

Lecture 13: Recurrent Neural Networks I: Basics

Xuming He
SIST, ShanghaiTech
Fall, 2019

Outline

- Recurrent Neural Networks
 - Sequence modeling problem
 - Autoregressive models
 - (Vanilla) RNN models
- Backpropagation through time
 - Computational graph
- Example: language modeling
 - Neural language models

Acknowledgement: Feifei Li et al's cs231n notes

Sequence modeling

- Modeling a sequence of tokens
 - Running example: sentences
- Goal: learn/build a good distribution of sentences
- Inputs: a corpus of sentences $s^{(1)}, \dots, s^{(N)}$
- Output: a distribution $p(s)$
- Common approach: **maximum likelihood**
 - Assume sentences are independent

$$\max \prod_{i=1}^N p(s^{(i)})$$

Sequence modeling

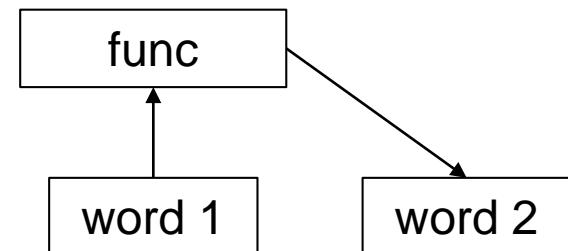
- What is $p(\mathbf{s})$?
- A sentence is a sequence of words w_1, w_2, \dots, w_T .

$$p(\mathbf{s}) = p(w_1, \dots, w_T) = p(w_1)p(w_2 | w_1) \cdots p(w_T | w_1, \dots, w_{T-1}).$$

- Essentially aim to predict the next word
- Markovian assumption
 - The distribution over the next word depends on the preceding few words. For example,

$$p(w_t | w_1, \dots, w_{t-1}) = p(w_t | w_{t-3}, w_{t-2}, w_{t-1}).$$

- Autoregressive model
 - Memoryless
 - Can be modeled by a parametrized function



Traditional language models

■ N-Gram model

- Autoregressive model: Markov assumption
- Use a conditional probability table

	cat	and	city	...
the fat	0.21	0.003	0.01	
four score	0.0001	0.55	0.0001	...
New York	0.002	0.0001	0.48	
:	:	:		

- Estimate the probabilities from the empirical distribution

$$p(w_3 = \text{cat} | w_1 = \text{the}, w_2 = \text{fat}) = \frac{\text{count}(\text{the fat cat})}{\text{count}(\text{the fat})}$$

- The phrases we're counting are called **n-grams** (where n is the length), so this is an n-gram language model.
 - Note: the above example is considered a 3-gram model, not a 2-gram model!

Traditional language models

- Problems with n-gram language models
 - The number of entries in the conditional probability table is exponential in the context length
 - Data sparsity: most n-grams never appear in the corpus
- Solutions
 - Use a short context (less expressive)
 - Smooth the probabilities (priors)
 - Using an ensemble of n-gram models with different n

Neural language model

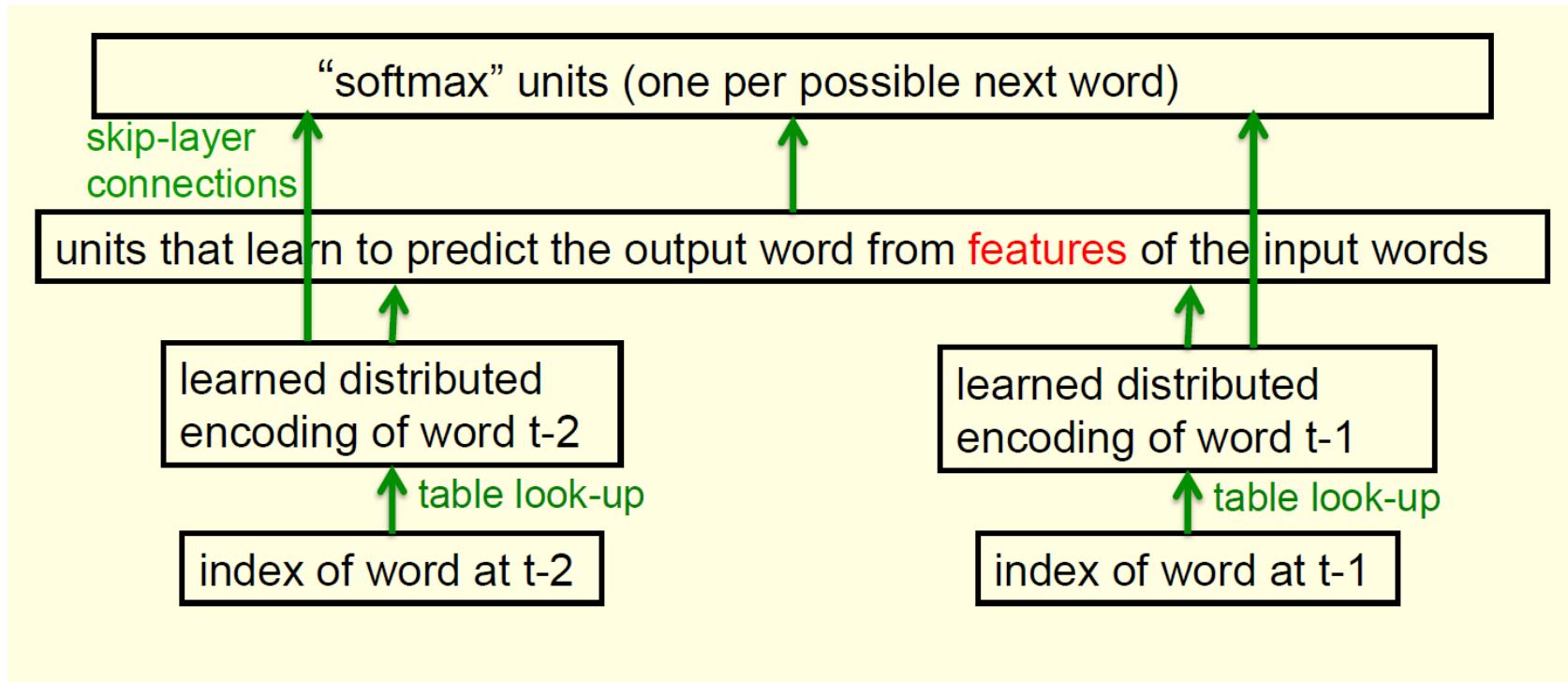
- Predicting the distribution of the next word given the previous K is a multiway classification problem
 - Inputs: previous K words
 - ✓ Output/Target: next word
 - Loss: cross-entropy

$$\begin{aligned}-\log p(\mathbf{s}) &= -\log \prod_{t=1}^T p(w_t | w_1, \dots, w_{t-1}) \\&= -\sum_{t=1}^T \log p(w_t | w_1, \dots, w_{t-1}) \\&= -\sum_{t=1}^T \sum_{v=1}^V t_{tv} \log y_{tv},\end{aligned}$$

where t_{iv} is the one-hot encoding for the i th word and y_{iv} is the predicted probability for the i th word being index v .

Neural language model

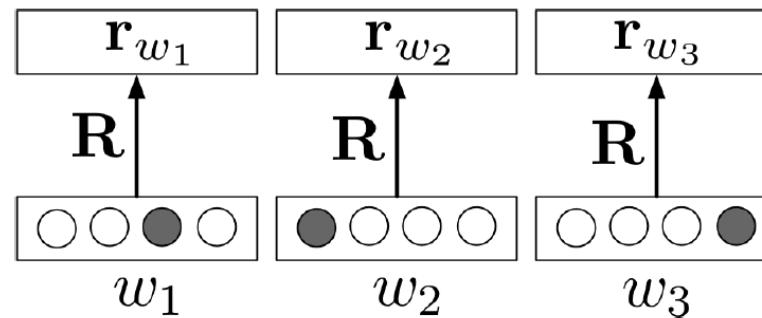
■ Model structure (context length = 2)



Neural language model

Word embedding

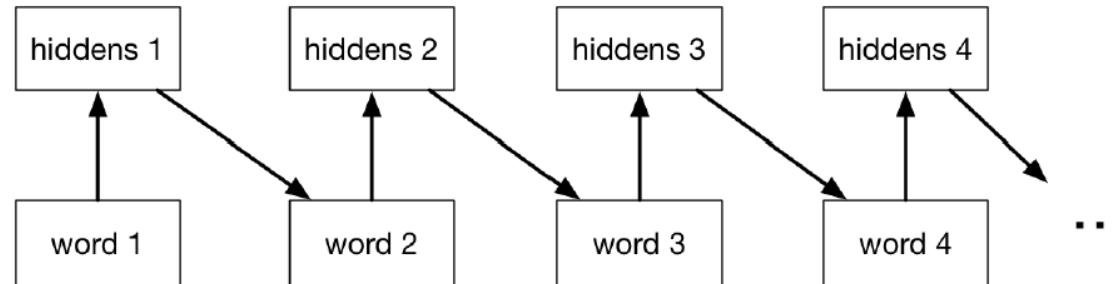
- If we use a 1-of-K encoding for the words, the first layer can be thought of as a linear layer with **tied weights**.



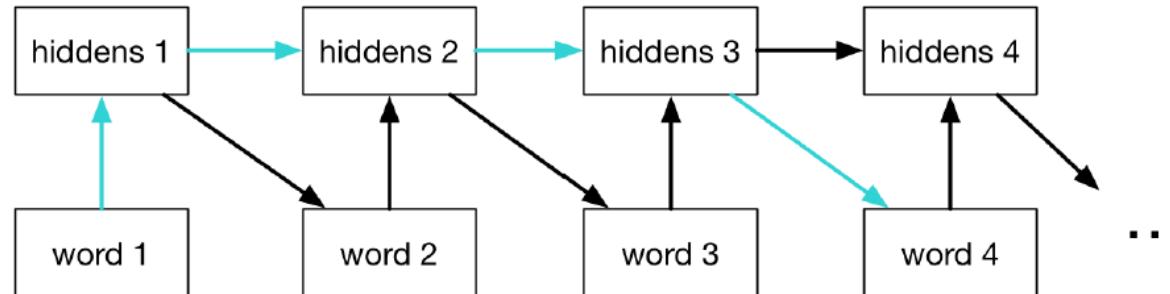
- The weight matrix basically acts like a lookup table. Each column is the **representation** of a word, also called an **embedding**, **feature vector**, or **encoding**.
 - “Embedding” emphasizes that it’s a location in a high-dimensional space; words that are closer together are more semantically similar
 - “Feature vector” emphasizes that it’s a vector that can be used for making predictions, just like other feature mappings we’ve looked at (e.g. polynomials)

Sequence modeling

- Problems?
- Autoregressive models are memoryless
 - Can only use information from their immediate context

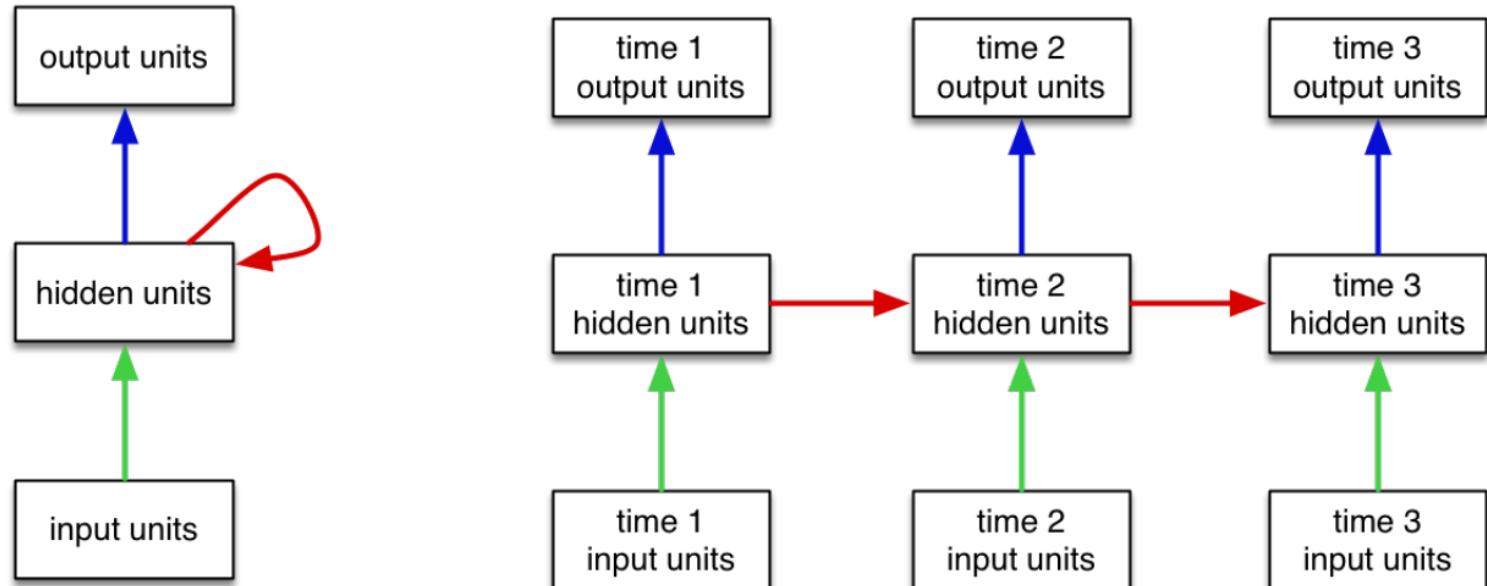


- Adding connections between hidden units
 - Having a memory lets the model use longer-term dependencies



