# Natural Language Processing
## CSE 325/425

Sihong Xie

Lecture 13:

- Recurrent neural networks (RNN)

(Not Recursive)
for trees

# Language model review

Bi-gram $$P(w_t|w_{t-1}) = \frac{\text{Count } w_t}{\text{Count } (w_{t-1}, w_t)}$$

$n$-gram $$P(w_t|[w_{t-1}, \ldots, w_{t-n+1}]) = \frac{\text{Count } [w_t, \ldots, w_{t-n+1}] + \varepsilon}{\text{Count } [w_{t-\cancel{1}}, \ldots, w_{t-n+1}] + |V|\varepsilon}$$

$C(\text{"race"})$

$\geq C(\text{"race a"})$

$\geq C(\text{"race a car"})$

$\geq \ldots$

Two issues

Laplacian smooth

- Data sparsity: the occurrences of many $[w_t, \ldots, w_{t-n+1}]$ are zeros.
- Model complexity: number of parameters increases exponentially in *n*.

**The two issues are related**: if we want longer range dependencies, we increase *n*, then both the data sparsity and model complexity become worse.

*The students walked in the room and asked the ___?___ about the quiz questions.*

w?

RNN:

$\Pr(w \mid \text{"The students} \ldots \ldots \text{asked the"}; \vec{\theta})$

tri-gram: $\Pr(w \mid \text{"asked the"})$

If Count("asked the") = 0 on training corpus

# Address the issues using neural network

Don't store the *n*-grams, but use a $\underbrace{\text{fixed-size model}}_{\theta \text{ is given}}$ to predict the *n*-grams.

$\Pr(w_t \mid w_{t-1} \ldots w_{t-n+1})$

- model complexity is fixed.
- no data sparsity issue (no *n*-gram is computed)
- can be generalized to unseen sequences.

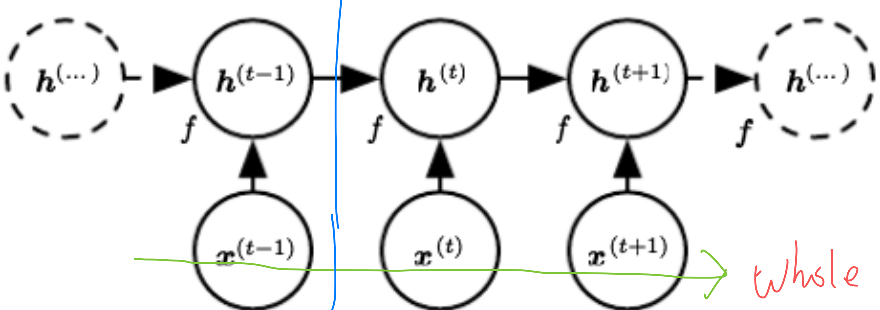
**Recurrent Neural Networks (RNN)**

history before $x^{(t)}$

$$= f(\boxed{f(h^{(t-2)},}\, x^{(t-1)}; \theta), x^{(t)}; \theta)$$

$$\boldsymbol{h}^{(t)} = f(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta})$$

"*Recurrent*" because the next $\boldsymbol{h}$ is compute by the same function that calculates the previous $\boldsymbol{h}$.




$\mathbb{R}^{d_1} \ni \boldsymbol{h}^{(t)}$ state summarizing what has happened before time step $t$. (theoretically speaking)

whole collection of parameters $\rightarrow \boldsymbol{\theta}$ a single model specifying how to transit to the next state (independent of $t$).

Current Input

$d_0, d_1$ : Integers of your choice

$x^{(t)} \in \mathbb{R}^{d_0}$ (e.g. word embedding by Glove for the $t$-th word)

# A running example

# Recurrent Neural Networks

supervisions (labels)

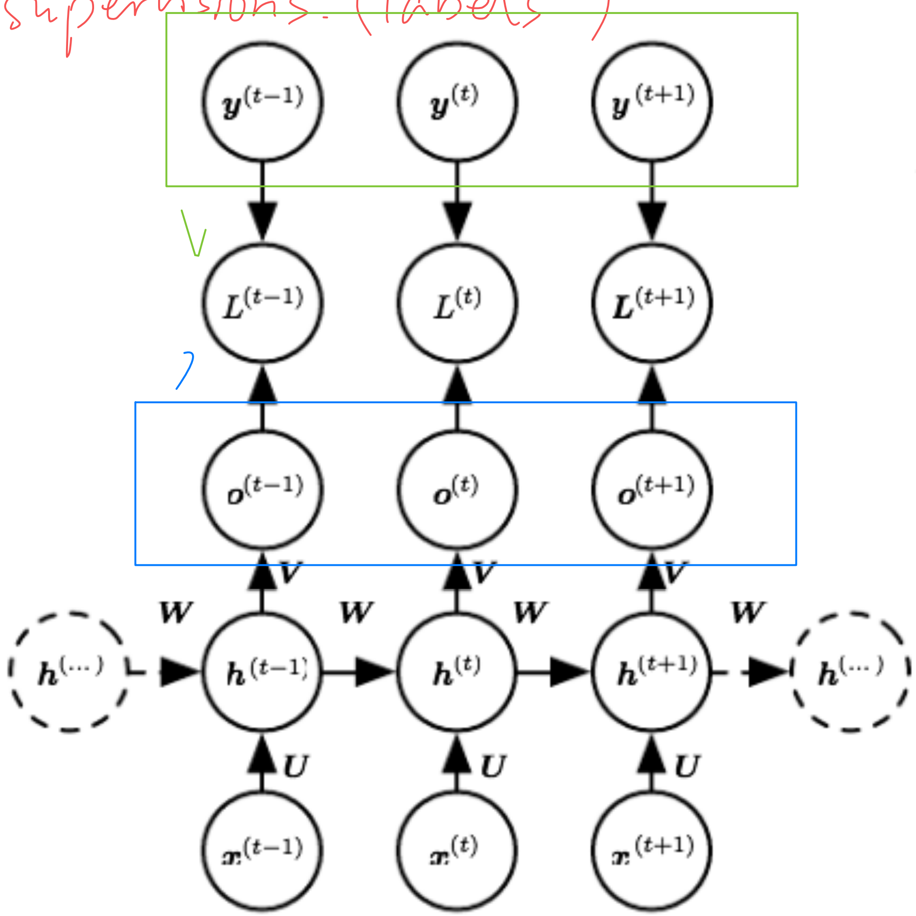
Output sequence
(e.g., POS tags)

