The Challenge of Sequential Modelling

Stanford University, 26 Sept. 2016

Yoshua Bengio
Montreal Institute for Learning Algorithms
Université de Montréal

PLUG: **Deep Learning**, MIT Press book in pre-sale,
Chapters will remain online

Université de Montréal

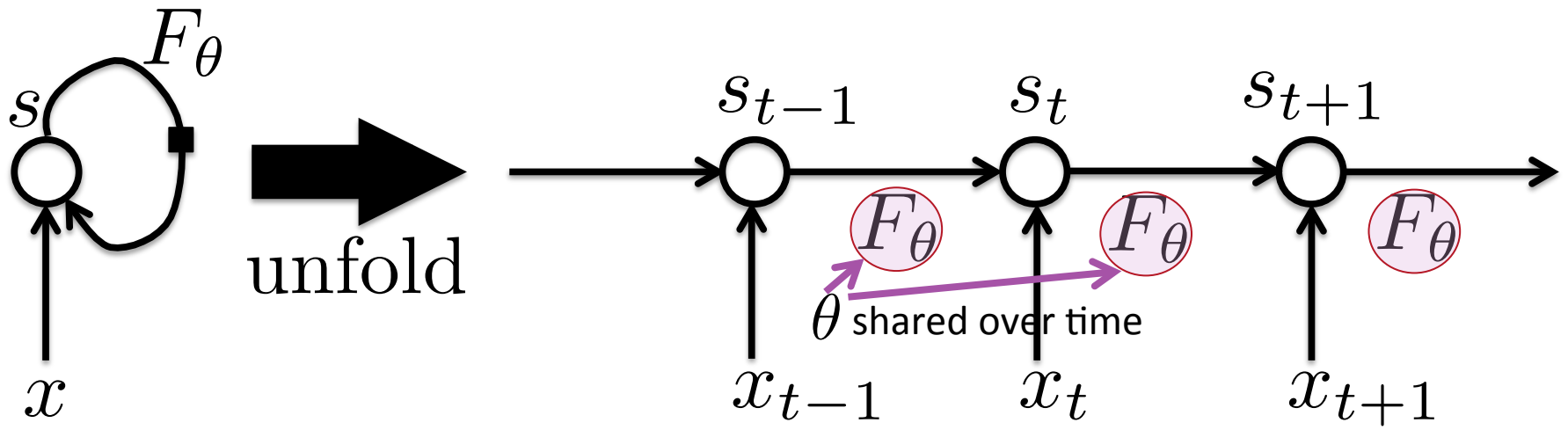
CIFAR | ICRA

MILA

Recurrent Neural Networks

- Selectively summarize an input sequence in a fixed-size state vector via a recursive update

$$s_t = F_\theta(s_{t-1}, x_t)$$



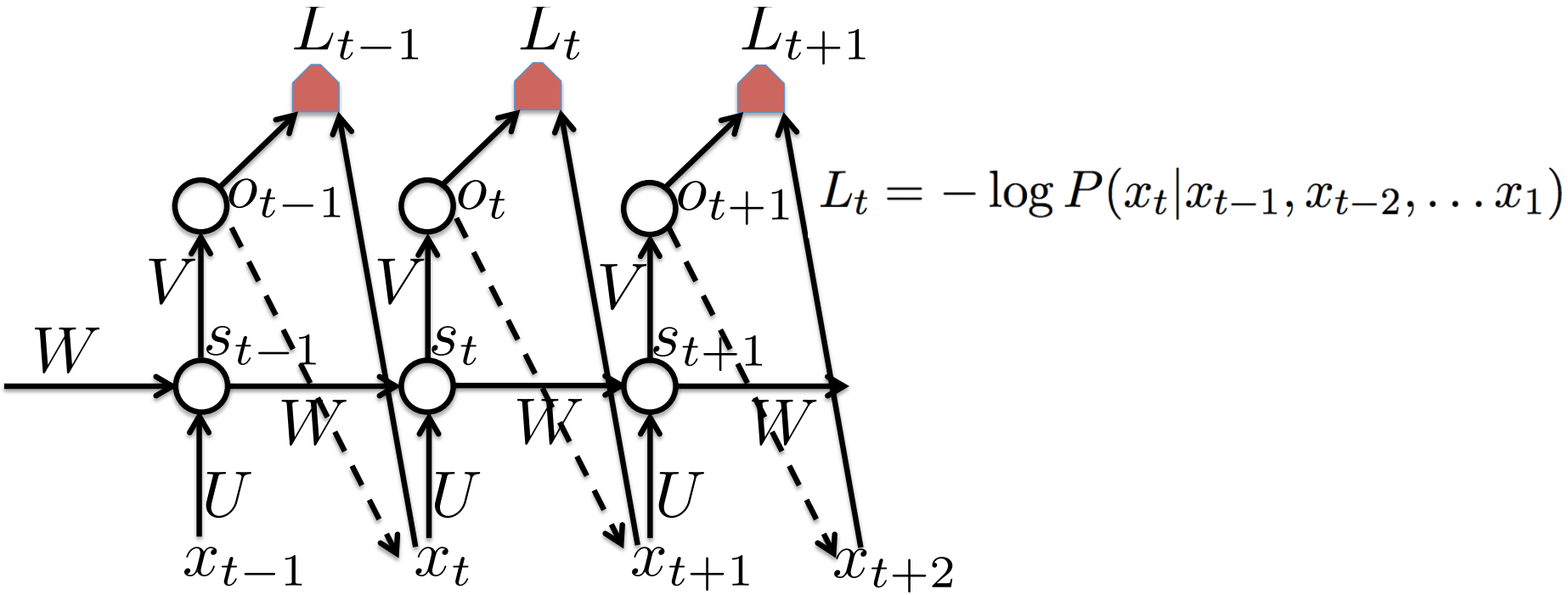
$$s_t = G_t(x_t, x_{t-1}, x_{t-2}, \ldots, x_2, x_1)$$

➔ **Generalizes naturally to new lengths not seen during training**

Generative RNNs

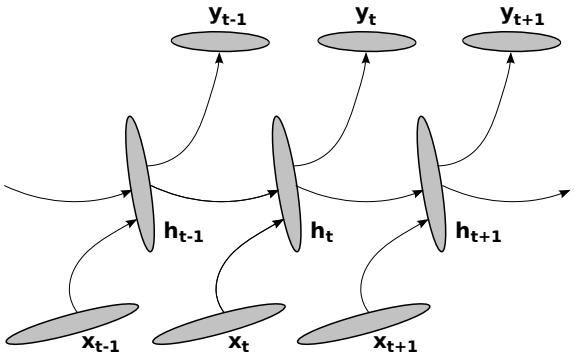
- An RNN can represent a fully-connected **directed generative model**: every variable predicted from all previous ones.

$$P(\mathbf{x}) = P(x_1, \ldots x_T) = \prod_{t=1}^{T} P(x_t | x_{t-1}, x_{t-2}, \ldots x_1)$$



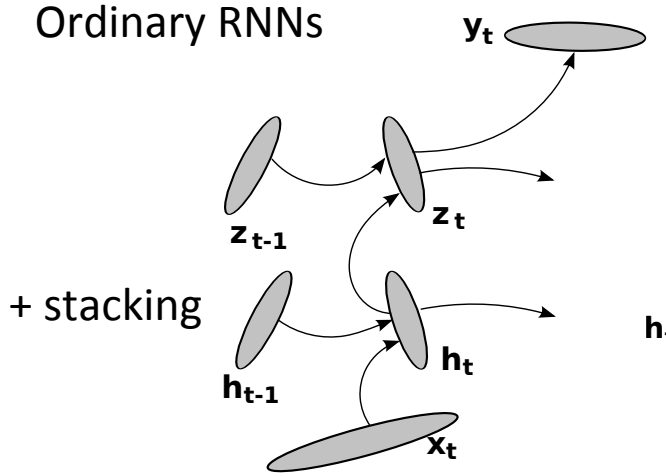
Increasing the Expressive Power of RNNs with more Depth

- ICLR 2014, *How to construct deep recurrent neural networks*

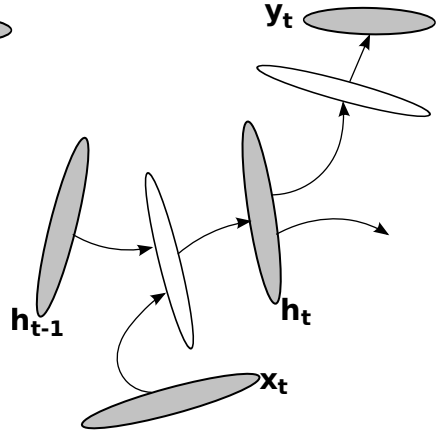


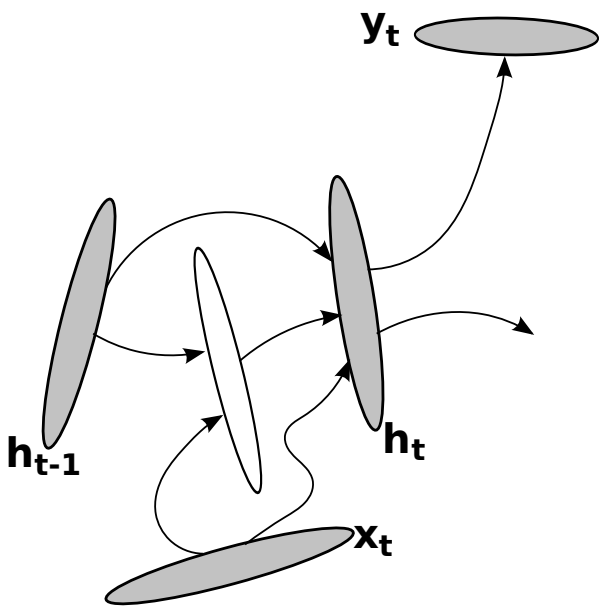
Ordinary RNNs

+ deep hid-to-out
+ deep hid-to-hid
+deep in-to-hid



+ stacking

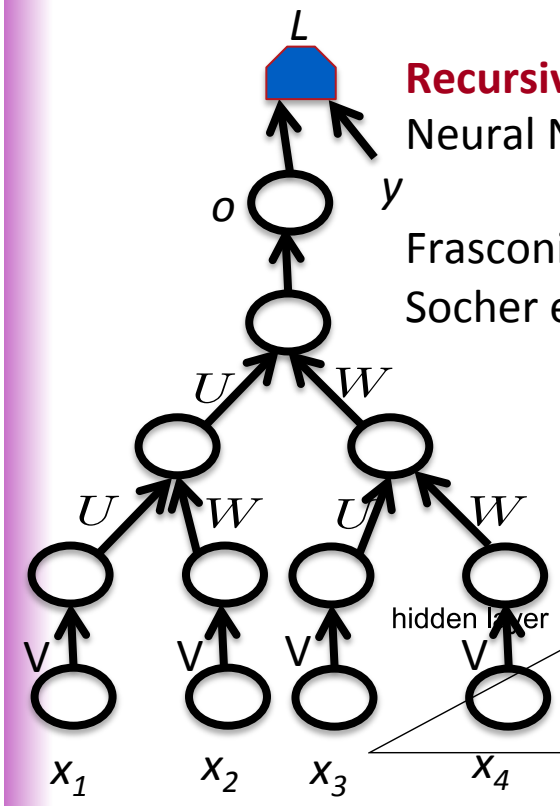




+ skip connections for creating shorter paths

Bidirectional RNNs, Recursive Nets, Multidimensional RNNs, etc.

