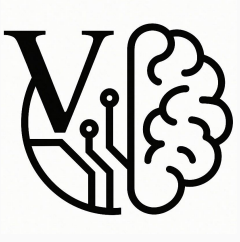
Towards understanding multi-step reasoning: Algorithmic and graph neural network perspectives

---

Hongyang R. Zhang

Assistant Professor, Northeastern University
Khoury College of Computer Sciences
Labpage: https://virtuosoresearch.github.io/

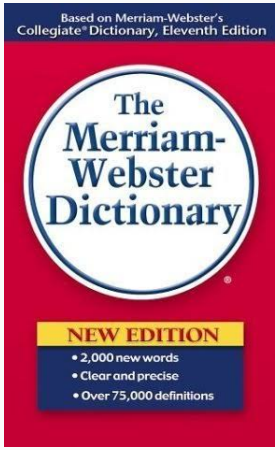


September 30, 2026

What is reasoning?

According to the Merriam-Webster dictionary, reasoning is defined as the drawing of inferences or conclusions through the use of reason. Two examples of reasoning used in a sentence:

- 1st example: Could you explain your reasoning?
- 2nd example: They told everyone the reasoning behind the decision.



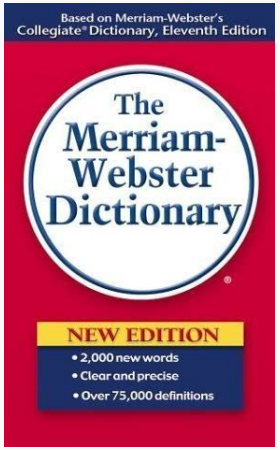
For the scope of today's talk, I shall refer to reasoning as the ability to solve a complex problem by decomposing this problem into sub-problems

- Why are we interested in this question? The ability of LLMs to solve math problems is remarkable.

What is reasoning?

According to the Merriam-Webster dictionary, reasoning is defined as the drawing of inferences or conclusions through the use of reason. Two examples of reasoning used in a sentence:

- 1st example: Could you explain your reasoning?
- 2nd example: They told everyone the reasoning behind the decision.



For the scope of today's talk, I shall refer to reasoning as the ability to solve a complex problem by decomposing this problem into sub-problems

- Why are we interested in this question? The ability of LLMs to solve math problems is remarkable.

Outline of today's talk

Specifically, I will talk about our recent work covering three directions:

Multitask algorithmic reasoning

A stability analysis of multi-step reasoning via error amplification

Grokking in modular arithmetic reasoning

Outline of today's talk

Specifically, I will talk about our recent work covering three directions:

Multitask algorithmic reasoning

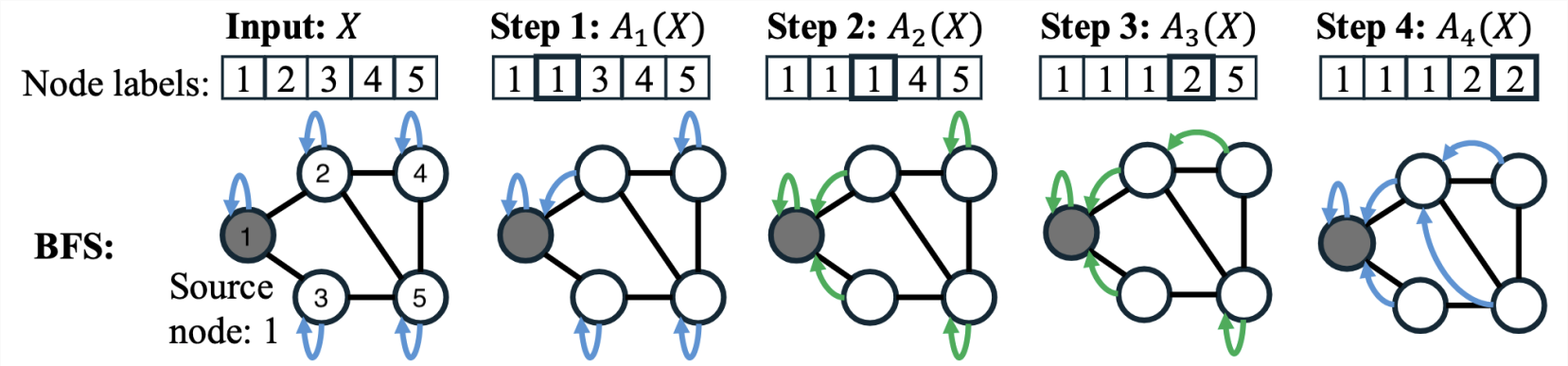
A stability analysis of multi-step reasoning via error amplification

Grokking in modular arithmetic reasoning

- First, I will talk about a synthetic benchmark to evaluate reasoning, since we know the correct reasoning process at every single step of an algorithm
- Once I set up the empirical framework, I will then provide a formal theoretical analysis of multi-step reasoning
- Lastly (time permitting), I will talk about grokking — a delayed generalization phenomenon — in modular arithmetic reasoning tasks

What is algorithmic reasoning?

Here is an example corresponding to encoding the intermediate steps and the final answer of breadth-first search (BFS) on a simple graph

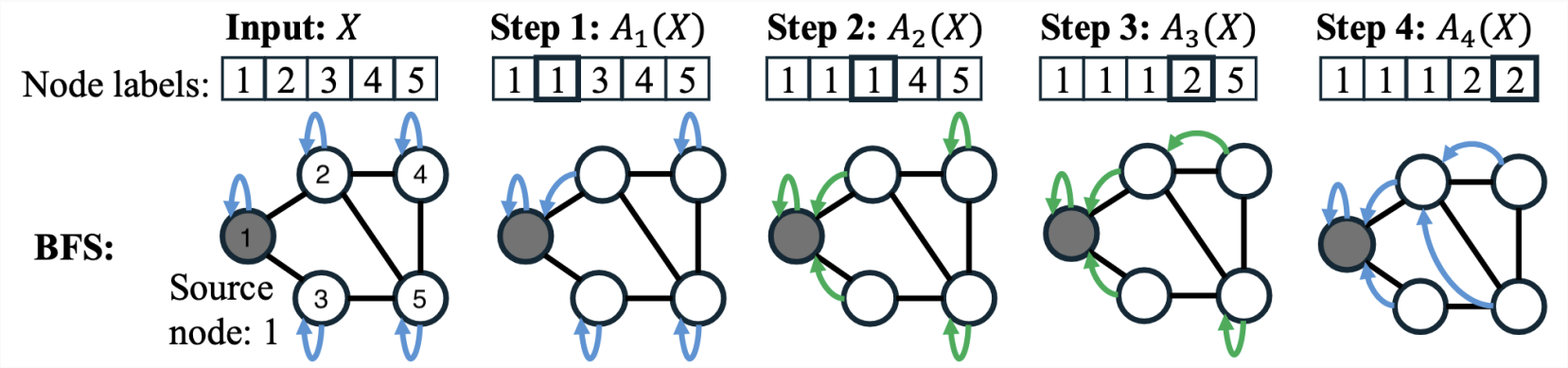


- Starting from node 1, the first step visits node 2. Hence, we update node 2's label to 1, its predecessor.
- Step 2 visits node 3, hence its node label is updated as 1.
- Step 3 visits node 4 (with predecessor 2) and Step 4 visits node 5 (with predecessor 2).

Question: Can a neural network learn to perform an algorithmic reasoning procedure?

What is algorithmic reasoning?

