Implement Transfer Learning for Kaggle's Doodle Recognition Challenge

Meng Zhang [*1] **Yizhen Wang** [*1]

Abstract

In this project, we apply the method of Transfer Learning to the field of hand-drawn sketches to solve the problem of doodle images recognition, which is one of Kaggle's competitions. We transfer the Inception V3 model, which is a popular pre-trained CNN framework for real high-quality images, to fit the doodle classification problem with a total of 7 million doodle images as the training data using TensorFlow on a powerful machine with 4 Tesla K80 Graphics Processing Units (GPUs).This method can achieve a 50% Top 3 Mean Average Precision (MAP@3) in no more than 1000 epochs of training within one hour, showing a higher learning slope of MAP@3 improvement. We finally achieve 60% validation accuracy as well as 80% MAP@3, which is better than traditional CNN methods without transfer learning.

1. Introduction

This project derives from a Kaggle Competition called "Quick Draw! Doodle Recognition Challenge"[1]. The goal of this challenge is to build a classifier to predict the categories of hand-drawn doodle images. Hand-drawn images are common in daily life, yet needs more attention of optimization of the models to classify them. Our motivation is to build a better classifier model based on neural network, to have faster training speed, as well as better performance on the accuracy of doodle recognition. Hopefully, this model will be helpful to provide novel methods about improving hand-drawn sketches recognition technique.

Image recognition is a popular topic in machine learning. However, existing solutions are inadequate to solve problems about hand-drawn images. Most existing methods, as we will discuss in the chapter "Related Work", focus on high-quality images, which are mostly photos with high resolution. And most pre-trained frameworks have been built based on super-large datasets of high-quality images. As a result, they may not fit well with doodle images.

Our solution to this challenge is to build our model based on Convolution Neural Network with the pre-trained Inception V3 framework. We will implement transfer learning to adapt the pre-trained Inception model to fit with the large-scale doodle image dataset, thus speed up training and improve the performance of our model. We will also use distributed computation with powerful GPU virtual machines so more data can be fed into training in a shorter time.

2. Related Work

Hand-drawn Sketch Recognition

Several previous works have researched hand-drawn sketch recognition.

(Kara & Stahovich, 2005) described a trainable, hand-drawn symbol recognizer based on a multi-layer recognition scheme. They developed a fast technique that uses a polar coordinate representation to achieve rotational invariance to deal with the problems like symbols being sensitive to missing part, rotation, and extra pen strokes. The limitation of this paper is that it focused on a very narrow category of hand-drawn images with simple geometric structures and high intra-class similarity.

(Zhang et al., 2010) described a PCA-based algorithm for face sketch recognition used for forensic applications and compared its performance with the human. They found the algorithm was superior with the sketches of less distinctive features. This paper, like (Kara & Stahovich, 2005), just focused on a very specific field of hand-drawn sketches, which cannot be applied in more general recognition problems.

(Sun et al., 2012) developed a Query-adaptive Shape Topic (QST) model to recognize an arbitrary but semantically meaningful sketch. The model can mine object topics and shape topics related to the sketch, in which, multiple layers of information are modeled in a generative process. This paper did not use machine learning, but rather traditional statistical modeling methods to build the model, and achieved

[*]Equal contribution [1]School of Information, University of Texas at Austin, Austin, United States. Correspondence to: Meng Zhang <mengzhang@utexas.edu>, Yizhen Wang <yizhenwang0811@gmail.com>.

[1]Link:https://www.kaggle.com/c/quickdraw-doodle-recognition

only about 43% of top1 recognition accuracy, which is not satisfying for image recognition.

Convolution Neural Network

Several related works studied Convolution Neural Network.

(Sahiner et al., 1996) used a Convolution Neural Network (CNN) to investigate the classification of regions of interest (ROI's) on mammograms as either mass or normal tissue. The research achieved the area under the test ROC curve of 0.87, which corresponded to a true-positive rate of 90% and a false-positive rate of 31%. While this study demonstrated the feasibility of applying CNN in image classification, it is applied in the medical field with specific medical images, which have a huge difference with hand-drawn doodle images.

