Statistical Natural Language Processing [COMP0087]

Recurrent Neural Networks

Vasileios Lampos

Computer Science, UCL

lampos.net

About this lecture

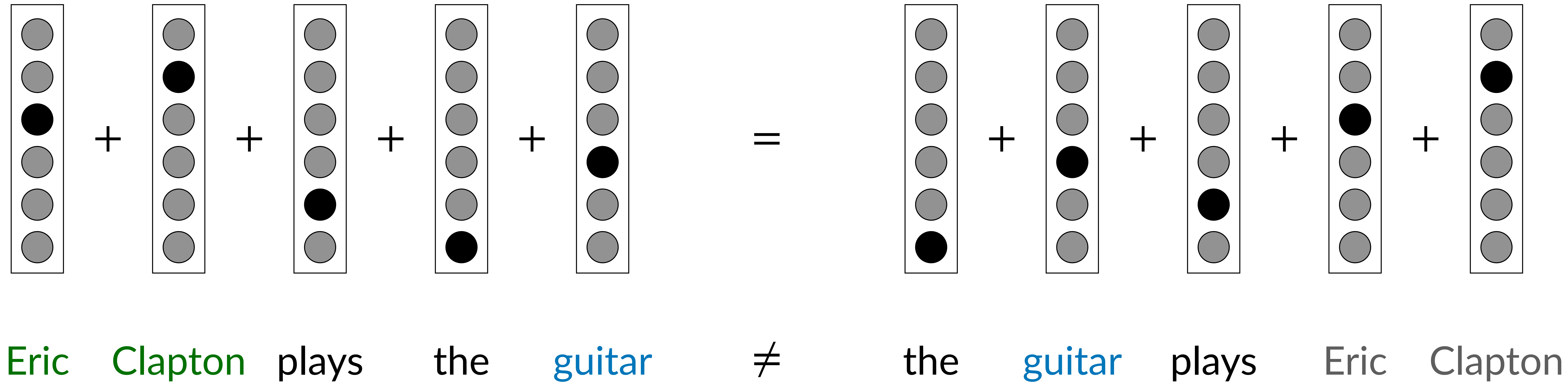
- In this lecture:
  - brief overview on language models (*more on this during the lecture by Dr. Oana-Maria Camburu*)
  - Recurrent Neural Networks
  - The Long Short-Term Memory (LSTM) architecture
  - Applications and extensions
  - slides: lampos.net/teaching

- **Reading / Lecture based on:** Chapters 3 (*less so*), 7 (*less so*), and 9 (*more so*) of "*Speech and Language Processing*" (SLP) by Jurafsky and Martin (2023) — web.stanford.edu/~jurafsky/slp3/

- Additional material
  - Difficulties in training RNNs — proceedings.mlr.press/v28/pascanu13.pdf
  - LSTMs — colah.github.io/posts/2015-08-Understanding-LSTMs/

Text order is important

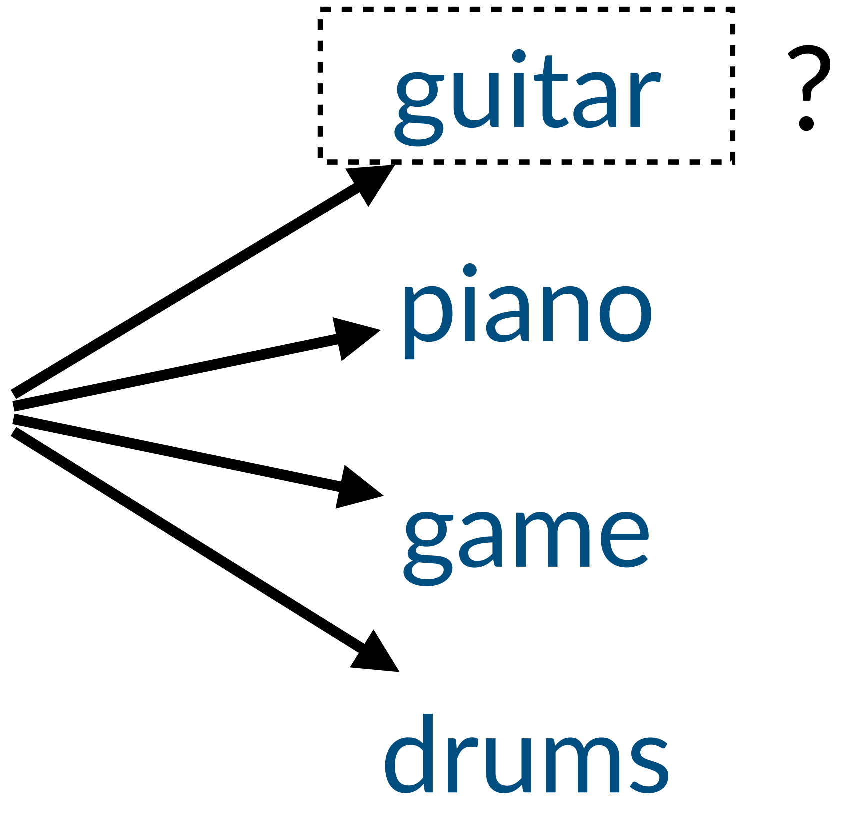
Language is a **sequence** of "events" over time



Language model

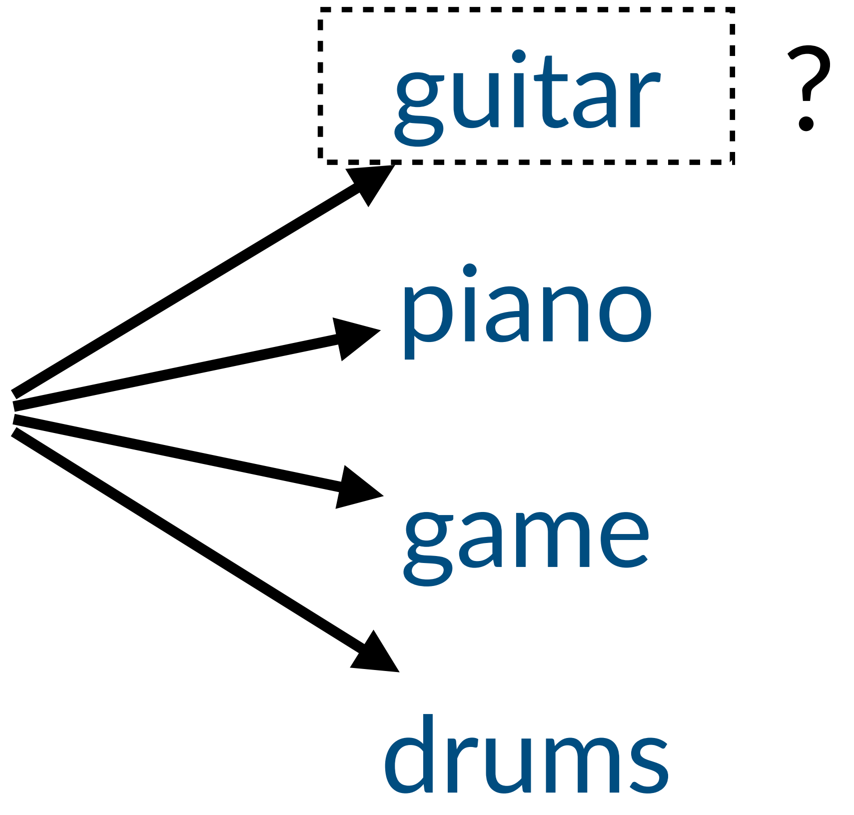
A language model predicts the next word of a word sequence:

… and all of a sudden Eric Clapton started to play the ________



A language model predicts the next word of a word sequence:

… and all of a sudden Eric Clapton started to play the ________



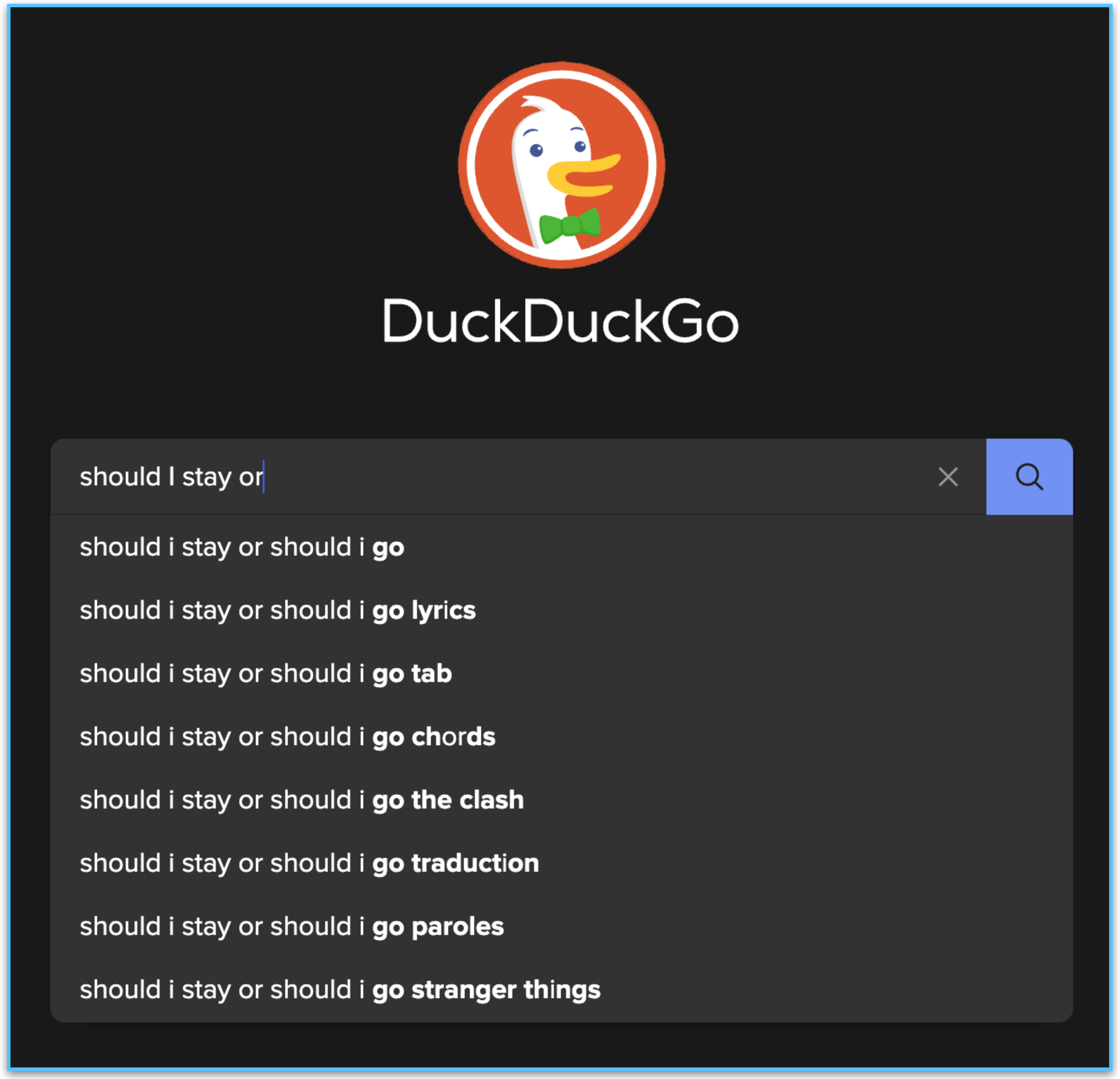
Language model

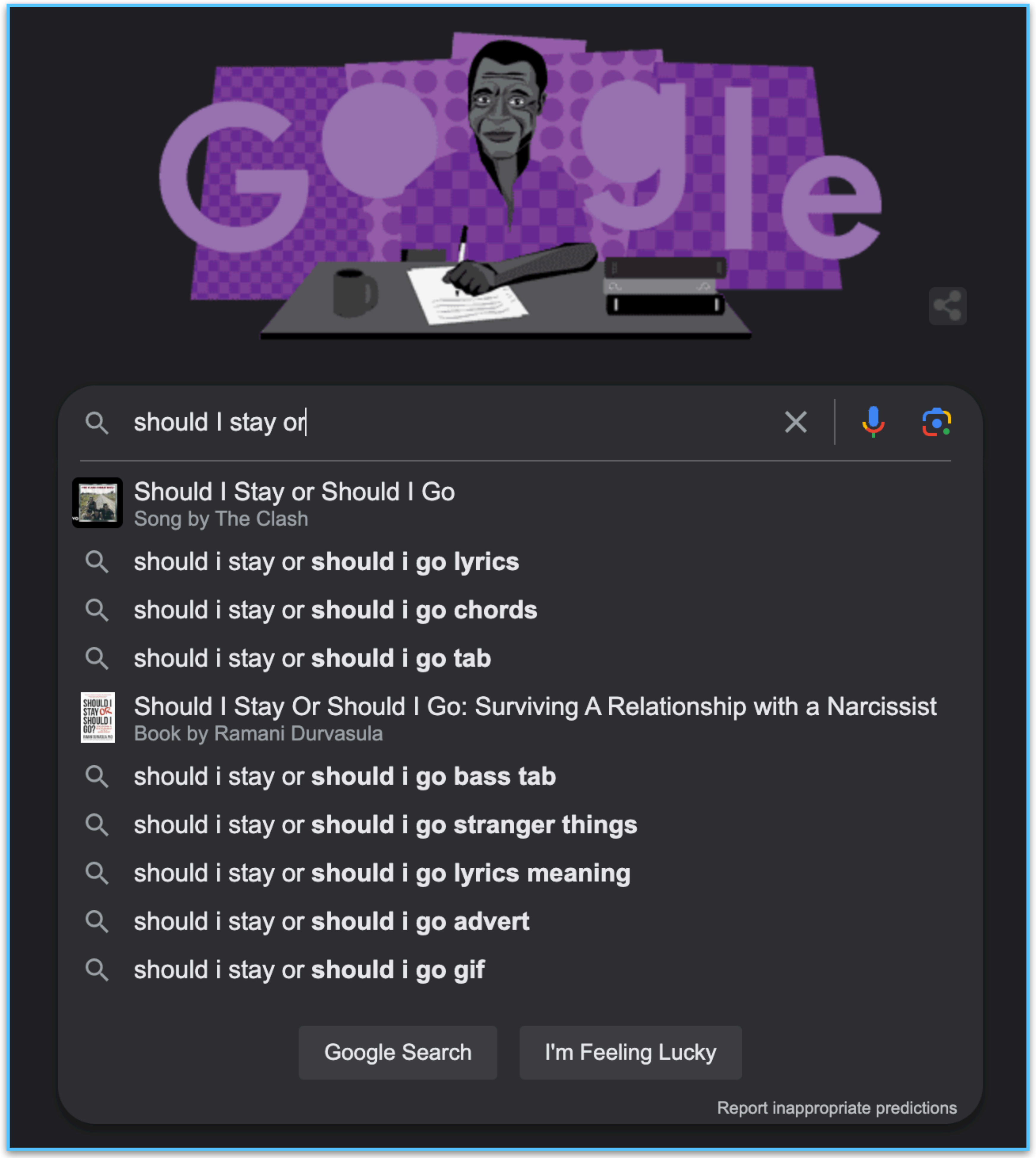
Given a sequence of words $x_1, x_2, \ldots, x_t$

compute the probability of the next word $p\left(x_{t+1} \mid x_t, x_{t-1}, \ldots, x_1\right)$

where $x_i \in \mathscr{V}$ (a word from our vocabulary)

We use language models all the time





Language model evaluation using perplexity (PPL)

$$\text{PPL} = \prod_{t=1}^{N} \left(\frac{1}{p_\ell(x_{t+1} \mid x_t, \ldots, x_1)} \right)^{\frac{1}{N}}$$

number of tokens in our corpus

lower is better

inverse probability of the corpus, according to the language model ℓ

Language model evaluation using perplexity (PPL)

$$\text{PPL} = \prod_{t=1}^{N} \left(\frac{1}{p_{\ell}\left(x_{t+1} \mid x_t, \ldots, x_1\right)} \right)^{\frac{1}{N}}$$

lower is better

number of tokens in our corpus

inverse probability of the corpus, according to the language model ℓ

Intuition: if $\text{PPL} = \delta$, then our uncertainty about the next word is ~ equivalent to the uncertainty of tossing a δ-sided dice and getting a δ

Language model evaluation using perplexity (PPL)

$$\text{PPL} = \prod_{t=1}^{N} \left(\frac{1}{p_\ell\left(x_{t+1} \mid x_t, \ldots, x_1\right)} \right)^{\frac{1}{N}}$$

number of tokens in our corpus

lower is better

inverse probability of the corpus, according to the language model ℓ

Intuition: if $\text{PPL} = \delta$, then our uncertainty about the next word is ~ equivalent to the uncertainty of tossing a δ-sided dice and getting a δ

$$\text{PPL} = \prod_{t=1}^{N} \left(\frac{1}{\hat{\mathbf{y}}^{[t]}_{x_{t+1}}} \right)^{\frac{1}{N}} = \cdots = \exp\left(L(\theta)\right)$$

the estimated prob. at word t that the next word is x_{t+1} based on the language model

see 3.8 in SLP

cross entropy loss of a language model parametrised by θ

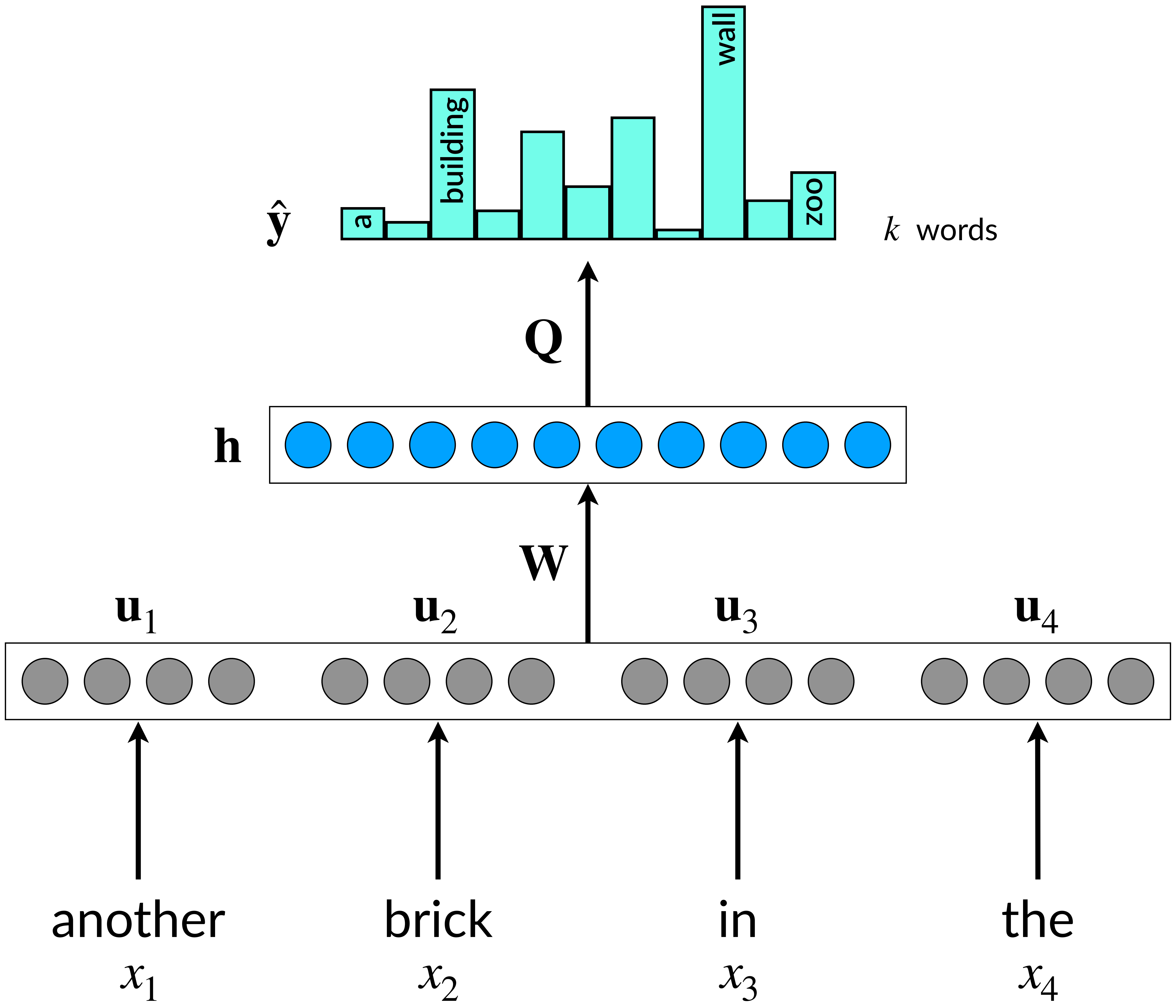
Language model evaluation using perplexity (PPL)

Model	PPL
Interpolated Kneser-Ney 5-gram *(2013)*	67.6
RNN-1024 + MaxEnt 9-gram *(2013)*	51.3
LSTM-2048 *(2016)*	43.7
2-layer LSTM-8192 *(2016)*	30
Adaptive input Transformer *(2019)*	23.02
GPT-2 *(2019)*	16.45

But of course, there is a limit on how low perplexity can realistically be!

Source 1: engineering.fb.com/2016/10/25/ml-applications/building-an-efficient-neural-language-model-over-a-billion-words/
Source 2: openreview.net/pdf?id=ByxZX20qFQ
Source 3: huggingface.co/docs/transformers/perplexity

A foundational neural language model

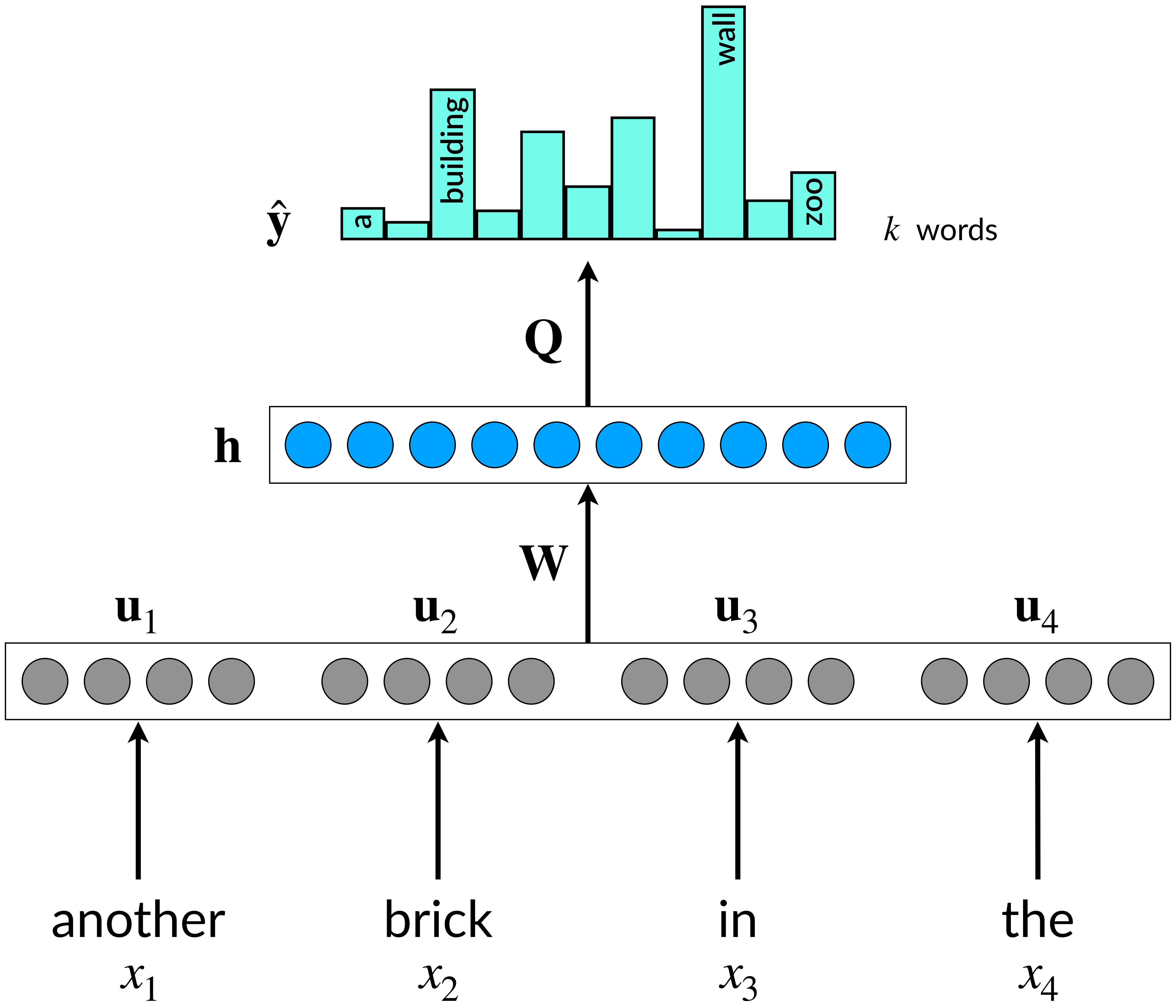


Paper: dl.acm.org/doi/pdf/10.5555/944919.944966

A foundational neural language model

$$\mathbf{u} = [\mathbf{u}_1; \mathbf{u}_2; \mathbf{u}_3; \mathbf{u}_4] \quad \in \mathbb{R}^{4d}$$

concatenate
word representations



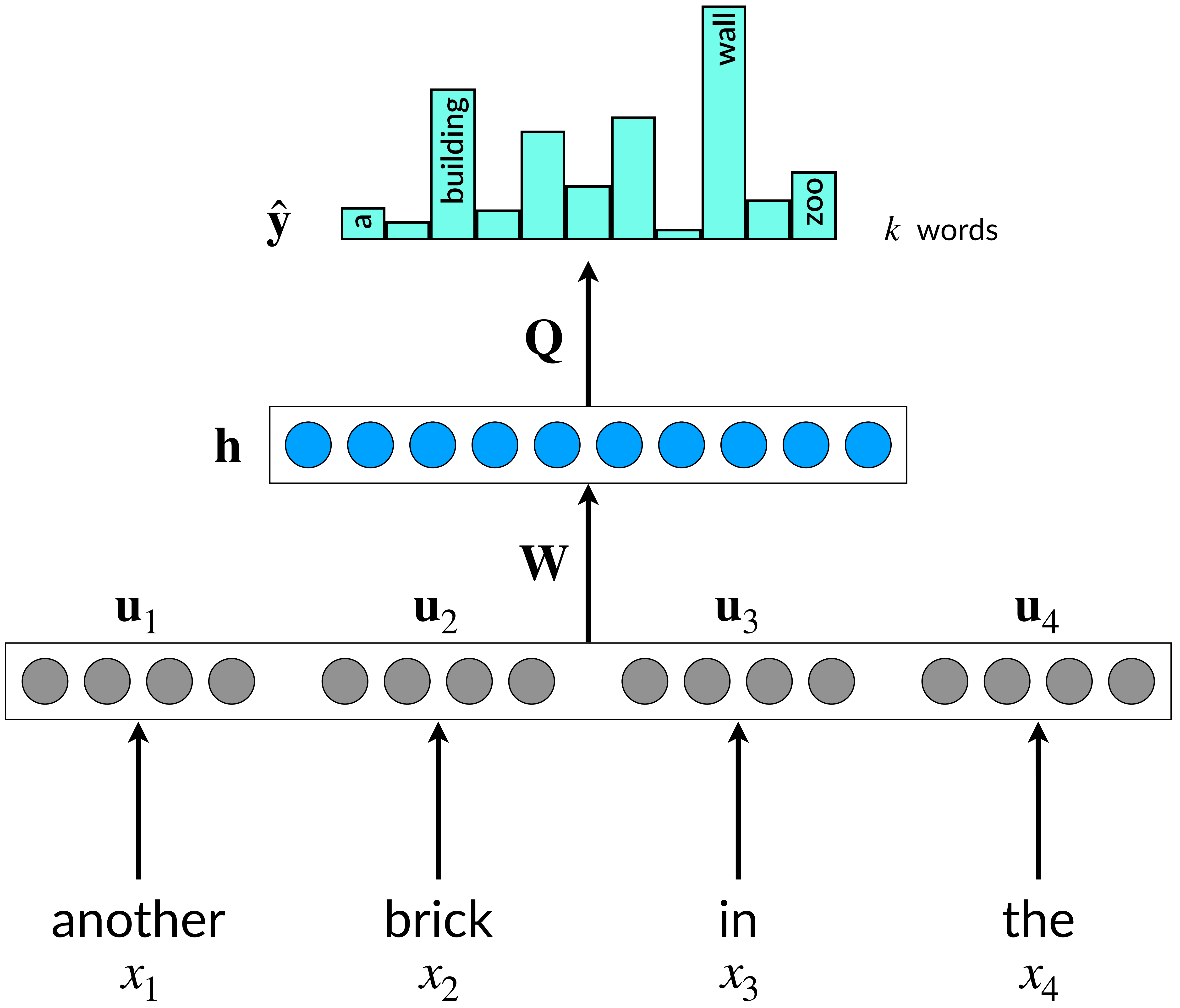
A foundational neural language model

$$\mathbf{h} = \sigma\left(\mathbf{W} \cdot \mathbf{u} + \mathbf{b}_W\right) \quad \in \mathbb{R}^m$$

$$\mathbf{W} \in \mathbb{R}^{m \times 4d}$$

$$\mathbf{u} = [\mathbf{u}_1; \mathbf{u}_2; \mathbf{u}_3; \mathbf{u}_4] \quad \in \mathbb{R}^{4d}$$

concatenate
word representations



A foundational neural language model

$$\hat{\mathbf{y}} = \text{softmax}\left(\mathbf{Q} \cdot \mathbf{h} + \mathbf{b}_Q\right) \quad \in [0,1]^k$$

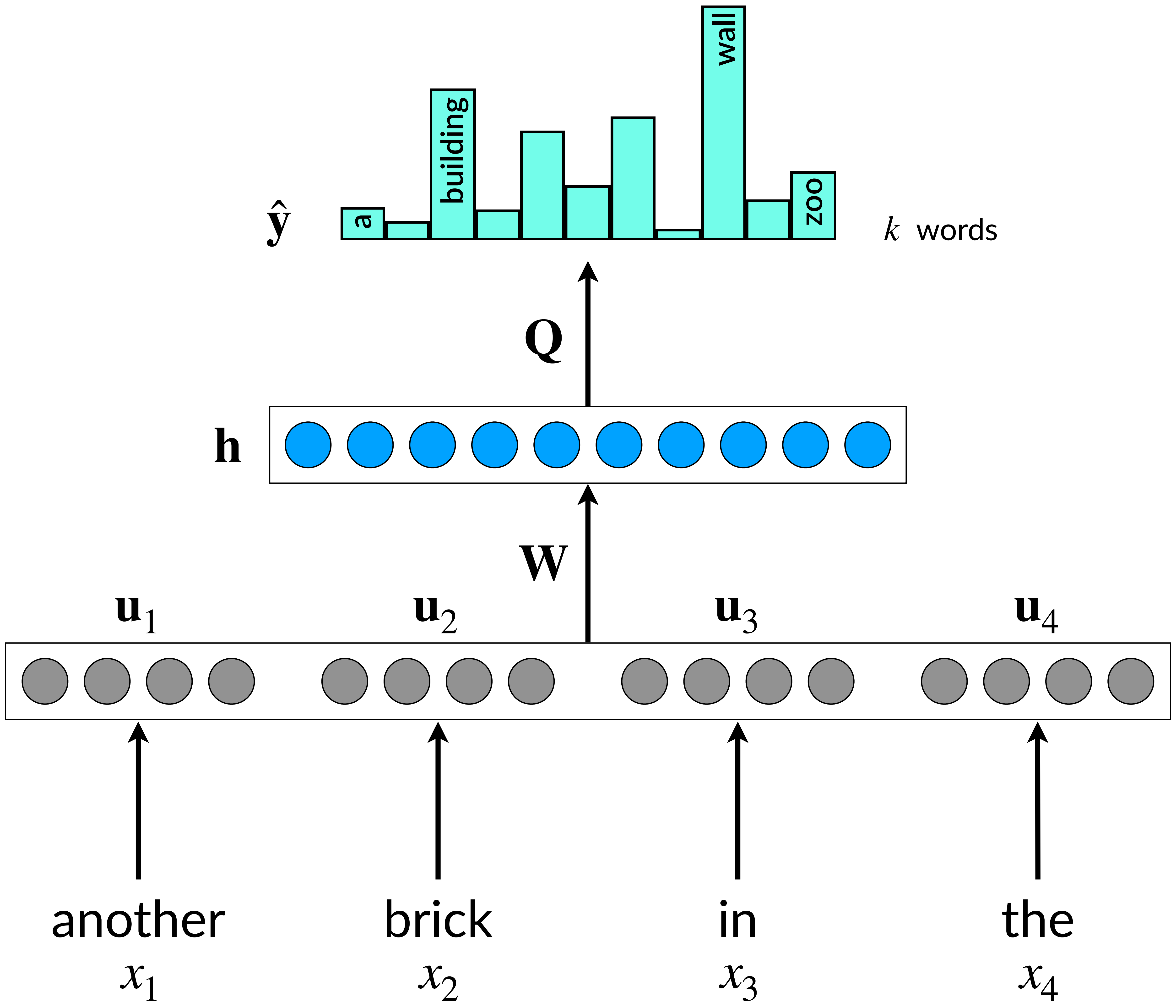
$$\mathbf{Q} \in \mathbb{R}^{k \times m}$$

$$\mathbf{h} = \sigma\left(\mathbf{W} \cdot \mathbf{u} + \mathbf{b}_W\right) \quad \in \mathbb{R}^m$$

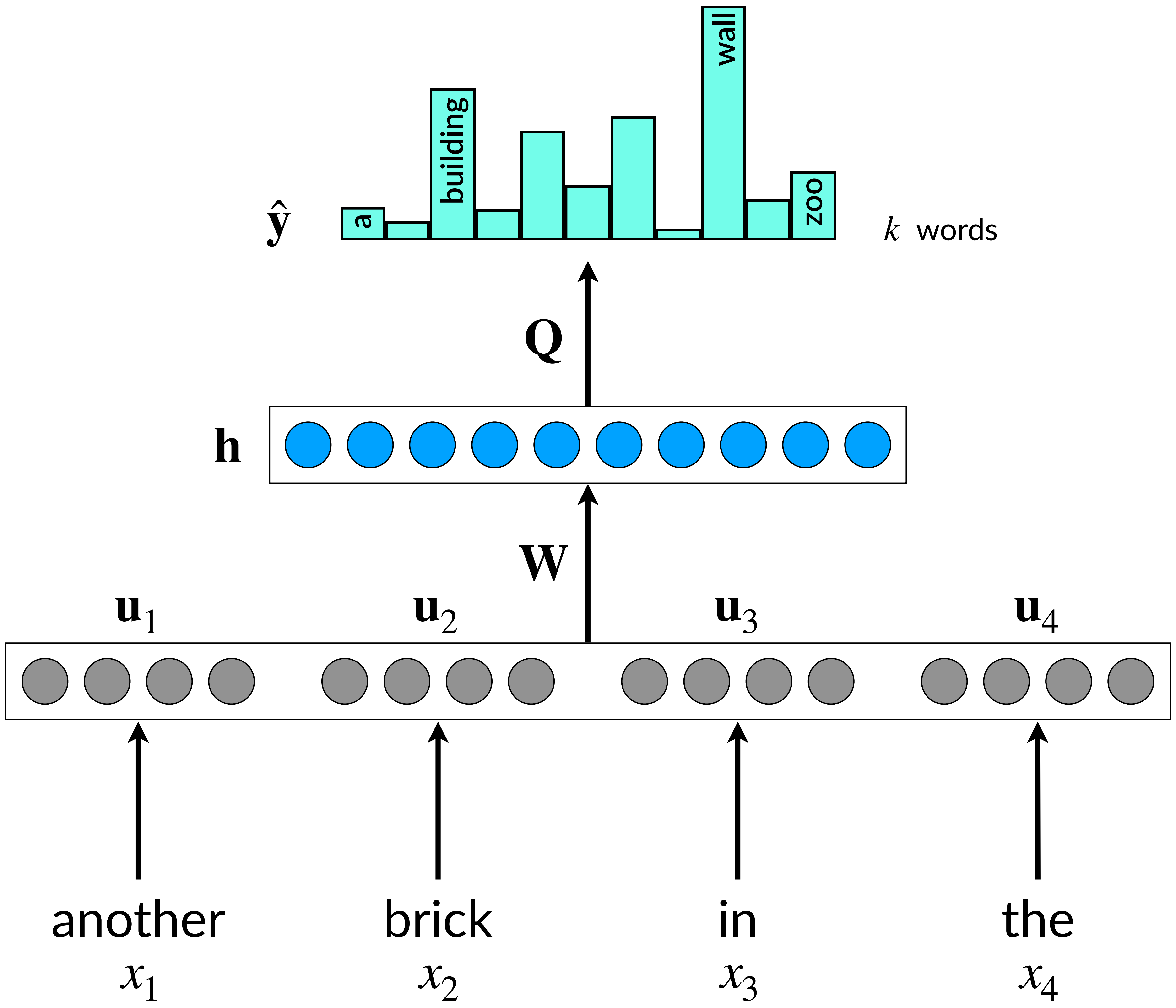
$$\mathbf{W} \in \mathbb{R}^{m \times 4d}$$

$$\mathbf{u} = [\mathbf{u}_1; \mathbf{u}_2; \mathbf{u}_3; \mathbf{u}_4] \quad \in \mathbb{R}^{4d}$$

concatenate
word representations



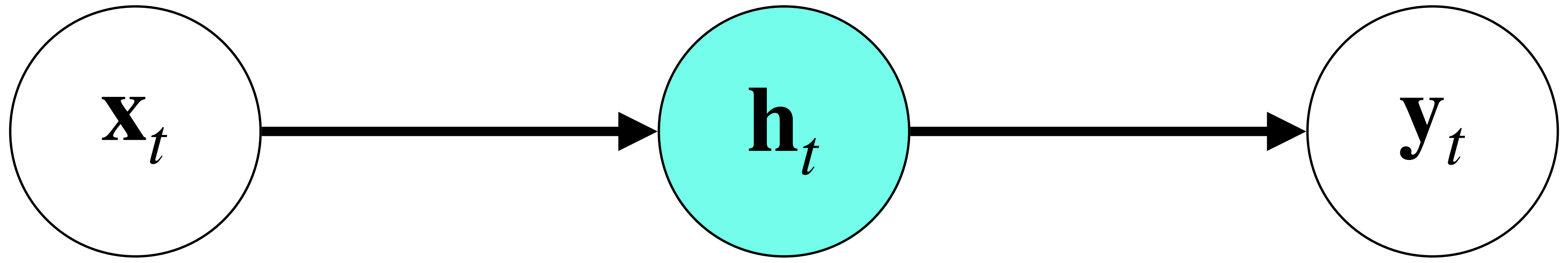
$$\hat{\mathbf{y}} = \text{softmax}\left(\mathbf{Q} \cdot \mathbf{h} + \mathbf{b}_Q\right) \quad \in [0,1]^k$$
$$\mathbf{Q} \in \mathbb{R}^{k \times m}$$
$$\mathbf{h} = \sigma\left(\mathbf{W} \cdot \mathbf{u} + \mathbf{b}_W\right) \quad \in \mathbb{R}^m$$
$$\mathbf{W} \in \mathbb{R}^{m \times 4d}$$
$$\mathbf{u} = [\mathbf{u}_1; \mathbf{u}_2; \mathbf{u}_3; \mathbf{u}_4] \quad \in \mathbb{R}^{4d}$$



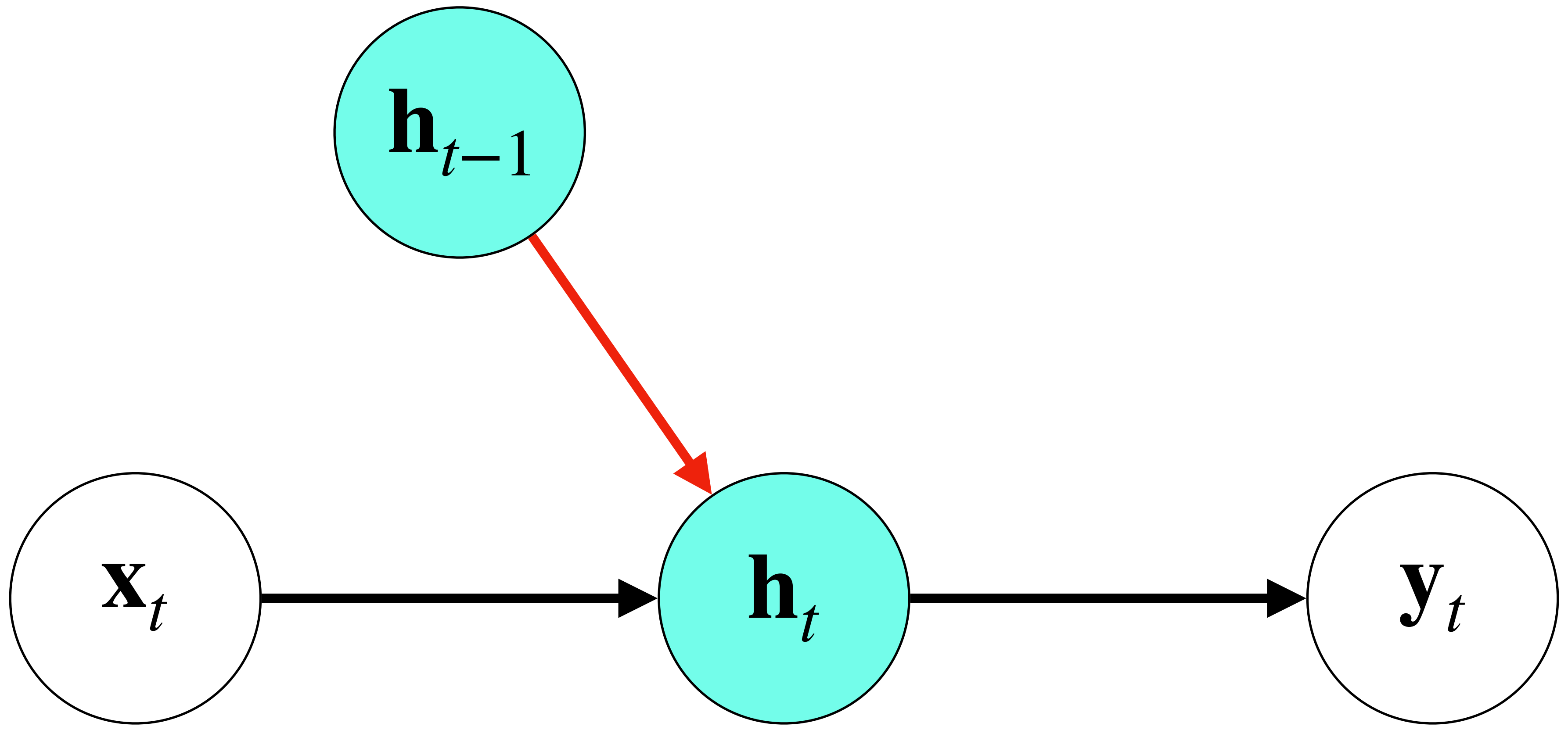
Issues!

