

Towards Revealing the Mystery behind Chain of Thought: A Theoretical Perspective

Guhao Feng*, Bohang Zhang*, Yuntian Gu*, Haotian Ye*, Di He, Liwei Wang

Peking University

2023.6.8

Index

- 1 Introduction
- 2 Preliminary
- 3 CoT is the Key to Solving Math Problems
- 4 CoT is the Key to Solving General Problems
- 5 Experiments

Index

- 1 Introduction
- 2 Preliminary
- 3 CoT is the Key to Solving Math Problems
- 4 CoT is the Key to Solving General Problems
- 5 Experiments

Capabilities of LLMs

Large Language Models (LLMs) have demonstrated emergent capabilities in various aspects:

- Generation: translation, summary, composition, ...
- Question answering
- Mathematics
- Coding
- Reasoning, Planning, Decision-making, ...

Autoregressive Transformers

- Most LLMs follow the autoregressive design paradigm [Radford et al., 2019, Brown et al., 2020, OpenAI, 2023, Zhang et al., 2022, Touvron et al., 2023, Chowdhery et al., 2022, Rae et al., 2021, Scao et al., 2022].

Autoregressive Transformers

- Most LLMs follow the autoregressive design paradigm [Radford et al., 2019, Brown et al., 2020, OpenAI, 2023, Zhang et al., 2022, Touvron et al., 2023, Chowdhery et al., 2022, Rae et al., 2021, Scao et al., 2022].
- Main idea: various tasks can be uniformly treated as sequence generation problems.

Autoregressive Transformers

- Most LLMs follow the autoregressive design paradigm [Radford et al., 2019, Brown et al., 2020, OpenAI, 2023, Zhang et al., 2022, Touvron et al., 2023, Chowdhery et al., 2022, Rae et al., 2021, Scao et al., 2022].
- Main idea: various tasks can be uniformly treated as sequence generation problems.
- The input along with the task description can be together encoded as a sequence of tokens, called the *prompt*.

| | | |
|---|--------------------------------|--------------------|
| 1 | Translate English to French: | ← task description |
| 2 | sea otter => loutre de mer | ← examples |
| 3 | peppermint => menthe poivrée | ← |
| 4 | plush girafe => girafe peluche | ← |
| 5 | cheese => | ← prompt |

Autoregressive Transformers

- Most LLMs follow the autoregressive design paradigm [Radford et al., 2019, Brown et al., 2020, OpenAI, 2023, Zhang et al., 2022, Touvron et al., 2023, Chowdhery et al., 2022, Rae et al., 2021, Scao et al., 2022].
- Main idea: various tasks can be uniformly treated as sequence generation problems.
- The input along with the task description can be together encoded as a sequence of tokens, called the *prompt*.
- The answer is generated by predicting subsequent tokens conditioned on the prompt in an autoregressive way.

| | | |
|---|--------------------------------|--------------------|
| 1 | Translate English to French: | ← task description |
| 2 | sea otter => loutre de mer | ← examples |
| 3 | peppermint => menthe poivrée | ← |
| 4 | plush girafe => girafe peluche | ← |
| 5 | cheese => | ← prompt |

Chain of Thought Prompting (CoT)

- Crucial for tasks involving math or reasoning [Wei et al., 2022, Kojima et al., 2022, Suzgun et al., 2022, Nye et al., 2022, Wies et al., 2023].
- Two typical triggering methods:

(a) Zero-shot

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: The answer (arabic numerals) is

(Output) 8 X

(b) Few-shot

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) The answer is 8. X

(c) Zero-shot-CoT

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: **Let's think step by step.**

(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. ✓

(d) Few-shot-CoT

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are $16 / 2 = 8$ golf balls. Half of the golf balls are blue. So there are $8 / 2 = 4$ blue golf balls. The answer is 4. ✓

Questions Regarding CoT

- How can we theoretically understand the power of CoT generation?
- How can these prompts trigger the CoT generation? Can we design better prompting strategies to further exploit the power of LLMs?
- How can CoT emerge in LLMs trained over massive data?