(Lawrence et al., 1997) presented a hybrid neural-network which combines CNN and other networks for human face recognition which compares favorably with other methods. The defection of this work is it used a small dataset with only 400 face images of 40 different persons. For large doodle datasets, the hybrid model may not fit well.

(Krizhevsky et al., 2012) trained a large, deep Convolution Neural Network to classify the 1.3 million high-resolution images in the LSVRC-2010 ImageNet with 1000 different classes. They achieved top-1 and top-5 error rates of 39.7% and 18.9% on the test dataset. This paper focused on high-resolution images. As a result, while its method is valuable for us to refer, it may not fit well with low-quality hand-drawn images.

TensorFlow

(Abadi et al., 2016) introduced the TensorFlow architecture and its programming model. It could be used in image classification and language modeling. Its flexible data flow representation enables power users to achieve excellent performance, but they have not yet determined default policies that work well for all users. They face the intriguing problem of providing a system that transparently and efficiently uses distributed resources, even when the structure of the computation unfolds dynamically.

(Vishnu et al., 2016) proposed a design to alleviate the distributed memory limitations of TensorFlow. They have considered several programming models, used Message Passing Interface (MPI) as the communication interface for parallelizing TensorFlow on distributed memory subsystems and specified the changes which were required to realize the implementation on distributed memory systems. The conclusion of it is that these changes are minimal and require no changes to the TensorFlow running time.

(Szegedy et al., 2017) presented three new network architectures in detail which includes Inception-ResNet-v1, Inception-ResNet-v2 and Inception-v4. They studied the combination of the two most recent ideas: Residual connections and the latest revised version of the Inception architecture and how the introduction of residual connections leads to dramatically improved training speed for the Inception architecture. Also, their latest models (with and without residual connections) outperform all our previous networks, just by virtue of the increased model size.

3. Methods

Transfer Learning

Transfer learning is a research method in machine learning that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem. In short, it is about using the models which have been trained on your own data.

As for our work by using transfer learning, we do not train the neural network directly from scratch. Instead, we train the last fully-connected Soft Max classifier of the network. There are two reasons for doing so. One is that our classification problem is not the same as the original classification problem. The pre-trained network is built for high-resolution real images classification, while our dataset is about low-resolution doodle images. Another reason is that the Convolution Neural Network could be separated into the Convolution part and the classification part. As for the Convolution part, it is to extract features from the images, for which the pre-trained model is very efficient and useful. After the extraction part, we use our own fully-connected classification layer to classify our dataset.

However, there are some risks that transfer learning will perform unsatisfying when the two problems have huge differences. For example, if you transfer a pre-trained model using the dataset about the natural scenery of image classification to learn face recognition, the performance may not be so good, because feature extractions of the human face and natural scenery are highly different, and the corresponding parameters after training are also different.

Inception V3

There are lots of pre-trained models, and in our project, we chose inception V3 pre-trained on ImageNet, which is the largest database of image recognition in the world and contains more than 20000 categories.

In general, the most direct way to improve network performance is to increase the depth and width of the network. Here, depth refers to the number of layers and the width refers to the number of neurons.

However, this method has the following problems:

- There are too many parameters for image classification problems, and if the training data set is limited, it is easy to produce an over-fitting result.
- The larger the network and the more parameters the network has, the greater the computational complexity will be, thus making it difficult for application;
- As the network gets deeper, it prone to have the gradient vanishing problem, where the gradient tends to disappear at the later epochs of training.

To address these problems, the GoogLeNet team proposed the Inception network structure, which is to construct a "basic neuron" structure to build a sparse, high computing performance network structure. By designing a sparse network structure, but able to generate dense data, it can not only increase the performance of the neural network but also ensure the efficiency of the use of computing resources.

In our work, we will use the pre-trained Inception V3 model as the Convolution part. Inception V3, compared to its original version, improves the factorization of the Convolutions, which brings better computation speed and better adaption to non-linear problems.

