SketchLattice: Latticed Representation for Sketch Manipulation

Yonggang Qi[1*] Guoyao Su[1*] Pinaki Nath Chowdhury[2] Mingkang Li[1] Yi-Zhe Song[2]
[1]Beijing University of Posts and Telecommunications, CN [2]SketchX, CVSSP, University of Surrey, UK
{qiyg, sgybupt, lmk}@bupt.edu.cn {p.chowdhury, y.song}@surrey.ac.uk

Abstract

*The key challenge in designing a sketch representation lies with handling the abstract and iconic nature of sketches. Existing work predominantly utilizes either, (i) a pixelative format that treats sketches as natural images employing off-the-shelf CNN-based networks, or (ii) an elaborately designed vector format that leverages the structural information of drawing orders using sequential RNN-based methods. While the pixelative format lacks intuitive exploitation of structural cues, sketches in vector format are absent in most cases limiting their practical usage. Hence, in this paper, we propose a lattice structured sketch representation that not only removes the bottleneck of requiring vector data but also preserves the structural cues that vector data provides. Essentially, sketch lattice is a set of points sampled from the pixelative format of the sketch using a lattice graph. We show that our lattice structure is particularly amenable to structural changes that largely benefits sketch abstraction modeling for generation tasks. Our lattice representation could be effectively encoded using a graph model, that uses significantly fewer model parameters (*13.5 *times lesser) than existing state-of-the-art. Extensive experiments demonstrate the effectiveness of sketch lattice for sketch manipulation, including sketch healing and image-to-sketch synthesis.*

1. Introduction

Research on freehand human sketches has become increasingly popular in recent years. Due to its ubiquitous ability in recording visual objects [6], sketches form a natural medium for human-computer interaction. Deriving a tailor-made representation for sketches sits at the core of sketch research, and has direct impact on a series of downstream applications such as sketch recognition [33, 27, 12], sketch-based image retrieval [3, 32, 17, 16], sketch-3D reconstruction [15, 9, 25, 19, 22], and sketch synthesis [14, 23]. Albeit a pivotal component, designing an effective representation is challenging since sketches are typically abstract and iconic.

*Equal contribution

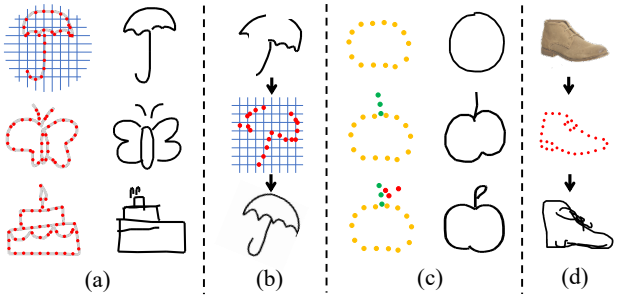

Figure 1. (a) Given lattice points sampled on input sketches (Left), our proposed Lattice-GCN-LSTM network can *recreate* a corresponding vector sketch (Right). (b) Given a corrupted sketch, the resulting lattice points are used to reconstruct a *similar* sketch accordingly. (c) The abstraction level of generated sketches is controllable by varying the density of latticed points. (d) Image-to-sketch synthesis by dropping a few lattice points along the edge of an object.

Prior works predominantly relied on encoding sketch in a pixelative format (i.e., an image) [6, 33, 21]. Although it provided the convenience of using off-the-shelf convolutional neural network effortlessly re-purposed for sketches, pixelative format lacks the intuitive exploitation of structural information. The presence of structural information is vital for sketch abstraction modeling [6, 20], which in turn is essential for downstream tasks that dictate structural manipulation such as sketch generation [7, 4, 10] and sketch synthesis [14, 23].

RNN-based approaches have consequently emerged as means to fully explore the sequential nature of sketches [7]. The research convention is to use *QuickDraw* [8] vector format where each sketch is represented as a list of offsets in x and y. Thanks to the stroke-level modeling, these approaches do offer a degree of flexibility in generation and synthesis tasks, yet they do so by imposing a strong assumption – all sketches have sequential stroke data to them. This assumption largely prohibits the application of RNN-based methods to work with sketches such as those drawn on a piece of paper. A natural question is therefore – is there a way to remove the bottleneck on requiring vector data but

at the same time preserve the structural cues that vector data provides?

To answer this question, we propose an alternative sketch representation inspired by the concept of lattice structures – SketchLattice. We define SketchLattice as a set of points sampled from the original 2D sketch image using a lattice graph as shown in Figure 1 (a). Such latticed sketch representation, although seemingly simplistic, is remarkably amenable to structural deformations, thereby providing vital benefits for sketch abstraction modeling and in subsequent downstream sketch generation tasks. Our proposed latticed representation can be easily and effectively encoded using a simple off-the-shelf graph convolutional network (GCN) [12, 5, 28], resulting in considerably fewer model parameters (13.5 times lesser) as compared to recent state-of-the-art techniques. This not only makes our proposed sketch representation easily deployable, thus making further progress towards practicality, but also reduces the difficulty in optimization and training to give a competitive performance.

Specifically, each point in SketchLattice is regarded as a graph node. Geometric proximity between nodes serves as guiding principle for constructing the adjacency matrix to form graph links. Intuitively, the proposed GCN-based feature extractor learns the topology of points in a sketch object. Despite being simple, our novel sketch representation is surprisingly effective for sketch generation. In particular, using the proposed latticed representation, we show how to recover a corrupted sketch using our Lattice-GCN-LSTM network, as represented in Figure 1(b). Additionally, we present a novel aspect in sketch representation, where the abstraction level in the generated sketch is controllable as shown in Figure 1(c), subject to the density of points sampled by the lattice graph. Furthermore, our method is also applicable to the problem of image-to-sketch synthesis by simply dropping a few key points along the edge of a target object as depicted in Figure 1(d).