- context / window size is fixed
- $\mathbf{W}$ grows if we increase the window
- word position is modelled explicitly and independently, i.e. there is no weight sharing between words

Recurrent Neural Network (RNN) — Intuition



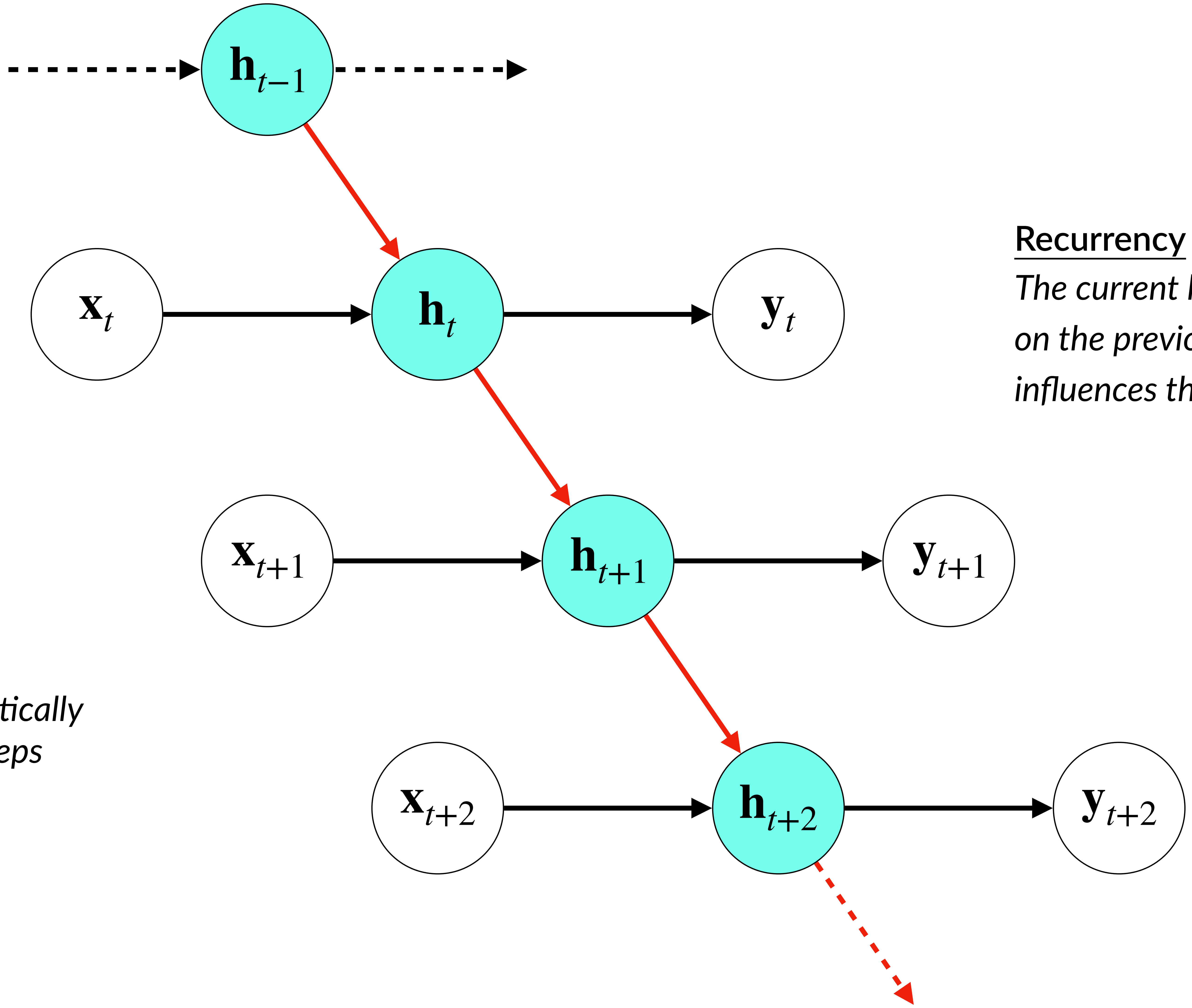
Recurrent Neural Network (RNN) — Intuition



<u>Recurrency</u>

The current hidden state $\mathbf{h}_t$ *depends on the previous hidden state* $\mathbf{h}_{t-1}$ *and influences the next hidden state* $\mathbf{h}_{t+1}$

Recurrent Neural Network (RNN) — Intuition

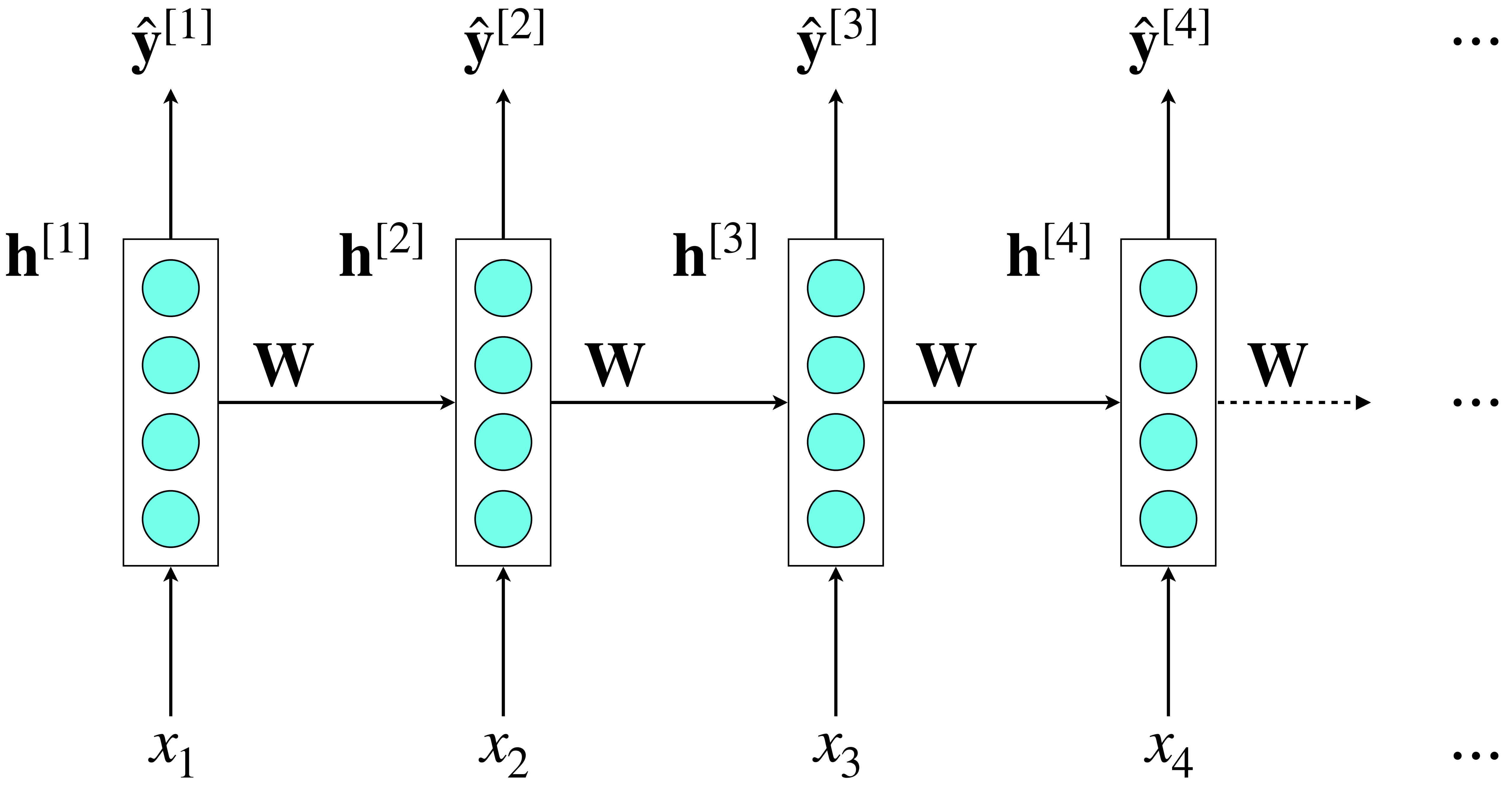


<u>Recurrency</u>

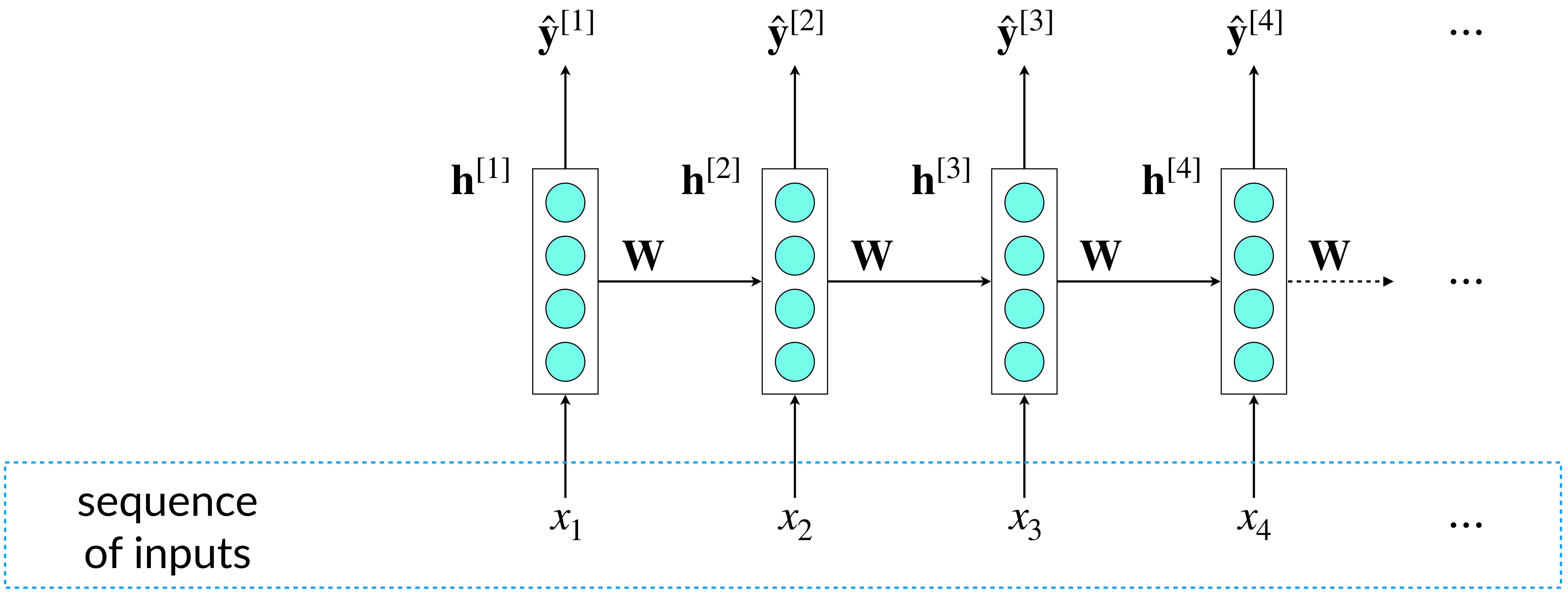
The current hidden state $\mathbf{h}_t$ depends on the previous hidden state $\mathbf{h}_{t-1}$ and influences the next hidden state $\mathbf{h}_{t+1}$

The RNN unrolls to a theoretically unlimited number of time steps

Recurrent Neural Networks (RNNs)

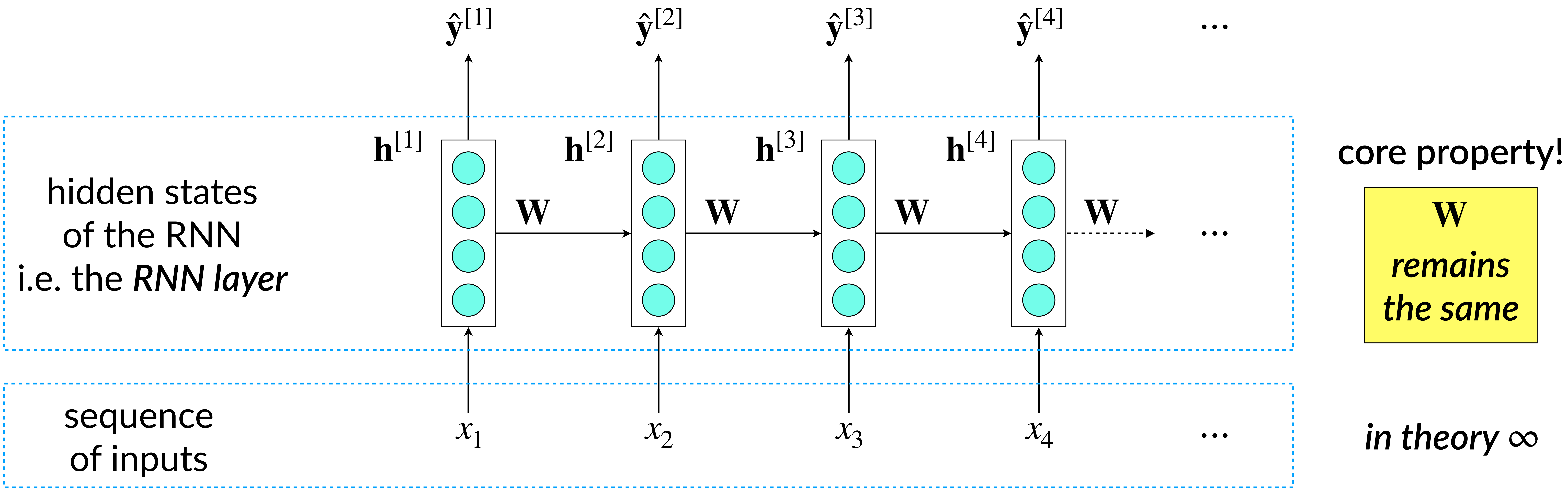


Recurrent Neural Networks (RNNs)

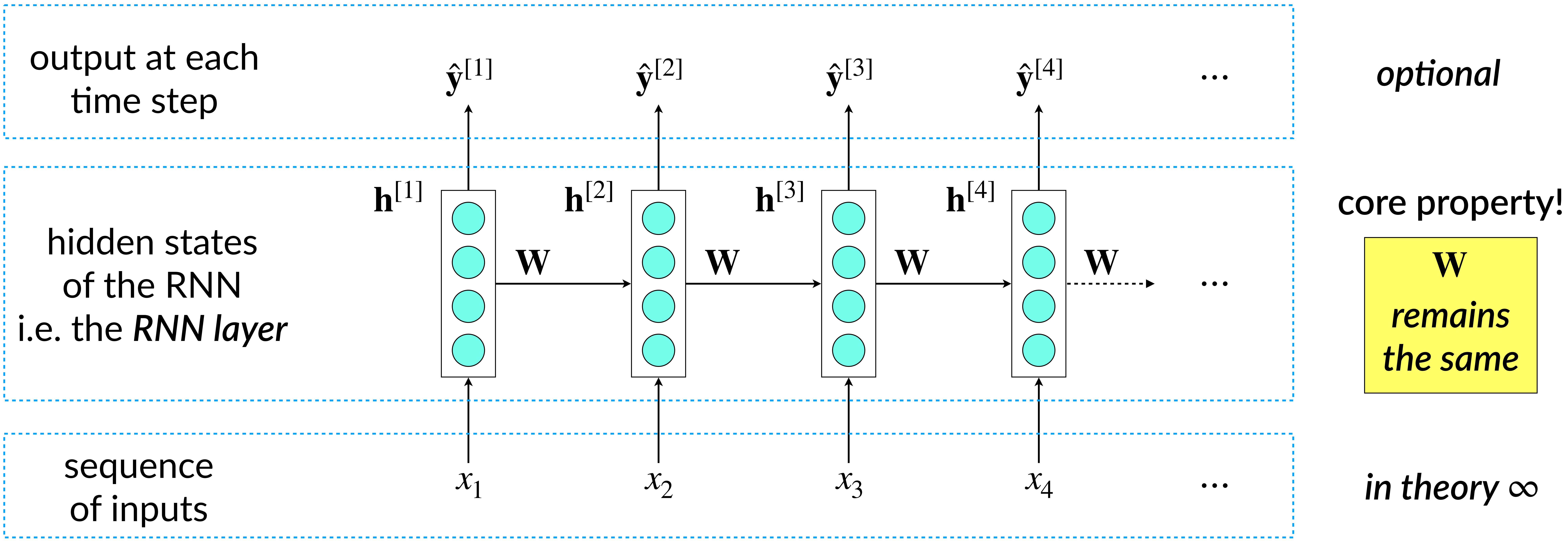


in theory ∞

Recurrent Neural Networks (RNNs)

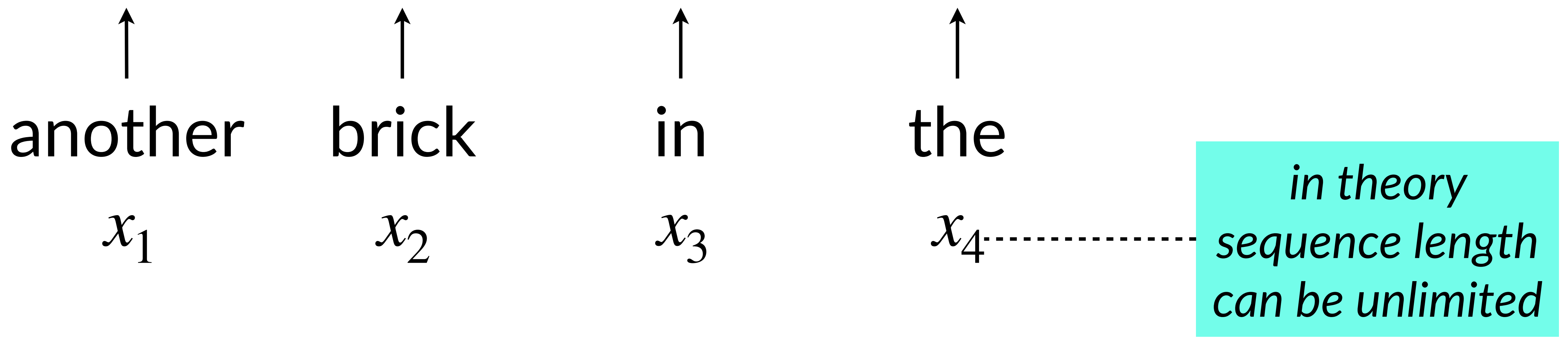


Recurrent Neural Networks (RNNs)

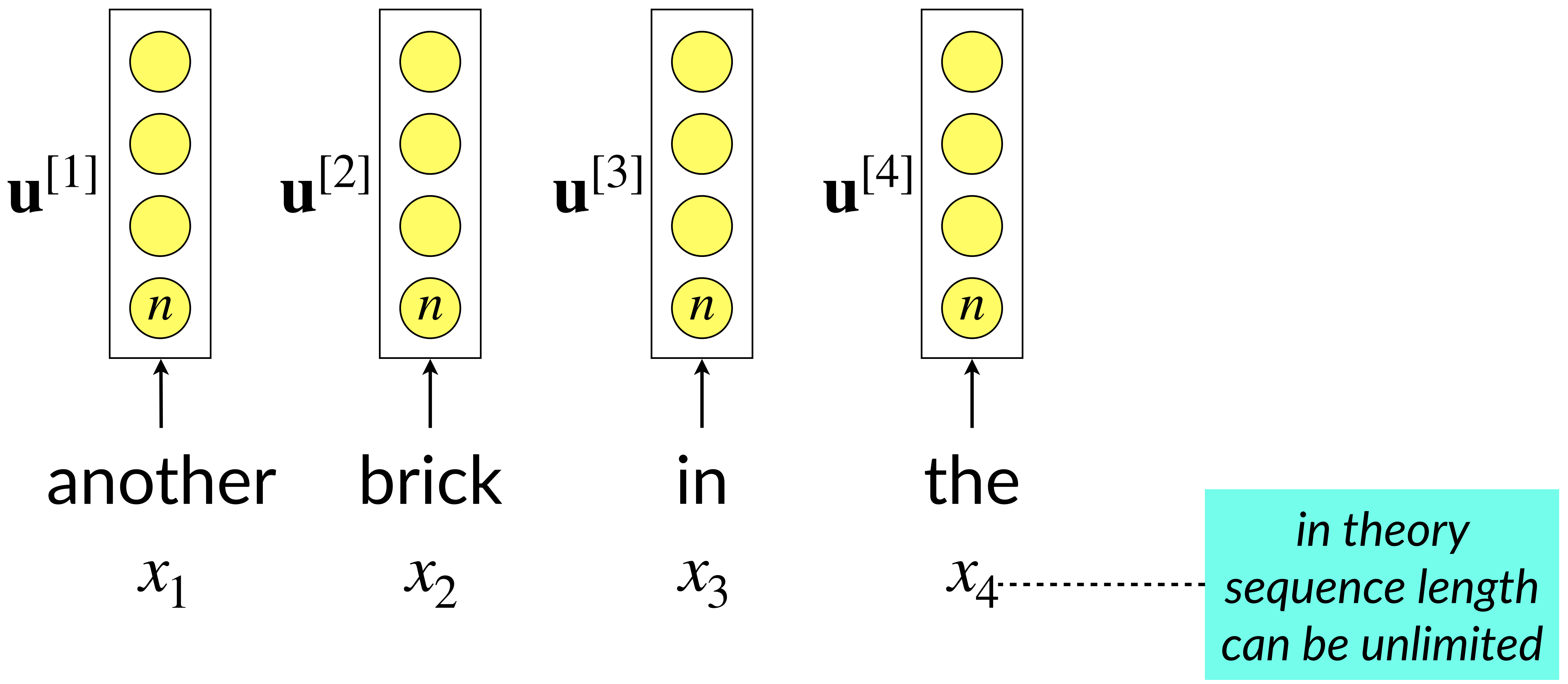


An RNN-based language model

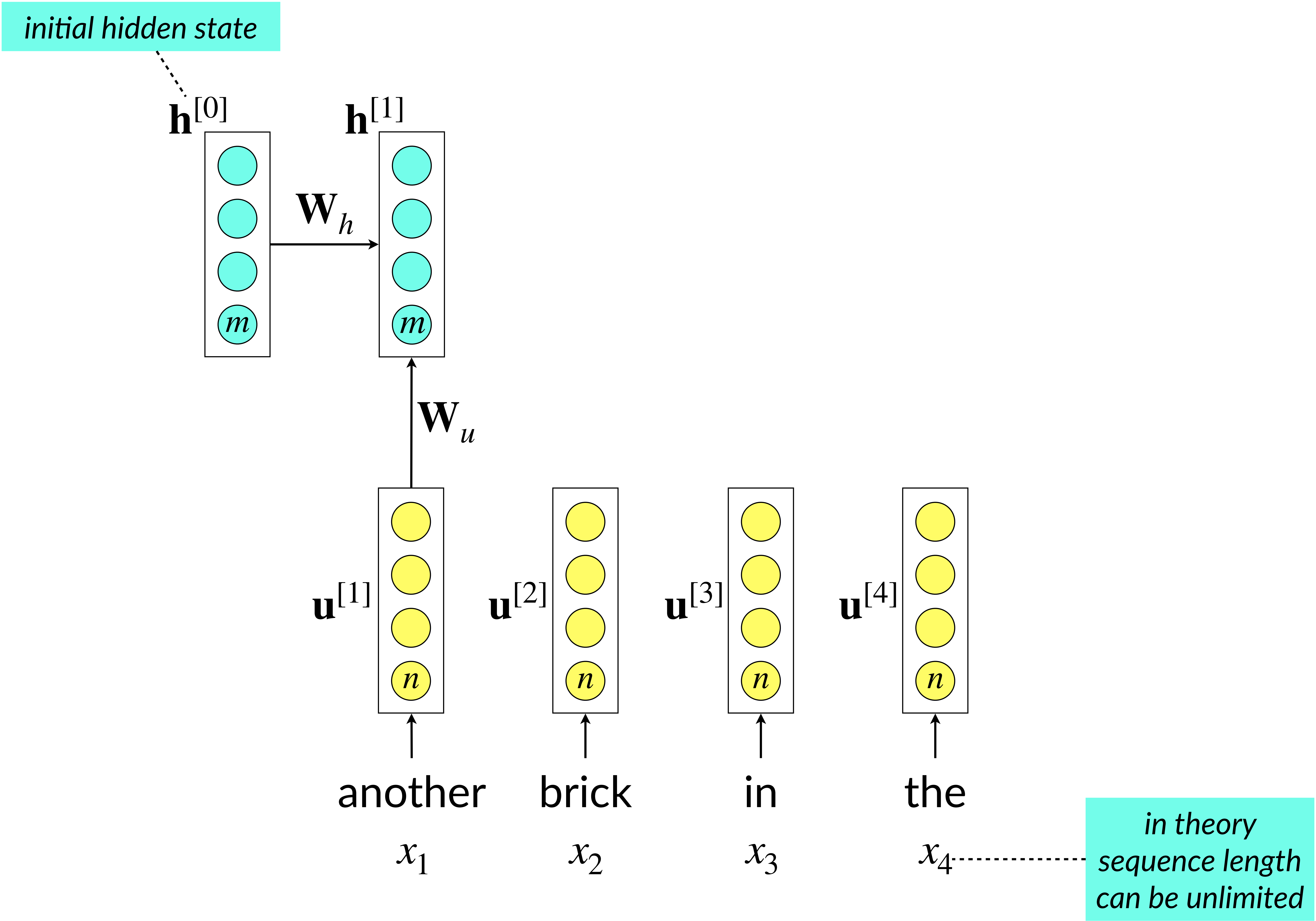
An RNN-based language model



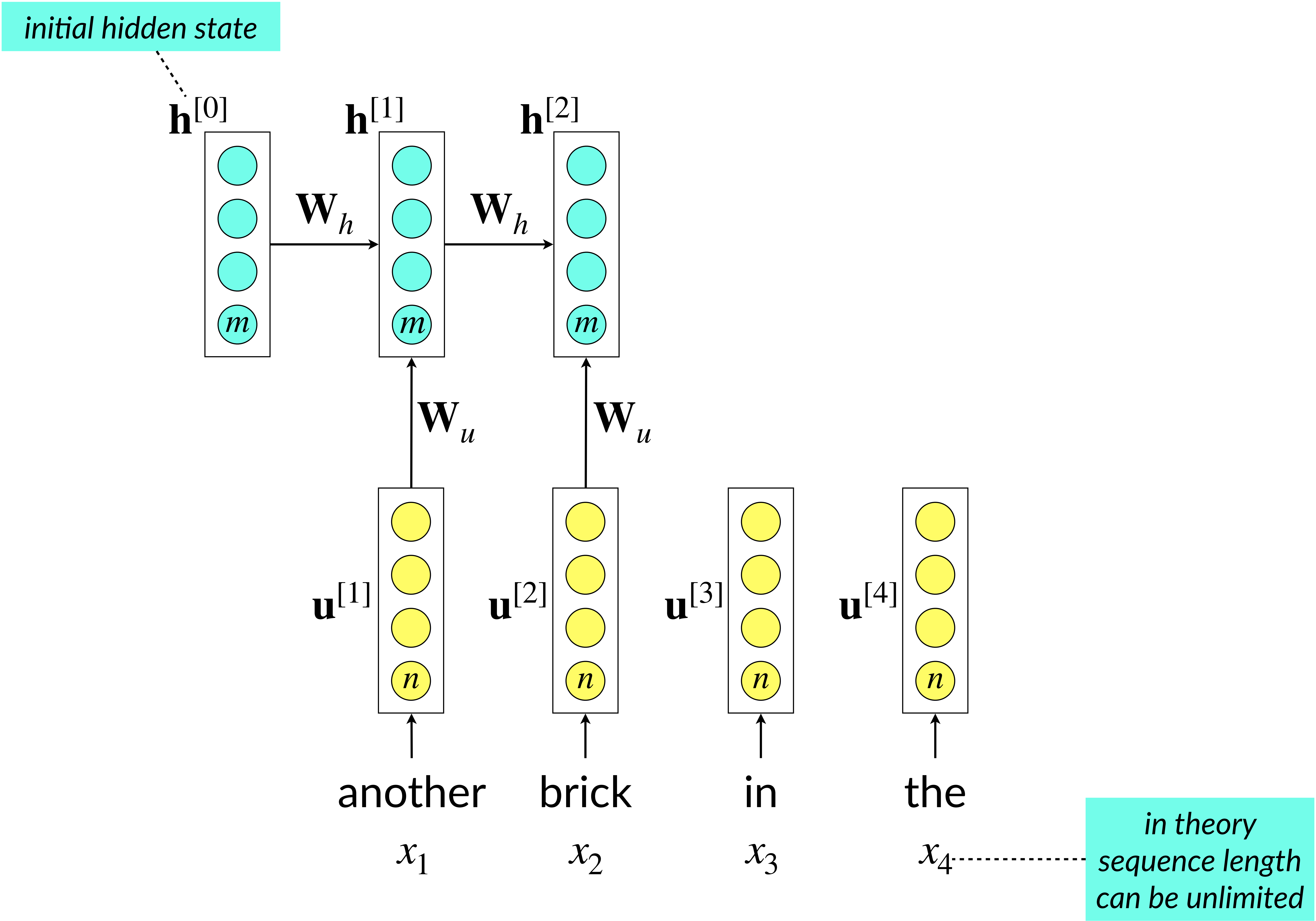
An RNN-based language model



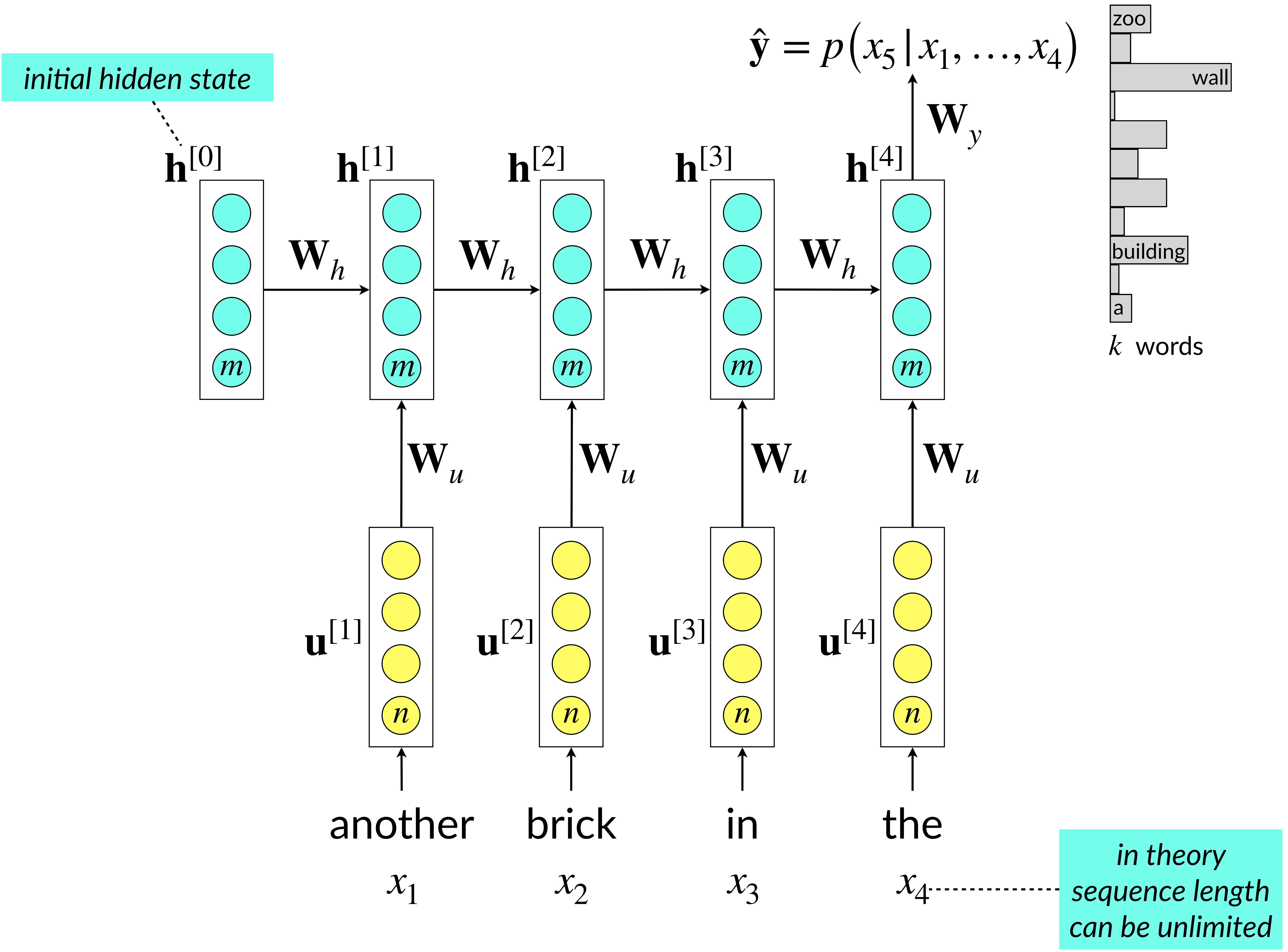
An RNN-based language model



An RNN-based language model



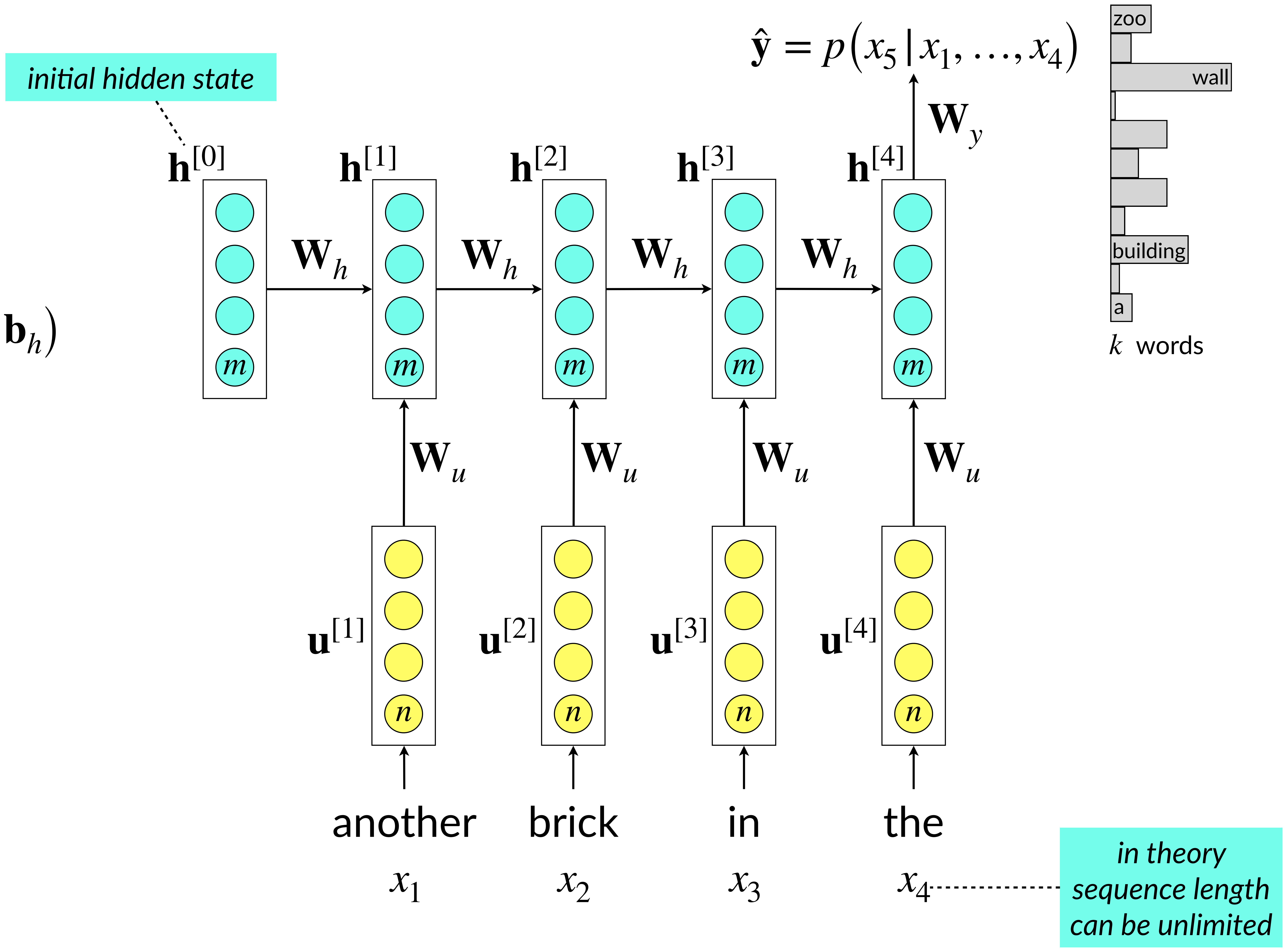
An RNN-based language model



Hidden states

$$\mathbf{h}^{[t]} = \sigma\left(\mathbf{W}_u \cdot \mathbf{u}^{[t]} + \mathbf{W}_h \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_h\right)$$

or use $\tanh(\cdot)$



An RNN-based language model

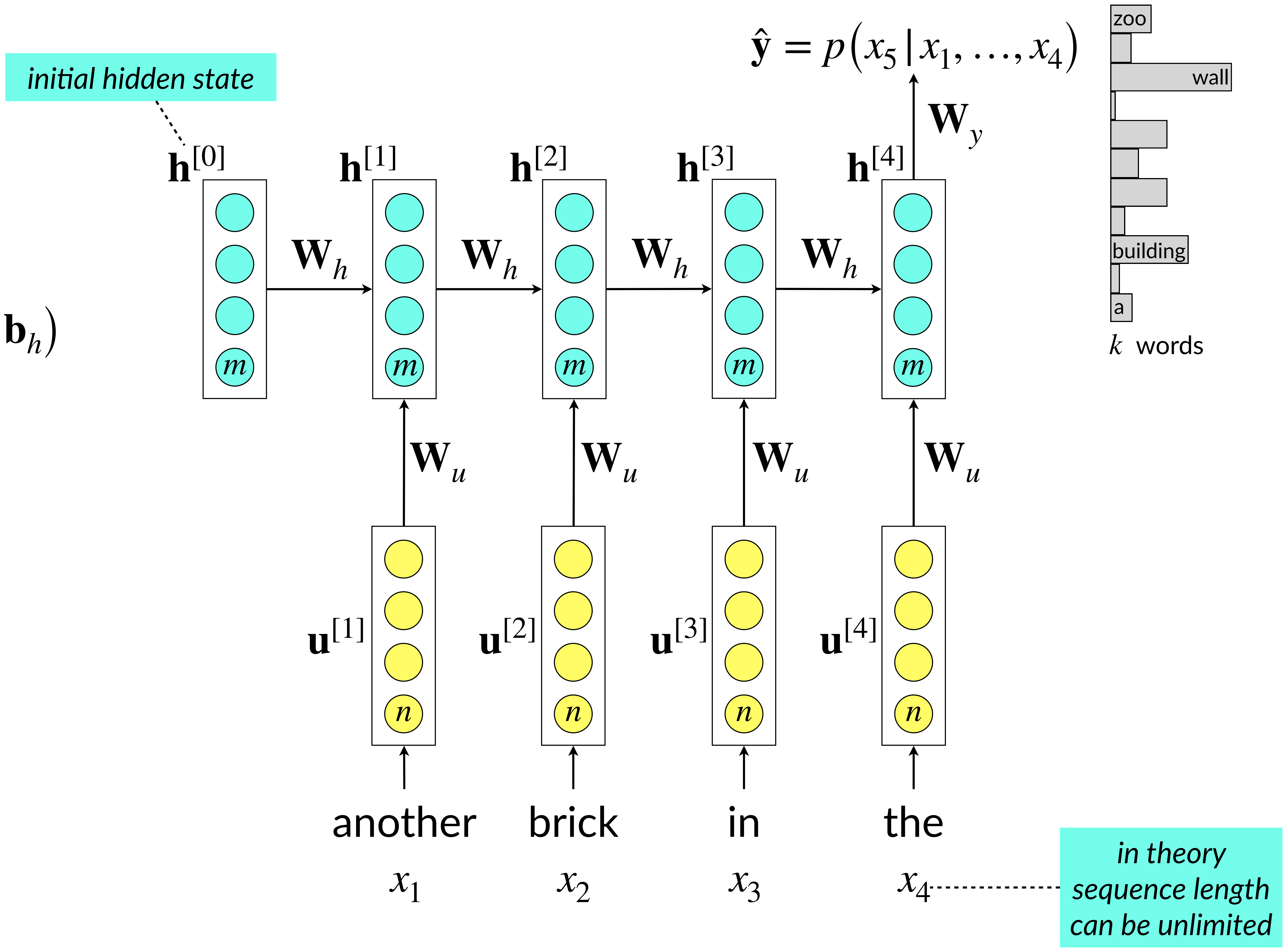
Output

$$\hat{\mathbf{y}} = \text{softmax}\left(\mathbf{W}_y \cdot \mathbf{h}^{[4]} + \mathbf{b}_y\right)$$

Hidden states

$$\mathbf{h}^{[t]} = \sigma\left(\mathbf{W}_u \cdot \mathbf{u}^{[t]} + \mathbf{W}_h \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_h\right)$$

or use $\tanh(\cdot)$



An RNN-based language model

Output

$$\hat{\mathbf{y}} = \text{softmax}\left(\mathbf{W}_y \cdot \mathbf{h}^{[4]} + \mathbf{b}_y\right)$$

Hidden states

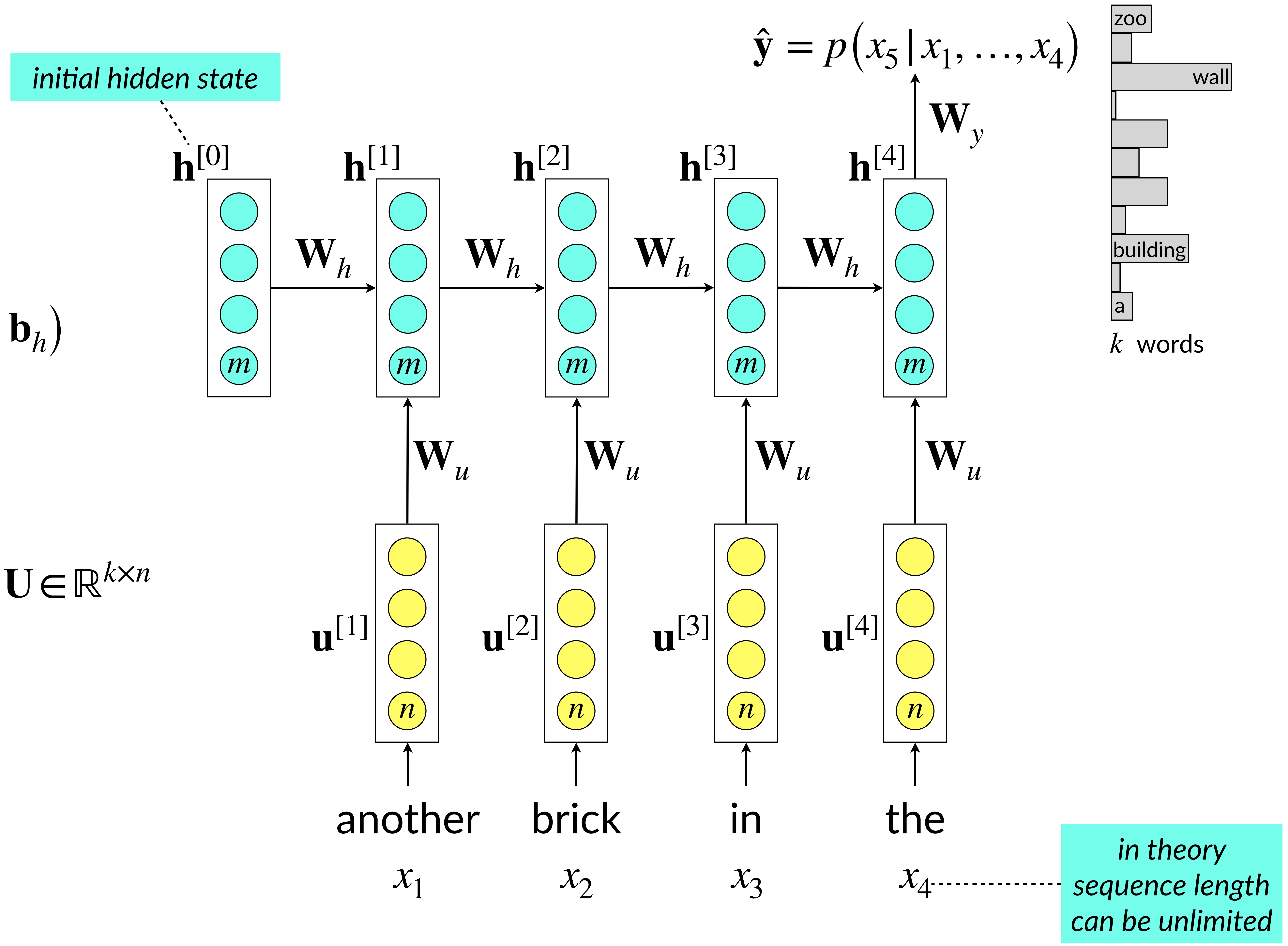
$$\mathbf{h}^{[t]} = \sigma\left(\mathbf{W}_u \cdot \mathbf{u}^{[t]} + \mathbf{W}_h \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_h\right)$$

or use $\tanh(\cdot)$

Dimensionalities?

$\mathbf{u}^{[t]} \in \mathbb{R}^n$ embedding of x_t from $\mathbf{U} \in \mathbb{R}^{k \times n}$

$\mathbf{h}^{[t]}, \mathbf{b}_h \in \mathbb{R}^m$



An RNN-based language model

Output

$\hat{\mathbf{y}} = \text{softmax}\left(\mathbf{W}_y \cdot \mathbf{h}^{[4]} + \mathbf{b}_y\right)$

Hidden states

$\mathbf{h}^{[t]} = \sigma\left(\mathbf{W}_u \cdot \mathbf{u}^{[t]} + \mathbf{W}_h \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_h\right)$

or use $\tanh(\cdot)$

Dimensionalities?

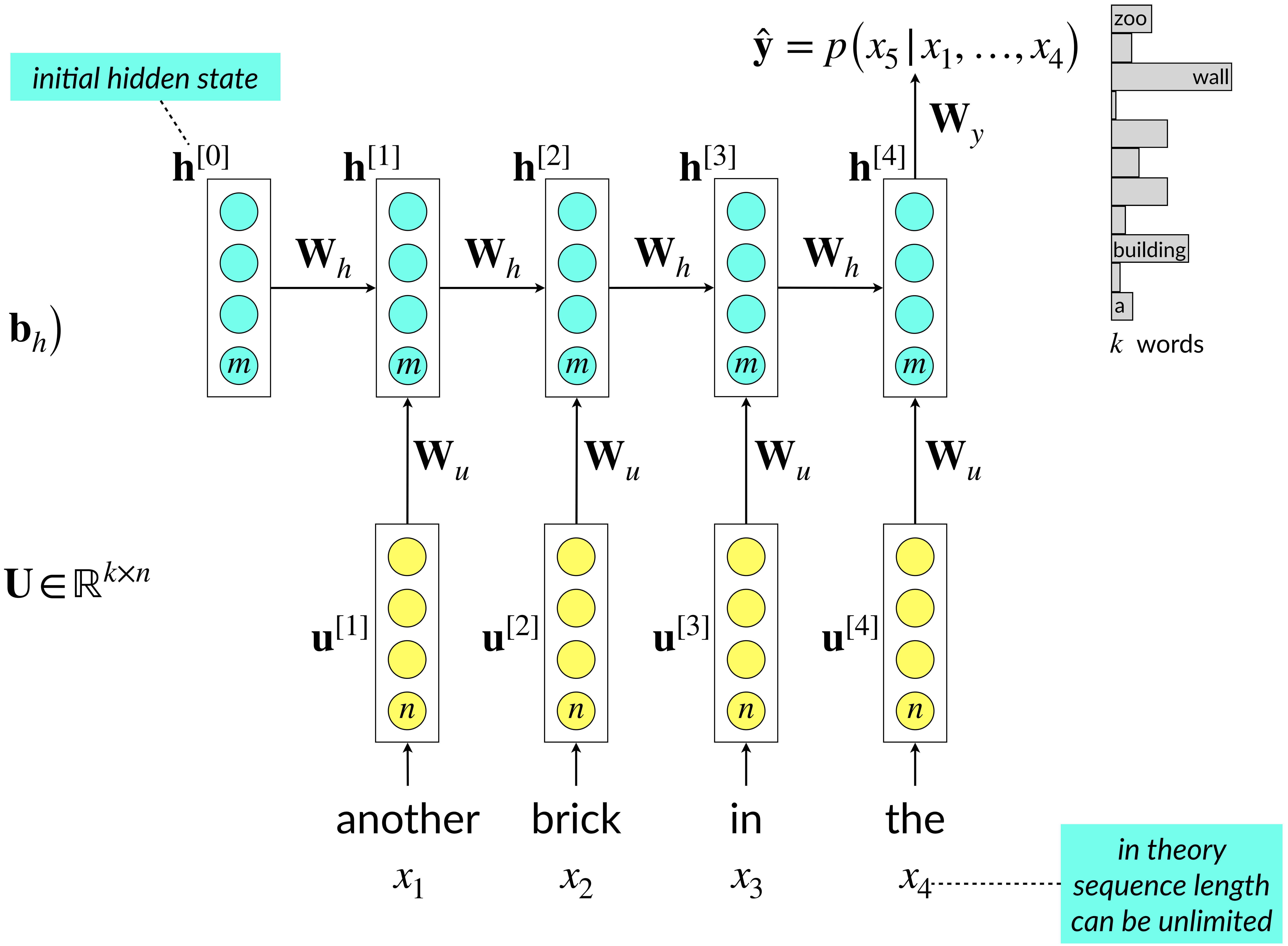
$\mathbf{u}^{[t]} \in \mathbb{R}^n$ embedding of x_t from $\mathbf{U} \in \mathbb{R}^{k \times n}$

$\mathbf{h}^{[t]}, \mathbf{b}_h \in \mathbb{R}^m$

$\hat{\mathbf{y}}, \mathbf{b}_y \in \mathbb{R}^k$

$\mathbf{W}_u \in \mathbb{R}^{m \times n}$, $\mathbf{W}_h \in \mathbb{R}^{m \times m}$

$\mathbf{W}_y \in \mathbb{R}^{k \times m}$



Output

$$\hat{\mathbf{y}} = \text{softmax}\left(\mathbf{W}_y \cdot \mathbf{h}^{[4]} + \mathbf{b}_y\right)$$

Hidden states

$$\mathbf{h}^{[t]} = \sigma\left(\mathbf{W}_u \cdot \mathbf{u}^{[t]} + \mathbf{W}_h \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_h\right)$$

or use $\tanh(\cdot)$

Dimensionalities?

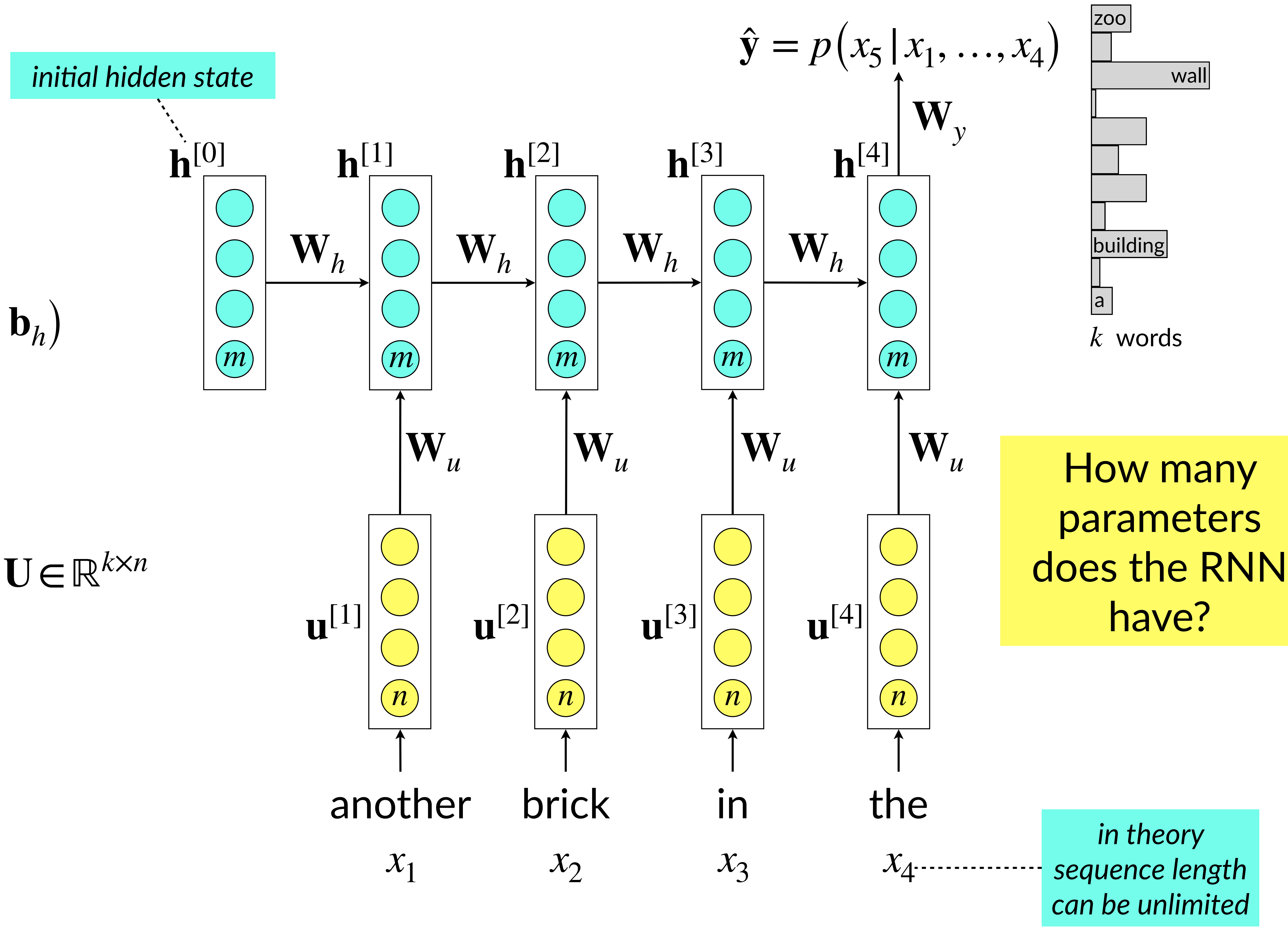
$\mathbf{u}^{[t]} \in \mathbb{R}^n$ embedding of x_t from $\mathbf{U} \in \mathbb{R}^{k \times n}$

$\mathbf{h}^{[t]}, \mathbf{b}_h \in \mathbb{R}^m$

$\hat{\mathbf{y}}, \mathbf{b}_y \in \mathbb{R}^k$

$\mathbf{W}_u \in \mathbb{R}^{m \times n}$, $\mathbf{W}_h \in \mathbb{R}^{m \times m}$

$\mathbf{W}_y \in \mathbb{R}^{k \times m}$



How many parameters does the RNN have?

An RNN-based language model

Output

$\hat{\mathbf{y}} = \text{softmax}\left(\mathbf{W}_y \cdot \mathbf{h}^{[4]} + \mathbf{b}_y\right)$

Hidden states

$\mathbf{h}^{[t]} = \sigma\left(\mathbf{W}_u \cdot \mathbf{u}^{[t]} + \mathbf{W}_h \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_h\right)$

or use $\tanh(\cdot)$

Dimensionalities?

