Generating Sequences with Recurrent Neural Networks

Alex Graves

Google DeepMind

Why Generate Sequences?

- To improve generalisation?
- To create synthetic training data?
- Practical tasks like speech synthesis?
- To simulate situations?
- To understand the data

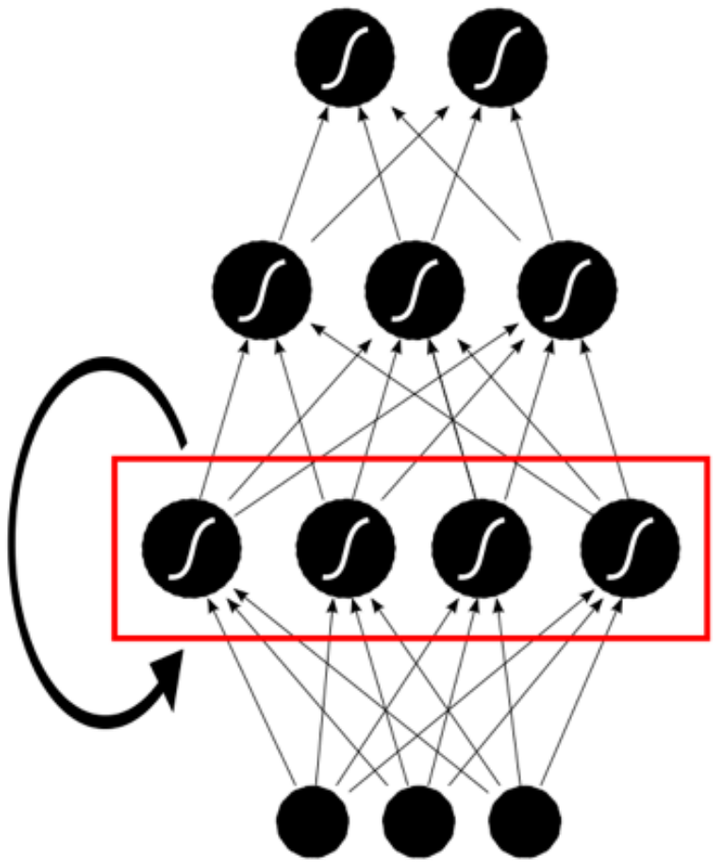
Generation and Prediction

- Easy way to generate a sequence: repeatedly predict what will happen next, treating your past predictions as if they were real
- In probabilistic terms, sampling from a conditional model

$$\Pr(\mathbf{x}) = \prod_t \Pr(x_t | x_{1:t-1})$$

- The closest computers get to dreaming…

Recurrent neural networks (RNNs)



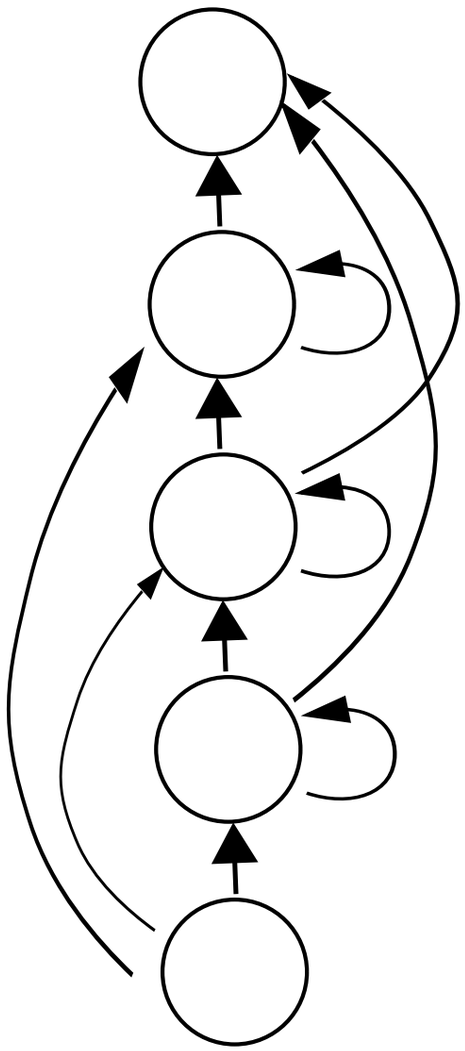
- Like feedforward networks except that one or more layers is connected to itself
- Self connections allow the network to build an internal representation of past inputs
- In effect they give the network memory

Prediction Architecture

output layer

Hidden layers

inputs



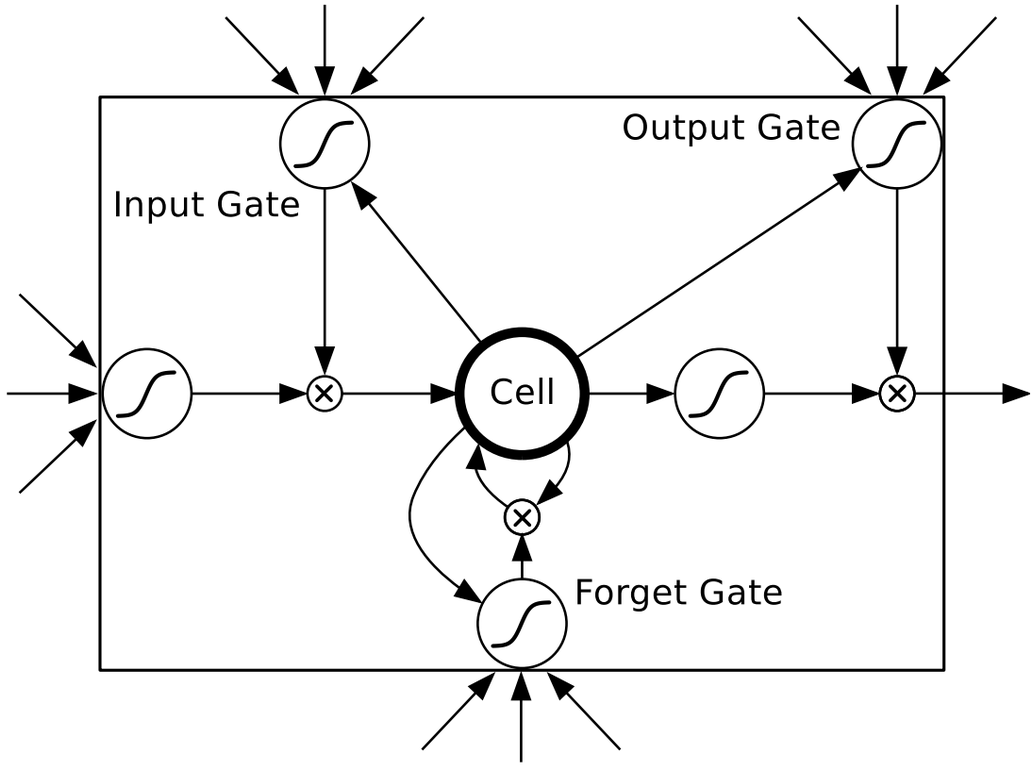
- Recurrent neural network with one or more hidden layers and skip connections
- Inputs arrive one at a time, outputs determine predictive distribution over next input
- Train by minimising log-loss:

$$\sum_{t=1}^{T} -\log \Pr(x_t | x_{1:t-1})$$

- Generate by sampling the output distribution and feeding into input

Long Short-Term Memory

- LSTM is an RNN architecture designed to have a longer memory. It uses linear memory cells surrounded by multiplicative gate units to store information



Input gate: scales input to cell (write)

Output gate: scales output from cell (read)

Forget gate: scales old cell value (reset)

