清華大學电子工程系
Department of Electronic Engineering, Tsinghua University

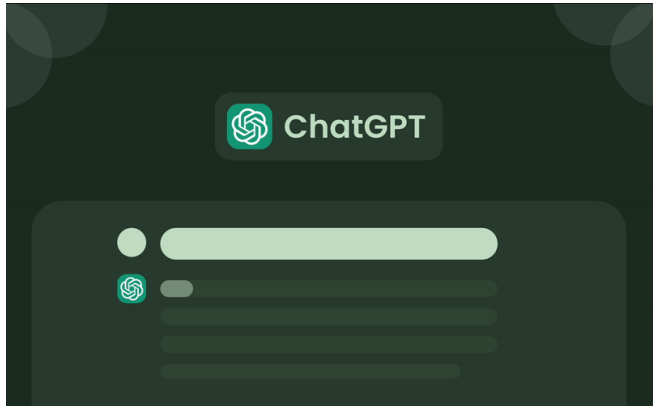
Think-at-Hard:
Selective Latent Iterations to Improve Reasoning Language Models

Tianyu Fu*, Yichen You*, Zekai Chen, Guohao Dai, Huazhong Yang, Yu Wang
equal contribution

contact for more questions: fuvty@outlook.com

Background: LLM

- LLMs are widely used across increasingly diverse domains and tasks



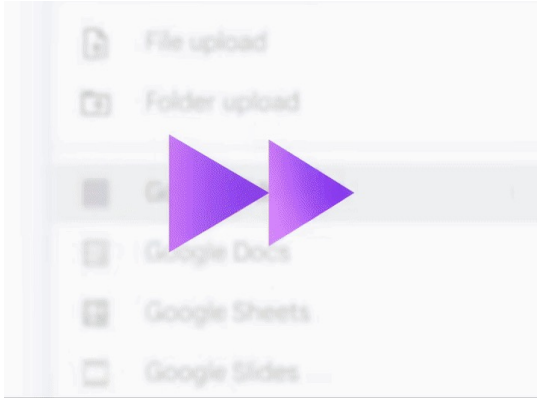
Chatbot

OpenAI ChatGPT、Google Gemini、Qwen……

Agent

OpenAI Operator,

Copilot



Creative Tools

Google Vids、Gamma

Specialized Models

Code,

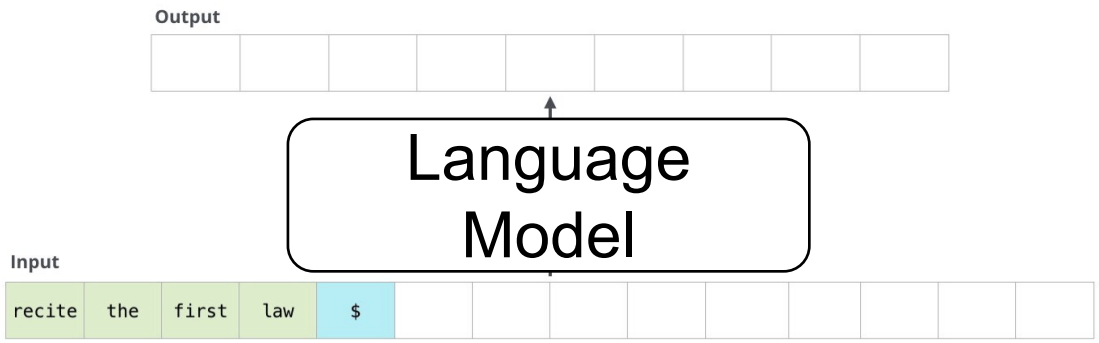
Math,

multi-modal

[1] Brown, Tom B. "Language models are few-shot learners." arXiv preprint arXiv:2005.14165 (2020).
[2] Wang, Lei, et al. "A survey on large language model based autonomous agents." Frontiers of Computer Science 18.6 (2024): 186345.
[3] Yin, Shukang, et al. "A survey on multimodal large language models." arXiv preprint arXiv:2306.13549 (2023).