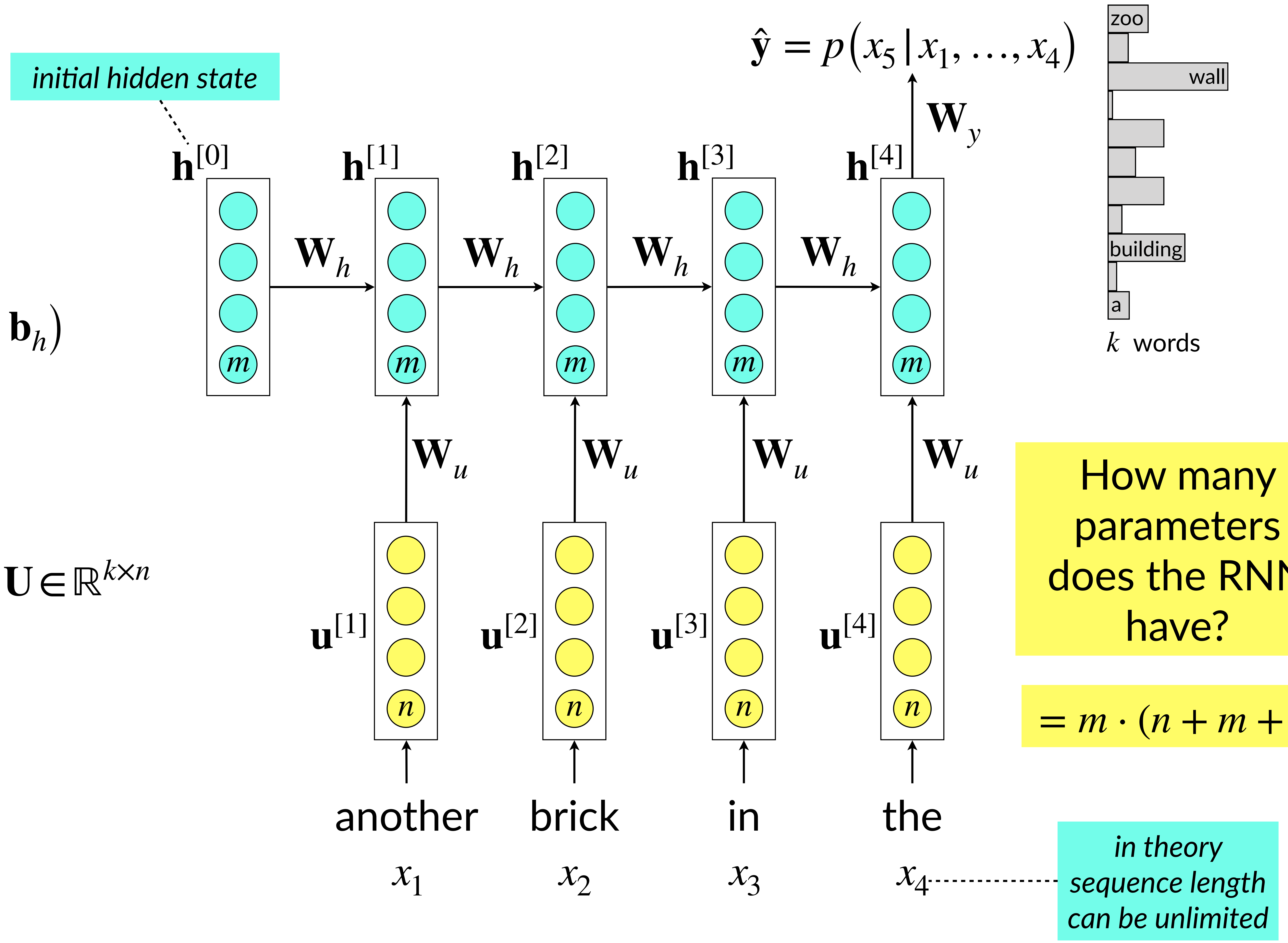
$\mathbf{u}^{[t]} \in \mathbb{R}^n$ embedding of x_t from $\mathbf{U} \in \mathbb{R}^{k \times n}$

$\mathbf{h}^{[t]}, \mathbf{b}_h \in \mathbb{R}^m$

$\hat{\mathbf{y}}, \mathbf{b}_y \in \mathbb{R}^k$

$\mathbf{W}_u \in \mathbb{R}^{m \times n}$, $\mathbf{W}_h \in \mathbb{R}^{m \times m}$

$\mathbf{W}_y \in \mathbb{R}^{k \times m}$

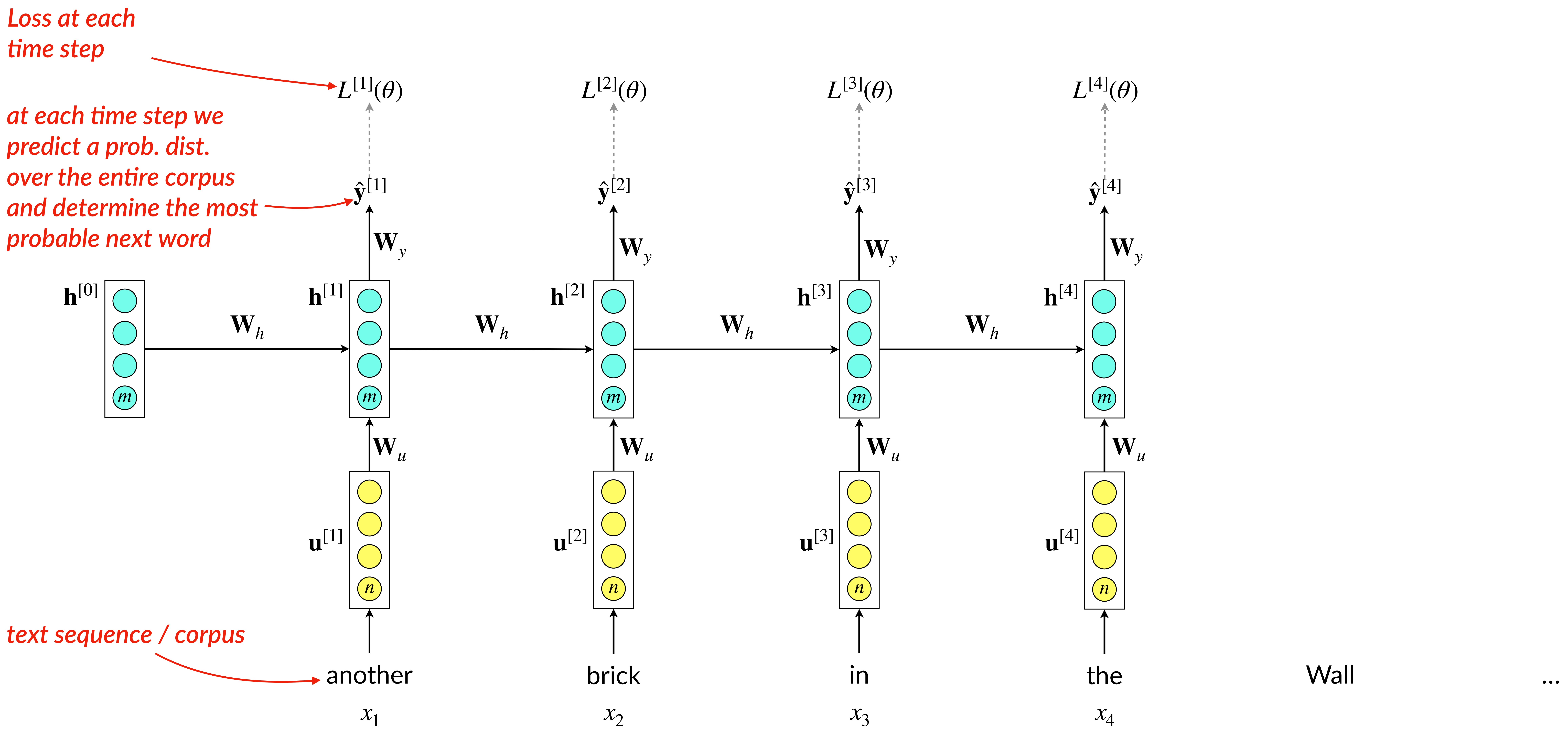


How many parameters does the RNN have?

$= m \cdot (n + m + 1)$

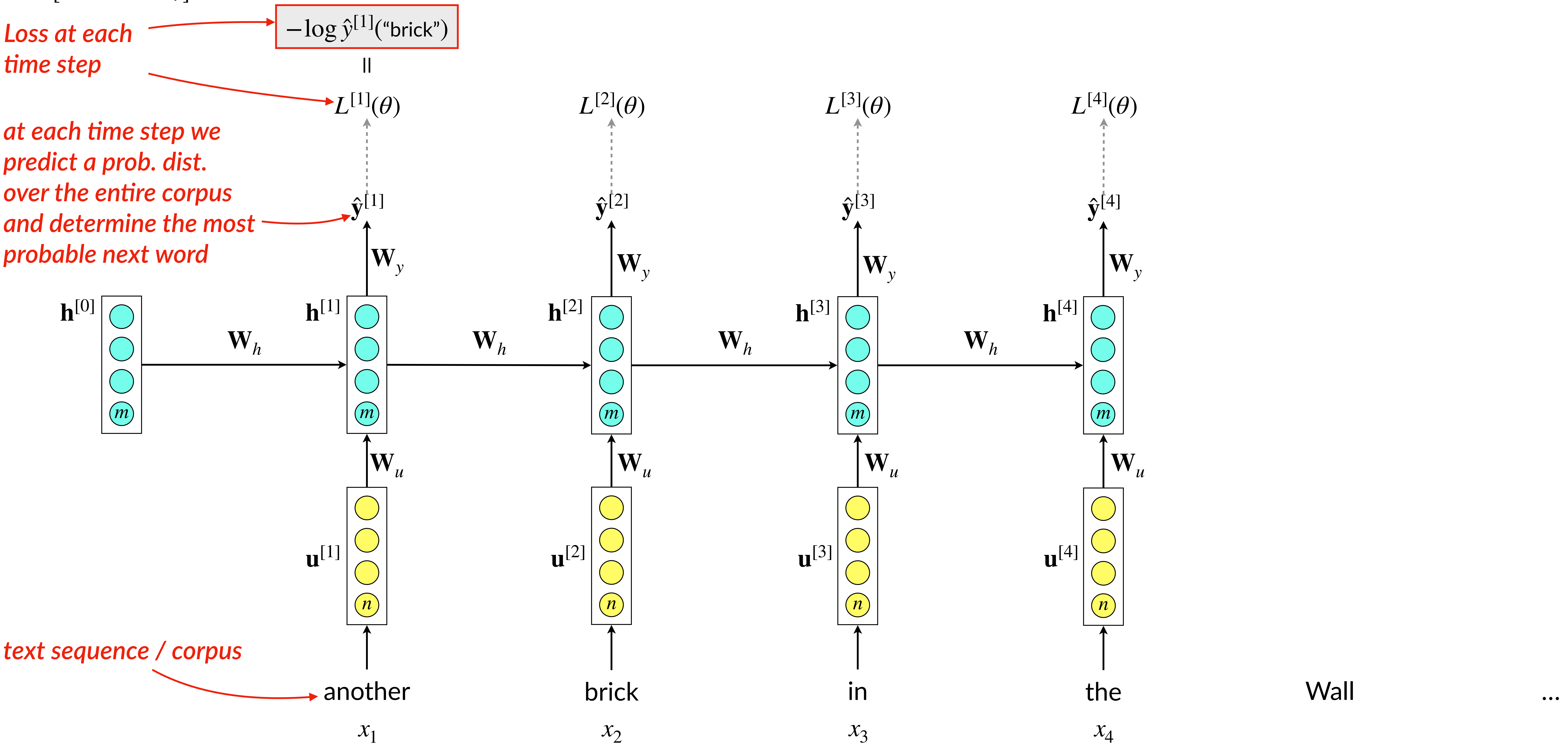
RNN training

$\theta = [\mathbf{W}_u, \mathbf{W}_h, \mathbf{W}_y]$



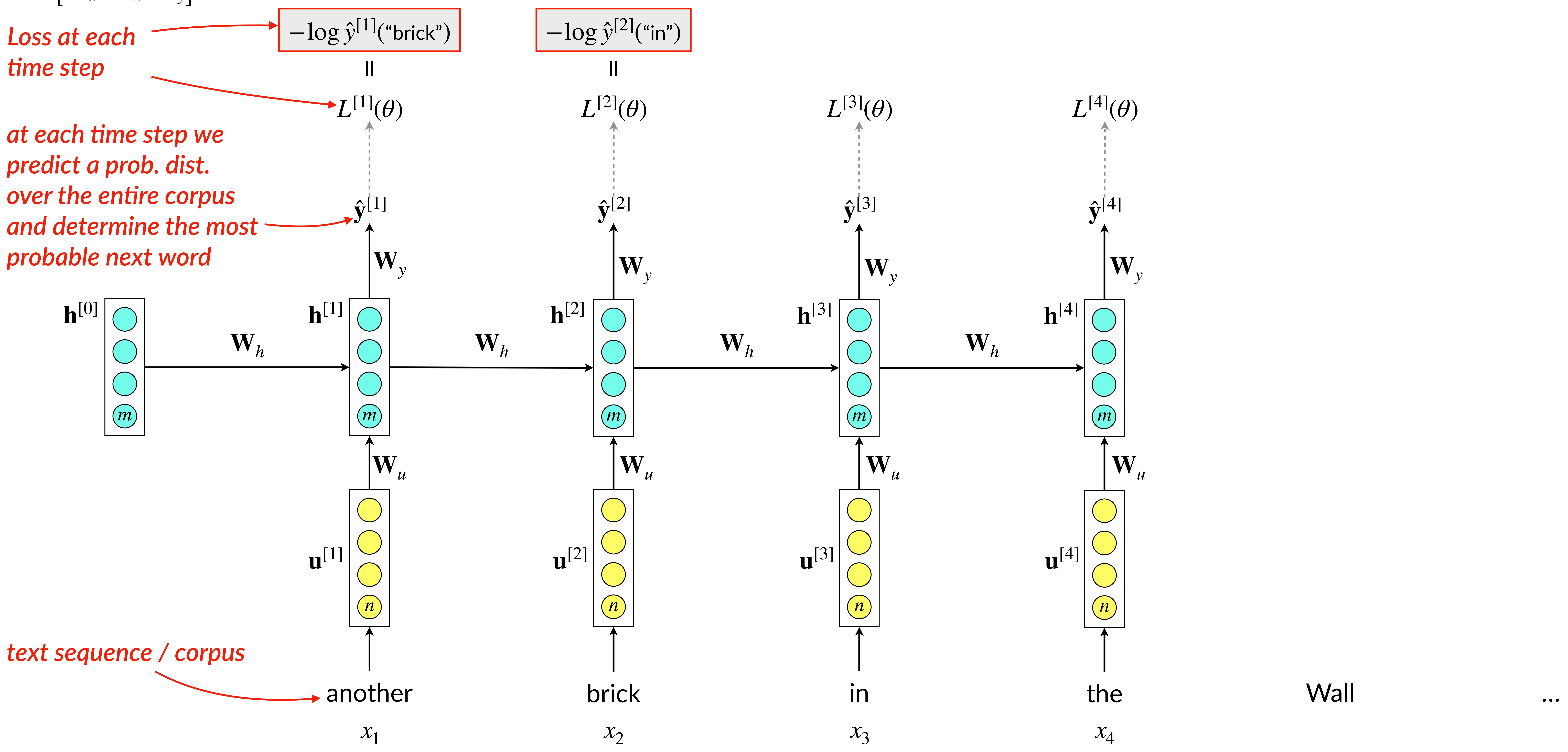
RNN training

$\theta = [\mathbf{W}_u, \mathbf{W}_h, \mathbf{W}_y]$



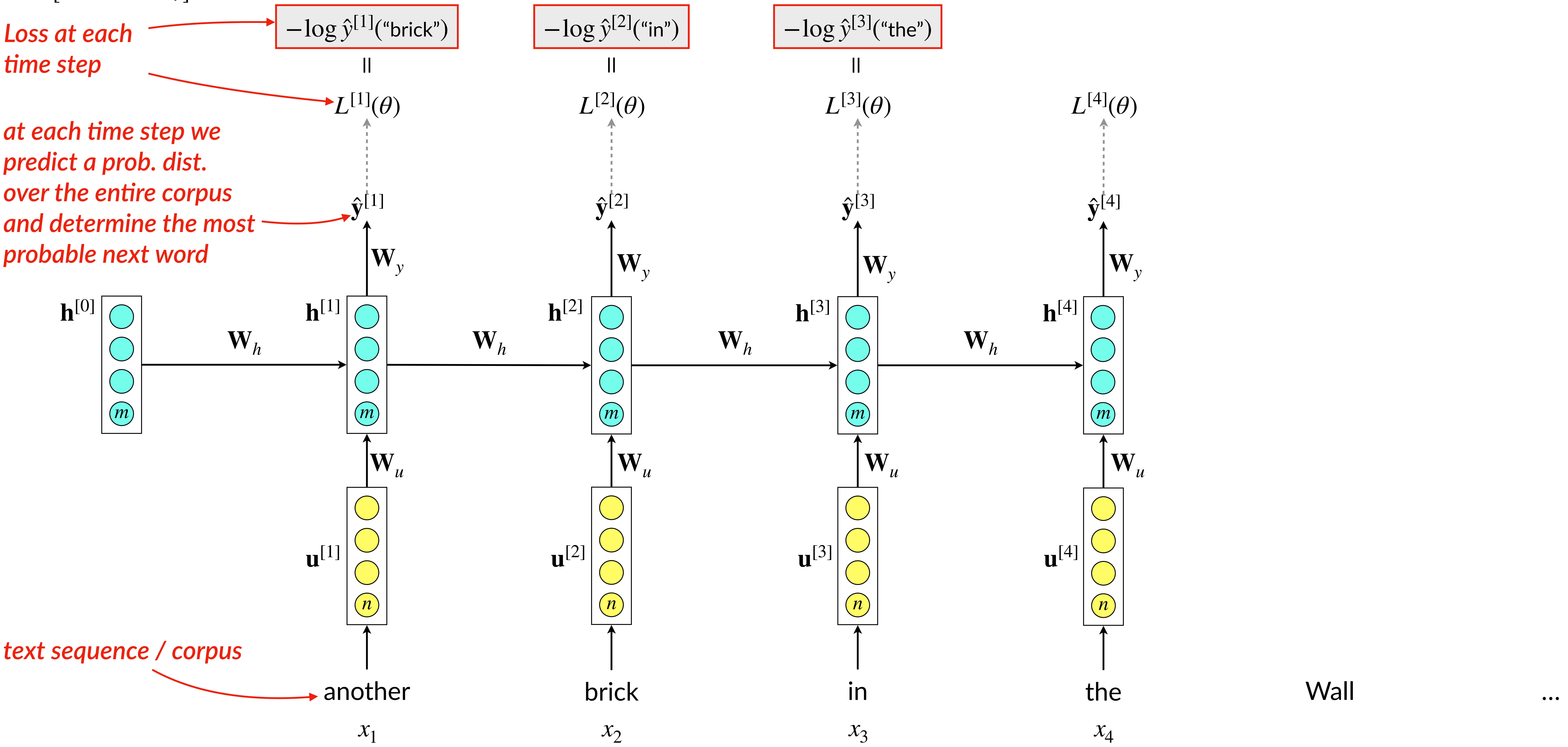
RNN training

$\theta = [\mathbf{W}_u, \mathbf{W}_h, \mathbf{W}_y]$



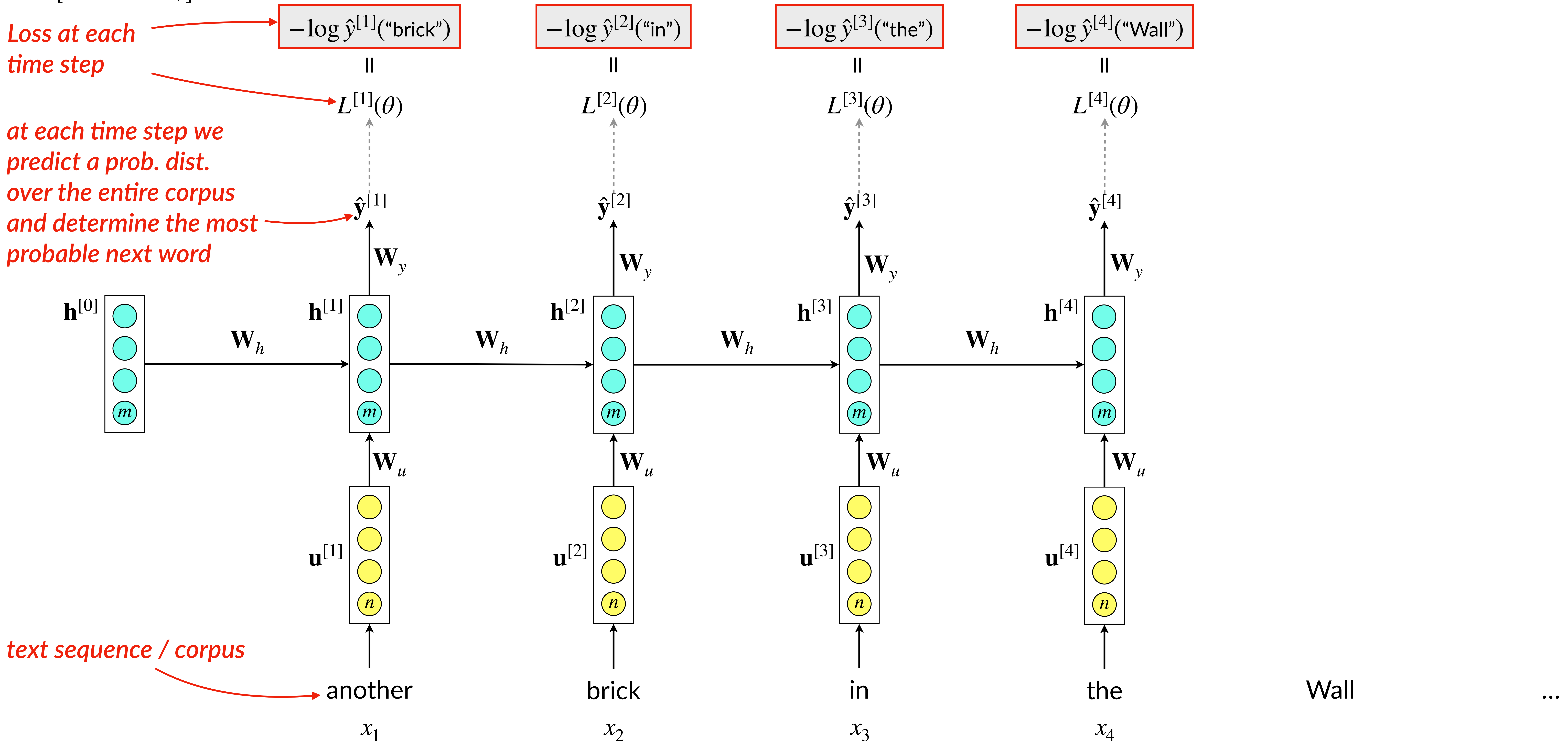
RNN training

$\theta = [\mathbf{W}_u, \mathbf{W}_h, \mathbf{W}_y]$



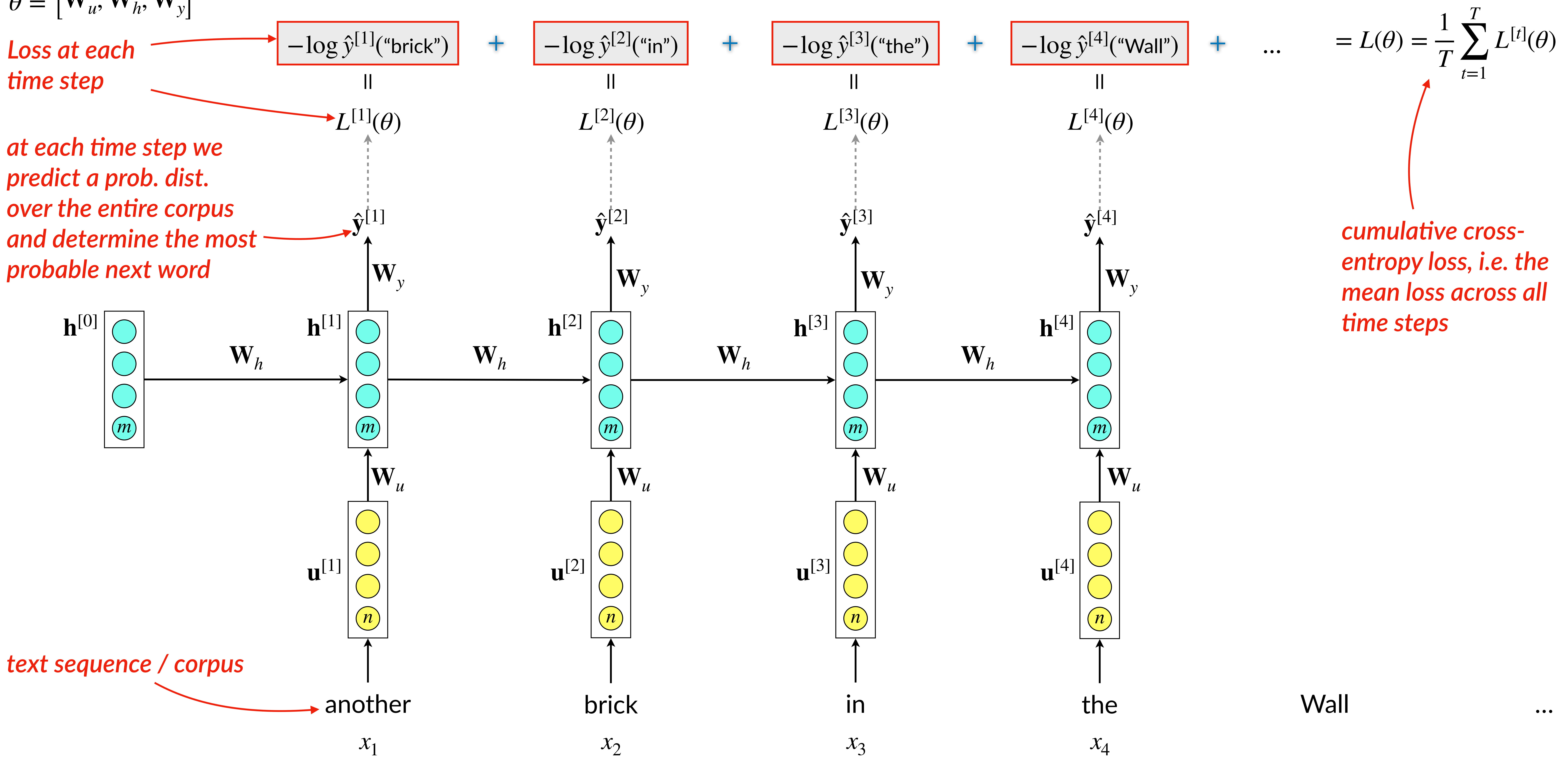
RNN training

$\theta = [\mathbf{W}_u, \mathbf{W}_h, \mathbf{W}_y]$



RNN training

$\theta = [\mathbf{W}_u, \mathbf{W}_h, \mathbf{W}_y]$



RNN training *in practice*

- The **number of tokens**, T, across a large corpus is obviously quite large!

$$L(\theta) = \frac{1}{T} \sum_{t=1}^{T} L^{[t]}(\theta)$$

- Computing $L(\theta)$ becomes too **computationally expensive**…
- Instead we (*once again*) work with a specified **window of text**, say a sentence
- We compute $L(\theta)$ for a batch of sentences, then compute the gradient of the loss with respect to the parameters of the network, and then update the parameters.
- We repeat this on a new batch until we eventually pass across the entire corpus.
- And then we go back to the beginning and repeat the entire process (*a new training epoch*), if necessary.

RNN training *in practice*

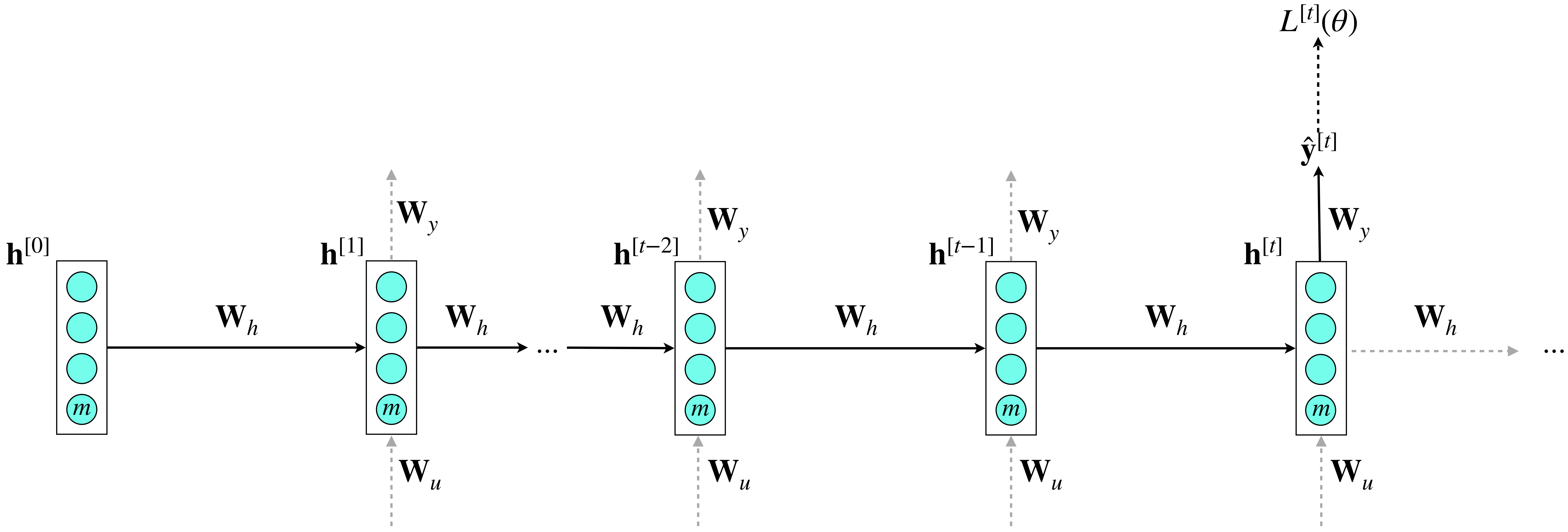
- The number of tokens, T, across a large corpus is obviously quite large!

$$L(\theta) = \frac{1}{T}\sum_{t=1}^{T} L^{[t]}(\theta)$$

how?

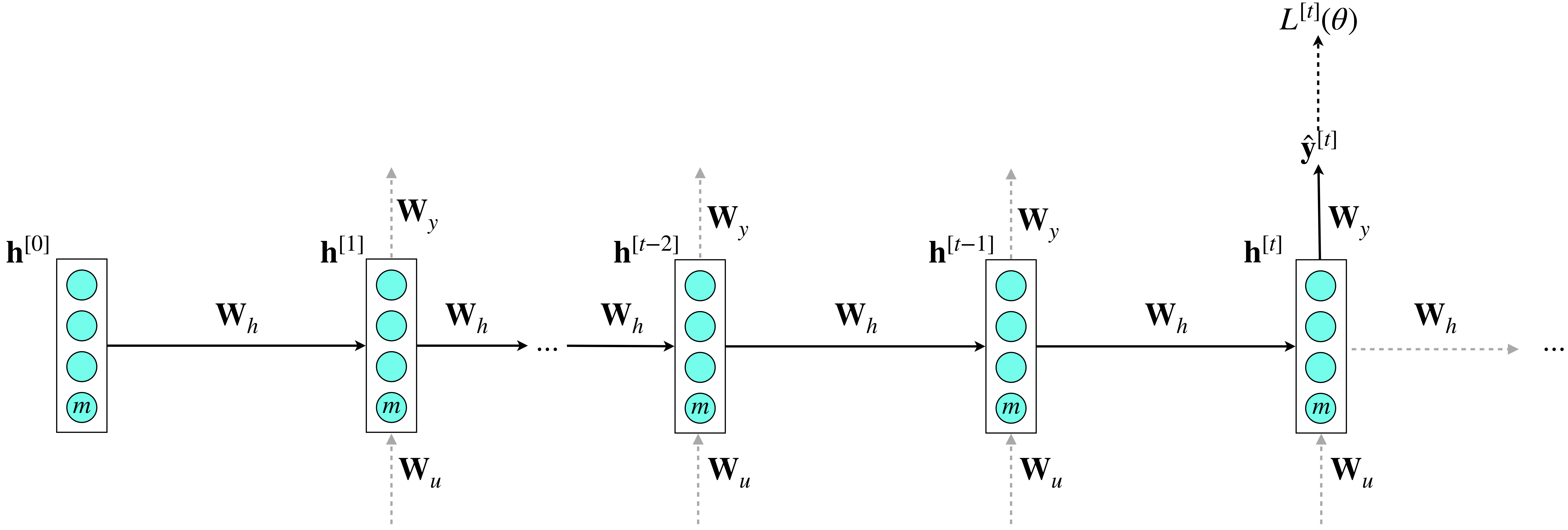
- Computing $L(\theta)$ becomes too computationally expensive...
- Instead we (*once again*) work with a specified window of text, say a sentence
- We compute $L(\theta)$ for a batch of sentences, **then compute the gradient of the loss with respect to the parameters of the network**, and then update the parameters.
- We repeat this on a new batch until we eventually pass across the entire corpus.
- And then we go back to the beginning and repeat the entire process (*a new training epoch*), if necessary.

Training the parameters of RNNs



During training, one of the derivatives we need to estimate is: $\dfrac{\partial L^{[t]}}{\partial \mathbf{W}_h}$

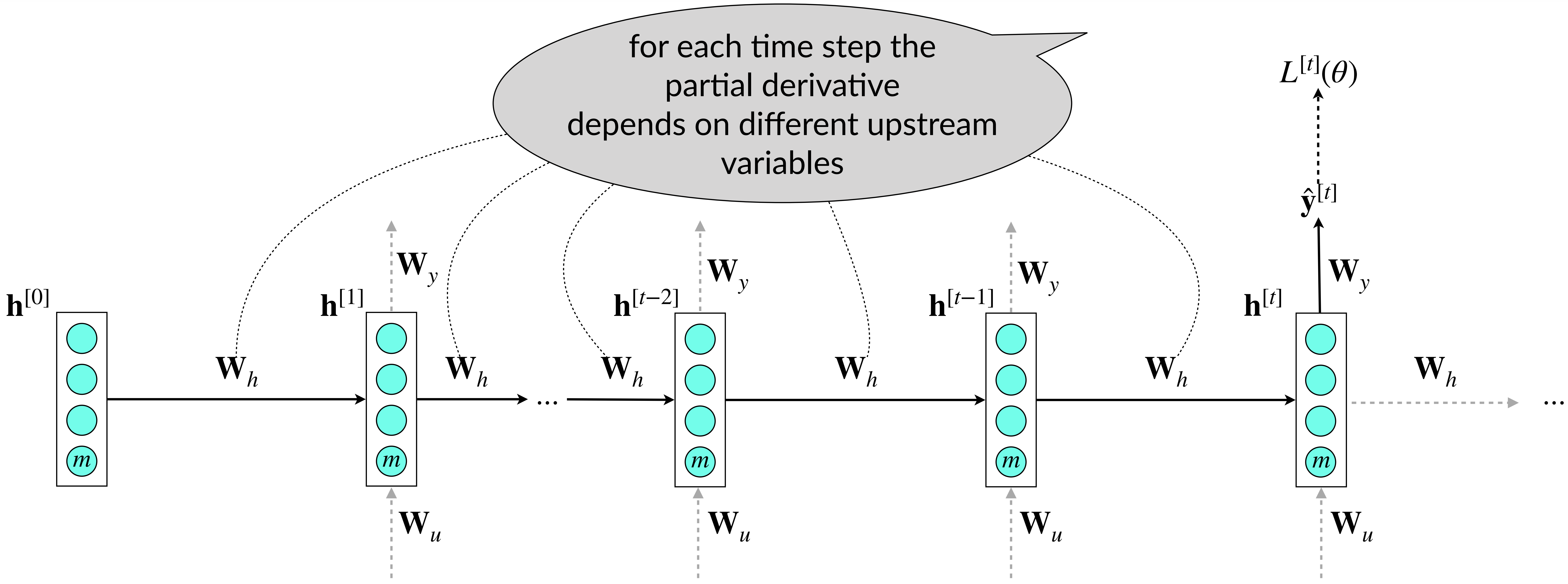
Training the parameters of RNNs



During training, one of the derivatives we need to estimate is: $\dfrac{\partial L^{[t]}}{\partial \mathbf{W}_h}$

This is given by: $\dfrac{\partial L^{[t]}}{\partial \mathbf{W}_h} = \sum_{i=1}^{t} \dfrac{\partial L^{[t]}}{\partial \mathbf{W}_h}\bigg|_{(i)}$ *we are summing up the gradients at each time step*

Training the parameters of RNNs



During training, one of the derivatives we need to estimate is: $\frac{\partial L^{[t]}}{\partial \mathbf{W}_h}$

This is given by: $$\frac{\partial L^{[t]}}{\partial \mathbf{W}_h} = \sum_{i=1}^{t} \left. \frac{\partial L^{[t]}}{\partial \mathbf{W}_h} \right|_{(i)}$$

we are summing up the gradients at each time step

Multivariable chain rule

Total derivative of a multivariable function $f\big(x(t), y(t)\big)$ that depends on two single variable functions $x(t)$ and $y(t)$

$$\frac{d}{dt} f\big(x(t), y(t)\big) = \frac{\partial f}{\partial x} \cdot \frac{dx}{dt} + \frac{\partial f}{\partial y} \cdot \frac{dy}{dt}$$

Multivariable chain rule

Total derivative of a multivariable function $f\big(x(t), y(t)\big)$ that depends on two single variable functions $x(t)$ and $y(t)$

$$\frac{d}{dt} f\big(x(t), y(t)\big) = \frac{\partial f}{\partial x} \cdot \frac{dx}{dt} + \frac{\partial f}{\partial y} \cdot \frac{dy}{dt}$$

Example

$$f(x, y) = 3x + y^2$$

$$x(t) = t^2$$

$$y(t) = t - 1$$

Multivariable chain rule

Total derivative of a multivariable function $f\big(x(t), y(t)\big)$ that depends on two single variable functions $x(t)$ and $y(t)$

$$\frac{d}{dt}f\big(x(t), y(t)\big) = \frac{\partial f}{\partial x} \cdot \frac{dx}{dt} + \frac{\partial f}{\partial y} \cdot \frac{dy}{dt}$$

Example

$$f(x, y) = 3x + y^2$$

$$x(t) = t^2$$

$$y(t) = t - 1$$

trivial solution
(*not always possible*)

$$f(x, y) = 3x(t) + y(t)^2$$

Multivariable chain rule

Total derivative of a multivariable function $f\big(x(t), y(t)\big)$ that depends on two single variable functions $x(t)$ and $y(t)$

$$\frac{d}{dt}f\big(x(t), y(t)\big) = \frac{\partial f}{\partial x} \cdot \frac{dx}{dt} + \frac{\partial f}{\partial y} \cdot \frac{dy}{dt}$$

Example

$$f(x, y) = 3x + y^2$$

$$x(t) = t^2$$

$$y(t) = t - 1$$

trivial solution
(*not always possible*)

$$f(x, y) = 3x(t) + y(t)^2$$

$$= 3t^2 + (t - 1)^2$$

Multivariable chain rule

Total derivative of a multivariable function $f\big(x(t), y(t)\big)$ that depends on two single variable functions $x(t)$ and $y(t)$

$$\frac{d}{dt}f\big(x(t), y(t)\big) = \frac{\partial f}{\partial x} \cdot \frac{dx}{dt} + \frac{\partial f}{\partial y} \cdot \frac{dy}{dt}$$

Example

$$f(x, y) = 3x + y^2$$

$$x(t) = t^2$$

$$y(t) = t - 1$$

trivial solution
(*not always possible*)

$$f(x, y) = 3x(t) + y(t)^2$$

$$= 3t^2 + (t-1)^2$$

$$= 4t^2 - 2t + 1$$

Multivariable chain rule

Total derivative of a multivariable function $f\big(x(t), y(t)\big)$ that depends on two single variable functions $x(t)$ and $y(t)$

$$\frac{d}{dt}f\big(x(t), y(t)\big) = \frac{\partial f}{\partial x} \cdot \frac{dx}{dt} + \frac{\partial f}{\partial y} \cdot \frac{dy}{dt}$$

Example

$$f(x, y) = 3x + y^2$$

$$x(t) = t^2$$

$$y(t) = t - 1$$

trivial solution
(*not always possible*)

$$f(x, y) = 3x(t) + y(t)^2$$
$$= 3t^2 + (t-1)^2$$
$$= 4t^2 - 2t + 1$$

$$\frac{df}{dt} = 8t - 2$$

Multivariable chain rule

Total derivative of a multivariable function $f\big(x(t), y(t)\big)$ that depends on two single variable functions $x(t)$ and $y(t)$

$$\frac{d}{dt} f\big(x(t), y(t)\big) = \frac{\partial f}{\partial x} \cdot \frac{dx}{dt} + \frac{\partial f}{\partial y} \cdot \frac{dy}{dt}$$

Example

$$f(x, y) = 3x + y^2$$

$$x(t) = t^2$$

$$y(t) = t - 1$$

trivial solution (*not always possible*)

$$f(x, y) = 3x(t) + y(t)^2$$

$$= 3t^2 + (t-1)^2$$

$$= 4t^2 - 2t + 1$$

$$\frac{df}{dt} = 8t - 2$$

multivariate chain rule

Multivariable chain rule

Total derivative of a multivariable function $f\big(x(t), y(t)\big)$ that depends on two single variable functions $x(t)$ and $y(t)$

$$\frac{d}{dt} f\big(x(t), y(t)\big) = \frac{\partial f}{\partial x} \cdot \frac{dx}{dt} + \frac{\partial f}{\partial y} \cdot \frac{dy}{dt}$$

Example

$$f(x, y) = 3x + y^2$$

$$x(t) = t^2$$

$$y(t) = t - 1$$

trivial solution
(*not always possible*)

$$f(x, y) = 3x(t) + y(t)^2$$
$$= 3t^2 + (t-1)^2$$
$$= 4t^2 - 2t + 1$$

$$\frac{df}{dt} = 8t - 2$$

multivariate
chain rule

$$\frac{df}{dt} =$$

Multivariable chain rule

Total derivative of a multivariable function $f\big(x(t), y(t)\big)$ that depends on two single variable functions $x(t)$ and $y(t)$

$$\frac{d}{dt} f\big(x(t), y(t)\big) = \frac{\partial f}{\partial x} \cdot \frac{dx}{dt} + \frac{\partial f}{\partial y} \cdot \frac{dy}{dt}$$

Example

$$f(x, y) = 3x + y^2$$

$$x(t) = t^2$$

$$y(t) = t - 1$$

trivial solution (*not always possible*)

$$f(x, y) = 3x(t) + y(t)^2$$

$$= 3t^2 + (t - 1)^2$$

$$= 4t^2 - 2t + 1$$

$$\frac{df}{dt} = 8t - 2$$

multivariate chain rule

$$\frac{df}{dt} = 3$$

Multivariable chain rule

Total derivative of a multivariable function $f\big(x(t), y(t)\big)$ that depends on two single variable functions $x(t)$ and $y(t)$

$$\frac{d}{dt}f\big(x(t), y(t)\big) = \frac{\partial f}{\partial x} \cdot \frac{dx}{dt} + \frac{\partial f}{\partial y} \cdot \frac{dy}{dt}$$

Example

$$f(x, y) = 3x + y^2$$

$$x(t) = t^2$$

$$y(t) = t - 1$$

trivial solution
(*not always possible*)

$$f(x, y) = 3x(t) + y(t)^2$$
$$= 3t^2 + (t-1)^2$$
$$= 4t^2 - 2t + 1$$

$$\frac{df}{dt} = 8t - 2$$

multivariate
chain rule

$$\frac{df}{dt} = 3 \cdot 2t \ +$$

Multivariable chain rule

Total derivative of a multivariable function $f\big(x(t), y(t)\big)$ that depends on two single variable functions $x(t)$ and $y(t)$

$$\frac{d}{dt}f\big(x(t), y(t)\big) = \frac{\partial f}{\partial x} \cdot \frac{dx}{dt} + \frac{\partial f}{\partial y} \cdot \frac{dy}{dt}$$

Example

$$f(x, y) = 3x + y^2$$

$$x(t) = t^2$$

$$y(t) = t - 1$$

trivial solution
(*not always possible*)

$$f(x, y) = 3x(t) + y(t)^2$$

$$= 3t^2 + (t - 1)^2$$

$$= 4t^2 - 2t + 1$$

$$\frac{df}{dt} = 8t - 2$$

multivariate chain rule

$$\frac{df}{dt} = 3 \cdot 2t \; + \; 2y$$

Multivariable chain rule

Total derivative of a multivariable function $f\big(x(t), y(t)\big)$ that depends on two single variable functions $x(t)$ and $y(t)$

$$\frac{d}{dt} f\big(x(t), y(t)\big) = \frac{\partial f}{\partial x} \cdot \frac{dx}{dt} + \frac{\partial f}{\partial y} \cdot \frac{dy}{dt}$$

Example

$$f(x, y) = 3x + y^2$$

$$x(t) = t^2$$

$$y(t) = t - 1$$

trivial solution (*not always possible*)

$$f(x, y) = 3x(t) + y(t)^2$$
$$= 3t^2 + (t - 1)^2$$
$$= 4t^2 - 2t + 1$$

$$\frac{df}{dt} = 8t - 2$$

multivariate chain rule

$$\frac{df}{dt} = 3 \cdot 2t \; + \; 2y \; \cdot 1$$

Multivariable chain rule

Total derivative of a multivariable function $f\big(x(t), y(t)\big)$ that depends on two single variable functions $x(t)$ and $y(t)$

$$\frac{d}{dt}f\big(x(t), y(t)\big) = \frac{\partial f}{\partial x} \cdot \frac{dx}{dt} + \frac{\partial f}{\partial y} \cdot \frac{dy}{dt}$$

Example

$$f(x, y) = 3x + y^2$$

$$x(t) = t^2$$

$$y(t) = t - 1$$

trivial solution
(*not always possible*)

$$f(x, y) = 3x(t) + y(t)^2$$

$$= 3t^2 + (t-1)^2$$

$$= 4t^2 - 2t + 1$$

$$\frac{df}{dt} = 8t - 2$$

multivariate chain rule

$$\frac{df}{dt} = 3 \cdot 2t + 2y \cdot 1$$

$$= 6t + 2(t-1)$$

Multivariable chain rule

Total derivative of a multivariable function $f\big(x(t), y(t)\big)$ that depends on two single variable functions $x(t)$ and $y(t)$

$$\frac{d}{dt}f\big(x(t), y(t)\big) = \frac{\partial f}{\partial x} \cdot \frac{dx}{dt} + \frac{\partial f}{\partial y} \cdot \frac{dy}{dt}$$

Example

$$f(x, y) = 3x + y^2$$

$$x(t) = t^2$$

$$y(t) = t - 1$$

trivial solution (*not always possible*)

$$f(x, y) = 3x(t) + y(t)^2$$

$$= 3t^2 + (t-1)^2$$

$$= 4t^2 - 2t + 1$$

$$\frac{df}{dt} = 8t - 2$$

multivariate chain rule

$$\frac{df}{dt} = 3 \cdot 2t + 2y \cdot 1$$

$$= 6t + 2(t-1)$$

$$= 8t - 2$$

Multivariable chain rule

Total derivative of a multivariable function $f\big(x(t), y(t)\big)$ that depends on two single variable functions $x(t)$ and $y(t)$

$$\frac{d}{dt} f\big(x(t), y(t)\big) = \frac{\partial f}{\partial x} \cdot \frac{dx}{dt} + \frac{\partial f}{\partial y} \cdot \frac{dy}{dt}$$

Example

$$f(x, y) = 3x + y^2$$

$$x(t) = t^2$$

$$y(t) = t - 1$$

helpful when a function is unknown!

