

Self-Supervised Learning for Enhancing Spatial Awareness in Free-Hand Sketch

Xin Wang¹, Tengjie Li¹, Sicong Zang^{1,2}, Shikui Tu^{1*} and Lei Xu^{1,3*}

¹Dept. of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China

²School of Computer Science and Technology, Donghua University, Shanghai 201620, China

³Guangdong Institute of Intelligence Science and Technology, Zhuhai, Guangdong 519031, China

{wang-x, 765127364}@sjtu.edu.cn, sczang@dhu.edu.cn, {tushikui, leixu}@sjtu.edu.cn

Abstract

Free-hand sketch, as a versatile medium of communication, can be viewed as a collection of strokes arranged in a spatial layout to convey a concept. Due to the abstract nature of the sketches, changes in stroke position may make them difficult to recognize. Recently, Graphic sketch representations are effective in representing sketches. However, existing methods overlook the significance of the spatial layout of strokes and the phenomenon of strokes being drawn in the wrong positions is common. Therefore, we developed a self-supervised task to correct stroke placement and investigate the impact of spatial layout on learning sketch representations. For this task, we propose a spatially aware method, named SketchGloc, utilizing multiple graphs for graphic sketch representations. This method utilizes grids for each stroke to describe the spatial layout with other strokes, allowing for the construction of multiple graphs. Unlike other methods that rely on a single graph, this design conveys more detailed spatial layout information and alleviates the impact of misplaced strokes. The experimental results demonstrate that our model outperforms existing methods in both our proposed task and the traditional controllable sketch synthesis task. Additionally, we found that SketchGloc can learn more robust representations under our proposed task setting. The source code is available at <https://github.com/CMACH508/SketchGloc>.

1 Introduction

Free-hand sketch is a versatile medium of communication, conveying subjective impressions of the objective world and serving as a vehicle for emotional expression [Xu *et al.*, 2022]. It is a unique visual form which can be viewed as a collection of strokes arranged in a particular spatial layout, including relative positions, orientations, inclusions, and more. The spatial layout of strokes is an important factor, because even minor changes in stroke position may make them difficult to recognize. Due to time constraints and the manual

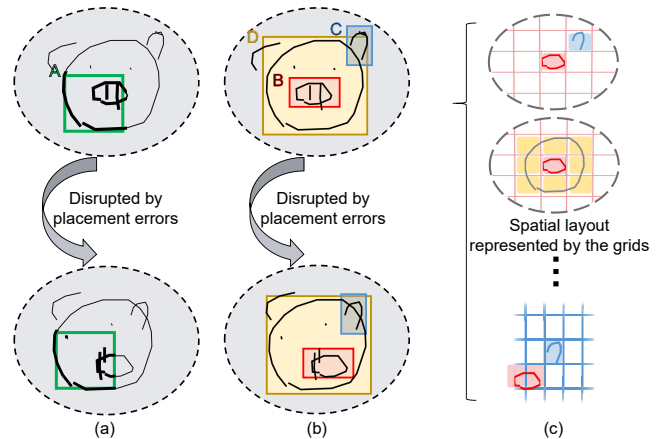


Figure 1: (a) Compared to the original sketch (Above), the sketch disturbed by placement errors (Below) exhibits significant semantic changes within the same local region (area A). (b) With the same placement errors, the spatial layout between strokes maintains the majority of information, including the relative positions, orientations, and inclusion relationships of boxes A, B, C. (c) Each stroke has distinct grids, distinguished by colors (red, blue). In the above image, one stroke (C) is in a cell of another stroke (B), altering the state (color) of the occupied cell (top-right). This is an intuitive grids representation of the (B, C) spatial layout. Subsequently, strokes will be linked based on specific occupied cell in specific graph(top-right). The middle image (B, D) and the bottom image (C, D) follow a similar pattern.

nature of hand drawing [Ha and Eck, 2018], sketches can be variable, abstract, and iconic, making it a challenge to learn robust and accurate sketch representations.

Recently, graphic sketch representations [Qi *et al.*, 2021; Su *et al.*, 2020; Zang *et al.*, 2023] have been proven in representing sketches effectively. These methods use local visual cues of a sketch as nodes and connect the nodes based on temporal proximity [Su *et al.*, 2020], spatial proximity [Qi *et al.*, 2021], or semantic proximity [Zang *et al.*, 2023] to construct the corresponding graph. Finally, a Graph Convolutional Network (GCN) [Kipf and Welling, 2016] is used to aggregate information from graph nodes. But even minor changes in the spatial layout of strokes can significantly affect these local visual cues.

However, existing methods overlook the significance of the

*Corresponding author

spatial layout of strokes. On one hand, strokes being drawn in the wrong positions is a common phenomenon. Factors such as casual sketching, time constraints, and physical conditions like micrographia [Armstrong and Okun, 2020] in patients with Parkinson’s disease can contribute to these unavoidable placement errors, hindering the comprehension of the sketch by humans or models. On the other hand, in contrast to the spatial layout, visual cues in local regions are susceptible to being disrupted by these placement errors. As shown by the variations in region A in Figure 1(a) and the small changes in the spatial layout of regions B, C, and D in Figure 1(b). Existing methods have difficulties in handling these placement errors of strokes because they construct graph with a specific type of proximity, rather than the overall spatial layout. These issues impede the extraction of spatial layout information and the acquisition of robust sketch representations.

