Vector Graphics Generation via Mutually Impulsed Dual-domain Diffusion

Zhongyin Zhao[1] Ye Chen[1] Zhangli Hu[1] Xuanhong Chen[1,2] Bingbing Ni[1,2*]
[1]Shanghai Jiao Tong University, Shanghai 200240, China
[2]USC-SJTU Institute of Cultural and Creative Industry
{zhao_zhongyin, chenye123, tension-2019, chen19910528, nibingbing}@sjtu.edu.cn

Abstract

Intelligent generation of vector graphics has very promising applications in the fields of advertising and logo design, artistic painting, animation production, etc. *However, current mainstream vector image generation methods lack the encoding of image appearance information that is associated with the original vector representation and therefore lose valid supervision signal from the strong correlation between the discrete vector parameter (drawing instruction) sequence and the target shape/structure of the corresponding pixel image. On the one hand, the generation process based on pure vector domain completely ignores the similarity measurement between shape parameter (and their combination) and the paired pixel image appearance pattern; on the other hand, two-stage methods (*i.e.*, generation-and-vectorization) based on pixel diffusion followed by differentiable image-to-vector translation suffer from wrong error-correction signal caused by approximate gradients. To address the above issues, we propose a novel generation framework based on dual-domain (vector-pixel) diffusion with cross-modality impulse signals from each other. First, in each diffusion step, the current representation extracted from the other domain is used as a condition variable to constrain the subsequent sampling operation, yielding shape-aware new parameterizations; second, independent supervision signals from both domains avoid the gradient error accumulation problem caused by cross-domain representation conversion. Extensive experimental results on popular benchmarks including font and icon datasets demonstrate the great advantages of our proposed framework in terms of generated shape quality.*

1. Introduction

Vector graphics are a representation of images using a sequence of drawing instructions. Unlike traditional raster graphics, vector graphics preserve their geometric properties and visual quality regardless of the resolution. Moreover, well-designed vector graphics, with their planar elements organized in a way that closely aligns with human cognitive logic for shapes, make editing more intuitive and convenient. As a result, vector graphics have gained significant popularity in the realms of creativity, art, design and the development of technologies capable of generating high-quality vector images has become a primary objective for designers.

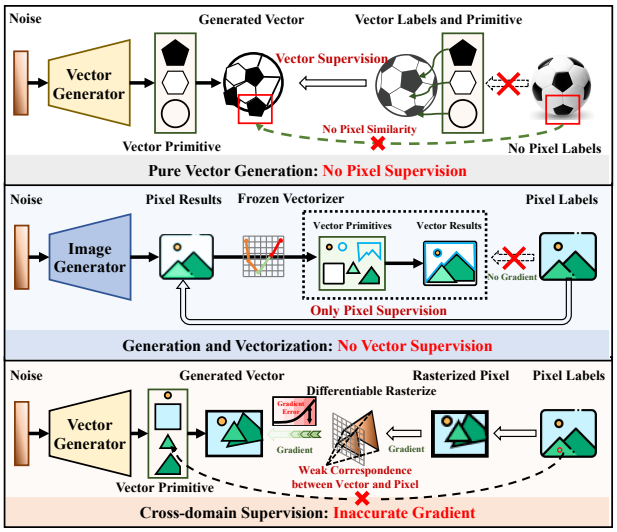

Figure 1. Current mainstream approaches each come with distinct challenges: 1. Pure Vector Generation: No pixel supervision. The similarity measurement between shape parameters (and their combination) and the paired pixel image appearance pattern is ignored. 2. Generation and Vectorization: No Vector Supervision. The quality of generated vector graphics relies heavily on the vectorizer. 3. Cross-domain Supervision: Inaccurate Gradient. Gradients estimated by differentiable rasterization are inaccurate, causing wrong error-correction signal.

However, current mainstream methods for generating vector graphics lack the encoding of image appearance associated with the original vector image, *i.e.*, offering no effective linkage between pixel pattern and the corresponding drawing primitives during learning. As a result, they

*Corresponding author: Bingbing Ni

lose the effective supervision signal from the strong correlation between the discrete vector parameter (drawing instruction) sequence and the target shape/structure of the corresponding pixelized image, which makes it difficult to generate vector graphics with high shape quality. Specifically, on the one hand, the generation process based on pure vector fields completely ignores the similarity measurement between shape instructions/parameters (and their combination) and the paired pixel image appearance patterns. Vector graphics are composed of independent instruction sequences, and the data describing vector graphics contain many heterogeneous attributes (such as instruction categories and parameters). This leads to discontinuity, non-smoothness, and non-uniformity of data distribution in the vector space from the perspective of shape and visual quality, which makes it difficult to guarantee the quality of shape without sufficient reference information guidance. For instance, Deepsvg [2] operates purely in the vector space and can only generate simple font data. The absence of supervised information from paired pixel images is a crucial factor resulting in serious shape quality degradation for more complex icon data. On the other hand, using a two-stage method that combines pixel-based generation techniques with a non-trainable pixel-to-vector translation module introduces the wrong error-correction signal caused by approximate gradients (as the vector-to-pixel process is essentially non-differentiable and the reverse process has one-to-many mappings). Since the vectorization process requires differentiable rasterization [11], this approach heavily relies on the approximate gradients calculated from the pixel domain, ignoring the sequential characteristics and combination properties of each vector primitive during the decoding process from vector to pixel. This not only compromises the authenticity and flexibility of the vector graphics but also hinders the interpretability of the generated instruction sequence for vectorization [3][20][13]. For example, Im2vec [20] generates vector graphics based on pixel image features and differentiable rasterization supervision signal, resulting in the unexpected issue of dense instruction stacking. Although SVG-VAE [13] does not employ differentiable rasterization, it combines the learning of pixel image distribution and vector graphic generation in a concatenated and staged training manner, leading to weak coupling of information between the two steps.

