Can Visual Scratchpads With Diagrammatic Abstractions Augment LLM Reasoning?

Joy Hsu[†] Stanford University
Gabriel Poesia Stanford University
Jiajun Wu Stanford University
Noah D. Goodman Stanford University

Abstract

When humans reason about complex text-based questions, we leverage diagrammatic abstractions drawn on a visual scratchpad. In this paper, we introduce and explore the capabilities of Visual-Scratchpad, a method that augments a *large language foundation model* (LLM) with diagrammatic execution and readout. We enable the LLM to generate drawing commands and to readout abstractions from the resulting picture. The visual readout operation uses a *visual foundation model*, optionally finetuned with expert iteration. Here, we show that although Visual-Scratchpad outperforms an inference-only LLM, it surprisingly yields worse performance compared to a single finetuned LLM. Through experiments, we propose that this gap is due to the failure mode of vision foundation models in understanding abstractions in diagrams.

1 Introduction

Diagrams are a powerful tool for reasoning, abstraction, and knowledge organization. Indeed, humans turn to using visual scratchpads when solving a wide array of challenges, from puzzles to math problems. Studies have highlighted the importance of diagrams for deductive and logical reasoning Bauer and Johnson-Laird [1993], Allwein and Barwise [1996], mathematical reasoning Jamnik [2001], Dörfler [2001], and scientific reasoning Cheng and Simon [1995].

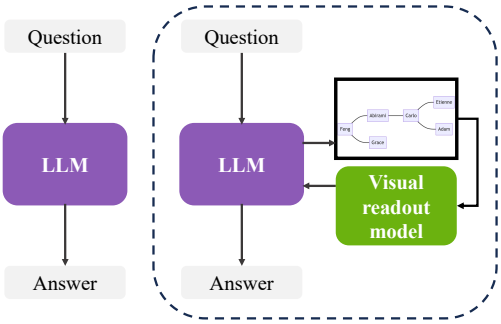

Figure 1: On the left is the canonical text-based task paradigm with an LLM, while on the right is the Visual-Scratchpad model, which interleaves execution from a language foundation model and a readout foundation model.

Can we equip large language models (LLMs) with a visual scratchpad for diagrammatic reasoning and thereby improve reasoning performance on *text-based* tasks? Many existing works have proposed chain-of-thought based LLM execution Wei et al. [2022], Wang et al. [2022], Zhang et al. [2022], which shows the importance of task decomposition for auto-regressive models, but only insofar as having scratchpads in the text domain. More recent works have explored augmenting LLMs with physics engines Liu et al. [2022], planning algorithms Hao et al. [2023], mathematical verifiers Poesia et al. [2023], and Python intepreters Gao et al. [2023], but their execution is constrained and they improve performance on directly related tasks of the same domain.

[†]Email: joycj@stanford.edu

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

In contrast, we introduce and explore the capabilities of Visual-Scratchpad, a framework that combines a *language foundation model* and a *visual readout foundation model* Bommasani et al. [2021], to leverage the benefits of diagrammatic abstractions for general reasoning tasks. To do so, we enable the LLM access to a visual scratchpad. The LLM can output draw commands to make annotations and then readout answers from the drawn diagram. Readout is by asking a visual foundation model to generate a natural language answer from an image and question. We hypothesize that there are many reasoning tasks in which it is easier to visually readout answers than to reason through text only. But importantly, the visual readout model must be able to interpret these abstractions. Can we teach a model to interpret diagrams as humans do, when humans are often trained with decades of embodied experiences? We use Pix2Struct Lee et al. [2023], a vision foundation model pretrained on a wide range of visual language understanding data, as our readout model. In addition, we further propose an expert iteration method that finetunes the base vision model with successful visual readout trajectories from a text-based train set.

We demonstrate that Visual-Scratchpad significantly improves upon an inference-only LLM baseline on questions about graphs and BIG-bench datasets that involve reasoning Srivastava et al. [2022]. However, we see that Visual-Scratchpad with interleaved execution of a language foundation model and readout foundation model performs *worse* than a finetuned LLM only. We conduct a series of experiments to show that visual readout for diagrammatic understanding, though simple for humans, is still a failure mode for vision foundation models. We conjecture that this is due to the fact that vision models lack the pretraining tasks and architecture that human vision is naturally evolved for, as well as lack the compute and data that language foundation models are trained with.

