

# Deep Learning

## 12. Recurrent Neural Networks

Dr. Konda Reddy Mopuri  
Dept. of AI, IIT Hyderabad  
Jan-May 2025

# So far...

① Perceptron, MLP, Gradient Descent (Backpropagation)

# So far...

- ① Perceptron, MLP, Gradient Descent (Backpropagation)
- ② CNNs

# So far...

- ① Perceptron, MLP, Gradient Descent (Backpropagation)
- ② CNNs
- ③ 'Feedforward Neural networks'

# Feedforward NNs: some observations

- ① Size of the i/p is fixed(?!)

# Feedforward NNs: some observations

- ① Size of the i/p is fixed(?!)
- ② Successive i/p are i.i.d.

# Feedforward NNs: some observations

- ① Size of the i/p is fixed(?!)
- ② Successive i/p are i.i.d.
- ③ Processing of successive i/p is independent of each other

# Consider 'auto-completion' task

deep|

deep — Search with Google

🕒 **kuldeep birdar**

🔍 deepika padukone

🔍 deepthi sunaina

🔍 deepak bagga

🔍 deepika pilli

🔍 deepthi sharma

① Successive i/p are not independent



# Consider 'auto-completion' task

deep|


deep — Search with Google

- 🕒 **kuldeep birdar**
- 🔍 deepika padukone
- 🔍 deepthi sunaina
- 🔍 deepak bagga
- 🔍 deepika pilli
- 🔍 deepthi sharma

- ① Successive i/p are not independent
- ② Length of the i/p is not fixed (→ predictions also)

# Consider 'auto-completion' task

🔍 deep|

 deep — Search with Google

🕒 **kuldeep birdar**

🔍 deep**pika padukone**

🔍 deep**thi sunaina**

🔍 deep**pak bagga**

🔍 deep**pika pilli**

🔍 deep**ti sharma**

- ① Successive i/p are not independent
- ② Length of the i/p is not fixed (→ predictions also)
- ③ Same underlying task at different 'time instances'

# Consider 'auto-completion' task

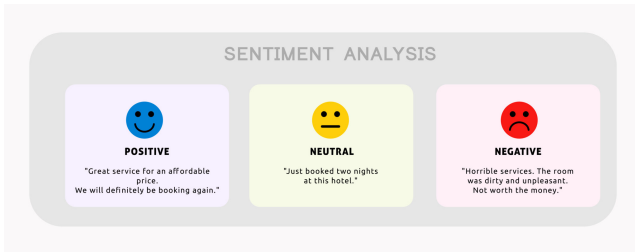
🔍 deep|

🔍 deep — Search with Google

- 🕒 **kuldeep birdar**
- 🔍 deep**pika padukone**
- 🔍 deep**thi sunaina**
- 🔍 deep**pak bagga**
- 🔍 deep**pika pilli**
- 🔍 deep**ti sharma**

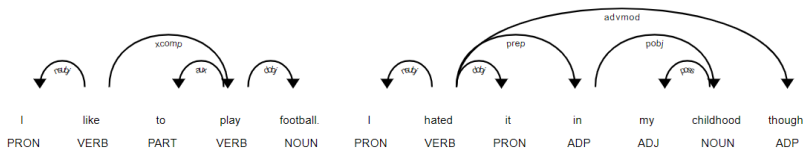
- ① Successive i/p are not independent
- ② Length of the i/p is not fixed (→ predictions also)
- ③ Same underlying task at different 'time instances'
- ④ **Sequence Learning Problems**

# Sequence Learning Tasks: Example



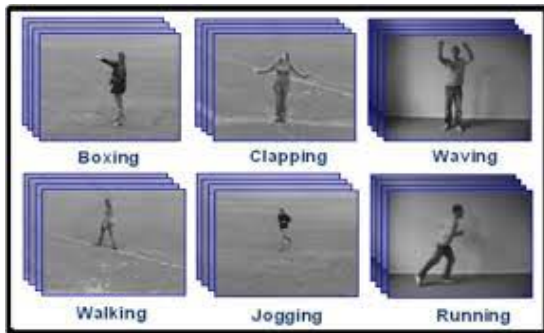
Sentiment Analysis (Source)

# Sequence Learning Tasks: Example



POS-Tagging (Source:Kaggle)

# Sequence Learning Tasks: Example



Action Recognition (Source)

# Sequence Learning Tasks: Example

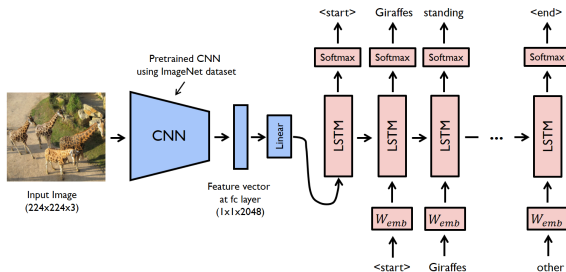
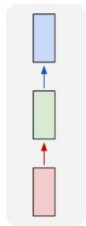


Image Captioning(Source)

# Sequence Learning Tasks: Variations

one to one

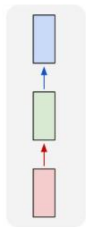


Source

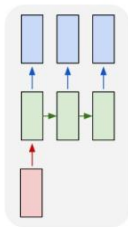


# Sequence Learning Tasks: Variations

one to one

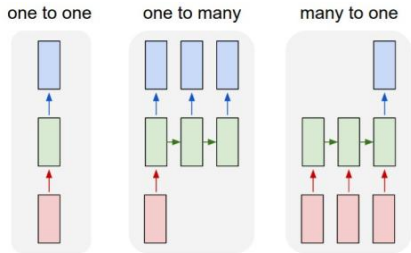


one to many



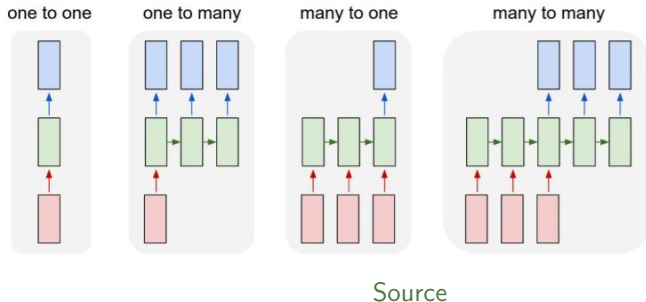
Source

# Sequence Learning Tasks: Variations

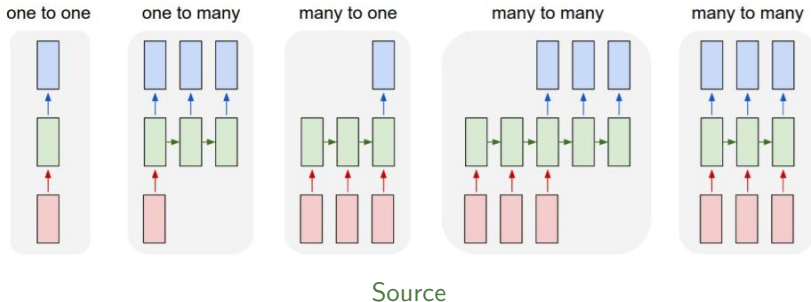


Source

# Sequence Learning Tasks: Variations



# Sequence Learning Tasks: Variations



# Can we not use the tools we already know?

- How about convolution?

