MUSE: A Training-free Multimodal Unified Semantic Embedder for Structure-Aware Retrieval of Scalable Vector Graphics and Images

Kyeongseon Kim
KAIST
ellakim@kaist.ac.kr

Baek Sung-Eun
POSTECH
seongeun@postech.ac.kr

Lee Jung-Mok
POSTECH
jungmok@postech.ac.kr

Tae-Hyun Oh
KAIST
thoh.kaist.ac.kr@gmail.com

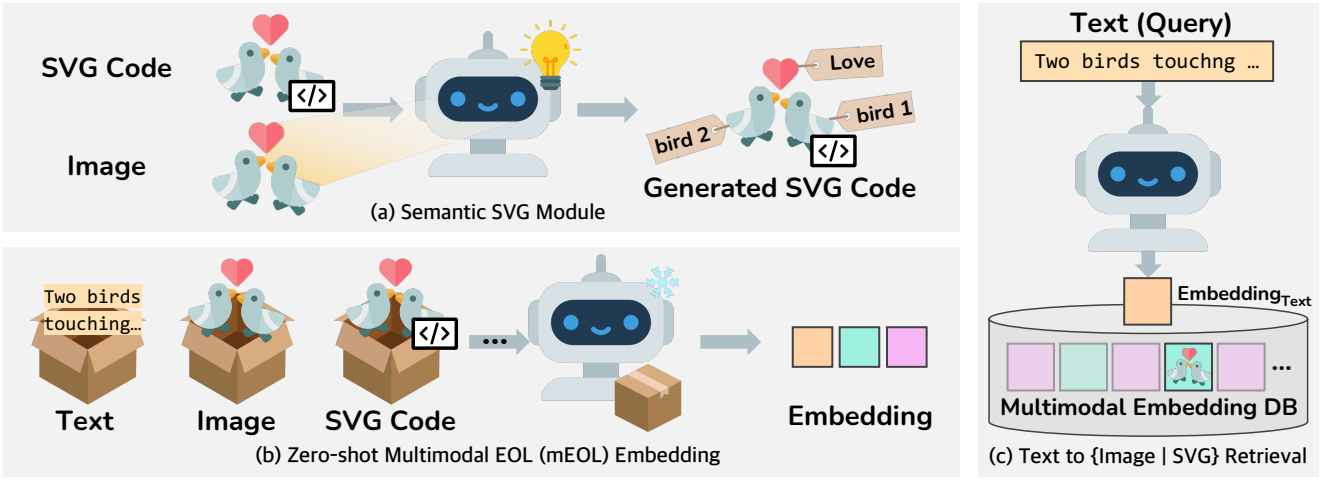


Figure 1. **Overview of Our Method.** We propose a training-free multimodal embedding method that aligns text, image, and SVG code. (a) To improve semantic grounding while using the structural information in SVG code, we apply a semantic SVG module that rewrites the code by replacing non-descriptive identifiers (*e.g.*, `"Layer_1"`) with semantically meaningful ones (*e.g.*, `"bird"`) via MLLM-guided visual reasoning. (b) Using the generated SVG from (a) and other modalities, our embedding strategy employs multimodal EOL (mEOL) prompting to project them into an aligned embedding space, without training. (c) We support cross-modal retrieval, where even long-form natural language queries retrieve semantically relevant SVG code and images with strong performance.

Abstract

While Scalable Vector Graphic (SVG) codes appear as either plain text or visually as images, they are structured representations that encode geometric and layout information. However, existing methods typically convert SVGs into raster image, discarding their structural details. Similarly, previous sentence embedding methods generate high-quality text embeddings but do not extend to structured or visual modalities such as SVGs. To address these challenges, we propose the first training-free multimodal embedding method that uses a Multimodal Large Language Model (MLLM) to project text, images, and SVG code into an aligned space. Our method consists of two main components: (1) multimodal Explicit One-word Limitation (mEOL), which produces compact, semantically grounded embeddings across modalities without training; and (2) a semantic SVG module that rewrites SVG code by generating missing or non-descriptive components through visual reasoning. This lets the model embed structural signals overlooked in prior work. Our approach not only introduces the first SVG retrieval setting but also achieves strong empirical performance, surpassing prior methods including training-based models by up to +20.5% Recall@1 on a repurposed VGBench dataset. These results demonstrate that structural cues can significantly enhance semantic alignment in multimodal embeddings, enabling

effective retrieval without any fine-tuning.

1. Introduction

Scalable Vector Graphics (SVGs) [11] are widely used in user interfaces [7] because they are compact, resolution-independent, and encode both visual content and structural information. They use geometric attributes to define shapes and optional identifiers to group elements or suggest semantic roles. For instance, a circle might be defined by `<circle cx="50" cy="50" r="40"/>` for geometry, while related elements can be grouped under an identifier `<g id="icon-group">`. These structural components encode more than just appearance. They can express relationships between parts, object categories, or semantic intent. For example, grouping elements under `<id="love_bird">` or `<id="wing">` introduces meaning that goes beyond raw geometry. However, many existing methods convert SVGs into raster images before analysis, and discard structural cues and treat SVGs as only visual inputs. As a result, downstream tasks such as retrieval, question answering, or semantic search suffer from degraded performance due to the loss of high-level structural and symbolic information [37].

To address these limitations, we propose a training-free multimodal embedding method that preserves and utilizes the structural information in SVGs. Our method embeds text, raster images, and SVG code into an aligned embedding space using a Multimodal Large Language Model (MLLM). The method consists of two components: First, we propose a training-free multimodal embedder using Explicit One-word Limitation (mEOL), which compresses diverse inputs into a single-word embedding. Second, we propose a semantic SVG module rewrites SVG code by generating meaningful identifiers and simplifying structure. These allows the model to leverage both appearance and symbolic structure for cross-modal retrieval, without any training or fine-tuning.

Our main contributions are summarized as follows:

- **Training-free Multimodal Embedder:** We propose a training-free multimodal embedder using an MLLM that supports retrieval across text, raster image, and SVG code modalities. Even when given long-form natural language queries, our method can retrieve semantically relevant results using both appearance and structure. To our knowledge, this is the first method to enables retrieval directly over raw SVG code without any training.
- **Semantic SVG Module:** We propose semantic SVG module that rewrites and simplifies SVG code. The module generates semantically meaningful identifiers to unlabeled elements and restructures complex code to improve embedding quality, while preserving the original rendering.

