

NIVeL: Neural Implicit Vector Layers for Text-to-Vector Generation

Vikas Thamizharasan^{*1,2} Difan Liu² Matthew Fisher²
 Nanxuan Zhao² Evangelos Kalogerakis¹ Michal Lukáč²

University of Massachusetts Amherst¹ Adobe Research²

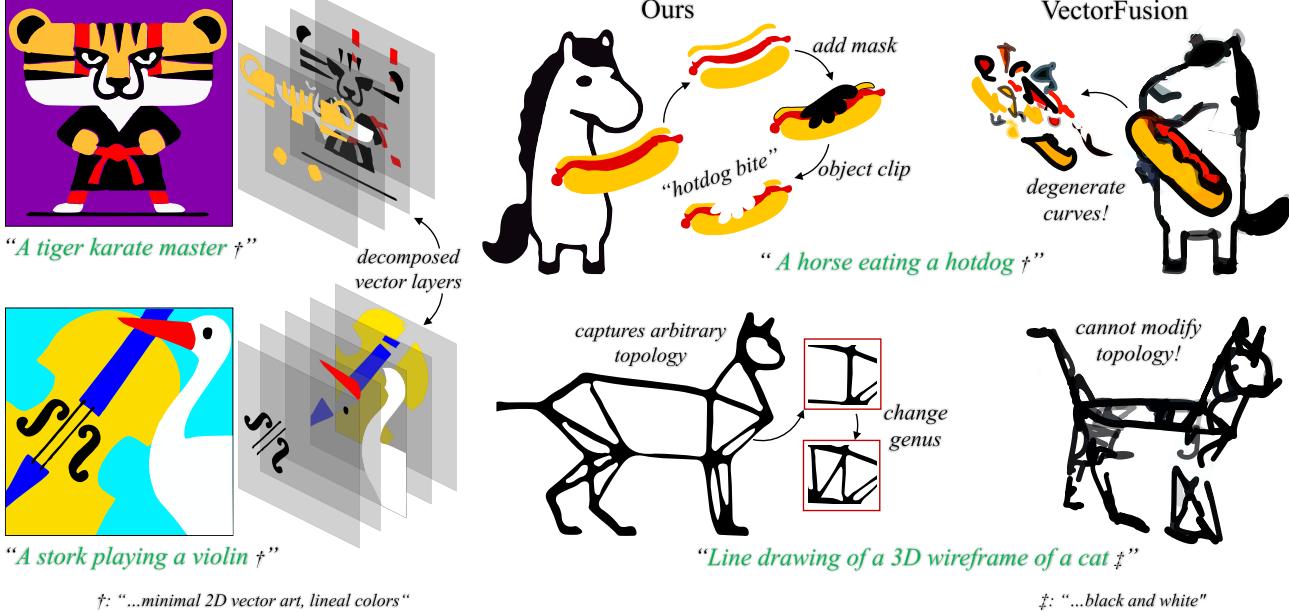


Figure 1. We present NIVeL, a neural implicit vector layer representation for generative text-to-vector graphics. Given an input **text prompt**, NIVeL outputs resolution-independent 2D shapes and colors in editable layers. Our neural representation can capture plausible shapes of arbitrary topology and genus, which were challenges of previous text-to-vector works (VectorFusion[16]) that directly operated on classical Bézier curves. Importantly, our results can easily be imported into traditional vector graphics software and make intuitive edits.

Abstract

The success of denoising diffusion models in representing rich data distributions over 2D raster images has prompted research on extending them to other data representations, such as vector graphics. Unfortunately due to their variable structure and scarcity of vector training data, directly applying diffusion models on this domain remains a challenging problem. Using workarounds like optimization via Score Distillation Sampling (SDS) is also fraught with difficulty, as vector representations are non-trivial to directly optimize and tend to result in implausible geometries such as redundant or self-intersecting shapes. NIVeL addresses these challenges by reinterpreting the problem on an alternative, intermediate domain which preserves the desirable properties of vector graphics – mainly sparsity of representation and resolution-independence. This alternative domain is based on neural implicit fields expressed in a set of decomposable, editable layers. Based on our experiments, NIVeL produces text-to-vector graphics results of significantly better quality than the state-of-the-art.

1. Introduction

Vector Graphics is a widely used representation to express visual concepts as a compact collection of primitives, such as Bézier curves, polygons, circles, lines and colors in a resolution-independent manner. Synthesizing vector graphics through generative models and high-level guidance, such as text prompts, would be highly desirable for automating modeling pipelines.

By leveraging massive scale, denoising diffusion has become the gold standard in generative raster imaging. In vector graphics however, no equivalent exists. The variable structure of the vector representation (e.g., varying number and types of primitives) means that we could only apply diffusion to a fixed subset of this domain at a time (i.e. the subdomain of vector graphics with one particular structure); and if we did so, we would not be able to find training data at requisite scale.

*Work was done during an internship at Adobe Research.
 Project website: <https://vikastmz.github.io/NIVeL/>

As a result, to enable generative vector applications, such as text-to-vector, one may turn to various workarounds, such as ex post facto vectorization of raster diffusion outputs. This is not optimal because these generated samples are not close to vector graphics; they typically contain shapes and appearance that cannot be modelled with classical vector graphic primitives without unnecessarily increasing complexity and largely degrading quality (Figure 2). Even if we accept this cost, optimal vectorization still requires a human to exhaustively search over various settings and hyper-parameters present in these systems.

Score Distillation Sampling [26] offers an alternative procedure, where a raster diffusion model can guide the optimization of a vector graphics representation, as proposed in VectorFusion [16]. But this quickly offers a different problem; in traditional vector representations, the structure of the representation depends on the content of the graphic. These changes in content are discrete, even changing the number and meaning of parameters being optimized, and so cannot be modeled by smooth optimization like SDS. Additionally, optimizing traditional vector representations on the image domain is often not stable, easily leading to degenerate geometries (Figure 1).

In this paper, we propose NIVeL, which makes use of an intermediate, vector-like representation as the optimization domain for Score Distillation Sampling. Derived from neural implicit functions, this representation maintains the resolution independence and simplicity of primitives we expect from vector graphics, while allowing for smooth arbitrary changes of shape topology without discrete jumps. Additionally, we show how to modify the SDS pipeline to reliably produce stable vector-like results in this representation, freeing us from the difficult problem of vectorizing arbitrary diffusion outputs.

Contributions. We introduce the following contributions:

- A new neural vector graphic representation based on a decomposable set of implicit fields, which captures arbitrary topology and is easier to optimize.
- A generative text-to-vector model that produces editable, layer-decomposed shapes, without any explicit prior, or supervision.
- Significant improvement in vector quality outputs over the current state-of-the-art (VectorFusion [16]).

2. Related Work

For decades now, vector graphics has been synonymous in both industrial and artistic practice with parametric primitives, such as Bézier curves [4], as also codified in one of several industrial standards (e.g. Scalable Vector Graphics [15]). Recent addition of some more expressive fill primitives [23] allow for richer appearance control where such is called for. Crucially, with visual features represented by

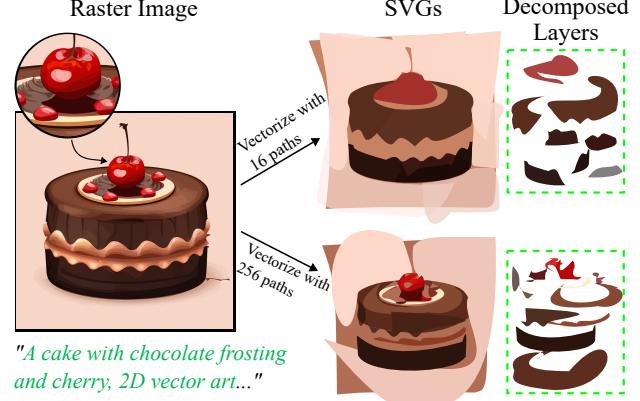


Figure 2. Sampling raster images from diffusion models, then applying vectorization leads to implausible geometries, redundant curves, and semantically meaningless layers. Here we show the results of sampling a text-to-image diffusion model DeepFloyd [2], then applying LIVE [8], a vectorizer that produces layer-decomposed SVGs. The sampled raster images often contain complex signals that are difficult to vectorize and interpret.

primitives, the structure of the representation changes with the content, which causes challenges in machine learning scenarios [30].

Sequence-generating methods. A first family of ML approaches attempt to address these challenges by applying sequence-generating learning techniques to the vector representation itself, e.g. by generating drawing commands [5, 10, 11, 18, 34–36]. Since the rendering process is non-smooth and non-trivial, it becomes difficult to maintain visual coherence and so these approaches struggle to scale.

Differentiable rasterizers. The alternative approach is to do the learning in the image domain by making use of a differentiable rasterization process [17, 27] as part of the pipeline. Such methods receive supervisory signal on the image domain to achieve a variety of effects [9, 28]. The difficulty here is that these methods need to fix the structure of the graphic, since backpropagating image-space gradients only gives a gradient signal for the existing continuous parameters of the graphic, but not for how the structure of the representation should be changed. Moreover, image-space supervision provides no accurate regularization signal on the geometry, and these optimization approaches thus often result in degenerate geometries that are difficult for users to interpret or edit.

