IconShop: Text-Guided Vector Icon Synthesis with Autoregressive Transformers

RONGHUAN WU, City University of Hong Kong, China
WANCHAO SU, City University of Hong Kong, China and Monash University, Australia
KEDE MA, City University of Hong Kong, China
JING LIAO*, City University of Hong Kong, China

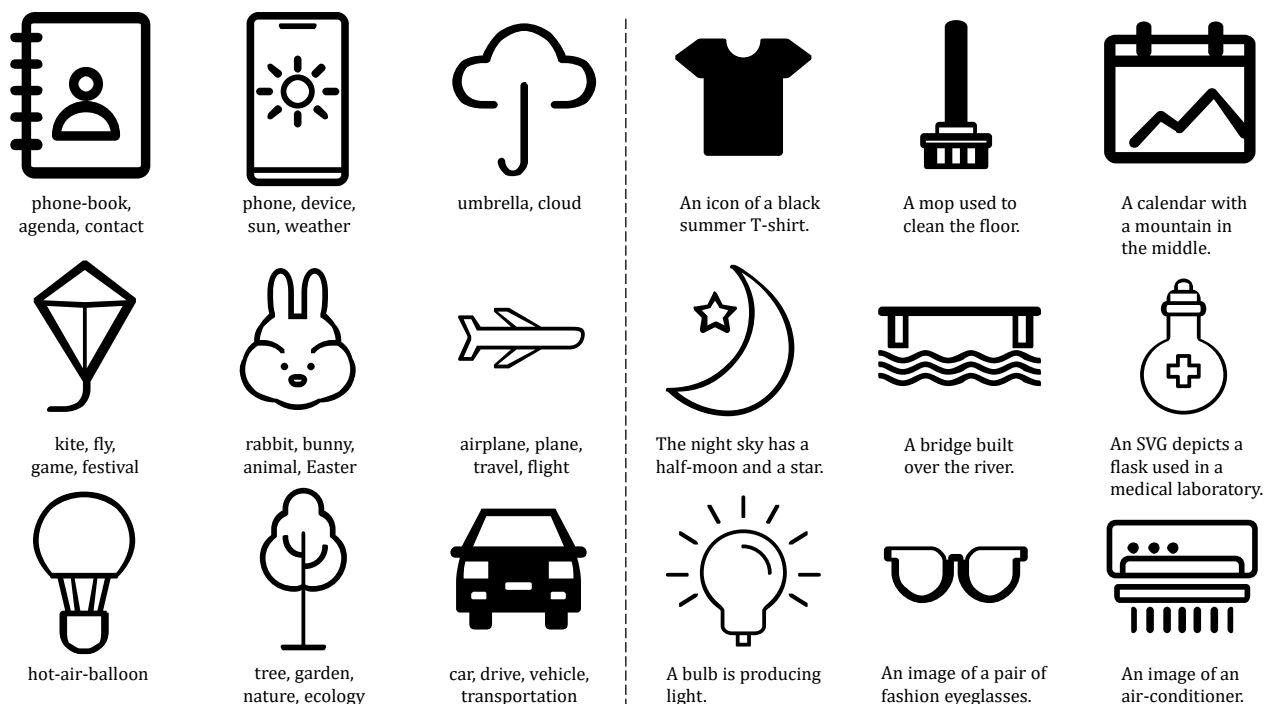|		
phone-book, agenda, contact	phone, device, sun, weather	umbrella, cloud
kite, fly, game, festival	rabbit, bunny, animal, Easter	airplane, plane, travel, flight
hot-air-balloon	tree, garden, nature, ecology	car, drive, vehicle, transportation
An icon of a black summer T-shirt.	A mop used to clean the floor.	A calendar with a mountain in the middle.
The night sky has a half-moon and a star.	A bridge built over the river.	An SVG depicts a flask used in a medical laboratory.
A bulb is producing light.	An image of a pair of fashion eyeglasses.	An image of an air-conditioner.

Fig. 1. Vector icons generated with text prompts. The proposed IconShop supports vector icon synthesis from keywords (left panel) and natural phrases and sentences (right panel).

Scalable Vector Graphics (SVG) is a popular vector image format that offers good support for interactivity and animation. Despite its appealing characteristics, creating custom SVG content can be challenging for users due to the steep learning curve required to understand SVG grammars or get familiar with professional editing software. Recent advancements in text-to-image generation have inspired researchers to explore vector graphics synthesis using either image-based methods (i.e., text → raster image → vector graphics) combining text-to-image generation models with image vectorization, or language-based methods (i.e., text → vector graphics script) through pretrained large language models. Nevertheless, these methods suffer from limitations in terms of generation quality, diversity, and flexibility. In this paper, we introduce IconShop, a text-guided vector icon synthesis method using autoregressive transformers. The key to success of our approach is to sequentialize and tokenize SVG paths (and textual descriptions as guidance) into a uniquely decodable token sequence. With that, we are able to exploit the sequence learning power of autoregressive transformers, while enabling both unconditional and text-conditioned icon synthesis. Through standard training to predict the next token on a large-scale vector icon dataset accompanied by textural descriptions, the proposed IconShop consistently exhibits better icon synthesis capability than existing image-based and language-based methods both quantitatively (using the FID and CLIP scores) and qualitatively (through formal subjective user studies). Meanwhile, we observe a dramatic improvement in generation diversity, which is validated

*Corresponding Author

Authors' addresses: Ronghuan Wu, City University of Hong Kong, China, ronghwu2-c@my.cityu.edu.hk; Wanchao Su, City University of Hong Kong, China and Monash University, Australia, wanchao_su@outlook.com; Kede Ma, City University of Hong Kong, China, kede.ma@cityu.edu.hk; Jing Liao, City University of Hong Kong, China, jingliao@cityu.edu.hk.

by the objective Uniqueness and Novelty measures. More importantly, we demonstrate the flexibility of IconShop with multiple novel icon synthesis tasks, including icon editing, icon interpolation, icon semantic combination, and icon design auto-suggestion.

1 INTRODUCTION

As a form of computer graphics, vector graphics represents visual content based directly on geometric shapes (via command lines and arguments), and is widely used in scientific and artistic applications, including architecture, surveying, 3D rendering, typography, and graphic design. Compared with raster graphics, vector graphics is preferred when a high degree of geometric precision is required across arbitrary scales, among which Scalable Vector Graphics (SVG) is a popular vector graphics file format extensively employed in creative industries. Generally, creating SVG content is difficult for non-professional users. It is tedious and time-consuming to gain adequate knowledge of SVG grammars and/or master professional editing software such as Adobe Illustrator. Recently, there have been impressive successes in generating raster images from text, providing a convenient and efficient means of fulfilling users' image design intents. Thus, it is highly desirable to build a computational system that can accomplish something similar in the field of SVG, allowing accurate and flexible SVG content synthesis guided by intuitive textual descriptions.

One straightforward approach is adapting raster image generation to vector synthesis by converting the image outputs of text-to-image generation models into vector graphics using image vectorization methods. While such image-based methods (i.e., text → raster image → vector graphics) directly incorporate recent advances in text-to-image generation (e.g., Stable Diffusion [Rombach et al. 2022]) into SVG generation, their results are often unsatisfactory. This is because text-to-image models are mostly trained to generate complex natural geometric shapes and color appearances, and are less probable to reproduce SVG styles with simple geometric primitives and flat colors. Furthermore, to fit such complex raster images, vectorization methods often use jagged paths with unwanted corners and crossovers, resulting in visible and annoying artifacts.

As SVG is based on Extensible Markup Language (XML), one research avenue for synthesizing SVG content from text is to train Sequence-To-Sequence (seq2seq) models that take text prompts as input and directly produce SVG scripts as output. We refer to these as language-based methods (i.e., text → vector graphics script). Despite the conceptual simplicity, SVG involves complex grammars, and naive tokenization of SVG as a form of natural language may result in complex and lengthy token sequences, which complicate subsequent probabilistic modeling. Preliminary experiments [Bubeck et al.

2023] show that Large Language Models (LLMs) like GPT-4 [OpenAI 2023] tend to combine basic geometric shapes, such as `Circle` and `Ellipse`, to convey semantic information with relatively good text-SVG alignment. However, the synthesis results show limited complexity and diversity (see Section 4.3), which is inadequate for real-world SVG applications.

In this paper, we develop an autoregressive transformer-based method that supports accurate and flexible SVG content synthesis guided by textual descriptions. We demonstrate the feasibility of our method in the context of black-and-white SVG icons due to the availability of a large dataset and the moderate complexity of these icons, as well as their wide range of applications (e.g., website design, visualizations, and print media). Consequently, we name our system IconShop. The key to success of IconShop is to exploit the sequential nature of SVG: An SVG script is composed of a sequence of paths, which in turn consists of a sequence of drawing commands (e.g., lines and curves, see Section 3.2). We thus concatenate all SVG paths in a uniquely decodable way to form a command sequence. Since the text prompt is sequential in nature as well, it can be straightforwardly prepended to the command sequence.

We thus tokenize and mask the combined sequence in a way [Aghajanyan et al. 2022; Bavarian et al. 2022; Fried et al. 2023] that admits standard training of IconShop (to predict the next token autoregressively), while enabling conditional SVG icon synthesis on the bidirectional context (also known as the filling-in-the-middle task). This can be effectively achieved by incorporating the right context into the left context (separated by a special token), which conforms to causal (i.e., autoregressive) masking.

