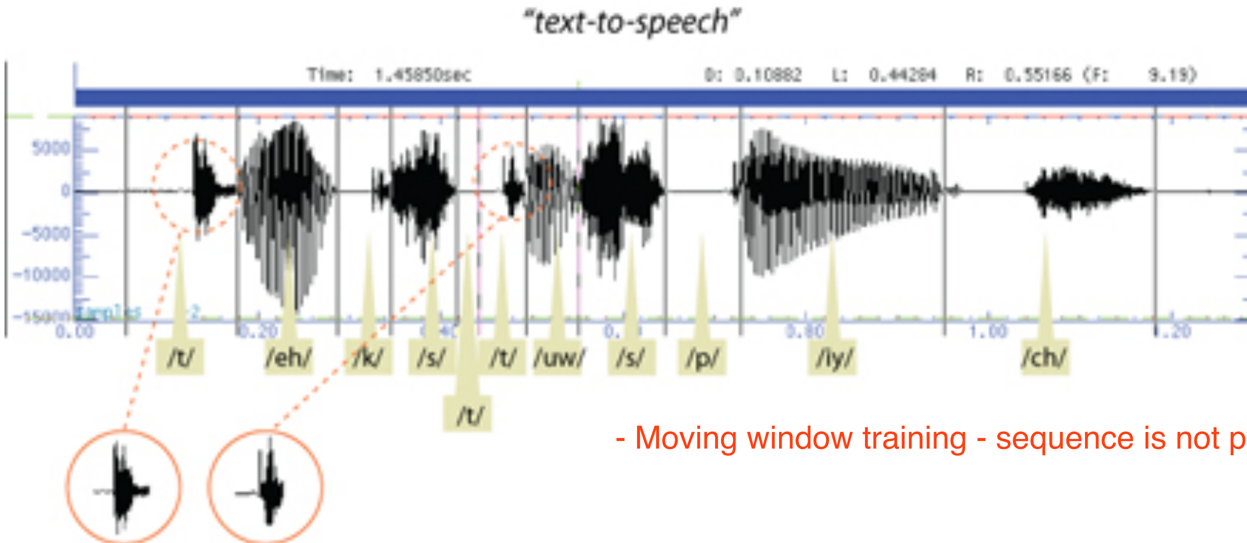
Lecture 17: RNN

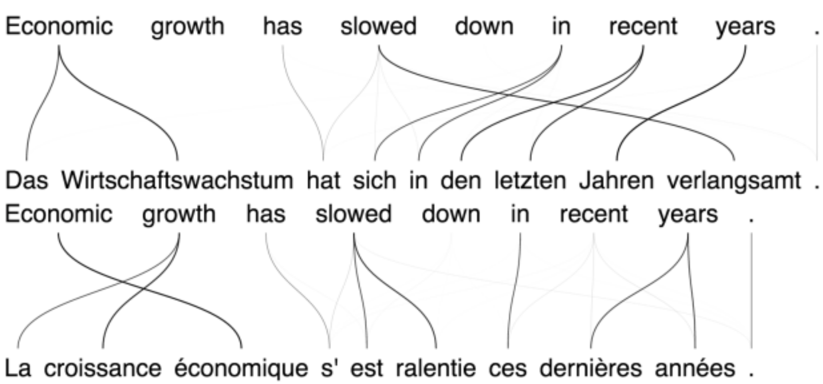
CS 109B, STAT 121B, AC 209B, CSE 109B

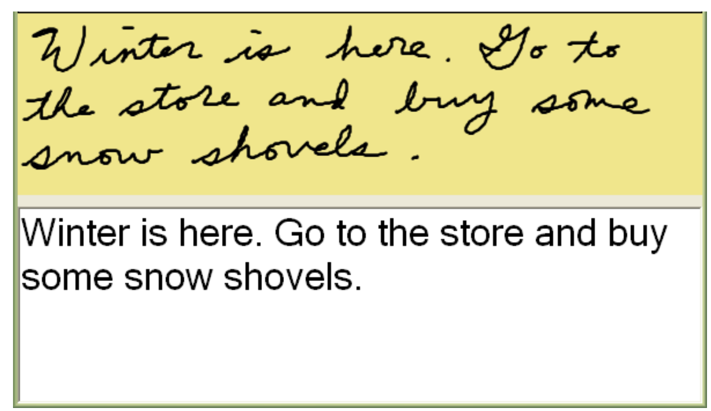
Mark Glickman and Pavlos Protopapas

Sequence Modeling



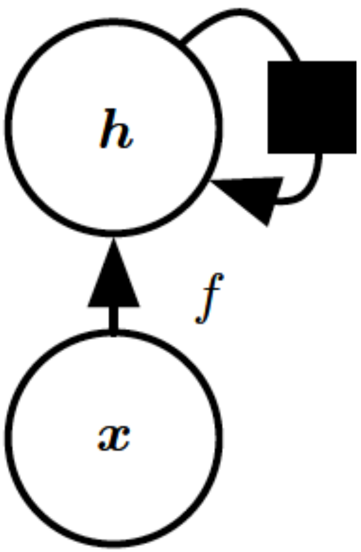
- Moving window training - sequence is not preserved



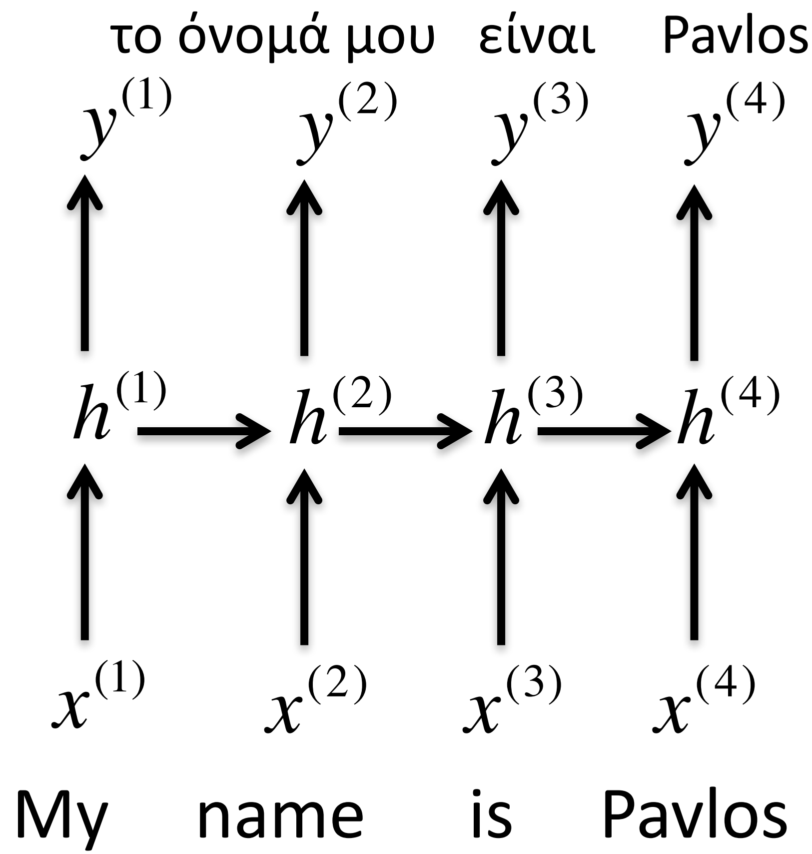


Recurrent Networks

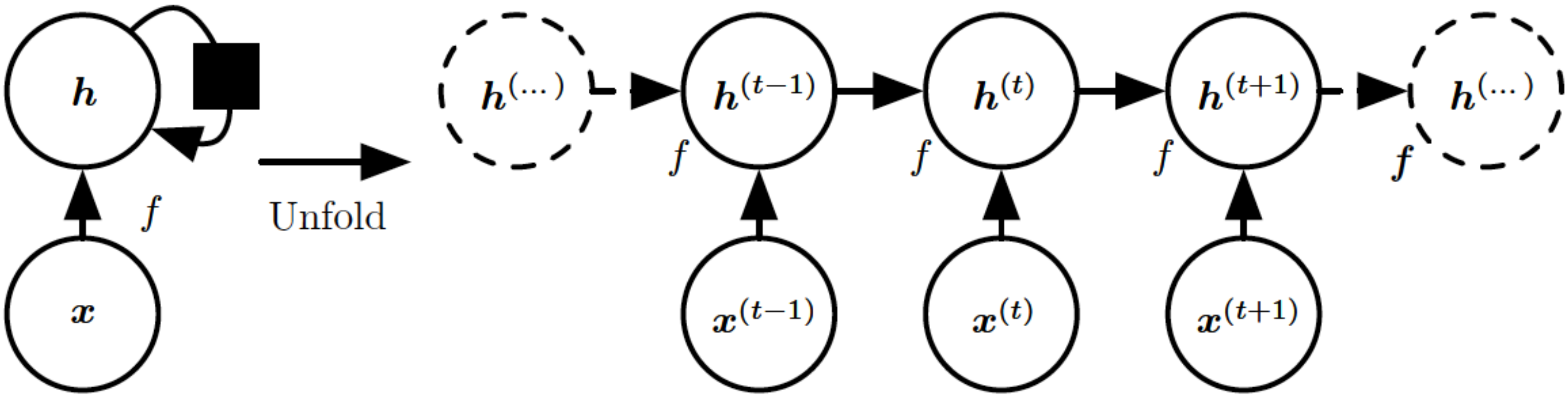
- Image/grid data: convolution networks
- Sequence data: parameter sharing across time



