SketchEdit: Editing Freehand Sketches at the Stroke-level

Tengjie Li[1] , **Shikui Tu**[1*] , **Lei Xu**[1,2*]

[1]Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China
[2]Guangdong Institute of Intelligence Science and Technology, Zhuhai, Guangdong 519031, China
{765127364, tushikui, leixu}@sjtu.edu.cn

Abstract

Recent sketch synthesis methods have demonstrated the capability of generating lifelike outcomes. However, these methods directly encode the entire sketches making it challenging to decouple the strokes from the sketches and have difficulty in controlling local sketch synthesis, e.g., stroke editing. Besides, the sketch editing task encounters the issue of accurately positioning the edited strokes, because users may not be able to draw on the exact position, and the same stroke may appear in various locations in different sketches. We propose SketchEdit to realize flexible editing of sketches at the stroke-level for the first time. To tackle the challenge of decoupling strokes, SketchEdit divides a drawing sequence of a sketch into a series of strokes based on the pen state, aligns the stroke segments to have the same starting position, and learns the embeddings of every stroke by a proposed stroke encoder. Moreover, we overcome the problem of stroke placement via a diffusion process, which progressively generates the locations for the strokes to be synthesized, using the stroke features as the guiding condition. Experiments demonstrate that SketchEdit is effective for stroke-level sketch editing and sketch reconstruction. The source code is publicly available at https://github.com/CMACH508/SketchEdit/.

1 Introduction

People may draw sketches to express their abstract concepts for the real world, and humans possess an extraordinary ability to create imaginative sketches. The objective of sketch synthesis is to mimic the human drawing process through machines, and the task is challenging due to the sketch's abstractness, sparsity, and lack of details. Recently, efforts have been made to learn efficient sketch representations and generate realistic sketches, such as Sketch-RNN [Ha and Eck, 2017], SketchHealer [Su *et al.*, 2020], SketchLattice [Qi *et al.*, 2021] and SP-gra2seq [Zang *et al.*, 2023a].

*Corresponding Author

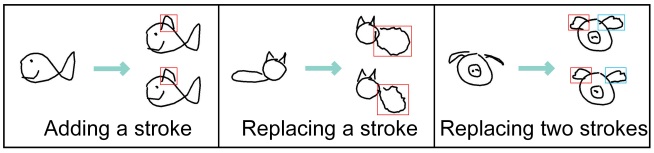

Figure 1: (Arrow left) Original sketches. (Arrow right) Edited sketches generated by our model. From left to right are: (1) A 'fin' is added to the fish. (2) A sheep's body replaces the body of a cat. (3) The wings of an angel replace the ears of a pig. The edited strokes are from the latent space of our model and the QuickDraw dataset.

However, whilst existing methods [Zang *et al.*, 2021; Zang *et al.*, 2023b; Wang *et al.*, 2023] exhibit effective control on generating sketches with certain global properties, they are unable to perform finer control on strokes. For example, researchers have focused on synthesizing sketches of particular categories, such as generating a "cat", but have difficulty in manipulating the shape of certain parts (e.g., the body) of the "cat". Furthermore, during the sketch creation process, users may incorporate fresh strokes or choose strokes for multiple revisions based on inspiration. This paper attempts to present a framework, which generates imaginative editing outcomes and may assist in the heuristic education of children, to mimic human sketch editing at the stroke-level as in Figure 1.

To achieve stroke-level editing, it is a key obstacle to pinpoint the strokes that require editing. For the conventional method [Ha and Eck, 2017] using a sequence of points to represent sketches, although the segments determined by the pen states can be directly used as strokes, the lengths of the obtained strokes are not the same, which is not convenient for editing the strokes and updating the sketch sequence. Rasterizing a sketch into an image is a common operation in sketch studies [Chen *et al.*, 2017; Yu *et al.*, 2015; Yu *et al.*, 2016]. However, these image-based methods lost details of the drawing order and the way sketches are drawn, making it more difficult to get the stroke information. Recently, the work [Qu *et al.*, 2023] provided an effective way to break down the sketch sequence into strokes for downstream tasks, where the stroke segments are padded to be of the same length. Inspired by this idea, we develop a stroke encoder to encode each stroke separately, without exchanging information with another stroke. This approach provides the flexibil-

ity to select strokes and edit them in the latent space of the encoder while minimizing the impact on the content of the rest part of the sketch.

Another challenge for stroke-level editing is how to appropriately place the strokes after the editing is done. As given in the second box of Figure 1, if we replace the cat's body with the sheep's body, the cat's head moves from the right to the left side of the image. If the cat's head is still in its original position, the generated sketch will be unrealistic. Here, we develop a diffusion model [Ho *et al.*, 2020] for accurate stroke placement. The diffusion model generates the stroke locations progressively through the denoising process, based on the features of all strokes to be synthesized. The diffusion model extends beyond the generation of single-category sketches, enabling the creation of more diverse results, e.g., a pig with wing-like ears. Furthermore, we fuse the stroke embeddings with the generated stroke locations and devise a sequence decoder to synthesize the final manipulated sketch. The stroke encoder and the sequence decoder are jointly pretrained under the autoencoder paradigm, with an extra image decoder to learn the local structure of sketches.

In summary, we propose a novel sketch editing method called *SketchEdit* and our contributions are as follows: (i) We develop the traditional task of sketch synthesis into a more controllable sketch editing task at the stroke-level for the first time. The proposed SketchEdit achieves this purpose well and enables the generation of creative sketches. (ii) We present a fresh perspective on the placement of sketch strokes without labeling, where strokes are synthesized akin to assembling building blocks. Given a set of base strokes, we first generate meaningful placements for them, and then combine the strokes into a meaningful sketch. (iii) Experiments show that our method performs significantly better than the state-of-the-art sketch generation models for the task of sketch reconstruction. This guarantees that the edited sketch effectively retains the visual properties of the original sketch for sketch editing at the stroke-level.