Our system firstly transfers pixels of the images extracted from the raw dataset on Kaggle into actual image files. Then we feed the images into the inception V3 model to train the Soft Max layer and get the bottleneck feature vectors of the images so that the training can be completed. Finally, we use a full-connection layer with our 340 doodle classes to predict the top3 most possible classes of each test doodle images. The flowchart of our system is shown as Figure 1.

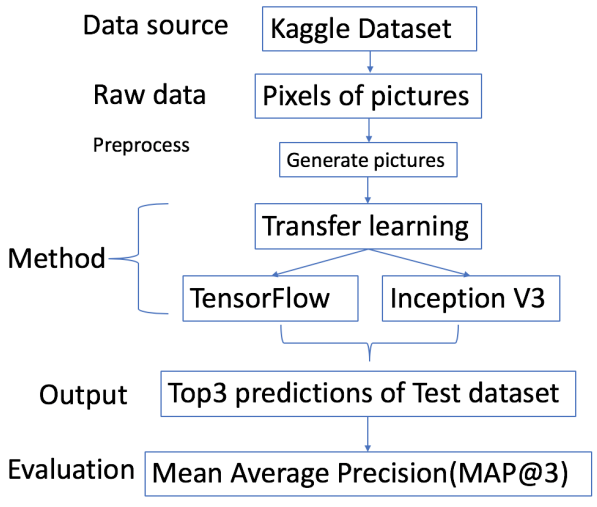

Figure 1. The flowchart of our system

4. Experimental Design

For this project, we build our model based on Convolution Neural Network using Inception V3 framework pre-trained with the ImageNet Dataset. We will first make data preprosessing to convert the data into the form we need for training. Then we will first train the Inception V3 model with a smaller dataset without fine-tuning its parameters, and compare its performance with the baseline. Then we will fine-tune the parameters of the system. Finally we will predict the categories of our test data with our trained model.

Data Preprocessing

The raw data provided by Kaggle on this competition are some .csv files separated into training data and test data. Each instance has several attributes including the coordinates of dots drawn on the plot, as well as some metadata of the sketch. Specifically, the coordinates of dots are separated into several strokes, each stroke contains several dots, and each dots has an x coordinate and a y coordinate. As a result, this attribute is a three-layer array. Since each sketch is different as the number of strokes and the number of dots in each stroke is not the same, we must first convert those dots into actual image files so that we can use them for training.

The training data contains the class label of each instance, while the test data does not. So we first convert the training instances into image files. We created 20000 images for each of the total of 340 classes, thus making about 7000000 images for training. Then we convert all of the test instances into image files. The test dataset has 112199 images. The strokes are all in black, and the background is white. This is to control the influence of colors in case it will affect the performance of training. Figure 2 shows some examples of the created sketches.

Modification of Inception V3 Model

The implementation of our transfer learning approach is to modify the Inception-v3 model,w here a new fully-connected layer was added. W We use the output of the bottleneck layer created by the Inception V3 model to train the fully-connected layer to handle doodle classification. Although using transfer learning slightly makes sacrifice of prediction accuracy, it is much quicker.

We use TensorFlow to realize Convolution Neural Network. Firstly, we create a tensor and a session and define a placeholder. Placeholder needs to be defined when passing the training data over at session time. For this time, we set the learning rate to be 0.0001 with no decay, batch size to be 128, and the optimizer to be the Adam Optimizer. We set the early stop function so that the training will stop if the loss stop decreasing for a long time.

