

# Style Customization of Text-to-Vector Generation with Image Diffusion Priors

Peiying Zhang  
City University of Hong Kong  
Hong Kong, China  
zhangpeiying17@gmail.com

Nanxuan Zhao  
Adobe Research  
San Jose, USA  
nanxuanzhao@gmail.com

Jing Liao\*  
City University of Hong Kong  
Hong Kong, China  
jingliao@cityu.edu.hk



Figure 1: Examples of vector graphics generated from text prompts in custom styles using our method, showcasing structural regularity and expressive diversity. Exemplar SVGs: the 1<sup>st</sup> and 3<sup>rd</sup> rows are from ©SVGRepo; the 2<sup>nd</sup> row is from ©iconfont.

## ABSTRACT

Scalable Vector Graphics (SVGs) are highly favored by designers due to their resolution independence and well-organized layer structure. Although existing text-to-vector (T2V) generation methods can create SVGs from text prompts, they often overlook an important need in practical applications: style customization, which is vital for producing a collection of vector graphics with consistent visual appearance and coherent aesthetics.

Extending existing T2V methods for style customization poses certain challenges. Optimization-based T2V models can utilize the priors of text-to-image (T2I) models for customization, but struggle with maintaining structural regularity. On the other hand, feed-forward T2V models can ensure structural regularity, yet they encounter difficulties in disentangling content and style due to limited SVG training data.

To address these challenges, we propose a novel two-stage style customization pipeline for SVG generation, making use of the advantages of both feed-forward T2V models and T2I image priors. In the first stage, we train a T2V diffusion model with a path-level representation to ensure the structural regularity of SVGs while preserving diverse expressive capabilities. In the second stage, we customize the T2V diffusion model to different styles by distilling customized T2I models. By integrating these techniques, our pipeline can generate high-quality and diverse SVGs in custom

styles based on text prompts in an efficient feed-forward manner. The effectiveness of our method has been validated through extensive experiments. The project page is <https://customsvg.github.io>.

## KEYWORDS

Vector Graphics, SVG, Diffusion Model, Style Customization, Text-Guided Generation

## 1 INTRODUCTION

Vector graphics, especially in the form of Scalable Vector Graphics (SVG), play an essential role in digital arts such as icons, clipart, and graphic design. By representing visual elements as geometric shapes, SVGs provide resolution independence, compact file sizes, and flexibility for layer-wise manipulation, making them highly favored by designers. Given the challenges of creating high-quality vector graphics, many recent works [Jain et al. 2022; Thamizhasan et al. 2024; Wu et al. 2023; Xing et al. 2024] have proposed algorithms in text-to-vector (T2V) generation. However, these methods overlook an important need in practical applications - style customization. Designers often customize a set of vector graphics with consistent visual appearance and aesthetic coherence. This is crucial for ensuring design quality, particularly in contexts like branding, user interfaces, and themed illustrations.

Simply extending existing T2V methods for style customization is difficult. Current T2V methods can be categorized into

\*Corresponding author

optimization-based and feed-forward methods. Optimization-based T2V methods, which either optimize a set of vector elements (e.g., cubic Bézier curves) to fit the images generated by T2I models [Ma et al. 2022; Zhang et al. 2023b], or directly optimize shape parameters using Score Distillation Sampling (SDS) loss [Poole et al. 2022] based on T2I models [Jain et al. 2022; Xing et al. 2023b; Zhang et al. 2024], can be extended for style customization by fine-tuning a T2I model on user-provided style examples. Although effective in adapting to new styles, these methods are time-consuming and often produce fragmented or cluttered paths. Such outputs overlook the inherent structural regularity and element relationships within SVG designs, making them difficult to edit or refine further.

Feed-forward T2V methods, on the other hand, are trained on SVG datasets using large language models (LLMs) [Rodriguez et al. 2023; Wu et al. 2023] or diffusion models [Thamizharasan et al. 2024; Xing et al. 2024], maintaining SVG regularity and attaining high-quality outcomes within their respective training domains. However, style customization of the T2V model presents significant challenges. The absence of large-scale, general-purpose text-SVG datasets makes it difficult for the T2V model to disentangle content and style semantics, limiting its ability to generalize to new styles. Consequently, a straightforward approach of fine-tuning a base T2V model with only a few exemplar SVGs, following T2I customization techniques [Hu et al. 2021; Kumari et al. 2022; Ruiz et al. 2022], often leads to overfitting on the exemplar SVGs. Nevertheless, acquiring a sufficient number of consistent style sample SVGs for fine-tuning is impractical due to the scarcity of such data.

Addressing these limitations, we propose a novel two-stage style customization pipeline for SVG generation using only a few exemplar SVGs. It combines the strengths of feed-forward T2V methods to ensure SVG structural regularity and T2I models to acquire powerful customization capabilities. In the first stage, we train a T2V model on black-and-white SVG datasets to focus on learning the contents and structures of SVGs. In the second stage, we learn various styles of SVGs by distilling priors from different customized T2I models. Our two-stage pipeline also helps the T2V model explicitly disentangle content and style semantics.

The aim of the first stage is to train a T2V generative model tailored for style customization. Considering that LLM-based methods generate SVG code in an autoregressive manner, limiting their ability to utilize raster images as supervision, we adopt a diffusion model as the base model to enable customization from the T2I model. As for the representation, global SVG-level representations [Xing et al. 2024] suffer from limited expressivity constrained by the dataset [Rombach et al. 2022], and point-level representations [Thamizharasan et al. 2024] are inefficient for complex SVGs. Thus, we select a path-level representation [Zhang et al. 2024] that ensures both compactness and expressivity. In this stage, our path-level T2V diffusion model learns to generate SVGs that feature text-aligned content and exhibit structural regularity.

In the second stage, we distill styles from a T2I diffusion model to enable style customization for the T2V diffusion model. Specifically, we fine-tune the T2I model using a small set of style images to generate diverse customized images, which serve as augmented data for training the T2V model. To facilitate image-based training, we employ a reparameterization technique [Song et al. 2020] to compute SVG predictions and render them as images, enabling the T2V

model to be updated via an image-level loss. After training, the T2V model can generate SVGs in learned custom styles corresponding to text prompts in a feed-forward manner.

We evaluate our method through comprehensive experiments across vector-level, image-level, and text-level metrics. The results demonstrate the effectiveness of our model in generating high-quality vector graphics with valid SVG structures and diverse customized styles, given input text prompts. Examples of style customization results produced by our framework are shown in Figure 1. Our key contributions are:

- We propose a novel two-stage T2V pipeline to disentangle content and style in SVG generation, which is also the first feed-forward T2V model capable of generating SVGs in custom styles.
- We design a T2V diffusion model based on path-level representations, ensuring structural regularity of SVGs while maintaining diverse expressive capabilities.
- We develop a style customization method for the T2V model by distilling styles from customized image diffusion models.