Questions Regarding CoT

- How can we theoretically understand the power of CoT generation?
- How can these prompts trigger the CoT generation? Can we design better prompting strategies to further exploit the power of LLMs?
- How can CoT emerge in LLMs trained over massive data?

We focus on the first aspect by answering two central questions:

- Are there indeed *inherent* limitations of LLMs in directly solving math/reasoning tasks (without CoT)?
- What is the essential reason behind the success of CoT in boosting the performance of LLMs?

Index

- 1 Introduction
- 2 Preliminary**
- 3 CoT is the Key to Solving Math Problems
- 4 CoT is the Key to Solving General Problems
- 5 Experiments

Autoregressive Transformers

- Input: a sequence of tokens s of length n .
- Initial embedding: $\mathbf{X}^{(0)} = [\mathbf{v}_1 + \mathbf{p}_1, \dots, \mathbf{v}_n + \mathbf{p}_n]^\top \in \mathbb{R}^{n \times d}$, where
 - ▶ each input token s_i is converted to a d -dimensional vector $\mathbf{v}_i = \text{Embed}(s_i) \in \mathbb{R}^d$;
 - ▶ $\mathbf{p}_i \in \mathbb{R}^d$ is the positional embedding.
- Propagation: L Transformer blocks follow, each of which transforms the input by

$$\mathbf{X}^{(l)} = \mathbf{X}^{(l-1)} + \text{Attn}^{(l)}(\mathbf{X}^{(l-1)}) + \text{FFN}^{(l)}\left(\mathbf{X}^{(l-1)} + \text{Attn}^{(l)}(\mathbf{X}^{(l-1)})\right),$$

- ▶ $\text{Attn}^{(l)}$ is a multi-head self-attention layer;
- ▶ $\text{FFN}^{(l)}$ is a 2-layer feed forward network with GeLU activation.

$$\text{FFN}^{(l)}(\mathbf{X}) = \sigma(\mathbf{X} \mathbf{W}_1^{(l)}) \mathbf{W}_2^{(l)}.$$

Autoregressive Transformers

- Multi-head self attention:

$$\text{Attn}^{(l)}(\mathbf{X}) = \sum_{h=1}^H \text{softmax} \left(\mathbf{X} \mathbf{W}_Q^{(l,h)} (\mathbf{X} \mathbf{W}_K^{(l,h)})^\top + \mathbf{M} \right) \mathbf{X} \mathbf{W}_V^{(l,h)} \mathbf{W}_O^{(l,h)},$$

- ▶ The matrix $\mathbf{M} \in \{-\infty, 0\}^{n \times n}$ is a causal mask defined as $M_{ij} = -\infty$ iff $i < j$. This ensures that each position i can only attend to preceding positions $j \leq i$.
- Output: $\mathbf{X}_{n,:}^{(L)} \in \mathbb{R}^d$ is used to predict the token s_{n+1} (via a softmax classifier).
- Autoregressive generation: By concatenating s_{n+1} to the end of the input sequence s , the above process can be repeated. The process continues iteratively until a designated End-of-Sentence token is generated.

Index

- 1 Introduction
- 2 Preliminary
- 3 CoT is the Key to Solving Math Problems**
- 4 CoT is the Key to Solving General Problems
- 5 Experiments

Motivation

- Transformer-based LLMs exhibit surprising math abilities in various aspects [OpenAI, 2023, Bubeck et al., 2023].
- Can we gain insights into their power in fundamental math tasks?

Motivation

- Transformer-based LLMs exhibit surprising math abilities in various aspects [OpenAI, 2023, Bubeck et al., 2023].
- Can we gain insights into their power in fundamental math tasks?
- We consider two tasks: arithmetic and equation, which serve as elementary building blocks in solving complex real-world math problems.
- Two setup:
 - ▶ Directly generate the answer;
 - ▶ Generate a complete CoT solution.

Illustration using GPT4

Two different ways to prompt GPT4: “[Direct]” means directly generating the answer, and “[Chain]” means generating intermediate steps.