We train IconShop on a large-scale vector icon dataset *FIGR-8-SVG* [Clouâtre and Demers 2019], consisting of monochromatic (i.e., black-and-white) SVG icons. We conduct comprehensive evaluations of IconShop in terms of generation quality and diversity under different synthesis settings. Our experimental results show that IconShop is superior to existing image-based and language-based methods in these two aspects both quantitatively and qualitatively. The synthesized SVG icons also show reasonable faithfulness to the corresponding text prompts. Moreover, we demonstrate the flexibility of IconShop with multiple novel icon synthesis tasks, including icon editing, icon interpolation, icon semantic combination, and icon design auto-suggestion.

2 RELATED WORK

Our work is related to text-to-image generation (Section 2.1), icon generation (Section 2.2), vector graphics generation (Section 2.3), and generative transformers (Section 2.4). Here we only provide a concise review of previous work that is closely related to ours, and a comprehensive treatment of the above areas is beyond the scope of this paper.

2.1 Text-to-Image Generation

Generating images from text is a challenging task that has gained substantial attention in recent years, and has gone through three stages of development: Generative Adversarial Networks (GANs) [Kang et al. 2023; Qiao et al. 2019; Reed et al. 2016; Xu et al. 2018; Zhang et al. 2017, 2018], seq2seq models based on transformers [Chang et al.

2023; Ding et al. 2021, 2022; Ramesh et al. 2021; Yu et al. 2022], and diffusion models [Nichol et al. 2021; Rombach et al. 2022; Saharia et al. 2022]. Specifically, a GAN [Goodfellow et al. 2014] involves training two neural networks - a generator and a discriminator - to play a minmax zero-sum game. The system learns to generate new images by respecting the training data distribution and text conditions. Text-conditioned GANs [Kang et al. 2023; Qiao et al. 2019; Reed et al. 2016; Xu et al. 2018; Zhang et al. 2017, 2018] are typically limited to modeling single and multiple object classes. Scaling them up to handle complex image datasets remains very challenging due to the instability occurred in the training procedure, until very recently [Kang et al. 2023]. Seq2seq models based on transformers operate by converting (and concatenating) the input text (and image) into a sequence of tokens for predicting another sequence of tokens that corresponds to the target image [Chang et al. 2023; Ding et al. 2021, 2022; Ramesh et al. 2021; Yu et al. 2022]. Text-only and image-only self-attention and text-image cross-attention are canonical computational mechanisms in seq2seq models to capture intricate text and image dependencies. Recently, diffusion models have emerged to be the new standard in text-to-image generation overnight. Typically, an unconditional diffusion model [Ho et al. 2020] initiates its process with Gaussian noise, and iteratively eliminates it to yield a natural image. Text-guided diffusion models [Nichol et al. 2021; Rombach et al. 2022; Saharia et al. 2022] leverage the text embedding either as input or through cross-attention. Although previous work tackles text-guided visual content generation like ours, they focus primarily on raster images with fixed resolution. In contrast, we aim for a different goal - text-guided vector icon synthesis with arbitrary scaling.

2.2 Icon Generation

Icons play a pivotal role in modern design and communication, offering concise representations of objects for quick understanding. Researchers have long been engaged in the exploration of icon creation. Setlur et al. [2005, 2011] proposed a retrieval-based method to create icons for files and hyperlinks: they began by extracting contexts from text content, then retrieved images from a database that matched the context, using image processing techniques to make these images more akin to query semantics. The ICONATE system [Zhao et al. 2020] facilitates text-to-icon generation by constructing a concept map where semantic concepts are linked to compound sub-concepts for new icon generation. When a user provides a text description, the system consults the concept map, combining several existing icons to create an icon that corresponds to the input text. Such retrieval-based approaches have limitations when it comes to the variation of each icon concept. Our IconShop supports text-to-icon generation with a focus on flexible editing capabilities. In the era of deep learning, there have been advancements in GANs to synthesize more diverse icons [Chen et al. 2022; Yang et al. 2021]. For instance, Iconify [Karamatsu et al. 2020] harnesses the domain conversion capabilities of CycleGAN [Zhu et al. 2017] to transform natural images into icon-style representations. The GAN-based icon generation solutions produce rasterized icon images while our Icon-Shop synthesizes vector icons and supports various applications to facilitate user production.

2.3 Vector Graphics Generation

In the early 2000s, SVG content can be created using PERL [Probets et al. 2001] with plentiful drawing commands, but requires extensive human intervention. Bergen and Ross [2012] automated the determination of the number and type of drawing commands by evolutionary computation to match a target raster image. These work can be seen as ancestors of vector graphics generation methods based on deep neural networks for learning editable SVG representations. SketchRNN [Ha and Eck 2017] is a pioneering deep representation learning model for vector sketches based on a seq2seq Variational Auto-Encoder (VAE) [Kingma and Welling 2013]. The encoder and the decoder were implemented by a bidirectional recurrent neural network (RNN) and an autoregressive RNN, respectively. The sketches were parameterized by polylines - a sequence of points with line segments drawn between consecutive points. Lopes et al. [2019] incorporated a VAE-learned raster image representation to aid SVG font synthesis, where the feasibility was demonstrated only on glyphs with a maximum of 50 commands. Modeling the layered structure of SVG, DeepSVG [Carlier et al. 2020] leverages two transformer-based encoders in tandem to map SVG icons from commands to path-level representations, and then to a global latent representation. Two decoders were paired up for SVG icon reconstruction. Although the reconstructed shapes look reasonable, DeepSVG fails to reproduce simple geometric relationships like perpendicularity and parallelism. Inspired by DeepSVG, Aoki and Aizawa [2022] made full use of the global latent representation in every stage of the decoder for synthesizing Chinese SVG characters. Wang and Lian [2021] introduced DeepVecFont, a method that exploits dual-modality information (i.e., both vector graphics and corresponding raster images) to synthesize vector glyphs. Along with a refinement process, their approach generates visually pleasing fonts. Despite demonstrated success, the above-mentioned methods fail to support text-guided SVG generation.

One straightforward approach of text-guided SVG content synthesis is to first generate a raster image with a trained text-to-image generation model (e.g., DALL·E [Ramesh et al. 2021] and Stable Diffusion [Rombach et al. 2022]), and then vectorize it using off-the-shelf image vectorization techniques (e.g., Potrace [Selinger 2003] and LIVE [Ma et al. 2022]). Another emerging line of research is to directly optimize SVG parameters for a pretrained vision-language model as the loss function. For instance, CLIPDraw [Frans et al. 2021] adopts the CLIP [Radford et al. 2021] model as the optimization objective to measure the embedding distance between the vector-to-raster image and the input textual description. Apart from the CLIP distance, VectorFusion [Jain et al. 2022] leverages the Score Distillation Sampling (SDS) loss [Poole et al. 2022] based on a pixel-space text-to-image diffusion model. Despite different design philosophies, the vectorization-based and optimization-based methods suffer from similar limitations. First, the vision-language models as the key enablers are pretrained on raster images of complex natural scenes, and thus can hardly provide guidance in synthesizing SVG-style images with simple geometric primitives and flat colors. Second, the generated paths are often jagged and messy, failing to reproduce accurate geometric relations such as parallelism and perpendicularity. Third, the per-SVG optimization can be painfully slow, making

Table 1. Overview of SVG commands. In our implementation, we employ three simple yet expressive commands (namely, M, L, and C) to represent vector icons. In the Example column, we assume the current point is located at $(0, 0)$. For a sequence of commands, the starting position of each command is the ending position of the previous command.

Name	Symbol	Argument	Explanation	Example
Move To	M	x, y	Move the cursor to the specified point (x, y).	M 20, 10 (20, 10)
Line To	L	x, y	Draw a line segment from the current point to the specified point (x, y).	L 20, 10 (20, 10)
Cubic Bézier	C	x_1, y_1 x_2, y_2 x, y	Draw a curved path from the current point to the specified point (x, y) using two control points (x_1, y_1) and (x_2, y_2).	C 13, 2 7, 8 20, 10 (13, 2) (7, 8) (20, 10)

it impractical for real-time applications. In contrast, our proposed system, IconShop, does not suffer from any of the above-mentioned problems. Once trained, IconShop can perform text-guided vector icon synthesis automatically and efficiently.

2.4 Transformers as Generative Models

Owing to its inherent capability to capture long-term dependencies and support parallel computing, Transformers [Vaswani et al. 2017] have emerged as a powerful class of generative models for producing a wide variety of outputs, ranging from natural languages [Brown et al. 2020; Radford et al. 2019; Raffel et al. 2020], audios [Huang et al. 2018; Li et al. 2019; Valle et al. 2020], and raster images [Chen et al. 2020; Esser et al. 2021]. Transformers can be made non-autoregressive and autoregressive. The non-autoregressive instantiation [Chang et al. 2023, 2022; Ding et al. 2022; Zhang et al. 2021] suggests to leverage the bidirectional context using BERT-like [Devlin et al. 2018] bidirectional Transformers for its sampling efficiency. The autoregressive instantiation [Ding et al. 2021; Ramesh et al. 2021; Yu et al. 2022] emphasizes the importance of learning to predict the next token in a causal way. Together with scaling, it unlocks the emerging abilities of LLMs. Inspired by [Aghajanyan et al. 2022; Bavarian et al. 2022], we unify non-autoregressive and autoregressive modeling of vector icons for various synthesis tasks.

3 ICONSHOP

In this section, we first briefly introduce autoregressive models (Section 3.1), and describe the SVG commands of vector icons, followed by our tokenization strategy (Section 3.2). We then describe the "causal" masking strategy, which enables our autoregressive model to perform the filling-in-the-middle task (Section 3.3). We next elaborate our model architecture (Section 3.4) and, finally, present the training objectives (Section 3.5).