To investigate the relation between the spatial layout of strokes and the acquisition of sketch representations, this paper designs a self-supervised task, called Sketch Reorganization, aimed at rectifying the misplacement of strokes which is common in everyday life. Proposing and addressing this task can aid in humans sketching and understanding sketches. Similarly, it can help acquire more robust sketch representations. Unlike conventional sketch restoration tasks [Zhao *et al.*, 2019; Bhunia *et al.*, 2021; Su *et al.*, 2020; Qi *et al.*, 2022], our task, for the first time, takes into account human-introduced errors at the vector level, which aligns with the sequential nature of sketches. Unfortunately, existing methods face challenges in accomplishing this task due to a lack of overall spatial layout awareness capability, which can alleviate the impact of placement errors.

In this paper, we propose SketchGloc: a spatially aware method with a multi-graphs for graphic sketch representations. To measure the spatial layout between strokes of varying sizes and shapes, we took inspiration from the object detection field’s practice [Zou *et al.*, 2023; Redmon *et al.*, 2016; Liu *et al.*, 2016; Girshick, 2015] of using grids and bounding boxes. The geometric localization module of SketchGloc constructs the spatial layout between strokes as multi-graphs using the grids. Specifically, the strokes are encoded as the nodes and the process of constructing the multiple graphs is as follows. We began by drawing different grids for each stroke. (e.g. red and blue gridlines in Figure 1(c) representing two different grids and strokes). Then, one stroke is placed on the grids of another stroke, which changes the state of occupied grid. (e.g. the colors of the cells occupied by the strokes in Figure 1(c) differ). Finally, two strokes will be linked on a specific graph based on whether a particular cell is occupied. (e.g. in Figure 1(c), two strokes (Above) are linked on the ‘top-right’ graph). Our method is particularly well-suited for sketches, as using grids to represent spatial layout helps alleviate the impact of placement errors on the spatial layout. Additionally, the final sketch representation is synthesized from multiple graphs, which facilitates learning of the overall spatial information of the sketch and produces an accurate representation.

The contributions of the paper can be summarized as follows:

- We design a self-supervised learning task ‘Sketch Reor-

ganization’ that takes the errors of stroke placement into account. This task helps to improve the robustness of the sketch representation.

- We propose SketchGloc, which constructs multi-graphs based on the spatial layout between strokes. By using grids to represent spatial layout, the impact of stroke placement errors is alleviated. This method helps to learn the overall spatial layout information and produces more accurate sketch representations.
- The experimental results show that SketchGloc significantly improves the robustness of representations in controllable sketch synthesis task and outperforms other methods in the Sketch Reorganization task. Additionally, the experiments show the practical potential of SketchGloc in aiding humans to draw and comprehend sketches.

2 Related Work

2.1 Sketch Representation Learning

The field of sketch generation [Ha and Eck, 2017; Zang *et al.*, 2021; Vinker *et al.*, 2022; Vinker *et al.*, 2023; Voynov *et al.*, 2023; Li *et al.*, 2024] has been a subject of extensive research, as sketches can be represented in various ways. Notably, the groundbreaking research conducted by sketch-rnn [Ha and Eck, 2017] has been highly recognized. Its significant contributions have captured the attention of scholars and researchers. This method falls into the encoder-decoder paradigm and consists of two Long Short-Term Memory (LSTM). Additionally, the pixel-based sketch encoders [Chen *et al.*, 2017; Zang *et al.*, 2021; Xu *et al.*, 2020a] have received significant performance improvement due to their remarkable spatial feature extraction ability. It is noteworthy that generative GAN-based [Ge *et al.*, 2020] and Diffusion-based models [Xing *et al.*, 2024; Qu *et al.*, 2023; Wang *et al.*, 2022] are effective for capturing the semantic properties of sketch representations due to their excellent generative performance.

In recent times, Graph Convolutional Network (GCN) based sketch encoders [Qi *et al.*, 2021; Qi *et al.*, 2022; Zang *et al.*, 2023] have emerged as a promising research method, capitalizing on the advantages of integrating temporal and spatial features. SketchLattice [Qi *et al.*, 2021] proposed a method for vector sketch representation, with latticed coordinates [Pan *et al.*, 2020] sampling as nodes, and edges connecting nodes based on similarity in the hidden space. In an effort to exploit the temporal characteristics of sketches, another approach was introduced by [Qi *et al.*, 2022], which incorporates both vector and pixel formats. This method sequentially samples sketches and employs features extracted by a Convolutional Neural Network (CNN) as graph nodes in the order of sampling. Furthermore, the semantic properties of node connectivity were explored by [Zang *et al.*, 2023], building upon the node construction method, resulting in improved experimental performance.

2.2 Self-Supervised Pretext Tasks

Self-supervised learning [Gidaris *et al.*, 2018] has achieved remarkable success in the field of images, excelling in var-

