Vanishing Gradients & Gated RNN Variants

Ling 282/482: Deep Learning for Computational Linguistics

C.M. Downey

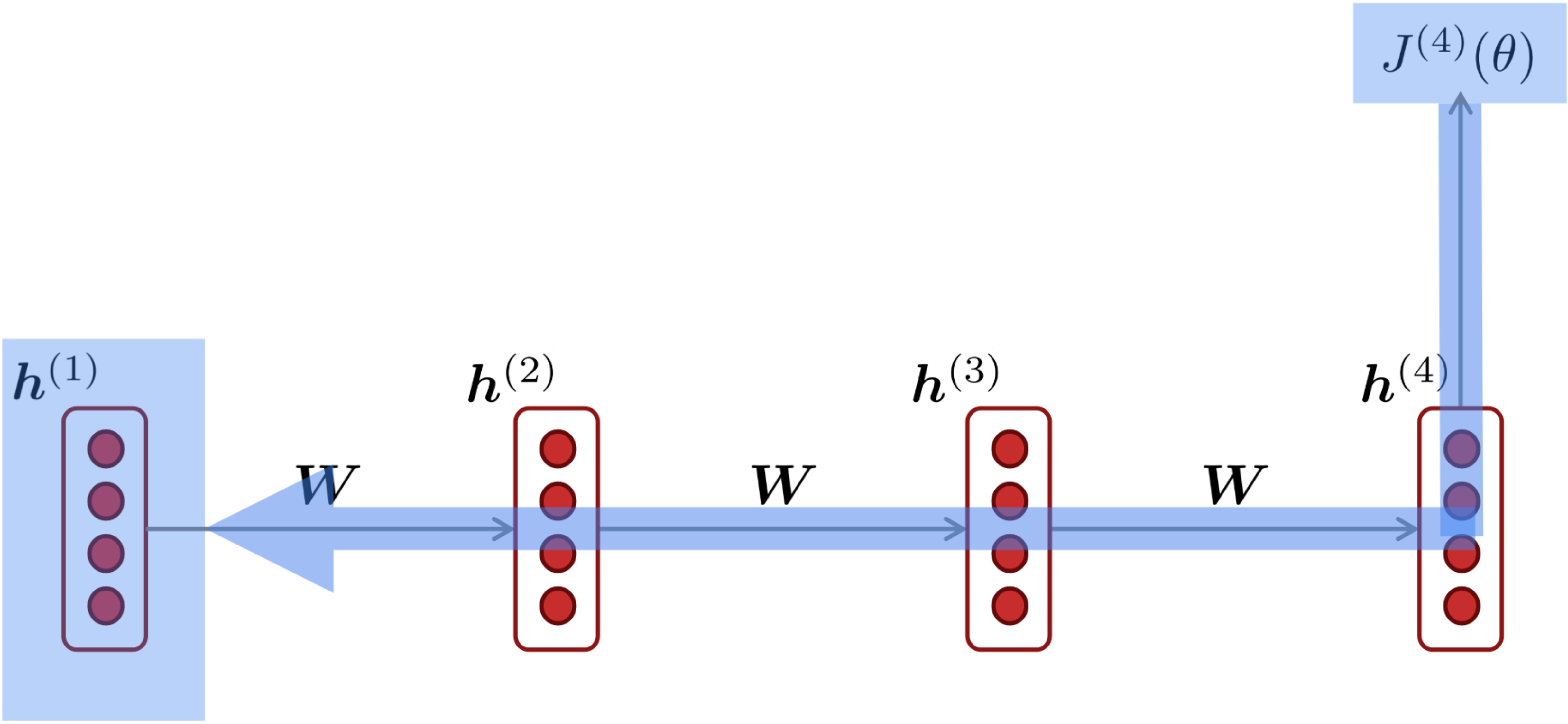
Fall 2025

Vanishing Gradients

Vanishing/Exploding Gradients Problem

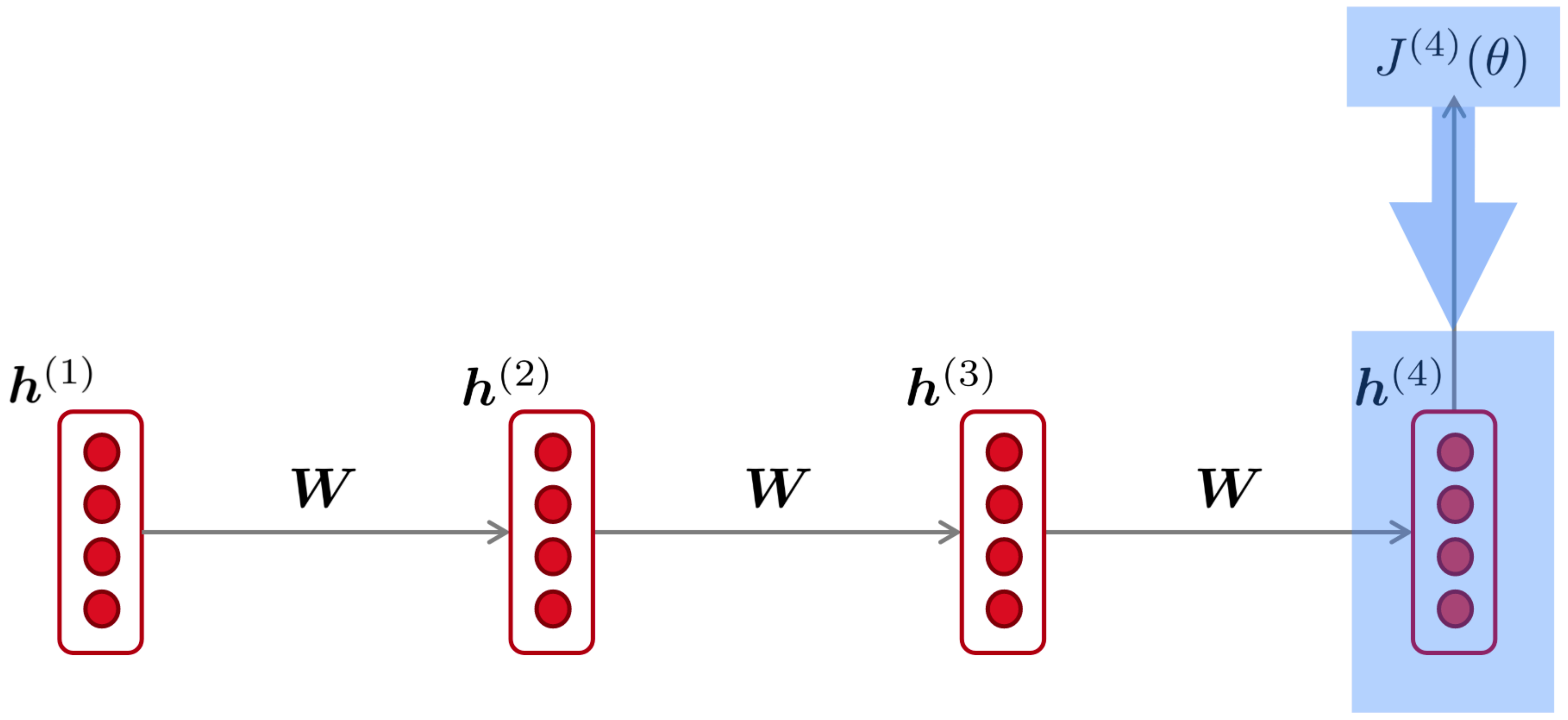
- Backpropagation with vanilla RNNs faces a major problem
- The gradients can **vanish** (**approach 0**) across time
- This makes it hard/impossible to learn **long distance dependencies**, which are rampant in natural language

Vanishing Gradients



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

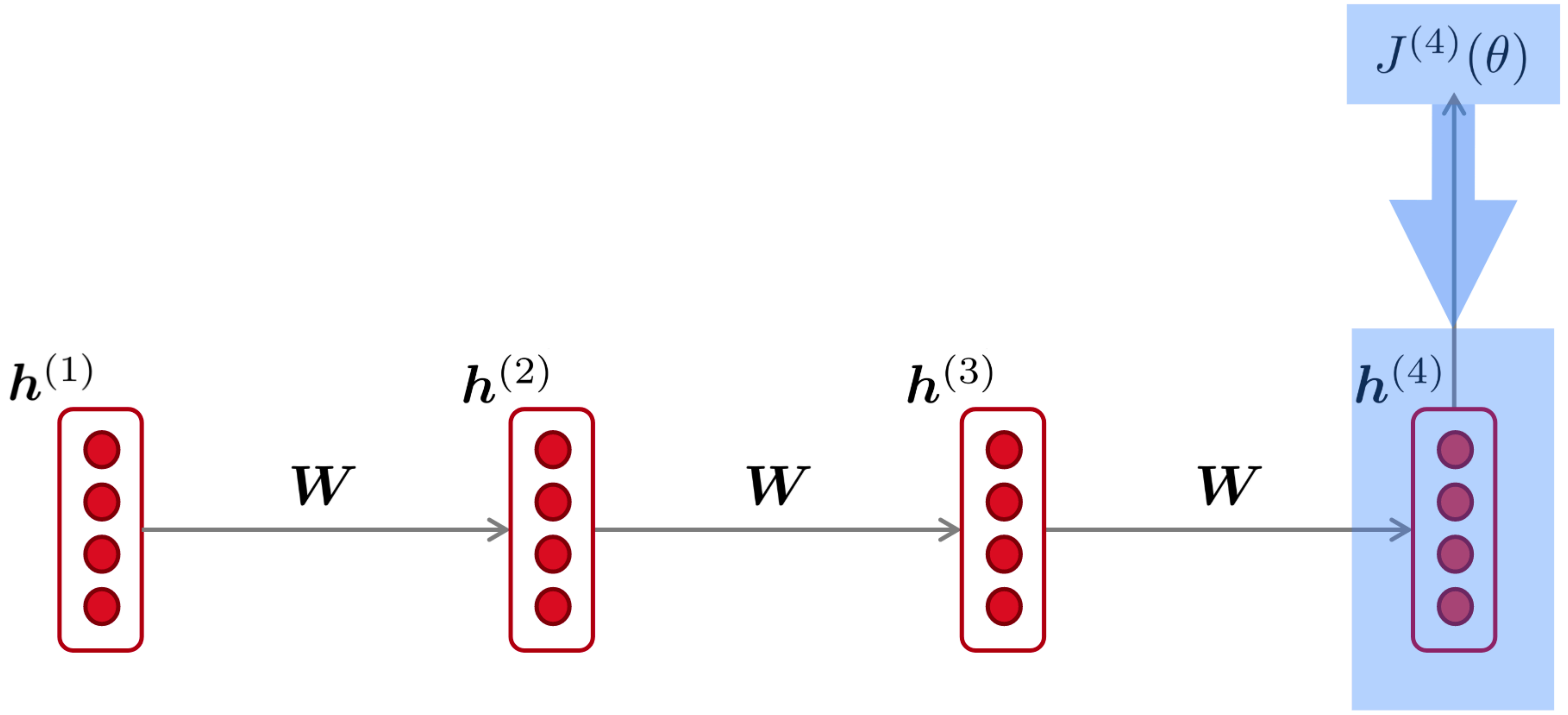
Vanishing Gradients



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

[source]

