SketchHealer: A Graph-to-Sequence Network for Recreating Partial Human Sketches

Guoyao Su[1]
sgybupt@bupt.edu.cn
Yonggang Qi ✉[1]
qiyg@bupt.edu.cn
Kaiyue Pang[2]
kaiyue.pang@qmul.ac.uk
Jie Yang[1]
janeyang@bupt.edu.cn
Yi-Zhe Song[2]
y.song@surrey.ac.uk

[1] Beijing University of Posts and Telecommunications, Beijing, China
[2] SketchX, CVSSP University of Surrey, UK

Abstract

To perceive and create a whole from parts is a prime trait of the human visual system. In this paper, we teach machines to perform a similar task by *recreating* a vectorised human sketch from its incomplete parts. This is fundamentally different to prior work on image completion (i) sketches exhibit a severe lack of visual cue and are of a sequential nature, and more importantly (ii) we ask for an agent that does not just fill in a missing part, but to *recreate* a *novel* sketch that closely resembles the partial input *from scratch*. Central to our contribution is a graph model that encodes both the visual and structural features over *multiple categories*. A novel sketch graph construction module is proposed that leverages the sequential nature of sketches to associate key parts centred around stroke junctions. The intuition is then that message passing within the said graph will naturally provide the healing power when it comes to missing parts (nodes). Finally, an off-the-shelf LSTM-based decoder is employed to decode sketches in a vectorised fashion. Both qualitative and quantitative results show that the proposed model significantly outperforms state-of-the-art alternatives.

1 Introduction

The human visual system is so remarkable in its ability to reason. One of its important tricks is to perceive a whole by reasoning on parts – the Kanizsa triangle [40] shown in Fig. 1(a) being a famous example. This had partially motivated a recent line of research on image completion [28, 32, 39, 44, 46, 51, 57] which aims at hallucinating missing pixels given contextual regions. Great strides have been made to date, with algorithms able to produce highly plausible filler patches. Such successes are however largely down to the data-driven nature of these algorithms, without much insight given towards reasoning.

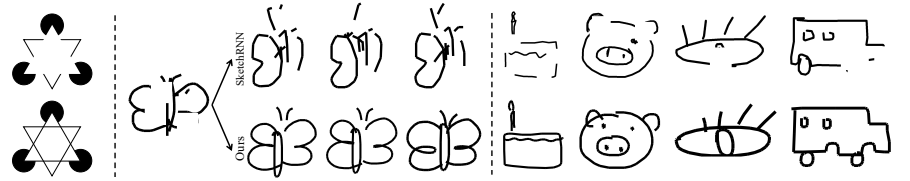

Figure 1: (a) Kanizsa triangle: human can easily perform completion of missing visual parts. (b) SketchRNN fails completely on sketch healing task. (c) Exemplary result on how partial human sketches can be successfully recreated by our proposed SketchHealer.

Others have fixated on human sketches as a medium to gain insight into the human visual system – to see is to sketch. That is, the sketching process to a large extent reflects the visual perception of an object. This has triggered a large body of research on human sketch understanding, some more application oriented [2, 3, 22, 34, 36, 53, 54], others starting to tackle insightful problems such as sketch synthesis [37], sketch abstraction [31] and sketch completion [24]. Sketch is commonly perceived to be a more challenging visual modality compared with photo, because (i) they lack visual features, (ii) they are abstract and iconic, and (iii) they are sequential in nature.

In this paper, we are also interested in studying sketches. In particular, we would like to use sketches to understand the visual reasoning problem of *devising the whole from parts* (albeit only to a superficial level). We differ significantly to the conventional problem of image completion. First, we do not treat sketches as pixelated photos, but as a sequence of strokes (represented in vector format) that reflects the actual drawing process. Furthermore, we require ourselves to generate a *novel* and *complete* sketch stroke-by-stroke that best resembles the partial input, other than just filling in the missing parts. Together, these constraints deviate us from image completion, and move towards a new problem which we call *sketch healing*.

On the outset, sketch healing is akin to the well studied problem of vector sketch synthesis. The pioneering work of SketchRNN [17], for example, already has the ability to generate realistic human-like sketch drawings at stroke-level, either from a random vector or conditioned on a partial sketch encoding. In hindsight though, we are completely different. We are not sketching towards a *recognisable* concept, but a complete sketch that closely *resembles* the partial input. For example, conditioned on the encoding of a partial butterfly sketch, SketchRNN is interested in sketching a *plausible* butterfly; we on the other hand are focusing on reproducing a complete version of the partial sketch, regardless of knowing whether it is a butterfly (Fig. 1(b)). We further insist on solving for random droppings of sketch parts as well, where more than one "hole" appear anywhere on a sketch. This setting is incompatible with the "completion mode" of SketchRNN, which dictates a strict sequential ordering, *i.e.*, the synthesised sketch must be *on top of* of existing input strokes.