Neural implicits. We are directly inspired by methods which address the above problem by adapting more flexible, vector-like structures, such as neural implicit fields

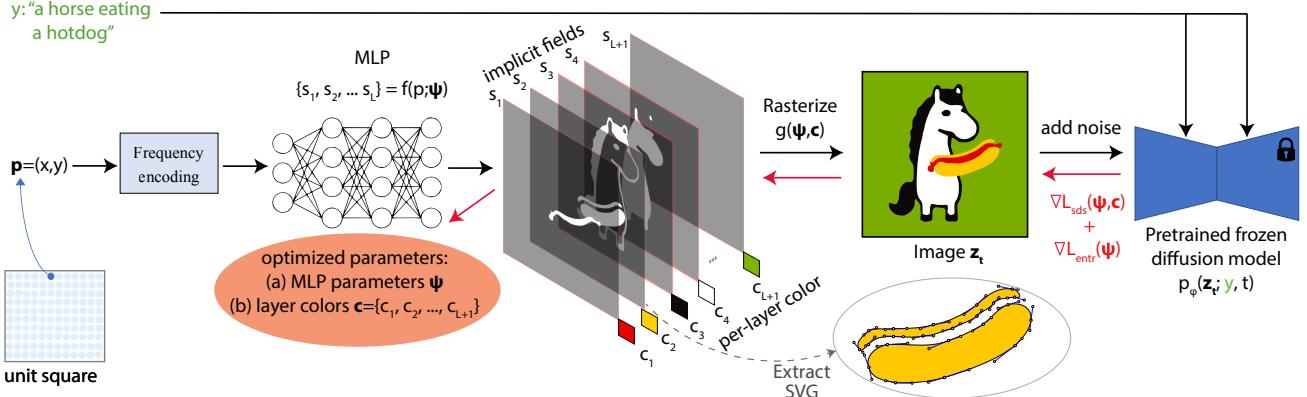


Figure 3. NIVeL’s architecture: given points p on a 2D unit square domain, our method incorporates a MLP-based network with parameters ψ that predicts a set of implicit fields on this domain, each representing a geometric shape. In addition, it predicts per-shape colors c . The representation is continuous, resolution-independent, and can easily converted to parametric curve formats (e.g., Bézier curves). To estimate the parameters, our method uses SDS-based optimization driven by a raster diffusion model conditioned on an input text prompt.

[6, 20, 24, 25, 29] proposed in 3D shape generation. These methods have the flexibility to make drastic changes to the shape topology without requiring any changes to structure. We seek to extend these to represent vector graphics and apply them to diffusion in particular.

Diffusion models for vector graphics. In the space of diffusion models [12], attempts to apply them to vector graphics have included diffusion directly on vector parameters [7, 31, 33]. Since using large foundational models often yields better results in general applications, Score Distillation Sampling [26, 32] has been the key enabler allowing the extraction of information from large-scale image-based models and application thereof to non-raster imaging. Combining this with differentiable rasterization has led to VectorFusion [16], a technique which allows for creating of vector images from text prompt as if by regular diffusion; however, like other rasterization-based methods, this one suffers from producing redundant and degraded geometry. In contrast, our method leverages the idea of neural implicit functions expressed in a set of decomposable layers to create an inherently more stable and interpretable representation, sharing some key vector properties (resolution-independence, compactness in the representation). In addition, it is easy to reinterpret the representation as a layered vector graphic. It also simultaneously renders in a differentiable manner for image-space supervision. As discussed in our experiments section, our method produces text-to-vector graphics results of significantly better fidelity than the state-of-the-art.

3. Method

Given an input text prompt, NIVeL’s goal is to synthesize 2D vector graphics representations (Figure 3). In the fol-

lowing section, we discuss our representation of geometric shapes and layers, which is used to mediate the synthesis of vector graphics in our approach. Then we discuss the optimization procedure for learning the parameters of this representation such that it synthesizes desirable vector images conditioned on text.

3.1. Representation

Shape Representation. Traditional vector graphics represent images as a set of geometric shapes defined on the 2D Cartesian plane, through parametric primitives, such as polylines and polygons, open or closed parametric paths (e.g., Bezier curves), and analytic primitives (e.g., arcs, circles, rectangles). These geometric shapes can be stacked on top of each other as different layers to create a target vector image. Inferring these geometric shapes by directly predicting these parametric primitives from high-level input is cumbersome due to their variable structure, as discussed in Section 1. We instead propose to represent geometric shapes in different layers as a 2D continuous implicit function f with learnable parameters ψ :

$$f(\mathbf{p}; \psi) : \mathcal{R}^2 \rightarrow [0, 1]^L.$$

The implicit takes as input a 2D point \mathbf{p} in the unit square $\mathbf{p} = (x, y) \in [0, 1]^2$ and outputs a probability whether the point is present in a shape (classification as 1) or absent from the shape (classification as 0) in each of the layers. The number of layers L is a hyper-parameter and represents an *upper bound* to the actual number of layers used in our representation i.e., some predicted layers can be empty (i.e. zero output for all their points), thus are unused in the final image. Similar to other neural implicit representations proposed in vision and graphics [37], we model f with the help of a MLP neural network (Figure 3).

Positional encoding. The network takes as input a positional encoding $\gamma(\mathbf{p})$ of the input (x, y) coordinates that maps each point into a higher dimensional space. The space is useful to capture shape variations at different frequencies, as also used in other neural implicit representations, such as NeRFs [21]:

$$\begin{aligned}\gamma(\mathbf{p}) &= [\sin(O\pi\mathbf{p}) \quad \cos(O\pi\mathbf{p})] \\ \text{where } O &= [2^0 \quad 2^1 \quad \dots \quad 2^{F-1}]^\top\end{aligned}\quad (1)$$

with F being a hyper-parameter in the representation denoting number of octaves.

Network architecture. The network processing these positional encodings is a sequence of fully connected layers with residual connections. More details about the MLP architecture and its hyperparameters are provided in our supplementary material. The MLP outputs a set of continuous values, or probabilities: $\mathbf{s} = \{s_l\}_{l=1}^L$, where each value s_l is bounded in $[0, 1]$ through the use of a per-layer sigmoid non-linearity. In short, our network implements the following transformation parameterized by the MLP parameters ψ :

$$\mathbf{s} = \text{sigmoid}\left(MLP(\gamma(\mathbf{p}); \psi)\right) \quad (2)$$

Color Representation. Traditional vector graphics also represent color per primitive. We represent per-layer color as a set of $L + 1$ learnable parameters $\mathbf{c} = \{\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_{L+1}\}$ taking RGB values: $\mathbf{c}_l \in [0, 1]^3$. The inclusion of the last color parameter \mathbf{c}_{L+1} is associated with the background color that a vector image can have, or in other words, the color of a dummy layer representing background.

Layer composition. The final vector image is created by stacking the predicted layer outputs on top of each other. The last dummy layer $L + 1$ is placed as the last layer in the stack, while the first layer created by the MLP outputs \mathbf{s}_1 is the front-most one. Any shape at layer l occludes shapes at layers $[l + 1, \dots, L + 1]$. The output continuous image is created as a layer compositing function $g(\mathbf{p}; \psi, \mathbf{c})$:

$$g(\mathbf{p}; \psi, \mathbf{c}) = \sum_{l=1}^{L+1} k_l \cdot c_l, \quad (3)$$

where $k_l = s_l \prod_{m < l} (1 - s_m)$ is a mixing coefficient for the layer at that point, computed from opacities of layers on top of it. We note that the final image is a continuous representation with “infinite” resolution, as in the case of vector images i.e., one can zoom into a region of the unit square without pixelization artifacts. We also note that the sequence of layers in the stack matters – the network should

learn to distribute shapes to layers such that their composition with the above function yields a desired target image. Fortunately, for learning purposes our compositing function is differentiable wrt both color parameters \mathbf{c} as well as ψ i.e., $\partial g / \partial \psi = \sum_l (\partial g / \partial k_l) \cdot (\partial k_l / \partial s_l) \cdot (\partial s_l / \partial \psi)$ – here, we omit input points \mathbf{p} for clarity.

3.2. Parameter estimation

As discussed in our introduction, compared to raster images, there is a lack of datasets including SVGs and text pairs. One possibility to circumvent this problem is to leverage generative models of raster images trained on massive datasets of generic images for vector image synthesis in a zero-shot generation setting. Most recent powerful generative models of raster images are based on diffusion models. One problem, however, is that these models output raster images with incompatible data types, style, and dimensionality compared to vector or implicit representations, such as ours. A common strategy to deal with this incompatibility is to perform parameter estimation and sample synthesis via optimization based on the score distillation approach [16, 26].

Score distillation from image-based diffusion models. The goal of score distillation in our setting is to estimate parameters of our implicit representation such that it synthesizes a sample output with high probability according to a pre-trained image-based diffusion model conditioned on text prompts related to vector styles (e.g., “minimal 2D vector art, lineal colors, line drawing”, see also Figure 1). The loss penalizes the KL-divergence of a unimodal Gaussian distribution centered at a learned sample produced by our model $g(\psi, \mathbf{c})$ and the data distribution $\mathbf{p}_\phi(\mathbf{z}; y, t)$ captured by the frozen diffusion model conditioned on text embeddings y . The loss is averaged over several time steps t sampled throughout the diffusion process:

$$\begin{aligned}\mathcal{L}_{\text{sds}}(\psi, \mathbf{c}) &= \\ \mathbb{E}_t \left[\frac{\sigma_t}{\alpha_t} w(t) \text{KL}(q(\mathbf{z}_t | g(\psi, \mathbf{c}); y, t) \| p_\phi(\mathbf{z}_t; y, t)) \right]\end{aligned}\quad (4)$$

where t is a timestep, $w(t)$ is a weighting function and α_t, σ_t are coefficients of the diffusion model depending on the timestep t . The loss perturbs the image produced by our model $g(\psi, \mathbf{c})$ with a random amount of noise corresponding to the timestep, and estimates an update direction that moves it towards a higher probability density region dictated by the diffusion model, while being constrained on the implicit data representation imposed by our model.

