Coupling Deep Textural and Shape Features for Sketch Recognition

Qi Jia
Dalian University of Technology
jiaqi@dlut.edu.cn

Xin Fan*
Dalian University of Technology
xin.fan@dlut.edu.cn

Meiyu Yu
Didi Chuxing
yumeiyu516@sina.com

Yuqing Liu
Dalian University of Technology
yuqingl@pku.org.cn

Dingrong Wang
Dalian University of Technology
wdrkl986739772@gmail.com

Longin Jan Latecki
Temple University
latecki@temple.edu

ABSTRACT

Recognizing freehand sketches with high arbitrariness is such a great challenge that the automatic recognition rate has reached a ceiling in recent years. In this paper, we explicitly explore the shape properties of sketches, which has almost been neglected before in the context of deep learning, and propose a sequential dual learning strategy that combines both shape and texture features. We devise a two-stage recurrent neural network to balance these two types of features. Our architecture also considers stroke orders of sketches to reduce the intra-class variations of input features. Extensive experiments on the TU-Berlin benchmark set show that our method achieves over 90% recognition rate for the first time on this task, outperforming both humans and state-of-the-art algorithms by over 19 and 7.5 percentage points, respectively. Especially, our approach can distinguish the sketches with similar textures but different shapes more effectively than recent deep networks. Based on the proposed method, we develop an on-line sketch retrieval and imitation application to teach children or adults to draw. The application is available as Sketch.Draw.

CCS CONCEPTS

• **Computing methodologies** → **Object recognition**.

KEYWORDS

sketch classification; shape features; recurrent neural network (RNN).

ACM Reference Format:
Qi Jia, Xin Fan, Meiyu Yu, Yuqing Liu, Dingrong Wang, and Longin Jan Latecki. 2020. Coupling Deep Textural and Shape Features for Sketch Recognition. In *Proceedings of the 28th ACM International Conference on Multimedia (MM '20), October 12–16, 2020, Seattle, WA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3394171.3413810

*Corresponding Author

MM '20, October 12–16, 2020, Seattle, WA, USA

ACM ISBN 978-1-4503-7988-5/20/10…$15.00
https://doi.org/10.1145/3394171.3413810

1 INTRODUCTION

Sketch recognition provides a powerful analysis tool for many computer vision applications [9, 27], such as forensic analysis [16], sketch-based image retrieval [10, 13, 23], and sketch-based 3D model retrieval [32].

Sketches, reflecting chief features of objects, have long been regarded as an effective way for human to communicate ideas. It is extremely challenging to automatically recognize free-hand sketches even for human being due to huge intra-class variations brought by flexible stroke orders and styles. In this work, we address this issue by integrating shape information that is inherent in sketches.

A sketch consists of simple geometric primitives, e.g., line segments and curves, showing evident differences with natural images. Nevertheless, previous works typically directly apply textural features that gain great success in natural images to sketch recognition. Hand-crafted features are first used in sketch recognition [7, 24]. These features highly depend on gradients information of images with rich textures that rarely exist in sketches. Recently, the learning based features generated by convoluted responses on image intensities (textures) have produced significant improvements over the handcrafted ones [39].

Geometric shape cues play an important role in object recognition. Recent neuropsychological studies reveal that humans have a specific brain area to process geometric shape information [38]. Learning models even trained on natural images from ImageNet prefer to categorize objects according to shapes rather than colors [25]. Unfortunately, traditional hand-crafted local shape features [1, 34] cannot be directly immigrated to characterize shapes for sketch recognition since they have a low discriminative ability to intra-class variations. Therefore, it is highly demanded to devise a new learning framework to cooperate shape and texture information of a sketch.

It is also worth noting that the sequential nature of sketch strokes provides additional information. In a recent work [28], Sarvadevabhatla *et al.* build a recurrent network that takes each stroke image as the input for sequential learning, yielding significant improvements on accuracy. However, the arbitrariness of stroke orders evidently alters the network inputs so that intra-class variations significantly increase. Meanwhile, a single stroke can only provide limited shape information. Thus, it is also crucial to investigate utilizing the stroke order while minimizing intra-class variations of input features, especially when we embed shape cues for sketch recognition.

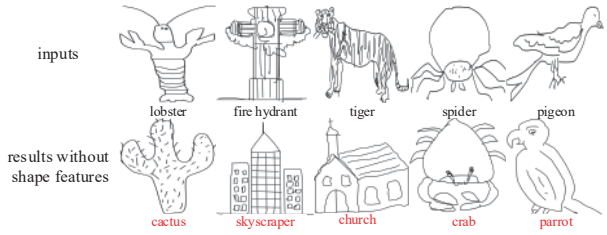

Figure 1: Some successful classification cases with added explicit shape features that failed without shape features.

We demonstrate the effects of added explicit shape features in Fig. 1. The first row shows five query sketches that were successfully recognized with added shape features but wrongly classified without shape features. The most similar shapes from their wrongly assigned classes are displayed in the second row. We notice similar texture features of corresponding sketches. For example, the horizontal and vertical lines are similar for both fire hydrant and skyscraper, and also the stripes on tiger and church are similar. On the other hand, when explicit shape features are added, we are able to distinguish different shapes of pigeon and parrot, possibly including subtle local differences such as the shape of their beaks.

This paper presents a sequential dual recurrent neural networks (SD-RNN) architecture for sketch recognition that combines both shape and texture of sketches as shown in Fig. 2. Five images are constructed by *accumulative strokes* according to the stroke order. Subsequently, coded shape context [37] and texture from Sketch-A-Net (SAN) [40] are fed into two cascaded gated recurrent units (GRUs). The outputs of these two GRUs stages produce the final classification. Our contributions are summarized as follows:

- To the best of our knowledge, explicit shape features are introduced into sketch recognition for the first time in a deep learning framework.
- Dual recurrent neural networks are cascaded into two stages to combine both shape and texture features, and to learn the balance of their contributions.
- By quantizing the strokes into five stages, we explore the sequential nature of strokes while reducing intra-class variations of input features.
- We develop an on-line sketch retrieval and imitation application to teach others to draw based on the proposed method.