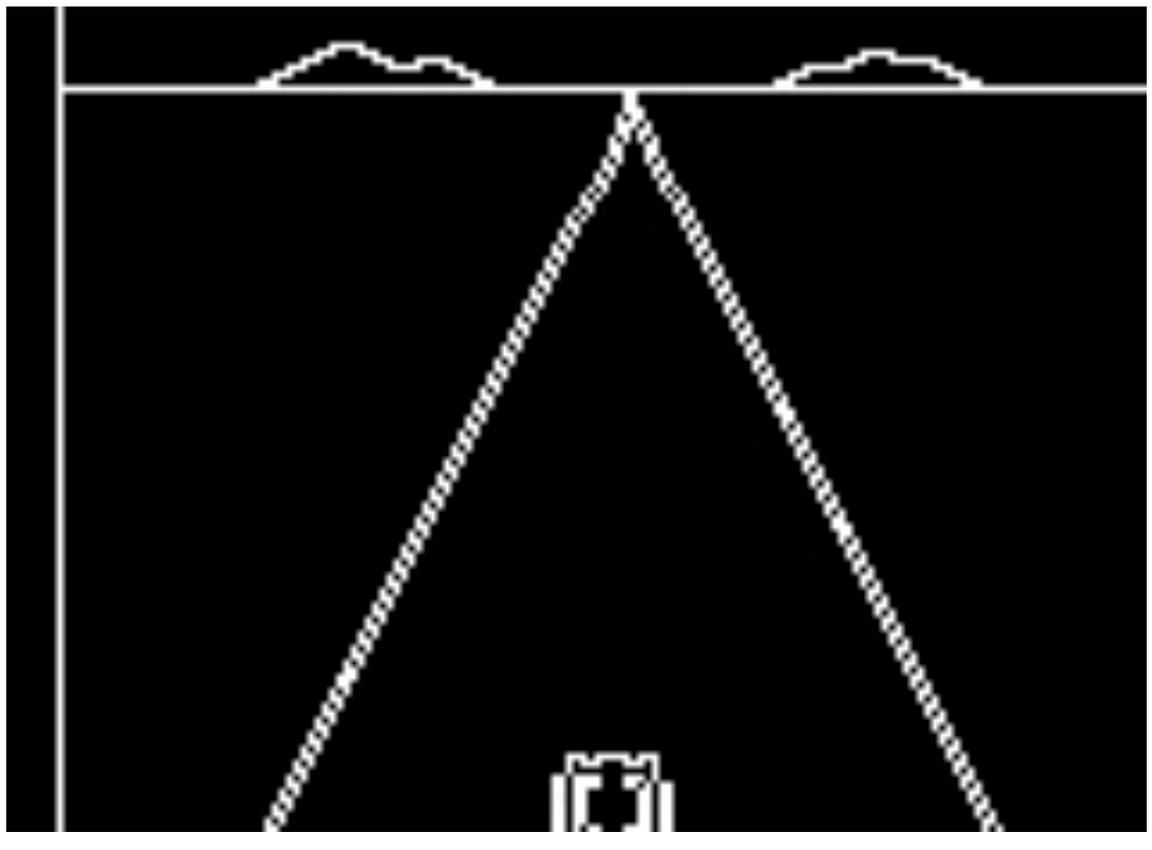
- S. Hochreiter and J. Schmidhuber, “Long Short-term Memory” Neural Computation 1997

Memory and Prediction

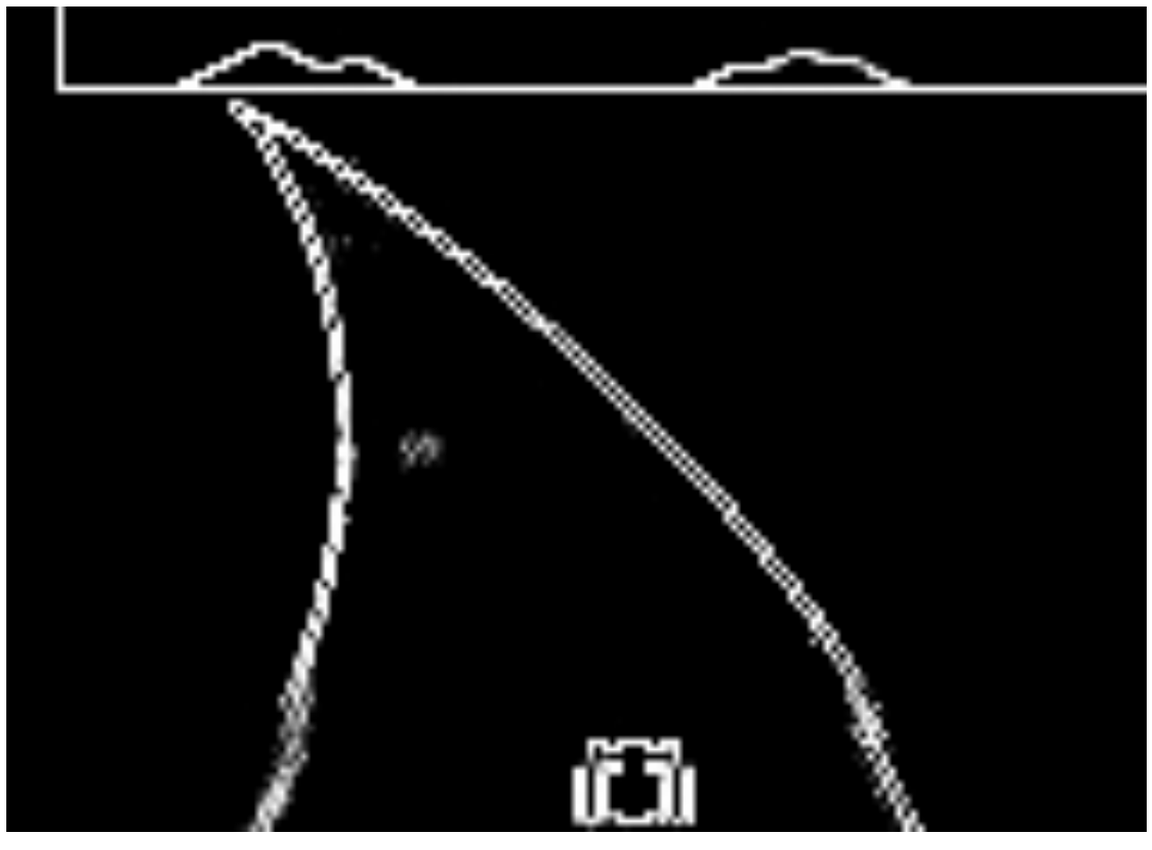
- Need to remember the past to predict the future
- Having a longer memory has several advantages:
  - can handle long range patterns
  - especially 'disconnected' patterns like balanced quotes and brackets
  - more robust to prediction errors

Atari Experiments

Real



Generated



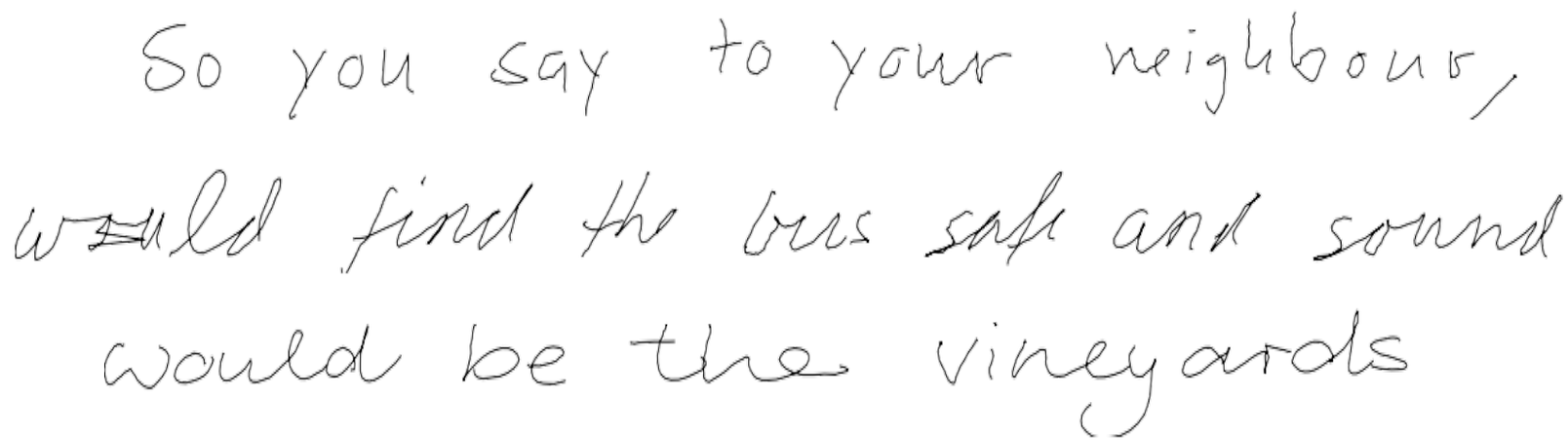
Karol Gregor, Ivo Danihelka, Andriy Mnih, Daan Wierstra…

Real

Generated

Handwriting Experiments

- Task: generate pen trajectories by predicting one (x,y) point at a time
- Data: IAM online handwriting, 10K training sequences, many writers, unconstrained style, captured from a whiteboard



- First problem: what to use for the density model?

