Theoretical Foundations of Chain-of-Thought with Trained Nonlinear Transformer Models

Presenter: Hongkang Li

FCRC Seminar

Authors: Hongkang Li (RPI), Meng Wang (RPI PI), Songtao Lu (IBM PI), Xiaodong Cui (IBM), Pin-Yu Chen (IBM PI)

Future of Computing Research Collaboration at Rensselaer

Chain-of-Thought (CoT) in Large Language Model (LLM)

Chain-of-Thought (CoT) enables the reasoning ability of LLM by augmenting the query using multiple examples with intermediate steps (few-shot) or necessary instructions (zero-shot).

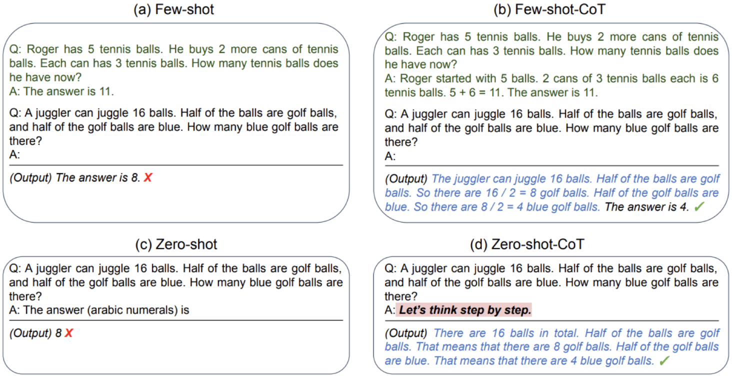
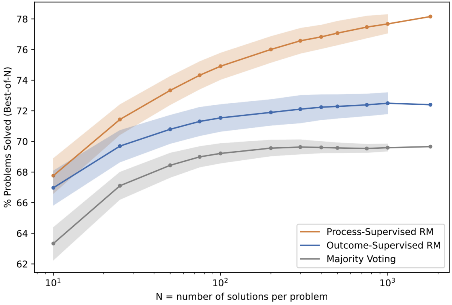


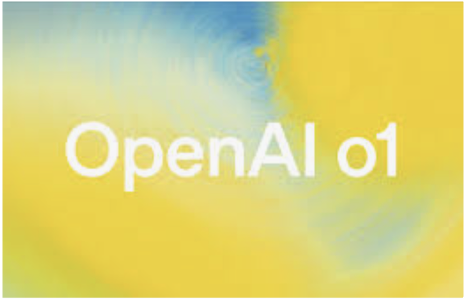
Figure 1: Few-shot and zero-shot CoT [Kojima et al.22]

Chain-of-Thought (CoT) in Large Language Model (LLM)

Besides testing, high-quality CoT data is also beneficial in LLM training.

- Process supervision [Lightman et al.24] with CoT data outperforms outcome supervision.
- CoT+RL is crucial for training GPT-o1.





Our focus

Despite the empirical success of Chain-of-Thought, one fundamental and theoretical question remains less explored, i.e.,

Why can a Transformer be trained to generalize on multi-step reasoning tasks via CoT?

Related Theoretical Works and Background

Existing works focus on the expressive power of Transformer in implementing CoT.

[Li et el.23]:

- Transformers can learn MLP functions by CoT.
- CoT=filtering + In-Context Learning (ICL). Filtering: attends to the relevant token. ICL: Use the filtered tokens to generate the output.

[Feng et al.23, Li et al.24]:

- Without CoT, Transformers can only solve limited problems unless the model size grows super-polynomially w.r.t. the sequence length.
- Transformers of constant size can be constructed to solve arithmetic/equation/Dynamic Programming tasks via CoT.

Related Theoretical Works and Background

Only a concurrent work, [Wen et al.25], provides the convergence and sample complexity analysis of zero-shot CoT using Transformers for **sparse parity** tasks: Given a length-n binary sequence, produce the XOR output of k selected entries.

