Edinburgh Research Explorer

Sketch-a-Net: A Deep Neural Network that Beats Humans

Sketch-a-Net: a Deep Neural Network that Beats Humans

Qian Yu, Yongxin Yang, Feng Liu, Yi-Zhe Song, Tao Xiang and Timothy M. Hospedales

Abstract We propose a deep learning approach to free-hand sketch recognition that achieves state-of-the-art performance, significantly surpassing that of humans. Our superior performance is a result of modelling and exploiting the unique characteristics of free-hand sketches, i.e. consisting of an ordered set of strokes but lacking visual cues such as colour and texture, being highly iconic and abstract, and exhibiting extremely large appearance variations due to different levels of abstraction and deformation. Specifically, our deep neural network, termed Sketch-a-Net has the following novel components: (i) We propose a network architecture designed for sketch rather than natural photo statistics. (ii) Two novel data augmentation strategies are developed which exploit the unique sketch-domain properties to modify and synthesise sketch training data at multiple abstraction levels. Based on this idea we are able to both significantly increase the volume and diversity of sketches for training, and address the challenge of varying levels of sketching detail commonplace in free-hand sketches. (iii) We explore different network ensemble fusion strategies, including a re-purposed joint Bayesian scheme, to further improve recognition performance. We show that state-of-the-art deep networks specifically engineered for photos of natural objects fail to perform well on sketch recognition, regardless whether they are trained using photos or sketches. Furthermore, through visualising the learned filters, we offer useful insights in to where the superior performance of our network comes from.

Q. Yu
Queen Mary University of London, Mile End, London, E1 4NS
Tel.: +44-2078826122
E-mail: q.yu@qmul.ac.uk

1 Introduction

Sketches are very intuitive to humans and have long been used as an effective communicative tool. With the proliferation of touchscreens, sketching has become easy and ubiquitous – we can sketch on phones, tablets and even watches. Research on sketches has consequently flourished in recent years, with a wide range of applications being investigated, including sketch recognition (Eitz et al, 2012; Schneider and Tuytelaars, 2014; Yu et al, 2015), sketch-based image retrieval (Eitz et al, 2011; Hu and Collomosse, 2013), sketch-based shape (Zou et al, 2015) or 3D model retrieval (Wang et al, 2015), and forensic sketch analysis (Klare et al, 2011; Ouyang et al, 2014).

Recognising free-hand sketches (i.e. objects such as cars drawn without any photo as reference) is an extremely challenging task (see Fig. 1). This is due to a number of reasons: (i) sketches are highly iconic and abstract, e.g., human figures can be depicted as stickmen; (ii) due to the free-hand nature, the same object can be drawn in totally different styles which results in varied levels of abstraction and deformation of sketches, e.g., a human figure sketch can be either a stickman or a portrait with fine details depending on the drawer; (iii) sketches lack visual cues, i.e., they consist of black and white lines instead of coloured pixels. A recent large-scale study on 20,000 free-hand sketches across 250 categories of daily objects puts human sketch recognition accuracy at 73.1% (Eitz et al, 2012), suggesting that the task is challenging even for humans.

Prior work on sketch recognition generally follows the conventional image classification paradigm, that is, extracting hand-crafted features from sketch images followed by feeding them to a classifier. Most hand-crafted features traditionally used for photos (such as HOG, SIFT and shape context) have been employed, which are often coupled with Bag-of-Words (BoW) to yield a final feature representations
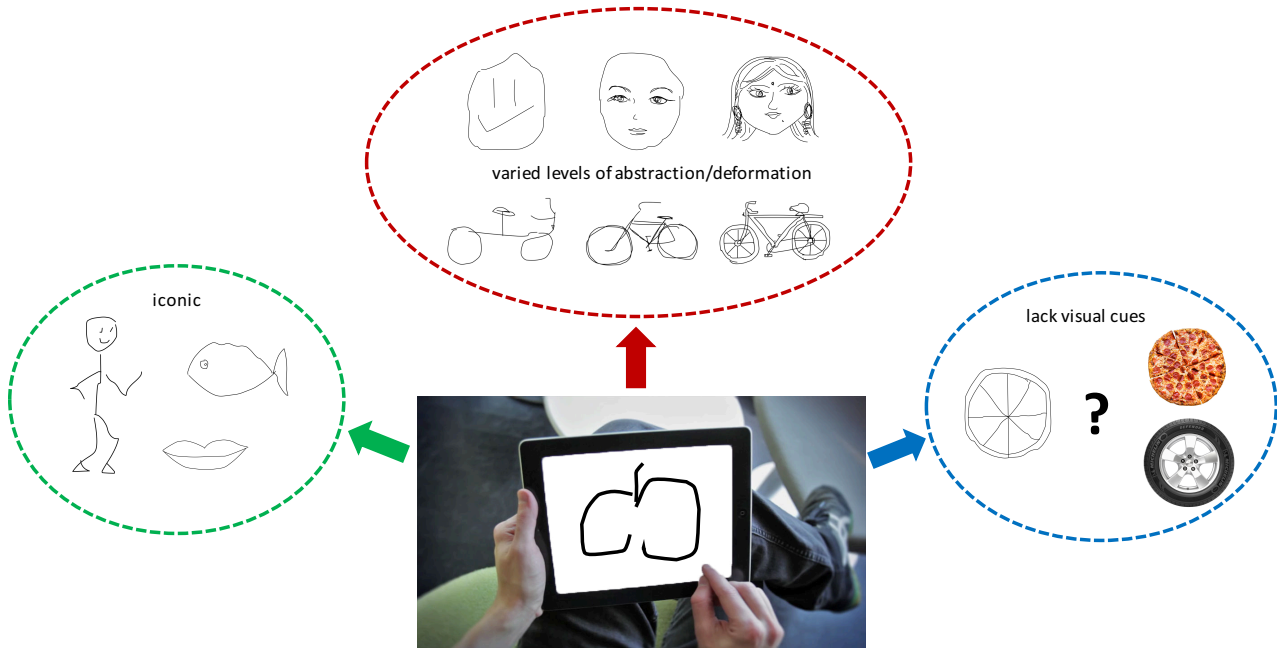
Fig. 1 Recognising a free-hand sketch is not easy due to a number of challenges

that can then be classified. However, existing hand-crafted features designed for photos do not account for the unique traits of sketches. More specifically, they ignore two key unique characteristics of sketches. First, a sketch is essentially an ordered list of strokes; they are thus sequential in nature. In contrast with photos that consist of pixels sampled all at once, a sketch is the result of an online drawing process. It had long been recognised in psychology (Johnson et al, 2009) that such sequential ordering is a strong cue in human sketch recognition, a phenomenon that is also confirmed by recent studies in the computer vision literature (Schneider and Tuytelaars, 2014). Second, free-hand sketches can be highly abstract and iconic and, coupled with varying drawing skills among different people, intra-class structure and appearance variations are often considerably larger than photos (see the examples of face and bicycle sketches from the TU-Berlin (Eitz et al, 2012) dataset in Fig. 1). Existing hand-crafted features such as HOG and classifiers such as SVMs are suited neither for capturing this large variation of abstraction and appearance variations, nor exploiting the ordered stroke structure of a sketch.

In this paper, we propose a novel deep neural network (DNN), Sketch-a-Net, for free-hand sketch recognition, which exploits the unique characteristics of sketch including multiple levels of abstraction and being sequential in nature. DNNs, especially deep convolutional neural networks (CNNs) have achieved tremendous successes recently in replacing representation hand-crafting with representation learning for a variety of vision problems (Krizhevsky et al, 2012; Simonyan and Zisserman, 2015). However, existing DNNs are

primarily designed for photos; we demonstrate experimentally that directly employing them for the sketch modelling problem produces little improvement over hand-crafted features, indicating special model architecture is required for sketches. One of the reasons for the failure of existing photo DNNs is that they typically require large amount of training data to avoid overfitting given millions of model parameters. However, the existing free-hand sketch datasets, the largest TU-Berlin dataset included, are far smaller than the photo datasets typically used for training photo DNNs, e.g. ImageNet. To this end, we design our model with the following considerations: (i) We introduce a novel CNN model with a number of architecture and learning parameter choices specifically for addressing the iconic and abstract nature of sketches. (ii) We develop a novel uniquely sketch-oriented data augmentation strategy that programatically deforms sketches both holistically and at local stroke level to generate a much larger and richer dataset for training. (iii) To deal with the variability in abstraction, and further enrich the training data, we also leverage another form of data augmentation by exploiting the temporal ordering of training strokes – and the tendency of people to sketch coarse detail first. In particular, we generate training sketches at various levels of abstraction by selectively removing detail strokes from the sketch which correspond to later-drawn strokes. (iv) A final network ensemble that works with various fusion schemes is formulated to further improves performance in practice.