2 Related Work

Sketch generation. Sketching, as a practical communication tool and medium for emotional expression, is impressive and expressive [Xu *et al.*, 2022; Ribeiro *et al.*, 2020; Alaniz *et al.*, 2022]. Its related generative tasks have attracted the interest of researchers [Zhou *et al.*, 2018; Das *et al.*, 2021; Pourreza *et al.*, 2023]. An essential work to this is Sketch-RNN [Ha and Eck, 2017], which is facilitating research into deep learning for the imitation of human drawing. Although Sketch-RNN is capable of accurately capturing the connection between drawing points, it falls short in perceiving the local structural information of images. Therefore, the subsequent methods [Chen *et al.*, 2017; Song *et al.*, 2018] convert the sequence of sketches into rasterized images and introduce Convolutional Neural Networks (CNNs) as a replacement or supplement to the RNN encoder. To improve the representational capabilities of the models, graph neural networks (GNNs) are introduced on top of the image representation [Su *et al.*, 2020; Qi *et al.*, 2022; Qi *et al.*, 2021; Zang *et al.*, 2023a]. These methods construct graphs by temporal proximity, spatial proximity, or synonymous proximity. Another methods to improve performance are to use a Gaussian Mixture Model (GMM) to model the latent space [Zang *et al.*, 2021; Zang *et al.*, 2023b] or design a Lmser-based network to learn stable sketch representations [Li *et al.*, 2024]. These models have struggled to decouple specific strokes, so our SketchEdit takes strokes as input rather than images or drawing points. There is also a similar class to our methodology, which views sketches as being comprised of multiple parts and requires labeling the components of the sketch, such as the head of a bird [Ge *et al.*, 2020]. However, due to the manual labeling being costly, our method is geared towards unlabeled and more basic strokes. Recently, some studies utilized a parametric representation of sketches [Vinker *et al.*, 2022; Xing *et al.*, 2023] for easy generation. These approaches lack an important feature of sketching, which is the ability to maintain the order in which human strokes are drawn.

Diffusion models. Diffusion models [Sohl-Dickstein *et al.*, 2015] have led to a boom in research, particularly in the field of image synthesis [Ho *et al.*, 2020; Dhariwal and Nichol, 2021]. Text-to-image (T2I) generation is a widely recognized application of diffusion models, which enables the rapid generation of artwork by providing prompts as a cue to large models [Ramesh *et al.*, 2021; Rombach *et al.*, 2022]. However, certain information remains difficult to convey solely through text, leading to the emergence of visual cues as conditions for diffusion models. Sketches are an effective tool for responding to structural information and are therefore regarded as control conditions by PITI [Voynov *et al.*, 2023], ControlNet [Zhang and Agrawala, 2023], T2I-Adapter [Mou *et al.*, 2023], and other methods. Recently some diffusion models [Wang *et al.*, 2023; Das *et al.*, 2023] about sketches have been proposed, which focus on modeling the points of the sketch rather than the stroke locations. Our approach differs from these pure diffusion models used for sketch generation in that SketchEdit is able to utilize the highly semantic latent space of the AE paradigm for flexible stroke editing.

3 Methodology

SketchEdit is constructed based on diffusion model to edit sketches at the stroke-level. The key step is to generate the locations of the strokes. This is achieved by the reverse denoising process of the diffusion model conditioned on stroke embeddings, as shown in Figure 2(a). The SketchEdit decouples sketch into several strokes without position information, allowing the user to conveniently select strokes for editing. Strokes and generated locations are eventually fed into a sequence decoder to synthesis the edited sketch. The pipeline of editing sketches are illustrated in Figure 2(b).

3.1 Sketch Representation

A sketch is represented by a sequence of L_p points, i.e., $\boldsymbol{\tau} = (\boldsymbol{p}_1, \boldsymbol{p}_2, ..., \boldsymbol{p}_{L_p})$. Each point $\boldsymbol{p}_i$ is a vector containing five elements. The first two are the coordinates of the absolute position, while the last three use the one-hot vector format to represent the three pen states of lift, touch, and the end of sketch. To proceed in the stroke-level, the sketch sequence

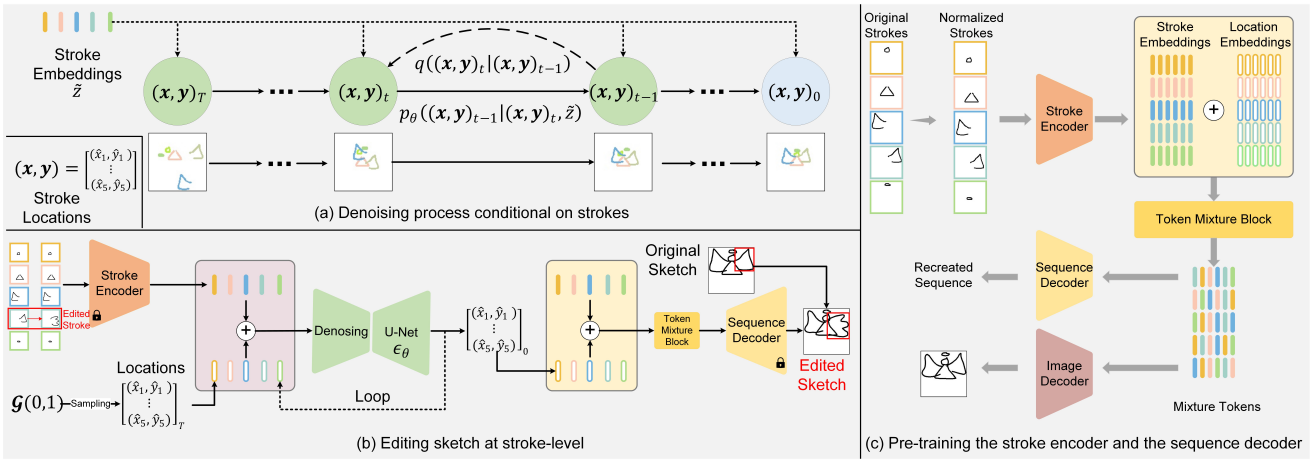

Figure 2: The overview of the proposed SketchEdit. (a) Denoising process conditional on strokes. Essentially, the goal is to reorganize strokes with confusing positions into meaningful sketches. (b) The pipeline for sketch editing by our method. The edited strokes are replaced at the input (or in the latent space) against the target strokes, and then the inverse denoising process is used to obtain meaningful stroke positions from the random noise. (c) Pre-training the stroke encoder and the sequence decoder which are used to generate stroke embeddings and synthesis target sketches for sketch editing task.

is broken down into a series of strokes, i.e., $(\boldsymbol{s}_1, \boldsymbol{s}_2, ..., \boldsymbol{s}_{L_s})$, where L_s denotes the number of strokes. We use $(\boldsymbol{x}, \boldsymbol{y}) = [(x_1, y_1), (x_2, y_2), \ldots, (x_{L_s}, y_{L_s})]$ to record the locations of the strokes, which are the coordinates of the first point of the stroke. In this paper, we also define the normalized stroke sequence $\tilde{\boldsymbol{s}}_i$ by subtracting the location (x_i, y_i) of the stroke from the coordinates of all the points in the stroke.

