Sketch Composer

Group *Neural Picasso* (Jiahao Liu , Zheyuan Zhou , Yun Li)

1. Introduction

We study the sketch generating problem. There has been a big body of literature on this topic, among which SketchRNN ([1] Ha and Eck, 2017) is most relevant to our work. SketchRNN proposed a recurrent neural network (RNN) model to construct sketches of common objects. The model was trained on a dataset of human-drawn sketches of different classes (e.g. apples, bicycles, etc.). In their paper, a sketch was represented as a sequence of points, and the authors applied a seq2seq VAE architecture to train the model from end to end. However, empirically, RNNs or even LSTMs cannot perfectly handle extremely long sequences, while human-drawn sketches often contain more than hundreds of points. This poses difficulty in the training process. Moreover, their model failed to reflect the compositionality of human sketches. When humans draw, they do not visualize an object as a sequence of points but instead as several composition parts. This human thinking process is summarized by the theory of "chunking" in behavioral psychology. The theory states that individuals process pieces of information set separately, and the pieces are bound together into a meaningful whole. Therefore, in order to overcome the two drawbacks discussed, we propose to decompose sketches into strokes (usually no longer than 20 points), and then generate sketches by composing these strokes.

2. Methodology

2.1 Data

The dataset we use is the Quick Draw dataset ([6]), which is a dataset of vector drawings obtained from *Quick, Draw!* ([7]), an online game where the players are asked to draw objects belonging to a particular object class in less than 20 seconds. The total dataset contains 50 million drawings across 345 classes. Each class of Quick Draw contains 70K training samples, 2.5K validation samples, and 2.5K test samples.

Each sketch in the dataset is represented as a sequence of strokes, and each stroke is represented as a sequence of two-dimensional points. The format of the drawing array is as the following:

```
[
  [  // First stroke
    [x0, x1, x2, x3, ...],
    [y0, y1, y2, y3, ...],
    [t0, t1, t2, t3, ...]
  ],
  [  // Second stroke
    [x0, x1, x2, x3, ...],
    [y0, y1, y2, y3, ...],
    [t0, t1, t2, t3, ...]
  ],
  ... // Additional strokes
]
```

FIGURE 1. Format of a Drawing Array in Dataset

The data is preprocessed by removing the strokes whose length is longer than 30 or shorter than 3. We also remove the sketches whose number of strokes is larger than 10.

2.2 Model

Our system can be broadly divided into two parts. First, an autoencoder learns the representation for individual strokes by minimizing the reconstruction loss. The learned latent representation can then be used for clustering and labeling these strokes. A sketch that is originally represented as a sequence of points will be transformed into a list of stroke labels. Then an LSTM-based variational autoencoder model (named *Stroke Composer*) learns to "compose" the stroke labels generated in the previous step. Additionally, a feed-forward Neural Network (named *StratNN*) is trained to predict the start point for each stroke. The model architecture is shown below, and we will elaborate on these parts in the following sections.

