Recurrent Neural Networks

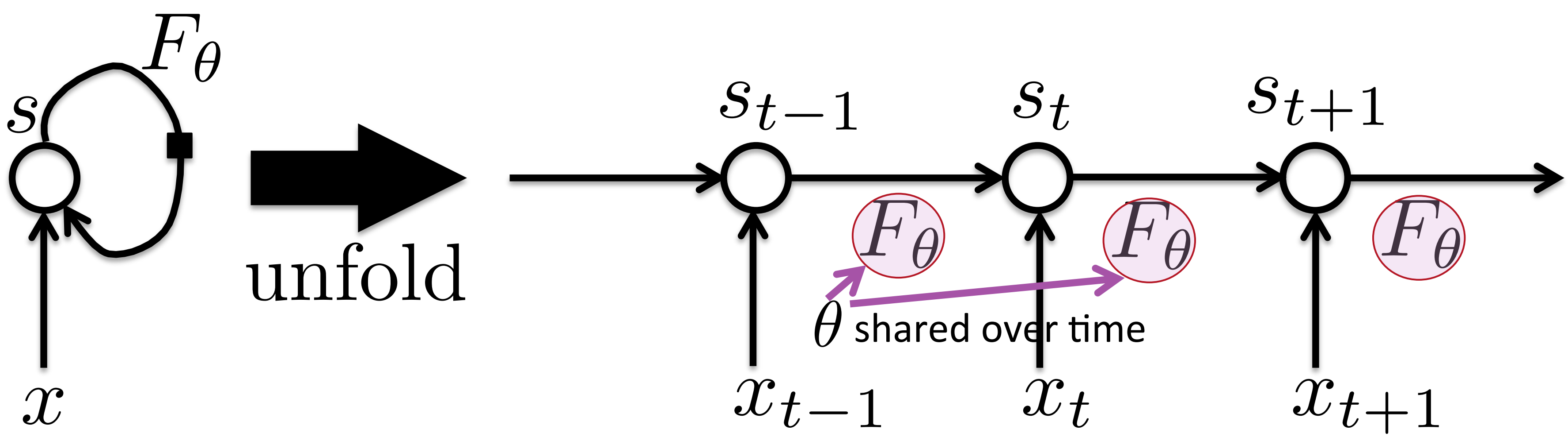
Deep Learning Summer School 2016
Yoshua Bengio
Montreal Institute for Learning Algorithms
Université de Montréal

PLUG: **Deep Learning**, MIT Press book in press,
Chapters will remain online

Recurrent Neural Networks

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

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

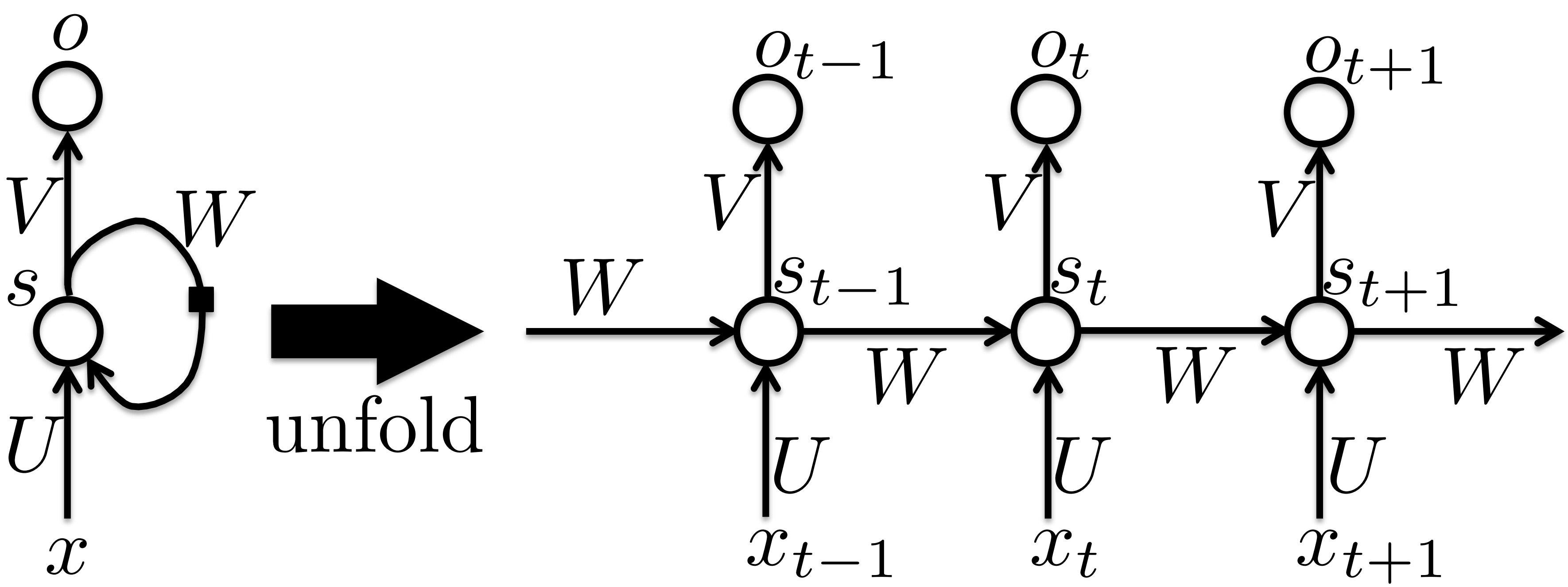


$$s_t = G_t(x_t, x_{t-1}, x_{t-2}, \ldots, x_2, x_1)$$

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

Recurrent Neural Networks

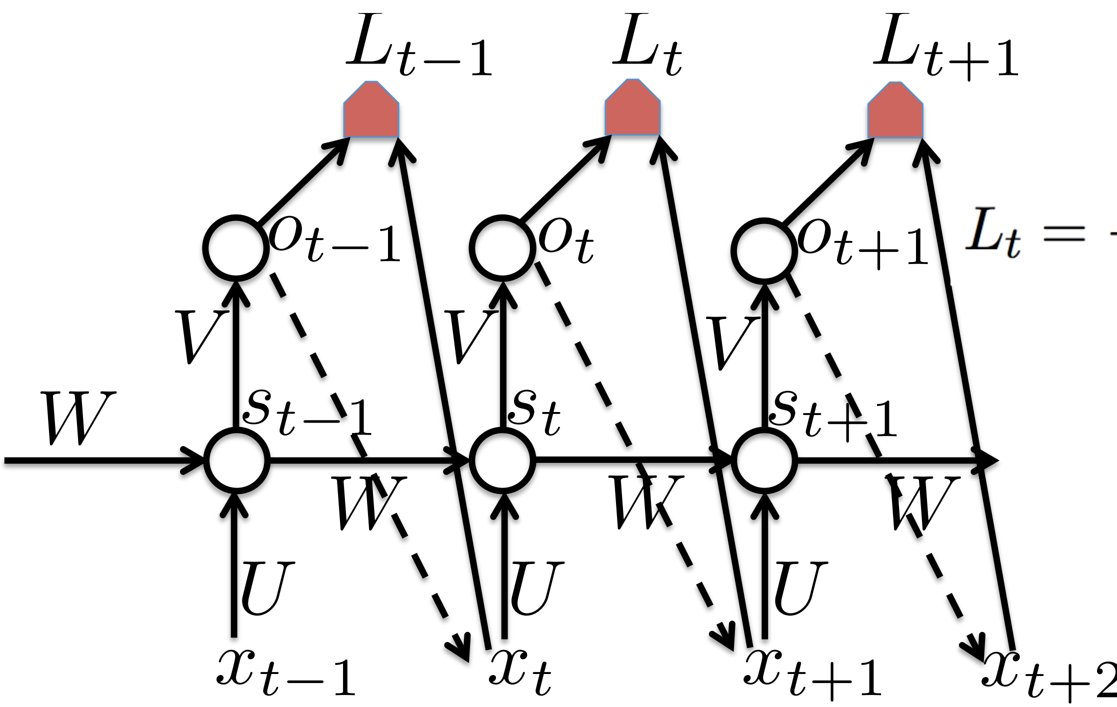
- Can produce an output at each time step: unfolding the graph tells us how to back-prop through time.



Generative RNNs

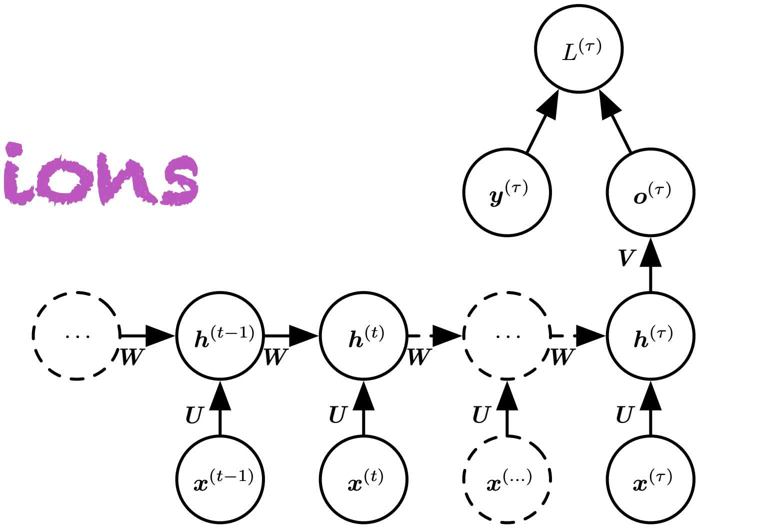
- An RNN can represent a fully-connected **directed generative model**: every variable predicted from all previous ones.

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



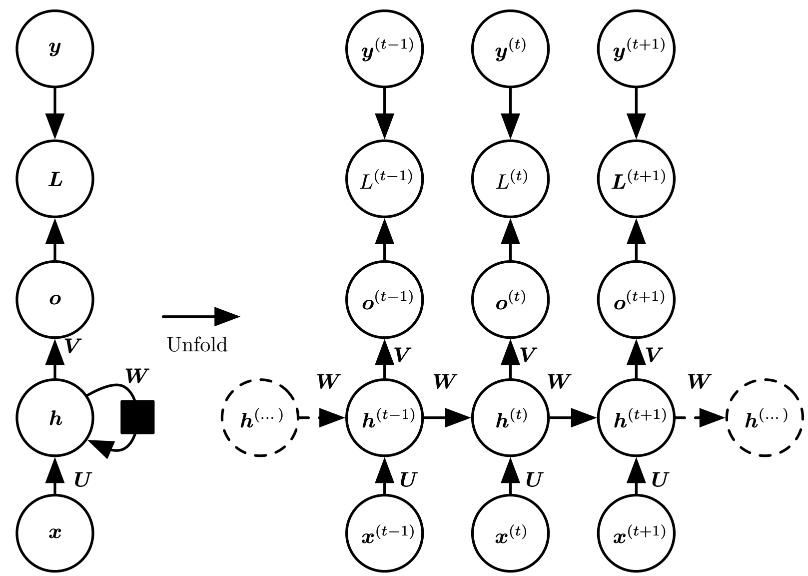
$$L_t = -\log P(x_t | x_{t-1}, x_{t-2}, \ldots x_1)$$

