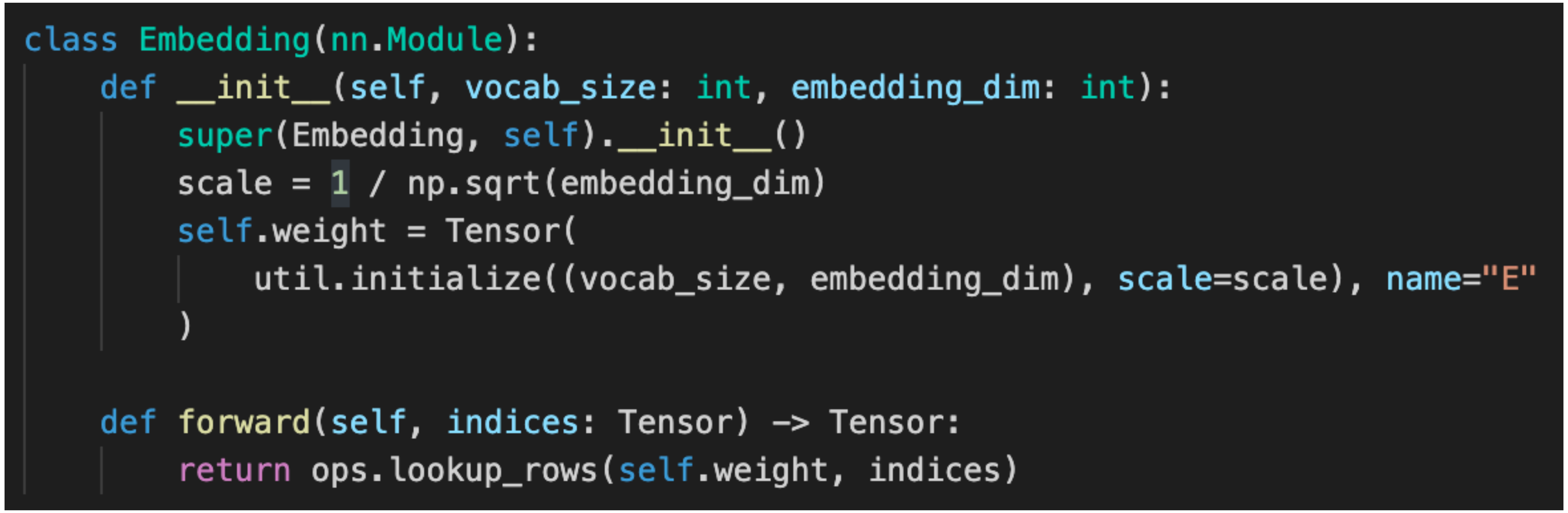
Recurrent Neural Networks: Vanishing Gradients, Gated Variants

LING 575K Deep Learning for NLP
Shane Steinert-Threlkeld
April 21 2021

Announcements

- Spec for run_hw3.sh in the pdf
- Word2Vec.forward:
  - Better argument names
  - target_indices: Tensor of shape [batch_size]
    - Integer indices of target words
  - context_indices: Tensor of shape [batch_size]
    - Integer indices of context words
- HW2, SGNS.forward: *one example at a time*
  - Here: a _batch_ of examples all at once

Embedding Layer



```python
class Embedding(nn.Module):
    def __init__(self, vocab_size: int, embedding_dim: int):
        super(Embedding, self).__init__()
        scale = 1 / np.sqrt(embedding_dim)
        self.weight = Tensor(
            util.initialize((vocab_size, embedding_dim), scale=scale), name="E"
        )

    def forward(self, indices: Tensor) -> Tensor:
        return ops.lookup_rows(self.weight, indices)
```

- Indices: [batch_size]
- Output of forward: [batch_size, embedding_dim]
- NB: don't need to call .forward, just call self.embedding(…) in Word2Vec.forward
- dot_product_rows: [batch_size, embedding_dim] x [batch_size, embedding_dim] —> [batch_size]

Plan for Today

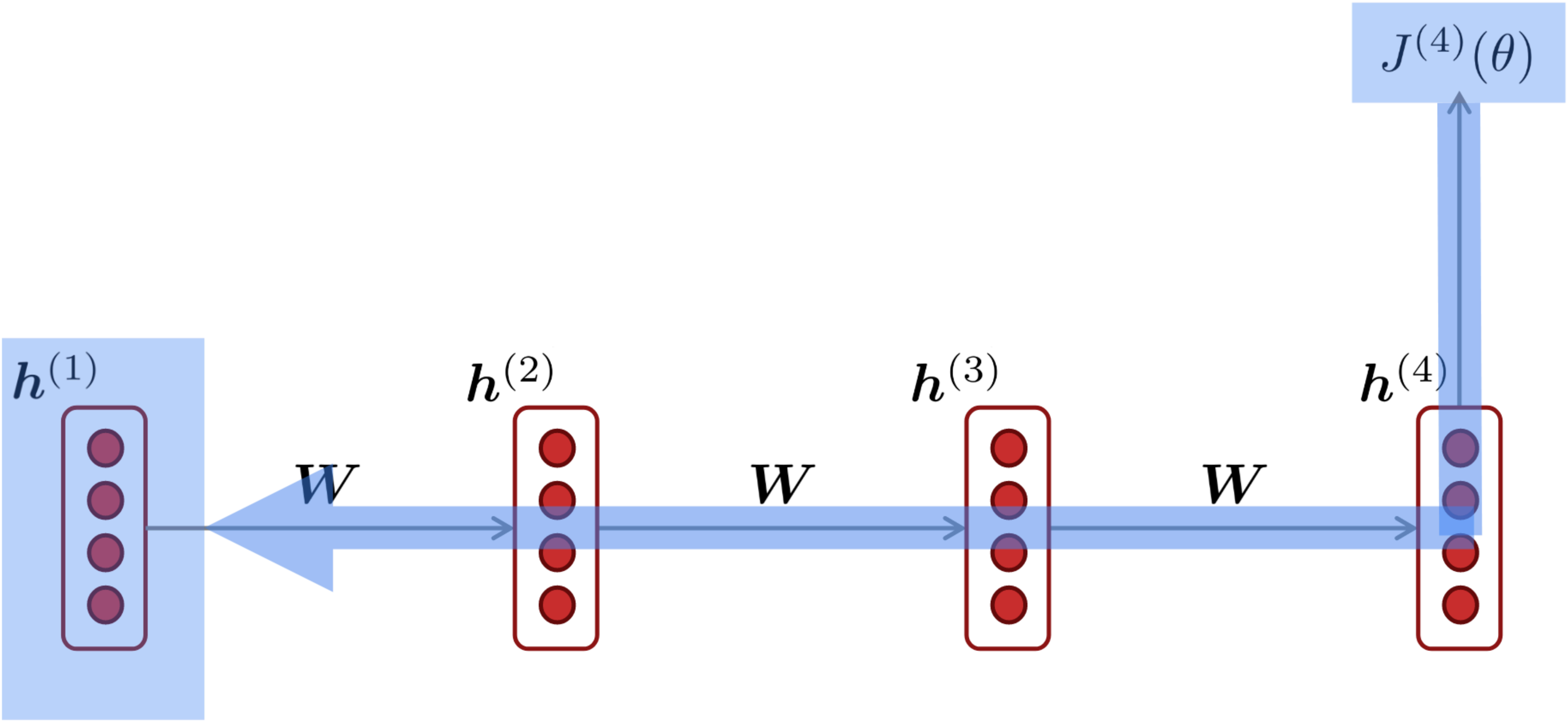
- Wrap up RNN slides
- Vanishing Gradients Problem
- Gating Based RNNs
  - LSTM
  - GRU

Vanishing Gradients

Vanishing/Exploding Gradients Problem

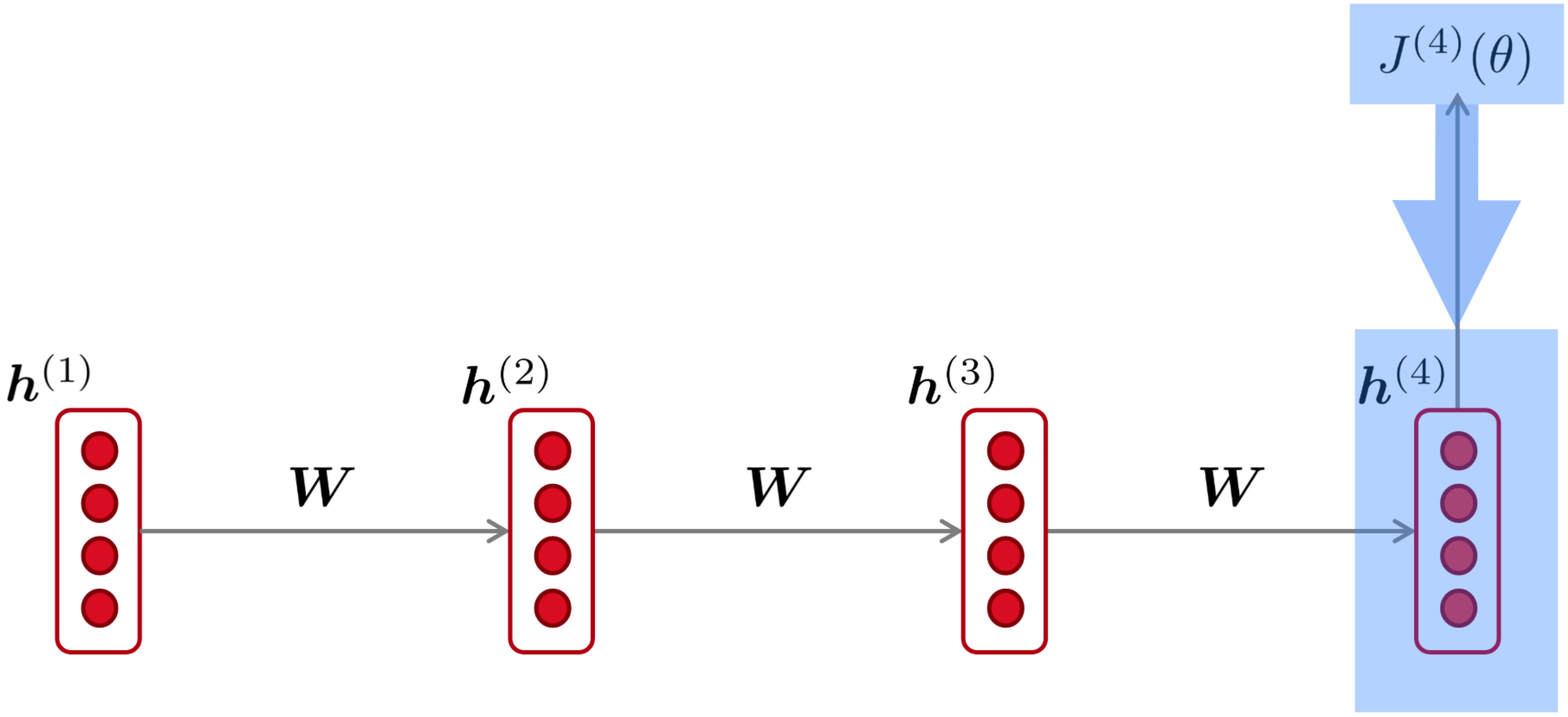
- BPTT with vanilla RNNs faces a major problem:
  - The gradients can *vanish* (approach 0) across time
  - This makes it hard/impossible to learn *long distance dependencies*, which are rampant in natural language

