The Academic College of Tel Aviv-Yaffo

The Faculty of the School of Computer Science

Exploring Multimodal Large Language Models for High-Quality Image-to-SVG Conversion

November 2025

Thesis submitted in partial fulfilment of the requirements for the M.Sc. degree in the School of Computer Science of the Academic College of Tel Aviv-Yaffo

By

Shachar Levy

The research work for the thesis has been carried out under the supervision of

Dr. Sarel Cohen

Contents

Abstract

Scalable Vector Graphics (SVGs) are essential for modern digital design due to their resolution independence and editability, however, automatically converting complex raster images into high-quality vector representations remains a significant challenge. Existing state-of-the-art multimodal models, such as GPT-4 [13] and Claude [14], often struggle to produce structurally accurate SVGs, resulting in abstract or distorted outputs. This thesis explores the potential of fine-tuning Vision Language Models (VLMs) to bridge this gap and generate high-fidelity SVGs from raster inputs.

To achieve this, we developed a comprehensive pipeline for generating high-quality vector training data. We constructed a massive dataset scaling from a curated set of 2.5k examples to approximately 1.8 million triplets of textual descriptions, raster images, and their corresponding SVG files. This pipeline leveraged several generative models and algorithms, including DALL-E, Stable Diffusion fine-tuned with DreamBooth [21], and Flux [22] for image generation, alongside Potrace [11] for converting raster images into SVG format. We also employed advanced techniques such as "offset noise" to ensure uniform white backgrounds and GPT-4 based caption simplification to create clean, vector-friendly silhouettes suitable for training.

For fine-tuning the VLM, we adapted the LLaVA [23] architecture using Low-Rank Adaptation (LoRA) [25] and gradient accumulation to handle memory constraints. Fine-tuning on our generated Flux dataset faced challenges due to SVG sequences frequently exceeding the model's context window. However, the OmniSVG paper [24] was published during the course of our work, presenting a unified approach for SVG generation. Utilizing the icon subset from the OmniSVG dataset, we successfully fine-tuned the model. Quantitative evaluation demonstrated that the fine-tuned model achieved substantially higher semantic alignment and pixel-level fidelity compared to the base model (the detailed metrics and numerical results are described in the experiments section).

Preliminary results of our work were presented in a poster at the 18th ACM International Systems and Storage Conference (SYSTOR 2025) [27].

Acknowledgments

I would like to express my sincere gratitude to everyone who supported and guided me throughout this research.

First, I am very grateful to my supervisor, Dr. Sarel Cohen, for guiding me throughout the project, answering all my questions - even on weekends - and introducing me to helpful colleagues.

I would also like to thank Ohad Rubin, PhD student at Tel Aviv University and my co-supervisor, for generously sharing his expertise, meeting with us in his free time, and patiently answering our questions.

My gratitude extends to Rania Briq, PhD student, for her valuable advice in computer vision and for taking the time to meet with us.

Finally, I would like to thank Prof. Ohad Fried for sharing his expertise in computer vision and providing helpful guidance.

1. Introduction

The purpose of this research is to evaluate whether existing pretrained multimodal language models (LMs) are capable of generating high-quality Scalable Vector Graphics (SVGs) given input images. SVG is a widely used vector format for high-quality, scalable image representation, but the challenge of automatically converting complex raster images into SVGs remains a major hurdle. The ability to automate this conversion with state-of-the-art multimodal models would have wide applications in web design, digital art, and other visual mediums that require high-quality vector images.

Pretrained multimodal LMs, such as PaliGemma [3], Pixtral [5], Emu3 [6], Llama 3.2 [7], and Llava 1.5 [23], have demonstrated capabilities in text-image understanding and generation. However, the potential of these models in vector-based graphic generation has not been fully explored. This research aims to fine-tune Llava 1.5 [23] and similar models to evaluate their ability to generate SVGs directly or facilitate the conversion process.
To support fine-tuning of vision-language models, an initial dataset of 2.5k high-quality tuples - each consisting of a raster image, its corresponding SVG representation, and a detailed text description - was created. This dataset served as the foundation for scaling to a much larger collection. Although the original target was 100k tuples, the dataset was ultimately expanded to 1.8 million high-quality tuples.

Motivation: Current Image-To-SVG State-of-The-Art Using ChatGPT and Claude
As you can see in Figure 1, the current state of the art in image-to-SVG in both ChatGPT [13] and Claude [14] is not good, resulting in very abstract, even absurd looking SVG images. This gives motivation to our work, as there seems to be a large gap for improvement.

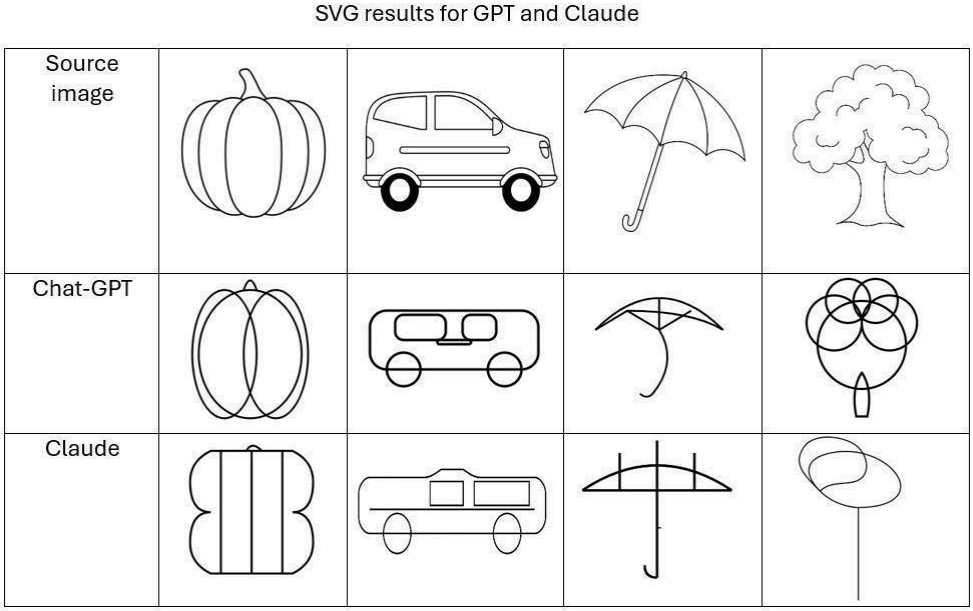


Figure 1: Current State-Of-The-Art using ChatGPT and Claude (raster image to SVG)

2. Related Work

Multimodal models like StarVector [1] have made notable advancements in image-to-SVG conversion. StarVector integrates a CLIP image encoder with StarCoder, a code generation model, to convert images into visual tokens and generate SVG code through next-token prediction. StarVector, trained on the SVG-Stack dataset consisting of over 2 million real-world SVG examples (see Figure 3), is designed to capture the nuances of SVG generation. However, its reliance on simplified SVG datasets can restrict its ability to produce highly detailed or intricate designs. Additionally, the tokenization limitations within its architecture may affect performance when handling more complex image representations.

