The Academic College of Tel Aviv-Yaffo

The Faculty of the School of Computer Science

Enhancing Large Language Models (LLMs) for Complex SVG Generation from Textual Descriptions

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

Maayan Mashhadi

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

Dr. Sarel Cohen

Contents

1 Abstract

Existing large language models (LLMs) like GPT and Claude can generate basic SVGs (Scalable Vector Graphics) based on textual descriptions. However, these SVGs tend to be simplistic and abstract. This project aims to enhance the ability of LLM to generate more complex, high-quality SVGs that offer realistic depictions of everyday objects, based just on textual descriptions.

To achieve this, we developed a pipeline for generating SVGs from text. We constructed a large, high-quality dataset consisting of triplets of textual descriptions, raster images, and their corresponding SVG files. This dataset was used to fine tune an LLM for the text to SVG generation task. The pipeline involves several generative models and algorithms, such as DALL-E, Stable Diffusion, and Flux, for image generation, and Potrace for converting raster images into SVG format. We also experimented with prompt engineering and fine-tuning methods to improve output quality.

For fine tuning the LLM, we used Cross-Entropy Loss for the next-token prediction task, where the model predicts the next token in the SVG sequence. The training process included gradient accumulation to handle larger batches. Fine tuning on our dataset was unsuccessful due to the LLM's small context window. However, we successfully fine tuned with the OmniSVG dataset on icons (which was released during our work). In the results of CLIP similarity score the model accepts 0.3 - which means the model can capture some broad aspects, but often miss specific details from the prompt, and for Aesthetic 4.57- which indicates the generated SVG are pleasing visuality, they are not broken, but not detailed enough.

We published poster about our new dataset in the The 18th ACM International System and Storage Conference (SYSTOR '25 Posters) [13]

Acknowledgments

I would like to thank the people who contributed to this research and generously shared their knowledge.

First, I would like to thank Dr. Sarel Cohen, my supervisor, for driving the project forward, kindly answering every question (even on weekends), and connecting me with other colleagues.

I would also like to thank Ohad Rubin, a PhD student at Tel Aviv University, who served as a co-supervisor. Thank you for sharing your expertise, meeting with us in your free time, and patiently answering our questions.

My thanks also go to Rania Briq, a PhD student, for sharing her knowledge in computer vision and for meeting with us in her free time.

Finally, I would like to thank Prof. Ohad Fried for sharing his knowledge in computer vision.

2 Introduction

SVG is an XML based image format used to describe vector graphics. Unlike raster images, which are composed of pixels, SVGs represent images through mathematical paths, shapes, and colors, allowing them to scale infinitely without losing quality. SVGs play a crucial role in modern image rendering, offering scalable resolution, flexible usability, and easy editing. Their use is especially widespread in web development and graphic design.

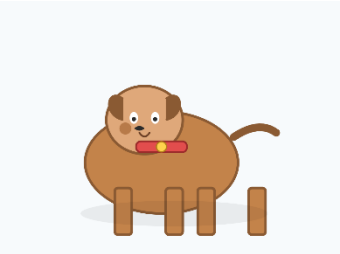
In recent years, the rapid advancement of LLMs improved their ability to generate structured outputs from textual descriptions. However, while models such as GPT and Claude have achieved impressive results in text to image generation, generating vector-based graphics directly from text remains a largely unexplored area. Unlike raster images, SVGs require a precise structural representation that captures both visual appearance and geometric relationships.

The importance of this work lies in improving the connection between textual input and high-quality SVG outputs using LLM, which could significantly expand the potential applications of these models in creative and technical domains.

In this thesis, we introduce a pipeline for generating SVG outputs from textual descriptions. Our approach involves two main components:

1. Constructs a high-quality dataset of text-image-SVG triplets, enabling the fine-tuning of an LLM for this task.
2. Leverages the fine-tuned model to produce SVGs from textual input.

As you can see in Figure 1, the current state of the art in text-to-SVG in both ChatGPT and Claude 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.

Prompt	ChatGPT	Claude
please create an SVG of a dog		

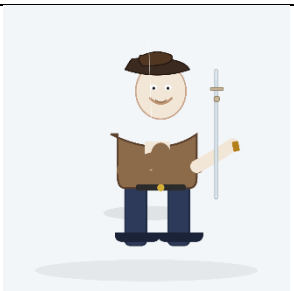
Please create an SVG of a man holding a sword, he is also wearing a cowboy hat		

Figure 1. Comparing results of ChatGPT and Claude for asking SVG of a specific object.

