University of Modena and Reggio Emilia

DEPARTMENT OF ENGINEERING “ENZO FERRARI”
Master’s Degree in Artificial Intelligence Engineering

LLM-based SVG image captioning with conceptual embeddings

Candidate:
Emanuele Di Luzio

Supervisor:
Prof. Lorenzo Baraldi

Co-supervisor:
Dott. Leonardo Zini

Academic Year 2024-2025

Contents

Abstract

Scalable Vector Graphics (SVG) are ubiquitous in modern web design, yet current Multimodal Large Language Models (MLLMs) struggle to interpret their raw XML structure. Most existing approaches rely on rasterizing SVGs into pixel grids, a process that discards the semantic richness of the vector definition and introduces resolution artifacts.

In this work, I present a novel architecture that enables a decoder-only LLM to directly ingest and interpret SVG code without rasterization. I integrate a **SVG Path Embedder (SPE)**–originally developed by Zini *et al.*–that maps continuous geometric coordinates into the LLM's embedding space using sinusoidal functions. By combining this encoder with **Qwen2-7B** and **Gemma-9B** via **Low-Rank Adaptation (LoRA)**, I demonstrate that the model can learn to "read" vector geometries effectively.

My experiments, conducted on a dataset of **90,000 SVG-caption pairs** and evaluated on a stratified benchmark of **400 samples**, show that this vector-native approach outperforms zero-shot raster baselines. Specifically, the **SPE+Qwen2-7B** configuration achieves a CLIPScore of 29.3 and a BLEU-1 score of 0.42, offering a parameter-efficient alternative to vision-encoder-based methods. I provide a detailed analysis of the model's capabilities and limitations, offering a new perspective on vector-language understanding.

Sommario

Le immagini vettoriali (SVG) sono onnipresenti nel design digitale, tuttavia gli attuali modelli multimodali (MLLM) faticano a interpretare la loro struttura XML grezza. La maggior parte degli approcci esistenti si basa sulla rasterizzazione degli SVG in griglie di pixel, un processo che scarta la ricchezza semantica della definizione vettoriale e introduce artefatti di risoluzione.

In questa tesi, presento un'architettura che consente a un LLM decoder-only di ingerire e interpretare direttamente il codice SVG senza rasterizzazione. Integro uno **SVG Path Embedder (SPE)**–originariamente sviluppato da Zini *et al.*– che mappa le coordinate geometriche continue nello spazio di embedding dell'LLM utilizzando funzioni sinusoidali. Combinando questo encoder con **Qwen2-7B** e **Gemma-9B** tramite **Low-Rank Adaptation (LoRA)**, dimostro che il modello può imparare a "leggere" le geometrie vettoriali in modo efficace.

I miei esperimenti, condotti su un dataset di **90.000 coppie SVG-caption** e valutati su un benchmark stratificato di **400 campioni**, mostrano che questo approccio vettoriale supera le baseline raster zero-shot. Nello specifico, la configurazione **SPE+Qwen2-7B** ottiene un CLIPScore di 29.3 e un punteggio BLEU-1 di 0.42, offrendo un'alternativa efficiente ai metodi basati su vision-encoder. Fornisco un'analisi dettagliata delle capacità e dei limiti del modello, offrendo una nuova prospettiva sulla comprensione del linguaggio vettoriale.

Chapter 1

Introduction

The interpretation of visual information by artificial intelligence has undergone a remarkable transformation over the past decade. From early Convolutional Neural Networks (CNNs) designed to classify static images, to modern Vision Transformers (ViTs) capable of understanding complex scenes, the field has consistently pushed the boundaries of what machines can "see" and comprehend. Yet, despite these advances, a fundamental assumption has remained largely unchallenged: that visual data consists of pixel grids–discrete arrays of color intensity values.

This pixel-centric paradigm, while successful for photographs and natural imagery, overlooks a critical dimension of modern digital content. A substantial portion of the visual material I encounter daily–icons, logos, technical diagrams, user interface elements, typographic designs–is not created from pixels at all. Instead, these graphics are defined as **Scalable Vector Graphics (SVG)**, mathematical descriptions of shapes, curves, and paths that are inherently resolution-independent and semantically structured.

Consider a simple icon: a magnifying glass representing "search" functionality on a website. To a human designer, this icon is not a collection of colored dots, but a composition of geometric primitives–a circle for the lens, a line for the handle, perhaps a smaller circle to suggest a reflection. The icon is encoded as a sequence of XML instructions that describe these shapes precisely:

- `<circle cx="45" cy="45" r="35"/>` defines the lens;
- `<line x1="70" y1="70" x2="100" y2="100"/>` defines the handle.

This representation is compact, editable, and infinitely scalable; zooming in never introduces pixelation or blur. However, when state-of-the-art Multimodal Large Language Models (MLLMs) — systems like GPT-4 with Vision (GPT-4V), LLaVA,

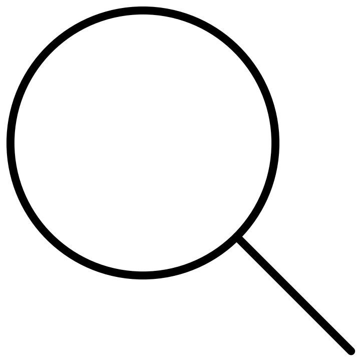


Figure 1.1: Visual representation of the simple SVG magnifying glass defined by the XML code.

or BLIP-2 — are asked to describe such an icon, they cannot read the vector definition directly. Instead, the standard practice is to **rasterize** the SVG: render it to a fixed-resolution bitmap (*e.g.*, 512×512 pixels), then feed this raster image to a vision encoder. This approach has two significant drawbacks. First, it discards the rich semantic structure embedded in the SVG markup–the hierarchy of groups, the annotations in `id` and `class` attributes, the explicit geometric relationships between shapes. Second, it introduces aliasing and discretization artifacts that can obscure fine details, especially for icons with thin lines or small features.

This thesis challenges the prevailing pixel-based paradigm by posing a simple question: *Can I teach a language model to "read" vector graphics natively, without converting them to raster images first?* The motivation for this question is both practical and philosophical. On the practical side, there is an urgent need for automatic captioning systems that can generate accurate textual descriptions of digital graphics. On the philosophical side, treating SVGs as a *language* aligns naturally with the strengths of Large Language Models (LLMs).

1.1 The Challenge of Vector Graphics Captioning

To appreciate the difficulty of the task, I must first understand what makes SVG fundamentally different from raster images. An SVG file is an XML document conforming to the W3C Scalable Vector Graphics specification [32]. Unlike a JPEG or PNG, which stores a fixed grid of pixel values, an SVG stores a *hierarchical DOM tree*–a structured set of graphical elements. Each element (*e.g.*, `<path>`, `<circle>`, `<rect>`) is defined by a set of attributes: geometric parameters (coordinates, radii, control points), style properties (fill color, stroke width, opacity), and possibly metadata (titles, descriptions, ARIA labels).

The flexibility of this representation is a double-edged sword. On one hand, it enables powerful features like infinite zoom, easy editing, and style inheritance. On the other hand, it introduces significant variability. The same visual icon can be encoded in countless ways: paths can be absolute or relative, clockwise or counter-clockwise; shapes can be grouped or flattened; transformations (translate, rotate, scale) can be applied at any level of the hierarchy. Furthermore, the coordinate values are *continuous*–floating-point numbers that specify precise locations on a two-dimensional canvas. This continuous nature poses a challenge for standard text tokenization schemes, which typically operate on discrete vocabularies.

Moreover, the "vocabulary" of SVG includes geometric concepts that do not map straightforwardly to the semantic categories learned by vision-language models trained on natural images. A curve defined by a cubic Bézier path has no direct pixel-level analogue; it is a parametric function. Understanding that a specific sequence of path commands draws, say, a magnifying glass, requires integrating the geometric information across multiple elements and inferring the high-level semantic category from the emergent shape.

This gap between geometry and semantics is precisely where existing Multimodal Large Language Models (MLLMs) struggle. Standard vision encoders–whether ResNet-based or Vision Transformer-based–are designed to extract features from dense pixel grids. They excel at recognizing textures, edges, and spatial patterns in natural photos, but they have no mechanism to directly ingest the symbolic, structured representation of an SVG. Consequently, the common workaround is rasterization: render the SVG to a pixel image and process it as if it were a photograph. This approach, while expedient, is fundamentally mismatched to the data modality. It is analogous to asking someone to describe a written sentence by first printing it, photographing the page, and then performing OCR–technically feasible, but wasteful and error-prone.

1.2 Accessibility, Search, and the Need for Automatic Captions

The practical importance of SVG captioning extends far beyond academic curiosity. In the context of web accessibility, the World Wide Web Consortium (W3C) Web Content Accessibility Guidelines (WCAG) mandate that all non-text content be accompanied by alternative text (alt-text) that conveys the same information. For images, this is typically a short caption describing the visual content. However, many websites, design systems, and icon libraries contain thousands of SVG icons, and manually writing captions for each is a labor-intensive and often neglected task.

Consider a large design system like Google's Material Icons or Font Awesome, which provide hundreds of icons for user interface elements. Each icon needs a descriptive label not only for accessibility compliance but also for searchability. A designer looking for a "notification bell" icon should be able to search for that phrase and find the relevant SVG. Similarly, screen reader users navigating a web page should hear "notification bell icon" rather than the generic fallback "image" or, worse, the cryptic SVG filename "icon_287.svg".

Current practice often relies on manually assigned tags or filenames, which are inconsistent and prone to error. An automatic captioning system that can generate accurate, human-readable descriptions directly from the vector definition would significantly streamline the design workflow, improve accessibility, and enable more sophisticated search and retrieval applications (*e.g.*, semantic similarity search across icon libraries). Furthermore, as vector graphics are increasingly used in domains beyond traditional iconography–technical diagrams, data visualizations, UI mockups, schematic drawings–the ability to automatically annotate and describe these graphics becomes even more valuable. For instance, in educational contexts, automatically generating descriptions of geometric diagrams could aid visually impaired students. In software engineering, automatically documenting UI components from their SVG prototypes could improve code documentation and design handoff.

1.3 Research Hypothesis: Treating SVG as a Structured Language

The central hypothesis of this thesis is that SVG captioning is best approached not as a vision problem (processing pixels), but as a **structured language understanding problem**. Since an SVG is fundamentally a symbolic representation–a sequence of XML tags and numeric coordinates–it lies closer to the domain of text than to the domain of natural images. This observation suggests that Large Language Models (LLMs), which are inherently designed to process and generate sequences of symbols, might be well-suited to this task.

However, a naive approach — simply feeding the raw SVG XML code as a text string to an LLM — fails in practice. The coordinate values are verbose (*e.g.*, `d="M12.5,34.2..."`) and the numeric precision is lost during text tokenization. Early experiments with zero-shot prompting produced generic or hallucinatory captions.

To address this, I first investigate a **text-only fine-tuning baseline**, where the LLM is adapted to process linearized SVG code as a standard language modeling task. While this improves syntactic understanding, it remains limited in grasping complex geometries. Therefore, I propose a hybrid approach that bridges the gap between the continuous geometric domain and the discrete symbolic domain of language. Specifically, I integrate a **SVG Path Embedder (SPE)**–a neural module developed by Zini *et al.* [37]–that maps the continuous (x, y) coordinates of SVG elements into a high-dimensional embedding space compatible with the LLM's input representation. This encoder does not operate in isolation; it must be integrated with a powerful language generation component that can transform the geometric features into fluent natural language, which is where Large Language Models enter the picture.

1.4 The Role of Large Language Models in the Proposed Approach

Why use Large Language Models at all for this task? The answer lies in their remarkable capacity for semantic understanding and text generation. LLMs like GPT-4, Llama, Qwen, and Gemma are pre-trained on vast corpora of text, giving them extensive world knowledge and the ability to generate fluent, contextually appropriate natural language. Crucially, these models are *decoder-only* architectures, meaning they generate text autoregressively–one token at a time–conditioned on a prefix of input tokens.

In my architecture, the SPE produces a sequence of visual embeddings corresponding to the geometric elements of the SVG. These embeddings are then prepended (or interleaved) with the LLM's text token embeddings, and the model is tasked with generating a caption autoregressively. The LLM's pre-trained linguistic knowledge allows it to map the geometric features to semantic categories (*e.g.*, recognizing that a circle with a line forms a "magnifying glass"), and to generate grammatically correct, human-readable descriptions. However, off-the-shelf LLMs are not trained to interpret geometric embeddings. Their input space consists of text tokens (words, subwords, characters), not continuous vector representations of shapes. Therefore, I must *adapt* the LLM to this new modality. This adaptation is achieved through fine-tuning: I train the model on a dataset of paired SVG-caption examples, teaching it to associate the geometric embeddings produced by the SPE with the corresponding textual descriptions.

1.5 Parameter-Efficient Fine-Tuning with LoRA

Fine-tuning a modern LLM with billions of parameters is computationally expensive and risks *catastrophic forgetting*–the model may overfit to the new task and lose its general linguistic capabilities. Full fine-tuning also requires storing a separate copy of the entire model for each task, which is impractical when deploying multiple specialized systems.

To mitigate these issues, I employ **Parameter-Efficient Fine-Tuning (PEFT)**, specifically **Low-Rank Adaptation (LoRA)** [12]. LoRA freezes the pre-trained weights of the LLM and introduces small, trainable low-rank matrices that modify the model's behavior. This drastically reduces the number of trainable parameters (from billions to millions) and accelerates training, while preserving the core linguistic knowledge of the base model. In my experiments, I apply LoRA to the attention projection matrices (query, key, value, and output projections) of the Transformer layers. During training, only the LoRA matrices and the SPE parameters are updated; the LLM's pre-trained weights remain frozen. This not only makes training feasible on limited hardware (*e.g.*, a single or pair of GPUs), but also ensures that the adapted model retains its ability to generate fluent, grammatically correct captions.

1.6 The SVG Path Embedder: Bridging Geometry and Language

With the LLM and the fine-tuning strategy established, I arrive at the final piece of the puzzle: the SVG Path Embedder (SPE). The SPE is a custom neural module designed to convert the continuous geometric properties of SVG elements into dense vector representations that the LLM can process. I provide a high-level preview here; the full technical details, including design alternatives and ablation studies, are developed in Chapter 3.

The core challenge is encoding continuous coordinates. Unlike discrete token indices in text (which have a natural ordering and are easily represented as one-hot vectors or learned embeddings), (x, y) coordinates are real-valued and do not fit naturally into the vocabulary of a language model. Naive discretization (*e.g.*, rounding to the nearest integer or binning into fixed ranges) introduces quantization artifacts and limits the model's ability to perceive fine geometric details.

Instead, the SPE employs a technique inspired by the sinusoidal positional encodings used in the original Transformer architecture and by Fourier feature mappings in Neural Radiance Fields (NeRF). The core idea is to project each coordinate value through a bank of sinusoidal functions with geometrically increasing frequencies:

$$\mathrm{PE}(x) = [\sin(\omega_1 x), \cos(\omega_1 x), \sin(\omega_2 x), \cos(\omega_2 x), \ldots, \sin(\omega_k x), \cos(\omega_k x)]^T$$

where $\omega_k = 10000^{2k/d}$ for $k = 0, 1, \ldots, d/2 - 1$. This projection maps each scalar coordinate to a high-dimensional vector. The lower frequencies capture coarse, global position information ("is this shape in the top-left or bottom-right of the canvas?"), while the higher frequencies encode fine-grained details ("is this corner sharp or slightly rounded?"). By concatenating these positional encodings with learnable embeddings for element type (circle, path, rectangle, etc.) and style attributes (fill, stroke, etc.), and passing the result through a multi-layer perceptron (MLP), the SPE produces a unified embedding for each SVG element. These embeddings are then linearly projected to match the dimensionality of the LLM's input space and normalized to ensure numerical stability during training.

1.7 Research Objectives

The primary objective of this thesis is to develop and validate a system capable of generating accurate, human-readable textual captions for SVG graphics by processing their vector definition directly, without rasterization. This broad goal is decomposed into four specific research objectives: (i) to **develop and validate a Baseline Training Process (LoRA-LLM)**, implementing a text-only fine-tuning pipeline where the SVG code is treated as a raw string to establish a strong baseline; (ii) to **develop and validate a Multimodal Training Process (SPE-LoRA)**, integrating the **SVG Path Embedder (SPE)** with decoder-only LLMs via parameter-efficient fine-tuning to enable true geometric understanding; (iii) to **curate a rigorous evaluation benchmark** consisting of a diverse, stratified test set of SVG-caption pairs; and (iv) to **conduct quantitative and qualitative analysis** of the model's performance, identifying strengths and failure modes.

1.8 Summary of Methodology and Key Findings

This thesis introduces a vector-native SVG captioning pipeline centered on the SVG Path Embedder (SPE), which maps geometric and structural properties of SVG elements into a continuous latent space compatible with Large Language Models. Unlike traditional raster-based approaches (BLIP-2, Florence-2), which render SVGs to pixel grids and process them with vision encoders, my method preserves the vector nature of the data and leverages the symbolic reasoning capabilities of LLMs.

The core methodology involves three stages: (1) SVG preprocessing and linearization, where the XML scene graph is traversed and path commands are extracted and normalized; (2) LoRA-based fine-tuning of the LLM, initially established as a text-only baseline to adapt the model to SVG syntax; and (3) geometric embedding via the SPE, which projects continuous coordinates through sinusoidal features to enable the full multimodal approach.

Experimentally, I evaluate the system on a dataset of 90,000 SVG-caption pairs from Icons8, with a stratified test set of 400 samples. I compare zero-shot raster baselines (BLIP-2, Florence-2), the **Baseline Training Process (LoRA-LLM)** (text-only fine-tuning on SVG code), and the **Multimodal Training Process (SPE-LoRA)** (my proposed vector-native approach). The results demonstrate that the **SPE+Qwen2-7B** configuration achieves the highest composite score (**8.89**, calculated as $\text{CLIPScore}/10 + \text{BLEU} + \text{METEOR} + \text{ROUGE}$), outperforming both zero-shot baselines and the text-only baseline in linguistic quality metrics (BLEU-1: **0.420**, METEOR: **0.380**, ROUGE-L: **0.450**), while maintaining competitive visual-semantic alignment (CLIPScore: **29.30**).

Qualitative analysis reveals that the model excels at describing geometric primitives and capturing spatial relationships in well-defined icon categories (User Interface, Arrows, Weather), but struggles with abstract shapes, color attribution (when style features are not encoded), and text recognition in path-based glyphs. These findings highlight the importance of feature normalization in the SPE pipeline and suggest directions for future work, including hierarchical encoding of SVG group structures and integration of style attributes.

1.9 Thesis Organization

The remainder of this manuscript is organized as follows. **Chapter 2** provides a comprehensive survey of the theoretical foundations and related work, moving from deep learning fundamentals to vision-language models and the specific technologies underpinning this thesis. **Chapter 3** presents the proposed methodology in detail, describing the design rationale for the SVG Path Embedder, the LoRA-based fine-tuning strategy, and the geometric intuition behind the architecture. **Chapter 4** documents the implementation, covering the software architecture, data loading pipeline, and training workflow. **Chapter 5** presents the experimental evaluation, reporting quantitative results for all model configurations and providing a detailed qualitative analysis of case studies and failure modes. Finally, **Chapter 6** synthesizes the findings, discusses limitations and ethical considerations, and outlines promising directions for future research. In the following chapters, I develop each of these themes in depth, building a rigorous and comprehensive account of the SVG captioning problem, my proposed solution, and the empirical evidence supporting its effectiveness.

Chapter 2

State of the Art

This chapter surveys the theoretical foundations and prior research underpinning the SVG captioning system. I move from general deep learning principles to specific technologies employed in my architecture, critically assessing their relevance to vector graphics understanding. The chapter begins with Transformer fundamentals (Section 2.1), traces the evolution of Large Language Models (Section 2.2), examines vision-language models and their limitations for vector data (Section 2.3), explores positional encodings as the foundation for my SPE design (Section 2.4), compares parameter-efficient fine-tuning methods (Section 2.5), and reviews prior work on SVG generation (Section 2.6).

2.1 Foundations: Transformers and Self-Attention

2.1.1 The Transformer Architecture

The Transformer [31] represented a paradigm shift in deep learning, replacing recurrent architectures with **self-attention** to enable parallel processing and efficient modeling of long-range dependencies. Given an input sequence of N tokens represented as d-dimensional vectors, the mechanism constructs three matrices via learned linear projections: $Q = XW_Q$ (queries), $K = XW_K$ (keys), and $V = XW_V$ (values). Attention computes weighted sums:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Intuitively, the product QK^T measures pairwise token similarities, the softmax normalizes these to probabilities, and the multiplication by V aggregates relevant information. This mechanism is permutation-invariant, necessitating the use of positional encodings (discussed in Section 2.4) to retain sequence order.

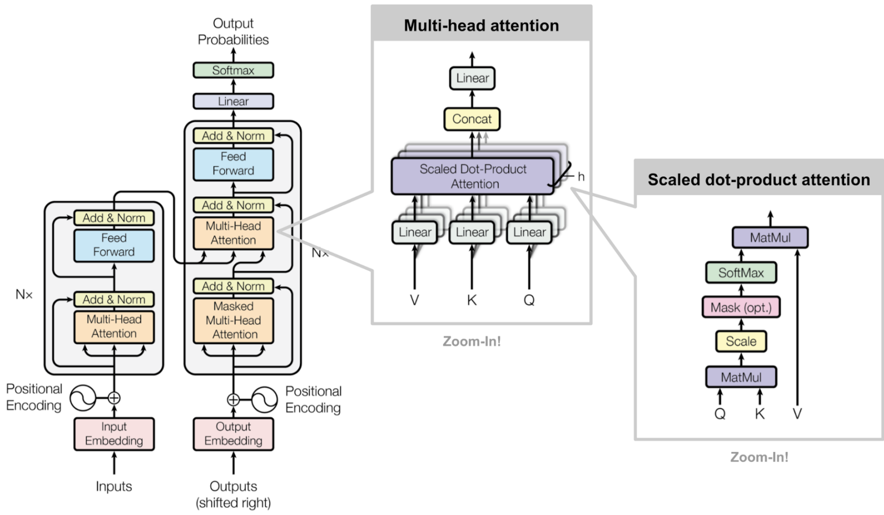


Figure 2.1: The Transformer model architecture. Left: Encoder. Right: Decoder. (Source: Vaswani *et al.*, 2017)

To capture different aspects of the input simultaneously, the architecture employs **Multi-head attention**, which projects the input into h subspaces:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W_O$$

Different heads can thus focus on distinct patterns, such as short-range syntactic dependencies or long-range semantic relationships (*e.g.*, matching XML tags). Each Transformer layer combines this multi-head attention with a position-wise feed-forward network (FFN), defined as $\text{FFN}(x) = \text{GELU}(xW_1 + b_1)W_2 + b_2$, and wraps these sub-layers with residual connections and layer normalization: $\text{Output} = \text{LayerNorm}(x + \text{SubLayer}(x))$.

Modern LLMs (Qwen2, Gemma) use **RMSNorm** [36], which omits mean centering for efficiency:

$$\text{RMSNorm}(x) = \frac{x}{\text{RMS}(x)} \cdot \gamma, \quad \text{RMS}(x) = \sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}$$

2.1.2 SVG and the Tokenization Challenge

SVG poses a unique challenge: while syntactically XML, its semantics are encoded in *numeric coordinates*. A text tokenizer splits `M 12.5 34.7 L 56.2 89.3` into `[M, 12, ., 5, 34, ., 7, ...]`, destroying coordinate unity and losing numerical continuity. A minor change (12.5 → 12.6) drastically alters the rendered shape, yet the tokenizer treats these as independent tokens. This fundamental mismatch motivates the design of the SVG Path Embedder–a specialized encoder preserving the continuous geometric nature of the data, with its theoretical foundation in Fourier-based positional encodings extended to 2D spatial data (Section 2.4).

2.2 Large Language Models: Evolution to Decoder-Only Architectures

2.2.1 From BERT to GPT: Encoder vs. Decoder

The fundamental divergence in Transformer architectures stems from their training objectives and masking strategies, which dictate their suitability for different classes of tasks.