Vanishing Gradients

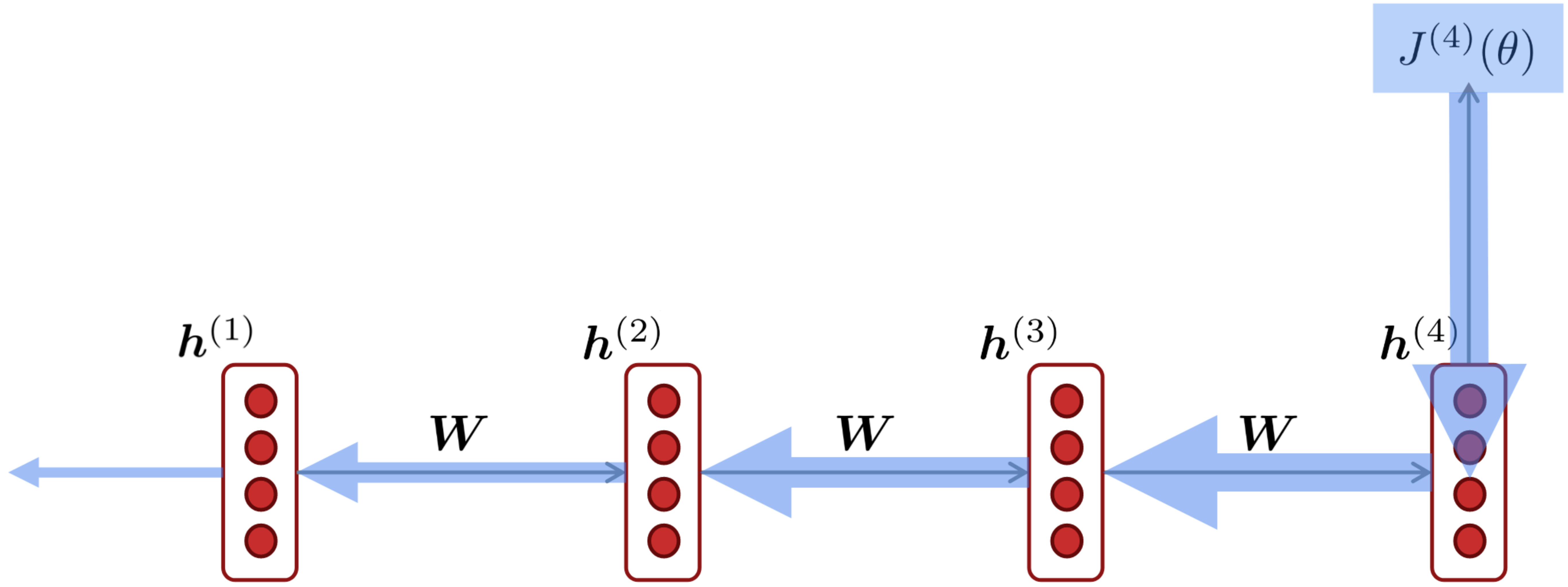


source

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

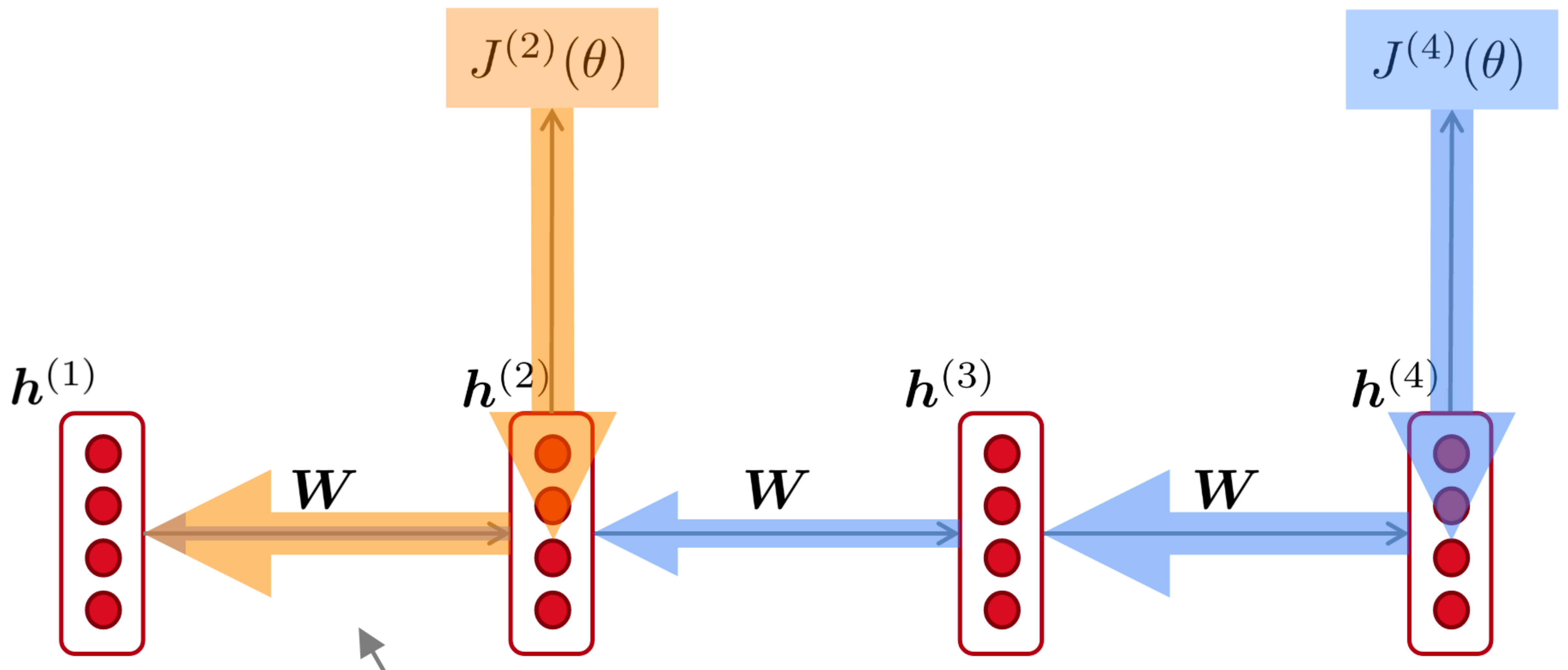
If these are small (depends on W), the effect from t=4 on t=1 will be very small

Vanishing Gradients



[source]

Vanishing Gradient Problem

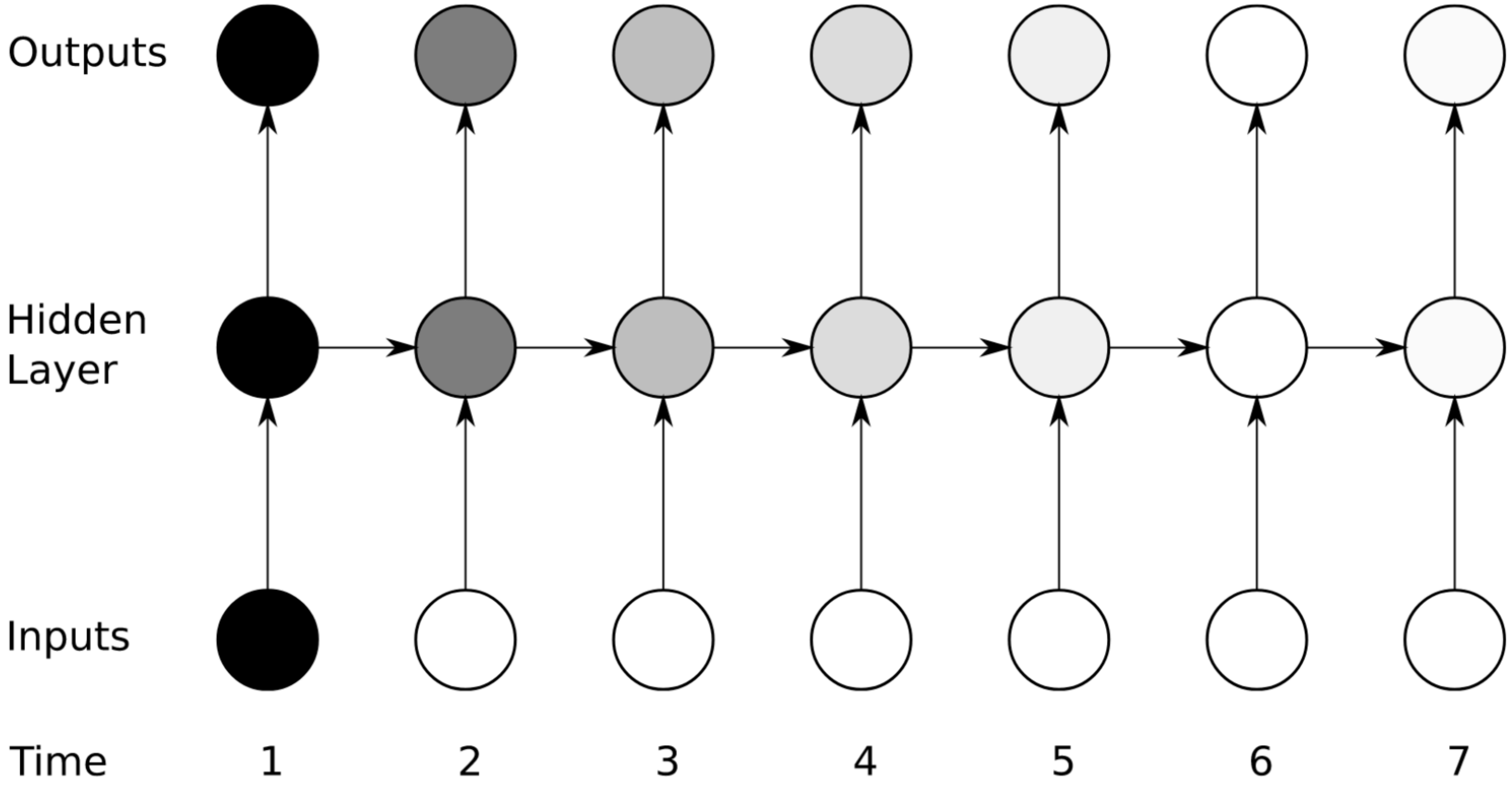


Gradient signal from faraway is lost because it's much smaller than gradient signal from close-by.

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

source

Vanishing Gradient Problem



Graves 2012

Examples of long-distance dependencies

- Gradient measures the **effect of the past on the future**
- If it vanishes between t and t+n, can't tell if:
    - There's no dependency in fact
    - The weights in our network just haven't yet captured the dependency
- Number agreement
    - The keys ____
    - The keys on the table ____
    - The keys next to the book on top of the table ____
- Selectional Preferences
    - The **family** moved from the city because they wanted a larger **house**.
    - The **team** moved from the city because they wanted a larger **market**.

