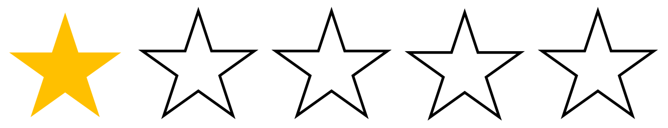
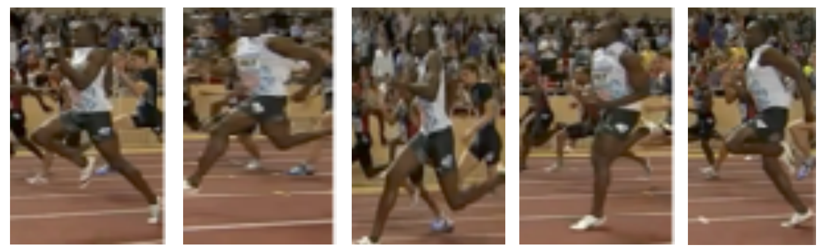
Recurrent Neural Nets

Introduction to NLP

Rahul Mishra
IIIT-Hyderabad
March 05, 2024

Sequence Data

Speech recognition		"The quick brown fox jumped over the lazy dog."
Music generation	∅	
Sentiment classification	"There is nothing to like in this movie."	
DNA sequence analysis	AGCCCCTGTGAGGAACTAG	AGCCCCTGTGAGGAACTAG
Machine translation	Voulez-vous chanter avec moi?	Do you want to sing with me?
Video activity recognition		Running
Name entity recognition	Yesterday, Harry Potter met Hermione Granger.	Yesterday, Harry Potter met Hermione Granger.

Motivating example

NLP

x: Harry Potter and Hermione Granger invented a new spell.

$\rightarrow x^{<1>}$ $x^{<2>}$ $x^{<3>}$ . . . . . $x^{<t>}$ . . . . $x^{<9>}$

$T_x = 9$

$\rightarrow$ y: 1 1 0 1 1 0 0 0 0

$y^{<1>}$ $y^{<2>}$ $y^{<3>}$ . . . . $y^{<9>}$

$T_y = 9$

$X^{(i)<t>}$ $T_x^{(i)} = 9$ 15

$y^{(i)<t>}$ $T_y^{(i)}$

Representing Words

$x^{<t>}$ (x, y)

$x \rightarrow y$

x: Harry Potter and Hermione Granger invented a new spell.

$x^{<1>}$ $x^{<2>}$ $x^{<3>}$ … $x^{<7>}$ $x^{<9>}$

Vocabulary

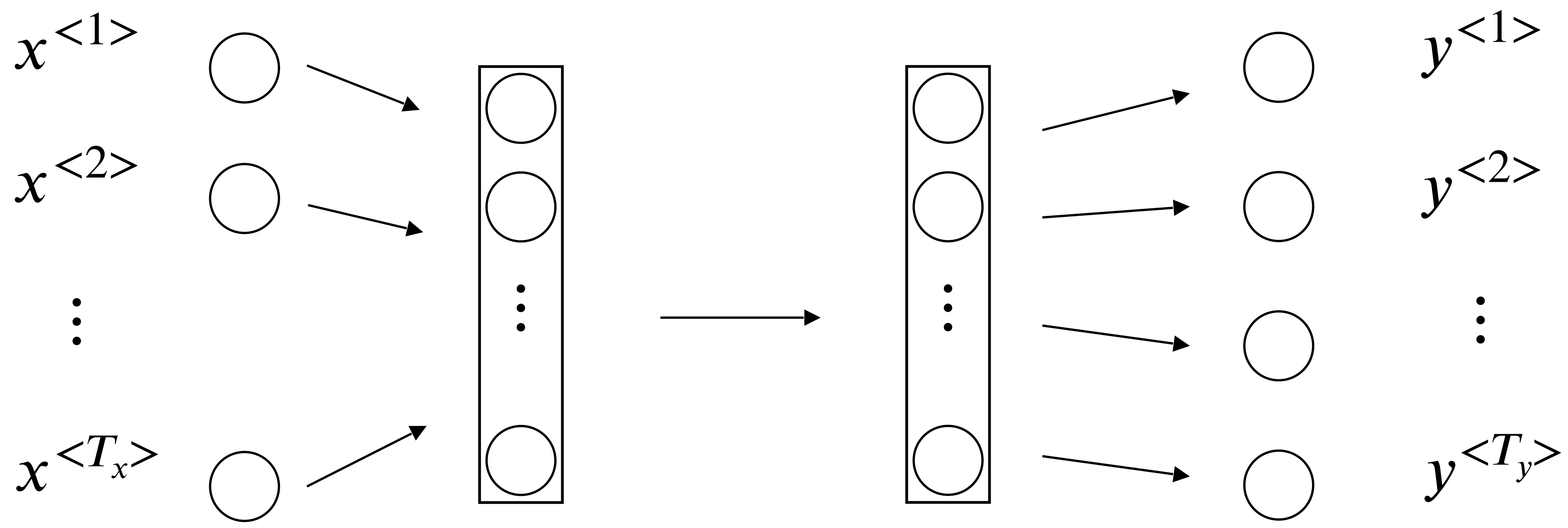
$$\begin{bmatrix} \text{a} \\ \text{aaron} \\ \vdots \\ \text{and} \\ \vdots \\ \text{harry} \\ \vdots \\ \text{potter} \\ \vdots \\ \text{zulu} \end{bmatrix} \begin{matrix} 1 \leftarrow \\ 2 \\ \vdots \\ 367 \leftarrow \\ \vdots \\ 4075 \\ \vdots \\ 6830 \\ \vdots \\ 10{,}000 \end{matrix}$$

<UNK> 10,000

$$\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{matrix} \\ \\ \\ \\ \leftarrow 4075 \\ \\ \\ \\ \end{matrix} \quad \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix} \begin{matrix} \\ \\ \\ \leftarrow 6830 \\ \\ \\ \end{matrix} \quad \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \begin{matrix} \\ \\ \leftarrow 367 \\ \\ \\ \end{matrix} \quad 10{,}000 \quad \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

One-hot

Why not a standard network?



Problems:

- Inputs, outputs can be different lengths in different examples.

- Doesn’t share features learned across different positions of text.

RNNs

RNN **Advantages**:

- Can process any length input
- Computation for step *t* can (in theory) use information from many steps back
- Model size doesn't increase for longer input context
- Same weights applied on every timestep, so there is symmetry in how inputs are processed.

RNN **Disadvantages**:

- Recurrent computation is slow
- In practice, difficult to access information from many steps back

output distribution

$$\hat{\boldsymbol{y}}^{(t)} = \mathrm{softmax}\left(\boldsymbol{U}\boldsymbol{h}^{(t)} + \boldsymbol{b}_2\right) \in \mathbb{R}^{|V|}$$

hidden states

$$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_e\boldsymbol{e}^{(t)} + \boldsymbol{b}_1\right)$$

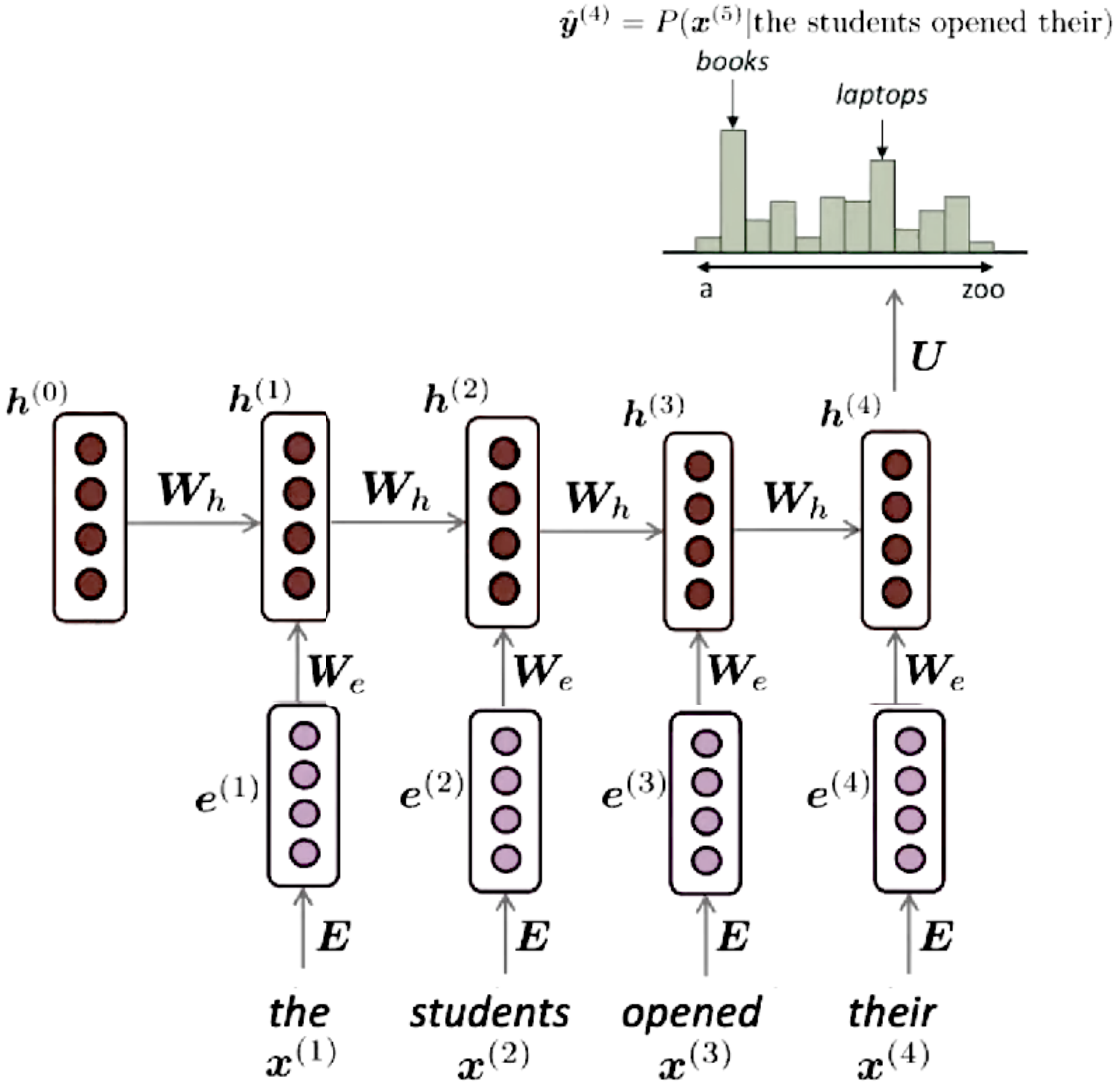
$\boldsymbol{h}^{(0)}$ is the initial hidden state