GD

Below I will ask you a few questions. When a question is prefixed with [Direct], please directly generate the answer WITHOUT giving any intermediate steps. On the contrary, when a question is prefixed with [Chain], please answer the question in details step by step, and give the final answer in the end.

Arithmetic: Direct

GD

[Direct] $3+(7-(7-2)\times(5-3))-(6-4\times(1+2))=?$



The result of the expression $3+(7-(7-2)\times(5-3))-(6-4\times(1+2))$ is -5.

Arithmetic: Chain of Thought



Sure, let's break this equation down following the order of operations, also known as BIDMAS/BODMAS (Brackets, Indices/Orders, Division and Multiplication (from left to right), Addition and Subtraction (from left to right)).

First, we'll resolve operations inside the brackets:

1. Inside the first set of brackets, we have $(7-2)$ which equals 5.
2. Inside the second set of brackets, we have $(5-3)$ which equals 2.
3. Inside the third set of brackets, we have $(1+2)$ which equals 3.

So the equation now looks like this: $3+(7-5\times 2)-(6-4\times 3)$

Next, we'll carry out the multiplication operations:

4. 5×2 equals 10.
5. 4×3 equals 12.

Our equation now becomes: $3+(7-10)-(6-12)$

Next, we'll do the subtraction:

6. $7-10$ equals -3.
7. $6-12$ equals -6.

Now the equation is: $3+(-3)-(-6)$

Finally, we'll do the addition and subtraction:

8. $3+(-3)$ equals 0.
9. $0-(-6)$ equals 6.

So, the result of the expression $3+(7-(7-2)\times(5-3))-(6-4\times(1+2))$ is 6.



Linear Equation: Direct

GD

[Direct] Solve the following linear equation:

$$3x + 3y + 12z = 6;$$

$$2x + 5y + 14z = 7;$$

$$2x + 4y + 15z = 6;$$



The solution to the system of linear equations is $x = 1$, $y = -1$, and $z = 0$.

Linear Equation: Chain of Thought

From the first simplified equation ($y + 2z = 1$), we can express $y = 1 - 2z$. Substitute y into the second simplified equation ($2y + 7z = 2$), we get:

$$2*(1 - 2z) + 7z = 2$$

$$2 - 4z + 7z = 2$$

$$3z = 0$$

$$z = 0$$

Substitute $z = 0$ back into $y + 2z = 1$, we get:

$$y + 2*0 = 1$$

$$y = 1$$

Substitute $y = 1$ and $z = 0$ into the first equation ($x + y + 4z = 2$), we get:

$$x + 1 + 4*0 = 2$$

$$x = 2 - 1$$

$$x = 1$$

So, the solution of this system of linear equations is $x = 1$, $y = 1$, and $z = 0$.

Problem Formulation

- We consider a simplified setting where all numbers are integers ranging from $\{0, \dots, p-1\}$ and arithmetic operations are performed in the finite field modulo p (p is a prime number).
- $\text{Arithmetic}(n, p)$: the task of evaluating arithmetic expressions (modulo p), where the input length is bounded by n .
- $\text{Equation}(m, p)$: the task of solving linear equations (modulo p) with no more than m variables.

| Arithmetic Expression | Linear Equations |
|--|--|
| <p>Input:</p> $(7 + 5) \div (6 + 4 \times 3 - 2 \times 7) =$ <p>Output:</p> $\begin{aligned} &12 \div (6 + 4 \times 3 - 2 \times 7) \\ &= 12 \div (6 + 12 - 2 \times 7) \\ &= 12 \div (18 - 2 \times 7) \\ &= 12 \div (18 - 14) \\ &= 12 \div 4 \\ &= 3 \end{aligned}$ | <p>Input:</p> $\begin{aligned} 3x + 3y + 12z &= 6; \\ 2x + 5y + 14z &= 7; \\ 2x + 4y + 15z &= 6; \end{aligned}$ \Rightarrow <p>Output:</p> $\begin{aligned} x + y + 4z &= 2; \\ 3y + 6z &= 3; \\ 2y + 7z &= 2; \end{aligned}$ $\Rightarrow \begin{aligned} x + 2z &= 1; \\ y + 2z &= 1; \\ 3z &= 0; \end{aligned}$ $\Rightarrow \begin{aligned} x &= 1; \\ y &= 1; \\ z &= 0; \end{aligned}$ |

Figure 1: Illustrations of CoT on two math tasks.