Recurrent Mixture Density Networks

- Network outputs parameterise a mixture distribution (usually Gaussian)
- Every prediction conditioned on all inputs so far

$$\Pr(x_{t+1}|x_{1:t}) = \sum_k w_k(x_{1:t})\mathcal{N}(x_{t+1}|\mu_k(x_{1:t}), \Sigma_k(x_{1:t}))$$

- Number of components is number of *choices* for what comes next

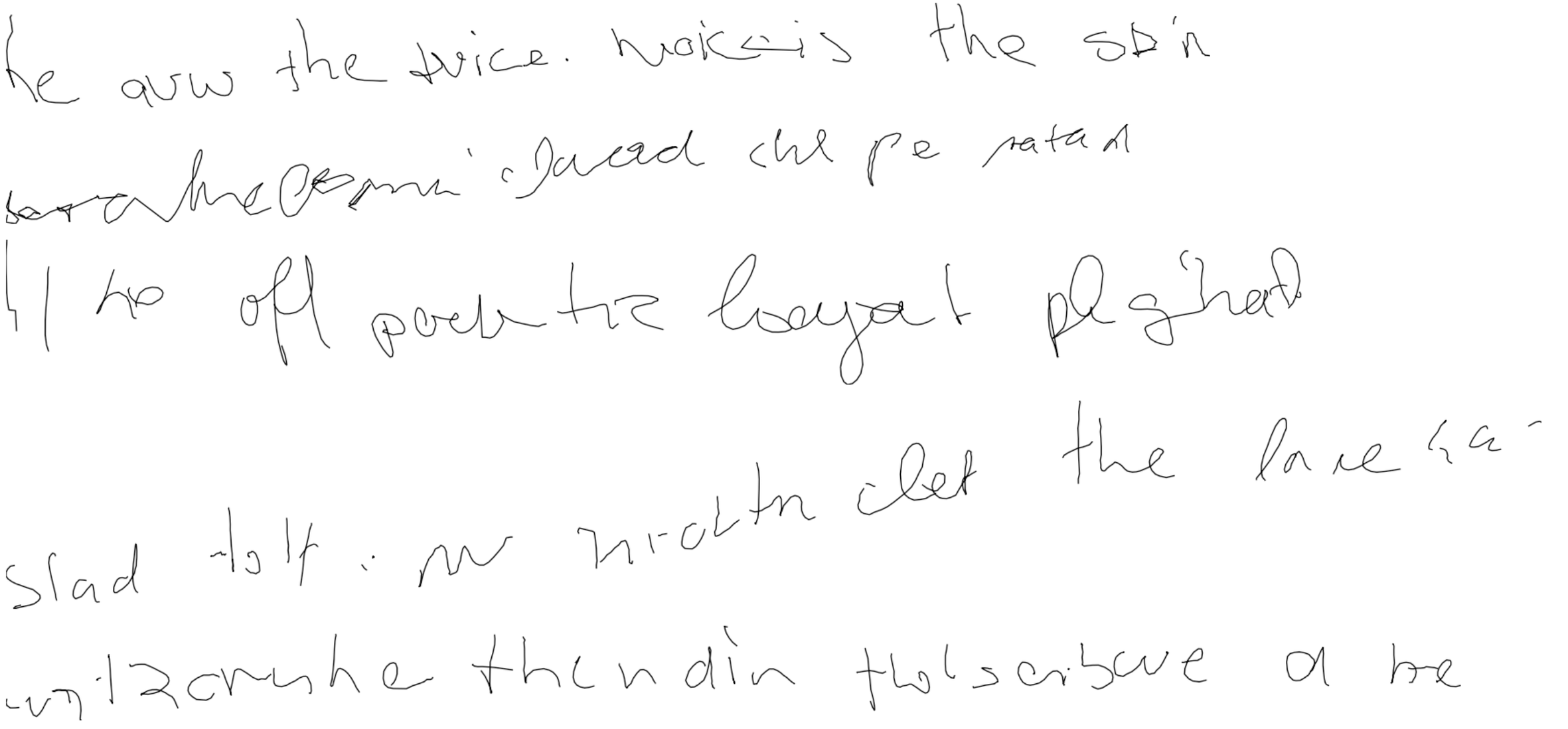
- M. Schuster, “Better Generative Models for Sequential Data Problems: Bidirectional Recurrent Mixture Density Networks”, NIPS 1999

Network Details

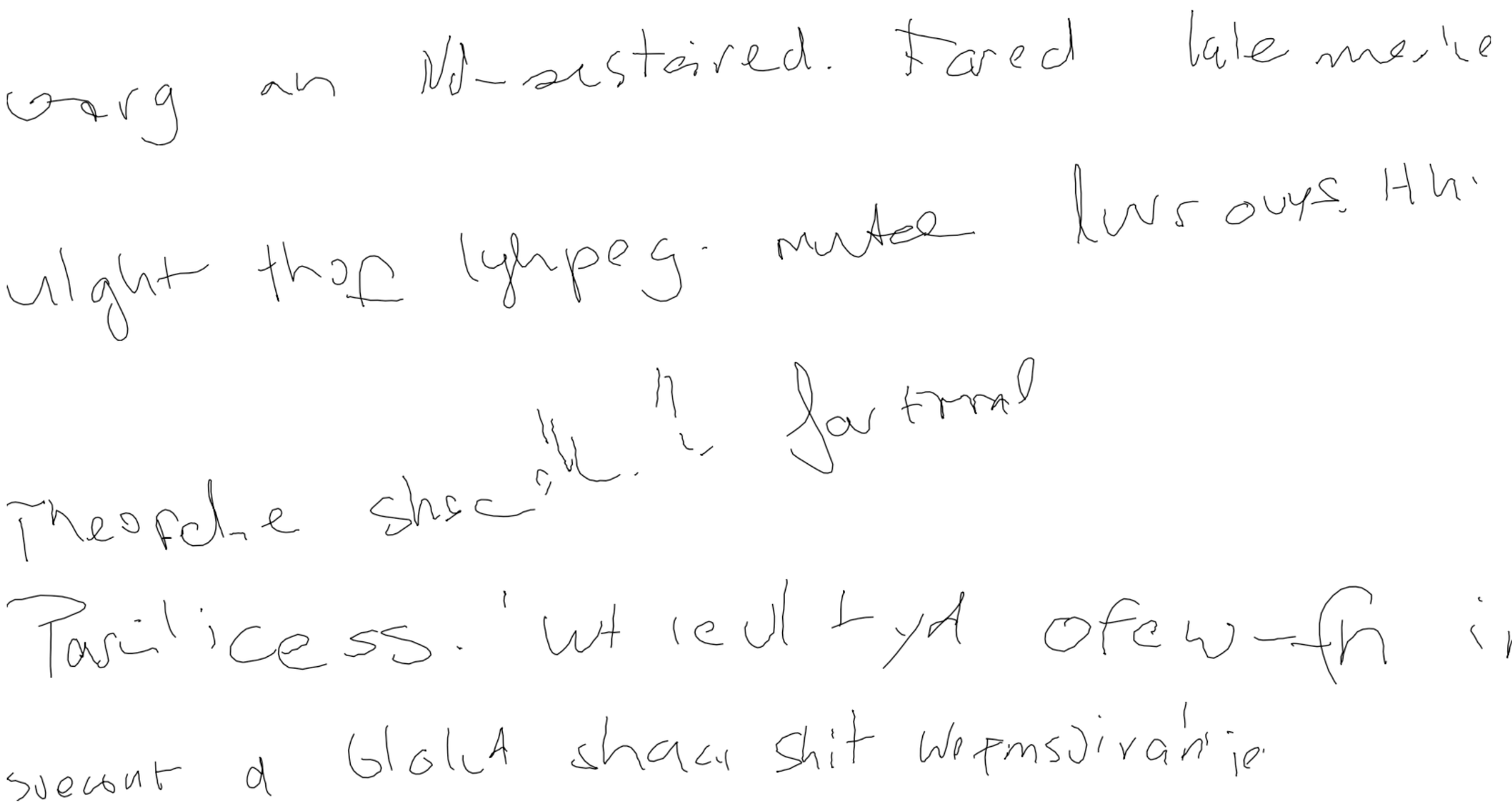
- 3 variables: co-ordinate deltas, pen up/down
- 20 two dimensional Gaussians for co-ords
- 1 sigmoid for up/down
- 3 hidden Layers, 400 LSTM cells in each
- Trained with RMSprop
- Retrained with adaptive weight noise*