3 Tools and Techniques

During our project, we used some tools and techniques for creating the new dataset and fine tuning

DALL-E - is a generative AI model developed by OpenAI that can create images directly from textual descriptions. It combines methods from both natural language processing and computer vision to produce visual content based on user prompts. Built on a transformer architecture, DALL-E first encodes the text input into a latent representation that captures the semantic and stylistic meaning of the prompt. This latent representation is then decoded by the model into a pixel-based image through a series of neural network layers.

Potrace - is an algorithm for transforming a bitmap into a smooth, scalable image. The input for this algorithm is a bitmap and the output is a vector image format (like SVG).

Stable Diffusion – is a text to image model that produce image from textual description. It does it by three steps:

1. Text Representation Generation - converts a text prompt into a text vector representation.
2. Image Representation Refining - it starts with random noise, refines the image representation little by little, with the guidance of the text representation.
3. Image Upscaling - it upscales the image representation into a high-resolution image.

LoRA - is a technique used to adapt machine learning models to new contexts in an efficient way. It does it by adding lightweight pieces to the original model, rather than changing the entire model. LoRA freezes the original weights and parameters of the model as they are. Then, on top of this original model, it adds a lightweight addition called a low-rank matrix, which is then applied to new inputs to get results specific to the context.

FLUX - is a generative AI model that transforms descriptive text prompts into high-quality, detailed images. It achieves state-of-the-art performance in image quality and prompt adherence. The model is based on rectified flow transformers, a modern technique that is generally more efficient than traditional diffusion models.

Llama 3 – is a family of open source LLM's. They are released in different sizes. Llama 3 was pre trained on a larger dataset of over 15 trillion tokens of publicly available, high-quality text and code.

4 Related Work

There are many different methods that explore the creation of SVG from textual descriptions.

VectorFusion [1] introduced a method for generating abstract vector graphics (SVG format) from text captions. It utilizes pre-trained Stable Diffusion, for sampling raster image from text input, this is an image that captures the semantic content of the text input, and then vectorizing this image. There is a two stage pipeline: sampling an image from Stable Diffusion, then vectorizing it automatically. Given text, we sample a raster image from Stable Diffusion. Then, it automatically traces the raster sample to convert it to an SVG using the off-the-shelf Layer-wise Image Vectorization program (LIVE) - which produces clean SVGs. It utilizes score distillation sampling (SDS) loss, which encourages the generated vector graphics to match the conditional distribution learned by the pre-trained Stable Diffusion text-to-image diffusion model.

SVGDreamer [2] presents an optimization-based approach for generating diverse vector graphics from text prompts. The method's process is divided into two stages: semantic-guided vectorization of images and SVG generation using VPSD optimization.

DiffSketcher [3] presents a method for generating vectorized sketches from text prompts. Built on a pre-trained text-to-image diffusion model, DiffSketcher optimizes a set of Bezier curves using an enhanced version of SDS loss. The approach works as follows: given a text prompt y that describes the target subject and a set of control points for strokes, it generates a sketch S that captures the text's semantic features. To initialize stroke control points, DiffSketcher extracts and merges attention maps from the U-Net within the latent diffusion model.

During our work, the paper **OmniSVG** [4] was released. They propose a unified framework built on pre-trained vision-language models (VLMs) to perform multimodal SVG generation (text to SVG, image to SVG, character-reference SVG). They discretize SVG commands and coordinates into tokens, thereby decoupling the structural logic of the vector graphics (e.g., commands, hierarchy) from low-level geometry (exact coordinate values).

Our work differs from the above works as the above results focus on utilizing and optimizing pre-trained Stable Diffusion or VLM, while our work focuses on utilizing LLM to generate SVG given a text prompt.

