Multimedia Tools and Applications (2020) 79:33891–33906
https://doi.org/10.1007/s11042-020-08706-y

Stroke classification for sketch segmentation by fine-tuning a developmental VGGNet16

Xianyi Zhu[1] · Jin Yuan[1] · Yi Xiao[1] · Yan Zheng[2] · Zheng Qin[1]

Received: 1 February 2019 / Revised: 14 December 2019 / Accepted: 28 January 2020 /
Published online: 9 March 2020

Abstract
Sketch segmentation and labeling face two challenges: few samples and few features. 3D data-driven methods use additional labeled 3D meshes to increase samples. However, they are not feasible for the abstract sketches that have no corresponding 3D meshes. And hand-crafted feature based methods, although need no 3D meshes, are sensitive to various strokes. To address the challenges, we explore transfer learning based on convolutional neural network (CNN) by fine-tuning a pre-trained CNN to classify strokes for sketch segmentation. We propose a novel informative input for the CNN, making the position information of strokes clear. To improve fine-tuning during transfer learning, we propose to add grouped filter layers to the CNN, making the CNN's representational capacity incremental. Compared with the state-of-arts, our experimental results achieve 9.7% improvement on the abstract sketch dataset, and 2% improvement on the sketch dataset that has corresponding 3D meshes.

Keywords Stroke classification · Sketch segmentation · Transfer learning · Fine-tuning · Convolutional neural network

1 Introduction

Freehand sketching is an intuitive method for people to conduct visual communication. With the spread of touchscreen devices, sketching becomes easy for humans at present. Many problems and applications on sketch have been researched in recent years, for example, sketch-based image retrieval and synthesis [35, 36, 44, 45, 57], sketch-based 3D model retrieval [17, 24], sketch recognition [7, 40, 53]. However, existing researches on sketch understanding almost focus on object level. Exact semantic understanding of sketch on

✉ Jin Yuan
yuanjin@hnu.edu.cn

[1] College of Computer Science and Electronic Engineering, Hunan University, Changsha, People's Republic of China

[2] College of Electrical and Information Engineering, Hunan University, Changsha, People's Republic of China

part level also deserves to be explored further, which helps to avoid manual segmentation operations in sketch-based user interfaces [4, 18, 49, 50].

In this paper, our task is to assign each stroke with a label. The label is one of a pre-defined set of labels, and the stroke's host sketch is from a given category. We believe that the task faces two major challenges. (i) *feature extraction* Sketches are composed of poorly drawn strokes that can not provide any color information. Strokes with the same label may have huge shape and length differences. And the position information of individual strokes is ambiguous. (ii) *sample size* Existing benchmark datasets [10, 39] only consist of a small number of samples. Each category in Huang dataset [10] has 30 sketch samples, and that in Schneider dataset [39] has 20 sketch samples. Each sketch in the datasets is composed of $2 \sim 30$ strokes.

To overcome the two challenges, we explore transfer learning based on convolutional neural network (CNN). CNNs [14, 16, 41] have been shown their ability of learning to extract effective features from raw inputs. And with transfer learning approaches [2, 19, 46, 47], we can make use of abundant natural image data to pre-train a powerful CNN model. Specifically, we adopt the effective fine-tuning technique to classify strokes. The procedure mainly includes (i) replacing the final "classifier" layer of a pre-trained neural network with a novel randomly initialized layer for the target task of interest, and (ii) adding novel units in front of the final "classifier" layer to develop the network's capacity which facilitates knowledge transfer to new tasks. The "developmental" network is then fine-tuned with additional passes of appropriately tuned gradient descent on the target training data. Figure 1 illustrates this procedure.

To the best of our knowledge, this is the first study introducing transfer learning with CNNs to the sketch segmentation and labeling task. In the technical phase, our main contributions are composed of three aspects:

(i) *Input representation* To classify strokes correctly, the input representation of strokes should involve both shape and position information. However, the position information of individual strokes is ambiguous. Therefore, we propose to explicitly fuse their host sketches with strokes to construct informative inputs for CNNs.

(ii) *Network structure* To construct an effective developmental network for our task, we propose to add grouped filter layers to the pre-trained VGGNet16 [41]. Compared with fully connected filters, grouped filters have been proven to learn better representations because of their sparse group relationships [12, 48, 54].

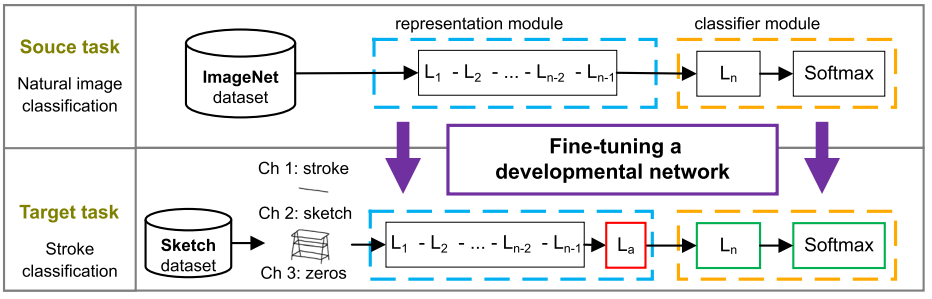


Fig. 1 Developing a pre-trained neural network VGGNet16, and then fine-tuning it on the target task (i.e., stroke image classification for sketch segmentation and labeling) with few examples. The network is pre-trained on the source task (i.e., ImageNet classification) with abundant data. To develop the network, we add a new layer L_a on the top of layer L_{n-1}

(iii) Extensive experimental results on the two benchmarks [10, 39] verify our input representation and network structure, and outperform those of the state-of-arts [10, 22, 39].

The rest of this paper is structured as follows. Section 2 discusses related work. Section 3 introduces our novel input representation and network structure. Section 4 and Section 5 provide the experimental results and conclusions, respectively.

