Handwriting Synthesis using Long Short-Term Memory

Manogna Mantripragada, Ninad Subhedar

Northeastern University, Boston, MA, USA

Abstract. This paper explores the approach for synthesizing handwriting sequences using Long Short-Term Memory Neural Networks. The work is based on a paper from 2014 on Generating Sequences from a Recurrent Neural Networks by Alex Graves. The neural network model will be trained using the "IAM On-Line Handwriting Database" which consists of XML files containing pen point co-ordinates and labeled text. These coordinates and labels are extracted and processed for feeding the model. The Neural Network is implemented with a variant of Recurrent Neural network called LSTM which work well with sequential data. The paper also explores Mixture Density Networks and Soft window network that aid in better training the LSTMs.

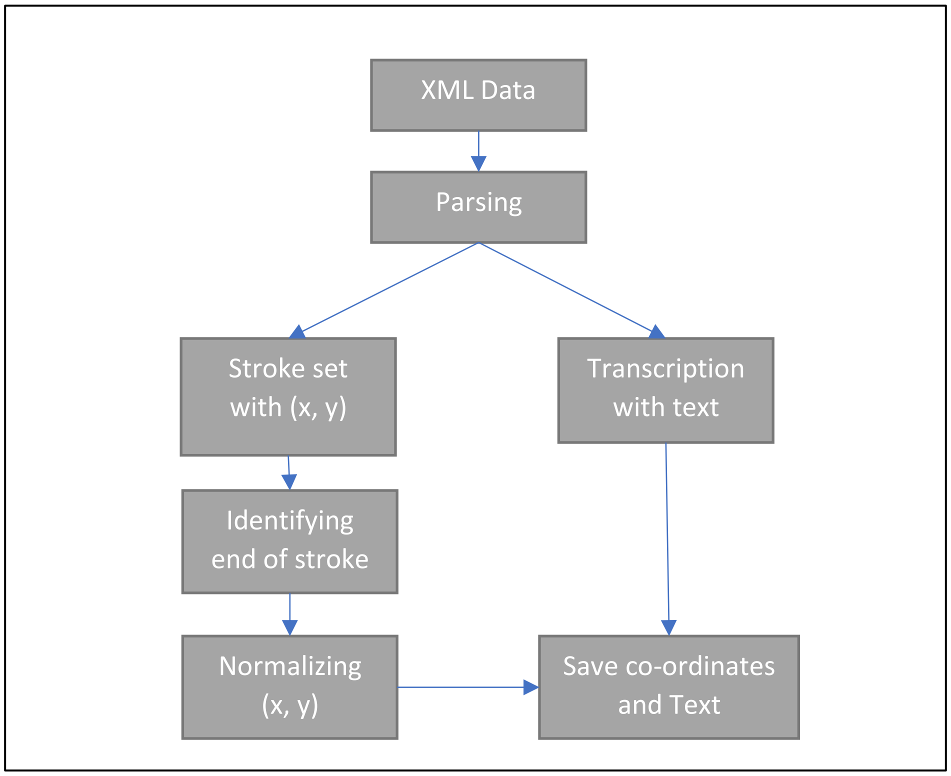
1 Introduction

Any human hand-written text is basically made up of continuous sequences of pen points. Every handwriting has its own styles and features. A few features like spacing between each character, the slant of the strokes define handwriting. Most neural networks when trained properly tend to model complex features or patterns in a dataset. In the case of handwritings, Neural networks are a good option. But considering the nature of the data i.e. being continuous and sequential, there is a need for a neural network that captures the previous data to predict the next data like Recurrent Neural networks. RNNs take their own predicted output as input to predict the next data point. This model is suitable for modeling and generating handwriting sequences as the nature of the next stroke depends on the current stroke.

The data used for training is taken from the IAM online handwriting database which contains the handwritten English text written on the whiteboard. The data contains forms of unconstrained handwritten text, acquired with the E-Beam System. The collected data is stored in XML-format, including the writer-id, the transcription and the setting of the recording. The data has multiple 'transcription' and 'strokes' tags. The Transcription tags contain the text and the Strokes tag contains the coordinates of each point in a single stroke.

