Handwriting Sequence Generation from ASCII

Jason Chen

Abstract

This project explores the ability of recurrent neural networks to generate realistic appearing handwriting from ASCII text. We replicate the results of the Graves paper on synthesizing handwriting sequences from ASCII. The key components of Graves' model include using deep long short term memory (LSTM) layers to capture long range dependencies in the points of handwritten text, a mixture density network (MDN) to capture the multimodal distribution of handwriting points, and an attention mechanism to focus on generating the output for a small window of one or two letters at a time. We are able to successfully reproduce the Graves' results with model-generated outputs indistinguishable from human-generated handwriting when presented to human judges.

Previous Work

There is some existing work on deep neural network based speech synthesis done by Zen and Senior. They cite limitations in existing approaches due to unimodal objective functions leading to a lack of ability to predict variances in their outputs. Their solution uses a mixture density output layer to improve the naturalness of the synthesized speech. The paper is a good parallel to evaluating and corroborating Graves' choice to use an MDN in the handwriting generation component.

Stanford NLP researchers Luong, Pham, and Manning published a paper exploring attention-based neural machine translation. The team has success with both a global and local attention mechanism that looks at either all or a subset of source words. To compare this to the Graves paper, the challenge is similar in that the length of the input and output sequences are very different, and Graves implements a differentiable soft window convolved with the ASCII input to learn an attentive alignment between the characters and the "pen" locations.

Lastly, we found two companies Bond and Handwrytten that offer services to generate custom notes and letters in personalized handwritten styles. They actually modified robotic printer setups to move a pen across paper. It is unclear whether their proprietary processes for learning each client's handwriting styles involve manual labor by calligraphers parsing sample writing or if they involve some machine learned algorithm as we show in this paper.

Dataset

The dataset used is the public IAM On-line Handwriting Database (IAM-OnDB). The segment of the database we are interested in contains samples of handwriting from 221 writers contributing 13,049 lines and 86,272 words from a dictionary size of 11,059. For this project, the raw data is parsed into a three dimensional time series where each point in the series contained point tuples of shape (xCoordinate, yCoordinate, penLifted). In this dataset, characters and lines consist of approximately 25 and 700 points on average, respectively.

We take the lines and break them down into a 90/10 train/dev split, or 11744 training and 1305 dev samples respectively. This would traditionally be bad practice due to the high possibility of overfitting on the training set. However, for the goal of generating realistic looking handwriting, overfitting is not a large issue.

turn down the Fact - Griffiths resolution

Figure 1: Handwriting sample

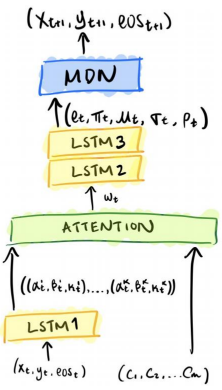
```
<StrokeSet>
  <Stroke colour="black" start_time="14719118.35" end_time="14719118.89">
    <Point x="418" y="3817" time="14719118.35"/>
    <Point x="418" y="3822" time="14719118.37"/>
    <Point x="419" y="3825" time="14719118.39"/>
    <Point x="417" y="3837" time="14719118.48"/>
```

Figure 2: XML raw data format

Model

3 layer LSTM Network

The backbone of this model is the LSTM network. We use a 3-layered LSTM network with a hidden state size of 400 each. LSTMs are a natural fit for this handwriting synthesis problem due to its affinity for sequential data and capturing long range dependencies. First, the points in handwriting are sequential data where previous points strongly influence where the next points should be. Furthermore, each character can be a long sequence of twenty five to forty points. Lastly, handwriting contains delayed strokes like crossing t's, which perform worse without LSTMs extended memory cells.



Mixed Density Network and Loss Fn

MDNs augment classical neural networks with a mixture density model and have shown to be useful when modeling multimodal distributions. The mixture density model consists of a collection of distributions, and the final target probability is calculated by taking a weighted combination of each distribution.