Conditional Distributions

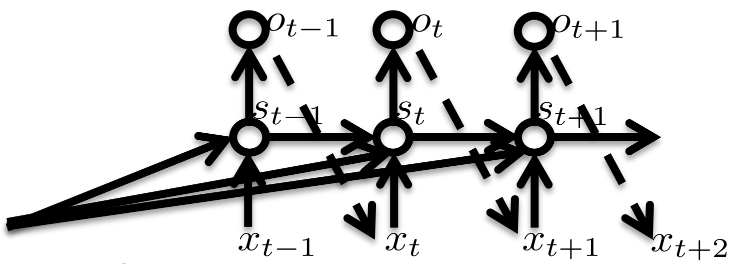


- Sequence to vector

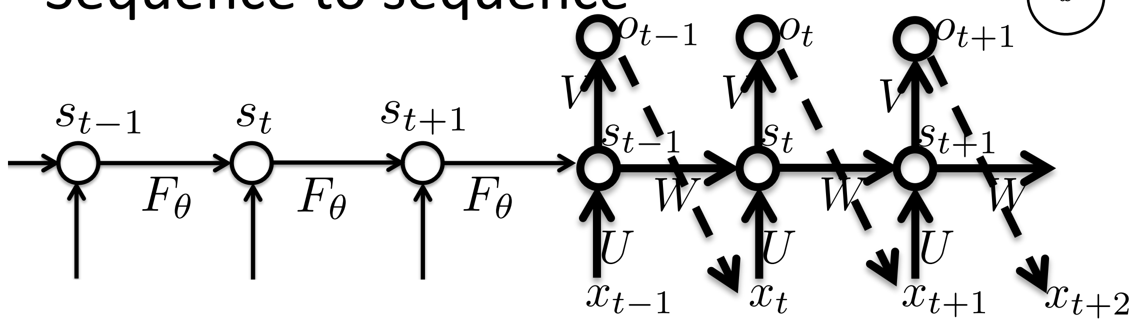
- Sequence to sequence of the same length, aligned



- Vector to sequence

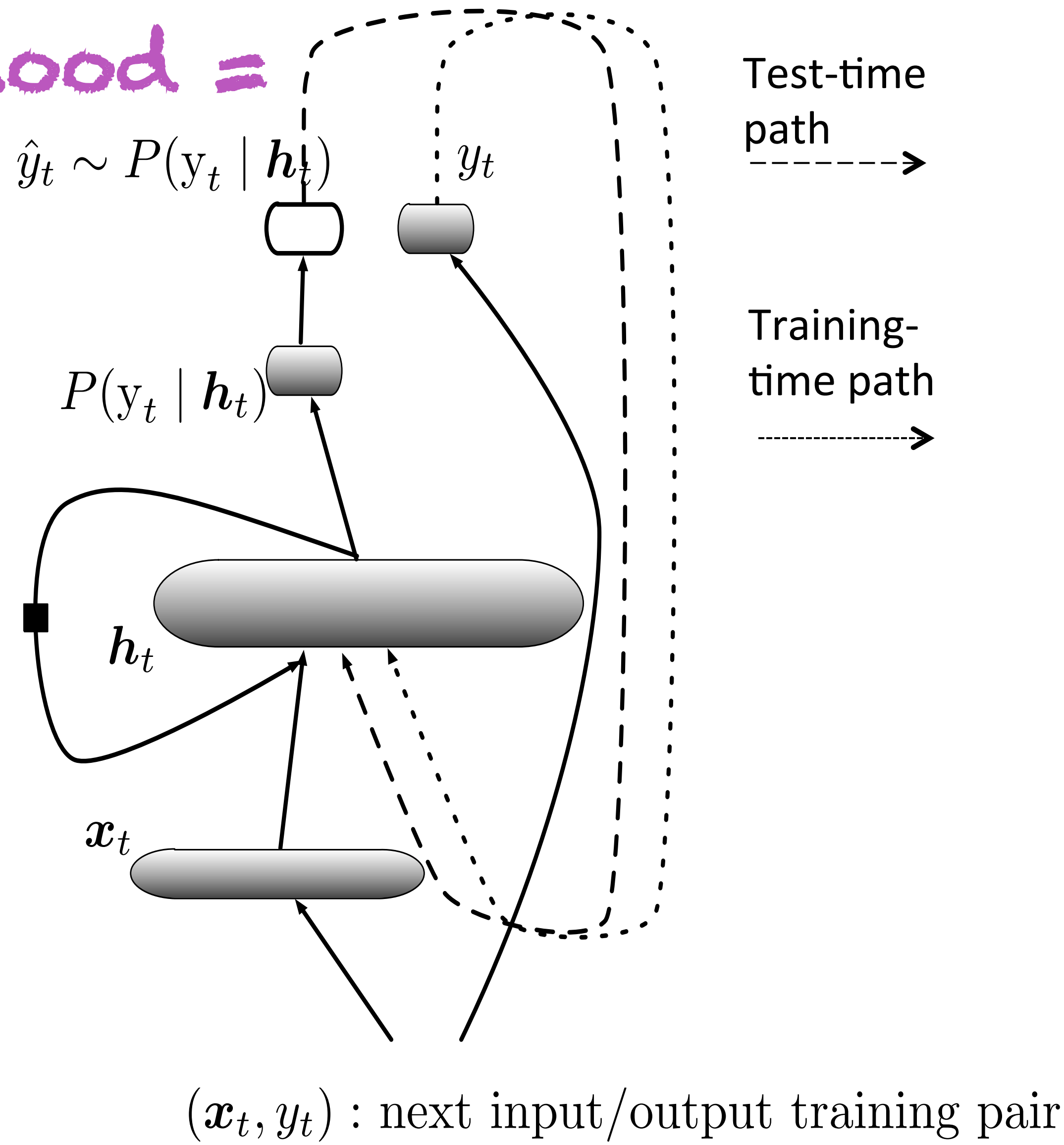


- Sequence to sequence



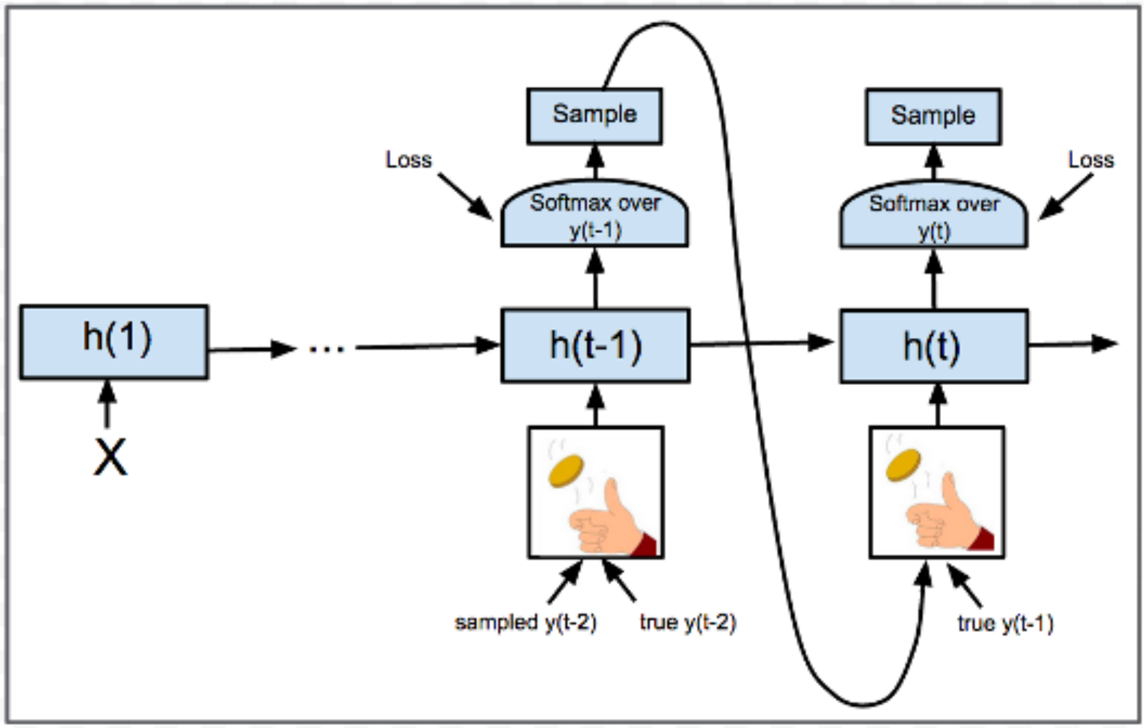
Maximum Likelihood = Teacher Forcing

- During training, past *y* in input is from training data
- At generation time, past *y* in input is generated
- Mismatch can cause ”compounding error”



Ideas to reduce the train/generate mismatch in teacher forcing

- Scheduled sampling *(S. Bengio et al, NIPS 2015)*



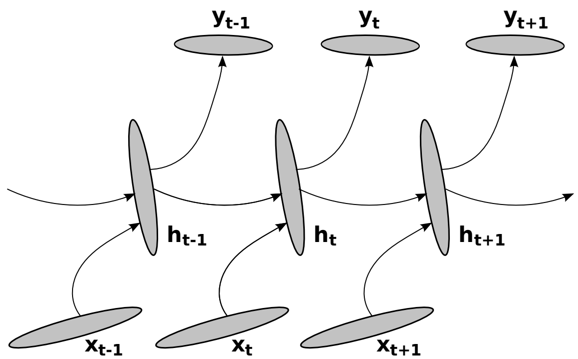
Related to
SEARN (Daumé et al 2009)
DAGGER (Ross et al 2010)

Gradually increase the probability of using the model's samples vs the ground truth as input.

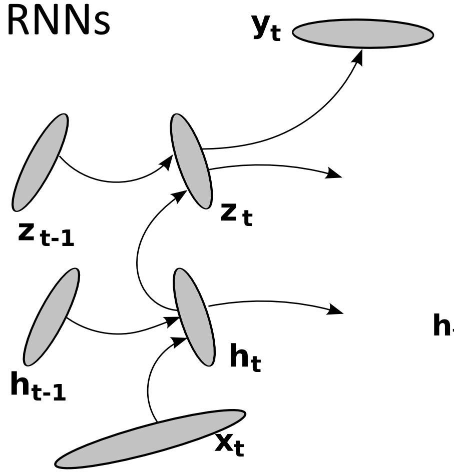
- Backprop through open-loop sampling recurrence & minimize long-term cost (but which one? GAN would be most natural → Professor Forcing)

Increasing the Expressive Power of RNNs with more Depth

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

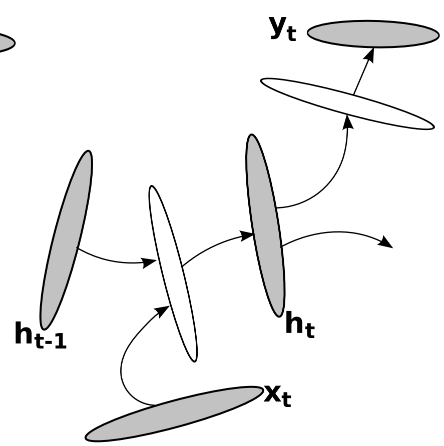


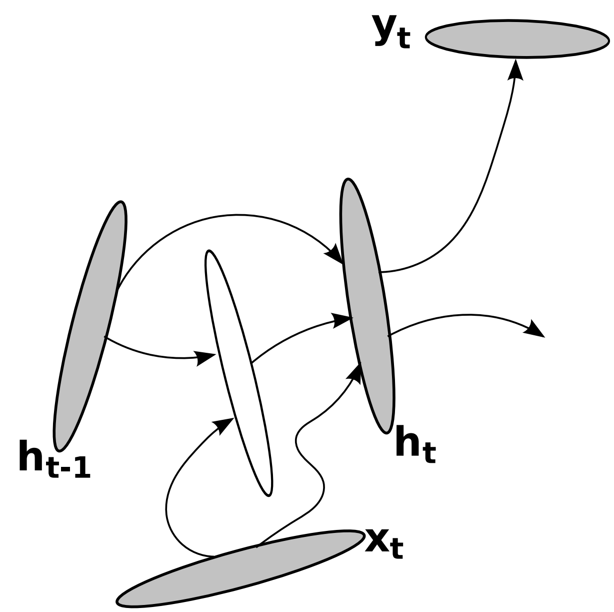
Ordinary RNNs



+ stacking

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

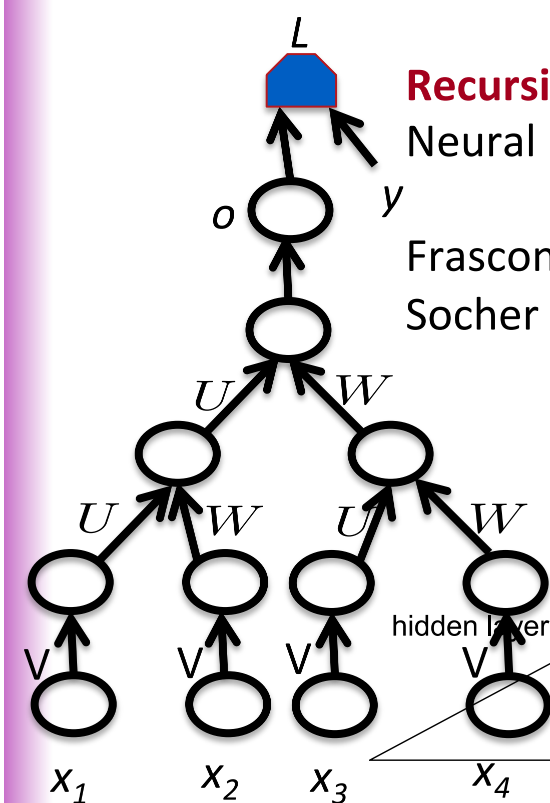




+ skip connections for creating shorter paths

Bidirectional RNNs, Recursive Nets, Multidimensional RNNs, etc.

