Pattern Recognition Letters 145 (2021) 16–22

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

# DCR: Disentangled component representation for sketch generation☆

Zhong Cao, Sen Cui, Changshui Zhang*

*Institute for Artificial Intelligence, Tsinghua University (THUAI), Beijing National Research Center for Information Science and Technology (BNRist), Department of Automation, Tsinghua University Beijing, P.R. China*

ARTICLE INFO

*Article history:*
Received 13 November 2019
Revised 27 September 2020
Accepted 17 January 2021
Available online 21 January 2021

*MSC:*
41A05
41A10
65D05
65D17

*Keywords:*
Disentangled representation
Sketch generation
Attention mechanism

ABSTRACT

We present a simple end-to-end model based on deep learning to automatically decompose sketched objects into components by disentangling the visual representation. The performance of visual representation learning based models degrades as categories increase. Rather than building a mapping from a static image to the whole sketch sequences, we propose an interpretable disentangled representation of sketch to understand component concepts and the relationship among such concepts. Our model takes the binary image of a sketched object and produces a component stroke sequence set corresponding to key components in the sketch. Experiments show that our method significantly outperforms all baselines quantitatively at the degree of disentanglement, and our method is more stable while training on tens of categories.



## 1. Introduction

Free-hand sketching is considered as an efficient means of expressing thoughts since we can share visual concepts with others in a simple and effective way. Compared to photo or edge generation, which contains redundant information about background and noise, sketch generation mainly focuses on artistic depictions from humans. The release of the QuickDraw dataset [6] fostered the application of the generative models on vector sketch. Sketch-RNN [6] is a sequence-to-sequence variational autoencoder model, which encodes sequences of strokes into a latent vector and decodes the latent vector into sequences of strokes. Sketch-pix2seq [4] model was proposed to generate sketches with CNN encoder to capture the local structure. However, existing studies on sketch generation are based on the global representation of sketch, lacking further understanding of its component, which is hard to get representations of all components of the sketch and more likely to generate incorrect sketches if training on tens of categories.

Our motivation is to learn basic component representations instead of global representations for better sketch generation. The Gestalt laws of psychologists [18] state that humans disentangle objects into visual patterns for better understanding. These visual patterns are basic components shared among all different kinds of sketches [12]. In this work, we propose a novel method for generating sketch point sequences from an input sketch pixel image as shown in Fig. 1. We argue that disentangled component representation (DCR) is helpful for sketch generation when specifying the category label or the component label. Moreover, these DCRs make our generation process model more interpretable, flexible, and diversifiable. Our method outperforms existing models with regard to multi-class generation quality. Beyond that, we can even generate new kinds of sketches by setting different components manually.

First, we learn about global image representations via a CNN autoencoder, and component sequence representations by LSTM autoencoder. Then a feature disentangling module (FDM) is proposed to disentangle the global image representations into DCRs. During testing phase, we can generate sketch sequences component-by-component from the input pixel image. Experiments on sketch perceptual grouping (SPG [12]) dataset and our online handwritten Chinese character (OLCC) dataset show that our method outperforms state-of-the-art method on the multi-class sketch generations.

To sum up, our main contributions in this work are: (1) We propose a novel sketch generation method, which excels in learning shared component representations between different categories. (2) We release a large online handwritten Chinese character dataset with stroke-level and image-level labels. (3) Experiments show that our method outperforms state-of-the-art methods be-

☆ Handled by Associate Editor: Haibin Yan

* Corresponding author.
*E-mail address:* zcs@mail.tsinghua.edu.cn (C. Zhang).

https://doi.org/10.1016/j.patrec.2021.01.016
0167-8655/

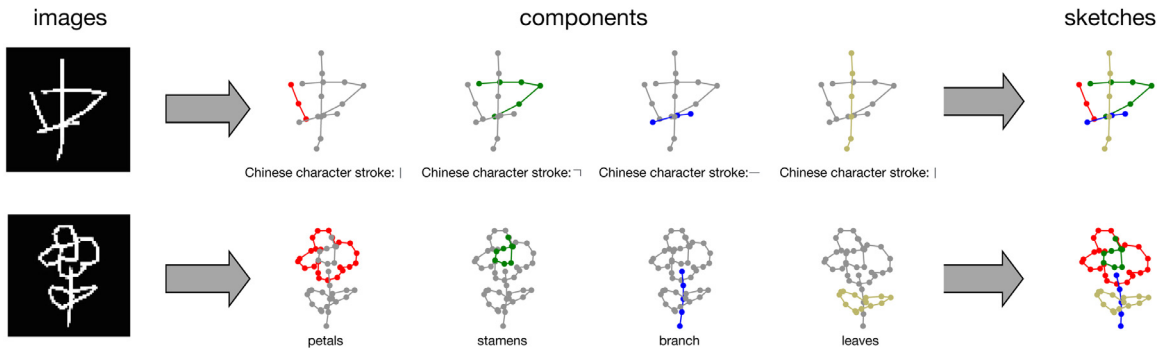


**Fig. 1.** Overview of our work. Our model takes a pixel image as input, disentangles the image into each component (drawn in red, green, blue, and yellow, respectively) and generates sketches as output. There is a Chinese character example from OLCC dataset on the top and a flower example from SPG dataset on the bottom. The Chinese character has four character strokes as its components. The flower has four components, such as the petals, the stamens, the branch, and the leaves respectively. 500 sketch categories of OLCC share 24 kinds of components and 25 sketch categories of SPG share 55 kinds of components. The numbers or types of components vary in different sketch categories. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

cause of DCR. (4) We also analyze the sketch generation of new categories with learned components.

