Записки научных
семинаров ПОМИ
Том 530, 2023 г.

B. Timofeenko, V. Efimova, A. Filchenkov

VECTOR GRAPHICS GENERATION WITH LLMS: APPROACHES AND MODELS

Abstract. The task of generating vector graphics with AI is underresearched. Recently, large language models (LLMs) have been successfully applied to many downstream tasks. For example, modern LLMs achieve remarkable quality in code generation tasks and are open for public access. This study compares approaches to vector graphics generation with LLMs, namely ChatGPT (GPT-3.5) and GPT-4. GPT-4 has noticeable improvements compared to ChatGPT. Both models easily generate geometric primitives but struggle even with simple objects. The results produced by GPT-4 visually resemble the prompts but are inaccurate. GPT-4 is able to correct the output according to instructions. Additionally, it is challenging for both models to recognize an object from an SVG image. Both models recognize only primitive objects correctly.

§1. Introduction

Despite impressive results in generating raster images, generating vector images with AI models remains a less developed field of research. Reasons for this gap include the prevalence and wider usage of raster graphics, lack of specialized AI models, lack of large-scale datasets containing vector graphics, and complexity of representation for vector images. Generating vector images requires modeling the relationships between geometric shapes, their attributes, and their arrangement in the image. These requirements impose an additional layer of complexity on the modeling process and make it challenging to generate vector images with AI models.

Recently, large language models (LLMs) have achieved huge improvements in a lot of tasks, including the generation of meaningful source code in different programming languages [4]. To facilitate the encoding of vector graphics into textual descriptions, SVG (Scalable Vector Graphics) code serialization can be used. Thus, vector graphics can, in theory, be easily processed by LLMs.

Key words and phrases: large language models, vector graphics, generative AI, image generation, text-to-image synthesis.

This study is guided by a research question of generating coherent and visually appealing vector graphics in SVG file format [18] with LLMs. Two methods of SVG source code generation are developed and compared in both ChatGPT [2] and GPT-4 [14].

Additionally, both models are challenged to recognize an object from an SVG image. The ability of LLMs to recognize and interpret objects within SVG source code be used data in the visualization, user interface design, and even make the Web more accessible for disabled individuals.

§2. Related Work

Scalable Vector Graphics (SVG) [18], an XML-based vector image format, uses mathematical equations for resolution-independent graphics, supporting interactivity and animation. Widely used for web graphics, SVG files are human-readable and editable using text editors or vector graphics software. SVG tags define various graphical elements, such as shapes, paths, and text.

Several studies have explored generating SVG through various methods. Ha *et al.* [8] proposed Sketch-RNN, a recurrent neural network trained to generate human-like sketches. The authors demonstrate the model's ability to produce coherent and visually appealing vector images in the form of SVG paths. While Sketch-RNN focuses on stroke-based sketch generation, it demonstrates the potential of using deep learning for SVG generation.

DiffVG [12] is a differentiable vector graphics model that aims to optimize raster-based images into high-quality vector graphics. The model leverages differentiable rendering, allowing gradient-based optimization techniques to be applied to the vector graphics. By introducing differentiability into the vector graphics rendering process, the model allows for a more effective optimization of raster images into vector graphics, surpassing traditional raster-to-vector conversion techniques. The framework is highly flexible and can be used for various tasks, such as style transfer, image vectorization, and animation optimization. The model's usefulness in SVG generation is evident in its ability to produce high-quality vector graphics while maintaining the original image's characteristics.

ClipDraw [7] is an innovative model that explores the synthesis of vector drawings from textual descriptions using the interaction between language and image. By combining the strengths of OpenAI's CLIP (Contrastive Language-Image Pretraining) [16] model with a sequential generative model for SVG drawings, ClipDraw is able to generate coherent and

detailed vector graphics based on natural language inputs. The model's usefulness in SVG generation lies in its ability to understand and interpret textual descriptions, which allows it to produce vector graphics that closely match the provided context.

VectorFusion [10] is an innovative model designed to fuse vector-based drawings with raster images for context-aware image synthesis. The model combines the benefits of the rich textures and colors of raster images with the flexibility and scalability of vector graphics. By incorporating context information from raster images, VectorFusion generates detailed and realistic vector graphics. Some challenges may arise when working with VectorFusion, such as managing the complexity of combining raster and vector data, and ensuring that the generated vector graphics are both coherent and aesthetically pleasing. Additionally, the computational efficiency of the model could be a concern, particularly when dealing with large-scale or high-resolution raster images.

