VectorFusion: Text-to-SVG by Abstracting Pixel-Based Diffusion Models

Ajay Jain* Amber Xie* Pieter Abbeel
UC Berkeley {ajayj,amberxie,pabbeel}@berkeley.edu

Figure 1. Text-to-SVG with VectorFusion. When (a) raster graphics sampled from Stable Diffusion are (b) auto-traced, they lose details that are hard to represent within the constraints of the abstraction. (c-d) VectorFusion improves fidelity and consistency with the caption by directly optimizing paths with a distillation-based diffusion loss. Find videos and more results at https://ajayj.com/vectorfusion.

Abstract

Diffusion models have shown impressive results in text-to-image synthesis. Using massive datasets of captioned images, diffusion models learn to generate raster images of highly diverse objects and scenes. However, designers frequently use vector representations of images like Scalable Vector Graphics (SVGs) for digital icons or art. Vector graphics can be scaled to any size, and are compact. We show that a text-conditioned diffusion model trained on pixel representations of images can be used to generate SVG-exportable vector graphics. We do so without access to large datasets of captioned SVGs. By optimizing a differentiable vector graphics rasterizer, our method, VectorFusion, distills abstract semantic knowledge out of a pretrained diffusion model. Inspired by recent text-to-3D work, we learn an SVG consistent with a caption using Score Distillation Sampling. To accelerate generation and improve fidelity, VectorFusion also initializes from an image sample. Experiments show greater quality than prior work, and demonstrate a range of styles including pixel art and sketches.

*Equal contribution

1. Introduction

Graphic designers and artists often express concepts in an abstract manner, such as composing a few shapes and lines into a pattern that evokes the essence of a scene. Scalable Vector Graphics (SVGs) provide a declarative format for expressing visual concepts as a collection of primitives. Primitives include Bézier curves, polygons, circles, lines and background colors. SVGs are the defacto format for exporting graphic designs since they can be rendered at arbitrarily high resolution on user devices, yet are stored and transmitted with a compact size, often only tens of kilobytes. Still, designing vector graphics is difficult, requiring knowledge of professional design tools.

Recently, large captioned datasets and breakthroughs in diffusion models have led to systems capable of generating diverse images from text including DALL-E 2 [28], Imagen [33] and Latent Diffusion [31]. However, the vast majority of images available in web-scale datasets are rasterized, expressed at a finite resolution with no decomposition into primitive parts nor layers. For this reason, existing diffusion models can only generate raster images. In theory,