- Learning using one-layer Transformers requires exponential to k samples without CoT when the number of parameters is limited.
- One-layer Transformers can provably learn sparse parity by CoT with almost linear samples with respect to k and n.

$$\text{No CoT} \quad \underbrace{0,1,0,1,0}_{\text{input}},\ \underbrace{0}_{\text{[EOS]}},\ \underbrace{0}_{\text{answer}} \in \{0,1\}^7, \text{ as } b_1 \oplus b_2 \oplus b_4 = 0 \oplus 1 \oplus 1 = 0.$$

$$\text{With CoT} \quad \underbrace{0,1,0,1,0}_{\text{input}},\ \underbrace{0}_{\text{[EOS]}},\ \underbrace{0,1}_{\text{CoT}},\ \underbrace{0}_{\text{answer}} \in \{0,1\}^9, \text{as } b_1 = 0, b_1 \oplus b_2 = 1.$$

Figure 2: *CoT for the sparse parity problem [Wen et al.25]*

Our work and major contributions

Our recent work "Training Nonlinear Transformers for Chain-of-Thought Inference: A Theoretical Generalization Analysis"[1] accepted by ICLR 2025 has the following contributions.

- A quantitative analysis of how the training can enable the CoT ability with nonlinear Transformers.
- A quantitative analysis of how context examples affect CoT performance.
- A theoretical characterization of when and why CoT outperforms ICL.

[1] https://openreview.net/forum?id=n7n8McETXw

Problem formulation

Consider learning K-steps reasoning tasks $f = f_K \circ f_{K-1} \circ \cdots \circ f_2 \circ f_1$ with few-shot CoT.
$\boldsymbol{P} = (\boldsymbol{E}_1, \boldsymbol{E}_2, \cdots, \boldsymbol{E}_{l_{tr}}, \boldsymbol{Q}_k)$ as the training prompt, where $\boldsymbol{E}_i = \begin{pmatrix} \boldsymbol{x}_i & \boldsymbol{y}_{i,1} & \cdots & \boldsymbol{y}_{i,K-1} \\ \boldsymbol{y}_{i,1} & \boldsymbol{y}_{i,2} & \cdots & \boldsymbol{y}_{i,K} \end{pmatrix}$ is the i-th context example, $\boldsymbol{Q}_k = \begin{pmatrix} \boldsymbol{z}_0 & \boldsymbol{z}_1 & \cdots & \boldsymbol{z}_{k-2} & \boldsymbol{z}_{k-1} \\ \boldsymbol{z}_1 & \boldsymbol{z}_2 & \cdots & \boldsymbol{z}_{k-1} & 0 \end{pmatrix}$ is the first k steps of the reasoning query for any k in $[K]$.

- The label for prediction is $\boldsymbol{z}_k$ for the k-th step, denoted as $\boldsymbol{z}$ for $\boldsymbol{P}$.
- Denote each column of $\boldsymbol{P}$ as $\boldsymbol{p}_i$. Add the positional encoding $\boldsymbol{c}_i$ (periodic) to each $\boldsymbol{p}_i$ to obtain $\tilde{\boldsymbol{p}}_i = \boldsymbol{p}_i + \boldsymbol{c}_{(i \mod K)}$.

Problem formulation

Learning model with $\Psi = \{\boldsymbol{W}_K, \boldsymbol{W}_Q, \boldsymbol{W}_V\}$:

$$f(\Psi; \boldsymbol{P}) = \sum_{i=1}^{\text{len}(P)-1} \boldsymbol{W}_V \tilde{\boldsymbol{p}}_i \text{softmax}((\boldsymbol{W}_K \tilde{\boldsymbol{p}}_i)^\top \boldsymbol{W}_Q \tilde{\boldsymbol{p}}_{query}). \tag{1}$$