Vanishing Gradients



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

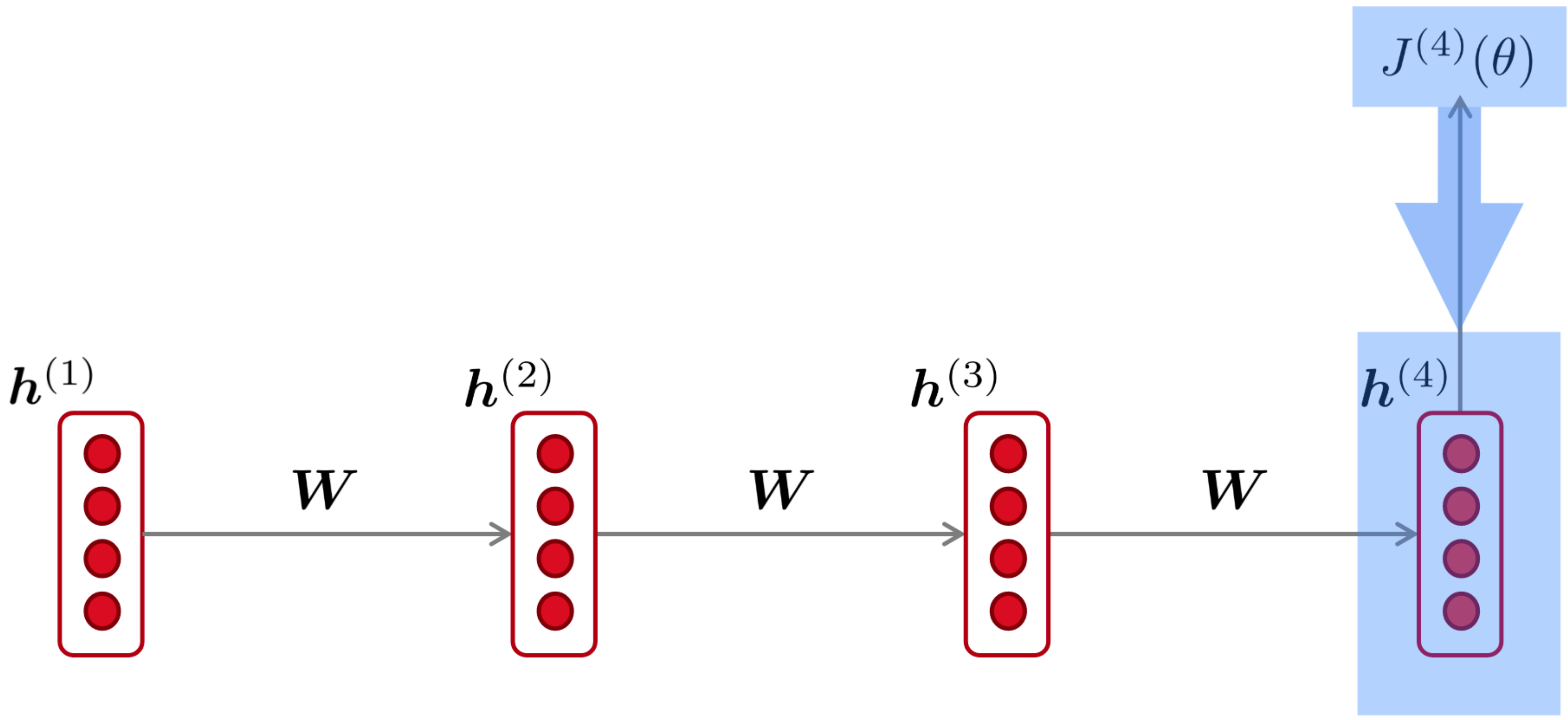
Vanishing Gradients



source

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

Vanishing Gradients

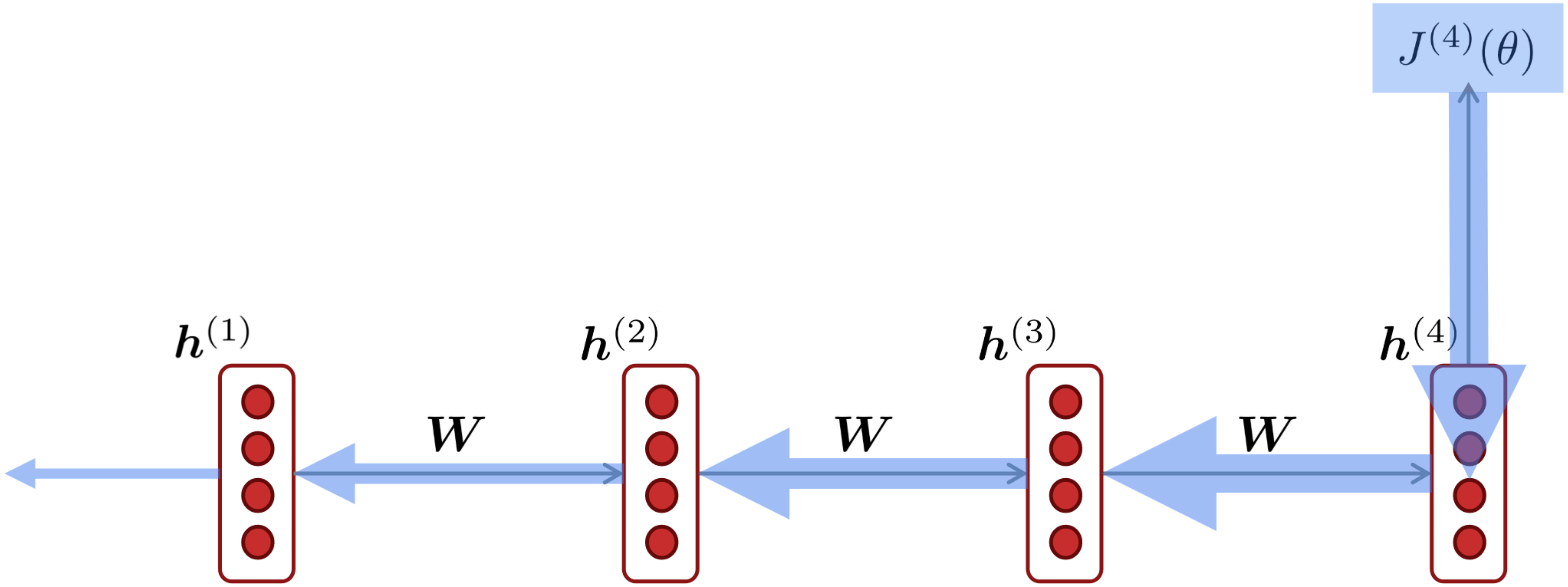


source

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

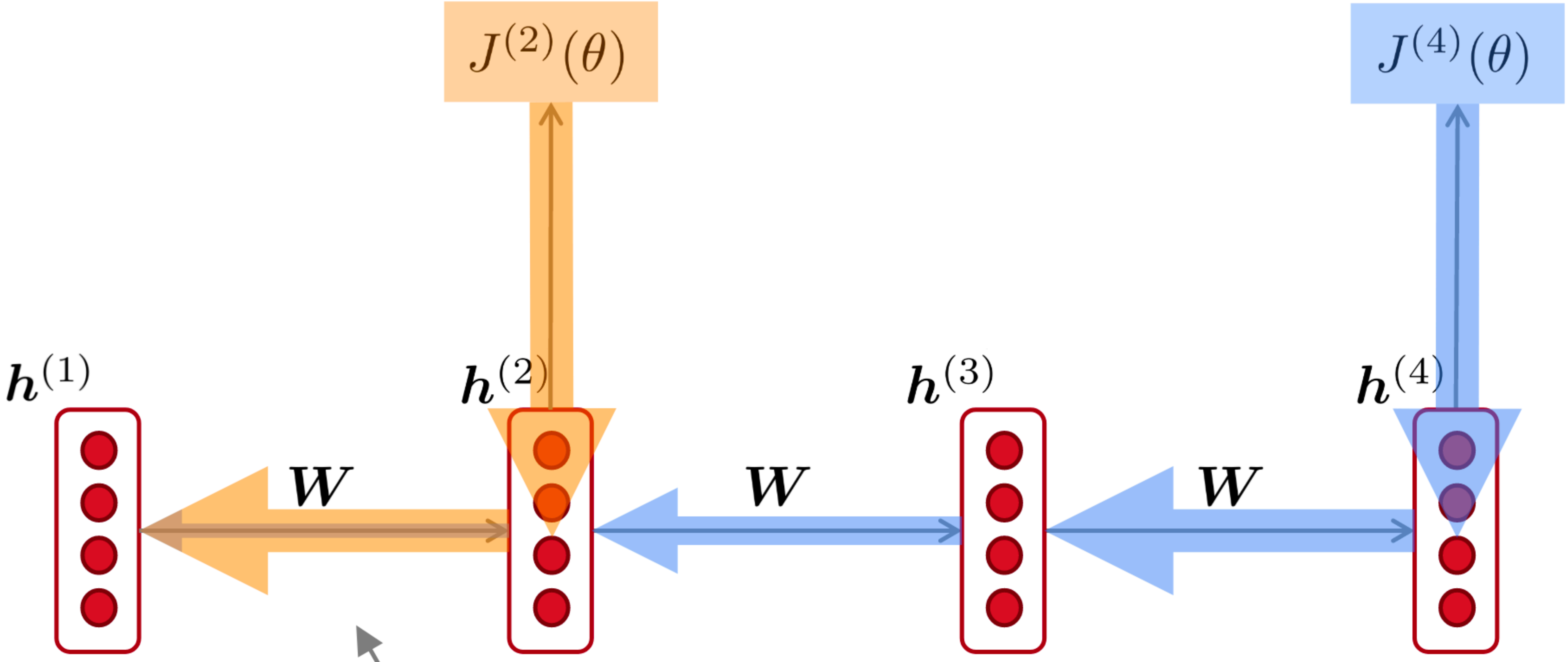
If these are small (depends on W), the effect from t=4 on t=1 will be very small

Vanishing Gradients



source

Vanishing Gradient Problem

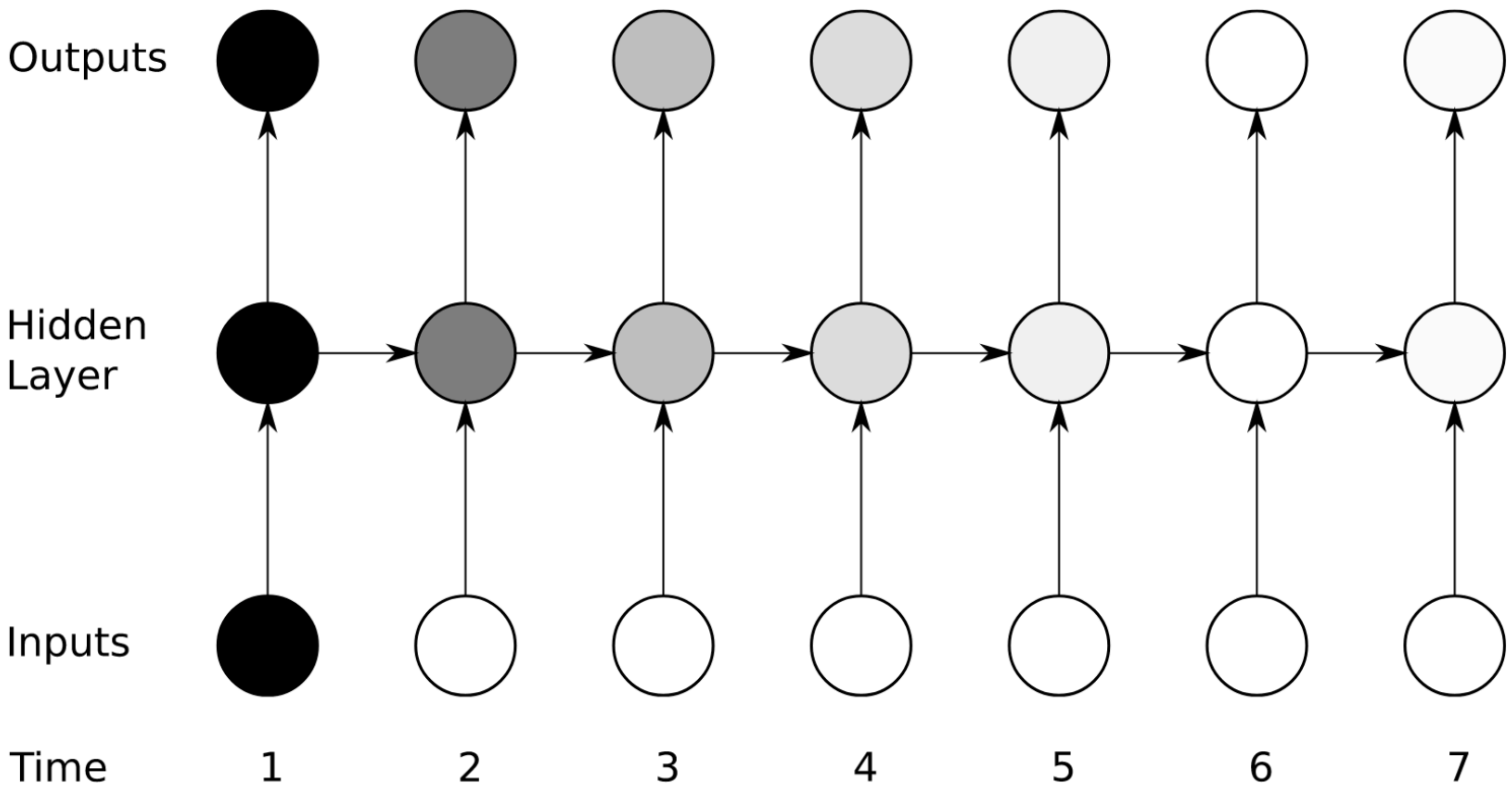


source

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.

Vanishing Gradient Problem



Graves 2012

Vanishing Gradient Problem

- 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

Examples of long-distance dependencies

- 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 (smooth) controlled by *gates*
  - f_t: forget gate
  - i_t: input gate
  - o_t: output gate

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

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

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

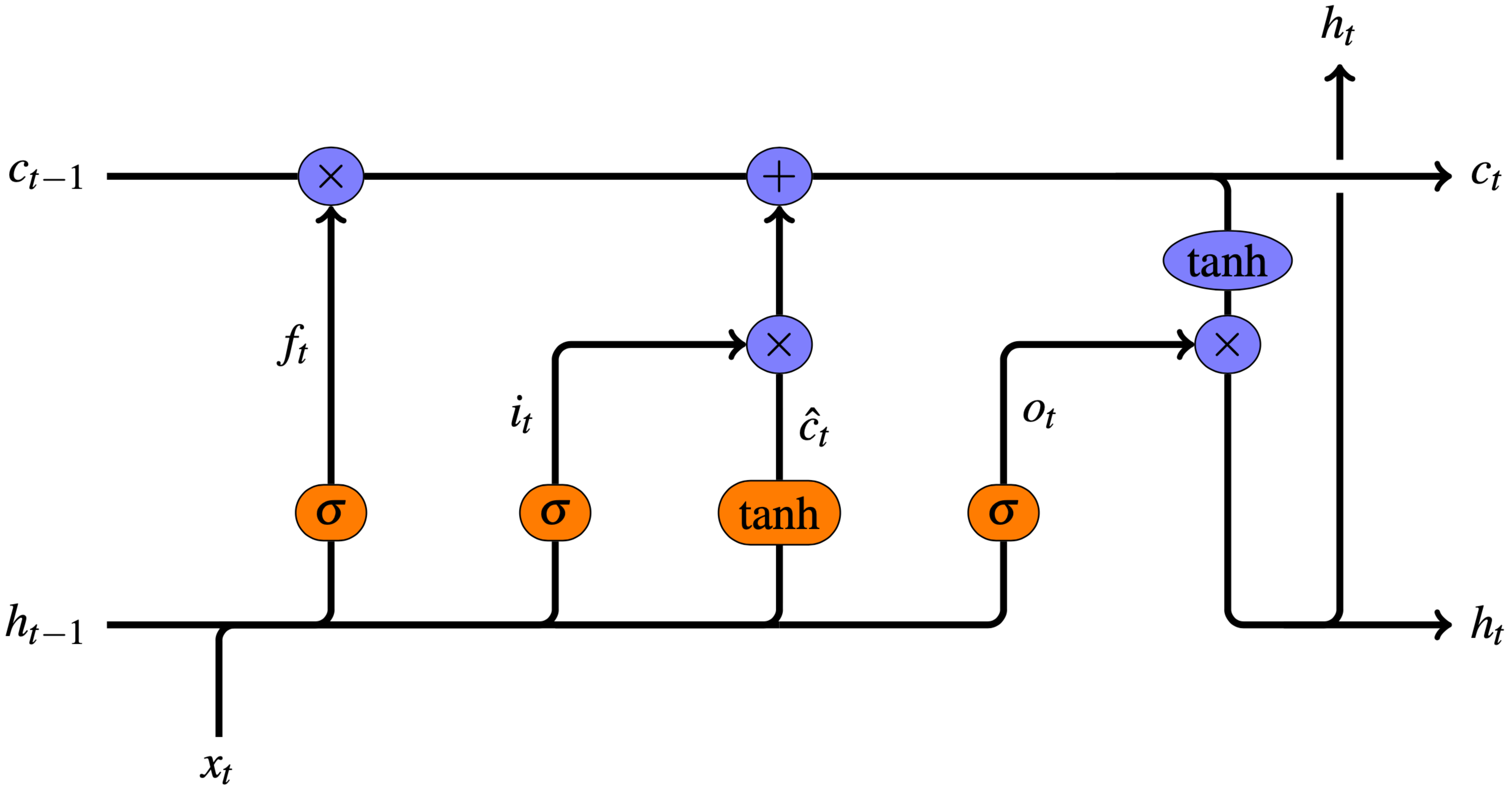
$$c_t = f_t \odot c_{t-1} + i_t \odot \hat{c}_t$$