Experiments on the TU-Berlin benchmark, the largest hand-free sketch dataset, show that our method achieves over 90% recognition rate, and outperforms both human and state-of-the-art algorithms by over 19 and 7 percentage points, respectively.

2 RELATED WORK

Sketch recognition has attracted much attention in recent years, science a large crowd-sourced dataset was published [9]. The dataset contains 20,000 sketches from daily objects distributed over 250 object categories. The fact that human can only correctly identify the sketches with 73% accuracy shows that sketch recognition is a challenging task even for humans.

Most existing works on sketch interpretation typically analyze an input sketch as a traditional texture image, without attempting to understand its shape nature. Hand-crafted features borrowed from texture images are first employed as representation. Eitz *et at.* use SIFT along with a special treatment to smooth gradients and sparse intensities in sketches [9]. Similarly, [29] applies Fisher vectors and spatial pyramid pooling to SIFT. Li *et al.* employ multiple-kernel learning (MKL) to find appropriate weights of various textural features for sketches [21]. Zhang *et al.* transfer the knowledge of a network learned from natural images to a sketch network [41]. However, these features highly depend on gradients information, which rarely exists in sketches.

Recently, learning based features are explored for sketch recognition due to the great success of deep learning in visual recognition [17]. Wang *et al.* use a variant of Siamese network for sketch-to-3D-shape retrieval [33]. In [28], sketch features are obtained by AlexNet. Zhang *et al.* design a hybrid network combined multiform features [42]. CousinNet [41]leverages natural images to guide the target network to extract powerful features for recognition. However, these sketch features are produced by deep networks designed for texture images, and little attention has been paid on the special nature of sketches, which mostly encode curves and lines. One exception is the Sketch-A-Net (SAN) work [39, 40]. Yu *et al.* take the sparse nature of sketches into account, and design a deep learning network that enlarges the pooling sizes and patches of filters. Sketch-A-Net surpasses the best result of humans for the first time. SketchPointNet shares similar idea and leverages sparse sampling points on sketch [36]. However, the shape nature of sketches is still ignored.

Shape representation has been studied in computer vision for a long time. Shape context [1] and its variants [26, 34] are among the most popular shape descriptors. Researchers also develop descriptors to accommodate a wide range of geometric transformations [15]. Motivated by the middle level "bag-of-features", Wang *et al.* develop a bag of contour fragments (BCF) approach that achieves the state-of-the-art for simple shape contour classification [37]. Different from shapes of natural objects [19], sketches have high flexibility and sequential information on strokes. One can hardly obtain satisfactory results by directly applying shape descriptors to sketch recognition. In this paper, we leverage a mid-level shape feature for sketches to reduce the variation of intra-classes. Meanwhile we also combine both shape and texture features to achieve the state-of-the-art recognition performance.

More specifically, a sketch is an ordered list of strokes. In Sketch-A-Net [39], the sequential nature of sketches is explored to produce abstract sketches for data augmentation. However, Sketch-A-Net does not build connections between sequential strokes. DVSF [12] improves Sketch-A-Net by combining sequential feature and integrated features. However, shape features are still ignored.

For time series like speech and natural language, many works resort to memorable network architectures. A recurrent neural network (RNN) [30, 31] is designed for processing input sequence, which can bridge the hidden units and deliver the outputs from former sequence to the latter ones. However, it has a significant limitation called 'vanishing gradient'. In order to overcome the limitation of RNN, a long short term memory (LSTM) [11] network, and recently gated recurrent unit (GRU) [3] have been proposed.

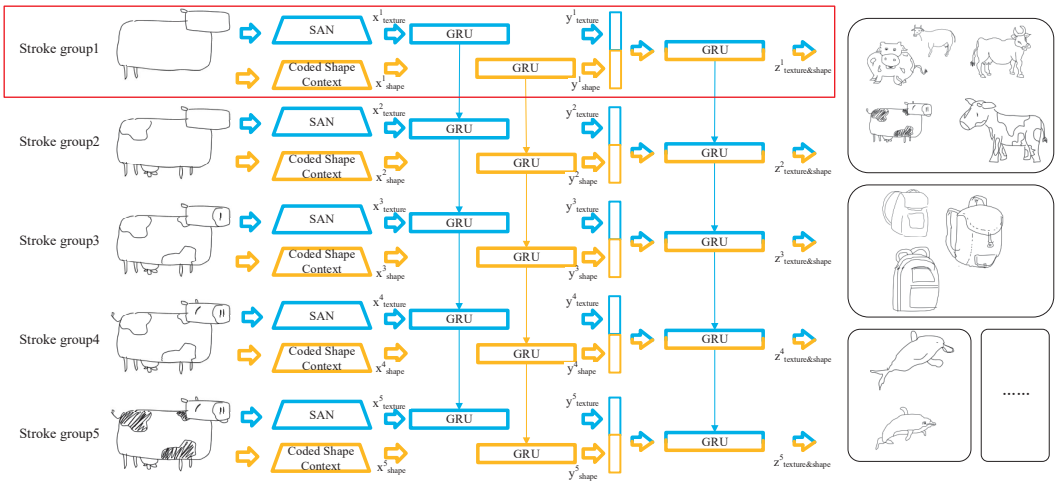

Figure 2: Overview of SD-RNN. Textural features of Sketch-A-Net ($x^1_{texture}, \ldots, x^5_{texture}$) and coded shape features ($x^1_{shape}, \ldots, x^5_{shape}$) (labeled in blue and yellow, respectively) extracted from each accumulative stroke image are taken as the inputs of GRUs . The corresponding prediction sequence $y^1_{shape}, \ldots, y^5_{shape}$ and $y^1_{texture}, \ldots, y^5_{texture}$ are combined and fed into the second stage GRUs. All the outputs of $z^1_{shape\&texture}, \ldots, z^5_{shape\&texture}$ generate the final classification result.

