Supplementary Conditional Stroke Recovery for Fine-Grained Sketch-Based Image Retrieval

Zhixin Ling, Zhen Xing, Jian Zhou, and Xiangdong Zhou†

Fudan University, Shanghai, China
{20212010005,zxing20,19212010009,xdzhou}@fudan.edu.cn

In this document, we provide additional materials to supplement our main submission. In Sec. 1, we provide more visualization for our stroke disorder algorithm. In Sec. 2, we elaborate our experiment setup. In Sec. 3, we discuss a loss similar to $\mathcal{L}_{da}$. In Sec. 4, we present details of CSR-GoogLeNet, CSR-InceptionV3, CSR-Densenet169 and CSR-ResNet18.

1 Some Details of Stroke Disorder Algorithm

In our main paper, to emphasize effectiveness of the recovery process, we present some extremely disordered cases and simply introduce the stroke extraction process. Ine this section, we visualize the stroke disorder algorithm step by step. The simplest case is shown in Fig. 1. Ln1 has already grouped the pixels into enough strokes. In most cases, however, these exists only one or two stroke in the sketch. So we execute Ln2-9 to divided the pixels into more strokes: **1)** we find a chokepoint; **2)** we obtain a direction of maximum variance in the neighborhood pixels, *i.e.*, $\mathbf{v}$; **3)** we cut the stroke along the direction perpendicular to $\mathbf{v}$. The processes are visualized in Fig.2. We cut a long stroke at the chokepoint because we assumes that a stroke is likely to end up there.

There exists some cases where the algorithm might not well even if the assumption is true. For example, when two ends of the stroke are connected to a larger stroke, the stroke is unlikely to be truncated since both ends are unlikely to be removed in Ln6. Case(1) in Fig.2 is a typical case. Many wheel spokes are connected with the hub and the rim. Although the algorithm tries to cut the rim side of the stroke, some strokes are still connected to the hub.

n_s controls the integrity of strokes. A larger breaks the strokes into more pieces. Empirically, setting $n_s = 10$ is marginally better than $n_s = 5$. Setting it to 2 might be the best for visualization, but it yields unsatisfactory performance.

2 Experiment Setup

2.1 Implementation Details

CSR is implemented using PyTorch 1.2.0 on a single GTX 1080Ti GPU on Ubuntu18.04 system with 64G memory. The CPU is i7-6800K. The GPU memory is 11.2GB.

† Corresponding author.

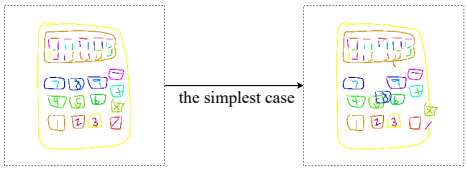

Fig. 1. The simplest case of Stroke Disorder algorithm(Alg 1 in the main paper).

Since FG-SBIR works in the last three years [4,1,2,5,10] adopt GoogLeNet [8], InceptionV3 [9] and DenseNet169 [3], we report results based on all three backbones in our paper. Considering that GoogLeNet is more light-weighted, we conduct all experiments based on GoogLeNet if not specified in this paper. We elaborate the network configuration based on GoogLeNet here and detailed configuration of InceptionV3 is similar. E^f is pretrained on ImageNet. Output of E^r is activated by a sigmoid function.

Temperature $\tau = 0.005$ and weight w is set to 10. m is set to 128. We use Adam [6] optimizer with $\beta_1 = 0.5, \beta_2 = 0.999$ to optimize our entire model for 100000 steps. The learning rate starts at 2×10^{-4} and is exponentially decayed to 2×10^{-5}. To increase the recovery difficulty gradually, p_d starts at 0.1 and linearly grows to 0.3. We dynamically set $\alpha = 1 - \alpha_p p_d, \alpha_p = 2$. Inputs of the feature extraction network are resized to 224×224 using CSR-GoogLeNet and CSR-Densenet169 while the input size is 299×299 using CSR-InceptionV3; A sketch-image pair is augmented by random horizontally flipping in pair. Besides, images are augmented by random color jittering, random RGB channel permutation and random rotation.