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

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

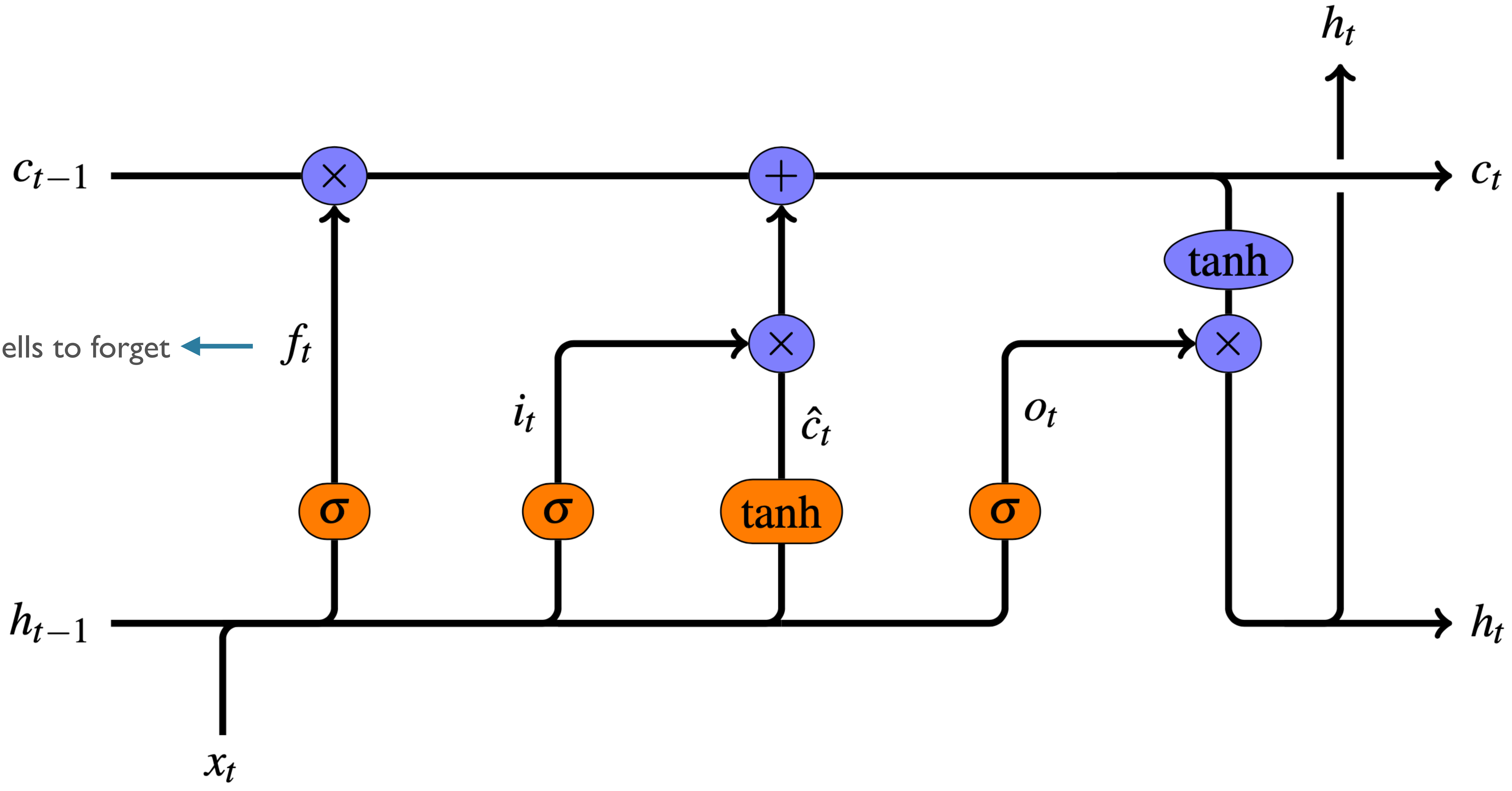
🤔🤔🤷‍♀️🤮

LSTMs



Steinert-Threlkeld and Szymanik 2019; Olah 2015

LSTMs



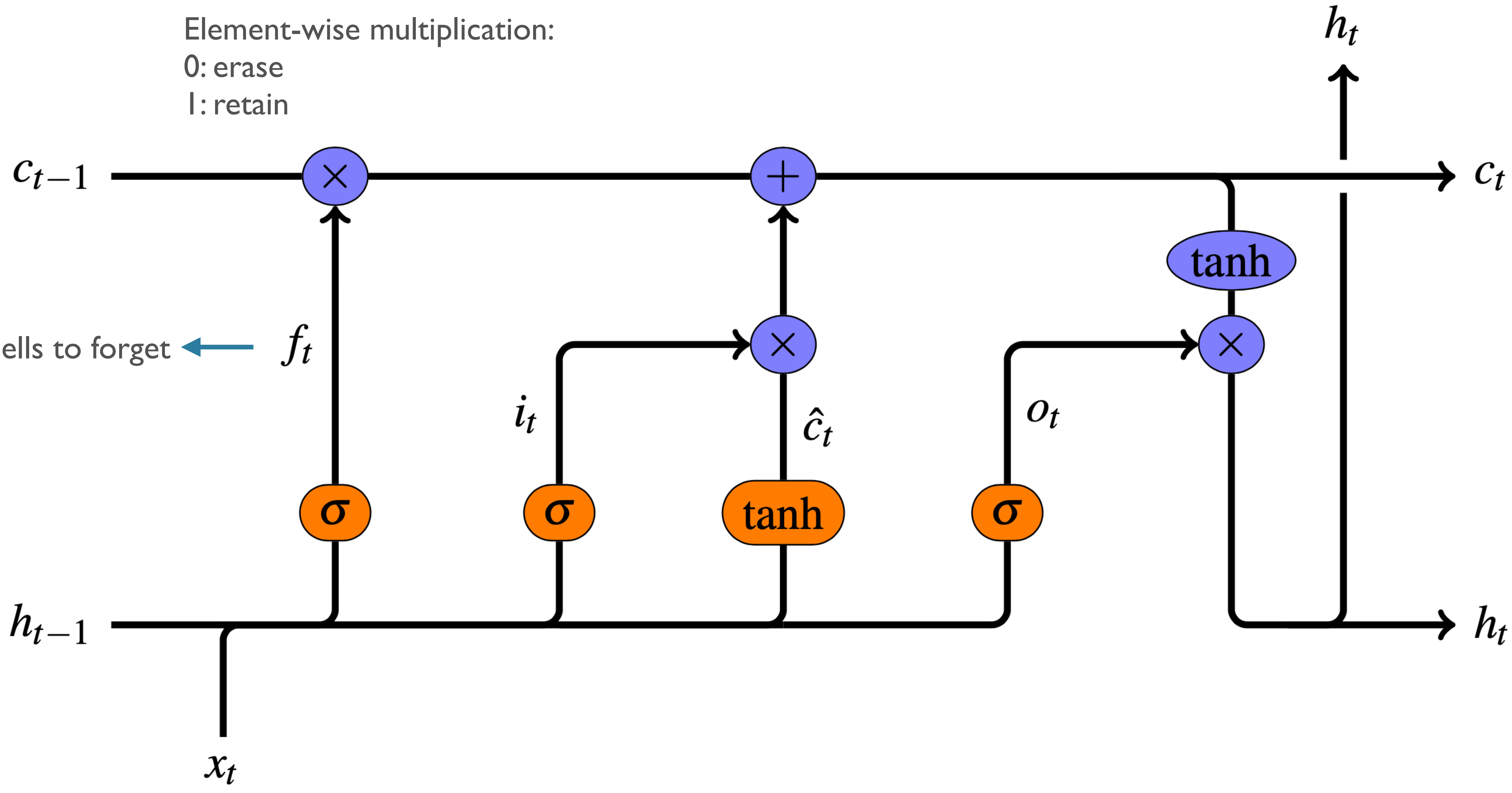
$f_t \in [0,1]^m$: which cells to forget

Steinert-Threlkeld and Szymanik 2019; Olah 2015

LSTMs

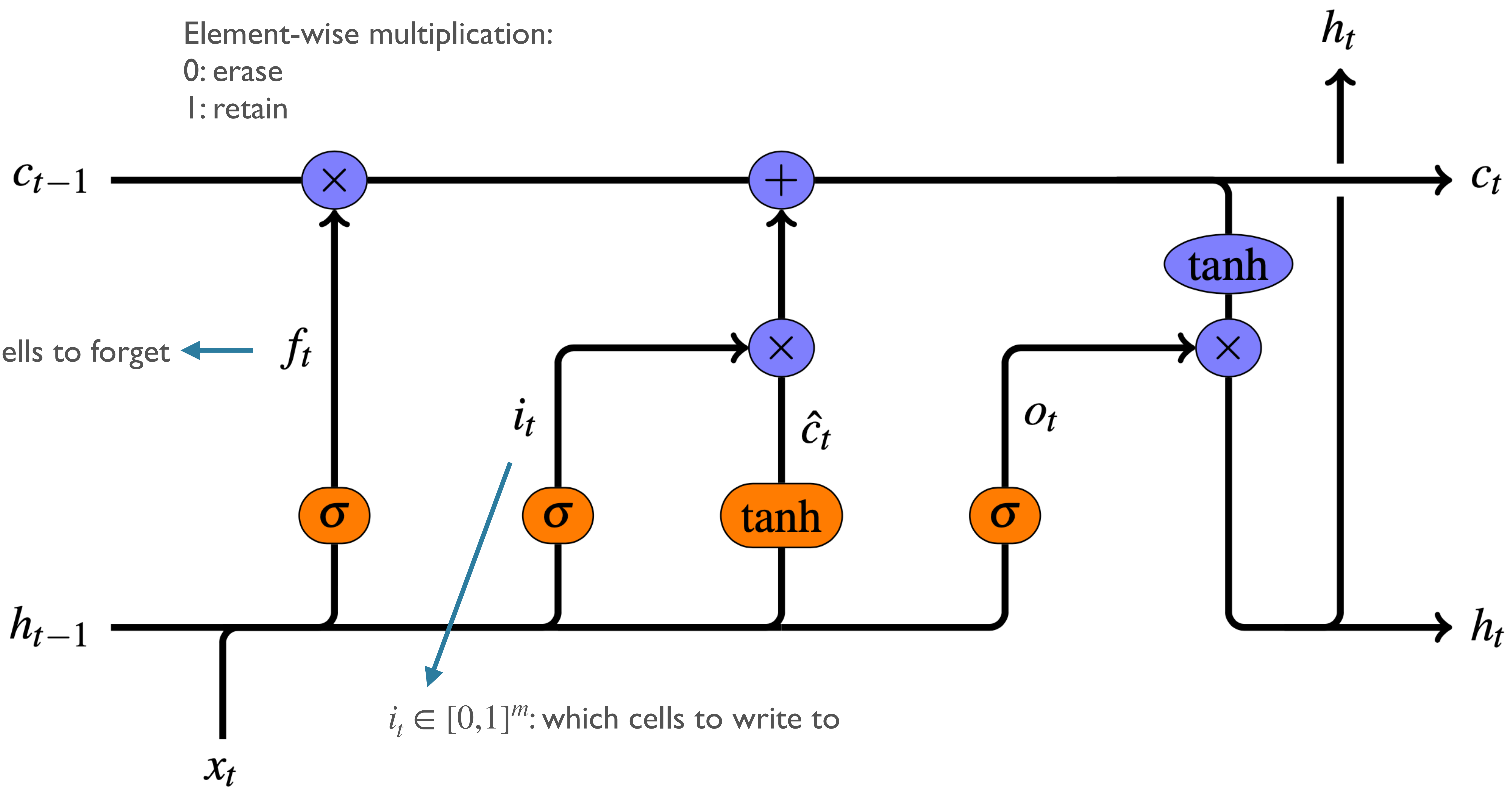
Element-wise multiplication:
0: erase
1: retain

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



Steinert-Threlkeld and Szymanik 2019; Olah 2015

LSTMs



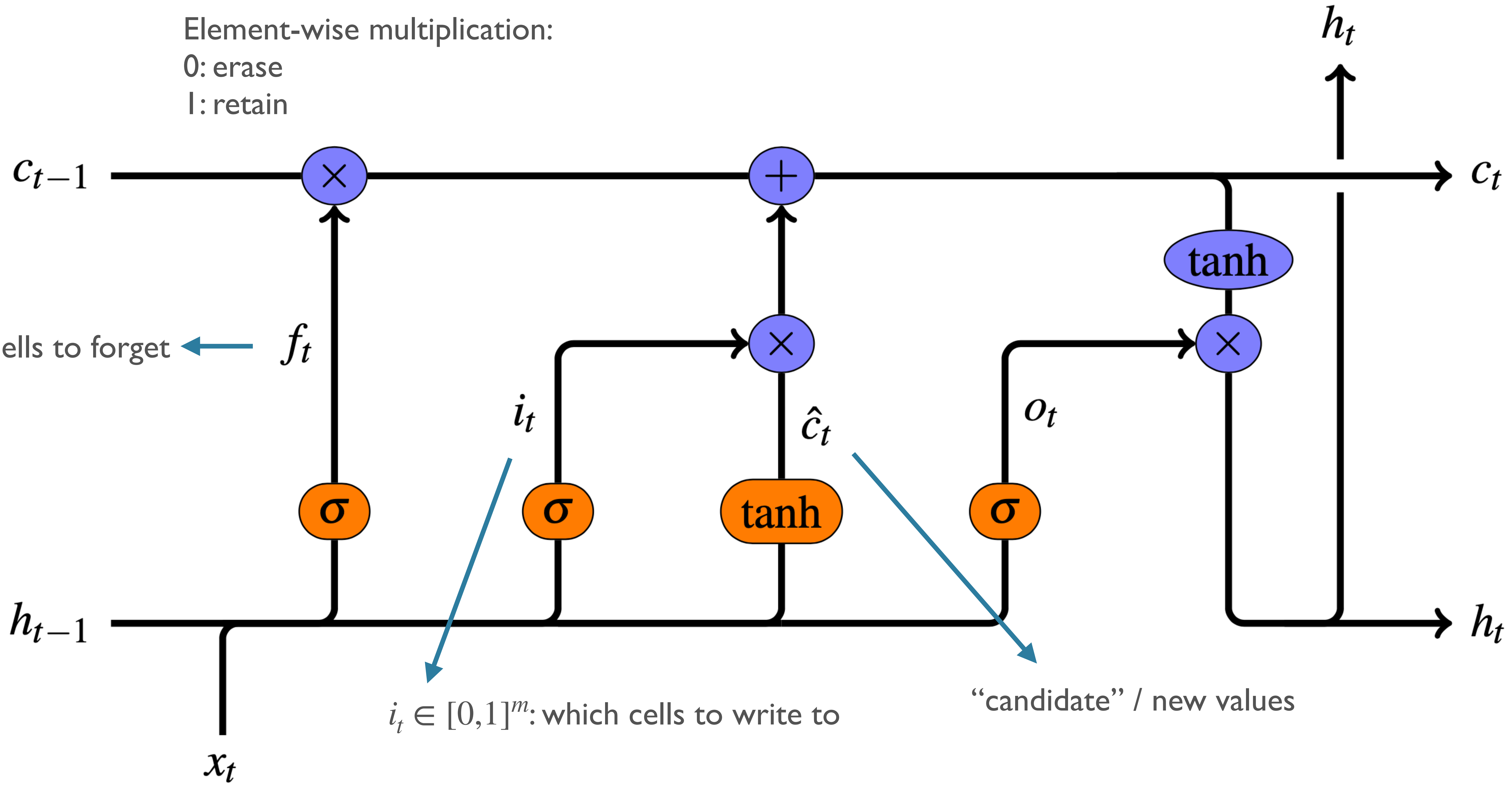
Element-wise multiplication:
0: erase
1: retain

