Language Models

LING575 Analyzing Neural Language Models
Shane Steinert-Threlkeld
April 6 2022

Outline

- Background
- Recurrent Neural Networks (LSTMs in particular)
  - ELMo
  - seq2seq + *attention*
- Transformers
  - BERT
- Snapshot of the current landscape

Reminders

- Group formation due tonight
  - Canvas discussion thread for people looking for a group
  - Enter groups in the Google Doc linked from hw1 page

Some Fun with CLIP

- One vision-and-language model [as asked about last time]: https://openai.com/blog/clip/
  - Text-based adversarial attacks:

Granny Smith	85.6%
iPod	0.4%
library	0.0%
pizza	0.0%
toaster	0.0%
dough	0.1%

Granny Smith	0.1%
iPod	99.7%
library	0.0%
pizza	0.0%
toaster	0.0%
dough	0.0%

Some Fun with CLIP

- One vision-and-language model [as asked about last time]: https://openai.com/blog/clip/
  - [Text-based adversarial attacks](#):

Some Fun with CLIP

- https://janellecshane.substack.com/p/sea-shanty-surrealism
- Follow-up model with better caption —> image direction

Recap

- Transfer learning: pre-train on one task, ‘transfer’ to new task
- For NLP: *language modeling* [unannotated data]
- Current state-of-the-art involves very large-scale pre-training
- To understand what such models learn, we need to know a bit about what they are and how they build representations

What is a language model?

- A language model parametrized by θ computes $P_\theta(w_1, \ldots, w_n)$

- Typically: $P_\theta(w_1, \ldots, w_n) = \prod_i P_\theta(w_i \mid w_1, \ldots, w_{i-1})$

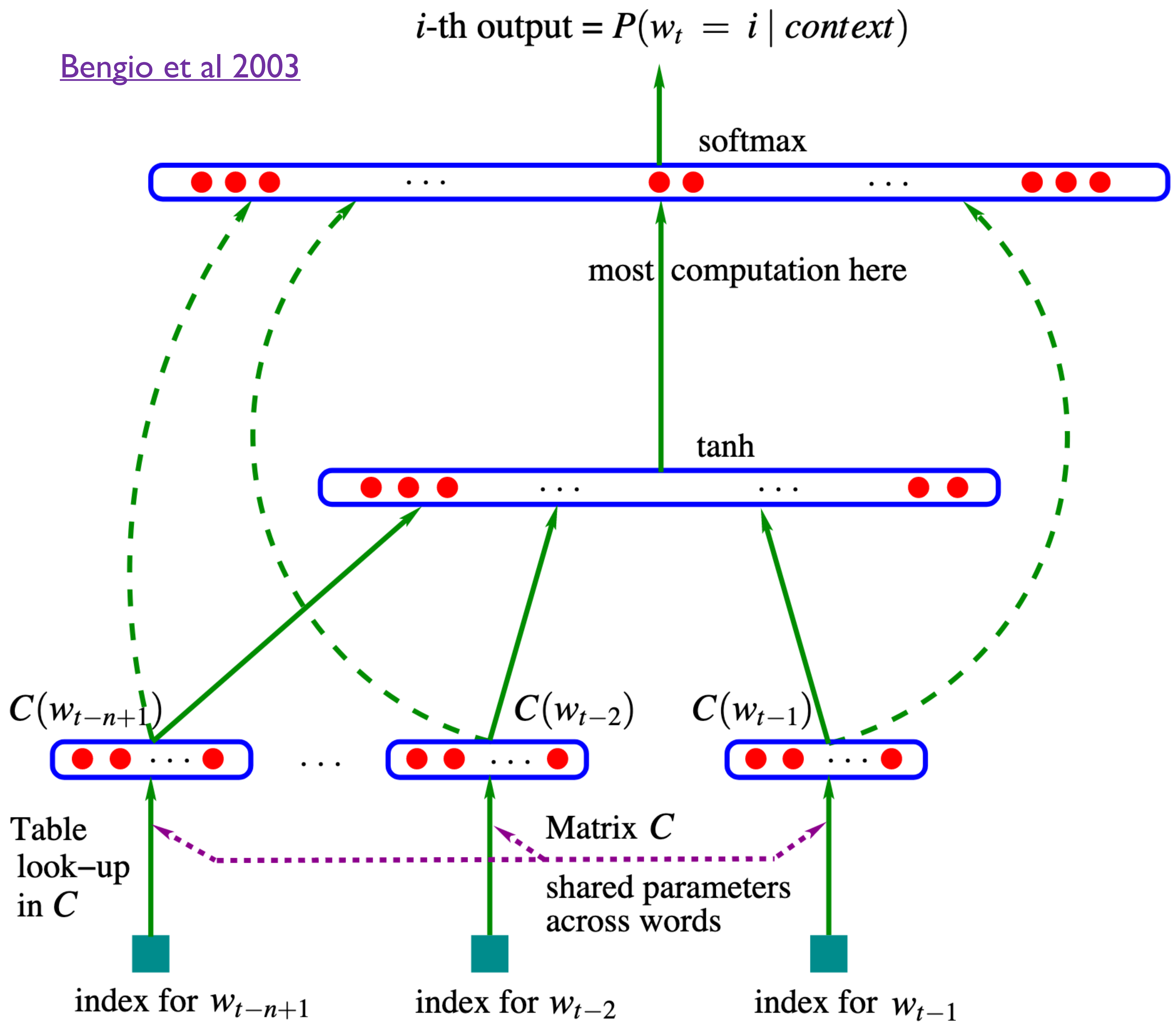
- E.g. of labeled data: “Today is the first day of 575.” —>
  - (<s>, Today)
  - (<s> Today, is)
  - (<s> Today is, the)
  - (<s> Today is the, first)