4.1 Attempts for Converting Raster Images to SVG Images that Did Not Work

We tried many clustering methods, to cluster each point that is part of a specific piece in the image, the clusters are converted into SVG paths. For each of these clusters, we compute the Bezier curve. We first attempted using the **Spectral Clustering** [5] algorithm, and then **DBSCAN** [6]. The results were not very good visuality and the number of tokens was very high. When we tried to compress those images, it removed the main features from the image as described below.

eps=4.00	eps= 4.67	eps= 5.11	eps=6.00

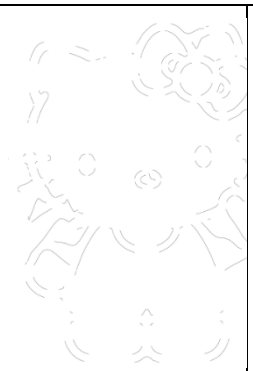
			
number of GPT-4 tokens: 19280	number of GPT-4 tokens: 16178	number of GPT-4 tokens: 1936	number of GPT-4 tokens: 1936

Figure 2. Our attempts to convert raster image to SVG using DBSCAN Algorithm, and the resulting token number in the final SVG image.

5 Our Method

We created two pipelines for creating the new large dataset. The first one, a small dataset of text description, image and their corresponding SVG - which classified to positive and negative according to **token count**. The second, is a scaling up dataset from the small one, a large dataset that consists of the triplets, who will be fine tuned on LLM.

5.1 Creating New Small Dataset

We found that there is a style of images that are “SVG friendly” - non realistic, black and white, with straight lines in the edges. To create these “SVG friendly” images, we introduce a pipeline that generates images using a generative model and then converts them into high-quality SVGs, by Potrace algorithm.

Our pipeline uses **OpenAI’s DALL-E** API for creating these raster images. We created a prompt template for DALL-E that describes a specific kind of image and injects into the template the object we would like to be generated from the prompt. Example of a template: “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."

As you can see, the prompt describes a detailed black and white image that emphasizes the main features that outline the shape and are simple without sharp edges. As a result, the DALL-E model generates images that are naturally easier to convert to SVG.

When we finish generating the appropriate raster images, we convert them to SVG via **Potrace** [7]. In our pipeline, we convert the jpg image to a bitmap which is the input for the Potrace algorithm.

The final step is to classify the SVG images according to their token count. An image is classified as **positive** if it has **less than** 8k tokens, otherwise it's negative. With the combination of the DALL-E API and Potrace we built a small high-quality SVG dataset consisting of tuples of 2500 raster images with their corresponding image descriptions (text) and SVG.