- The unfolded architecture needs not be a straight chain

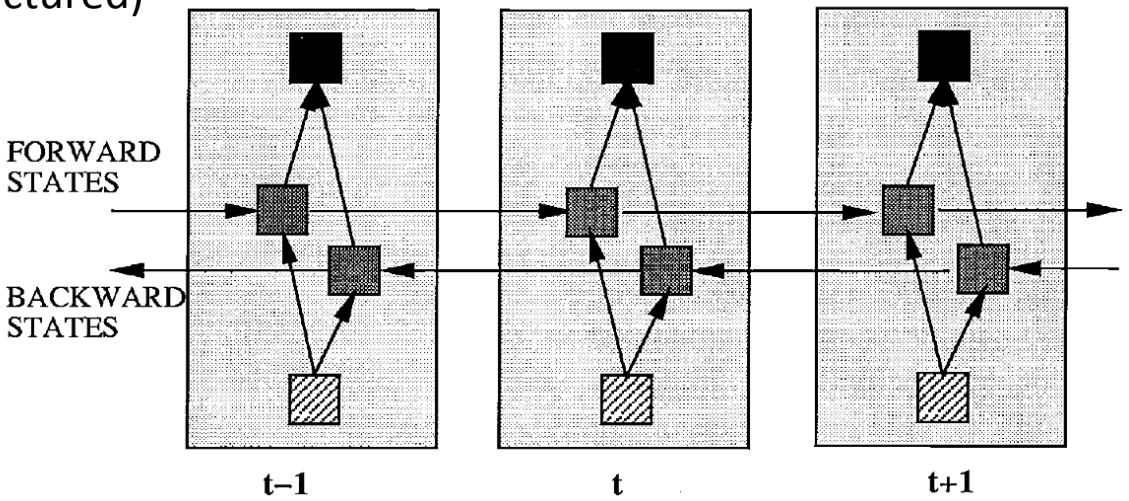


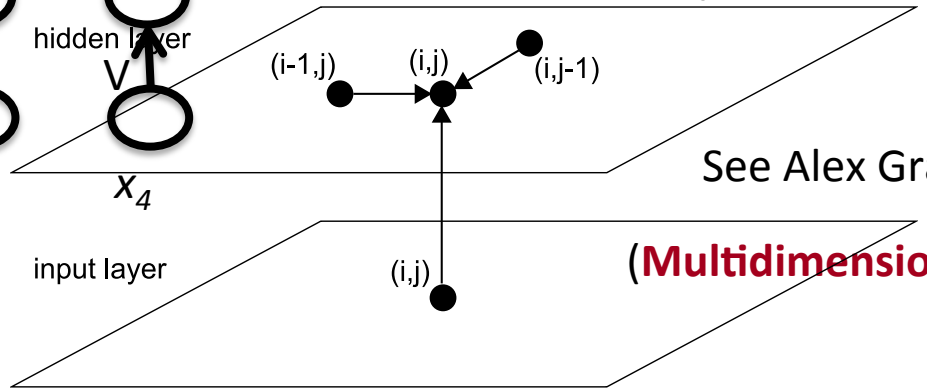
Recursive (tree-structured) Neural Nets:

Frasconi et al 97

Socher et al 2011

Bidirectional RNNs (Schuster and Paliwal, 1997)





See Alex Graves's work, e.g., 2012

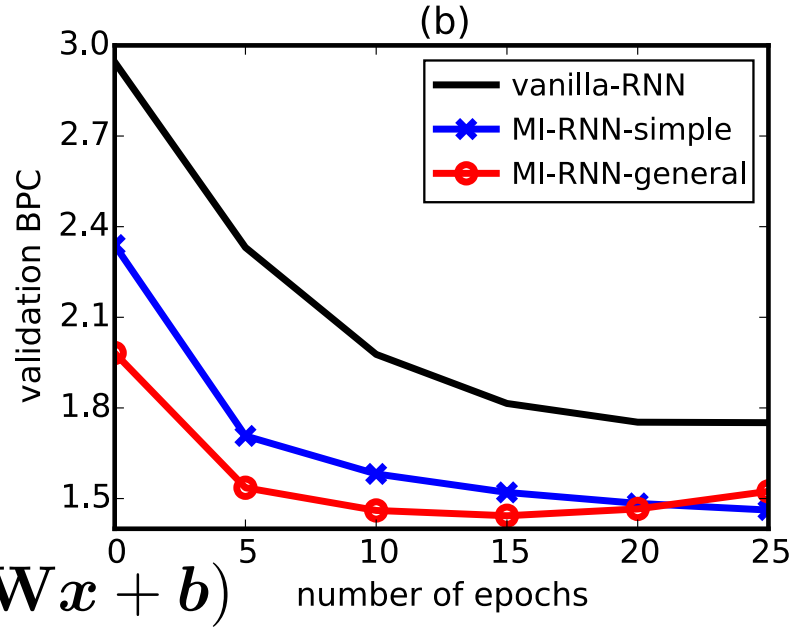
(**Multidimensional** RNNs, Graves et al 2007)

Multiplicative Interactions

(Wu et al, 2016, arXiv: 1606.06630, NIPS2016)

- Multiplicative Integration RNNs:
  - Replace $\phi(\mathbf{W}\boldsymbol{x} + \mathbf{U}\boldsymbol{z} + \mathbf{b})$
  - By $\phi(\mathbf{W}\boldsymbol{x} \odot \mathbf{U}\boldsymbol{z} + \mathbf{b})$
  - Or more general:

$$\phi(\boldsymbol{\alpha} \odot \mathbf{W}\boldsymbol{x} \odot \mathbf{U}\boldsymbol{z} + \boldsymbol{\beta}_1 \odot \mathbf{U}\boldsymbol{z} + \boldsymbol{\beta}_2 \odot \mathbf{W}\boldsymbol{x} + \boldsymbol{b})$$



Experiments on speech recognition, but also on language modeling and learning word embeddings.

WSJ Corpus	CER	WER
DRNN+CTCbeamsearch [17]	10.0	14.1
Encoder-Decoder [18]	6.4	9.3
LSTM+CTCbeamsearch [19]	9.2	8.7
Eesen [20]	-	**7.3**
LSTM+CTC+WFST (ours)	6.5	8.7
MI-LSTM+CTC+WFST (ours)	**6.0**	8.2

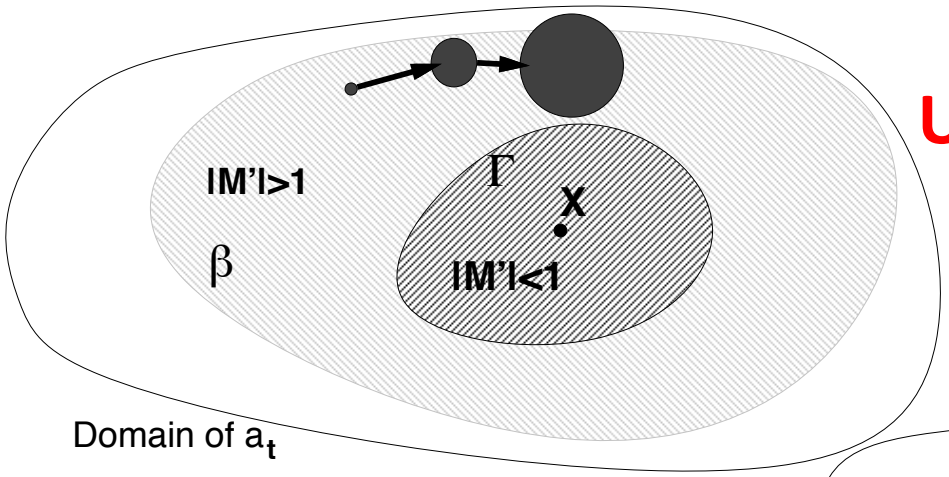
Learning Long-Term Dependencies with Gradient Descent is Difficult



Y. Bengio, P. Simard & P. Frasconi, IEEE Trans. Neural Nets, **1994**

Robustly storing 1 bit in the presence of bounded noise

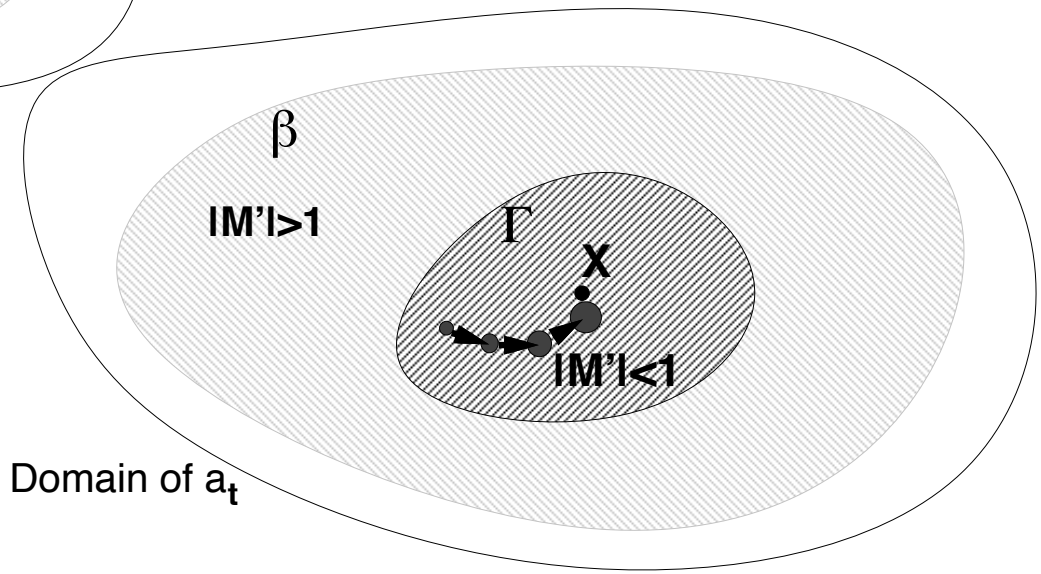
- With spectral radius > 1, noise can kick state out of attractor



UNSTABLE

- Not so with radius<1

CONTRACTIVE
→ STABLE



Storing Reliably → Vanishing gradients

- Reliably storing bits of information requires spectral radius<1
- The product of T matrices whose spectral radius is < 1 is a matrix whose spectral radius converges to 0 at exponential rate in T

$$L = L(s_T(s_{T-1}(\dots s_{t+1}(s_t, \dots))))$$

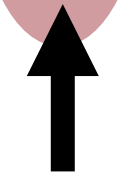
$$\frac{\partial L}{\partial s_t} = \frac{\partial L}{\partial s_T}\frac{\partial s_T}{\partial s_{T-1}}\cdots\frac{\partial s_{t+1}}{\partial s_t}$$

- If spectral radius of Jacobian is < 1 → propagated gradients vanish

Why it hurts gradient-based learning

- Long-term dependencies get a weight that is exponentially smaller (in T) compared to short-term dependencies

$$\frac{\partial C_t}{\partial W} = \sum_{\tau \leq t} \frac{\partial C_t}{\partial a_\tau} \frac{\partial a_\tau}{\partial W} = \sum_{\tau \leq t} \frac{\partial C_t}{\partial a_t} \frac{\partial a_t}{\partial a_\tau} \frac{\partial a_\tau}{\partial W}$$



Becomes exponentially smaller for longer time differences, when spectral radius < 1

To store information robustly the dynamics must be contractive

- The RNN gradient is a product of Jacobian matrices, each associated with a step in the forward computation. To store information robustly in a finite-dimensional state, the dynamics must be contractive [Bengio et al 1994].

$$L = L(s_T(s_{T-1}(\ldots s_{t+1}(s_t, \ldots))))$$

$$\frac{\partial L}{\partial s_t} = \frac{\partial L}{\partial s_T}\frac{\partial s_T}{\partial s_{T-1}}\cdots\frac{\partial s_{t+1}}{\partial s_t}$$

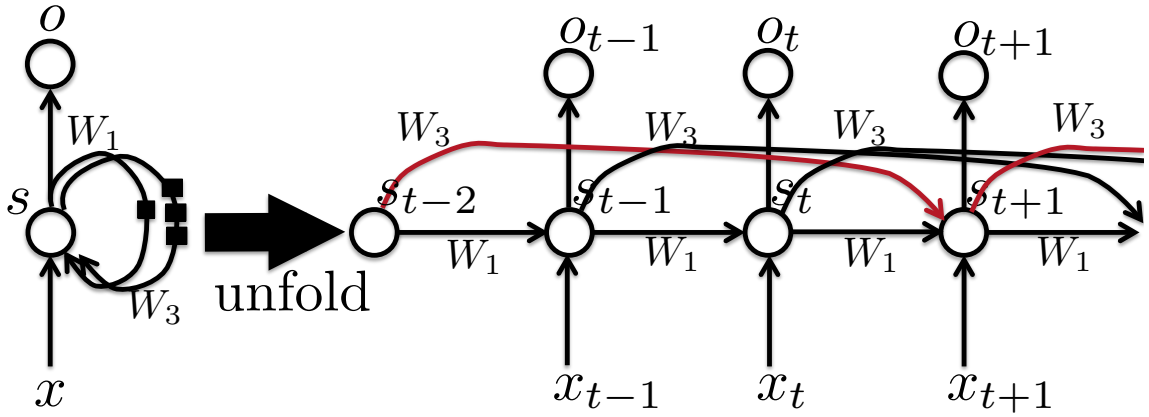
Storing bits robustly requires e-values<1