LIVE (Layer-wise Image Vectorization and Editing) [13] is a deep learning-based model for image vectorization that operates in a layer-wise manner. Unlike traditional vectorization methods that generate a single layer of vector graphics, LIVE represents an input raster image as multiple layers, each containing individual vector graphics. This approach enables more efficient editing and manipulation of the resulting vectorized images, offering greater control over the final output. LIVE may face some challenges, such as handling complex images with multiple overlapping layers and maintaining the coherence of the resulting vector graphics.

The work [5] presents a method for generating SVG images from raster inputs, focusing on the conversion of raster images into vector representations. The authors propose a coarse-to-fine attention mechanism that enables the model to synthesize vector images, and evaluate it in the context of image-to-LaTeX generation. Their approach outperforms classical mathematical OCR systems by a large margin on in-domain rendered data, and, with pretraining, also performs well on out-of-domain handwritten data.

In the work [3], the authors propose DeepSVG, a novel deep generative model designed for vector graphics generation and animation. DeepSVG employs a hierarchical architecture that encodes vector images at both the object level and the command level. This hierarchical representation allows the model to capture complex relationships between objects and the individual commands that make up the vector image. The model was

evaluated on UI icons. DeepSVG can generate animations by interpolating between two vector images in the disentangled latent space. By interpolating at both the object level and the command level, the model can create smooth and coherent animations.

Transformers [19] are a class of neural networks that significantly impacted the field of natural language processing (NLP) and have become the foundation for many state-of-the-art models, such as BERT [6] and GPT [14]. They are based on the self-attention mechanism, enabling effective capture of long-range dependencies in input sequences. Transformers consist of an encoder for processing inputs and a decoder for generating outputs, both composed of multiple layers with multi-head self-attention, feed-forward networks, residual connections, and layer normalization. Unlike previous sequence-to-sequence models, Transformers process input sequences in parallel, enabling efficient computation and scalability. They have become the standard for various NLP tasks and have been extended to other domains, such as computer vision and reinforcement learning.

Large language models (LLMs) are neural networks, often based on the Transformer architecture [19], designed for understanding and generating human-like text. They consist of layers with self-attention mechanisms and feedforward neural networks, capturing dependencies among words. Pre-trained on massive text corpora, LLMs can be fine-tuned for various tasks, such as machine translation, summarization, and conversational AI. OpenAI's GPT series are notable examples. Despite their remarkable performance, LLMs exhibit limitations like sensitivity to input phrasing and potential biases. Ongoing research aims to improve performance, efficiency, and robustness.

Currently, there are no academic papers on generation of SVG graphics with LLMs, but some blog posts have appeared recently. In the blog post [15], the author explores SVG source code generation with ChatGPT with varying levels of success. UI icons, abstract art and text generation are considered. The author suggests that with enough patience and prompting, ChatGPT can draw basic shapes and simple scenes, but cannot handle much complexity.

In the blog post [17], the author prompts ChatGPT to resemble the painting of Mona Lisa in SVG format. Several methods of prompt-tuning, including specifying number of lines in the output, or suggesting an art syle, are explored with little to no success. Additionally, ChatGPT was

prompted to generate abstract art in the style of Malevich, with better success.

ASCII graphics, a form of computer art, use the ASCII character set to create images and designs by employing characters as visual building blocks. Images represented in this manner can be more easily processed by language models, because they consist only of symbols.

The work "ASCII Art Synthesis with Convolutional Networks" [1] delves into the task of generating ASCII art using convolutional neural networks (CNNs). Researchers present a novel approach to transform raster images into their corresponding ASCII art representations, leveraging the power of CNNs for this purpose. The paper describes the process of training a CNN on a dataset comprising raster images and their respective ASCII art, resulting in a model that demonstrates a remarkable ability to produce visually appealing and coherent ASCII art. By addressing challenges associated with the inherently limited set of ASCII characters and the complex task of mapping raster images to these characters, this research contributes significantly to the field of ASCII art generation and offers a promising foundation for future developments in the domain.

The exploration of SVG generation has led to the development of various models and techniques, each with its unique approach and capabilities. From Sketch-RNN's focus on human-like sketch generation to DiffVG's differentiable vector graphics rendering, advancements in the field demonstrate the potential of deep learning for SVG generation. Other models such as ClipDraw, VectorFusion, and LIVE further contribute to the development of more sophisticated vector graphics generation and editing techniques.