trivial solution (*not always possible*)

$$f(x, y) = 3x(t) + y(t)^2$$

$$= 3t^2 + (t-1)^2$$

$$= 4t^2 - 2t + 1$$

$$\frac{df}{dt} = 8t - 2$$

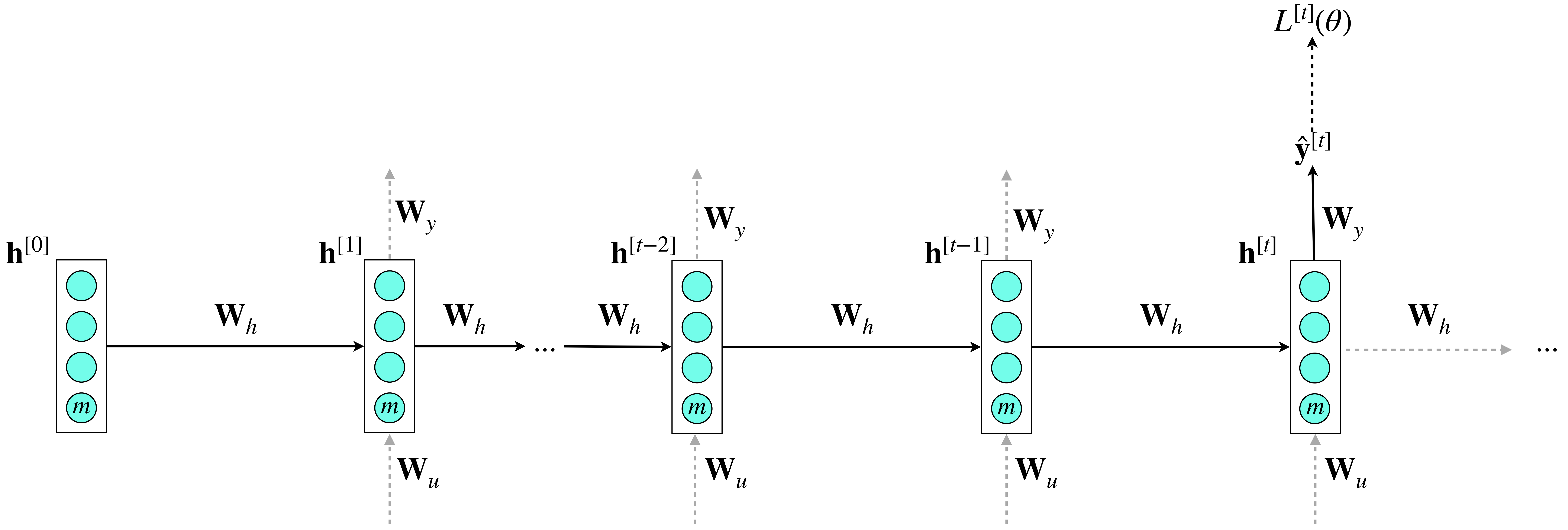
multivariate chain rule

$$\frac{df}{dt} = 3 \cdot 2t + 2y \cdot 1$$

$$= 6t + 2(t-1)$$

$$= 8t - 2$$

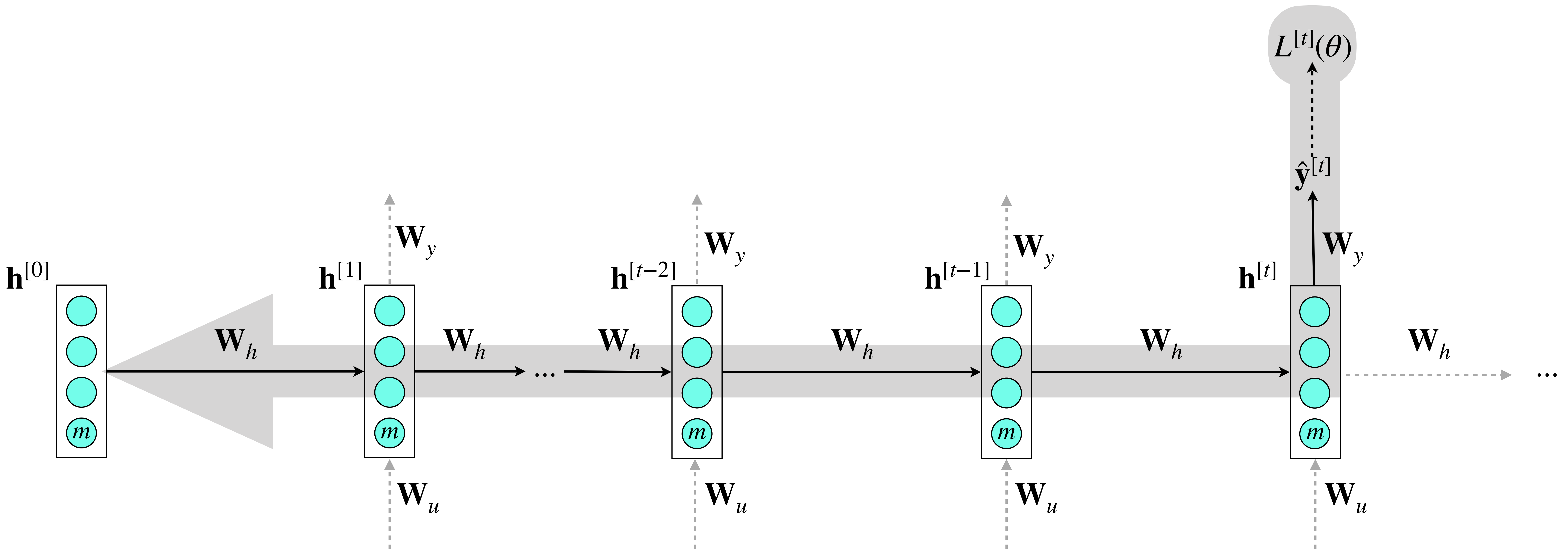
Backpropagation Through Time (BPTT)



$$\frac{\partial L^{[t]}}{\partial \mathbf{W}_h} = \sum_{i=1}^{t} \frac{\partial L^{[t]}}{\partial \mathbf{W}_h}\bigg|_{(i)}$$

backpropagation over time steps $t, t-1, \ldots, 0$, summing gradients, a.k.a. **backpropagation through time** (BPTT)

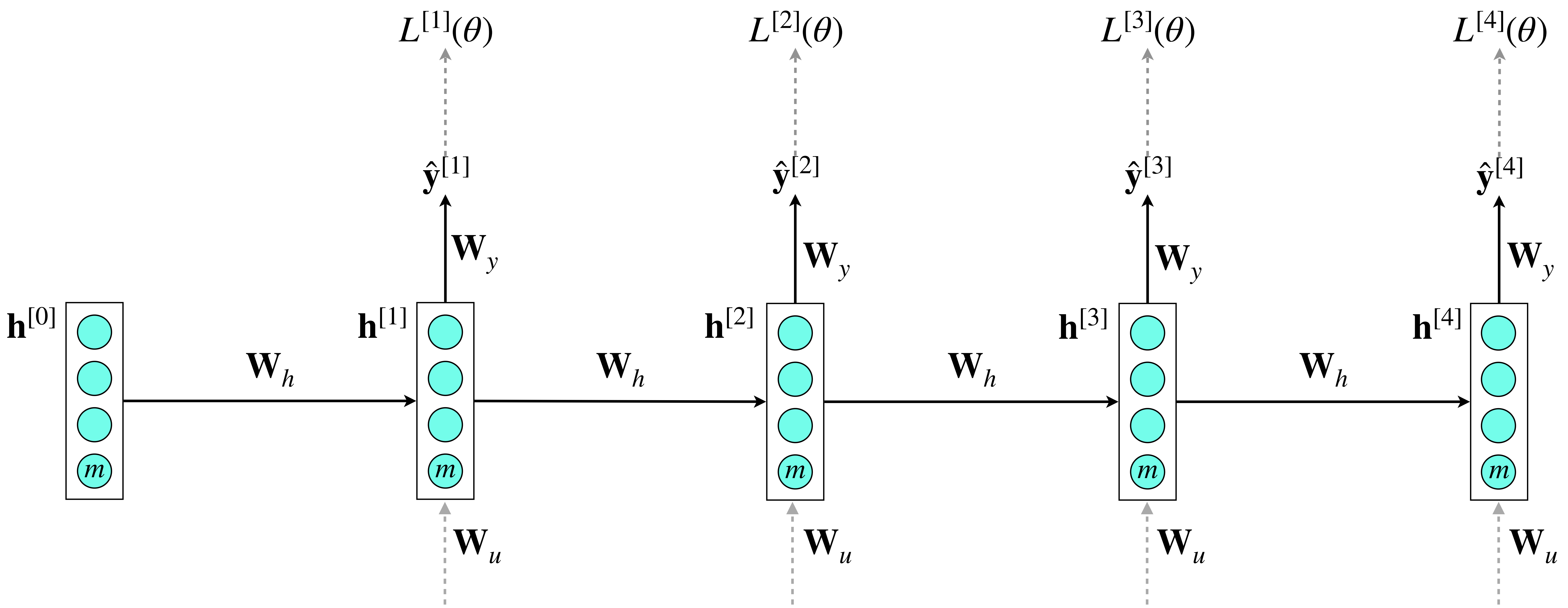
Backpropagation Through Time (BPTT)



$$\frac{\partial L^{[t]}}{\partial \mathbf{W}_h} = \sum_{i=1}^{t} \frac{\partial L^{[t]}}{\partial \mathbf{W}_h}\bigg|_{(i)}$$

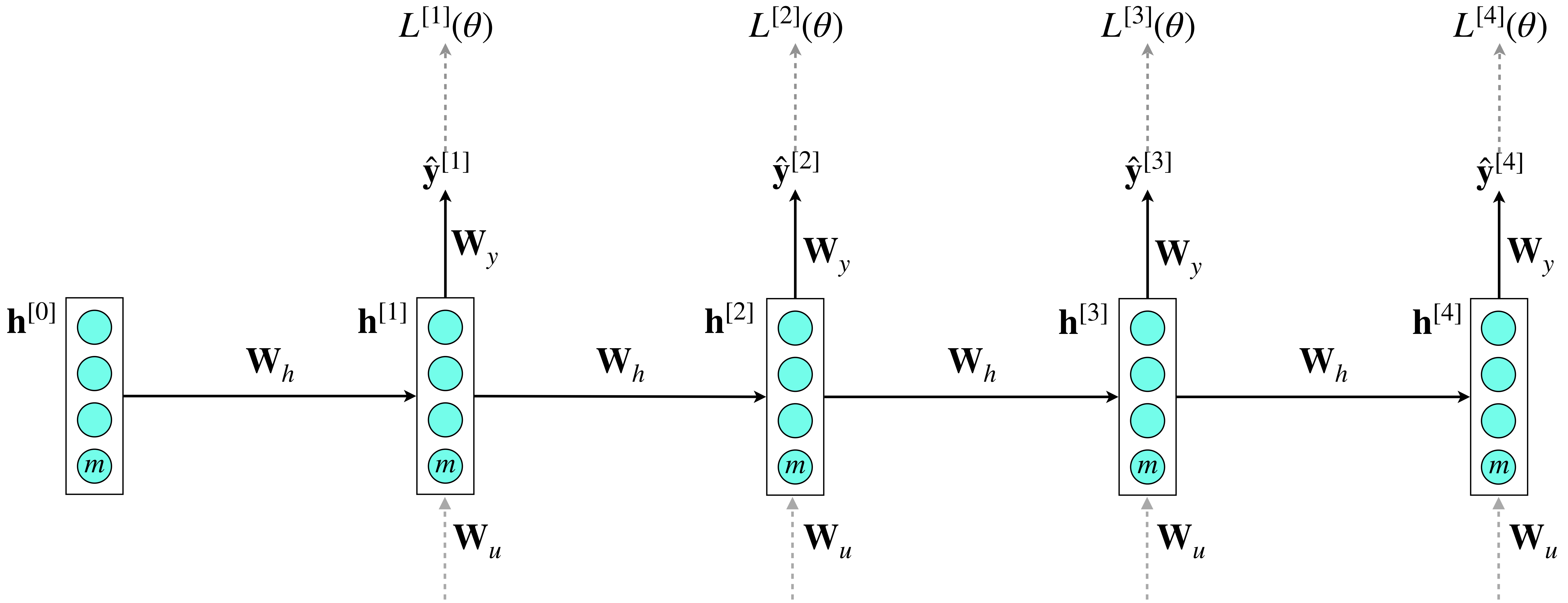
backpropagation over time steps $t, t-1, \ldots, 0$, summing gradients, a.k.a. **backpropagation through time** (BPTT)

Backpropagation through time (BPTT)



Backpropagation through time (BPTT)

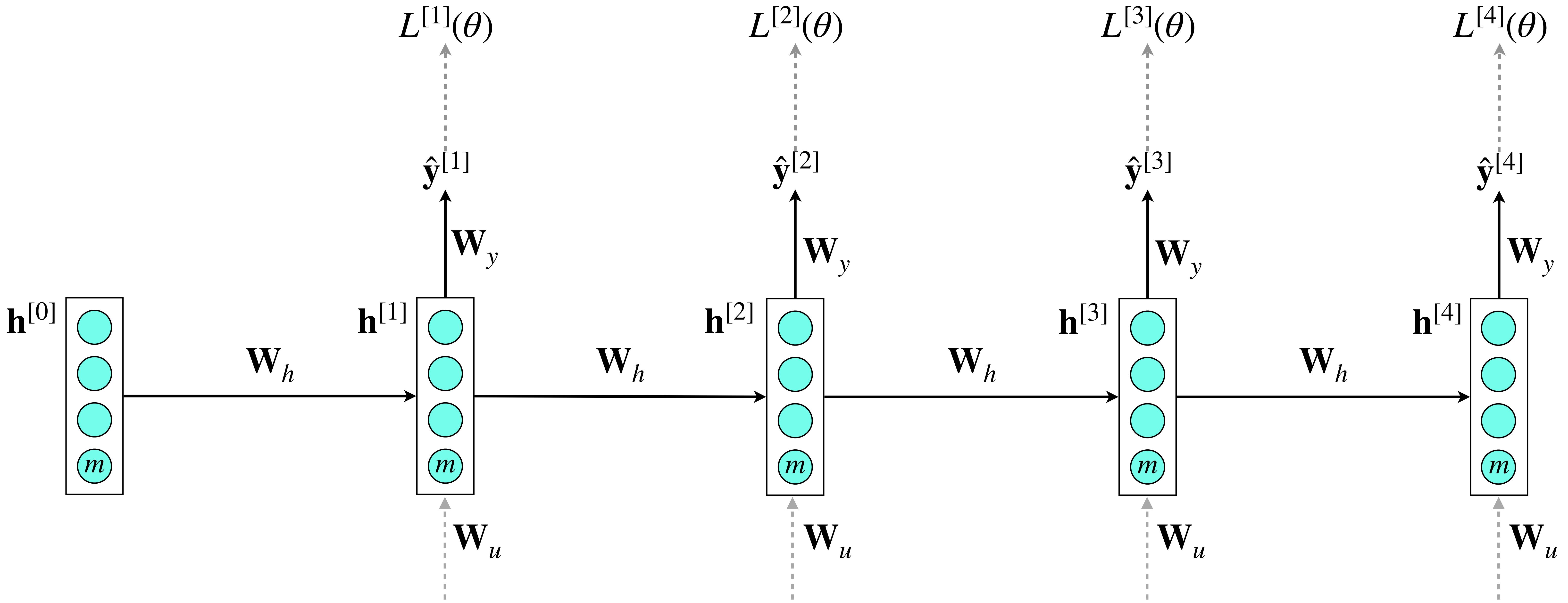
$$L(\theta) = \frac{1}{4}\sum_{t=1}^{4} L^{[t]}(\theta)$$



Backpropagation through time (BPTT)

$$L(\theta) = \frac{1}{4}\sum_{t=1}^{4} L^{[t]}(\theta)$$

$$\frac{\partial L}{\partial \mathbf{W}_h} = \sum_{t=1}^{4} \frac{\partial L^{[t]}}{\partial \mathbf{W}_h}$$

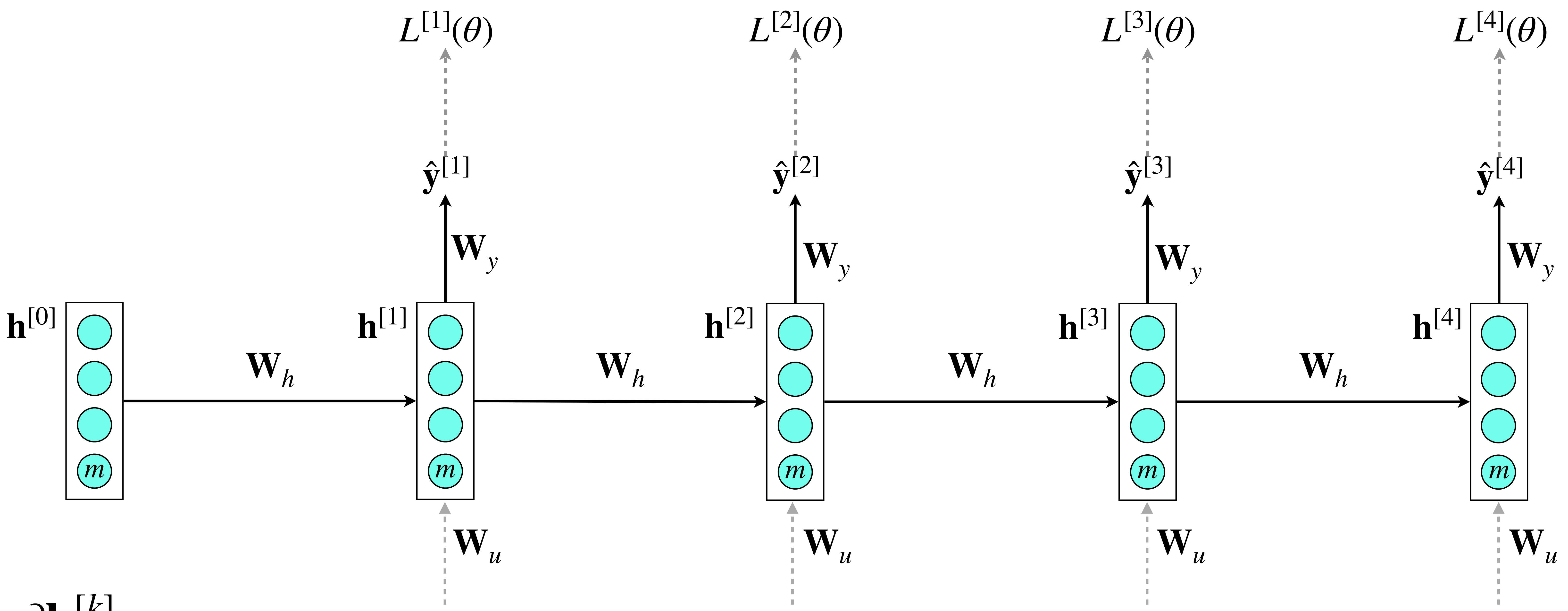


Backpropagation through time (BPTT)

$$L(\theta) = \frac{1}{4}\sum_{t=1}^{4} L^{[t]}(\theta)$$

$$\frac{\partial L}{\partial \mathbf{W}_h} = \sum_{t=1}^{4} \frac{\partial L^{[t]}}{\partial \mathbf{W}_h}$$

$$\frac{\partial L^{[t]}}{\partial \mathbf{W}_h} = \sum_{k=1}^{t} \frac{\partial L^{[t]}}{\partial \hat{\mathbf{y}}^{[t]}} \cdot \frac{\partial \hat{\mathbf{y}}^{[t]}}{\partial \mathbf{h}^{[t]}} \cdot \frac{\partial \mathbf{h}^{[t]}}{\partial \mathbf{h}^{[k]}} \cdot \frac{\partial \mathbf{h}^{[k]}}{\partial \mathbf{W}_h}$$



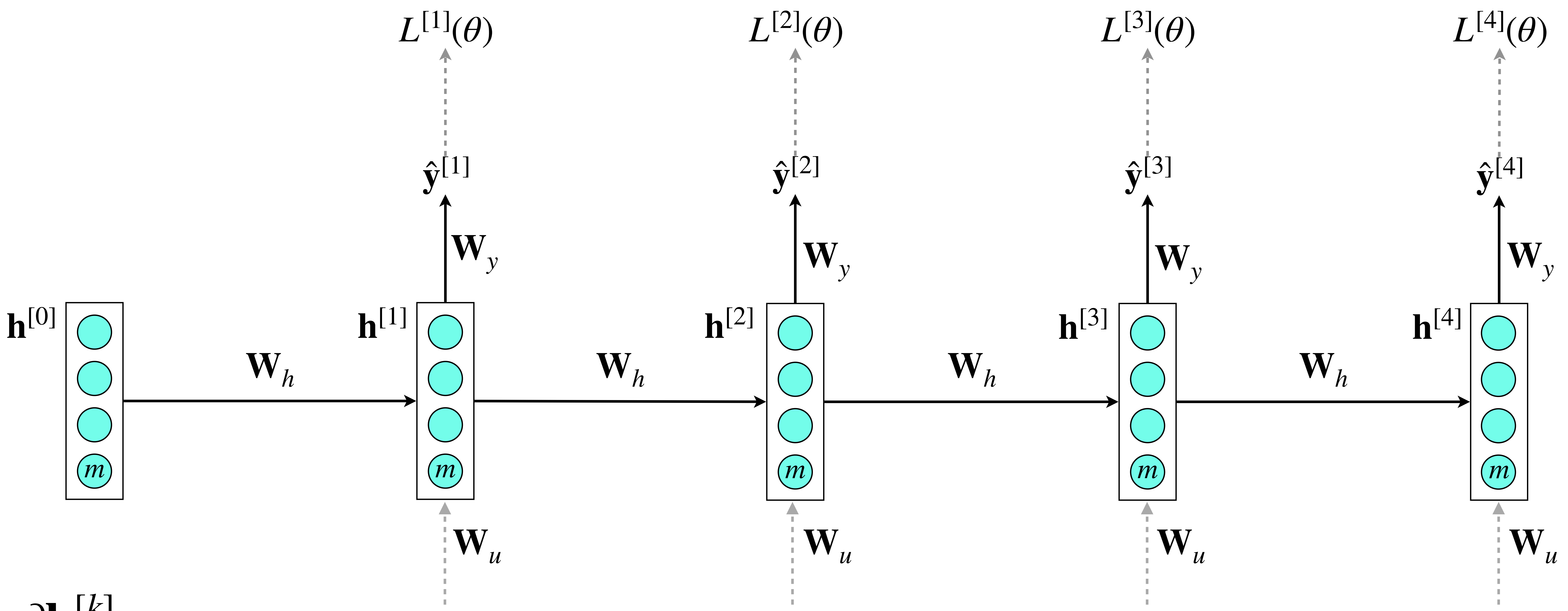
Backpropagation through time (BPTT)

$$L(\theta) = \frac{1}{4}\sum_{t=1}^{4} L^{[t]}(\theta)$$

$$\frac{\partial L}{\partial \mathbf{W}_h} = \sum_{t=1}^{4} \frac{\partial L^{[t]}}{\partial \mathbf{W}_h}$$

$$\frac{\partial L^{[t]}}{\partial \mathbf{W}_h} = \sum_{k=1}^{t} \frac{\partial L^{[t]}}{\partial \hat{\mathbf{y}}^{[t]}} \cdot \frac{\partial \hat{\mathbf{y}}^{[t]}}{\partial \mathbf{h}^{[t]}} \cdot \frac{\partial \mathbf{h}^{[t]}}{\partial \mathbf{h}^{[k]}} \cdot \frac{\partial \mathbf{h}^{[k]}}{\partial \mathbf{W}_h}$$

$$\frac{\partial \mathbf{h}^{[t]}}{\partial \mathbf{h}^{[k]}} = \prod_{j=k+1}^{t} \frac{\partial \mathbf{h}^{[j]}}{\partial \mathbf{h}^{[j-1]}}$$



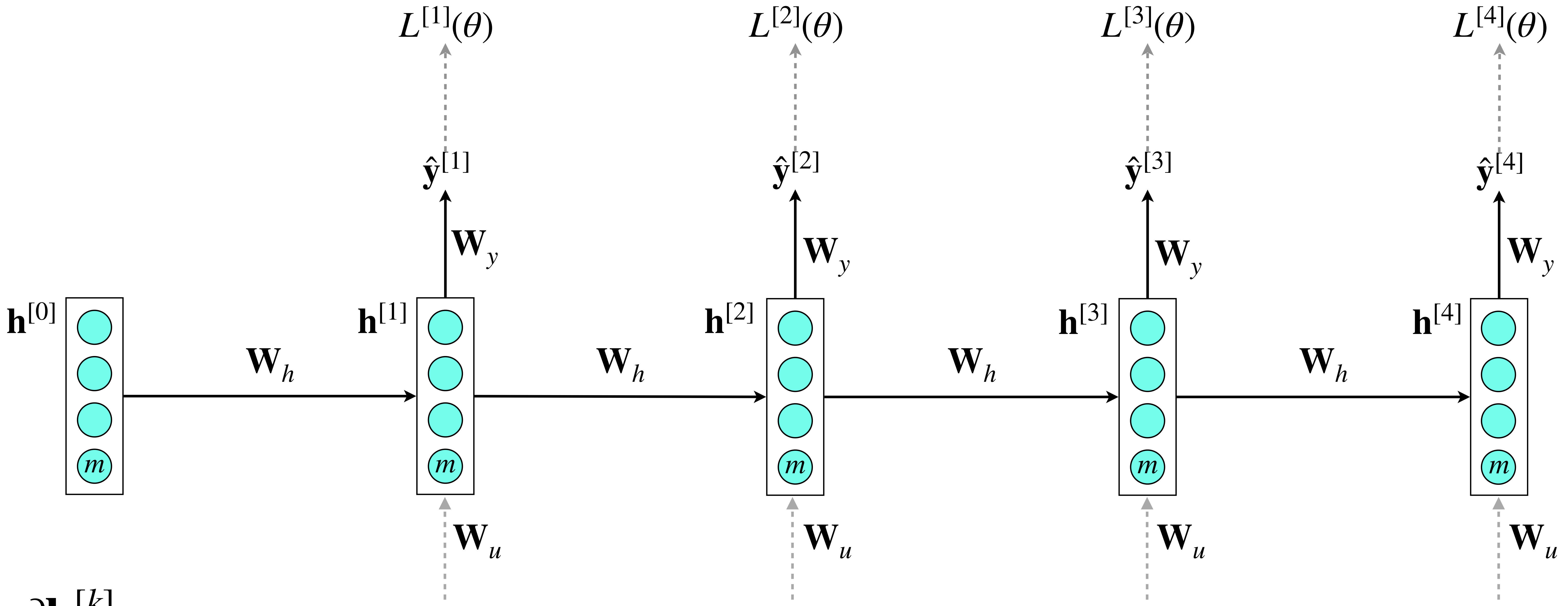
Backpropagation through time (BPTT)

$$L(\theta) = \frac{1}{4}\sum_{t=1}^{4} L^{[t]}(\theta)$$

$$\frac{\partial L}{\partial \mathbf{W}_h} = \sum_{t=1}^{4} \frac{\partial L^{[t]}}{\partial \mathbf{W}_h}$$

$$\frac{\partial L^{[t]}}{\partial \mathbf{W}_h} = \sum_{k=1}^{t} \frac{\partial L^{[t]}}{\partial \hat{\mathbf{y}}^{[t]}} \cdot \frac{\partial \hat{\mathbf{y}}^{[t]}}{\partial \mathbf{h}^{[t]}} \cdot \frac{\partial \mathbf{h}^{[t]}}{\partial \mathbf{h}^{[k]}} \cdot \frac{\partial \mathbf{h}^{[k]}}{\partial \mathbf{W}_h}$$

$$\frac{\partial \mathbf{h}^{[t]}}{\partial \mathbf{h}^{[k]}} = \prod_{j=k+1}^{t} \frac{\partial \mathbf{h}^{[j]}}{\partial \mathbf{h}^{[j-1]}}$$



e.g. if $t = 4$ and $k = 1$

$$\frac{\partial \mathbf{h}^{[4]}}{\partial \mathbf{h}^{[1]}} = \prod_{j=2}^{4} \frac{\partial \mathbf{h}^{[j]}}{\partial \mathbf{h}^{[j-1]}} = \frac{\partial \mathbf{h}^{[2]}}{\partial \mathbf{h}^{[1]}} \cdot \frac{\partial \mathbf{h}^{[3]}}{\partial \mathbf{h}^{[2]}} \cdot \frac{\partial \mathbf{h}^{[4]}}{\partial \mathbf{h}^{[3]}}$$

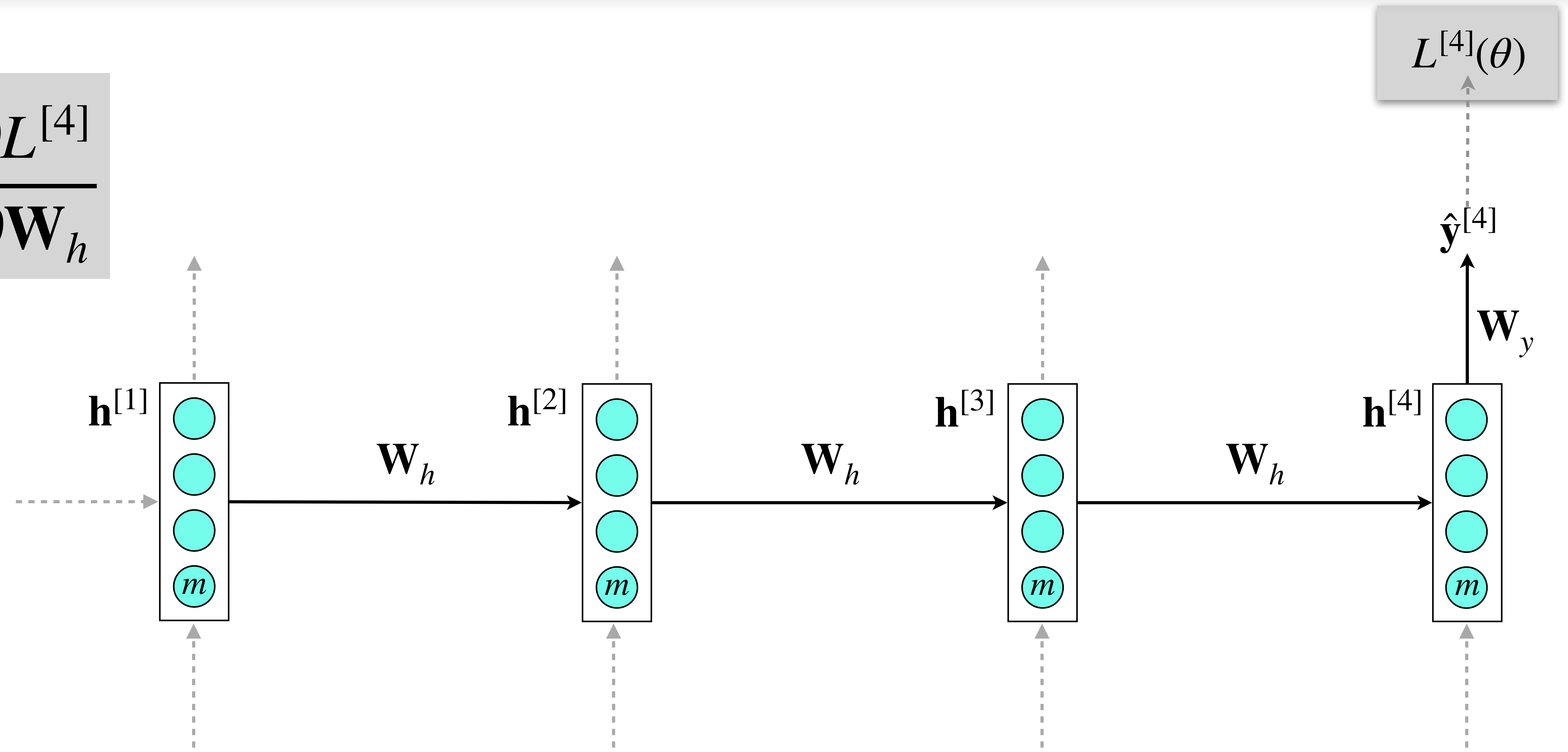
Vanishing (*or exploding*) gradients

$$\frac{\partial L}{\partial \mathbf{W}_h} = \sum_{t=1}^{4} \frac{\partial L^{[t]}}{\partial \mathbf{W}_h} = \frac{\partial L^{[1]}}{\partial \mathbf{W}_h} + \frac{\partial L^{[2]}}{\partial \mathbf{W}_h} + \frac{\partial L^{[3]}}{\partial \mathbf{W}_h} + \frac{\partial L^{[4]}}{\partial \mathbf{W}_h}$$

$$\frac{\partial L^{[4]}}{\partial \mathbf{W}_h} = \sum_{k=1}^{4} \frac{\partial L^{[4]}}{\partial \hat{\mathbf{y}}^{[4]}} \cdot \frac{\partial \hat{\mathbf{y}}^{[4]}}{\partial \mathbf{h}^{[4]}} \cdot \frac{\partial \mathbf{h}^{[4]}}{\partial \mathbf{h}^{[k]}} \cdot \frac{\partial \mathbf{h}^{[k]}}{\partial \mathbf{W}_h}$$

let's focus on this component of the sum

$$\propto \frac{\partial \mathbf{h}^{[4]}}{\partial \mathbf{h}^{[1]}} + \frac{\partial \mathbf{h}^{[4]}}{\partial \mathbf{h}^{[2]}} + \frac{\partial \mathbf{h}^{[4]}}{\partial \mathbf{h}^{[3]}} + \frac{\partial \mathbf{h}^{[4]}}{\partial \mathbf{h}^{[4]}}$$



Vanishing (*or exploding*) gradients

$$\frac{\partial L}{\partial \mathbf{W}_h} = \sum_{t=1}^{4} \frac{\partial L^{[t]}}{\partial \mathbf{W}_h} = \frac{\partial L^{[1]}}{\partial \mathbf{W}_h} + \frac{\partial L^{[2]}}{\partial \mathbf{W}_h} + \frac{\partial L^{[3]}}{\partial \mathbf{W}_h} + \frac{\partial L^{[4]}}{\partial \mathbf{W}_h}$$

$$\frac{\partial L^{[4]}}{\partial \mathbf{W}_h} = \sum_{k=1}^{4} \frac{\partial L^{[4]}}{\partial \hat{\mathbf{y}}^{[4]}} \cdot \frac{\partial \hat{\mathbf{y}}^{[4]}}{\partial \mathbf{h}^{[4]}} \cdot \frac{\partial \mathbf{h}^{[4]}}{\partial \mathbf{h}^{[k]}} \cdot \frac{\partial \mathbf{h}^{[k]}}{\partial \mathbf{W}_h}$$

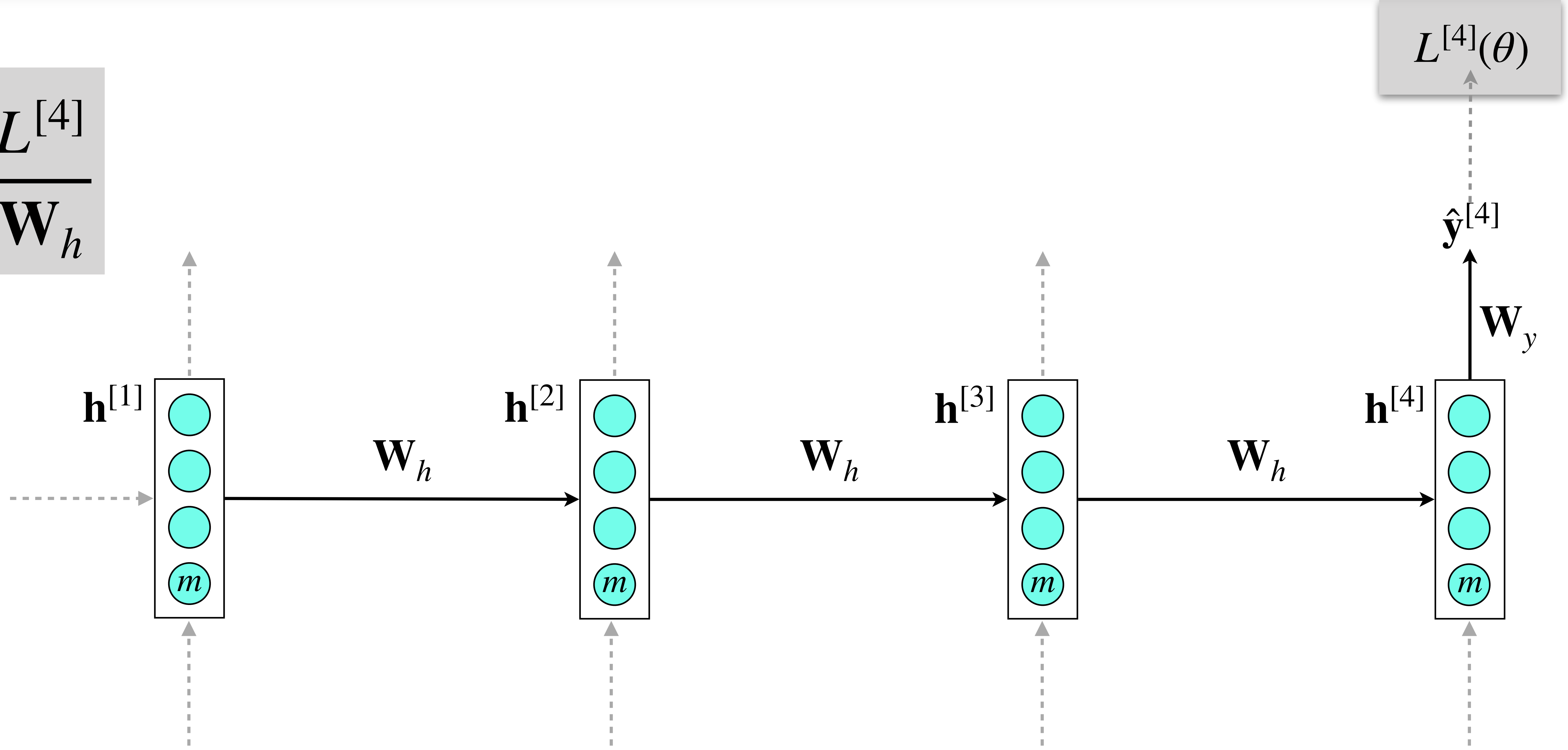
let's focus on this component of the sum

$$\propto \frac{\partial \mathbf{h}^{[4]}}{\partial \mathbf{h}^{[1]}} + \frac{\partial \mathbf{h}^{[4]}}{\partial \mathbf{h}^{[2]}} + \frac{\partial \mathbf{h}^{[4]}}{\partial \mathbf{h}^{[3]}} + \frac{\partial \mathbf{h}^{[4]}}{\partial \mathbf{h}^{[4]}}$$

$$\propto \sum_{k=1}^{4} \prod_{j=k+1}^{4} \frac{\partial \mathbf{h}^{[j]}}{\partial \mathbf{h}^{[j-1]}}$$

recall

$$\frac{\partial \mathbf{h}^{[4]}}{\partial \mathbf{h}^{[k]}} = \prod_{j=k+1}^{4} \frac{\partial \mathbf{h}^{[j]}}{\partial \mathbf{h}^{[j-1]}}$$



Vanishing (*or exploding*) gradients

$$\frac{\partial L}{\partial \mathbf{W}_h} = \sum_{t=1}^{4} \frac{\partial L^{[t]}}{\partial \mathbf{W}_h} = \frac{\partial L^{[1]}}{\partial \mathbf{W}_h} + \frac{\partial L^{[2]}}{\partial \mathbf{W}_h} + \frac{\partial L^{[3]}}{\partial \mathbf{W}_h} + \frac{\partial L^{[4]}}{\partial \mathbf{W}_h}$$

$$\frac{\partial L^{[4]}}{\partial \mathbf{W}_h} = \sum_{k=1}^{4} \frac{\partial L^{[4]}}{\partial \hat{\mathbf{y}}^{[4]}} \cdot \frac{\partial \hat{\mathbf{y}}^{[4]}}{\partial \mathbf{h}^{[4]}} \cdot \frac{\partial \mathbf{h}^{[4]}}{\partial \mathbf{h}^{[k]}} \cdot \frac{\partial \mathbf{h}^{[k]}}{\partial \mathbf{W}_h}$$

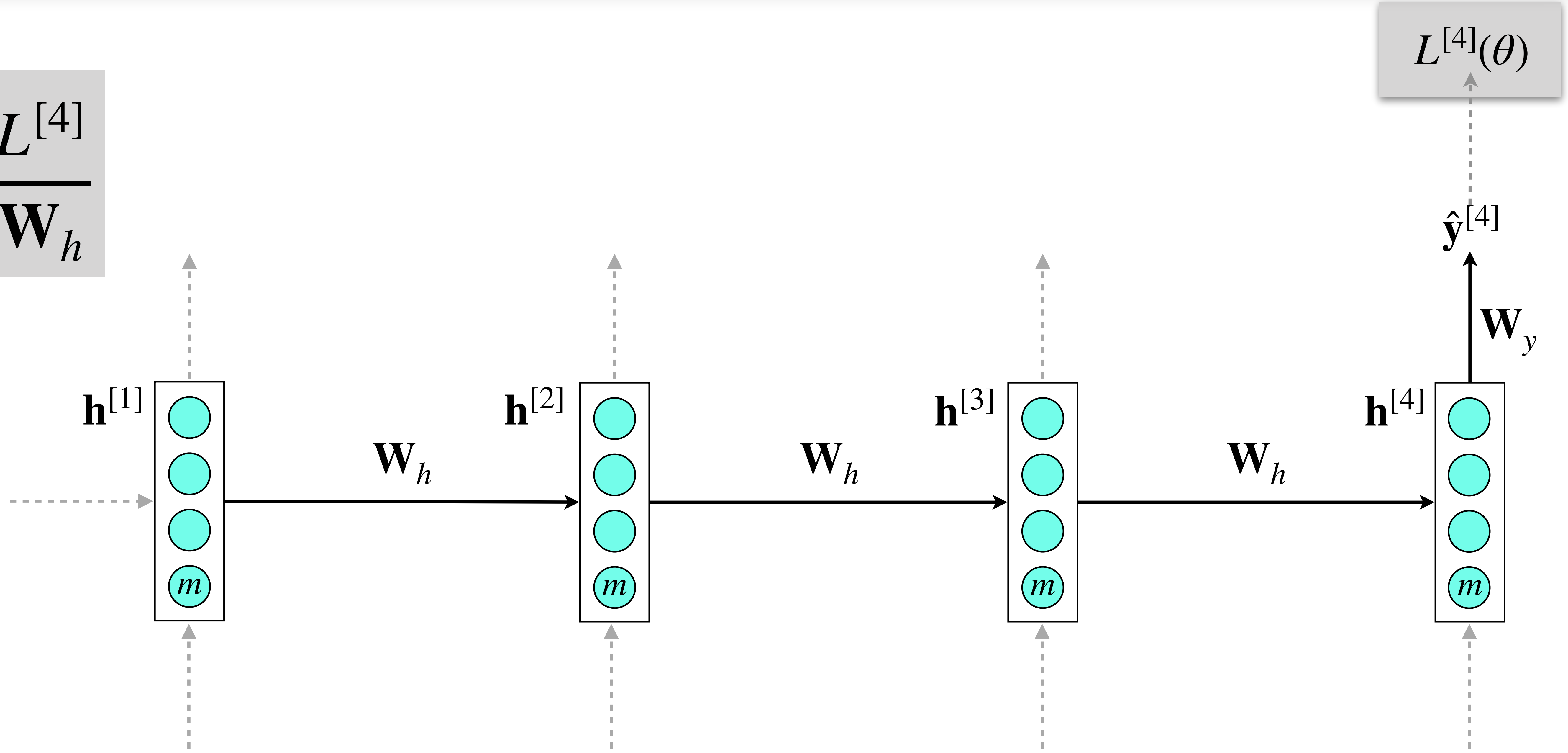
let's focus on this component of the sum

$$\propto \frac{\partial \mathbf{h}^{[4]}}{\partial \mathbf{h}^{[1]}} + \frac{\partial \mathbf{h}^{[4]}}{\partial \mathbf{h}^{[2]}} + \frac{\partial \mathbf{h}^{[4]}}{\partial \mathbf{h}^{[3]}} + \frac{\partial \mathbf{h}^{[4]}}{\partial \mathbf{h}^{[4]}}$$

$$\propto \sum_{k=1}^{4} \prod_{j=k+1}^{4} \frac{\partial \mathbf{h}^{[j]}}{\partial \mathbf{h}^{[j-1]}}$$

recall

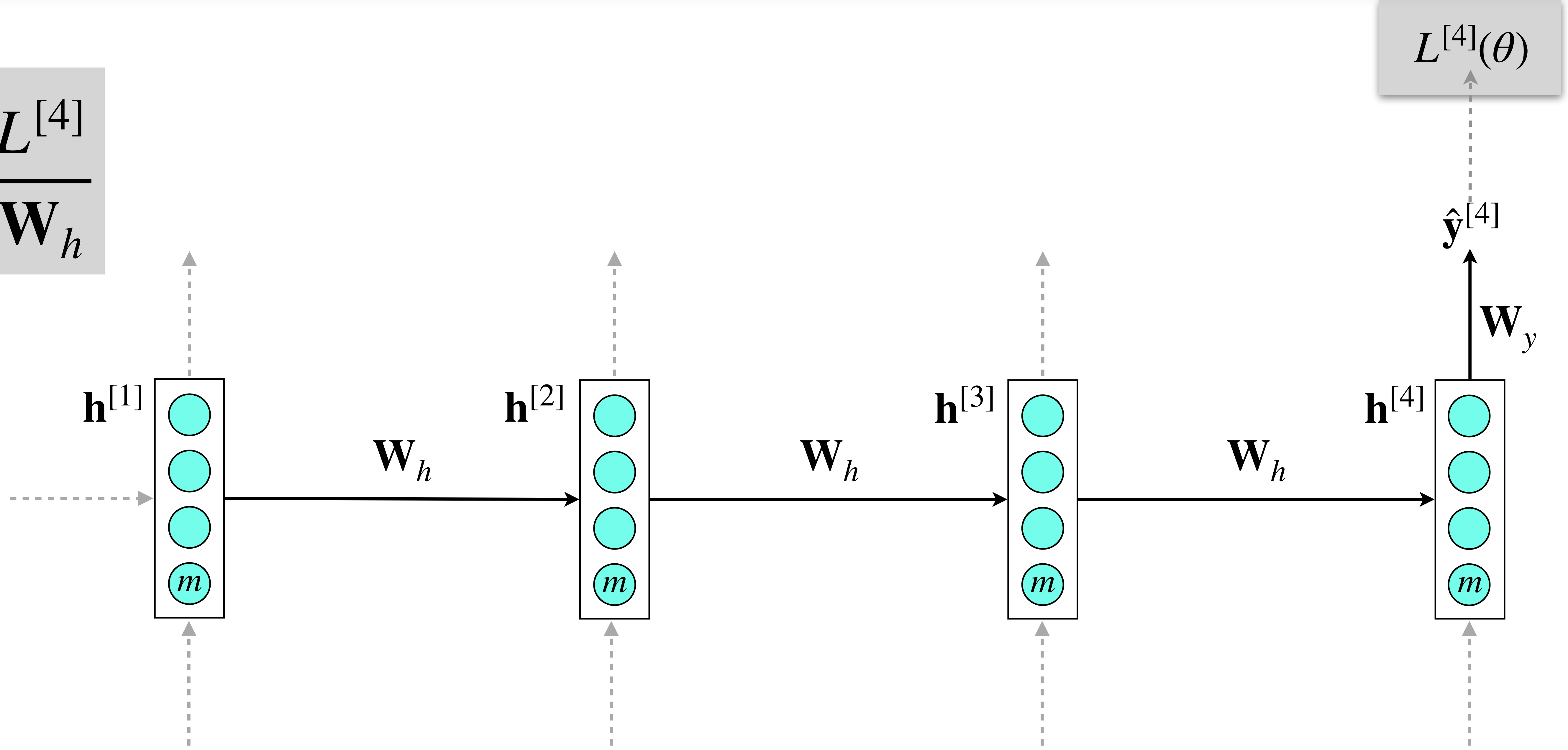
$$\frac{\partial \mathbf{h}^{[4]}}{\partial \mathbf{h}^{[k]}} = \prod_{j=k+1}^{4} \frac{\partial \mathbf{h}^{[j]}}{\partial \mathbf{h}^{[j-1]}}$$



$$\Longrightarrow \frac{\partial L^{[4]}}{\partial \mathbf{W}_h} \propto \frac{\partial \mathbf{h}^{[2]}}{\partial \mathbf{h}^{[1]}} \cdot \frac{\partial \mathbf{h}^{[3]}}{\partial \mathbf{h}^{[2]}} \cdot \frac{\partial \mathbf{h}^{[4]}}{\partial \mathbf{h}^{[3]}} + \frac{\partial \mathbf{h}^{[3]}}{\partial \mathbf{h}^{[2]}} \cdot \frac{\partial \mathbf{h}^{[4]}}{\partial \mathbf{h}^{[3]}} + \frac{\partial \mathbf{h}^{[4]}}{\partial \mathbf{h}^{[3]}}$$

Vanishing (*or exploding*) gradients

$$\frac{\partial L}{\partial \mathbf{W}_h} = \sum_{t=1}^{4} \frac{\partial L^{[t]}}{\partial \mathbf{W}_h} = \frac{\partial L^{[1]}}{\partial \mathbf{W}_h} + \frac{\partial L^{[2]}}{\partial \mathbf{W}_h} + \frac{\partial L^{[3]}}{\partial \mathbf{W}_h} + \frac{\partial L^{[4]}}{\partial \mathbf{W}_h}$$



$$\frac{\partial L^{[4]}}{\partial \mathbf{W}_h} = \sum_{k=1}^{4} \frac{\partial L^{[4]}}{\partial \hat{\mathbf{y}}^{[4]}} \cdot \frac{\partial \hat{\mathbf{y}}^{[4]}}{\partial \mathbf{h}^{[4]}} \cdot \frac{\partial \mathbf{h}^{[4]}}{\partial \mathbf{h}^{[k]}} \cdot \frac{\partial \mathbf{h}^{[k]}}{\partial \mathbf{W}_h}$$

let's focus on this component of the sum

$$\propto \frac{\partial \mathbf{h}^{[4]}}{\partial \mathbf{h}^{[1]}} + \frac{\partial \mathbf{h}^{[4]}}{\partial \mathbf{h}^{[2]}} + \frac{\partial \mathbf{h}^{[4]}}{\partial \mathbf{h}^{[3]}} + \frac{\partial \mathbf{h}^{[4]}}{\partial \mathbf{h}^{[4]}}$$

$$\propto \sum_{k=1}^{4} \prod_{j=k+1}^{4} \frac{\partial \mathbf{h}^{[j]}}{\partial \mathbf{h}^{[j-1]}}$$

recall

$$\frac{\partial \mathbf{h}^{[4]}}{\partial \mathbf{h}^{[k]}} = \prod_{j=k+1}^{4} \frac{\partial \mathbf{h}^{[j]}}{\partial \mathbf{h}^{[j-1]}}$$

what if these are small (or large)?

$$\Longrightarrow \frac{\partial L^{[4]}}{\partial \mathbf{W}_h} \propto \frac{\partial \mathbf{h}^{[2]}}{\partial \mathbf{h}^{[1]}} \cdot \frac{\partial \mathbf{h}^{[3]}}{\partial \mathbf{h}^{[2]}} \cdot \frac{\partial \mathbf{h}^{[4]}}{\partial \mathbf{h}^{[3]}} + \frac{\partial \mathbf{h}^{[3]}}{\partial \mathbf{h}^{[2]}} \cdot \frac{\partial \mathbf{h}^{[4]}}{\partial \mathbf{h}^{[3]}} + \frac{\partial \mathbf{h}^{[4]}}{\partial \mathbf{h}^{[3]}}$$

Vanishing (*or exploding*) gradients

$$\frac{\partial L}{\partial \mathbf{W}_h} = \sum_{t=1}^{4} \frac{\partial L^{[t]}}{\partial \mathbf{W}_h} = \frac{\partial L^{[1]}}{\partial \mathbf{W}_h} + \frac{\partial L^{[2]}}{\partial \mathbf{W}_h} + \frac{\partial L^{[3]}}{\partial \mathbf{W}_h} + \frac{\partial L^{[4]}}{\partial \mathbf{W}_h}$$

$$\frac{\partial L^{[4]}}{\partial \mathbf{W}_h} = \sum_{k=1}^{4} \frac{\partial L^{[4]}}{\partial \hat{\mathbf{y}}^{[4]}} \cdot \frac{\partial \hat{\mathbf{y}}^{[4]}}{\partial \mathbf{h}^{[4]}} \cdot \frac{\partial \mathbf{h}^{[4]}}{\partial \mathbf{h}^{[k]}} \cdot \frac{\partial \mathbf{h}^{[k]}}{\partial \mathbf{W}_h}$$

