Linking Sketch Patches by Learning Synonymous Proximity for Graphic Sketch Representation

Sicong Zang, Shikui Tu, Lei Xu

Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China
{sczang, tushikui, leixu}@sjtu.edu.cn

Abstract

Graphic sketch representations are effective for representing sketches. Existing methods take the patches cropped from sketches as the graph nodes, and construct the edges based on sketch's drawing order or Euclidean distances on the canvas. However, the drawing order of a sketch may not be unique, while the patches from semantically related parts of a sketch may be far away from each other on the canvas. In this paper, we propose an order-invariant, semantics-aware method for graphic sketch representations. The cropped sketch patches are linked according to their global semantics or local geometric shapes, namely the synonymous proximity, by computing the cosine similarity between the captured patch embeddings. Such constructed edges are *learnable* to adapt to the variation of sketch drawings, which enable the message passing among synonymous patches. Aggregating the messages from synonymous patches by graph convolutional networks plays a role of denoising, which is beneficial to produce robust patch embeddings and accurate sketch representations. Furthermore, we enforce a clustering constraint over the embeddings jointly with the network learning. The synonymous patches are self-organized as compact clusters, and their embeddings are guided to move towards their assigned cluster centroids. It raises the accuracy of the computed synonymous proximity. Experimental results show that our method significantly improves the performance on both controllable sketch synthesis and sketch healing.

Introduction

Free-hand sketches is a traditional medium for social communication and conveying human emotions. They are vivid and impressive, but are always abstract, iconic and lack-of-details. Though sketches are with various visual appearances, a sketch only consists of several pen strokes. It is challenging to learn accurate and robust sketch representations.

Recently, graphic sketch representation is effective for representing sketches. A sketch is cropped into small patches (Su et al. 2020; Qi et al. 2022b) or constructed as coordinates on lattice (Qi et al. 2021), regarded as the graph nodes. These nodes are linked by edges according to the Euclidean distances on the canvas (spatial proximity) (Qi et al. 2021) or the sketch drawing order (temporal proximity) (Su et al. 2020; Qi et al. 2022b). Usually, graph convolutional network (GCN) (Kipf and Welling 2016) is utilized to aggregate the feature from the node itself and the ones from its neighboring nodes to get the final node representation. Such a GCN-based learning is promising on sketch generation as the neighboring nodes are helpful additional sources for the structural patterns, but its performance largely depends on the principle to connect small patches for message passing.

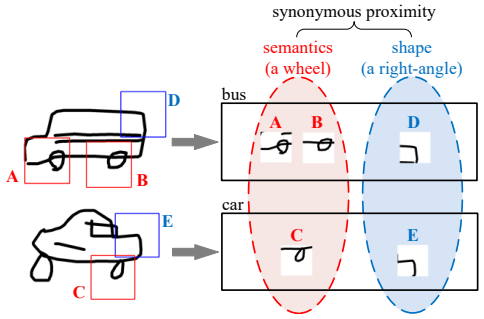


Figure 1: The definition of synonymous proximity. Sketch patches can be semantically related (A, B and C all contain a wheel) or collect pen strokes with analogous shape (D and E are similar in a right-angle-shape). We summarize these neighboring rules as the synonymous proximity.

Neither the temporal nor the spatial proximity is robust against the variation of sketch drawings for patch linking. On one hand, the drawing order of a sketch is not unique. For example, when drawing a cat, its head may be drawn before or after its body, which produces different groups of edges in the constructed graph. On the other hand, the patches from semantically related parts of a sketch may be far away from each other on the canvas. For example, the patches of two ears of a cat may not be linked by an edge. The message from a patch of one ear is difficult to reach a patch of the other ear, as it has to go through a long, multi-hop path (if any). The messages would be diluted or interrupted by other patches from different sketch parts along the path.

It motivates us to link the patches by semantics or geometric shapes. More specifically, if two cropped sketch patches are semantically related (e.g., the patches A, B and C in Fig. 1 all contain a wheel), or their collections of pen strokes share analogous shapes (e.g., D and E in Fig. 1 are similar in a right-angle-shape), we link them by a graph edge and name

this neighboring rule as *synonymous proximity*. Synonym is a term borrowed from the field of linguistics, describing that two words or phrases are with exactly or nearly the same meanings. We use this word to represent the similarity between the patch contents. Compared with the edges linked by temporal or spatial proximity, our principle is *learnable* to adapt to the variation of sketch drawings. Aggregating the messages from the synonymous patches plays a role of denoising, which is beneficial to causally produce robust patch embeddings and accurate sketch representations.

Furthermore, we enforce a clustering constraint on the embeddings of patches cropped from different sketches jointly with the network learning. The synonymous inter-sketch patches (e.g., A and C in Fig. 1) are encoded close together as clusters, and their embeddings are forced to move towards their assigned cluster centroids, which are learned from the entire training set. Thus, the embeddings become compact for computing accurate synonymous proximity.

To realize the above idea, we propose synonymous proximity based graph to sequence (SP-gra2seq)[1] to dynamically link patches for graphic sketch representation. A sketch is cropped into patches which are embedded by a convolutional neural network (CNN) encoder. We construct the sketch graph by synonymous proximity computed by cosine similarity between the patch embeddings. A GCN encoder is devised to aggregate the patch features for the final sketch representation. Moreover, we cluster the inter-sketch patch embeddings jointly with the network training, and push the embeddings towards their assigned cluster centroids to stabilize the estimation of synonymous proximity. To summarize, we make the following contributions:

1. We propose SP-gra2seq to learn graphic sketch representations by linking sketch patches by synonymous proximity. The constructed graph edges are learnable to adapt to the variation of sketch drawings.
2. SP-gra2seq enforces a regularization by clustering the embeddings of inter-sketch patches. The regularization improves the computation of the synonymous proximity.
3. Experimental results show that SP-gra2seq significantly improves the state-of-the-art performance on controllable sketch synthesis (Zang, Tu, and Xu 2021) and sketch healing (Su et al. 2020; Qi et al. 2022b).

