Leveraging Large Language Models for Scalable Vector Graphics-Driven Image Understanding

Mu Cai*1, Zeyi Huang*1, Yuheng Li1, Haohan Wang2, Yong Jae Lee1

¹University of Wisconsin–Madison ²University of Illinois Urbana-Champaign

Abstract

Recently, large language models (LLMs) have made significant advancements in natural language understanding and generation. However, their potential in computer vision remains largely unexplored. In this paper, we introduce a new, exploratory approach that enables LLMs to process images using the Scalable Vector Graphics (SVG) format. By leveraging the XML-based textual descriptions of SVG representations instead of raster images, we aim to bridge the gap between the visual and textual modalities, allowing LLMs to directly understand and manipulate images without the need for parameterized visual components. Our method facilitates simple image classification, generation, and in-context learning using only LLM capabilities. We demonstrate the promise of our approach across discriminative and generative tasks, highlighting its (i) robustness against distribution shift, (ii) substantial improvements achieved by tapping into the incontext learning abilities of LLMs, and (iii) image understanding and generation capabilities with human guidance. Our code, data, and models can be found here https://github.com/mu-cai/svg-llm.

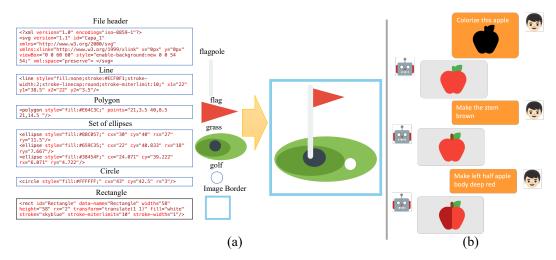


Figure 1: (a) The left example showcases an SVG representation, illustrating a golf course. Each geometric shape in the SVG code represents a distinct object or line within the two-dimensional graphics. For instance, the red polygon represents a flag in the graphical image. (b) On the right, we provide an example that shows that LLMs are able to understand and generate shape, color, and relationships between different elements in an interactive manner.

^{*}Equal Contribution. Order determined by random dice rolling.

1 Introduction

Large language models (LLMs) like ChatGPT [26] and GPT-4 [27] have gained prominence for their remarkable reasoning, in-context learning, and open-ended task abilities [8]. While large vision models (LVMs) [13, 24, 9] have also achieved impressive results in various tasks, they appear to exhibit fewer of these abilities.

As we delve deeper into the distinct reasoning, in-context learning, and open-ended task abilities of LLMs and LVMs, it is essential to recognize that both the task complexity and the data structure they handle play a crucial role in shaping their capabilities. LLMs like GPT-4 utilize a decoder-only architecture to handle diverse tasks, facilitated by the shared textual modality of both input and output. Conversely, vision tasks, with their pixel-based inputs and highly variable outputs—from labels and bounding boxes to segmentation masks, generated images, or textual captions—impose greater complexity and constraints on the capabilities of LVMs. Moreover, LLMs leverage the internet's diverse textual data and the inherent sequential structure of language to learn complex relationships and generate contextually relevant responses. In contrast, the continuous nature of visual data may also make it more challenging for vision models to discern complex patterns and relationships compared to the discrete structure of language data [5, 12, 22].

Recently, various vision language models [27, 33, 23] have been developed to capitalize on the impressive abilities of LLMs by integrating them with vision techniques. These efforts aim to bridge the gap between the two modalities by incorporating vision-specific components, such as the Vision Transformer [13] (ViT), which maps input images into embeddings or a set of continuous tokens. This approach has advanced vision-language understanding, but it relies on an additional visual encoder to convert images into a latent representation that the LLM can comprehend and align with text-based embeddings. Additionally, generative tasks may necessitate a visual decoder to revert these latent representations back into images.

An intriguing question that follows is whether we can harness the impressive abilities of large language models, such as reasoning, in-context learning, and open-ended task abilities, to tackle vision tasks without any visual components. That is, can we represent images using text-based descriptions that detail shapes, edges, and colors, to enable LLMs to directly and effectively understand and manipulate images?

Scalable Vector Graphics (SVG) [16] is a format for describing two-dimensional graphics in a manner that integrates with the web. Unlike raster images (like JPEG, PNG, or BMP) that are pixel-based, SVGs are defined in XML and use mathematical equations to depict shapes, curves, lines, and colors, as shown in Figure 1. One of the key advantages of using SVG over rasterized images is the potential to leverage large language models like GPT-4 for understanding and manipulating image content based on text prompts, opening up unique opportunities and applications beyond the traditional realm of vision models.

In this paper, we explore the potential of leveraging LLMs to perform a variety of vision tasks. Specifically, we convert raster images into SVG format and input these, along with tailored prompts, into LLMs. Our method is evaluated across discriminative and generative visual tasks, with promising initial results. In discriminative tasks, the SVG representations demonstrate shape-biased robustness against distribution shifts, notably outperforming raster-based methods on distribution shift benchmarks. Additionally, we enhance image classification by harnessing the in-context learning capabilities of LLMs, exceeding the performance of zero-shot learning. In generative tasks, our approach enables image generation and editing based on interactive, chat-based feedback, refining generated results to meet human expectations. Our method excels in visual prompting tasks on synthetic benchmarks, outperforming state-of-the-art methods by utilizing the strong reasoning capabilities of LLMs. Through the use of human-engineered prompts, we showcase the ability of LLMs to identify and execute transformations related to color, shape, style, and content within SVGs, generating credible outcomes. Lastly, we present preliminary results suggesting the potential of LLMs to perform complex visual tasks, like segmenting real images.

