Sketch-a-Net that Beats Humans

Qian Yu*
q.yu@qmul.ac.uk
Yongxin Yang*
yongxin.yang@qmul.ac.uk
Yi-Zhe Song
yizhe.song@qmul.ac.uk
Tao Xiang
t.xiang@qmul.ac.uk
Timothy M. Hospedales
t.hospedales@qmul.ac.uk

School of Electronic Engineering and Computer Science
Queen Mary, University of London
London, E1 4NS
United Kingdom

Abstract

We propose a multi-scale multi-channel deep neural network framework that, for the first time, yields sketch recognition performance surpassing that of humans. Our superior performance is a result of explicitly embedding the unique characteristics of sketches in our model: (i) a network architecture designed for sketch rather than natural photo statistics, (ii) a multi-channel generalisation that encodes sequential ordering in the sketching process, and (iii) a multi-scale network ensemble with joint Bayesian fusion that accounts for the different levels of abstraction exhibited in free-hand sketches. 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 photo or sketch. Our network on the other hand not only delivers the best performance on the largest human sketch dataset to date, but also is small in size making efficient training possible using just CPUs.

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 a much easier undertaking for many – 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 [6, 21], sketch-based image retrieval [8, 9], sketch-based 3D model retrieval [25], and forensic sketch analysis [12, 20].

Recognising free-hand sketches (e.g. asking a person to draw a car without any instance of car as reference) is an extremely challenging task. 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 with hugely varied levels of detail/abstraction, e.g., a human figure sketch can be either a stickman or a portrait with fine

* These authors contributed equally to this work

Pages 7.1-7.12
DOI: https://dx.doi.org/10.5244/C.29.7

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% [6], 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 that can then be classified. However, existing hand-crafted features designed for photos do not account for the unique abstract and sparse nature of sketches. Furthermore, they ignore a key unique characteristics of sketches, that is, 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 [11] 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 [21]. However, none of the existing approaches attempted to embed sequential ordering of strokes in the recognition pipeline even though that information is readily available.

In this paper, we propose a novel deep neural network (DNN), Sketch-a-Net, for free-hand sketch recognition, which is specifically designed to accommodate the unique characteristics of sketches 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 [13, 22]. 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. To this end, our Sketch-a-Net has three key features that distinguish it from the existing DNNs: (i) a number of model architecture and learning parameter choices specifically for addressing the iconic and abstract nature of sketches; (ii) a multi-channel architecture designed to model the sequential ordering of strokes in each sketch; and (iii) a multi-scale network ensemble to address the variability in abstraction and sparsity, followed by a joint Bayesian fusion scheme to exploit the complementarity of different scales. The overall model is small in size, being 7 times smaller than the classic AlexNet [13] in terms of the number of parameters, therefore making it efficient to train independently of special hardware, i.e. GPUs.

Our contributions are summarised as follows: (i) for the first time, a representation learning model based on DNN is presented for sketch recognition in place of the conventional hand-crafted feature based sketch representations; (ii) we demonstrate how sequential ordering information in sketches can be embedded into the DNN architecture and in turn improve sketch recognition performance; (iii) we propose a multi-scale network ensemble that fuses networks learned at different scales together via joint Bayesian fusion to address the variability of levels of abstraction in sketches. Extensive experiments on the largest hand-free sketch benchmark dataset, the TU-Berlin sketch dataset [6], show that our model significantly outperforms existing approaches and can even beat humans at sketch recognition.

2 Related Work

Free-hand Sketch Recognition: Early studies on sketch recognition worked with professional CAD or artistic drawings as input [10, 18, 23, 28]. Free-hand sketch recognition had not attracted much attention until very recently when a large crowd-sourced dataset was published in [6]. 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 [6, 17, 21] use SVM as the classifier and differ only in what hand-crafted features borrowed from photos are used as representation. Li et al. [17] 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. Very recently, Schneider and Tuytelaars [21] demonstrated that Fisher Vectors, an advanced feature representation scheme 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 these great efforts, no attempt was made thus far for either designing or learning feature representations specifically for sketches. Moreover, the role of sequential ordering in sketch recognition remains unaddressed. In this paper, we turn to DNNs which have shown great promise in many areas of computer vision [13, 22] for representation learning. Our learned representation uniquely exploits the sequential ordering information of strokes in a sketch and is able to cope with multiple levels of abstraction in the same sketch category. Note that the optical character recognition (OCR) community has exploited stroke ordering with some success [26], yet the problem of encoding sequential information is harder on sketches – handwriting characters have relatively fixed structural ordering therefore simple heuristics often suffice; sketches on the the other hand exhibit a much higher degree of intra-class variation in stroke ordering, which motivates us to resort to the powerful DNNs to learn the most suitable sketch representation.