Parameters of Variation

- Model architecture:
  - Feed-forward, Recurrent (w/ sub-types), Transformer-based
  - # parameters, #FLOPS per forward / backward pass
- Tokenization + token representation
- Pre-training variant:
  - Pure LM
  - Masked LM (plus …)
  - Replaced token detection
  - Denoising auto-encoding
  - …
- Training procedure
  - data source, size, shuffled at any level?, …
  - Multilingual / monolingual?
- Often hard to make direct comparisons! (Though see [Clark et al 2020](#))

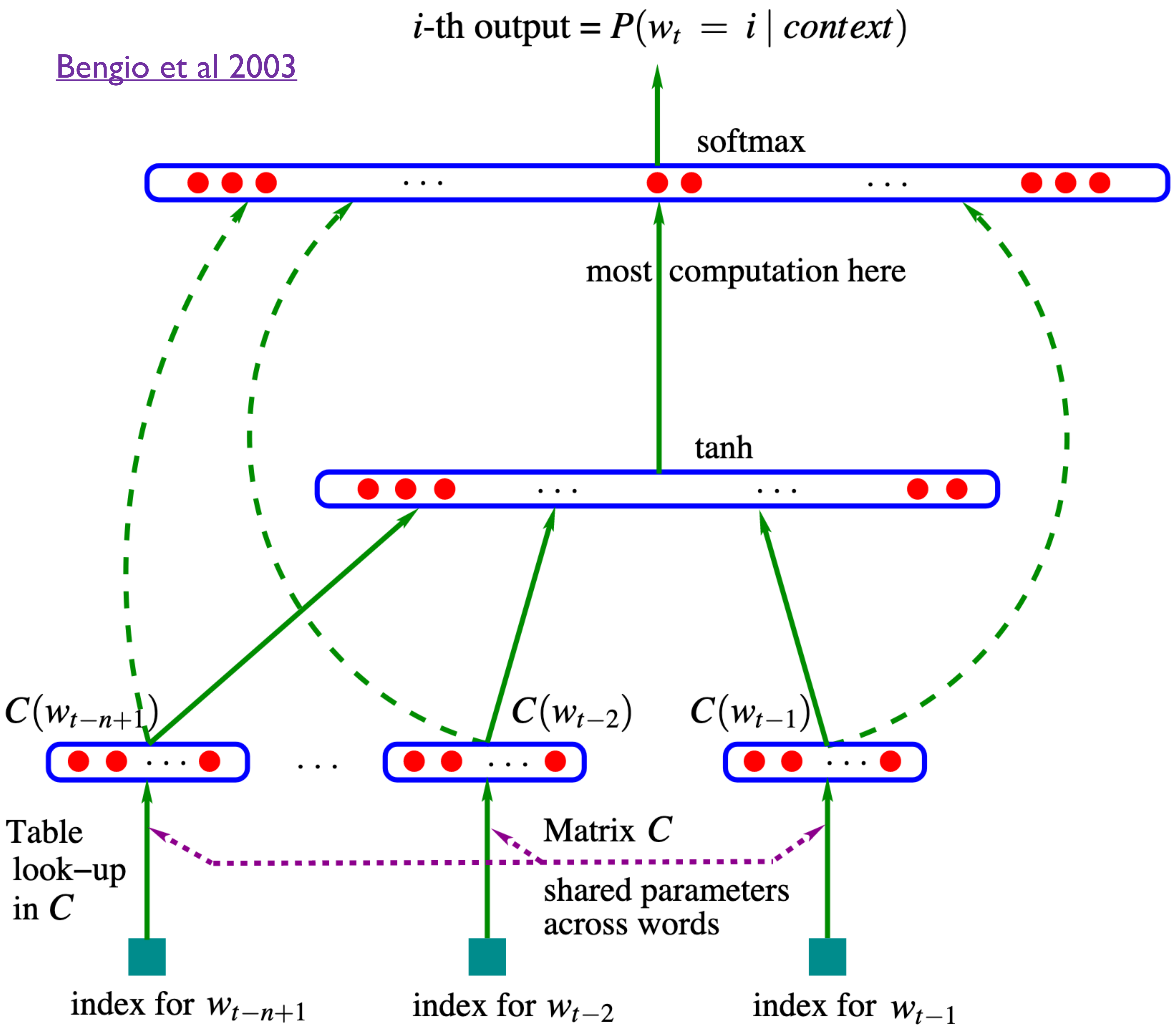
The earliest (?) neural LM

Bengio et al 2003



The earliest (?) neural LM

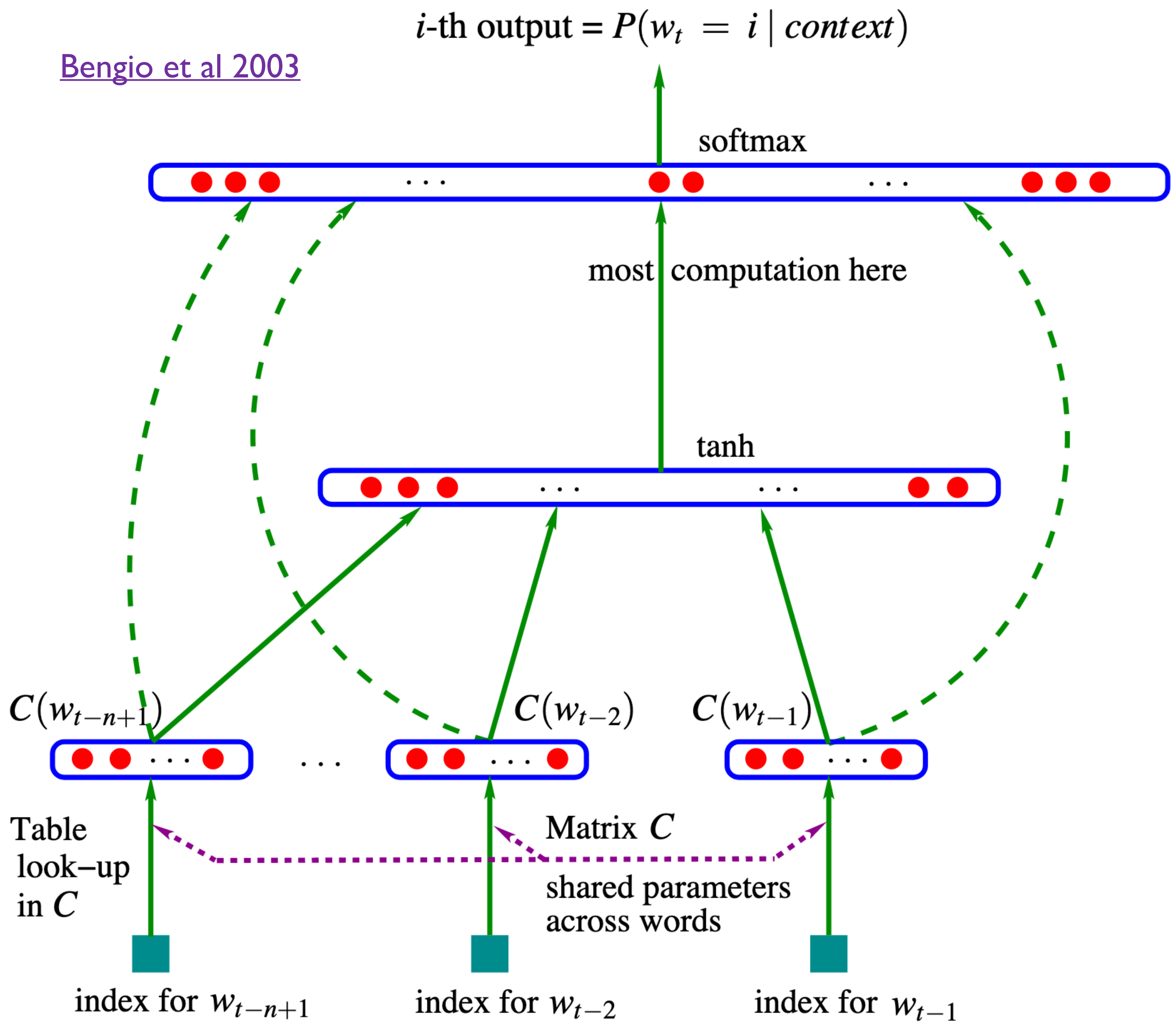
Bengio et al 2003



w_t: one-hot vector

The earliest (?) neural LM

Bengio et al 2003

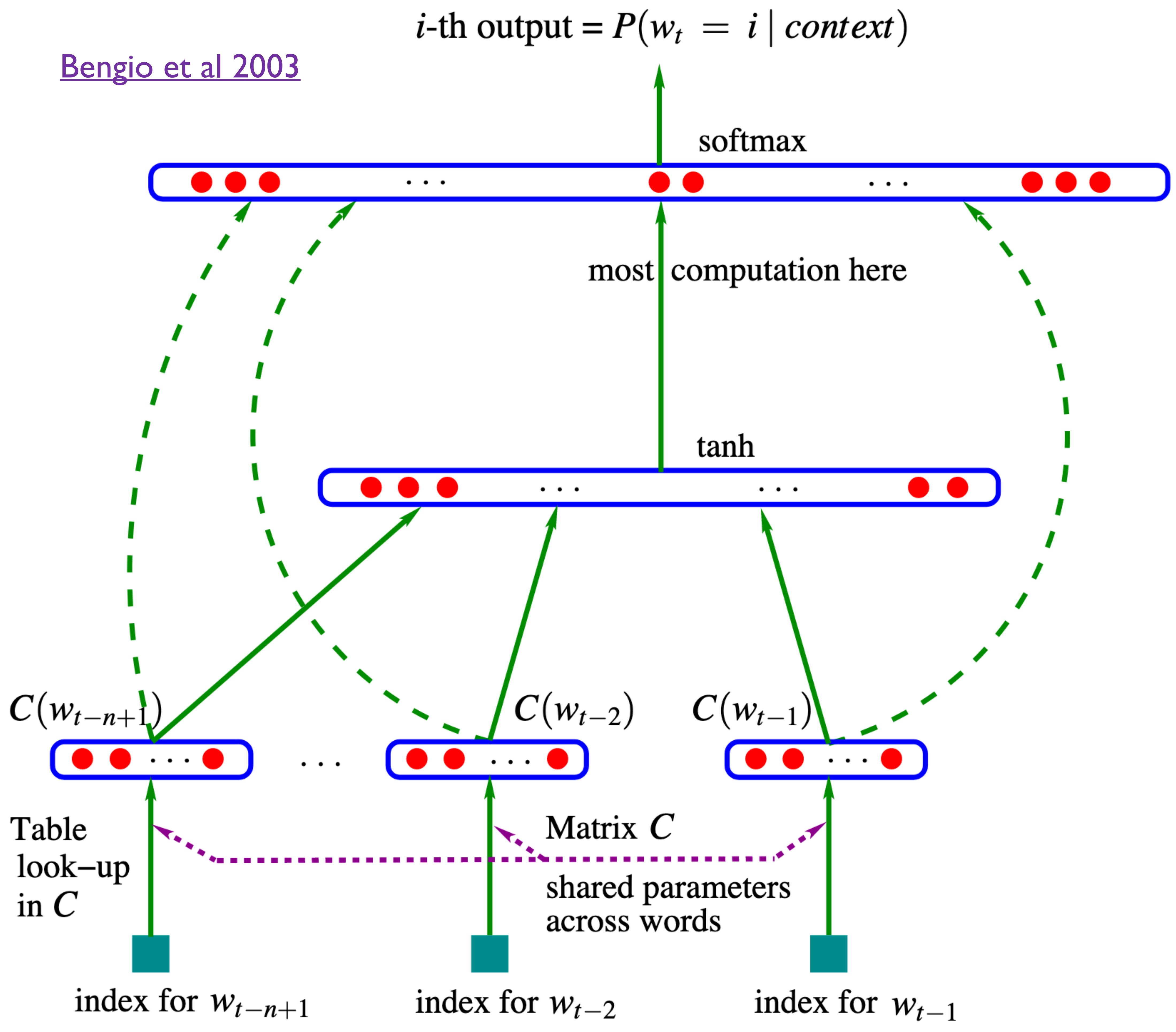


$$\text{embeddings} = \text{concat}(Cw_{t-1}, Cw_{t-2}, \ldots, Cw_{t-(n+1)})$$

w_t: one-hot vector

The earliest (?) neural LM

