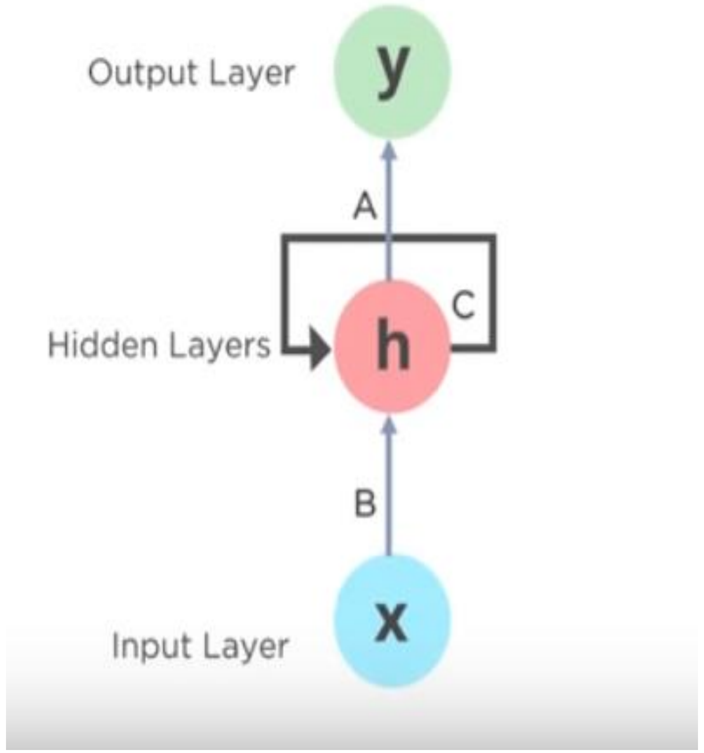
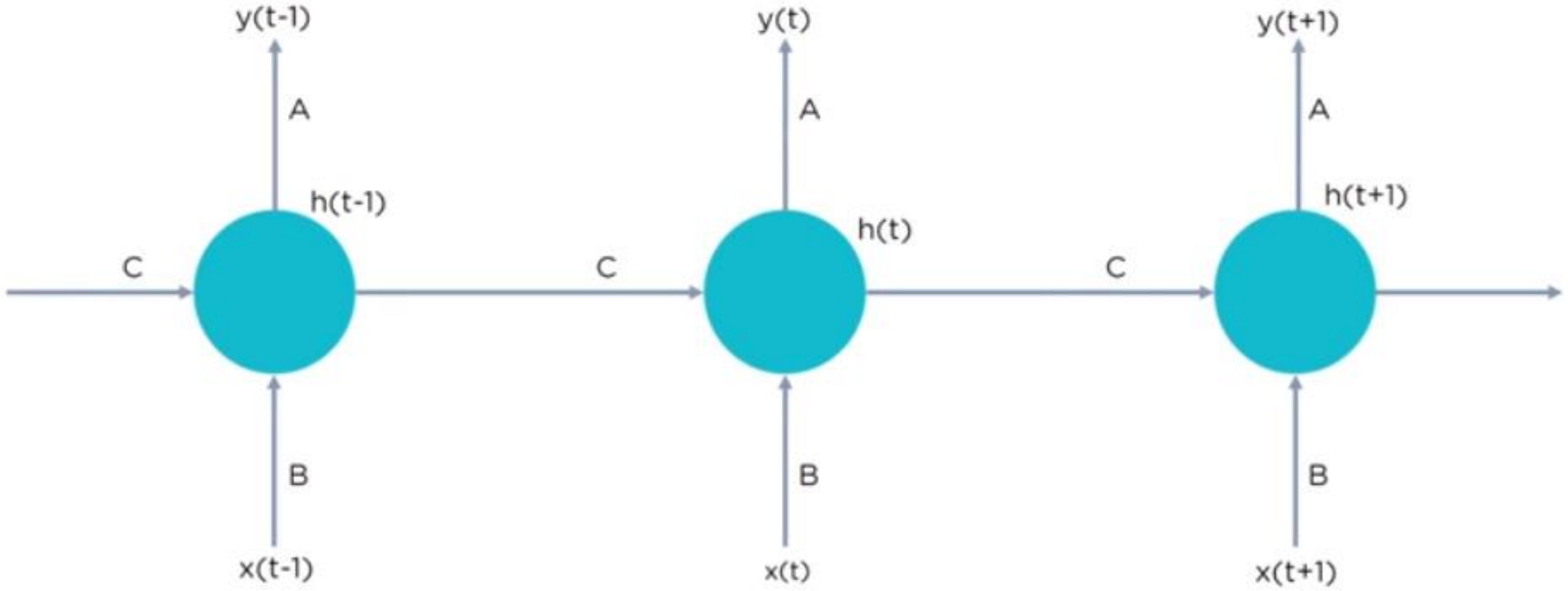
Generating Sequences with Recurrent NN