Loss function

Output units

Hidden states

Input sequence
(e.g., sentence)




**Training data:**

$\{x^{(1)}, \dots, x^{(\tau)}\}, \{y^{(1)}, \dots, y^{(\tau)}\}$

$\tau$ : /tau/

**Trainable parameters:**

$$\theta = \{U, W, V, b, c\}$$

$V$ : maps from *h* to *o*

$W$ : maps from *h* to *h*

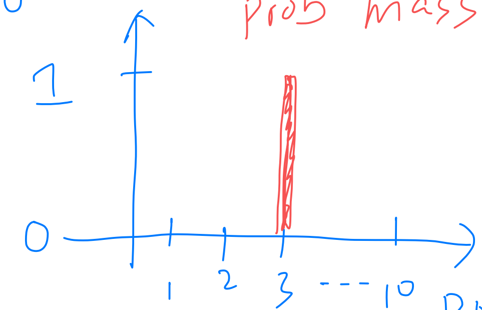
$U$ : maps from *x* to *h*

$b, c$ : biases

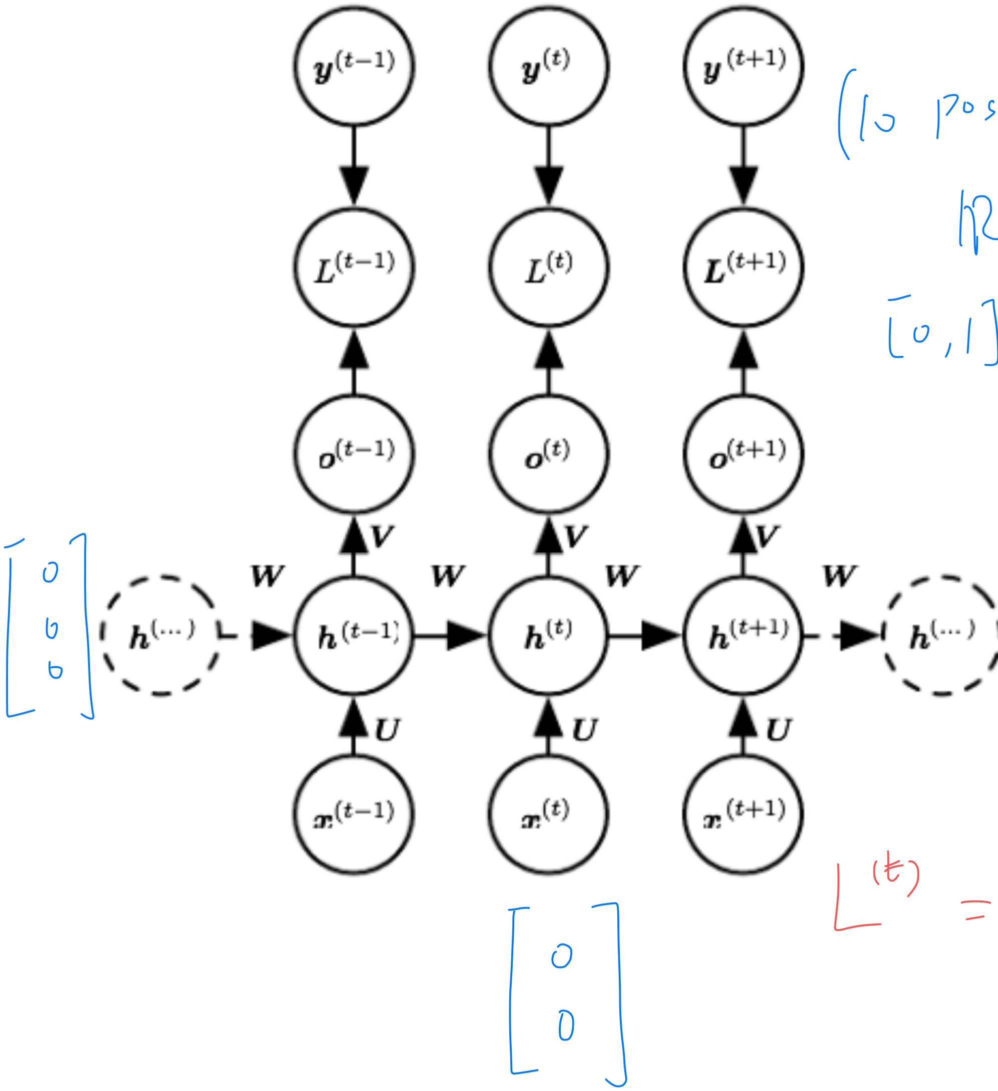
# RNN forward pass

ground Truth $\vec{y}^{(t)}$

prob mass




one hot encoding




(10 pos tags)

$$\boldsymbol{a}^{(t)} = \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)},$$

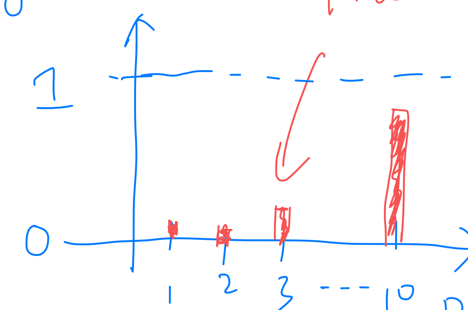
$$\boldsymbol{h}^{(t)} = \tanh(\boldsymbol{a}^{(t)}),$$ activation function

$$\mathbb{R}^{10} \ni \boldsymbol{o}^{(t)} = \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)},$$

$$[0,1]^{10} \ni \hat{\boldsymbol{y}}^{(t)} = \mathrm{softmax}(\boldsymbol{o}^{(t)}),$$ probability distribution

