StarVector: Generating Scalable Vector Graphics Code from Images and Text

Juan A. Rodriguez[1,2,4] Abhay Puri[1] Shubham Agarwal[1,2] Issam H. Laradji[1,5] Pau Rodriguez[6*]
Sai Rajeswar[1,2] David Vazquez[1] Christopher Pal[1,2,3] Marco Pedersoli[4]

[1]ServiceNow Research [2]Mila - Quebec AI Institute [3]Canada CIFAR AI Chair [4]ÉTS, Montréal, Canada
[5]UBC, Vancouver, Canada [6]Apple MLR, Barcelona, Spain [*] External collaboration

https://starvector.github.io/

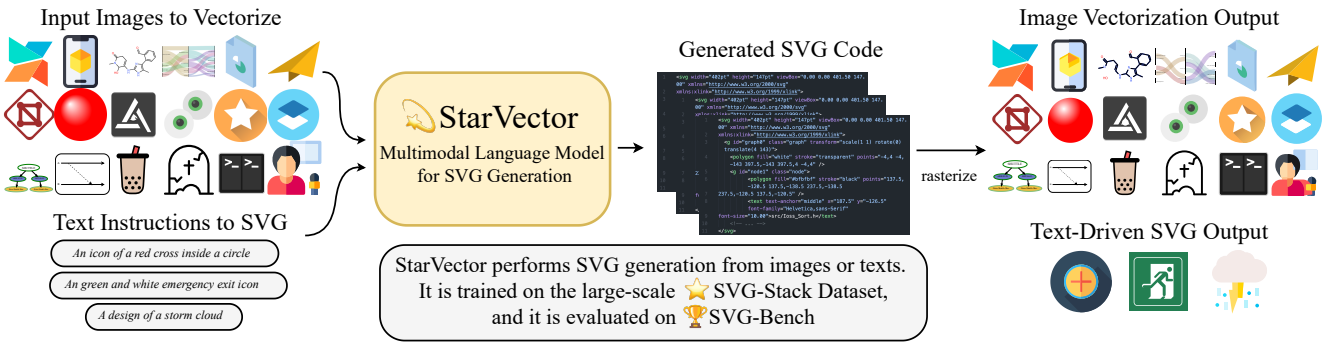

Figure 1. **StarVector: A foundation model for SVG generation.** StarVector's multimodal architecture allows input from raster images or text instructions. It converts a variety of raster visuals, including icons, logos, and technical diagrams, into vector graphics or generates new SVGs from text. (Left) Inputs: raster images and text. (Right) Outputs: vectorized images (SVG)

Abstract

Scalable Vector Graphics (SVGs) are vital for modern image rendering due to their scalability and versatility. Previous SVG generation methods have focused on curve-based vectorization, lacking semantic understanding, often producing artifacts, and struggling with SVG primitives beyond path curves. To address these issues, we introduce StarVector, a multimodal large language model for SVG generation. It performs image vectorization by understanding image semantics and using SVG primitives for compact, precise outputs. Unlike traditional methods, StarVector works directly in the SVG code space, leveraging visual understanding to apply accurate SVG primitives. To train StarVector, we create SVG-Stack, a diverse dataset of 2M samples that enables generalization across vectorization tasks and precise use of primitives like ellipses, polygons, and text. We address challenges in SVG evaluation, showing that pixel-based metrics like MSE fail to capture the unique qualities of vector graphics. We introduce SVG-Bench, a benchmark across 10 datasets, and 3 tasks: Image-to-SVG, Text-to-SVG generation, and diagram generation. Using this setup, StarVector achieves state-of-the-art performance, producing more compact and semantically rich SVGs.

1. Introduction

Vector graphics represent an archetypal form of image representation, where visual compositions are constituted by scalable primitive shapes [33, 43, 50, 50]. For modern image rendering, Scalable Vector Graphics (SVGs) [60] have become the standard for representing vector graphics. The SVG format [25] provides a comprehensive set of primitives and styling options. At its core, the *path* represents basic curves [60]. Combined with primitives like *polygon* or *ellipse*, SVGs define complex designs precisely.

The task of *image vectorization*, i.e., converting pixel-based raster images into SVGs, stands as a fundamental challenge in vector graphics. The main challenge lies in developing methods that generalize across diverse domains, from fonts and logos to complex illustrations and diagrams [7, 8, 69, 70]. Traditional approaches often rely on approximating images through multiple *paths* [43, 50, 51, 59, 87]. This strategy can be inefficient as shown in Fig. 2 (Right). For instance, a circle shape could be represented as long *path* or, more precisely and compactly, as a single `<circle/>` primitive. Similarly, text elements should