3.2 Diffusion Model for Generating Locations

Forward process. Given a set of stroke locations $(\boldsymbol{x}, \boldsymbol{y})_0 \sim q((\boldsymbol{x}, \boldsymbol{y})_0)$, we apply the Markov diffusion process in DDPMs [Ho *et al.*, 2020] here. The noise sampled from Gaussian distribution is gradually added to $\boldsymbol{x}$ and $\boldsymbol{y}$:

$$
\begin{aligned}
q((\boldsymbol{x}, \boldsymbol{y})_{1:T}|(\boldsymbol{x}, \boldsymbol{y})_0) &= q((\boldsymbol{x}, \boldsymbol{y})_0) \prod_{t=1}^{T} q((\boldsymbol{x}, \boldsymbol{y})_t|(\boldsymbol{x}, \boldsymbol{y})_{t-1}), \\
q((\boldsymbol{x}, \boldsymbol{y})_t|(\boldsymbol{x}, \boldsymbol{y})_{t-1}) &= \mathcal{N}((\boldsymbol{x}, \boldsymbol{y})_t; \sqrt{1-\beta_t}(\boldsymbol{x}, \boldsymbol{y})_{t-1}, \beta_t \boldsymbol{I}),
\end{aligned} \tag{1}
$$

where β_t represents the noise schedule at time t.

Reverse process. The reverse process aims to recreate the true locations from a Gaussian noise input $(\boldsymbol{x}, \boldsymbol{y})_T$. Similar with the DDPMs [Ho *et al.*, 2020], A U-Net [Ronneberger *et al.*, 2015] like network is utilized to predict the noise $\epsilon_\theta((\boldsymbol{x}, \boldsymbol{y})_t, t)$. However, stroke locations have no explicit semantic information, so it is necessary to introduce strokes as a condition. Thus, the network for predicting noise is modified to $\epsilon_\theta((\boldsymbol{x}, \boldsymbol{y})_t, t, \tilde{\boldsymbol{s}})$. To decrease computational complexity and leverage high-level semantic information, as illustrated in Figure 2, we utilize the stroke embeddings $\tilde{\boldsymbol{z}}$ as the condition rather than the strokes $\tilde{\boldsymbol{s}}$. The reverse denoising process can be formalized as:

$$
\begin{aligned}
p_\theta((\boldsymbol{x}, \boldsymbol{y})_{t-1}|(\boldsymbol{x}, \boldsymbol{y})_t, \tilde{\boldsymbol{z}}) &= \mathcal{N}((\boldsymbol{x}, \boldsymbol{y})_{t-1}; \boldsymbol{\mu}_\theta((\boldsymbol{x}, \boldsymbol{y})_t, t, \tilde{\boldsymbol{z}}), \boldsymbol{\sigma}_t^2 \boldsymbol{I}), \\
\boldsymbol{\mu}_\theta((\boldsymbol{x}, \boldsymbol{y})_t, t, \tilde{\boldsymbol{z}}) &= \frac{1}{\alpha_t}((\boldsymbol{x}, \boldsymbol{y})_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta((\boldsymbol{x}, \boldsymbol{y})_t, t, \tilde{\boldsymbol{z}})),
\end{aligned} \tag{2}
$$

where $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$. In practice, we use the DDIM-based [Song *et al.*, 2020] generation process for accelerated sampling.

3.3 Editing Freehand Sketches at the Stroke-level

In this subsection, we provide the process of editing sketch at the stroke-level. First, users pick the stroke $\tilde{\boldsymbol{s}}_i$ they want to edit from the sketch $\boldsymbol{\tau}$. The edited stroke $\hat{\boldsymbol{s}}_i$ can either be drawn by the users or selected from the stroke gallery to replace $\tilde{\boldsymbol{s}}_i$. Taking the angle shown in Figure 2(b) as an example, we have obtained the strokes $\hat{\boldsymbol{s}}(\tilde{\boldsymbol{s}}_1, \tilde{\boldsymbol{s}}_2, \tilde{\boldsymbol{s}}_3, \hat{\boldsymbol{s}}_4, \tilde{\boldsymbol{s}}_5)$ after editing. Then, the stroke encoder calculates the stroke embeddings $\hat{\boldsymbol{z}}(\tilde{\boldsymbol{z}}_1, \tilde{\boldsymbol{z}}_2, \tilde{\boldsymbol{z}}_3, \hat{\boldsymbol{z}}_4, \tilde{\boldsymbol{z}}_5)$. As the encoding process does not involve the exchange of stroke information, stroke substitution in the latent space, such as replacing $\tilde{\boldsymbol{z}}_4$ with $\hat{\boldsymbol{z}}_4$, is also possible.

Next, we apply the reverse process of diffusion model to denoise random noise $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})_T$ conditional on $\hat{\boldsymbol{z}}$, resulting in generated stroke locations $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})_0$. Finally, the stroke embeddings $\hat{\boldsymbol{z}}$ and the stroke locations $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})_0$ are fed into the token mixture block and sequence decoder to synthesis the target sketch $\hat{\boldsymbol{\tau}}$.

3.4 Constructing the Stroke Encoder, the Sequence Decoder, and the Image Decoder

After converting the sketch sequence to the normalized stroke representation, the resulting tensor $\tilde{\boldsymbol{s}} \in \mathbb{R}^{L_s \times L_n \times 5}$ is obtained, where L_n is the number of points in a stroke. A position-sensitive block must act as the backbone of the stroke encoder to extract features form $\tilde{\boldsymbol{s}}$ because significant changes in the shape of the stroke occur when any two points in the sequence are interchanged. Token-based MLPs [Tolstikhin *et al.*, 2021] fulfil this requirement, and thus we consider gMLP [Liu *et al.*, 2021] as the basic component. Since we do not wish for any exchange of information to occur dur-

ing the encoding stage between the strokes, we can intuitively treat the first dimension of $\tilde{\boldsymbol{s}}$ as the batch size.