Gating Based RNNs: LSTM and GRU

LSTMs

- Long Short-Term Memory (Hochreiter and Schmidhuber 1997)
- The gold standard / **default RNN**
  - If someone says "RNN" now, **they almost always mean "LSTM"**
- Originally: to solve the vanishing/exploding gradient problem for RNNs
  - Vanilla: **re-writes** the entire hidden state at every time-step
  - LSTM: **separate** hidden state and memory
    - Read, write to/from memory; can preserve **long-term information**

LSTMs

$$f_t = \sigma\left(W^f \cdot h_{t-1}x_t + b^f\right)$$

$$i_t = \sigma\left(W^i \cdot h_{t-1}x_t + b^i\right)$$

$$\hat{c}_t = \tanh\left(W^c \cdot h_{t-1}x_t + b^c\right)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \hat{c}_t$$

$$o_t = \sigma\left(W^o \cdot h_{t-1}x_t + b^o\right)$$

$$h_t = o_t \odot \tanh\left(c_t\right)$$

LSTMs

🤔

$$f_t = \sigma\left(W^f \cdot h_{t-1}x_t + b^f\right)$$

$$i_t = \sigma\left(W^i \cdot h_{t-1}x_t + b^i\right)$$

$$\hat{c}_t = \tanh\left(W^c \cdot h_{t-1}x_t + b^c\right)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \hat{c}_t$$

$$o_t = \sigma\left(W^o \cdot h_{t-1}x_t + b^o\right)$$

$$h_t = o_t \odot \tanh\left(c_t\right)$$

LSTMs

🤔

- Key innovation:
  - $c_t, h_t = f(x_t, c_{t-1}, h_{t-1})$
  - c_t: a **memory cell**
- Reading/writing controlled by **gates**
  - f_t: forget gate
  - i_t: input gate
  - o_t: output gate

$$f_t = \sigma\left(W^f \cdot h_{t-1}x_t + b^f\right)$$

$$i_t = \sigma\left(W^i \cdot h_{t-1}x_t + b^i\right)$$

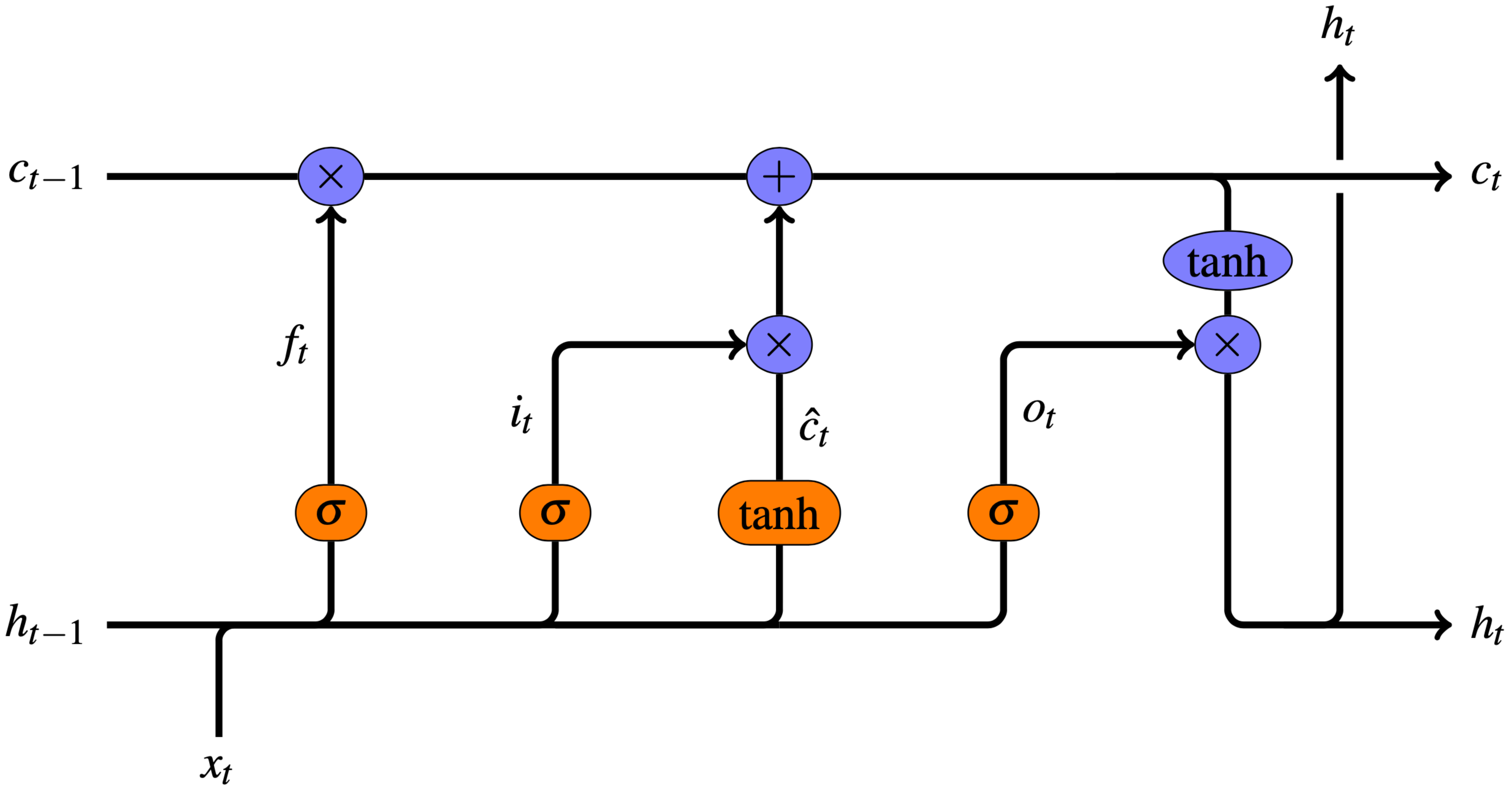
$$\hat{c}_t = \tanh\left(W^c \cdot h_{t-1}x_t + b^c\right)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \hat{c}_t$$

$$o_t = \sigma\left(W^o \cdot h_{t-1}x_t + b^o\right)$$

$$h_t = o_t \odot \tanh\left(c_t\right)$$

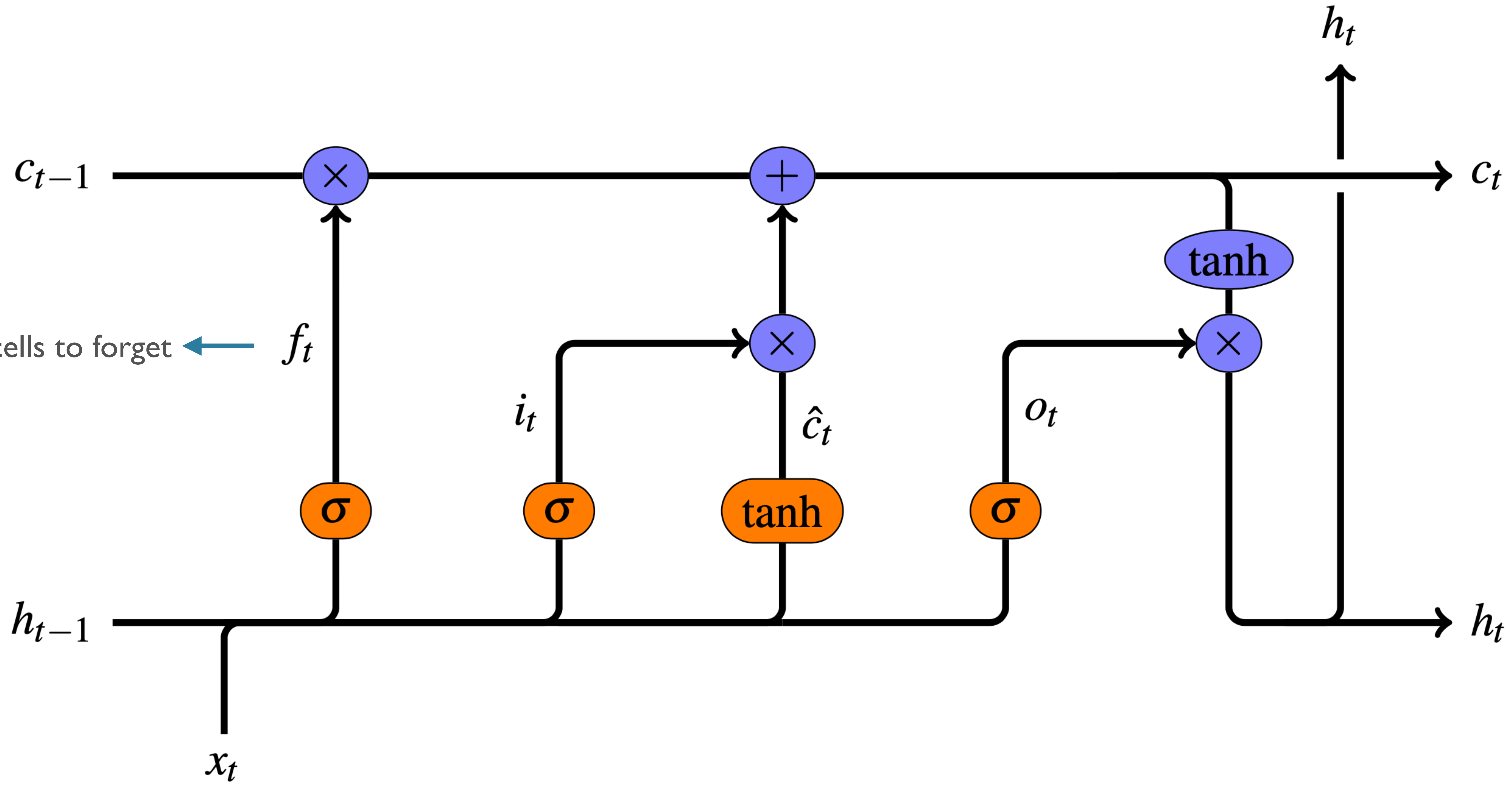
LSTMs



Steinert-Threlkeld and Szymanik 2019; Olah 2015

LSTMs

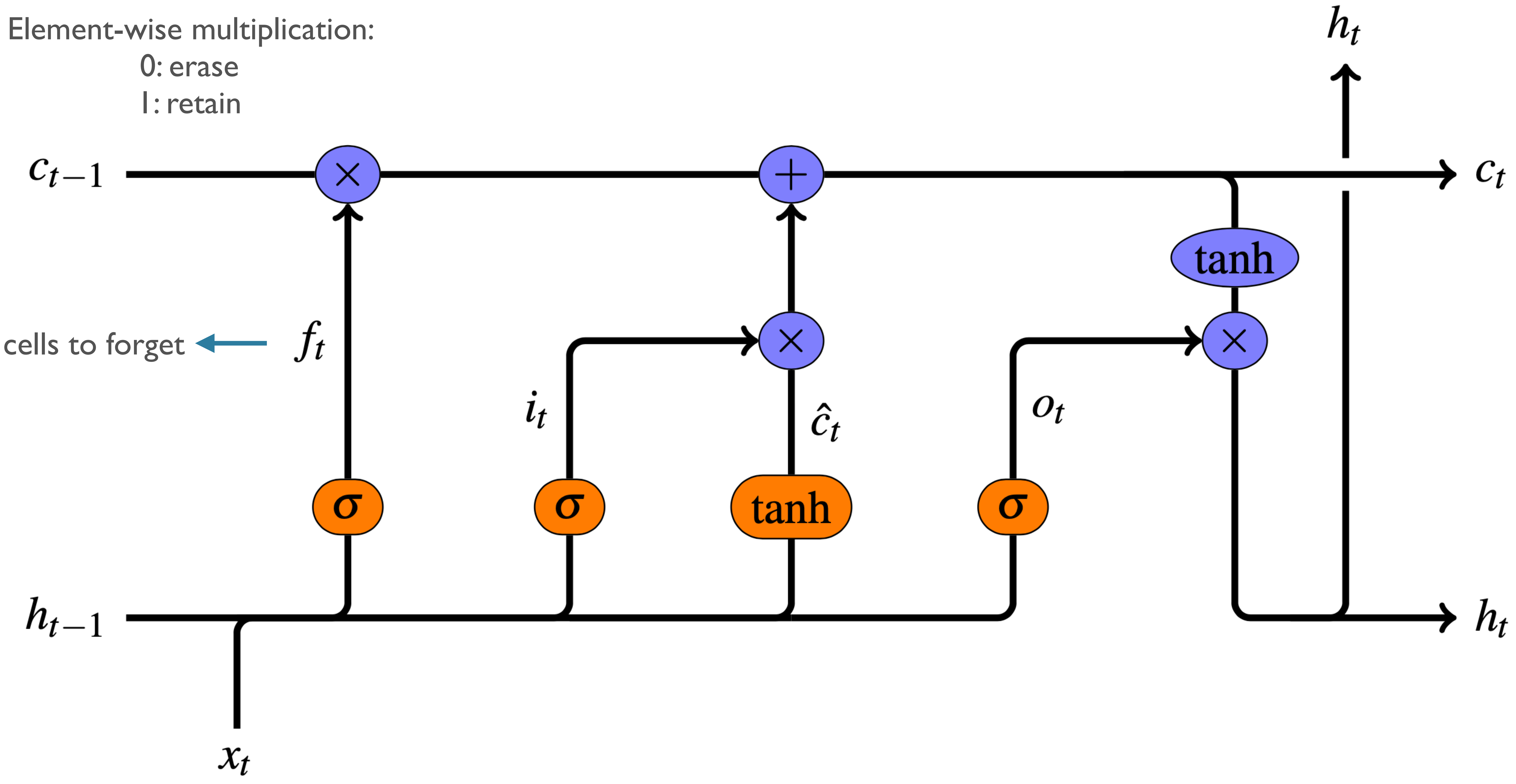
$f_t \in [0,1]^m$: which cells to forget