3.1 Preliminaries on Autoregressive Models

An autoregressive model specifies that the current state depends only on its previous states. In probabilistic terms, this corresponds to the chain rule of probability:

$$p(S) = \prod_{n=1}^{N} p(S_n | S_1, \ldots, S_{n-1}), \qquad (1)$$

in which we factorize the joint probability of a sequence of random variables $S = (S_1, \ldots, S_N)$ into a product of conditional probabilities. At the n-th instance, autoregressive models take the values of previous $n - 1$ random variables (or the most recent ones if a Markov window is applied) as input, and compute the conditional probability distribution of S_n, from which we are able to draw a sample as its prediction. Here we resort to autoregressive models for SVG icon synthesis because it fits naturally in the sequential nature of SVG and textural descriptions.

3.2 SVG Representation and Tokenization

SVG offers a range of features and syntax options, allowing users to create their original work with great flexibility. For example, the Rect command creates a rectangular shape controlled by the starting point, width, and height arguments, like <Rect x="80" y="90" width="100" height="100"/>. The Transform attribute applies an affine transformation to an existing shape, like <Rect Transform="rotate (-10 50 100)" x="80" y="90" width="100" height="100"/>. If we try to come up with a universal data structure to represent all possible SVG commands and attributes, such a data structure would become highly complex, which may hinder the probabilistic modeling of SVG icons. To bypass this issue, we choose to limit the number of commands and attributes, while maintaining their expressiveness to capture the essence of SVG icons. In other words, we seek a compact SVG representation that makes the probabilistic modeling easier.

Inspired by DeepSVG [Carlier et al. 2020], we simplify every SVG icon by removing all attributes and using only three basic commands: Move To, Line To, and Cubic Bézier (see Table 1 for explanations and examples). Other complex commands (e.g., Rect, Circle, and Ellipse) can be approximated by combinations of these basic commands with negligible visual differences. For example, we can use
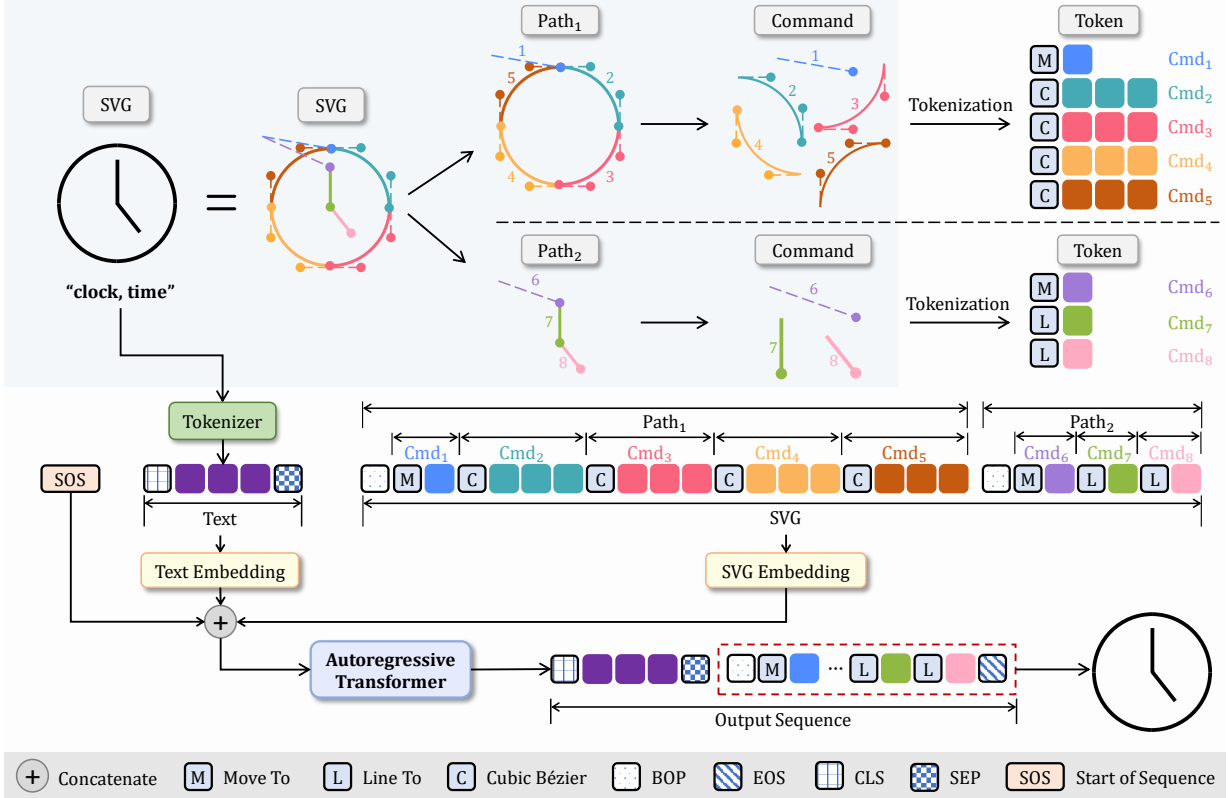
Fig. 2. The system diagram of IconShop. In this example, the clock **icon** has two paths, each comprising three types of basic commands (refer to Table 1). For Path$_1$, there is one M (Move To) command and four Bézier curves. For Path$_2$, there is one M command and two line segments. To tokenize these paths, we first concatenate the commands of the two paths to form a single sequence. We then convert each command's 2D location argument (x, y) into 1D using the formula $x \times w + y$, where w is the default SVG image width. We use a pretrained text encoder to tokenize and embed the **text** "clock, time", where <CLS> and <SEP> mark the beginning and the end of the text input. We then concatenate the embedded text and SVG sequences, and add an <SOS> at the beginning. This concatenated sequence is sent to an autoregressive transformer for joint probability modeling.

four line segments to construct a Rect, and concatenate four Bézier curves to form a Circle. An SVG text script G, after the simplification, contains M *paths*, $G = \{P_i\}_{i=1}^{M}$, where P_i is the i-th path, and each path P_i in turn consists of N_i *commands*, $P_i = \{C_i^j\}_{j=1}^{N_i}$, where C_i^j is the j-th command in the i-th path. A command, $C_i^j = (U_i^j, V_i^j)$, contains its type $U_i^j \in \{M, L, C\}$ and the corresponding location argument V_i^j.

The next step is to convert the SVG script into a discrete sequence of tokens, which are subject to autoregressive modeling. We introduce an intuitive SVG tokenization approach, which is composed of four major steps. First, flatten the layered structure of the SVG script by concatenating commands from different paths to form a single command sequence. To uniquely decode the flattened command sequence back to the layered representation, we prepend a special token, <BOP> (i.e., *beginning-of-path*), before the first command of each path. Second, assign distinct tokens to each command type (i.e., M, L, C). Third, map the 2D location argument to 1D using row-major order, which roughly halves the length of the token sequence. For example, suppose the default width of an SVG image is w, we

transform a 2D location argument (x, y) to a 1D argument using the formula $x \times w + y$. Fourth, append a special token, <EOS> (i.e., *end-of-SVG*), at the end of the sequence that indicates the completion of an SVG icon sequence. An example of a sequence created using the above steps is shown in Figure 2.

3.3 Masking Scheme

Autoregressive models have been shown to be effective in generating token sequences from scratch, but they are constrained to doing so in a causal direction (i.e., from left to right). This constraint hinders SVG synthesis performance in tasks such as icon editing, where we need to fill in missing content based on the bidirectional context. Some methods have been developed to expand the ability of autoregressive models to perform the filling-in-the-middle task by massaging training data without modifying the model architecture. For example, CM3 [Aghajanyan et al. 2022] and InCoder [Fried et al. 2023] use a *"causal" masking* strategy that randomly selects several chunks in the input sequence, and move them to the end. In [Bavarian et al. 2022], a similar strategy was implemented to learn to fill in the middle without sacrificing the original causal generation

capability. In training IconShop, we incorporate a similar "casual" masking strategy to unify non-autoregressive and autoregressive modeling of SVG token sequences.

For a given input sequence $S^{(0)}$, we first select a random chunk called *span*, based on which we split the sequence into three parts, [Left : Span : Right], where ":" represents concatenation. We replace the span with a special <Mask> token to obtain a new sequence $S^{(1)}$ = [Left : <Mask> : Right]. Next, we add the same <Mask> token to the beginning of the span and add a <EOM> token (i.e., *end-of-mask*) to the end of the span to create a new sequence $S^{(2)}$ = [<Mask> : Span : <EOM>]. Finally, we concatenate $S^{(1)}$ and $S^{(2)}$ to form

$$S = [\text{Left} : \text{<Mask>} : \text{Right} : \text{<Mask>} : \text{Span} : \text{<EOM>}], \quad (2)$$

which is sent to the model for probabilistic modeling. The masked sequence S conveys the following information: 1) the first <Mask> token indicates the original position of the span, 2) the second <Mask> token denotes the beginning of the span, and 3) the <EOM> token marks the end of the span. During training, we randomly apply this masking strategy to 50% of the training data, while leaving the remaining 50% unchanged. We exclude the <Mask> token from the cross-entropy loss calculation to discourage its generation during inference.

We now explain how this masking technique enables autoregressive models to perform the filling-in-the-middle (i.e., non-autoregressive) generation without modifying the architecture. Suppose we have a token sequence [Left : Right], and want to fill in the middle chunk between Left and Right. As in Eq. (2), we add two <Mask> tokens to the sequence to create [Left : <Mask> : Right : <Mask>], and send it to the model for seq2seq generation until the <EOM> token occurs, giving rise to the sequence [Left : <Mask> : Right : <Mask> : Span : <EOM>]. After that, we move the predicted Span back to its original position, i.e., the position of the first <Mask> token, to obtain the final output [Left : Span : Right]. Since the model leverages both Left and Right contexts to fill in the middle Span chunk, we achieve non-autoregressive modeling through autoregressive prediction, and thus unify these two.

