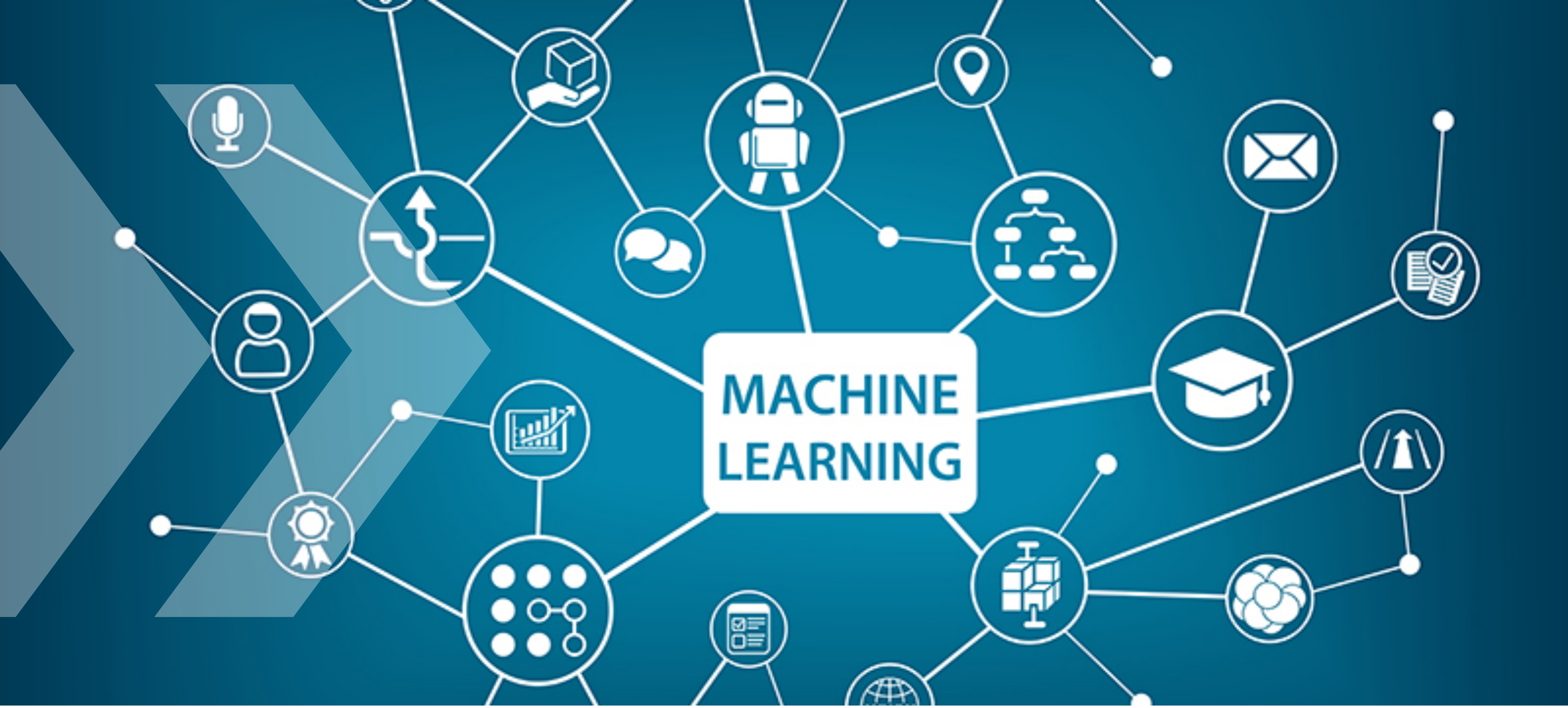


CS480/680: Intro to ML

Lecture 15: Recurrent Neural Networks (RNNs)

UNIVERSITY OF WATERLOO

some slides are adapted from Stanford cs231n course slides

Recap: MLPs and CNNs

MLPs:

- Fixed length of input and output
- No parameter sharing
- Units fully connected to the previous layer

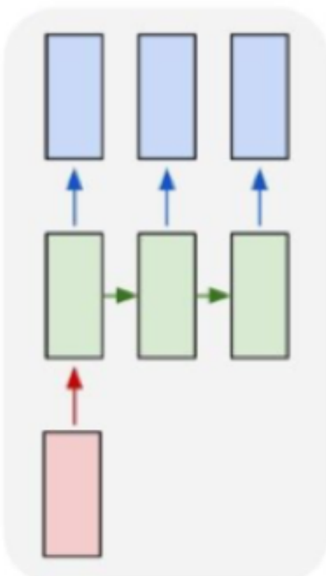
CNNs

- Fixed length of ~~input~~ and output
- Parameter sharing
- Units only connected to a small region of the previous layer

RNN: examples (1)

Variable length of input/output

one to many



e.g., image captioning

Two people walking on the beach with surfboards

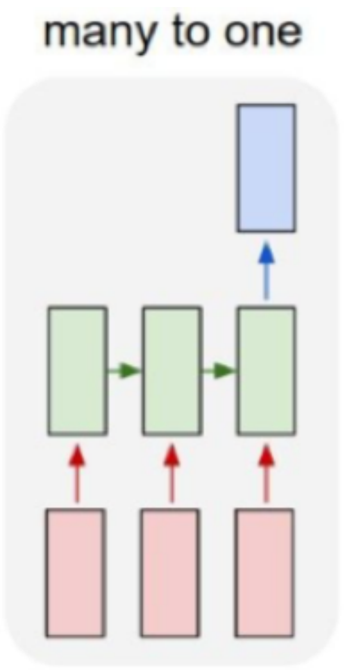
A tennis player in action on the court

Two giraffes standing in a grassy field

A man riding a dirt bike on a dirt track

RNN: examples (2)

