Whiteboard-of-Thought: Thinking Step-by-Step Across Modalities

Sachit Menon
Columbia University
sachit.menon@columbia.edu

Richard Zemel
Columbia University
zemel@cs.columbia.edu

Carl Vondrick
Columbia University
vondrick@cs.columbia.edu

whiteboard.cs.columbia.edu

Abstract

When presented with questions involving visual thinking, humans naturally switch reasoning modalities, often forming mental images or drawing visual aids. Large language models have shown promising results in arithmetic and symbolic reasoning by expressing intermediate reasoning in text as a chain of thought, yet struggle to extend this capability to answer text queries that are easily solved by visual reasoning, even with extensive multimodal pretraining. We introduce a simple method, *whiteboard-of-thought* prompting, to unlock the visual reasoning capabilities of multimodal large language models across modalities. Whiteboard-of-thought prompting provides multimodal large language models with a metaphorical 'whiteboard' to draw out reasoning steps as images, then returns these images back to the model for further processing. We find this can be accomplished with no demonstrations or specialized modules, instead leveraging models' existing ability to write code with libraries such as Matplotlib and Turtle. This simple approach shows state-of-the-art results on four difficult natural language tasks that involve visual and spatial reasoning. We identify multiple settings where GPT-4o using chain-of-thought fails dramatically, including more than one where it achieves 0% accuracy, while whiteboard-of-thought enables up to 92% accuracy in these same settings. We present a detailed exploration of where the technique succeeds as well as its sources of error.

1 Introduction

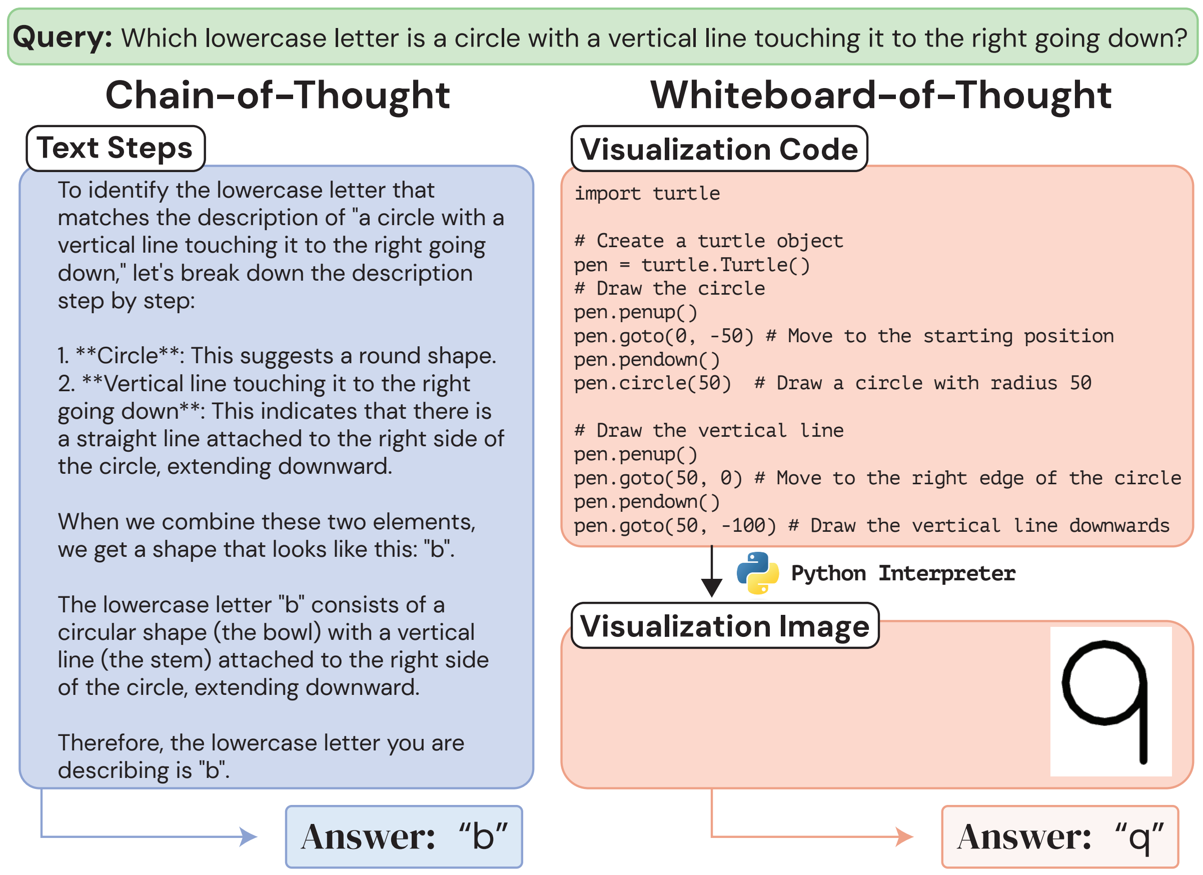
Which lowercase letter is a circle with a vertical line touching it to the right going down?

This question may sound rather trivial. Likely, you solved it by forming a mental image as you proceeded through the sentence, first placing the circle then adding the line to recognize the letter 'q.' If there had been more pieces to keep track of, you might instead resort to pen and paper, but follow a similar process.

Humans excel at this type of *visual thinking*. We readily interweave reasoning in words and reasoning in images to solve problems and communicate ideas (Tversky, 2011); we form images not only for directly visual reasoning, but also to draw maps during spatial navigation (Card et al., 1999) and even process abstract concepts (Bobek and Tversky, 2016).

Large language models (LLMs) have revolutionized NLP (Brown et al., 2020; Vaswani et al., 2017), showing the power of scaling parameter count and training data in all sorts of reasoning tasks. Chain-of-thought (CoT) prompting (Wei et al., 2022; Kojima et al., 2023), which allows a language model to solve complex problems by writing out intermediate steps as text before producing a final answer, and similar techniques have shown huge promise in tasks such as arithmetic and symbolic reasoning. It is thus natural to ask: can LLMs solve tasks that we, as humans, solve with visual thinking?

We find that for some tasks involving visual and spatial reasoning, even the best LLMs fail dramatically. In Fig. 1, we see GPT-4o (OpenAI et al., 2023) claim the answer to our earlier question is in fact 'b'; it does not perform the correct implicit visual reasoning about left, right, up, and down to reach the correct answer. 'b' was observed sampling deterministically; with temperature, it picks randomly among 'b', 'p', 'd', and 'q', i.e., the letters with the correct features but in different arrangements. It references the 'bowl' and the 'stem' to narrow down potential options then fails to perform the spatial reasoning to determine what the described orientation *means*. A formal background in typography should not be needed to answer the question; anyone who can create the correct visual with a basic level of visual recognition will solve it easily. This highlights the significant differences between the two reasoning processes.



Query: Which lowercase letter is a circle with a vertical line touching it to the right going down?

To identify the lowercase letter that matches the description of "a circle with a vertical line touching it to the right going down," let's break down the description step by step:

1. **Circle**: This suggests a round shape.
2. **Vertical line touching it to the right going down**: This indicates that there is a straight line attached to the right side of the circle, extending downward.

When we combine these two elements, we get a shape that looks like this: "b".

The lowercase letter "b" consists of a circular shape (the bowl) with a vertical line (the stem) attached to the right side of the circle, extending downward.

Therefore, the lowercase letter you are describing is "b".

```
import turtle

# Create a turtle object
pen = turtle.Turtle()
# Draw the circle
pen.penup()
pen.goto(0, -50) # Move to the starting position
pen.pendown()
pen.circle(50)  # Draw a circle with radius 50

# Draw the vertical line
pen.penup()
pen.goto(50, 0) # Move to the right edge of the circle
pen.pendown()
pen.goto(50, -100) # Draw the vertical line downwards
```

Figure 1: For queries that are trivial with visual reasoning, chain-of-thought (which produces the text on the left) can fail in surprising ways. Whiteboard-of-thought (which produces the code, image, and text on the right) provides an alternative to perform intermediate reasoning with images.

Our key idea is that visual reasoning tasks demand visuals. We leverage the abilities of multimodal large language models (MLLMs), which accept inputs not only in text but other modalities such as images, to achieve this. We show that providing MLLMs the ability to create and reason with explicit visuals – like a whiteboard showing intermediate thoughts – unlocks capabilities resembling visual thinking.