Related Work

Representing Sketches by Different Formats

When represented by a sequence of pen strokes, the temporal features of a sketch are captured by the recurrent neural network (RNN)-based (Ha and Eck 2018) or transformer-based (Lin et al. 2020; Ribeiro et al. 2020) encoders. When represented by an image, its spatial relationships among pixels are extracted by CNNs (Chen et al. 2017; Tian et al. 2021). Moreover, both sketch formats are used simultaneously for learning efficient representations (Song et al. 2018; Choi et al. 2019; Xu et al. 2020; Li et al. 2022).

Recently, graph neural networks (GNNs) (Scarselli et al. 2008) and GCNs were utilized for sketch recognition (Yang et al. 2020; Xu, Joshi, and Bresson 2021; Li et al. 2021), sketch segmentation (Yang et al. 2021; Qi et al. 2022a), image retrieval (Zhang et al. 2020b) and sketch synthesis (Su et al. 2020; Qi et al. 2021, 2022b). They focused on learning graphic sketch representations by highlighting the proximity among different parts of a sketch. A sketch was represented by the cropped patches (Su et al. 2020; Qi et al. 2022b) or the latticed coordinates on the canvas (Qi et al. 2021) and were regarded as the graph nodes. These nodes were linked by graph edges based on either the temporal proximity following the sketch drawing order (Su et al. 2020; Qi et al. 2022b) or the spatial proximity revealed by the Euclidean distances (Qi et al. 2021). However, these edge constructions rely on the inherent sketch attributes, which are specific to sketch individuals. The huge variation of sketch drawings may reduce the accuracy for their representations.

Different from the existing methods, we link the sketch patches by the learnable synonymous proximity for accurate, causal and robust sketch representations.

Constraining the Distribution of Sketch Codes

Constructing a proper latent structure for modeling the sketch data manifold contributes to efficient sketch representations. When the sketch codes are single Gaussian distributed, e.g., in sketch-rnn (Ha and Eck 2018), sketches in different categories are chaotically encoded in a latent cluster, reducing the representing performance on multi-categorized datasets. To make the codes more freely encoded, sketch-pix2seq (Chen et al. 2017) removed the Kullback-Leibler (KL) divergence term from the objective. RPCL-pix2seq (Zang, Tu, and Xu 2021) self-organized Gaussian mixture model (GMM) distributed codes to constrain sketches with similar patterns in a compact Gaussian component of GMM.

Inspired by the aforementioned articles, we cluster and regularize the latent embeddings of the sketch patches jointly with the network training. The clustered embeddings are more compact and self-organized, and thus it raises the accuracy of the computed synonymous proximity.

Methodology

SP-gra2seq falls into the encoder-decoder framework. A sketch is cropped into patches which are embedded by a CNN encoder. The sketch graph is constructed based on the learnable synonymous proximity computed by the patch embeddings. The features captured from patches are aggregated by a GCN encoder for the RNN decoder to reconstruct the input sketch.

Linking Sketch Patches by Synonymous Proximity

The **graph nodes** of a sketch are defined as the sketch patches following (Su et al. 2020; Qi et al. 2022b). Firstly, we crop M patches $\boldsymbol{p}_{tm} \in \mathbb{R}^{256\times256}$ $(1 \le m \le M)$ from the sketch image $\boldsymbol{S}_t \in \mathbb{R}^{640\times640}$. The selecting rule for the cropping positions is adopted from (Su et al. 2020; Qi et al. 2022b), but we enlarge the patch size from 128×128 as in (Su et al. 2020; Qi et al. 2022b) to 256×256, ensuring that our patches collect enough drawing strokes for composing meaningful sketch parts, e.g., a cropped wheel in Fig. 1.

[1]The codes are in: https://github.com/CMACH508/SP-gra2seq.

Two patches with similar semantic contents or analogous geometric shape of the drawing strokes are synonymous. For example, the patches A, B and C in Fig. 1 are in close synonymous proximity of one another, as all of them collect a wheel from the vehicles. We employ a CNN encoder $q_{\boldsymbol{\phi}}(\boldsymbol{v}|\boldsymbol{p})$ to capture the embeddings $\boldsymbol{v}_{ti}$ and $\boldsymbol{v}_{tj} \in \mathbb{R}^{512\times 1}$ $(1 \leq i, j \leq M)$ from the patches $\boldsymbol{p}_{ti}$ and $\boldsymbol{p}_{tj}$, respectively. $q_{\boldsymbol{\phi}}$ consists of seven convolutional layers (channel numbers as 8, 32, 64, 128, 256, 512 and 512) with the kernel size 2×2 and the ReLU activation function, followed by max pooling and batch normalization. And we compute the cosine similarity to measure their **synonymous proximity**:

$$\cos(\boldsymbol{v}_{ti}, \boldsymbol{v}_{tj}) = \frac{\boldsymbol{v}_{ti}^{\mathrm{T}}\boldsymbol{v}_{tj}}{||\boldsymbol{v}_{ti}||_2 \cdot ||\boldsymbol{v}_{tj}||_2}, \tag{1}$$

where $||\cdot||_2$ denotes the L2 norm. A large value of $\cos(\boldsymbol{v}_{ti}, \boldsymbol{v}_{tj})$ indicates that the corresponding $\boldsymbol{p}_{ti}$ is in close synonymous proximity to $\boldsymbol{p}_{tj}$. As the patch embeddings are in a high dimensional space (e.g., 512 dimensions in this paper), the ratio of the Euclidean distances of the nearest and farthest neighbors to a given target is almost 1 (Beyer et al. 1999). Hence, we use the cosine similarity as the metric.