Solving this sketch healing task is non-trivial. It requires a sketch-specific representation that not only accommodates the unique traits of sketches (abstract and sequential), but also robust enough towards missing parts. This is made even more challenging since we are after a more generic healer that works over *multiple categories*, other than training a single model per category. In this paper, we resort to the power of Graph Convolutional Networks (GCN) [10, 21] for that. To encapsulate the temporal trait of sketches, we introduce a novel sketch graph construction module that organises key sketch parts in accordance with the

order of drawing. More specifically, we select representative stroke points as graph nodes, and form the edge links via an adjacency matrix based on temporal proximity. To further tackle the abstract nature of sketches, we sample candidate parts at key stroke junction points to ensure each part captures the most visual information. At training, we randomly drop a fixed percentage of nodes from this graph to produce a set of incomplete graphs. It follows that these partial graphs which correspond to partial sketches are fed into a GCN network to learn the final graph embedding. Finally we employ the same LSTM decoder as that used by SketchRNN to output a vectorised sketch stroke-by-stroke. We show this seemingly simple formulation can already solve the sketch healing problem well, and is able to synthesise complete sketches better than state-of-the-arts (Fig. 1(c)). The effectiveness of our approach is however intuitive: (i) the learned graph offers part-oriented and structure-aware sketch representation that is robust to node removal, and (ii) at testing, the inherent node message passing mechanism inside GCNs works naturally to fill in the missing gaps.

Our contribution can be summarised as follows: (i) we propose the problem of sketch healing, as an interesting yet changeling alternative to conventional sketch synthesis, (ii) a novel graph-to-sequence network is proposed to solve this problem, where an unique sketch graph construction algorithm is introduced to tackle the unique traits of sketches. We evaluate our method on 17 categories selected from QuickDraw [12] dataset and validate its superiority over two state-of-the-art vector sketch synthesis baselines. We also show interesting vector arithmetic properties, on incomplete sketches and across object categories.

2 Related Work

Vector Sketch Generation Much progress [4, 5, 16, 17, 18, 19, 26, 29, 55, 58] has been made on image generation tasks both in the supervised [15] and unsupervised settings [20, 50, 59]. Given a cat image, we can now translate to other category of animals [60], render it in Monet style [23] or even make it 3D animated [35]. This is in stark contrast with the very few existing works that focus on vector image generation, where its temporal and spatial nature bring more challenges. The seminal work of [11] propose a sequence-to-sequence model and for the first time achieved realistic vector handwritten digits generation in a wide variety of styles. The once continuous data is discretised into a set of points and step-by-step point prediction is enabled by mixture of density networks. With the availability of large-scale crowd-sourced sketch datasets, this model is then adapted in [6, 12] and achieved both unconditional and conditional vector sketch-to-sketch synthesis. Vector sketch generation is also extended beyond a single domain. [37] propose the first deep stroke-level photo-to-sketch synthesis model. To cope with the intrinsic noisy supervision of photo-sketch pairs, they address the limitations of cross-domain image translation models based on multi-task supervised and unsupervised hybrid learning. In this paper, we study a different problem – sketch healing, that takes a partial sketch as input and output *novel* sketch that closely resembles the input, while others focus on sketch synthesis (*i.e.*, to sketch a recognisable rendition) [6, 12], and photo-sketch synthesis [37].

Graphical Sketch Representation Graph convolutional network (GCN) [10, 21] was proposed to extend deep neural networks to data with graph structures. By applying GCN-based models, state-of-the-art performance has been achieved over a range of vision tasks, such as image classification [7], image captioning [49], scene understanding [47] and 3D mesh deformation [30, 41]. The sequential nature of sketch and the visual sparsity it presents make it an ideal data domain for graphical representations. But only until very recently, GCN-based

sketch visual learning were attempted in [45] and [48] for the problem of sketch recognition and segmentation respectively. They both construct their graph nodes based on the absolute coordinates of the sampled sketch points and transform them via multi-layer perceptrons and appropriate pooling methods. In contrast, our proposed SketchHealer uniquely embed the temporal drawing order to build adjacency matrices.

Image Inpainting/Completion Image inpainting/completion is to synthesise visual contents of a plausible hypothesis in a missing or damaged region. There are two broad lines of work aiming to tackle this task: exemplary-based methods [1, 8, 9, 43] search and paste visual patches from other known regions in the gallery. These algorithms work well for stationary images (*e.g.*, textures) but can lead to complete failure on non-stationary natural images. Deep generative CNN-based methods [28, 32, 39, 44, 51] directly generate pixels inside the missing patch based on the semantics learned from large-scale dataset in an end-to-end fashion. This usually involves an encoder-decoder paradigm with various key modifications devised - including partial [25] and gated [52] convolutions, use of contextual attention modules [38] and adversarial discriminators [14]. User guidance is also explored to improve inpainting results including edge lines [33], semantic label maps [42] and colour palettes [56]. To our best knowledge, the only inpainting work on sketches is [24], which devises a cascade network to refine the completions in an iterative manner. SketchHealer is fundamentally different in that we not only tackle vector sketches, but also in the very nature of the problem itself – rather than just filling the holes, it heals a corrupted incomplete sketch by generating a novel full counterpart.