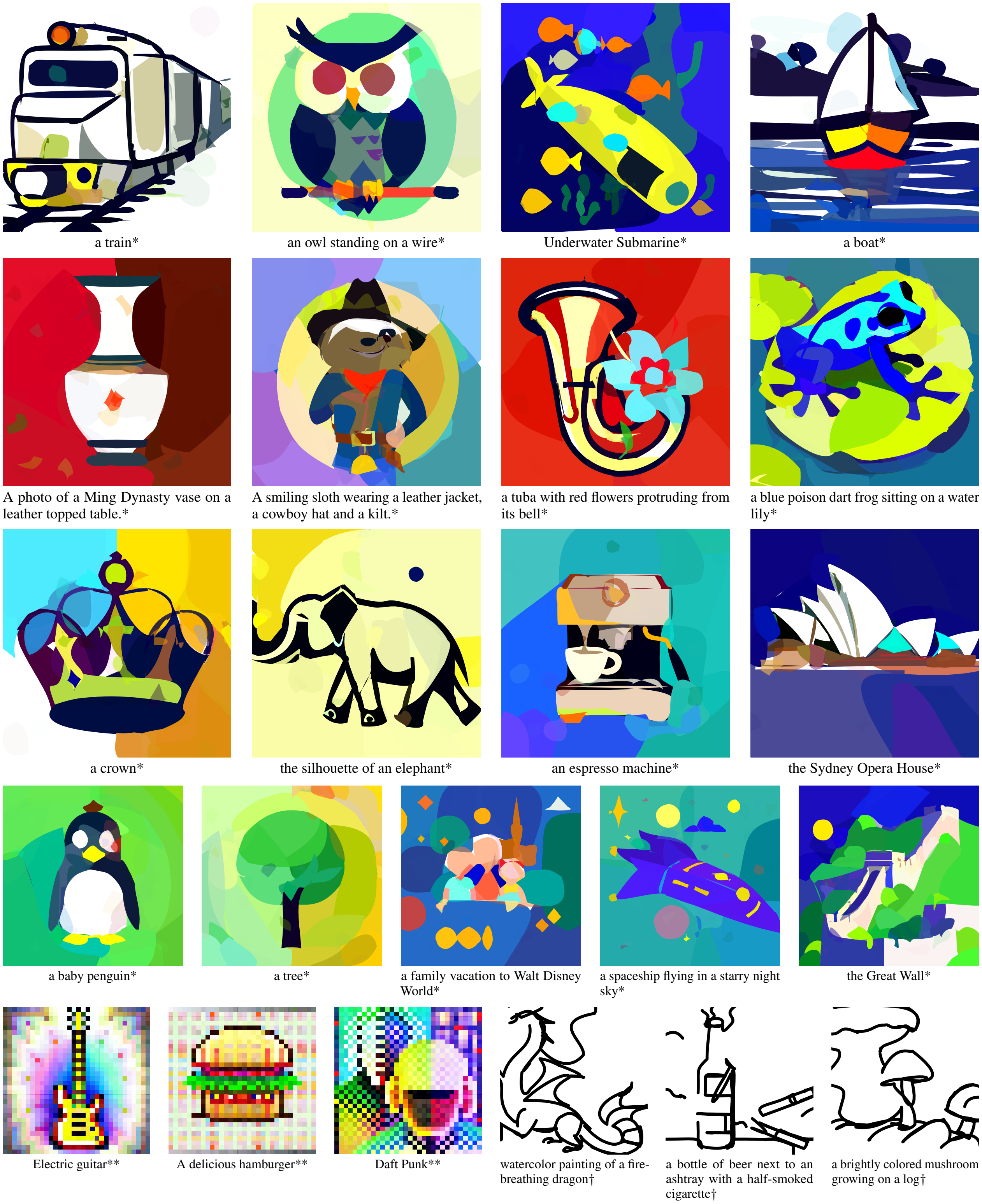

Figure 2. Given a caption, VectorFusion generates abstract vector graphics in an SVG format. We use a pre-trained diffusion model trained only on rasterized images to guide a differentiable vector renderer. VectorFusion supports diverse objects and styles. To select a style such as flat polygonal vector icons, abstract line drawings or pixel art, we constrain the vector representation to subset of possible primitive shapes and use different prompt modifiers to encourage an appropriate style: * ...minimal flat 2d vector icon. lineal color. on a white background. trending on artstation, ** ...pixel art. trending on artstation, †...minimal 2d line drawing. trending on artstation. Please see videos of the optimization process on our project webpage.

diffusion models could be trained to directly model SVGs, but would need specialized architectures for variable-length hierarchical sequences, and significant data collection work.

How can we use diffusion models pretrained on pixels to generate high-quality vector graphics? In this work, we provide a method for generating high quality abstract vector graphics from text captions, shown in Fig. 1.

We start by evaluating a two phase text-to-image and image-to-vector baseline: generating a raster image with a pretrained diffusion model, then vectorizing it. Traditionally, designers manually convert simple rasterized images into a vector format by tracing shapes. Some ML-based tools [19] can automatically approximate a raster image with an SVG. Unfortunately, we find that text-to-image diffusion models frequently produce complex images that are hard to represent with simple vectors, or are incoherent with the caption (Fig 1, Stable Diffusion + LIVE).

To improve quality of the SVG and coherence with the caption, we incorporate the pretrained text-to-image diffusion model in an optimization loop. Our approach, VectorFusion, combines a differentiable vector graphics renderer [16] and a recently proposed *score distillation sampling* (SDS) loss [26] to iteratively refine shape parameters. Intuitively, score distillation converts diffusion sampling into an optimization problem that allows the image to be represented by an arbitrary differentiable function. In our case, the differentiable function is the forward rasterization process, and the diffusion model provides a signal for improving the raster. To adapt SDS to text-to-SVG synthesis, we make the following contributions:

- We extend score distillation sampling to open source latent space diffusion models like Stable Diffusion,
- improve efficiency and quality by initializing near a raster image sample,
- propose SVG-specific regularization including path reinitialization,
- and evaluate different sets of shape primitives and their impact on style.

In experiments, VectorFusion generates iconography, pixel art and line drawings from diverse captions. VectorFusion also achieves greater quality than CLIP-based approaches that transfer a discriminative vision-language representation.

2. Related Work

A few works have used pretrained vision-language models to guide vector graphic generation. VectorAscent [11] and CLIPDraw [4] optimize CLIP's image-text similarity metric [27] to generate vector graphics from text prompts, with a procedure similar to DeepDream [23] and CLIP feature visualization [5]. StyleCLIPDraw [35] extends CLIPDraw to condition on images with an auxiliary style loss with a pretrained VGG16 [36] model. Arnheim [3] parameterizes SVG paths with a neural network, and CLIP-CLOP [22] uses an evolutionary approach to create image collages. Though we also use pretrained vision-language models, we use a generative model rather than a discriminative model.

Recent work has shown the success of text-to-image generation. DALL-E 2 [28] learns an image diffusion model conditioned on CLIP's text embeddings. Our work uses Stable Diffusion [31] (SD), a text-to-image latent diffusion model. While these models produce high-fidelity images, they cannot be directly transformed into vector graphics.

A number of works generate vector graphics from input images. We extend the work of Layer-wise Image Vectorization (LIVE) [19], which iteratively optimizes closed Bézier paths with a differentiable rasterizer, DiffVG [16].

We also take inspiration from inverse graphics with diffusion models. Diffusion models have been used in zero-shot for image-to-image tasks like inpainting [18]. DDPM-PnP [6] uses diffusion models as priors for conditional image generation, segmentation, and more. DreamFusion [26] uses 2D diffusion as an image prior for text-to-3D synthesis with a more efficient loss than DDPM-PnP, discussed in Section 3.3. Following [26], we use diffusion models as transferable priors for vector graphics. Concurrent work [20] also adapts the SDS loss for latent-space diffusion models.

3. Background

3.1. Vector representation and rendering pipeline

Vector graphics are composed of primitives. For our work, we use paths of segments delineated by control points. We configure the control point positions, shape fill color, stroke width and stroke color. Most of our experiments use closed Bézier curves. Different artistic styles are accomplished with other primitives, such as square shapes for pixel-art synthesis and unclosed Bézier curves for line art.

To render to pixel-based formats, we rasterize the primitives. While many primitives would be needed to express a realistic photogaph, even a few can be combined into recognizable, visually pleasing objects. We use DiffVG [16], a differentiable rasterizer that can compute the gradient of the rendered image with respect to the parameters of the SVG paths. Many works, such as LIVE [19], use DiffVG to vectorize images, though such transformations are lossy.