DNNs for Visual Recognition: 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 [3]. When first introduced in the 1980s, CNNs were the preferable solution for small problems only (e.g. LeNet [14] for handwritten digit recognition). Their practical applications were severely bottlenecked by the high computational cost when the number of classes and training data are large. However with the recent proliferation of modern GPUs, this bottleneck has been largely alleviated. Nonetheless, it was not until the introduction of ReLU neurons (instead of TanH), max-pooling (instead of average pooling) and dropout regularisation that DNNs maximised their effectiveness and regained their popularity [13]. 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. More recently, it was shown that even deeper networks with smaller filters [22] are preferable for photo image recognition. Despite these great strides, to the best of our knowledge, all 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 embedding the unique characteristics of sketches into the network design, our Sketch-a-Net advances sketch recognition to the over-human level.

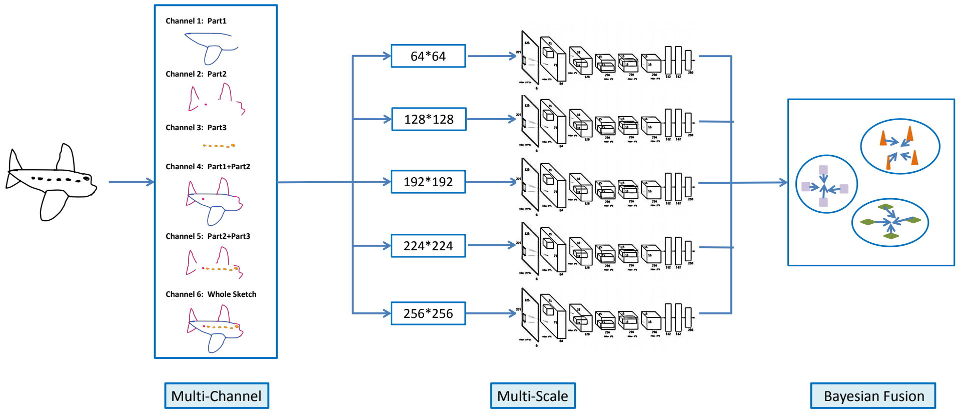

Figure 1: Illustration of our overall framework.

3 Methodology

In this section we introduce the three key technical components 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 our simple but novel generalisation that gives a DNN the ability to exploit the stroke ordering information that is unique to sketches (Sec. 3.2). We then introduce a multi-scale ensemble of networks to address the variability in the levels of abstraction with a joint Bayesian fusion method for exploiting the complementarity of different scales (Sec. 3.3). Fig. 1 illustrates our overall framework.

3.1 A CNN for Sketch Recognition

Our Sketch-a-Net is a deep CNN. Despite all the efforts so far, it remains an open question how to design the architecture of CNNs given a specific visual recognition task; but most recent recognition networks [1, 22] now follow a design pattern of multiple convolutional layers followed by fully connected layers, as popularised by the work of [13].

Our specific architecture is as follows: first we use five convolutional layers, each with rectifier (ReLU) [15] units, while the first, second and fifth layers are 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 [8] is applied on the first two fully connected layers. The final layer has 250 output units corresponding to 250 categories (that is 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 [13] and DeCAF [4].

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.

Commonalities between Sketch-a-Net and Photo-Oriented CNN Architectures

Filter Number: In both our Sketch-a-Net and recent photo-oriented CNNs [13, 22], 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 [1].

Unique Aspects in our Sketch-a-Net Architecture

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 [13], the current trend of research [27] is moving toward ever smaller filters: very recent [22] 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) [13] implements a form of lateral inhibition, which is found in real neurons. This is used pervasively in contemporary CNN recognition architectures [1, 13, 22]. 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 [22]. 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 pooling areas [13]. We found this brings $\sim 1\%$ improvement without much additional computation.