3.4 Model Architecture

We employ the Transformer decoder [Vaswani et al. 2017] to implement our autoregressive model, as it effectively captures the long-range interdependencies among various tokens that constitute a vector icon sequence. Specifically, the model consists of three modules: an SVG embedding module to encode the SVG sequence, a text embedding module to encode the text sequence, and a transformer (decoder) module to exploit text-SVG correlations, and learn the joint probability distribution of the combined token sequence, sampling from which produces novel sequences that are not present in the training set.

3.4.1 SVG Embedding Module. As discussed previously, an SVG sequence is represented by six distinct categories of tokens: 1) Command type, 2) 1D location argument, 3) Beginning-of-path token <BOP>, 4) End-of-SVG token <EOS>, 5) Mask token <Mask>, and 6) End-of-mask token <EOM>. By default, each icon is constrained within a 100×100 bounding box, resulting in 100^2 possible values for

the 1D location argument. Thus, a one-hot vector e with a dimension of $10,007(= 3 + 100^2 + 1 + 1 + 1 + 1)$ suffices to represent all possible token cases. We then use a learnable embedding matrix $W \in \mathbb{R}^{D \times 10007}$ to transform the one-hot vector into an embedding vector of size D. We incorporate two extra learnable matrices $W^x, W^y \in \mathbb{R}^{D \times 100}$ to augment the location information as suggested in [Xu et al. 2022]:

$$v_i \leftarrow We_i + W^x e_i^x + W^y e_i^y, \quad (3)$$

where $e_i \in \mathbb{R}^{10007 \times 1}$ is the one-hot vector of the i-th token, and $e_i^x, e_i^y \in \mathbb{R}^{100 \times 1}$ are one-hot encodings of the 2D coordinates, respectively.

3.4.2 Text Embedding Module. LLMs trained on a large corpus of textual data have the ability to capture intricate word interrelationships, including synonymy and antonymy, as suggested in [Saharia et al. 2022]. Here we make use of the tokenization and word embedding layers from a pretrained BERT [Turc et al. 2019] model, and fix them to tokenize and embed textual inputs. The tokenizer adds a <CLS> token to the beginning of the text, and appends an <SEP> token at the end of the text, indicating the start and the end of the text sequence, respectively.

3.4.3 Transformer Module. Our autoregressive transformer model consists of a stack of 12 identical layers. Each layer is a standard transformer decoder block, comprising (masked) multi-head attention, layer normalization, and feed-forward layers, all interconnected via residual connections. The autoregressive transformer ultimately produces a D-dimensional vector at the n-th token position, which is conditioned on its preceding $n - 1$ tokens. A linear layer followed by the softmax function is applied to obtain the probabilities of all possible tokens at the n-th position.

3.5 Training Objective

Since the text descriptions and SVG scripts in the training set have varying lengths, we pad both the text token sequence S^{text} and the icon token sequence S^{icon} with zeros to a fixed length (50 for the text and 512 for the icon in our implementation). We then concatenate them to obtain the target sequence $S = [S^{\text{text}} : S^{\text{icon}}]$.

The autoregressive transformer is trained to predict the next token based on previous tokens. To prepare the input sequence S^{in}, we remove the last token of S (i.e., <EOS>), and add an <SOS> token at the start. This essentially shifts S to the right by one position, which enables initial autoregressive prediction with an empty context. The transformer outputs a token sequence $\hat{S} = [\hat{S}^{\text{text}} : \hat{S}^{\text{icon}}]$ sequentially. Our objective is to minimize the cross-entropy loss between the target and output tokens at each position, and then combine text and icon losses with a weighted sum [Ramesh et al. 2021]:

$$\ell^{\text{text}} = \text{CE}(S^{\text{text}}, \hat{S}^{\text{text}}),$$
$$\ell^{\text{icon}} = \text{CE}(S^{\text{icon}}, \hat{S}^{\text{icon}}), \quad (4)$$
$$\ell^{\text{total}} = \ell^{\text{text}} + \lambda \ell^{\text{icon}},$$

where CE() is the standard cross-entropy function, and $\lambda = 7.0$ is the weighting to control the relative importance between the text and icon reconstruction.
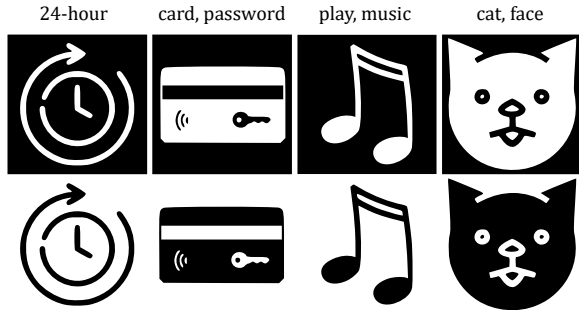
Fig. 3. Monochromatic icon samples from the *FIGR-8-SVG* dataset (1st row). Each icon is associated with several discrete keywords as textual descriptions. We remove the black box by intensity inversion to improve the visual quality (2nd row). Sample icons are from ©Noun Project.
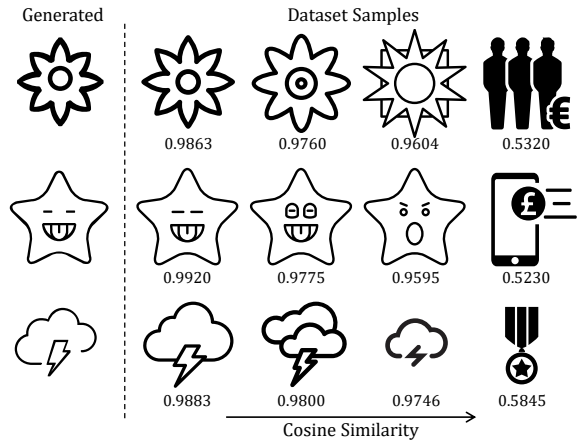

Fig. 4. We use the CLIP image encoder to extract image features, and calculate the cosine similarity between the generated icons and the samples in the dataset. As visually distinct icons may have high cosine similarity scores due to the simple black strokes against the white background, we set a relatively high threshold of 0.98 to determine whether two icons are identical when computing the "Uniqueness" and "Novelty" scores. Sample icons are from ©Noun Project.

4 EXPERIMENTS

In this section, we first present in detail the data processing procedure for the icon dataset with textual descriptions (Section 4.1). We then introduce an ablation study to validate the efficacy of the model architecture under both conditional and unconditional generation settings (Section 4.2). Lastly, we compare the proposed Icon-Shop with alternative solutions, demonstrating that IconShop yields higher-quality results (Section 4.3).

4.1 Data Preparation

4.1.1 SVG Dataset. We use the *FIGR-8-SVG* dataset [Clouâtre and Demers 2019] that consists of 1.5 million monochromatic (black-and-white) vector icons. Typically, the first step for SVG data processing is transforming icons with varied grammars into standardized representations. Fortunately, in the *FIGR-8-SVG* dataset, all icons have been converted to a uniform representation with discretized arguments. We show several icon examples from the dataset in the first row of Figure 3. We further enhance the visual attractiveness of each icon by removing the outer black box by intensity inversion. The corresponding enhanced icons are shown in the second row. After the SVG tokenization described in Section 3.2, we set the maximum length of an icon sequence to 512, filtering out those with longer lengths. This results in about 1.1 million samples, among which, we select 300,000 samples for model training and experimentation. We partition these samples into 90% for training, 5% for validation, and 5% for testing.

4.1.2 Text Input. In the *FIGR-8-SVG* dataset, every vector icon is annotated with discrete keywords, such as "cat,face". Training IconShop only with keywords would constrain its capacity to generate icons from natural language phrases and sentences. Inspired by InstructPix2Pix [Brooks et al. 2022], which fine-tunes GPT-3 [Brown et al. 2020] to produce editing instructions and captions, we use LLMs (ChatGPT[1] in particular) to expand these keywords into natural language phrases and sentences. The prompt given to ChatGPT is "Write the simplest sentence from keywords: #{keywords}. Do not add additional facts".

---

[1]https://openai.com/blog/chatgpt

4.1.3 Implementation Details. We implement IconShop using *PyTorch*. The training process employs the Adam optimizer [Kingma and Ba 2014] with a learning rate of 0.0006, along with linear warm-up, decay, and gradient clipping. The dropout rate is set to 0.1. The BERT version is google_bert_uncased_L-12_H-512_A-8. Sampled and shuffled discrete keywords (i.e., a subset of the original keywords), natural language phrases and sequences (generated by ChatGPT), and blank text are, respectively, trained with ratios of 60%, 30%, and 10%, with a total minibatch size of 192. We train IconShop for 300 epochs. We use an NVIDIA A100 GPU in the following experiments to test IconShop, which takes 1.38 seconds to generate an SVG icon sequence on average. We generate 32 icons for each text description, and use ImageReward [Xu et al. 2023] (a text-image scoring model that is calibrated against human preferences) automatic result selection, which is a standard practice in text-to-image generation. Please refer to the supplementary material for a collection of uncurated results.

