Auto Generation Model for Drawings

Xinrui Ying Jingzhi Zhang Ye Zeng Siqi Du Yuheng Zhang Mingyang Xia

1 Introduction

$Draw\ and\ Guess$ is a popular game in which one player draws a picture based on a key word (category) and another player guesses the key word (category). But sometimes the player doesn't draw it well enough for the other player to guess. From this, we came up with the idea of using machines to help us drawing.

In this project, our team want to develop a auto generative model and train the machine to complete a figure based on current sketch drawing. This model will mainly utilize the structure of recurrent neural network(RNN), which will take one or a sequences of strokes as input, and predict what the object might be and complete the rest of the strokes.

2 Motivation

Drawing can help people to expand their imagination and stimulate our creativity, more importantly, it is a crucial way for us to observe and document the world since childhood. There are already many work related to paints, such as training the machine to mimic famous artists style or transforming a picture to a specific painting style. Based on the previous achievement, we want to develop a more interactive system that can incorporate both human and machine to accomplish the same work. This model might enable many creative application, one potential application is that it can help children to draw a simple sketch, or it can even inspire the artists based on what the machine drawn in an unprecedented and novel way.

3 Problem Formulation

3.1 Dataset and Environment

We obtained the dataset from $Quick,\ Draw!$, which contains vector drawings where people are asked to draw a particular object in less than 20 seconds. Overall, it has about 75K samples (70K Training, 2.5K Validation, 2.5K Test), images are selected randomly from each category. We will also be providing all 50 million samples from 345 categories if we think it's necessary to use them all. The tutorials and links for getting the dataset is following: `https://github.com/googlecreativelab/quickdraw-dataset#get-the-data`

We are going to use Python 3 programming language for most of our developing, and using GPU and TensorFlow.

3.2 Existing works

There are a lot of existing works that imitate painting of human with different ways:

Reinforcement Learning Approach to Automatic Stroke Generation Artist Agent, Automatic Stroke Generation in Oriental Ink Painting, which inputs a digital picture and produce a series of paint brush strokes.However, these works focusing on drawing of the original images instead of generate new vector images.

Neural Network Approach to Image Generation with vector images: different approaches to transform between digital images and sketches: `https://twitter.com/bgondouin/status/818571935529377792` and Image-to-Image Translation with Conditional Adversarial Networks. However, there are few work on generating vector images due to the lack of public datasets. Related previous work including Recurrent Net Dreams Up Fake Chinese Characters in Vector Format with TensorFlow, which modeling Chinese characters as series of pen stroke actions. And ShadowDraw,predicting the finished drawing from unfinished brush strokes, which used a dataset of raster images and extracted vectorized features. The Sketchy dataset provided 70K vector images with related digital images, made it possible for us to have deeper exploration of sketches. Here we use a larger dataset of vector sketches from Quick Draw dataset.

Sketch Recognition Approaches in recognizing free hand sketches using local features : Sketch-Based Image Retrieval: Benchmark and Bag-of-Features Descriptors, which compares different local features Shape Context, Spark feature, Histogram of Oriented Gradients (HOG) and sketched HOG, HOG based features outperform others. Because HOG is a highly optimized feature descriptor for encoding edge and gradient properties of images, and human sketches' main property is the edge and gradient. We did not use local feature, we generate new vector images based on probability distribution .

4 Approach and concrete models

We use recurrent neural network-based generative model to produce sketches of specific objects in vector format. Each stroke will be a five dimensional vector $(p_1, p_2, p_3, p_4, p_5)$.

- p_1, p_2 is the Δx, Δy between the current starting point to the previous point.
- $p_3 == 1$ if the next point will be connected with current point.
- $p_4 == 1$ is the next point will not be connected with current point.
- $p_5 == 1$ indicated finished.

Therefore, only one of p_3, p_4, p_5 will be 1 and the other two will be 0 for any vector.

