From Language to Pixels: Task Recognition and Task Learning in LLMs

Janek Falkenstein, Carolin Schuster, Alex Berger and **Georg Groh**
Technical University of Munich, Germany
{j.falkenstein, carolin.schuster, a.berger}@tum.de
grohg@in.tum.de

Abstract

Large language models (LLMs) can perform unseen tasks by learning from a few in-context examples. How in-context learning works is still uncertain. We investigate the mechanisms of in-context learning on a challenging non-language task. The task requires the LLM to generate pixel matrices representing images of basic shapes. We introduce a framework to analyze if this task is solved by recognizing similar formats from the training data (task recognition) or by understanding the instructions and learning the skill de novo during inference (task learning). Our experiments demonstrate that LLMs generate meaningful pixel matrices with task recognition and fail to learn such tasks when encountering unfamiliar formats. Our findings offer insights into LLMs' learning mechanisms to guide future research on their seemingly human-like behavior.

1 Introduction

The rapid development of LLMs increases the chance of misinterpreting their capabilities. It is crucial to explore the potential of LLMs and research their underlying mechanisms. This helps in interpreting LLMs outputs, guides future research to improve language models, and informs their use and regulation.

Since Brown et al. (2020) demonstrated the abilities of GPT-3, there has been a growing interest in researching LLMs' potential to generalize to other tasks. Many studies demonstrated LLMs' abilities in solving reasoning tasks (Li et al., 2021; Srivastava et al., 2023; Huang and Chang, 2023). Contrary, it has been shown that LLMs still struggle to reason about seemingly simple tasks where humans perform superior (Valmeekam et al., 2022; Binz and Schulz, 2023; Huang and Chang, 2023). These findings demonstrate ambiguity in LLMs' general abilities and the need to further probe challenging tasks.

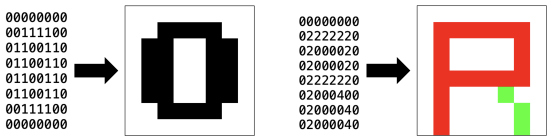

Figure 1: Two 8×8 pixel matrices representing simple images generated by an LLM using in-context learning.

In addition to the ambiguity about general capabilities, it is also unclear how and why LLMs benefit from in-context learning, i.e., how they can learn from instructions and demonstrations in the prompt. Brown et al. (2020) suggest that in-context learning could stem from genuinely learning new tasks, identifying tasks learned during training, or a mix of both approaches. However, the mechanisms of in-context learning remain uncertain.

On one hand, the inner workings (e.g., attention patterns) are too vast and complex to analyze directly. On the other hand, finding a higher-level abstraction to describe the model's internal processes is challenging. It is crucial to differentiate LLMs' achievements from human-like comprehension and reasoning as their training misses critical contexts, such as communicative intent (Bender and Koller, 2020; Asher et al., 2023). Especially terms like "learning" and "understanding" should be used cautiously (Bender et al., 2021; Shanahan, 2022). However, higher-level frameworks can help to explain LLM behaviors and assess LLMs' potential (Shanahan et al., 2023).

We adopt the abstraction level from Brown et al. (2020) and the terminology proposed by Pan et al. (2023) and distinguish between **task learning** (TL) and **task recognition** (TR). TR assesses how well LLMs can identify tasks through demonstrations and apply their pre-trained priors. TL describes the ability to learn new input-label mappings not encountered during training. Building on this, we introduce a framework that involves breaking tasks into subtasks and analyzing each for TL and TR to thoroughly investigate in-context learning.

We instruct a model to generate pixel art as 8×8 pixel matrices (see Figure 1). The experiments solely use the model's inherent tokens without additional fine-tuning or incorporating image models. While investigating this task, we demonstrate unique capabilities, pinpoint where the pixel matrix task falls along the spectrum between TL and TR, and establish and validate a straightforward framework of breaking complex tasks into subtasks to clarify how LLMs achieve the observed capabilities.

Our main contributions are summarized as follows:

- We show that LLMs can create pixel matrices of digits, letters, and simple everyday objects.
- We find that LLMs rely on task recognition to generate meaningful pixel matrices, which indicates strong task recognition but limited task learning abilities for uncommon non-language tasks.
- We propose a framework to break tasks into subtasks to better explain the capabilities of LLMs and separate the effects of TL and TR.

2 Related Work

Text-to-image models The success of LLMs has influenced the research on multi-modal models for vision and language. Many text-to-image models use text encoders from LLM transformers and combine them with an image encoder (e.g., Radford et al., 2021; Alayrac et al., 2022; Saharia et al., 2022). Others adhere closer to the LLM architecture. Koh et al. (2023) have used visual encoders to ground an LLM in the visual domain, enabling image captioning and text-to-image tasks. Similarly, Pourreza et al. (2023) generates images represented as brush strokes. They add an image feature extractor and cross-attention blocks to provide visual feedback during stroke generation. These studies demonstrate that LLMs possess implicit knowledge about images and the visual domain. Our task is distinct in that it generated images without extra image encoders or training.

Pure LLMs on image generation Probing the ability of LLMs to create images without fine-tuning or adding layers has been done by benchmarks that assemble tasks to assess LLM performances. BIG-bench (Srivastava et al., 2023) includes some tasks that involve ASCII art and Bang et al. (2023) let ChatGPT draw country flags with Scalable Vector Graphics (SVG) code. Chalamalasetti et al. (2023) benchmark LLMs on describing 5×5 grids filled with two different symbols and following such descriptions. In addition, some blog posts explore ChatGPT's ability to draw images with SVG (Pu, 2022; Shahir, 2023 Shiryaev, 2022). To our knowledge, no comprehensive study has explored LLMs' capacity for generating visuals, and we are the first to assess the pixel matrix image format.

In-context learning How in-context learning works is still disputed. Some recent studies compared in-context learning to implicit fine-tuning or Bayesian inference (Dai et al., 2023; Von Oswald et al., 2023; Xie et al., 2022). Whether this capability arises from comprehending the instruction or recognizing the task from the training data is still an open research question. Pan et al. (2023) introduced the terminology employed in our paper and distinguished cases where TL and TR were applied. Other studies experimented with modifying the prompt or comparing tasks, aiming to ascertain whether LLMs employ TR or TL (Reynolds and McDonell, 2021; Min et al., 2022). We diverge in our approach by closely examining one particular task concerning TL or TR to contribute insights to the broader understanding of in-context learning and to provide a framework for future task evaluations.