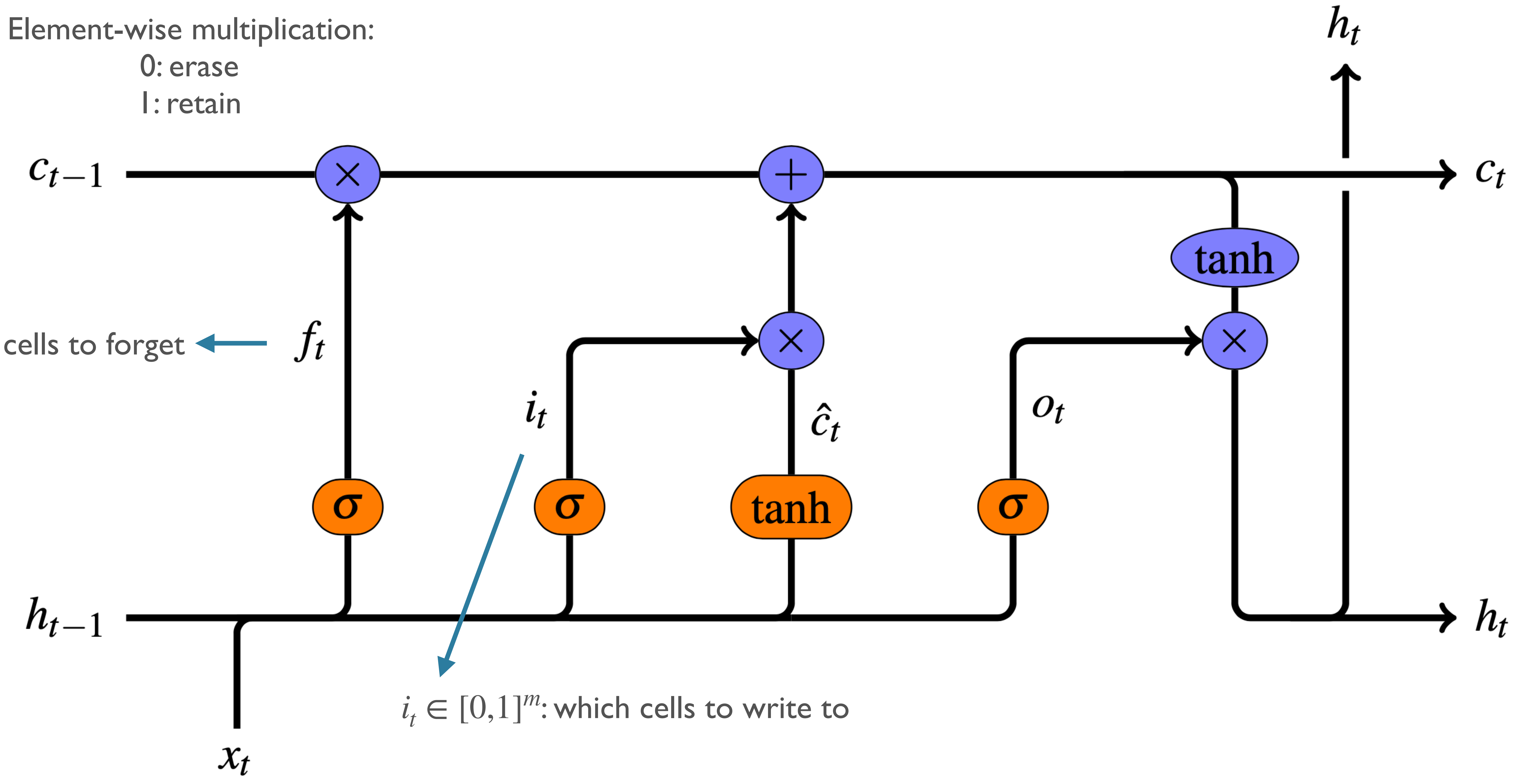
LSTMs

Element-wise multiplication:
0: erase
1: retain

$f_t \in [0,1]^m$: which cells to forget



LSTMs

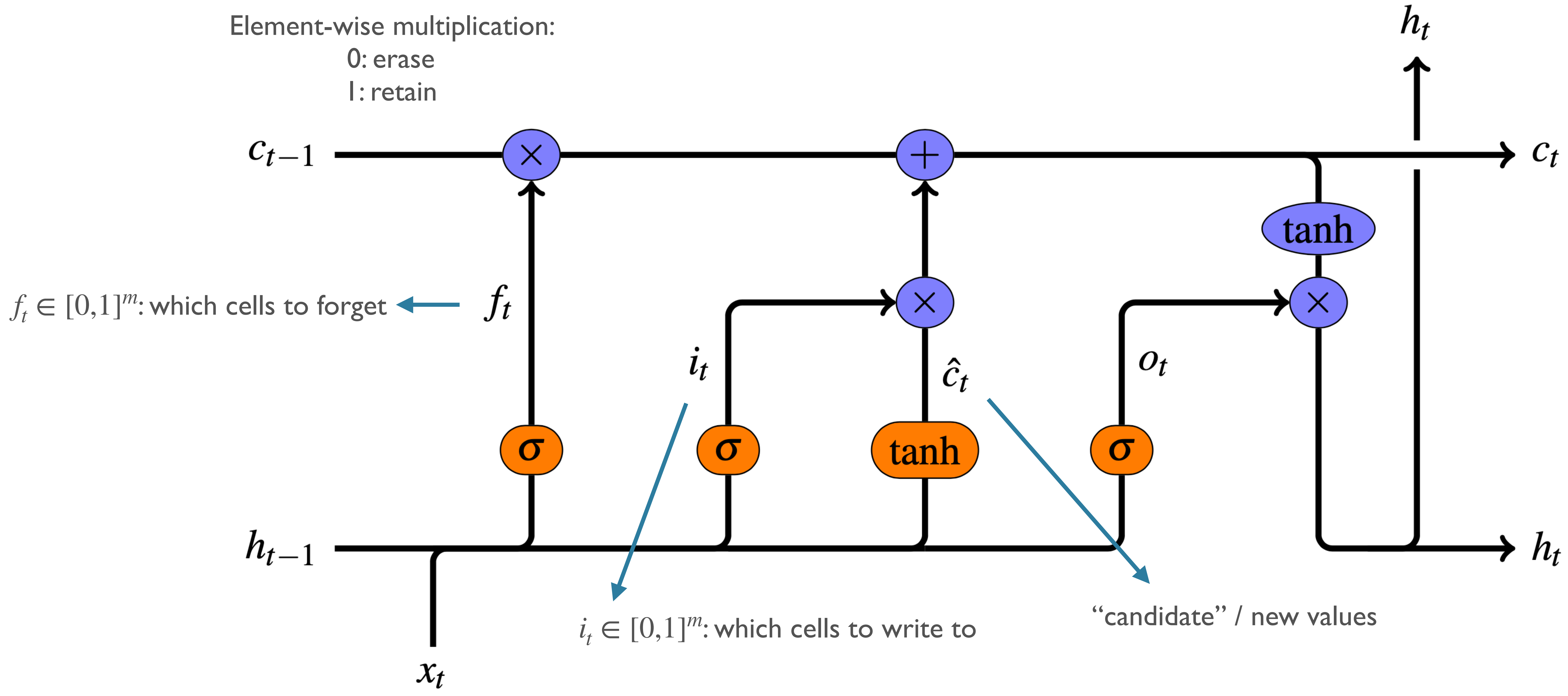


Element-wise multiplication:
0: erase
1: retain

$f_t \in [0,1]^m$: which cells to forget

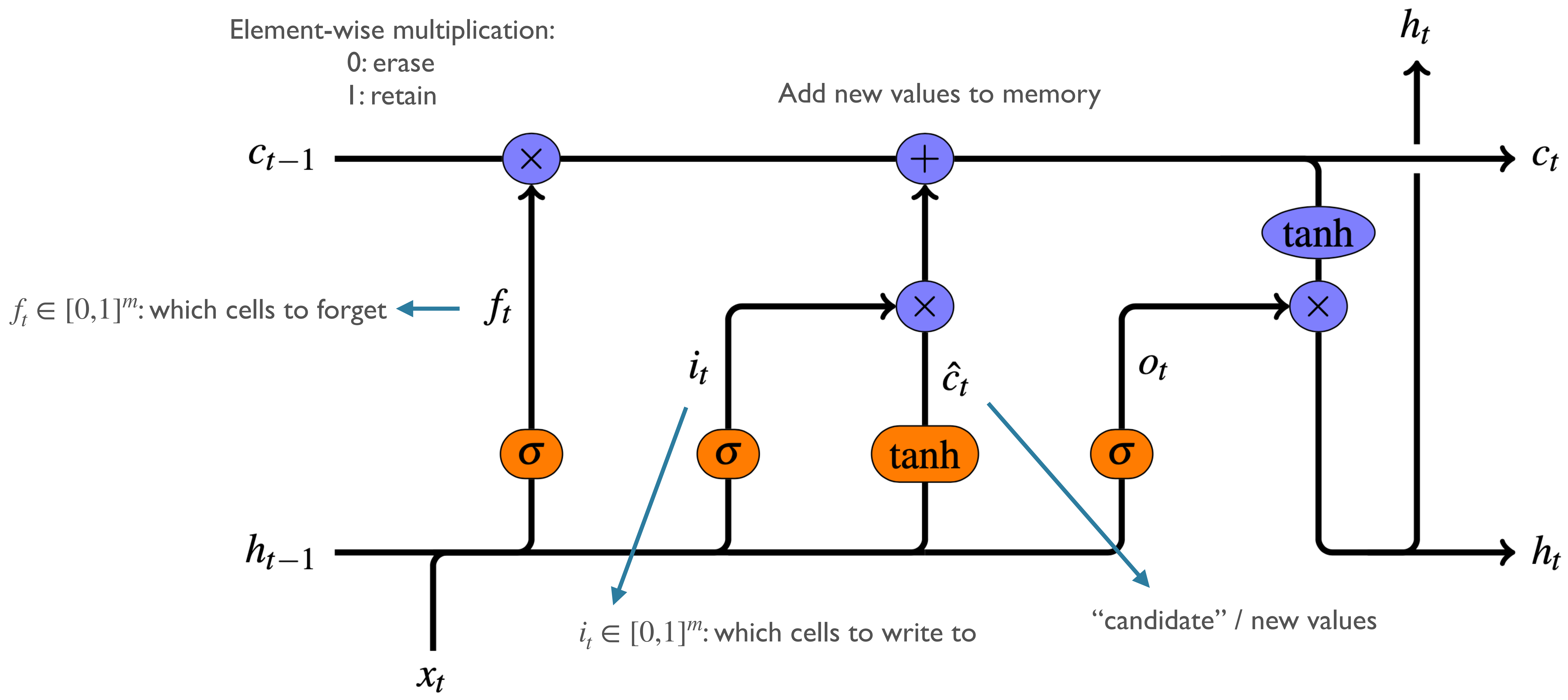
$i_t \in [0,1]^m$: which cells to write to