GRU outperforms LSTM in many cases by learning smaller number of parameters [4]. Sarvadevabhatla *et al.* [28] take the order of strokes as a sequence and feed their features stroke by stroke to GRU. They achieve the best published recognition performance. However, different form handwritten characters with relatively fixed structural ordering, sketches exhibit a much higher degree of intra-class variation in stroke order. These variations definitely bring unstable features. In contrast, our work aims to learn a stable sequential features capturing both shape and texture information for sketch recognition.

3 EXPLORING STROKE ORDERS

The stroke order of the same sketch varies more severely than hand written digits and characters. Every person may have his/her own order of strokes when drawing the same object [9]. There are two pairs of sketches divided by the horizontal dash line, i.e., cows and elephants, as shown in Fig. 3. Each pair belong to the same class but drawn by different individuals. The two vertical dash lines separate the figure into three columns, the left two of which demonstrate the first ten strokes and the the complete sketches, respectively. The first stroke of the 'cow' in one sketch of Fig. 3 lies in its back while the other in its belly. The stroke orders evidently vary in the same class. These variations definitely bring unstable features (will be discussed in experiments). Inevitably, these unstable inputs affect the convergence and further classification accuracy.

Therefore, we use *accumulative strokes* instead of single one in order to reduce input variations for the same class. This treatment is motivated by humans' drawing habit. Most people prefer to draw from the outline to details, which can be validated in the last column of Fig. 3. The right column shows the accumulative stroke images derived from stroke orders: the first image shows the first 20% strokes, and the second one gives the first 40% in the stroke sequence, and so on. The image variations in the same accumulative group (corresponding to one step in our SD-RNN) clearly decrease.

We take statistics on the number of the strokes, and find that 54.7% sketches have the number of strokes between 6 and 20, while 16% equal or less than 5. Thus, we take five groups with about four strokes in each group. If the number of groups was too small, there would exist strokes with both outer contour and details, while larger number of groups may result in only one stroke for each group, degrading to stroke-by-stroke strategy. Supposing that there are N strokes for a sketch S, $(s_1, s_2, \ldots, s_{N-1}, s_N)$, we can construct an sequence with five images. The first one contains the strokes from s_1 to $s_{N/5}$, the second one from s_1 to $s_{2N/5}$, and the last one shows the complete sketch.

Similar to [18], we also expand each accumulative image to ten by cropping and reflection. These ten accumulative stroke images are taken as the input at one step of GRU. Figure 4 shows an input sequence of one accumulative stroke group. From the right to left, the inputs are the crops of original images when time step t is odd, and the inputs are the crops of orignial reflections for even ts. The crop order for each original image is the top left, bottom left, top right, bottom right, and finally center. As we have five accumulative stroke images for each sketch and ten crops for each image, we finally have 50 input images in total. Their corresponding features are fed into our deep network described in the next section.

4 INTEGRATING SHAPE AND TEXTURE WITH RECURRENT NEURAL NETWORKS

In this section, we first introduce the shape and texture features for sketches. Then we present a specially designed network to combine and balance these features.

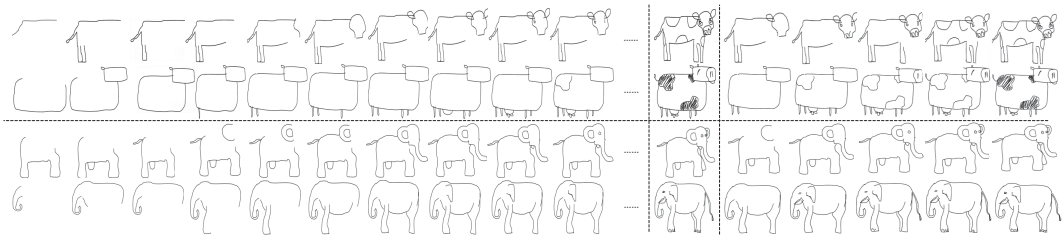

Figure 3: Comparisons of single stroke (left) and accumulative stroke images (right).

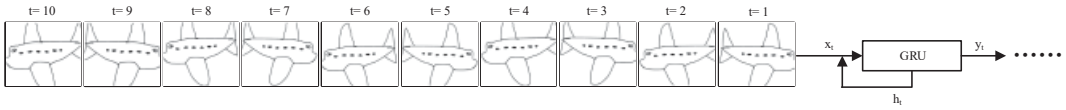

Figure 4: An input image sequence for one accumulative sketch image.

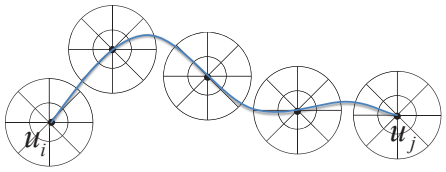

Figure 5: Shape context features are collected at five sample points on a stroke between points u_i and u_j.

4.1 Shape features of strokes

Sketches are composed of strokes, which contain both local and global shape information. Thus, we adopt strokes as basic shape element for learning a shape codebook and building our shape representation.

For each stroke, we describe it using shape context [1]. We sample 5 reference points on the stroke equidistantly, and then compute 5 shape context histograms based on the reference points individually. Shape context descriptor for each stroke is a concatenation of the 5 shape context histograms, e.g., see Fig. 5.

Each stroke of a sketch can be drawn in different styles, rendering its high flexibility. In order to handle these intra-class variations, we leverage a coding method [37] to generate close features from similar strokes. We first apply the k-means algorithm [8] on shape context descriptors of randomly selected strokes. The cluster centers are regarded as the codebook. Thereafter, we can use M prototypes to describe the whole stroke space as shown in Fig. 6 where the colorful dots stand for the cluster centers. In our experiments, M is set to 500 as in [37]. We use a fast and effective scheme, local-constraint linear coding (LLC) [35], to generate the final coded representation of strokes. As shown in Fig. 6, similar strokes may have the same k nearest neighbors so that we take the k ($k = 5$ in our paper) nearest neighbors to encode each shape feature. Finally, we run max-pooling [14] on all stroke features of each sketch to obtain more discriminative features with 500 dimensions.