# Recurrent Neural Networks (RNN)

- ① NNs designed to solve sequence learning tasks

# Recurrent Neural Networks (RNN)

- ① NNs designed to solve sequence learning tasks
- ② Characteristics
  - ① Model the dependence among the i/p

# Recurrent Neural Networks (RNN)

- ① NNs designed to solve sequence learning tasks
- ② Characteristics
  - ① Model the dependence among the i/p
  - ② Handle variable length of i/p



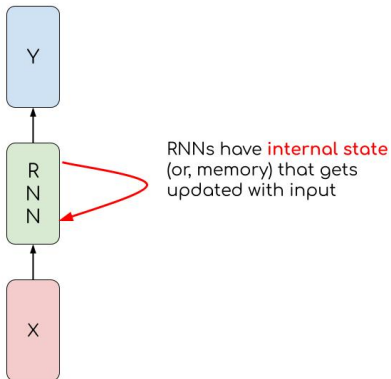
# Recurrent Neural Networks (RNN)

- ① NNs designed to solve sequence learning tasks
- ② Characteristics
  - ① Model the dependence among the i/p
  - ② Handle variable length of i/p
  - ③ Same function applied at all time instances

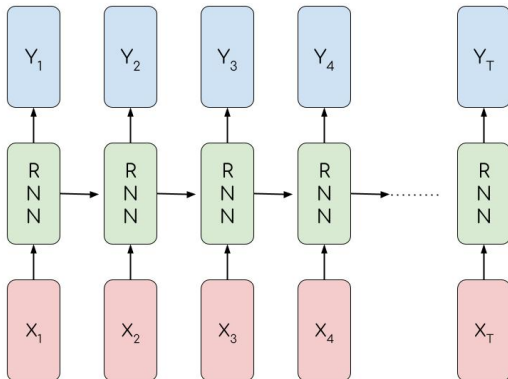
# Recurrent Neural Networks (RNN)

- ① NNs designed to solve sequence learning tasks
- ② Characteristics
  - ① Model the dependence among the i/p
  - ② Handle variable length of i/p
  - ③ Same function applied at all time instances
- ③ They are **Non-linear Auto-regressive Models**

# RNNs: internal state



# RNNs: unfolding



① Apply the same transformation at every time step  $\rightarrow$  'Recurrent' NNs

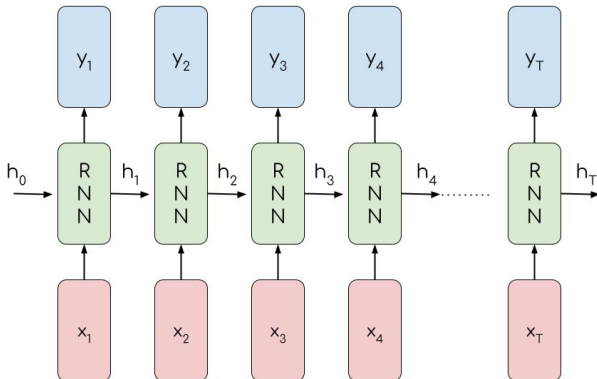
- ① Apply the same transformation at every time step  $\rightarrow$  'Recurrent' NNs
- ② i/p sequence  $x_t \in \mathbb{R}^D$

- ① Apply the same transformation at every time step  $\rightarrow$  'Recurrent' NNs
- ② i/p sequence  $x_t \in \mathbb{R}^D$
- ③ Initial recurrent state  $h_0 \in \mathbb{R}^Q$

- ① Apply the same transformation at every time step  $\rightarrow$  'Recurrent' NNs
- ② i/p sequence  $x_t \in \mathbb{R}^D$
- ③ Initial recurrent state  $h_0 \in \mathbb{R}^Q$
- ④ RNN model computes sequence of recurrent states iteratively  
$$h_t = \phi(x_t, h_{t-1}; w)$$



# RNNs



# Elmon RNN (1990)

- ① Start with  $h_0 = 0$

# Elmon RNN (1990)

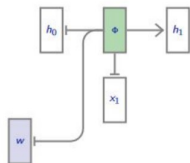
- ① Start with  $h_0 = 0$
- ②  $h_t = \tanh(W_{xh} \cdot x_t + W_{hh} \cdot h_{t-1} + b_h)$

# Elmon RNN (1990)

- ① Start with  $h_0 = 0$
- ②  $h_t = \tanh(W_{xh} \cdot x_t + W_{hh} \cdot h_{t-1} + b_h)$
- ③  $y_t = \text{softmax}(W_{hy} \cdot h_t + b_y)$

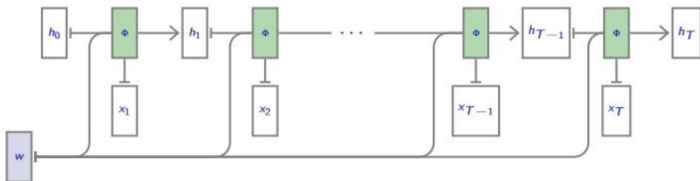
# RNNs as computational graph

- 1 Use the same set of parameters at each time step



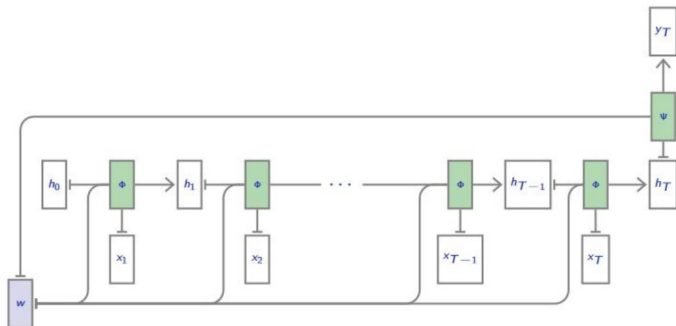
# RNNs as computational graph

- 1 Use the same set of parameters at each time step



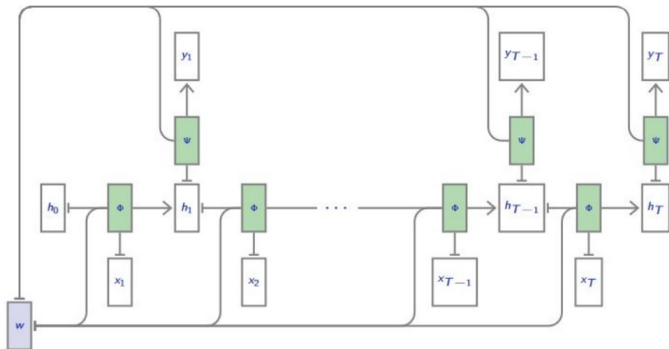
# RNNs as computational graph

- 1 Use the same set of parameters at each time step



# RNNs as computational graph

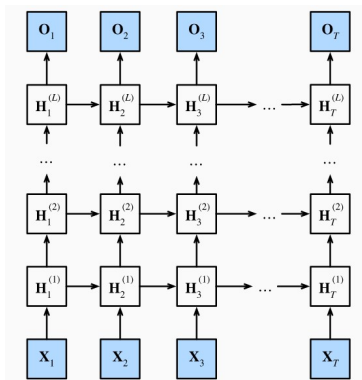
- 1 Use the same set of parameters at each time step
- 2 Flexible to realize different variants (with some tricks!)





