Sketchformer++: A Hierarchical Transformer Architecture for Vector Sketch Representation

Pengfei Xu[1], Banhuai Ruan[1], Youyi Zheng[2], and Hui Huang[1]

[1] Shenzhen University, Shenzhen, China
[2] Zhejiang University, Hangzhou, China

Abstract. With the rising ubiquity of digital touch devices and sketch-based interfaces, freehand sketching has become an essential mode of visual communication. Nevertheless, interpreting these often ambiguous and sparse sketches poses challenges for computers. This paper presents *Sketchformer++*, a hierarchical transformer architecture for the neural representation of vector sketches. It treats a vector sketch as a three-level structure, *i.e.*, sketch level, stroke level, and segment level. Three self-attention modules are adopted in the network architecture, corresponding to the sketch hierarchy. The semantics of sketches are aggregated from local to global, resulting in neural representations of sketches. Extensive experiments show that *Sketchformer++* exhibits superior performance in various downstream tasks, including sketch reconstruction, sketch recognition, and sketch semantic segmentation, demonstrating its robustness and effectiveness in sketch representation.

Keywords: Vector sketch · Transformer · Hierarchy · Neural representation · Sketch recognition · Sketch semantic segmentation.

1 Introduction

With the continuous evolution of digital touch devices, such as smartphones and tablets, freehand sketching is increasingly becoming one of the prevalent and efficient means of visual communication. Different from photos that possess rich textural and color information, sketches are often in an ambiguous and sparse form. While humans can easily recognize the semantics of sketches, interpreting sketches remains a challenge for computers.

Many previous works have tried to understand the semantics of sketches in different tasks, *e.g.*, sketch recognition [21, 31, 32], sketch semantic segmentation [15, 27, 32, 36], and sketch-based image retrieval [2, 6, 20, 33]. They treat sketches as different modalities. Some methods [13, 15, 30, 37, 38] consider sketches as raster images and use convolutional neural networks [12] to estimate the semantics of sketches. These methods might be versatile for sketches in various drawing styles, *e.g.*, sketches containing hatching or shadow, but they neglect the structures of vector sketches. Alternatively, sketches can be viewed as graphs [4, 25, 28, 32, 35] and fed to graph neural networks [32] for semantic estimation. These methods emphasize the structure of sketches but ignore the drawing order, as discussed in [36]. Another line of research [8, 18]

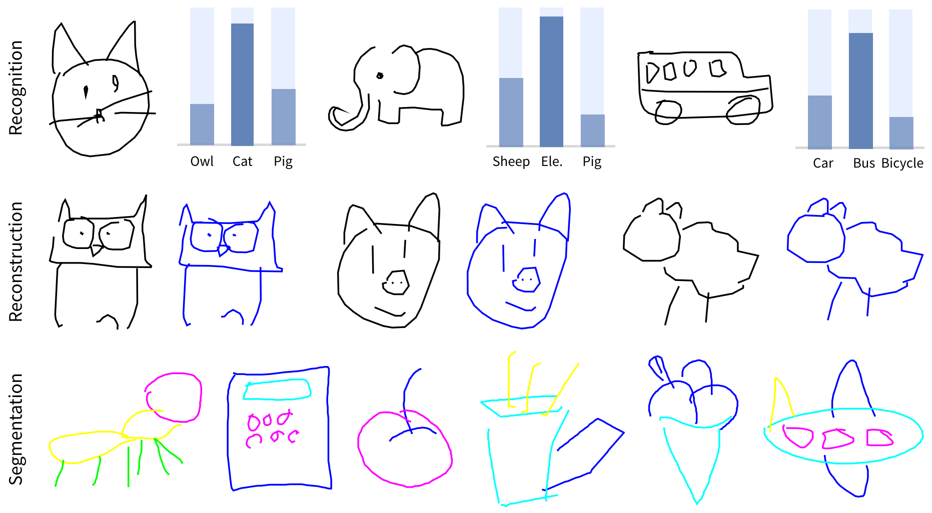


Fig. 1. Our method can help perform various downstream tasks, including sketch reconstruction, sketch recognition, and sketch semantic segmentation.

considers sketches as sequences of points and adopts RNN-based [8] or transformer-based [18] architectures to extract the semantics of sketches. These methods exploit the structure and drawing order of sketches, but fail to consider the inherent hierarchy possessed by sketches.

Sketches naturally possess hierarchical structures [31, 32]: a sketch is composed of a set of strokes, and a stroke is composed of a sequence of points. Due to the existence of this hierarchy, the correlations between different parts of a sketch also have different levels. For example, there exist low-level correlations, *i.e.*, the correlations between points, and high-level correlations, *i.e.*, the correlations between strokes. Low-level correlations are critical for forming a local part of a sketch, *e.g.*, a stroke, and high-level correlations are important for composing a whole sketch. It would be essential to consider the correlations at different levels when estimating the semantics of sketches.

Based on the above observation, we propose a hierarchical transformer architecture for the neural representation of vector sketches, named *Sketchformer++*. We treat a vector sketch as a three-level structure, *i.e.*, sketch level, stroke level, and segment level. We introduce the extra segment level since some strokes in a sketch may have excessive lengths. Such long strokes may contain multiple semantics and should be decomposed into simpler entities [31]. With this hierarchical structure, we devise three corresponding levels of self-attention mechanisms. From the lowest level, *i.e.*, the segment level, to the highest level, *i.e.*, the sketch level, we hierarchically aggregate the sketch semantics from local to global, resulting in three latent spaces that embed segments, strokes, and sketches, respectively. The constructed latent spaces help perform various downstream tasks (see Figure 1), including sketch reconstruction, sketch recognition, and sketch semantic segmentation. Our experiments show that our method advances existing methods in these tasks, demonstrating the effectiveness of sketch neural representation estimated by our method.

2 Related work

Representation learning refers to the methods of automatically learning representations from raw data to make subsequent downstream tasks easier or more meaningful. Representation learning is a core topic in computer vision, especially when dealing with images. However, sketches differ from ordinary images in that they often contain only contours, edges, and certain feature lines without color and texture information. Therefore, representation learning for sketches requires specific strategies and methods. Before the widespread use of deep learning models, traditional image-based descriptors, such as the Histogram of Oriented Gradients (HOG) [10] and Scale-Invariant Feature Transform (SIFT) [11], were commonly used. Although limited in their ability to capture complex patterns, these handcrafted features provided a basic understanding of the sketch semantics. With the booming development of deep learning, many learning-based methods emerged to obtain neural representations of vector sketches. We review some representation learning methods for sketches and classify them according to their required sketch formats. Then we discuss the related methods that exploit the hierarchical structure of sketches.