LSTMs



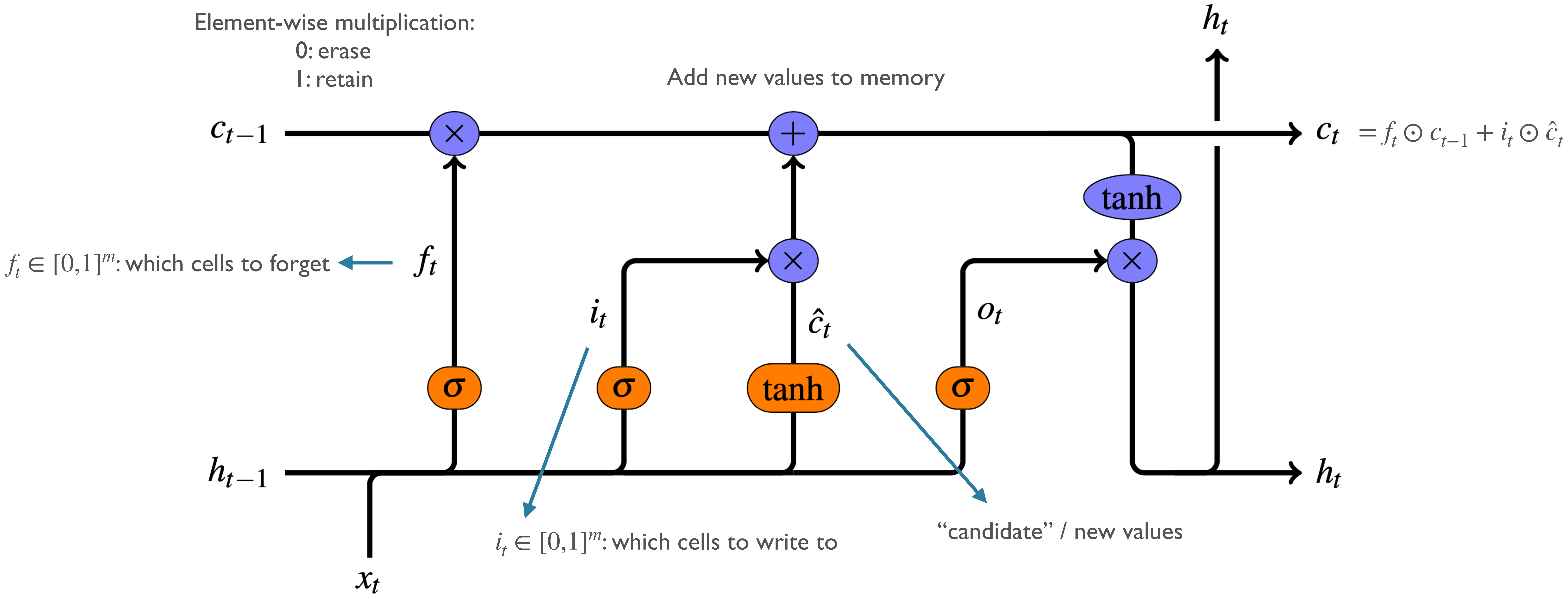
Element-wise multiplication:
0: erase
1: retain
h_t
c_{t-1}
c_t
tanh
$f_t \in [0,1]^m$: which cells to forget
f_t
i_t
$\hat{c}_t$
o_t
σ
σ
tanh
σ
h_{t-1}
h_t
x_t
$i_t \in [0,1]^m$: which cells to write to
"candidate" / new values

LSTMs



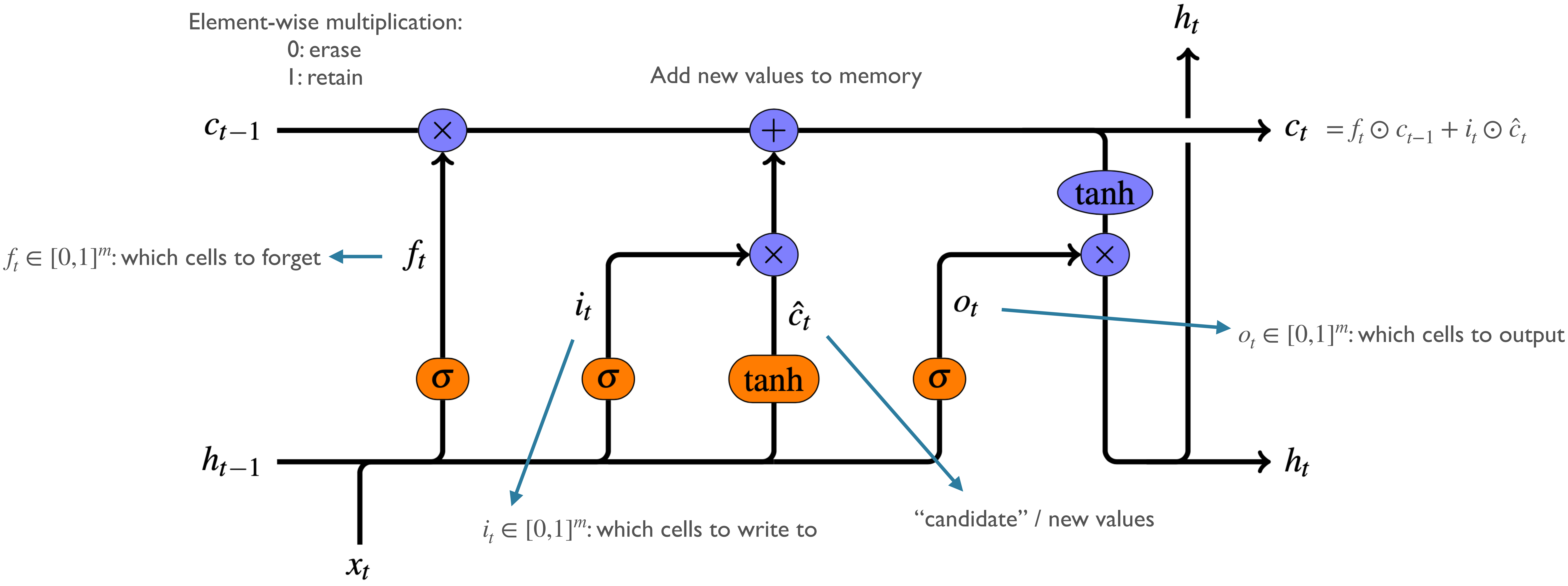
Element-wise multiplication:
0: erase
1: retain
Add new values to memory
h_t
c_{t-1}
c_t
tanh
$f_t \in [0,1]^m$: which cells to forget
f_t
i_t
$\hat{c}_t$
o_t
σ
σ
tanh
σ
h_{t-1}
h_t
$i_t \in [0,1]^m$: which cells to write to
"candidate" / new values
x_t

LSTMs



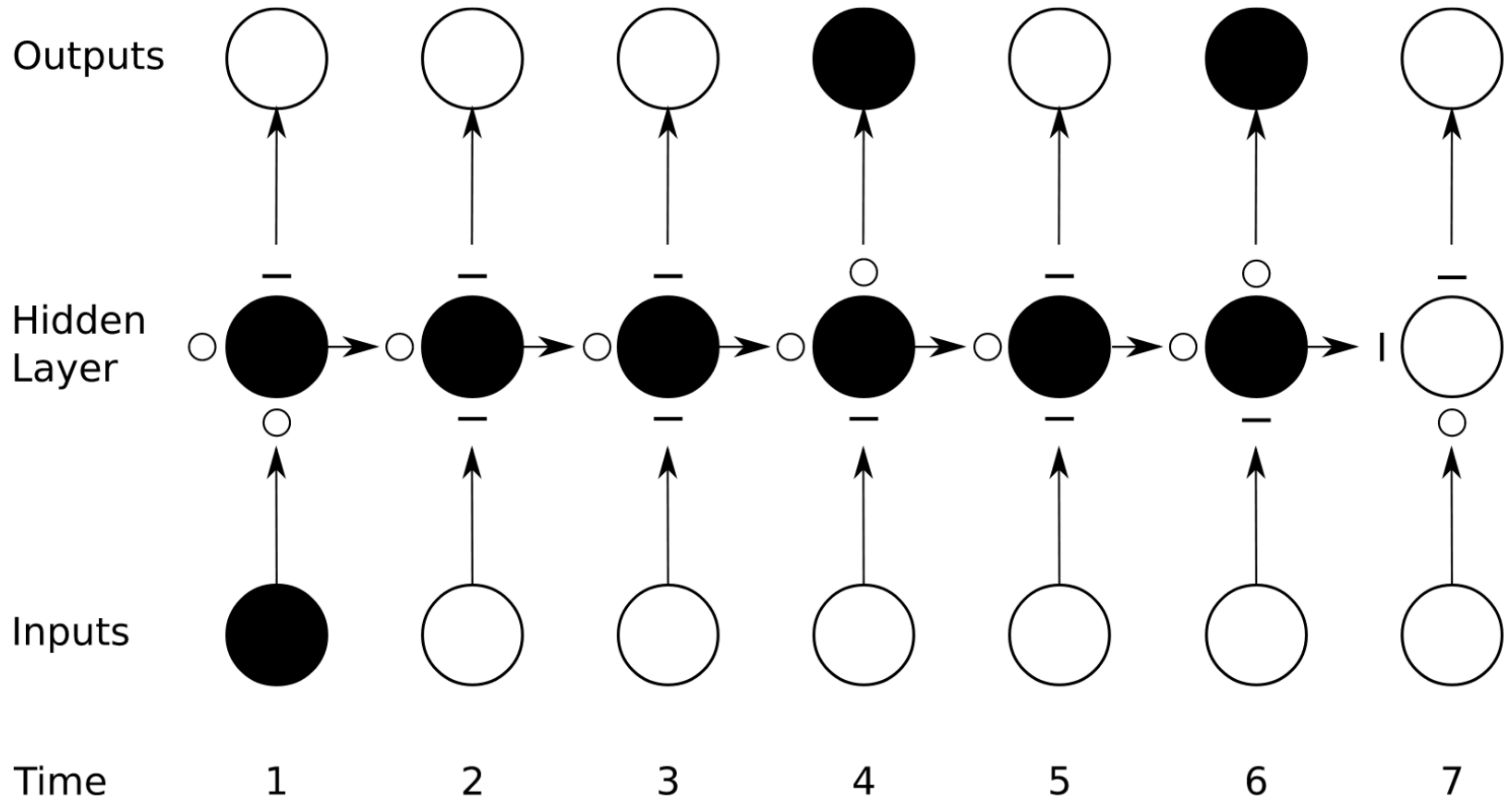
Element-wise multiplication:
0: erase
1: retain
Add new values to memory
h_t
c_{t-1}
c_t $= f_t \odot c_{t-1} + i_t \odot \hat{c}_t$
tanh
$f_t \in [0,1]^m$: which cells to forget
f_t
i_t
$\hat{c}_t$
o_t
σ
σ
tanh
σ
h_{t-1}
h_t
$i_t \in [0,1]^m$: which cells to write to
"candidate" / new values
x_t

LSTMs



Element-wise multiplication:
0: erase
1: retain
Add new values to memory
h_t
c_{t-1}
c_t $= f_t \odot c_{t-1} + i_t \odot \hat{c}_t$
tanh
$f_t \in [0,1]^m$: which cells to forget
f_t
i_t
$\hat{c}_t$
o_t
$o_t \in [0,1]^m$: which cells to output
σ
σ
tanh
σ
h_{t-1}
h_t
x_t
$i_t \in [0,1]^m$: which cells to write to
"candidate" / new values

LSTMs solve vanishing gradients



Graves 2012

The Emergence of Number and Syntax Units in LSTM Language Models

Yair Lakretz
Cognitive Neuroimaging Unit
NeuroSpin center
91191, Gif-sur-Yvette, France
`yair.lakretz@gmail.com`