3.2. Diffusion models

Diffusion models are a flexible class of likelihood-based generative models that learn a distribution by denoising. A diffusion model generates data by learning to gradually map samples from a known prior like a Gaussian toward the data distribution. During training, a diffusion model optimizes a variational bound on the likelihood of real data samples [37], similar to a variational autoencoder [15]. This bound reduces to a weighted mixture of denoising objectives [9]:

$$\mathcal{L}_{\text{DDPM}}(\phi, \mathbf{x}) = \mathbb{E}_{t,\epsilon}\left[w(t)\|\epsilon_\phi(\alpha_t \mathbf{x} + \sigma_t \epsilon) - \epsilon\|_2^2\right] \quad (1)$$

where $\mathbf{x}$ is a real data sample and $t \in \{1, 2, \ldots T\}$ is a uniformly sampled timestep scalar that indexes noise schedules α_t, σ_t [14]. ϵ is noise of the same dimension as the image sampled from the known Gaussian prior. Noise is added by interpolation to preserve variance. ϵ_ϕ is a learned denoising autoencoder that predicts the noise content of its input. For images, ϵ_ϕ is commonly a U-Net [9, 32], and the weighting function $w(t) = 1$ [9]. Denoising diffusion models can be trained to predict any linear combination of $\mathbf{x}$ and ϵ, such as the clean, denoised image $\mathbf{x}$, though an ϵ parameterization is simple and stable.

At test time, a sampler starts with a draw from the prior $\mathbf{x}_T \sim \mathcal{N}(0, 1)$, then iteratively applies the denoiser to update the sample while decaying the noise level t to 0. For example, DDIM [38] samples with the update:

$$\begin{aligned} \hat{\mathbf{x}} &= (\mathbf{x}_t - \sigma_t \epsilon_\phi(\mathbf{x}_t))/\alpha_t, && \text{Predict clean image} \\ \mathbf{x}_{t-1} &= \alpha_{t-1}\hat{\mathbf{x}} + \sigma_{t-1}\epsilon_\phi(\mathbf{x}_t) && \text{Add back noise} \end{aligned} \tag{2}$$

For text-to-image generation, the U-Net is conditioned on the caption y, $\epsilon_\phi(\mathbf{x}, y)$, usually via cross-attention layers and pooling of the features of a language model [24]. However, conditional diffusion models can produce results incoherent with the caption since datasets are weakly labeled and likelihood-based models try to explain all possible images. To increase the usage of a label or caption, classifier-free guidance [10] superconditions the model by scaling up conditional model outputs and guiding away from a generic unconditional prior that drops y:

$$\hat{\epsilon}_\phi(\mathbf{x}, y) = (1 + \omega) * \epsilon_\phi(\mathbf{x}, y) - \omega * \epsilon_\phi(\mathbf{x}) \tag{3}$$

CFG significantly improves coherence with a caption at the cost of an additional unconditional forward pass per step.

High resolution image synthesis is expensive. Latent diffusion models [31] train on a reduced spatial resolution by compressing 512×512 images into a relatively compact 64×64, 4-channel latent space with a VQGAN-like autoencoder (E, D) [2]. The diffusion model ϵ_ϕ is trained to model the latent space, and the decoder D maps back to a high resolution raster image. We use Stable Diffusion, a popular open-source text-to-image model based on latent diffusion.

3.3. Score distillation sampling

Diffusion models can be trained on arbitrary signals, but it is easier to train them in a space where data is abundant. Standard diffusion samplers like (2) operate in the same space that the diffusion model was trained. While samplers can be modified to solve many image-to-image tasks in zero-shot such as colorization and inpainting [37, 39], until recently, pretrained image diffusion models could only generate rasterized images.

In contrast, image encoders like VGG16 trained on ImageNet and CLIP (Contrastive Language–Image Pre-training) [27] have been transferred to many modalities like mesh texture generation [23], 3D neural fields [12, 13], and vector graphics [4, 11]. Even though encoders are not generative, they can generate data with test time optimization: a loss function in the encoder's feature space is backpropagated to a learned image or function outputting images.

DreamFusion [26] proposed an approach to use a pretrained pixel-space text-to-image diffusion model as a loss function. Their proposed Score Distillation Sampling (SDS) loss provides a way to assess the similarity between an image and a caption:

$$\mathcal{L}_{\text{SDS}} = \mathbb{E}_{t,\epsilon}\left[\sigma_t/\alpha_t w(t) \mathrm{KL}(q(\mathbf{x}_t | g(\theta); y, t) \| p_\phi(\mathbf{x}_t; y, t))\right].$$

p_ϕ is the distribution learned by the frozen, pretrained diffusion model. q is a unimodal Gaussian distribution centered at a learned mean image $g(\theta)$. In this manner, SDS turned sampling into an optimization problem: an image or a differentiable image parameterization (DIP) [23] can be optimized with $\mathcal{L}_{\text{SDS}}$ to bring it toward the conditional distribution of the teacher. This is inspired by probability density distillation [41]. Critically, SDS only needs access to a pixel-space prior p_ϕ, parameterized with the denoising autoencoder $\hat{\epsilon}_\phi$. It does not require access to a prior over the parameter space θ. DreamFusion [26] used SDS with the Imagen pixel space diffusion model to learn the parameters of a 3D Neural Radiance Field [21]. In practice, SDS gives access to loss gradients, not a scalar loss:

$$\nabla_\theta \mathcal{L}_{\text{SDS}} = \mathbb{E}_{t,\epsilon}\left[w(t)\left(\hat{\epsilon}_\phi(\mathbf{x}_t; y, t) - \epsilon\right)\frac{\partial \mathbf{x}}{\partial \theta}\right] \tag{4}$$

4. Method: VectorFusion

In this section, we outline two methods for generating abstract vector representations from pretrained text-to-image diffusion models, including our full VectorFusion approach.