Presenting a paper by Alex Graves

Recurrent Neural Networks

- Sequential data- Predict what’s next
- Internal(short)Memory
- The predicted value depends on the previous inputs
- Fuzzy
- No curse of dimensionality





$h(t) = f_c\,(h(t-1), x(t))$

$h(t)$ = new state
f_c = function with parameter c
$h(t-1)$ = old state
$x(t)$ = input vector at time step t

Issues:

- Memory handling
- Instability while generating sequences

Solution:

- Add noise to data
- Better Memory- Model such as Long Short Term Memory

Prediction Network

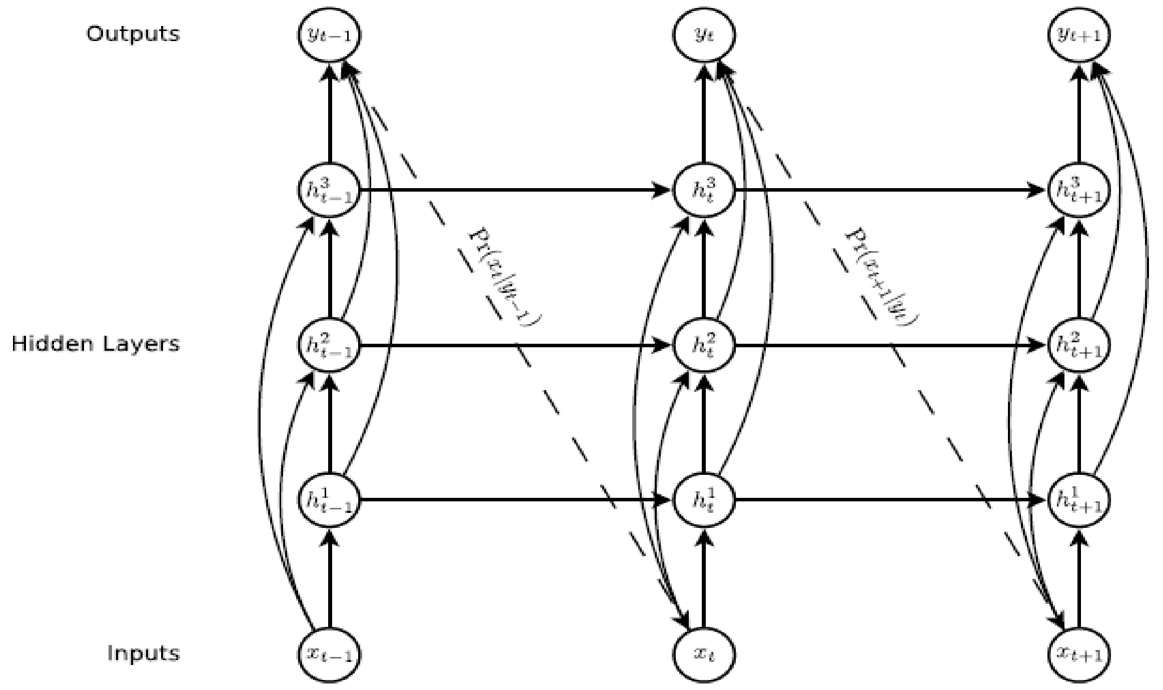


Figure 1: **Deep recurrent neural network prediction architecture.** The circles represent network layers, the solid lines represent weighted connections and the dashed lines represent predictions.