Bengio et al 2003



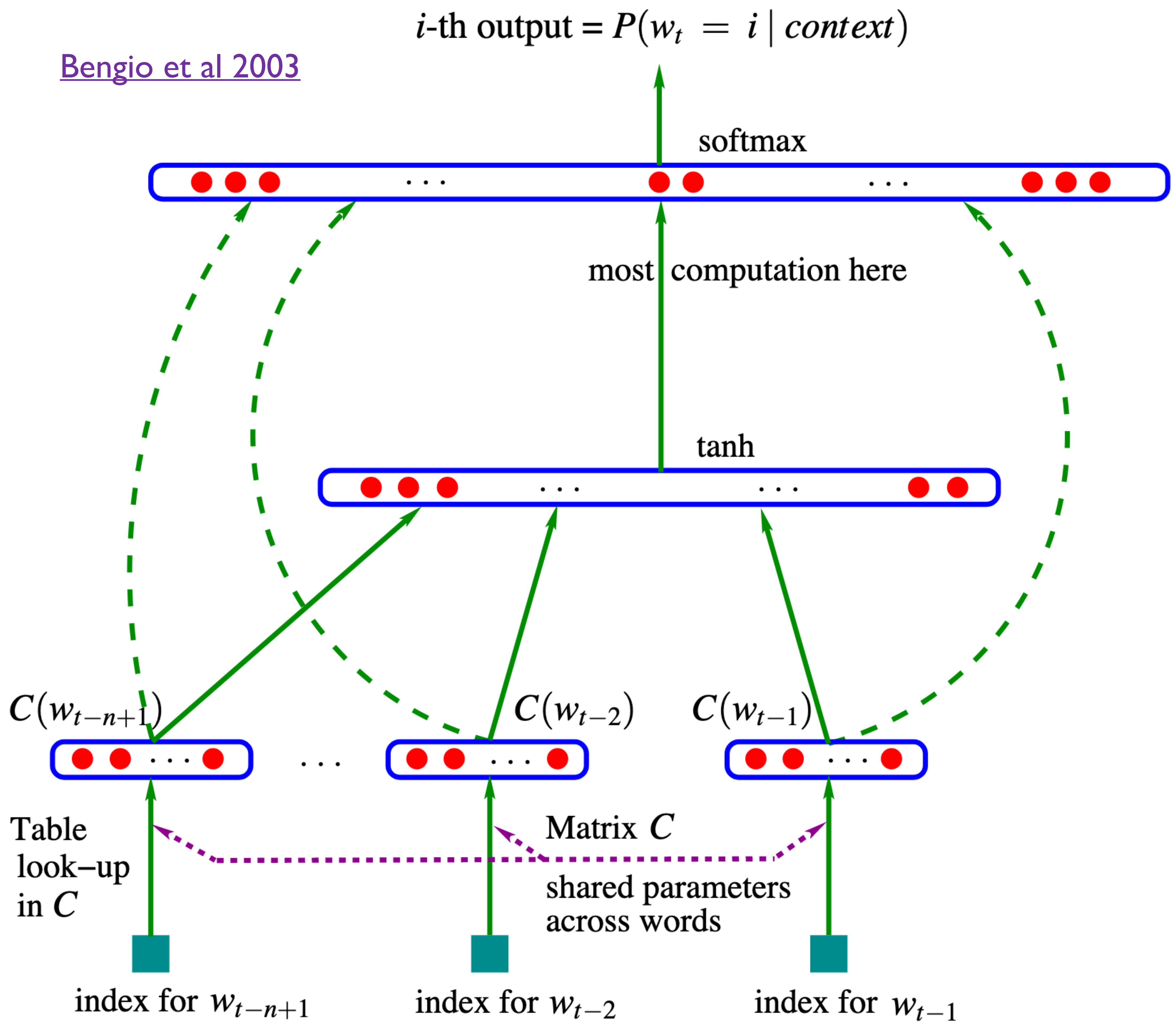
$$\text{hidden} = \tanh(W_1 \text{embeddings} + b_1)$$

$$\text{embeddings} = \text{concat}(Cw_{t-1}, Cw_{t-2}, \ldots, Cw_{t-(n+1)})$$

w_t: one-hot vector

The earliest (?) neural LM

Bengio et al 2003



probabilities = $\mathrm{softmax}(W_2 \mathrm{hidden} + b_2)$

hidden = $\tanh(W_1 \mathrm{embeddings} + b_1)$

embeddings = $\mathrm{concat}(Cw_{t-1}, Cw_{t-2}, \ldots, Cw_{t-(n+1)})$

w_t: one-hot vector

Some (but not all) details

Some (but not all) details

- Loss (the standard one): *cross-entropy*. In the classification/LM case:

$$L(\theta) = \frac{1}{T}\sum_{i=1}^{T} -\log \mathrm{probabilities}(w_i)$$

Some (but not all) details

- Loss (the standard one): *cross-entropy*. In the classification/LM case:

$$L(\theta) = \frac{1}{T}\sum_{i=1}^{T} -\log \mathrm{probabilities}(w_i)$$

Some (but not all) details

- Loss (the standard one): *cross-entropy*. In the classification/LM case:

$$L(\theta) = \frac{1}{T}\sum_{i=1}^{T} - \log \mathsf{probabilities}(w_i)$$

Some (but not all) details

- Loss (the standard one): *cross-entropy*. In the classification/LM case:

$$L(\theta) = \frac{1}{T}\sum_{i=1}^{T} -\log \text{probabilities}(w_i)$$

- Training data: Brown corpus (~1M tokens; |V| approx 14.5k after removing rare words), and AP news (~14M tokens; |V| approx 18k)