4.1. A baseline: text-to-image-to-vector

We start by developing a two stage pipeline: sampling an image from Stable Diffusion, then vectorizing it automatically. Given text, we sample a raster image from Stable Diffusion with a Runge-Kutta solver [17] in 50 sampling steps with guidance scale $\omega = 7.5$ (the default settings in the Diffusers library [43]). Naively, the diffusion model generates photographic styles and details that are very difficult to express with a few constant color SVG paths. To encourage image generations with an abstract, flat vector style, we append a suffix to the text: *"minimal flat 2d vector icon. lineal color. on a white background. trending on artstation"*. This prompt was tuned qualitatively.

Because samples can be inconsistent with captions, we sample K images and select the Stable Diffusion sample that is most consistent with the caption according to CLIP ViT-B/16 [27]. CLIP reranking was originally proposed by [29]. We choose K=4.

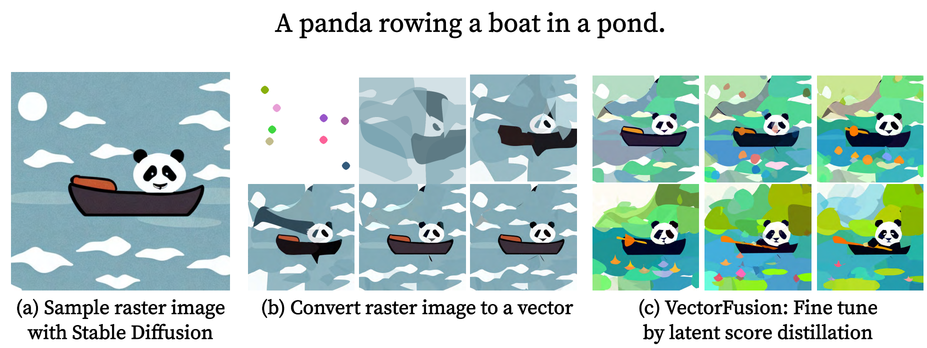

Figure 3. VectorFusion generates SVGs in three stages. **(a)** First, we sample a rasterized image from a text-to-image diffusion model like Stable Diffusion with prompt engineering for iconographic aesthetics. **(b)** Since this image is finite resolution, we approximate it by optimizing randomly initialized vector paths with an L2 loss. The loss is backpropagated through DiffVG, a differentiable vector graphics renderer, to tune path coordinates and color parameters. Paths are added in stages at areas of high loss following [19]. **(c)** However, the diffusion sample often fails to express all the attributes of the caption, or loses detail when vectorized. VectorFusion finetunes the SVG with a latent score distillation sampling loss to improve quality and coherence.

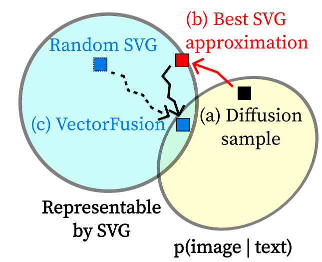

Figure 4. Conceptual diagram motivating our approach. While vectorizing a rasterized diffusion sample is lossy, VectorFusion can either finetune the best approximation or optimize a random SVG from scratch to sample an SVG that is consistent with the caption.

Next, we automatically trace the raster sample to convert it to an SVG using the off-the-shelf Layer-wise Image Vectorization program (LIVE) [19]. LIVE produces relatively clean SVGs by initializing paths in stages, localized to poorly recontructed, high loss regions. To encourage paths to explain only a single feature of the image, LIVE weights an L2 reconstruction loss by distance to the nearest path,

$$\mathcal{L}_{\text{UDF}} = \frac{1}{3}\sum_{i=1}^{w\times h} d'_i \sum_{c=1}^{3}(I_{i,c} - \hat{I}_{i,c})^2 \quad (5)$$

where I is the target image, $\hat{I}$ is the rendering, c indexes RGB channels in I, d'_i is the unsigned distance between pixel i, and the nearest path boundary, and w, h are width and height of the image. LIVE also optimizes a self-intersection regularizer $\mathcal{L}_{\text{Xing}}$

$$\mathcal{L}_{\text{Xing}} = D_1(\text{ReLU}(-D_2)) + (1 - D_1)(\text{ReLU}(D_2)), \quad (6)$$

where D_1 is the characteristic of the angle between two segments of a cubic Bézier path, and D_2 is the value of $\sin(\alpha)$ of that angle. For further clarifications of notation, please refer to LIVE [19].

This results in a set of paths $\theta_{\text{LIVE}} = \{p_1, p_2, \dots p_k\}$. Figure 3(b) shows the process of optimizing vector parameters in stages that add 8-16 paths at a time. Figure 1 shows more automatic conversions. While simple, this pipeline often creates images unsuitable for vectorization.

4.2. Sampling vector graphics by optimization

The pipeline in 4.1 is flawed since samples may not be easily representable by a set of paths. Figure 4 illustrates the problem. Conditioned on text, a diffusion model produces samples from the distribution $p_\phi(\mathbf{x}|y)$. Vectorization with LIVE finds a SVG with a close L2 approximation to that image without using the caption y. This can lose information, and the resulting SVG graphic may no longer be coherent with the caption.