Several layers are used to extract the stroke embeddings $\tilde{\boldsymbol{z}}$. Firstly, each point in a stroke is treated as a token, which then interacts through the network with other points. Next, these tokens are summed for aggregation to get $\tilde{\boldsymbol{z}}_{enc} \in \mathbb{R}^{L_s \times d_{model1}}$, where d_{model1} denotes the dimension of the tokens. The stroke embeddings $\tilde{\boldsymbol{z}} \in \mathbb{R}^{L_s \times d_{model2}}$ are calculated as followings:

$$\begin{aligned} \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\sigma}} = f_{linear}(\tilde{\boldsymbol{z}}_{enc}), \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\sigma}} \in \mathbb{R}^{L_s \times d_{model2}}, \\ \tilde{\boldsymbol{z}} = \tilde{\boldsymbol{\mu}} + \tilde{\boldsymbol{\sigma}} \times \boldsymbol{\epsilon}_{enc}, \boldsymbol{\epsilon}_{enc} \sim \mathcal{G}(\mathbf{0}, \boldsymbol{I}), \end{aligned} \tag{3}$$

where $f_{linear}(\cdot)$ and d_{model2} represents a linear projection and the dimension of stroke embeddings, respectively. The reparameterization trick [Kingma and Welling, 2013] employed in Equation (3) serves to effectively constrain the latent space, resulting in improved continuity.

Then, we map the stroke locations $(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{R}^{L_s \times 2}$ to the location embeddings $\boldsymbol{z}_{loc} \in \mathbb{R}^{L_s \times d_{model2}}$. The summation of $\tilde{\boldsymbol{z}}$ and $\boldsymbol{z}_{loc}$ is fed into a token mixture block to mixture the information of different strokes. The resulting $\boldsymbol{z}_{mix} \in \mathbb{R}^{L_s \times d_{model2}}$ is subsequently sent to both the sequence decoder and the image decoder. The decoders utilize spatial projection to increase the number of tokens before reconstructing either the sequence $\tilde{\boldsymbol{\tau}}(\tilde{\boldsymbol{p}}_1, \tilde{\boldsymbol{p}}_2, ..., \tilde{\boldsymbol{p}}_{L_p})$ or the image $\tilde{\boldsymbol{I}}$. The backbone of token mixing block and sequence decoder is gMLP, while the image decoder is built based on CNNs. Thanks to the powerful global capture capability of gMLP, we can decode all sequence points simultaneously, rather than using the autoregressive approach [Ha and Eck, 2017; Chen *et al.*, 2017; Su *et al.*, 2020]. This still result in good reconstruction outcomes.

3.5 Two-stage Training

Pre-training the stroke encoder, the sequence decoder, and the image decoder. After completing end-to-end training, the stroke encoder and the sequence decoder can effectively reconstruct sketches. There are three training objectives. The first is for the output of the sequence decoder, where our goal is to minimize the negative log-likelihood function of the generated probability distribution:

$$\mathcal{L}_{seq} = -\mathbb{E}_{u_\phi(\tilde{z}|\tilde{S})} \log v_\xi(\tilde{\boldsymbol{\tau}}|\tilde{\boldsymbol{z}}, (\boldsymbol{x}, \boldsymbol{y})). \tag{4}$$

The training goal in Sketch-RNN [Ha and Eck, 2017] also pursues this aim, with the difference being the absolute or relative coordinates modeling. Second, for calculating the image reconstruction loss $\mathcal{L}_{img}$, we utilize the traditional mean square error (MSE). Finally, to improve the representational power of the model [Zang *et al.*, 2021; Zang *et al.*, 2023b], GMM modeling is carried out in the encoder's latent space. We initialize K Gaussian components and the appropriate number is determined automatically with the aid of RPCL [Xu *et al.*, 1993]. The corresponding loss function is formalized as follows:

$$\mathcal{L}_{GMM} = \sum_{i=1}^{L_s} KL(u_\phi(\tilde{\boldsymbol{z}}_i, k|\tilde{\boldsymbol{s}}_i)||o_\psi(\tilde{\boldsymbol{z}}_i, k)), \tag{5}$$

where $\tilde{\boldsymbol{z}}_i$ is the stroke embedding correspond to the stroke $\boldsymbol{s}_i$ and the KL term is calculated as in [Jiang *et al.*, 2016]. The parameters of the GMM are learned by an EM-like algorithm, details of which can be found in [Zang *et al.*, 2021]. In summary, the overall objective is:

$$\mathcal{L}_{AE} = \mathcal{L}_{seq} + \mathcal{L}_{img} + \lambda \mathcal{L}_{GMM}, \tag{6}$$

where λ is a hyperparameter and we set it to 0.0001 in practice.

Training the diffusion model. In this stage, the previously trained parameters of the stroke encoder and the sequence decoder are fixed, and the following are the training objectives of the diffusion model:

$$\min_\theta \mathbb{E}||\epsilon - \epsilon_\theta((\boldsymbol{x}, \boldsymbol{y})_t, t, \tilde{\boldsymbol{z}}))||_2^2. \tag{7}$$

4 Experiment

4.1 Preparation

Dataset. Two datasets are selected from the largest sketch dataset QuickDraw [Ha and Eck, 2017] for experiments. **DS1** is a 17-category dataset [Su *et al.*, 2020; Qi *et al.*, 2022]. The specific categories are: *airplane, angel, alarm clock, apple, butterfly, belt, bus, cake, cat, clock, eye, fish, pig, sheep, spider, umbrella, the Great Wall of China*. These categories are common in life and the instances in the categories are globally similar in appearance. **DS2** [Zang *et al.*, 2021] is a multi-style and comparatively small dataset for synthesized sketches, comprising five categories: *bee, bus, flower, giraffe, and pig*. Each category contains 70000 sketches for training and 2500 sketches for testing.

Implementation details. The AdamW optimizer [Loshchilov and Hutter, 2017] is applied to train the proposed model with parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$ and $weight\ decay = 0.01$. We use the CosineAnnealingLR scheduler [Smith and Topin, 2019] with the peak learning rates are 0.002 and 0.0005 for the pre-trained model and the diffusion model, respectively. We set drop path rate to 0.1. All the sketch is padded to the same length, i.e. $L_p = 180$. Each sketch is break down into $L_s = 25$ strokes and each stroke contains 96 points. For the pre-trained network, we train it with 15 epochs and the batch size is 200. There are 8 gMLP blocks in the stroke encoder with $d_{model1} = 96$ and $d_{ffn1} = 384$. The token mixture block and the sequence decoder includes 2 and 12 gMLP blocks, respectively. We set $d_{model2} = 128$ and $d_{ffn2} = 512$ for these blocks. We train the U-Net of the diffusion model with 40 epochs with the batch size is 768. The encoder and the decoder both consist of 12 gMLP blocks. The d_{model} and d_{ffn} in these blocks are 96 and 384, respectively. We consider the linear noise schedule for the model with $\beta_t \in (0.0001, 0.02)$. We take 60 steps for DDIM sampling in default and truncate the stroke locations at $(-1, 1)$ for better performance. More implementation details are in the Supplementary material.

