

# Exploring the potential of GPT-3 for image generation using pixel matrixes

Writeup LIGN167 – Janek Falkenstein

## Contents

Introduction .....	1
Model: The Pipeline .....	1
Datasets – Prompts and Examples .....	2
Training Data .....	2
Chain of thought.....	2
Test Data .....	3
Experiments and Results .....	4
Experiment 1 – Numbers.....	4
Experiment 2 – Numbers, Punctuations, letters, objects .....	5
Experiment 3 – Numbers, Punctuations, letters.....	5
Experiment 4 – Simple Chain of Thought .....	6
Experiment 5 – Descriptive Chain of Thought .....	6
Outlook .....	7
Conclusion .....	7
GitHub Link.....	7

## Introduction

Generative models have recently emerged as a powerful tool for various tasks in machine learning, such as language generation, image generation, and even music generation. Among these models, GPT-3 (short for "Generative Pretrained Transformer 3") stands out as one of the largest and most powerful language models to date. In this project, I explore the potential of using GPT-3 to generate images by representing them as pixel matrixes. My approach is inspired by the recent mainstream success of using large transformer-based models for image generation, such as DALL-E and BigGAN.<sup>1</sup>

To achieve the desired completions, I optimized the prompt to generate rows of numbers that form a pixel matrix either of size 8 by 8. The numbers are either 0 or 1, hence, representing a black-and-white image. I implemented a simple script that converts the output of GPT-3 completions into actual images and a small web application with Flask. The web application allows one to enter an object and shows the resulting generated image after making an API call to GPT-3. My goal was to optimize the completion for a simple test set consisting of images of numbers, letters, punctuation symbols, and simple objects. I conducted experiments with different prompts, including techniques like few-shot learning and chain of thought.

The results show that GPT-3 understands the task and has visual understanding about objects and their context. It is able to break down an object into its graphical representations and can indeed generate images in form of pixel matrixes when given the appropriate prompts. However, the results are still lacking consistency. Although the generated images are still relatively simple and not as detailed as those generated by state-of-the-art image generation models, they demonstrate the potential of using GPT-3 for creative tasks such as image generation, producing music, or videos.

## Model: The Pipeline

For this project, I built two applications. One is a simple web application to showcase my idea to others and to play around. The other is a program to conduct experiments. The web app was almost completely built on the OpenAI quick start python code which can be found in their GitHub library<sup>2</sup>. It uses the flask framework. The user types in an object which is appended to a prompt. Afterward, the app makes an OpenAI API call and uses the response to create a simple 8 by 8 image to display in the browser.

The program to conduct experiments is almost does the same thing. The differences lay in how the input and outputs are handled. Instead of a user manually typing in the objects, the program can read multiple inputs from the *test\_data.txt* file. Each line in that file will be appended to a prompt and is used in one API call. Between each API call, the program will sleep to avoid reaching the API call rate limit of 40,000 tokens per minute. The first part of the prompt which is part of the experiments is hard-coded and must be manually changed inside the python file to test different variations. The variations used in this project are saved inside the *experiments\_result* folder. The completions of the API calls are saved in the *raw\_response.txt* file. Additionally, each completion is converted to an image by filtering out all non-pixel matrix values in the response and with the help of the python PIL library. The images are upscaled to 400 by 400 pixels for better display and saved in the *generated\_images* folder.

In summary, to conduct experiments, the desired object descriptions must be put into the *test\_data.txt* file and after running the *experiments.py* file the raw responses are found in the *raw\_responses.txt* file. The generated images are found in the *generated\_images* folder.

In the experiments, I use the code-davinci-002 model as this has shown the most promising results during informal experiments. The temperature was set rather low to 0.3 since the limiting amount of testing capacity and the rate limit prevented me from increasing the Best Of parameter. The stop

---

<sup>1</sup> ChatGPT has written most of the first paragraph

<sup>2</sup> <https://github.com/openai/openai-quickstart-python>

sequence used is ###. However, these parameters have not been thoroughly tested and it could make sense to test them in further work.

## Datasets – Prompts and Examples

The focus of this project was to generate simple and small black-and-white images. The goal was to test how well GPT-3 can generate images of the following four categories: numbers, letters, punctuation, shapes, and simple objects (e.g., a heart).