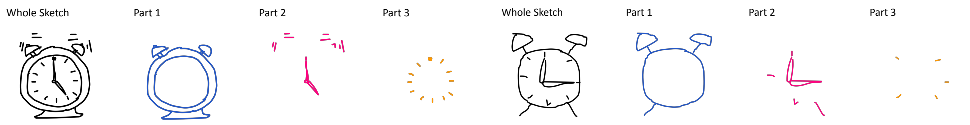

Figure 2: Illustration of stroke ordering in sketching with the Alarm Clock category.

Higher Dropout: Deeper neural networks generally improve performance but risk overfitting [22]. Recent CNN successes [1, 13, 22] deal with this using the (very large) ImageNet dataset [3] for training, and dropout [8] regularisation (randomly setting units activation to zero). Since a sketch dataset is typically much smaller than ImageNet, we compensate for this by setting a much higher dropout rate of 50%.

Lower Computational Cost: The total number of parameters in Sketch-a-Net is 8.5 million, which is relatively small for modern CNNs. For example, the classic AlexNet [13] has 60 million parameters (7 times larger), and recent state-of-the-art [22] reaches 144 million.

3.2 Modelling Sketch Stroke Order with Multiple Channels

Stroke Ordering: The order of drawn strokes 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. To provide intuition about this, Fig. 2 illustrates some sketches in the Alarm Clock category, with strokes broken down into three parts according to stroke order. Clearly there are different sketching strategies in terms of which semantic parts to draw first, but it is common to draw the main outline first, followed by details, as a recent study also found [6]. Modelling stroke ordering information is thus useful in distinguishing categories that are similar in their parts but differ in their typical ordering.

Modelling Stroke Order: We propose a simple but effective approach to modelling the sequential order of strokes by extending Sketch-a-Net to a multi-channel CNN: discretising strokes into three sequential groups (Fig. 2), and treating these parts as different channels in the first layer. Specifically, we use the three stroke parts to generate six images containing combinations of the stroke parts. As illustrated in Fig. 1, the first three images contain the three parts alone; the next two contain pairwise combinations of two parts, and the third is the original sketch of all parts. Our Sketch-a-Net described in Sec. 3.1 is then modified to take the six channel images as input (i.e. the first layer convolution filter size is changed to $15 \times 15 \times 6$). This multi-channel model has a couple of advantages: (i) the relative importance of early versus late strokes are learned automatically by back propagation training; (ii) it is a simple and efficient modification of the existing architecture: the number of parameters and hence training time is only increased by 1% compared to the single channel Sketch-a-Net.

3.3 A Multi-scale Network Ensemble with Bayesian Fusion

The next challenging aspect of sketch recognition to be addressed is the variability in sketching abstraction. To deal with this we introduce an ensemble of our multi-channel Sketch-a-Nets. For each network in the ensemble we learn a model of varying coarseness by blurring its training data to different degrees. Specifically, we create a 5 network ensemble by blurring – downsampling and then upsampling by to the original 256×256 pixel image size.

The downsample sizes are: $256, 224, 192, 128, 64$. Each network in the ensemble is independently trained by backdrop using one of these blur levels.

The multi-scale Sketch-a-Net ensemble can be used for classifying a test image using score-level fusion, i.e., averaging the softmax scores. However, this fusion strategy treats each network thus each scale equally without discrimination. Alternatively, one could concatenate the CNN learned representations in each network and feed them to a downstream classifier [4]. However, again no scale and feature selection is possible with this feature-level fusion strategy. In this work, we propose to take the ($512D$) activation of the penultimate layer of our network as a representation, and apply the recent Joint Bayesian (JB) fusion method [2] to exploit the complementarity between different scales.