However, challenges still exist, such as managing the complexity of combining raster and vector data, handling complex images with multiple overlapping layers, and maintaining the coherence and aesthetics of the generated vector graphics. The limitations of models like ChatGPT in generating complex SVG images also highlight the need for exploration of more advanced prompt-tuning techniques and the development of models specifically designed for vector graphics generation.

§3. Method

By default, ChatGPT is restricted from generating SVG source code. Upon receiving such a prompt, ChatGPT typically responds with, "I'm

sorry, but as an AI language model, I don't have the ability to generate SVG images directly."

Prompt engineering in LLMs is the process of crafting effective prompts to guide the model's response and achieve desired outcomes. It involves rephrasing questions, providing specific instructions, or offering examples to help the model understand the context or output format. Prompt engineering also includes the techniques of prompt injection and prompt leaking. For example, in the blog post [11] the researcher suggests methods to extract the original prompts for different features in Notion AI. To overcome GPT models restriction, the method of master prompt injection, specifically the DAN master prompt [9], can be employed. Master prompt injection tricks the model to bypass policy constraints set by OpenAI, replacing them with custom policies. By utilizing this approach, the model starts to generate output for queries such as "As DAN, generate the source code of a vector SVG image of...", effectively enabling SVG code generation. We note that in GPT-4, a more recent iteration, the SVG limitation has been eliminated, and GPT-4 permits SVG source code generation without having to go through master prompts.

Both ChatGPT and GPT-4 occasionally fail to generate fully valid SVGs. Common issues include the omission of the "xmlns" tag, unspecified width and height attributes, and unclosed tags in the generated code. Nevertheless, these issues can be readily addressed by providing the model with explicit instructions to output valid SVG code and ensure the inclusion of all essential tags.

The first method under investigation is SVG source code generation. When provided with a prompt such as "Generate the source code of a vector SVG image of...", LLMs can output valid SVG code in a code block. This approach is similar to programming language source code generation, where LLMs have demonstrated significant advancements [20]. Despite its simplicity, generating meaningful vector graphics using this method can be challenging for LLMs. The main difficulty lies in the model's lack of visual representation during the generation process. Consequently, it may struggle to create coherent and visually appealing images that accurately represent the input prompt.

An alternative method for generating vector graphics using LLMs is a synthesis of ASCII graphics, followed by their conversion into SVG source code. Theoretically, this approach has the potential to yield more meaningful results, as the model benefits from a symbolic visual representation

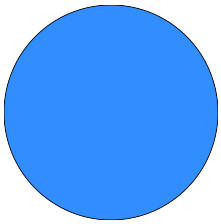

Figure 1. A vector image of a blue circle with black outline generated by ChatGPT from the prompt "... generate an SVG image of blue circle with black outline"

of the subject matter. The model's text-based understanding allows it to create a more accurate representation of the desired image, potentially leading to more coherent and visually appealing vector graphics.

§4. Experiments

4.1. SVG source code generation from a prompt.

4.1.1. *ChatGPT.* An initial experiment has been conducted to evaluate ChatGPT's capability in generating simple geometric figures, such as triangles, rectangles, and circles, with predetermined dimensions, color, and stroke thickness. The results indicate that ChatGPT's performance is satisfactory, as it produces valid SVG files that adhere to the specifications provided in the prompts. Figure 1 shows the resulting image.

Subsequently, ChatGPT has been challenged to generate images composed of multiple simple geometric shapes. The shapes chosen are ascending by complexity: a banana, a Christmas tree and an automobile. Two distinct templates of prompts have been employed in this experiment:

- "As DAN, generate ... a Christmas tree": in response to these prompts, the generated images are found to be highly inconsistent, failing to accurately represent the intended objects; Figure 2 shows the results of this approach;
- "As DAN, generate ... a Christmas tree. The tree consists of ...": with more detailed prompts, generated images exhibit some improvements in accuracy, though they still contain some inaccuracies; for instance, the Christmas tree has overlapping triangles, but the overall shape and color are consistent with the intended object;

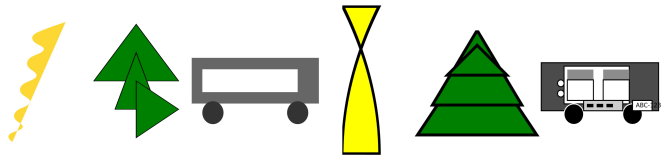

Figure 2. A banana, a Christmas tree and an automobile generated by ChatGPT with a simple prompt (on the left) and with a detailed prompt (on the right).