We thus introduce ‘**whiteboard-of-thought** (WoT)’: we provide MLLMs with a metaphorical ‘whiteboard’ to draw out the results of intermediate reasoning steps as *images*, then prompt them to use their visual input capabilities to produce answers or perform further reasoning from the visuals made by the model itself. We find that leveraging models’ existing ability to write code with visual graphics libraries such as Turtle and Matplotlib proves sufficient to create visuals useful for solving visual reasoning tasks without requiring a single example.

We demonstrate the potential of this idea on three BIG-Bench (Srivastava et al., 2022) tasks involving understanding ASCII art, as well as a recent difficult benchmark probing spatial reasoning abilities (Yamada et al., 2024), establishing a large perfomance gap between WoT and CoT. We further breakdown what types of problems are best suited to performing reasoning on visual tokens rather than text tokens. Finally, we identify current limitations of MLLM abilities and provide a detailed analysis of where WoT fails.

2 Preliminaries

We provide a brief overview of the main concepts underlying this work here.

LLMs and MLLMs Large language models have seen substantial success across a wide range of natural language tasks by scaling data and parameter counts (Brown et al., 2020; Vaswani et al., 2017). Recent work has extended advances in language modeling to the multimodal input setting, leading to multimodal large language models (MLLMs). These models generate text conditioned on not other text as context, but also inputs from other modalities, enabling tasks such as captioning or visual question answering. Of particular note for this work, the large amount of image-and-text data available on the Internet has enabled this effort to prove especially successful for image inputs, achieving state-of-the-art results across a wide range of computer vision tasks (Li et al., 2023; Alayrac et al., 2022; Liu et al., 2023).

Chain of thought prompting Despite the successes of scaling for large language models, directly producing answers to complex, multi-step reasoning tasks, such as arithmetic or symbolic tasks, historically has remained difficult. Wei et al. (2022) introduced a simple technique to substantially improve LLMs' performance on these tasks: few-shot chain-of-thought, prompting the model to break down complex queries and answer step-by-step by providing examples of the desired step-by-step reasoning as context. Kojima et al. (2023) broke this reliance on in-context exemplars to develop zero-shot chain-of-thought (in this work, referred to as simply 'chain-of-thought' or CoT). Instead of using handwritten examples, they perform two simple steps to elicit step-by-step reasoning. First, they prompt the model with the query and an additional instruction to 'think step by step'; the model then generates text comprising a reasoning trace. Second, they feed the reasoning trace back to the model, along with the instruction to produce an answer. This approach highlighted the substantial potential benefits of intermediate reasoning in the attractive zero-shot setting.

Tool augmented large language models Other work has shown that large language models can be induced to use external tools, such as calculators, to aid this intermediate reasoning. Nye et al. (2021) provide language models with a scratchpad: a dedicated buffer of text designated for intermediate computation, trained to mimic Python code execution. Gao et al. (2023b) and Chen et al. (2023) instead use LLMs' ability to write code to delegate simple computation to the Python interpreter, providing the computed results back to the LLM as text. To the best of our knowledge, we are the first to consider this ability's potential for generating visualizations to aid with intermediate reasoning.

3 Whiteboard-of-Thought

The goal of this work is to equip MLLMs with the ability to create images and visually process them to better answer queries. Our approach operates this whiteboard by synthesizing drawing code, executing this code to create the drawing, and parsing the resulting image before producing a final answer. Fig. 1 shows an example of the full procedure.

Creating visuals with MLLMs. Current MLLMs typically do not inherently possess the ability to produce outputs in the visual domain. Instead, we will show how we can create visuals using a model that only produces text.

The images we create for visual reasoning tend to be minimal, abstract, and symbolic (Tversky, 2011). We use code as a natural way to create such visuals. Leveraging what models already know about common Python libraries like Matplotlib or Turtle enables this capability to emerge zero-shot, without needing any specialized, hand-crafted modules (though these could be made to adapt the technique to specific domains). We discuss other approaches such as text-to-image models in the Appendix.

In order to generate the code, we provide the MLLM with the query and prompt it to write code to visualize it. For each query, we prompt the MLLM with the input `You write code to create visualizations using the {Matplotlib/Turtle} library in Python, which the user will run and provide as images. Do NOT produce a final answer to the query until considering the visualization.` along with the query. The model then decides what visualization code to write based on the query. Full prompting and inference details for code generation can be found in the Appendix.

The resulting code is then passed to a runtime environment to render it in image form. In this case, we use the Python interpreter with the previously-mentioned visualization libraries.

Processing the generated visuals. To process the resulting image, we make use of the MLLM's intrinsic multimodal input capacity. This obviates the need for external tools, like handcrafted visual modules (Gupta and Kembhavi, 2022; Surís et al., 2023), leading to a tightly self-contained method.

In summary, the model is given a prompt including the query, the knowledge that it can use the aforementioned visualization libraries, and that it will be provided the resulting images along with the query to perform further steps towards producing a final answer. It produces code, which is then executed to create an image. The resulting visual is returned back to the model to perform the next step or produce an answer.

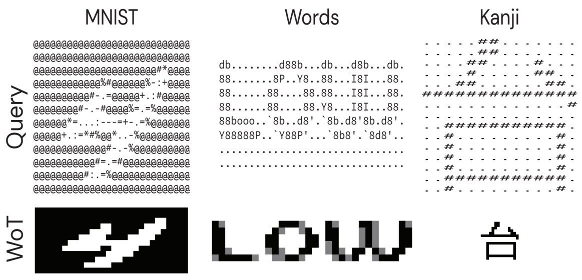


Figure 2: Example queries for each of the three ASCII understanding BIG-Bench tasks (Srivastava et al., 2022) we consider, along with the WoT visualization for each.

	MNIST	Word	Kanji
Direct	19.6	24.8	1.1
CoT	21.6	27.2	1.1
WoT (ours)	**66.0**	**66.4**	**73.8**

Table 1: **ASCII recognition accuracy.** MLLMs fail to perform the task with text alone. WoT unlocks visual processing to achieve substantial gains.

4 Experiments

We conduct experiments in two broad classes of natural language tasks that involve visual reasoning. First, we consider three datasets from BIG-Bench (Srivastava et al., 2022) that involve understanding information represented as ASCII text graphics. Next, we consider tasks involving natural language navigation under different spatial conditions (Yamada et al., 2024).

We perform all experiments in the zero-shot regime, and compare to two baselines without visualization: directly prompting the model for an answer ('Direct') and zero-shot chain-of-thought (Kojima et al., 2023) ('CoT'). We use a temperature of 0 and greedy decoding for generation. For all experiments, we use GPT-4o (gpt-4o-2024-05-13) as the underlying MLLM as it has each of the necessary capabilities enabling our model and the baselines – zero-shot chain-of-thought as well as the ability to produce code outputs and to accept image inputs. Full prompts and other generation details can be found in the Appendix.

4.1 ASCII Understanding

We begin with a clearly visual task found in BIG-Bench: ASCII understanding. Recent work has shown that even the strongest language models struggle to recognize ASCII representations, and that this failure can even be exploited to perform highly effective jailbreak attacks resulting in unintended and unsafe behaviors that bypass state-of-the-art defense techniques (Jiang et al., 2024).

ASCII art highlights our ability to subconsciously switch between processing modalities: it requires reinterpreting characters that usually possess some natural language interpretation (e.g., '=' as an equals sign) in a visual context, focusing on their arrangement and spatial relationships (e.g., '======' as a horizontal line). For humans, written text is typically processed with the same input modality as images (our eyes), allowing us to engage in visual thinking without any intermediate processing.

Consider the difficulty of understanding ASCII art being read aloud. This can be thought of as similar to how LLMs process ASCII: as text tokens, distinct from any visual tokens they may be able to process if they have multimodal capabilities. Thus, ASCII presents an interesting testing ground for evidence of visual thinking in MLLMs.

We consider three domains of ASCII understanding each comprising a task in BIG-Bench (Srivastava et al., 2022): ASCII MNIST digit recognition, ASCII word recognition, and ASCII kanji (Japanese logographic character) recognition. Examples of each of these can be found in Fig. 2 (along with the WoT visualization for each). Dataset and evaluation details can be found in the Appendix.