3 Methodology

Our goal is to recreate a vector sketch $S = (s_1, s_2, \ldots, s_n)$ from its partial version $\hat{S}$. A sketch is a set of points, where each segment s_i is constructed between two consecutive points as a 5 dimensional vector $(\Delta x, \Delta y, ps_1, ps_2, ps_3)$. $(\Delta x, \Delta y)$ is the offset distance in the x and y directions of the pen from the previous point. (ps_1, ps_2, ps_3) is a one-hot vector describing current pen states, where $(1,0,0), (0,1,0), (0,0,1)$ denote touching, lifting and the end of sketch drawing respectively. $\hat{S}$ is obtained from S by randomly removing a proportion n points. Our key component is a GCN-based encoder, which maps a partial sketch $\hat{S}$ of its graphical form $G = (V, E)$ to a latent vector $z \in \mathbb{R}^{d_{model}}$. z is then leveraged as input to sequentially sample the output sketch S in a LSTM decoder. A schematic illustration is shown in Fig. 2.

3.1 Sketch Graph Construction

Graph Nodes V We consider two types of points as representative graph nodes: (i) the starting point of each stroke, which determines the main structural layout; (ii) internal points sparsely sampled within each single stroke in order to capture a rough path trace. We sample one graph node in every four points in a stroke throughout this paper. Consequently, a set of graph nodes $V = (v_1, v_2, \ldots, v_m)$ is a subset of S. While being more compact, V still preserve the key geometric landmarks of input sketch.

Graph Edges E Temporal-based nearest neighbour strategy is used to construct the edge links between nodes. That is, for each node v_i, we will connect it with the graph nodes nearby in accordance to their drawing orders in the original sequence of stroke points. We link v_i with its four nearest graph nodes, two prior to its rendering as parent nodes, and two

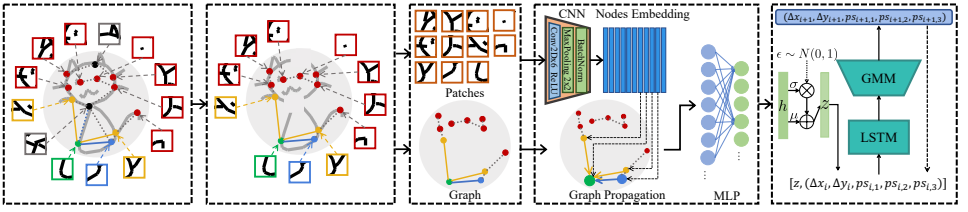

Figure 2: A schematic illustration of SketchHealer. (a) A full sketch s in its graph form. For each graph node v_i, a visual patch is cropped out as p_i. (b) A corrupted partial sketch $\hat{S}$ in its graph form by masking out a fractional of nodes from S. The associated edge links are removed as well. (c) Model input: graph G and the visual patches P for $\hat{S}$. (d) GCN-based encoder. A graph node (green) will attend to its nearest neighbourhood (blue) and the second nearest (yellow) through graph propagation. (e) LSTM decoder. More details in text.

after its presence as child nodes. An adjacency matrix $\mathbf{A} \in \mathbb{R}^{m \times m}$ can then be formed, where each entry a_{ij} represents the link strength between nodes v_i and v_j. We empirically found $a_{ij} = 0.3$ for edge link between two nearest nodes and $a_{ij} = 0.2$ for its linkage to a second nearest node to work well. a_{ij} is zero-valued to indicate no inter-node connections and for self-connection a_{ii}, we simply take its value to 0.5 for regularisation purpose.

Visual Patch P To assign each graph node v_i in $V = (v_1, v_2, \ldots, v_m)$ with its associated visual cue, a local image patch centred around each node is acquired. Specifically, we first render out a raster sketch image of size 640×640 from its vector format and crop a square visual patch p_i of size 128×128 based on the normalised coordinate of v_i. The relatively large patch size is to make sure enough informative visual cues are still captured given the sparse nature of human sketches. A set of node-driven patches $P = (p_1, p_2, \ldots, p_m)$ is thus obtained.

From Full S **to Partial** $\hat{S}$ To form a graph for partial sketch as final input, we randomly remove a fraction of graph nodes by a probability of p_{mask} and cut the connections in the resulting edge links. The corresponding image patch p_i in P will also become void.