In addition to the above SDS loss, we found useful to include a regularization term penalizing uncertainty in our model’s mixing coefficients. Since we do not model translucent shapes, we prefer these mixing coefficients to

be skewed towards either 0 or 1 (see also Figure 4). The preference can be expressed through the following entropy loss applied to the per-layer mixing coefficients k_l , involving only the MLP parameters ψ :

$$\mathcal{L}_{entr}(\psi) = - \sum_l k_l \log(k_l) \quad (5)$$

To learn the parameters of our model, we minimize a weighted sum of the SDS loss and the above entropy loss:

$$\mathcal{L}_{total} = \mathcal{L}_{sds}(\psi, \mathbf{c}) + \lambda \mathcal{L}_{entr}(\psi) \quad (6)$$

where λ is a hyperparameter serving as weighting term for the entropy loss.

Initialization. To initialize the above SDS-based optimization, one possibility is to start with random values for parameters $\{\psi, \mathbf{c}\}$. Unfortunately, this strategy is suboptimal. The initial random value of parameters often correspond to areas of low probability density according to the image-based diffusion model, where the SDS-based gradients tend to be noisy, slowing convergence or even leading to undesirable local maxima causing implausible results (see also Section 4 for an ablation wrt this strategy).

An alternative strategy for initialization is to sample a raster image from the diffusion model, then initialize our parameters such that the output of our model reconstructs the sample image as closely as possible. More specifically, given a sample RGB raster image $\hat{\mathbf{z}}$ produced by executing the reverse diffusion of an existing pretrained diffusion model conditioned on the input text prompt, the parameters can be pre-trained to minimize the loss:

$$\mathcal{L}_{rec} = \mathcal{L}_2(\psi, \mathbf{c}) + \lambda' \mathcal{L}_{entr}(\psi) \quad (7)$$

where \mathcal{L}_2 is summed over the pixel locations of the raster image: $\mathcal{L}_2(\psi, \mathbf{c}) = \sum_{\mathbf{p} \in \mathbf{R}} \|g(\mathbf{p}; \psi, \mathbf{c}) - \hat{\mathbf{z}}(\mathbf{p})\|^2$ (the effect of including the entropy loss is highlighted in Fig. 4). Still, this initialization strategy was also suboptimal, as shown in our experiments. The reason is that the sample raster images tend to incorporate a photographic style with high-frequency texture variation, details and natural background that are not typically present in illustrations modeled by artists in vector graphics, even with text prompts related to vector style, as also noted in VectorFusion [16].

To overcome these challenges, we employ to an approach where the initialization of our model is not based on fitting it to a sample raster image, but instead a low-frequency image representation better matching the output parameterizaton of our model. More specifically, during our initialization phase, we train a model mapping continuous image coordinates $\mathbf{p} = (x, y) \in [0, 1]^2$ to RGB values using the same MLP-based architecture as in Eq. 2. Its output is a predicted per-point RGB color $\mathbf{z}(\mathbf{p}; \theta)$ in $[0, 1]^3$, and

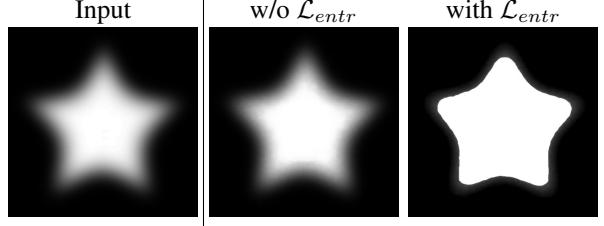


Figure 4. Given an input sampled image (left), we estimate the NIVeL’s parameters through L2 reconstruction loss without entropy L_{entr} (middle), or with entropy (right). The entropy results in a cleaner shape with delineated boundaries.

θ are the MLP parameters estimated via SDS optimization guided by the same image-based diffusion model i.e., we use Eq. 4 with the above implicit RGB image generator instead. An important design choice of this generator is that the used positional encoding $\gamma(\mathbf{p})$ incorporates a limited set of octaves (up to $F = 6$ bands in our implementation) to encourage low-frequency texture and background variation in the output image.

After estimating the parameters θ of the above implicit RGB generator, we initialize the parameters $\{\psi, \mathbf{c}\}$ of our model using the reconstruction-based loss \mathcal{L}_{rec} . Finally, we fine-tune the parameters with our main loss \mathcal{L} (Equation 6). This combination of initialization and fine-tuning offered the best results in our experiments.

3.3. Implementation details

First, we note that our implementation *will become publicly available upon acceptance*. Below we discuss important implementation details.

Image-based diffusion model. We use the open sourced pre-trained DeepFloyd model [2] to compute our SDS gradients during the initialization and fine-tuning phase. DeepFloyd performs denoising diffusion in pixel-space, unlike Stable Diffusion which performs it in latent space. This avoids the computational bottleneck of back-propagating through the image encoder and, in practice, we observe better results and faster convergence with DeepFloyd.

Network architecture We trained two versions of our model for evaluation: a 12,000 (12K) parameter MLP with 64 hidden nodes and 4 layers, and a smaller 1,000 (1K) parameter MLP with 32 hidden nodes and 3 layers. We set F (number of octaves) to 6 for the former and 2 for the latter. Note, the frequency encoding is a fixed function without additional learnable parameters. Both these models contain LeakyReLU activations and Sigmoid at the final layer. We set L (the number of layers) to 5 for all experiments to have a comparable trainable parameter count to VectorFusion. We discard unused layers (all zero outputs) after optimization. We jitter the query points during training for addi-

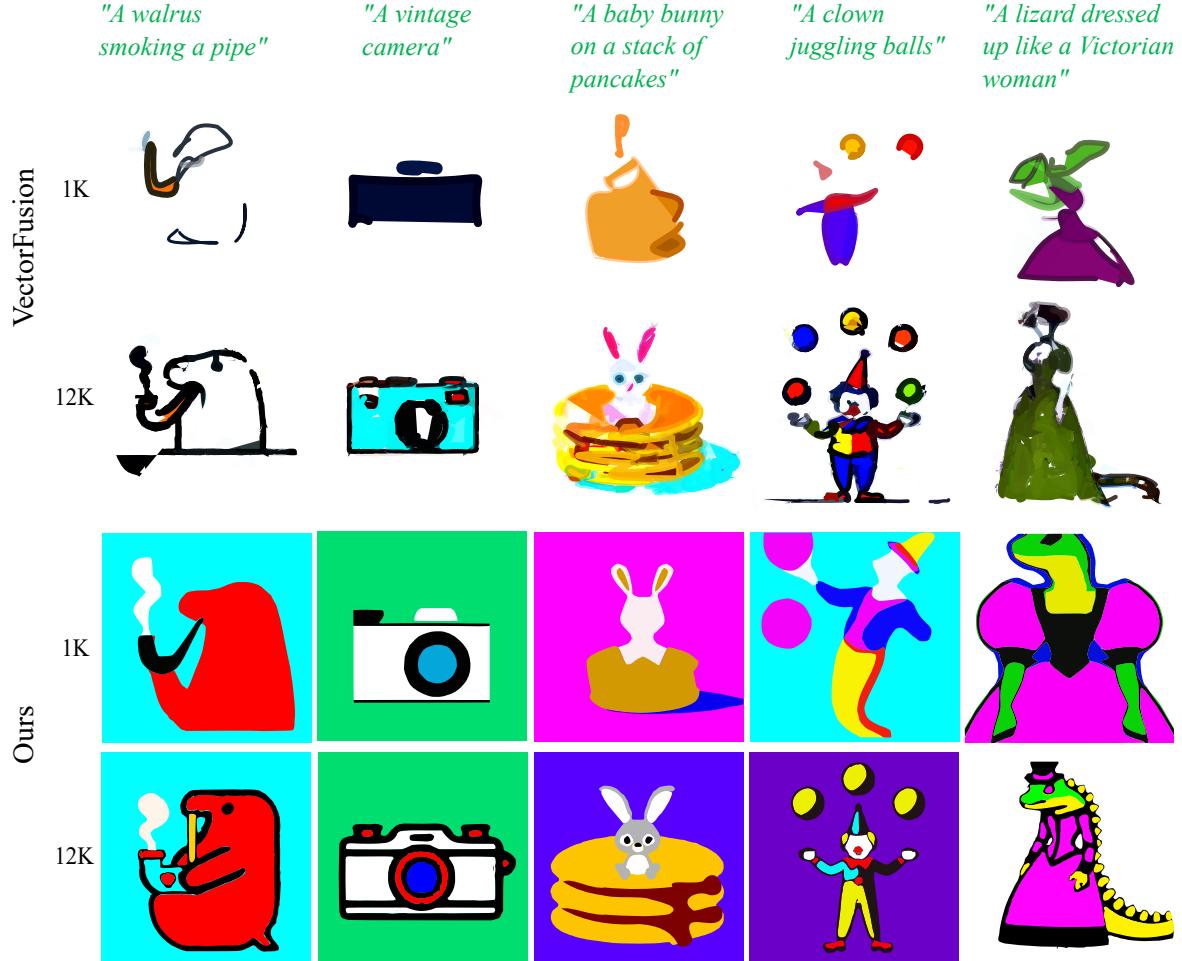


Figure 5. Text-to-Vector Graphics generation results. We compare generated SVG results for the [input text prompt](#) between NIVeL (ours) vs VectorFusion, at two settings involving $1K$ or $12K$ number of parameters. Our vector results contain much cleaner shape geometry across diverse topologies while VectorFusion’s SVGs contain redundant, degenerate curves, and self-intersecting shapes. Our method also remains robust at a low capacity ($1K$ parameters), faithfully capturing the abstraction of the concepts in the input text prompt.

tional robustness to evaluation points during inference. We implement our model using the tiny-cuda-nn library [22].