Results can be found in Table 1. We find that state of the art MLLMs are largely incapable of performing visual representation on these textual inputs. Prompting for step-by-step reasoning in words provides little benefit. However, providing a whiteboard to enable models to create and consider their own visualizations unlocks the visual thinking abilities latent in the MLLM, leading to substantial performance improvements.

Text-based approaches vs visualization At first glance, the baseline performance on the MNIST and word recognition tasks may seem surprisingly high, potentially giving the impression that with sufficient scale, text alone could conceivably solve these tasks. Upon deeper inspection of where the baselines succeed compared to where WoT succeeds, however, this notion quickly falls apart.

Examining the word recognition task, we find that the text-only baselines fail to solve *every* instance that actually requires visual understanding, i.e., that does not directly use the characters from

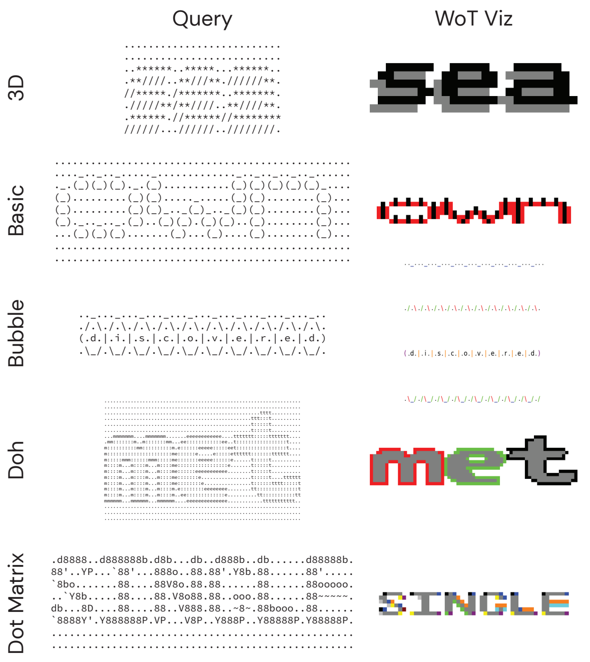


Figure 3: The different forms of ASCII in the BIG-Bench ASCII Word Recognition task and the visualizations made by WoT. Note that 'Bubble' simply includes the word with some additional characters, and 'Doh' forms the shape of each letter out of itself. (CoT results: 'ascii', 'hello', 'discovered', 'meet', 'goodbye'.) Best viewed with zoom.

the word to form the ASCII string. In more detail, the BIG-Bench ASCII word recognition task uses five forms of ASCII, shown in Fig. 3 along with visualizations made by WoT for each. Note that 'Bubble' simply includes the word with some additional characters, and 'Doh' forms the shape of each letter out of itself. In other words, they can be solved without considering the queries visually. We see in Table 2 that the only examples which the text-based approaches recognize come from these categories. The small boost CoT provides comes entirely from the 'Doh' category, which can be solved with linguistic reasoning. On the other hand, WoT achieves dramatically higher performance on every category other than 'Bubble', that being the only fully 'non-visual' of the five.

Does ASCII understanding necessitate 'reasoning'? One might contend that, for the ASCII understanding task, 'constructing' a helpful visual could simply be rendering the ASCII directly as an image, i.e., a fixed procedure with no dynamic reasoning involved. At a motivational level, we are using these tasks to evaluate our more general-purpose method; the goal is not to devise an ASCII-specific technique. Nevertheless, we conduct an experiment comparing to this fixed baseline for the word recognition task. That is, rather than generating code on a per-query basis to create visualizations as in WoT, for this baseline we render the text directly as images and evaluate whether MLLMs can understand these images directly.

We find that this approach results in an accuracy of 22.0% – comparable to (and in fact lower than) the text-only baseline. Why? We manually inspect the results, and find the errors fall into two broad qualitative categories. First, ASCII art rendered as text seems to result in some level of task confusion, similar to 'typographic attacks' (Noever and Noever, 2021; Menon et al., 2022; Goh et al., 2021; Materzynska et al., 2022; Ilharco et al., 2022), between performing the intended task and the character-level OCR task. Second, many of the produced outputs are 'technology'-related, for instance 'hello world' or 'Google'; we hypothesize this may stem from the style of ASCII writing may be spuriously correlated with these concepts on the Internet. At a high level, drawing yourself a confusing visual is worse than drawing no visual at all.

On the other hand, WoT allows the model to reason about how to render the visual. As seen in Fig. 3, different types of ASCII input can be better served by different visuals. For instance, direct rendering may be suitable for 'Bubble', while others such as '3D' can be made more legible – even for humans – with more careful visualization.

Error analysis One of the benefits WoT presents is the ease of error attribution: that is, for each instance, was the error due to issues in producing the visualization, such as code execution errors or incorrect visualization, or issues in visual recognition from the generated visuals?

As the ASCII MNIST dataset is originally derived from actual images, it provides an avenue to 'ground truth visualizations.' By measuring the performance of the MLLM on the real images as an 'upper bound,' we can obtain insight into how much of the error can be attributed to each of these causes. If the model were able to produce the 'ideal' visualization for every data point, how much would its performance be limited by its visual perception capabilities? We find that the MLLM obtains an accuracy of 80.8% on actual MNIST images. This suggests a substantial proportion of the remaining error can be attributed to perception.

	3D	Basic	Bubble	Doh	Dot Matrix
Direct	0.0	0.0	**100.0**	50.0	0.0
CoT	0.0	0.0	**100.0**	62.5	0.0
WoT (ours)	**92.1**	**78.0**	42.1	**89.6**	**11.8**

Table 2: **ASCII word accuracy breakdown**. Using text alone fails *every* instance of 'visual' ASCII (see Fig. 3).

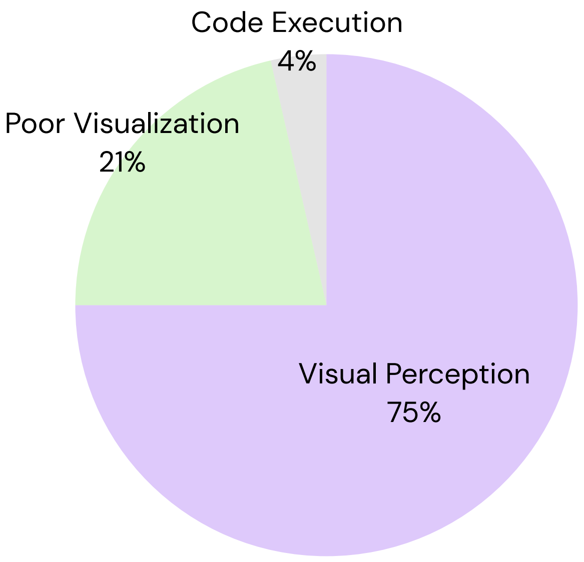
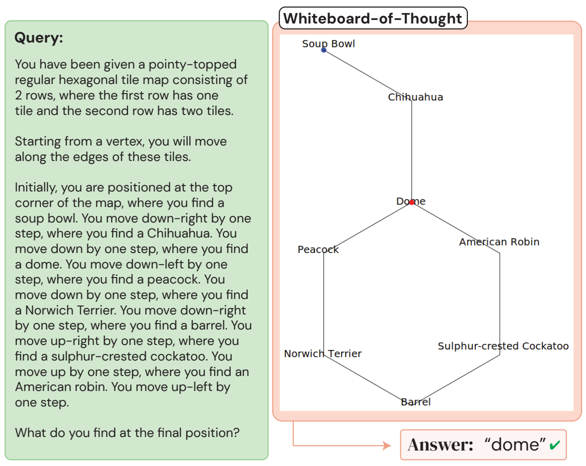


Figure 4: A qualitative breakdown of the sources of error for WoT evaluated on the ASCII MNIST task.


Query:

You have been given a pointy-topped regular hexagonal tile map consisting of 2 rows, where the first row has one tile and the second row has two tiles.

Starting from a vertex, you will move along the edges of these tiles.

Initially, you are positioned at the top corner of the map, where you find a soup bowl. You move down-right by one step, where you find a Chihuahua. You move down by one step, where you find a dome. You move down-left by one step, where you find a peacock. You move down by one step, where you find a Norwich Terrier. You move down-right by one step, where you find a barrel. You move up-right by one step, where you find a sulphur-crested cockatoo. You move up by one step, where you find an American robin. You move up-left by one step.