To address both issues, we propose a novel generative framework based on dual-domain (vector-pixel) diffusion with mutual impulse signals. Firstly, we construct a synchronized and parallel diffusion model for both vector and pixel domains. In each diffusion step, a shape-aware cross-attention module is introduced by extracting self-domain representations as well as cross-domain representations, to generate next-step samples from this cross-domain conditional distribution. This mutual impulse design strengthens the similarity measurement and builds immediate linkage/interaction between shape instructions/parameters and the corresponding pixel patterns, which helps to guide the vector sampling process toward a more appearance-compatible manner. Secondly, during the training of our dual-domain diffusion model, we utilize independent supervision signals in the vector and pixel domains. This approach avoids the reliance on differentiable rasterization, addressing issues such as gradient error accumulation and unrealistic/inflexible instruction sequences. Moreover, to achieve more visually reasonable/plausible vector graphics, we propose a differentiable geometric constraint by utilizing prior knowledge of vector graphics such as smoothness and closure within our training framework, which helps to further optimize the shape visual quality of the generated vector graphics with higher accuracy in detail representation and more naturally artistic essence. Extensive experimental results demonstrate the state-of-the-art (SOTA) performance achieved by our approach, in both quantitative and qualitative evaluations. In font benchmark datasets, our method generates diverse font samples with regularity and artistic appeal. Similarly, in complex icon datasets, our generated samples consider the correctness of sequential characteristics and combination properties of vector primitives, as well as the visual quality of shape.

2. Related works

Vector Graphics Generation. Vector graphics, unlike raster images, utilize well-defined geometric primitives (*e.g.*, curves, Bézier shapes) to describe visual concepts, facilitating vision representation in a compact and scalable parameter format. While generative models achieve remarkable results in the raster domain, their applications to vector graphics are not well explored. Most existing methods focus on advanced shape synthesis [6] or sketch generation [21] based on a simple dataset [7]. SVG-VAE [13] is a pioneering work to build sequential generative models on vector graphics, achieving compelling font generation results in a Variational Auto-encoder (VAE) framework. However, SVG-VAE fails to handle complex vector graphics with more than 10 commands. Deepsvg [2] further expands the scope of vector graphics generation by modeling the hierarchical structure of vector graphics with a transformer-based generative model. Both SVG-VAE and Deepsvg are limited to simple data with fixed patterns. Iconshop [26] sequentializes and tokenizes vector graphics paths into a uniquely decodable token sequence and achieves remarkable text-guided vector graphics icon generation quality with autoregressive transformers. However, Iconshop overfits the feature mapping between text and vector space, leading to non-smoothness in shape transition. Im2Vec [20] generates vector graphics without supervision in the vector domain, which uses raster images as inputs and

optimizes the generated vectors via the gradients estimated by differentiable rasterization [11]. Nevertheless, directly aligning vector graphics with the raster images ignores the natural expression and logic (temporal ordering) of vector graphics, causing intrinsic information loss.

Diffusion Models. While diffusion models make remarkable achievements in many domains such as natural language processing [12], shape generation [1], 3D shape modeling [14], and particularly image generation [17][9][5], their applications to vector graphics generation are not well explored. Sketchknitter [25] learns data distributions over the stroke-point locations and achieves compelling sketch generation with a diffusion model. However, it faces challenges in generating vector graphics with multiple shapes and filled properties. VectorFusion [10] distills [18] the pre-trained text-to-image diffusion model [22] for iterative generation of vector graphics. However, it relies heavily on the gradients estimated by differentiable rasterization [11] and is limited by the bias of pre-trained image diffusion.

3. Method

In this section, we introduce in detail our proposed vector-pixel dual-domain diffusion pipeline, as illustrated in Fig. 2. This pipeline possesses two parallel diffusion processes operating on the vectorized image representation domain as well as the corresponding/matched pixel image domain in a synchronized manner, ensuring complementary supervision signals from both domains. A cross-attention mechanism based conditional feature injection module is developed to guide the shape-vector diffusion sampling process towards a more appearance-oriented way, which plays a crucial role in establishing the dependency/interaction structure between the pixel pattern and the associated shape (and shape combinations) parameterization. In addition, a shape prior regularizer is dedicatedly proposed to ensure local shape smoothness and global shape closure. Details are given as follows.

3.1. Prerequisite: Vector Graphics Representation

Vector graphics represent images through a collection of closed shapes, denoted as $\mathbf{v} = \{s_1, ..., s_{n_\mathbf{v}}\}$. Each shape s_i is constructed using specific drawing instructions, such as lines, rectangles, circles and Bézier curves, where $s_i = \{c_{i,1}, ..., c_{i,n_{s_i}}\}$ (n_{s_i} signifies the number of instructions within a given shape, which varies across different shapes. i is the subscript number of the shape within a vector image $\mathbf{v}$).

These instructions comprise a shape type $t_{i,j}$ and corresponding geometric parameters $a_{i,j}$, formulated as $c_{i,j} = \{t_{i,j}, a_{i,j}\}$. j is the subscript number of the instruction within a shape. Drawing notations from Deepsvg's [2], we streamline vector graphics attributes and focus on four essential instruction types including 1) Move To ('m', encoded to $t_{i,j} = (-1,-1)\cdot\lambda$, $a_{i,j} = (x,y)$); 2) Line To ('l', encoded to $t_{i,j} = (-1,1)\cdot\lambda$, $a_{i,j} = (x,y)$); 3) Cubic Bezier ('c', encoded to $t_{i,j} = (1,-1)\cdot\lambda$, $a_{i,j} = (x_1,y_1,x_2,y_2,x,y)$); 4) Close Shape ('z', encoded to $t_{i,j} = (1,1)\cdot\lambda$, $a_{i,j} = \varnothing$). λ is the signal amplitude of the instruction type. (x,y) is ending point coordinate and $(x_1,y_1),(x_2,y_2)$ are control point coordinates. Each shape s_i begins with the 'm' command as $c_{i,1}$, progresses through 'l' or 'c' commands for drawing, and concludes with the 'z' command as $c_{i,n_{s_i}}$ to ensure the shape's closure. We fill the geometric parameters to the same dimension. Especially for the 'z' command, we fill (x,y) of $c_{i,1}$ as its ending point. By concatenating instructions of all shapes, we obtain the representation of the vector image $\mathbf{v}$ as $\mathbf{v} \in \mathbb{R}^{n\times c_\mathbf{v}}$.