Figure 2: StarVector pipeline (taken from StarVector paper [1])

In contrast, our research aims to advance SVG generation by leveraging a high-resolution SVG dataset that we specifically curated. This dataset contains intricate examples, allowing a more detailed assessment of SVG quality (Figure 6 shows examples from our initial small dataset). By fine-tuning a multimodal model on this data, we aim to improve its ability to generate complex, high-quality SVGs, surpassing models such as GPT-4V and Claude 3, while addressing limitations of traditional SVG generation methods and API-based approaches.

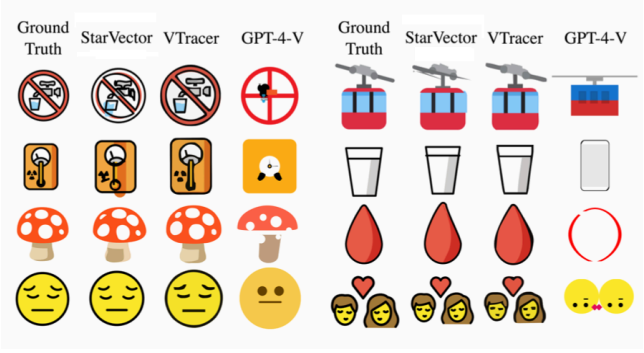


Figure 3: Dataset examples used in StarVector paper and comparison to VTracer [20] and GPT4-v [13] (taken from StarVector paper [1])

During dataset creation, we ensured that images were sufficiently complex to reflect realistic scenarios, rendered at appropriate resolution, and converted into SVGs that preserved key geometric and structural details. We initially constrained SVG descriptions to a maximum of 8k tokens to fit large language models, later reducing this limit to 4k tokens for practical training. Converting raster images to SVGs is a critical step in this process. Prior work - including StrokeNUWA [2], StarVector [1], and research on SVG compression [10] - highlighted the benefits of vectorization for compact and high-quality representations, but these approaches often struggled to consistently produce SVGs suitable for effective fine-tuning.

During the course of our work, the paper OmniSVG [24] was published, presenting a unified approach for multimodal SVG generation, including Text-to-SVG, Image-to-SVG, and character-reference SVG tasks. OmniSVG discretizes SVG commands and coordinates into tokens, effectively separating the structural logic of vector graphics (e.g., commands and hierarchy) from low-level geometric details such as exact coordinates. This strategy allows vision-language models to better capture the underlying structure of SVGs rather than memorizing pixel-level or coordinate-specific information, providing valuable guidance as we approach the fine-tuning stage of our multimodal model.

2.1 Attempts at Generating SVG Images that Did Not Work

During the development of the SVG image generation pipeline, several methodologies were explored to improve the quality and structural precision of the resulting vector graphics. One of the early attempts relied on a multi-step process that combined image resizing, edge detection, clustering, and Bezier curve fitting. The workflow began by resizing each raster image to a consistent width (default 5000px) while preserving its aspect ratio. This normalization step ensured uniform behavior in downstream edge-detection operations.

The resized image was then prepared for edge extraction by converting it to grayscale and applying a series of visual adjustments, including brightness, saturation, and contrast enhancement, as well as hue-shifting and sepia filtering when needed. Edge detection was performed using the Canny algorithm, producing a binary edge map where detected boundaries appeared as white contours on a black background. This representation served as the input for subsequent structural analysis.

To group the detected edge points into meaningful shapes, the DBSCAN clustering algorithm [8] was used. DBSCAN was chosen due to its ability to identify arbitrarily shaped clusters and handle noise effectively - properties that are particularly useful when dealing with natural images. For each cluster, Bezier curve fitting was applied to approximate the contours with smooth mathematical curves. These curves were then converted into SVG path commands, forming the final vector representation. Example outputs of this approach are provided in

Figure-4. Despite producing visually interpretable results, the method introduced inaccuracies and structural artifacts, making it unsuitable for building a high-quality dataset at scale.

In another approach, Pillow was used for image manipulation alongside edge-detection methods from scikit-image to generate a sketch-like representation of the raster images. In this pipeline, the Sobel filter was applied to extract intensity gradients, producing a grayscale edge image that resembled a hand-drawn sketch. This sketch was then passed to Potrace [11], a vectorization tool that converts bitmap outlines into smooth Bézier paths. While Potrace is effective for simple, high-contrast silhouettes, its output in this context lacked the structural fidelity required for detailed images. The resulting SVGs often contained oversimplified contours, missing internal boundaries, and irregular path shapes. Consequently, the quality produced by this method was insufficient for building a high-quality vector dataset.

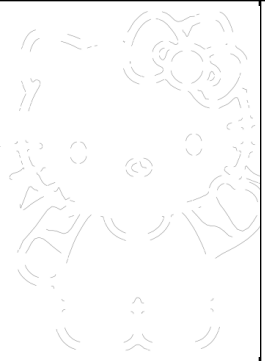
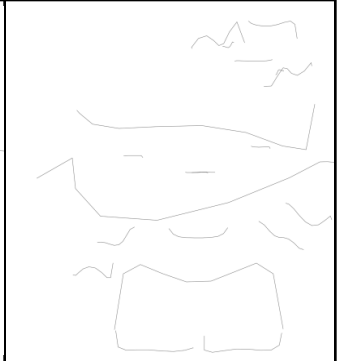
eps=4.00	eps= 4.67	eps= 5.11	eps=6.00
			
number of GPT-4 tokens: 19280	number of GPT-4 tokens: 16178	number of GPT-4 tokens: 1936	number of GPT-4 tokens: 1936

Figure 4: The eps parameter in DBSCAN controls the maximum distance between two samples for them to be considered part of the same cluster.