What do you find at the final position?

Figure 5: Example WoT visual for spatial navigation.

To understand the remaining sources of error for WoT further, we conducted an in-depth error analysis of the results on the ASCII MNIST task. For each error, we qualitatively categorized each error by its apparent cause, finding three broad categories. First, if no image was produced at all, e.g. due to the visualization script raising an error, we labeled the cause 'code execution.' Otherwise, the authors determined if they would find it easy to produce the correct answer from the generated image, judging it a visual perception error if so and poor visualization if not. The results of this analysis can be found in Fig. 4. We find that indeed, the errors largely stem from visual perception.

4.2 Spatial Navigation

Next, we consider the task of understanding the spatial implications of natural language navigation instructions. Given a sequence of spatial instructions like in Fig. 5, humans typically solve these tasks with visual thinking, such as creating a mental image or drawing a physical map (Garvert et al., 2017; Tversky, 2011; Bobek and Tversky, 2016). We aim to understand whether MLLMs can solve these tasks in text alone, possibly suggesting some level of implicit visual thinking for spatial navigation, or if providing a whiteboard to draw an explicit map can provide additional value.

A simple navigation task appeared in BIG-Bench (Srivastava et al., 2022), but only considers movement forwards and backwards in a straight line.The recent work of Yamada et al. (2024) presents a more complex evaluation suite for probing the spatial understanding of LLMs, including navigation across a variety of spatial structures. In particular, we note a distinction between the 2D grid navigation setting – 'Square' and 'Rhombus', here defined as the square rotated 45 degrees – and the non-grid geometries ('Circle', 'Hexagon', and 'Triangle'). We therefore elect to consider this evaluation suite. We present an example, along with the WoT visualization result, in Fig. 5. Details can be found in the Appendix.

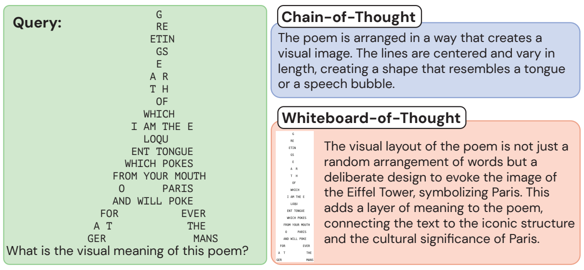
We present the results for navigation on different spatial structures from Yamada et al. (2024) in Table 3. Consistent with Yamada et al. (2024), we observe that LLMs using text excel in the 2D grid setting but not other geometries, which Yamada et al. (2024) hypothesize may be due to the grid setting 1) being more easily represented as coordinates in text than any other setting, especially as an upright 'Square,' and 2) potentially having more

	Non-Grid Structures			2D Grids		Overall
	Circle	**Hexagon**	**Triangle**	**Square**	**Rhombus**	**Avg**
Direct	14	3	16	68	**63**	33
CoT	25	8	26	**98**	51	42
WoT (ours)	**41**	**61**	**55**	50	52	**52**

Table 3: **Spatial navigation results.** While reasoning in text may be suitable for 2D grid structures, non-grid geometries see large improvements from drawings.

data presented in that form on the Internet, e.g., related to tabular data, city grids, and 2D maze coding problems. We note that while humans may *write* about square grids most often in text, grid cells – which humans use to navigate physical spaces and even map conceptual spaces (Constantinescu et al., 2016) – represent space as a hexagonal grid (Hafting et al., 2005), raising interesting questions about how spatial comprehension differs between LLMs and humans. WoT shows more consistent performance across different geometries, obviating the reliance on 2D-grid specific textual knowledge and highlighting the general applicability of the approach. This results in reduced performance on square grids, but higher performance across all other shapes, with the difficult hexagon geometry in particular seeing a jump from 8% with CoT to 61%.

5 In-the-Wild Examples


Query:

```
            G
            RE
           ETIN
            GS
            E
           A R
           T H
            OF
          WHICH
        I AM THE E
          LOQU
        ENT TONGUE
       WHICH POKES
      FROM YOUR MOUTH
       O       PARIS
      AND WILL POKE
    FOR           EVER
   A T              THE
  GER                 MANS
```

What is the visual meaning of this poem?

Chain-of-Thought

The poem is arranged in a way that creates a visual image. The lines are centered and vary in length, creating a shape that resembles a tongue or a speech bubble.

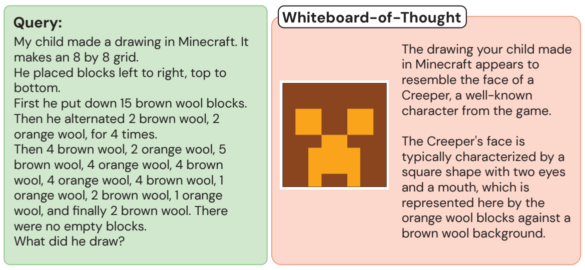
Whiteboard-of-Thought

The visual layout of the poem is not just a random arrangement of words but a deliberate design to evoke the image of the Eiffel Tower, symbolizing Paris. This adds a layer of meaning to the poem, connecting the text to the iconic structure and the cultural significance of Paris.

Figure 6: Calligram understanding: WoT correctly identifies the meaning behind the poem, while CoT picks up on the word 'TONGUE' to hallucinate a response.

In this section we showcase some instances of WoT in the wild.

Calligrams Calligrams are poems where the visual organization of the words adds to the meaning of the poem. Fully understanding a calligram requires consideration of the text not only from a linguistic perspective, but also a visual one. The famous poem by Guillaume Apollinaire – English translation: Claudia Habergham (Feshchenko, 2019) – in Fig. 6 is meant to evoke the Eiffel Tower. Despite being given the same input information, CoT hallucinates the shape of tongue or speech bubble, likely due to the word 'tongue' appearing in the poem. WoT, on the other hand, is able to correctly analyze the poem by first creating a visual. (Interestingly, we find GPT-4o has the original French poem memorized, and claims the words form the Eiffel Tower regardless of the actual shape given; perhaps unsurprising, as this may be one of the most famous calligrams ever made.)


Query:

My child made a drawing in Minecraft. It makes an 8 by 8 grid.
He placed blocks left to right, top to bottom.
First he put down 15 brown wool blocks.
Then he alternated 2 brown wool, 2 orange wool, for 4 times.
Then 4 brown wool, 2 orange wool, 5 brown wool, 4 orange wool, 4 brown wool, 4 orange wool, 4 brown wool, 1 orange wool, 2 brown wool, 1 orange wool, and finally 2 brown wool. There were no empty blocks.
What did he draw?

Whiteboard-of-Thought

The drawing your child made in Minecraft appears to resemble the face of a Creeper, a well-known character from the game.

The Creeper's face is typically characterized by a square shape with two eyes and a mouth, which is represented here by the orange wool blocks against a brown wool background.

Figure 7: WoT could be used in settings like video games, where actions are expressed in text (e.g., code or logs) but have visual results.

Video game art Video games and other digital spaces allow for players to interact with virtual worlds, rendered visually, through predefined actions. This has long led to players creating art through any means the game offers, often with impressive results. For games aimed at children in particular, it is an important task to detect if the visuals players create are content-appropriate. Given only text logs or description of a player's actions, it is difficult for models to process the visual implications of said actions, even though the actions fully define the resulting visuals. WoT provides a solution to this, by actually producing what the result would look like. We provide a simple example towards this goal in Fig. 7.

6 Related Work

6.1 Intermediate reasoning for language models

The success of chain-of-thought (Kojima et al., 2023; Wei et al., 2022) in arithmetic and symbolic reasoning tasks led to substantial interest in the area from the NLP community and beyond. (Yao et al., 2023) generalizes CoT to perform search over trees of candidate rationales. The concurrent 'Visualization-of-Thought' (Wu et al., 2024) prompts models to produce a text pseudo-visualization, e.g., ASCII, and present results suggesting this may improve spatial reasoning as measured in the 2D grid navigation setting of Yamada et al. (2024); compared to our approach, this restricts the category of what can be visualized, and cannot use what the model may have learned in other modalities. Zhang et al. (2023) and Lu et al. (2022) use chain-of-thought style rationales with image and text inputs.