2 Related Work

2.1 Diagrams for abstraction

Many psychologists have studied the benefits of diagrams for human reasoning Bauer and Johnson-Laird [1993], Allwein and Barwise [1996], Kim et al. [2000], Jamnik [2001], Dörfler [2001], Chen and Herbst [2013], Abrahamsen and Bechtel [2015], Cheng and Simon [1995], and have shown that diagrams assist in abstraction, knowledge organization, and discovery of new conclusions. Visual-Scratchpad aims to enable LLMs to leverage the same benefits, and provides a solution for improving visual readout from diagrams.

Prior works in computer vision have introduced models that enable diagrammatic understanding. One salient work is Pix2Struct Lee et al. [2023], a pretrained image-to-text model for visual language understanding that serves as a readout foundation model. Pix2Struct is pretrained to learn to parse masked screenshots of web pages into simplified HTML, and learns signals such as OCR, language modeling, and image captioning Mishra et al. [2019], Masry et al. [2022], Bai et al. [2021], Li et al. [2020], Wang et al. [2021], Mathew et al. [2021, 2022]. However, it lacks the ability to reason about diagrammatic abstractions as Visual-Scratchpad requires. To augment the vision model, we leverage weights from finetuning Pix2Struct on the AI2 Diagrams dataset, a dataset of diagrams for question answering Kembhavi et al. [2016]. This dataset focuses on multiple choice questions which require parsing from annotations in diagrams. We build upon Pix2Struct with AI2D, and improve the readout foundation model further with expert iteration. Our expert iteration method finetunes the model from successful reasoning trajectories, requiring language supervision only for training the vision model.

2.2 Large language models with APIs

In recent years, several works have proposed giving LLMs access to pre-defined APIs. Liu et al. [2022] proposed leveraging a computational physics engine to simulate possible outcomes for physical reasoning tasks. Poesia et al. [2023] introduced a logic guide backed by a theorem-proving language for logical reasoning tasks. Hao et al. [2023] and Gao et al. [2023] integrated planning algorithms and Python interpreters respectively to steer LLMs. In contrast, we propose exploring a visual scratchpad with diagrammatic abstractions to augment LLM reasoning. Visual-Scratchpad enables autoregressive models to generate the appropriate abstractions to visualize. Notably, these diagrams are not designed for any specific domain, and hence can improve reasoning across different tasks.

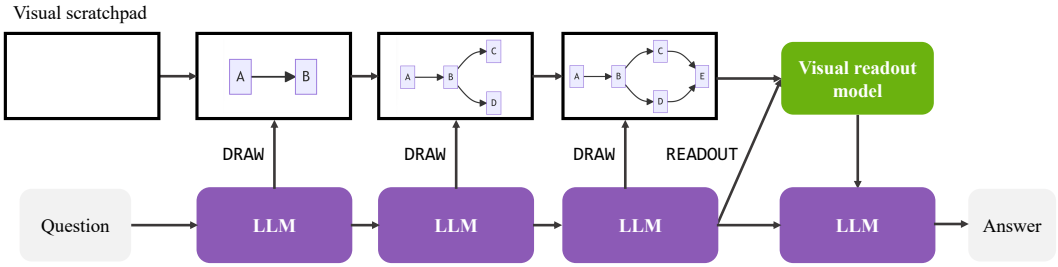

Figure 2: Visual-Scratchpad leverages a language foundation model to output step-by-step draw instructions on a scratchpad, and a readout foundation model to interpret the drawn diagram.

3 Methods

We explore the capabilities of Visual-Scratchpad, which consists of a core reasoning framework realized by a language foundation model, as well as draw and readout APIs realized by a readout foundation model. The reasoning framework is a combination of GPT with a minimal set of visual scratchpad usage examples that are neither constrained nor domain specific (Section 3.1). The draw API enables the framework to output commands that draw abstractions on a visual scratchpad (Section 3.2), while the readout API enables it to ask questions to the readout model, which extracts information from the visual scratchpad (Section 3.3). In Figure 2, we present the full framework.

3.1 Reasoning framework

We use GPT-3.5 Brown et al. [2020] as our reasoning framework backbone, and leverage it to (1) generate draw commands with the correct abstractions for the visual scratchpad, and (2) interpret the natural language response from the readout model to answer the given question.

We prompt GPT with API descriptions as well as *simple* and *general* examples of draw and readout usage. In these examples, diagrammatic draw commands are enclosed in brackets, and prefixed with a special token `DRAW`. Similarly, readout questions to the visual scratchpad are invoked with the token `READOUT`. Below, we show one of three examples specified in the prompt, which contains arbitrary entity names, as well as a simpler underlying diagram compared to our evaluation set. We show in experiments that GPT can generalize to more complex question types and diagrams.

