SKETCHPOINTNET: A COMPACT NETWORK FOR ROBUST SKETCH RECOGNITION

Xiangxiang Wang *Xuejin Chen* *Zhengjun Zha*

University of Science and Technology of China

ABSTRACT

Sketch recognition is a challenging image processing task. In this paper, we propose a novel point-based network with a compact architecture, named SketchPointNet, for robust sketch recognition. Sketch features are hierarchically learned from three mini PointNets, by successively sampling and grouping 2D points in a bottom-up fashion. SketchPointNet exploits both temporal and spatial context in strokes during point sampling and grouping. By directly consuming the sparse points, SketchPointNet is very compact and efficient. Compared with state-of-the-art techniques, SketchPointNet achieves comparable performance on the challenging TU-Berlin dataset while it significantly reduces the network size.

***Index Terms*—** Sketch recognition, point set, deep neural network, stroke pattern

1. INTRODUCTION

With the popularity of mobile and potable devices, sketching is getting easier for common users. As a powerful and effective tool for communication, freehand sketches have drawn a large amount of attention in image processing research. Many approaches have been proposed for sketch recognition [1, 2, 3, 4, 5, 6].

Freehand sketch possesses following unique characteristics, which make sketch recognition a challenging task. First, a sketch only contains extremely sparse visual signals as it is a very brief representation without any color or texture signals. A large portion of pixels within a sketch image are blank. Second, freehand sketches present large intra-class variance as sketches for the same object drawn by different users are extremely diverse. The degree of shape distortions varies drastically from average users to professional artists. Third, the order of which the strokes are drawn in a sketch matters a lot, and thus provides temporal context for sketch recognition. As claimed in previous research [1], sketches are typically drawn in a coarse-to-fine strategy. Details are usually added to the sketch later with short strokes. Therefore, designing the technique for sketch classification should take the sparsity and temporal context into account.

Ealier techniques for sketch recognition were mainly designed for recognizing hand-drawn numbers or letters [7, 8, 9]. Hse and Newton [7] used Zernike moments as features for sketches. Laviola et al. [8] developed an approach for common mathematical symbol recognition. Fonseca and Jorge [9] designed geometrical features according to the profiles of stokes. These approaches achieved good performance on handwritten symbols, which are simpler than sketches of common objects. Eitz et al. [1] comprehensively analyzed the way of how human draw sketches, and published the TU-Berlin benchmark, a large dataset including 20000 sketches of 250 categories of objects. This dataset is significantly challenging, and humans only achieve a 73.1% recognition accuracy. They extracted HOG features and used a support vector machine (SVM) for classification. Other methods [2, 3] also extracted hand-crafted features and used SVM classifiers. However, for common objects, the representation capability of hand-crafted features is highly limited.

CNN-based sketch recognition. With the great success of deep neural network (DNN) in natural image recognition, convolutional neural networks (CNN) have been applied to sketch recognition. A certain number of existing approaches, such as Sketch-a-Net [4] and DeepSketch [5], straightforwardly convert freehand sketches into 2D images and exploit CNN to extract features for recognition from sketch images. These CNN-based approaches achieve state-of-the-art performance on the challenging TU-Berlin benchmark. However, regarding sketches as 2D pixel array, these CNN-based approaches require a large amount of network parameters to be learned. As mentioned above, compared to natural images, freehand sketches are very sparse in 2D space. The data sparsity could greatly improve the network compactness. Therefore, we adopt point-based DNN to design a compact network.

Point-based DNNs. Compared to CNN, the brand-new concept, which directly takes the discrete points as input, has been presented for point cloud processing recently [10, 11, 12]. The point-based DNN significantly reduces the model space and computational complexity. PointNet [10] presented the first deep network that directly consumes points for many applications of point clouds. Its subsequent version, PointNet++ [11] used a hierarchical architecture to extract features from 3D point clouds for more robust recognition and segmentation. Li et al. [12] proposed a χ-transformation to permute unordered points into a latent potentially canonical order for feature learning. While point-based networks were designed for point sets, they only exploit the spatial pat-