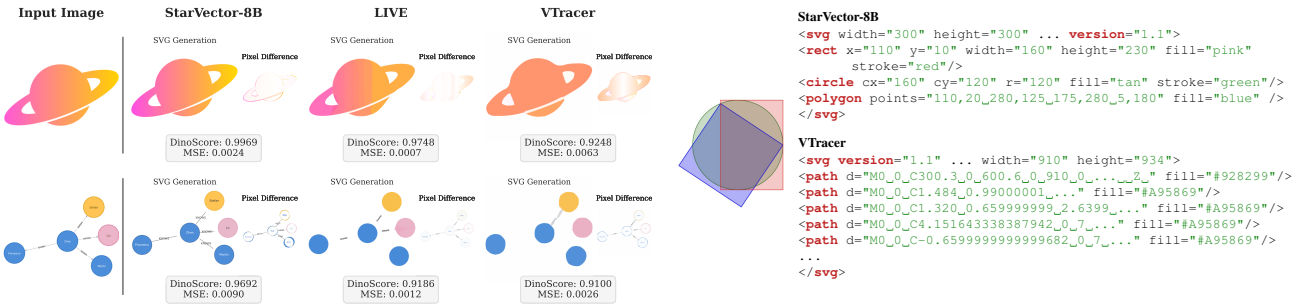

Figure 2. **(Left) Image Vectorization** results using StarVector-8B, LIVE, and VTracer. Each row shows the input image, generated SVGs, and pixel-wise difference maps to highlight accuracy. StarVector-8B better preserves shapes, color gradients, and text, despite minor misplacements. Notably, *MSE often misaligns with visual quality*, e.g., regarding the 'planet' example, StarVector's MSE (0.009) is higher than LIVE's (0.0012) and VTracer's (0.0039), yet StarVector preserves the color gradient. For the 'diagram' example, StarVector preserves the text. DinoScore better reflects these details, consistently favoring StarVector. **(Right) Curve vs Primitive-based Vectorization.** SVG code generated by StarVector and VTracer for the given image. StarVector effectively leverages shape primitives, resulting in a compact and precise vectorization. VTracer decomposes the image into numerous paths, resulting in a more complex result with less semantic clarity.

be vectorized as editable `<text/>` primitives to retain the original textual content. This balance between curve-based shape approximation and accurately recognizing primitives has been previously unexplored and remains a core challenge in modern vectorization.

Previous vectorization approaches fall into two main categories: traditional image processing methods and deep learning (DL)-based. Image processing methods [51, 59, 87] employ pixel-level analysis to trace vector curves, but often produce overly complex representations with artifacts and lack semantic understanding (Figure 2). While DL approaches [13, 14, 67, 89] have advanced vector graphics modeling through latent variable models and differentiable rendering [43, 50], they typically struggle with generalization beyond specific domains and underutilize SVG primitives. This limits their effectiveness for complex SVGs like scientific diagrams and precludes their use in modern multimodal tasks such as text-driven SVG generation [24, 65, 72, 96].

Recent advancements in Multimodal Large Language Models (MLLMs) [2, 45] have integrated visual understanding into transformer [84] architectures while demonstrating strong code generation capabilities [1, 42, 49, 55]. Building on these developments, *we introduce image vectorization as an inverse rendering and code generation task*, leveraging MLLMs to generate SVG code directly from input images. This approach naturally encompasses the full range of SVG primitives, enhancing both semantic understanding and generation capabilities (Table 6).

We introduce 💫 StarVector, a foundational MLLM for SVG generation. StarVector processes both images and text instructions to produce compilable SVG code, leveraging SVG primitives to accurately represent vector graphics. We build upon the **Star**Coder works [42, 49] to connect the code generation research with SVG generation. Figure 3 describes the model architecture. It integrates an image encoder that projects images into visual tokens, and a transformer language model for learning the relationships between instructions, visual features, and SVG code sequences, to perform image vectorization (Image-to-SVG) or text-driven SVG Generation (Text-to-SVG) tasks. StarVector, performs primitive-aware vectorization through learned semantic understanding, effectively leveraging SVG primitives without explicit pixel reconstruction objectives. To address the lack of large-scale SVG datasets for training StarVector, we introduce ⭐ SVG-Stack, containing over 2M SVG samples paired with rendered images and textual descriptions.

Additionally, we find that conventional metrics like MSE fail to adequately assess vector graphics fidelity, as demonstrated in Figure 2. Instead, we propose DinoScore, a perceptual similarity metric that better correlates with human perception of visual quality, and introduce 🏆 SVG-Bench, a comprehensive evaluation framework spanning 10 datasets and 3 tasks: Image-to-SVG, Text-to-SVG, and diagram generation.

Contributions

1. We introduce 💫 **StarVector**, an MLLM capable of image vectorization and text-driven SVG Generation, uniquely preserving SVG primitives rather than producing multiple curves—a previously unexplored skill.
2. We create ⭐ **SVG-Stack**, a large-scale dataset with 2M samples, supporting Image-to-SVG and Text-to-SVG.
3. We develop 🏆 **SVG-Bench**, an MLLM benchmark with 10 datasets across 3 SVG tasks.
4. We conduct extensive experiments and evaluations, including human assessments, demonstrating StarVector's