All of these works, ultimately, express their intermediate results as text in some form; our work instead considers the potential of using images to express intermediate reasoning for MLLMs.

6.2 Tool usage and code augmentation

Scratchpads Nye et al. (2021) form a philosophical parallel to our whiteboards, aiming to augment a language model allowing for additional computation (in their case, a text buffer trained on Python execution traces). PAL and PoT (Gao et al., 2023b; Chen et al., 2023) achieve impressive results on arithmetic word problems by using the Python interpreter. Toolformer (Schick et al., 2023) demonstrates a training method to induce language models to invoke API calls to tools. VisProg (Gupta and Kembhavi, 2022) and ViperGPT (Surís et al., 2023) provide an API of visual modules to an LLM to perform visual reasoning, using a domain specific language and the Python interpreter respectively. HuggingGPT (Shen et al., 2023) shows LLMs can instead use existing models from the HuggingFace Transformers library (Wolf et al., 2020).

6.3 Visual and spatial reasoning in LLMs and MLLMs

We are by no means the first to observe the limited success of LLMs and MLLMs on tasks requiring visual and spatial reasoning. The capacity of these models for grounding – tying knowledge from the textual domain to that of other modalities, such as vision – is controversial. Patel and Pavlick (2022) suggests LLMs that have only seen text can perform few-shot mapping of new concepts, such as spatial directions and color, onto grounded world representations. Prompted by this, Yamada et al. (2024) perform an in-depth evaluation of the spatial understanding of state-of-the-art LLMs, creating an extensive benchmark assessing understanding of navigation instructions on different spatial structures; they find that while GPT-4 in particular performs successful spatial navigation on 2D grids but that in general, this does not hold for other spatial structures. Jiang et al. (2024) shows that LLMs are not capable of recognizing ASCII art which requires visual rather than simply textual understanding, and make use of this shortcoming to develop a jailbreak method that bypasses all current methods for defense and elicits unsafe behavior from state-of-the-art LLMs.

Zhang et al. (2024); Gao et al. (2023a); Kazemi et al. (2023) show that current MLLMs struggle to understand mathematical diagrams, such as geometric figures and graphs. Concurrent work by Wang et al. (2024) finds the same, suggesting a domain-specific language for vector graphics as an alternative input to images for LLMs to perform low-level visual reasoning. Meanwhile, Huang et al. (2023) and Han et al. (2023) show MLLMs perform poorly on chart understanding. As discussed in the Limitations section, these weaknesses pose barriers to the application of WoT in these domains with current models. We are confident that as the visual capabilities of MLLMs continue to improve in these domains, WoT's performance will similarly grow.

7 Conclusions

We propose whiteboard-of-thought, a simple, zero-shot method to unlock visual reasoning across modalities in multimodal large language models. We accomplish this by generating code that can create a visual, then returning the visual back to the model for further reasoning. This work demonstrates whiteboard-of-thought's capabilities across multiple tasks requiring visual and spatial reasoning that have thus far proved challenging for current state-of-the-art models with text reasoning. As the abilities of these models to generate code, understand visual inputs, and perform general reasoning continue to improve, we expect the results of whiteboard-of-thought to similarly grow.

8 Limitations

One challenge is that WoT requires accurate vision systems. As detailed in the discussion of Fig. 4, a large proportion of the current errors stem from visual perception. Computer vision has advanced substantially in recent years, but there are still limitations.

For example, a natural domain in which humans apply visual thinking is geometry; yet, even state-of-the-art MLLMs are not yet capable of understanding detailed geometric figures (see Related Work). As computer vision advances, our method will only grow more useful.

Acknowledgments This research is based on work partially supported by the NSF AI Institute for Artificial and Natural Intelligence Award #2229929, the DARPA ECOLE program, the NSF CAREER Award #2046910, and a gift from Amazon. S.M. is supported by the CAIRFI PhD fellowship.

References

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. 2022. Flamingo: a Visual Language Model for Few-Shot Learning. arXiv preprint. ArXiv:2204.14198 [cs].

Eliza Bobek and Barbara Tversky. 2016. Creating visual explanations improves learning. Cognitive Research: Principles and Implications, 1(1):27.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. arXiv:2005.14165 [cs]. ArXiv: 2005.14165.

S. Card, J. Mackinlay, and B. Shneiderman. 1999. Readings in information visualization - using vision to think.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023. Program of Thoughts Prompting: Disentangling Computation from Reasoning for Numerical Reasoning Tasks. arXiv preprint. ArXiv:2211.12588 [cs].

Alexandra O. Constantinescu, Jill X. O'Reilly, and Timothy E. J. Behrens. 2016. Organizing conceptual knowledge in humans with a gridlike code. Science, 352(6292):1464–1468. Publisher: American Association for the Advancement of Science.

Vladimir Feshchenko. 2019. Graphic Translation of Experimental Verse as a Strategy of Poetic Text's Transcreation. Studia Metrica et Poetica, 6:94–115.

Jiahui Gao, Renjie Pi, Jipeng Zhang, Jiacheng Ye, Wanjun Zhong, Yufei Wang, Lanqing Hong, Jianhua Han, Hang Xu, Zhenguo Li, and Lingpeng Kong. 2023a. G-LLaVA: Solving Geometric Problem with Multi-Modal Large Language Model. Publisher: [object Object] Version Number: 1.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023b. PAL: Program-aided Language Models. arXiv preprint. ArXiv:2211.10435 [cs].

Mona M Garvert, Raymond J Dolan, and Timothy Ej Behrens. 2017. A map of abstract relational knowledge in the human hippocampal–entorhinal cortex. eLife, 6:e17086. Publisher: eLife Sciences Publications, Ltd.

Gabriel Goh, Nick Cammarata †, Chelsea Voss †, Shan Carter, Michael Petrov, Ludwig Schubert, Alec Radford, and Chris Olah. 2021. Multimodal Neurons in Artificial Neural Networks. Distill, 6(3):e30. Tex.ids= goh_multimodal_2021.

Tanmay Gupta and Aniruddha Kembhavi. 2022. Visual programming: compositional visual reasoning without training.

Torkel Hafting, Marianne Fyhn, Sturla Molden, May-Britt Moser, and Edvard I. Moser. 2005. Microstructure of a spatial map in the entorhinal cortex. Nature, 436(7052):801–806. Publisher: Nature Publishing Group.

Yucheng Han, Chi Zhang, Xin Chen, Xu Yang, Zhibin Wang, Gang Yu, Bin Fu, and Hanwang Zhang. 2023. ChartLlama: A Multimodal LLM for Chart Understanding and Generation. Publisher: arXiv Version Number: 1.

Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P. Kingma, Ben Poole, Mohammad Norouzi, David J. Fleet, and Tim Salimans. 2022. Imagen Video: High Definition Video Generation with Diffusion Models. arXiv preprint. ArXiv:2210.02303 [cs].

Kung-Hsiang Huang, Mingyang Zhou, Hou Pong Chan, Yi R. Fung, Zhenhailong Wang, Lingyu Zhang, Shih-Fu Chang, and Heng Ji. 2023. Do LVLMs Understand Charts? Analyzing and Correcting Factual Errors in Chart Captioning. Publisher: arXiv Version Number: 1.

Gabriel Ilharco, Mitchell Wortsman, Samir Yitzhak Gadre, Shuran Song, Hannaneh Hajishirzi, Simon Kornblith, Ali Farhadi, and Ludwig Schmidt. 2022. Patching open-vocabulary models by interpolating weights. arXiv preprint. ArXiv:2208.05592 [cs].

Ajay Jain, Amber Xie, and Pieter Abbeel. 2023. VectorFusion: text-to-SVG by abstracting pixel-based diffusion models. 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 1911–1920. Conference Name: 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) ISBN: 9798350301298 Place: Vancouver, BC, Canada Publisher: IEEE.

Fengqing Jiang, Zhangchen Xu, Luyao Niu, Zhen Xiang, Bhaskar Ramasubramanian, Bo Li, and Radha Poovendran. 2024. ArtPrompt: ASCII Art-based Jailbreak Attacks against Aligned LLMs. arXiv preprint. ArXiv:2402.11753 [cs].

Mehran Kazemi, Hamidreza Alvari, Ankit Anand, Jialin Wu, Xi Chen, and Radu Soricut. 2023. GeomVerse: A Systematic Evaluation of Large Models for Geometric Reasoning. Publisher: arXiv Version Number: 1.