# Multi-layered RNNs

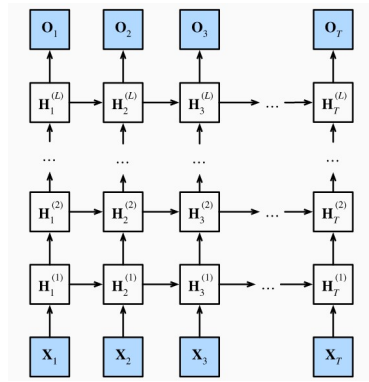
- Stack multiple RNNs between i/p and o/p layers



Source

# Multi-layered RNNs

- Stack multiple RNNs between i/p and o/p layers
- $H_t^{(l)} = W_{xh}^{(l)} \cdot X_t + W_{hh}^{(l)} \cdot H_{t-1}^{(l)} + b_h^{(l)}$

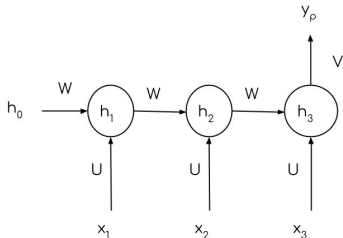


Source

# Backpropagation Through Time (BPTT)



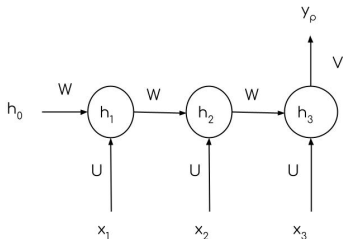
- 1 Consider a many-to-one variant RNN (e.g. sentiment analysis)



# Backpropagation Through Time (BPTT)



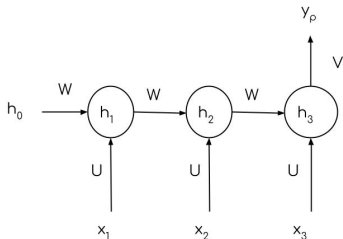
- 1 Consider a many-to-one variant RNN (e.g. sentiment analysis)
- 2 Let's separate the parameters into  $U$ ,  $V$ , and  $W$



# Backpropagation Through Time (BPTT)



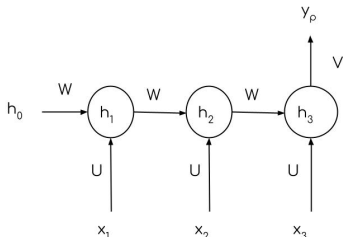
- Let's now perform SGD  
(assume loss  $L$  is  
formulated on  $y_p$ )



# Backpropagation Through Time (BPTT)



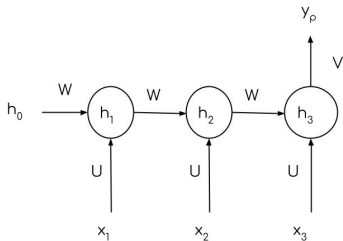
- 1 Let's now perform SGD  
(assume loss  $L$  is formulated on  $y_p$ )
- 2  $\rightarrow$  we need to compute  $\frac{\partial L}{\partial V}$ ,  $\frac{\partial L}{\partial W}$ , and  $\frac{\partial L}{\partial U}$



# Backpropagation Through Time (BPTT)



$$\textcircled{1} \frac{\partial L}{\partial V} = \frac{\partial L}{\partial y_p} \frac{\partial y_p}{\partial V} =$$
$$\frac{\partial L}{\partial y_p} \cdot \frac{\partial y_p}{\partial z_3} \cdot \frac{\partial z_3}{\partial V}$$

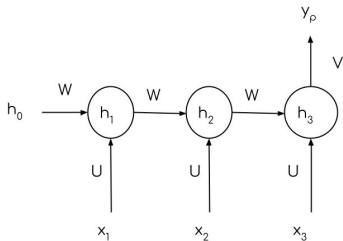


# Backpropagation Through Time (BPTT)



$$\textcircled{1} \quad \frac{\partial L}{\partial V} = \frac{\partial L}{\partial y_p} \frac{\partial y_p}{\partial V} =$$
$$\frac{\partial L}{\partial y_p} \cdot \frac{\partial y_p}{\partial z_3} \cdot \frac{\partial z_3}{\partial V}$$

$$\textcircled{2} \quad y_p = \text{softmax}(z_3) \text{ and}$$
$$z_3 = V \cdot h_3 + b_y$$





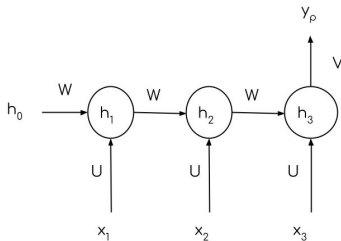
# Backpropagation Through Time (BPTT)



$$\textcircled{1} \quad \frac{\partial L}{\partial V} = \frac{\partial L}{\partial y_p} \frac{\partial y_p}{\partial V} =$$
$$\frac{\partial L}{\partial y_p} \cdot \frac{\partial y_p}{\partial z_3} \cdot \frac{\partial z_3}{\partial V}$$

$$\textcircled{2} \quad y_p = \text{softmax}(z_3) \text{ and}$$
$$z_3 = V \cdot h_3 + b_y$$

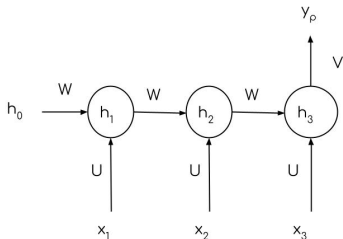
$\textcircled{3}$  Since we know that  $h_3, b_y$  are independent of  $V$ , we can compute  $\frac{\partial L}{\partial V}$  in a single step



# Backpropagation Through Time (BPTT)

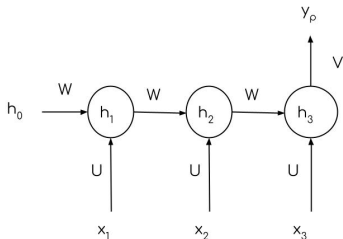


① Let's now consider  $\frac{\partial L}{\partial W}$

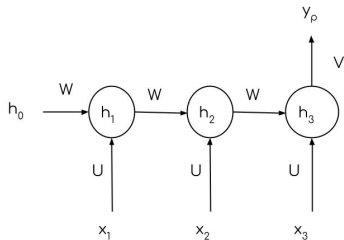


# Backpropagation Through Time (BPTT)

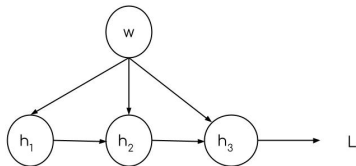
- 1 Let's now consider  $\frac{\partial L}{\partial W}$
- 2 There are multiple 'W's in the computational graph!



# Backpropagation Through Time (BPTT)

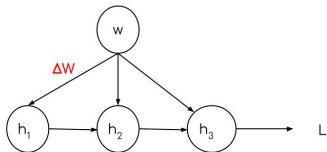


- 1 For ease of understanding



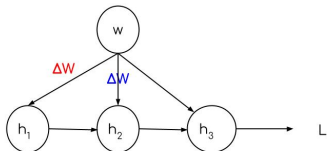
# Backpropagation Through Time (BPTT)

- ①  $\Delta w$  change in  $W \rightarrow$   
 $\left( \frac{\partial h_1}{\partial W} \cdot \Delta w \right)$  change in  $h_1$



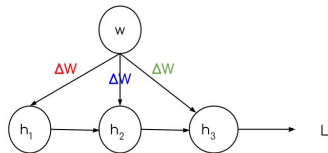
# Backpropagation Through Time (BPTT)

- ①  $\Delta w$  change in  $W \rightarrow$   
 $\left( \frac{\partial h_1}{\partial W} \cdot \Delta w \right)$  change in  $h_1$
- ②  $\Delta w$  change in  $W \rightarrow$   
 $\left( \frac{\partial h_2}{\partial W} \cdot \Delta w \right)$  change in  $h_2$



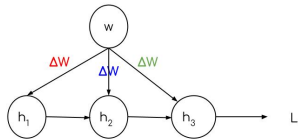
# Backpropagation Through Time (BPTT)

- ①  $\Delta w$  change in  $W \rightarrow$   
 $\left( \frac{\partial h_1}{\partial W} \cdot \Delta w \right)$  change in  $h_1$
- ②  $\Delta w$  change in  $W \rightarrow$   
 $\left( \frac{\partial h_2}{\partial W} \cdot \Delta w \right)$  change in  $h_2$
- ③  $\Delta w$  change in  $W \rightarrow$   
 $\left( \frac{\partial h_3}{\partial W} \cdot \Delta w \right)$  change in  $h_3$