word embeddings

$$\boldsymbol{e}^{(t)} = \boldsymbol{E}\boldsymbol{x}^{(t)}$$

words / one-hot vectors

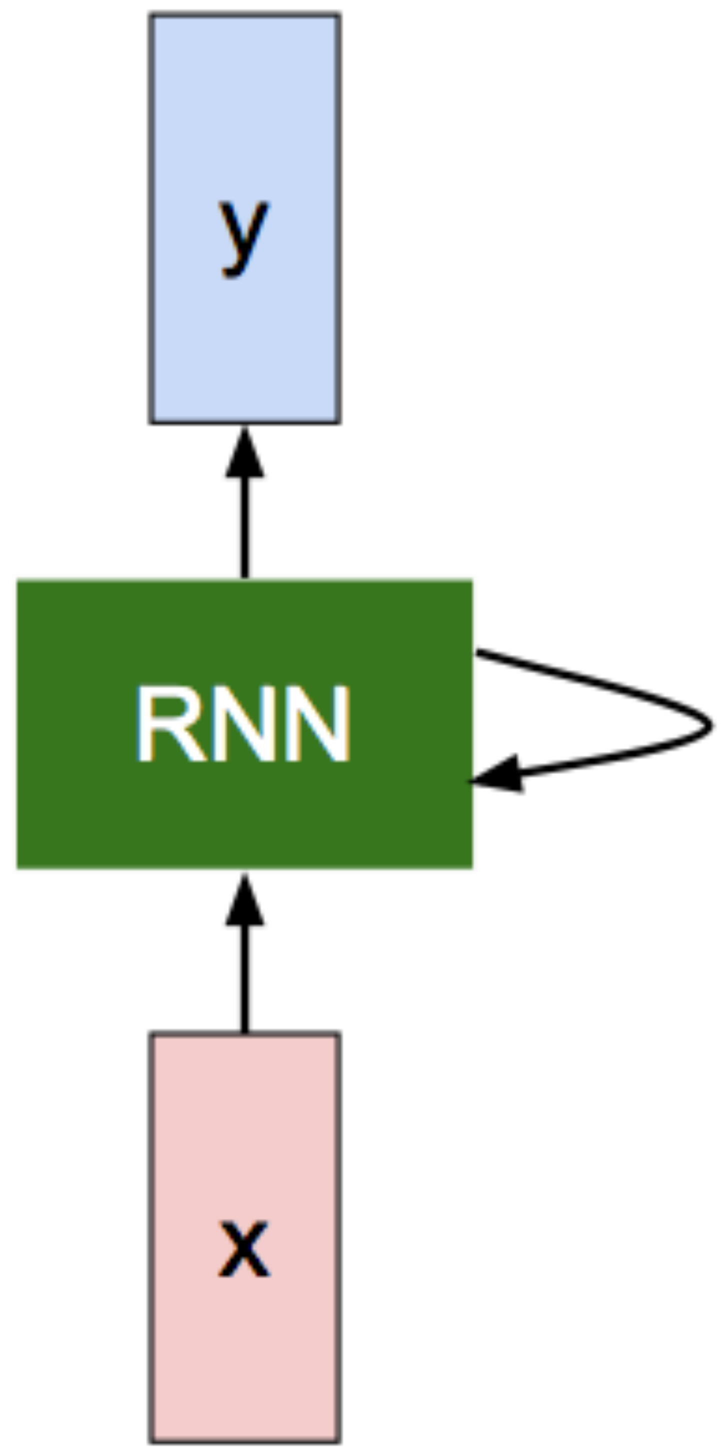
$$\boldsymbol{x}^{(t)} \in \mathbb{R}^{|V|}$$



RNN

- Idea: We can process a sequence of vectors **x** by applying a recurrence formula at every time step t

- Note: Same parameters, weights and function used at each step.



Training RNN

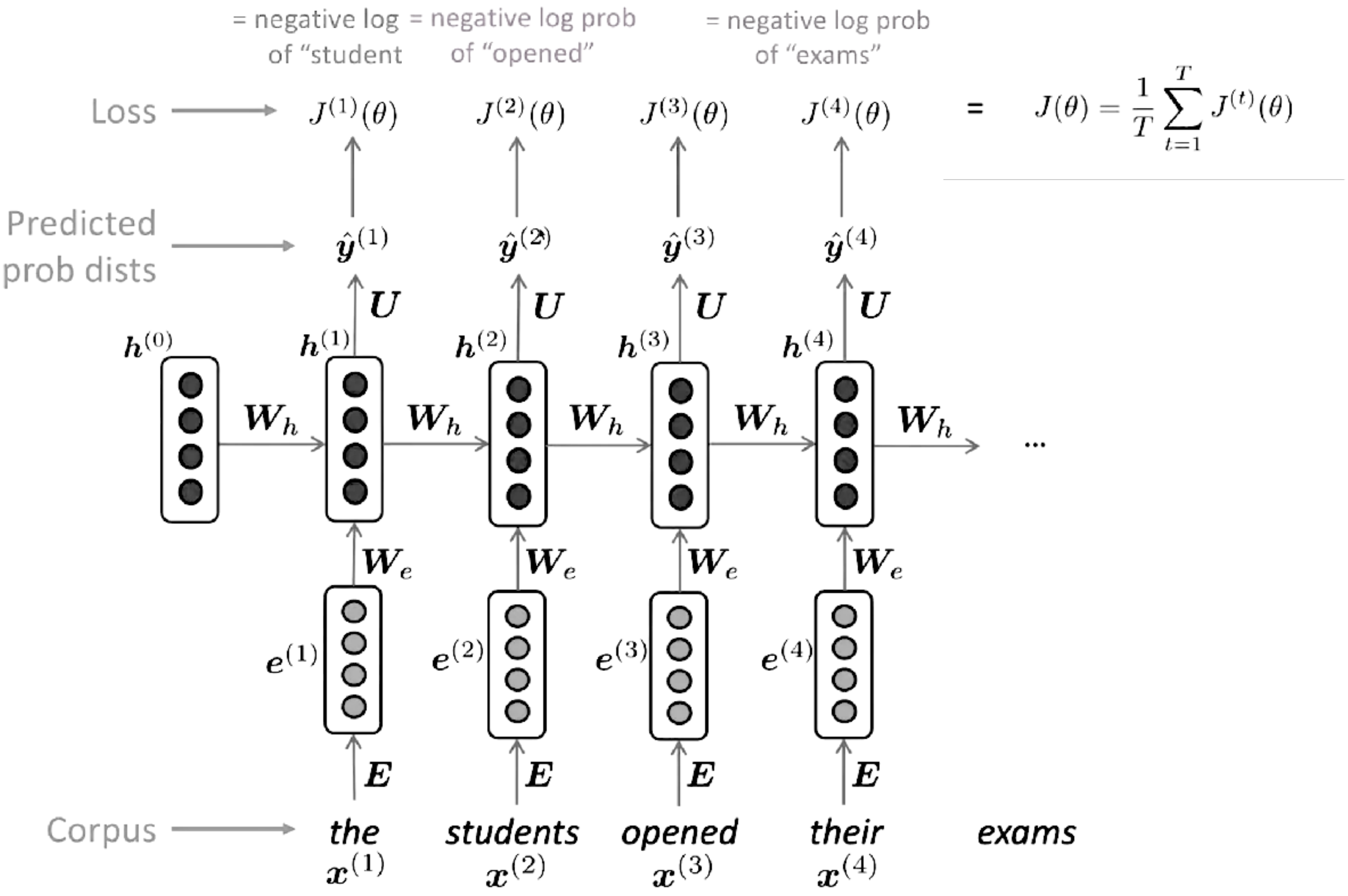
Loss function on step *t* is cross-entropy between predicted probability distribution $\hat{\boldsymbol{y}}^{(t)}$, and the true next word $\boldsymbol{y}^{(t)}$ (one-hot for $\boldsymbol{x}^{(t+1)}$):

$$J^{(t)}(\theta) = CE(\boldsymbol{y}^{(t)}, \hat{\boldsymbol{y}}^{(t)}) = -\sum_{w \in V} \boldsymbol{y}_w^{(t)} \log \hat{\boldsymbol{y}}_w^{(t)} = -\log \hat{\boldsymbol{y}}_{\boldsymbol{x}_{t+1}}^{(t)}$$