In summary, our work makes the following contributions:

• We propose a new approach to processing images with LLMs, converting raster images into SVG format for direct processing, thereby creating new possibilities for image understanding and manipulation.

- We demonstrate that SVG representation is more of a shape-biased format, which shows robustness to distribution shifts and notably surpasses raster-based methods on a distribution shift benchmark. Furthermore, we can enhance image classification performance by leveraging the in-context learning abilities of LLMs, surpassing the results of zero-shot learning.
- We attempt to achieve modification and generation of images with interactive chat-based feedback by modifying and generating SVG with LLMs. Moreover, we show LLMs' ability to identify and apply transformations between SVG pairs, recognize diverse transformations, and generate valid outcomes, suggesting their potential in understanding complex visual tasks, such as real image segmentation.

While our research demonstrates the potential of using SVG with LLMs, there are a couple of key limitations. Specifically, the standard SVG representation is not as effective in handling photographic content due to the loss of fine-grained details. Naively countering this by incorporating more details in the SVG representation can lead to a sequence length that is prohibitively long for current Transformer-based LLMs. Despite these limitations, we believe that our work offers promising initial results for the integration of LLMs and SVG in visual tasks. Addressing the aforementioned limitations could lead to more powerful image representation algorithms and pave the way for more versatile and comprehensive artificial intelligence systems.

2 Related Work

2.1 Scalable Vector Graphics

Vector graphics describe images as collections of parameterized shape primitives such as polygons, circles, and rectangles, rather than a regular raster grid of pixel values [28]. This representation is extensively supported by web browsers and can be rendered without any special software or plugins [3]. Primitives are usually characterized by a set of coordinates delineating their contour and the associated color. This leads to a compact and infinitely scalable representation where the appearance can be easily modified by adjusting stroke or color parameters. Consequently, vector graphics are the preferred choice among graphic artists and designers, as images maintain their sharpness regardless of the zoom level. Encapsulated PostScript (EPS) and Scalable Vector Graphics (SVG) are two notable vector-based formats [16].

SVG format stores images as XML-based text files that define geometrical objects and their properties [16], shown in Figure 1. This enables easy editing, manipulation, and embedding, which makes SVG particularly versatile for web applications and graphic design tasks [3]. EPS is another vector format for high-quality graphics that can be resized without losing quality [18]. In this paper, we employ large language models (LLMs) to understand images in the SVG format, achieving robust shape-color debiasing along with enhanced visual understanding and generation.

2.2 In-Context Learning

In-context learning seeks to enhance model performance by providing additional context during inference, typically in the form of several "question-answer" pairs. Although initially proven effective in natural language processing (NLP) [29], in-context learning has only recently been introduced to the vision domain. Flamingo [2] was the first to apply in-context learning to image and video understanding tasks like Question-Answering (QA). Subsequently, [5] investigated in-context generation using an MAE-VQGAN architecture.

Specifically, [5] pretrained a neural network to fill in missing patches of grid-like images to enable in-context learning for unseen tasks such as image segmentation. However, the authors sourced the pretraining dataset from a large set of figures in computer vision papers, most of which were hand-crafted and lacked clear connections with natural images. Our paper directly applies incontext learning by feeding SVG data into the LLM, demonstrating the potential for robust image understanding and generation capabilities without the need for a trained visual encoder.

2.3 Large Language Models

Large Language Models (LLMs) have attracted much attention in recent years due to their remarkable performance across numerous natural language processing tasks. GPT-3 [6], developed by OpenAI, is

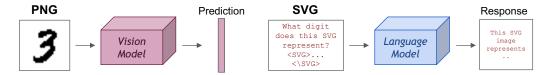


Figure 2: Architecture of our model. We illustrate the major difference between standard vision methods (left) and our method (right) in solving vision tasks.

a prime example of this category, boasting an immense scale of 175 billion parameters and human-like text generation capabilities. In a similar vein, BERT [11] (Bidirectional Encoder Representations from Transformers), introduced by Google, takes advantage of the transformer architecture and has substantially enhanced the state-of-the-art across various tasks by learning deep bidirectional representations. ChatGPT [26], another noteworthy model, is a GPT variant specifically designed for human-like conversational abilities. The most recent iteration, GPT-4 [27], succeeds GPT-3 [7] and carries on the LLM advancements in terms of scale and performance. These models lay the groundwork for our research, enabling us to investigate their potential in more complex tasks such as image processing and understanding. Our work effectively illustrates the applicability of LLMs to SVG-based image understanding and generation, paving the way for novel applications and research directions in the visual domain.