SDS scheduler and optimization. We use the vanilla timestep scheduler proposed in DreamFusion [26]. We note that we explored with alternative timestep schedulers (e.g., Dreamtime [13]), yet it did not offer any improvements in our setting. We perform the SDS-based optimization using AdamW [19]. We set the loss hyperparameters $\lambda = 10^{-5}$, $\lambda' = 10^{-4}$, batch size to 3, and total optimization iterations to 8000 for all our experiments. Our final model takes 5 minutes to converge on a single NVIDIA A100 gpu. We provide a model card in table 2.

Converting implicit layers to Bézier curves. We extract iso-curves from our layered implicit shapes through marching squares at any prescribed resolution. We query f_ψ at $N \times N$ grid points \mathbf{p} to produce L raster shapes ($N = 2048$

in our implementation). For our experiments, we set $L = 5$ as the upper bound on the number of layers. We analyse the effect of different L on the generated results for a given text prompt in section 5. We then fit cubic Bézier curves to each of our layers using an open source image tracing software (Inkscape [14]).

4. Experiments

Dataset. We use the prompt dataset curated by VectorFusion [3] to generate our qualitative results and compute our quantitative metrics. This dataset consists of two prompt families: (i) line drawings with prefix and suffix “*Line drawing of ..., minimal 2d line drawing, on a white background, black and white*”, and (ii) minimal lineal colors with suffix “*..., minimal flat 2d vector art, lineal color, on a white background, trending on artstation*”. We include all prompts in section E.

Method	Parameters	CLIP L/14			
		R-Prec ↑		Sim ↑	
		mean	std	mean	std
VectorFusion	1K	59.5	1.51	21.4	0.78
	12K	71.3	2.37	26.7	0.73
NIVeL	1K	68.2	1.26	25.1	0.81
	12K	78.5	1.40	32.0	0.72

Table 1. CLIP metrics computed with the *clip-vit-large-patch14* pre-trained model on rasterized SVG results of NIVeL and VectorFusion [16] under our two parameter settings. Both methods are optimized with DeepFloyd [2]. We also show the official reported results from VectorFusion when optimized with Stable Diffusion (row “with SD”). We note their implementation is not available, thus, the number of parameters in their experiments is unknown.

Comparison. We compare with VectorFusion [16], which shares the same goal of creating vector art from text prompt as our method. We note that we use our own implementation of VectorFusion, since it is not publicly available. Their method optimizes randomly initialized cubic Bezier curves with SDS. For a fair comparison, we plugged the same diffusion model DeepFloyd [2] (DF for short) as ours. In addition, we adjust both VectorFusion and our method to use a comparable number of parameters for optimization. More specifically, we use the following two settings:

1. **1K parameters.** *VectorFusion* uses 16 paths each containing up to 5 cubic Bézier curves. Each Bézier curve has 17 parameters (control points, stroke width and color, fill color)^γ. *NIVeL* uses an MLP with 32 hidden nodes and 3 layers and $F = 2$ (number of octaves).
2. **12K parameters:** *VectorFusion* uses 256 paths each containing up to 5 cubic Bézier curves (17 parameters each). *NIVeL* uses an MLP with 64 hidden nodes and 4 layers with $F = 6$ octaves.

For both settings, we extract Bezier curves for our method, using the procedure discussed in Section 3.3. We note that our output number of Bezier paths is not the same with VectorFusion. For our method, their number is automatically selected according to Inkscape’s implementation operating on our output. We argue that automatically adjusting the number of paths depending on the target output is more desirable than imposing a hard constraint on their number.

Evaluation metrics. First, we report the CLIP-based score (denoted as “Sim”) [1] also used in VectorFusion [16]. The score measures the cosine similarity between the embeddings for a raster image and the embeddings for a text caption. To this end, we rasterize the Bezier curves extracted from VectorFusion and NIVeL in the same resolu-

^γ 4 control points * 2 (x,y) + 1 (stroke width) + 8 (rgba for both stroke and fill color) = 17

Q1: Which image looks better?



Q2: Which image better matches the text prompt?



Figure 6. User study voting results.

tion (224×224) using the same rasterizer [14]. The CLIP score is averaged over all prompts. We also report the R-Precision, as also used in VectorFusion (“R-Prec”). This is the percent of output images having maximal CLIP similarity with the correct input caption among all text prompts. Finally, we conduct a perceptual evaluation based on a user study asking participants to compare results between our method and VectorFusion in terms of their plausibility and degree of matching with the input text.

Quantitative comparisons. Table 1 reports results for the CLIP-based similarity and precision measures, comparing our method and VectorFusion. We outperform VectorFusion for both parameter setting, demonstrating the parameter efficiency of our representation and faithfulness of our generation to the input prompt. The table row “VectorFusion (with SD)” represents their originally reported metrics when optimizing randomly initialized curves with Stable Diffusion (SD for short) on the same dataset [16]. We note that their implementation is not available, thus, the number of parameters used in their experiments is unknown.

Qualitative comparisons. Figures 1, 5, and 11 compare the results of our method and VectorFusion qualitatively. Compared to VectorFusion, our results are devoid of artifacts (floating curves, degenerate geometry) and capture cleaner shapes and appearance (i.e., properties commonly associated with vector graphics). In addition, our representation is parameter-efficient: with just 1K parameters, our generated results are faithful to the prompt and capture the input concepts. On the other hand, VectorFusion cannot faithfully capture the geometry of objects with low parameter count. With 12K parameters, VectorFusion’s generated results include redundant curves and suffers from the aforementioned artifacts, making their results much harder to edit intuitively (Figure 1). Section B details results on the line drawing style.

User study. We also conducted two Amazon MTurk studies as additional perceptual evaluations. In the first study, each questionnaire page showed participants a randomly ordered pair of raster outputs from our method and Vec-

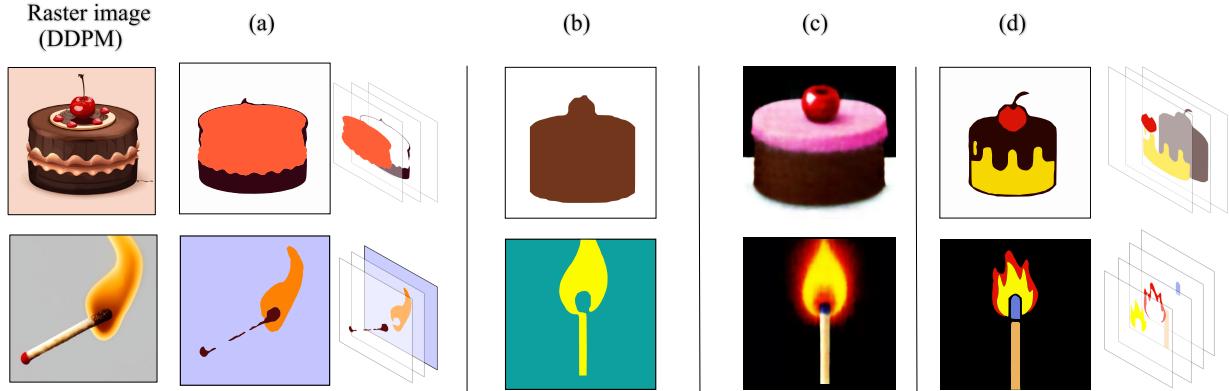


Figure 7. For the prompts “*A cake with chocolate frosting and cherry, 2D vector art*” and “*A match stick on fire, 2D vector art*” we sample raster images from a pre-trained text-to-image Diffusion Model via DDPM (a) We optimize NIVeL by using the reconstruction-based loss \mathcal{L}_{rec} on the sampled image (no SDS optimization). (b) We optimize NIVeL by using our SDS-based loss \mathcal{L}_{total} with random initialization. (c) We optimize NIVeL by using the reconstruction-based loss \mathcal{L}_{rec} on the RGB generator network (no SDS optimization), (d) We optimize NIVeL by fine-tuning with SDS-based loss after initialization from the RGB generator network. The last strategy offers the most visually compelling vector art outputs.

torFusion. We asked participants “which image looked better”, explaining to them in our instructions that they should choose based on which image had fewer artifacts and looked more aesthetically plausible. Participants could pick either image, specify “none” or “both” images looked good enough. We asked questions twice in a random order to verify participants’ reliability. We had 100 reliable participants, each comparing 10 unique pairs generated from our pool of text prompts (total 1000 comparisons).

In our second study, we showed participants randomly ordered pairs of outputs from NIVeL and VectorFusion, along with the input text prompt, and asked them “which image better matches the input text prompt”. Participants could again pick either image, specify “none” or “both” images matched the input text well. We also asked questions twice in a random order to verify participants’ reliability. We again had 100 reliable participants, each comparing 10 unique pairs as before (total 1000 comparisons).

Figure 6 summarizes the percentage of votes for the above two types of questions. We observe that our method’s outputs were preferred by a significantly larger proportions of participants for both questions.