Our contributions are summarized as follows: (i) we propose SketchLattice, a novel latticed representation for sketches using an extremely simple formulation i.e., a set of points sampled from a sketch image using a lattice graph. (ii) Our latticed representation can be easily and effectively encoded using a simple graph model that use fewer model parameters, thereby making important progress towards efficiency. (iii) We show how the abstraction level of generated sketches is controllable by varying the density of points sampled from an image using our lattice graph.

2. Related Work

Sketch Data Format There are mainly two types of data formats for sketch representation – image-based and sequential representation. While the former treats sketch as a conventional 2D image with pixel values (i.e., the pixelative format), the latter considers sketch as an elaborately designed set of ordered stroke points (i.e., vector format), represented by offset coordinates along x and y directions with pen states (touch, lift and end) [7]. Traditional sketch feature extractors are usually CNN-based approaches [33, 4] that can directly take sketch image in pixelative format during input. However, this is highly redundant due to the sparsity of line drawings in a sketch image that necessitates heavy engineering efforts [33]. Additionally, CNN-based approaches cannot effectively capture structural cues since it does not encode position and orientation of objects, leading to sub-par results on generation models.

In contrast, sequential representation is sketch-specific [8], designed according to the drawing habit of humans which is constructed stroke-by-stroke. Such sequential, vector format representation allows modeling sketches using RNN-based methods. This resulted in impressive results such as sketch generation and sketch synthesis using long short-term memory (LSTM) [7, 24]. Although promising, such RNN-based approaches require a vectorized data format at the input, which leads to a major bottleneck limiting practical usage in the absence of vector sketches, like sketches drawn on a piece of paper. Therefore, we aim to propose an unexplored technique, by employing a more practical lattice sketch representation, i.e., SketchLattice, that avoids storing stroke orders while still keeping strong spatial evidence that vector sketches typically provide.

Graphical Sketch Embedding Graph convolutional networks (GCNs) [2, 12] were originally designed to deal with structured data, such as knowledge graphs or social networks, by generalizing neural networks on graphs. In the past few years, exciting developments have been made that explore GCNs capabilities for various vision tasks including image classification [5], captioning [31], image understanding [1], action recognition [13], 3D object detection [34], and shape analysis [26]. Yet, until recently, a few attempts [30, 29] started to apply GCNs on sketch embedding. The existing visual sparsity and spatial structure of sketch strokes are naturally compatible with graphical representations. However, the dominating approach in sketch research assumes access to a vector format where the stroke orders are required, thus resulting in a major limitation in real-world cases. On contrary, ours provides a generic approach that explores the geometrical proximity in a latticed representation of sketch. We also show how our proposed graphical sketch embedding can be used additionally for tasks, involving sketch generation and image-to-sketch translation.

3. Methodology

Overview We describe the Lattice-GCN-LSTM network where the central idea is a novel sketch representation tech-

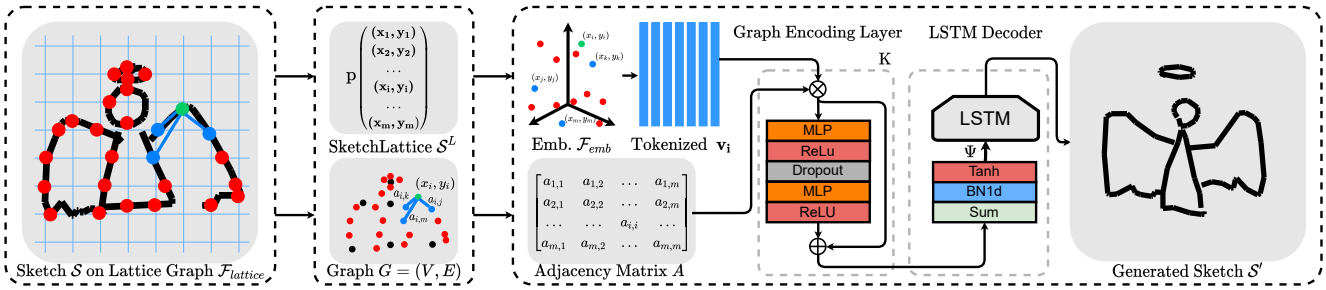

Figure 2. A schematic representation of Lattice-GCN-LSTM architecture. An input sketch image or the edge map of an image object is given to our lattice graph to sample lattice points. All overlapping points between the *dark pixel* in sketch map and uniformly spread lines in lattice graph are sampled. Given the lattice points, we construct a graph using proximity principles. A graph model is used to encode SketchLattice into a latent vector. Finally, a generative LSTM decoder recreates a vector sketch which resembles the original sketch image.

nique that (i) transforms an input 2D sketch image $\mathcal{S}$ into a set of points $\mathcal{S}^L = \{p_1, p_2, \ldots, p_m\}$, using a lattice graph $\mathcal{F}_{lattice}$. Each point $p_i = (x, y)$ in $\mathcal{S}^L$ represents the absolute coordinates x and y in the $\mathcal{S}$. We call $\mathcal{S}^L$ as the lattice format representation of $\mathcal{S}$. (ii) Our novel lattice format $\mathcal{S}^L$ could be seamlessly transformed to a graphical form $G = (V, E)$ that is encoded into a d-dimensional sketch-level embedding vector $\Psi \in \mathbb{R}^d$ using a simple off-the-shelf GCN-based model. (iii) We observe how this sketch-level embedding vector Ψ could help in downstream tasks such as sketch generation by using existing LSTM-based decoding models. Figure 2 offers a schematic illustration.