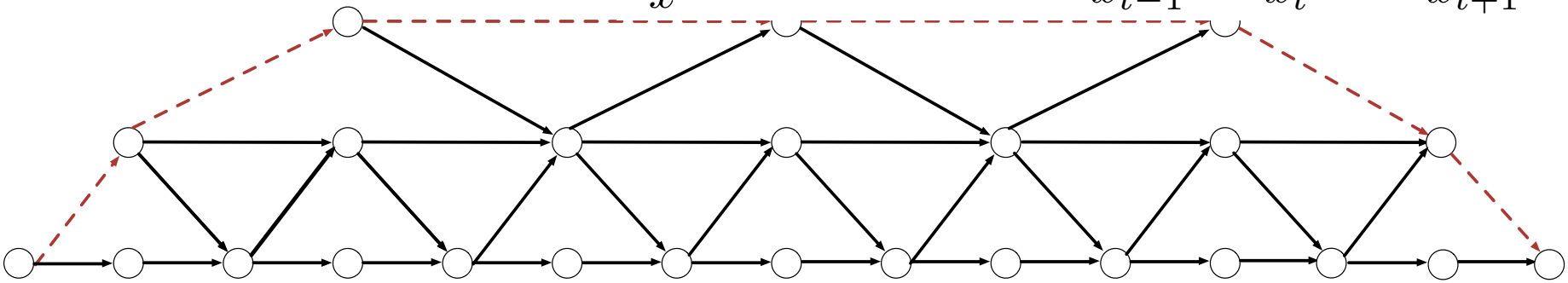
- Problems:
  - e-values of Jacobians > 1 → *gradients explode*
  - **or e-values < 1 → *gradients shrink & vanish***
  - or random → variance grows exponentially

→ **Gradient clipping** *(Pascanu, Mikolov & Bengio ICML2013)*

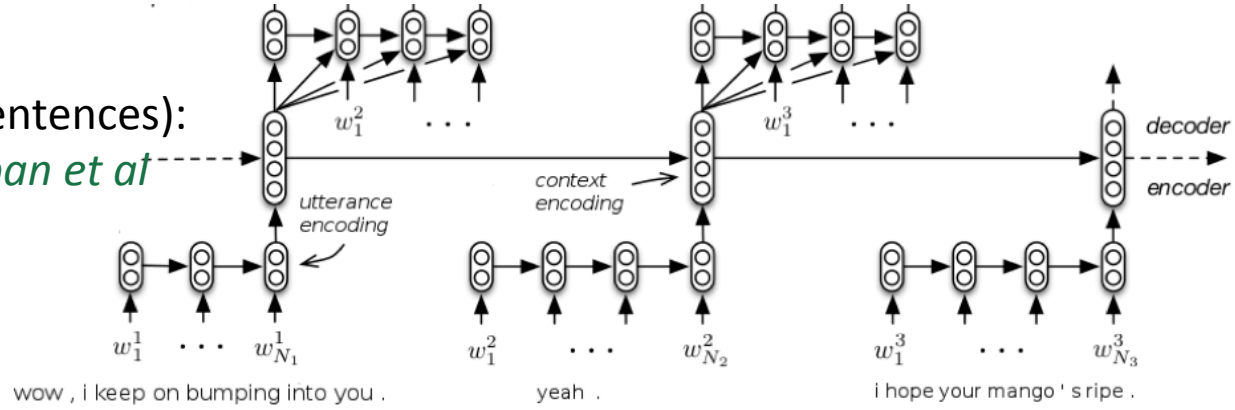
Delays & Hierarchies to Reach Farther

- Delays and multiple time scales, *Elhihi & Bengio NIPS 1995, Koutnik et al ICML 2014*
- *How to do this right?*
- *How to automatically and adaptively do it?*





Hierarchical RNNs (words / sentences):
Sordoni et al CIKM 2015, Serban et al AAAI 2016

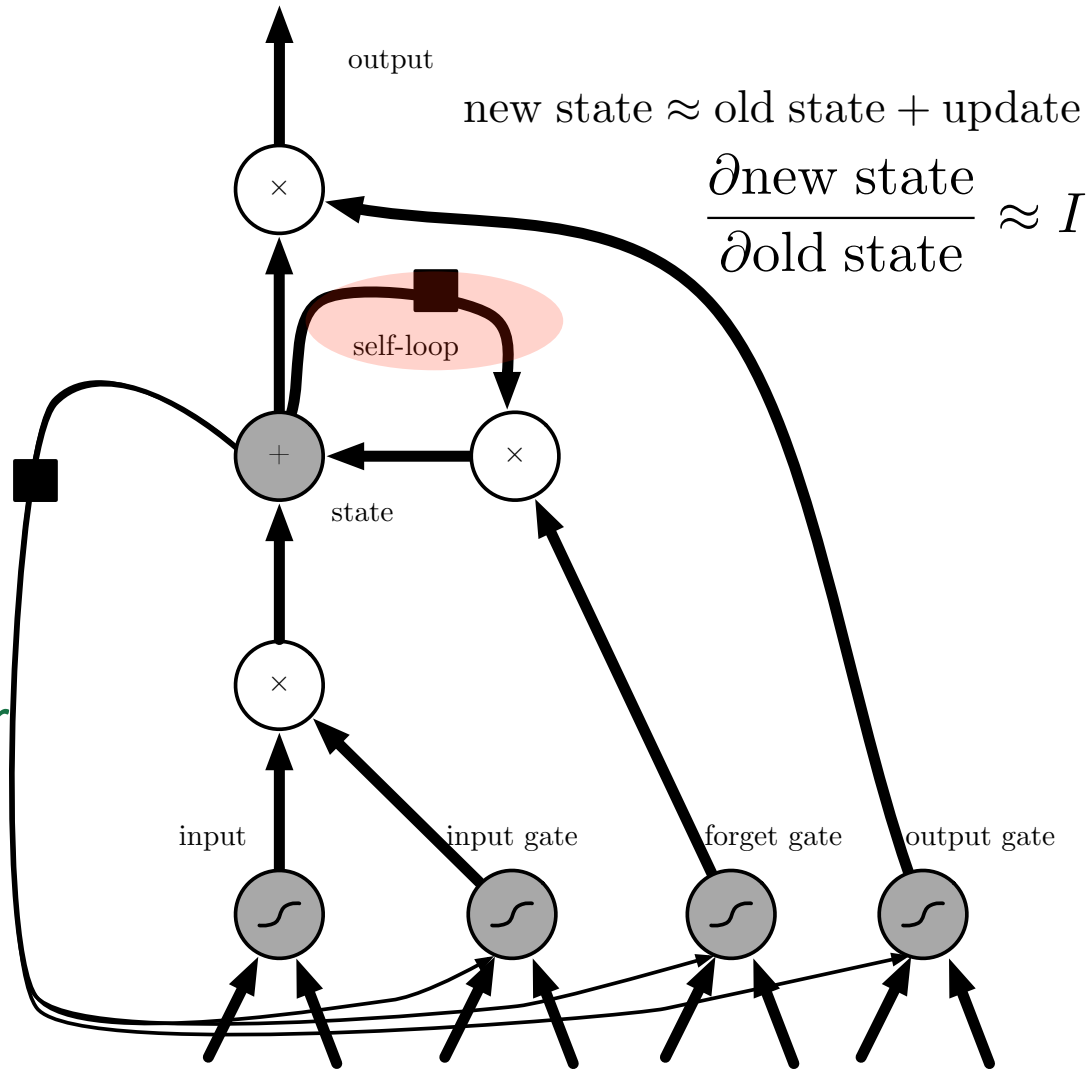


Fighting the vanishing gradient: LSTM & GRU

(Hochreiter 1991); first version of the LSTM, called Neural Long-Term Storage with self-loop

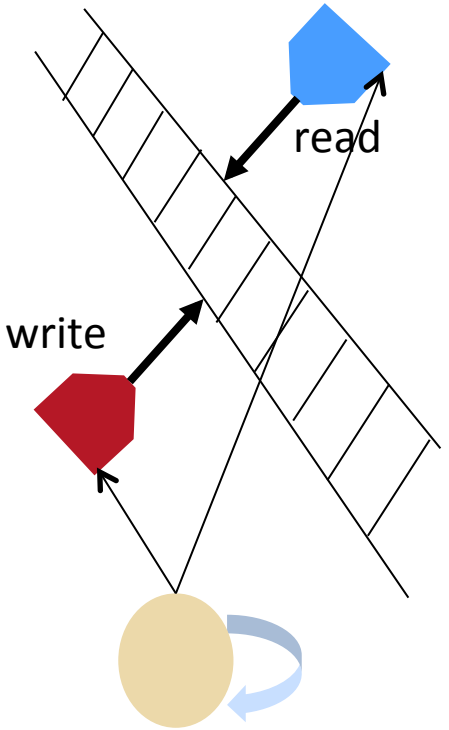
- Create a path where gradients can flow for longer with a self-loop
- Corresponds to an eigenvalue of Jacobian slightly less than 1
- LSTM is now **heavily used** *(Hochreiter & Schmidhuber 1997)*
- GRU light-weight version *(Cho et al 2014)*

LSTM: (Hochreiter & Schmidhuber 1997)



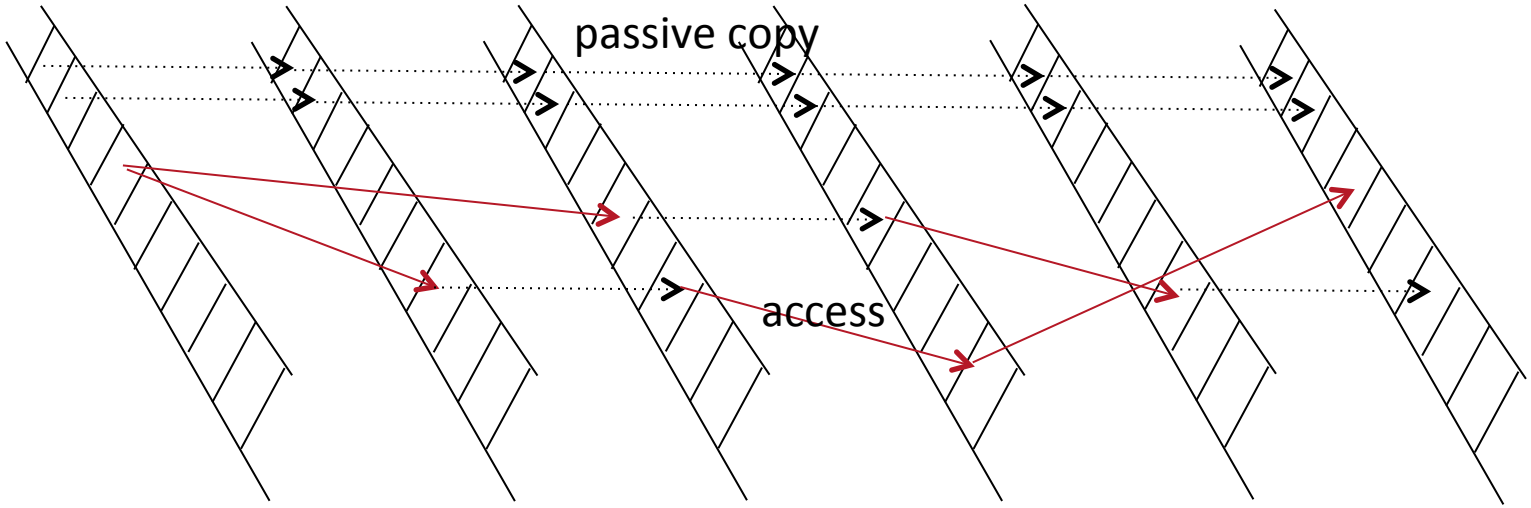
Fast Forward 20 years: Attention Mechanisms for Memory Access

- Neural Turing Machines *(Graves et al 2014)*
- and Memory Networks *(Weston et al 2014)*
- Use a content-based attention mechanism *(Bahdanau et al 2014)* to control the read and write access into a memory
- The attention mechanism outputs a softmax over memory locations