Our contributions are summarised as follows:

- A representation learning model based on DNN is presented for sketch recognition in place of the conventional hand-crafted feature based sketch representations.
- We exploit sequential ordering information in sketches to capture multiple levels of abstraction naturally existing in free-hand sketches.
- We propose a simple but powerful deformation model that synthesises new sketches to generate richer training data.
- We further apply ensemble fusion and pre-training strategies to boost the recognition performance.
- We visualise what the model has learned to help gain deeper insight into why the model works for sketch recognition.

To validate the effectiveness of our Sketch-a-Net, experiments on the largest hand-free sketch benchmark, TU-Berlin (Eitz et al, 2012) have been carried out. The results show that our model significantly outperforms existing sketch recognition approaches and beats humans by a significant margin.

2 Related Work

Free-hand Sketch Recognition: Early studies on sketch recognition worked with professional CAD or artistic drawings as input (Lu et al, 2005; Jabal et al, 2009; Zitnick and Parikh, 2013; Sousa and Fonseca, 2009). Free-hand sketch recognition had not attracted much attention until very recently when a large crowd-sourced dataset was published in (Eitz et al, 2012). Free-hand sketches are drawn by non-artists using touch sensitive devices rather than purpose-made equipments; they are thus often highly abstract and exhibit large intra-class deformations. Most existing works (Eitz et al, 2012; Schneider and Tuytelaars, 2014; Li et al, 2015) use SVM as the classifier and differ only in what hand-crafted features borrowed from photos are used as representation. Li et al (2015) demonstrated that fusing different local features using multiple kernel learning helps improve the recognition performance. They also examined the performance of many features individually and found that HOG generally outperformed others. Yanık and Sezgin (2015) proposed to use active learning to achieve a target recognition accuracy while reducing the amount of manual annotation. Recently, Schneider and Tuytelaars (2014) demonstrated that Fisher Vectors, an advanced feature encoding successfully applied to image recognition, can be adapted to sketch recognition and achieve near-human accuracy (68.9% vs. 73.1% for humans on the TU-Berlin sketch dataset). Despite this progress, little effort has been made for either designing or learning representations specifically for sketches. Moreover, the role of sequential ordering in sketch recognition has generally received little attention. While the optical character recognition (OCR) community has exploited stroke ordering with

some success (Yin et al, 2013), exploiting sequential information is harder on sketches – handwriting characters have relatively fixed structural ordering; while sketches exhibit a much higher degree of intra-class variation in stroke order.

DNNs for Visual Recognition: Our Sketch-a-Net is a deep convolutional neural network. Artificial neural networks (ANN) are inspired by Hubel and Wiesel (1959), who proposed a biological model of cat's primary visual cortex, in which they found and named two distinct types of cells: simple cell and complex cell. These two types of cells correspond to convolution and pooling operators respectively in neural network models, and these two essential blocks have been commonly used by almost every model in this area since Neocognitron (Fukushima, 1980) was introduced. LeNet (Le Cun et al, 1990) employed backpropagation for training multi-layer neural networks (later re-branded as deep learning), and backpropagation and its varieties are now the standard training methods for such architecture.

ANN models were not so popular before five years ago, because (i) there are many hard-to-tune design choices e.g., activation functions, number of neurons, and number of layers, (ii) complicated NN models, esp. ones with many layers are difficult to train because of the vanishing gradient problem, (iii) NN advantages rely on sufficiently large training data and hence fast computers. These issues are progressively being overcome: (a) ReLU's efficacy has made it the dominant choice of activation function. (b) Layer-wise pre-training (e.g., RBM (Hinton et al, 2006)) can give a good initialisation for later supervised fine-tuning. Hessian-free optimization also partially alleviates the problem. (c) Most importantly, modern computers allow backpropagation to work on deeper networks within reasonable time (Schmidhuber, 2015), particularly when equipped with – now relatively affordable – GPUs. (d) Thanks to crowdsourcing, large-scale labelled data are now available in many areas.

Deep Neural Networks (DNNs) have recently achieved impressive performance for many recognition tasks across different disciplines. In particular, Convolutional Neural Networks (CNNs) have dominated top benchmark results on visual recognition challenges such as ILSVRC (Deng et al, 2009). An important advantage of DNNs, particularly CNNs, compared with conventional classifiers such as SVMs, lies with the closely coupled nature of presentation learning and classification (i.e., from raw pixels to class labels in a single network), which makes the learned feature representation maximally discriminative. Very recently, it was shown that even deeper networks with smaller filters (Simonyan and Zisserman, 2015) are preferable for photo recognition. Despite these advances, most existing image recognition DNNs are optimised for photos, ultimately making them perform sub-optimally on sketches. In this paper, we show that directly applying successful photo-oriented DNNs to sketches leads to little improvement over hand-crafted feature based

methods. In contrast, by designing an architecture for sketches (e.g., with larger rather than smaller filters) as well as data augmentation for designed sketches (e.g., exploiting stroke timing for varying training data abstraction), our Sketch-a-Net achieves state of the art recognition performance.

DNNs for Sketch Modelling: Very few existing works exploit DNNs for sketch modelling. One exception is the sketch-to-3D-shape retrieval work in Wang et al (2015). Designed for cross-domain (sketch to photo) matching, it uses a variant of Siamese network where the photo branch and sketch branch have the same architecture without any special treatment of the unique sketch images. In our work, a different recognition problem is tackled resulting in a very different network architecture. Importantly, the architecture and the model training strategy are carefully designed to suit the characteristics of free-hand sketch.

An early preliminary version of this work was published in Yu et al (2015). Compared to the earlier version of Sketch-a-Net (SN1.0) in Yu et al (2015), there are a number of modifications in the current network (SN2.0). Specifically, SN1.0 addressed the stroke ordering and sketch abstraction issues by – at both training and testing time: (i) segmenting the sketch in time, and processing the segments by different input channels in the DNN, and (ii) processing the sketch at multiple scales as different members of a network ensemble. In contrast, in SN2.0 we move all these considerations to the data augmentation stage. In particular, we use stroke timing and geometry information to define a simple but powerful data augmentation strategy that synthesises sketches at varying abstraction levels, and deforms them to achieve a much richer training set. The result is a simplified smaller model that is more broadly applicable to pixmaps at testing time. In addition, the newly introduced data augmentation strategies and simplifier network architecture (i.e. less model parameters) all help to alleviate the problem of overfitting to scarce sketch data. As a result, while SN1.0 just about beats humans on the sketch recognition task using the TU-Berlin benchmark (74.9% vs. 73.1%), SN2.0 beats humans by a large margin (77.95% vs. 73.1%). Further comparison of these two networks is discussed in Section 4.2.

3 Methodology

In this section we introduce the key features of our framework. We first detail our basic CNN architecture and outline the important considerations for Sketch-a-Net compared to the conventional photo-oriented DNNs (Sec. 3.1). We next explain how our simple but powerful data augmentation approach exploits stroke timing information to generate training sketches at various abstraction levels (Sec. 3.2). We then further demonstrate how stroke-geometry can be used to programatically generate more diverse sketches to further en-

hance the richness of the training set (Sec. 3.3). Fig. 2 illustrates our overall framework.

3.1 A CNN for Sketch Recognition

Sketch-a-Net is a deep CNN. Despite all efforts so far, it remains an open question how to design CNN architecture for a specific visual recognition task; but most recent recognition networks (Chatfield et al, 2014; Simonyan and Zisserman, 2015) now follow a design pattern of multiple convolutional layers followed by fully connected layers, as popularised by the work of Krizhevsky et al (2012).

Our specific architecture is as follows: first we use five convolutional layers, each with rectifier (ReLU) (LeCun et al, 2012) units, with the first, second and fifth layers followed by max pooling (Maxpool). The filter size of the sixth convolutional layer (index 14 in Table 1) is 7×7, which is the same as the output from previous pooling layer, thus it is precisely a fully-connected layer. Then two more fully connected layers are appended. Dropout regularisation (Hinton et al, 2012) is applied on the first two fully connected layers. The final layer has 250 output units corresponding to 250 categories (the number of unique classes in the TU-Berlin sketch dataset), upon which we place a softmax loss. The details of our CNN are summarised in Table 1. Note that for simplicity of presentation, we do not explicitly distinguish fully connected layers from their convolutional equivalents.

Most CNNs are proposed without explaining why parameters, such as filter size, stride, filter number, padding and pooling size, are chosen. Although it is impossible to exhaustively verify the effect of every free (hyper-)parameter, we discuss some points that are consistent with classic designs, as well as those that are specifically designed for sketches, thus considerably different from the CNNs targeting photos, such as AlexNet (Krizhevsky et al, 2012) and DeCAF (Donahue et al, 2015).