For our MDN, we use M mixture components consisting of bivariate Gaussian distributions for the x and y offsets and a Bernoulli distribution for the end-of-stroke indicator. The Gaussian distributions are defined by their means, standard deviations, correlation, and mixture weight. There are 7M MDN parameters. We learn these parameters by setting them to be outputs from the final network layer.

$$\hat{y}_t = (\hat{e}_t, \{\hat{\pi}_t^j, \hat{\mu}_t^j, \hat{\sigma}_t^j, \hat{\rho}_t^j\}_{j=1}^M) = b_y + \sum_{n=1}^{N} W_{h^n y} h_t^n + b_y$$

$$e_t = \frac{1}{1+\exp(\hat{e}_t)} \qquad \pi_t^j = \exp(\hat{\beta}_t) \qquad \mu_t^j = \hat{\mu}_t^j \qquad \sigma_t^j = \exp(\hat{\sigma}_t^j) \qquad \rho_t^j = tanh(\hat{\rho}_t^j)$$

$$\mathcal{L}(\mathbf{x}) = -log\Big(\sum_j \pi_t^j \mathcal{N}(x_{t+1} \mid \mu_t^j, \sigma_t^j, \rho_t^j)\Big) - \begin{cases} log(e_t) & \text{for } (x_{t+1})_3 = 1 \\ log(1-e_t) & \text{otherwise} \end{cases}$$

Attention Mechanism

In this project, we build the attention layer to generate a sliding window over the letters being translated to handwriting. As longer lines of ASCII text can require up to 700 handwritten output points, giving the model an idea of several characters to focus on can greatly improve the quality of the output.

$$\phi(t,u) = \sum_{k=1}^{K} \alpha_t^k \exp(-\beta_t^k(\kappa_t^k - u)^2) \qquad w_t = \sum_{u=1}^{U} \phi(t,u) c_u$$

w is a U-length vector representing the attention weight of each character. phi(t,u) gives the window weight of one-hot vector c at time t. Alpha weights the influence of the k-th distribution, beta controls the width of the window, and kappa directs the location of the window within the sequence. We learn these 3K parameters by setting them to be outputs from the first hidden layer of the network.

$$(\hat{\alpha}_t, \hat{\beta}_t, \hat{\kappa}_t) = W_{h^1 att} h_t^1 + b_{att}$$

$$\alpha_t = \exp(\hat{\alpha}_t) \qquad \beta_t = \exp(\hat{\beta}_t) \qquad \kappa_t = \kappa_{t-1} + \exp(\hat{\kappa}_t)$$

Results / Discussion

Handwriting Sampling and Synthesis

After training the model, we can synthesize new handwriting from ASCII by initializing a tuple and feeding it once through the network. Then, we feed the previous output as the input for the next iteration. We define a stopping heuristic that checks whether the attention mechanism's window weight thinks the network is past the final character. Note that since this generative problem has no test metric, the training loss is relatively inconsequential.

After training the model we can synthesize new handwriting from ASCII by initializing an x_1, y_1, eos sample and feeding it once through the network. Then we feed the previous output as the input for the next iteration. We define a stopping heuristic that checks whether the

Primed Sampling

Priming the model to mimic a specific style is also possible. To do so, we start off with the stylized T-length input sequence x mapping to its ASCII sequence c. For the first T steps, we feed the network x_t instead of the output from the previous timestep, effectively "clamping" the model to a specific style. Afterwards, the network will proceed as usual, feeding as inputs the outputs from the previous timestep.

Priming the model to mimic a specific style is also possible.

To do so, we start off with the stylized T length input sequence

I do not know of I do not know ← This line is the original style

Biasing MDNs

We can bias the model towards better handwriting by providing a bias term that reduces the standard deviations in the MDN Gaussian distributions. We are effectively giving preference to "better" handwriting closer to the mean of each distribution.

We can bias the model towards better handwriting by providing a bias term that reduces the standard deviations in the MDN Gaussian distributions. We are effectively giving preference to better handwriting closer to the mean.

Conclusion and Future Work

We are fairly satisfied with being able to replicate the Graves paper with such success. With a some MDN bias, the output is very clean and legible. At the same time, the style priming worked well and was able to capture distinctive elements of each style. One of biggest possible improvements for this project would come from increasing the size of the dataset. Today, we could easily create an app to record new handwriting samples on tablets. Another direction for future work would be applying the model towards speech synthesis. The problem domain is very similar to handwriting synthesis, but it would be potentially much more challenging due to the higher dimensionality of speech and audio data when compared to the points of handwriting data.