Here is an example corresponding to encoding the intermediate steps and the final answer of breadth-first search (BFS) on a simple graph

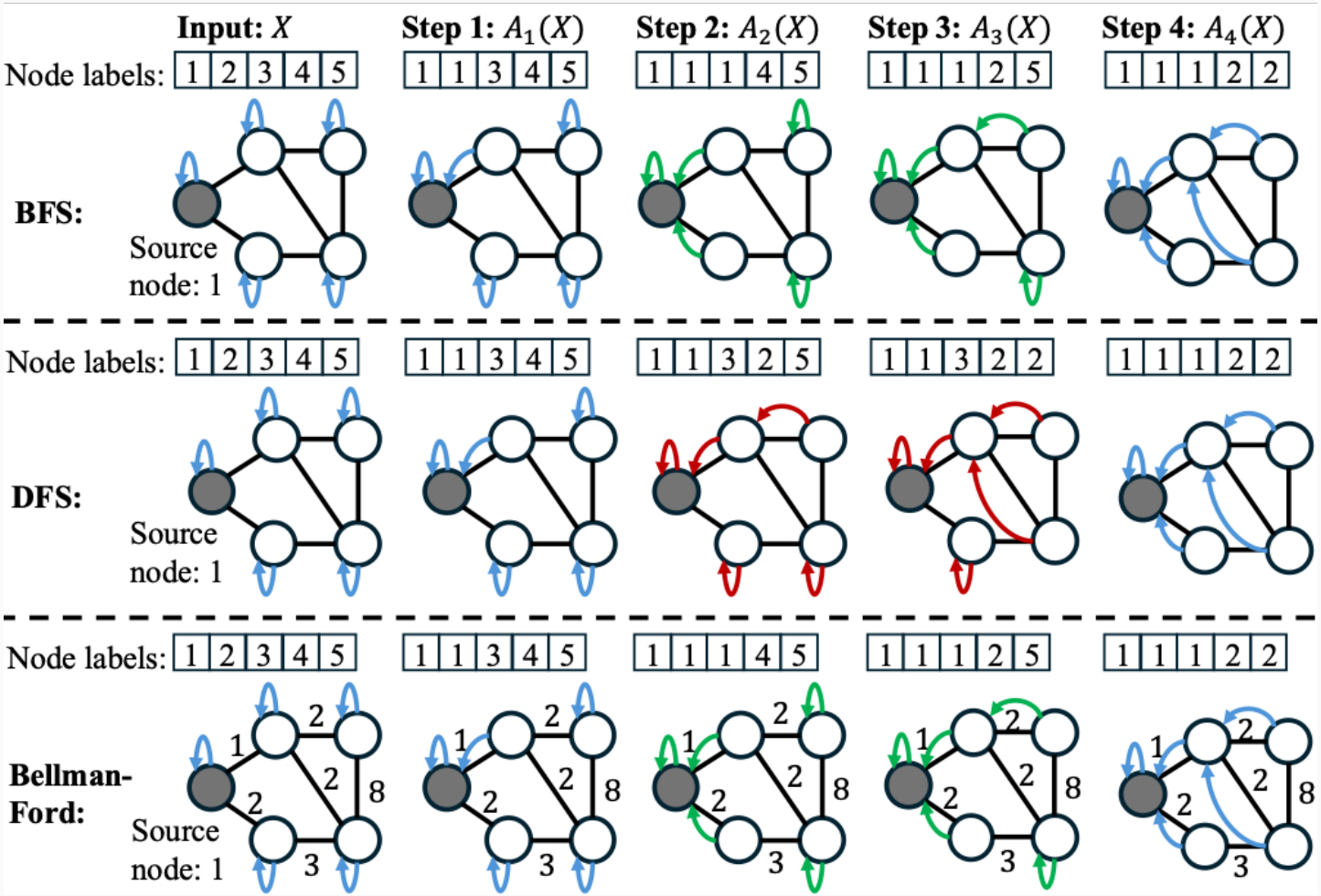


- Starting from node 1, the first step visits node 2. Hence, we update node 2's label to 1, its predecessor.
- Step 2 visits node 3, hence its node label is updated as 1.
- Step 3 visits node 4 (with predecessor 2) and Step 4 visits node 5 (with predecessor 2).

Question: Can a neural network learn to perform an algorithmic reasoning procedure?

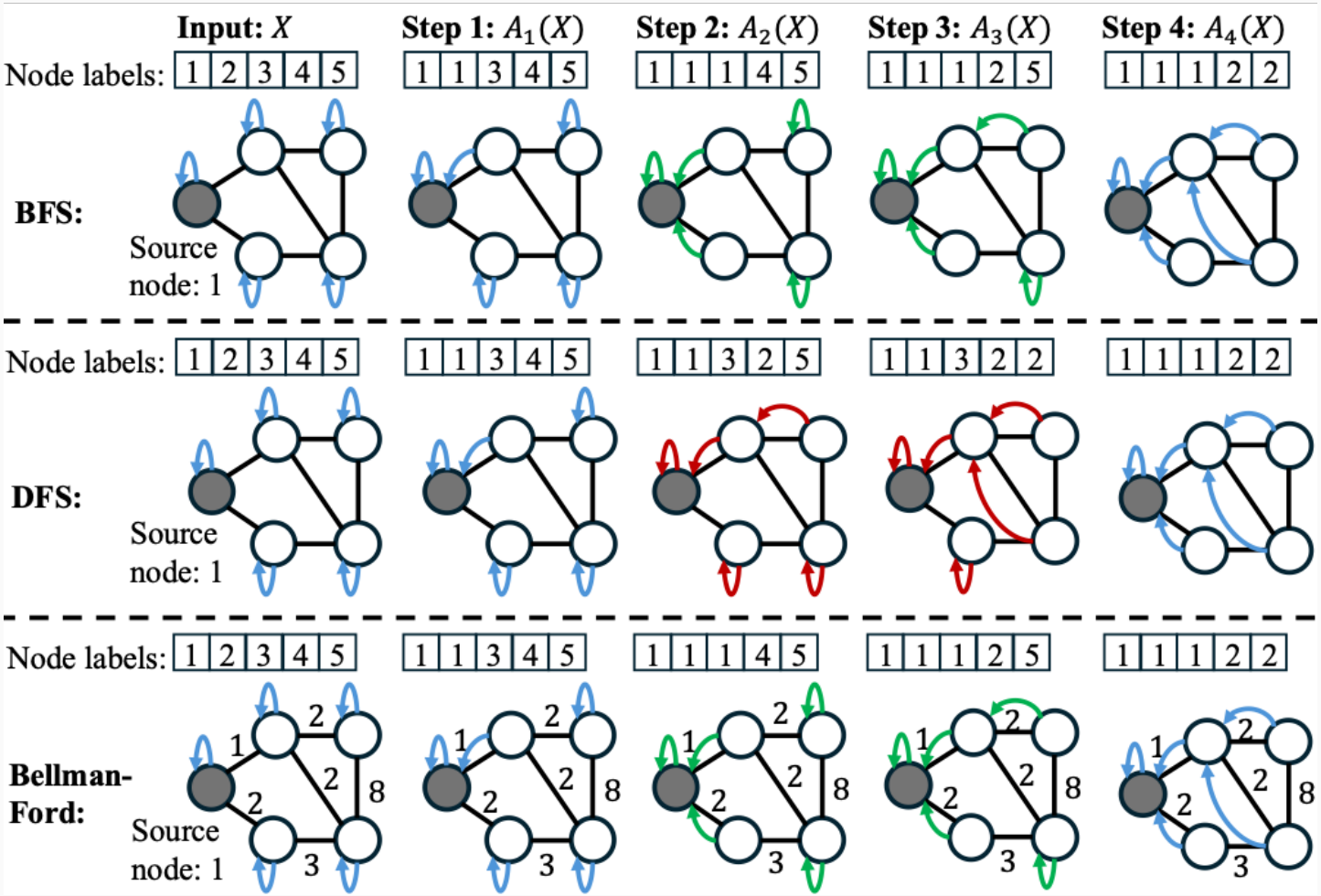
Multitask algorithmic reasoning

Even more ambitiously, can a neural network learn to perform not just one, but multiple algorithmic reasoning tasks simultaneously?



Labeling conflicts in multitask algorithmic reasoning

For example, how about breadth-first search (BFS), depth-first search (DFS), and Bellman-Ford? Note that these algorithms have different encodings in the intermediate steps (indicated by colors)



Problem formulation of algorithmic reasoning

Next, introduce the formal problem setup corresponding to algorithmic reasoning

List of notations:

- The input graph is given by X
- The j-th intermediate step label of an algorithm A on input X is given by $A_j(X)$
- We would like to learn a neural network with parameters W such as a graph neural network (or a graph transformer) given by f_W
- The prediction of the network at step j on input X is given by $f_W(X; j)$
- Training objective minimizes the averaged loss over all steps in the execution:

$$\sum_X \frac{1}{S(X)} \sum_{j=1}^{S(X)} \ell(f_W(X; j), A_j(X))$$

$S(X)$ is the number of steps in the execution of algorithm A on input X.

Evaluation in twelve algorithmic reasoning tasks

In this experiment, we consider twelve classical graph algorithms, using the input-output encoding scheme from the Example above.