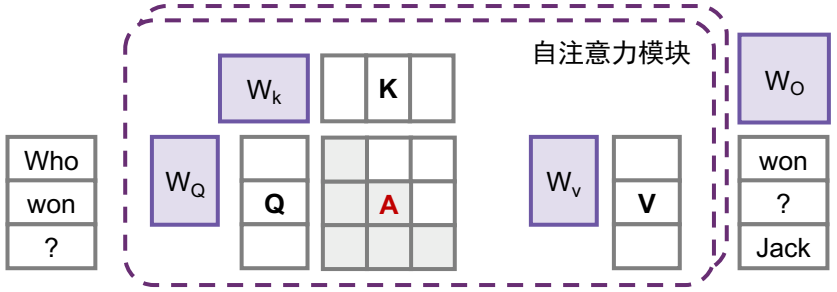
LLM Autoregressive Generation

- Generation scheme of LLM: prefill and decode



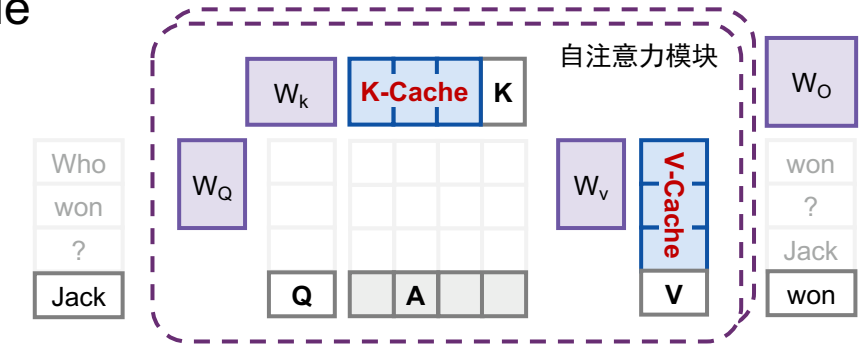
Autoregressive generation

prefill



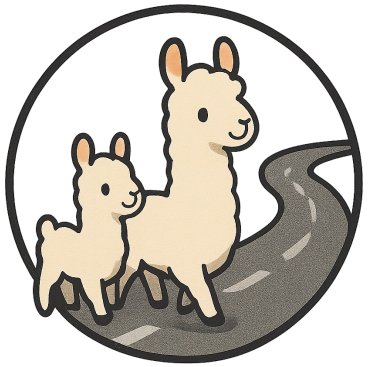
high-parallel

decode



low-parallel

Dynamic Token Budget Allocation



R2R

2025.06
NeurIPS'25

90:5:5 Law
most next-token predictions are easy

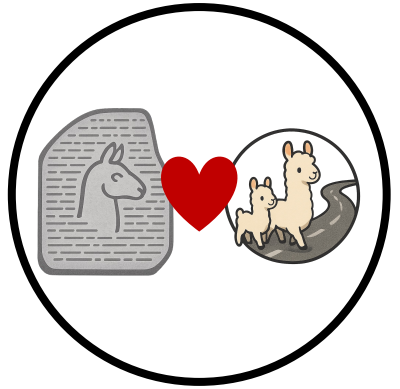
SLM + LLM
route only hard tokens to LLM at token level

TaH

2025.10
arXiv

Latent Overthinking
many easy tokens gone wrong at the second latent iteration

Selective Latent Iteration
think twice only on hard tokens



?

2026.?
something new

Recent Work: SLM-LLM Mix Inference (R2R)

Use small language model (SLM) and LLM for different reasoning steps

Motivation

Fast but weak SLM
slow but strong LLM

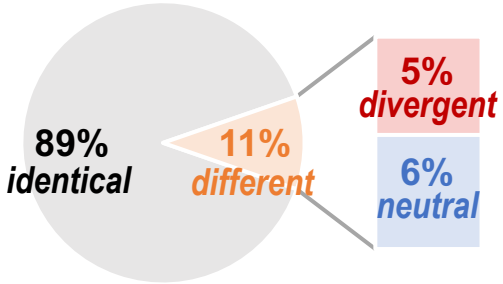
Tested results on AIME'24-25

Type	Model	Accuracy	Latency (s / question)
SLM	R1-1.5B	☹ 9%	☺ 199
LLM	R1-32B	☺ 45%	☹ 498

We should selectively use SLM and LLM for different generation steps, constructing a **fast and strong** mix-inference method

Insight

Given same context, SLM and LLM predictions are often identical



- 89% **identical** predictions
- 11% **different** predictions
  - some are **neutral**, like alternative expressions
  - only few **diverge** the meaning, logic, or conclusion of reasoning

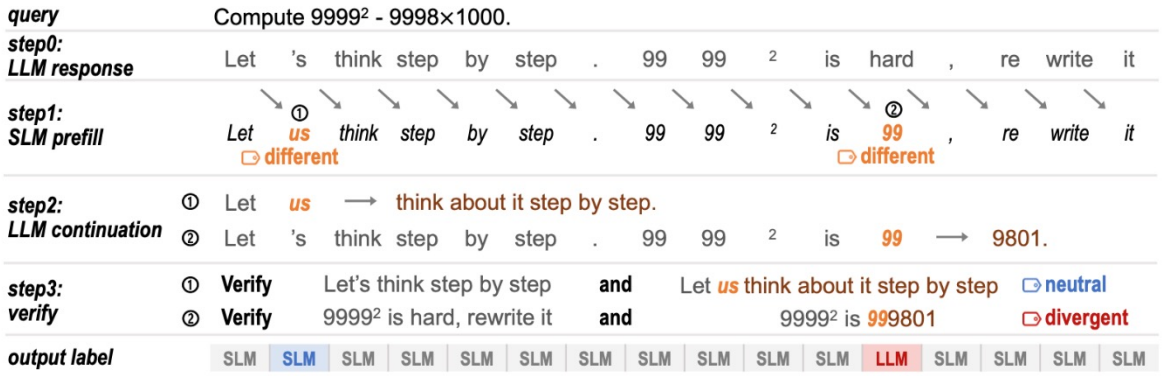
[1] Tianyu, Fu, et al. "Efficiently Navigating Divergent Reasoning Paths with Small-Large Model Token Routing" *NeurIPS'25*.