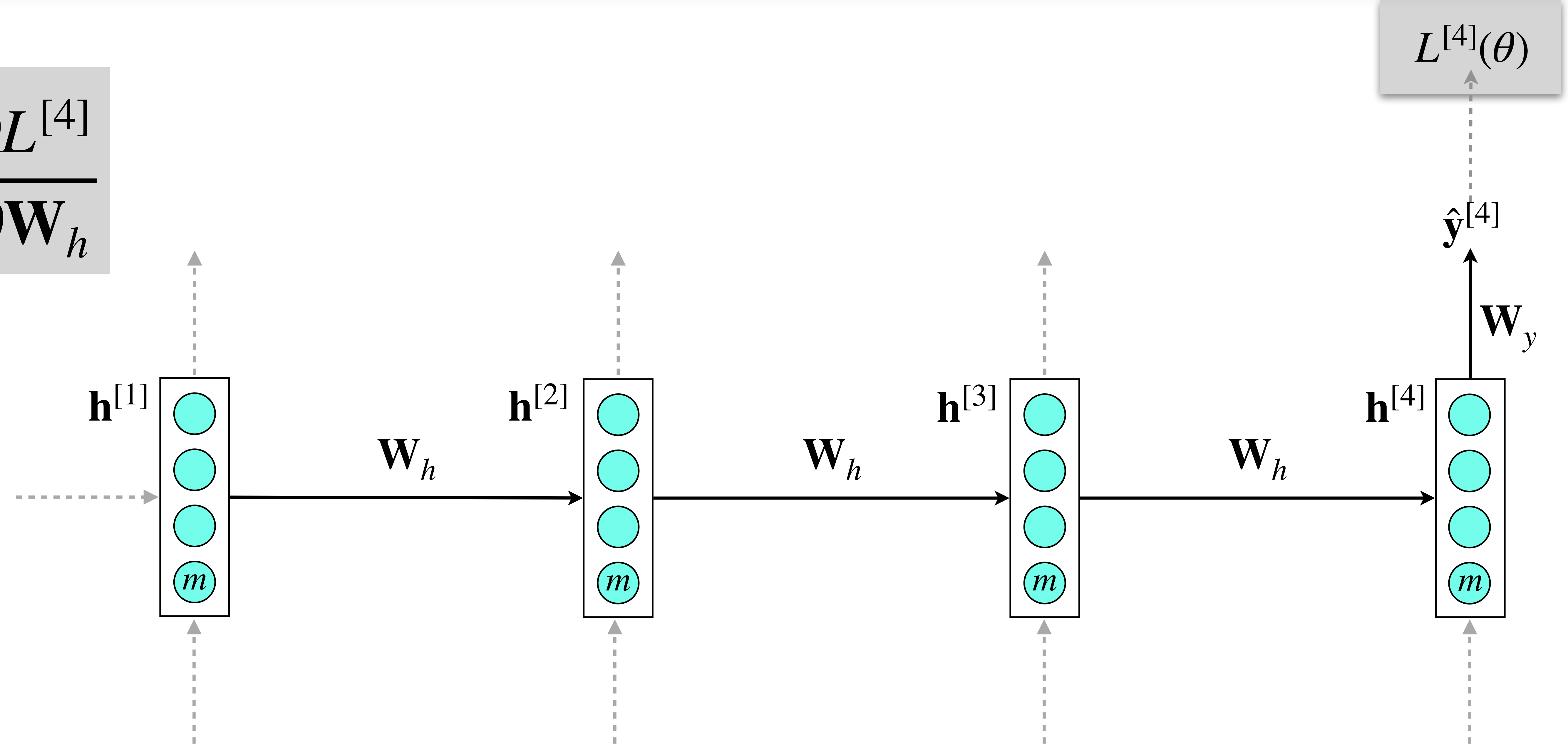
let's focus on this component of the sum

$$\propto \frac{\partial \mathbf{h}^{[4]}}{\partial \mathbf{h}^{[1]}} + \frac{\partial \mathbf{h}^{[4]}}{\partial \mathbf{h}^{[2]}} + \frac{\partial \mathbf{h}^{[4]}}{\partial \mathbf{h}^{[3]}} + \frac{\partial \mathbf{h}^{[4]}}{\partial \mathbf{h}^{[4]}}$$

$$\propto \sum_{k=1}^{4} \prod_{j=k+1}^{4} \frac{\partial \mathbf{h}^{[j]}}{\partial \mathbf{h}^{[j-1]}}$$

recall

$$\frac{\partial \mathbf{h}^{[4]}}{\partial \mathbf{h}^{[k]}} = \prod_{j=k+1}^{4} \frac{\partial \mathbf{h}^{[j]}}{\partial \mathbf{h}^{[j-1]}}$$



what if these are small (or large)? → *vanishing (or exploding) gradient as we backpropagate!*

$$\Longrightarrow \frac{\partial L^{[4]}}{\partial \mathbf{W}_h} \propto \frac{\partial \mathbf{h}^{[2]}}{\partial \mathbf{h}^{[1]}} \cdot \frac{\partial \mathbf{h}^{[3]}}{\partial \mathbf{h}^{[2]}} \cdot \frac{\partial \mathbf{h}^{[4]}}{\partial \mathbf{h}^{[3]}} + \frac{\partial \mathbf{h}^{[3]}}{\partial \mathbf{h}^{[2]}} \cdot \frac{\partial \mathbf{h}^{[4]}}{\partial \mathbf{h}^{[3]}} + \frac{\partial \mathbf{h}^{[4]}}{\partial \mathbf{h}^{[3]}}$$

Vanishing (*or exploding*) gradients — Proof intuition

$$\mathbf{h}^{[t]} = \sigma\left(\mathbf{W}_u \cdot \mathbf{u}^{[t]} + \mathbf{W}_h \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_h\right)$$

Vanishing (*or exploding*) gradients — Proof intuition

$$\mathbf{h}^{[t]} = \sigma\left(\mathbf{W}_u \cdot \mathbf{u}^{[t]} + \mathbf{W}_h \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_h\right)$$

let's ignore the activation function σ

$$\frac{\partial \mathbf{h}^{[t]}}{\partial \mathbf{h}^{[t-1]}} =$$

Vanishing (*or exploding*) gradients — Proof intuition

$$\mathbf{h}^{[t]} = \sigma\left(\mathbf{W}_u \cdot \mathbf{u}^{[t]} + \mathbf{W}_h \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_h\right)$$

let's ignore the activation function σ

$$\frac{\partial \mathbf{h}^{[t]}}{\partial \mathbf{h}^{[t-1]}} = \mathbf{W}_h$$

Vanishing (*or exploding*) gradients — Proof intuition

$$\mathbf{h}^{[t]} = \sigma\left(\mathbf{W}_u \cdot \mathbf{u}^{[t]} + \mathbf{W}_h \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_h\right)$$

let's ignore the activation function σ

$$\frac{\partial \mathbf{h}^{[t]}}{\partial \mathbf{h}^{[t-1]}} = \mathbf{W}_h$$

let's now see what happens when we compute the partial derivative of hidden state $\mathbf{h}^{[t]}$ w.r.t. the hidden state ξ time steps before it, i.e. $\mathbf{h}^{[t-\xi]}$

$$\frac{\partial \mathbf{h}^{[t]}}{\partial \mathbf{h}^{[t-\xi]}} = \frac{\partial \mathbf{h}^{[t]}}{\partial \mathbf{h}^{[t-1]}} \cdot \ldots \cdot \frac{\partial \mathbf{h}^{[t-\xi-1]}}{\partial \mathbf{h}^{[t-\xi]}}$$

Vanishing (*or exploding*) gradients — Proof intuition

$$\mathbf{h}^{[t]} = \sigma\left(\mathbf{W}_u \cdot \mathbf{u}^{[t]} + \mathbf{W}_h \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_h\right)$$

let's ignore the activation function σ

$$\frac{\partial \mathbf{h}^{[t]}}{\partial \mathbf{h}^{[t-1]}} = \mathbf{W}_h$$

$$\frac{\partial \mathbf{h}^{[t]}}{\partial \mathbf{h}^{[t-\xi]}} = \underbrace{\frac{\partial \mathbf{h}^{[t]}}{\partial \mathbf{h}^{[t-1]}} \cdot \ldots \cdot \frac{\partial \mathbf{h}^{[t-\xi-1]}}{\partial \mathbf{h}^{[t-\xi]}}}_{\xi \text{ components}} = \mathbf{W}_h^{\xi}$$

let's now see what happens when we compute the partial derivative of hidden state $\mathbf{h}^{[t]}$ w.r.t. the hidden state ξ time steps before it, i.e. $\mathbf{h}^{[t-\xi]}$

Vanishing (*or exploding*) gradients — Proof intuition

$$\mathbf{h}^{[t]} = \sigma\left(\mathbf{W}_u \cdot \mathbf{u}^{[t]} + \mathbf{W}_h \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_h\right)$$

let's ignore the activation function σ

$$\frac{\partial \mathbf{h}^{[t]}}{\partial \mathbf{h}^{[t-1]}} = \mathbf{W}_h$$

$$\frac{\partial \mathbf{h}^{[t]}}{\partial \mathbf{h}^{[t-\xi]}} = \underbrace{\frac{\partial \mathbf{h}^{[t]}}{\partial \mathbf{h}^{[t-1]}} \cdot \ldots \cdot \frac{\partial \mathbf{h}^{[t-\xi-1]}}{\partial \mathbf{h}^{[t-\xi]}}}_{\xi \text{ components}} = \mathbf{W}_h^{\xi}$$

let's now see what happens when we compute the partial derivative of hidden state $\mathbf{h}^{[t]}$ w.r.t. the hidden state ξ time steps before it, i.e. $\mathbf{h}^{[t-\xi]}$

- If $\mathbf{W}_h$ has eigenvalues < 1, gradients become exponentially smaller as time steps ξ increase $\implies$ gradients will become 0, i.e. vanish

Vanishing (*or exploding*) gradients — Proof intuition

$$\mathbf{h}^{[t]} = \sigma\left(\mathbf{W}_u \cdot \mathbf{u}^{[t]} + \mathbf{W}_h \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_h\right)$$

let's ignore the activation function σ

$$\frac{\partial \mathbf{h}^{[t]}}{\partial \mathbf{h}^{[t-1]}} = \mathbf{W}_h$$

$$\frac{\partial \mathbf{h}^{[t]}}{\partial \mathbf{h}^{[t-\xi]}} = \underbrace{\frac{\partial \mathbf{h}^{[t]}}{\partial \mathbf{h}^{[t-1]}} \cdot \ldots \cdot \frac{\partial \mathbf{h}^{[t-\xi-1]}}{\partial \mathbf{h}^{[t-\xi]}}}_{\xi \text{ components}} = \mathbf{W}_h^{\xi}$$

let's now see what happens when we compute the partial derivative of hidden state $\mathbf{h}^{[t]}$ w.r.t. the hidden state ξ time steps before it, i.e. $\mathbf{h}^{[t-\xi]}$

- If $\mathbf{W}_h$ has eigenvalues < 1, gradients become exponentially smaller as time steps ξ increase $\implies$ gradients will become 0, i.e. vanish
- If $\mathbf{W}_h$ has eigenvalues > 1 $\implies$ gradients will explode

Vanishing (*or exploding*) gradients — Proof intuition

$$\mathbf{h}^{[t]} = \sigma\left(\mathbf{W}_u \cdot \mathbf{u}^{[t]} + \mathbf{W}_h \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_h\right)$$

let's ignore the activation function σ

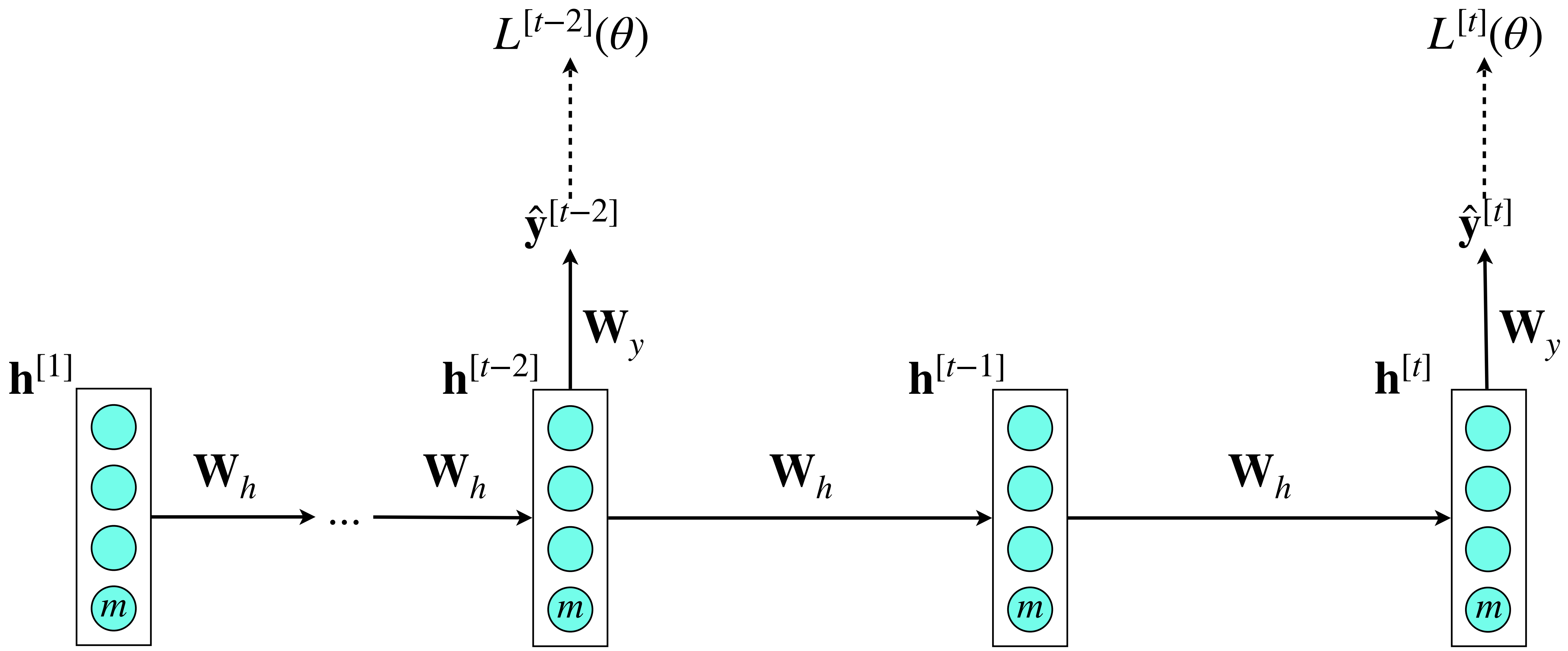
$$\frac{\partial \mathbf{h}^{[t]}}{\partial \mathbf{h}^{[t-1]}} = \mathbf{W}_h$$

$$\frac{\partial \mathbf{h}^{[t]}}{\partial \mathbf{h}^{[t-\xi]}} = \underbrace{\frac{\partial \mathbf{h}^{[t]}}{\partial \mathbf{h}^{[t-1]}} \cdot \ldots \cdot \frac{\partial \mathbf{h}^{[t-\xi-1]}}{\partial \mathbf{h}^{[t-\xi]}}}_{\xi \text{ components}} = \mathbf{W}_h^{\xi}$$

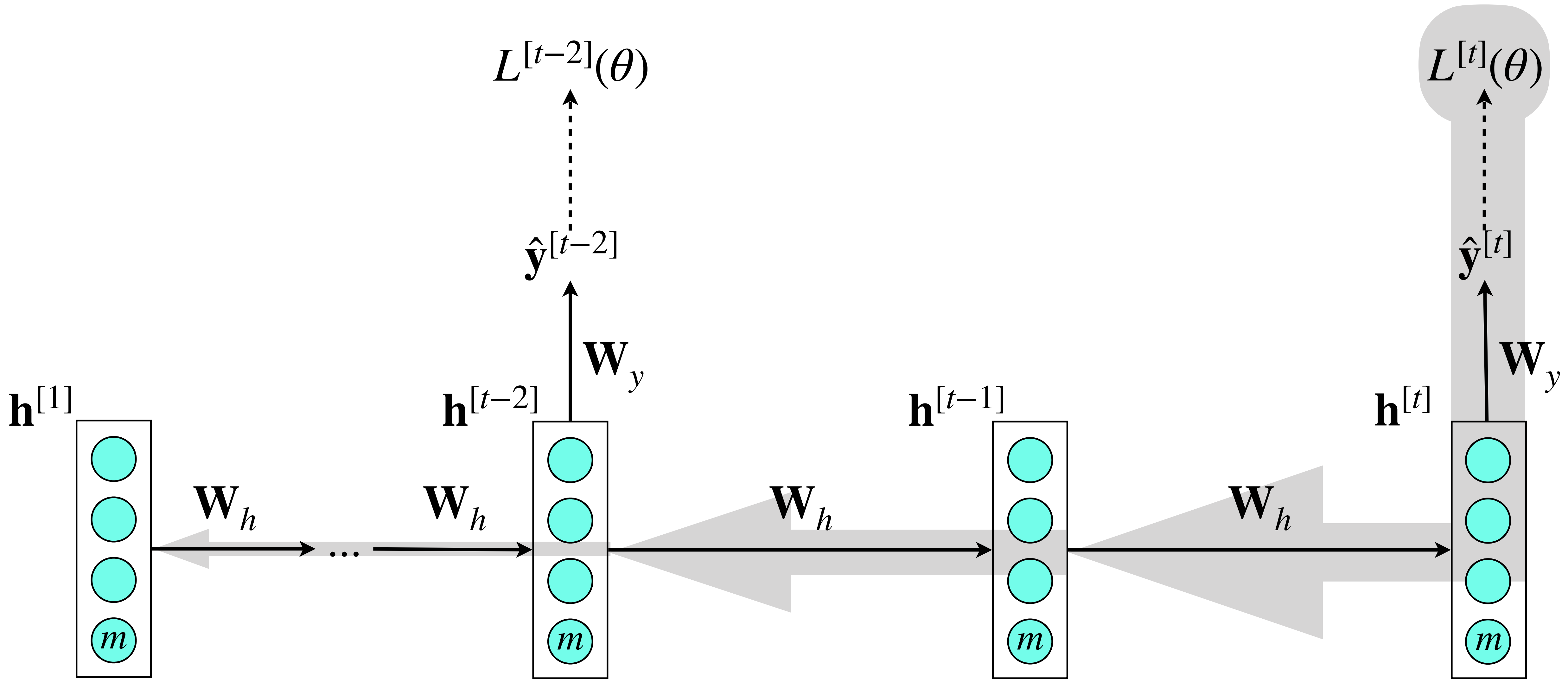
let's now see what happens when we compute the partial derivative of hidden state $\mathbf{h}^{[t]}$ w.r.t. the hidden state ξ time steps before it, i.e. $\mathbf{h}^{[t-\xi]}$

- If $\mathbf{W}_h$ has eigenvalues < 1, gradients become exponentially smaller as time steps ξ increase $\Longrightarrow$ gradients will become 0, i.e. vanish
- If $\mathbf{W}_h$ has eigenvalues > 1 $\Longrightarrow$ gradients will explode
- Similar outcome when we re-introduce an activation function

Vanishing gradients are an issue because...

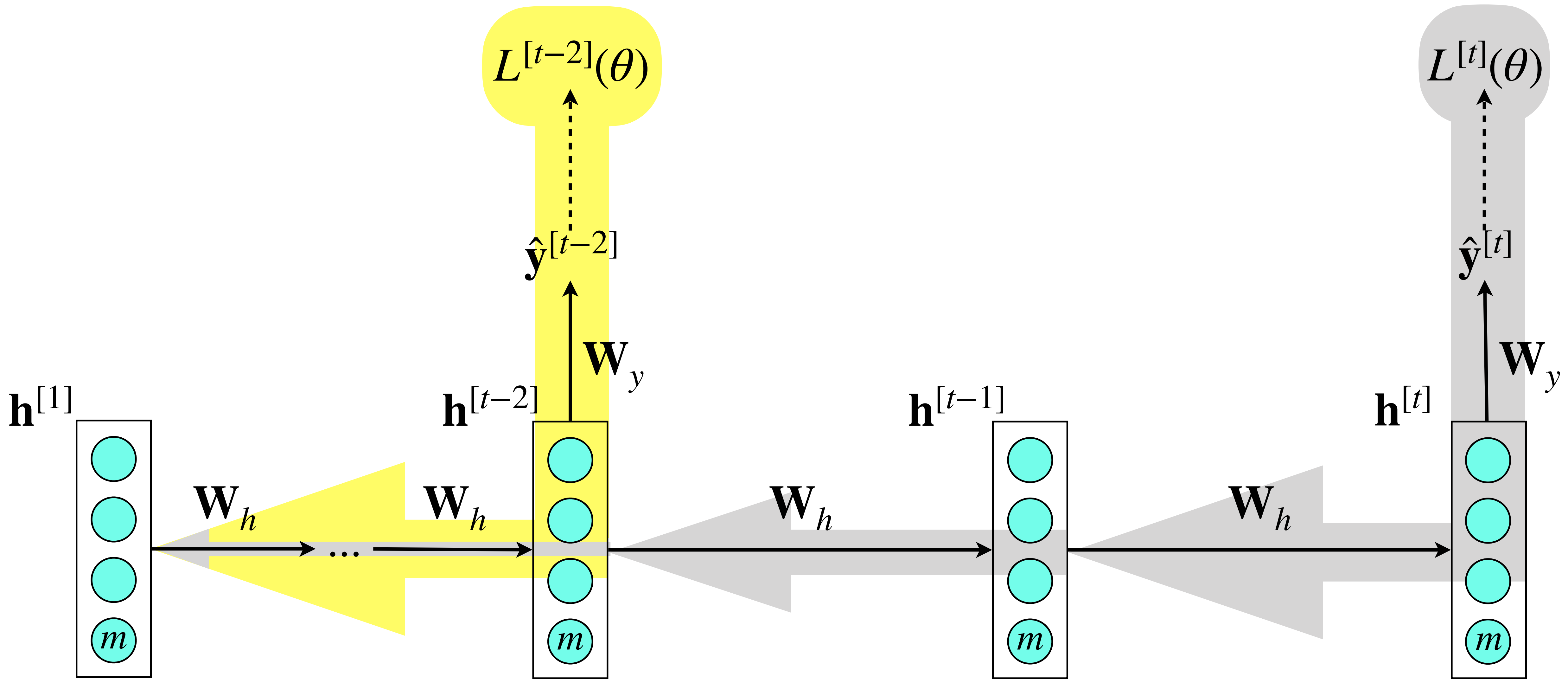


Vanishing gradients are an issue because...



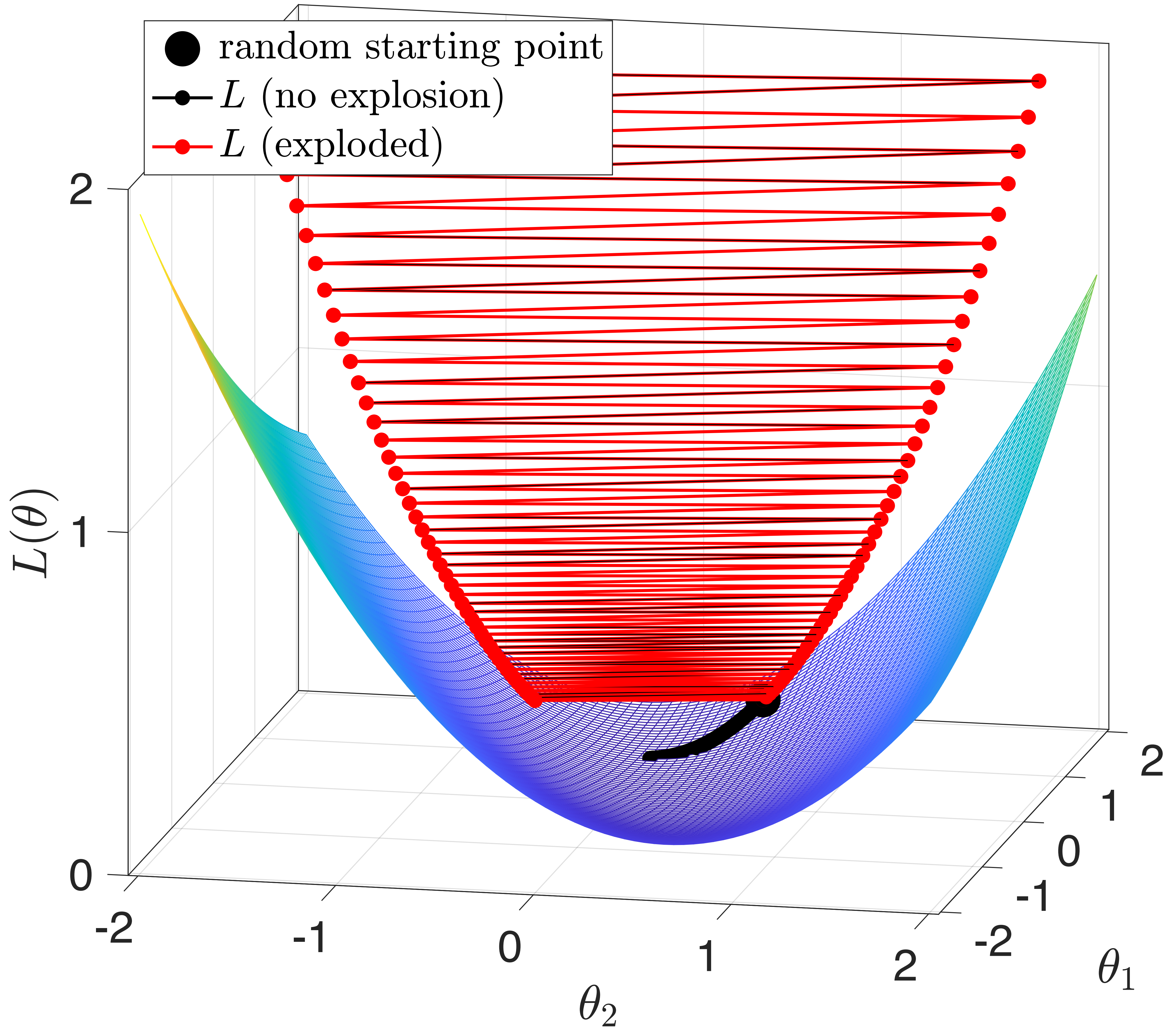
- Signal (gradient) from early states that are distant to the current state is lost $\Longrightarrow$long-terms effects are not captured

Vanishing gradients are an issue because...



- Signal (gradient) from early states that are distant to the current state is lost $\Longrightarrow$ long-terms effects are not captured
- NB: Parameters will still be updated, but based on shorter-term gradients that have not vanished.

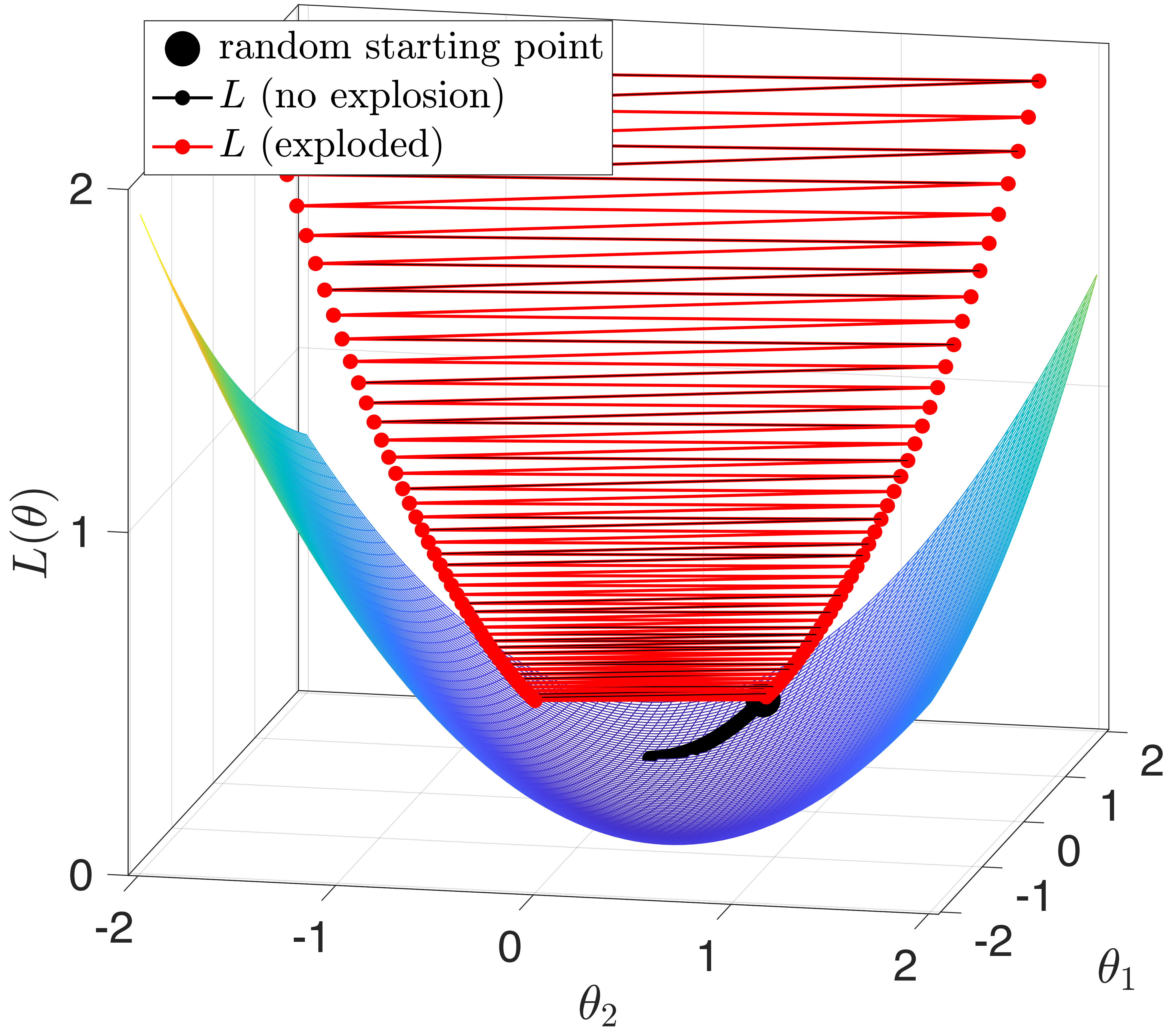
Exploding gradients



Exploding gradients

- Large gradients, $\frac{\partial L}{\partial \theta_j}$, mean large learning steps during optimisation

$$\theta_{j+1} = \theta_j - \eta \frac{\partial L}{\partial \theta_j}$$

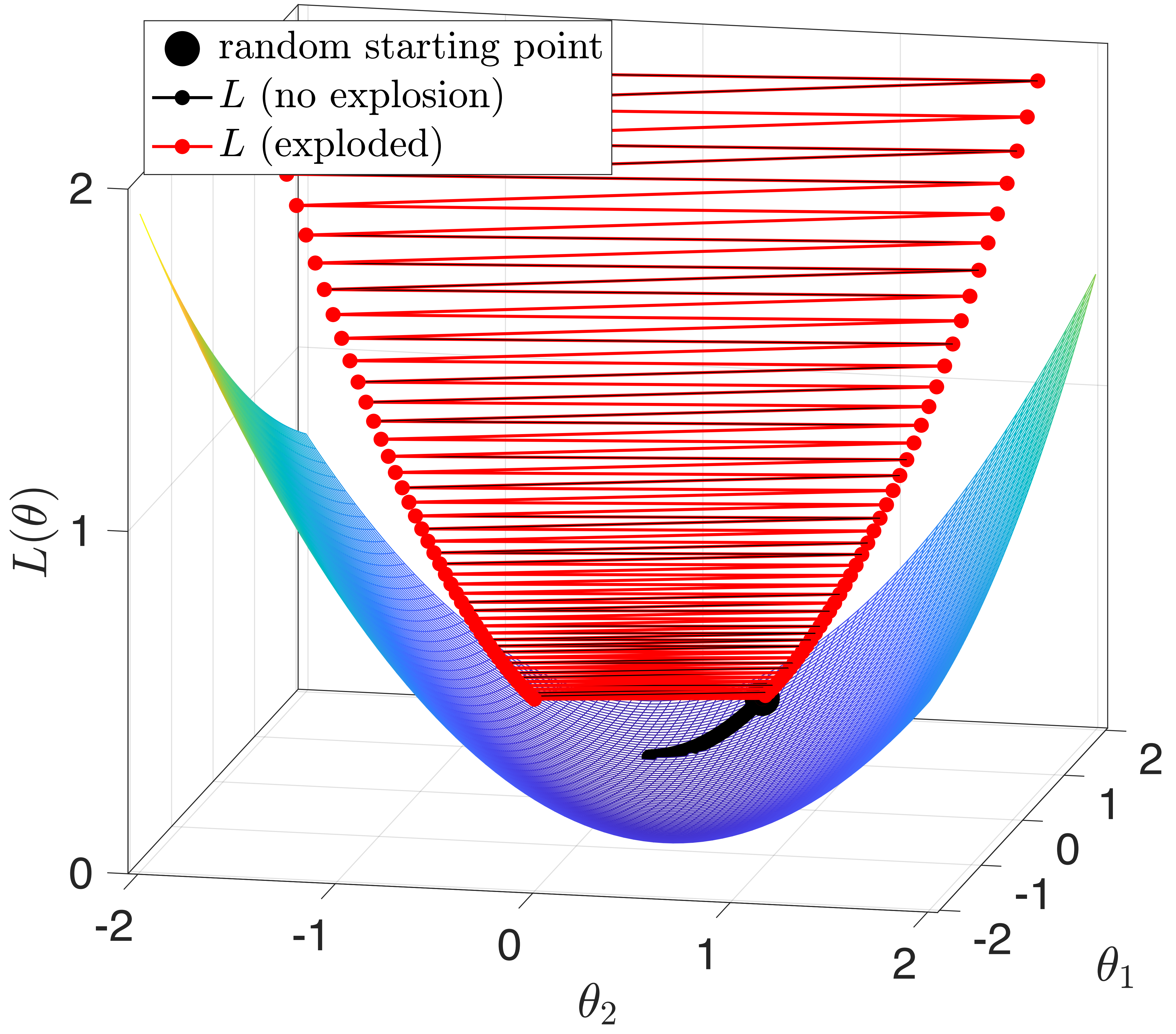


Exploding gradients

- Large gradients, $\frac{\partial L}{\partial \theta_j}$, mean large learning steps during optimisation

$$\theta_{j+1} = \theta_j - \eta \frac{\partial L}{\partial \theta_j}$$

- This would possibly result in a poor parameter setting from which we might not be able to recover, especially while using large learning steps

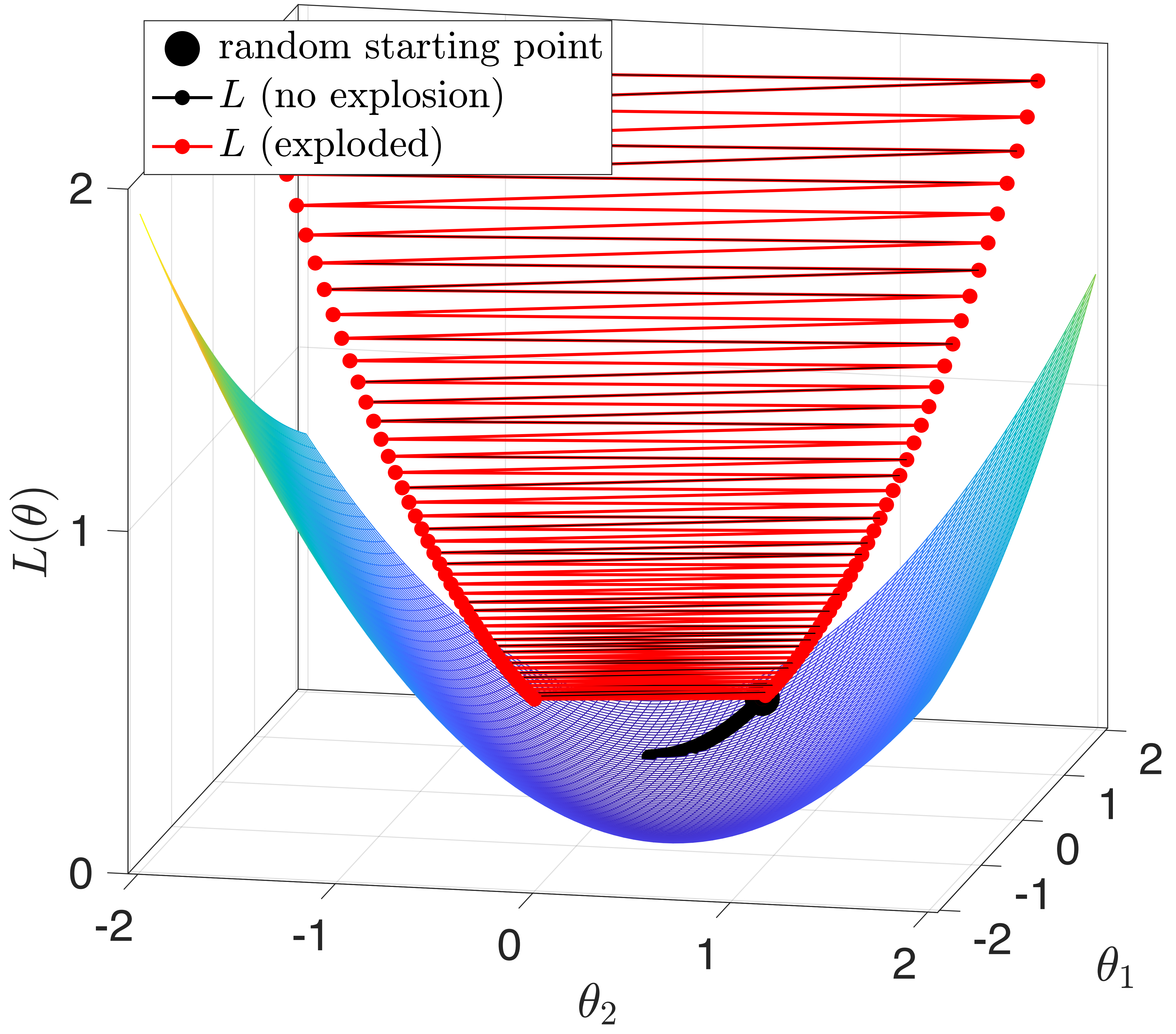


Exploding gradients

- Large gradients, $\frac{\partial L}{\partial \theta_j}$, mean large learning steps during optimisation

$$\theta_{j+1} = \theta_j - \eta \frac{\partial L}{\partial \theta_j}$$

- This would possibly result in a poor parameter setting from which we might not be able to recover, especially while using large learning steps
- The worst penalty to pay would be NaN / Inf errors in the NN parameters; training will have to be restarted



An "easy" solution to exploding gradients — Gradient clipping

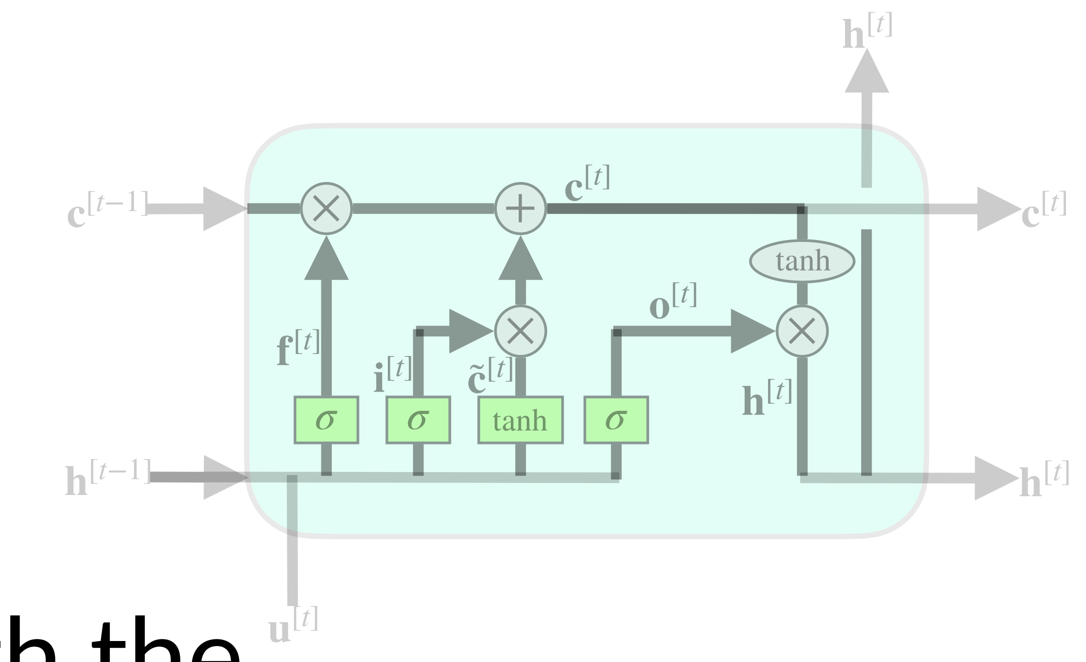
- If the L2 norm of the gradient is greater than a threshold γ, simply scale the gradient down, i.e. clip it!

$$
\begin{aligned}
&\mathbf{q} = \frac{\partial L}{\partial \theta} \\
&\textbf{if } \ \|\mathbf{q}\| \geq \gamma \ \textbf{ then} \\
&\qquad \mathbf{q} = \frac{\gamma}{\|\mathbf{q}\|} \cdot \mathbf{q} \\
&\textbf{endif}
\end{aligned}
$$

- We are still taking a step in the same direction, albeit a smaller one

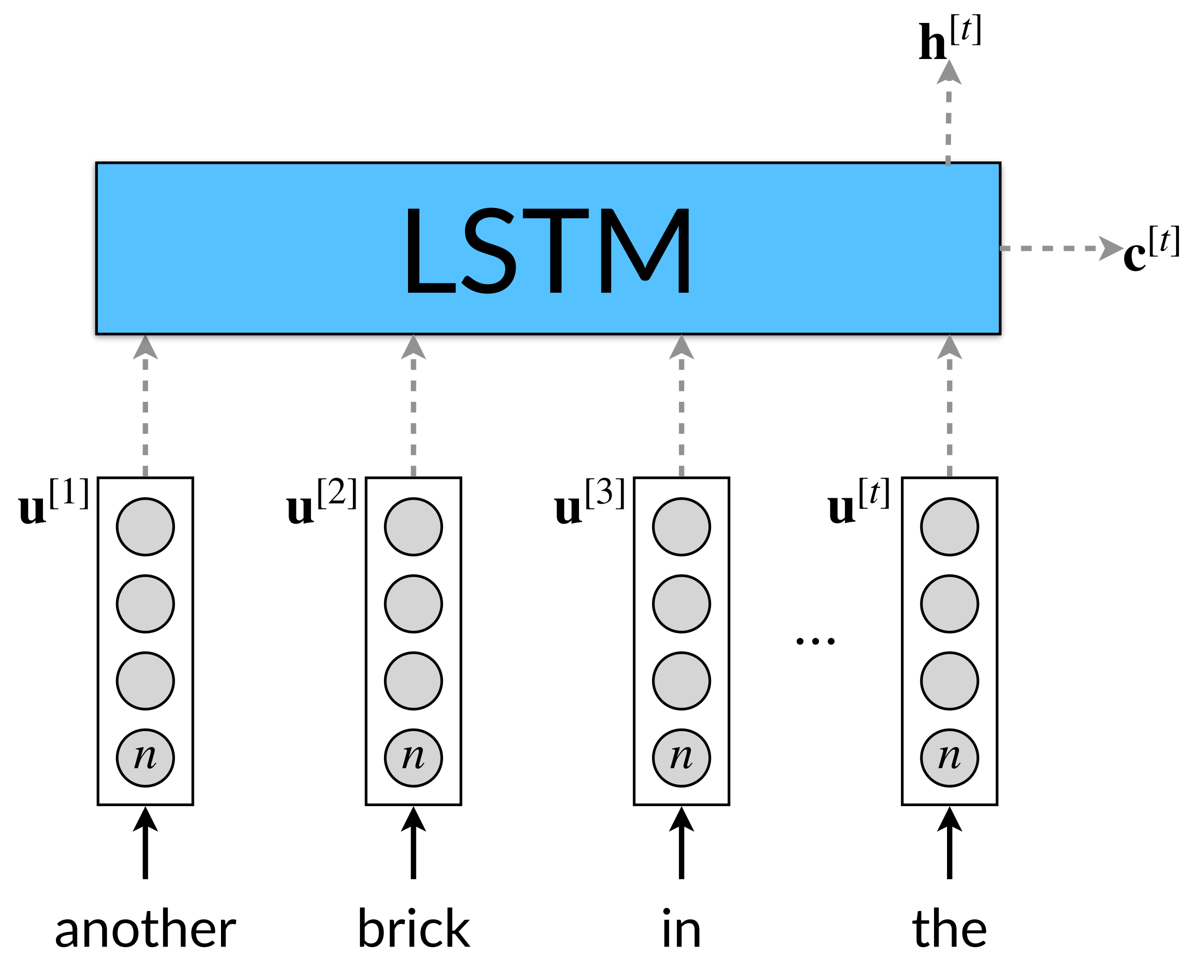
- We need to learn / set the threshold γ; a good heuristic 0.5 to 10 times the average norm of the gradient over a sufficient number of updates

Long Short-Term Memory (LSTM) — A better RNN

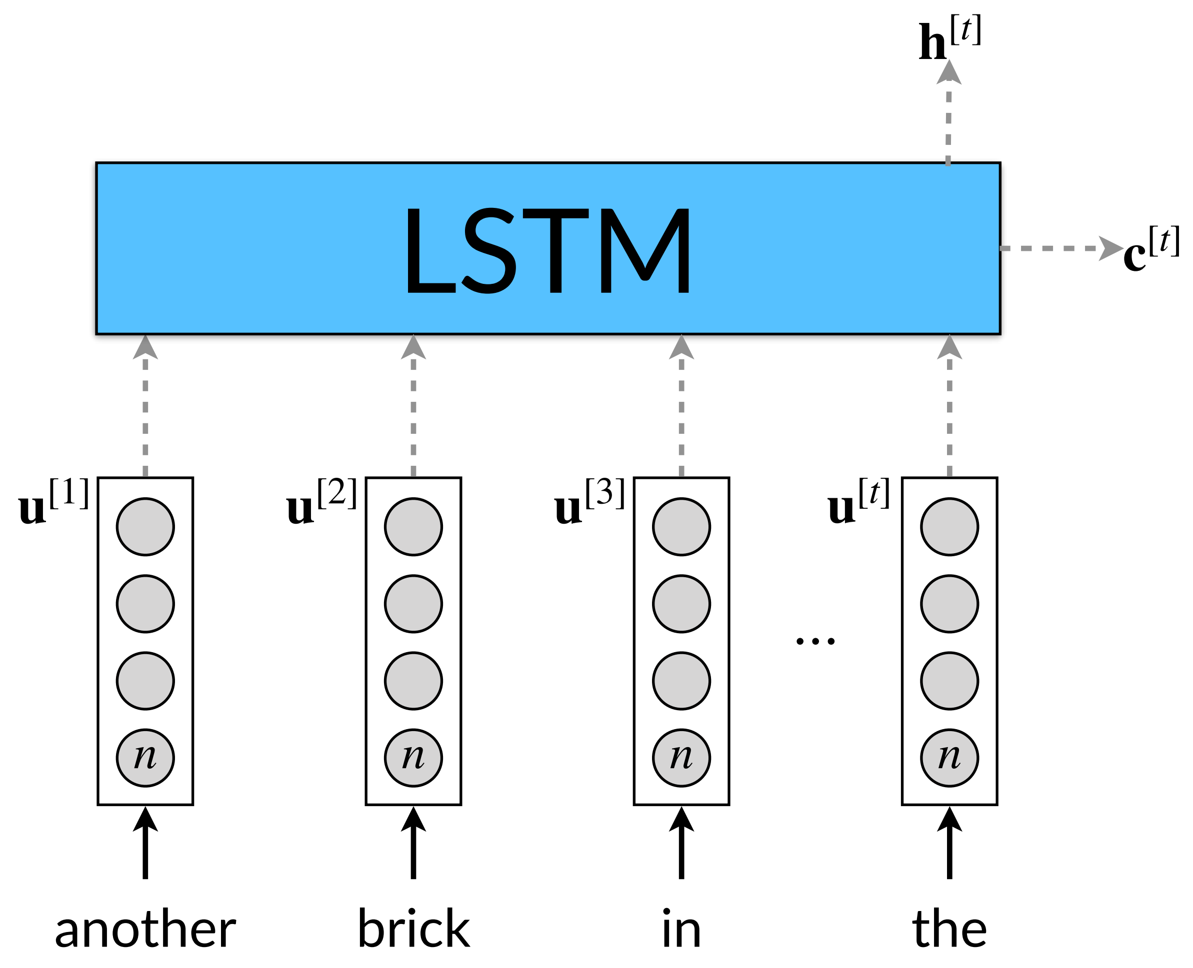


- Simple RNNs **fail to maintain information over many time steps** as their architecture does not have explicit components to do so
- Long Short-Term Memory (LSTM) is an update to the RNN architecture with the aim of solving the problem of vanishing gradients
- The LSTM has a hidden state like the simple RNN, but also a "cell" state, both being n-dimensional vectors
- The cell is designed to store more long-term information and acts like a memory module — the LSTM can read, delete, and write information to the cell
- 3 new n-dimensional vectors control what is read, deleted, and written; however their decisions are "probabilistic" $\in [0,1]$ for each of the n dimensions (not 0 or 1) and are learned during optimisation

Long Short-Term Memory (LSTM)