Image-based methods. These methods treat sketches as raster images and feed them to the networks to obtain the neural representation of the sketches. Bhunia *et al.* [1] proposed a self-supervised learning approach, training an encoder-decoder architecture with raster sketches as inputs to obtain their neural representations. Yu *et al.* [33] proposed a multi-scale multi-channel deep neural network by using a CNN with a large convolutional kernel to adapt to the sparsity of stroke pixels. These aforementioned image-based methods have exclusively focused on the spatial information of sketches, neglecting the temporal and structural information inherent in the sketching process. Yet, the drawing sequence of sketches is paramount to understanding their essence. Consequently, our approach employs Transformer [23] as the backbone network to capture the temporal nuances of sketch drawing.

Graph-based methods. With the rise of Graph Neural Networks (GNN), many studies have attempted to employ this network to estimate the semantics of points. Wang *et al.* [26] introduced DGCNN, which uses GNNs to capture the relationship between local and global features of point sets, thereby obtaining their neural representations. Qi *et al.* [16] proposed SketchLattice, which integrates a raster sketch with a lattice, extracts key points on this raster sketch, and subsequently constructs a GNN-based encoder based on these key points. Zang *et al.* [34] partition a sketch image into multiple segments, each of which is encoded by a CNN. These neural representations of the segments are then fed to a GNN-based encoder-decoder architecture. These aforementioned graph-based approaches overlook the sequential information inherent in sketches. In contrast, a primary focus of our proposed method is to exploit both the local and global sequential information of sketches.

Sequence-based methods. Recent works have begun to directly take vector sketches as input, in the form of point sequences. A representative work is SketchRNN [8], which exploits a sequence-to-sequence (seq2seq) network architecture based on LSTM [9], and is trained on a large public sketch dataset named QuickDraw. The inherent limitation of this LSTM-based architecture results in the loss of semantic information when dealing with sketches containing long sequences. As the emergence of Transformer [23], this issue was resolved by Sketchformer [18], a transformer-based method. Although Sketchformer solves the problem of semantic information loss that exists in SketchRNN, it simply considers sketches as a sequence of points, neglecting the hierarchical structure possessed by sketches. In our research, we exploit the hierarchical structure of sketches and propose multiple attention mechanisms to extract the semantics of sketches from local to global. This makes our method more effective in estimating the neural representation of sketches.

Hierarchical structure of sketches. Our method is not the first that recognizes the importance of the sketch's hierarchical structure. Yang *et al.* [32] presented SketchGNN, a convolutional graph neural network for sketch semantic segmentation. This network extracts the sketch features as three levels, *i.e.*, point level, stroke level, and sketch level. Zheng *et al.* [36] proposed Sketch-Segformer, a transformer-based network for sketch semantic segmentation. This network exploits both sketch-wise and stroke-wise self-attentions. S^{3}Net [31] is a network for sketch recognition. It uses an RNN to extract segment-level features and a GCN to extract sketch-level features. SSR-GNNs [4] is a network that first estimates a sketch's stroke features and then constructs a graph to estimate the sketch feature. Although all these works have implicitly or explicitly exploited the hierarchical structure of sketches for different tasks, none of them have attempted to estimate the neural representation of sketches, which is, in contrast, the focus of our work.

3 Method

We propose a transformer-based deep neural network to estimate the neural representation of vector sketches. It exploits the inherent hierarchical structure of sketches. The hierarchy has three levels, *i.e.*, sketch level, stroke level, and segment level. At each level, we use a transformer to estimate the neural representations of sketches and use them to achieve different tasks including sketch recognition and sketch semantic segmentation. In the following, we first introduce the sketch formats adopted in our method and then explain the details of our method.

3.1 Data representation

Existing transformer-based methods usually represent a sketch as a sequence of points, with additional symbols to indicate the drawing state. For example, SketchRNN [8] and Sketchformer [18] adopt the 'stroke-5' format. With this format, a point is represented as $(\Delta x, \Delta y, p_1, p_2, p_3)$, where $(\Delta x, \Delta y)$ is the point's relative position to the previous

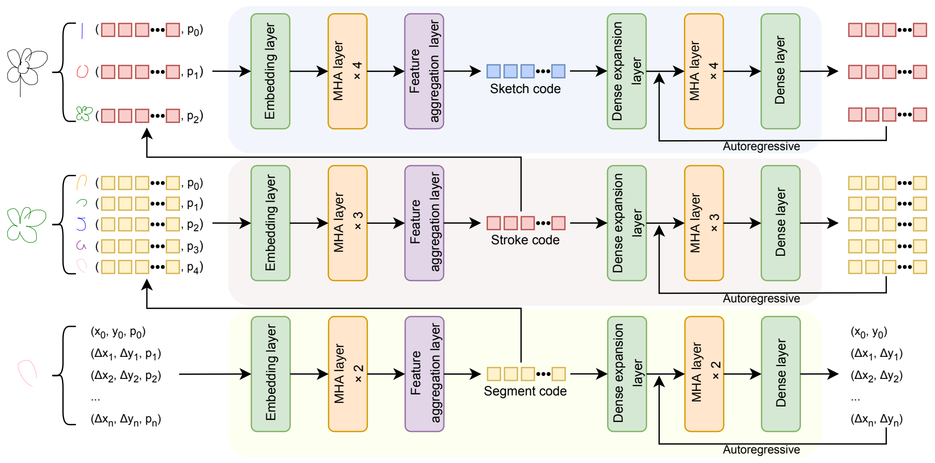


Fig. 2. The hierarchical transformer architecture of our method. Our network architecture consists of three levels, corresponding to the hierarchical structure of sketches. Given a sketch with a hierarchical structure, we can embed this sketch into a latent space in a bottom-up manner.

point; p_1 (draw), p_2 (lift), and p_3 (end) indicate the drawing states. Since our method exploits the hierarchical structures of sketches, we adopt the 'stroke-3' format to represent a point. Given a vector sketch, its sketch-level and stroke-level structures can be easily determined. The additional segment-level structure is obtained by splitting long strokes into short segments. We adopt a simple strategy for the splitting: we treat a segment as a sequence of N_{pt} points. After extracting the segment-level structure, we represent the vector sketch hierarchically.

At the segment level, a segment is a sequence of points. Each point in this segment is represented as $(\Delta x, \Delta y, p)$, where p indicates the validity of the point. For the first point of the segment, we store its absolute position. According to our stroke splitting strategy, a segment contains at most N_{pt} points. If a segment contains less than N_{pt} points, we use a special token to pad the corresponding point sequence.

At the stroke level, a stroke is a sequence of segments. Each segment in this stroke is not represented as a sequence of points, but a latent code estimated by our method to be described later. We adopt the segment's neural representation since it has already aggregated the semantics of the segment. Therefore, a segment is represented as (f_{sg}, p), where f_{sg} is the latent code and p indicates the validity of this segment. The maximum segments N_{sg} of a stroke can be determined by datasets. If a stroke contains less than N_{sg} segments, we use another token to pad the corresponding segment sequence.