Following Qian *et. al.* [11], we report top-1 accuracy (acc@1) and top-10 accuracy (acc@10). Top-n accuracy is defined as acc@n $= \mathbb{E}_s \sum_i^n \text{match}(\widetilde{p}_{s,i}, s)$, where $\widetilde{p}_{s,i}$ is the i-th retrieved image for query sketch s and $\text{match}(\widetilde{p}_{s,i}, s)$ indicates whether $\widetilde{p}_{s,i}$ matches s.

2.2 Datasets

We evaluate our CSR method on four benchmark fine-grained sketch-image datasets: Sketchy [7], QMUL-Shoe, QMUL-Chair, and QMUL-ShoeV2 [11].

Sketchy [7] totally contains 74425 sketches and 12500 images from 125 categories, making it the largest fine-grained sketch-image dataset. Each image is paired with at least five sketches in this dataset. Images in the Sketchy dataset can be quite noisy and the paired sketches along with the image can also be slightly misaligned (*e.g.*, the sketch can be slightly moved, rotated, or distorted). Besides, there could be several very similar images. These factors makes this dataset quite challenging. Following [7], we randomly sample 90% images and their paired sketches for training and use the rest for testing. We fill in the training batch with sketch-image pairs from the same category.

QMUL-Shoe and **QMUL-Chair** [11] contain 419 and 297 sketch-image pairs respectively, where each image is paired with only one sketch. All the