The JB framework models *pairs* of instances (in this case CNN activations), by full covariance Gaussians, under the prior assumption that each instance x is a sum of its (Normally distributed) category mean and instance specific deviation: $x = \mu + \varepsilon$. In particular it learns two full covariance Gaussians, representing pairs from the same category and different categories respectively, i.e., it models $p(x_1, x_2|H_I)$ and $p(x_1, x_2|H_E)$ where x_1 and x_2 are instances, and H_I and H_E are the matched and mismatched pair hypotheses respectively. JB provides an EM algorithm for learning these covariances Σ_I and Σ_E respectively. Once learned, optimal Bayesian matching can be done using a likelihood ratio test:

$$r(x_1, x_2) = \log \frac{P(x_1, x_2 \mid H_I)}{P(x_1, x_2 \mid H_E)} \tag{1}$$

which turns out to be equivalent [2] to a metric learner capable of learning strong metrics with more degrees of freedom than traditional Mahalanobis metrics.

Although initially designed for *verification*, we re-purpose JB for classification here. Let each x represent the $5 \times 512 = 2560D$ concatenated feature vector from our network ensemble. Training: Using this activation vector as a new representation for the training data, we train the JB model, thus learning a good metric. Testing: Given the activation vectors of train and test data, we use the likelihood-ratio test (Eq. 1) to compare each test point to the full train set. With this mechanism to match test to train points, final classification is achieved with K-Nearest-Neighbour (KNN) matching[1][2]. 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 at different scales.

4 Experiments

Dataset: We evaluate our model on the TU-Berlin sketch dataset [6], 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, thus providing a diversity of both categories and sketching styles within each category. 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).

[1]We set $k = 5$ in this work and the regularisation parameter of JB is set to 0.1

[2]For robustness at test time, we also take 10 crops and reflections of each train and test image [13]. 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: Data augmentation is commonly with CNNs to reduce overfitting. We performed data augmentation by replicating the sketches with a number of transformations. Specifically, for each input sketch, we did horizontal reflection, rotation (in the range [-5, +5] degrees) and systematic combinations of horizontal and vertical shifts (up to 32 pixels). Thus, when using two thirds of the data for training, the total pool of training instances is $(20000 \cdot 0.67) \times (32 \cdot 32 \cdot 11 \cdot 2) = 300M$, increasing the size by a factor of 22,528.

Competitors: We compared our results with a variety of alternatives. These included the conventional **HOG-SVM** pipeline [6], structured **ensemble matching** [16], **multi-kernel SVM** [17], the current state-of-the-art Fisher Vector Spatial Pooling **(FV-SP)** [21], and DNN based models including **AlexNet** [13] and **LeNet** [15]. AlexNet is a large deep CNN designed for classifying ImageNet LSVRC-2010 [3] images. It has five convolutional layers and 3 fully connected layers. We used two versions of AlexNet: (i) **AlexNet-SVM**: following common practice [4], 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. Finally, 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.

HOG-SVM [6]	Ensemble [16]	MKL-SVM [17]	FV-SP [21]	
56%	61.5%	65.8%	68.9	
AlexNet-SVM [13]	AlexNet-Sketch [13]	LeNet [15]	Sketch-a-Net	Human [6]
67.1%	68.6%	55.2%	**74.9%**	73.1%

Table 2: Comparison with state of the art results on sketch recognition

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 purpose designed for sketch [6, 16, 21], as well as the state-of-the-art photo-oriented CNN model [13] repurposed for sketch; (ii) we show for the first time, an automated sketch recognition model can surpass human performance on sketch recognition (74.9% by our Sketch-a-Net vs. 73.1% for humans based on the study in [6]); (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 images. 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 shallow. This suggests that a deeper/more complex model is necessary to cope with the larger intra-class variations exhibited in sketches; (v) last but not least, upon close 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 human. For example, for 'seagull', 'flying-bird', 'standing-bird' and 'pigeon', all of which belong to the coarse semantic category of 'bird', Sketch-a-Net obtained an average accuracy of 42.5% while human only achieved 24.8%. 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 23.9% for 'seagull' which is nearly 10 times better.