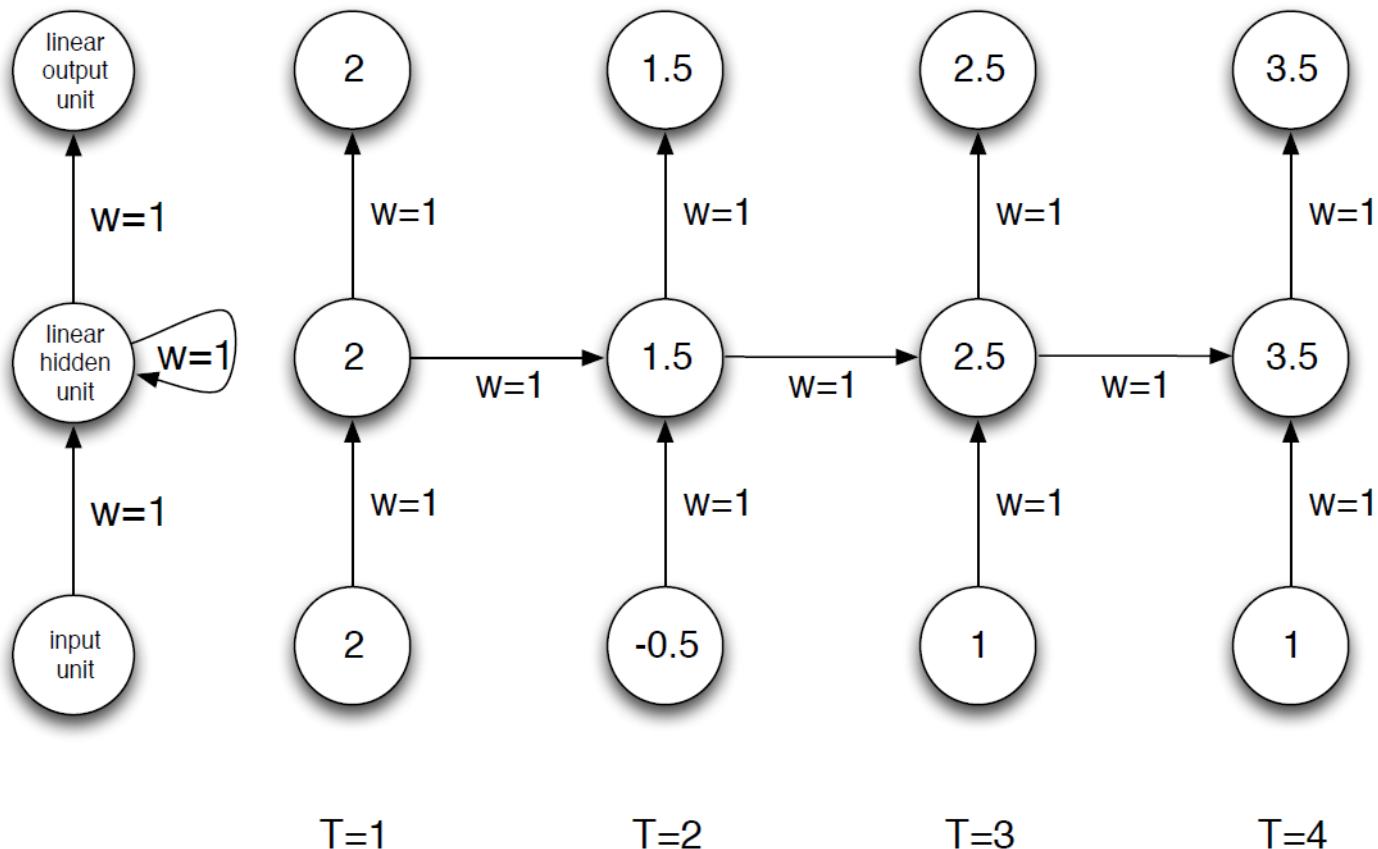
Recurrent Neural Network

- Recurrent Neural Network as a dynamical system with one set of hidden units feeding into themselves
 - The network's graph has self-loops
- The RNN's graph can be unrolled by explicitly representing the units at all time steps
 - The weights and biases are shared



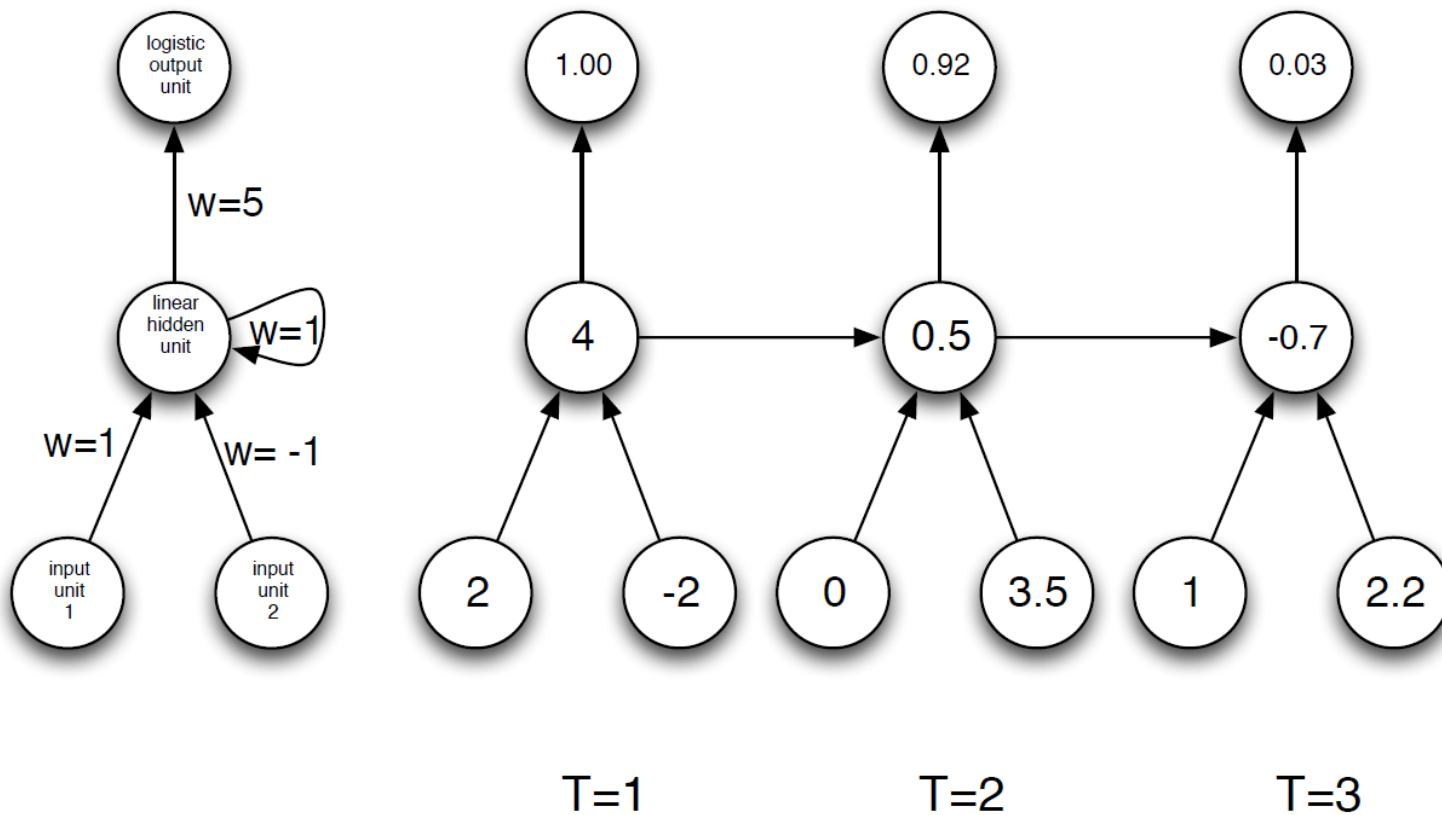
RNN examples

■ Summation network



RNN examples

■ Summation & comparison network



RNN examples

- Parity-check network
- Problem: determine the parity of a sequence of binary inputs

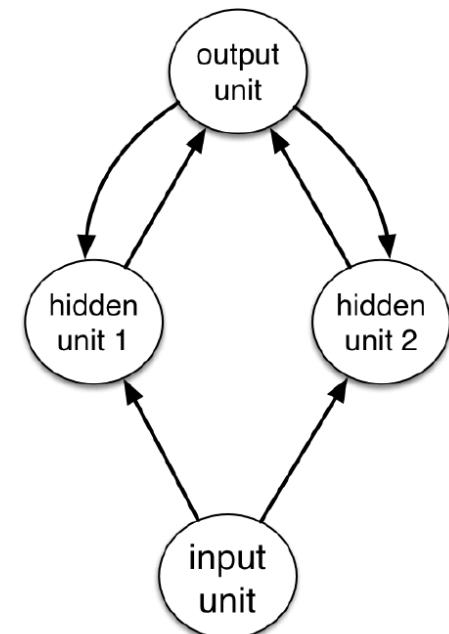
Parity bits: 0 1 1 0 1 1 →

Input: 0 1 0 1 1 0 1 0 1 1

- Each parity bit is the XOR of the input and the previous parity bit
- Hard to solve with a shallow feed-forward network

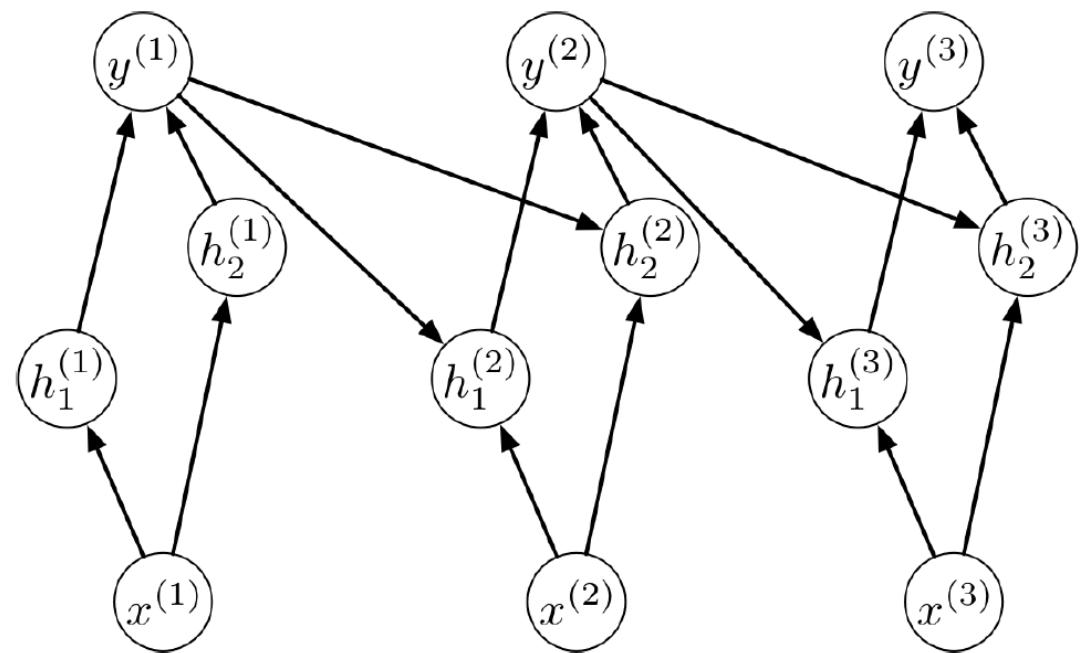
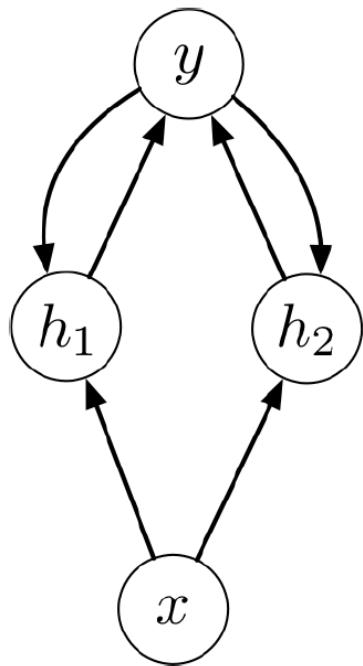
RNN examples

- Parity-check network
- Problem: determine the parity of a sequence of binary inputs
 - Each parity bit is the XOR of the input and the previous parity bit
 - Easy for RNN to solve the task
- Strategy
 - The output units tracks the current parity
 - The hidden units help compute the XOR
 - All hidden and output units are binary threshold units



RNN examples

- Parity-check network
 - Unrolling in time

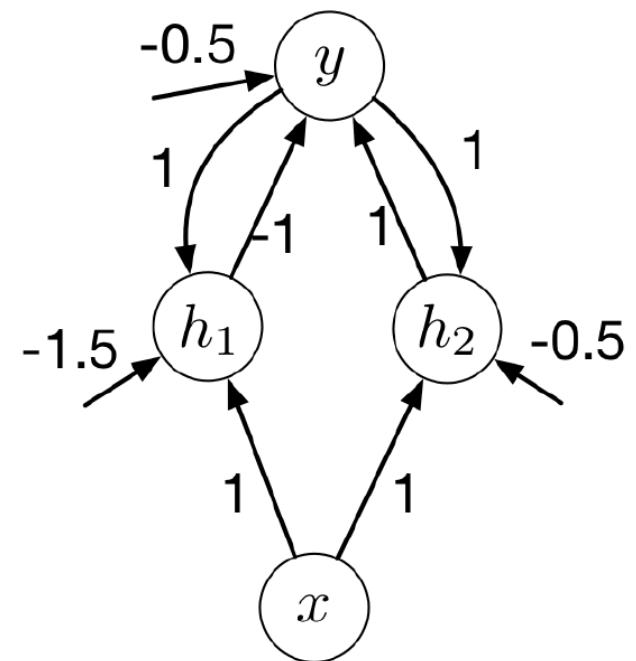


RNN examples

■ Parity-check network

- Use hidden units to compute XOR
- Pick weights and biases as in the multilayer perceptrons

$y^{(t-1)}$	$x^{(t)}$	$h_1^{(t)}$	$h_2^{(t)}$	$y^{(t)}$
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0