At the sketch level, a sketch is a sequence of strokes. Similarly, we adopt the neural representation of strokes. Thus, a stroke is represented as (f_{st}, p), where f_{st} is the stroke's latent code estimated by our method and p indicates this stroke's validity. The maximum strokes N_{st} of a sketch can be determined by datasets. If a sketch contains less than N_{st} strokes, we also use a token to pad the corresponding stroke sequence.

3.2 Hierarchical transformer architecture

As illustrated in Figure 2, our network architecture consists of three levels, corresponding to the hierarchical structure of sketches. We introduce three pairs of transformer-based encoders and decoders, *i.e.*, segment encoder/decoder, stroke encoder/decoder, and sketch encoder/decoder. Given a sketch with a hierarchical structure, we can embed this sketch into a latent space in a bottom-up manner. First, we obtain the neural representations of the segments with the segment encoder. With the latent codes of the segments, we can obtain the neural representations of the strokes with the stroke encoder. Finally, the neural representation of the whole sketch can be estimated by the sketch encoder. The decoders can reconstruct the sketch by feeding the obtained sketch code to the decoders, in a top-down way. Below we describe the encoders/decoders in detail.

Segment encoder/decoder. The segment encoder takes as input a segment, which consists of a sequence of points. The first layer of this encoder is an embedding layer, which includes a dense layer to convert a point $(\varDelta x, \varDelta y, p)$ into a feature vector and an addition of the positional code [23]. After this layer, a point is converted into an embedding feature $f_{\mathrm{pt}} \in \mathbb{R}^{d^{\mathrm{f}}_{\mathrm{pt}}}$. Then a segment can be represented as $\mathbf{F}_{\mathrm{sg}} \in \mathbb{R}^{N_{\mathrm{pt}} \times d^{\mathrm{f}}_{\mathrm{pt}}}$, where N_{pt} is the maximum number of points in a segment. This layer is followed by L_{sg} basic MHA (multi-head attention) layers [23]. After these blocks, we obtain the hidden feature of this segment $\mathbf{F}'_{\mathrm{sg}} \in \mathbb{R}^{N_{\mathrm{pt}} \times d^{\mathrm{f}}_{\mathrm{pt}}}$. To obtain a latent code of this segment, we add an additional layer to aggregate the hidden features as Sketchformer [18] does. Finally, this segment encoder produce a segment code $z_{\mathrm{sg}} \in \mathbb{R}^{d^{\mathrm{z}}_{\mathrm{sg}}}$.

The segment decoder is the inverse of the segment decoder. Given a segment code z_{sg}, we feed it to a dense expansion layer, which includes a dense layer and an addition of positional codes, to obtain the hidden feature of the corresponding segment $\hat{\mathbf{F}}_{\mathrm{sg}} \in \mathbb{R}^{N_{\mathrm{pt}} \times d^{\mathrm{f}}_{\mathrm{pt}}}$. This hidden feature is then fed to L_{sg} basic MHA layers to obtain a updated hidden feature $\hat{\mathbf{F}}'_{\mathrm{sg}} \in \mathbb{R}^{N_{\mathrm{pt}} \times d^{\mathrm{f}}_{\mathrm{pt}}}$. To reconstruct the segment, we append a dense layer to produce the points in this segment in an autoregressive way.

Stroke encoder/decoder. The stroke encoder has a similar architecture to the segment encoder. It takes as input a stroke, which consists of a sequence of segments. We adopt the neural representations of the segments, thus a segment is in the form of (z_{sg}, p). After a similar embedding layer, a segment is converted into an embedding feature $f_{\mathrm{sg}} \in \mathbb{R}^{d^{\mathrm{f}}_{\mathrm{sg}}}$ and the stroke can be represented as $\mathbf{F}_{\mathrm{st}} \in \mathbb{R}^{N_{\mathrm{sg}} \times d^{\mathrm{f}}_{\mathrm{sg}}}$, where N_{sg} is the maximum number of segments in a stroke. After L_{st} basic MHA layers and feature aggregation layer, we can obtain the neural representation of the stroke as $z_{\mathrm{st}} \in \mathbb{R}^{d^{\mathrm{z}}_{\mathrm{st}}}$. The stroke decoder is similar to the segment decoder. It consists of a dense expansion layer, L_{st} MHA layers, and a dense layer. It consumes a stroke code and produces a list of segment codes in an autoregressive way.

Sketch encoder/decoder. The sketch encoder and decoder are similar to the stroke encoder and decoder in terms of architecture and encoding/decoding procedure. Given a sketch consisting of a sequence of strokes, each of which is in the form of (z_{st}, p), an embedding layer converts each stroke into an embedding feature $f_{\mathrm{st}} \in \mathbb{R}^{d^{\mathrm{f}}_{\mathrm{st}}}$, and the sketch can be represented as $\mathbf{F}_{\mathrm{sk}} \in \mathbb{R}^{N_{\mathrm{st}} \times d^{\mathrm{f}}_{\mathrm{st}}}$, where N_{st} is the maximum number of strokes in a sketch. The sketch encoder produces a sketch code $z_{\mathrm{sk}} \in \mathbb{R}^{d^{\mathrm{z}}_{\mathrm{sk}}}$. The sketch decoder consumes this sketch code and produces a list of stroke codes. There are L_{sk} MHA layers in the sketch encoder and decoder.

3.3 Training

We adopt the reconstruction task for the network training. To improve the training efficiency, the three pairs of encoders and decoders are trained separately. We first train the segment encoder and decoder. After this training, we use the segment encoder to obtain the neural representations of all segments. The obtained segment codes are then used for the training of the stroke encoder and decoder. In the end, the sketch encoder and decoder can be trained with the stroke codes.

Loss. For each pair of encoder and decoder, we adopt L_1 and L_2 loss for the training. Since the decoders produce the points, segment codes, and stroke codes in an autoregressive way, we also include a binary cross-entropy loss for the prediction of the EOS (end of sequence) symbol. Therefore, the loss function is $L = \alpha L_1 + \beta L_2 + \gamma L_{\mathrm{CE}}$, where α, β, and γ are the weights for these losses. For different pairs of encoders and decoders, the computations of the loss are slightly different. For the segment encoder/decoder, the loss is computed based on the positions of points. For the stroke and sketch encoders/decoders, the losses are computed based on the segment and stroke codes, respectively.

4 Experiments

We investigate the effectiveness of our method according to three tasks, *i.e.*, sketch reconstruction, sketch recognition, and sketch semantic segmentation. We first introduce the datasets for the experiments, then describe the implementation details, and finally demonstrate the results of the experiments.