German Kruszewski
Facebook AI Research
Paris, France
`germank@gmail.com`

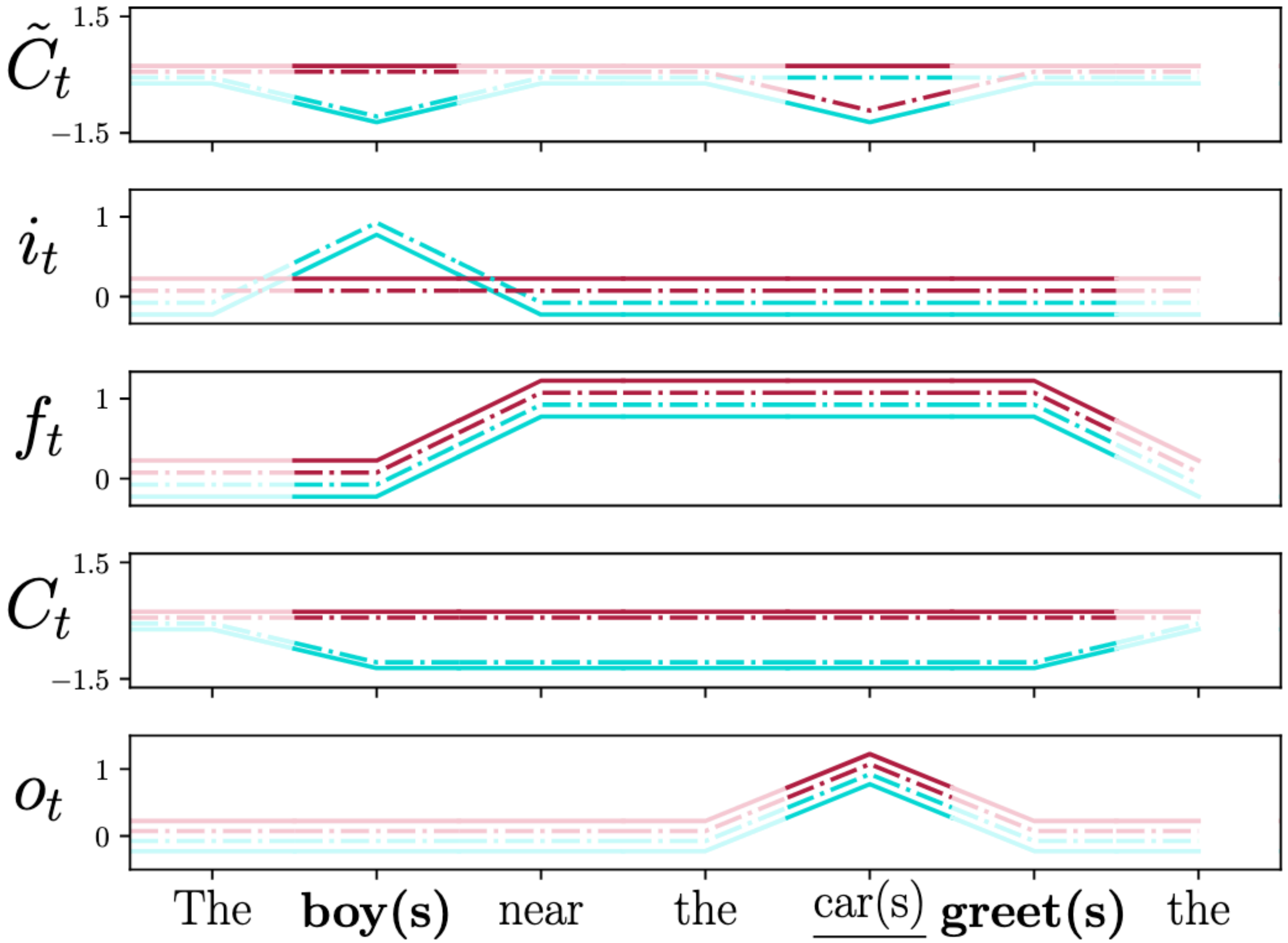
Theo Desbordes
Facebook AI Research
Paris, France
`tdesbordes@fb.com`

Dieuwke Hupkes
ILLC, University of Amsterdam
Amsterdam, Netherlands
`d.hupkes@uva.nl`

Stanislas Dehaene
Cognitive Neuroimaging Unit
NeuroSpin center
91191, Gif-sur-Yvette, France
`stanislas.dehaene@gmail.com`

Marco Baroni
Facebook AI Research
Paris, France
`mbaroni@fb.com`

Cell dynamics for storing number info



“The BiLSTM Hegemony”

- Chris Manning, in 2017:

To a first approximation,
the de facto consensus in NLP in 2017 is
that no matter what the task,
you throw a BiLSTM at it, with
attention if you need information flow

source

Gated Recurrent Unit (GRU)

- Cho et al 2014: gated like LSTM, but **no separate memory cell**
  - "Collapses" execution/control and memory
- Fewer gates = **fewer parameters**, higher speed
  - Update gate
  - Reset gate

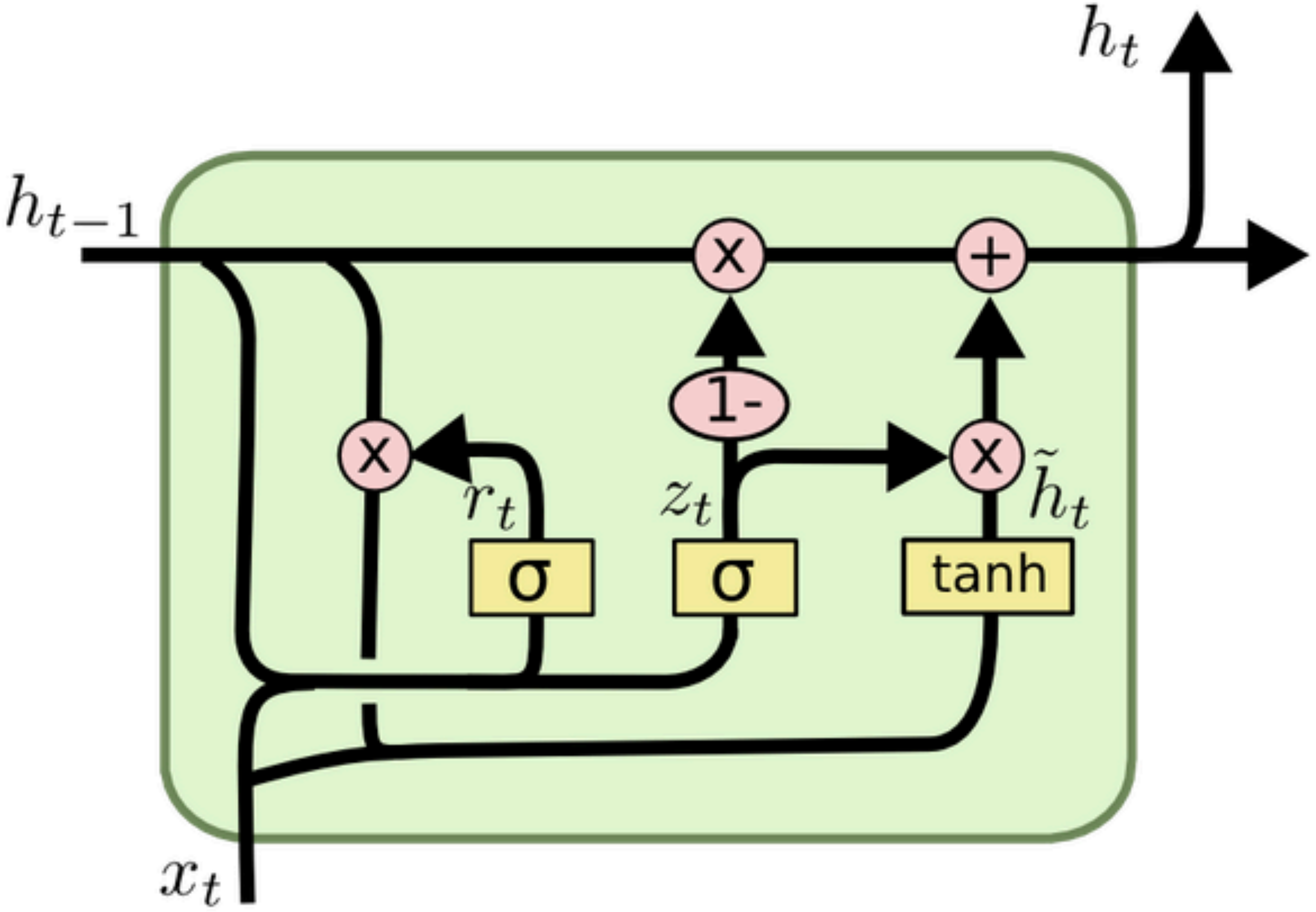
$$u_t = \sigma(W_u h_{t-1} + U_u x_t + b_u)$$