3 Tasks and Experiments

We first introduce the architecture, dataset, and implementation details in Section 3.1, 3.2, and 3.3. We then demonstrate LLMs have the capability to understand SVG representation for image recognition in Section 3.4, including zero-shot recognition, in-context learning, fine-tuning with LLMs, as well as robustness to distribution shift. Finally, we evaluate image generation and editing with interactive chat-based feedback by modifying and generating SVG with LLMs in Section 3.5.

3.1 Architecture

Figure 2 illustrates the major difference between our method and the vision methods in solving vision tasks. In particular, we convert raster images into the SVG format, and then input these SVG images, coupled with thoughtfully crafted prompts, into the LLMs to accomplish a diverse range of vision tasks.

3.2 Dataset

3.2.1 Human Designed SVG Dataset

We collect a dataset from the public collection of SVG images.² Specifically, we collect the digits and icons to demonstrate image recognition and generation capabilities. Examples are shown in Figure 3 (a) and (b).

3.2.2 Convert Raster Images to SVG

Directly convert using curve tracing. Given the rich set of natural images in raster format, we utilize the curve tracing algorithm to convert RGB images into the SVG format.³ Specifically, we convert MNIST [10] to SVG format using this approach, shown in Figure 3 (c).

Use segmentation prior knowledge for conversion. An image typically contains both high-level features such as object shape and low-level features such as texture. However, directly converting natural images into SVG format often leads to excessive attention to fine low-level details, resulting in very long SVG files. Therefore, biasing the representation on object shape can help build a more efficient SVG format. We use a universal segmentation model, Segment Anything Model (SAM) [21], to convert the image into a set of masks, which leads to a high-level abstraction of the scene.

 $^{^2}$ https://www.svgrepo.com/, https://www.kaggle.com/datasets/victorcondino/svgicons

³https://github.com/visioncortex/vtracer

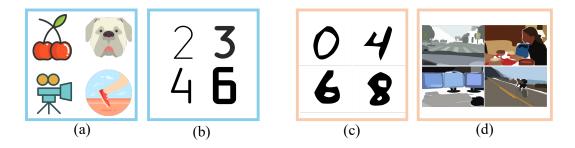


Figure 3: Visualization of our datasets. (a) and (b) are human-designed SVG vectors and icons. (c) and (d) are converted from raster images. Specifically, (c) is generated using curve tracing from MNIST [10], while (d) is generated using SAM [21] and curve tracing sequentially.

Table 1: Image classification results using PNG and SVG format. CMNIST denotes the Colored-MNIST dataset. ICL denotes in-context learning. We use a Mini-MNIST dataset (100 images) to evaluate both zero-shot and in-context learning using the GPT4 API. ResNet and Vicuna are trained on the 60k black and white training set, and evaluated on the 10k test set.

Method	ResNet18	Zero-Shot	ICL (#1)	ICL (#3)	ICL (#10)	Fine-tuning (Vicuna)
Image Format	PNG	SVG	SVG	SVG	SVG	SVG
MNIST	97.85%	20%	24%	26%	28%	97.24%
CMNIST-(A)	72.48%	16%	19%	23%	26%	95.69%
CMNIST-(B)	29.59%	13%	20%	20%	23%	92.88%

Specifically, we apply SAM to PASCAL-VOC dataset [15], then extract 20 masks with the largest area, where the color of each mask is the average pixel value within the mask, shown in Figure 3 (d).

3.3 Implementation Details

The strong language understanding and reasoning capability of the LLM is the key to the success of SVG understanding. For zero-shot image recognition and generation, we adopt GPT-4 [27]. In addition, we also fine-tune Vicuna-7B [30] using the MNIST training set, and then evaluate classification performance on the test set.

3.4 Image Recognition

We first conduct image recognition under various settings, which demonstrate the effectiveness of using LLM for SVG understanding. Due to resource (i.e., frequency and rate) constraints, we use a Mini-MNIST, which has 10 samples per class (i.e., overall 100 samples), to test the performance for zero-shot and in-context learning using GPT-4 [27]. For the fine-tuned Vicuna, we use the whole test set (\sim 10k images).

Zero-shot image classification. Due to the natural textual representation of SVG, we can directly embed SVG within the prompt. For example, we can conduct zero-shot image classification by prompting What semantic category does this SVG image belong to? <SVG>, where <SVG> denotes the actual SVG code embedded within the prompt. We directly feed the prompt along with the query SVG to GPT-4 to get the zero-shot accuracy on MNIST. Shown in Table 1, we get 20% accuracy, meaning that GPT-4 owns a weak image prior to the SVG format.

In-context learning for image classification. Though LLM can solve the text understanding tasks in a zero-shot manner, recent studies reveal that in-context learning can effectively boost performance [6], where a list of input-output examples can significantly improve accuracy. However, it is still hard to conduct in-context learning for visual representations, especially for image generation [5]. On the contrary, in-context learning can be naturally conducted using SVG due to its textual representation.