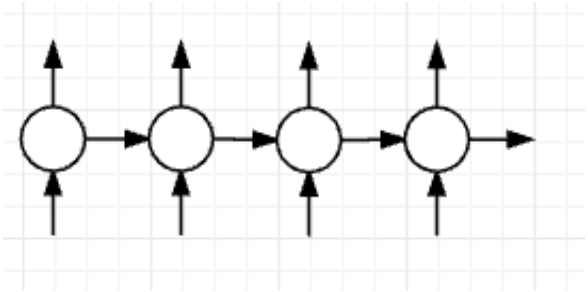


Large Memory Networks: Sparse Access Memory for Long-Term Dependencies

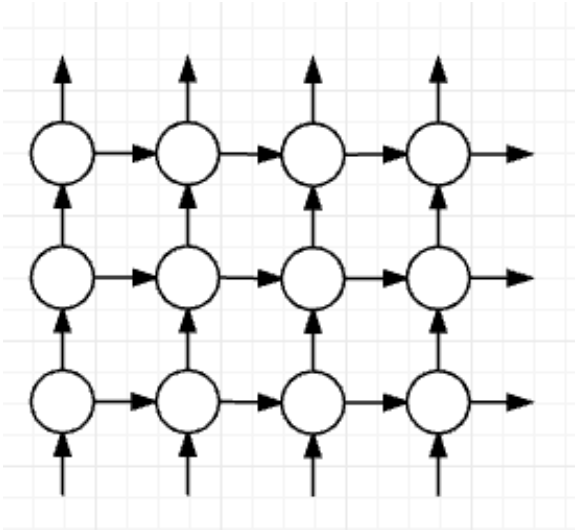
- Memory = part of the state
- Memory-based networks are special RNNs
- A mental state stored in an external memory can stay for arbitrarily long durations, until it is overwritten (partially or not)
- Forgetting = vanishing gradient.
- Memory = **higher-dimensional state**, avoiding or reducing the need for forgetting/vanishing



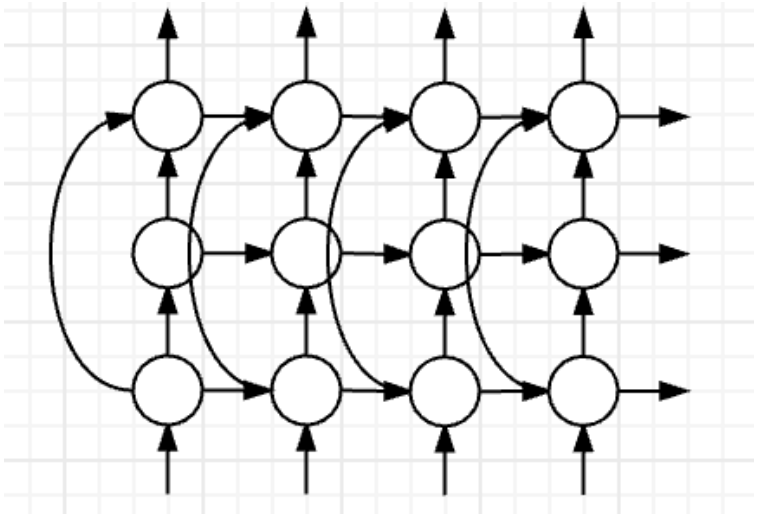
What RNN architectures do you use in your research?



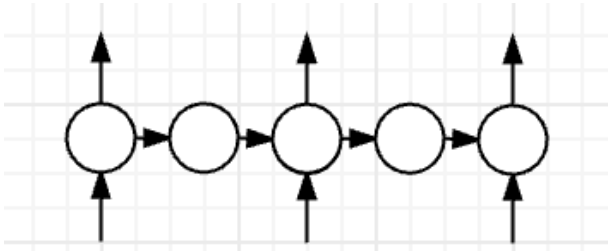
Simple RNN



Stacked RNN



Skip connections
Graves et al. 2013

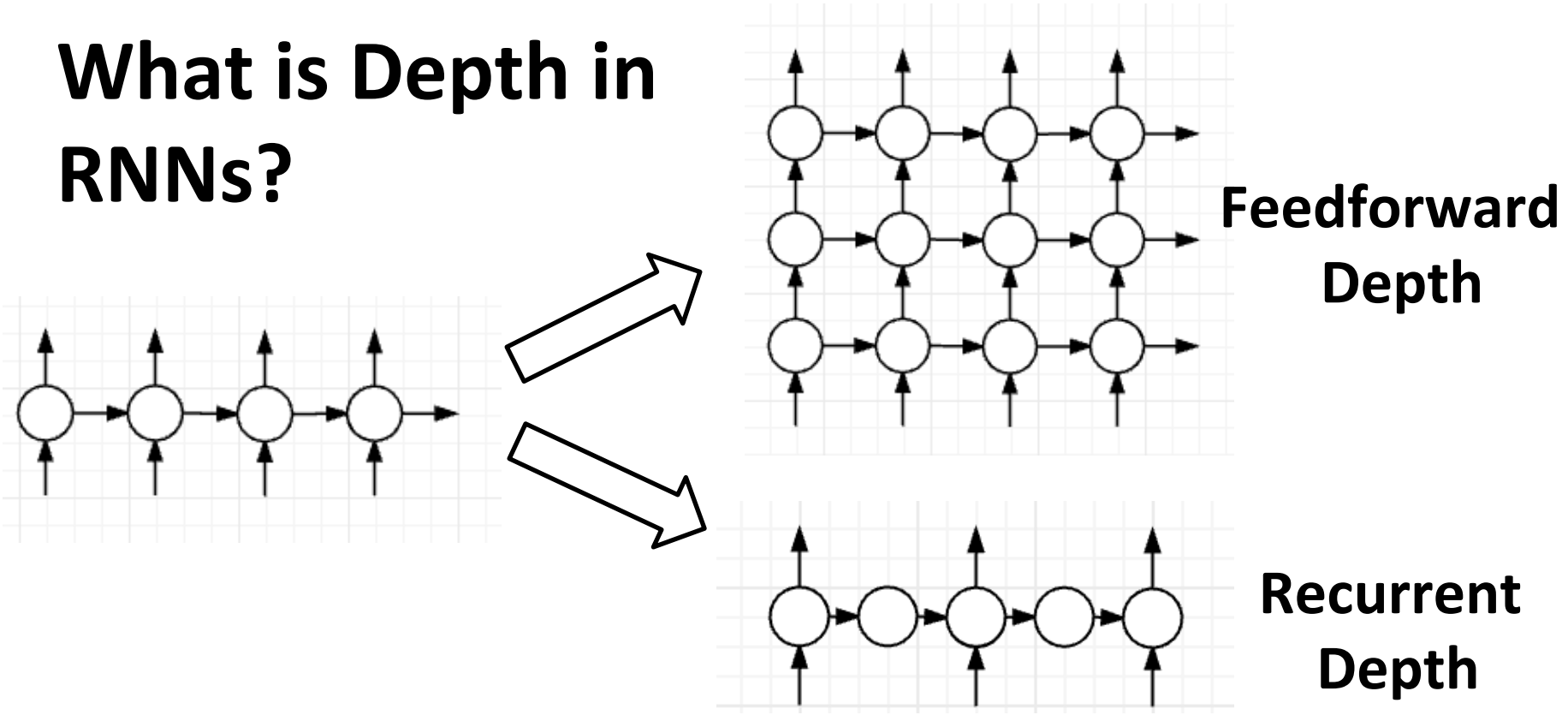


"Deep" RNN
Pascanu et al. 2013

MILA

Depth in RNNs

What is Depth in RNNs?



(Architectural Complexity Measures of Recurrent Neural Networks Zhang et al 2016, arXiv:1602.08210)

MILA

Depth in RNNs

- Recurrent depth :

the length of the *longest* path over time, divide by time

$$d_r = \lim_{n \to +\infty} \frac{\mathfrak{D}_i(n)}{n} = \max_{\vartheta \in C(\mathcal{G}_c)} \frac{l(\vartheta)}{\sigma_s(\vartheta)}$$

- Feedforward depth :

consider the input-output nonlinearities while eliminating the effect of recurrent depth

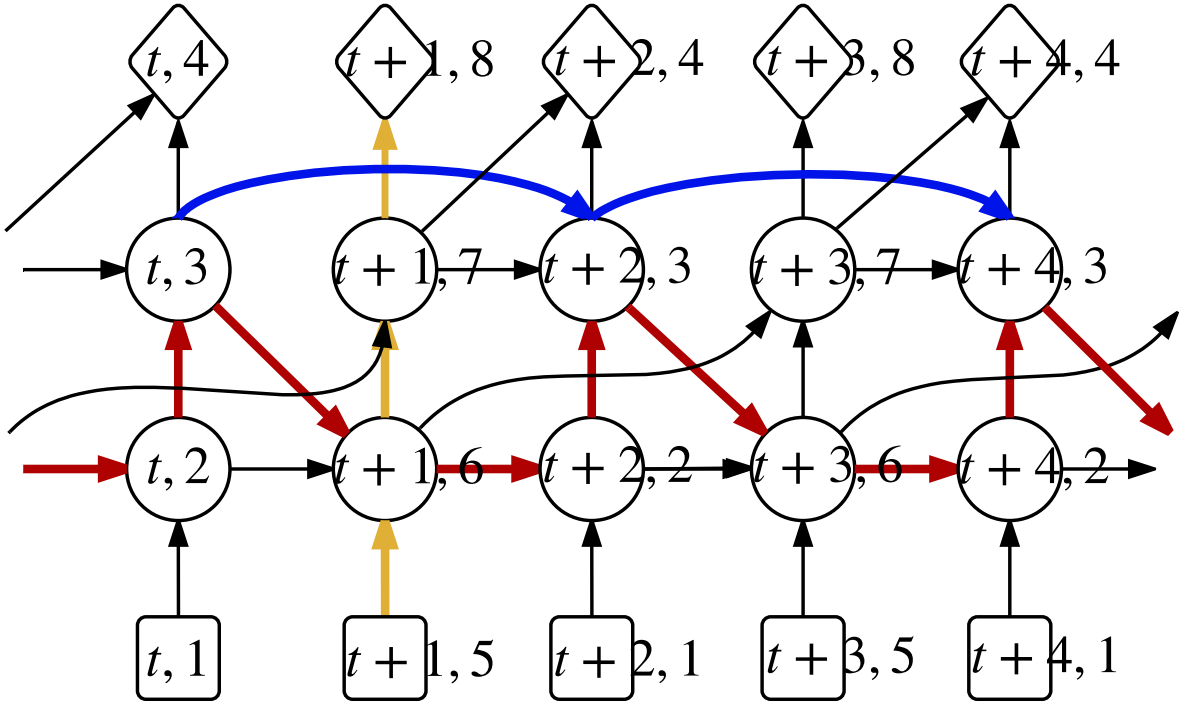
$$d_f = \sup_{i,n \in \mathbb{Z}} \mathfrak{D}_i^*(n) - n \cdot d_r$$

MILA

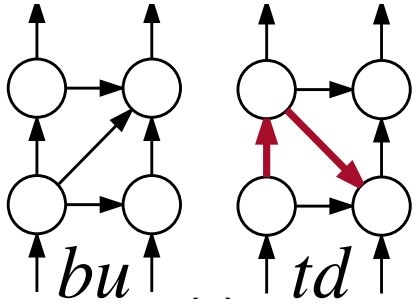
Designing the RNN Architecture

(Architectural Complexity Measures of Recurrent Neural Networks Zhang et al 2016, arXiv:1602.08210)

- **Recurrent depth**: max path length divided by sequence length
- **Feedforward depth**: max length from input to nearest output
- **Skip coefficient**: shortest path length divided sequence length



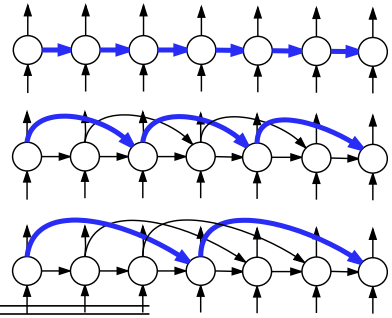
It makes a difference



- Impact of change in recurrent depth

DATASET	MODELS\ARCHS	*sh*	*st*	*bu*	*td*
PennTreebank	*tanh* RNN	1.54	1.59	1.54	**1.49**
text8	*tanh* RNN-SMALL	1.80	1.82	1.80	**1.77**
	tanh RNN-LARGE	1.69	1.67	1.64	**1.59**
	LSTM-SMALL	1.65	1.66	1.65	**1.63**
	LSTM-LARGE	1.52	1.53	1.52	**1.49**