Cropped from the same sketch $\boldsymbol{S}_t$, if $\boldsymbol{p}_{ti}$ and $\boldsymbol{p}_{tj}$ are in close synonymous proximity, we link them with a **graph edge**, e.g., linking A and B in Fig. 1. We store the computed synonymous proximity in an adjacency matrix $\boldsymbol{A}_t \in \mathbb{R}^{M\times M}$. The element a_{ij} in $\boldsymbol{A}_t$ denotes the synonymous relationships between $\boldsymbol{v}_{ti}$ and $\boldsymbol{v}_{tj}$, which is

$$a_{ij} = \begin{cases} 1, & j = i, \\ 0.5 \cdot \cos(\boldsymbol{v}_{ti}, \boldsymbol{v}_{tj}), & j = j^*, \\ & j^* = \underset{m\neq i}{\arg\max} \cos(\boldsymbol{v}_{ti}, \boldsymbol{v}_{tm}), \\ 0.2 \cdot \cos(\boldsymbol{v}_{ti}, \boldsymbol{v}_{tj}), & j = j', \\ & j' = \underset{m\neq i, j^*}{\arg\max} \cos(\boldsymbol{v}_{ti}, \boldsymbol{v}_{tm}), \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

For a graph with M nodes, the node $\boldsymbol{v}_{ti}$ is linked to $\boldsymbol{v}_{tj^*}$ and $\boldsymbol{v}_{tj'}$ with the top-2 values of cosine similarity among $\boldsymbol{v}_{tm}$ $(m \neq i)$. a_{ij} is proportional to the value of cosine similarity. As $\boldsymbol{v}_{tj^*}$ is more synonymously related to $\boldsymbol{v}_{ti}$ than $\boldsymbol{v}_{tj'}$, it is offered with a larger coefficient 0.5 to transport more messages from $\boldsymbol{v}_{tj^*}$ to the target $\boldsymbol{v}_{ti}$. Moreover, each node $\boldsymbol{v}_{ti}$ is added with a self-connection, i.e., $a_{ii} = 1$. The rest elements on the ith row of $\boldsymbol{A}_t$ are valued as 0, indicating that these corresponding patches are less synonymous to $\boldsymbol{p}_{ti}$. The graph edges are constructed by the learnable synonymous proximity, according to the dynamic patch embeddings produced from the CNN encoder. It allows SP-gra2seq to automatically link the sketch patches for message passing during the graphic sketch representation learning.

Graphic Sketch Representation

With the constructed sketch graph, we integrate the information of all patch nodes via the synonymous proximity for the final sketch representation. We also resize the full sketch $\boldsymbol{S}_t$ to obtain $\boldsymbol{p}_{t0}$, which is with the same size as $\boldsymbol{p}_{tm}$. Its captured embedding $\boldsymbol{v}_{t0}$ from the CNN encoder $q_{\boldsymbol{\phi}}$ is regarded as a global observation of $\boldsymbol{S}_t$ to cooperate with the local details via the sketch patches $\boldsymbol{p}_{tm}$.

We build the GCN encoder $q_{\boldsymbol{\xi}}(\boldsymbol{y}|\boldsymbol{V}, \tilde{\boldsymbol{A}})$ to compute the final latent code $\boldsymbol{y}_t$ for sketch $\boldsymbol{S}_t$. $\boldsymbol{V}_t = [\boldsymbol{v}_{t0}, \boldsymbol{v}_{t1}, \cdots, \boldsymbol{v}_{tM}]^{\mathrm{T}}$ concatenates all the outputs from the CNN encoder and $\tilde{\boldsymbol{A}}_t \in \mathbb{R}^{(M+1)\times(M+1)}$ is a matrix shown in Eq. (3),

$$\tilde{\boldsymbol{A}}_t = \begin{bmatrix} 0.5 & \mathbf{0}^{\mathrm{T}} \\ 0.5 \cdot \mathbf{1} & \boldsymbol{A}_t \end{bmatrix}, \boldsymbol{A}_t = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1M} \\ a_{21} & a_{22} & \cdots & a_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ a_{M1} & a_{M2} & \cdots & a_{MM} \end{bmatrix}, \tag{3}$$

where $\mathbf{0}$ and $\mathbf{1}$ are two $M \times 1$ vectors with all elements valued as 0 and 1, respectively. And the element a_{ij} in $\boldsymbol{A}_t$ is computed by Eq. (2).

We construct a GCN layer with the weight $\boldsymbol{W}$ as

$$\boldsymbol{F}_t = \mathrm{ReLU}\left(\tilde{\boldsymbol{D}}_t^{-\frac{1}{2}}\tilde{\boldsymbol{A}}_t\tilde{\boldsymbol{D}}_t^{-\frac{1}{2}}\boldsymbol{V}_t\boldsymbol{W}\right), \tag{4}$$

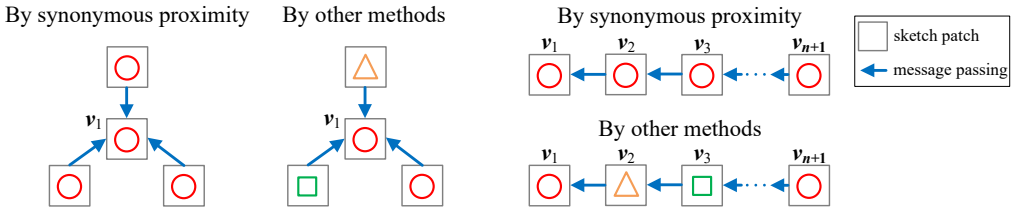
where $\tilde{\boldsymbol{D}}_t$ represents the degree matrix of $\tilde{\boldsymbol{A}}_t$. The embedding $\boldsymbol{v}_{t0}$ extracted from the full sketch is weighted by the split first column of $\tilde{\boldsymbol{A}}_t$ with the weights 0.5 in Eq. (3), cooperating with $\{\boldsymbol{v}_{tm}\}_{m=1}^{M}$ from the sketch patches to yield a comprehensive feature $\boldsymbol{F}_t$. $\boldsymbol{F}_t$ is sent into a pair of fully connected layers to obtain two vectors $\boldsymbol{\mu}_t$ and $\boldsymbol{\sigma}_t$, which are for computing the final code $\boldsymbol{y}_t$ by $\boldsymbol{y}_t = \boldsymbol{\mu}_t + \boldsymbol{\sigma}_t \odot \boldsymbol{\epsilon}$. The operation $\odot$ denotes the Hadamard product and $\boldsymbol{\epsilon}$ is randomly sampled from a standard Gaussian distribution $\mathcal{G}(\boldsymbol{\epsilon}|\mathbf{0}, \boldsymbol{I})$.