Example: Machine Translation



Unfolding the network

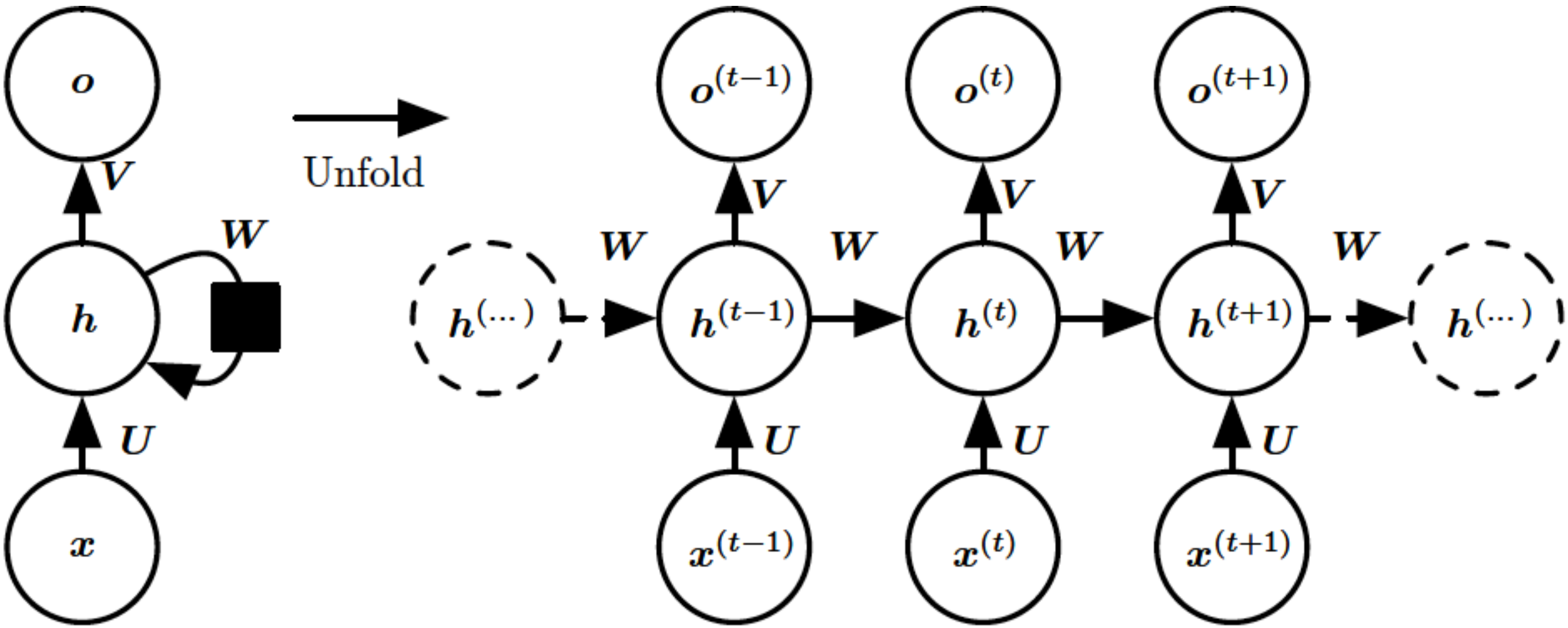


Input sequence

Sequence length may vary for each input

Hidden-to-hidden Recurrence

E.g. language traslation



Recurrent connections between hidden units

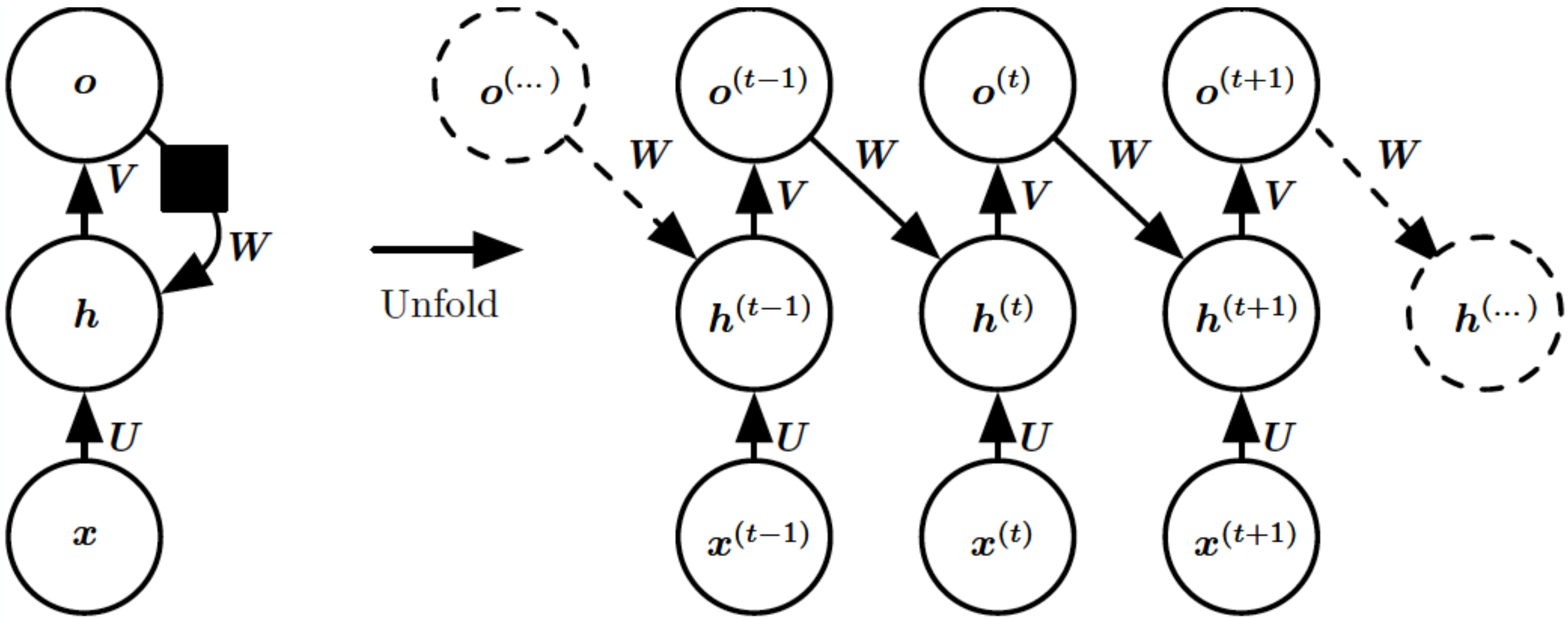
Hidden-to-hidden Recurrence

$$h^{(t)} = \sigma(\mathbf{W}h^{(t-1)} + \mathbf{U}x^{(t-1)} + b)$$

$$\hat{y}^{(t)} = \text{softmax}(\mathbf{V}h^{(t)} + c)$$

Output-to-output Recurrence

E.g. auto text completion

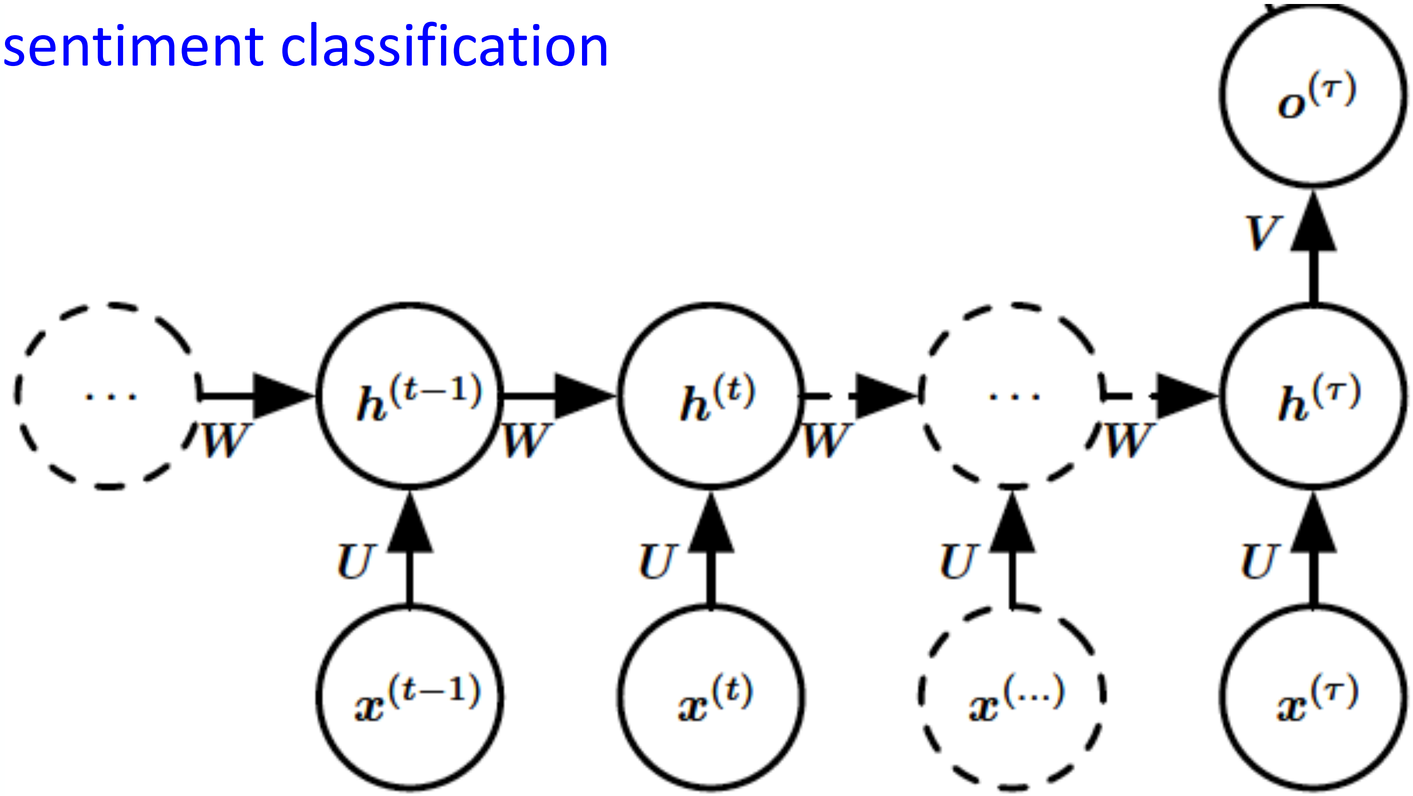


Recurrent connections between output and hidden units

Single Output RNN

Positive /Negative

E.g. sentiment classification



Product review

Output summarizes input sequence

Outline