Decomposing Tasks Letting the model explain each reasoning step (Chain-of-Thought) has been shown to improve results (Lampinen et al., 2022; Kojima et al., 2022; Wei et al., 2022). Other studies have shown that explicitly decomposing complex tasks and solving the subtasks enhances performance (Zhou et al., 2022; Khot et al., 2022; Prasad et al., 2023; Radhakrishnan et al., 2023). Our experiments do not focus on prompting or explicitly breaking down tasks to improve performance. However, these results support our proposed framework: when analyzing LLMs, it is useful to consider subtasks, as models likely implicitly decompose tasks and solve subtasks through either TL or TR.

3 Method

Our experimental setup aims to assess LLMs' general capabilities on the challenging pixel-matrix task and determine whether it is addressed through

TR or TL. Simultaneously, we demonstrate our framework's utility in explaining LLMs' capabilities and differentiating between TL and TR.

We choose the pixel matrix task because it is particularly challenging. It requires the model to translate instructions and visual knowledge into a new format. The task's complexity helps to clearly distinguish between TR and TL and assess our framework. Simpler tasks can be solved in many different ways, and the likelihood that the model recognizes or learns arbitrary parts increases, complicating the decision between TR and TL.

By thoroughly examining a single task, we provide evidence contributing to a broader understanding of in-context learning and the differentiation between TR and TL.

3.1 Prompt Structures and Dataset

The prompts for the pixel matrix experiments consist of a task description and four demonstrations. The description introduces the concept of pixels and pixel matrices. The demonstrations show example pixel matrices with labels. These 8×8 pixel matrices are depicted with 8 rows, each containing 8 pixel symbols (e.g., 0 and 1, or G and K). After the four examples, another label specifying the object to be sketched is added. See Figure 2 for the prompt structure. The model is expected to generate a corresponding pixel matrix.

We evaluate the performance of creating pixel matrices across four categories: *digits*, *letters*, *punctuation symbols*, and *real-world objects*. The *real-world objects* are simple enough to be displayable on an 8×8 pixel canvas (e.g., chess board, padlock, or sun). Appendix C includes a complete enumeration of the objects. In total, we have 93 different objects. We generate ten instances for each object. We test each of the four object categories separately.

3.2 Experimental Setup

For the experiments, we modified the pixel symbols, the few-shot examples, and the task description of the prompt. We used the *gpt-3.5-turbo-0613* model accessed through the OpenAI API. Other open-source models, such as Bloom (Scao et al., 2022), GPT-Neox20B (Black et al., 2022), and Starcoder (Li et al., 2023), cannot generate meaningful pixel matrices (see Appendix E.3). Therefore, we focus our analysis on GPT-3.5.

```
Images displayed on a computer screen are a
collection of color dots, called pixels. [...]
We can represent different objects by creating a
pixel matrix which consists of 0s and 1s. [...]

Here is an example of an 8 by 8 pixel matrix
showing  three:
00000000
00111110
00000110
00111100
00001110
00000110
00111100
00000000
###

[... three more examples ...]

This is an example of a grid of pixels that
form an image of [object]:
```

Figure 2: Prompt structure used for generating pixel matrices in the experiments. The descriptions and examples were adjusted according to each experiment.

General capabilities We conducted a *Baseline* experiment that shows the model's general capabilities and acts as a baseline for the other experiments. The pixel symbols are 0 and 1 and represent white and black. Furthermore, we conducted experiments with color representations to evaluate advanced capabilities and determine the limits of the pixel matrix task. The *Color Digits* experiment added four additional digits for red, blue, yellow, and green, while the *Color RGB* experiment used RGB code values as pixel symbols.

Task recognition vs. task learning These experiments aim to understand whether GPT-3.5 approaches the pixel matrix task through TR or TL. We follow three hypotheses as indicators for TL:

- Performance should be independent of the frequency in the training data and equal across tasks of the same difficulty.
- Performance should deteriorate when providing misleading instructions.
- Performance should improve when making the task easier.

Following these hypotheses, we designed three experiments.

The first experiment, (*GK Pixels*), substitutes the pixel values 0 and 1 with two letters chosen uniformly at random from the Latin alphabet. Black-and-white pixel matrices with 0s and 1s are prevalent in the training data, while matrices with values

G and K are not. Simple objects like digits and letters represented as matrices are likely well represented and learned during training, enabling TR. In contrast, the GK Pixel matrix format must be recognized or learned from the context.

The second experiment, *Wrong Labels*, uses mislabeled examples in the prompt. We hypothesize that a deteriorating performance would indicate TL because misleading examples make the instructions of the task unclear. However, a constant performance in this experiment means that in-context learning helps to identify a given task rather than learning it (Min et al., 2022; Pan et al., 2023). We conducted the *Wrong Labels* experiment by mislabeling each of the four prompt examples. We also experimented without any examples (zero-shot) to test if the model could learn from the instruction of the prompt alone.

GPT-3.5's tokenizer[1] combines multiple pixel symbols of our baseline matrix format into one token (see Figure 7). Consequently, a single pixel matrix can be represented by various tokens, expanding the token vocabulary from two to more than ten. Hence, learning the task becomes more challenging because it is necessary to understand the meaning of each potential token. For the third experiment (*One Token*), we inserted spaces between pixel symbols to ensure one token corresponds to a single pixel symbol and make the task easier to learn. Such adjustment enhanced results for straightforward pattern completion tasks (Mirchandani et al., 2023). In the case of TL, we anticipate improved results for our task.

In addition to these three experiments, we use the *baseline* results to compare objects similarly challenging to sketch but unevenly represented in the training data or of different difficulty but evenly represented in the training data. If the results align with the assumed difficulty, it suggests TL without relying on pixel matrices in the training data. For TR, only the training data is relevant. Consequently, when two similarly challenging objects have uneven representation, the one more prevalent during training should yield more accurate results.

Breaking tasks into subtasks With this set of experiments, we demonstrate how manually decomposing tasks can help distinguish between TL and TR. If the model fails to complete the subtasks, we infer that the main task is solved by TR. When the model succeeds in difficult tasks not seen during training, we do not immediately attribute this to TL but instead examine the subtasks. Labeling the process as TL is inappropriate if the model simply combines subtasks it solved using TR.

