the proof. The marginal probability distribution to optimize would be

$$P(y \mid x) = \sum_{z \sim p(z|x)} P(y \mid x, z)$$

The latent variable perspective is particularly useful for understanding methods that involve collecting multiple parallel CoTs or searching over the CoT—these algorithms can be seen as sampling from the posterior $P(z \mid x, y)$. This view also suggests the benefits of using the log loss $\log P(y \mid x)$ as the target objective to optimize, as the log loss objective has been so effective in pretraining.

2 Thinking in Tokens

The strategy of generating intermediate steps before generating short answers, particularly for math problems, was explored by Ling, et al. 2017, who introduced the AQUA-RAT dataset, and then expanded by Cobbe et al. 2021, who introduced the Grade School Math (GSM) dataset. Cobbe et al. train a generator with supervised learning on human-written solutions and verifiers that predict the correctness of a candidate solution; they

can then search over these solutions. Nye et al. (2021) experimented with intermediate thinking tokens as "scratchpads" and Wei et al. (2022) coined the now-standard term **chain-of-thought** (CoT).

Early work on improving CoT reasoning involved doing supervised learning on human-written reasoning traces or model-written traces filtered for answer correctness, where the latter can be seen as a rudimentary form of reinforcement learning (RL). Some other work found that one could significantly boost math performance of instruction tuned models by prompting them appropriately, with "think step by step" (Kojima et al. 2022) or more complex prompting to encourage the model to reflect on related knowledge first (Yasunaga et al. 2023).

Later work found that the CoT reasoning capabilities can be significantly improved by doing reinforcement learning on a dataset of problems with automatically checkable solutions, such as STEM problems with short answers, or coding tasks that can be checked with unit tests (Zelikman et al. 2022, Wang et al., 2023, Liu et al., 2023). This approach rose to prominence with the announcement of o1-preview, o3, and the R1 tech report (DeepSeek-AI, 2025), which showed that a simple recipe where a policy gradient algorithm could lead to strong performance.

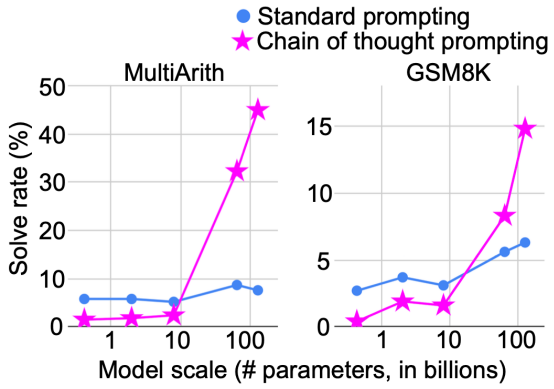

Figure 1: Chain-of-thought prompting leads to higher success rate of solving math problems. Larger models benefit more from thinking time. (Image source: Wei et al. 2022)

2.1 Branching and Editing

The fundamental intent of test-time compute is to adaptively modify the model's output distribution at test time. There are various ways of utilizing test time resources for decoding to select better samples and thus alter the model's predictions towards a more desired distribution. Two main approaches for improving the decoding process are parallel sampling and sequential revision.

- **Parallel sampling** generates multiple outputs simultaneously, meanwhile providing guidance per step with process reward signals or using verifiers to judge the quality at the end. It is the most widely adopted decoding method to improve test time performance, such as best-of-N or beam search. Self-consistency (Wang et al. 2023) is commonly used to select the answer with majority vote among multiple CoT rollouts when the ground truth is not available.

- **Sequential revision** adapts the model's responses iteratively based on the output in the previous step, asking the model to intentionally reflect its existing response and correct mistakes. The revision process may have to rely on a fine-tuned model, as naively relying on the model's intrinsic capability of self-correction without external feedback may not lead to improvement (Kamoi et al. 2024, Huang et al. 2024).