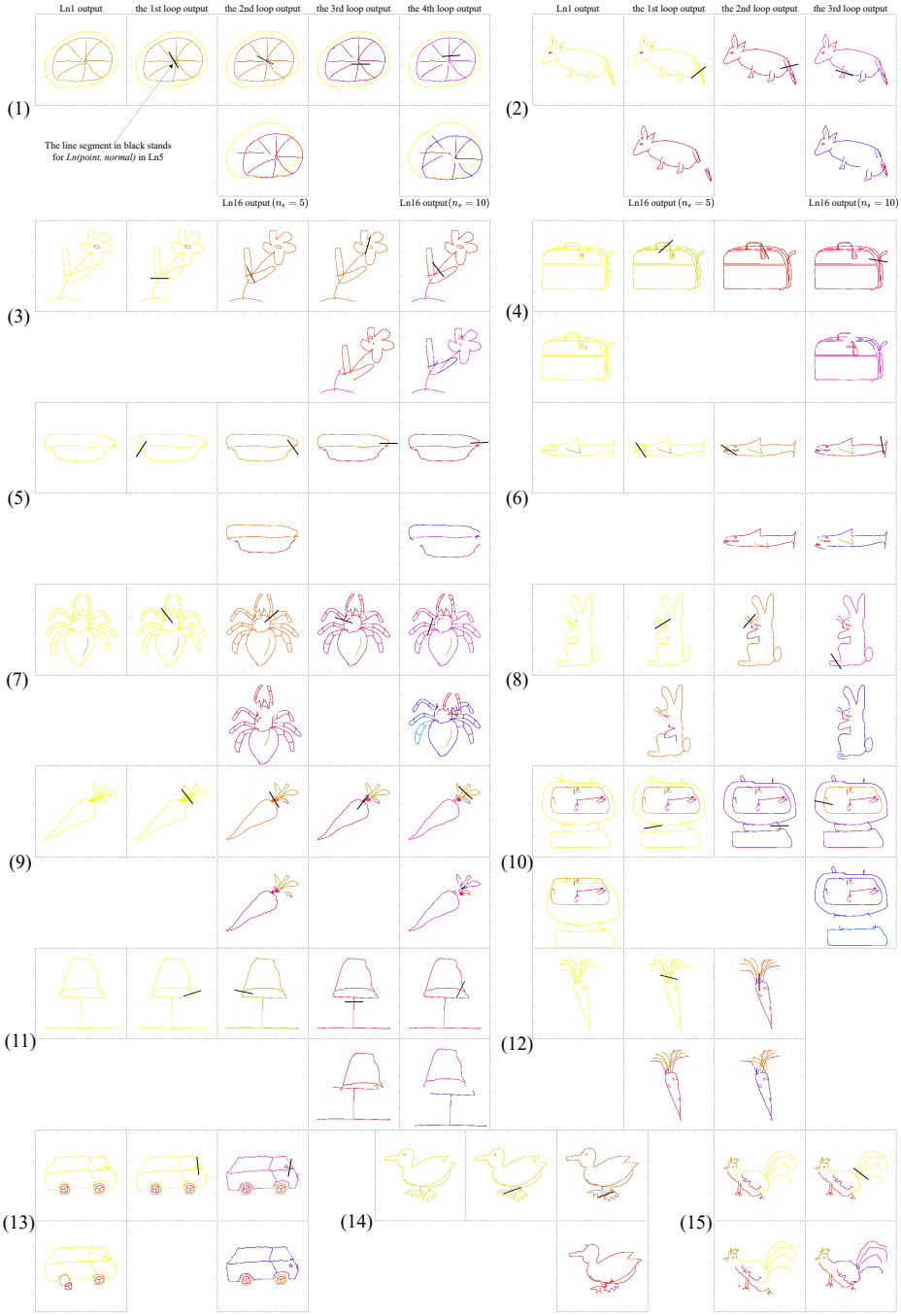

Fig. 2. Stroke Disorder algorithm visualization(Alg 1 in the main paper). We show how the algorithm runs. Different colors stands for different strokes.

images and sketches in these two datasets are clean and well aligned. Following the split of [11], 304 (*resp.*, 200) pairs of QMUL-Shoe (*resp.*, QMUL-Chair) are used for training and the rest for testing.

QMUL-ShoeV2 [11] is an extension of QMUL-Shoe dataset. There are totally 2000 images and 6730 sketches in the dataset, where each image is paired with at least three sketches. Similarly, images and sketches in this dataset are clean and well aligned. Following [4], we use 200 images and their paired sketches for testing and the rest for training.

3 A Loss Similar to $\mathcal{L}_{da}$

In our main paper, we defines $\mathcal{L}_{da}$:

$$\mathcal{L}_{da} = -\log \frac{e^{sim(\mathbf{f}_s,\mathbf{f}_{p_+})} + \alpha e^{sim(\mathbf{f}_{s'},\mathbf{f}_{p_+})}}{e^{sim(\mathbf{f}_s,\mathbf{f}_{p_+})} + \alpha e^{sim(\mathbf{f}_{s'},\mathbf{f}_{p_+})} + \sum_{p_-}(e^{sim(\mathbf{f}_s,\mathbf{f}_{p_-})} + \alpha e^{sim(\mathbf{f}_{s'},\mathbf{f}_{p_-})})}. \tag{1}$$

The loss encourages $\mathbf{f}_{p_+}$ to move towards $\mathbf{f}_{s'}$ when closing the distance between $\mathbf{f}_s$ and $\mathbf{f}_{p_+}$. It might be noticed that a sum of two $\mathcal{L}_{sa}$ might also be able to complete the same job:

$$\mathcal{L}_{ssa} = -\log \frac{e^{sim(\mathbf{f}_s,\mathbf{f}_{p_+})}}{e^{sim(\mathbf{f}_s,\mathbf{f}_{p_+})} + \sum_{p_-} e^{sim(\mathbf{f}_s,\mathbf{f}_{p_-})}} - \beta \log \frac{e^{sim(\mathbf{f}_{s'},\mathbf{f}_{p_+})}}{e^{sim(\mathbf{f}_{s'},\mathbf{f}_{p_+})} + \sum_{p_-} e^{sim(\mathbf{f}_{s'},\mathbf{f}_{p_-})}}, \tag{2}$$

where β works similarly to α. $\mathcal{L}_{da}$ couples $sim(\mathbf{f}_s, \mathbf{f}_{p_+})$ and $sim(\mathbf{f}_{s'}, \mathbf{f}_{p_+})$ more closely than $\mathcal{L}_{ssa}$. Therefore, $\mathcal{L}_{da}$ does better in restricting the growth of $sim(\mathbf{f}_{s'}, \mathbf{f}_{p_+})$ when maximizing $sim(\mathbf{f}_s, \mathbf{f}_{p_+})$. As a result, $\mathcal{L}_{da}$ outperforms $\mathcal{L}_{ssa}$ by 0.6% acc@1 on Sketchy and 1.1% acc@1 on QMUL-ShoeV2.

4 Detailed CSR Configuration

The CSR network architectures based on GoogLeNet [8], InceptionV3 [9] and Densenet169 [3] are similar. The detailed of CSR-GoogLeNet, CSR-InceptionV3 and CSR-Densenet169 are illustrated in Fig. 3 Fig. 4, Fig. 5 and Fig. 6. The differences among CSR-GoogLeNet, CSR-InceptionV3, CSR-Densenet169 and CSR-ResNet18 are listed in Tab. 1.