For VectorFusion, we adapt Score Distillation Sampling to support latent diffusion models (LDM) like the open source Stable Diffusion. We initialize an SVG with a set of paths $\theta = \{p_1, p_2, \dots p_k\}$. Every iteration, DiffVG renders a 600×600 image $\mathbf{x}$. Like CLIPDraw [4], we augment with perspective transform and random crop to get a 512×512 image $\mathbf{x}_{\text{aug}}$. Then, we propose to compute the SDS loss in latent space using the LDM encoder E_ϕ, predicting $\mathbf{z} = E_\phi(\mathbf{x}_{\text{aug}})$. For each iteration of optimization, we diffuse the latents with random noise $\mathbf{z}_t = \alpha_t\mathbf{z} + \sigma_t\epsilon$, denoise with the teacher model $\hat{\epsilon}_\phi(\mathbf{z}_t, y)$, and optimize the SDS loss using a latent-space modification of Equation 4:

$$\nabla_\theta\mathcal{L}_{\text{LSDS}} = \mathbb{E}_{t,\epsilon}\left[w(t)\Big(\hat{\epsilon}_\phi(\alpha_t\mathbf{z}_t + \sigma_t\epsilon, y) - \epsilon\Big)\frac{\partial\mathbf{z}}{\partial\mathbf{x}_{\text{aug}}}\frac{\partial\mathbf{x}_{\text{aug}}}{\partial\theta}\right] \quad (7)$$

Since Stable Diffusion is a discrete time model with $T = 1000$ timesteps, we sample $t \sim \mathcal{U}(50, 950)$ For efficiency, we run the diffusion model $\hat{\epsilon}_\theta$ in half-precision. We found it important to compute the Jacobian of the encoder $\partial\mathbf{z}/\partial\mathbf{x}_{\text{aug}}$ in full FP32 precision for numerical stability. The term $\partial\mathbf{x}_{\text{aug}}/\partial\theta$ is computed with autodifferentiation through the augmentations and differentiable vector graphics rasterizer, DiffVG. $\mathcal{L}_{\text{LSDS}}$ can be seen as an adaptation of $\mathcal{L}_{\text{SDS}}$ where the rasterizer, data augmentation and frozen LDM encoder

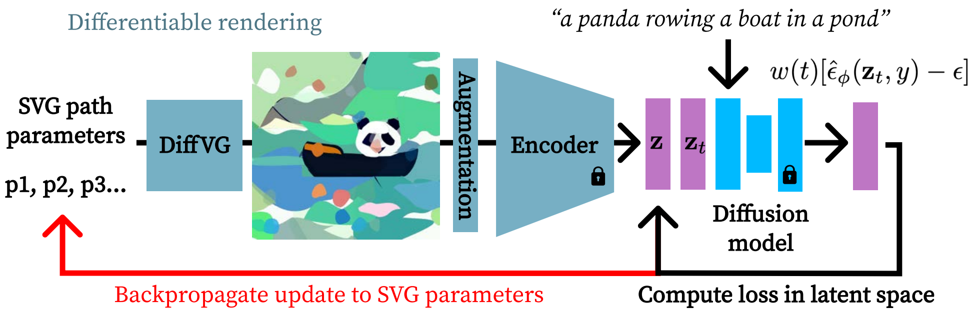

Figure 5. An overview of VectorFusion's latent score distillation optimization procedure. We adapt Score Distillation Sampling [26] to support a vector graphics renderer and a latent-space diffusion prior for raster images. First, we rasterize the SVG given path parameters. We apply data augmentations, encode into a latent space, compute the Score Distillation loss on the latents, and backpropagate through the encoding, augmentation and renderering procedure to update paths.

are treated as a single image generator with optimizable parameters θ for the paths. During optimization, we also regularize self-intersections with (6).

4.3. Reinitializing paths

In our most flexible setting, synthesizing flat iconographic vectors, we allow path control points, fill colors and SVG background color to be optimized. During the course of optimization, many paths learn low opacity or shrink to a small area and are unused. To encourage usage of paths and therefore more diverse and detailed images, we periodically reinitialize paths with fill-color opacity or area below a threshold. Reinitialized paths are removed from optimization and the SVG, and recreated as a randomly located and colored circle on top of existing paths.

4.4. Stylizing by constraining vector representation

Users can control the style of art generated by VectorFusion by modifying the input text, or by constraining the set of primitives and parameters that can be optimized. We explore three settings: iconographic vector art with flat shapes, pixel art, and sketch-based line drawings.

Iconography We use closed Bézier paths with trainable control points and fill colors. Our final vectors have 64 paths, each with 4 segments. For VectorFusion from scratch, we initialize 64 paths randomly and simultaneously, while for SD + LIVE + SDS, we initialize them iteratively during the LIVE autovectorization phase. We include details about initialization parameters in the supplement. Figure 6 qualitatively compares generations using 16, 64 and 128 paths (SD + LIVE initialization with K=20 rejection samples and SDS finetuning). Using fewer paths leads to simpler, flatter icons, whereas details and more complex highlights appear with greater numbers of paths.

Pixel art Pixel art is a popular video-game inspired style, frequently used for character and background art. While an image sample can be converted to pixel art by downsampling, this results in blurry, bland, and unrecognizable images. Thus, pixel art tries to maximize use of the available shapes to clearly convey a concept. Pixray [44] uses square SVG polygons to represent pixels and uses a CLIP-based loss following [4, 11]. VectorFusion able to generate meaningful and aesthetic pixel art from scratch and with a Stable Diffusion initialization, shown in Fig. 2 and Fig. 7. In addition to the SDS loss, we additionally penalize an L2 loss on the image scaled between -1 and 1 to combat oversaturation, detailed in the supplement. We use 32×32 pixel grids.

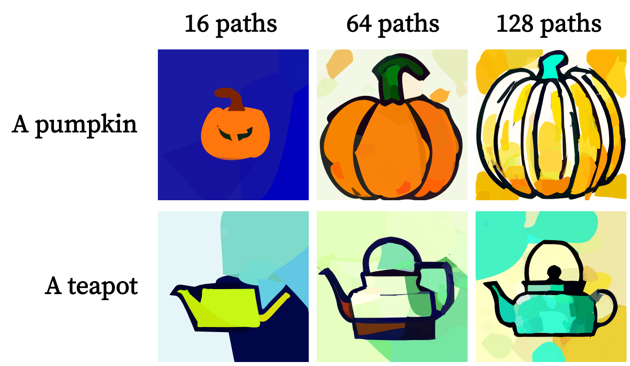

Figure 6. The number of Bézier paths controls the level of detail in generated vector graphics.