Variable length of input/output

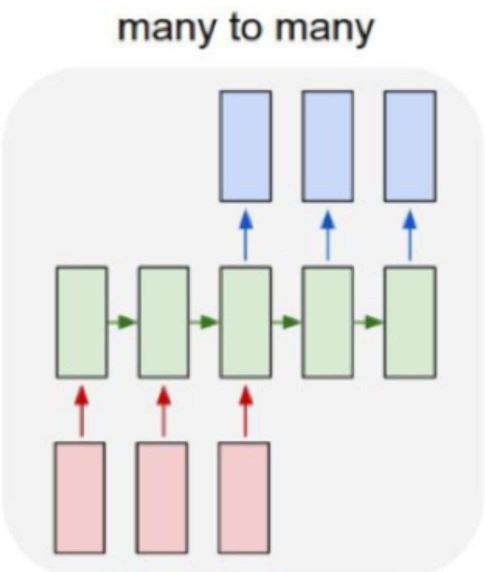


- ``There is nothing to like in this movie.'' → 0
- ``This movie is fantastic!'' → 1

e.g., sentiment classification

RNN: examples (3)

Variable length of input/output

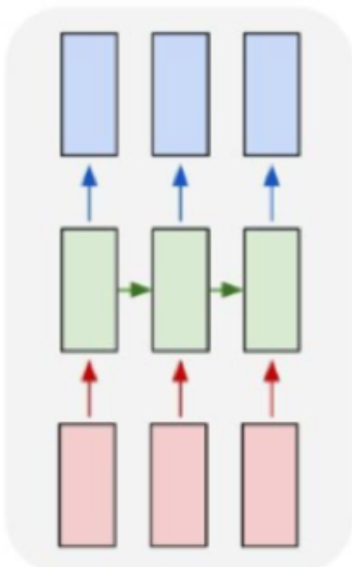


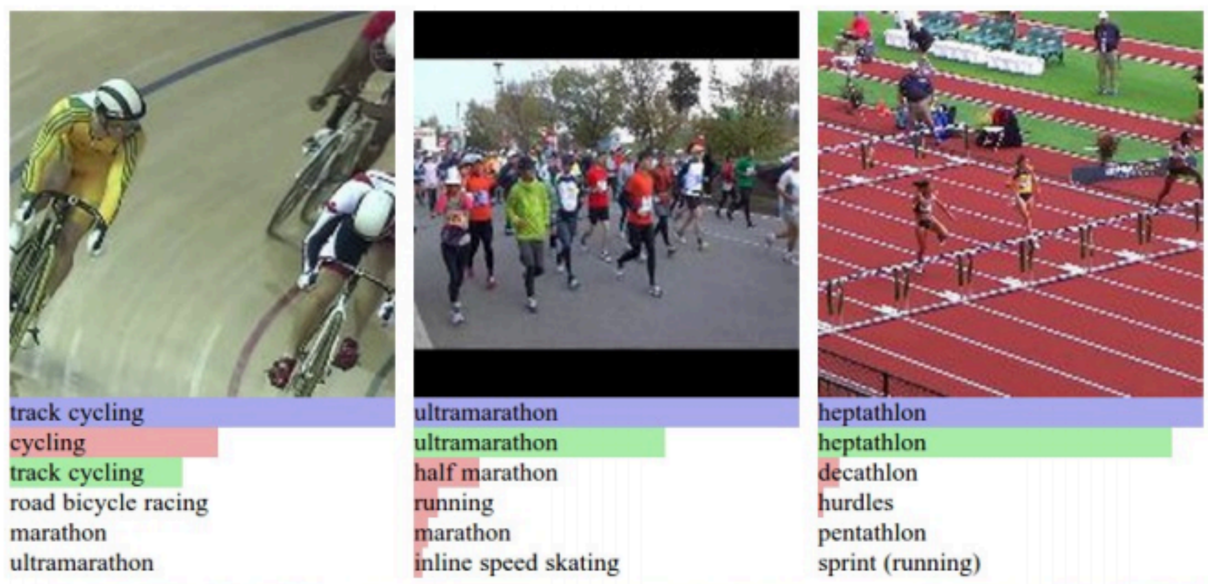
e.g., machine translation

你想和我一起唱歌吗? → Do you want to sing with me?

RNN: examples (4)

Variable length of input/output





e.g., video classification on frame level

RNN: parameter sharing

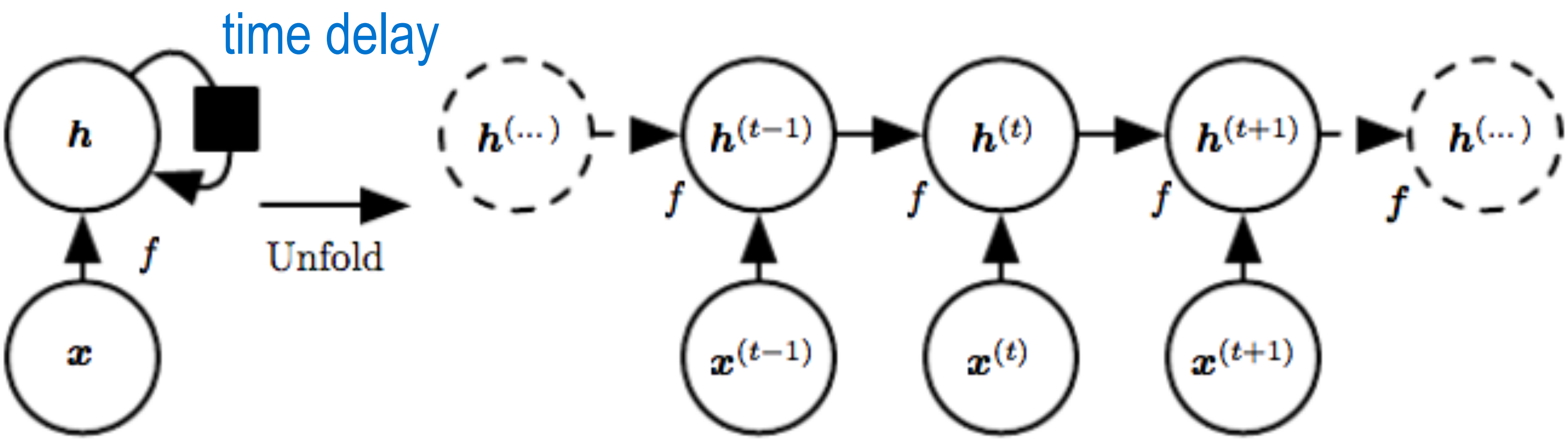
Parameter sharing: unit is produced using the same update rule applied to previous ones