Recent Work: SLM-LLM Mix Inference (R2R)

Label divergent token, then train a neural token-router, utilizing LLMs only for path-divergent tokens during SLM generation

Method

Label divergent tokens generate model preference training data



Key idea:

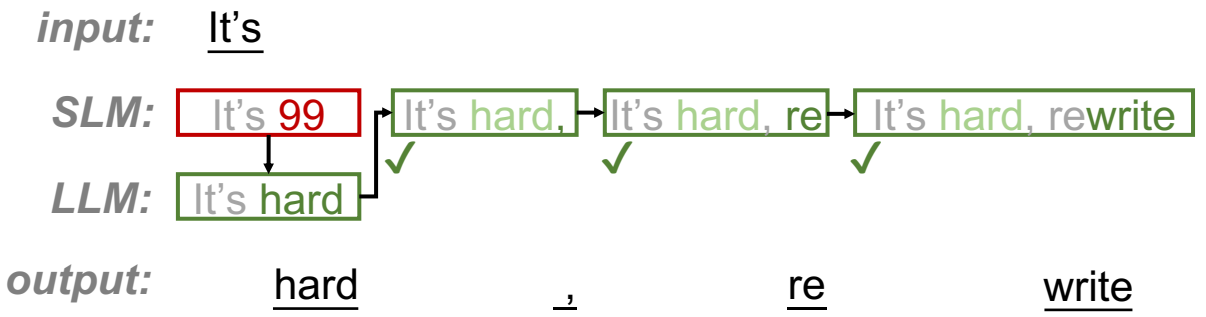
Step1. **find** all predictions where SLM-LLM **differ**

Step2. from the difference, let LLM **continue generation** until the end of current sentence, to understand difference's impact

Step3. ask **another LLM to verify** if difference causes **divergence**

Method

Train neural router, route to LLM for divergent SLM tokens



Routing scheme:

We train a 56M neural router

Given SLM output token & its last-layer hidden states,

it **classifies** whether this token is **divergent,**

Immediately route to LLM if predicted as divergent

[1] Tianyu, Fu, et al. "Efficiently Navigating Divergent Reasoning Paths with Small-Large Model Token Routing" *Submitted to NeurIPS'25. [Under Review]*

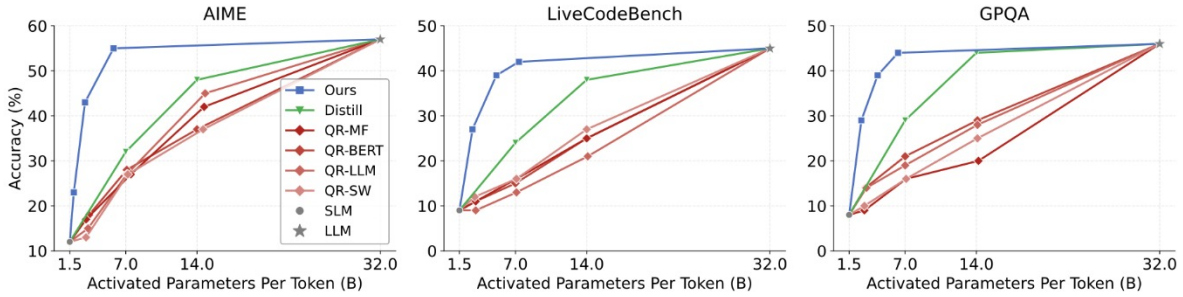
Experimental Results

Mixing R1-1.5B & 32B, uing **R2R** with **5.6B** avg. activated param. per token achieve **performance exceeding R1-14B**

Result

Performance-Efficiency Pareto Frontier

same avg. parameter, better accuracy



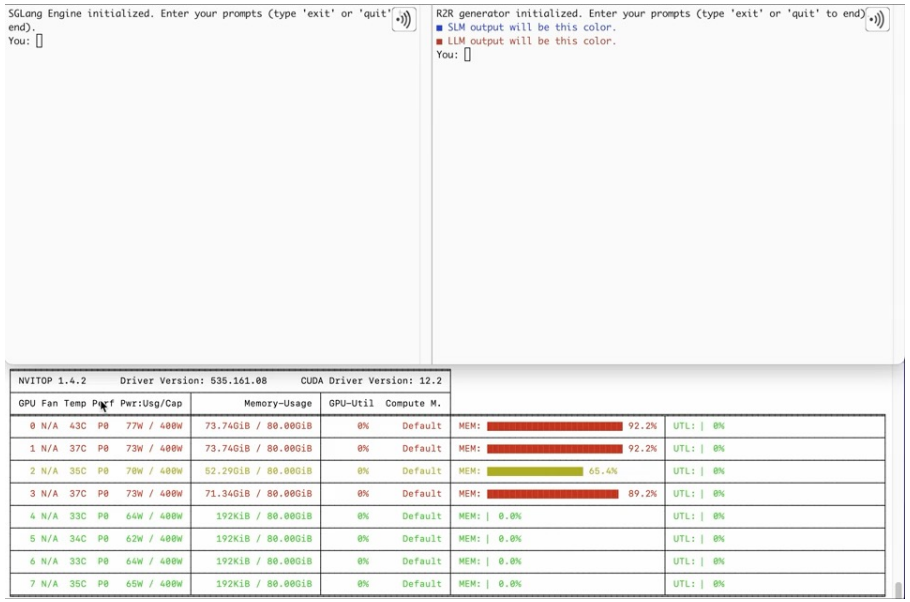
For R2R-5.6B (mix R1-1.5B & 32B)