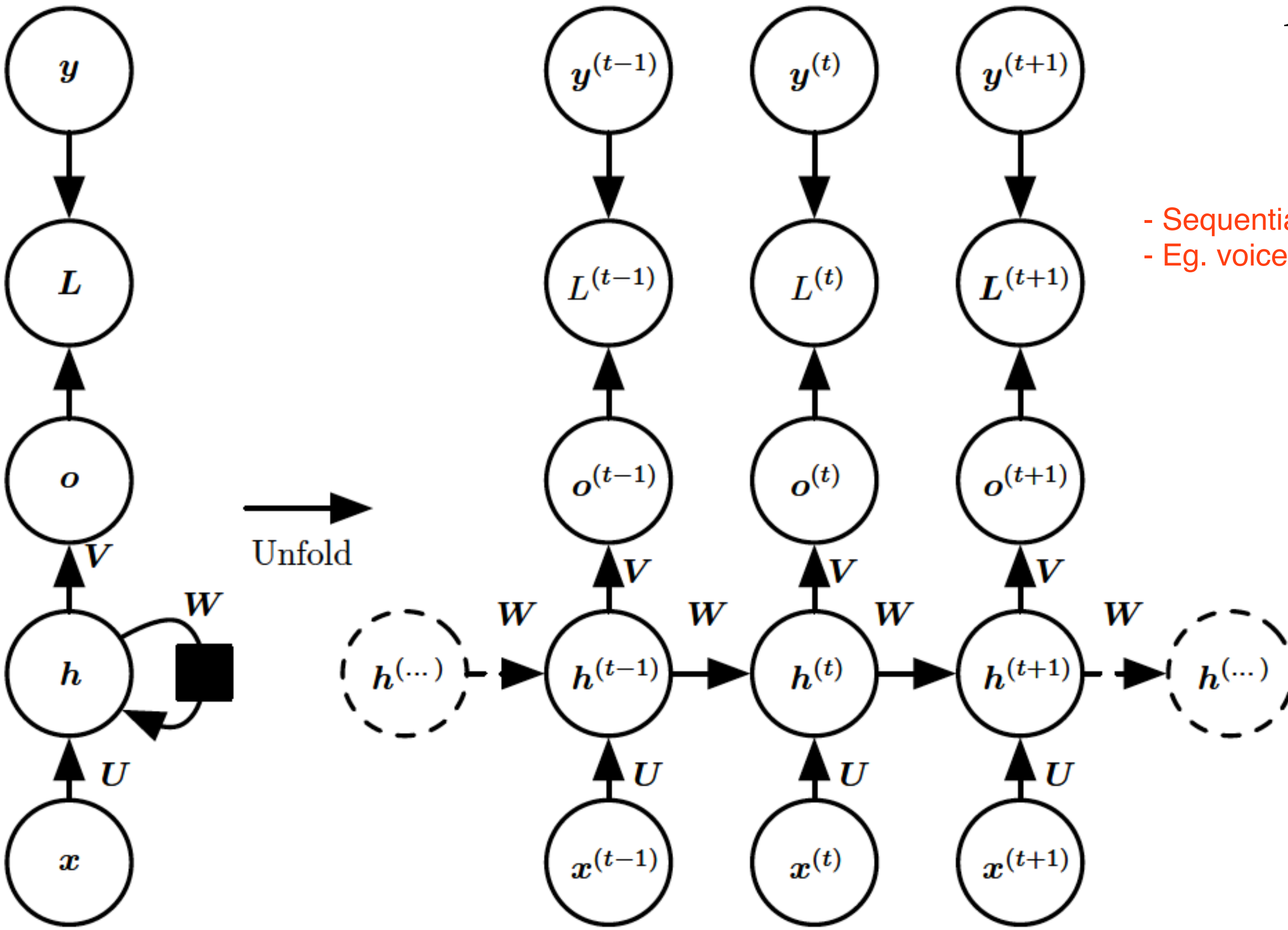
- RNN as a graphical model
- RNN training
- Long-term dependencies
- Gated RNN
- RNN variants

Loss Computation

Target outputs

$$L = \sum_t L^{(t)}$$

- Sequential - cannot be parallelised
- Eg. voice to text: each word an x



Conditioned on Target Outputs

- Log-likelihood (cross-entropy)

$$-\log P\left(o^{(t)} = y^{(t)} \middle| x^{(1)},\ldots,x^{(t)},\ y^{(1)},\ldots,y^{(t-1)}\right)$$

Prediction at *t*

Target at *t*

Conditioned on Target Outputs

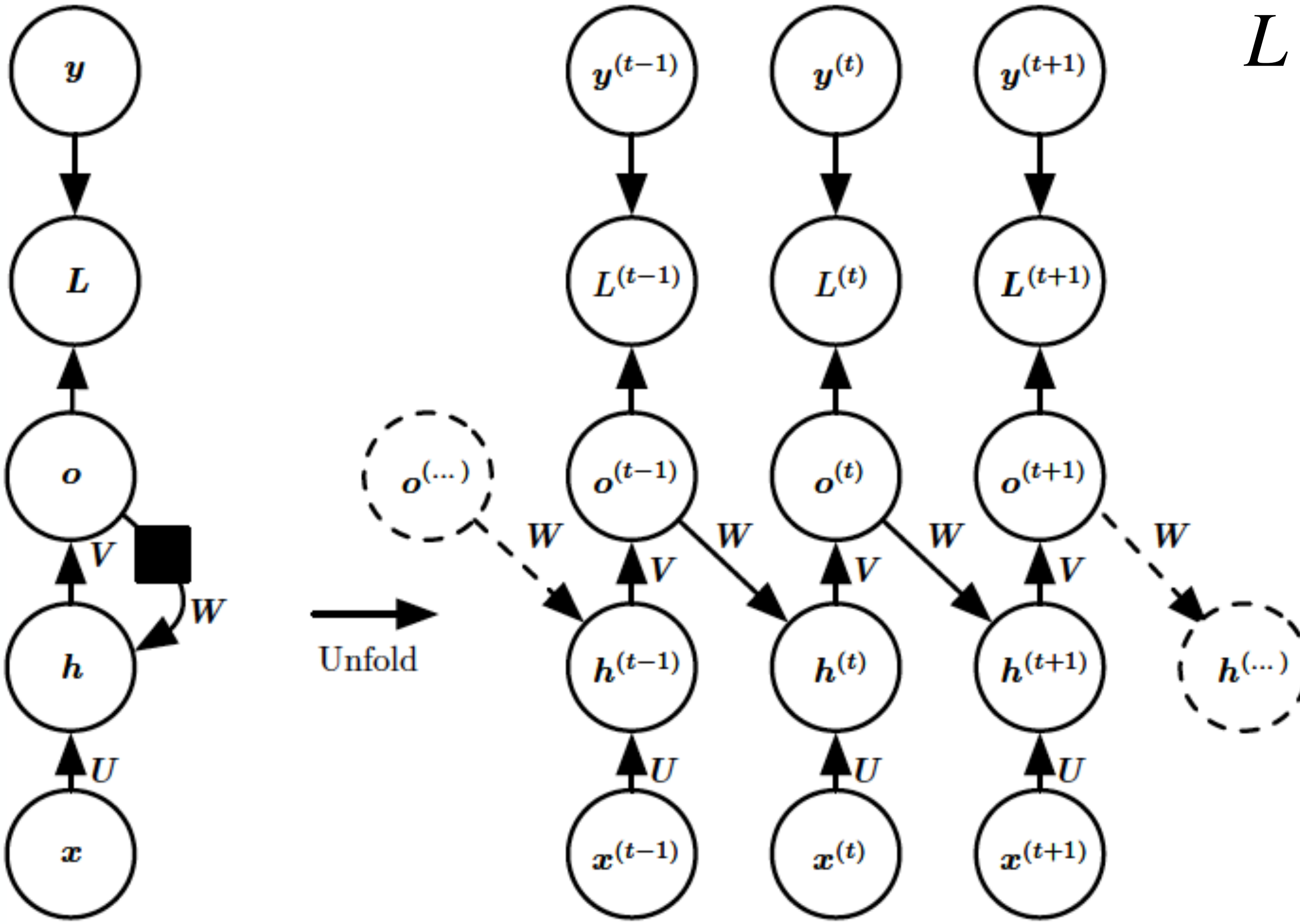
- Log-likelihood (cross-entropy)

$$-\log P\left(\underbrace{o^{(t)}}_{\text{Prediction at } t} = \underbrace{y^{(t)}}_{\text{Target at } t} \middle| x^{(1)},\ldots,x^{(t)},\ y^{(1)},\ldots,y^{(t-1)}\right)$$