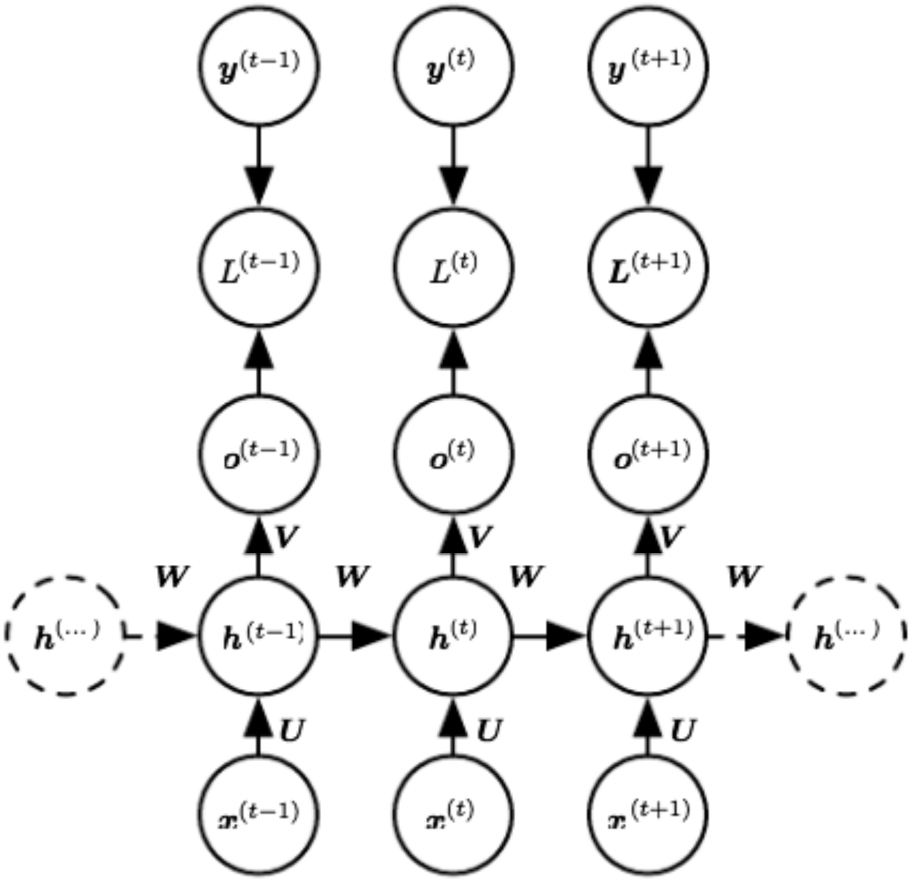
prediction $\hat{\vec{y}}^{(t)}$

$\mathrm{Prob}(y^{(t)} = 3 \mid ---)$




$$L^{(t)} = L(\vec{y}^{(t)}, \hat{\vec{y}}^{(t)}) = -\log \Pr(y^{(t)} = 3 \mid \cdots)$$

# RNN forward pass




Negative log likelihood (NLL) loss,
or the "perplexity"

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

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

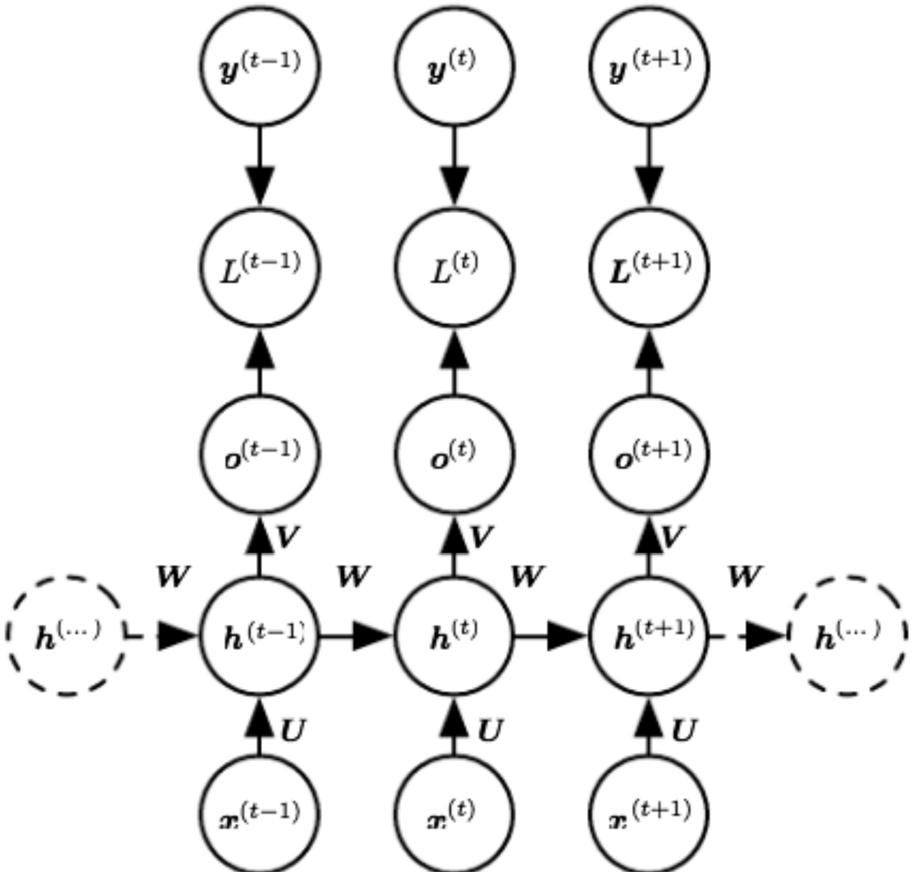
the ground truth label.