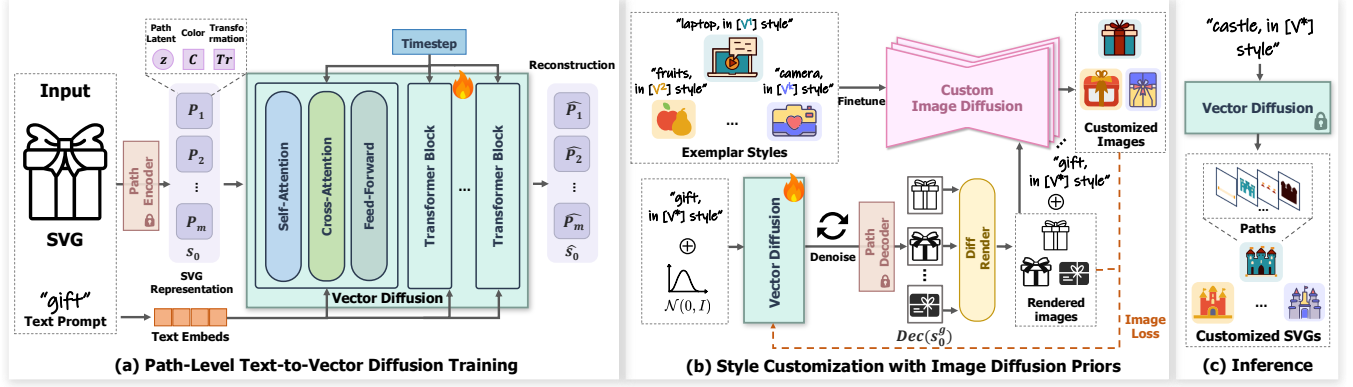
## 2 RELATED WORK

### 2.1 Optimization-based T2V Generation

Optimization-based methods leverage pre-trained vision-language models, such as CLIP [Radford et al. 2021] or diffusion models [Rombach et al. 2022], combined with differentiable rendering [Li et al. 2020] to directly optimize SVG paths. CLIP-based methods [Frans et al. 2022; Schaldenbrand et al. 2022; Song et al. 2022; Vinker et al. 2022] maximize image-text alignment within CLIP latent space. Recent works exploit Score Distillation Sampling (SDS) loss [Poole et al. 2022; Wang et al. 2023b] to capitalize on the strong visual and semantic priors of T2I diffusion models. These methods can produce static [Iluz et al. 2023; Jain et al. 2022; Xing et al. 2023a,b; Zhang et al. 2024] or animated [Gal et al. 2023; Wu et al. 2024] SVGs aligned with text prompts. However, each optimization typically requires tens of minutes per SVG, making them impractical for real design scenarios. Alternatively, some commercial tools [Illustrator 2023; Illustroke 2024] integrate T2I models with vectorization techniques [Dominici et al. 2020; Favreau et al. 2017; Hoshyari et al. 2018; Kopf and Lischinski 2011; Ma et al. 2022; Selinger 2003] to convert raster images into SVGs. Despite their visually appealing results, these methods often include multiple fragmented paths and lack coherent layer relationships in SVGs, complicating further edits. While some methods [Warner et al. 2023; Zhang et al. 2023b] adapt paths from an exemplar SVG via semantic correspondences to preserve layer structure, they are unsuitable for style customization when the source and target differ significantly in semantics.

### 2.2 Feed-forward T2V Generation

Feed-forward methods have explored to learn SVG properties from specialized datasets using large language models or diffusion models. LLM-based approaches [Rodriguez et al. 2023; Tang et al. 2024; Wu et al. 2023] treat SVG scripts as text by designing special tokenization schemes, allowing command sequences to be combined with text tokens in an autoregressive manner. Diffusion-based methods [Thamizharasan et al. 2024; Wang et al. 2023a; Xing et al. 2024] design various SVG representations and model architectures within



**Figure 2: Our two-stage style customization pipeline for SVGs. (a) In Stage 1, we train a path-level T2V diffusion model on black-and-white SVG datasets to focus on learning the contents and structures of SVGs. (b) In Stage 2, we learn various styles of SVGs by distilling priors from different customized T2I models. (c) After training, our T2V model can generate SVGs in custom styles learned during Stage 2 in a feed-forward manner by appending the corresponding style token to the text prompt. Exemplar SVGs are from ©SVGRRepo.**

the vector domain. Although these feed-forward pipelines are conceptually elegant, their generalization capabilities are constrained by the absence of large-scale, general-purpose vector graphics datasets. Consequently, they are limited to producing SVGs in fixed styles, while our approach supports diverse customized styles in a feed-forward manner.

### 2.3 Customization of T2I Generation

Recent advances in T2I customization have enabled flexible adaptation of concepts and styles using only a few reference images [Gal et al. 2022; Kumari et al. 2022; Ruiz et al. 2022]. The pioneering approach DreamBooth [Ruiz et al. 2022] fine-tunes the entire diffusion model by associating user-provided concepts with a unique token. Parameter-efficient fine-tuning (PEFT) methods [Mangrulkar et al. 2022] propose modifying only specific network components, such as low-rank weight offsets [Frenkel et al. 2025; Hu et al. 2021], cross-attention blocks [Kumari et al. 2022; Ye et al. 2023], or adapter layers [Mou et al. 2024; Sohn et al. 2023]. While effective for customizing powerful T2I diffusion models, these techniques are challenging to apply to T2V models, which have limited generalization ability and struggle to disentangle content and style semantics, often leading to overfitting when fine-tuned with only a few exemplar SVGs. Our two-stage style customization pipeline addresses these limitations by leveraging T2I diffusion priors to help the T2V model learn various styles of SVGs.

## 3 OVERVIEW

Our goal is to generate SVGs that are customized to specific styles while aligning with the semantics of given text prompts and maintaining structural regularity. To achieve this, we propose a novel two-stage style customization pipeline designed to disentangle content and style in SVG generation. An illustration of the pipeline is shown in Figure 2.

*Path-Level Text-to-Vector Diffusion Training (Section 4).* In the first stage, we train a T2V model that focuses on learning the contents

and structures of SVGs. We adopt a path-level representation that ensures both compactness and expressivity, and train this path-level T2V diffusion model on black-and-white SVG datasets.

*Style Customization with Image Diffusion Priors (Section 5).* In the second stage, we aim to customize the T2V model to generate SVGs in diverse new styles with only a few exemplars. We fine-tune the T2I diffusion model on a small set of style images to produce diverse customized images, which are used as augmented data to train the T2V model through an image-level loss.

## 4 PATH-LEVEL TEXT-TO-VECTOR DIFFUSION TRAINING

In the first stage, we train a T2V diffusion model to generate SVGs aligned with text semantics while ensuring structural regularity. To achieve this, we adopt a compact and expressive path-level representation, and train the model on black-and-white datasets, focusing on learning SVG content and structure.

### 4.1 SVG Representation

An SVG can be represented as a set of paths, denoted as  $SVG = \{Path_1, Path_2, \dots, Path_m\}$ . A parametric path can be defined as a series of cubic Bézier curves connected end-to-end and filled with a uniform color  $c$ , represented as  $Path_i = (p_1, p_2, \dots, p_d, c)$ , where  $\{p_j\}_{j=1}^d$  are the  $d$  control points used to define the cubic Bézier curves. In contrast to recent approaches that use global SVG-level representations [Xing et al. 2024], which are constrained in expressivity by the limitations of the SVG dataset, or point-level representations [Thamizharasan et al. 2024], which become inefficient for complex SVGs, we adopt a path-level representation that balances compactness and expressivity.

T2V-NPR [Zhang et al. 2024] introduced a path-level SVG VAE designed to effectively capture common shape patterns and geometric constraints within its latent space, ensuring smooth path outputs. Following the methodology of T2V-NPR, we leverage a

pre-trained SVG VAE to encode the  $d$  control points of each path into a latent vector  $z_i$ . This latent vector is then combined with the associated color  $C_i$  and transformation parameters  $Tr_i$  for the  $i$ -th path, denoted as  $\mathbf{P}_i = (z_i, C_i, Tr_i)$ . Using this path-level representation, an SVG tensor can be represented as a sequence of  $m$  paths in the latent space, denoted as  $\mathbf{s}_0 = (\mathbf{P}_1, \mathbf{P}_2, \dots, \mathbf{P}_m)$ , where  $\mathbf{s}_0 \in \mathbb{R}^{d_P \times m}$  and  $d_P$  is the dimension of the path embeddings.