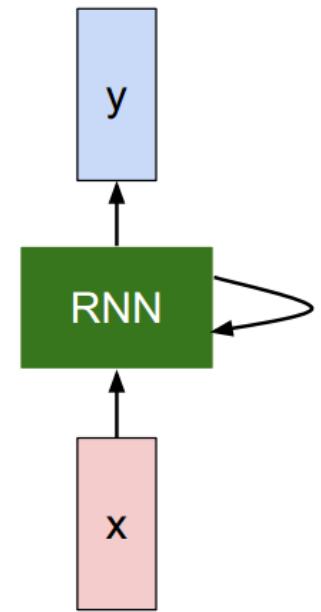
Recurrent Neural Network

■ General formulation

We can process a sequence of vectors \mathbf{x} by applying a **recurrence formula** at every time step:

$$h_t = f_W(h_{t-1}, x_t)$$

new state old state input vector at
 some function some time step
 with parameters W



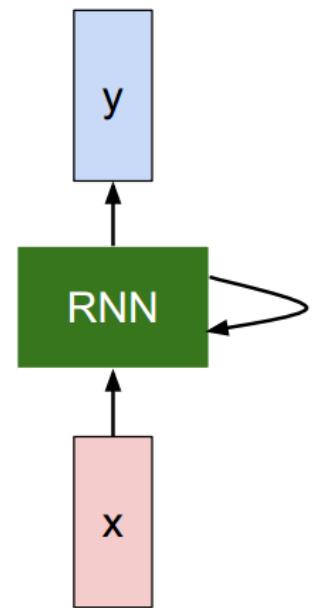
Recurrent Neural Network

■ General formulation

We can process a sequence of vectors \mathbf{x} by applying a **recurrence formula** at every time step:

$$h_t = f_W(h_{t-1}, x_t)$$

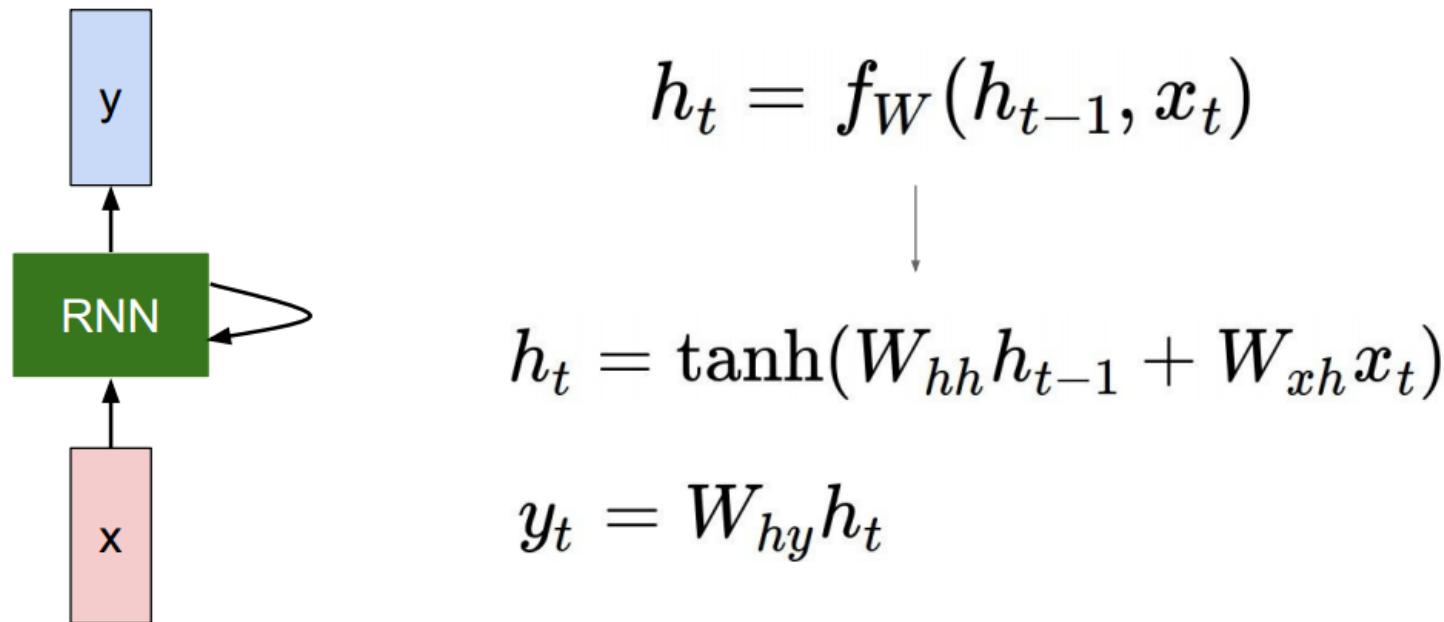
Notice: the same function and the same set of parameters are used at every time step.



(Vanilla)Recurrent Neural Network

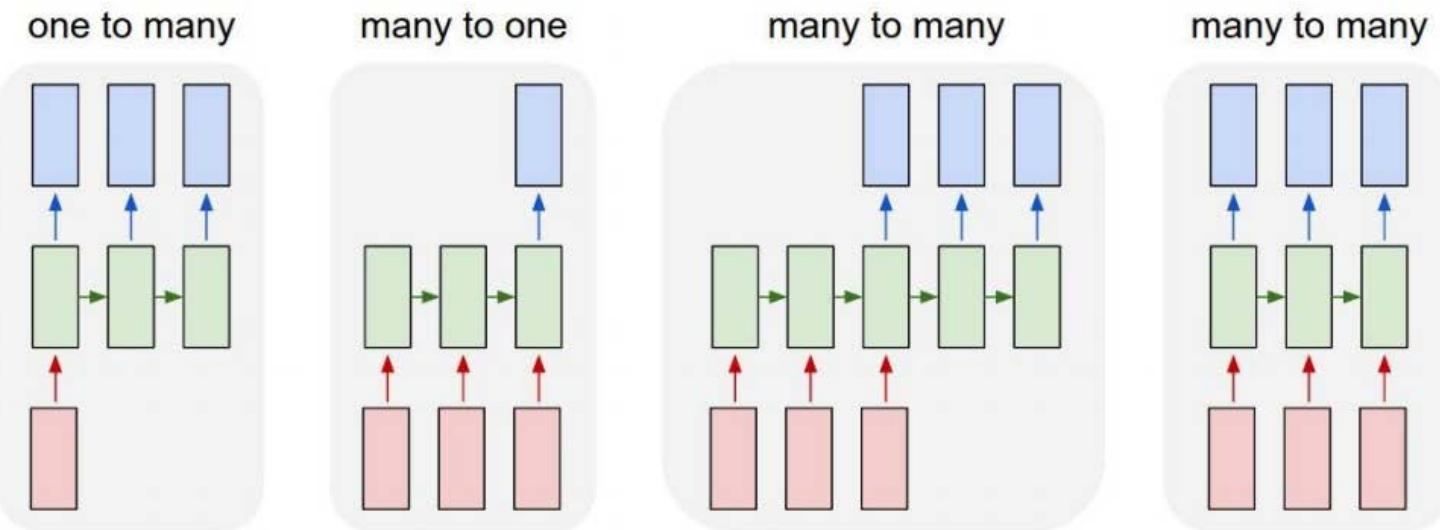
■ General formulation

The state consists of a single “*hidden*” vector \mathbf{h} :



Recurrent Neural Network

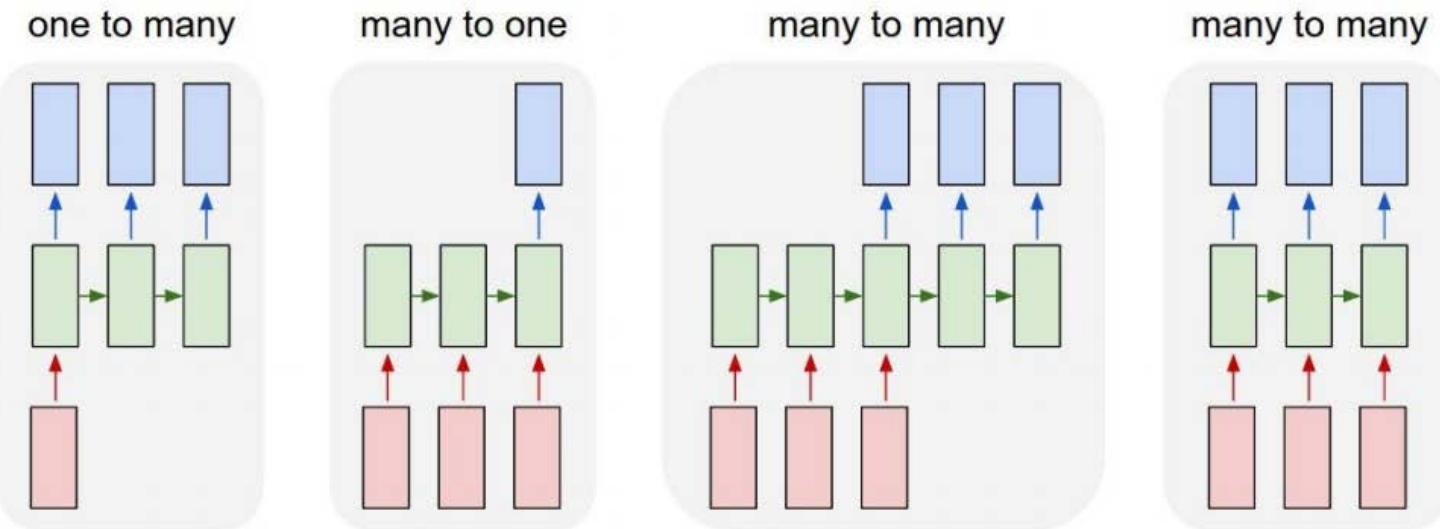
■ Recurrent Neural Networks: model variants



e.g. **Image Captioning**
image -> sequence of words

Recurrent Neural Network

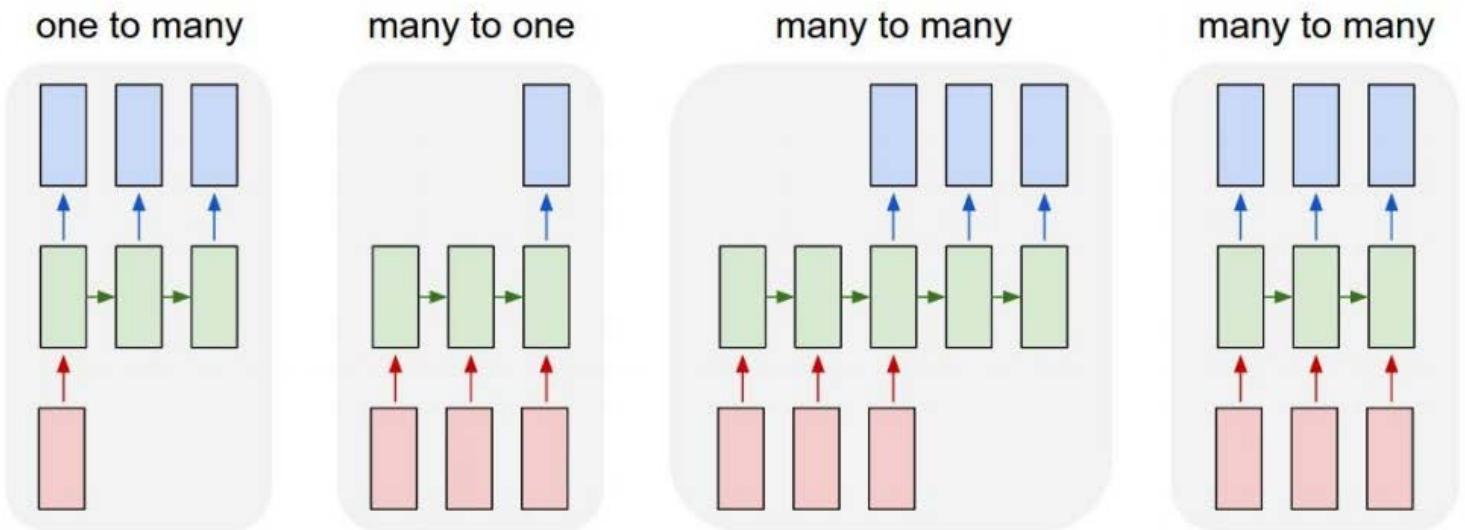
■ Recurrent Neural Networks: model variants



e.g. **Sentiment Classification**
sequence of words -> sentiment

Recurrent Neural Network

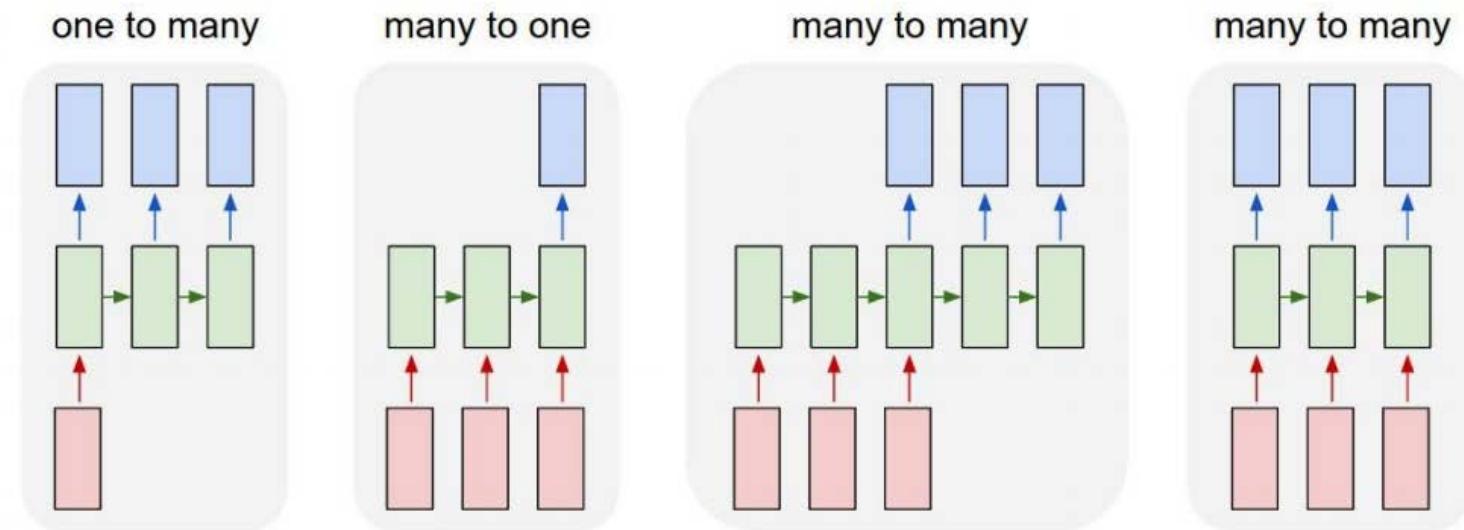
■ Recurrent Neural Networks: model variants



e.g. **Machine Translation**
seq of words -> seq of words

Recurrent Neural Network

■ Recurrent Neural Networks: model variants



e.g. **Video classification on frame level**

Recurrent Neural Network

■ Sequential Processing of Non-Sequence Data

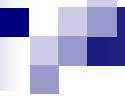
Classify images by taking a series of “glimpses”



Ba, Mnih, and Kavukcuoglu, "Multiple Object Recognition with Visual Attention", ICLR 2015.