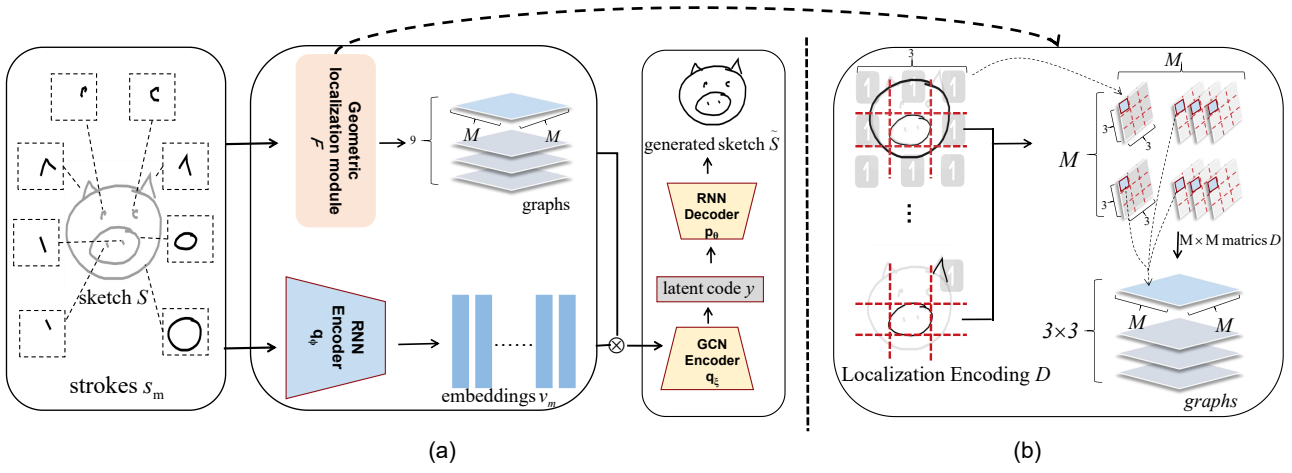


Figure 2: (a) The overview of the SketchGloc approach. Strokes are fed into the LSTM encoder to obtain representations as graph nodes. The spatial geometric localization module computes the spatial layout encoding between strokes and subsequently maps it to graphs. Finally, the GCN encoder calculates the final code, which is utilized by the LSTM decoder to regenerate the sketch. (b) The operation of the spatial geometric localization module. The module constructs a three-by-three grids centered on the current stroke, where each cell represents a piece of spatial layout information. The spatial layout encoding between the current stroke and the target stroke is determined by the latter’s position on this grid.

ious tasks such as classification and semantic segmentation. Defining an appropriate pre-text task is crucial in self-supervised learning. In the field of image analysis, prevalent pretext tasks include predicting relative positions [Doersch *et al.*, 2015], image completion [Doersch *et al.*, 2015], image coloring [Pathak *et al.*, 2016], and deformation prediction [Doersch *et al.*, 2015], among others. In the field of sketch, there are also research efforts on pre-text tasks for self-supervised learning. These tasks differ from those in the image domain, as they must consider the unique data characteristics [Xu *et al.*, 2020b] of sketch.

Existing tasks for self-supervised learning of sketches are akin to those in the image field, primarily focusing on visual feature prediction or complementation [Xu *et al.*, 2020b; Su *et al.*, 2020]. While these methods are tailored for sketch data format, their primary focus remains on leveraging vector features to aid model learning, specifically targeting the prediction of missing visual features.

3 Methodology

Figure 2 provides an overview of the SketchGloc method, which is utilized to achieve robust learning of representations for vector sketches and to address our proposed self-supervised task (Sketch Reorganization) specific to sketch vector features. Initially, we consider the stroke s in a sketch S as the fundamental unit for model input. A sketch S is represented as a sequence of points $(\Delta x, \Delta y, p_1, p_2, p_3)$, where (p_1, p_2, p_3) denotes the state information at the time the sketch is drawn. The values $(1, 0, 0)$, $(0, 1, 0)$, and $(0, 0, 1)$ correspond to touch, lift, and end of pen, respectively. We use the lifted state $(0, 1, 0)$ during a single drawing as a marker for transition from the end of the previous stroke s_{m-1} to the beginning of the new stroke s_m . The processed sketch with M strokes is represented as $S = \{s_m | 1 \leq m \leq M\}$, where each stroke has at most N points. For each stroke

s_m , we feed it into the LSTM, as it has been widely used in previous works, encoder q_ϕ to obtain representations $v_m (1 \leq m \leq M)$ as graph nodes. To establish edges between graph nodes v_m , we utilize the Geometric Positioning Module \mathcal{F} to compute the spatial layout coding D between strokes and subsequently transform it into nine adjacency matrices A , which depends on the grids settings in this paper. This process enables GCN to effectively capture the spatial layout information between strokes. Finally, the GCN encoder q_ξ calculates the final code y , which is utilized by the LSTM decoder p_θ to generate the sketch \tilde{S} .

3.1 Linking Strokes by Geometric Localization Module

As is well known, the spatial layout between two strokes (s_{m_1}, s_{m_2}) can be straightforwardly described by the grids. To address more intricate stroke spatial layout relationships, we leverage multiple spatial layout information. In this paper, a nine-bit spatial layout encoding $D = \{d_i | 1 \leq i \leq 9\}$ is introduced to represent nine distinct spatial layout information. We formalize it:

$$D^{(m_1, m_2)} = \mathcal{F}(s_{m_1}, s_{m_2}), (1 \leq m_1, m_2 \leq M), \quad (1)$$

$$D^{(m_1, m_2)} = \begin{bmatrix} d_1 & d_2 & d_3 \\ d_4 & d_5 & d_6 \\ d_7 & d_8 & d_9 \end{bmatrix}, \quad (2)$$

where $\mathcal{F} : R^{N \times 5} \times R^{N \times 5} \rightarrow F_2^{M \times M \times 9}$. Each element $d_i \in D^{(m_1, m_2)}$ indicates whether s_{m_2} is in the i_{th} cell of the grids of s_{m_1} .