Model Training: Following theoretical works [Li et el.23, Feng et al.23, Li et al.24], we use CoT data for model training. Given the training set $\{\boldsymbol{P}^n, z^n\}_{n=1}^N$, we train the model with empirical risk minimization.

$$\min_{\Psi} R_N(\Psi) := \frac{1}{N} \sum_{n=1}^{N} \ell(\Psi; \boldsymbol{P}^n, z^n) \tag{2}$$

- The query and context inputs are sampled from a distribution $\mathcal{D}$.
- The task f^n is sampled from a distribution $\mathcal{T}$. The training tasks form a set $\mathcal{T}_{tr} \subset \mathcal{T}$.
- $\ell(\Psi; \boldsymbol{P}^n, z^n) = 1/2 \cdot \|z^n - f(\Psi; \boldsymbol{p}^n)\|^2$ is the squared loss.
- The model is trained via stochastic gradient descent (SGD).

Problem formulation

Chain-of-Thought Inference:

The testing prompt $\boldsymbol{P} = (\boldsymbol{E}_1, \boldsymbol{E}_2, \cdots, \boldsymbol{E}_{l_{ts}}, \boldsymbol{p}_{query})$, where $\boldsymbol{p}_{query} = \begin{pmatrix} \boldsymbol{x}_{query} \\ 0 \end{pmatrix}$. Predict $\{\boldsymbol{z}_k\}_{k=1}^{K}$.

Let $\boldsymbol{P}_1 = \boldsymbol{P}$, $\boldsymbol{P}_0$ be the columns of $\boldsymbol{P}$ before the query. In the k-th step inference,

- Generate $\boldsymbol{v}_k$ as the most probable output from the set $\mathcal{Y}$ of all possible outputs (greedy decoding):

$$\boldsymbol{v}_k = \arg\min_{\boldsymbol{u}\in\mathcal{Y}} \frac{1}{2}\|F(\Psi; \boldsymbol{P}) - \boldsymbol{u}\|^2. \tag{3}$$

- Use $\boldsymbol{v}_k$ to update $\boldsymbol{P}_k$ and construct $\boldsymbol{P}_{k+1}$:

$$\boldsymbol{P}_k = (\boldsymbol{P}_{k-1} \begin{pmatrix} \boldsymbol{v}_{k-1} \\ \boldsymbol{v}_k \end{pmatrix}), \boldsymbol{P}_{k+1} = (\boldsymbol{P}_k \begin{pmatrix} \boldsymbol{v}_k \\ 0 \end{pmatrix}) \tag{4}$$

The CoT generalization error on test data distribution $\mathcal{D}'$ and test task $f \in \mathcal{T}'$:
$R_{CoT,\boldsymbol{x}_{query}\sim\mathcal{D}',f\in\mathcal{T}'}(\Psi) = \mathbb{E}_{\boldsymbol{x}_{query}\sim\mathcal{D}'}[\frac{1}{K}\sum_{k=1}^{K} \mathbb{1}[\boldsymbol{z}_k = \boldsymbol{v}_k]]$.

In-Context Learning Inference:

The testing prompt $\boldsymbol{P}=(\boldsymbol{E}_1,\boldsymbol{E}_2,\cdots,\boldsymbol{E}_{l_{ts}},\boldsymbol{p}_{query})$, where $\boldsymbol{p}_{query}=\begin{pmatrix}\boldsymbol{x}_{query}\\0\end{pmatrix}$, and $\boldsymbol{E}_i=\begin{pmatrix}\boldsymbol{x}_i & 0 & \cdots & 0\\ \boldsymbol{y}_{i,K} & 0 & \cdots & 0\end{pmatrix}$ is the i-th context example. Predict $\boldsymbol{z}_K$.

Generate the output $\boldsymbol{v}=\arg\min_{\boldsymbol{u}\in\mathcal{Y}}\frac{1}{2}\|F(\Psi;\boldsymbol{P})-\boldsymbol{u}\|^2$.