- Impact of change in skip coefficient



RNN(*tanh*)	s = 1	s = 5	s = 9	s = 13	s = 21
MNIST	34.9	46.9	74.9	85.4	**87.8**
	s = 1	s = 3	s = 5	s = 7	s = 9
*p*MNIST	49.8	79.1	84.3	**88.9**	88.0

LSTM	s = 1	s = 3	s = 5	s = 7	s = 9
MNIST	56.2	**87.2**	86.4	86.4	84.8
	s = 1	s = 3	s = 4	s = 5	s = 6
*p*MNIST	28.5	25.0	60.8	62.2	**65.9**

Model	MNIST	*p*MNIST
*i*RNN[25]	97.0	≈82.0
*u*RNN[24]	95.1	91.4
LSTM[24]	**98.2**	88.0
RNN(*tanh*)[25]	≈35.0	≈35.0
*s*tanh(s = 21, 11)	98.1	**94.0**

Architecture, s		(1), 1	(2), 1	(3), $\frac{k}{2}$	(4), k
MNIST	k = 17	39.5	39.4	54.2	**77.8**
	k = 21	39.5	39.9	69.6	**71.8**
*p*MNIST	k = 5	55.5	66.6	74.7	**81.2**
	k = 9	55.5	71.1	78.6	**86.9**

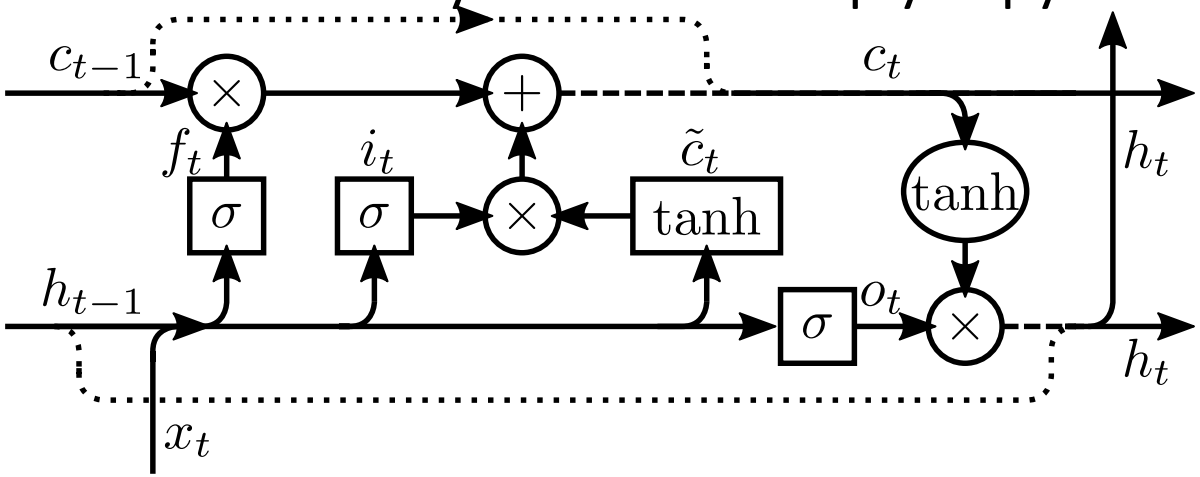
Table 2: Results for MNIST/*p*MNIST. **Top-left**: test accuracies with different s for *tanh* RNN. **Top-right**: test accuracies with different s for LSTM. **Bottom**: compared to previous results. **Bottom-right**: test accuracies for architectures (1), (2), (3) and (4) for *tanh* RNN.

Near-Orthogonality to Help Information Propagation

- Initialization to orthogonal recurrent W *(Saxe et al 2013, ICLR2014)*
- Unitary matrices: all e-values of matrix are 1 *(Arjowski, Amar & Bengio ICML 2016)*

$$\mathbf{W} = \mathbf{D}_3\mathbf{R}_2\mathcal{F}^{-1}\mathbf{D}_2\mathbf{\Pi}\mathbf{R}_1\mathcal{F}\mathbf{D}_1$$

- Zoneout: randomly choose to simply copy the state unchanged

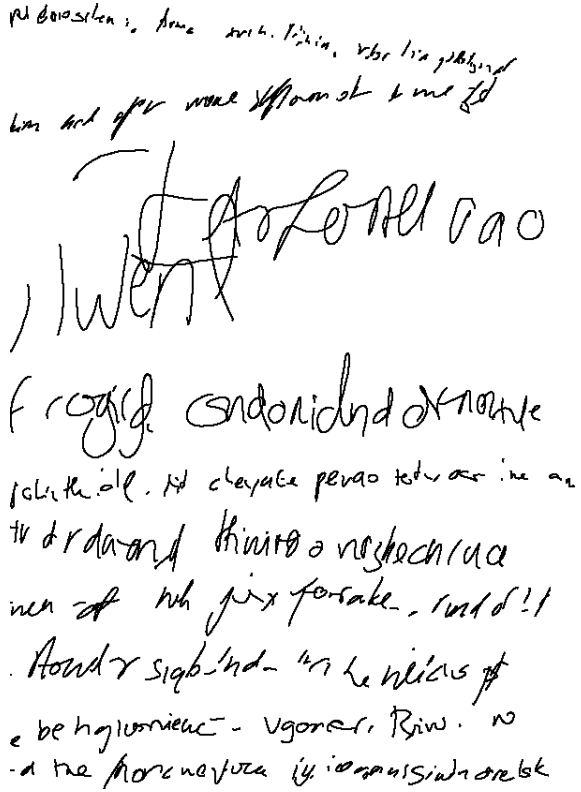


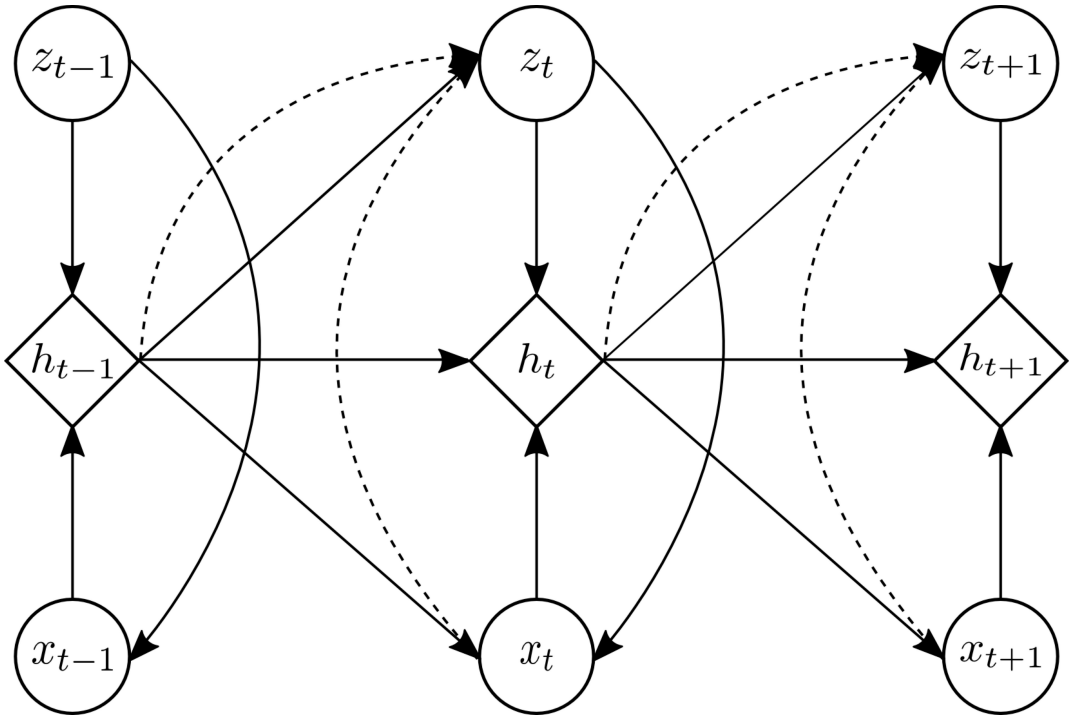
(Krueger et al 2016, submitted)

Variational Generative RNNs

Injecting higher-level variations / latent variables in RNNs

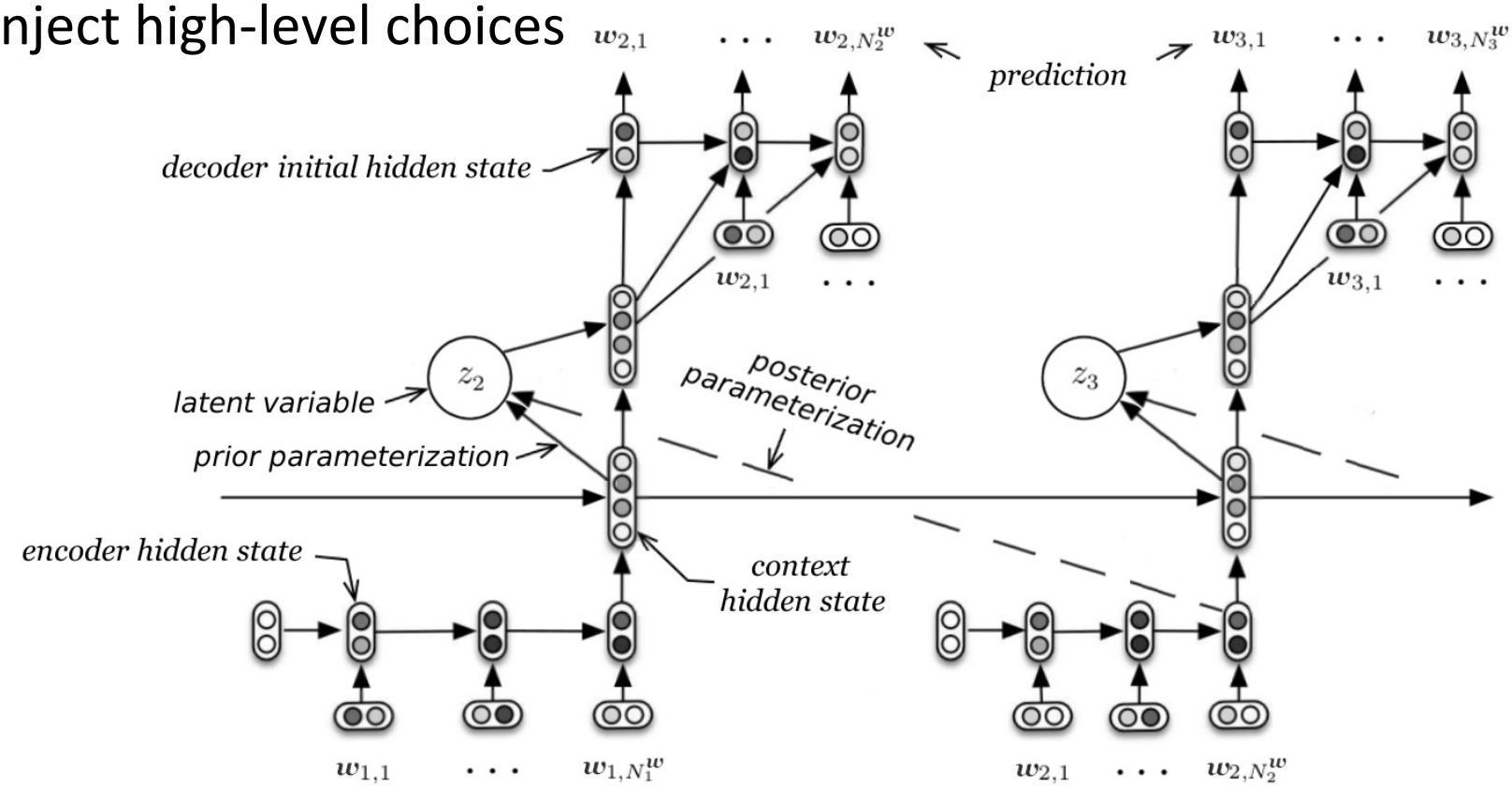
- *(Chung et al, NIPS'2015)*
- Regular RNNs have noise injected only in input space
- VRNNs also allow noise (latent variable) injected in top hidden layer; more « high-level » variability