- The unfolded architecture needs not be a straight chain

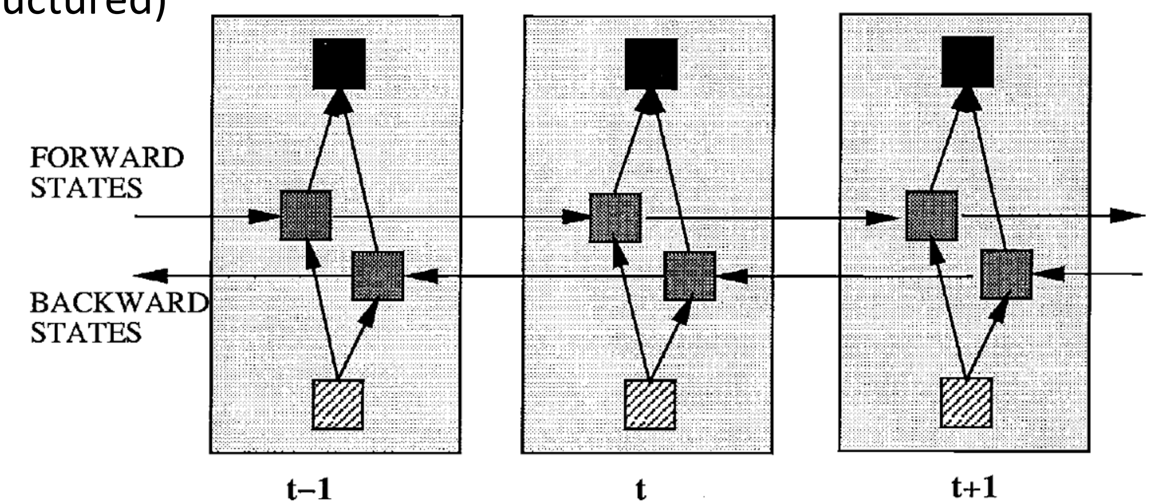


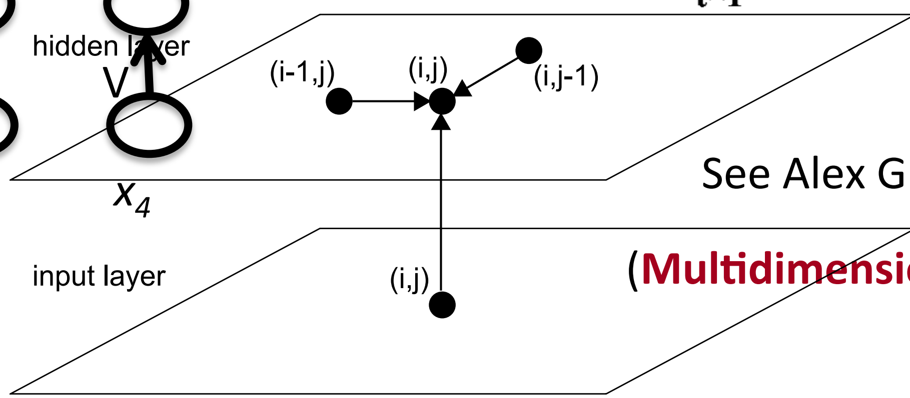
Recursive (tree-structured) Neural Nets:

Frasconi et al 97

Socher et al 2011

Bidirectional RNNs (Schuster and Paliwal, 1997)





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

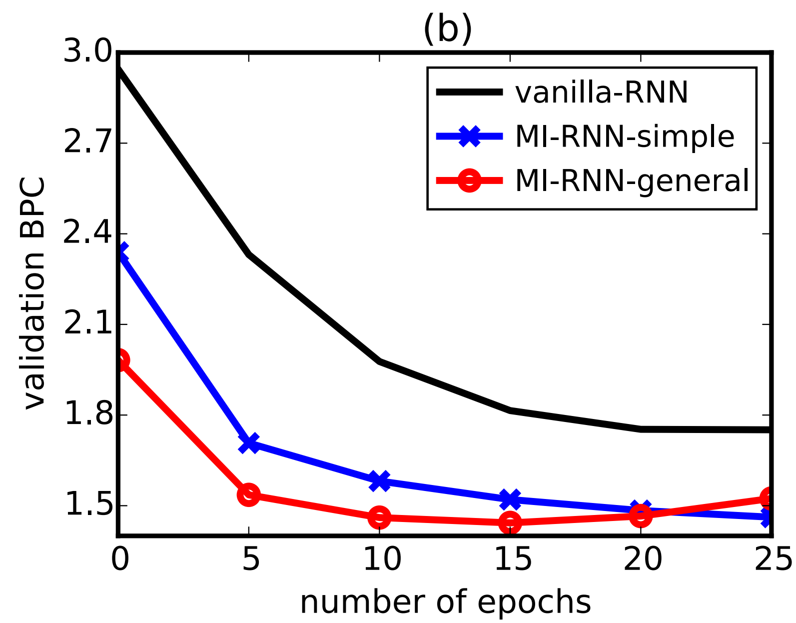
(**Multidimensional** RNNs, Graves et al 2007)

Multiplicative Interactions

(Wu et al, 2016, arXiv:1606.06630)

- 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:

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



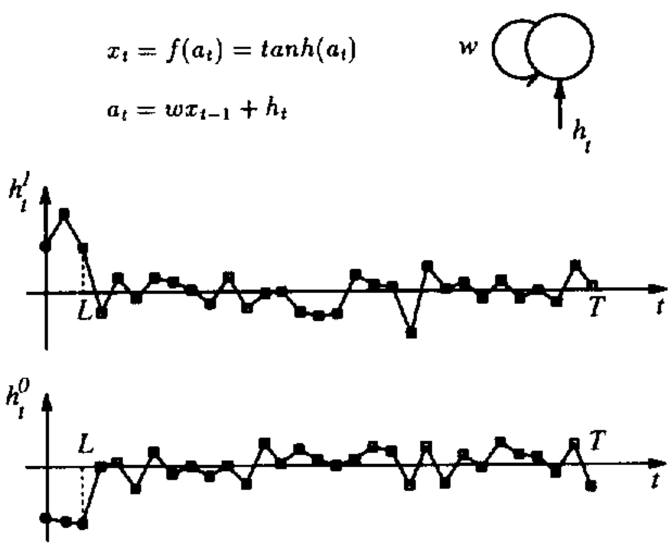
Learning Long-Term Dependencies with Gradient Descent is Difficult

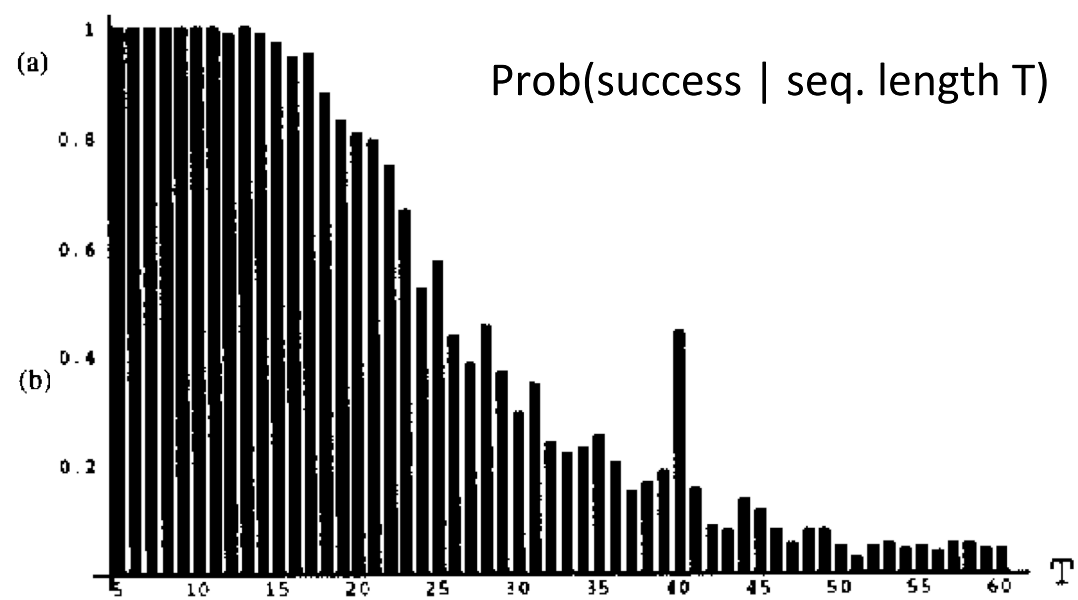
  

Y. Bengio, P. Simard & P. Frasconi, IEEE Trans. Neural Nets, **1994**

Simple Experiments from 1991 while I was at MIT

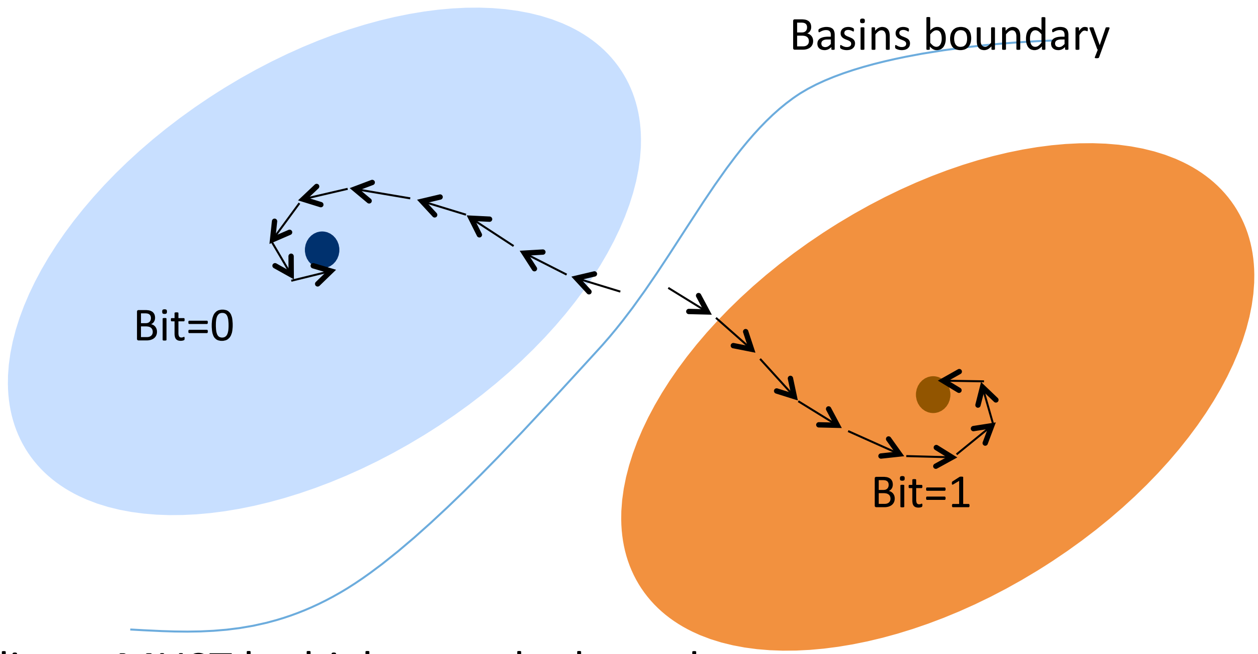
- 2 categories of sequences
- Can the single tanh unit learn to store for T time steps 1 bit of information given by the sign of initial input?