Datasets. We train our model on the QucikDraw dataset [8]. It contains 50 million sketches with 345 categories. Each category consists of 70,000 training samples, 2,500 validation samples, and 2,500 testing samples. For each category, we randomly selected 700 sketches from the training samples as our training set, and 100 sketches from the validation samples as our validation set. In total, the training set has 241,500 sketches and the validation set has 34,500 sketches. Similar to existing works [18, 32, 36], we adopt the Ramer-Douglas Peucker (RDP) algorithm [5] to simplify the sketches. In the sketch reconstruction and sketch recognition tasks, we evaluate our method on the QucikDraw dataset. In the sketch semantic segmentation task, we test our method on the SPG [15] and the SketchSeg-150K [27] datasets, both of which are constructed based on the QuickDraw dataset.

Hyperparameters	Value
N_{pt}: maximum points in a segment	20
N_{sg}: maximum segments in a stroke	20
N_{st}: maximum strokes in a sketch	20
$d^{\mathrm{f}}_{\mathrm{pt}}$: dimension of point embedding feature	128
$d^{\mathrm{f}}_{\mathrm{sg}}$: dimension of segment embedding feature	128
$d^{\mathrm{f}}_{\mathrm{st}}$: dimension of stroke embedding feature	256
$d^{\mathrm{f}}_{\mathrm{sk}}$: dimension of sketch embedding feature	512
L_{sg}: number of MHA layers in segment level	2
L_{st}: number of MHA layers in stroke level	3
L_{sk}: number of MHA layers in sketch level	4
H: number of attention heads in MHA layers	8
dropout	0.1
optimizer	Adam
scheduler	linear warmup and decay
learning rate	0.0001

Table 1. The hyperparameters in our model.

Implementation details. As shown in Figure 2, our model consists of three pairs of transformer-based encoders and decoders. In the segment encoder and decoder, the maximum number of points in a segment N_{pt} is 20; the dimension of the point embedding feature $d^{\mathrm{f}}_{\mathrm{pt}}$ is 128; the number of MHA (multi-head attention) layers L_{sg} is 2; the dimension of the produced segment code $d^{\mathrm{z}}_{\mathrm{sg}}$ is 128. In the stroke encoder and decoder, the maximum number of segments in a stroke N_{sg} is 20; the dimension of the segment embedding feature $d^{\mathrm{f}}_{\mathrm{sg}}$ is 256; the number of MHA layers L_{st} is 3; the dimension of the produced stroke code $d^{\mathrm{z}}_{\mathrm{st}}$ is 256. In the sketch encoder and decoder, the maximum number of strokes in a sketch N_{st} is 20; the dimension of the stroke embedding feature $d^{\mathrm{f}}_{\mathrm{st}}$ is 512; the number of MHA layers L_{sk} is 4; the dimension of the produced sketch code $d^{\mathrm{z}}_{\mathrm{sk}}$ is 512. In all the MHA layers, the number of attention heads H is 8. For the loss functions, for all pairs of encoders and decoders, we set α as 0.3, β as 0.7, and γ as 1. The learning rate is 0.0001 initially and is adjusted with a linear warmup and decay. These hyperparameters are also listed in Table 1.

4.1 Sketch reconstruction

The sketch reconstruction is achieved by first obtaining a latent code of a sketch with the encoder and then reconstructing the sketch from the code with the decoder. The reconstruction quality implies whether the latent code is an appropriate representation of the sketch. In this task, we compare our method with SketchRNN [8] and Sketchformer [18], both of which have encoders to convert a sketch into a latent code. SketchRNN and Sketchformer treat a sketch as a sequence of points. They use RNN and Transformer as their network architecture. The experiment is conducted on the QuickDraw dataset. We use the pre-trained models of SketchRNN and Sketchformer for this experiment. SketchRNN provides six pre-trained models, each of which was trained with sketches in one single category. The selected categories include owl, sheep, cat,

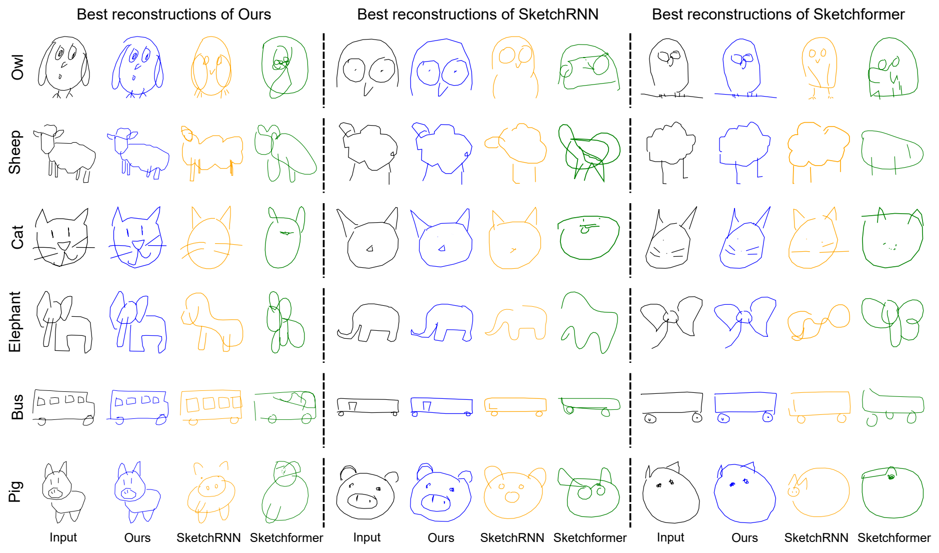

Best reconstructions of Ours
Best reconstructions of SketchRNN
Best reconstructions of Sketchformer
Owl
Sheep
Cat
Elephant
Bus
Pig
Input
Ours
SketchRNN
Sketchformer

Fig. 3. The qualitative results of the sketch reconstruction. We select six categories in the Quick-Draw dataset for evaluation. For each category, we select three sketches, each of which is the best reconstruction of a compared method across all samples. The best reconstruction is determined by the *CD* score. Our method achieves the best sketch reconstruction, indicating that the neural representation estimated by our method better retains the semantic information of sketches.

elephant, bus, and pig. Sketchformer provides one single pre-trained model trained with sketches in all categories. Similar to Sketchformer, we provide one single model trained with all sketches. The training data of our model and Sketchformer's pre-trained model are the same.

Evaluation metrics. We adopt the Chamfer distance (*CD*) [7, 29], the earth mover's distance (*EMD*) [19], and the CLIP similarity (*CLIPSim*) [17, 24] as the evaluation metrics. *CD* and *EMD* measure the geometry similarity between the input sketch and the reconstructed sketch, and *CLIPSim* reflects the semantic similarity between them. To compute the *CLIPSim* score between two sketches, we first render them as images and then use the equation introduced in [24] for the computation.