- Conditioned on past inputs and outputs, output at time *t* is independent of future outputs

Conditioned on Predicted Outputs

$$L = \sum_t L^{(t)}$$



Conditioned on Predicted Outputs

- Log-likelihood

Past predictions
instead of true outputs

$$-\log P\left(o^{(t)} = y^{(t)} \middle| x^{(1)}, \ldots, x^{(t)},\ \underleftrightarrow{o^{(1)}, \ldots, o^{(t-1)}}\right)$$

Prediction at *t*

Target at *t*

- Likelihood function can break into pieces

Conditioned on Predicted Outputs

- Log-likelihood

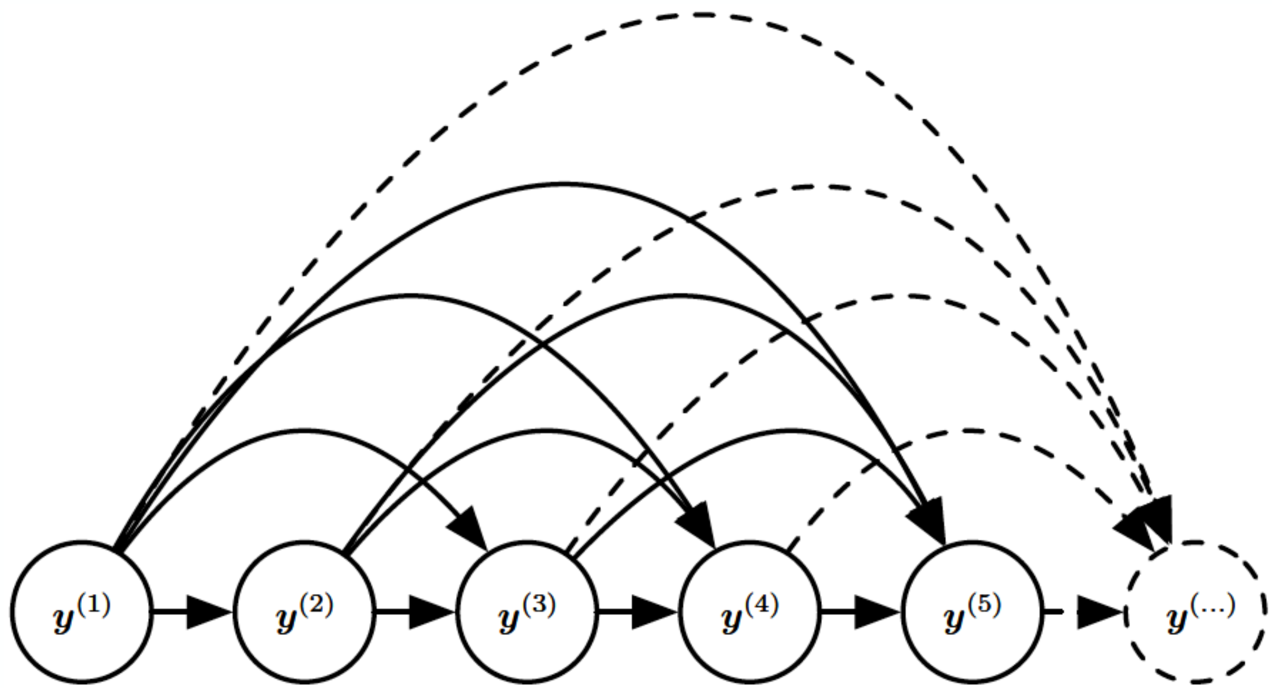
Past predictions
instead of true outputs

$$-\log P\left(\underbrace{o^{(t)}}_{\text{Prediction at } t} = \underbrace{y^{(t)}}_{\text{Target at } t} \middle| x^{(1)}, \ldots, x^{(t)},\ \underleftrightarrow{o^{(1)}, \ldots, o^{(t-1)}} \right)$$

- Conditioned on inputs, output at time *t* is independent of everything else

Fully-connected graphical model

Simple example: *Predict day's stock prices based on previous prices*

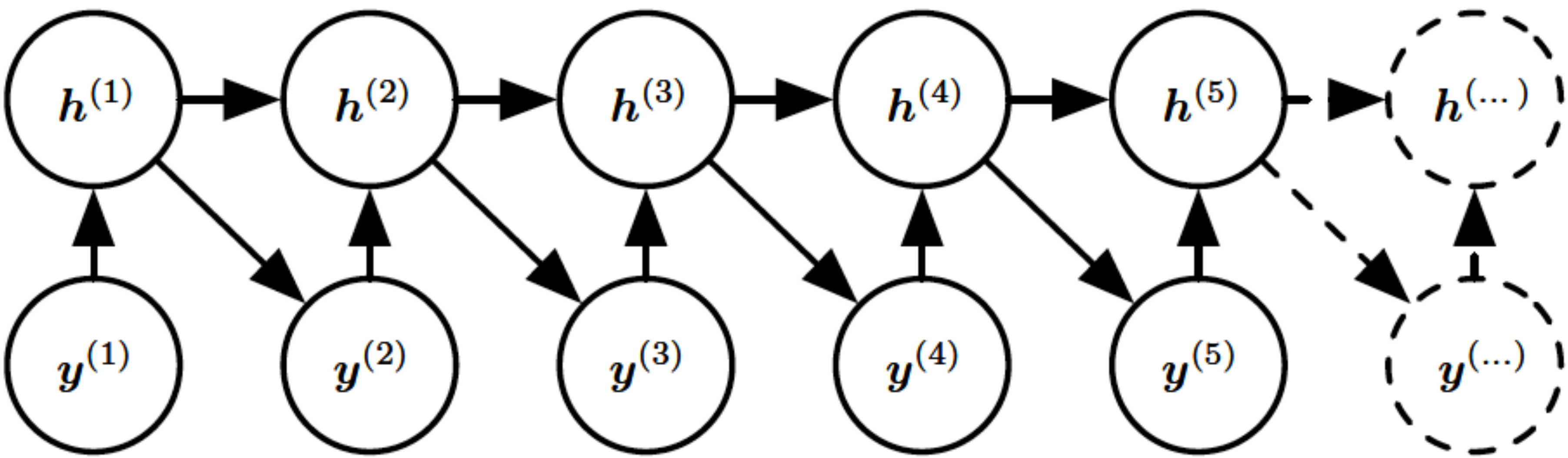


Inefficient parametrization

RNN graphical model

Simple example: *Predict day's stock prices based on previous prices*

Graphical model details what we cannot get out of writing equation directly



Efficient parametrization,
but stationary distribution

Outline

- RNN as a graphical model
- RNN training
- Long-term dependencies
- Gated RNN
- RNN variants

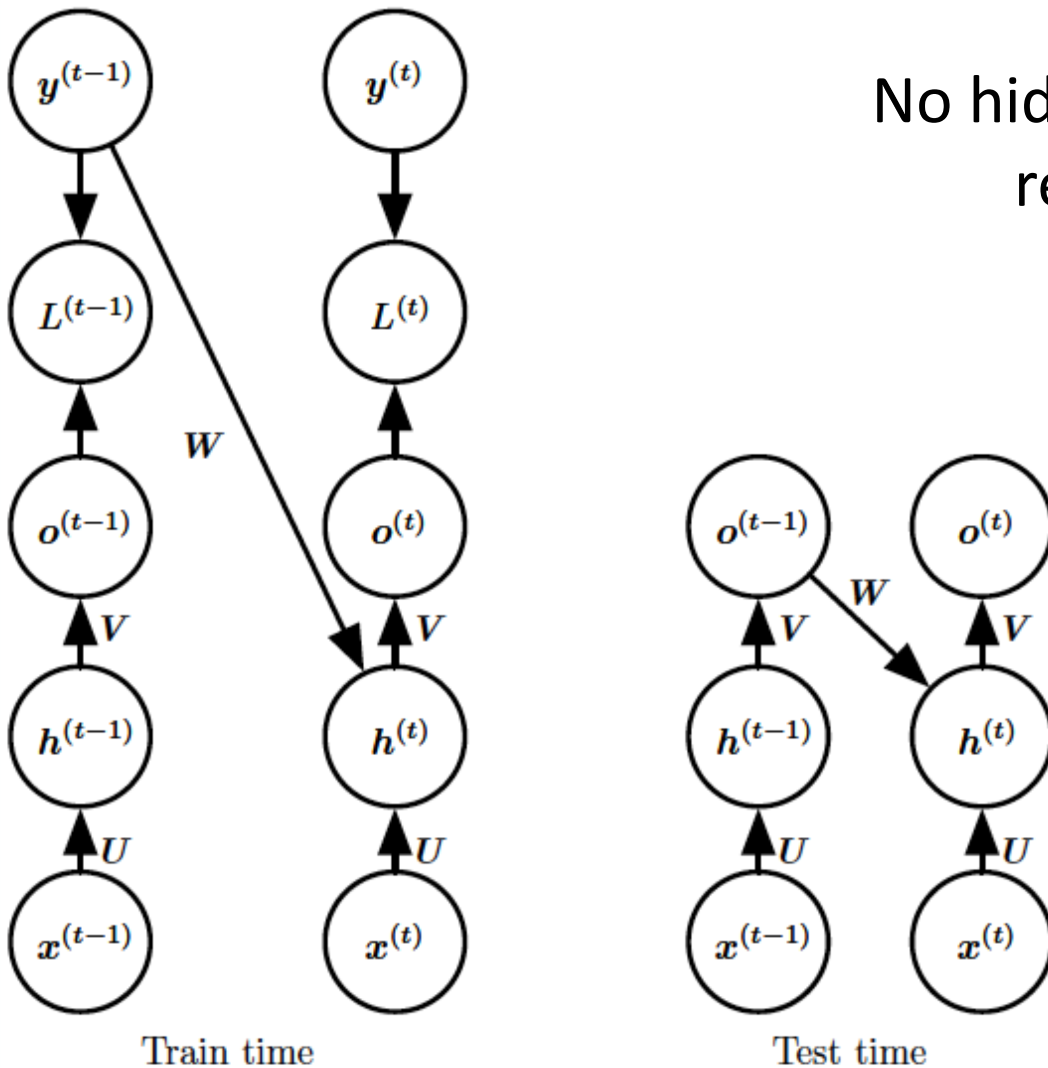
Backprop Through Time

- For each input, unfold network for the sequence length T
- Back-propagation: apply forward and backward pass on unfolded network
- Memory cost: $O(T)$