Commonalities between Sketch-a-Net and Photo-CNN

Filter Number: In both our Sketch-a-Net and recent photo-oriented CNNs (Krizhevsky et al, 2012; Simonyan and Zisserman, 2015), the number of filters increases with depth. In our case the first layer is set to 64, and this is doubled after every pooling layer (indicies: $3 \rightarrow 4$, $6 \rightarrow 7$ and $13 \rightarrow 14$) until 512.

Stride: As with photo-oriented CNNs, the stride of convolutional layers after the first is set to one. This keeps as much information as possible.

Padding: Zero-padding is used only in L3-5 (Indices 7, 9 and 11). This is to ensure that the output size is an integer number, as in photo-oriented CNNs (Chatfield et al, 2014).
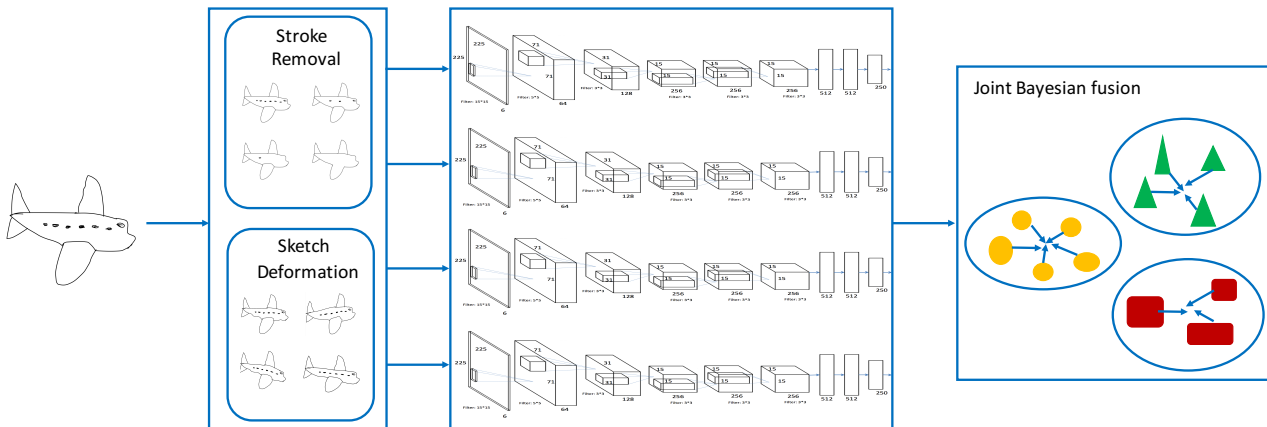
Unique Aspects in our Sketch-a-Net Architecture

Fig. 2 Illustration of our overall framework

Index	Layer	Type	Filter Size	Filter Num	Stride	Pad	Output Size
0		Input	-	-	-	-	225×225
1	L1	Conv	15×15	64	3	0	71×71
2		ReLU	-	-	-	-	71×71
3		Maxpool	3×3	-	2	0	35×35
4	L2	Conv	5×5	128	1	0	31×31
5		ReLU	-	-	-	-	31×31
6		Maxpool	3×3	-	2	0	15×15
7	L3	Conv	3×3	256	1	1	15×15
8		ReLU	-	-	-	-	15×15
9	L4	Conv	3×3	256	1	1	15×15
10		ReLU	-	-	-	-	15×15
11	L5	Conv	3×3	256	1	1	15×15
12		ReLU	-	-	-	-	15×15
13		Maxpool	3×3	-	2	0	7×7
14	L6	Conv(=FC)	7×7	512	1	0	1×1
15		ReLU	-	-	-	-	1×1
16		Dropout (0.50)	-	-	-	-	1×1
17	L7	Conv(=FC)	1×1	512	1	0	1×1
18		ReLU	-	-	-	-	1×1
19		Dropout (0.50)	-	-	-	-	1×1
20	L8	Conv(=FC)	1×1	250	1	0	1×1

Table 1 The architecture of Sketch-a-Net.

Larger First Layer Filters: The size of filters in the first convolutional layer might be the most sensitive parameter, as all subsequent processing depends on the first layer output. While classic networks use large 11×11 filters (Krizhevsky et al, 2012), the current trend of research (Zeiler and Fergus, 2014) is moving towards ever smaller filters: very recent (Simonyan and Zisserman, 2015) state of the art networks have attributed their success in large part to use of tiny 3×3 filters. In contrast, we find that larger filters are more appropriate for sketch modelling. This is because sketches lack texture information, e.g., a small round-shaped patch can be recognised as eye or button in a photo based on texture, but this is infeasible for sketches. Larger filters thus help to capture more structured context rather than textured information. To this end, we use a filter size of 15×15.

No Local Response Normalisation: Local Response Normalisation (LRN) (Krizhevsky et al, 2012) implements a form of lateral inhibition, which is found in real neurons. This is used pervasively in contemporary CNN recognition architectures (Krizhevsky et al, 2012; Chatfield et al, 2014; Simonyan and Zisserman, 2015). However, in practice LRN's benefit is due to providing "brightness normalisation". This is not necessary in sketches since brightness is not an issue in line-drawings. Thus removing LRN layers makes learning faster without sacrificing performance.

Larger Pooling Size: Many recent CNNs use 2×2 max pooling with stride 2 (Simonyan and Zisserman, 2015). It efficiently reduces the size of the layer by 75% while bringing some spatial invariance. However, we use the modification: 3×3 pooling size with stride 2, thus generating overlapping
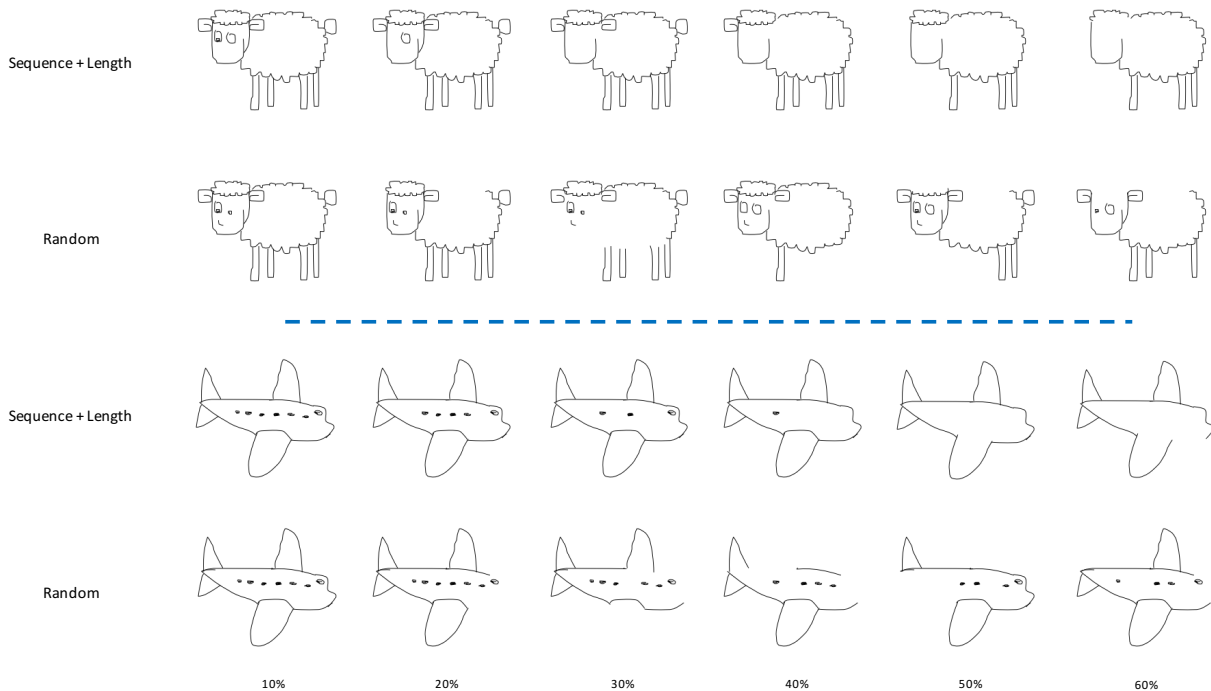
Fig. 3 Examples of ordered stroke removal for temporal modelling, with comparison to random stroke removal. From left to right, we show sketches after removing 10%-60% percent of strokes at a 10% interval

pooling areas (Krizhevsky et al, 2012). We found this brings $\sim 1\%$ improvement without much additional computation.

3.2 Exploiting Stroke order

Stroke ordering is key information associated with sketches drawn on touchscreens compared to conventional photos where all pixels are captured in parallel. Although this information exists in main sketch datasets such as TU-Berlin, existing work has generally ignored it. Clearly the specific stroke ordering of each sketch within the same category is different, but their sequences follow a general rule that the main outline will be drawn first and then followed by details (Eitz et al, 2012).