3.2. Dual-Diffusion: Modelling and Pipeline

We could rasterize the vector image $\mathbf{v}$ to obtain the corresponding pixel image $\mathbf{p}$. $\mathbf{v}$ is a serialized vector variable with one-dimensional attribute arrangement (*i.e.* instructions), $\mathbf{v} \in \mathbb{R}^{n\times c_\mathbf{v}}$, and $\mathbf{p}$ is a grid-based matrix (or tensor) variable with two-dimensional spatial attribute arrangement (*i.e.* pixel values), $\mathbf{p} \in \mathbb{R}^{h\times w\times c_\mathbf{p}}$.

To jointly diffuse $\mathbf{v}$ and $\mathbf{p}$, we can concatenate them into a joint space $\mathbf{x} = \{\mathbf{v},\mathbf{p}\}$ with proper dimensionality reshaping. In this way, we can consider $p(\mathbf{x}) = p(\mathbf{v},\mathbf{p})$ as a joint probabilistic distribution, and conventional diffusion theory and computation could be readily applied to iteratively generate both vector pixel representation of the same image instance. However, $\mathbf{v}$ and $\mathbf{p}$ are strongly dependent since the pixel image $\mathbf{p}$ could be fully reconstructed by executing rendering instructions by $\mathbf{v}$. It is also worth mentioning that given pixel image $\mathbf{p}$, there might exist multiple feasible encodings of $\mathbf{v}$, *i.e.*, one-to-many mapping. To this end, conventional diffusion process/pipeline should be modified/upgraded to cope with our scenario, *i.e.*, joint sampling two inter-dependent and mutually constrained high dimensional variables.

In the meantime, the ultimate goal of building this dual-domain diffusion process is to encourage cross-domain supervision signal to be fully exploited to correctly guide the sampling process to generate appearance-compatible shape samples. Namely, we wish to well establish the dependency structure between the vector instruction (and their combined instruction sequence) and the corresponding pixel patterns in a bi-directional way (*i.e.*, in both inference and training), therefore error-correction signals from both domains could help to boost the model learning of each domain. To facilitate the above objective, our pipeline design is two-fold. First, during each diffusion sampling step, we inject guidance signal from the other domain as conditional variables, which helps to regulate/modulate the generated samples to convey more structural constraint information from the other knowledge side. For example, when generating

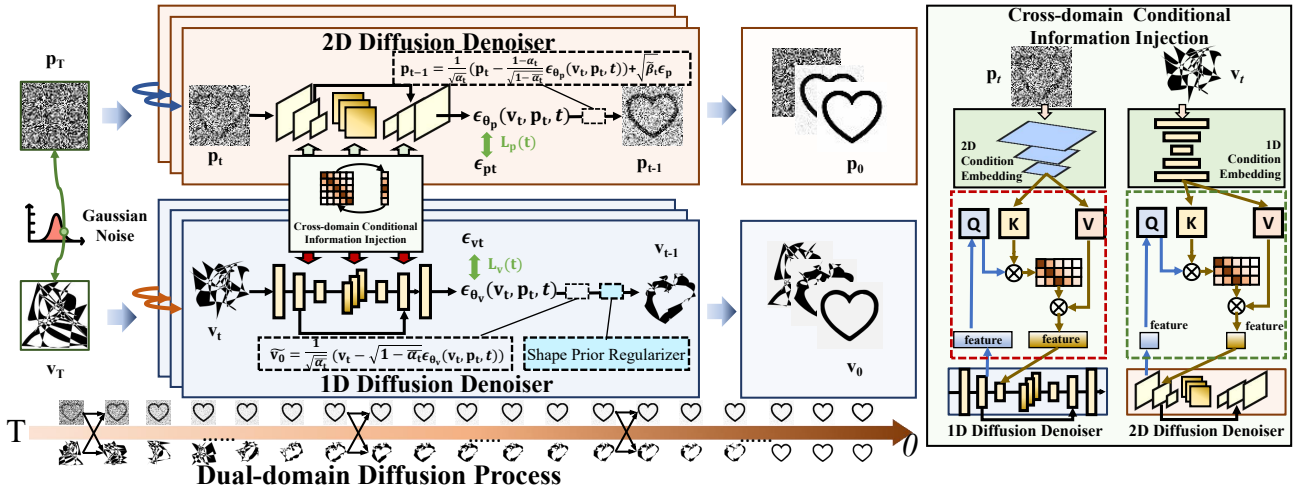

Figure 2. **Overview of our framework**. Our framework possesses two synchronous diffusion processes operating on the vector domain and the corresponding pixel domain, enabling mutual gradient propagation. Cross-domain Conditional Information Injection is developed to guide the shape-vector diffusion sampling process towards a more appearance-oriented way, which is crucial for establishing the mutual interaction between the pixels and the associated shape parameterization. Additionally, a shape prior regularizer is dedicatedly proposed to ensure local shape smoothness and global shape closure.

a vector sample, this module checks whether the shape of such sample well fits the pixel appearance pattern. Second, the diffusion flow of each domain is directly supervised by the respective ground truth. Therefore, incorrect gradients induced by approximate inverse rendering (from pixel to vector) could be properly avoided.