$$\begin{aligned}\boldsymbol{h}^{(t)} &= g^{(t)}(\boldsymbol{x}^{(t)}, \boldsymbol{x}^{(t-1)}, \boldsymbol{x}^{(t-2)}, \ldots, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(1)}) \\ &= f(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta}).\end{aligned}$$

- $h^{(t-1)}$: old state at time t-1, memory
- $x^{(t)}$: new input at time t
- $\boldsymbol{\theta}$: parameter, shared in different time steps
- same f() function used for every time step

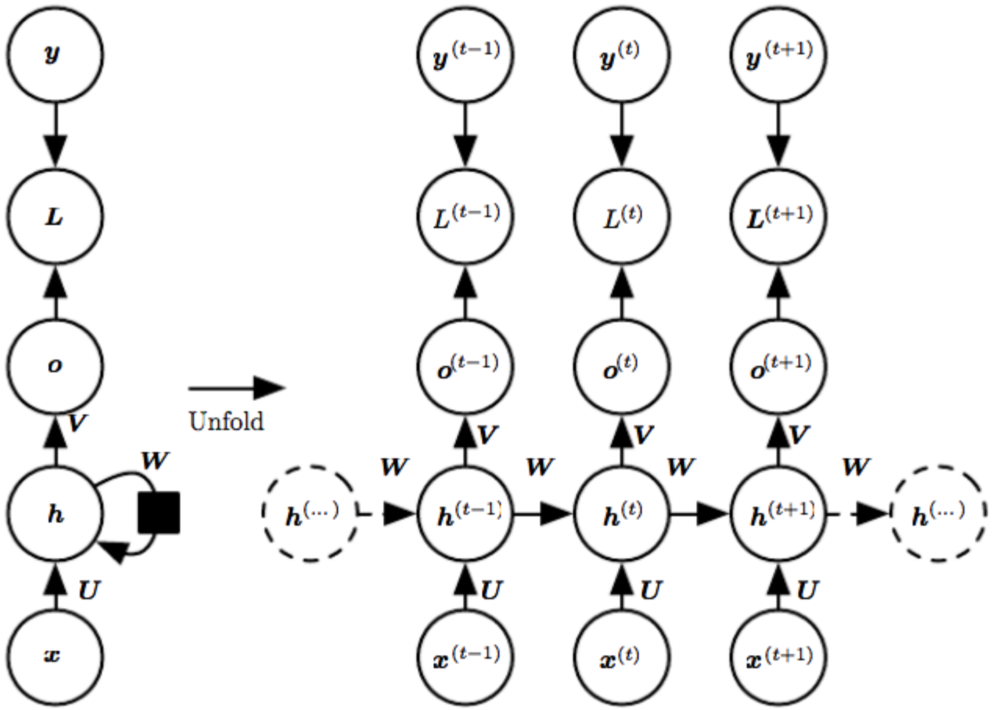
Unfolding computational graphs (1)

RNN with no output



Unfolding computational graphs (2)

- Example: RNN produces a (discrete) output at each time step and has recurrent connections between hidden units (many to many)



$$\begin{aligned}
\boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)},\\
\boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{a}^{(t)}),\\
\boldsymbol{o}^{(t)} &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)},\\
\hat{\boldsymbol{y}}^{(t)} &= \operatorname{softmax}(\boldsymbol{o}^{(t)}),
\end{aligned}$$

$$L\left(\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(\tau)}\}, \{\boldsymbol{y}^{(1)}, \dots, \boldsymbol{y}^{(\tau)}\}\right) = \sum_t L^{(t)}$$