## 2. Related work

### 2.1. Sketch generation

Sketch generation has recently been studied in handwritten characters on the Omniglot dataset [11]. Lake et al. achieves human-level performance but requires parses from a hand-crafted algorithm to initialize training and generating handwritten characters [11]. More recently, a series of Sketch-RNN based models [3,4,6,9,17,22] have been proposed to produce new sketches based on existing observations. Sketch-RNN [6], a sequence-to-sequence variational autoencoder, has been widely used in generating sketch drawings. The model explores encoding the sequences of strokes into a latent space of embedding vectors using a bidirectional RNN [16] based on LSTM [7] and reconstructs the sequences of strokes via an autoregressive RNN [2]. There are limitations on the insufficient quality of the generated sketches and incapability to generate multiple categories of sketches in Sketch-RNN. There are some works [3,4,17] developing sketch generation method with CNN encoder to capture the structure information from the images. These works (i.e., CNN autoencoder, LSTM autoencoder) are the backbone of variational inference for sketch generation.

### 2.2. Photo-to-sketch translation

Most of the previous image-to-image translation works [5,20,23] rely on the assumption of pixel-to-pixel level correspondence to a certain extent. However, it is not suitable for sketch images since the supervision is noisy and weak at the pixel level. Photo-to-sketch translation is employed to generate sketches from a CNN encoder embedding feature, resulting in clean and sharp line strokes instead of pixel images. Sketch-pix2seq [4], AI-sketcher [3] and photo-to-sketch [17] has extended the sketch generation technique to extract structure information via CNN-based encoder. These models capture the relative positions of strokes in the sketch to improve drawing quality. Sketch-pix2seq [4] was the first to generate sketches of multiple categories, AI-sketcher [3] developed a high-quality sketch generation model combining CNN-based encoder and RNN-based encoder. Moreover, photo-to-sketch [17] applied the sketch generation method to the real-world photos, such as shoes and chairs, with the idea of cycle consistency. These methods contributes to sketch generation from images, but there are still some issues that can be further studied, such as exploring DCRs shared among different categories.

### 2.3. Sketch segmentation and labeling

It is a hard task to segment all pixels with the right labels, even with well-designed models and parameters. Recent studies [8,13,15] have explored multiple data-driven approaches to achieve sketch pixel image segmentation of free-hand sketch images. There are also studies [9,12,19] based on the RNN model to achieve segmentation and labeling on sketch point sequences. Facing the difficulty of segmenting and labeling on each pixels or point coordinates, we are still able to disentangle the whole sketch into separated components in the view of representations.

## 3. Method

### 3.1. Sketch, component and strokes

We represent a sketch as a list of points, and each point can be drawn via a 5-D stroke vector as in Ha and Eck [6]: $s_{tj} = [\Delta x, \Delta y, p_1, p_2, p_3]$, where the first two elements are the offsets in the $x$ and $y$ directions of the pen moving from the previous point to the current point, $[p_1, p_2, p_3]$ is a one-hot binary vector of three possible pen states, indicating whether to sketch, whether to lift the pen up until the next point and whether to stop sketching. Then we divide all strokes $\{s_{tj}\}$ into $n$ components. Each component is composed of a sequence of strokes $s_{tj}$ annotated with component type category $cls$ and location bounding box $loc$. Remind that stroke $s_{tj}$ in our paper represents the offset between the adjacent point coordinates and pen states while sketching. It has a different meaning from Chinese character strokes, i.e.,

$$\mathbf{Component_t} = S_t\{s_{t1}, s_{t2}, \ldots, s_{tk_i}\}, cls_t, loc_t, \quad (1)$$

where $s_{tj}$ means the $j$th stroke in $t$th component of the sketch.

### 3.2. Our model architecture

Our work tackles the problem of learning DCRs for sketch generation. Our model includes three main modules, as illustrated in Fig. 2, a CNN autoencoder module $E_I$ as well as $D_I$ for learning image-level representations $f_{conv}$ of pixel image, an LSTM autoencoder module $E_S$ as well as $D_S$ for learning the corresponding DCRs $z_t$ of sketch components, and an FDM for learning to disentangle image-level representations. During test phase, our model first gets the global image representation of input pixel image $\mathbf{I}$ via image encoder $E_I$, then disentangles the representation into DCRs via FDM, finally outputs the sketch component-by-component. Section 3.2.1 proposes FDM, the main difference among our method and existing methods. Section 3.2.2 introduces

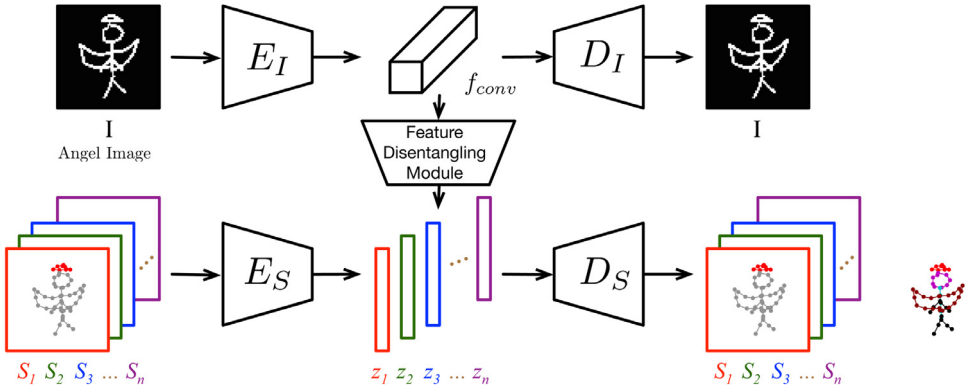


**Fig. 2.** Overview of model architecture, including image-level encoder-decoder module $E_I, D_I$ for learning the image-level representation on pixel image $\mathbf{I}$, component-level encoder-decoder module $E_S, D_S$ for learning DCR on each component $S_t$ and FDM for disentangling image-level representations to DCRs.