To start the project different prompts without including examples were tested. However, it was not possible to get the intended completion. The complex task of this project is difficult to describe in a way that GPT-3 understands what it is supposed to do. It is interesting to note that this is probably not a weakness of GPT-3 but rather the problem of humans finding a correct and complete description of their task in a natural language. By giving the model a few examples of desired pairs of prompts and pixel matrixes, GPT-3 can learn to better understand the task and generate more accurate and consistent images. Few-shot learning has been proven to be a powerful tool for pre-trained models on specific tasks. Therefore, most of the experiments during this project focused on the few-shot training data given in the prompt. Nevertheless, the following paragraph was included in every prompt to introduce the task:

*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.*

*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.*<sup>3</sup>

## Training Data

I created the datasets for this project myself since I did not find an existing dataset with black and white images with 8 by 8 pixels. In addition, because of few-shot training, a few examples are sufficient. I have split the data into categories numbers, letters, punctuation, shapes, and objects. Each experiment without a chain of thought included the introduction prompt above and a set of examples in the format shown in Figure 1.

## Chain of thought

The chain of thought prompting method has been shown to improve the completion of GPT-3<sup>4</sup>. This method breaks down the examples inside the prompt into a series of intermediate reasoning steps. The use of the chain of thought can help GPT-3 to better understand the context and meaning of the words and phrases it is generating, which can improve the overall quality of the text it produces. Or in my case, I hope to improve the quality of the resulting images. For chain of thought prompts I added the following to the introduction paragraph of the prompt:

*In the following examples, the images are created step-by-step from top to bottom. The examples are not complete, it is possible to use other structures. It is very important that each pixel matrix consists of only 8 rows.*

I used two main types of chain of thought. First, I simply split up each pixel matrix into three parts as shown in Figure 2.

---

<sup>3</sup> The prompt was written with the help of OpenAI forum members: <https://community.openai.com/t/generating-pixel-matrixes-of-black-and-white-images-with-gpt-3/23325>

<sup>4</sup> Wei, Jason, et al. "Chain of thought prompting elicits reasoning in large language models." arXiv preprint arXiv:2201.11903 (2022).

*Here is an example of a grid of pixels that form an image of the number six:*

```
00000000
00111100
01100000
01100000
00111100
01100110
00110100
00000000
###
```

*Figure 1: few-shot training example*

*Here is an example of a grid of pixels that form an image of the letter Z:*

*First, the first two rows of the pixel matrix:*

```
11111111
00000011
```

*Then the middle part of the image:*

```
00000110
00011000
00110000
01100000
```

*Last, the last two rows of the pixel matrix:*

```
11000000
11111111
###
```

*Figure 2: few-shot training example with a simple chain of thought*

The results of prompts with examples as in Figure 2 were not satisfying. The second type of chain of thought describes what each part of the image is representing. Figure 3 shows two different examples.

*Here is an example of creating a 8 by 8 grid of pixels step-by-step to form an image of the number three:*

*The number 3 was originally denoted by three lines. Overtime these lines have been curved, rotated and connected to form the current representation.*

*The top two rows of an 8 by 8 pixel matrix representing the number three show the first curved line and it looks like an arc or semicircle:*

```
00111100
01000010
```

*The next part of the image of a three represents the second horizontal line with two additional slightly curved vertical strokes:*

```
00000110
00111100
00000110
```

*The bottom of the number three looks like the top but rotated by 180 degrees. It is a semicircle and the open part points upwards:*

```
01000010
00111100
```

*The last row of the 8 by 8 pixel matrix is not needed to form the number three and is therefore left white:*

```
00000000
###
```

*Figure 3: Two example of few-shot training with descriptive chain of thought*

*Here is an example of an 8 by 8 pixel matrix showing the capital letter F including the chain of thought:*

*Graphically the capital letter F consists of one large vertical line on the left and two horizontal lines: one at the top and one at in middle of the vertical line.*

*The first row of the grid of pixels forms the first horizontal line of the letter F:*

```
01111110
```

*The following part of the pixel matrix shows part of the large vertical line that is in between the two horizontal lines:*

```
01000000
01000000
```

*The next part of the letter F is the second horizontal line:*

```
01111110
```

*The bottom half of the image shows the remaining of the vertical line:*

```
01000000
01000000
01000000
01000000
###
```

I adjusted examples of Figure 3 further as the completions did often generate matrixes with more than 8 rows. Figure 4 shows which examples led to a completion with a consistent number of rows.