# Backpropagation Through Time (BPTT)

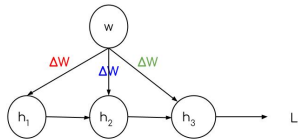
$$\textcircled{1} \Delta L = \frac{\partial L}{\partial h_1} \cdot \Delta h_1 + \frac{\partial L}{\partial h_2} \cdot \Delta h_2 + \frac{\partial L}{\partial h_3} \cdot \Delta h_3$$





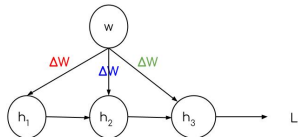
# Backpropagation Through Time (BPTT)

$$\begin{aligned} \textcircled{1} \quad \Delta L &= \\ & \frac{\partial L}{\partial h_1} \cdot \Delta h_1 + \frac{\partial L}{\partial h_2} \cdot \Delta h_2 + \frac{\partial L}{\partial h_3} \cdot \Delta h_3 \\ \textcircled{2} \quad \frac{\partial L}{\partial W} &= \frac{\partial L}{\partial h_1} \frac{\partial h_1}{\partial W} + \frac{\partial L}{\partial h_2} \frac{\partial h_2}{\partial W} + \frac{\partial L}{\partial h_3} \frac{\partial h_3}{\partial W} \end{aligned}$$



# Backpropagation Through Time (BPTT)

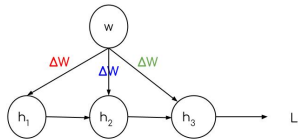
$$\begin{aligned} \textcircled{1} \quad \Delta L &= \frac{\partial L}{\partial h_1} \cdot \Delta h_1 + \frac{\partial L}{\partial h_2} \cdot \Delta h_2 + \frac{\partial L}{\partial h_3} \cdot \Delta h_3 \\ \textcircled{2} \quad \frac{\partial L}{\partial W} &= \frac{\partial L}{\partial h_1} \frac{\partial h_1}{\partial W} + \frac{\partial L}{\partial h_2} \frac{\partial h_2}{\partial W} + \frac{\partial L}{\partial h_3} \frac{\partial h_3}{\partial W} \\ \textcircled{3} \quad \frac{\partial L}{\partial h_3} &= \frac{\partial L}{\partial y_p} \frac{\partial y_p}{\partial h_3} \end{aligned}$$



# Backpropagation Through Time (BPTT)



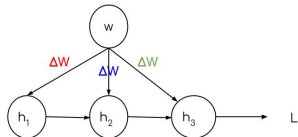
- ①  $\Delta L = \frac{\partial L}{\partial h_1} \cdot \Delta h_1 + \frac{\partial L}{\partial h_2} \cdot \Delta h_2 + \frac{\partial L}{\partial h_3} \cdot \Delta h_3$
- ②  $\frac{\partial L}{\partial W} = \frac{\partial L}{\partial h_1} \frac{\partial h_1}{\partial W} + \frac{\partial L}{\partial h_2} \frac{\partial h_2}{\partial W} + \frac{\partial L}{\partial h_3} \frac{\partial h_3}{\partial W}$
- ③  $\frac{\partial L}{\partial h_3} = \frac{\partial L}{\partial y_p} \frac{\partial y_p}{\partial h_3}$
- ④  $\frac{\partial L}{\partial h_2} = ?$



# Backpropagation Through Time (BPTT)

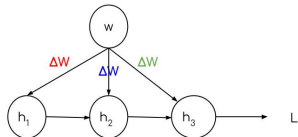


- ①  $\Delta L = \frac{\partial L}{\partial h_1} \cdot \Delta h_1 + \frac{\partial L}{\partial h_2} \cdot \Delta h_2 + \frac{\partial L}{\partial h_3} \cdot \Delta h_3$
- ②  $\frac{\partial L}{\partial W} = \frac{\partial L}{\partial h_1} \frac{\partial h_1}{\partial W} + \frac{\partial L}{\partial h_2} \frac{\partial h_2}{\partial W} + \frac{\partial L}{\partial h_3} \frac{\partial h_3}{\partial W}$
- ③  $\frac{\partial L}{\partial h_3} = \frac{\partial L}{\partial y_p} \frac{\partial y_p}{\partial h_3}$
- ④  $\frac{\partial L}{\partial h_2} = ?$
- ⑤  $\frac{\partial L}{\partial h_2} = \frac{\partial L}{\partial h_3} \frac{\partial h_3}{\partial h_2}$



# Backpropagation Through Time (BPTT)

- ①  $\Delta L = \frac{\partial L}{\partial h_1} \cdot \Delta h_1 + \frac{\partial L}{\partial h_2} \cdot \Delta h_2 + \frac{\partial L}{\partial h_3} \cdot \Delta h_3$
- ②  $\frac{\partial L}{\partial W} = \frac{\partial L}{\partial h_1} \frac{\partial h_1}{\partial W} + \frac{\partial L}{\partial h_2} \frac{\partial h_2}{\partial W} + \frac{\partial L}{\partial h_3} \frac{\partial h_3}{\partial W}$
- ③  $\frac{\partial L}{\partial h_3} = \frac{\partial L}{\partial y_p} \frac{\partial y_p}{\partial h_3}$
- ④  $\frac{\partial L}{\partial h_2} = ?$
- ⑤  $\frac{\partial L}{\partial h_2} = \frac{\partial L}{\partial h_3} \frac{\partial h_3}{\partial h_2}$
- ⑥  $\frac{\partial L}{\partial h_1} = \frac{\partial L}{\partial h_3} \frac{\partial h_3}{\partial h_1} = \frac{\partial L}{\partial h_3} \frac{\partial h_3}{\partial h_2} \frac{\partial h_2}{\partial h_1}$

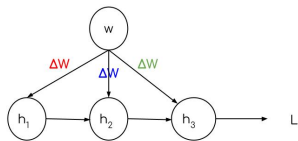


# Backpropagation Through Time (BPTT)

$$\textcircled{1} \quad \frac{\partial L}{\partial W} = \frac{\partial L}{\partial h_1} \frac{\partial h_1}{\partial W} + \frac{\partial L}{\partial h_2} \frac{\partial h_2}{\partial W} + \frac{\partial L}{\partial h_3} \frac{\partial h_3}{\partial W}$$

$$\textcircled{2} \quad \frac{\partial L}{\partial h_2} = \frac{\partial L}{\partial h_3} \frac{\partial h_3}{\partial h_2}$$

$$\textcircled{3} \quad \frac{\partial L}{\partial h_1} = \frac{\partial L}{\partial h_3} \frac{\partial h_3}{\partial h_1} = \frac{\partial L}{\partial h_3} \frac{\partial h_3}{\partial h_2} \frac{\partial h_2}{\partial h_1}$$





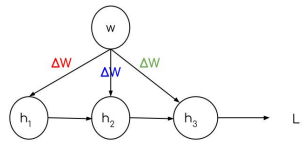
# Backpropagation Through Time (BPTT)

$$① \quad \frac{\partial L}{\partial W} = \frac{\partial L}{\partial h_1} \frac{\partial h_1}{\partial W} + \frac{\partial L}{\partial h_2} \frac{\partial h_2}{\partial W} + \frac{\partial L}{\partial h_3} \frac{\partial h_3}{\partial W}$$

$$② \quad \frac{\partial L}{\partial h_2} = \frac{\partial L}{\partial h_3} \frac{\partial h_3}{\partial h_2}$$

$$③ \quad \frac{\partial L}{\partial h_1} = \frac{\partial L}{\partial h_3} \frac{\partial h_3}{\partial h_1} = \frac{\partial L}{\partial h_3} \frac{\partial h_3}{\partial h_2} \frac{\partial h_2}{\partial h_1}$$

$$④ \quad \frac{\partial L}{\partial W} = \frac{\partial L}{\partial h_3} \frac{\partial h_3}{\partial h_2} \frac{\partial h_2}{\partial h_1} \frac{\partial h_1}{\partial W} + \frac{\partial L}{\partial h_3} \frac{\partial h_3}{\partial W}$$