More specifically, a sketch is an ordered list of strokes, some of which convey broad aspects of the sketch, and others convey fine details. In general the broad-brush strokes are characterised by being drawn first, and by being longer – with the detail oriented strokes being later and shorter (Eitz et al, 2012). Importantly, the order of drawing strokes also corresponds to having different levels of abstraction: to draw a sketch of the same object category, some people would stop after some outline/coarse contours of the objects are drawn, whilst some other people prefer less abstract/more elaborate sketching style by adding more detailed/shorter strokes. We exploit these characteristics of sketch stroke or-

der to generate new sketches at multiple abstractions by progressively removing detail from each training sketch.

Specifically, given a sketch consisting of a set of N ordered strokes $S = \{s\}_i^N$ indexed by i, the order of the stroke and its length are used together to compute the probability of removing the i-th stroke as:

$$Pr_i = \frac{1}{Z} \cdot e^{\alpha * o_i} / e^{\beta * l_i}, \quad s.t. \ Z = \sum_i e^{\alpha * o} / e^{\beta * l} \quad (1)$$

where o_i and l_i are the sequence order and length of the i-th stroke, α and β are weights for these two factors, and Z is a normalisation constant to ensure a discrete probability distribution. Overall, the later and the shorter a stroke is, the more likely it will be removed.

Figure 3 illustrates how stroke removal can be used to increase abstraction by showing sketches with 10%-60% of strokes progressively removed with this method, with comparison to a random stroke removal alternative. Clearly with our stroke removal techniques, sketches become increasingly abstract as only longer and earlier strokes are retained, whereas the random scheme produces unrealistic looking sketches. Our approach provides a simple yet powerful way to exploit the unique properties of sketches to provide data augmentation as well as modelling sketch abstraction. An quantitative experiment that compares random stroke removal and the proposed can be found in Section 4.2.
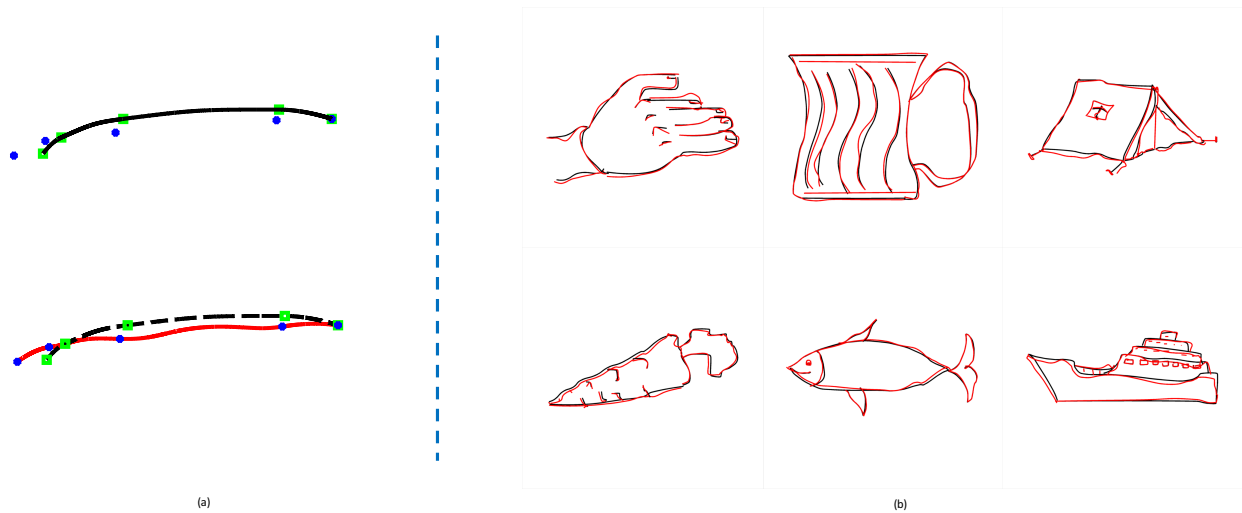
Fig. 4 Local sketch deformation. (a) demonstrates how we do local deformation. The green squares represent control points while the blue dots are the new deformed positions of these points. (b) shows several example sketches with local deformation. The original stroke are shown in black while the distorted ones are in red.
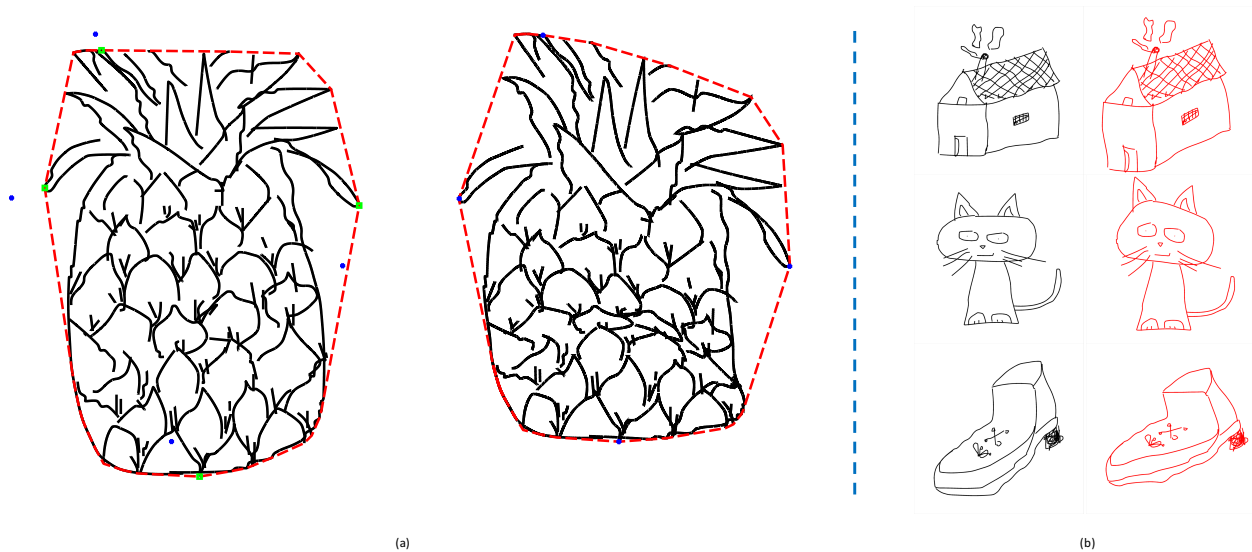
Fig. 5 Global sketch deformation. (a) demonstrates how we do global deformation. (b) shows the effect of global deformation. The left column are the original ones while the right are deformed sketches.

3.3 Sketch Deformation

The above stroke removal strategy is essentially a data augmentation strategy to deal with the naturally present variations of abstraction levels in sketches as well as enrich the scarce available sketch training data. Data augmentation is critical for learning DNNs – existing successful photo CNNs (Krizhevsky et al, 2012; Chatfield et al, 2014; Simonyan and Zisserman, 2015) obtain excellent performance in large part due to data augmentation achieved by translating, rotating, and scaling input images. In this section,

another data augmentation strategy is introduced which exploits another unique property of sketches compared to photos, that is, stroke geometry information is available, at least at training time. Specifically we present another simple but powerful data augmentation technique that exploits modelling of strokes to programatically generate a much richer array of training data.