4.1.4 Metrics. To evaluate the quality of the generated SVG icons, we use the Fréchet Inception Distance (FID) [Heusel et al. 2017], which measures the distance between the image features of synthesized and "ground-truth" SVG icons from *FIGR-8-SVG*. Specifically, we obtain the image features of rendered and rasterized SVG icons using the CLIP image encoder [Radford et al. 2021]. We also compute the CLIP score to measure text-SVG alignment (i.e., the semantic similarity between the text input and the visual icon output). Additionally, following SkexGen [Xu et al. 2022], we calculate the "Novelty" and the "Uniqueness" scores, which stand for the proportion of generated data absent from a target set (i.e., the training set from *FiGR-8-SVG*) and occurring only once among all generated samples, respectively. We generate 20,000 icons unconditionally and 7,000 icons with textual guidance. The cosine similarity between
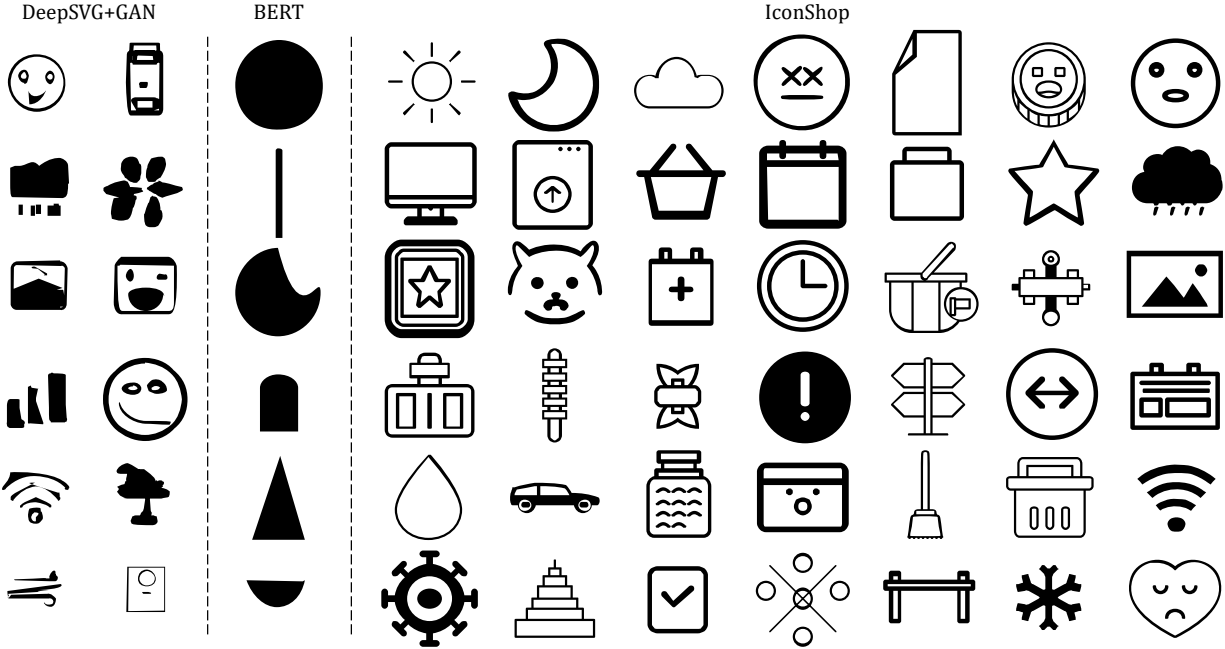
Fig. 5. Icons randomly generated by DeepSVG+GAN, BERT and IconShop, respectively. Our approach creates icons with form consistency, high-precision of recognizability, geometry simplicity, and good composition. Icons produced by DeepSVG+GAN do not meet such desired properties, while BERT only synthesizes basic geometric shapes with essentially no semantics.

the CLIP features is used to determine whether the two icons are identical with a threshold of 0.98. Figure 4 gives an intuitive visual comparison of different similarity values.

4.2 Ablation on Different Network Architectures

In this subsection, we conduct two ablation studies to illustrate the effects of two key components of IconShop on the synthesis performance. The first study shows the importance of exploiting the sequential nature of the SVG icons in generation by comparing our seq2seq model to a GAN model. The second study demonstrates the necessity of our autoregressive model augmented with the "causal" masking scheme by comparing it to a non-autoregressive training strategy (i.e., BERT [Devlin et al. 2018]).

4.2.1 Seq2seq versus Layered Modeling. As previously mentioned, an SVG file is layered in structure, consisting of higher-layer paths and lower-layer commands. Here we use DeepSVG+GAN as a baseline to testify our choice of the seq2seq generation mechanism.

The original DeepSVG reconstructs an SVG icon by first obtaining path-level representations and then aggregating them to a global representation. To make a fair comparison between IconShop and DeepSVG, we need to enable the original DeepSVG to generate new icons and support text-guided generation. To achieve these goals, we retrain DeepSVG for SVG icon reconstruction on our dataset. We then train a conditional GAN, which takes the textual feature as input and predicts the latent representation precomputed from the encoder of DeepSVG as output, which can be straightforwardly decoded into an SVG icon by passing through the decoder of DeepSVG. In addition to text-guided generation, we also enable the GAN to

generate icons unconditionally by replacing the text feature with random noise 10% of the time during training.

We compare both *random generation* and *text-guided generation* performance. For random generation, we produce 20,000 icons for each method. For text-guided generation, we choose both discrete keywords and natural phrases/sentences commonly used in design scenarios as text inputs, and generate 7,000 icons for each model. As shown in Figures 5 and 6, IconShop is capable of generating visually recognizable and eye-catching icons while preserving salient geometric relationships, such as perpendicularity, parallelism, and symmetry. We believe such consistent high-quality generation results arise because we prioritize the sequential modeling of SVG icons. In contrast, DeepSVG+GAN presents visually worse results than ours, which we attribute to its averaging operation over commands and paths, resulting in a loss of geometric details in the generated icons. Please refer to the supplementary material for more qualitative results with diverse prompts.

We also quantitatively evaluate the random generation and text-guided generation results in Tables 2a and 2b. We find that Icon-Shop synthesizes results with significantly lower FID scores in both random and text-guided generation tasks, which provides a strong indication of its superior synthesis capability. Regarding Uniqueness and Novelty as measures of generation diversity, IconShop performs comparably well to DeepSVG+GAN. It is crucial to bear in mind that these two quantitative scores must be considered along with icon quality. Merely achieving high scores does not necessarily imply good results. Specifically, the high Uniqueness and Novelty values achieved by DeepSVG+GAN are largely attributed to the noticeable
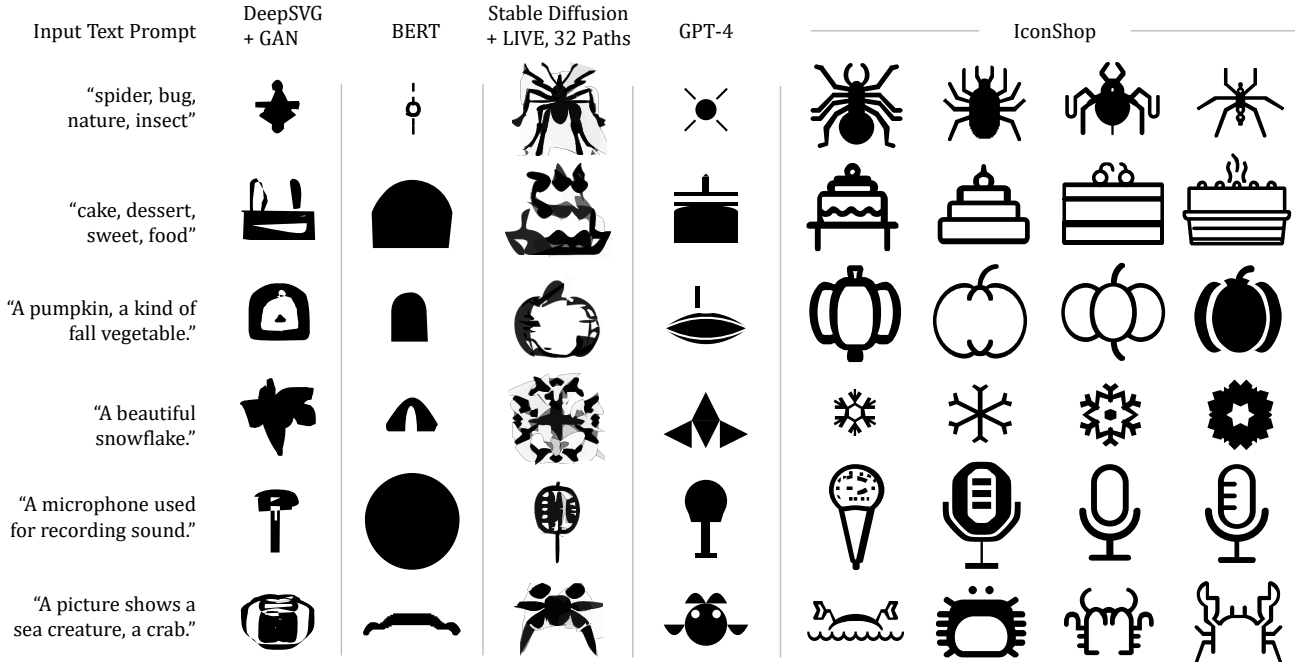
Fig. 6. We evaluate the text-guided icon synthesis of IconShop against several other methods, including DeepSVG+GAN, BERT, Stable Diffusion+LIVE, and GPT-4. The results show that Stable Diffusion+LIVE provides poor vectorization results, in terms of geometrical and semantic precision of icon forms, while GPT-4 is only able to combine basic geometric shapes with limited recognizability. IconShop clearly outperforms the competing methods in terms of icon generation quality.

Table 2. We evaluate IconShop through random generation and text-guided generation tasks. We use the FID score, with features extracted from the CLIP image encoder, to assess generation quality. We also compute the percentage of unique and novel icons as measures of generation diversity. For text-guided generation, we employ the CLIP score to show the semantic alignment between the text input and the generated icons.