Results. Since SketchRNN's pre-trained models can only handle six categories, *i.e.*, owl, sheep, cat, elephant, bus, and pig, we only select these categories for evaluation. We randomly select 500 sketches from the testing samples for reconstruction in each category. Table 2 shows the quantitative comparison between the compared methods. According to the evaluation metrics, our method achieves the best performance on sketch reconstruction. Figure 3 shows some representative reconstruction results obtained by the compared methods. For each category, we select three sketches, each of which is the best reconstruction of a compared method across all samples. The best reconstruction is determined by the *CD* score. This qualitative comparison also indicates that our method achieves the best sketch reconstruction. From these quantitative and

Method	*CD* ↓	*EMD* ↓	*CLIPSim* ↓
SketchRNN [8]	0.001573	0.0331	0.1347
Sketchformer [18]	0.001116	0.0651	0.1172
Ours w/o segment level	0.000742	0.0293	0.1025
Ours w/o stroke level	0.001482	0.0864	0.1413
Ours	**0.000461**	**0.0154**	**0.0717**

Table 2. Quantitative comparisons on the sketch reconstruction task between the compared methods. In all metrics, our method achieves the best performance, indicating that the neural representation estimated by our method better retains the semantic information of sketches.

qualitative comparisons, we conclude that the neural representation estimated by our method better retains the semantic information of sketches.

4.2 Sketch recognition

Our method estimates the neural representation of sketches. This representation is used for sketch recognition. To accomplish this task, we append a dense layer and a softmax layer to the sketch encoder, fix the parameters of our model, and train the parameters of the appended layers. We compare our method with a list of existing works, including Sketch-a-Net [33], SketchRNN [8], Sketchformer [18], S^3Net [31]. Sketch-a-Net is a CNN-based method for sketch recognition. SketchRNN and Sketchformer treat sketches as a sequence of points, using RNN and Transformer as the network architectures. S^3Net is the state of the art in the task of sketch recognition. It also considers the hierarchical structure of sketches but adopts RNN for extracting segment-level features and GNN for sketch-level features. The experiment is conducted on the QuickDraw dataset. We select 7,000 sketches from the training samples in each category for the training of the appended layers, all 2,500 sketches from the validation samples for the validation, and all 2,500 sketches from the testing samples for the computation of the recognition accuracy.

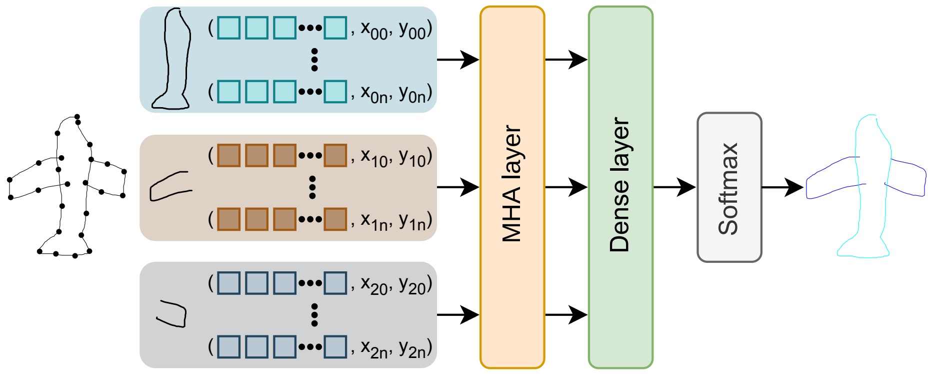
Results. Table 3 shows the recognition accuracy of the compared methods. Our method achieves the best accuracy among all methods. The recognition accuracy of Sketch-a-Net is low, implying that CNN-based methods are less appropriate for extracting the semantics of sketches. The performance of SketchRNN is not satisfactory. Although it considers the drawing order of sketches, the limitation of RNN architecture makes it less effective for long sequences. Sketchformer achieves a better result, indicating the advantage of Transformer architecture. S^3Net achieves state-of-the-art performance, demonstrating the necessity of considering the hierarchical structure of sketches. Our method also recognizes the importance of this hierarchical structure and adopts a powerful transformer-based architecture, therefore reaching the highest recognition accuracy.

4.3 Sketch semantic segmentation

Sketch semantic segmentation can be achieved with the stroke's neural representation estimated by our stroke encoder. We design a lightweight transformer-based network

Method	(Acc %)
Sketch-a-Net [33]	68.71
SketchRNN [8]	67.69
Sketchformer [18]	77.68
S^3Net [31]	84.22
S^3Net (Stroke-5)	85.10
Ours w/o segment level	80.63
Ours w/o stroke level	78.82
ours	**85.73**

Table 3. Quantitative comparisons on the sketch recognition task between the compared methods. Our method outperforms the state-of-the-art methods, demonstrating the effectiveness of our network architecture.



$(\square\square\square\cdots\square, x_{00}, y_{00})$
$(\square\square\square\cdots\square, x_{0n}, y_{0n})$
$(\square\square\square\cdots\square, x_{10}, y_{10})$
$(\square\square\square\cdots\square, x_{1n}, y_{1n})$
$(\square\square\square\cdots\square, x_{20}, y_{20})$
$(\square\square\square\cdots\square, x_{2n}, y_{2n})$
MHA layer
Dense layer
Softmax

Fig. 4. Our designed lightweight segmentation network in the sketch semantic segmentation task. It exploits a simple transformer architecture. With the aid of neural representation estimated by our method, this network achieves state-of-the-art performance.

for this task (see Figure 4). This segmentation network contains 4 basic MHA layers, a dense layer, and a softmax layer. Given a sketch, we first extract the stroke codes of its strokes. For each point in this sketch, we form a feature vector by concatenating the neural code of the stroke that contains this point and the absolution position of this point. This leads to a sequence of feature vectors. We then feed this sequence to the segmentation network. We evaluate our method on this task with the SPG dataset [15] and the SketchSeg-150K dataset [27]. Both of these datasets are constructed upon the QuickDraw dataset. The SPG dataset consists of 25 categories, each of which contains 800 sketches. We split the sketches in each category in a ratio of 13:2:1 for training, testing, and validation. The SketchSeg-150K dataset consists of 20 categories, each of which contains 7,500 sketches. The splitting ratio of this dataset is the same as the SPG dataset.