3.2 Model Architecture

GCN-based Encoder Our proposed SketchHealer encoder consists of six convolutional layers with kernel size 2×2 followed by max pooling and batch normalisation. By feeding each p_i into the encoder, we produce a visual feature vector $f_{v_i} \in \mathbb{R}^d$ for each node v_i. Then feature propagation is executed to form an updated node feature $u_{v_i} \in \mathbb{R}_d$, where a node v_i attends to all its linked neighbours defined in the adjacency matrix A. Such a spatial-dependent approach is natural to provide a healing effect for the absence of certain parts and enables more robust representation. Formally, we formulate this as follows:

$$u_{v_i} = \sum_{j=1}^{m} a_{ij} f_{v_j} \tag{1}$$

We then integrate all node features into a single vector $h \in \mathbb{R}^{d_{model}}$ for representing $\hat{S}$:

$$h = (w_1, w_2, \ldots, w_m) \times [g(u_{v_1}), g(u_{v_2}), \ldots, g(u_{v_m})]^T \tag{2}$$

where m is set as the maximum number of nodes among all training sketches ($m = 25$ in our case), $g : \mathbb{R}^d \rightarrow \mathbb{R}^{d_{model}}$ is a multilayer perceptron (MLP) unit, $(w_1, w_2, \ldots, w_m)$ is a learnable weight vector to linearly combine the MLP-produced node vectors $[g(u_{v_1}), g(u_{v_2}), \ldots, g(u_{v_m})]$. To introduce generative components, h is further projected into two vectors, $\mu \in \mathbb{R}^{d_{model}}$ and $\sigma \in \mathbb{R}^{d_{model}}$, along with a vector of IID Gaussian variables $\mathcal{N}(0,1)$ of size d_{model}, to construct the final latent vector z:

$$z = \mu + \sigma \odot \mathcal{N}(0,1),\ \mu = W_\mu h,\ \sigma = exp(\frac{W_\sigma h}{2}) \tag{3}$$

LSTM Decoder Taking latent vector z as condition, a LSTM decoder is used to sequentially sample output sketch strokes. Concretely, the previous point s_{i-1} together with the latent vector z are formed as input at each time step, *i.e.* $x_i = [s_{i-1}; z]$. Then the next hidden state is obtained by $[h_i; c_i] = LSTM_{forward}(x_i, [h_{i-1}; c_{i-1}])$. Hence, the output is given by $y_i = w_y h_i + b_y$, where vector $y_i \in \mathbb{R}^{6M+3}$ is then unpacked into a set of parameters:

$$y_i = [(\Pi, \mu_x, \mu_y, \delta_x, \delta_y, \rho_{xy})_1 \ldots (\Pi, \mu_x, \mu_y, \delta_x, \delta_y, \rho_{xy})_M (q_1, q_2, q_3)] \tag{4}$$

where the first M sets of parameters are used to form a Gaussian mixture model (GMM) with M normal distributions to predict $(\Delta x, \Delta y)$ by:

$$p(\Delta x, \Delta y) = \sum_{j=1}^{M} \Pi_j \mathcal{N}(\Delta x, \Delta y | \mu_{x,j}, \mu_{y,j}, \delta_{x,j}, \delta_{y,j}, \rho_{xy,j}) \qquad s.t. \sum_{j=1}^{M} \Pi_j = 1 \tag{5}$$

The last three parameters (q_1, q_2, q_3) are used to estimate pen state (ps_1, ps_2, ps_3) via a categorical distribution, *i.e.* $ps_k = \frac{\exp(q_k)}{\sum_{j=1}^{3} \exp(q_k)}, k = 1, 2, 3$. Refer to [12] for more details.

3.3 Model Learning and Deployment

Learning Objectives To train a *multi-class* generator, we follow [6] to remove the KL-divergence term, and only seek to maximise the posterior probability $q_\phi(z|\hat{S})$ of the generated data points to the target data distribution $p_\theta(S|z)$. The objective function is defined as:

$$\min E_{q_\phi(z|\hat{S})}[\log p_\theta(S|z)] \tag{6}$$

SketchHealer Deployment Once trained, it is straightforward to apply the SketchHealer. Given a latent vector z encoded from a corrupted sketch input, we feed it together with a manually-defined starting point $(\Delta x = 0, \Delta y = 0, ps_1 = 1, ps_2 = 0, ps_3 = 0)$ into the LSTM decoder. The generated data point will be fed again with z to produce the next data point in a recurrent manner, until the stop signal is reached, *i.e.* $(ps_1 = 0, ps_2 = 0, ps_3 = 1)$.

4 Experiment

4.1 Experimental Setting

Dataset We evaluate our proposed model on *QuickDraw* [13], which is the largest human sketch drawing dataset to date. It provides over 50M vector sketches across 345 object categories, where we select a subset for our experiments. In particular, the 17 categories we choose generally respect the following rules: (i) both complex and simple drawings

are included, *e.g.* `angel` and `belt`; (ii) instances inside categories exhibit similar global appearances but only differ in very local subtle details. *e.g.* `cat` and `pig`; (iii) common life object category contains diverse sub-category variations, *e.g.* `bus`, `umbrella` and `clock`. A full list can be found in Fig. 6(a). For each class, 70,000 sketches are used for training, and 2,500 for testing for each class.