3. Our Method

We first created a small, high-quality dataset of 2.5k examples, where each example included a raster image, its matching SVG version, and a clear text description. This initial set helped define the structure and quality we wanted and served as the foundation for creating a larger dataset of 10k examples, which was ultimately scaled up to 1.8 million high quality image - SVG - text tuples.

3.1 Creating an Initial Small Dataset

We developed a pipeline integrating generative methods using OpenAI's DALL·E API for image creation. Once a JPG image is generated, we convert it to SVG using the Potrace algorithm [11], which detects image edges and fits smooth curves, ensuring that both the structural integrity and descriptive capacity of the image are preserved. This approach allows us to produce diverse and complex images that can be effectively converted into scalable vector graphics. For generating the textual description of the images, we used GPT-4 Turbo, enhancing the dataset with rich, contextually accurate descriptions. By combining DALL·E for image generation with Potrace for SVG conversion, we have built a robust, high-quality SVG dataset that supports our project's goals. The process is detailed in Figure 5.

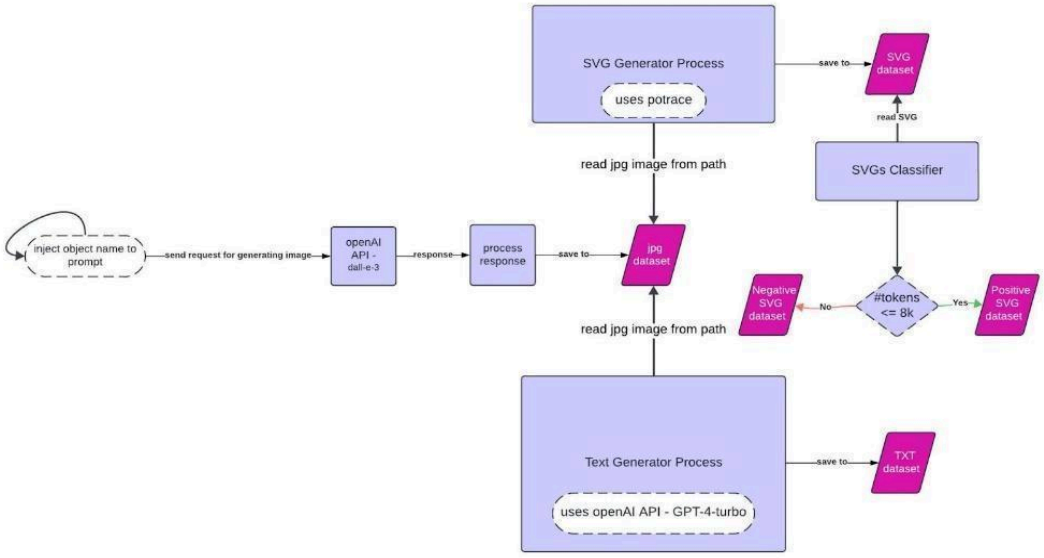


Figure 5: Our pipeline for generating an initial dataset of 2,500 high-quality SVG images. More precisely, we generate tuples of (text, raster image, SVG), for every instance.

Here we present several examples of our pipeline:

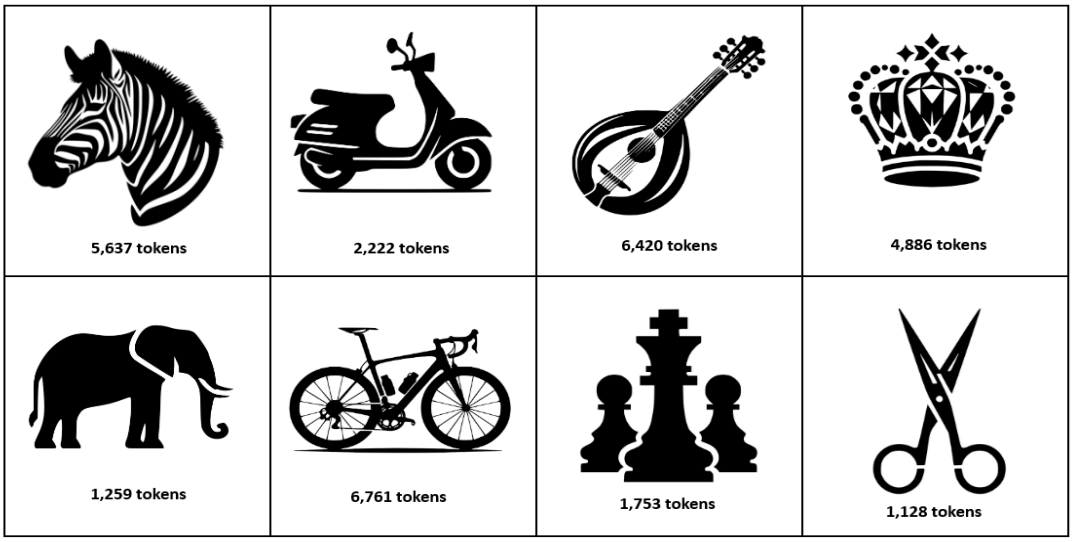


Figure 6. Several SVGs generated by our pipeline from Figure 3. All of these SVG images contain at most 8K tokens (after tokenization). As you can see, these images are of much higher quality than previous attempts, and also fit into a 8K-context of an LLM. The prompt used for generating the raster images before converting to SVG is: “A black silhouette of a {object} with well-defined main features that clearly outline the subject's overall shape. The silhouette shows high contrast between the sharp main contours and the bright white background. The silhouette is very simple and lacks sharp edges and intricate details." where the objects in the figure are {zebra, motorcycle, ukulele, crown, elephant, bicycles, chess, and scissors} correspondingly.

3.2 Scaling Up to Large Dataset (10k samples)

To grow our dataset beyond the first 2.5k examples, we fine-tuned a Stable Diffusion model using DreamBooth [21]. We trained it on our small curated set of raster images, SVGs, and text descriptions so the model would learn the style we needed. Our goal was to generate raster images that convert easily into clean SVGs, so we focused on producing images with simple shapes, minimal details, and a plain white background. We used prompt engineering to push the model toward this minimal, high-contrast style.