same parameters for every step

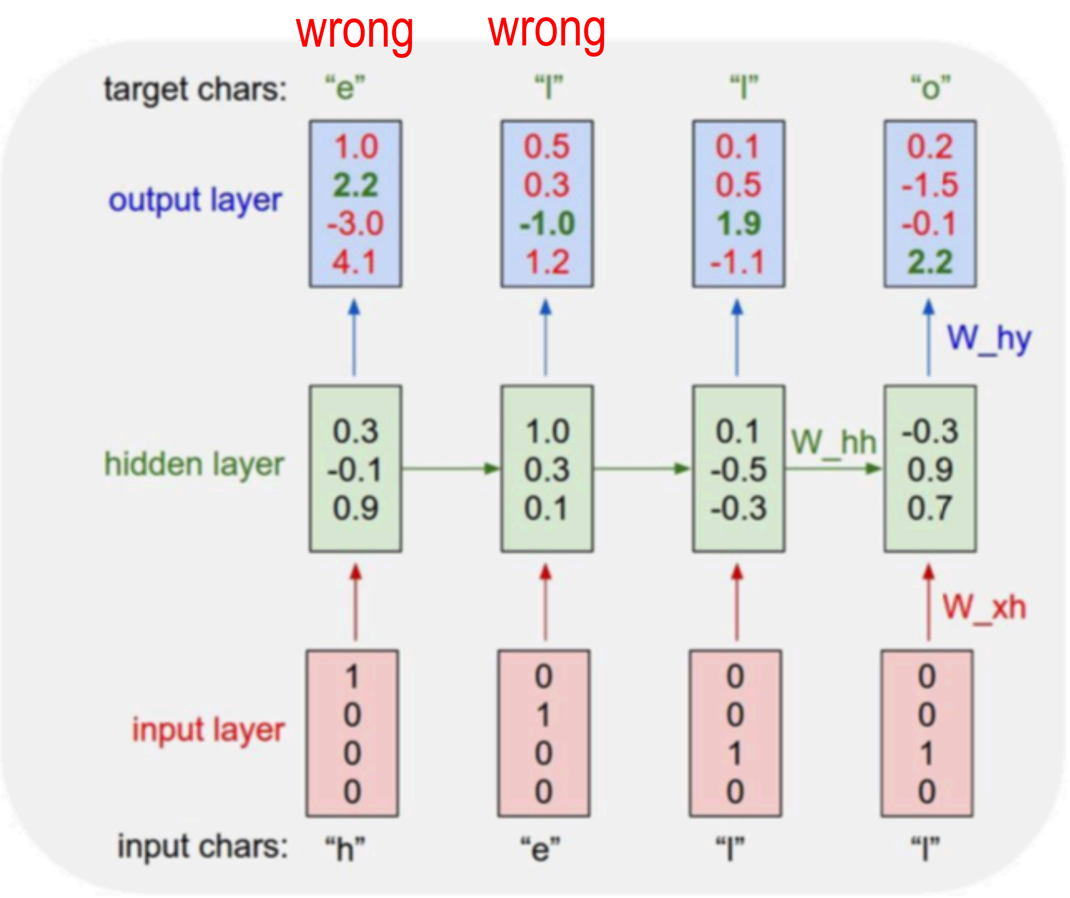
Vanilla RNN: one single hidden vector h

Language model: training

Example: Character-level Language Model

Vocabulary: [h,e,l,o]

Example training sequence: **"hello"**

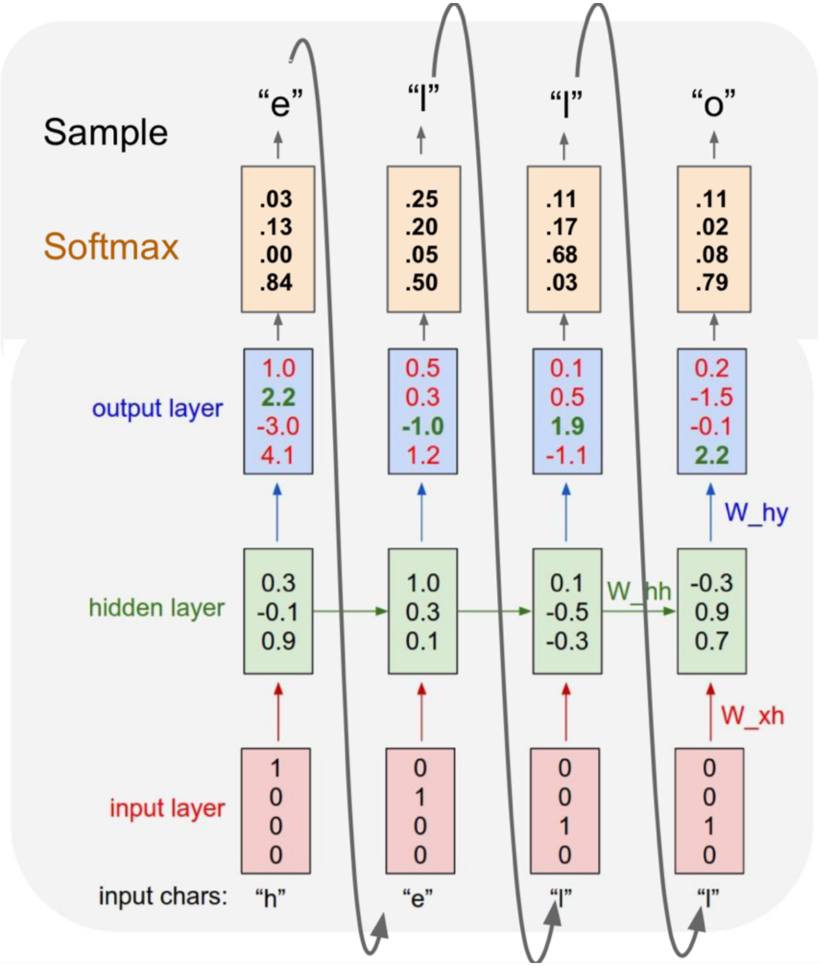


Language model: testing

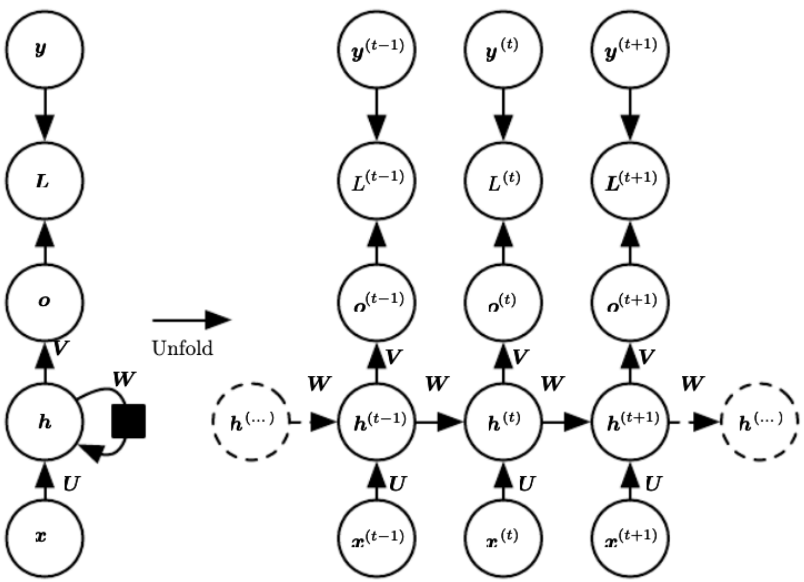
Example: Character-level Language Model Sampling