Rationale behind Sketch Deformation Part of the challenge of sketches is the intra-class diversity: different people can draw exactly the same object in so many different ways. This intra-class diversity is largely due to variation in levels
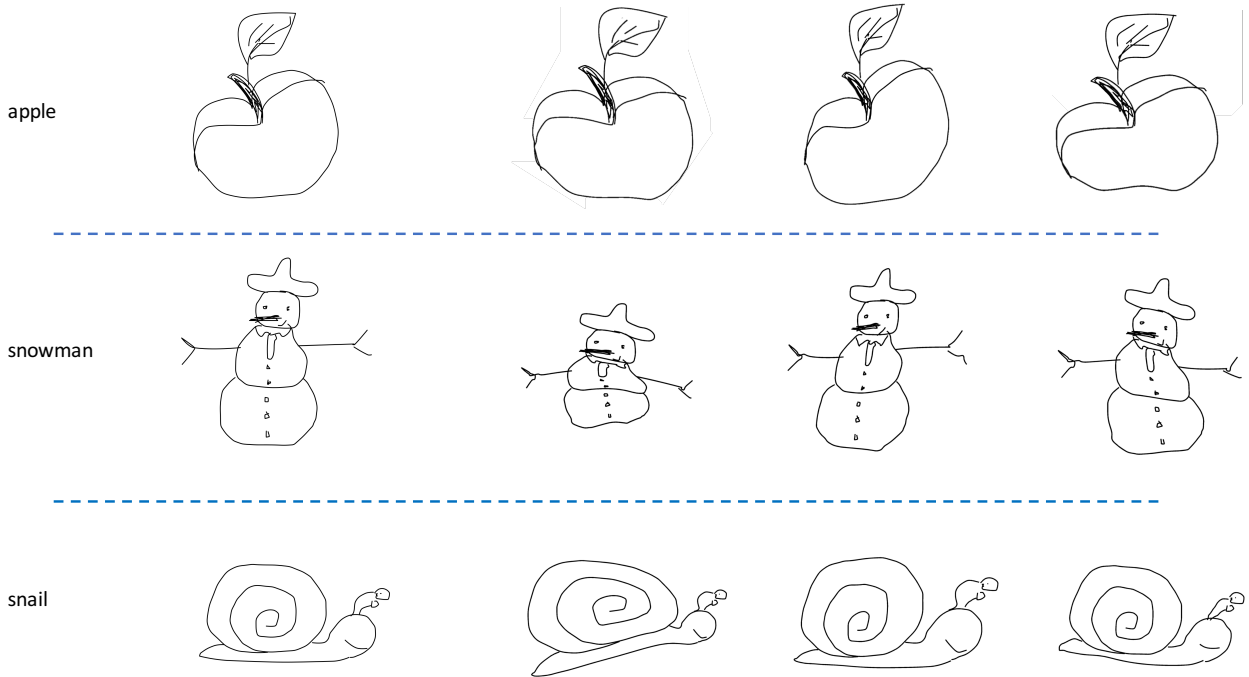
Fig. 6 Sketches with both local and global deformation. In each row, the first is the original sketch, and the following three are distorted sketches.

of deformation, curvature and length in individual strokes. This inspired us to simulate sketches drawn by different people by programatically modifying stroke and object geometry to generate more diverse variants of each input sketch. In particular, we deform each input sketch both locally and globally.

Local Deformation   Local deformation accounts for the stroke-level variations. In particular, when conducting local deformation, we first need to select pivot points. In vector graphic sketch data, such as the scalable vector graphics (SVG) used by the TU-Berlin dataset, each sketch S is represented as a list of strokes $S = \{s\}_i^N$ (i is the ordered stroke index). Each stroke in turn is composed of a set of segments: $s = \{b\}_j^{n_i}$ where each segment b_j is a cubic bezier spline

$$\mathbf{b}(t) = (1-t)^3\mathbf{p}_0 + 3(1-t)^2t\mathbf{p}_1 + 3(1-t)t^2\mathbf{p}_2 + t^3\mathbf{p}_3,$$
$$0 \le t \le 1 \quad (2)$$

and $\mathbf{p}_0$ and $\mathbf{p}_3$ are the endpoints of each bezier curve. Choosing the endpoints of each segment $\mathbf{p}_0$ and $\mathbf{p}_3$ as the pivot points for i-th stroke (green squares in Fig. 4(a)), we jitter the pivot points according to:

$$\mathbf{p} := \mathbf{p} + \epsilon, \quad s.t. \ \epsilon \sim \mathcal{N}(0, r\mathbf{I}) \quad (3)$$

where the standard deviation of the Gaussian noise is the ratio between the linear distance between endpoints and actual length of the stroke. This means that strokes with shorter length and smaller curvature are probabilistically deformed more, while long and curly strokes are deformed less. After getting the new position of pivot points (blue points in Fig. 4(a)), we then employ the Moving Least Squares (MLS) algorithm (Schaefer et al, 2006) to get new position of all points along the stroke. In Fig. 4(a), the red line indicates the distorted stroke while the black is the original one. Fig. 4(b) show several example sketches with local deformation.

Global Deformation   In addition to locally deforming individual strokes, we also globally deform the sketch as a whole. First we apply convex hull algorithm to find the outline shape of the sketch (red outline in Fig. 5(a)), and use the vertices of the convex polygon whose x/y coordinate is the smallest/largest as the pivot points. As with local deformation, we apply Eq. 3 to get their new positions and use MLS to compute new position of all points in the sketch. As shown in Fig. 5(a), green points indicate the pivot points for global deformation and blue ones are pivot points after translation. Through comparing two red convex polygons, we can see the effect of global deformation. Fig. 5(b) displays some sketches with global deformation.

In our experiment, we combine these two kinds of deformation together, first applying local deformation and followed by global deformation. Fig. 6 shows the deformation effect. Our experiments show that both deformation strategies contribute to the final recognition performance (see Sec. 4).

3.4 Ensemble Fusion

To further improve recognition performance in practice, we explore different fusion methods, including a re-purposed joint Bayesian solution. One common practice is to concatenate the CNN learned representations in each network and feed them to a downstream classifier (Donahue et al, 2015), which we called "feature-level fusion". Another common fusion strategy is score-level fusion, which simply averages the softmax outputs of the ensemble. However, these two fusion strategies treat each network equally and thus cannot explicitly exploit inter-model correlations.

Joint Bayesian (JB) (Chen et al, 2012), initially designed for face verification, were designed to fully explore such inter-model correlations. Here, we re-purpose JB for classification. Let each x represent the $4 \times 512 = 2048D$ concatenated feature vector from our network ensemble. Training: Using this activation vector as a new representation for the data, we train the JB model, thus learning a good metric that exploits intra-ensemble correlation. Testing: Given the activation vectors of train and test data, we compare each test point to the full train set using the likelihood test. With this metric to compare test and train points, final classification is achieved with K-Nearest-Neighbour (KNN) matching[1]. Note that in this way each feature dimension from each network is fused together, implicitly giving more weight to more important features, as well as finding the optimal combination of different features of different models.

We evaluate these three fusion strategies (feature-level fusion, score-level fusion and joint Bayesian fusion). Comparison results and further discussion can be found later in Section 4.2.

4 Experiments

4.1 Dataset and Settings

Dataset: We evaluate our model on the TU-Berlin sketch dataset (Eitz et al, 2012), which is the largest and now the most commonly used human sketch dataset. It contains 250 categories with 80 sketches per category. It was collected on Amazon Mechanical Turk (AMT) from 1,350 participants. We rescaled all images to 256×256 pixels in order to make it comparable with previous work. Also following previous work we performed 3-fold cross-validation within this dataset (2 folds for training and 1 for testing).

Data Augmentation: Data augmentation is critical for CNNs to reduce the risk of overfitting. We first performed classical

---

[1] We set $k = 30$ in this work and the regularisation parameter of JB is set to 1. For robustness at test time, we also take 10 crops and reflections of each train and test image (Krizhevsky et al, 2012). This inflates the KNN train and test pool by 10, and the crop-level matches are combined to image predictions by majority voting.

data augmentation by replicating the sketches with a number of transformations. Specifically, for each input sketch, we did horizontal reflection and systematic combinations of horizontal and vertical shifts (up to 32 pixels). These conventional data augmentation strategies will increase the training data size by a factor of $32 \times 32 \times 2$ fold. Our stroke removal (see Sec. 3.2) and sketch deformation strategies (see Sec. 3.3) produce a further $13\times$ more training data (each training sample, has 6 synthesised sketches with stroke removal + 6 sketches with sketch deformation + the original sketch). Thus, when using two thirds of the data for training, the total pool of training instances is $13 \times (20000 \cdot 0.67) \times (32 \cdot 32 \cdot 2)$, increasing the training set size by a factor of $13 \times 32 \times 32 \times 2 = 26,624$.

Pre-training: Apart from 'dropout' and data augmentation, another strategy to avoid overfitting is via pre-training on a (larger) auxiliary dataset. The key challenge here is how to create the auxiliary data source which are similar in nature to our sketch data – photos are abundant but sketches are hard to find. It is noted that, consisting of black and white lines, to some degree, sketch is similar to object edges extracted from photos. We therefore take the Sketch-a-Net architecture, and pre-train it using the edge maps extracted from the ImageNet-1K photos (Deng et al, 2009). All the edge maps are extracted from bounding box areas by using the edge detection method in Zitnick and Dollár (2014); therefore only images with bounding boxes provided are used. This step gives our model a better initialisation compared with being initialised randomly. Our experiments show that pre-training helps to achieve better performance (Tab. 6).

Settings: We implemented our network using Caffe (Jia et al, 2014). The initial learning rate is set to 0.001, and mini-batch to 135. During training, each sketch is randomly cropped to a 225×225 sub-image. Both the novel data augmentation methods described in Sec. 3.2 and Sec. 3.3, and traditional data augmentation are applied. We train a 4-network ensemble and then use JB to fuse them. In particular, we extract the output of the penultimate layer (fc7) from 4 networks and concatenate them as the final feature representation. We then employ JB and get the classification result (as described in Sec. 3.4). More specifically, at testing time, we do multi-view testing by cropping each sample 10 times (4 corner, 1 centre and flipped). Each of these crops is put through the ensemble and classified by JB. The views are then fused by majority vote. Overall, the final feature representation of each testing sample is a $10 \times (4 \cdot 512D)$ matrix.