Calculations

Hidden layer activations:

$$h_t^1 = \mathcal{H}\left(W_{ih^1} x_t + W_{h^1 h^1} h_{t-1}^1 + b_h^1\right)$$
$$h_t^n = \mathcal{H}\left(W_{ih^n} x_t + W_{h^{n-1} h^n} h_t^{n-1} + W_{h^n h^n} h_{t-1}^n + b_h^n\right)$$

Output sequence:

$$\hat{y}_t = b_y + \sum_{n=1}^{N} W_{h^n y} h_t^n$$
$$y_t = \mathcal{Y}(\hat{y}_t)$$

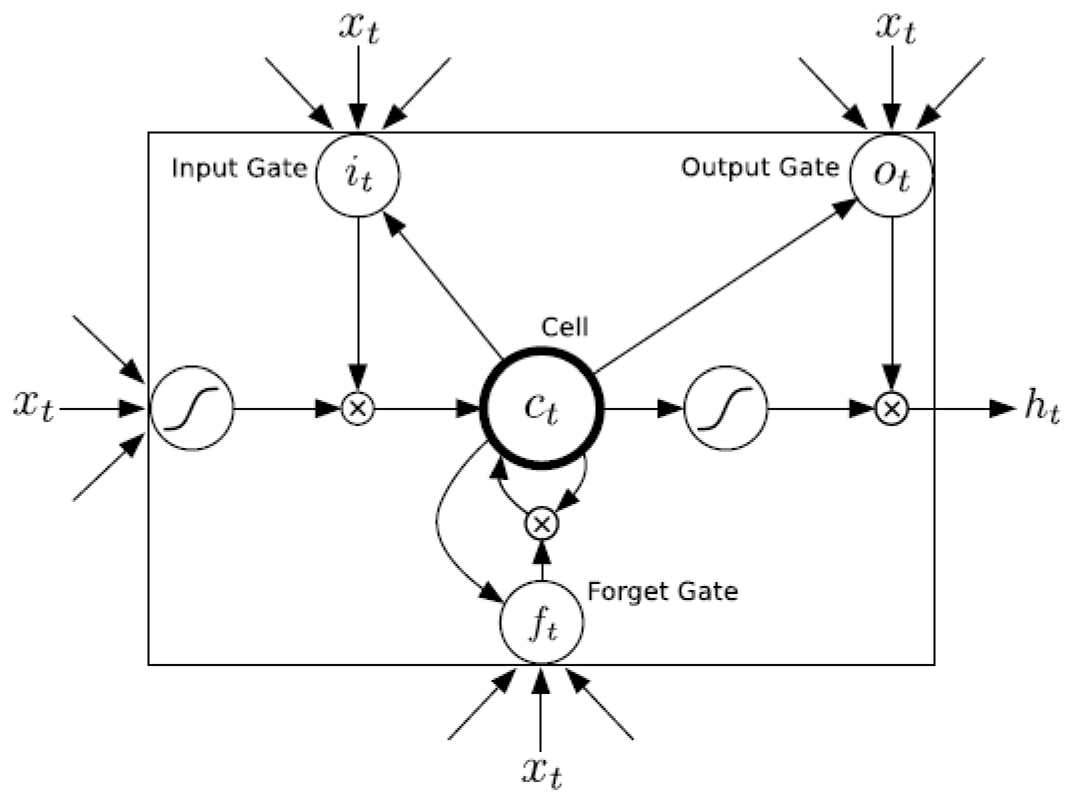
Probability for input sequence:

$$\Pr(\mathrm{x}) = \prod_{t=1}^{T} \Pr(x_{t+1} | y_t)$$

Cost/Loss:

$$\mathcal{L}(\mathrm{x}) = -\sum_{t=1}^{T} \log \Pr(x_{t+1} | y_t)$$

Long Short Term Memory(LSTM)