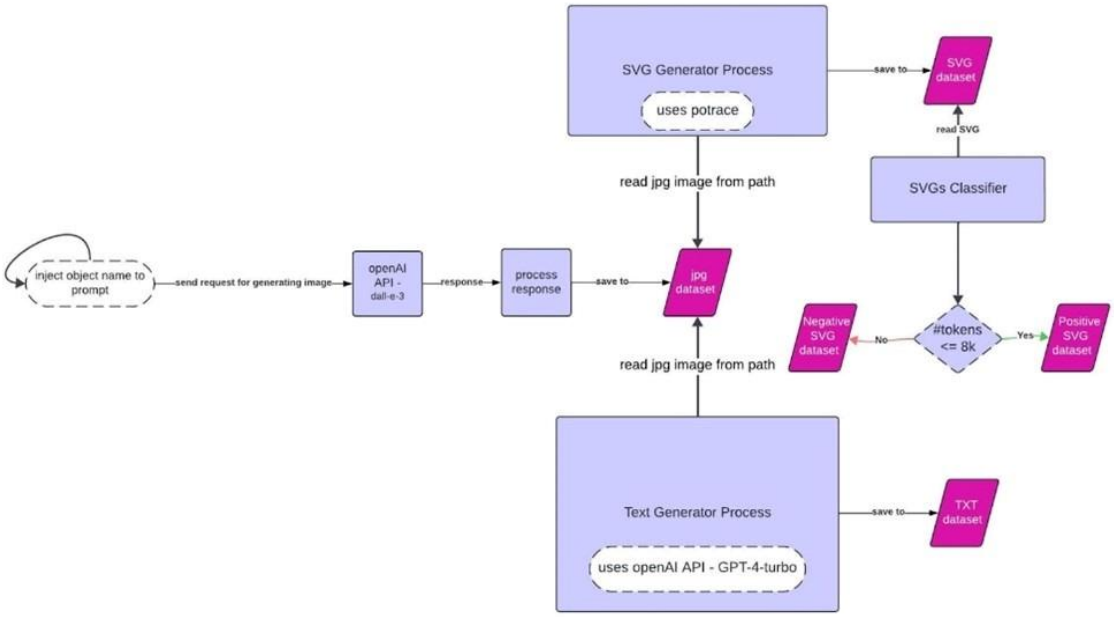


Figure 3. Small dataset creation process

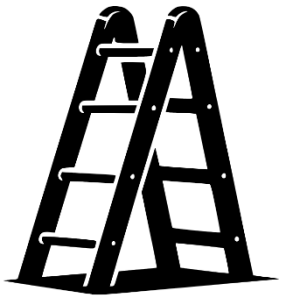
		
Number of tokens: 2356	Number of tokens: 2363	Number of tokens:4514

Figure 4. illustrations of a ladder, a fish and a pineapple (left to right) in SVG format from our small dataset.

5.2 Scaling Up to Large Dataset

We used the small dataset as a basis for creating the larger dataset. In this part we created other pipelines for the larger dataset creation that involved fine tune Stable Diffusion, and prompt engineering with Flux. This dataset will serve as the dataset of fine tune LLM.

5.3 Fine Tune Stable Diffusion

We fine tune **Stable Diffusion v1.5** [8] with the method of **LoRa** on the small dataset, for efficient fine tuning. After that, for inference we performed **Grid Search** [9], for searching the best result of images by: trying different wording prompts, number of inference step range (the larger the number, the image is more detailed and complex, but takes more time), seeds (the initial starting point of the image generation) and scale range (how much the model instructed by the prompt or has creative freedom). We chose the best combination results according to the SVG output.

At the end, we converted the images output of the stable diffusion to SVG with Potrace. At first, the results weren't good because the stable diffusion tends to generate dark background images, although the prompt indicates a white background. This resulted in a larger SVG output in terms of tokens count. We converted the dark background to white with the CV2 python library.

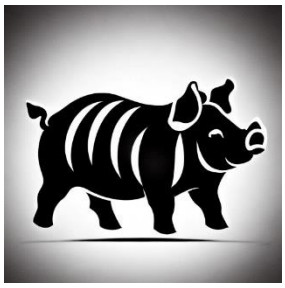
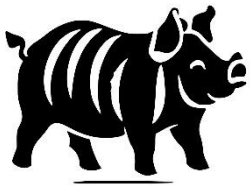
Prompt	Before Fine Tune	Generated Image (After Fine Tune)	Converted Background
A black silhouette of a pig with white background			
A black silhouette of man and woman holding hands with white background			

Figure 5. Example of prompts and generated images before and after fine tuning and converted the background to white (from left to right).

5.3.1 Scaling Up to 10K samples

The fine tune process is a loop of: generating (in inference time) images of the friendly SVG style (after fine tuning on the small dataset) and then feed it to the model for continuing fine tune, the wrapper of prompts be: a black-and-white silhouette of {processed_line} on a solid white background.

For the inference and the continue fine tune, we used two datasets of image description and their corresponding image, **COCO** dataset and **2M text to image** dataset, and we used only the textual description of those datasets for diverse results. The descriptions were too complex for the model (were long with complex adjectives). To avoid it, we used GPT 4o mini for simplification with prompt engineering method - we described some examples of prompts to the model and it learns to generate simplified prompts like the examples.

"""I have a caption and I want to make it slightly simpler (w.r.t background and colors) and specific, and turn it a caption that describes the same thing If the caption is not appropriate to be converted, respond with “NO”. For example:

1. "pink photo of Tokyo" → "buildings in Tokyo"
2. "Nickelodeon Paw Patrol'Calling All Pups' Soft Potty Seat" → "NO"

do this for the following caption: "{context}".