## Test Data

For testing the results of my experiments, I made a collection of 20 things to generate an image of. It includes examples of the categories of numbers, letters, punctuation, shapes, and real-life objects. Each prompt was tested to generate these 20 images and the image was assessed manually and rather informally by myself. 1 point was given if the image was unequivocal recognizable, 0.5 points if parts of the image are recognizable, and 0 points if the image is non-sense.

Here is an example of creating a 8 by 8 grid of pixels step-by-step to form an image of the number three:

The number 3 was originally denoted by three lines. Overtime these lines have been curved, rotated and connected to form the current representation.

The top two rows of an 8 by 8 pixel matrix representing the number three show the first curved line and it looks like an arc or semicircle (2 pixel rows):

00111100

01000010

The next part of the image of a three represents the second horizontal line with two additional slightly curved vertical strokes (3 pixel rows):

00000110

00111100

00000110

The bottom of the number three looks like the top but rotated by 180 degrees. It is a semicircle and the open part points upwards (2 pixel rows):

01000010

00111100

The last row of the 8 by 8 pixel matrix is not needed to form the number three and is therefore left white (1 pixel row):

00000000

This makes a total of  $2+3+2+1=8$  pixel rows.

###

Figure 4: Few-shot training example with a descriptive chain of thought and counting the number of pixel rows.

## Experiments and Results

Throughout the project, I have conducted several experiments to test different prompts. This chapter will describe the main ones and compare the results. After quickly noticing that it is very hard to design a prompt for the pixel matrix generation task without examples, I focused on finding the perfect set of examples to include in the prompt.

### Experiment 1 – Numbers

I included four number examples in the prompt of the first experiment. The numbers three, eight, six, and two were used.

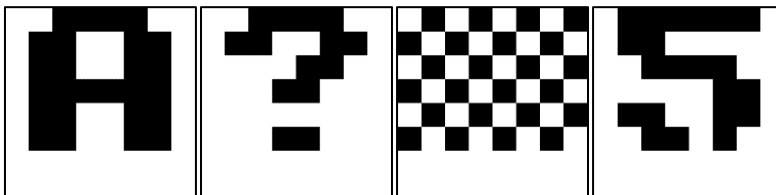


Figure 5: Correctly generated images with a simple prompt including four examples (Letter A, question mark, chess board, number 5)

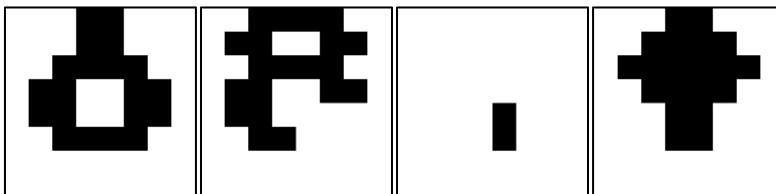


Figure 6: Incorrectly generated images with a simple prompt (triangle, letter R, exclamation point, animal)

The total score for all 20 generated images was 11. Most points were lost in the object category. On the one hand, GPT-3 correctly generated complex structures like the question mark and concepts that are far away from numbers (the chess board) although it was not consistent. On the other hand, simpler structures like the exclamation point and the triangle failed. Additionally, GPT-3 avoided the last two rows in all 20 images. This is likely because the last row was empty in all four examples of the prompt. I can not explain why also the second to last row was left empty by GPT-3 as this row was rather highly populated by black pixels in the prompt examples.

This first experiment shows a glimpse of how powerful GPT-3 can be. A simple prompt with less than 350 tokens is enough to teach it the concept of graphically representing symbols by a pixel matrix.

GPT-3 was specifically designed to process and generate natural language text and it is not explicitly designed to generate images or other visual content. Therefore, the ability of GPT-3 to generate pixel matrixes suggests that the model has learned to generalize beyond its training data and is able to generate novel content that it has not seen before.

### Experiment 2 – Numbers, Punctuations, letters, objects

After including only numbers as examples in the prompt in the first experiment, this experiment includes the same examples as before plus examples of the other categories (letters, punctuation symbols, and objects). In total 13 examples were provided.