The next stage scaled this workflow up further - eventually reaching 1.8 million samples - using a stronger model called Flux [22].

3.2.1 Stable Diffusion Fine-Tuning Using DreamBooth

Stable Diffusion [12] is a text-to-image generative model based on the diffusion framework, where an image is created by gradually transforming random noise into a coherent visual output. The model learns this transformation by reversing a noise-adding process: it takes a noisy latent representation and denoises it step by step, guided by text embeddings that describe the desired scene.

The architecture contains several key components:

- **Latent Space Representation:**
  Rather than generating images directly in pixel space, Stable Diffusion works in a compressed latent space using a Variational Autoencoder (VAE). This significantly reduces memory and computation cost while preserving visual detail.

- **Diffusion Process:**
  During training, the model observes a forward diffusion process that progressively adds noise to the latent image. It then learns the reverse process - removing noise over multiple steps to recover the image described by the prompt.

- **U-Net Backbone:**
  The denoising network is a U-Net, chosen for its ability to capture multi-scale features. It receives the noisy latent, a timestep, and the text embedding, and outputs a cleaner latent representation.

- **Text Conditioning:**
  Text prompts are encoded using a transformer-based text encoder. These embeddings guide the U-Net so the final image aligns semantically with the prompt.

To adapt Stable Diffusion to our specific style - minimalist silhouettes, strong foreground - background separation, and fully white backgrounds - we finetuned the model using DreamBooth [21]. DreamBooth enables personalization: the model can internalize the visual characteristics of a small training set and reproduce them consistently during generation. In our case, this allowed Stable Diffusion to learn the clean icon-like structure required for efficient SVG vectorization.

However, as noted in Guttenberg's blog [26], diffusion models naturally gravitate toward mid-range brightness values. Their noise distribution makes it difficult to generate extremely bright or extremely dark regions, which causes problems when producing images with uniform white backgrounds - an essential requirement for low token SVGs.

Before applying any correction, the standard noise sampled during training is:

```
noise = torch.randn_like(latents)
```

To overcome this brightness limitation, we adopted offset noise, which shifts the noise distribution by adding a low-frequency bias. This encourages the model to produce brighter images and maintain large areas of uniform white:

```
noise = torch.randn_like(latents) + 0.1 * torch.randn(latents.shape[0],
latents.shape[1], 1, 1)
```

3.2.2 The inference process of Stable Diffusion

To expand our dataset from 2.5K to 10K high quality triplets, we ran a large-scale inference process using a Stable Diffusion model fine-tuned with DreamBoot [21]. To create diverse input prompts, we drew inspiration from two large captioning datasets: the COCO dataset, which contains everyday images paired with short human-written captions, and the Text-to-Image-2M dataset, which contains millions of image-text pairs collected for training generative models. We used only the textual descriptions from these datasets, mixing object and scene phrases to produce thousands of unique prompt candidates. However, many of these captions were too long or too complex - often filled with adjectives, background clutter, or scene details that made it difficult for Stable Diffusion to generate clean, simple silhouettes.

To solve this, we used GPT-4 Turbo with a prompt-engineering approach. We provided the model with several examples showing how to turn a complex caption into a short, abstract, silhouette-friendly description. The model learned to simplify captions consistently and returned “NO” when a caption was unsuitable.

The exact prompt format we used was:

```
I have a caption and I want to make it slightly simpler (especially the background and colors) and more specific, and turn it into a caption describing a minimalist abstract silhouette of the same subject.
If the caption is not appropriate to be converted, respond with “NO”.

For example:
"A dog rolling in the snow at sunset" →
"An abstract minimalist silhouette of a dog rolling in the snow, clean outlines, uniform white background."

"An armchair that looks like an apple" →
"An armchair that looks like an apple, minimalist abstract silhouette, clean outlines, uniform white background."

"pink photo of Tokyo" →
"Buildings in Tokyo, minimalist abstract silhouette, clean outlines, uniform white background."

"Anti-fracking protest rocks NY governor's state of the state address" →
"A group of people protesting in front of a building, minimalist abstract silhouette, clean outlines, uniform white background."

"st peter's square: St Peters Square in Rome Italy" →
"A city square with a recognizable landmark, minimalist abstract silhouette, clean outlines, uniform white background."

"Nickelodeon Paw Patrol 'Calling All Pups' Soft Potty Seat" →
"NO"

Now process the following caption: "{context}"
Format the output as:
<simplified>
```

After simplification, each processed caption was wrapped in a consistent style template to match the aesthetic learned during DreamBooth training-specifically:

```
“a black-and-white silhouette of {processed_line} on a solid white background.”
```

We then fed these prompts into our fine-tuned Stable Diffusion model. Because the model had learned minimalist shapes, strong foreground-background separation, and clean white backgrounds, it produced images that were generally easy to convert into SVGs. Each generated raster image was passed through our Potrace-based vectorization pipeline, after which we applied two filtering rules: (1) we removed SVGs that exceeded 4,000 tokens to maintain compactness, and (2) we kept only samples with sufficiently high CLIP similarity to their prompts. Images that passed both tests were used to improve the fine-tuning. The iterative process of the fine-tuning is shown in Figure 7.