Gregor et al, "DRAW: A Recurrent Neural Network For Image Generation", ICML 2015

Figure copyright Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra, 2015. Reproduced with permission.

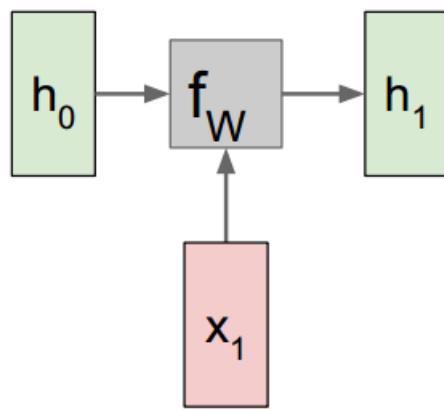


Outline

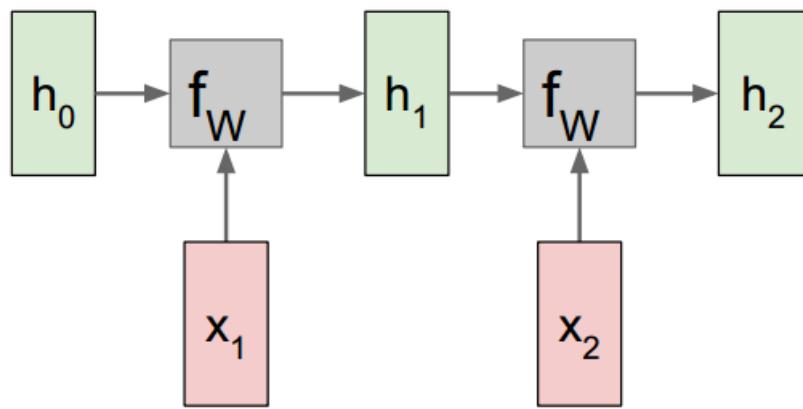
- Recurrent Neural Networks
 - Sequence modeling problem
 - Autoregressive models
 - (Vanilla) RNN models
- Backpropagation through time
 - Computational graph
- Example: language modeling
 - Neural language models

Acknowledgement: Feifei Li et al's cs231n notes

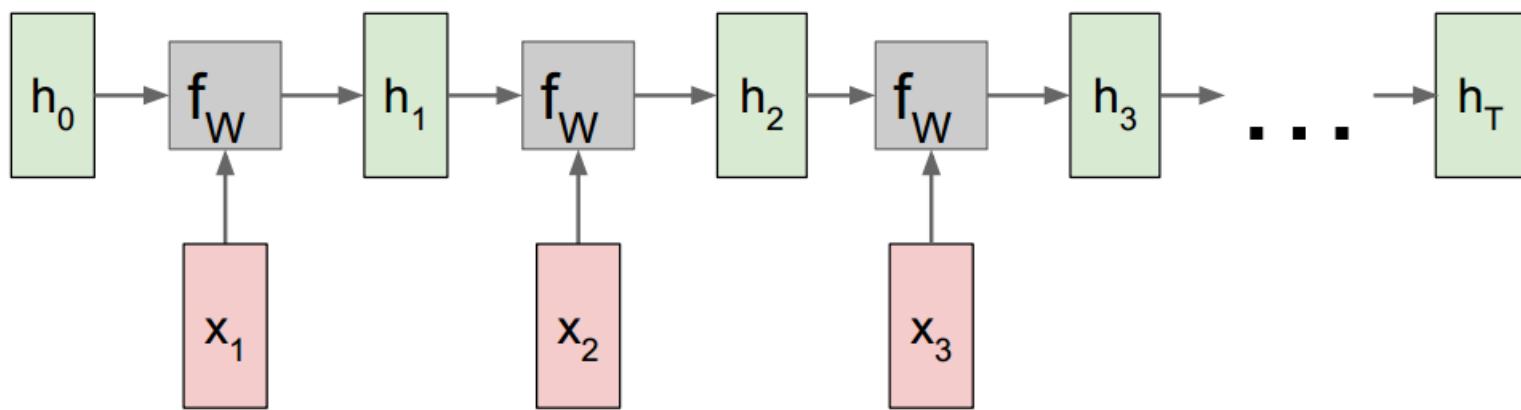
RNN: Computational Graph



RNN: Computational Graph

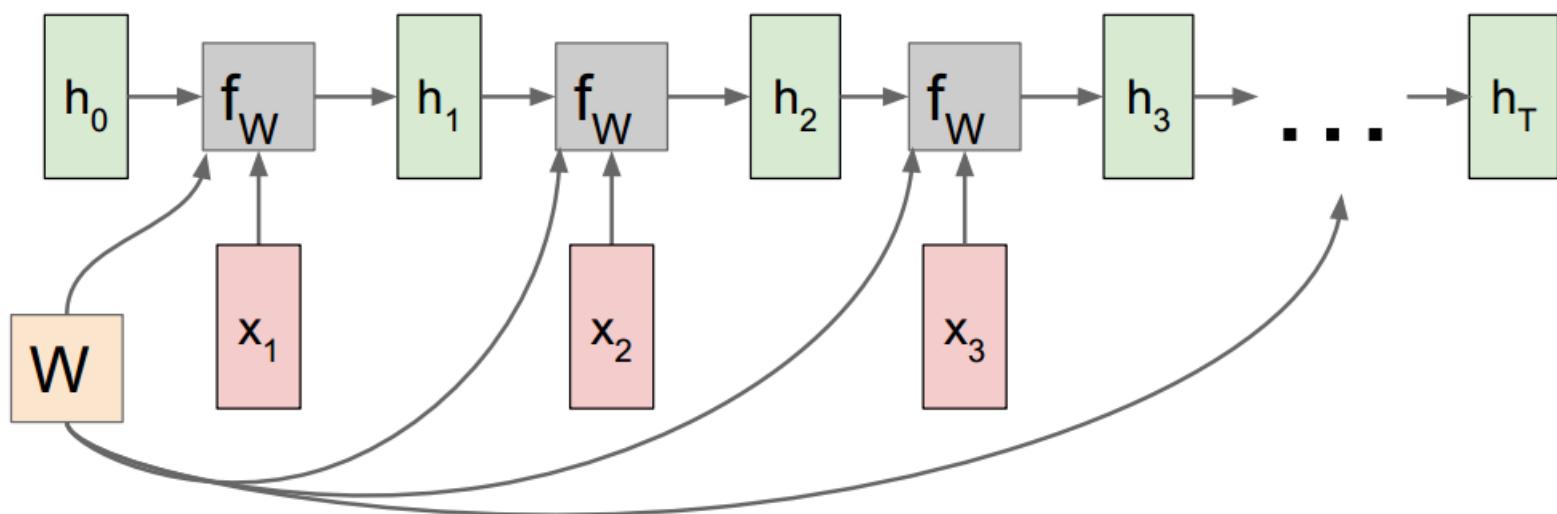


RNN: Computational Graph

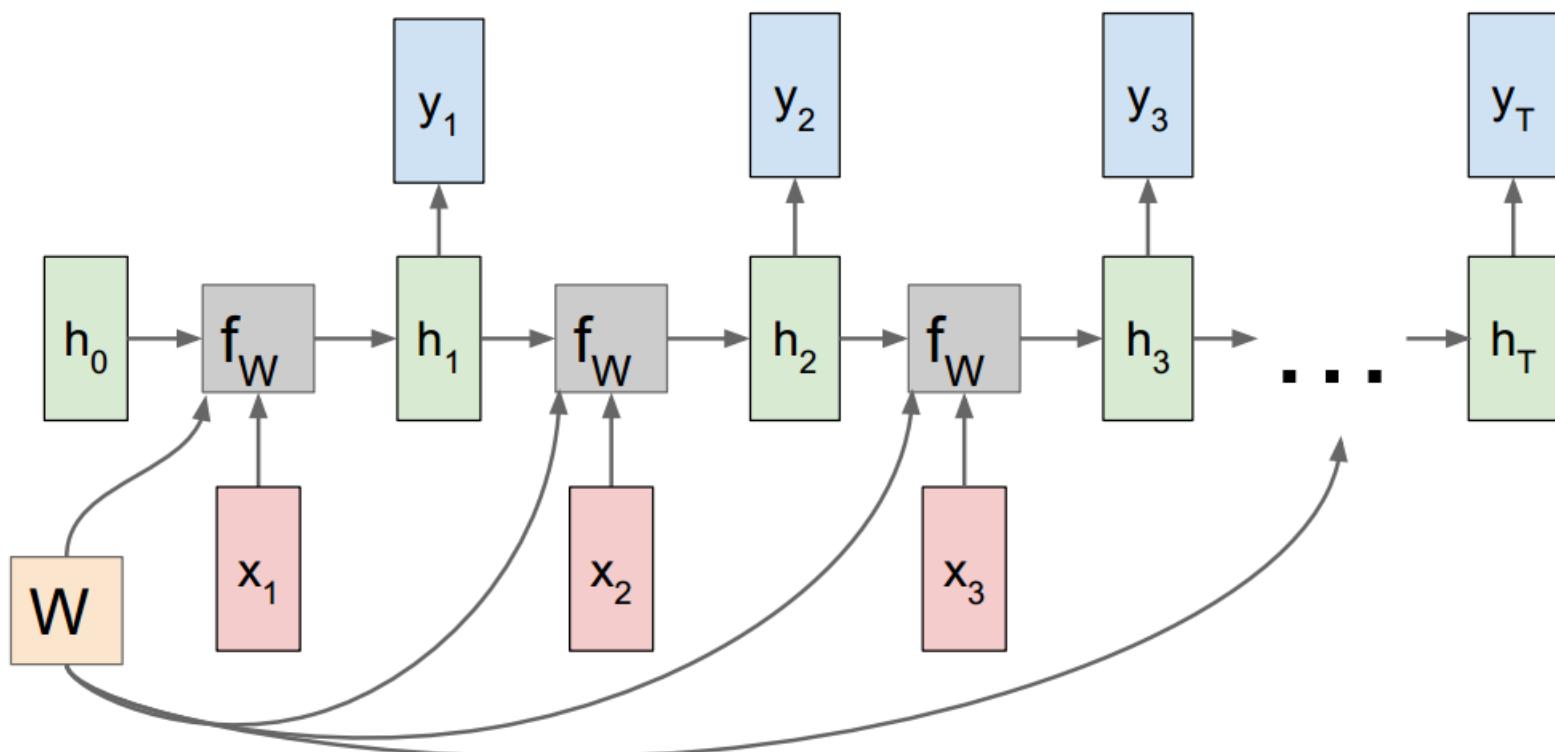


RNN: Computational Graph

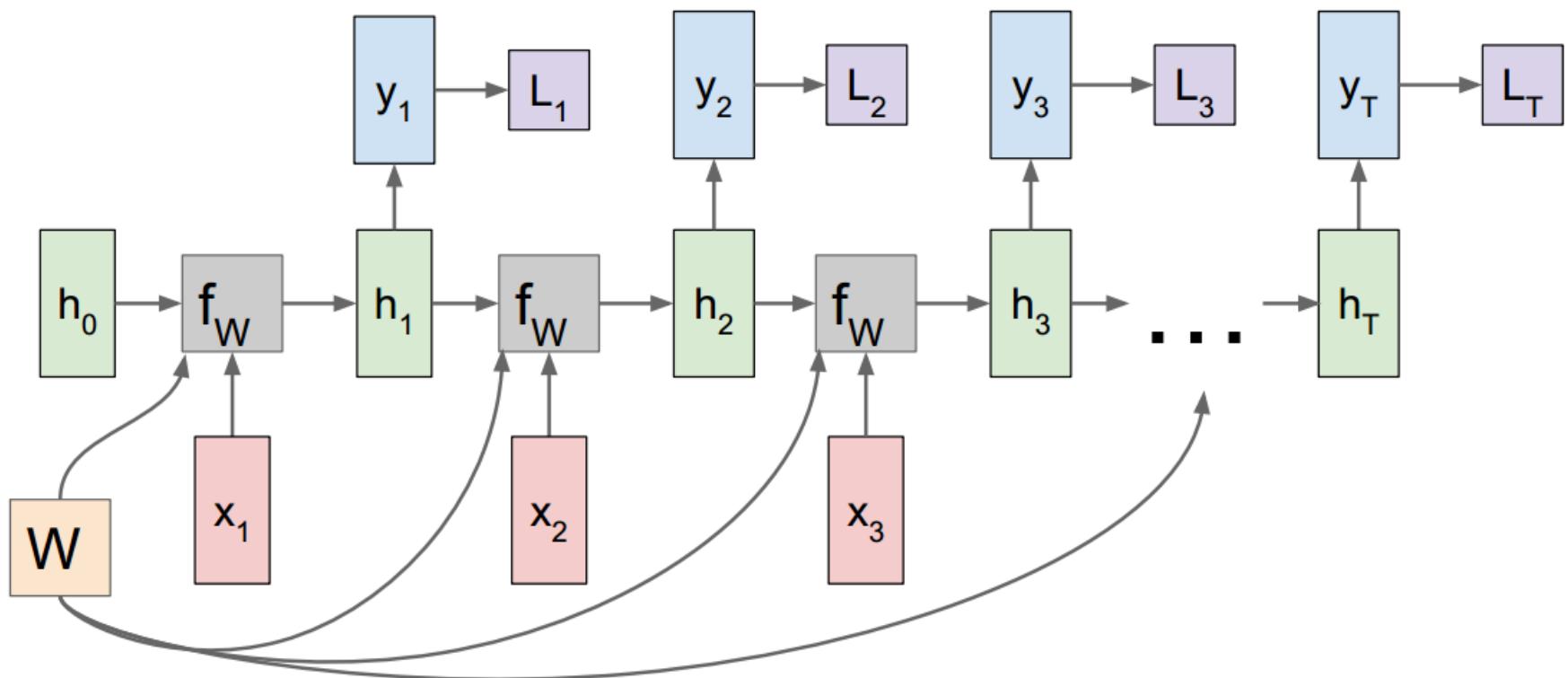
Re-use the same weight matrix at every time-step