2. Related Work

2.1. Scalable Vector Graphics (SVGs)

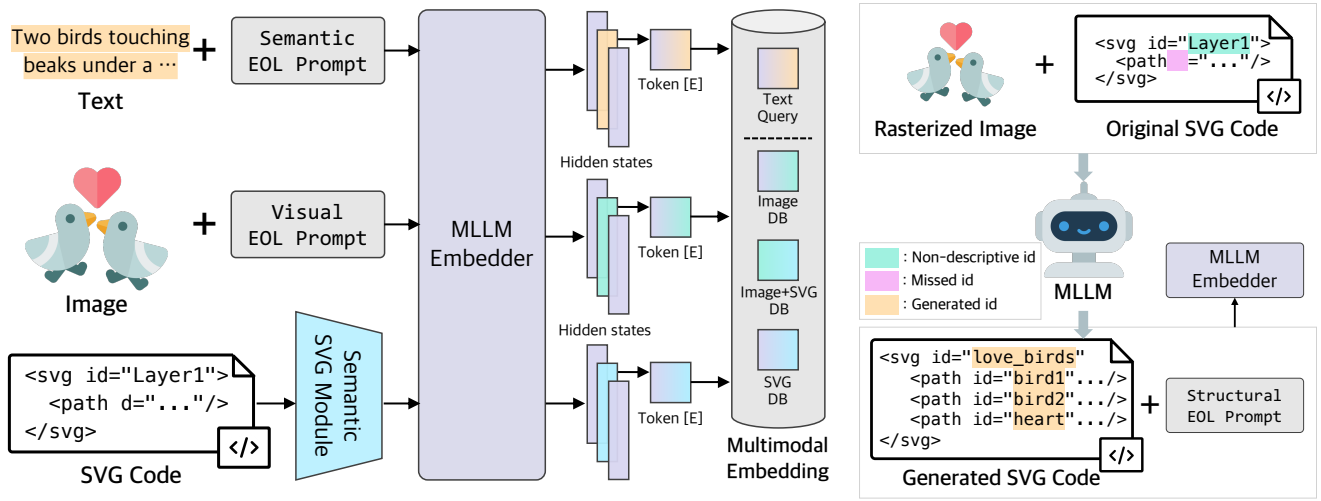
Scalable Vector Graphics (SVGs) encode visual information through geometric paths and structural attributes representing shapes and paths [5]. These elements not only make them well-suited for scalable applications [3, 14, 35, 39, 41] across various devices and resolutions, but also convey more than appearance. However, previous SVG studies such as generation [15, 33, 34, 38, 40], understanding [5, 8, 30, 44] and SVG-to-SVG retrieval [6] remain challenging because lengthy and complex structures are difficult for LLMs or MLLMs to understand without additional training [4, 37]. To address this challenge, recent studies have transformed complex SVGs into alternative representations to help models better grasp the content and context of SVG code [25, 37] and improve SVG-to-SVG retrieval performance [6]. However, such methods generally rely on additional model fine-tuning and these approaches do not support text-to-SVG retrieval. In other words, retrieval over structured SVGs from natural language remains largely unexplored, especially in training-free settings. Therefore, we propose the first training-free method designed specifically for text-to-SVG retrieval tasks, which addresses this gap by regenerating SVGs and injecting semantics to make them interpretable by MLLMs.

2.2. Sentence Embedding

Sentence embedding maps a sentence into a compact vector that captures its semantic meaning. Early approaches used pretrained encoders like BERT [17, 32], or contrastive learning methods [9, 12], which train the model to bring semantically similar sentences closer in embedding space. More recent methods leverage large language models (LLMs) for embedding [2, 21, 26, 28, 29]. These involve fine-tuning the model or training additional layers for specific downstream tasks. In contrast, some methods extract sentence embeddings without any parameter updates by guiding the LLM to compress a sentence into a single word. This strategy, known as Explicit One-word Limitation (EOL), prompts the model with a template such as `"This sentence: [text] means [MASK]"`. Variants such as PromptEOL [18], KEEOL [43], MetaEOL [22], and GenEOL [36] differ mainly in how they modify the prompt. These changes alone have shown strong performance improvements in training-free setups. However, existing EOL methods are limited to plain text. They do not generalize to non-text modalities such as images or structured SVG code. We are the first to extend EOL to multimodal inputs.

2.3. Multimodal Joint Embedding

Embedding is a fixed-size vector representation that contains the semantics of input data and can be used for diverse downstream tasks such as classification, clustering, reranking, and retrieval tasks [9, 12, 17, 27, 32]. Multimodal



(a) Illustration of our training-free multimodal EOL embedding.

(b) Pipeline of the semantic SVG module.

Figure 2. **Components of Our Training-free Multimodal Embedding pipeline.** (a) Given text, image, or SVG inputs, each input is paired with a multimodal EOL (mEOL): Semantic EOL Prompt for text and Visual EOL Prompt for images. Then, our training-free MLLM embedder generates an aligned embedding. The embedding is extracted from the hidden state of the final output token `[E]` at the penultimate layer. (b) Before SVG embedding, the Semantic SVG Module generates SVGs by completing missing or non-descriptive ids (*e.g.*, `id="Layer_1"` to `id="bird1"`) and simplifying the code structure through visual reasoning over the rendered image. This process improves the structural grounding of embeddings without altering visual appearance. The Generated SVG with the Structural EOL are then fed into the MLLM embedder to produce SVG or Image+SVG embeddings.

embedding methods aim to align representations across different modalities, such as text and image. Earlier models such as [23, 24, 31, 42] use modality-specific encoders to project modalities and train them with contrastive learning on large paired datasets. More recent approaches leverage MLLMs [19, 20], which enable to encode both images and text. Such methods often train the model with contrastive losses to effectively produce aligned embeddings. However, such training-based approaches can be costly, especially for specialized data formats (*e.g.*, SVG). Therefore, we propose a training-free multimodal embedding method for SVG using MLLMs. By extending EOL to non-text modalities, our approach enables aligned embeddings across text, raster images, and SVG code, without any fine-tuning.