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

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

Steinert-Threlkeld and Szymanik 2019; Olah 2015

LSTMs



Element-wise multiplication:
0: erase
1: retain

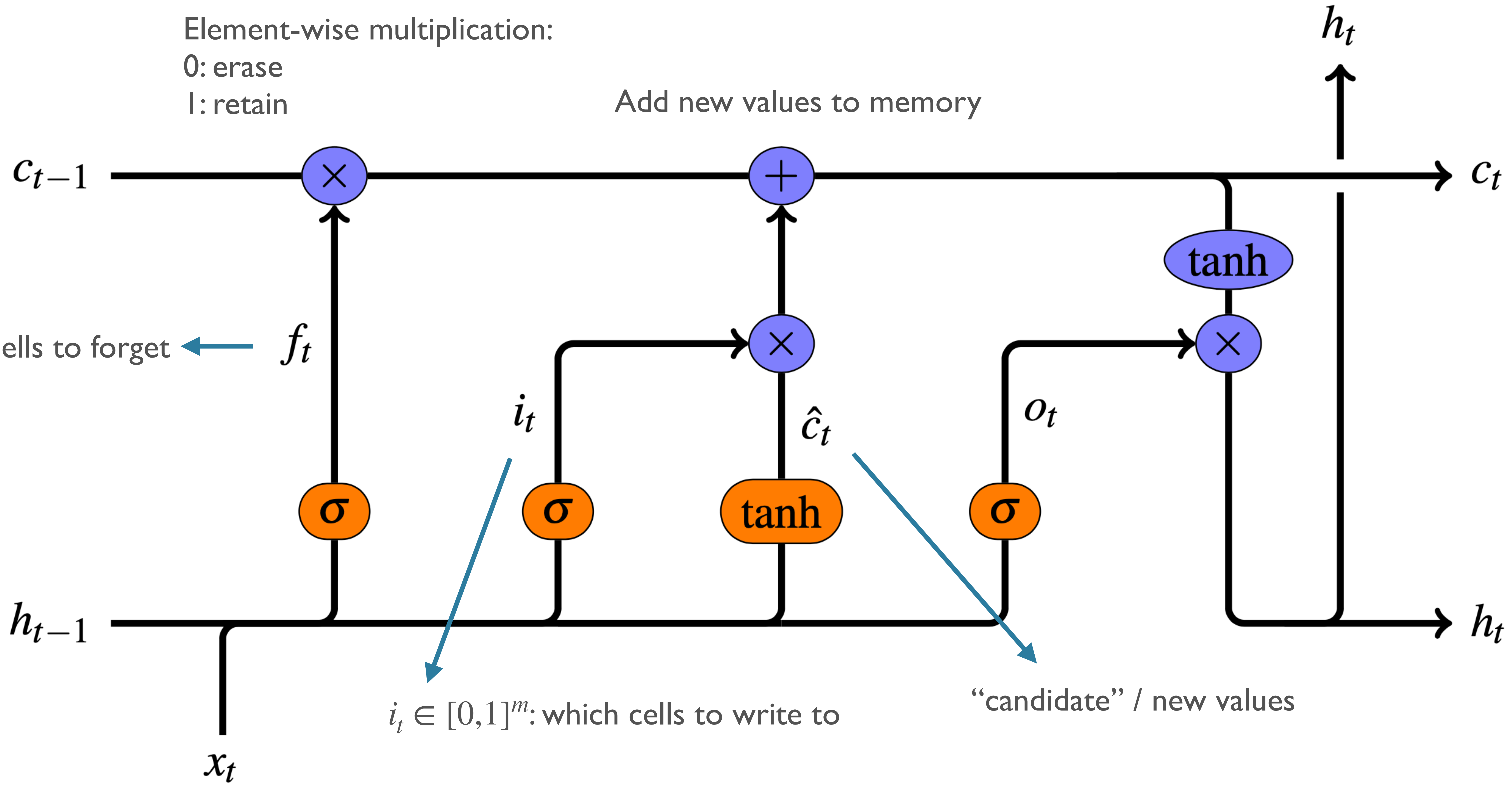
$f_t \in [0,1]^m$: which cells to forget

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

“candidate” / new values

Steinert-Threlkeld and Szymanik 2019; Olah 2015

LSTMs



Element-wise multiplication:
0: erase
1: retain

Add new values to memory

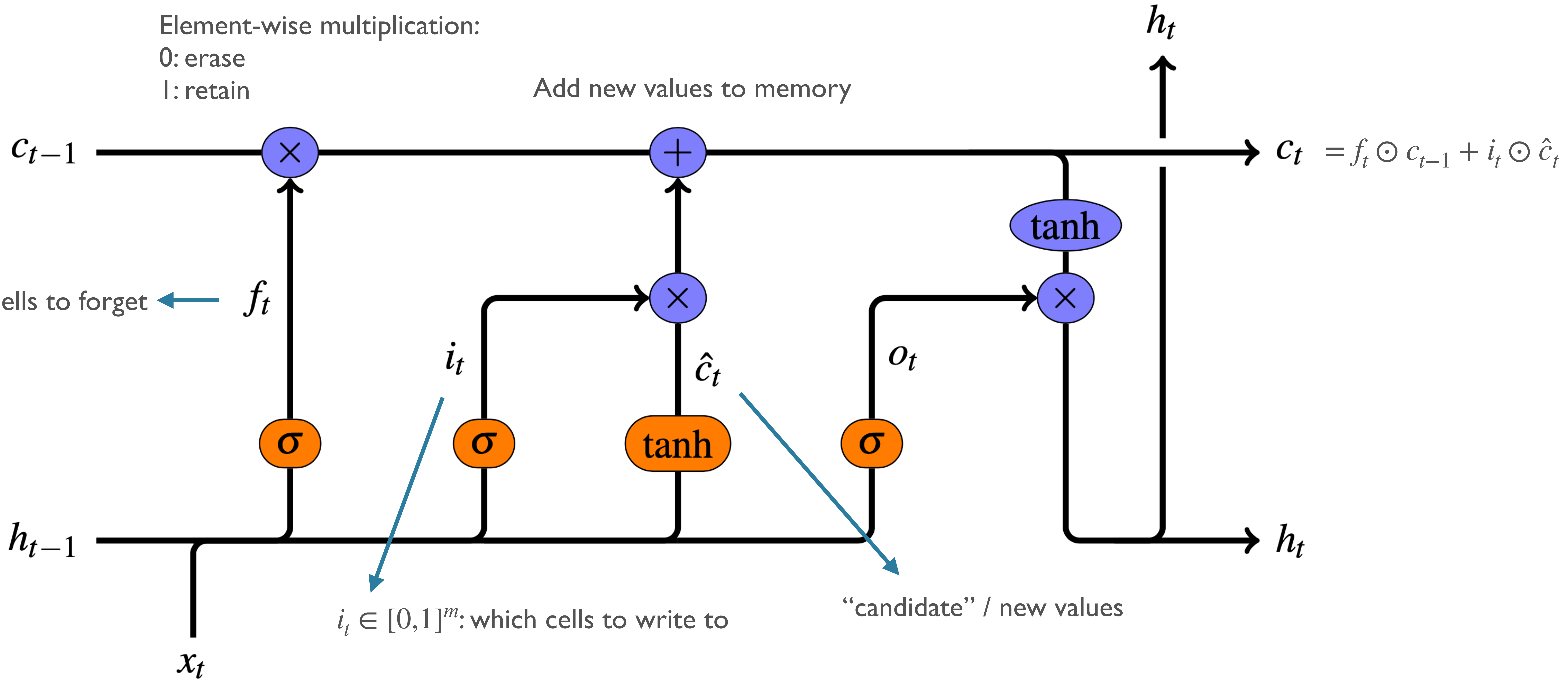
$f_t \in [0,1]^m$: which cells to forget

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

"candidate" / new values

Steinert-Threlkeld and Szymanik 2019; Olah 2015

LSTMs



Element-wise multiplication:
0: erase
1: retain

Add new values to memory

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

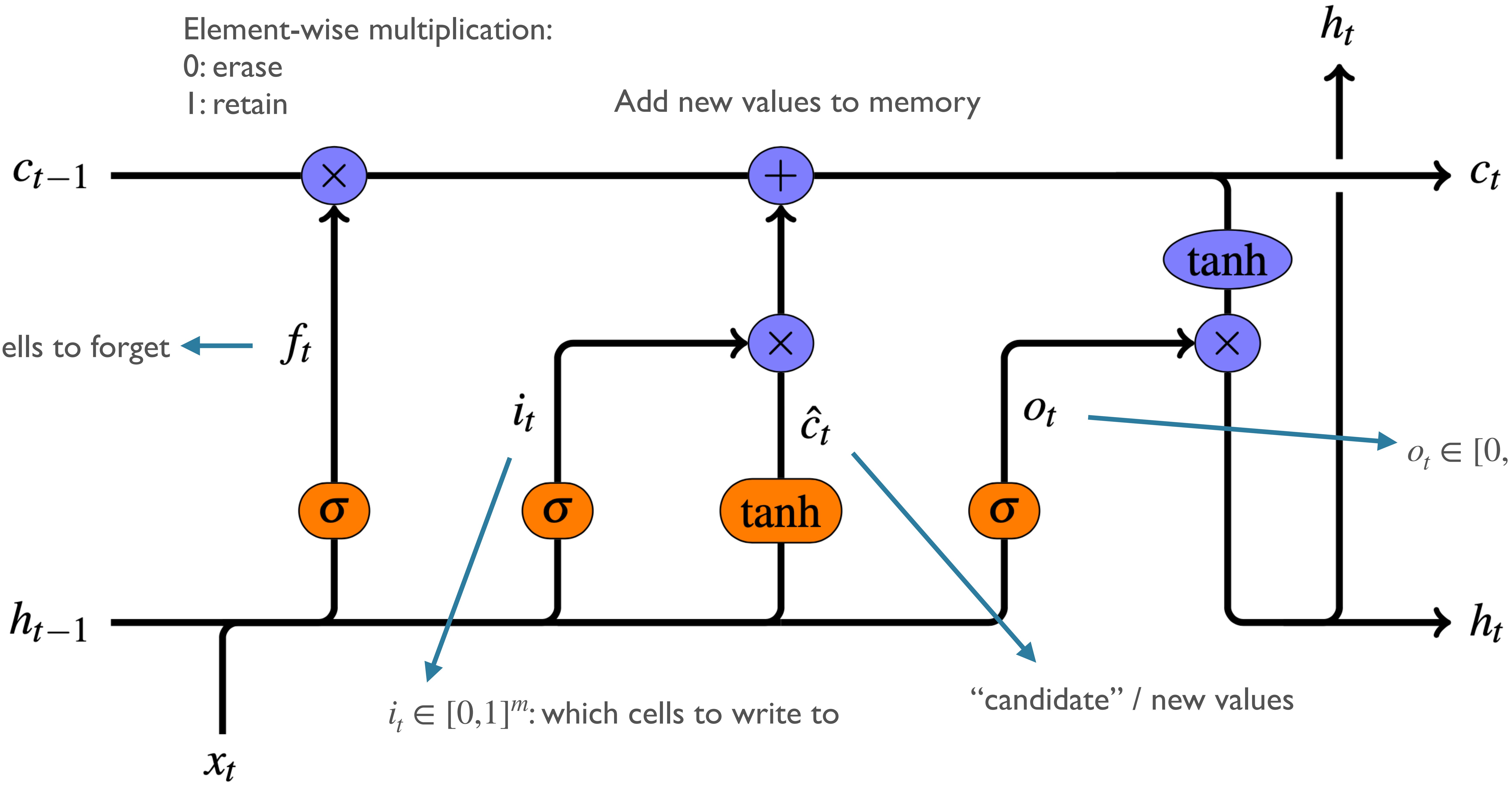
$f_t \in [0,1]^m$: which cells to forget

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

"candidate" / new values

Steinert-Threlkeld and Szymanik 2019; Olah 2015

LSTMs



Element-wise multiplication:
0: erase
1: retain

Add new values to memory

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

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

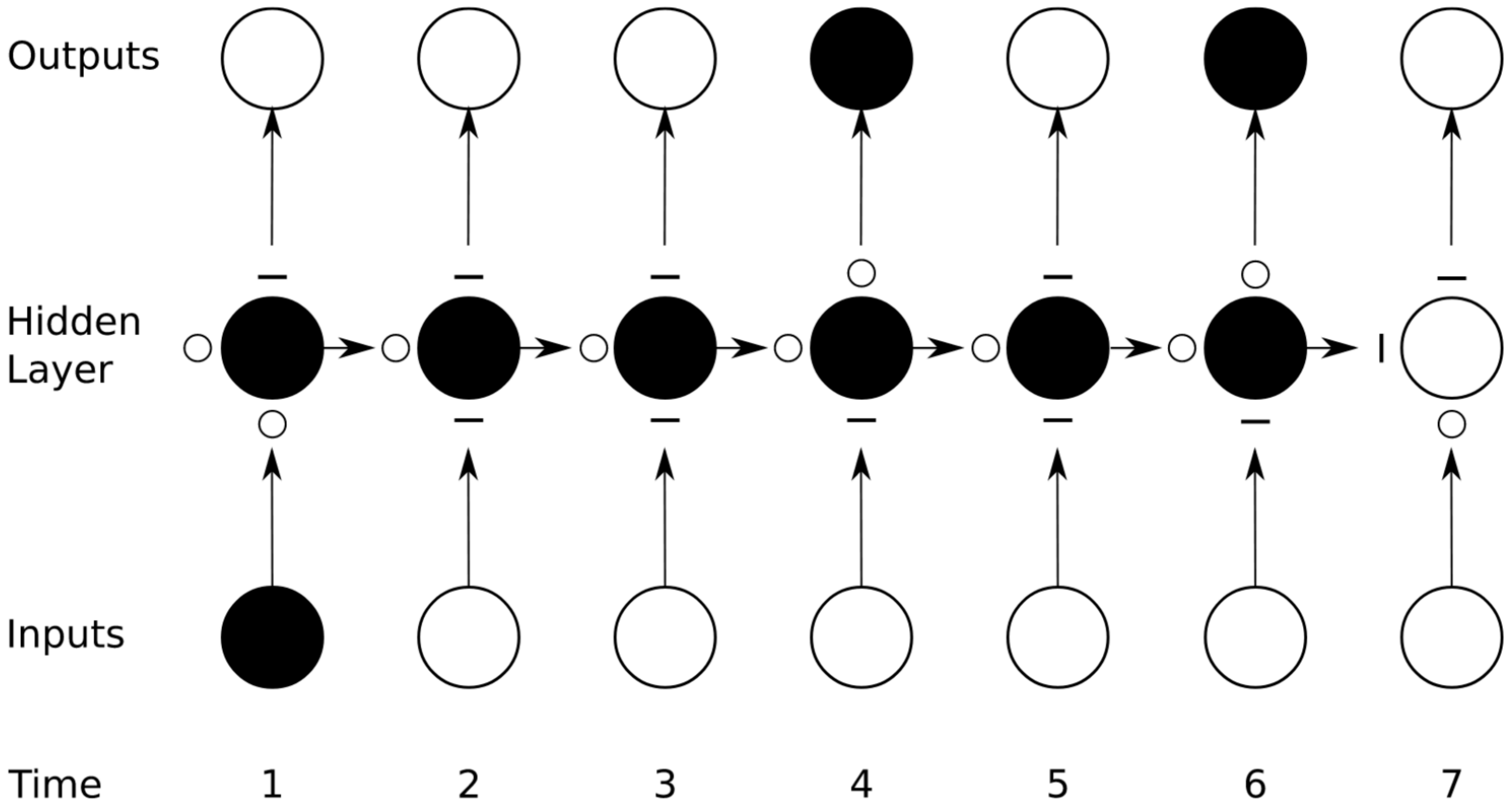
$o_t \in [0,1]^m$: which cells to output