To effectively integrate information in the graph network q_ξ based on the spatial layout between strokes, we split the spatial layout encoding matrix D into nine different graph adjacency matrices $\{A^{(i)} | 1 \leq i \leq 9\}$. Each adjacency matrix $A^{(i)}$ represents one bit of spatial information for each

pair of strokes. At the output layer of the graph network q_ξ , a post-fusion strategy will be implemented to combine the results of message passing in each spatial layout. Formally, we describe this process as follows:

$$a_{m_1, m_2}^{(i)} = d_i \in \mathcal{D}^{(m_1, m_2)}, (1 \leq m_1, m_2 \leq M), \quad (3)$$

$$\mathbf{A}^{(i)} = \begin{bmatrix} a_{1,1}^{(i)} & a_{1,2}^{(i)} & \cdots & a_{1,M}^{(i)} \\ a_{2,1}^{(i)} & a_{2,2}^{(i)} & \cdots & a_{2,M}^{(i)} \\ \vdots & \vdots & \ddots & \vdots \\ a_{M,1}^{(i)} & a_{M,2}^{(i)} & \cdots & a_{M,M}^{(i)} \end{bmatrix}. \quad (4)$$

For a specific graph adjacency matrix $\mathbf{A}^{(i)}$, we impose a constraint that restricts the propagation of message among graph nodes to occur exclusively in the same spatial layout, thereby appropriately adjusting its significance. Furthermore, each adjacency matrix is self-connected, as indicated by $a_{k,k}^{(i)} = 1$.

The spatial layout of strokes obtained through grids is in a discrete form, which aligns well with the construction requirements of graphs. Grids precision can be controlled by adjusting the number of grid cells, such as setting it to 5×5 . However, free-hand sketches are inherently sparse. Therefore, we choose for a 3×3 grid, which is deemed suitable for the current task.

3.2 Graph Networks Based on Stroke-Level Sketches

We utilized spatial layout encoding to construct the sketch GCN Encoder $q_\xi^{(i)}(\mathbf{y}|\mathbf{V}, \mathbf{A}^{(i)})$. To comprehensively utilize all spatial layout information in the spatial layout encoding \mathcal{D} , we employ nine distinct GCN layers $q_\xi^{(i)} \in q_\xi$ to integrate the information from graph nodes in each graph. Each GCN layer $q_\xi^{(i)}$ is constructed for graph nodes $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_M\}$ with adjacency matrices $\mathbf{A}^{(i)}$ to compute the propagation results $\mathbf{h}^{(i)}$ of sketch strokes in the i_{th} grid. The weight of each GCN layer $q_\xi^{(i)}$ is denoted by $\mathbf{W}^{(i)}$:

$$\mathbf{h} = \sum_{i=1}^9 \mathbf{h}^{(i)} = \sum_{i=1}^9 \text{ReLU}(\mathbf{A}^{(i)} \mathbf{V} \mathbf{W}^{(i)}). \quad (5)$$

This final feature \mathbf{h} is then passed through our fully connected layer to yield two vectors $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$. These vectors are computed using the formula $\mathbf{y} = \boldsymbol{\mu} + \boldsymbol{\sigma} \times \boldsymbol{\varepsilon}$, where $\boldsymbol{\varepsilon}$ represents noise sampled from the Gaussian distribution $\mathcal{G}(0, 1)$. The resulting vectors $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ constitute the final code \mathbf{y} for the sketch \tilde{S} .

3.3 Disrupt Vector Sketch

To achieve Sketch Reorganization task, we require a method for creating a sketch with placement errors based on a complete sketch. The random noise $\boldsymbol{\varepsilon}$ follows a Gaussian distribution $\mathcal{G}(\boldsymbol{\varepsilon}|0, 1)$. To control the impact of the noise $\boldsymbol{\varepsilon}$ and maintain the disruption of the sketch at a reasonable level, we introduce an additional parameter ($scale = 10\%, 30\%$). The $scale$ represents a proportion of the sketch size of S , and its

Algorithm 1: Disturbing sketch strokes

Data: $S = s_1, s_2, \dots, s_M, scale$

Result: S

```

1 Initialization:  $i \leftarrow 0$ 
2 Calculate the size of the sketch  $S$ 
  // add noise
3 while  $i \neq n$  do
4    $S_{[i,0]}, S_{[i,1]} \leftarrow x_{abs} + S_{[i,0]}, y_{abs} + S_{[i,1]}$ 
  // Adding noise to each of the
  strokes
5   if  $i \neq 0$  then
6      $\epsilon_x, \epsilon_y \sim \mathcal{G}(0, 1), \mathcal{G}(0, 1)$ 
7     Add noise  $\epsilon_x \times size \times scale, \epsilon_y \times size \times$ 
       $scale$  to  $s_i$ 
8   end
9    $i \leftarrow i + 1$ 
10 end
```

product with the size of the sketch is used as a multiplier for Gaussian noise $\boldsymbol{\varepsilon}$. We accomplish this step by introducing random noise $\boldsymbol{\varepsilon} \sim \mathcal{G}(\boldsymbol{\varepsilon}|0, 1)$ to strokes positions.

3.4 Training a SketchGloc via Sketch Reconstruction

We pass the final code \mathbf{y} of the sketch S to a Recurrent Neural Network(RNN) decoder $p_\theta(S|\mathbf{y})$, which sequentially reconstructs the sketch \tilde{S} . Our training objective is defined as follows:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\xi}|S) = E_{q_{\boldsymbol{\phi}, \boldsymbol{\xi}}(\mathbf{y}|S)}[\log p_\theta(S|\mathbf{y})]. \quad (6)$$

The first likelihood function in the objective necessitates reconstructing the correct input in sequence format. Similarly, for sketch S which is disrupted, we input the decoder with its final code \mathbf{y} , and the RNN decoder p_θ will generate the complete sketch in sequence format.