2 Related work

Sketch segmentation and labeling Our work is highly related to the three papers [10, 22, 39]. The early paper [10] first proposed and addressed the sketch segmentation and labeling task, and achieved impressive results. Their idea is transferring part segmentations and labels from a 3D model database to sketches. Manual viewpoint tuning and an additional labeled 3D model database are necessary to obtain optimal results. Therefore, their method is not available to the abstract sketches that do not have corresponding 3D models. To address these disadvantages, Schneider and Tuytelaars [39] adapted a conditional random field model [15] to automatic segmenting sketches without dependence of 3D model databases. Their unary classifier adopts Fisher Vectors [34] to encode the handcrafted feature (dense SIFT [25], and spatial coordinates). However, their method may be sensitive to various strokes due to the handcrafted feature's limited descriptive ability. Our feature, in contrast, learned by CNN, is able to encode generic features. Li et al. [22] recently proposed a fully convolutional neural network to process rasterized sketches, and then leveraged strokes' information to improve results with a graph-cut model [23]. To solve the challenge of few samples, their training data are acquired by rendering corresponding 3D models with labels so that their method can not work on abstract sketches without corresponding 3D models. Instead of increasing sketch samples, our method leverages a large number of existing natural image samples (e.g., the ImageNet dataset [32]), and is available with few sketch samples.

Our work is related to interactive selecting strokes [27], object-level sketch instance segmentation in sketch scenes [43], predicting whether strokes belong to the same group rather than what group [29, 30], stroke instance segmentation for rasterized sketch vectorization [13], fully automatic parsing of rasterized sketches [38], sketch-based 3D modeling [4, 18, 49, 50, 58], and image pre-processing [59, 60].

Transfer learning with CNNs When CNNs are trained on a sufficiently diverse and large dataset, their features can transfer across a large range of novel tasks [26, 52]. And fine-tuning is an effective strategy for transfer learning with CNNs [5, 6, 28, 51]. The pioneering work [8] proposed the fine-tuning technique to transfer knowledge from a generative model to a discriminative one. The works on fine-tuning CNNs can be classified into two categories: the one that adopts a fixed capacity model [1, 11, 31, 56], and the other one that constructs an increasing capacity model [28, 33, 47] by introducing new units to facilitate transfer. Our method belongs to the latter category. Different from the previous works [28, 47], we first propose to add grouped filter layers rather than fully connected layers to improve our model's capacity.

Our work is related to cross-domain semantic segmentation by leveraging domain adaptation [9, 20, 21, 37, 55]. To address domain shift problems between real data and synthetic data, different techniques were proposed, such as curriculum style learning [55], adversarial training [9, 37], etc.

3 Our method

As shown in Fig. 1, the pipeline of our method consists of three steps. (i) Firstly, we pre-train a neural network VGGNet16 on ImageNet [32] dataset, which can initialize the network with powerful feature extracting ability. (ii) Secondly, we increase the capacity of the pre-trained VGGNet16 by adding a new layer L_a on the top of layer L_{n-1}. Section 3.2 provides more technical details. (iii) Thirdly, we fine-tune the developmental VGGNet16 on the sketch datasets. To involve shape and position information, we fuse stroke and sketch as inputs. Please see Section 3.1 for the input details.

3.1 Input representation

A sketch $\mathcal{S}$ is composed of a sequence of strokes $\{s_i\}_{i=1}^{m}$, where m is the total number of strokes, and each stroke s_i is formed by a sequence of points. Each point only consists of x and y coordinates. This storage format is commonly applied for sketches in existing sketch datasets [3, 10, 39].

The format of our input to the pre-trained network (e.g., VGGNet16 [41] in this paper) must be a fixed-size three-channel image, $I \in \mathbb{R}^{width \times height \times channel}$, where $width = 224$, $height = 224$, and $channel = 3$. Because the network is pre-trained on the natural color image dataset, e.g., ImageNet [32]. To construct this type of input, for a stroke s_i, we propose to rasterize it and its host sketch $\mathcal{S}$ together. Specifically, the stroke s_i is rasterized and stored in the first channel, and its host sketch $\mathcal{S}$ is rasterized and stored in the second channel. The third channel is filled with zero values since this channel is not used. The second row in Fig. 1 illustrates each channel ("Ch 1 to 3").

We conduct this rasterization to consider two factors: shape (i.e., the first channel) and position (i.e., the first two channels). Some strokes have obvious shape differences (e.g., the airplane's wings, windows and body, in Fig. 2a), and some strokes have strong position relations with their host sketch (e.g., the table's top and midsupport, in Fig. 2b). Both the two factors facilitate to recognize a stroke's category.

3.2 Fine-tuning a developmental network

3.2.1 Fine-tuning and VGGNet16

Fine-tuning is the essential technique to transfer a pre-trained CNN to a novel target task with a small amount of target training data. The CNN is pre-trained with abundant data from a source domain, e.g., the ImageNet dataset [32] for natural image classification. The CNN can be defined as a sequence of two modules: a feature representation module $\mathcal{F}_S$ and

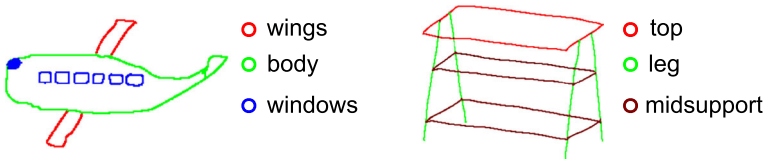


Fig. 2 Shape and position factors. **a** strokes wings, windows and body have obvious shape differences; **b** strokes top and midsupport have strong position relations with their host sketch

a classifier module $\mathcal{C}_S$, shown as in Fig. 1 (1st row). The module $\mathcal{F}_S$ contains the layers L_1 to L_{n-1}, while the module $\mathcal{C}_S$ contains the final layer L_n and the softmax, where n is the total number of weight layers. More specifically, for the pre-trained VGGNet16, its module $\mathcal{F}_S$ consists of 15 weight layers and can be expressed as follows:

$$\begin{cases} F_0 = I & , I \in \mathbb{R}^{224\times224\times3} \\ F_j = \sigma(\mathbf{W}_j * F_{j-1} + \mathbf{b}_j) & , j \in \{1, 3, 5, 6, 8, 9, 11, 12\} \\ F_j = maxpool(\,\sigma(\mathbf{W}_j * F_{j-1} + \mathbf{b}_j)\,) & , j \in \{2, 4, 7, 10, 13\} \\ F_j = \sigma(\mathbf{W}_j F_{j-1} + \mathbf{b}_j) & , j \in \{14, 15\} \end{cases} \tag{1}$$

where F_0 is the input I, $*$ represents convolution operation, σ is the non-linear function ReLU, F_j represents the feature extracted from F_{j-1}, *maxpool* is max pooling operation, and $\mathbf{W}_j$ and $\mathbf{b}_j$ stand for the weight and bias in the j_{th} layer, respectively. For the case $j = 15$, F_{15} is the feature outputted by $\mathcal{F}_S$. The module $\mathcal{C}_S$ of VGGNet16 can be written as follows:

$$P_S = softmax(\mathbf{W}_{16}F_{15} + \mathbf{b}_{16}) \tag{2}$$

where *softmax* is softmax operation, $P_S \in \mathbb{R}^{N_S}$ represents a probability vector outputted by $\mathcal{C}_S$, and N_S is the number of category in the source domain.

In classic fine-tuning, the target CNN's feature representation module $\mathcal{F}_T$ reserves $\mathcal{F}_S$'s structure and parameters $\Theta_T^{\mathcal{F}} = \Theta_S^{\mathcal{F}}$ (e.g., $\{\mathbf{W}_j, \mathbf{b}_j\}_{j=1}^{15}$), while its classifier module $\mathcal{C}_T$ is a new one with parameters $\Theta_T^{\mathcal{C}}$ randomly initialized. A partial of (or all) the parameters $\Theta_T^{\mathcal{F}}$ and $\Theta_T^{\mathcal{C}}$ are fine-tuned by continuing the backpropagation. Since $\mathcal{F}_T$ and $\mathcal{F}_S$ own the same network structure, the representational capacity is not increased during transfer.

A recent study suggests that fine-tuning a developmental network is superior to fine-tuning a fixed network during transfer learning [47]. A developmental network consists of two modules: a new representation module $\hat{\mathcal{F}}_T = \mathcal{F}_T \cup \{u_k\}_{k=1}^{t}$, where $\{u_k\}_{k=1}^{t}$ is t newly added units, and the classifier module $\mathcal{C}_T$. Notionally, new units can be added into a pre-trained network in multiple ways. Section 3.2.2 describes our way to construct the developmental network for the stroke classification task.

3.2.2 Our developmental network

In the module $\mathcal{F}_T$, its beginning layers tend to represent universal features, while its ending layers tend to encode task-related features [26, 52]. Inspired by this, we put new units at ending layers. Particularly, we construct a completely new top layer L_a, leading to the developmental representation module $\hat{\mathcal{F}}_T$, as shown in Fig. 1 (second row). The layer L_a can play an adapting role that takes into account novel compositions of pre-existing units, thereby avoiding a large number of modifications to the pre-trained layers for their adaptation to the new target task.

For our task, the layer L_a is a sequential block: (a) dropout with probability $= 0.5$, (b) grouped convolutional filters with non-linear function ReLU, and (c) dropout with probability $= 0.5$. Figure 3 illustrates an instance of L_a's computational flow. This structure leverages the advantages of dropout and filter groups. For example, dropout is a regularization technique for reducing over-fitting in neural networks by preventing complex co-adaptations on training data [42]. And grouped filters, although initially designed for network subdivision in multi-GPU training [14], are now applied to learn better representations because of their sparse group relationships [12, 48, 54].

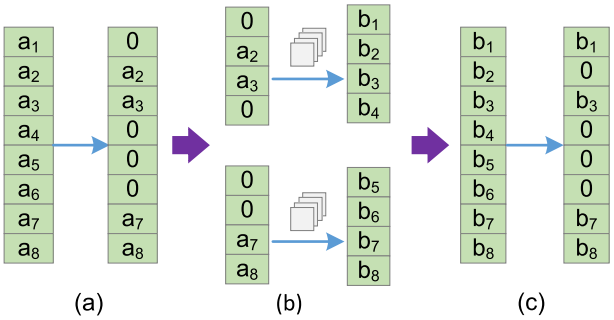


Fig. 3 An example about L_a's computational flow. **a** & **c** dropout with probability = 0.5; **b** two filter groups in which each group has four convolutional kernels

Mathematically, for the pre-trained VGGNet16, given the feature F_{15}, the feature F_a outputted by the developmental module $\hat{\mathcal{F}}_T$ can be formulated as follows:

$$F_a = \mathbf{W}_{P_2} \bullet \sigma(\ (\mathbf{W}_G \hat{*} (\mathbf{W}_{P_1} \bullet F_{15}) + \mathbf{b}_G\ ) \tag{3}$$

where $\bullet$ refers to Hadamard product operation, $\hat{*}$ refers to grouped convolution operation, $\mathbf{W}_{P_1}$ and $\mathbf{W}_{P_2}$ represent the 01 vectors corresponding to dropout, and $\mathbf{W}_G$ and $\mathbf{b}_G$ denote the weight and bias of grouped convolutional filters. For the classifier module $\mathcal{C}_T$, it can be written as follows:

$$P_T = softmax(\mathbf{W}_{17} F_a + \mathbf{b}_{17}) \tag{4}$$

where $P_T \in \mathbb{R}^{N_T}$ represents a probability vector outputted by $\mathcal{C}_T$, and N_T is the number of category in the target domain.

4 Experimental results

Datasets We evaluate our method on two public benchmark datasets: Huang dataset [10] and Schneider dataset [39] (a subset of TU-Berlin dataset [3])[1] . Huang dataset consists of 10 object categories and each category has 30 sketches, while Schneider dataset consists of 6 object categories and each category has 20 sketches. The two datasets represent two different types of sketches respectively. The sketches in Huang dataset resemble real-world 3D objects somewhat, while those in Schneider dataset are more abstract. To reduce overfitting, we apply data augmentation by rotation and horizontal mirror. Specifically, we rotate strokes by ± 5, ± 10, ± 15, ± 20 and ± 25 degrees, and then flip the rotated strokes and the unrotated strokes horizontally.