Variational Hierarchical RNNs for Dialogue Generation (Serban et al 2016)

- Lower level = words of an utterance (turn of speech)
- Upper level = state of the dialogue
- Inject high-level choices

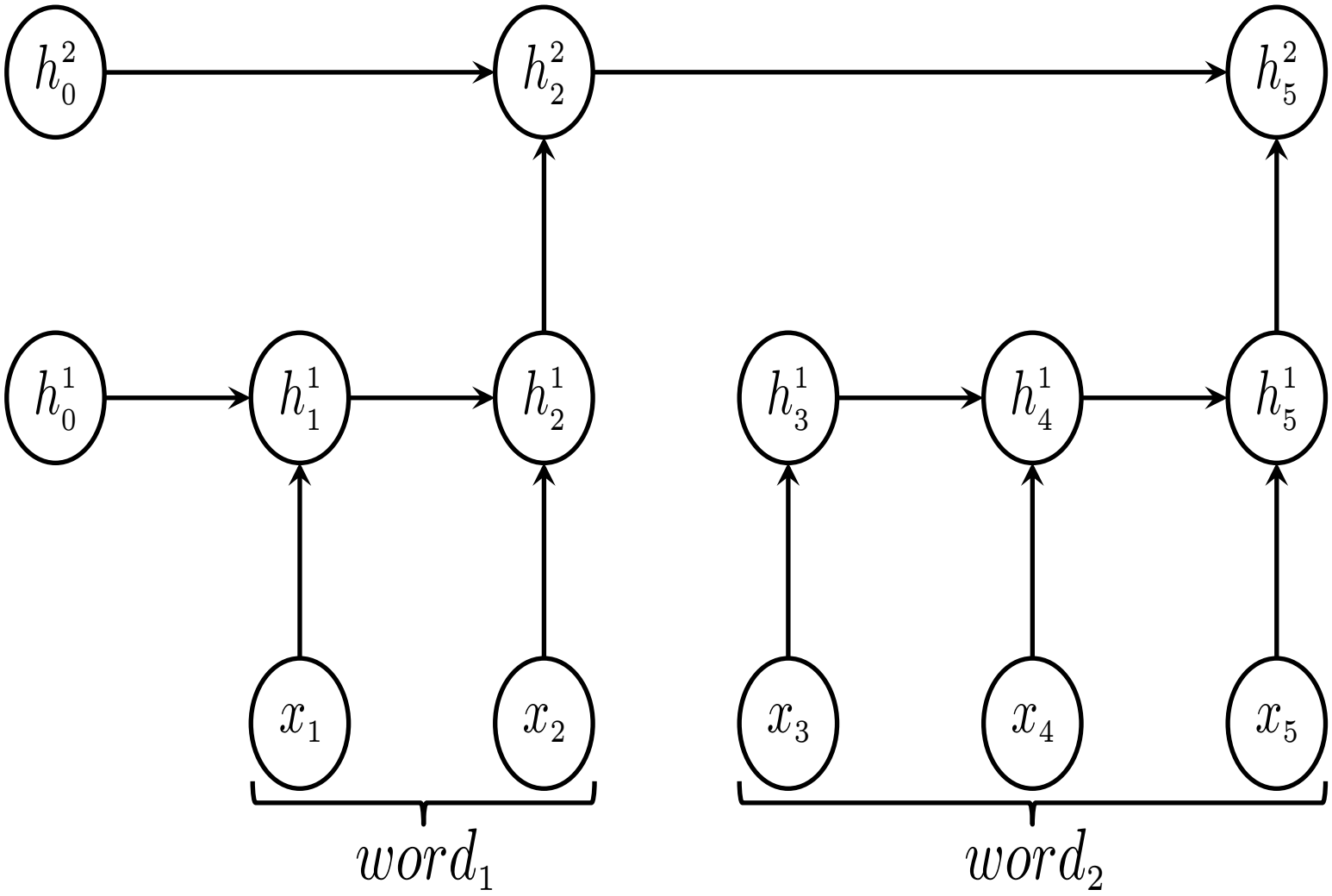


Hierarchical Multiscale Recurrent Neural Networks (Chung, Ahn & Bengio arXiv:1609.01704)

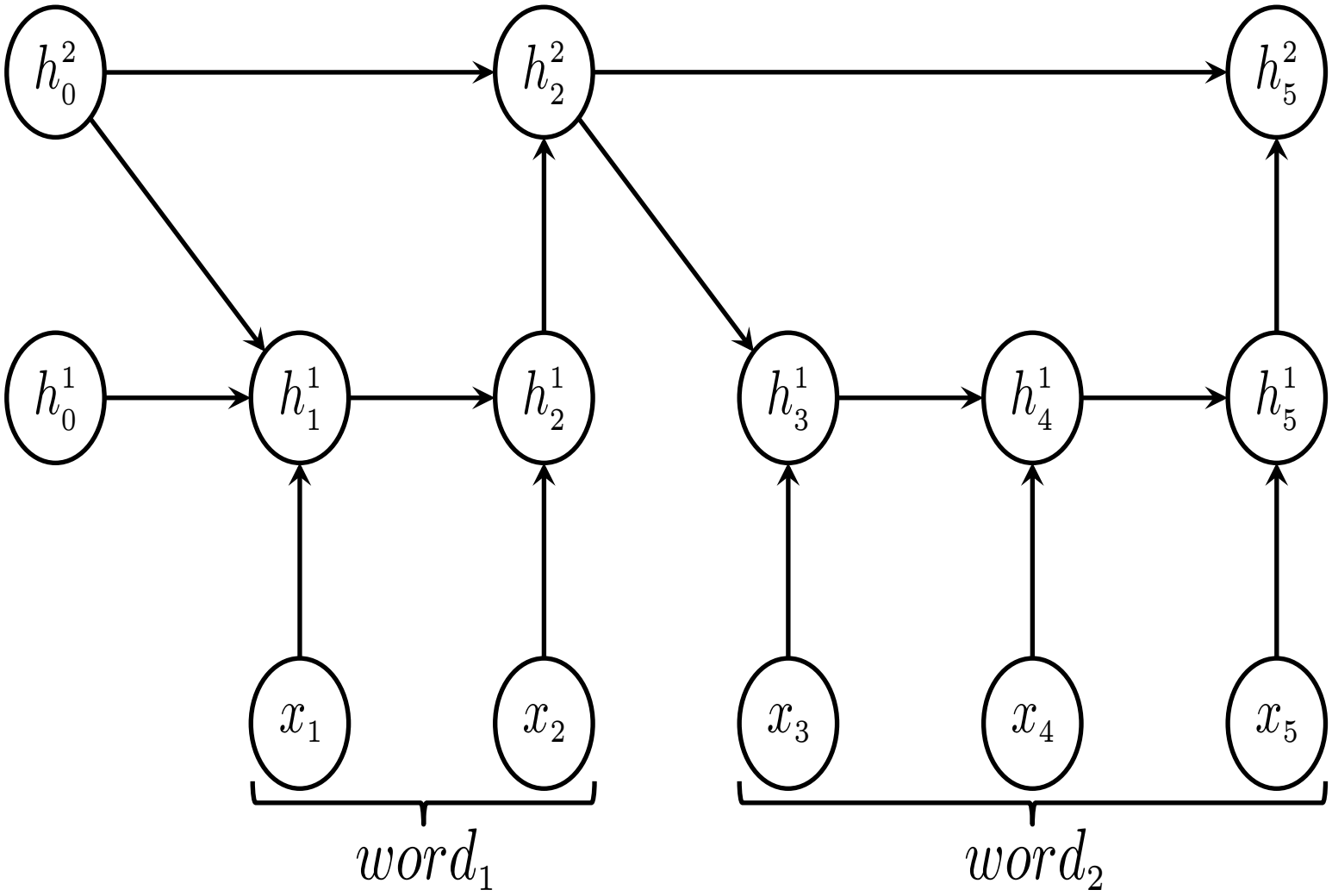
- How to learn to update higher-level states at the right points in time, corresponding to slower time scales, but not necessarily on a fixed (clockwork) schedule, rather an adaptive, dynamic one?

- If each level updates once every 10 steps of lower level then 3rd level updates once every 1000 raw steps!
  - Advantage for propagating gradients for long-term dependencies
  - Computational advantage (but need HARD decisions)