## 4.2 Vector Denoiser

The vector denoiser is trained to reverse a Gaussian diffusion process, enabling the generation of SVG tensors from noisy inputs. In the forward diffusion process, Gaussian noise is progressively added to a sample SVG tensor  $\mathbf{s}_0$  over  $T$  time steps, ultimately transforming it into a unit Gaussian noise  $\mathbf{s}_T \sim \mathcal{N}(0, \mathbf{I})$  [Ho et al. 2020]. At each time step  $t \in \{1, 2, \dots, T\}$ , the vector denoiser predicts the noise content  $\epsilon_\theta$  to be removed from the noisy SVG representation  $\mathbf{s}_t$ .

*Model Architecture.* We adopt a transformer architecture based on DiT [Peebles and Xie 2023] as the backbone for our vector denoiser. As shown in Figure 2(a), the model takes the noisy tensor  $\mathbf{s}_t$  as input and is conditioned on both the text prompt  $\mathbf{y}$  and the time step  $t$ . Each transformer block consists of self-attention, cross-attention, and feed-forward layers. The text prompt  $\mathbf{y}$  is encoded into feature embeddings using the CLIP text encoder [Radford et al. 2021], which interact with vector features through cross-attention similar to [Chen et al. 2023]. Time step embeddings are injected via adaptive layer normalization [Chen et al. 2024] in each transformer block.

*Training Objective.* The training of the T2V diffusion model follows the denoising diffusion probabilistic model (DDPM) framework [Ho et al. 2020]. At each time step  $t$ , Gaussian noise  $\epsilon \sim \mathcal{N}(0, \mathbf{I})$  is added to the original SVG tensor  $\mathbf{s}_0$ , resulting in a noisy representation  $\mathbf{s}_t = \sqrt{\alpha_t}\mathbf{s}_0 + \sqrt{1 - \alpha_t}\epsilon$ , where  $\alpha_t$  is a time-dependent coefficient controlling the noise level. The vector denoiser is trained to predict the added noise  $\epsilon$  in  $\mathbf{s}_t$ , conditioned on the text prompt  $\mathbf{y}$  and the time step  $t$ . The training objective is to minimize the  $L_2$  distance between the predicted noise  $\epsilon_\theta$  and the actual noise  $\epsilon$ :

$$\mathcal{L}_{\text{DM}} = \mathbb{E}_{\mathbf{s}_0, \mathbf{y}, \epsilon, t} [\|\epsilon - \epsilon_\theta(\mathbf{s}_t, t, \mathbf{y})\|_2^2]. \quad (1)$$

## 4.3 Training Details

We train our T2V diffusion model using the *FIGR-8-SVG* dataset [Clouâtre and Demers 2019], which consists of black-and-white vector icons. By eliminating stylistic variations, this dataset enables the model to focus on learning the structures and semantics of SVGs in the first stage. To preprocess the data, we follow the same steps as IconShop [Wu et al. 2023] to obtain valid SVG data and their corresponding text descriptions. Examples from the dataset are shown in Figure 3(a). For the raw SVG data, we convert all other primitive shapes (e.g., lines, rectangles and ellipses) into cubic Bézier curves, which are encoded as path embeddings as described in Section 4.1. Since the number of paths varies across SVGs, we pad the sequences of path tensors with zeros to a fixed length of 32 and filter out SVGs with more paths. This results in 210,000 samples for model training.

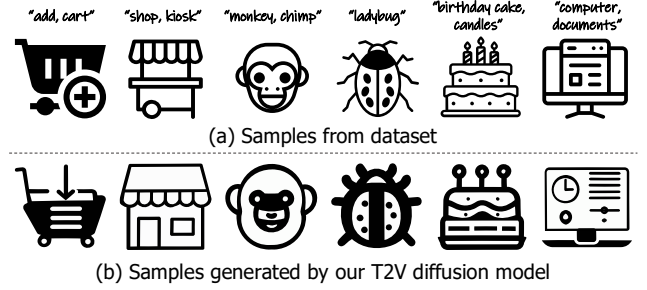


Figure 3: (a) SVG examples from the dataset. (b) SVG samples generated from random noise by our T2V diffusion model in Stage 1.

In our implementation, we set  $d_P = 28$  and  $m = 32$  for the SVG representation and normalize the SVG embeddings to the range of  $[-1, 1]$ . The model architecture consists of 28 transformer blocks, a hidden dimension of 800, and 12 attention heads. We configure the number of diffusion steps to  $T = 1000$  and employ a cosine noise schedule [Ho et al. 2020]. During training, we apply classifier-free guidance [Ho and Salimans 2022] by randomly zeroing the text prompt  $\mathbf{y}$  with a probability of 10%. We use the Adam optimizer with an initial learning rate of  $3 \times 10^{-5}$ . The T2V diffusion network is trained for 3000 epochs with a batch size of 64, taking approximately 6 days on 8 A6000 GPUs.

After the first stage of training, our T2V diffusion model generates high-quality SVGs that align with text prompts while maintaining the structural integrity of the SVGs. In Figure 3(b), we show several SVG samples generated by our model from random noise.

## 5 STYLE CUSTOMIZATION WITH IMAGE DIFFUSION PRIORS

In the second stage, we aim to enable style customization for the T2V diffusion model using only a few exemplar SVGs. A straightforward method of fine-tuning the T2V model with such a small dataset leads to overfitting. To address this issue, we distill style priors from different customized T2I models to generate a diverse set of customized images, which serve as augmented data for training the T2V model via an image-level loss.

### 5.1 Style Distillation from Image Diffusion

T2I diffusion models serve as a powerful prior for generating diverse images in customized styles. To enable the model to produce images in desired styles, we fine-tune a base T2I diffusion model "SD-v1-5" checkpoint<sup>1</sup>, using a small set of style reference images. By applying the DreamBooth-LoRA method [Hu et al. 2021; Ruiz et al. 2022], we create distinct LoRAs for each style. After fine-tuning, we concatenate a unique token  $[V^*]$  with the text prompt (e.g., "in  $[V^*]$  style") to generate customized images in the corresponding style using the specific LoRA.

Inspired by distillation techniques in T2I diffusion models [Luhman and Luhman 2021; Yin et al. 2024], which generate noise-image pairs by running the sampling steps of the teacher diffusion model

<sup>1</sup><https://huggingface.co/runwayml/stable-diffusion-v1-5>

to train the student model, we apply a similar approach to generate customized images as guidance for training. Given a text prompt  $y$ , we randomly sample Gaussian noise  $\epsilon$  and input both into the T2V model. The T2V model performs DDPM denoising process to generate an SVG representation  $\mathbf{s}_0^g$ , which is then passed through a pre-trained path decoder [Zhang et al. 2024] and a differentiable rasterizer  $\mathcal{R}$  [Li et al. 2020] to produce an image  $I_0^g = \mathcal{R}(\text{Dec}(\mathbf{s}_0^g))$ . At this stage,  $I_0^g$  is a black-and-white style image. To ensure that the guidance image generated by the T2I model aligns with the structure of  $I_0^g$  without significant deviations, we integrate ControlNet [Zhang et al. 2023a] into the customized T2I model. This ensures overall structural alignment between the customized image  $I_0^c$  and  $I_0^g$ . By using the Canny edge map of  $I_0^g$  as a control image, we preserve the structural integrity of the original SVG while incorporating the desired style. Using this approach, we can generate the corresponding  $(\mathbf{s}_0^g, I_0^c)$  pair from random noise given a text prompt  $y$ , and fine-tune the T2V model with image-level loss.