Sketches Line drawings are perhaps the most abstract representation of visual concepts. Line drawings such as Pablo Picasso's animal sketches are immediately recognizable, but bear little to no pixel-wise similarity to real subjects. Thus, it has been a long-standing question whether learning systems can generate semantic sketch abstractions, or if they are fixated on low-level textures. Past work includes directly training a model to output strokes like Sketch-RNN [7], or optimizing sketches to match a reference image in CLIP feature space [42]. As a highly constrained representation, we optimize only the control point coordinates of a set of

Table 1. Evaluating the consistency of text-to-SVG generations using 64 primitives with input captions. Consistency is measured with CLIP R-Precision and CLIP similarity score ($\times$100). Higher is better. We compare a CLIP-based approach, CLIPDraw, with diffusion baselines: the best of K raster samples from Stable Diffusion (SD), converting the best of K samples to vectors with LIVE [19], and VectorFusion from scratch or initialized with the LIVE converted SVG. VectorFusion generations are significantly more consistent with captions than Stable Diffusion samples and their automatic vector conversions. CLIPDraw is trained to maximize CLIP score, so it has artificially high scores.

		Caption consistency			
		CLIP L/14		OpenCLIP H/14	
Method	K	R-Prec	Sim	R-Prec	Sim
CLIPDraw (scratch)	–	**85.2**	27.2	77.3	**31.7**
Stable Diff (**raster**)	1	67.2	23.0	69.5	26.7
+ rejection sampling	4	81.3	24.1	80.5	28.2
SD init + LIVE	1	57.0	21.7	59.4	25.8
+ rejection sampling	4	69.5	22.9	65.6	27.6
VectorFusion (scratch)	–	76.6	24.3	69.5	28.5
+ SD init + LIVE	1	78.1	**29.1**	78.1	29.3
+ rejection sampling	4	81.3	24.5	**78.9**	29.4

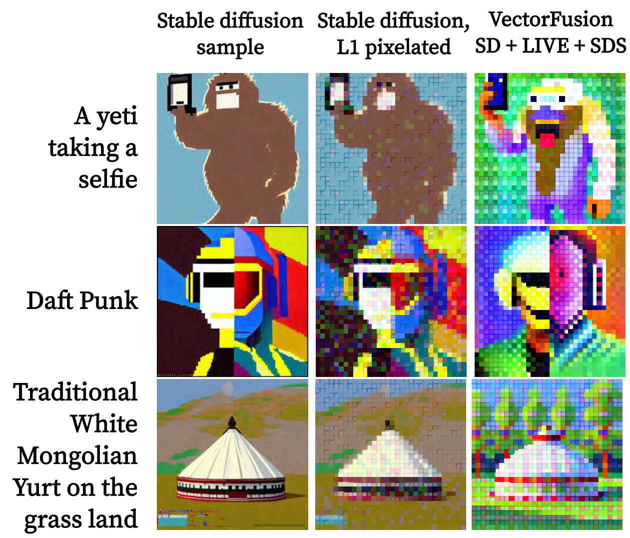

Figure 7. **VectorFusion generates coherent, pleasing pixel art.** Stable Diffusion can generate a pixel art style, but has no control over the regularity and resolution of the pixel grid (left). This causes artifacts and blurring when pixelating the sample into a 32x32 grid, even with a robust L1 loss (middle). By finetuning the L1 result, VectorFusion improves quality and generates an abstration that works well despite the low resolution constraint.

fixed width, solid black Bézier curves. We use 16 strokes, each 6 pixels wide with 5 segments, randomly initialized and trained from scratch, since the diffusion model inconsistently generates minimal sketches.

5. Experiments

In this section, we quantitatively and qualitatively evaluate the text-to-SVG synthesis capabilities of VectorFusion guided by the following questions. In Section 5.2, we ask *(1) Are SVGs generated by VectorFusion consistent with representative input captions?* and *(2) Does our diffusion optimization-based approach help compared to simpler baselines?* In Section 5.3, we qualitatively compare VectorFusion's diffusion-based results with past CLIP-based methods. Section 5.4 and 5.5 describe pixel and sketch art generations. Overall, VectorFusion performs competitively on quantitative caption consistency metrics, and qualitatively produces the most coherent and visually pleasing vectors.

5.1. Experimental setup

It is challenging to evaluate text-to-SVG synthesis, since we do not have target, ground truth SVGs to use as a reference. We collect a diverse evaluation dataset of captions and evaluate text-SVG coherence with automated CLIP metrics. Our dataset consists of 128 captions from past work and benchmarks for text-to-SVG and text-to-image generation: prompts from CLIPDraw [4] and ES-CLIP [40], combined with captions from PartiPrompts [45], DrawBench [34], DALL-E 1 [29], and DreamFusion [26]. Like previous works, we calculate CLIP R-Precision and cosine similarity.

CLIP Similarity We calculate the average cosine similarity of CLIP embeddings of generated images and the text captions used to generate them. Any prompt engineering is excluded from the reference text. As CLIP Similarity increases, pairs will generally be more consistent with each other. We note that CLIPDraw methods directly optimize CLIP similarity scores and have impressive metrics, but rendered vector graphics are sketch-like and messy unlike the more cohesive VectorFusion samples. We provide examples in Figure 8. To mitigate this effect, we also evaluate the open source Open CLIP ViT-H/14 model, which uses a different dataset for training the representations.