We compare training one neural network on each algorithm, vs. branching neural networks, an architecture that fits a single neural network for all twelve algorithms [LZD+26a]:

Architecture	A single GNN for each	A branching neural net for all
BFS	$100.0_{\pm 0.0}$	$99.6_{\pm 0.3}$
DFS	$38.7_{\pm 1.5}$	$38.7_{\pm 1.5}$
Topological sort	$66.0_{\pm 2.6}$	$74.2_{\pm 1.7}$
Articulation points	$99.8_{\pm 0.2}$	$93.0_{\pm 0.2}$
Bridges	$82.6_{\pm 2.0}$	$96.2_{\pm 1.4}$
SCC Kosaraju	$93.1_{\pm 3.1}$	$92.0_{\pm 1.8}$
MST Kruskal	$69.9_{\pm 1.5}$	$66.4_{\pm 1.3}$
MST Prim	$53.8_{\pm 1.3}$	$52.2_{\pm 2.4}$
Dijkstra	$70.9_{\pm 2.3}$	$68.8_{\pm 1.7}$
Bellman Ford	$63.8_{\pm 4.1}$	$61.8_{\pm 2.4}$
DAG	$89.2_{\pm 1.4}$	$89.1_{\pm 1.5}$
Floyd Warshall	$69.4_{\pm 0.5}$	$63.1_{\pm 1.3}$

Summary of our main empirical findings

1. Depth-first search (DFS) is notably hard to "learn" as compared to breadth-first search (BFS): for inputs generated on Erdős-Rényi random graph inputs with 16 nodes, existing neural net architectures achieve less than 40% accuracy — this is also consistent with earlier findings [VBB+22]
2. Learning multiple algorithmic reasoning processes is as easy as learning a single process: for all twelve algorithms tested in this evaluation, training a single model with separate prediction heads for each algorithm plus a shared base turns out to perform as well as training separate models each
3. For text description versions of the same problem, fine-tuning existing LLMs follows similar patterns, but the evaluation also becomes more challenging

All of these findings point to natural open questions around the expressivity of neural networks in terms of algorithmic reasoning

Summary of our main empirical findings

1. Depth-first search (DFS) is notably hard to "learn" as compared to breadth-first search (BFS): for inputs generated on Erdős-Rényi random graph inputs with 16 nodes, existing neural net architectures achieve less than 40% accuracy — this is also consistent with earlier findings [VBB+22]
2. Learning multiple algorithmic reasoning processes is as easy as learning a single process: for all twelve algorithms tested in this evaluation, training a single model with separate prediction heads for each algorithm plus a shared base turns out to perform as well as training separate models each
3. For text description versions of the same problem, fine-tuning existing LLMs follows similar patterns, but the evaluation also becomes more challenging

All of these findings point to natural open questions around the expressivity of neural networks in terms of algorithmic reasoning

Summary of our main empirical findings

1. Depth-first search (DFS) is notably hard to "learn" as compared to breadth-first search (BFS): for inputs generated on Erdős-Rényi random graph inputs with 16 nodes, existing neural net architectures achieve less than 40% accuracy — this is also consistent with earlier findings [VBB+22]
2. Learning multiple algorithmic reasoning processes is as easy as learning a single process: for all twelve algorithms tested in this evaluation, training a single model with separate prediction heads for each algorithm plus a shared base turns out to perform as well as training separate models each
3. For text description versions of the same problem, fine-tuning existing LLMs follows similar patterns, but the evaluation also becomes more challenging

All of these findings point to natural open questions around the expressivity of neural networks in terms of algorithmic reasoning

Summary of our main empirical findings

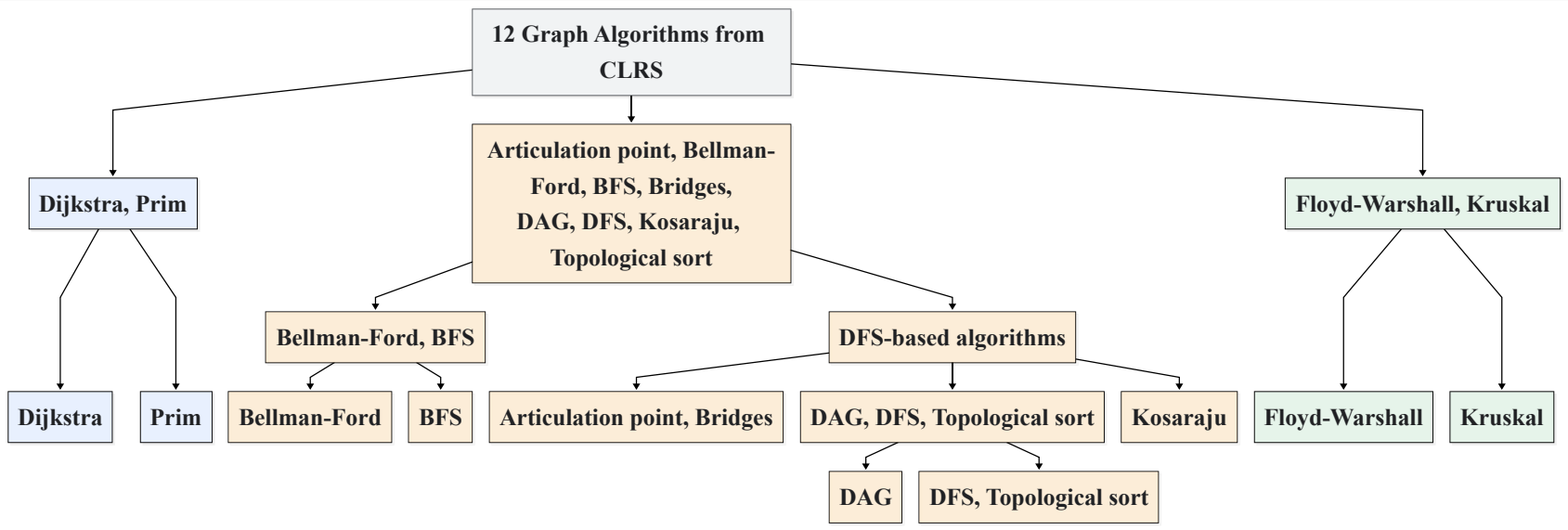
1. Depth-first search (DFS) is notably hard to "learn" as compared to breadth-first search (BFS): for inputs generated on Erdős-Rényi random graph inputs with 16 nodes, existing neural net architectures achieve less than 40% accuracy — this is also consistent with earlier findings [VBB+22]
2. Learning multiple algorithmic reasoning processes is as easy as learning a single process: for all twelve algorithms tested in this evaluation, training a single model with separate prediction heads for each algorithm plus a shared base turns out to perform as well as training separate models each
3. For text description versions of the same problem, fine-tuning existing LLMs follows similar patterns, but the evaluation also becomes more challenging

All of these findings point to natural open questions around the expressivity of neural networks in terms of algorithmic reasoning

A new architecture for multitask algorithmic reasoning

In the paper [LZD+26a], we also propose a new algorithm to estimate branching neural networks

An illustration of the tree partitioning corresponding to 12 algorithms



Outline

Multitask algorithmic reasoning

A stability analysis of multi-step reasoning via error amplification

Grokking in modular arithmetic reasoning

Problem setup

Next, we present a formal setup to analyze the above problem

- $\mathcal{X}$: a feature space
- $f_W : \mathcal{X}^\star \to \mathcal{X}$, next-token predictor with parameters $W \in \mathbb{R}^P$
- For an input sequence $\mathbf{x} \in \mathcal{X}^\star$: assume there exists an underlying procedure that generates the ground-truth sequence $[\mathbf{x}, y^{(1)}, \ldots, y^{(T(\mathbf{x}))}]$. $y^{(t)} \in \mathcal{X}$: output at step t. $T(\mathbf{x})$: total number of steps
- Algorithmic reasoning: $\mathbf{x}$ is a graph plus a query, $y^{(i)}$ is an intermediate execution step, the underlying procedure is an algorithm

At inference time, f_W generates for $T(\mathbf{x})$ steps given $\mathbf{x}$, autoregressively:

$$y^{(t)} = f_W([\mathbf{x}, \widehat{y}^{(1)}, \ldots, \widehat{y}^{(t-1)}])$$

During training, f_W predicts the next token conditioned on the ground-truth

$$\mathbf{z}^{(t)}(\mathbf{x}) = [\mathbf{x}, y^{(1)}, \ldots, y^{(t)}], \text{ for } t = 1, \ldots, T(\mathbf{x})$$