Full Model (M-Cha+M-Sca)	M-Cha+S-Sca	S-Cha+S-Sca	AlexNet-Sketch
74.9%	72.6%	72.2%	68.6%

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

Contributions of Individual Components: Compared to conventional photo-oriented DNNs such as AlexNet, our Sketch-a-Net has three distinct features: (i) the specific network architecture (see Sec. 3.1), (ii) the multi-channel structure for modelling stroke ordering (see Sec. 3.2), and (iii) the multi-scale network ensemble to deal with variable levels of abstraction (see Sec. 3.3). In this experiment, we evaluate the contributions of each new feature. Specifically, we examined two stripped-down versions of our full model (multi-channel-multi-scale (M-Cha+M-Sca)): multi-channel-single-scale (M-Cha+S-Sca) Sketch-a-Net which uses only one network scale (the original scale of 256×256), and single-channel-single-scale (S-Cha+S-Sca) Sketch-a-Net which uses only sketches at the original scale. Results in Table 2 show that all three features contribute to the final strong performance of Sketch-a-Net. In particular, (i) the improvement of S-Cha+S-Sca over AlexNet-Sketch shows that our sketch-specific network architecture is effective; (ii) M-Cha+S-Sca achieved better performance than S-Cha+S-Sca, indicating the multi-channel features worked; (iii) the best result is achieved when all three new features are combined.

Joint Bayesian	Feature Fusion	Score Fusion
74.9%	72.8%	74.1%

Table 4: Comparison of different fusion strategies.

Comparison of Different Fusion Strategies: Given an ensemble of Sketch-a-Net at different scales, various fusion strategies can be adopted for the final classification task. Table 4 compares our joint Bayesian fusion method with the two most commonly adopted alternatives: feature level fusion and score level fusion. For feature level fusion, we treat each single scale network as a feature extractor, and concatenate the 512D outputs of their penultimate layers into a single feature. We then trained a linear SVM based on this $5 \times 512 = 2560$D feature vector. For score level fusion, we average the 250D softmax probabilities of each network in the ensemble to make a final prediction. For JB fusion, we take the same 2560D concatenated feature vector used by feature fusion, but perform KNN matching with JB similarity metric, rather than SVM classification. Interestingly, although score fusion is better than vanilla SVM feature fusion, JB makes much better use of the concatenated feature vector because the full covariance model better learns how to weight the outputs of the networks and exploit their complementarity.

Qualitative Results: Figure 3 shows some qualitative results. Some examples of surprisingly tough successes are shown in green. Mistakes made by the network (red) (intended category of the sketcher in black) are very reasonable. 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 Fig. 4, the filters in the first layer of Sketch-a-Net (Fig. 4(left)) learn something very similar to the biologically plausible Gabor filters (Fig. 4(right)) [7]. 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 [19, 24].

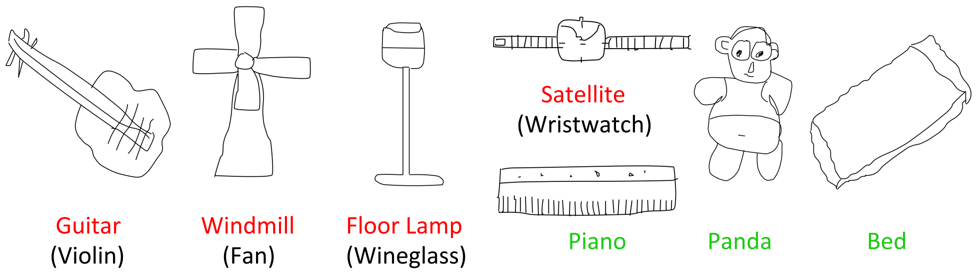

Figure 3: Qualitative illustration of recognition successes (green) and failures (red).

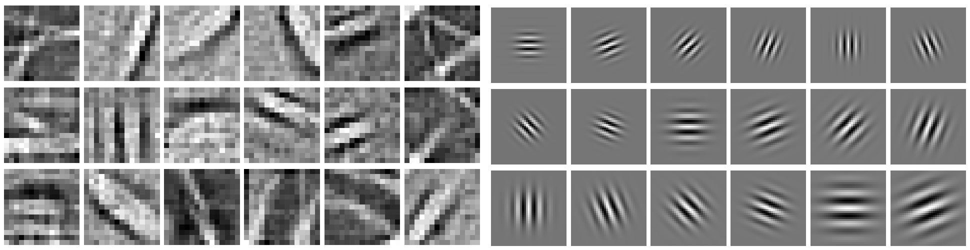

Figure 4: Visualisation of the learned filters. Left: randomly selected filters from the first layer in our model; right: the real parts of some Gabor filters.