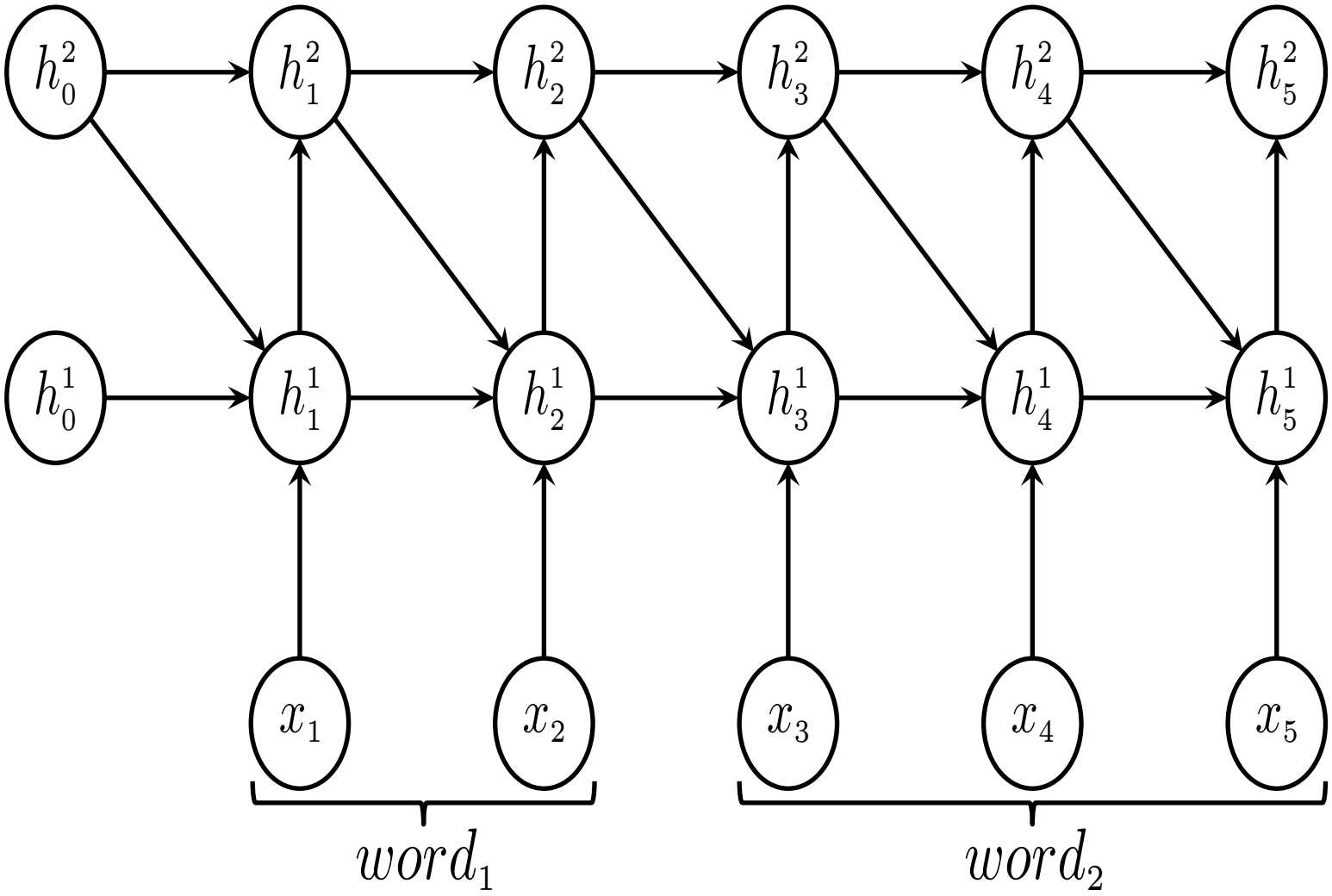
Motivation



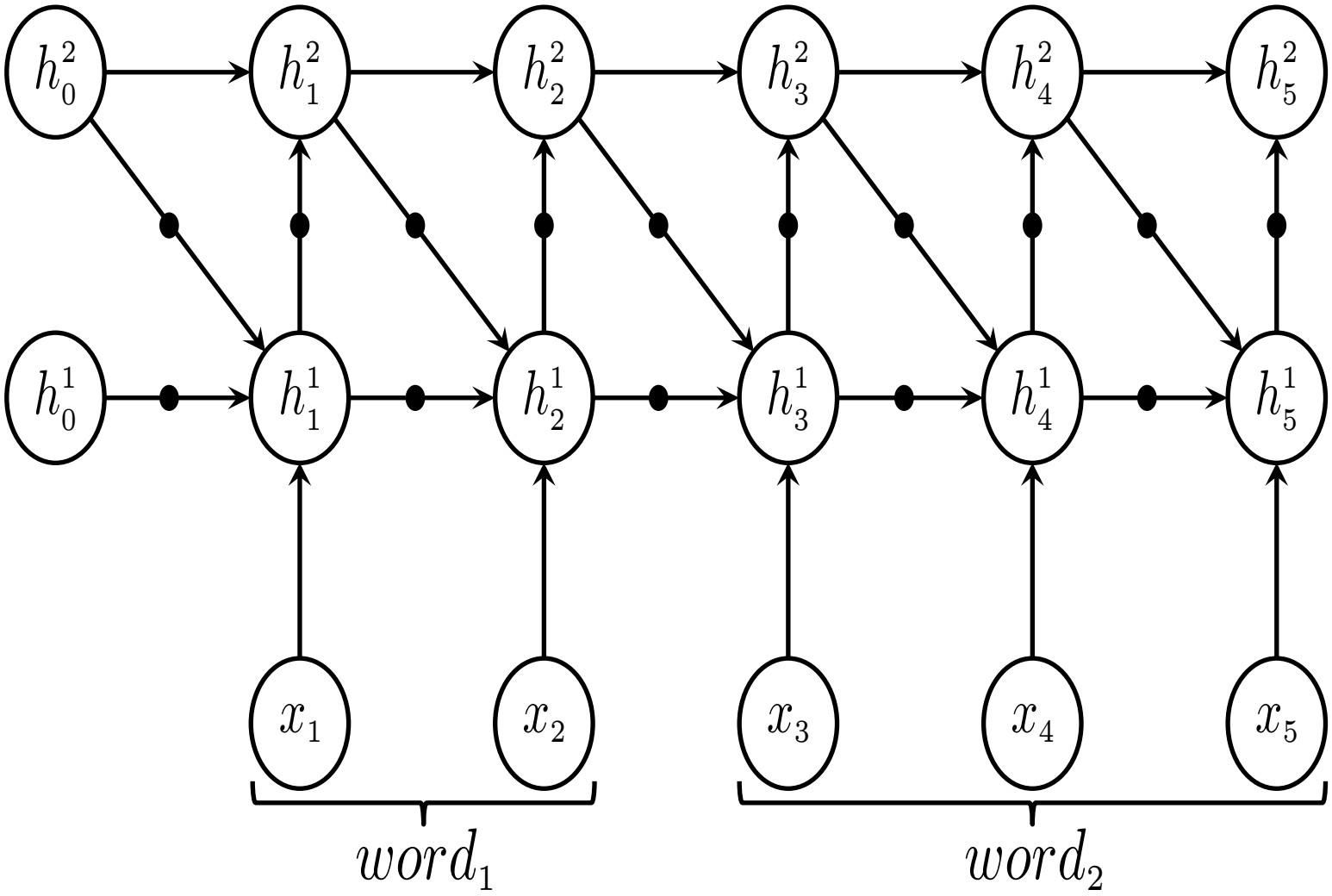
Motivation



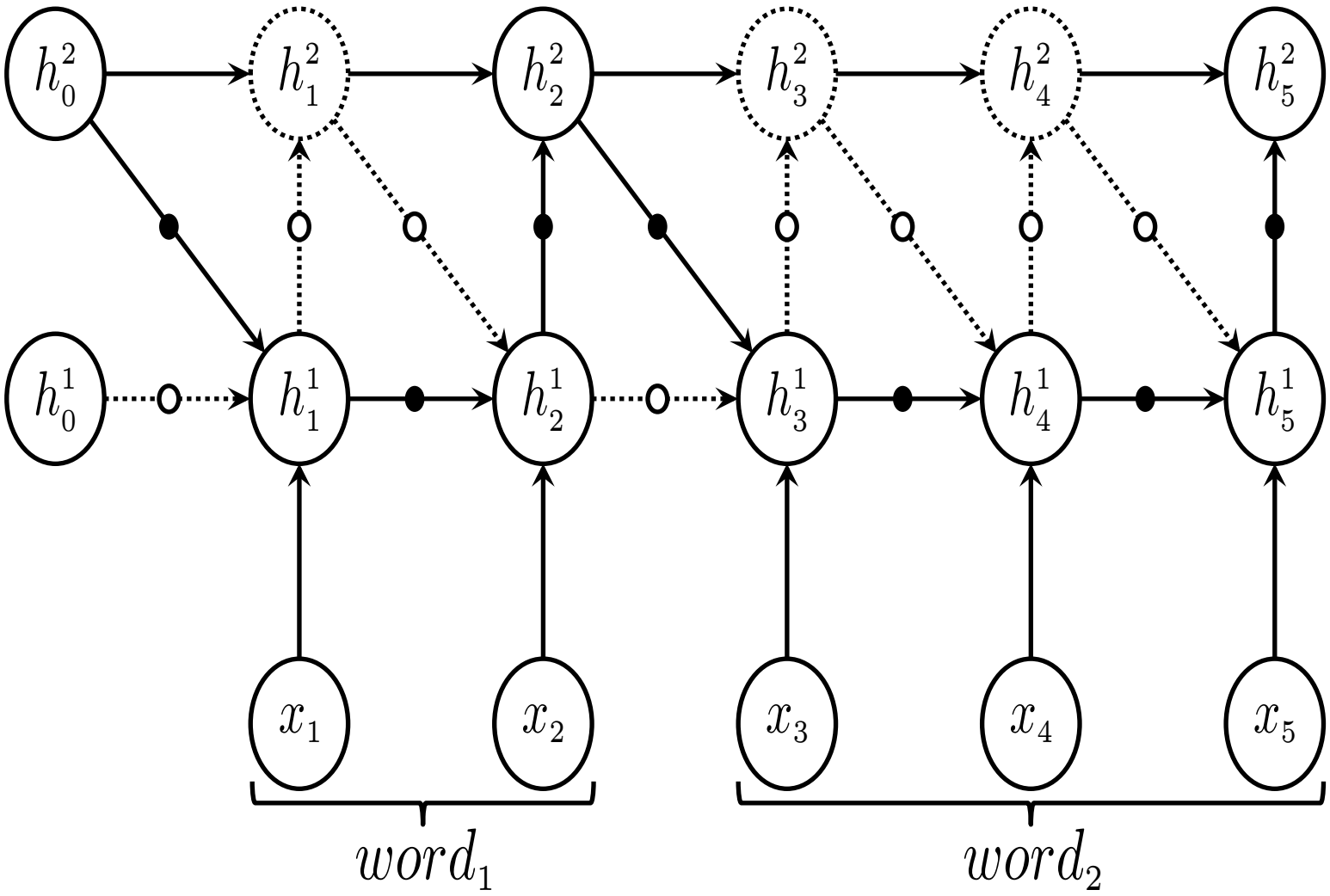
Motivation


h_0^2
h_1^2
h_2^2
h_3^2
h_4^2
h_5^2
h_0^1
h_1^1
h_2^1
h_3^1
h_4^1
h_5^1
x_1
x_2
x_3
x_4
x_5
$word_1$
$word_2$

Motivation

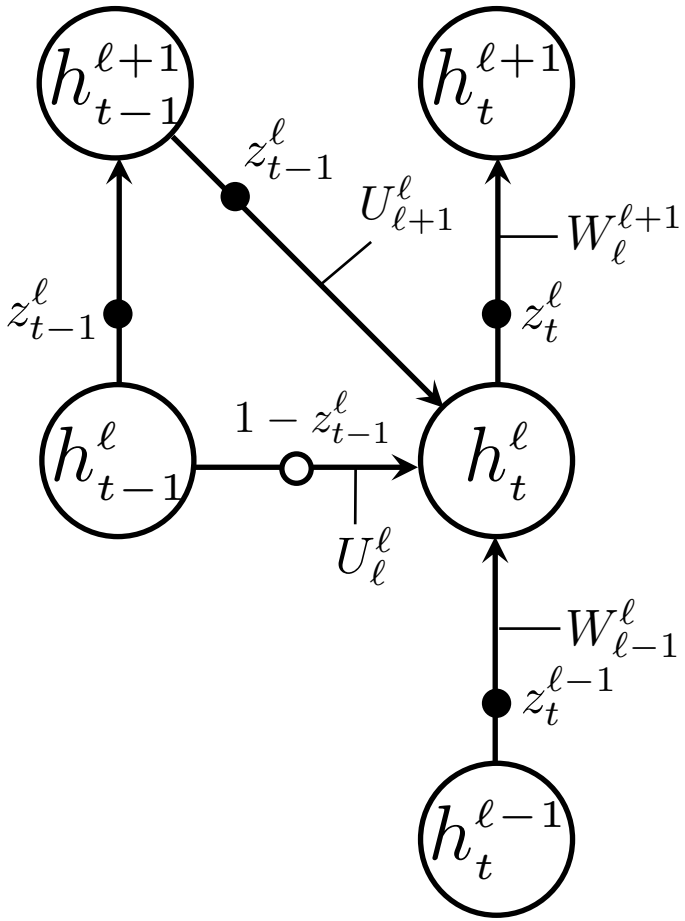

h_0^2
h_1^2
h_2^2
h_3^2
h_4^2
h_5^2
h_0^1
h_1^1
h_2^1
h_3^1
h_4^1
h_5^1
x_1
x_2
x_3
x_4
x_5
$word_1$
$word_2$

Motivation


h_0^2
h_1^2
h_2^2
h_3^2
h_4^2
h_5^2
h_0^1
h_1^1
h_2^1
h_3^1
h_4^1
h_5^1
x_1
x_2
x_3
x_4
x_5
$word_1$
$word_2$

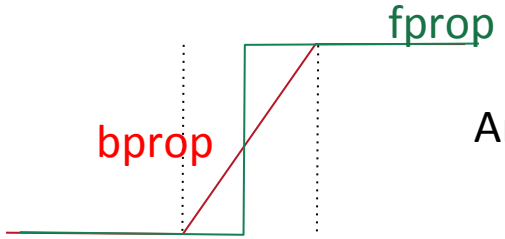
Hierarchical Multiscale Recurrent Neural Networks: Gating Mechanism

- Decide to copy the current level state vs updating it from lower-level state
- If updating it, restart the lower level RNN, conditioned on upper-level state
- Need to take a binary decision in order to actually get a pure copy and avoid leakage of gradients
- Need that binary decision to be stochastic to have a chance to learn it



How to backprop through stochastic binary decisions?