3. Methodology

In this section, we propose a semantic SVG module (Sec. 3.2) and a zero-shot mutimodal Explicit One-word Limitation (mEOL) embedder (Sec. 3.3).

3.1. Pipeline Overview

Our pipeline consists of three main steps, as shown in Fig. 2. (1) Generating SVGs using Semantic SVG module, (2) Training-free multimodal embedding, and (3) Retrieval using cosine similarity. First, we complement the SVG code by replacing non-descriptive identifiers with meaningful labels (*e.g.*, `"Layer_1"` to `"bird"`) and simplifying the code using a MLLM, which enriches the semantic information without changing visual appearance. Next, the generated SVG codes, images, and textual inputs are fed into the MLLM with each prompt, generating embeddings represented by the hidden state of the last token that compactly represent both visual and semantic features. Finally, retrieval is performed by computing cosine similarity between query and database embeddings in an aligned representation space. Through these steps, our pipeline can address text-to-SVG retrieval effectively, bridging the gap between various modalities.

3.2. Semantic SVG Module

SVG code often includes attributes such as `id` and `class` that are useful for styling or scripting, but many SVGs contain non-descriptive or missing identifiers (*e.g.*, `id="Layer_1"`, `id="path123"`). In addition, SVGs frequently contain overly complex and redundant structures, making them difficult for language models to interpret. As a result, existing methods typically convert SVGs into raster images, discarding their symbolic structure and limiting the possibility. To address this, we propose a Semantic SVG Module that completes and leverages the structural information of SVG code. The module jointly analyzes the raw SVG code and its rendered image using MLLM, and generates a semantically enriched and compact version of the input.

The overall process of the semantic SVG module (Fig. 2b) consists of four components: (1) Input: The MLLM takes the original SVG code along with its rendered image. (2) Image & SVG Analysis: It analyzes both the SVG structure and its rendered image to identify visually salient objects (*e.g.*, `"bird"`) by using the joint context of code and image.

(3) Object Matching & Code Simplification: Detected visual objects are aligned with the corresponding SVG elements, and redundant or unnecessarily nested components are simplified based on geometric structure and relative position in the code and image. (4) ID assignment: Each matched SVG element is assigned a meaningful identifier based on the detected object, replacing non-descriptive labels (*e.g.*, `id"=Layer_1"`) with semantic ones (*e.g.*, `id="bird"`), or assigning new IDs to elements with missing identifiers.

Generated SVG encodes both symbolic and structural information that was previously ignored. This enables, for the first time, training-free retrieval from text to structured SVG code. Combined with the SVG EOL Prompt, this generated SVG code significantly improves embedding quality and retrieval performance.

3.3. Training-free Multimodal Embedder

In this section, we propose a novel training-free multimodal embedding method, extending Explicit One-word Limitation (EOL) [36, 43] to support diverse modalities, including text, raster images, and SVG code. We refer to this extension as *multimodal EOL (mEOL)*. Our method guides a MLLM to generate a compact semantic representation for each modality by mEOL to summarize the input as a single word, as shown in Fig. 2a. MLLMs operates autoregressively, generating one token at a time based on prior context. Using this property, we prompt the model with a concise instruction "$[X]$ `means in one word:`". Given a multimodal input $[X]$, the model generates a single token that summarizes the input's semantic meaning. Each modality's instruction encourages the model to focus on the most salient features of each modality and to extract a condensed semantic representation such as structural or visual cues as a single output token:

- **Text.** The instruction emphasizes both the semantic meaning of the description and its connection to visual attributes such as shape or layout.
- **Image.** The instruction guides the model to focus on visual features including color, structure, and object-level details.
- **SVG.** The instruction requests interpretation of the vector code in terms of its rendered meaning, directing attention to symbolic groupings and visual structure.
- **Image+SVG**. The model is instructed to jointly reason over rasterized appearance and SVG semantics, capturing both rendered style and underlying structure.

To generalize and support inputs from different modalities, our mEOL designs distinct modality-aware instructions tailored to each input type:

`This ⟨[X]⟩ ⟨instruction⟩ in one word:`

This format is consistently applied to text, images, and SVG code. The detailed instructions for each modality are provided in Supplementary Table 1. Conditioned on mEOL, we extract hidden states across the layers, denoted as $\mathcal{H} = [h^{(1)}, h^{(2)}, \ldots, h^{(L)}] \in \mathbf{R}^{L \times T}$, where L is number of layers and T is the whole token length. Among those hidden states, we select the hidden state of last token at the penultimate layer as the final embedding:

$$e_{mEOL} = h_{\mathrm{T}}^{(L-1)}$$

This representation compactly captures the semantic summary of the input, informed by MLLM's rich prior knowledge. Compared to the conventional feature extractors, our multimodal embedder first projects various modalities into the MLLM's aligned text space, and makes embeddings into one using MLLM's strong prior knowledge. This allows training-free retrieval across modalities, including structured SVG code and image which prior EOL methods could not support.

4. Results and Analysis

We first describe our experimental settings in Sec. 4.1. In Sec. 4.2, we demonstrate the effectiveness of our method by evaluating its performance and qualitative results in text-to-raster-image retrieval performance. In Sec. 4.3, we extend our method to additional modalities and show that our method implements the most effective EOL method compared to existing EOL approaches. Finally, we present appropriate settings for extracting embeddings and analyses through various ablation studies in Sec. 4.4.

4.1. Experimental Settings

Subset	Category	# of Samples	Avg. SVG Length	Components
SVG [37]	Usage	1,426	~2,400	Q, option (A-D), A

Table 2. **Statistics of the Repurposed VGBench Used in Our Experiments**. We adapt VGBench dataset for the text-to-SVG retrieval task by repurposing its annotations.