Jing Yu Koh, Daniel Fried, and Ruslan Salakhutdinov. 2023a. Generating images with multimodal language models. Neural Information Processing Systems, abs/2305.17216. ArXiv:2305.17216 [cs].

Jing Yu Koh, Ruslan Salakhutdinov, and Daniel Fried. 2023b. Grounding Language Models to Images for Multimodal Generation. arXiv preprint. Tex.ids= koh_grounding_2023a arXiv: 2301.13823 [cs].

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2023. Large Language Models are Zero-Shot Reasoners. arXiv preprint. ArXiv:2205.11916 [cs].

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023. BLIP-2: Bootstrapping Language-Image Pretraining with Frozen Image Encoders and Large Language Models. arXiv preprint. ArXiv:2301.12597 [cs].

Fangyu Liu, Guy Emerson, and Nigel Collier. 2023. Visual Spatial Reasoning. Transactions of the Association for Computational Linguistics, 11:635–651.

Pan Lu, Swaroop Mishra, Tony Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. 2022. Learn to Explain: Multimodal Reasoning via Thought Chains for Science Question Answering. Publisher: arXiv Version Number: 2.

Joanna Materzynska, Antonio Torralba, and David Bau. 2022. Disentangling visual and written concepts in CLIP. In 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 16389–16398, New Orleans, LA, USA. IEEE. Conference Name: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) ISBN: 9781665469463 Place: New Orleans, LA, USA Publisher: IEEE.

Sachit Menon, Ishaan Preetam Chandratreya, and Carl Vondrick. 2022. Task Bias in Vision-Language Models. Publisher: arXiv Version Number: 1.

David A. Noever and S. M. Noever. 2021. Reading isn't believing: adversarial attacks on multi-modal neurons. ArXiv.

Maxwell Nye, Anders Andreassen, Guy Gur-Ari, H. Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, D. Luan, Charles Sutton, and Augustus Odena. 2021. Show Your Work: Scratchpads for Intermediate Computation with Language Models. ArXiv. Tex.ids= nye_show_2021 arXiv: 2112.00114 [cs] number: arXiv:2112.00114.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor

Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, C. J. Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2023. GPT-4 Technical Report.

Roma Patel and Ellie Pavlick. 2022. Mapping Language Models to Grounded Conceptual Spaces.

Juan A Rodriguez, David Vazquez, Issam Laradji, Marco Pedersoli, and Pau Rodriguez. 2023a. FigGen: text to scientific figure generation. Publisher: arXiv Version Number: 3.

Juan A. Rodriguez, David Vazquez, Issam Laradji, Marco Pedersoli, and Pau Rodriguez. 2023b. OCR-VQGAN: taming text-within-image generation. 2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), pages 3678–3687. Tex.ids= rodriguez_ocrvqgan_2023 ISBN: 9781665493468 conferenceName: 2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) place: Waikoloa, HI, USA publisher: IEEE.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Bjorn Ommer. 2022. High-Resolution Image Synthesis with Latent Diffusion Models. In 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 10674–10685, New Orleans, LA, USA. IEEE.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. 2022. Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding. Publisher: arXiv Version Number: 1.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language Models Can Teach Themselves to Use Tools. arXiv preprint. ArXiv:2302.04761 [cs].

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. HuggingGPT: solving AI tasks with ChatGPT and its friends in hugging face. arXiv preprint. Tex.ids= shen_hugginggpt_2023 arXiv: 2303.17580 [cs].

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, Agnieszka Kluska, Aitor Lewkowycz, Akshat Agarwal, Alethea Power, Alex Ray, Alex Warstadt, Alexander W. Kocurek, Ali Safaya, Ali Tazarv, Alice Xiang, Alicia Parrish, Allen Nie, Aman Hussain, Amanda Askell, Amanda Dsouza, Ambrose Slone, A. Rahane, Anantharaman S. Iyer, Anders Andreassen, Andrea Madotto, Andrea Santilli, Andreas Stuhlmuller, Andrew M. Dai, Andrew La, Andrew Kyle Lampinen, Andy Zou, Angela Jiang, Angelica Chen, Anh Vuong, Animesh Gupta, Anna Gottardi, Antonio Norelli, Anu Venkatesh, Arash Gholamidavoodi, Arfa Tabassum, Arul Menezes, Arun Kirubarajan, A. Mullokandov, Ashish Sabharwal, Austin Herrick, Avia Efrat, Aykut Erdem, Ayla Karakacs, B. R. Roberts, B. S. Loe, Barret Zoph, Bartlomiej Bojanowski, Batuhan Ozyurt, Behnam Hedayatnia, Behnam Neyshabur, Benjamin Inden, Benno Stein, Berk Ekmekci, Bill Yuchen Lin, B. Howald, Bryan Orinion, Cameron Diao, Cameron Dour, Catherine Stinson, Cedrick Argueta, C'esar Ferri Ram'irez, Chandan Singh, Charles Rathkopf, Chenlin Meng, Chitta Baral, Chiyu Wu, Chris Callison-Burch, Chris Waites, Christian Voigt, Christopher D. Manning, Christopher Potts, Cindy Ramirez, Clara Rivera, Clemencia Siro, Colin Raffel, Courtney Ashcraft, Cristina Garbacea, Damien Sileo, Daniel H. Garrette, Dan Hendrycks, D. Kilman, Dan Roth, Daniel Freeman, Daniel Khashabi, Daniel Levy, D. Gonz'alez, Danielle R. Perszyk, Danny Hernandez, Danqi Chen, Daphne Ippolito, D. Gilboa, David Dohan, D. Drakard, David Jurgens, Debajyoti Datta, Deep Ganguli, Denis Emelin, Denis Kleyko, Deniz Yuret, Derek Chen, Derek Tam, Dieuwke Hupkes, Diganta Misra, Dilyar Buzan, Dimitri Coelho Mollo, Diyi Yang, Dong-Ho Lee, Dylan Schrader, Ekaterina Shutova, E. D. Cubuk, Elad Segal, Eleanor