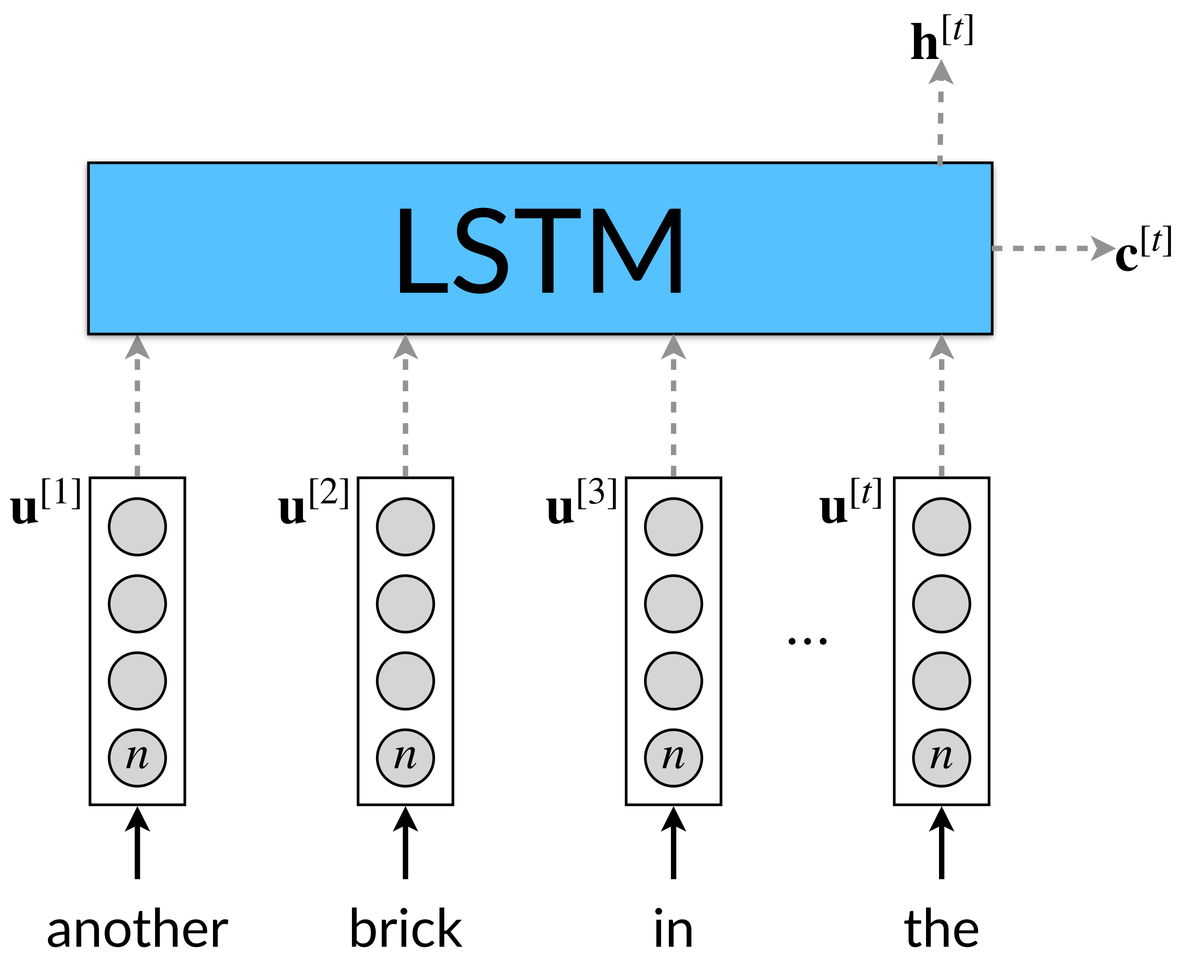
Long Short-Term Memory (LSTM)



<u>New content</u>: similarly to simple RNN, there is an input sequence $\mathbf{u}^{[1]}, \dots, \mathbf{u}^{[t]}$ and there is a dependency to the previous hidden state $\mathbf{h}^{[t-1]}$

$$\tilde{\mathbf{c}}^{[t]} = \tanh\left(\mathbf{U}_c \cdot \mathbf{u}^{[t]} + \mathbf{W}_c \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_c\right)$$

Long Short-Term Memory (LSTM)



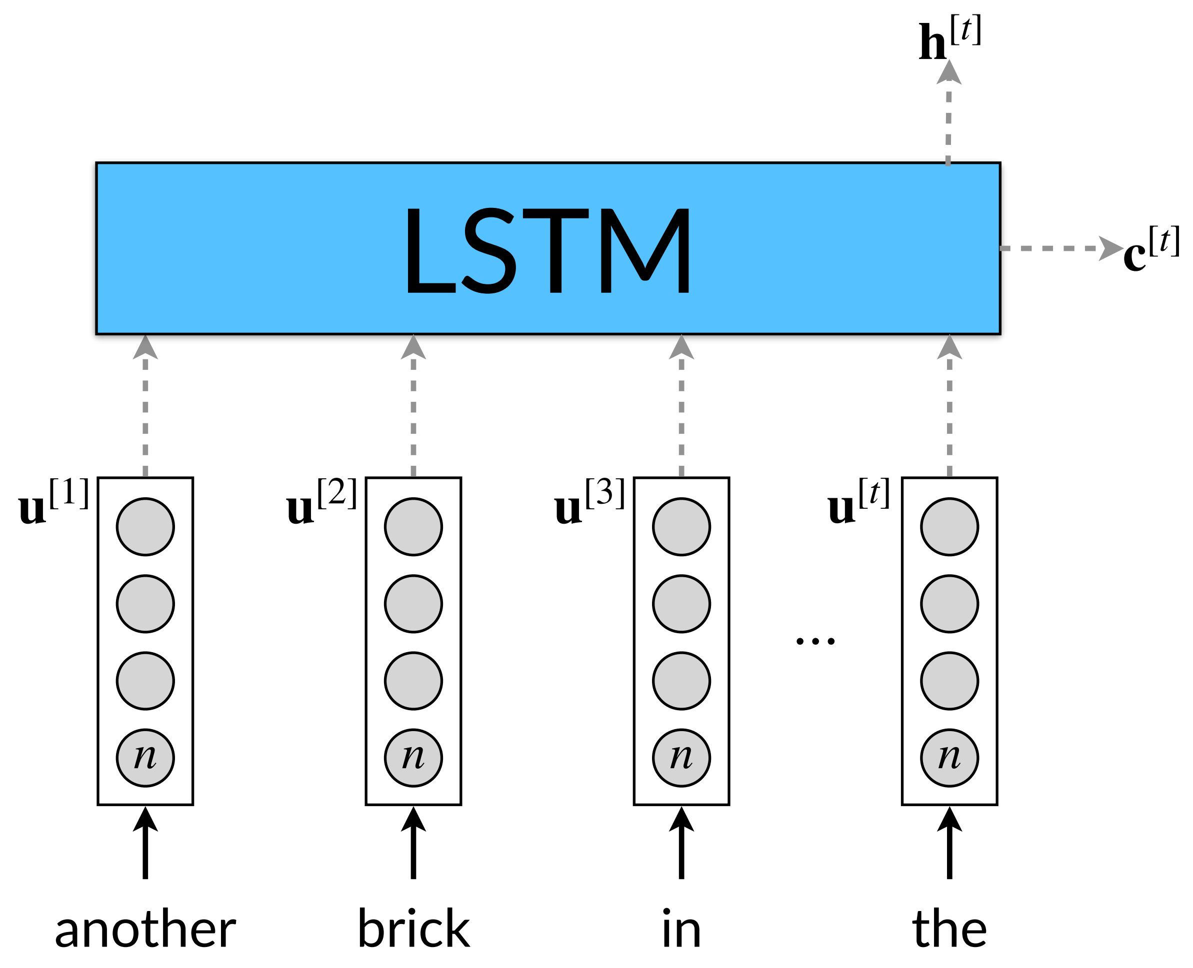
New content: similarly to simple RNN, there is an input sequence $\mathbf{u}^{[1]}, \dots, \mathbf{u}^{[t]}$ and there is a dependency to the previous hidden state $\mathbf{h}^{[t-1]}$

$$\tilde{\mathbf{c}}^{[t]} = \tanh\left(\mathbf{U}_c \cdot \mathbf{u}^{[t]} + \mathbf{W}_c \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_c\right)$$

Forget gate: what should be forgotten from the previous cell state; $0 \rightarrow 1$ ~ forget → keep.

$$\mathbf{f}^{[t]} = \sigma\left(\mathbf{U}_f \cdot \mathbf{u}^{[t]} + \mathbf{W}_f \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_f\right)$$

Long Short-Term Memory (LSTM)



New content: similarly to simple RNN, there is an input sequence $\mathbf{u}^{[1]}, \ldots, \mathbf{u}^{[t]}$ and there is a dependency to the previous hidden state $\mathbf{h}^{[t-1]}$

$$\tilde{\mathbf{c}}^{[t]} = \tanh\left(\mathbf{U}_c \cdot \mathbf{u}^{[t]} + \mathbf{W}_c \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_c\right)$$

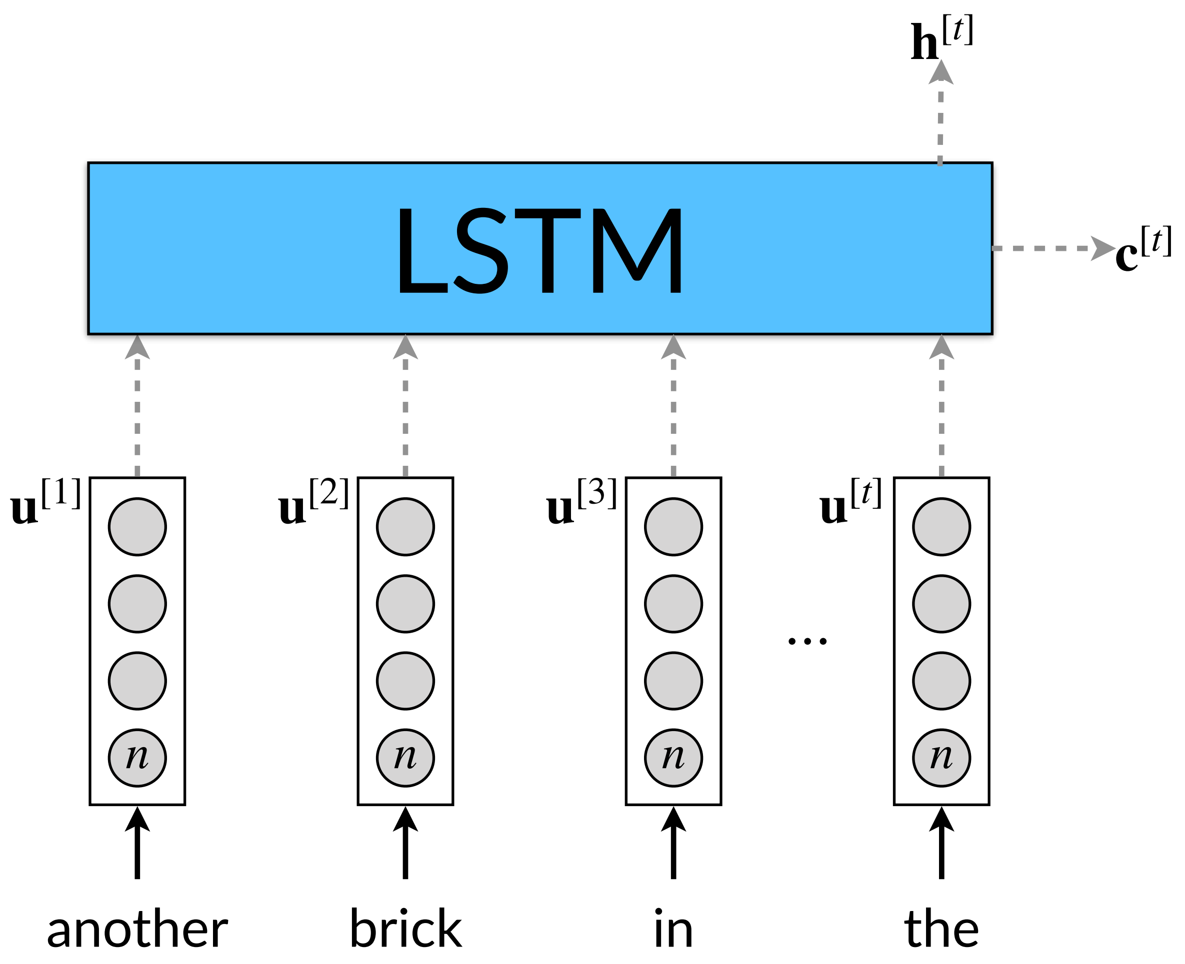
Forget gate: what should be forgotten from the previous cell state; $0 \rightarrow 1$ ~ forget → keep.

$$\mathbf{f}^{[t]} = \sigma\left(\mathbf{U}_f \cdot \mathbf{u}^{[t]} + \mathbf{W}_f \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_f\right)$$

Input gate: what should be kept from the new content?

$$\mathbf{i}^{[t]} = \sigma\left(\mathbf{U}_i \cdot \mathbf{u}^{[t]} + \mathbf{W}_i \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_i\right)$$

Long Short-Term Memory (LSTM)



New content: similarly to simple RNN, there is an input sequence $\mathbf{u}^{[1]}, \ldots, \mathbf{u}^{[t]}$ and there is a dependency to the previous hidden state $\mathbf{h}^{[t-1]}$

$$\tilde{\mathbf{c}}^{[t]} = \tanh\left(\mathbf{U}_c \cdot \mathbf{u}^{[t]} + \mathbf{W}_c \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_c\right)$$

Forget gate: what should be forgotten from the previous cell state; $0 \rightarrow 1$ ~ forget $\rightarrow$ keep.

$$\mathbf{f}^{[t]} = \sigma\left(\mathbf{U}_f \cdot \mathbf{u}^{[t]} + \mathbf{W}_f \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_f\right)$$

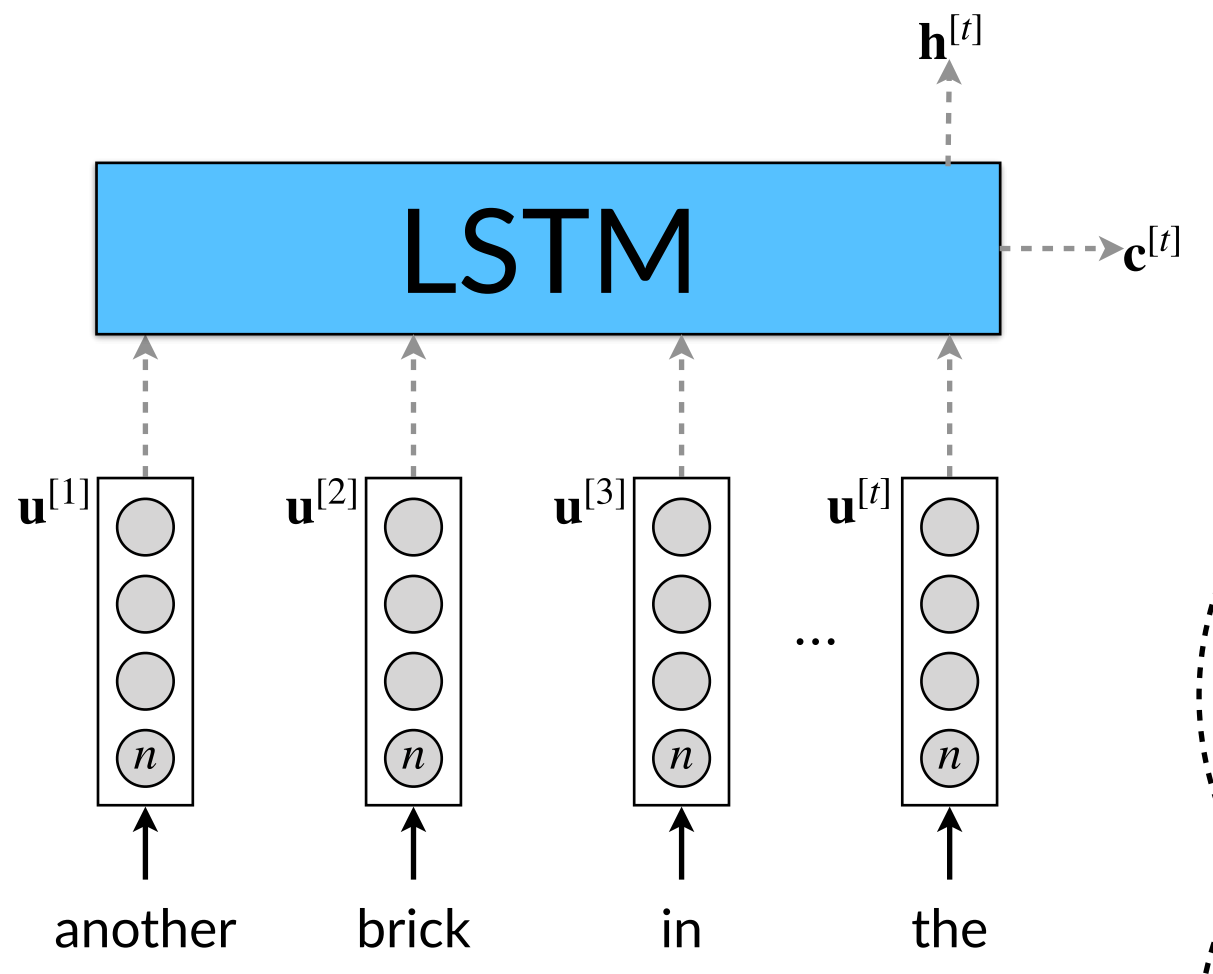
Input gate: what should be kept from the new content?

$$\mathbf{i}^{[t]} = \sigma\left(\mathbf{U}_i \cdot \mathbf{u}^{[t]} + \mathbf{W}_i \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_i\right)$$

Cell state: forget some past, keep some present

$$\mathbf{c}^{[t]} = \mathbf{f}^{[t]} \odot \mathbf{c}^{[t-1]} + \mathbf{i}^{[t]} \odot \tilde{\mathbf{c}}^{[t]}$$

Long Short-Term Memory (LSTM)



New content: similarly to simple RNN, there is an input sequence $\mathbf{u}^{[1]}, \ldots, \mathbf{u}^{[t]}$ and there is a dependency to the previous hidden state $\mathbf{h}^{[t-1]}$

$$\tilde{\mathbf{c}}^{[t]} = \tanh\left(\mathbf{U}_c \cdot \mathbf{u}^{[t]} + \mathbf{W}_c \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_c\right)$$

Forget gate: what should be forgotten from the previous cell state; $0 \to 1$ ~ forget → keep.

$$\mathbf{f}^{[t]} = \sigma\left(\mathbf{U}_f \cdot \mathbf{u}^{[t]} + \mathbf{W}_f \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_f\right)$$

Input gate: what should be kept from the new content?

$$\mathbf{i}^{[t]} = \sigma\left(\mathbf{U}_i \cdot \mathbf{u}^{[t]} + \mathbf{W}_i \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_i\right)$$

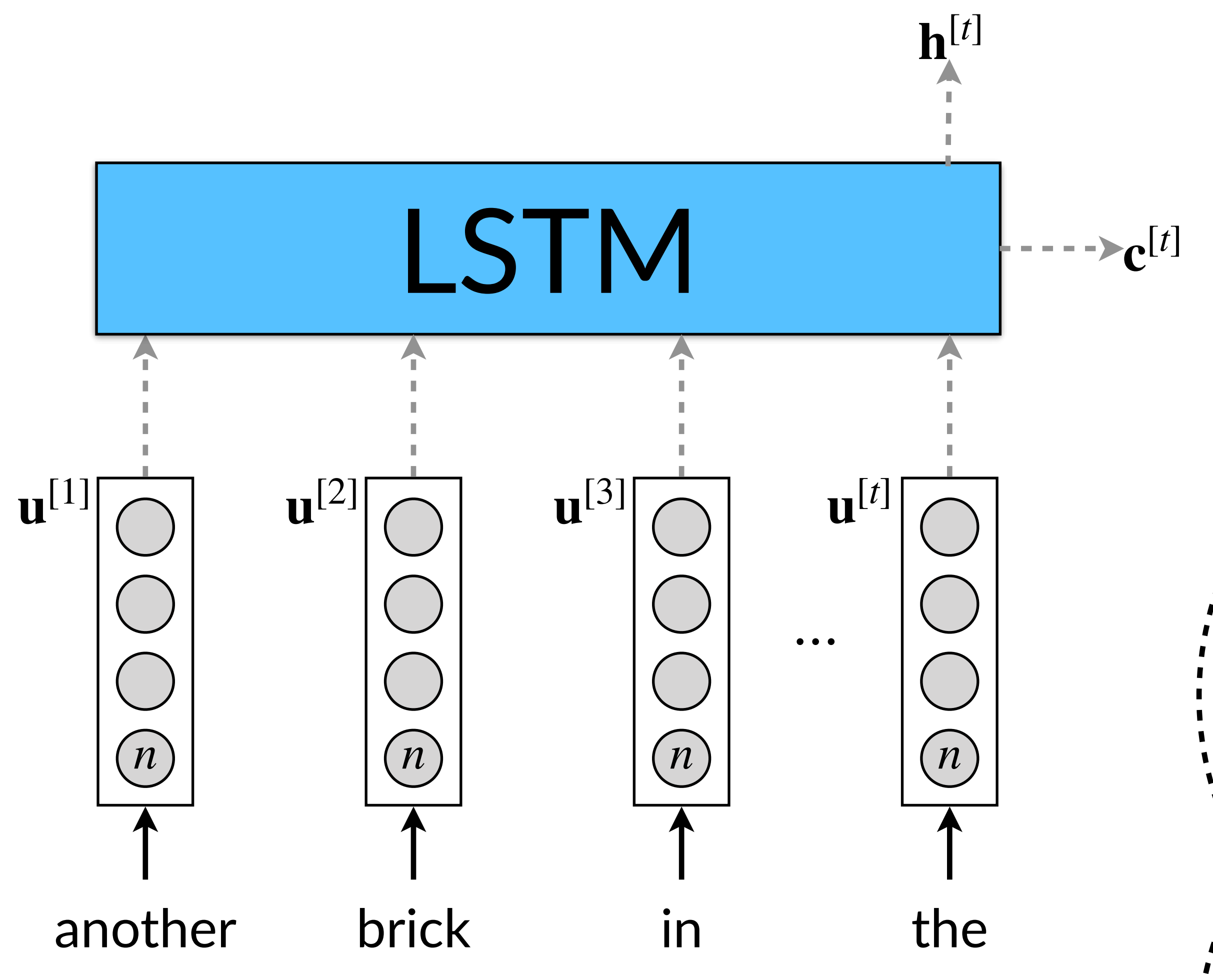
Cell state: forget some past, keep some present

$$\mathbf{c}^{[t]} = \mathbf{f}^{[t]} \odot \mathbf{c}^{[t-1]} + \mathbf{i}^{[t]} \odot \tilde{\mathbf{c}}^{[t]}$$

Output gate: what parts of the cell state will be passed on to the hidden state

$$\mathbf{o}^{[t]} = \sigma\left(\mathbf{U}_o \cdot \mathbf{u}^{[t]} + \mathbf{W}_o \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_o\right)$$

Long Short-Term Memory (LSTM)



New content: similarly to simple RNN, there is an input sequence $\mathbf{u}^{[1]}, \ldots, \mathbf{u}^{[t]}$ and there is a dependency to the previous hidden state $\mathbf{h}^{[t-1]}$

$$\tilde{\mathbf{c}}^{[t]} = \tanh\left(\mathbf{U}_c \cdot \mathbf{u}^{[t]} + \mathbf{W}_c \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_c\right)$$

Forget gate: what should be forgotten from the previous cell state; $0 \rightarrow 1$ ~ forget → keep.

$$\mathbf{f}^{[t]} = \sigma\left(\mathbf{U}_f \cdot \mathbf{u}^{[t]} + \mathbf{W}_f \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_f\right)$$

Input gate: what should be kept from the new content?

$$\mathbf{i}^{[t]} = \sigma\left(\mathbf{U}_i \cdot \mathbf{u}^{[t]} + \mathbf{W}_i \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_i\right)$$

Cell state: forget some past, keep some present

$$\mathbf{c}^{[t]} = \mathbf{f}^{[t]} \odot \mathbf{c}^{[t-1]} + \mathbf{i}^{[t]} \odot \tilde{\mathbf{c}}^{[t]}$$

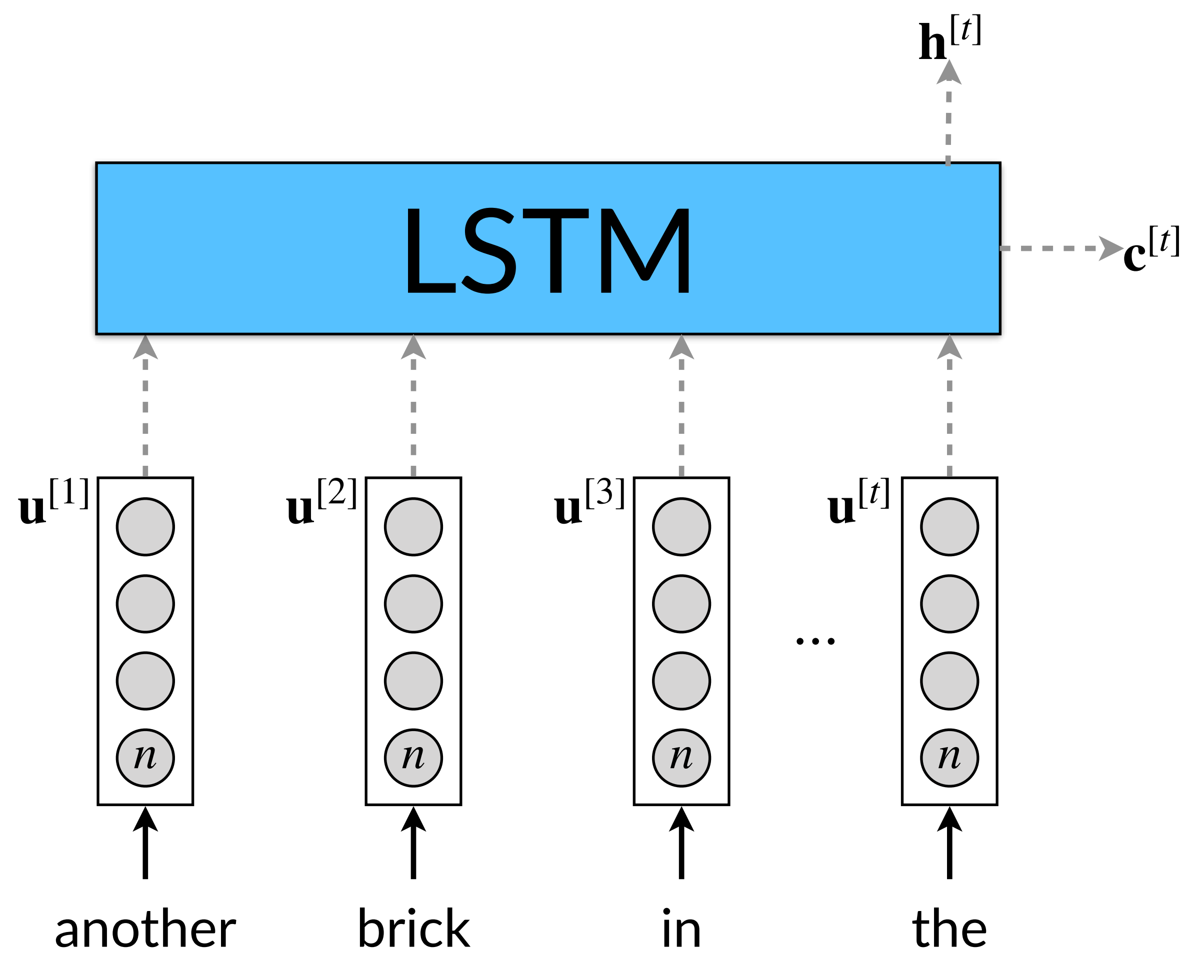
Output gate: what parts of the cell state will be passed on to the hidden state

$$\mathbf{o}^{[t]} = \sigma\left(\mathbf{U}_o \cdot \mathbf{u}^{[t]} + \mathbf{W}_o \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_o\right)$$

Hidden state: maintains the part of the cell state that will pass from the output gate

$$\mathbf{h}^{[t]} = \mathbf{o}^{[t]} \odot \tanh\left(\mathbf{c}^{[t]}\right)$$

Long Short-Term Memory (LSTM)



all gate values range from 0 to 1 given the sigmoid activation (σ)

$\mathbf{f}^{[t]}, \mathbf{i}^{[t]}, \mathbf{o}^{[t]} \in (0,1)^n$

$\tilde{\mathbf{c}}^{[t]}, \mathbf{h}^{[t]} \in (-1,1)^n$ and $\mathbf{c}^{[t]} \in \mathbb{R}^n$

$\odot$ Hadamard or element-wise product

<u>New content</u>: similarly to simple RNN, there is an input sequence $\mathbf{u}^{[1]}, \ldots, \mathbf{u}^{[t]}$ and there is a dependency to the previous hidden state $\mathbf{h}^{[t-1]}$

$$\tilde{\mathbf{c}}^{[t]} = \tanh\left(\mathbf{U}_c \cdot \mathbf{u}^{[t]} + \mathbf{W}_c \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_c\right)$$

<u>Forget gate</u>: what should be forgotten from the previous cell state; $0 \rightarrow 1$ ~ forget $\rightarrow$ keep.

$$\mathbf{f}^{[t]} = \sigma\left(\mathbf{U}_f \cdot \mathbf{u}^{[t]} + \mathbf{W}_f \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_f\right)$$

<u>Input gate</u>: what should be kept from the new content?

$$\mathbf{i}^{[t]} = \sigma\left(\mathbf{U}_i \cdot \mathbf{u}^{[t]} + \mathbf{W}_i \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_i\right)$$

<u>Cell state</u>: forget some past, keep some present

$$\mathbf{c}^{[t]} = \mathbf{f}^{[t]} \odot \mathbf{c}^{[t-1]} + \mathbf{i}^{[t]} \odot \tilde{\mathbf{c}}^{[t]}$$

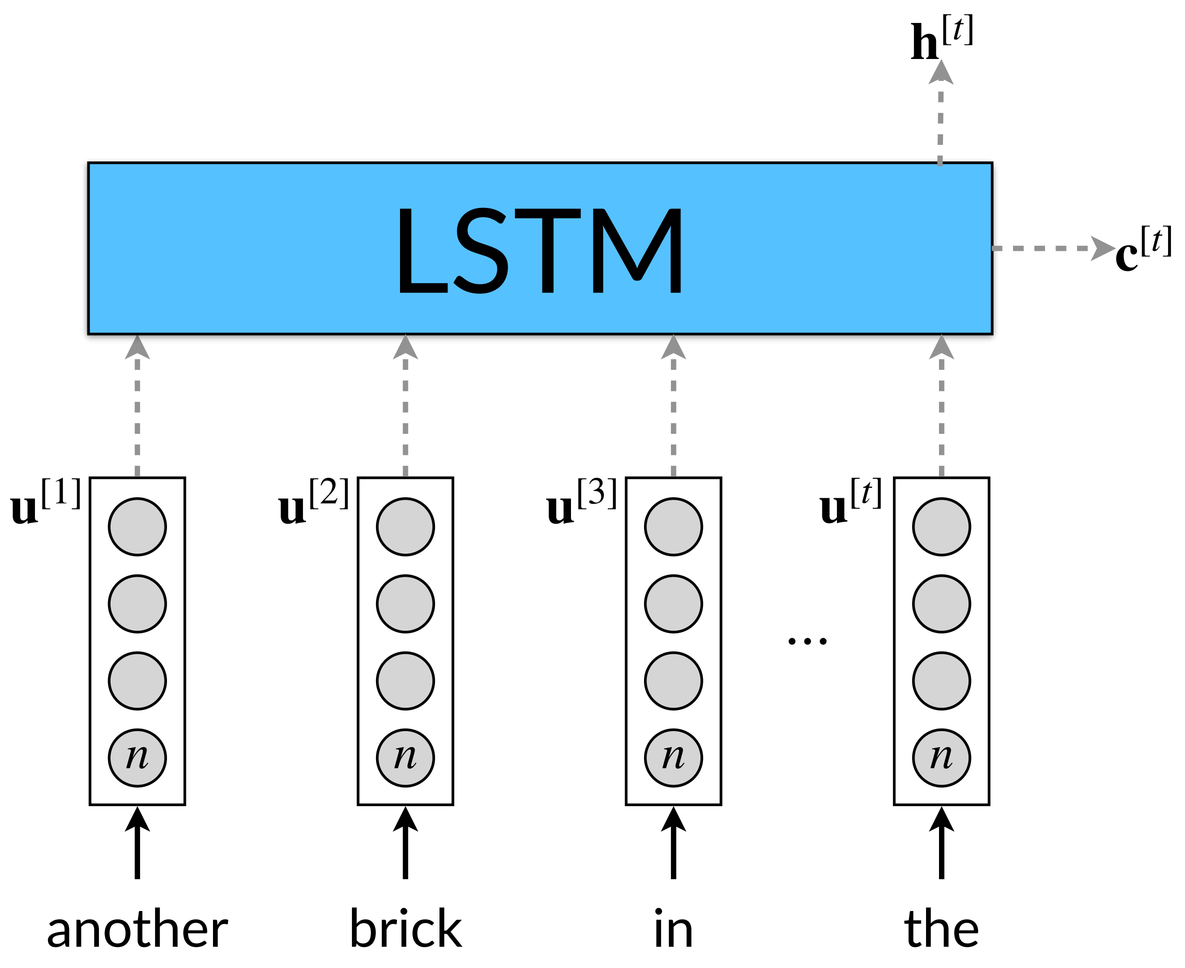
<u>Output gate</u>: what parts of the cell state will be passed on to the hidden state

$$\mathbf{o}^{[t]} = \sigma\left(\mathbf{U}_o \cdot \mathbf{u}^{[t]} + \mathbf{W}_o \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_o\right)$$

<u>Hidden state</u>: maintains the part of the cell state that will pass from the output gate

$$\mathbf{h}^{[t]} = \mathbf{o}^{[t]} \odot \tanh\left(\mathbf{c}^{[t]}\right)$$

Long Short-Term Memory (LSTM)



all gate values range from 0 to 1 given the sigmoid activation (σ)

$\mathbf{f}^{[t]}, \mathbf{i}^{[t]}, \mathbf{o}^{[t]} \in (0,1)^n$

$\tilde{\mathbf{c}}^{[t]}, \mathbf{h}^{[t]} \in (-1,1)^n$ and $\mathbf{c}^{[t]} \in \mathbb{R}^n$

$\odot$ Hadamard or element-wise product

independent from each other could be computed in parallel

New content: similarly to simple RNN, there is an input sequence $\mathbf{u}^{[1]}, \ldots, \mathbf{u}^{[t]}$ and there is a dependency to the previous hidden state $\mathbf{h}^{[t-1]}$

$$\tilde{\mathbf{c}}^{[t]} = \tanh\left(\mathbf{U}_c \cdot \mathbf{u}^{[t]} + \mathbf{W}_c \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_c\right)$$

Forget gate: what should be forgotten from the previous cell state; $0 \rightarrow 1$ ~ forget $\rightarrow$ keep.

$$\mathbf{f}^{[t]} = \sigma\left(\mathbf{U}_f \cdot \mathbf{u}^{[t]} + \mathbf{W}_f \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_f\right)$$

Input gate: what should be kept from the new content?

$$\mathbf{i}^{[t]} = \sigma\left(\mathbf{U}_i \cdot \mathbf{u}^{[t]} + \mathbf{W}_i \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_i\right)$$

Cell state: forget some past, keep some present

$$\mathbf{c}^{[t]} = \mathbf{f}^{[t]} \odot \mathbf{c}^{[t-1]} + \mathbf{i}^{[t]} \odot \tilde{\mathbf{c}}^{[t]}$$

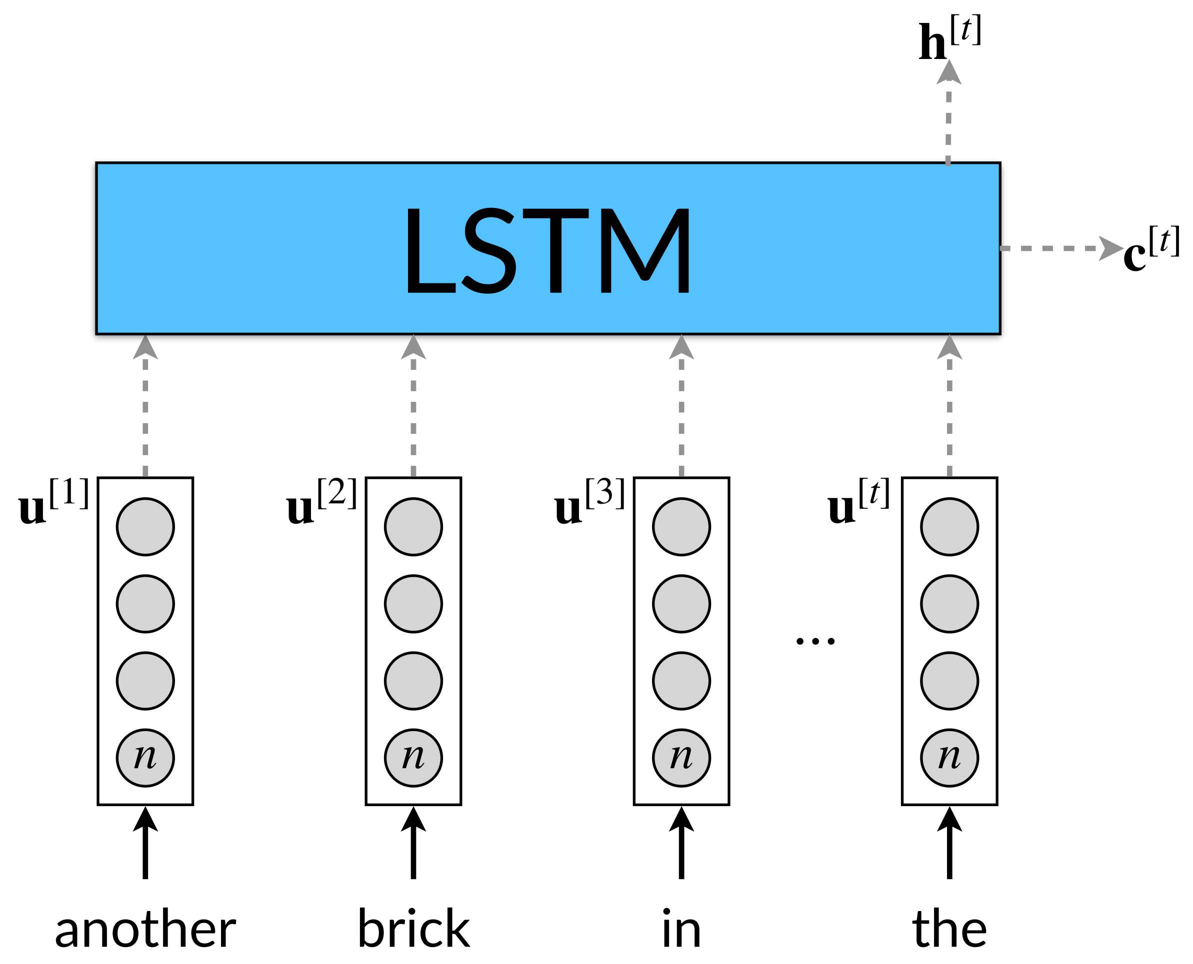
Output gate: what parts of the cell state will be passed on to the hidden state

$$\mathbf{o}^{[t]} = \sigma\left(\mathbf{U}_o \cdot \mathbf{u}^{[t]} + \mathbf{W}_o \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_o\right)$$

Hidden state: maintains the part of the cell state that will pass from the output gate

$$\mathbf{h}^{[t]} = \mathbf{o}^{[t]} \odot \tanh\left(\mathbf{c}^{[t]}\right)$$

Long Short-Term Memory (LSTM)



all gate values range from 0 to 1 given the sigmoid activation (σ)

$\mathbf{f}^{[t]}, \mathbf{i}^{[t]}, \mathbf{o}^{[t]} \in (0,1)^n$

$\tilde{\mathbf{c}}^{[t]}, \mathbf{h}^{[t]} \in (-1,1)^n$ and $\mathbf{c}^{[t]} \in \mathbb{R}^n$

$\odot$ Hadamard or element-wise product

independent from each other could be computed in parallel

New content: similarly to simple RNN, there is an input sequence $\mathbf{u}^{[1]}, \ldots, \mathbf{u}^{[t]}$ and there is a dependency to the previous hidden state $\mathbf{h}^{[t-1]}$

$$\tilde{\mathbf{c}}^{[t]} = \tanh\left(\mathbf{U}_c \cdot \mathbf{u}^{[t]} + \mathbf{W}_c \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_c\right)$$

Forget gate: what should be forgotten from the previous cell state; $0 \rightarrow 1$ ~ forget $\rightarrow$ keep.

$$\mathbf{f}^{[t]} = \sigma\left(\mathbf{U}_f \cdot \mathbf{u}^{[t]} + \mathbf{W}_f \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_f\right)$$

Input gate: what should be kept from the new content?

$$\mathbf{i}^{[t]} = \sigma\left(\mathbf{U}_i \cdot \mathbf{u}^{[t]} + \mathbf{W}_i \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_i\right)$$

Cell state: forget some past, keep some present

$$\mathbf{c}^{[t]} = \mathbf{f}^{[t]} \odot \mathbf{c}^{[t-1]} + \mathbf{i}^{[t]} \odot \tilde{\mathbf{c}}^{[t]}$$

Output gate: what parts of the cell state will be passed on to the hidden state

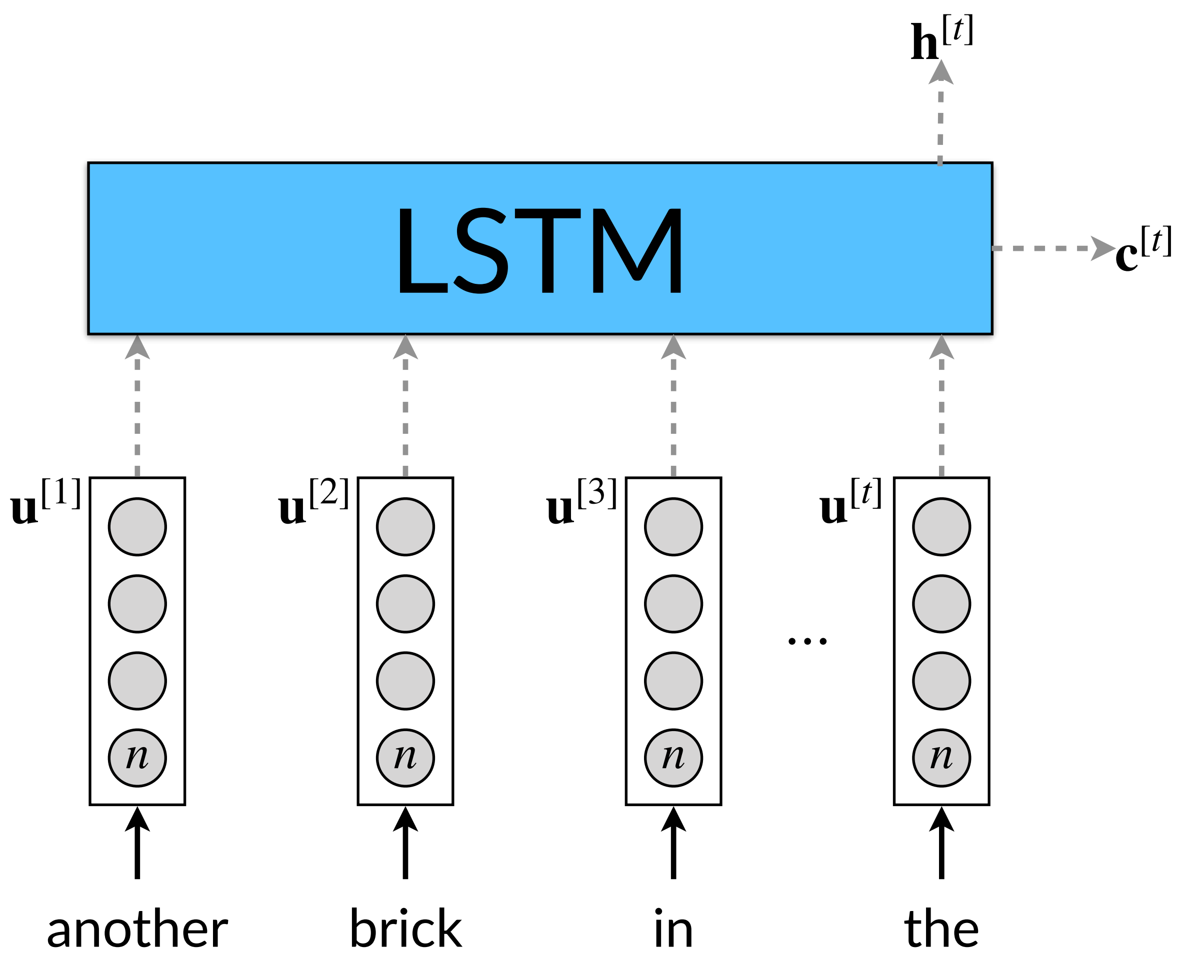
$$\mathbf{o}^{[t]} = \sigma\left(\mathbf{U}_o \cdot \mathbf{u}^{[t]} + \mathbf{W}_o \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_o\right)$$

Hidden state: maintains the part of the cell state that will pass from the output gate

$$\mathbf{h}^{[t]} = \mathbf{o}^{[t]} \odot \tanh\left(\mathbf{c}^{[t]}\right)$$

If $\mathbf{u}^{[t]} \in \mathbb{R}^m$, how many parameters?