We will use the Mean Average Precision @3 (MAP@3),

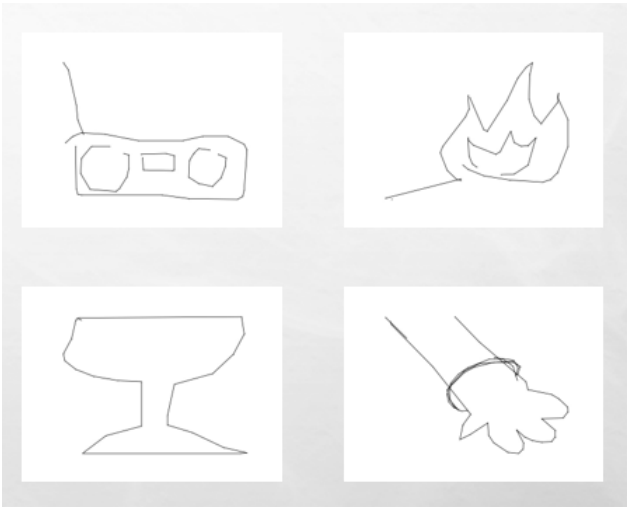

Figure 2. Examples of the Converted Sketches

which is the mean average of the accuracy of the top 3 predicted classes, as our evaluation metric. This is the requirement of the Kaggle Competition. The equation of MAP@3 is shown as below, where U is the number of scored drawings in the test data, P(k) is the precision at cutoff k, and n is the number of predictions per drawing.

$$MAP@3 = \frac{1}{U}\sum_{u=1}^{U}\sum_{k=1}^{min(n,3)} P(k) \tag{1}$$

We will use a Keras model as the baseline for comparison. There's a kaggle kernel which already build and validated this competition.[2] We will use the result of this kernel as our baseline.

Fine-tune Parameters

During this process, we mainly tune three parameters: Learning Rate, Batch Size, and Optimizer.

- **Learning Rate** If we just choose unchanging learning rate, the loss will first decrease quickly to some threshold and never keep dropping any more. So we decide to add learning rate decay loss validation stop decrease at a high-level value.

- **Batch size** A good selection of batch size can determine the descending direction firstly. There are lots of benefits of increasing batch size to a reasonable extent. First, memory utilization and parallelization efficiency of large matrix multiplication are improved. Second, the number of iterations needed to complete an epoch was reduced, and the processing speed for the same amount of data was accelerated further. Third, in a certain range, generally speaking, the larger batch size is, the more accurate the descending direction it determines and the smaller the training shock will be. We tried batch size from 64, 128 and 256. As we could see in Figure 3, the larger batch size, the better the result. However, there is no obvious increase for the accuracy and if batch size is too large, training speed would decrease. At last, we choose that the batch size is 256.

- **Optimizer** There are four kinds of optimizer we have tried: Adam, Adagrad, GradientDescent, and RMSprop. Figure 4 shows the curves of loss function and the validation accuracy with the increase of training steps using these four optimizers. From the figure, the Adam Optimizer has the best performance that both the loss function drops and the validation rises the quickest .After the bias correction of Adam gradient, the learning rate of each epoch has a fixed range, so that the parameters are relatively stable. Adam calculates different adaptive learning rates for different parameters and Adam is suitable for large data sets and high dimensional space. According to the result of comparison, we choose Adam Optimizer in our project.

Testing

Our final step is to predict the top 3 categories for the given dataset by Kaggle which contains 112199 images without labels. Since Inception V3 model feature vectors are used for transfer training as the input to the new fully-connected layer, the test steps are not quite the same as the normal ones. We test as follows:

1. Inception-v3 model needs to be loaded before testing to obtain the feature vector of the image.

2. Then the model of transfer training is loaded and the feature vector is used as input to obtain the predicted value.

After finishing the test, we return the top 3 predictions with the highest Soft Max value from the Inception V3 model for every test images. We store the key id of the images together with their 3 predictions in one file, as our final submission of this competition.

5. Experimental Results

Performance Comparison between Inception V3 Model with Baseline

According to our experiment result. Our Inception V3 model(without fine-tuning parameters) with 200 images per class, has 57% of validation accuracy, and 71% of MAP@3. In contrast, our baseline trained 6000 images per class, and