3.1. Latticed Sketch

The input to our proposed latticed sketch representation is a sketch image $\mathcal{S} \in \mathbb{R}^{w \times h}$ where w and h represent the width and height of $\mathcal{S}$ respectively. We extract the latticed sketch $\mathcal{S}^L$ from $\mathcal{S}$ using the lattice graph $\mathcal{F}_{lattice}$. Our lattice graph $\mathcal{F}_{lattice}$ is a grid that constitutes of uniformly distributed $2n$ horizontal and vertical lines, arranged in a criss-cross manner. The optimal value of n for any given sketch image $\mathcal{S}$, could be empirically determined during inference without further training. As shown in Figure 2, we construct $\mathcal{S}^L$ by sampling the set of all overlapping points $\mathcal{S}^L = \{p_1, p_2, \ldots, p_m\}$ between a *black pixel* in sketch image $\mathcal{S}$ representing a stroke region, and the $2n$ horizontal or vertical lines in $\mathcal{F}_{lattice}$. Formally, we define $\mathcal{S}^L$ as:

$$\mathcal{S}^L = \mathcal{F}_{lattice}(\mathcal{S}) \tag{1}$$

Although extremely simple, this novel latticed sketch representation $\mathcal{S}^L$ is very informative since it can express the topology (i.e., overall structure and shape) of the original sketch image $\mathcal{S}$, without the need of vector data. Additionally, our latticed sketch representation is very flexible because, (i) there is no constrain for the size of input sketch image $(w \times h)$ and the original aspect ratio is maintained, (ii) the abstraction level of the generated sketches modulates depending on the sampling density from the lattice graph $\mathcal{F}_{lattice}$ by varying the value of n. Increasing the value of n would result in more detailed sketches, whereas decreasing it would lead to highly abstract sketches. Figure 1(c) and 4 demonstrate how adding more sample points p_i changes the abstraction level of generated sketches.

3.2. Graph Construction

Graph Nodes V The SketchLattice $\mathcal{S}^L$ can be effectively encoded by a simple graph model which not only consumes fewer model parameters, thereby increasing efficiency, but also allows easier optimization resulting in a model better trained to give a state-of-the-art performance. For each point $p_i \in \mathcal{S}^L$ we calculate $\mathbf{v}_i \in V$ representing elements of the set V, denoting graph nodes. To ensure that the encoding process is amenable to structural changes, each point p_i is tokenized by a learnable embedding function $\mathcal{F}_{emb}(\cdot) : \mathbb{R}^2 \mapsto \mathbb{R}^d$ that maps the absolute point location $p_i = (x, y)$ to a d-dimensional vector space. Formally,

$$\mathbf{v}_i = \mathcal{F}_{emb}(p_i) \tag{2}$$

where $\mathbf{v}_i \in \mathbb{R}^d$ is the resulting tokenized vector representation. Maintaining the original aspect ratio, we resize and pad the input sketch image $\mathcal{S} \in \mathbb{R}^{w \times h}$ to a size of (256, 256) before applying the lattice graph $\mathcal{F}_{lattice}$. Hence, the vocabulary size of the learned embedding function $\mathcal{F}_{emb}$ is 256^2. Our intuition is that, by using an embedding function $\mathcal{F}_{emb}$ that tokenizes each point location (x, y) to a d-dimensional vector $\mathbf{v}_i$, the model would learn to obtain similar embedding features for nearby points. Hence the resulting representation would be more robust to the frequently observed shape deformation and be more amenable that largely benefits sketch abstraction modeling in the generation tasks.

Graph Edges E A straight-forward yet efficacious approach based on the geometric proximity principles is adopted to construct the graph edge links among nodes, based on the corresponding latticed points' locations $p_i =$

(x, y). Specifically, we first compute the euclidean distance between every pair of nodes $(\mathbf{v}_i, \mathbf{v}_j)$ by $d_{i,j} = ||p_i - p_j||_2$. Then we follow either of the two options: (i) Each node $\mathbf{v}_i \in V$ is connected to its *nearest* neighbor or (ii) each node $\mathbf{v}_i \in V$ is connected to its *nearby* neighbors that are *"close enough"*, i.e., $norm(d_{i,j}) < d_T$, where $norm(d_{i,j})$ is a normalized distance in (0,1). d_T is a pre-defined distance threshold whose value is empirically found to be 0.2 in our case. An adjacency matrix $\mathbf{A} \in \mathbb{R}^{m \times m}$ is constructed by setting the link strength to $a_{i,j} = 1 - norm(d_{i,j})$ for a pair of linked nodes (v_i, v_j), such that a smaller distance would result in a larger score. All the disconnected nodes, $a_{i,j}$ are set to 0.

3.3. Graphical Sketch Encoder

Given the graph nodes $V = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m\}$ and their corresponding adjacency matrix $\mathbf{A}$, we employ a simple graph model to compute our final sketch-level latent vector $\Psi \in \mathbb{R}^d$. The resulting vector Ψ allows downstream applications including sketch healing, and image-to-sketch translation. We employ a stack of K identical graph encoding layers, followed by a fully connected (FC) layer, batch normalization and non-linear activation function $Tanh$.

For each i^{th} node $\mathbf{v}_i^k$ in graph encoding layer $k \in [1, K]$, a feature propagation step is executed to produce the updated node feature $\hat{\mathbf{v}}_i^k$, where each node $\mathbf{v}_i^k$ attends to all its linked neighbors with non-zero link strength, defined in the adjacency matrix $\mathbf{A}$. We compute $\hat{\mathbf{v}}_i^k$ as:

$$\hat{\mathbf{v}}_i^k = \sum_{j=1}^{m} a_{i,j} \mathbf{v}_j^k \tag{3}$$

Such a mechanism incorporating spatial awareness not only facilitates message passing among connected nodes, but also adds robustness to missing parts in a lattice sketch while encoding. This greatly benefits downstream tasks such as sketch healing [24]. A graph convolution is applied to the resulting rich spatially dependent feature $\hat{v}_i^k$ as:

$$\mathbf{v}_i^{k+1} = [ReLU(MLP_\Theta(\hat{\mathbf{v}}_i^k))]_{\times 2} \tag{4}$$

where each encoding layer consists of two multi-layer perceptron (MLP) units, both of which is followed by a rectified linear unit (ReLU). We employ dropout and residual connection in each encoding layer as shown in Figure 2. The final feature vectors of nodes from the K^{th} graph encoding layer are integrated into a single vector which is further fed into a sequence of FC layer, batch normalization, and $Tanh$ to compute our sketch-level latent representation $\Psi \in \mathbb{R}^d$.