Problem setup

Next, we present a formal setup to analyze the above problem

- $\mathcal{X}$: a feature space
- $f_W : \mathcal{X}^\star \to \mathcal{X}$, next-token predictor with parameters $W \in \mathbb{R}^p$
- For an input sequence $\mathbf{x} \in \mathcal{X}^\star$: assume there exists an underlying procedure that generates the ground-truth sequence $[\mathbf{x}, y^{(1)}, \ldots, y^{(T(\mathbf{x}))}]$. $y^{(t)} \in \mathcal{X}$: output at step t. $T(\mathbf{x})$: total number of steps
- Algorithmic reasoning: $\mathbf{x}$ is a graph plus a query, $y^{(i)}$ is an intermediate execution step, the underlying procedure is an algorithm

At inference time, f_W generates for $T(\mathbf{x})$ steps given $\mathbf{x}$, autoregressively:

$$y^{(t)} = f_W([\mathbf{x}, \widehat{y}^{(1)}, \ldots, \widehat{y}^{(t-1)}])$$

During training, f_W predicts the next token conditioned on the ground-truth

$$\mathbf{z}^{(t)}(\mathbf{x}) = [\mathbf{x}, y^{(1)}, \ldots, y^{(t)}], \text{ for } t = 1, \ldots, T(\mathbf{x})$$

Problem setup

Next, we present a formal setup to analyze the above problem

- $\mathcal{X}$: a feature space
- $f_W : \mathcal{X}^\star \to \mathcal{X}$, next-token predictor with parameters $W \in \mathbb{R}^p$
- For an input sequence $\mathbf{x} \in \mathcal{X}^\star$: assume there exists an underlying procedure that generates the ground-truth sequence $[\mathbf{x}, y^{(1)}, \ldots, y^{(T(\mathbf{x}))}]$. $y^{(t)} \in \mathcal{X}$: output at step t. $T(\mathbf{x})$: total number of steps
- Algorithmic reasoning: $\mathbf{x}$ is a graph plus a query, $y^{(i)}$ is an intermediate execution step, the underlying procedure is an algorithm

At inference time, f_W generates for $T(\mathbf{x})$ steps given $\mathbf{x}$, autoregressively:

$$y^{(t)} = f_W([\mathbf{x}, \widehat{y}^{(1)}, \ldots, \widehat{y}^{(t-1)}])$$

During training, f_W predicts the next token conditioned on the ground-truth

$$\mathbf{z}^{(t)}(\mathbf{x}) = [\mathbf{x}, y^{(1)}, \ldots, y^{(t)}], \text{ for } t = 1, \ldots, T(\mathbf{x})$$

Problem setup

Next, we present a formal setup to analyze the above problem

- $\mathcal{X}$: a feature space
- $f_W : \mathcal{X}^\star \to \mathcal{X}$, next-token predictor with parameters $W \in \mathbb{R}^p$
- For an input sequence $\mathbf{x} \in \mathcal{X}^\star$: assume there exists an underlying procedure that generates the ground-truth sequence $[\mathbf{x}, y^{(1)}, \ldots, y^{(T(\mathbf{x}))}]$. $y^{(t)} \in \mathcal{X}$: output at step t. $T(\mathbf{x})$: total number of steps
- Algorithmic reasoning: $\mathbf{x}$ is a graph plus a query, $y^{(i)}$ is an intermediate execution step, the underlying procedure is an algorithm

At inference time, f_W generates for $T(\mathbf{x})$ steps given $\mathbf{x}$, autoregressively:

$$y^{(t)} = f_W([\mathbf{x}, \widehat{y}^{(1)}, \ldots, \widehat{y}^{(t-1)}])$$

During training, f_W predicts the next token conditioned on the ground-truth

$$\mathbf{z}^{(t)}(\mathbf{x}) = [\mathbf{x}, y^{(1)}, \ldots, y^{(t)}], \text{ for } t = 1, \ldots, T(\mathbf{x})$$

Problem setup

Next, we present a formal setup to analyze the above problem

- $\mathcal{X}$: a feature space
- $f_W : \mathcal{X}^\star \to \mathcal{X}$, next-token predictor with parameters $W \in \mathbb{R}^p$
- For an input sequence $\mathbf{x} \in \mathcal{X}^\star$: assume there exists an underlying procedure that generates the ground-truth sequence $[\mathbf{x}, y^{(1)}, \ldots, y^{(T(\mathbf{x}))}]$. $y^{(t)} \in \mathcal{X}$: output at step t. $T(\mathbf{x})$: total number of steps
- Algorithmic reasoning: $\mathbf{x}$ is a graph plus a query, $y^{(i)}$ is an intermediate execution step, the underlying procedure is an algorithm

At inference time, f_W generates for $T(\mathbf{x})$ steps given $\mathbf{x}$, autoregressively:

$$y^{(t)} = f_W([\mathbf{x}, \widehat{y}^{(1)}, \ldots, \widehat{y}^{(t-1)}])$$

During training, f_W predicts the next token conditioned on the ground-truth

$$\mathbf{z}^{(t)}(\mathbf{x}) = [\mathbf{x}, y^{(1)}, \ldots, y^{(t)}], \text{ for } t = 1, \ldots, T(\mathbf{x})$$

Problem setup

Next, we present a formal setup to analyze the above problem

- $\mathcal{X}$: a feature space
- $f_W : \mathcal{X}^\star \to \mathcal{X}$, next-token predictor with parameters $W \in \mathbb{R}^p$
- For an input sequence $\mathbf{x} \in \mathcal{X}^\star$: assume there exists an underlying procedure that generates the ground-truth sequence $[\mathbf{x}, y^{(1)}, \ldots, y^{(T(\mathbf{x}))}]$. $y^{(t)} \in \mathcal{X}$: output at step t. $T(\mathbf{x})$: total number of steps
- Algorithmic reasoning: $\mathbf{x}$ is a graph plus a query, $y^{(i)}$ is an intermediate execution step, the underlying procedure is an algorithm

At inference time, f_W generates for $T(\mathbf{x})$ steps given $\mathbf{x}$, autoregressively:

$$y^{(t)} = f_W([\mathbf{x}, \widehat{y}^{(1)}, \ldots, \widehat{y}^{(t-1)}])$$

During training, f_W predicts the next token conditioned on the ground-truth

$$\mathbf{z}^{(t)}(\mathbf{x}) = [\mathbf{x}, y^{(1)}, \ldots, y^{(t)}], \text{ for } t = 1, \ldots, T(\mathbf{x})$$

Problem setup

Next, we present a formal setup to analyze the above problem

- $\mathcal{X}$: a feature space
- $f_W : \mathcal{X}^\star \to \mathcal{X}$, next-token predictor with parameters $W \in \mathbb{R}^p$
- For an input sequence $\mathbf{x} \in \mathcal{X}^\star$: assume there exists an underlying procedure that generates the ground-truth sequence $[\mathbf{x}, y^{(1)}, \ldots, y^{(T(\mathbf{x}))}]$. $y^{(t)} \in \mathcal{X}$: output at step t. $T(\mathbf{x})$: total number of steps
- Algorithmic reasoning: $\mathbf{x}$ is a graph plus a query, $y^{(i)}$ is an intermediate execution step, the underlying procedure is an algorithm

At inference time, f_W generates for $T(\mathbf{x})$ steps given $\mathbf{x}$, autoregressively:

$$y^{(t)} = f_W([\mathbf{x}, \widehat{y}^{(1)}, \ldots, \widehat{y}^{(t-1)}])$$

During training, f_W predicts the next token conditioned on the ground-truth

$$\mathbf{z}^{(t)}(\mathbf{x}) = [\mathbf{x}, y^{(1)}, \ldots, y^{(t)}], \text{ for } t = 1, \ldots, T(\mathbf{x})$$

Problem statement

Training objective: Let $S = \left\{ (\mathbf{x}_i, y_i^{(1)}, \ldots, y_i^{(T(\mathbf{x}_i))}) \right\}_{i=1}^{n}$ denote a set of n training examples, where $\mathbf{x}_i \sim \mathcal{D}$ for some input distribution $\mathcal{D}$:

$$L(f_W; S) = \frac{1}{n} \sum_{i=1}^{n} \sum_{t=0}^{T(\mathbf{x}_i)-1} \ell(f_W(\mathbf{z}_i^{(t)}), y_i^{(t+1)})$$

Prediction: Let

$$\widehat{\mathbf{z}}^{(t)} = [\mathbf{x}, \widehat{y}^{(1)}, \ldots, \widehat{y}^{(t)}]$$

Inference loss:

$$L(f_W; \mathcal{D}) = \mathop{\mathbb{E}}_{\mathbf{x} \sim \mathcal{D}} \left[\ell(f_W(\widehat{\mathbf{z}}^{(T(\mathbf{x})-1)}), y^{(T(\mathbf{x}))}) \right]$$

Questions:

- How do errors from preceding steps accumulate throughout generation?
- Under what conditions will the inference loss relative to the ground truth remain bounded, even for large T?