[2]Link:https://www.kaggle.com/jpmiller/image-based-cnn

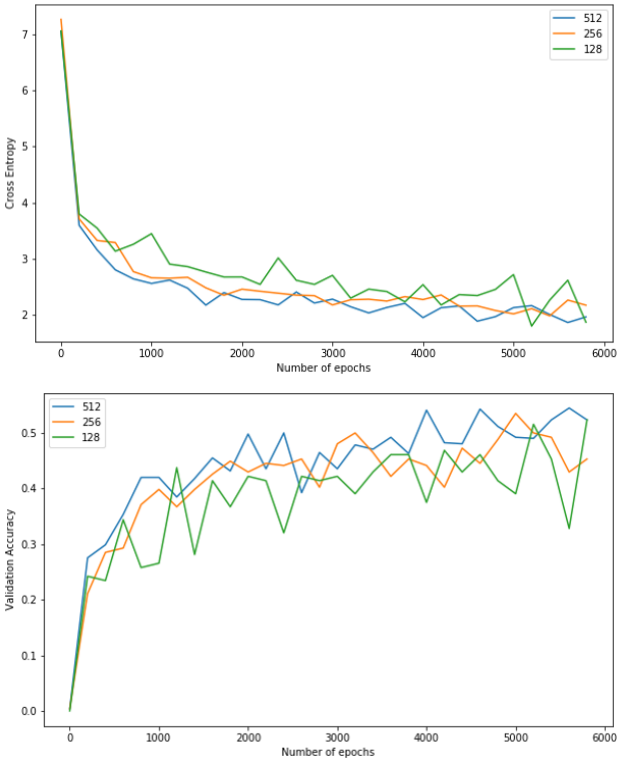

Figure 3. Loss and Accuracy Curves of Different Batch Sizes

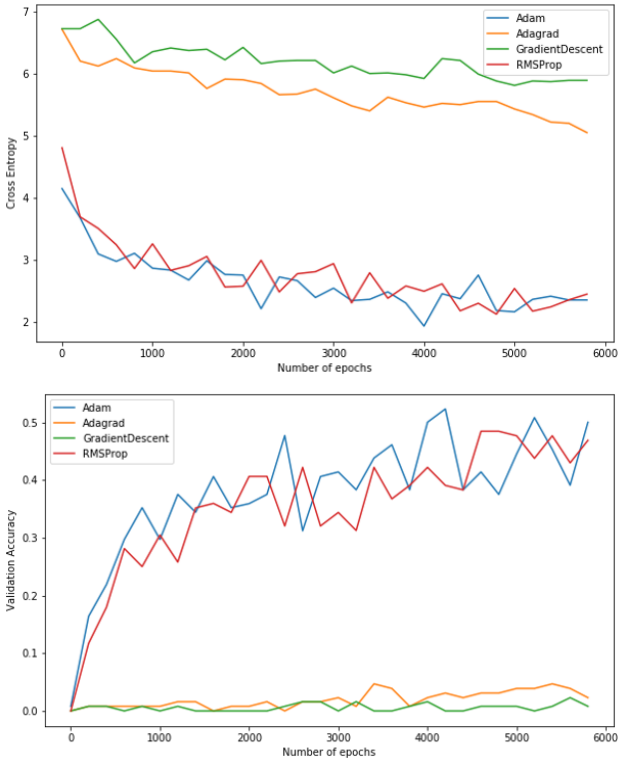

Figure 4. Loss and Accuracy Curves of Different Optimizers

had 60% of MAP@3. Our model achieves better prediction performance, with much less training data. The comparison is shown as Table 1.

Table 1. Comparison between Inception V3 and the Baseline

Model	Data	Accuracy	MAP@3
Inception V3	200/class	0.57	0.73
Baseline	6000/class	-	0.60

Performance of Our System after Fine-tuning Parameters

After four weeks of fine-tuning of the parameters of our system and training, with more than 17000 epochs, according to our last check point, the validation class accuracy has risen over 60%, and the MAP@3 has been over 80%. The comparison between our model and the baseline are shown as Table 2.

Table 2. Comparison between Our Model and the Baseline

Model	Data	Accuracy	MAP@3
Our Model	20000/class	0.60	0.80
Baseline	6000/class	-	0.60

Testing

We fit our model with the test dataset which has 112199 images with no labels. For each image, we print three most possible categories together with its key id, and save the results in one file for our submission of this Kaggle competition. Figure 5 is a screenshot of part of our submission file.