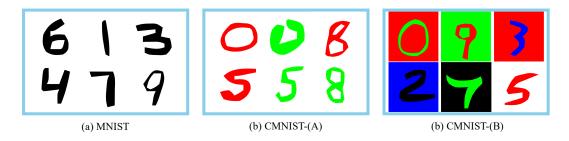


Figure 4: Visualization of our MNIST datasets. (a) Vanilla MNIST dataset. (b) Colored-MNIST-(A) dataset. (c) Colored-MNIST-(B) dataset.

We follow the typical settings for in-context learning, where the prompt is composed of the task instruction, a list of question-answer (a.k.a, input-output) pairs, and then the final question [32, 25]. For example, for MNIST [10] classification, given 2 in-context pairs, the exemplar prompt can be "Instruction: Please predict the digit label for each of the following SVG images. <Q1>:<A1>; <Q2>:<A2>; <Q3>:". Here <Q#>, denotes the SVG code, <A*> denotes the digit label. As a result, the LLM can give a more accurate answer based on the given input-output pairs.

We consider three representative cases here: (i) Randomly choose 1 class different from the class of the query sample, then randomly choose 1 sample from this class, (ii) Similar to the above case, but choose 3 classes (all classes but the query class), and (iii) Randomly choose a sample for all 10 classes. Shown in Table 1, even with 1 in-context sample, the model shows a 4% accuracy improvement compared to zero-shot classification, demonstrating the capability of LLM to learn the visual concepts within the context. With 1 sample per class, the test accuracy climbs to 28%, which is 8% higher than the zero-shot classification performance.

Fine-tuning Vicuna. The zero-shot and few-shot experiments demonstrate the existence of the visual understanding capability of LLM. However, its visual reasoning capability could be limited by the pretraining data, since SVG data may not be extensive in the pretraining stage of LLM. In order to explore the limit of the visual understanding capability of LLM, we fine-tune Vicuna on the training set of MNIST (60k images) as the instruction tuning dataset, where the SVG data is the query from the "human", and the label is the response of the "Assistant". After training for 3 epochs, we evaluate the MNIST test accuracy. Shown in Table 1, fine-tuned Vicuna achieves 97.24%, which is comparable to the 97.85% achieved by ResNet18 trained on the PNG version for 10 epochs. Our image recognition results validate the potential of understanding images via LLM using SVG, which could be further boosted by incorporating more supervised/unsupervised data into the pretraining stage.

Generalization: Colored-MNIST Classification One desirable property of SVG is its shape-color disentanglement, where shape and color attributes are encoded at different blocks in SVG. Therefore, SVG can naturally be better at debiasing color from shape. In contrast, convolutional neural networks (CNN) are known to be biased toward color for image recognition tasks [17]. Here we create two Colored-MNIST datasets to evaluate the strong shape-color debiasing ability of SVG. (A) We randomly color the foreground into either red or green with 50% probability. Furthermore, we propose a stronger Colored-MNIST dataset: (B) For each background and foreground, the color is chosen randomly from black, white, red, blue, and green, but the background and foreground colors need to be different, i.e., there are 20 combinations. The visualizations of Colored-MNIST datasets are shown in Figure 4. After training on the normal black-and-white MNIST training set, we evaluate ResNet18 and Vicuna on the Colored-MNIST (A) and (B). As Table 1 shows, ResNet18 achieves a much lower accuracy of 72.48% and 29.59% for each case. On the other hand, the performance of our model is much stronger, showing strong robustness. We believe that with advanced image discretization techniques such as using SAM [21] prior knowledge, the advantage of utilizing a discrete representation of images could be magnified even further.

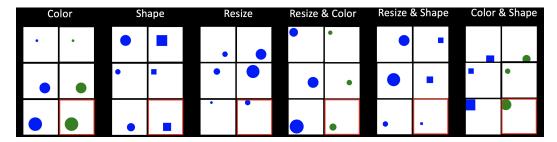


Figure 5: Synthetic data study results. The generation results of our method are annotated with a red square.

Table 2: Synthetic data study results. We report the color-aware mIOU on the six tasks [5]. Our method surpasses the best SOTA competitors significantly. It demonstrates that GPT4 is able to understand and reason shape, color, and size transformation using SVG representation.

Method	Color	Shape	Size	Color Shape	Color Size	Shape Size
VQGAN [14]	7.0	19.1	16.2	7.4	2.2	18.4
BEiT [4]	40.9	31.4	7.1	33.1	21.2	13.0
MAE [19]	70.2	44.0	34.7	19.3	19.0	46.0
MAE-VQGAN [5]	40.4	46.5	42.0	20.4	18.3	40.3
Ours (SVG with GPT4)	100.0	92.6	100.0	92.6	100.0	86.5

3.5 Image Generation

In this section, we aim to demonstrate that LLMs can understand and generate SVG code, allowing them to create new graphics or modify existing ones based on text prompts, without pixel-level manipulations.

3.5.1 Synthetic Data Study

To evaluate the capabilities of LLMs towards understanding SVG, we follow [5] to create a set of 3 simple synthetic tasks and 3 of their combinations, and evaluate each model on 100 examples per task.