Baselines To our best knowledge, there are two conditional vector sketch generation frameworks publicly available, **SketchRNN** [12] and **SketchPix2seq** [6], both of which we compare with. SketchRNN is a sequence-to-sequence model that takes in the offset distance between consecutive points as temporal input. For fair comparison, we retrain SketchRNN without KL-divergence term, which is shown to be beneficial for multi-class scenario. SketchPix2seq differs in its convolutional encoder, which scraps the temporal sequential point representation of sketches and accepts raster sketch image input instead, with the hope for better visual learning via CNNs. We note these baselines were not specifically designed for the sketch healing task, yet they still represent the closest alternatives and are procedural-wise compatible once re-purposed.

Evaluation protocol Apart from qualitative comparisons, we design two metrics to allow quantitative evaluations. Our measures specifically answer two questions: (i) how recognisable are the vector sketches generated by different models? (ii) how recognisable are the latent vectors encoded by different type of encoders? A good score for the former requires the sketches healed from their partial parts to be realistic and diverse, while the latter calls for a strong healing ability from encoders to produce robust feature representations. More specifically, we formulate two metrics: (i) *sketch recognition accuracy*: we train a multi-category classifier by AlexNet using the all the training split of 345 categories in *QuickDraw* dataset. The classifier is then used to assign a class label to measure how recognisable a generated sketch is. 2500 testing sketches from each of 17 categories are used for this purpose. (ii) *sketch-to-sketch retrieval accuracy*: The resulting feature from an encoder is expected to retrieve sketches of the same label in the gallery. We form our query with 500 testing sketches from each of the 17 categories and the rest as gallery.

Implementation details Our model is implemented on PyTorch [27] with a single Nvidia Tesla T4 GPU. The Adam optimiser is used with the parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\varepsilon = 10^{-8}$. The learning rate is set to 10^{-3} with a decay rate of 0.999 every iteration. The proportion of stroke points to mask out during model training is set as a fixed value of $p_{mask} = 10\%$, and tested for different level of sketch corruptions. Code is available at https://github.com/sgybupt/SketchHealer.

4.2 Results

Qualitative results We illustrate some examples produced by our SketchHealer under different values of p_{mask} in Fig. 3. The following observations can be made: (i) SketchHealer is not only able to render a novel sketch just like humans do, but can also faithfully recreate the essential subtle visual elements even when the majority part of specific visual cues are missing in the partial input. For example, the halo over the head of angle keeps presented up to $p_{mask} = 70\%$, despite the input sometimes only shows very weak evidence of halo visual signals. (ii) Given one human sketch and different random removals of visual elements on different levels, SketchHealer delivers consistent generation results - albeit subtle details are uniquely rendered, global appearances and structures are unanimously kept. (iii) The sensitiveness of our proposed model to different corruption levels of inputs vary across object categories. But overall, the model performs reasonably well when $p_{mask} \leq 50\%$. We further

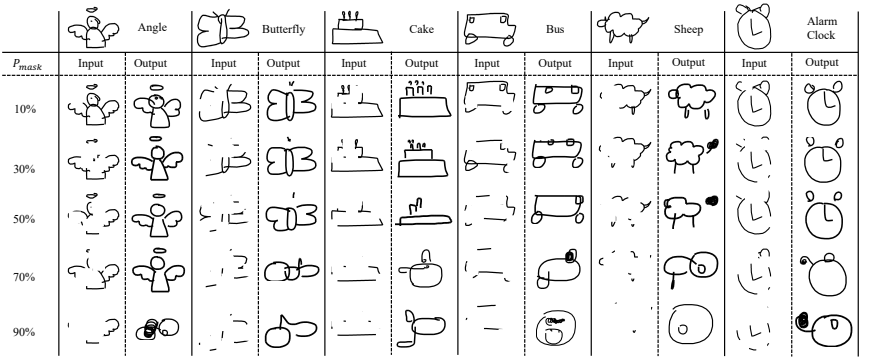

Figure 3: Exemplary results of SketchHealer under different corruption values of p_{mask}.