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

“candidate” / new values

Steinert-Threlkeld and Szymanik 2019; Olah 2015

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

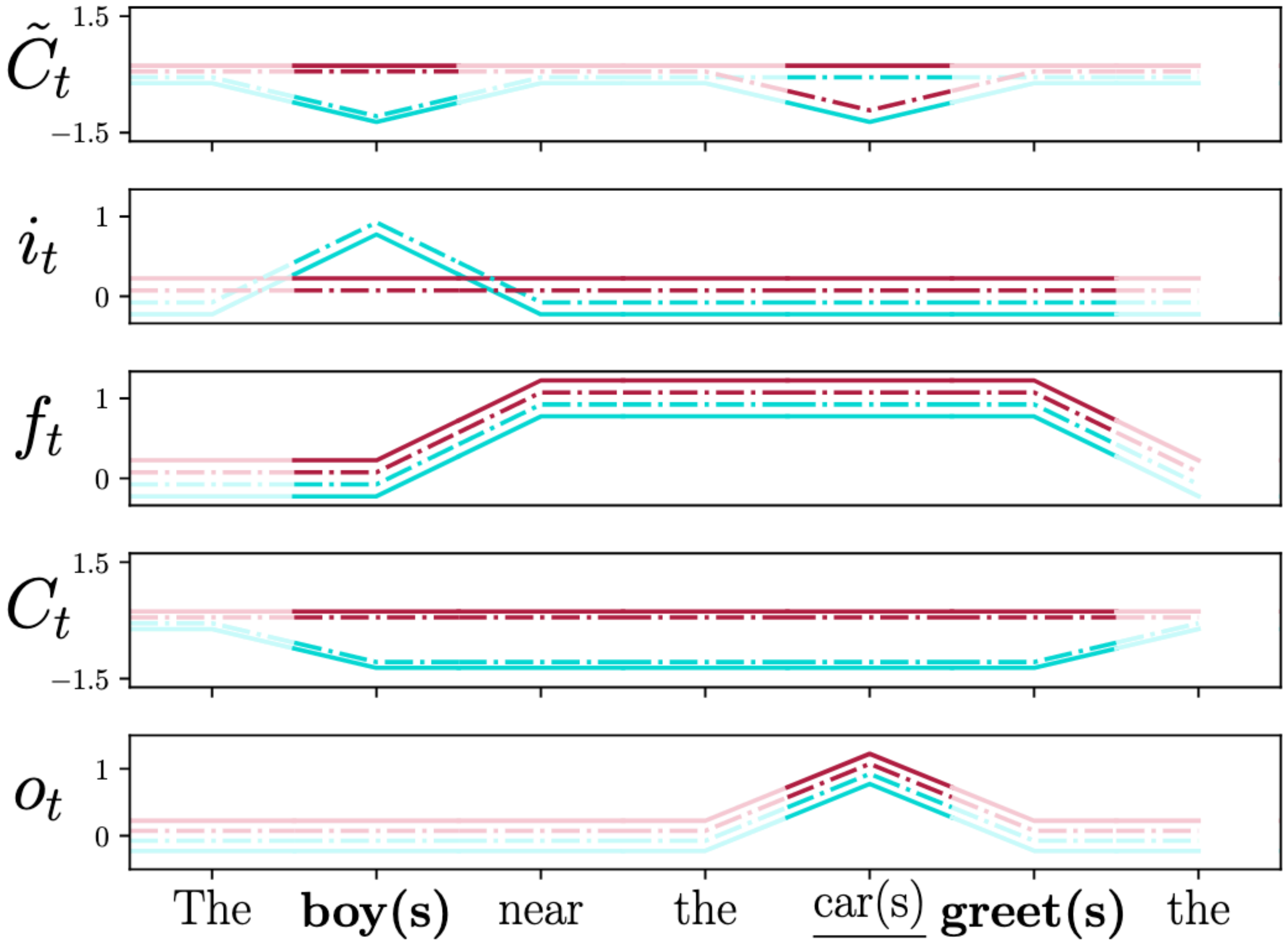
Theo Desbordes
Facebook AI Research
Paris, France
tdesbordes@fb.com

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

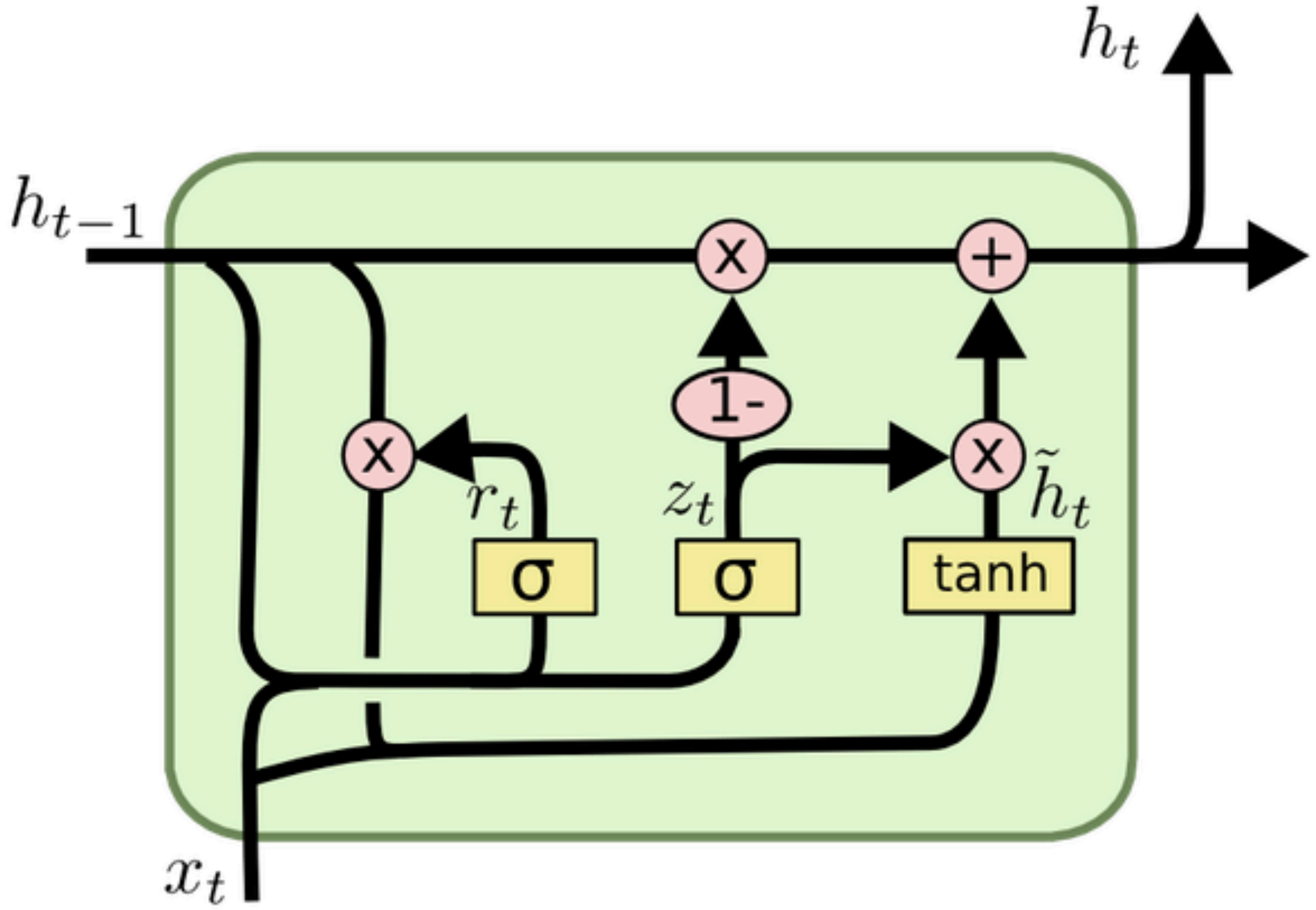
$$u_t = \sigma(W_u h_{t-1} + U_u x_t + b_u)$$

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

$$\tilde{h}_t = \tanh(W_h(r_t \odot h_t) + U_h x_t + b_h)$$

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

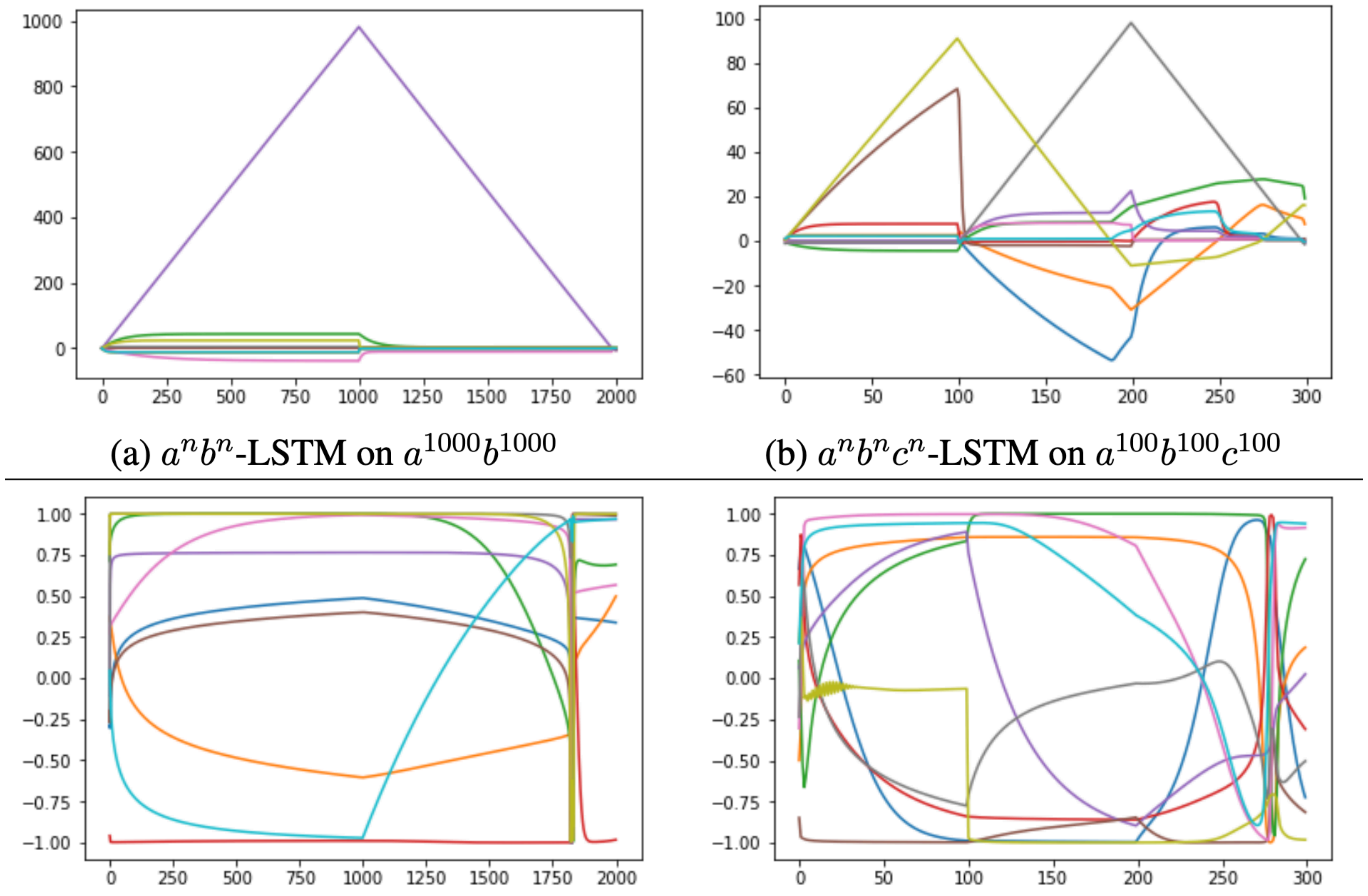
Gated Recurrent Unit



source

LSTM vs GRU

- Generally: LSTM 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 $a^{100}b^{100}c^{100}$

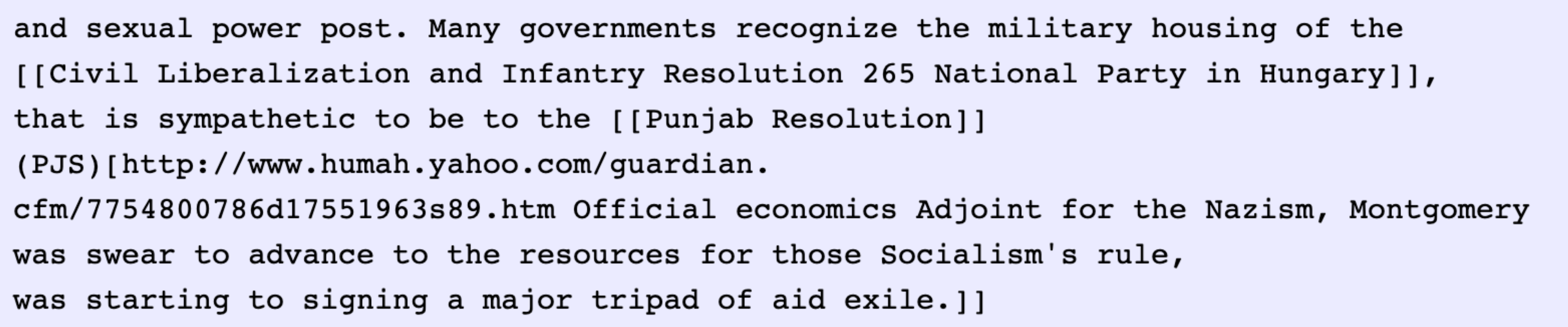
source

Odds and Ends

Fun with LSTM (character) LMs

- Generating text with an LM:
  - Feed initial token (e.g. BOS, or just a word/character)
  - Generate probability over next tokens
    - Sample next token from this distribution
  - Repeat until [EOS | max length | other criterion]

Fun with LSTM (character) LMs

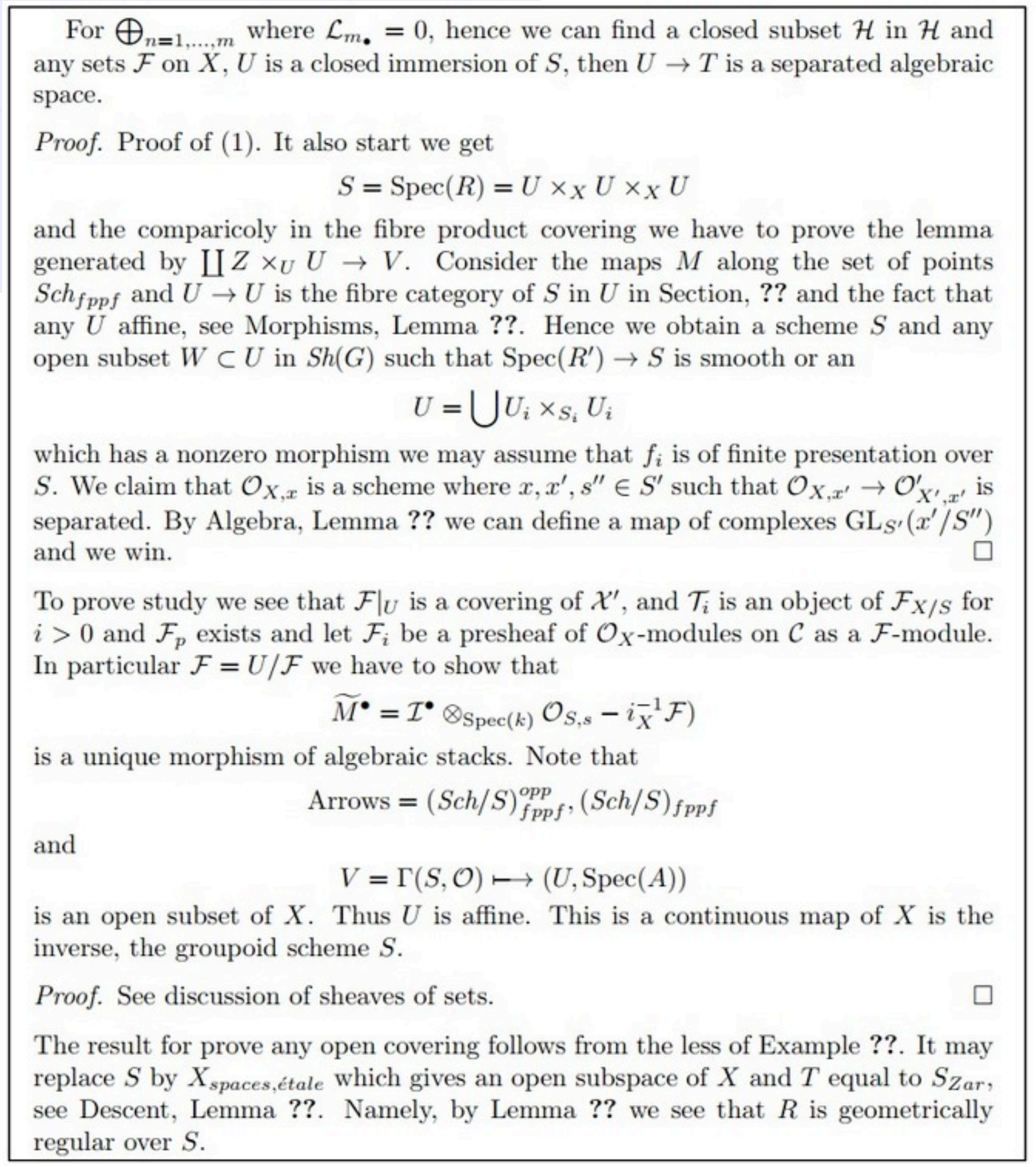


```
and sexual power post. Many governments recognize the military housing of the
[[Civil Liberalization and Infantry Resolution 265 National Party in Hungary]],
that is sympathetic to be to the [[Punjab Resolution]]
(PJS)[http://www.humah.yahoo.com/guardian.
cfm/7754800786d17551963s89.htm Official economics Adjoint for the Nazism, Montgomery
was swear to advance to the resources for those Socialism's rule,
was starting to signing a major tripad of aid exile.]]
```

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

Fun with LSTM (character) LMs

```
and sexual power post. Many governments recognize the military housing of the
[[Civil Liberalization and Infantry Resolution 265 National Party in Hungary]],
that is sympathetic to be to the [[Punjab Resolution]]
(PJS)[http://www.humah.yahoo.com/guardian.
cfm/7754800786d17551963s89.htm Official economics Adjoint for the Na
was swear to advance to the resources for those Socialism's rule,
was starting to signing a major tripad of aid exile.]]
```



For $\bigoplus_{n=1,\dots,m}$ where $\mathcal{L}_{m_\bullet} = 0$, hence we can find a closed subset $\mathcal{H}$ in $\mathcal{H}$ and any sets $\mathcal{F}$ on X, U is a closed immersion of S, then $U \to T$ is a separated algebraic space.