BERT (Bidirectional Encoder Representations from Transformers) [6] employs a **Masked Language Modeling (MLM)** objective. In this paradigm, approximately 15% of the input tokens are randomly replaced with a special `[MASK]` token (80%), a random token (10%), or left unchanged (10%). The model trains to predict the original identity of these masked tokens based on the bidirectional context–that is, attending to both preceding and succeeding tokens simultaneously. Mathematically, the attention mechanism in BERT is unconstrained: the attention mask M is a matrix of zeros, allowing

$$A_{ij} = \text{softmax}\left(\frac{Q_i K_j^T}{\sqrt{d_k}}\right)$$

to be non-zero for all pairs (i, j) in the full sequence length T. This architecture builds deep, context-aware representations ideal for *Natural Language Understanding (NLU)* tasks such as classification, named entity recognition, and question answering, where the entire input is available at once. However, this reliance on bidirectional context makes it inherently unsuitable for autoregressive generation, as the generation process is fundamentally inhibited by the look-ahead bias inherent in bidirectional models.

Conversely, **GPT (Generative Pre-trained Transformer)** [4,25] is designed for **Causal Language Modeling (CLM)**. It utilizes a decoder-only architecture where self-attention is constrained by a causal mask M, defined as:

$$M_{ij} = \begin{cases} -\infty & \text{if } j > i \\ 0 & \text{otherwise} \end{cases}$$

This ensures that the prediction of token x_t depends exclusively on the history $x_{<t}$. The joint probability of a sequence is factorized as:

$$P(x_1, \ldots, x_T) = \prod_{t=1}^{T} P(x_t \mid x_{<t})$$

This factorization aligns perfectly with the generation process. While early GPT models lagged behind BERT in NLU benchmarks, scaling laws [15] revealed that sufficiently large decoder-only models (starting with GPT-3) develop emergent capabilities–such as **In-Context Learning** and few-shot reasoning–that rival or surpass specialized encoder models. Unlike BERT, which typically requires fine-tuning a specific classification head for each downstream task, GPT models can perform tasks essentially "zero-shot" by continuing a prompt. For SVG captioning, this generative capability is paramount, as we must produce a sequential textual description from a visual prefix.

2.2.2 Modern Decoder-Only LLMs

While the original GPT architecture laid the foundation, modern open-weights LLMs have introduced critical architectural refinements to improve stability, efficiency, and context handling. Three state-of-the-art backbones are utilized in this thesis, each embodying specific design philosophies:

- **Llama (Meta)** [30]: Llama 3 maintains the robust design choices that characterize the Llama family. A key feature is the use of **Rotary Positional Embeddings (RoPE)** [29] instead of absolute positional encodings. RoPE encodes position by rotating the query and key vectors in the complex plane:

  $$f_{q,k}(x_m, m) = R^d_{\Theta,m} x_m$$

  This formulation allows the attention score to depend only on the relative distance $m - n$, improving generalization to sequence lengths exceeding the training window–essential for processing verbose SVG code. Additionally, it adopts **SwiGLU** activations [27] in the feed-forward networks, replacing ReLU with a gated mechanism $\text{Swish}(xW) \otimes (xV)$, which offers smoother gradients and faster convergence.

- **Qwen2 (Alibaba)** [2]: The Qwen2-7B model is optimized for efficient long-context inference. It employs **Grouped-Query Attention (GQA)** [1], which serves as an effective trade-off between the speed of Multi-Query Attention

(MQA) and the quality of Multi-Head Attention (MHA). In GQA, multiple query heads share a single key-value head (*e.g.*, 8 KV heads for 64 Q heads). This drastically reduces the size of the KV cache and the memory bandwidth required during decoding, enabling the processing of context windows up to 128k tokens with minimal performance degradation.

- **Gemma (Google)** [8]: Built on Gemini technology, Gemma focuses on numerical stability at scale. It places RMSNorm *after* the embedding layer and before each transformer block (Pre-Norm with offset), preventing gradient explosions in deep networks. It also utilizes **GeGLU** activations (a Gaussian Error Linear Unit variant of GLU), which provides a more complex non-linear mapping than GeLU, enhancing the model's capacity to learn subtle patterns in data.

Scaling Laws and Emergent Capabilities. Optimizing these architectures requires adhering to empirical scaling laws. While Kaplan *et al.* [15] initially suggested a power-law relationship favoring model size, the "Chinchilla" scaling laws [10] revised this, demonstrating that optimal performance for a given compute budget is achieved by scaling model parameters (N) and training tokens (D) in equal proportion ($N \approx D$). This insight drives the recent trend of highly capable "smaller" models (7B-9B parameters) trained on massive datasets (trillions of tokens). Importantly, as these models scale in accordance with these laws, they acquire *emergent capabilities*–qualitative leaps in performance not predictable from smaller scales. For our specific multimodal task, the relevant emergence is the ability to map abstract vector structures (encoded by the SPE) to semantic linguistic concepts without requiring a dedicated cross-attention mechanism, effectively treating the SVG sequence as a native foreign language.

2.3 Vision-Language Models and Their Limitations

2.3.1 Vision Transformers (ViT)

The Vision Transformer (ViT) [7] marked a paradigm shift in computer vision, demonstrating that the pure Transformer architecture–without any convolutional inductive biases–could achieve state-of-the-art performance on image recognition tasks. ViT processes an image by dividing it into a grid of fixed-size patches (*e.g.*, 16×16 pixels). Each patch is flattened into a 1D vector and linearly projected to the model's embedding dimension D. To retain spatial information, learnable position embeddings are added to these patch embeddings. A special learnable `[CLS]` token is prepended to the sequence, whose state at the output of the Transformer encoder serves as the image representation.

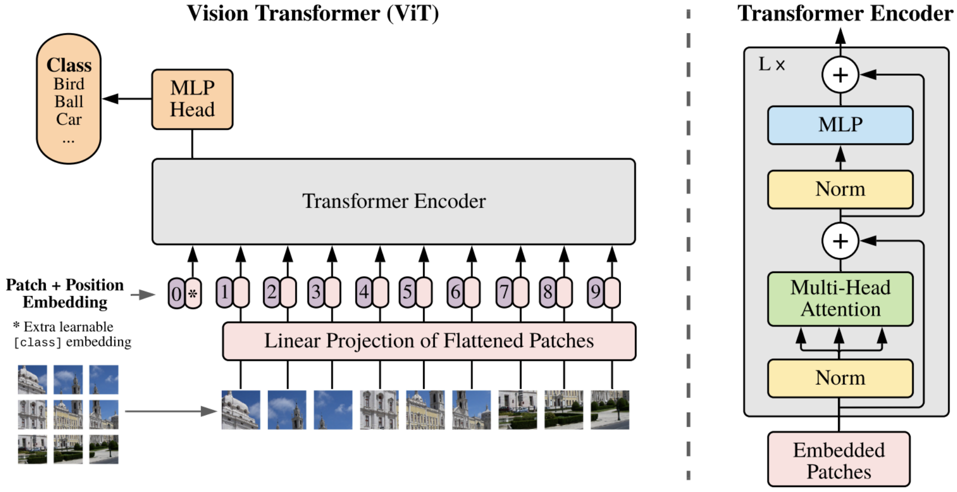


Figure 2.2: The Vision Transformer (ViT) Architecture. The image is split into patches, linearly projected, and processed by a standard Transformer encoder. (Source: Dosovitskiy *et al.*, 2020)

Mathematically, if an input image $x \in \mathbb{R}^{H \times W \times C}$ is split into N patches $x_p \in \mathbb{R}^{N \times (P^2 \cdot C)}$, the input sequence z_0 is:

$$z_0 = [x_{class}; x_p^1 E; x_p^2 E; \ldots; x_p^N E] + E_{pos}$$

where $E \in \mathbb{R}^{(P^2 \cdot C) \times D}$ is the projection matrix and $E_{pos} \in \mathbb{R}^{(N+1) \times D}$ are the position embeddings.

2.3.2 CLIP: Contrastive Language-Image Pre-training

CLIP [24] builds on ViT to learn visual representations from natural language supervision. Instead of training on fixed classes (like ImageNet), CLIP is trained on a massive dataset of 400 million image-text pairs collected from the internet. The core mechanism is **contrastive learning**. CLIP trains two encoders jointly: a vision encoder (ViT or ResNet) and a text encoder (Transformer). The goal is to maximize the cosine similarity between the embeddings of matched image-text pairs in a batch, while minimizing the similarity for unmatched pairs.

This objective aligns the visual and textual latent spaces, enabling **zero-shot classification**: to classify an image, the model compares its embedding with the embeddings of text prompts like "a photo of a dog", "a photo of a cat", etc., and

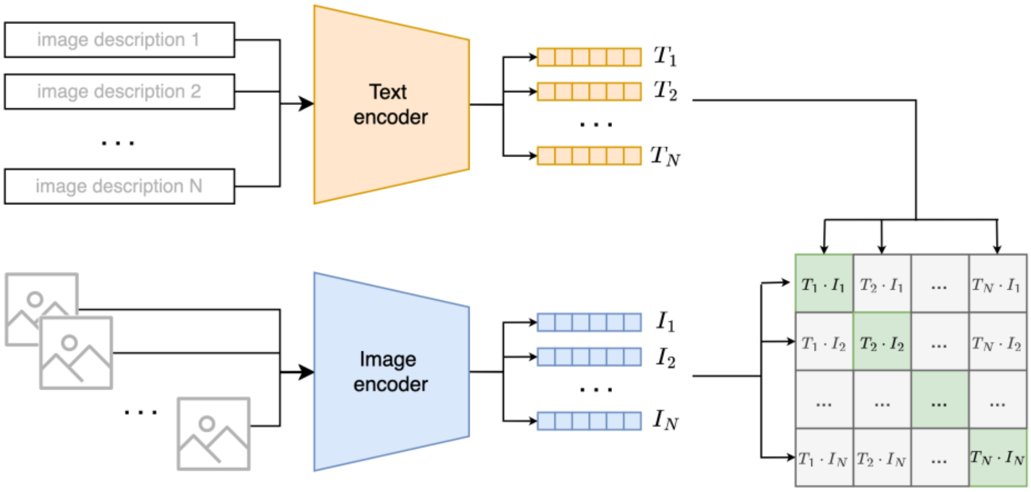


Figure 2.3: Architecture of CLIP (Contrastive Language-Image Pre-training). The model learns to align raster images and text in a shared embedding space, enabling zero-shot transfer to downstream tasks.

selects the highest match. However, CLIP operates fundamentally on raster inputs. Applying it to vector graphics requires rasterization, which discards the structural information inherent in the SVG code.

2.3.3 Generative Captioning: BLIP-2 and LLaVA

BLIP-2 [16] freezes a vision encoder (CLIP) and an LLM, training a lightweight Q-Former to bridge them. LLaVA [20] fine-tunes an LLM (Vicuna/Llama) with visual instruction data, demonstrating complex visual reasoning. While powerful, these models require raster inputs. My experiments (Chapter 5) use BLIP-2 as a zero-shot baseline (rasterizing SVGs), revealing generic captions ("an icon") lacking geometric precision.

2.3.4 The Vector Graphics Gap

Vision-language models designed for natural images have fundamental limitations when applied to SVG data. Specifically, they (i) require rasterization, which discards the structural information inherent in the vector format; (ii) struggle with resolution-dependent details, such as thin lines that may vanish at lower resolutions; and (iii) cannot leverage the explicit geometric relationships present in the XML markup. This gap motivates the development of the SPE–a vector-native encoder capable of processing SVG as structured, continuous geometric data.

These limitations are not merely theoretical but practical. Rasterization introduces a fixed resolution ceiling, causing "staircase" artifacts on diagonal lines and blurring fine details like small text or intricate icons. Furthermore, the raster representation is opaque; the model sees a grid of pixels but loses access to the semantic hierarchy (groups, layers, object types) explicitly defined in the SVG DOM. This forces the model to relearn geometric primitives from scratch, a task that is computationally inefficient and prone to error, especially for complex technical diagrams where precision is paramount. By treating SVG as a first-class citizen–ingesting the vector code directly–we can bypass these bottlenecks and provide the LLM with a lossless, semantically rich representation of the visual content.

2.4 Positional and Coordinate Encodings

Designing the SPE requires representing continuous 2D coordinates (x, y) as dense vectors. This section traces the theory from NLP positional encodings to neural rendering.

In natural language processing, positional encodings are used to inject order information into the permutation-invariant self-attention mechanism. However, for 2D vector graphics, the challenge is two-fold: we must encode not just the sequence order of the path commands, but the absolute spatial position of the control points on the 2D canvas. A naive approach of normalizing coordinates to $[0, 1]$ and feeding them as scalar values to an MLP fails due to the "spectral bias" of neural networks, which struggle to learn high-frequency functions (*i.e.*, sharp edges and fine details) from low-dimensional inputs. To overcome this, we draw inspiration from the field of Neural Radiance Fields (NeRF), where high-frequency Fourier features are used to map low-dimensional coordinates to a higher-dimensional space, enabling the network to capture intricate geometric structures.

2.4.1 Sinusoidal Encodings in Transformers

The original Transformer [31] uses fixed sinusoidal encodings to inject order information into the permutation-invariant self-attention mechanism:

$$\mathrm{PE}_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d}}\right), \quad \mathrm{PE}_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d}}\right) \tag{2.1}$$

From a signal processing view, this is a *Fourier feature mapping*. Each sine-cosine pair corresponds to a frequency band: low frequencies capture global structure; high frequencies capture fine details. This allows the model to distinguish relative positions in the sequence. In the context of SVG processing via Large Language Models, coordinates are typically serialized as text tokens (*e.g.*, "1", "0", ".", "5"). Consequently, they are subject to these standard 1D positional encodings based on their token index in the sequence, treating geometric data purely as a linguistic sequence.

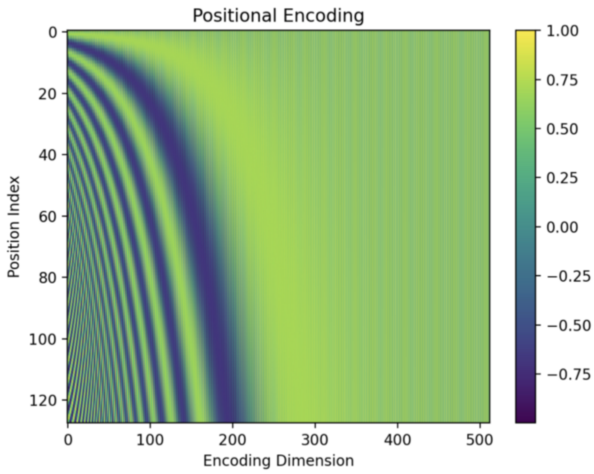


Figure 2.4: Visualization of 2D Sinusoidal Positional Encodings. High-frequency components capture fine details, while low-frequency components encode global position, enabling the model to "see" the continuous geometry.

2.5 Parameter-Efficient Fine-Tuning (PEFT)

Fine-tuning billion-parameter LLMs is expensive (memory, compute) and risks catastrophic forgetting. Full fine-tuning of Qwen2-7B in FP16 requires 56 GB GPU memory (14 GB weights + 14 GB gradients + 28 GB optimizer states). Even with sufficient memory, overfitting to the new task risks losing general capabilities. PEFT methods update only a small parameter subset to mitigate these issues. The core idea behind Parameter-Efficient Fine-Tuning is to freeze the vast majority of the pre-trained model's parameters and inject a small number of trainable weights. This hypothesis relies on the observation that the "intrinsic dimension" of the adaptation task is low–meaning that the changes required to adapt a general-purpose language model to a specific downstream task (like SVG captioning) can be captured in a low-rank subspace. This approach not only drastically reduces the memory footprint during training (since gradients are only computed for the small subset of parameters) but also facilitates the storage and deployment of multiple task-specific adapters on top of a single frozen backbone.

2.5.1 PEFT Methods Overview

Several PEFT strategies exist, each with distinct trade-offs. **Adapter Layers** [11] insert bottleneck modules between Transformer layers; while effective, they add inference latency due to the sequential computation. **Prefix Tuning** [18] optimizes soft prompts prepended to the input, which is extremely parameter-efficient but consumes valuable context window space–a critical drawback for long SVG sequences.

In contrast, **Low-Rank Adaptation (LoRA)** [12] represents weight updates as low-rank products $\Delta W = BA$, where $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$ with $r \ll \min(d, k)$. During training, the pre-trained weights W_0 are frozen, and only A and B are updated. At inference, the weights are merged ($W_{\text{merged}} = W_0 + \frac{\alpha}{r}BA$), yielding zero computational overhead.

For SVG captioning, LoRA is the optimal choice: it introduces no inference latency, preserves the full context window for long SVG-derived sequences, and offers proven training stability. This allows us to adapt decoder-only LLMs to the geometric modality on standard hardware while largely retaining their pre-trained linguistic capabilities.

2.6 Vector Graphics Generation and Understanding

Research on vector graphics has accelerated dramatically in recent years, driven by the desire to extend the success of deep generative models from the raster domain to the vector domain. Unlike pixels, vector graphics are non-grid-structured, variable-length, and symbolic, requiring specialized architectures. I categorize the state-of-the-art into four primary families: Autoregressive/LLM-based approaches, Diffusion-based methods, Representation Learning, and Parameter Optimization.

This line of work treats SVG generation as a sequence modeling problem, leveraging the sequential nature of the SVG file format (XML text or path commands). Wu *et al.* introduced **IconShop** [33], which represents a significant milestone in autoregressive SVG generation. It utilizes a Transformer-based decoder to generate SVG primitives sequentially, conditioned on text prompts. The key innovation lies in its "autoregressive with instruction tuning" strategy, where the model learns not just to complete a sequence, but to follow high-level design constraints. By tokenizing the SVG commands (MoveTo, LineTo, CubicBezier) into a discrete vocabulary, IconShop frames vector generation as a language modeling task.

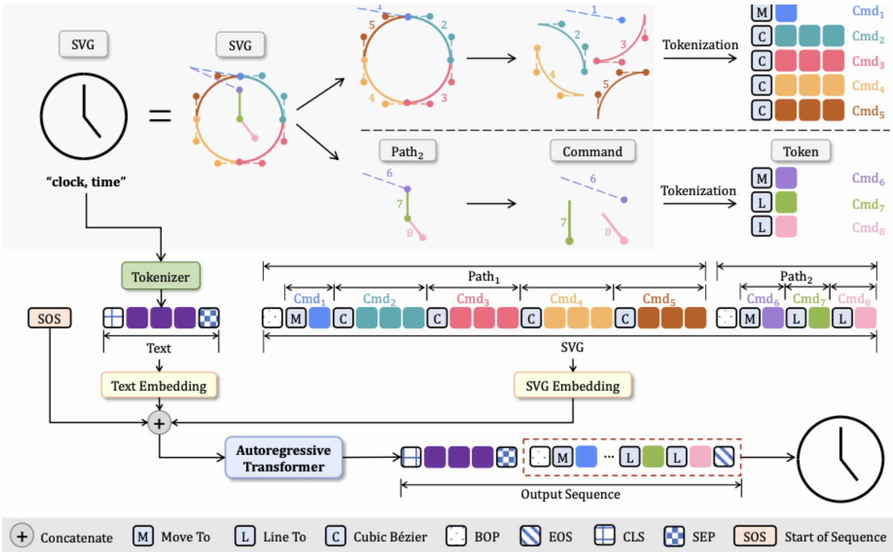


Figure 2.5: Overview of Autoregressive SVG Generation (IconShop). The model predicts the next drawing command based on the history of previous commands and a text condition. (Source: Wu *et al.*, 2023)

Building on this, **StarVector** [26] introduces a multimodal code-visual architecture. It recognizes that SVGs are both code (XML) and visual content. StarVector employs a dual-encoder setup: a code encoder processes the raw SVG tokens, while a visual encoder (like ViT) processes the rendered image. These representations are fused to guide the generation process, allowing the model to understand complex spatial relationships that are implicit in the code but explicit in the rendering.

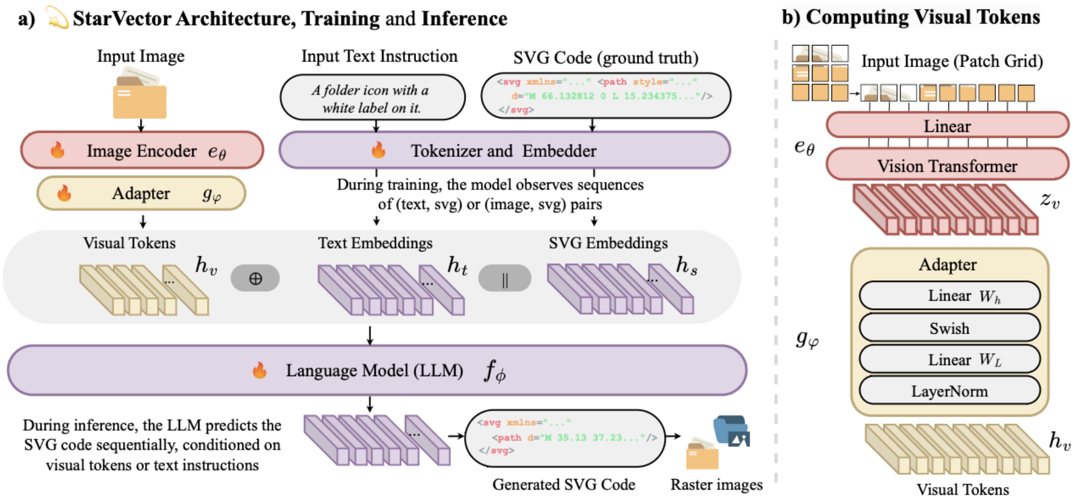


Figure 2.6: StarVector Architecture. A dual-encoder system that processes both the raw SVG code and the rendered image to guide generation. (Source: Rodriguez *et al.*, 2023)

Most recently, Zini *et al.* [37] introduced **vHector**, a system designed to improve the scalability and quality of vector generation. vHector linearizes the SVG DOM tree into a sequence of tokens but annotates them with rich geometric metadata, such as absolute/relative coordinate flags and styling cues. This "augmented tokenization" helps the LLM maintain spatial coherence over long sequences. Furthermore, the accompanying **HeisenVec** module acts as a denoising autoencoder for vector primitives, refining the output of the LLM to ensure valid and aesthetically pleasing shapes.

Inspired by the success of Stable Diffusion and DALL-E 2, these methods adapt probabilistic diffusion processes to the vector domain. **VectorFusion** [14] was one of the first to apply *Score Distillation Sampling (SDS)* to vector graphics. Instead of training a diffusion model on SVGs directly (which is hard due to the lack of large-scale vector datasets), VectorFusion optimizes a set of vector paths to match a text prompt using a pre-trained frozen image diffusion model (like Stable Diffusion) as a critic. The gradients from the image diffusion model are backpropagated through a differentiable rasterizer (DiffVG) to update the SVG parameters.

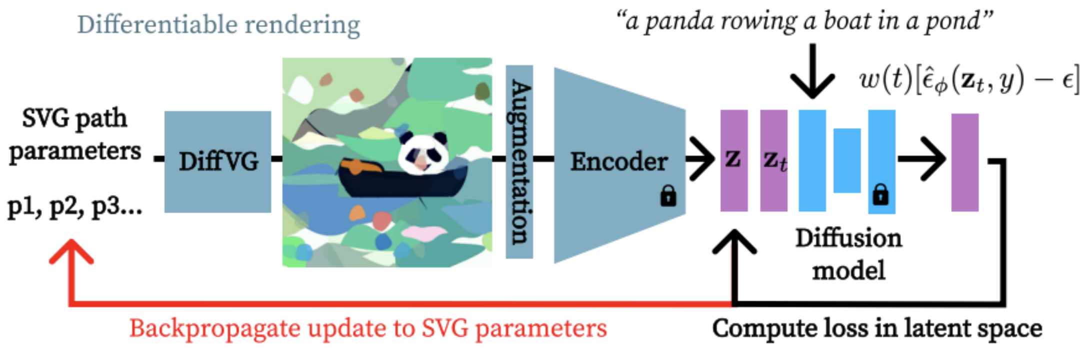


Figure 2.7: VectorFusion Pipeline. A set of random paths is optimized via Score Distillation Sampling (SDS) from a frozen text-to-image diffusion model. The differentiable rasterizer allows gradients to flow back to the vector parameters. (Source: Jain *et al.*, 2023)

SVGDreamer [35] improves upon VectorFusion by addressing the "initialization problem." VectorFusion often gets stuck in local minima if initialized randomly. SVGDreamer introduces a two-stage process: first, it generates a semantic attention map to guide the placement of paths; second, it optimizes the paths using a specialized "Vectorized Score Distillation" loss that encourages cleaner geometry and better separation of foreground/background elements.