Tasks and Evaluation. Every pair in our example set includes an SVG showcasing a colored shape along with a corresponding SVG with specific transformations. The transformations consist of color, size, or a combination of these aspects. We delve into a more detailed description of each task in the appendix. For evaluation purposes, we adopt the method from [5], for measuring and reporting the color-aware mean Intersection over Union (mIOU).

Prompt. Given two example pairs and a query SVG, we structure the text prompt in the same fashion for all tasks. The prompt is designed to figure out the common transformation in the two examples first and then transform that query into the corresponding key SVG code. We include the prompt details in the appendix.

Qualitative and quantitative results. The results are presented in Table 2. See Figure 5 for our generated results. Our method outperforms the SOTA across six tasks by a clear margin. We believe that GPT4 can clearly understand simple shape, color, and size transformations by analyzing the SVG code without any pixel-level information.

3.5.2 Style and Content Extrapolation

In this section, we assess if LLMs can extrapolate SVG codes with more challenging transformations, such as content and style.

Style generation: We present LLMs with sample SVG letters. The first task is to figure out the style in the given examples. Then, given a new test query, the second task is to transform this given query

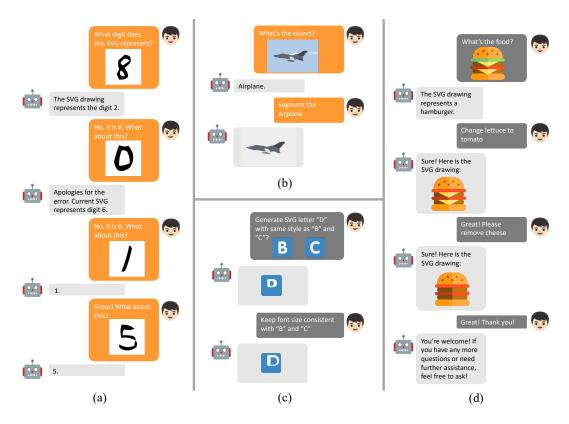


Figure 6: In-context learning and image generation capabilities of SVG with LLMs. (a) With human feedback, LLM gradually performs better on digit classification. (b) LLM powers SVG with the capability of image recognition and referring segmentation. (c) With human feedback, the content generation performance becomes better. (d) LLM can recognize and manipulate specific parts of the hamburger, such as removing or replacing them.

so that it adheres to the same stylistic conventions as the example letters. We show some qualitative results in Figure 6. More results can be found in the appendix.

Content generation: LLMs are shown two examples of SVG code pairs. Each pair consists of a query and key pair (both are numbers), where the query describes an SVG code of a number, and the key describes the SVG code of another number with an introduced mathematical operation. The operation can consist of add, subtract, multiply, and divide. The mathematical operation should be held in both example pairs. The first task is to figure out the mathematical operation in the two examples. Then, given a new test query SVG number, the second task is to identify what number it is and follow the mathematical operation discovered to generate the corresponding test key number. We include qualitative results in Figure 7. The prompt details can be found in the appendix.

3.5.3 Referring Segmentation

The objective of the task is to label pixels in an image or video that correspond to an object instance referred by a linguistic expression. SVG representation has two advantages. First, language instruction is naturally embedded within the prompt, thus a separate design of the image segmentation model is not needed. Second, a large corpus of text and programming languages including XML are seen during pretraining, benefiting the vision-language understanding ability.

SVG is typically composed of several colored polygons, where each of them can correspond to a part of the object. Therefore, we can use the referring segmentation instructions to guide the LLM in finding the corresponding SVG code. Shown in Figure 6 (b) and (d), LLM can localize the object decently well. In (b), the majority of the airplane is selected as foreground, while in (d), not only is the lettuce recognized, but also the two pieces of cheese are localized and subsequently removed.

Query	Key										
12	24	12	24	60	36	60	36	21	14	10	15
6	12	6	18	25	1	25	15	12	8	50	25
30	60	30	42	40	16	40	24		5		30

Figure 7: Understanding SVG content through the lens of GPT-4: GPT-4 demonstrates its ability to generate accurate content by analyzing the correlation between provided example number pairs, and subsequently applying this relationship to ascertain the corresponding test key number. Remarkably, in scenarios where the relationship exhibits ambiguity, GPT-4 showcases its proficiency in identifying multiple possible interpretations.

4 Discussion

While our research demonstrates the potential of using Scalable Vector Graphics (SVG) with large language models (LLMs) to tackle visual tasks without a parameterized visual encoder, the major limitation of SVG representation is the loss of fine details: Though our method of converting raster images into SVG format and leveraging XML-based textual descriptions allows for efficient processing of crisp graphics and designs, it is not as effective in handling photographic content. As a result, fine-grained details, such as image textures, may be lost during conversion. Conversely, when the SVG code incorporates an excessive level of detail, its sequence length can become prohibitively long, which can pose challenges for the training and inference of current Transformer-based LLMs. Developing hybrid representations that can retain the advantages of both discrete and continuous data, while preserving finer details, is an area for future exploration. For example, in LLMs, the processing unit is the token, which can correspond to one or several words. However, in SVG, we would prefer to have a specific embedding module for each geometric primitive in SVG, such as circles, polygons, and so on.