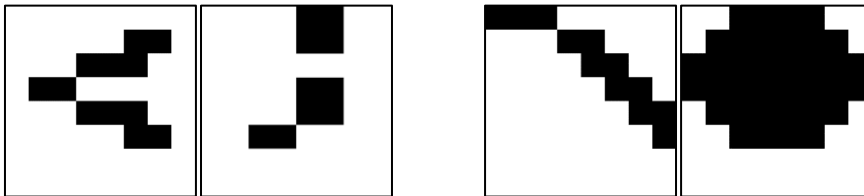


Figure 7: Generated images by a prompt including examples of different categories (greater than symbol, semicolon, number 7, and a skull)

The total score for all 20 generated images was 8. Some symbols like the greater than sign and the semicolon improved. However, on some test cases for example the number 7 this prompt performed worse. In addition, there was no improvement seen in the letter category and barely any in the object category. Even though examples of both categories were included in the prompt.

Overall, the result was worse compared to the prompt of the first experiment. Despite the fact that all the information that was included in the first prompt was also included during this experiment. This shows that adding more examples in the prompt is not the solution to improve completions of GPT-3. This is counterintuitive but confirms the fact that few-shot training is very different from classical machine learning.

### Experiment 3 – Numbers, Punctuations, letters

In the previous experiment, 13 different examples were provided in the prompt and it performed worse than with a simpler prompt. The idea for this experiment is to reduce the number of examples but still keep a variety of categories. The final prompt had two numbers, two punctuations, and two letters as examples.

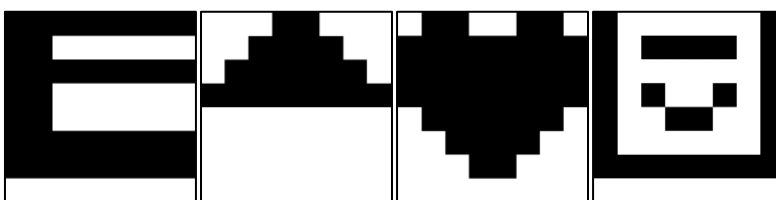


Figure 8: Generated images by a prompt that includes a total of 6 examples from three different categories (letter B, triangle, heart, face)

The total score for all 20 generated images was 12 which is higher than for experiment 1. This experiment scored relatively high on objects but failed on some images like letter B or question marks that have been generated successfully by other prompts. It must be noted that the results are not deterministic, and the differences could also be explained by randomization. However, the results seem to be consistently better than in experiment 2 where too many examples apparently puzzled GPT-3. All I changed in this prompt was to take out some examples in the prompt. Sometimes less is more. Since this experiment shows that GPT-3 is also able to generate objects like the heart and the face, it is very likely possible to build a prompt that generates consistently good images for all examples in the training set.

## Experiment 4 – Simple Chain of Thought

To further improve the prompt in the next experiments I tried the chain of thought method. The base of the prompt will be with two examples each of numbers, punctuation signs, and letters. The first idea was to adjust the examples according to the format seen in Figure 2. The results were very mixed. It generated a perfect chess board and heart and hints at the correct images are visible for most examples but also four completely white images and one completely black image was created.

## Experiment 5 – Descriptive Chain of Thought

Since experiment 4 was not successful, this experiment tries to include a proper chain of thought where each step is a logical intermediate step. Each image is broken down into different visual parts that can be drawn one after another. I also added a sentence to count the number of rows that have been generated to make sure each completion consists of exactly 8 pixel rows. An example of the prompt can be seen in Figure 3.

In this experiment, I saved the raw completion in addition to the generated images. These responses reveal how GPT-3 thinks when generating the images.

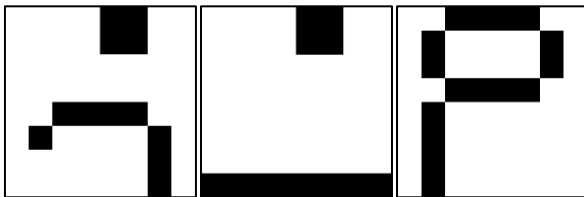


Figure 9: GPT-3 has the right thoughts when creating them but fails to put them together correctly.

The question mark is a punctuation mark that consists of a dot and a curved line.

