HANDWRITNG GENERATION

- PRANAVA RAMAN BMS - 2019103555
- PREETI KRISHNAVENI - 2019103560
- ANUSREE V - 2019103507

Introduction

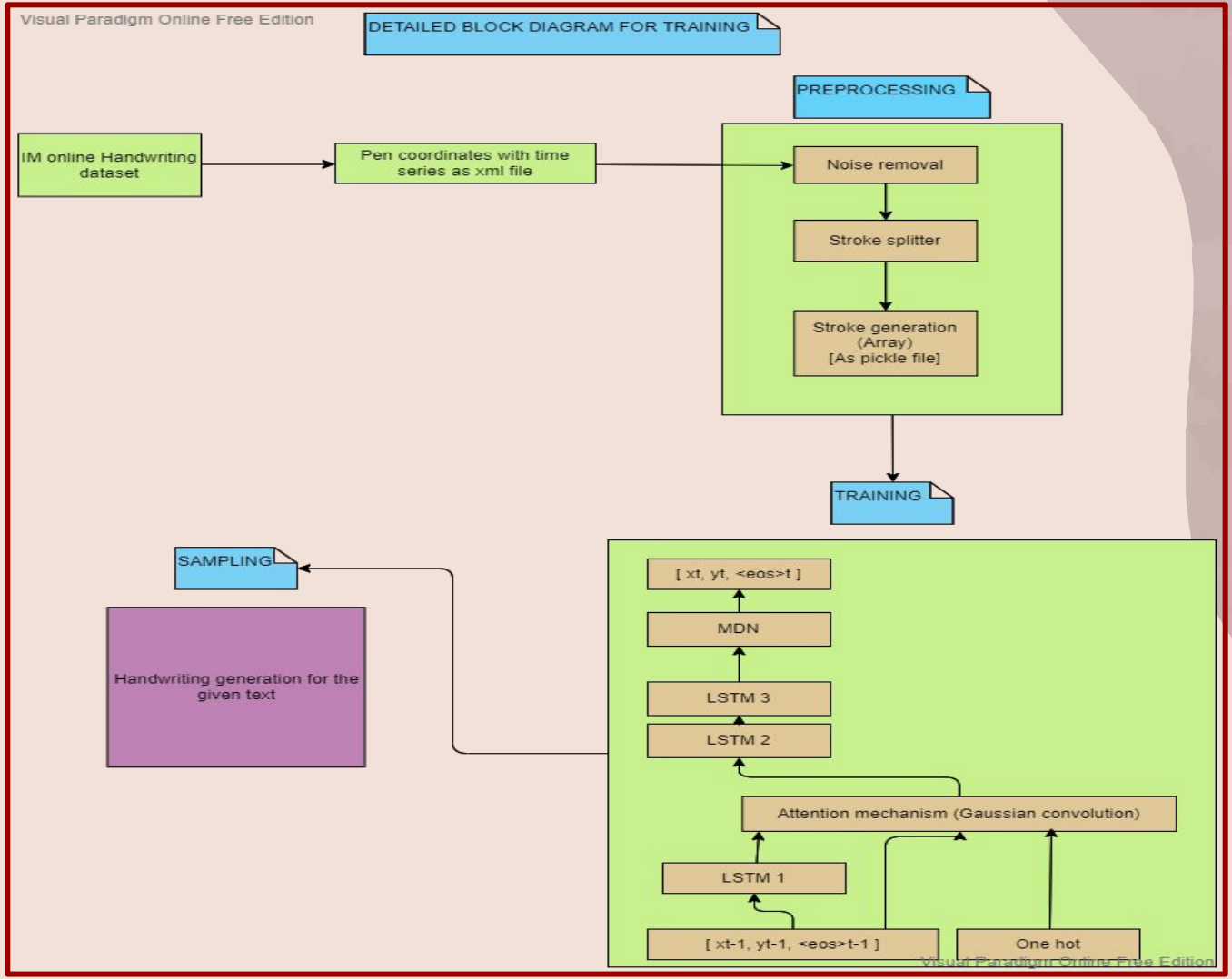
Recurrent neural networks (RNNs) are a rich class of dynamic models that have been used to generate sequences in domains as diverse as music, text and motion capture data. RNNs can be trained for sequence generation by processing real data sequences one step at a time and predicting what comes next. Assuming the predictions are probabilistic, novel sequences can be generated from a trained network by iteratively sampling from the network's output distribution, then feeding in the sample as input at the next step. In this project, we are using RNNs to generate handwriting from different styles of handwriting data and a given input.