Proof. Proof of (1). It also start we get

$$S = \mathrm{Spec}(R) = U \times_X U \times_X U$$

and the comparicoly in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \to V$. Consider the maps M along the set of points Sch_{fppf} and $U \to U$ is the fibre category of S in U in Section, **??** and the fact that any U affine, see Morphisms, Lemma **??**. Hence we obtain a scheme S and any open subset $W \subset U$ in $Sh(G)$ such that $\mathrm{Spec}(R') \to S$ is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that f_i is of finite presentation over S. We claim that $\mathcal{O}_{X,x}$ is a scheme where $x, x', s'' \in S'$ such that $\mathcal{O}_{X,x'} \to \mathcal{O}'_{X',x'}$ is separated. By Algebra, Lemma **??** we can define a map of complexes $\mathrm{GL}_{S'}(x'/S'')$ and we win. □

To prove study we see that $\mathcal{F}|_U$ is a covering of $\mathcal{X}'$, and $\mathcal{T}_i$ is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and $\mathcal{F}_p$ exists and let $\mathcal{F}_i$ be a presheaf of $\mathcal{O}_X$-modules on $\mathcal{C}$ as a $\mathcal{F}$-module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\widetilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\mathrm{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1}\mathcal{F})$$

is a unique morphism of algebraic stacks. Note that

$$\mathrm{Arrows} = (Sch/S)^{opp}_{fppf}, (Sch/S)_{fppf}$$

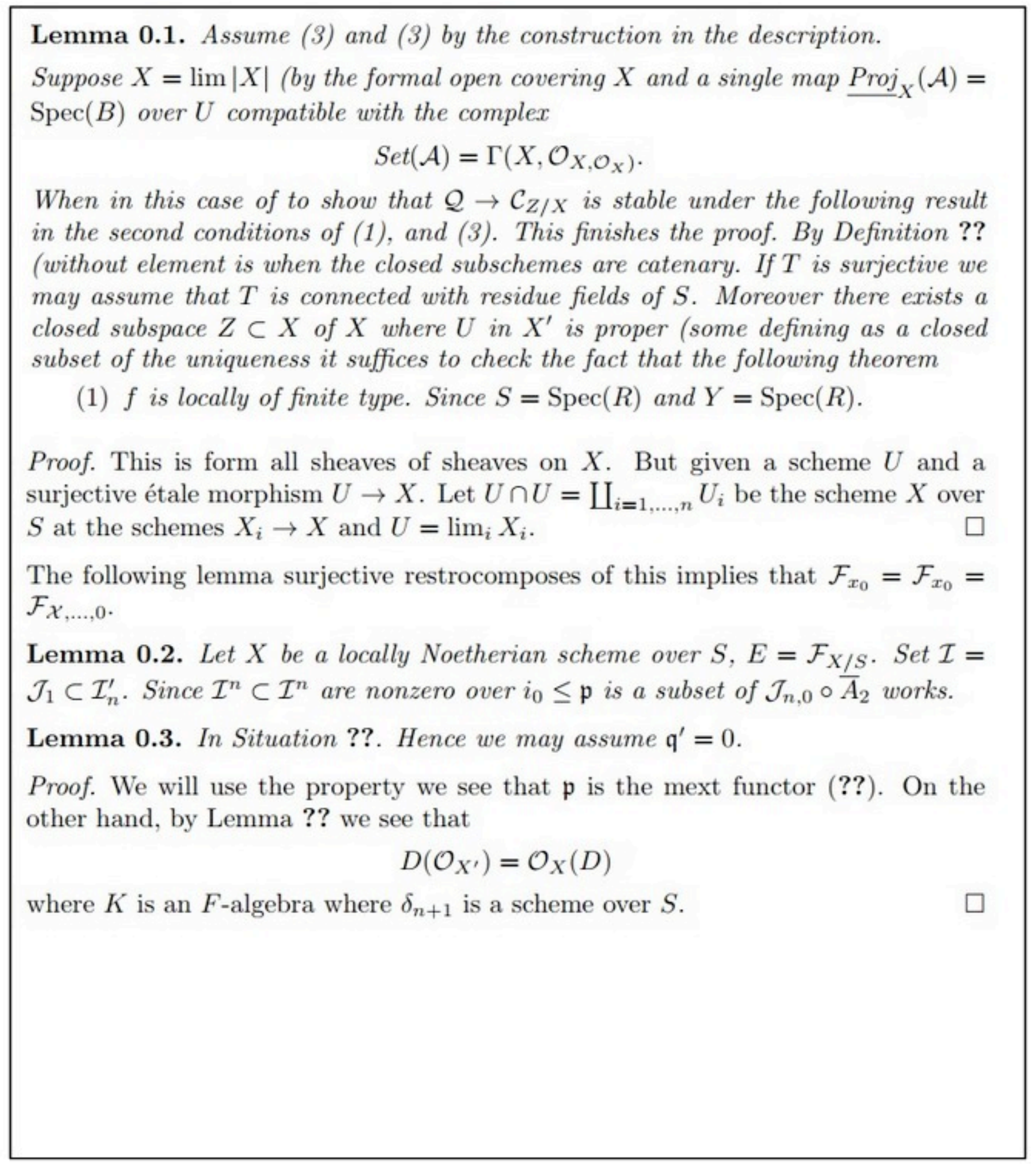
and

$$V = \Gamma(S, \mathcal{O}) \longmapsto (U, \mathrm{Spec}(A))$$

is an open subset of X. Thus U is affine. This is a continuous map of X is the inverse, the groupoid scheme S.

Proof. See discussion of sheaves of sets. □

The result for prove any open covering follows from the less of Example **??**. It may replace S by $X_{spaces,étale}$ which gives an open subspace of X and T equal to S_{Zar}, see Descent, Lemma **??**. Namely, by Lemma **??** we see that R is geometrically regular over S.



Lemma 0.1. *Assume (3) and (3) by the construction in the description.*

Suppose $X = \lim |X|$ *(by the formal open covering* X *and a single map* $\underline{Proj}_X(\mathcal{A}) = \mathrm{Spec}(B)$ *over* U *compatible with the complex*

$$Set(\mathcal{A}) = \Gamma(X, \mathcal{O}_{X,\mathcal{O}_X}).$$

When in this case of to show that $\mathcal{Q} \to \mathcal{C}_{Z/X}$ *is stable under the following result in the second conditions of (1), and (3). This finishes the proof. By Definition* **??** *(without element is when the closed subschemes are catenary. If* T *is surjective we may assume that* T *is connected with residue fields of* S*. Moreover there exists a closed subspace* $Z \subset X$ *of* X *where* U *in* X' *is proper (some defining as a closed subset of the uniqueness it suffices to check the fact that the following theorem*

(1) *f is locally of finite type. Since* $S = \mathrm{Spec}(R)$ *and* $Y = \mathrm{Spec}(R)$.

Proof. This is form all sheaves of sheaves on X. But given a scheme U and a surjective étale morphism $U \to X$. Let $U \cap U = \coprod_{i=1,\dots,n} U_i$ be the scheme X over S at the schemes $X_i \to X$ and $U = \lim_i X_i$. □

The following lemma surjective restrocomposes of this implies that $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{\mathcal{X},\dots,0}$.

Lemma 0.2. *Let* X *be a locally Noetherian scheme over* S, $E = \mathcal{F}_{X/S}$*. Set* $\mathcal{I} = \mathcal{J}_1 \subset \mathcal{I}'_n$*. Since* $\mathcal{I}^n \subset \mathcal{I}^n$ *are nonzero over* $i_0 \leq \mathfrak{p}$ *is a subset of* $\mathcal{J}_{n,0} \circ \overline{A}_2$ *works.*

Lemma 0.3. *In Situation* **??***. Hence we may assume* $\mathfrak{q}' = 0$.

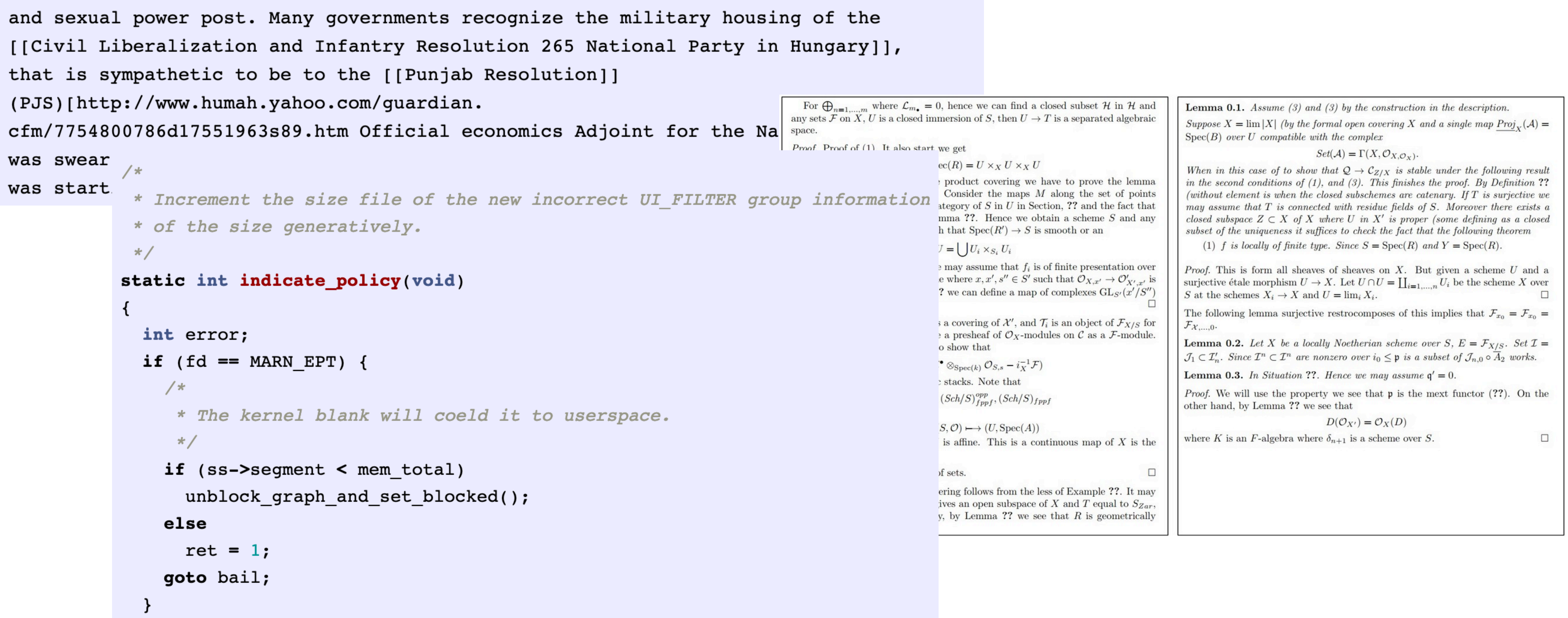
Proof. We will use the property we see that $\mathfrak{p}$ is the mext functor (**??**). On the other hand, by Lemma **??** we see that

$$D(\mathcal{O}_{X'}) = \mathcal{O}_X(D)$$

where K is an F-algebra where δ_{n+1} is a scheme over S. □

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

Fun with LSTM (character) LMs



```
and sexual power post. Many governments recognize the military housing of the
[[Civil Liberalization and Infantry Resolution 265 National Party in Hungary]],
that is sympathetic to be to the [[Punjab Resolution]]
(PJS)[http://www.humah.yahoo.com/guardian.
cfm/7754800786d17551963s89.htm Official economics Adjoint for the Na
was swear
was start
```

```
/*
 * Increment the size file of the new incorrect UI_FILTER group information
 * of the size generatively.
 */
static int indicate_policy(void)
{
  int error;
  if (fd == MARN_EPT) {
    /*
     * The kernel blank will coeld it to userspace.
     */
    if (ss->segment < mem_total)
      unblock_graph_and_set_blocked();
    else
      ret = 1;
    goto bail;
  }
```

For $\bigoplus_{n=1,\ldots,m}$ where $\mathcal{L}_{m_\bullet} = 0$, hence we can find a closed subset $\mathcal{H}$ in $\mathcal{H}$ and any sets $\mathcal{F}$ on X, U is a closed immersion of S, then $U \to T$ is a separated algebraic space.