To test one subtask explicitly, we generated textual descriptions of objects' shapes and visuals using a simple prompt. For the GK task, we tested the subtasks of translating 01 pixel matrices to GK pixel matrices. For the specific prompts, see Appendix A. If LLMs can generate GK pixel matrices correctly, one could assume they first recognize the 01 pixel matrices from training data and then use the prompt to translate 01 pixel matrices to GK pixel matrices, solving two subtasks and combining the results.

We also tested our pixel matrix task using a different image format by having the model generate SVG code instead. This format allows for a more straightforward combination of different shapes because overlapping shapes do not affect each other. Examining the generation of *real-world objects* can reveal whether the model predominantly replicates SVG code for the specific object or decomposes the object into subparts and combines them.

3.3 Evaluation

We converted each pixel matrix to an image with a simple Python script. Then, we conducted a classification study with three annotators. Each annotator described the generated images by specifying the *digits*, *letters*, and *punctuation symbols* they observed without knowing the possible set of characters. For *real-world objects*, we provided the correct answer. We let the annotator decide whether the respective object is recognizable because even a good image on an 8×8 pixel canvas is challenging to recognize without context. We counted the percentage of generations correctly classified or marked as recognizable for each experiment.

4 Results

This section presents the results of our experiments. The quantitative results are summarized in Table 1.

4.1 General Capabilities

We demonstrate the general capabilities of GPT-3.5 to create pixel matrices, including colorful images.

[1] https://platform.openai.com/tokenizer

[2] The objects of the mislabeled demonstrations are also generated and evaluated for *digits* and *letters*, potentially adversely impacting the displayed score by about 20%.

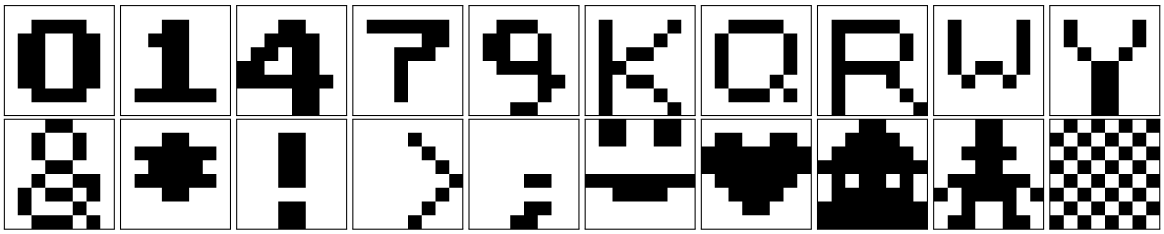

Figure 3: Selected positive examples of the baseline experiment: digits 0, 1, 4, 7, and 9; letters K, Q, R, W, and Y; the symbols ampersand, asterisk, exclamation point, and semi-colon; the objects sad face, heart, house, stick figure, and chess board.

Experiments	Digits	Letters	Punct.	Objs.
Baseline	73%	76%	45%	14%
Color Digits	56%	65%	32%	3%
Color RGB	15%	6%	10%	6%
GK Symbols	36%	37%	21%	2%
One Token	72%	72%	36%	12%
Wrong Label [2]	50%	67%	35%	12%

Table 1: Comparing experiment results: Percentage of recognizable images across generation tasks and image categories. Results obtained by human evaluation.

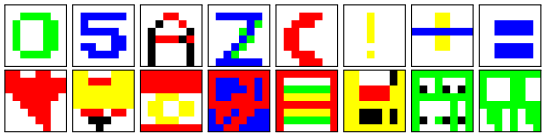

Figure 4: Images generated with pixel values for color: *digits*, *letters*, and *punctuation symbols* (top row); two hearts, suns, windows, and cacti (bottom row).

Basic experiment For *digits*, 73 % were correctly identified by the annotators, with the only exception being the number 4, which is only recognizable in three out of ten instances. For two-digit numbers (10 and 32), none of the generated examples accurately display the number. Instead, some pixel matrices represent other meaningful objects, like the letters A or H, or the number 9.

Almost all *letters* are consistently generated correctly and recognizable. Exceptions occurred with more complex letters. The letter E often features more than three horizontal lines, and W and M are occasionally wrong. Surprisingly, the letter V consistently appears as an X in the pixel matrix. The German umlauts Ä, Ö, and Ü fail to show corresponding vowels, often resulting in seemingly random pixel matrices.

Except for the percent and dollar signs, most *punctuation symbols* are generated correctly at least once in ten instances. Even complex symbols like the ampersand are successfully abstracted and generated on an 8×8 pixel matrix. More common symbols like the comma and exclamation marks are consistently generated. Left closing symbols yield good results, while not a single right closing symbol is generated correctly (see Figure 8). The output typically corresponds to the left symbol when requesting a right closing symbol.

Simple and common everyday items like the heart, sad face, and house yield mostly recognizable images. For the stick figure and cat, some instances are of high quality, while others seem completely random. Results for the remaining objects appear mostly random. Compare Figure 3 for selected positive examples.

Adding color Results for *digits*, *letters*, and *punctuation symbols* show slightly lower accuracy than simple black-and-white images. Complex letters such as W and M become even less recognizable than in the basic experiment. Certain letters comprised multiple colors.

The quality of the generated pixel matrices in the *real-world object* category declines. The images display vibrant colors, but the colors lack an association with the specified objects. The images for the star and sun do not contain more yellow, and those for a window or glass do not contain more blue. Only the cactus images consistently appear green. Compare Figure 4 for selected images from these experiments.

For the experiment with RGB color codes as pixel symbols, the model more often did not adhere to the format, i.e., generating an output that is not a pixel matrix. Only 15 % of the digits were recognized compared to 56 % with a pixel matrix of 5 color values.