Long Short-Term Memory (LSTM)



independent from each other could be computed in parallel

New content: similarly to simple RNN, there is an input sequence $\mathbf{u}^{[1]}, \ldots, \mathbf{u}^{[t]}$ and there is a dependency to the previous hidden state $\mathbf{h}^{[t-1]}$

$$\tilde{\mathbf{c}}^{[t]} = \tanh\left(\mathbf{U}_c \cdot \mathbf{u}^{[t]} + \mathbf{W}_c \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_c\right)$$

Forget gate: what should be forgotten from the previous cell state; $0 \rightarrow 1$ ~ forget $\rightarrow$ keep.

$$\mathbf{f}^{[t]} = \sigma\left(\mathbf{U}_f \cdot \mathbf{u}^{[t]} + \mathbf{W}_f \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_f\right)$$

Input gate: what should be kept from the new content?

$$\mathbf{i}^{[t]} = \sigma\left(\mathbf{U}_i \cdot \mathbf{u}^{[t]} + \mathbf{W}_i \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_i\right)$$

Cell state: forget some past, keep some present

$$\mathbf{c}^{[t]} = \mathbf{f}^{[t]} \odot \mathbf{c}^{[t-1]} + \mathbf{i}^{[t]} \odot \tilde{\mathbf{c}}^{[t]}$$

Output gate: what parts of the cell state will be passed on to the hidden state

$$\mathbf{o}^{[t]} = \sigma\left(\mathbf{U}_o \cdot \mathbf{u}^{[t]} + \mathbf{W}_o \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_o\right)$$

Hidden state: maintains the part of the cell state that will pass from the output gate

$$\mathbf{h}^{[t]} = \mathbf{o}^{[t]} \odot \tanh\left(\mathbf{c}^{[t]}\right)$$

all gate values range from 0 to 1 given the sigmoid activation (σ)

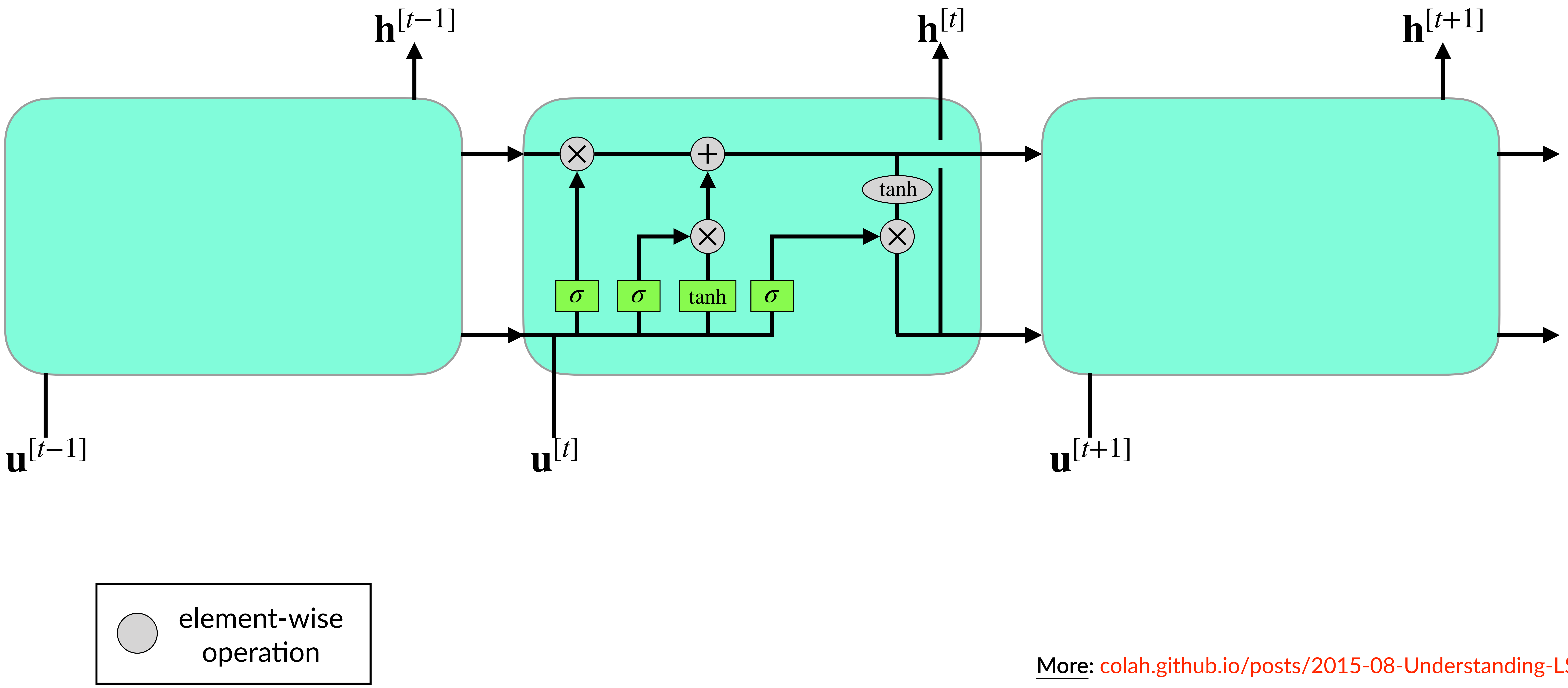
$\mathbf{f}^{[t]}, \mathbf{i}^{[t]}, \mathbf{o}^{[t]} \in (0,1)^n$

$\tilde{\mathbf{c}}^{[t]}, \mathbf{h}^{[t]} \in (-1,1)^n$ and $\mathbf{c}^{[t]} \in \mathbb{R}^n$

$\odot$ Hadamard or element-wise product

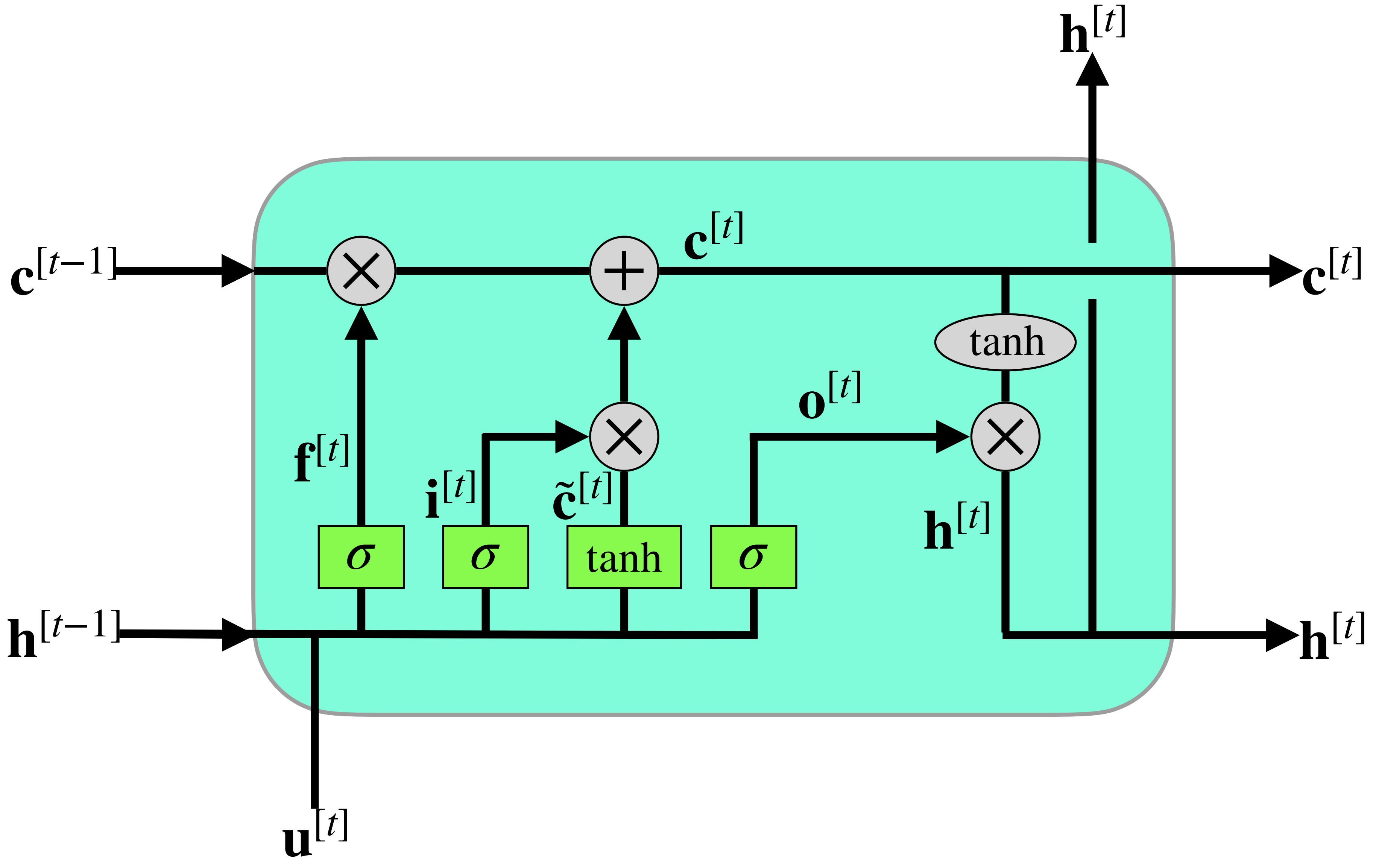
If $\mathbf{u}^{[t]} \in \mathbb{R}^m$, how many parameters? $= 4 \cdot n \cdot (m + n + 1)$

The LSTM (*confusing/artistic*) schematic



More: colah.github.io/posts/2015-08-Understanding-LSTMs/

The LSTM (*confusing/artistic*) schematic



$$\tilde{\mathbf{c}}^{[t]} = \tanh\left(\mathbf{U}_c \cdot \mathbf{u}^{[t]} + \mathbf{W}_c \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_c\right)$$

$$\mathbf{f}^{[t]} = \sigma\left(\mathbf{U}_f \cdot \mathbf{u}^{[t]} + \mathbf{W}_f \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_f\right)$$

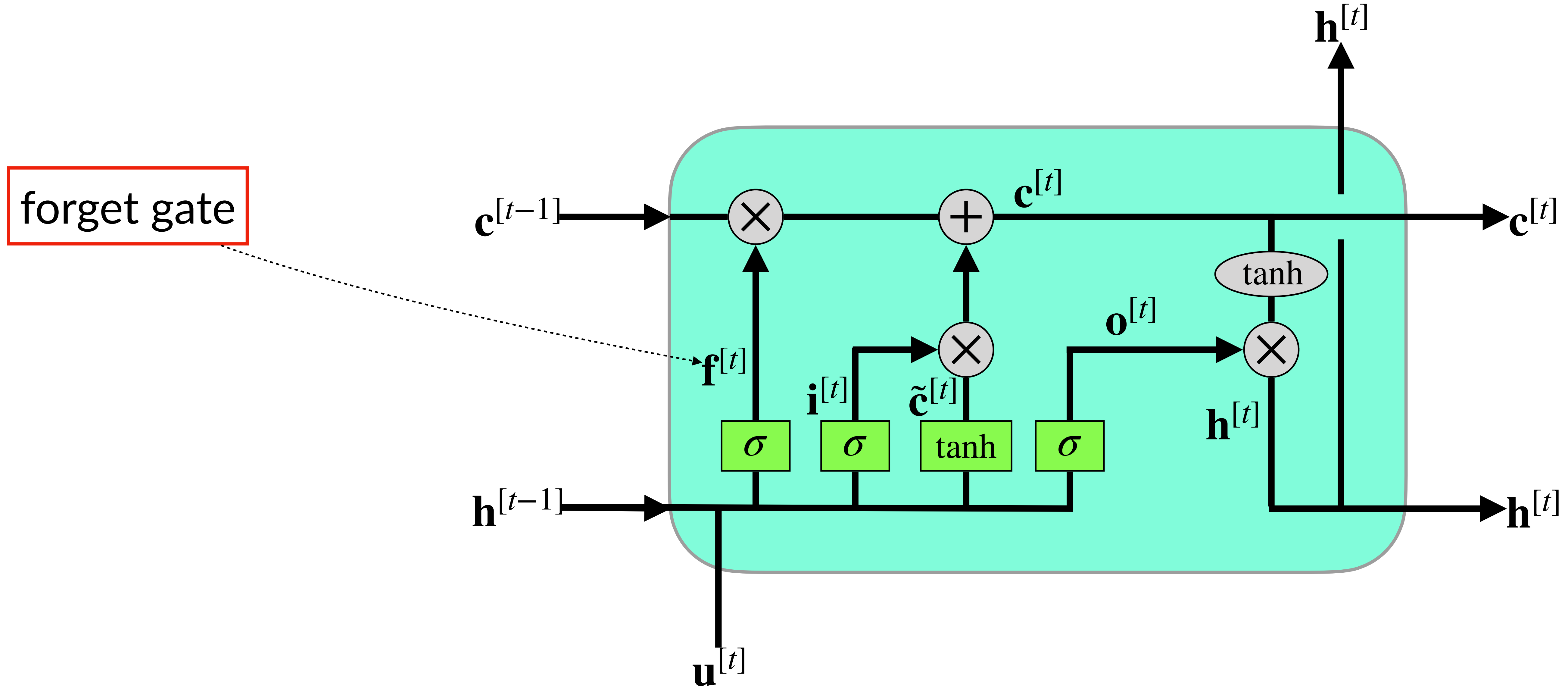
$$\mathbf{i}^{[t]} = \sigma\left(\mathbf{U}_i \cdot \mathbf{u}^{[t]} + \mathbf{W}_i \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_i\right)$$

$$\mathbf{c}^{[t]} = \mathbf{f}^{[t]} \odot \mathbf{c}^{[t-1]} + \mathbf{i}^{[t]} \odot \tilde{\mathbf{c}}^{[t]}$$

$$\mathbf{o}^{[t]} = \sigma\left(\mathbf{U}_o \cdot \mathbf{u}^{[t]} + \mathbf{W}_o \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_o\right)$$

$$\mathbf{h}^{[t]} = \mathbf{o}^{[t]} \odot \tanh\left(\mathbf{c}^{[t]}\right)$$

The LSTM (*confusing/artistic*) schematic



$$\tilde{\mathbf{c}}^{[t]} = \tanh\left(\mathbf{U}_c \cdot \mathbf{u}^{[t]} + \mathbf{W}_c \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_c\right)$$

$$\mathbf{f}^{[t]} = \sigma\left(\mathbf{U}_f \cdot \mathbf{u}^{[t]} + \mathbf{W}_f \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_f\right)$$

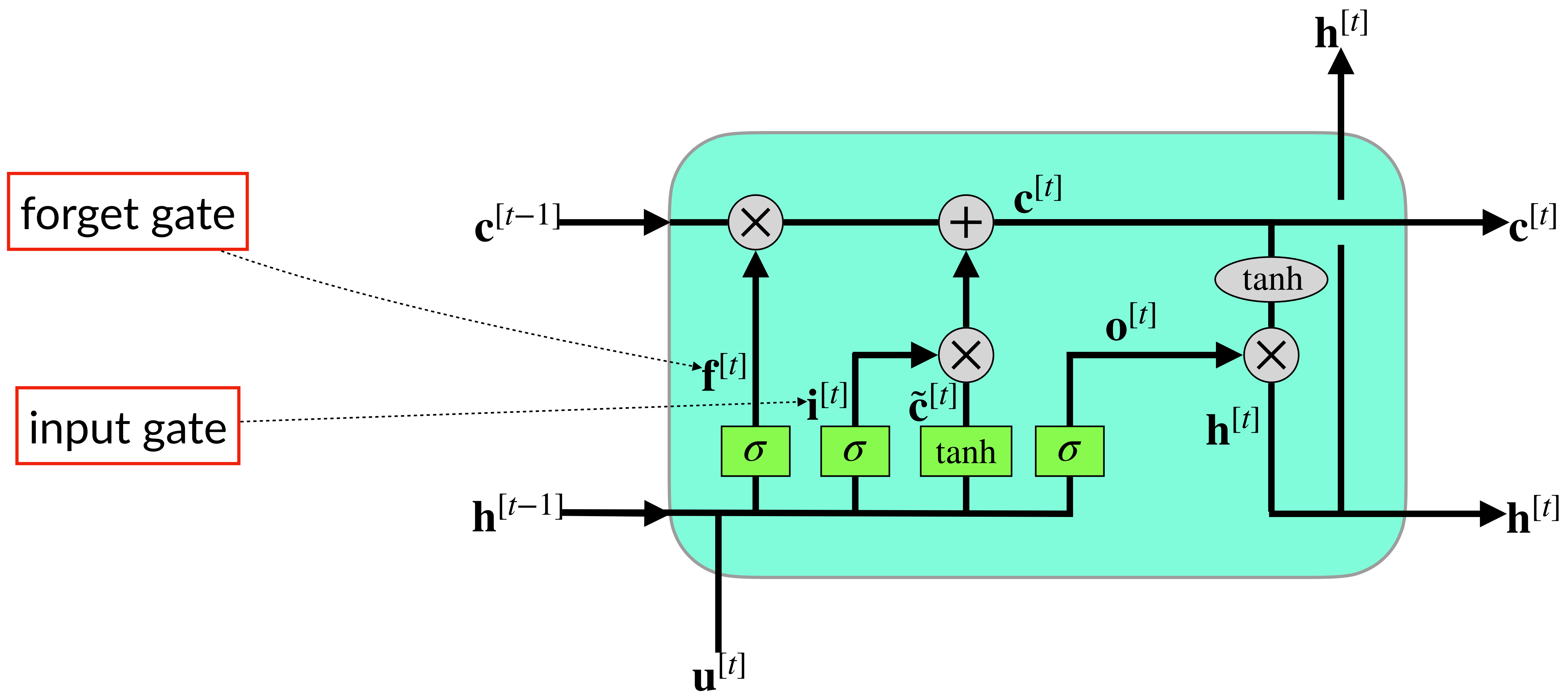
$$\mathbf{i}^{[t]} = \sigma\left(\mathbf{U}_i \cdot \mathbf{u}^{[t]} + \mathbf{W}_i \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_i\right)$$

$$\mathbf{c}^{[t]} = \mathbf{f}^{[t]} \odot \mathbf{c}^{[t-1]} + \mathbf{i}^{[t]} \odot \tilde{\mathbf{c}}^{[t]}$$

$$\mathbf{o}^{[t]} = \sigma\left(\mathbf{U}_o \cdot \mathbf{u}^{[t]} + \mathbf{W}_o \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_o\right)$$

$$\mathbf{h}^{[t]} = \mathbf{o}^{[t]} \odot \tanh\left(\mathbf{c}^{[t]}\right)$$

The LSTM (*confusing/artistic*) schematic



$$\tilde{\mathbf{c}}^{[t]} = \tanh\left(\mathbf{U}_c \cdot \mathbf{u}^{[t]} + \mathbf{W}_c \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_c\right)$$

$$\mathbf{f}^{[t]} = \sigma\left(\mathbf{U}_f \cdot \mathbf{u}^{[t]} + \mathbf{W}_f \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_f\right)$$

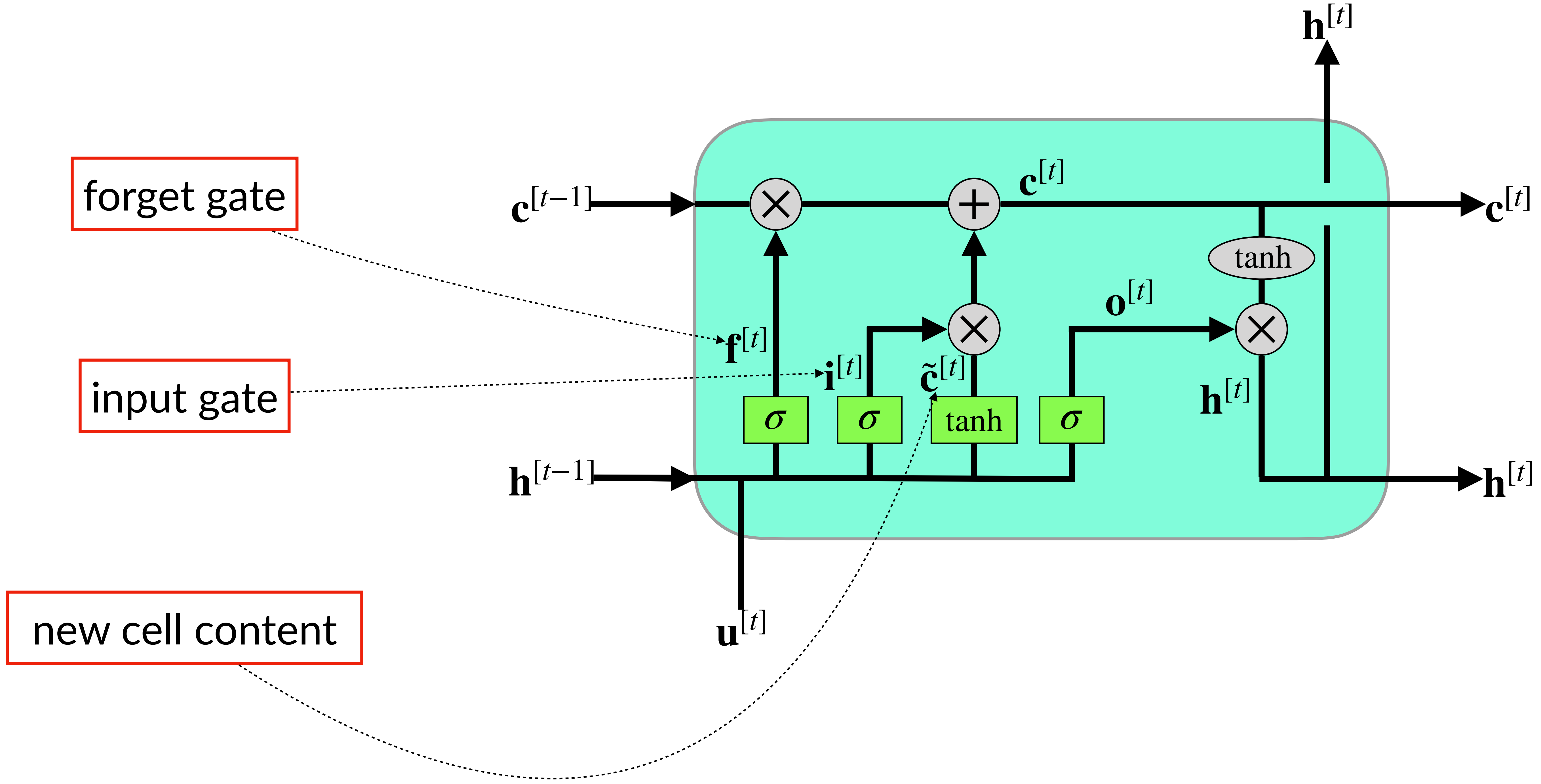
$$\mathbf{i}^{[t]} = \sigma\left(\mathbf{U}_i \cdot \mathbf{u}^{[t]} + \mathbf{W}_i \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_i\right)$$

$$\mathbf{c}^{[t]} = \mathbf{f}^{[t]} \odot \mathbf{c}^{[t-1]} + \mathbf{i}^{[t]} \odot \tilde{\mathbf{c}}^{[t]}$$

$$\mathbf{o}^{[t]} = \sigma\left(\mathbf{U}_o \cdot \mathbf{u}^{[t]} + \mathbf{W}_o \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_o\right)$$

$$\mathbf{h}^{[t]} = \mathbf{o}^{[t]} \odot \tanh\left(\mathbf{c}^{[t]}\right)$$

The LSTM (*confusing/artistic*) schematic



$$\tilde{\mathbf{c}}^{[t]} = \tanh\left(\mathbf{U}_c \cdot \mathbf{u}^{[t]} + \mathbf{W}_c \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_c\right)$$

$$\mathbf{f}^{[t]} = \sigma\left(\mathbf{U}_f \cdot \mathbf{u}^{[t]} + \mathbf{W}_f \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_f\right)$$

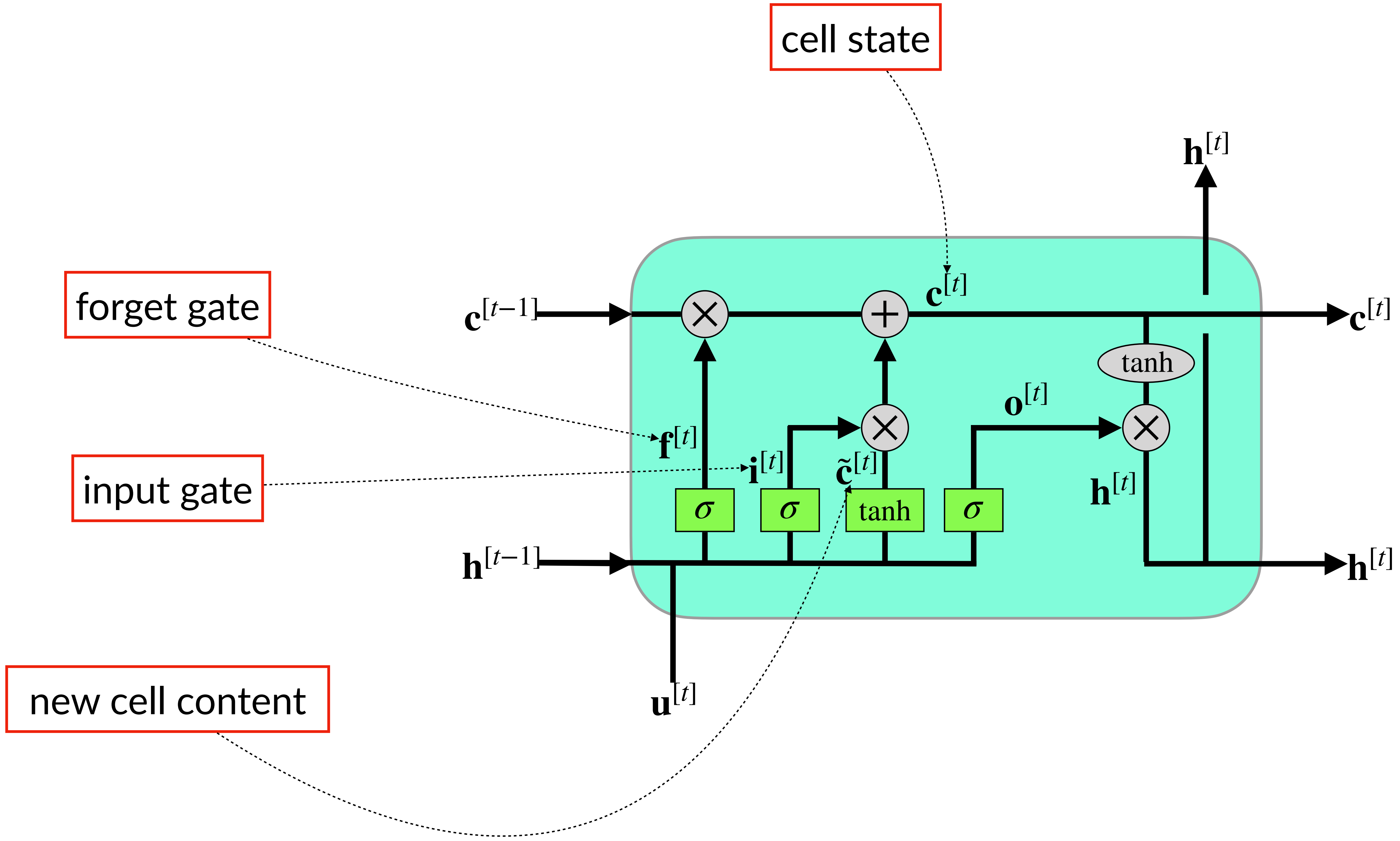
$$\mathbf{i}^{[t]} = \sigma\left(\mathbf{U}_i \cdot \mathbf{u}^{[t]} + \mathbf{W}_i \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_i\right)$$

$$\mathbf{c}^{[t]} = \mathbf{f}^{[t]} \odot \mathbf{c}^{[t-1]} + \mathbf{i}^{[t]} \odot \tilde{\mathbf{c}}^{[t]}$$

$$\mathbf{o}^{[t]} = \sigma\left(\mathbf{U}_o \cdot \mathbf{u}^{[t]} + \mathbf{W}_o \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_o\right)$$

$$\mathbf{h}^{[t]} = \mathbf{o}^{[t]} \odot \tanh\left(\mathbf{c}^{[t]}\right)$$

The LSTM (*confusing/artistic*) schematic



$$\tilde{\mathbf{c}}^{[t]} = \tanh\left(\mathbf{U}_c \cdot \mathbf{u}^{[t]} + \mathbf{W}_c \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_c\right)$$

$$\mathbf{f}^{[t]} = \sigma\left(\mathbf{U}_f \cdot \mathbf{u}^{[t]} + \mathbf{W}_f \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_f\right)$$

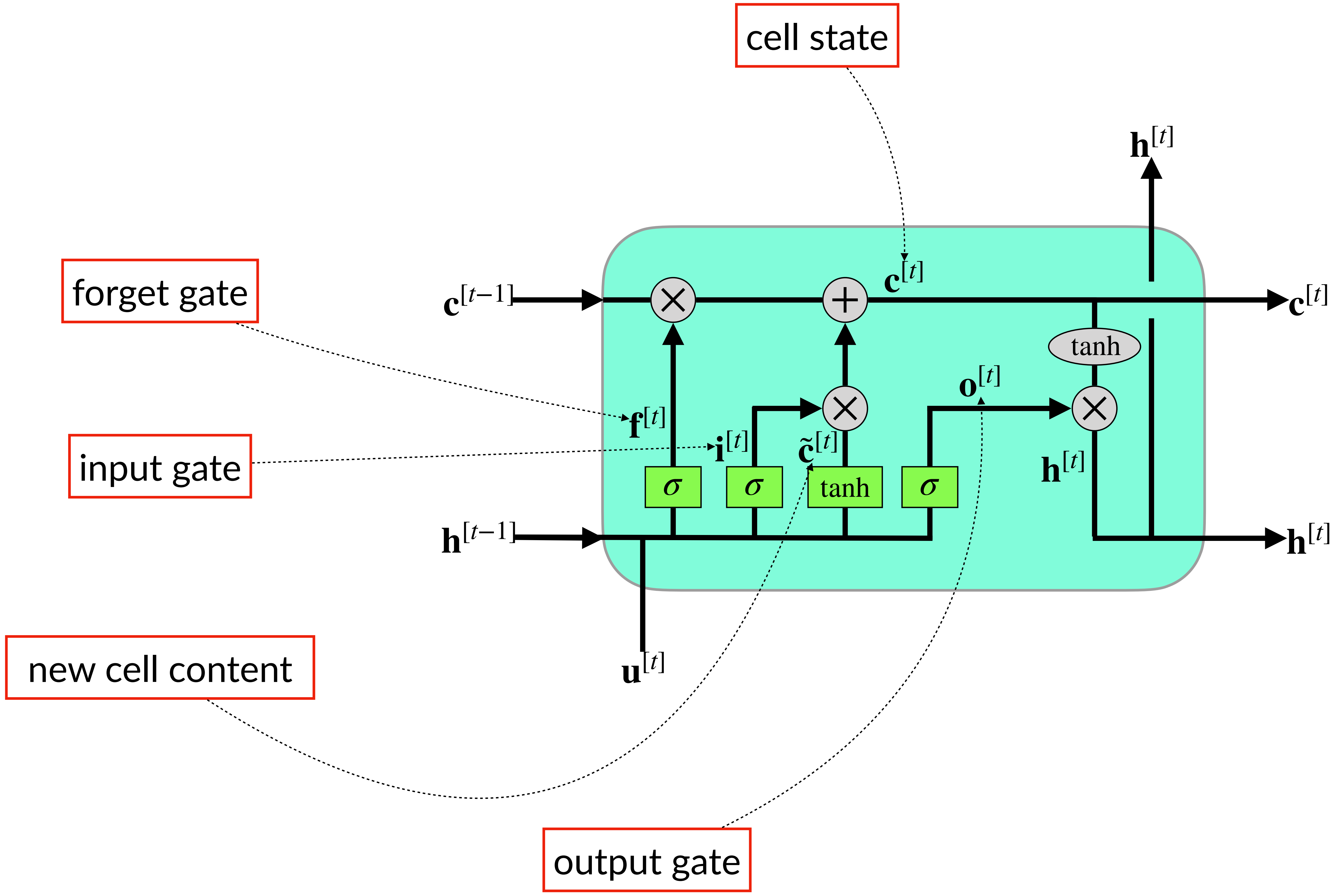
$$\mathbf{i}^{[t]} = \sigma\left(\mathbf{U}_i \cdot \mathbf{u}^{[t]} + \mathbf{W}_i \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_i\right)$$

$$\mathbf{c}^{[t]} = \mathbf{f}^{[t]} \odot \mathbf{c}^{[t-1]} + \mathbf{i}^{[t]} \odot \tilde{\mathbf{c}}^{[t]}$$

$$\mathbf{o}^{[t]} = \sigma\left(\mathbf{U}_o \cdot \mathbf{u}^{[t]} + \mathbf{W}_o \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_o\right)$$

$$\mathbf{h}^{[t]} = \mathbf{o}^{[t]} \odot \tanh\left(\mathbf{c}^{[t]}\right)$$

The LSTM (*confusing/artistic*) schematic



$$\tilde{\mathbf{c}}^{[t]} = \tanh\left(\mathbf{U}_c \cdot \mathbf{u}^{[t]} + \mathbf{W}_c \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_c\right)$$

$$\mathbf{f}^{[t]} = \sigma\left(\mathbf{U}_f \cdot \mathbf{u}^{[t]} + \mathbf{W}_f \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_f\right)$$

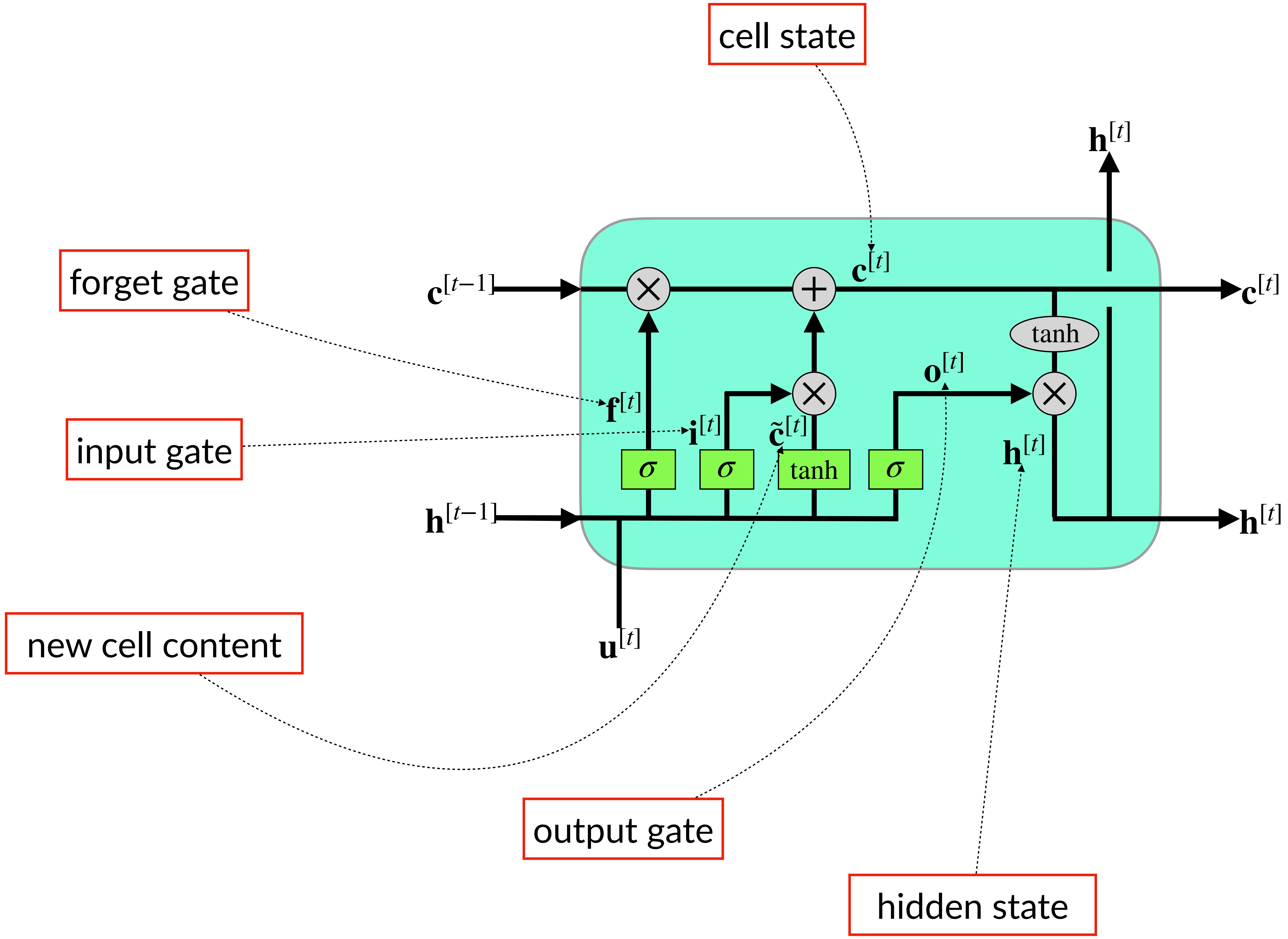
$$\mathbf{i}^{[t]} = \sigma\left(\mathbf{U}_i \cdot \mathbf{u}^{[t]} + \mathbf{W}_i \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_i\right)$$

$$\mathbf{c}^{[t]} = \mathbf{f}^{[t]} \odot \mathbf{c}^{[t-1]} + \mathbf{i}^{[t]} \odot \tilde{\mathbf{c}}^{[t]}$$

$$\mathbf{o}^{[t]} = \sigma\left(\mathbf{U}_o \cdot \mathbf{u}^{[t]} + \mathbf{W}_o \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_o\right)$$

$$\mathbf{h}^{[t]} = \mathbf{o}^{[t]} \odot \tanh\left(\mathbf{c}^{[t]}\right)$$

The LSTM (*confusing/artistic*) schematic



$$\tilde{\mathbf{c}}^{[t]} = \tanh\left(\mathbf{U}_c \cdot \mathbf{u}^{[t]} + \mathbf{W}_c \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_c\right)$$

$$\mathbf{f}^{[t]} = \sigma\left(\mathbf{U}_f \cdot \mathbf{u}^{[t]} + \mathbf{W}_f \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_f\right)$$

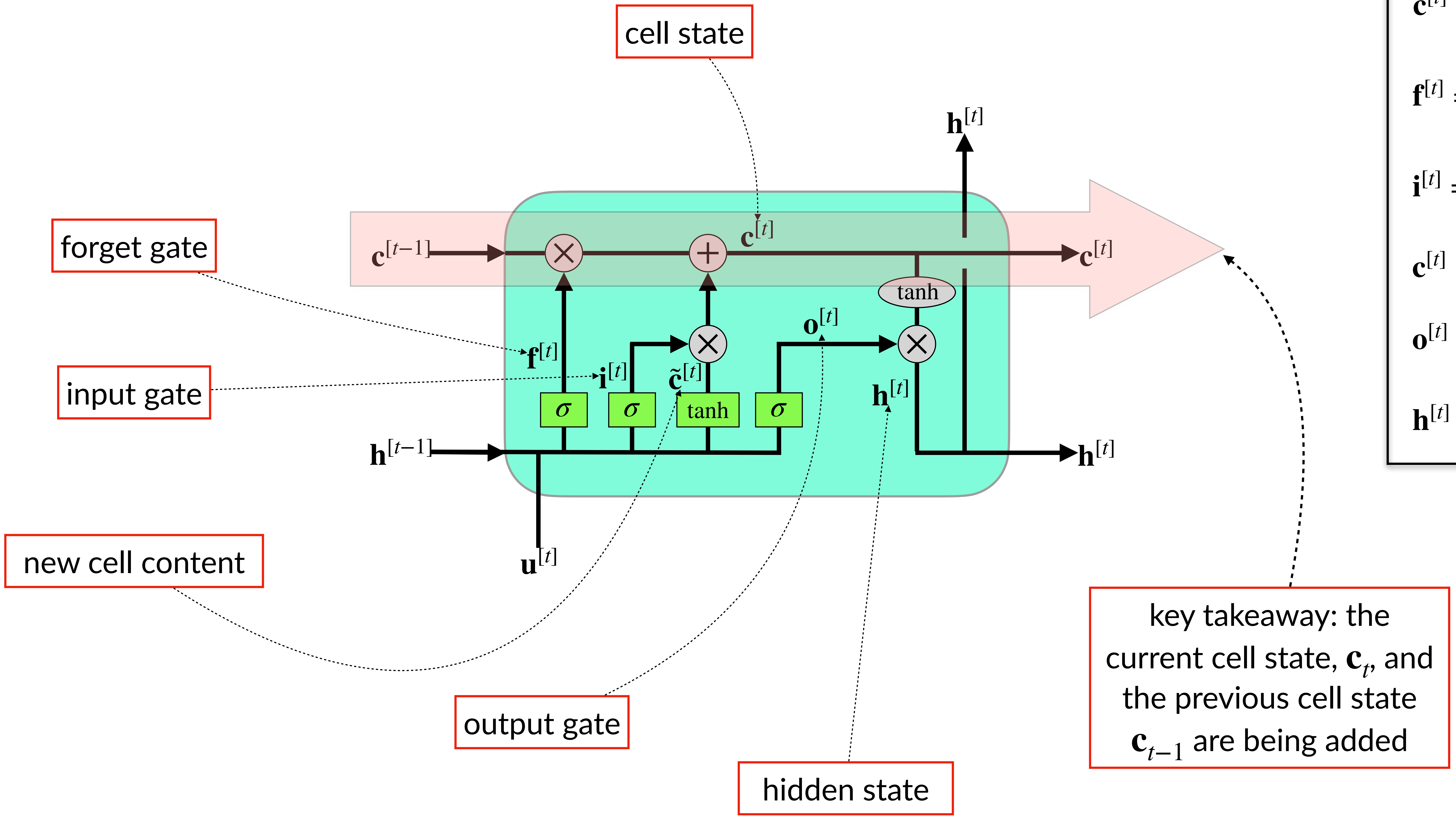
$$\mathbf{i}^{[t]} = \sigma\left(\mathbf{U}_i \cdot \mathbf{u}^{[t]} + \mathbf{W}_i \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_i\right)$$

$$\mathbf{c}^{[t]} = \mathbf{f}^{[t]} \odot \mathbf{c}^{[t-1]} + \mathbf{i}^{[t]} \odot \tilde{\mathbf{c}}^{[t]}$$

$$\mathbf{o}^{[t]} = \sigma\left(\mathbf{U}_o \cdot \mathbf{u}^{[t]} + \mathbf{W}_o \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_o\right)$$

$$\mathbf{h}^{[t]} = \mathbf{o}^{[t]} \odot \tanh\left(\mathbf{c}^{[t]}\right)$$

The LSTM (*confusing/artistic*) schematic



$$\tilde{\mathbf{c}}^{[t]} = \tanh\left(\mathbf{U}_c \cdot \mathbf{u}^{[t]} + \mathbf{W}_c \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_c\right)$$

$$\mathbf{f}^{[t]} = \sigma\left(\mathbf{U}_f \cdot \mathbf{u}^{[t]} + \mathbf{W}_f \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_f\right)$$

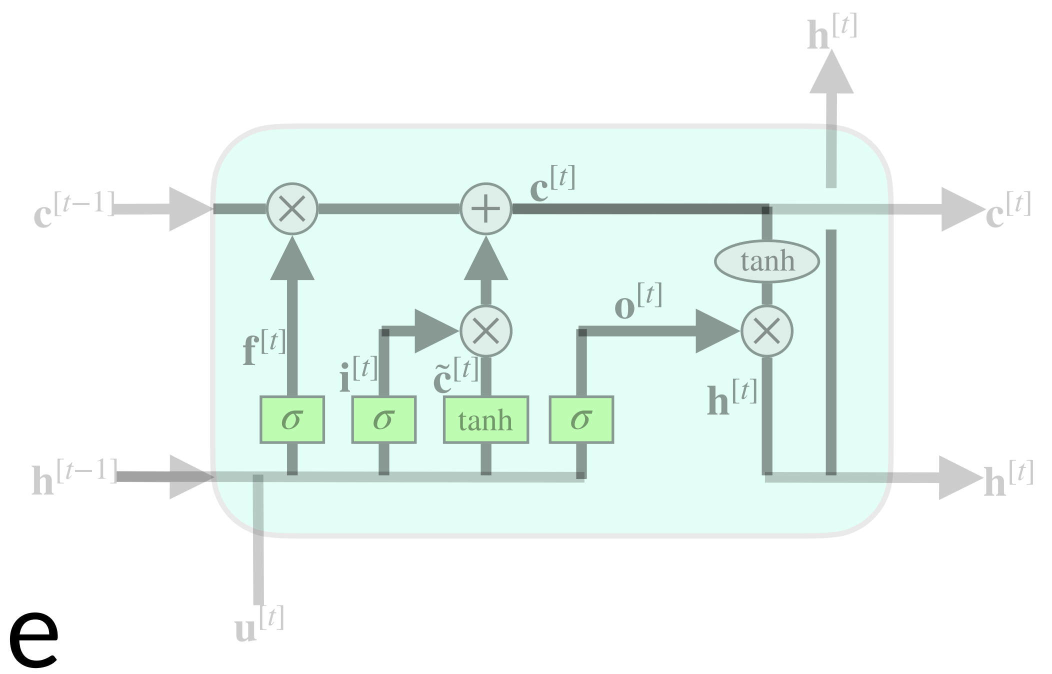
$$\mathbf{i}^{[t]} = \sigma\left(\mathbf{U}_i \cdot \mathbf{u}^{[t]} + \mathbf{W}_i \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_i\right)$$

$$\mathbf{c}^{[t]} = \mathbf{f}^{[t]} \odot \mathbf{c}^{[t-1]} + \mathbf{i}^{[t]} \odot \tilde{\mathbf{c}}^{[t]}$$

$$\mathbf{o}^{[t]} = \sigma\left(\mathbf{U}_o \cdot \mathbf{u}^{[t]} + \mathbf{W}_o \cdot \mathbf{h}^{[t-1]} + \mathbf{b}_o\right)$$

$$\mathbf{h}^{[t]} = \mathbf{o}^{[t]} \odot \tanh\left(\mathbf{c}^{[t]}\right)$$

LSTM resolves the vanishing gradient issue



- LSTM can preserve information over many time steps using its gates
- LSTM: If the forget gate value is set to $\mathbf{f}_i^{[t]} = 1$ for a cell dimension i and the corresponding input gate value $\mathbf{i}_i^{[t]} = 0$, then the cell value from the previous time step, $\mathbf{c}_i^{[t-1]}$, is maintained intact
- Simple RNN: much harder to maintain previous state information given at least an entire row of the recurrent matrix $\mathbf{W}_h$ should be set to 1 which in turn will invalidate the entire RNN rationale: $\mathbf{h}_j^{[t]} \propto \mathbf{W}_h[j, :] \cdot \mathbf{h}^{[t-1]}$
- Depends on the task, but say an RNN can model ~10 time steps accurately, then an LSTM can probably capture ~100 time steps

Text generation with RNNs

Source: trekhleb.dev/machine-learning-experiments/#/experiments/RecipeGenerationRNN

Input: "Fish and chips"

Recipe RNN LM output

Name: Fish and chips with Broccoli and Salad of Creamy Thyme Broth

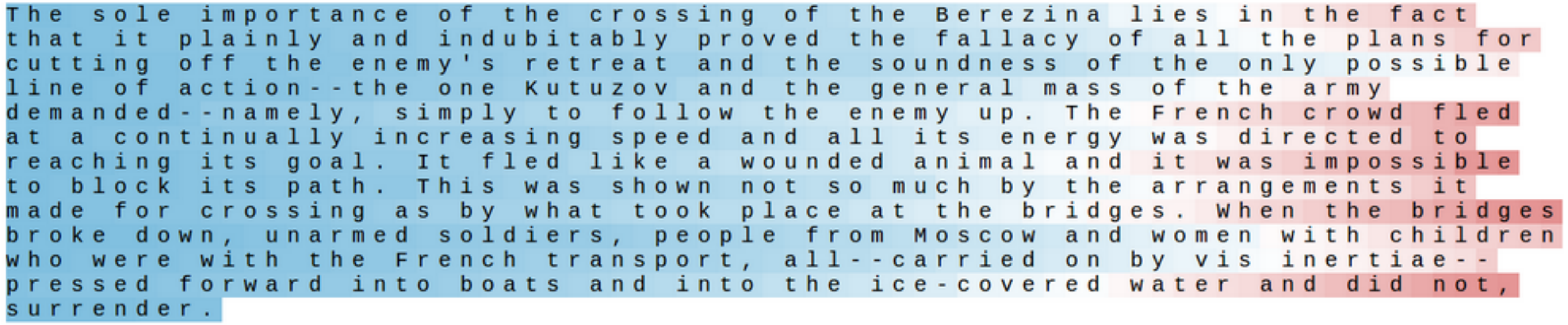
Ingredients:
- 1 cup frozen peas, thawed
- 1/4 cup chopped fresh cilantro leaves
- 1 tablespoon finely chopped fresh dill
- 1/2 cup sugar
- 1/2 cup corn tortillas
- 1 cup shredded smoked mozzarella or parmesan cheese
- 1/2 cup white wine
- 1 cup chicken broth
- Salt and pepper

Instructions: *Season salad with salt and pepper. In a large saute pan over medium-high heat, cook poblano pepper for 1 minute. Add broccoli rabe, spring onions, thyme, and bay leaves and sprinkle with salt and pepper to taste. Cook until vegetables are soft, about 10 minutes. Add the spinach and stir until completely melted. Add sugar and simmer until sauce thickens, about 1 minute. Remove from heat and stir in lemon juice. Serve with steamed roasted garlic bread.*

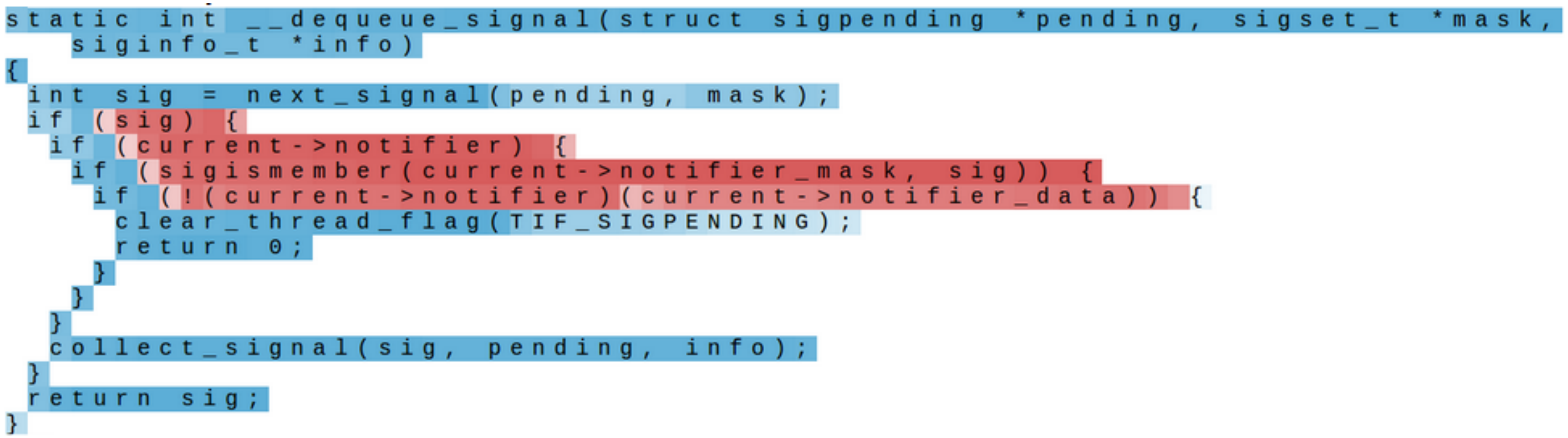
Text generation with RNNs — Trends captured by LSTM cells

Certain LSTM cells "learn" to have larger values...

