Free-hand Sketch Recognition Classification

Wayne Lu
Stanford University
waynelu@stanford.edu

Elizabeth Tran
Stanford University
eliztran@stanford.edu

Abstract

*Automatic recognition of sketches differs from other areas of image classification because sketches of the same object can vary based on artistic style and drawing ability. In addition, sketches are less detailed and thus harder to distinguish than photographs. Using a publicly available dataset of 20,000 sketches across 250 classes from Eitz et al. [**?**], we are applying ConvNets (CNNs) in order to improve performance to increase the recognition accuracy on sketches drawn by different people. In our experiments, we analyze the effects of several hyperparameters on overall performance.*

1. Introduction

Sketching is one of the primary methods people use to communicate visual information. Since the era of primitive cave paintings, humans have used simple illustrations to represent real-world objects and concepts. Sketches are often abstract and stylized, varying based on artistic ability and style. In addition, sketches emphasize defining characteristics of real-world objects and ignore features which are either less important or more difficult to draw. For example, texture is almost never rendered unless it is important for recognition, such as the spikes on a hedgehog. In this way, sketches can be interpreted as a distillation of human visual recognition schemas.

Sketch recognition attempts to recognize the intent of the user while allowing the user to draw in an unconstrained manner. This allows for the user to not have to spend time being trained how to draw on the system, nor will the system need to be trained on how to recognize each users particular drawing style. Deciphering freehand sketches can be viewed under the lens of image category recognition, a well studied problem in the computer vision community.

However, the sketch recognition problem differs from traditional photographic image classification. First, sketches are less visually complex than photographs. Whereas color photographs have three color channels per pixel, sketches are encoded as either black-and-white or grayscale. Photographs contain visual information throughout the image whereas sketches consist primarily of blank space. Second, sketches and photographs have different sources of intraclass variation. Whereas photographic image classification faces obstacles such as camera angle, occlusion, and illumination, photographs are still grounded in reality. On the other hand, sketches differ based on artistic style, which is unique to every artist. While people can agree on what an object looks like, how they ultimately render the object can vary significantly.

2. Related Work

Since SketchPad [**?**], sketch recognition has introduced sketching as a means of human computer interaction. Computer vision has since tried different approaches to achieve better results in multiple application areas. Eitz et al. [**?**] was able to demonstrate classification rates can be achieved for computational sketch recognition by using local feature vectors, bag of features sketch representation and SVMs to classify sketches. Schneider et al. [**?**] then modified the benchmark proposed by Eitz et al [**?**] by making it more focused on how the image should like, rather than the original drawing intention, and they also used SIFT, GMM based on Fisher vector encoding, and SVMs to achieve sketch recognition.

Convolutional neural networks (CNN) have emerged as a powerful framework for feature representation and recognition [**?**]. Convolutional neural networks are a type of neurobiologically inspired feed-forward artificial neural network which consist of multiple layers of neurons, and the neurons in each layer are then collected into sets. At the input layer, where the data gets introduce to the CNN, these neuron sets map to small regions of input image. Deeper layers of the network can be composed of local or global pooling (fully-connected) layers which combine outputs of the neuron sets from previous layer. The pooling is typically achieved through convolution-like operations. Deep neural networks (DNN), especially CNNs, can automatically learn features instead of manually extracting features and its multi layers learning can get more effective expression. When it comes to CNN design, the trend in the past few

years has pointed in one direction: deeper networks (cite alex net). This move towards deeper networks has been beneficial for many applications. The most prominent application has been object classification, where the deeper the neural network, the better the performance. However, current existing CNNs are designed for photos, and they are trained on a large amount of data to avoid overfitting.

CNNs, despite of having better classification performance, are harder to train, and an effective way to solve this problem is suggested using a Residual Network(ResNets) [cite this]. In a traditional activation layer, when the signal is sent backwards, the gradient must pass through all their convolutional layers, which can cause trouble due to the nonlinearities which are involved. ResNet, on the other hand, provides the networks with shortcut at each layer to make sending the gradient signal backwards more smoothly without the intermediate layers being diminished This shortcut allows addressing the problem of vanishing gradients and faster training. An example of a ResNet building block can be seen in Figure 1.

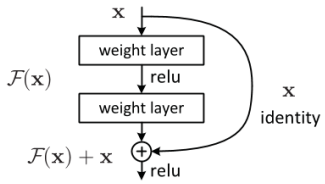

Figure 1. Model of a basic residual unit [cite]

Sketches, on the other hand, require special model architectures. In 2012, Eitz et al. [**?**] released the largest sketch object dataset. Since its release, a number of approaches have been proposed to recognize freehand sketches. In Yu et al. [**?**], they proposed Sketch-a-Net, a different type of CNN that is customizable towards sketches. While Sarvadevabhatla et al. [**?**] used two popular ConvNets (ImageNet and a modified LeNet) to fine-tuned their parameters on the TU-Berlin sketch dataset in order to extract deep features from CNNs to recognize hand-drawn sketches.