2 Approach

The objective of the Neural network is to learn to predict the sequence of points from the existing handwriting data. To start with this the coordinates and the corresponding text was extracted from the handwriting database. This data was then cleaned and normalized. Below is a high-level flow for preprocessing the data.



2.1 Pre-processing pseudocode:

```
Step 1: Parsing the XML files
Step 2: Extracting the text and the coordinates from the XML file.
Step 3: Identify the end of the stroke by observing the end of stroke value. If It is 1, then the pen is lifted (end of stroke) and if it's 0 it is the begin of stroke.
Step 4: Normalizing the co-ordinates
Step 5: Save the coordinates
```

2.2 Soft Window Layer

The processed data is feed into a soft window to guide the neural network by representing a relation between the handwriting strokes and the text characters. The relation is defined by the below equation:

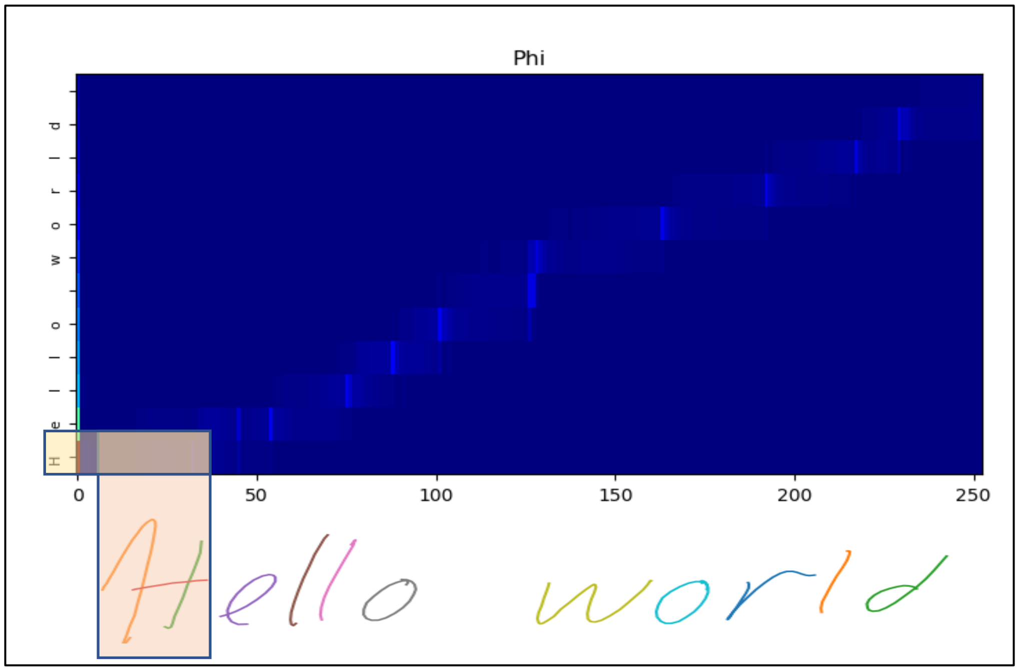
$$window = \sum_{u=1}^{U}\left(\sum_{k=1}^{K}\alpha_t^k e^{\left(-\beta_t^k\left(\kappa_t^k-u\right)^2\right)}\right)c_u$$

$$\alpha_t = e^{\widehat{\alpha_t}} \rightarrow Importance\ of\ the\ window$$

$$\beta_t = e^{\widehat{\beta_t}} \rightarrow Width\ Of\ the\ window$$

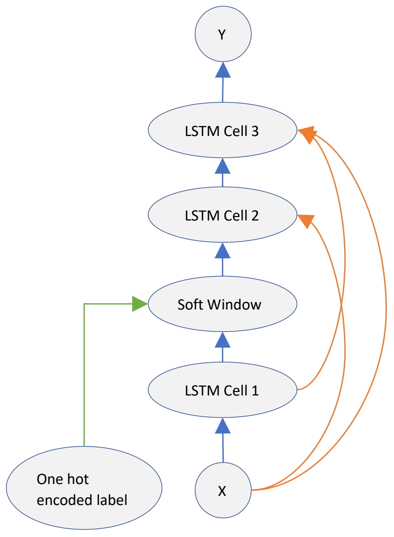
$$\kappa_t = \kappa_{t-1} - e^{\widehat{\kappa_t}} \rightarrow offset\ for\ the\ window$$

Below figure illustrates how the window co-relates the text to the set of strokes. The α, β, κ values are forwarded to the next LSTM layer.