## 5.2 Style Fine-tuning

Given a generated SVG representation  $\mathbf{s}_0^g$  and the corresponding customized image  $I_0^c$ , we fine-tune the T2V model towards new custom styles using image-level loss and diffusion loss. In the forward diffusion process, Gaussian noise is added to  $\mathbf{s}_0^g$ , resulting in a noisy representation  $\mathbf{s}_t^g$ . We apply a reparameterization technique [Song et al. 2020] to predict the denoised SVG tensor  $\hat{\mathbf{s}}_0^g$  at each time step, by rewriting the closed-form sampling distribution for the forward diffusion process as:

$$\hat{\mathbf{s}}_0^g = (\mathbf{s}_t^g - \sqrt{1 - \bar{\alpha}_t} \cdot \epsilon_\theta(\mathbf{s}_t^g, t)) / \sqrt{\bar{\alpha}_t}. \quad (2)$$

By predicting  $\hat{\mathbf{s}}_0^g$ , we obtain the rendered image  $\hat{I}_0^g = \mathcal{R}(\text{Dec}(\hat{\mathbf{s}}_0^g))$ . The image loss is computed as the MSE between the rendered image  $\hat{I}_0^g$  and the customized image  $I_0^c$ :

$$\mathcal{L}_{\text{img}} = \omega_t \|\hat{I}_0^g - I_0^c\|^2, \quad (3)$$

where  $\omega_t$  is a time-dependent weighting function designed to stabilize training by deactivating the loss term for noisier time steps. We set  $\omega_t = (1 - \bar{\alpha}_t)$  empirically, following [Crowson et al. 2024]. The image loss guides the T2V model to predict SVGs that match the customized image, reflecting the desired style. Additionally, we incorporate the diffusion loss  $\mathcal{L}_{\text{DM}}$  defined on  $\hat{\mathbf{s}}_0^g$ , as described in Equation 1, to help the model learn the new data distribution of the predicted SVGs  $\hat{\mathbf{s}}_0^g$ . Specifically, Gaussian noise is added to  $\hat{\mathbf{s}}_0^g$ , and the model predicts this noise, with the diffusion loss calculated using Equation 1. During training, the T2V model is updated based on the combined loss function  $\mathcal{L} = \mathcal{L}_{\text{img}} + \mathcal{L}_{\text{DM}}$ .

To enable our T2V diffusion model to generate SVGs in diverse custom styles, we select 200 distinct style reference sets from SVGRepo<sup>2</sup>, iconfont<sup>3</sup> and Freepik<sup>4</sup>, each set containing a small collection of exemplar SVGs (ranging from 1 to 30 SVGs per set). We train the T2V model simultaneously across all 200 styles, with each style distinguished by a unique token "[V\*]". The training process lasts for 80K iterations with a batch size of 20, where each iteration generates 20 pairs of  $(\mathbf{s}_0^g, I_0^c)$  from randomly sampled text prompts

and Gaussian noise across different styles. We employ a learning rate of  $4 \times 10^{-6}$ , taking approximately 6 days using 8 A6000 GPUs. After style fine-tuning, our T2V model can generate SVGs in the learned custom styles based on text prompts in a feed-forward manner.

Our method also supports fine-tuning on a single style or incrementally adding a new style with only a few exemplars via either full-model or LoRA fine-tuning. Similar to DreamBooth [Ruiz et al. 2022], the former approach fine-tunes the full T2V model with a new style represented by a token "[V\*]" that is distinct from all existing style tokens. For a parameter-efficient alternative, we fine-tune external LoRA weights for the attention layers of DiT blocks [Hu et al. 2021], adapting the model to a new style without introducing an additional style token.

## 6 EXPERIMENTS

*Experiment Setup.* To evaluate our method, we randomly select 5 text prompts from the FIGR-8-SVG dataset [Clouâtre and Demers 2019] for each of the 200 styles, resulting in a total of 1000 vector graphics. For each style, we append the special style token "in [V\*] style" to the respective text prompt. During testing, we use the DDPM sampler with 1000 steps and classifier-free guidance with a scale of 3 to achieve better results. Generating an SVG takes around 25 seconds on an NVIDIA-A6000.

*Evaluation Metrics.* We evaluate the quality of our results from vector-level, image-level, text-level perspectives. For **vector-level** evaluation, we use a path VAE [Zhang et al. 2024] trained on the FIGR-8-SVG dataset to encode SVG paths into latent vectors. We calculate the FID between these latents and the ground truth paths from FIGR-8-SVG, to evaluate how well the paths align with well-designed vector graphics. For **image-level** evaluation, we evaluate the style alignment and the visual aesthetics of SVGs. We measure style alignment by calculating the average cosine similarity between CLIP image features of style references and rendered SVG images [Sohn et al. 2023]. We use the Aesthetic score [Schuhmann 2022] to evaluate the overall image quality. For **text-level** evaluation, we calculate the CLIP cosine similarity [Radford et al. 2021] between the text prompt and the rendered SVGs to measure semantic alignment.

*Baselines.* We compare our proposed pipeline with two types of T2V generation schemes: optimization-based methods and feed-forward methods.

**Optimization-based methods** rely on pre-trained T2I models, so we first perform style-tuning on T2I diffusion models using the method described in Section 5.1. For vectorization with T2I methods, we generate customized images and optimize the SVGs to fit the images using two distinct vectorization techniques: a traditional method, Potrace [Selinger 2003], and a deep learning-based method LIVE [Ma et al. 2022]. For text-guided SVG optimization, we compare three approaches: VectorFusion [Jain et al. 2022] using SDS loss, SVGDreamer [Xing et al. 2023b] and T2I-NPR [Zhang et al. 2024], which use VSD loss. We use 64 paths for SVG optimization. To enhance alignment with the exemplar style, we begin by using the vectorized outputs from customized images as the initial SVGs.

<sup>2</sup><https://www.svgrepo.com>

<sup>3</sup><https://www.iconfont.cn>

<sup>4</sup><https://www.freepik.com>



**Feed-forward methods** include language-based and diffusion-based approaches. For the former, we use GPT-4o [Achiam et al. 2023] to generate customized SVGs via providing curated in-context examples in the prompts. Specifically, we supply raster images and corresponding SVG scripts of the exemplar style and let GPT-4o act as an SVG code generator, to generate SVGs that match the style of the exemplars with the given text prompts. For the latter, since no diffusion-based T2V models are publicly available yet, we reproduce VecFusion [Thamizharasan et al. 2024] as a base T2V model. To achieve style customization, we compare two approaches: (1) an vector-based fine-tuning method, in which we fine-tune VecFusion with a small set of exemplar SVGs [Ruiz et al. 2022]; (2) a neural style transfer (NST) method for SVG [Efimova et al. 2023], where we first generate an SVG using the base model, then select an exemplar SVG as the style reference and apply style transfer to the model’s output.

## 6.1 Comparisons

We evaluate the performance of our method by comparing it with baselines qualitatively and quantitatively. The quantitative results are provided in Table 1 and the qualitative results are presented in Figure 4, Figure 5, Figure 9 and Figure 10. As shown in Table 1, our method outperforms the others from a comprehensive perspective.

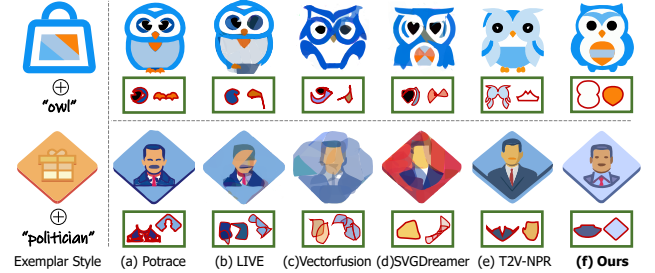
*Comparisons with Optimization-based Methods.* Vectorization with T2I methods reconstruct customized images through color-based image segmentation and curve fitting. However, as shown in Figure 4 and Figure 9(c), while Potrace [Selinger 2003] produces visually appealing outputs by faithfully reconstructing the customized images, it struggles with overly complex vector elements and lacks layer organization, as indicated by the higher Path FID in Table 1. This results in disorganized structures, reduced semantic clarity, and increased complexity in the SVGs. Such issues are common in image vectorization methods and contradict professional design principles, which prioritize simplicity and clarity in vector graphics. LIVE [Ma et al. 2022] faces similar challenges, producing SVGs containing numerous irregular and broken paths. Zoomed-in illustrations in Figure 9 highlight the issues of overcomplicated and fragmented paths (shown within green boxes).