We evaluate our method using a repurposed version of the SVG dataset from VGBench [37], originally designed for vector graphic question answering (VGQA). The dataset contains thousands instances covering unique SVG images. Each instance includes an SVG code snippet, SVG code, a natural language question, four multiple-choice options (A–D), and a ground-truth answer label. Among those, we specifically use the subset of Usage category for SVG question answering. For example, a question such as *"What does this SVG image likely represent?"* may be accompanied by options like *[A: Global connectivity issues, B: Worldwide music distribution, C: Ecological awareness, D: Travel booking services]*, with the correct answer being *A*.

To reframe this into a text-to-visual retrieval task, we concatenate the question and correct answer A into a single natural language query (Q+A), evaluate whether the correct raster image can be retrieved from multimodal embedding database. The detailed statistics in Table 2. On average,

Model	Training	Query: Text / Database: Image					
		Recall@1	Recall@5	Recall@10	Recall@15	Recall@20	MRR
CLIP [31]	✗	0.1453	0.2442	0.2733	0.2878	0.2951	0.1854
BLIP [24]	✗	0.2078	0.3081	0.3387	0.3547	0.3648	0.2521
SigLIP [42]	✗	0.0465	0.1076	0.1483	0.1628	0.1788	0.0780
VLM2Vec [19]	✓	0.2137	0.3110	0.3314	0.3459	0.3532	0.2567
LLaVE [20]	✓	0.2326	0.3343	0.3561	0.3706	0.3735	0.2757
LLaMA-3.2-11B [13] (Ours)	✗	0.3387	0.5785	0.6526	0.6933	0.7224	0.4428
Qwen2.5-VL-7B [1] (Ours)	✗	**0.3503**	**0.6177**	**0.6846**	**0.7137**	**0.7456**	**0.4634**

Table 1. **Text-to-Raster Image Retrieval Performance on the Repurposed VGBench Dataset.** We compare our method with commonly used Vision-Language Models (VLMs) in text-to-image retrieval [24, 31, 42] and embedding methods utilizing Multimodal Large Language Model (MLLM) [19, 20]. Our method outperforms prior VLMs and MLLM-based approaches across all top-k recall metrics.

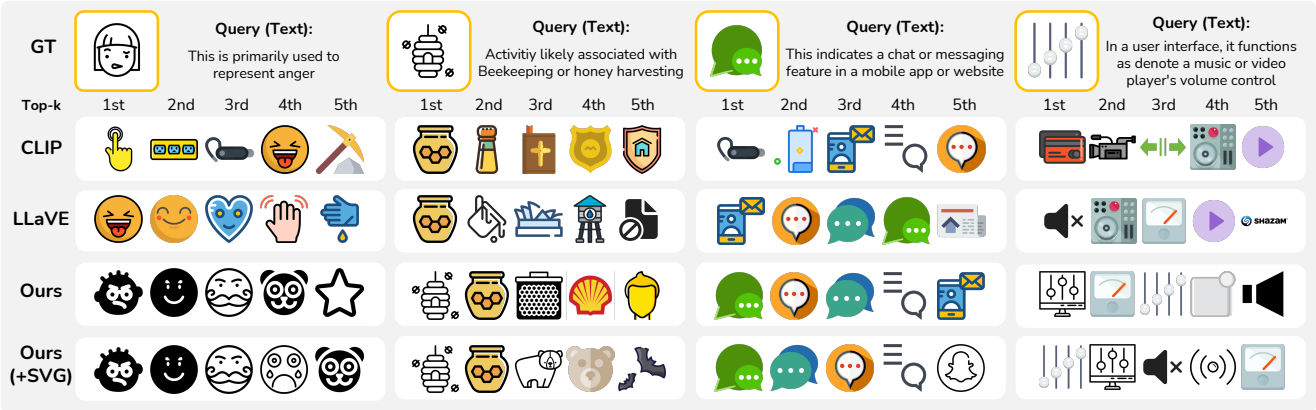


Figure 3. **Qualitative Examples of Text-to-Image, Text-to-Image with SVG Code Retrieval.** The icon with the yellow border represents the target image. Given a text query, each model retrieves images ranked from 1st to 5th based on cosine similarity. Compared with CLIP and LLaVE, our methods retrieved more semantically aligned with the query.

SVGs in VGBench contain approximately 2,400 characters of raw XML code, reflecting nontrivial structural complexity. As illustrated in Fig. 2a, we treat this textual input as the **query**, while the retrieval **database** is composed of all 1426 examples, represented in three formats: (1) rasterized image, (2) raw SVG code, and (3) a combination of both. This setup allows us to evaluate retrieval across different modalities such as text-to-image, text-to-SVG, and text-to-hybrid (image+SVG) without any training. The retrieval database (DB) consists of SVG codes and raster images used throughout the VGQA Usage subset. For each query, we compute its embedding using our method and perform cosine similarity retrieval against the DB. We report standard retrieval metrics, including Recall@1, Recall@5, and Recall@10, and Mean Reciprocal Rank (MRR) to assess how well the model retrieves the ground-truth icon based on its description. Recall@k measures whether the correct item appears in the top-k results, while MRR reflects the inverse rank of the first correct retrieval result, and provides a measure of how early the relevant item appears in the ranked list.

For comparative baselines, we have selected the encoder-based methods [24, 31, 42] and MLLM-based encoding methods [19, 20]. All models are evaluated under the same retrieval setup using cosine similarity between the extracted text and raster image embeddings.

For our training-free multimodal embedding method, we have used two MLLMs: LLaMA-3.2-11B and Qwen2.5-VL-7B. We used the last token hidden states of the penultimate layer as embeddings. We provide empirical evidence supporting our embedding design choices in Sec. 4.4, focusing on embedding method and layer selection.