4.2 Texture features for sketches

We leverage Sketch-A-Net [40] to extract texture features of the sketches. Sketch-A-Net is a variant of a CNN tailored to sketch recognition, consisting of eight layers. The first five layers are convolutional layers, and the last three are fully connected. Each convolutional layer is constructed upon rectifier (ReLU) units, while the first, second and fifth layers are followed by max pooling. For each input image, Sketch-A-Net produces a 512 dimensional feature vector from the last fully-connected layer.

In [40], five versions of the same deep network are run, each for a different resolution of the input image, and the final classification is obtained by a joint Bayesian classifier [2] applied to the five outputs. In this paper, we employ only the network from [40] trained for sketches of size 256×256 pixels, and take the 512-dimensional vector of the last fully-connected layer as the texture features of sketches without any retraining.

4.3 Integrating shape and texture features with recurrent neural networks

Shape features describe the geometric structure of sketches along strokes, which can be viewed as 1D curves. In contrast, texture features describe 2D patches on surface of sketches, which are treated as gray scale images. In order to combine and balance shape and texture features while exploring the sequential nature of strokes, we devise a sequential dual recurrent neural network (SD-RNN) architecture.

4.3.1 Sequential dual recurrent neural network (SD-RNN). We take GRUs as basic blocks to combine both shape and texture features and exploit the sequential nature of sketches, since GRUs have a smaller number of parameters and perform better than LSTM in sketch recognition [28].

The overview of our SD-RNN is shown in Fig. 2. As shown in the red frame, there are two GRUs in the first stage, which receive the two types of features as inputs, respectively. Then a third GRU in the second stage combines the outputs of the first stage, i.e., memorized shape and texture features. The first unit inside the third GRU is a

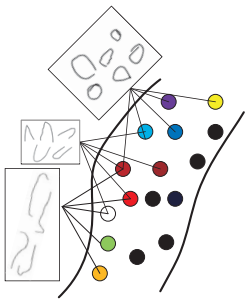

Figure 6: Encoded features of similar shapes. The colorful circle points stand for the cluster centers, and several of these clusters represent a stroke.

linear function that reduces the dimensionality of the concatenation of the two vectors of shape and texture features. The output of the third GRU is densely connected to a final softmax layer, acting as the classifier. Each of the three GRUs of this two-stage network contains 128 hidden units.

The obtained SD-RNN is run five times on different accumulative stroke groups, which is illustrated in Fig. 2 with five copies of the same network linked by vertical arrows to illustrate the temporal dependencies of the input sequence. This architecture is able to learn the weights of the two types of features, and thus balance their influence to produce the desired performance. As a result, our SD-RNN embraces learnable textural features, memorized shape features, and learnable fusion of these two. We present more details now.

Taking the shape feature as an example, the GRU shape network in the first stage learns a mapping from the input x^t_{shape} to the output y^t_{shape} for $t = 1, \ldots, 5$, which is given by

$$r_t = \sigma\left(W_{xr}x^t_{shape} + W_{hr}h_{t-1} + b_r\right), \tag{1}$$

$$z_t = \sigma\left(W_{xz}x^t_{shape} + W_{hz}h_{t-1} + b_z\right), \tag{2}$$

$$\tilde{h_t} = \tanh(W_{xh}x^t_{shape} + U\,(r_t \odot h_{t-1}) + b_h), \tag{3}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h_t}, \tag{4}$$

$$y^t_{shape} = W_{hy}h_t, \tag{5}$$

where $\sigma(\cdot)$ is an element-wise logistic Sigmoid function. h_t is the hidden state and it is regulated by the gating unit r_t, z_t and $\tilde{h_t}$. The operator $\odot$ denotes the element-wise vector product. The Ws, bs and U are trainable parameters. Similarly, we can get the t-th output for texture as $y^t_{texture}$. More details about GRUs can be found in [5].

4.3.2 Features fusion and balance. Both shape and texture features are fed into the respective GRUs of the first-stage, outputting two 128 dimensional vectors, $y^t_{texture}$ and y^t_{shape}. For the third GRU in the second stage, the t-th input is $[(y^t_{texture})^\top (y^t_{shape})^\top]^\top$, and the output is $z^t_{texture\&shape}$.

We add a linear function $W_c[(y^t_{texture})^\top (y^t_{shape})^\top]^\top + b_c$ at the beginning of the second-stage GRU (just before Eq. 1) so that our SD-RNN has the ability to automatically learn the weights of two features for recognition. The weights of the linear function W_c are 256×128 as both $y_{texture}$ and y_{shape} have 128 dimensions. These parameters in W_c and b_c are fine-tuned in the learning process, resulting in the increased performance of our network.

For each of the five SD-RNNs in Fig. 2, we obtain a C-dimensional vector of class probabilities $z^t_{shape\&texture}$ with C being the number of classes ($C = 250$ in our experiments) and $t = 1, \ldots, 5$. We sum the prediction vectors of all 5 steps, and choose the class ID with the max value of the summation as the final output.

5 EXPERIMENTAL RESULTS AND ANALYSIS

In this section, we first describe the dataset and experiment settings. Subsequently, we validate the effectiveness of the shape feature and stroke orders. Finally, our method is compared with the state-of-the-art and humans with the same protocol in order to demonstrate the overall performance of SD-RNN. The results show that our method does not only produce the accuracy over 90% for the first time, but also beats the state-of-the-art over 14 percentage points.

5.1 Experiment settings

We evaluate SD-RNN on the TU-Berlin sketch dataset [9], the largest and most commonly used sketch dataset currently available. It consists of 20,000 sketches of 250 categories, 80 sketches per category. The dataset was collected on Amazon Mechanical Turk (AMT) from 1350 participants. Thus the dataset guarantees the diversity of object categories and sketch styles within every category. As is commonly the case, we use 67% of the data for training, and 33% for testing.

As the performance measure we use the standard classification accuracy, which is defined as the ratio of the number of correctly classified sketches to the total number of sketches in the test dataset.