Vocabulary: [h,e,l,o]

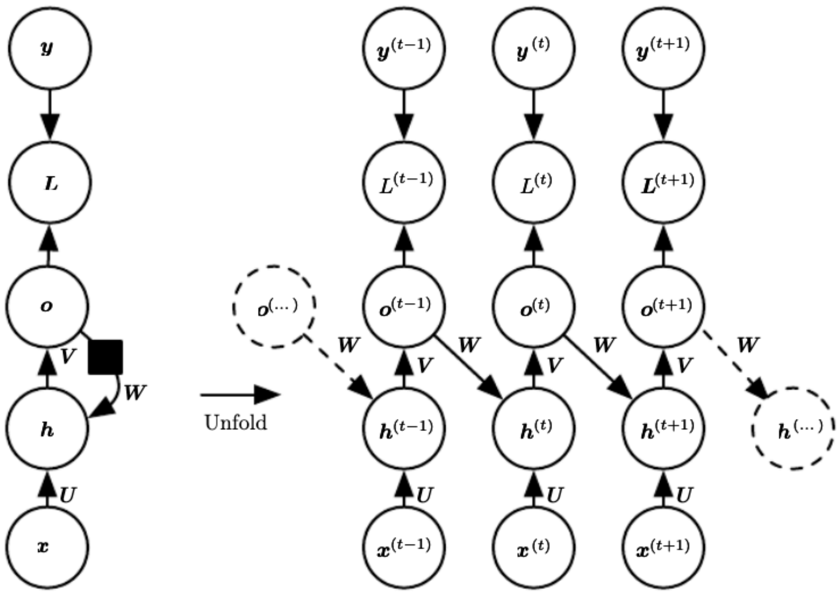
At test-time sample characters one at a time, feed back to model



Sequential vs Parallel

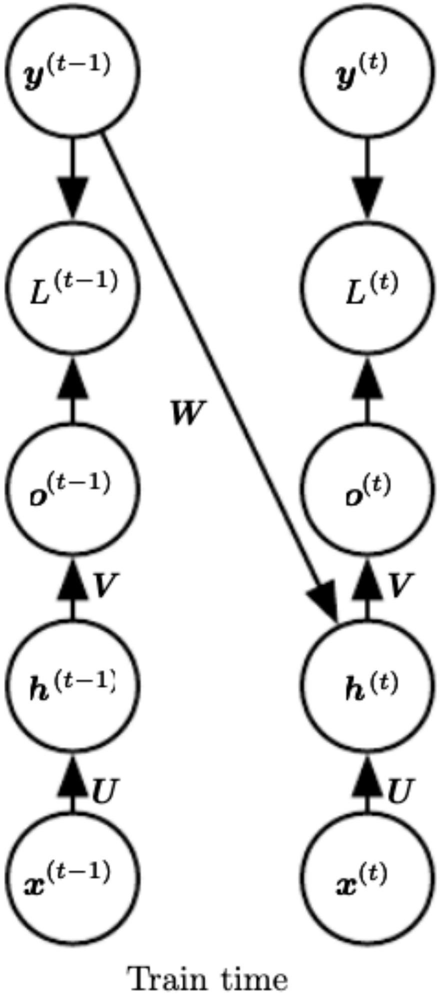


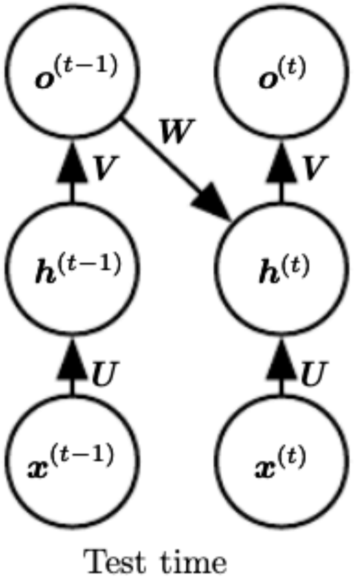
Turing complete



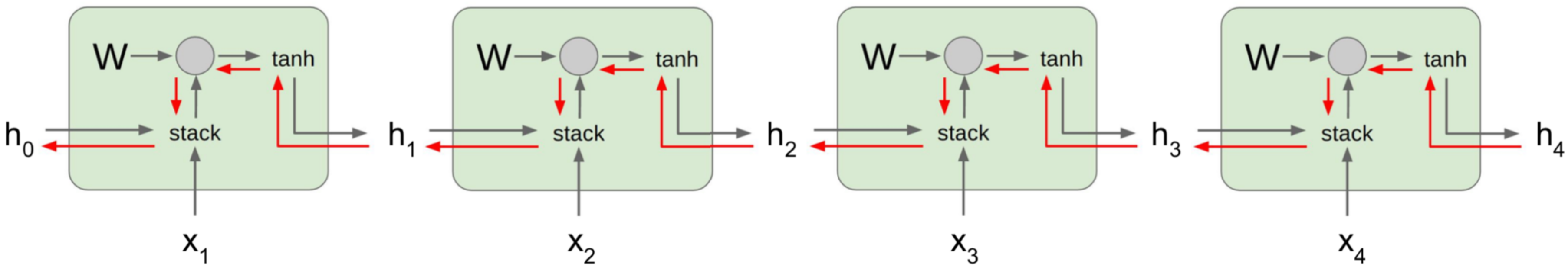
NOT Turing complete

Teacher Forcing





Vanilla RNN gradient problems



Challenge of long-term backprop

Assume W has eigen-decomposition $\boldsymbol{W} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^T; \boldsymbol{W}^t = \boldsymbol{Q}\boldsymbol{\Lambda}^t\boldsymbol{Q}^T$