Limitation

We didn't use all of the dataset provided by Kaggle. On average, each category has more than 100 thousand of images. Limited by the computation power of our machine, and the timeline of this project, we only used 20000 images per category. Besides, we did not compare our performance with any sate-of-the-art CNN models. Since we do not find any papers about applying CNN to doodle image classification, we can not make sure if our method is good enough to be one of the most accurate models for this field of problems.

6. Conclusions

Our work is a Kaggle Competition about hand-drawn doodle image classification. Traditional Convolution Neural Network methods focus on real images with high resolution and may not fit well with doodle images. We use the

```
key_id,word
9000003627287624,radio train ant
9000010688666847,hockey_puck toaster pool
9000023642890129,toe castle fork
9000038588854897,mountain the_eiffel_tower necklace
9000052667981386,fireplace parachute parrot
9000057427034623,fence stitches jail
9000065506980882,wine_glass hourglass wine_bottle
9000087586309806,dolphin key spoon
9000092580281382,mosquito arm eyeglasses
9000096661653918,hourglass vase anvil
9000102548572430,octopus snowflake campfire
9000109525154374,stove oven telephone
9000117423882596,eyeglasses mountain camel
9000118400618039,panda hurricane blackberry
9000119463725679,harp watermelon helmet
9000156567772007,pencil crayon shorts
9000159584429954,sailboat ice_cream popsicle
9000161602785100,train hospital calendar
```

Figure 5. Example of Our Submission File

transfer learning method based on the pre-trained Inception V3 model, adapting the pre-trained model to fit with the large scale doodle image dataset, thus speed up training and improve the accuracy. We use a total of 7 million doodle images as the training data, and train the model with TensorFlow on a powerful machine with 4 Tesla K80 GPUs. This method can achieve a 50% Top 3 Mean Average Precision (MAP@3) in no more than 1000 epochs of training within one hour, showing a higher learning slope of MAP@3 improvement. We finally achieve 60% accuracy as well as 80% MAP@3. From our experiments, we found that transfer learning, compared to other main stream CNN methods, has quicker training speed and better performance for hand-drawn doodle images. In future works, increasing the scale of training data may further improve the performace of our system.

References

Abadi, Martín, Barham, Paul, Chen, Jianmin, Chen, Zhifeng, Davis, Andy, Dean, Jeffrey, Devin, Matthieu, Ghemawat, Sanjay, Irving, Geoffrey, Isard, Michael, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pp. 265–283, 2016.

Kara, Levent Burak and Stahovich, Thomas F. An image-based, trainable symbol recognizer for hand-drawn sketches. *Computers & Graphics*, 29(4):501–517, 2005.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Lawrence, Steve, Giles, C Lee, Tsoi, Ah Chung, and Back, Andrew D. Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, 8(1):98–113, 1997.

Sahiner, Berkman, Chan, Heang-Ping, Petrick, Nicholas, Wei, Datong, Helvie, Mark A, Adler, Dorit D, and Goodsitt, Mitchell M. Classification of mass and normal breast tissue: a convolution neural network classifier with spatial domain and texture images. *IEEE transactions on Medical Imaging*, 15(5):598–610, 1996.

Sun, Zhenbang, Wang, Changhu, Zhang, Liqing, and Zhang, Lei. Query-adaptive shape topic mining for hand-drawn sketch recognition. In *Proceedings of the 20th ACM international conference on Multimedia*, pp. 519–528. ACM, 2012.

Szegedy, Christian, Ioffe, Sergey, Vanhoucke, Vincent, and Alemi, Alexander A. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, volume 4, pp. 12, 2017.

Vishnu, Abhinav, Siegel, Charles, and Daily, Jeffrey. Distributed tensorflow with mpi. *arXiv preprint arXiv:1603.02339*, 2016.

Zhang, Yong, McCullough, Christine, Sullins, John R, and Ross, Christine R. Hand-drawn face sketch recognition by humans and a pca-based algorithm for forensic applications. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 40(3):475–485, 2010.