Format your response as

<simplified>

Figure 6. Prompt engineering method for simplification

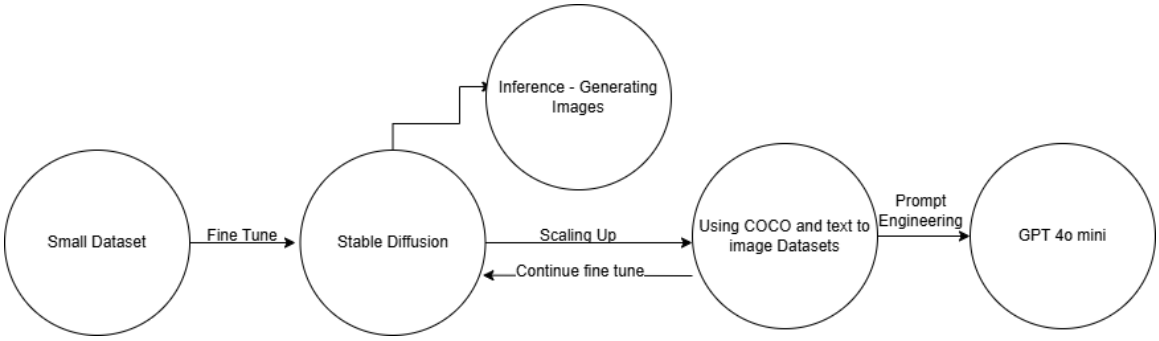


Figure 7. Scaling up process

In the inference time, we filtered the images with **tokens count** of the converted SVG and **CLIP similarity score** threshold. When an image got a CLIP score that above from the threshold, and less than 4K tokens count, the image accepted.

The results weren’t high quality enough. Most of them were good, part of the images wasn’t detailed enough and related to the prompt, part of them were absurd with anomalies. We assumed that the model isn’t strong enough to produce detailed images when the prompt requires a silhouette style.

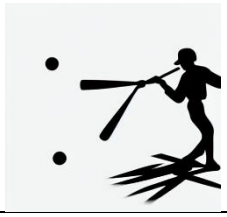
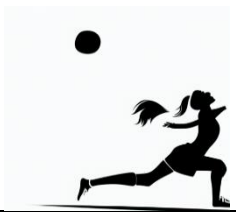
Simplified Prompt (without wrapper)	Generated Image
a girl sitting with presents and a dog on the floor	
a baby on a bed with a laptop	
a batter swinging a bat	
A girl playing volleyball in a gym	

Figure 8. Examples of generated images at different quality levels (from highest quality at the top to lowest at the bottom)

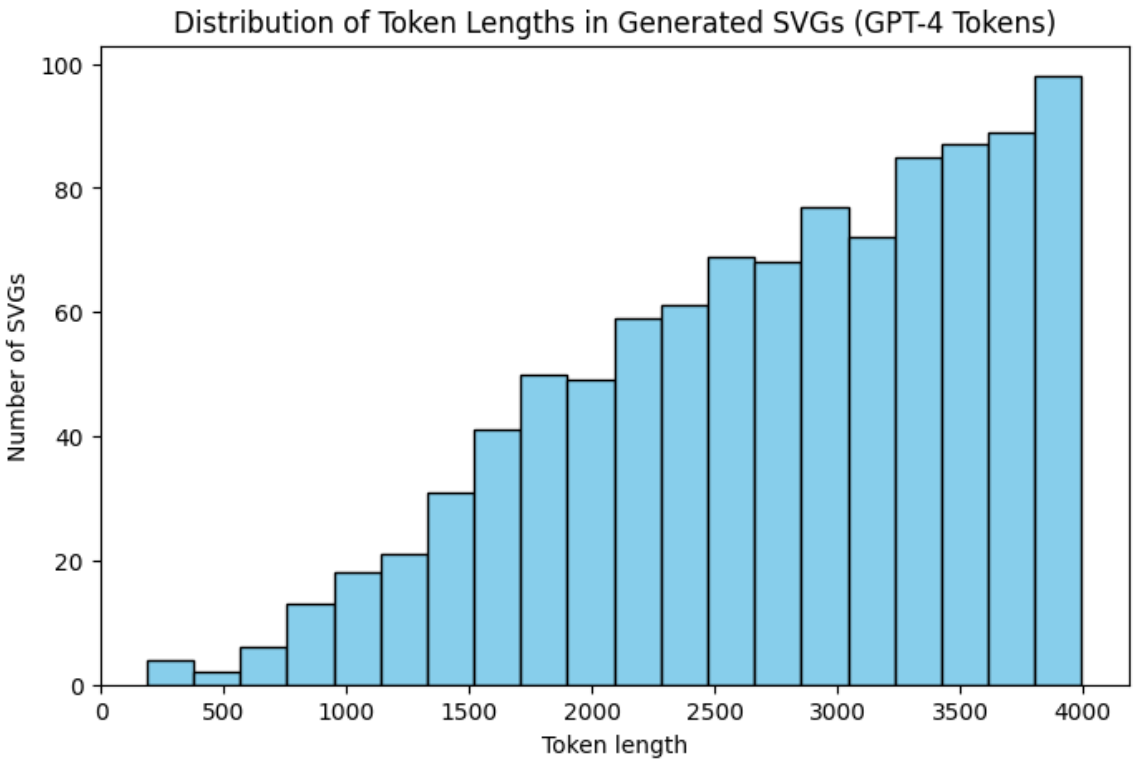


Figure 9. Distribution of token lengths (subset of 1000 images)