* A. Graves, “Practical Variational Inference for Neural Networks”, NIPS 2011

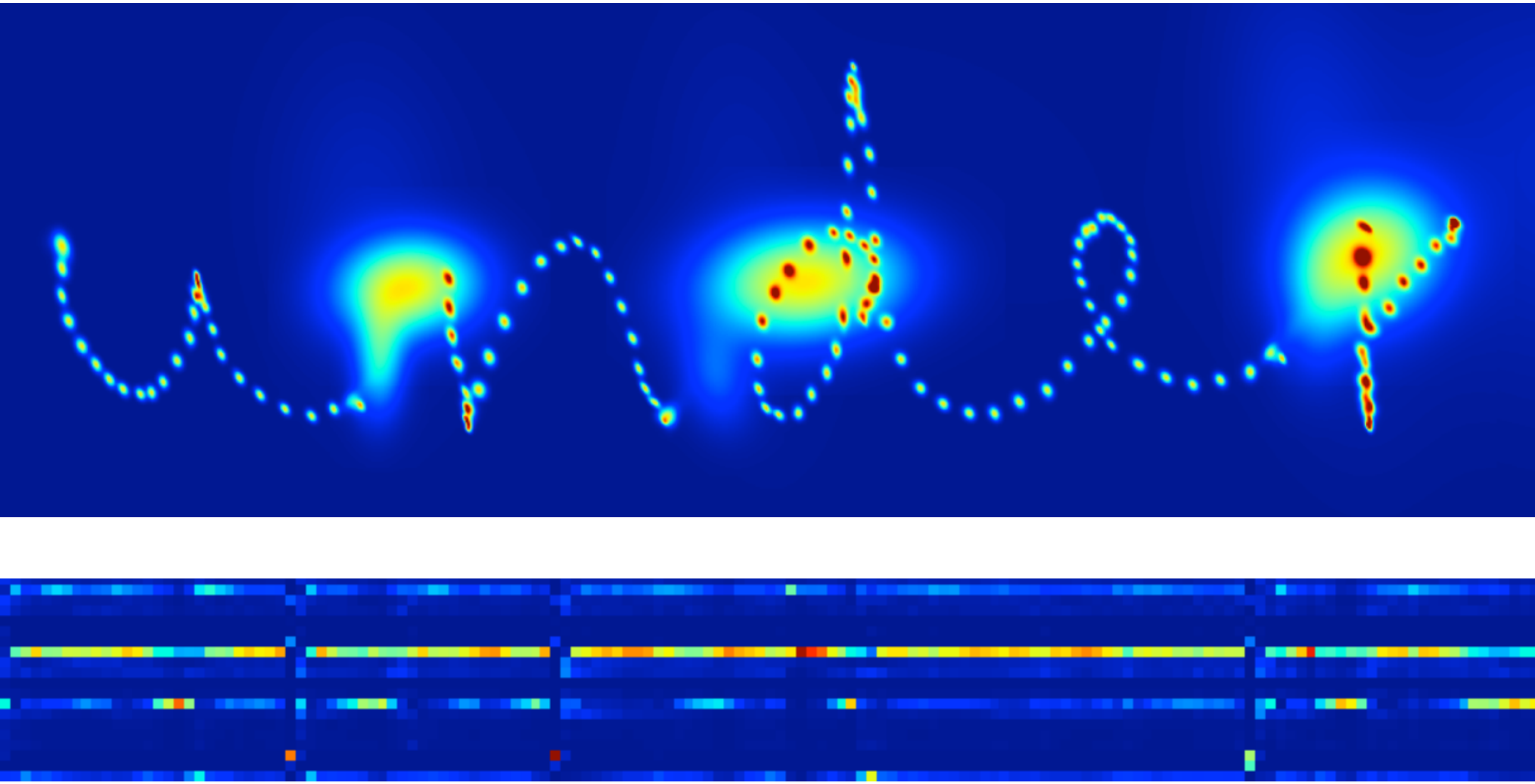
Samples



Samples



Output Density



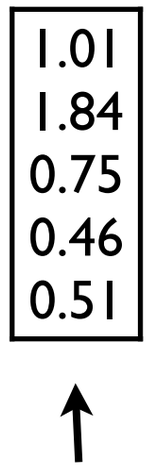
Handwriting Synthesis

- Want to tell the network *what* to write without losing the distribution over *how* it writes
- Can do this by conditioning the predictions on a text sequence
- Problem: alignment between text and writing unknown
- Solution: before each prediction, let the network decide where to look in the text sequence

Gaussian ‘Window’

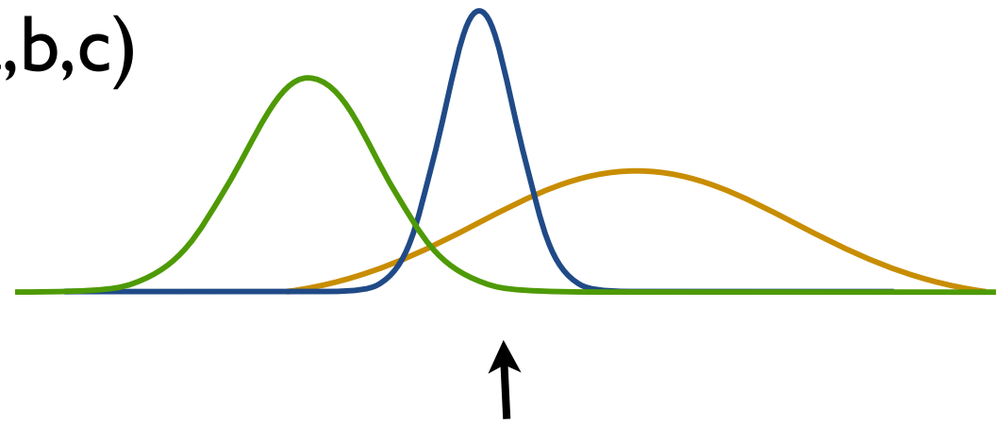
Window vector (input to net)

$$v^{t+1} = \sum_{i=1}^{S} w_i^t s_i$$



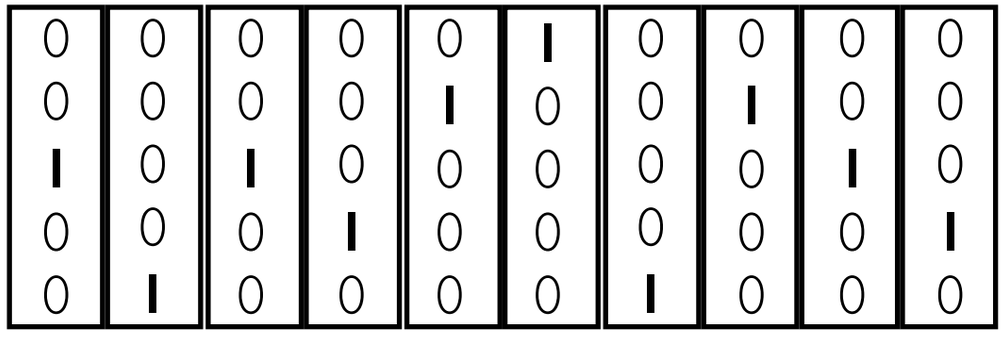
Window weights (net outputs for a,b,c)

