S^{3}NET: GRAPH REPRESENTATIONAL NETWORK FOR SKETCH RECOGNITION

Lan Yang[1,2], *Aneeshan Sain*[2], *Linpeng Li*[1], *Yonggang Qi*[1], *Honggang Zhang*[1], *Yi-Zhe Song*[2]

[1]Beijing University of Posts and Telecommunications, China
[2]SketchX, CVSSP, UniversityofSurrey, United Kingdom
{ylan, lilinpeng, qiyg, zhhg}@bupt.edu.cn; {a.sain, y.song}@surrey.ac.uk

ABSTRACT

Sketches are distinctly different to photos. They are highly abstract and exhibit a severe lack of visual cues. Prior works have therefore explored additional traits unique to sketches to help recognition such as stroke ordering. In this paper, we pioneer in studying the role of structure in sketches, for the task of sketch recognition. In particular, we propose a novel graph representation specifically designed for sketches, which follows the inherent hierarchical relationship ("**s**egment-**s**troke-**s**ketch") of sketching elements. By conforming to this hierarchy, we also introduce a joint network that encapsulates both the *structural* and *temporal* traits of sketches for sketch recognition, termed *S*^{3}*Net*. *S*^{3}*Net* employs a recurrent neural network (RNN) to extract segment-level features, followed by a graph convolutional network (GCN) to aggregate them into sketch-level features. The RNN first encodes *temporal* cues in sketches while its outputs are used as node embedding to construct a hierarchical sketch-graph. The GCN module then takes in this sketch-graph to produce a *structure-aware* embedding for sketches. Extensive experiments on the *QuickDraw* dataset, exhibit superior performance over state-of-the-arts, surpassing them by over 4%. Ablative studies further demonstrate the effectiveness of the proposed structural graph for both inter-class, and intra-class feature discrimination. Code is available at: `https://github.com/yanglan0225/s3net`.

Index Terms— Sketch Recognition, Graph Convolutional Network

1. INTRODUCTION

Recent advances in touchscreen devices have made sketching a much easier task for many. Research on sketches has consequently flourished. Sketch recognition is a central task in sketch understanding, and acts as the basis for downstream tasks such as sketch synthesis [1], forensic sketch analysis [2], and sketch-based image retrieval [3, 4, 5]. Sketches are however very different to photos. The high extent of abstraction and variance in a sketch pertaining to its real-life object, makes sketch recognition quite a challenging yet appealing task.

Earlier studies had focused on designing hand-crafted feature extractors from photo feature representations. With the advent of deep learning, Yu *et al.* [3] proposed a sketch-specific convolutional neural network named Sketch-a-Net. Not only did it outperform earlier hand-crafted features significantly, but also surpassed human performance. With the release of QuickDraw [6] dataset having millions of free-hand human sketches, it was possible to approach sketch research using some data-driven approaches. Xu *et al.* [7] proposed Sketch-Mate combining RNN with conventional CNN under a dual-branch setting. Such works focus on visual cues, although some of them involve temporal cues as well.

These CNN based works ignored structural specifications of sketch. Structure plays an important role in sketch related works as, despite various appearances, structures are inherently invariant. Using the perception of structure, humans are able to recognize sketches easily.

We believe incorporating this fundamental idea of inherent structure would be highly beneficial to sketch recognition. This intuition guided us to explore the possibilities and implications of a sketch-graph. Graph is a powerful tool to represent structure related information. Quite a few works have discussed on formulating graph in computer vision, such as the graph of a point cloud[8], gesture[9], and skeleton[10]. However there is no clear standard on building a graph for each sketch, as there is neither any accepted rule for a definite node nor a clear relationship to connect them.