How to store 1 bit? Dynamics with multiple basins of attraction in some dimensions

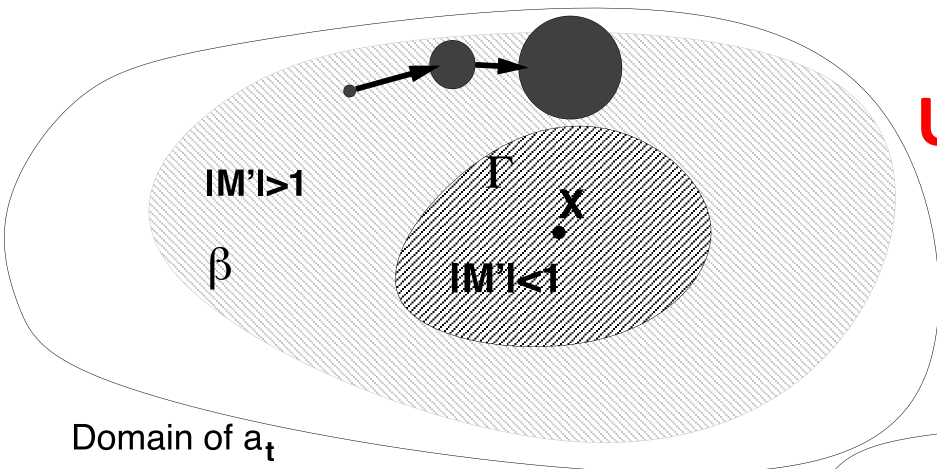
- Some subspace of the state can store 1 or more bits of information if the dynamical system has multiple basins of attraction in some dimensions



Note: gradients MUST be high near the boundary

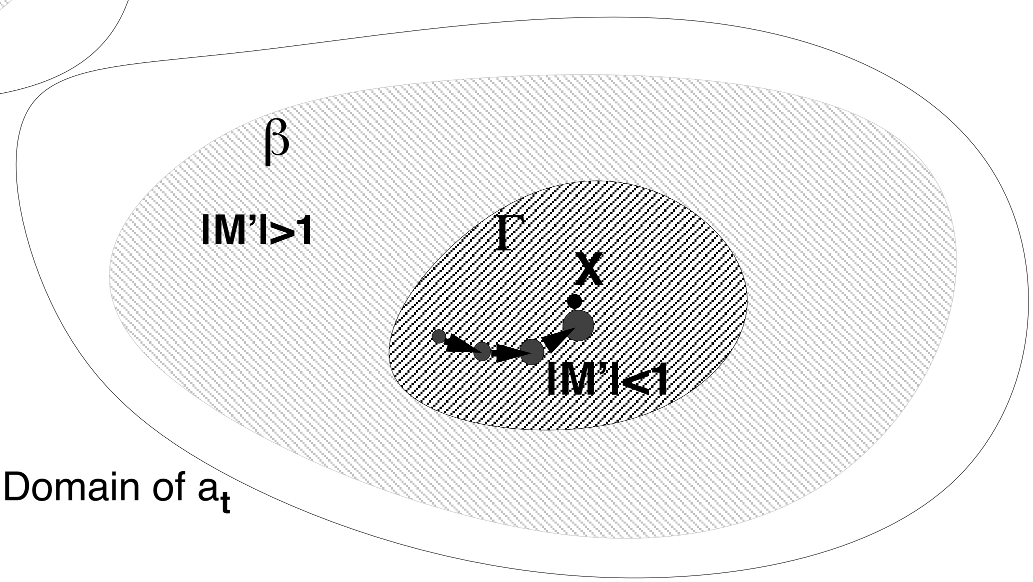
Robustly storing 1 bit in the presence of bounded noise

- 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

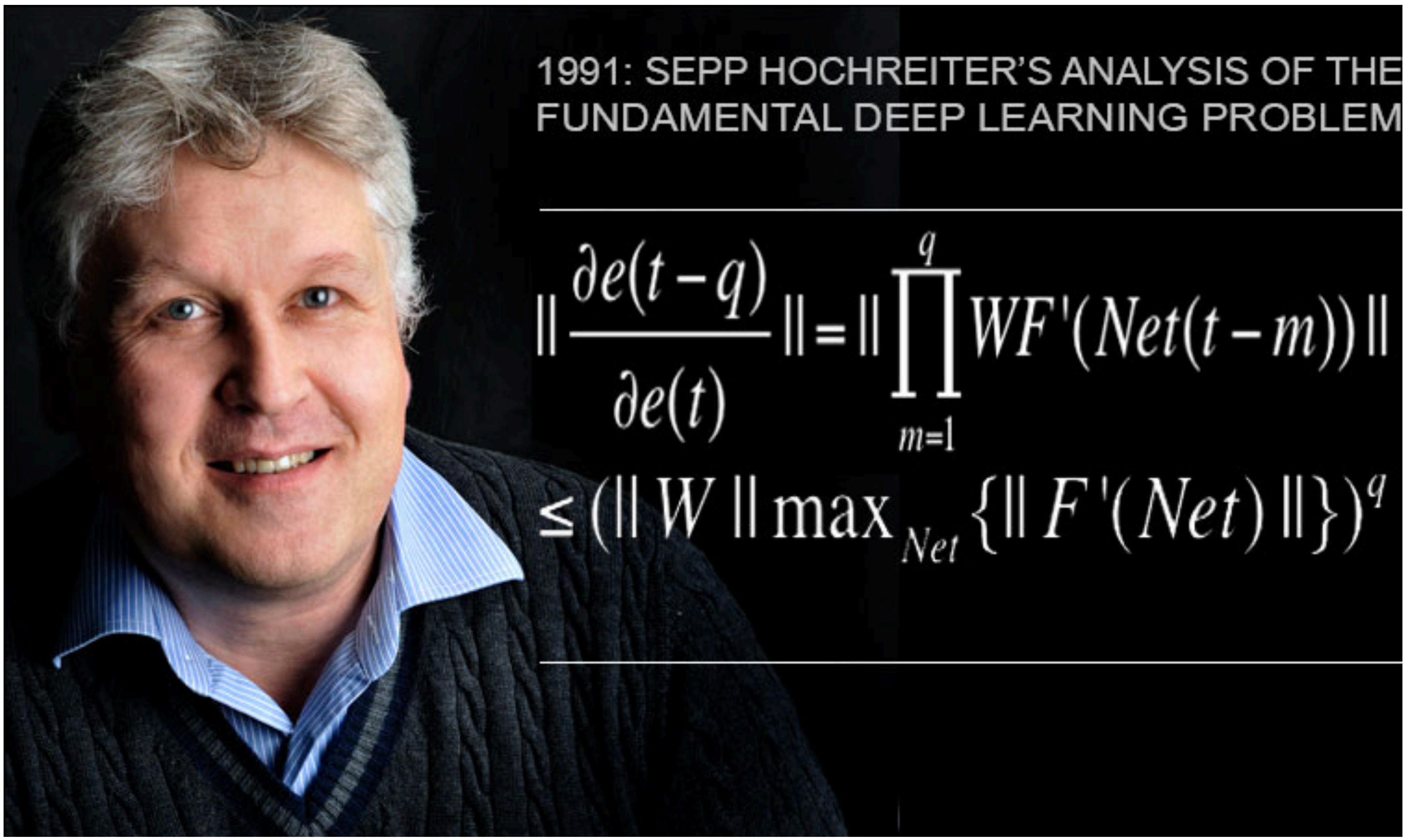
$$L = L(s_T(s_{T-1}(\ldots s_{t+1}(s_t, \ldots))))$$

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

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

Vanishing or Exploding Gradients

- Hochreiter's 1991 MSc thesis (in German) had independently discovered that backpropagated gradients in RNNs tend to either vanish or explode as sequence length increases



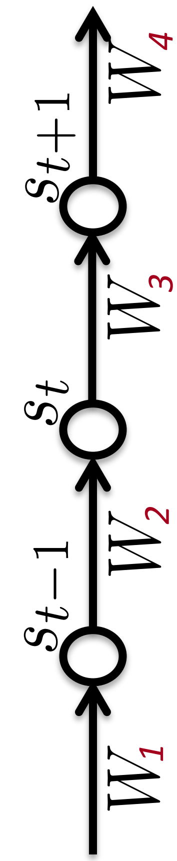
Why it hurts gradient-based learning

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

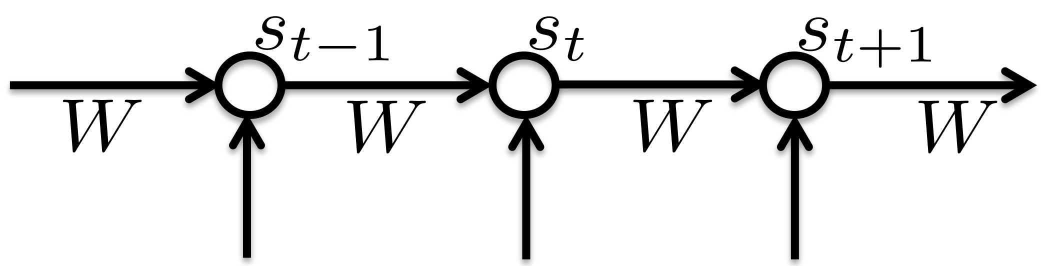
$$\frac{\partial C_t}{\partial W} = \sum_{\tau \leq t} \frac{\partial C_t}{\partial a_\tau} \frac{\partial a_\tau}{\partial W} = \sum_{\tau \leq t} \frac{\partial C_t}{\partial a_t} \frac{\partial a_t}{\partial a_\tau} \frac{\partial a_\tau}{\partial W}$$

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

Vanishing Gradients in Deep Nets are Different from the Case in RNNs



- If it was just a case of vanishing gradients in deep nets, we could just rescale the per-layer learning rate, but that does not really fix the training difficulties.



