Deep Sketch Network for Million-scale Sketch Recognition

Kegang Xu *Dan Xia* *Hongwei Zhang*

tosmast@stanford.edu *danxia@stanford.edu* *hongweiz@stanford.edu*

Abstract

Unlike photos, sketches are highly abstract and iconic. The style of drawing varies differently from country to country. With the availability of a million-scale sketch dataset from Google, our project explored the effectiveness of multiple architectures to predict the categories of the sketches. Our project built an end to end Deep Sketch Network by combining CNN and RNN together to leverage both the pixel based image dataset and the sequencing information from stroke dataset. We used noisy removal technique on data preprocess. We proved bigger batchsize could alleviate the impact of the noisy data in this project. Also we adopted center loss to differentiate the inter-class feature to improve the accuracy.

1. Introduction

QuickDraw is Google Open Challenge for sketch drawing recognition. It is an experimental game in a playful way to show how AI works. The dataset includes 50M+ sketch drawings and 340 categories. Each category includes 15K ~ 20K drawings. Advancing on the sketch recognition could greatly help OCR, ASR and NLP.

There are the following challenges in our project.

- Million-scale sketches. It includes 50M free hand drawing in total.
- The sketches are highly abstract and iconic different from static visual image.
- Different countries have their own traditional style due to different cultures.
- Noisy data in the train set. The drawing could be incomplete and misleading

In our project, we feed pixel image to CNN and stroke sequence to RNN. Pixel images is transferred on the fly from stroke dataset. The output is the category of the image.

2. Related Work

Sketches recognition is an extremely challenging task. They are highly abstract and iconic, different from normal image and photos. Sketches are usually drawn by hand and they vary in size, style and strokes. Both make even people hard to recognize the category of sketch sometimes. Also sketching is drawn in sequence, so it is a dynamic process rather than a static image which only consists of the colored pixel. D. Ha[1] and Peng Xu (2018)[2] designed SketchMate, a deep hashing framework for sketch retrieval to work on a multi-million scale human sketch dataset. The key of the design is a two-branch CNN-RNN architecture which is adapted to explore the temporal ordering of strokes with the novel sketch center loss. A ground-breaking work was done by Yu et al.[4], which beats human performance on sketch for the first time. It designed a multi-channel architecture to model the sequential ordering of strokes in each sketch and ensembled a multi-scale network to address the variability in abstraction and sparsity. A recent work also proposed to combine static sketch visual characteristics and dynamic temporal information in a single deep model [5]. In this project, we tried to design a new network combined with CNN and RNN [7] which inputs the pixel image and temporal sequence information [4][5] and center loss in [2].

3. Dataset

In Quick-drawing website [6], there are two different kinds of dataset, pixel based sketch and stroke vector data. In this project, we are going to use both data, the pixel image as the input to CNN and the stroke vector data as the input to RNN.

3.1 Examples of Noisy Data

Sketch data collected with crowdsourcing are inevitably noisy, such as incomplete drawing or completely irrelevant drawing. Here are some examples in the mug category:

Figure1. Incomplete

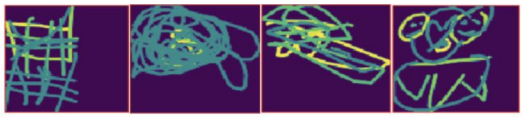

Figure 2. Irrelevant drawings

3.2 Different style of drawing per country

Different countries likely have different culture and are in different stage of modernization. Therefore, it may lead to different impression and different styles of drawing. Below are some examples from the category of cell phone.

Figure 3. Great British smart phone style

Figure 4. South Africa cordless phone style

3.3 Data Preprocessing

The website already provides the data with the size of 28*28 and the colors of 256. But the resolution of 28*28 is very low for the human. Intuitively we need to improve the resolution while we have to consider the GPU memory restrictions. Finally, we generated 96*96 image with one channel of 256 colors from stroke data on the fly.

To reduce the training time, we chose 40 categories from the dataset based on the difficult level of accuracy with pre-trained model. According to [8], we choose the whole dataset of these class to train our DSN.

3.4 Data Augmentation

Data augmentation is a common skill in image recognition to reduce overfitting. We performed the following data augmentation:

- Remove the noisy data by calculating the entropy histogram of each class. Keep the images in [0.05,0.95] percentile in Figure 5 and it reduces the highly abstract and overly drawing pictures
- Create one channel image with each stroke of different colors as input to CNN.
- Horizontal flip data is used as data enhancement since all categories could have different orientations.