In this paper, we propose a method of building a sketch-graph based on some general internal relationship in a sketch. We have observed that every sketch stores an ascending hierarchical relationship "segment-stroke-sketch". Using this relationship we build a sketch-graph such that nodes can represent elements at different levels, connected to one another using rules as described later in Sec 3.3. Based on this sketch-graph, we propose a joint network comprising a RNN sub-module and a GCN sub-module connected together via a graph-building module. The RNN module encodes the stroke temporal information, whereas, the GCN module aggregates a global embedding for each sketch. Finally, an uniquely designed graph-building module bridges the two sub-networks. Motivation behind using GCN sources from the fact that graph convolutional operation can aggregate nodal features

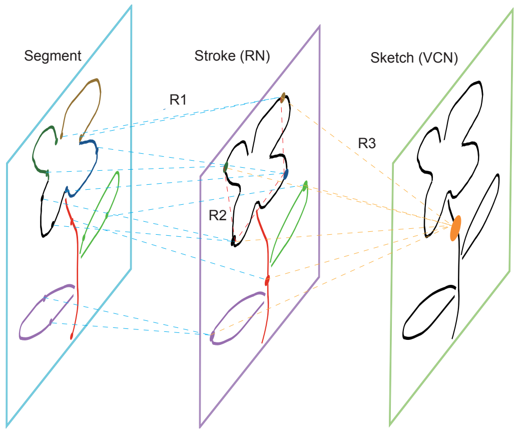

Fig. 1. **Hierarchical sketch-graph construction.** Leftmost frame shows color-coded segments obtained from original sketch after stroke-cutting. For each group, a representative node (central frame) is selected to simplify connections in a sketch-graph. Finally, a virtual central node (VCN) is added as a global embedding (rightmost frame). Edges can connect a representative node to its segment nodes (blue), or 2 representative nodes of the same stroke based on temporal order (red), or the VCN to any representative node (yellow).

from its neighbours and update accordingly to obtain a better distinguishable global embedding. We have trained S^{3}Net on a subset of QuickDraw [6] dataset having 4,312,500 sketches. Results obtained from various experiments show our model S^{3}Net to have outperformed state-of-the-arts significantly. This shows that a highly distinguishable embedding representation can be obtained by incorporating temporal and structural traits of a sketch.
To summarise, our main contributions in this work are:
(i) we propose an effective approach to build a sketch-graph implementing the segment-stroke-sketch hierarchical relationship.
(ii) Based on that, we propose a joint network which can learn temporal and structural cues simultaneously, the *S^{3}Net*.

2. RELATED WORK

Sketch Dataset Collecting free-hand sketches for performing vision tasks is quite difficult and time consuming. This is because numerous sketch samples with a high degree of variation is required to supply sufficient representation of each category. TU-Berlin [11] is one such dataset that hosts a collection of 20,000 human sketches, consisting of 250 object categories with 80 samples in each. Several datasets have been collected thereafter for various tasks [3, 12] thus promoting multiple studies on sketch. Out of all such datasets, QuickDraw [6] is the largest so far. With a capacity at 50 million, it provides 345 object categories, having more than 100,000 sketches per category, thus alleviating the lack of large scale paired sketch datasets.
Sketch Recognition Sketch recognition is a fundamental problem in sketch related studies. One approach towards solving it is to focus on the combination of hand-crafted features and classifiers [11, 13, 14] in a fashion similar to traditional image recognition. On the other hand, Sketch-a-Net [3] employed convolutional neural network (CNNs) for sketch recognition using the TU-Berlin [11] dataset, thus introducing deep learning in this sphere of sketch. However, such works focused only on visually abstract traits of sketches instead of sequential ordering of its strokes. That issue was alleviated by involving RNNs, as in [15] where a sequence of cumulative stroke images was fed into a Gated Recurrent Unit-based architecture for sketch recognition. Recently it has been shown that stroke-level ordering information helps in learning more discriminative feature representation [7], by simultaneously encoding visual abstract and temporal traits via a two-branch late fusion network.
Graph Convolutional Network Various deep learning paradigms like CNNs, RNNs, and Variational Auto Encoders (VAEs) can effectively extract latent representation from Euclidean data. However most of such applications deal with non-euclidean data in the real world. Overcoming this issue, Graph Convolutional Networks (GCNs) provide a solution for such non-euclidean data. For instance, Thomas N *et al.* [16] trains a semi-supervised node classification network via a localized first-order approximation of spectral graph convolution. Node classification [16], link prediction [17], and graph classification [18] illustrate some more applications of GCNs.