The ICL generalization error on test data distribution $\mathcal{D}'$ and test task $f\in\mathcal{T}'$:
$R_{ICL,\boldsymbol{x}_{query}\sim\mathcal{D}',f\in\mathcal{T}'}(\Psi)=\mathbb{E}_{\boldsymbol{x}_{query}\sim\mathcal{D}'}[\mathbb{1}[\boldsymbol{z}_K=\boldsymbol{v}]]$.

Data modeling

The training tasks are the transition between M orthogonal training-relevant (TRR) patterns $\{\boldsymbol{\mu}_i\}_{i=1}^{M}$. The testing tasks are the transition between M' testing-relevant (TSR) patterns $\{\boldsymbol{\mu}'_i\}_{i=1}^{M'}$.

For a task $f = f_K \circ f_{K-1} \circ \cdots \circ f_2 \circ f_1$,

- The k-th step label of $\boldsymbol{x}_{query}$ is $\boldsymbol{z}_k = f_k(\boldsymbol{z}_{k-1})$, $\boldsymbol{z}_0 = \boldsymbol{x}_{query}$.
- The k-th step label of the i-th context example $\boldsymbol{x}_i$ is $\boldsymbol{y}_{i,k} = f_k(\boldsymbol{y}_{i,k-1})$, $\boldsymbol{y}_{i,0} = \boldsymbol{x}_i$.

Positional encodings are orthogonal to TRR/TSR patterns.

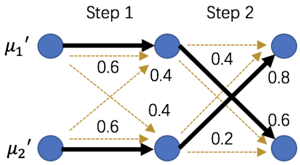
Example: Task of "Country$\rightarrow$ Capital $\rightarrow$ President": $\boldsymbol{E}_1$=(USA, Washington DC, Trump). $\boldsymbol{p}_{query}$=France. $\boldsymbol{P} = (\boldsymbol{E}_1, \boldsymbol{p}_{query})$. Then, the label $\boldsymbol{z}_1$=Paris, $\boldsymbol{z}_2$=Macron. Each word corresponds to a certain pattern $\boldsymbol{\mu}_i$.

Data modeling

Training: All $\boldsymbol{z}_k$, $\boldsymbol{y}_{i,k}$ are chosen from TRR patterns $\{\boldsymbol{\mu}_i\}_{i=1}^{M}$.
Testing: All $\boldsymbol{z}_k$, $\boldsymbol{y}_{i,k}$ are chosen from TSR patterns $\{\boldsymbol{\mu}'_i\}_{i=1}^{M'}$ plus a bounded noise. Testing examples may contain erroneous steps, and we use transition matrices to characterize the error.

- Step-wise transition matrix: $\boldsymbol{A}_k^f$ is the transition matrix in the k-th step of the task f. The correct output must be the most probable output by $\boldsymbol{A}^f$.
- K-steps transition matrix: $\boldsymbol{B}^f = \prod_{k=1}^{K} \boldsymbol{A}_k^f$.

Example: correct paths are $\boldsymbol{\mu}'_1 \to \boldsymbol{\mu}'_1 \to \boldsymbol{\mu}'_2$, $\boldsymbol{\mu}'_2 \to \boldsymbol{\mu}'_2 \to \boldsymbol{\mu}'_1$. Testing prompts may contain incorrect paths like $\boldsymbol{\mu}'_1 \to \boldsymbol{\mu}'_1 \to \boldsymbol{\mu}'_1$. Step-wise transition matrices: $\boldsymbol{A}_1^f = \begin{pmatrix} 0.6 & 0.4 \\ 0.4 & 0.6 \end{pmatrix}$, $\boldsymbol{A}_2^f = \begin{pmatrix} 0.4 & 0.6 \\ 0.8 & 0.2 \end{pmatrix}$. K-steps transition matrix: $\boldsymbol{B}^f = \begin{pmatrix} 0.56 & 0.44 \\ 0.64 & 0.36 \end{pmatrix}$.