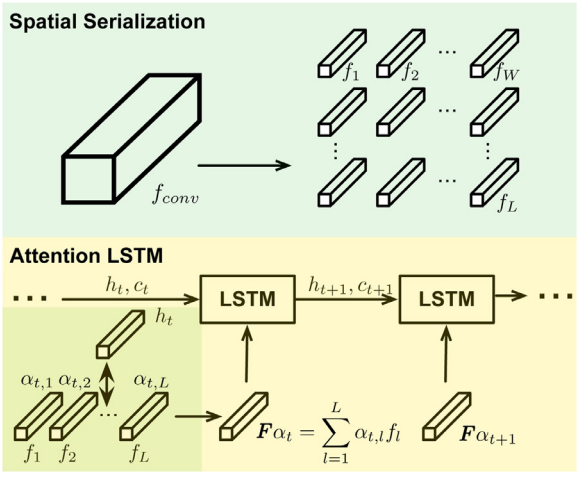


**Fig. 3.** Features disentangling module, disentangling the image-level feature $f_{conv}$ into DCRs $\{\tilde{z}_t\}_{t=1}^{t=n}$. We predict the distribution of DCRs and the location of the components by FDM.

the LSTM autoencoder, which encodes the sketch components to $z_t$ and generates sketches based on these $z_t$ and component locations. Section 3.2.3 describes how to extract suitable global image features while preserving local details for potential disentangle to DCRs.

### 3.2.1. Feature disentangling module

The main purpose of this part is to disentangle image-level feature $f_{conv} \in \mathbb{R}^{W\times H\times C}$ to DCRs $\{z_1, z_2, \ldots, z_n\}$ where $z_t \in \mathbb{R}^D, n \le N$, and $N$ is the maximum number of components in all sketches. Besides, we need to regress the bounding boxes $loc_t$ of each components because relative coordinates can be calculated via component strokes. Our disentangling module is a spatial attention LSTM model [14], composed of three sub-modules: *spatial serialization*, *attention LSTM*, and *mapping network* (Fig. 3).

*Spatial Serialization Network* takes an image-level feature $f_{conv} \in \mathbb{R}^{W\times H\times C}$ as input, and then expands the channel dimensions to get feature $f'_{conv} \in \mathbb{R}^{W\times H\times C'}$ with a $1\times 1$ convolution layer, and serializes the feature according to their spatial regions. We obtain a spatial embedding sequence $\boldsymbol{F} = [f_1, f_2, \ldots, f_l, \ldots, f_L]$ where $f_l \in \mathbb{R}^{C'}, L = W \times H$.

*Attention LSTM Network* Given the spatial image features $\boldsymbol{F} \in \mathbb{R}^{C'\times L}$ and hidden state $h_t \in \mathbb{R}^d$ of the LSTM, we feed them through a linear layer neural network respectively to get vectors $\boldsymbol{u} \in \mathbb{R}^{d'\times L}$, $v_t \in \mathbb{R}^{d'}$, and get inner product followed by a softmax function to generate the attention distribution over the $L$ regions of the image:

$$\begin{aligned} \boldsymbol{u} &= \tanh(W_f \boldsymbol{F} + b_f), \boldsymbol{u} \in \mathbb{R}^{d'\times L} \\ v_t &= \tanh(W_h h_t + b_h), v_t \in \mathbb{R}^{d'} \\ \alpha_t &= \mathrm{softmax}(\boldsymbol{u}^{\mathrm{T}} v_t), \alpha_t \in \mathbb{R}^L. \end{aligned} \tag{2}$$

Based on the attention distribution, the input of LSTM can be obtained by $\boldsymbol{F}\alpha_t = \sum_{l=1}^{L} \alpha_{t,l} f_l$, where $\alpha_{t,l}$ is the $l$th value of $\alpha_t$. Then the next hidden state $h_{t+1}$ and cell state $c_{t+1}$ of the LSTM can be calculated as:

$$h_{t+1}, c_{t+1} = \mathrm{LSTM}(\boldsymbol{F}\alpha_t, h_t, c_t). \tag{3}$$

*Mapping Network* learns a mapping from hidden state $h_t$ to the means and variances $\tilde{\mu}_t$, $\tilde{\sigma}_t$ of the distribution $p(\tilde{z}_t)$ with two fully connected layers (FC512-BatchNorm512-FC128), from which the predicted DCR $\tilde{z}_t$ is sampled. $\odot$ is Hadamard product.

$$\tilde{z}_t = \tilde{\mu}_t + \tilde{\sigma}_t \odot \mathcal{N}(0, I). \tag{4}$$

We also predict the bounding boxes $\tilde{loc}_t = [x, y, w, b]$ with two fully connected layers (FC512-BatchNorm512-FC4), where $x, y$ for the component center, and $w, b$ for the component scale. We can combine all translated and scaled components as generated sketches. Since different sketches may have different number of components, we set the ground truth $loc_t$ as $(0, 0, 0, 0)$ for $n < t \le N$. Similar to variational inference [10], the objective is to maximize an evidence lower bound (ELBO) using a log-likelihood based reconstruction term. The loss contains three parts. One is the Kullback–Leibler divergence between two Gaussian distributions of $z_t$ and Gaussian distribution. The second is the Kullback–Leibler divergence between the distributions of the DCR $\tilde{z}_t$ calculated from FDM and the ground truth DCR $z_t$ calculated from the component encoder $E_S$ (see details in Section 3.2.2). The third is the $\mathcal{L}_2$ loss of the locations of the components. The FDM loss for each sketch is as follows:

$$\begin{aligned} \mathcal{L}^{\mathrm{feat}} = &\sum_{t=1}^{n} [-\mathcal{D}_{KL}(p(\tilde{z}_t) \| \mathcal{N}(0, I)) - \mathcal{D}_{KL}(p(\tilde{z}_t) \| p(z_t))] \\ &+ \sum_{t=1}^{N} [\mathcal{L}_2(\tilde{loc}_t, loc_t)], \end{aligned} \tag{5}$$

where $p(\tilde{z}_t), \tilde{loc}_t$ are the predictions of FDM, and $p(z_t), loc_t$ are the ground truth distribution of DCR and the ground truth location of the component. It is difficult, sensitive and hard for generation to regress DCR directly. In contrast, learning the distribution of DCR contributes to convergence and better performance of sketch generation. We replace the Kullback–Leibler divergence with $\mathcal{L}_2$ loss function of direct regression of DCR as a comparative experiment 'w/o KL', see more details in Section 4.4 ablation study.