- REINFORCE: correlate the action with the reward, very high variance estimator
- **Straight-through estimator** (Hinton 2012, Bengio, Leonard & Courville arXiv 2013, Courbariaux et al NIPS 2015)
  - Heuristic but worked very well in several settings



Anneal the slope during training

- MuProp (Gu et al ICLR 2016, arXiv 1511.05176) combines a soft decision with a hard one, unbiased baseline of REINFORCE uses the gradient of the loss wrt soft decision
- VIMCO (Mnih & Rezende ICLR 2016, arXiv:1602.06725)

Hierarchical Multiscale Recurrent Neural Networks: Results (Chung, Ahn & Bengio arXiv:1609.01704)

- Automatically segmenting so as to better predict the next character

- SF-LSTM is not comparable (changes parameters on the fly)

Text8	
Model	**BPC**
td-LSTM (Zhang *et al.*, 2016)	1.63
HF-MRNN (Mikolov *et al.*, 2012)	1.54
MI-RNN (Wu *et al.*, 2016)	1.52
Skipping-RNN (Pachitariu and Sahani, 2013)	1.48
MI-LSTM (Wu *et al.*, 2016)	1.44
Batch-normalized LSTM (Cooijmans *et al.*, 2016)	1.36
HM-LSTM	**1.30**

Hutter Prize Wikipedia	
Model	**BPC**
SF-LSTM (Rocki, 2016b)*	1.37
Stacked LSTM (Graves, 2013)	1.67
MRNN (Sutskever *et al.*, 2011)	1.60
GF-LSTM (Chung *et al.*, 2015a)	1.58
Grid-LSTM (Kalchbrenner *et al.*, 2015)	1.47
MI-LSTM (Wu *et al.*, 2016)	1.44
Recurrent Highway Networks (Zilly *et al.*, 2016)	1.42
Recurrent Memory Array Structures (Rocki, 2016a)	1.40
Layer-normalized LSTM (Ba *et al.*, 2016)†	1.46
HM-LSTM	**1.37**

Hierarchical Multiscale Recurrent Neural Networks: Results (Chung, Ahn & Bengio arXiv:1609.01704)

- Automatically segmenting so as to better predict the next character

- SF-LSTM is not comparable (changes parameters on the fly)

Text8	
Model	**BPC**
td-LSTM (Zhang *et al.*, 2016)	1.63
HF-MRNN (Mikolov *et al.*, 2012)	1.54
MI-RNN (Wu *et al.*, 2016)	1.52
Skipping-RNN (Pachitariu and Sahani, 2013)	1.48
MI-LSTM (Wu *et al.*, 2016)	1.44
Batch-normalized LSTM (Cooijmans *et al.*, 2016)	1.36
HM-LSTM	**1.30**

Hutter Prize Wikipedia	
Model	**BPC**
SF-LSTM (Rocki, 2016b)*	1.37
Stacked LSTM (Graves, 2013)	1.67
MRNN (Sutskever *et al.*, 2011)	1.60
GF-LSTM (Chung *et al.*, 2015a)	1.58
Grid-LSTM (Kalchbrenner *et al.*, 2015)	1.47
MI-LSTM (Wu *et al.*, 2016)	1.44
Recurrent Highway Networks (Zilly *et al.*, 2016)	1.42
Recurrent Memory Array Structures (Rocki, 2016a)	1.40
Layer-normalized LSTM (Ba *et al.*, 2016)†	1.46
HM-LSTM	**1.37**

Analysis

Penn Treebank Line 1

Penn Treebank Line 2

Penn Treebank Line 3

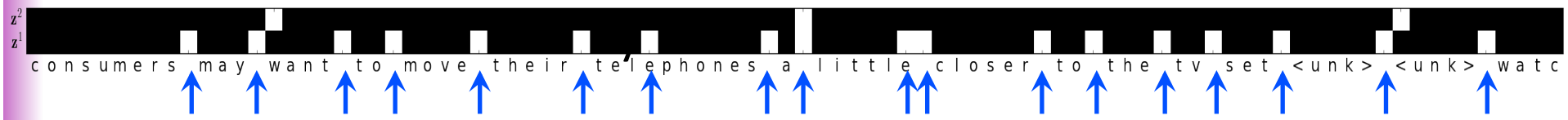
Wikipedia Line 1

Wikipedia Line 2

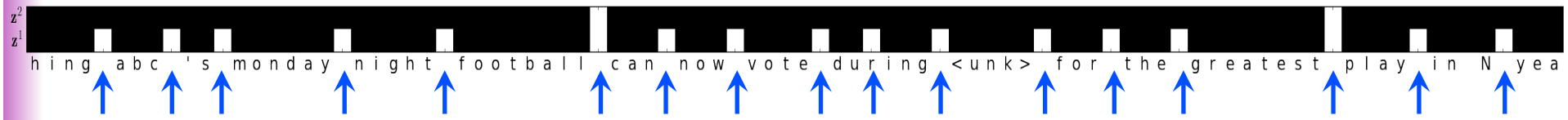
Wikipedia Line 3

Analysis

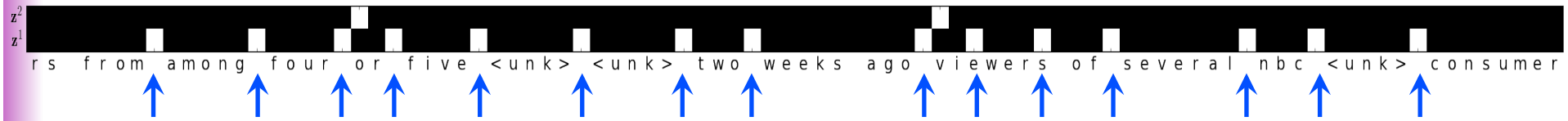
Penn Treebank Line 1



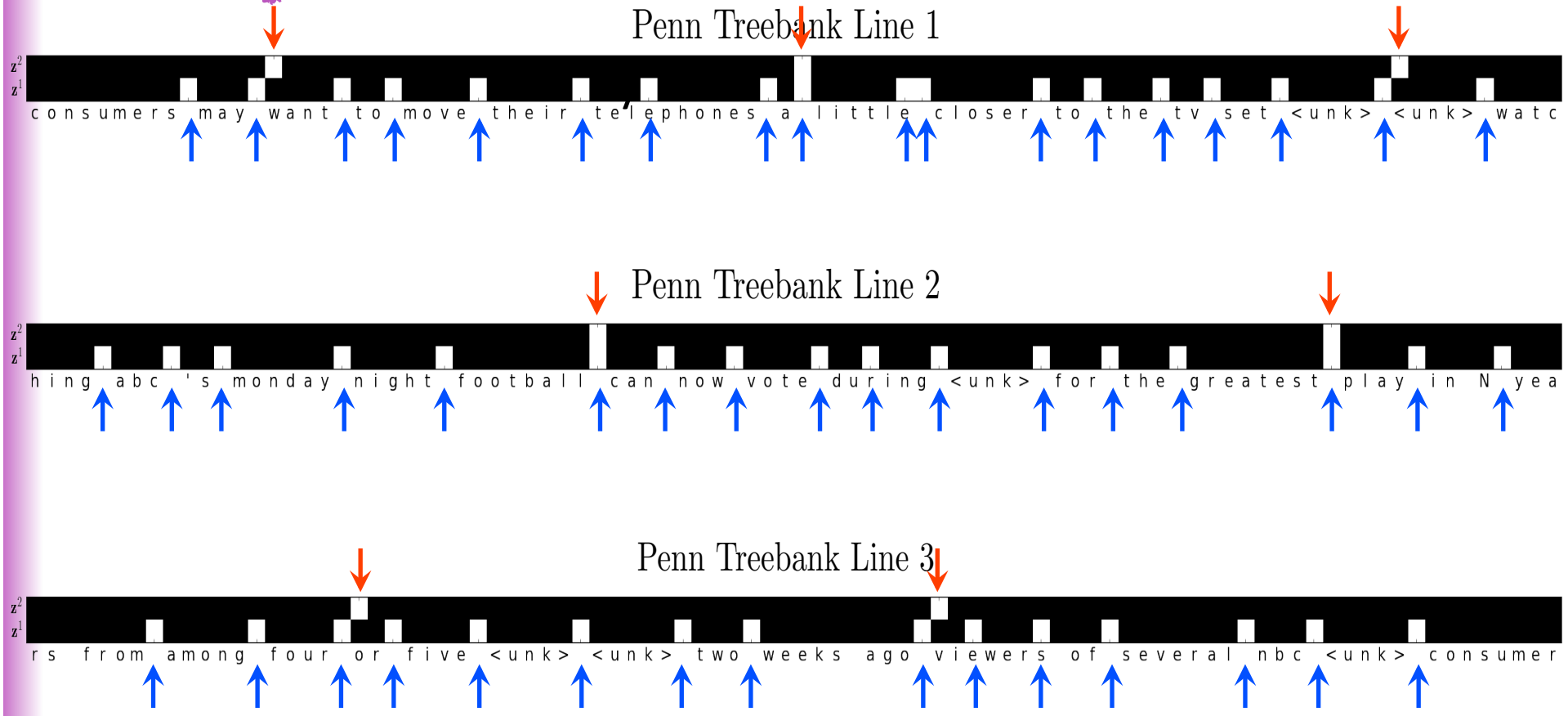
Penn Treebank Line 2



Penn Treebank Line 3

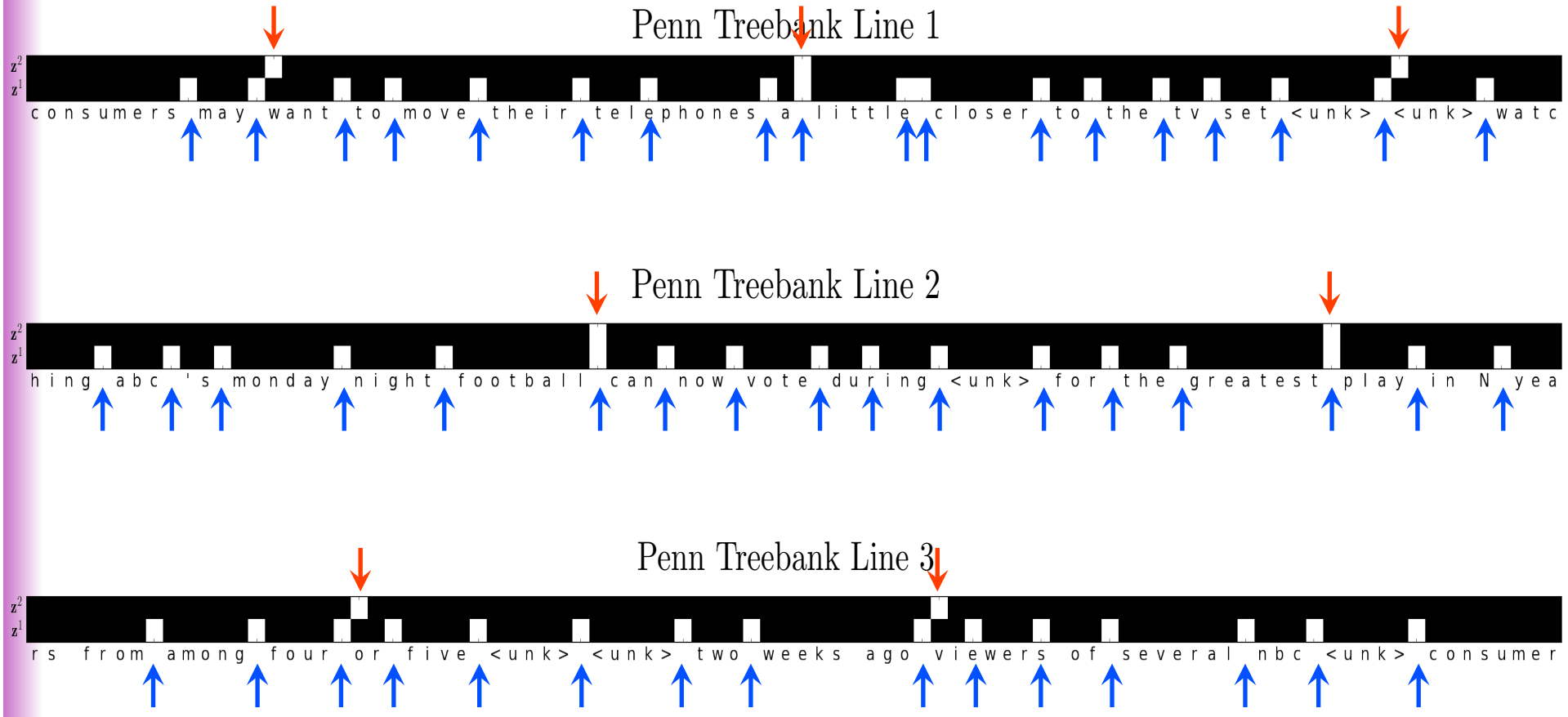


Analysis


Penn Treebank Line 1
consumers may want to move their telephones a little closer to the tv set <unk> <unk> watc
Penn Treebank Line 2
hing abc 's monday night football can now vote during <unk> for the greatest play in N yea
Penn Treebank Line 3
rs from among four or five <unk> <unk> two weeks ago viewers of several nbc <unk> consumer

Analysis

- Detected Boundaries by the HM-LSTM



$= 270 \times \frac{1}{5}$ $\approx 54 \times \frac{1}{8}$

1st layer: 270 updates, 2nd layer: 54 updates, 3rd layer: 7 updates

Montreal Institute for Learning Algorithms

MILA

Université de Montréal