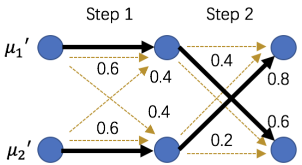


Data modeling

Important quantities:

- The **primacy** ρ^f **for** $\boldsymbol{A}^f$ **and** ρ_o^f **for** $\boldsymbol{B}^f$: The gap between the correct reasoning and incorrect reasoning of each step.
- **Min-max trajectory transition probability** τ^f: The minimum probability of the most probable K-step reasoning trajectory over the initial TSR pattern.
- **Min-max input-label transition probability** τ_o^f: The minimum probability of the most probable output over the initial TSR pattern.

Example: $\boldsymbol{A}_1^f = \begin{pmatrix} 0.6 & 0.4 \\ 0.4 & 0.6 \end{pmatrix}$, $\boldsymbol{A}_2^f = \begin{pmatrix} 0.4 & 0.6 \\ 0.8 & 0.2 \end{pmatrix}$. $\boldsymbol{B}^f = \begin{pmatrix} 0.56 & 0.44 \\ 0.64 & 0.36 \end{pmatrix}$. Then, $\tau^f = \min\{0.6 \cdot 0.6, 0.6 \cdot 0.8\} = 0.36$, $\tau_o^f = \min\{0.56, 0.64\} = 0.56$.



Theoretical Results

Define α and α' as the fraction of context examples with input sharing the same TRR or TSR pattern as the query input, respectively.

Theorem 1

For any $\epsilon > 0$, as long as

1. *the training tasks and samples are selected such that every TRR pattern is equally likely in every inference step and in each training batch,*
2. *the number of examples in training prompts $l_{tr} \geq \Omega(\alpha^{-1})$*
3. *and the number of iterations $T = \Theta(\alpha^{-2}K^3 + MK(\alpha^{-1} + \epsilon^{-1}))$,*

and the batch size $B \geq \Omega(\epsilon^{-2})$, then with a high probability, the loss of the returned model is less than $\mathcal{O}(\epsilon)$.

- The required number of context examples is proportional to α^{-1}.
- The required numbers of iterations and samples increases as M, K^3 and α^{-2} increase.

Theoretical Results

Theorem 2 (CoT generalization)

With a model trained as in Theorem 1, as long as

1. *each TSR pattern $\boldsymbol{\mu}_i'$ contains a non-trivial component in the span of the TRR pattern $\boldsymbol{\mu}_i$,*
2. *the number of examples in testing prompts $l_{ts} \geq \Omega((\alpha' \tau^f \rho^f)^{-2})$,*

then with a high probability, we have the CoT generalization error $= 0$.

- Generalization with out-of-domain patterns can be successful if the testing data has a strong correlation with training data.
- A more informative prompt (larger α') and more accurate inference examples (larger τ^f and ρ^f) can reduce the required testing prompt length.

Theoretical Results

Comparison with ICL:

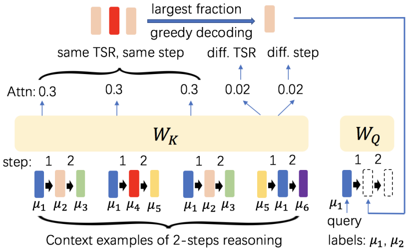
We first propose Condition 1: the correct final output is the most probable final output by $\boldsymbol{B}^f$. The previous condition does not satisfy this condition.

Theorem 3 (ICL generalization)

1. *If condition 1 does not hold, then the ICL generalization error* $\geq \Omega(1)$.
2. *If condition 1 holds, and* $l_{ts} \geq \Omega((\alpha' \tau_o^f \rho_o^f)^{-2})$, *we have the ICL generalization error* $= 0$.

Because Condition 1 is not required for CoT generalization, CoT performs better than ICL if Condition 1 fails.