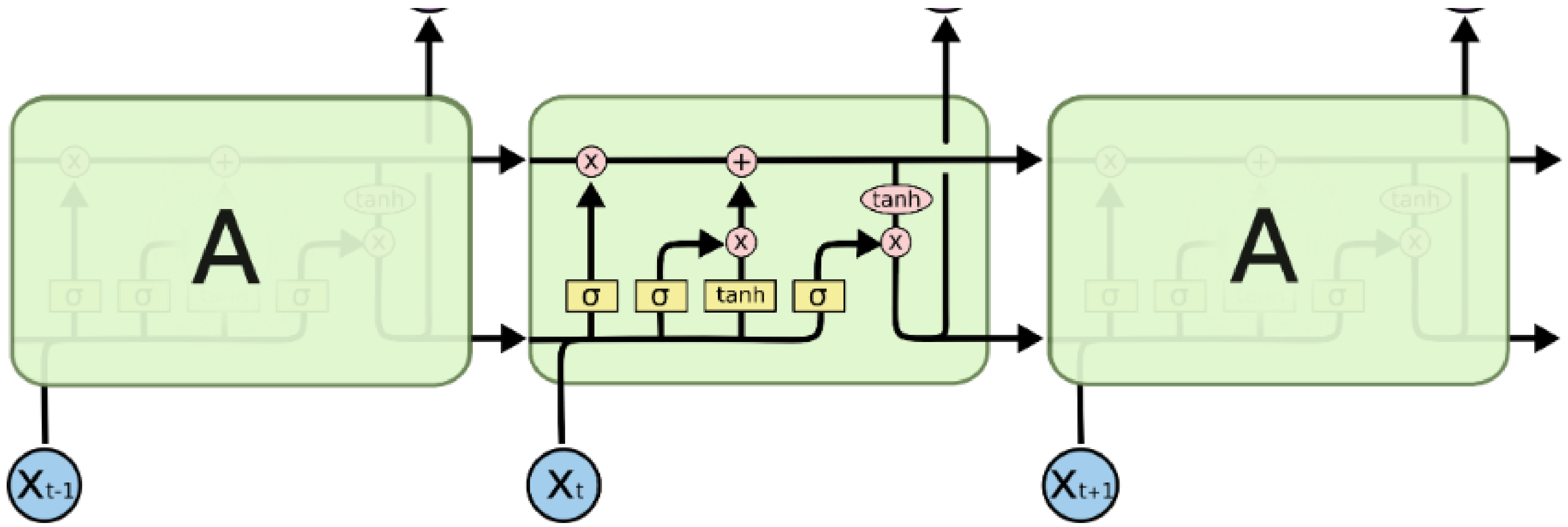


Peephole LSTM

$$
\begin{aligned}
i_t &= \sigma\left(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i\right)\\
f_t &= \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f\right)\\
c_t &= f_t c_{t-1} + i_t \tanh\left(W_{xc}x_t + W_{hc}h_{t-1} + b_c\right)\\
o_t &= \sigma\left(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o\right)\\
h_t &= o_t \tanh(c_t)
\end{aligned}
$$

Gradient: Full gradient is calculated via backpropagation.

Basic LSTM structure



Text Prediction

- Data is discrete and one hot encoded
- K text classes
  - K can be a large value
  - Requires large amount of training data
  - High computational cost at softmax layer
- Character level models

$$\Pr(x_{t+1} = k | y_t) = y_t^k = \frac{\exp\left(\hat{y}_t^k\right)}{\sum_{k'=1}^{K} \exp\left(\hat{y}_t^{k'}\right)}$$

Experiments

1. Penn Treebank portion of Wall Street Journal corpus
    - Gauge the predictive power of the model
    - Model: single hidden layer with 1000 LSTM units
    - Dynamic evaluation
    - Regularization: Weight noise and adaptive weight noise