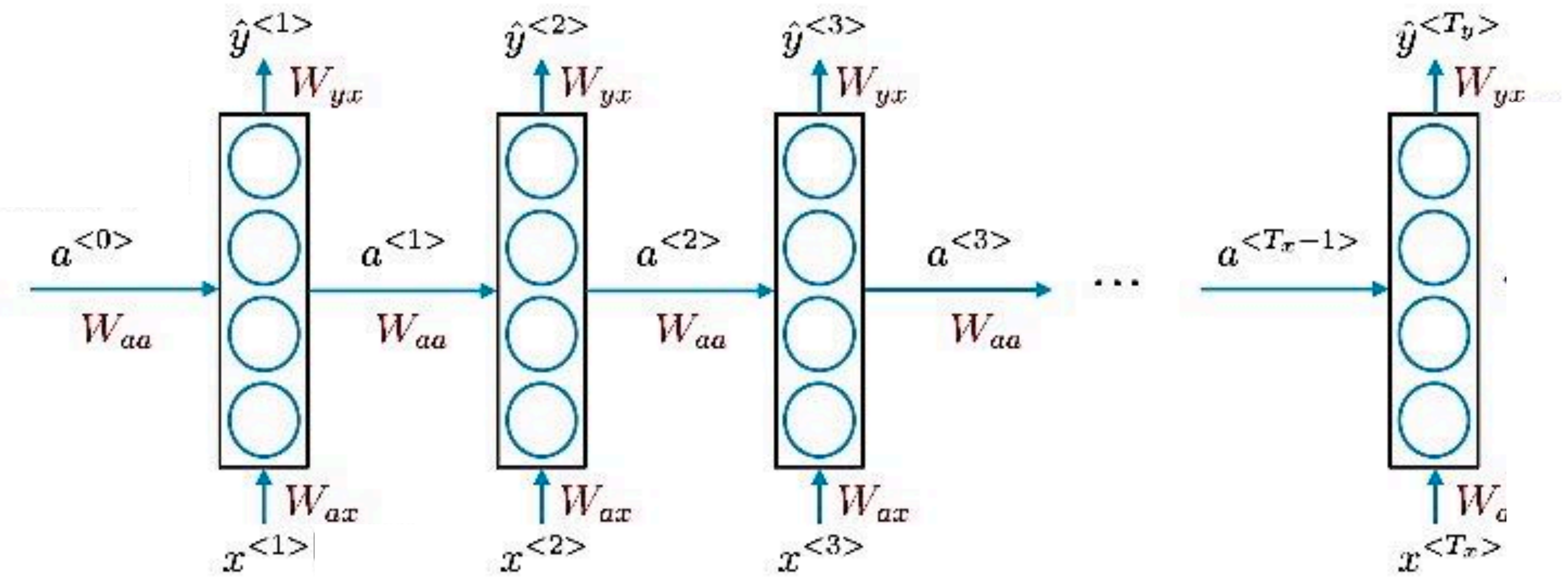
Average this to get overall loss

$$J(\theta) = \frac{1}{T} \sum_{t=1}^{T} J^{(t)}(\theta) = \frac{1}{T} \sum_{t=1}^{T} -\log \hat{\boldsymbol{y}}_{\boldsymbol{x}_{t+1}}^{(t)}$$

Training RNN



Forward Propagation



$$a^{<0>} = 0$$

$$a^{<1>} = g_1\left(W_{aa}a^{<0>} + W_{ax}x^{<1>} + b_a\right)$$

$g_1() \leftarrow$ tanh /ReLU

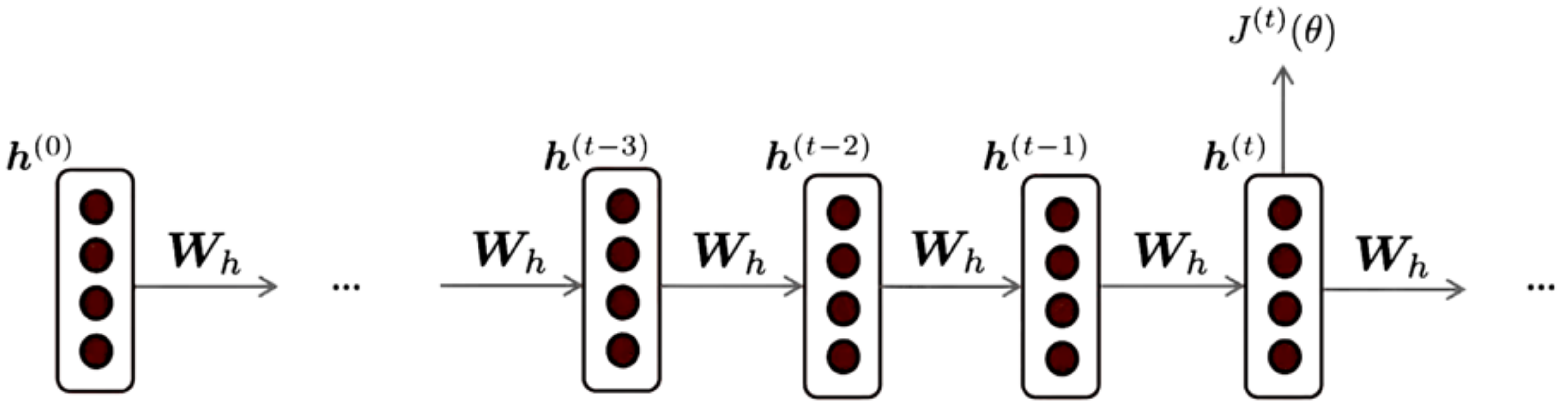
$$\hat{y}^{<1>} = g_2\left(W_{ya}a^{<1>} + b_y\right)$$

$g_2() \leftarrow$ Sigmoid/Softmax/Linear

$$a^{<t>} = g_1\left(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a\right)$$

$$\hat{y}^{<t>} = g_2\left(W_{ya}a^{<t>} + b_y\right)$$

Training RNN: Backdrop in RNN

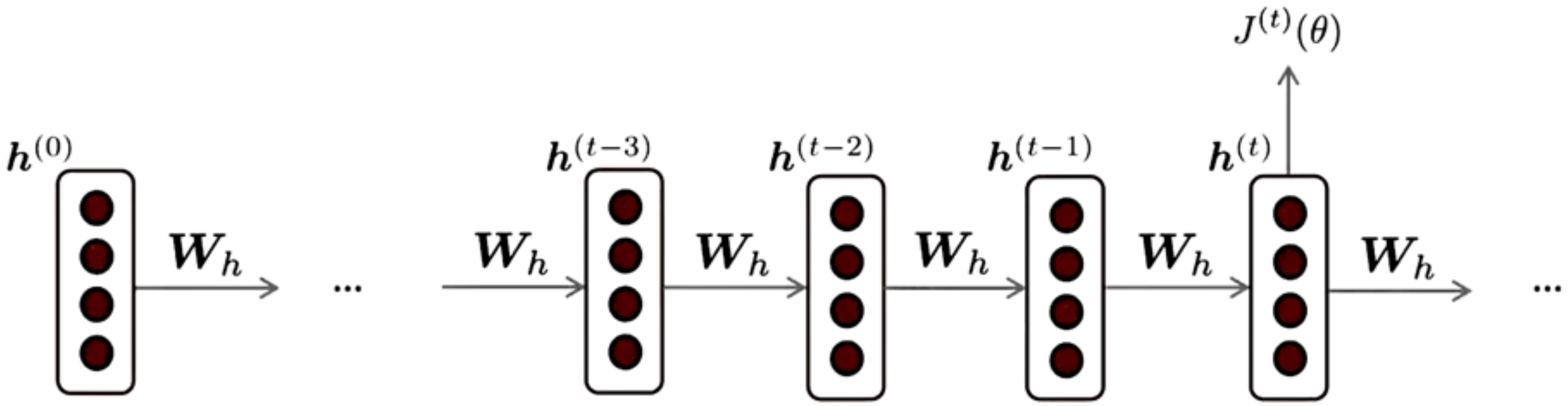


<u>Question:</u> What's the derivative of $J^{(t)}(\theta)$ w.r.t. the repeated weight matrix $\boldsymbol{W}_h$?

<u>Answer:</u> $$\frac{\partial J^{(t)}}{\partial \boldsymbol{W_h}} = \sum_{i=1}^{t} \frac{\partial J^{(t)}}{\partial \boldsymbol{W_h}}\bigg|_{(i)}$$

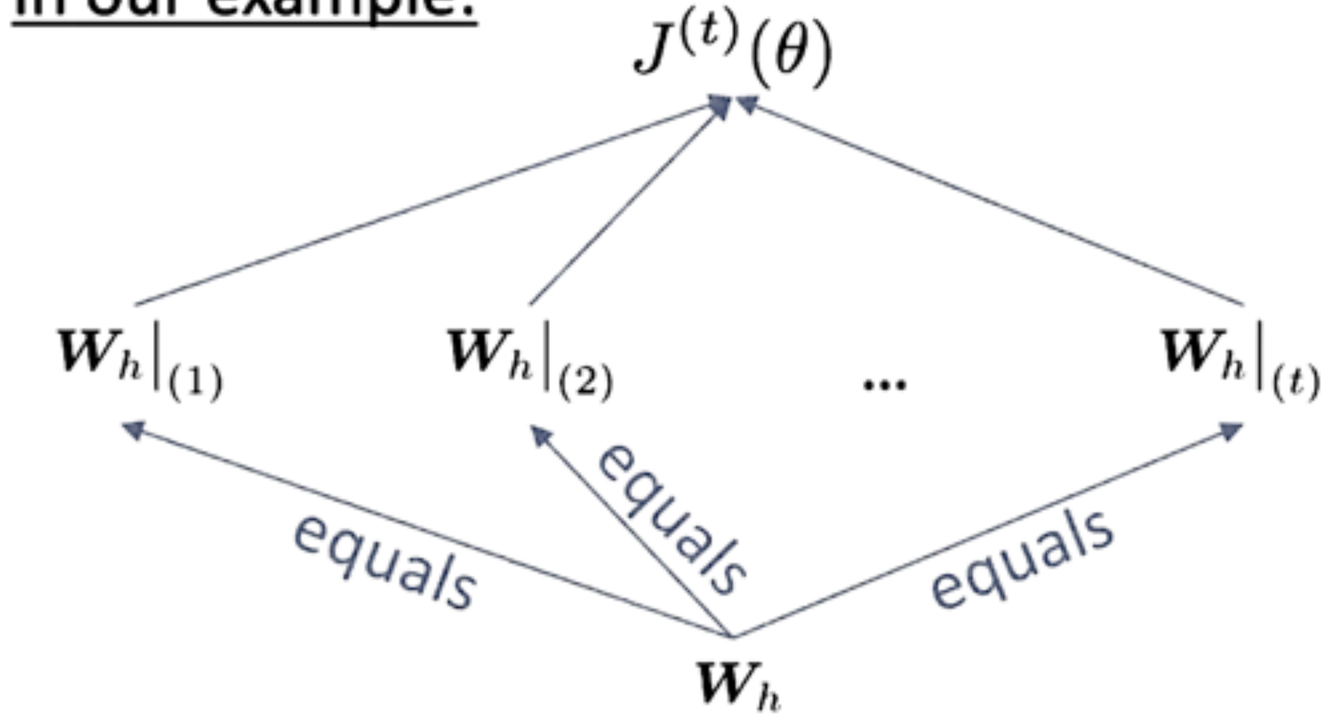
"The gradient w.r.t. a repeated weight is the sum of the gradient w.r.t. each time it appears"

Training RNN: Backdrop in RNN



Question: What's the derivative of $J^{(t)}(\theta)$ w.r.t. the repeated weight matrix $\boldsymbol{W}_h$?