- Largest eigenvalue > 1: exploding gradient
- Largest eigenvalue < 1: vanishing gradient
    - ➢ Impossible to learn correlation between temporally distant events

UNIVERSITY OF WATERLOO

Exploding gradient

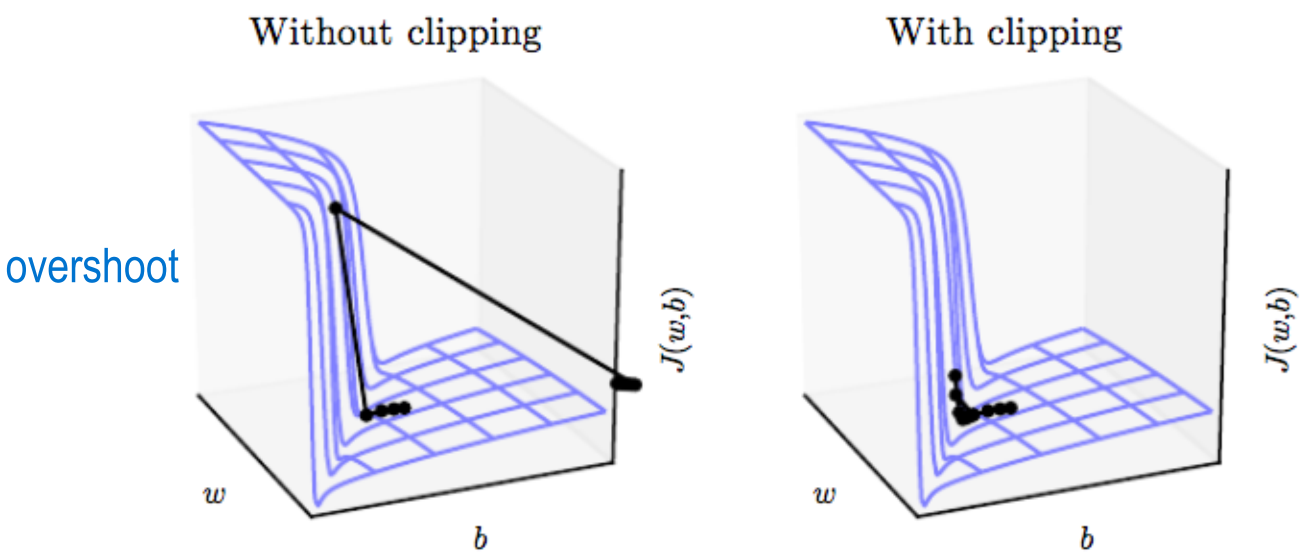
Common solution: clipping gradient

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

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

- g: gradient
- v: threshold

If the norm of gradient exceeds some threshold, clip it!



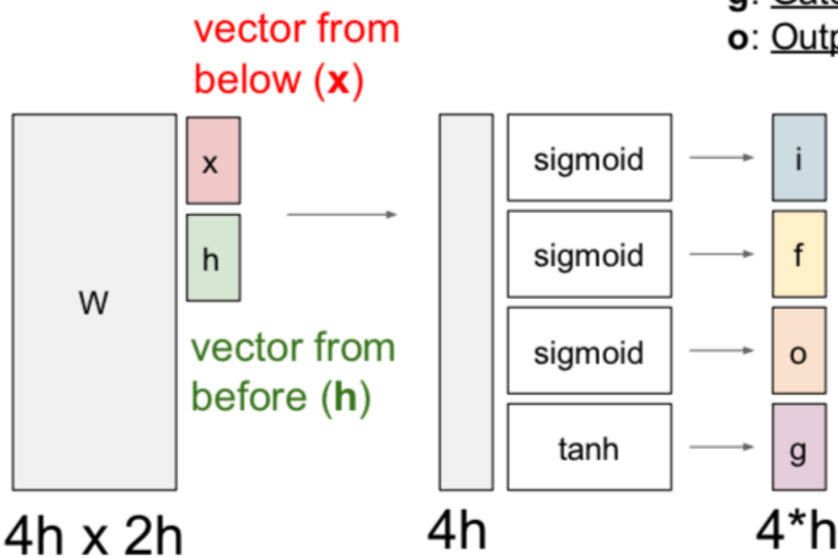
Vanishing gradient

Common solution: create paths along which the product of gradients is near 1, e.g., LSTM, GRU

Long Short Term Memory (LSTM)