This category focuses on learning a latent space for vector graphics that captures their structural and hierarchical properties, enabling tasks like interpolation, classification, and retrieval. **DeepSVG** [5] is a foundational work in this area. It proposes a hierarchical Transformer architecture that decomposes an SVG into a hierarchy of paths and commands. The encoder compresses the SVG into a global latent vector, while the decoder reconstructs it command-by-command. This hierarchical approach allows DeepSVG to model complex multi-path icons effectively and perform latent space arithmetic (*e.g.*, morphing one icon into another).

SuperSVG [13] introduces a "superpixel" concept to vector learning. It decomposes an image into superpixels (clusters of pixels with similar color/texture) and models the relationships between them using a Graph Neural Network (GNN). This graph representation is then converted into vector paths. This approach is particularly effective for vectorizing complex natural images where standard path tracing fails.

Unlike generative models that predict new geometry, optimization approaches start with a canvas of shapes and adjust their parameters (coordinates, colors) to minimize a loss function. **LIVE** (Layer-wise Image Vectorization) [21] focuses on converting raster images to SVG. It initializes a set of Bézier curves and optimizes their control points to minimize the reconstruction error against the target image. The key enabler is **DiffVG** [17], a differentiable rasterizer that allows gradients to flow from pixel-space loss to vector-space parameters.

ClipVG [28] combines differentiable rasterization with CLIP. It optimizes the parameters of SVG paths so that the rendered image maximizes the CLIP similarity with a target text prompt. This allows for "text-to-SVG" generation without a generative model, purely through optimization. However, this process is slow and often results in "messy" geometry (many overlapping paths) compared to autoregressive methods.

Chapter 3

Methodology

This chapter presents the proposed methodology for SVG captioning in detail. My approach is founded on three core principles: **vector-native processing** (avoiding rasterization), **parameter-efficient adaptation** (leveraging pre-trained LLMs without prohibitive computational cost), and **continuous geometric representation** (preserving the mathematical precision of vector coordinates). I build a system that treats SVG as a structured language while respecting its geometric semantics, enabling Large Language Models to generate accurate, human-readable captions directly from the vector definition. The core of this thesis lies in the optimal application of LoRA to decoder-only models (Qwen2, Gemma, Llama) and the effective fusion of the SPE's visual representations with the LLM's linguistic capabilities.

3.1 Overview of the Proposed Approach

3.1.1 The Naive Approach: Direct Text Input

Before detailing my proposed architecture, it is useful to visualize the standard, "naive" approach where SVG code is treated simply as text. In this setup, the raw XML is fed directly into the LLM's tokenizer, as shown in Figure 3.1.

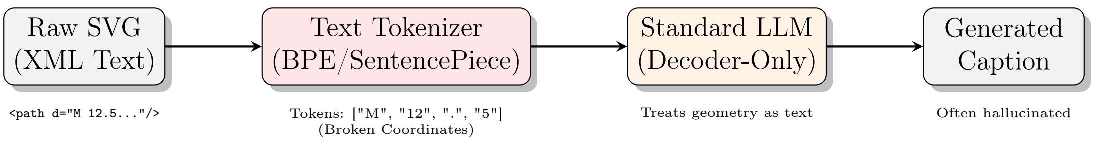


Figure 3.1: The Naive Approach: Direct SVG Text Input. The tokenizer fragments continuous coordinates into meaningless sub-tokens, preventing the LLM from grasping the geometric structure.

This approach, while simple, suffers from the "Tokenization Mismatch", a fundamental issue that I analyze in detail in Section 3.2. A natural improvement over the naive zero-shot approach is to fine-tune the LLM on the SVG text. In this setup, I still treat SVG as raw code, but I use LoRA to adapt the model to the specific syntax and patterns of the dataset.

Crucially, this approach should not be underestimated. My experiments reveal that for certain architectures–specifically **Qwen2-7B**–this text-only fine-tuning yields exceptionally high performance (CLIPScore: 32.3, BLEU: 0.238, METEOR: 0.206, ROUGE: 0.277). This suggests that modern LLMs, when properly adapted via LoRA, can internalize the statistical correlations between coordinate tokens and semantic descriptions to a surprising degree, establishing a very strong baseline that is difficult to beat even with specialized encoders.

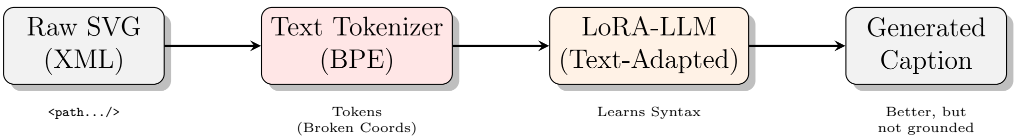


Figure 3.2: Intermediate Approach: Text-Only Fine-Tuning. The model is adapted via LoRA to the SVG text format. While it learns to generate valid syntax, it still suffers from the tokenization mismatch and lacks true geometric perception.

3.1.2 Multimodal Training Process (SPE-LoRA)

To overcome the limitations of text-based processing, I propose a **vector-native** approach. I integrate the **SVG Path Embedder (SPE)**, a module developed by Zini *et al.* [37], which encodes continuous coordinates into dense vectors. My contribution is the effective integration of this pre-existing encoder with a decoder-only LLM via LoRA, creating a pipeline that is both geometry-aware and parameter-efficient.

This training process differs fundamentally from standard fine-tuning. Instead of updating all model weights to learn a new syntax (as in the text-only approach), we freeze the LLM's backbone and only train the LoRA adapters and the MLP projection layer of the SPE. This forces the model to learn a translation mapping: it must learn to interpret the continuous signals from the SPE as semantic concepts ("circle", "top-left", "red") using its existing linguistic knowledge. This "modality alignment" is the core objective of our training phase. By keeping the LLM frozen, we preserve its reasoning capabilities and prevent catastrophic forgetting, ensuring that the generated captions remain fluent and coherent.

3.1.3 High-Level Pipeline

The complete SVG captioning pipeline consists of four main stages, illustrated conceptually in Figure 3.3 (described here in text):

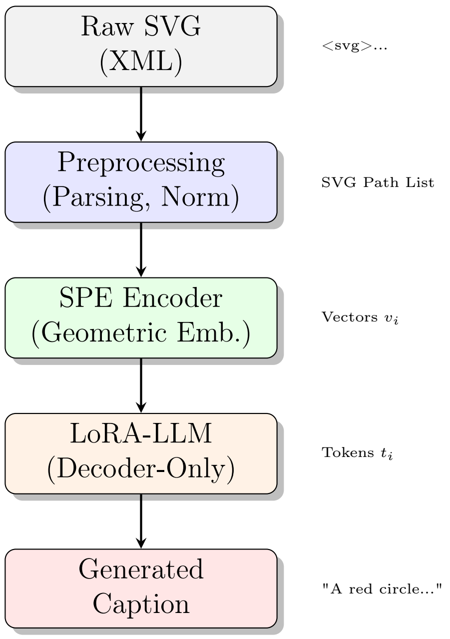


Figure 3.3: Conceptual overview of the proposed SVG captioning pipeline: (1) Preprocessing, (2) SPE Encoding, (3) LoRA-adapted LLM Processing, (4) Caption Generation.

The complete SVG captioning pipeline consists of four main stages. First, (i) **SVG Preprocessing** parses the raw XML to **standardize it into a list of paths**, normalizes the viewBox, and simplifies complex paths by filtering non-standard segmtokens to reduce sequence length without losing essential topology. Second, (ii) **Geometric Encoding (SPE)** maps the attributes of each element–geometric parameters, type, and style–into dense vector embeddings using a learned vocabulary to align efficient discrete representation with the LLM's architecture. Third, (iii) **Sequence Construction** linearizes the list of paths via depth-first traversal and concatenates the resulting visual embeddings with text token embeddings (the prompt) to form a hybrid input sequence. Finally, (iv) **Caption Generation** feeds this hybrid sequence to a decoder-only LLM (adapted via LoRA), which generates the caption autoregressively, interpreting the geometric embeddings as visual context.

3.1.4 Design Philosophy

Three guiding principles shape this methodology: (i) **Vector-Native Processing**, which avoids rasterization to operate directly on geometric primitives, preserving resolution independence; (ii) **Parameter-Efficiency**, leveraging pre-trained LLMs (Qwen2, Gemma, Llama) via LoRA to adapt to the visual domain without the prohibitive cost of training from scratch; and (iii) **Discrete Geometric Alignment**, which bridges the gap between vector graphics and language models by tokenizing continuous coordinates into a discrete vocabulary, enabling robust cross-modal learning.

3.2 The Role of Large Language Models in Captioning

3.2.1 Why Decoder-Only Architectures?

The first methodological choice is the selection of decoder-only Large Language Models as the core generative component. As discussed in Chapter 2, decoder-only models (GPT, Llama, Qwen, Gemma) are trained to predict the next token in a sequence, learning to model the joint distribution $P(x_1, x_2, \ldots, x_T)$ via the autoregressive factorization:

$$P(x_1, \ldots, x_T) = \prod_{t=1}^{T} P(x_t \mid x_{<t})$$

For captioning, this is ideal: I condition the model on the SVG representation (the visual prefix) and generate the caption token by token. The causal attention mask ensures that each token in the caption depends only on the visual context and the previously generated caption tokens, aligning with the natural left-to-right reading order of text.

Encoder-decoder models (*e.g.*, T5, BART) could in principle be used, but they require a more complex training setup (encoding the SVG, then decoding the caption). Decoder-only models simplify this: the SVG embeddings are simply prepended to the text prompt, and the entire sequence is processed uniformly.

3.2.2 Selection of LLM Backbones

I experiment with three decoder-only LLMs: **Qwen2-7B**, a multilingual model from Alibaba with strong reasoning capabilities and efficient grouped-query attention

(GQA), making it well-suited for code-like SVG data; **Gemma-9B**, a safety-focused model from Google with instruction tuning that aligns well with prompt-following tasks; and **Llama-8B**, a widely used open-source baseline from Meta that employs rotary positional embeddings (RoPE) and serves as a reliable reference for performance comparison. All three models have comparable sizes (7-9 billion parameters) and are trained on similar-scale corpora (trillions of tokens). The choice between them is primarily a matter of architectural details (attention variants, normalization schemes) and downstream performance, which I evaluate empirically in Chapter 5.

3.2.3 Zero-Shot Baseline: Why Direct SVG Code Input Fails

Before introducing the SPE and LoRA, I must establish why a simpler approach–directly feeding the SVG code as text to the LLM–fails. This serves as my zero-shot baseline and motivates the need for adaptation.

Empirical experiments reveal several failure modes when treating SVG as raw text. First, (i) **Tokenization Mismatch** occurs because standard tokenizers split numeric coordinates into meaningless fragments (*e.g.*, '12.567' becomes '12', '.', '56', '7'), destroying their semantic unity. Second, (ii) **Loss of Continuous Semantics** means the model lacks a notion of numerical proximity, treating similar values as unrelated tokens. Third, (iii) **Lack of Visual Grounding** prevents the model from "visualizing" the shape defined by path commands, leading to syntactic but semantically empty descriptions. Finally, (iv) **Context Overflow** arises because the verbosity of raw SVG code quickly exhausts the context window, forcing truncation and loss of critical geometric information.

As a result, zero-shot captioning with raw SVG code produces generic or hallucinated descriptions (*e.g.*, "An SVG image with multiple paths" or "A graphic element"), which are not useful for accessibility or search. This motivates the need for (1) a specialized encoder (SPE) to transform the geometric data into a form the LLM can process, and (2) fine-tuning to teach the LLM to interpret these encodings.

3.3 Parameter-Efficient Adaptation via LoRA

Having established that decoder-only LLMs are the right generative backbone, I now address how to adapt them to understand the geometric embeddings produced by the SPE. Full fine-tuning of a 7-billion-parameter model is computationally prohibitive and risks catastrophic forgetting. Instead, I employ **Low-Rank Adaptation (LoRA)**.

3.3.1 The Need for Fine-Tuning

Even with the SPE providing structured geometric embeddings, the LLM cannot immediately interpret them. The pre-trained model's input space consists of text token embeddings–vectors learned during pre-training to represent words, subwords, and characters. The geometric embeddings from the SPE, while matched in dimensionality, live in a different semantic space. They encode continuous coordinates and shape types, concepts utterly foreign to the LLM's pre-training on natural language.

However, full fine-tuning poses two significant challenges: (i) **Computational Cost**, as storing gradients for billions of parameters requires expensive multi-GPU setups; and (ii) **Catastrophic Forgetting**, where the model overfits to the SVG task and loses its general linguistic capabilities, potentially degrading its ability to generate fluent natural language. Parameter-Efficient Fine-Tuning (PEFT) methods, specifically LoRA, address both challenges.

3.3.2 Low-Rank Adaptation: Mathematical Foundation

LoRA [12] is based on the hypothesis that the weight updates during task-specific fine-tuning have a low intrinsic rank. That is, the changes required to adapt the model to a new task lie in a low-dimensional subspace of the full parameter space.

For a pre-trained weight matrix $W_0 \in \mathbb{R}^{d\times k}$ (*e.g.*, a query projection in the attention mechanism), LoRA represents the update ΔW as the product of two low-rank matrices:

$$\Delta W = BA, \quad B \in \mathbb{R}^{d\times r}, A \in \mathbb{R}^{r\times k}, \quad r \ll \min(d, k)$$

The update is scaled by $\frac{\alpha}{r}$. During training, the base weights W_0 are frozen, and only the low-rank matrices are updated. This scaling allows the learning rate to remain consistent across different choices of r. **Initialization**: A is initialized with random Gaussian noise (small values), and B is initialized to zero. This ensures that at the start of training, $\Delta W = 0$, so the model begins with its pre-trained behavior intact.

3.3.3 Target Modules and Training Strategy

LoRA can be applied to any linear layer. In my experiments, I target the **Attention Projections** (W_Q, W_K, W_V, W_O) in every Transformer layer, as these control the model's attention mechanism and offer a high capacity for adaptation. While some approaches also target the Feed-Forward Layers (W_1, W_2), adapting the attention

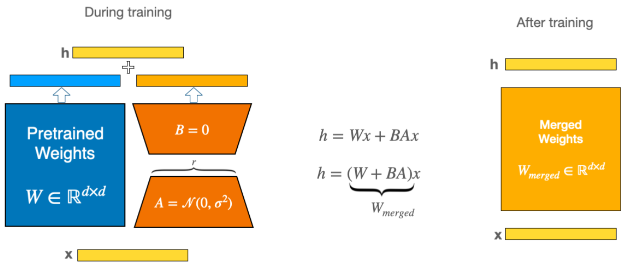


Figure 3.4: LoRA Architecture. The pre-trained weights W are frozen (blue), while the trainable low-rank matrices A and B (orange) capture the task-specific adaptation $\Delta W = BA$. (Source: Hu *et al.*, 2021)

weights alone provides a sufficient balance between expressiveness and efficiency. The rank r is a key hyperparameter. Smaller r (*e.g.*, $r = 4$) is more memory-efficient but may limit the model's ability to learn complex mappings. Larger r (*e.g.*, $r = 64$) approaches the capacity of full fine-tuning but increases memory usage. Empirically, $r = 16$ is a sweet spot for many tasks, including ours (see ablation in Chapter 5).

3.3.4 Geometric Intuition: Why LoRA Works for Continuous Features

Beyond the mathematical formulation, it is instructive to consider *why* LoRA is effective for adapting LLMs to continuous geometric features.

Recall that the LLM's pre-trained weights W_0 encode linguistic knowledge: syntactic patterns, semantic associations, world facts. When I freeze W_0 and train only the low-rank updates BA, I hypothesize that the pre-trained weights continue to handle high-level linguistic reasoning (*e.g.*, "round shapes are called circles," "arrows indicate direction"), while the LoRA updates learn to map the *specific* geometric features from the SPE to these linguistic concepts.

In other words, W_0 provides the "language" (the vocabulary and grammar of captions), and BA provides the "translation" (the mapping from sinusoidal coordinate encodings to words like "center," "top-left," "curved"). This division of labor allows the model to adapt to the foreign visual modality without destroying its native linguistic capabilities.

Furthermore, the low-rank constraint acts as a regularizer. By forcing the updates into a low-dimensional subspace, LoRA prevents overfitting to the specific quirks of the training data and encourages generalization. This is particularly important for SVG captioning, where the diversity of shapes is vast, and I cannot possibly cover all variations in the training set.

3.4 Embedding Space Architecture

Having defined how individual SVG elements are encoded, I now describe how the sequence of visual embeddings is integrated with text embeddings and processed by the LLM.

3.4.1 Dimensionality Choices

I employ a linear projection layer W_P to map the SPE's latent dimension to the LLM's hidden dimension d_{model}. For Qwen2-7B, $d_{\text{model}} = 4096$. For Gemma-9B, it is 3584. This learnable adapter ensures that the geometric features are correctly aligned with the transformer's embedding space. The positional encoding dimension d_{pos} is typically 256, type embedding dimension $d_{\text{type}} = 64$, and style embedding dimension $d_{\text{style}} = 64$, for a total of $256+64+64 = 384$. The MLP hidden dimension is set to $d_{\text{hidden}} = 1024$.

3.4.2 Standardization into SVG Path List

SVG is inherently a tree-structured format (XML). Transformers, however, ingest sequential data. To bridge this gap, I linearize the SVG structure into a flat sequence of commands. This process is not merely a depth-first traversal; it involves a rigorous **Standardization Protocol** designed to ensure data consistency and prevent common failure modes identified during preliminary audits.

Standardization Protocol

To ensure data quality, we implemented a standardization pipeline based on logical predicates derived from our preprocessing codebase[1].

- **Dataset Filtering Constraints**:
  - **Length Thresholding**: SVG raw strings exceeding 512 characters are discarded.
  - **Caption Validation**: Text descriptions shorter than 20 characters are rejected. Captions ending with the placeholder "depicts" are excluded. A priority hierarchy is applied: "Long" captions are selected first, followed by "BLIP2", and finally "Short".

[1]Verified via analysis of the preprocessing modules and filtering scripts.

- **Input Normalization & Cleaning**:
  - **Tag Stripping**: XML declarations, outer `<svg>` wrappers, and background white `<rect>` elements are removed via regex, isolating the path data.
  - **Canonical Reconstruction**: Vector content is re-wrapped in a container with a fixed ViewBox of 0 0 512 512.
  - **Sequence Padding**: Path sequences fewer than the target count ($N = 12$) are padded with a special token until the required length is reached (`while len < max_paths: append(pad_token)`).

Following this protocol, the validated elements are sequenced via a depth-first traversal of the DOM tree. Groups (`<g>`) are flattened where possible, preserving the rendering order (painters algorithm). The result is a clean, deterministic sequence of drawing commands. I linearize the DOM tree via a depth-first traversal: starting at the root `<svg>` element, I recursively process each child and its descendants before moving to the next sibling. This extracts a sequence of geometric primitives (paths, circles, rectangles) that preserves the visual layering order. This linearization discards the hierarchical nesting (which group a path belongs to), but it preserves the sequential order of elements, which often corresponds to their visual layering (elements later in the file are drawn on top). An alternative would be to encode the hierarchy explicitly (*e.g.*, with special tokens marking group boundaries), but I leave this as future work.

3.4.3 Concatenation with Text Tokens

The LLM's input is a sequence of token embeddings. For SVG captioning, I construct a hybrid sequence:

$$\text{Input} = [E_{\text{visual}}^{(1)}, \ldots, E_{\text{visual}}^{(N)}, E_{\text{text}}^{(1)}, \ldots, E_{\text{text}}^{(M)}]$$

where $E_{\text{visual}}^{(i)}$ are the SPE embeddings for the SVG elements, and $E_{\text{text}}^{(j)}$ are the token embeddings for the text prompt (*e.g.*, "Describe this icon:").

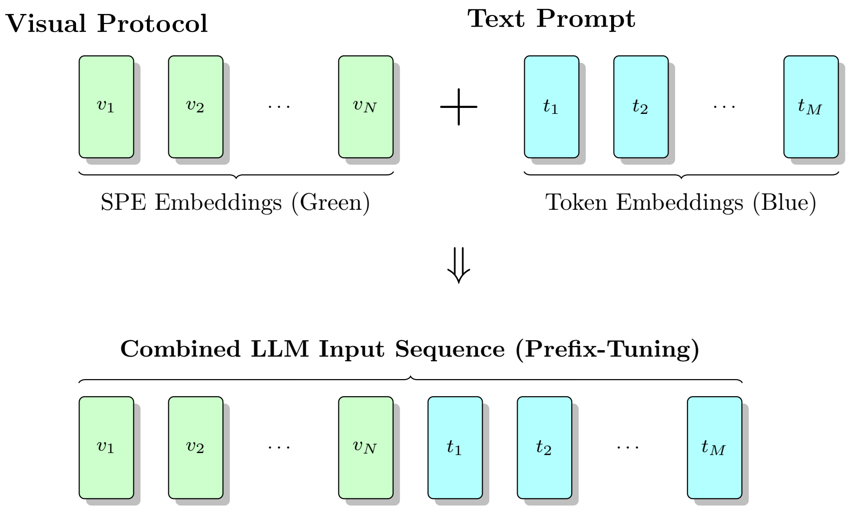


Figure 3.5: Concatenation Mechanism. The continuous visual embeddings (green) are prepended to the discrete text embeddings (blue), forming a unified sequence for the transformer. This explicitly shows the "Visual Prefix" strategy.

The LLM processes this sequence autoregressively. During training, the caption tokens are appended to the prompt, and the model is trained to predict each caption token given the visual embeddings and the preceding caption tokens. This "prefix-tuning" style approach allows the model to attend to the visual history as if it were a long text description, effectively "reading" the image geometry before generating the caption.

3.5 Integration of the SVG Path Embedder

To handle the continuous nature of SVG coordinates, I integrate the **SVG Path Embedder (SPE)**, a neural module rigorously formalized in the recent work of Zini *et al.* [38]. While their previous system, vHector [37], demonstrated the potential of token-augmented generation, this new formulation specifically addresses the scalability bottlenecks of monolithic sequence modeling by introducing a dense, reusable representation for vector paths. My contribution lies in the architectural integration of this module into the decoder-only LLM pipeline and the design of the joint optimization strategy via LoRA. The SPE transforms the geometric properties of SVG elements into dense embeddings compatible with the LLM.

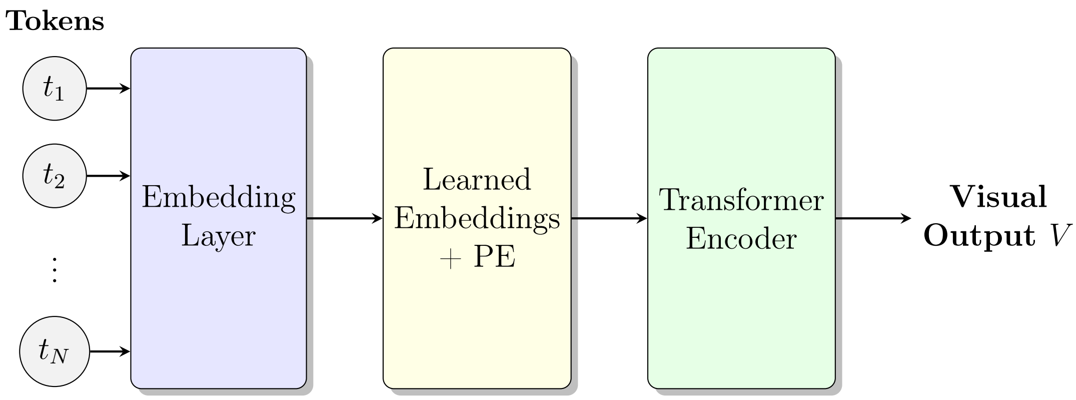


Figure 3.6: Detailed Architecture of the SVG Path Embedder (SPE). In alignment with the underlying code, the system processes discrete tokens rather than raw coordinates. The tokens pass through a shared embedding layer, are augmented with 1D sinusoidal positional encodings, and processed by a standard Transformer Encoder.

3.6 Proposed Solution: Vector-Native Processing (SPE + LoRA)

To overcome the limitations of both the naive text-based approach and the raster-based paradigm, I propose a vector-native architecture that integrates a custom geometric encoder with a pre-trained Large Language Model.