Metrics Like prior works [10, 39], two metrics are used to evaluate sketch segmentation and labeling: *pixel metric*, the percentage of pixels that are assigned with correct labels; *component metric*, the percentage of strokes that are assigned with correct labels. The pixel metric has a bias towards large components that have more pixels, and the component metric gives a bias to small components that are of larger numbers.

Implementation Our experimental platform is a PC with an Intel Core I7-4790k 4.0 GHz CPU, and an Nvidia GTX1080ti GPU with 11GB memory. For a fair comparison, we choose the VGGNet [41] with 16 weight layers, pre-trained on ImageNet dataset [32], to

[1]The two datasets are available on the ACM digital library website.

construct our developmental network. Our layer L_a adopts 256 units and 16 convolutional filter groups in which each group has $16 = 256 \div 16$ filter kernels. The network is implemented with Matconvnet library on Matlab framework. We use stochastic gradient descent (SGD) to perform fine-tuning for 10 epochs. The new layers L_a and L_n are fine-tuned at a learning rate of 0.01, which is ten times larger than that of the pre-trained layers L_1 to L_{n-1}.

4.1 Evaluation on Huang dataset

Firstly, we validate the effects of our input representation and network structure, respectively. We perform 3-fold cross validation. The 30 sketches in each category are splitted into 3 equal subsets. In each run, a single subset (10 sketches) is used as testing data, while the other two subsets (20 sketches) are used as training data for fine-tuning our network. This is repeated for all 3 folds.

As shown in Tables 1 and 2, the best results are in bold. Compared with only inputting individual strokes (i.e., "w/o fusion"), our input representation, the input fusion of stroke and sketch, makes significant effects, leading to average pixel accuracy 4.8% improvement (82.0% vs. 77.2%) and average component accuracy 6.5% improvement (83.3% vs. 76.8%). For every category except *airplane*, the accuracies produced by our input representation are higher than those produced by "w/o fusion". In addition, we compared our input fusion with the feature level fusion, the concatenation of deep features of a stroke and a sketch before the classifier module (see Fig. 4). On average, our input fusion's performance is are slightly superior to that of the feature level fusion, e.g., 82.0% vs. 80.1%.

To validate our network structure, we compare with other two alternative structures of L_a: (1) "w/o L_a" denotes the network with a fixed capacity, (2) "L_a=FC" denotes replacing our grouped filters with fully connected filters. And "L_a=GC" denotes ours. Note Table 2, both the average component accuracy of all categories and individual component accuracies of every category produced by our "L_a=GC" are higher than those produced by "w/o L_a" and "L_a=FC". However, in Table 1, our pixel accuracies of *airplane* and *rifle* are slightly lower than those produced "w/o L_a". This may be caused by the reason: "w/o L_a" classifies correctly more long strokes than "L_a=GC" does in category *airplane* and *rifle*.

Table 1 Ablation study of fusion of stroke and sketch, network structure and training size on Huang dataset, using pixel accuracy(%)

Category	Fusion of stroke & sketch			Network structure			Training size	
	w/o fusion	feature fusion	input fusion	w/o La	La=FC	La=GC	3-fold	5-fold
Airplane	80.2	**80.2**	79.8	**80.3**	79.6	79.8	79.8	**85.8**
Bicycle	80.9	82.4	**89.7**	85.9	87.1	**89.7**	**89.7**	89.3
Cdlbrm	68.7	71.9	**75.1**	72.3	73.7	**75.1**	75.1	**76.9**
Chair	66.6	70.4	**72.0**	68.4	65.0	**72.0**	72.0	**72.5**
Fourleg	83.4	84.2	**85.5**	80.6	83.3	**85.5**	85.5	**87.6**
Human	74.3	78.0	**81.6**	78.4	80.1	**81.6**	81.6	**84.6**
Lamp	89.4	92.0	**96.9**	93.2	93.4	**96.9**	96.9	**98.6**
Rifle	62.4	67.9	**69.2**	**70.9**	69.2	69.2	69.2	**69.3**
Table	76.5	**82.9**	81.1	77.4	79.2	**81.1**	**81.1**	80.5
Vase	89.0	**91.1**	89.8	86.7	87.0	**89.8**	**89.8**	88.8
Average	77.2	80.1	**82.0**	79.4	79.7	**82.0**	82.0	**83.4**

Table 2 Ablation study of fusion of stroke and sketch, network structure and training size on Huang dataset, using component accuracy (%)

Category	Fusion of stroke & sketch			Network structure			Training size	
	w/o fusion	feature fusion	input fusion	w/o La	La=FC	La=GC	3-fold	5-fold
Airplane	76.2	**77.2**	75.3	72.1	72.9	**75.3**	75.3	**82.8**
Bicycle	73.3	75.2	**86.5**	79.6	82.9	**86.5**	86.5	**87.0**
Cdlbrm	72.3	75.1	**80.4**	75.8	73.8	**80.4**	80.4	**83.1**
Chair	65.7	70.1	**72.4**	68.5	66.1	**72.4**	72.4	**74.5**
Fourleg	84.8	86.4	**88.4**	80.6	83.3	**88.4**	88.4	**90.6**
Human	77.2	80.3	**86.7**	81.7	84.8	**86.7**	86.7	**90.0**
Lamp	87.7	90.1	**98.2**	92.8	92.6	**98.2**	98.2	**99.0**
Rifle	65.1	71.5	**77.0**	74.1	75.8	**77.0**	77.0	**78.5**
Table	73.8	**79.7**	76.7	73.0	75.7	**76.7**	76.7	**77.1**
Vase	91.6	**93.1**	91.7	89.2	89.4	**91.7**	**91.7**	90.8
Average	76.8	79.9	**83.3**	78.7	79.7	**83.3**	83.3	**85.3**

Secondly, we perform a 5-fold cross validation to evaluate the effect of larger training size. Specifically, in each run, four subsets (24 sketches) are used as training data in each run, while the other subset (6 sketches) are used as testing data. We repeat this for all 5 folds. Compared with 3-fold cross validation, the average result about pixel metric improves 1.4% (83.4% vs. 82.0% in Table 1), while that about component metric increases 2% (85.3% vs. 83.3% in Table 2). This experiment demonstrates that having more training samples is helpful for our method. Figure 5 shows our representative results in which their pixel accuracies are above 90%.