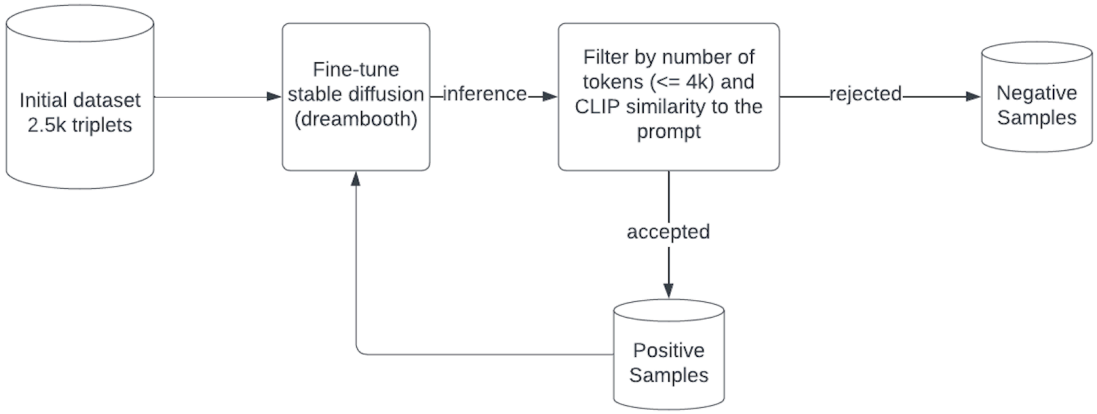


Figure 7: Iterative Fine-Tuning Process of SD Dreambooth

The final results were mixed. Many images were clean and structurally correct, converting smoothly into SVGs. (See Figure 8 – Good Inference Examples.) Others contained broken silhouettes, missing structure, or unusual artifacts. (See Figure 9 – Problematic Inference Examples.) These limitations suggested that DreamBooth-tuned Stable Diffusion was not strong enough for reliable large-scale silhouette generation, motivating our transition to a more capable model (Flux) for scaling from 10K to 1.8M samples.

Figure 8: Good Inference Examples of finetuned SD Dreambooth

Figure 9: Problematic Inference Examples of finetuned SD Dreambooth

3.3 Flux Fine-Tuning and Large-Scale Expansion to 1.8 Million Samples

After building the initial 10K dataset with Stable Diffusion, we moved to a stronger model Flux [22], created by Black Forest Labs to scale our dataset to more than a million examples. Flux is a modern diffusion model built with a hybrid multimodal transformer architecture and scaled to 12B parameters, making it far more capable than Stable Diffusion for producing structured, clean, and consistent images.

We fine-tuned Flux using a lightweight LoRA [25] (weight 0.1). The LoRA did not change the style dramatically - it only encouraged slightly simpler shapes and more stable outlines. The real performance improvement came from two ingredients we kept from the SD stage: (1) simplified captions, and (2) a consistent silhouette-style wrapper prompt.

For captioning, we reused the same simplified COCO and text-to-image-2M descriptions simplified by GPT-4 Turbo. These short captions helped Flux focus on the core object without adding unnecessary textures or backgrounds. We then wrapped every simplified caption inside a consistent SVG-friendly template:

```
"minimal {processed_line} with clean outlines, flat fills, high contrast, scalable vector-style graphic."
```

We also used a negative prompt to prevent the model from adding unwanted details, telling it to avoid: photorealistic, noise, texture, watercolor, gradients, film grain, shading.

Using this setup, we generated approximately 1.8 million raster images with the FLUX.1-schnell model. Every image was then processed through Potrace [11], which produced SVGs with clean structure and controlled token counts. Only images whose SVGs met our quality constraints (e.g., low token count, clean silhouettes, proper shape structure) were kept.

The final result is a large-scale dataset of about 1.8 million triplets - caption, raster image, and SVG. To our knowledge, this is one of the largest and most carefully curated datasets designed specifically for vector friendly image generation, SVG reasoning, and image-to-vector learning at scale.

Figure 10: Examples of inference images generated by the fine-tuned FLUX model

4. Fine-Tuning LLaVA for Image-to-SVG Generation

4.1 Dataset Selection

We initially considered using our large-scale FLUX [22] dataset, which contains 1.8M triplets of raster images, captions, and SVGs. However, many of the converted SVG sequences were longer than 4,000 tokens, exceeding the context window of the LLaVA [23] model we used. This caused truncation, preventing the model from seeing the full SVG and making fine-tuning ineffective. To address this, we used the OmniSVG [24] dataset, which contains illustrations and icons. We selected the icon subset, which contains simple, compact examples with short SVG sequences that fit within the model's context window, allowing effective training for Image-to-SVG generation.

4.2 LLaVA Architecture

LLaVA [23] (Large Language and Vision Assistant) is a large multimodal model that integrates visual and textual information. It connects a pretrained vision encoder to a large language model via a trainable multimodal fusion layer, converting image features into tokens that the language model can process alongside text. The model is autoregressive, predicting one token at a time, and is instruction-tuned to follow structured prompts. This architecture enables LLaVA to generate structured outputs, such as SVG code, directly from images.

4.3 Training Principles

To adapt LLaVA-1.5-7b for Image-to-SVG generation, we employed several key strategies:

- **Gradient Accumulation:**
  Due to limited GPU memory, we trained with a small batch size (1 sample) but accumulated gradients over multiple steps before updating the weights. This simulates a larger effective batch size, stabilizing training without exceeding memory limits.

- **Multimodal Input with Captions:**
  While the core task is Image-to-SVG, including textual captions during training improves model performance. Captions provide explicit guidance about the icon's semantic content, helping the model generate accurate SVGs even in ambiguous or noisy images. In the training prompt, the image is represented by a special token that signals the model to use visual features from the input image, and the caption is included directly in the instruction text. This combination allows the model to align visual information with the intended semantic meaning of the icon.
  Additionally, a fraction of samples are text-only (Text-to-SVG), which helps the model retain its instruction-following ability and prevents forgetting the Image-to-SVG task.

- **LoRA-based Model Adaptation:**
  Low-Rank [25] Adaptation (LoRA) was applied to the language and multimodal fusion layers while keeping the vision encoder frozen. This focuses training on the cross-modal mapping and SVG sequence generation, reducing computational cost.

- **Data Collation and Loss Masking:**
  Sequences were padded within each batch, and loss was computed only on the generated SVG tokens. Prompt tokens were masked to prevent the model from learning the instruction text as part of the output.

- **Loss Function:** We use autoregressive next-token prediction with cross-entropy loss, computed only on the SVG output tokens. Prompt and padding tokens are masked so the model learns to generate the SVG sequence rather than copying the instruction text.

4.4 Instruction-Tuned Prompts

The model was trained using structured prompts for the Image-to-SVG task:

```
### Instruction:
Recreate this icon as a compact, valid SVG based on the image. Use
viewBox="0 0 256 256"; prefer <path>, <rect>, <circle>; no external CSS.
```

In this prompt, the image is provided through the model's image input token, while the caption appears in the instruction text. This setup allows the model to generate SVG code that reflects both the visual structure and the intended semantic meaning of the icon.

- **Output Stabilization:** Captions provide clear semantic instructions, helping the model generate accurate SVGs even when the input image contains noise.