Baselines. We consider 3 types of models as the baselines. Sketch-RNN [Ha and Eck, 2017] employs a VAE [Kingma and Welling, 2013] framework to learn sketch representations from *sequences*. Sketch-pix2seq [Chen *et al.*, 2017] and

RPCL-pix2seq [Zang *et al.*, 2021] take sketch *images* as input to learn local structural information form sketches. Based on the rasterized sketch images, SketchHealer [Su *et al.*, 2020], SketchLattice [Qi *et al.*, 2021], and SP-gra2seq [Zang *et al.*, 2023a] introduce the *graphs* for better representations.

Metrics. To evaluate the performance of the SketchEdit, we select Rec [Zang *et al.*, 2021], FID [Heusel *et al.*, 2017], LPIPS [Zhang *et al.*, 2018], and CLIP Score [Radford *et al.*, 2021; Hessel *et al.*, 2021] as the metrics. To classify whether the recreated sketches belong to the original category, two sketch-a-nets [Yu *et al.*, 2015] are trained on DS1 and DS2, respectively. Rec is the success rate of recognition named by [Zang *et al.*, 2021]. The CLIP Score in this paper specifically measures the similarity between the generated sketch and the original sketch.

4.2 Editing Sketches at the Stroke-level

Stroke-level sketch editing involves modifying distinct strokes while minimizing the impact on the overall structure. In this subsection, we provide qualitative analysis and some applications related to the stroke-level sketch editing, including stroke replacement, interpolation between strokes, and stroke addition.

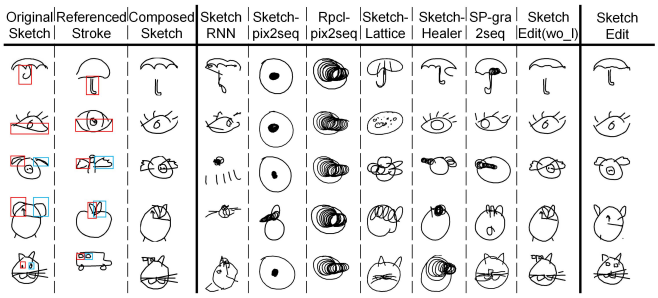

Figure 3: Examples of stroke replacement. The strokes in the first column of each row are replaced by strokes from the same color box in the second column. The third column is the results after the strokes have been replaced. The sketches in the middle are synthesized based on the third column of sketches as input. The final column depicts the outcome of our technique. SketchEdit(w_ol) denotes generating sketches with original locations.

Replacing strokes. Sketches typically consist of various basic shapes and strokes from other sketches can be conveniently reused to edit the intended sketch, as illustrated in Figure 3 and Figure 4. The recycled shapes may comprise constituents from the identical class with clearly defined meanings, for example, an airplane fuselage, an umbrella handle, and so on. Apart from that, SketchEdit enables a sensible synthesis of strokes from different categories of sketches. Some examples are provided in Figure 3, for instance, the alarm clock's bells have been replaced by apple stems, and the SketchEdit has found a "logical" place for the apple stem. Since the stroke encoder encodes only the structure of the strokes and extracts features with high-level semantics while avoiding the influence of sketch position information, the resulting stroke embeddings are of high-quality. This assists the diffusion model in learning stroke position relationships in a more effective manner and enables the sequence decoder to efficiently generate after incorporating position information. Figure 3 also gives the synthesis results of the other methods and our technique of using the original locations. The methods of comparison encode the entirety of the sketch, and misplaced strokes can significantly impact the synthesis, making it difficult to generate recognizable sketches. Editing sketches to produce creative results using existing basic shapes can be challenging if the adverse impact of the initial stroke position on the outcome is not reduced. Our objective in normalizing strokes and implementing a diffusion model is specifically to address this issue. Although our sub-method cannot produce high-quality sketches utilizing the initial stroke placements, it expertly preserves the visual features of each individual stroke. *Effective stroke reconstruction is a crucial requirement for stroke-level editing, ensuring minimal alteration to the overall sketch structure.* More reconstruction-related content will be discussed in the next subsection.

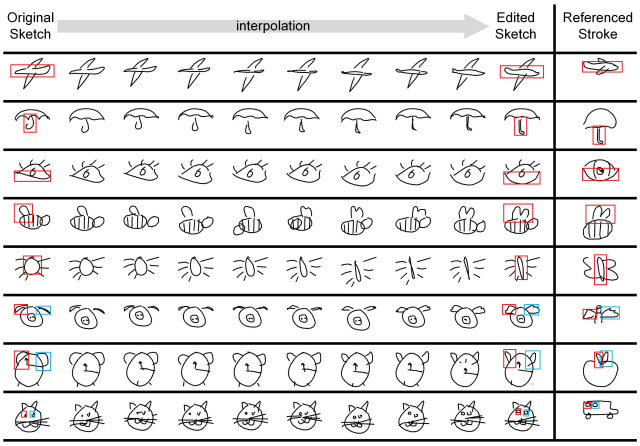

Figure 4: Stroke interpolation results. Boxes of the same color in each row denote the respective modified strokes. Creative sketches can be generated through the interpolation between strokes in the latent space and the locations of strokes are produced by our diffusion model.

Interpolating between strokes. Figure 4 provides some examples of interpolation between strokes. For edited strokes, the transition from source to target is smooth as the stroke encoder acquires a well-organized and impact latent space. During the process of interpolation, some stroke positions experience a perturbation, e.g., the cats move up and down slightly. This effect is partly attributed to a degree of randomness, which is caused by the initial Gaussian noise of the inverse process in the diffusion model [Ho *et al.*, 2020], and the sampling production of the sequences [Ha and Eck, 2017]. This randomness does not typically affect the overall structure of the edited sketch, and only has negative consequences in certain instances, as exemplified by the bees illustrated in Figure 4.