- **Comparing R1-14B, 1.50x speedup, 1.07x accuracy**
- **Comparing R1-32B, 2.76x speedup,** achieving 92% of its accuracy with only 11%-15% LLM usage

Demo

Reaching 84.3 token/s on two A800-80GB GPUs

source code



R1-32B
Finished in: **1min 12s**

R2R
Finished in: **32s**

[1] Tianyu, Fu, et al. "Efficiently Navigating Divergent Reasoning Paths with Small-Large Model Token Routing" *Submitted to NeurIPS'25. [Under Review]*

Background

- Require strong reasoning model under limited parameter size

Device Type	Example GPU	Memory Size
Server	NVIDIA H100	80 GB
PC	RTX 4090	24 GB
Smartphone	A17 Pro	**8 GB**

Model Size (Qwen3)	AIME'24	MMLU-Redux
235B-A22B	85.7	92.7
32B	81.4	90.9
8B	76.0	87.5
4B	73.8	83.7
1.7B	**48.3**	**73.9**
0.6B	**10.7**	**55.6**

Background

- Why cannot small LLMs perform well?
  - few tokens gone wrong
    - evident by R2R
  - certain tokens are fundamentally harder to predict
    - encode critical logic
    - reasoning directions
    - …
  - limited computation per decoding step for small LLM

Motivation

- Oracle: latent overthinking
  - uniformly think-twice involves more errors
  - think-at-hard solves most cases
    - hard token: tokens that cannot be accurately predicted in a single forward pass

Figure 1: Selective iteration can mitigate latent overthinking. (a) Toy example. Uniform latent iteration (always think-twice) can fix wrong predictions, but may also overthink and corrupt correct ones. (b) Next-token prediction accuracy of finetuned Qwen3-1.7B variants. Always think-twice causes more errors than corrections over direct reply. In contrast, the think-at-hard oracle, which iterates only when the first-pass prediction is wrong, achieves substantial improvements with minimal harm. While this oracle signal is unavailable in practice, it highlights the potential of selective iteration.

Motivation

- Oracle: latent overthinking
  - think-at-hard solves most cases
    - hard token: tokens that cannot be accurately predicted in a single forward pass

Training	Inference	Accuracy
Standard	Standard	52
AlwaysThink	AlwaysThink	38/−14
AlwaysThink	TaH-Oracle	77/+25
TaH-Oracle	TaH-Decider	54/+ 2
TaH-Oracle	TaH-Oracle	80/+28

Table 4: Impact of iteration strategies on Qwen3-0.6B (first 100 MATH500 samples).

Can LLMs selectively dedicate latent iterations only to hard tokens?

key question

Challenges

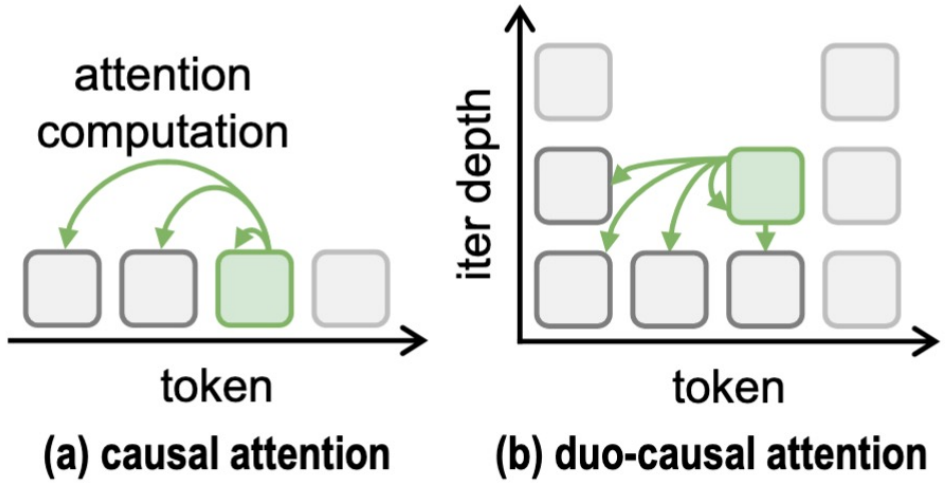
- Model architecture
  - support dynamic iteration depths
    - enable cross-depth information flow
    - while maintaining the parallel computation required for training and prefilling
  - handle the shift in prediction objectives across iterations
    - while maximizing parameter reuse
- Training scheme
  - remain stable despite the tight coupling between components
    - optimal iteration policy depends on the prediction qualities at each depth
    - predictions in turn depend on the policy's iteration choices