Running cost: Our Sketch-a-Net model was implemented using Matlab based on the MatConvNet [[illegible]] toolbox. We trained our 5-network ensemble for 230 epochs each, with each instance undergoing random data augmentation during each iteration. This took roughly 80 hours in total on a 2.60GHz CPU (without explicit parallelisation), or 10 hours using a NVIDIA K40-GPU. Note that this means Sketch-a-Net was not trained for long enough to use the full pool of available data augmentations.

Reproducibility: For reproducibility and to support future research, our training and testing pipeline is made available at http://www.eecs.qmul.ac.uk/~tmh/.

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 1.8% on a large scale sketch benchmark dataset. Key to the superior performance of our method lies with the specifically designed network model 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.

Acknowledgements: This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 640891. We gratefully acknowledge the support of NVIDIA Corporation for the donation of the GPUs used for this research.

References

[1] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *BMVC*, 2014.

[2] D. Chen, X. Cao, L. Wang, F. Wen, and J. Sun. Bayesian face revisited: A joint formulation. In *ECCV*, 2012.

[3] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

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

[5] M. Eitz, K. Hildebrand, T. Boubekeur, and M. Alexa. Sketch-based image retrieval: Benchmark and bag-of-features descriptors. *TVCG*, 2011.

[6] M. Eitz, J. Hays, and M. Alexa. How do humans sketch objects? In *SIGGRAPH*, 2012.

[7] D. Gabor. Theory of communication. part 1: The analysis of information. *Journal of the Institution of Electrical Engineers-Part III: Radio and Communication Engineering*, 93 (26):429–441, 1946.

[8] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. In *arXiv preprint arXiv:1207.0580*, 2012.

[9] R. Hu and J. Collomosse. A performance evaluation of gradient field hog descriptor for sketch based image retrieval. *CVIU*, 2013.

[10] M. F. A. Jabal, M. S. M. Rahim, N. Z. S. Othman, and Z. Jupri. A comparative study on extraction and recognition method of cad data from cad drawings. In *International Conference on Information Management and Engineering (ICIME)*, 2009.

[11] G. Johnson, M. D. Gross, J. Hong, and E. Yi-Luen Do. Computational support for sketching in design: a review. *Foundations and Trends in Human-Computer Interaction*, 2009.

[12] B. F. Klare, Z. Li, and A. K. Jain. Matching forensic sketches to mug shot photos. *TPAMI*, 2011.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[14] Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. In *NIPS*, 1990.

[15] Y. LeCun, L. Bottou, G. B. Orr, and K. Müller. Efficient backprop. *Neural networks: Tricks of the trade*, pages 9–48, 2012.

[16] Y. Li, Y. Song, and S. Gong. Sketch recognition by ensemble matching of structured features. In *BMVC*, 2013.

[17] Y. Li, T. M. Hospedales, Y. Song, and S. Gong. Free-hand sketch recognition by multi-kernel feature learning. *CVIU*, 2015.

[18] T. Lu, C. Tai, F. Su, and S. Cai. A new recognition model for electronic architectural drawings. *Computer-Aided Design*, 2005.

[19] B. A. Olshausen and Field D. J. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 1996.

[20] S. Ouyang, T. Hospedales, Y. Song, and X. Li. Cross-modal face matching: beyond viewed sketches. In *ACCV*, 2014.

[21] R. G. Schneider and T. Tuytelaars. Sketch classification and classification-driven analysis using fisher vectors. In *SIGGRAPH Asia*, 2014.

[22] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[23] P. Sousa and M. J. Fonseca. Geometric matching for clip-art drawing retrieval. *Journal of Visual Communication and Image Representation*, 20(2):71–83, 2009.

[24] M. F. Stollenga, J. Masci, F. Gomez, and J. Schmidhuber. Deep networks with internal selective attention through feedback connections. In *NIPS*, 2014.

[25] F. Wang, L. Kang, and Y. Li. Sketch-based 3d shape retrieval using convolutional neural networks. In *arXiv preprint arXiv:1504.03504*, 2015.

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

[27] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014.

[28] C. L. Zitnick and D. Parikh. Bringing semantics into focus using visual abstraction. In *CVPR*, 2013.