The Mechanism of CoT



1. When conducting the k-th step reasoning of the query, the trained model assigns dominant attention weights on the prompt columns that are also the k-th step and share the same TSR pattern as the query.
2. Then, the fraction of the correct TSR pattern is the largest in the output of each step to generate the accurate output by greedy decoding.

Experiments

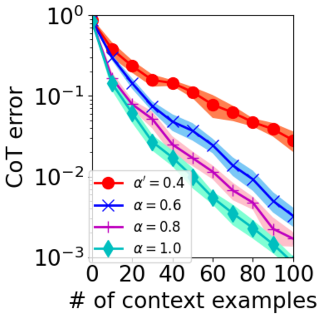


Figure 3: ***CoT testing error with different α'.***

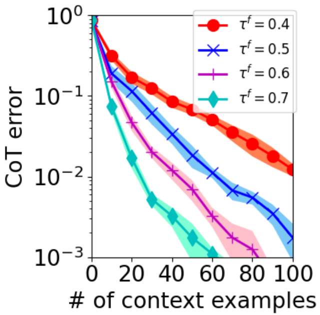


Figure 4: ***CoT testing error with different τ^f.***

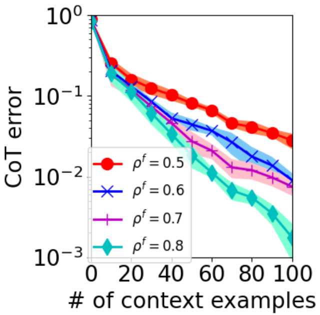


Figure 5: ***CoT testing error with different ρ^f.***

More CoT testing examples are needed when α', τ^f, or ρ^f is small.

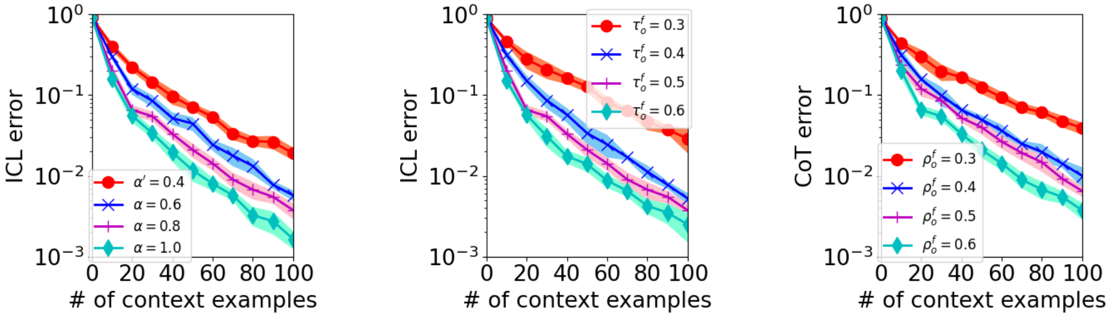


Figure 6: ICL testing error with different α'.

Figure 7: ICL testing error with different τ_o^f.

Figure 8: ICL testing error with different ρ_o^f.

More ICL testing examples are needed when α', τ_o^f, or ρ_o^f is small.

Experiments

- Whether increasing the number of context examples can improve the ICL performance depends on whether Condition 1 holds, while CoT does not require this.
- Two-stage training dynamics: 1, learn position information. 2, learn pattern information.

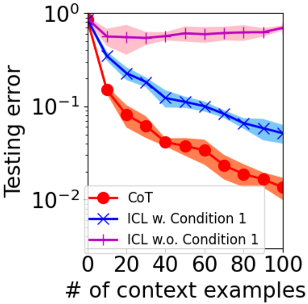


Figure 9: *Comparison between CoT and ICL w./w.o. Condition 1*

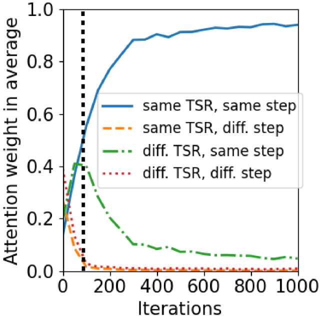


Figure 10: *Mechanism of Transformers for CoT*

There exists at least one head in each layer of the Transformer that implements CoT as the characterized mechanism.