Adding new strokes. In our method, the way to adding new strokes to a sketch is similar with stroke replacement. Ini-

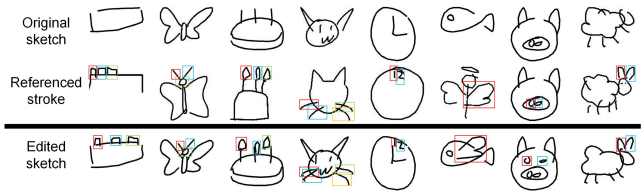

Figure 5: Adding strokes to sketches. Boxes of the same color in each column denote the corresponding referenced strokes and added strokes.

tially, the first n padded strokes are replaced with the target strokes in the input field or in the latent space. The new stroke set is then used to generate the stroke locations and the edited sketch. Figure 5 shows the results of adding strokes. The diffusion model generates reasonable locations for the added strokes, an interesting example of which is the clock in the sixth column. In the referenced sketch, the stroke '1' and the stroke '2' are used as a whole to represent the clock scale '12'. However, in the original sketch, there is not sufficient space to include scale '12', and therefore the two strokes are more appropriately represented as scale '1' and scale '2', respectively. Reusing the strokes from the original sketch, our method can also respond effectively. For example, the nostrils of the pig in the penultimate column are reused, and the diffusion model predicts them to the location of the eyes.

4.3 Sketch Reconstruction

Sketch reconstruction (also called controllable sketch synthesis [Zang *et al.*, 2021; Zang *et al.*, 2023a]) requires the model to recreate the sketch $\tilde{\boldsymbol{\tau}}$ from the input $\boldsymbol{\tau}$. High-quality sketch reconstruction is essential to maintaining a consistent visual appearance between the edited sketch and the original sketch. In this subsection, we compare the SketchEdit with other sketch synthesis methods. It should be noticed that, when we use the original locations of the strokes rather than the generated locations for SketchEdit, the comparison becomes fair, because the inputs to the baselines are full sketches (including stroke locations). In this subsection, SketchEdit(w_ol) denotes that SketchEdit recreates sketches with original locations instead of generated locations.

Qualitative analysis. Figure 6 presents the qualitative comparisons. Compared to other approaches, SketchEdit(w_ol) is capable of reconstructing sketches with high-quality, without introducing additional noisy strokes, while preserving the structural patterns of the sketches. To prevent generated sketches from changing category, the model must first learn an accurate representation of the category-level. A failure case is that Sketch-pix2seq [Chen *et al.*, 2017] reconstructs the last column of the Great Wall into a belt. Capturing structural information at the instance-level is a challenging undertaking. While nearly all the competitors reproduced "cakes" as "cakes", the generated results displayed significant structural changes. Furthermore, the existence of multiple styles within the same sketch category poses a challenge to sketch reconstruction. The proposed SketchEdit(w_ol) shows significant preservation of detail about sketch instances, which is the basis for our sketch editing task. When we reconstructed

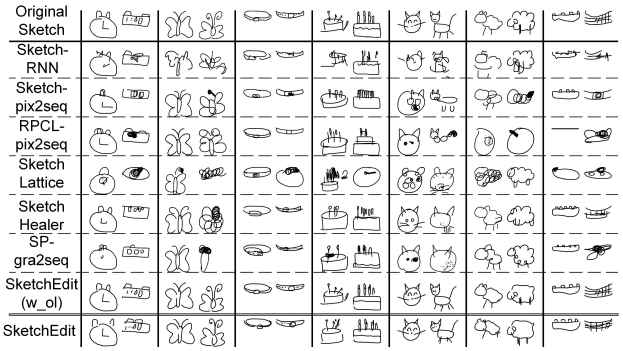

Figure 6: The exemplary result of reconstructed sketches by the proposed SketchEdit and other models. The categories from left to right are alarm clock, butterfly, belt, cake, cat, sheep, and the Great Wall of China.

the sketch using the completed SketchEdit, the results are slightly degraded, mainly reflecting the subtle movement of some of the strokes. Nevertheless, SketchEdit is still superior to baselines in terms of visualization.

Dataset	Model	Rec(↑)	FID(↓)	LPIPS(↓)	CLIP-S(↑)
DS1	Sketch-RNN	64.51%	6.87	0.33	91.82
	Sketch-pix2seq	66.99%	42.03	0.34	90.04
	RPCL-pix2seq	69.86%	44.09	0.32	90.37
	SketchLattice	48.88%	48.70	0.44	87.06
	SketchHealer	76.76%	21.62	0.32	92.15
	SP-gra2seq	76.60%	21.92	0.33	92.01
	SketchEdit(w_ol)	**84.32%**	**3.12**	**0.11**	**96.73**
	SketchEdit	80.15%	3.58	0.29	94.73
DS2	Sketch-RNN	77.74%	10.45	0.40	90.29
	Sketch-pix2seq	88.36%	42.78	0.37	90.22
	RPCL-pix2seq	90.66%	27.32	0.35	90.80
	SketchLattice	77.54%	50.92	0.45	87.80
	SketchHealer	90.93%	24.43	0.36	91.28
	SP-gra2seq	91.12%	21.69	0.37	91.15
	SketchEdit(w_ol)	**93.42%**	**5.88**	**0.19**	**94.25**
	SketchEdit	87.22%	7.64	0.36	92.20

Table 1: The performance for sketch reconstruction.

Quantitative analysis. Table 1 reports the sketch reconstruction performance of the proposed method and its competitors. SketchEdit(w_ol) significantly outperforms other methods across all metrics. The proposed method captures global dependencies in sketch sequences more efficiently, while the proposed sequence decoder addresses the challenge of stacked layers in RNN and the deeper network improves reconstruction results. However, due to the data-driven nature of the gMLP block, it lacks adequate inductive bias, resulting in a less prominent advantage of SketchEdit(w_ol) on the smaller DS2 compared to DS 1. Similar to the results of the qualitative analysis, the metrics for SketchEdit, which employs generated locations for synthesizing sketches, decreased in comparison to SketchEdit(w_ol). On DS1, SketchEdit outperforms the other methods significantly, whereas on DS2, it is able to achieve comparable results. For Sketch-RNN [Ha and Eck, 2017], the FID metrics and other metrics present a distinct phenomenon because there exists a considerable domain gap between the sequences and the images, resulting in a disparity between the distribu-

tions learned by the image-based approach and the sequence sketches.

4.4 Ablation Study

In this subsection, we discuss the sampling process, model components, and why the diffusion model was chosen. We conduct the ablation study on DS1.