- Can't do that with RNNs because the weights are shared, & total true gradient = sum over different "depths"

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

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

RNN Tricks

(Pascanu, Mikolov, Bengio, ICML 2013; Bengio, Boulanger & Pascanu, ICASSP 2013)

- Clipping gradients (avoid exploding gradients)
- Leaky integration (propagate long-term dependencies)
- Momentum (cheap 2nd order)
- Initialization (start in right ballpark avoids exploding/vanishing)
- Sparse Gradients (symmetry breaking)
- Gradient propagation regularizer (avoid vanishing gradient)
- Gated self-loops (LSTM & GRU, reduces vanishing gradient)

Dealing with Gradient Explosion by Gradient Norm Clipping

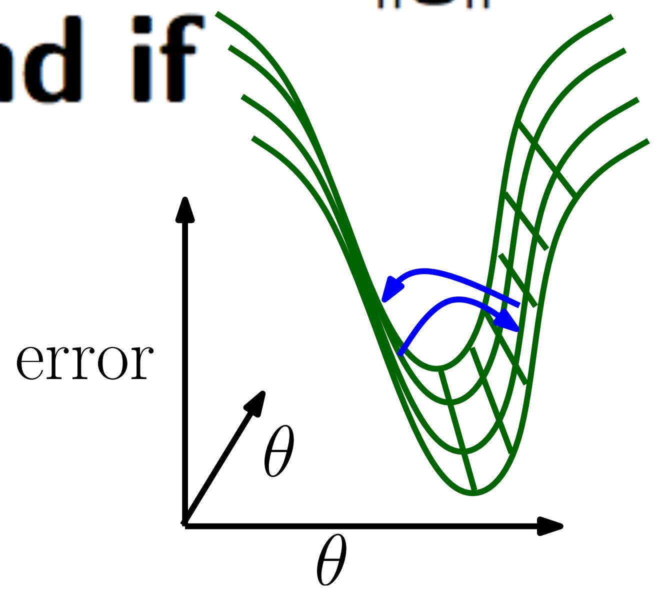
(Mikolov thesis 2012;
Pascanu, Mikolov, Bengio, ICML 2013)

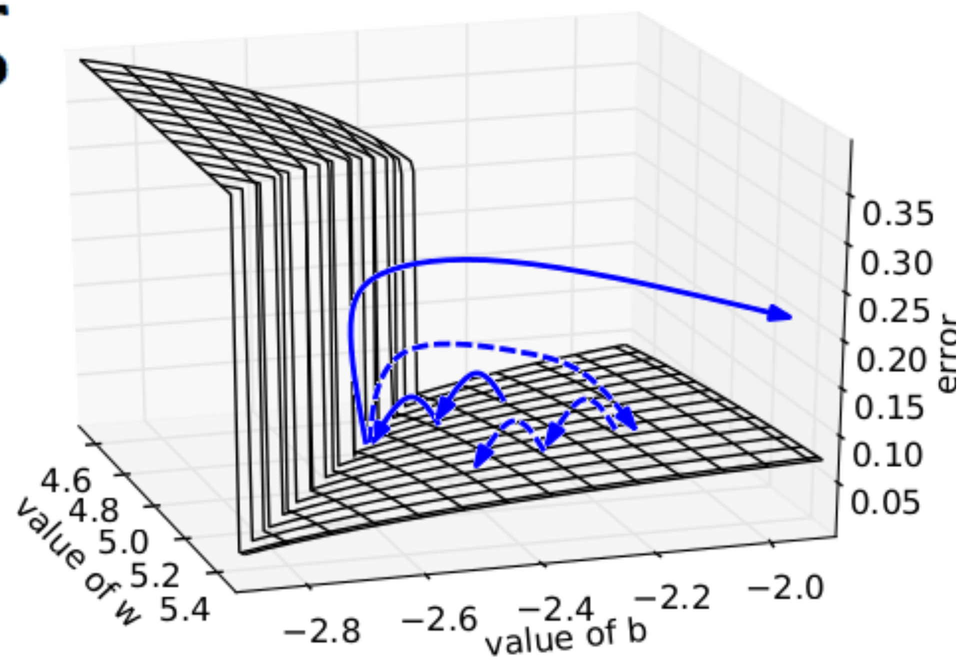
$$\hat{\mathbf{g}} \leftarrow \frac{\partial error}{\partial \theta}$$

$$\textbf{if} \quad \|\hat{\mathbf{g}}\| \geq threshold \ \textbf{then}$$

$$\quad \hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|}\hat{\mathbf{g}}$$

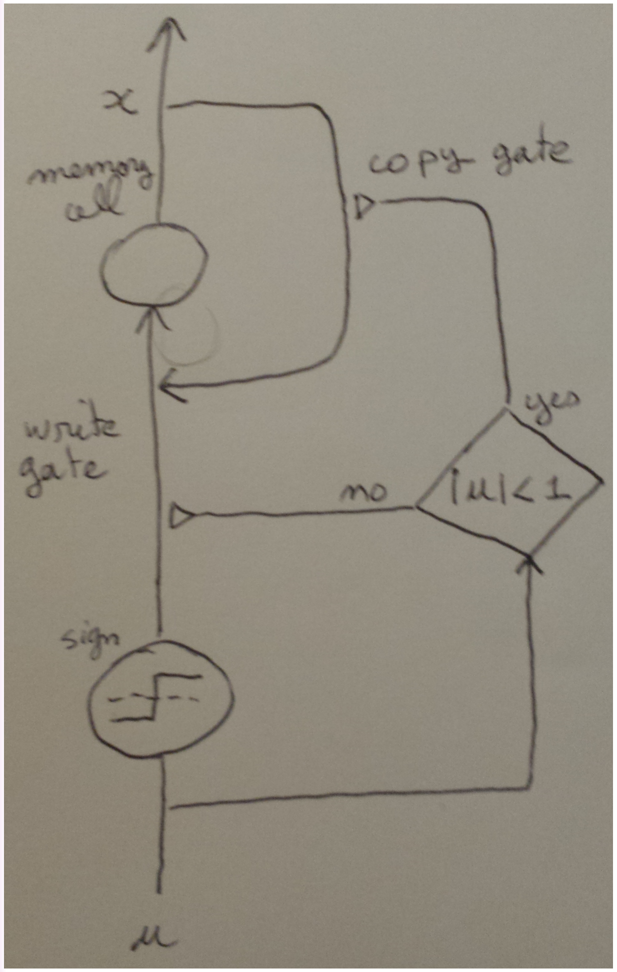
$$\textbf{end if}$$





Conference version (1993) of the 1994 paper by the same authors had a predecessor of GRU and targetprop

(The problem of learning long-term dependencies in recurrent networks, Bengio, Frasconi & Simard ICNN'1993)



IV. A TRAINABLE FLIP-FLOP

- Flip-flop unit to store 1 bit, with gating signal to control when to write

$$x_{t+1} = f(x_t, u_t)$$

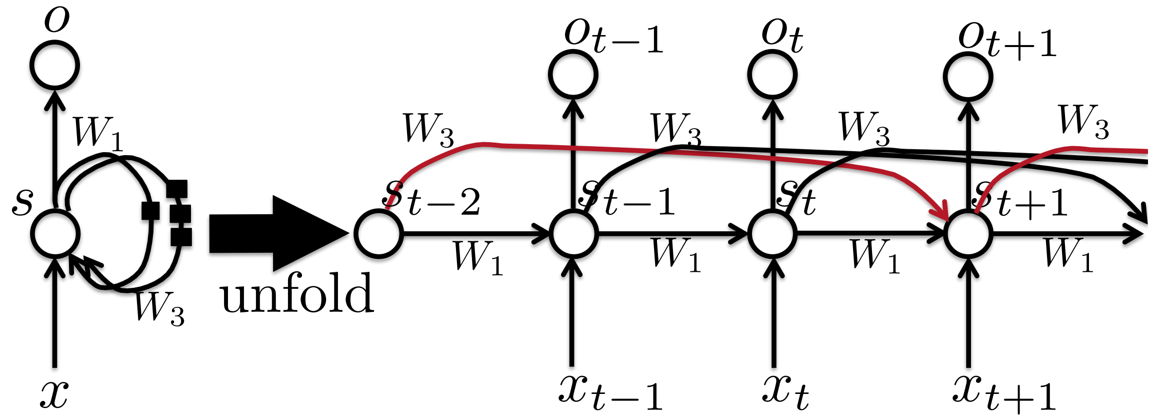
$$f(x,u) = \begin{vmatrix} 1 & \text{if } |u|<1 \text{ and } x \geq 0 \\ & \text{or if } u \geq 1 \\ -1 & \text{otherwise} \end{vmatrix} \tag{8}$$

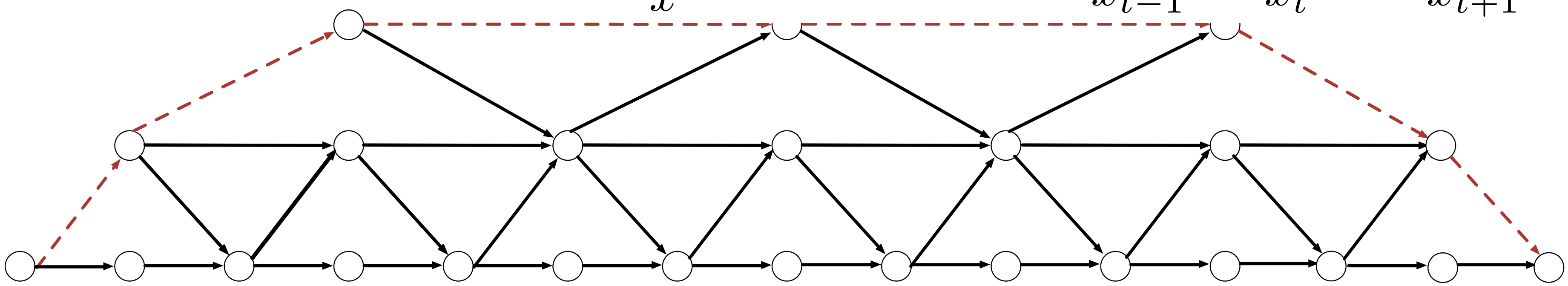
- Pseudo-backprop through it by a form of targetprop

$$\Delta x(\Delta f, u) = \begin{vmatrix} \Delta f & \text{if } |u| < 1 \\ 0 & \text{otherwise} \end{vmatrix} \tag{11}$$