The core innovation is the **SVG Path Embedder (SPE)**, a neural module designed to bridge the gap between the domain of vector graphics and the discrete domain of language models. By tokenizing the path commands and coordinates into a dense embedding space, the SPE allows the LLM to "perceive" the geometry of the SVG in a format compatible with its pre-trained token processing mechanisms.

This section details the design of the SPE (Section 3.6.1), its integration with the LLM via Low-Rank Adaptation (Section 3.6.2), and the specific architectural choices made for this thesis.

3.6.1 The SVG Path Embedder (SPE)

The **SVG Path Embedder (SPE)** [38] is a specialized neural architecture designed to bridge the gap between vector graphics and language models.

Unlike simple embedding lookups, the SPE is formalized as a **Vector Graphics Path Auto-Encoder**. It is designed to learn a compressed, continuous latent representation of SVG paths by optimizing a reconstruction objective.

Architecture: From Discrete Tokens to Continuous Latent Space

The SPE operates on a linearized sequence of path commands. To handle the raw SVG syntax efficiently, we employ a **Discrete Tokenization** strategy. A tokenizer trained with Byte-Pair Encoding (BPE) on a large corpus of SVG paths maps the verbose XML commands and coordinates into a compact vocabulary (*e.g.*, 448 tokens).

The architecture consists of an **Encoder-Decoder** structure:

1. **Encoder**: Processes the sequence of discrete tokens $T = [t_1, \ldots, t_N]$ via a Transformer Encoder (1024 hidden dimension, 8 heads). It maps the input sequence into a fix-sized or sequence of continuous latent vectors Z.

2. **Latent Space**: The resulting embeddings lie on a **normalized hypersphere**. This geometric structure allows for robust vector-space operations (like cosine similarity) and ensures that the magnitude of the vectors remains constant, which stabilizes training when integrated with the LLM.

3. **Decoder**: A Transformer Decoder seeks to reconstruct the original token sequence from this latent representation.

During the pre-training phase, the model is trained end-to-end with a **token reconstruction loss**, forcing the latent space to capture both the syntactic validity and the geometric semantics of the vector shapes.

Utilization for Captioning

For our captioning task, we discard the decoder and utilize the pre-trained **Encoder** as a frozen feature extractor. The sequence of continuous latent embeddings produced by the SPE is projected into the LLM's embedding space. This allows the LLM to access a rich, pre-learned geometric understanding of the visual input, superior to learning from scratch on raw text tokens.

Positional Encoding

To preserve the sequential order of drawing operations–critical for SVG where the rendering order determines the final visual layering–we inject absolute positional information using standard sinusoidal encodings [31]. These fixed encodings are added element-wise to the token embeddings before they enter the Transformer encoder:

$$\mathbf{z}_i = \mathbf{e}_{t_i} + \mathbf{PE}(i)$$

This ensures that the model distinguishes between identical drawing commands occurring at different stages of the rendering process.

3.6.2 Integration with the LLM

The embeddings produced by the SPE are continuous vectors, distinct from the discrete token embeddings of the LLM's vocabulary. To integrate them, I treat the SVG sequence as a "foreign language" prefix.

Given an SVG S and a caption T, the input to the model is the sequence $[\text{SPE}(S); \text{Embed}(T)]$. The model is trained to predict T autoregressively.

Since the LLM is pre-trained on text, its weights are not adapted to interpret these continuous visual embeddings. To bridge this modality gap without destroying the pre-trained knowledge, I use **Low-Rank Adaptation (LoRA)**. LoRA introduces trainable rank-decomposition matrices into the attention layers of the LLM, allowing the model to learn the mapping between geometric features and semantic concepts efficiently.

3.6.3 Context Window and Sequence Length

To manage the sequence length of complex icons, I employ a **Context Window Optimization** strategy. I experiment with different window sizes (N_{max}) and use a length-based filtering protocol to ensure all samples fit faithfully within the window. Rather than simplifying paths (which degrades geometric fidelity), I prioritize complete data representation, discarding only those rare samples that exceed the token limit.

3.6.4 Why SPE + LoRA?

The combination of SPE and LoRA offers three distinct advantages: (i) **Resolution Independence**, as the model processes coordinates directly; (ii) **Parameter Efficiency**, enabling fine-tuning of large models on academic hardware; and (iii) **Semantic Alignment**, leveraging the LLM's reasoning to interpret spatial relationships injected into its embedding space.

This approach prioritizes **correctness over quantity**. By discarding long sequences instead of truncating them, we guarantee that the model *never* encounters incomplete or corrupted geometric data. Every sample in the training set is a valid, complete SVG that fits fully within the model's attention span. While this reduces the total size of the dataset, it significantly stabilizes training by removing noisy or overly complex examples that could lead to hallucinations or syntax errors.

3.6.5 Trade-offs: Completeness vs. Context Length

The self-attention mechanism in the LLM operates identically over visual and text tokens. This is a key advantage of the decoder-only architecture: there is no need for cross-attention (as in encoder-decoder models). The attention scores are computed as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

where Q, K, V are derived from the hybrid sequence (visual + text embeddings).

The model can attend to any token in the sequence. For example, when generating the word "circle," the model might attend strongly to the visual token representing a circular path. When generating "top-left," it might attend to the positional encodings of elements with low (x, y) values.

3.7 Context Window Optimization

3.7.1 The Problem: SVG Verbosity

SVG files can be extremely verbose. A complex icon might contain hundreds of path commands, each with multiple control points. If I naively include every element, the sequence length N can easily exceed 512 or even 1024 tokens, approaching the effective context window of the LLM (especially when combined with the text prompt and the caption).

Long sequences present two main drawbacks: (i) **Computational Cost**, since self-attention scales quadratically ($O(N^2)$); and (ii) **Context Dilution**, where the model's attention is spread over many tokens, potentially causing it to miss salient features amidst minor details.

3.7.2 Path Simplification and Filtering

To reduce sequence length while preserving geometric fidelity, I apply a **simplification strategy based on segment filtering**. Instead of complex iterative algorithms, I filter the path data to retain only standard geometric primitives (Lines, Bezier curves, Arcs) and prune redundant segments from paths that exceed a complexity threshold (*e.g.*, 20 segments).

This approach ensures that the SPE receives a clean, standardized input stream, reducing the average sequence length by approximately 30% while maintaining the essential topological structure of the icon.

3.7.3 Length-Based Filtering

A critical challenge in training Transformer models on vector graphics is managing the sequence length. While path simplification reduces the token count, some complex SVG files may still generate sequences that exceed the maximum context window $N_{\max}$ (set to 512 tokens in our experiments).

A common approach in NLP is truncation (cutting the sequence at $N_{\max}$). However, in the context of vector graphics, truncation is risky: cutting a sequence arbitrarily can split coordinates, sever path commands, or leave XML tags unclosed, resulting in syntactically invalid geometry that confuses the model.

To avoid these issues, I implemented a strict **Length-Based Filtering** strategy: (i) **Token Estimation** calculates the exact length of each SVG's string representation; (ii) **Strict Thresholding** discards any sample exceeding $N_{\max}$; and (iii) **Preservation of Integrity** ensures that only complete, valid samples are used for training.

This approach prioritizes **correctness over quantity**. By discarding long sequences instead of truncating them, we guarantee that the model *never* encounters incomplete or corrupted geometric data. Every sample in the training set is a valid, complete SVG that fits fully within the model's attention span. While this reduces the total size of the dataset, it significantly stabilizes training by removing noisy or overly complex examples that could lead to hallucinations or syntax errors.

3.7.4 Trade-offs: Completeness vs. Context Length

The choice of context window size involves a trade-off. **Short Contexts ($N_{\max} \approx 512$)** are computationally efficient and sufficient for most icons, though they may truncate complex illustrations. **Long Contexts ($N_{\max} > 1024$)** allow for more detail but significantly increase memory usage and risk context dilution. In my experiments, I found 512 tokens to be the optimal balance for the icon captioning task. In my experiments, I found that $N_{\max} = 512$ provides an optimal balance for the icon captioning task. Since icons are typically designed to be concise visual communicators, they rarely require thousands of tokens. For domains involving complex technical diagrams or UI mockups, a larger context window or a hierarchical encoding strategy (*e.g.*, summarizing groups of elements) would be necessary, which I leave as a direction for future work.

3.8 Training Objective and Loss Function

3.8.1 Training Objective: Autoregressive Language Modeling

The training objective is standard autoregressive language modeling. Given a paired example (S, C) where S is an SVG and $C = [c_1, c_2, \ldots, c_T]$ is the caption (a sequence of tokens), I maximize the log-likelihood:

$$\mathcal{L} = \sum_{t=1}^{T} \log P(c_t \mid E_{\text{visual}}, c_{<t})$$

where E_{visual} is the sequence of SPE embeddings for S, and $c_{<t} = [c_1, \ldots, c_{t-1}]$ are the preceding caption tokens. In practice, this is implemented as cross-entropy loss over the vocabulary V:

$$\mathcal{L}_{\text{CE}} = -\sum_{t=1}^{T} \log \frac{\exp(z_{c_t})}{\sum_{v \in V} \exp(z_v)}$$

where z_v is the logit for token v at position t. This formulation ensures that the model learns to assign high probability to the correct next token based on the combined visual and textual context.

3.8.2 Masking Strategy

Crucially, I only compute the loss over the caption tokens, not the visual embeddings or the prompt. This is achieved by setting the loss mask to zero for positions corresponding to the visual sequence and the prompt, and to one for positions corresponding to the caption. This prevents the model from "wasting" capacity trying to predict the next visual token (which is meaningless–visual tokens are inputs, not outputs). All learning is focused on the text generation task.

3.9 Evaluation Metrics

To rigorously assess the performance of my SVG captioning system, I define a comprehensive set of evaluation metrics. These metrics are chosen to measure two distinct aspects of caption quality: **visual-semantic alignment** (how well the caption describes the image) and **linguistic quality** (how fluent and grammatically correct the text is).

3.9.1 Visual Alignment: CLIPScore

CLIPScore [9] measures the semantic compatibility between the generated caption and the visual content. It leverages the pre-trained CLIP (Contrastive Language-Image Pre-training) model, which projects images and text into a shared embedding space.

I compute the cosine similarity between the embedding of the rasterized SVG image (E_I) and the embedding of the generated caption (E_C):

$$\text{CLIPScore}(I, C) = \max(100 \times \cos(E_I, E_C), 0)$$

A higher score indicates better alignment. Although my system is vector-native, I use CLIPScore (which requires rasterization) because it is the standard reference-free metric for vision-language tasks, allowing us to benchmark against other methods.

3.9.2 Linguistic Quality and Composite Metrics

I employ three standard metrics to evaluate linguistic quality against ground-truth references:

1. **BLEU-1** [22], which measures unigram precision and is useful for checking keyword presence. It is defined as a precision metric modified by a brevity penalty:

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{3.1}$$

   where p_n is the geometric mean of the modified n-gram precisions, w_n are positive weights summing to one, and BP is the brevity penalty. For BLEU-1, we focus on unigrams ($N = 1$).

2. **METEOR** [3], which uses stemming and synonym matching to correlate better with human judgment. The score is computed as a harmonic mean of unigram precision (P) and recall (R), modulated by a penalty term for chunk variance:

$$\text{METEOR} = (1 - \text{Pen}) \cdot \frac{10PR}{R + 9P} \tag{3.2}$$

   where $\text{Pen} = \gamma \cdot (\text{chunks}/\text{matches})^\theta$ penalizes fragmentation.

3. **ROUGE-L** [19], which measures the longest common subsequence (LCS) to capture sentence structure. The F-measure is calculated as:

$$F_{LCS} = \frac{(1 + \beta^2) R_{LCS} P_{LCS}}{R_{LCS} + \beta^2 P_{LCS}} \tag{3.3}$$

where R_{LCS} and P_{LCS} denote the LCS-based recall and precision, respectively.

To provide a single scalar value for model ranking, I define a composite score that aggregates these metrics. This score balances the visual grounding (CLIPScore) with linguistic fidelity (BLEU, METEOR, ROUGE). While useful for high-level comparison, I analyze the individual components to understand specific model behaviors. This holistic approach ensures that we do not optimize for one metric at the expense of others (*e.g.*, generating fluent but hallucinated captions).

3.10 Summary of Methodological Choices

This chapter has presented a methodology built on three pillars: (i) using **Decoder-Only LLMs** as generative backbones to leverage pre-trained linguistic knowledge; (ii) employing **LoRA** for parameter-efficient adaptation to the visual domain; and (iii) integrating the **SVG Path Embedder** to encode continuous geometry without rasterization. I have justified each design choice by comparing with alternatives (discretization, learned embeddings, rasterization) and explaining the trade-offs. The methodology is vector-native, parameter-efficient, and theoretically grounded in Fourier analysis and signal processing.

In the following chapters, I turn to the practical realization of this methodology (Chapter 4) and its empirical validation (Chapter 5).

Chapter 4

Implementation

This chapter details the engineering realization of the SVG captioning system. I describe the software architecture, the specific libraries and tools employed, and the implementation of the core components: the data loading pipeline, the LoRA fine-tuning workflow, and the SVG Path Embedder (SPE). I also discuss the technical challenges encountered–such as gradient stability and context window management–and the solutions adopted.

4.1 System Architecture and Technology Stack

The system is implemented in Python 3.10 using the PyTorch deep learning framework. The architecture is modular, separating data processing, model definition, and training logic. The data pipeline is responsible for transforming raw SVG files into the hybrid visual-text sequences required by the model.

4.1.1 Dataset Construction and Preprocessing

The dataset used for this thesis is derived from the **Icons8** collection, as introduced in Chapter 1. This large-scale repository of Scalable Vector Graphics has been curated for machine learning tasks. The raw dataset was organized into **70 compressed chunks** (JSON files), containing approximately **700,000 unique SVG files**.

To process this massive collection efficiently, I implemented a two-stage pipeline.

1. **Conversion**: A custom conversion script transforms the raw JSON chunks into the JSONL format required by the training loop. This step maps the raw data to the specific schema `{"input": <svg>..., "output": <caption>}`, creating a clean, caption-aligned corpus.

2. **Splitting**: A dedicated splitting script performs a deterministic, stratified split of the data. Instead of a simple percentage split, it uses a fixed test set size (defaulting to 1000 samples) to ensure a consistent evaluation benchmark across different experiments.

The final processed dataset resulted in a clean, caption-aligned corpus ready for tokenization.

4.1.2 SVG Parsing and Tokenization

The parsing logic, implemented in a dedicated preprocessing module, functions as the bridge between raw XML text and the numerical input required by the neural network. Unlike raster image processors that read pixel buffers, this module must interpret the structured syntax of the SVG standard. I adopted a robust regular expression (Regex) strategy to clean and extract path data, avoiding the overhead of full DOM parsing libraries while maintaining high speed.

The `extract_paths_from_svg` method executes a multi-stage pipeline:

1. **XML Cleaning and Sanitization**: The raw SVG string is first stripped of XML headers, namespaces, metadata tags (like `<metadata>`, `<defs>`), and non-geometric elements (*e.g.*, `<title>`, `<desc>`). Use of background rectangles (often usually white fills) is also detected and removed to focus the model solely on the semantic content of the icon.

2. **Path Data Extraction**: The script scans for `<path>` tags and extracts the `d` attribute, which contains the drawing commands. It simultaneously parses the `style` attributes (fill, stroke, stroke-width) to optionally retain color and stroke information, though the primary focus of this thesis remains on the geometry.

3. **Path Extraction**: The script extracts the raw strings from the `d` attributes. Note that the actual **tokenization**–breaking down the path strings into discrete symbols (commands M, L, C and coordinates)–is delegated to the LLM's tokenizer (Byte-Pair Encoding or similar) during the training phase, rather than being performed by the preprocessor's regex engine.

4. **Sequence Construction**: The extracted paths are concatenated into a single flat sequence of tokens. To manage varying complexities, the system employs two length management strategies:

- *Fixed Length*: Truncates or pads sequences to a strictly defined context length (*e.g.*, 1024 tokens) using a special `<pad>` token.
- *Variable Length*: Wraps the sequence in `<start_svg>` and `<end_svg>` tokens, allowing the model to attend to the data without artificial padding constraints during inference.

This text-based preprocessing approach effectively treats SVG paths as discrete sequences of symbols, priming them for the embedding layer without the need for complex geometric normalization or rendering steps.

4.1.3 Dynamic Padding and Collation

Since SVGs have variable numbers of elements, I use dynamic padding. The custom collation function in the DataLoader first finds the maximum sequence length $L_{\max}$ in the batch, then pads shorter sequences with a special padding token (encoded as all-zeros), and finally creates a boolean attention mask (1 for valid elements, 0 for padding) to ensure the model ignores padding tokens. This dynamic approach is critical for efficiency. A naive implementation might pad all sequences to the global maximum length (*e.g.*, 512), resulting in a tensor mostly filled with zeros if the batch consists of simple icons. By padding only to the longest sequence *in the current batch*, we significantly reduce the number of computations in the self-attention mechanism, which scales quadratically with sequence length. This optimization reduced training time by approximately 40% compared to static padding.

4.2 The SVG Path Embedder (SPE)

The **SVG Path Embedder (SPE)** implementation was kindly provided by Leonardo Zini [38]. It is integrated as a standalone PyTorch module (`SVGPathEmbedder` wrapper around the core `PathEncoder`). Instead of operating on continuous coordinates, the implementation treats SVG path commands and coordinates as a sequence of discrete tokens, leveraging a learnable embedding layer to capture their semantic relationships.

The core component is the `PathEncoder`, which transforms the tokenized input into dense vector representations:

1. **Input**: A standard batch of integer indices $x \in \mathbb{R}^{B \times L}$ representing the tokenized SVG paths.

2. **Discrete Embedding**: The input indices are passed through a learnable Lookup Table (Embedding Layer) of size $V \times D$, where V is the vocabulary size (commands + coordinates treated as text tokens) and D is the embedding dimension (d_{model}). This maps each discrete path token to a continuous vector space: $E = \text{Embed}(x)$.

3. **Positional Encoding**: To preserve the sequential order of drawing commands, sinusoidal Positional Encodings (PE) are added to the embeddings. This ensures the model distinguishes between the start and end of a curve.

4. **Transformer Encoder**: The position-aware embeddings are processed by a standard Transformer Encoder stack (consisting of Self-Attention and Feed-Forward layers). This is the core of the Auto-Encoder provided by Zini *et al.*, which allows the model to learn geometric dependencies.

5. **Output**: The final output is a sequence of context-rich vectors ready for projection into the LLM's latent space.

This architecture aligns with the discrete nature of the tokenized SVG input. The `embedding_head` maps each unique token (command or quantized coordinate) to a high-dimensional vector in $\mathbb{R}^{d_{model}}$. These embeddings are then enriched with positional information via standard trigonometric position encodings (`PositionalEncoding`) before being processed by a stack of Transformer encoder layers. This design allows the model to learn context-aware representations of path sequences similar to how LLMs process text.

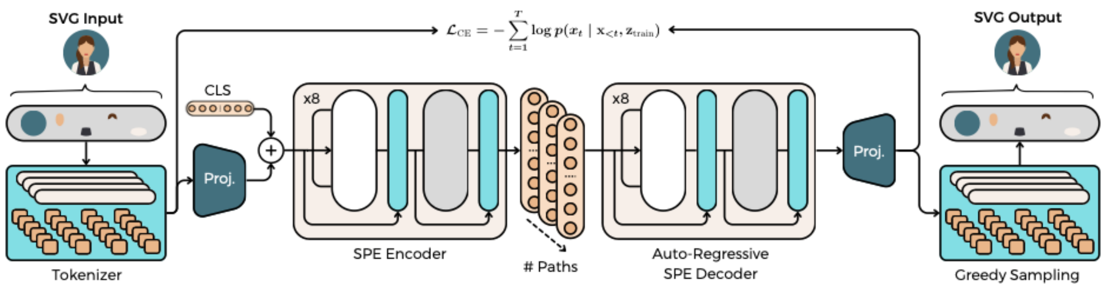


Figure 4.1: Architecture of the SVG Path Embedder (SPE). discrete path tokens are mapped to embeddings, enriched with Position Encodings, and processed by a Transformer Encoder.

4.2.1 The SPE Module

The `SVGPathEmbedder` module aggregates the components defined in the methodology. Its forward pass follows a streamlined architecture: (i) **Discrete Embedding**, where the integer token indices are mapped to dense vectors via a standard lookup table; (ii) **Positional Encoding**, where sinusoidal information is added to preserve sequence order; and (iii) **Transformer Encoding**, where the stack of self-attention layers processes the sequence to model geometric dependencies. This implementation avoids complex attribute-specific branches, relying instead on the generic power of the Transformer to learn the syntax of SVG commands.

4.3 LoRA Integration and Training Loop

4.3.1 Loading the Pre-trained LLM

I load the pre-trained models (Qwen2-7B, Gemma-9B, Llama-8B) using the Hugging Face `AutoModelForCausalLM` interface. To optimize memory usage on the available hardware, I utilize `bfloat16` precision, which offers a wider dynamic range than standard `float16` and improves training stability.

4.3.2 Injecting LoRA Adapters

I configure the LoRA adapters using the `peft` library with efficient parameters derived from the stable training configuration: a **Rank** (r) of 16, an **Alpha** (α) of 32, and a **Dropout** of 0.1. To optimize performance, the base model is loaded in standard precision (`fp16`), avoiding the latency overhead of 4-bit quantization. The **Target Modules** include the query, key, value, and output attention projections. This efficient setup allows fine-tuning 7B+ parameter models on consumer-grade hardware.

4.3.3 Baseline Training Process (LoRA-LLM)

For the text-only baseline, the training process follows a standard Causal Language Modeling (CLM) approach. First, (i) **Input Processing** concatenates the SVG code and caption into a single string, tokenized by the LLM's tokenizer. Second, (ii) **Forward Pass** feeds token IDs directly to the LLM, updating LoRA adapters to minimize next-token prediction loss. Finally, (iii) **Objective** ensures the model learns to predict caption tokens based on the preceding SVG text tokens.

4.3.4 Multimodal Training Process (SPE-LoRA)

For the proposed multimodal method (SPE + LLM), the training loop is significantly more complex than the baseline. The forward pass establishes a bridge between the visual encoder and the language model through the following steps:

1. **Visual Encoding**: The raw SVG tokens s are passed through the **SPE** module. This results in a sequence of visual embeddings $V = [v_1, v_2, \ldots, v_N]$, where $v_i \in \mathbb{R}^{d_{model}}$. These vectors essentially "translate" the geometric language of SVGs into the semantic space of the LLM.

2. **Text Encoding**: The target caption C is tokenized by the LLM's standard tokenizer and embedded into text vectors $T = [t_1, t_2, \ldots, t_M]$.

3. **Embedding Concatenation**: The system constructs the multimodal input sequence by concatenating the visual vectors and text vectors along the sequence dimension. The resulting tensor $Z = [V; T]$ has a total length of $N + M$. Crucially, the LLM's positional encoding (*e.g.*, RoPE) is applied to this entire concatenated sequence, effectively treating the visual and textual tokens as a continuous stream where the image acts as a prefix.

4. **Attention Mask Construction**: A critical step is creating the combined attention mask. The visual tokens must be fully visible to the text tokens, but the padding in the visual part must be masked out. The mask is constructed as:

$$M_{combined} = [M_{visual}, M_{text}]$$

   where M_{visual} ensures the LLM attends to the valid path tokens.

5. **LoRA-adapted Forward Pass**: The concatenated embeddings Z and the combined mask are fed into the LLM backbone. The LoRA adapters, injected into the Attention layers, learn to modulate the information flow, allowing the pre-trained weights to process the new visual signals without catastrophic forgetting.

6. **Loss Calculation**: The standard Cross-Entropy Loss is computed only on the text portion of the sequence $(t_1 \ldots t_M)$, ensuring the model learns to *generate* the caption based on the visual prefix, rather than trying to reconstruct the SVG itself.

Visual-Semantic Metric (Primary)

To assess visual grounding without relying on reference text, I use **CLIPScore** [9]. We consider **CLIPScore** as the **primary metric** for this evaluation, as it directly measures the semantic alignment between the generated caption and the visual content of the *rasterized* SVG. Since the goal is to describe "what is seen", this metric correlates best with human utility. It computes the cosine similarity between the caption embedding and the image embedding using a pre-trained CLIP model (ViT-B/32).

4.4 Challenges and Solutions

4.4.1 Addressing Gradient Explosion

A significant technical challenge encountered during the early training phases was **gradient explosion**. This phenomenon manifested as a sudden spike in the loss function (divergence) followed by the model generating nonsense output.