Log-precision Transformer

- We consider a practical setting called the log-precision Transformer.
- Intuitively, it refers to a Transformer whose internal neurons can only store floating-point numbers within a finite $O(\log n)$ bit precision where n is the maximal length of the input sequence.
- Example: 16/32 bits machine precision v.s. a maximal sentence length of 2048 in GPT.
- Why *log*-precision? It implies that the number of values each neuron can take is *polynomial* in the input length, which is both reasonable and a *necessary* condition for representing important quantities like positional embedding.

Negative Results

Theorem

Assume $\text{TC}^0 \neq \text{NC}^1$. For any prime number p , integer L , and any polynomial Q ,

- there exists a problem size n such that **no** log-precision autoregressive Transformer with depth L and hidden dimension $d \leq Q(n)$ can **directly** solve the problem $\text{Arithmetic}(n, p)$.
 - there exists a problem size m such that **no** log-precision autoregressive Transformer with depth L and hidden dimension $d \leq Q(m)$ can **directly** solve the problem $\text{Equation}(m, p)$.
- Our theorems imply that in order to directly output the answers, the size of the model will grow **super-polynomially** in the input length for both problems.

Key Insight: Circuit complexity

- TC^0 and NC^1 are two standard computation complexity classes, and it is widely believed that $TC^0 \subsetneq NC^1$:

$$NC^0 \subsetneq AC^0 \subsetneq TC^0 \subset NC^1 \subset P \subset NP.$$

- A pure Transformer represents a class of shallow circuits with complexity upper bounded by TC^0 .
- On the other hand, we prove that the complexity of both math problems above are lower bounded by NC^1 by applying *reduction* from NC^1 -complete problems:
 - ▶ Boolean Formula Evaluation Problem
 - ▶ Automaton Membership Testing
- Take away: the reason is not due to the (serialized) computational cost of these problems but rather to their *parallel complexity*!

How About generating a CoT solution?

Theorem

Fix any prime p . For any integer $n > 0$, there exists an autoregressive Transformer with constant hidden size d (independent of n), depth $L = 5$, and 5 heads in each layer that can generate the CoT solution for all inputs in $\text{Arithmetic}(n, p)$. Moreover, all parameter values in the Transformer are bounded by $O(\text{poly}(n))$.

Theorem

Fix any prime p . For any integer $m > 0$, there exists an autoregressive Transformer with constant hidden size d (independent of m), depth $L = 5$, and 5 heads in each layer that can generate the CoT solution for all inputs in $\text{Equation}(m, p)$. Moreover, all parameter values in the Transformer are bounded by $O(\text{poly}(m))$.

Insights into the Proof

- Our proof reveals the significance of several key components in the Transformer design:
 - ▶ One attention head can perform the following two basic operations: (conditional) copy and (conditional) reduction.
 - ▶ Multi-head attention can perform multiple copy/reduction operation in parallel.
 - ▶ The MLP can perform multiplication, linear transformation, conditional selection, and look-up table.
 - ▶ Residual connection can reserve the history information.
- We use these basic operations to form parallel algorithms that solve both math tasks.

Discussions

- The polynomial upper bound of parameters guarantees that the construction can be implemented using log-precision.
- These CoT derivations are purely written in a **readable** math language format, largely resembling how human write solutions.
 - ▶ In a broad sense, our findings provide evidence that LLMs have the potential to convey meaningful human thoughts through *grammatically precise* sentences.
- How can LLMs equipped with CoT bypass the impossibility results?
 - ▶ This can be understood via the *effective depth* of the Transformer circuit.
 - ▶ Employing CoT creates dependency between output tokens and leads to a significantly deeper circuit with depth proportional to the length of the CoT solution, yielding an expressivity far beyond TC^0 .