We illustrate the relative advantage of our method which is driven by synonymous proximity in Fig. 2, where for convenience we use simple geometric shapes (e.g., circle, triangle, square) to roughly categorize the structural patterns in a patch. For example, in Fig. 2(a), the target node $\boldsymbol{v}_1$ representing a circle receives the messages from its neighbors. If the graph edges are constructed by our synonymous proximity, aggregating the messages from the neighbors of circle plays a role of denoising, which is beneficial to produce robust and accurate embedding of $\boldsymbol{v}_1$. However, if the edges are built by other methods, e.g., the temporal proximity, the passing messages may bring noises (e.g., messages from the square and the triangle) to interrupt the circle representation.

The message passing can form multi-hop paths after going through the stacked GCN layers by Eq. (4). Although two patch nodes, like $\boldsymbol{v}_1$ and $\boldsymbol{v}_{n+1}$ in Fig. 2(b), which are actually very similar to each other, may not have an edge between them for not being at the top-3 neighbors of each other according to Eq. (2), they are still likely to be connected in a multi-hop path. With synonymous proximity, the multi-hop message passing will strengthen the patch embeddings $\boldsymbol{v}_1$ and lead to accurate sketch representations. When using temporal proximity or spatial proximity, it is even more likely to connect the patch nodes of different patterns (e.g., the square and the triangle) in a long multi-hop path, leading to inconsistent, corrupted information propagation.

Clustering Sketch Patches from Different Sketches

Patches cropped from different sketches can be in close synonymous proximity of one another as well, e.g., the patches



(a) 1-hop message passing **(b) *n*-hop message passing**

Figure 2: Message aggregation by GCNs for learning the patch embedding $\boldsymbol{v}_1$. We use simple geometric shapes, e.g., circle, triangle, square, for convenience to roughly categorize the structural patterns in a patch. (a) 1-hop message passing. When using synonymous proximity, a patch with a circle is linked to other ones with circles. The target $\boldsymbol{v}_1$ receives the messages from other circles to achieve a robust circle representation. But in other methods, $\boldsymbol{v}_1$ may receive the messages from a square or a triangle, which could be noises to interrupt the circle representation. (b) n-hop message passing by using stacked GCN layers. Actually, two synonymous nodes $\boldsymbol{v}_1$ and $\boldsymbol{v}_{n+1}$ may not be linked for not being at the top-3 neighbors of each other. But they could be connected in a multi-hop path. With synonymous proximity, multi-hop message passing would strengthen $\boldsymbol{v}_1$. But in other methods, it is more likely to connect the patch nodes of different patterns (e.g., the square) in a long multi-hop path.

A and C in Fig. 1. Though these inter-sketch patches cannot be linked by edges, we enforce a clustering constraint on their embeddings jointly with the network training.

At the (τ)-th training iteration, the network is fed with a mini-batch of N sketches $\{\boldsymbol{S}_t\}_{t=1}^{N}$. Thus, we capture all $N \times M$ embeddings $\boldsymbol{v}_{tm}$ and cluster them in the latent space.

$$q^{(\tau)}(k|\boldsymbol{v}_{tm}) = \begin{cases} 1, & k = \arg\max_{k^*} \cos(\boldsymbol{v}_{tm}, \boldsymbol{c}_{k^*}^{(\tau-1)}), \\ 0, & \text{otherwise.} \end{cases}$$

$$\boldsymbol{c}_k^{(\tau)} = \eta \cdot \frac{\sum_{t=1}^{N}\sum_{m=1}^{M} q^{(\tau)}(k|\boldsymbol{v}_{tm}) \cdot \boldsymbol{v}_{tm}}{\sum_{t=1}^{N}\sum_{m=1}^{M} q^{(\tau)}(k|\boldsymbol{v}_{tm})} + (1-\eta) \cdot \boldsymbol{c}_k^{(\tau-1)}, \tag{5}$$

where $\boldsymbol{c}_k \in \mathbb{R}^{512\times 1}$ $(1 \le k \le K)$ and η denote the cluster centroids and the learning rate for updating $\boldsymbol{c}_k$, respectively. The node $\boldsymbol{v}_{tm}$ is assigned to the k-th cluster, if $\boldsymbol{v}_{tm}$ and $\boldsymbol{c}_k$ are with the largest cosine similarity among all K centroids.

The nodes $\{\boldsymbol{v}_{tm}\}$ with analogous patterns are assigned to the same cluster, wherever $\boldsymbol{v}_{tm}$ is from. Hence, the cluster centroid $\boldsymbol{c}_k$ generalizes the synonymous features from the entire training set. We guide $\boldsymbol{v}_{tm}$ to move towards its assigned cluster centroid to self-organize the compact embeddings, and thus the computation of the synonymous proximity is more robust against the variation of sketch drawings.

Training an SP-gra2seq via Sketch Reconstruction

We send the final code $\boldsymbol{y}_t$ for sketch $\boldsymbol{S}_t$ into the RNN decoder $p_{\boldsymbol{\theta}}(\boldsymbol{S}|\boldsymbol{y})$, whose network architecture is adopted from sketch-rnn (Ha and Eck 2018), to reconstruct the sketch $\hat{\boldsymbol{S}}_t$ in a sequence format. Our objective is to maximize

$$\mathcal{L}(\boldsymbol{\theta},\boldsymbol{\xi},\boldsymbol{\phi}|\boldsymbol{S}) = \sum_{t=1}^{N}\Bigg[\mathrm{E}_{q_{\boldsymbol{\phi},\boldsymbol{\xi}}(\boldsymbol{y}_t|\boldsymbol{S}_t)}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{S}_t|\boldsymbol{y}_t)\right] - \lambda \sum_{m=1}^{M}\sum_{k=1}^{K} q(k|\boldsymbol{v}_{tm}) \cdot ||\boldsymbol{v}_{tm} - \mathrm{sg}(\boldsymbol{c}_k)||_2\Bigg], \tag{6}$$

where $\mathrm{sg}(\cdot)$ stands for the stop-gradient operator. The first log-likelihood term in Eq. (6) aims to reconstruct the input by the sequence-formed sketch generation, and we calculate this term following (Ha and Eck 2018). The second term weighted by λ is a regularization by pushing $\boldsymbol{v}_{tm}$ towards its assigned cluster centroid $\boldsymbol{c}_k$. We remove the KL divergence term $\mathrm{KL}\left[q_{\boldsymbol{\phi},\boldsymbol{\xi}}(\boldsymbol{y}|\boldsymbol{S}_t)||p(\boldsymbol{y}_t)\right]$ from the objective (Chen et al. 2017; Su et al. 2020; Qi et al. 2021) to encourage the latent code $\boldsymbol{y}_t$ to be more freely encoded.