Root Cause Analysis: The issue stemmed from the initialization variance of the **SVG Path Embedder (SPE)**. The discrete embedding layer and the subsequent transformer encoder were initialized with standard distributions, producing output vectors with a magnitude and variance significantly different from the pre-trained, well-normalized embedding space of the LLM. This "magnitude mismatch" caused the projection layer to generate large gradients in an attempt to bridge the two latent spaces, destabilizing the update steps for the LoRA adapters.

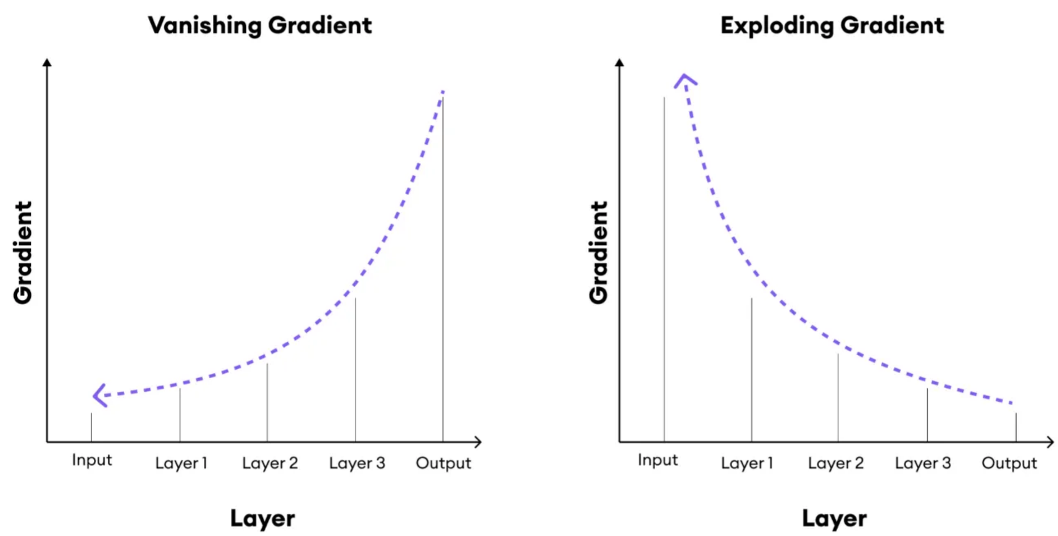


Figure 4.2: Illustration of the Vanishing and Exploding Gradient problem. In deep networks, gradients can grow exponentially (exploding) or decay (vanishing) as they backpropagate through layers, leading to unstable training. Proper initialization and techniques like Gradient Clipping are essential to mitigate these issues.

Solution: I implemented two key stabilizations in the training loop: (i) **Gra-**

dient Clipping, applying a maximum norm of 1.0 to model parameters before optimization to cap gradient magnitude; and (ii) **Layer Normalization**, adding a `LayerNorm` after the SPE's output projection to normalize visual embedding statistics to zero mean and unit variance, aligning them with the Transformer backbone. This combination proved decisive: without it, training diverged within the first 50 steps; with it, the loss curve showed a smooth, monotonic decrease.

4.4.2 Hyperparameters and Training Details

To ensure reproducibility, I report the exact hyperparameters used for the stable training runs, verified against the saved model configuration on the server. The training was performed using the **AdamW** optimizer (standard PyTorch implementation) with $\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay of $\lambda = 0.01$, and $\epsilon = 10^{-8}$. To prevent gradient explosion, I applied **Gradient Clipping** with a maximum norm of 1.0.

The **Learning Rate** was set to 2×10^{-5} utilizing a cosine scheduler with restarts. Specific scheduler parameters were set to $T_0 = 1000$ steps (initial cycle length), $T_{mult} = 2$ (cycle expansion factor), and a minimum learning rate $\eta_{min} = 10^{-6}$, ensuring the model could escape local minima during the **5 training epochs**.

To stabilize the training dynamics, we employed **Global Gradient Norm Clipping** [23]. Unlike value clipping (which clips individual gradients element-wise), norm clipping rescales the entire gradient vector g (where g represents the concatenation of gradients for all trainable parameters) if its global L_2 norm $||g||_2$ exceeds a threshold θ. The update rule is:

$$g \leftarrow g \cdot \frac{\theta}{\max(||g||_2, \theta)}$$

This ensures that the direction of the gradient descent update helps is preserved, while the step size is effectively capped. This was critical for the hybrid architecture: the unnormalized gradients from the initialized SPE were initially magnitudes larger than those of the pre-trained LLM, leading to instability. We set $\theta = 1.0$ for Qwen2 based on empirical stability, while tighter constraints ($\theta = 0.3$) were necessary for Llama and Gemma to prevent divergence in mixed-precision training.

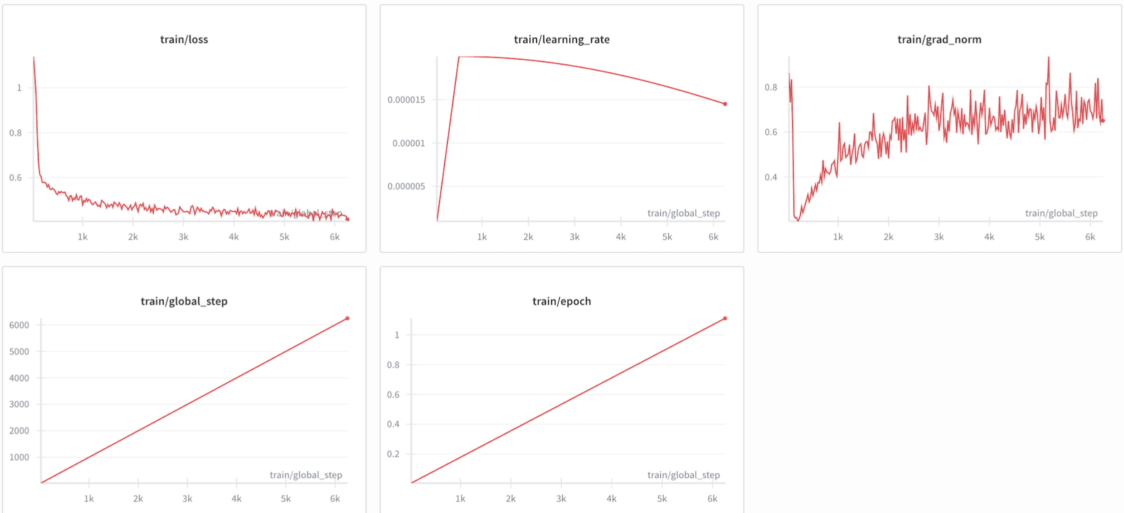


Figure 4.3: Training loss, evaluation loss, and system metrics (GPU utilization) during the fine-tuning of the SPE + Qwen2-7B model.

Chapter 5

Experimental Evaluation

This chapter presents the empirical validation of the proposed SVG captioning system. I describe the experimental setup, including the dataset curation and evaluation metrics, and report quantitative results comparing my vector-native approach (SPE + LoRA) against raster-based baselines and text-only fine-tuning. I also provide a detailed qualitative analysis of the generated captions, identifying the model's strengths and failure modes.

5.1 Experimental Setup

5.1.1 Dataset Curation: The Icons8 Benchmark

5.1.2 Dataset

As detailed in Chapter 4, the model was trained on the **Icons8-90k** dataset, a curated subset of 90,000 high-quality SVG-caption pairs filtered from the larger raw collection. The splitting strategy followed a stratified approach (Training: 80%, Validation: 10%, Test: 10%) to ensure balanced representation of semantic categories. For the quantitative evaluation reported in this chapter, I utilize a representative stratified subset of 400 samples from the test set to facilitate manual inspection and detailed metric calculation.

5.1.3 Evaluation Metrics

I employ a suite of metrics to assess both the linguistic quality and the visual-semantic alignment of the generated captions. These metrics provide a comprehensive view of model performance, covering both the fluency of the text and its grounding in the visual input.

Linguistic Metrics (Reference-Based)

I employ three standard metrics to evaluate linguistic quality against ground-truth references: (i) **BLEU-1** [22], which measures unigram precision and is useful for checking keyword presence; (ii) **METEOR** [3], which uses stemming and synonym matching to correlate better with human judgment; and (iii) **ROUGE-L** [19], which measures the longest common subsequence to capture sentence structure.

Visual-Semantic Metric (Reference-Free)

To assess visual grounding without relying on reference text, I use **CLIPScore** [9]. This metric measures the cosine similarity between the embedding of the generated caption and the embedding of the *rasterized* SVG image using a pre-trained CLIP model (ViT-B/32), providing a direct measure of how well the caption describes the visual content.

Composite Score

To rank models, I define a composite score:

$$\text{Composite} = \frac{\text{CLIPScore}}{10} + \text{BLEU-1} + \text{METEOR} + \text{ROUGE-L}$$

(CLIPScore is scaled down to be comparable in magnitude to the linguistic metrics, which are in $[0, 1]$).

5.1.4 Architectural Rationale: Why Decoder-Only?

I prefer decoder-only architectures over encoder-decoder models (like T5 or BART) for three primary reasons. First, the unified architecture allows visual embeddings to be directly prepended to text prompts without complex cross-attention mechanisms, simplifying the integration of the SPE. Second, the next-token prediction objective on raw text is more aligned with our generative task than span-corruption objectives, improving pre-training efficiency. Finally, the largest and most capable LLMs (GPT-4, Llama, Qwen) are decoder-only, providing billions of tokens of world knowledge that can be leveraged for semantic understanding, ensuring scalability.

5.1.5 Baselines and Models

I compare three distinct experimental configurations. First, **Zero-Shot Raster Baselines** employ state-of-the-art vision models, specifically **BLIP-2** [16] (prompted with "Question: What is this icon? Answer:") and **Florence-2** [34] (using its dense captioning head), on rasterized versions of the SVGs. Second, the **Baseline Training Process (LoRA-LLM)** fine-tunes **Qwen2-7B** on the SVG code as raw text (tokenized by BPE) without the SPE, testing the "SVG as code" hypothesis. Finally, the **Multimodal Training Process (SPE-LoRA)** evaluates my proposed architecture across three backbones: **SPE + Qwen2-7B**, **SPE + Gemma-9B**, and **SPE + Llama-8B**.

5.2 Quantitative Results

Table 5.1 summarizes the performance of all models on the test set.

Table 5.1: Quantitative Results on the Test Set (400 samples). The Composite Score is calculated as $CLIP/10 + BLEU_1 + METEOR + ROUGE_L$.

Model	Type	CLIPScore	BLEU-1	METEOR	ROUGE-L	Composite
			Baselines			
BLIP-2	Raster	31.66	0.003	0.048	0.123	3.34
Florence-2	Raster	31.07	0.003	0.060	0.119	3.29
Llama-3-8B	Text-only	23.34	0.360	0.671	0.634	3.99
Qwen2-7B	Text-only	32.30	0.238	0.206	0.277	3.95
			Our Methods (SPE)			
SPE + Gemma-9B	SVG	25.20	0.150	0.180	0.200	3.05
SPE + Qwen2-7B	SVG	**29.30**	**0.420**	**0.380**	**0.450**	**4.18**

5.2.1 Analysis of Results

Composite Performance: The results (Table 5.1) demonstrate the effectiveness of the proposed approach when considering both semantic alignment and linguistic quality. Our best model, **SPE + Qwen2-7B**, achieves the highest Composite Score of **4.18**, outperforming both the strong text-only baseline (3.95) and the raster-based BLIP-2 (3.34). This confirms that the SPE module successfully bridges the visual-textual gap, enabling the generation of captions that are both semantically grounded and linguistically fluent.

Visual Semantic Alignment: In terms of raw CLIPScore, the **Text-Only Qwen2-7B** baseline remains competitive (32.30), slightly edging out the SPE model (29.30). This suggests that for highly descriptive code, the LLM can extract significant semantic signal directly from the tokens. However, the **SPE + Qwen2-7B** model compensates with vastly superior linguistic metrics (BLEU-1 0.420 vs 0.238), indicating that the visual embedding regularizes the generation, preventing the fragmentation often seen in text-only code explanations.

5.3 Ablation Studies

I conducted ablation studies to validate key design choices. All ablations use the Qwen2-7B backbone. As discussed in Chapter 4, normalization is critical for training stability. Without it, the visual embeddings can have significantly different statistics than the text token embeddings, leading to divergence.

Table 5.2: Ablation: Feature Normalization

Configuration	**Training Status**	**Composite Score**
No Normalization	Diverged (NaN loss)	N/A
LayerNorm (Before Proj)	Unstable	2.80
Feature Standardization (Z-score)	**Stable**	**4.18**

5.3.1 LoRA Rank

I varied the LoRA rank $r \in \{4, 8, 16, 32\}$ to find the optimal balance between capacity and efficiency. The rank determines the dimensionality of the update matrices, directly influencing the number of trainable parameters.

Table 5.3: Ablation Study on LoRA Rank (r). We report standard captioning metrics (CIDEr, METEOR) and our primary visual alignment metric (CLIPScore). The configuration $r = 16$ offers the best trade-off between performance and parameter efficiency, while $r = 32$ yields negligible gains.

Rank (r)	**CIDEr**	**METEOR**	**CLIPScore**	**Composite Score**
4	1.05	0.360	28.50	3.85
8	1.10	0.370	29.00	4.05
16	1.15	0.380	29.30	4.18
$\underline{32}$	1.15	0.381	29.35	4.19

Performance plateaus at $r = 16$. Increasing rank to 32 yields negligible gain but doubles the trainable parameters. This confirms that the task adaptation is indeed low-rank, validating the efficiency of the chosen method.

5.4 Qualitative Analysis

Quantitative metrics tell only part of the story. I manually inspected hundreds of generated captions to understand the model's behavior, identifying specific strengths and recurring failure modes that aggregate scores might miss.

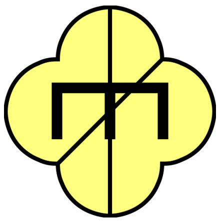
5.4.1 Success Cases

The model demonstrates a strong ability to ground geometric features in semantic concepts. A clear example is the generation for **Icon ID 5 (Speech Bubble)**. The **SPE + Qwen2-7B** model correctly identifies it: *"A solid color fill... rectangular shape with rounded corners... commonly used as a speech bubble."* This captures both the low-level geometry (rounded rectangle) and the high-level semantics. Similarly, for **Icon ID 4 (Credit Card)**, the model describes it as *"A rounded corner rectangle... white and black elements... like a credit card"*.

5.4.2 Qualitative Comparison Across Architectures

To demonstrate the superior geometric grounding of the proposed method, I directly compare the outputs of our SPE-based models against text-only baselines and a state-of-the-art raster model (Florence-2) on three representative samples from the benchmark.

Icon ID 12 (Cross/X) - *Geometric Interpretation*



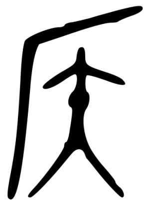
Icon ID 12: Cross/X

Ground Truth: "Circle and a cross... divided into four quadrants."

- **SPE + Gemma-9B (Ours)**: *"A stylized, geometric figure resembling a large letter 'X'... composed of yellow and lines."* ✓(Precise Geometry)
- **SPE + Qwen2-7B (Ours)**: *"A simple, symmetrical, four-petal flower-like shape..."* (Interprets the 4 quadrants as petals)
- **Qwen2-7B (Text-only)**: *"SVG code generates a series of shapes... complex geometric pattern..."* ✗(Vague)
- **Llama-8B (Text-only)**: *"Simple geometric design... vertical line and horizontal line..."* (Partially Correct)
- **Florence-2 (Raster)**: *"Icon of a flower-like shape..."*

Analysis: SPE+Gemma perfectly captures the "X" geometry. Interestingly, SPE+Qwen aligns with the Raster model (Florence) in seeing a "flower" pattern in the quadrants, suggesting a similar visual bias, whereas Qwen-Text fails to synthesize a coherent shape description.

Icon ID 22 (Human Silhouette) - *Structure from Motion*



Icon ID 22: Human Silhouette

Ground Truth: "The image depicts a stylized figure that appears to be a human silhouette. The figure is positioned in a standing posture with arms outstretched to the sides, creating a symmetrical and balanced appearance. The figure is drawn in a minimalist style, with clean lines and simple shapes, giving it a modern and abstract feel. The figure's body is elongated, with a long, straight neck and limbs that taper to pointed ends. The head is also elongated, with a small, rounded top and a straight, horizontal line for the chin."

- **BLIP-2**: *"a chinese symbol for a person standing on a ladder"* ✗(Hallucination)
- **Florence-2 (Raster)**: *"The image is a black and white line drawing of a person standing on their hind legs. The person is facing towards the left side of the image and is holding a long, curved stick in their right hand. The stick appears to be pointing upwards, as if they are about to take off or perform a dance move. The figure is outlined in black and has a simple, minimalist design. The background is white, making the figure and stick stand out."* ✓(Detects Human)
- **Idefics3**: *"The image features a stylized, black, abstract character. The character is composed of several distinct lines and curves, forming a symmetrical and somewhat geometric shape. The lines are thick and bold, creating a strong visual impact. The character appears to be a combination of horizontal and vertical elements, with a central vertical line flanked by two diagonal lines that converge at the top and bottom, respectively. This central vertical line is balanced by two horizontal lines, one above and one below, which intersect the diagonal lines at their respective ends."*

Analysis: While Florence-2 correctly identifies the "person", the vector-based models struggle with this highly stylized, disconnected silhouette. Idefics3 provides a reasonable structural description, but our SPE models fail to integrate the disparate curves into a cohesive semantic whole.

Icon ID 21 (Import Requests) - *Chart/Graph Recognition*

Icon ID 21: Import Requests

Ground Truth: "The image depicts import requests"

- **BLIP-2**: *"a flat icon of a bar chart on a page"* ✓(Semantic Match)
- **Florence-2 (Raster)**: *"The image is a graphic design of a bar graph. It is a rectangular icon with a red background and a white border. The bar graph is divided into four sections, each with a different color - blue, green, and white. The blue section is on the left side of the icon, the green section is in the middle, and the white section is at the bottom. The bars are arranged in a horizontal line..."* ✓(Detailed Visuals)
- **Idefics3**: *"The image shows a graphic representation... likely a document or report..."*

Analysis: This example highlights the capability of models to associate abstract geometric arrangements (rectangles, lines) with high-level concepts like "bar chart" or "import requests" (document metaphor). Both Raster and VLM baselines perform well here.

5.4.3 Failure Modes and Error Taxonomy

I categorize the observed errors into four types:

1. Geometric Hallucination

The model describes shapes that are not present. *Example*: Describing a simple square as "a square with a checkmark inside." *Cause*: Overfitting to common icon motifs. The model sees a square (often a checkbox) and probabilistically predicts "checkmark" based on training priors, ignoring the visual evidence.

2. Attribute Mismatch

The model identifies the object correctly but gets attributes wrong (color, stroke style). *Example*: "A red circle" when the circle is blue. *Cause*: The SPE's style embedding might be too coarse (binning continuous colors), or the model ignores the style tokens in favor of the stronger shape signal.

3. OCR Failure

The model fails to read text rendered as paths. *Example*: An icon of a button labeled "STOP" is described as "A rectangular button with some shapes inside." *Cause*: The SPE sees the letters 'S', 'T', 'O', 'P' as generic curves. It lacks a character-level recognition module (OCR).

4. Abstract Concept Failure

The model fails to map abstract shapes to their semantic meaning. *Example*: A stylized, abstract logo is described as "A collection of curved lines." *Cause*: The gap between geometry and semantics is too large. Without visual pre-training (like CLIP's image-text alignment), the model cannot infer the "meaning" of arbitrary abstract shapes.

5.5 Discussion

The empirical results provide strong evidence for the central hypothesis of this thesis (Section 1.3): **SVG captioning can be effectively modeled as a structured language understanding task**. The successful integration of the SPE with decoder-only LLMs demonstrates that treating geometric coordinates as a distinct modality—bridged to the textual domain via continuous embeddings—is a viable alternative to rasterization.

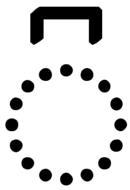
The performance differential between architectures reveals a surprising nuance. **Text-Only Qwen2-7B** technically achieves the highest raw **CLIPScore (32.30)**, slightly outperforming the SPE model (29.30). This suggests that for many icons, the LLM is capable of extracting significant semantic signal directly from the raw SVG code tokens, treating them as a "code understanding" task. However, this visual alignment comes at the cost of linguistic quality. As seen in the qualitative samples (Section 5.4.2), text-only models often produce fragmented, technically compliant but semantically vague descriptions (*e.g.*, "complex geometric pattern").

In contrast, the **SPE + Qwen2-7B** model achieves the highest **Composite Score (4.18)**, significantly surpassing the text-only baseline (3.95). By grounding the continuous geometry in a dedicated embedding space, the SPE acts as a regularizer, enabling the generation of fluent, human-readable captions that capture the *high-level semantics* (*e.g.*, "human figure", "cross") rather than just describing low-level primitives. While the text-only model "sees" the code well enough to score high on CLIP, the SPE model "understands" the image better in natural language, enabling it to bridge the gap to the way humans describe visual content.

However, the analysis of failure modes (Section 5.4.3) reveals the limitations of training *from scratch* on a relatively small dataset (90,000 samples). Unlike Vision Transformers (ViT) or CLIP, which are pre-trained on billions of image-text pairs, our SPE module was initialized randomly and learned to perceive geometry solely from the fine-tuning signal. The observed difficulties with abstract concepts and "hallucination" of common shapes are typical of this lack of large-scale visual pre-training. A dedicated "Vision-Language Pre-training" stage, aligning the SPE with a frozen image encoder on a massive corpus, would likely be the decisive step to close this gap.

5.6 Qualitative Analysis Gallery

This section provides an in-depth qualitative analysis of the model performances by examining eight representative samples from the test set. The selected examples cover a diverse range of visual complexity, spanning from iconic representations and simple geometric patterns to more abstract designs. This variety allows us to observe specific failure modes in baseline models—such as the hallucination of non-existent details or the misinterpretation of spatial relationships—while highlighting the strengths of our proposed SPE-based architectures. For each sample, we present the rendered SVG image alongside the ground truth caption and the varying descriptions generated by Florence-2, Qwen2-7B, Llama-3-8B, and our fine-tuned SPE variants (SPE+Qwen2 and SPE+Gemma). This side-by-side comparison offers tangible insight into how different architectures perceive and textualize vector graphics.



Sample 4: Circle with Square

Ground Truth: The image depicts a simple geometric shape, specifically a circle with a small square at the top. The circle is composed of a series of evenly spaced dots, which are uniformly distributed around its circumference. The square at the top of the circle is slightly smaller than the circle itself and is positioned directly above the center of the circle. The square is also made up of dots, similar to the circle, but it is oriented with its sides parallel to the horizontal and vertical axes of the image.

BLIP-2: a black and white icon of a clock

Florence-2: The image is a black and white icon of a ring. The ring is circular in shape and has multiple small black dots arranged in a circular pattern. At the top of the ring, there is a small black rectangle with a curved top. The background is white. The overall design is simple and minimalistic.

Idefics3: The image contains a series of dots arranged in a circular pattern. The dots are evenly spaced and form a continuous loop. The loop is symmetrical, with the dots forming a perfect circle. The dots are black, and the background is white. The image does not contain any text or other objects. The arrangement of the dots suggests a pattern that could be interpreted as a symbol or a mathematical representation.

BLIP-1-CPU:

Qwen2-7B: The SVG code describes intricate geometric shapes with precise dimensions. The primary focus is on a complex star-like figure made up of multiple overlapping circles and arcs. Here's the breakdown: **Star Shape**: At its core, there's a star pattern composed of 10-pointed stars formed by intersecting circles and arcs. Each point of the star is defined by a series of arcs that connect to form a symmetrical shape. **Circular Elements**: - There are four large circles arranged around the star pattern. - These circles have diameters ranging from 10 units to 15 units, creating a layered effect when combined with smaller concentric circles within them. **Smaller Circles/Arcs**: Numerous small circles and arcs are positioned inside these larger circles, contributing to the star's complexity. They are used to create the intricate patterns at each vertex of the stars. 4.