DATASET

We used IAM Handwriting Database to train the model. As far as datasets go, Although it's very small in size (less than 50 MB once parsed), preprocessing it generates a huge dataset. A total of 657 writers contributed to the dataset and each has a unique handwriting style.

The data itself is a three-dimensional time series. The first two dimensions are the (x, y) coordinates of the pen tip and the third is a binary 0/1 value where 1 signifies the end of a stroke. Each line has around 500 pen points and an annotation of ascii characters.

BLOCK DIAGRAM



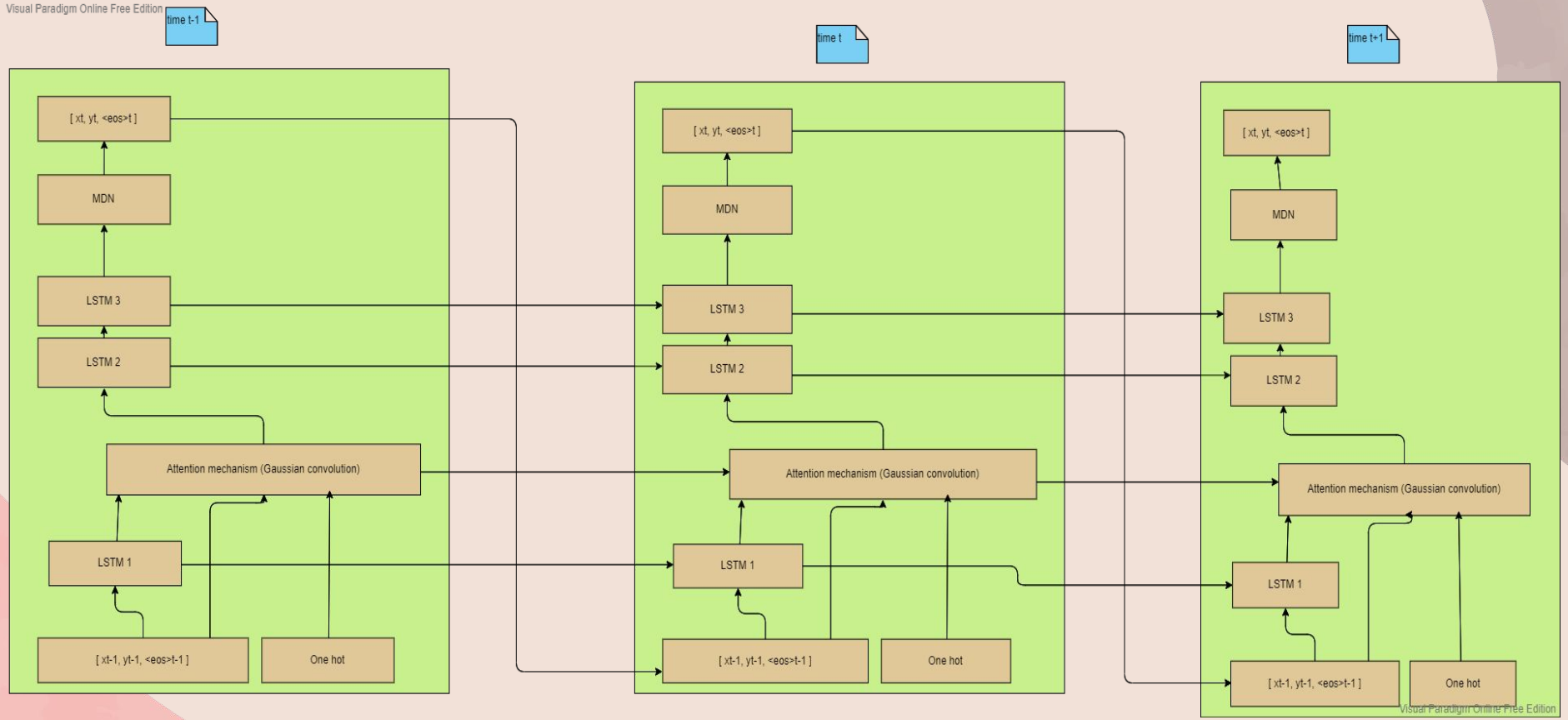
Completed

To be done

Module - 1
Preprocessing (Data loader)

Here the time series data is converted into strokes data with corresponding ascii input labels. We first take the inputs and find the distance between the adjacent pen co ordinates. We remove noises at this part by removing points that are too distant. Then the strokes are converted into arrays and along with their labels, they are given output as pickle files. This is used as input for training purposes. Multiple styles of writing are saved as different pickle files.