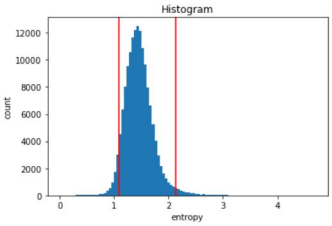

Figure 5. Entropy histogram of Mug

4. Network Architecture

4.1 Deep Sketch Network

Our Deep Sketch Network (DSN) is a hybrid network of RNN and CNN. RNN is good at presenting the sequential dataset while CNN could perform well in image recognition. DSN is shown in Figure 6. The RNN built on stroke

data and CNN network built on pixel data would be concatenated together and be fed into attention layer and then followed by fully connected layer FC. Then we compute the cross-entropy loss and center loss entropy in the end.

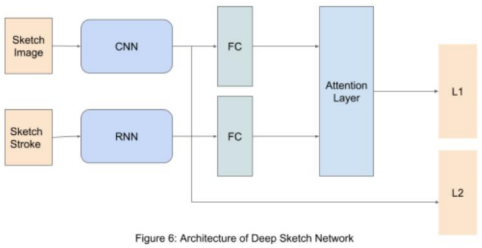

Figure 6: Architecture of Deep Sketch Network

4.2 Center Class Feature Vector

The dataset includes some noisy data, e.g, incomplete strokes and drawings. In[2], it defines the loss function of inter-class to differentiate the categories and remove some noisy data. At the last layer of CNN and RNN, a class feature vector could be obtained. Every image is compared to the center of the class feature vector. We will keep 90% images of each category's entropy between 0.05 and 0.95. After removing the noisy data, we could calculate the mean of the class feature vector after the attention layer.

4.3 Loss and Metric

DSN Loss: $\text{Loss} = \ell_1 + \alpha \cdot \ell_2$

- ℓ_1 is the loss of cross-entropy for categories
- ℓ_2 is the Euclidian distance between Fn and Cn. Cn represents the mean feature vector of one class. Fn is the feature vector of each sample. Feature vector is the combination of the output of RNN and CNN right before FC layer.
- $\ell_2 = \frac{1}{\mathrm{N}} \sum_{\mathrm{n}=1}^{\mathrm{N}} ||Fn - Cn||^2$

According to the challenge, mean average precision@3(MAP3) is used for metric measure in the project.

- $\text{MAP@3} = \frac{1}{\mathrm{U}} \sum_{n=1}^{U} \sum_{k=1}^{Min(k,3)} P(k)$

4.4 Attention Layer

Instead of averaging or max pooling, attention mechanism allows us to learn the weights for each vector before summing. Human visual attention focuses on a certain area with high resolution and perceive the other area with low resolution.

5.1 Experiments

In our experiments, we tuned some key hyperparameters for the DSN. We evaluated the performances of 5 models and analyzed their results.

Learning rate tuning

In order to get the best learning rate parameter for DSN, we tried several different learning rate values: $3*10^{-3}$, $3*10^{-4}$ and $3*10^{-5}$. We recorded the accuracy and loss change for every 75 iterations, and drew the trend in the right diagram.

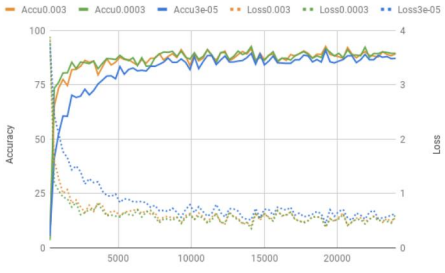

Figure 7. Accuracy &Loss Vs Iteration

From the diagram, we can see the training was going well with all three learning rate values since loss decreases and accuracy increases gradually. Finally, all three learning rates achieved comparable accuracy and loss. The different learning rates impact the training significantly at early stage: bigger learning rate will cause loss to decrease faster and accuracy to increase faster. Since

$3*10^{-4}$ produces the best result, we pick $3*10^{-4}$ as the learning rate for the final results. We also checked the training loss and validation loss. Training loss is slightly lower than validation loss. So there is no overfitting at training time.

Besides that, we also trained α value with 0.1,0.01 and 0.001. The accuracy for different α is quite similar, 0.01 is the best one.

Batch size tuning

Batch size is another hyperparameter we tuned for DSN. As the above section mentioned, learning rate $3*10^{-4}$ is a good choice and is used in this analysis.