5.4 FLUX - Scaling Up to ~1.5M samples

FLUX [10] is a generative model who is stronger and more sophisticated model from stable diffusion and created by Black Forest Lab. From their official blog: "All public FLUX.1 models are based on a hybrid architecture of multimodal and parallel diffusion transformer blocks and scaled to 12B parameters. We improve over previous state-of-the-art diffusion models".

As mentioned above, FLUX produced better results than diffusion models such as Stable Diffusion, so we aimed to use FLUX with prompt engineering to generate more detailed and higher quality images. We tried different prompts for generating images and chose the accurate one. Prompt example: minimal {object} with clean outlines, flat fills, high contrast, scalable vector-style graphic. There is also a negative prompt, which instructs the model not to generate images related to the following concepts: photorealistic, noise, texture, watercolor, gradients, film grain, and shading. We used the FLUX.1-schnell model for inferencing, COCO and 2M text to image datasets (that mentioned before) for prompts. At the end we

successfully generated approximately 1.5M images with FLUX, and new dataset of triplets of text, images and SVG.

Simplified Prompt	Generated Image
a boy wearing headphones looking at a computer	
a woman marking a cake with a knife	

Figure 10. Examples of generated images from FLUX model

6 Fine Tune Llama

Llama 3 [11] is a family generative text models released by Meta. On this project we fine tune Llama 3 8B instruct with LoRa (for a more efficient process), 8B indicates the size of the parameters. The model input is a text and the output is generated text or code only.

The task for the fine tuning was next token prediction - the model predicts the next token in the SVG output. The text input wrapped with special tokens that indicate the start and the end of the input: <|ctx|> and <|objs|> , and the output SVG wrapped with special token that indicates the end of the SVG: <|end_svg|>

It's a self-supervised learning technique, where the model generates implicit labels for unstructured data instead of relying on a labeled dataset. In this setup, the

model learns to predict the next token in the SVG sequence given the previous ones.

We tried fine tune Llama on the FLUX dataset - which yielded good results but most of them consist of more than 8K tokens in the converted SVG. The context window of Llama is 8K tokens. As a result, during the fine-tune process, many tokens of the SVG were cut, which led to the use of an inappropriate SVG, and the fine tune on our dataset failed. To overcome it, we used the OmniSVG dataset.

6.1 Fine Tune Llama on OmniSVG Dataset

OmniSVG dataset divided into illustration and icon, and we used the icon dataset - for simpler samples, and we fine tune Llama model.

Each training sample is a pair of (description, SVG) wrapped with prefix and suffix prompt, for example:

```
Instruction:

Create a valid minimal SVG for: a red apple icon

Response:

<svg ...> ... </svg>
```

In addition, we used different Hugging Face's libraries and filtered by the size of the SVG (tokens count). We used a technique called **gradient accumulation** [12] for our core gradient process. Gradient accumulation is a memory optimization technique that simulates a larger effective batch size by accumulating the gradients calculated from several sequential smaller mini batches before performing a single weight update. This technique was necessary because the model is a heavy one, allowing us to manage GPU memory constraints and achieve the desired large batch size for better convergence.