Methods

- Model architecture
  - support dynamic iteration depths
    - enable cross-depth information flow
    - while maintaining the parallel computation required for training and prefilling
  - handle the shift in prediction objectives across iterations
    - while maximizing parameter reuse
- Training scheme
  - remain stable despite the tight coupling between components
    - optimal iteration policy depends on the prediction qualities at each depth
    - predictions in turn depend on the policy's iteration choices

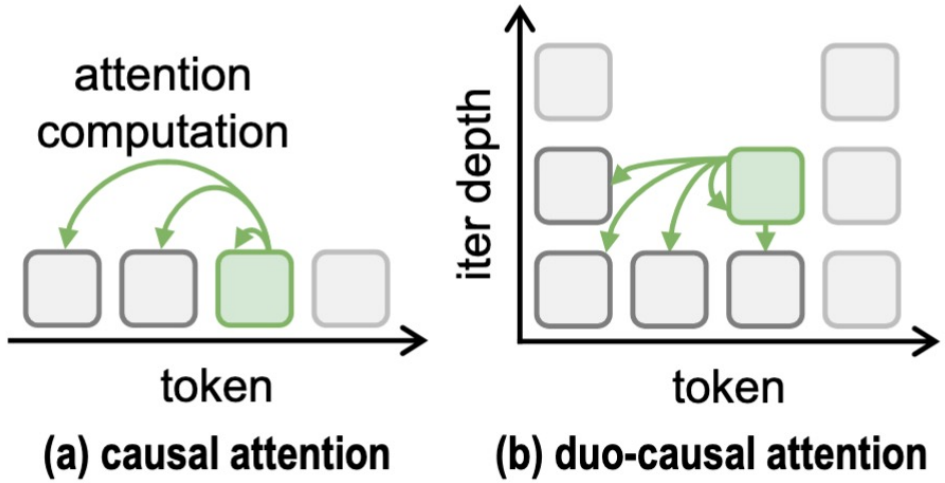
Methods: Model Architecture

- Duo-causal attention
  - motivation
    - previous: casual attention attend within each iteration depth
    - ours: tokens may stop at different depths, limiting deeper tokens to **access contexts stopped at shallower iterations**
    - efficient training requires all tokens at depth *d* be **computable in parallel**,
      - without depending on deeper tokens *d'*>*d* that have not yet been computed.



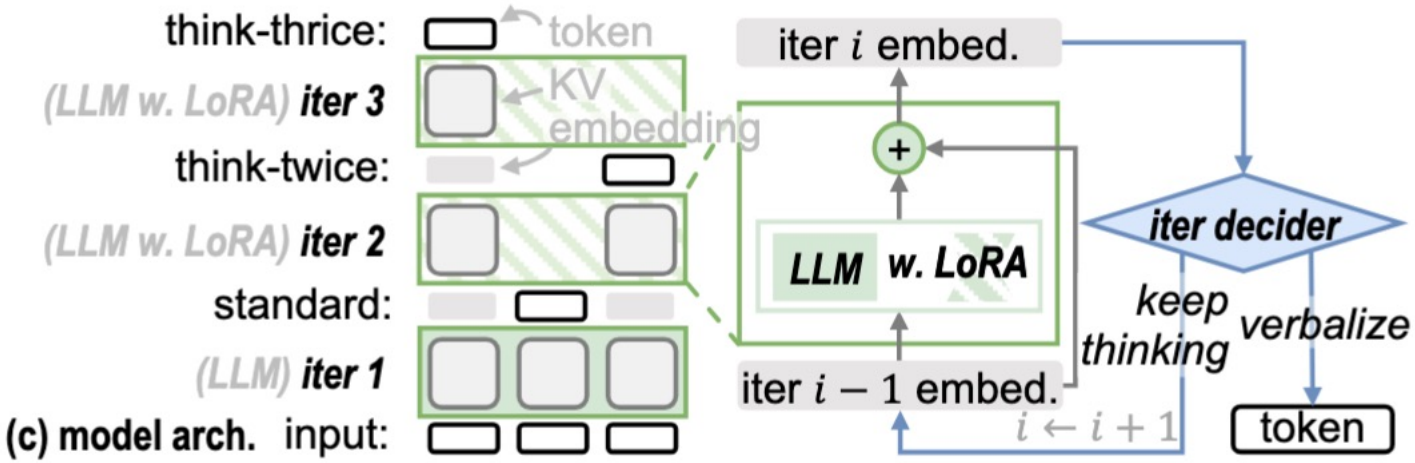
Methods: Model Architecture

- Duo-causal attention
  - solution
    - **extends *causality* to a 2-D space**
      - previous positions
      - shallower iteration depths