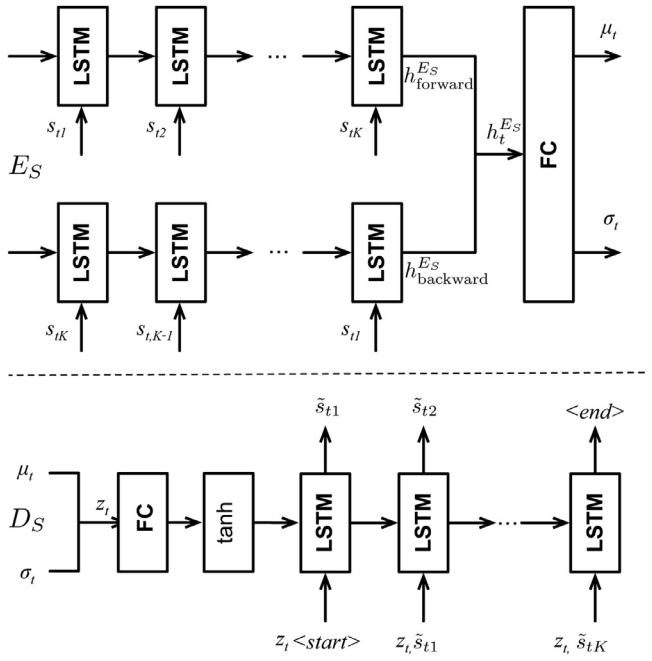


**Fig. 4.** Component-level autoencoder module, including component-level encoder $E_S$, component-level decoder $D_S$ on all basic components. The autoencoder module learns DCR $z_t$ of each component $S_t$.

### 3.2.2. Component-level encoder-decoder module

The main purpose of this module, including stroke-level encoder $E_S$, stroke-level decoder $D_S$, is to model various components of the sketch. There are various categories of sketches composed of limited types of basic components. For example, we draw an animal sketch with the animal's head, eyes, body, legs, and so on. The differences between different animals can be found in the components as well as the relationship between these components. We can model different categories better with the knowledge in learning basic components, especially in multi-class sketch generation tasks.

Unlike the previous methods [1,3,6] which takes all strokes of a sketch to obtain a hidden vector for sketch generation, our method preprocesses the whole sketch into components, and then learns DCRs on these components shared among different categories. Specifically, our model takes strokes sequence $S_t\left\{s_{t1}, s_{t2}, \ldots, s_{tk_t}\right\}$ of $t$th component of sketch as input, gets DCR $z_t$ represented by a bidirectional LSTM, and passes $z_t$ through an unidirectional LSTM to output the sequence strokes $S_t$. As illustrated in Fig. 4, our encoder-decoder module on the components is composed of component-level encoder $E_S$, component-level decoder $D_S$. We feed the strokes $\left\{s_{t1}, s_{t2}, \ldots, s_{tk_t}\right\}$, and also the same sequence in reverse order, $\left\{s_{tk_t}, \ldots, s_{t2}, s_{t1}\right\}$ into BiLSTM made up by two encoding LSTMs, to obtain two sequences of hidden states and then concatenate the last forward and backward hidden states to a joint hidden state:

$$h_t^{E_S} = \left[h_{\text{forward}}^{E_S}; h_{\text{backward}}^{E_S}\right]. \tag{6}$$

The hidden state $h_t^{E_S}$ is then transformed into two vectors $\mu_t$ and $\sigma_t$, which are the parameters of a normal distribution used to capture the distribution of the $t$th component. A latent vector $z_t$ is randomly sampled from the distribution via a vector of IID Gaussian variables, as the approach in VAE [10]:

$$\mu_t = W_\mu h_t^{E_S} + b_\mu, \sigma_t = \exp\left(\frac{W_\sigma h_t^{E_S} + b_\sigma}{2}\right).$$
$$z_t = \mu_t + \sigma_t \odot \mathcal{N}(0, I). \tag{7}$$

Our decoder $D_S$ is a unidirectional LSTM that outputs strokes $\left\{\tilde{s}_{t1}, \tilde{s}_{t2}, \ldots, \tilde{s}_{tk_t}\right\}$ step by step if given DCR $z_t$, where $\tilde{s}_{tj} = [\tilde{\Delta x}, \tilde{\Delta y}, q_1, q_2, q_3]$. We adopt $\mathcal{L}_2$ loss to measure the similarity between the generated offset $\left[\tilde{\Delta x}, \tilde{\Delta y}\right]$ sequence and the original offset $[\Delta x, \Delta y]$ sequence, and use the cross-entropy loss $\mathcal{L}_{\text{CE}}$ to measure the difference between the predicted pen state $[q_1, q_2, q_3]$ and the ground truth pen state $[p_1, p_2, p_3]$. As the length $k_t$ of $t$th component is usually shorter than $K$ (the max length of all strokes), we set $s_{tj}$ to be $(0, 0, 0, 0, 1)$ for $j \geq k_t$ to pad strokes to the fixed length $K$. The reconstruction loss of all components of one sketch is:

$$\mathcal{L}^S = \sum_{t=1}^{n}\sum_{j=1}^{K}\left(\mathcal{L}_2(\tilde{s}_{tj}[:2], s_{tj}[:2]) + \mathcal{L}_{\text{CE}}(\tilde{s}_{tj}[2:], s_{tj}[2:])\right), \tag{8}$$

where $\tilde{s}_{tj}$, $s_{tj}$ are generated and original sequences of component strokes, respectively.