4 Experiments

We selected controllable sketch synthesis [Zang *et al.*, 2021] and the Sketch Reorganization task proposed by us to validate whether SketchGloc has learned accurate and robust graphic sketch representations.

4.1 Preparation

Datasets. We evaluated SketchGloc on QuickDraw [Ha and Eck, 2018], a large vector sketch dataset containing tens of millions of human free-hand sketches across 345 classes. To account for variability between different sketch classes, we utilized three datasets proposed by [Zang *et al.*, 2021] in our experiments. DS1 and DS2 were taken from [Zang *et al.*, 2021], with DS1 including sketches of bee, word, giraffe, bus, and pig, allowing us to assess sketch generation with large gaps between classes. DS2 encompassed sketches of airplane, angel, apple, butterfly, bus, cake, fish, spider, the Great Wall, and umbrella, presenting more diverse categorical patterns to evaluate the model’s ability to distinguish across multiple categories. DS3 [Qi *et al.*, 2021] includes car, cat

Models	DS1				DS2				DS3			
	Rec(\uparrow)	Ret(top-)(\uparrow)			Rec(\uparrow)	Ret(top-)(\uparrow)			Rec(\uparrow)	Ret(top-)(\uparrow)		
		1	10	50		1	10	50		1	10	50
sketch-rnn	50.33	0.38	2.84	9.33	46.28	10.93	23.73	48.38	57.64	3.72	13.42	26.14
sketch-pix2seq	83.99	13.45	30.12	49.99	85.46	50.94	71.38	80.15	79.13	22.92	47.55	58.19
Song et al.	91.77	16.41	36.43	52.22	86.98	58.84	76.84	80.06	83.28	25.47	43.39	56.16
RPCL-pix2seq	93.18	17.86	38.87	55.30	88.73	53.19	71.60	87.91	81.80	28.80	59.05	77.52
SketchHealer	91.04	58.80	82.15	91.33	94.04	87.54	96.19	98.26	87.03	68.52	82.37	86.57
SketchLattice	75.91	6.55	14.01	26.72	71.80	6.91	14.76	28.82	62.21	5.90	10.36	19.39
SP-gra2seq	95.91	94.88	99.11	99.72	94.85	90.83	98.29	99.08	89.83	94.05	98.72	99.57
ours	96.03	98.75	99.72	99.93	94.34	97.07	98.54	99.18	88.91	98.69	99.51	99.75

Table 1: Controllable sketch synthesis performance (%) on three datasets.



































































												
	10%	30%	10%	30%	10%	30%	10%	30%	10%	30%	10%	30%
Disrupted Sketch												
sketch-pix2seq												
sketchLattice												
SP-gra2seq												
ours												

Figure 3: Examples of diverse models designed for Sketch Reorganization task. We provide a qualitative comparison of results between RNN-based, CNN-based, and GCN-based models, along with the outcomes of SketchGloc.

and horse additionally to DS1, constituting a more challenging data set. For each category, we used 70,000 sketches for training, 2,500 for validation, and 2,500 for testing.

Baselines. Considering the encoder classification [Xu *et al.*, 2022; Li *et al.*, 2020; Lin *et al.*, 2020; Ribeiro *et al.*, 2020] of sketch generation models, we selected six models as baselines for our experiments. The sketch-rnn [Ha and Eck, 2017] model, widely used as a benchmark for tasks involving vector sketch generation, is the most representative model for processing sketches in sequence format. Additionally, sketch-pix2seq [Chen *et al.*, 2017] and RPCL-pix2seq [Zang *et al.*, 2021] take images as input and leverage local structure concept to enhance performance. Moreover, RPCL-pix2seq further imposes constraints on the latent space. Song et al. [Song *et al.*, 2018] model processes both sequences and images, treating each sequence-image pair as a joint input. Sketch-Healer [Su *et al.*, 2020] extracts information from CNN and utilizes a graph network to derive a sketch representation. SketchLattice [Qi *et al.*, 2021] constructs a node representation from point-only Euclidean distance within a latticed

coordinates network to establish node edges. The semantic properties of node connectivity were explored by [Zang *et al.*, 2023], building upon the node construction method, resulting in improved experimental performance. The models were trained using the officially published source code.

We set the number of strokes M , and the batch size N , to 50 and 128, respectively. The Adam optimizer was employed for learning with parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Additional experimental details are provided in the appendix.