In our example:

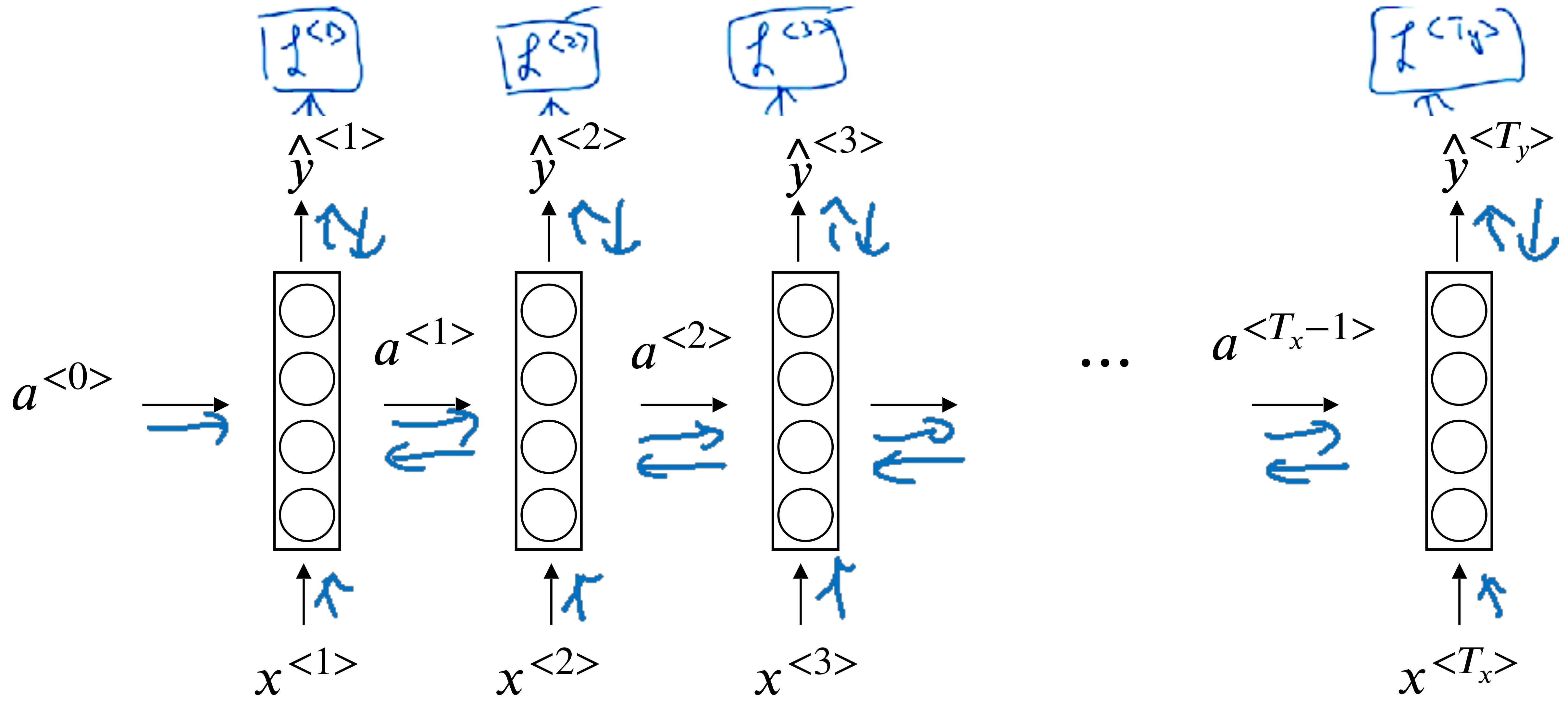


Apply the multivariable chain rule:

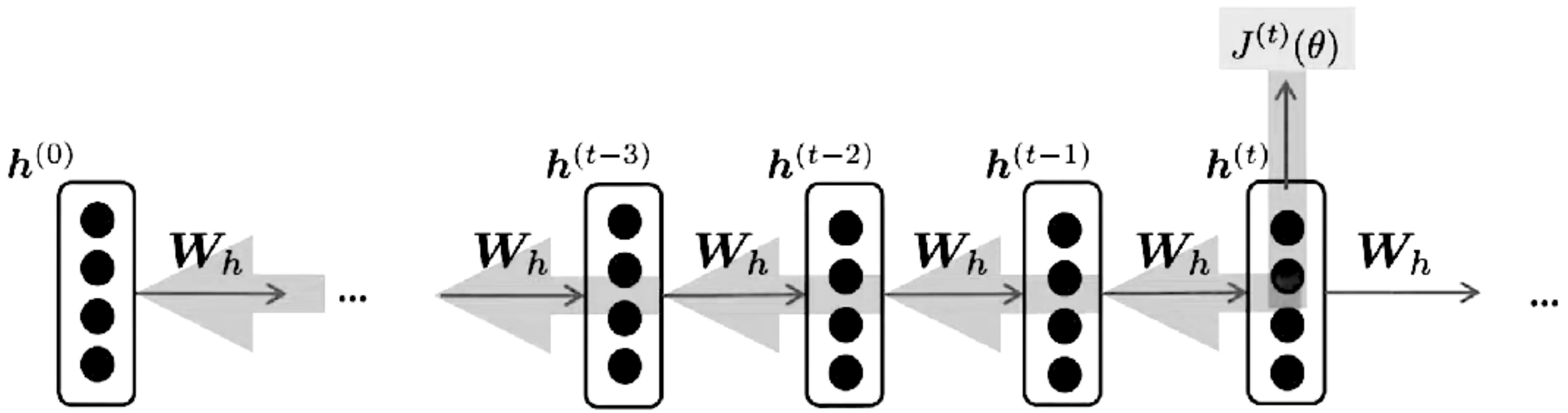
$$\frac{\partial J^{(t)}}{\partial \boldsymbol{W}_h} = \sum_{i=1}^{t} \frac{\partial J^{(t)}}{\partial \boldsymbol{W}_h}\bigg|_{(i)} \overbrace{\boxed{\frac{\partial \boldsymbol{W}_h|_{(i)}}{\partial \boldsymbol{W}_h}}}^{=1}$$

$$= \sum_{i=1}^{t} \frac{\partial J^{(t)}}{\partial \boldsymbol{W}_h}\bigg|_{(i)}$$

Backpropagation Through Time



Backpropagation Through Time

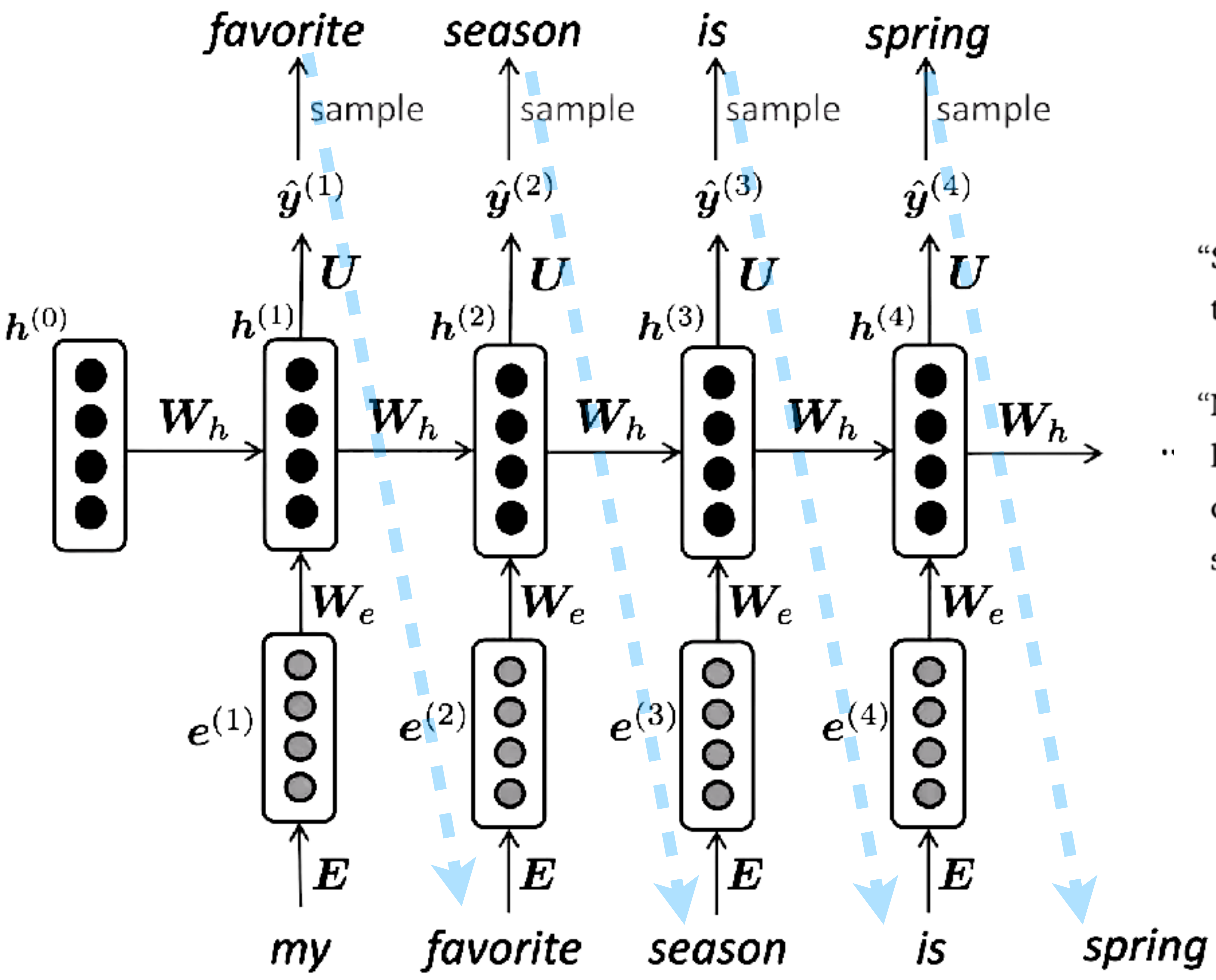


$$\frac{\partial J^{(t)}}{\partial \boldsymbol{W_h}} = \boxed{\sum_{i=1}^{t} \left.\frac{\partial J^{(t)}}{\partial \boldsymbol{W_h}}\right|_{(i)}}$$

Question: How do we calculate this?