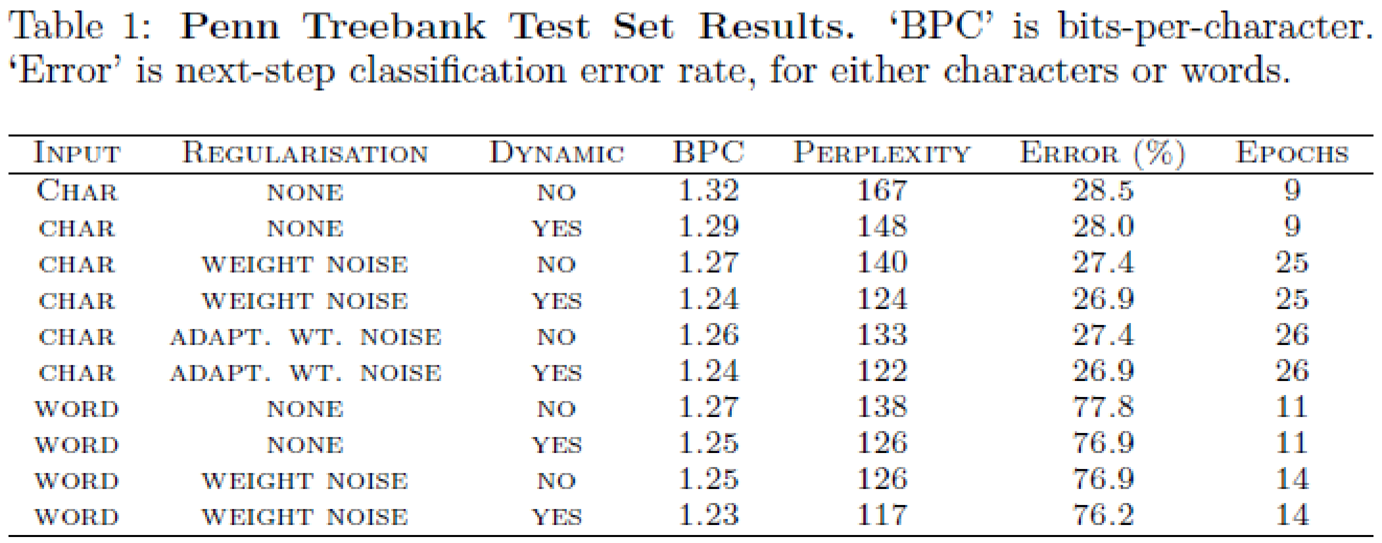


Table 1: **Penn Treebank Test Set Results.** 'BPC' is bits-per-character. 'Error' is next-step classification error rate, for either characters or words.

Input	Regularisation	Dynamic	BPC	Perplexity	Error (%)	Epochs
Char	none	no	1.32	167	28.5	9
char	none	yes	1.29	148	28.0	9
char	weight noise	no	1.27	140	27.4	25
char	weight noise	yes	1.24	124	26.9	25
char	adapt. wt. noise	no	1.26	133	27.4	26
char	adapt. wt. noise	yes	1.24	122	26.9	26
word	none	no	1.27	138	77.8	11
word	none	yes	1.25	126	76.9	11
word	weight noise	no	1.25	126	76.9	14
word	weight noise	yes	1.23	117	76.2	14

Experiments

2. Wikipedia Experiment

- Not a traditional corpora
- Data contains a total of 205 one-byte Unicode symbols
- Model: 7 hidden layers with 700 LSTM cells.
- The internals states(h_t and c_t) were reset every 100 byte sequence.
- Performance is better with Dynamic Evaluation.

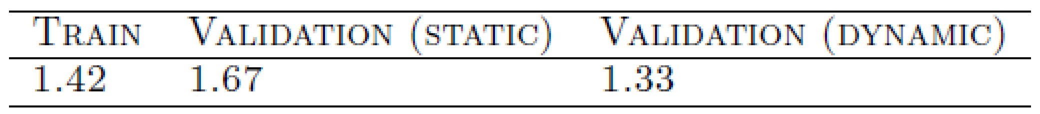


Table 2: **Wikipedia Results (bits-per-character)**

TRAIN	VALIDATION (STATIC)	VALIDATION (DYNAMIC)
1.42	1.67	1.33

Hand writing Prediction

- I AM online handwriting database(real-valued sequence)
- Data consists of X and Y coordinates and the point in seq when the pen was lifted.
- Network is trained to predict of X and Y coordinates and end of stroke.
- Issues: Memory as each letter requires 25 time steps and line requires 700. Also predicting delayed strokes.
- Challenge : How to get a predictive distribution suitable for real-valued inputs