Thirdly, we compare with the state-of-the-art results [10, 22, 39]. Schneider et al. [39] performed 3-fold cross validation, while Huang et al. [10] and Li et al. [22] adopted additional labeled 3D mesh data without cross validation. The labels of 3D meshes are the same as those of sketches, and the number of 3D meshes is larger than 20. In other words, their [10, 22] training size is larger than that of 3-fold cross validation. To make a fair comparison (Table 3), we compare our results yielded by 3-fold cross validation (Ours-3) with Schneider [39], and we compare our results yielded by 5-fold cross validation (Ours-5) with Huang [10] and Li [22]. Note that the pixel accuracy is more reliable than the component accuracy, because their [10, 22, 39] component accuracy is the percentage of correctly labeled segmented strokes. In contrast, our component accuracy is the percentage

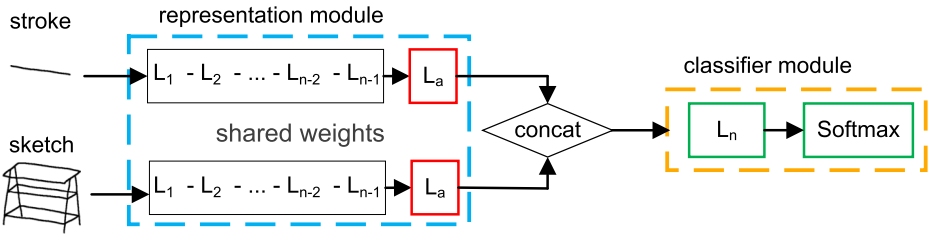


Fig. 4 The feature level fusion of stroke and sketch. For reading convenience, the channels filled with zero values are not illustrated

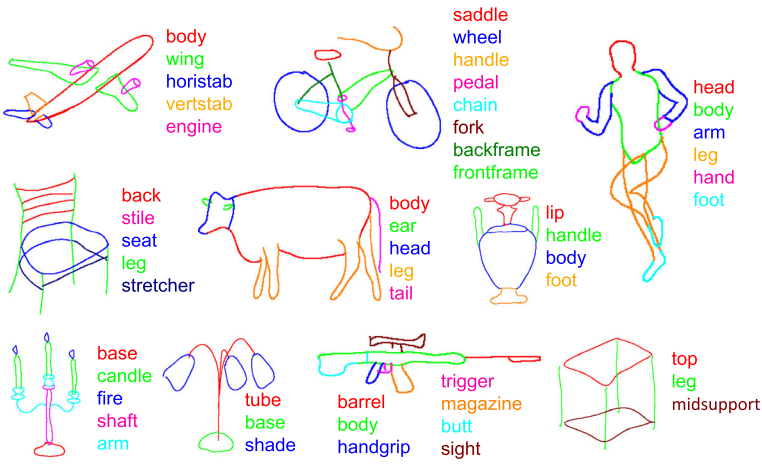


Fig. 5 The representative results on Huang dataset with our method

of correctly labeled unsegmented strokes. Table 3 shows that our average pixel accuracy is 8.7% higher than Schneider [39] (e.g., 82.0% vs. 73.3%) and 2% higher than that of the state-of-art [22] (e.g., 83.4% vs. 81.4%). In Table 3, the best results are in bold.

4.2 Evaluation on Schneider dataset

We perform 4-fold cross validation experiments on Schneider dataset, to be consistent with data splitting as Schneider et al.[39]. Due to these abstract sketches having no corresponding 3D meshes, Huang et al. [10] and Li et al. [22]'s methods are not available. Schneider dataset contains clean and unclean data. The unclean data means that some strokes belong to unseen labels. Note that our training procedures are performed on clean data.

Table 3 Comparison with state-of-the-art results (i.e., Schneider [39], Huang [10], and Li [22]) on Huang dataset

Category	Pixel accuracy (%)					Component accuracy (%)				
	Schneider	Ours-3	Huang	Li	Ours-5	Schneider	Ours-3	Huang	Li	Ours-5
Airplane	55.1	**79.8**	82.4	85.7	**85.8**	48.7	75.3	66.2	76.9	**82.8**
Bicycle	79.7	**89.7**	78.2	86.2	**89.3**	68.6	**86.5**	66.4	77.1	**87.0**
Cdlbrm	72.0	**75.1**	72.7	**78.3**	76.9	66.2	**80.4**	56.7	69.9	**83.1**
Chair	66.5	**72.0**	**76.5**	75.5	72.5	61.6	**72.4**	63.1	70.3	**74.5**
Fourleg	81.5	**85.5**	80.2	84.9	**87.6**	74.2	**88.4**	67.2	75.5	**90.6**
Human	69.7	**81.6**	79.1	80.5	**84.6**	63.1	**86.7**	64.0	72.8	**90.0**
Lamp	82.9	**96.9**	92.1	87.8	**98.6**	77.2	**98.2**	89.3	83.8	**99.0**
Rifle	67.8	**69.2**	**75.9**	70.6	69.3	65.1	**77.0**	62.2	65.7	**78.5**
Table	74.5	**81.1**	79.1	**81.4**	80.5	65.6	76.7	69.0	**77.3**	77.1
Vase	83.3	**89.8**	71.9	82.7	**88.8**	79.1	**91.7**	63.1	78.0	**90.8**
Average	73.3	**82.0**	78.8	81.4	**83.4**	66.9	**83.3**	66.7	74.7	**85.3**