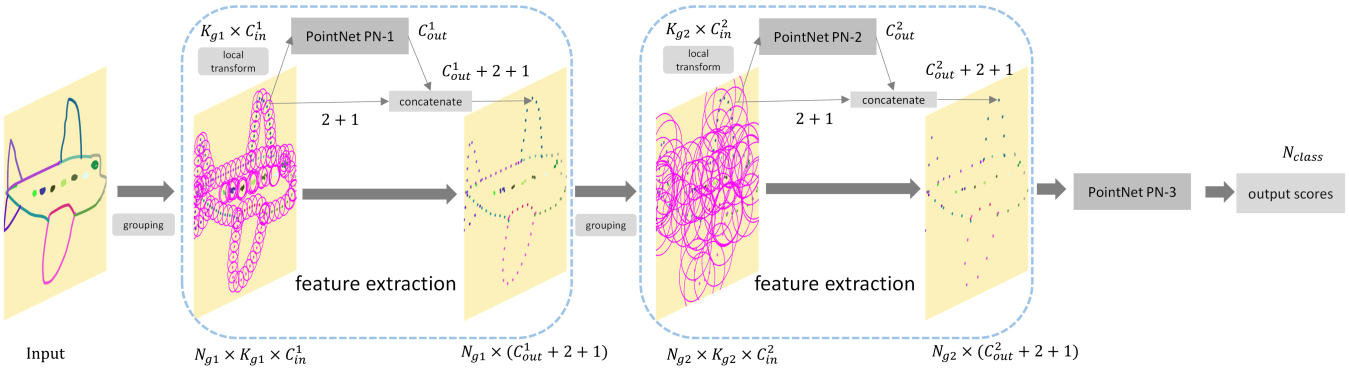

Fig. 1. SketchPointNet architecture. Given the input sketch, three mini PointNets are used to hierarchically extract features. Points are sampled along strokes and grouped in a bottom-up fashion.

tern. In addition to spatial patterns, sketches contain temporal context, which is an another important hint for recognition.

In this paper, we propose a point-based deep network, named SketchPointNet, which takes advantages of both spatial and temporal pattern for efficient and robust sketch recognition. Comparing with existing ConvNet-based sketch recognition approaches, we treat each sketch as a series of points, which is a sparser representation than images. Comparing with the point-based DNNs, our SketchPointNet takes the stroke order into account. By hierarchically extracting features from low level to the global level using three mini PointNets, the proposed SketchPointNet achieves comparable performance with state-of-the-art techniques on the challenging TU-Berlin dataset, and significantly reduces the number of network parameters.

2. METHODOLOGY

In this section, we will first introduce the network architecture of our SketchPointNet. Next, we describe the algorithm details of point sampling and grouping from sketch strokes, as well as the network to extract both local and global patterns for recognition.

Architecture of SketchPointNet. Fig. 1 shows the architecture of our SketchPointNet. The directly captured stroke paths from various input devices are typically contact points that distribute unevenly along strokes with different movement speeds and device sizes. To make them directly comparable, we resample the input sketch $\mathcal{S}$ into N evenly distributed points as a point set $\mathbb{P} = \{P_i | i = 1, \ldots, N\}$. A transformation net is firstly employed to regress a transformation matrix to align the input sketch, similar with PointNet [11]. Given the resampled and transformed point set, we extract the sketch features hierarchically with three mini PointNets (PN-1, PN-2, PN-3) from the lower level to the global level. In each level, we group the point set $\mathbb{P}$ into K_g groups according to their spatial and temporal neighborhood in different size. Stroke features are extracted using a mini PointNet [11] for each point group. From the global pattern extracted and used to predict the scores for the N_{class} sketch categories in the third PointNet (PN-3).

While the overall architecture of our SketchPointNet is similar with PointNet++ [11], the main difference lies on the sampling and grouping procedure. Targeting at 3D point clouds, PointNet++ [11] considers the spatial distribution of points in the 3D space. In contrast, not only the spatial pattern is important, the temporal pattern is also important for sketch recognition. During our sampling and grouping, the temporal and spatial information is integrated together to extract sketch features for robust recognition.

Point sampling. To make various sketches comparable, we sample N_{pt} equidistantly spaced points along strokes. By sampling along strokes, the temporal information is encoded in the point distribution. The length of each stroke is calculated and the total length of all strokes is $L_{sum} = \sum_{i=1}^{M} L_i$, where M is the stroke number. Given the desired point number N_{pt}, we assign each stroke $N_i = \frac{L_i}{L_{sum}} N_{pt}$ points according to its stroke length L_i. Then we reample each stroke into B_i points equidistantly along the stroke. Fig. 2 shows the sampling results with different number of points. We can see that the main structure is well preserved during the sampling procedure without losing much details, even with a very small number (N=128) of points.