Many works have been proposed for the sketch semantic segmentation task, *e.g.*, SketchSeg [27], SPGSeg [14], DeepLabv3+ [3], FastSeg [15], CBL [22], SketchGNN [32], and Sketch-Segformer [36]. Among these methods, SketchGNN and Sketch-Segformer are the state of the arts and achieve the best performance on this task. We, therefore,

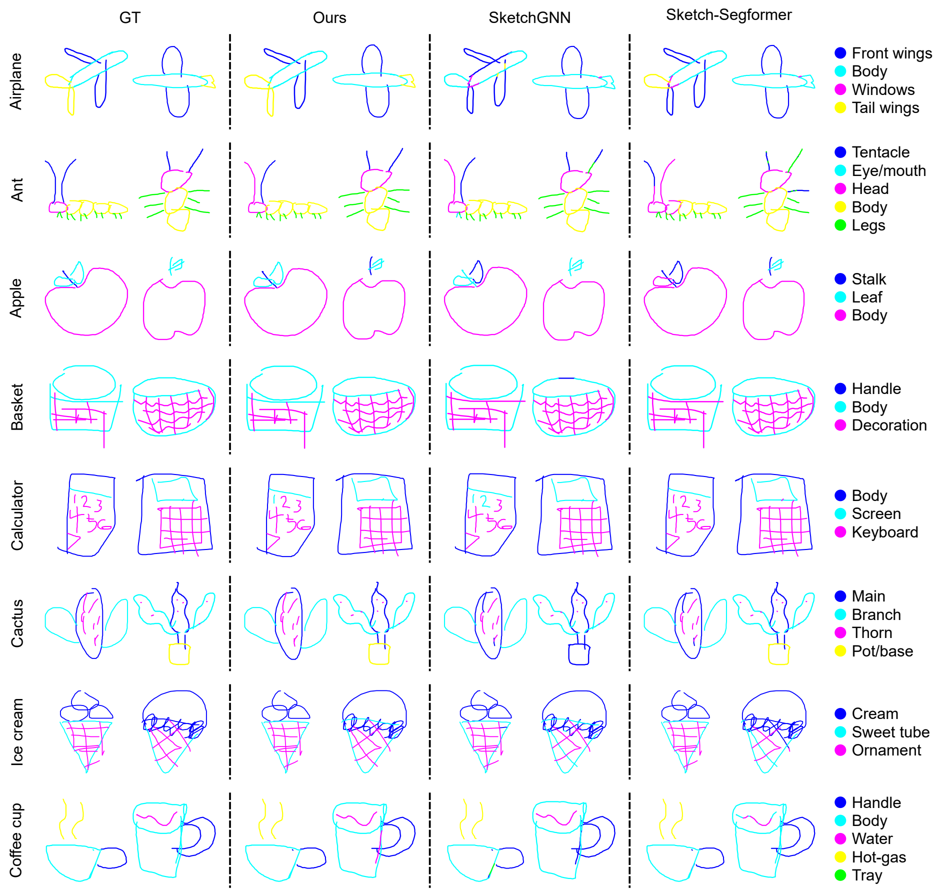


Fig. 5. The qualitative results of the sketch semantic segmentation. In general, our method reaches the state-of-the-art methods in this task. For some categories, our method achieves better performance.

compare our method with SketchGNN and Sketch-Segformer to show the effectiveness of the sketch's neural representation estimated by our method.

Evaluation metrics. We use the widely adopted pixel metric (*P-metric*) and component metric (*C-metric*) for evaluation. Given a sketch and its corresponding segmentation, *P-metric* is defined as the percentage of points that are predicted with the correct semantic labels; *C-metric* is defined as the percentage of strokes that are predicted with the correct semantic labels. In our experiment, we consider a stroke to be correctly labeled if more than 75% of the points in this stroke are assigned the correct label.

Results. Table 4 and 5 show the comparison results on the SPG and SketchSeg-150K datasets respectively. Figure 5 shows the visual segmentation results produced by the compared methods. For the SPG dataset, according to *P-metric* and *C-metric*, our method

Category	SketchGNN		Sketch-Segformer		Ours		Ours w/o segment level		Ours w/o stroke level	
	P-metric	*C-metric*	*P-metric*	*C-metric*	*P-metric*	*C-metric*	*P-metric*	*C-metric*	*P-metric*	*C-metric*
Airplane	96.9	92.9	96.7	92.0	**97.1**	**93.7**	95.2	90.9	77.5	66.1
Alarm clock	98.0	95.7	**98.4**	95.3	98.2	**96.0**	95.9	92.3	85.9	74.9
Ambulance	**94.1**	**90.1**	92.6	88.6	93.8	89.9	91.0	85.6	79.4	67.6
Ant	93.5	91.9	93.9	92.3	**94.5**	**92.5**	93.1	91.7	78.1	76.2
Apple	97.1	91.8	97.3	**93.1**	**97.4**	93.0	95.4	87.8	90.9	77.8
Backpack	92.4	85.9	**93.2**	**86.5**	92.6	85.7	90.1	82.0	75.7	58.5
Basket	97.5	97.0	96.9	96.7	**97.9**	**97.6**	95.7	95.3	88.1	85.6
Butterfly	**99.4**	**97.3**	98.8	96.5	98.6	96.7	98.0	95.7	87.9	79.6
Cactus	97.2	94.5	96.5	95.3	**97.4**	**96.2**	96.3	95.6	87.6	84.1
Calculator	99.1	98.2	99.2	98.0	**99.4**	**98.5**	99.2	98.2	94.8	89.5
Campfire	96.9	95.6	**97.5**	96.1	97.0	**96.3**	95.0	91.5	89.5	87.2
Candle	99.2	**98.3**	**99.3**	98.2	98.7	96.7	98.2	95.8	92.1	83.8
Coffee cup	98.9	97.1	99.0	98.3	**99.3**	**98.4**	97.3	95.2	91.0	81.2
Crab	95.6	93.6	**96.5**	**93.7**	96.2	93.4	94.5	91.2	77.8	67.8
Duck	97.8	96.1	**98.1**	**96.8**	97.5	95.9	96.8	94.0	85.6	76.8
Face	**98.4**	**97.3**	98.2	97.1	98.0	**97.3**	96.7	94.5	89.5	75.3
Ice cream	94.7	95.1	96.9	95.4	**97.8**	**96.2**	92.2	90.9	88.7	80.5
Pig	**98.9**	97.7	**98.9**	97.6	98.3	**98.0**	96.1	97.4	83.4	73.0
Pineapple	**98.7**	95.2	98.6	**96.3**	98.2	96.1	93.2	94.9	95.8	89.4
Suitcase	**99.6**	**98.0**	99.5	97.9	99.2	97.4	97.2	94.0	92.3	82.4
Average	97.3	**95.9**	97.3	95.1	**97.4**	95.3	95.7	92.7	86.7	77.9

Table 4. Quantitative comparison on the sketch semantic segmentation task between the compared methods. The compared methods are tested on the SPG dataset. Our method achieves state-of-the-art performance. The rightmost four columns list the results of our method by removing the segment and stroke level structures respectively. In these two settings, the hierarchy of sketches is reduced from three to two.

is comparable to these two state-of-the-art methods. Each of the compared methods has advantages for certain categories. On average, our method is slightly better, but the lead is not significant. For the SketchSeg-150K dataset, the performance of the compared methods is even closer. We thus conclude that our method has been one of the state-of-the-art methods in the sketch semantic segmentation task. It is worth noting that, sketch semantic segmentation is only an application of our methods. In contrast, the other two methods are specifically devised for this task. In addition, our method only exploits the very basic transformer blocks for estimating the neural representation of sketches. It is expected that the performance of our method could be further improved if more sophisticated learning techniques are adopted.