### 3.2.3. Image-level encoder-decoder module

The image autoencoder module consists of CNN encoder and decoder. CNN encoder $E_I$ takes a pixel image $\mathbf{I}$ as input to get the representation, followed by CNN decoder $D_I$ to reconstruct the image as $\tilde{\mathbf{I}}$. We denote the Convolution-BatchNorm-leakyReLU block as $\text{CONV}_{KernelSize}^{Stride}Channel$, and set padding as 1 when kernel size is 3, padding as 0 when kernel size is 1. Then we build image encoder as $\text{CONV}_3^2 64 - \text{CONV}_3^1 128 - \text{CONV}_1^1 64 - \text{CONV}_3^2 128 - \text{CONV}_3^2 64 - \text{CONV}_3^2 64 - \text{CONV}_3^2 64$. The last three block output features with size $8 \times 8$, $4 \times 4$ and $2 \times 2$ respectively when input size is $64 \times 64$. We upsample the last block features with stride 4, and the second last block features with stride 2, and concate these upsample features with the third last block features to get image-level representation $f_{conv}$ as size $8 \times 8$ with 192 channels. Denote TransposeConvolution-BatchNorm-ReLU block as CT, and we build decoder as $\text{CT}_1^1 128 - \text{CT}_3^2 64 - \text{CT}_1^1 128 - \text{CT}_3^2 64 - \text{CT}_3^2 32 - \text{CONV}_3^1 1$. Remind that the final block in decoder is convolution block rather than transpose convolution block. We adopt MSE loss function $\mathcal{L}_{\text{MSE}}$ on the reconstruction images.

$$\mathcal{L}^{\text{I}} = \mathcal{L}_{\text{MSE}}(\tilde{\mathbf{I}}, \mathbf{I}), \tag{9}$$

Representation $f_{conv}$ combining multi-scale layer features performs better than the feature from one of the layers. We replace $f_{conv}$ with only features from one of the last three blocks of encoder, and increase the channel of the layer to 192 and upsample to $8 \times 8$ for fair comparison. We guess that both global image features and local image features are beneficial to obtain DCRs. See more details in Section 4.4 ablation study.

## 3.3. Training and inference

Our training procedure has three main information paths, one is self-reconstruction of sketch image, $\mathbf{I} \rightarrow E_I(\mathbf{I}) \rightarrow D_I(E_I(\mathbf{I}))$, one is self-reconstruction of the component sequences, $S_t \rightarrow E_S(S_t) \rightarrow D_S(E_S(S_t))$, the last is from the pixel image to the sketch, $\mathbf{I} \rightarrow E_I(\mathbf{I}) \rightarrow \left\{\tilde{z}_t, \tilde{loc}_t\right\}_{t=1,2,\ldots,n} \rightarrow \left\{D_S(\tilde{z}_t), \tilde{loc}_t\right\}_{t=1,2,\ldots,n}$. DCRs come from FDM rather than $D_S$ as in the last information paths at testing phase.

We adopt four kinds of training strategies. The first one is training CNN autoencoder $E_I, D_I$ with $\mathcal{L}^I$ and LSTM autoencoder $E_S, D_S$ with $\mathcal{L}^S$ to get $f_{conv}$ and DCRs individually, then freeze these four modules and train FDM to learn disentangling $f_{conv}$ into DCRs with $\mathcal{L}^{feat}$. The second one is training all five modules together with sum of three losses $\mathcal{L}^I$, $\mathcal{L}^S$ and $\mathcal{L}^{feat}$. The third one is training $E_I, D_I, E_S, D_S$ individually to get pretrain modules (the same as the first strategy), and fine-tune these five modules via sum of all three losses. The fourth one is progressive training strategy as follows.

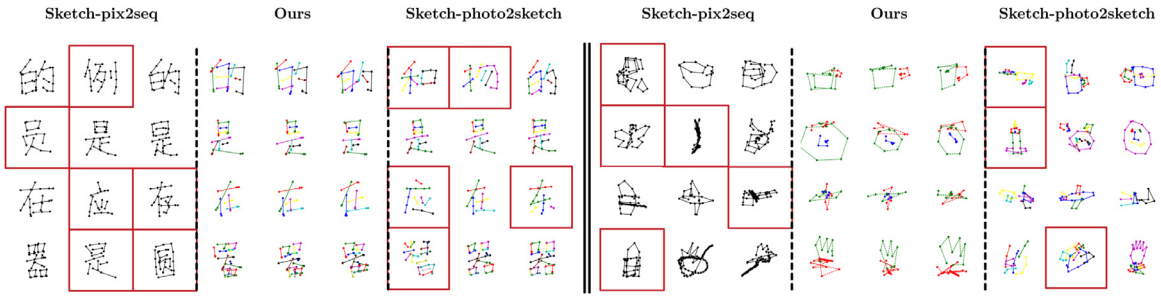


**Fig. 5.** Multi-class generation results. The left parts are results on Chinese characters while the right on SPG (coffee cup, alarm clock, airplane, campfire from up to bottom). The samples in the red rectangle are wrong as their categories are not expected. (About 1/2 are wrong for sketch-pix2seq, About 1/3 are wrong for sketch-photo2sketch, and the quantitative results are shown in Table 1.) (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Our test goal is to generate sketch directly from $\tilde{z}_t, \tilde{loc}_t$ rather than $z_t, loc_t$. Even if the distribution $p(\tilde{z}_t)$ of DCR calculated from FDM is close to the ground truth distribution $p(z_t)$ of DCR calculated by $E_S$, the generated component $D_S(\tilde{z}_t)$ may be sometimes different from $D_S(z_t)$. We have introduced that $D_S$ generates sketches based on DCRs sampled from $p(z_t)$ in Fig. 4. Now we change the sample strategy as sampling DCRs from $p(z_t)$ with probability $P_{\text{progress}}$ and from $p(\tilde{z}_t)$ with probability $(1 - P_{\text{progress}})$. At the first some epochs of training, we set $P_{\text{progress}}$ close to 1 and generate sketches mostly based on DCRs sampled from $p(z_t)$. As the progresses of training, we can gradually decay $P_{\text{progress}}$ to 0 and generate sketches mostly based on DCRs sampled from $p(\tilde{z}_t)$. We set $P_{\text{progress}} = 0.9^{\text{epoch}}$ in experiments. The training loss under this strategy is the sum of all three loss function $\mathcal{L}^I$, $\mathcal{L}^{S-progress}$ and $\mathcal{L}^{feat}$. $\mathcal{L}^{S-progress}$ is the same form as $\mathcal{L}^S$ but only replacing the input of $D_S$ with new sampled DCRs. We compare these four training strategies and find the fourth one outperforms others training strategies. See more details in Section 4.4 ablation study.