CLIP R-Precision For a more interpretable metric, we also compute CLIP Retrieval Precision. Given our dataset of captions, we calculate CLIP similarity scores for each caption and each rendered image of generated SVGs. R-Precision is the percent of SVGs with maximal CLIP Similarity with the correct input caption, among all 128.

5.2. Evaluating caption consistency

As a baseline, we generate an SVG for each caption in our benchmark using CLIPDraw [4] with 64 strokes and their default hyperparameters. We sample 4 raster graphics per prompt from Stable Diffusion as an oracle. These are

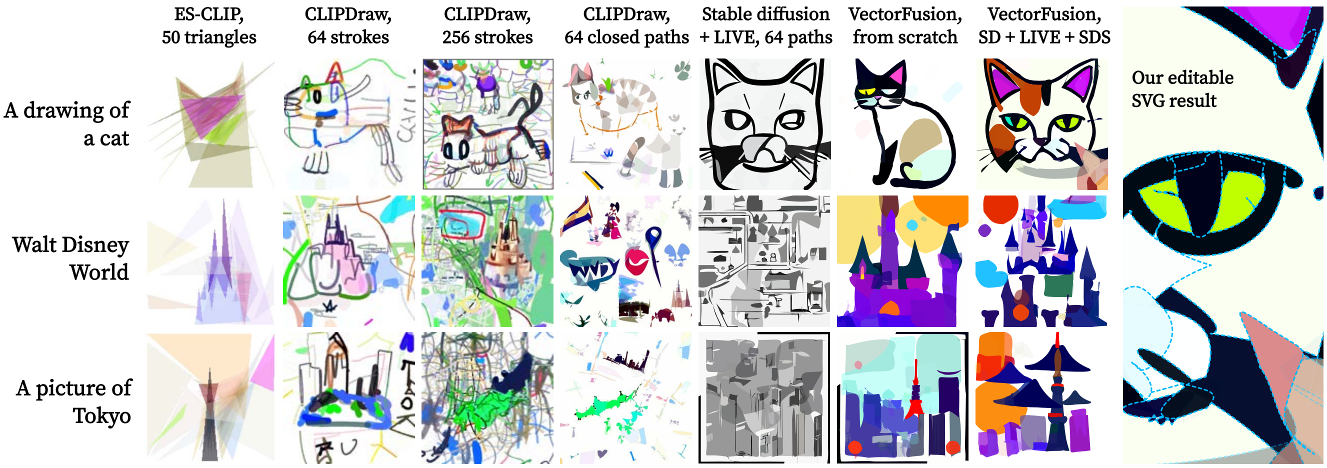

Figure 8. VectorFusion produces more coherent vector art than baselines that optimize CLIP score, even with fewer paths (64 shapes). On the right, the resulting SVG can be enlarged to arbitrary scale. Individual paths are highlighted with blue dashed lines. The result can be edited intuitively by the user in design software.

selected amongst with CLIP reranking (rejection sampling). Stable Diffusion produces rasterized images, not SVGs, but can be evaluated as an oracle with the same metrics. We then autotrace the samples into SVGs using LIVE with 64 strokes, incrementally added in 5 stages of optimization. Finally, we generate with VectorFusion, trained from scratch on 64 random paths per prompt, or initialized with LIVE.

5.3. Comparison with CLIP-based approaches

Figure 8 qualitatively compares diffusion with CLIP-guided text-to-SVG synthesis. ES-CLIP [40] is an evolutionary search algorithm that searches for triangle abstractions that maximize CLIP score, whereas CLIPDraw uses gradient-based optimization. VectorFusion produces much clearer, cleaner vector graphics than CLIP baselines, because we incorporate a generative prior for image appearance. However, a generative prior is not enough. Optimizing paths with the latent SDS loss $\mathcal{L}_{\text{LSDS}}$ (right two columns) further improves vibrancy and clarity compared to tracing Stable Diffusion samples with LIVE.

5.4. Pixel art generation

VectorFusion generates aesthetic and relevant pixel art. Figure 2 shows that VectorFusion from scratch can generate striking and coherent samples. Figure 7 shows our improvements over L1-pixelated Stable Diffusion samples, which are pixelated by minimizing an L1 loss with respect to square colors.

5.5. Sketches and line drawings

Figure 2 includes line drawing samples. VectorFusion produces recognizable and clear sketches from scratch without any image reference, even complex scenes with multiple objects. In addition, it is able to ignore distractor terms irrelevant to sketches, such as *“watercolor”* or *“Brightly colored”* and capture the semantic information of the caption.

6. Discussion

We have presented VectorFusion, a novel text-to-vector generative model. Without access to datasets of captioned SVGs, we use pretrained diffusion models to guide generation. The resulting abstract SVG representations can be intuitively used in existing design workflows. Our method shows the effectiveness of distilling generative models compared to using contrastive models like CLIP. In general, we are enthusiastic about the potential of scalable generative models trained in pixel space to transfer to new tasks, with interpretable, editable outputs. VectorFusion provides a reference point for designing such systems.

VectorFusion faces certain limitations. For instance, forward passes through the generative model are more computationally expensive than contrastive approaches due to its increased capacity. VectorFusion is also inherently limited by Stable Diffusion in terms of dataset biases [1] and quality, though we expect that as text-to-image models advance, VectorFusion will likewise continue to improve.

Acknowledgements

We thank Paras Jain, Ben Poole, Aleksander Holynski and Dave Epstein for helpful discussions about this project. VectorFusion relies upon several open source software libraries [8, 16, 19, 25, 30, 43]. This work was supported in part by the BAIR Industrial Consortium.