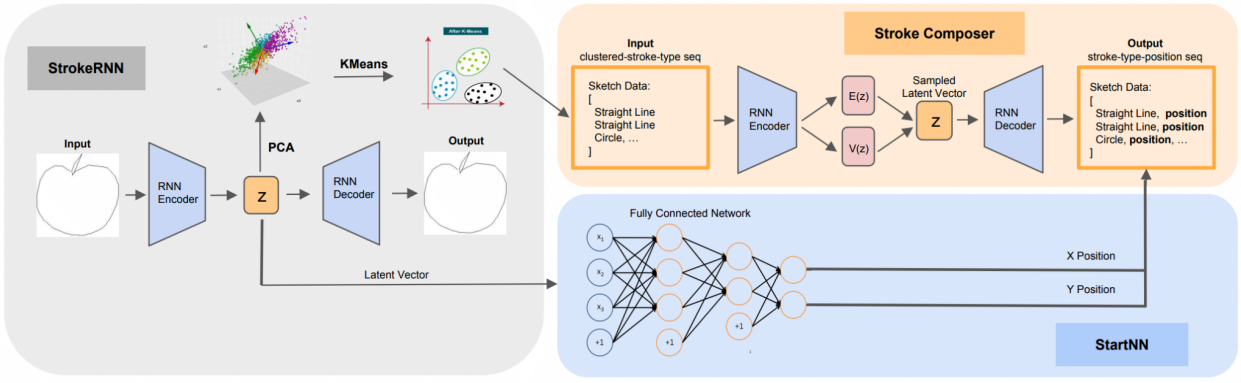


FIGURE 2. Model Architecture Overview

2.2.1 Learning Stroke Representation

To learn the representation of strokes, we train an autoencoder (*StrokeRNN*) to learn to reconstruct the strokes. A stroke is originally formatted as a sequence of points, therefore we use LSTM as the encoder and the decoder. The autoencoder is trained by minimizing the Mean Square Error between inputs and outputs. The reconstruction loss function is shown below. Here N refers to the number of batches, L refers to the length of the stroke, y represents the predicted relative position, and y-hat represents the ground truth.

$$L = \frac{1}{N}\sum_{j}^{N}\sum_{i}^{L}\|y_i - \hat{y_i}\|^2$$

Besides that, additional feed-forward deep networks are trained to predict the absolute start point of the stroke (*StartNN*). We formulate it as a regression problem and train the neural network by minimizing the Mean Square Error. Here s refers to the predicted starting point and s-hat represents the ground truth.

$$L = \frac{1}{N}\sum_{i}^{N}\|s_i - \hat{s_i}\|^2$$

After training the autoencoder, we take the latent vector of the input stroke sequence as its representation. Then we apply the K-Means algorithm to cluster the strokes. However, the latent vector is of high dimension, while K-Means fail in such a situation. Therefore, we pass the latent vectors into PCA to reduce the dimension to 16 and then cluster the latent vectors. We use “CategoryName_ClusterIndex” to label the strokes, and thereby transform the sketch data into a sequence of stroke labels. For example, a sketch can be represented as [“apple_1”, “apple_3”].

2.2.2 Sketch Composer

The Sketch Composer learns to predict the next stroke label based on the current input and the previous steps. Specifically, we use a variational Seq2Seq to learn the sequential data. Both the encoder and the decoder are LSTMs, and the network is trained by simultaneously minimizing the reconstruction loss and KL-divergence. Since the LSTM predicts categorical labels, we select Cross-Entropy Loss as the reconstruct loss function. The loss function is shown below:

$$l_i(\theta, \phi) = -\mathbb{E}_{z\sim q_\theta(z|x_i)}[\log p_\phi(x_i \mid z)] + \mathbb{KL}(q_\theta(z \mid x_i)\|p(z))$$

3. Challenges

One challenge is to come up with the architecture of the models. In the beginning, we only had the idea that the original paper (SketchRNN) had several weaknesses, but we did not know how to solve these problems. We had several brainstorms before establishing the current scheme. Another challenge we encountered in our project was modeling up the VAE for the Sketch Composer network. The LSTM encoder has two sets of final states as output - the hidden state and the cell state. We were not sure how these two states should be fed into the VAE and then passed on to the decoder LSTM. After rounds of trial and error, we ended up passing only the cell state from the encoder LSTM to the VAE (and then the

decoder LSTM). We initialize the hidden state in the decoder LSTM with zeros. We find this structure to give the best reconstructing results. Since the encoder output of the cell state captures an aggregation of data from all previous time-steps that have been processed, whereas the hidden state captures only the characterization of the last time-step's data, we think using only the cell state would be fine for our purpose.

4. Results

We show some generated sketch samples in the following figure. Our model can generate human identifiable results.