4.2. Main Results

Quantitative Results. In Table 1, we report the retrieval performance of the SVG dataset, given the concatenated text query (Q+A), and raster image as database. It shows that our method consistently outperforms encoder-based approaches such as CLIP [31], BLIP [24], and SigLIP [42], which rely on contrastive learning. Notably, our training-free approach exhibits strong generalizability across various Vision-Language Models (VLMs), achieving better performance than encoder-based methods without the need for additional training. While SigLIP suffers from limitations due to its maximum token length, potentially degrading its performance on longer queries, our method effectively handles lengthy and complex inputs by summarizing them into a com-

Model	EOL	Database format (Query: Text)								
		SVG			Image			Image + SVG		
		Recall@1	Recall@10	Recall@20	Recall@1	Recall@10	Recall@20	Recall@1	Recall@10	Recall@20
Mistral-7B	PromptEOL	0.0087	0.0203	0.0392	-	-	-	-	-	-
	KEEOL	0.0087	0.0247	0.0451	-	-	-	-	-	-
	Ours	**0.1439**	**0.2922**	**0.3314**	-	-	-	-	-	-
Qwen2.5-VL-7B	PromptEOL	0.0131	0.0218	0.0422	0.3183	0.6483	0.6846	0.3154	0.6628	0.7137
	KEEOL	0.0116	0.0218	0.0422	0.2703	0.5494	0.6017	0.2500	0.5567	0.6177
	Ours	**0.1744**	**0.2994**	**0.3314**	**0.3503**	**0.6846**	**0.7456**	**0.3765**	**0.7224**	**0.7776**
LLaMA-3.2-11B	PromptEOL	0.0102	0.0320	0.0177	**0.3387**	**0.6555**	0.7151	**0.3270**	0.6483	0.7093
	KEEOL	0.0087	0.0276	0.0161	0.2762	0.5509	0.6308	0.2573	0.5203	0.6134
	Ours	**0.1599**	**0.3038**	**0.3474**	**0.3387**	0.6526	**0.7224**	0.3169	**0.6628**	**0.7427**

Table 3. **Comparison of EOL Family and Baselines across Different Models and Database Formats.** Our multimodal EOL with semantic SVG module significantly improves retrieval accuracy. Note that Mistral-7B does not support image modality; therefore, related cells are left blank.

pact word embedding. Furthermore, in evaluations against VLM2Vec [19] and LLaVE [20], our approach achieves superior performance despite not being explicitly trained for embedding extraction like such methods, highlighting its efficacy and robustness.

Qualitative Results. Figure 3 represents qualitative results of the top-5 retrieved SVG icons given a text query. We compare an encoder-based method, *i.e*., CLIP, and a MLLM-based method, *i.e*., LLaVE, along with our proposed method. For visualization clarity, we simplify the textual queries in the figure. As illustrated in the first column, our method is the only one that successfully captures the intended concept of "anger", retrieving a semantically meaningful icon. In contrast, LLaVE retrieves generic face-like icons, and CLIP fails to retrieve a relevant icon, retrieving only a loosely related result at the 4th position. In the second column, our method demonstrates that the semantic SVG module contributes to capturing not only the structural attributes of the object but also its textual semantics. While CLIP tends to retrieve icons based solely on dominant visual features (*e.g*., yellow-colored icons), and LLaVE retrieves less relevant results beyond the top-2, our approach consistently retrieves semantically and visually aligned icons (1st–4th results). Moreover, our method successfully retrieves icons that reflect deeper semantic associations. For instance, the presence of bear icons in response to the "honey" query shows that our model understands the contextual relationship, rather than relying solely on visual resemblance. Also across multiple examples, our method consistently preserves both the structural attributes and the semantic meaning of the target concepts, resulting in qualitatively superior retrieval outcomes.

4.3. Comparisons with EOL Methods

In this section, we compare our embedding method with previous EOL-based methods [18, 43]. For PromptEOL and KEEOL, we differed the prompt setting following each method. Thus, we extract embeddings from each model using the prompt templates proposed by each method. We extract text embeddings (query) and raster image embeddings (database) from the last token's hidden state of the penultimate layer. SVG data is represented in text form, so we treat it similarly to regular text embeddings, enabling text-to-raster image retrieval using LLMs. Our method includes the semantic SVG module step using MLLM, while traditional EOL methods do not. For the model's generalizability, we have tested each EOL method on LLM [1, 13, 16].

Text-to-SVG Code Retrieval. For text-to-SVG code retrieval, we conducted experiments on both LLM (Mistral) and MLLMs (Qwen2.5-VL-7B, LLaMA-3.2-11B). In Table 3 (SVG), traditional EOL methods struggle to retrieve appropriate SVG code matching text queries. However, our method, which preprocesses SVG data, significantly outperforms other methods. This result implies that semantic SVG module helps LLMs and MLLMs better understand SVG data, demonstrating that our method effectively improves text-to-SVG code retrieval.

Text-to-Raster Image Retrieval. As shown in Table 3 (Image), our method retrieves raster images more effectively compared to others. Interestingly, KEEOL performs worse than PromptEOL, suggesting that the longer query text in KEEOL might weaken its embedding representation, making it less suitable for raster image retrieval tasks.

Text-to-Raster Image Retrieval with generated SVG Code. Table 3 shows that combining raster images and generated SVG code embeddings helps to retrieve images more accurately. Through MLLM's aligned space, we can generate a combined embedding by simply feeding both raster image and SVG code to MLLM. Compared to PromptEOL and KEEOL, our method uses concise yet informative SVG data alongside raster image embeddings, capturing both visual and structural elements. Thus, our method generates embeddings that are better suited for accurate retrieval tasks.

Model	Database Format	Recall@1	Recall@10	Recall@20
LLaMA-3.2-11B	Image	**0.3387**	0.6526	0.7224
	Image + SVG	0.3110	0.6570	0.7209
	Image + $\text{SVG}_{\text{generated}}$	0.3169	**0.6628**	**0.7427**
Qwen2.5-VL-7B	Image	0.3503	0.6846	0.7456
	Image + SVG	0.3605	0.7122	0.7558
	Image + $\text{SVG}_{\text{generated}}$	**0.3765**	**0.7224**	**0.7776**

Table 4. **Retrieval Performance depending on the Representation used for SVGs.** We measured the retrieval performance for various database formats. In the database format, *Image* refers to the case where only raster images are used. *Image + SVG* refers to the case where raster images are used alongside the full SVG code, which does not incorporate our method. *Image* + $SVG_{generated}$ refers to the case where raster images are used together with the generated SVG code, which applies our method. The experimental results show that the retrieval performance improves when the generated SVG code, which incorporates our method, is used alongside the raster image.