References

[1] Abeba Birhane, Vinay Uday Prabhu, and Emmanuel Kahembwe. Multimodal datasets: misogyny, pornography, and malignant stereotypes, 2021. 8

[2] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis, 2020. 4

[3] Chrisantha Fernando, S. M. Ali Eslami, Jean-Baptiste Alayrac, Piotr Mirowski, Dylan Banarse, and Simon Osindero. Generative art using neural visual grammars and dual encoders, 2021. 3

[4] Kevin Frans, Lisa B. Soros, and Olaf Witkowski. Clipdraw: Exploring text-to-drawing synthesis through language-image encoders. *CoRR*, abs/2106.14843, 2021. 3, 4, 5, 6, 7

[5] Gabriel Goh, Nick Cammarata †, Chelsea Voss †, Shan Carter, Michael Petrov, Ludwig Schubert, Alec Radford, and Chris Olah. Multimodal neurons in artificial neural networks. *Distill*, 2021. https://distill.pub/2021/multimodal-neurons. 3

[6] Alexandros Graikos, Nikolay Malkin, Nebojsa Jojic, and Dimitris Samaras. Diffusion models as plug-and-play priors. *arXiv:2206.09012*, 2022. 3

[7] David Ha and Douglas Eck. A neural representation of sketch drawings, 2017. 6

[8] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, Sept. 2020. 8

[9] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 2020. 3, 4

[10] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv:2207.12598*, 2022. 4

[11] Ajay Jain. Vectorascent: Generate vector graphics from a textual description, 2021. 3, 4, 6

[12] Ajay Jain, Ben Mildenhall, Jonathan T. Barron, Pieter Abbeel, and Ben Poole. Zero-shot text-guided object generation with dream fields. *CVPR*, 2022. 4

[13] Ajay Jain, Matthew Tancik, and Pieter Abbeel. Putting nerf on a diet: Semantically consistent few-shot view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5885–5894, October 2021. 4

[14] Diederik P Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *NeurIPS*, 2021. 4

[15] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *ICLR*, 2014. 3

[16] Tzu-Mao Li, Michal Lukáč, Gharbi Michaël, and Jonathan Ragan-Kelley. Differentiable vector graphics rasterization for editing and learning. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 39(6):193:1–193:15, 2020. 3, 8

[17] Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds, 2022. 4

[18] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models, 2022. 3

[19] 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 conference on computer vision and pattern recognition*, 2022. 3, 5, 7, 8

[20] Gal Metzer, Elad Richardson, Or Patashnik, Raja Giryes, and Daniel Cohen-Or. Latent-nerf for shape-guided generation of 3d shapes and textures, 2022. 3

[21] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. *ECCV*, 2020. 4

[22] Piotr Mirowski, Dylan Banarse, Mateusz Malinowski, Simon Osindero, and Chrisantha Fernando. Clip-clop: Clip-guided collage and photomontage. In *Proceedings of the Thirteenth International Conference on Computational Creativity*, 2022. 3

[23] Alexander Mordvintsev, Nicola Pezzotti, Ludwig Schubert, and Chris Olah. Differentiable image parameterizations. *Distill*, 2018. 3, 4

[24] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. In *ICML*, 2022. 4

[25] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 8

[26] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv*, 2022. 3, 4, 6, 7

[27] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. *ICML*, 2021. 3, 4

[28] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents, 2022. 1, 3

[29] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. *ICML*, 2021. 4, 7

[30] E. Riba, D. Mishkin, D. Ponsa, E. Rublee, and G. Bradski. Kornia: an open source differentiable computer vision library for pytorch. In *Winter Conference on Applications of Computer Vision*, 2020. 8

[31] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021. 1, 3, 4

[32] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. 4

[33] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. *arXiv:2205.11487*, 2022. 1

[34] Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J. Fleet, and Mohammad Norouzi. Image super-resolution via iterative refinement, 2021. 7

[35] Peter Schaldenbrand, Zhixuan Liu, and Jean Oh. Styleclipdraw: Coupling content and style in text-to-drawing translation, 2022. 3

[36] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014. 3

[37] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. *ICML*, 2015. 3, 4

[38] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *CoRR*, abs/2010.02502, 2020. 4

[39] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *ICLR*, 2021. 4

[40] Yingtao Tian and David Ha. Modern evolution strategies for creativity: Fitting concrete images and abstract concepts, 2021. 7, 8

[41] Aäron van den Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George van den Driessche, Edward Lockhart, Luis C. Cobo, Florian Stimberg, Norman Casagrande, Dominik Grewe, Seb Noury, Sander Dieleman, Erich Elsen, Nal Kalchbrenner, Heiga Zen, Alex Graves, Helen King, Tom Walters, Dan Belov, and Demis Hassabis. Parallel WaveNet: Fast high-fidelity speech synthesis. *ICML*, 2018. 4

[42] Yael Vinker, Ehsan Pajouheshgar, Jessica Y. Bo, Roman Christian Bachmann, Amit Haim Bermano, Daniel Cohen-Or, Amir Zamir, and Ariel Shamir. Clipasso: Semantically-aware object sketching, 2022. 6

[43] Patrick von Platen, Suraj Patil, Anton Lozhkov, Pedro Cuenca, Nathan Lambert, Kashif Rasul, Mishig Davaadorj, and Thomas Wolf. Diffusers: State-of-the-art diffusion models. https://github.com/huggingface/diffusers, 2022. 4, 8

[44] Tom White. Pixray. 6

[45] Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, Ben Hutchinson, Wei Han, Zarana Parekh, Xin Li, Han Zhang, Jason Baldridge, and Yonghui Wu. Scaling autoregressive models for content-rich text-to-image generation. *arXiv:2206.10789*, 2022. 7