Index

- 1 Introduction
- 2 Preliminary
- 3 CoT is the Key to Solving Math Problems
- 4 CoT is the Key to Solving General Problems**
- 5 Experiments

CoT Can Implement Dynamic Programming

- We next switch our attention to a more general setting beyond mathematics.
- We find that LLMs with CoT are theoretically capable of emulating a powerful decision-making framework: *Dynamic Programming* (DP).
- Basic idea of DP: breaking down a complex problem into a series of small subproblems that can be tackled in a sequential manner.
- The decomposition ensures that there is a significant interconnection (overlap) among various subproblems, so that each subproblem can be efficiently solved by utilizing the answers (or other relevant information) obtained from previous ones.

CoT Can Implement Dynamic Programming

- Key concepts in DP:

- ▶ State space \mathcal{I} equipped with a partial ordering \prec
- ▶ Transition function T :

$$\text{dp}(i) = T(i, s, \{(j, \text{dp}(j)) : j \prec i\}),$$

This paper considers a restricted setting

$$\text{dp}(i) = f(i, s_{g_1(i)}, \dots, s_{g_J(i)}, \text{dp}(h_1(i)), \dots, \text{dp}(h_K(i))),$$

- ▶ Aggregation function A :

$$A(\{(i, \text{dp}(i)) : i \in \mathcal{I}\}, s) = u(\square_{i \in \mathcal{A}} \text{dp}(i)),$$

DP Examples

- Longest Increasing subsequence
- Edit Distance

| Problem | Longest increasing subsequence | Edit distance |
|----------------------|--|--|
| Input | A string s of length n | Two strings $s^{(1)}, s^{(2)}$ of length $n_1 = s^{(1)} $ and $n_2 = s^{(2)} $, concatenated together |
| State space | $\{(j, k) : j \in [n], k \in \{0, \dots, j-1\}\}$ | $\{0, \dots, n_1\} \times \{0, \dots, n_2\}$ |
| Transition function | $\text{dp}(j, k) = \begin{cases} 1 & \text{if } k=0 \\ \max(\text{dp}(j, k-1), \text{dp}(k, k-1) \times \mathbb{I}[s_j > s_k] + 1) & \text{if } k > 0 \end{cases}$ | $\text{dp}(j, k) = \begin{cases} ak & \text{if } j=0 \\ bj & \text{if } k=0 \\ \min(\text{dp}(j, k-1) + a, \text{dp}(j-1, k) + b, \text{dp}(j-1, k-1) + c\mathbb{I}[s_j^{(1)} \neq s_k^{(2)}]) & \text{otherwise} \end{cases}$ |
| Aggregation function | $\max_{i \in [n]} \text{dp}(i, i-1)$ | $\text{dp}(n_1, n_2)$ |

CoT Can Implement Dynamic Programming

We prove that autoregressive Transformers can generate the DP reasoning chain in the following format:

input 1 | \dots | input N | $(i_1, \text{dp}(i_1))$ \dots $(i_{|\mathcal{I}|}, \text{dp}(i_{|\mathcal{I}|}))$ final answer

Theorem (Informal)

Consider a DP problem satisfying some regularity assumptions. For any integer $n \in \mathbb{N}$, there exists an autoregressive Transformer with constant depth L , hidden dimension d and attention heads H (independent of n), such that the answer generated by the Transformer is correct for all input sequences s of length no more than n . Moreover, all parameter values are bounded by $O(\text{poly}(n))$.

Impossibility Results

- Many DP problems are intrinsically hard to be solved by a bounded-depth Transformer without CoT.
- One celebrate example is the Context-Free Grammar (CFG) Membership Testing, which tests whether an input string belongs to a pre-defined context-free language.

Theorem

Assume $TC^0 \neq P$. There exists a context-free language such that for any depth L and any polynomial Q , there exists a sequence length $n \in \mathbb{N}$ where no log-precision autoregressive transformer with depth L and hidden dimension $d \leq Q(n)$ can generate the correct answer for the CFG Membership Testing problem for all input strings of length n .