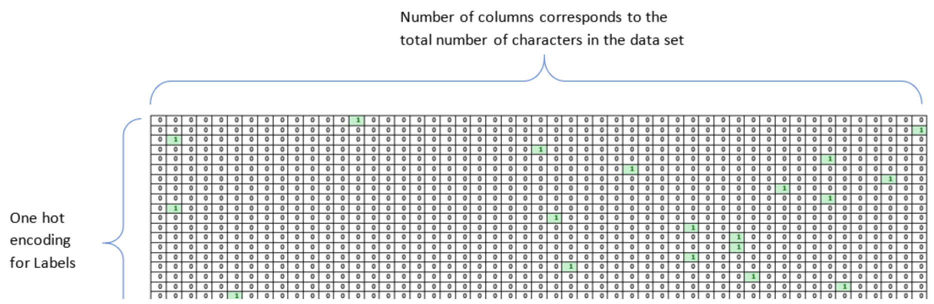


2.3 LSTM Layer

The main LSTM layer was designed using the standard LSTM Cell in TensorFlow library. Three Hidden LSTM Cell each consisting of 400 Cell were stacked to form the LSTM layer. The input to these layers comes from the coordinates of the penpoint and its corresponding labels along with the output of the soft window. Every layer has skip connections from the inputs to all hidden layers and from all hidden layers to the outputs. This makes it easier to train the networks by reducing the number of processing. This also helps in handling the `vanishing gradient' problem. The below figure shows the high-level design of the LSTM layer. The connections marked in orange are the skip connections. And the connection marked green is the input from the one hot encoded text.



The text data had to be one hot encoded into a matrix such that each sentence has its own matrix and the rows signify a single character while the column indicates which character is to be selected. The following figure shows a representation of the one hot encoded matrix.


Number of columns corresponds to the total number of characters in the data set
One hot encoding for Labels

The green boxes in the above figure are the places in the matrix where a character is present in the sentence and is indicated by 1.

2.4 Mixture Density Layer

The output from the LSTM layer is forwarded to a Mixture Density Layer which converts the output to a mixture of distributions which is used to create a probability distribution that represents the coordinates for the pen point and the end strokes. The mixture consists of a bivariate Gaussian which can be used to predict the coordinates and a Bernoulli distribution used to predict the end stroke probability. This layer generates the following elements which are used to generate the above-mentioned distributions.

$$e_t = \frac{1}{(1 + \exp(\hat{e}_t))} \quad \rightarrow \quad End\ stroke\ probablity$$

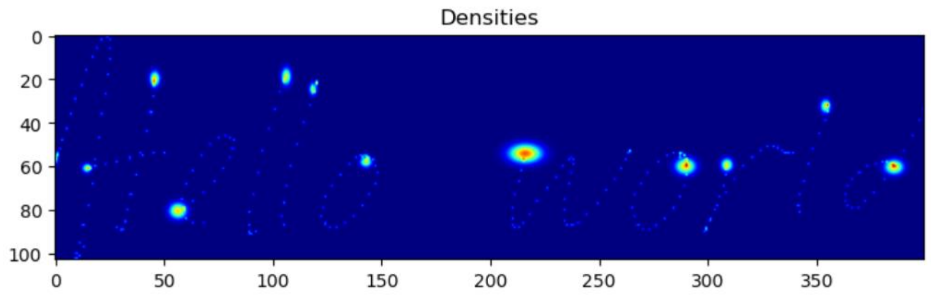
$$\pi_t = \frac{\exp(\hat{\pi}_t)}{\sum_{j=1}^{M} \exp(\hat{\pi}_t)} \quad \rightarrow \quad Mixute\ weights$$

$$\mu_t = \hat{\mu}_t \quad \rightarrow \quad Set\ of\ means$$

$$\sigma_t = exp(\hat{\sigma}_t) \quad \rightarrow \quad Set\ of\ Standard\ deviations$$

$$\rho_t = tanh(\rho_t) \quad \rightarrow \quad Co - relations\ between\ the\ co - ordinates$$