$$w_i^t = \sum_{k=1}^{K} a_k^t \exp\left(-b_k^t [c_k^t - i]^2\right)$$

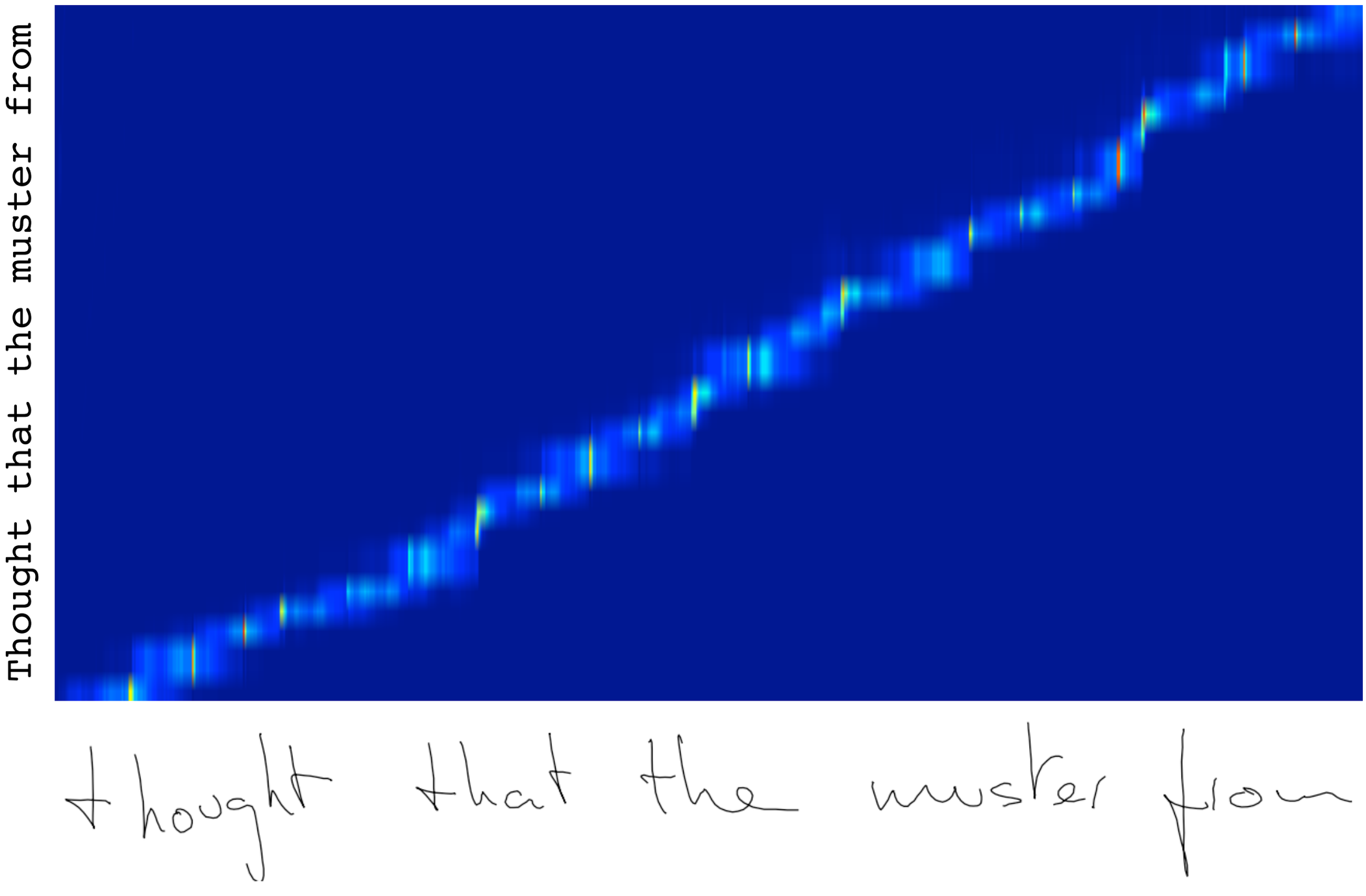


Input vectors (one-hot)

$(s_1, \ldots, s_S)$



Window Alignment



Which is Real?

that a doctor should be

that a doctor should be

thaet a docaus should ae

that a doctor should be

that a doctor should be

that a doctor should be

Which is Real?

of presentlee alty inremmbering

of preseng reality in remmbering

of present reality in remembering

of present reality in remembering

of present reality in remembering

Which is Real?

from his travels it might have been

from his travels it might have been

from his travels 'it might have been

from his travels it mightmare been

from his travels it might have been

from his travels - it might have been

Biased Sampling

when the samples are biased

towards more probable sequences

they get easier to read

but less interesting to look at.

Primed Sampling

when the sample starts with real data

(prison welfare Officer complement)

if continues in the same style

(He dismissed the idea)

Primed *and* Biased

Take the breath away where they are

when the network is primed
and biased, it writes
in a cleaned up version
of the original style

Demo

http://www.cs.toronto.edu/~graves/handwriting.html

Results

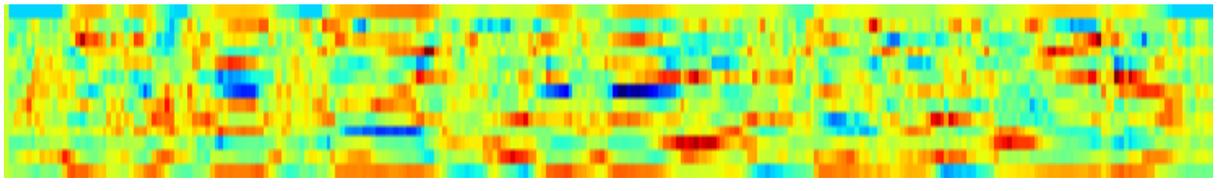
Network	Δ
1 layer prediction	+15
3 layer prediction (baseline)	0
3 layer synthesis	-56
3 lay. synth. + adapt. wt. noise	-86


BBC TOUR

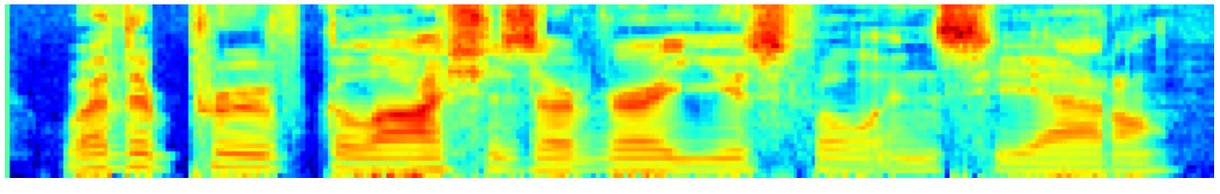
Speech Synthesis

- Speech synthesis (a.k.a. text-to-speech) is harder than handwriting for several reasons:
  - Need to go from letters to phonemes
  - People are more sensitive to 'mistakes' in speech
  - Not obvious how to represent the data

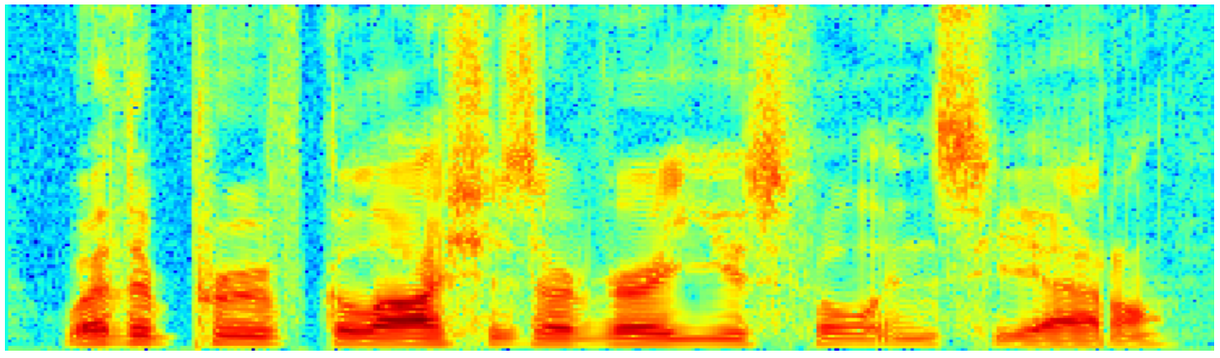
Audio Representations



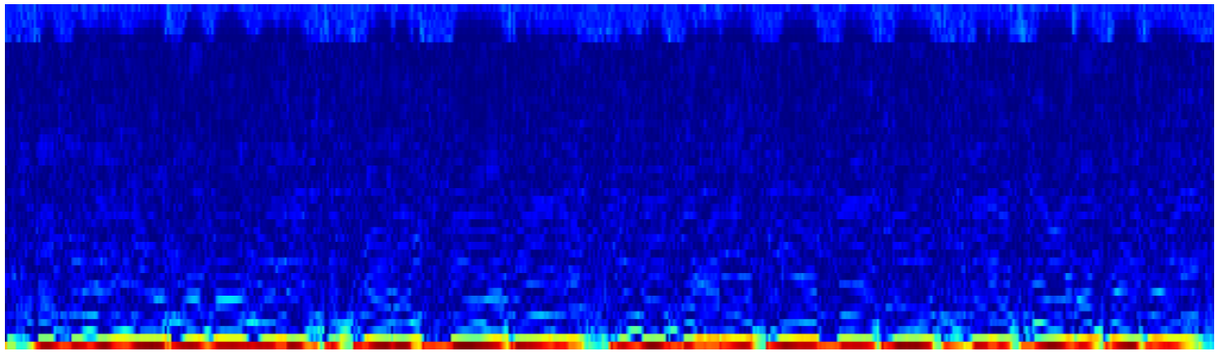
Mel-frequency cepstral coefficients (MFCCs)