$$r_t = \sigma(W_r h_{t-1} + U_r x_t + b_r)$$

$$\tilde{h}_t = \tanh(W_h(r_t \odot h_{t-1}) + U_h x_t + b_h)$$

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

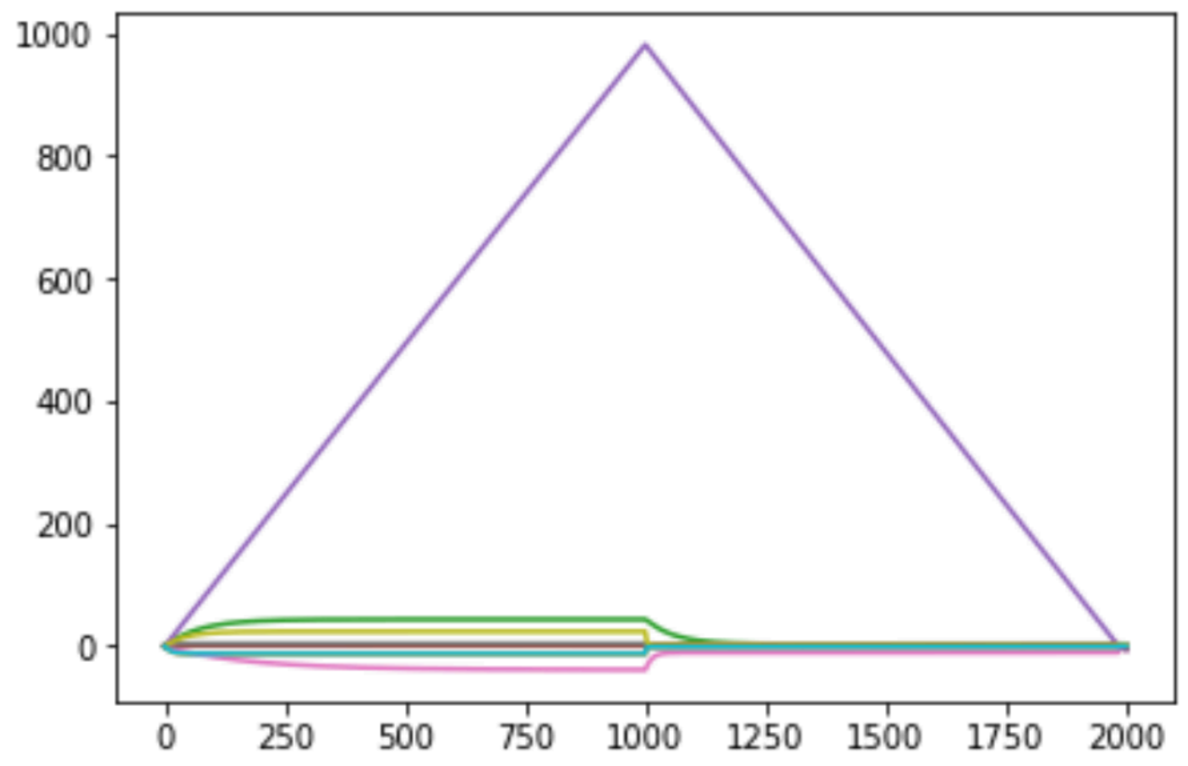
Gated Recurrent Unit



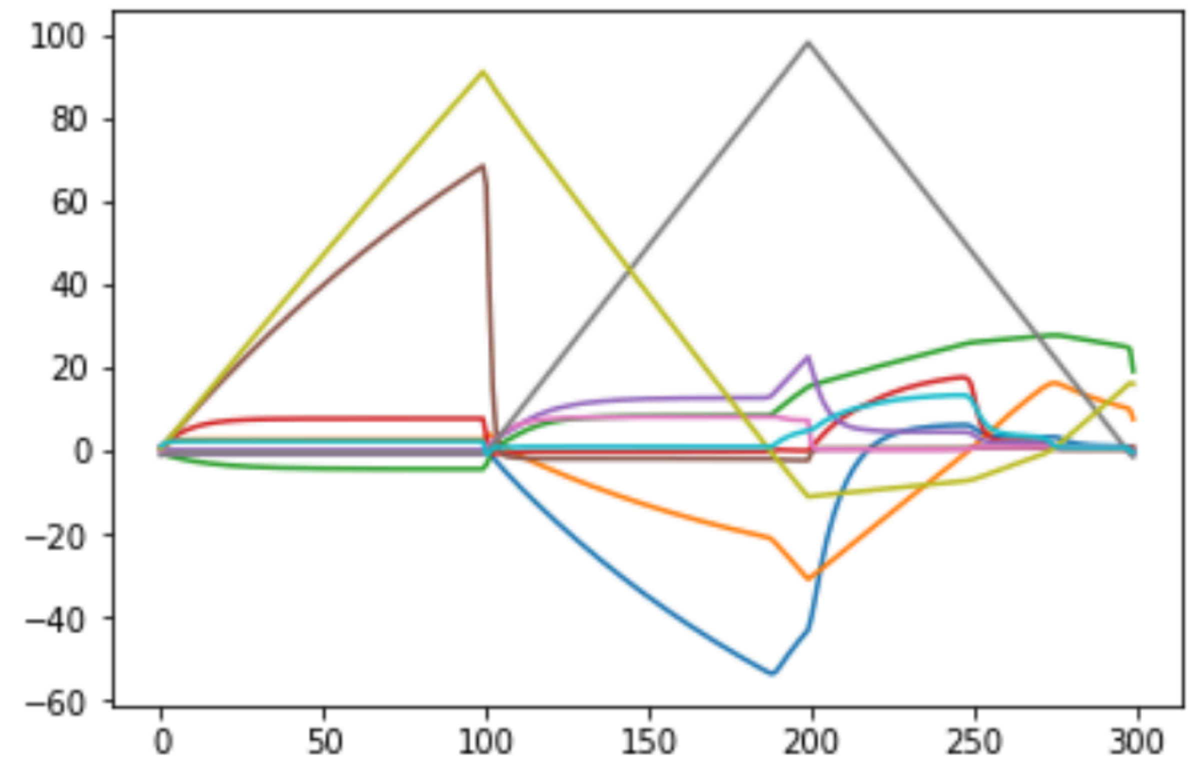
source

LSTM vs GRU

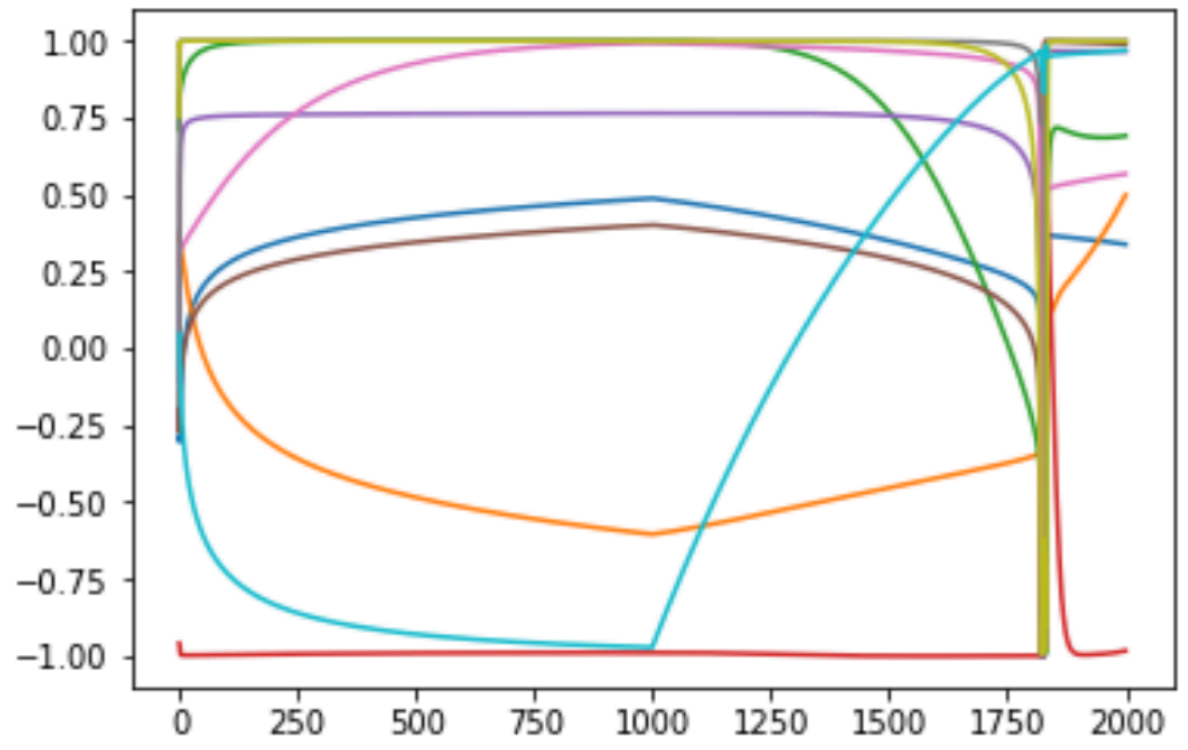
- Generally: **LSTM is a good default choice**
  - GRU can be used if **speed** and **fewer parameters** are important
- Full differences between them **not fully understood**
- Performance often comparable, but: LSTMs can store **unboundedly large values in memory**, and seem to e.g. count better



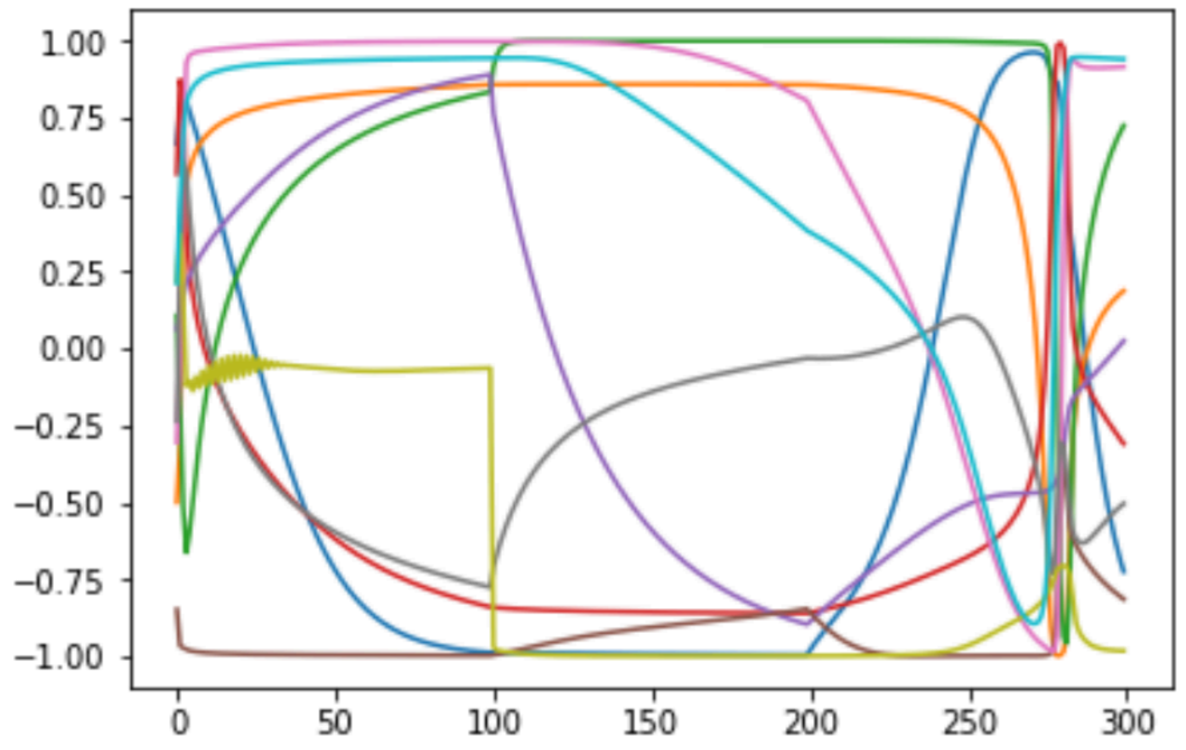
(a) $a^n b^n$-LSTM on $a^{1000} b^{1000}$



(b) $a^n b^n c^n$-LSTM on $a^{100} b^{100} c^{100}$



(c) $a^n b^n$-GRU on $a^{1000} b^{1000}$



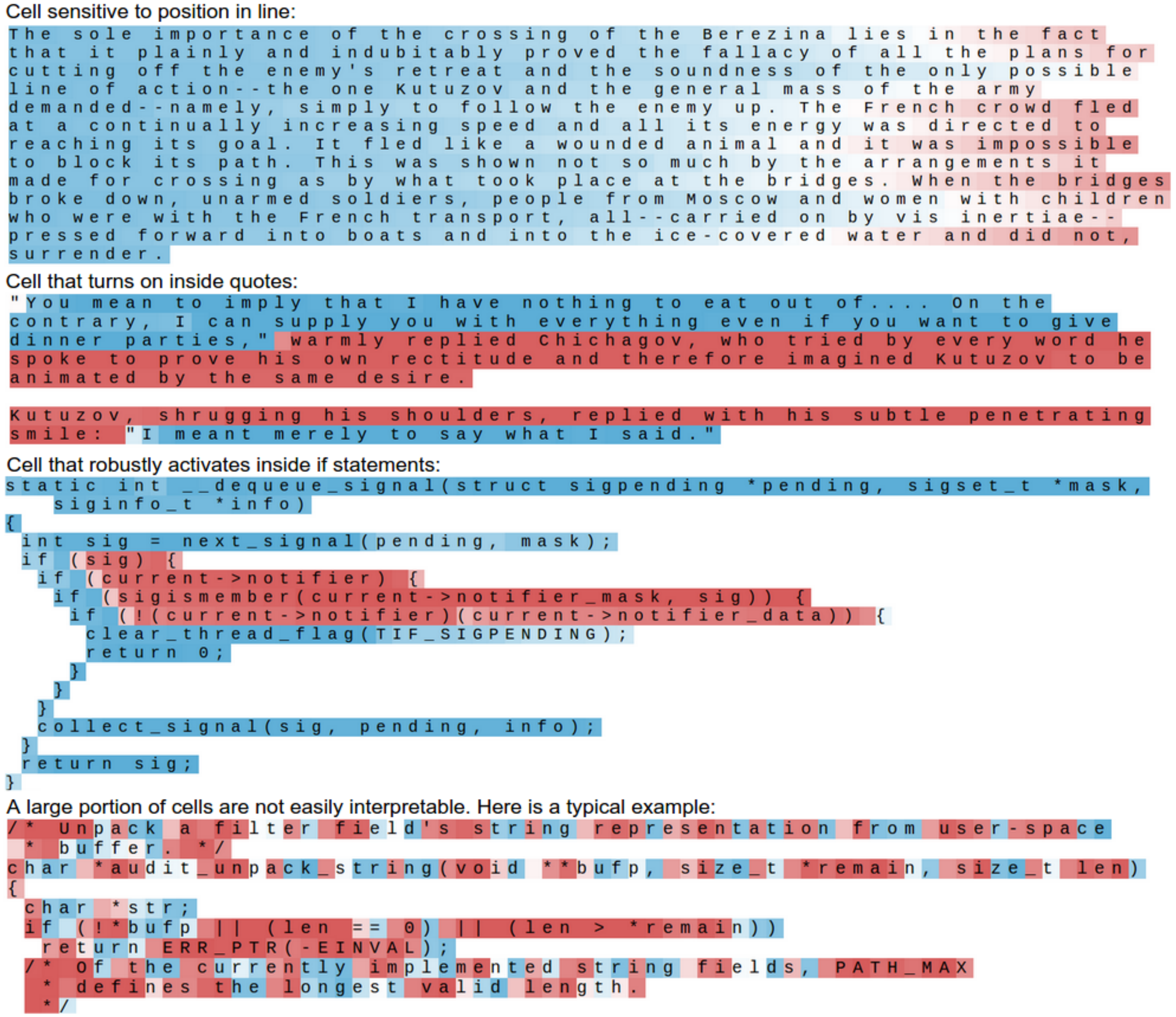
(d) $a^n b^n c^n$-GRU on $a^{100} b^{100} c^{100}$

source

Odds and Ends

Cell Interpretation

- “The Unreasonable Effectiveness of RNNs” (Karpathy 2015): http://karpathy.github.io/2015/05/21/rnn-effectiveness/