Answer: Backpropagate over timesteps *i=t,…,0*, summing gradients as you go. This algorithm is called **"backpropagation through time"** [Werbos, P.G., 1988, *Neural Networks* **1**, and others]

Generating Text using RNNs

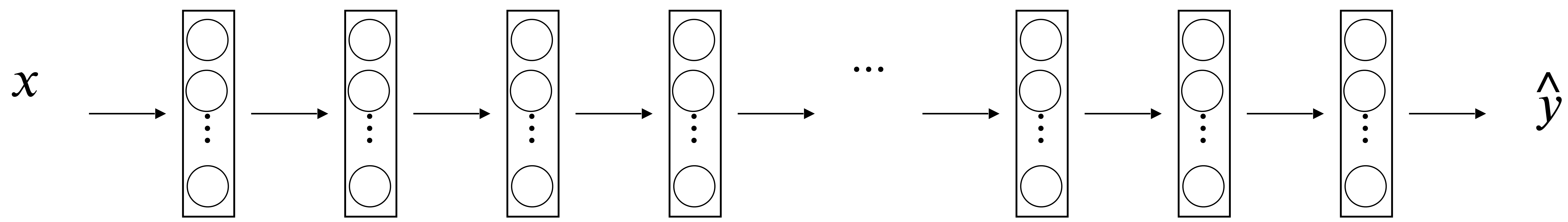


"Sorry," Harry shouted, panicking—"I'll leave those brooms in London, are they?"

"No idea," said Nearly Headless Nick, casting low close by Cedric, carrying the last bit of treacle Charms, from Harry's shoulder, and to answer him the common room perched upon it, four arms held a shining knob from when the spider hadn't felt it seemed. He reached the teams too.

Source: https://medium.com/deep-writing/harry-potter-written-by-artificial-intelligence-8a9431803da6

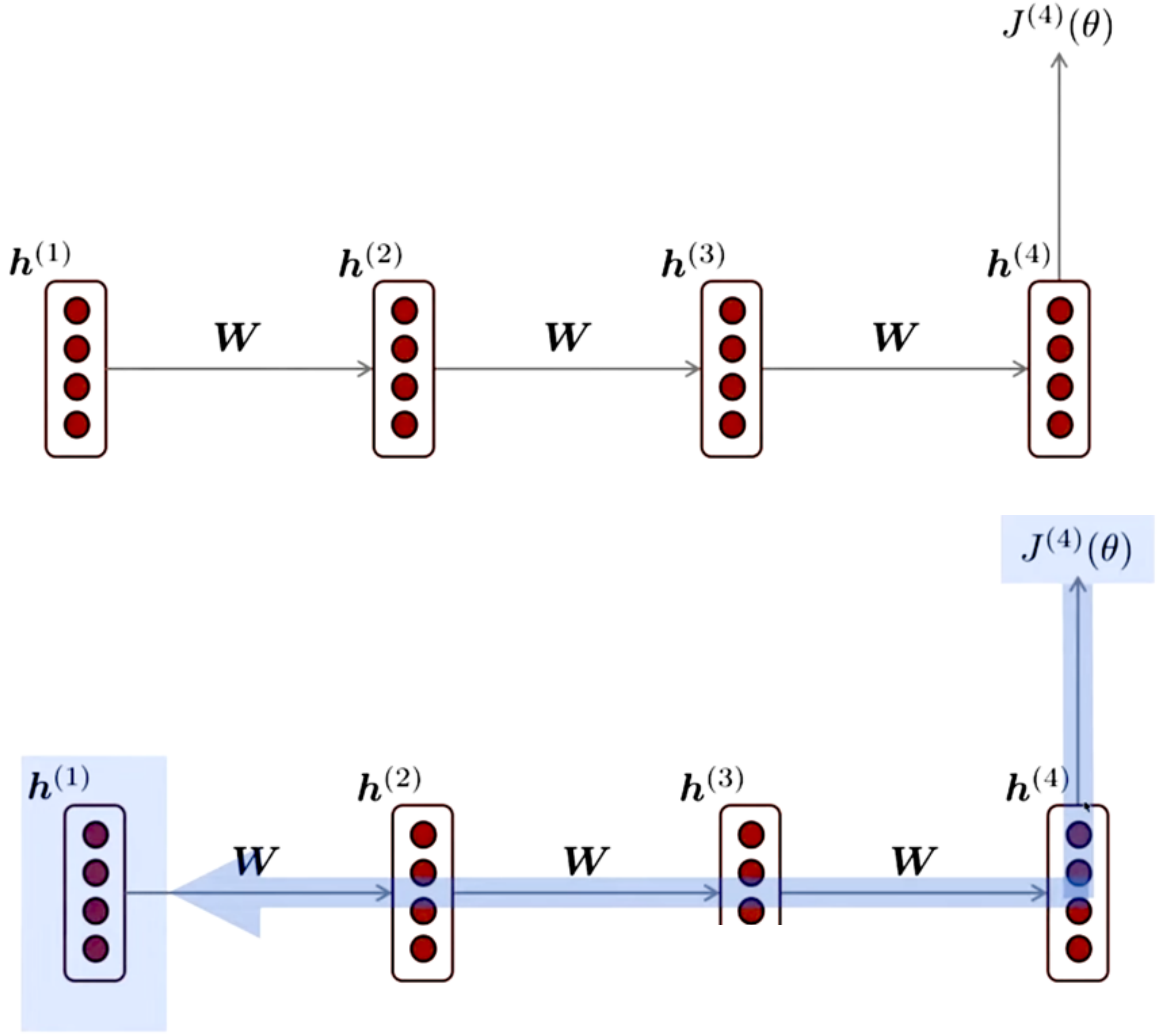
Vanishing and Exploding Gradient



The cat, that already ate.........., was full

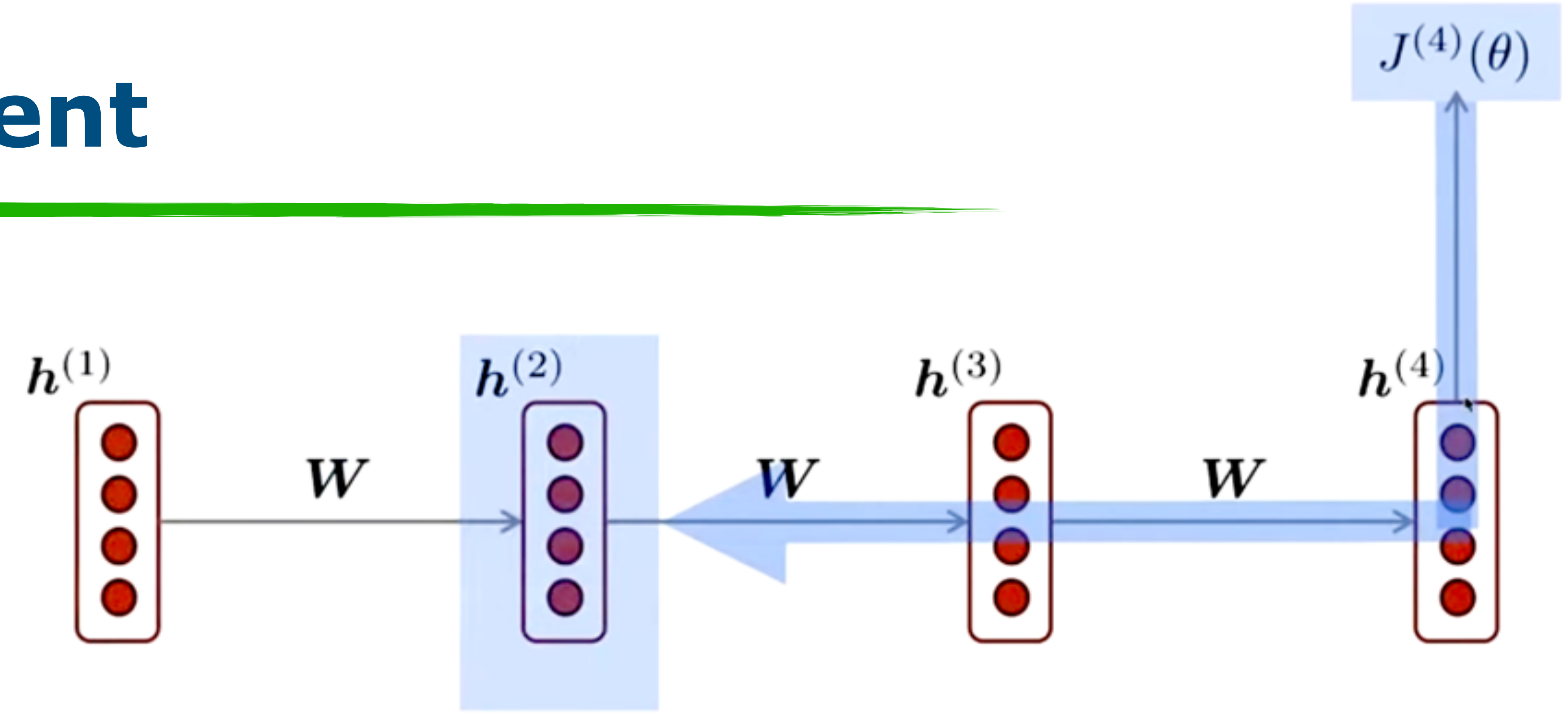
The cats, that already ate.........., were full