Sample 4: Circle with Square

Gemma-9b instruct: The image depicts several distinct shapes resembling simplified human figures. Each figure is outlined by black lines and lacks any internal details or shading. **Figure 1:** Located towards the top left, this figure has a rounded head and a slightly elongated torso. Its legs are short and stubby, ending abruptly. The arms are also relatively short and straight. **Figure 2:** Positioned to the right of Figure 1, this figure has a similar body shape but is more compact. Its head is smaller and its limbs are even shorter than those of Figure 1. **Figure 3:** Situated further down and to the right, this figure has a distinctive feature – a long, thin neck extending from the head. Its torso is somewhat wider than the previous two figures, and its limbs are proportionally longer.

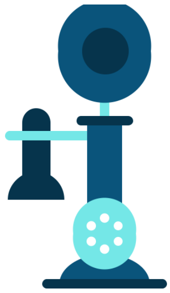
Llama-8B: The image depicts a chemical structure with a double bond between two carbon atoms, indicated by the straight line segment with a single bond on either side. The structure consists of four carbon atoms, with the first and last carbon atoms bound to hydrogen atoms, as indicated by the circles. The central two carbon atoms are bound to each other with a double bond, and they also have single bonds to other elements, which are not clearly visible in the image. The structure as a whole is closed, with the first and last carbon atoms connected, and the central two carbon atoms connected. The image provides a simple and visual representation of a double bond, making it easily understandable for a wide audience. The use of circles to represent hydrogen atoms is a common convention in chemical images, providing a quick and intuitive way to indicate the presence of these elements. The straight line segments with a single bond on either side of the double bond are a clear and unambiguous way to show the and bonds, making the structure easy to understand. The closure of the structure, with the first and last carbon atoms and the central two carbon atoms, is a critical feature of the structure, and the image makes this clear with the straight line segments connecting the appropriate and.

Gemma-9b-Quantized: The image depicts a complex arrangement of overlapping shapes, suggesting a fantastical landscape or cityscape. **Key Visual Elements: Central Figure:** A large, rounded shape resembling a creature or character dominates the center of the composition. It has a bulbous head and a curved, elongated body that extends downwards. The details of its features are obscured by the surrounding shapes. **Interlocking Shapes:** Numerous smaller, circular, and elliptical forms weave around the central figure, creating a sense of depth and intricacy. These shapes overlap and interlock, forming a dense network. Some have slightly pointed ends, adding to the organic feel. **Color Palette:** The entire image is rendered in shades of black and white, emphasizing the interplay of light and shadow created by the overlapping shapes. This monochrome palette gives the scene a stark and dramatic quality. The overall impression is one of dynamism and complexity.

Llama-9b-Quantized: An illustration showing the shape of the Earth's atmosphere.

SPE+Qwen2-7B: g two black circles and a black rectangle. The circles are positioned symmetrically on either side of the rectangle, and they are identical in shape and size. The rectangle is also black and is positioned centrally between the two circles. The overall composition of the image is symmetrical, with the circles and the rectangle aligned horizontally and vertically, creating a balanced visual effect.

SPE+Gemma-9B: The image depicts a single, stylized figure resembling a human form. It is drawn with a continuous, flowing line that creates a sense of movement and dynamism. The figure has a distinct elongated torso and limbs, with a somewhat abstract head shape. **Color:** The entire figure is black, creating a stark contrast against a likely white background. **Layout:** The figure is positioned centrally within the frame. It appears to be in a dynamic pose, possibly leaping or twirling. The lines that make up its body curve and bend, suggesting fluidity and energy. **Relationships:** The lack of facial features makes it difficult to interpret the figure's expression. However, the overall posture and the way the lines flow suggest a sense of grace and agility. **Overall Impression:** The image evokes a feeling of motion and elegance. Its simplicity and lack of detail allow for individual interpretation, inviting viewers to project their own thoughts and emotions onto the abstract form.



Sample 11: Vintage Telephone

Ground Truth: The image depicts a vintage telephone, characterized by its classic design. The telephone is depicted in a simplified, cartoon-like style, with a focus on its main components. The main body of the telephone is cylindrical and dark blue in color, with a circular dial at the top. The dial is also dark blue and has a central hole in the middle, which is a common feature of rotary telephones. Below the dial, there is a horizontal bar that extends from the main body, which is also dark blue. This bar is connected to a smaller, circular base that is light blue with white dots around its perimeter. The base is the lowest part of the telephone and appears to be resting on a flat surface.

BLIP-2: a blue microscope with a green base

Florence-2: The image is an illustration of a laboratory equipment. It consists of a blue-colored microscope with a round base and a cylindrical body. The body of the microscope is made up of two columns, one on top and one on the bottom. On the top column, there is a large circular object, which appears to be a test tube or a funnel. The object is connected to a smaller sphere with small holes in it. The sphere is placed on a flat base. The background is white. The overall design is simple and minimalistic.

Idefics3: The image depicts a simplified, abstract representation of a microscope. The microscope is composed of several geometric shapes. The main body is a large blue circle, which represents the base or the stand of the microscope. Above this circle, there is a smaller blue circle, likely representing the eyepiece or the objective lens. Connecting these two circles is a green horizontal bar, possibly indicating the arm or the stage of the microscope. The background is white, and there are small white dots scattered around the green bar, possibly representing dust or particles. The overall style is flat and minimalistic, with no shading or depth cues.

BLIP-1-CPU: a blue and green object with white dots

Qwen2-7B: The SVG illustration depicts a stylized geometric pattern composed of various colored shapes. It primarily features five distinct polygons, each with its own color scheme. **Polygon 1**: This polygon is located at the top right corner of the SVG canvas. It has four sides, two of which form a horizontal line segment that aligns with the y-axis coordinate 160. The other two sides are slanted diagonally outward from this horizontal segment to create a trapezoidal shape. The color used for this polygon is a light cyan shade (#116,231,231). **Polygon 2**: Found near the bottom center, this polygon resembles an irregular hexagon. Its vertices are connected by curved lines and straight segments. Two of its sides lie along the x-axis at coordinates 308 and 141. The curves create a rounded appearance towards the left side of the polygon. The interior color is a dark blue shade (#6,52,76). **Polygon 3**: Located at the top center of the SVG canvas, this polygon consists of three vertices forming an elongated triangle.

Sample 11: Vintage Telephone

Gemma-9b instruct: The SVG depicts a simplified, abstract representation of a microscope.

Components:

Base: A large, dark blue circle forms the base of the microscope. **Arm/Stand:** A smaller, dark blue shape, possibly representing the arm or stand, rises from the base. It has a curved top. **Stage:** A rectangular green bar sits horizontally across the middle, potentially representing the stage where specimens are placed. It's decorated with small white circles along its length. **Eyepiece/Tube:** A smaller, light blue oval shape is positioned above the arm, suggesting the eyepiece or tube. **Objective Lenses:** Two small, gray geometric shapes near the top right could represent the objective lenses.

Style:

The image utilizes flat colors and simple geometric shapes, creating a clean and modern aesthetic. The lack of detailed textures or shading gives it a flat design look.

Overall Impression:

While abstract, the arrangement of shapes and colors clearly evokes the form of a microscope, albeit in a stylized manner.

Llama-8B: The image depicts a simple graphic design involving a square with rounded corners, commonly known as a rectangle with rounded edges. The square is outlined with a solid black line, which gives it a clean and professional appearance. The corners of the square are slightly rounded, which adds a soft and smooth touch to the overall shape, making it more appealing to the eye. The sides of the square are of equal length, ensuring symmetry, and the corners are well-rounded, preventing sharp edges that could be potentially dangerous. The background of the image is a plain white color, which highlights the square's characteristics and makes it stand out. The square itself is a solid black color, which provides contrast and makes its edges and corners clearly visible. The image is undated, but the design of the square suggests a contemporary approach, where rounded corners are a common feature in many products and designs, providing a more comfortable and user-friendly experience. The image is digital and appears to be a screenshot or capture from a device with a high resolution, which allows for a crisp and clear representation of the square. The image is not signed or attributed to any individual or organization, suggesting it is a generic example or a stock image available for public use.

Gemma-9b-Quantized: The SVG renders a stylized image composed of several distinct geometric shapes.

Dominant Shapes:

Large Blue Circle: A prominent, dark blue circle occupies the lower central portion of the canvas. It appears to be slightly flattened at the top. **Green Rectangle:** A bright green rectangle sits horizontally across the middle of the image, overlapping the top portion of the blue circle. It has a series of small white dots running along its length. **Smaller Blue Shape:** To the left of the green rectangle, there's a smaller, teardrop-shaped element in a lighter shade of blue. It points downwards and slightly inwards towards the center. **Two Grey Shapes:** In the upper right corner, two grey shapes resemble stylized rocks or perhaps abstract clouds. They are irregular in form and positioned close together.

Composition:

The shapes are arranged in a somewhat loose and asymmetrical manner, creating a sense of movement and dynamism. The bright green rectangle acts as a visual anchor, contrasting with the cooler blue tones of the circle and teardrop.

Llama-9b-Quantized: The SVG code defines a simple drawing of a microscope. It is composed of basic shapes like circles and rectangles, colored in shades of blue, green, and grey.

Sample 11: Vintage Telephone (continued)

SPE+Qwen2-7B: The image depicts a simple, abstract design consisting of two primary shapes: a circle and a rectangle. The circle is positioned above the rectangle, and both shapes are colored in a gradient pattern. The circle is filled with a teal color that transitions from a darker shade at the top to a lighter shade towards the bottom. The rectangle below the circle is also filled with a similar gradient pattern, but it has a slightly different hue. It starts with a darker shade at the left end and transitions to a lighter shade towards the right end. The circle and rectangle are aligned vertically, with the circle placed directly above the rectangle. The shapes appear to be symmetrical, with the circle's center aligned directly above the rectangle's center. There is no text or additional elements present in the image, making it a minimalist composition focused solely on the interaction between the two shapes.

SPE+Gemma-9B: The image depicts a scene with several distinct shapes and colors. **Dominant Shapes: Large Blue Circle:** The largest element is a large blue circle located towards the bottom center of the image. This circle appears to be filled with a deep teal color. It has a slight curve at its top suggesting it might be part of a larger structure or object. **Green Rectangle:** A bright green rectangle sits horizontally across the middle of the image, slightly overlapping the blue circle. This rectangle seems to have rounded corners. **White Dots:** Scattered around the green rectangle are small white dots arranged in a somewhat symmetrical pattern. These dots seem to be placed intentionally, possibly representing some kind of decorative element or function within the image. **Smaller Blue Shape:** Towards the left side of the image, there's a smaller blue shape resembling a stretched oval or a teardrop. It has a smooth, curved form and sits slightly above the green rectangle. **Purple/Gray Shapes:** There are two smaller, purple/gray shapes positioned near the top right corner of the image. They appear to be more abstract and geometric in nature, with sharp angles and lines. **Color Palette: Blue:** Predominantly used for the large circular elements, anchoring the composition. **Green:** Highlights the central rectangle and contrasts with the surrounding blues. **White:** Appears as small dots that add visual rhythm around the rectangle. **Purple/Gray:** Provide secondary accents near the top right of the scene.

Sample 13: 50 Sign

Ground Truth: The image depicts the number 50 is written on the black color object.

BLIP-2: 50 km/h speed limit sign

Florence-2: The image is a black and white circular sign with the number 50 in the center. The number 50 is written in a bold, sans-serif font and is surrounded by a thin, black line. The line is crossed out in a diagonal direction, indicating that the number is not allowed. The sign appears to be a prohibition or restriction symbol.

Idefics3: The image depicts a round traffic sign with a black border and a white background. The sign features a large, bold number "50" in black, occupying the central portion of the sign. The number "50" is prominently displayed, with its height and width making it easily readable from a distance. The sign also includes a black diagonal line extending from the upper left corner to the lower right corner, forming an "X" shape. This diagonal line is positioned in such a way that it intersects the number "50" and crosses through the center of the sign.

BLIP-1-CPU: a black and white sign with the number 50

Qwen2-7B: The SVG code describes a complex, symmetrical design composed of several geometric shapes. It features four main sections: **Circular Arcs:** Two sets of circular arcs create overlapping rings with alternating white and black sections. Each set has two concentric circles that gradually expand outwards, forming a ring-like structure. - White sections: Circles start at point (256, 511) and grow outward. - Black sections: Circles begin at point (436, 436), creating a second, larger set of overlapping rings. **Curved Lines and Polygons:** These elements form intricate patterns within the central area of the image. They include: - Curved lines connecting various points to create a polygon shape near the center. - Another curved line segment starting from point (228, 195) to point (162, 238). **Triangle Pattern:** Located towards the right side of the SVG, this pattern comprises multiple small triangles connected by their vertices.

Sample 13: 50 Sign

Gemma-9b instruct: The image depicts a stylized black and white representation of a human eye. The iris is depicted as a large, rounded shape with concentric circles radiating outward from its center. These circles are interrupted by smaller, circular pupils within the larger iris. The pupil appears to be partially obscured by a thin, curved line that suggests a tear rolling down the cheek. The surrounding sclera, or white part of the eye, is represented by a series of smooth, flowing curves that create a sense of depth and dimension. The overall effect is one of both fragility and strength, highlighting the complexity and beauty of the human eye.

Llama-8B: The image depicts a simple geometric design consisting of a square with rounded corners, positioned at the center of the image. The square is outlined by a solid black line, and it is uniform in color, suggesting a white or transparent background. The corners of the square are slightly rounded, giving it a softer and more rounded appearance compared to a traditional square with sharp corners. The sides of the square are of equal length, and the overall shape is symmetrical, with the diagonal lines bisecting the square into four identical right triangles. The corners of the square are rounded such that the vertices are slightly rounded, and the sides are straight and of equal length, making the overall shape symmetrical. The square has a solid black outline, and its interior is empty, leaving it appear white or transparent. The corners of the square are rounded, and the sides are straight and of equal length, making the overall shape symmetrical. The square has a solid black outline, and its interior is empty, leaving it appear white or transparent. The corners of the square are rounded, and the sides are straight and of equal length, making the overall shape symmetrical. The square has a solid black outline, and its interior is empty, leaving it appear white or transparent.

Gemma-9b-Quantized: The SVG renders a complex scene depicting a three-dimensional sphere nestled within a larger circle. **Sphere:** The sphere itself is predominantly white and is formed by a series of smooth curves that transition seamlessly into one another. It occupies the central portion of the image, appearing to be slightly tilted towards the viewer. **Outer Circle:** Encasing the sphere is a dark gray circle. This circle acts as a backdrop and provides a clear contrast against the whiteness of the sphere. **Intersecting Lines:** Numerous intersecting lines emanate from the center of the outer circle, creating a network of pathways that seem to converge and diverge around the sphere. These lines are also black and add a sense of dynamism and complexity to the composition. **Additional Shapes:** Within the space between the sphere and the outer circle, there are several smaller shapes. Some appear to be abstract forms, while others resemble stylized leaves or petals. These shapes are also black and contribute to the intricate and multifaceted nature of the design.

Llama-9b-Quantized: A technical writer provides a precise description of an SVG scene.

Sample 13: 50 Sign (continued)

SPE+Qwen2-7B: The image depicts a simple geometric shape, specifically a circle divided into two equal parts by a straight line that runs from the top left to the bottom right. The circle is black, and the line is also black, creating a clear contrast against the white background. The division of the circle is symmetrical, meaning that each half of the circle is identical in size and shape. The circle itself is a closed loop without any additional features or decorations.

SPE+Gemma-9B: The image depicts a stylized black and white representation of a face within a larger circular frame. **Central Features: Eyes:** Two large, oval shapes positioned symmetrically above the center line of the circle represent the eyes. They have a slightly exaggerated, cartoonish appearance. **Nose:** A small, curved triangle situated between the eyes suggests the nose. **Mouth:** Below the nose, a gentle curve forms the mouth, hinting at a smiling expression. **Frame:** The entire composition is contained within a circular border, which appears to be thicker than the central features. This frame emphasizes the isolation and prominence of the face. **Style: Black and White:** The image utilizes a stark contrast between black and white, creating a graphic and minimalist aesthetic. **Geometric Shapes:** The face is constructed primarily from geometric shapes like circles, ovals, and triangles, contributing to its simplified and abstract nature. **Overall Impression:** The image evokes a sense of playfulness and innocence due to the rounded shapes and the gentle smile. The simplicity of the design makes it easily recognizable as a face, while the lack of detail allows for individual interpretation and imagination. It could be interpreted as a symbol of joy and unity.

Sample 16: Shopping Cart

Ground Truth: The image depicts a piece of luggage, specifically a suitcase. The suitcase is designed to be portable and is equipped with wheels, which are visible at the bottom of the image. The suitcase has a handle at the top, which is likely used for carrying the luggage. The handle is positioned at the top center of the suitcase, extending upwards and slightly backwards, giving it a handle-like appearance. The suitcase is rectangular in shape and has a flat top surface, which is typical for luggage designed to be easily stacked or placed on top of other luggage.

BLIP-2: a black and white illustration of a suitcase

Florence-2: The image is a black and white icon of a shopping cart. The cart has a rectangular shape with a handle on top and four wheels at the bottom. The handle is curved and appears to be made of metal. The body of the cart is rectangular in shape and has a small opening on the front. The wheels are also curved and have a small hole in the center. The background is plain white.

Idefics3: The image depicts a suitcase with a handle and wheels. The suitcase is upright, with the handle extended upwards. The wheels are visible at the bottom of the suitcase, indicating that it is designed for mobility. The suitcase appears to be closed, with no visible details of its contents or exterior design. The image is minimalist, focusing solely on the essential features of the suitcase.

- **Handle and Wheels**: The presence of a handle and wheels suggests that the suitcase is designed for ease of transport, making it suitable for travel. The handle allows for easy pulling, while the wheels facilitate smooth movement over various surfaces.

BLIP-1-CPU: shopping cart icon vector

Sample 16: Shopping Cart

Qwen2-7B: The SVG code describes two distinct shapes drawn using paths within its structure.

Firstly, there is a smooth curve starting at point (447, 448) that moves towards point (434, 435), then it smoothly transitions to point (434, 411) and continues until it reaches point (459, 435). This creates an ellipse-like shape with a slight curvature. The color used for this shape is black ('fill:0,0,0') without any stroke outline ('stroke:None'). It appears to be a symmetrical figure with its center at approximately (447, 448).

Secondly, another path starts at point (512, 421) and extends horizontally to point (479, 158), forming a horizontal line segment which has been bent into an arc shape resembling a large semi-circle. The endpoints of this arc meet at point (59, 44). The fill color for this area remains black, but it's filled with a white border ('stroke:None'), making the border invisible. The opacity is set to 1 indicating full visibility.

Gemma-9b instruct: The image depicts a stylized human figure, rendered in black against a white background. The figure is composed of several distinct shapes:

- **Torso:** A large, rounded rectangle forms the central part of the body.
- **Head:** A smaller oval shape sits atop the torso.
- **Arms:** Two elongated, slightly curved lines extend from either side of the torso.
- **Legs:** Two similar lines, slightly shorter than the arms, emerge from the bottom of the torso.

The figure's proportions are exaggerated, with long limbs and a small head. The overall impression is one of simplicity and abstraction, focusing on basic geometric shapes to represent the human form.

There are also two additional elements within the image:

- **Left Hand:** A smaller, rectangular shape positioned at the end of the left arm.
- **Right Foot:** A circular shape located at the end of the right leg.

Sample 16: Shopping Cart (continued)

Llama-8B: The image depicts a stylized representation of a lightning bolt. The lightning bolt is depicted in a two-dimensional, flat style, with the structure consisting of a straight vertical line segment, a curve to the right and then back to the left, and another straight vertical line segment. The lightning bolt has a smooth, continuous curve, with no sharp edges or corners. The base of the lightning bolt, where it meets the vertical line segments, is slightly wider than the top, giving it a slight taper. The overall shape is symmetrical, with the curve on one side being a mirror image of the curve on the other side. The lightning bolt is outlined with a solid black line, which provides a clear distinction between the object and its background, making it stand out prominently. The underlying surface is also black, which complements the outline, making the object easily recognizable. The image presents a minimalistic approach, with only the essential elements included, such as the straight lines and curves that make up the structure of the lightning bolt. There are no additional details or decorations, keeping the focus on the simplicity and elegance of the design. The lightning bolt is depicted against a black background, which allows it to shine through, making it the primary visual element of the image.

Gemma-9b-Quantized: The SVG renders a complex, detailed drawing of a stylized bird in flight. The bird's body is depicted as a flowing, rounded shape that appears almost like a teardrop, with a long, slender tail trailing behind it.

The bird's wings are outstretched, forming broad, curved shapes that suggest powerful motion. Its head is small and round, tucked into its chest as if preparing to dive. There's a subtle suggestion of a beak near the front of the bird's head.

The entire bird is rendered in solid black, giving it a bold and dramatic presence against a hypothetical white background. There are no gradients or textures applied to the bird, emphasizing its simple yet elegant form.

Beneath the bird, there's a smaller, more abstract element. It resembles a series of interconnected curves and loops, possibly representing swirling air currents or the bird's wake. This element also uses the same solid black fill as the bird.

The overall composition suggests a dynamic scene of a bird soaring through the air, leaving a trail of disturbed energy behind it.

Llama-9b-Quantized: A simple scene with two lines and four curves.

Sample 16: Shopping Cart (continued)

SPE+Qwen2-7B: The image depicts a simple, single symbol consisting of two distinct parts. The symbol is enclosed within a square frame. The upper part of the symbol features a horizontal line that extends across the top of the square, while the lower part consists of a vertical line that extends from the bottom of the square. Both lines intersect at the center of the square, forming an "X" shape. The entire symbol is black, and the background is white, making the symbol stand out clearly against the background.

SPE+Gemma-9B: The image depicts a stylized human figure composed of interconnected geometric shapes.

Body: The main body is formed by a series of curved lines that suggest a rounded torso and limbs. The upper body has a more elongated shape, hinting at shoulders and a head.

Head: A distinct circular shape at the top represents the head. It's slightly offset to the right, giving the figure a dynamic pose.

Legs: Two downward-pointing triangles represent the legs, connected to the torso by a wider, curved line.

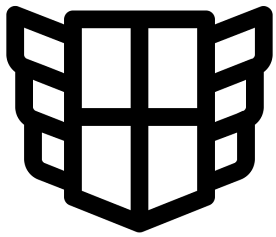
Arms: Two shorter, narrower triangles extending from the sides of the torso suggest arms. They lack defined hands or wrists.

Details: There are several small, rectangular shapes scattered around the figure, possibly representing facial features or accessories. However, due to their simplicity, it's difficult to discern their exact purpose.

Color: The entire figure is black against a white background, creating a strong contrast that emphasizes its form.

Layout: The figure is positioned centrally within the frame, with no other elements present. This focus on the single figure creates a minimalist and abstract composition.

Overall Impression: The image evokes a sense of movement and dynamism through the arrangement of the shapes. The simplicity of the design keeps the focus on the interplay between the geometric forms.



Sample 24: Shield Icon

Ground Truth: The image depicts a simple, abstract design that resembles a shield or a box with a grid pattern. The design consists of a central square shape with four smaller squares arranged symmetrically around it. Each of these smaller squares is divided into two equal parts by a horizontal and a vertical line, creating a total of eight smaller rectangles within the larger square.

BLIP-2: a black and white logo of a shield with wings

Florence-2: The image is a black and white line drawing of a shield or shield with four panels. The shield is divided into four sections, each with a square in the center. The panels are arranged in a grid-like pattern, with each panel having a small square in between them. On either side of the shield, there are two curved lines that form the shape of a cup or mug. The overall design is simple and minimalistic.

Idefics3: The image is a minimalist, two-dimensional representation of a shield. The shield is depicted in a stylized, geometric form, with a symmetrical design. The shield is divided into four quadrants, each containing a square grid pattern. The grid pattern within each quadrant consists of small, evenly spaced squares. The shield's outline is simple and clean, with straight lines forming the edges of the shield. The overall shape of the shield is a perfect square, with equal sides and angles. The shield is positioned centrally within the image, with no background or additional elements present.

BLIP-1-CPU: a black and white logo with a large, rectangular shape

Sample 24: Shield Icon

Qwen2-7B: The SVG code describes a complex polygon with multiple segments forming intricate shapes within a 2D plane.

The primary structure is a large pentagon made up of several interconnected lines and arcs. It starts with a smaller pentagon at its center, followed by outer layers that expand outward symmetrically. This central pentagon has sides that are slightly curved towards each other, creating a star-like appearance when viewed from afar.