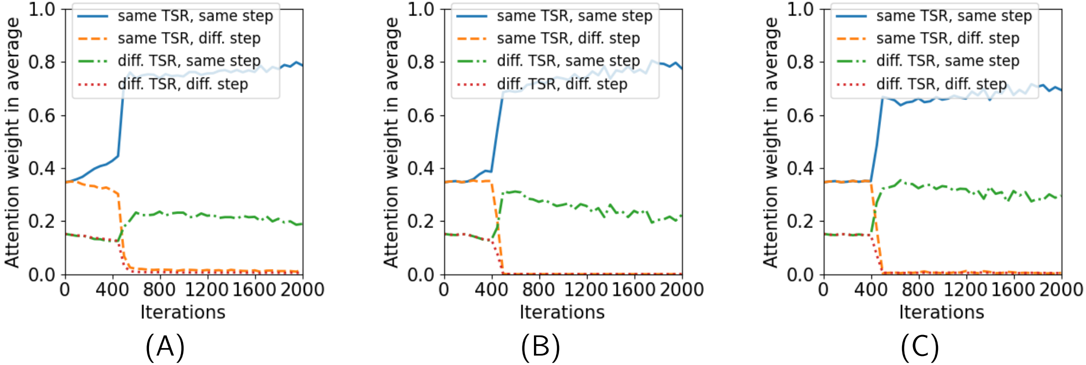


Figure 11: Training dynamics of Transformers. (A) Layer 1, Head 2 (B) Layer 2 Head 2 (C) Layer 3 Head 2.

Summary

- This work provides the training dynamics analysis of nonlinear Transformer towards CoT generalization on new tasks with noisy and partially inaccurate context examples.
- This work also characterizes the requirements for a guaranteed CoT generalization with a provable mechanism.
- This work theoretically studies when CoT is better than ICL.

Future Directions

- Can the analysis of Chain-of-Thought or In-Context Learning be extended to non-Transformer models, e.g., Mamba?
- How can Transformers reason with limited Chain-of-Thought data?

Future Direction I: Mamba

- Mamba can be represented as linear attention plus nonlinear gating.

$$
\begin{aligned}
F(\Psi; \boldsymbol{P}) &= \sum_{i=1}^{l} G(i, l; \boldsymbol{w}) \boldsymbol{x}_l \boldsymbol{W}_C \boldsymbol{W}_B \boldsymbol{x}_i^\top \boldsymbol{x}_i, \\
\text{where } G(i, l; \boldsymbol{w}) &= \begin{cases} \prod_{j=i+1}^{l} (1 - \sigma(\boldsymbol{x}_j \boldsymbol{w})) \sigma(\boldsymbol{x}_i \boldsymbol{w}), & i < l \\ \sigma(\boldsymbol{x}_l \boldsymbol{w}), & i = l. \end{cases}
\end{aligned} \tag{5}
$$

- Existing work [Park et al.24] shows that Mamba works better than Transformers in ignoring fixed and prevalent outliers and parity problems by ICL.

Problem formulation: Consider a fixed backdoor attack $\boldsymbol{v}_*$ that is possibly added to the context inputs to generate a certain binary label. Train with backdoor-attacked examples. Compare Mamba and Transformers.

Future Direction I: Mamba

- Our work plans to theoretically study the optimization dynamics and generalization of Mamba compared with Transformers in ICL.
- Mamba can be provably trained to defend the backdoor attack vector $\boldsymbol{v}_*$ better than Transformers due to the existence of the nonlinear gating.
- The mechanism of Mamba is to attend to tokens that share the same relevant pattern without the attack vector as the query and are located close to the query.