The probabilistic modeling of the proposed dual-domain diffusion process is as follows. Similar to the fundamental theory of diffusion models, the training of dual-diffusion consists of both forward and reverse processes. At time step t, the forward process takes a data sample $\mathbf{x}_0 = \{\mathbf{v}_0, \mathbf{p}_0\}$ (where $\mathbf{v}_0$ and $\mathbf{p}_0$ are corresponding data pairs), and obtains a noisy data sample $\mathbf{x}_t$ by fusing $\mathbf{x}_0$ with additive Gaussian noise. The noisy data sample satisfies the distribution $q(\mathbf{x}_t|\mathbf{x}_0)$ (see Eq. (1)). Here, $\mathbf{I_v}$ denotes the identity matrix corresponding to $\mathbf{v}$, $\mathbf{I_p}$ denotes the identity matrix corresponding to $\mathbf{p}$, and $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$, where $\bar{\alpha}_t$ is a noise schedule that gradually changes from 1 to 0.

$$\begin{aligned} &\mathbf{x}_t = \{\mathbf{v}_t, \mathbf{p}_t\} \\ &\mathbf{v}_t = \sqrt{\bar{\alpha}_t}\mathbf{v}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_{\mathbf{v}t}\ ;\ \boldsymbol{\epsilon}_{\mathbf{v}t} \sim \mathcal{N}(\mathbf{0}, \mathbf{I_v}) \\ &\mathbf{p}_t = \sqrt{\bar{\alpha}_t}\mathbf{p}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_{\mathbf{p}t}\ ;\ \boldsymbol{\epsilon}_{\mathbf{p}t} \sim \mathcal{N}(\mathbf{0}, \mathbf{I_p}) \\ &q(\mathbf{x}_t|\mathbf{x}_0) = q(\mathbf{v}_t|\mathbf{v}_0) \cdot q(\mathbf{p}_t|\mathbf{p}_0) \end{aligned} \tag{1}$$

The reverse process of dual-diffusion learns the denoising process from $\mathbf{x}_t$ at time t to $\mathbf{x}_{t-1}$ at time $t-1$, starting from a pure Gaussian noise $\mathbf{x}_T$ and progressively denoising to eventually obtain the generated sample $\mathbf{x}_0$. Specifically, we consider the reverse conditional probability $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$. Since $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ follows a Gaussian distribution [16], and $\mathbf{x}_t = \{\mathbf{v}_t, \mathbf{p}_t\}$ are independent in each dimension during the noise addition process, we can derive the mathematical representation in Eq. (2), where $\alpha_t = 1 - \beta_t$, $\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t$.

$$\begin{aligned} &q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = q(\mathbf{v}_{t-1}|\mathbf{v}_t, \mathbf{v}_0) \cdot q(\mathbf{p}_{t-1}|\mathbf{p}_t, \mathbf{p}_0) \\ &= \mathcal{N}(\mathbf{v}_{t-1}; \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{v}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_{\mathbf{v}t}\Big), \tilde{\beta}_t\mathbf{I_v}) \\ &\cdot \mathcal{N}(\mathbf{p}_{t-1}; \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{p}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_{\mathbf{p}t}\Big), \tilde{\beta}_t\mathbf{I_p}) \end{aligned} \tag{2}$$

We employ a neural network $\theta = \{\theta_{\mathbf{v}}, \theta_{\mathbf{p}}\}$ to approximate the conditional probability distribution during the reverse process, denoted as $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ in Eq. (3).

$$\begin{aligned} &p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = p_{\theta_{\mathbf{v}}}(\mathbf{v}_{t-1}|\mathbf{v}_t, \mathbf{p}_t) \cdot p_{\theta_{\mathbf{p}}}(\mathbf{v}_{t-1}|\mathbf{v}_t, \mathbf{p}_t) \\ &= \mathcal{N}(\mathbf{v}_{t-1}; \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{v}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_{\theta_{\mathbf{v}}}(\mathbf{v}_t, \mathbf{p}_t, t)\Big), \tilde{\beta}_t\mathbf{I_v}) \\ &\cdot \mathcal{N}(\mathbf{p}_{t-1}; \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{p}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_{\theta_{\mathbf{p}}}(\mathbf{v}_t, \mathbf{p}_t, t)\Big), \tilde{\beta}_t\mathbf{I_p}) \end{aligned} \tag{3}$$

Using the cross-entropy between the data distribution and the predicted distribution as the loss objective, and following the transformation of the variational lower bound as used by Sohl-Dickstein et al. [23] ($L_t = D_{\mathrm{KL}}(q(\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_t \mid \mathbf{x}_{t+1}))$), we can derive a similar training loss in Eq. (4)

$$\begin{aligned} L_{\mathbf{v}}(t) &= \|\boldsymbol{\epsilon}_{\mathbf{v}t} - \boldsymbol{\epsilon}_{\theta_{\mathbf{v}}}(\mathbf{v}_t, \mathbf{p}_t, t)\|^2 \\ L_{\mathbf{p}}(t) &= \|\boldsymbol{\epsilon}_{\mathbf{p}t} - \boldsymbol{\epsilon}_{\theta_{\mathbf{p}}}(\mathbf{v}_t, \mathbf{p}_t, t)\|^2 \\ L_t &= L_{\mathbf{v}}(t) + L_{\mathbf{p}}(t) \end{aligned} \tag{4}$$

During the sampling process, through Eq. (3), it is readily apparent that the prediction of $\mathbf{x}_{t-1} = \{\mathbf{v}_{t-1}, \mathbf{p}_{t-1}\}$

from $\mathbf{x}_t = \{\mathbf{v}_t, \mathbf{p}_t\}$ can be calculated using Eq. (5), where $\boldsymbol{\epsilon}_\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_\mathbf{v})$ and $\boldsymbol{\epsilon}_\mathbf{p} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_\mathbf{p})$.

$$\begin{aligned}\mathbf{v}_{t-1} &= \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{v}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_{\theta_\mathbf{v}}(\mathbf{v}_t, \mathbf{p}_t, t)\Big) + \sqrt{\tilde{\beta}_t}\boldsymbol{\epsilon}_\mathbf{v} \\ \mathbf{p}_{t-1} &= \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{p}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_{\theta_\mathbf{p}}(\mathbf{v}_t, \mathbf{p}_t, t)\Big) + \sqrt{\tilde{\beta}_t}\boldsymbol{\epsilon}_\mathbf{p}\end{aligned} \tag{5}$$

3.3. Cross-domain Conditional Information Injection via Cross-Attention

For the proposed dual-domain diffusion model, following Ho et al. [9], we respectively use a 1D U-Net and a 2D U-Net as the base diffusion architecture for the vector-domain denoiser $\boldsymbol{\epsilon}_{\theta_\mathbf{v}}$ and the pixel-domain denoiser $\boldsymbol{\epsilon}_{\theta_\mathbf{p}}$, as depicted in Fig. 2. Two data flows run synchronously on this architecture: a data flow purely operates on the vector representation domain including diffusion denoising for forward and supervision by the score of noisy data for backward, and similarly another data flow runs on the pixel domain.