## 4. Experiments

We do experiments on handwritten Chinese characters and sketches of human painting. We constructed an online handwritten Chinese character dataset (OLCC) for this purpose. It contains 171,000 sketches distributed over 500 categories, with each sketch manually annotated into 1–16 components. We hired 187 people and 155 wrote 500 categories 500 categories twice, 32 wrote 500 categories once and 13 quit halfway. We split the dataset into train set and valid set as 283:59. We also test our method on simplified drawings from the Sketch Perceptual Grouping dataset [12], the largest free-hand sketch with component labels, 3–8 components for each category. It contains 25 categories, 800 samples each category, composelected from QuickDraw, which is by far the largest public sketch dataset collected by Google. Please refer to the appendix for detailed experimental settings.

### 4.1. Multi-class generation

Our method can generate samples with more specific component information because of FDM. And our model is more stable to generate tens of categories because it can capture both component-level and image-level representations. Fig. 5 shows the comparison of our model with sketch-pix2seq and sketch-photo2seq. The accuracy of our results is greatly improved after extracting the DCRs rather than mapping the whole image representation to all of the stroke sequences.

### 4.2. Generation quality

In order to quantify the quality of multi-class generation and the effectiveness of disentangling, we compare the classification accuracy of generated images and that of generated components. These classifiers are trained on training dataset, please refer to supplementary for more details about classifiers. Table 1 presents the results of the ablation study and comparison to sketch-pix2seq and sketch-photo2seq. As there are not component-level labels in sketch-pix2seq or sketch-photo2seq generated results, we use sketch-segmentation [9] to get disentangled component-level labels on primary data and the results generated by sketch-pix2seq and sketch-photo2seq methods. The accuracy of the generated images shows generation quality on multiple categories, and the accuracy of the disentangled components shows the effectiveness of disentangling. Table 1 shows our method outperforms others by the better generation quality and disentangling accuracy.

### 4.3. Generation diversity

We compare the generation diversity [3] of our method, sketch-pix2seq, and sketch-photo2seq in five categories, respectively, based on two datasets. Specifically, we generate a set of 50 sketches in each of the five preselected categories in each dataset. We calculate the pairwise distances between generated sketches in the same category based on perceptual hash [21]. A large average distance means higher generation diversity. We find that the mean of distances in sketch-pix2seq is generally larger than the mean in sketch-photo2seq. So we do a $t$-test experiment to compare our method and sketch-pix2seq. The unpaired $t$-test in Table 2 shows that our method and sketch-pix2seq have no significant difference, although samples generated by our method are almost classified in the correct categories, which means our method has higher generation diversity in the correct categories.

### 4.4. Ablation study

We compare three kinds of image-level representations as $f_{conv}$. The last three convolution block features of our $D_I$ are 64 channels with $8 \times 8$, $4 \times 4$, $2 \times 2$ sizes. We upsample $4 \times 4$ features to $8 \times 8$ with stride 2, and $2 \times 2$ features to $8 \times 8$ with stride 4. Then we concate these three features and obtain 192 channels with $8 \times 8$ size $f_{conv}$. For comparison, we just use one of the three layer features as $f_{conv}$. Specifically, we take the $i$th last features and upsample $(3 - i)$ times with stride 2 to get $8 \times 8$ representations, where $i = 1, 2, 3$, then expand channels to 192 with a $1 \times 1$ convolution layer, and denote the representations as 'w $f_{2\times2}$', 'w $f_{4\times4}$', and 'w $f_{8\times8}$', respectively. The results are shown in the first part of Table 3. We think multi-layer representations contribute to disentangling images into components since different components vary in shape and size.

We then compare three kinds of FDM. We explore that whether learning DCR distribution $p(\tilde{z}_t)$ to ground truth distribution $p(z_t)$ is more effective than learning direct regression of DCR or not. We

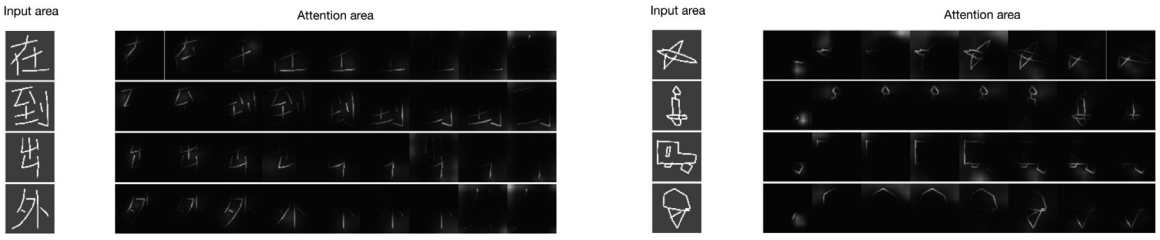


**Fig. 6.** Attention examples. The focus area of attention moves with the change of the generated components.

**Table 1**
Comparison of generation quality. The image classifier and component classifier are trained on the training dataset and used to test the quality of our sketch generations. Human level means the sketch images were drawn by humans.