Data augmentation is employed to reduce the risk of overfitting. In order to increase the number of sketches per category, we apply several transformations on each sketch, including horizontal reflection and rotation ([-5,-3,0,+3,+5] degrees), followed by systematic combinations of horizontal and vertical shifts (±15 pixels). The data augmentation procedure results in $18\,(10 + 8 = 18) \times 80 = 1,440$ sketches per category, a total number of $1440 \times 250 = 360,000$ sketches evenly distributed over 250 categories.

For shape features, we apply shape context on 5 points with equal interval of each stroke. The number of bins of shape context is set to be 60. Thus, the dimension of the shape context descriptor for each stroke is 300 with the size of the codebook 500. We implement SD-RNN on Torch [6], and set the initial learning rate to 0.002 with weights decay 0.99. The batch size is set to 100.

5.2 Contributions of shape features

Table 1 lists the classification accuracy of SD-RNN obtained by only shape features, only texture features, and both in order to validate the effects of different features in our method. We can see that the accuracies with only shape and texture features can only reach 30.32% and 77.03% respectively, in contrast to 92.65% when we combine both in our architecture. Shape features may not be able to gain good performance by itself, but they play an important role in sketch recognition, improving the accuracy of SD-RNN with only

Table 1: Evaluation on the Contributions of Shape Features

Our SD-RNN Method	Recognition Accuracy (%)
Shape Feature Only	30.32
Texture Feature Only	77.03
Our SD-RNN Method	92.65

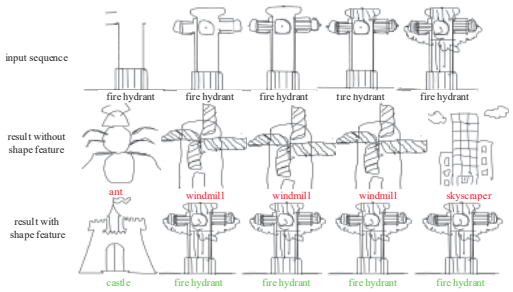

Figure 7: Classification results without and with shape features at different time steps. The first row shows input image sequence, the second and third row exhibit the results without and with shape features.

texture features over 15 percentage points. Simply concatenating shape and texture features with fixed weights cannot output desired performance as ours because SD-RNN is capable of learning their contributions adaptive to object categories.

In order to give an intuitive demonstration of the effects of shape features, we present some false recognition examples without shape features but successfully recognized with shape features in Fig. 1. The first row lists five query sketches including lobster, fire hydrant, tiger, spider, and pigeon. The second row shows that SD-RNN without shape features produces wrong classification results for all these queries. It can be explained as texture features can represent detail features of the sketch, such as the stripes on tiger and church, and the horizontal and vertical lines for both fire hydrant and skyscraper. However, the shape of these wrong matched pairs are totally different. With the help of shape features, all these query sketches can be recognized sucessfully. Our method can even distinguish pigeon and parrot with subtle local differences on the beak.

Further, we illustrate the classification results for each time step shown in Fig. 7. The first row shows the query sketches represented by five accumulative stroke images, the second and third rows provides the results without and with shape features at each time step, respectively. The results of each output in the second column are all incorrect, while the third one produces correct results from the second output of the sequence. These results demonstrate that shape features can also discriminate geometrically similar strokes.

5.3 Contributions of SD-RNN architecture

In order to validate the effect of SD-RNN architecture, we compare the result of the proposed method with Sketch-A-Net architecture coupled both shape and texture features. We are able to boost the 74.9% classification accuracy of Sketch-A-Net to 84% by concatenating shape context features with Sketch-A-Net deep features with

Table 2: Evaluation on Different Fusion Strategies

Method	Recognition Accuracy (%)
direct concatenation	88.33
maxpooling	90.46
our method with linear function	92.65

Table 3: Sketch Recognition on TU-Berlin Dataset

Method	Recognition Accuracy(%)
HOG-SVM [9]	56.0
Ensemble [22]	61.5
MKL-SVM [21]	65.8
FV-SP [29]	68.9
Humans	73.1
AlexNet-SVM [17]	67.1
AlexNet-Sketch [17]	68.6
LeNet [20]	55.2
Sketch-A-Net 1.0 [40]	74.9
Sketch-A-Net 2.0 [39]	78.0
Hybrid-Conv [42]	85.1
SketchPointNet [36]	74.2
DVSF [12]	79.6
CousinNet [41]	80.1
Our SD-RNN method	92.65

a linear SVM. However, there is still a notable gap between the result (92.65%) of our SD-RNN and this naive strategy. Thus, the proposed SD-RNN architecture can not only integrate shape and texture features, but also memorize and pass the features in each step to obtain the final performance.

5.4 Comparisons of different stroke order strategies

In order to validate the robustness of our accumulative stroke strategy, we compare the features produced by the accumulative stroke images with that of the linear order of all individual strokes in [28]. Since, typically, the first stroke exhibits significant variations person by person, we peer into the input shape and texture features of the first stroke, and compare the feature variances of the first stroke and our first accumulative stroke group.

For fair comparison, each feature vector is normalized by Eq. (6), where x_i and $\tilde{x_i}$ $(i = 1, 2, \ldots, n)$ are the elements of x_{shape} or $x_{texture}$ before and after normalization, respectively, and n is the dimension of the vector.

$$\tilde{x_i} = \frac{x_i}{\sqrt{(x_1)^2 + (x_2)^2 + \ldots + (x_n)^2}} \tag{6}$$

Figs. 8(a) and (b) show the variances of shape and texture features, respectively. The x-axis shows the label of the 250 categories, and y-axis is the value of the variances. The variances of ours and the first stroke strategy in each category are labeled in blue plus sign and red star, respectively. The shape feature variances of our strategy are smaller for 242 out of 250 categories, while our variances on

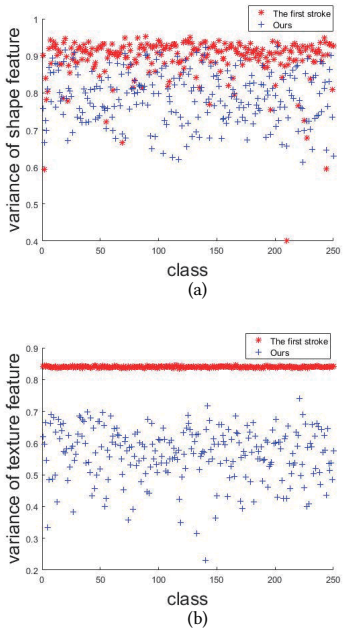

Figure 8: Comparison of (a) variances of shape and (b) texture features for 250 sketch classes.