In this paper, we explore the use of deep convolutional neural networks (DCNN) architectures for sketch recognition. To the best of our knowledge, recurrent neural networks have not been utilized for online sketch recognition.

3. Dataset

Eitz et al. [**?**] defined a taxonomy of 250 object categories gathered from 20,000 human sketches (TU-Berlin sketch benchmark), and it is currently the largest curated dataset of hand drawn sketches. The dataset has a total of 250 classes, and it is split into three categories: exhaustive, recognizable, and specific. Images within this dataset are available as 1111 x 1111 px PNG files, which we have resized to 128 x 128 px grayscale images. An example sketch from this dataset can been see in Figure 2. Even though the database covers a range of object categories, its major shortcoming comes from ambiguous drawn sketches [**?**]. In order to address this problem, Rosalia et al. [**?**] were able to find a subset of 160 unambiguous object categories to make a more reliable benchmark. However, we are choosing to use all 250 object catergories in order to compare our accuracies with other models.

Figure 2. Example sketch from the TU Berlin dataset

4. Approach

The ResNet model introduced in [cite] is our starting point for the network design. ResNet is specifically designed for images in ImageNet and accepts images with size 256 256 and classifies them in 1000 categories. However, since ResNet is used for images and we are trying to classify sketches, we did some customization to the original network. We chose to use this ResNet architecure in order to improve performance over traditional multiclass support vector classification. For our framework, we used Tensorflow to implement and train different inspiried ResNet Models.

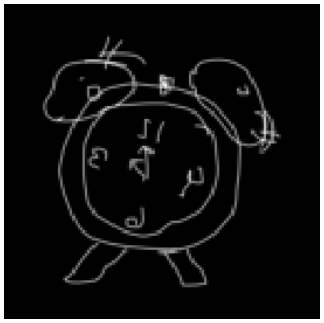

Figure 3. Example sketch of inverted with inverted color value that was fed into our model

The original images from the TU-Berlin dataset are first rescaled from 1111x1111 px to 128x128 px via bilinear in-

terpolation. Though there are 250 classes in the TU-Berlin dataset, there are only 80 sketches per class. To overcome overfitting, we need an augmentation method to increase the size of the dataset. To augment the training data, we generated additional training sketch examples by horizontally flipping existing training examples as well as inverting all examples to better match the effects of our input dropout. This was a similar method in [[**?**]], where they proposed image distortion to improve generalization. The resized images are then read into memory as grayscale 128x128 arrays.

Layer	Output Size	Feature Depth
Input	128x128x1	
Dropout	128x128x1	
7x7 conv, 64, /2	64x64x64	
3x3 residual unit, 64	64x64x64	
3x3 residual unit, 64	64x64x64	
3x3 residual unit, 64	64x64x64	
Dropout	64x64x64	
3x3 residual unit, 128, /2	32x32x128	
3x3 residual unit, 128	32x32x128	
3x3 residual unit, 128	32x32x128	
Dropout	32x32x128	
3x3 residual unit, 256, /2	16x16x256	
3x3 residual unit, 256	16x16x256	
3x3 residual unit 256	16x16x256	
Dropout	16x16x256	
3x3 residual unit, 512, /2	8x8x512	
3x3 residual unit, 512	8x8x512	
3x3 residual unit, 512	8x8x512	
8x8 Average Pooling	512	
Dropout	512	
FC-250	250	

Figure 4. ConvNet architecture. The "-f" notation refers to a layer with a fxf filter. The "/s" notation refers to a layer with stride s. Each residual unit consists of

Our approach follows a fairly standard image classification pipeline. We folded our data into three folds: xxx training, xxx validation, xxx testing. Image arrays are fed into a CNN with the final output being fed to a 250-unit fully connected layer to generate logits for each class. We then apply a softmax loss function to the logits and optimize the loss using the Adam algorithm. Our framework of our best proposed method is shown in Figure 4.

5. Experiment Results

We tried to train multiple deep ResNets varying from x to x layers to see which one performs better on the TU-Berlin dataset. With varying amount of layers to make the model deeper, we also tried several approaches of dropout to counter overfitting as well as different sizes of width for our filters. A more aggressive dropout of 60 and a lesser aggressive dropout of 25 does not improve our validation accuracy. We also tried a variation of widith for our layers. Using these different parameters, our best model was our Vanilla ResNet. This best performing model had a total of xx layers. With this model, we were able to achieve a test accuracy of 65.15, but the other types of models fall within this range of accurarcy. The results are brought in Table 1. Note that all the models are trained for the same number of epochs. (See Table 1 and Figure 6).