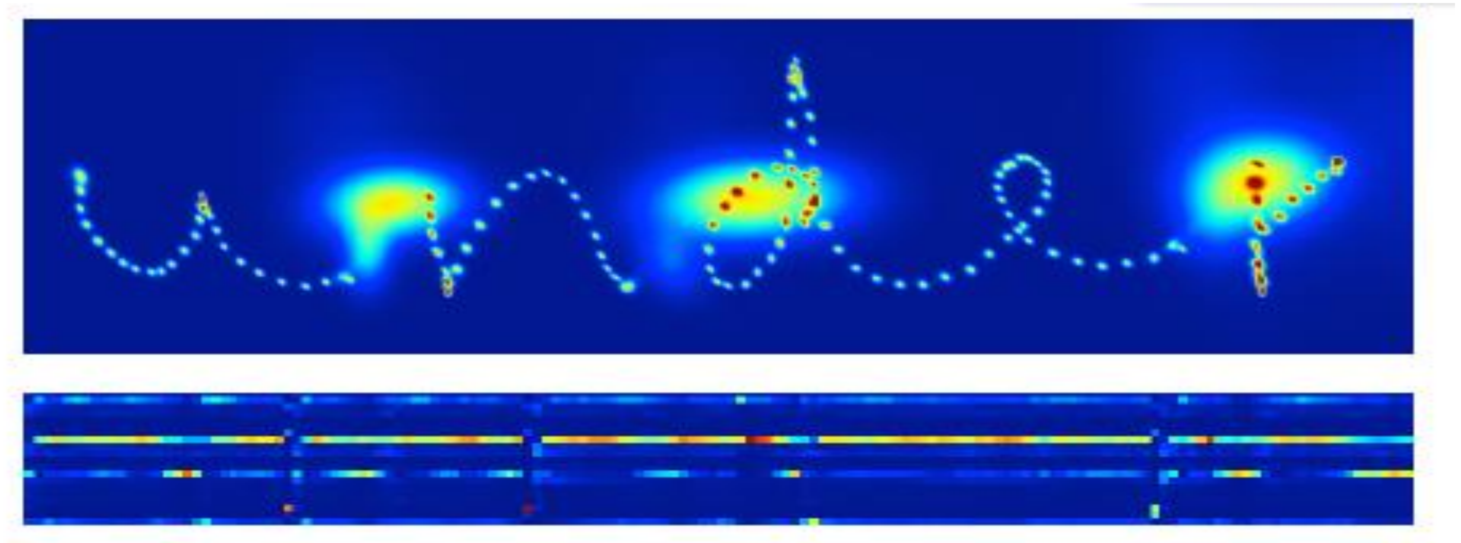
Mixture Density Outputs

- Use the Output of NN to parameterize a mixture distribution.
- Used when there is several distribution- here there us bivariate gaussian and Bernoulli distribution
- Mixture weight outputs are normalized with a softmax function to get valid discrete distribution
- Mixture density network are trained by maximizing the log probability density of the targets under the induced distributions.
- Number of components is the number of choices the network has for the next output given the inputs so far.
- Each output vector y_t therefore consists of the end of stroke probability *e*, along with a set of means , standard deviations , correlations and mixture weights for the M mixture components.

Experiment:

- Model:
  - Network Input layer of size 3
  - Output layer size is 121(20 weights, 40 means, 40 standard deviations and 20 correlations and 1 end-of stroke probability)
  - Two network architecture
    - 3 hidden layers with 400 LSTM
    - 1 hidden layer with 900 LSTM
- networks were trained with rmsprop, a form of stochastic gradient descent where the gradients are divided by a running average of their recent magnitude. [

Results



Network	Regularisation	Log-Loss	SSE
1 layer	none	-1025.7	0.40
3 layer	none	-1041.0	0.41
3 layer	adaptive weight noise	-1057.7	0.41

Handwriting Synthesis

- Generating handwriting from text
- Challenge: Sequences have varied length. Not determined until generated.
- Use of RNN transducer was not encouraging
- Fix: Soft window convoluted to text string and given as input to prediction network.