| Model | OLCC | | SPG | |
|---|---|---|---|---|
| | acc@image | acc@component | acc@image | acc@component |
| Sketch-Segmentation [9] | – | 84.7% | – | 65.4% |
| Sketch-pix2seq [4] | 47.8% | 45.6% | 33.1% | 15.2% |
| Sketch-photo2seq [17] | 71.4% | 33.7% | 55.3% | 13.1% |
| Ours | **87.9%** | **99.9%** | **66.0%** | **82.3%** |
| Human Level | 100% | – | 92.3% | – |

**Table 2**
Comparison of generation diversity.

| **Model** | □ | △ | □ | △ | □ | △ | □ | △ | □ | △ |
|---|---|---|---|---|---|---|---|---|---|---|
| **Mean** | 31.923 | 32.049 | 30.866 | 30.792 | 31.906 | 31.863 | 31.754 | 31.806 | 31.974 | 31.952 |
| **SD** | 16.830 | 15.178 | 19.159 | 21.188 | 15.190 | 16.261 | 16.280 | 17.327 | 16.277 | 16.178 |
| **t** | 0.777 | | 0.405 | | 0.268 | | 0.310 | | 0.135 | |
| **p** | 0.437 | | 0.686 | | 0.789 | | 0.756 | | 0.892 | |

| **Model** | □ | △ | □ | △ | □ | △ | □ | △ | □ | △ |
|---|---|---|---|---|---|---|---|---|---|---|
| **Mean** | 31.919 | 32.104 | 31.895 | 32.088 | 32.122 | 31.916 | 31.890 | 31.906 | 31.846 | 31.934 |
| **SD** | 17.032 | 15.434 | 16.577 | 17.008 | 15.186 | 17.029 | 15.905 | 17.732 | 16.567 | 15.787 |
| **t** | 1.138 | | 1.168 | | 1.268 | | 0.394 | | 0.546 | |
| **p** | 0.255 | | 0.243 | | 0.205 | | 0.968 | | 0.585 | |

□**Sketch-pix2seq**, △**Our model**

**Table 3**
Ablation study.

| Model | OLCC Accuracy | | SPG Accuracy | |
|---|---|---|---|---|
| | Image | Component | Image | Component |
| w $f_{2\times2}$ | 64.7% | 75.3% | 34.2% | 45.1% |
| w $f_{4\times4}$ | 76.2% | 84.6% | 56.4% | 64.2% |
| w $f_{8\times8}$ | 80.4% | 87.3% | 60.1% | 78.5% |
| w/o lstm | 44.0% | 85.2% | 35.4% | 72.3% |
| w/o KL | 75.1% | 94.5% | 48.9% | 80.5% |
| w/o attn | 83.6% | 95.7% | 63.5% | 81.2% |
| independent | 47.2% | 54.9% | 35.3% | 47.6% |
| end-to-end | 77.6% | 78.8% | 58.9% | 67.1% |
| fine-tune | 82.4% | 81.8% | 62.3% | 79.2% |
| Ours | **87.9%** | **99.9%** | **66.0%** | **82.3%** |

replace the two negative Kullback–Leibler divergenc in Eq. (5) as $\mathcal{L}_2(\tilde{z}_t, z_t)$, denoted as w/o KL. We also try removing attention mechanism, and input $f_1, f_2, \ldots, f_L$ sequentially into LSTM, denoted as w/o attn. And we try a toy model 'w/o lstm' consist of a $1 \times 1$ convolution layer followed by three fully connected layer replacing FDM. We first change the channel of $f_{conv}$ to $N$ with a $1 \times 1$ convolution layer, then flatten each channel $8 \times 8$ to vector $\mathbb{R}^{64}$, and map each vector to each DCR via three linear layers FC64-BatchNorm-FC512-BatchNorm-FC(128+4), where first 128 dimension for $z_t$ and last 4 dimension for bounding box $loc_t$. Results show the improvements brought by LSTM, variational inference and attention mechanism.

We finally compare four training strategies introduced in Section 3.3. The fisrt one 'independent' is training CNN autoencoder and LSTM autoencoder for 40 epochs individually, and training only FDM for 40 epochs. The second one 'end-to-end' is training all five modules together with sum of losses for 80 epochs. The third one 'fine-tune' is training individually for 40 epochs, and fine-tune all modules for 40 epochs. The fourth one 'progress' is progressive training strategy that sampling DCRs from $p(z_t)$ with probability $P_{progresses}$ and from $p(\tilde{z}_t)$ with probability $(1 - P_{progresses})$, where $P_{progresses} = 0.9^{\text{epoch}}$, epoch $= 1, \ldots, 80$. The results 'independent' are quite far behind others, we guess it is hard to learn FDM if fixing the image-level representations and DCRs.

### 4.5. Visualization

*Attention mechanism* Our FDM is an attention LSTM to disentangle the image-level feature into the DCRs. Section 4.4 show the multi-layer representations and attention mechanism help to improve the performance of multi-class sketch generations. In this

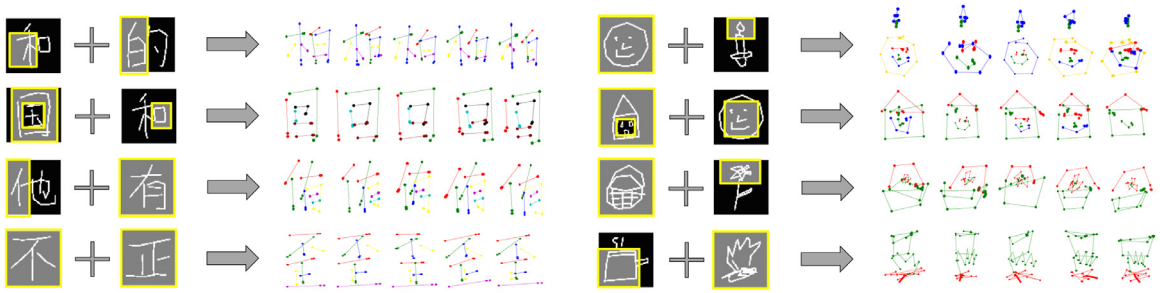

**Fig. 7.** Composed generation results. The model decomposes two sketch images into components, combines the yellow highlight parts at stroke-level representation, and outputs new class examples. For example, the second line on the right is replacing windows of house with the facial expression. Because the model can output the sketch component-by-component, thus we can manually combine different components from two input images.

paragraph, we upsample the attention map to the original image size and visualize the attention distribution during the sketch generation process as show in Fig. 6. Our FDM can focus on the corresponding regions component-by-component.