In summary, while our approach presents limitations, it offers promising initial results for the integration of LLMs and SVG in visual tasks. Addressing these limitations could lead to more powerful image representation algorithms and pave the way for more versatile and comprehensive artificial intelligence systems.

5 Conclusion

This paper explored the possibility of enabling large language models (LLMs) to "see" and process images through the Scalable Vector Graphics (SVG) format. By converting raster images into SVG representations and leveraging XML-based textual descriptions, we showed that LLMs can directly understand and manipulate images without a parameterized visual encoder. Our method leverages the intrinsic abilities of LLMs, thus eliminating the need for specialized vision encoders in the processing of images.

We demonstrated the efficacy of our proposed approach across discriminative and generative tasks, revealing the underlying shape-color disentanglement nature of SVG. Through these experiments, we showed that our method can improve image classification performance by directly exploiting the in-context learning capabilities of LLMs.

This research can open the door to new opportunities in the realm of computer vision by integrating the powerful capabilities of LLMs with SVG format. We believe that our work provides an initial exploratory step for future research in the integration of LLMs and SVG for the development of advanced image representation formats and more complex vision tasks. As we continue to explore the potential of large language models on visual input, this approach could inspire further progress in the understanding of visual data with multi-modal fusion approaches.

Appendix

A Experiment Details

A.1 Raster Images to SVG Conversion

One of the most fundamental pieces of information for visual perception is object shape. Our method can be conceptualized as selectively diminishing details from an image, prioritizing the extraction of less significant shapes. This guided process of reduction offers a quantitative way to manage the amount of visual data present within an image. Within this framework, we perceive segmentation as an example of extreme simplification, whereas vectorization serves as a more moderate form of the same. Here we introduce how we use such two approaches to convert the raster images to SVG.

Image Vectorization. The vector tracing algorithm operates in a sequential three-step process. Initially, pixels are transformed into a defined path. Subsequently, this path is condensed into a simplified polygonal representation. Lastly, the polygon is refined and approximated using a curve-fitting (tracing) technique, which enhances its smoothness.

There are several online tools to convert the raster images (jpg and png) into vector graphics (SVG), such as Adobe Illustrator [1], Inkscape [20], and VTracer [31]. We experiment with all of them and found that VTracer leads to the best balance between SVG complexity (code length) and rich semantic representation.

In MNIST [10], we use the default hyperparameters during conversion. Specifically, we (i) first binarize the MNIST pixel value from the continuous range [0, 255] to the binary set $\{0, 255\}$ using the threshold 127.5, (ii) set the foreground to black, and the background to white, and (iii) apply the vector tracing algorithm VTracer.

Segmentation Prior. As mentioned earlier, segmentation can provide a strong prior for object shape. We want a generalist model that can segment any image, i.e., not trained and thus biased towards a certain dataset. The Segment Anything (SA) [21] project introduces such an image segmentation model, the Segment Anything Model (SAM), and a large-scale dataset, SA-1B, with the aim of achieving powerful generalization and zero-shot transfer across diverse segmentation tasks, demonstrating competitive results often surpassing prior fully supervised methods. We use the default hyper-parameters of SAM to segment the whole image into a set of masks without class labels, where the color of each mask is represented by the average value of the pixels within the mask. Specifically, we sample 32 query points per side (1024 points overall) to generate the image mask. Then we select the top 20 masks with the highest area as the final representation for an image.

We then use VTracer to transform the mask into SVG format. Note that, to reduce the final SVG, we adjust several settings: we set the number of significant bits to use in an RGB channel to 0; we set the minimum angle displacement degree to splice a spline to 90; we set the color difference between gradient layers to be 35; we consider a corner to have a minimum momentary angle of 0 degrees; we discard patches smaller than 16 pixels in size; and we perform iterative subdivide smoothing until all segments are shorter than 10 pixels.

A.2 Fine-tuning Dataset for Vicuna

We use the same JSON format in Vicuna [30] to construct the fine-tuning dataset. We use all the training samples in MNIST, translating to 60,000 SVG images. For each sample, we construct one round of conversation: (i) From human: "Which digit does the following SVG reflect? <SVG code here">, and (ii) From GPT: "<label">. Here <label> denotes the digit label, which ranges from 0 to 9. Then we use this dataset to fine-tune Vicuna using the default hyper-parameters for 3 epochs.

A.3 Prompt Engineering

In this section, we provide the details of prompt engineering for each task. The prompt is designed to figure out the common transformation in the SVG example pairs first (each example pair consists of

⁴https://github.com/lm-sys/FastChat

a query and a key) and then transform the query into the corresponding key SVG by following the discovered common transformation.

In-context Image Classification. In this task, in-context examples are aimed to provide more context information using several image-label pairs, thus facilitating the final classification. The specific prompt utilized for this purpose using 3 in-context examples is detailed below: "Instruction: please predict the digit number for each of the following SVG images. Please think step by step, and closely look at the key identifying image characteristics. Please just tell me the image class, no other information is needed. Q: What digit does this SVG image represent? <SVG code here> A: This SVG image represents digit <label> Q: What digit does this SVG image represents digit <label> Q: What digit does this SVG image represent? <SVG code here> A: This SVG image represents digit <label> Q: What digit does this SVG image represents digit <label> Q: What digit does this SVG image represents digit <label> Q: What digit does this SVG image represents digit .