4.2 Task Recognition vs. Task Learning

The results demonstrated in this section reveal insights into where the pixel matrix task lies on the spectrum between TR and TL. The corresponding discussion can be found in Section 5.2.

```
- The fundamental shape is a vertical line
positioned in the center of the 8x8 grid.
- At the top of the line, there is a small
horizontal line extending towards the right
side of the grid, connected to the vertical
line's midsection.
- The bottom part of the vertical line extends
slightly below the grid's baseline, forming a
slight curve.
```

Figure 5: Excerpt from a generated description for the digit 7 with incorrect shapes and inconsistencies.

```
The simplified 8x8 pixel representation of a
house consists of a square shape measuring 6
pixels in height and 6 pixels in width,
representing the main body of the house. On top
of the square, centered horizontally, there is
a triangle shape measuring 4 pixels in height
and 6 pixels in width, representing the roof.
The top row of the triangle is aligned with the
top row of the square.
```

Figure 6: Excerpt from a generated description for a real-world object, illustrating the ability to describe individual components accurately, yet assembling them incohesive.

Different pixel symbols When substituting 0 and 1 with the letters G and K the model generates pixel matrices in the correct format. Some instances are generated correctly. Still, overall, the results are significantly worse across all four categories (see Table 1).

Wrong labels The performance remains largely unchanged compared to the baseline when the labels of the demonstrations are wrong. The numbers in Table 1 for *digits*, *letters*, and *punctuation* are lower for this experiment because these datasets test objects that were wrongly labeled. Nevertheless, occasionally the model ignores the incorrect labels and produces correct pixel matrices. The zero-shot experiment outputs do not adhere to the format and reveal that the model needs examples to recognize the format.

One token per pixel symbol We observe slightly lower accuracy than the basic experiment, particularly for *punctuation symbols*. The output more often does not conform to the correct format, generating a message stating it is a language model and cannot generate images.

4.3 Breaking Tasks into Subtasks

Translating 01 pixel matrices to GK pixel matrices is successful in most instances (51 out of 60). Generated textual descriptions often lack coherence for

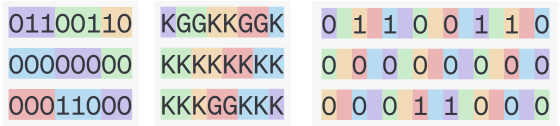

Figure 7: Comparing tokenization of different image formats: each color-coded sequence represents symbols combined into a single token. Images are screenshots from the OpenAI tokenizer webpage.

digits, *letters*, and most *punctuation symbols*. The described shapes appear random for all numbers except for 0, 1, and 8 (see Figure 5). Some level of abstraction is observed for *real-world objects*, but coherency keeps lacking. The generated texts mention useful shapes for an object but unrealistically combine them (see Figure 6).

According to our evaluation, 35% of SVG images of *real-world objects* are correct, a much higher score than for all other experiments. With only a few exceptions, the resulting images display the colors relevant to the desired object. Inaccurate images frequently present correct parts of objects, but the model fails to assemble the details in a hierarchical, cohesive way (compare Figure 9).

5 Discussion

In this section, we examine our experiment results. We discuss the overall performance in the pixel matrix task, assess if they tackle the task through TL or TR, and evaluate if breaking down the task into subtasks is a valid framework to explain LLM abilities.

5.1 LLMs Pixel Matrix Capabilities

GPT-3.5 showcases a solid ability to generate simple images in the form of pixel matrices without fine-tuning or including an image layer.

The capability of an LLM to use tokens trained to represent text for other purposes, such as representing pixels, highlights its potential beyond language generation (e.g., pattern completion). The results also indicate that language models possess information about different modalities, such as images. While image models rely on explicit images for training, LLMs have the advantage that textual descriptions inherently involve abstraction and the omission of (potentially) unnecessary details. Therefore, LLMs could help to increase the generalizability of image models.

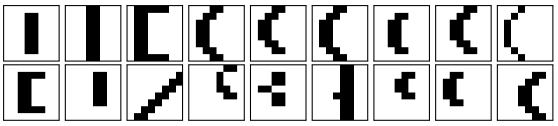

Figure 8: Left (top row) and right (bottom row) closing symbols, emphasizing the discrepancy in image quality for closely related objects.

5.2 Pixel Matrix Task Is Solved by Task Recognition

We found evidence that the pixel matrix task cannot be solved without TR. This evidence includes the deteriorated outcomes with uncommon (GK) pixel values and unaltered results despite incorrect examples in the prompt. Additionally, our zero-shot experiments confirm that relying solely on the instruction part of the prompt is insufficient for the model to learn the task. Altering the format to represent each pixel by one token does not improve the results despite simplifying the task and making it easier to learn.

Further, several pairs of objects with the same difficulty level are solved in only one of two instances. These object pairs include digits vs. number 10, letter A vs. letter Ä, and left closing symbols vs. right closing symbols (see Figure 8). If the model *understands* the tasks (rather than only replicating training instances), it would be able to solve either both or none of the pair's instances. We assume that the former object in each pair is significantly more present during training than the latter. Similarly, drawing the complex shapes of some *digits*, *punctuation symbols*, and *letters* appears more challenging than drawing simple *real-world objects*. For instance, copying an ampersand is challenging, whereas a simple smiley face is not. However, the results are consistently better for such complex shapes than for simple *real-world objects*.

Our findings indicate that the pixel matrix task is solved through TR, not TL. Thus, it shows that LLMs lack human-like understanding and generalization capabilities despite sometimes seeming otherwise. Rather, the large data corpus is extremely powerful and includes samples of uncommon tasks. Given the current architecture, we assert that LLMs will not attain understanding and reasoning. Enhancing current LLMs requires incorporating even more diverse data and improved training.

5.3 What Task Is Recognized?

The results of the baseline experiment show that the pixel matrix task has been learned only to a limiting degree during training. Language models are not extensively trained on text-to-pixel matrix data, making it challenging to generalize this task to new images by combining seen objects and concepts. We conclude that it is more appropriate to say that it recognizes the task of drawing a specific object instead of a general text-to-image task. Therefore, the model cannot generalize to pixel matrices for objects it has not encountered during training, as this ability is neither acquired during training nor inference. In conclusion, it's crucial to pinpoint the exact task that is recognized and solved to not overestimate the capabilities.

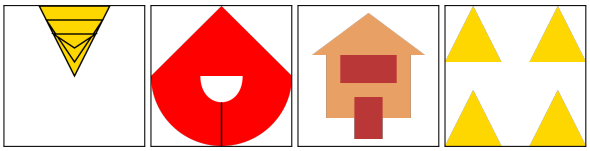

Figure 9: SVG-generated images (crown, wine glass, house, and lightning flash) exhibit a lack of cohesiveness in the arrangement of their components.