Table 4 Ablation study of fusion of stroke and sketch, and network structure on Schneider dataset

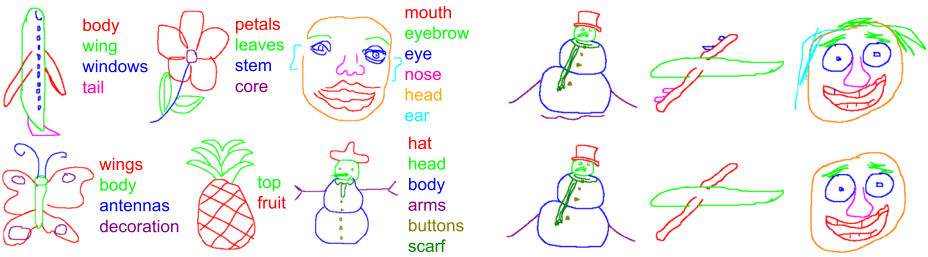
Category	Pixel accuracy (%)						Component accuracy (%)					
	Fusion of stroke & sketch			Network structure			Fusion of stroke & sketch			Network structure		
	w/o fusion	feature fusion	input fusion	w/o La	La=FC	La=GC	w/o fusion	feature fusion	input fusion	w/o La	La=FC	La=GC
Airplane	85.9	**89.2**	89.0	**90.5**	89.4	89.0	83.4	86.2	**88.3**	**89.1**	86.1	88.3
Butterfly	85.1	88.3	**95.8**	93.0	88.2	**95.8**	84.0	87.2	**93.1**	90.2	86.5	**93.1**
Face	91.1	92.8	**97.3**	89.0	92.1	**97.3**	84.0	89.9	**92.3**	81.6	86.8	**92.3**
Flower	83.8	81.7	**86.8**	81.1	85.8	**86.8**	85.3	84.0	**87.6**	78.0	83.7	**87.6**
Pineapple	98.7	98.8	**99.0**	96.3	97.1	**99.0**	98.2	**99.1**	98.9	95.7	97.0	**98.9**
Snowman	90.5	93.1	**95.1**	88.8	91.8	**95.1**	82.0	89.0	**96.7**	86.7	90.1	**96.7**
Average	89.2	90.6	**93.8**	89.8	90.7	**93.8**	86.1	89.2	**92.8**	86.9	88.3	**92.8**

Table 5 Comparison with state-of-the-art results, (i.e., Schneider [39]), on Schneider dataset

Category	Pixel accuracy (%) tested on clean data		Component accuracy (%) tested on clean data		Pixel accuracy (%) tested on unclean data	
	Schneider	Ours	Schneider	Ours	Schneider	Ours
Airplane	77.6	**89.0**	78.3	**88.3**	74.5	**87.9**
Butterfly	78.8	**95.8**	78.6	**93.1**	78.8	**92.8**
Face	88.3	**97.3**	84.2	**92.3**	75.6	**94.0**
Flower	78.3	**86.8**	73.7	**87.6**	77.8	**82.4**
Pineapple	96.2	**99.0**	96.0	**98.9**	96.2	**98.9**
Snowman	85.3	**95.1**	81.8	**96.7**	80.0	**90.1**
Average	84.1	**93.8**	82.1	**92.8**	80.5	**91.0**

Firstly, we perform testing on the clean data. Table 4 reports the effects of our input fusion and network structure. On average, the mean accuracy of our input fusion is 3% higher than that of the feature level fusion, and the feature level fusion gets $1 \sim 3\%$ average improvement than the "w/o fusion" does. And the structure with grouped filters ("L_a=GC") obtains higher mean accuracy than the structures "w/o L_a" and "L_a=FC" do. Compared with start-of-the-art results [39] (Schneider in Table 5), our average pixel average accuracy is 9.7% higher than that of Schneider, and our accuracies of all categories are also higher than theirs. Figure 6a shows our representative results in which their pixel accuracies are above 90%. In Table 4 and 5, the best results are in bold.

Secondly, we perform testing on the unclean data to evaluate our method in real-world environments. Though strokes with unseen labels are always classified with wrong labels, we still produce good results for strokes with known labels. Specifically, Fig. 6b shows our three segmented unclean sketches. The strokes with unknown labels (e.g., the floor under the *snowman*, the turbines on the *airplane*, and the hairs on the *face*) are misclassified, but the other strokes are assigned with right labels. Note the final column in Table 5, the overall accuracy is satisfactory. Our average pixel accuracy is 91.0%, which is 10.5% higher than that of Schneider [39]. Our gap between clean and unclean data is 2.8%, while that of Schneider is 3.6%.



(a) tested on clean data. (b) tested on unclean data.

Fig. 6 **a** The representative results on Schneider dataset with our method. **b** Results on unclean data with our method (first row), ground truth on clean data (second row)

5 Conclusion

We have explored transfer learning based on CNN by fine-tuning a pre-trained VGGNet16 to classify strokes for sketch segmentation. We proposed a novel input representation, the fusion of strokes and their host sketches, to make shape and position informative. To construct our developmental network for fine-tuning during transfer learning, we proposed to add grouped filter layers instead of fully connected filter layers. Because grouped filters have been proven to encode better feature representations. Extensive experimental results on two benchmarks verify our method and outperform those of the state-of-arts [10, 22, 39].

Acknowledgments The work is supported by the National Key R&D Program of China (2018YFB0203904), NSFC from PRC (61872137, 61502158, 61502157, 61472131, 61772191), Hunan NSF (2017JJ3042), and Science and Technology Key Projects of Hunan Province (2015TP1004, 2015SK2087, 2015JC1001, 2016JC2012).