3.4. Sketch Generation by LSTM decoder

Following [7, 4, 24], we design a generative LSTM decoder that generates the sequential sketch strokes in vector format. Accordingly, the sketch-level latent vector Ψ is projected into two vectors $\mu \in \mathbb{R}^d$ and $\sigma \in \mathbb{R}^d$, then from which we can sample a random vector $z \in \mathbb{R}^d$ by using the reparameterization trick [11] to introduce stochasticity in the generation process via an IID Gaussian variable $\mathcal{N}(0, I)$:

$$\begin{aligned} z &= \mu + \sigma \odot \mathcal{N}(0, I) \\ \mu = W_\mu \Psi, &\quad \sigma = exp(\frac{W_\sigma \Psi}{2}) \end{aligned} \tag{5}$$

where W_μ and W_σ are learned through backpropagation [7]. The latent vector z is used as a condition for the LSTM decoder to sequentially predict sketch strokes. Specifically, the output stroke representation s_{t-1} from the previous time step, together with latent vector z serve as inputs to update the LSTM hidden state h_{t-1} by:

$$h_t = LSTM_{forward}(h_{t-1}; [s_{t-1}, z]) \tag{6}$$

where $[\cdot]$ represents concatenation operation. Next, a linear layer is used to predict a output stroke representation for current time step, i.e., $s_t = W_s h_t + b_s$, where W_s and b_s are learnable weight and bias. The final stroke coordinates are derived from s_t with the help of Gaussian mixture models, to generate the vector sketch format, represented by $\mathcal{S}'$. We refer readers to [7, 8] for more details.

3.5. Model Training and Deployment

Our proposed graphical sketch encoder and the generative LSTM decoder are trained end-to-end for sketch generation. Note that, although we require vector sketches to train the LSTM decoder for the purpose of vector sketch generation, our model fully works on image sketch input, rather than vector data during inference. Following [7], the goal is to minimize the negative log-likelihood of the generated probability distribution to explain the training data $\mathcal{S}$, which can be defined as:

$$\min E_{q_\phi(z|\mathcal{S})}[-\log p_\theta(\mathcal{S}|z)] \tag{7}$$

which seeks to reconstruct the vector sketch representation $\mathcal{S}$ from the predicted latent vector z. Upon training, the decoder generates a vector sketch conditioned on the graphical encoded latent vector z, obtained from our lattice sketch $\mathcal{S}^L$ given any image sketch, thus being more effective for practical applications.

4. Experiments

The ability to be amenable to structural changes and enable appropriate abstraction modeling for sketch generation are the two key aspects that our proposed SketchLattice representation aims to address. Specifically, we adopt the challenging task of sketch healing to testify the robustness of our novel sketch representation towards frequently occurring structural deformation in sketches. Additionally, we

also observe how our proposed approach could be utilized to perform the task of image-to-sketch translation.

Implementation details We implement our model on PyTorch [18] using a single Nvidia Tesla T4 GPU. Optimization is performed using the Adam optimizer with parameters $\beta_1 = 0.9$, $\beta_2 = 0.99$ and $\epsilon = 10^{-8}$. Value of learning rate is set to 10^{-3} along with a decay rate of 0.999 in every iteration. A gradient clipping strategy is adopted to prevent gradient from exploding during the training of LSTM decoder. Essentially, we force the gradient value to 1.0 if the actual value is larger than 1.0. The optimal value for the number of graph encoding layer is $K = 2$.

4.1. Sketch Healing

The task of sketch healing [24] was proposed akin to vector sketch synthesis. Specifically, given a partial sketch drawing, the objective is to recreate a sketch which can best resemble the partial sketch.

From full $\mathcal{S}^L$ to partial $\hat{\mathcal{S}}^L$ Given the lattice sketch representation $\mathcal{S}^L$ from an input sketch image $\mathcal{S}$, we randomly drop a fraction of lattice points in $\mathcal{S}^L$ with some probability P_{mask} to generate a partial SketchLattice, represented by $\hat{\mathcal{S}}^L$. Hence, the graph edges linked to the removed node are also disconnected, thereby simultaneously modifying the adjacency matrix $\mathbf{A}$. One can think of P_{mask} as the corruption level of an input sketch image.

4.1.1 Experimental Settings

Dataset Following the footsteps of [27, 24], we use *QuickDraw* [8] for evaluation since it is currently the largest doodle sketch dataset. More specifically, a small subset of 10 categories are selected such that it includes (i) both complex and simple drawings, (ii) intra-category objects having high degree of resemblance among each other, and (iii) presence of common life object categories that contain diverse sub-categories such as `bus` and `umbrella`. In each category we use $70k$ training and $1k$ testing sketches. The selected 10 categories are as follows: `airplane`, `angel`, `apple`, `butterfly`, `bus`, `cake`, `fish`, `spider`, `The Great Wall`, `umbrella`.

Competitors We compare our proposed Lattice-GCN-LSTM network with three most popular alternatives for vector sketch generation: **SketchRNN** (SR) [7], **SketchPix2seq** (Sp2s) [4], and **SketchHealer** (SH) [24]. Input to SketchRNN is a set of offsets in x and y directions from a vector sketch representation. The key to SketchRNN is a sequence-to-sequence model which is trained without the KL-divergence term. This is done to maintain the fairness of comparison since the KL-divergence term has shown to be beneficial for multi-class scenarios [4]. SketchPix2seq on the other hand replaces its encoding module with a CNN-based encoder that accepts a pixelative format i.e., sketch image. It is expected that such a design will help capture better visual information. Note that, although both SketchRNN and SketchPix2seq were not specifically designed for the task of sketch healing, following [24], we employ these techniques since they are procedural-wise compatible after being re-purposed. The only work specifically addressing the task of sketch healing is SketchHealer [24]. Given a vector sketch as input, SketchHealer converts it into a graphical form where each stroke is considered as a node. Next, visual image patches are extracted from each node region. A GCN-based model is applied for encoding a random vector z. The decoding procedure of SketchHealer is identical to ours where z is fed into a generative LSTM decoder to generate the corresponding vector sketch. Additionally, we retrain a variant of SketchHealer only using visual cues (**SH-VC**), for which stroke order is unavailable. Hence, geometric proximity principles are utilized to construct graph edges, similar to ours. This examines the performance of SH [24] in the absence of vector sketches.