5.4 Decomposing Tasks to Evaluate Capabilities of LLMs

The previous results show that the model applies TR to solve the experiments with 01-pixel matrices. The pixel matrix task can be broken down into subtasks, such as accessing visual information about the object, decomposing objects into basic shapes, generating different shapes on a pixel matrix, and combining this information to form a pixel matrix representing a certain object. We show that the generated textual descriptions for *digits*, *letters*, and *punctuation symbols* were inadequate. Also, combining different shapes on a pixel matrix is challenging, as overlapping parts result in new tokens. The model relies on TR because it fails to solve these subtasks sufficiently.

In contrast, the *GK pixel* and the *SVG code* task are solved by solving subtasks. TR alone does not explain the results of the *GK pixel* matrix task because the model likely did not encounter such pixel matrices during training. We believe the model breaks down the task into creating a 01-pixel matrix and exchanging 0s and 1s with Gs and Ks. We show that an LLM can solve both individual subtasks. In the *SVG code* experiments, combining various shapes is accomplished by concatenating corresponding lines of code, which makes combining subtasks easier than for the pixel matrix task. The results show that the model draws different parts of objects but fails to put them together with

correct dimensions and spatial correlations (see Figure 9). This suggests that the model decomposes the object into its visual parts, recognizes these from training data, and combines them into one image rather than relying directly on TR.

Our framework of breaking down tasks into subtasks helps to assess TL and TR abilities. On the one hand, as in the case of pixel matrices, it helps to identify difficult subtasks and areas for improvement, such as targeted training data. On the other hand, the SVG and GK experiments demonstrate how our framework helps explain LLM capabilities and distinguish between TL and TR. When examining tasks like the GK pixel matrix, one might mistakenly conclude that the model is learning from scratch. However, decomposing the task reveals that simpler subtasks might be recognized from the training data. This prevents premature conclusions about TL but raises a philosophic question: How much must the model decompose the task, and how simple must the subtasks be to classify it as TL?

6 Conclusion

Our study demonstrates that LLMs can generate pixel matrices representing objects such as digits, letters, and simple real-world items. Our experiments show evidence of strong task recognition and limited task learning ability. We argue that breaking down complex tasks into smaller subtasks is a useful framework for evaluating and explaining LLM capabilities, preventing misleading conclusions when assessing the task's overall performance, and helping to distinguish between TL and TR.

As LLMs improve their ability to recognize tasks, locate relevant training data, and break down complex tasks, distinguishing between task learning and task recognition becomes increasingly complex. Although one might argue that this, at the core, represents task recognition of subtasks, future research is needed to explore how this process compares to human learning.

7 Limitations

Our evaluation captures broader trends, and our discussion is based on conclusions drawn from clear tendencies in the results rather than small differences. More thorough evaluations and comparisons across different models could strengthen the results. We have conducted a short ablation study with different models (see Appendix E.3). As the outputs depend on prompts, more prompt engineering and other prompts could have yielded different results.

We also rely on the generated outputs and general knowledge about LLMs to interpret their results because empirical analysis of the model's internal workings (e.g., evaluating attention patterns) is extremely resource-intensive. Thus, we interpret LLMs, highly complex statistical distributions, using simple human-like concepts (e.g., breaking down tasks).

References

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, et al. 2022. Flamingo: a visual language model for few-shot learning. In *Advances in Neural Information Processing Systems*.

Nicholas Asher, Swarnadeep Bhar, Akshay Chaturvedi, Julie Hunter, and Soumya Paul. 2023. Limits for learning with language models. In *Proceedings of the 12th Joint Conference on Lexical and Computational Semantics (*SEM 2023)*, pages 236–248, Toronto, Canada. Association for Computational Linguistics.

Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, et al. 2023. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. *arXiv preprint arXiv:2302.04023*.

Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pages 610–623.

Emily M. Bender and Alexander Koller. 2020. Climbing towards NLU: On meaning, form, and understanding in the age of data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5185–5198, Online. Association for Computational Linguistics.

Marcel Binz and Eric Schulz. 2023. Using cognitive psychology to understand gpt-3. *Proceedings of the National Academy of Sciences*, 120(6):e2218523120.

Sidney Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Pieler, Usvsn Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. 2022. GPT-NeoX-20B: An open-source autoregressive language model. In *Proceedings of BigScience Episode #5 – Workshop on Challenges & Perspectives in Creating Large Language*

Models, pages 95–136, virtual+Dublin. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Kranti Chalamalasetti, Jana Götze, Sherzod Hakimov, Brielen Madureira, Philipp Sadler, and David Schlangen. 2023. clembench: Using game play to evaluate chat-optimized language models as conversational agents. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 11174–11219, Singapore. Association for Computational Linguistics.

Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Shuming Ma, Zhifang Sui, and Furu Wei. 2023. Why can GPT learn in-context? language models secretly perform gradient descent as meta-optimizers. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4005–4019, Toronto, Canada. Association for Computational Linguistics.

Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards reasoning in large language models: A survey. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1049–1065, Toronto, Canada. Association for Computational Linguistics.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2022. Decomposed prompting: A modular approach for solving complex tasks. *arXiv preprint arXiv:2210.02406*.

Jing Yu Koh, Ruslan Salakhutdinov, and Daniel Fried. 2023. Grounding language models to images for multimodal generation. *arXiv preprint arXiv:2301.13823*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Andrew Lampinen, Ishita Dasgupta, Stephanie Chan, Kory Mathewson, Mh Tessler, Antonia Creswell, James McClelland, Jane Wang, and Felix Hill. 2022. Can language models learn from explanations in context? In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 537–563, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Belinda Z. Li, Maxwell Nye, and Jacob Andreas. 2021. Implicit representations of meaning in neural language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1813–1827, Online. Association for Computational Linguistics.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, et al. 2023. Starcoder: may the source be with you! *arXiv preprint arXiv:2305.06161*.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Suvir Mirchandani, Fei Xia, Pete Florence, brian ichter, Danny Driess, Montserrat Gonzalez Arenas, Kanishka Rao, Dorsa Sadigh, and Andy Zeng. 2023. Large language models as general pattern machines. In *7th Annual Conference on Robot Learning*.