$$=-\sum_t \log p_{\text{model}}\left(y^{(t)} \mid \{\boldsymbol{x}^{(1)},\dots,\boldsymbol{x}^{(t)}\}\right)$$

$$\hat{\boldsymbol{y}}^{(t)} = \operatorname{softmax}(\boldsymbol{o}^{(t)})$$

# A running example

# RNN back propagation




time

***BPTT*** (Back Propagation through Time)

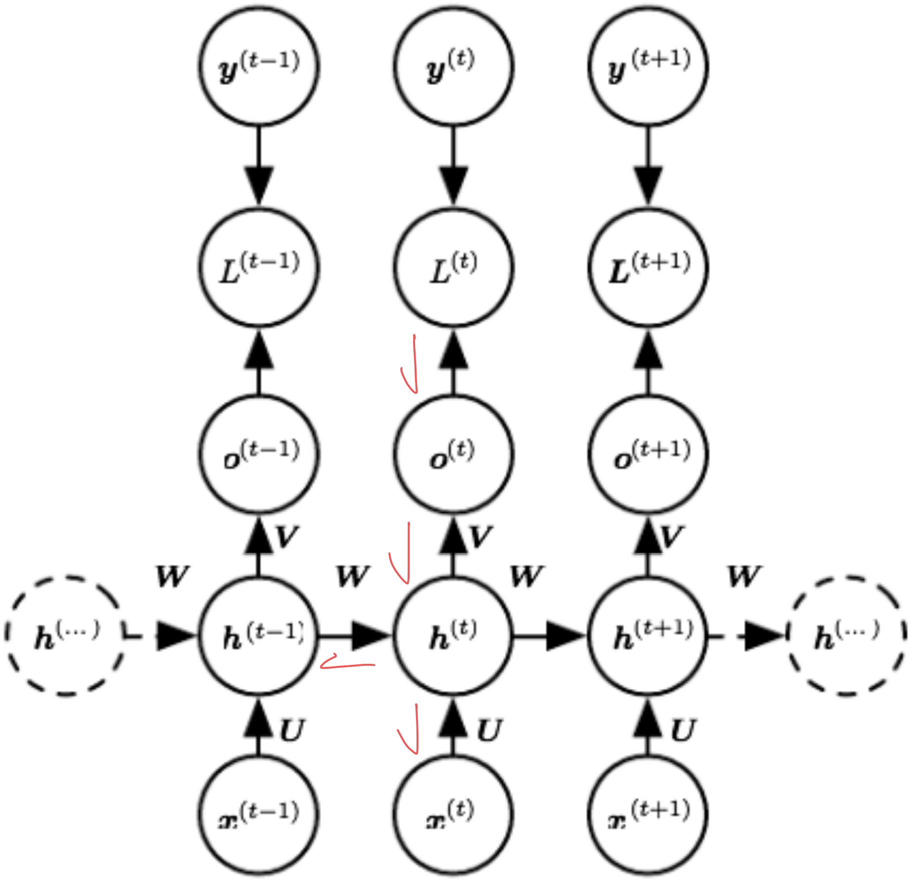
- Used for gradient descent training;
- A special name for RNN back-propagation;
- Need all information in the forward pass, making BPTT sequential and hard to parallelize.
- $\boldsymbol{\theta} = \{\boldsymbol{U}, \boldsymbol{W}, \boldsymbol{V}, \boldsymbol{b}, \boldsymbol{c}\}$ used in all steps.
- Two derivative rules applied:

$$\nabla_x(f(x) + g(x)) = \nabla_x f(x) + \nabla_x g(x)$$

$$\nabla_x(f(g(x)) = \underbrace{\nabla_g f(g(x))}_{\text{Jacobian}} \times \underbrace{\nabla_x g(x)}_{\text{vector}}$$

# RNN back propagation

$L = \sum_{t=1}^{T} L^{(t)}$ = sum of all local losses.




***BPTT*** (Back Propagation through Time)

- focus on each step t (the final gradient is the sum of all gradients at each step.

$$\frac{\partial L}{\partial L^{(t)}} = 1$$

$$(\nabla_{o^{(t)}} L)_i = \frac{\partial L}{\partial o_i^{(t)}} = \frac{\partial L}{\partial L^{(t)}} \frac{\partial L^{(t)}}{\partial o_i^{(t)}} = \underbrace{\hat{y}_i^{(t)} - \mathbf{1}_{i=y^{(t)}}}_{\text{error vector over tags.}}$$

$$\begin{bmatrix} 0.1 \\ 0.2 \\ \vdots \\ 0.01 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

(an element of $\nabla_{o^{(t)}} L$ )

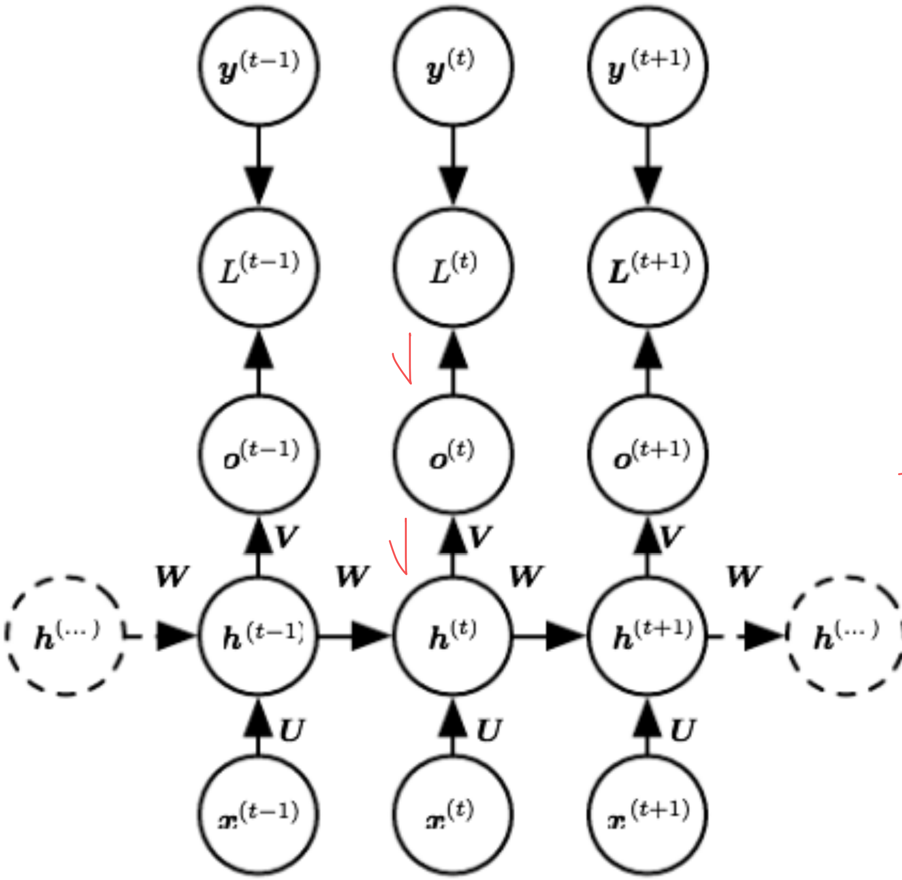
$$L^{(t)} = -\log \Pr(y^{(t)} \mid x^{(1)} \dots x^{(t)}; \theta)$$