Ablation. In Figure 7 we demonstrate the importance of a good initialization strategy. A randomly initialized network is often prone to failure modes, wherein it allocates entire shapes into the initial layers (Figure 7b). This prevents the generation of semantically meaningful decomposed shapes. Another possibility is to initialize our network by fitting it to reconstruct a sampled image from the diffusion model (Figure 7a). This initialization yields a rather coarse, oversimplified layers. In contrast, initializing the network by fitting it to reconstruct the implicit RGB generator’s output

provides a more appealing starting point (Figure 7c). Fine-tuning the network through SDS-based optimization after this initialization provides the best results (Figure 7d). Effects of using different random seeds is detailed in section A. We additionally visualize the output of SDS optimization iterations in section C.

5. Conclusion

We presented a method that proposes a layered implicit field representation as a mediator for generating vector graphics. We demonstrated significantly better results than the state-of-the-art in the case of text-to-vector synthesis.

Limitations and future work. Our method still has limitations that can spur new research directions for generative vector graphics. First, our representation is currently bounded by an upper number of layers. Generating layers dynamically would alleviate this issue. Currently, we use an open-source vectorizer to convert our implicit field to parametric curves. It would be worth investigating a differentiable implicit-to-vector module to perform this conversion. Investigating better SDS-based optimization strategies e.g., with more adaptive time-step schedulers and more stability could further improve our results. Finally, extending our representation to the 3D domain for predicting 3D parametric modeling primitives guided by neural implicits is generally a worthwhile research direction.

Acknowledgments. We thank Dmitry Petrov, Matheus Gadelha, and Thibault Grouex for their helpful discussions. Our project was funded by Adobe Research.

References

- [1] openai/clip-vit-large-patch14. https://torchmetrics.readthedocs.io/en/stable/multimodal/clip_score.html#id3. 7
- [2] Deepfloyd/if-i-xl-v1.0. <https://huggingface.co/DeepFloyd/IF-I-XL-v1.0>. 2, 5, 7
- [3] Svg text prompts dataset. https://ajayj.com/vectorfusion/svg_bench_prompts.txt. 6
- [4] Pierre Bezier. Courbes et surfaces, mathématiques et cao, 4. Hermès, Paris, 1986. 2
- [5] Alexandre Carlier, Martin Danelljan, Alexandre Alahi, and Radu Timofte. Deepsvg: A hierarchical generative network for vector graphics animation, 2020. 2
- [6] Yinbo Chen, Sifei Liu, and Xiaolong Wang. Learning continuous image representation with local implicit image function. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8628–8638, 2021. 3
- [7] Ayan Das, Yongxin Yang, Timothy Hospedales, Tao Xiang, and Yi-Zhe Song. Chirodiff: Modelling chirographic data with diffusion models, 2023. 3
- [8] Zheng-Jun Du, Liang-Fu Kang, Jianchao Tan, Yotam Gingold, and Kun Xu. Image vectorization and editing via linear gradient layer decomposition. *ACM Transactions on Graphics (TOG)*, 42(4), 2023. 2
- [9] Kevin Frans, Lisa Soros, and Olaf Witkowski. Clipdraw: Exploring text-to-drawing synthesis through language-image encoders. In *Advances in Neural Information Processing Systems*, 2022. 2
- [10] Yaroslav Ganin, Tejas Kulkarni, Igor Babuschkin, S. M. Ali Eslami, and Oriol Vinyals. Synthesizing programs for images using reinforced adversarial learning. *CoRR*, abs/1804.01118, 2018. 2
- [11] David Ha and Douglas Eck. A neural representation of sketch drawings. In *International Conference on Learning Representations*, 2018. 2
- [12] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 3
- [13] Yukun Huang, Jianan Wang, Yukai Shi, Xianbiao Qi, Zheng-Jun Zha, and Lei Zhang. Dreamtime: An improved optimization strategy for text-to-3d content creation, 2023. 6
- [14] Inkscape Project. Inkscape. 6, 7
- [15] Dean Jackson and Craig Northway. Scalable vector graphics (svg) full 1.2 specification. World Wide Web Consortium, Working Draft WD-SVG12-20050413, 2005. 2
- [16] Ajay Jain, Amber Xie, and Pieter Abbeel. Vectorfusion: Text-to-svg by abstracting pixel-based diffusion models. *arXiv*, 2022. 1, 2, 3, 4, 5, 7
- [17] 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. 2
- [18] Raphael Gontijo Lopes, David R Ha, Douglas Eck, and Jonathon Shlens. A learned representation for scalable vector graphics. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7929–7938, 2019. 2
- [19] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. 6
- [20] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 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. In *ECCV*, 2020. 4
- [22] Thomas Müller. tiny-cuda-nn, 2021. 6
- [23] Alexandrina Orzan, Adrien Bousseau, Holger Winnemöller, Pascal Barla, Joëlle Thollot, and David Salesin. Diffusion curves: A vector representation for smooth-shaded images. In *ACM Transactions on Graphics*, 2008. 2
- [24] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019. 3
- [25] Despoina Paschalidou, Angelos Katharopoulos, Andreas Geiger, and Sanja Fidler. Neural parts: Learning expressive 3d shape abstractions with invertible neural networks, 2021. 3
- [26] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv*, 2022. 2, 3, 4, 6
- [27] Pradyumna Reddy, Paul Guerrero, Matt Fisher, Wilmot Li, and Niloy J Mitra. Discovering pattern structure using differentiable compositing. *ACM Transactions on Graphics (TOG)*, 39(6):1–15, 2020. 2
- [28] Pradyumna Reddy, Michael Gharbi, Michal Lukac, and Niloy J. Mitra. Im2vec: Synthesizing vector graphics without vector supervision, 2021. 2
- [29] Pradyumna Reddy, Zhifei Zhang, Matthew Fisher, Hailin Jin, Zhaowen Wang, and Niloy J Mitra. A multi-implicit neural representation for fonts. *arXiv preprint arXiv:2106.06866*, 2021. 3
- [30] Dmitriy Smirnov, Matthew Fisher, Vladimir G. Kim, Richard Zhang, and Justin Solomon. Deep parametric shape predictions using distance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2
- [31] Vikas Thamizharasan, Difan Liu, Shantanu Agarwal, Matthew Fisher, Michael Gharbi, Oliver Wang, Alec Jacobson, and Evangelos Kalogerakis. Vecfusion: Vector font generation with diffusion, 2023. 3
- [32] Haochen Wang, Xiaodan Du, Jiahao Li, Raymond A. Yeh, and Greg Shakhnarovich. Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation, 2022. 3
- [33] 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*, 2023. 3
- [34] Yizhi Wang and Zhouhui Lian. Deepvecfont: Synthesizing high-quality vector fonts via dual-modality learning. *ACM Transactions on Graphics*, 40(6), 2021. 2

- [35] Yuqing Wang, Yizhi Wang, Longhui Yu, Yuesheng Zhu, and Zhouhui Lian. Deepvecfont-v2: Exploiting transformers to synthesize vector fonts with higher quality. *arXiv preprint arXiv:2303.14585*, 2023.
- [36] Ronghuan Wu, Wanchao Su, Kede Ma, and Jing Liao. Icon-shop: Text-based vector icon synthesis with autoregressive transformers. *arXiv preprint arXiv:2304.14400*, 2023. [2](#)
- [37] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. *Computer Graphics Forum*, 2022. [3](#)

Appendix

A. Additional Ablations

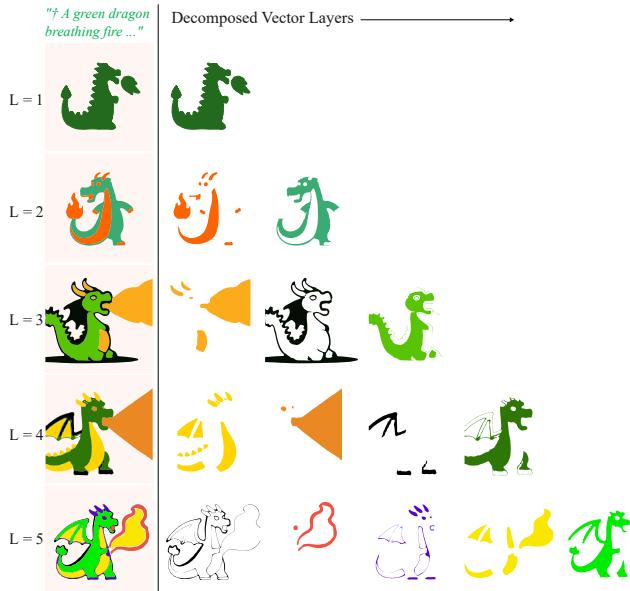


Figure 8. For the given [input text prompt](#) we show our generated results changing only the value of L (the number of output decomposed layers). †: "... minimal 2D vector art, lineal colors"

Effect of number of layers L . The number of decomposed layers generated by our system is set by the hyperparameter L . In Figure 8, we show the effect on the generated results for values of L ranging from 1 (minimum) to 5 (default value in our experiments). We fix the prompt and all other settings for all results. Our method can generate plausible shapes and colors under each constrained setting. We observe presence of distinguishable semantic parts as the number of layers L increase.

Effect of random seeds. Figure 9 and 10 show the stochasticity of the results achieved by changing the random seed of the image diffusion model.