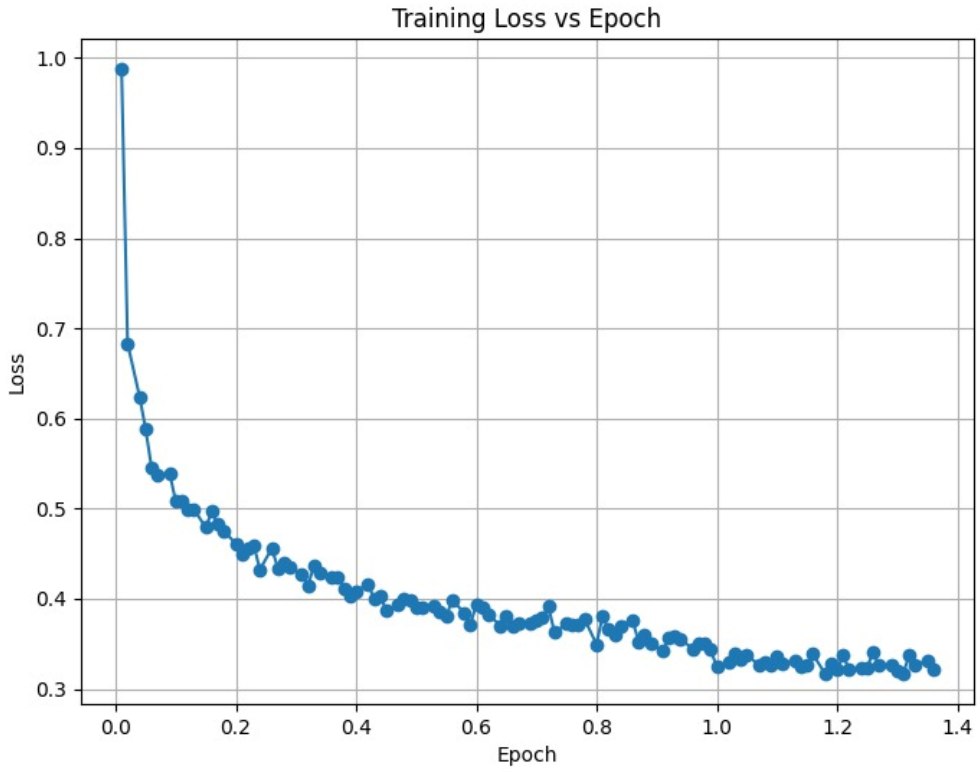


Figure 12. Loss function relative to ephocs during training

After finishing the fine tune process, we inference on the model.

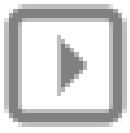
Text	Before Fine Tune	After Fine Tune
A gray folder features a white padlock icon symbolizing security or privacy		
A gray clock icon shows time with minimalist hands pointing at 10:10		

Figure 13. Inference results on the fine tuned LLM - text description and the generated SVG, compared before fine tune.

7 Experiments

To evaluate our model's performance, we performed quantitative analyses.

CLIP similarity score is a text-to-image similarity metric that calculates the cosine similarity between the visual embedding of a generated image and the textual embedding of a caption. Using the CLIP model, which was pretrained on a vast dataset of image-text pairs, this score effectively measures how well an image aligns with a given text by positioning both representations in a shared embedding space.

We converted the SVG outputs to image and then checked the similarity score between the prompt and the converted image.

CLIP Similarity Score	0.3

Aesthetic Score represents a numerical value that captures the visual appeal of an image, often calculated using CLIP-based or other perceptual metrics. We used an Aesthetic Predictor Network, which was trained on human rating. It essentially learned what combinations of visual features humans find aesthetically pleasing. The final number is the predicted aesthetic score based on what humans would rate it.

We converted the SVG outputs to image and then checked the Aesthetic score.

Aesthetic Score	4.57

The quantitative evaluation of the generated SVGs performed using two metrics: the CLIP similarity score and the Aesthetic score. Each measures a different aspect of performance - semantic alignment with the textual input and visual quality of the generated output, respectively.

The achieved CLIP similarity score of 0.3 suggests that the relationship between the textual descriptions and the generated SVGs is limited. While the model able to capture general object categories (like “triangle”, “circle”, “arrow”), it often failed to reproduce the specific semantic details of the prompt, such as color attributes, or contextual relationships between objects.

The low score can be explained by many reasons: first, the generated SVG were most of them simple, and the CLIP model trained on raster images with complex

texture and color. Therefore, CLIP may underestimate minimalist SVG images even when they are structurally correct.

Second, the model fine tuned in a self supervised technique without visual feedback, relying on token prediction. As a result, the model learns the syntactic structure of SVG sequences but lacks the ability to assess the visual correctness of the rendered output.

The Aesthetic score of 4.57 shows that even though the generated SVGs didn't always match the text meaning correct, they still looked good visually. This score comes from a model that predicts how appealing an image looks to people, based on things like composition, balance, and overall visual order. A value above 4 on a 1-10 scale means the images were clear, and nicely structured.