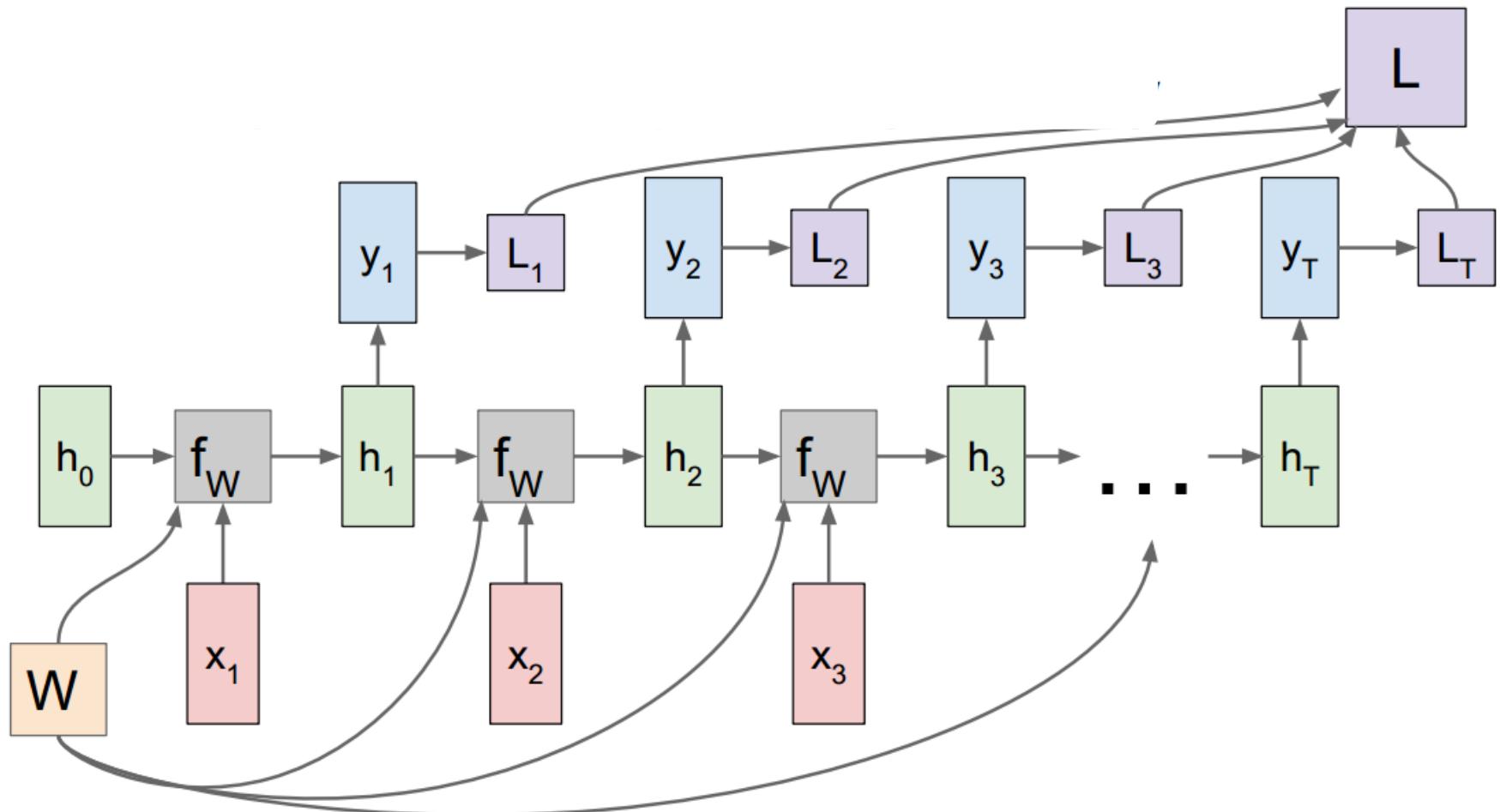
RNN: Computational Graph: Many to Many



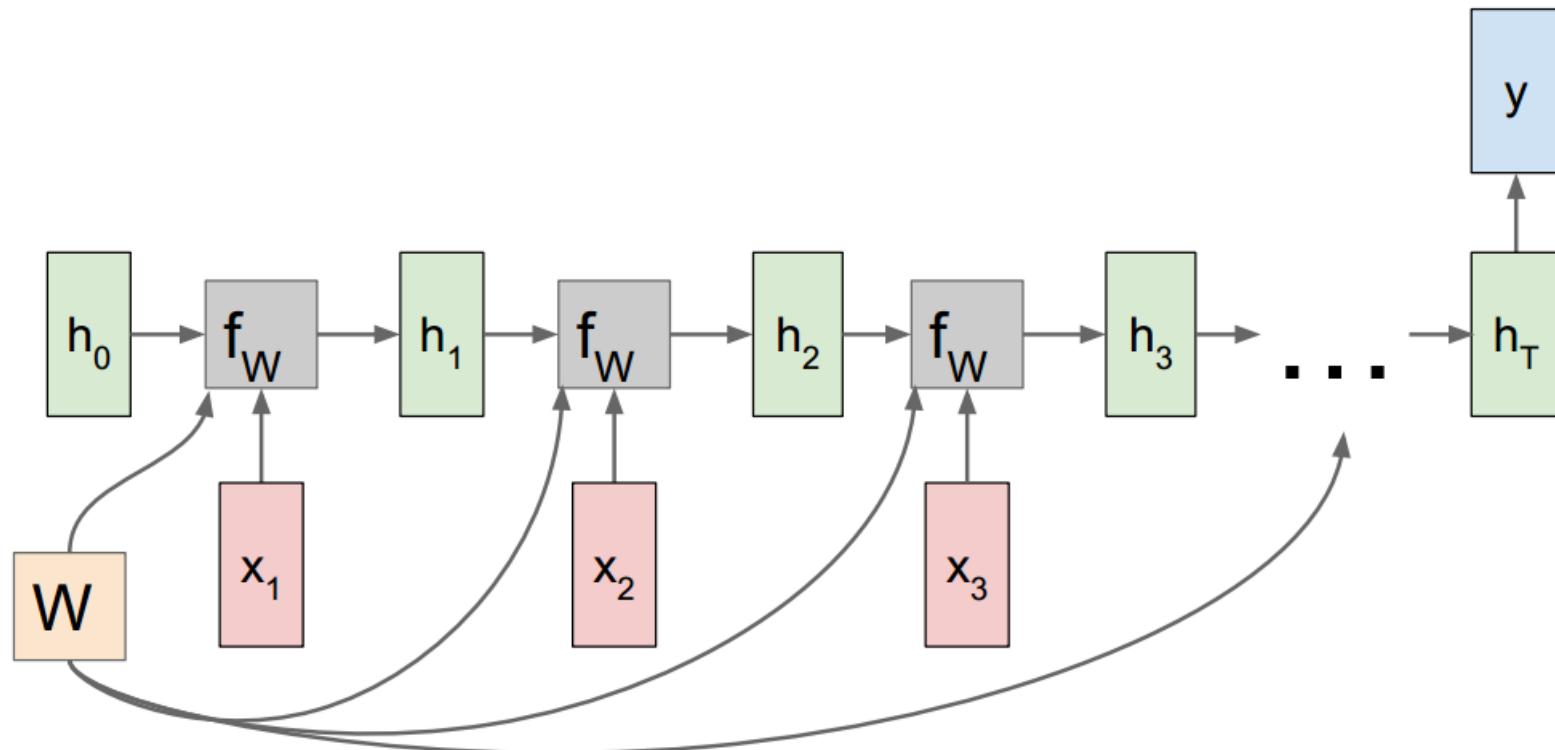
RNN: Computational Graph: Many to Many



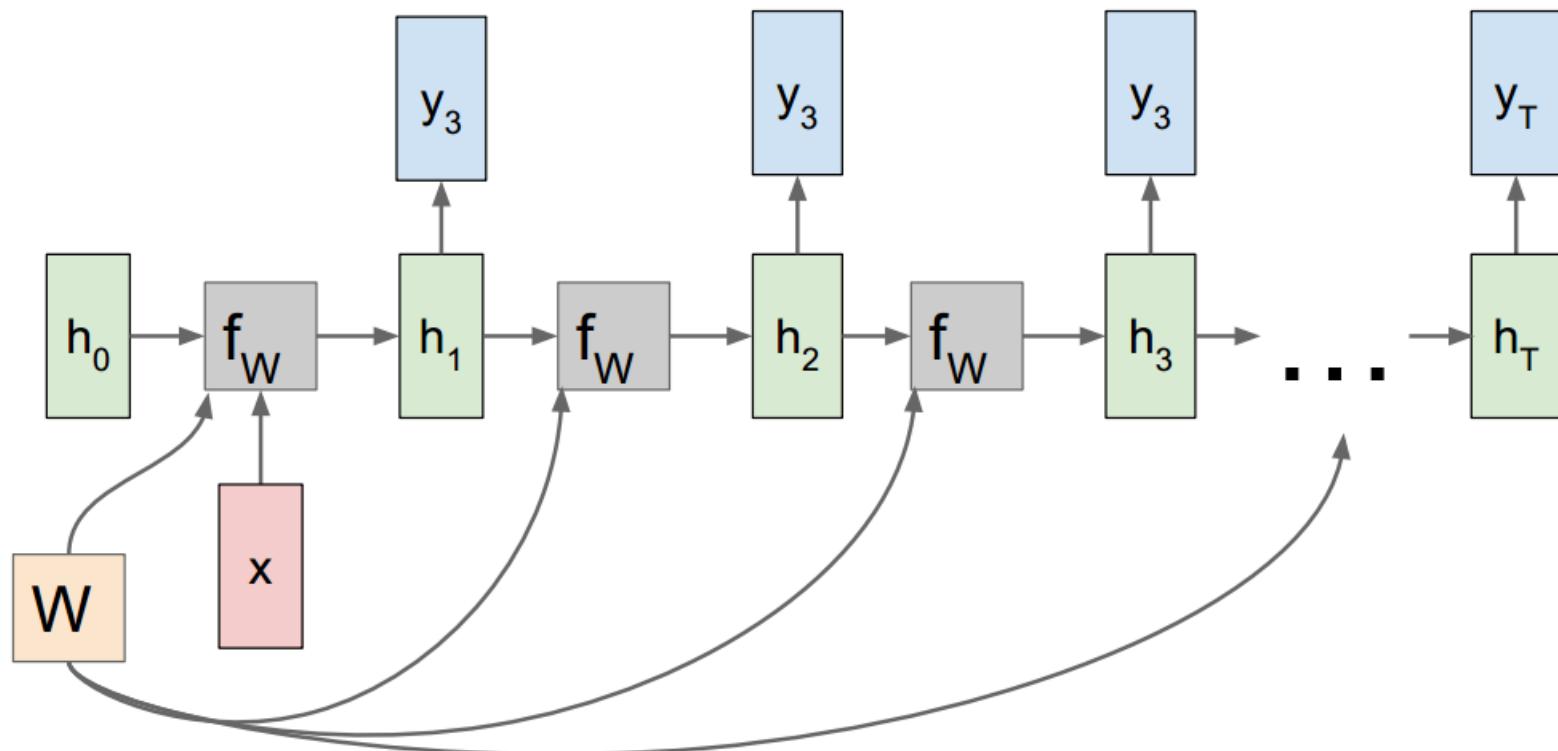
RNN: Computational Graph: Many to Many



RNN: Computational Graph: Many to One



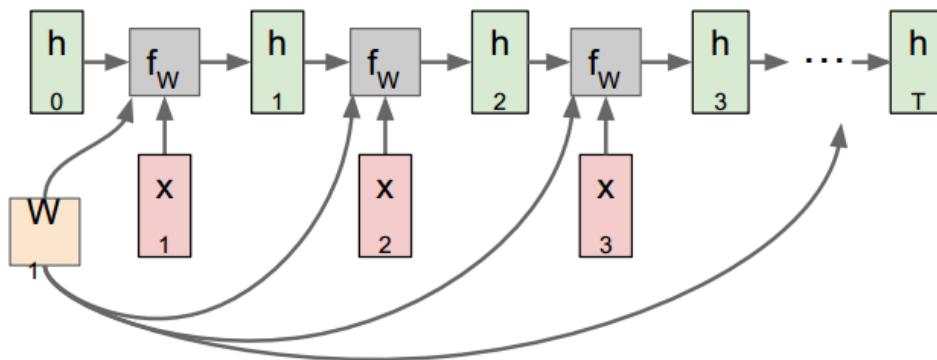
RNN: Computational Graph: One to Many



Sequence to Sequence

■ Many-to-one + one-to-many

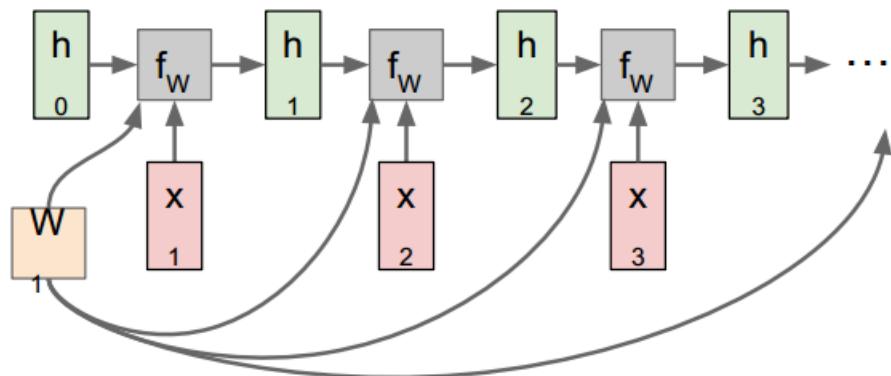
Many to one: Encode input sequence in a single vector