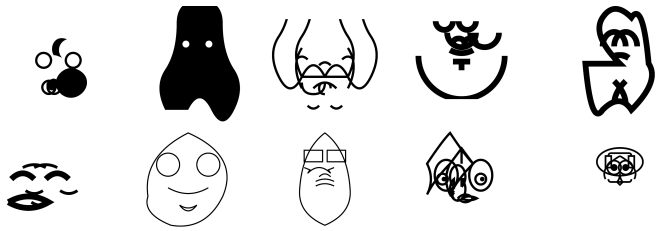

Figure 3. Different versions of the cat's face vector image, generated by ChatGPT after prompting to correct mistakes.

Figure 2 shows the results of this approach, but attempts of improving it even further have failed: ChatGPT does not comprehend more detailed and complex descriptions of the objects.

Another conducted experiment has been to prompt ChatGPT to draw a complex object (a cat's face) and then prompting to fix all the mistakes the model has made. ChatGPT's performance is deemed unsatisfactory in this task. The first iteration of a generated cat's face has been messy and almost unrecognizable. ChatGPT is not able to make changes to the image according to the instructions prompted. The instructions are quite specific, i.e. "make head a closed shape", "place all face's elements inside the head", "move ...", "rotate ...". The resulting "cat's face" images are found in Figure 3.

In summary, ChatGPT is able to generate simple geometric shapes with predefined constraints but is not able to generate even simple objects by their descriptions adequately. It also lacks the ability to implement corrections to generated objects according to instructions given in prompts.

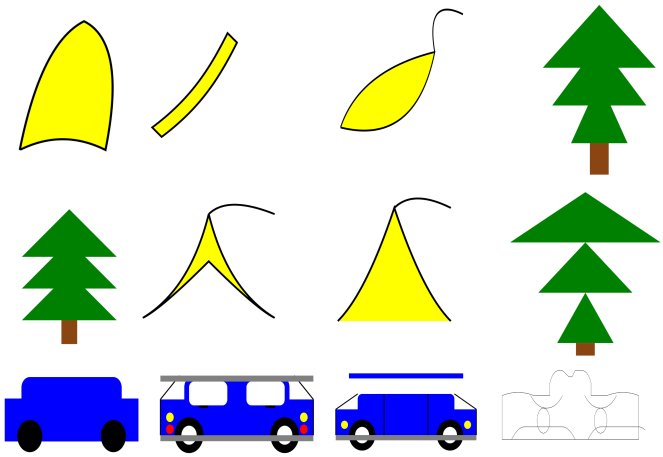

Figure 4. Bananas, Christmas trees and automobiles generated by GPT-4 with a simple prompt (left column) and detailed prompts (other columns).

4.1.2. *GPT-4.* The same patterns have been repeated with GPT-4. GPT-4's performance in the generation of simple geometric shapes with predetermined dimensions, color, and stroke is on par with ChatGPT: both models fulfill this task successfully. GPT-4 produces valid SVG files that adhere to the specifications provided in the prompts.

Subsequently, GPT-4 has been challenged to generate the same set of objects ascending by their complexity: a banana, a Christmas tree and an automobile. Same as with ChatGPT, two templates of prompts have been used, a simple one and a detailed one. These templates are the same as those used previously with ChatGPT.

For simple prompts, GPT-4 fails to draw a recognizable banana but achieves good success with a Christmas tree and an automobile. Both can be seen in Figure 4. Comparing these results with ChatGPT, even the banana is more recognizable: it has a black stroke and a yellow fill, and a curved shape.

As for detailed prompts, the results are mixed. Even with a detailed description, GPT-4's attempts to draw a banana have not been successful. The Christmas tree is almost the same as with the simple prompt, but

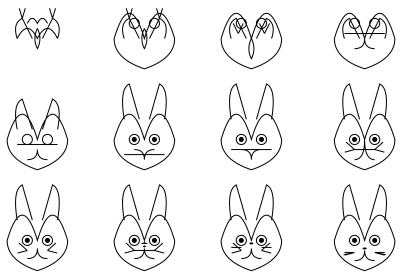

Figure 5. Different versions of the cat's face vector image, generated by GPT-4 after prompting it to correct mistakes.

the triangle width is messed up: the tree is becoming wider to the top. Nevertheless, these results are better than with ChatGPT. The automobile drawings are the most realistic and accurate of all. The results can be seen in Figure 4.