Text-guided SVG optimization methods leverage score distillation in T2I diffusion models to optimize a set of shapes. VectorFusion [Jain et al. 2022] and SVGDreamer [Xing et al. 2023b] directly optimize the control points of paths to generate text-conforming SVGs. However, due to the high degrees of freedom, the paths may undergo complex transformations that result in jagged and cluttered shapes, leading to visually unappealing outcomes. T2V-NPR [Zhang et al. 2024] tackles the issue of irregular paths by learning a latent representation of paths and reduces fragmentation by merging shapes with similar colors. However, while it produces smooth paths, it cannot guarantee the semantic integrity of the shapes, as the merging operation overlooks their semantic meaning. This can lead to semantically ambiguous paths, such as the merging of an owl’s eye with its wing, as shown in the first row of Figure 4.

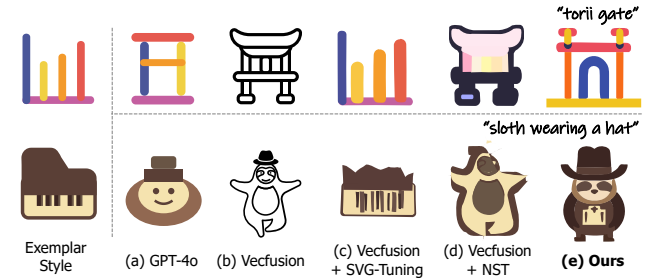
Overall, optimization-based methods that rely only on image supervision overlook the inherent design principles and layer structure of SVGs. Consequently, the generated SVGs often contain redundant shapes and disorganized layers, making them difficult

**Table 1: Quantitative comparison with existing methods.**

Methods		Path FID ↓	Style Alignment ↑	Visual Aesthetic ↑	Text Alignment ↑
Optimization	Potrace	44.29	0.665	5.522	0.294
	LIVE	52.43	0.578	4.686	0.258
	Vectorfusion	53.76	0.557	4.892	0.276
	SVGDreamer	48.51	0.564	5.013	0.281
	T2V-NPR	40.25	0.608	5.237	0.290
Feed-forward	GPT-4o	38.14	0.549	5.041	0.251
	VecF + SVG-FT	45.05	<b>0.726</b>	4.980	0.223
	VecF + NST	58.12	0.573	4.574	0.245
	<b>Ours</b>	<b>37.51</b>	0.661	<b>5.527</b>	<b>0.297</b>



**Figure 4: Qualitative comparison with optimization-based T2V methods. Exemplar SVGs are from ©SVGRepo.**



**Figure 5: Qualitative comparison to feed-forward T2V methods. Exemplar SVGs are from ©SVGRepo.**

to edit. Moreover, these methods are time-consuming due to their iterative optimization, typically requiring tens of minutes per SVG, which makes them impractical for real design scenarios. In contrast, our T2V diffusion model learns vector properties, such as valid path semantics and layer structure, by training on a well-designed SVG dataset. Our novel two-stage training strategy enables feed-forward generation of well-structured SVGs in a few seconds, while achieving visually appealing results in diverse custom styles.

*Comparisons with Feed-forward Methods.* Regarding language-based methods, though provided with in-context SVG examples, GPT-4o [Achiam et al. 2023] can only generate simple combinations of basic primitive shapes (e.g., circles and rectangles) to align with text prompts, as illustrated in Figure 5(a). It fails to produce the complex geometric details required for professional SVGs, resulting in outputs inadequate for graphic design.

Although the original VecFusion model [Thamizharasan et al. 2024], trained on the *FIGR-8-SVG* dataset, generates high-quality

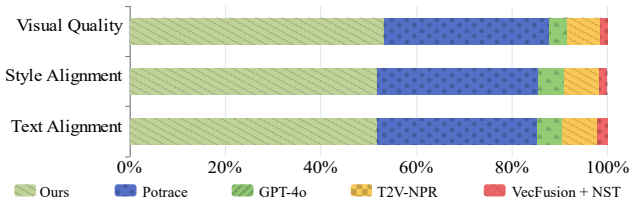


Figure 6: User Study. We show the human preferences in %.

results within its trained domains, it cannot be extended to style customization using existing methods. When applying an vector-based fine-tuning approach, where VecFusion is fine-tuned with a small set of exemplar SVGs [Ruiz et al. 2022], its limited generalization ability prevents it from generating semantically correct SVGs in new custom styles. Instead, the model overfits to the exemplar SVGs, simply reconstructing them rather than adapting to diverse prompts. As a result, the generated outputs exhibit high Style Alignment but poor Text Alignment in Table 1.

NST [Efimova et al. 2023] applies style transfer to the SVGs generated by VecFusion using a style loss in image space. Although this method directly inherits the original layer-wise properties, the optimized SVGs often have messy visual appearances. Furthermore, it struggles to capture fine-grained style features, leading to poor style consistency.

In contrast, our method excels at adapting to the style, effectively capturing details from user-provided style, such as color schemes and design patterns. It achieves high visual quality while preserving the structure of the output SVGs.

## 6.2 User Study

We conducted a perceptual study to evaluate our style customization of T2V generation from three perspectives: overall SVG quality, style alignment and semantic alignment. We randomly selected 20 text prompts from the dataset and generated SVGs using both the baseline methods and our approach. Each question presented the results of different methods in a random order, and 30 participants were given unlimited time to select the best result among five options for each evaluation metric. Figure 6 demonstrates the superior performance of our method, as it achieves the highest preference in all evaluation metrics. Specifically, our method obtains 53.2% of votes for overall SVG quality, 51.8% for style alignment, and 51.7% for semantic alignment. The results show the effectiveness of our method in generating high-quality SVGs in custom styles from text prompts that align more closely with human perception.

## 6.3 Ablation Study

*Ablation on SVG Representation.* Instead of using our path-level representation for training the T2V diffusion model, another baseline is to use a global SVG-level representation. Following the DeepSVG [Carrier et al. 2020] architecture, we train a transformer-based VAE on the *FIGR-8-SVG* dataset, where all paths with their properties (including the path order, control points and color) are encoded into a single latent vector. We then replace our path-level representation with this SVG-level representation for T2V diffusion model

Table 2: Ablation study on SVG representation and style customization with image diffusion priors.

Methods	Path FID ↓	Style Alignment ↑	Visual Aesthetic ↑	Text Alignment ↑
SVG-level Rep	45.16	0.406	3.285	0.288
SVG-FT	57.32	<b>0.722</b>	4.926	0.221
<b>Ours</b>	<b>37.51</b>	0.661	<b>5.527</b>	<b>0.297</b>

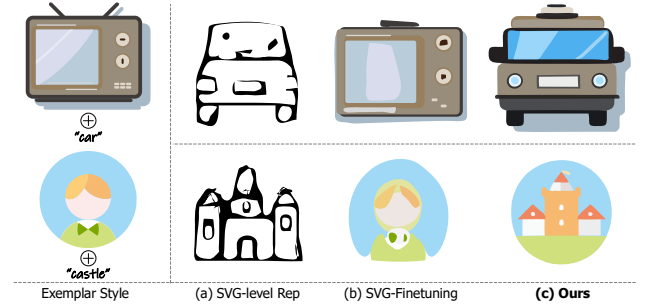


Figure 7: Qualitative results on ablation study. Exemplar SVGs are from ©SVGRepo.

training and subsequent style distillation. However, the global SVG-level representation is constrained by the geometry and color limitations of the dataset, which restricts its ability to generate SVGs in only a fixed style. As a result, it fails to adapt to new custom styles, as shown in Figure 7(a). In contrast, our path-level representation maintains both compactness and expressivity, allowing for flexible and diverse SVG customizations.