Module - 2
Training



Long Short-term Memory (LSTM) Cell

The Long Short-Term Memory (LSTM) Cell. At the core of the Graves handwriting model are three Long Short-Term Memory (LSTM) Recurrent Neural Networks (RNNs). We could just as easily have used Gated Recurrent Units (GRUs), Recurrent Highway Networks (RHNs), or some other seq2seq cell. TensorFlow provides a built-in API for these models so it doesn't really matter. If you don't know what recurrent neural networks or LSTMs are, read this post to see how they work and this post to see what they can do.

These networks use a differentiable form of memory to keep track of time-dependent patterns in data. LSTMs, for example, use three different tensors to perform 'erase', 'write', and 'read' operations on a 'memory' tensor: the f, i, o and C tensors respectively. RNNs are extremely good at modeling sequential data.

Mixed Density Network (MDN)

Think of Mixture Density Networks as neural networks which can measure their own uncertainty. Their output parameters are μ, σ, and ρ for several multivariate Gaussian components. They also estimate a parameter for each of these distributions. Think of π as the probability that the output value was drawn from that particular component's distribution.

Since MDNs parameterize probability distributions, they are a great way to capture randomness in the data. In the handwriting model, the MDN learns to how messy or unpredictable to make different parts of handwriting. For example, the MDN will choose Gaussian with diffuse shapes at the beginning of strokes and Gaussians with peaky shapes in the middle of strokes.

The Attention Mechanism

Imagine that we want our model to write ‘Hello world.’ In order to get the information about which characters make up this sentence, the model uses a differentiable attention mechanism. In technical terms, it is a Gaussian convolution over a one-hot ascii encoding. Think of this convolution operation as a soft window through which the handwriting model can look at a small subset of characters, ie. the letters ‘wo’ in the word world. Since all the parameters of this window are differentiable, the model learns to shift the window from character to character as it writes them

The model learns to control the window parameters remarkably well. For example, the bright stripes in the first plot are the model’s way of encoding the end of a pen stroke. We never hard-coded this behavior!

OUTPUT - PREPROCESSING

linestrokes-XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<WhiteboardCaptureSession>
  <WhiteboardDescription>
    <SensorLocation corner="top_left"/>
    <DiagonallyOppositeCoords x="6512" y="1376"/>
    <VerticallyOppositeCoords x="966" y="1376"/>
    <HorizontallyOppositeCoords x="6512" y="787"/>
  </WhiteboardDescription>
  <StrokeSet>
    <Stroke colour="black" start_time="769.05" end_time="769.64">
      <Point x="1073" y="1058" time="769.05"/>
      <Point x="1072" y="1085" time="769.07"/>
      <Point x="1066" y="1117" time="769.08"/>
      <Point x="1052" y="1152" time="769.10"/>
      <Point x="1030" y="1196" time="769.12"/>
      <Point x="1009" y="1242" time="769.13"/>
      <Point x="994" y="1286" time="769.14"/>
      <Point x="980" y="1317" time="769.16"/>
      <Point x="971" y="1336" time="769.18"/>
      <Point x="968" y="1344" time="769.19"/>
      <Point x="966" y="1339" time="769.20"/>
      <Point x="972" y="1340" time="769.22"/>
      <Point x="978" y="1320" time="769.24"/>
      <Point x="991" y="1298" time="769.25"/>
      <Point x="1003" y="1266" time="769.27"/>
      <Point x="1016" y="1231" time="769.28"/>
      <Point x="1021" y="1184" time="769.30"/>
      <Point x="1030" y="1143" time="769.31"/>
      <Point x="1040" y="1108" time="769.33"/>
      <Point x="1049" y="1077" time="769.34"/>
      <Point x="1055" y="1049" time="769.36"/>
      <Point x="1058" y="1021" time="769.37"/>
      <Point x="1064" y="1006" time="769.38"/>
      <Point x="1071" y="1006" time="769.40"/>
      <Point x="1071" y="1006" time="769.42"/>
      <Point x="1074" y="1013" time="769.43"/>
      <Point x="1083" y="1042" time="769.45"/>
      <Point x="1097" y="1082" time="769.46"/>
      <Point x="1114" y="1124" time="769.48"/>
```

ascii-XML

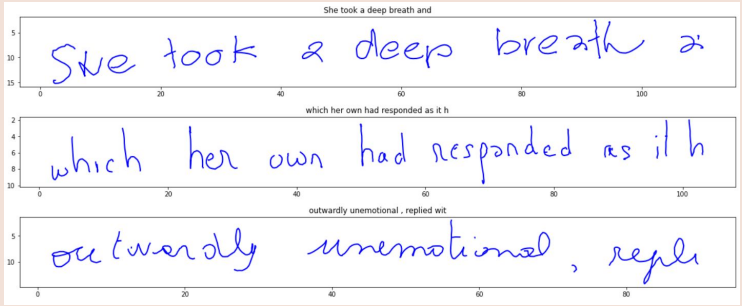
```
OCR:

A MOVE to stop Mr . Gaitskell from nominating
any more Labour life Peers is to be made at a
meeting of Labour 0M Ps tomorrow . Mr . Michael
Foot has put down a resolution on the subject
and he is to be backed by Mr . Will Griffiths ,

CSR:

A MOVE to stop Mr . Gaitskell from nominating
any more Labour life Peers is to be made at a
meeting of Labour 0M Ps tomorrow . Mr . Michael
Foot has put down a resolution on the subject
and he is to be backed by Mr . Will Griffiths
```

Output after preprocessing - visualization



OUTPUT - TRAINING

```
TRAINING MODE...
<__main__.Args object at 0x7f1ad473c190>

loading data...
        loaded dataset:
                11262 train individual data points
                592 valid individual data points
                351 batches
building model...
        using alphabet abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
/tensorflow-1.15.2/python3.7/tensorflow_core/python/client/session.py:1750: UserWarning: An interactive session is already active. This can cause out-of-memory
  warnings.warn('An interactive session is already active. This can '
attempt to load saved model...
no saved model to load. starting new session
training...
learning rate: 9.999999747378752e-05
0/125000, loss = 3.090, regloss = 0.03090, valid_loss = 3.061, time = 46.975
10/125000, loss = 3.106, regloss = 0.31705, valid_loss = 3.056, time = 0.446
20/125000, loss = 3.039, regloss = 0.57752, valid_loss = 3.046, time = 0.454
30/125000, loss = 2.912, regloss = 0.81331, valid_loss = 3.033, time = 0.446
40/125000, loss = 3.039, regloss = 1.02522, valid_loss = 3.014, time = 0.823
50/125000, loss = 2.982, regloss = 1.21226, valid_loss = 2.990, time = 0.444
60/125000, loss = 3.012, regloss = 1.37867, valid_loss = 2.960, time = 0.446
70/125000, loss = 2.879, regloss = 1.52687, valid_loss = 2.917, time = 0.447
80/125000, loss = 2.799, regloss = 1.65591, valid_loss = 2.855, time = 0.459
90/125000, loss = 2.761, regloss = 1.76637, valid_loss = 2.763, time = 0.446
100/125000, loss = 2.564, regloss = 1.85541, valid_loss = 2.612, time = 0.449
110/125000, loss = 2.413, regloss = 1.91864, valid_loss = 2.374, time = 0.445
120/125000, loss = 2.009, regloss = 1.94675, valid_loss = 2.033, time = 0.448
130/125000, loss = 1.797, regloss = 1.93835, valid_loss = 1.641, time = 0.451
140/125000, loss = 1.370, regloss = 1.89288, valid_loss = 1.328, time = 0.450
150/125000, loss = 1.228, regloss = 1.83421, valid_loss = 1.215, time = 0.449
```

REFERENCES

- Generating Sequences With Recurrent Neural Networks: Alex Graves [1308.0850.pdf (arxiv.org)]
- Realistic Handwriting Generation Using Recurrent Neural Networks and Long Short-Term Networks | SpringerLink
- Grzego/handwriting-generation: Implementation of handwriting generation with use of recurrent neural networks in tensorflow. Based on Alex Graves paper (https://arxiv.org/abs/1308.0850). (github.com)
- greydanus/scribe: Realistic Handwriting with Tensorflow (github.com)