B. Qualitative Results

In Figure 11 we compare NIVeL (ours) vs VectorFusion for a line drawing style. Since the style involves shape outlines or contours, here we use a single layer ($L = 1$). For the rest of the hyperparameters, we set $F = 6$ octaves in



Figure 9. The effect of changing the random seed for the given [input text prompt](#). †: "... minimal 2D vector art, lineal colors"

the positional encoding, and the MLP architecture has $12K$ parameters. For VectorFusion, we use 256 paths each containing upto 5 Bézier curves for a total of also $12K$ parameters. We visualize the generated SVGs without the curve fill attribute to demonstrate the accuracy and smoothness of our produced line drawings of shapes compared to VectorFusion's results which contain redundant overlapping and self-intersecting lines.

C. Visualization of SDS gradients

The SDS gradients provided by the frozen pre-trained diffusion model contain rich and visually interpretable signals for shape generation. In Figure 12 and Figure 13, we show the SDS gradients visualized as an RGB image (1st row) and the generated images (2nd row) across a subset of the 8000 optimization steps. In Figure 12, we initialize one layer with a Gaussian blob and show the evolution of the generated shape during the optimization. Note the ability of our representation to freely add or remove holes i.e change genus. Our entropy-based loss helps in producing clean boundaries. Similarly, in Figure 13, we show these effects on multi-layer generations. We set the initialized layers to be either 2D boxes or ellipses. The choice of these two initialization procedures allows an easier interpretation of the gradient signals. This also demonstrates that the ini-

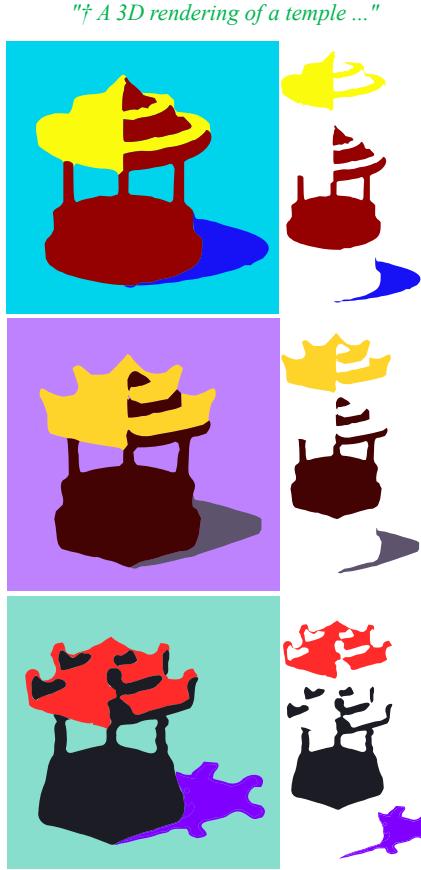


Figure 10. Rich priors learned by the image diffusion model are preserved in our generated shapes. Specifically, the shape of the shadow matches the style and structure of the roof of the temple across the three shown synthesis runs. Each generation uses a different random seed. †: "... minimal 2D vector art, lineal colors"

tialization for NIVeL can be handcrafted, if desired. For the two prompts, "A grizzly bear karate master..." and "A flamingo karate master...", one can observe the effect of the initialization not just on the generated shapes during the optimization but also in the SDS gradients. These two initialization show the different outcomes explored via SDS and how geometric properties of the initial shapes are preserved i.e., the presence of sharp corners with the boxes.

D. User study details

Figure 14 shows a screenshot of the webpage containing an example question used in our perceptual evaluation questionnaires. Each questionnaire was released via the MTurk platform. It contained 10 unique questions, each asking for one comparison between NIVeL and VectorFusion. Then these 10 questions were repeated in the questionnaire in a random order. In these repeated questions, the order of compared illustrations was flipped. If a worker gave more than 3 inconsistent answers for the repeated questions, then he/she was marked as "unreliable". Each participant was

Hyperparameters	
Guidance scale	14.0
t (timestep)	$\sim \mathcal{U}(0, 1)$
L	5
Iterations	8000
Learning rate (Color)	$5e^{-3}$
Batch size	3
λ	$1e^{-5}$
λ'	$1e^{-4}$

12K Model	
F	6
Layers	4
Hidden nodes	64
Activation	LeakyReLU
Learning rate (MLP)	$1e^{-2}$

1K Model	
F	2
Layers	3
Hidden nodes	32
Activation	LeakyReLU
Learning rate (MLP)	$1e^{-3}$

Table 2. Model cards of NIVeL’s variants used in our experiments.

allowed to perform the questionnaire only once. The results are shown in Figure 6 of the main text.

E. List of prompts used in our experiments

Below we show all the text prompts used in our comparisons with VectorFusion (including our user study), as discussed in the main text.

- "Line drawing of Third eye, minimal 2d line drawing, on a white background, black and white"
- "Line drawing of a baby penguin, minimal 2d line drawing, on a white background, black and white"
- "Line drawing of a ladder, minimal 2d line drawing, on a white background, black and white"
- "Line drawing of a crown, minimal 2d line drawing, on a white background, black and white"
- "Line drawing of A cat as 3D rendered in Unreal Engine, minimal 2d line drawing, on a white background, black and white"
- "Line drawing of Fast Food, minimal 2d line drawing, on a white background, black and white"
- "Line drawing of Happiness, minimal 2d line drawing, on a white background, black and white"
- "Line drawing of Family vacation to Walt Disney World, minimal 2d line drawing, on a white background, black and white"
- "Line drawing of a bottle of beer next to an ashtray with a half-smoked

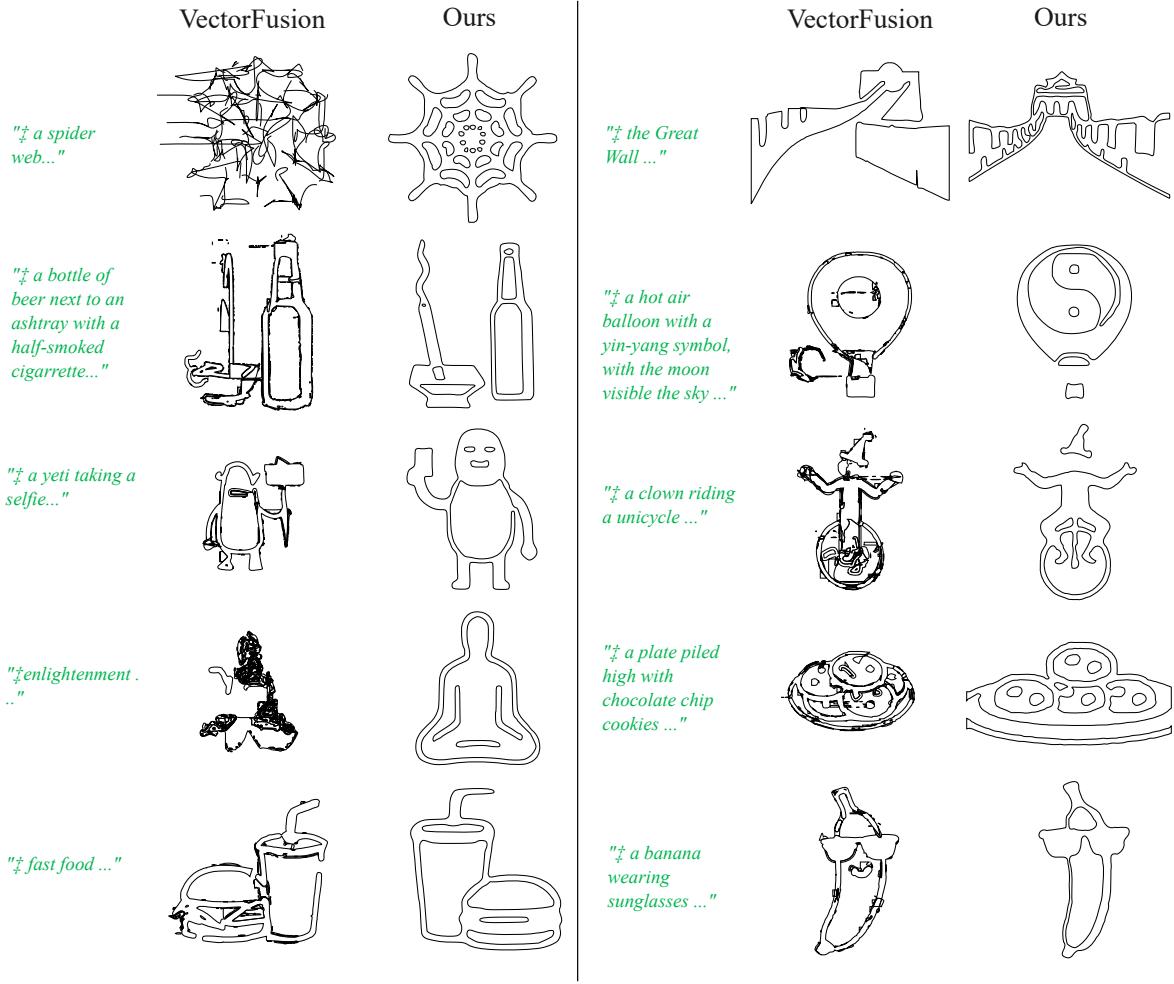


Figure 11. Text-to-Vector Graphics generation results. We compare generated SVG results for the [input text prompt](#) between NIVeL (ours) vs VectorFusion . Our vector results contain much cleaner shape geometry across diverse topology while VectorFusion’s SVGs contain redundant, degenerate curves, and self-intersecting shapes. †: "Line drawing of ... black and white"

cigarette, minimal 2d line drawing, on a white background, black and white"