CSR Backbone	input size	size of $\mathbf{e}$	retrieval dimention	batch size
GoogLeNet	224x224	56×56	$1024 + 3m$	96
InceptionV2	299x299	73×73	$2048 + 3m$	64
Densenet169	224x224	56×56	$1664 + 3m$	48
ResNet18	224x224	56×56	$512 + 3m$	96

Table 1. CSR architecture setting comparison using different backbones.

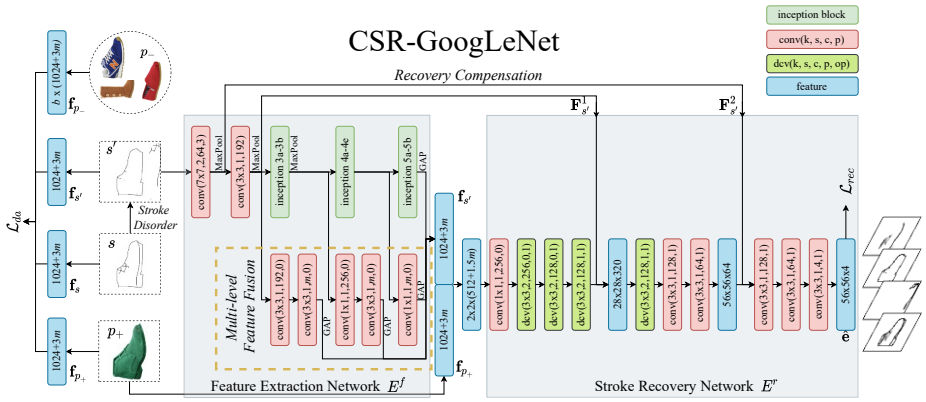

Fig. 3. The overall CSR network architecture based on GoogLeNet. "GAP" is Global Average Pooling and "MaxPool" is max pooling (stride=2, kernel=3x3). In the legends, "conv"/"dcv" denote a convolutional/deconvolutional layer. "k", "s", "c", "p" and "op" denote kernel size, stride, output channel, padding and output padding respectively. There is batch normalization and a RELU rectifier following each convolutional or deconvolutional layer, which is omitted in the illustration for cleanness.

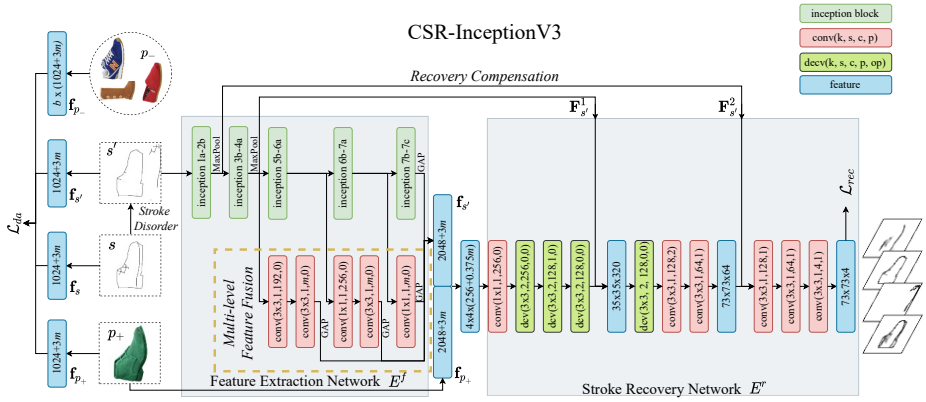

Fig. 4. The overall CSR network architecture based on InceptionV3.