4.2 Controllable Sketch Synthesis

Controllable sketch synthesis involves generating accurate input sketches, Model’s performance is evaluated using the metrics Rec and Ret , as proposed in [Zang *et al.*, 2021]. Rec indicates whether the generated sketch \tilde{S} belongs to the same class as its input sketch S . For performance testing, we trained three sketch-a-net [Yu *et al.*, 2017] models on three datasets (DS1,DS2,DS3). Additionally, Ret measures whether the input sketch S and the generated sketch \tilde{S} are encoded as close as possible in the latent space. To achieve

Scale	Models	DS1				DS2				DS3			
		Rec(\uparrow)	Ret(top-)(\uparrow)			Rec(\uparrow)	Ret(top-)(\uparrow)			Rec(\uparrow)	Ret(top-)(\uparrow)		
			1	10	50		1	10	50		1	10	50
10%	sketch-rnn	46.12	0.12	1.06	4.97	51.23	0.19	0.94	3.61	48.98	0.09	0.57	2.78
	sketch-pix2seq	64.34	1.54	7.82	20.15	50.05	0.92	4.49	13.75	48.95	1.02	5.73	14.50
	SketchLattice	64.79	5.04	16.92	28.58	81.37	3.15	12.94	19.92	78.62	3.01	16.32	23.53
	SP-gra2seq	89.45	70.41	93.38	98.63	88.44	82.97	96.13	99.52	85.41	71.67	89.62	95.74
	ours	93.74	98.20	99.16	99.63	91.71	96.79	98.15	98.34	86.64	97.10	98.76	99.18
30%	sketch-rnn	45.21	0.10	1.12	4.53	46.97	0.08	0.51	2.04	48.01	0.02	0.32	1.98
	sketch-pix2seq	54.19	0.96	3.12	12.94	42.04	0.65	3.57	9.19	40.85	0.89	4.01	14.94
	SketchLattice	38.23	0.57	1.78	6.86	34.63	0.96	1.71	4.54	35.19	0.58	1.28	4.68
	SP-gra2seq	73.40	44.27	71.46	89.24	63.89	46.52	71.36	87.60	70.45	48.82	69.54	87.91
	ours	91.70	96.96	98.39	99.52	89.36	96.40	97.74	98.24	85.91	95.08	97.36	99.15

Table 2: Sketch Reorganization performance (%) for the three datasets. Scale denotes the ratio of the variance of the placement errors affecting the strokes in the sketch to the size of the sketch boundary.

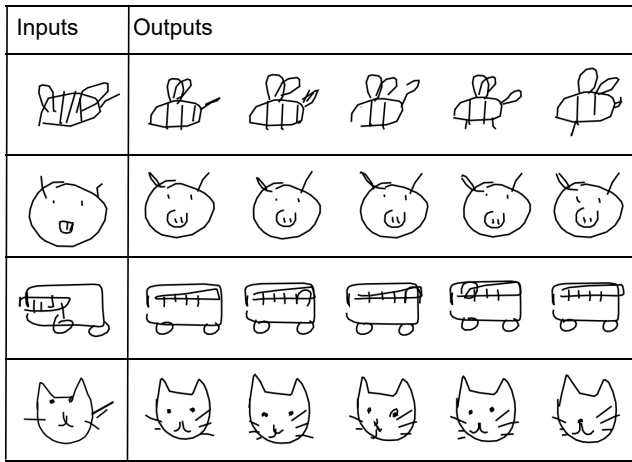


Figure 4: The left column shows low-quality input sketches, while the multiple columns on the right showcase several output sketches generated by SketchGloc. These outputs demonstrate a significant and consistent improvement in sketch quality.

this, we fed the output \tilde{S} of the network, along with the original input S , into the encoder to obtain the latent variables \tilde{y} vs. y . The original y values obtained using the entire test set were used to retrieve \tilde{y} . Ret represents the success rate of the retrieval process, as shown in Figure 5.

Table 1 reports the controllable sketch synthesis performance. SketchGloc considers the use of successive strokes of a sketch as graph nodes and incorporates multiple graphs based on the spatial layout relationships of the strokes, making it better suited for generating vector sketches. By directly encoding the stroke information and passing the sketch over the graph space using a geometric localization adjacency matrix, the method can accurately generate sketches into the latent code. Our approach enables the simultaneous existence of nine graph structures based on different geometric information, leading to accurate sketch representation. As a result, our method is comparable to current state-of-the-art methods.

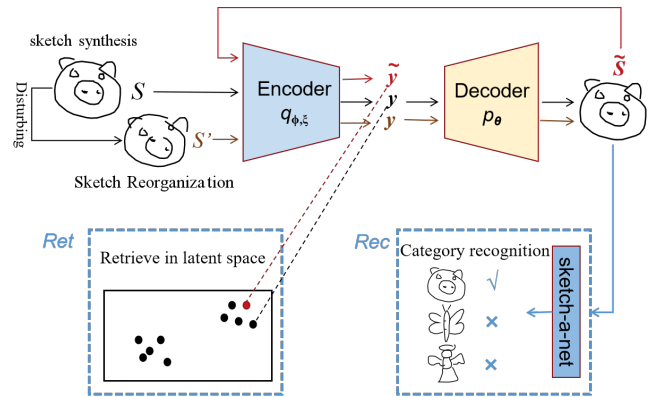


Figure 5: Calculating Rec and Ret for controllable sketch synthesis and Sketch Reorganization.

4.3 Sketch Reorganization

Sketch Reorganization is a task aimed at reorganizing a complete original sketch S using a disturbed sketch S' . The success of Sketch Reorganization is determined by how similar the generated sketch \tilde{S} is to the original sketch S . To evaluate the performance of the models in this task, we use metrics Rec and Ret [Zang *et al.*, 2021], similar to those used in controllable sketch synthesis. These metrics help us answer two key questions: (1) After Sketch Reorganization, how distinguishable are the outputs of different models? (2) Are the latent variable codes stable across models?