Point grouping. To extract features of the input sketch, we group a set of points on the strokes and feed them into a mini PointNet, which is flexible with the number of input points. Given a desired number of groups N_{group} at a certain level, we first sample N_{group} points from the input sketch as the group centers using the sampling method described above. Then, for each group, we extract a set of points equidistantly on the input strokes in an area of radius r surrounding the group center. The number K_{group} of points in different groups varies over the sketch region. We set an upper bound for K_{group} to reduce the computational complexity. As Fig. 3 shows, the points in a local region with different

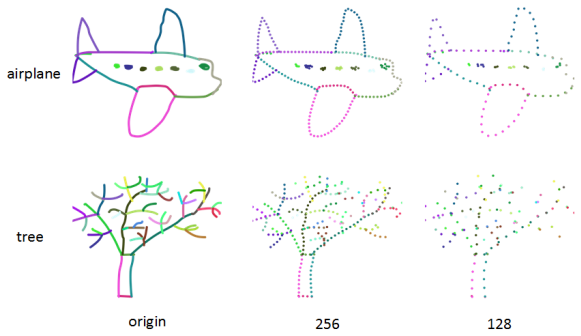

Fig. 2. Sketch resampling with different number of points. Each continuous stroke is shown in a unique color.

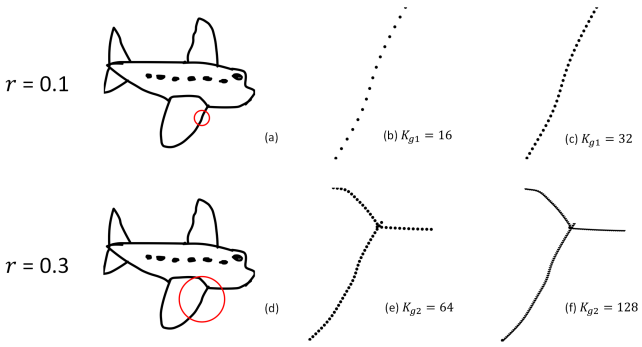

Fig. 3. By grouping points in local regions of different radious r, features of different level are extracted. The middle and right columns show the sampled points with different number of points K_g in a local region.

radius present different local patterns. By sampling points along strokes, both temporal and spatial patterns are preserved during the grouping procedure, which are essential for sketch recognition.

Pattern extraction. We use a modified version of the PointNet [10] to extract the sketch pattern in a local region that contains various number of points. The architecture of our mini PointNet is shown in Fig. 4. For each group, a mini PointNet is fed with K_g points in its local region with data size $K_g \times C_{in}$. C_{in} is the number of data channel of each point. Each mini PointNet consists three parts. The first part is K_g shared-weight three-layer perceptrons (with M_1^1,M_2^1,M_3^1 neurons at each layer respectively), which converts the input data of each point into another feature space. The second part is a max-pooling layer to aggregate features of the K_g points into a fixed-dimension feature. The third part is a three-layer perceptron (with M_1^2,M_2^2,M_3^2 neurons at each layer respectively). The output of the mini PointNet is a $C_{out} = M_3^2$ dimensional feature vector.

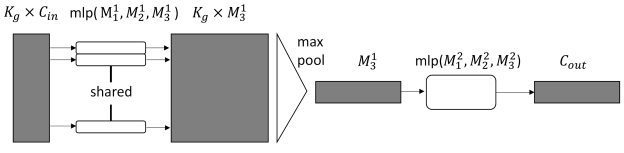

Fig. 4. Architecture of our mini PointNet.

Initially, we have the 2D coordinates and stroke order of each point for the normalized sketch. We translate each point into a local frame relative to the center point of each group. Therefore, the data channel $C_{in}^1 = 2+1$ for the first PointNet PN-1, including the 2D local coordinates and 1-D stroke order. The output of PN-1 is a C_{out}^1 dimensional feature vector for each group. Different with PointNet++ [11] for 3D point clouds, the stroke order does matter for sketch recognition, because human typically draw the overall shape and then add fine details later. The strokes drawn earlier have more impact effects for recognizing a coarse sketch.