It is straightforward and reasonable to encourage direct feature interaction at each resolution layer in the vector and pixel domains by Multilayer Perceptron after proper dimensionality reshaping. However, In the U-Net architecture, the effective information capacity of features is constrained. A substantial portion of this capacity is devoted to data distribution information, which limits the availability of shape and visual information critical for interaction.

To efficiently introduce cross-domain knowledge guidance/interaction, we embed a cross-attention module (named appearance-aware injection in the vector domain and shape-aware injection in the pixel domain) after the convolution operation at each resolution level of the U structure, aiming at injecting strong reference information from the other domain to regulate/modularize the output sample of the current step. Namely, we can rely on the appearance-aware injection scheme to form the conditional sampling probability $p_{\theta_\mathbf{v}}(\mathbf{v}_{t-1}|\mathbf{v}_t, \mathbf{p}_t)$ for sampling $\mathbf{v}_{t-1}$ at time step $t-1$, $\mathbf{v}_{p-1}$ vice versa.

More concretely, we construct 2D Conditional Embedding using the ResNet architecture, which extracts the 2D features from $\mathbf{p}_t$, flattens them to 1D, and passes them through fully connected layers to produce the output embedding. Appearance-aware injection receives convolutional features $f^i_\mathbf{v}$ at the current resolution level i and $E_{\mathbf{p}_t}$ obtained by $\mathbf{p}_t$ via 2D Conditional Embedding from pixel domain as the local conditional representation, outputting features with shape perception $\widehat{f^i_\mathbf{v}}$(see Eq. (6), where $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ are linear mappings projecting the representations into the same dimensional space).

$$\widehat{f^i_\mathbf{v}} = \text{softmax}\Big(\frac{\mathbf{Q}(f^i_\mathbf{v})\mathbf{K}(E_{\mathbf{p}_t})^\top}{\sqrt{d}}\Big)\cdot\mathbf{V}(E_{\mathbf{p}_t}) \tag{6}$$

On the contrary, we construct 1D Conditional Embedding through 1D U-Net architecture with linear attention in each resolution layer, which outputs the 1D feature of $\mathbf{v}_t$ as embedding. Shape-aware injection receives convolutional features $f^i_\mathbf{p}$ at the current resolution level i and $E_{\mathbf{v}_t}$ obtained by $\mathbf{v}_t$ via 1D Conditional Embedding from vector domain as the local conditional representation, outputting features with shape perception $\widehat{f^i_\mathbf{p}}$ (see Eq. (7)). Both $\widehat{f^i_\mathbf{v}}$ and $\widehat{f^i_\mathbf{p}}$ are then fed to their respective subsequent resolution levels.

$$\widehat{f^i_\mathbf{p}} = \text{softmax}\Big(\frac{\mathbf{Q}(f^i_\mathbf{p})\mathbf{K}(E_{\mathbf{v}_t})^\top}{\sqrt{d}}\Big)\cdot\mathbf{V}(E_{\mathbf{v}_t}) \tag{7}$$

Moreover, supervision signals are available for both data flows. By developing a mutual information injection structure and independently supervising dual pathways, the vector generation model gains an enhanced understanding of the dependency relationship between pixel and vector representations. This allows for the generation of vector instructions that more accurately reflect/interpret user's design specifications.

3.4. Shape Prior Regularizer and Training Objectives

For each closed shape s_i in a vector graphic $\mathbf{v}$, we consider that it should satisfy the following two priors: 1) local shape smoothness, closely related to the quality of shape local details; and 2) global shape closure, essential for maintaining the overall integrity and completeness of the shape.

Local shape smoothness. s_i is made up of several drawing instructions, ($s_i = \{c_{i,1}, ..., c_{i,n_{s_i}}\}$). To encourage local smoothness, we constrain the tangents of consecutive curves as $\vec{l}^e_{i,j}$ and $\vec{l}^s_{i,j+1}$ to be as close as possible. To this end, we calculate the angle distance between $\vec{l}^e_{i,j}$ and $\vec{l}^s_{i,j+1}$ and introduce binary weights $\omega^s_{i,j}$, such that $\omega^s_{i,j} = 1$ when angle value less than the angle threshold α_0; otherwise it is 0. The local smoothness of $\mathbf{v}$ could be expressed by the loss function $L_s(\mathbf{v})$ as:

$$L_s(\mathbf{v}) = \frac{\sum_{i=1}^{n_\mathbf{v}}\sum_{j=1}^{n_{s_i}-1}\omega^s_{i,j}\left(1 - \frac{\vec{l}^e_{i,j}\cdot\vec{l}^s_{i,j+1}}{\|\vec{l}^e_{i,j}\|\|\vec{l}^s_{i,j+1}\|}\right)^2}{\sum_{i=1}^{n_\mathbf{v}}\sum_{j=1}^{n_{s_i}-1}\omega^s_{i,j}} \tag{8}$$

Global shape closure. Abuse of the 'z' instruction for closure without constraint on the starting and ending points of s_i can lead to discontinuity. We therefore explicitly constrain the starting and ending points of a shape to be as close as possible. To accommodate the non-differentiable nature of recognizing the 'z' instruction, we use an approximate weight $\frac{1}{1+e^{k_c\|t_{i,j}-(\lambda,\lambda)\|}}$ to assign high weight to the 'z' instruction and extremely low weight elsewhere. We calculate the closure loss function $L_c(\mathbf{v})$ for $\mathbf{v}$ as Eq. (9) where $p^e_{i,j}$