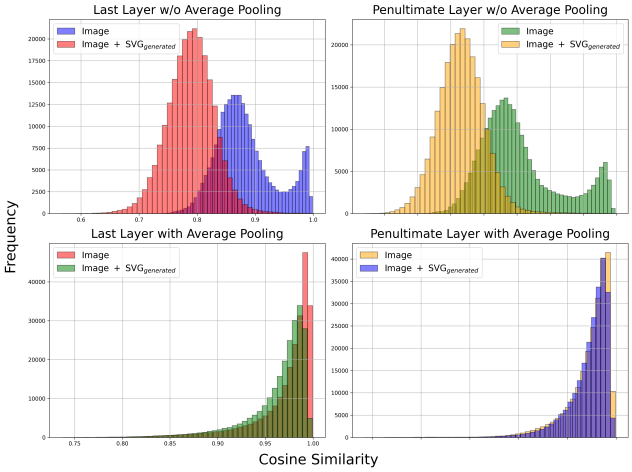


Figure 4. **Self-Cosine Similarity Distributions of Raster Image Embeddings and Embeddings Combining Raster Images with generated SVG Code in Qwen2.5-VL-7B.** We find that the embeddings combining raster images and generated SVG code tended to have lower cosine similarity scores in all settings. This suggests that adding generated SVG code can lead to more distinct and useful embeddings.

Method	Recall@1	Recall@5	Recall@10	Recall@20
One word (Ours)	**0.3169**	**0.5596**	**0.6628**	**0.7427**
Two words	0.2340	0.4491	0.5262	0.6177
Three words	0.2384	0.4273	0.4971	0.5988
Four words	0.2253	0.4273	0.5160	0.5828
Sentence	0.1105	0.2137	0.2674	0.3401

Table 5. **Ablation of mEOL prompt formulation.** Our mEOL prompt design represents information compactly as an one word. When prompt design request loger compressed forms, retrieval performance tends to decrease as length increases.

Model	Penultimate Layer	Last token	Recall@1	Recall@5
LLaMA-3.2-11B	✗	✗	0.0262	0.0610
	✓	✗	0.0044	0.0145
	✗	✓	0.3212	0.5698
	✓	✓	**0.3387**	**0.5785**
Qwen2.5-VL-7B	✗	✗	0.0073	0.0203
	✓	✗	0.0015	0.0087
	✗	✓	0.3387	0.5930
	✓	✓	**0.3503**	**0.6177**

Table 6. **Ablation on Raster Image Embedding Layer and Pooling Method.** Using the last token from the penultimate layer without pooling achieves the best performance. The last token indicates that average pooling has not been applied.

4.4. Ablation Studies

Ablation of Semantic SVG Module. Table 4 shows the comparison result of using the raster image-only embedding for SVG representation and the mixture of raster image and SVG code embedding for SVG representation. Using the naive mixture embedding (Image+SVG) reduces the performance compared to raster image-only embedding, but increases and surpasses raster image-only embedding performance when using our mixture embedding with semantically generated SVG code and the raster image. This implies that simply feeding heterogeneous representations may introduce noise and degrade retrieval quality. However, when the generated SVG code is first enriched to capture high-level semantics, the resulting embedding provides complementary information to the raster image. This enables the model to better align visual and textual semantics, leading to improved retrieval performance. Figure 4 shows that raster image embeddings with generated SVG code are distributed at significantly lower cosine similarity scores compared to raster image embeddings without SVG code. This result indicates that our method effectively captures distinctive semantic features, thereby creating a more clearly differentiated embedding space. Consequently, our approach effectively integrates structural cues from the SVG code and visual features from the raster image, forming a more discriminative representation for multimodal retrieval tasks.

Ablation of mEOL Prompt Formulation. We use a prompt design that instructs the model to compactly represent the key information of the input. While our method compresses core information into a single word, Table 5 presents retrieval performance changes when this compressed representation is extended to two words or sentences. The results indicate that retrieval performance tends to decrease as the compressed information becomes longer. We hypothesize that this occurs due to the critical information being distributed across multiple tokens in longer compressed forms, making them less effective for retrieval tasks.

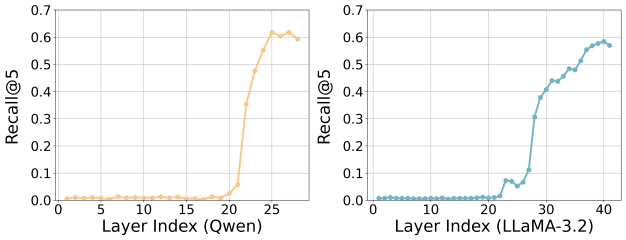


Figure 5. **Ablation on Layer Index for Raster Image Embedding Extraction.** We evaluate retrieval performance across layers when selecting the last token embedding. Performance generally improves in the later layers, with the penultimate layer consistently offering the best trade-off between stability and retrieval quality.

Ablation of Token Embedding for Raster Images. Traditional EOL embedding methods [18, 22, 36, 43] typically uses the method to extract the representation from either the averaging all input token embeddings or the last token of the last hidden layer. As we extend EOL methods to MLLMs, we need to verify whether the tendencies of MLLMs regarding these options are similar to those observed in traditional EOL methods. Table 6 represents the result of the embedding methods and the use of the penultimate layer. As shown, using the penultimate layer with the simple last token shows the best results, followed by the use of the last layer with the last token.

Ablation of Layer Index. Specifically, we examine which hidden layers of the MLLM are most suitable for obtaining useful embeddings. Figure 5 indicates that recall@5 sharply increase in the later layers for both models. This implies that embeddings from later layers are more informative compared to the early layers. Therefore, following traditional EOL methods, we compare embeddings from the last layer and the penultimate layer. Figure 6 (top) shows the self-similarity distribution of raster image embeddings extracted for the penultimate layer and last layer. The left plot shows the distribution when using the last token embedding, and the right plot shows the distribution with average pooling over all tokens. In both cases, embeddings from the last layer exhibit a right-shifted distribution compared to those from the penultimate layer, indicating lower variance and expressiveness of the embeddings, similar results are shown at [10]. Thus, using the penultimate layer can improve embedding expressiveness.