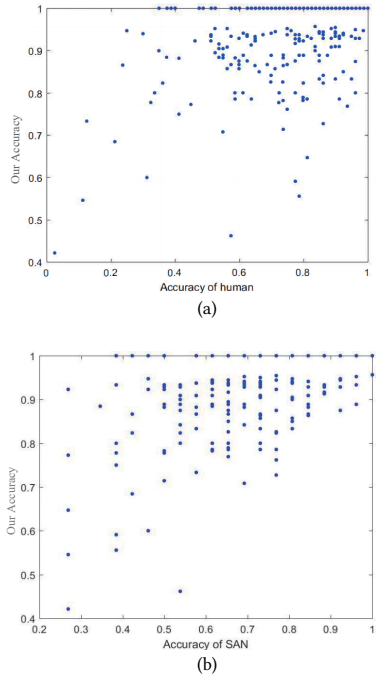

Figure 9: Accuracy comparisons between our SD-RNN and human in (a), and Sketch-A-Net [39] in (b).

Figure 10: Illustration of recognition successes and failures.

textures are smaller for *all* categories. This is clearly visible, since almost all blue pluses are below the red stars.

5.5 Comparisons on different feature fusion strategies

Given two features, various fusion strategies can be adopted for the final classification. Table 2 compares our linear function fusion method with the alternative direct concatenation and maxpooling fusion. For the direct concatenation fusion, we concatenate the shape and texture vectors obtained as the outputs of the first stage to a single vector as the inputs to the second stage. The maxpooling combination reaches the accuracy as 90.46%. The maxpooling makes the combined feature more discriminative than the concatenation one. Our linear function fusion can still achieve the best performance among the three. It is also possible to consider other fusion strategies, e.g., fusion before the input of the first-stage GRUs or after the second-stage GRU layer.

5.6 Comparisons with the state-of-the-art

We compare SD-RNN with the state-of-the-art methods for sketch recognition. These methods can be divided into two groups. One combines hand-crafted features and classifiers, e.g., HOG-SVM [9], structured ensemble matching [22], multi-kernel SVM [21] and Fisher Vector Spatial Pooling (FV-SP) [29]. The other is deep learning based methods including AlexNet [17], LeNet [20], AlexNet-FC-GRU [28], two versions (Sketch-A-Net 1.0 [40] and Sketch-A-Net 2.0 [39]) of Sketch-A-Net framework, DVSF [12], Hybrid convolutional neural network [42], SketchPointNet [36], and CousinNet [41].

Table 3 shows the classification accuracy of the compared methods on the TU-Berlin dataset. The listed results of other methods were generated by their own implementations or reported in their papers. In general, methods based on deep networks obtain better performance than those based on hand-crafted features. The accuracy of the methods based on hand-crafted features is lower than humans [9]. This is because the existing hand-crafted features are well-designed for traditional images but not suitable for abstract and sparse sketches. Among the existing deep learning based methods, Sketch-A-Net 1.0 [40] is the first method that beats humans with the accuracy 74.9%, and the improved version (Sketch-A-Net 2.0) obtains the accuracy 78.0%. DVSF [12] improves Sketch-A-Net by combining integrated sketch features, and obtains the accuracy of 79.6%. SketchPointNet [36] obtains similar accuracy of 74.2% by

taking the sparse of sketch into consideration. Our experiments cover all the categories and keep the same proportion for training (2/3) and testing (1/3) as these works do. Our method outperforms these methods by more than 13% in accuracy.

Among the methods introducing additional information, Cousin-Net [41] leverages natural images to guide the target network to extract powerful features for recognition. Seventy-two instances (90% of the total) in each category are used for training, while we only use 66.7% (2/3) for training. However, we outperform Cousin-Net by 12.55 percentage points. Another method that also combined multiform features is Hybrid convolutional neural network [42]. Sketches are represented and learned by point-set in one branch, and AlexNet is used for the other branch. The hybrid network reaches the highest accuracy of 85.1% among the existing methods. However, this is still 7.55 percentage points lower than our SD-RNN method.

Further, we separately demonstrate the results of 250 categories compared with human and Sketch-A-Net [39] implemented on the same dataset and the same protocol with ours, and having the closest accuracy to ours. The horizontal coordinate of each point of the 250 blue points in Fig. 9(a) shows the human accuracy, while the value of the vertical coordinate shows the accuracy of our method. Similarly, our results compared with Sketch-A-Net are shown in Fig. 9(b). For both figures, we can see most points lie in the upper triangle, which means higher accuracy of our approach than the other two. It is notable that our method obtains 100% rate on 110 categories, while humans have only one and Sketch-A-Net only two categories with 100% accuracy. The lowest accuracy for our method is 42.11% on the category of 'seagull', which is also difficult for human and Sketch-A-Net to recognize with the accuracies as low as 2.50% and 26.92%, respectively.

Figure 10 shows some tough examples. The ground truth is labeled in black, while our results are labeled in green and other methods in red. Our method succeeds in the first row but other methods fail. In the second row, mistakes made by our method seem reasonable. One may expect humans making similar mistakes. The clear challenge level of sketch ambiguity demonstrates why reliable sketch based communication is hard even for humans.