Experiments

We choose controllable sketch synthesis (Zang, Tu, and Xu 2021) and sketch healing (Su et al. 2020; Qi et al. 2022b) to testify whether our method learns accurate and robust graphic sketch representations.

Preparation

Datasets. Three datasets from *QuickDraw* (Ha and Eck 2018) are selected for experimental comparison. Dataset 1 (DS1) and dataset 2 (DS2) are adopted from (Zang, Tu, and Xu 2021) and (Qi et al. 2021), respectively. DS1 (bee, bus, flower, giraffe and pig) evaluates the representation learning on the large variations of the sketches within the same category. DS2 (airplane, angel, apple, butterfly, bus, cake, fish, spider, the Great Wall and umbrella) evaluates whether the methods are sensitive to categorical patterns. We also introduce the most challenging dataset 3 (DS3), which is an enhanced version of DS1 with three newly introduced categories (car, cat and horse). DS3 evaluates whether the methods can still recognize the sketch categories from the shared non-categorical patterns between categories (e.g., wheels can be found in cars or buses). Each category contains $70K$ training, $2.5K$ valid and $2.5K$ test samples ($1K = 1000$).

Baselines. We compare our SP-gra2seq with eight baseline models. Sketch-rnn (Ha and Eck 2018) learns the sketch representations from sketch sequences. Sketch-pix2seq (Chen et al. 2017) and RPCL-pix2seq (Zang, Tu, and Xu 2021) both use sketch images as input, and constrain the sketch codes with a proper distribution to learn better representations. Song et al. (Song et al. 2018) is fed with sketch sequences and images as pairs simultaneously. SketchHealer

Models	DS1				DS2				DS3			
	Rec	Ret (top-)			Rec	Ret (top-)			Rec	Ret (top-)		
		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
SketchHealer 2.0	93.13	57.19	84.54	90.26	90.94	87.37	94.59	97.60	87.37	50.67	76.11	82.42
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
SketchLattice$^+$	95.18	72.74	91.60	97.14	94.30	90.56	97.78	**99.27**	89.49	87.27	96.82	98.98
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**

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

(Su et al. 2020), SketchLattice (Qi et al. 2021) and SketchHealer 2.0 (Qi et al. 2022b) learn graphic sketch representations by linking graph nodes based on the temporal or spatial proximity. Especially, the source code of SketchHealer 2.0 is not released yet, and the article does not provide the details to develop a differentiable sketch rasterisation module for calculating the perceptual loss term. We employ a bi-directional long-short term memory (Bi-LSTM) to calculate the perceptual loss term over the sketch sequences. Besides, we use the Gumbel-softmax (Jang, Gu, and Poole 2016) to make the sampling process differentiable. Moreover, we also produce an extended SketchLattice named SketchLattice$^+$, whose node embeddings are captured from sketch patches as in SketchHealer, instead of the coordinates on lattice. A lattice of 8×8 is set on the canvas to determine the cropping positions, and the Euclidean distances between them are utilized to construct graph edges as in SketchLattice.

When training an SP-gra2seq, the patch number M, the mini-batch size N, the learning rate η for updating clustering centroids and the weight λ in the objective are fixed at 20, 256, 0.05 and 0.25, respectively. The numbers of cluster centroids K are 30, 50 and 50 for three datasets, respectively. We employ Adam optimizer for the network learning with the parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. And the learning rate starts from 10^{-3} with a decay rate of 0.95 for each training epoch.

Controllable Sketch Synthesis

Controllable sketch synthesis requires the method to generate sketches $\hat{\boldsymbol{S}}_t$ with exact patterns as the input $\boldsymbol{S}_t$. We quantitatively evaluate the performance by metrics Rec and Ret proposed by (Zang, Tu, and Xu 2021). Rec indicates whether the generated sketch $\hat{\boldsymbol{S}}_t$ and its corresponding input $\boldsymbol{S}_t$ are in the same category. We respectively pre-train three sketch-a-net (Yu et al. 2017) as classifiers to calculate Rec for three datasets. Ret represents whether $\hat{\boldsymbol{S}}_t$ is well controlled to preserve both the categorical and the detailed non-categorical patterns from $\boldsymbol{S}_t$. More specifically, when feeding the network with $\boldsymbol{S}_t$, we obtain its latent code $\boldsymbol{y}_t$ and the generated sketch $\hat{\boldsymbol{S}}_t$. By sending $\hat{\boldsymbol{S}}_t$ back to the same encoder, we get its corresponding code $\hat{\boldsymbol{y}}_t$. We retrieve the original $\boldsymbol{y}_t$ from $\boldsymbol{Y} = \{\boldsymbol{y}_t(\boldsymbol{S}_t) | \boldsymbol{S}_t \in \text{test set}\}$ with $\hat{\boldsymbol{y}}_t$, and Ret is the successful retrieving rate. Both Rec and Ret are calculated from the entire test set.

Table 1 reports the controllable sketch synthesis performance. By learning the graphic sketch representations, SketchHealer (Su et al. 2020), SketchHealer 2.0 (Qi et al. 2022b) and SketchLattice$^+$ achieve significant improvement on Ret, compared with the other baselines. The temporal or spatial proximity for constructing sketch graphs improves grasping the sketch details. Thus, the generated sketches resemble the corresponding input, leading to high Ret performance. Especially, SketchHealer 2.0 uses a perceptual loss term to guide the network to preserve more global semantics of sketches, instead of exactly reconstructing the local input details. As these two objectives are trade-off (Qi et al. 2022b), SketchHealer outperforms SketchHealer 2.0 a little on controllable synthesis. Besides, the original SketchLattice (Qi et al. 2021) is a light weight model by learning the representations from the coordinates only. SketchLattice fails to restore the sketch details in the generations, resulting in poor performance on Ret.