Evaluation setup We adopt a similar evaluation setup in [23] and [24] for quantitative evaluation to understand the effectiveness of our novel latticed sketch representation. First, we evaluate the quality of generated vector sketches (transformed to pixelative format), via *sketch recognition accuracy*. A pre-trained multi-category classifier with AlexNet architecture is used, which is trained on the training split of 345 *QuickDraw* categories. Higher recognition accuracy essentially signifies the ability of the network to generate realistic sketches. It also indicates that the network is able to accurately model the underlying data distribution by effectively encoding a sketch into an accurate and informative sketch representation. We use 1000 testing sketches from each of the 10 selected categories for a thorough evaluation. Second, we judge the recognizability of the encoded sketch-level latent vector Ψ by performing a *sketch-to-sketch retrieval task*. The objective is, given the encoded representation of a sketch Ψ, we expect to retrieve sketches of the same category from a gallery of sketches. A higher retrieval accuracy signifies that the network has a strong *sketch healing* ability, due to its amenable and robust sketch representation.

4.1.2 Results

Qualitative Results We illustrate some examples produced by our lattice-based sketch generator under different values of P_{mask} in Figure 3. We can observe that (i) our latticed representation is robust to partial missing parts such that ours can still generate a novel complete sketch even up to $P_{mask} = 30\%$. (ii) The generated sketch is sensitive to the number of the obtained sampling points, where more points lead to greater details in the generated sketch. Tak-

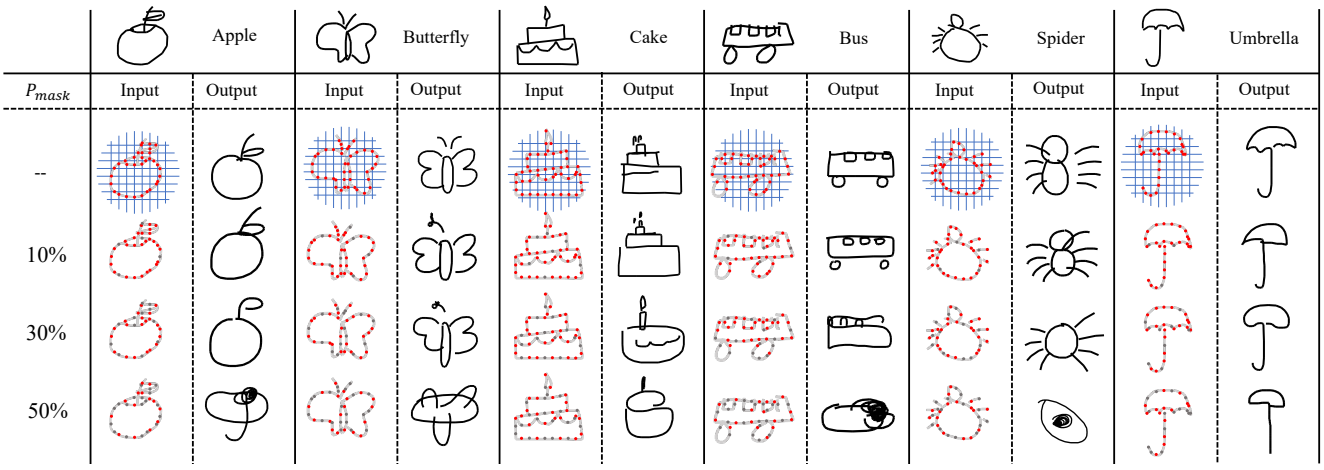

Figure 3. Exemplary results of generated sketch from SketchLattice under different corruption level of mask probability P_{mask} in *Quick-Draw* dataset. With an increase of P_{mask}, the generated sketch becomes more abstract. For $P_{mask} \leq 30\%$ we observe satisfactory generated sketches, but for $P_{mask} = 50\%$, the generated new sketches are struggle to faithfully recover the original sketch.

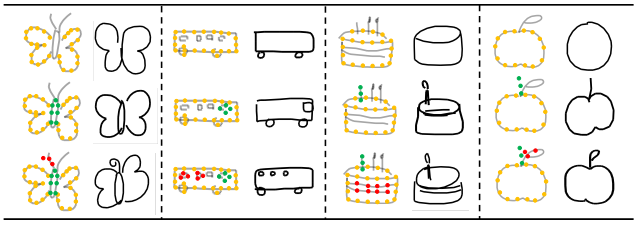

Figure 4. Examples showing how adding more lattice points in different stages (color coded), result in the Lattice-GCN-LSTM network to progressively generate more detailed representation of the category.

ing the example of `cake` as shown in Figure 3, we observe how the bottom and candle regions are simplified when increasing P_{mask}. (iii) On further lifting P_{mask} to 50%, our model can hardly generate a satisfactory sketch, given the severe absence of input sampling points.

We further show that the abstraction level of generated sketch modulates when varying the number and position of input lattice points, as shown in Figure 4. For example, if we only drop points to form the wings of butterfly, the resulting generated butterfly will be extremely simple. While the body and antenna start to show up when we produce more points to convey the intention of such corresponding details. Similar trends can be found in other classes as well.

Quantitative Results As discussed in Section 4.1.1, we compare the performance of different models under the two metrics (recognition accuracy and Top-1 retrieval) as shown in Table 1. We can observe from Table 1 that our approach outperforms other baseline methods on recognition accuracy, suggesting that the healed sketches obtained from ours are more likely to be recognized as objects in the correct categories. Importantly, we can also observe that, unlike

Table 1. Recognition accuracy (Acc) and Retrieval results (Top-1) according to different corruption levels (P_{mask}). We use "nearby" proximity principle and optimal value of $n = 32$. Notice that neither vector format (VF) input nor visual cues (VC) are required by using our method during test.