"Line drawing of A Japanese woodblock print of one cat, minimal 2d line drawing, on a white background, black and white"

"Line drawing of A torii gate, minimal 2d line drawing, on a white background, black and white"

"Line drawing of an elephant, minimal 2d line drawing, on a white background, black and white"

"Line drawing of A spaceship flying in a starry sky, minimal 2d line drawing, on a white background, black and white"

"Line drawing of Enlightenment, minimal 2d line drawing, on a white background, black and white"

"Line drawing of A 3D rendering of a temple, minimal 2d line drawing, on a white background, black and white"

"Line drawing of a fire-breathing dragon, minimal 2d line drawing, on a white background, black and white"

"Line drawing of A realistic photograph of a cat, minimal 2d line drawing, on a white background, black and white"

"Line drawing of a tree, minimal 2d line drawing, on a white background, black and white"

"Line drawing of a hot air balloon with a yin-yang symbol, with the moon visible in the daytime sky, minimal 2d line drawing, on a white background, black and white"

"Line drawing of A 3D wireframe model of a cat, minimal 2d line drawing, on a white background, black and white"

"Line drawing of Hashtag, minimal 2d line drawing, on a white background, black and white"

"Line drawing of Yeti taking a selfie, minimal 2d line drawing, on a white background, black and white"

"Line drawing of A dragon-cat hybrid, minimal 2d line drawing, on a white background, black and white"

"Line drawing of a tall horse next to a red car, minimal 2d line drawing, on a white background, black and white"

"Line drawing of Underwater Submarine, minimal 2d line drawing, on a white background, black and white"

"Line drawing of A drawing of a cat, minimal 2d line drawing, on a white

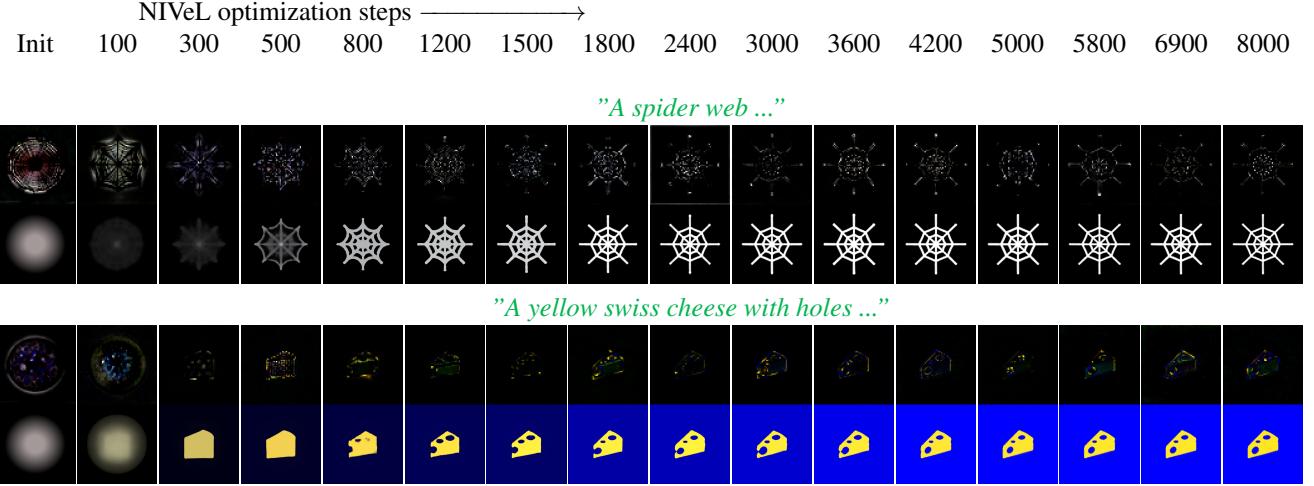


Figure 12. We visualize the SDS gradients in rgb color space (1st row) and generated results (2nd row) during optimization for two prompts. Initialization (Init) is the leftmost column. Columns on its right show the generated result and gradients for the i^{th} optimization step. The number of layers is $L = 1$ here.

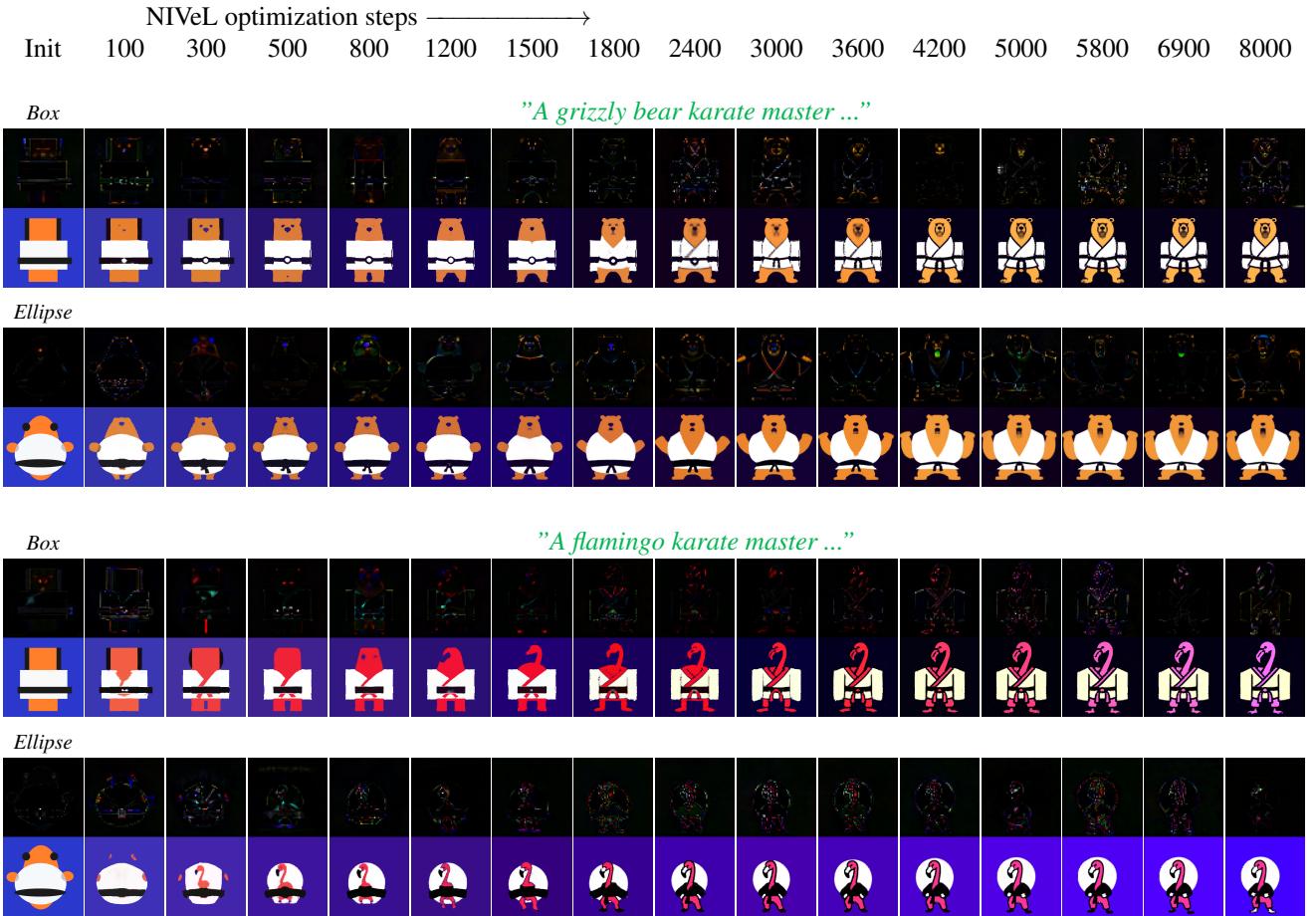


Figure 13. We visualize the SDS gradients in rgb color space (1st row) and generated results (2nd row) during optimization for two prompts. Initialization (Init) is the leftmost column. Columns on its right show the generated result and gradients for the i^{th} optimization step. The number of layers is $L = 3$ here.

Question 6 out of 20

Look at the images below.



Above you see two graphics illustration that were created based on the instruction: "generate a tiger karate master". Which one better matches the text in the instruction?

- LEFT
- RIGHT
- can't tell - BOTH match the text equally well.
- can't tell - NONE match the text well.