$$= -\log \text{softmax}(o^{(t)})$$

$$= -\log \frac{\exp(o^{(t)}_{\hat{y}^{(t)}})}{\sum_{j=1}^{10} \exp(o_j^{(t)})}$$

# RNN backprop (base case)

$$\mathbb{R}^2 \ni \frac{\partial L}{\partial h^{(\tau)}} = \begin{bmatrix} \frac{\partial L}{\partial h_1^{(\tau)}} \\ \frac{\partial L}{\partial h_2^{(\tau)}} \end{bmatrix} = \begin{bmatrix} V_{:1}^T \frac{\partial L}{\partial o^{(\tau)}} \\ V_{:2}^T \frac{\partial L}{\partial o^{(\tau)}} \end{bmatrix} = V^T \frac{\partial L}{\partial o^{(\tau)}}$$



***BPTT*** (Back Propagation through Time)

- Base case: at the final step $\tau$

$$\nabla_{\boldsymbol{h}^{(\tau)}} L = \boldsymbol{V}^\top \nabla_{\boldsymbol{o}^{(\tau)}} L = V^T\left(\hat{y}^{(\tau)} - y^{(\tau)}\right)$$

since $\boldsymbol{o}^{(\tau)} = \boldsymbol{V}\boldsymbol{h}^{(\tau)} + \boldsymbol{c}$

$$\frac{\partial L}{\partial o^{(\tau)}} \qquad \underset{o^{(\tau)}}{\begin{bmatrix} \square \\ \square \\ \vdots \\ \square \end{bmatrix}} = \underset{V}{\begin{bmatrix} \square & \square \\ \vdots & \vdots \\ \square & \square \end{bmatrix}} \underset{h^{(\tau)}}{\begin{bmatrix} \square \\ \square \end{bmatrix}} + \underset{c}{\begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}}$$

$j$ $\quad h_1^{(\tau)}$

$o^{(\tau)} \to L^{(\tau)} \to L$

$$\frac{\partial L}{\partial h_1^{(\tau)}} = \sum_{j=1}^{10} V_{j1} \frac{\partial L}{\partial o_j^{(\tau)}} = V_{:1}^T \frac{\partial L}{\partial o^{(\tau)}}$$

$[\,—\,]$ $\quad [\,|\,]$

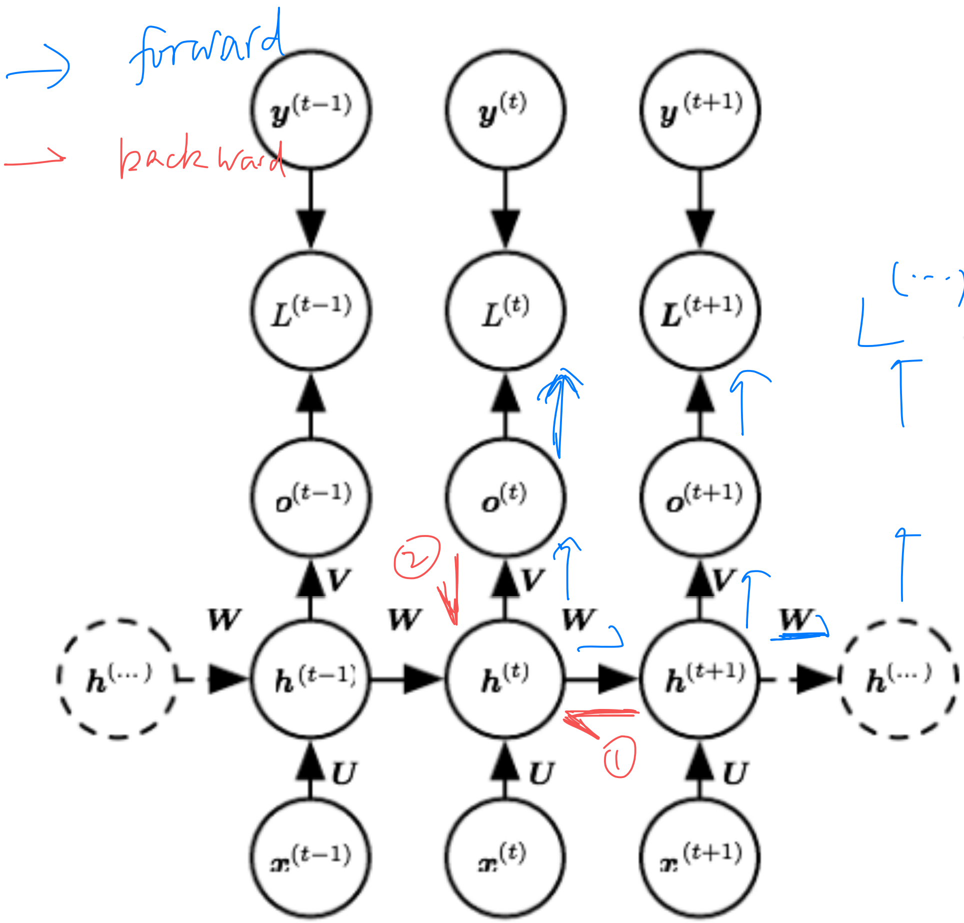
$\vec{o}^{(t)} = \text{softmax}(\vec{c} + V\vec{h}^{(t)})$