Parallel sampling is simple, intuitive and easier to implement, but bounded by the model capability of whether it can achieve the correct solution in one-go. Sequential explicitly asks the model to reflect on mistakes but it is slower and requires extra care during implementation as it does run the risk of correct predictions being modified to be incorrect or introducing other types of hallucinations. These two methods can be used together. Snell et al. (2024) showed that easier questions benefit from purely sequential test-time compute, whereas harder questions often perform best with an optimal ratio of sequential to parallel compute.

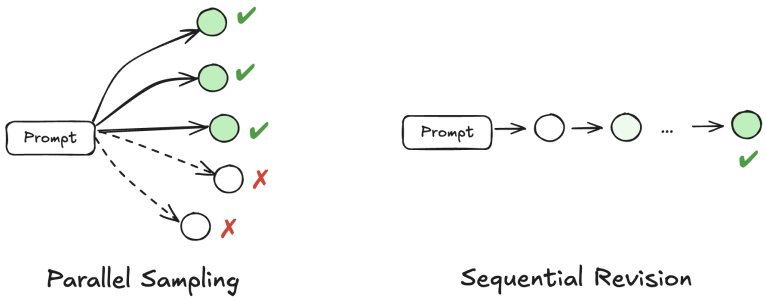

Figure 2: Illustration of parallel sampling vs sequential revision.

3 Citation

Please cite this work as:

```
Weng, Lilian. "Why We Think". Lil'Log (May 2025). https://lilianweng.github.io/posts/
05-01-thinking/
```

Or use the BibTex citation:

```
@article{weng2025think,
  title = {Why We Think},
  author = {Weng, Lilian},
  journal = {lilianweng.github.io},
  year = {2025},
  month = {May},
  url = "https://lilianweng.github.io/posts/2025-05-01-thinking/"
}
```

4 References

1 Alex Graves. *"Adaptive Computation Time for Recurrent Neural Networks."* arXiv preprint arXiv:1603.08983 (2016). https://arxiv.org/abs/1603.08983

2 Wang Ling, et al. *"Program Induction by Rationale Generation: Learning to Solve and Explain Algebraic Word Problems."* arXiv preprint arXiv:1705.04146 (2017). https://arxiv.org/abs/1705.04146

3 Karl Cobbe, et al. *"Training Verifiers to Solve Math Word Problems."* arXiv preprint arXiv:2110.14168 (2021). https://arxiv.org/abs/2110.14168

4 Jason Wei, et al. *"Chain of Thought Prompting Elicits Reasoning in Large Language Models."* NeurIPS 2022. https://arxiv.org/abs/2201.11903

5 Maxwell Nye, et al. *"Show Your Work: Scratchpads for Intermediate Computation with Language Models."* arXiv preprint arXiv:2112.00114 (2021). https://arxiv.org/abs/2112.00114

6 Daniel Kahneman. *Thinking, Fast and Slow.* Farrar, Straus and Giroux (2013).

7 Takeshi Kojima, et al. *"Large Language Models are Zero-Shot Reasoners."* NeurIPS 2022. https://arxiv.org/abs/2205.11916

8 Michihiro Yasunaga, et al. *"Large Language Models as Analogical Reasoners"* arXiv preprint arXiv:2310.01714 (2023). https://arxiv.org/abs/2310.01714

9 Eric Zelikman, et al. *"STaR: Bootstrapping Reasoning With Reasoning."* NeurIPS 2022. https://arxiv.org/abs/2203.14465

10 Xuezhi Wang, et al. *"Self-consistency Improves Chain of Thought Reasoning in Language Models."* ACL 2023. https://arxiv.org/abs/2203.11171

11 Ryo Kamoi, et al. *"When Can LLMs Actually Correct Their Own Mistakes? A Critical Survey of Self-Correction of LLMs."* TACL 2024. https://arxiv.org/abs/2406.01297

12 Jie Huang, et al. *"Large Language Models Cannot Self-Correct Reasoning Yet."* ICLR 2024. https://arxiv.org/abs/2310.01798