Then, increasing the radius r_2 for grouping points, which is equivalent to increasing the receptive field, we extract sketch patterns in a higher level. For the second PointNet PN-2, the input is the $K_{g2} \times C_{in}^2$ data, where K_{g2} is the number of points in the local region of a group, $C_{in}^2 = 2 + 1 + C_{out}^1$ for each group, including the 2-D local coordinate of the group center, 1-D stroke order and the C_{out}^1-D feature extracted for the local region from the first PointNet. Thus, features of little groups that extracted from PN-1 are merged together through PN-2. The output of PN-2 is a C_{out}^2-D feature vector for each group.

Sketch classification. After the two-level local pattern extraction, we feed the $N_{g2} \times C_{out}^2$ data into the third mini PointNet PN-3, which produces $C_{out}^3 = N_{class}$ dimensional vector as the predicted scores for the N_{class} categories.

3. EXPERIMENTS AND EVALUATION

In this section, we evaluate our method on the TU-Berlin sketch benchmark. It contains in total 20000 sketches of 250 categories. Each category has 80 instances. We conducted a group of experiments to compare our method with other methods in both recognition accuracy and number of trainable parameters. We performed three-fold cross validation within the dataset using the same strategy with previous methods [4, 6]. Dataset augmentation is also performed to ease the over-fitting problem, same as [4].

Training strategy. Our SketchPointNet, which is composed of all fully-connected layers, can be trained end-to-end by back-propagation and Adam, with the softmax classification loss. For the multilayer perceptrons in the entire network, we use parameters mlp(8,16,64), mlp(64,16,2) in PN-1, mlp(32,32,128), mlp(128,64,29) in PN-2, mlp(64,128,1024), mlp(512,256,250) in PN-3. However, the entire network is easily converged to a local optima and suffers from over-fitting. Moreover, the third PointNet PN-3 plays a different role from PN-1 and PN-2. It captures more global patterns

for classification. The network parameters are densely distributed in PN-3, which tends to converge on the training set without sufficient training for PN-1 and PN-2.

To solve this problem, we developed a three-step training strategy. We re-initialize parameters randomly in PN-3 to ensure that the parameters in PN-1 and PN-2 are trained sufficiently. In the first step, we randomly initialize all the parameters in the SketchPointNet and train it in an end-to-end fashion. In the second step, we initialize the first two mini PointNets PN-1 and PN-2 for local pattern extraction with the parameters obtained in the first step, and randomly initialize the parameters in the third mini PointNet PN-3. Then we retrain the entire network. In the third step, the second step is repeated for three times. Table 1 shows the improvement of recognition accuracy with our three-step training strategy.

Table 1. Recognition accuracy under different training step.

Training	step 1	step 2	step 3
split 1	70.8 %	73.8%	74.3%
split 2	71.2%	73.6%	74.3%
split 3	70.8%	72.5%	74.0%

Accuracy and Model Size. We compare our SketchPointNet with state-of-the-art approaches on the TU-Berlin benchmark, considering both the recognition accuracy and model compactness. Comparing with traditional techniques using hand-crafted features, our SketchPointNet performs significantly better. We compare our method with two types of approaches using deep networks. The first type of methods include Sketch-a-Net [4], DeepSketch 1 [5], and DeepSketch 2 [6]. These methods consider sketches as raster images and use deep convolutional networks as image classification. The second type includes PointNet++ [11], PointCNN [12], which treat sketches as a set of points.

The comparison is presented in Table 2. As shown, our algorithm achieves comparable performance 74.22% with image-based convolutional networks. Compared to the vanilla version without any fusion part, our SketchPointNet achieves higher performance than Sketch-a-Net (vanilla) [4], and comparable results with DeepSketch 1 [5]. However, as designed, our SketchPointNet is very compact by using lots of shared weights. It has about 5 times less parameters than Sketch-a-Net (vanilla) [4] and 30 times less parameters than DeepSketch 1 [5].

Compared with the point-based methods (PointNet++ [11] and PointCNN [12]), our SketchPointNet improves the recognition accuracy by more than 6%. This is mainly because our algorithm considers both temporal and spatial patterns while they focus on the spatial structure only.