Vanishing Gradient



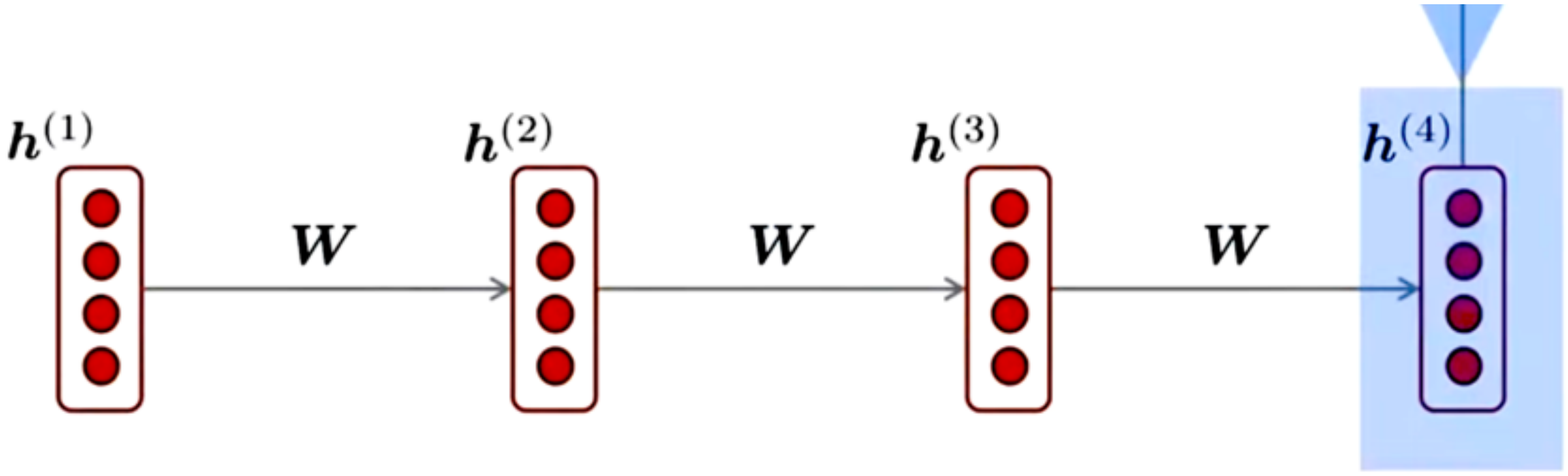
$$\frac{\partial J^{(4)}}{\partial h^{(1)}} = ?$$

Vanishing Gradient



$$\frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(1)}} = \frac{\partial \boldsymbol{h}^{(2)}}{\partial \boldsymbol{h}^{(1)}} \times \frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(2)}}$$

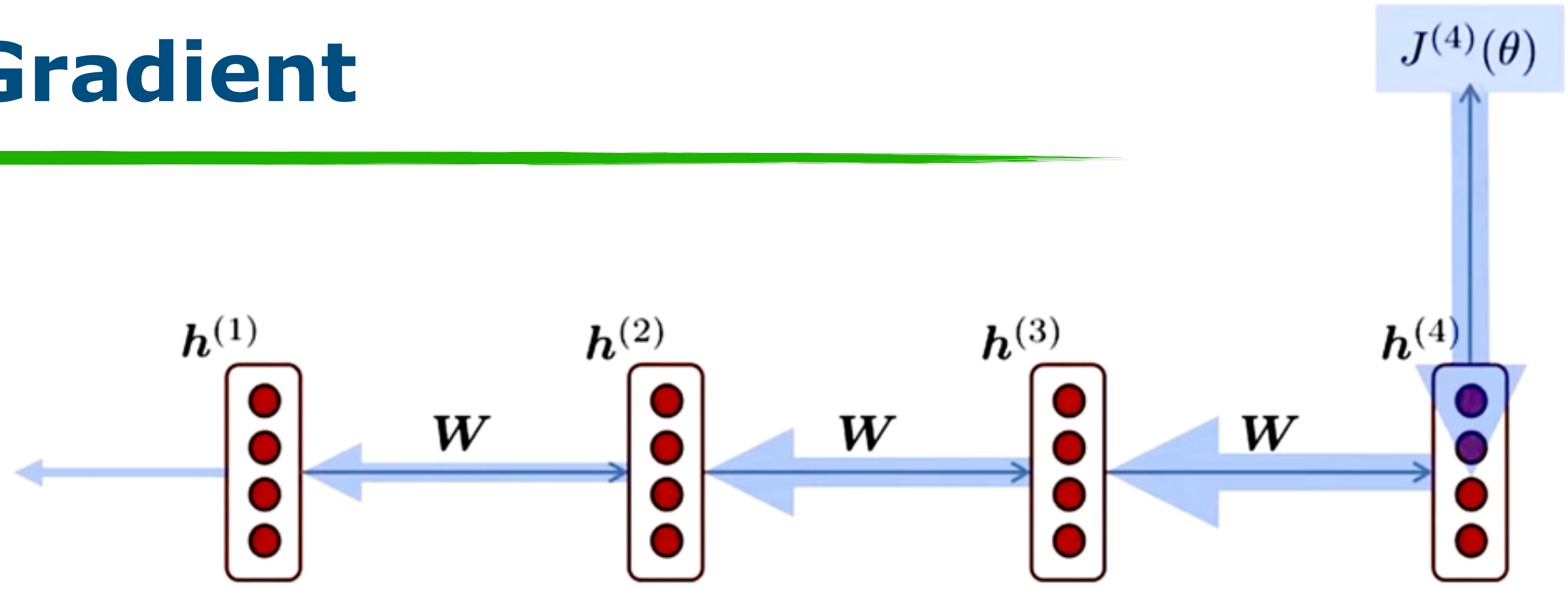
chain rule!



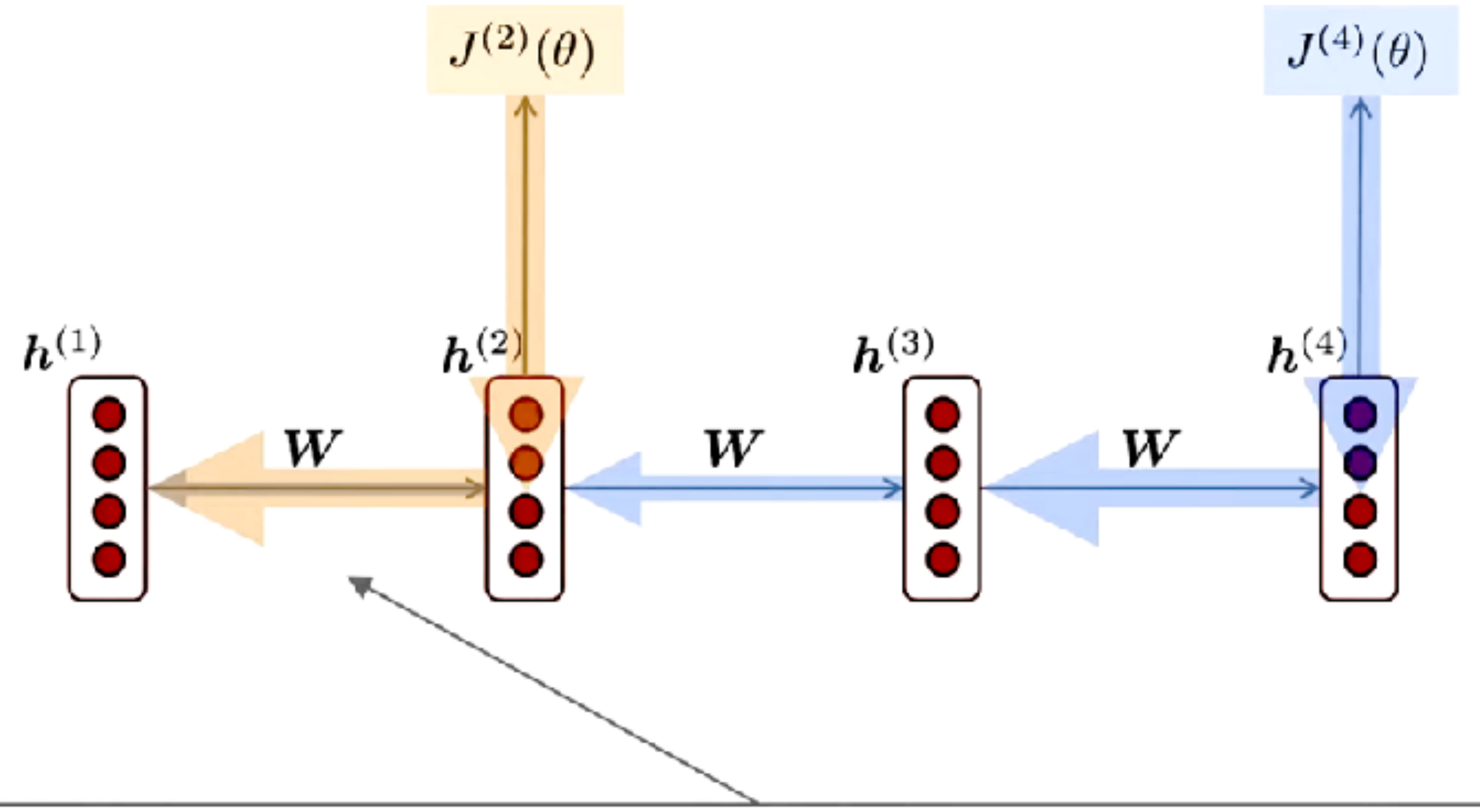
$$\frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(1)}} = \frac{\partial \boldsymbol{h}^{(2)}}{\partial \boldsymbol{h}^{(1)}} \times \qquad \frac{\partial \boldsymbol{h}^{(3)}}{\partial \boldsymbol{h}^{(2)}} \times \qquad \frac{\partial \boldsymbol{h}^{(4)}}{\partial \boldsymbol{h}^{(3)}} \times \frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(4)}}$$

chain rule!