ELMo (**E**mbeddings from **L**anguage **M**odels)

Peters et al NAACL 2018

ELMo (**E**mbeddings from **L**anguage **M**odels)

Peters et al NAACL 2018

ELMo

Deep contextualized word representations

Matthew E. Peters[†]**, Mark Neumann**[†]**, Mohit Iyyer**[†]**, Matt Gardner**[†]**,**
{matthewp,markn,mohiti,mattg}@allenai.org

Christopher Clark[*]**, Kenton Lee**[*]**, Luke Zettlemoyer**[†*]
{csquared,kentonl,lsz}@cs.washington.edu

[†]Allen Institute for Artificial Intelligence
[*]Paul G. Allen School of Computer Science & Engineering, University of Washington

Abstract

We introduce a new type of *deep contextualized* word representation that models both (1) complex characteristics of word use (e.g., syntax and semantics), and (2) how these uses vary across linguistic contexts (i.e., to model polysemy). Our word vectors are learned functions of the internal states of a deep bidirectional language model (biLM), which is pre-trained on a large text corpus. We show that these representations can be easily added to existing models and significantly improve the state of the art across six challenging NLP problems, including question answering, textual entailment and sentiment analysis. We also present an analysis showing that exposing the deep internals of the pre-trained network is crucial, allowing downstream models to mix different types of semi-supervision signals.

guage model (LM) objective on a large text corpus. For this reason, we call them ELMo (Embeddings from Language Models) representations. Unlike previous approaches for learning contextualized word vectors (Peters et al., 2017; McCann et al., 2017), ELMo representations are deep, in the sense that they are a function of all of the internal layers of the biLM. More specifically, we learn a linear combination of the vectors stacked above each input word for each end task, which markedly improves performance over just using the top LSTM layer.

Combining the internal states in this manner allows for very rich word representations. Using intrinsic evaluations, we show that the higher-level LSTM states capture context-dependent aspects of word meaning (e.g., they can be used without modification to perform well on supervised

ELMo

Deep contextualized word representations

Matthew E. Peters[†]**, Mark Neumann**[†]**, Mohit Iyyer**[†]**, Matt Gardner**[†]**,**
{matthewp,markn,mohiti,mattg}@allenai.org

Christopher Clark[*]**, Kenton Lee**[*]**, Luke Zettlemoyer**[†*]
{csquared,kentonl,lsz}@cs.washington.edu

[†]Allen Institute for Artificial Intelligence
[*]Paul G. Allen School of Computer Science & Engineering, University of Washington

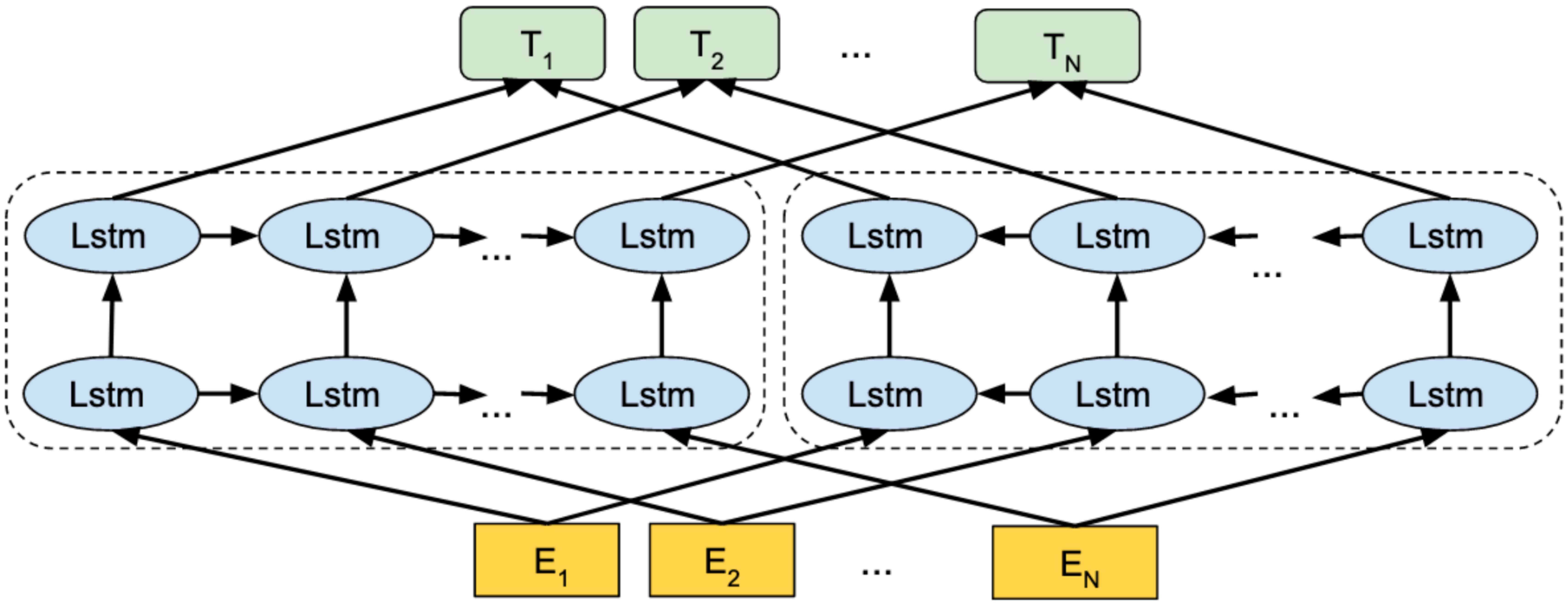
Abstract

We introduce a new type of *deep contextualized* word representation that models both (1) complex characteristics of word use (e.g., syntax and semantics), and (2) how these uses vary across linguistic contexts (i.e., to model polysemy). Our word vectors are learned functions of the internal states of a deep bidirectional language model (biLM), which is pre-trained on a large text corpus. We show that these representations can be easily added to existing models and significantly improve the state of the art across six challenging NLP problems, including question answering, textual entailment and sentiment analysis. We also present an analysis showing that exposing the deep internals of the pre-trained network is crucial, allowing downstream models to mix different types of semi-supervision signals.

guage model (LM) objective on a large text corpus. For this reason, we call them ELMo (Embeddings from Language Models) representations. Unlike previous approaches for learning contextualized word vectors (Peters et al., 2017; McCann et al., 2017), ELMo representations are deep, in the sense that they are a function of all of the internal layers of the biLM. More specifically, we learn a linear combination of the vectors stacked above each input word for each end task, which markedly improves performance over just using the top LSTM layer.

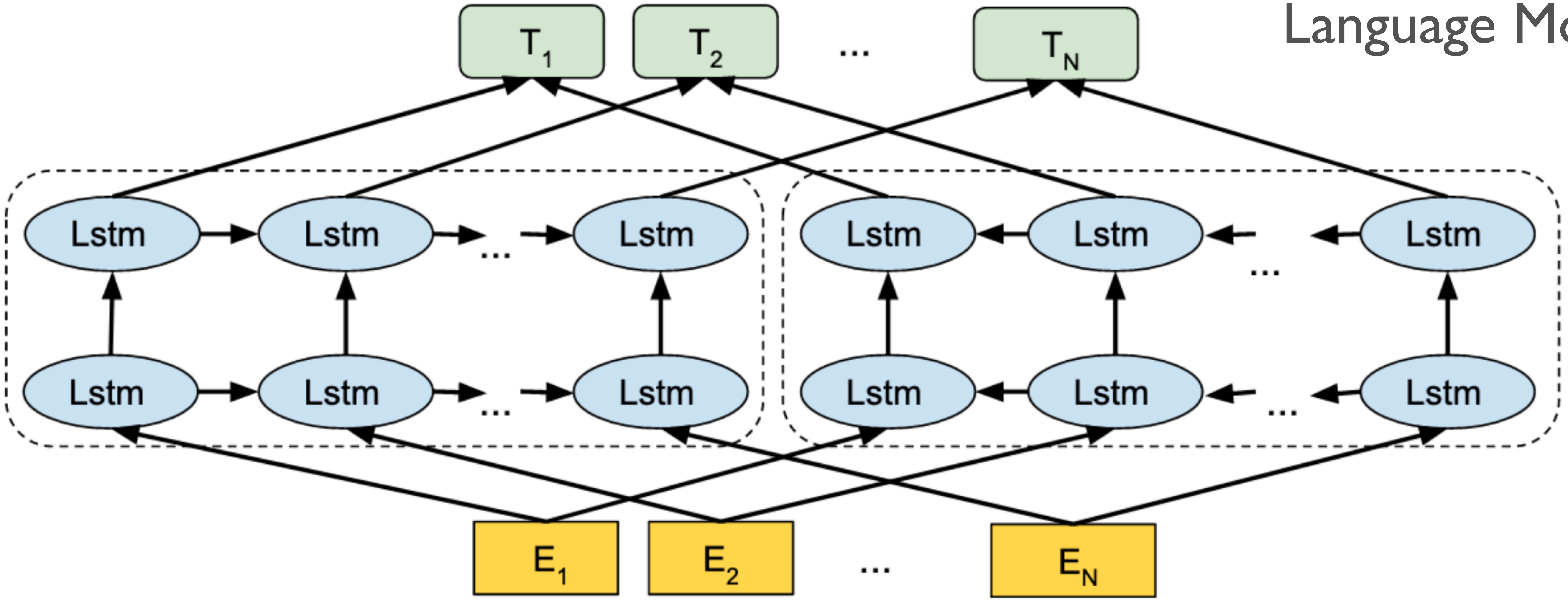
Combining the internal states in this manner allows for very rich word representations. Using intrinsic evaluations, we show that the higher-level LSTM states capture context-dependent aspects of word meaning (e.g., they can be used without modification to perform well on supervised

ELMo Model



Source: BERT paper

ELMo Model



Deep
Bidirectional
LSTM
Language Model

Source: BERT paper

Summary

- Vanilla / Simple / Elman RNNs:
  - Powerful, but susceptible to **vanishing gradients**
  - Because of **re-writing entire hidden state** each time step
- LSTMs + GRUs:
  - Use **gates** to control information flow
  - Additive connections across time steps help alleviate vanishing gradient problem
  - Interpretable and very powerful
- Moving forward: sequence-to-sequence (+ attention), and then overcoming a major RNN bottleneck (Transformers)