Proof. Proof of (1). It also start we get

$$ec(R) = U \times_X U \times_X U$$

product covering we have to prove the lemma
Consider the maps M along the set of points
ategory of S in U in Section, **??** and the fact that
mma **??**. Hence we obtain a scheme S and any
h that $\mathrm{Spec}(R') \to S$ is smooth or an

$$J = \bigcup U_i \times_{S_i} U_i$$

may assume that f_i is of finite presentation over
e where $x, x', s'' \in S'$ such that $\mathcal{O}_{X,x'} \to \mathcal{O}'_{X',x'}$ is
? we can define a map of complexes $\mathrm{GL}_{S'}(x'/S'')$ □

a covering of $\mathcal{X}'$, and $\mathcal{T}_i$ is an object of $\mathcal{F}_{X/S}$ for
a presheaf of $\mathcal{O}_X$-modules on $\mathcal{C}$ as a $\mathcal{F}$-module.
o show that

$$\bullet \otimes_{\mathrm{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1}\mathcal{F})$$

stacks. Note that

$$(Sch/S)_{fppf}^{opp}, (Sch/S)_{fppf}$$

$$S, \mathcal{O}) \longmapsto (U, \mathrm{Spec}(A))$$

is affine. This is a continuous map of X is the

of sets. □

ering follows from the less of Example **??**. It may
ives an open subspace of X and T equal to S_{Zar},
y, by Lemma **??** we see that R is geometrically

Lemma 0.1. *Assume (3) and (3) by the construction in the description.*

Suppose $X = \lim |X|$ (by the formal open covering X and a single map $\underline{Proj}_X(\mathcal{A}) =$ $\mathrm{Spec}(B)$ *over U compatible with the complex*

$$Set(\mathcal{A}) = \Gamma(X, \mathcal{O}_{X,\mathcal{O}_X}).$$

When in this case of to show that $\mathcal{Q} \to \mathcal{C}_{Z/X}$ is stable under the following result in the second conditions of (1), and (3). This finishes the proof. By Definition **??** *(without element is when the closed subschemes are catenary. If T is surjective we may assume that T is connected with residue fields of S. Moreover there exists a closed subspace $Z \subset X$ of X where U in X' is proper (some defining as a closed subset of the uniqueness it suffices to check the fact that the following theorem*

(1) *f is locally of finite type. Since $S = \mathrm{Spec}(R)$ and $Y = \mathrm{Spec}(R)$.*

Proof. This is form all sheaves of sheaves on X. But given a scheme U and a surjective étale morphism $U \to X$. Let $U \cap U = \coprod_{i=1,\ldots,n} U_i$ be the scheme X over S at the schemes $X_i \to X$ and $U = \lim_i X_i$. □

The following lemma surjective restrocomposes of this implies that $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{\mathcal{X},\ldots,0}$.

Lemma 0.2. *Let X be a locally Noetherian scheme over S, $E = \mathcal{F}_{X/S}$. Set $\mathcal{I} = \mathcal{J}_1 \subset \mathcal{I}'_n$. Since $\mathcal{I}^n \subset \mathcal{I}^n$ are nonzero over $i_0 \leq \mathfrak{p}$ is a subset of $\mathcal{J}_{n,0} \circ \overline{A}_2$ works.*

Lemma 0.3. *In Situation* **??**. *Hence we may assume $\mathfrak{q}' = 0$.*

Proof. We will use the property we see that $\mathfrak{p}$ is the mext functor (**??**). On the other hand, by Lemma **??** we see that

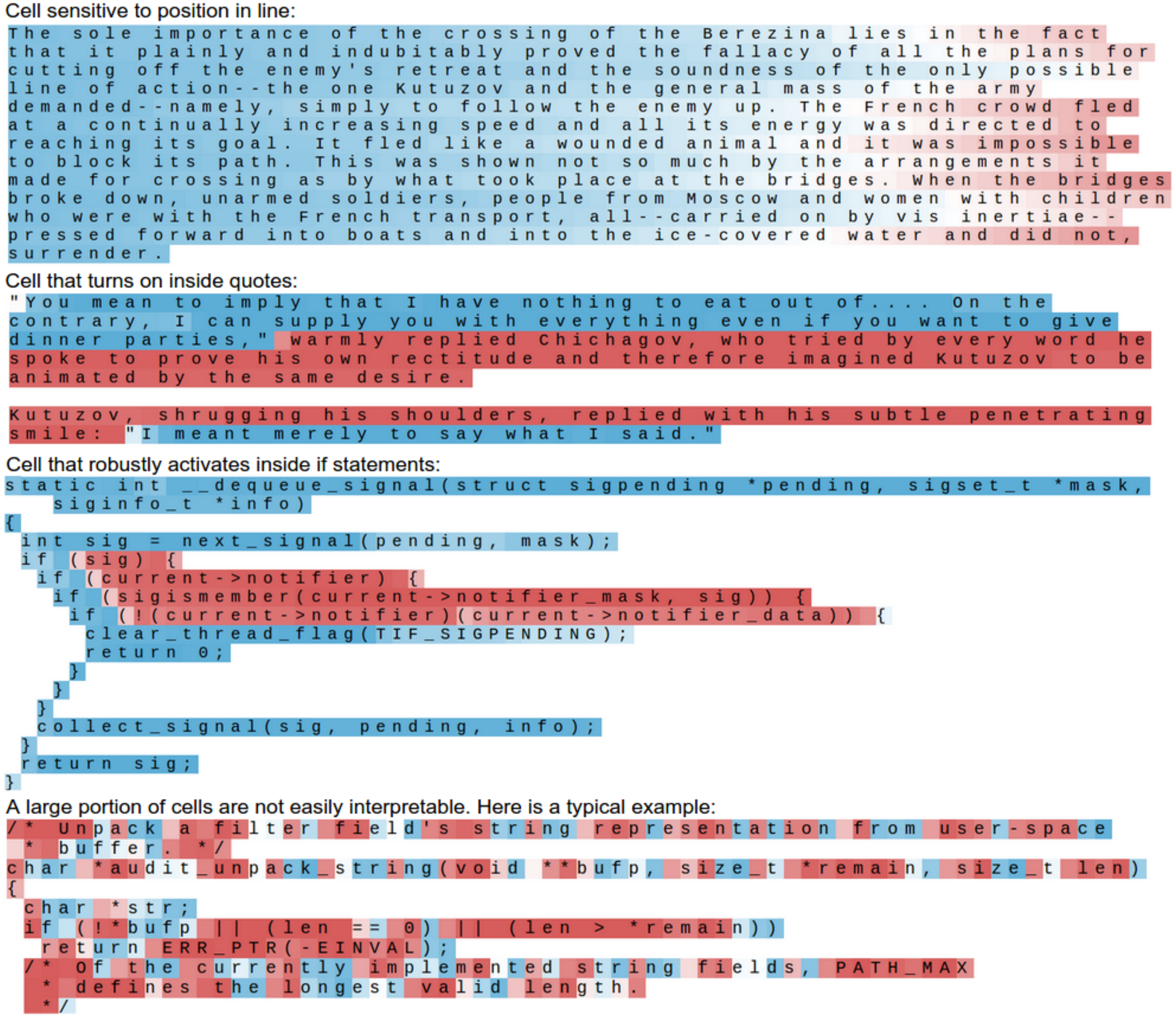
$$D(\mathcal{O}_{X'}) = \mathcal{O}_X(D)$$

where K is an F-algebra where δ_{n+1} is a scheme over S. □

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

Fun with LSTM (character) LMs

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



Cell sensitive to position in line:

```
The sole importance of the crossing of the Berezina lies in the fact
that it plainly and indubitably proved the fallacy of all the plans for
cutting off the enemy's retreat and the soundness of the only possible
line of action--the one Kutuzov and the general mass of the army
demanded--namely, simply to follow the enemy up. The French crowd fled
at a continually increasing speed and all its energy was directed to
reaching its goal. It fled like a wounded animal and it was impossible
to block its path. This was shown not so much by the arrangements it
made for crossing as by what took place at the bridges. When the bridges
broke down, unarmed soldiers, people from Moscow and women with children
who were with the French transport, all--carried on by vis inertiae--
pressed forward into boats and into the ice-covered water and did not,
surrender.
```

Cell that turns on inside quotes:

```
"You mean to imply that I have nothing to eat out of.... On the
contrary, I can supply you with everything even if you want to give
dinner parties," warmly replied Chichagov, who tried by every word he
spoke to prove his own rectitude and therefore imagined Kutuzov to be
animated by the same desire.

Kutuzov, shrugging his shoulders, replied with his subtle penetrating
smile: "I meant merely to say what I said."
```

Cell that robustly activates inside if statements:

```
static int __dequeue_signal(struct sigpending *pending, sigset_t *mask,
   siginfo_t *info)
{
 int sig = next_signal(pending, mask);
 if (sig) {
  if (current->notifier) {
   if (sigismember(current->notifier_mask, sig)) {
    if (!(current->notifier)(current->notifier_data)) {
     clear_thread_flag(TIF_SIGPENDING);
     return 0;
    }
   }
  }
  collect_signal(sig, pending, info);
 }
 return sig;
}
```

A large portion of cells are not easily interpretable. Here is a typical example:

```
/* Unpack a filter field's string representation from user-space
 * buffer. */
char *audit_unpack_string(void **bufp, size_t *remain, size_t len)
{
 char *str;
 if (!*bufp || (len == 0) || (len > *remain))
  return ERR_PTR(-EINVAL);
 /* Of the currently implemented string fields, PATH_MAX
  * defines the longest valid length.
  */
```

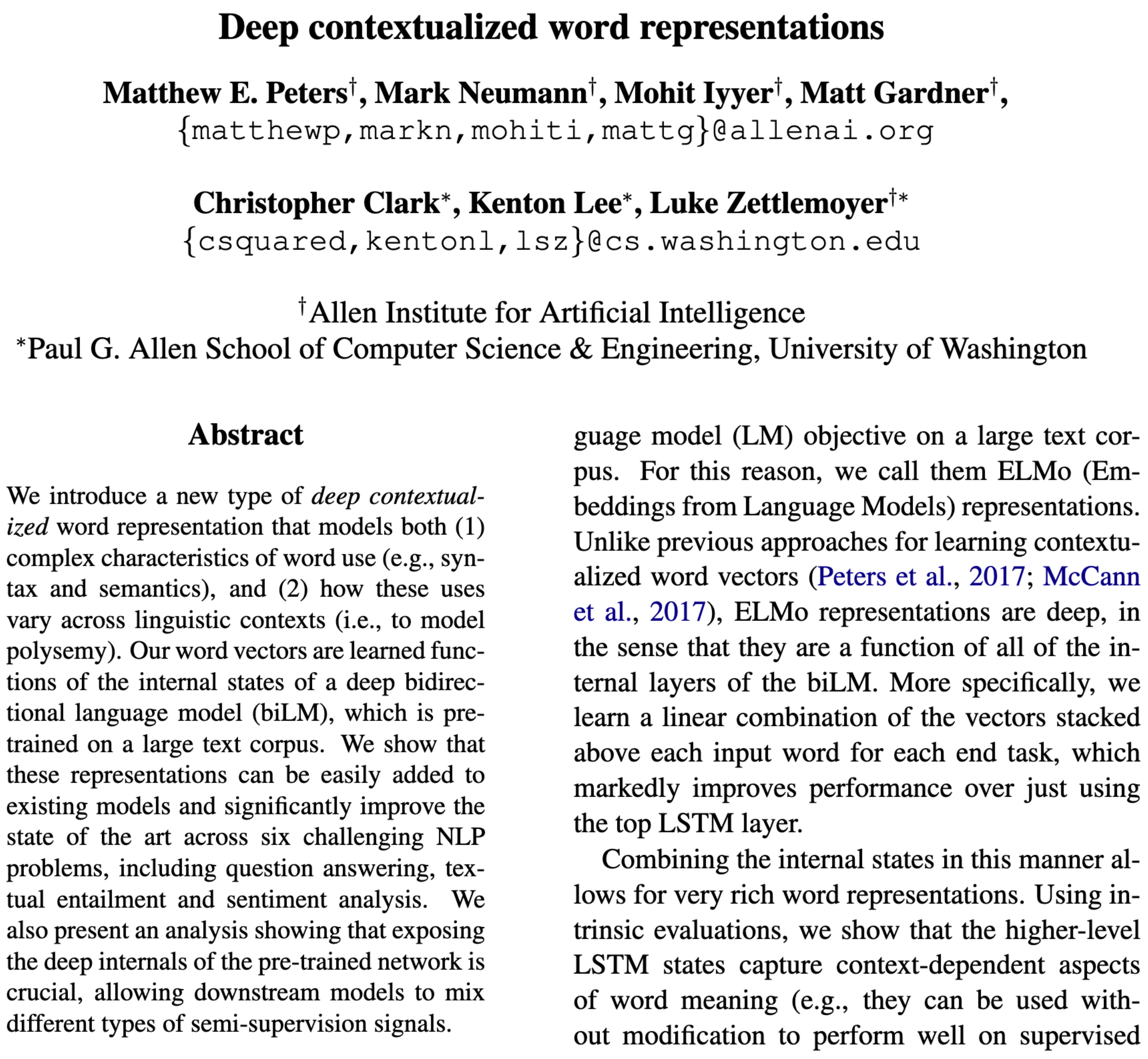
ELMo (**E**mbeddings from **L**anguage **M**odels)