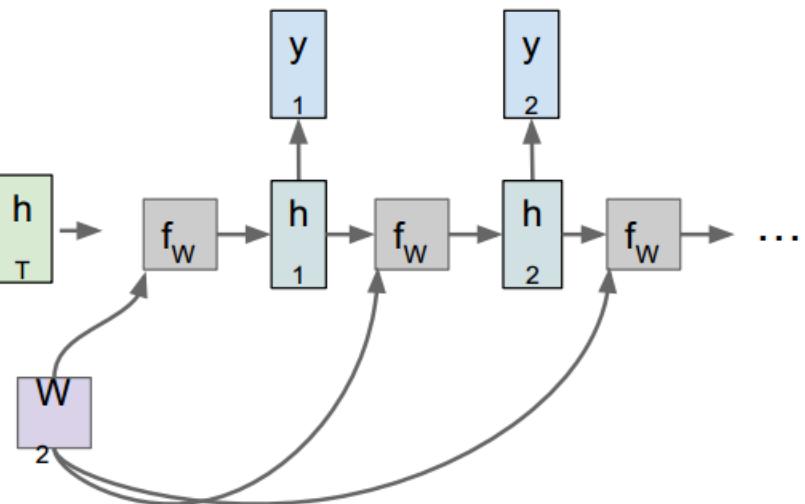
Sequence to Sequence

■ Many-to-one + one-to-many

Many to one: Encode input sequence in a single vector

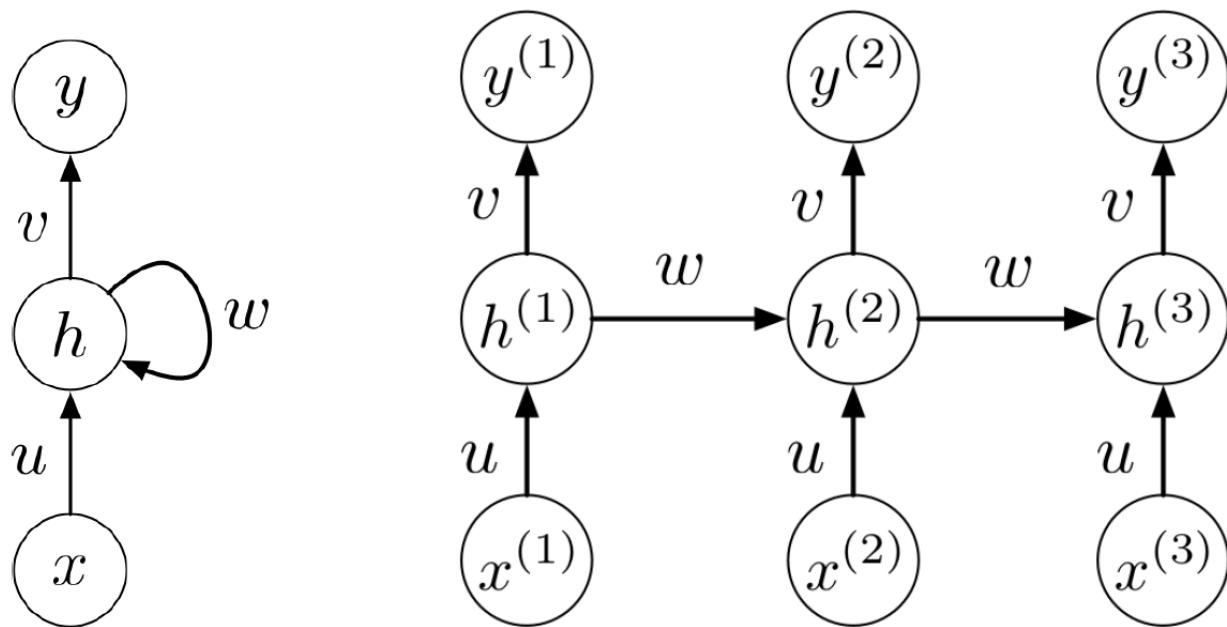


One to many: Produce output sequence from single input vector



BPTT example

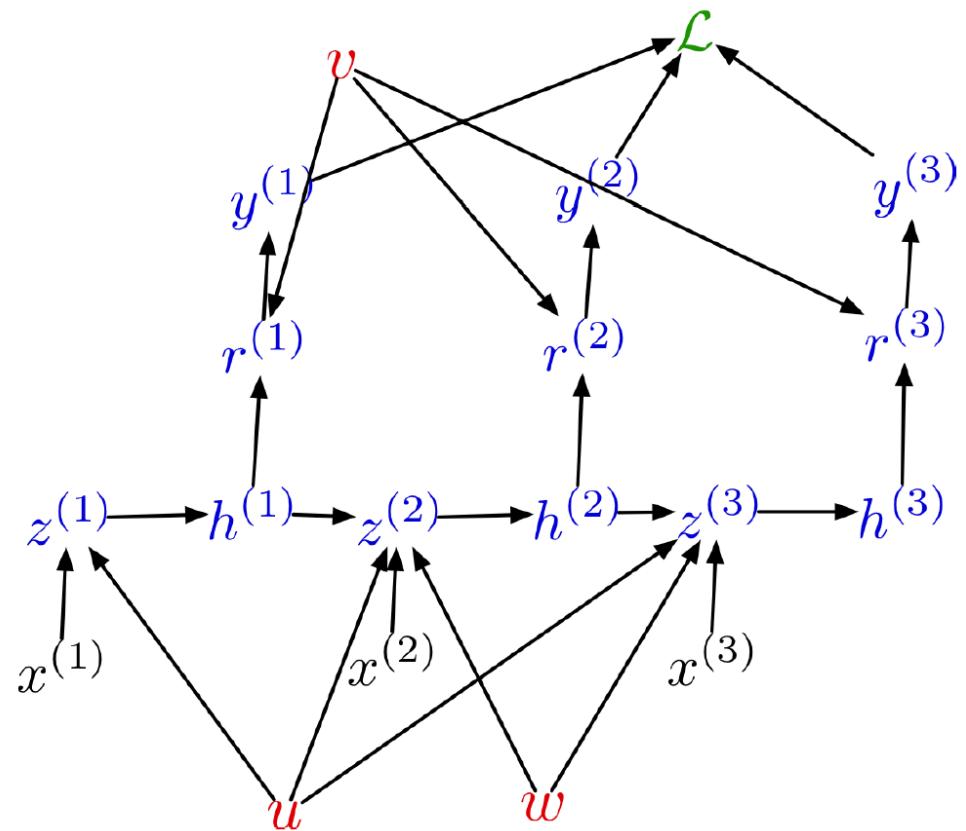
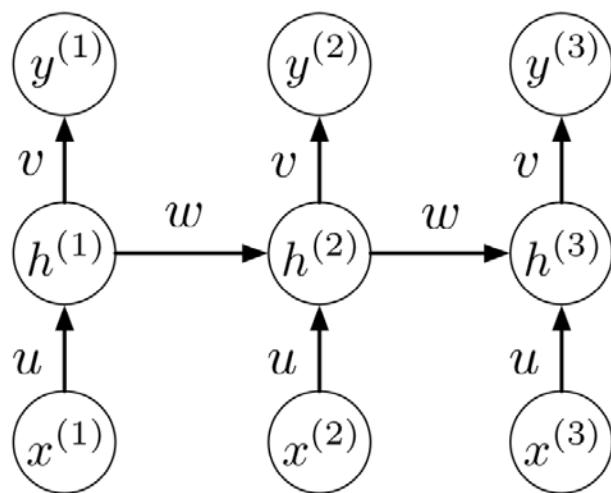
- A simple network
 - Everything is scalar



BPTT example

■ A simple network

- Everything is scalar
- Unrolled computation graph with shared parameters

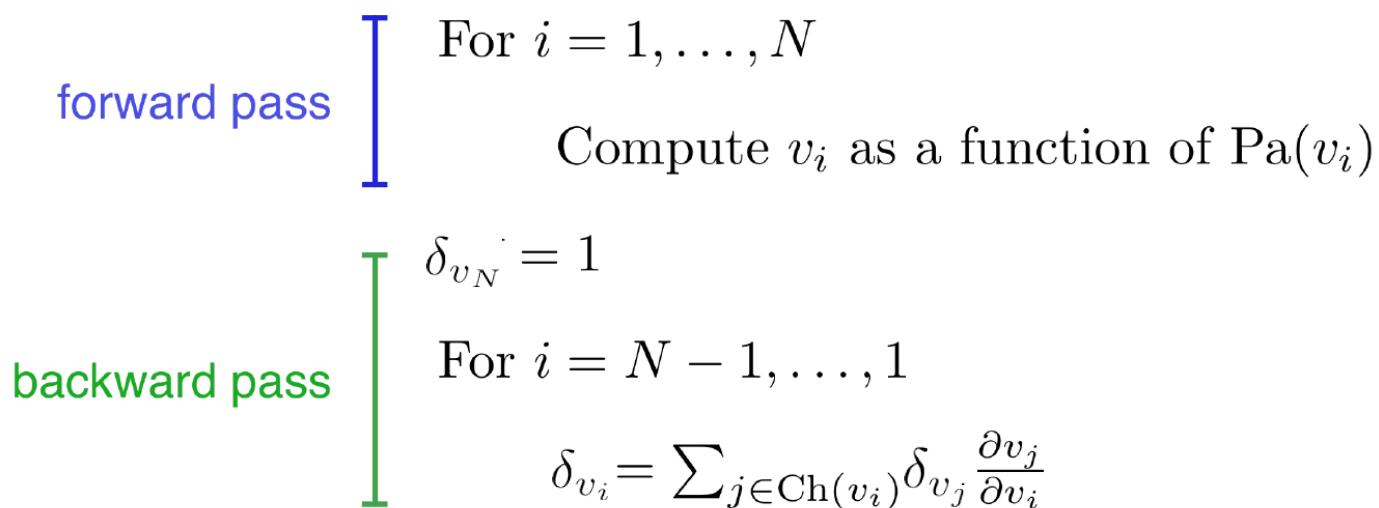


Recall: General Backpropagation

■ Given a computation graph

Let v_1, \dots, v_N be a **topological ordering** of the computation graph
(i.e. parents come before children.)

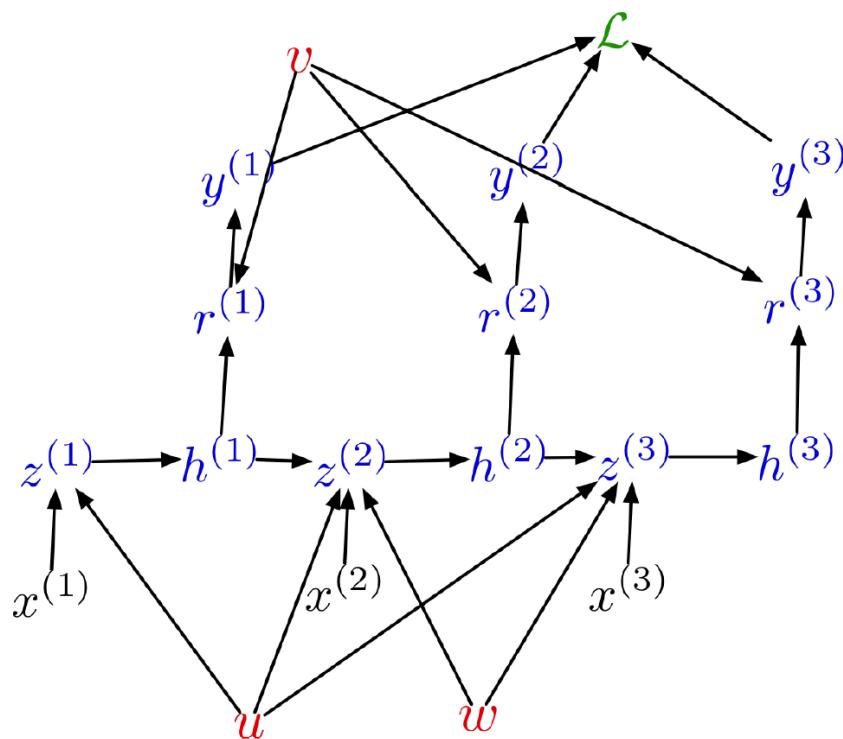
v_N denotes the variable we're trying to compute derivatives of (e.g. loss)