Problem statement

Training objective: Let $S = \left\{ (\mathbf{x}_i, y_i^{(1)}, \ldots, y_i^{(T(\mathbf{x}_i))}) \right\}_{i=1}^{n}$ denote a set of n training examples, where $\mathbf{x}_i \sim \mathcal{D}$ for some input distribution $\mathcal{D}$:

$$L(f_W; S) = \frac{1}{n} \sum_{i=1}^{n} \sum_{t=0}^{T(\mathbf{x}_i)-1} \ell(f_W(\mathbf{z}_i^{(t)}), y_i^{(t+1)})$$

Prediction: Let

$$\widehat{\mathbf{z}}^{(t)} = [\mathbf{x}, \widehat{y}^{(1)}, \ldots, \widehat{y}^{(t)}]$$

Inference loss:

$$L(f_W; \mathcal{D}) = \mathop{\mathbb{E}}_{\mathbf{x} \sim \mathcal{D}} \left[\ell(f_W(\widehat{\mathbf{z}}^{(T(\mathbf{x})-1)}), y^{(T(\mathbf{x}))}) \right]$$

Questions:

- How do errors from preceding steps accumulate throughout generation?
- Under what conditions will the inference loss relative to the ground truth remain bounded, even for large T?

Problem statement

Training objective: Let $S = \left\{ (\mathbf{x}_i, y_i^{(1)}, \ldots, y_i^{(T(\mathbf{x}_i))}) \right\}_{i=1}^{n}$ denote a set of n training examples, where $\mathbf{x}_i \sim \mathcal{D}$ for some input distribution $\mathcal{D}$:

$$L(f_W; S) = \frac{1}{n} \sum_{i=1}^{n} \sum_{t=0}^{T(\mathbf{x}_i)-1} \ell(f_W(\mathbf{z}_i^{(t)}), y_i^{(t+1)})$$

Prediction: Let

$$\widehat{\mathbf{z}}^{(t)} = [\mathbf{x}, \widehat{y}^{(1)}, \ldots, \widehat{y}^{(t)}]$$

Inference loss:

$$L(f_W; \mathcal{D}) = \mathop{\mathbb{E}}_{\mathbf{x} \sim \mathcal{D}} \left[\ell(f_W(\widehat{\mathbf{z}}^{(T(\mathbf{x})-1)}), y^{(T(\mathbf{x}))}) \right]$$

Questions:

- How do errors from preceding steps accumulate throughout generation?
- Under what conditions will the inference loss relative to the ground truth remain bounded, even for large T?

Problem statement

Training objective: Let $S = \left\{ (\mathbf{x}_i, y_i^{(1)}, \ldots, y_i^{(T(\mathbf{x}_i))}) \right\}_{i=1}^{n}$ denote a set of n training examples, where $\mathbf{x}_i \sim \mathcal{D}$ for some input distribution $\mathcal{D}$:

$$L(f_W; S) = \frac{1}{n} \sum_{i=1}^{n} \sum_{t=0}^{T(\mathbf{x}_i)-1} \ell(f_W(\mathbf{z}_i^{(t)}), y_i^{(t+1)})$$

Prediction: Let

$$\widehat{\mathbf{z}}^{(t)} = [\mathbf{x}, \widehat{y}^{(1)}, \ldots, \widehat{y}^{(t)}]$$

Inference loss:

$$L(f_W; \mathcal{D}) = \mathop{\mathbb{E}}_{\mathbf{x} \sim \mathcal{D}} \left[\ell(f_W(\widehat{\mathbf{z}}^{(T(\mathbf{x})-1)}), y^{(T(\mathbf{x}))}) \right]$$

Questions:

- How do errors from preceding steps accumulate throughout generation?
- Under what conditions will the inference loss relative to the ground truth remain bounded, even for large T?

Main derivation

Notations:

- $e^{(t)}(\mathbf{x}) = f_W(\mathbf{z}^{(t-1)}) - y^{(t)}$: prediction error conditioned on ground truth
- $\widehat{e}^{(t)}(\mathbf{x}) = f_W(\widehat{\mathbf{z}}^{(t-1)}) - y^{(t)}$: prediction error conditioned on preceding predictions
- $J_i^{(t)}(\mathbf{x}) = \frac{\partial f_W(\mathbf{z}^{(t)})}{\partial y^{(i)}}$: Jacobian matrix measured in the input space

Derivation

Consider the first-order Taylor expansion in the input space,

$$f_W(\widehat{\mathbf{z}}^{(t-1)}) - f_W(\mathbf{z}^{(t-1)}) = \sum_{i=1}^{t-1} J_i^{(t)}(\mathbf{x})\, \widehat{e}^{(i)}(\mathbf{x}) + \mathcal{R}^{(t)},$$

for every t, with expansion error $\mathcal{R}^{(t)}$. By recursively applying this expansion, an error from step i can propagate into a later step. To capture this, define

$$\rho_i^{(t)} = \sup_{\mathbf{x}\in\mathcal{X}^*} \left\| J_i^{(t)}(\mathbf{x}) \right\| + \sum_{k=i+1}^{t-1} \left(\sup_{\mathbf{x}\in\mathcal{X}^*} \left\| J_k^{(t)}(\mathbf{x}) \right\| \cdot \rho_i^{(k)} \right), \text{ with } \rho_t^{(t)} = 1$$

$$\sup_{1\le t\le T-1, \mathbf{x}\in\mathcal{X}} \left\| e^{(t)}(\mathbf{x}) \right\| = \varepsilon_t, \quad \sup_{1\le t\le T-1, \mathbf{x}\in\mathcal{X}} \left\| \hat{e}^{(t)}(\mathbf{x}) \right\| = r_t$$

Main derivation

Notations:

- $e^{(t)}(\mathbf{x}) = f_W(\mathbf{z}^{(t-1)}) - y^{(t)}$: prediction error conditioned on ground truth
- $\widehat{e}^{(t)}(\mathbf{x}) = f_W(\widehat{\mathbf{z}}^{(t-1)}) - y^{(t)}$: prediction error conditioned on preceding predictions
- $J_i^{(t)}(\mathbf{x}) = \frac{\partial f_W(\mathbf{z}^{(t)})}{\partial y^{(i)}}$: Jacobian matrix measured in the input space

Derivation

Consider the first-order Taylor expansion in the input space,

$$f_W(\widehat{\mathbf{z}}^{(t-1)}) - f_W(\mathbf{z}^{(t-1)}) = \sum_{i=1}^{t-1} J_i^{(t)}(\mathbf{x})\, \widehat{e}^{(i)}(\mathbf{x}) + \mathcal{R}^{(t)},$$

for every t, with expansion error $\mathcal{R}^{(t)}$. By recursively applying this expansion, an error from step i can propagate into a later step. To capture this, define

$$\rho_i^{(t)} = \sup_{\mathbf{x}\in\mathcal{X}^*} \left\| J_i^{(t)}(\mathbf{x}) \right\| + \sum_{k=i+1}^{t-1} \left(\sup_{\mathbf{x}\in\mathcal{X}^*} \left\| J_k^{(t)}(\mathbf{x}) \right\| \cdot \rho_i^{(k)} \right), \text{ with } \rho_t^{(t)} = 1$$

$$\sup_{1\le t\le T-1, \mathbf{x}\in\mathcal{X}} \left\| e^{(t)}(\mathbf{x}) \right\| = \varepsilon_t, \quad \sup_{1\le t\le T-1, \mathbf{x}\in\mathcal{X}} \left\| \hat{e}^{(t)}(\mathbf{x}) \right\| = r_t$$