Figure 3. **Generation results on Font and Icon datasets.** Our method generates high-quality vector graphics with well-organized details.

represents the ending point of $c_{i,j}$.

$$L_c(\mathbf{v}) = \frac{\sum_{i=1}^{n_\mathbf{v}} \sum_{j=2}^{n_{s_i}} \frac{1}{1+e^{k_c \| t_{i,j} - (\lambda,\lambda) \|}} \| p^e_{i,j-1} - p^e_{i,j} \|^2}{\sum_{i=1}^{n_\mathbf{v}} \sum_{j=2}^{n_{s_i}} \frac{1}{1+e^{k_c \| t_{i,j} - (\lambda,\lambda) \|}}} \tag{9}$$

Based on the above constraints, we build an additional neural network f_{θ_r} named Shape Prior Regularizer (SPR) after the denoiser to further regularize/improve the sample exploring the above prior knowledge of vector graphics. At time t, after obtaining the predicted noise $\boldsymbol{\epsilon}_{\theta_\mathbf{v}}(\mathbf{v}_t, \mathbf{p}_t, t)$, we predict the original data in a parametric manner as $\widetilde{\mathbf{v}}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{v}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_{\theta_\mathbf{v}}(\mathbf{v}_t, \mathbf{p}_t, t))$, and then obtain $\widetilde{\mathbf{v}}_0^{'} = f_{\theta_r}(\widetilde{\mathbf{v}}_0)$. We require $L_t^i = \|\widetilde{\mathbf{v}}_0^{'} - \widetilde{\mathbf{v}}_0\|^2$ to maintain the integrity of the overall information. Therefore, for a diffusion step t, the total loss function L_t^r is as:

$$L_t^r = \omega_i \cdot L_t^i + \omega_s \cdot L_s(\mathbf{v}_t) + \omega_c \cdot L_c(\mathbf{v}_t) \tag{10}$$

During training, the gradient of L_t^r is back-propagated only for $t < t_r$. During sampling, when $t < t_r$, we pass the predicted noise through parametric calculation and the regular network to obtain $\widetilde{\mathbf{v}}_0^{'}$, which is then transmitted to the next denoising step. This approach integrates prior knowledge into the final generated sample $\mathbf{v}_0$.

Overall training objectives. We introduce ω_p to balance the weights of both domain losses. We also introduce ω_{rt} to control the backpropagation of L_t^r only for $t < t_r$: $\omega_{rt} = 1$ when $t < t_r$, and $\omega_{rt} = 0$ when $t \geq t_r$. The overall loss function L is given by:

$$L = \mathbb{E}_{t\sim[1,T],\mathbf{x}_0,\boldsymbol{\epsilon}_{\mathbf{v}t},\boldsymbol{\epsilon}_{\mathbf{p}t}}\Big[L_{\mathbf{v}t} + \omega_p L_{\mathbf{p}t} + \omega_{rt} L_t^r\Big] \tag{11}$$

4. Experiment

4.1. Experimental Setup

Dataset. We employ two datasets in our study. The first is the Font data benchmark from [2]. The second is the Icon dataset, which is specifically constructed for our study based on FIGR-8-SVG dataset [4]. To maintain a high sample density across various instruction sequence lengths, we cap the maximum instruction sequence length at 96 and filter out categories with insufficient samples. Consequently, we build Icon dataset with 82 classes containing a total of 218K samples. Our training, generation, and extension experiments are conducted on these two datasets.

Evaluation Metrics. The evaluation metrics mainly consider two aspects: 1) Similarity between data distribution of generated samples and the original dataset. Since vector samples need to meet visual design requirements, we employ CLIP [19] to extract features of images rasterized by vector graphics for evaluation. We calculate the Fréchet Inception Distance (FID) [8] between the features of generated samples and the entire dataset. 2) Visual quality of the generated samples. We identify the top k instances in the original dataset that are most similar to a generated sample, based on the Euclidean Distance of the CLIP feature. Furthermore, we compute the average Euclidean distance of the CLIP feature between the generated sample and these k instances from the original dataset. This metric is referred to as the Visualization Distance (VD). Since CLIP features comprise both visual and shape information, Visualization

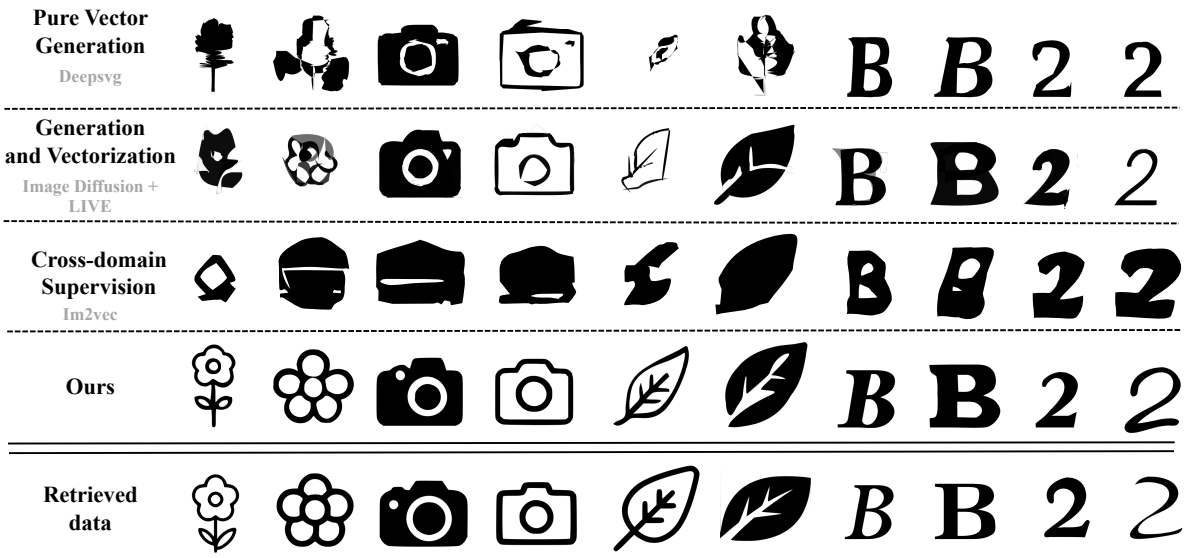

Figure 4. **Comparisons with other methods.** For fair comparisons, we utilize the same retrieved data to sample from different methods. Deepsvg [2] provides irregular shapes, especially on more detailed and complex samples. Image Diffusion + LIVE [15] shows a marginal improvement in visual quality. The results generated by Im2vec [20] have a much larger error compared with other methods. Our approach achieves excellent results in both artistic appeal and the structural integrity of the shapes.