The graph edges constructed by SP-gra2seq are equipped with synonymous awareness, which are more adaptive to the divergent sketch drawings, especially when sketches are rich in non-categorical patterns, e.g., in DS3. The passing messages through the edges are more reliable to produce accurate patch embeddings. Hence, SP-gra2seq achieves the best performance on controllable synthesis.

Moreover, Fig. 3 presents a comparison between different principles for constructing the graph edges. When using the synonymous proximity, the blue lines regarded as the graph edges can cross the entire canvas to link the most synonymous patches, e.g., the connected bus wheels and windows in the 2nd column and the connected butterfly wings in the 5th column. The passing messages are more freely activated for efficient and accurate sketch representations. But these sketch parts fail to be linked by temporal or spatial proximity, as they are temporally separated in drawing order or with huge distances on the canvas.

Sketch Healing

Sketch healing requires the method to recreate a full sketch $\hat{\boldsymbol{S}}_t$ from a corrupted sketch $\boldsymbol{S}_t^{\text{m}}$ with masks. If $\hat{\boldsymbol{S}}_t$ resembles the original $\boldsymbol{S}_t$ ($\boldsymbol{S}_t^{\text{m}}$ is corrupted from $\boldsymbol{S}_t$), it is regard-

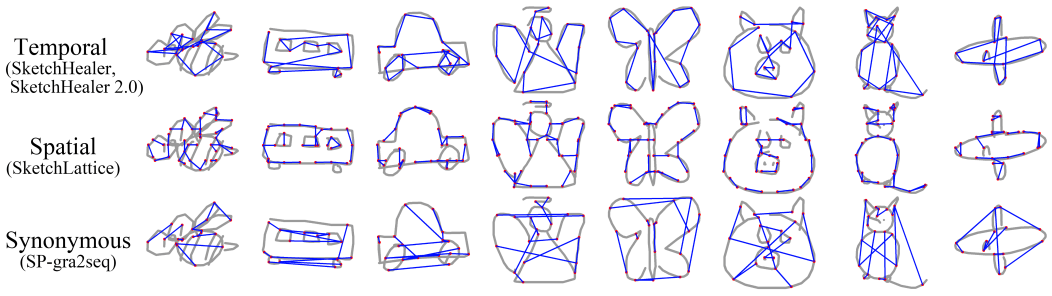


Figure 3: The constructed edge links by temporal, spatial and the proposed synonymous proximity. The red dots denote the patching centers (or the coordinates on lattice) as graph nodes. In this figure, each node is linked to its "nearest" neighboring node with a blue line by the calculated proximity, representing the graph edge.

Mask	Models	DS1				DS2				DS3			
		Rec	*Ret* (top-)			*Rec*	*Ret* (top-)			*Rec*	*Ret* (top-)		
			1	10	50		1	10	50		1	10	50
10%	SketchHealer	70.38	14.25	27.91	45.51	70.56	35.22	55.86	66.24	60.91	15.10	37.89	53.10
	SketchHealer 2.0	90.18	19.45	41.51	59.94	87.07	39.40	64.01	81.20	76.45	15.55	39.99	61.67
	SketchLattice	54.18	1.09	5.48	14.00	52.97	1.34	6.86	17.41	44.14	0.71	4.19	11.36
	SketchLattice$^+$	89.98	19.86	43.59	63.70	90.52	46.98	70.21	83.12	78.28	22.56	44.89	62.11
	SP-gra2seq	**92.90**	**41.24**	**65.74**	**80.16**	**91.24**	**50.42**	**73.35**	**85.18**	**83.38**	**40.20**	**63.40**	**77.52**
30%	SketchHealer	59.00	0.23	3.48	10.76	61.26	7.98	19.04	35.03	48.90	0.43	7.36	15.79
	SketchHealer 2.0	79.05	4.66	14.54	28.44	75.08	10.05	24.51	39.57	60.74	3.75	11.66	22.36
	SketchLattice	32.03	0.08	2.05	5.96	31.73	0.88	3.64	9.25	23.06	0.41	2.41	5.96
	SketchLattice$^+$	70.91	1.02	5.26	14.28	81.76	10.43	26.44	42.90	67.31	2.75	11.13	23.68
	SP-gra2seq	**84.85**	**12.87**	**29.39**	**45.58**	**82.85**	**12.19**	**29.37**	**46.53**	**71.07**	**5.65**	**17.40**	**32.90**

Table 2: Sketch healing performance (%) on three datasets. "Mask" denotes the probability for the sketch patches to be masked.

ed as a successful sketch healing. Especially, our approach for masking sketches is different from SketchHealer, SketchHealer 2.0 and SketchLattice. We firstly select the masking centers on the canvas following the rule for selecting the cropping centers (Su et al. 2020). Each masking center corresponds to a 256×256 patch, which has a probability of 10% or 30% to be masked. Sketch patches cropping or the coordinates selection are proceeded after the masking process. But according to the officially released code of SketchHealer in *GitHub*, the patch cropping and masking are applied by turns in the drawing order. If two patches A and B are overlapped, A is cropped in front of B without being masked, but B is masked. Pixels located in the overlap leak out to patch A, making the corrupted sketches much easier to be represented. SketchLattice applies a point-level masking by randomly dropping a fraction of lattice points. Its masked region is smaller than ours by patch-level masking. Besides, our mask size is 256×256 on the original 640×640 sketch canvas, which is four times larger than the ones (128×128) in both SketchHealer and SketchHealer 2.0. When calculating the metrics for different methods, a sketch is always covered with *the same masks* to make the comparison fair.

We evaluate sketch healing by the similar metrics Rec and Ret, but adjust the Ret calculation to fit this task. Firstly, we randomly mask $\boldsymbol{S}_t$ to obtain the corrupted sketch input $\boldsymbol{S}_t^{\text{m}}$. $\boldsymbol{S}_t^{\text{m}}$ is fed into the network to generate the corresponding $\hat{\boldsymbol{S}}_t$. Secondly, we compute the codes $\boldsymbol{y}_t$ and $\hat{\boldsymbol{y}}_t$ for $\boldsymbol{S}_t$ and $\hat{\boldsymbol{S}}_t$, respectively. Finally, we retrieve $\boldsymbol{y}_t$ from $\boldsymbol{Y} = \{\boldsymbol{y}_t(\boldsymbol{S}_t) | \boldsymbol{S}_t \in \text{test set}\}$ with $\hat{\boldsymbol{y}}_t$ to compute Ret. When calculating the metrics for different methods, a sketch is always covered with the same masks to make the comparison fair. Both metrics are calculated from the entire test set.