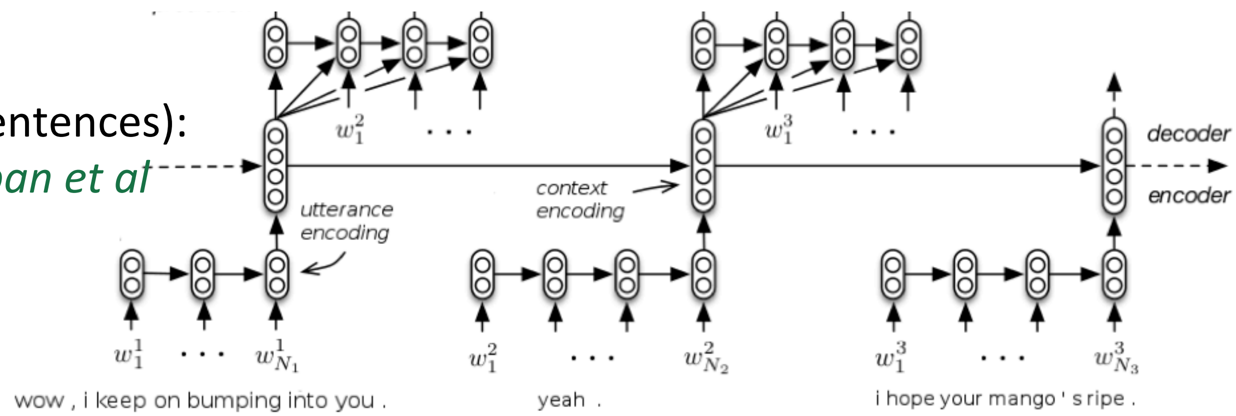
Delays & Hierarchies to Reach Farther

- 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

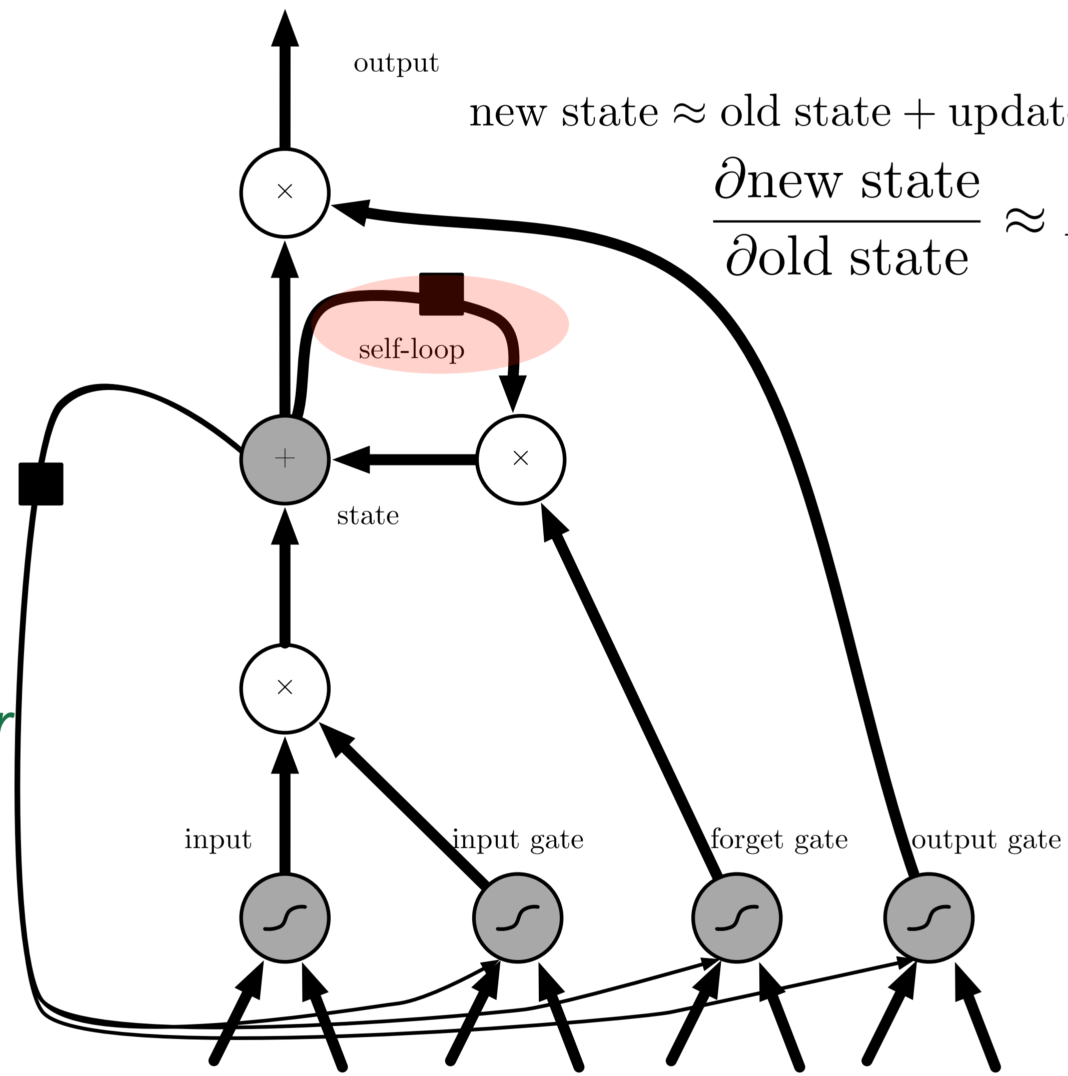
(Hochreiter 1991); first version of the LSTM, called Neural Long-Term Storage with self-loop

- 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)

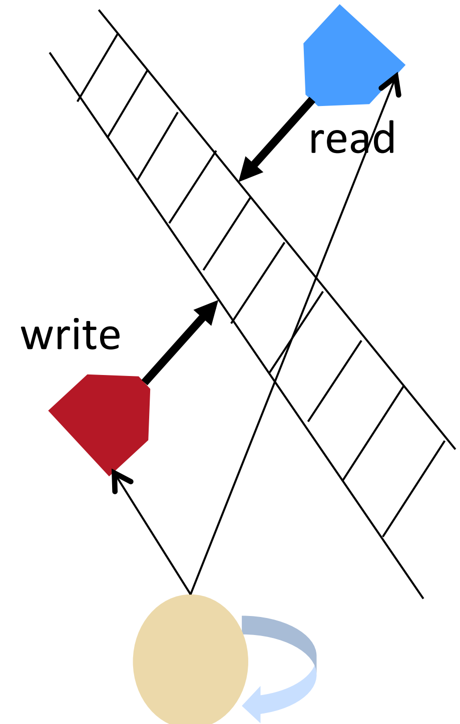
$$\text{new state} \approx \text{old state} + \text{update}$$

$$\frac{\partial \text{new state}}{\partial \text{old state}} \approx I$$



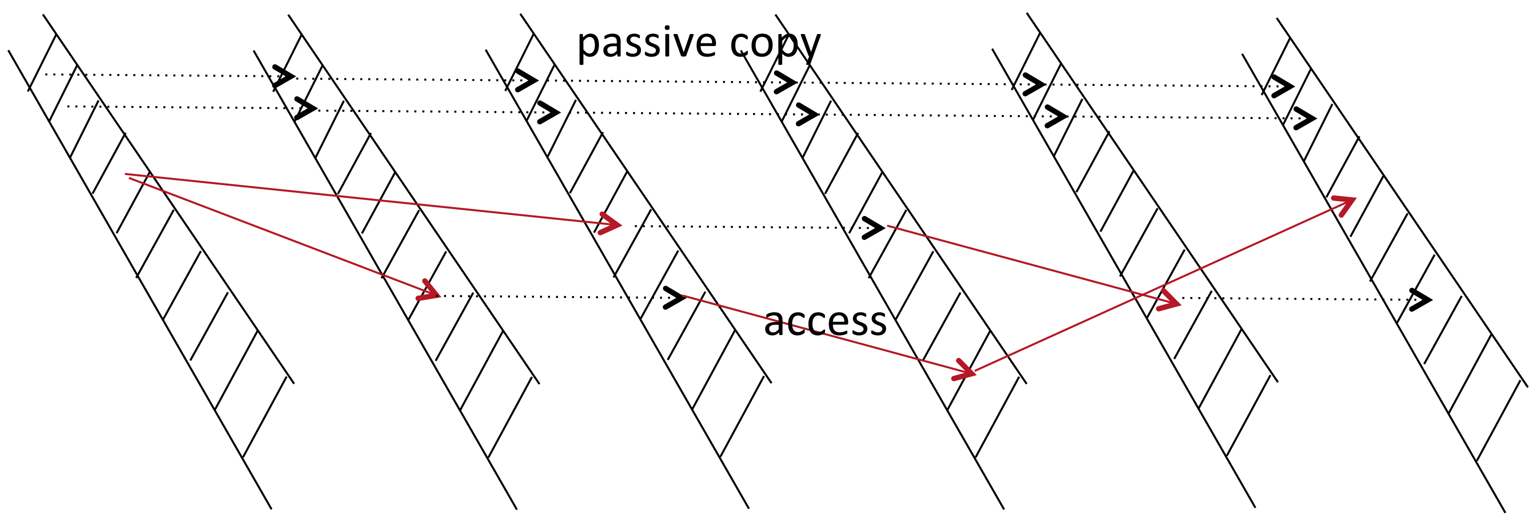
Fast Forward 20 years: Attention Mechanisms for Memory Access

- 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

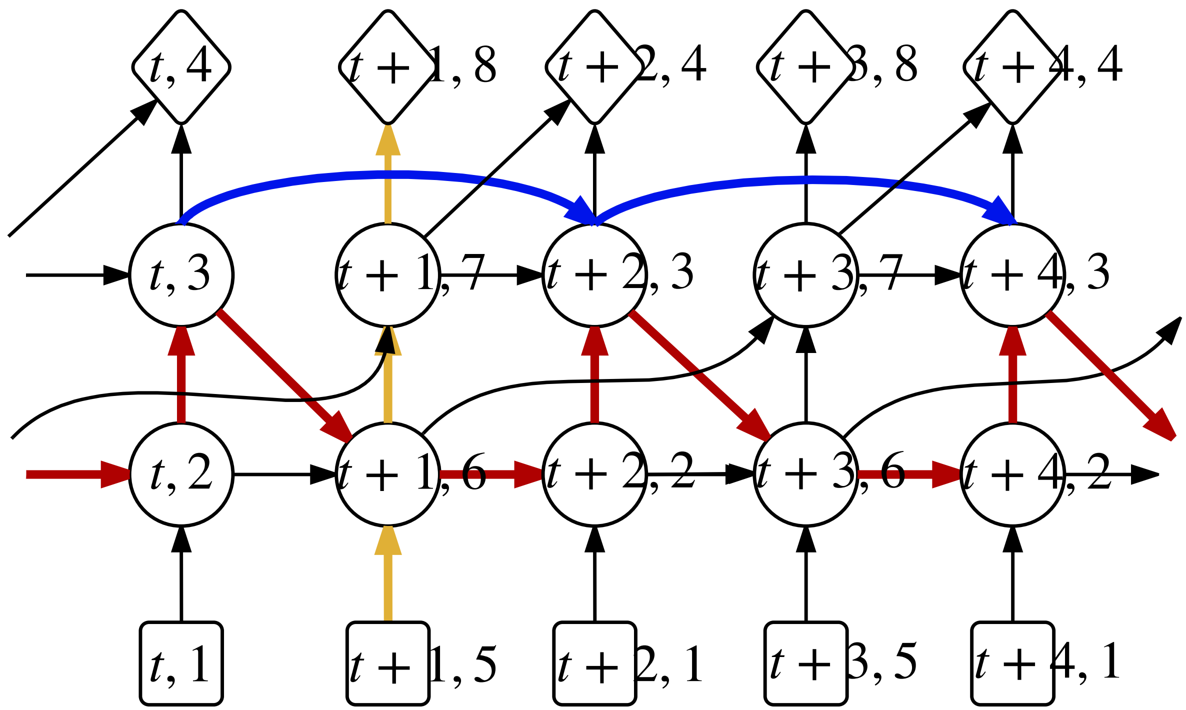
- Memory = part of the state
- Memory-based networks are special RNNs
- A mental state stored in an external memory can stay for arbitrarily long durations, until it is overwritten (partially or not)
- Forgetting = vanishing gradient.
- Memory = **higher-dimensional state**, avoiding or reducing the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



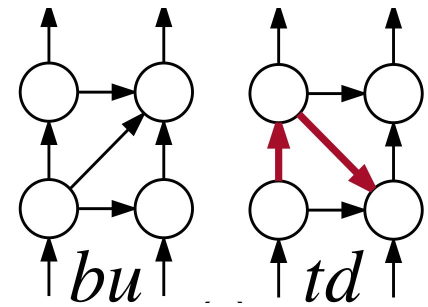
Designing the RNN Architecture

*(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

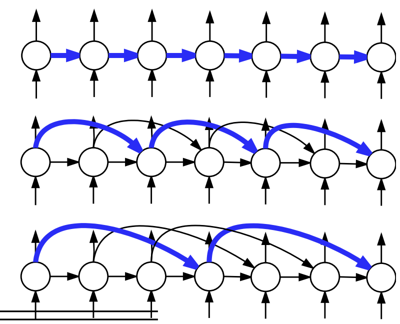


It makes a difference



- 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**

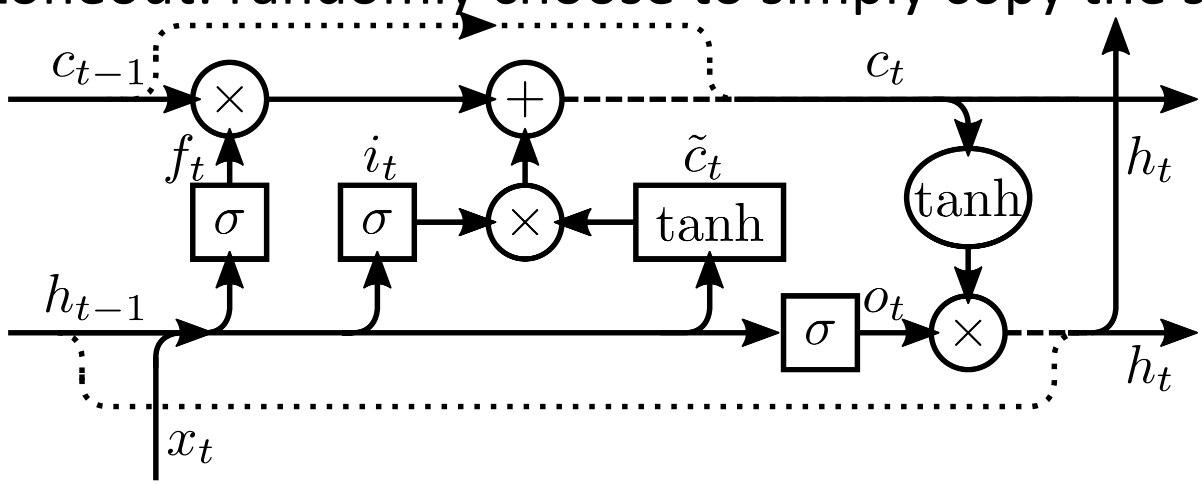
Table 2: Results for MNIST/*p*MNIST. **Top-left**: test accuracies with different s for $tanh$ RNN. **Top-right**: test accuracies with different s for LSTM. **Bottom**: compared to previous results. **Bottom-right**: test accuracies for architectures (1), (2), (3) and (4) for $tanh$ RNN.