The dot (2 pixel rows): {...}

The curved line (6 pixel rows): {...}

The exclamation point is a punctuation mark consisting of a dot and a vertical line.

The dot (2 pixel rows): {...}

The vertical line (6 pixel rows): {...}

The number 9 is represented by a semicircle that is connected to a vertical line.

The semicircle (4 pixel rows): {...}

The vertical line (4 pixel rows): {...}

Figure 10: The completion without the pixels of the examples that are also shown in Figure 9. This shows that GPT-3 understands more than the image suggest.

The results were mixed. The images themselves were often unrecognizable. However, looking at the raw completion one is able to understand how GPT-3 thinks. In most cases, GPT-3 has the right idea but is not 100% able to convert that into reasonable images. Even in complex structures like the face, GPT-3 can simplify the task in a way that would allow it to draw it onto an 8 by 8 image (Figure 11). It also seems like GPT-3 sticks to the building pieces that were given in the prompt (e.g., horizontal line, diagonal, or loop). It is possible that the prompt is limiting the completion in that way.

The graphic representation of a face consists of a circle and two lines that represent the eyes and a mouth.

The first part of the image is the top part of the face (2 pixel rows):

00111100

01000010

The next part of the face is the bottom part of the face (2 pixel rows):

01000010

00111100

The next part of the face is the left eye (2 pixel rows):

00111100

00111100

The next part of the face is the right eye (2 pixel rows):

00111100

00111100

The last part of the face is the mouth (2 pixel rows):

01111110

01111110

Figure 11: The completion shows that GPT-3 is aware of what shapes are needed to draw a simple face but fails to arrange them correctly.

## Outlook

The last experiment once again proved that GPT-3 is very powerful and understands a lot about contexts it has not been trained on directly. I assume that with the help of a chain of thought and a few well-designed examples, I can create a prompt that consistently performs very well on my test set. However, several adjustments to the prompt are still needed. Unfortunately, this project had a very limited time frame, and I could not proceed to do more tests.

I have several ideas on how to further improve the prompt in the future:

- Improve the set of examples (e.g., solve the problem of the empty last row, include more variety, and make sure no unwanted patterns can be detected by GPT-3).
- Adjust the chain of thought so GPT-3 has more shapes to draw with. Most of my examples included lines, diagonals, and loops. Not many other shapes were used to describe the parts of the image. The idea is to give GPT-3 a complete set of building pieces.
- Reduce the examples given in the chain of thought prompt. As shown in the first experiments sometimes less is more. Maybe six examples are too much, and GPT-3 tries to copy the examples too strictly. With fewer examples, GPT-3 might be able to find its own complete set of building pieces to generate images piece by piece.
- Increase the pixel size to 16 by 16 since it could be that it is not able to express itself on an 8 by 8 canvas. Drawing on an 8 by 8 pixel image needs a lot of abstraction.
- Include more colors by using more numbers in the pixel matrix. This would allow GPT-3 to express itself even more. This would be interesting especially when trying to generate objects like trees and animals.
- Let a native speaker or GPT-3 rewrite my prompts. Since GPT-3 is trained on natural language the exact semantics of the sentences can be crucial for the result. I let GPT-3 rewrite the beginning of my prompt but the first experiments did not show a clear improvement.

## Conclusion

In conclusion, the experiments show that GPT-3 has the potential to generate pixel matrixes that represent simple objects, such as numbers and shapes. By using appropriate prompts and techniques, such as few-shot learning and chain of thought, I was able to generate images that were reasonably accurate and diverse. However, I also observed that the quality and consistency of the generated images varied depending on the chosen prompts and examples, and I was unable to find a prompt that consistently produced very good results for multiple types of objects.

Overall, the results suggest that GPT-3 is a promising tool for generating images using pixel matrixes, but there is still room for improvement in terms of the quality and consistency of the generated images. Further research and experimentation with different prompts and fine-tuning techniques may be necessary to fully leverage the capabilities of GPT-3 for this task. Additionally, it would be interesting to explore the use of GPT-3 for generating larger and more complex images by using larger pixel matrixes or other representation methods.<sup>5</sup>

## GitHub Link

<https://github.com/J-falkenstein/lign167>

---

<sup>5</sup> This conclusion was written mostly by ChatGPT.