Main lemma

Informal statement: assume that f_W is twice-differentiable, the loss function and its Jacobian matrix are Lipschitz-continuous, and let

$$L^\star(f_W; \mathcal{D}) = \mathop{\mathbb{E}}_{\mathbf{x}\sim\mathcal{D}} \left[\ell\left(f_W(\mathbf{z}^{(T-1)}, y^{(T)}) \right) \right]$$

For any W,

$$\left| L(f_W; \mathcal{D}) - L^\star(f_W; \mathcal{D}) \right| \lesssim \sum_{t=1}^{T-1} \rho_t^{(T)} \varepsilon_t + \sum_{t=2}^{T} \rho_t^{(T)} \left(\sum_{i=1}^{t-1} r_i \right)^2$$

Main lemma

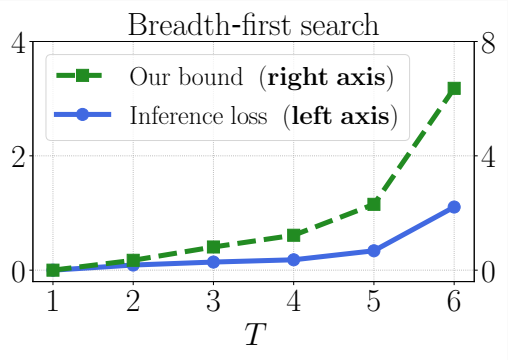
Informal statement: assume that f_W is twice-differentiable, the loss function and its Jacobian matrix are Lipschitz-continuous, and let

$$L^\star(f_W;\mathcal{D}) = \mathop{\mathbb{E}}_{\mathbf{x}\sim\mathcal{D}}\left[\ell\left(f_W(\mathbf{z}^{(T-1)}, y^{(T)})\right)\right]$$

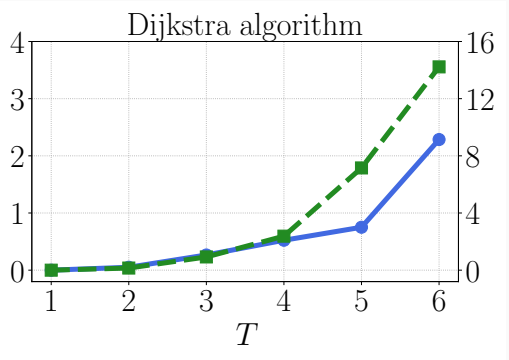
For any W,

$$\left|L(f_W;\mathcal{D}) - L^\star(f_W;\mathcal{D})\right| \lesssim \sum_{t=1}^{T-1}\rho_t^{(T)}\varepsilon_t + \sum_{t=2}^{T}\rho_t^{(T)}\left(\sum_{i=1}^{t-1} r_i\right)^2$$

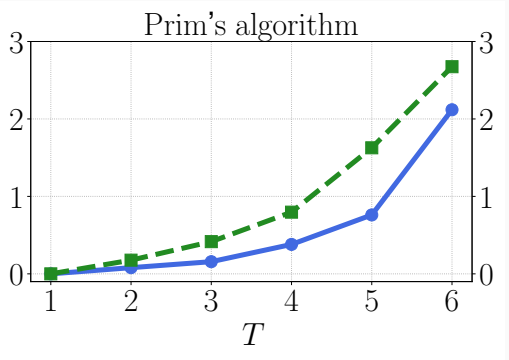
Numerical illustration: Comparing the above with actual inference loss. This simple numerical approximation can track the growth of the inference loss, and scales exponentially with t! (Results are evaluated using QWen-1.5B)



(a) Breadth-first search



(b) Dijkstra



(c) Prim's algorithm

Case studies on in-context linear and quadratic regression

Synthetic data: The chain-of-thought data is generated via gradient descent trajectories [HWL25; HWS+25]

- Linear regression: given $x_i \in \mathbb{R}^d$, the desired output is $y_i = x_i^\top \theta^\star + \epsilon$, for some unknown $\theta^\star \in \mathbb{R}^d$ and Gaussian noise ϵ. The "intermediate hints" are given by gradient descent iterates on n examples $(x_1, y_1), \ldots, (x_n, y_n)$:

$$\theta_{i+1} = \theta_i - \frac{\eta}{n} \sum_{k=1}^{n} \nabla (x_k^\top \theta_i - y_k)^2,$$

- Quadratic regression: Similar to the above, but instead $y_i = (x_i^\top \theta^\star)^{\odot 2} + \epsilon$

Case studies on in-context linear and quadratic regression

Synthetic data: The chain-of-thought data is generated via gradient descent trajectories [HWL25; HWS+25]

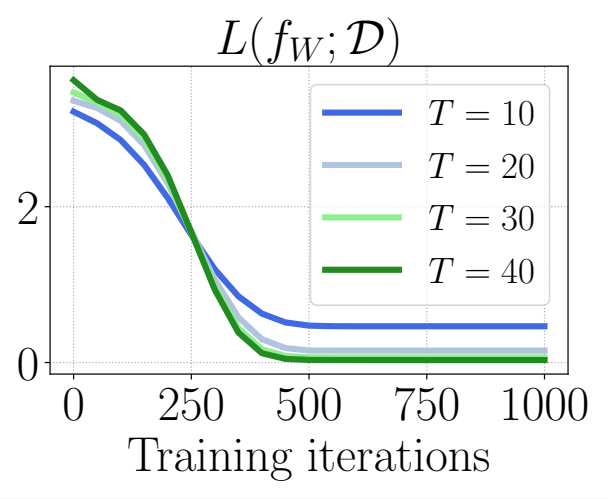
- Linear regression: given $x_i \in \mathbb{R}^d$, the desired output is $y_i = x_i^\top \theta^\star + \epsilon$, for some unknown $\theta^\star \in \mathbb{R}^d$ and Gaussian noise ϵ. The "intermediate hints" are given by gradient descent iterates on n examples $(x_1, y_1), \ldots, (x_n, y_n)$:

$$\theta_{i+1} = \theta_i - \frac{\eta}{n} \sum_{k=1}^{n} \nabla (x_k^\top \theta_i - y_k)^2,$$

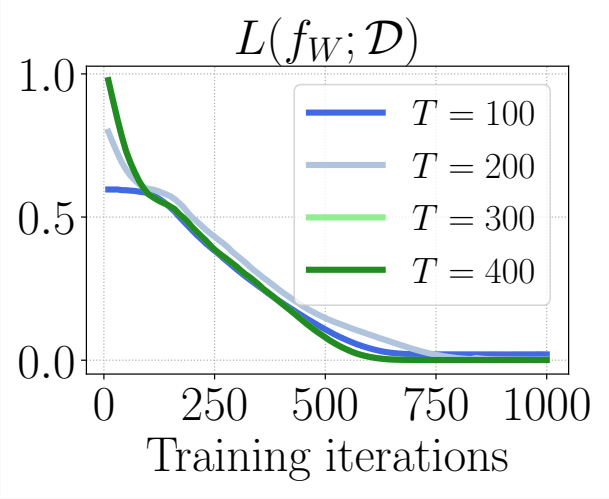
- Quadratic regression: Similar to the above, but instead $y_i = (x_i^\top \theta^\star)^{\odot 2} + \epsilon$

Numerical validation

Transformers trained with gradient descent converge to zero error in the above synthetic chain-of-thought data:



(a) Linear regression

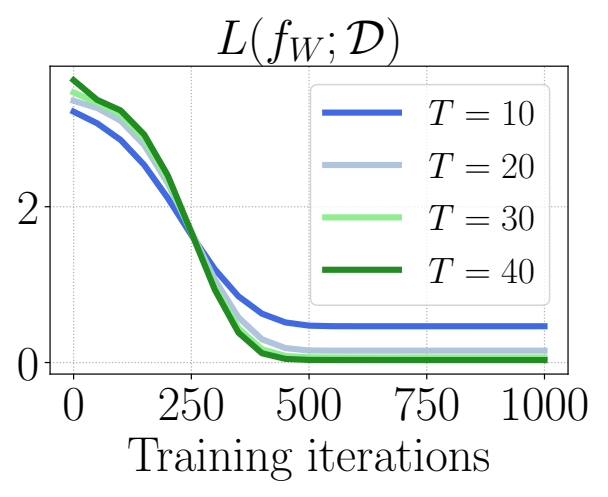


(b) Quadratic regression

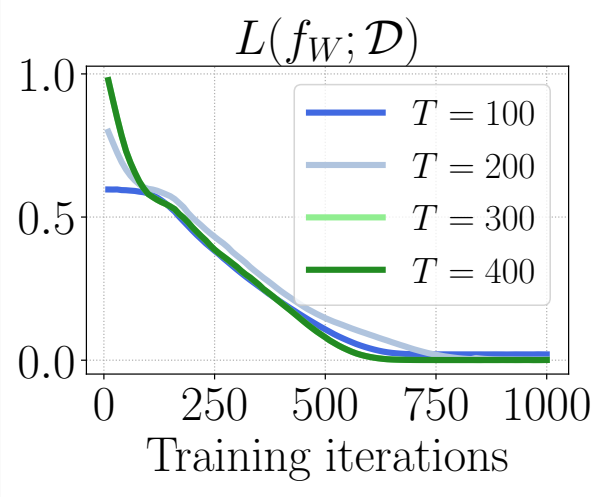
Building on prior work [HWL25; HWS+25], we rigorously prove that this is true, showing that the stability measures decrease to zero