	FID ↓	Uniqueness% ↑	Novelty% ↑
DeepSVG+GAN	11.95	**98.72**	**99.22**
BERT	43.61	2.06	19.90
IconShop	**6.08**	78.77	85.10

(a) Random Generation

	FID ↓	Uniqueness% ↑	Novelty% ↑	CLIP Score ↑
DeepSVG+GAN	12.01	**97.59**	**99.01**	21.78
BERT	35.10	14.41	50.30	22.03
IconShop	**4.65**	68.29	68.60	**25.74**

(b) Text-Guided Generation

visual distortions (e.g., jittering curves) it suffers from rather than novel instantiations of the same object/concept, as evidenced in Figure 5. We regard this as "fake diversity" if it is solely interpreted without being conditioned on acceptable generation quality. Moreover, IconShop obtains the highest CLIP score in the text-guided generation task, signaling the outstanding ability to produce icons that accurately reflect the text semantics.

4.2.2 Autoregressive versus Non-Autoregressive Modeling.
Recent research [Chang et al. 2023, 2022] shows that a non-autoregressive

training paradigm for generating raster images is feasible. These approaches use bidirectional Transformers (i.e., BERT) as the fundamental model architecture. The departure from the autoregressive to non-autoregressive modeling allows parallel prediction of multiple tokens, and opens the door to diverse image editing tasks such as inpainting. We explore the possibility of using BERT to produce SVG token sequences, as an alternative to our "causal" masking scheme. It is noteworthy that, unlike raster image generation which is often fixed in length, SVG icon generation expects output sequences of varying lengths as a manifestation of generation diversity. This makes it a more difficult task because in SVG icon generation, BERT needs to not only model the likelihood of next tokens but also determine where to terminate the sequence.

After training the BERT model on the same dataset as ours, we assess the generation quality and diversity of its outputs. Quantitative evaluations are conducted similarly to DeepSVG+GAN in both random (Table 2a) and text-guided generation (Table 2b). BERT produces worse results in both tasks than ours and DeepSVG+GAN. The qualitative results in Figure 5 also suggest that BERT can only generate simple geometric shapes such as circles and rectangles that are relatively insipid. A closer inspection of the filled tokens reveals that the end-of-SVG token, <EOS>, may frequently appear in multiple positions due to parallel prediction, which results in early termination when reconstructing SVG icons. Therefore, despite the impressive ability of the non-autoregressive BERT in sequence editing, it is inferior to the autoregressive counterpart (as in IconShop) in modeling and producing sequences of varying lengths.

4.3 Comparison to the State-of-the-Art

We compare the proposed IconShop to two types of text-to-SVG generation schemes: image-based and language-based methods. For the former, we use Stable Diffusion [Rombach et al. 2022] to create raster images, which are transformed into SVG images using the LIVE [Ma et al. 2022] program. To encourage icon-style results, we include keywords (like "monochrome" and "line art") in the text prompts. For the latter, we employ GPT-4, arguably the best-performing LLM so far, to generate SVG scripts directly. To initiate the conversation, we provide a text prompt at the system level asking it to act as an SVG code generator.

Figure 6 shows icons synthesized by the competing methods. We find that the outcomes of the Stable Diffusion model often fall short of expectations. After all, it is trained on raster images, which as expected struggles to produce icon-style images even with some prompt engineering. After applying LIVE, the resulting SVG icons usually present unsmooth and inconsistent structures, many of which are not semantically recognizable. In addition, Stable Diffusion is inefficient in generation due to the iterative optimization-based vectorization by LIVE. With respect to the language-based method, GPT-4 has relatively strong capabilities in generating SVG icon scripts purely from text prompts, with moderate text-SVG alignment. Nevertheless, the results are manifested as simple combinations of primitive shapes with no complex overlaying, which are inadequate for graphics design, and also suffer from the recognizability problem (see the third and fifth rows). In stark contrast, IconShop produces results with the highest visual quality in terms of form consistency, high precision of recognizability, text-SVG alignment, and geometry simplicity. Reasonable generation diversity is also well observed from IconShop. More visual results of the text-guided generation and comparison to VectorFusion [Jain et al. 2022] and CLIPDraw [Frans et al. 2021] can be found in our supplementary material.

4.4 Subjective User Study

To formally validate the perceptual gains by IconShop, we conduct a subjective user study, consisting of three tasks to visually assess 1) random generation quality, 2) text-guided generation quality, and 3) text-SVG alignment. In the first task, we first familiarize users with high-quality icons from the training set. We then present them a total of 15×5 icons in random order, one-fifth of which are generated by (or sampled from) DeepSVG+GAN, Stable Diffusion+LIVE, GPT-4, IconShop, and *FIGR-8-SVG*, respectively. Users are forced to give a binary decision of whether each presented icon is of high quality. In the second task, we randomly select ten text prompts, and generate the corresponding icons by the four competing methods. Users are then asked to select two icons that they believe are of the best visual quality. The setup of the third task is identical to that of the second task. The difference is that this time users need to pick two icons that best match the corresponding text prompt.

We conduct the user study via an online questionnaire, with 79 participants. For the first task, we obtain 79 (users) × 15 (icons) × 5 (methods) = 5,925 human judgments. We calculate the average percentage of high-quality icons identified by users for each method, and list the results in the first row of Table 3. We see that

Table 3. Subjective user study results. In each task, we report the average percentage of selected (i.e., high-quality) icons by users.

	User Selection% ↑		
	Quality (random)	Quality (text)	Alignment (text)
DeepSVG+GAN	54.09	51.90	29.24
Stable Diffusion+LIVE	15.95	49.49	72.78
GPT-4	2.95	2.15	1.77
IconShop	82.11	**96.33**	**96.20**
Dataset	**83.71**	\	\

IconShop approaches the performance to that of the "Dataset" as the upper bound, and is clearly better than the other three methods. This shows that IconShop is able to produce high-quality icons that consistently "fool" the subjects in the random (unconditional) generation setting.

For the second and third tasks, we, respectively, collect 79 (users) × 10 (text prompts) × 2 (selections) = 1,580 human judgments for each task. We report the average percentage of user-selected icons by each method in the last two rows of Table 3. It is clear that icons synthesized by IconShop are most frequently selected, indicating the highest quality and the best text-SVG alignment in the text-guided generation task. We also perform one-way ANalysis Of VAriance (ANOVA) tests, and the p-values of the three tasks are all less than 0.001, suggesting that the perceptual gains by IconShop are statistically significant. In summary, the proposed IconShop demonstrates the highest quality for both random and text-guided generation, with a strong text-SVG alignment.

5 APPLICATIONS

In this section, we explore four practical applications of IconShop: icon editing, icon interpolation, icon semantic combination, and icon design auto-suggestion. These applications streamline the process of vector icon synthesis, enhancing user productivity and experience significantly.

5.1 Icon Editing

Thanks to the unification of non-autoregressive and autoregressive modeling through the "causal" masking strategy outlined in Section 3.3, the proposed IconShop facilitates icon editing as exemplified in Figure 7. IconShop is capable of filling in missing content based on the bidirectional context, in either random or text-guided generation scenario. This leads to precise, consistent, and diverse restoration of the missing paths in icons.

5.2 Icon Interpolation

In DeepSVG, each SVG icon is mapped into a latent representation space, which allows meaningful algebraic manipulations such as interpolation. Although IconShop is not accompanied by such a latent space, as it is not designed to learn SVG icon representations, such interpolation can be performed through text embeddings. Figure 8
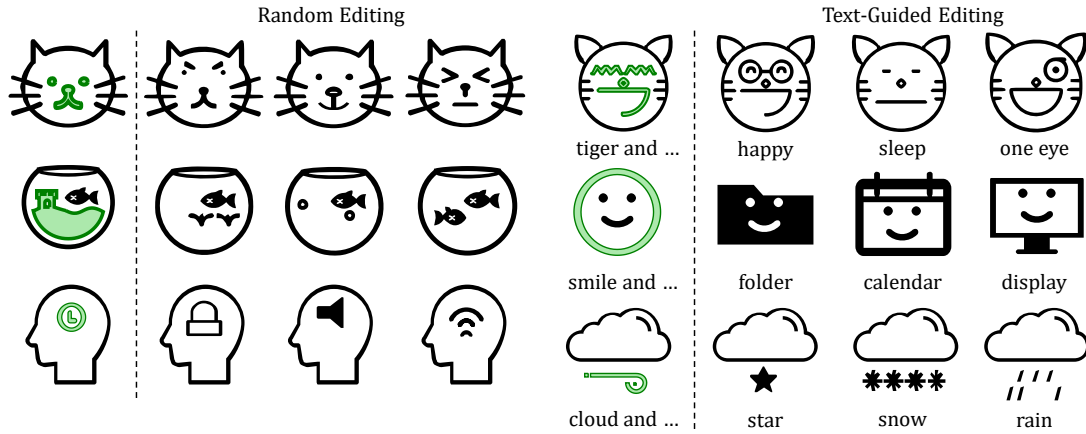
Fig. 7. IconShop enables both random and text-guided icon editing. We use the green color to highlight the paths to be edited.
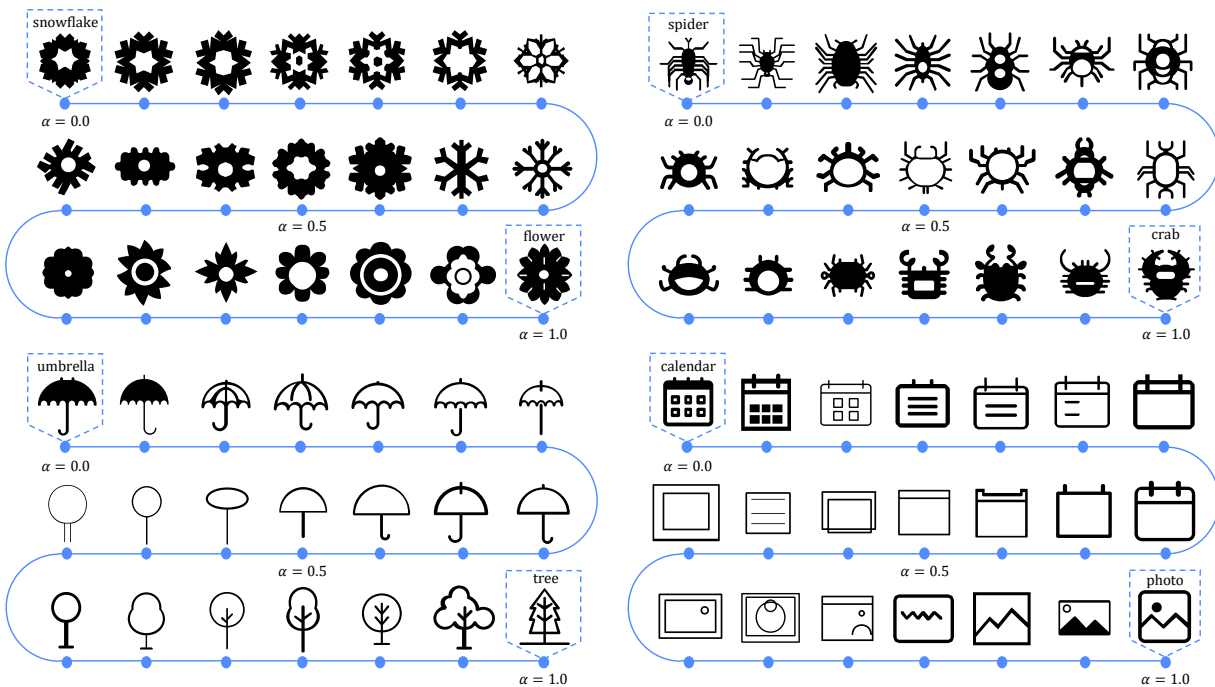