References

1. Chu B, Madhavan V, Beijbom O, Hoffman J, Darrell T (2016) Best practices for fine-tuning visual classifiers to new domains. In: ECCV Workshops, pp 435–442
2. Ding Z, Fu Y (2018) Deep transfer low-rank coding for cross-domain learning. IEEE Trans Neural Netw Learn Syst 30(6):1768–1779
3. Eitz M, Hays J, Alexa M (2012) How do humans sketch objects? ACM Trans Graph 31(4):44:1–44:10
4. Fan L, Wang R, Xu L, Deng J, Liu L (2013) Modeling by drawing with shadow guidance. Comput Graph Forum 32(7):157–166
5. Girshick RB, Donahue J, Darrell T, Malik J (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. In: CVPR, pp 580–587
6. Hariharan B, Arbeláez PA, Girshick RB, Malik J (2015) Hypercolumns for object segmentation and fine-grained localization. In: CVPR, pp 447–456
7. He JY, Wu X, Jiang YG, Zhao B, Peng Q (2017) Sketch recognition with deep visual-sequential fusion model. In: ACM Multimedia. ACM, pp 448–456
8. Hinton GE, Salakhutdinov R (2006) Reducing the dimensionality of data with neural networks. Science 313(5786):504–507
9. Hoffman J, Tzeng E, Park T, Zhu J, Isola P, Saenko K, Efros AA, Darrell T (2018) Cycada: cycle-consistent adversarial domain adaptation. In: ICML, pp 1994–2003
10. Huang Z, Fu H, Lau RW (2014) Data-driven segmentation and labeling of freehand sketches. ACM Trans Graph 33(6):175:1–175:10
11. Huh M, Agrawal P, Efros AA (2016) What makes imagenet good for transfer learning?. arXiv: 1608.08614
12. Ioannou Y, Robertson D, Cipolla R, Criminisi A (2017) Deep roots: improving cnn efficiency with hierarchical filter groups. In: CVPR, pp 1231–1240
13. Kim B, Wang O, Öztireli AC, Gross M (2018) Semantic segmentation for line drawing vectorization using neural networks. Comput Graph Forum 37(2):329–338
14. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: NIPS, pp 1097–1105
15. Lafferty J, McCallum A, Pereira F (2001) Conditional random fields: probabilistic models for segmenting and labeling sequence data. In: ICML. Morgan Kaufmann, San Francisco, pp 282–289
16. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324
17. Li B, Lu Y, Johan H, Fares R (2017) Sketch-based 3d model retrieval utilizing adaptive view clustering and semantic information. Multimed Tool Appl 76(24):26603–26631
18. Li C, Pan H, Liu Y, Tong X, Sheffer A, Wang W (2018) Robust flow-guided neural prediction for sketch-based freeform surface modeling. ACM Trans Graph 37(6):238:1–238:12
19. Li J, Lu K, Huang Z, Zhu L, Shen HT (2018) Transfer independently together: A generalized framework for domain adaptation. IEEE Trans Cybern 49(6):2144–2155
20. Li J, Jing M, Lu K, Zhu L, Shen HT (2019) Locality preserving joint transfer for domain adaptation. IEEE Trans Image Process 28(12):6103–6115

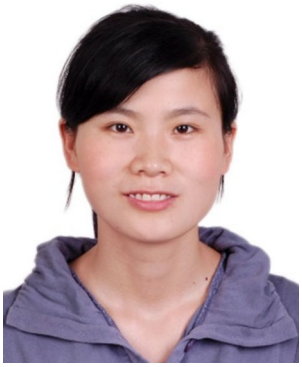
21. Li J, Lu K, Huang Z, Zhu L, Shen HT (2019) Heterogeneous domain adaptation through progressive alignment. IEEE Trans Neural Netw Learning Syst 30(5):1381–1391
22. Li L, Fu H, Tai CL (2019) Fast sketch segmentation and labeling with deep learning. IEEE Comput Graph Appl 39(2):38–51
23. Li SZ (1994) Markov random field models in computer vision. In: ECCV. Springer, pp 361–370
24. Li Y, Lei H, Lin S, Luo G (2018) A new sketch-based 3d model retrieval method by using composite features. Multimed Tool Appl 77(2):2921–2944
25. Lowe DG (1999) Object recognition from local scale-invariant features. In: ICCV. IEEE, pp 1150–1157
26. Mou L, Meng Z, Yan R, Li G, Xu Y, Zhang L, Jin Z (2016) How transferable are neural networks in nlp applications? In: EMNLP, pp 479–489
27. Noris G, Sỳkora D, Shamir A, Coros S, Whited B, Simmons M, Hornung A, Gross M, Sumner R (2012) Smart scribbles for sketch segmentation. Comput Graph Forum 31(8):2516–2527
28. Oquab M, Bottou L, Laptev I, Sivic J (2014) Learning and transferring mid-level image representations using convolutional neural networks. In: CVPR, pp 1717–1724
29. Qi Y, Guo J, Li Y, Zhang H, Xiang T, Song YZ (2013) Sketching by perceptual grouping. In: ICIP. IEEE, pp 270–274
30. Qi Y, Song YZ, Xiang T, Zhang H, Hospedales T, Li Y, Guo J (2015) Making better use of edges via perceptual grouping. In: CVPR, pp 1856–1865
31. Razavian AS, Azizpour H, Sullivan J, Carlsson S (2014) CNN features off-the-shelf: an astounding baseline for recognition. In: CVPR Workshops, pp 512–519
32. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M et al (2015) Imagenet large scale visual recognition challenge. Int J Comput Vis 115(3):211–252
33. Rusu AA, Rabinowitz NC, Desjardins G, Soyer H, Kirkpatrick J, Kavukcuoglu K, Pascanu R, Hadsell R (2016) Progressive neural networks. arXiv: 1606.04671
34. Sánchez J, Perronnin F, Mensink T, Verbeek J (2013) Image classification with the fisher vector: theory and practice. Int J Comput Vis 105(3):222–245
35. Sangkloy P, Burnell N, Ham C, Hays J (2016) The sketchy database: learning to retrieve badly drawn bunnies. ACM Transactions on Graphics (TOG) 35(4):119
36. Sangkloy P, Lu J, Fang C, Yu F, Hays J (2017) Scribbler: controlling deep image synthesis with sketch and color. In: CVPR, pp 5400–5409
37. Sankaranarayanan S, Balaji Y, Jain A, Lim S, Chellappa R (2018) Learning from synthetic data: addressing domain shift for semantic segmentation. In: CVPR, pp 3752–3761
38. Sarvadevabhatla RK, Dwivedi I, Biswas A, Manocha S et al (2017) Sketchparse: towards rich descriptions for poorly drawn sketches using multi-task hierarchical deep networks. In: ACM Multimedia. ACM, pp 10–18
39. Schneider RG, Tuytelaars T (2016) Example-based sketch segmentation and labeling using crfs. ACM Trans Graph 35(5):151:1–151:9
40. Seddati O, Dupont S, Mahmoudi S (2017) Deepsketch 3. Multimed Tool Appl 76(21):22333–22359
41. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. In: ICLR
42. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: wa simple way to prevent neural networks from overfitting. J Mach Learn Res 15(1):1929–1958
43. Sun Z, Wang C, Zhang L, Zhang L (2012) Free hand-drawn sketch segmentation. In: ECCV. Springer, pp 626–639
44. Tan G, Chen H, Qi J (2016) A novel image matting method using sparse manual clicks. Multimed Tool Appl 75(17):10213–10225
45. Wan L, Xiao Y, Dou N, Leung CS, Lai YK (2018) Scribble-based gradient mesh recoloring. Multimed Tool Appl 77(11):13753–13771
46. Wang W, Chen Z, Liu J, Qi Q, Zhao Z (2012) User-based collaborative filtering on cross domain by tag transfer learning. In: Proceedings of the 1st International Workshop on Cross Domain Knowledge Discovery in Web and Social Network Mining. ACM, pp 10–17
47. Wang YX, Ramanan D, Hebert M (2017) Growing a brain: fine-tuning by increasing model capacity. In: CVPR, pp 2471–2480
48. Xie G, Wang J, Zhang T, Lai J, Hong R, Qi GJ (2018) Interleaved structured sparse convolutional neural networks. In: CVPR, pp 8847–8856
49. Xu B, Chang W, Sheffer A, Bousseau A, McCrae J, Singh K (2014) True2form: 3d curve networks from 2d sketches via selective regularization. ACM Trans Graph 33(4):1–13
50. Xu K, Chen K, Fu H, Sun WL, Hu SM (2013) Sketch2scene: sketch-based co-retrieval and co-placement of 3d models. ACM Transactions on Graphics (TOG) 32(4):123:1–123:15
51. Yang S, Ramanan D (2015) Multi-scale recognition with dag-cnns. In: ICCV, pp 1215–1223