- Same as batch propagation

Case of Output Recurrence

No hidden-to-hidden recurrence



Case of Output Recurrence

Loss at time t:

Teacher Forcing

$$L^{(t)} = -\log P\left(o^{(t)} = y^{(t)} \middle| x^{(1)}, \ldots, x^{(t)},\ y^{(1)}, \ldots, y^{(t-1)}\right)$$

Use ground truth from previous time steps

Loss at different time steps are *decoupled*

Outline

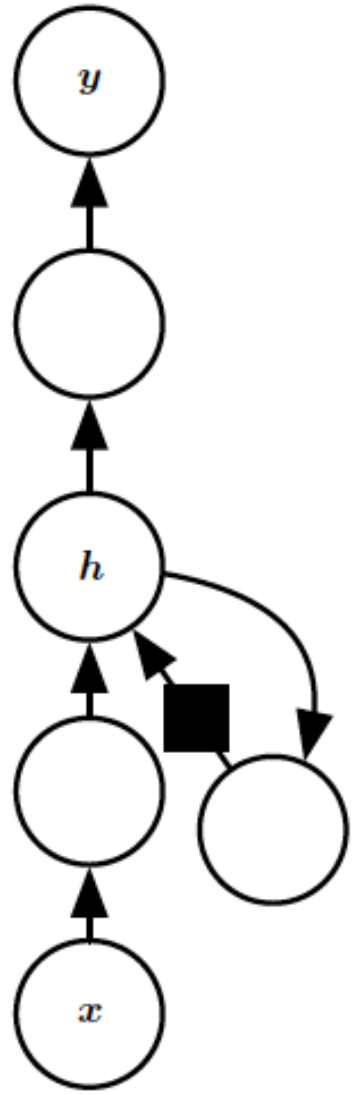
- RNN as a graphical model
- RNN training
- Long-term dependencies
- Gated RNN
- RNN variants

Deep Recurrent Nets

- Hard to train as more layers added
- Problem with BP over time: long seq. - idea of vanishing gradient persists

Multiple layers between recurrent state and output

Multiple layers between input and recurrent state



Multiple layers between current and previous hidden states

Long-term Dependencies

- Unfolded networks can be very deep
- Long-term interactions are given exponentially smaller weights than small-term interactions
- Gradients tend to either *vanish* or *explode*

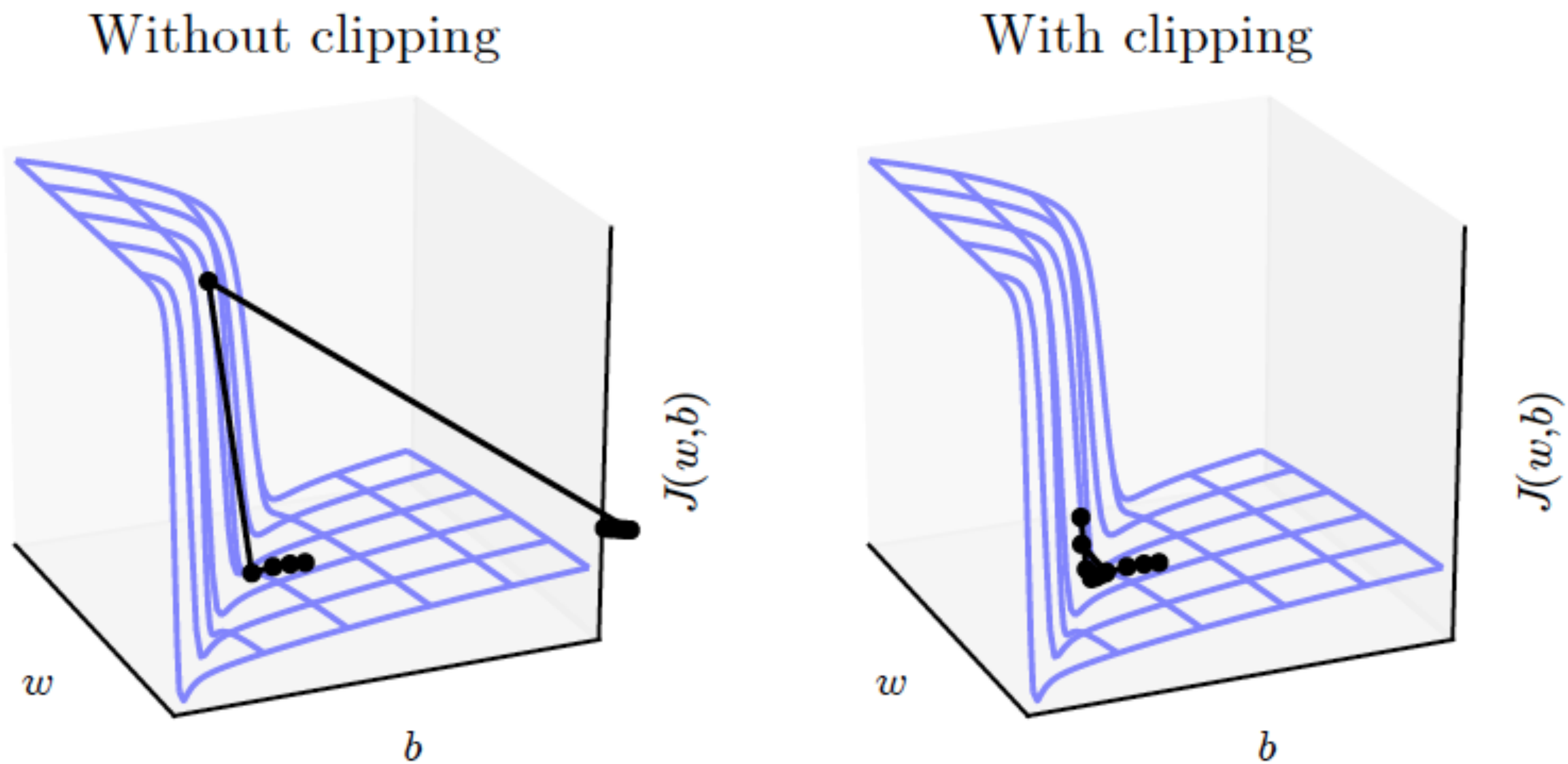
Gradient Clipping

- Prevents exploding gradients
- Clip the norm of gradient before update:

$$\text{if } ||\boldsymbol{g}|| > v$$

$$\boldsymbol{g} \leftarrow \frac{\boldsymbol{g}v}{||\boldsymbol{g}||}$$

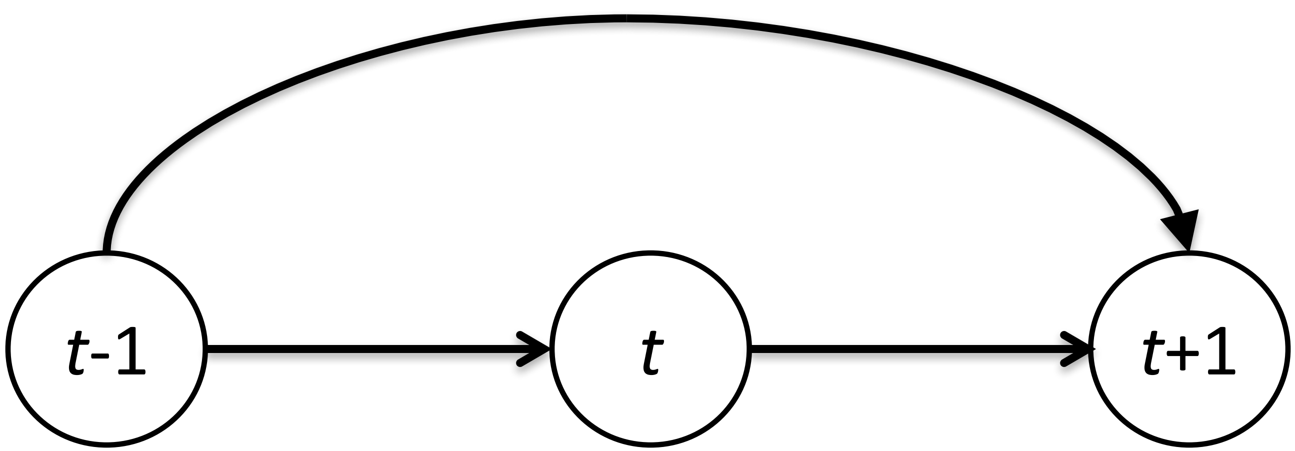
Gradient Clipping



Skip Connections

- Add additional connections between units *d* time steps apart
- Creating paths through time where gradients neither vanish or explode