Mel-scale Filterbanks



Spectrogram

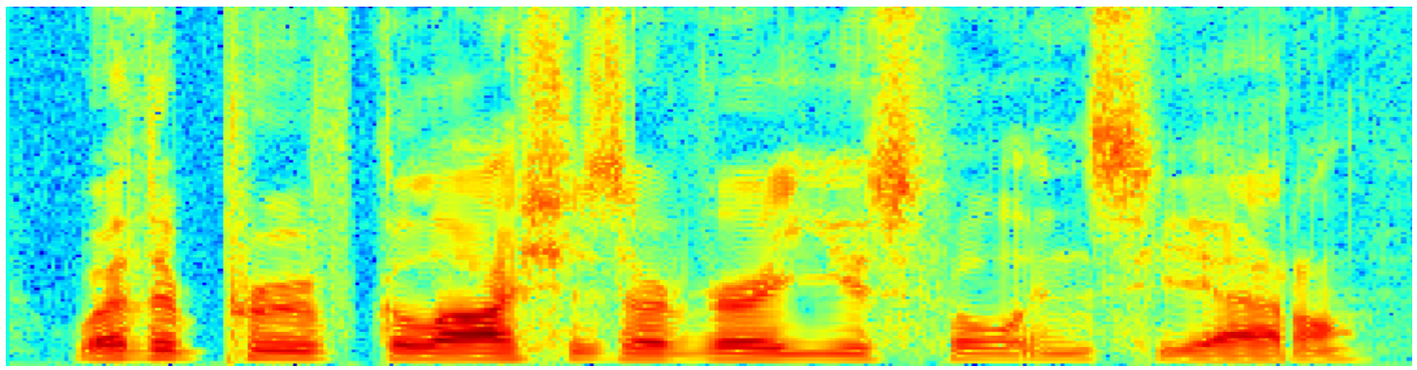


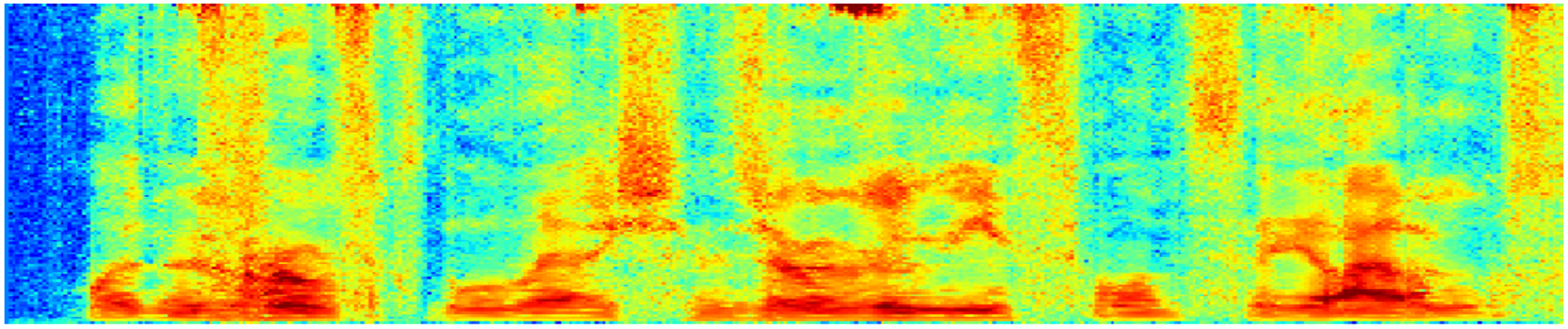
Vocoder: MFCCs, log f0, aperiodicity, voiced/unvoiced

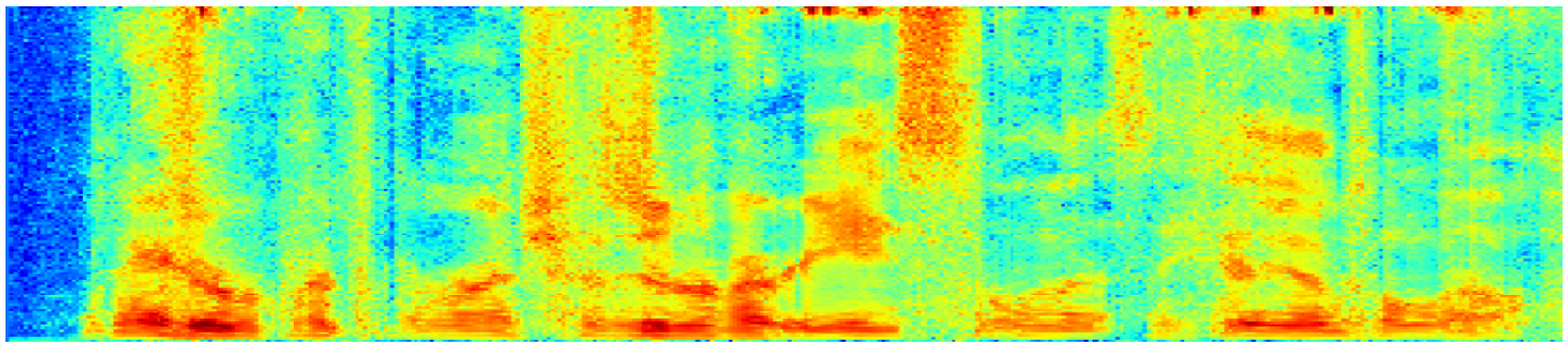
Spectrogram Experiments

- TIMIT database (~4K training utterances, phonetic transcripts, many speakers)
- Mixture of Gaussians density model
- Looks nice, sounds bad (no phase)

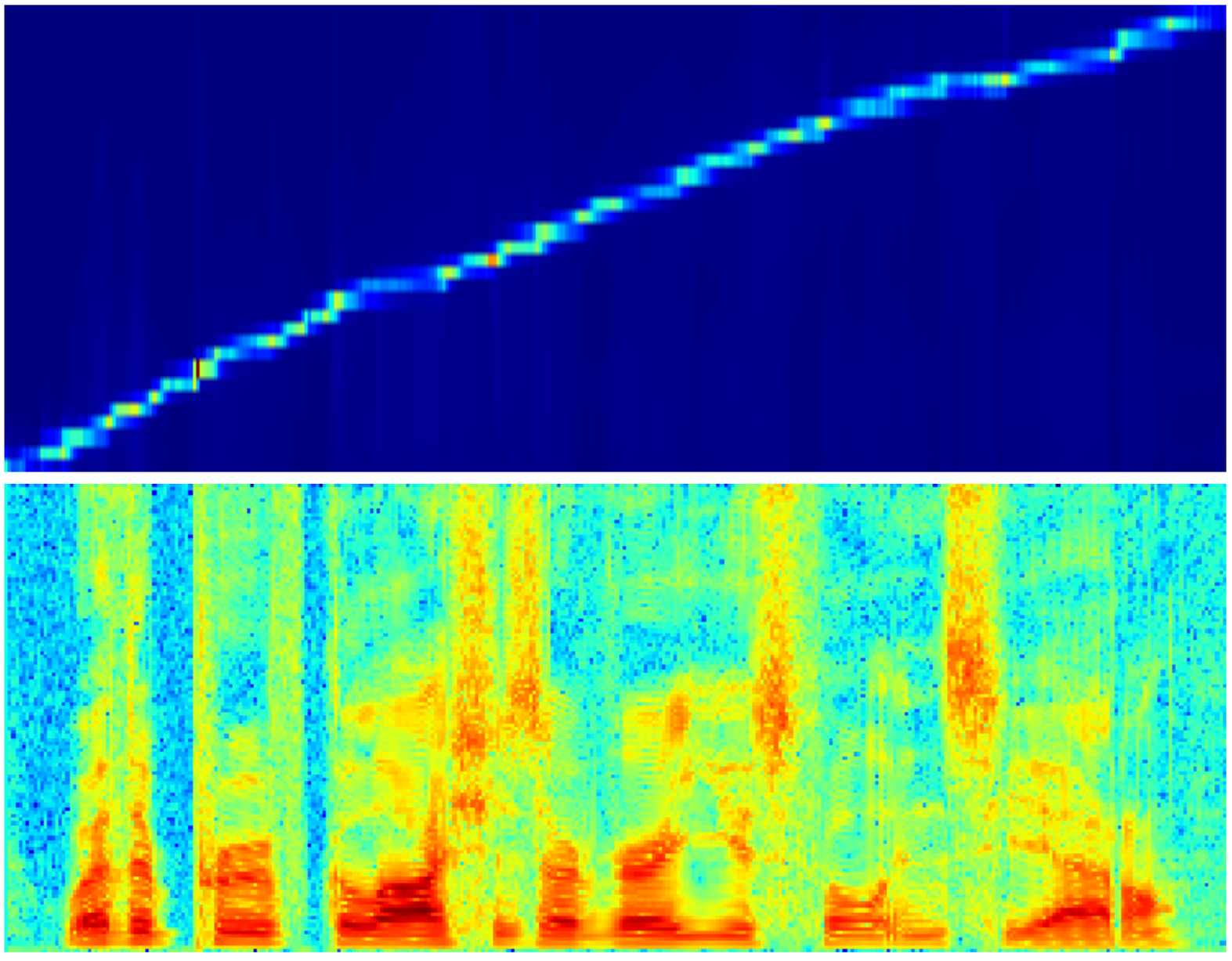
Generated Spectrograms







Alignment



Vocoder Experiments

- Large single speaker database (~35K utterances, female American voice)
- Multidimensional RNN density model
- Trained with and without phonetic transcripts

Where Next?