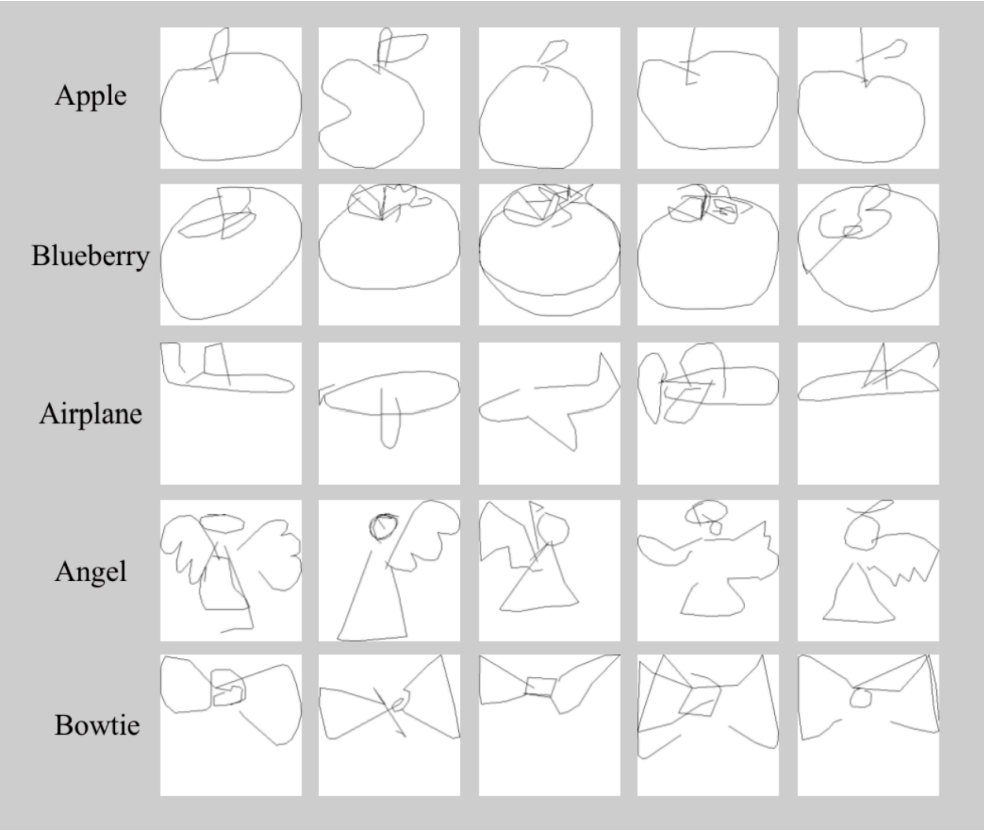


FIGURE 3. Sketch Composer Final Results for Different Categories

We also show the loss for different categories of different epochs. Table 1 shows the reconstruction loss of the *StrokeRNN* model. Table 2 shows the sum of reconstruction loss and KL divergence of the Stroke

Composer model. Table 3 shows the mean square error of the *StartNN* model.

	Bowtie	**Angel**	**Airplane**	**Blueberry**	**Ant**	**Basket**	**Bed**	**Bird**
Epoch 0	704.68	362.9	671.63	339.82	259.5	731.83	997.08	347.31
Epoch 500	37.64	13.63	37.9	15.75	**24.22**	34.22	64.38	13.85
Epoch 800	**34.25**	**13.45**	**36.71**	**15.73**	25.47	**30.85**	**47.05**	**11.38**

TABLE 1. *StrokeRNN* Reconstruction Loss for Different Categories on Different Epochs

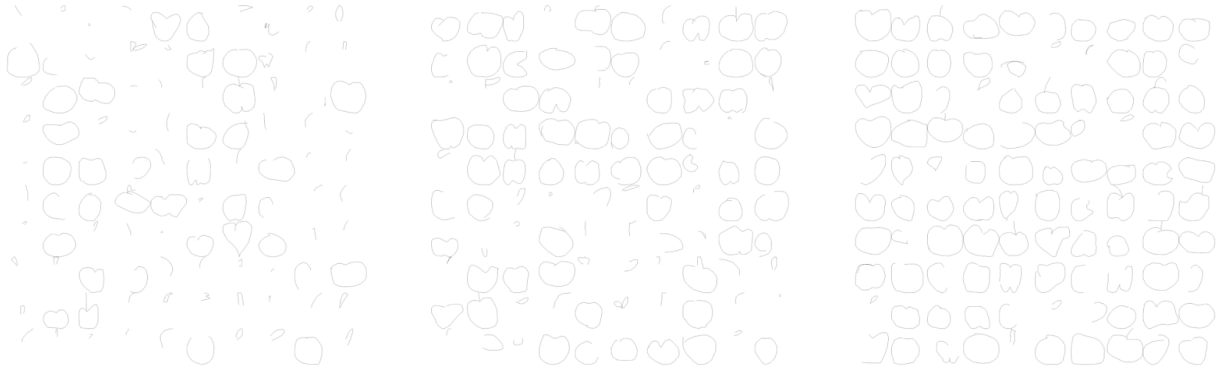
	Bowtie	**Angel**	**Airplane**	**Blueberry**	**Ant**	**Basket**	**Bed**	**Bird**
Epoch 0	1.39	1.41	1.42	1.45	1.45	1.45	1.46	1.49
Epoch 500	**0.16**	0.24	**0.2**	0.24	0.22	**0.21**	0.22	0.23
Epoch 800	0.18	**0.22**	0.19	**0.22**	**0.21**	0.24	**0.20**	**0.23**