- **Better Generalization:** Textual descriptions allow the model to learn broader mappings between high-level concepts and SVG structures, improving its ability to handle diverse content.

- **Task Retention:** Including text-only samples prevents the model from forgetting the core Image-to-SVG task, ensuring robust SVG generation even when images are partially missing or degraded.

Training a vision-language model (VLM) on images alone reduces the task to a direct pixel-to-SVG mapping, making it harder for the model to capture the intended semantics and increasing sensitivity to noise. Adding text provides an additional layer of conceptual understanding, guiding the model toward producing SVGs that reflect the intended content more faithfully.

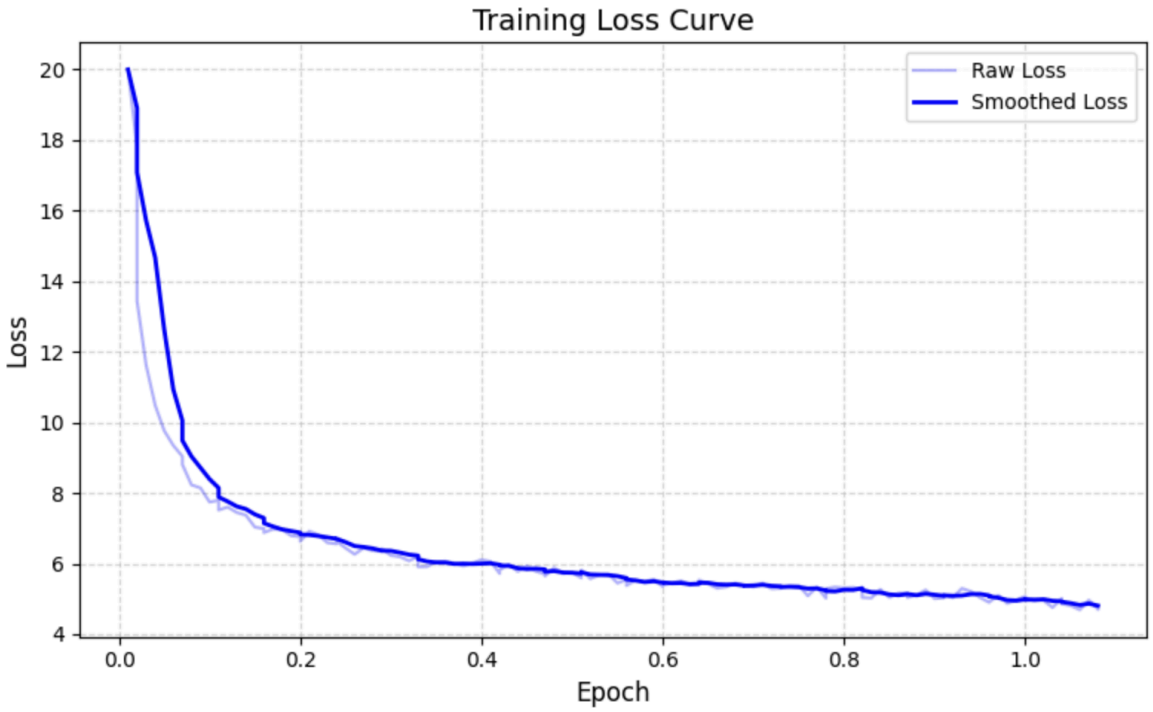


Figure 11: The training loss curve of the Llava-1.5-7b

5. Experiments

To evaluate the performance of our Image-to-SVG generation model, we conducted both quantitative and qualitative experiments, comparing our model against prior approaches and assessing improvements after fine-tuning.

5.1 Qualitative Evaluation

Qualitative assessment involved visually inspecting the generated SVGs on known datasets. The evaluation focused on preservation of image details, aesthetic appeal, and overall fidelity. Figure 12 illustrates examples of high-quality SVGs generated by our pipeline.

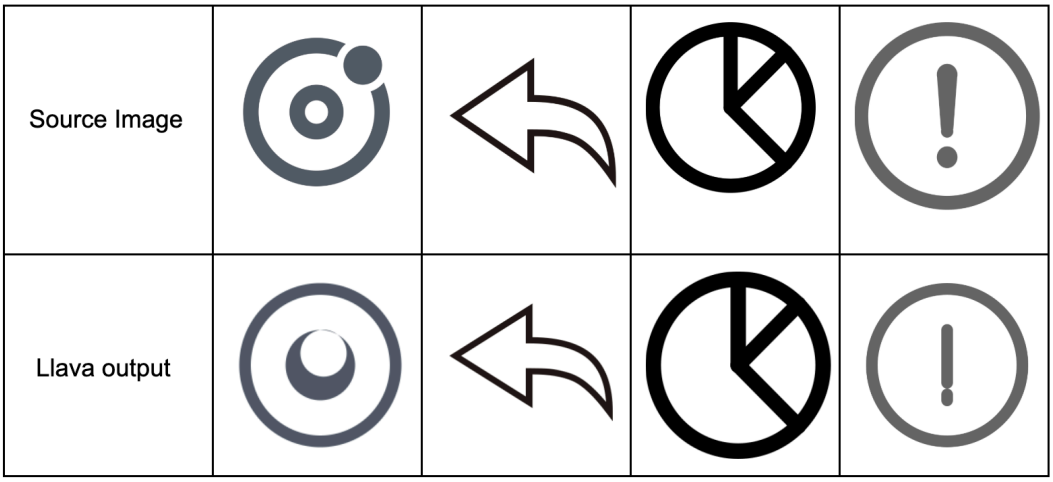


Figure 12: Examples of image to SVG results of the finetuned Llava-1.5-7b

5.2 Quantitative Evaluation

For rigorous quantitative evaluation, we employed a combination of raster-based and vector-based metrics, some of which were adopted from prior work on StarVector [1].

In our experiments, we evaluated every predicted SVG by comparing it to its corresponding ground-truth SVG across a wide suite of metrics. All SVG outputs - both ground truth and model prediction - were first rasterized at a fixed resolution to enable consistent pixel-based measurements. For each sample, we computed classical metrics such as MSE, SSIM, PSNR, and IoU, as well as perceptual and semantic metrics including LPIPS and CLIP similarity. In

addition, to capture geometric fidelity at the vector level, we computed the Chamfer Distance directly on the SVG path coordinates. These metrics were averaged across the entire evaluation set, providing a comprehensive quantitative view of reconstruction quality, perceptual similarity, and vector-shape accuracy.