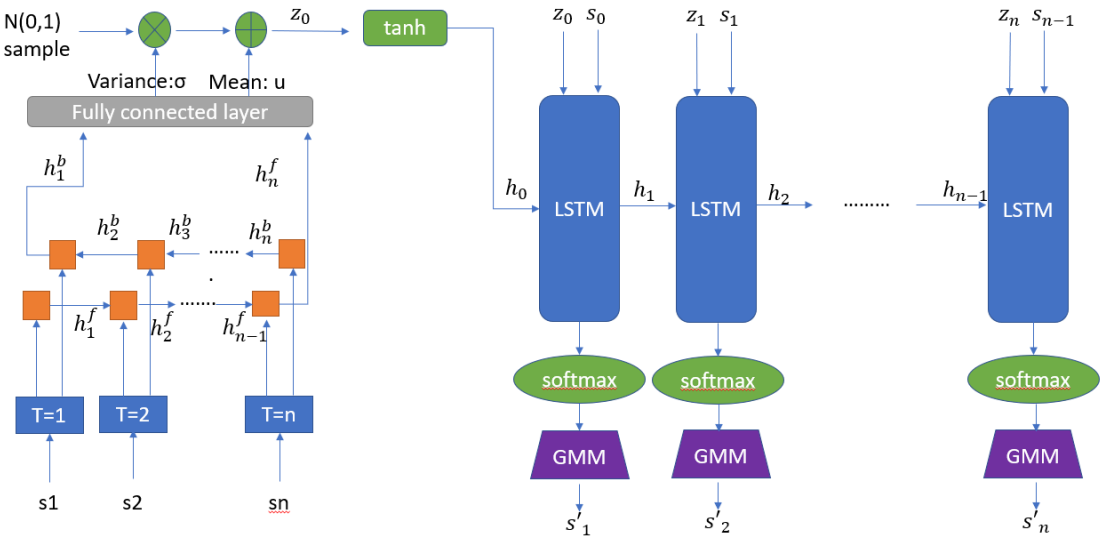

Figure 1: model

We also introduce an optional a bidirectional RNN model as variational encoder, that takes a sketch as an input and generate a latent vector z. Specifically, we feed the sketch sequence, S, and also the

same sketch sequence in reverse order, $S_{reverse}$, into two encoding RNNs and the forward sequence generate the μ and the reverse sequence generate σ. We then use normal distribution that takes the generated μ and σ to compute the latent vector z.

Our RNN at time step i will take s_i, h_i and an optional latent vector z. The output at each time step are the parameters for a probability distribution of the next data point s_{i+1}. We will model the (p_1, p_2) as a Gaussian mixture model with M normal distributions, with will have (q_3, q_4, q_5) as categorical distribution to GMM model and (p_3, p_4, p_5) as the ground truth.

$$p(\Delta x, \Delta y) = \sum_{j=0}^{m} \Pi_j N(\Delta x, \Delta y | \mu_{x,j}, \mu_{y,j}, \sigma_{x,j}, \sigma_{y,j}, \rho_{x,y,j})$$

where μ is mean, σ is standard deviations and ρ is the correlation. The vector Π is of dimension M, our output will also contains 3 logits needed to generate (q_3, q_4, q_5).

To make our model simple as to know when to stop, we introduce a constrain to the maximum number of steps to stop drawing. If somehow we stop early than the maximum steps, we will make $s_i = (0, 0, 0, 0, 1)$ for all remaining steps.

we want to maximizes the log-likehood of the generated probability distribution, so our loss for RNN is simply defined as

$$L_p = \frac{1}{-N_{max}} \sum_{j=0}^{m} \Pi_{j,i} N(\Delta x_i, \Delta y_i | \mu_{x,j,i}, \mu_{y,j,i}, \sigma_{x,j,i}, \sigma_{y,j,i}, \rho_{x,y,j,i})$$

$$L_s = \frac{1}{-N_{max}} \sum_{i=0}^{N_{max}} p_{3,i} \log(q_{3,i}) + p_{4,i} \log(q_{4,i}) + p_{5,i} \log(q_{5,i})$$

$$L_r = L_s + L_p$$

Our loss for VAE encoder part is simply the base kl divergent loss with a weight w_{kl}.