Question: Wug is pointing to bac. bac is pointing to wom. wom is pointing to kij. Is wug pointing to kij?

Visual scratchpad context: The question is describing a sequence of relations between some items. Let's carefully go through and use draw commands for each of the relations. We first annotate wug pointing to bac `[DRAW: wug POINT bac]`. We then annotate bac pointing to wom `[DRAW: bac POINT wom]`. Finally, we annotate wom pointing to kij `[DRAW: wom POINT kij]`. To find the relation between wug and kij, we readout from the diagram `[READOUT: is node wug pointing to node kij?]`.

READOUT response: Yes.

Visual scratchpad context: For the question "Is wug pointing to kij?", the diagram tells us that node wug points to node kij, hence the answer is Yes. Answer: Yes.

3.2 Drawing on a visual scratchpad

We enable the LLM to draw to a visual scratchpad with commands backed by the Mermaid diagramming and charting tool library Sveidqvist [2014], Dubovitskiy [2023]. The draw commands include annotating abstractions such as labelled nodes, as well as connections between nodes (e.g., unconnected, connected, directed). See Figure 3 for example images of the visual scratchpad.

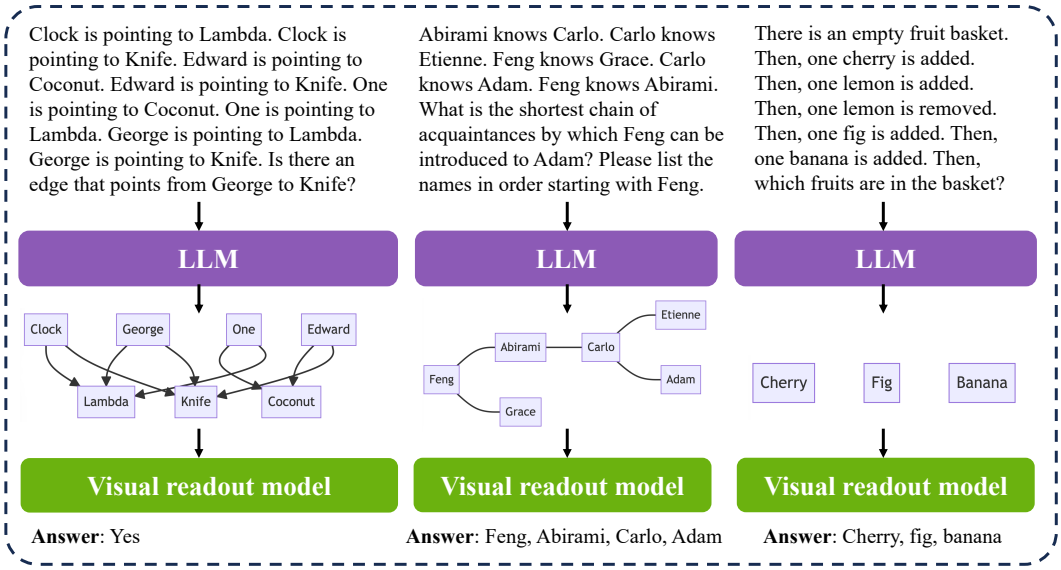

Figure 3: Examples of Visual-Scratchpad executing draw commands on a scratchpad.

We render the full image when the readout command is evoked with the Mermaid diagramming library. This visual scratchpad image then becomes input to the readout model, along with the question generated by the LLM.

3.3 Readout from a visual scratchpad

The Visual-Scratchpad framework finetunes a visual readout foundation model to better understand abstractions in the scratchpad image. We use Pix2Struct Lee et al. [2023], a visual question answering model pretrained on a variety of visual language understanding tasks, and finetune it to more accurately interpret abstractions from the drawn diagram given a question generated by the LLM. To do so, we propose expert iteration on a training dataset to train the vision model. The model is first finetuned on noisy GPT generated data, to then bootstrap learning from the text-based train set.

We generate question and answer pairs with GPT, by tasking the LLM to first create a graph with diverse node names and of arbitrary complexity, then generate corresponding question and answers. We specify only one example of the downstream question, and encourage GPT to generate data that follows a more faithful question and answer distribution to the target task. We prompt GPT to generate 5 diverse question and answer pairs in this way, until a target number of 5000 train and 500 validation examples is achieved.