Metrics:

1. **Mean Squared Error (MSE)**
   MSE measures the average squared difference between corresponding pixel intensities in the generated image I_i and the reference image $I_i^{'}$:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(I_i - I_i^{'})^2$$

   Lower MSE indicates better reconstruction fidelity.

2. **Structural Similarity Index (SSIM) [19]**

   SSIM, evaluates perceptual similarity by comparing luminance, contrast, and structure:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y+C_1)(2\sigma_{xy}+C_2)}{(\mu_x^2+\mu_y^2+C_1)(\sigma_x^2+\sigma_y^2+C_2)}$$

   where μ_x and μ_y are the means of pixel intensities in images x and y, σ_x^2 and σ_y^2 are the variances, σ_{xy} is the covariance, and C_1 and C_2 are constants to stabilize the division. SSIM, introduced in [19], captures perceptual similarity by comparing patterns of pixel intensity, luminance, and contrast in the rasterized SVGs and aligns closely with human visual perception.

3. **Peak Signal-to-Noise Ratio (PSNR) [17]**
   PSNR quantifies pixel-level fidelity, capturing differences in intensity values: higher PSNR indicates clearer, less noisy outputs.

4. **Intersection over Union (IoU) [18]**
   IoU measures spatial alignment between rasterized generated SVGs and ground truth:

$$IoU = \left|\frac{A \cap B}{A \cup B}\right|$$

   where A and B are the sets of pixels in the reference and generated images. Higher IoU reflects better shape alignment.

5. **Learned Perceptual Image Patch Similarity (LPIPS)**
   LPIPS assesses perceptual similarity using deep features, providing additional insight into visual quality beyond pixel-level metrics. Lower LPIPS indicates higher perceptual similarity.

6. **CLIP Similarity**
   We also compute the cosine similarity between image and SVG embeddings using CLIP, evaluating semantic alignment between input images and generated SVGs.

7. **Chamfer Distance (CD) [16]**
   CD is a vector-based metric that measures similarity between point sets representing the reference and generated SVG shapes. Lower CD indicates better geometric fidelity without rasterization

Average Quantitative Evaluation Metrics for Image-to-SVG Generation

Metric	Before Fine-Tune	After Fine-Tune	Improvement (%)
MSE	0.089515	0.037797	57.776%
SSIM	0.736128	0.758728	3.070%
PSNR	49.543561	59.061121	19.210%
IoU	0.029689	0.128543	332.965%
LPIPS	0.406613	0.224022	44.905%
CLIP Sim	0.868794	0.949437	9.282%
Chamfer	0.179853	0.096910	46.117%

Figure 13

Results Analysis

The table, described in Figure 13, reports the average quantitative evaluation metrics for our Image-to-SVG generation model, both before and after fine-tuning. These metrics capture different aspects of fidelity, from pixel-level accuracy to vector-based geometric similarity.

After fine-tuning, all metrics show substantial improvements, reflecting the model's enhanced ability to generate SVGs that faithfully represent the input images. In particular:

- **Mean Squared Error (MSE)** decreased from 0.0895 to 0.0378, indicating a substantial reduction in pixel-level differences between the rasterized SVGs and the original images.

- **Structural Similarity Index (SSIM)** increased from 0.7361 to 0.7587, suggesting better preservation of perceptual structures such as shape, contrast, and luminance in the generated SVGs.

- **Peak Signal-to-Noise Ratio (PSNR)** improved from 49.54 to 59.06, highlighting clearer and more accurate reconstruction at the pixel level.

- **Learned Perceptual Image Patch Similarity (LPIPS)** dropped from 0.4066 to 0.2240, confirming that the generated images are perceptually closer to the originals, according to human-aligned features.

- **CLIP** Similarity rose from 0.8688 to 0.9494, reflecting that the semantic content of the generated SVGs aligns more closely with the input images.

- **Chamfer Distance** decreased from 0.1798 to 0.0969, indicating that the geometric outlines of the vector shapes match the original shapes more precisely.

- **Intersection over Union (IoU)** shows some improvement (0.0297 to 0.1285) compared to prior results, but remains relatively low overall. This is expected because the dataset primarily contains icons with very sparse and compact shapes. In such cases, even minor spatial shifts in the generated SVGs can lead to large reductions in pixel-wise overlap, while perceptual and geometric similarity remains high.

6. Conclusion and Future Work

In this work, we set out to build a large dataset of (caption, raster image, SVG) triplets for training and evaluating future vision - language models on image-to-SVG tasks. We began with a small curated set and expanded it through iterative generation, filtering, and refinement (using SD dreambooth). After reaching 10K samples, we scaled up using the Flux model and ultimately produced approximately 1.8M triplets, each converted to SVG through our vectorization pipeline.

Looking ahead, there are additional techniques from recent research that we plan to explore. The OmniSVG paper proposes several strategies for simplifying and normalizing SVGs so they are easier for models to learn. For example, they use tools like picosvg to remove extra structural elements and rewrite shapes in a cleaner form - turning what might otherwise be many line segments into a structured command such as:

```
<rect x="80" y="90" width="100" height="100"/>
```

They also simplify SVGs into small sets of atomic path commands and remove elements like <g> (group) and transform. Incorporating or adapting some of these ideas may help reduce noise, unify structure, or make SVG sequences more model-friendly.

Future work will focus on evaluating these simplification and normalization methods within our pipeline, as well as exploring iterative SVG generation - splitting complex scenes into smaller parts instead of producing a single long sequence. We also plan to test techniques that may reduce inference cost, such as multi-token prediction or KV-cache compression, especially important for long SVG token sequences. As models continue to improve, there is room to investigate stronger diffusion models or hybrid approaches to further enhance SVG quality and token efficiency.

Overall, while we observed clear improvements as our dataset grew, the work also highlights current limitations, such as long SVG sequences, context-window constraints, and variability across samples. Producing SVGs directly from text or images is feasible but still challenging, and there is substantial room for refinement. Our dataset and pipeline provide a foundation for advancing vector-aware vision-language modeling, and we aim to continue improving both the SVG representations and the models that learn from them.