3. METHODOLOGY

3.1. Preliminaries

To begin with, a graph can be represented as $G = (V, E)$ where $V = \{v_1, v_2, ..., v_n\}$ is the set of nodes, and $e_{ij} = (v_i, v_j) \in E$ denotes an edge. An adjacency matrix can be denoted as $A \in \{0,1\}^{n*n}$ where $A_{ij} = 1$ if $e_{ij} \in E$. Furthermore each node of the graph associates itself with a feature vector $h_v \in R^d$. Therefore, a graph holds the node feature matrix $X \in R^{n*d}$.

3.2. Sketch Format

It has been widely accepted that a sketch is represented as a sequence of strokes with each stroke consisting of a set of segments. This format owing to its sequential appearance, is suitable as inputs to a recurrent neural network, which has been adapted to model the temporal property in a sketch.
As introduced in Sketch-RNN [6], a segment can be presented in a stroke-3 format, $s_i = (\triangle x, \triangle y, p)$. The first two elements are offsets from last segment in x and y directions respectively, while $p \in \{0, 1\}$ represents the drawing state where, p=1 if s_i is the end point of a stroke, and 0 otherwise.

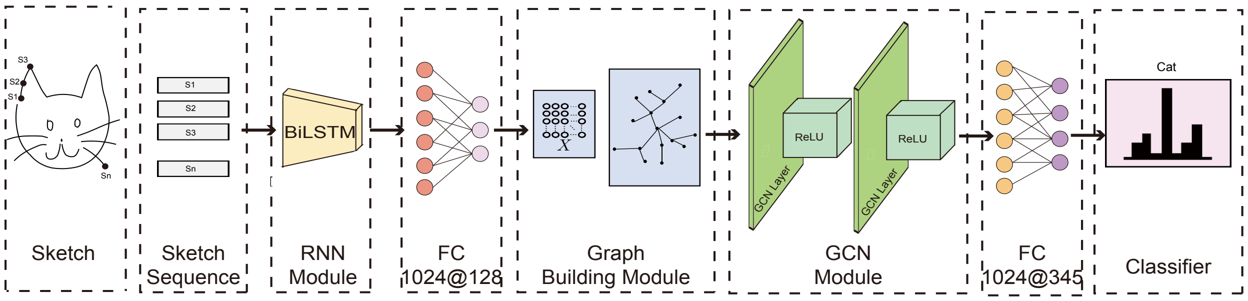

Fig. 2. **S^{3}Net**. The RNN sub-module takes in a sketch sequence and outputs a node embedding matrix via a fully-connected layer. That matrix is used to build a sketch graph which is subsequently fed into the GCN sub-module. The feature generated by GCN is used for classification via a fully-connected layer.

3.3. Graph Learning on Sketch

Building a sketch-graph poses an inevitable challenge while trying to explore structural invariance in sketch. To tackle this problem we propose an intuitive method in this paper, validating its results via experimentation.

Nodes: We treat each segment of a sketch sequence as a node in a sketch-graph G, and impose a hierarchical principle based on stroke segmentation to construct the edges. Stroke segmentation is quite meaningful for constructing a sketch-graph as too long a stroke might contain multiple semantics which is detrimental to modelling sketch structural characteristics. According to Li *et al.* [1], medium strokes have better semantics which is valuable to sketch-graph construction. Stroke length and turn-points are the two factors to be considered during stroke segmentation procedure. While turning points bring in semantic variations, histogram of stroke-lengths decide the length-cutting threshold based on each class. Long strokes are cut into shorter segments based on those two factors. Segments in a sketch sequence are divided into different groups viz, uncut short strokes and stroke segments of long strokes. For each group, a representative node is selected to simplify the connections in a sketch-graph. Finally, a virtual central node is added to the sketch-graph to get global embedding as shown in Fig. 1.

Edges: There are 3 ways of connecting two nodes via an edge. (i) By connecting a representative node to its remaining nodes from the same segment group. (ii) By chronologically connecting two representative nodes from the same stroke based on their temporal order. (iii) By joining the virtual central node (VCN) with any representative node. Hence, a pair of nodes when connected by an edge, relates to a relation corresponding to either of these three rules. Therefore, we get a hierarchical graph structure "segment-stroke-sketch" of S^{3}Net, as illustrated in Fig 1.