13 Noah Shinn, et al. *"Reflexion: Language Agents with Verbal Reinforcement Learning."* arXiv preprint arXiv:2303.11366 (2023). https://arxiv.org/abs/2303.11366

14 Yunxiang Zhang, et al. *"Small Language Models Need Strong Verifiers to Self-Correct Reasoning."* ACL Findings 2024. https://arxiv.org/abs/2404.17140

15 Hao Liu, et al. *"Chain of Hindsight Aligns Language Models with Feedback."* arXiv preprint arXiv:2302.02676 (2023). https://arxiv.org/abs/2302.02676

16 Sean Welleck, et al. *"Generating Sequences by Learning to Self-Correct."* arXiv preprint arXiv:2211.00053 (2023). https://arxiv.org/abs/2211.00053

17 Yuxiao Qu, et al. *"Recursive Introspection: Teaching Language Model Agents How to Self-Improve."* arXiv preprint arXiv:2407.18219 (2024). https://arxiv.org/abs/2407.18219

18 Aviral Kumar, et al. *"Training Language Models to Self-Correct via Reinforcement Learning."* arXiv preprint arXiv:2409.12917 (2024). https://arxiv.org/abs/2409.12917

19 Hunter Lightman, et al. *"Let's Verify Step by Step."* arXiv preprint arXiv:2305.20050 (2023). https://arxiv.org/abs/2305.20050

20 Yuxi Xie, et al. *"Self-Evaluation Guided Beam Search for Reasoning."* NeurIPS 2023. https://arxiv.org/abs/2305.00633

21 Yangzhen Wu, et al. *"Inference Scaling Laws: An Empirical Analysis of Compute-Optimal Inference for Problem-Solving with Language Models"* ICLR 2025. https://arxiv.org/abs/2408.00724

22 Dongwei Jiang, et al. *"RATIONALYST: Pre-training Process-Supervision for Improving Reasoning"* arXiv preprint arXiv:2410.01044 (2024). https://arxiv.org/abs/2410.01044

23 Xuezhi Wang and Denny Zhou. *"Chain-of-Thought Reasoning Without Prompting."* arXiv preprint arXiv:2402.10200 (2024). https://arxiv.org/abs/2402.10200

24 DeepSeek-AI. *"DeepSeek-V3 Technical Report."* arXiv preprint arXiv:2412.19437 (2024). https://arxiv.org/abs/2412.19437

25 DeepSeek-AI. *"DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning."* arXiv preprint arXiv:2501.12948 (2025). https://arxiv.org/abs/2501.12948

26 Luyu Gao, Aman Madaan & Shuyan Zhou, et al. *"PAL: Program-aided Language Models."* ICML 2023. https://arxiv.org/abs/2211.10435

27 Shunyu Yao, et al. *"ReAct: Synergizing Reasoning and Acting in Language Models."* ICLR 2023. https://arxiv.org/abs/2210.03629

29 Bowen Baker, et al. *"Monitoring Reasoning Models for Misbehavior and the Risks of Promoting Obfuscation."* arXiv preprint arXiv:2503.11926 (2025). https://arxiv.org/abs/2503.11926

30 Wojciech Zaremba, et al. *"Trading Inference-Time Compute for Adversarial Robustness."* arXiv preprint arXiv:2501.18841 (2025). https://arxiv.org/abs/2501.18841

31 Tamera Lanham, et al. *"Measuring Faithfulness in Chain-of-Thought Reasoning"* arXiv preprint arXiv:2307.13702 (2023). https://arxiv.org/abs/2307.13702

32 Boshi Wang, et al. *"Towards Understanding Chain-of-Thought Prompting: An Empirical Study of What Matters."* ACL 2023. https://arxiv.org/abs/2212.10001

33 Miles Turpin, et al. *"Language Models Don't Always Say What They Think: Unfaithful Explanations in Chain-of-Thought Prompting."* NeurIPS 2023. https://arxiv.org/abs/2305.04388