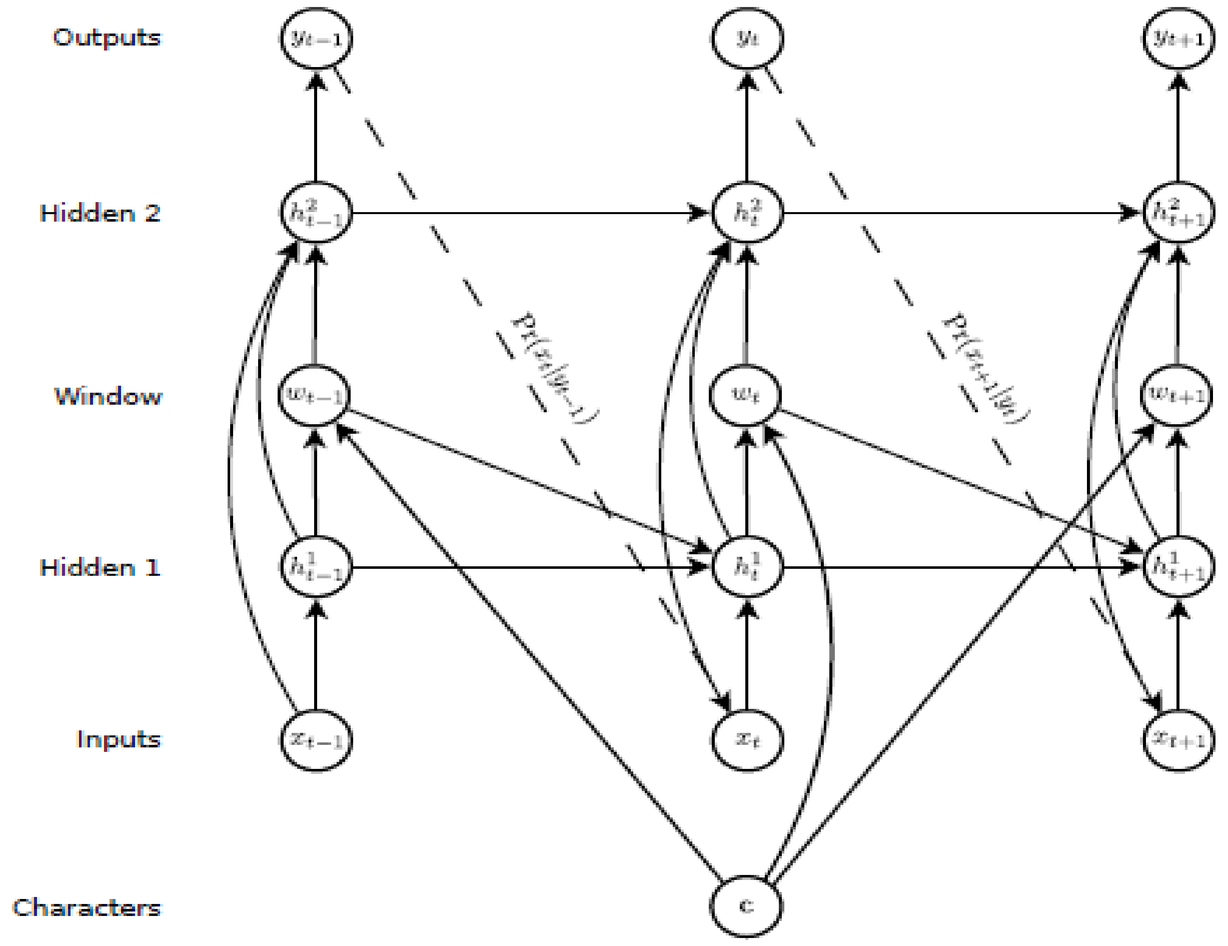


Figure 12: **Synthesis Network Architecture** Circles represent layers, solid lines represent connections and dashed lines represent predictions. The topology is similar to the prediction network in Fig. 1, except that extra input from the character sequence **c**, is presented to the hidden layers via the window layer (with a delay in the connection to the first hidden layer to avoid a cycle in the graph).

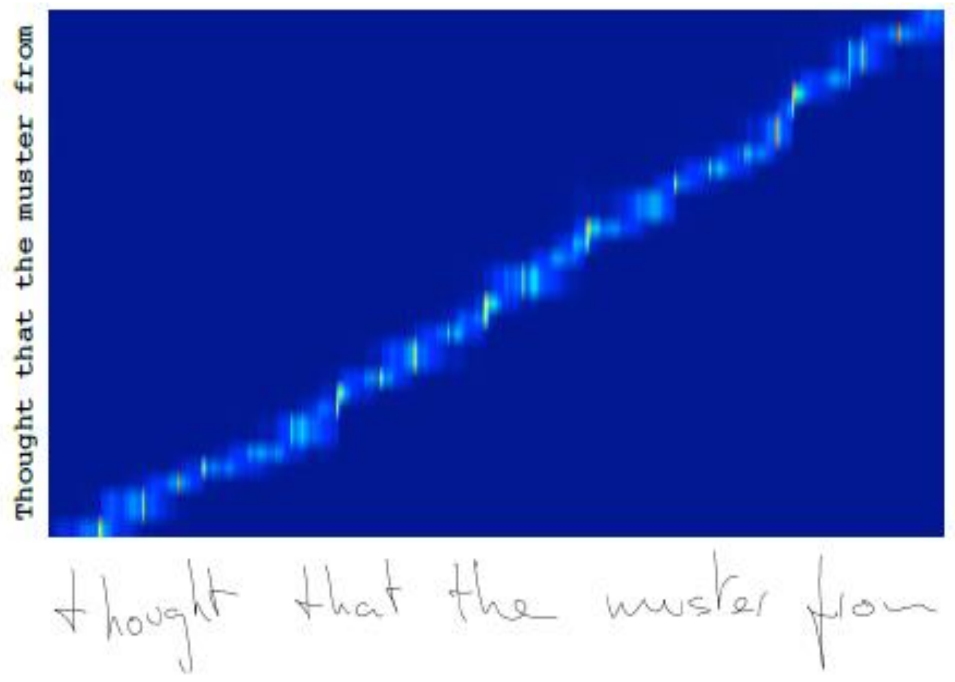
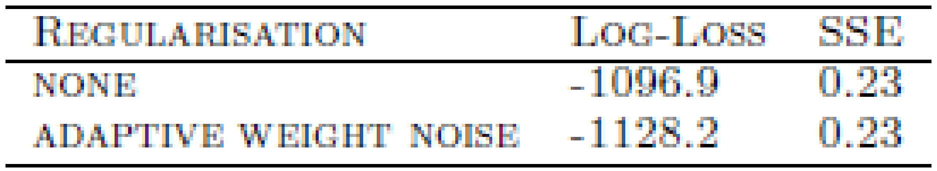