$y^{(t)} \to L^{(t)}$ (MLP)

$V, \vec{c}$

$\vec{h}^{(t)}$

# RNN backprop (recurrent)




***BPTT*** (Back Propagation through Time)

- Recursively, at any step $1 \leq t < \tau$

$L^{(\dots)}$

$$\nabla_{h^{(t)}} L = \left(\frac{\partial h^{(t+1)}}{\partial h^{(t)}}\right)^\top (\nabla_{h^{(t+1)}} L) + \left(\frac{\partial o^{(t)}}{\partial h^{(t)}}\right)^\top (\nabla_{o^{(t)}} L)$$

$$= W^\top \text{diag}\left(1 - \left(h^{(t+1)}\right)^2\right) (\nabla_{h^{(t+1)}} L) + \boxed{V^\top (\nabla_{o^{(t)}} L)}$$

$$o^{(t)} = Vh^{(t)} + c$$

$$\frac{\partial L^{(t)}}{\partial o^{(t)}} = \hat{\vec{y}}^{(t)} - \vec{y}^{(t)} \quad \text{(error vector)}$$

$0 \leq t < \tau$

$$\frac{\partial L^{(t)}}{\partial h^{(t)}} = V^\top (\hat{\vec{y}}^{(t)} - \vec{y}^{(t)})$$

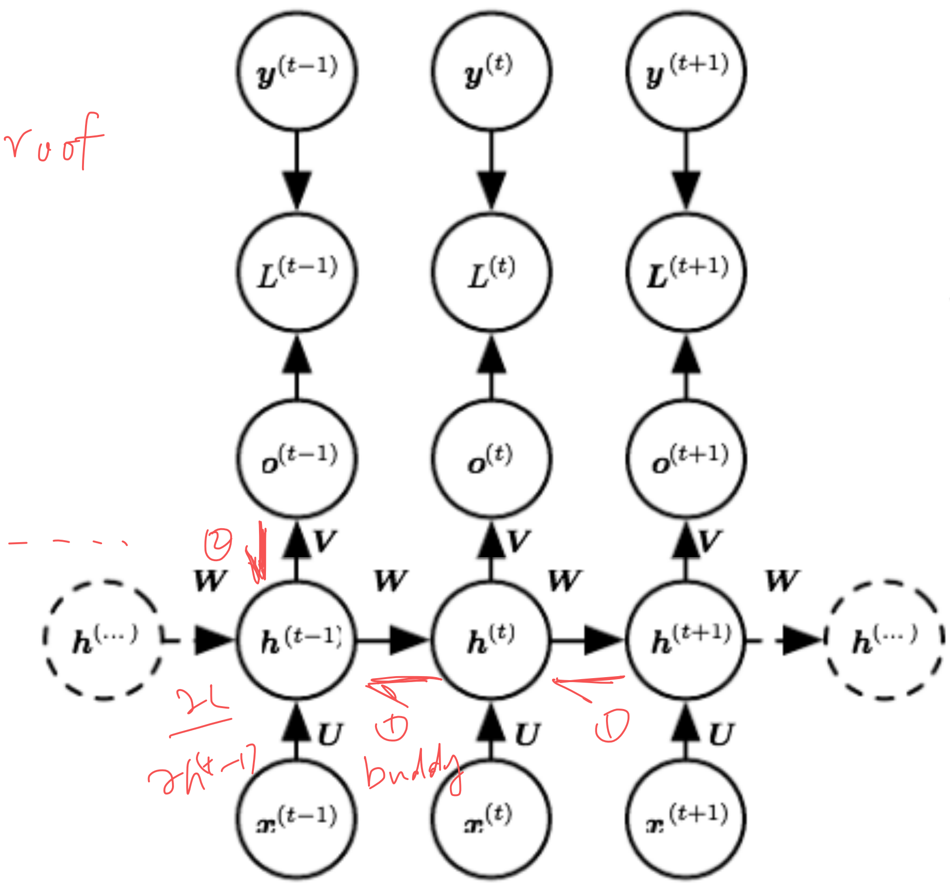
$$\frac{\partial \tanh(x)}{\partial x} = (1-\tanh(x))(1+\tanh(x)) = 1-\tanh^2(x)$$

$$\mathrm{diag}\left(\begin{bmatrix}1\\1\\1\end{bmatrix} - \begin{bmatrix}\tanh^2(h_1^{(t+1)})\\ \tanh^2(h_2^{(t+1)})\\ \tanh^2(h_3^{(t+1)})\end{bmatrix}\right)$$

$$\mathbb{1} - {h^{(t+1)}}^2$$

$$= \begin{bmatrix} 1-\tanh^2(h_1^{(t+1)}) & 0 & 0 \\ 0 & 1-\tanh^2(h_2^{(t+1)}) & \vdots \\ 0 & & 1-\tanh^2 \ldots \end{bmatrix}$$

# RNN backprop (recurrent)




***BPTT*** (Back Propagation through Time)

- Recursively, at any step $1 \leq t < \tau$

for $t = \tau - 1, \ldots, 1$

$$\nabla_{h^{(t)}} L = \left(\frac{\partial h^{(t+1)}}{\partial h^{(t)}}\right)^{\top} (\nabla_{h^{(t+1)}} L) + \left(\frac{\partial o^{(t)}}{\partial h^{(t)}}\right)^{\top} (\nabla_{o^{(t)}} L)$$

$$= \boxed{W^{\top} \mathrm{diag}\left(1 - \left(h^{(t+1)}\right)^2\right) (\nabla_{h^{(t+1)}} L)} + V^{\top} (\nabla_{o^{(t)}} L)$$

$$\begin{bmatrix}0\\0\\0\end{bmatrix} \quad h^{(t+1)} = \tanh(a^{(t+1)}) \qquad \text{(element-wise)}$$