TABLE 2. Stroke Composer CE and KL Loss for Different Categories on Different Epochs

	Bowtie	**Angel**	**Airplane**	**Blueberry**	**Ant**	**Basket**	**Bed**	**Bird**
Epoch 0	15178.34	8490.41	21040.8	5250.58	40296.33	5981.5	8692.87	9356.87
Epoch 50	1595.27	1722.58	2219.32	4429.05	3170.22	1373.44	2565.85	2565.81
Epoch 100	**1340.86**	**1625.07**	**1691.57**	**3610.42**	**3147.34**	**1322.1**	**2116.45**	**2425.45**

TABLE 3. *StartNN* Mean Square Error Loss for Different Categories on Different Epochs

In the following figure, we demonstrate a stroke cluster (in the apple category) given by K-Means on different *StrokeRNN* training epochs. We can see the model gradually learns to optimize the hidden representation for strokes, and thus the K-Means algorithm can easily cluster these strokes based on their semantics. The following figure shows the “apple body” cluster.



(10 epochs) (100 epochs) (500 epochs)

FIGURE 4. Stroke Clustering of "Apple Body" on Different Epochs

5. Reflection and Discussion

- **Q**: How do you feel your project ultimately turned out? How did you do relative to your base/target/stretch goals?
    - **A**: We are satisfied with our final result, and we think we have achieved our basic goals.
- **Q**: Did your model work out the way you expected it to?
    - **A**: Yes. Our model is able to compose the strokes into sketches.
- **Q**: How did your approach change over time? What kind of pivots did you make, if any?
    - **A**: We first think of composing the strokes into sketches by using several parallel VAEs. However, we realize it's important to "give names" to (or say discretize) the strokes and apply an LSTM VAE to learn to compose these discrete labels.
- **Q**: Would you have done differently if you could do your project over again?
    - **A**: We would carefully design the data processing code. We would index each stroke data in a table (just like an entry in the database) which would make the future training process much easier.
- **Q**: What do you think you can further improve on if you had more time?
    - **A**: Stroke positions are related to each other but in our model, we ignored this point. If we had more time, we would train a Graph Neural Network to learn such relationships.
- **Q**: What are your biggest takeaways from this project/what did you learn?
    - **A**: We learn that we need to carefully design the data processing API to support the different needs when exploring different deep learning architectures.

6. Reference

[1] Ha, D., & Eck, D. (2017). A neural representation of sketch drawings. *arXiv preprint arXiv:1704.03477*.
[2] Ge, S., Goswami, V., Zitnick, C. L., & Parikh, D. (2020). Creative sketch generation. *arXiv preprint arXiv:2011.10039*.
[3] Kingma, D. P., & Welling, M. (2013). Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*.
[4] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735-1780.
[5] Baggio, G., Van Lambalgen, M., & Hagoort, P. (2012). The processing consequences of compositionality. In *The Oxford handbook of compositionality* (pp. 655-672). Oxford University Press.
[6] Quick Draw Dataset, available at https://github.com/googlecreativelab/quickdraw-dataset.
[7] *Quick, Draw!*, available at https://quickdraw.withgoogle.com/.