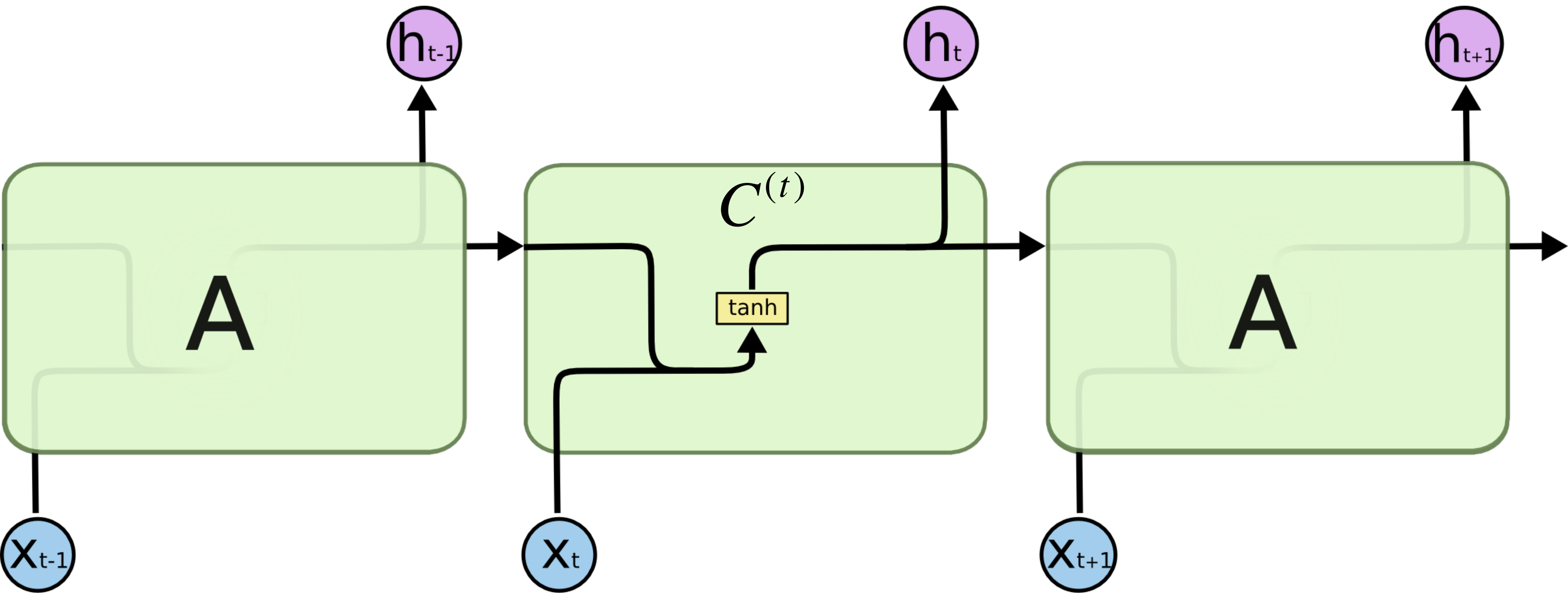
-Prevent extreme gradients



Leaky Units

- Linear self-connections
- Maintain cell state: running average of past hidden activations

Standard RNN

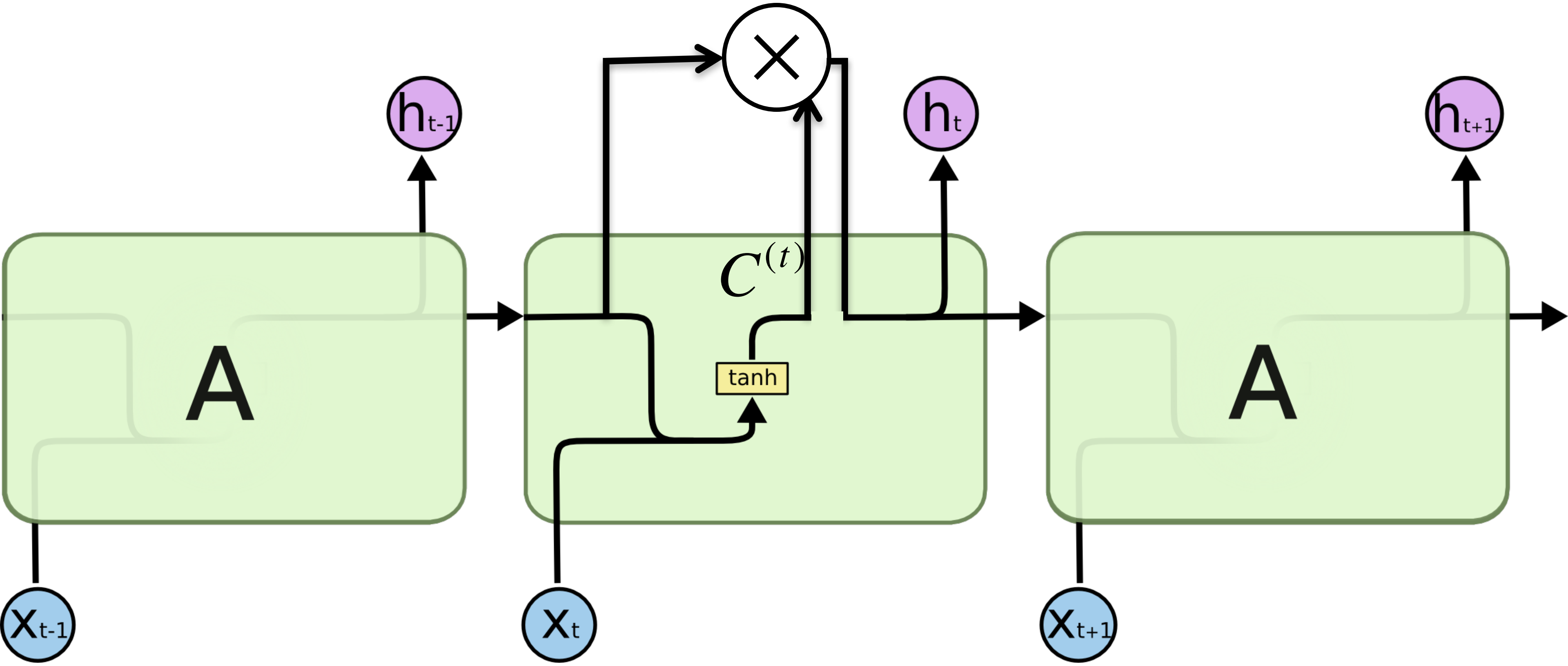


$$C^{(t)} = \tanh(\mathbf{W}h^{(t-1)} + \mathbf{U}x^{(t-1)})$$

$$h^{(t)} = C^{(t)}$$

colah.github.io

Leaky Unit



$$C^{(t)} = \tanh(\mathbf{W}h^{(t-1)} + \mathbf{U}x^{(t-1)})$$

$$h^{(t)} = \alpha h^{(t-1)} + (1\text{-}\alpha)C^{(t)}$$

colah.github.io

Outline

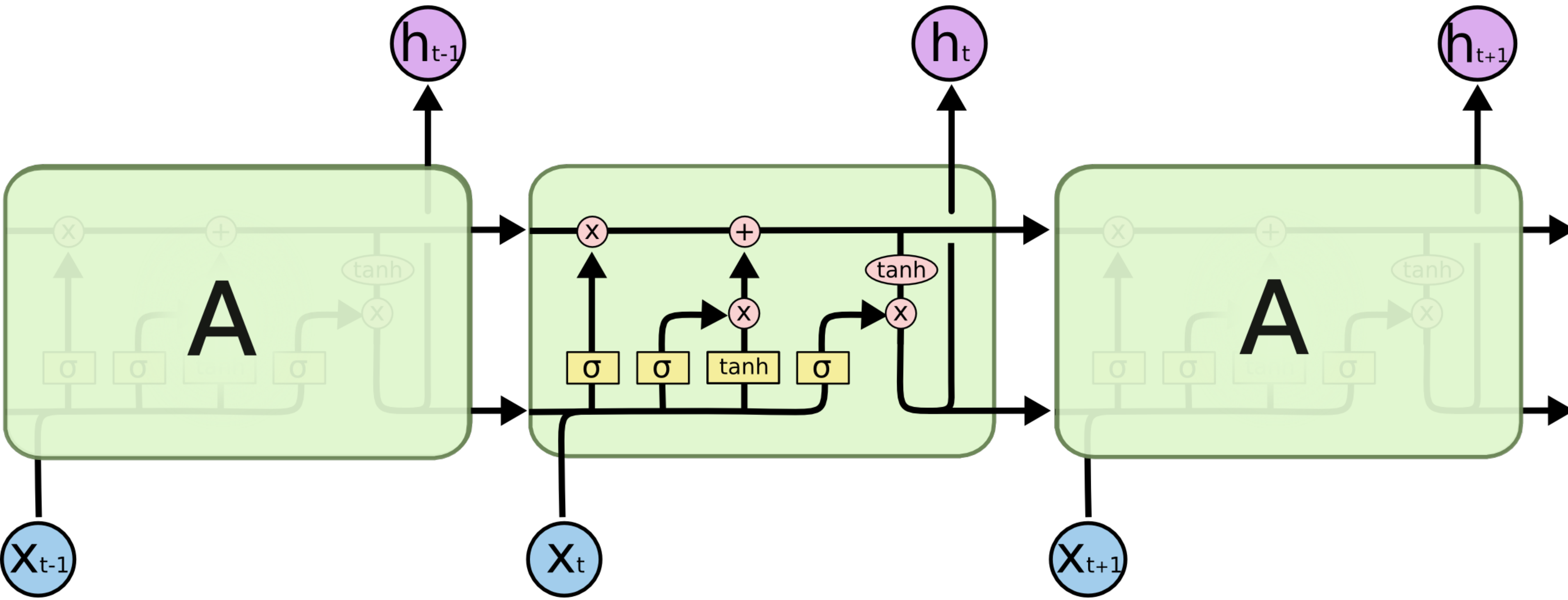
- RNN as a graphical model
- RNN training
- Long-term dependencies
- Gated RNN
- RNN variants