Figure 13: **Window weights during a handwriting synthesis sequence**
$\phi(t,u)$ is the *window weight* of c_u at timestep t.

Experiment:

- Character level transcriptions from I AM online handwriting database.
- Model
  - 3 hidden layers with 400 LSTM 20 bivariate Gaussian mixture components at the output layer and a size 3 input layer
  - Character sequence was encoded with one-hot vectors[Size 57].



REGULARISATION	LOG-LOSS	SSE
NONE	-1096.9	0.23
ADAPTIVE WEIGHT NOISE	-1128.2	0.23

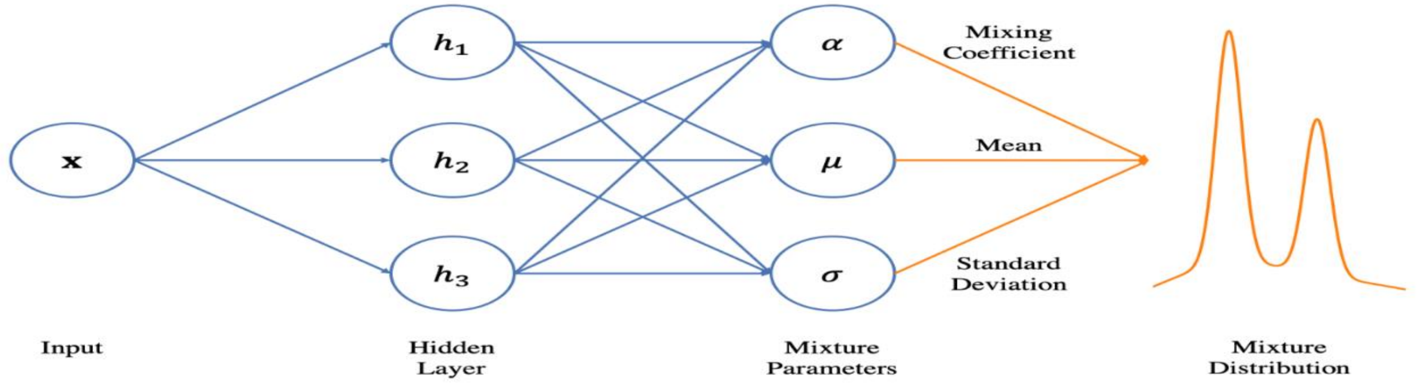
Handwriting Synthesis Results.

Conclusion and Future works:

- Paper talks about LSTM and how it can be used to generate sequences.
- both convolutional mechanism that allows a recurrent network to condition its predictions on an auxiliary annotation sequence, and used this approach to synthesis diverse and realistic samples of online handwriting. Furthermore, it has shown how these samples can be biased towards greater legibility, and how they can be modelled on the style of a particular writer.
- Speech or music synthesis
- Further aspects in handwriting

Questions

- What does dynamic evaluation mean/do ?
  - Allow the network to adapt its weights as it is being evaluated. In the LSTM model it is mentioned that the full gradient is calculated after the complete sequence rather than approx. gradient calculation at each time step.
- What are mixture density outputs/networks and why are they used ?
  - The Mixture Density Network consists of a feed-forward neural network whose outputs determine the parameters in a mixture density model. The mixture model then represents the conditional probability density function of the target variables, conditioned on the input vector to the neural network. NN provides the parameters for multiple distributions, which are then mixed by some weights. These weights are also provided by the NN



Questions:

- What is a window in the handwriting synthesis ?
  Is the layer that handles the character sequence input.

Resources:

- Mixture Density Network:
  - https://towardsdatascience.com/a-hitchhikers-guide-to-mixture-density-networks-76b435826cca
  - https://publications.aston.ac.uk/373/1/NCRG_94_004.pdf
- LSTM: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21