34 James Chua & Owain Evans. *"Are DeepSeek R1 And Other Reasoning Models More Faithful?"* arXiv preprint arXiv:2501.08156 (2025). https://arxiv.org/abs/2501.08156

35 Yanda Chen et al. *"Reasoning Models Don't Always Say What They Think"* arXiv preprint arXiv:2505.05410 (2025). https://arxiv.org/abs/2505.05410

36 Edward Yeo, et al. *"Demystifying Long Chain-of-Thought Reasoning in LLMs."* arXiv preprint arXiv:2502.03373 (2025). https://arxiv.org/abs/2502.03373

37 Mostafa Dehghani, et al. *"Universal Transformers."* ICLR 2019. https://arxiv.org/abs/1807.03819

38 DeLesley Hutchins, et al. *"Block-Recurrent Transformers."* NeurIPS 2022. https://arxiv.org/abs/2203.07852

39 Aydar Bulatov, et al. *"Recurrent Memory Transformers."* NeurIPS 2022. https://arxiv.org/abs/2207.06881

40 Jonas Geiping, et al. *"Scaling up Test-Time Compute with Latent Reasoning: A Recurrent Depth Approach."* arXiv preprint arXiv:2502.05171 (2025). https://arxiv.org/abs/2502.05171

41 Herel & Mikolov. *"Thinking Tokens for Language Modeling."* AITP 2023. https://arxiv.org/abs/2405.08644

42 Sachin Goyal et al. *"Think before you speak: Training Language Models With Pause Tokens."* ICLR 2024. https://arxiv.org/abs/2310.02226

43 Eric Zelikman, et al. *"Quiet-STaR: Language Models Can Teach Themselves to Think Before Speaking."* arXiv preprint arXiv:2403.09629 (2025). https://arxiv.org/abs/2403.09629

44 Wangchunshu Zhou et al. *"Towards Interpretable Natural Language Understanding with Explanations as Latent Variables."* NeurIPS 2020. https://arxiv.org/abs/2011.05268

45 Du Phan et al. *"Training Chain-of-Thought via Latent-Variable Inference."* NeurIPS 2023. https://arxiv.org/abs/2312.02179

46 Yangjun Ruan et al. *"Reasoning to Learn from Latent Thoughts."* arXiv preprint arXiv:2503.18866 (2025). https://arxiv.org/abs/2503.18866

47 Xuezhi Wang et al. *"Rationale-Augmented Ensembles in Language Models."* arXiv preprint arXiv:2207.00747 (2022). https://arxiv.org/abs/2207.00747

48 Jared Kaplan, et al. *"Scaling Laws for Neural Language Models."* arXiv preprint arXiv:2001.08361 (2020). https://arxiv.org/abs/2001.08361

49 Niklas Muennighoff & Zitong Yang, et al. *"s1: Simple test-time scaling."* arXiv preprint arXiv:2501.19393 (2025). https://arxiv.org/abs/2501.19393

50 Peiyi Wang, et al. *"Math-Shepherd: Verify and Reinforce LLMs Step-by-step without Human Annotations"* arXiv preprint arXiv:2312.08935 (2023). https://arxiv.org/abs/2312.08935

51 Yixin Liu, et al. *"Improving Large Language Model Fine-tuning for Solving Math Problems."* arXiv preprint arXiv:2310.10047 (2023). https://arxiv.org/abs/2310.10047

52 Charlie Snell, et al. *"Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters."* arXiv preprint arXiv:2408.03314 (2024). https://arxiv.org/abs/2408.03314

53 OpenAI. o1-preview: *"Learning to reason with LLMs."* Sep 12, 2024. https://openai.com/index/learning-to-reason-with-llms/

54 OpenAI. o3: *"Introducing OpenAI o3 and o4-mini."* Apr 16, 2025. https://openai.com/index/introducing-o3-and-o4-mini/