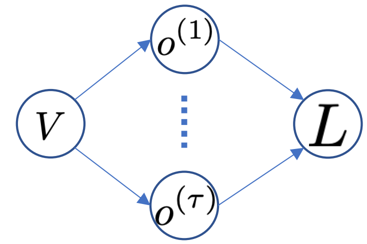
$$h_i^{(t+1)} = \tanh(a_i^{(t+1)})$$

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

$$a^{(t+1)} = Wh^{(t)} + Ux^{(t+1)} + b$$

$$\begin{bmatrix}0\\0\\0\end{bmatrix} = \begin{bmatrix}0&0&0\\0&0&0\\0&0&0\end{bmatrix}\begin{bmatrix}0\\0\\0\end{bmatrix} + \ldots$$

# RNN backprop (params)




***BPTT*** (Back Propagation through Time)

- at any step $1 \leq t < \tau$

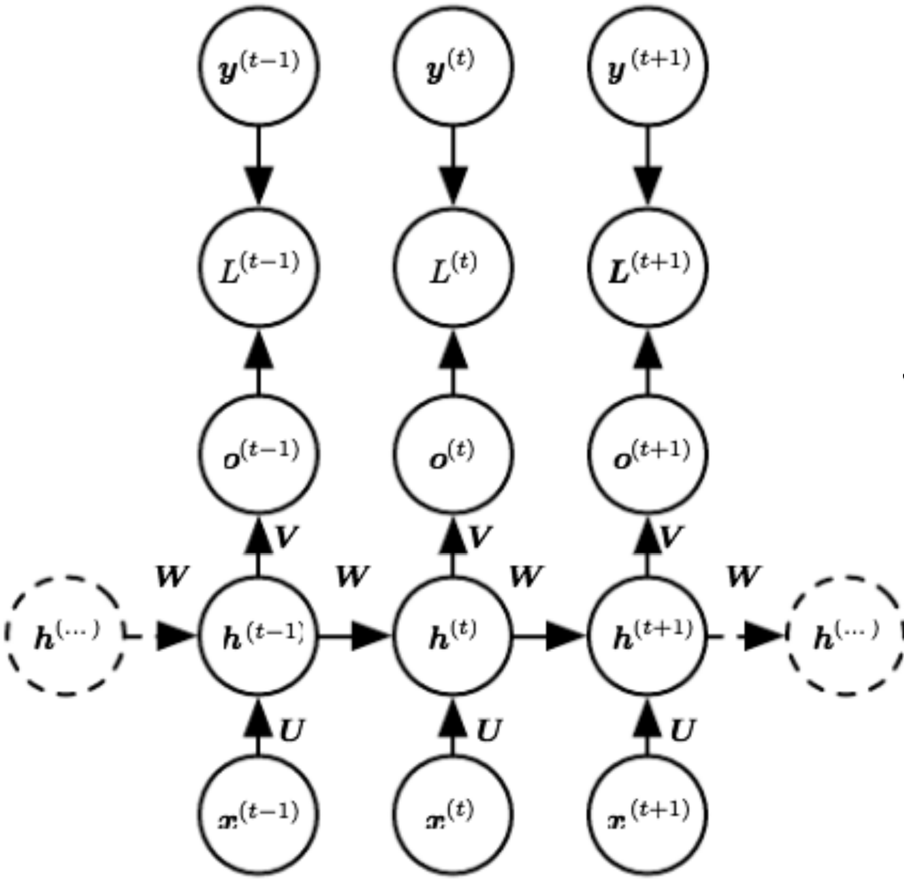
since $\boldsymbol{o}^{(t)} = V\boldsymbol{h}^{(t)} + \boldsymbol{c}$

$$\nabla_{\boldsymbol{V}} L^{(t)} = (\nabla_{\boldsymbol{o}^{(t)}} L^{(t)}) \boldsymbol{h}^{(t)\top} \qquad \nabla_{\boldsymbol{c}} L^{(t)} = \nabla_{\boldsymbol{o}^{(t)}} L^{(t)}$$

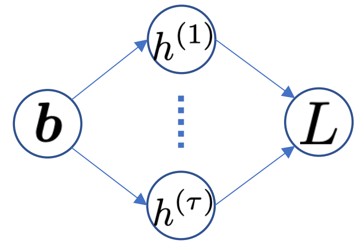
accumulate over time

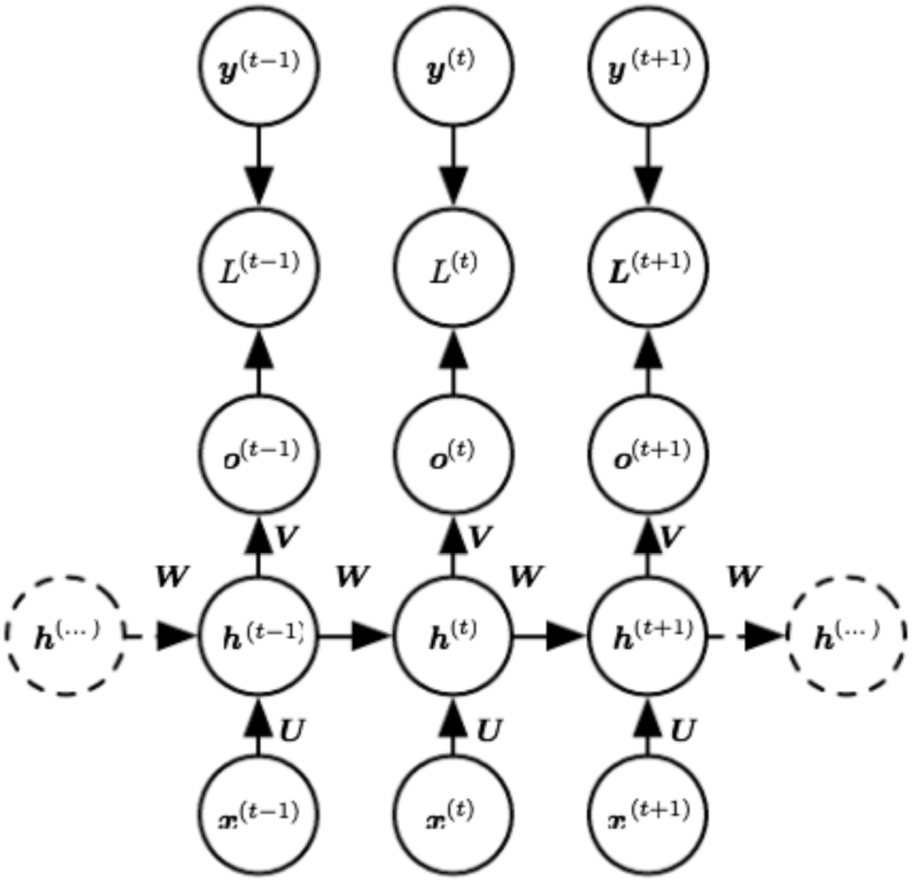
$$\nabla_{\boldsymbol{V}} L = \sum_t \sum_i \left( \frac{\partial L}{\partial o_i^{(t)}} \right) \nabla_{\boldsymbol{V}^{(t)}} o_i^{(t)} = \sum_t (\nabla_{\boldsymbol{o}^{(t)}} L) \, \boldsymbol{h}^{(t)\top}$$