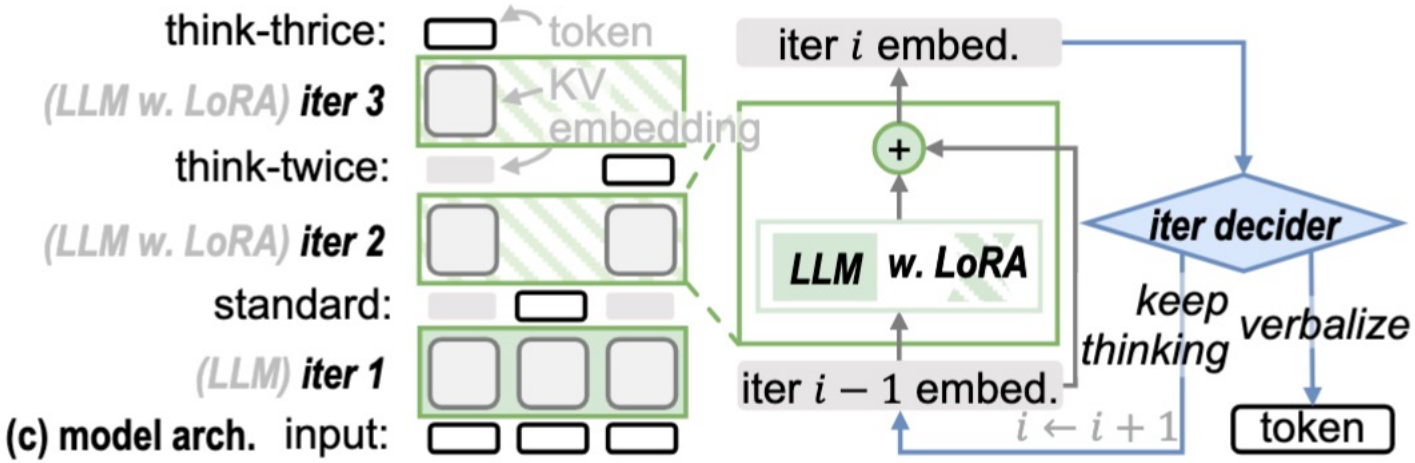
Methods: Model Architecture

- LoRA design
  - motivation
    - previous: use identical weights across all iterations
    - objective: objective shift
      - later iterations shift from next-token prediction to current-token refinement
      - the model **stop and think** rather than predicting next-next token



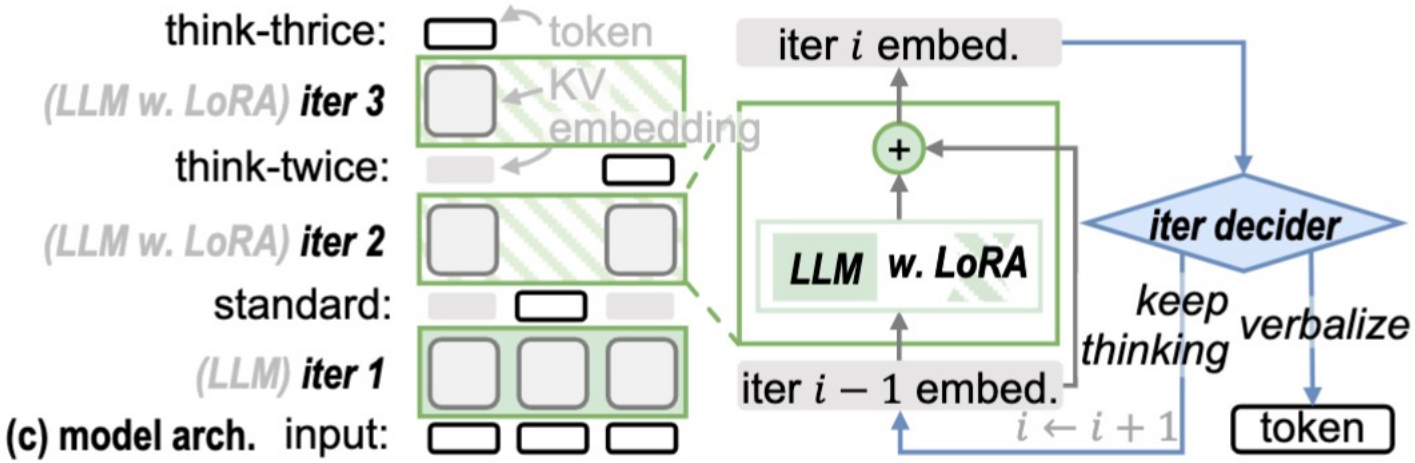
Methods: Model Architecture

- LoRA design
  - solution
    - **LoRA on iterations *d>1***
      - base LLM focus on latent embeddings
      - adapter handles the objective shift
    - residual connection across layers



Methods: Model Architecture

- Iteration decider
  - solution
    - MLP
    - determine whether to continue latent thinking or output the verbalized token after each iteration.
    - take shallow, middle, and final hidden states from the backbone LLM



Methods: Training Scheme

- Training scheme
  - motivation
    - backbone network & neural iteration decider are tightly coupled
      - backbone: generates hidden states as inputs for the decider
      - iteration decider: controls the backbone's KV cache and iterations
    - training both simultaneously causes instability due to mutual distribution shifts