3.4. Network Architecture

Overview As mentioned earlier, sketches are significantly different from general color images. They are highly abstract, consisting of only a few strokes. The temporal and structural characteristics are quite typical of a sketch thus making them crucial to sketch-research. In this work, we pioneer in proposing the **RNN+GCN** architecture, over an end-to-end learning paradigm. These two sub-networks interact via a graph building module in a concatenated fashion.

Architecture As shown in Fig 2, S^{3}Net consists of three sub-modules viz, (i) A RNN encoder that inputs a sketch sequence and outputs a time-step state as a node embedding, (ii) A Graph Building Module which receives that node embedding and builds a sketch-graph as illustrated in Sec 3.3, (iii) A GCN module which inputs that sketch-graph and delivers a global embedding in the feature space.

For an input sketch, sequence $S = [s_1, s_2..., s_n]$ represents a forward sequence, where s_i is a segment. This is fed into a forward pass of BiLSTM, from which we obtain a hidden state output h. Simultaneously, a reverse sequence $S' = [s_n, s_{n-1}, ..., s_1]$ is fed as the input for a backward pass of BiLSTM and translated to a hidden output h'. These two resulting directional outputs are concatenated together as joint hidden states. We use a fully-connected layer, mapping the hidden state of each time step to 128D thus obtaining a martix $X \in R^{n\times 128}$ as the node embedding. Initializing embedding of VCN with a zero vector provides the final node embedding matrix as $X_{(n+1)\times 128}$. Therefore, the node feature matrix in a sketch-graph is given as $X \in R^{(n+1)\times 128}$, with the adjacency matrix $A \in R^{(n+1)\times(n+1)}$ being constructed as mentioned in Sec 3.3.

The "message passing" architecture, a general framework of spatial-based graph convolution, is described as follows:

$$X^{(k)} = M(A, X^{(k-1)}; \theta^{(k)}) \qquad (1)$$

where $X^{(0)}$ is a node feature matrix X, $X^{(k)}$ is the node feature computed after k GCN layers, M is the message passing function which depends on adjacency matrix A and parameters $\theta^{(k)}$. For implementation, a popular variant [16] is used, where M is implemented with a linear transformation and a ReLU readout function:

$$M(A, X^{(k-1)}; \theta^{(k)}) = ReLU(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}X^{(k-1)}W^k) \qquad (2)$$

where $\tilde{A} = A + I$, $\tilde{D} = \sum_j \tilde{A}_{ij}$ and W^k is a parameter matrix. To obtain a better global feature, we employ a differentiable pooling [19] operation. This lets us use GCN's output at layer $k-1$ to learn assigning of nodes to clusters in layer k. Following DIFFPOOL, we receive a coarse sketch-graph which contributes to learning a global feature for graph classification.

Loss Our loss function includes three terms viz, a Negative Log Likelihood Loss (NLL) $L(y)$, an Auxiliary Link Prediction L_{LP} and the Entropy of the Cluster Assignment L_E, where last two terms are the same to [19]. Formally, our final loss is defined as:

$$L = L(y) + L_{LP} + L_E \tag{3}$$

$$L(y) = -log(y) \tag{4}$$

$$L_{LP} = ||A^{(l)}, S^{(l)} {S^{(l)}}^T||_F \tag{5}$$

$$L_E = \frac{1}{n}\sum_{i=1}^{n} H(S_i) \tag{6}$$

$L(y)$ is used to minimize the distance between the outputs of model and labels distributions. L_{LP} aims to alleviate falling into spurious local minimum early in training, where $A^{(l)}$ and $S^{(l)}$ are the adjacency and assignment matrices at layer l respectively. $||\cdot||_F$ refers to the Frobenius norm, H is the entropy function, and S_i, the i^{th} row of S. L_E aims to regularize the entropy of the cluster assignment.