Hagerman, Elizabeth Barnes, E. Donoway, Ellie Pavlick, E. Rodolà, Emma Lam, Eric Chu, Eric Tang, Erkut Erdem, Ernie Chang, Ethan A. Chi, Ethan Dyer, E. Jerzak, Ethan Kim, Eunice Engefu Manyasi, Evgenii Zheltonozhskii, Fanyue Xia, F. Siar, Fernando Mart'inez-Plumed, Francesca Happ'e, François Chollet, Frieda Rong, Gaurav Mishra, Genta Indra Winata, Gerard de Melo, Germán Kruszewski, Giambattista Parascandolo, Giorgio Mariani, Gloria Xinyue Wang, Gonzalo Jaimovitch-L'opez, Gregor Betz, Guy Gur-Ari, Hana Galijasevic, Hannah Kim, Hannah Rashkin, Hannaneh Hajishirzi, Harsh Mehta, H. Bogar, Henry Shevlin, Hinrich Schutze, H. Yakura, Hongming Zhang, Hugh Mee Wong, Ian Ng, Isaac Noble, Jaap Jumelet, Jack Geissinger, John Kernion, Jacob Hilton, Jaehoon Lee, J. Fisac, James B. Simon, James Koppel, James Zheng, James Zou, Jan Koco'n, Jana Thompson, Janelle Wingfield, Jared Kaplan, Jarema Radom, Jascha Narain Sohl-Dickstein, Jason Phang, Jason Wei, J. Yosinski, Jekaterina Novikova, Jelle Bosscher, Jennifer Marsh, Jeremy Kim, Jeroen Taal, Jesse Engel, Jesujoba Oluwadara Alabi, Jiacheng Xu, Jiaming Song, Jillian Tang, Jane W. Waweru, John Burden, John Miller, John U. Balis, Jonathan Batchelder, Jonathan Berant, Jorg Frohberg, Jos Rozen, J. Hernández-Orallo, Joseph Boudeman, Joseph Guerr, Joseph Jones, Joshua Tenenbaum, Joshua S. Rule, Joyce Chua, Kamil Kanclerz, Karen Livescu, K. Krauth, Karthik Gopalakrishnan, Katerina Ignatyeva, K. Markert, Kaustubh D. Dhole, Kevin Gimpel, Kevin Omondi, K. Mathewson, Kristen Chiafullo, Ksenia Shkaruta, K. Shridhar, Kyle McDonell, Kyle Richardson, Laria Reynolds, Leo Gao, Li Zhang, Liam Dugan, Lianhui Qin, Lidia Contreras-Ochando, Louis-Philippe Morency, Luca Moschella, Luca Lam, Lucy Noble, Ludwig Schmidt, Luheng He, Luis Oliveros Col'on, Luke Metz, Lutfi Kerem cSenel, Maarten Bosma, Maarten Sap, Maartje ter Hoeve, Maheen Farooqi, Manaal Faruqui, Mantas Mazeika, Marco Baturan, Marco Marelli, Marco Maru, Maria Jose Ram'irez Quintana, Marie Tolkiehn, Mario Giulianelli, Martha Lewis, Martin Potthast, Matthew L. Leavitt, Matthias Hagen, M. Schubert, Medina Baitemirova, Melody Arnaud, M. McElrath, Michael Yee, Michael Cohen, Michael Gu, M. Ivanitskiy, Michael Starritt, M. Strube, Michal Swkedrowski, Michele Bevilacqua, Michihiro Yasunaga, Mihir Kale, Mike Cain, Mimee Xu, Mirac Suzgun, Mitch Walker, Monica Tiwari, Mohit Bansal, Moin Aminnaseri, Mor Geva, Mozhdeh Gheini, T. MukundVarma, Nanyun Peng, Nathan A. Chi, Nayeon Lee, Neta Gur-Ari Krakover, Nicholas Cameron, Nicholas Roberts, Nick Doiron, Nicole Martinez, Nikita Nangia, Niklas Deckers, Niklas Muennighoff, N. Keskar, Niveditha Iyer, Noah Constant, Noah Fiedel, Nuan Wen, Oliver Zhang, Omar Agha, Omar Elbaghdadi, Omer Levy, Owain Evans, Pablo Antonio Moreno Casares, P. Doshi, Pascale Fung, P. Liang, Paul Vicol, Pegah Alipoormolabashi, Peiyuan Liao, Percy Liang, Peter Chang, P. Eckersley, Phu Mon Htut, Pi-Bei Hwang, P. Milkowski, P. Patil, Pouya Pezeshkpour, Priti Oli, Qiaozhu Mei, Qing Lyu, Qinlang Chen, Rabin Banjade, Rachel Etta Rudolph, Raefer Gabriel, Rahel Habacker, Ramon Risco, Raphael Milliere, Rhythm Garg, Richard Barnes, R. Saurous, Riku Arakawa, Robbe Raymaekers, Robert Frank, Rohan Sikand, Roman Novak, Roman Sitelew, Ronan Le Bras, Rosanne Liu, Rowan Jacobs, Rui Zhang, R. Salakhutdinov, Ryan Chi, Ryan Lee, Ryan Stovall, Ryan Teehan, Rylan Yang, Sahib Singh, Saif M. Mohammad, Sajant Anand, Sam Dillavou, Sam Shleifer, Sam Wiseman, Samuel Gruetter, Samuel R. Bowman, S. Schoenholz, Sanghyun Han, Sanjeev Kwatra, Sarah A. Rous, Sarik Ghazarian, Sayan Ghosh, Sean Casey, Sebastian Bischoff, Sebastian Gehrmann, Sebastian Schuster, Sepideh Sadeghi, Shadi S. Hamdan, Sharon Zhou, Shashank Srivastava, Sherry Shi, Shikhar Singh, Shima Asaadi, S. Gu, Shubh Pachchigar, Shubham Toshniwal, Shyam Upadhyay, Shyamolima Debnath, Siamak Shakeri, Simon Thormeyer, S. Melzi, Siva Reddy, S. Makini, Soo-Hwan Lee, Spencer Torene, Sriharsha Hatwar, S. Dehaene, Stefan Divic, Stefano Ermon, Stella Biderman, Stephanie Lin, Stephen Prasad, Steven T. Piantadosi, Stuart M. Shieber, Summer Misherghi, S. Kiritchenko, Swaroop Mishra, Tal Linzen, Tal Schuster, Tao Li, Tao Yu, Tariq Ali, Tatsunori Hashimoto, Te-Lin Wu, T. Desbordes, Theodore Rothschild, Thomas Phan, Tianle Wang, Tiberius Nkinyili, Timo Schick, T. Kornev, T. Tunduny, Tobias Gerstenberg, T. Chang, Trishala Neeraj, Tushar Khot, Tyler Shultz, Uri Shaham, Vedant Misra, Vera Demberg, Victoria Nyamai, Vikas Raunak, V. Ramasesh, Vinay Uday Prabhu, Vishakh Padmakumar, Vivek Srikumar, W. Fedus, W. Saunders, William Zhang, Wout Vossen, Xiang Ren, Xiaoyu Tong, Xinran Zhao, Xinyi Wu, Xudong Shen, Yadollah Yaghoobzadeh, Yair Lakretz, Yangqiu Song, Yasaman Bahri, Yejin Choi, Yichi Yang, Yiding Hao, Yifu Chen, Yonatan Belinkov, Yu Hou, Yu Hou, Yuntao Bai, Zachary Seid, Zhuoye Zhao, Zijian Wang, Zijie J. Wang, Zirui Wang, and Ziyi Wu. 2022. Beyond the imitation game: quantifying and extrapolating the capabilities of language models. ArXiv.

Dídac Surís, Sachit Menon, and Carl Vondrick. 2023. ViperGPT: visual inference via python execution for reasoning. In 2023 IEEE/CVF International Conference on Computer Vision (ICCV), pages 11854–11864, Paris, France. IEEE. Conference Name: 2023 IEEE/CVF International Conference on Computer Vision (ICCV) ISBN: 9798350307184 Place: Paris, France Publisher: IEEE.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. 2022. Challenging BIG-Bench Tasks and Whether Chain-of-Thought Can Solve Them. arXiv preprint. ArXiv:2210.09261 [cs].

Barbara Tversky. 2011. Visualizing thought. Topics in Cognitive Science, 3(3):499–535.

Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need.

Zhenhailong Wang, Joy Hsu, Xingyao Wang, Kuan-Hao Huang, Manling Li, Jiajun Wu, and Heng Ji. 2024. Text-Based Reasoning About Vector Graphics. arXiv preprint. ArXiv:2404.06479 [cs].

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, E. Chi, F. Xia, Quoc Le, and Denny Zhou. 2022. Chain of Thought Prompting Elicits Reasoning in Large Language Models. ArXiv.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. HuggingFace's transformers: state-of-the-art natural language processing. arXiv preprint. ArXiv:1910.03771 [cs].

Wenshan Wu, Shaoguang Mao, Yadong Zhang, Yan Xia, Li Dong, Lei Cui, and Furu Wei. 2024. Visualization-of-Thought Elicits Spatial Reasoning in Large Language Models. arXiv preprint. ArXiv:2404.03622 [cs].

Ximing Xing, Chuang Wang, Haitao Zhou, Jing Zhang, Qian Yu, and Dong Xu. 2023. DiffSketcher: text guided vector sketch synthesis through latent diffusion models. Tex.ids= xing_diffsketcher_2023 publisher: arXiv versionNumber: 4.

Yutaro Yamada, Yihan Bao, Andrew K. Lampinen, Jungo Kasai, and Ilker Yildirim. 2024. Evaluating Spatial Understanding of Large Language Models. arXiv preprint. Tex.ids= yamada_evaluating_2023, yamada_evaluating_2024 arXiv: 2310.14540 [cs] publisher: [object Object] versionNumber: 3.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. Publisher: arXiv Version Number: 2.

Lili Yu, Bowen Shi, Ramakanth Pasunuru, Benjamin Muller, Olga Golovneva, Tianlu Wang, Arun Babu, Binh Tang, Brian Karrer, Shelly Sheynin, Candace Ross, Adam Polyak, Russell Howes, Vasu Sharma, Puxin Xu, Hovhannes Tamoyan, Oron Ashual, Uriel Singer, Shang-Wen Li, Susan Zhang, Richard James, Gargi Ghosh, Yaniv Taigman, Maryam Fazel-Zarandi, Asli Celikyilmaz, Luke Zettlemoyer, and Armen Aghajanyan. 2023. Scaling autoregressive multi-modal models: pretraining and instruction tuning. arXiv.org, abs/2309.2591. ArXiv:2309.02591 [cs].