Methods: Training Scheme

- Training scheme
  - solution
    - backbone network & neural iteration decider are sequentially trained to align an oracle iteration policy π
    - oracle iteration policy π
      - determine token hardness using a frozen reference LLM
        - let the reference model correctly predicts next token with a standard forward pass
          - *easy:* correct prediction
          - *hard:* false prediction
      - adopt the supervised fine-tuned (SFT) variant of the base model

Methods: Training Scheme

- Training scheme
  - solution
    - two stage training
      - stage 1: backbone supervision under π
        - optimize the LLM by executing iterations according to π
        - compute the standard next-token prediction loss, but applied at the oracle-determined depth for each token
      - stage 2: decider imitation under frozen backbone
        - freeze the backbone model
        - binary cross-entropy, imitate the continuation labels from the oracle policy

Experiment

- Setup
  - Base model:
    - Qwen3-0.6B-Base
    - Qwen3-1.7B-Base
  - Baselines
    - Standard: always verbalizes directly and reduces to the original Qwen model;
    - AlwaysThink: applies the maximum number of latent iterations to all tokens;
    - SoftThink: latent space thinking
  - Method
    - we prune one layer from the base model so that \name matches the parameter count of baselines. The layer is chosen to minimize the increase in validation loss.
    - we also report results for an unpruned variant, TaH+, which adds less than 3\% extra parameters

Experiment

Table 1: Accuracy comparison of different baselines across five benchmarks and two model sizes. Subscripts indicate improvement over Standard.

Param.	Benchmark	Method					
		Standard	**Routing**	**SoftThink**	**AlwaysThink**	**TaH**	**TaH+**
0.6B	GSM8K	62.5	45.6	61.3	54.6	**64.4**	**68.8**
	MATH500	47.2	27.3	48.8	32.8	**51.2**	**54.2**
	AMC23	23.4	10.9	24.1	20.3	**32.5**	**30.6**
	AIME25	**4.2**	1.0	2.5	1.5	**4.2**	**5.0**
	OlympiadBench	18.8	7.4	19.4	10.2	**23.9**	**24.0**
	Average	31.2	18.5	31.2	23.9	**35.2**/+4.0	**36.5**/+5.3
1.7B	GSM8K	82.1	71.2	79.6	79.3	**84.5**	**85.8**
	MATH500	68.4	60.0	68.8	61.8	**74.4**	**73.0**
	AMC23	42.2	42.2	43.1	42.5	**48.4**	**51.2**
	AIME25	13.3	10.2	12.9	10.0	**17.9**	**14.6**
	OlympiadBench	33.0	30.6	33.4	30.0	**38.8**	**41.2**
	Average	47.8	36.8	47.6	44.7	**52.8**/+5.0	**53.2**/+5.4

Experiment

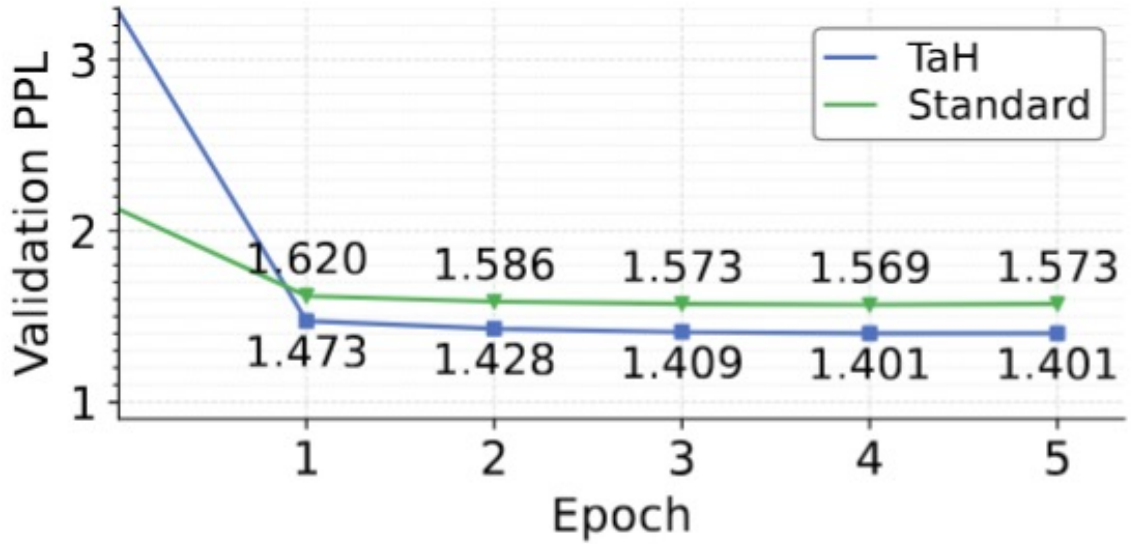


Figure 3: Training dynamics of the LLM backbone on Qwen3-0.6B-Base. TaH converges rapidly and achieves lower perplexity.