Numerical validation

Transformers trained with gradient descent converge to zero error in the above synthetic chain-of-thought data:



(a) Linear regression



(b) Quadratic regression

Building on prior work [HWL25; HWS+25], we rigorously prove that this is true, showing that the stability measures decrease to zero

Outline

Multitask algorithmic reasoning

A stability analysis of multi-step reasoning via error amplification

Grokking in modular arithmetic reasoning

Grokking in modular arithmetic tasks

Grokking is a delayed generalization phenomenon in modular arithmetic tasks such as $a + b \pmod{p}$

- A transformer model first achieves 100% training accuracy, while test accuracy remains low
- After many more training epochs, test accuracy increases

Hessian regularization via noise injection: Instead of minimizing the loss ℓ, minimize the perturbed loss

$$\ell_{\mathcal{Q}}(f_{\mathcal{W}}) = \mathop{\mathbb{E}}_{\mathcal{U} \sim \mathcal{N}(0, \sigma^2 \mathrm{Id})} [\ell(f_{\mathcal{W}+\mathcal{U}})] \,, \tag{1}$$

leading to the following noise-injected gradient update [ZLJ24] (also known as *randomized smoothing* [DBW12])

$$\mathcal{W} \leftarrow \mathcal{W} - \eta \nabla_{\mathcal{W}} \ell(f_{\mathcal{W}+\mathcal{U}}(x), y)$$

Grokking in modular arithmetic tasks

Grokking is a delayed generalization phenomenon in modular arithmetic tasks such as $a + b \pmod{p}$

- A transformer model first achieves 100% training accuracy, while test accuracy remains low
- After many more training epochs, test accuracy increases

Hessian regularization via noise injection: Instead of minimizing the loss ℓ, minimize the perturbed loss

$$\ell_{\mathcal{Q}}(f_W) = \mathop{\mathbb{E}}_{U \sim \mathcal{N}(0, \sigma^2 \,\mathrm{Id})} [\ell(f_{W+U})] \,, \tag{1}$$

leading to the following noise-injected gradient update [ZLJ24] (also known as *randomized smoothing* [DBW12])

$$W \leftarrow W - \eta \nabla_W \ell(f_{W+U}(x), y)$$

Illustration of grokking in modular arithmetic tasks

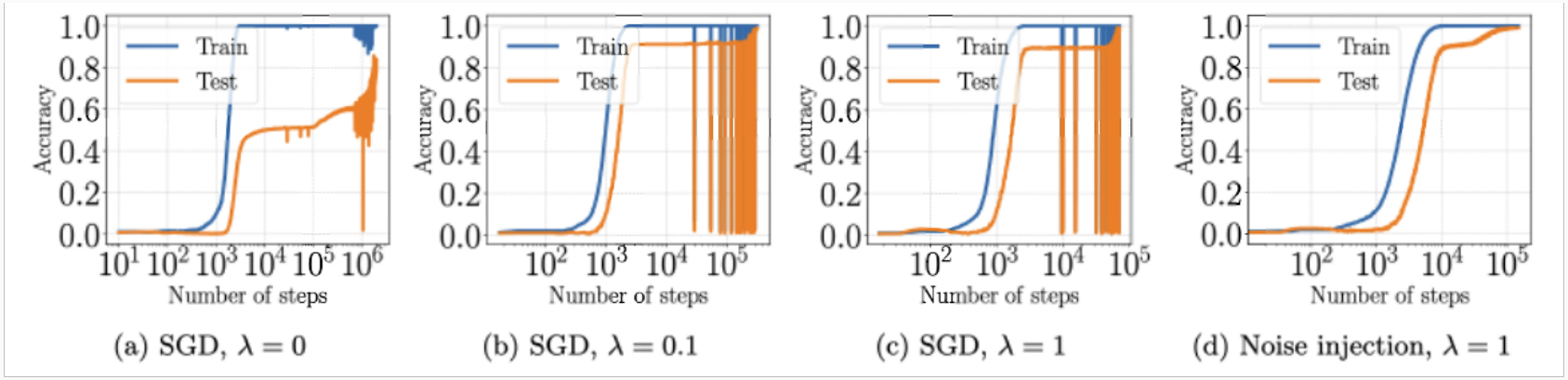


Figure 3: Comparing the training curves of transformer models for modular arithmetic tasks using weight decay (with varying levels indicated by λ), and Hessian regularization (via noise injection)

Hessian regularization stabilizes the training curves and reduces the number of grokking steps

Illustration of Hessian regularization

First-order Taylor expansion of the perturbed loss

Recall equation (1), $\mathcal{Q} = \mathcal{N}(W, \sigma^2 \mathrm{Id})$, $\mathcal{U} \sim \mathcal{N}(0, \sigma^2 \mathrm{Id})$, and

$$\ell_{\mathcal{Q}}(f_W) = \mathop{\mathbb{E}}_{W \sim \mathcal{Q}} [\ell(f_W)] = \mathop{\mathbb{E}}_{\mathcal{U}} [\ell(f_{W+\mathcal{U}})]$$

whose second-order Taylor expansion is

$$\mathop{\mathbb{E}}_{\mathcal{U}} \left[\ell(f_W) + \sum_{i=1}^{l} \langle \nabla_{W_i} \ell(f_W), U_i \rangle + \frac{1}{2} \sum_{i=1}^{l} \langle \nabla^2_{W_i} \ell(f_{W_i}), U_i U_i^\top \rangle \right] + O(\sigma^3)$$

$$= \ell(f_W) + \underbrace{\frac{\sigma^2}{2} \operatorname{Tr} \left[\nabla^2_W \ell(f_W) \right]}_{\text{Hessian trace penalty weighted by } \frac{\sigma^2}{2}} + O(\sigma^3) \tag{2}$$

Illustration of Hessian regularization

First-order Taylor expansion of the perturbed loss

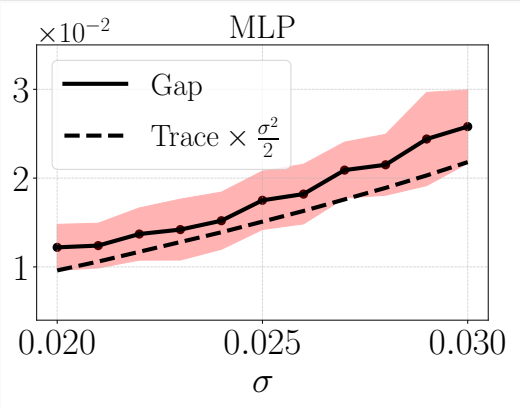
Recall equation (1), $\mathcal{Q} = \mathcal{N}(W, \sigma^2 \mathrm{Id})$, $\mathcal{U} \sim \mathcal{N}(0, \sigma^2 \mathrm{Id})$, and

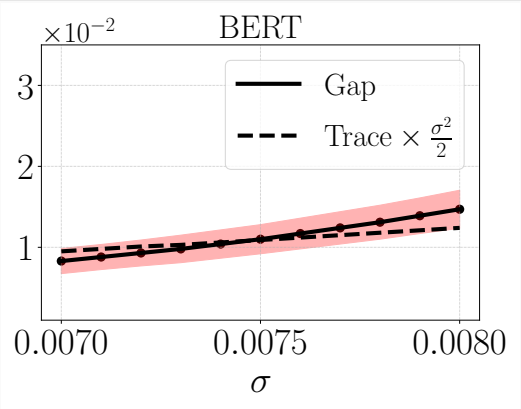
$$\ell_{\mathcal{Q}}(f_W) = \mathop{\mathbb{E}}_{W \sim \mathcal{Q}} [\ell(f_W)] = \mathop{\mathbb{E}}_{\mathcal{U}} [\ell(f_{W+\mathcal{U}})]$$