towards the end of a line



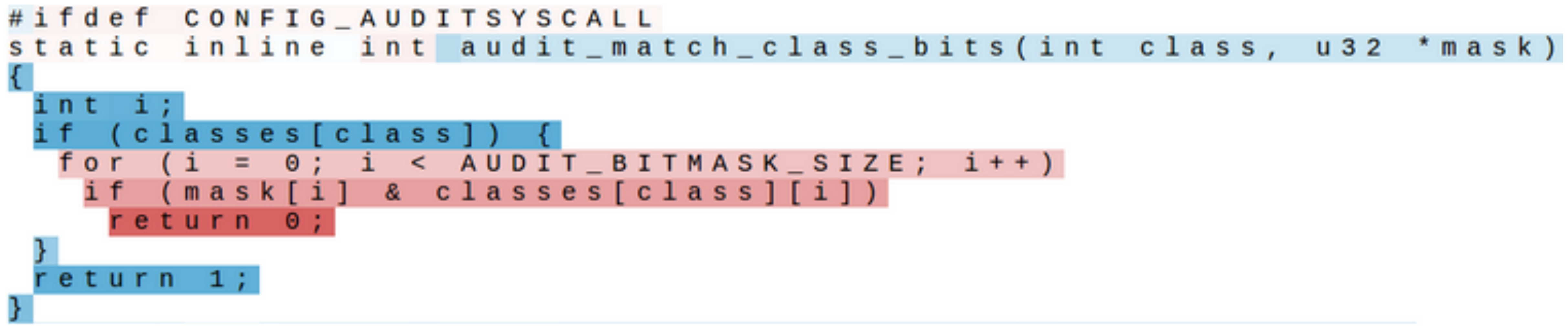
inside if statements



Source: karpathy.github.io/2015/05/21/rnn-effectiveness/

Certain LSTM cells “learn” to have larger values…

when the code expression’s depth increases

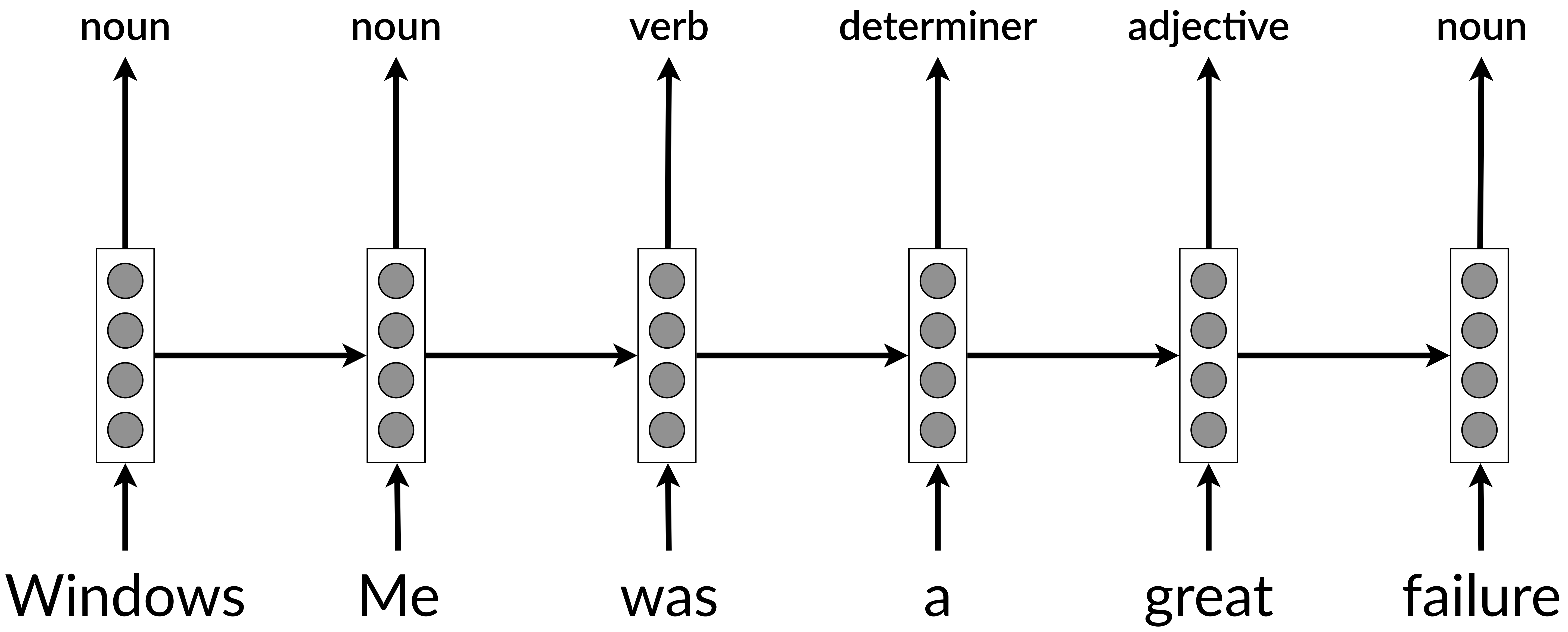


inside comments or double quotes

Source: karpathy.github.io/2015/05/21/rnn-effectiveness/

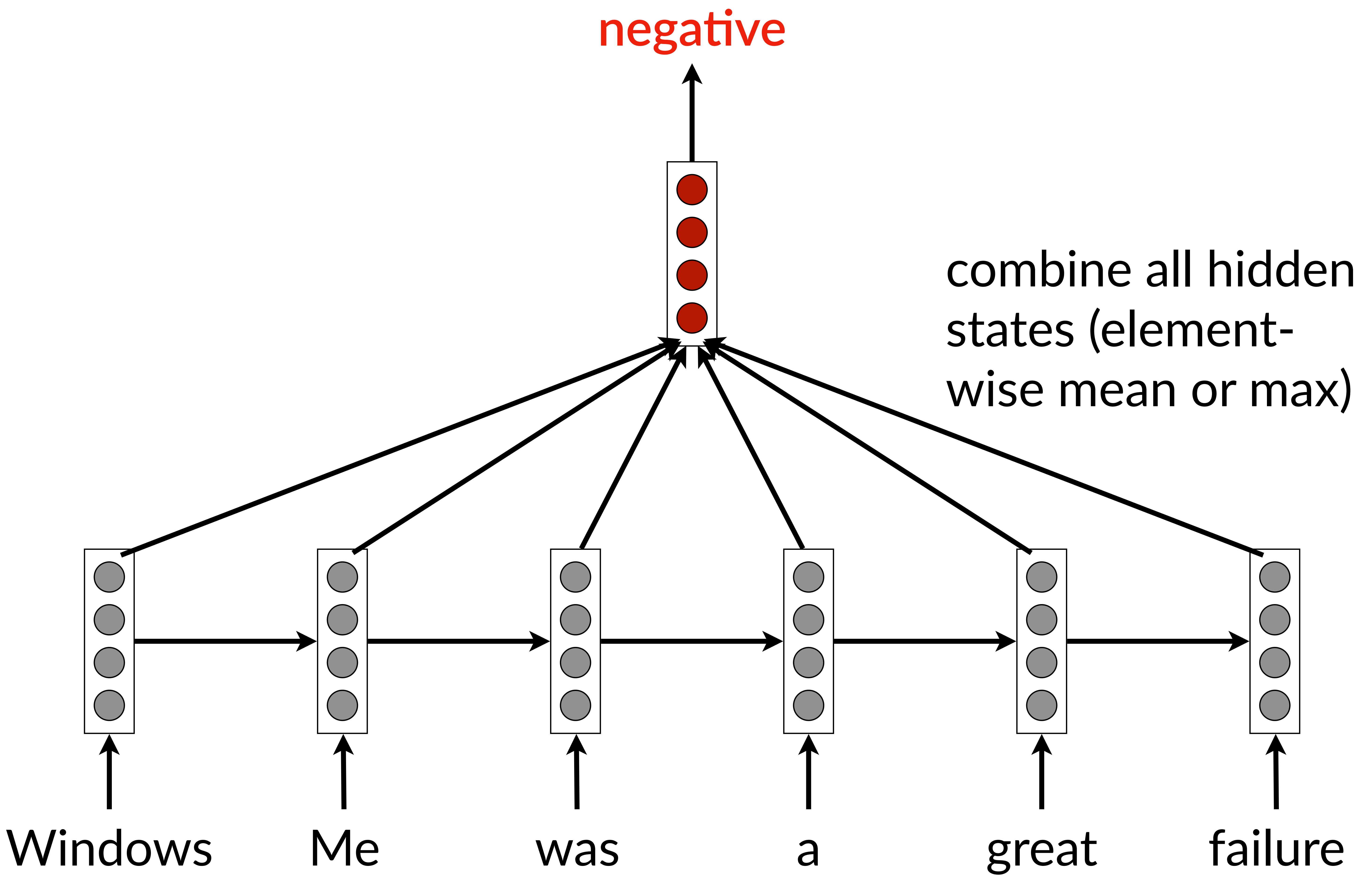
RNN applications — Sequence tagging

e.g. tasks like **part-of-speech** (POS) tagging and **named entity recognition** (NER)

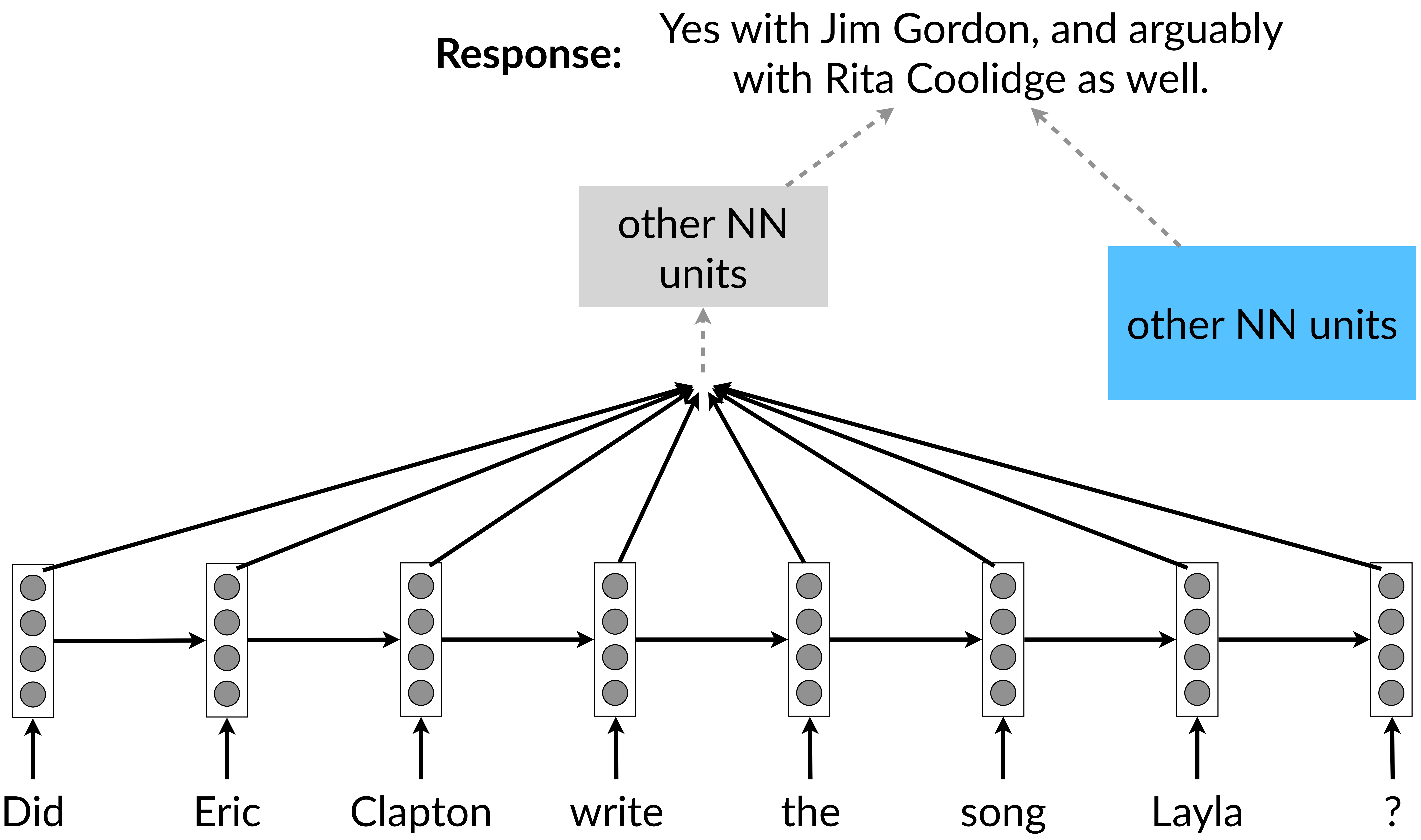


RNN applications — Sentence encoding

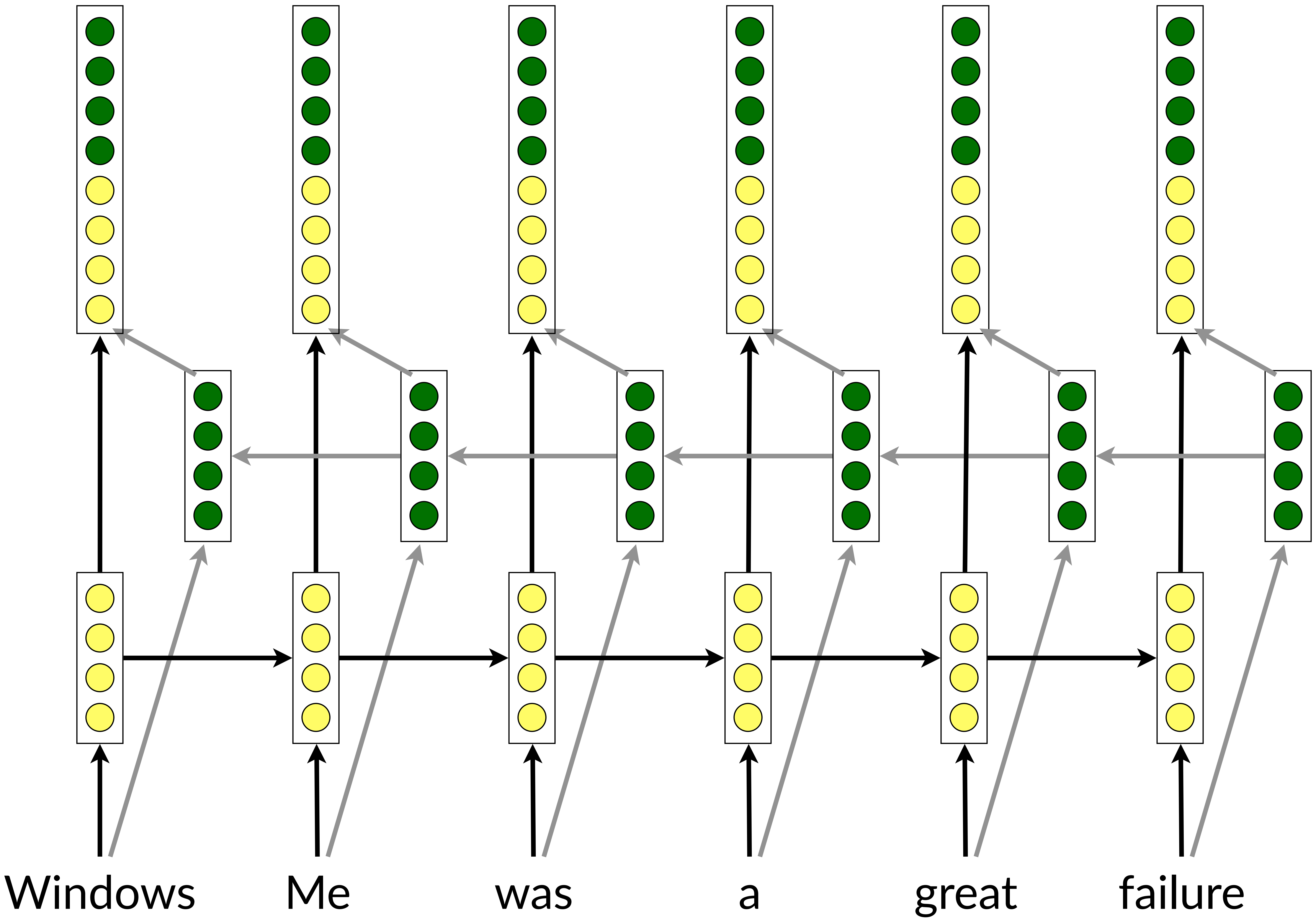
e.g. *text / sentence, sentiment classification*



RNN applications — Encoding units in larger architectures

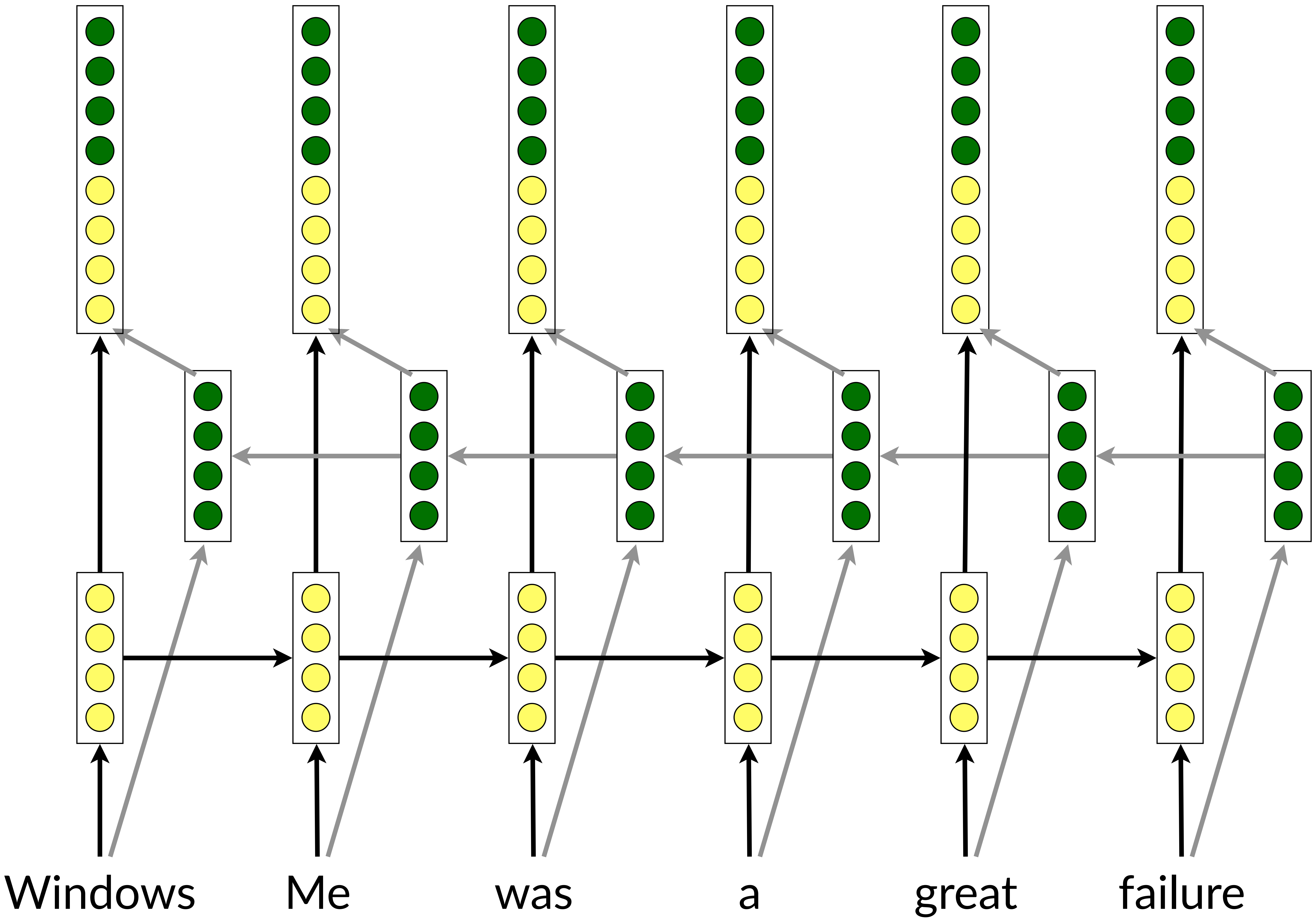


Bidirectional RNNs

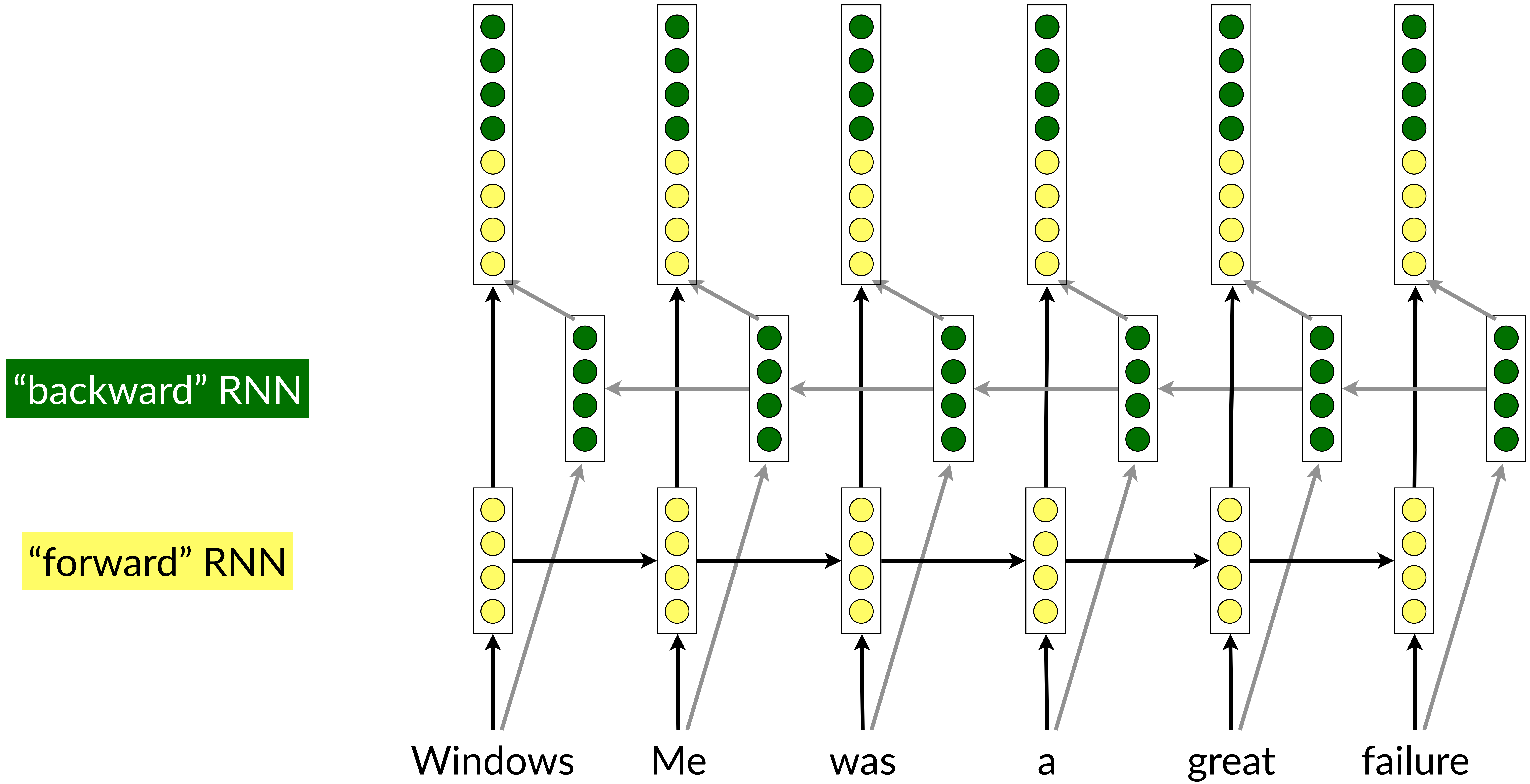


Bidirectional RNNs

“forward” RNN

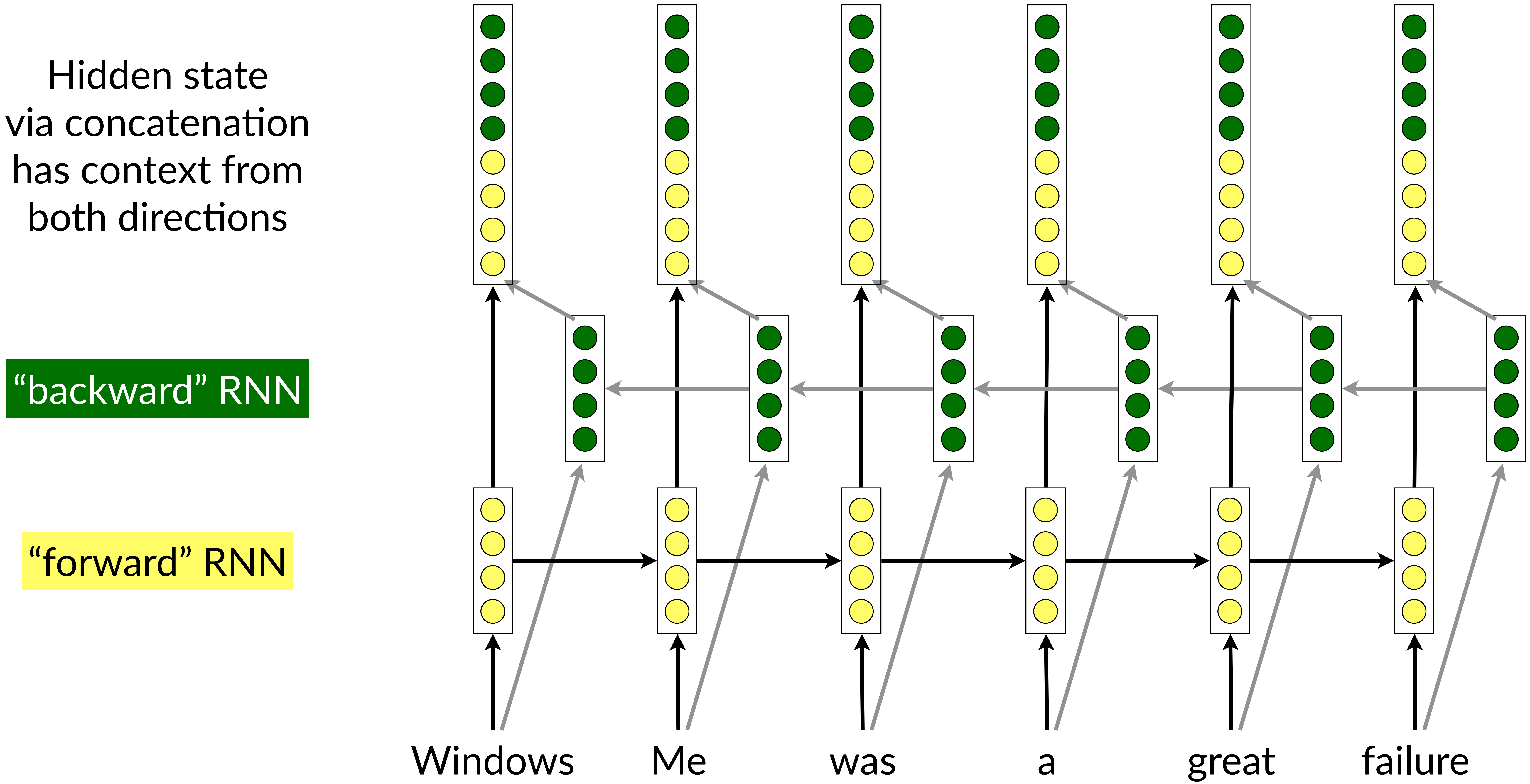


Bidirectional RNNs



Bidirectional RNNs

Hidden state via concatenation has context from both directions

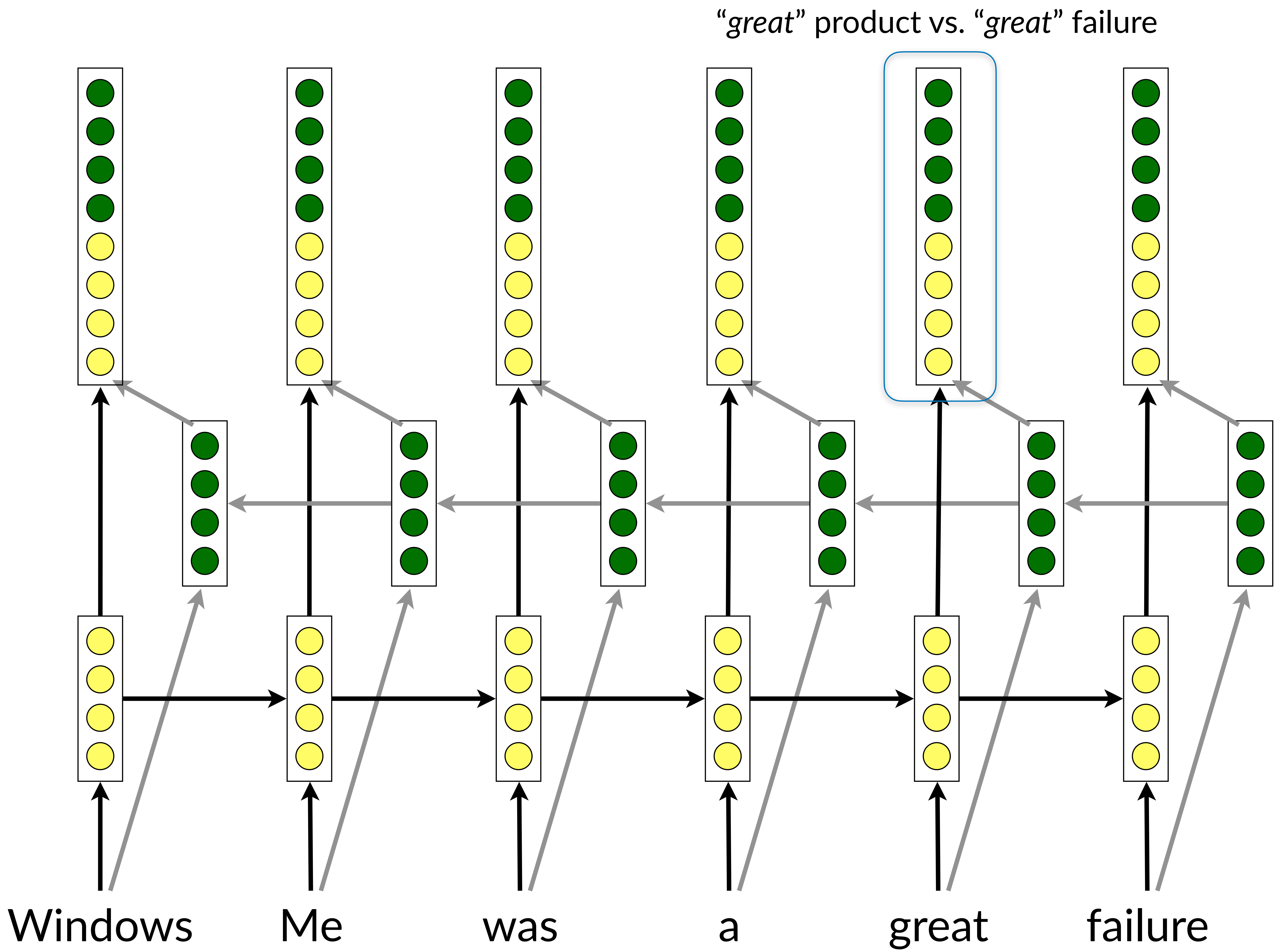


Bidirectional RNNs

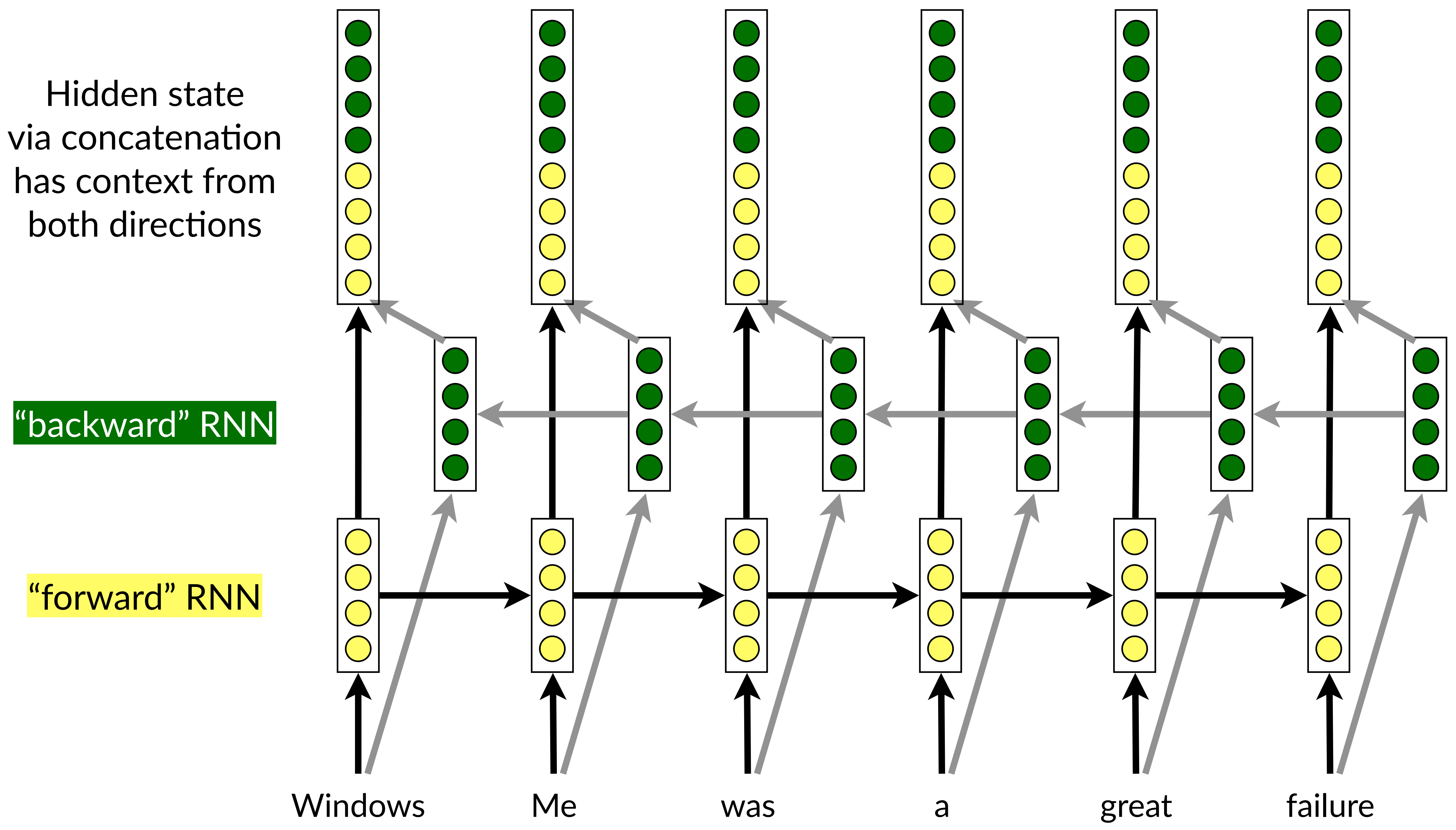
Hidden state
via concatenation
has context from
both directions

“backward” RNN

“forward” RNN



Bidirectional RNNs



hidden state of the bidirectional RNN

$$\mathbf{h}^{[t]} = \left[\overrightarrow{\mathbf{h}}^{[t]}; \overleftarrow{\mathbf{h}}^{[t]}\right]$$

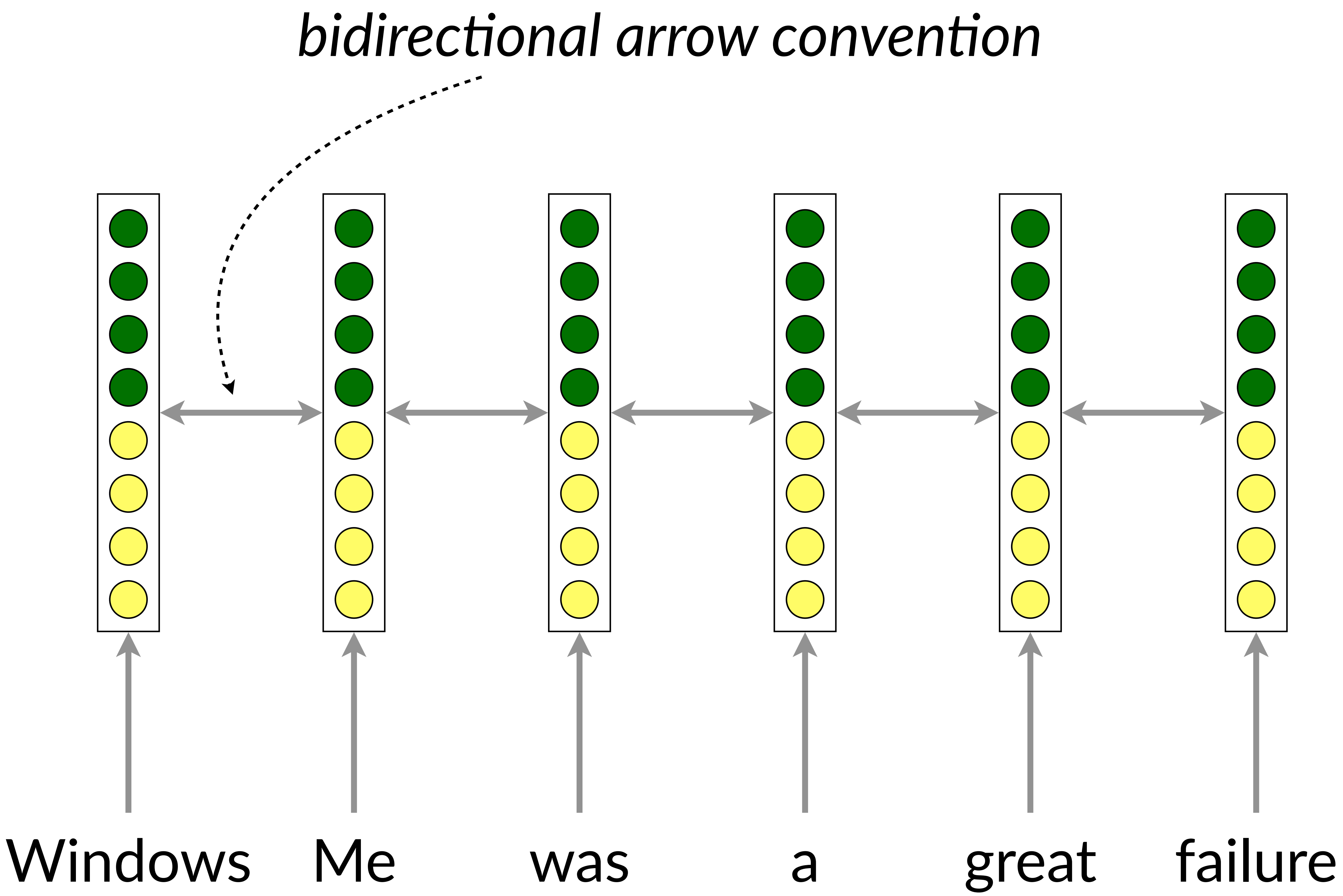
$$\overleftarrow{\mathbf{h}}^{[t]} = \mathrm{RNN_B}\left(\overleftarrow{\mathbf{h}}^{[t+1]}, \mathbf{u}^{[t]}\right)$$

$$\overrightarrow{\mathbf{h}}^{[t]} = \mathrm{RNN_F}\left(\overrightarrow{\mathbf{h}}^{[t-1]}, \mathbf{u}^{[t]}\right)$$

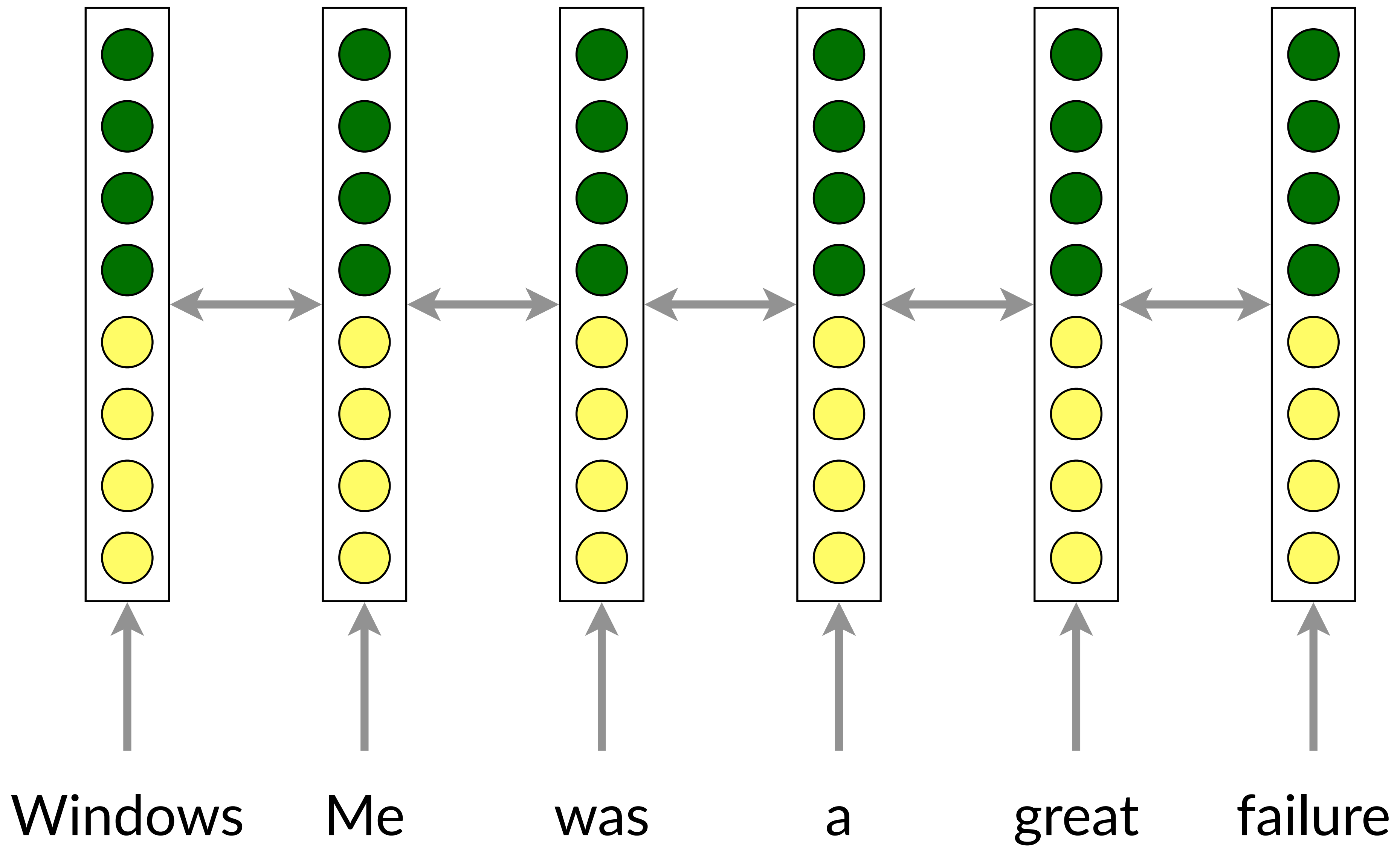
different weights

Bidirectional RNNs

Hidden state of the bidirectional RNN



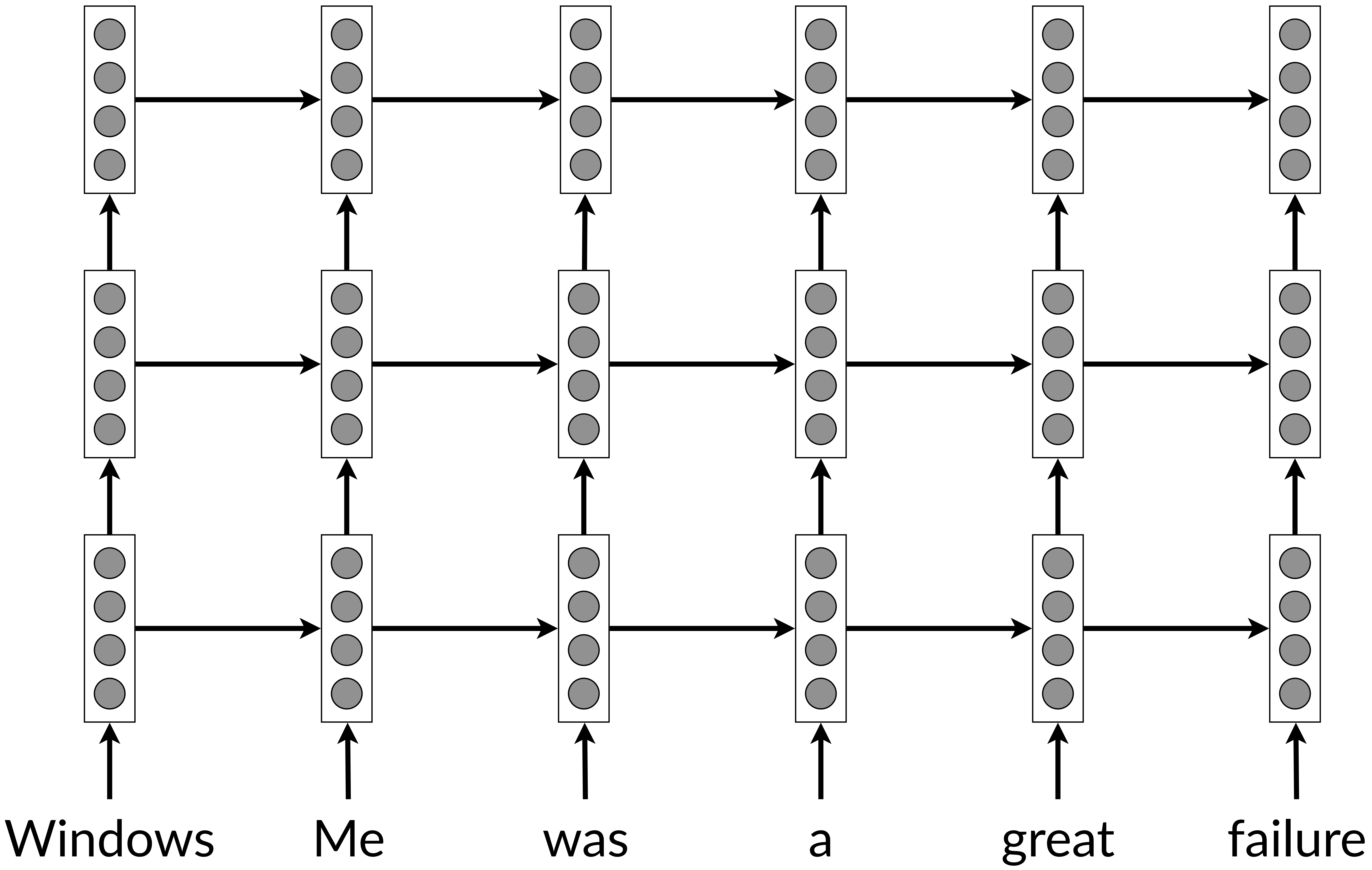
Bidirectional RNNs



- Bidirectional RNNs are very effective in sequence classification
- They requires access to the entire sequence, i.e. not necessarily great for language models (*text generators*)
- Bidirectional NNs are strong predictors, i.e. BERT: Bidirectional Encoder Representations from Transformers
  aclanthology.org/N19-1423.pdf

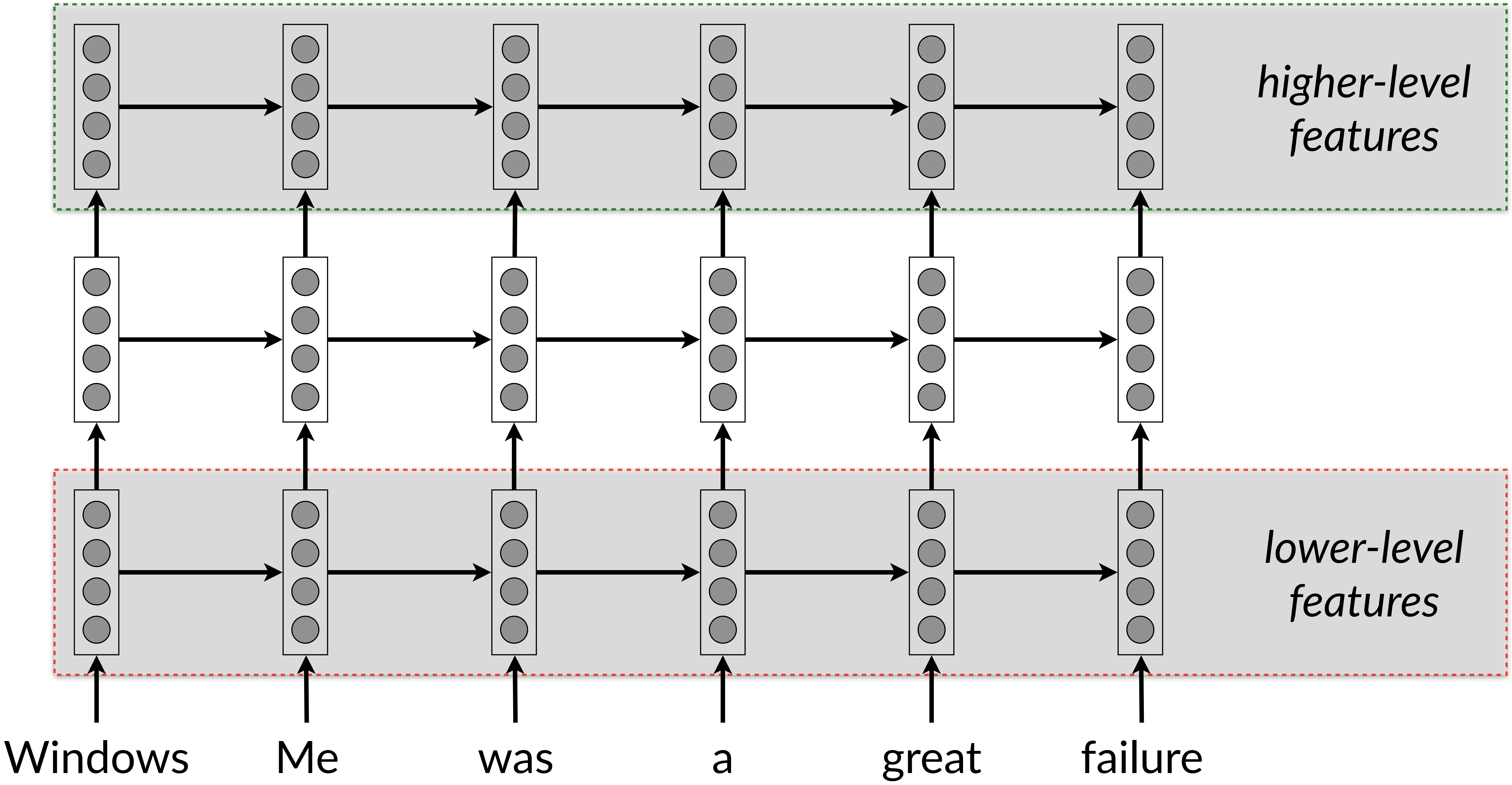
Stacked (*multi-layer*) RNNs

the output of one RNN layer (hidden state) becomes the input to the next



Stacked (*multi-layer*) RNNs

the output of one RNN layer (hidden state) becomes the input to the next



Next lecture *with me*

- Monday, March 18 (*last week*)
- Self-invited "guest" lecture on "*Modelling infectious disease prevalence using web search activity*"