In expert iteration, we first train a base Pix2Struct model with noisy GPT generated data, then use this model to find successful, expert readout trajectories on the training dataset. Visual-Scratchpad then further finetunes itself on this refined data, to decrease the effect of noisy data on the readout model. See Algorithm 1 for a detailed description. Notably, Visual-Scratchpad only requires text-based data to train the visual readout foundation model, by collecting scratchpad image and readout answer pairs that enable the LLM to correctly answer the question. The scratchpad is automatically generated by the LLM, and the readout answer by the vision foundation model. Although this training procedure requires a dataset from the downstream task, the training requires finetuning of Pix2Struct, instead of the LLM itself, which can be costly.

4 Experiments

We present analyses of Visual-Scratchpad on three datasets with different text-based reasoning tasks. In Section 4.1, we examine comparisons of this interleaved execution of language and visual foundation model to prior LLMs. In Section 4.2, we present ablations of Visual-Scratchpad across six variations, to better understand the framework's capabilities and failure modes.

Algorithm 1 Visual-Scratchpad with expert iteration.

Input: D_{train}: the labeled train question answering dataset; f: LLM backbone in Visual-Scratchpad; $\mathcal{R}(v, q; \theta)$: readout model; e single question example from test dataset.
Output: θ: $\mathcal{R}(v, q; \theta)$.

Generate visual question answering data D_{gen} from f with e in the prompt.
Pretrain $\mathcal{R}$ with D_{gen}.

$D_{\text{expert}} \leftarrow []$
for $i \leftarrow 0$ **do**
    $(q_{\text{train}}, a_{\text{train}}) \sim D_{\text{train}}$
    $v, q \leftarrow f(q_{\text{train}})$
    $a \leftarrow f(\mathcal{R}(v, q; \theta))$
    if $a = a_{\text{train}}$ **then**
        Update D_{expert} with $(v, q, \mathcal{R}(v, q; \theta))$.
    end if
end for
Finetune $\mathcal{R}$ with D_{expert}.

Table 1: Comparison of Visual-Scratchpad to prior work; Visual-Scratchpad outperforms an inference only LLM model, but yields worse performance compared to a finetuned LLM model.

Method	Train	Diagrams	Acquaintances	Fruit basket
GPT-3.5	✗	0.42	0.66	0.60
GPT-3.5-Finetuned	✓	0.88	0.90	1.00
GPT-3.5-Visual-Scratchpad	✓	0.78	0.88	0.92

We evaluate Visual-Scratchpad on a Diagrams dataset, which we generated with questions about different types of graphs, as well as two BIG-bench Srivastava et al. [2022] datasets that require multi-step reasoning. The Diagrams dataset is constructed with base graphs of the following forms: unconnected, connected, and directed. Questions include those that ask for a viable path between two nodes in a graph, existence of directed edges, number of nodes in the diagram, etc. The questions and answers are procedurally generated. Additionally, we test on the BIG-bench datasets: Acquaintances and Fruit basket. They contain questions describing complex relations, and require natural language answers such as ordering of entities. All datasets include 5000 train, 500 val, and 50 test examples, following the evaluation dataset of BIG-bench. We use exact match as an accuracy metric.

4.1 Comparison to prior work

We compare Visual-Scratchpad to the inference-only base GPT model and the finetuned GPT model. In Table 1, we see that the Visual-Scratchpad framework significantly improves upon the base GPT model on all tasks, demonstrating that LLMs augmented with a visual scratchpad of diagrammatic abstractions can indeed reliably reason about text-based tasks more effectively.

However, in comparison to a finetuned GPT model on the *same* training dataset, Visual-Scratchpad yields surprisingly *worse* performance. Contrary to our hypothesis that two interleaved foundation models would improve performance, the single finetuned LLM achieves higher accuracy. Errors that Visual-Scratchpad makes include incorrectly reading out node names, repeating node names multiple times in readout answers, and misidentifying edges between nodes in the diagram. We further explore this failure mode in the analyses below.

4.2 Ablations

We present ablation results showing that the visual readout foundation model in Visual-Scratchpad underperforms in comparison to the LLM. To explore this, we conduct experiments on three settings: Visual-Scratchpad with (1) a readout model that does not require a train set, with (2) a readout model that *does* require a train set, and with (3) an oracle readout model, backed by human vision.

Table 2: Ablations of Visual-Scratchpad on three main settings: (1) without access to a train set, (2) with access, as well as (3) with an oracle readout model.