# Backpropagation Through Time (BPTT)

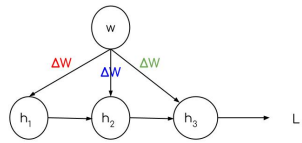
$$① \quad \frac{\partial L}{\partial W} = \frac{\partial L}{\partial h_1} \frac{\partial h_1}{\partial W} + \frac{\partial L}{\partial h_2} \frac{\partial h_2}{\partial W} + \frac{\partial L}{\partial h_3} \frac{\partial h_3}{\partial W}$$

$$② \quad \frac{\partial L}{\partial h_2} = \frac{\partial L}{\partial h_3} \frac{\partial h_3}{\partial h_2}$$

$$③ \quad \frac{\partial L}{\partial h_1} = \frac{\partial L}{\partial h_3} \frac{\partial h_3}{\partial h_1} = \frac{\partial L}{\partial h_3} \frac{\partial h_3}{\partial h_2} \frac{\partial h_2}{\partial h_1}$$

$$④ \quad \frac{\partial L}{\partial W} = \frac{\partial L}{\partial h_3} \frac{\partial h_3}{\partial h_2} \frac{\partial h_2}{\partial h_1} \frac{\partial h_1}{\partial W} + \frac{\partial L}{\partial h_3} \frac{\partial h_3}{\partial W}$$

$$⑤ \quad \frac{\partial L}{\partial W} = \sum_{k=1}^3 \frac{\partial L}{\partial h_3} \frac{\partial h_3}{\partial h_k} \frac{\partial h_k}{\partial W}$$







# Backpropagation Through Time (BPTT)

$$① \quad \frac{\partial L}{\partial W} = \frac{\partial L}{\partial h_1} \frac{\partial h_1}{\partial W} + \frac{\partial L}{\partial h_2} \frac{\partial h_2}{\partial W} + \frac{\partial L}{\partial h_3} \frac{\partial h_3}{\partial W}$$

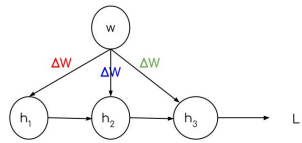
$$② \quad \frac{\partial L}{\partial h_2} = \frac{\partial L}{\partial h_3} \frac{\partial h_3}{\partial h_2}$$

$$③ \quad \frac{\partial L}{\partial h_1} = \frac{\partial L}{\partial h_3} \frac{\partial h_3}{\partial h_1} = \frac{\partial L}{\partial h_3} \frac{\partial h_3}{\partial h_2} \frac{\partial h_2}{\partial h_1}$$

$$④ \quad \frac{\partial L}{\partial W} = \frac{\partial L}{\partial h_3} \frac{\partial h_3}{\partial h_2} \frac{\partial h_2}{\partial h_1} \frac{\partial h_1}{\partial W} + \frac{\partial L}{\partial h_3} \frac{\partial h_3}{\partial W}$$

$$⑤ \quad \frac{\partial L}{\partial W} = \sum_{k=1}^3 \frac{\partial L}{\partial h_3} \frac{\partial h_3}{\partial h_k} \frac{\partial h_k}{\partial W}$$

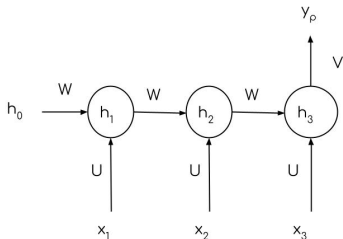
$$⑥ \quad \frac{\partial L}{\partial W} = \sum_{k=1}^3 \frac{\partial L}{\partial h_3} \left( \prod_{j=k+1}^3 \frac{\partial h_j}{\partial h_{j-1}} \right) \frac{\partial h_k}{\partial W}$$



# Backpropagation Through Time (BPTT)

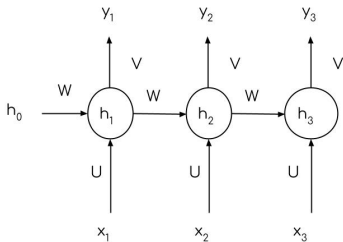


① Similarly  $\frac{\partial L}{\partial U}$



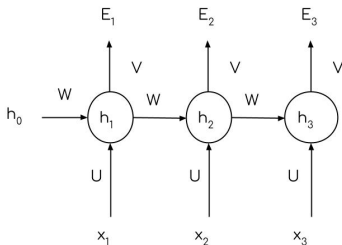
# Backpropagation Through Time (BPTT)

- 1 Consider a many-to-many variant RNN (e.g. PoS tagging)



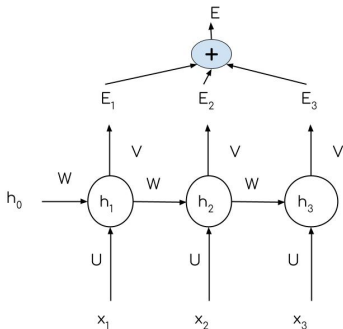
# Backpropagation Through Time (BPTT)

- 1 Consider a many-to-many variant RNN (e.g. PoS tagging)
- 2 Full sequence is one training example (although there is an error computed at each time step)



# Backpropagation Through Time (BPTT)

- 1 Consider a many-to-many variant RNN (e.g. PoS tagging)
- 2 Total error is the sum of errors at each time step (**need not always!**)



# Backpropagation Through Time (BPTT)



- ① At times, sequences can be quite lengthy!

# Backpropagation Through Time (BPTT)



- ① At times, sequences can be quite lengthy!
- ② Need to perform BPTT through many layers

# Backpropagation Through Time (BPTT)

- ① At times, sequences can be quite lengthy!
- ② Need to perform BPTT through many layers
- ③

$$\frac{\partial L}{\partial W} = \sum_{k=1}^3 \frac{\partial L}{\partial h_3} \left( \prod_{j=k+1}^3 \frac{\partial h_j}{\partial h_{j-1}} \right) \frac{\partial h_k}{\partial W}$$



# Backpropagation Through Time (BPTT)

- ① At times, sequences can be quite lengthy!
- ② Need to perform BPTT through many layers
- ③

$$\frac{\partial L}{\partial W} = \sum_{k=1}^3 \frac{\partial L}{\partial h_3} \left( \prod_{j=k+1}^3 \frac{\partial h_j}{\partial h_{j-1}} \right) \frac{\partial h_k}{\partial W}$$

- ④ Leads to Vanishing Gradient problem!