Vanishing Gradient



$$\frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(1)}} = \boxed{\frac{\partial \boldsymbol{h}^{(2)}}{\partial \boldsymbol{h}^{(1)}}} \times \boxed{\frac{\partial \boldsymbol{h}^{(3)}}{\partial \boldsymbol{h}^{(2)}}} \times \boxed{\frac{\partial \boldsymbol{h}^{(4)}}{\partial \boldsymbol{h}^{(3)}}} \times \frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(4)}}$$



Gradient signal from far away is lost because it's much smaller than gradient signal from close-by.

So, model weights are updated only with respect to near effects, not long-term effects.

Exploding Gradient

If the gradient becomes too big, then the SGD update step becomes too big:

$$\theta^{new} = \theta^{old} - \overbrace{\alpha}^{\text{learning rate}} \underbrace{\nabla_\theta J(\theta)}_{\text{gradient}}$$

Gradient clipping: if the norm of the gradient is greater than some threshold, scale it down before applying SGD update

Algorithm 1 Pseudo-code for norm clipping

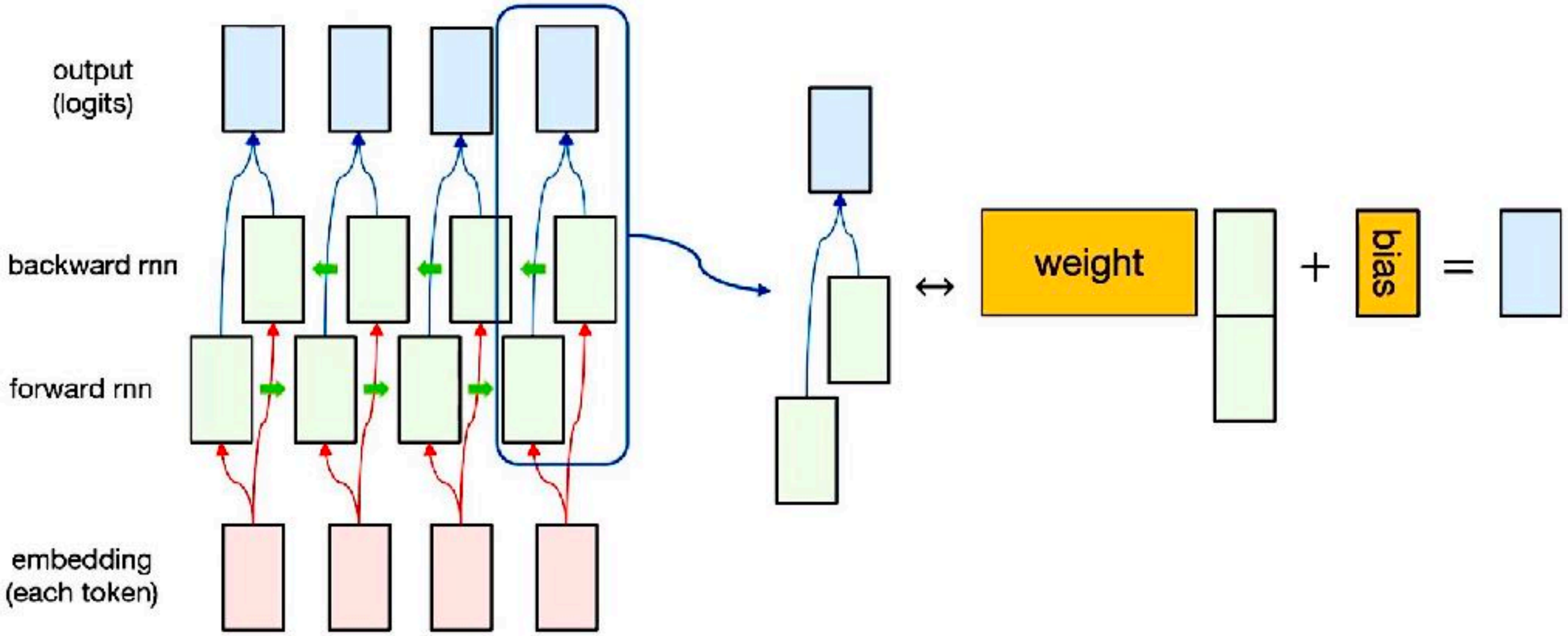
$\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$
if $\|\hat{\mathbf{g}}\| \geq threshold$ **then**
  $\hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|} \hat{\mathbf{g}}$
end if

Intuition: take a step in the same direction, but a smaller step

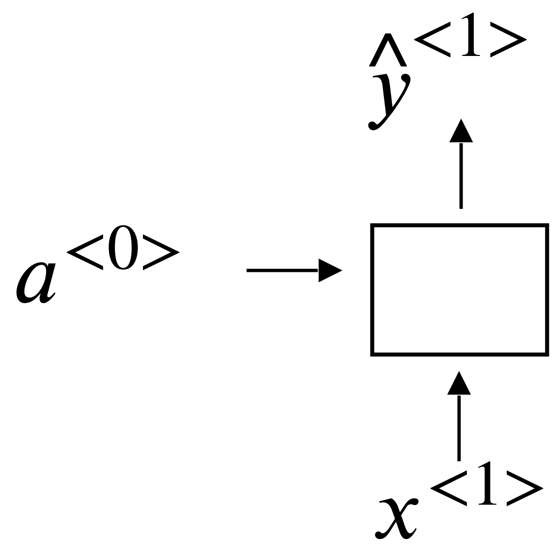
Another Problem

He said, “Teddy Roosevelt was a great President.”

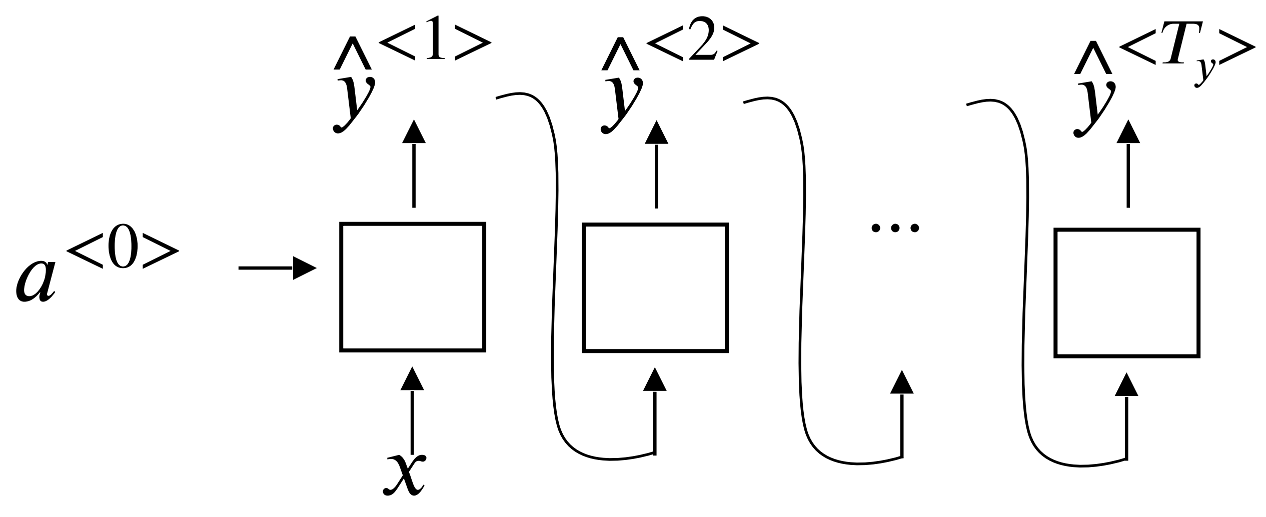
He said, “Teddy bears are on sale!”