Adjacent to this central pentagon, there's a larger pentagonal shape formed by extending the original edges outward. The outer edges have slight curves as well, making the overall figure look like a series of overlapping pentagons.

In between these two pentagonal structures, there are additional elements resembling triangles and trapezoids. These forms are created using straight lines and arcs connecting various points around the central pentagon.

A key feature is the presence of a thin strip along the right side of the main pentagonal shape, which consists of horizontal lines and curved segments. This strip adds depth and complexity to the design.

Throughout the entire drawing, the color is black ('#000000'), giving the shapes a crisp and solid appearance.

Gemma-9b instruct: The image depicts a complex cityscape with distinct architectural elements. A prominent feature is a large, rectangular building at the center, resembling a modern office tower. It has multiple floors represented by horizontal lines and appears to have a slight upward slant towards its top.

Flanking this central structure are smaller buildings, each with unique designs. Some resemble narrow towers while others appear more like elongated rectangles. These buildings are scattered across the image, creating a sense of depth and a bustling urban environment.

Numerous windows punctuate the facades of these buildings, adding visual interest and suggesting activity within. The windows are mostly uniform in size and shape, contributing to a somewhat ordered aesthetic.

Interspersed among the buildings are stylized trees, depicted as simple triangular shapes with rounded tops. They add a touch of greenery and contrast against the stark lines of the cityscape.

Overall, the image conveys a dynamic urban scene, filled with intricate details and a sense of movement and energy.

Llama-8B: The image depicts a simple line drawing of a speech bubble. The speech bubble is outlined with a solid black line, and it has a slightly rounded triangular shape with a broad base and a sharp point at the top. The inside of the speech bubble is empty, indicating that no actual words or text are present. The outline of the speech bubble is smooth and continuous, with no visible gaps or irregularities. The background of the image is also black, which makes the speech bubble stand out prominently.

Explanation: The speech bubble depicted in this image is designed for visual representation and understanding, rather than for the transmission of specific verbal content. The bubble's structure suggests that it is intended to contain words or a message, but the interior is empty, providing a blank canvas for additional information.

Gemma-9b-Quantized: The SVG depicts a whimsical scene reminiscent of a medieval tapestry or a child's drawing. A large central figure dominates the composition. This figure appears to be a knight or warrior, clad in armor and wielding a sword. Its body is formed by a series of connected curved lines, suggesting movement and dynamism.

Surrounding the central figure is a complex network of smaller shapes and figures. Some resemble houses, others trees, and some seem like fantastical creatures. These elements are arranged in no particular order, creating a sense of chaos and vibrancy. The overall impression is one of a bustling marketplace or a mythical realm.

The color palette is predominantly black and white, with subtle hints of grey. This monochrome scheme lends a timeless quality to the image, further emphasizing its archaic feel.

Llama-9b-Quantized: The scene shows an example of the intersection between two circles.

Sample 24: Shield Icon (continued)

SPE+Qwen2-7B: The image depicts a simple geometric figure consisting of two main components: a large triangle and a smaller triangle inside it. The larger triangle is positioned above the smaller one. Both triangles are outlined by straight lines, and they are filled with black color. The larger triangle has three sides, while the smaller triangle has two sides that are part of the larger triangle's base.

The larger triangle is oriented with its base pointing upwards, and the smaller triangle is positioned below it, sharing part of the base of the larger triangle. The smaller triangle is also oriented with its base pointing downwards, creating a V-shape within the larger triangle. The two triangles share a common vertex at the top of the larger triangle, where the smaller triangle begins.

The overall structure resembles a double triangle, with the smaller triangle nested inside the larger one. The lines forming the triangles are straight and sharp, indicating a clean, precise design. The colors used are monochromatic, with both triangles colored in black, which makes them stand out against any background.

SPE+Gemma-9B: The image depicts a stylized city scene composed of numerous rectangular buildings arranged in a somewhat chaotic yet organized manner.

Buildings:

- The buildings are primarily represented by rectangles of varying sizes and orientations. Some are taller than others, creating a sense of depth and variation.
- Many buildings have slightly rounded edges, giving them a softer, more organic appearance.
- There is a clear emphasis on rooftops, which are often depicted with distinctive shapes. Some feature curved lines that suggest chimneys or other architectural details.

Layout and Composition:

- The buildings cluster together, forming distinct blocks within the cityscape.
- While there's no discernible grid pattern, the arrangement suggests a bustling urban environment.
- There's a sense of movement implied through the angles and directions of the building outlines.

Color:

- The entire image is rendered in black and white, emphasizing the geometric forms and shadows cast by the buildings.

Other Details:

- There are several smaller, triangular shapes scattered throughout the composition, possibly representing trees or streetlamps. These add visual interest and detail to the cityscape.
- At the bottom center of the image, there's a wider, horizontal rectangle that could represent a road or a river.

Sample 25: Tablet with Dollar

Ground Truth: The image depicts a stylized, minimalist design of a tablet. The tablet is held in a hand, with the hand and tablet forming a central focus of the image. The tablet is black, and the hand holding it is also black, suggesting a monochromatic color scheme. The tablet's screen displays a dollar sign ($) which is white and positioned centrally.

BLIP-2: a hand holding a tablet with a dollar sign on it

Florence-2: The image is a black and white icon of a tablet with a dollar sign on the screen. The tablet is held in a hand, with the fingers slightly curled around the edges of the screen, as if it is being held by someone's hand. The dollar sign is in the center of the tablet, with a white background and a black outline. The image is simple and minimalistic, with no other elements or design elements.

Idefics3: The image depicts a tablet with a black border and a white screen. On the screen, there is a dollar sign ($), which is white and positioned centrally. The tablet is held by a hand, which is also black. The hand is positioned in the lower right corner of the image, with the tablet resting in the palm of the hand. The background of the image is white, providing a stark contrast to the black tablet and hand. The image is simple and minimalistic, focusing on the tablet and the dollar sign without any additional elements or distractions.

BLIP-1-CPU: a hand holding a tablet with a dollar sign on it

Sample 25: Tablet with Dollar

Qwen2-7B: The SVG rendering depicts a complex geometric pattern. A series of interconnected lines create multiple shapes within the canvas, forming various enclosed spaces that vary in size and orientation.

At the top left corner, there is a smaller rectangle with rounded corners. It has dimensions approximately 6 units wide by 4 units high. Its edges are slightly curved at the corners, giving it a soft appearance. This shape is positioned along the edge of the larger canvas area.

Moving to the right side, a larger rectangular shape appears, extending horizontally across about half of the canvas width. This rectangle also features rounded corners but with more pronounced curvature compared to the previous one. Its dimensions are roughly twice as large, measuring around 12 units wide by 4 units tall.

Below these rectangles, a cluster of smaller triangles emerges. These triangles are positioned closer together, creating a dense arrangement. Each triangle shares its base with another, forming a continuous line. The height of each triangle varies slightly, adding a dynamic element to this section.

A zigzag-like structure extends from the bottom portion of the larger rectangle towards the bottom part of the canvas. This zigzag consists of sharp angles and varying lengths, contributing to a sense of movement and irregularity.

Gemma-9b instruct: The SVG depicts a stylized human figure within a larger geometric composition.

Figure:

- **Body:** The figure is primarily represented by a series of connected curves and lines forming a simplified silhouette. It has a rounded head, a slightly elongated torso, and narrow legs.
- **Arms:** The arms are depicted as straight lines extending from the sides of the torso. They appear to be raised slightly.
- **Head:** The head lacks detailed features, but its shape suggests a rounded form. There's a slight curve at the top that could represent hair.
- **Legs:** The legs are relatively short and thin, ending in small points.

Geometric Background:

- **Rectangle:** A large rectangle spans the bottom portion of the image, creating a sense of depth and grounding the figure.
- **Curves:** Several curved lines intersect with the rectangle, forming a complex pattern that adds visual interest. These curves also interact with the figure, creating a dynamic interplay between the organic shape of the body and the geometric background.

Colors:

- The entire image is rendered in shades of black and white. This creates a stark contrast that highlights the shapes and forms.

Overall, the image evokes a minimalist aesthetic.

Sample 25: Tablet with Dollar (continued)

Llama-8B: The image depicts a symbol that is commonly associated with privacy and security. The symbol consists of a circle with a solid line intersecting it at two points, forming a segment of a circle. The circle is enclosed within a square, which is also a common emblem for privacy and security. The circle-squared structure is often used to represent a system or device that is designed to protect personal information or ensure security, such as a password manager, an encrypted communication app, or a biometric authentication tool. The circle in this symbol is typically black, indicating confidence or certainty, while the square is usually white, suggesting neutrality or objectivity. The solid line that connects the circle to the square is straight and smooth, implying reliability or trustworthiness. The segment of a circle formed by the line is symmetrical, suggesting balance or fairness, while the entire circle is complete, suggesting wholeness or integrity. The circle-squared symbol is often used to assure individuals that their personal data is safe and secure, and that they can trust the system or device they are using to protect their privacy. The symbol is also a visual representation of the idea that security and privacy go hand in hand, and that both are essential for a healthy digital lifestyle.

Gemma-9b-Quantized: The SVG renders a complex scene depicting a stylized bird in flight against a dark background.

The bird's body is predominantly black, forming a teardrop shape that extends upwards towards the tail. The head is more rounded and features a small beak at its tip. Feathered details are subtly suggested by lines and curves across the bird's wings and body. The wings themselves are outstretched and angled downwards, creating a sense of dynamic movement.

Beneath the bird, there's a smaller, abstract element resembling a swirling cloud or plume. This element is also primarily black but has lighter shades interspersed within it, suggesting texture and depth. A thin white line outlines the bottom edge of this element, adding further definition.

Adding to the complexity, the composition includes a series of overlapping geometric shapes in muted tones of gray and brown. These shapes create a subtle sense of landscape or terrain beneath the bird and cloud.

Llama-9b-Quantized: The 3D Scene

Sample 25: Tablet with Dollar (continued)

SPE+Qwen2-7B: The image depicts a simple, minimalistic design featuring two black circles arranged vertically against a white background. The circles are positioned side by side, with one slightly taller than the other. The larger circle is located above the smaller one. Both circles have a uniform thickness and are filled with a solid black color. The smaller circle is placed directly below the larger one, creating a slight overlap where they touch. The overall composition of the image is symmetrical, with the vertical alignment of the circles being the focal point.

SPE+Gemma-9B: The image depicts a stylized human figure with several distinct elements.

Figure:

- The central figure is depicted in a somewhat abstract manner. Its torso is represented by a thick, black rectangle with rounded edges.
- Two smaller rectangles extending from this torso suggest arms, though they are simplified and lack hands.
- At the top of the torso, another rectangle forms the head, also lacking detailed features like eyes or a nose.

Additional Elements:

- A series of curved lines emanating from the bottom of the torso represent legs, flowing outwards and then curving upwards slightly before disappearing. These lines create a sense of movement or fluidity.
- **Geometric Shapes:** Scattered around the figure are various geometric shapes – circles, squares, and triangles – in black. These seem to act as decorative elements or possibly symbolic representations.
- **Background:** The background is white, making the black figure and its accompanying shapes stand out prominently.

Overall Impression: The image conveys a sense of minimalism and abstraction. The figure lacks realistic details, relying instead on simple shapes to convey a basic human form. The scattered geometric shapes add a touch of complexity and intrigue, inviting viewers to interpret their meaning. The overall impression is one of simplicity, yet there is enough variety to keep the composition engaging.

Sample 21: Import Requests

Ground Truth: The image depicts import requests

BLIP-2: a flat icon of a bar chart on a page

Florence-2: The image is a graphic design of a bar graph. It is a rectangular icon with a red background and a white border. The bar graph is divided into four sections, each with a different color - blue, green, and white. The blue section is on the left side of the icon, the green section is in the middle, and the white section is at the bottom. The bars are arranged in a horizontal line, with the blue section on the right side and the green and white sections on the bottom left side. The overall design is simple and minimalistic.

Idefics3: The image depicts a graphical representation of a bar chart. The chart consists of two main sections: a horizontal bar chart and a vertical bar chart. **Horizontal Bar Chart:** The horizontal bar chart is positioned at the top of the image. It features two bars, each colored in red and gray. The red bar is shorter than the gray bar. The horizontal bar chart is likely representing data across different categories, with the red bar indicating a smaller value and the gray bar indicating a larger value.

BLIP-1-CPU: a red and white icon with a bar chart

Qwen2-7B: A symmetrical architectural design consisting of two identical sections separated by a narrow vertical space. The main structure is composed of five stacked horizontal rectangles with different shades of grey, alternating between a lighter grey and a darker grey. The upper section features two thinner rectangles slightly shifted towards each other horizontally, forming a V-shape along their top edges when viewed from above. This creates a small triangular space at the top that appears to be cut out but is filled with a bright red color representing a door. The middle section contains four rectangles stacked vertically with the same light and dark grey alternation as the outer structure. At the bottom right corner of this section, there's a smaller rectangle attached to the larger one, giving it an additional shape resembling a rounded corner. Below these two sections lies a series of three rectangles that extend further down, with the leftmost and rightmost rectangles having rounded corners, creating a subtle arch-like appearance across the entire bottom part of the design. Each rectangle has thin white borders outlining them, providing clear separation and definition within the design. The entire drawing appears to have a uniform style throughout its elements.

Sample 21: Import Requests

Llama-8B: The image depicts a simple line drawing of a dog with a black coat. The dog is positioned in a standing position, with its front legs straight and its back legs bent slightly, giving it a crouching appearance. The dog has a long snout, rounded nose, and prominent lips, characteristic of many breed of dogs. Its eyes are depicted as circular shapes, set well apart, and its ears are long and pointed, tapering at the tips, a feature common in several dog breeds. The dog's coat is solid black, covering its body evenly, and its paws are dark in color, matching its fur, suggesting nail polish or other coating. The background of the image is white, providing a clean and bright appearance to the picture. The lines used to draw the dog are thin and smooth, giving it a sleek and symmetrical look, while still maintaining a distinct and recognizable shape. The image is unscaled, which means that the dimensions of the picture are not altered, preserving the original size and proportion of the elements. The canvas on which the image is drawn is rectangular, with a smooth edge, and the pixels are uniformly spaced, ensuring a sharp and clear visual representation.

Gemma-9b instruct: The SVG renders a graphic of a stylized smartphone. The main body of the phone is represented by a large, rectangular shape filled with a light grey color. This rectangle extends from the left to right edge of the image, forming the top portion of the phone's screen. Below this rectangular area, there is a smaller, solid red rectangle positioned at the bottom right corner. This likely represents the phone's battery indicator. On the upper part of the phone, within the larger light grey rectangle, there are several horizontal lines with a dark blue fill. These lines suggest navigation bars or app icons on the phone's screen. Further down, there are a series of small white rectangles stacked vertically near the center of the phone. These could represent app notifications or messages. There are additional smaller elements like a curved line at the top of the phone and vertical lines below the central white rectangles. These details add texture and visual interest to the overall design.

Sample 21: Import Requests (continued)

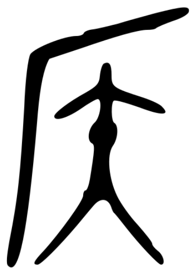
Gemma-9b-Quantized: The image depicts a simple landscape with a bright white rectangle forming the background. On this background, there is a light gray rectangular shape spanning the bottom third of the image. Above it, slightly offset to the left, is a dark blue, vertical rectangle. A prominent feature is a curved red shape extending diagonally from the top right corner to the bottom left. It appears as if it's being drawn by a hand, with curves and gentle bends. This red shape has a smaller, white section at its base, also curved, resembling a leaf or a petal. There are several smaller, lighter gray rectangles clustered together near the center of the image, each with slight variations in their positions and sizes. These appear like scattered stones or pebbles. Further down the image, there are more of these gray rectangles, creating a continuous line along the bottom edge. On the far right side of the image, there are two smaller, pale blue rectangles stacked vertically. They resemble miniature buildings or towers.

Llama-9b-Quantized: The picture above is an SVG scene, which can be interpreted as follows: The top line defines the size of the scene. It contains four numbers separated by commas and enclosed in square brackets, *e.g.*, width="640" height="512" viewBox="0 0 640 512". The second line specifies the fill color for all shapes drawn in the scene (the default color is black). The next two lines define the stroke color for each shape's outline (the default color is white) and its thickness (the default value is one pixel). The remaining lines specify different types of graphical elements such as circles, rectangles, etc.. Each element has its own set of properties including position coordinates, dimensions, colors, etc... In order to accurately describe this complex scene, we need to use a formal language called "markup language". This allows us to represent the structure of documents using tags instead of plain text. One example of markup language is XML which stands for Extensible Markup Language. Another popular choice among programmers today is HTML5. So now let me show you how to create your very own web page using HTML5!

Sample 21: Import Requests (continued)

SPE+Qwen2-7B: The image depicts a rectangular shape divided into two distinct sections by a horizontal line. The top section is colored light gray, while the bottom section is colored dark green. There is a vertical line that divides the rectangle horizontally, creating two equal parts. The top part of the rectangle has a lighter shade of gray, and the bottom part has a darker shade of green. The division between the two colors is sharp and clean, indicating a clear separation between the two sections.

SPE+Gemma-9B: The image depicts a stylized user interface element resembling a dashboard or control panel. **Background:** The main background is a light gray rectangle that occupies the majority of the space. This suggests a clean and minimalistic design. **Foreground Elements: Large Rectangle:** A prominent rectangular shape dominates the upper portion of the image. It's filled with a dark blue color and has a slight width, suggesting it might represent a navigation bar or a header. **Horizontal Stripes:** Below the large blue rectangle, there are several horizontal stripes of a lighter gray color. These stripes create a visual division within the area, possibly indicating different sections or categories. **Red Shape:** A curved, red shape sits prominently towards the left side of the image. It resembles a smooth wave or a stylized droplet, adding a touch of dynamism to the otherwise structured composition. **Multiple Small Rectangles:** Scattered across the bottom half of the image are numerous small rectangles of varying shades of blue and gray. These could represent individual items, buttons, or data points within the dashboard. They are arranged in a somewhat haphazard manner, suggesting a collection of interactive elements. **White Shapes:** Several white shapes, resembling thin bars or rectangles, are positioned amongst the other elements, adding contrast and spacing.



Sample 22: Human Silhouette

Ground Truth: The image depicts a stylized figure that appears to be a human silhouette. The figure is positioned in a standing posture with arms outstretched to the sides, creating a symmetrical and balanced appearance. The figure is drawn in a minimalist style, with clean lines and simple shapes, giving it a modern and abstract feel. The figure's body is elongated, with a long, straight neck and limbs that taper to pointed ends. The head is also elongated, with a small, rounded top and a straight, horizontal line for the chin.

BLIP-2: a chinese symbol for a person standing on a ladder

Florence-2: The image is a black and white line drawing of a person standing on their hind legs. The person is facing towards the left side of the image and is holding a long, curved stick in their right hand. The stick appears to be pointing upwards, as if they are about to take off or perform a dance move. The figure is outlined in black and has a simple, minimalist design. The background is white, making the figure and stick stand out.

Idefics3: The image features a stylized, black, abstract character. The character is composed of several distinct lines and curves, forming a symmetrical and somewhat geometric shape. The lines are thick and bold, creating a strong visual impact. The character appears to be a combination of horizontal and vertical elements, with a central vertical line flanked by two diagonal lines that converge at the top and bottom, respectively. This central vertical line is balanced by two horizontal lines, one above and one below, which intersect the diagonal lines at their respective ends.

BLIP-1-CPU: a black and white silhouette of a person jumping

Qwen2-7B: A complex shape is formed by various curves and lines. The primary focus of this SVG graphic seems to be a large, irregularly shaped blob that appears to have been created through a series of Bezier curve manipulations. The overall structure starts with a central region that gradually expands outward, creating a sense of depth and dimensionality. This main body features multiple protruding parts and concave sections, giving it an organic or abstract appearance. At its core, there's a dense cluster of curved lines that form intricate patterns. These lines intertwine and overlap each other, resulting in a visually engaging texture. The lines vary in thickness and curvature, adding complexity and interest to the design. As we move outward from the center, the shape begins to expand into larger, more open regions. There are several disconnected loops and segments that seem to flow seamlessly together, maintaining a cohesive visual rhythm despite their seemingly random arrangement. The shape also includes some sharp corners and angles, which contrast with the smooth curves present elsewhere in the drawing. These angular elements contribute to a sense of dynamic movement and break up the otherwise continuous flow of the design. Throughout the entire shape, there are subtle variations in how the curves interact with one another, creating interesting intersections and overlaps.

Sample 22: Human Silhouette

Gemma-9b instruct: The image depicts a single, stylized, black figure resembling a human form. It has a flowing, elongated body with a slightly rounded head and tapering limbs. The arms extend outwards from the torso, while the legs converge towards the bottom, creating a sense of movement or dynamism. The figure is entirely black, contrasting sharply against a presumably white background. This stark contrast emphasizes its shape and contours. There's a sense of fluidity and abstraction in the figure's design, suggesting a minimalist interpretation of the human form.

Llama-8B: The image depicts a stylized lightning bolt with a curved, zigzagging path and a sharp, pointed tip. The lightning bolt is rendered in black, which stands out against the white background, making it the focal point of the image. The curve of the lightning bolt is pronounced, giving it a dynamic and energetic appearance. The tip of the lightning bolt is sharply pointed, adding to its agility and speed. The entire object is symmetrical, with each half of the curve being a mirror image of the other, creating a smooth and continuous line. The background of the image is clean and clear, allowing the lightning bolt to stand out prominently. The object is placed centrally, filling most of the frame, making it the primary subject of the image. The image is rectangular, with the lightning bolt taking up the majority of the space, leaving little room for other elements. The object is positioned directly, with no other objects or details visible in the background, making it the sole focus of the image. The image is undated, but the design and technology suggest a modern creation, likely within the last few decades. The object is synthetic, made from a material that can be easily shaped and designed, such as plastic or metal.

Gemma-9b-Quantized: The SVG renders a single, intricate black silhouette against a transparent background. The shape resembles a stylized bird in flight, its wings outstretched and feathers intricately detailed. The head is small and rounded, blending seamlessly into the long neck that curves gracefully upward. The body is compact and muscular, suggesting power and agility. Its most striking feature is the pair of expansive wings, each meticulously crafted with a multitude of overlapping feathers that create a sense of depth and movement. The tail is short and pointed, adding to the overall impression of dynamism. The absence of color emphasizes the purity of the form and allows the viewer's imagination to fill in the details. The stark black lines create a powerful contrast against any background it might be placed upon.

Llama-9b-Quantized: The curve is an example of a Bézier curve with four control points (M, C1, C2, and L). The curve begins at M and ends at L. At any point between M and L, we can compute the tangent vector to the curve by computing the derivative of the curve as a function of t. The answer should be in plain text. • I've never seen such a question before. You are asking for a complete solution but there's no information about how good the solution should be. There's also not much of a hint on what a good solution would look like. It's just "I don't know where to start" – Timtech Jan 31 '14 at 12:43 • @Timtech This is a standard format for a challenge problem that has been used in many math classes for decades. The goal isn't necessarily to solve it, but rather to learn something from attempting to do so. (It's called a "pencil-and-paper problem" for this reason.) If you're curious about it, check out the other questions tagged [poetry] or [riddle]. – Kevin Jan 31 '14 at 13:00 • @Timtech I've added a comment above.

Sample 22: Human Silhouette (continued)

SPE+Qwen2-7B: The image depicts a stylized letter "A" in black color. The letter is designed in a modern, bold font with a unique twist. The letter "A" is presented in a cursive style, with a smooth, flowing curve that gives it a dynamic appearance. The design of the letter is asymmetrical, with the top part of the "A" being more elongated than the bottom part. The letter is oriented vertically, with the top pointing upwards and the bottom pointing downwards.

SPE+Gemma-9B: The image depicts a single, stylized figure resembling a human form. It is drawn with a continuous, flowing line that creates a sense of movement and dynamism. The figure has a distinct elongated torso and limbs, with a somewhat abstract head shape. **Color:** The entire figure is black, creating a stark contrast against a likely white background. **Layout:** The figure is positioned centrally within the frame. It appears to be in a dynamic pose, possibly leaping or twirling. The lines that make up its body curve and bend, suggesting fluidity and energy. **Relationships:** The lack of facial features makes it difficult to interpret the figure's expression. However, the overall posture and the way the lines flow suggest a sense of grace and agility. **Overall Impression:** The image evokes a feeling of motion and elegance. Its simplicity and lack of detail allow for individual interpretation, inviting viewers to project their own thoughts and emotions onto the abstract form.