4. EXPERIMENTS

4.1. Datasets

Dataset splits The sketches come from Google QuickDraw dataset [6], which is by far the largest free-hand sketch dataset. It contains 345 categories contributed by players of the game "Quick, Draw!". For each category, there are 9K, 1K, 2.5K samples for training, validation, and testing respectively. Experimental dataset for S^{3}Net consists of 4,312,500 sketches overall. It is to be noted that players were given only 20 seconds to complete their art in the "Quick, Draw!" game. Therefore, the dataset contains many confused classes like "Cake" and "Birthday Cake", "Cup" and "Coffee Cup", "Hexagon" and "Octagon" etc. These similar category-pairs add on to the challenges of sketch recognition. Furthermore, experiments have been based on the original datasets without any data augmentation or pre-processing.

Implementation Details RNN subnetwork uses Bidirectional LSTM with two layers, whose input dimension is 3D, with a hidden size of 512 for each layer. Thereafter it connects to a fully-connected layer to map features into 128D. Our GCN subnetwork uses GraphSAGE [20] and DIFFPOOL [19] architectures with the chief difference of adding just one DIFFPOOL layer. S^{3}Net is implemented via PyTorch on a single Nvidia Geforce GTX 1080Ti GPU. We set initial learning rate to 0.001 and optimise using an Adam optimizer. The batch size is kept at 250, setting the dropout probability in RNN subnetwork to 0.5.

Method	Acc.
ResNet-50 [21]	78.76%
AlexNet [22]	73.76%
LSTM [23]	78.35%
BiLSTM [24]	79.87%
Sketch-a-Net [25]	68.71%
SketchMate [7]	80.51%
Doodle-Variant [26]	78.13%
SketchFormer [27]	77.68%
SketchGCN [28]	70.04 %
S^{3}Net	**84.22%**
S^{3}Net (Stroke-5)	**85.10%**

Table 1. A comparative study of classification accuracy over state-of-art methods on QuickDraw dataset.

4.2. Competitors

We have evaluated S^{3}Net against several popular networks for image classification, such as **ResNet** [21] and **AlexNet** [22]. For feeding into CNNs, we transfer the sketch sequences into a raster pixel sketch scaled at $[224, 224, 3]$ with each channel tiled equally. **LSTM** [23] also serves as a classical RNN model used to process sequenced data. Other such CNN-based strong baselines for sketch recognition include **Sketch-a-Net** [25] and **SketchMate** [7]. Doodle-to-search [26], which is state-of-the-art for ZS-SBIR, is tailored for sketch recognition denoted as **Doodle-variant**. Specifically, the domain loss and semantic loss are removed from [26] because they are unavailable in our classification problem, and a cross-entropy loss is used instead for classification. **SketchFormer** [27] is a transformer-based representation for encoding free-hand sketches. One of their experimental setting, TForm-Cont, is with a continuous sketch modeling using 'stroke-5' format. Using 'stroke-5' we provide comparative results in that contrast, denoted as S3Net (Stroke-5). **SketchGCN** [28], a state-of-the-art for sketch segmentation, has been adjusted here for sketch recognition. The size of their multi-layer perceptron is expanded from 32 to 64, such that a 768D feature generated by SketchGCN, can be used for classification.

4.3. Results and Analysis

After comparing S^{3}Net with several state-of-the-art methods, we present the following quantitative results from Table 1: (i) S^{3}Net achieves the highest accuracy of **0.8422**, which is 4% above SOTA (0.8051). It proves that combining temporal and structural characteristics, classify abstract sketches better in feature space. (ii) CNNs and RNNs show a *narrow* performance gap (0.7876 **vs** 0.7987); which confirms that discriminative power of features based on temporal cues is *slightly* better than those based on appearance. (iii) Having more pen states for each segment, *S^{3}Net (Stroke-5)* outperforms Sketch-

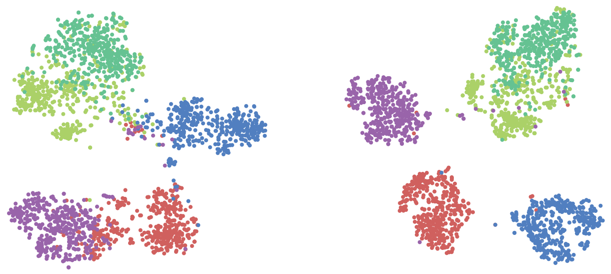

Fig. 3. Figure shows the feature distribution in feature space of RNN method (**Left**) and S^{3}Net (**Right**). We visualize five color coded different categories viz, "basket", "bear", "teddy-bear", "blackberry" and "purse". (T-SNE has been used for dimensional reduction)