5.7 Online application

Base on the proposed method, we designed a novel app to teach children and adults to draw [1]. After a user first creates a rough sketch, the app suggests most similar complete sketches. The user can then select one of them and imitate it by retracing. The system overview is shown in Fig. 11. The app runs in real-time and submits the rough sketch to the server. Both geometry and texture features are extracted and fed into the proposed SD-RNN on the server. The recognition result and its corresponding instances are sent back to the app. Users can choose any sketch they like to imitate. Our results show that the proposed network architecture, and on-line application generalize well to real user input and enable high quality retrieval results without additional post-processing. Some retrieval instances of our proposed online app are shown in Fig. 12.

[1]The application is available as Sketch.Draw.

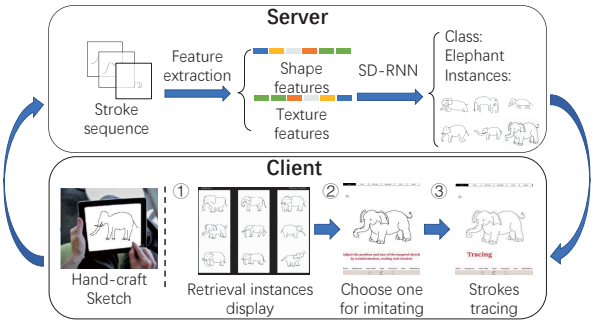

Figure 11: System overview of the proposed online application.

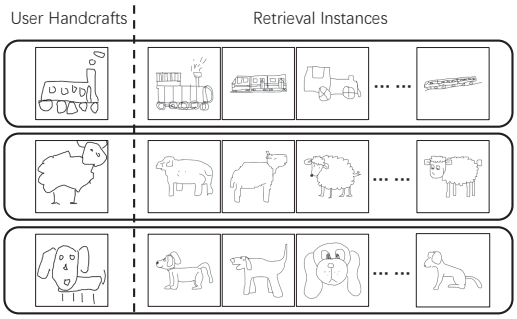

Figure 12: Retrieval instances of proposed online application.

6 CONCLUSION

In this paper, we propose a sequential dual recurrent neural network that combines both shape and texture features for sketch recognition. Experimental results demonstrate that we achieve the best performance compared with the state-of-the-art and humans on sketch recognition. We use shape features to characterize geometric information of sketches, which nicely complement texture features, and combine both feature types in a deep learning framework. Moreover, we explore the sequential nature of accumulative stroke images rather than direct order of individual strokes. Thus, our framework can enable interesting applications such as our sketch retrieval and stick figure imitation framework. The learned features of sketches can also be used in some other sketch-related applications, such as sketch-based emoji retrieval, image and 3D shape retrieval.

ACKNOWLEDGMENTS

This work is partially supported by the Natural Science Foundation of China under grant Nos. 61733002, 61632006 and 61876030, Fundamental Research Funds for the Central University under grant No. DUT19RC(3)004, Natural Science Foundation of Liaoning Province under grant No.20170540173, and by the NSF under Grant No. IIS-1814745.

REFERENCES

[1] Serge Belongie, Jitendra Malik, and Jan Puzicha. 2002. Shape Matching and Object Recognition Using Shape Contexts. *IEEE TPAMI* 24, 4 (2002), 509–522.

[2] Dong Chen, Xudong Cao, Liwei Wang, Fang Wen, and Jian Sun. 2012. Bayesian face revisited: A joint formulation. In *European Conference on Computer Vision*. Springer, 566–579.

[3] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[4] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

[5] Junyoung Chung, Caglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. 2015. Gated Feedback Recurrent Neural Networks.. In *ICML*. 2067–2075.

[6] Ronan Collobert, Samy Bengio, and Johnny Mariéthoz. 2002. *Torch: a modular machine learning software library.* Technical Report. Idiap.

[7] Navneet Dalal and Bill Triggs. 2005. Histograms of Oriented Gradients for Human Detection. In *2005 IEEE Computer Society Conference on CVPR (CVPR 2005)*. 886–893. https://doi.org/10.1109/CVPR.2005.177

[8] Richard O Duda, Peter E Hart, and David G Stork. 2000. Pattern Classification and Scene Analysis Part 1: Pattern Classification. *Wiley, Chichester* (2000).

[9] Mathias Eitz, James Hays, and Marc Alexa. 2012. How do humans sketch objects? *ACM Trans. Graph.* 31, 4 (2012), 44:1–44:10. https://doi.org/10.1145/2185520.2185540

[10] Mathias Eitz, Kristian Hildebrand, Tamy Boubekeur, and Marc Alexa. 2011. Sketch-Based Image Retrieval: Benchmark and Bag-of-Features Descriptors. *IEEE Trans. Vis. Comput. Graph.* 17, 11 (2011), 1624–1636. https://doi.org/10.1109/TVCG.2010.266

[11] Alex Graves. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (1997), 1735.

[12] Jun-Yan He, Xiao Wu, Yu-Gang Jiang, Bo Zhao, and Qiang Peng. 2017. Sketch recognition with deep visual-sequential fusion model. In *Proceedings of the 25th ACM international conference on Multimedia*. 448–456.

[13] Rui Hu and John P. Collomosse. 2013. A performance evaluation of gradient field HOG descriptor for sketch based image retrieval. *CVIU* 117, 7 (2013), 790–806. https://doi.org/10.1016/j.cviu.2013.02.005

[14] Yongzhen Huang, Zifeng Wu, Liang Wang, and Tieniu Tan. 2014. Feature coding in image classification: A comprehensive study. *IEEE TPAMI* 36, 3 (2014), 493–506.

[15] Qi Jia, Xin Fan, Yu Liu, Haojie Li, Zhongxuan Luo, and He Guo. 2016. Hierarchical projective invariant contexts for shape recognition. *PR* 52 (2016), 358 – 374. https://doi.org/10.1016/j.patcog.2015.11.003

[16] Brendan F. Klare, Zhifeng Li, and Anil K. Jain. 2011. Matching Forensic Sketches to Mug Shot Photos. *IEEE TPAMI* 33, 3 (2011), 639.

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*. 1106–1114.