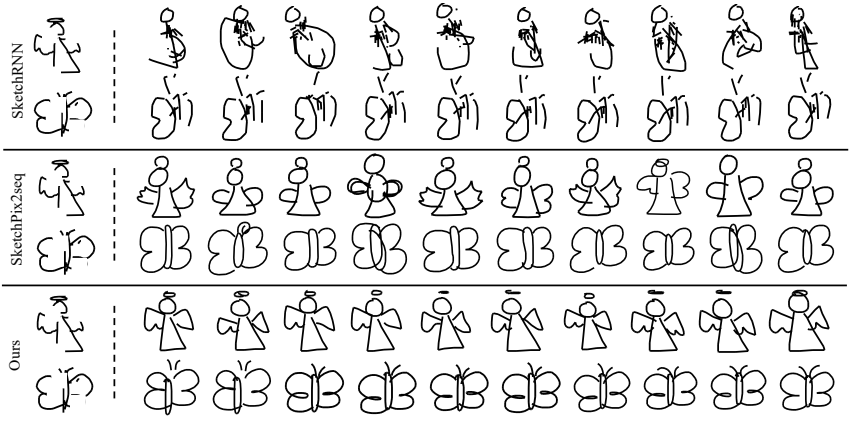

Figure 4: Qualitative comparisons between the proposed SketchHealer and two other state-of-the-art baseline methods. $p_{mask} = 10\%$ throughout.

qualitatively compare with two baselines in Fig. 4. Even under a corruption level of 10%, SketchRNN completely fails to recreate a desired vector sketch. And while SketchPix2seq performs considerably better, its gap with SketchHealer is significant: see how the wings of the angle and butterfly get healed in multiple generative renderings using our proposed method.

Quantitative results We compare the performance of different models under the two metrics (Sec. 4.1) in Table. 1 and Table. 2: (i) Under the recognition metric, SketchHealer beats two baselines when $p_{mask} \neq 0$. Interestingly, when the un-corrupted full sketch is fed as input, SketchRNN achieves the best score, but collapses dramatically even when only 10% of stroke points are masked out and a complete failure when the proportion rises to 30%. (ii) Under the retrieval metric, our model still outperforms all baselines. However, this time, the significant improvement over SketchRNN even manifests in the full sketch input. This suggests that our GCN-based encoder is not limited to the scenario for healing partial vector data input, but applicable to discriminative modelling of sequential data at large. (iii) Compared with baseline methods, our GCN-based encoder also shows surprisingly stable behaviour when the corrupted level of sketch input increases. In particular, its discriminative power seems not to be affected at all. We also visualise some sketch-to-sketch retrieval

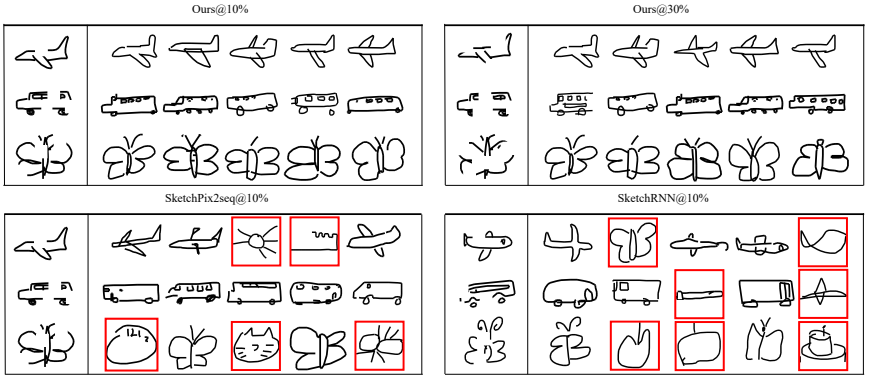

Figure 5: Qualitative comparisons for sketch-to-sketch retrieval results. Top 5 is returned. Red bounding box indicates false positive from wrong category.

Table 1: Recognition result(%) obtained from generated sketch.

Competitor	Top1	Top3	Top5	Top10
Human	71.99	87.42	91.48	94.86
SketchRNN@0%	57.98	75.59	81.92	88.02
SketchPix2seq@0%	22.14	30.11	34.46	40.56
Ours@0%	50.64	65.63	70.61	76.71
SketchRNN@10%	24.41	39.12	46.23	56.28
SketchPix2seq@10%	21.88	28.50	31.92	36.91
Ours@10%	49.77	64.76	69.92	76.08
SketchRNN@30%	3.14	7.14	10.25	15.91
SketchPix2seq@30%	9.51	13.40	16.06	20.26
Ours@30%	42.25	56.81	62.14	68.75

Table 2: Retrieval result(%) obtained on encoded z.

Competitor	Top1	Top3	Top5	Top10
SketchRNN@0%	64.59	79.61	84.92	90.82
SketchPix2seq@0%	61.29	79.48	85.14	91.54
Ours@0%	85.68	93.45	94.98	96.86
SketchRNN@10%	50.65	69.76	77.60	86.21
SketchPix2seq@10%	45.20	68.34	77.54	87.09
Ours@10%	85.74	93.08	95.01	96.89
SketchRNN@30%	43.48	63.67	72.03	82.39
SketchPix2seq@30%	27.66	51.82	63.52	78.99
Ours@30%	85.47	93.01	94.89	96.79

results in Fig. 5. Even under mild condition where $p_{mask} = 10\%$, SketchRNN and SketchPix2seq have clearly many more false positives. In contrast, our GCN-based encoder is not only category-discriminative in the more challenging setting ($p_{mask} = 30\%$), but also learns to respect finer-grained details (*e.g.* the dense side-by-side windows of the bus).