Competitors: We compared our Sketch-a-Net model with a variety of alternatives. They can be categorised into two groups. The first group follows the conventional handcrafted features+ classifier pipeline. These included the **HOG-SVM** method (Eitz et al, 2012), structured **ensemble matching**

(Li et al, 2013), **multi-kernel SVM** (Li et al, 2015), and the current state-of-the-art Fisher Vector Spatial Pooling **(FV-SP)** (Schneider and Tuytelaars, 2014). The second group used DNNs. These included **AlexNet** (Krizhevsky et al, 2012) and **LeNet** (LeCun et al, 2012). AlexNet is a large deep CNN designed for classifying ImageNet LSVRC-2010 (Deng et al, 2009) images. It has five convolutional layers and 3 fully connected layers. We used two versions of AlexNet: (i) **AlexNet-SVM**: following common practice (Donahue et al, 2015), it was used as a pre-trained feature extractor, by taking the second 4096D fully-connected layer of the ImageNet-trained model as a feature vector for SVM classification. (ii) **AlexNet-Sketch**: we re-trained AlexNet for the 250-category sketch classification task, i.e. it was trained using the same data as our Sketch-a-Net. Although LeNet is quite old, we note that it is specifically designed for handwritten digits rather than photos. Thus it is potentially more suited for sketches than the photo-oriented AlexNet. Finally, the earlier version of Sketch-a-Net (Yu et al, 2015), denoted SN1.0, was also compared.

4.2 Results

Comparative Results: We first report the sketch recognition results of our full Sketch-a-Net, compared to state-of-the-art alternatives as well as humans in Table 2. The following observations can be made: (i) Sketch-a-Net significantly outperforms all existing methods purposefully designed for sketch (Eitz et al, 2012; Li et al, 2013; Schneider and Tuytelaars, 2014), as well as the state-of-the-art photo-oriented CNN model (Krizhevsky et al, 2012) re-purposed for sketch; (ii) we show that an automated sketch recognition model can surpass human performance on sketch recognition (77.95% by our Sketch-a-Net vs. 73.1% for humans by a clear margin based on the study in (Eitz et al, 2012)); (iii) Sketch-a-Net is superior to AlexNet, despite being much smaller with only 14% of the total number of parameters of AlexNet. This verifies that new network design is required for sketch im-

Models	Accuracy
HOG-SVM (Eitz et al, 2012)	56%
Ensemble (Li et al, 2013)	61.5%
MKL-SVM (Li et al, 2015)	65.8%
FV-SP (Schneider and Tuytelaars, 2014)	68.9%
AlexNet-SVM (Krizhevsky et al, 2012)	67.1%
AlexNet-Sketch (Krizhevsky et al, 2012)	68.6%
LeNet (LeCun et al, 2012)	55.2%
SN1.0 (Yu et al, 2015)	74.9%
Our Full Model	**77.95%**
Humans (Eitz et al, 2012)	73.1%

Table 2 Comparitive results on sketch recognition

ages. In particular, it is noted that either trained using the larger ImageNet data (67.1%) or the sketch data (68.6%), AlexNet cannot beat the best hand-crafted feature based approach (68.9% of FV-SP); (iv) among the deep DNN based models, the performance of LeNet (55.2%) is the weakest. Although designed for handwriting digit recognition, a task similar to that of sketch recognition, the model is much simpler and shallower. This suggests that a deeper model is necessary to cope with the larger intra-class variations exhibited in sketches; (v) Compared with the earlier version of Sketch-a-Net (SN1.0), the improved SN2.0 model is clearly superior thanks to the lower model complexity and more richly augmented training data.

Upon closer category-level examination, we found that Sketch-a-Net tends to perform better at fine-grained object categories. This indicates that Sketch-a-Net learned a more discriminative feature representation capturing finer details than conventional hand-crafted features, as well as humans. For example, for 'seagull' and 'pigeon', both of which belong to the coarse semantic category of 'bird'. Sketch-a-Net obtained an average accuracy of 40.4% while human only achieved 16.9%. In particular, the category 'seagull', is the worst performing category for human with an accuracy of just 2.5%, since it was mostly confused with other types of birds. In contrast, Sketch-a-Net yielded 26.92% for 'seagull' which is nearly 10 times better. Further insights on how this performance improvement is achieved will be provided via model visualisation later.

Contributions of Individual Components: Compared to conventional photo-oriented DNNs such as AlexNet, our Sketch-a-Net has four distinct features: (i) the network architecture (see Sec. 3.1), (ii) the stroke removal utilising stroke ordering (see Sec. 3.2) to synthesise variable levels of abstraction, (iii) the sketch deformation approach to deal with large sketching appearance variations (see Sec. 3.3), and (iv) the joint Bayesian (JB) fusion with a network ensemble (see Sec. 3.4). In this section, we evaluate the contributions of each new feature. Specifically, we examined five stripped-down versions of our full model: single Sketch-a-Net with all features but no JB/ensemble (**SN-NoJB**), Sketch-a-Net only with stroke removal (**SN-SR**) which is how we exploit temporal information, Sketch-a-Net only with sketch deformation (**SN-SD**) which accounts for varied levels of abstraction and our basic Sketch-a-Net model (**SN-vanilla**). The results in Table 3 show that all four strategies contribute to the final strong performance of Sketch-a-Net. In particular, (i) the improvement of SN-vanilla over AlexNet-Sketch shows that our sketch-specific network architecture is effective; (ii) SN-SD and SN-SR improved the performance of SN by 3% and 2%, respectively, indicating that both stroke removal and sketch deformation strategy worked; (iii) the best result is achieved when all four strategies are combined.

Our Full Model	SN-NoJB	SN-SD	SN-SR	SN-vanilla	AlexNet-Sketch
77.95%	77.06%	75.52%	74.57%	72.20%	68.60%

Table 3 Evaluation on the contributions of individual components of Sketch-a-Net.

Global+Local	Global	Local
75.52%	75.40%	73.86%

Table 4 Evaluation of novel data augmentation strategies: Global and Local deformations.

Ordered removal	74.57%
Random removal	73.85%

Table 5 Comparison results of different stroke removal strategies.

With Pre-training	77.06%
Without Pre-training	76.06%

Table 6 Contributions of pre-training

Feature Fusion	Score Fusion	joint Bayesian
77.12%	77.82%	77.95%

Table 7 Comparison of different fusion strategies.

Random vs. Ordered Removal: To quantitatively verify the value of our proposed stroke removal technique, we also trained a network using augmented sketches that had undergone random stroke removal. The experimental results with comparison to the proposed stroke removal technique can be found in Table 5. It is clear to see that our proposed stroke removal strategy can better simulate the original sketch at multiple abstraction levels, resulting in higher performance.

Local vs. Global Deformation: In our sketch deformation approach, we apply deformation at both local and global levels. To find out which part brings about more improvement for our model, in this experiment, we compared the contributions of different deformation strategies. Specifically, we trained Sketch-a-Net with the data processed only by local or global deformation, and then compare their performance. From the results listed in Table 4, we can see that both types of deformations help, but the global deformation brings a larger improvement. This makes sense since local deformation provides subtler stroke-level variations while global deformation modifies the whole instance.

Effects of Pre-training: Table 6 shows the contribution of pre-training. The results show that our model benefits from the pre-training even though the auxiliary data was from a different domain (extracted edges from photo images).

Comparison of Different Fusion Strategies: Given an ensemble of Sketch-a-Net (4 models in our experiments) obtained by varying the mini-batch orders, various fusion strategies can be adopted for the final classification task. Table 7 compares our joint Bayesian fusion method with the alternative feature-level and score-level fusion methods.

Note that all the models we use here were trained with our stroke removal and sketch deformation data augmentation, as well as the pre-training strategy. For feature-level fusion, we treated each single network as a feature extractor, and concatenated the 512D outputs of their penultimate layers into a single feature. We then trained a linear SVM based on this $4 \times 512 = 2048$D feature vector. For score-level fusion, we averaged the 250D softmax probabilities of each network in the ensemble to make a final prediction. For JB fusion, we took the same 2048D concatenated feature vector used by feature-level fusion, but performed KNN matching with JB similarity metric, rather than SVM classification.