- Improve speech synthesis (brute force?)
- Learn high level features (strokes, letters, words...) rather than adding them manually
- Learn to write as well as read:
  Neural Turing Machine

We're hiring!

joinus@deepmind.com

www.google.com/jobs

Thank you for your attention !

Text Experiments

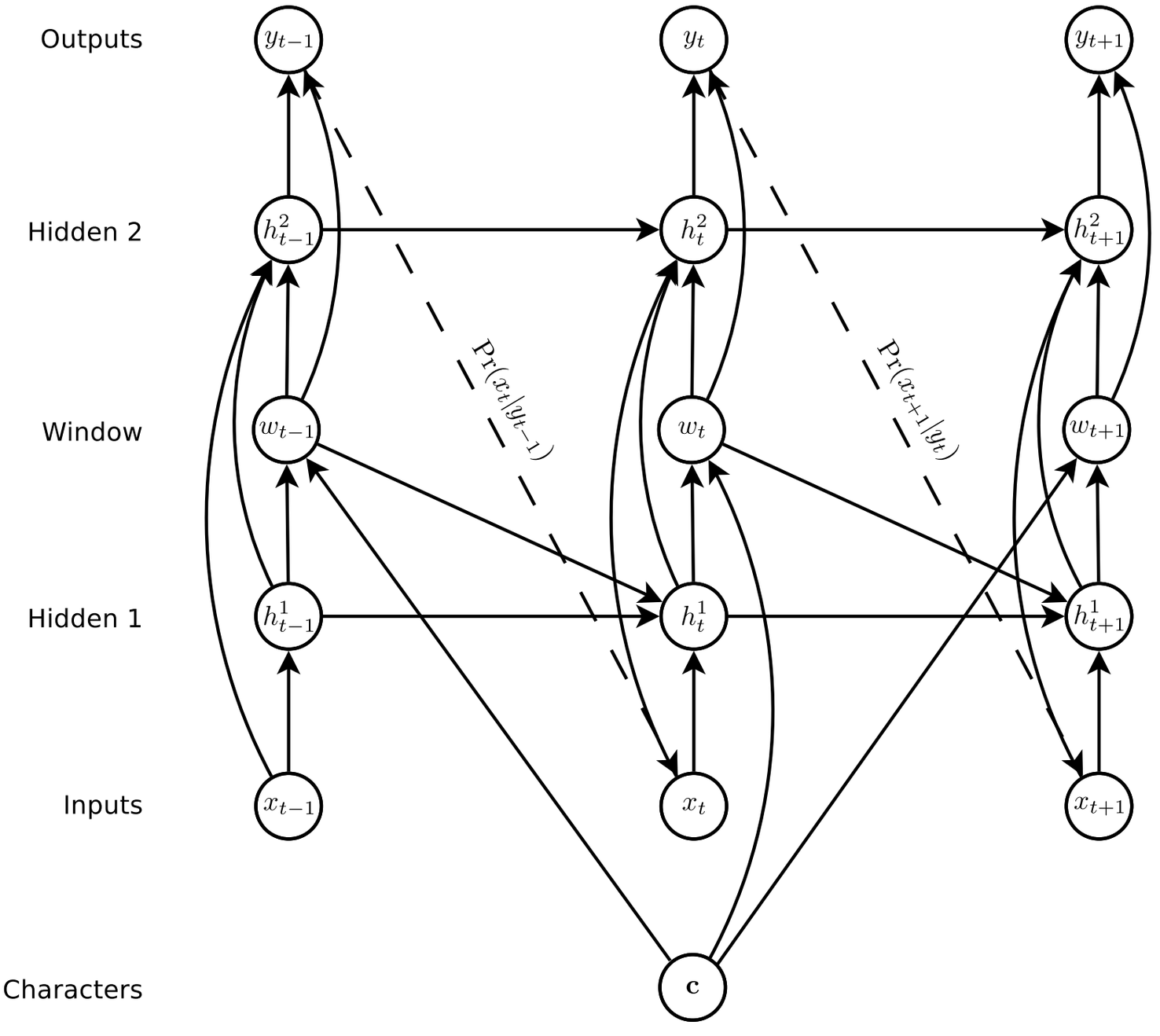
- Task: generate text sequences one character at a time
- Data: raw wikipedia XML from Hutter challenge (100 MB)
- Softmax output layer (205 units)
- 5 hidden layers of 700 LSTM cells, ~21M weights
- Trained with SGD (took forever!)

Compression Results

Method	BPC train	test
bsc	1.67	1.54
ppmonstr	1.53	1.43
zpaq	1.63	1.53
LSTM	**1.42**	**1.33**
kingsize	1.33	1.20

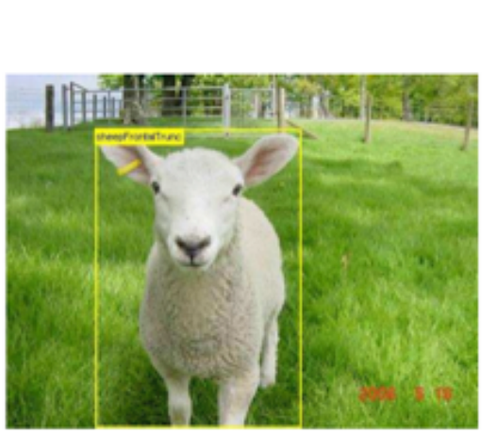
Matt Mahoney: http://encode.ru/threads/1825-Compression-with-recurrent-neural-networks

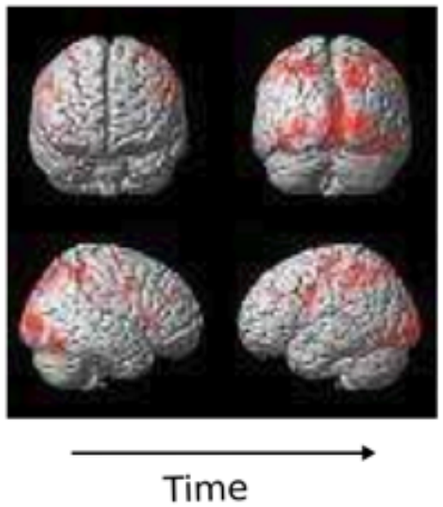
Network Architecture



Multidimensional recurrent neural networks (MDRNNs)

- Normal recurrent nets are only applicable to 1D sequences
- But their properties (e.g. robustness to distortion and flexible use of context) are also desirable for multidimensional data such as images (2D), video (3D), fMRI scans (4D)...

