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

The prototype of this project is the Google AI game '*Quick, Draw!*'. Inspired by/Learned from TensorFlow tutorial, [1] we would like 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 data provided by the Google AI game "*Quick, Draw!*" [2]. This dataset is a collection of 50 million drawings across 345 categories which are all contributed by the player of this game. The reason why we choose this dataset is that it is the original data set Google used to generate their AI model for 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. But, due to the computing and storage memory limits, we decide to train our model only on 10 categories of drawings. For each category, there are 10000 training images and 1000 test images. 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 rendered into 28 ×28 grayscale bitmaps.

The approach we will take is a supervised learning method integrating with TensorFlow. The input features will be vectors of the pixel value of the drawing and the expected outputs will be the categories for drawings.

3. Assessment Metrics

In this project, we will use a multi-class cross-entropy loss function integrating with a regularization penalty as the loss metric. 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 outputs are close to the desired outputs for all training inputs, the cross-entropy will be close to zero. Otherwise, the larger differences between the actual outputs and desired outputs, the larger cross-entropy loss. On the other hand, compared with other loss functions, such as the squared error cost, the cross-entropy cost can avoid the problem of learning rate slowing down. By computing the derivative of the cross-entropy cost with respect to the weights, we know that the derivatives are controlled by the output error. 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 images created by ourselves and realistic photographs.

We will use the prototype model from the TensorFlow tutorial [1] and a Random Forest integrating with PCA dimensionality reduction as the baseline models. Our model is generated by tuning the parameters and structures of the prototype one. By comparing them, we can evaluate whether our

tuning method is effective or not. On the other side, the Random Forest method is very powerful and efficient in dealing with multi-class classification problems with low-dimensional features. We can compare our deep learning method with this shallow learning method, and hope for a better classification result.

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 may slightly compromise accuracy in our simpler model to learn classification faster.

4. Approach

Our model will have a CNN LSTM architecture. The CNN contains convolutional layers, pooling layers and fully connected layers with drop-out. It can take in a freehand sketching graph in the format of vectors of pixel points and decrease the dimension of features by extracting major information from the large input pixel matrix. Then we use LSTM units to learn the extracted information since LSTM performs excellently in dealing with sequential data. We think that strokes in the freehand sketching are always interdependent with each other, which makes the LSTM a suitable tool for processing freehand drawing data. We hope that the LSTM units can capture and memorize relationships between neighbor pixel points, so that the model can better understand the sketching.

The first limitation is that this model is only trained on black-and-white freehand sketching, therefore, it may not be able to classify other types of images accurately, such as realistic photographs with RGB colors. But we will try to convert this chromatic image into one-dimension pixel matrices by computing the mean value and apply our model on it. The second limitation is that we only train on 10 categories images, therefore, it can only work for these10 categories.

In this project, we will train the model on local computers with TensorFlow. Considering that the prototype model may not have desirable outcomes, we would like to avoid any expenses and service outage problems at current stage, and update it 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