Method	Train	Diagrams	Acquaintances	Fruit basket
Visual-Scratchpad + AI2D	✗	0.16	0.10	0.12
Visual-Scratchpad + GPT gen	✗	0.28	0.10	0.10
Visual-Scratchpad + GPT task-specific	✗	0.30	0.46	0.50
Visual-Scratchpad + pseudo-label	✓	0.74	0.86	0.88
Visual-Scratchpad + expert iteration	✓	0.78	0.88	0.92
Visual-Scratchpad + oracle readout	-	0.98	0.94	1.00

For the first setting which requires an inference dataset only, we compare performance of three Visual-Scratchpad variants. The first is Visual-Scratchpad with a base Pix2Struct readout model pretrained on AI2D (Visual-Scratchpad + AI2D), the second is with the readout model finetuned on GPT generated data (Visual-Scratchpad + GPT gen), and the third is with the readout model finetuned on GPT generated data with one example of the task-specific downstream question (Visual-Scratchpad + GPT task-specific). In Table 2, we see that the readout foundation model with AI2D weights yields surprisingly poor results, despite pretraining across a wide range of visual language understanding tasks, and finetuning on a diagrammatic domain. In comparison, finetuning with a GPT generated task-specific dataset improves the performance significantly, while only requiring one example question on the downstream task.

For the second setting which requires a train set, we ablate Visual-Scratchpad with a readout model finetuned through expert iteration (only finetuning on scratchpad and readout pairs that yield correct text responses), and with a readout model finetuned through pseudo-labelling on the full train set (finetuning on all examples, relying on the draw commands to be accurate). We see that having access to a train dataset to finetune the readout foundation model significantly improves accuracy. In addition, we can rely on the LLM's ability to accurately decompose the task into correct abstractions in the visual scratchpad, to generate a dataset for the vision model while only having labelled question and answer pairs in text.

Importantly, in Table 2, we show that in the third setting of Visual-Scratchpad with an oracle readout model, we achieve significantly improved performance and higher accuracy across tasks compared to a finetuned LLM in Table 1. Our results demonstrate that Visual-Scratchpad's performance is constrained by the abilities of the readout foundation model; with a visual readout model that matches human performance, Visual-Scratchpad serves as a powerful framework for reasoning.

5 Discussion

Although Visual-Scratchpad outperforms an inference-only LLM model, it still surprisingly yields worse performance than a simple finetuned LLM. Our experiments show that this is due to the visual readout foundation model's failure mode in interpreting diagrams. Though in many tasks, humans leverage vision to do reasoning, abstraction, and knowledge organization, visual foundation models have not been sufficiently developed to achieve the same level of performance. Indeed, even the recently announced GPT-4 Vision OpenAI [2023] has noted limitations such as missing symbols when processing complex images, for example, diagrams with text and detailed components. We conjecture that vision models can be improved with the correct set of pretraining tasks, data, and architecture that resemble human evolution and training; once they reach the level of language foundation models, Visual-Scratchpad should prove to be a successful paradigm for reasoning.

6 Conclusion

Inspired by the human use of diagrams for reasoning, we introduce and explore the capabilities of Visual-Scratchpad, a framework that reasons with language and visual readout foundation models, on a scratchpad with diagrammatic abstractions. We demonstrate that our framework enables foundation models to create and interpret visual abstractions as intermediate reasoning steps, and can improve performance of inference-only LLMs in text-based domains. However, importantly we examine the intriguing phenomenon of a single finetuned LLM outperforming Visual-Scratchpad, and provide analyses of this failure mode with our current language and vision foundation models.

References

Adele Abrahamsen and William Bechtel. Diagrams as Tools for Scientific Reasoning. *Review of Philosophy and Psychology*, 6:117–131, 2015.

Gerard Allwein and Jon Barwise. *Logical Reasoning with Diagrams*. Oxford University Press, 1996.

Chongyang Bai, Xiaoxue Zang, Ying Xu, Srinivas Sunkara, Abhinav Rastogi, Jindong Chen, et al. UIBert: Learning Generic Multimodal Representations for UI Understanding. *arXiv preprint arXiv:2107.13731*, 2021.

Malcolm I Bauer and Philip N Johnson-Laird. How Diagrams Can Improve Reasoning. *Psychological Science*, 4(6):372–378, 1993.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the Opportunities and Risks of Foundation Models. *arXiv preprint arXiv:2108.07258*, 2021.