Fig. 8. We linearly interpolate two text embedding vectors using the formula $v = (1 - \alpha) \cdot v^{(1)} + \alpha \cdot v^{(2)}$, where $\alpha \in [0, 1]$. Subsequently, we generate icons corresponding to each interpolated vector v. The results show that IconShop learns a smooth mapping between the text and SVG spaces.

shows a collection of icons generated from interpolated text embedding vectors, demonstrating that IconShop has learned a smooth mapping between the text and SVG spaces.

5.3 Icon Semantic Combination

Text-to-image models demonstrate an impressive ability to combine textual inputs, encouraging generation of novel concepts that do not exist in the training data, such as "avocado chair" generated by

DALL·E [Ramesh et al. 2021]. In our experiments, we find that Icon-Shop also learns to create innovative and reasonable combinations, as shown in Figure 9.

5.4 Icon Design Auto-Suggestion

One advantage of automated icon generation is to support both designers and non-specialists in expressing their creative ideas. A desired feature of such an automated system is the ability to suggest possible placements for subsequent paths on a canvas, which would significantly improve work efficiency and productivity. Relying on
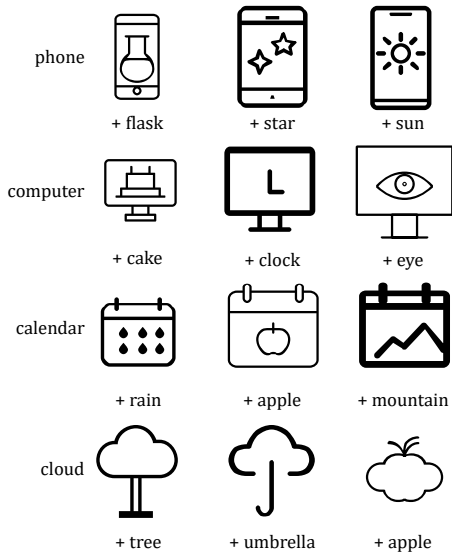
Fig. 9. IconShop learns to produce creative icons by combining semantics of different text prompts.
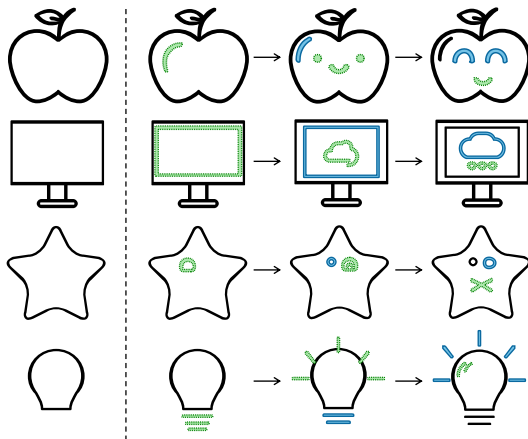

Fig. 10. IconShop is able to suggest subsequent paths for users to design icons, with significantly boosted efficiency. We highlight paths suggested by IconShop with the green color and paths drawn by users with the blue color. Even if users diverge from the suggested route, IconShop still predicts subsequent paths that are compatible with the users' chosen paths.

the autoregressive transformer, IconShop is able to predict the next path that users may choose in their icon creation processes (see Figure 10). Please refer to the supplemental material for the video demonstrations of our auto-suggestion system for SVG icon design.

6 CONCLUSION AND DISCUSSION

We have introduced IconShop, an autoregressive transformer-based method proficient at generating vector icons from textual descriptions. IconShop stands out from both image-based methods that combine text-to-image generation and image vectorization, and
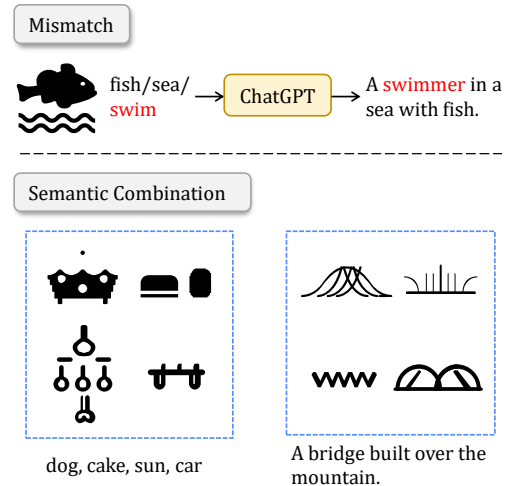

Fig. 11. Limitations contain text-SVG mismatches (top panel) and suboptimal semantic icon combination (bottom panel). In the bottom panel, we present two text inputs that significantly deviate from the dataset distribution.

language-based techniques that treat SVG scripts as natural languages. Comprehensive experiments showcase the effectiveness and flexibility of IconShop in terms of generation quality, diversity, text-icon alignment, and wide applicability.

IconShop exhibits impressive SVG icon synthesis performance by exploiting the sequential nature of SVG scripts and unifying non-autoregressive and autoregressive modeling, but it is not without limitations (see Figure 11). First, natural language phrases and sentences generated by ChatGPT may inadvertently result in text-SVG mismatches. This problem can be mitigated by using a higher-quality SVG dataset with accurate text annotations or by manual filtering. Second, the semantic icon combination performance by IconShop may not be as remarkable as text-to-image generation, because most icons in the *FIGR-8-SVG* dataset contain a single object situated at the center, occupying a significant portion of the space. In challenging scenarios where the input text significantly deviates from the dataset distribution, our model produces icons that do not align well with the text. But these icons consistently display regular shapes and symmetrical structures, as illustrated in Figure 11. We believe proper data augmentation such as scaling and merging SVG data to create new icons has the potential to improve the combination performance. Finally, IconShop is restricted to black-and-white icon generation, a task tractable to generative modeling due to its simplicity and well-defined SVG grammars. While the generated results are eye-catching, their usability as computer system icons and UI elements is not guaranteed to be optimal, because the synthesis process might overlook specific user requirements (e.g., preferred styles and complexity levels), resulting in potential inconsistencies with the overall design theme and user expectations. But IconShop could potentially be expanded to yield multicolored icons or more general SVG content (e.g., clip art) with proper modifications. For example, for multicolored icons, one potential solution is to encode color into a token.

REFERENCES

Armen Aghajanyan, Bernie Huang, Candace Ross, Vladimir Karpukhin, Hu Xu, Naman Goyal, Dmytro Okhonko, Mandar Joshi, Gargi Ghosh, Mike Lewis, and Luke Zettlemoyer. 2022. CM3: A Causal Masked Multimodal Model of the Internet. *arXiv preprint arXiv:2201.07520*.

Haruka Aoki and Kiyoharu Aizawa. 2022. SVG Vector Font Generation for Chinese Characters with Transformer. In *IEEE International Conference on Image Processing*. 646–650.

Mohammad Bavarian, Heewoo Jun, Nikolas Tezak, John Schulman, Christine McLeavey, Jerry Tworek, and Mark Chen. 2022. Efficient Training of Language Models to Fill in the Middle. *arXiv preprint arXiv:2207.14255*.

Steven Bergen and Brian J. Ross. 2012. Automatic and Interactive Evolution of Vector Graphics Images with Genetic Algorithms. *The Visual Computer* 28, 1, 35–45.

Tim Brooks, Aleksander Holynski, and Alexei A. Efros. 2022. InstructPix2Pix: Learning to Follow Image Editing Instructions. *arXiv preprint arXiv:2211.09800*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*. 1877–1901.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin-Tat Lee, Yuanzhi Lee, Scott Lundberg, Harsha Nori, Hamid Palangi, Macro T. Ribeiro, and Yi Zhang. 2023. Sparks of Artificial General Intelligence: Early Experiments with GPT-4. *arXiv preprint arXiv:2303.12712*.