References

[1] Juan A. Rodriguez et al. StarVector: Generating Scalable Vector Graphics Code from Images. 2023. arXiv: 2312 . 11556 [cs.CV]. url: https://arxiv.org/abs/2312.11556

[2] Zecheng Tang et al. “StrokeNUWA - Tokenizing Strokes for Vector Graphic Synthesis”. In: Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024. 2024. url: https://openreview.net/forum?id=eVlx8DaG9h

[3] Lucas Beyer et al. PaliGemma: A versatile 3B VLM for transfer. 2024. arXiv: 2407.07726 [cs.CV]. url: https://arxiv.org/abs/2407.07726

[4] Google AI. (n.d.). Fine-tuning PaliGemma. Retrieved October 27, 2024, from https://ai.google.dev/gemma/docs/paligemma/fine-tuning-paligemma

[5] Pravesh Agrawal et al. Pixtral 12B. 2024. arXiv: 2410.07073 [cs.CV]. url: https://arxiv.org/abs/2410.07073

[6] Xinlong Wang et al. Emu3: Next-Token Prediction is All You Need. 2024. arXiv: 2409.18869 [cs.CV]. url: https://arxiv.org/abs/2409.18869

[7] Meta AI. (2024). LLaMA 3.2. Retrieved October 27, 2024, from https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/

[8] Martin Ester et al. “A density-based algorithm for discovering clusters in
1. large spatial databases with noise”. In: Proceedings of the Second Inter-
2. national Conference on Knowledge Discovery and Data Mining. KDD’96.
3. Portland, Oregon: AAAI Press, 1996.
4. url: https://www.dbs.ifi.lmu.de/Publikationen/Papers/KDD-96.final.frame.pd

[9] Francis Bach and Michael Jordan. “Learning Spectral Clustering”. In: Advances in Neural Information Processing Systems (NeurIPS). Ed. by S. Thrun, L. Saul, and B. Sch¨olkopf. Vol. 16. MIT Press, 2003. url: https://proceedings.neurips.cc/paper_files/paper/2003/file/d04863f100d59b3eb688a11f95b0ae60-Paper.pdf

[10] S.R. Ginn. “On the Compression of SVG Images”. Master’s thesis. Faculty of Science (FNWI), University of Amsterdam, 2017. url: https://scripties.uba.uva.nl/search?id=record_25356

[11] Peter Selinger. “Potrace : a polygon-based tracing algorithm”. In: 2003.
1. url: https://api.semanticscholar.org/CorpusID:1419652

[12] Patrick Esser et al. Scaling Rectified Flow Transformers for High-Resolution
1. Image Synthesis. 2024. arXiv: 2403 . 03206 [cs.CV]. url: https://arxiv.org/abs/2403.03206

[13] OpenAI et al. GPT-4 Technical Report. 2024. arXiv: 2303.08774 [cs.CL]. url: https://arxiv.org/abs/2303.08774

[14] The Claude 3 Model Family: Opus, Sonnet, Haiku”. In: url: https://api.semanticscholar.org/CorpusID:268232499

[15] Jun-Yan Zhu et al. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. 2020. arXiv: 1703.10593 [cs.CV]. url: https://arxiv.org/abs/1703.10593

[16] Haoqiang Fan, Hao Su, and Leonidas Guibas. A Point Set Generation Network for 3D Object Reconstruction from a Single Image. 2016. arXiv:1612.00603 [cs.CV]. url: https://arxiv.org/abs/1612.00603

[17] Alain Hor´e and Djemel Ziou. “Image Quality Metrics: PSNR vs. SSIM”. In: 2010 20th International Conference on Pattern Recognition. 2010, pp. 2366–2369. doi: 10.1109/ICPR.2010.579. url: https://ieeexplore.ieee.org/document/5596999

[18] Hamid Rezatofighi et al. Generalized Intersection over Union: A Metric and A Loss for Bounding Box Regression. 2019. arXiv: 1902.09630 [cs.CV]. url: https://arxiv.org/abs/1902.09630

[19] Zhou Wang et al. “Image quality assessment: from error visibility to structural similarity”. In: IEEE Transactions on Image Processing 13.4 (2004), pp. 600–612. doi: 10.1109/TIP.2003.819861. url: https://ieeexplore.ieee.org/document/1284395

[20] Yao Liu et al. Vtracer: A Deep Learning-based Tool for Vectorizing Raster Images into Scalable Vector Graphics (SVGs). Accessed: 2024-11-09. 2024. url: https://github.com/facebookresearch/VTracer

[21] Nataniel Ruiz et al. DreamBooth: Fine Tuning Text-to-Image Diffusion Models for Subject-Driven Generation. 2023. arXiv: 2208.12242 [cs.CV]. url: https://arxiv.org/abs/2208.12242

[22] Labs, B. F., Batifol, S., Blattmann, A., Boesel, F., Consul, S., Diagne, C., … Smith, L. (2025). FLUX.1 Kontext: Flow matching for in-context image generation and editing in latent space. arXiv. https://arxiv.org/abs/2506.15742

[23] Li, Jing Yang, Jie Yu, Xiyao Wang, Bin Qin, Yumeng Wang, Zizhen Yan, Ziyong Feng, Ziwei Liu, Bo Li, and Jiankang Deng. 2025. LLaVA-OneVision-1.5: Fully Open Framework for Democratized Multimodal Training. arXiv:2509.23661 [cs.CV] https://arxiv.org/abs/2509.23661

[24] Yang, Y., Cheng, W., Chen, S., Zeng, X., Yin, F., Zhang, J., Wang, L., Yu, G., Ma, X., & Jiang, Y. (2025). OmniSVG: A Unified Scalable Vector Graphics Generation Model. arXiv. https://arxiv.org/abs/2504.06263

[25] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., & Chen, W. (2021). LoRA: Low-Rank Adaptation of Large Language Models. arXiv. https://arxiv.org/abs/2106.09685

[26] Nicholas Guttenberg. 2023. Diffusion Models with Offset Noise. https://www.crosslabs.org/blog/diffusion-with-offset-noise

[27] Shachar Levy, Maayan Mashhadi, Sarel Cohen, Ohad Rubin. The 18th ACM International Systems and Storage Conference (SYSTOR 2025). VisVec: A Milestone Towards Compressing Images by Converting Them to SVG using LLMs.