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. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.

Chia-Ling Chen and Patricio Herbst. The Interplay Among Gestures, Discourse, and Diagrams in Students' Geometrical Reasoning. *Educational Studies in Mathematics*, 83(2):285–307, 2013.

Peter CH Cheng and Herbert A Simon. Scientific Discovery and Creative Reasoning with Diagrams. *The Creative Cognition Approach*, pages 205–228, 1995.

Willi Dörfler. Diagrams as Means and Objects of Mathematical Reasoning. In *Annual Conference on Didactics of Mathematics*, pages 39–49. Citeseer, 2001.

Kirill Dubovitskiy. Show Me ChatGPT Plugin. `https://github.com/bra1nDump/show-me-chatgpt-plugin`, 2023.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: Program-aided Language Models. In *International Conference on Machine Learning*, pages 10764–10799. PMLR, 2023.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with Language Model is Planning with World Model. *arXiv preprint arXiv:2305.14992*, 2023.

Mateja Jamnik. *Mathematical Reasoning with Diagrams*. University of Chicago Press Chicago, 2001.

Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Minjoon Seo, Hannaneh Hajishirzi, and Ali Farhadi. A Diagram is Worth a Dozen Images. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 235–251. Springer, 2016.

Jinwoo Kim, Jungpil Hahn, and Hyoungmee Hahn. How Do We Understand a System with (So) Many Diagrams? Cognitive Integration Processes in Diagrammatic Reasoning. *Information Systems Research*, 11(3):284–303, 2000.

Kenton Lee, Mandar Joshi, Iulia Raluca Turc, Hexiang Hu, Fangyu Liu, Julian Martin Eisenschlos, Urvashi Khandelwal, Peter Shaw, Ming-Wei Chang, and Kristina Toutanova. Pix2Struct: Screenshot Parsing as Pretraining for Visual Language Understanding. In *International Conference on Machine Learning*, pages 18893–18912. PMLR, 2023.

Yang Li, Gang Li, Luheng He, Jingjie Zheng, Hong Li, and Zhiwei Guan. Widget Captioning: Generating Natural Language Description for Mobile User Interface Elements. *arXiv preprint arXiv:2010.04295*, 2020.

Ruibo Liu, Jason Wei, Shixiang Shane Gu, Te-Yen Wu, Soroush Vosoughi, Claire Cui, Denny Zhou, and Andrew M Dai. Mind's Eye: Grounded Language Model Reasoning through Simulation. *arXiv preprint arXiv:2210.05359*, 2022.

Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. ChartQA: A Benchmark for Question Answering about Charts with Visual and Logical Reasoning. *arXiv preprint arXiv:2203.10244*, 2022.

Minesh Mathew, Dimosthenis Karatzas, and CV Jawahar. DocVQA: A Dataset for VQA on Document Images. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2200–2209, 2021.

Minesh Mathew, Viraj Bagal, Rubèn Tito, Dimosthenis Karatzas, Ernest Valveny, and CV Jawahar. InfographicVQA. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1697–1706, 2022.

Anand Mishra, Shashank Shekhar, Ajeet Kumar Singh, and Anirban Chakraborty. OCR-VQA: Visual Question Answering by Reading Text in Images. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 947–952. IEEE, 2019.

OpenAI. GPT-4V(ision) System Card. `https://cdn.openai.com/papers/GPTV_System_Card.pdf`, 2023.

Gabriel Poesia, Kanishk Gandhi, Eric Zelikman, and Noah D Goodman. Certified Reasoning with Language Models. *arXiv preprint arXiv:2306.04031*, 2023.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. Beyond the Imitation Game: Quantifying and Extrapolating the Capabilities of Language Models. *arXiv preprint arXiv:2206.04615*, 2022.

Knut Sveidqvist. Mermaid Diagramming and Charting Tool. `https://mermaid.js.org`, 2014.

Bryan Wang, Gang Li, Xin Zhou, Zhourong Chen, Tovi Grossman, and Yang Li. Screen2Words: Automatic Mobile UI Summarization with Multimodal Learning. In *The 34th Annual ACM Symposium on User Interface Software and Technology*, pages 498–510, 2021.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency Improves Chain of Thought Reasoning in Language Models. *arXiv preprint arXiv:2203.11171*, 2022.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought Prompting Elicits Reasoning in Large Language Models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. Automatic Chain of Thought Prompting in Large Language Models. *arXiv preprint arXiv:2210.03493*, 2022.