Synthetic Data Study: In this task, the objective is to conduct an analytical evaluation of the provided two example pairs, examining changes that occur in aspects such as color, shape, and size. The insight gathered from this analysis will then be used to adapt the given query into its corresponding key. The specific prompt utilized for this purpose is detailed below: "Please perform the following task carefully. In this task, you will be shown two examples of Scalable Vector Graphics (SVG) code pairs. Each pair consists of a query and key pair, where the query describes the SVG code of the original image, and the key describes the SVG code of the transformed image. Each will be named "Example Query #" and "Example Key #" respectively. Your first task is to figure out the common transformation in the two examples. The transformation can consist of color, shape, size, or any combination thereof. Then, given a new test query SVG code (named "Test Query"), your second task is to transform that query into the corresponding key SVG code (named "Test Key"), following the common transformation that you discovered in the two example pairs. Here are the two example query and key pairs: Example Query 1: <SVG code here>; Example Key 1:<SVG code here>; Example Query 2:<SVG code here>; Example Key 2:<SVG code here>; Here are the test query and key pair: Test Query: <SVG code here>; Test Key:"

Content Extrapolation: In this task, LLMs are presented with SVG code pairs, each containing a query-key set that depicts numbers. The key introduces a consistent mathematical operation (addition, subtraction, multiplication, or division) to the query number. The tasks are to identify this operation in the examples and apply it to new test queries to generate corresponding test keys. To facilitate a more comprehensive understanding of SVG number codes for the LLM, we initially present the SVG codes for numbers 0 through 9 to the LLM prior to posing specific queries. The specific prompt utilized for this purpose is detailed below: "Please perform the following task carefully. In this task, you will be shown two examples of Scalable Vector Graphics (SVG) code pairs. Each pair consists of a query and key pair, where the query describes an SVG code of an integer number, and the key describes the SVG code of another integer number with an introduced mathematical operation. Each will be named "Example Query #" and "Example Key #" respectively. In addition to the example pairs, you will be shown a new test query SVG code (named "Test Query''). Your first task is to identify which number each example query, example key, and test query represents. Your second task is to figure out all the possible mathematical operations that are held for all given example pairs. The operation could be add, subtract, multiply, and divide (the subtract or multiply factor could be a fraction). Then, according to the numbers of example pairs and test query you identified, your third task is to predict the corresponding test key number (named "Test Key"), following all the mathematical operations that you discovered in the given example pairs. Finally, you need to generate the corresponding SVG code of the test key number. Here are the two example query and key pairs:

Example Query 1: <SVG code here>; Example Key 1:<SVG code here>; Example Query 2:<SVG code here>; Example Key 2:<SVG code here>; Here are the test query and key pair: Test Query: <SVG code here>; Test Key: (Note: think about four operations one by one, and the operation should be consistent for all given example pairs)"

B More Chat Results

Image Recognition and Manipulation. In this section, we provide more examples for chatbased image recognition and manipulation using GPT4 [27]. The qualitative results are shown in Figure 8: (a) SVG representation empowers robust in-context digit recognition capability given different background and foreground colors, (b) GPT4 can recognize and depict the details of a dog with the prompt: "a stylized bear or a similar mammal with a round face and ears." Furthermore, GPT-4 can identify the location of the dog's left eye and remove it. (c) GPT4 is capable of recognizing a natural image from the CIFAR-10 dataset.

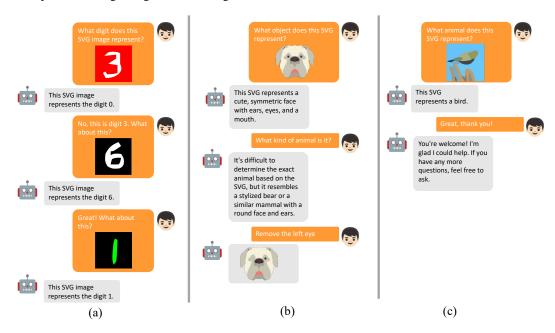


Figure 8: More qualitative results of chat-based image recognition and manipulation. (a) In-context digit recognition in Colored-MNIST-(B). (b) GPT can explain and manipulate the dog SVG image. (c) GPT4 can also recognize the bird from a CIFAR-10 example.