$$\nabla_{\boldsymbol{c}} L = \sum_t \left( \frac{\partial \boldsymbol{o}^{(t)}}{\partial \boldsymbol{c}} \right)^\top \nabla_{\boldsymbol{o}^{(t)}} L = \sum_t \nabla_{\boldsymbol{o}^{(t)}} L$$



# RNN backprop (params)




***BPTT*** (Back Propagation through Time)

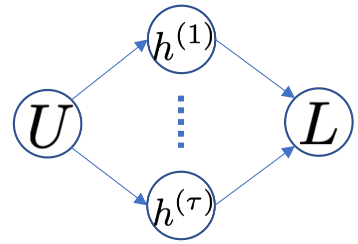
- at any step $1 \leq t < \tau$

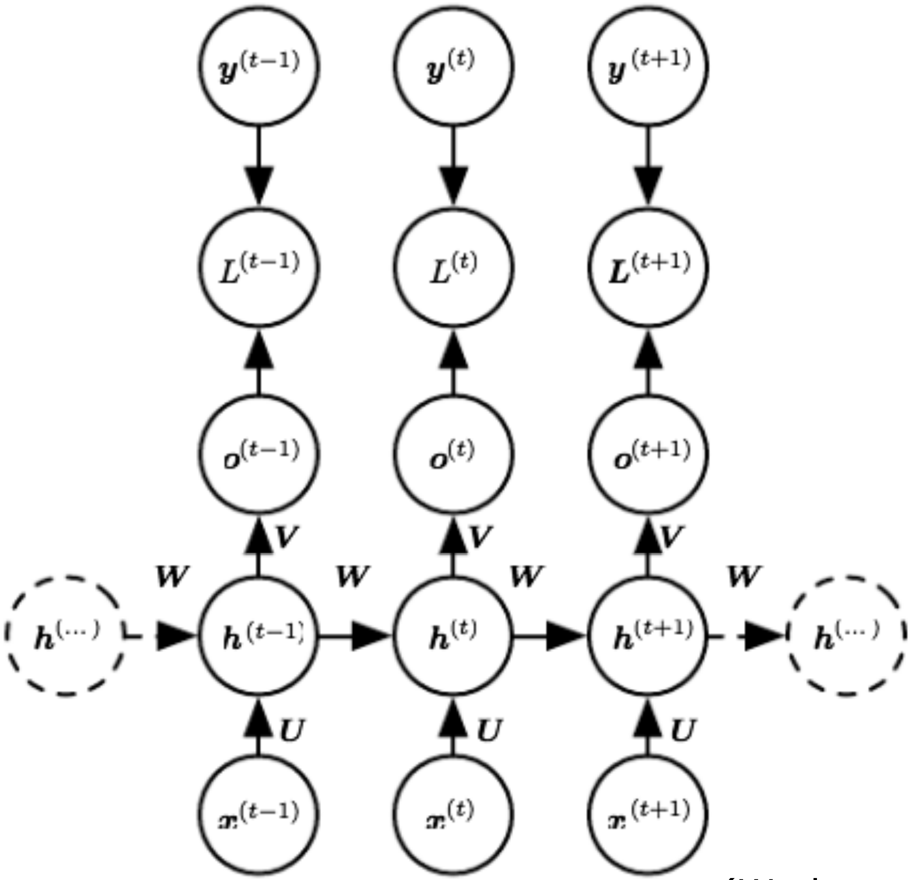
since $\boldsymbol{h}^{(t)} = \tanh(\boldsymbol{a}^{(t)})$

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

$$\nabla_{\boldsymbol{b}} L = \sum_t \left(\frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{b}^{(t)}}\right)^{\top} \nabla_{\boldsymbol{h}^{(t)}} L = \sum_t \mathrm{diag}\left(1 - \left(\boldsymbol{h}^{(t)}\right)^2\right) \nabla_{\boldsymbol{h}^{(t)}} L$$

# RNN backprop (params)




***BPTT*** (Back Propagation through Time)

- at any step $1 \leq t < \tau$

since $\boldsymbol{h}^{(t)} = \tanh(\boldsymbol{a}^{(t)})$

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

$$\nabla_{\boldsymbol{U}} L = \sum_t \mathrm{diag}\left(1 - \left(\boldsymbol{h}^{(t)}\right)^2\right)(\nabla_{\boldsymbol{h}^{(t)}} L)\, \boldsymbol{x}^{(t)\top}$$

(We have done $\nabla_{\boldsymbol{h}^{(t-1)}} L$ , and leave $\nabla_{\boldsymbol{W}} L$ as an exercise.)