Table 2 reports the sketch healing performance. Light weight SketchLattice is heavily sensitive to the coordinate selection due to the limited information input. If the generated sketches contain improper strokes (e.g., the redundant circular strokes), the overlapping points between the improper strokes and the lattice are counted as corruptions, resulting in inaccurate sketch representations.

SketchHealer 2.0 focuses on preserving the global semantics from the corrupted sketches by applying the perceptual loss in the objective, instead of exactly reconstructing the local details. Accordingly, when the sketches are corrupted with a large masking probability, e.g., 30%, SketchHealer 2.0 can still recognize the categorical and the semantic contents from scratch and generates resembled sketches.

For a partially masked patch, SP-gra2seq seeks its synonymous neighbors and constructs their edge linkings. The passing messages may contain the missing information under the masks, which benefits the patch embedding learning to be more comprehensive and accurate. Besides, as the patch embeddings are self-organized in compact clusters, a corrupted patch could still be encoded near the cluster centroid to make the representation robust. Thus, SP-gra2seq achieves the best sketch healing performance.

Fig. 4 presents the qualitative comparisons. When a s-

Mask	Random linkings	DS1				DS2				DS3			
		Rec	*Ret* (top-)			*Rec*	*Ret* (top-)			*Rec*	*Ret* (top-)		
			1	10	50		1	10	50		1	10	50
0%	✓	95.50	66.79	87.81	95.01	90.23	31.56	61.80	80.22	86.90	56.88	80.57	90.64
	×	**95.91**	**94.88**	**99.11**	**99.72**	**94.85**	**90.83**	**98.29**	**99.08**	**89.83**	**94.05**	**98.72**	**99.57**
10%	✓	91.57	38.91	64.67	79.52	87.35	18.53	42.98	63.13	82.63	32.22	56.32	72.01
	×	**92.90**	**41.24**	**65.74**	**80.16**	**91.24**	**50.42**	**73.35**	**85.18**	**83.38**	**40.20**	**63.40**	**77.52**
30%	✓	84.67	11.34	27.75	44.38	76.93	6.42	19.04	35.05	69.65	**8.92**	**22.33**	**36.92**
	×	**84.85**	**12.87**	**29.39**	**45.58**	**82.85**	**12.19**	**29.37**	**46.53**	**71.07**	5.65	17.40	32.90

Table 3: Controllable sketch synthesis (Mask = 0%) and sketch healing performance of SP-gra2seq by linking the sketch patches randomly or by the introduced synonymous proximity. "Random linkings" denotes the sketch patches are randomly linked.

Mask	Clustering constraint	DS1				DS2				DS3			
		Rec	*Ret* (top-)			*Rec*	*Ret* (top-)			*Rec*	*Ret* (top-)		
			1	10	50		1	10	50		1	10	50
0%	×	94.51	80.90	95.09	98.19	85.84	41.00	71.80	87.31	83.93	57.50	83.52	93.45
	✓	**95.91**	**94.88**	**99.11**	**99.72**	**94.85**	**90.83**	**98.29**	**99.08**	**89.83**	**94.05**	**98.72**	**99.57**
10%	×	92.55	40.69	65.64	79.69	90.08	41.08	66.21	80.72	**83.84**	39.03	62.23	76.74
	✓	**92.90**	**41.24**	**65.74**	**80.16**	**91.24**	**50.42**	**73.35**	**85.18**	83.38	**40.20**	**63.40**	**77.52**
30%	×	82.18	10.93	26.52	42.02	77.20	10.80	25.61	45.56	70.35	4.72	15.05	29.47
	✓	**84.85**	**12.87**	**29.39**	**45.58**	**82.85**	**12.19**	**29.37**	**46.53**	**71.07**	**5.65**	**17.40**	**32.90**

Table 4: Controllable sketch synthesis (Mask = 0%) and sketch healing performance of SP-gra2seq by applying the clustering constraint over the inter-sketch patch embeddings or not.

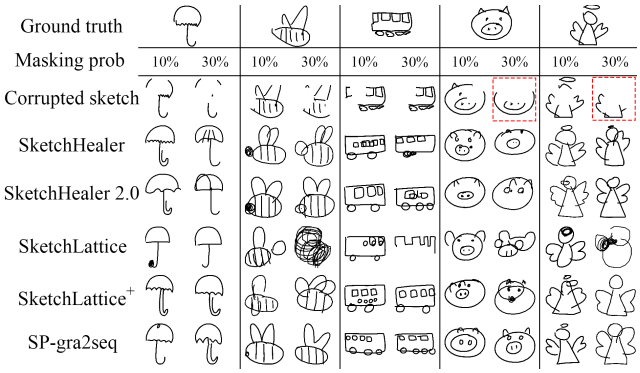


Figure 4: Exemplary sketch healing results. The red dash boxes denote the sketches with masked key characteristics.

ketch is with masked key characteristics, e.g., the pig head with missing top part in the red bounding box, SP-gra2seq recognizes its category from the partial corrupted nose and successfully recreates a pig head. But the others fail. SP-gra2seq is powerful to restore the original sketch details.

Performance Gained from Synonymous Linkings

Synonymous proximity activates the message communications between the sketch patches positioned far away on the canvas by capturing their dynamic long-range dependencies as (Zhang et al. 2020a). This section verifies whether the performance gained by SP-gra2seq is from the connections between the synonymous patches or from the ones between the "non-local" patches. We randomly link the non-local patches by filling the adjacency matrix $\boldsymbol{A}_t$ in Eq. (3) with the values sampled from the uniform distribution $\mathcal{U}(0, 1)$, which activate the communications between the long-ranged patches without considering their synonymous relations. The results are in Table 3. The message passing by random affinities interrupts the model to focus on the local details in target patch. The performance drops especially when the sketches are lightly corrupted. It demonstrates that our performance gain is from the proposed synonymous proximity.