Jane Pan, Tianyu Gao, Howard Chen, and Danqi Chen. 2023. What in-context learning "learns" in-context: Disentangling task recognition and task learning. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8298–8319, Toronto, Canada. Association for Computational Linguistics.

Reza Pourreza, Apratim Bhattacharyya, Sunny Panchal, Mingu Lee, Pulkit Madan, and Roland Memisevic. 2023. Painter: Teaching auto-regressive language models to draw sketches. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 305–314.

Archiki Prasad, Alexander Koller, Mareike Hartmann, Peter Clark, Ashish Sabharwal, Mohit Bansal, and Tushar Khot. 2023. Adapt: As-needed decomposition and planning with language models. *arXiv preprint arXiv:2311.05772*.

Evan Pu. 2022. Probing compositional understanding of chatgpt with svg. https://evanthebouncy.medium.com/probing-compositional-understanding-of-chatgpt-with-svg-74ec9ca106b4 accessed 12/10/2023.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

Ansh Radhakrishnan, Karina Nguyen, Anna Chen, Carol Chen, Carson Denison, Danny Hernandez, Esin Durmus, Evan Hubinger, Jackson Kernion, Kamilė Lukošiūtė, et al. 2023. Question decomposition improves the faithfulness of model-generated reasoning. *arXiv preprint arXiv:2307.11768*.

Laria Reynolds and Kyle McDonell. 2021. Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI EA '21, New York, NY, USA. Association for Computing Machinery.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. 2022. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.

Jamshaid Shahir. 2023. Writing code to produce images with chatgpt. https://towardsdatascience.com/image-generation-with-chatgpt-68c98a061bec accessed 12/10/2023.

Murray Shanahan. 2022. Talking about large language models. *arXiv preprint arXiv:2212.03551*.

Murray Shanahan, Kyle McDonell, and Laria Reynolds. 2023. Role play with large language models. *Nature*, pages 1–6.

Denis Shiryaev. 2022. Drawing mona lisa with chatgpt. https://neural.love/blog/chatgpt-svg accessed 12/10/2023.

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. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*.

Karthik Valmeekam, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. 2022. Large language models still can't plan (a benchmark for LLMs on planning and reasoning about change). In *NeurIPS 2022 Foundation Models for Decision Making Workshop*.

Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, Joao Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. 2023. Transformers learn in-context by gradient descent. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 35151–35174. PMLR.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2022. An explanation of in-context learning as implicit bayesian inference. In *International Conference on Learning Representations*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. 2022. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*.

A Prompt Details

This section supplements Section 3 by offering additional details and complete examples of the prompts used in our experiments.

A.1 Main Experiments

All prompts consist of one natural language instruction that describes the pixel matrix task by introducing pixels and the concept of pixel matrices and pixel symbols. This part is adjusted if the pixel symbols change. The second part of the prompts contains the demonstrations adapted to the object category and pixel symbols.

Independent of the pixel format, the prompt includes examples of numbers 2, 3, 8, and 6 for digits. The prompt examples for letters are A, F, G, and H. In the category of punctuation symbols, the examples are the forward slash, division sign, question mark, and the dot symbol, and for real-world objects, the examples are the smiley face, umbrella, tree, and deer.

See Figure 10 for a complete prompt for the baseline experiment.

A.2 Generating Descriptions

Refer to Figure 12 for the prompt used to generate descriptions of the objects' shapes and appearances.

A.3 Translating 01 to GK pixel matrices

Refer to Figure 13 for the prompt used to translate 01 pixel matrices to GK pixel matrices.

B Model Configurations

Pre-experiments revealed that a temperature of 1 was a good balance between randomness to ensure differences between each instance while maintaining the required format. We implemented a stop sequence for OpenAI models and restricted generated tokens to 100 for huggingface models. Apart from these modifications, we adhered to the default settings provided by the respective model APIs.

C Enumeration of All Objects

We evaluated the performance of creating pixel matrices across four categories: digits, letters, punctuation symbols, and real-world objects. Digits range from 0 to 9, with additional two-digit numbers, 10 and 32. The letter category comprises all Latin alphabet letters, including German umlauts (Ä, Ü, Ö) and eszett (ß), totaling 30 letters. The punctuation

```
Images displayed on a computer screen are a
collection of color dots, called pixels. If you
look really closely at the screen, you will be
able to see the individual pixels. The
collection of pixels that make up an image are
stored as a matrix.
We can represent different objects (e.g.,
numbers, letters, or shapes) by creating a
pixel matrix which consists of 0s and 1s. The
matrix should be of the size 8 by 8. Each entry
represents a pixel of a black or a white pixel.
That means the image has a display capable of 8
pixels in width and 8 pixels in height. Since
there are only 64 pixels in total the objects
to be displayed are significantly simplified.

Here is an example of an 8 by 8 pixel matrix
showing  three:
00000000
00111110
00000110
00111100
00001110
00000110
00111100
00000000
###
Here is an example of a grid of pixels that
form an image of  two:
00000000
00111100
01100110
00001100
00011000
00110000
01111110
00000000
###
Here is an example of a grid of pixels that
form an image of  eight:
00000000
00111100
01100110
00111100
01100110
00110100
00011000
00000000
###
Here is an example of a grid of pixels that
form an image of  six:
00000000
00111100
01100000
01100000
00111100
01100110
00111100
00000000
###
This is an example of a grid of pixels that
form an image of [object]:
```

Figure 10: The full prompt used to generate digits for the baseline experiments, with a placeholder for the requested digit.

```
Images displayed on a computer screen are
actually a collection of dots of color, called
pixels. If you look really closely at the
screen, you will be able to see the individual
pixels. The collection of pixels that make up
an image are stored as a matrix. Each pixel can
represent a different color.
We can represent different objects (e.g.,
numbers, letters, or shapes) by creating a
pixel matrix which consists different symbols
and each symbol stands for a different color.
In our 8x8 pixel matrix, we use the following
symbols to encode color: white (represented by
"0"), black ("1"), red ("2"), yellow ("3), green
("4"), and blue ("5"). This means each entry of
the matrix is either a 0, a 1, a 2, a 3, a 4, or
a 5. The matrix should be of the size 8 by 8.

Here is an example of an 8 by 8 pixel matrix
showing a smiley face:
31133113
31133113
33333333
33313333
13333331
31333313
33111133
33333333
###

Here is an example of a grid of pixels that
form an image of a tree:
00444400
04444440
44444444
44444444
44444444
00011000
00011000
00011000
###

Here is an example of a grid of pixels that
form an image of an umbrella:
00555500
05555550
55555555
00001000
00001000
00001000
00101000
00011000
###

Here is an example of a grid of pixels that
form an image of a deer:
10001000
11111000
01010000
11110001
01111111
01111111
01010101
01010101
###
```