8 Conclusion

The primary objective of this study was to investigate the feasibility of generating SVG directly from text prompts using an LLM. To achieve it, we first wanted to create a new dataset consisting of text, image and their corresponding SVG. The raster images were generated in a specific style suitable for conversion to vector format, ensuring consistency and clarity in the resulting SVG files. Various methods were explored to build this dataset, including fine-tuning Stable Diffusion model and utilizing the FLUX generative model to generate images aligned with the textual prompts. At the end, we generated ~1.5M triplets with FLUX.

During our experiments with fine-tuning, we found that training the LLM directly on our dataset didn't work well. This was mostly because the model's limited context window. Then we decided to switch using the OmniSVG dataset for fine-tuning.

The quantitative results, with an average CLIP score of 0.3 and an Aesthetic score of 4.57, indicate that the model generated visually coherent SVGs but with limited semantic alignment to the input text.

Overall, this study shows that generating SVGs directly from text using an LLM is possible, but there are still some limitations that can be improved in future work.

9 Future Work

To improve fine-tuning results on the LLM, we plan to experiment with chunking the SVG outputs. Instead of generating the entire SVG in one long sequence, the model will produce it in smaller, structured parts, for example, by splitting it into separate “scenes” that can later be combined into a complete SVG. This iterative approach could help the model better handle long sequences and capture more detailed relationships between text and vector structure.

In addition, we aim to build on the ideas presented in the OmniSVG paper. One of the main limitations of their approach is that, during inference, the model needs to generate tens of thousands of tokens for complex samples, which makes the generation process slow. The authors suggest that using multi-token prediction and key-value cache compression could help reduce the generation cost and improve efficiency. In our future work, we plan to explore these directions and test whether such techniques can also enhance text-to-SVG generation performance.

References

Our repository: https://github.com/MaayanMashhadi/Thesis

1. Ayush Jain, Angjoo Xie, and Pieter Abbeel. “VectorFusion: Text-to-SVG by Abstracting Pixel-Based Diffusion Models”. In: arXiv preprint arXiv:2306.15613 (2023).
2. Ximing Xing et al. “SVGDreamer: Text guided SVG generation with diffusion model”. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2024, pp. 4546–4555
3. Ximing Xing et al. “Diffsketcher: Text guided vector sketch synthesis through latent diffusion models”. In: Advances in Neural Information Processing Systems 36 (2023), pp. 15869–15889.
4. Yang, Y., Cheng, W., Chen, S., Zeng, X., Zhang, J., Wang, L., … Jiang, Y.-G. (2025). OmniSVG: A Unified Scalable Vector Graphics Generation Model. arXiv Preprint Arxiv:2504. 06263.
5. 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
6. Martin Ester et al. “A density-based algorithm for discovering clusters in large spatial databases with noise”. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining. KDD’96. Portland, Oregon: AAAI Press, 1996.
   url: https://www.dbs.ifi.lmu.de/Publikationen/Papers/KDD-96.final.frame.pd
7. Peter Selinger. “Potrace : a polygon-based tracing algorithm”. In: 2003. url: https://api.semanticscholar.org/CorpusID:1419652
8. Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller,
   Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell,
   Tim Dockhorn, Zion English, Kyle Lacey, Alex Goodwin, Yannik Marek, and Robin
   Rombach. 2024. Scaling Rectified Flow Transformers for High-Resolution Image
   Synthesis. arXiv:2403.03206 [cs.CV] https://arxiv.org/abs/2403.03206

9. Liashchynskyi, P., & Liashchynskyi, P. (2019). Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS. *CoRR*, *abs/1912.06059*. Retrieved from http://arxiv.org/abs/1912.06059

10. 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 [Cs.GR]*. Retrieved from http://arxiv.org/abs/2506.15742
11. 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/
12. [Scaling Deep Contrastive Learning Batch Size under Memory Limited Setup](https://aclanthology.org/2021.repl4nlp-1.31/) (Gao et al., RepL4NLP 2021)
13. Our poster: https://systor25posters.hotcrp.com/doc/systor25posters-paper10-poster_pdf.pdf?cap=hcav10QrKeTGfpnnYjBtcmQHXiSDyB