# Backpropagation Through Time (BPTT)

- ① At times, sequences can be quite lengthy!
- ② Need to perform BPTT through many layers

③

$$\frac{\partial L}{\partial W} = \sum_{k=1}^3 \frac{\partial L}{\partial h_3} \left( \prod_{j=k+1}^3 \frac{\partial h_j}{\partial h_{j-1}} \right) \frac{\partial h_k}{\partial W}$$

- ④ Leads to Vanishing Gradient problem!
- ⑤ No impact of earlier time steps at later times (**difficult to learn long-term dependencies!**)

# Backpropagation Through Time (BPTT)

- ① In some cases  $\left( \prod_{j=k+1}^3 \frac{\partial h_j}{\partial h_{j-1}} \right)$  may lead to exploding gradients

# Backpropagation Through Time (BPTT)

- ① In some cases  $\left( \prod_{j=k+1}^3 \frac{\partial h_j}{\partial h_{j-1}} \right)$  may lead to exploding gradients
- ② But, not much of an issue

# Backpropagation Through Time (BPTT)

- ① In some cases  $\left( \prod_{j=k+1}^3 \frac{\partial h_j}{\partial h_{j-1}} \right)$  may lead to exploding gradients
- ② But, not much of an issue
  - Easy to diagnose (NaN)

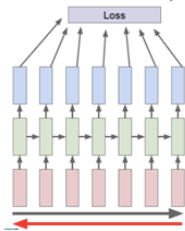
# Backpropagation Through Time (BPTT)

- ① In some cases  $\left( \prod_{j=k+1}^3 \frac{\partial h_j}{\partial h_{j-1}} \right)$  may lead to exploding gradients
- ② But, not much of an issue
  - Easy to diagnose (NaN)
  - Gradient clipping

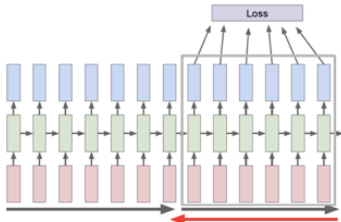
# Backpropagation Through Time (BPTT)

- ① In some cases  $\left( \prod_{j=k+1}^3 \frac{\partial h_j}{\partial h_{j-1}} \right)$  may lead to exploding gradients
- ② But, not much of an issue
  - Easy to diagnose (NaN)
  - Gradient clipping
- ③ Better initialization, Regularization, short time sequences (Truncation)

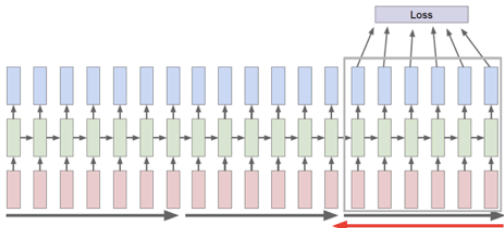
# Backpropagation Through Time (BPTT)



(a)



(b)



(c)

## Truncated BPTT (CS231n)



# Handling long-term dependencies

## ① Architectural modifications to RNNs

# Handling long-term dependencies

## ① Architectural modifications to RNNs

- LSTM (1997 by Sepp Hochreiter and Jürgen Schmidhuber; Improved by Gers et al. in 2000)

## ① Architectural modifications to RNNs

- LSTM (1997 by Sepp Hochreiter and Jürgen Schmidhuber; Improved by Gers et al. in 2000)
- GRU (Cho et al. 2014)

## ① Long Short-Term Memory

- ① Long Short-Term Memory
- ② At a time 't', **hidden state**  $h^{(t)}$  and **cell state**  $c^{(t)}$

- ① Long Short-Term Memory
- ② At a time 't', **hidden state**  $h^{(t)}$  and **cell state**  $c^{(t)}$ 
  - Cell stores long-term information

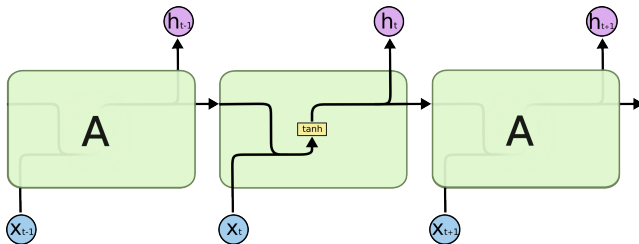
- ① Long Short-Term Memory
- ② At a time 't', **hidden state**  $h^{(t)}$  and **cell state**  $c^{(t)}$ 
  - Cell stores long-term information
  - LSTM can **erase, write, and read** information from the cell

- ① Long Short-Term Memory
- ② At a time 't', **hidden state**  $h^{(t)}$  and **cell state**  $c^{(t)}$ 
  - Cell stores long-term information
  - LSTM can **erase, write, and read** information from the cell
- ③ What to erase/write/read is controlled by corresponding **gates**



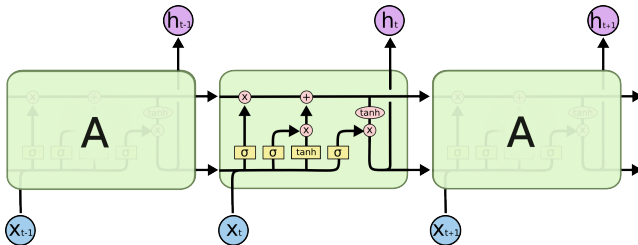
- ① Long Short-Term Memory
- ② At a time 't', **hidden state**  $h^{(t)}$  and **cell state**  $c^{(t)}$ 
  - Cell stores long-term information
  - LSTM can **erase, write, and read** information from the cell
- ③ What to erase/write/read is controlled by corresponding **gates**
  - At time t, elements of the gates can be 0 (closed), 1 (open), or in-between

- ① Long Short-Term Memory
- ② At a time 't', **hidden state**  $h^{(t)}$  and **cell state**  $c^{(t)}$ 
  - Cell stores long-term information
  - LSTM can **erase, write, and read** information from the cell
- ③ What to erase/write/read is controlled by corresponding **gates**
  - At time t, elements of the gates can be 0 (closed), 1 (open), or in-between
  - Gates are dynamically computed based on the context



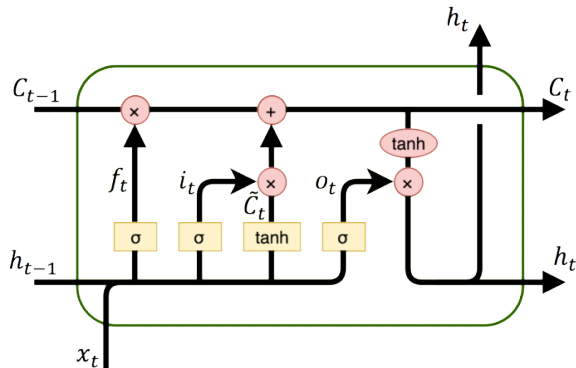
RNNs are chain of repeating moduels. Basic RNN (Colah's blog)

# LSTM



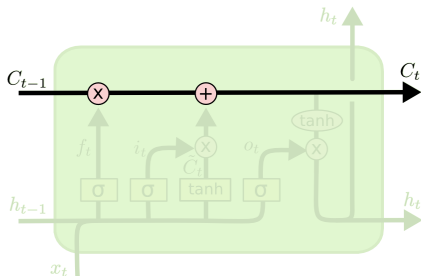
RNNs are chain of repeating moduels. LSTM (Colah's blog)

# LSTM



The LSTM node. (Colah's blog)

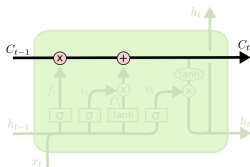
# LSTM: The cell state



Cell state in LSTM (Colah's blog)

# LSTM: The cell state

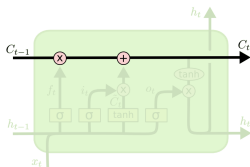
- 1 Info. can flow through unchanged



Cell state in LSTM (Colah's blog)

# LSTM: The cell state

- 1 Info. can flow through unchanged
- 2 Gates can add/remove information to cell state

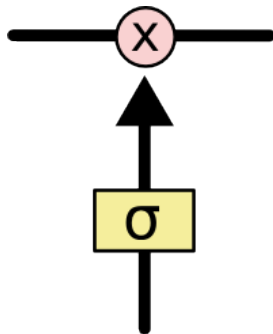


Cell state in LSTM (Colah's blog)



# LSTM: The gates

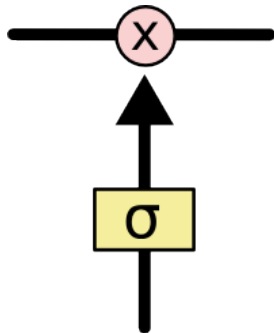
- 1 Sigmoid neural nets (o/p numbers in  $[0, 1]$ )



Cell state in LSTM (Colah's blog)