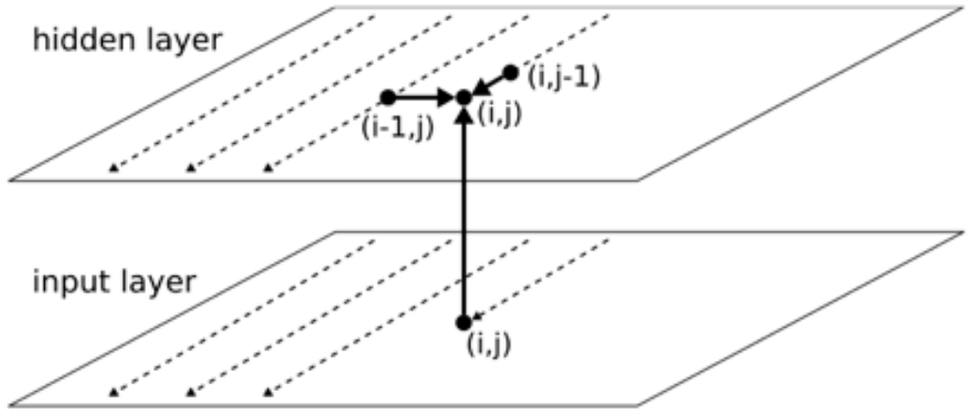




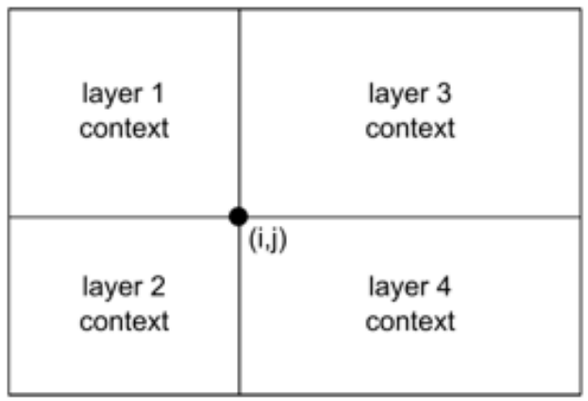
- MDRNNs generalise recurrent nets to an arbitrary number of spacetime dimensions

How MDRNNs work

- Basic idea: replace the single recurrent connection by a separate connection for each dimension in the data.

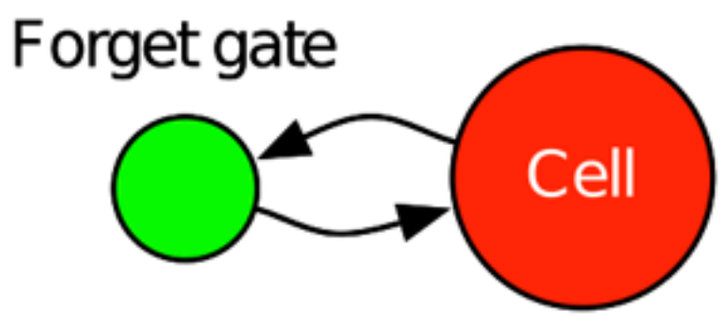


- The network scans the data line by line so that the previous hidden activations are calculated before the current one
- 2^n separate hidden layers are used for n dimensional data, to provide context in all directions



Multidimensional LSTM

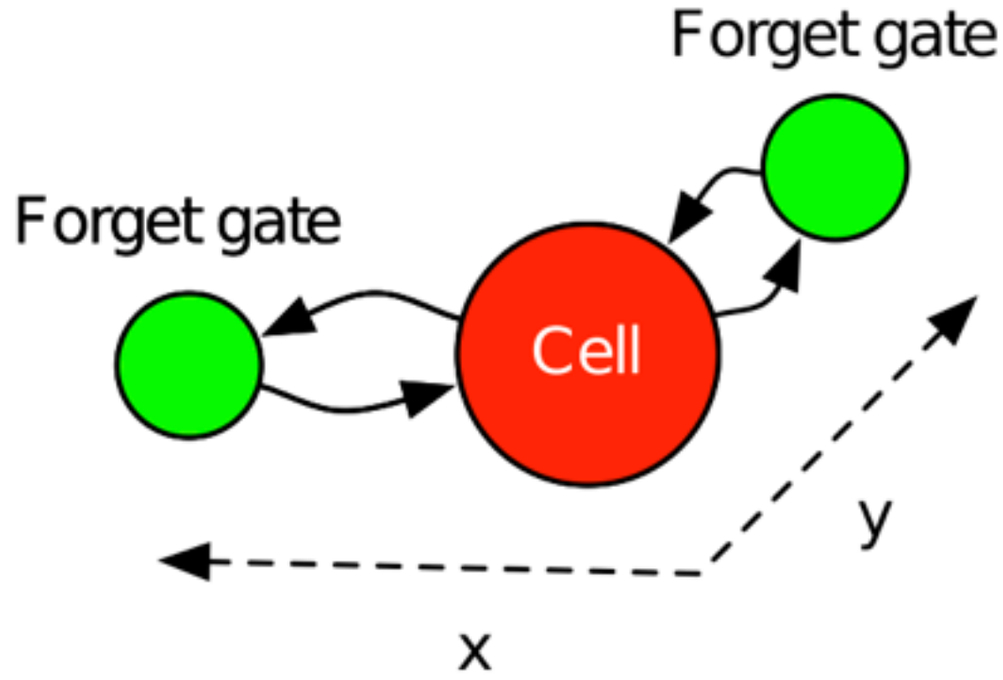
- Standard LSTM is explicitly one dimensional



Graves et. al., 2007

Multidimensional LSTM

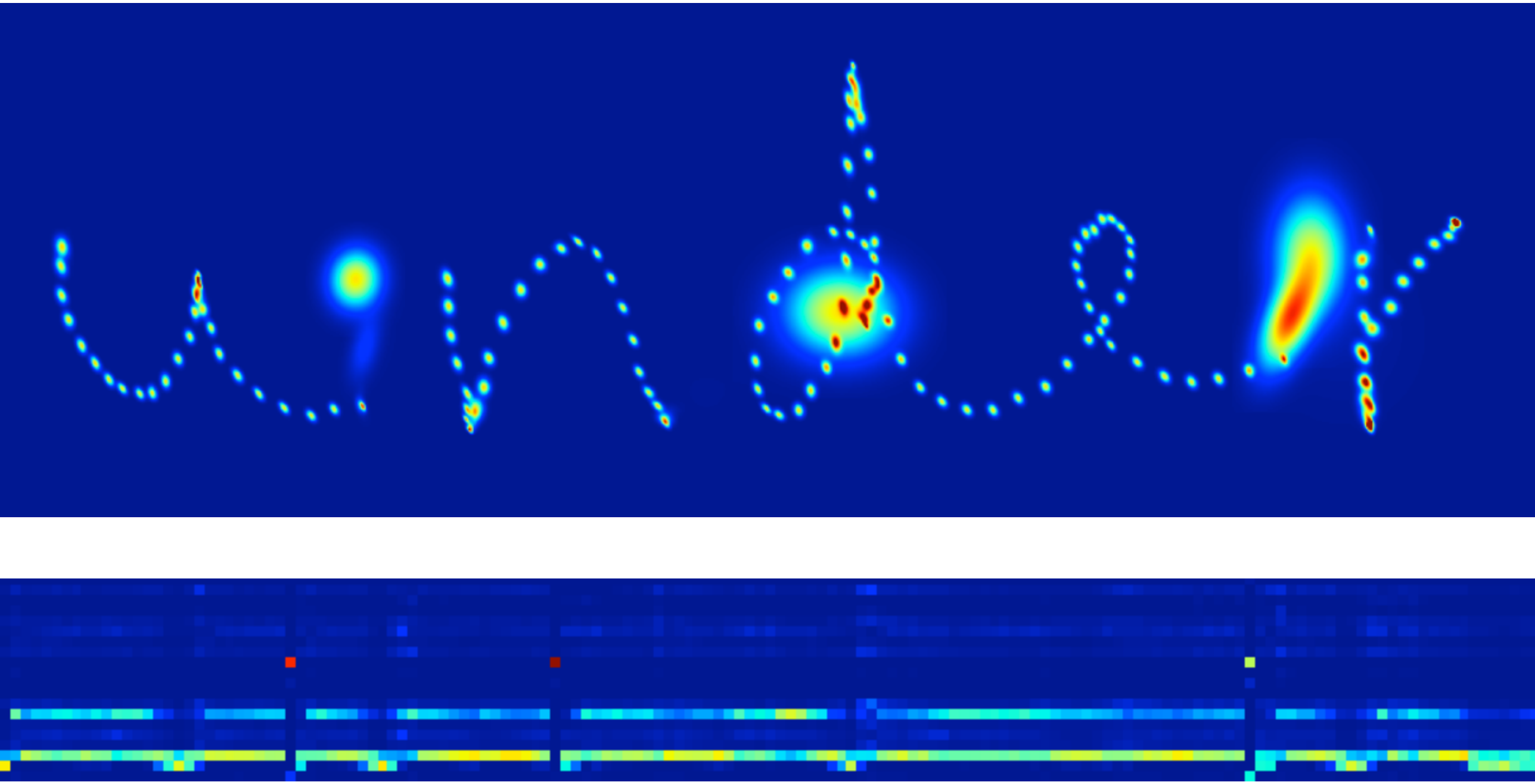
- Standard LSTM is explicitly one dimensional



- But can make it *n* dimensional by giving the memory cells *n* self connections (with *n* forget gates)
- Multidimensional LSTM can access long range context in all directions

Graves et. al., 2007

Synthesis Density



Prediction Density