Long Short-Term Memory

- Handles long-term dependencies
- Leaky units where weight on self-loop α is context-dependent
- Allow network to decide whether to accumulate or forget past info

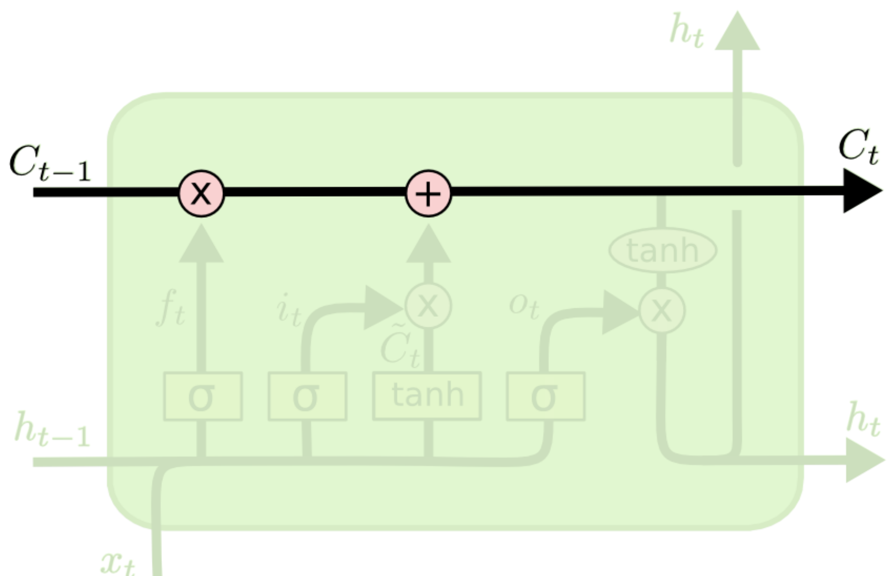
- Type of RNN
- Popular - remember what we have learnt so far

Long Short-Term Memory



colah.github.io

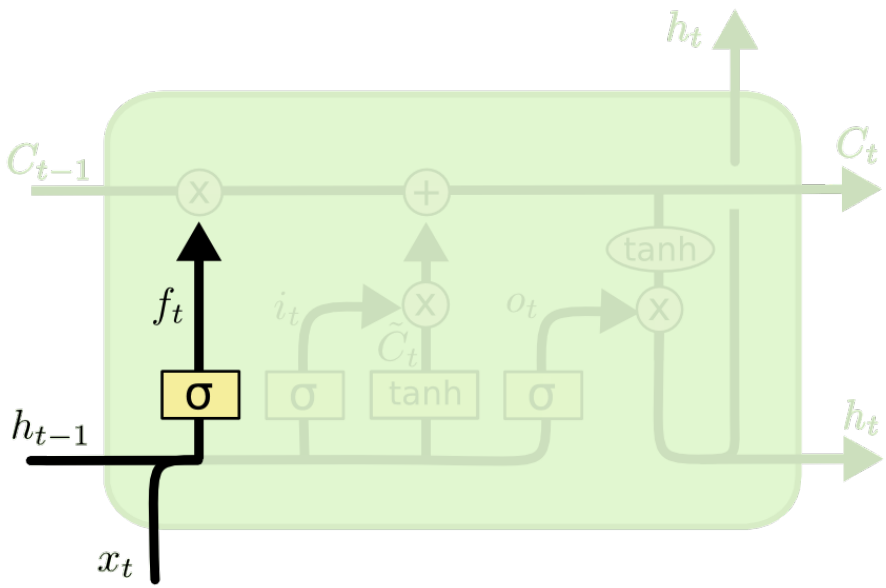
Cell State



colah.github.io

Forget Gate

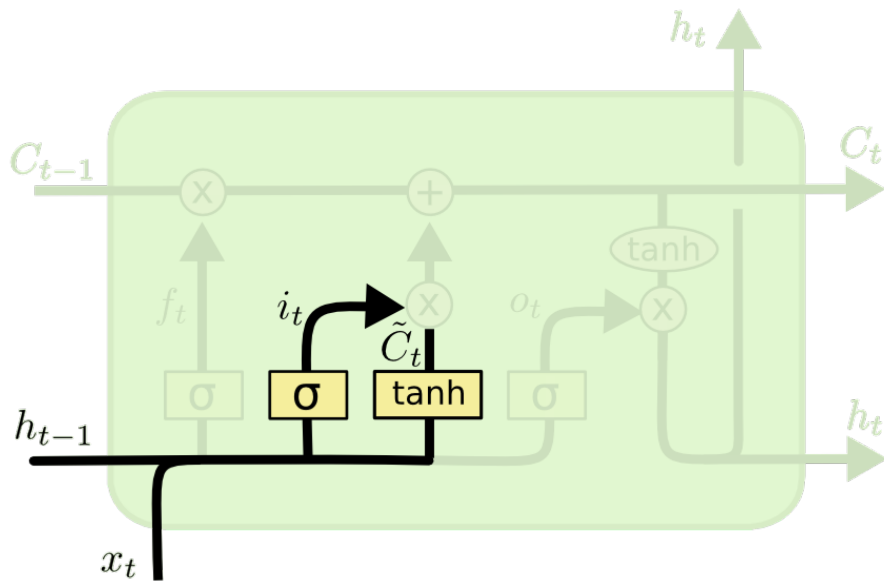
Intuition:
- Capture what we have learnt so far - what to remember and how much
- Prior given more weight than new