Figure 11: Complete prompt used for experiments with new symbols representing different colors for the real-world object category. The first part and the demonstrations are adjusted.

```
Describe a simplified visual representation of
[object] which can be used to create an 8x8
pixel artwork of [object]. Emphasize only the
essential features for recognition, omitting
intricate details due to space constraints.
Deliver a concise description of the
fundamental shape and distinctive traits and if
necessary mention proportions, alignments, and
spatial relationships in a simplified rendition
of [object].
```

Figure 12: Prompt for generating descriptions of objects shapes which were added to pixel matrix prompt with the idea to enhance the outputs.

```
I want you to translate a 01 pixel matrix to a
GK pixel matrix. Replace every 0 with a G and
every 1 with a K.
Here is an example of a 01 pixel matrix:
00000000
00111100
01100000
01100000
00111100
01100110
00111100
00000000
###
Translation:
GGGGGGGG
GGKKKKGG
GKKGGGGG
GKKGGGGG
GGKKKKGG
GKKGGKKG
GGKKKKGG
GGGGGGGG
###

[...three more examples...]

Here is an example of a 01 pixel matrix:
[Exmaple 01 pixel matrix]
Translation:
```

Figure 13: Prompt for generating descriptions of objects shapes which were added to pixel matrix prompt with the idea to enhance the outputs.

digits	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 32
letters	A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, Ä, Ö, Ü, ß
punct.	comma, semicolon, exclamation point, equal sign, plus sign, hashtag, dollar sign, percent sign, ampersand, asterisk, left parenthesis, right parenthesis, left bracket, right bracket, left curly brace, right curly brace, less than sign, greater than sign, backslash, underscore, colon, single quote, double quote, at sign, caret
objs.	sad face, cup, heart, wine glass half full, cactus, key, skull, mouse, crown, lightning flash, padlock, cat, crab, a chess board, a house, coffee, car, window, chair, star, mountain, sun, boat, stick figure, fly

Table 2: Enumeration of all generated objects in our experiments.

symbol category encompasses 25 different symbols, while real-world objects consist of 26 diverse items. All objects are enumerated in Table C.

D Evaluation

We recruited 3 students from our university to annotate the images generated in our experiments. Figure 14 shows our whole set of instructions.

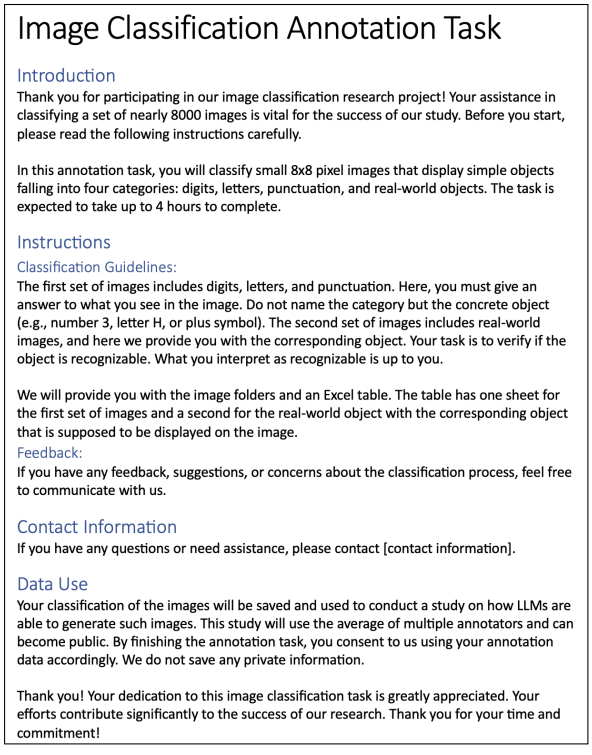
Image Classification Annotation Task

Introduction

Thank you for participating in our image classification research project! Your assistance in classifying a set of nearly 8000 images is vital for the success of our study. Before you start, please read the following instructions carefully.

In this annotation task, you will classify small 8x8 pixel images that display simple objects falling into four categories: digits, letters, punctuation, and real-world objects. The task is expected to take up to 4 hours to complete.

Instructions

Classification Guidelines:

The first set of images includes digits, letters, and punctuation. Here, you must give an answer to what you see in the image. Do not name the category but the concrete object (e.g., number 3, letter H, or plus symbol). The second set of images includes real-world images, and here we provide you with the corresponding object. Your task is to verify if the object is recognizable. What you interpret as recognizable is up to you.

We will provide you with the image folders and an Excel table. The table has one sheet for the first set of images and a second for the real-world object with the corresponding object that is supposed to be displayed on the image.

Feedback:

If you have any feedback, suggestions, or concerns about the classification process, feel free to communicate with us.

Contact Information

If you have any questions or need assistance, please contact [contact information].

Data Use

Your classification of the images will be saved and used to conduct a study on how LLMs are able to generate such images. This study will use the average of multiple annotators and can become public. By finishing the annotation task, you consent to us using your annotation data accordingly. We do not save any private information.

Thank you! Your dedication to this image classification task is greatly appreciated. Your efforts contribute significantly to the success of our research. Thank you for your time and commitment!

Figure 14: The complete instructions given to the human annotators.

E Complementary Results

Some figures showin images resulting from our experiments.

E.1 GPT-4 Color RGB

See Figure 16 for selected images generated by GPT-4 with RGB color codes as pixel symbols.

E.2 Images with Scalable Vector Graphics

Compare Figure 17 for images generated with SVG code.