# LSTM: The gates

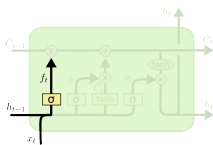
- ① Sigmoid neural nets (o/p numbers in  $[0, 1]$ )
- ② Point-wise multiplication operation



Cell state in LSTM (Colah's blog)

# LSTM: The forget gate

- 1 Decides what to throw away from cell state (e.g. forgetting the gender of old subject in light of a new one)

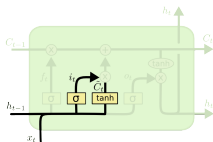


$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

Forget gate in LSTM (Colah's blog)

# LSTM: The input gate

- 1 Next is to decide what new to store in cell state (e.g. add the gender of a new subject)

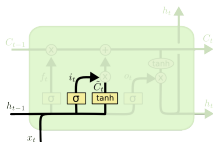


$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

Input gate in LSTM (Colah's blog)

# LSTM: The input gate

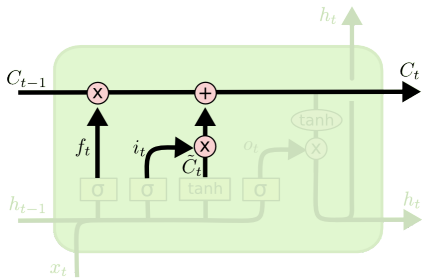
- 1 Next is to decide what new to store in cell state (e.g. add the gender of a new subject)
- 2 Done in two steps
  - input gate decides what to update
  - A tanh layer creates a candidate cell state



$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

Input gate in LSTM (Colah's blog)

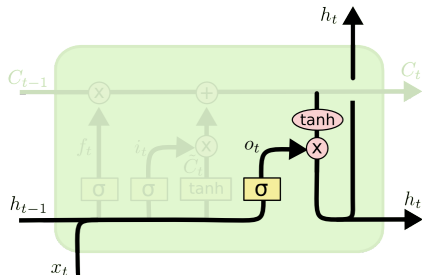
# LSTM: The cell state update



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Cell state update in LSTM (Colah's blog)

# LSTM: The output



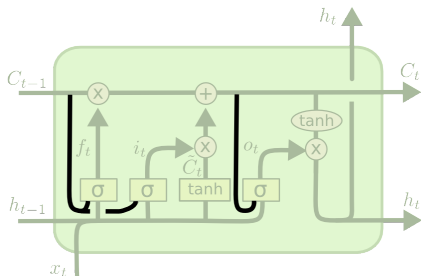
$$o_t = \sigma (W_o [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh (C_t)$$

Output computation in LSTM (Colah's blog)

e.g. may be a verb that is coming next in case of a language model

# LSTM variant: Peephole connections



$$f_t = \sigma (W_f \cdot [C_{t-1}, h_{t-1}, x_t] + b_f)$$

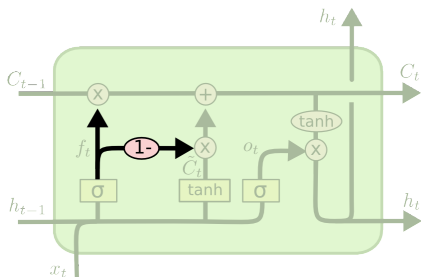
$$i_t = \sigma (W_i \cdot [C_{t-1}, h_{t-1}, x_t] + b_i)$$

$$o_t = \sigma (W_o \cdot [C_t, h_{t-1}, x_t] + b_o)$$

Variant with gates looking into the Cell state in LSTM by Ger et al. (Colah's blog)



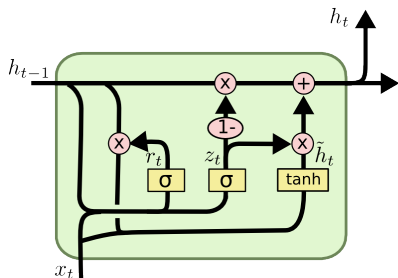
# LSTM variant: Coupled i/p and forget gates



$$C_t = f_t * C_{t-1} + (1 - f_t) * \tilde{C}_t$$

Variant with coupled input and forget gates. (Colah's blog)

# LSTM → GRU



$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$

$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t])$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Gated Recurrent Unit (Colah's blog)

# LSTM: handling the vanishing gradients



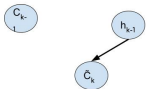
① Via the gates!

# LSTM: handling the vanishing gradients



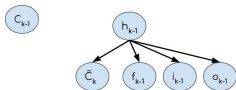
- 1 Computational graph at time  $k - 1$

# LSTM: handling the vanishing gradients



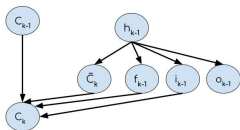
$$\textcircled{1} \quad \tilde{C}_k = \tanh(W_c[h_{t-1}, x_t] + b_c)$$

# LSTM: handling the vanishing gradients



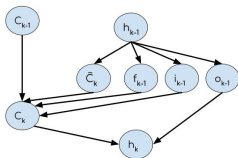
① All the gates

# LSTM: handling the vanishing gradients



1 Next cell state

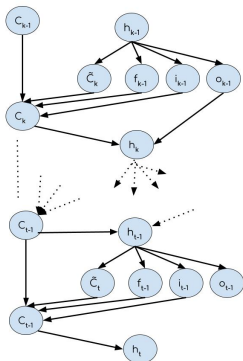
# LSTM: handling the vanishing gradients



1 Next hidden state

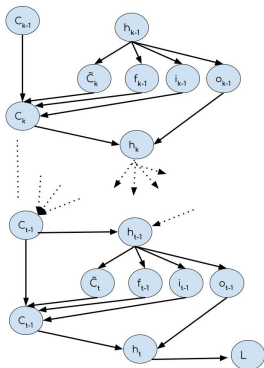


# LSTM: handling the vanishing gradients



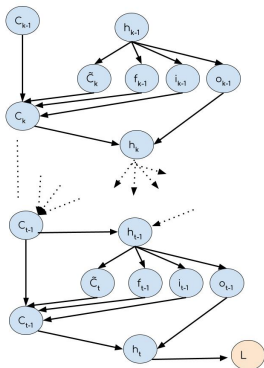
1 Running till time step 't'

# LSTM: handling the vanishing gradients



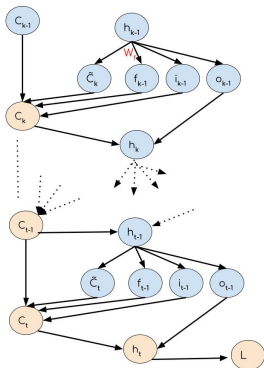
- 1 Consider loss computation

# LSTM: handling the vanishing gradients



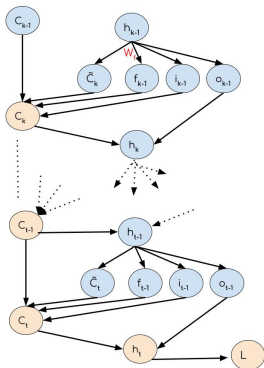
- 1 Let's know if the gradient flows to an arbitrary time step 'k'

# LSTM: handling the vanishing gradients



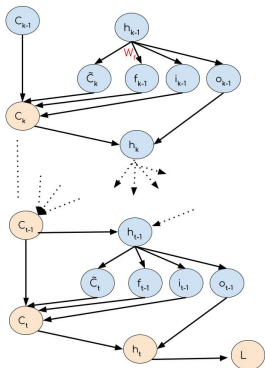
- 1 Specifically, let's consider if gradient flows to  $W_f$  through  $C_k$