NEXT

Figure 14. Screenshot of an example question used in our MTurk questionnaires.

background, black and white”

“Line drawing of a photograph of a fiddle next to a basketball on a ping pong table, minimal 2d line drawing, on a white background, black and white”

“Line drawing of Forest Temple as 3D rendered in Unreal Engine, minimal 2d line drawing, on a white background, black and white”

“Line drawing of Horse eating a cupcake, minimal 2d line drawing, on a white background, black and white”

“Line drawing of A realistic painting of a sailboat, minimal 2d line drawing, on a white background, black and white”

“Line drawing of the Great Wall, minimal 2d line drawing, on a white background, black and white”

“Line drawing of a boat, minimal 2d line drawing, on a white background, black and white”

“Line drawing of A watercolor painting of a cat, minimal 2d line drawing, on a white background, black and white”

“Line drawing of The space between infinity, minimal 2d line drawing, on a white background, black and white”

“Line drawing of a basketball to the left of two soccer balls on a gravel driveway, minimal 2d line drawing, on a white background, black and white”

“Line drawing of The Eiffel Tower, minimal 2d line drawing, on a white background, black and white”

“Line drawing of A painting of a starry night sky, minimal 2d line drawing, on a white background, black and white”

“Line drawing of Translation, minimal 2d line drawing, on a white background, black and white”

“Line drawing of a triangle, minimal 2d line drawing, on a white background, black and white”

“Line drawing of a circle, minimal 2d line drawing, on a white background, black and white”

“Line drawing of a dragon breathing fire, minimal 2d line drawing, on a white background, black and white”

“Line drawing of a group of squirrels rowing crew, minimal 2d line drawing, on a white background, black and white”

“Line drawing of a fox and a hare tangoing together, minimal 2d line drawing, on a white background, black and white”

“Line drawing of a plate piled high with chocolate chip cookies, minimal 2d line drawing, on a white background, black and white”

“Line drawing of a walrus smoking a pipe, minimal 2d line drawing, on a white background, black and white”

“Line drawing of a stork playing a violin, minimal 2d line drawing, on a white background, black and white”

“Line drawing of a match stick on fire, minimal 2d line drawing, on a white background, black and white”

“Line drawing of a friendship, minimal 2d line drawing, on a white background, black and white”

“Line drawing of a banana with sun glasses, minimal 2d line drawing, on a white background, black and white”

“Line drawing of a clock with dials, minimal 2d line drawing, on a white background, black and white”

“Line drawing of a classic wristwatch, Minimal 2D line drawing, On a white background, black and white”

“Line drawing of a vintage camera, Minimal 2D line drawing, On a white background, black and white”

“Line drawing of a coffee cup and saucer, Minimal 2D line drawing, On a white background, black and white”

“Line drawing of a Clown on a unicycle, Minimal 2D line drawing, On a white background, black and white”

“Line drawing of a Utah teapot, Minimal 2D line drawing, On a white background, black and white”

“Line drawing of a computer vision conference, Minimal 2D line drawing, On a white background, black and white”

“Line drawing of a Science conference, Minimal 2D line drawing, On a white background, black and white”

“Line drawing of a human hand, Minimal 2D line drawing, On a white background, black and white”

“Line drawing of a still life, Minimal 2D line drawing, On a white background, black and white”

“A 3D wireframe model of a cat, minimal 2D vector art, lineal color”

“a bottle of beer next to an ashtray with a half-smoked cigarette, minimal 2D vector art, lineal color”

“A human hand, minimal 2D vector art, lineal color”

“A dragon breathing fire, minimal 2D vector art, lineal color”

“A group of squirrels rowing crew, minimal 2D vector art, lineal color”

“A fox and a hare tangoing together, minimal 2D vector art, lineal color”

“A plate piled high with chocolate chip cookies, minimal 2D vector art, lineal color”

“A walrus smoking a pipe, minimal 2D vector art, lineal color”

“A stork playing a violin, minimal 2D vector art, lineal color”

“A match stick on fire, minimal 2D vector art, lineal color”

“A tiger karate master, minimal 2D vector art, lineal color”

“A squirrel dressed up like a victorian woman, lineal color”

“A baby bunny sitting on top of a stack of pancakes, minimal 2D vector art”

“A baby python sitting on top of a stack of books, minimal 2D vector art”

“A vintage camera, minimal 2D vector art, lineal color”

“A coffee cup and saucer, minimal 2D vector art, lineal color”

“A Clown on a unicycle, minimal 2D vector art, lineal color”

“A still life, minimal 2D vector art, lineal color”

"A banana with sun glasses, minimal 2D vector art, lineal color"
"A clock with dials, minimal 2D vector art, lineal color"
"A classic wristwatch, minimal 2D vector art, lineal color"
"A Utah teapot, minimal 2D vector art, lineal color"
"A computer vision conference, minimal 2D vector art, lineal color"
"A Science conference, minimal 2D vector art, lineal color"
"A friendship, minimal 2D vector art, lineal color"
"A tree, minimal 2D vector art, lineal color"
"A hot air balloon with a yin-yang symbol, with the moon visible in the daytime sky, minimal 2D vector art, lineal color"
"Yeti taking a selfie, minimal 2D vector art, lineal color"
"A dragon-cat hybrid, minimal 2D vector art, lineal color"
"A spider web, minimal 2D vector art, lineal color"
"Underwater Submarine, minimal 2D vector art, lineal color"
"A boiling water on a fire stove, minimal 2D vector art, lineal color"
"A photograph of a fiddle next to a basketball on a ping pong table, minimal 2D vector art, lineal color"
"Forest Temple as 3D rendered in Unreal Engine, minimal 2D vector art, lineal color"
"Horse eating a cupcake, minimal 2D vector art, lineal color"
"A sailboat, minimal 2D vector art, lineal color"
"the Great Wall, minimal 2D vector art, lineal color"
"A boat, minimal 2D vector art, lineal color"
"A fluid simulation, minimal 2D vector art, lineal color"
"The space between infinity, minimal 2D vector art, lineal color"
"A basketball to the left of two soccer balls on a gravel driveway, minimal 2D vector art, lineal color"
"A triangle, minimal 2D vector art, lineal color"
"A circle, minimal 2D vector art, lineal color"
"A Japanese woodblock print of one cat, minimal 2D vector art, lineal color"
"A torii gate, minimal 2D vector art, lineal color"
"An elephant, minimal 2D vector art, lineal color"
"A spaceship flying in a starry sky, minimal 2D vector art, lineal color"
"Enlightenment, minimal 2D vector art, lineal color"
"Third eye, minimal 2D vector art, lineal color"
"A baby penguin, minimal 2D vector art, lineal color"
"A ladder, minimal 2D vector art, lineal color"
"A crown, minimal 2D vector art, lineal color"
"A cat as 3D rendered in Unreal Engine, minimal 2D vector art, lineal color"
"Fast Food, minimal 2D vector art, lineal color"
"Happiness, minimal 2D vector art, lineal color"
"Family vacation to Walt Disney World, minimal 2D vector art, lineal color"
"Line drawing of A 3D wireframe model of a Monkey, 2d vector art line drawing, black and white"
"Line drawing of a bottle of beer next to an ashtray with a half-smoked cigarette, 2d vector art line drawing, black and white"
"Line drawing of A human hand showing the peace sign, 2d vector art line drawing, black and white"
"Line drawing of A plate piled high with chocolate chip cookies, 2d vector art line drawing, black and white"
"Line drawing of A flamingo playing a cello, 2d vector art line drawing, black and white"
"Line drawing of A leopard karate master, 2d vector art line drawing, black and white"
"Line drawing of a Lizard dressed up like a victorian woman, lineal color"
"Line drawing of A vintage telephone, 2d vector art line drawing, black and white"
"Line drawing of An orange with sun glasses, 2d vector art line drawing, black and white"
"Line drawing of A duck taking a selfie, 2d vector art line drawing, black and white"
"Line drawing of A dragon-corgi hybrid, 2d vector art line drawing, black and white"
"Line drawing of Horse eating a hotdog, 2d vector art line drawing, black and white"
"Line drawing of An elephant surfing, 2d vector art line drawing, black and white"
"Line drawing of A plate of healthy fast Food, 2d vector art line drawing, black and white"
"A 3D wireframe model of a Monkey, minimal 2D vector art, lineal color"
"a bottle of beer next to an ashtray with a half-smoked cigarette, lineal color"
"A human hand showing the peace sign, minimal 2D vector art, lineal color"
"A dragon breathing fire, minimal 2D vector art, lineal color"
"A group of squirrels rowing crew, minimal 2D vector art, lineal color"
"A fox and a hare tangoing together, minimal 2D vector art, lineal color"
"A plate piled high with chocolate chip cookies, minimal 2D vector art, lineal color"
"A walrus smoking a pipe, minimal 2D vector art, lineal color"
"A flamingo playing a cello, minimal 2D vector art, lineal color"
"A match stick on fire, minimal 2D vector art, lineal color"
"A leopard karate master, minimal 2D vector art, lineal color"
"a Lizard dressed up like a victorian woman, minimal 2D vector art, lineal color"
"A baby python sitting on top of a stack of books, minimal 2D vector art, lineal color"
"A vintage telephone, minimal 2D vector art, lineal color"
"A coffee cup and saucer, minimal 2D vector art, lineal color"
"A Clown juggling balls, minimal 2D vector art, lineal color"
"A still life painting, minimal 2D vector art, lineal color"
"An orange with sun glasses, minimal 2D vector art, lineal color"
"A symbol of friendship, minimal 2D vector art, lineal color"
"A banyan tree, minimal 2D vector art, lineal color"
"A duck taking a selfie, minimal 2D vector art, lineal color"
"A dragon-corgi hybrid, minimal 2D vector art, lineal color"
"A spider web, minimal 2D vector art, lineal color"
"A Temple as 3D rendered in Unreal Engine, minimal 2D vector art, lineal color"
"Horse eating a hotdog, minimal 2D vector art, lineal color"
"A luxury boat, minimal 2D vector art, lineal color"
"A fluid simulation, minimal 2D vector art, lineal color"

"The space between infinity, minimal 2D vector art, lineal color"

"A torii gate, minimal 2D vector art, lineal color"

"An elephant surfing, minimal 2D vector art, lineal color"

"A spaceship flying in a starry sky, minimal 2D vector art, lineal color"

"Enlightenment, minimal 2D vector art, lineal color"

"Third eye, minimal 2D vector art, lineal color"

"A baby penguin and a polar bear taking an impossible selfie, minimal 2D vector art, lineal color"

"A plate of healthy fast Food, minimal 2D vector art, lineal color"