BPTT example

■ A simple network

- Unrolled computation graph with shared parameters



Activations:

$$\bar{\mathcal{L}} = 1$$

$$\overline{y^{(t)}} = \bar{\mathcal{L}} \frac{\partial \mathcal{L}}{\partial y^{(t)}}$$

$$\overline{r^{(t)}} = \overline{y^{(t)}} \phi'(r^{(t)})$$

$$\overline{h^{(t)}} = \overline{r^{(t)}} v + \overline{z^{(t+1)}} w$$

$$\overline{z^{(t)}} = \overline{h^{(t)}} \phi'(z^{(t)})$$

Parameters:

$$\bar{u} = \sum_t \overline{z^{(t)}} x^{(t)}$$

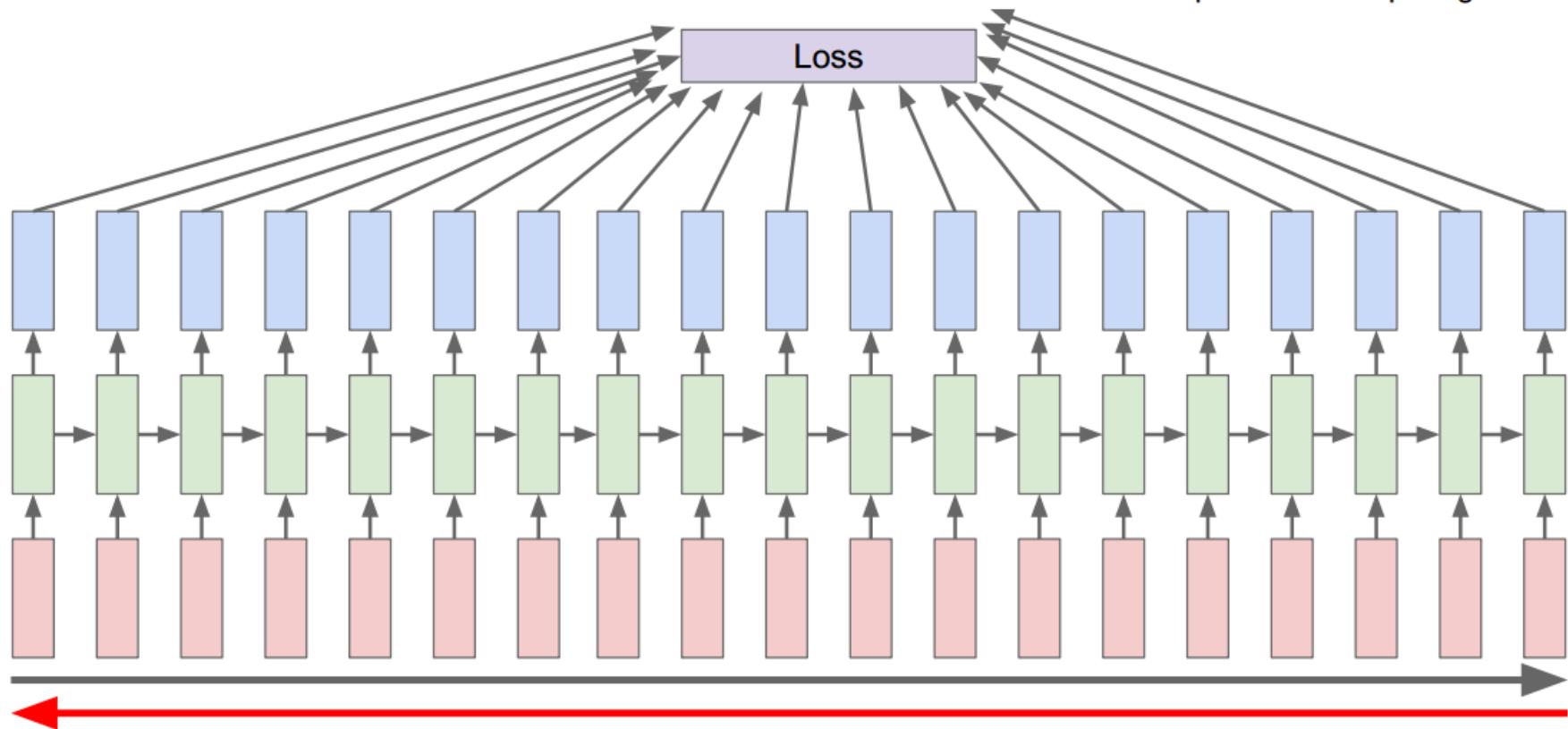
$$\bar{v} = \sum_t \overline{r^{(t)}} h^{(t)}$$

$$\bar{w} = \sum_t \overline{z^{(t+1)}} h^{(t)}$$

Recurrent Neural Network

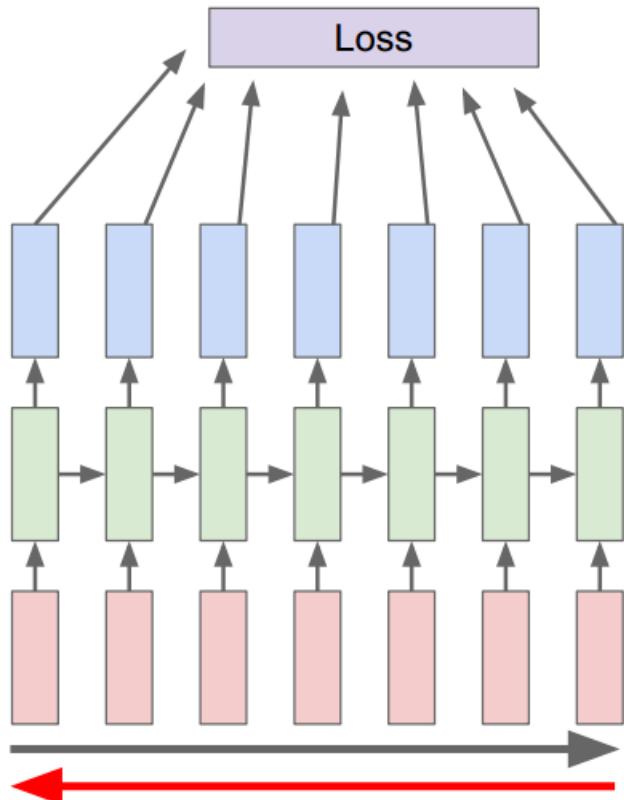
Backpropagation through time

Forward through entire sequence to compute loss, then backward through entire sequence to compute gradient



Recurrent Neural Network

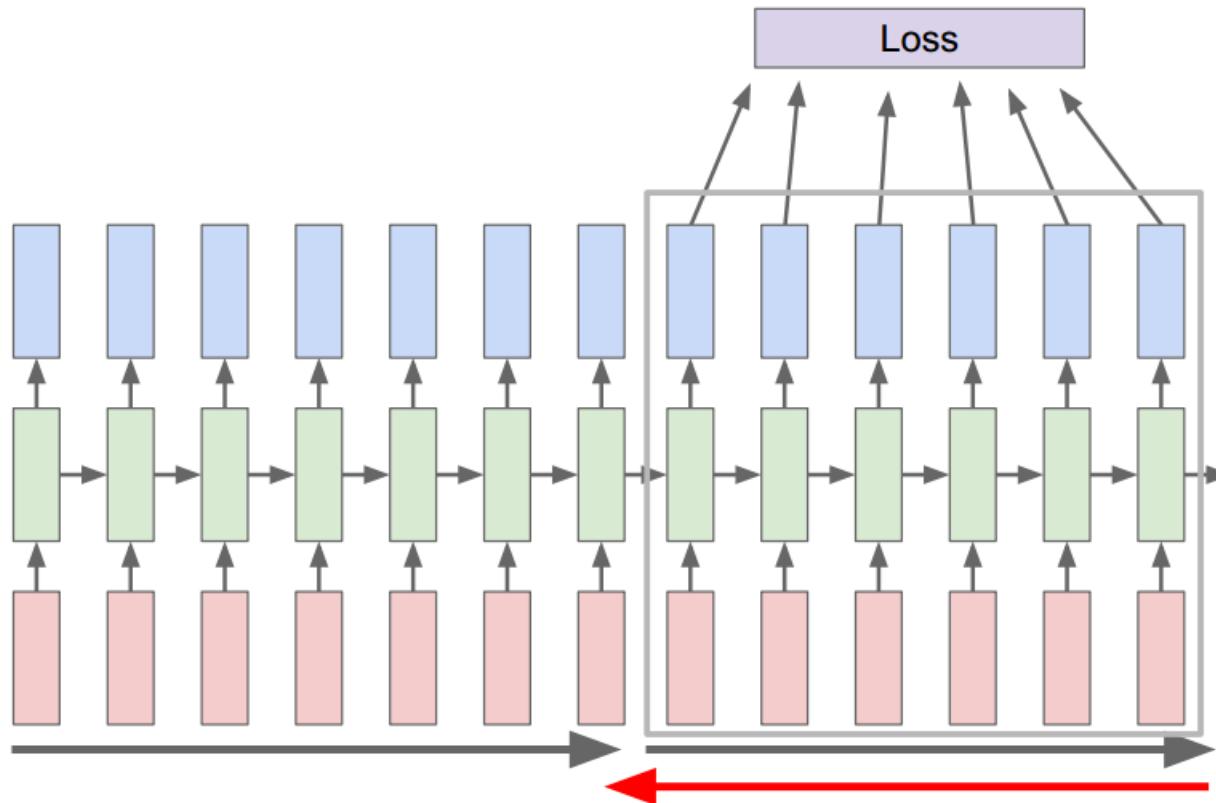
Truncated Backpropagation through time



Run forward and backward
through chunks of the
sequence instead of whole
sequence

Recurrent Neural Network

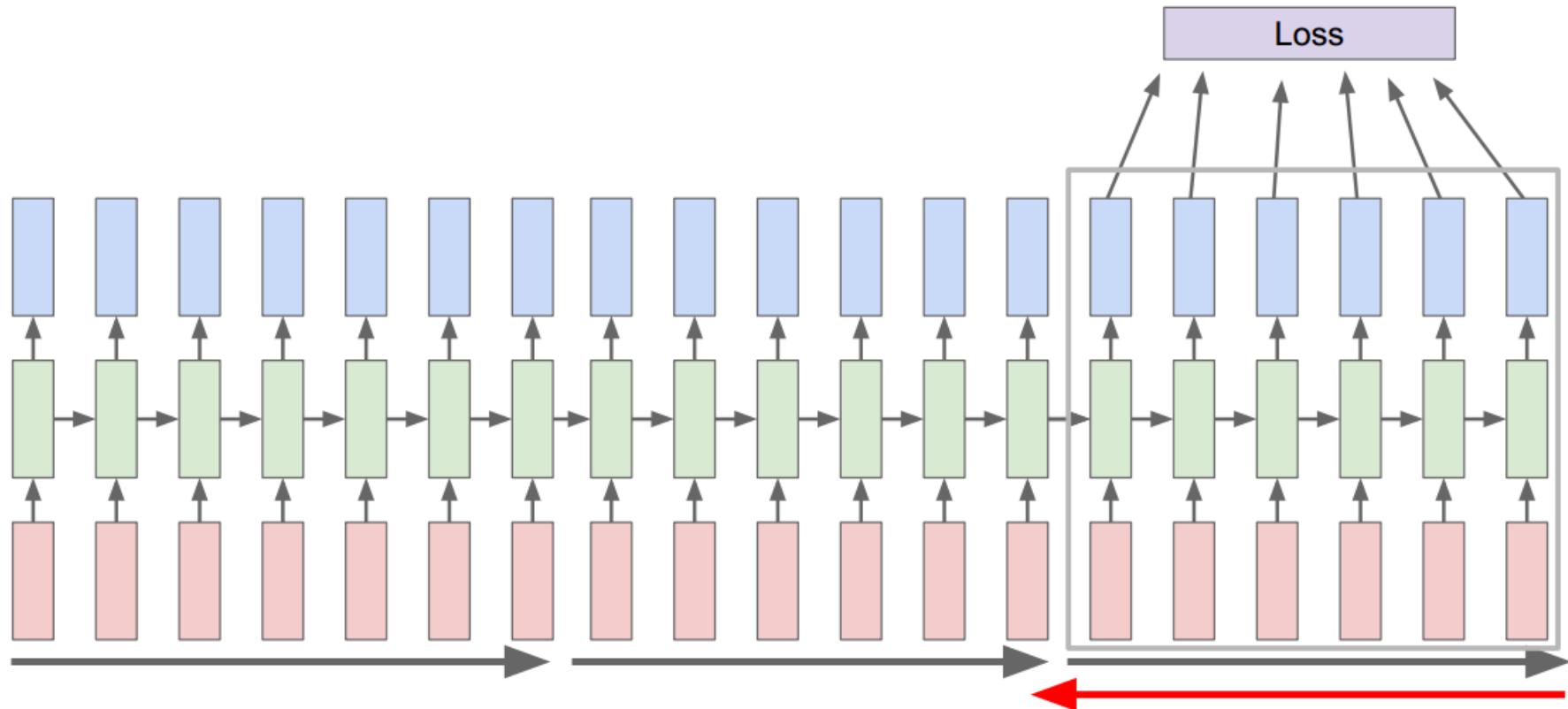
Truncated Backpropagation through time

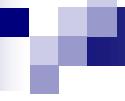


Carry hidden states forward in time forever, but only backpropagate for some smaller number of steps

Recurrent Neural Network

Truncated Backpropagation through time





Outline

- Recurrent Neural Networks
 - Sequence modeling problem
 - Autoregressive models
 - (Vanilla) RNN models
- Backpropagation through time
 - Computational graph
- Example: language modeling
 - Neural language models

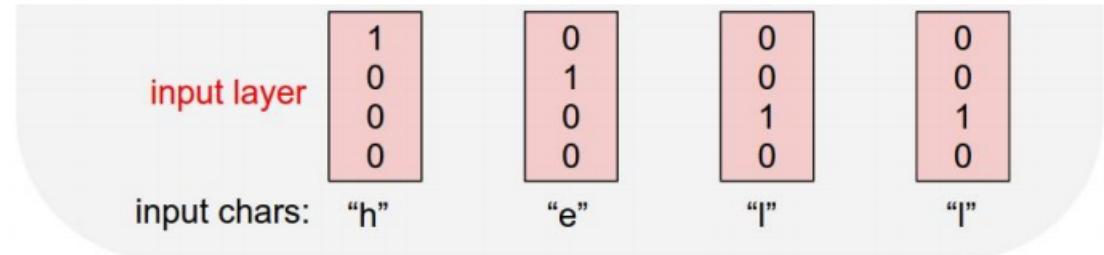
Acknowledgement: Feifei Li et al's cs231n notes

RNN for language modeling

**Example:
Character-level
Language Model**

Vocabulary:
[h,e,l,o]

Example training
sequence:
“hello”