- Therefore, CoT significantly improves the expressiveness of LLMs, allowing them to solve even P-complete problems.

Index

- 1 Introduction
- 2 Preliminary
- 3 CoT is the Key to Solving Math Problems
- 4 CoT is the Key to Solving General Problems
- 5 Experiments**

Experiments

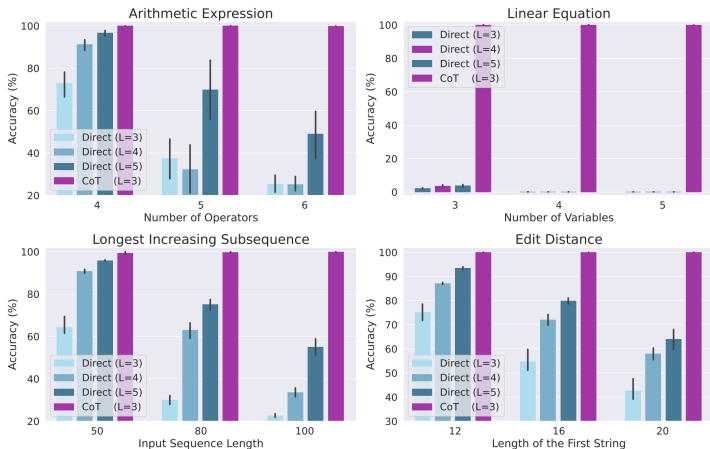


Figure 2: Model performance on different tasks. For all tasks and various difficulty levels, autoregressive Transformers with CoT consistently outperform Transformers trained on direct datasets. In particular, 3-layer Transformers already succeed in these tasks with almost perfect accuracy, while deeper Transformers ($L = 3, 4, 5$) trained on the direct datasets typically fail.

Experiments: Length Extrapolation

Trained on data with number of operators less than 16, and test on longer samples.

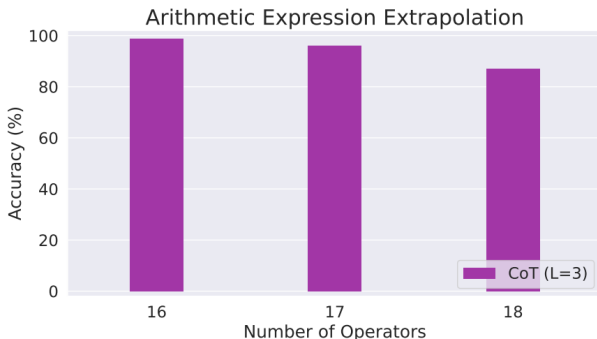


Figure 3: Performance of the length extrapolation experiment, tested on sequences that are longer than those in training.

Future Work

- How do prompts influence the CoT generation?
- How to understand the role of model size played in CoT generation quality [Wei et al., 2022]?
- How can CoT emerge in LLMs trained over massive data?
 - ▶ Generalization: it may be an important direction for theoretically studying how LLMs can generalize from CoT demonstrations (even in the out-of-distribution setting, e.g., length extrapolation).
 - ▶ Data: how can models learn CoT solutions when there are only limited CoT demonstrations in training (or even purely from direct datasets)?
- Designing better architectures to solve complex problems without resorting to CoT generation
 - ▶ Incorporating **memory**?

Thank You

Our paper can be found at
<https://arxiv.org/abs/2305.15408>

References I

- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Advances in neural information processing systems*, volume 33, pages 1877–1901, 2020.
- Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with GPT-4. *arXiv preprint arXiv:2303.12712*, 2023.
- Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.
- Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, 2022.

References II

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. Show your work: Scratchpads for intermediate computation with language models. In *Deep Learning for Code Workshop*, 2022.

OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*, 2021.

References III

- Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*, 2022.
- Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*, 2022.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022.

References IV

Noam Wies, Yoav Levine, and Amnon Shashua. Sub-task decomposition enables learning in sequence to sequence tasks. In *The Eleventh International Conference on Learning Representations*, 2023. URL <https://openreview.net/forum?id=BrJATVZDWEH>.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.