$$f^{(t)} = \sigma(W^f h^{(t-1)} + U^f x^{(t)})$$

colah.github.io

Input Gate

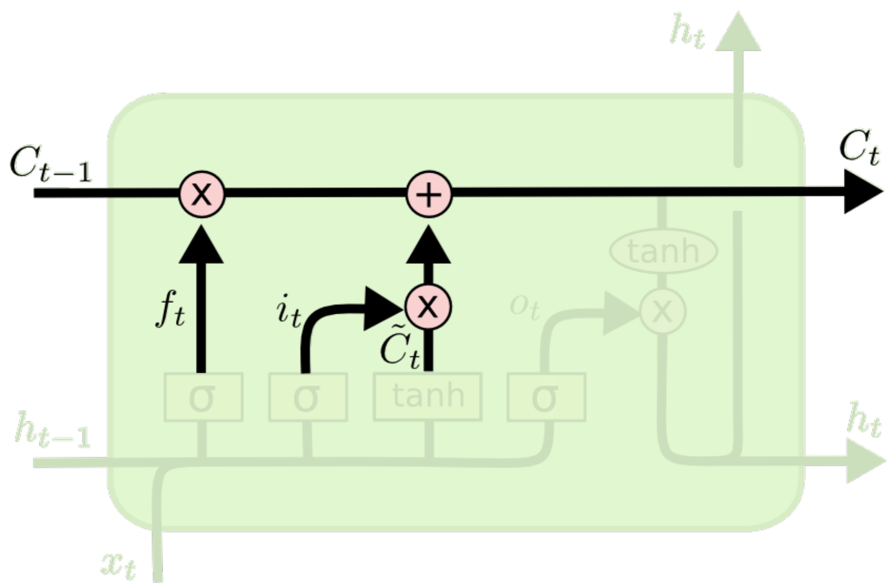


$$i^{(t)} = \sigma(W^i h^{(t-1)} + U^i x^{(t)})$$

$$\tilde{C}^{(t)} = \tanh(W h^{(t-1)} + U x^{(t)})$$

colah.github.io

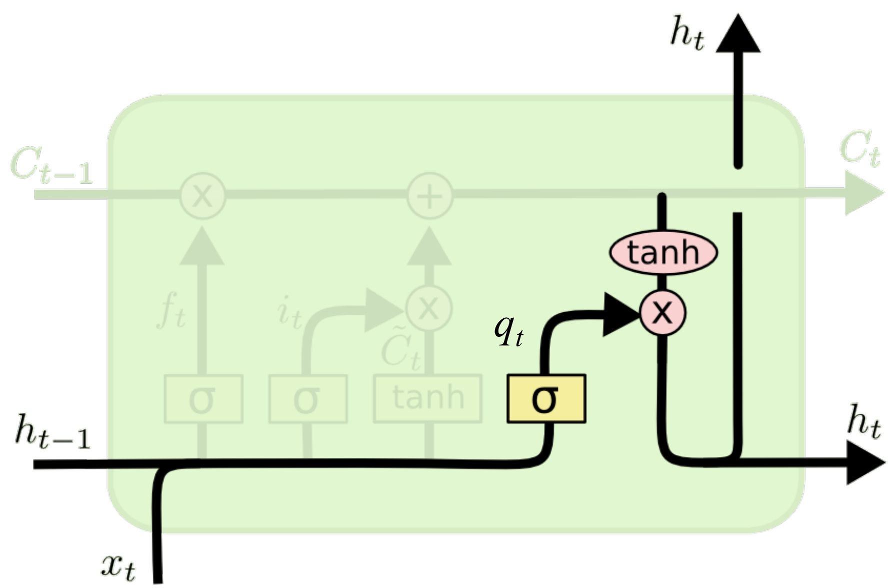
Cell State Update



$$C^{(t)} = f^{(t)}C^{(t-1)} + i^{(t)}\tilde{C}^{(t)}$$

colah.github.io

Output Gate



$$q^{(t)} = \sigma(W^o h^{(t-1)} + U^o x^{(t)})$$

$$h^{(t)} = \tanh(C^{(t)})$$

colah.github.io

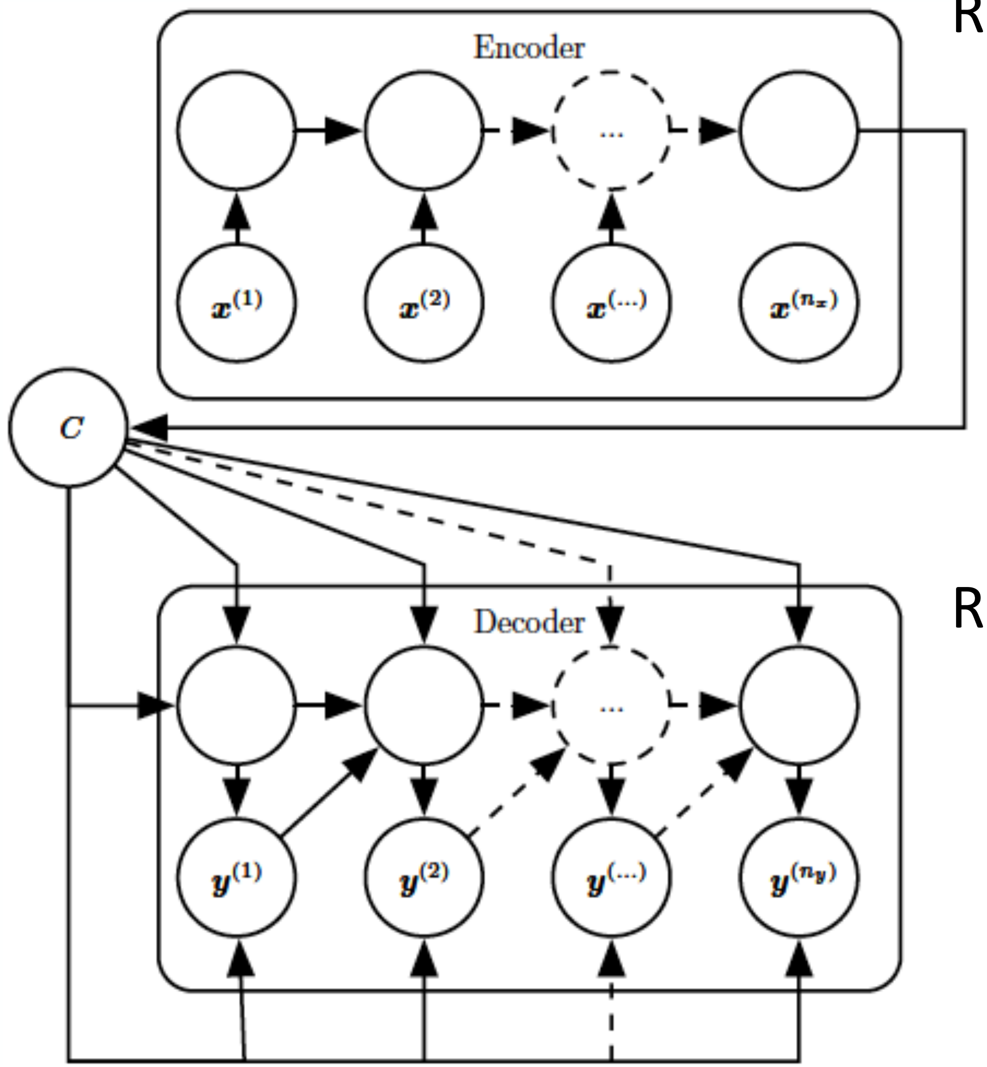
Outline

- RNN as a graphical model
- RNN training
- Long-term dependencies
- Gated RNN
- RNN variants

Encoder-decoder Networks

- Compressing sequence into something smaller
- Echo State: One type of NN: only need to learn final weights, not all

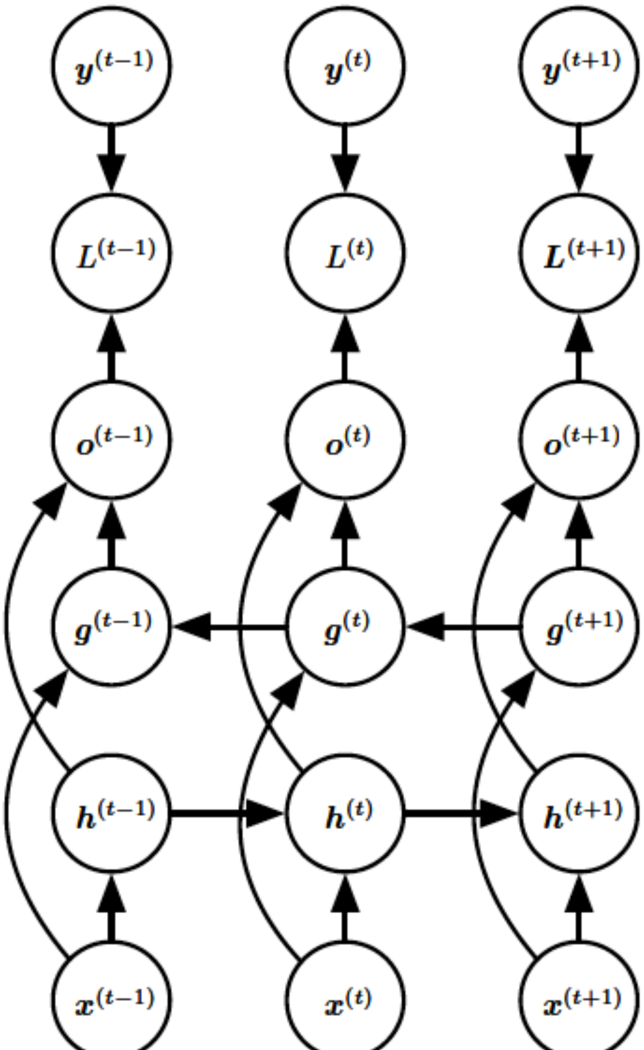
RNN



RNN

RNN output sequence can be of different length than input sequence

Bidirectional Network

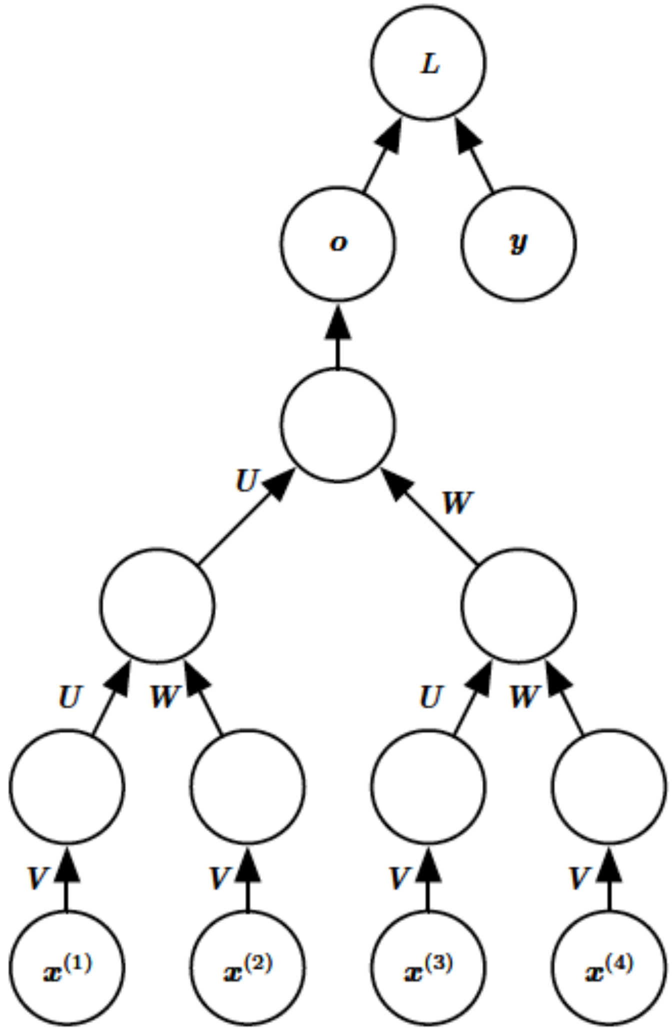


Output prediction may depend on *whole input sequence*

E.g. speech recognition: current sound may depend on future phonemes

Backprop?

Recursive Network



Tree structure vs. chain
E.g. parse tree in NLP

Reduce network depth by using taller trees