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[19] Longin Jan Latecki, Rolf Lakamper, and Ulrich Eckhardt. 2000. Shape Descriptors for Non-rigid Shapes with a Single Closed Contour. In *Proc. of CVPR*. 424–429.

[20] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, R. E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. 1989. Handwritten Digit Recognition with a Back-Propagation Network. In *Advances in NIPS 2, [NIPS Conference, Denver, Colorado, USA, November 27-30, 1989]*. 396–404.

[21] Yi Li, Timothy M. Hospedales, Yi-Zhe Song, and Shaogang Gong. 2015. Free-hand sketch recognition by multi-kernel feature learning. *CVIU* 137 (2015), 1–11. https://doi.org/10.1016/j.cviu.2015.02.003

[22] Yi Li, Yi-Zhe Song, and Shaogang Gong. 2013. Sketch Recognition by Ensemble Matching of Structured Features.. In *BMVC*, Vol. 1. 2.

[23] Hangyu Lin, Yanwei Fu, Peng Lu, Shaogang Gong, Xiangyang Xue, and Yu-Gang Jiang. 2019. Tc-net for isbir: Triplet classification network for instance-level sketch based image retrieval. In *Proceedings of the 27th ACM International Conference on Multimedia*. 1676–1684.

[24] David G. Lowe. 2004. Distinctive Image Features from Scale-Invariant Keypoints. *IJCV* 60, 2 (2004), 91–110. https://doi.org/10.1023/B:VISI.0000029664.99615.94

[25] Samuel Ritter, David G. T. Barrett, Adam Santoro, and Matt M. Botvinick. 2017. Cognitive Psychology for Deep Neural Networks: A Shape Bias Case Study. (2017).

[26] Edgar Roman-Rangel, Carlos Pallan Gayol, Jean-Marc Odobez, and Daniel Gatica-Perez. 2011. Searching the past: an improved shape descriptor to retrieve maya hieroglyphs.. In *ACM Multimedia*. 163–172.

[27] P. Sangkloy, J. Lu, C. Fang, F. Yu, and J. Hays. 2017. Scribbler: Controlling Deep Image Synthesis with Sketch and Color. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 6836–6845.

[28] Ravi Kiran Sarvadevabhatla, Jogendra Kundu, and R. Venkatesh Babu. 2016. Enabling My Robot To Play Pictionary: Recurrent Neural Networks For Sketch Recognition. In *Proceedings of the 2016 ACM Conference on Multimedia Conference, MM 2016, Amsterdam, The Netherlands, October 15-19*. 247–251. https://doi.org/10.1145/2964284.2967220

[29] Rosália G Schneider and Tinne Tuytelaars. 2014. Sketch classification and classification-driven analysis using Fisher vectors. *ACM TOG* 33, 6 (2014), 174.

[30] Ilya Sutskever, James Martens, and Geoffrey E. Hinton. 2011. Generating Text with Recurrent Neural Networks. In *Proceedings of the 28th ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*. 1017–1024.

[31] Oriol Vinyals, Suman V Ravuri, and Daniel Povey. 2012. Revisiting recurrent neural networks for robust ASR. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. 4085–4088.

[32] Fang Wang, Le Kang, and Yi Li. 2015. Sketch-based 3D shape retrieval using Convolutional Neural Networks. In *IEEE CVPR*. 1875–1883. https://doi.org/10.1109/CVPR.2015.7298797

[33] Fang Wang, Le Kang, and Yi Li. 2015. Sketch-based 3d shape retrieval using convolutional neural networks, In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. *CVPR*, 1875–1883.

[34] Junwei Wang, Xiang Bai, Xingge You, Wenyu Liu, and Longin Jan Latecki. 2012. Shape matching and classification using height functions. *Pattern Recognition Letters* 33, 2 (2012), 134–143.

[35] Jinjun Wang, Jianchao Yang, Kai Yu, Fengjun Lv, Thomas Huang, and Yihong Gong. 2010. Locality-constrained linear coding for image classification. In *CVPR, 2010 IEEE Conference on*. IEEE, 3360–3367.

[36] Xiangxiang Wang, Xuejin Chen, and Zhengjun Zha. 2018. Sketchpointnet: A Compact Network for Robust Sketch Recognition. In *2018 25th IEEE International Conference on Image Processing (ICIP)*. IEEE, 2994–2998.

[37] Xinggang Wang, Bin Feng, Xiang Bai, Wenyu Liu, and Longin Jan Latecki. 2014. Bag of contour fragments for robust shape classification. *Pattern Recognition* 47, 6 (2014), 2116–2125.

[38] X. Wang, C. He, M. V. Peelen, S. Zhong, G. Gong, A Caramazza, and Y. Bi. 2017. Domain Selectivity in the Parahippocampal Gyrus Is Predicted by the Same Structural Connectivity Patterns in Blind and Sighted Individuals. *Journal of Neuroscience* 37, 18 (2017), 4705.

[39] Qian Yu, Yongxin Yang, Feng Liu, Yi Zhe Song, Tao Xiang, and Timothy M. Hospedales. 2016. Sketch-a-Net: A Deep Neural Network that Beats Humans. *IJCV* (2016), 1–15.

[40] Qian Yu, Yongxin Yang, Yi-Zhe Song, Tao Xiang, and Timothy M. Hospedales. 2015. Sketch-a-Net that Beats Humans. In *Proceedings of the BMVC 2015, Swansea, UK, September 7-10, 2015*. 7.1–7.12. https://doi.org/10.5244/C.29.7

[41] Kaihao Zhang, Wenhan Luo, Lin Ma, and Hongdong Li. 2019. Cousin network guided sketch recognition via latent attribute warehouse. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 9203–9210.

[42] Xingyuan Zhang, Yaping Huang, Qi Zou, Yanting Pei, Runsheng Zhang, and Song Wang. 2020. A Hybrid convolutional neural network for sketch recognition. *Pattern Recognition Letters* 130 (2020), 73 – 82. https://doi.org/10.1016/j.patrec.2019.01.006 Image/Video Understanding and Analysis (IUVA).