Distance (VD) can measure the visual quality of the generated samples based on the k retrieved references.

Implementation Details. In our model configuration, the Shape Prior Regularizer incorporates the Transformer model [24]. The weights of the loss functions are set as $\omega_{\mathbf{p}} = 0.2$, $\omega_i = 1.0$, $\omega_s = 0.05$, and $\omega_c = 2.0$, respectively. Regarding data representation, the size of the rasterized image $\mathbf{p}$ is fixed at 64×64 pixels, and the amplitude of vector graphic instructions λ is fixed at 1.0. The total steps of the diffusion model $T = 1000$, the angle threshold $\alpha_0 = 5°$, the parameter k_c (see Eq. (9)) is fixed at 5.0, and t_r (see Eq. (10)) is set to 32. For the training process, we utilize a batch size of 120 and conduct the training for $700k$ steps. For the testing process, k is set to 5 for Visualization Distance calculation.

4.2. Visual and Quantitative Comparison

For visual comparison, we generate 128 samples for each category in both datasets. For the Font dataset, 7936 samples are generated over 62 categories; for the Icon dataset, 10496 samples are generated over 82 categories. We show our generated vector graphics in Fig. 3. Our results demonstrate that the samples generated by our method exhibit a high level of semantic recognizability, coupled with a natural and artistic aesthetic. Furthermore, these samples are characterized by their clear and well-organized details. (For additional information, please refer to the supplementary materials.) We randomly select vector images in the original dataset and retrieve them by Euclidean distance of the CLIP feature within the samples generated by different methods. In Fig. 4, we present the most similar generated sample of each method during retrieval. The samples of Pure Vector Generation (Deepsvg [2]) lack regularity, particularly in more detailed and complex samples such as the flowers and leaves illustrated in Fig. 4). The two-stage method (Image Diffusion + LIVE [15]) shows a marginal improvement in visual quality, but mutual occlusion occurs among the primitives of the flower shown in Fig. 4. Cross-domain Supervision (Im2vec [20]) even fails to reasonably generate the proper number of primitives. The generated results have a much larger error compared with other methods. Our approach demonstrates excellent performance in both artistic appeal and the structural integrity of the shapes.

Method	FID↓		VD↓	
	Icon	Font	Icon	Font
Deepsvg [2]	27.42	8.66	0.0482	0.0190
Two-stage [15]	15.90	11.7	0.0195	0.0120
Im2Vec [20]	43.69	28.34	0.0817	0.0643
Ours	**6.09**	**6.66**	**0.0119**	**0.0066**

Table 1. **Quantitative comparisons.** We report the FID and VD results on Icon and Font datasets. "Two-stage" denotes the method using "Image Diffusion + LIVE".

On different datasets, we calculate evaluation metrics for the generated samples by category and then compute the average evaluation metrics. We present the average metrics of each method on Font and Icon datasets in Tab. 1. The results indicate that our method achieves the best performance in both data distribution fitting and visualization quality. In contrast, Deepsvg struggles to control visual in-

formation, resulting in poor quality, especially on complex Icon dataset. Due to the gradient errors and instability of differentiable rasterization, Im2vec and Image Diffusion + LIVE tend to be underperformed.

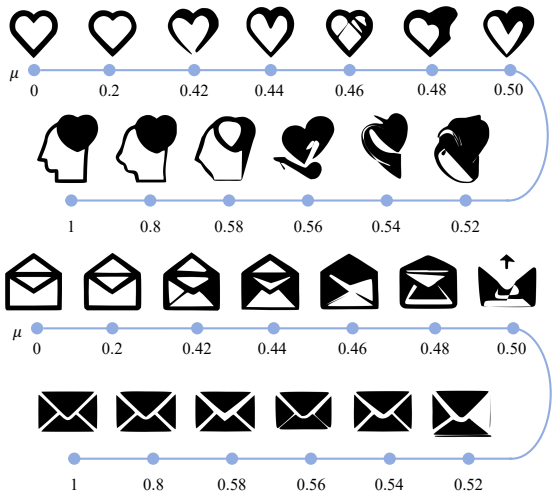

Figure 5. Visualization of interpolation results. Our method achieves smooth shape transition.

4.3. Interpolation Results

To verify the smoothness and continuity in the noisy data space for our method, as well as the ability to exhibit transitions at the shape level during the iterative denoising process, we conduct the following interpolation experiments. Following DDPM [9], we randomly encode two unified variables $\mathbf{x}_0 = \{\mathbf{v}_0, \mathbf{p}_0\}$ and $\mathbf{x}_0^{'} = \{\mathbf{v}_0^{'}, \mathbf{p}_0^{'}\}$ into the noisy data space at diffusion step t: $\mathbf{x}_t = \{\mathbf{v}_t, \mathbf{p}_t\}$ and $\mathbf{x}_t^{'} = \{\mathbf{v}_t^{'}, \mathbf{p}_t^{'}\}$. Then, through the denoising process, $\bar{\mathbf{x}}_t = (1-\mu)\mathbf{x}_0 + \mu\mathbf{x}_0^{'}$ is decoded back to the original space yielding $\bar{\mathbf{x}}_0 = \{\bar{\mathbf{v}}_0, \bar{\mathbf{p}}_0\}$, with $\bar{\mathbf{v}}_0$ as the final interpolated result. We consider the instruction type encoding $(\pm\lambda, \pm\lambda)$. When two instruction type encodings are blended with weights μ_1 and μ_2 ($\mu_1 > \mu_2$), the result semantics remain consistent with the instruction type corresponding to μ_1. An equilibrium probabilistic space for instruction type is achieved only at $\mu = 0.5$. Therefore, during interpolation, we conduct dense interpolation around $\mu = 0.5$, and sparser at other values, to accurately capture nuances in the interpolation process with $t = 400$. The results for two groups under Icon data, as shown in Fig. 5, demonstrate that our method achieves a smooth transition while retaining good visual quality.