[Hochreiter et al., 1997]

f: Forget gate, Whether to erase cell
i: Input gate, whether to write to cell
g: Gate gate (?), How much to write to cell
o: Output gate, How much to reveal cell



$$\boxed{\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

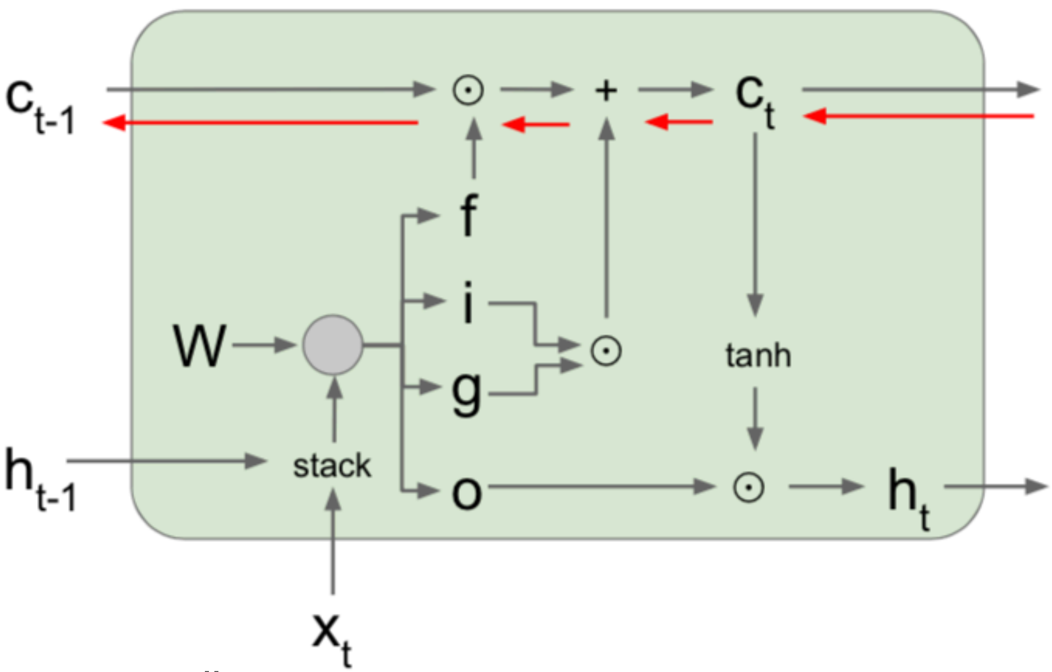
element-wise product

UNIVERSITY OF WATERLOO

Long short-term memory (LSTM)

Long Short Term Memory (LSTM): Gradient Flow

[Hochreiter et al., 1997]



Backpropagation from c_t to c_{t-1} only elementwise multiplication by f, no matrix multiply by W

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

- c_t: cell state
- h_t: hidden state

LSTM: cell state

Update of cell state

input gate	forget gate	behavior
0	1	remember the previous value
1	1	add to the previous value
0	0	erase the value
1	0	overwrite the value

can simulate a counter

Gated recurrent unit (GRU)

$$
\begin{aligned}
r_t &= \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r) \\
z_t &= \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z) \\
\tilde{h}_t &= \tanh(W_{xh}x_t + W_{hh}(r_t \odot h_{t-1}) + b_h) \\
h_t &= z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t
\end{aligned}
$$

GRU: 2 gates (r, z)

$$
\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}
$$

$$
\begin{aligned}
c_t &= f \odot c_{t-1} + i \odot g \\
h_t &= o \odot \tanh(c_t)
\end{aligned}
$$

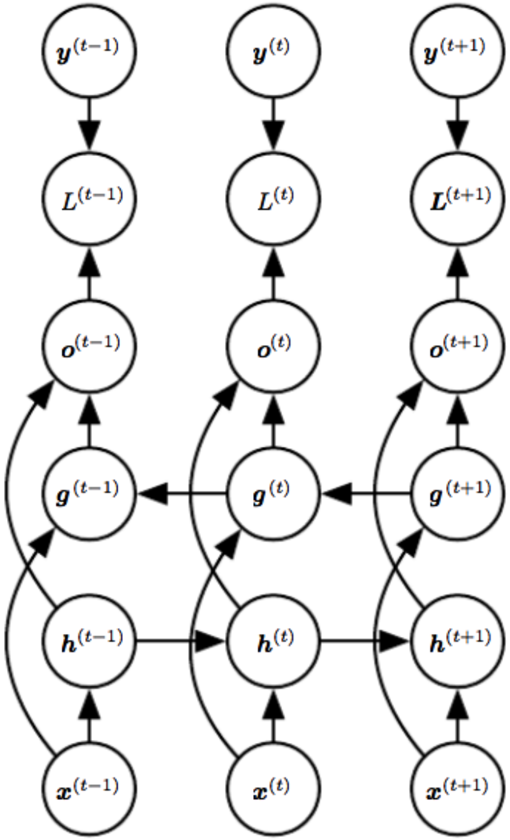
LSTM: 3 gates (i, f, o)

Bidirectional RNNs

- Idea: combine an RNN that moves forward through time with another RNN that moves backward through time
- Motivation: need future information

- E.g., name entity recognition
  - He said: ``Teddy bears are on sale!"   (not a person's name)
  - He said: ``Teddy Roosevelt was a great President!" (yes)