Renrui Zhang, Dongzhi Jiang, Yichi Zhang, Haokun Lin, Ziyu Guo, Pengshuo Qiu, Aojun Zhou, Pan Lu, Kai-Wei Chang, Peng Gao, and Hongsheng Li. 2024. MathVerse: Does Your Multi-modal LLM Truly See the Diagrams in Visual Math Problems? arXiv preprint. ArXiv:2403.14624 [cs].

Zhuosheng Zhang, Aston Zhang, Mu Li, Hai Zhao, George Karypis, and Alex Smola. 2023. Multimodal Chain-of-Thought Reasoning in Language Models. arXiv preprint. ArXiv:2302.00923 [cs].

A Discussion of Text-to-Image Models

Another interesting option to consider might be text-to-image models (Rombach et al., 2022; Saharia et al., 2022; Ho et al., 2022), which are already integrated into some LLMs (**?**), or MLLMs that natively generate sequences of multiple modalities (Yu et al., 2023; Koh et al., 2023b,a). Text-to-image models have seen great successes, creating diverse and high-quality images from given text prompts(Rombach et al., 2022; Saharia et al., 2022; Ho et al., 2022). Despite this, it remains difficult to use them to create precise figures and drawings that would be suitable for visual reasoning, though there are exciting preliminary works in this direction (Rodriguez et al., 2023a,b; Jain et al., 2023; Xing et al., 2023). As more effective methods for precisely controlling text-to-image models emerge, these could be easily integrated into WoT.

B Code and Details for Qualitative Examples

All qualitative examples created with temperature 0, max tokens of 2048, top p of 1, frequency penalty of 0, presence penalty of 0, and with history (when the model considers the image, it also sees previous exchanges including the generated code, discussed more in Experimental Details). All generated code for qualitative figures will be released with the code for experiments. We present an example for the ASCII creation setting (from Fig. 3) here.

Python Code for ASCII Art

Listing 1: Python Code for ASCII Art

```python
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors
import numpy as np

# Define the ASCII art
ascii_art = [
    "".........................."",
    "".........................."",
    ""..******..*****...******.."",
    "".**////..**///**.//////**."",
    ""//*****./*******..*******."",
    ""./////**/**////..**////**."",
    "".******.//******//********"",
    ""//////...//////..////////.""
]

# Define the color mapping
color_map = {
    '.': (1, 1, 1),  # white
    '*': (0, 0, 0),  # black
    '/': (0.5, 0.5, 0.5)  # gray
}

# Determine the size of the grid
rows = len(ascii_art)
cols = max(len(row) for row in ascii_art)

# Create an array to hold the colors
image = np.ones((rows, cols, 3))

# Fill the array with colors based on the ASCII art
for i, row in enumerate(ascii_art):
    for j, char in enumerate(row):
        if char in color_map:
            image[i, j] = color_map[char]

# Create the figure and axis
fig, ax = plt.subplots(figsize=(6, 6))

# Display the image
ax.imshow(image, interpolation='nearest')

# Remove the axes
ax.axis('off')

# Show the plot
plt.show()
```

C Experimental Details

Here we provide details relevant to each experiment.

For all experiments, we use GPT-4o (gpt-4o-2024-05-13) as the underlying MLLM as it has each of the necessary capabilities enabling WoT – in particular, the ability to produce code outputs and to accept image inputs. For the initial query, we use a temperature of 0, max tokens of 2048, top p of 1, frequency penalty of 0.05, and presence penalty of 0. We use these settings for all direct queries and chain-of-thought experiments as well. For the subsequent image query, we use the same settings but with max tokens set to 256 and no frequency penalty.

As all experiments use a temperature of 0 and deterministic sampling, we do not perform multiple runs.

We do not use any GPU resources of our own for these experiments, instead using the OpenAI API for LLM/MLLM queries.

The core prompt enabling WoT is "You write code to create visualizations using the {Matplotlib/Turtle} library in Python, which the user will run and provide as images.", provided as a system prompt. We describe some additional details here that we used to condition current MLLMs into the desired behavior. For all experiments, we append "Do NOT produce a final answer to the query until considering the visualization." to the system prompt, as we found GPT-4o would often directly produce an answer without waiting for an image to be provided otherwise, often by hallucinating that it already had seen the image. Curiously, we found that the OpenAI content filter initially flagged a large proportion of examples, especially those that appeared pixelated and stretched to the borders, as inappropriate content. We added a white border and resized all images before sending them as queries, which somehow avoids the filter. We use the prompt from (Yamada et al., 2024) for direct answer prompting: "You are given a task to solve. Make sure to output an answer after "Answer:" without any explanation."

The code from the produced response for visualization typically is contained as a code block in a broader response. We extract this with the regular expression `re.findall(r"“‘python(.*?)“‘", text, re.DOTALL)`; we note this may present a source of error in failing to detect code that is not enclosed in such a block.

We found that providing both the previous history and the image in the image understanding query as opposed to passing on just the image obtained results within 1% for our evaluation setting, so we elect to use the latter in general as it uses substantially fewer tokens. This choice could have interesting implications for e.g., faithfulness, where the result is guaranteed to be faithful to the image with this approach (vs not in the other case). We leave this to future work.

The ASCII understanding tasks from BIG-Bench (Srivastava et al., 2022) were not included in BIG-Bench Hard (Suzgun et al., 2022) due to being "not even worth attempting with chain-of-thought." As such, we follow a similar procedure to construct evaluation splits here. In particular, we take a subset of 250 randomly chosen examples from the evaluation splits of each of the three tasks, and manually verify that they can be answered. (Running 69984 queries would also be prohibitively costly.) We remove the choices for all multiple choice questions (e.g., MNIST being given options 0-9.) For the kanji recognition task, we only include problems from the pronunciation subtask rather than the translation subtask due to lack of data. We invoke Matplotlib as the visualization tool for each of these experiments. For the image creation query, we provide the ASCII content and append each of the following: “Write Python code with Matplotlib to render the ASCII art as an image." This avoids the model forgetting its task and directly answering. “Let the main figure be called fig with size 6,6." This makes it easy for us to save the generated figures. “Ensure each character in the input is considered. Remember colors are matplotlib.colors, and colors must be RGB to be displayed. Remember not all rows are necessarily the same length." These prevented common execution errors that did not seem to reflect the true visualization capabilities of the model or were due to formatting issues in the original ASCII. For the image prompt, While we hope that as models continue to improve, these additions will be less necessary, we believe they do not detract from the results. To evaluate answers, we convert MNIST digits to integers and compare equality with ground truth, marking an instance incorrect if the type conversion fails; for word and kanji recognition, we convert the output string to lowercase and perform exact string matching.

We use data created by (Yamada et al., 2024) to evaluate spatial understanding of LLMs for the spatial navigation experiments. In particular, we use the set described in Section 3.1 of that work, “Do different spatial structure features affect model performance?" In particular, this includes the ring/circle, hexagon, square, rhombus, and triangle, with 100 examples for each for 500 total. We use Turtle as the visualization tool for this task. For the image creation query, we provide the navigation instructions and append each of the following: “Use Python code with Turtle to visualize each step." As otherwise the model would still forget its instructions and produce an answer directly. “All directions are in reference to up at setheading(90)." To provide a frame of reference for the language given. “Name the turtle t; let the step size be 200; mark the final position with a red dot (do not write the final position as text). All other steps may be written as text." This allows us to easily save the resulting visualization. We found one limitation of the current method to be that the created visual was often correct, but had text overlap that prevented legibility (e.g., writing ‘Final Position’ on top of the item found at that spot); this could be avoided by returning the image back to the model and asking it to modify the code, but doing this for every query is costly, resulting in these modifications. It may be interesting to explore models revising their own visuals as future work.

D Potential Risks

While this work may help against some forms of visual adversarial attacks (Jiang et al., 2024), opening up the possibility for text queries to involve image processing could have unforeseen consequences for new forms of attacks. In addition, improving the reasoning abilities of LLMs and MLLMs in general may involve some risks, such as to certain forms of employment; however, we do not see particular ways our technique contributes to that past the general setting.