4.4 Ablation study

Our method only adopts the basic transformer blocks and does not take advantage of other advanced learning techniques. The loss function is also simple. The novelty of our method lies in the adoption of the hierarchical structure of sketches and the associated encoding/decoding procedure. To test the necessity of our design, we conduct the following ablation study. We remove the segment and stroke level structure respectively and reduce the hierarchy of sketches from three to two. Further reducing the hierarchy would lead to the architecture of Sketchformer.

We conduct the ablation study on all the tasks, including sketch reconstruction, sketch recognition, and sketch semantic segmentation. The quantitative results can be

Category	SketchGNN		Sketch-Segformer		Ours	
	P-metric	*C-metric*	*P-metric*	*C-metric*	*P-metric*	*C-metric*
Angel	98.6	97.1	**99.4**	**99.1**	98.3	98.5
bird	98.8	98.1	**99.5**	**99.0**	99.2	98.4
Bowtie	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
Butterfly	99.8	99.8	**100.0**	**100.0**	**100.0**	**100.0**
Candle	**99.2**	**98.1**	99.1	97.5	98.6	98.0
Cup	98.0	98.0	**98.0**	**98.1**	**98.0**	**98.1**
Door	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
Dumbbell	99.9	99.9	**100.0**	**100.0**	99.9	99.9
Envelope	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
Face	99.2	97.6	**99.3**	**97.7**	98.3	97.6
Ice	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
Lamp	95.6	95.6	**97.0**	**96.5**	96.2	95.6
Lighter	99.9	99.9	**99.7**	**99.9**	**99.7**	99.8
Marker	98.9	98.9	99.1	98.9	**99.3**	**99.1**
Mushroom	99.5	97.9	**99.7**	**98.6**	99.1	98.3
Pear	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
Plane	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
Spoon	87.3	87.3	88.5	88.4	**92.0**	**89.1**
Traffic	95.7	97.1	96.1	97.3	**98.6**	**97.6**
Van	99.5	99.4	**99.6**	**99.6**	99.1	99.1
Average	98.5	98.3	**98.8**	**98.5**	**98.8**	**98.5**

Table 5. Quantitative comparison on the sketch semantic segmentation task between the compared methods. The compared methods are tested on the SketchSeg-150K dataset. Our method achieves state-of-the-art performance.

found in Table 2, 3, and 4. These results confirm the necessity and effectiveness of our three-level hierarchical transformer architecture.

5 Conclusion

In this work, we have presented *Sketchformer++*, a novel transformer-based neural network that estimates the neural representation of vector sketches. It exploits the inherent hierarchical structure of sketches, adopts three attention mechanisms, and aggregates the semantics of sketches from local to global. It consists of three pairs of encoders and decoders, which help extract the neural representation of sketches at different levels. The estimated neural representation of sketches facilitates several downstream tasks, including sketch reconstruction, sketch recognition, and sketch semantic segmentation. The experiments have shown that our method reaches or outperforms the state of the art in these tasks, indicating the effectiveness of the neural representation estimated by our method.

Our method has some limitations. First, our method requires the hierarchical structure of sketches, therefore it can not be directly applied to raster sketches. A preprocessing for sketch vectorization is necessary in this case. Second, our method may fail

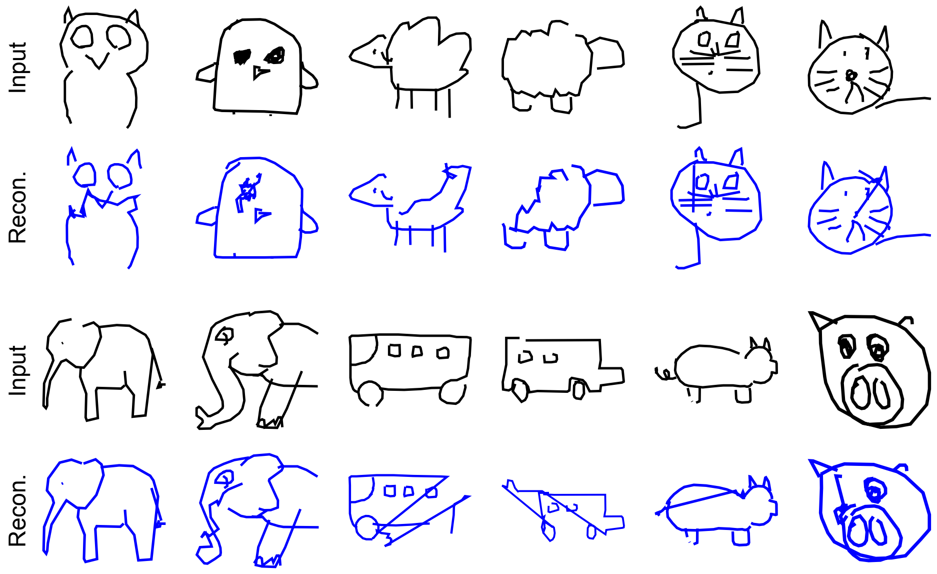


Fig. 6. Our method may fail to retain all the semantics of a sketch, thus the reconstruction may contain artifacts.

to retain all the semantics of a sketch. For example, the reconstruction may contain artifacts as shown in Figure 6, although most reconstructions are of high quality. Third, when accomplishing certain tasks, *e.g.*, sketch recognition on a specific dataset, our method requires two rounds of training, thus is less efficient than the methods specifically designed for this task. To avoid training our model on every dataset, it is possible to train a general model by increasing the parameters of our model and feeding sufficient sketches to the model.

Our method may promote many research directions. By aligning the constructed latent space of vector sketches with a latent space of raster sketches, it would be possible to achieve sketch vectorization. With the aid of CLIP, it would be able to convert texture images to sketches. It would be also interesting to extend our method to other applications, such as sketch healing or sketch completion.