Human Study We further conduct a human study to verify the subjective quality of synthesised sketches. We first recruit a total of 10 human judges. 50 sketch samples are then randomly selected across all 17 categories, where each sample has two corrupted versions associated, at mask ratio $p_{mask} = 10\%$ and $p_{mask} = 30\%$, respectively. For each corrupted sketch, we generate a triplet of healed sketches using different methods (Ours, SketchRNN [12], SketchPix2seq [6]), where the ordering is randomised. We show each participant, the corrupted input sketch, and the triplet of healed ones at once. Each participant is then asked to mark one from the triplet that they think the best resembles the corrupted sketch. Results in Table. 3 show where the selected best healed sketches come from. It clearly reveals that sketches recreated by ours are more likely to be chosen as the best, and the superiority is further enhanced when increasing the corruption level of input sketch.

Table 3: Human study results(%).

	SketchRNN	SketchPix2seq	**Ours**
$p_{mask} = 10\%$	15.05	14.97	69.98
$p_{mask} = 30\%$	11.16	12.72	76.12

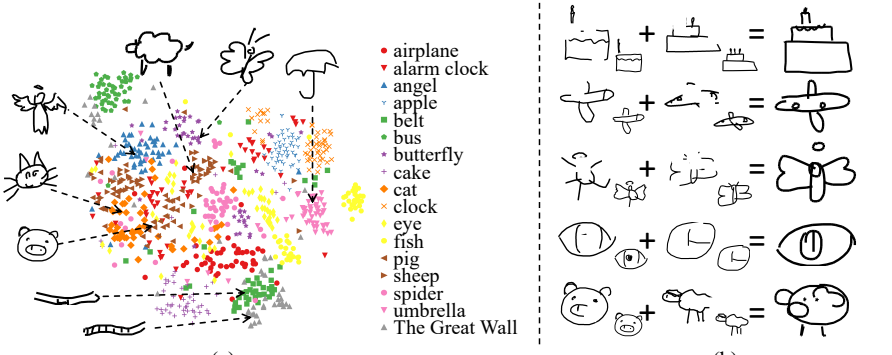

Figure 6: (a) Latent space visualisation by t-SNE. (b) Sketch drawing analogy.

4.3 Further analysis

Visualisation of z To further visualise the discriminative power of GCN-based encoder, we randomly select 100 sketches for each class in the test set, and visualise their latent vectors z using t-SNE in Fig. 6(a). We can observe that (i) Intra-category instances tend to cluster together, indicating category-level discriminative understanding has been achieved yet in an unsupervised way. (ii) Semantically similar categories sit closer and partly overlap, *e.g.* `cat` and `pig`, while distinct categories, *e.g.* `bus` and `umbrella`, are pushed further away.

Vector Arithmetic on z The result in Fig. 3 suggests that latent vector z encodes controllable conceptual feature for corrupted partial sketch. In this section, we want to find out whether we can use the features in z to augment another z without such feature. We conduct vector addition on z from two visually distinct partial sketches and feed the sum to the decoder. Fig. 6(b) shows some successful examples. See how a corrupted pig sketch plus a partial sheep transforms into a novel rendering with visual traits from both.

5 Conclusion

We introduced the problem of sketch healing that asks a new question: given a partial sketch, can we synthesise a complete and novel sketch that best resembles the partial input. We achieved this by introducing a graph model that importantly gives us both feature robustness and flexibility in handling missing information. On graph construction, we uniquely encapsulated two unique traits of sketches (temporal and abstract) to make the graph model more sketch-specific. By experiments, we show SketchHealer is able to consistently produce complete sketches that closely resemble the partial input, whereas alternatives re-purposed for the problem work less well.

6 Acknowledgement

This work was supported by the National Natural Science Foundation of China (NSFC) under 61601042 and 61671078, and conducted while Yonggang Qi was visiting SketchX Lab under China Scholarship Council (CSC) funding 201906475001.

References

[1] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. In *ACM Transactions on Graphics (ToG)*, 2009.

[2] Itamar Berger, Ariel Shamir, Moshe Mahler, Elizabeth Carter, and Jessica Hodgins. Style and abstraction in portrait sketching. *ACM Transactions on Graphics (TOG)*, 2013.

[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 *ACMMM*, 2010.

[4] Caroline Chan, Shiry Ginosar, Tinghui Zhou, and Alexei A Efros. Everybody dance now. In *ICCV*, 2019.

[5] Qifeng Chen and Vladlen Koltun. Photographic image synthesis with cascaded refinement networks. In *ICCV*, 2017.

[6] 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.

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

[8] Alexei A Efros and William T Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 2001.