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)*

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

- Zoneout: randomly choose to simply copy the state unchanged

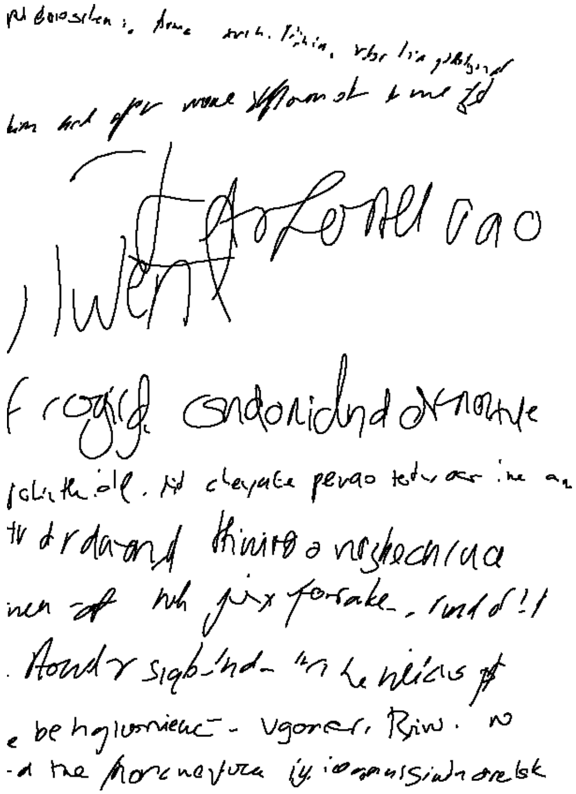


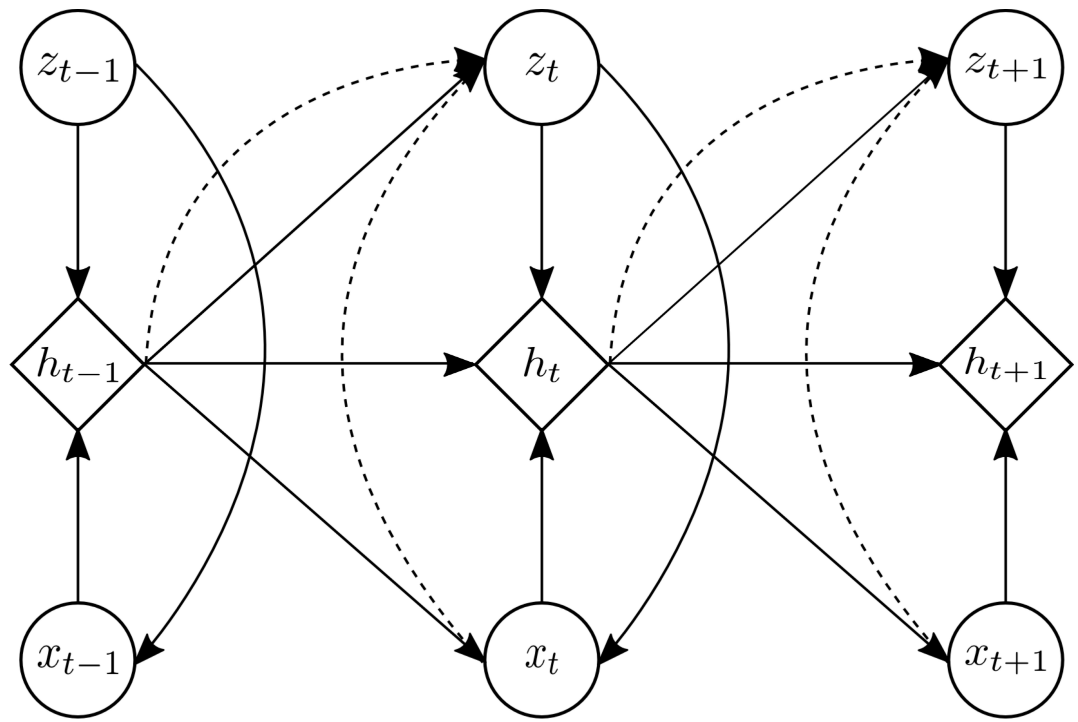
(Krueger et al 2016, submitted)

Variational Generative RNNs

Injecting higher-level variations / latent variables in RNNs

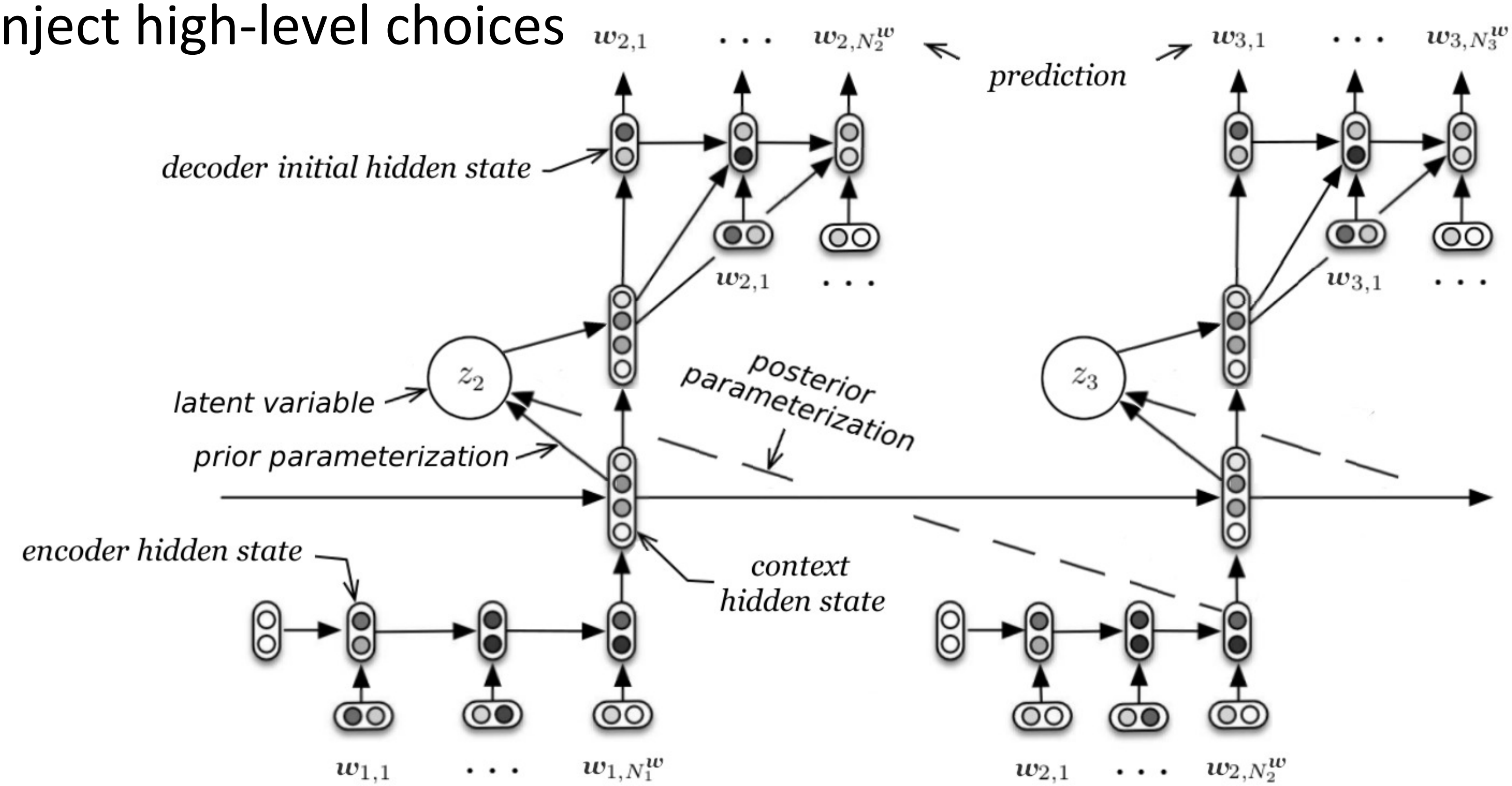
- *(Chung et al, NIPS'2015)*
- Regular RNNs have noise injected only in input space
- VRNNs also allow noise (latent variable) injected in top hidden layer; more « high-level » variability





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



VHRNN Results – Twitter Dialogues

Table 1: Wins, losses and ties (in %) of the VHRED model against the baselines based on the human study on Twitter (mean preferences $\pm$ 90% confidence intervals)

	Short Contexts			Long Contexts		
Opponent	**Wins**	**Losses**	**Ties**	**Wins**	**Losses**	**Ties**
VHRED vs LSTM	32.3 ± 2.4	$\mathbf{42.5 \pm 2.6}$	25.2 ± 2.3	$\mathbf{41.9 \pm 2.2}$	36.8 ± 2.2	21.3 ± 1.9
VHRED vs HRED	$\mathbf{42.0 \pm 2.8}$	31.9 ± 2.6	26.2 ± 2.5	$\mathbf{41.5 \pm 2.8}$	29.4 ± 2.6	29.1 ± 2.6
VHRED vs TF-IDF	$\mathbf{51.6 \pm 3.3}$	17.9 ± 2.5	30.4 ± 3.0	$\mathbf{47.9 \pm 3.4}$	11.7 ± 2.2	40.3 ± 3.4

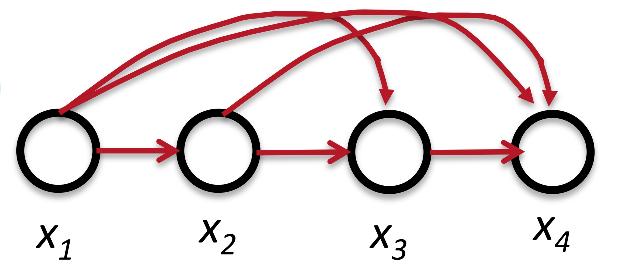
Table 2: Twitter examples for the neural network models. The $\rightarrow$ token indicates a change of turn.