Acknowledgments. This work was supported in parts by NSFC (62072316, 62172363, U21B2023), NSF of Guangdong Province (2023A1515011297), DEGP Innovation Team (2022KCXTD025), Shenzhen Science and Technology Program (KQTD20210811090044003), and Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Bhunia, A.K., Chowdhury, P.N., Yang, Y., Hospedales, T.M., Xiang, T., Song, Y.Z.: Vectorization and rasterization: Self-supervised learning for sketch and handwriting. In: Pro-

ceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5672–5681 (2021)

2. Cao, Y., Wang, H., Wang, C., Li, Z., Zhang, L., Zhang, L.: Mindfinder: interactive sketch-based image search on millions of images. In: Proceedings of the 18th ACM international conference on Multimedia. pp. 1605–1608 (2010)
3. Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-decoder with atrous separable convolution for semantic image segmentation. In: Proceedings of the European conference on computer vision (ECCV). pp. 801–818 (2018)
4. Cheng, S., Ren, Y., Yang, Y.: Ssr-gnns: Stroke-based sketch representation with graph neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5131–5141 (2022)
5. Douglas, D.H., Peucker, T.K.: Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. Cartographica: the international journal for geographic information and geovisualization **10**(2), 112–122 (1973)
6. Dutta, A., Akata, Z.: Semantically tied paired cycle consistency for zero-shot sketch-based image retrieval. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5089–5098 (2019)
7. Fan, H., Su, H., Guibas, L.J.: A point set generation network for 3d object reconstruction from a single image. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 605–613 (2017)
8. Ha, D., Eck, D.: A neural representation of sketch drawings. arXiv preprint arXiv:1704.03477 (2017)
9. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
10. Hu, R., Collomosse, J.: A performance evaluation of gradient field hog descriptor for sketch based image retrieval. Computer Vision and Image Understanding **117**(7), 790–806 (2013)
11. Klare, B., Jain, A.K.: Sketch-to-photo matching: a feature-based approach. In: Biometric technology for human identification VII. vol. 7667, pp. 11–20. SPIE (2010)
12. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)
13. Lee, J., Kim, E., Lee, Y., Kim, D., Chang, J., Choo, J.: Reference-based sketch image colorization using augmented-self reference and dense semantic correspondence. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 5801–5810 (2020)
14. Li, K., Pang, K., Song, Y.Z., Xiang, T., Hospedales, T.M., Zhang, H.: Toward deep universal sketch perceptual grouper. IEEE Transactions on Image Processing **28**(7), 3219–3231 (2019)
15. Li, L., Fu, H., Tai, C.L.: Fast sketch segmentation and labeling with deep learning. IEEE computer graphics and applications **39**(2), 38–51 (2018)
16. Qi, Y., Su, G., Chowdhury, P.N., Li, M., Song, Y.Z.: Sketchlattice: Latticed representation for sketch manipulation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 953–961 (2021)
17. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: International conference on machine learning. pp. 8748–8763. PMLR (2021)
18. Ribeiro, L.S.F., Bui, T., Collomosse, J., Ponti, M.: Sketchformer: Transformer-based representation for sketched structure. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 14153–14162 (2020)
19. Rubner, Y., Tomasi, C., Guibas, L.J.: The earth mover's distance as a metric for image retrieval. International journal of computer vision **40**, 99–121 (2000)
20. Sangkloy, P., Burnell, N., Ham, C., Hays, J.: The sketchy database: learning to retrieve badly drawn bunnies. ACM Transactions on Graphics (TOG) **35**(4), 1–12 (2016)

21. Song, J., Pang, K., Song, Y.Z., Xiang, T., Hospedales, T.M.: Learning to sketch with shortcut cycle consistency. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 801–810 (2018)
22. Tang, L., Zhan, Y., Chen, Z., Yu, B., Tao, D.: Contrastive boundary learning for point cloud segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8489–8499 (2022)
23. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
24. Vinker, Y., Pajouheshgar, E., Bo, J.Y., Bachmann, R.C., Bermano, A.H., Cohen-Or, D., Zamir, A., Shamir, A.: Clipasso: Semantically-aware object sketching. ACM Transactions on Graphics (TOG) **41**(4), 1–11 (2022)
25. Wang, F., Lin, S., Li, H., Wu, H., Cai, T., Luo, X., Wang, R.: Multi-column point-cnn for sketch segmentation. Neurocomputing **392**, 50–59 (2020)
26. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph cnn for learning on point clouds. ACM Transactions on Graphics (tog) **38**(5), 1–12 (2019)
27. Wu, X., Qi, Y., Liu, J., Yang, J.: Sketchsegnet: A rnn model for labeling sketch strokes. In: 2018 IEEE 28th International Workshop on Machine Learning for Signal Processing (MLSP). pp. 1–6. IEEE (2018)
28. Xu, P., Joshi, C.K., Bresson, X.: Multigraph transformer for free-hand sketch recognition. IEEE Transactions on Neural Networks and Learning Systems **33**(10), 5150–5161 (2021)
29. Yan, C., Vanderhaeghe, D., Gingold, Y.: A benchmark for rough sketch cleanup. ACM Transactions on Graphics (TOG) **39**(6), 1–14 (2020)
30. Yang, B., Chen, X., Hong, R., Chen, Z., Li, Y., Zha, Z.J.: Joint sketch-attribute learning for fine-grained face synthesis. In: MultiMedia Modeling: 26th International Conference, MMM 2020, Daejeon, South Korea, January 5–8, 2020, Proceedings, Part I 26. pp. 790–801. Springer (2020)
31. Yang, L., Sain, A., Li, L., Qi, Y., Zhang, H., Song, Y.Z.: S3net: Graph representational network for sketch recognition. In: 2020 IEEE International Conference on Multimedia and Expo (ICME). pp. 1–6. IEEE (2020)
32. Yang, L., Zhuang, J., Fu, H., Wei, X., Zhou, K., Zheng, Y.: Sketchgnn: Semantic sketch segmentation with graph neural networks. ACM Transactions on Graphics (TOG) **40**(3), 1–13 (2021)
33. Yu, Q., Yang, Y., Liu, F., Song, Y.Z., Xiang, T., Hospedales, T.M.: Sketch-a-net: A deep neural network that beats humans. International journal of computer vision **122**, 411–425 (2017)
34. Zang, S., Tu, S., Xu, L.: Linking sketch patches by learning synonymous proximity for graphic sketch representation. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 37, pp. 11096–11103 (2023)
35. Zheng, Y., Xie, J., Sain, A., Ma, Z., Song, Y.Z., Guo, J.: Ende-gnn: An encoder-decoder gnn framework for sketch semantic segmentation. In: 2022 IEEE International Conference on Visual Communications and Image Processing (VCIP). pp. 1–5. IEEE (2022)
36. Zheng, Y., Xie, J., Sain, A., Song, Y.Z., Ma, Z.: Sketch-segformer: Transformer-based segmentation for figurative and creative sketches. IEEE Transactions on Image Processing (2023)
37. Zhu, X., Xiao, Y., Zheng, Y.: 2d freehand sketch labeling using cnn and crf. Multimedia Tools and Applications **79**(1-2), 1585–1602 (2020)
38. Zhu, X., Yuan, J., Xiao, Y., Zheng, Y., Qin, Z.: Stroke classification for sketch segmentation by fine-tuning a developmental vggnet16. Multimedia Tools and Applications **79**, 33891–33906 (2020)