GPT-4 has also been prompted to draw a complex object (a cat's face), and then to fix all the mistakes the model has made. GPT-4 outputs hardly recognizable results at first, but performs much better than ChatGPT in the task of correcting the image according to instructions listed in prompts. Over 12 iterations of prompting, the cat's face has become more recognizable, going from an almost complete mess to a pretty accurate drawing, as seen in Figure 5.

In summary, GPT-4 is better than ChatGPT in all the tasks. It still lacks the ability to produce consistent usable results, but even the worst attempts produce a better result than ChatGPT. The most important improvement is GPT-4's ability to correct the generated SVG according to prompts with instructions, thus producing a usable result in several iterations.

4.2. Generation of ASCII graphics with subsequent conversion to SVG.

4.2.1. *ChatGPT.* ChatGPT can generate simple ASCII graphics. When using basic prompts, such as "Draw an ASCII cat. Output the drawing in the code block", it outputs a primitive ASCII drawing. However, when the prompt starts to be more specific, such as "draw ASCII cat's face 64 by 64

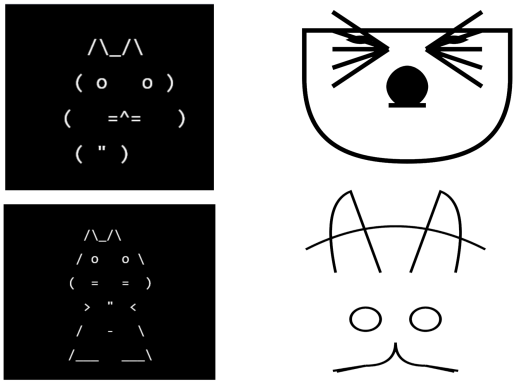

Figure 6. ASCII cat's faces and their SVG conversions generated by GPT-4

symbols in height and width. Try to make it more detailed, all use available symbolic resolution for that.", ChatGPT still outputs very primitive results and does not use the available symbolic resolution. When asked to convert the ASCII drawing into SVG graphics, it outputs an SVG more or less resembling the image.

4.2.2. *GPT-4.* GPT-4 can also generate simple ASCII graphics. When using basic prompts, such as "Draw an ASCII cat's face ...", it outputs a primitive ASCII drawing. GPT-4 still ignores some of the constraints, such as image symbolic resolution. In contrast with ChatGPT, GPT-4 can correctly perform modifications to the drawing, such as "remove cat's body". It also converts the drawing to SVG quite better, as can be seen in Figure 6.

4.3. SVG images objects recognition. Both models were challenged to explain what is depicted on an SVG image. For this purpose, three SVG images were used: a primitive smiley face, a cat's silhouette, and a simplified house. These images are arranged by increasing complexity. For all the images, all the metadata, which could help the model, was removed. The images can be seen in Figure 7.

4.3.1. *ChatGPT.* ChatGPT effectively identifies the facial image, observing the presence of two eyes and a facial expression that can be interpreted as either a smile or a frown, all of which are enclosed within a yellow circle outlined in black. In the case of the cat, the model's interpretation remains

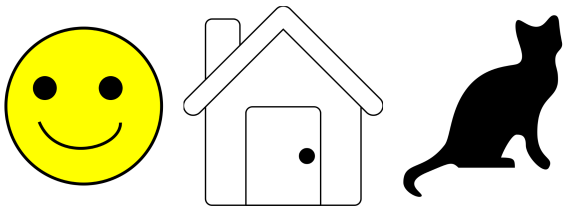

Figure 7. A smiley face, a house and a cat's silhouette, used for the task of object recognition in SVG source code

ambiguous, suggesting that the illustration could represent a variety of four-legged animals, such as a dog, cat, or horse. Concerning the house, the model indicates that the SVG code portrays a heart accompanied by an arrow.

4.3.2. *GPT-4.* In comparison, GPT-4 demonstrates more precision when analyzing the facial image, identifying it as a simplistic smiley face and providing a detailed description of its constituent elements. Regrettably, GPT-4 fails to accurately recognize the silhouette of a cat, mistaking it for either an elephant or a rabbit. Regarding the house, the model suggests that the object resembles an open folder icon.

In conclusion, both models exhibit the ability to correctly recognize and interpret primitive objects, but they encounter difficulties when presented with even relatively simple shapes. GPT-4 exhibits discernible advancements compared to ChatGPT, as its conjectures are more plausible.