[9] Alexei A Efros and Thomas K Leung. Texture synthesis by non-parametric sampling. In *ICCV*, 1999.

[10] Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *IJCNN*. IEEE, 2005.

[11] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

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

[13] David Ha and Douglas Eck. The quick, draw! dataset. https://github.com/googlecreativelab/quickdraw-dataset, 2018.

[14] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Globally and locally consistent image completion. *ACM Transactions on Graphics (ToG)*, 2017.

[15] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.

[16] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016.

[17] Justin Johnson, Agrim Gupta, and Li Fei-Fei. Image generation from scene graphs. In *CVPR*, pages 1219–1228, 2018.

[18] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

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

[20] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks. In *ICML*, 2017.

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

[22] Ke Li, Kaiyue Pang, Jifei Song, Yi-Zhe Song, Tao Xiang, Timothy M Hospedales, and Honggang Zhang. Universal sketch perceptual grouping. In *ECCV*, 2018.

[23] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Universal style transfer via feature transforms. In *NIPS*, 2017.

[24] Fang Liu, Xiaoming Deng, Yu-Kun Lai, Yong-Jin Liu, Cuixia Ma, and Hongan Wang. Sketchgan: Joint sketch completion and recognition with generative adversarial network. In *CVPR*, 2019.

[25] Guilin Liu, Fitsum A Reda, Kevin J Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. Image inpainting for irregular holes using partial convolutions. In *ECCV*, 2018.

[26] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

[27] 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.

[28] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016.

[29] Albert Pumarola, Antonio Agudo, Aleix M Martinez, Alberto Sanfeliu, and Francesc Moreno-Noguer. Ganimation: Anatomically-aware facial animation from a single image. In *ECCV*, 2018.

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

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

[32] Min-cheol Sagong, Yong-goo Shin, Seung-wook Kim, Seung Park, and Sung-jea Ko. Pepsi: Fast image inpainting with parallel decoding network. In *CVPR*, 2019.

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

[34] Yuming Shen, Li Liu, Fumin Shen, and Ling Shao. Zero-shot sketch-image hashing. In *CVPR*, 2018.

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

[36] Edgar Simo-Serra, Satoshi Iizuka, and Hiroshi Ishikawa. Mastering sketching: adversarial augmentation for structured prediction. *ACM Transactions on Graphics (TOG)*, 2018.

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

[38] Yuhang Song, Chao Yang, Zhe Lin, Xiaofeng Liu, Qin Huang, Hao Li, and C-C Jay Kuo. Contextual-based image inpainting: Infer, match, and translate. In *ECCV*, 2018.

[39] Yuhang Song, Chao Yang, Yeji Shen, Peng Wang, Qin Huang, and C-C Jay Kuo. Spg-net: Segmentation prediction and guidance network for image inpainting. *arXiv preprint arXiv:1805.03356*, 2018.

[40] Catherine Tallon, Olivier Bertrand, Patrick Bouchet, and Jacques Pernier. Gamma-range activity evoked by coherent visual stimuli in humans. *European Journal of Neuroscience*, 7(6):1285–1291, 1995.

[41] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *ECCV*, 2018.

[42] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *CVPR*, 2018.

[43] Marta Wilczkowiak, Gabriel J Brostow, Ben Tordoff, and Roberto Cipolla. Hole filling through photomontage. In *BMVC*, 2005.

[44] Wei Xiong, Jiahui Yu, Zhe Lin, Jimei Yang, Xin Lu, Connelly Barnes, and Jiebo Luo. Foreground-aware image inpainting. In *CVPR*, 2019.

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

[46] Chao Yang, Xin Lu, Zhe Lin, Eli Shechtman, Oliver Wang, and Hao Li. High-resolution image inpainting using multi-scale neural patch synthesis. In *CVPR*, 2017.

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

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

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

[50] Zili Yi, Hao Zhang, Ping Tan, and Minglun Gong. Dualgan: Unsupervised dual learning for image-to-image translation. In *ICCV*, 2017.

[51] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Generative image inpainting with contextual attention. In *CVPR*, 2018.

[52] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Free-form image inpainting with gated convolution. In *ICCV*, 2019.

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

[54] 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. *IJCV*, 2017.

[55] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *ICCV*, 2017.

[56] Richard Zhang, Jun-Yan Zhu, Phillip Isola, Xinyang Geng, Angela S Lin, Tianhe Yu, and Alexei A Efros. Real-time user-guided image colorization with learned deep priors. *arXiv preprint arXiv:1705.02999*, 2017.

[57] Chuanxia Zheng, Tat-Jen Cham, and Jianfei Cai. Pluralistic image completion. In *CVPR*, 2019.

[58] Hang Zhou, Yu Liu, Ziwei Liu, Ping Luo, and Xiaogang Wang. Talking face generation by adversarially disentangled audio-visual representation. In *AAAI*, 2019.

[59] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017.

[60] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. In *NIPS*, 2017.