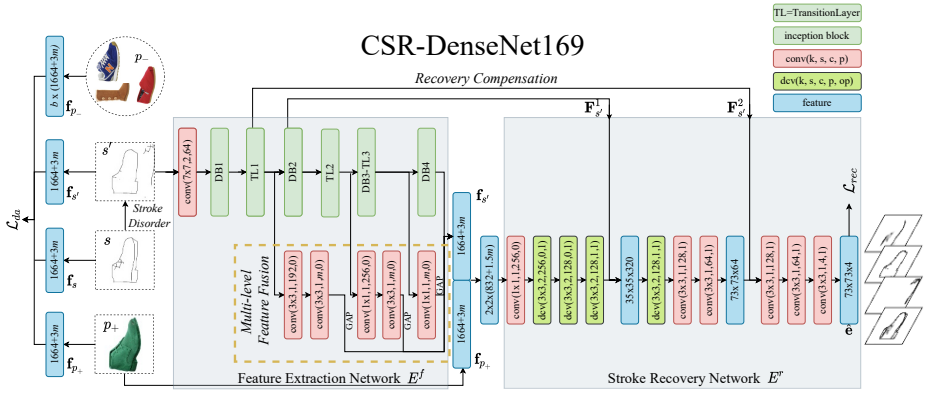

Fig. 5. The overall CSR network architecture based on DenseNet169.

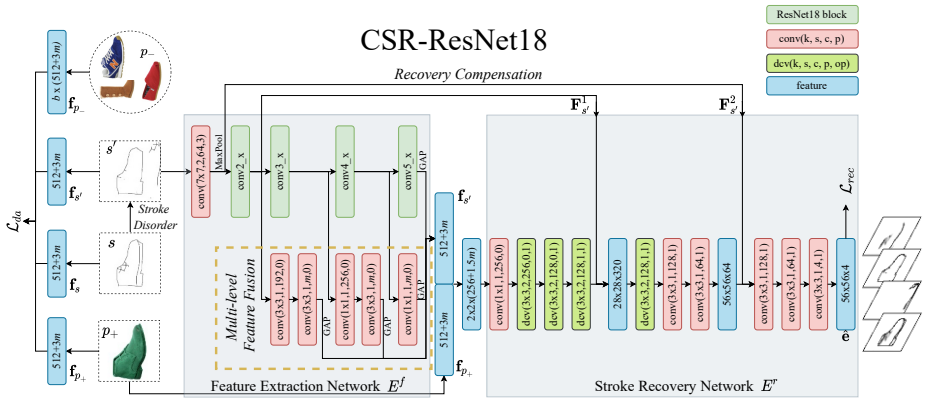

Fig. 6. The overall CSR network architecture based on ResNet18.

References

1. Bhunia, A.K., Chowdhury, P.N., Sain, A., Yang, Y., Xiang, T., Song, Y.: More photos are all you need: Semi-supervised learning for fine-grained sketch based image retrieval. In: Computer Vision and Pattern Recognition (CVPR) (2021)
2. Bhunia, A.K., Yang, Y., Hospedales, T.M., Xiang, T., Song, Y.Z.: Sketch less for more: On-the-fly fine-grained sketch-based image retrieval. In: Computer Vision and Pattern Recognition (CVPR) (2020)
3. Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Computer Vision and Pattern Recognition (CVPR). pp. 2261–2269 (2017)
4. Lin, H., Fu, Y., Lu, P., Gong, S., Xue, X., Jiang, Y.G.: Tc-net for isbir: Triplet classification network for instance-level sketch based image retrieval. In: ACM International Conference on Multimedia (ACM MM) (2019)
5. Pang, K., Yang, Y., Hospedales, T.M., Xiang, T., Song, Y.: Solving mixed-modal jigsaw puzzle for fine-grained sketch-based image retrieval. In: Computer Vision and Pattern Recognition (CVPR) (2020)
6. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in PyTorch. In: NeurIPS Autodiff Workshop (2017)
7. Sangkloy, P., Burnell, N., Ham, C., Hays, J.: The sketchy database: Learning to retrieve badly drawn bunnies. ACM Transactions on Graphics (TOG) (2016)
8. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: computer vision and pattern recognition (CVPR). pp. 1–9 (2015)
9. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Computer Vision and Pattern Recognition (CVPR). pp. 2818–2826 (June 2016)
10. Yanfei, W., Fei, H., Yuejie, Z., Rui, F., Tao, Z., Weiguo, F.: Deep cascaded cross-modal correlation learning for fine-grained sketch-based image retrieval. Pattern Recognition(PR) (2019)
11. Yu, Q., Liu, F., Song, Y.Z., Xiang, T., Hospedales, T.M., Loy, C.C.: Sketch me that shoe. In: Computer Vision and Pattern Recognition (CVPR) (2016)