Sampling process. The DDIM [Song *et al.*, 2020] sampling method features a hyperparameter η. If $\eta = 0$, the inverse process is deterministic sampling. Conversely, if $\eta = 1$, the inverse process involves the original DDPM [Ho *et al.*, 2020] generation process. Both scenarios are examined, alongside an evaluation of the impact of truncating stroke locations in the sampling process. Table 2 reports the reconstruction performance for different sampling settings. When the positions are not truncated at each sampling step, $\eta = 1$ gives better results than deterministic sampling for the same step. This issue may arise if the initial stroke generation occurs outside the canvas area. DDIM sampling is significantly impacted by this phenomenon and faces challenges in producing satisfactory outcomes, therefore, limiting stroke position can be considered a viable solution. We observe that a sampling of 10 steps gives better results than a sampling of 5 steps, with additional steps having possible negative consequences. Due to the sensitivity and lack of semantic information in the sketch stroke locations, the predicted noise by U-Net may not be entirely accurate. Further steps may lead to an accumulation of errors which could negatively impact performance.

Settings			Performance			
Steps	η	L.T.	Rec(↑)	FID(↓)	LPIPS(↓)	CLIP-S(↑)
1	-	-	11.26%	54.71	0.57	87.92
5	1	×	79.84%	3.78	0.28	94.69
10	1	×	80.16%	3.62	0.28	94.72
30	1	×	79.20%	3.65	0.29	94.66
60	1	×	79.62%	3.79	0.29	94.59
5	1	✓	80.19%	3.79	0.27	94.71
10	1	✓	80.44%	3.61	0.28	94.74
30	1	✓	79.72%	3.68	0.29	94.69
60	1	✓	79.01%	3.76	0.29	94.61
5	0	×	79.73%	3.82	0.28	94.65
10	0	×	79.66%	3.70	0.28	94.66
30	0	×	78.24%	3.82	0.29	94.55
60	0	×	77.68%	3.98	0.30	94.46
5	0	✓	80.22%	3.76	0.27	94.72
10	0	✓	80.82%	3.58	0.27	94.79
30	0	✓	80.47%	3.53	0.28	94.77
60	0	✓	80.15%	3.58	0.28	94.73
MLP	-	-	78.95%	4.40	0.28	94.53

Table 2: Sketch reconstruction performance based on the generated locations by diffusion models for different sampling settings. 'L.T.' denotes location truncation.

Why diffusion model? To demonstrate the effectiveness of the diffusion model, we train an MLP to directly predict the stroke locations as a competitor. Its architecture is identical to the noise predictor's U-Net and we fixed the noise input and time step to '0'. Compared with MLP, the autoregressive generation process of the diffusion model is better suited to capturing the structural relationship between sketch strokes, resulting in a more effective approximation of the original stroke position distribution. As shown in Table 2, the greatest discrepancy between the MLP and diffusion models is in the FID metric. This implies that MLP finds it challenging to accurately anticipate appropriate locations via stroke embeddings to produce sketches resembling human creations.

Model components. The role of components for pretrained models is evaluated. No image decoder is included and no token mixture block is shared between the two decoders denoted by SketchEdit(wo_i) and SketchEdit(wo_s), respectively. Table 3 reports the results of the models containing different components. SketchEdit(full) and SketchEdit(wo_s) with image encoders have performance advantages over SketchEdit(wo_i). This is because the use of image reconstruction allows the network to learn shape information and spatial relationships. Similarly, SketchEdit(wo_s) would make learning image-related information difficult for the token mixture block at the sequence decoder. As shown in Figure 7, some strokes overlap in the results produced by SketchEdit(wo_s) and SketchEdit(wo_i) which reduces the quality of the recreated sketch. In addition, SketchEdit(full) has marginally fewer parameters compared to SketchEdit(wo_s) as it only employs a single token mixture block.

	Model	Rec(↑)	FID(↓)	LPIPS(↓)	CLIP-S(↑)
O.L.	SketchEdit(wo_i)	83.56%	3.80	0.13	96.34
	SketchEdit(wo_s)	84.20%	3.21	0.12	96.45
	SketchEdit(full)	**84.32%**	**3.12**	**0.11**	**96.73**
G.L.	SketchEdit(wo_i)	79.50%	4.22	0.28	94.51
	SketchEdit(wo_s)	79.62%	3.75	0.29	94.55
	SketchEdit(full)	**80.15%**	**3.58**	**0.29**	**94.73**

Table 3: The performance for sketch reconstruction with the original locations and the generated locations. 'O.L.' and 'G.L.' denote original locations and generated locations, respectively.

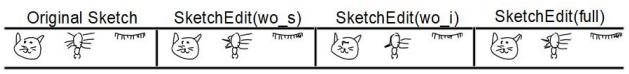

Figure 7: Comparison of recreated sketches across various models.

5 Conclusion

In this paper, we develop the traditional sketch synthesis task to the more controllable sketch editing task at the stroke-level and propose the SketchEdit to realize it. We have focused on decoupling independent strokes from sketches to enable editing operations at the stroke-level. The core of our methodology is to employ the diffusion model to acquire reasonable positions and recreate meaningful sketches based on the strokes. Experimental results demonstrate that SketchEdit can edit sketches without altering categories and facilitate the production of innovative sketches across various categories. Meanwhile, SketchEdit which efficiently preserves the spatial structure of sketches and supports the parallel reconstruction of sketch sequences, surpasses the state-of-the-art methods significantly in preserving visual features of sketches.

Acknowledgements

This work was supported by the Shanghai Municipal Science and Technology Major Project, China (Grant No. 2021SHZDZX0102), and by the National Natural Science Foundation of China (62172273). Shikui Tu and Lei Xu are co-corresponding authors.

References

[Alaniz *et al.*, 2022] Stephan Alaniz, Massimiliano Mancini, Anjan Dutta, Diego Marcos, and Zeynep Akata. Abstracting sketches through simple primitives. In *European Conference on Computer Vision*, pages 396–412. Springer, 2022.

[Chen *et al.*, 2017] Yajing Chen, Shikui Tu, Yuqi Yi, and Lei Xu. Sketch-pix2seq: a model to generate sketches of multiple categories. *arXiv preprint arXiv:1709.04121*, 2017.

[Das *et al.*, 2021] Ayan Das, Yongxin Yang, Timothy M Hospedales, Tao Xiang, and Yi-Zhe Song. Cloud2curve: Generation and vectorization of parametric sketches. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7088–7097, 2021.

[Das *et al.*, 2023] Ayan Das, Yongxin Yang, Timothy M Hospedales, Tao Xiang, and Yi-Zhe Song. Chirodiff: Modelling chirographic data with diffusion models. In *The Eleventh International Conference on Learning Representations*, 2023.