We tried different batch sizes: 64, 128, 256, 350. We did not train bigger size because our final train work is done under GPU memory limitation. Batch sizes bigger than 350 will cause out of memory issue.

From the diagram, we can see bigger batch size will improve accuracy. Batch size 350 is more stable and has better accuracy than others. Batch size 128 is not stable and has a dip due to the noise data. We proved batchsize could alleviate the impacts of the noisy in our project like described in [9].

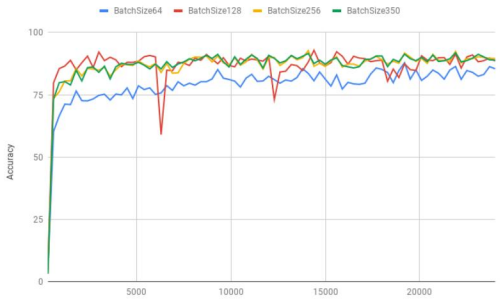

Figure 8. Batch Size Vs Accuracy

5.2 Results and Discussions

Model	MAP@3
Baseline(CNN)	86.12
RNN	86.08
CNN(Resnet50)	87.37
DSN	91.47
DSN+	91.76

Table 1. Models Vs Accuracy

There are five models in our projects. We designed a basic and simple CNN model as our baseline. The accuracy of the baseline is 86.12%. Then we modified resnet50 [3] to fit in one channel and image size input 96*96 to reduce the network scale. We got 87.37% accuracy on resnet50. We also designed a RNN with two layers of 1D CNN followed by 256 hidden state LSTM. We used 0.1 dropout with learning rate 5e-4 and batch size 1024 at training time. Eventually we got 86.08% on this model. RNN is harder to train than CNN, e.g. more sensitive to the learning rate and hard to get high accuracy in this sketch. Then we developed our DSN and introduced DSN+ with center loss. As shown in the table 1, DSN improves the performance to 91.47% and DSN+ gets the best accuracy 91.76%. The center loss intends to use center class feature to characterize the intra-class variants. The center loss does help with accuracy but it does not improve so much as 1% accuracy like the original author [2] got. The reason could be that we extracted feature vectors at different layer and from the different dataset than that paper.

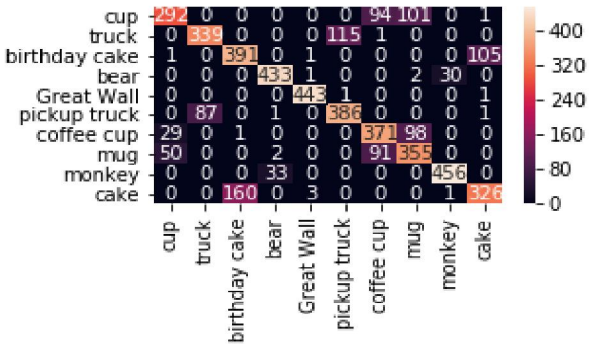

Figure 9. Confusion Matrix

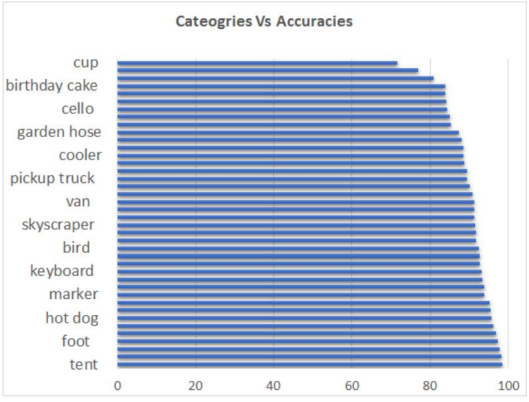

Figure 10. Category Vs Accuracy

Figure 10 shows the category Vs Accuracy. We trained 40 categories and the difficult level of recognition for each class vary greatly. Cup category has the lowest accuracy of 71.6% and the tent gets the highest accuracy 98.6%. On one hand, DSN did not do well in some categories. In Figure 9 Confusion Matrix, we could know there are three confusion groups:

- Cup, coffee cup and mug.
- Birthday Cake and Cake
- Bear and Monkey

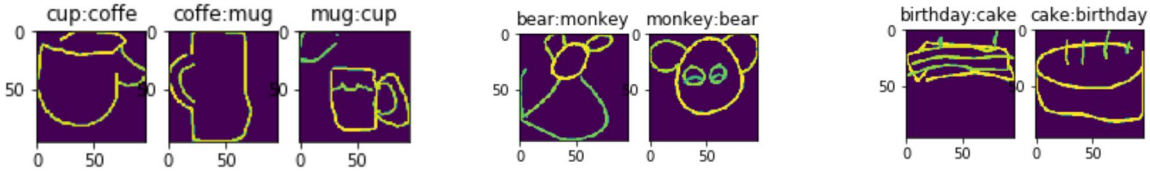

Figure 11. Confusion Group (X:Y means X is misabled as Y)

In a total of 512 cup samples, 94 is mislabeled as coffee cup and 101 as mug. In order to analyze them, we drew the samples in Figure 11. It is hard even for human to differentiate among cup, coffee cup, and mug. Bear: monkey means bear is mislabed as monkey. It is too abstract to tell the class based on the head. On the other hand, there are some noisy data which makes DSN fail to recognize correctly. For example, Cake: birthday Cake appear to be a noisy sample. It seems DSN correctly classified this sample but the label of data sample is not accurate.

Image size

Due to GPU computational and memory resource limit, our project tried to keep a small network and explored 3 kinds of small image size, 28*28, 64*64 and 96*96. As shown in Figure 12, we saw significant accuracy improvement 11.59% when we change image size from 28*28 while it is 2.4% increase from 64*64 and 96*96. We could expect a little more accuracy if we choose 128*128 and 256*256.

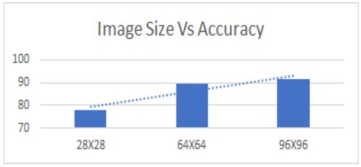

Image size	MAP3
28X28	77.80
64X64	89.39
96X96	91.76

Figure 12. Image size Vs Accuracy

6. Conclusion

We designed an end to end network Deep Sketch Network. We achieved 91.74% accuracy with the help of the center loss. We tried to incorporate the sequence information with the different colors. The sketch dataset has quite a lot of noisy data. We removed the incomplete and overly complicate drawing by computing entropy histogram. Due to the limit resource of GPU, we could only explore batch size 256 and image resolution 96*96 in the end. But our experiments show bigger batch size could alleviate the impact of the noisy data and train the network more robust and efficiently. Higher image resolution requires huge GPU computation and memory. We could expect it would increase the accuracy slightly. When we completed the DSN, Kaggle competition has been closed. So we could not run our DSN in Kaggle dataset.

If we have more time and more resources, we would like to explore the following areas in the future.

- Explore more CNN like seresnet. Use 3 channels by encoding more information like timestamp from raw strokes dataset.
- Explore the better RNN, like attention layer and bidirectional GRU.
- Try to train high resolution image with bigger batch size if possible.

7.Contribution

All three authors equally contribute the project. Kegang Xu mainly worked on the architecture of the network, and CNN baseline model training. Hongwei Zhang focused on the RNN network design and trained the network. Dan Xia worked on Loss center implementation, visualization, and data process. We all contributed to the tuning, debugging, and design for the end to end network.

Our code is on Github at https://github.com/tosmaster/imagevision/

Reference:

[1] D. Ha and D. Eck. A neural representation of sketch drawings. arXiv preprint arXiv:1704.03477, 2017. 2, 3, 5, 6

[2] Peng Xu,etc. SketchMate: Deep Hashing for Million-Scale Human Sketch Retrieval. arXiv:1804.0140.

[3] Kaiming He & Xiangyu Zhang et al. Deep Residual Learning for Image Recognition. arXiv 1512.03385

[4] Q. Yu & F. Liu et al. Sketch-a-Net that Beats Humans.2015, arxiv: 1501.07873

[5] Q. Yu & F. Liu et al. Sketch-a-net: A deep neural network that beats humans. IJCV 2017

[6] https://www.kaggle.com/c/quickdraw-doodle-recognition/data

[7] Jiang Wang & Yi Yang et al. CNN-RNN: A Unified Framework for Multi-label Image Classification. arXiv:1803.09218

[8] David Rolnick, Andreas Veit, Serge Belongie, Nir Shavit, Deep Learning is Robust to Massive Label Noise,2018.02, arXiv 1705.10694

[9] Priya Goyal, etc, Accurate, Large Minibatch SGD:Training ImageNet in 1 Hour,2018.4, arXiv 1706.02677