*Ablation on Style Customization with Image Diffusion Priors.* We compare our image-based style customization method with a vector-based fine-tuning approach. Specifically, in the second stage, we directly fine-tune our T2V model using a small set of exemplar SVGs, following the fine-tuning techniques of T2I diffusion models [Ruiz et al. 2022]. However, as shown in Figure 7(b), this method leads to overfitting on the exemplar SVGs, causing the model to simply reconstruct them rather than aligning with the text semantics, as reflected by a high style alignment score and a low text alignment score in Table 2. In contrast, our style distillation method from image diffusion takes advantage of the strong visual priors in T2I diffusion models to generate customized images as augmented data, enabling diverse style customizations of SVGs.

## 7 CONCLUSION

In this paper, we present a novel two-stage pipeline for style customization of SVGs. Our approach disentangles content and style semantics in the T2V diffusion model, ensuring structural regularity and expressive diversity in the generated SVGs. By employing a path-level T2V diffusion model and distilling styles from T2I diffusion priors, our method produces high-quality SVGs in custom styles from text prompts in a feed-forward manner. While our method excels at SVG style customization, it has limitations. First, our T2V model is trained on the *FIGR-8-SVG* dataset, which

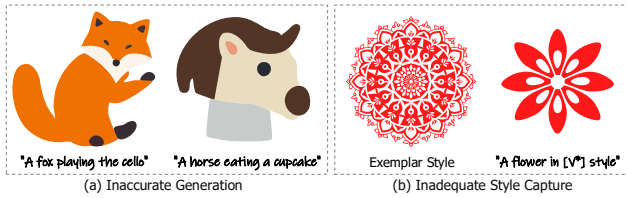


Figure 8: Failure cases. The exemplar SVG is from ©iconfont.

contains only simple class labels, limiting the model’s semantic understanding of SVG content. For example, as shown in Figure 8(a), semantic elements like "cello" and "cupcake" are inaccurate when the text descriptions exceed the training domain’s capacity. This could be mitigated with a larger and higher-quality SVG dataset with detailed annotations. Second, it may lose fine-grained stylistic details for overly complex style references, as depicted in Figure 8(b). Our model can be used to synthesize SVG data, and with advanced diffusion model techniques, it enables flexible control and editing, which we plan to explore in future work.

## ACKNOWLEDGMENTS

The work described in this paper was substantially supported by a GRF grant from the Research Grants Council (RGC) of the Hong Kong Special Administrative Region, China [Project No. CityU 11216122].

## REFERENCES

- Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).
- Alexandre Carlier, Martin Danelljan, Alexandre Alahi, and Radu Timofte. 2020. Deepsvg: A hierarchical generative network for vector graphics animation. *Advances in Neural Information Processing Systems* 33 (2020), 16351–16361.
- Junsong Chen, Jincheng Yu, Chongjian Ge, Lewei Yao, Enze Xie, Yue Wu, Zhongdao Wang, James Kwok, Ping Luo, Huchuan Lu, et al. 2023. Pixart- $\alpha$ : Fast training of diffusion transformer for photorealistic text-to-image synthesis. *arXiv preprint arXiv:2310.00426* (2023).
- Jian Chen, Ruiyi Zhang, Yufan Zhou, Rajiv Jain, Zhiqiang Xu, Ryan Rossi, and Changyou Chen. 2024. Towards aligned layout generation via diffusion model with aesthetic constraints. *arXiv preprint arXiv:2402.04754* (2024).
- Louis Clouâtre and Marc Demers. 2019. Figr: Few-shot image generation with reptile. *arXiv preprint arXiv:1901.02199* (2019).
- Katherine Crowson, Stefan Andreas Baumann, Alex Birch, Tanishq Mathew Abraham, Daniel Z Kaplan, and Enrico Shippole. 2024. Scalable high-resolution pixel-space image synthesis with hourglass diffusion transformers. In *Forty-first International Conference on Machine Learning*.
- Edoardo Alberto Dominici, Nico Schertler, Jonathan Griffin, Shayan Hoshayari, Leonid Sigal, and Alla Sheffer. 2020. Polyfit: Perception-aligned vectorization of raster clip-art via intermediate polygonal fitting. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 77–1.
- Valeria Efimova, Artyom Chebykin, Ivan Jarsky, Evgenii Prosvirnin, and Andrey Filchenkov. 2023. Neural Style Transfer for Vector Graphics. *arXiv preprint arXiv:2303.03405* (2023).
- Jean-Dominique Favreau, Florent Lafarge, and Adrien Bousseau. 2017. Photo2clipart: Image abstraction and vectorization using layered linear gradients. *ACM Transactions on Graphics (TOG)* 36, 6 (2017), 1–11.
- Kevin Frans, Lisa Soros, and Olaf Witkowski. 2022. Clipdraw: Exploring text-to-drawing synthesis through language-image encoders. *Advances in Neural Information Processing Systems* 35 (2022), 5207–5218.
- Yarden Frenkel, Yael Vinker, Ariel Shamir, and Daniel Cohen-Or. 2025. Implicit style-content separation using b-lora. In *European Conference on Computer Vision*. Springer, 181–198.
- Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit H Bermano, Gal Chechik, and Daniel Cohen-Or. 2022. An image is worth one word: Personalizing text-to-image generation using textual inversion. *arXiv preprint arXiv:2208.01618* (2022).