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

[Ge *et al.*, 2020] Songwei Ge, Vedanuj Goswami, Larry Zitnick, and Devi Parikh. Creative sketch generation. In *International Conference on Learning Representations*, 2020.

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

[Hessel *et al.*, 2021] Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning. *arXiv preprint arXiv:2104.08718*, 2021.

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

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

[Jiang *et al.*, 2016] Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. Variational deep embedding: An unsupervised and generative approach to clustering. *arXiv preprint arXiv:1611.05148*, 2016.

[Kingma and Welling, 2013] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[Li *et al.*, 2024] Tengjie Li, Sicong Zang, Shikui Tu, and Lei Xu. Lmser-pix2seq: Learning stable sketch representations for sketch healing. *Computer Vision and Image Understanding*, 240:103931, 2024.

[Liu *et al.*, 2021] Hanxiao Liu, Zihang Dai, David So, and Quoc V Le. Pay attention to mlps. *Advances in Neural Information Processing Systems*, 34:9204–9215, 2021.

[Loshchilov and Hutter, 2017] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[Mou *et al.*, 2023] Chong Mou, Xintao Wang, Liangbin Xie, Jian Zhang, Zhongang Qi, Ying Shan, and Xiaohu Qie. T2i-adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models. *arXiv preprint arXiv:2302.08453*, 2023.

[Pourreza *et al.*, 2023] Reza Pourreza, Apratim Bhattacharyya, Sunny Panchal, Mingu Lee, Pulkit Madan, and Roland Memisevic. Painter: Teaching auto-regressive language models to draw sketches. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 305–314, 2023.

[Qi *et al.*, 2021] Yonggang Qi, Guoyao Su, Pinaki Nath Chowdhury, Mingkang Li, and Yi-Zhe Song. Sketchlattice: Latticed representation for sketch manipulation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 953–961, 2021.

[Qi *et al.*, 2022] Yonggang Qi, Guoyao Su, Qiang Wang, Jie Yang, Kaiyue Pang, and Yi-Zhe Song. Generative sketch healing. *International Journal of Computer Vision*, 130(8):2006–2021, 2022.

[Qu *et al.*, 2023] Zhiyu Qu, Yulia Gryaditskaya, Ke Li, Kaiyue Pang, Tao Xiang, and Yi-Zhe Song. Sketchxai: A first look at explainability for human sketches. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23327–23337, 2023.

[Radford *et al.*, 2021] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[Ramesh *et al.*, 2021] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021.

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

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

[Ronneberger *et al.*, 2015] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[Smith and Topin, 2019] Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial intelligence and machine learning for multi-domain operations applications*, volume 11006, pages 369–386. SPIE, 2019.

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

[Song *et al.*, 2018] Jifei Song, Kaiyue Pang, Yi-Zhe Song, Tao Xiang, and Timothy M Hospedales. Learning to sketch with shortcut cycle consistency. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 801–810, 2018.

[Song *et al.*, 2020] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

[Su *et al.*, 2020] Guoyao Su, Yonggang Qi, Kaiyue Pang, Jie Yang, and Yi-Zhe Song. Sketchhealer a graph-to-sequence network for recreating partial human sketches. In *Proceedings of The 31st British Machine Vision Virtual Conference (BMVC 2020)*, pages 1–14. British Machine Vision Association, 2020.

[Tolstikhin *et al.*, 2021] Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in neural information processing systems*, 34:24261–24272, 2021.

[Vinker *et al.*, 2022] Yael Vinker, Ehsan Pajouheshgar, Jessica Y Bo, Roman Christian Bachmann, Amit Haim Bermano, Daniel Cohen-Or, Amir Zamir, and Ariel Shamir. Clipasso: Semantically-aware object sketching. *ACM Transactions on Graphics (TOG)*, 41(4):1–11, 2022.

[Voynov *et al.*, 2023] Andrey Voynov, Kfir Aberman, and Daniel Cohen-Or. Sketch-guided text-to-image diffusion models. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–11, 2023.

[Wang *et al.*, 2023] 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.

[Xing *et al.*, 2023] Ximing Xing, Chuang Wang, Haitao Zhou, Jing Zhang, Qian Yu, and Dong Xu. Diffsketcher: Text guided vector sketch synthesis through latent diffusion models. *arXiv preprint arXiv:2306.14685*, 2023.

[Xu *et al.*, 1993] Lei Xu, Adam Krzyzak, and Erkki Oja. Rival penalized competitive learning for clustering analysis, rbf net, and curve detection. *IEEE Transactions on Neural networks*, 4(4):636–649, 1993.

[Xu *et al.*, 2022] Peng Xu, Timothy M Hospedales, Qiyue Yin, Yi-Zhe Song, Tao Xiang, and Liang Wang. Deep learning for free-hand sketch: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 45(1):285–312, 2022.

[Yu *et al.*, 2015] Qian Yu, Yongxin Yang, Yi-Zhe Song, Tao Xiang, and Timothy Hospedales. Sketch-a-net that beats humans. *arXiv preprint arXiv:1501.07873*, 2015.

[Yu *et al.*, 2016] Qian Yu, Feng Liu, Yi-Zhe Song, Tao Xiang, Timothy M Hospedales, and Chen-Change Loy. Sketch me that shoe. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 799–807, 2016.

[Zang *et al.*, 2021] Sicong Zang, Shikui Tu, and Lei Xu. Controllable stroke-based sketch synthesis from a self-organized latent space. *Neural Networks*, 137:138–150, 2021.

[Zang *et al.*, 2023a] Sicong Zang, Shikui Tu, and Lei Xu. Linking sketch patches by learning synonymous proximity for graphic sketch representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 11096–11103, 2023.

[Zang *et al.*, 2023b] Sicong Zang, Shikui Tu, and Lei Xu. Self-organizing a latent hierarchy of sketch patterns for controllable sketch synthesis. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[Zhang and Agrawala, 2023] Lvmin Zhang and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. *arXiv preprint arXiv:2302.05543*, 2023.

[Zhang *et al.*, 2018] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

[Zhou *et al.*, 2018] Tao Zhou, Chen Fang, Zhaowen Wang, Jimei Yang, Byungmoon Kim, Zhili Chen, Jonathan Brandt, and Demetri Terzopoulos. Learning to doodle with deep q networks and demonstrated strokes. In *British Machine Vision Conference*, page 4, 2018.