Context	**Response**
jamaica blood!! <url> → lol. yes [name]!. [name]! → what's poppin in toronto homie	**VHRED:** nothing much. just chillin. what's good with you? **LSTM:** something!!!!!!! **HRED:** something!!!!!!
whhhhhhyyyy is it suddenly cold in tallahassee today?? this is florida! how can i dream of the beach when my fingers are icey cold?? → sittin on deck in sun looking at lake travis in austin! enjoying day b4 birthday	**VHRED:** the sun is shining!!! **LSTM:** i know!!!!!!!!! **HRED:** something!!!!!!!
reaches for [name]! → - approaches him warily - hey there buddy. - swings him into the air - how you doin? → tilts his head and your wariness and reaches for your face [name].	**VHRED:** - looks up at you and smiles - you're so cute. - kisses your cheek - **LSTM:** - smiles - i'm fine thank you. how are you? **HRED:** - grins, looking at you as i look at you -
i think i want an ipad2. but i already have an ibook. want vs. need. → if you have an ibook. then you deserve an ipad!	**VHRED:** that's what i'm thinking. i'm going to buy a new one. **LSTM:** i want one!!!! **HRED:** i don't know what to do with it. i don't know what to do.
ah echt? dan anrtwoord ik volgende keer via twitter als je me smst. → haha doe maar p. goed bezig hoor myrthe d → goed bezig? die snap ik even niet p	**VHRED:** haha, ik heb er geen zin in de something, maar ik heb er geen something meer something **LSTM:** haha something something something **HRED:** something something something something something something something something
i love katie waissel more than like, anything in life. → i wanna c the show again! i need <number> meet her again! n i want a b-day tweet nxt week doubt that will happen x	**VHRED:** me too. i'm going to have to wait for her to come back. i'm so excited. **LSTM:** i'm sure she will. i'm sure she'll be fine. **HRED:** i'm sure she'll be fine, i'm sure she'll be fine.
my math teacher over here talkn bout nickels nd i'm tryna make big money i need to be in calculus → lol	**VHRED:** i need to go to the gym and get my hair done lol **LSTM:** lol **HRED:** how u been

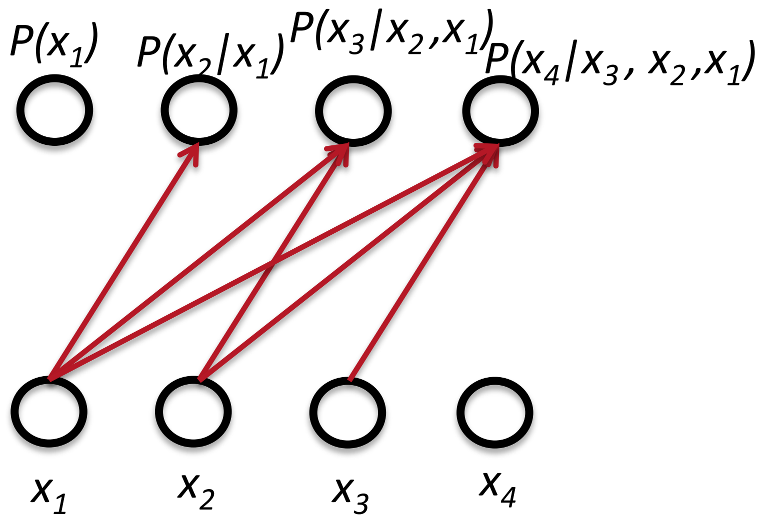
Other Fully-Observed Neural Directed Graphical Models

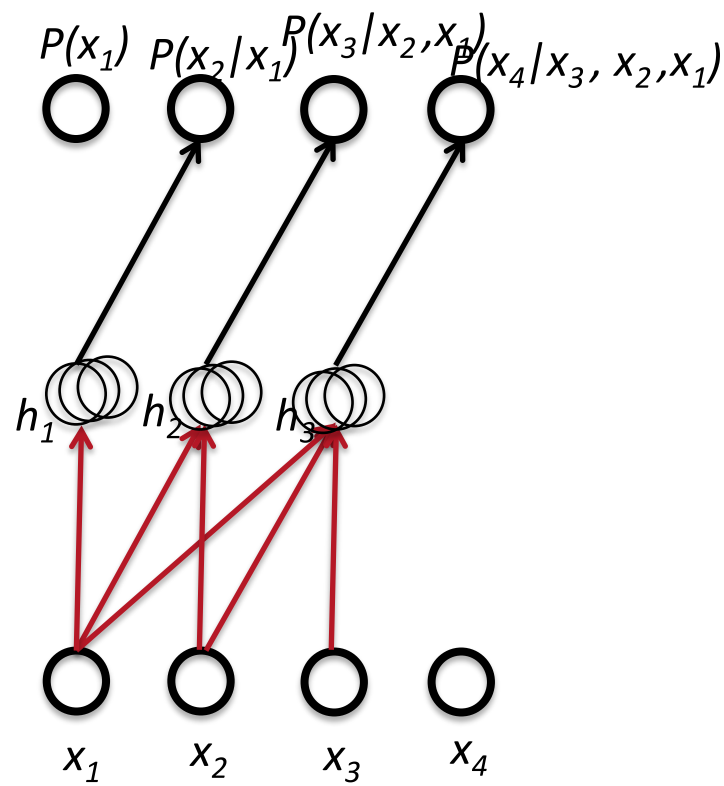
Neural Auto-Regressive Models

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



- Decomposes the joint of a fully observed directed model in terms of conditionals
- Logistic auto-regressive: *(Frey 1997)*



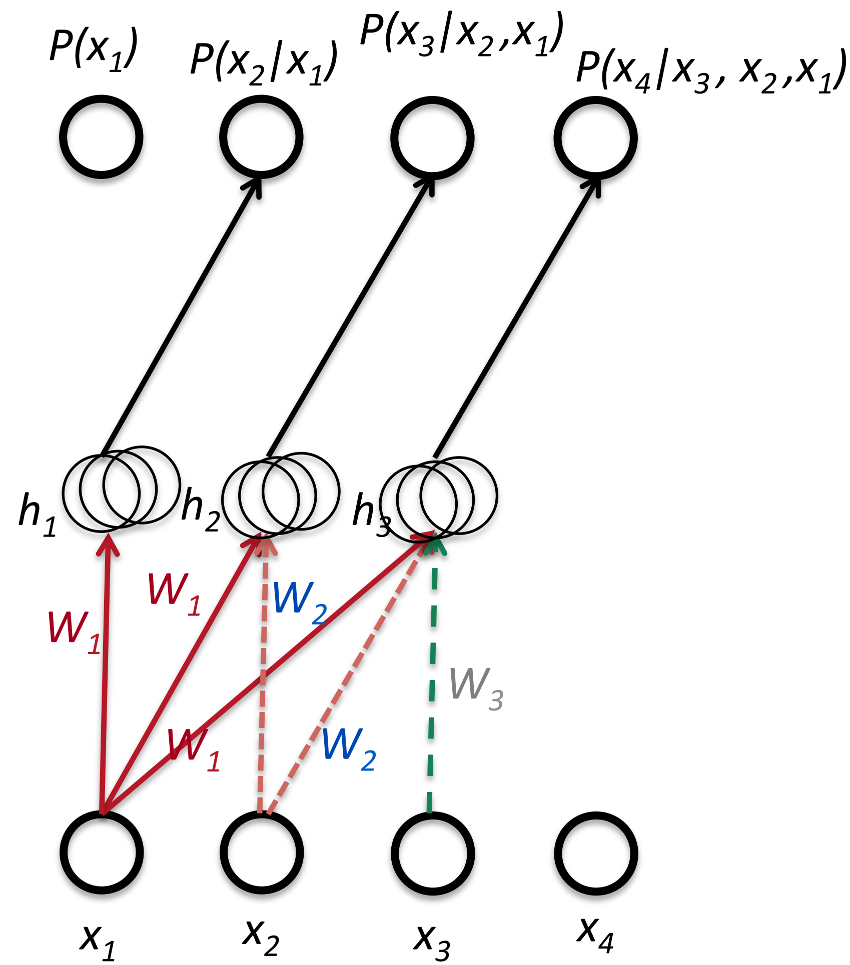


- First neural version: *(Bengio&Bengio NIPS'99)*

NADE: Neural AutoRegressive Density Estimator

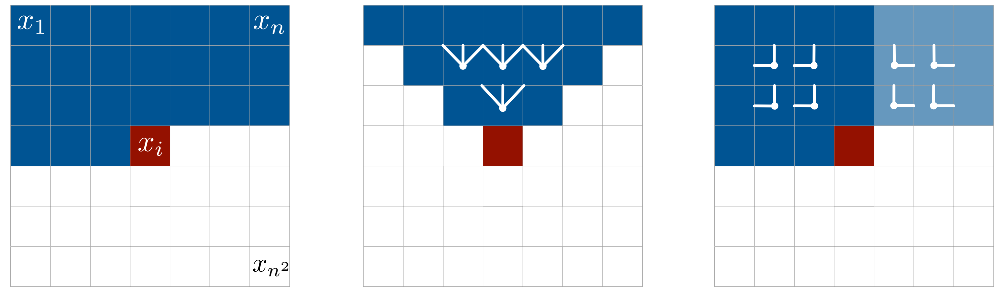
(Larochelle & Murray AISTATS 2011)

- Introduces smart sharing between some weights so that the different hidden groups use the same weights to the same input but look at more and more of the inputs.

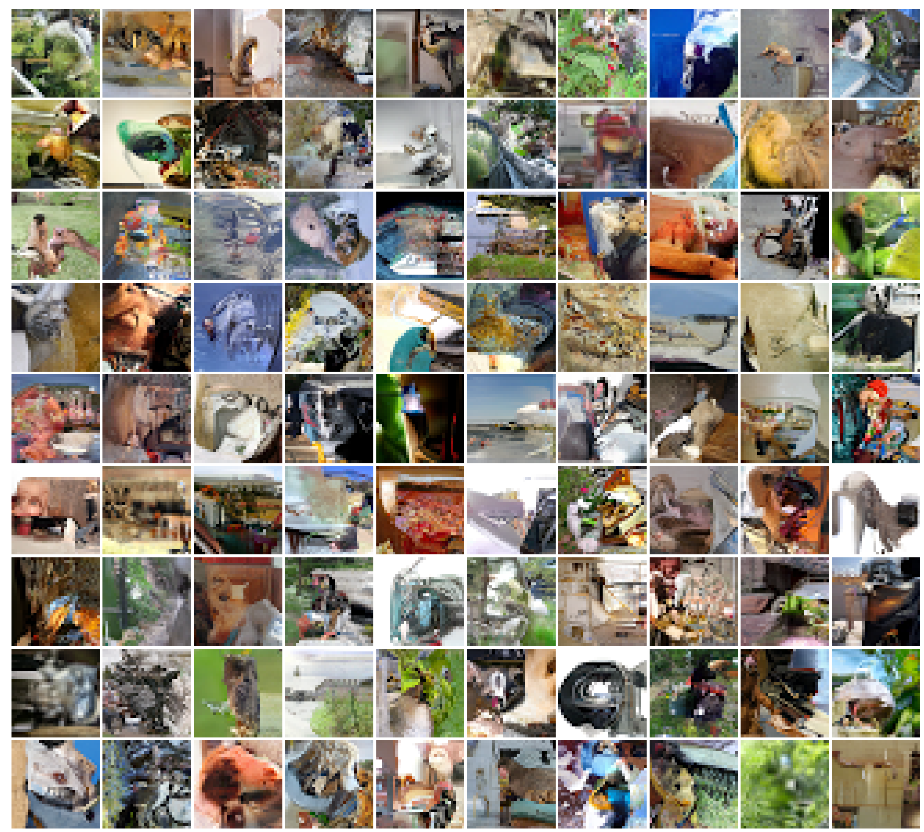


Pixel RNNs

(van den Oord et al ICML 2016, best paper)



- Similar to NADE and RNNs but for 2-D images
- Surprisingly sharp and realistic generation
- Gets texture right but not necessarily global structure



Forward Computation of the Gradient

- BPTT does not seem biologically plausible and is memory-expensive
- RTRL *(Real-Time Recurrent Learning, Williams & Zipser 1989, Neural Comp.)*
  - Practically useful: online learning, no need to store all the past states and revisit history backwards (which is biologically weird)
  - Compute the gradients forward in time, rather than backwards
    - Think about multiplying many matrices left-to-right vs right-to-left
  - **BUT** exact computation is O(nhidden x nweights) instead of O(nweights), to recursively compute dh(t)/dW ← all params
- Recently proposed, *approximate* the forward gradient using an efficient stochastic estimator (rank 1 estimator of dh/dW tensor) *(Training recurrent networks online without backtracking, Ollivier et al arXiv: 1507.07680)*

Montreal Institute for Learning Algorithms

MILA

Université de Montréal