52. Yosinski J, Clune J, Bengio Y, Lipson H (2014) How transferable are features in deep neural networks? In: NIPS, pp 3320–3328
53. Yu Q, Yang Y, Liu F, Song YZ, Xiang T, Hospedales TM (2017) Sketch-a-net: a deep neural network that beats humans. Int J Comput Vis 122(3):411–425
54. Zhang T, Qi GJ, Xiao B, Wang J (2017) Interleaved group convolutions. In: ICCV. IEEE, pp 4383–4392
55. Zhang Y, David P, Gong B (2017) Curriculum domain adaptation for semantic segmentation of urban scenes. In: ICCV, pp 2039–2049
56. Zheng L, Zhao Y, Wang S, Wang J, Tian Q, 2016 Good practice in CNN feature transfer. arXiv: 1604.00133
57. Zhou S, Zhou C, Xiao Y, Tan G (2018) Patchswapper: a novel real-time single-image editing technique by region-swapping. Comput Graph 73:80–87
58. Tan G, Zhu X, Liu X (2017) A free shape 3d modeling system for creative design based on modified catmull-clark subdivision. Multimedia Tools Appl 76(5):6429–6446
59. Zheng Y, Cao X, Xiao Y, Zhu X, Yuan J (2019) Joint residual pyramid for joint image super-resolution. J. Visual Communication and Image Representation 58:53–62
60. Tan G, Zhang Q, Zhu X, Hu H, Wu X (2020) Fingerprint Liveness Detection based on Guided Filtering and Hybrid Image Analysis. IET Image Processing 1–7. https://doi.org/10.1007/s00158-007-0093-7

Xianyi Zhu is currently a PhD. student at the College of Computer Science and Electronic Engineering, Hunan University, China. He received his Master's degree from Hunan University in 2016. His research interests include computer vision and computer graphics.

Jin Yuan received the Bachelor's degree and Master's degree from Fudan University in 2005 and 2008, and the PhD. degree in Electronic Engineering from National University of Singapore in 2012. He is currently an assistant professor in the college of Computer Science and Electronic Engineering, Hunan University, China. His research interests include multimedia retrieval and machine learning.

Yi Xiao received the Bachelor's degree and Master's degree in Mathematics from Sichuan University in 2005 and 2008, and the PhD. degree in Electronic Engineering from City University of Hong Kong in 2012. He is currently an Associate Professor in the college of Computer Science and Electronic Engineering, Hunan University, China. His research interests include computer graphics, image processing, GPU computing and neural networks.



Yan Zheng received the Bachelor's degree from Department of Mathematics, Hebei Normal University in 2006, and Master's degree from Department of Mathematics, Sichuan University in 2009, and the PhD. degree in Department of Mechanical and Biomedical Engineering, City University of Hong Kong in 2012. She is currently an Assistant Professor in the college of Electrical and Information Engineering, Hunan University, China. Her research interests include switched system, positive systems and machine learning.

Zheng Qin received the Ph.D. degree in computer software and theory from Chongqing University, China, in 2001. From 2010 to 2011, he served as a Visiting Scholar with the Department of Computer Science, Michigan State University. He is currently a Professor at the College of Computer Science and Electronic Engineering, Hunan University, where he also serves as the Vice Dean. He also serves as the Director of the Hunan Key Laboratory of Big Data Research and Application and the Vice Director of the Hunan Engineering Laboratory of Authentication and Data Security. His main interests are network and data security, privacy, data analytics and applications, machine learning, and applied cryptography.