- Rinon Gal, Yael Vinker, Yuval Alaluf, Amit H Bermano, Daniel Cohen-Or, Ariel Shamir, and Gal Chechik. 2023. Breathing Life Into Sketches Using Text-to-Video Priors. *arXiv preprint arXiv:2311.13608* (2023).
- Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems* 33 (2020), 6840–6851.
- Jonathan Ho and Tim Salimans. 2022. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598* (2022).
- Shayan Hoshayari, Edoardo Alberto Dominici, Alla Sheffer, Nathan Carr, Zhaowen Wang, Duygu Ceylan, and I-Chao Shen. 2018. Perception-driven semi-structured boundary vectorization. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–14.
- Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685* (2021).
- Adobe Illustrator. 2023. Turn ideas into illustrations with Text to Vector Graphic. <https://www.adobe.com/products/illustrator/text-to-vector-graphic.html>.
- Illustrator. 2024. Stunning vector illustrations from text prompts. <https://illustrator.com/>.
- Shir Iluz, Yael Vinker, Amir Hertz, Daniel Berio, Daniel Cohen-Or, and Ariel Shamir. 2023. Word-as-image for semantic typography. *arXiv preprint arXiv:2303.01818* (2023).
- Ajay Jain, Amber Xie, and Pieter Abbeel. 2022. VectorFusion: Text-to-SVG by Abstracting Pixel-Based Diffusion Models. *arXiv preprint arXiv:2211.11319* (2022).
- Johannes Kopf and Dani Lischinski. 2011. Depixelizing pixel art. In *ACM SIGGRAPH 2011 papers*. 1–8.
- Nupur Kumari, Bingliang Zhang, Richard Zhang, Eli Shechtman, and Jun-Yan Zhu. 2022. Multi-Concept Customization of Text-to-Image Diffusion. *arXiv preprint arXiv:2212.04488* (2022).
- Tzu-Mao Li, Michal Lukáč, Michaël Gharbi, and Jonathan Ragan-Kelley. 2020. Differentiable vector graphics rasterization for editing and learning. *ACM Transactions on Graphics (TOG)* 39, 6 (2020), 1–15.
- Eric Luhman and Troy Luhman. 2021. Knowledge distillation in iterative generative models for improved sampling speed. *arXiv preprint arXiv:2101.02388* (2021).
- Xu Ma, Yuqian Zhou, Xingqian Xu, Bin Sun, Valerii Filev, Nikita Orlov, Yun Fu, and Humphrey Shi. 2022. Towards layer-wise image vectorization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 16314–16323.
- Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. 2022. PEFT: State-of-the-art Parameter-Efficient Fine-Tuning methods. <https://github.com/huggingface/peft>.
- Chong Mou, Xintao Wang, Liangbin Xie, Yanze Wu, Jian Zhang, Zhongang Qi, and Ying Shan. 2024. T2i-adaptor: Learning adapters to dig out more controllable ability for text-to-image diffusion models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 4296–4304.
- William Peebles and Saining Xie. 2023. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 4195–4205.
- Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. 2022. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988* (2022).
- Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*. PMLR, 8748–8763.
- Juan A Rodriguez, Shubham Agarwal, Issam H Laradji, Pau Rodriguez, David Vazquez, Christopher Pal, and Marco Pedersoli. 2023. StarVector: Generating Scalable Vector Graphics Code from Images. *arXiv preprint arXiv:2312.11556* (2023).
- Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10684–10695.
- Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. 2022. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. *arXiv preprint arXiv:2208.12242* (2022).
- Peter Schaldenbrand, Zhixuan Liu, and Jean Oh. 2022. Styleclipdraw: Coupling content and style in text-to-drawing translation. *arXiv preprint arXiv:2202.12362* (2022).
- Christoph Schuhmann. 2022. Improved Aesthetic Predictor. <https://github.com/christophschuhmann/improved-aesthetic-predictor>.
- Peter Selinger. 2003. Potrace: a polygon-based tracing algorithm.
- Kihyuk Sohn, Nataniel Ruiz, Kimin Lee, Daniel Castro Chin, Irina Blok, Huiwen Chang, Jarred Barber, Lu Jiang, Glenn Entis, Yuanzhen Li, et al. 2023. Styledrop: Text-to-image generation in any style. *arXiv preprint arXiv:2306.00983* (2023).
- Yiren Song, Xning Shao, Kang Chen, Weidong Zhang, Minzhe Li, and Zhongliang Jing. 2022. CLIPVG: Text-Guided Image Manipulation Using Differentiable Vector Graphics. *arXiv preprint arXiv:2212.02122* (2022).
- Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. 2020. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456* (2020).
- Zezechang Tang, Chenfei Wu, Zekai Zhang, Mingheng Ni, Shengming Yin, Yu Liu, Zhengyuan Yang, Lijuan Wang, Zicheng Liu, Juntao Li, et al. 2024. StrokeNUWA: Tokenizing Strokes for Vector Graphic Synthesis. *arXiv preprint arXiv:2401.17093* (2024).



- Vikas Thamizharasan, Difan Liu, Shantanu Agarwal, Matthew Fisher, Michaël Gharbi, Oliver Wang, Alec Jacobson, and Evangelos Kalogerakis. 2024. VecFusion: Vector Font Generation with Diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7943–7952.
- Yael Vinker, Ehsan Pajouheshgar, Jessica Y Bo, Roman Christian Bachmann, Amit Haim Bermano, Daniel Cohen-Or, Amir Zamir, and Ariel Shamir. 2022. Clipasso: Semantically-aware object sketching. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–11.
- Qiang Wang, Haoge Deng, Yonggang Qi, Da Li, and Yi-Zhe Song. 2023a. Sketchknitter: Vectorized sketch generation with diffusion models. In *The Eleventh International Conference on Learning Representations*.
- Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. 2023b. ProlificDreamer: High-Fidelity and Diverse Text-to-3D Generation with Variational Score Distillation. *arXiv preprint arXiv:2305.16213* (2023).
- Jeremy Warner, Kyu Won Kim, and Bjoern Hartmann. 2023. Interactive Flexible Style Transfer for Vector Graphics. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. 1–14.
- Ronghuan Wu, Wanchao Su, Kede Ma, and Jing Liao. 2023. IconShop: Text-Guided Vector Icon Synthesis with Autoregressive Transformers. *ACM Transactions on Graphics (TOG)* 42, 6 (2023), 1–14.
- Ronghuan Wu, Wanchao Su, Kede Ma, and Jing Liao. 2024. AniClipart: Clipart animation with text-to-video priors. *International Journal of Computer Vision* (2024), 1–17.
- Ximing Xing, Juncheng Hu, Jing Zhang, Dong Xu, and Qian Yu. 2024. SVGFusion: Scalable Text-to-SVG Generation via Vector Space Diffusion. *arXiv preprint arXiv:2412.10437* (2024).
- Ximing Xing, Chuang Wang, Haitao Zhou, Jing Zhang, Qian Yu, and Dong Xu. 2023a. DiffSketcher: Text Guided Vector Sketch Synthesis through Latent Diffusion Models. *arXiv preprint arXiv:2306.14685* (2023).
- Ximing Xing, Haitao Zhou, Chuang Wang, Jing Zhang, Dong Xu, and Qian Yu. 2023b. SVGDreamer: Text Guided SVG Generation with Diffusion Model. *arXiv preprint arXiv:2312.16476* (2023).
- Hu Ye, Jun Zhang, Sibio Liu, Xiao Han, and Wei Yang. 2023. Ip-adapter: Text compatible image prompt adapter for text-to-image diffusion models. *arXiv preprint arXiv:2308.06721* (2023).
- Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T Freeman, and Taesung Park. 2024. One-step diffusion with distribution matching distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6613–6623.
- Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. 2023a. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 3836–3847.
- Peiyang Zhang, Nanxuan Zhao, and Jing Liao. 2023b. Text-Guided Vector Graphics Customization. In *SIGGRAPH Asia 2023 Conference Papers*. 1–11.
- Peiyang Zhang, Nanxuan Zhao, and Jing Liao. 2024. Text-to-vector generation with neural path representation. *ACM Transactions on Graphics (TOG)* 43, 4 (2024), 1–13.