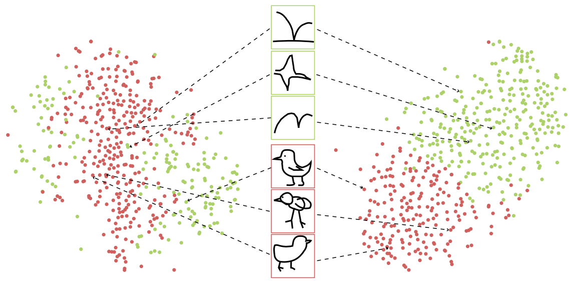

Fig. 4. Clustering result of "bird" data showing different color coded sub-clusters. Left: RNN; Right: S^{3}Net. We randomly select three samples points from the two sub-clusters in S^{3}Net and visualize them respectively. And we find the corresponding sample points in RNN sub-clusters.

Former, and even beats the original Stroke-3 setting, reaching **0.8510**, but comes at a cost of twice the training time. Furthermore, Fig. 3 shows GCN-generated features to obtain more separable sub-clusters than RNN ones, thus proving that GCNs can get better recognition performance using this structure. Interestingly, we also discovered our learned graph representation is also able to discriminate sketches within categories, roughly by their topological structures. Fig. 4 offers an illustrative example of the category "bird". It can be seen that S^{3}Net features tend to form two clear clusters of distinct structural patterns, whereas RNN features are more mixed.

4.4. Ablation Study

We conducted an ablation study to analyse different ways of building a sketch-graph (Table 2). **Node** denotes the component that is considered as a node in the sketch-graph. We define 4 options: (i) *Stroke*: Each stroke represents a node. (ii) *Segment*: Every segment denotes one node. (iii) *Segment+Stroke*: A stroke and its corresponding segment, are both selected as nodes. Hence total number of nodes equals the sum of all strokes and segments. (iv) *Segment+RN+VCN*: In this setting, nodes are built as mentioned in Sec 3.3 following a "segment-stroke-sketch" framework. **Edge** denotes the way in which nodes are connected in term of elements in the adjacency matrix. We define 3 such ways: (i) *Full*: Every pair of nodes will be connected. Hence all elements are 1 except those on the principle diagonal as we don't add self-loops. (ii) *Random*: Half of all elements are 1, except on the diagonal. (iii) *Hierarchy*: Edges are built as mentioned in Sec 3.3. **Stroke-cut** denotes a Boolean sense specifying whether we cut short a long stroke or not. Table 2 findings: (i) Considering experiments 1 and 2, we see *'Full'* connection outperforms *'Random'* one. This proves that the latter aggregates quite a lot of noisy messages resulting in performance degradation. (ii) Considering experiments 1 and 3, setting each stroke as a node fetches lower accuracy than setting each segment as one. A probable reason might be that the number of strokes in a sketch is usually less than 20, thus restricting adequate message-passing. (iii) Considering experiments 3 and 5, where adding middle level representative nodes is the differentiating factor, we find similar performances when it's set to *'Full'* connection. However comparing experiment 5 with 6, we find that hierarchical connections improve performance significantly. (iv) Finally after comparing experiment 7 with 9, we conclude from inference that the complete hierarchical structure delivers the best performance. This proves the supreme efficiency of S^{3}Net in building a sketch-graph.

#	Node	Edge	Stroke-cut	Acc.
1	Stroke	Full	×	68.98%
2		Random	×	63.77%
3	Segment	Full	×	71.79%
4		Random	×	64.31%
5	Segment+Stroke	Full	×	71.90%
6		Hierarchy	×	75.76%
7		Hierarchy	√	78.19%
8	Segment+RN +VCN	Full	×	72.90%
9		Hierarchy	√	**84.22%**

Table 2. Results of ablation experiments. The column '#' marks different methods of building a sketch-graph.