Method	VF	VC	#Params	P_{mask}	Acc	Top-1
SR [7]	✓	✗	0.67 M	10%	25.08%	50.65%
				30%	3.44%	43.48%
Sp2s [4]	✗	✓	1.36 M	10%	24.26%	45.20%
				30%	10.54%	27.66%
SH [24]	✓	✓	1.10 M	10%	50.78%	85.74%
				30%	43.26%	85.47%
SH-VC [24]	✗	✓	1.10 M	10%	-	58.48%
				30%	-	50.87%
Ours	✗	✗	0.08 M	10%	55.50%	76.02%
				30%	54.79%	73.71%

other competitors which are very sensitive to the corruption level, ours can maintain a stable recognition accuracy even when P_{mask} increases up to 30%. For the task of sketch-to-sketch retrieval, we can see that ours achieves the second best, which is inferior to SketchHealer. However, SketchHealer depends heavily on stroke order provided by vectorized sketches, evidenced by the dramatic decrease of retrieval performance (Table 1 SH vs SH-VC). This signifies the importance and superiority of our approach in the absence of vector input, which is a common practice in real-world cases. In addition, our network is much lighter than the other competitors having far less parameters (13.5 times lesser than SketchHealer), since our approach avoids the use of expensive CNN-based operation. We further include two classes `circle` and `clock` as distraction to `apple` for evaluation. A slightly better result 77.80% (vs 76.02%) can be observed (12 classes, P_{mask}=10%), suggesting good

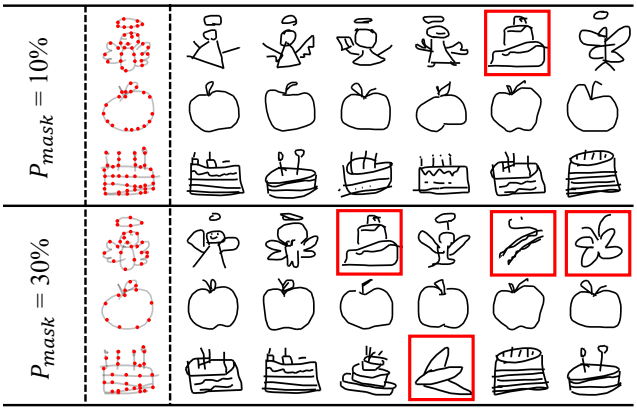

Figure 5. Qualitative retrieval results (Top 6) for sketch-to-sketch retrieval for *QuickDraw* categories under different corruption levels of P_{mask}. Red bounding boxes denote false positive.

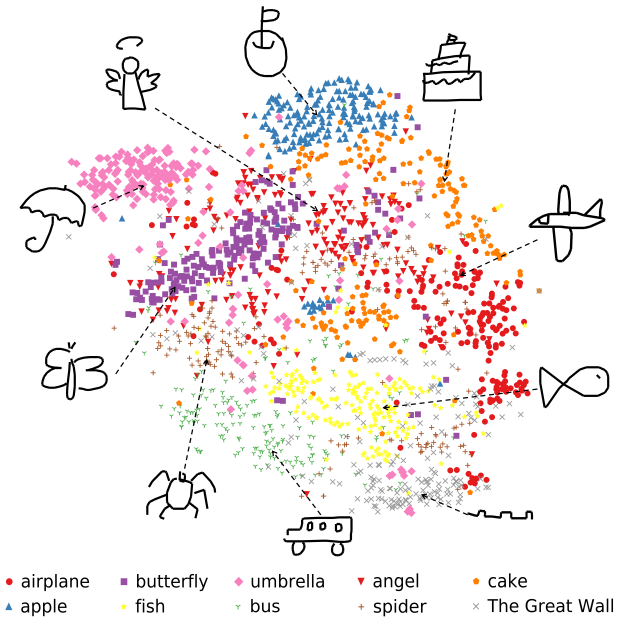

Figure 6. T-SNE plot on sketch-level latent vector Ψ for the selected 10 *QuickDraw* categories, that demonstrate the discriminative power of our lattice-based graphical encoder. Intra-category instances tend to cluster together, suggesting category-level discriminative ability.

scalibility and robustness of our latticed representation. Figure 5 shows some examples of sketch-to-sketch retrieval.

Visualization of Ψ To further visualize the discriminative power of our lattice-based graphical encoder, we randomly select 100 sketches from each class in the test set, and visualize their latent vectors Ψ using t-SNE in Figure 6. We observe that intra-category instances tend to cluster together, suggesting a category-level discriminative ability.

Ablation Study A thorough ablative study using sketch recognition accuracy, is conducted to verify the effectiveness of our different design choices, such as (i) the size of lattice graph, (ii) proximity principle for graph construction, and (iii) importance of residual connection used in our graphical sketch encoder. As shown in Table 2, we can observe that (i) increasing the value of n, that will simultaneously increase density of sampling lattice points from $\mathcal{S}^L$, directly transpires in an improvement of recognition accuracy. From $n > 32$ we start to observe saturation of performance, thereby indicating the optimal value for $n = 32$. (ii) For construction of graph, we observe that more neighbors (nearby) works better than using only the nearest one for graph construction. (iii) removing residual connection leads to significant drop in performance, thereby establishing its importance.

Table 2. Ablative study (Top-1 and Top-3 recognition accuracy) on *QuickDraw* measuring the contribution of (i) sampling density or Grid n from our lattice graph $\mathcal{F}_{lattice}$ (ii) residual connection in latticed-based graphical encoder (iii) effective proximity principle to construct graph from lattice points; for different corruption values of $P_{mask} = \{10\%, 30\%\}$.