# LSTM: handling the vanishing gradients



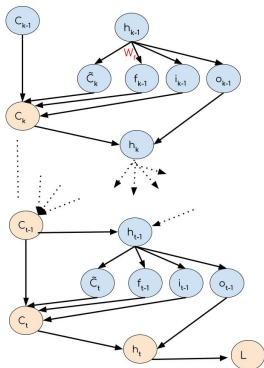
- 1 Specifically, let's consider if gradient flows to  $W_f$  through  $C_k$
- 2 Note that there are multiple paths between  $L$  and  $C_k$  (but, consider one such path as highlighted)

# LSTM: handling the vanishing gradients



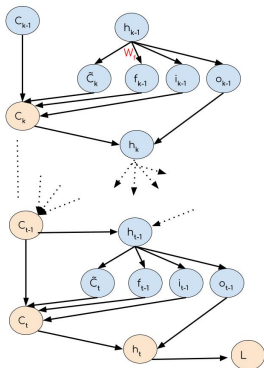
① Grad = 
$$\frac{\partial L}{\partial h_t} \frac{\partial h_t}{\partial C_t} \frac{\partial C_t}{\partial C_{t-1}} \cdots \frac{\partial C_{k+1}}{\partial C_k}$$

# LSTM: handling the vanishing gradients



- 1 Grad =  $\frac{\partial L}{\partial h_t} \frac{\partial h_t}{\partial C_t} \frac{\partial C_t}{\partial C_{t-1}} \cdots \frac{\partial C_{k+1}}{\partial C_k}$
- 2  $\frac{\partial L}{\partial h_t}$  doesn't vanish (no intermediate nodes)

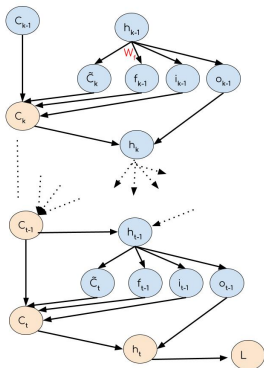
# LSTM: handling the vanishing gradients



- 1 Grad =  $\frac{\partial L}{\partial h_t} \frac{\partial h_t}{\partial C_t} \frac{\partial C_t}{\partial C_{t-1}} \cdots \frac{\partial C_{k+1}}{\partial C_k}$
- 2  $\frac{\partial L}{\partial h_t}$  doesn't vanish (no intermediate nodes)
- 3  $h_t = o_t \odot \sigma(C_t)$

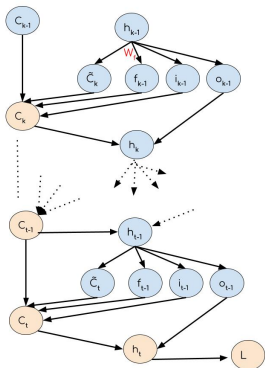


# LSTM: handling the vanishing gradients



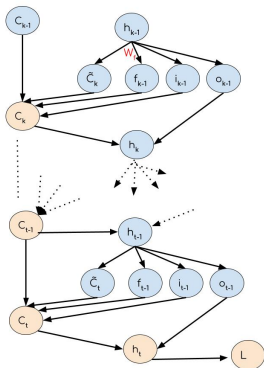
- 1 Grad =  $\frac{\partial L}{\partial h_t} \frac{\partial h_t}{\partial C_t} \frac{\partial C_t}{\partial C_{t-1}} \cdots \frac{\partial C_{k+1}}{\partial C_k}$
- 2  $\frac{\partial L}{\partial h_t}$  doesn't vanish (no intermediate nodes)
- 3  $h_t = o_t \odot \sigma(C_t)$
- 4  $\rightarrow \frac{\partial h_t}{\partial C_t} = \mathbb{D}(o_t \odot \sigma'(C_t))$   
(diagonal matrix)

# LSTM: handling the vanishing gradients



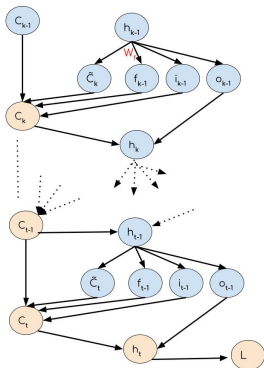
$$\textcircled{1} C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$

# LSTM: handling the vanishing gradients



- ①  $C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$
- ② Note that  $\tilde{C}_t$  depends on  $C_{t-1}$ , and for simplicity assume the gradient from that term vanishes

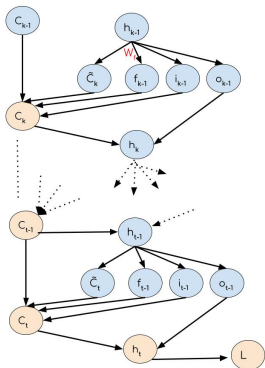
# LSTM: handling the vanishing gradients



- 1  $C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$
- 2 Note that  $\tilde{C}_t$  depends on  $C_{t-1}$ , and for simplicity assume the gradient from that term vanishes
- 3 Grad =

$$\frac{\partial L}{\partial h_t} \frac{\partial h_t}{\partial C_t} \frac{\partial C_t}{\partial C_{t-1}} \cdots \frac{\partial C_{k+1}}{\partial C_k}$$

# LSTM: handling the vanishing gradients



- 1  $C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$
- 2 Note that  $\tilde{C}_t$  depends on  $C_{t-1}$ , and for simplicity assume the gradient from that term vanishes
- 3 Grad = 
$$\frac{\partial L}{\partial h_t} \frac{\partial h_t}{\partial C_t} \frac{\partial C_t}{\partial C_{t-1}} \cdots \frac{\partial C_{k+1}}{\partial C_k}$$
- 4 Grad = 
$$L' \cdot \mathbb{D}(o_t \odot \sigma'(C_t)) \mathbb{D}(f_t) \cdot \mathbb{D}(f_{t-1}) \cdots \mathbb{D}(f_{k+1})$$

# LSTM: handling the vanishing gradients



$$\textcircled{1} \text{ Grad} = L' \cdot \mathbb{D}(o_t \odot \sigma'(C_t)) \mathbb{D}(f_t) \cdot \mathbb{D}(f_{t-1}) \dots \mathbb{D}(f_{k+1})$$

# LSTM: handling the vanishing gradients

- ①  $\text{Grad} = L' \cdot \mathbb{D}(o_t \odot \sigma'(C_t)) \mathbb{D}(f_t) \cdot \mathbb{D}(f_{t-1}) \dots \mathbb{D}(f_{k+1})$
- ② Red term vanishes only if during the forward pass this product caused the information to vanish (by the time 't')!

# LSTM: handling the vanishing gradients

- ①  $\text{Grad} = L' \cdot \mathbb{D}(o_t \odot \sigma'(C_t)) \mathbb{D}(f_t) \cdot \mathbb{D}(f_{t-1}) \dots \mathbb{D}(f_{k+1})$
- ② Red term vanishes only if during the forward pass this product caused the information to vanish (by the time 't')!
- ③ That means, gradient will vanish only if dependency in the forward pass vanishes! (which makes sense)



# LSTM: handling the vanishing gradients

- ①  $\text{Grad} = \mathbf{L}' \cdot \mathbf{D}(o_t \odot \sigma'(C_t)) \mathbf{D}(f_t) \cdot \mathbf{D}(f_{t-1}) \dots \mathbf{D}(f_{k+1})$
- ② Red term vanishes only if during the forward pass this product caused the information to vanish (by the time 't')!
- ③ That means, gradient will vanish only if dependency in the forward pass vanishes! (which makes sense)
- ④ Gates do the same regulation in backward pass as they do in the forward

① Dominated until 2015

- ① Dominated until 2015
- ② Driven applications such as handwriting recognition, ASR, Machine translation, Parsing, image captioning, VQA, etc.

- ① Dominated until 2015
- ② Driven applications such as handwriting recognition, ASR, Machine translation, Parsing, image captioning, VQA, etc.
- ③ Attention and Transformers are becoming more popular lately