As shown in the Table 2, our method outperforms the other methods in Sketch Reorganization. SketchGloc successfully alleviates the impact of placement errors on the spatial layout, carefully considering the possibility of errors in graph node connections. We employ a geometric localization module, a suitable module for obtaining spatial information for sketches, enabling accurate learning of sketch graph representations. Table 2 also reports the metrics of other methods. sketch-rnn [Ha and Eck, 2017] requires high accuracy in sequence data, leading to poor performance in the sketch representation when errors occur in the strokes, hindering

SR task	DS1				DS2				DS3			
	Rec(\uparrow)	Ret(top-)(\uparrow)			Rec(\uparrow)	Ret(top-)(\uparrow)			Rec(\uparrow)	Ret(top-)(\uparrow)		
		1	10	50		1	10	50		1	10	50
no	96.03	98.75	99.72	99.93	94.34	97.07	98.54	99.18	88.91	98.69	99.51	99.75
yes	94.69	99.34	99.86	99.94	92.05	97.15	98.28	99.27	87.57	98.57	99.65	99.87

Table 3: Improved robustness in sketch representation through Sketch Reorganization(SR) task. Evaluation of model performance trained under the Sketch Reorganization task setting to determine if the model was trained with the ‘Disrupt’ option set to ‘yes’ in this task configuration.

Scale	Random Links	DS1				DS2				DS3			
		Rec(\uparrow)	Ret(top-)(\uparrow)			Rec(\uparrow)	Ret(top-)(\uparrow)			Rec(\uparrow)	Ret(top-)(\uparrow)		
			1	10	50		1	10	50		1	10	50
10%	yes	90.36	95.23	97.12	98.67	87.83	94.73	96.21	98.07	82.97	93.12	96.28	98.26
	no	93.74	98.20	99.16	99.63	91.71	96.79	98.15	98.34	86.64	97.10	98.76	99.18
30%	yes	85.94	94.63	95.92	97.60	83.95	92.62	95.58	97.01	80.92	91.51	94.17	97.09
	no	91.70	96.96	98.39	99.52	89.36	96.40	97.74	98.24	85.91	95.08	97.36	99.15

Table 4: Performance implications of the geometric localization module for the Sketch Reorganization task. The method is to connect the graph nodes represented by sketch strokes on sketch graph either in a randomized way or in a spatial layout way. Random connection means that the graph nodes are connected randomly when they are on the sketch graph.

proper reasoning about the semantics of the original sketch. sketch-pix2seq [Chen *et al.*, 2017] and SketchLattice [Qi *et al.*, 2021] methods, sensitive to the visual structure of the sketch, produce inaccurate representations when even small disruptions in the strokes cause significant changes in pixel space. SP-gra2seq [Zang *et al.*, 2023] exhibits performance improvements at *scale* of 10%, leveraging local graph nodes, but struggles to reason effectively about overall sketch semantics at *scale* = 30%.

Additionally, we provide a qualitative comparison in the Figure 3, showing that SketchGloc achieves nearly perfect Sketch Reorganization at *scale* = 10%, resulting in plausible and comparable representations to the original sketch. In contrast, other methods do not reason effectively for most examples, especially at *scale* = 30%, where all methods except ours fail almost completely. SketchGloc still provides satisfactory results for most examples.

Certainly, it is worth mentioning that our method not only accomplishes the Sketch Reorganization task but also possesses the capability to optimize sketches. This is due to the assumption of errors in sketch inputs for the reorganization task. As shown in the Figure 4, quality of recreated sketches are improved. The generated sketches show a consistent and significant improvement in quality, contributing to an improved ability for individuals to both create and understand sketches. This advancement can also optimize the quality of sketches from impaired drawers, such as individuals with Parkinson’s disease [Armstrong and Okun, 2020], assisting in communication or restoring cognitive abilities [Chancellor *et al.*, 2014]. Readers can also optimize their understanding when faced with low-quality sketches.

4.4 Ablation Study

Performance Gained From Sketch Reorganization Task.

As shown in the Table 3, under the Sketch Reorganization

task settings(*SR* = *yes*), representations learned are more robust, achieving higher *Ret* performance in most cases. This is because, in the context of this task, sketch representations learn more robust features, making it less prone to confusion with similar sketches. This indicates the rationality of applying this task to sketch datasets.

Performance Gained From Geometric Localization Module.

In order to investigate whether the introduction of the spatial geometric localization module in SketchGloc facilitates the establishment of relationships between stroke nodes, we conduct a comparative analysis in this section. We contrast Sketch Reorganization outcomes when employing a randomly linked sketch graph. Our experimentation involves calculating results at *scale* = 10% and *scale* = 30%. As presented in Table 4, we observed that the randomly connected graph, achieved by altering the adjacency matrix *A* in Eq.(4), yields inferior Sketch Reorganization performance at *scale* = 10% compared to our proposed method. Furthermore, this adverse impact becomes particularly pronounced when the sketch encounters a greater degree of disruption.

5 Conclusion

This paper introduces the Sketch Reorganization task, which aims to guide machines in correcting placement errors of strokes. To address this issue, we propose the SketchGloc method. This method utilizes spatial layout information to construct multiple graphs, enabling overall spatial layout information and enhancing the accuracy of sketch representations. By using grids to represent spatial layout can accommodate placement errors, the robustness of representation is improved. The effectiveness of SketchGloc has been demonstrated through extensive experiments involving controllable sketch synthesis and Sketch Reorganization.

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 (grants No. 62172273), and the Fundamental Research Funds for the Central Universities (2232024D-28).