(can you add in the percent sign?)

Method	%
ResNet-aggro-dropout	65.3
ResNet Vanilla	65.15
ResNet Weak Dropout	65.6
ResNet Wide	65.3
ResNet Wide V2	63.9
ResNet Wider	58.8
ResNet Widish	62.05
ResNet Widest	55.8

Table 1. Validation performance comparison between our different parameters

In Figure 5, we compared all of our model's training and validation accuracies. Looking at the the training accuaries, we see that there is an increasment over the amount of epochs. The ResNet-Vanilina archertecure increases most rapidly compared to the others models, but the ResNet-Weak-dropout model reaches the same training accurercies as the Vanilla model. The ResNet-Vanilina and ResNet-Weak-dropout archertecure have the highest training accuaracy compared to the other models. On the other hand, all our validation accuracies level out near the end of training. There is a drastic drop in validation accuaries for Widish-ResNet model.

The difference between training accuracy and validation accuracy is a good indicator of over fitting and based on our results we realized that ResNets are more prone to overfitting. These results show that adding a simple shortcut connection can improve the accuracy in the classification task and make the training process much faster, but the trade of is that residual networks are more prone to over-fitting which is undesirable. Our results also suggest that depth is the largest contributing factor to classification accuracy, with dropout regularization providing minor improvement. Widening layers does not result in noticeable changes, and the need to reduce memory consumption by reducing network depth in fact leads to lower accuracy.

Table 2 summarizes the overall performance in terms of average recognition accuracy for various architectures compared to our highest performing model. While our CNN

Method	%
SIFT-varient + BoF + SVM [?]	56
IDM + SVM [?]	71.30
ConvNet [?]	75.42
ConvNet [?]	77.69
ConvNet – Ours	65.15

Table 2. Test accuracy comparison of our model versus other methods.

model?s performance is not quite as strong as the CNN model of xxx [deepsketch peeps – i think i messed up with the bib somehow that it isn't showing on mine :( ], in terms of test accuaries, we would like to point out that the authors used a larger CNN on a larger external dataset.

Our experiments illustrate the complexity of free-hand sketch recognition. hough our model was able to outperform the traditional multi-class support vector classifiers (SVMs) [1] , there is still more improvement that can be done. Along with that, we were not able to train the model as long as we would have hoped. If you notice in Figure 6 the validation accruracry for our models has not completely leveled out.

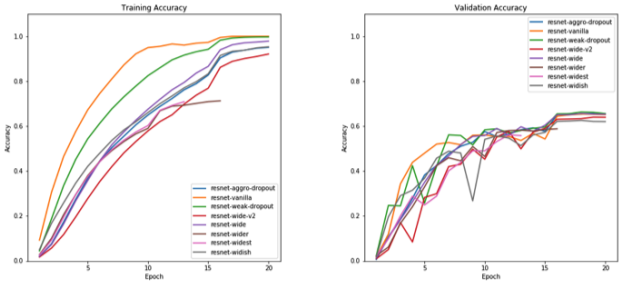

Figure 5. Training and Validation comparision between all our models – wayne can you make this to be the entire top page? i don't know how to use latex well and it's way too small ;A;

6. Conclusion

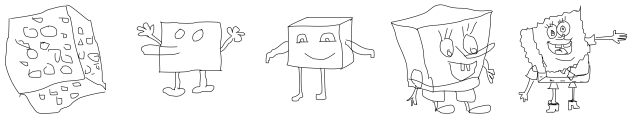

Figure 6. Examples of interclass variations of Spongebob from the TU Berlin dataset highlighting the varability

In this paper, we have presented our CNN architecture for freehand sketch recognition. We see that in our model, where free-hand sketches are inherently hard to classify due to large amounts of intraclass variation and interclass overlap, along with a lack of complex visual information, in contrast to traditional image recognition which instead deals with an overabundance of visual information. Sketches are usually centered around the edges, and for most neural networks edge detection is within the first few layers.

Our experiments showed that moderate dropout with deep networks provides the best results. Overaggressive dropout and shallow networks both lead to lower accuracy. Notably, compensating for shallowness with increased width does not result in improved accuracy. ResNets are more powerful for very deep networks and and can hurt the performance for shallow networks. ResNets, however, does show promising landscape in deep learning.

Some things that we can try next is use other networks to train on the TU- Berlin dataset, such as VGG or the original arertecture of ResNet. The TU-Berlin dataset, though is the largest, is still relatively limited.

Our framework source-code and associated data (pre-trained models) can be accessed at: github.com/krinkels/sketch-recognition