4.4. Ablation Study

In our ablation studies, we assess the impact of two key components in our methodology: Cross-domain Conditional Information Injection via Cross-Attention and Shape Prior Regularizer. Our method without Cross-domain Conditional Information Injection equals to pure vector diffusion. The results are shown in Tab. 2. It reveals that Cross-domain Conditional Information Injection via Cross-Attention significantly enhances the overall quality of the generated distribution. Shape Prior Regularizer contributes to quality improvement, particularly in finer details. Visual comparisons in Fig. 6 illustrate that the absence of Cross-domain Conditional Information Injection leads to less effective information transfer, resulting in poor visual effects. Compared to scenarios without Shape Prior Regularizer, our complete method demonstrates more precise closure of shapes and smoother transitions at the junctions of drawing curves.

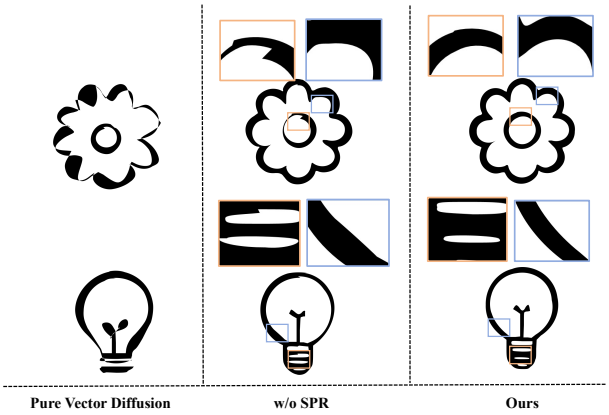

Figure 6. Visualization of generated instances. We utilize the same retrieved data to sample from different methods.

Method	Pure Vector Diffusion	w/o SPR	Ours
FID↓	7.70	6.37	**6.09**
VD↓	0.0214	0.0133	**0.0119**

Table 2. We investigate the crucial designs of our method on Icon dataset. Pure Vector Diffusion equals to our method without Cross-domain Conditional Information Injection.

5. Conclusion

In this work, we propose a novel generation framework for vector graphics based on dual-domain (vector-pixel) diffusion with mutual impulse. Experimental results on various datasets and tasks demonstrate that our framework generates high visual quality vector graphics with natural artistic beauty and well-organized details.

6. Acknowledgment

This work was supported by National Science Foundation of China (U20B2072, 61976137). This work was also partially supported by Grant YG2021ZD18 from Shanghai Jiaotong University Medical Engineering Cross Research. This work is partially supported by STCSM 22DZ2229005.

References

[1] Ruojin Cai, Guandao Yang, Hadar Averbuch-Elor, Zekun Hao, Serge Belongie, Noah Snavely, and Bharath Hariharan. Learning gradient fields for shape generation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pages 364–381. Springer, 2020. 3

[2] 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, 3, 6, 7

[3] Ye Chen, Bingbing Ni, Xuanhong Chen, and Zhangli Hu. Editable image geometric abstraction via neural primitive assembly. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 23514–23523, 2023. 2

[4] Louis Clouâtre and Marc Demers. Figr: Few-shot image generation with reptile, 2019. 6

[5] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021. 3

[6] Kevin Ellis, Daniel Ritchie, Armando Solar-Lezama, and Josh Tenenbaum. Learning to infer graphics programs from hand-drawn images. *Advances in neural information processing systems*, 31, 2018. 2

[7] David Ha and Douglas Eck. A neural representation of sketch drawings. *arXiv preprint arXiv:1704.03477*, 2017. 2

[8] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017. 6

[9] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 3, 5, 8

[10] 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. 3

[11] Tzu-Mao Li, Michal Lukáč, Michaël Gharbi, and Jonathan Ragan-Kelley. Differentiable vector graphics rasterization for editing and learning. *ACM Transactions on Graphics (TOG)*, 39(6):1–15, 2020. 2, 3

[12] Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. Diffusion-lm improves controllable text generation. *Advances in Neural Information Processing Systems*, 35:4328–4343, 2022. 3

[13] 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

[14] Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2837–2845, 2021. 3

[15] Xu Ma, Yuqian Zhou, Xingqian Xu, Bin Sun, Valerii Filev, Nikita Orlov, Yun Fu, and Humphrey Shi. Towards layer-wise image vectorization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16314–16323, 2022. 7

[16] Jerzy Neyman. *Proceedings of the Berkeley Symposium on Mathematical Statistics and Probability*. Univ of California Press, 1949. 4

[17] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021. 3

[18] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022. 3

[19] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. 6

[20] Pradyumna Reddy, Michael Gharbi, Michal Lukac, and Niloy J Mitra. Im2vec: Synthesizing vector graphics without vector supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7342–7351, 2021. 2, 7

[21] Leo Sampaio Ferraz Ribeiro, Tu Bui, John Collomosse, and Moacir Ponti. Sketchformer: Transformer-based representation for sketched structure. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14153–14162, 2020. 2

[22] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022. 3

[23] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015. 4

[24] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 7

[25] Qiang Wang, Haoge Deng, Yonggang Qi, Da Li, and Yi-Zhe Song. Sketchknitter: Vectorized sketch generation with diffusion models. In *The Eleventh International Conference on Learning Representations*, 2022. 3

[26] Ronghuan Wu, Wanchao Su, Kede Ma, and Jing Liao. Iconshop: Text-based vector icon synthesis with autoregressive transformers. *arXiv preprint arXiv:2304.14400*, 2023. 2