5. CONCLUSION

In this paper, we have proposed an approach for building a hierarchical sketch-graph to encode sketch structure. By designing a novel joint network for sketch recognition, sketch-specific traits have been explored, including temporal cues and structural invariance. Furthermore, our ablation study demonstrates the advantage of our hierarchical sketch-graph. In the future, we will investigate automatic means of constructing the initial sketch graph, and explore self-supervised feature learning by adopting structural invariance as pretexts.
Acknowledgement This work was partly supported by National Natural Science Foundation of China No.61806184, No.61601042, and BUPT Excellent PhD Student Foundation.

6. REFERENCES

[1] Yi Li, Yi Zhe Song, Timothy M. Hospedales, and Shaogang Gong, "Free-hand sketch synthesis with deformable stroke models," *IJCV*, 2017.

[2] Shuxin Ouyang, Timothy Hospedales, Yi-Zhe Song, and Xueming Li, "Cross-modal face matching: beyond viewed sketches," in *ACCV*, 2014.

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

[4] Ke Li, Kaiyue Pang, Yi-Zhe Song, Timothy M Hospedales, Tao Xiang, and Honggang Zhang, "Synergistic instance-level subspace alignment for fine-grained sketch-based image retrieval," *TIP*, 2017.

[5] Mathias Eitz, Kristian Hildebrand, Tamy Boubekeur, and Marc Alexa, "Sketch-based image retrieval: Benchmark and bag-of-features descriptors," *TVCG*, 2010.

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

[7] Peng Xu, Yongye Huang, Tongtong Yuan, Kaiyue Pang, Yi-Zhe Song, Tao Xiang, Timothy M Hospedales, Zhanyu Ma, and Jun Guo, "Sketchmate: Deep hashing for million-scale human sketch retrieval," in *CVPR*, 2018.

[8] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem, "Deepgcns: Can gcns go as deep as cnns?," in *ICCV*, 2019.

[9] Liuhao Ge, Zhou Ren, Yuncheng Li, Zehao Xue, Yingying Wang, Jianfei Cai, and Junsong Yuan, "3d hand shape and pose estimation from a single rgb image," in *CVPR*, 2019.

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

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

[12] Patsorn Sangkloy, Nathan Burnell, Cusuh Ham, and James Hays, "The sketchy database: learning to retrieve badly drawn bunnies," *ACM TOG*, 2016.

[13] Rosália G Schneider and Tinne Tuytelaars, "Sketch classification and classification-driven analysis using fisher vectors," *ACM TOG*, 2014.

[14] Yi Li, Yi-Zhe Song, and Shaogang Gong, "Sketch recognition by ensemble matching of structured features.," in *BMVC*, 2013.

[15] Ravi Kiran Sarvadevabhatla, Jogendra Kundu, et al., "Enabling my robot to play pictionary: Recurrent neural networks for sketch recognition," in *ACM MM*, 2016.

[16] Thomas N. Kipf and Max Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.

[17] Muhan Zhang and Yixin Chen, "Link prediction based on graph neural networks," in *NIPS*, 2018.

[18] Hanjun Dai, Bo Dai, and Le Song, "Discriminative embeddings of latent variable models for structured data," in *ICML*, 2016.

[19] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec, "Hierarchical graph representation learning with differentiable pooling," in *NIPS*, 2018.

[20] Will Hamilton, Zhitao Ying, and Jure Leskovec, "Inductive representation learning on large graphs," in *NIPS*, 2017.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012.

[23] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, 1997.

[24] Alex Graves and Jürgen Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural networks*, 2005.

[25] 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.

[26] Sounak Dey, Pau Riba, Anjan Dutta, Josep Llados, and Yi-Zhe Song, "Doodle to search: Practical zero-shot sketch-based image retrieval," in *CVPR*, 2019.

[27] Leo Sampaio Ferraz Ribeiro, Tu Bui, John Collomosse, and Moacir Ponti, "Sketchformer: Transformer-based representation for sketched structure," *arXiv preprint arXiv:2002.10381*, 2020.

[28] Lumin Yang, Jiajie Zhuang, Hongbo Fu, Kun Zhou, and Youyi Zheng, "Sketchgcn: Semantic sketch segmentation with graph convolutional networks," *arXiv preprint arXiv:2003.00678*, 2020.