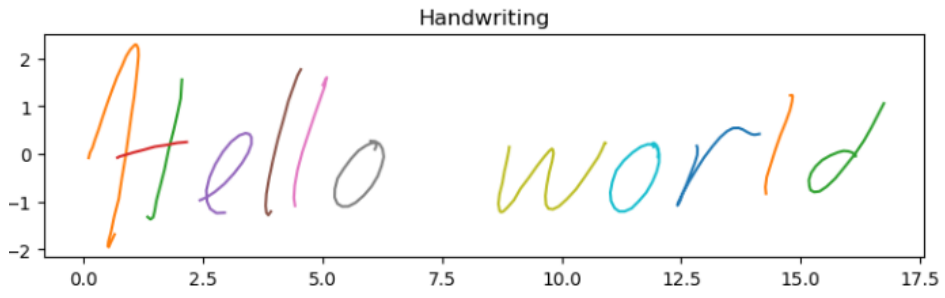
These variables are used to generate probability distributions and then pull random samples from the distributions to get the predicted co-ordinates and the predicted end strokes. The following figure shows the mixture distributions for each coordinate.



Each point in the above plot represents the mixture density for each coordinate.

2.5 Handwriting generation

The output of the Mixture density layer can be sampled, and the coordinates can be generated along with the end of strokes for the text. Each co-ordinate can be plotted on a graph to represent a stroke. To generate the coordinates the trained model is feed with one hot encoded text and then the model outputs the probability densities from which random samples can be taken to generate a set of coordinates. Below is an example of the coordinates generated from the model and then plotted onto a graph. The colors in the graph represent individual strokes.



3 Prediction and Loss function

The probability density function was defined as follows:

$$\Pr(x_{t+1}|y_t) = \sum_{j=1}^{M} \pi_t^j \ \mathcal{N}(x_{t+1}|\mu_t^j, \sigma_t^j, \rho_t^j) \begin{cases} e_t & \text{if } (x_{t+1})_3 = 1 \\ 1 - e_t & \text{otherwise} \end{cases}$$

where

$$\mathcal{N}(x|\mu, \sigma, \rho) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} \exp\left[\frac{-Z}{2(1-\rho^2)}\right]$$

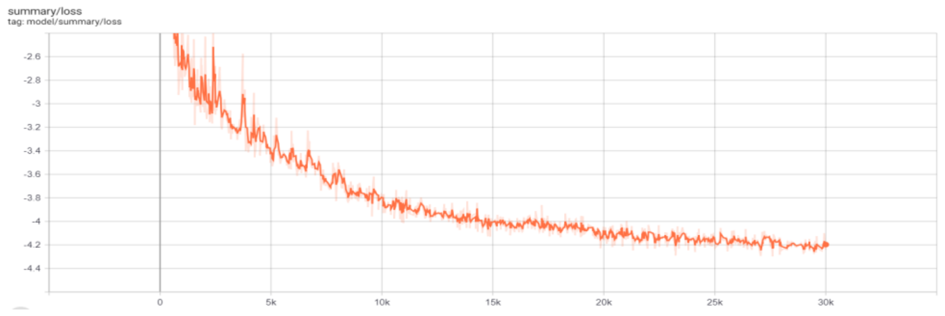
with

$$Z = \frac{(x_1-\mu_1)^2}{\sigma_1^2} + \frac{(x_2-\mu_2)^2}{\sigma_2^2} - \frac{2\rho(x_1-\mu_1)(x_2-\mu_2)}{\sigma_1\sigma_2}$$