References

- [Armstrong and Okun, 2020] Melissa J Armstrong and Michael S Okun. Diagnosis and treatment of parkinson disease: a review. *Jama*, 323(6):548–560, 2020.
- [Bhunia et al., 2021] Ayan Kumar Bhunia, Pinaki Nath Chowdhury, Yongxin Yang, Timothy M Hospedales, Tao Xiang, and Yi-Zhe Song. Vectorization and rasterization: Self-supervised learning for sketch and handwriting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5672–5681, 2021.
- [Chancellor et al., 2014] Bree Chancellor, Angel Duncan, and Anjan Chatterjee. Art therapy for alzheimer’s disease and other dementias. *Journal of Alzheimer’s Disease*, 39(1):1–11, 2014.
- [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.
- [Doersch et al., 2015] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE international conference on computer vision*, pages 1422–1430, 2015.
- [Ge et al., 2020] Songwei Ge, Vedanuj Goswami, C Lawrence Zitnick, and Devi Parikh. Creative sketch generation. *arXiv preprint arXiv:2011.10039*, 2020.
- [Gidaris et al., 2018] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.
- [Girshick, 2015] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.
- [Ha and Eck, 2017] David Ha and Douglas Eck. A neural representation of sketch drawings. *arXiv preprint arXiv:1704.03477*, 2017.
- [Ha and Eck, 2018] David Ha and Douglas Eck. A neural representation of sketch drawings. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.
- [Kipf and Welling, 2016] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.
- [Li et al., 2020] Lei Li, Changqing Zou, Youyi Zheng, Qingkun Su, Hongbo Fu, and Chiew-Lan Tai. Sketch-r2cnn: an rnn-rasterization-cnn architecture for vector sketch recognition. *IEEE transactions on visualization and computer graphics*, 27(9):3745–3754, 2020.
- [Li et al., 2024] Tengjie Li, Sicong Zang, Shikui Tu, and Lei Xu. Lmsr-pix2seq: Learning stable sketch representations for sketch healing. *Computer Vision and Image Understanding*, 240:103931, 2024.
- [Lin et al., 2020] Hangyu Lin, Yanwei Fu, Xiangyang Xue, and Yu-Gang Jiang. Sketch-bert: Learning sketch bidirectional encoder representation from transformers by self-supervised learning of sketch gestalt. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6758–6767, 2020.
- [Liu et al., 2016] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *Computer Vision—ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 21–37. Springer, 2016.
- [Pan et al., 2020] Chen Pan, Yafeng Han, and Jiping Lu. Design and optimization of lattice structures: A review. *Applied Sciences*, 10(18):6374, 2020.
- [Pathak et al., 2016] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.
- [Qi et al., 2021] Yonggang Qi, Guoyao Su, Pinaki Nath Chowdhury, Mingkan Li, and Yi-Zhe Song. Sketchlattice: Latticed representation for sketch manipulation. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 933–941. IEEE, 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, Tao Xiang, and Yi-Zhe Song. Sketchdreamer: Interactive text-augmented creative sketch ideation. *arXiv preprint arXiv:2308.14191*, 2023.
- [Redmon et al., 2016] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.
- [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.
- [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.
- [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 *31st British Machine Vision Conference 2020, BMVC 2020, Virtual Event, UK, September 7-10, 2020*. BMVA Press, 2020.
- [Vinker *et al.*, 2022] Yael Vinker, Ehsan Pajouheshgar, Jessica Y Bo, Roman Christian Bachmann, Amit Haim Bermanto, Daniel Cohen-Or, Amir Zamir, and Ariel Shamir. Clipasso: Semantically-aware object sketching. *ACM Transactions on Graphics (TOG)*, 41(4):1–11, 2022.
- [Vinker *et al.*, 2023] Yael Vinker, Yuval Alaluf, Daniel Cohen-Or, and Ariel Shamir. Clipascene: Scene sketching with different types and levels of abstraction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4146–4156, 2023.
- [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.*, 2022] Qiang Wang, Haoge Deng, Yonggang Qi, Da Li, and Yi-Zhe Song. Sketchknitter: Vectorized sketch generation with diffusion models. In *The Eleventh International Conference on Learning Representations*, 2022.
- [Xing *et al.*, 2024] Ximing Xing, Chuang Wang, Haitao Zhou, Jing Zhang, Qian Yu, and Dong Xu. Diffsketcher: Text guided vector sketch synthesis through latent diffusion models. *Advances in Neural Information Processing Systems*, 36, 2024.
- [Xu *et al.*, 2020a] Peng Xu, Yongye Huang, Tongtong Yuan, Tao Xiang, Timothy M Hospedales, Yi-Zhe Song, and Liang Wang. On learning semantic representations for million-scale free-hand sketches. *arXiv preprint arXiv:2007.04101*, 2020.
- [Xu *et al.*, 2020b] Peng Xu, Zeyu Song, Qiyue Yin, Yi-Zhe Song, and Liang Wang. Deep self-supervised representation learning for free-hand sketch. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(4):1503–1513, 2020.
- [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.*, 2017] Qian Yu, Yongxin Yang, Feng Liu, Yi-Zhe Song, Tao Xiang, and Timothy M Hospedales. Sketch-a-net: A deep neural network that beats humans. *International journal of computer vision*, 122:411–425, 2017.
- [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.*, 2023] 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*, volume 37, pages 11096–11103, 2023.
- [Zhao *et al.*, 2019] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, 30(11):3212–3232, 2019.
- [Zou *et al.*, 2023] Zhengxia Zou, Keyan Chen, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years: A survey. *Proceedings of the IEEE*, 2023.