Although JB fusion can still achieve the best performance among the three fusion strategies, it is interesting to note that it does not outperform the other two simpler baselines as much as reported in Yu et al (2015). This is because the single branch in our current model performs much better than before. JB was initially designed for face identification/verification, when it was used for model fusion: combing multiple models (e.g., neural networks) together (model fusion). However it has an implicit assumption that each model should be reasonably different. This is usually obtained by using different resolutions or face regions. However, in our current case, the trained neural networks are not only individually better, but also closer to each other in terms of both top layer (score-level fusion) and penultimate layer (feature-level fusion), so there is very little room for JB to improve.

Comparison of SN1.0 and SN2.0: This work is an extension of Yu et al (2015). In broad terms, SN2.0 proposed in this work differs with SN1.0 (Yu et al, 2015) in (i) how stroke ordering information is exploited, and (ii) how data argumentation is achieved via local and global stroke deformation. As a result, SN2.0 can achieve better overall recognition performance even without ensemble fusion, which was previously verified to work well for SN1.0. A detailed comparison between the two networks is offered in Table 8. In particular, (i) We replaced the multi-channel multi-scale model with current single-channel single-scale model, making the whole architecture much simpler and faster for training; (ii) Correspondingly, the total number of parameters has reduced from 42.5 million (8.5 million/network $\times$ 5 networks in SN1.0) to 8.5 million; and (iii) The number of training data is also reduced a lot. Apart from traditional data augmentation (i.e., rotation, translation etc.), we replicated the training data 30 times (6 channel $\times$ 5 scales) in SN1.0 while we now only replicate it 13 times (each sketch generates 13 inputs, including 1 original sketch, 6 new sketches
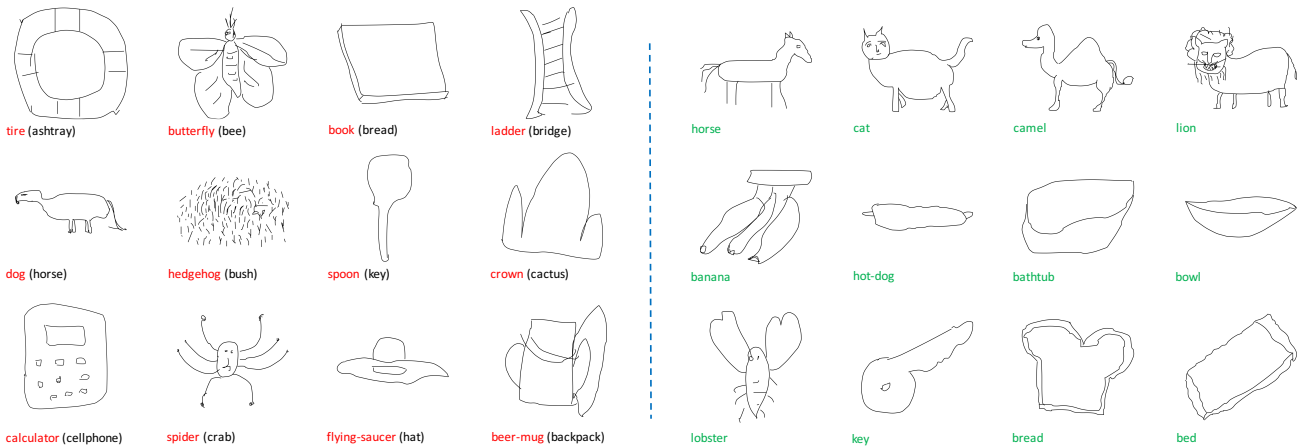
Fig. 7 Qualitative illustration of recognition successes (green) and failures (red).
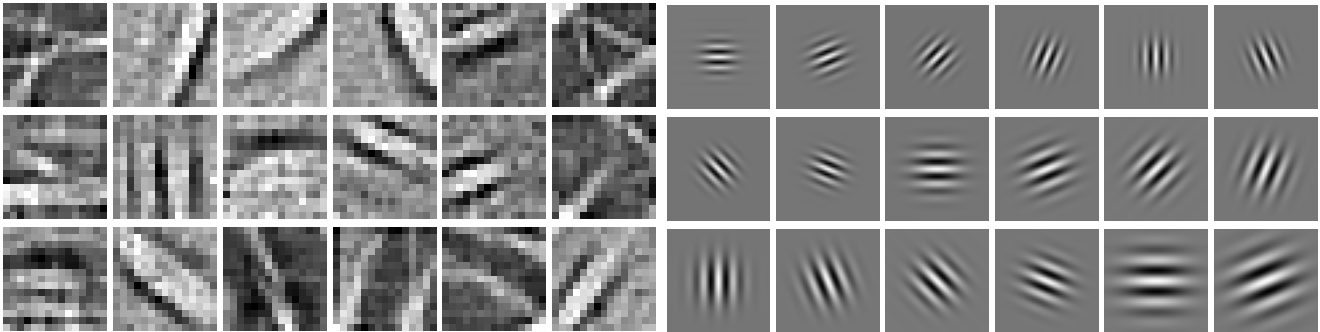

Fig. 8 Visualisation of the learned filters. Left: randomly selected filters from the first layer in our model; right: the real parts of some Gabor filters

Table 8 Comparison of SN1.0 and SN2.0

| Aspect	SN1.0	SN2.0 (without JB)
Architecture	multi-channel & multi-scale	single-channel & single-scale
Num. of parameters	42.5 million	8.5 million
Multiple of training data	30	13
Performance	74.9%	77.06%

with stroke removal and 6 with sketch deformation). The performance of SN2.0 (without joint Bayesian fusion) is more than 2% higher than SN1.0. This indicates the efficacy of our new data augmentation strategies.

Qualitative Results: Figure 7 shows some qualitative results. Some examples of surprisingly tough successes are shown in green. Mistakes made by the network (red) (intended category of the sketches in black) are very reasonable. One would expect humans would make similar mistakes. The clear challenge level of their ambiguity demonstrates why reliable sketch-based communication is hard even for humans.

What Has Been Learned by Sketch-a-Net: As illustrated in Fig. 8, the filters in the first layer of Sketch-a-Net (Fig. 8(left)) learn something very similar to the biologically plausible Gabor filters (Fig. 8(right)) (Gabor, 1946). This is interesting because it is not obvious that learning from sketches

should produce such filters, as their emergence is typically attributed to learning from the statistics of natural images (Olshausen and J., 1996; Stollenga et al, 2014).

To visualise a CNN by showing its filters is only meaningful for the first convolutional layer, because it is directly applied on pixels. It is hard to get intuition from observing higher level filters as they are applied on features. So we use the deconvolution method proposed by Zeiler and Fergus (2014) to visualise the filters of higher-level convolutional layers. To do this, we searched for patches that maximise the response of specific filters on the training set, and then uses deconvolution to show which part of the patch contributed most to the activation. Fig. 9 shows the visualisation of some selected filters ranging from layer conv2 to conv5, and for each filter we used 9 patches from different images for visualisation. We can see that features of higher level convolutional layers are more semantic: features of conv2
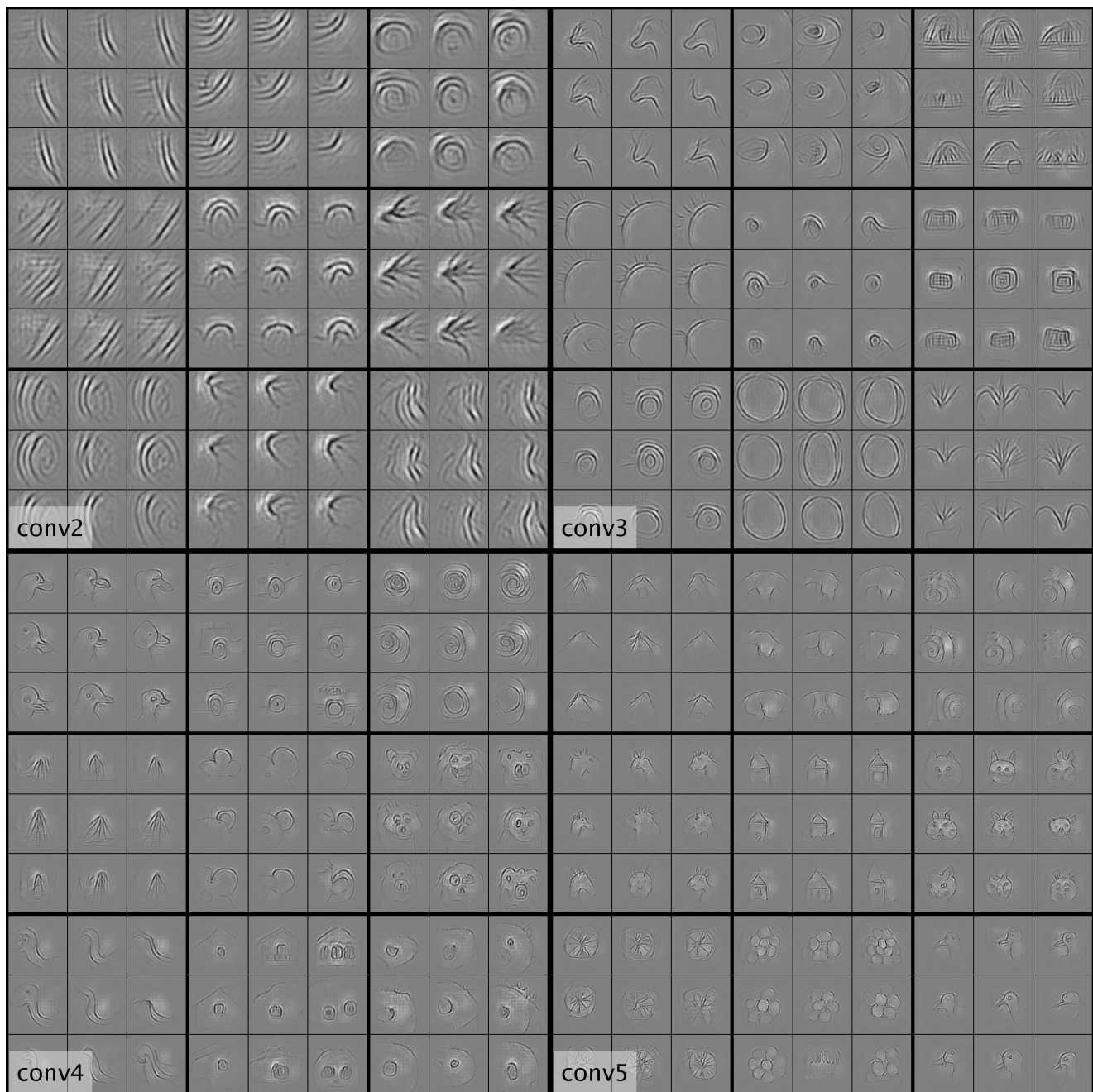
Fig. 9 Visualisation of the learned filters by deconvolution. Through visualisation of the filters by deconvolution, we can see that filter of higher-level layer are modeling more complex concepts. For example, what neurons represented in conv2 are basic building blocks to compose other concepts like lines, circles and textures; layer conv3 learns more mid-level concepts or object parts, like eye and wheel; and in conv4 and conv5, neurons are representing complex concepts like head, roof, and body.