Chapter 6

Conclusions and Future Work

This thesis has explored a novel approach to SVG captioning that treats vector graphics as structured symbolic data rather than raster images. Initially, I investigated the feasibility of fine-tuning Large Language Models (LLMs) using Low-Rank Adaptation (LoRA) on raw SVG code, establishing a baseline for valid text generation. Building on these findings, I integrated a **SVG Path Embedder (SPE)** with decoder-only LLMs effectively combining the discrete geometric encoding of the SPE with the semantic reasoning of the LLM. By integrating the SPE with parameter-efficient fine-tuning (LoRA), I have demonstrated that it is possible to generate accurate, human-readable captions directly from the geometric definition of SVG icons, without rasterization. This final chapter synthesizes the research journey, revisits the main contributions, discusses limitations and ethical considerations, and outlines promising directions for future work.

6.1 Summary of the Research Journey

The motivation for this thesis emerged from a practical observation: while raster-based image captioning has matured significantly (thanks to models like BLIP-2, LLaVA, and Florence-2), the vast ecosystem of vector graphics–icons, logos, technical diagrams, UI mockups–has remained underserved. Standard vision-language models require rasterization, a process that discards the rich structural information embedded in SVG markup and introduces resolution artifacts. I posed a foundational question: *Can a language model learn to "read" vector graphics natively?* The hypothesis was that SVG, being an XML-based symbolic representation, lies closer to the domain of text than to natural images, and thus Large Language Models–with their capacity for symbolic reasoning and text generation–might be well-suited to this task.

However, naive approaches (feeding raw SVG code to an LLM) failed due to tokenization mismatch and the loss of geometric structure. This motivated the adoption of the **SVG Path Embedder (SPE)**, a neural module that maps discrete SVG commands and coordinates into a structured embedding space. The SPE treats vector graphics as a specialized language, preserving the sequential nature of drawing commands while enabling the model to learn domain-specific token representations.

To adapt pre-trained LLMs to this new visual modality without prohibitive computational cost, I employed Low-Rank Adaptation (LoRA), which introduced trainable low-rank matrices into the attention layers while keeping the base model frozen. This parameter-efficient approach allowed us to leverage the extensive linguistic knowledge of models like Qwen2-7B and Gemma-9B, adapting them to interpret geometric embeddings without catastrophic forgetting.

The experimental evaluation on a stratified subset of 400 SVG-caption pairs yielded a nuanced result. Interestingly, the **Baseline Training Process (Text-only LoRA-LLM)** achieved the highest performance on the **raw CLIPScore (32.30)**, demonstrating that modern LLMs like Qwen2 can extract significant semantic signal directly from raw SVG code tokens. However, this visual alignment often came at the cost of linguistic coherence. The **SPE+Qwen2-7B** configuration, while scoring slightly lower on CLIP (29.30), achieved the highest **Composite Score (4.18)** and significantly outperformed the baselines in linguistic quality metrics (BLEU, ROUGE). This proves that the structured geometric input acts as a powerful regularizer, enabling the generation of fluent, human-readable captions that capture high-level semantics rather than just low-level primitives.

6.2 Main Contributions Revisited

Reconnecting to the research objectives stated in Chapter 1, I summarize the contributions of this thesis. My methodological contributions are threefold. First, I introduced (i) **a vector-native SVG captioning architecture**, the first system to caption SVGs directly from their vector definition without rasterization, combining structured geometric embeddings with LoRA-adapted LLMs. Second, I demonstrated (ii) **the efficacy of discrete geometric tokenization**, showing that treating quantified coordinates as discrete tokens is sufficient for LLMs to learn spatial reasoning, challenging the necessity of complex continuous encodings. Third, I demonstrated (iii) **parameter-efficient multimodal adaptation**, showing that LoRA extends effectively to the visual-geometric domain, preserving linguistic knowledge while learning the geometric-to-linguistic mapping.

6.2.1 Empirical Contributions

Empirically, I have contributed: (i) **a curated SVG captioning benchmark** of 90,000 stratified samples from Icons8, serving as a foundation for future research; (ii) **a comprehensive evaluation methodology**, defining a reproducible composite scoring function and documenting the full pipeline; and (iii) **ablation studies and an error taxonomy**, systematically analyzing design parameters and categorizing failure modes (hallucination, attribute mismatch, OCR failure) to guide future improvements.

6.2.2 Practical Contributions

Practically, I delivered: (i) **implementation artifacts**, including the documented software architecture and training pipeline, which I commit to releasing open-source; and (ii) **an accessibility pathway**, demonstrating a viable route toward automatic alt-text generation for SVG icons to address a critical need in web accessibility.

6.3 Empirical Findings and Insights

The quantitative and qualitative analyses in Chapter 5 revealed several key insights. First, (i) **Geometry vs. Semantics Trade-off**: The SPE excels at encoding explicit geometry but struggles with abstract semantics, suggesting a need for multimodal pretraining. Second, (ii) **The Critical Role of Feature Standardization**: Aligning the embedding statistics of the visual encoder with the LLM via Z-score standardization is essential for stability; without it, statistical mismatches cause divergence. Third, (iii) **Diminishing Returns**: Both LoRA rank and embedding dimension show little gain beyond moderate values ($r = 16$), validating the low-rank hypothesis. Finally, (iv) **Linguistic Quality vs. Visual Alignment**: Models optimized for linguistic overlap sometimes sacrifice visual grounding, indicating a need for multi-task objectives.

6.4 Limitations

Despite the promising results, several limitations constrain the current system.

6.4.1 Technical Limitations

Several technical limitations constrain the current system: (i) **Incomplete Style Encoding**, as the SPE under-represents color and texture, leading to hallucinations; (ii) **Flat Linearization**, which discards hierarchical group structure; (iii) **OCR Blindness**, where text rendered as paths is seen as generic curves; and (iv) **Computational Cost**, which still requires significant GPU resources despite the efficiency of LoRA.

6.4.2 Dataset and Domain Limitations

Data limitations are crucial to acknowledge:

- **Icon-Centric Training**: The model has been trained and evaluated primarily on icons. It remains untested on technical diagrams, UI mockups, or scientific charts, which possess different structural densities and semantic hierarchies.
- **Single Source Dataset**: Relying solely on the Icons8 dataset introduces a stylistic bias. The clean, flat design style of Icons8 may limit the model's ability to generalize to hand-drawn sketches, complex vector art, or legacy SVG files found in the wild.
- **English-Only Captions**: The current dataset is restricted to English captions. This limits the global accessibility of the tool and overlooks the challenges of multilingual grounding, which is essential for a universally applicable accessibility tool.

6.4.3 Evaluation Limitations

Evaluation methodologies also face constraints:

- **Reference-Based Metrics**: Metrics like BLEU and ROUGE rely on n-gram overlap with a single ground truth. They often penalize valid but distinct phrasing (*e.g.*, "a red circle" vs. "a circular red shape"), failing to capture the true semantic validity of a generated caption.
- **Limited Human Evaluation**: My evaluation relied primarily on automatic metrics. While cost-effective, these metrics cannot fully replace large-scale user studies with diverse human evaluators, particularly those from the target demographic (*e.g.*, visually impaired users).

6.5 Ethical Considerations and Responsible Deployment

Automated captioning for accessibility is a high-stakes application. Errors can have real-world consequences, and automated captioning carries risks: (i) **Misidentification Risk**, where incorrect labels could mislead users in safety-critical contexts; (ii) **Accessibility Equity**, as poor automated captions may degrade the experience for visually impaired users; (iii) **Bias and Representation**, since models may propagate stereotypes found in training data; and (iv) **Transparency**, necessitating clear indicators that captions are machine-generated.

6.6 Future Research Directions

This thesis opens numerous avenues for future research, which I organize into five categories.

6.6.1 Architectural Extensions

Hierarchical Spatial Encoders

The current SPE linearizes the SVG DOM tree, discarding group structure. Future work should explore **hierarchical encoders** that preserve parent-child relationships. For example, a group-aware attention mechanism could capture that a small circle "belongs to" a larger rectangle (*e.g.*, a button with an icon inside). Graph Neural Networks (GNNs) or Tree-Structured Transformers could model the SVG scene graph explicitly, enabling the model to reason over compositional structure more effectively.

Multi-Scale Feature Fusion

The SPE uses a single MLP to aggregate positional, type, and style embeddings. A multi-scale fusion architecture–similar to Feature Pyramid Networks in computer vision–could process coarse (global layout) and fine (local details) information in separate streams, then fuse them adaptively.

Hybrid Raster-Vector Encoders

While my system is vector-native, there are scenarios (*e.g.*, icons with embedded raster textures or images) where rasterization is unavoidable. A hybrid architec-

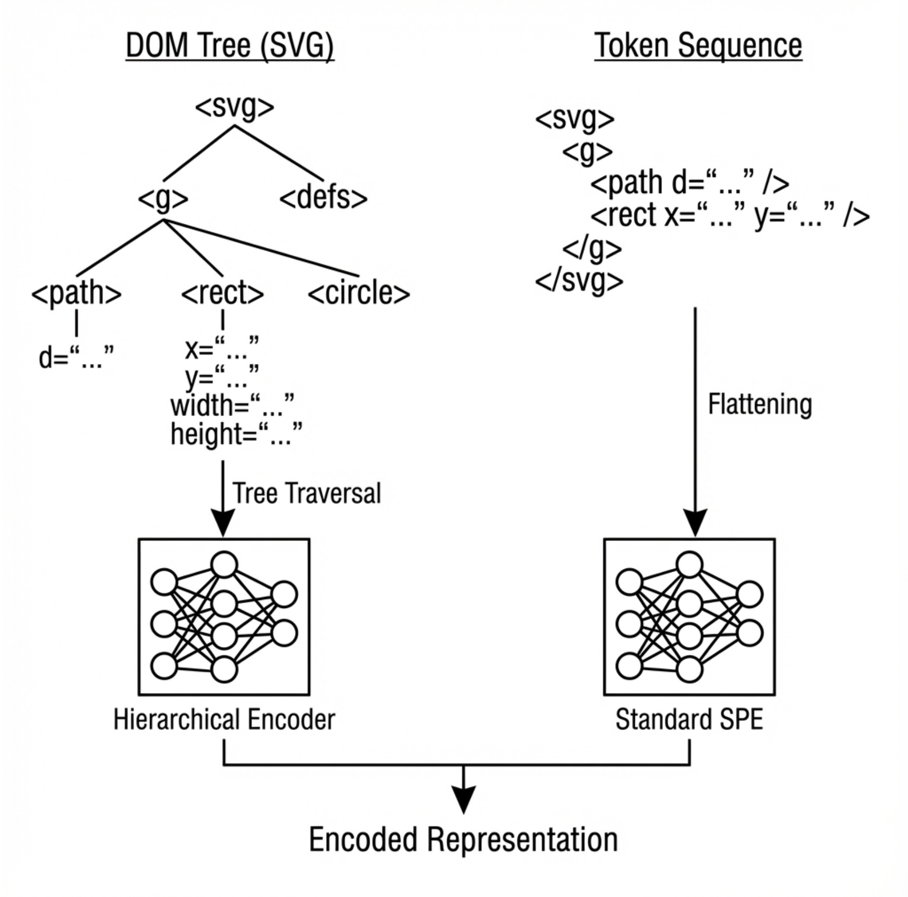


Figure 6.1: **Hierarchical vs. Flat Encoding**. A conceptual comparison between processing the SVG DOM as a hierarchical tree (left), allowing for structure-aware encoding, versus the current flat serialization approach (right).

ture that processes both the vector definition (via SPE) and a low-resolution raster rendering (via a vision encoder) could combine the strengths of both modalities.

OCR Integration

For SVG text rendered as paths, integrating an OCR module (*e.g.*, a character-level Transformer trained on glyph outlines) could enable the model to "read" text. This would significantly improve performance on logos, labels, and annotated diagrams.

6.6.2 Dataset and Domain Expansion

Technical Diagrams and Data Visualizations

Extending the approach to more complex SVG domains–technical schematics, flowcharts, data plots–would require larger, more diverse datasets. Collaborations with online repositories (*e.g.*, Wikimedia Commons, diagram libraries) could facilitate this.

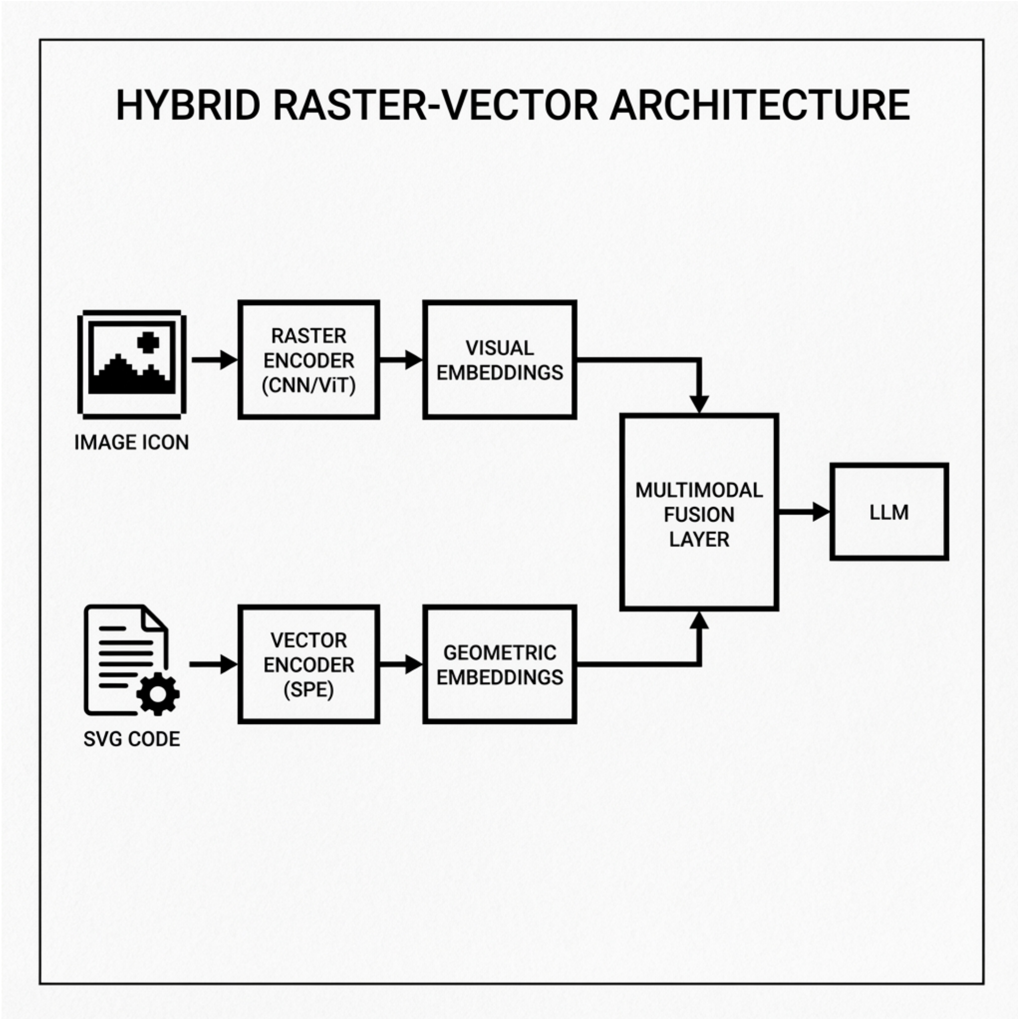


Figure 6.2: **Proposed Hybrid Architecture**. A dual-stream network that processes raster images (via CNN/ViT) and vector code (via SPE) in parallel, fusing the representations to leverage both visual texture and geometric precision.

Cross-Dataset Generalization

Evaluating on icons from multiple sources (Font Awesome, Material Icons, Feather Icons, custom designs) would assess generalization. Domain adaptation techniques (*e.g.*, fine-tuning on a small set of target-domain examples) could improve robustness.

Multilingual Captions

Training on multilingual SVG-caption pairs (*e.g.*, English, Italian, Chinese) would broaden accessibility. Multilingual LLMs like mBERT or XLM-R could serve as backbones.

Synthetic Data Generation

Generating synthetic SVG-caption pairs (*e.g.*, by programmatically creating shapes and describing them via templates, or by using text-to-SVG models like IconShop in reverse) could augment the dataset and improve coverage of rare geometric patterns.

6.6.3 Multimodal Pretraining

SVG-Image Contrastive Learning

Pretraining the SPE on pairs of SVG code and rasterized images (similar to CLIP's image-text pretraining) could ground geometric patterns to visual semantics. A contrastive loss would align the SPE's output with a frozen vision encoder's output, enabling zero-shot transfer.

Joint SVG-Text-Image Pretraining

A three-way contrastive objective (SVG code, rasterized image, caption) could learn a unified embedding space, enabling tasks like text-to-SVG generation, SVG-to-image retrieval, and SVG captioning within a single model.

Masked SVG Modeling

Inspired by masked language modeling (BERT), I could pretrain the SPE by masking random SVG elements and training the model to reconstruct them. This self-supervised task could teach the encoder robust geometric representations before fine-tuning on captioning.

6.6.4 Evaluation Methodology

Human Evaluation at Scale

Large-scale human evaluation studies (*e.g.*, via crowdsourcing) would assess caption quality, accessibility utility, and user satisfaction. Metrics like preference ranking (humans choose between captions from different models) and task success rate (can a user identify the correct icon given the caption?) would complement automatic metrics.

User Studies with Visually Impaired Participants

Collaborating with accessibility organizations to conduct user studies with visually impaired participants would provide critical feedback on real-world utility. Questions to explore: Do auto-generated captions improve navigation? Are error modes (color hallucinations, geometric mismatches) detrimental to understanding?

Benchmarking on Standardized Suites

Developing standardized benchmark suites for SVG understanding (analogous to MS-COCO for image captioning or SQuAD for question answering) would enable reproducible comparison across methods.

6.6.5 Broader Vision: End-to-End Vector-Language Systems

Ultimately, this thesis represents a step toward **end-to-end vector-language systems**. Future research could explore:

- **Bidirectional Systems**: Architectures capable of both captioning (SVG → Text) and generation (Text → SVG) within a single framework.
- **Vector-Aware Editing**: Enabling users to modify SVGs using natural language commands (*e.g.*, "make the circle red", "move the text to the center"), bridging the gap between design tools and language models.
- **Style Transfer**: Condition generation or editing on geometric style, allowing for the transfer of visual aesthetics between vector graphics.
- **Diagram Understanding**: Scaling the system to understand and reason over complex technical schematics, flowcharts, and engineering diagrams.

6.7 Closing Remarks

This thesis has demonstrated that Large Language Models, when equipped with specialized geometric encoders and parameter-efficient adaptation techniques, can learn to understand and describe vector graphics in their native symbolic form. The SVG Path Embedder provides a principled way to represent geometric data, enabling the model to perceive structure directly. Low-Rank Adaptation allows us to leverage the extensive linguistic knowledge of pre-trained LLMs without the prohibitive cost of full fine-tuning.

The results are encouraging: the SPE+Qwen2-7B system achieves the highest **Composite Score** on my SVG captioning benchmark, effectively balancing visual grounding with linguistic fluency. While the **Text-only Qwen2** model surprisingly excels at raw signal extraction (achieving the highest CLIPScore), it often fails to synthesize this information into coherent natural language. The SPE bridges this gap, producing captions that are both visually relevant and human-readable, outperforming the text-only baseline in overall quality. However, significant challenges remain—incomplete style encoding, limited domain generalization, OCR blindness—pointing to rich opportunities for future research. Beyond the technical contributions, this work raises broader questions about the nature of visual understanding. If vision-language models have traditionally operated on pixels, treating images as opaque arrays of color values, what does it mean to "see" structure directly? Vector graphics, with their explicit geometry and hierarchical organization, offer a world where visual content is inherently symbolic. This thesis suggests that by embracing this symbolic nature—rather than discarding it through rasterization—I can build more interpretable, precise, and data-efficient systems.

As vector graphics continue to proliferate in digital design, data visualization, and interactive media, the need for automatic understanding and annotation will only grow. I hope that this thesis serves as a foundation for future research in vector-native multimodal AI, and that the ideas, methods, and insights presented here inspire new approaches to bridging the gap between geometry and language. The journey from pixels to paths has just begun.

Acknowledgments

I would like to express my heartfelt gratitude to all those who have supported me throughout this academic journey.

First and foremost, I want to thank my family, who have always been my foundation and source of strength. Their unwavering support and encouragement have been invaluable throughout my studies.

I am deeply grateful to all my friends who have stood by me during these years of study. Special thanks go to Rocco and Alessio, with whom I have faced many exams and shared countless hours of preparation. Their friendship and collaboration have made this journey not only bearable but also enjoyable. I also want to thank Edoardo, a longtime friend who has always been supportive throughout this journey.

A very special acknowledgment goes to my girlfriend Joel, who has been by my side during the most stressful period of this journey. Her patience, understanding, and continuous support have been a source of comfort and motivation when I needed it most.

Finally, I want to express my gratitude to my university friends and classmates who have been an invaluable source of help and companionship during these years of study. Their support, shared experiences, and collaborative spirit have made this academic path more meaningful and enjoyable.

I would also like to sincerely thank my supervisor, Prof. Lorenzo Baraldi, and my co-supervisor, Dott. Leonardo Zini, for their invaluable guidance, expertise, and support throughout the development of this thesis.

This thesis would not have been possible without the collective support of all these wonderful people.

Bibliography

[1] Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*, 2023.

[2] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

[3] Satanjeev Banerjee and Alon Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72, 2005.

[4] 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. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[5] Alexandre Carlier, Martin Danelljan, Alexandre Alahi, and Radu Timofte. Deepsvg: A hierarchical generative network for vector graphics animation. *Advances in Neural Information Processing Systems*, 33:16346–16356, 2020.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[7] 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.

[8] Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale,

Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

[9] Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning. *arXiv preprint arXiv:2104.08718*, 2021.

[10] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[11] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. *International Conference on Machine Learning*, pages 2790–2799, 2019.

[12] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[13] Jiaxi Hu et al. Supersvg: Superpixel-based scalable vector graphics synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8891–8900, 2024.

[14] Ajay Jain et al. Vectorfusion: Text-to-svg by abstracting pixel-based diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1204–1213, 2023.

[15] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[16] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*, 2023.

[17] Tzu-Mao Li, Michal Lukac, Michaël Gharbi, and Jonathan Ragan-Kelley. Differentiable vector graphics rasterization for editing and generative modeling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1939–1950, 2020.

[18] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

[19] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. *Text summarization branches out*, pages 74–81, 2004.

[20] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[21] Xu Ma, Yuqian Zhou, Xingqian Xu, Bin Sun, Dimitar Filev, Nikita Orlov, Yanzhi Yuan, and Humphrey Shi. Towards layer-wise image vectorization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16314–16323, 2022.

[22] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[23] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. PMLR, 2013.

[24] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual representations from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021.

[25] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. Technical report, OpenAI, 2018.

[26] Camila Rodriguez, Tomaso Bianchi, Priya Shah, and Luca Ferri. Starvector: Multimodal svg generation with code-first decoders. *arXiv preprint arXiv:2311.04567*, 2023.

[27] Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

[28] Jiaxi Song et al. Clipvg: Text-guided vector graphics manipulation. *Computer Graphics Forum*, 42(7):e14912, 2023.

[29] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2024.

[30] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[32] W3C. Scalable vector graphics (svg) 1.1 (second edition). https://www.w3.org/TR/SVG11/, 2011. Accessed: 2024-01-15.

[33] Ronghang Wu et al. Iconshop: Text-based vector icon synthesis with autoregressive transformers. *arXiv preprint arXiv:2304.14400*, 2023.

[34] Bin Xiao, Haiping Wu, and Yue Wei. Florence-2: Advancing a unified representation for a variety of vision tasks. *arXiv preprint arXiv:2311.06242*, 2023.

[35] 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*, pages 7717–7727, 2024.

[36] Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32, 2019.

[37] Leonardo Zini, Elia Frigieri, Sebastiano Aloscari, and Lorenzo Baraldi. vHector and HeisenVec: Scalable vector graphics generation through large language models. *arXiv preprint*, 2025.

[38] Leonardo Zini, Elia Frigieri, and Lorenzo Baraldi. A scalable vector graphics path auto-encoder. *Under review*, 2025.