Style Extrapolation: LLMs are provided with five example pairs and are tasked with deciphering the stylistic attributes inherent in these examples. Following this, a new test query is presented to the LLMs. Their objective is to modify this query into the corresponding key, ensuring that it aligns with the same stylistic principles showcased in the example pairs. The specific prompt utilized for this purpose is detailed below: "Please perform the following task carefully. In this task, you will be shown five examples of Scalable Vector Graphics (SVG) code pairs. Each pair consists of a query and key pair (both are English letter), where the query describes the SVG code of the original image, and the key describes the SVG code of the transformed image. Each will be named "Example Query #" and "Example Key #" respectively. Your first task is to figure out the common transformation in the five examples. The transformation can consist of color, shape, size, style, font, and background changes, or any combination thereof. Even though you cannot see the images, and only their SVG codes, you need to discover the transformations that are happening at the image level and not just at the code level. Be detailed, and try to discover every change, and

the most important change is that the paths in the SVG code between each query and key is different due to the common transformation but the shapes of the letters that query and key are representing remain the same. Then, given a new test query SVG code (named "Test Query"), your second task is to transform that query into the corresponding key SVG code (named "Test Key''), following the common transformation that you discovered in the five example pairs. To help you better understand the transformation, I will also inform you of what letter each query and key represent. You need to find the shape of each query and key by analyzing their path. Here are the five example query and key pairs: Example Query 1 (letter B):; Example Key 1 (letter B):<SVG code here>; Example Query 2 (letter R):<SVG code here>; Example Key 2 (letter R):<SVG code here>; Example Query 3 (letter Z): <SVG code here>; Example Key 3 (letter Z): <SVG code here>; Example Query 4 (letter E): <SVG code here>; Example Key 4 (letter E): <SVG code here>; Example Query 5 (letter N): <SVG code here>; Example Key 5 (letter N): <SVG code here>; Here is the test query and key pair: Test Query (letter #):; Test Key: " The qualitative results are shown in figure 9.

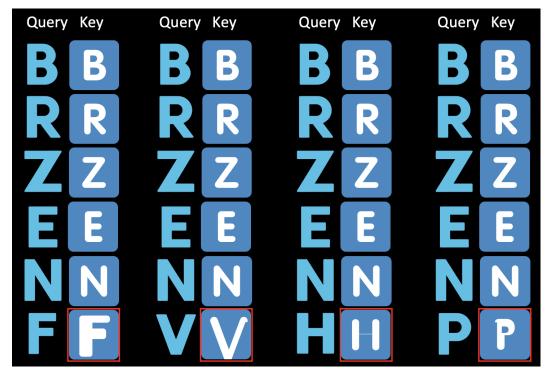


Figure 9: More qualitative results of style extrapolation. The generation results of our method are annotated with a red square.

References

- [1] Adobe Inc. Adobe illustrator. https://adobe.com/products/illustrator, 2019.
- [2] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *NeurIPS*, 35:23716–23736, 2022.
- [3] Greg J Badros, Jojada J Tirtowidjojo, Kim Marriott, Bernd Meyer, Will Portnoy, and Alan Borning. A constraint extension to scalable vector graphics. In *WWW*, pages 489–498, 2001.
- [4] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. BEit: BERT pre-training of image transformers. In *International Conference on Learning Representations*, 2022.
- [5] Amir Bar, Yossi Gandelsman, Trevor Darrell, Amir Globerson, and Alexei Efros. Visual prompting via image inpainting. *NeurIPS*, 35:25005–25017, 2022.
- [6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In NeurIPS, 2020.
- [7] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *NeurIPS*, 33:1877–1901, 2020.
- [8] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.
- [9] Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, et al. Scaling vision transformers to 22 billion parameters. *arXiv preprint arXiv:2302.05442*, 2023.
- [10] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.
- [11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.
- [12] Anuj Diwan, Layne Berry, Eunsol Choi, David Harwath, and Kyle Mahowald. Why is winoground hard? investigating failures in visuolinguistic compositionality. *arXiv* preprint arXiv:2211.00768, 2022.
- [13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.
- [14] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12873–12883, 2021.
- [15] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, January 2015.
- [16] Jon Ferraiolo, Fujisawa Jun, and Dean Jackson. *Scalable vector graphics (SVG) 1.0 specification*. iuniverse Bloomington, 2000.
- [17] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *ICLR*, 2019.
- [18] Andreas R Gruber, Ronny Lorenz, Stephan H Bernhart, Richard Neuböck, and Ivo L Hofacker. The vienna rna websuite. *Nucleic acids research*, 36(suppl_2):W70–W74, 2008.
- [19] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009, 2022.

- [20] Inkscape Project. Inkscape. https://inkscape.org, 2020.
- [21] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. *arXiv* preprint arXiv:2304.02643, 2023.
- [22] Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and brain sciences*, 40:e253, 2017.
- [23] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *arXiv*:2304.08485, 2023.
- [24] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11976–11986, 2022.
- [25] Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? EMNLP, 2022.
- [26] OpenAI. Chatgpt. https://openai.com/blog/chatgpt/, 2023.
- [27] OpenAI. Gpt-4 technical report. 2023.
- [28] Zhong-Ren Peng and Chuanrong Zhang. The roles of geography markup language (gml), scalable vector graphics (svg), and web feature service (wfs) specifications in the development of internet geographic information systems (gis). *Journal of Geographical Systems*, 6:95–116, 2004.
- [29] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.
- [30] Vicuna. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. https://vicuna.lmsys.org/, 2023.
- [31] VTracer. Vtracer. https://www.visioncortex.org/vtracer-docs, 2020.
- [32] Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit bayesian inference. *ICLR*, 2022.
- [33] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*, 2023.