Illustration



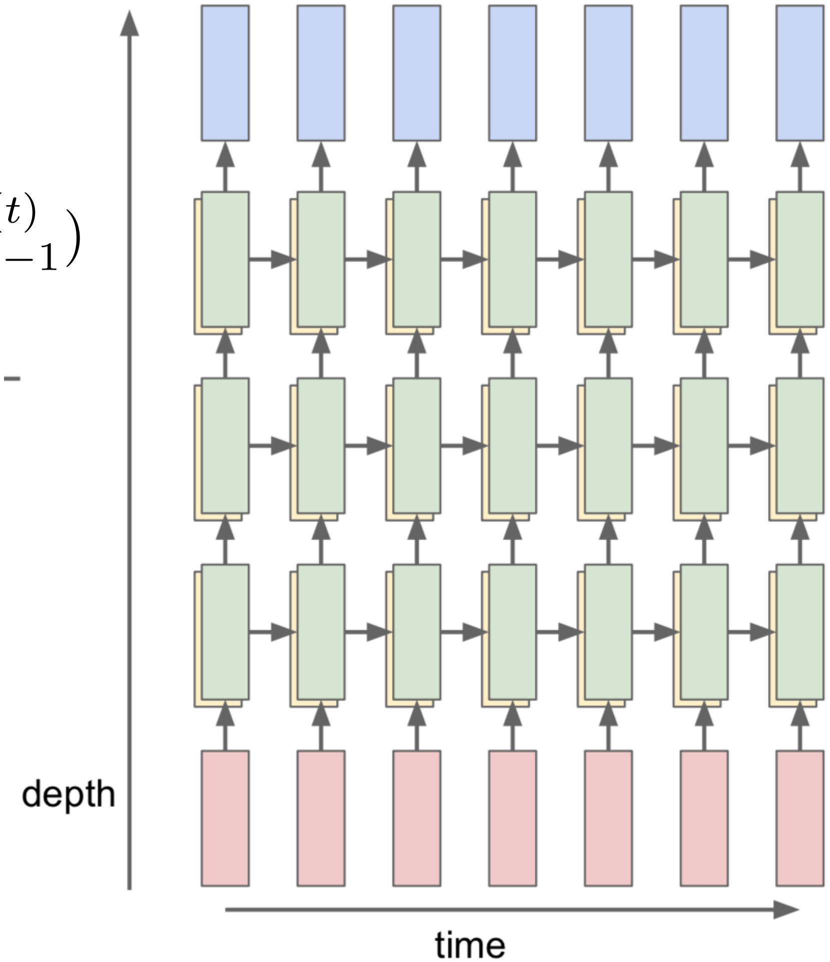
- $h^{(t)}$: propagate information forward
- $g^{(t)}$: propagate information backward
- $o^{(t)}$: output uses both $h^{(t)}$ and $g^{(t)}$:

Deep RNNs

Usually 2-3 hidden layers

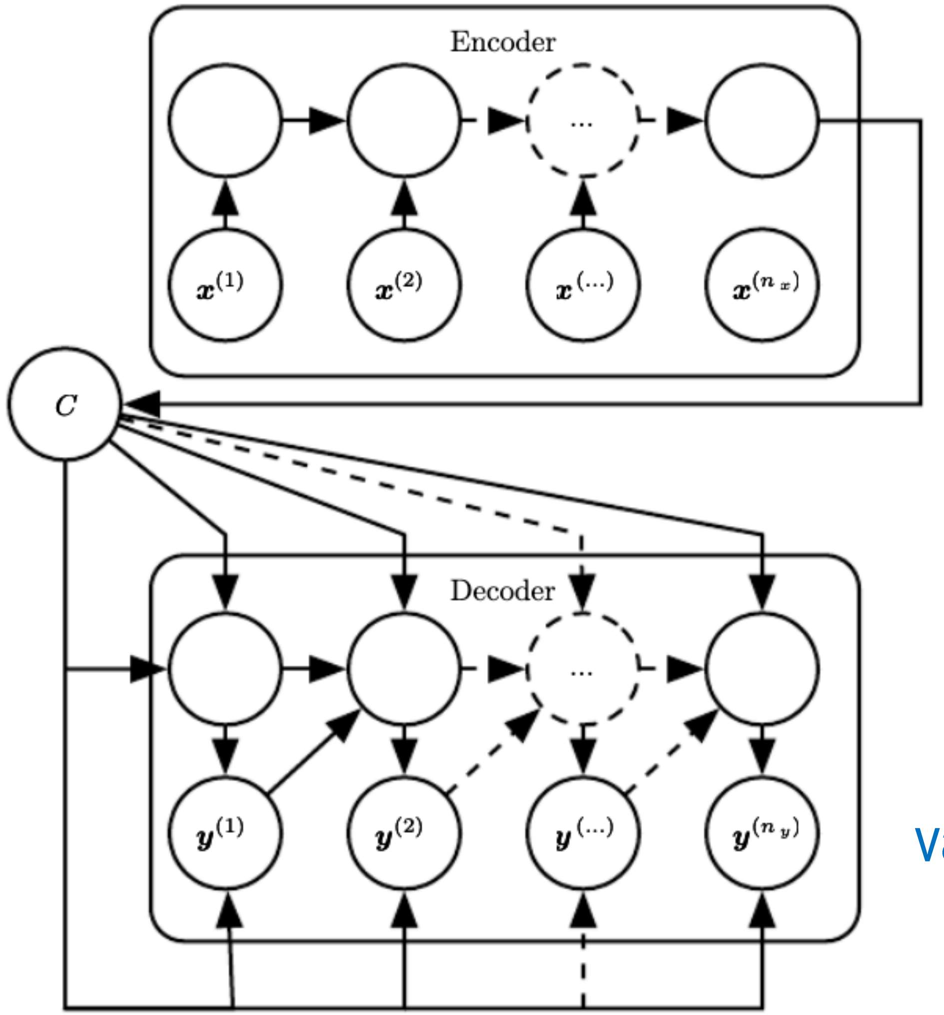
$$\mathbf{h}_l^{(t)} = \tanh(\mathbf{b}_l + \mathbf{W}_{l1}\mathbf{h}_l^{(t-1)} + \mathbf{W}_{l2}\mathbf{h}_{l-1}^{(t)})$$

- l: l-th hidden layer
- t: t-th time step



Encoder-Decoder

fixed dim context



variable dim input

variable dim output

widely used for Neural Machine Translation

Questions?

UNIVERSITY OF WATERLOO