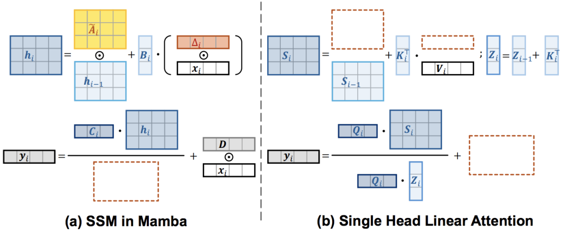


Figure 12: Mamba=linear Transformer+nonlinear gating.

Future Direction II: Self-Taught

- Recall CoT data is widely used in LLM training to improve performance.
- We usually do not have enough CoT data with intermediate steps for training.
- Self-Taught: Given $D = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^N$, use the pretrained model M to generate intermediate steps $\boldsymbol{r}_i$ to form CoT data $D' = \{\boldsymbol{x}_i, \boldsymbol{r}_i, \boldsymbol{y}_i\}_{i=1}^N$. Then, use D' to fine-tune M.

This work plans to theoretically study
(a) whether the model M can generate correct intermediate steps;
(b) the generalization ability of learning with generated CoT data.

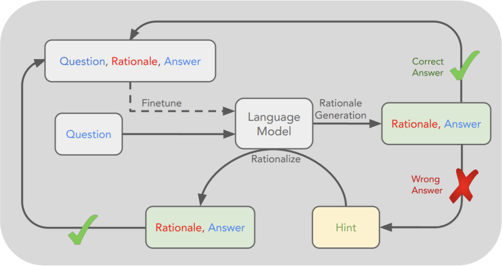


Figure 13: *Self-Taught. Figure from [Zelikman et al.23]*

Thank you!

Q & A

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, Yusuke Iwasawa
Large Language Models are Zero-Shot Reasoners
In *Neurips 2022.*

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever & Karl Cobbe
Let's verify step by step
In *International conference on Learning Representations 2024.*

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter et al.
Chain-of-Thought Prompting Elicits Reasoning in Large Language Models
In *Neurips 2022.*

Yingcong Li, Kartik Sreenivasan, Angeliki Giannou, Dimitris Papailiopoulos, Samet Oymak
Dissecting Chain-of-Thought: Compositionality through In-Context Filtering and Learning
In *Neurips 2023.*

Zhiyuan Li, Hong Liu, Denny Zhou, Tengyu Ma
Chain of Thought Empowers Transformers to Solve Inherently Serial Problems

In *International conference on Learning Representations 2024.*

Kaiyue Wen, Huaqing Zhang, Hongzhou Lin, Jingzhao Zhang
From Sparse Dependence to Sparse Attention: Unveiling How Chain-of-Thought Enhances Transformer Sample Efficiency
In *International conference on Learning Representations 2025.*

Guhao Feng, Bohang Zhang, Yuntian Gu, Haotian Ye, Di He, Liwei Wang
Towards Revealing the Mystery behind Chain of Thought: A Theoretical Perspective
In *Neurips 2023.*

Kai Yang, Jan Ackermann, Zhenyu He, Guhao Feng, Bohang Zhang, Yunzhen Feng, Qiwei Ye, Di He, and Liwei Wang.
Do efficient transformers really save computation?
https://arxiv.org/pdf/2402.13934.pdf

Jongho Park, Jaeseung Park, Zheyang Xiong, Nayoung Lee, Jaewoong Cho, Samet Oymak, Kangwook Lee, Dimitris Papailiopoulos.
Can Mamba Learn How to Learn? A Comparative Study on In-Context Learning Tasks.

In *International conference on Machine Learning 2024.*

Eric Zelikman, Yuhuai Wu, Jesse Mu, Noah D. Goodman
STaR: Self-Taught Reasoner Bootstrapping Reasoning With Reasoning.
In *Neurips 2022.*