are just texture and simple patterns; object parts like eye of animals begin to emerge in conv3; and in conv4 and conv5, the filter can capture more complex concepts like head and tent. The scale of learned concepts also becomes larger as we goes deeper, since the receptive field of the neurons has become larger. In particular, it is noted that bird head like filters were learned as early as conv2 and where progressively refined as the network goes deeper. This partially ex-

plains why the model performs particularly well on the fine-grained bird recognition tasks compared to humans.

It is also interesting to see the CNN trained on sketches behaves somehow different from the model trained with natural images, e.g., from AlexNet trained on ImageNet (Zeiler and Fergus, 2014). As sketches are more abstract and discard many details, object across categories are more likely to share mid-level representations, i.e, a single filter can be

used by multiple object classes. For example, in conv4, wheel-like feature can shared both by donut and cameras; eye-like feature can both shared by animals and house, and in conv5, clustered blobs can shared by grape and flower.

Running cost: We trained our 4-network ensemble for 180K iterations each, with each instance undergoing random data augmentation during each iteration. This took roughly 20 hours using a NVIDIA K40-GPU.

5 Conclusion

We have proposed a deep neural network based sketch recognition model, which we call Sketch-a-Net, that beats human recognition performance by 5% on a large scale sketch benchmark dataset. Key to the superior performance of our method lies with the specifically designed network model and several novel training strategies that accounts for unique characteristics found in sketches that were otherwise unaddressed in prior art. The learned sketch feature representation could benefit other sketch-related applications such as sketch-based image retrieval and automatic sketch synthesis, which could be interesting venues for future work.

References

Chatfield K, Simonyan K, Vedaldi A, Zisserman A (2014) Return of the devil in the details: Delving deep into convolutional nets. In: BMVC

Chen D, Cao X, Wang L, Wen F, Sun J (2012) Bayesian face revisited: A joint formulation. In: ECCV

Deng J, Dong W, Socher R, Li L, Li K, Fei-Fei L (2009) Imagenet: A large-scale hierarchical image database. In: CVPR

Donahue J, Jia Y, Vinyals O, Hoffman J, Zhang N, Tzeng E, Darrell T (2015) Decaf: A deep convolutional activation feature for generic visual recognition. In: ICML

Eitz M, Hildebrand K, Boubekeur T, Alexa M (2011) Sketch-based image retrieval: Benchmark and bag-of-features descriptors. TVCG

Eitz M, Hays J, Alexa M (2012) How do humans sketch objects? In: SIGGRAPH

Fukushima K (1980) Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics

Gabor D (1946) Theory of communication. part 1: The analysis of information. Journal of the Institution of Electrical Engineers-Part III: Radio and Communication Engineering

Hinton GE, Osindero S, Teh YW (2006) A fast learning algorithm for deep belief nets. Neural Computation

Hinton GE, Srivastava N, Krizhevsky A, Sutskever I, Salakhutdinov RR (2012) Improving neural networks by preventing co-adaptation of feature detectors. In: arXiv preprint arXiv:1207.0580

Hu R, Collomosse J (2013) A performance evaluation of gradient field hog descriptor for sketch based image retrieval. CVIU

Hubel DH, Wiesel TN (1959) Receptive fields of single neurons in the cat's striate cortex. Journal of Physiology

Jabal MFA, Rahim MSM, Othman NZS, Jupri Z (2009) A comparative study on extraction and recognition method of cad data from cad drawings. In: International Conference on Information Management and Engineering

Jia Y, Shelhamer E, Donahue J, Karayev S, Long J, Girshick R, Guadarrama S, Darrell T (2014) Caffe: Convolutional architecture for fast feature embedding. arXiv preprint arXiv:14085093

Johnson G, Gross MD, Hong J, Yi-Luen Do E (2009) Computational support for sketching in design: a review. Foundations and Trends in Human-Computer Interaction

Klare BF, Li Z, Jain AK (2011) Matching forensic sketches to mug shot photos. TPAMI

Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: NIPS

Le Cun Y, Boser B, Denker JS, Henderson D, Howard RE, Hubbard W, Jackel LD (1990) Handwritten digit recognition with a back-propagation network. In: NIPS

LeCun Y, Bottou L, Orr GB, Müller K (2012) Efficient back-prop. Neural networks: Tricks of the trade

Li Y, Song Y, Gong S (2013) Sketch recognition by ensemble matching of structured features. In: BMVC

Li Y, Hospedales TM, Song Y, Gong S (2015) Freehand sketch recognition by multi-kernel feature learning. CVIU

Lu T, Tai C, Su F, Cai S (2005) A new recognition model for electronic architectural drawings. Computer-Aided Design

Olshausen BA, J FD (1996) Emergence of simple-cell receptive field properties by learning a sparse code for natural images. Nature

Ouyang S, Hospedales T, Song Y, Li X (2014) Cross-modal face matching: beyond viewed sketches. In: ACCV

Schaefer S, McPhail T, Warren J (2006) Image deformation using moving least squares. TOG

Schmidhuber J (2015) Deep learning in neural networks: An overview. Neural Networks

Schneider RG, Tuytelaars T (2014) Sketch classification and classification-driven analysis using fisher vectors. In: SIGGRAPH Asia

Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. In: ICLR

Sousa P, Fonseca MJ (2009) Geometric matching for clip-art drawing retrieval. Journal of Visual Communication and Image Representation

Stollenga MF, Masci J, Gomez F, Schmidhuber J (2014) Deep networks with internal selective attention through feedback connections. In: NIPS

Wang F, Kang L, Li Y (2015) Sketch-based 3d shape retrieval using convolutional neural networks. In: CVPR

Yanık E, Sezgin TM (2015) Active learning for sketch recognition. Computers & Graphics

Yin F, Wang Q, Zhang X, Liu C (2013) Icdar 2013 chinese handwriting recognition competition. In: International Conference on Document Analysis and Recognition

Yu Q, Yang Y, Song YZ, Xiang T, Hospedales TM (2015) Sketch-a-net that beats humans. In: BMVC

Zeiler M, Fergus R (2014) Visualizing and understanding convolutional networks. In: ECCV

Zitnick CL, Dollár P (2014) Edge boxes: Locating object proposals from edges. In: ECCV

Zitnick CL, Parikh D (2013) Bringing semantics into focus using visual abstraction. In: CVPR

Zou C, Huang Z, Lau RW, Liu J, Fu H (2015) Sketch-based shape retrieval using pyramid-of-parts. arXiv preprint arXiv:150204232