	Grid	Residual	Proximity	Top-1	Top-3
Ours (P_{mask} = 10%)	8	✗	nearest	26.56%	33.77%
		✓	nearest	36.16%	45.82%
		✓	nearby	38.86%	51.36%
	16	✗	nearest	14.47%	19.12%
		✓	nearest	40.64%	50.05%
		✓	nearby	42.56%	51.71%
	32	✓	nearby	**55.50%**	**64.72%**
	64	✓	nearby	45.71%	53.56%
Ours (P_{mask} = 30%)	8	✗	nearest	16.97%	24.02%
		✓	nearest	34.70%	43.76%
		✓	nearby	36.41%	47.75%
	16	✗	nearest	13.76%	18.43%
		✓	nearest	40.59%	49.56%
		✓	nearby	39.07%	48.41%
	32	✓	nearby	**54.79%**	**64.74%**
	64	✓	nearby	45.07%	53.50%

On Complex Sketches To testify the robustness and applicability of our proposed latticed representation, we further investigate the performance when dealing with complex data. Specifically, we examine the recognition accuracy of the most complicated sketches (top 25%) over all categories, based on the number of strokes. From the results shown in Table 3, we can see that ours outperforms other competitors when healing the most complex sketches.

Table 3. Sketch recognition accuracy on the most complex cases, i.e., top 25% sketches (stroke-wise) over all categories.

P_{mask}	SR [7]	Sp2s [4]	SH [24]	Ours
10%	0.26	0.18	0.39	**0.43**
30%	0.03	0.08	0.37	**0.42**

Human Study To gain more insights about the fidelity of the healed sketches, a human study is additionally conducted. We recruited 10 participants. 50 sketch samples across all 10 classes were randomly selected. Each sample has two corrupted instances associated, at mask ratio 10% and 30%, respectively. For each corrupted sketch, we generate a group of healed sketches using different methods (SketchHealer, SketchRNN, SketchPix2seq, and ours). We show each participant, the corrupted sketch, and the group of four healed versions in random order. Each participant is then asked to pick a healed sketch that best resembles the corrupted input. Results in Table 4 reveals that according to human, sketches healed by our method resemble the corrupted input the best, at both corruption levels.

Table 4. Human study on fidelity of healed sketches (in %).

P_{mask}	SR [7]	Sp2s [4]	SH [24]	Ours
10%	5.80	11.55	38.98	**43.67**
30%	1.46	7.60	26.70	**64.24**

4.2. Image-to-Sketch Synthesis

Our Lattice-GCN-LSTM network can be applied to image-to-sketch translation. Essentially, given an input image, the corresponding edges are extracted using an off-the-shelf edge extractor [35]. Next, we transform the edge map into a latticed sketch representation using our lattice graph. The resulting SketchLattice could be seamlessly encoded via our graphical encoder. Finally, a vector sketch can be produced using the generative LSTM decoder. Our objective is to generate a sketch that best resembles the ground-truth sketch drawn by humans. Once trained, for any input image, we can obtain some representative lattice points based on the corresponding edges and lattice graph. Then, our generative LSTM decoder can generate a sketch from the sketch-level encoded representation Ψ, as stated in section 3.4.

Experimental Settings We use QMUL-shoe-v2 [32], a fine-grained sketch-based image retrieval dataset, to evaluate our image-to-sketch synthesis approach. In total, there are 6648 one-to-one mappings of image-to-sketch pairs. The dataset is split into two parts, i.e., 6000 pairs for training and the rest of 648 pairs for testing. We chose the value of n for our lattice graph as 32 and adopt the "nearby" strategy for graph construction. LS-SCC [23], a current state-of-the-art, is adopted for comparison.

Human Study We conduct a user study that judges two aspects: (i) reality of the generated sketches, i.e., whether a sketch "looks" like being drawn by a human or not, and (ii) similarity between the produced sketch and its target photo. Specifically, we show triplets of images, i.e., a photo shoe, and two corresponding sketches generated by LS-SCC [23] and ours in random order to 10 *new* participants. Each participant is asked to, (i) choose which of the two sketches "looks" more like a human drawing (REAL), and (ii) identify the sketch that best resembles the photo shoe (SIM).

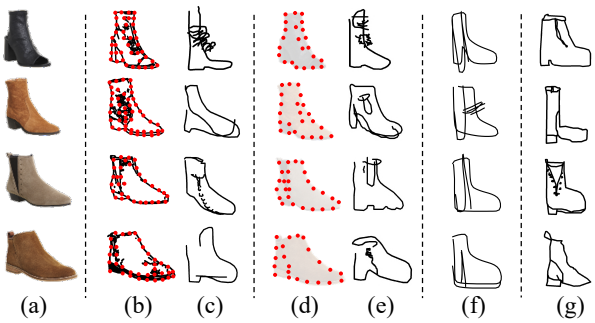

Figure 7. Image-to-sketch synthesis examples. (a) The original photos from Shoes-V2 dataset. (b) The lattice points on edge of the photo shoes. (c) Sketches generated by our model. (d) The points introduced by human referred to the photos. (e) Sketches given by our model using lattice points shown in (d). (f) Sketches generated by LS-SCC [23] for comparison. (g) Human drawn sketches according to the photos.

Table 5. Human study on reality (REAL %) and similarity (SIM %) of the generated sketches.

Method	REAL	SIM
LS-SCC [23]	44.82	49.77
Ours	**55.18**	**50.23**

Results and Analysis Some qualitative results are shown in Figure 7, where we can see that results from both LS-SCC [23] and ours are far from satisfactory when compared to the human-drawn sketches, yet our generated sketches depict more detailed features, such as the "heel", "sole" and the "zipper". The human study results in Table 5 suggest that the produced sketches by our model are closer to human drawing, while equally effective to depict real shoes compared to LS-SCC.

5. Conclusion

We introduced a novel sketch representation, SketchLattice, that not only removes the bottleneck on having vector data, but also preserves the essential structural cues that vector data provides. This result in a sketch representation that is particularly amenable to structural changes that allows better abstraction modeling. We show this new representation helps multiple sketch manipulation tasks, such as sketch healing and image-to-sketch synthesis, where it outperforms state-of-the-art alternatives despite using significantly less parameters.

Acknowledgement

This work was supported by the National Natural Science Foundation of China (NSFC) under 61601042.