Ablation of Embedding Method. Next, we test the effect of two embedding methods: using the last token, *i.e.*, newly generated one, or average pooling the whole tokens' hidden vectors. Figure 6 (bottom) demonstrates that when average pooling is used, the self-similarity distribution becomes concentrated near 1. This means pooling the whole hidden states makes embeddings overly similar, reducing their distinct

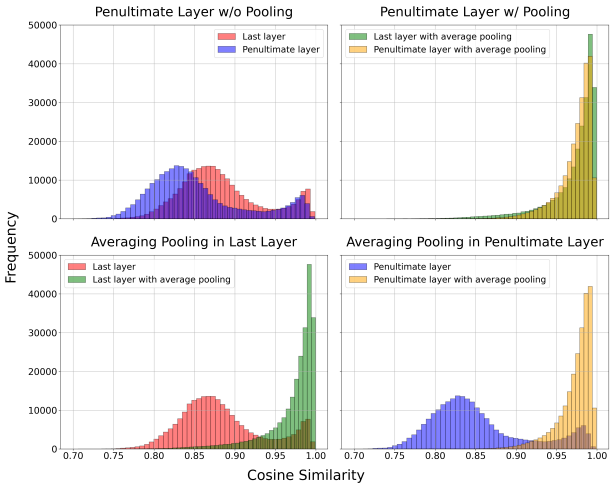


Figure 6. **Self-Cosine Similarity Distributions of Raster Image Embeddings in Qwen2.5-VL-7B before and after with and without Penultimate Layer or Pooling. (top)** The visualization of the self-similarity distribution based on the layer index used for extracting embeddings. We observe that the cosine similarity distribution is located at relatively lower values when using the penultimate layer compared to the last layer. **(bottom)** The visualization of the self-similarity distribution based on the method used to extract embeddings. We find that when average pooling is not applied, the cosine similarity distribution is concentrated at much lower values. Therefore, using the penultimate layer without pooling exhibits the best separation in the embedding space.

semantic information for retrieval tasks.

In the context of SVGs, where each sample may contain many components with diverse structural formats and numerical values, simple averaging over all token embeddings may fail to capture the proper semantic or textual information of the input. But the embedding derived from the last token, which implies the MLLM's interpretation of the input tokens, can be more effective in preserving semantic or textual information. Therefore, we conclude that using the last token embedding is better suited for our framework, which handles the structured and variable nature of SVG inputs.

5. Conclusion

In this work, we introduce a training-free multimodal embedding method that enables text, image, and SVG code inputs to be embedded into a aligned representation space. By using multimodal EOL and the semantic SVG module, our approach significantly improves retrieval accuracy without requiring any additional tuning. The proposed method uses MLLM to encode text, image, and SVG code into an aligned embedding space, enabling the efficient use of the elements inherent in each modality, both for general data and in the case of the special format, *e.g.*, SVG. This implies the potential for embedding approaches using language

models, such as EOL, to be extended to other modalities. Our approach enables efficient and accurate retrieval directly from text and offers a foundation for future applications that require semantic understanding of vector graphics.

References

[1] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025. 5, 6

[2] Parishad BehnamGhader, Vaibhav Adlakha, Marius Mosbach, Dzmitry Bahdanau, Nicolas Chapados, and Siva Reddy. Llm2vec: Large language models are secretly powerful text encoders. *arXiv preprint arXiv:2404.05961*, 2024. 2

[3] Qi Bing, Chaoyi Zhang, and Weidong Cai. Deepicon: A hierarchical network for layer-wise icon vectorization. In *2024 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, pages 70–77. IEEE, 2024. 2

[4] Mu Cai, Zeyi Huang, Yuheng Li, Utkarsh Ojha, Haohan Wang, and Yong Jae Lee. Leveraging large language models for scalable vector graphics-driven image understanding. *arXiv preprint arXiv:2306.06094*, 2023. 2

[5] Mu Cai, Zeyi Huang, Yuheng Li, Utkarsh Ojha, Haohan Wang, and Yong Jae Lee. An investigation on llms' visual understanding ability using svg for image-text bridging. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 5377–5386. IEEE, 2025. 2

[6] Defu Cao, Zhaowen Wang, Jose Echevarria, and Yan Liu. Svgformer: Representation learning for continuous vector graphics using transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10093–10102, 2023. 2

[7] Alexandre Carlier, Martin Danelljan, Alexandre Alahi, and Radu Timofte. Deepsvg: A hierarchical generative network for vector graphics animation. *Advances in Neural Information Processing Systems*, 33:16351–16361, 2020. 2

[8] Siqi Chen, Xinyu Dong, Haolei Xu, Xingyu Wu, Fei Tang, Hang Zhang, Yuchen Yan, Linjuan Wu, Wenqi Zhang, Guiyang Hou, et al. Svgenius: Benchmarking llms in svg understanding, editing and generation. *arXiv preprint arXiv:2506.03139*, 2025. 2

[9] Yung-Sung Chuang, Rumen Dangovski, Hongyin Luo, Yang Zhang, Shiyu Chang, Marin Soljačić, Shang-Wen Li, Wen-tau Yih, Yoon Kim, and James Glass. Diffcse: Difference-based contrastive learning for sentence embeddings. *arXiv preprint arXiv:2204.10298*, 2022. 2

[10] Kawin Ethayarajh. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings. *arXiv preprint arXiv:1909.00512*, 2019. 8

[11] Jon Ferraiolo, Fujisawa Jun, and Dean Jackson. *Scalable vector graphics (SVG) 1.0 specification*. iuniverse Bloomington, 2000. 2

[12] Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*, 2021. 2

[13] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. 5, 6

[14] Juncheng Hu, Ximing Xing, Jing Zhang, and Qian Yu. Vectorpainter: Advanced stylized vector graphics synthesis using stroke-style priors. *arXiv preprint arXiv:2405.02962*, 2024. 2

[15] Ajay Jain, Amber Xie, and Pieter Abbeel. Vectorfusion: Text-to-svg by abstracting pixel-based diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1911–1920, 2023. 2