Fig. 5 shows two groups of examples. In the top row, the three examples are correctly recognized as their ground-truth categories. Compared to PointNet, which investigates critical points for each class, our SketchPointNet extracts critical patterns for each category. Take the first sketch as example, the head part is recognized as a panda, though its pose is more like a human. Similarly, SketchPointNet recognizes the second sketch as tiger mainly from the head part and the stripes on its body. Although our network can handle some challenging cases, there are still some challenging cases. As the bottom row in Fig. 5 shows, the sketches are extremely ambiguous cases.

Table 2. Comparison with state-of-the-art results on both recognition accuracy and model compactness.

Method	Accuracy	#Parameters
Sketch-a-Net [4]	77.95%	8.2M
DeepSketch 2 [6]	77.69%	276M
DeepSketch 1 [5]	75.42%	55.1M
Sketch-a-Net(vallina) [4]	72.6%	8.2M
PointNet++ [11]	66.53%	1.73M
PointCNN [12]	67.72%	0.89M
Ours	74.22%	1.64M

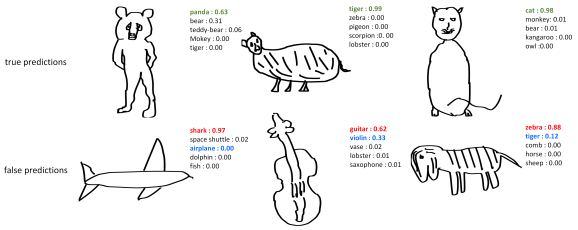

Fig. 5. Some easy cases (top row) and challenging cases (bottom row) for SketchPointNet. The green labels indicate correct predictions. The red lables indicate the wrong classification, while the ground-truth categories are shown in blue.

4. CONCLUSION

In this paper, we propose SketchPointNet, a compact deep neural network for robust sketch recognition. Our network directly consumes 2D points on freehand sketches, which are sparsely distributed in the 2D space. By sampling and grouping points while considering both temporal order and spatial distribution, local features are learned hierarchically using three mini PointNets. The experiments on the challenging TU-Berlin dataset demonstrated that our SketchPointNet achieves a high recognition accuracy, and significantly reduces the number of network parameters.

Acknowledgements. This work was supported by the National Key Research & Development Plan of China under Grant No. 2016YFB1001402 and the National Natural Science Foundation of China under Grants. 61472377, 61632006, 61331017,61622211, and 61472392.

5. REFERENCES

[1] M. Eitz, J. Hays, and M. Alexa, "How do humans sketch objects?," *ACM Trans. Graph.*, vol. 31, pp. 44:1–44:10, 2012.

[2] Y. Li, T.M. Hospedales, Y.Z. Song, and S. Gong, "Free-hand sketch recognition by multi-kernel feature learning," *Computer Vision and Image Understanding*, vol. 137, pp. 1–11, 2015.

[3] R.G. Schneider and T. Tuytelaars, "Sketch classification and classification-driven analysis using fisher vectors," *ACM Trans. Graph.*, vol. 33, pp. 174:1–174:9, 2014.

[4] Q. Yu, Y. Yang, Y.Z. Song, T. Xiang, and T.M. Hospedales, "Sketch-a-net that beats humans," in *BMVC*, 2015.

[5] O. Seddati, S. Dupont, and S. Mahmoudi, "Deepsketch: Deep convolutional neural networks for sketch recognition and similarity search," *CBMI*, pp. 1–6, 2015.

[6] O. Seddati, S. Dupont, and S. Mahmoudi, "Deepsketch 2: Deep convolutional neural networks for partial sketch recognition," *CBMI*, pp. 1–6, 2016.

[7] H.H. Hse and A.R. Newton, "Sketched symbol recognition using zernike moments," *ICPR*, vol. 1, pp. 367–370 Vol.1, 2004.

[8] J.J LaViola and R.C. Zeleznik, "Mathpad2: a system for the creation and exploration of mathematical sketches," in *SIGGRAPH Courses*, 2004.

[9] M.J. Fonseca and J.A. Jorge, "Using fuzzy logic to recognize geometric shapes interactively," in *FUZZ*, 2000.

[10] C.R. Qi, H. Su, K. Mo, and L.J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

[11] C.R. Qi, H. Su, K. Mo, and L.J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in Neural Information Processing Systems 30*, pp. 5105–5114. 2017.

[12] Y. Li, R. Bu, M. Sun, and B. Chen, "Pointcnn," *arXiv:1801.07791*, 2018.