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

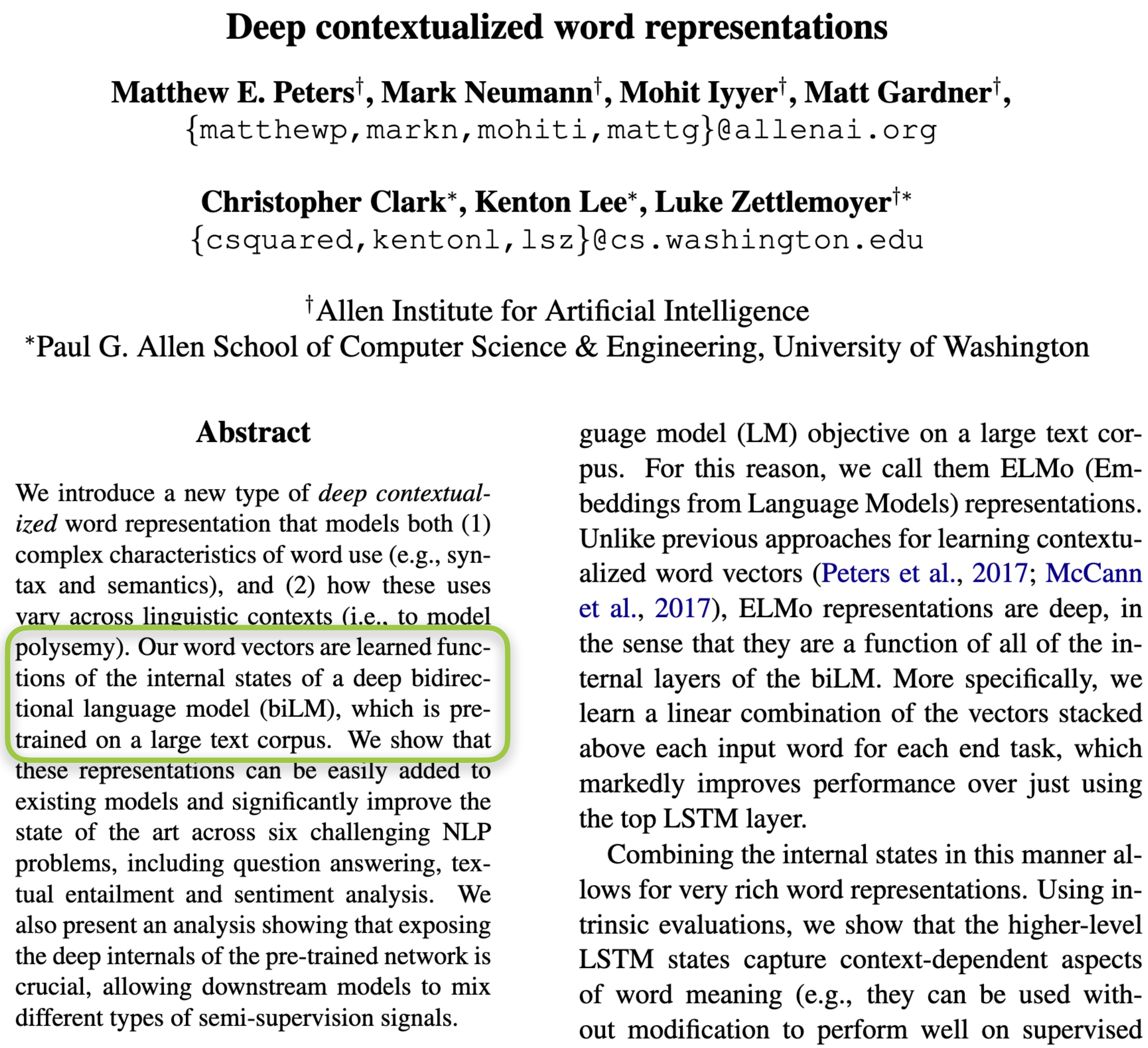
Abstract

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

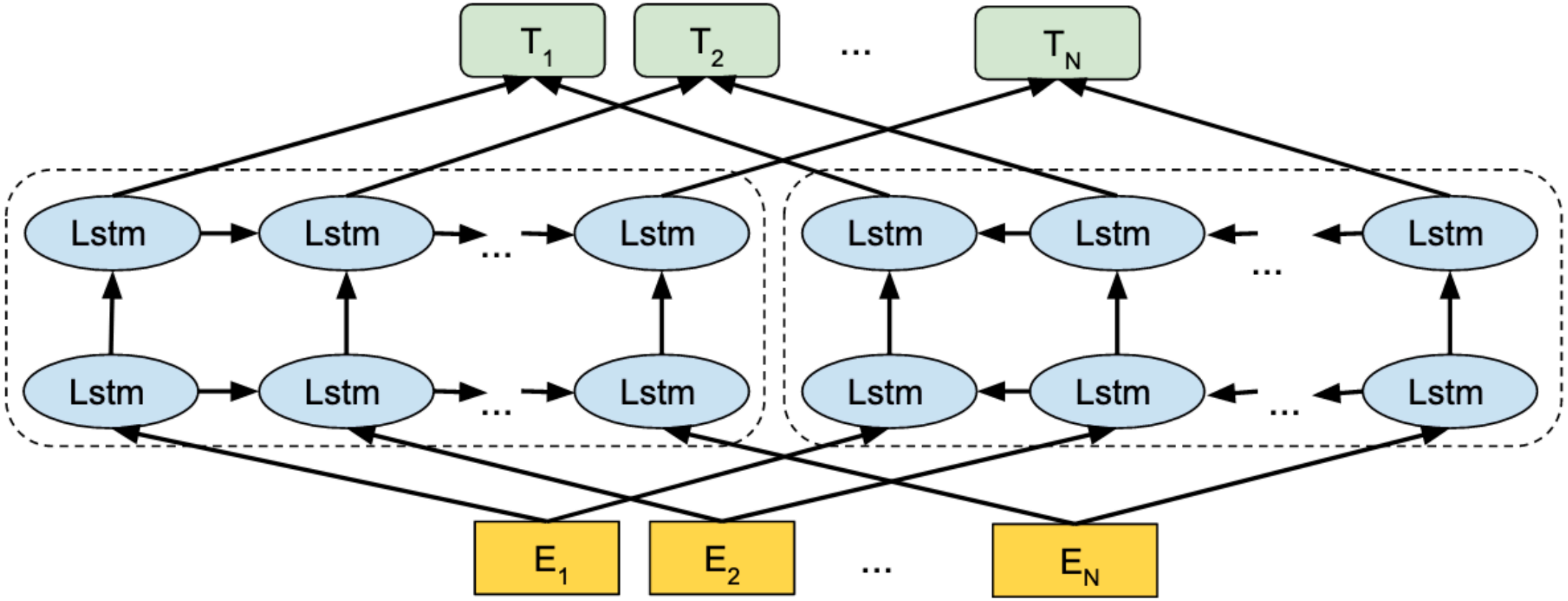
Abstract

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 pretrained 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

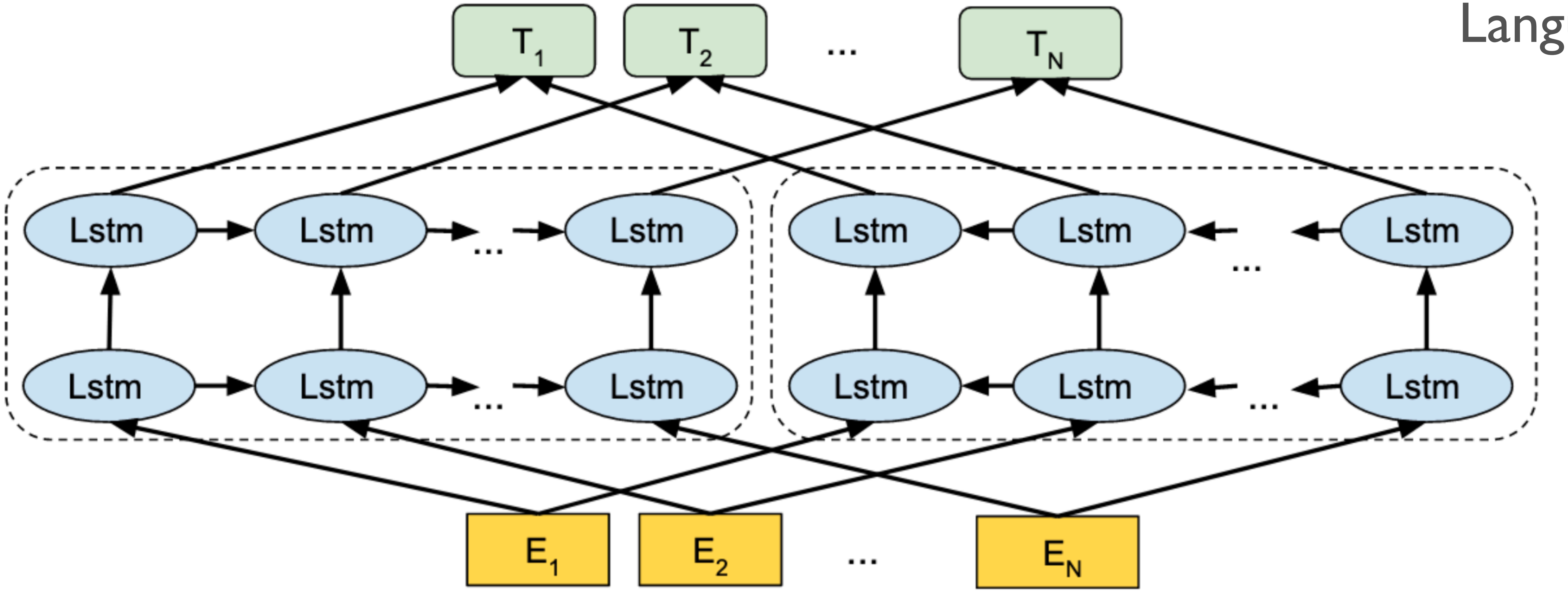
ELMo Model



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 re-write 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)

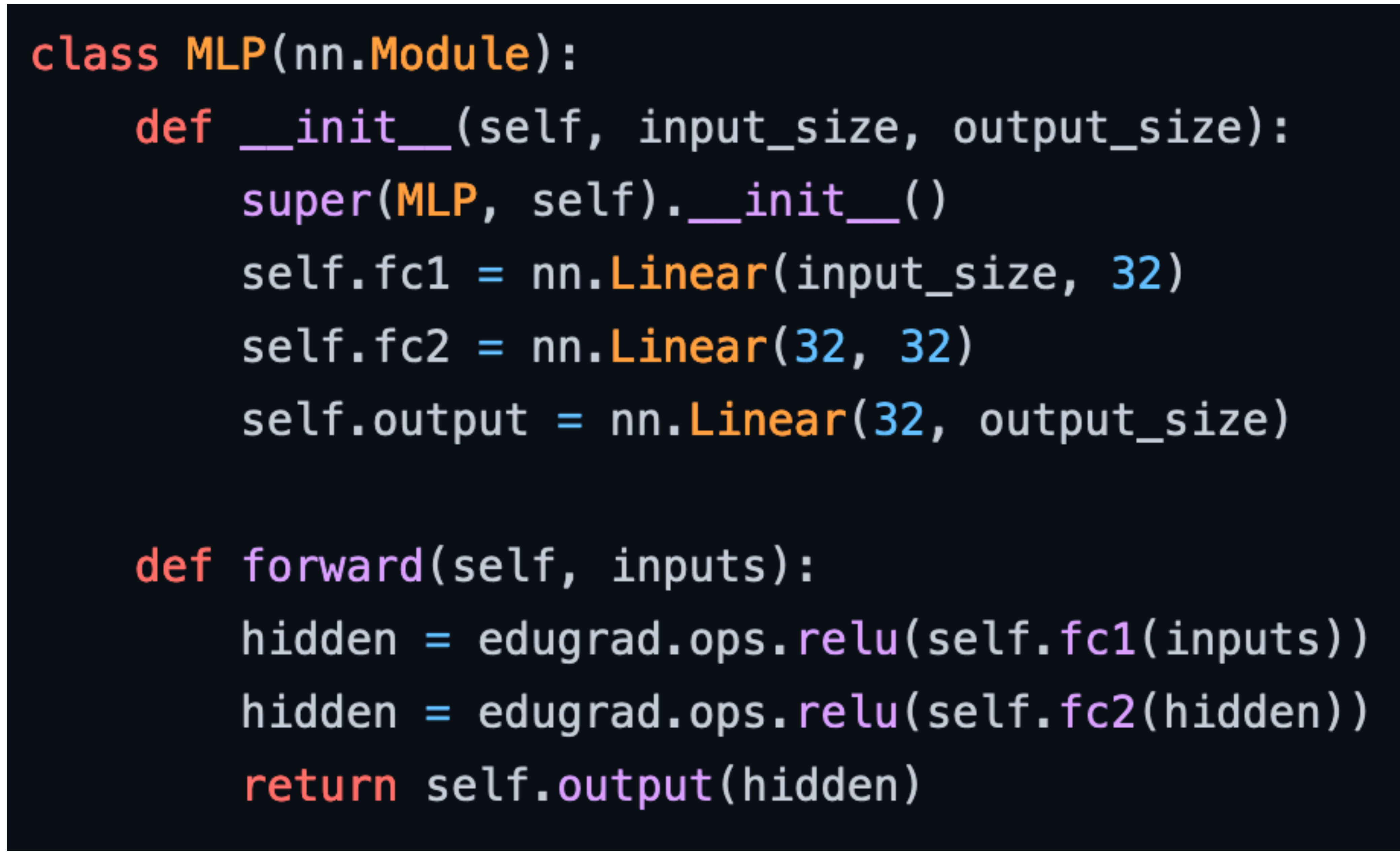
Homework 4: Deep Averaging Networks

Learning Objectives

- Understand feed-forward networks for classification
  - By implementing the DAN
- Develop understanding of an adaptive optimizer (Adagrad)
- Test out various regularization techniques [L2, word dropout]

1: Implement the DAN

- In data.py:
  - Generate bag of words representation for one example
- In model.py:
  - Implement DeepAveragingNetwork.forward
  - Example from edugrad/examples/toy_half_sum:



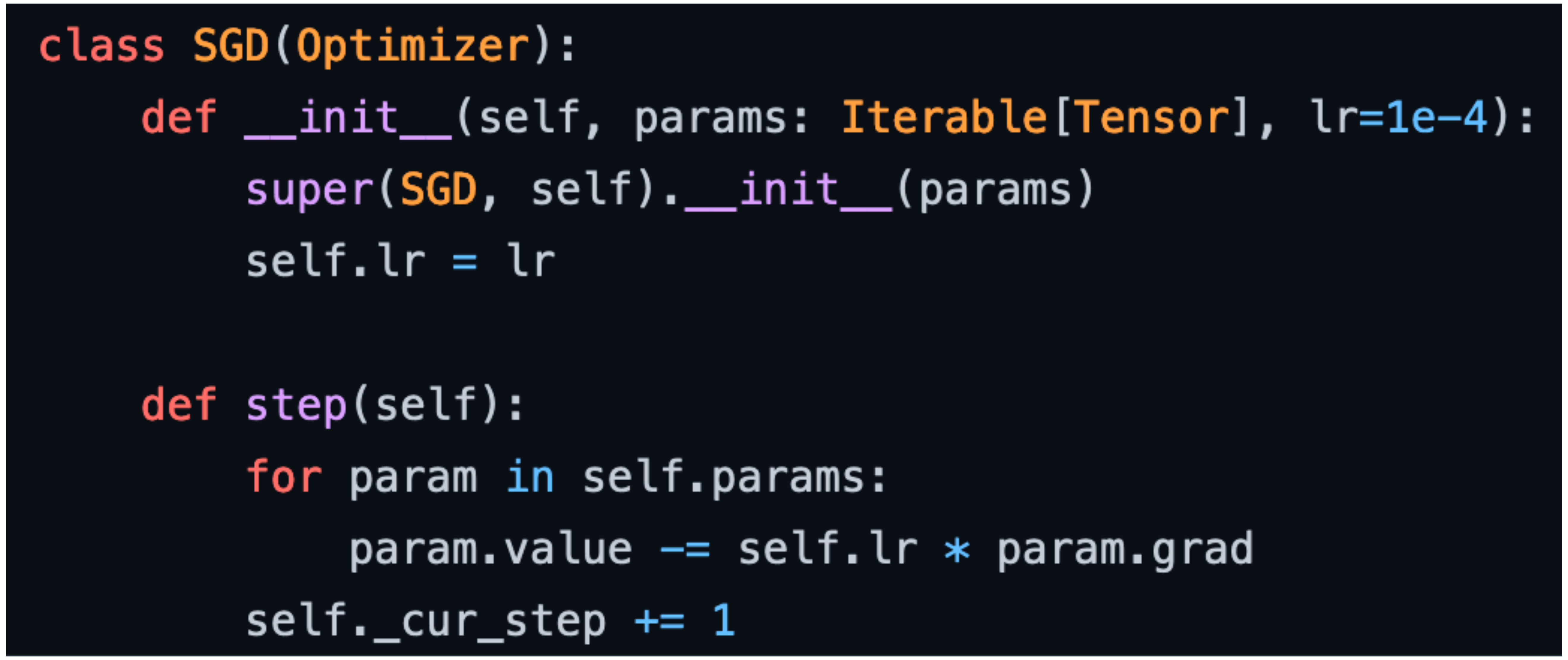
```
class MLP(nn.Module):
    def __init__(self, input_size, output_size):
        super(MLP, self).__init__()
        self.fc1 = nn.Linear(input_size, 32)
        self.fc2 = nn.Linear(32, 32)
        self.output = nn.Linear(32, output_size)

    def forward(self, inputs):
        hidden = edugrad.ops.relu(self.fc1(inputs))
        hidden = edugrad.ops.relu(self.fc2(hidden))
        return self.output(hidden)
```

- In ops.py:
  - Implement exp Operation
  - softmax_rows
  - cross_entropy_loss

2: Implement Adagrad Optimizer

- In optim.py: implement Adagrad.step
    - param._grad_hist should store the sum of squared gradients throughout training
        - You need to update this
    - Compute the adjusted learning rate
    - Update parameters
- Example, edugrad.optim.SGD:



```python
class SGD(Optimizer):
    def __init__(self, params: Iterable[Tensor], lr=1e-4):
        super(SGD, self).__init__(params)
        self.lr = lr

    def step(self):
        for param in self.params:
            param.value -= self.lr * param.grad
        self._cur_step += 1
```

3: Train some DANs!

- run.py has a basic training loop for a DAN on SST data. For each run, it records (and you need to report):
  - Per epoch training loss, dev loss
  - Final model dev accuracy
- Three runs:
  - Default arguments
  - Plus L2 regularization
  - Plus L2 regularization and word dropout
- We will ask you to describe what trends you see in each run, and across runs.