[16] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023. 6

[17] Ting Jiang, Jian Jiao, Shaohan Huang, Zihan Zhang, Deqing Wang, Fuzhen Zhuang, Furu Wei, Haizhen Huang, Denvy Deng, and Qi Zhang. Promptbert: Improving bert sentence embeddings with prompts. *arXiv preprint arXiv:2201.04337*, 2022. 2

[18] Ting Jiang, Shaohan Huang, Zhongzhi Luan, Deqing Wang, and Fuzhen Zhuang. Scaling sentence embeddings with large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 3182–3196, 2024. 2, 6, 8

[19] Ziyan Jiang, Rui Meng, Xinyi Yang, Semih Yavuz, Yingbo Zhou, and Wenhu Chen. Vlm2vec: Training vision-language models for massive multimodal embedding tasks. *arXiv preprint arXiv:2410.05160*, 2024. 3, 5, 6

[20] Zhibin Lan, Liqiang Niu, Fandong Meng, Jie Zhou, and Jinsong Su. Llave: Large language and vision embedding models with hardness-weighted contrastive learning. *arXiv preprint arXiv:2503.04812*, 2025. 3, 5, 6

[21] Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Nv-embed: Improved techniques for training llms as generalist embedding models. *arXiv preprint arXiv:2405.17428*, 2024. 2

[22] Yibin Lei, Di Wu, Tianyi Zhou, Tao Shen, Yu Cao, Chongyang Tao, and Andrew Yates. Meta-task prompting elicits embeddings from large language models. *arXiv preprint arXiv:2402.18458*, 2024. 2, 8

[23] Junnan Li, Ramprasaath Selvaraju, Akhilesh Gotmare, Shafiq Joty, Caiming Xiong, and Steven Chu Hong Hoi. Align before fuse: Vision and language representation learning with momentum distillation. In *Advances in Neural Information Processing Systems*, pages 9694–9705. Curran Associates, Inc., 2021. 3

[24] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International conference on machine learning*, pages 12888–12900. PMLR, 2022. 3, 5

[25] Raphael Gontijo Lopes, David Ha, Douglas Eck, and Jonathon Shlens. A learned representation for scalable vector graphics.

In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7930–7939, 2019. 2

[26] Rui Meng, Ye Liu, Shafiq Rayhan Joty, Caiming Xiong, Yingbo Zhou, and Semih Yavuz. Sfrembedding-mistral: enhance text retrieval with transfer learning. *Salesforce AI Research Blog*, 3:6, 2024. 2

[27] Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*, 2022. 2

[28] Niklas Muennighoff, SU Hongjin, Liang Wang, Nan Yang, Furu Wei, Tao Yu, Amanpreet Singh, and Douwe Kiela. Generative representational instruction tuning. In *ICLR 2024 Workshop: How Far Are We From AGI*, 2024. 2

[29] Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith B Hall, Daniel Cer, and Yinfei Yang. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. *arXiv preprint arXiv:2108.08877*, 2021. 2

[30] Zeju Qiu, Weiyang Liu, Haiwen Feng, Zhen Liu, Tim Z Xiao, Katherine M Collins, Joshua B Tenenbaum, Adrian Weller, Michael J Black, and Bernhard Schölkopf. Can large language models understand symbolic graphics programs?(2024). *URL https://arxiv. org/abs/2408.08313*, 57. 2

[31] 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 models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021. 3, 5

[32] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019. 2

[33] Juan A Rodriguez, Shubham Agarwal, Issam H Laradji, Pau Rodriguez, David Vazquez, Christopher Pal, and Marco Pedersoli. Starvector: Generating scalable vector graphics code from images. *CVPR*, 2025. 2

[34] Yiren Song, Danze Chen, and Mike Zheng Shou. Layertracer: Cognitive-aligned layered svg synthesis via diffusion transformer. *arXiv preprint arXiv:2502.01105*, 2025. 2

[35] Zecheng Tang, Chenfei Wu, Zekai Zhang, Mingheng Ni, Shengming Yin, Yu Liu, Zhengyuan Yang, Lijuan Wang, Zicheng Liu, Juntao Li, et al. Strokenuwa: Tokenizing strokes for vector graphic synthesis. *arXiv preprint arXiv:2401.17093*, 2024. 2

[36] Raghuveer Thirukovalluru and Bhuwan Dhingra. Geneol: Harnessing the generative power of llms for training-free sentence embeddings. *arXiv preprint arXiv:2410.14635*, 2024. 2, 4, 8

[37] Zhenhailong Wang, Joy Hsu, Xingyao Wang, Kuan-Hao Huang, Manling Li, Jiajun Wu, and Heng Ji. Visually descriptive language model for vector graphics reasoning. *arXiv preprint arXiv:2404.06479*, 2024. 2, 4

[38] Ronghuan Wu, Wanchao Su, and Jing Liao. Chat2svg: Vector graphics generation with large language models and image diffusion models. *CVPR*, 2024. 2

[39] Ximing Xing, Juncheng Hu, Jing Zhang, Dong Xu, and Qian Yu. Svgfusion: Scalable text-to-svg generation via vector space diffusion. *arXiv preprint arXiv:2412.10437*, 2024. 2

[40] Ximing Xing, Haitao Zhou, Chuang Wang, Jing Zhang, Dong Xu, and Qian Yu. Svgdreamer: Text guided svg generation with diffusion model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4546–4555, 2024. 2

[41] Yiying Yang, Wei Cheng, Sijin Chen, Xianfang Zeng, Jiaxu Zhang, Liao Wang, Gang Yu, Xingjun Ma, and Yu-Gang Jiang. Omnisvg: A unified scalable vector graphics generation model. *arXiv preprint arXiv:2504.06263*, 2025. 2

[42] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 11975–11986, 2023. 3, 5

[43] Bowen Zhang, Kehua Chang, and Chunping Li. Simple techniques for enhancing sentence embeddings in generative language models. In *International Conference on Intelligent Computing*, pages 52–64. Springer, 2024. 2, 4, 6, 8

[44] Bocheng Zou, Mu Cai, Jianrui Zhang, and Yong Jae Lee. Vgbench: Evaluating large language models on vector graphics understanding and generation. *CoRR*, 2024. 2