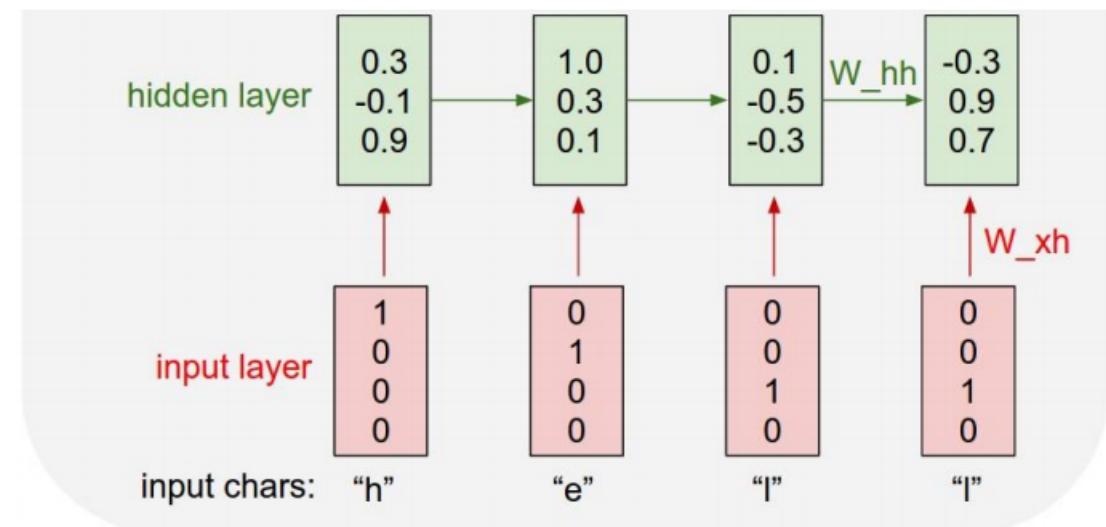
RNN for language modeling

**Example:
Character-level
Language Model**

Vocabulary:
[h,e,l,o]

Example training
sequence:
“hello”

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

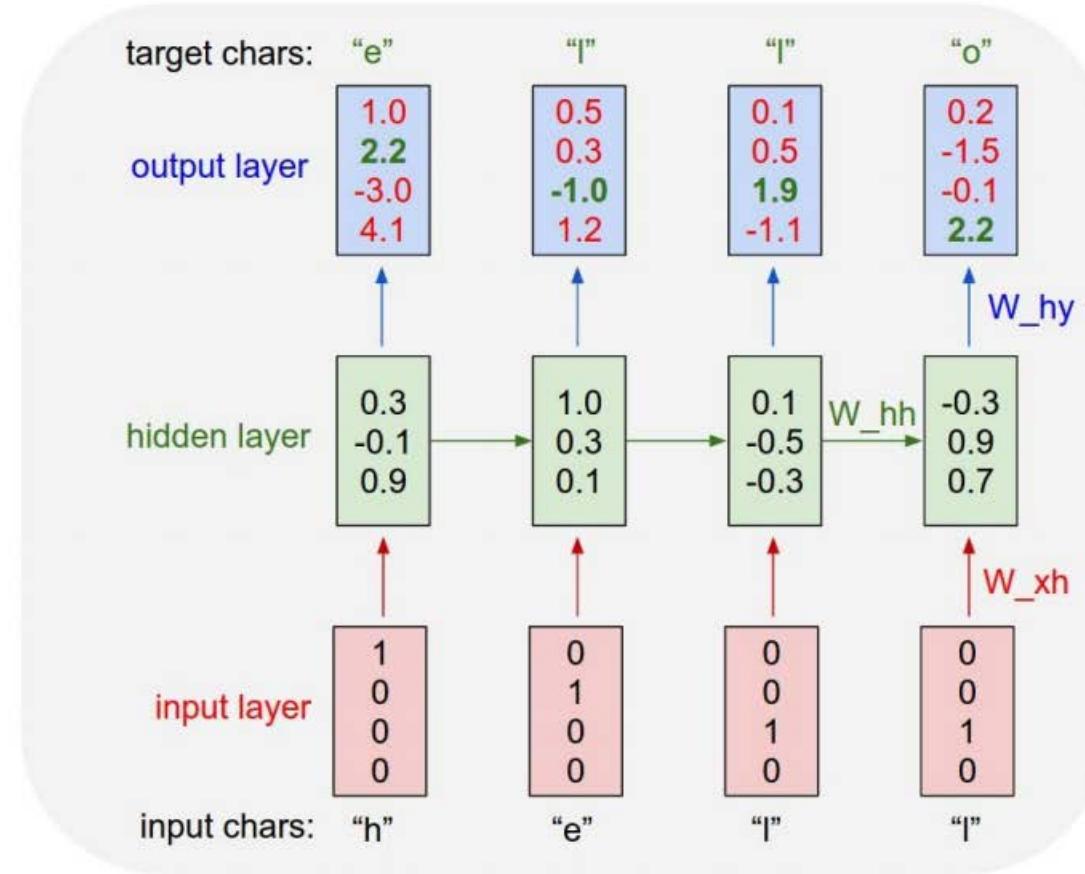


RNN for language modeling

**Example:
Character-level
Language Model**

Vocabulary:
[h,e,l,o]

Example training
sequence:
“hello”

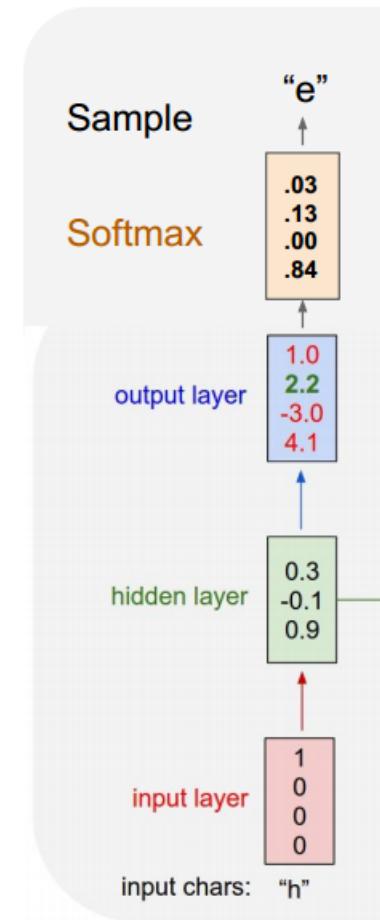


RNN for language modeling

**Example:
Character-level
Language Model
Sampling**

Vocabulary:
[h,e,l,o]

At test-time sample
characters one at a time,
feed back to model

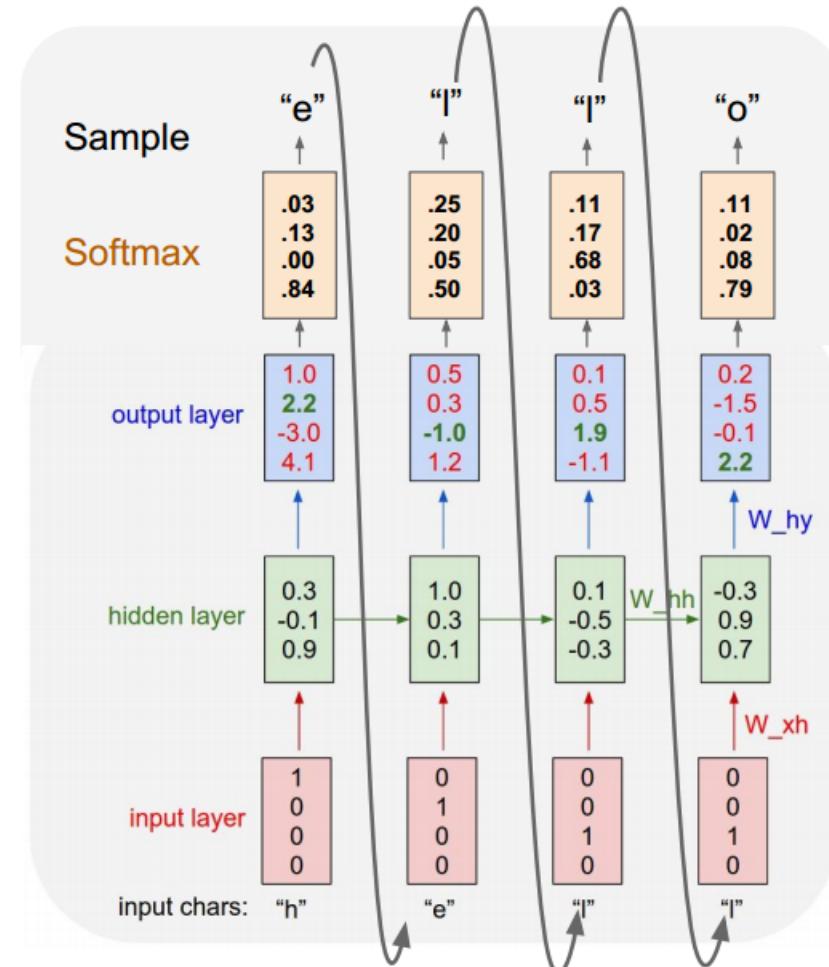


RNN for language modeling

**Example:
Character-level
Language Model
Sampling**

Vocabulary:
[h,e,l,o]

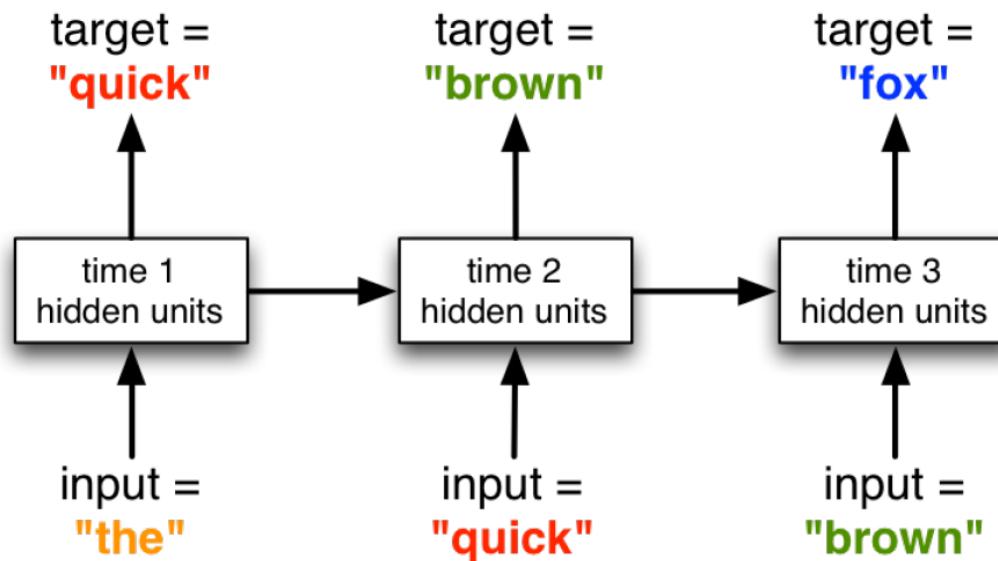
At test-time sample
characters one at a time,
feed back to model



RNN for language modeling

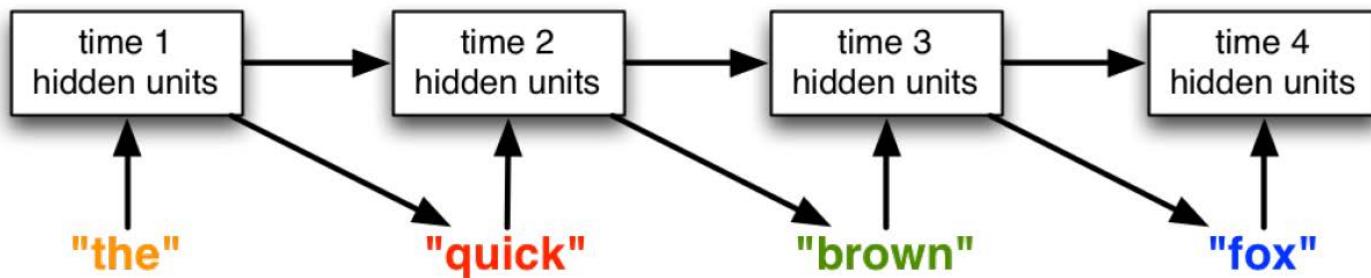
■ Modeling at word level

- Each word is represented as an indicator vector
- The model predicts a distribution over words



RNN for language modeling

- Generating from a RNN language model
 - The outputs are fed back to the network



- Training time: the inputs are the token from the training set (**teacher forcing**).

RNN for language modeling

at first:

tyntd-iafhatawiaoahrdemot lytdws e ,tfti, astai f ogoh eoase rrranbyne 'nhthnee e
plia tkldrgd t o idoe ns,smtt h ne etie h,hregtrs nigtike,aoaenns lng

↓ train more

"Tmont thithey" fomesscerliund
Keushey. Thom here
sheulke, anmerenith ol sivh I lalterthend Bleipile shuwyl fil on aseterlome
coaniogennc Phe lism thond hon at. MeiDimorotion in ther thize."

↓ train more

Aftair fall unsuch that the hall for Prince Velzonski's that me of
her hearly, and behs to so arwage fiving were to it beloge, pavu say falling misfort
how, and Gogition is so overelical and ofter.

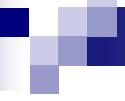
↓ train more

"Why do what that day," replied Natasha, and wishing to himself the fact the
princess, Princess Mary was easier, fed in had oftened him.
Pierre aking his soul came to the packs and drove up his father-in-law women.

RNN for language modeling

■ Some remaining challenges

- Vocabularies can be very large once you include people, places, etc. It's computationally difficult to predict distributions over millions of words.
- How do we deal with words we haven't seen before?
- In some languages (e.g. Chinese), it's hard to determine what should be considered a word.



Summary

■ RNN

- RNN for sequence modeling
- RNNs allow a lot of flexibility in architecture design
- Language modeling in NLP

■ Next time:

- LSTM, GRU