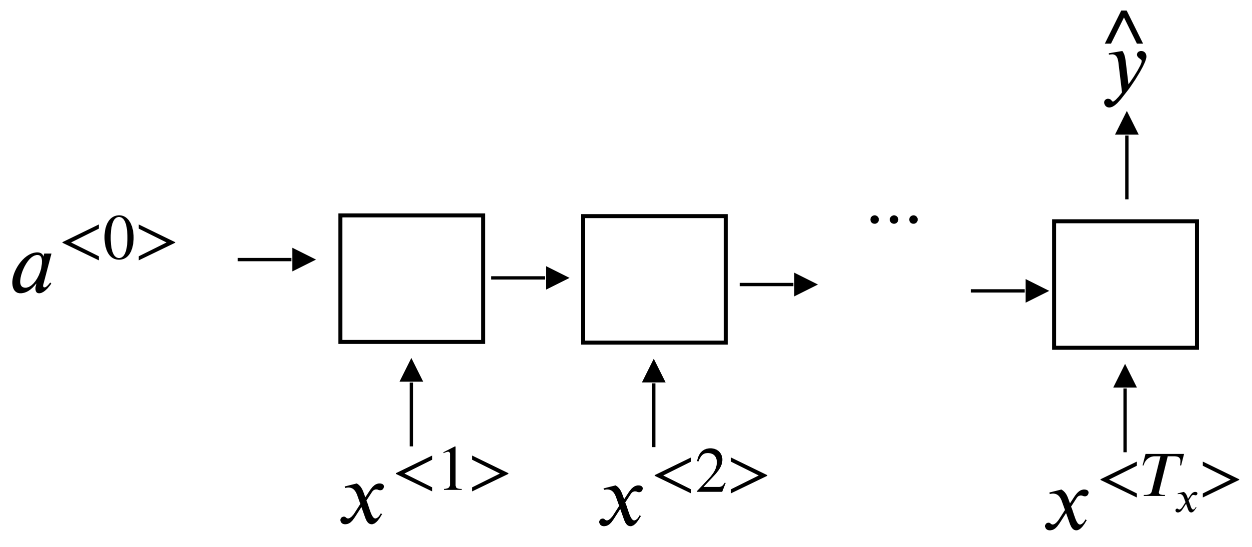
RNN architecture types



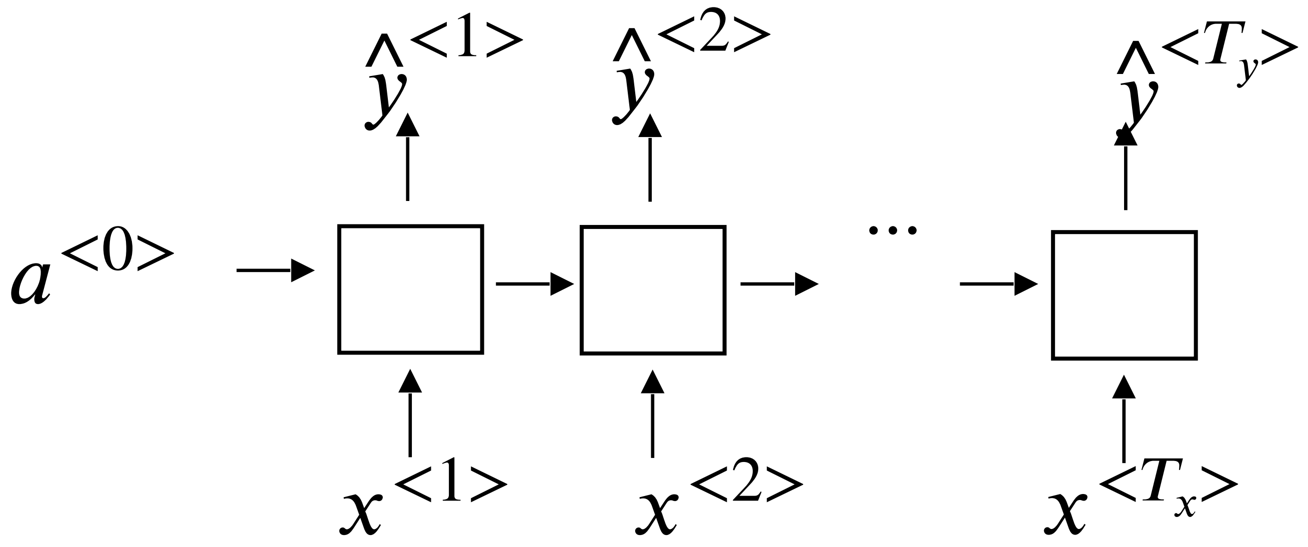
One to one



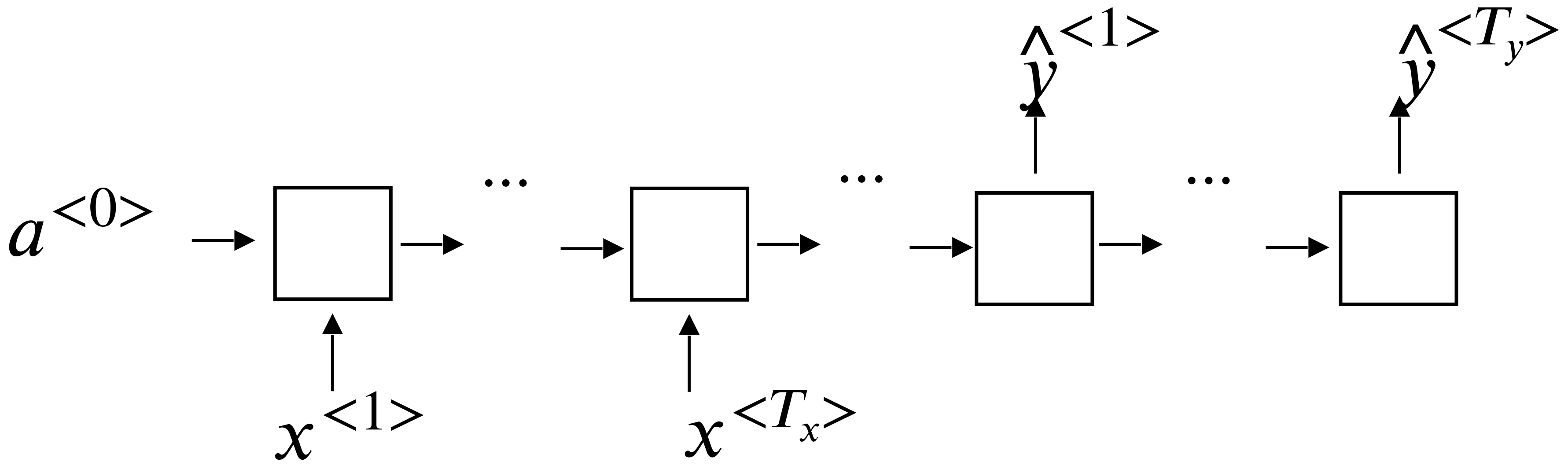
One to many



Many to one

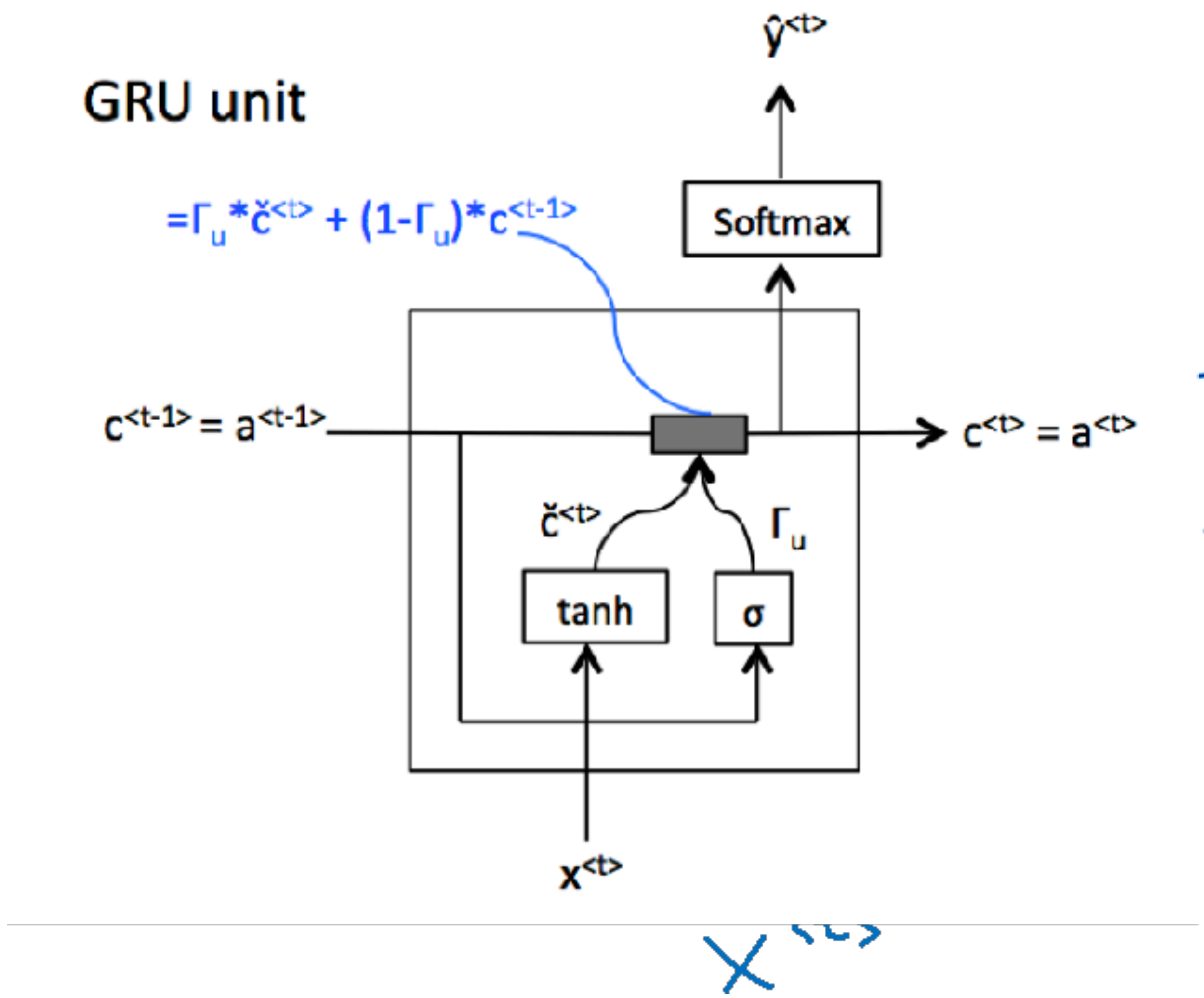


Many to many



Many to many

GRU: Gated Recurrent Unit



c = memory cell

$\rightarrow c^{<t>} = a^{<t>}$

$\rightarrow \tilde{c}^{<t>} = \tanh(W_c[c^{<t-1>}, x^{<t>}] + b_c)$

$\rightarrow \Gamma_u = \sigma(W_u[c^{<t-1>}, x^{<t>}] + b_u)$

"update"

$c^{<t>} = \underset{\Gamma_u = 1}{\Gamma_u} * \tilde{c}^{<t>} + (1-\Gamma_u) * c^{<t-1>}$

The cat, which already ate …, was full.

[Cho et al., 2014. On the properties of neural machine translation: Encoder-decoder approaches]
[Chung et al., 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling]

Full GRU

$\tilde{h}$ $$\tilde{c}^{<t>} = \tanh(W_c[\Gamma_r * c^{<t-1>}, x^{<t>}] + b_c)$$

$$\begin{array}{ll} u & \left\{ \begin{array}{l} \Gamma_u = \sigma(W_u[c^{<t-1>}, x^{<t>}] + b_u) \\ \Gamma_r = \sigma(W_r[c^{<t-1>}, x^{<t>}] + b_r) \end{array} \right. \\ r & \end{array}$$

h $$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \left(1 - \Gamma_u\right) + c^{<t-1>}$$

The cat, which ate already, was full.