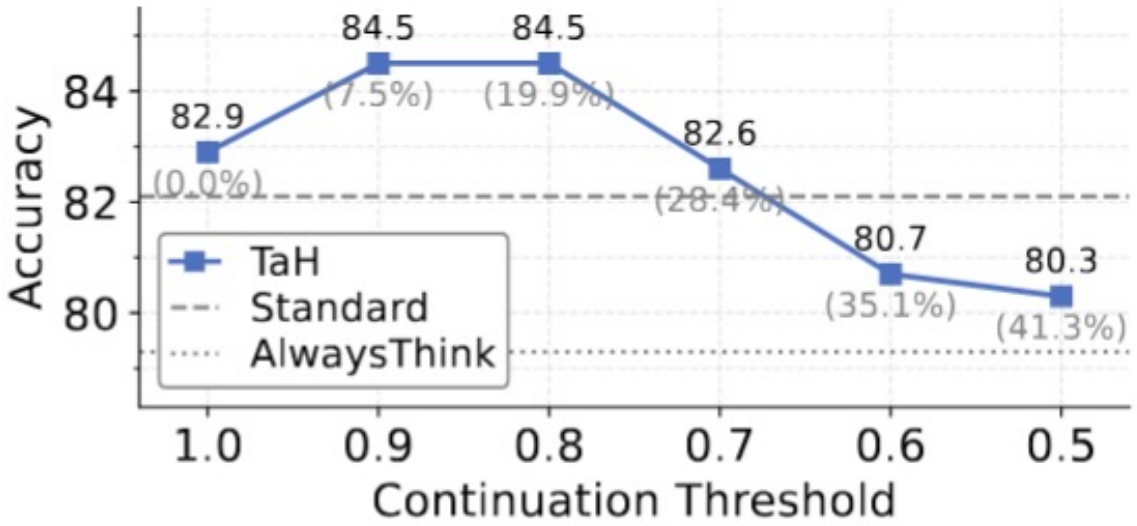


Figure 4: GSM8K accuracy with respect to continuation threshold. Numbers in brackets indicate the percentage of tokens that iterate twice.

Experiment

Table 2: Ablation study on architecture designs.

LoRA	Residual	Attention	MATH500	AMC23	Olympiadbench	Average
✓	✓	Duo-causal	51.2	**32.5**	**23.9**	**35.9**
✗	✓	Duo-causal	**51.6**	29.7	22.4	34.6 /−1.3
✗	✗		49.2	22.5	21.2	31.0 /−4.9
✓	✓	*Causal*	47.8	24.4	19.4	30.5 /−5.4

Table 3: Ablation study on training schemes.

Supervision	Iter. Behavior	MATH500	AMC23	Olympiadbench	Average
Token-only	Oracle	**51.2**	**32.5**	**23.9**	**35.9**
Token+latent	Oracle	49.4	29.6	15.9	31.6 /−4.3
Token-only	*Iter. decider*	44.8	24.1	17.3	28.7 /−7.2
	Dynamic	11.0	2.8	2.7	5.5/−30.4

Experiment

Training	Inference	Accuracy
Standard	Standard	52
AlwaysThink	AlwaysThink	38/−14
AlwaysThink	TaH-Oracle	77/+25
TaH-Oracle	TaH-Decider	54/+ 2
TaH-Oracle	TaH-Oracle	80/+28

Table 4: Impact of iteration strategies on Qwen3-0.6B (first 100 MATH500 samples).

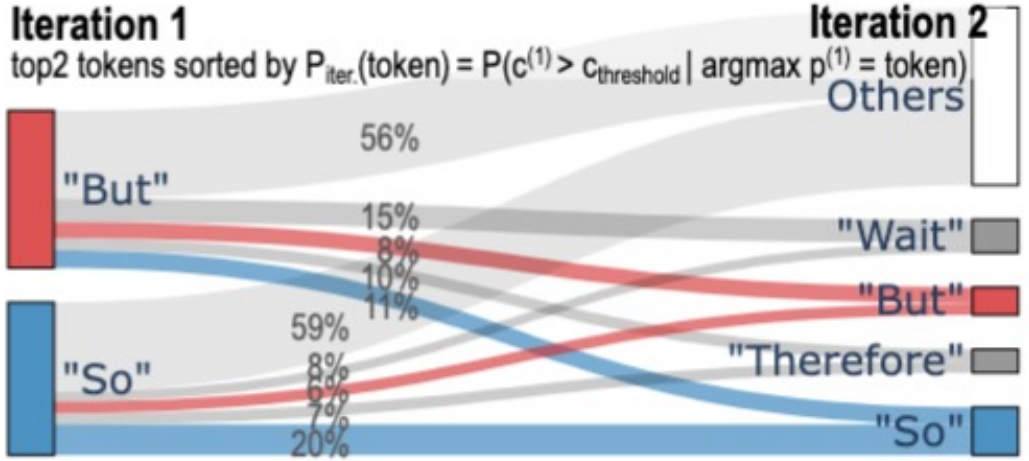


Figure 5: Next-token prediction changes across iterations. Top2 tokens that think-twice most are visualized.

Star If You Like

- thu-nics/
  - C2C
  - TaH
  - R2R
  - MoA
  - FrameFusion

清華大學电子工程系
Department of Electronic Engineering, Tsinghua University

Thanks for Listening

Think-at-Hard: Selective Latent Iterations to Improve Reasoning Language Models