Generating Quick Drawings with LSTM Recurrent Neural Networks

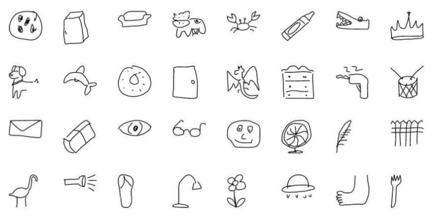
Michelle McGhee (mmcghee@stanford.edu), James Ortiz (jameso2@stanford.edu)
Department of Computer Science, Stanford University

Motivation

- We want to create a program that takes in as inputs an object class and the first few points of a drawing and returns a prediction of the complete drawing as output.
- For this task, we train an LSTM recurrent neural network that had been developed to take in a string of text and output the handwritten version of the text.
- Our goal was to see if we could use a neural network that had been used to generate handwriting in order to complete drawings given that writing and drawing are related tasks.

Data

- We used Google's Quick Draw dataset, a collection of 50 million drawings across 345 categories.
- The drawings are stored as collections of points *(x, y, t)*, where *x, y* are the x- and y- coordinates, and *t* is an indicator variable indicating whether the point occurs at the end of a stroke.
- We focused on one object category, i.e. cars, to test the neural network's ability to complete drawings.
- Specifically, we used 99,740 drawings of cars:
  - 96,740 for our train set
  - 100 for our test set

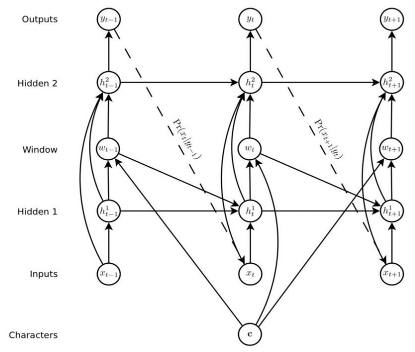


Encoding the Input Testing Data

- We used a GloVe model trained on a Wikipedia corpus of 6 billion tokens to compute a 50-dimensional word embedding for the object class.
- We then extracted the first fifty points from each test drawing and flattened these arrays into vectors.
- Finally, we concatenated the word embedding to each of the flattened vectors and fed the resulting vectors as inputs to the neural network.

LSTM Recurrent Neural Networks

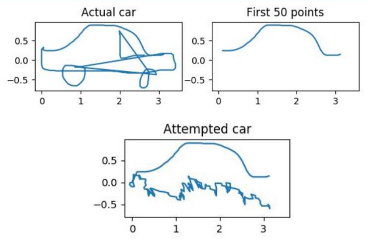
- We used the LSTM RNN architecture developed by Alex Graves for handwriting generation, which consists of a window layer, 2 hidden layers, and a mixture density output layer.
  - We removed the window layer, i.e. the attention mechanism used to determine which parts of the string of text were most relevant to the handwritten points being generated.
  - We also changed the initial activation from a zero vector to the vector encodings described above.



Future Work

- Using a different architecture, perhaps one involving CNNs to capture image information, could be interesting to try out.
- Using a more sophisticated encoding mechanism for our input (object class + first 50 points of drawing)
- Expanding the number of object classes we train on and generate. Cars are somewhat difficult to draw, we could try something easier.

Results



This graph shows an attempt at generating a car drawing, given the first 50 points of the test image. We tested on 100 test images and calculated the mean squared error between the generated image and the true test image.

➔ Average error across all 100 test images = **17.68**.

Discussion

- Overall, the neural network was unable to complete the drawings accurately.
- Part of the reason may be that we chose a difficult object class, as drawing a car involves a fair amount of complexity.

References

Alex Graves Paper: https://arxiv.org/pdf/1308.0850.pdf
Handwriting generation implementation: https://github.com/Grzego/handwriting-generation
Quick Draw dataset: https://github.com/googlecreativelab/quickdraw-dataset