E.3 Model Comparison

By default, we employed *gpt-3.5-turbo-0613* accessed through the OpenAI API. We further compared various models (Bloom (Scao et al., 2022), GPT-Neox20B (Black et al., 2022), Starcoder (Li et al., 2023)) on the pixel matrix task. Our primary aim is to contrast models fine-tuned on code with instruction-tuned LLMs based on our hypothesis that with TL, the performance should be independent of the frequency in the training data. Anticipating a higher occurrence of pixel matrices in code-based data and better instruction comprehension by classic LLMs, we hypothesize that the traditional models will excel if TL is applied. Conversely, with TR, the code models should demonstrate better performance.

GPT-3.5, with 175 billion parameters, demonstrates the best performance. In contrast, Bloom, also equipped with 175 billion parameters, adheres to the format but fails to solve any pixel matrix correctly. Gpt-Neox with 20 billion parameters does not generate any meaningful pixel matrix. The smallest model (15.5 billion parameters), Stracoder, fine-tuned specifically for code-related tasks, displays the best performance besides GPT-3.5. Its outputs for digits and letters are nearly as good as GPT-3.5, but it struggles with punctuation symbols and real-world objects.

GPT-4 shows significant improvements compared to GPT-3.5. It creates much more meaningful real-world objects on an 8x8 canvas (e.g., padlock, flash, key, or cactus), and it consistently creates colorful digits with RGB codes as pixel values, which GPT-3.5 cannot generate (see Figure 16).

E.4 16×16 pixel matrices

One thought was that an 8×8 pixel canvas might be too limiting, hindering the LLM from generating meaningful images of real-world objects like a cat

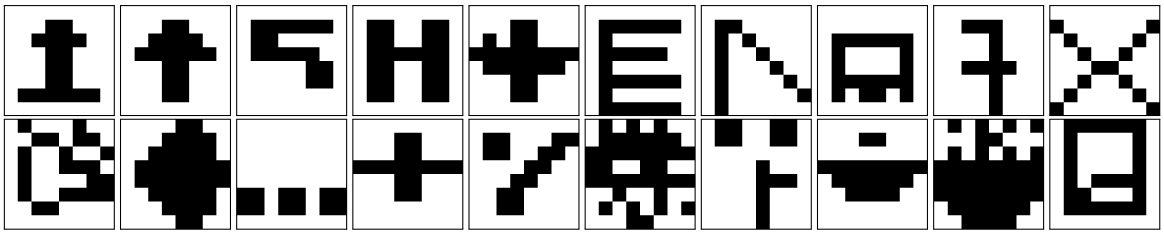

Figure 15: Negative examples from the baseline experiment: digits 1, 4, 5, 10, and 32; letters E, N, Ö, T, and V; the symbols at-sign, dollar sign, double quote, equal sign, and percent sign; the objects boat, cup, fly, star and wine glass.

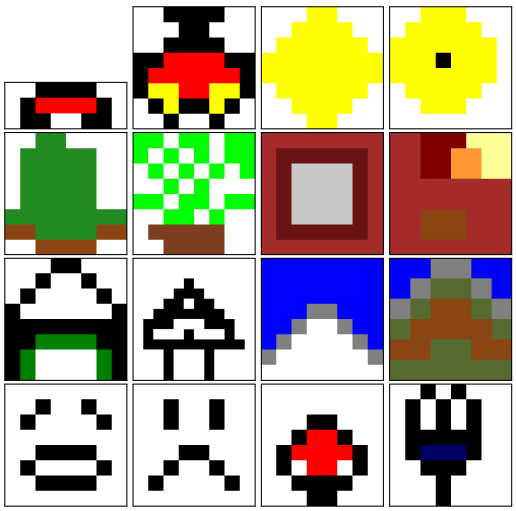

Figure 16: Selected instances of images generated by GPT-4 on an 8 by 8 pixel matrix with RGB values as pixel symbols showing two instances of each object: car, sun, cactus, coffe, house, mountain, sad face, wine glass. The magenta stripes resulting from translating slightly off output textual format to images.

or a boat. We conducted experiments with a pixel matrix of size 16×16. However, the overall results showed a slight degradation, and we did not see any improvements for specific objects where a larger canvas may be beneficial.

F Supplementary Discussion

We attributed some observed behavior to the autoregressive architecture of GPT-like models. For example, the letter V was never correctly generated and always resulted in the letter X. We assume that after correctly generating the first line of the matrix according to V, the subsequent generations favored a pixel matrix showing the letter X due to its higher frequency in training compared to that of a V. If the prompt would be truly "understood", the attention to the previous pixels should not overshadow the requested object.

In the *One Token* Experiment, overall results

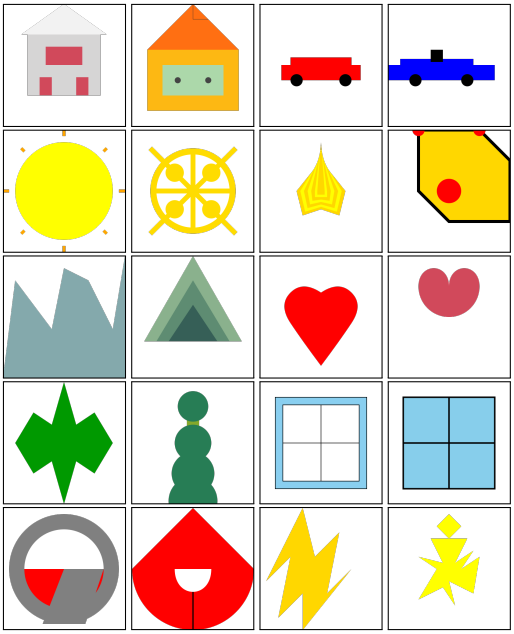

Figure 17: Example results from experiments with SVG code as image format showing two instances of each object: house, car, sun, crown, mountain, heart, cactus, window, wine glass, and lightning flash.

showed a slight decline, but notable improvements were observed with the new matrix format, especially for the object 'chess board'. An additional experiment with the chess board showed that while 42 out of 60 generations were correct with the basic matrix format, all 60 chess board generations were correct with the new format. We assume this might be because this object resembles a pattern completion task, which is less error-prone with fewer tokens Mirchandani et al. (2023).

We have conducted most of our experiments with a format that used more than ten different tokens to represent a pixel matrix as the tokenizer combines sequences of 0s and 1s. During our experiments with G and K as symbols, we assume that it translated in 01-matrices to the new format even though the tokens are different and the number of tokens changes (see Figure 7). Thus, the token embeddings represent even uncommon non-semantic similarities.