References

[1] Somak Aditya, Yezhou Yang, Chitta Baral, Yiannis Aloimonos, and Cornelia Fermüller. Image understanding using vision and reasoning through scene description graph. *CVIU*, 2018. 2

[2] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013. 2

[3] Yang Cao, Hai Wang, Changhu Wang, Zhiwei Li, Liqing Zhang, and Lei Zhang. Mindfinder: interactive sketch-based image search on millions of images. In *ACM MM*, 2010. 1

[4] 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. 1, 2, 4, 5, 6, 7, 8

[5] Zhao-Min Chen, Xiu-Shen Wei, Peng Wang, and Yanwen Guo. Multi-label image recognition with graph convolutional networks. In *CVPR*, 2019. 2

[6] Mathias Eitz, James Hays, and Marc Alexa. How do humans sketch objects? *TOG*, 2012. 1

[7] David Ha and Douglas Eck. A neural representation of sketch drawings. In *ICLR*, 2018. 1, 2, 4, 5, 6, 7, 8

[8] David Ha and Douglas Eck. The quick, draw! dataset. https://github.com/googlecreativelab/quickdraw-dataset, 2018. 1, 2, 4, 5

[9] Zi-Hang Jiang, Qianyi Wu, Keyu Chen, and Juyong Zhang. Disentangled representation learning for 3d face shape. In *CVPR*, 2019. 1

[10] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019. 1

[11] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2013. 4

[12] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *ICPR*, 2016. 1, 2

[13] Maosen Li, Siheng Chen, Xu Chen, Ya Zhang, Yanfeng Wang, and Qi Tian. Actional-structural graph convolutional networks for skeleton-based action recognition. In *CVPR*, 2019. 2

[14] Runtao Liu, Qian Yu21, and Stella X Yu. Unsupervised sketch to photo synthesis. *ECCV*, 2020. 1

[15] Zhaoliang Lun, Matheus Gadelha, Evangelos Kalogerakis, Subhransu Maji, and Rui Wang. 3d shape reconstruction from sketches via multi-view convolutional networks. In *3DV*, 2017. 1

[16] Kaiyue Pang, Ke Li, Yongxin Yang, Honggang Zhang, Timothy M Hospedales, Tao Xiang, and Yi-Zhe Song. Generalising fine-grained sketch-based image retrieval. In *CVPR*, 2019. 1

[17] Kaiyue Pang, Yi-Zhe Song, Tony Xiang, and Timothy M Hospedales. Cross-domain generative learning for fine-grained sketch-based image retrieval. In *BMVC*, 2017. 1

[18] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 5

[19] Anurag Ranjan, Timo Bolkart, Soubhik Sanyal, and Michael J Black. Generating 3d faces using convolutional mesh autoencoders. In *ECCV*, 2018. 1

[20] Umar Riaz Muhammad, Yongxin Yang, Yi-Zhe Song, Tao Xiang, and Timothy M Hospedales. Learning deep sketch abstraction. In *CVPR*, 2018. 1

[21] Patsorn Sangkloy, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. Scribbler: Controlling deep image synthesis with sketch and color. In *CVPR*, 2017. 1

[22] Meng-Li Shih, Shih-Yang Su, Johannes Kopf, and Jia-Bin Huang. 3d photography using context-aware layered depth inpainting. In *CVPR*, 2020. 1

[23] Jifei Song, Kaiyue Pang, Yi-Zhe Song, Tao Xiang, and Timothy M Hospedales. Learning to sketch with shortcut cycle consistency. In *CVPR*, 2018. 1, 5, 8

[24] Guoyao Su, Yonggang Qi, Kaiyue Pang, Jie Yang, and Yi-Zhe Song. Sketchhealer a graph-to-sequence network for recreating partial human sketches. In *BMVC*, 2020. 2, 4, 5, 6, 7, 8

[25] Jiayun Wang, Jierui Lin, Qian Yu, Runtao Liu, Yubei Chen, and Stella X Yu. 3d shape reconstruction from free-hand sketches. *arXiv preprint arXiv:2006.09694*, 2020. 1

[26] Xin Wei, Ruixuan Yu, and Jian Sun. View-gcn: View-based graph convolutional network for 3d shape analysis. In *CVPR*, 2020. 2

[27] Peng Xu, Chaitanya K Joshi, and Xavier Bresson. Multi-graph transformer for free-hand sketch recognition. *arXiv preprint arXiv:1912.11258*, 2019. 1, 5

[28] Jianwei Yang, Jiasen Lu, Stefan Lee, Dhruv Batra, and Devi Parikh. Graph r-cnn for scene graph generation. In *ECCV*, 2018. 2

[29] Lan Yang, Aneeshan Sain, Linpeng Li, Yonggang Qi, Honggang Zhang, and Yi-Zhe Song. S 3 net: Graph representational network for sketch recognition. In *ICME*, 2020. 2

[30] Lumin Yang, Jiajie Zhuang, Hongbo Fu, Kun Zhou, and Youyi Zheng. Sketchgcn: Semantic sketch segmentation with graph convolutional networks. In *CVPR*, 2020. 2

[31] Ting Yao, Yingwei Pan, Yehao Li, and Tao Mei. Exploring visual relationship for image captioning. In *ECCV*, 2018. 2

[32] Qian Yu, Feng Liu, Yi-Zhe Song, Tao Xiang, Timothy M Hospedales, and Chen-Change Loy. Sketch me that shoe. In *CVPR*, 2016. 1, 8

[33] Qian Yu, Yongxin Yang, Yi-Zhe Song, Tao Xiang, and Timothy Hospedales. Sketch-a-net that beats humans. In *BMVC*, 2015. 1, 2

[34] Jesus Zarzar, Silvio Giancola, and Bernard Ghanem. Pointrgcn: Graph convolution networks for 3d vehicles detection refinement. *arXiv preprint arXiv:1911.12236*, 2019. 2

[35] C Lawrence Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. In *ECCV*, 2014. 8