The Impact of Clustering Constraint

We cancel the clustering process in Eq. (5) and remove the second term in Eq. (6). Table 4 offers the results for controllable sketch synthesis and sketch healing. When training SP-gra2seq without the clustering constraint, the performance for both tasks reduces in most cases. It is because the synonymous patches are not encoded in compact clusters in the latent space. If a patch is covered with masks or is challenging to be recognized due to the drawing manner, the captured embedding could be far away from the cluster centroid, where the patch ought to be mapped closely. As a result, the patch embeddings may be unreliable and further reduce the sketch representing performance.

Conclusion

We have presented SP-gra2seq for learning graphic sketch representations. Intra-sketch patches are linked by the learnable synonymous proximity. The message aggregation from the synonymous patches plays a role of denoising for robust patch embeddings and accurate sketch representations. Moreover, we cluster the inter-sketch patch embeddings to yield compact clusters of synonymous patches for computing accurate synonymous proximity. Experiments on controllable sketch synthesis and sketch healing demonstrate the effectiveness of our method.

Acknowledgments

This work was supported by the National Key R&D Program of China (2018AAA0100700) of the Ministry of Science and Technology of China, BirenTech Research and Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102). Shikui Tu and Lei Xu are the corresponding authors.

References

Beyer, K.; Goldstein, J.; Ramakrishnan, R.; and Shaft, U. 1999. When is nearest neighbor meaningful? In *International conference on database theory*, 217–235. Springer.

Chen, Y.; Tu, S.; Yi, Y.; and Xu, L. 2017. Sketch-pix2seq: A model to generate sketches of multiple categories. *arXiv preprint arXiv:1709.04121*.

Choi, J.; Cho, H.; Song, J.; and Yoon, S. M. 2019. Sketchhelper: Real-time stroke guidance for freehand sketch retrieval. *IEEE Transactions on Multimedia*, 21(8): 2083–2092.

Ha, D.; and Eck, D. 2018. A neural representation of sketch drawings. In *International Conference on Learning Representations*.

Jang, E.; Gu, S.; and Poole, B. 2016. Categorical reparameterization with Gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Kipf, T. N.; and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Li, H.; Jiang, X.; Guan, B.; Wang, R.; and Thalmann, N. M. 2022. Multistage spatio-temporal networks for robust sketch recognition. *IEEE Transactions on Image Processing*, 31: 2683–2694.

Li, H.; Jiang, X.; Guarn, B.; and Thalmann, N. M. 2021. Efficient sketch recognition via compact spatial embedding graph neural networks. In *2021 IEEE International Conference on Multimedia and Expo*, 1–6.

Lin, H.; Fu, Y.; Xue, X.; and Jiang, Y.-G. 2020. 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*, 6758–6767.

Qi, A.; Gryaditskaya, Y.; Xiang, T.; and Song, Y.-Z. 2022a. One sketch for all: One-shot personalized sketch segmentation. *IEEE Transactions on Image Processing*.

Qi, Y.; Su, G.; Chowdhury, P. N.; Li, M.; and Song, Y.-Z. 2021. SketchLattice: Latticed representation for sketch manipulation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 953–961.

Qi, Y.; Su, G.; Wang, Q.; Yang, J.; Pang, K.; and Song, Y.-Z. 2022b. Generative Sketch Healing. *International Journal of Computer Vision*, 1–16.

Ribeiro, L. S. F.; Bui, T.; Collomosse, J.; and Ponti, M. 2020. Sketchformer: Transformer-based representation for sketched structure. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 14153–14162.

Scarselli, F.; Gori, M.; Tsoi, A. C.; Hagenbuchner, M.; and Monfardini, G. 2008. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1): 61–80.

Song, J.; Pang, K.; Song, Y.-Z.; Xiang, T.; and Hospedales, T. M. 2018. Learning to sketch with shortcut cycle consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 801–810.

Su, G.; Qi, Y.; Pang, K.; Yang, J.; and Song, Y.-Z. 2020. Sketchhealer a graph-to-sequence network for recreating partial human sketches. In *Proceedings of The 31st British Machine Vision Virtual Conference*, 1–14. British Machine Vision Association.

Tian, J.; Xu, X.; Wang, Z.; Shen, F.; and Liu, X. 2021. Relationship-preserving knowledge distillation for zero-shot sketch based image retrieval. In *Proceedings of the 29th ACM International Conference on Multimedia*, 5473–5481.

Xu, P.; Huang, Y.; Yuan, T.; Xiang, T.; Hospedales, T. M.; Song, Y.-Z.; and Wang, L. 2020. On learning semantic representations for million-scale free-hand sketches. *arXiv preprint arXiv:2007.04101*.

Xu, P.; Joshi, C. K.; and Bresson, X. 2021. Multigraph transformer for free-hand sketch recognition. *IEEE Transactions on Neural Networks and Learning Systems*.

Yang, L.; Sain, A.; Li, L.; Qi, Y.; Zhang, H.; and Song, Y.-Z. 2020. S3net: Graph representational network for sketch recognition. In *2020 IEEE International Conference on Multimedia and Expo*, 1–6. IEEE.

Yang, L.; Zhuang, J.; Fu, H.; Wei, X.; Zhou, K.; and Zheng, Y. 2021. SketchGNN: Semantic sketch segmentation with graph neural networks. *ACM Transactions on Graphics*, 40(3): 1–13.

Yu, Q.; Yang, Y.; Liu, F.; Song, Y.-Z.; Xiang, T.; and Hospedales, T. M. 2017. Sketch-a-net: A deep neural network that beats humans. *International Journal of Computer Vision*, 122(3): 411–425.

Zang, S.; Tu, S.; and Xu, L. 2021. Controllable stroke-based sketch synthesis from a self-organized latent space. *Neural Networks*, 137: 138–150.

Zhang, L.; Xu, D.; Arnab, A.; and Torr, P. H. 2020a. Dynamic graph message passing networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 3726–3735.

Zhang, Z.; Zhang, Y.; Feng, R.; Zhang, T.; and Fan, W. 2020b. Zero-shot sketch-based image retrieval via graph convolution network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 12943–12950.