All the components derived from the Mixture density layer are used to calculate the probability density function. And the loss function is a negative log loss defined as below

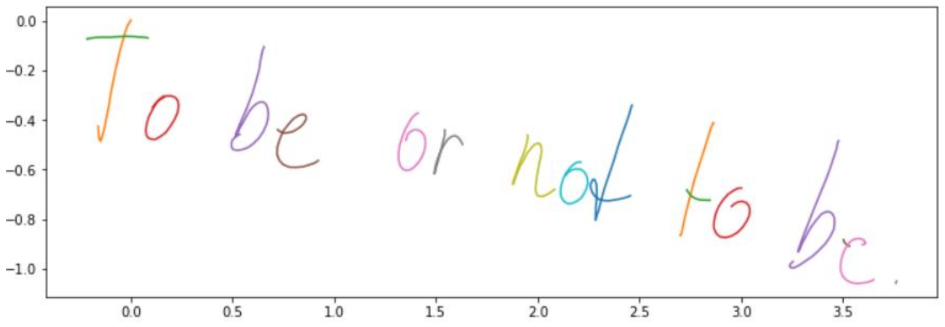
$$\mathcal{L}(\text{x}) = \sum_{t=1}^{T} -\log\left(\sum_j \pi_t^j \mathcal{N}(x_{t+1}|\mu_t^j, \sigma_t^j, \rho_t^j)\right) - \begin{cases} \log e_t & \text{if } (x_{t+1})_3 = 1 \\ \log(1-e_t) & \text{otherwise} \end{cases}$$

This function was used for optimizing the learning process for the neural network. The optimizer used for training was Adam Optimizer and the learning rate was set to 0.001. The complete model was trained for 30 epochs and each epoch had 1000 batches. Below graph shows the loss values for each step. The loss decreased from -2.431 to -4.197



4 Observations & Conclusion

The model learns from the handwriting data provided and does generate handwriting co-ordinates with individual strokes. For long sequences of text, the model does not tend to maintain the angle of handwriting. For example, in the below figure the text starts to bend downwards.



This could possibly be due to a smaller number of long sentences and due to this model is not trained to learn long continuous handwriting sequences. Furthermore, the model outputs random styles of handwriting while generation. This could be solved by priming the model just before generation which is not covered in this scope. The model works well with block letters as compared to cursive handwriting.

5 Contribution

The 60% of the design and implementation was derived from the original paper Generating Sequences from a Recurrent Neural Networks by Alex Graves [1] and handwriting generation with the use of recurrent neural networks in TensorFlow [7]. Remaining 40 % includes modification to the code to write individual layers and introducing a 3 layered LSTM with an Adam Optimizer for training the neural network.

6 Citations

[1] Generating Sequences with Recurrent Neural Networks (https://arxiv.org/abs/1308.0850)

[2] Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM) (https://youtu.be/WCUNPb-5EYI)

[3] Long short-term memory (https://en.wikipedia.org/wiki/Long_short-term_memory)

[4] IAM On-Line Handwriting Database (http://www.fki.inf.unibe.ch/databases/iam-on-line-handwriting-database)

[5] MIT 6. S191: Recurrent Neural Networks (https://youtu.be/_h66BW-xNgk?list=PLtBw6njQRU-rwp5__7C0oIVt26ZgjG9NI)

[6] Scribe: realistic handwriting with TensorFlow (https://greydanus.github.io/2016/08/21/handwriting/)

[7] handwriting generation with the use of recurrent neural networks in TensorFlow (https://github.com/Grzego/handwriting-generation)

[8] Deep Learning Lecture 13: Alex Graves on Hallucination with RNNs (https://www.youtube.com/watch?v=-yX1SYeDHbg&t=2217s)

[9] The magic of LSTM Neural networks

[10] (https://towardsdatascience.com/learn-how-recurrent-neural-networks-work-84e975feaaf7)

[11] How Recurrent Networks Work

[12] (https://towardsdatascience.com/learn-how-recurrent-neural-networks-work-84e975feaaf7)

License