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 in order to improve performance over traditional multi-class support vector classifiers (SVMs) using by experimenting with different architecture methods, such as quadrant pooling and fine tuning, and hyperparameters to asses the overall performance of our proposed model.

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.

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 intra-class 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.

In this paper, we explore the use of deep convolutional neural networks (DCNN) architectures for sketch recognition.

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 [?]. However, current existing CNNs are designed for photos, and they are trained on a large amount of data to avoid overfitting. Sketches, on the other hand, require special model architectures. 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.

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 1. Even though the database covers a range of object categories, its major shortcoming comes from ambiguous drawn sketches [?]. In order to address this problem, Ro?alia et al. [?] were able to find a subset of 160 unambiguous object categories to make a more reliable benchmark. Due to the ambigiuity of some images, we are planning to use this curated set of 160 categories.

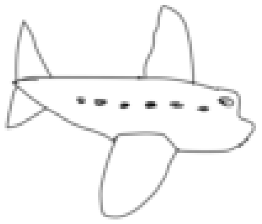

Figure 1. Example sketch from TU Berlin dataset.

4. Approach

We first preprocess the dataset to produce 128 x 128 px images. The original images are rescaled from 1111 x 1111px to 128 x 128px via bilinear interpolation. The resized images are then read into memory as grayscale 128x128 arrays.

Our approach follows a fairly standard image classification pipeline. 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.

At the moment, we are still experimenting with ConvNet architectures. We have been exploring novel architectures from scratch using convolutional, batch normalization, and pooling layers, but as of this writing, nothing has been very successful. Due to time constraints, we will change our approach after this milestone to using existing architectures such as AlexNet or ResNet as starting points to explore from. We will likely also employ the method of image distortion used by Seddati et al. [?] to improve generalization.

5. Experiments

The first model that we train is a single sketch CNN. We would pass an individual sketch through the CNN. The CNN has 3 convolutional layers consisting of 128 x 128, 64 x 64, and 32 x 32 filters respectively. Each convolutional layer is followed by a batch normalization, ReLU (Rectified Linear Unit) activation function, and a 2 x 2 max pooling. After the convolutional layer is a fully-connected layer, we use cross entropy as a cost function. With this model, we were able to achieve a test accuracy of 0.004.

Method	mAP
SIFT-varient + BoF + SVM [?]	56
IDM + SVM [?]	71.30
ConvNet [?]	75.42
ConvNet [?]	77.69
ConvNet –Ours	0.004

Table 1. Performance comparison of our models versus other methods.

In order to improve our network, we are going to try different parameters such as CNN with different levels of regularization, CNN with dropout, CNN with data augmentation, and CNN with dropout and data augmentation.