*Manually generation* We analyze the sketch generation of new categories with learned components in this paragraph. Once DCRs of various components are learned, we can manually generate sketches by composing DCRs as shown in Fig. 7. We provide manual how new categories are composed of learned DCRs by setting the value of $loc_t$. For example, people usually express "It is a happy house" by sketching a happy face on the wall as shown in the right second line of Fig. 7. If provided DCRs of happy faces and houses as well as their location on the final sketch, the model can output a house with a happy face. Although the manual generation conditions are harsh, it still provides a potential research direction on how to generate sketches flexible.

## 5. Conclusion

In this paper, we tackle the problem of generating sketches component-by-component. We have proposed a sketch generation method based on DCRs. Experiments on two datasets show that our method outperforms the state-of-the-art method in terms of the component accuracy and sketch generation accuracy. It is more stable when training on tens of categories. In the future, we will investigate weather we can achieve unsupervised DCRs sketch generation. In addition, by making a large dataset of Chinese characters with component-level annotation available, we encourage further research and development in interactive teaching and sketch generation.

## Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgments

This work is funded by the Natural Science Fundation of China (NSFC) and the German Research Foundation (DFG) in Project Crossmodal Learning (NSFC 62061136001/ DFG TRR-169 and NSFC 61876095).

## Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.patrec.2021.01.016.

## References

[1] S. Balasubramanian, V. N. Balasubramanian, et al., Teaching GANs to sketch in vector format, arXiv preprint arXiv:1904.03620 (2019).
[2] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, S. Bengio, Generating sentences from a continuous space, arXiv preprint arXiv:1511.06349 (2015).
[3] N. Cao, X. Yan, Y. shi, C. Chen, Ai-sketcher: A deep generative model for producing high-quality sketches, Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 33. No. 01. 2019.
[4] Y. Chen, S. Tu, Y. Yi, L. Xu, Sketch-pix2seq: a model to generate sketches of multiple categories, arXiv preprint arXiv:1709.04121 (2017).
[5] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Advances in Neural Information Processing Systems, 2014, pp. 2672–2680.
[6] D. Ha, D. Eck, A neural representation of sketch drawings, arXiv preprint arXiv:1704.03477 (2017).
[7] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[8] Z. Huang, H. Fu, R.W.H. Lau, Data-driven segmentation and labeling of freehand sketches, ACM Trans. Graph. (TOG) 33 (6) (2014) 175.
[9] K. Kaiyrbekov, M. Sezgin, Stroke-based sketched symbol reconstruction and segmentation, arXiv preprint arXiv:1901.03427 (2019).
[10] D.P. Kingma, M. Welling, Auto-encoding variational bayes, arXiv preprint arXiv:1312.6114 (2013).
[11] B.M. Lake, R. Salakhutdinov, J.B. Tenenbaum, Human-level concept learning through probabilistic program induction, Science 350 (6266) (2015) 1332–1338.
[12] K. Li, K. Pang, J. Song, Y.-Z. Song, T. Xiang, T.M. Hospedales, H. Zhang, Universal sketch perceptual grouping, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 582–597.
[13] L. Li, H. Fu, C.-L. Tai, Fast sketch segmentation and labeling with deep learning, IEEE Comput. Graph. Appl. 39 (2) (2018) 38–51.
[14] J. Lu, C. Xiong, D. Parikh, R. Socher, Knowing when to look: Adaptive attention via a visual sentinel for image captioning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 375–383.
[15] R.G. Schneider, T. Tuytelaars, Example-based sketch segmentation and labeling using CRFS, ACM Trans. Graph. (TOG) 35 (5) (2016) 151.
[16] M. Schuster, K.K. Paliwal, Bidirectional recurrent neural networks, IEEE Trans. Signal Process. 45 (11) (1997) 2673–2681.
[17] J. Song, K. Pang, Y.-Z. Song, T. Xiang, T.M. Hospedales, Learning to sketch with shortcut cycle consistency, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 801–810.
[18] J. Wagemans, J.H. Elder, M. Kubovy, S.E. Palmer, M.A. Peterson, M. Singh, R. von der Heydt, A century of gestalt psychology in visual perception: I. Perceptual grouping and figure–ground organization, Psychol. Bull. 138 (6) (2012) 1172.
[19] X. Wu, Y. Qi, J. Liu, J. Yang, Sketchsegnet: a RNN model for labeling sketch strokes, in: 2018 IEEE 28th International Workshop on Machine Learning for Signal Processing (MLSP), IEEE, 2018, pp. 1–6.
[20] Z. Yi, H. Zhang, P. Tan, M. Gong, Dualgan: unsupervised dual learning for image-to-image translation, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2849–2857.
[21] C. Zauner, Implementation and benchmarking of perceptual image hash functions. Master's thesis, Austria. (2010).
[22] X.-Y. Zhang, F. Yin, Y.-M. Zhang, C.-L. Liu, Y. Bengio, Drawing and recognizing chinese characters with recurrent neural network, IEEE Trans. Pattern Anal. Mach. Intell. 40 (4) (2018) 849–862.
[23] J.-Y. Zhu, T. Park, P. Isola, A.A. Efros, Unpaired image-to-image translation using cycle-consistent adversarial networks, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2223–2232.