Teach the computer to recognize freehand sketching

Fengdi Li, Yarou Xu, Yifan Wu

1. Project's Goal and Objectives：

In this project, we want to teach the computer to learn to recognize doodling. After training with pre-classified sketching, we hope the model can correctly classify the object belongs to the previous classes drawn by ourselves.

After training on pre-labeled sketches, we expect that the model will recognize any drawing that belongs to the pre-trained categories.

The prototype of this project is the Google AI game 'Quick, Draw!'. We learn from the Tensorflow tutorial [1] and want to tune this model by adjusting several hyperparameters, such as the number of convolutional layers, the number of LSTM layers, the embedding matrix dimensions, and adding regularizations into the loss function. Also, we will try to add pooling layers between convolutional layers to decrease input dimensions and add drop-out layers to avoid over-fitting and force hidden units to learn more information.

2. Data：

We will use the preprocessed training data for the Google AI game "Quick, Draw!"[2]. This dataset is a collection of 50 million drawings across 345 categories which are contributed by the player of this game. Due to the computing and storage memory limits, we decide to train our version of model on only 10 categories of drawings. For each category, there are 10000 training data and 1000 test data. The reason why we choose this dataset is that it is the original data set Google used to generate their AI model in Quick, Draw!, and successful models have been trained using this dataset. It will be the appropriate data to use if we want to make changes to the model networks structure.

Each hand-drawn training picture is a vector that tagged with metadata such as what the players are asked to draw, the player's country, and a unique identifier for this drawing across all drawing. The preprocessed data is uniformly scaled, stroke width adjusted to one-pixel width, aligned to the top-left corner to have the minimum value of 0 and simplified using the Ramer-Douglas_Peucker algorithm[3] with an epsilon value of 2.0. The data is formatted as TFRecord files.

One of the possible limitations of this data is that, since it was prefiltered to limit the data size and everyone have their own drawing style, it is possible that our dataset is not diverse enough within each category. In the future, it will boost the robustness of the model if the drawings are sampled across different countries to accommodate different drawing styles for the same object.

The approach we will take is a supervised method integrating with TensorFlow. The input features will be the vector values of the stroke in the drawing and the expected output will be the predicted category for drawing.

3. Assessment Metrics

In this project, we will use a multi-class cross-entropy loss function with a regularization penalty. Because the output of our model after the SoftMax layer is the probability of each category which ranges from 0 to 1, when the actual output is close to the desired output for all training inputs, the cross-entropy will be close to zero. Otherwise, the larger difference between

the actual output and desired output, the larger cross-entropy loss. On the other hand, compared with other loss function, such as the squared error cost, the cross-entropy cost can avoid the problem of learning slowing down. By computing the derivative of the cross-entropy cost with respect to the weights, we know that the derivative is controlled by the error of the outputs. Therefore, the larger the error, the faster the model will learn.

The baseline dataset that will be used to evaluate our model comes from the original Google AI Quick Draw dataset. And the ratio of the test set over the training set is 1:10. Also, we will test the model with plots created by ourselves and realistic photographs.

We will use the prototype model from the Tensorflow tutorial [1] and a Random Forest combining with PCA dimensionality reduction model as the baseline models. Our model is to tune the prototype one, by comparing them, we can evaluate whether our tuning method is effective or not. The Random Forest method is very powerful and efficient when dealing with multi-class classification problems with low dimension features. We can compare our deep learning method with this shallow learning method, and hope that our model will outperform it.

Now there are many state-of-the-art convolutional neural networks such as VGG16, VGG19, ResNet50, InceptionV3, and Xception for image classification. And all of them are very complicated neural networks which require vast computing and storage resources. We hope our simpler model can learn classification faster with a slight reduction in accuracy.

4. Approach

Our model will contain a neural network which consists of a convolutional neural network, a Recurrent neural network and a SoftMax output layer. This neural network will take in a freehand sketching plot in the format of a matrix of pixel points and learn the correlations and patterns between each pixel so that it can recognize the object category. The CNN contains convolutional layers, pooling layers and fully connected layers with drop-out. It can decrease features' dimensions while extracting major information from the large input pixel matrix and passing it to the following RNN. In the RNN, we use LSTM units to learn the extracted information, because LSTM performs excellently in dealing with sequential data. We think that lines in a freehand drawing are always depended on each other, and this makes LSTM a suitable tool. And we hope that the LSTM units can capture and memory relationships between neighboring pixel point so that the model can better understand the sketching plot. This model is trained on black-and-white images, therefore, we assume that it can only perform well on the same type of images. It may not be able to classify realistic photographs accurately, because they are in RGB colors with three dimensions. But we will try to convert them into one-dimension pixel matrixes by computing the mean value and apply our model on it. The second limitation of our model is that we only train it with 10 categories images, therefore, it can only predict these10 categories.

In this project, we will only train the model on local computers with Tensorflow, because running our model on the cloud is too expensive and may suffer from service outages problem. And we are not sure that our project will have amazing outcomes, the cost may not be worth it. However, we may update our model by training with more categories' sketches on the cloud in the future.

Reference:
[1] https://www.tensorflow.org/tutorials/sequences/recurrent_quickdraw
[2] https://console.cloud.google.com/storage/browser/quickdraw_dataset/sketchrnn/?pli=1
[3] https://en.wikipedia.org/wiki/Ramer%E2%80%93Douglas%E2%80%93Peucker_algorithm