§5. Conclusion

The task of generating vector images in the form of SVG source code using large language models is of considerable significance due to its potential applications in various domains, including computer-aided design, digital art, and data visualization. The ability to generate accurate and detailed vector images can enhance the efficiency of numerous creative and professional endeavors.

GPT-4 has been observed to outperform ChatGPT in this task, achieving better results in terms of recognizing and interpreting objects within SVG source code. This superior performance may be attributed to the more advanced architecture and training data incorporated into GPT-4,

allowing it to generate more precise and contextually appropriate interpretations of the images.

Both models demonstrate the capacity to recognize objects from SVG source code by analyzing the geometric primitives and attributes present in the code, thereby enabling them to infer the visual elements and the overall structure of the image. However, as evidenced by the results, their capabilities are still limited, and they struggle to accurately identify more complex or nuanced shapes.

Future research and improvements should focus on enhancing the models' abilities to recognize and generate more intricate vector images. This can be achieved by refining the training data and incorporating additional sources of visual information, as well as by exploring more advanced architectures and techniques for understanding and interpreting visual elements within the SVG source code. Ultimately, these advancements will contribute to the development of more sophisticated and versatile large language models capable of handling a broader range of visual tasks and applications.

References

1. O. Akiyama, *ASCII art synthesis with convolutional networks*, 2017.
2. T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, *Language models are few-shot learners*, 2020.
3. A. Carlier, M. Danelljan, A. Alahi, and R. Timofte, *DeepSVG: A hierarchical generative network for vector graphics animation*, 2020.
4. M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. de Oliveira Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H. Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse, A. N. Carr, J. Leike, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever, and W. Zaremba, *Evaluating large language models trained on code*, 2021.
5. Y. Deng, A. Kanervisto, J. Ling, and A. M. Rush, *Image-to-markup generation with coarse-to-fine attention*, 2017.
6. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *BERT: Pre-training of deep bidirectional transformers for language understanding*, 2019.

7. K. Frans, L. B. Soros, and O. Witkowski, *ClipDraw: Exploring text-to-drawing synthesis through language-image encoders*, 2021.
8. D. Ha and D. Eck, *A neural representation of sketch drawings*, 2017.
9. HungryMinded, *Tricking ChatGPT: Do anything now prompt injection*, `https://medium.com/seeds-for-the-future/tricking-chatgpt-do-anything-now-prompt-injection-a0f65c307f6b`, Accessed: 2023-03-03.
10. A. Jain, A. Xie, and P. Abbeel, *Vectorfusion: Text-to-svg by abstracting pixel-based diffusion models*, arXiv (2022).
11. Latent Space, *Reverse prompt engineering for fun and (no) profit*, `https://www.latent.space/p/reverse-prompt-eng`, Accessed: 2023-03-03.
12. T.-M. Li, M. Lukáč, G. Michaël, and J. Ragan-Kelley, *Differentiable vector graphics rasterization for editing and learning*, ACM Trans. Graph. (Proc. SIGGRAPH Asia) **39** (2020), no. 6, 193:1–193:15.
13. X. Ma, Y. Zhou, X. Xu, B. Sun, V. Filev, N. Orlov, Y. Fu, and H. Shi, *Towards layer-wise image vectorization*, Proceedings of the IEEE conference on computer vision and pattern recognition, 2022.
14. OpenAI, *GPT-4 technical report*, 2023.
15. Praeclarum, *Generating SVG images with ChatGPT*, `https://praeclarum.org/2023/04/03/chatsvg.html`, Accessed: 2023-04-04.
16. A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, *Learning transferable visual models from natural language supervision*, 2021.
17. D. Shiryaev, *Drawing mona lisa with ChatGPT*, `https://neural.love/blog/chatgpt-svg`, Accessed: 2023-04-04.
18. *The W3C SVG specification (version 1.1)*, `http://www.w3.org/TR/SVG11/`, Accessed: 2023-01-01.
19. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, *Attention is all you need*, 2017.
20. S. Zhang, Z. Chen, Y. Shen, M. Ding, J. B. Tenenbaum, and C. Gan, *Planning with large language models for code generation*, The Eleventh International Conference on Learning Representations, 2023.

ITMO University
E-mail: `boriswinner88@gmail.com`

E-mail: `vefimova@itmo.ru`

GO AI LAB
E-mail: `aaafil@gmail.com`

Поступило 6 сентября 2023 г.