$$L = L_r + w_{kl} L_{kl}$$

5 Experiments

We designed two main experiments with our model for different applications purposes; one is VAE Reconstrucion, the other is Autocomplete with RNN. The first experiment is aimed to generate various reconstruction sketches S' with the original sketch S as inputs, as we want the model to produce different forms of the same object. Moreover, we expect our model to be more interactive and finish an incomplete sketch from humans. Thus, we design our second experiment only using the decoder as the backbone to generate complete sketches based on the inputs, which are either one stroke or a sequence of strokes.

To conduct our experiments, we chose six categories from the dataset $Quick, Draw!$, which are birds, horses, skulls, trees, computer, dragon. In each experiment, We will record the losses and evaluate the influences of different parameters on the final results as well as the overall model performances for different classes.

5.1 Reconstruction VAE

First, since our model is Variational Autoencoders like structure. Thus, the loss of the model is consists of Reconstruction Loss Lr, and the Kullback-Leibler Divergence Loss L_{KL}.We also set a weight *w* to balance the importance of these two losses. To evaluate the significance of the weight w and how it influences the overall result, we will set different values of w from small to big; there are 0, 0.25, 0.5, 1. Second, the essence of VAE is to compress the data through the encoder and reconstruct

the data through the decoder. The latent space is the key to this process, as we want to keep the primary information of the data while reducing the dimension and keeping the reconstruction loss low, the dimension of the latent space is crucial. We will set different size of latent space which are 32, 64 and 128 to find the best value.

5.2 Autocomplete RNN

In the second experiment, we train will train the model to auto-complete unfinished drawings. To achieve this, we will feed a series of points to the model and then generate a complete drawing based on the given points in specific category. Once the sequence of points are fed into the RNN, we can get a hidden state. Then we will feed the hidden state back to the RNN to generate new points and finish the sketch. In the experiment, we choose LSTM as the RNN cell type. We train on Birds, Horses, Skulls, Trees, Computer, Dragon 6 categories of drawings. And we tune 2 hyper-parameters of the model: size of RNN—rnnsize and the max number points of one drawing—lenmax. First, we tried different number of hidden units in RNN. And we tune the hyper-parameter rnnsize with values: 128, 256, 512. Second, it is important to decide when to stop drawing for the auto-complete process. If it stops early, the sketch may not complete. If it stops to late, then there may be redundant strokes. And the number of the strokes of the drawings in the datasets are various. Therefore it is difficult to decide the maximum number and to train. We define the max number points of one drawing as a hyper-parameter lenmax and train on these values: 150, 200, 250.

6 Remaining Steps (Timeline)

Nov.14-Nov-23: Continue doing the remaining experiment.
Nov.3-24 Developing the drawing demo
Nov.24 Project representation
Dec.1 Project final report

7 Breakdowns of individual contributions

Xinrui Ying: I worked on building the bidirectional RNN model that takes the sketch vectors as input and generate latent vector z. I also train and test our model on category "Computer".

Siqi Du: literature reviewing, participating in designing the model, writing the training part, trainning and testing our model on the class "Skull".

Yuheng Zhang: participating in designing the model, writing the js code part for interactive usage and providing a trained model on the class "Tree".

Jingzhi Zhang: Read papers and participated in designing the models. Wrote the code of the decoder part of vae and trained the model on the computer class of the dataset.

Mingyang Xia: Read papers and participated in designing the models. Wrote the code of the transfer the model into jason format which could be used in demo. I also provided a trained model on the class "Dragon".

Ye Zeng: I participated in background research and designed our RNN based VAE model, and also programmed the loss function for the encoder and decoder. For the experiment, I take charge of training the model on the "Birds" class.