Alexandre Carlier, Martin Danelljan, Alexandre Alahi, and Radu Timofte. 2020. DeepSVG: A Hierarchical Generative Network for Vector Graphics Animation. In *Advances in Neural Information Processing Systems*. 16351–16361.

Huiwen Chang, Han Zhang, Jarred Barber, Aaron Maschinot, Jose Lezama, Lu Jiang, Ming-Hsuan Yang, Kevin Murphy, William T. Freeman, Michael Rubinstein, Yuanzhen Li, and Dilip Krishnan. 2023. Muse: Text-To-Image Generation via Masked Generative Transformers. *arXiv preprint arXiv:2301.00704*.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T. Freeman. 2022. MaskGIT: Masked Generative Image Transformer. In *IEEE Conference on Computer Vision and Pattern Recognition*. 11315–11325.

Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. 2020. Generative Pretraining from Pixels. In *International Conference on Machine Learning*. 1691–1703.

Yinpeng Chen, Zhiyu Pan, Min Shi, Hao Lu, Zhiguo Cao, and Weicai Zhong. 2022. Design What You Desire: Icon Generation from Orthogonal Application and Theme Labels. In *ACM International Conference on Multimedia*. 2536–2546.

Louis Clouâtre and Marc Demers. 2019. FIGR: Few-shot Image Generation with Reptile. *arXiv preprint arXiv:1901.02199*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*.

Ming Ding, Zhuoyi Yang, Wenyi Hong, Wendi Zheng, Chang Zhou, Da Yin, Junyang Lin, Xu Zou, Zhou Shao, Hongxia Yang, and Jie Tang. 2021. CogView: Mastering Text-to-Image Generation via Transformers. In *Advances in Neural Information Processing Systems*. 19822–19835.

Ming Ding, Wendi Zheng, Wenyi Hong, and Jie Tang. 2022. CogView2: Faster and Better Text-to-Image Generation via Hierarchical Transformers. *arXiv preprint arXiv:2204.14217*.

Patrick Esser, Robin Rombach, and Bjorn Ommer. 2021. Taming Transformers for High-Resolution Image Synthesis. In *IEEE Conference on Computer Vision and Pattern Recognition*. 12873–12883.

Kevin Frans, Lisa B. Soros, and Olaf Witkowski. 2021. CLIPDraw: Exploring Text-to-Drawing Synthesis through Language-Image Encoders. *arXiv preprint arXiv:2106.14843*.

Daniel Fried, Armen Aghajanyan, Jessy Lin, Sida Wang, Eric Wallace, Freda Shi, Ruiqi Zhong, Wen-tau Yih, Luke Zettlemoyer, and Mike Lewis. 2023. InCoder: A Generative Model for Code Infilling and Synthesis. In *International Conference on Learning Representations*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*. 2672–2680.

David Ha and Douglas Eck. 2017. A Neural Representation of Sketch Drawings. *arXiv preprint arXiv:1704.03477*.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2017. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *Advances in Neural Information Processing Systems*. 6629–6640.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems*. 6840–6851.

Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Ian Simon, Curtis Hawthorne, Andrew M. Dai, Matthew D. Hoffman, Monica Dinculescu, and Douglas Eck. 2018. Music Transformer. *arXiv preprint arXiv:1809.04281*.

Ajay Jain, Amber Xie, and Pieter Abbeel. 2022. VectorFusion: Text-to-SVG by Abstracting Pixel-Based Diffusion Models. *arXiv preprint arXiv:2211.11319*.

Minguk Kang, Jun-Yan Zhu, Richard Zhang, Jaesik Park, Eli Shechtman, Sylvain Paris, and Taesung Park. 2023. Scaling up GANs for Text-to-Image Synthesis. *arXiv preprint arXiv:2303.05511*.

Takuro Karamatsu, Gibran Benitez-Garcia, Keiji Yanai, and Seiichi Uchida. 2020. Iconify: Converting Photographs into Icons. In *Joint Workshop on Multimedia Artworks Analysis and Attractiveness Computing in Multimedia*. 7–12.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*.

Diederik P. Kingma and Max Welling. 2013. Auto-Encoding Variational Bayes. *arXiv preprint arXiv:1312.6114*.

Naihan Li, Shujie Liu, Yanqing Liu, Sheng Zhao, and Ming Liu. 2019. Neural Speech Synthesis with Transformer Network. In *AAAI Conference on Artificial Intelligence*. 6706–6713.

Raphael G. Lopes, David Ha, Douglas Eck, and Jonathon Shlens. 2019. A Learned Representation for Scalable Vector Graphics. In *IEEE International Conference on Computer Vision*. 7930–7939.

Xu Ma, Yuqian Zhou, Xingqian Xu, Bin Sun, Valerii Filev, Nikita Orlov, Yun Fu, and Humphrey Shi. 2022. Towards Layer-wise Image Vectorization. In *IEEE Conference on Computer Vision and Pattern Recognition*. 16314–16323.

Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. 2021. GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models. *arXiv preprint arXiv:2112.10741*.

OpenAI. 2023. GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774*.

Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. 2022. Dreamfusion: Text-to-3D using 2D Diffusion. *arXiv preprint arXiv:2209.14988*.

Steve Probets, Julius Mong, David Evans, and David Brailsford. 2001. Vector Graphics: From PostScript and Flash to SVG. In *ACM Symposium on Document Engineering*. 135–143.

Tingting Qiao, Jing Zhang, Duanqing Xu, and Dacheng Tao. 2019. MirrorGAN: Learning Text-to-image Generation by Redescription. In *IEEE Conference on Computer Vision and Pattern Recognition*. 1505–1514.

Alec Radford, Jong-Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning Transferable Visual Models From Natural Language Supervision. In *International Conference on Machine Learning*. 8748–8763.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. *OpenAI blog* 1, 8, 9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *The Journal of Machine Learning Research* 21, 1, 5485–5551.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. 2021. Zero-Shot Text-to-Image Generation. In *International Conference on Machine Learning*. 8821–8831.

Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. 2016. Generative Adversarial Text to Image Synthesis. In *International Conference on Machine Learning*. 1060–1069.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-Resolution Image Synthesis with Latent Diffusion Models. In *IEEE Conference on Computer Vision and Pattern Recognition*. 10684–10695.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed K. S. Ghasemipour, Burcu-Karagol Ayan, S. Sara Mahdavi, Rapha G. Lopes, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. 2022. Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding. In *Advances in Neural Information Processing Systems*. 36479–36494.

Peter Selinger. 2003. Potrace: A polygon-based tracing algorithm.

Vidya Setlur, Conrad Albrecht-Buehler, Amy A. Gooch, Sam Rossoff, and Bruce Gooch. 2005. Semanticons: Visual Metaphors as File Icons. In *Computer Graphics Forum*,

Vol. 24. 647–656.

Vidya Setlur, Samuel Rossoff, and Bruce Gooch. 2011. Wish I Hadn't Clicked That: Context Based Icons for Mobile Web Navigation and Directed Search Tasks. In *International Conference on Intelligent User Interfaces*. 165–174.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-Read Students Learn Better: On the Importance of Pre-training Compact Models. *arXiv preprint arXiv:1908.08962v2*.

Rafael Valle, Kevin Shih, Ryan Prenger, and Bryan Catanzaro. 2020. Flowtron: An Autoregressive Flow-based Generative Network for Text-to-Speech Synthesis. *arXiv preprint arXiv:2005.05957*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Advances in Neural Information Processing Systems*. 6000–6010.

Yizhi Wang and Zhouhui Lian. 2021. DeepVecFont: Synthesizing High-quality Vector Fonts via Dual-modality Learning. *ACM Transactions on Graphics* 40, 6, 1–15.

Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. 2023. ImageReward: Learning and Evaluating Human Preferences for Text-to-Image Generation. *arXiv preprint arXiv:2304.05977*.

Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. 2018. AttnGAN: Fine-Grained Text to Image Generation with Attentional Generative Adversarial Networks. In *IEEE Conference on Computer Vision and Pattern Recognition*. 1316–1324.

Xiang Xu, Karl D. D. Willis, Joseph G. Lambourne, Chin-Yi Cheng, Pradeep K. Jayaraman, and Yasutaka Furukawa. 2022. SkexGen: Autoregressive Generation of CAD Construction Sequences with Disentangled Codebooks. In *International Conference on Machine Learning*. 24698–24724.

Hongyi Yang, Chengqi Xue, Xiaoying Yang, and Han Yang. 2021. Icon Generation Based on Generative Adversarial Networks. *Applied Sciences* 11, 17, 7890.

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. 2022. Scaling Autoregressive Models for Content-Rich Text-to-Image Generation. *arXiv preprint arXiv:2206.10789*.

Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N. Metaxas. 2017. StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks. In *IEEE International Conference on Computer Vision*. 5907–5915.

Zhu Zhang, Jianxin Ma, Chang Zhou, Rui Men, Zhikang Li, Ming Ding, Jie Tang, Jingren Zhou, and Hongxia Yang. 2021. M6-UFC: Unifying Multi-Modal Controls for Conditional Image Synthesis via Non-Autoregressive Generative Transformers. *arXiv preprint arXiv:2105.14211*.

Zizhao Zhang, Yuanpu Xie, and Lin Yang. 2018. Photographic Text-to-Image Synthesis with a Hierarchically-nested Adversarial Network. In *IEEE Conference on Computer Vision and Pattern Recognition*. 6199–6208.

Nanxuan Zhao, Nam-Wook Kim, Laura M. Herman, Hanspeter Pfister, Rynson W. H. Lau, Jose Echevarria, and Zoya Bylinskii. 2020. ICONATE: Automatic Compound Icon Generation and Ideation. In *ACM Conference on Human Factors in Computing Systems*. 1–13.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. 2017. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. In *IEEE International Conference on Computer Vision*. 2223–2232.