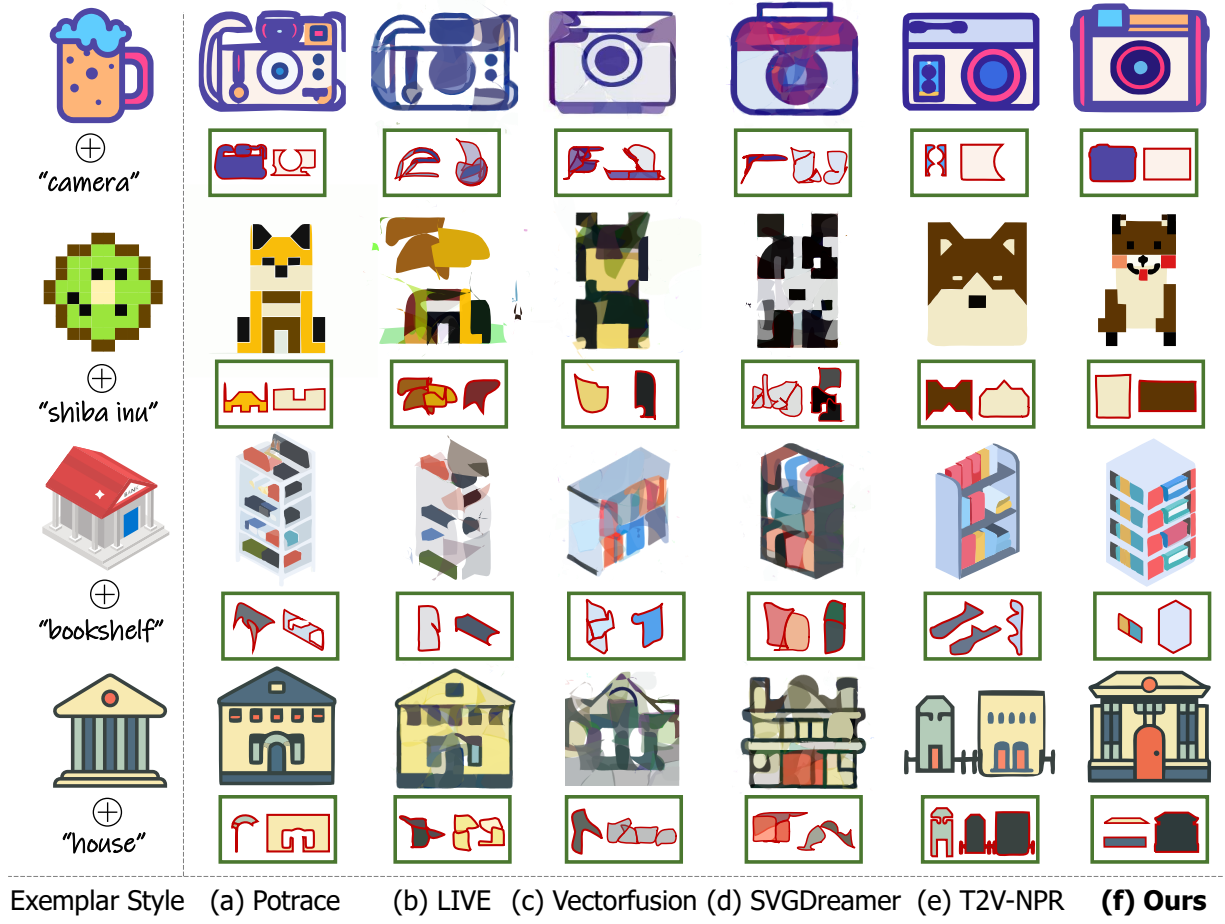


Figure 9: More qualitative comparison with optimization-based T2V methods. Exemplar SVGs: the 1<sup>st</sup>, 2<sup>nd</sup> and 4<sup>th</sup> rows are from ©SVGRepo; the 3<sup>rd</sup> row is from ©Freepik.

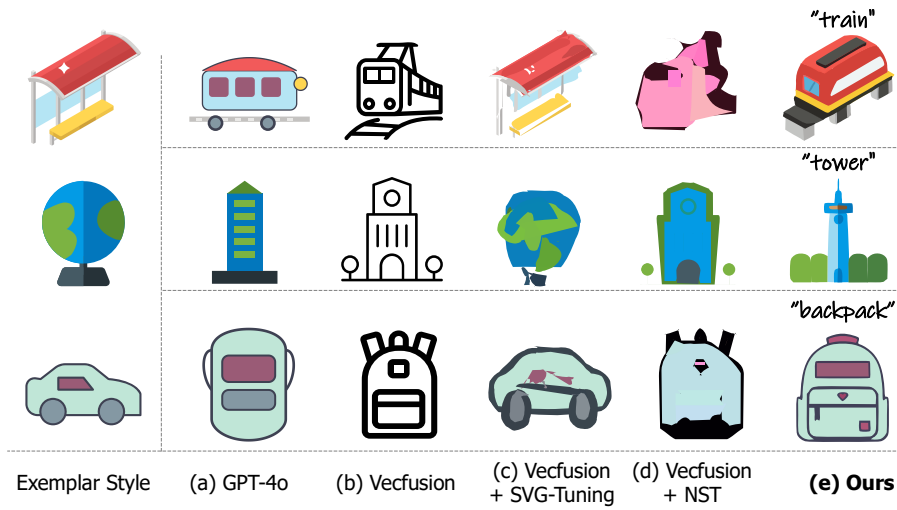


Figure 10: More qualitative comparison to feed-forward T2V methods. Exemplar SVGs: the 1<sup>st</sup> row is from ©Freepik; the 2<sup>nd</sup> and 3<sup>rd</sup> rows are from ©SVGRepo.

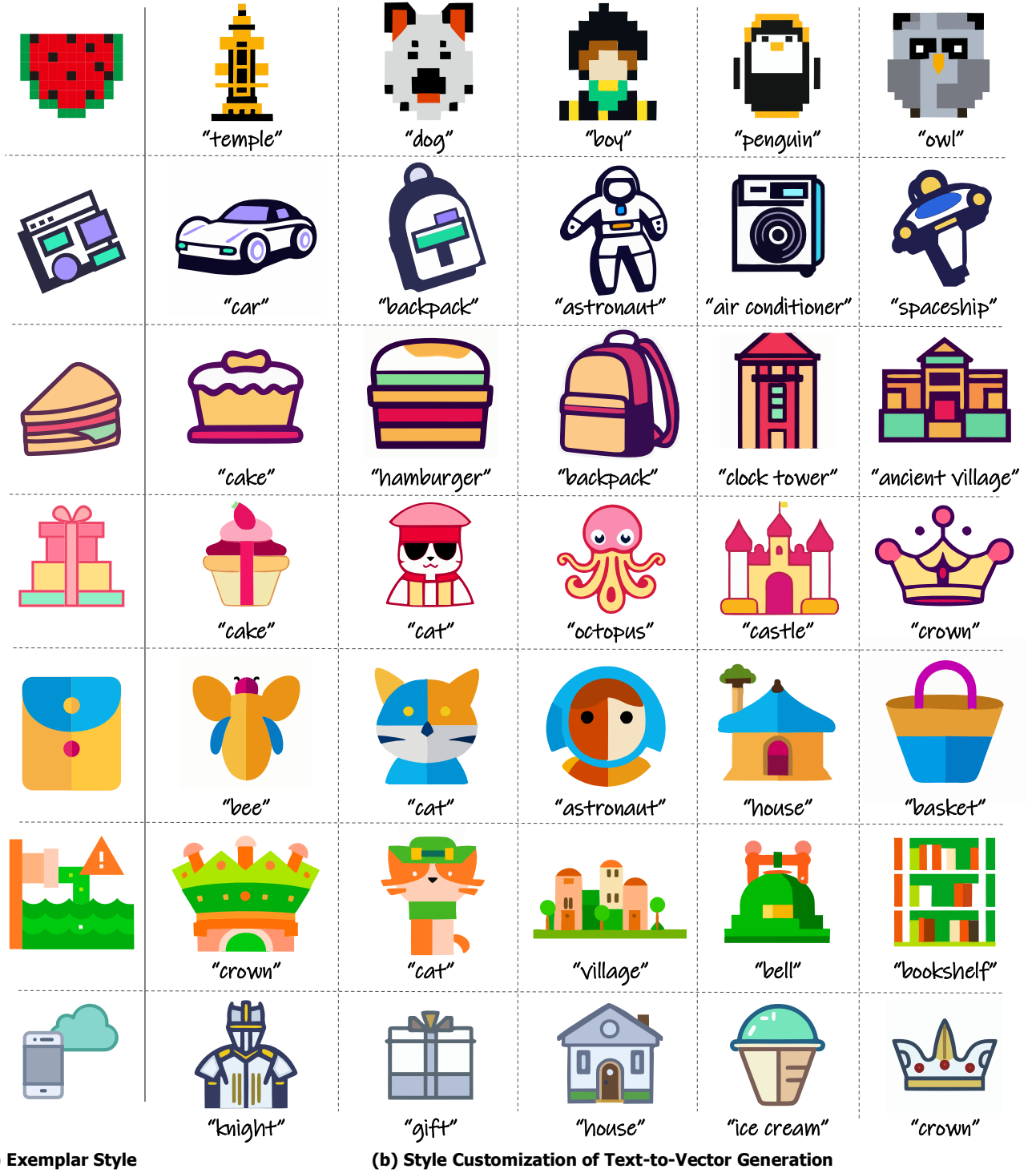


Figure 11: More results of our style customization of T2V generation. Exemplar SVGs: the 1<sup>st</sup>, 2<sup>nd</sup>, 3<sup>rd</sup>, 4<sup>th</sup>, 5<sup>th</sup> and 7<sup>th</sup> rows are from ©SVGRepo; the 6<sup>th</sup> row is from ©Freepik.