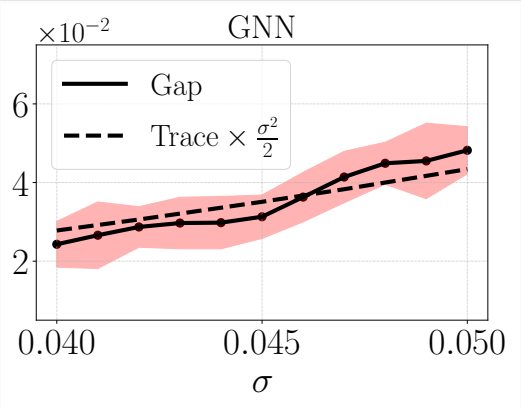
whose second-order Taylor expansion is

$$\mathop{\mathbb{E}}_{\mathcal{U}} \left[\ell(f_W) + \sum_{i=1}^{l} \langle \nabla_{W_i} \ell(f_W), U_i \rangle + \frac{1}{2} \sum_{i=1}^{l} \langle \nabla^2_{W_i} \ell(f_{W_i}), U_i U_i^\top \rangle \right] + O(\sigma^3)$$

$$= \ell(f_W) + \underbrace{\frac{\sigma^2}{2} \mathrm{Tr} \left[\nabla^2_W \ell(f_W) \right]}_{\text{Hessian trace penalty weighted by } \frac{\sigma^2}{2}} + O(\sigma^3) \qquad (2)$$







Using AI to solve an open question from prior work

One-sided matrix completion [ZZN+26]

- $M \in \mathbb{R}^{n \times d}$: a ground-truth matrix
- $\widehat{M}$: observe each sample of M with probability $p \in (0, 1)$
- Ultra-sparse regime: when $p = \frac{C}{d}$ (for some constant C), accurate recovery of M given $\widehat{M}$ is not possible. What could one hope to learn about M?
- One-sided recovery: imagine that M is a tall-and-skinny matrix (e.g., movie/music ratings), recover the row-vector subspace, $T = \frac{1}{n} M^\top M$

Prior art: known when $C = 2$, but the general case remains open [CLV23]

- My student, coauthors, and I have worked on this problem for a year but did not reach a resolution
- Around April 2025, we started using GPT-4. With the help of GPT-4, in two months we worked out the main proof, which involved a lot of tedious calculations and extensive use of concentration results

Using AI to solve an open question from prior work

One-sided matrix completion [ZZN+26]

- $M \in \mathbb{R}^{n \times d}$: a ground-truth matrix
- $\widehat{M}$: observe each sample of M with probability $p \in (0, 1)$
- Ultra-sparse regime: when $p = \frac{C}{d}$ (for some constant C), accurate recovery of M given $\widehat{M}$ is not possible. What could one hope to learn about M?
- One-sided recovery: imagine that M is a tall-and-skinny matrix (e.g., movie/music ratings), recover the row-vector subspace, $T = \frac{1}{n} M^\top M$

Prior art: known when $C = 2$, but the general case remains open [CLV23]

- My student, coauthors, and I have worked on this problem for a year but did not reach a resolution
- Around April 2025, we started using GPT-4. With the help of GPT-4, in two months we worked out the main proof, which involved a lot of tedious calculations and extensive use of concentration results

Using AI to solve an open question from prior work

One-sided matrix completion [ZZN+26]

- $M \in \mathbb{R}^{n \times d}$: a ground-truth matrix
- $\widehat{M}$: observe each sample of M with probability $p \in (0, 1)$
- Ultra-sparse regime: when $p = \frac{C}{d}$ (for some constant C), accurate recovery of M given $\widehat{M}$ is not possible. What could one hope to learn about M?
- One-sided recovery: imagine that M is a tall-and-skinny matrix (e.g., movie/music ratings), recover the row-vector subspace, $T = \frac{1}{n} M^\top M$

Prior art: known when $C = 2$, but the general case remains open [CLV23]

- My student, coauthors, and I have worked on this problem for a year but did not reach a resolution
- Around April 2025, we started using GPT-4. With the help of GPT-4, in two months we worked out the main proof, which involved a lot of tedious calculations and extensive use of concentration results

Using AI to solve an open question from prior work

One-sided matrix completion [ZZN+26]

- $M \in \mathbb{R}^{n \times d}$: a ground-truth matrix
- $\widehat{M}$: observe each sample of M with probability $p \in (0, 1)$
- Ultra-sparse regime: when $p = \frac{C}{d}$ (for some constant C), accurate recovery of M given $\widehat{M}$ is not possible. What could one hope to learn about M?
- One-sided recovery: imagine that M is a tall-and-skinny matrix (e.g., movie/music ratings), recover the row-vector subspace, $T = \frac{1}{n} M^\top M$

Prior art: known when $C = 2$, but the general case remains open [CLV23]

- My student, coauthors, and I have worked on this problem for a year but did not reach a resolution
- Around April 2025, we started using GPT-4. With the help of GPT-4, in two months we worked out the main proof, which involved a lot of tedious calculations and extensive use of concentration results

Conclusion

- We conduct an evaluation of algorithmic reasoning, and find that existing neural network models (e.g., graph neural nets, transformers) can memorize multiple execution processes simultaneously [LZD+26a]
  - Discrepancy between BFS vs. DFS: explanation via expressivity?
- We propose a stability measure to analyze multi-step reasoning based on the product of the spectral norms of Jacobian matrices, and correlate this measure with actual inference losses on large models [LZD+26b]
  - The convergence analysis for linear and quadratic regression builds on several prior works on provable chain-of-thought learning [HWL25; JVB+25], and symbolic multi-step reasoning [YHL+25]

Paper

Code

Cao, S., Liang, P., and Valiant, G. (2023). **"One-sided matrix completion from two observations per row".** In: *International Conference on Machine Learning*. PMLR, pp. 3599–3624 (43–46).

Duchi, J. C., Bartlett, P. L., and Wainwright, M. J. (2012). **"Randomized smoothing for stochastic optimization".** In: *SIAM Journal on Optimization* 22.2, pp. 674–701 (38, 39).

Huang, J., Wang, Z., and Lee, J. D. (2025a). **"Transformers Learn to Implement Multi-step Gradient Descent with Chain of Thought".** In: *International Conference on Learning Representations (ICLR)* (33–36, 47).

Huang, Y., Wen, Z., Singh, A., Chi, Y., and Chen, Y. (2025b). **"Transformers provably learn chain-of-thought reasoning with length generalization".** In: *NeurIPS* (33–36).

Joshi, N., Vardi, G., Block, A., Goel, S., Li, Z., Misiakiewicz, T., and Srebro, N. (2025). **"A theory of learning with autoregressive chain of thought".** In: *Annual Conference on Learning Theory (COLT)* (47).

Li, D., Zhang, Z., Duan, M., Dobriban, E., and Zhang, H. R. (2026a). **"Efficiently Learning Branching Networks for Multitask Algorithmic**

Reasoning". In: *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)* (11, 16, 47).

Li, D., Zhang, Z., Duan, M., and Zhang, H. R. (2026b). **"Quantifying the Stability of Multi-Step Reasoning via Error Amplification".** In: *Advances in Neural Information Processing Systems* (47).

Veličković, P., Badia, A. P., Budden, D., Pascanu, R., Banino, A., Dashevskiy, M., Hadsell, R., and Blundell, C. (2022). **"The CLRS Algorithmic Reasoning Benchmark".** In: *International Conference on Machine Learning (ICML)* (12–15).

Yang, T., Huang, Y., Liang, Y., and Chi, Y. (2025). **"Multi-head transformers provably learn symbolic multi-step reasoning via gradient descent".** In: *Advances in Neural Information Processing Systems* 38, pp. 116194–116285 (47).

Zhang, H. R., Li, D., and Ju, H. (2024). **"Noise Stability Optimization for Finding Flat Minima: A Hessian-based Regularization Approach".** In: *Transactions on Machine Learning Research* (38, 39).

Zhang, H. R., Zhang, Z., Nguyen, H. L., and Lan, G. (2026). **"One-Sided Matrix Completion from Ultra-Sparse Samples".** In: *Transactions on Machine Learning Research* (43–46).