Some (but not all) details

- Loss (the standard one): *cross-entropy*. In the classification/LM case:

$$L(\theta) = \frac{1}{T}\sum_{i=1}^{T} - \log \text{probabilities}(w_i)$$

- Training data: Brown corpus (~1M tokens; |V| approx 14.5k after removing rare words), and AP news (~14M tokens; |V| approx 18k)

Some (but not all) details

- Loss (the standard one): *cross-entropy*. In the classification/LM case:

$$L(\theta) = \frac{1}{T}\sum_{i=1}^{T} - \log \text{probabilities}(w_i)$$

- Training data: Brown corpus (~1M tokens; |V| approx 14.5k after removing rare words), and AP news (~14M tokens; |V| approx 18k)

- **Primary result:** NNLM significantly better test-set perplexity than most sophisticated n-gram LMs

Outline

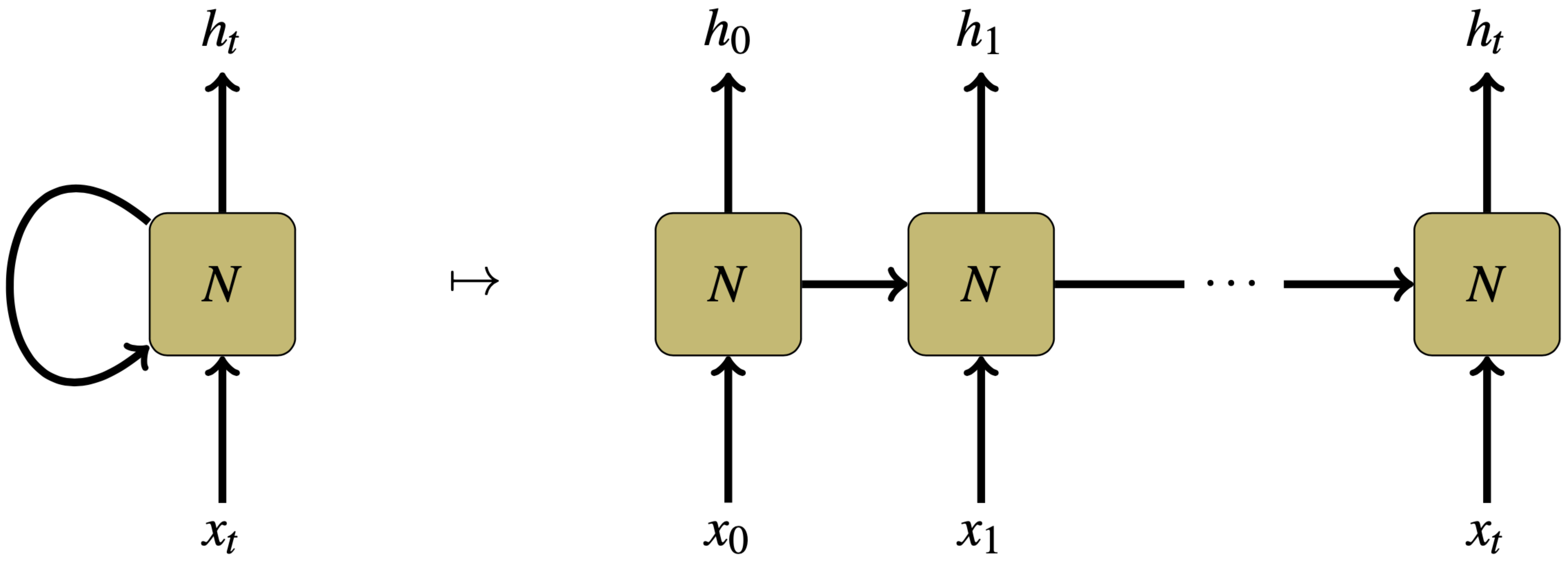
- Background
- **Recurrent Neural Networks (LSTMs in particular)**
  - ELMo
  - seq2seq + *attention*
- Transformers
  - BERT
- Snapshot of the current landscape

Recurrent Neural Networks

RNNs: high-level

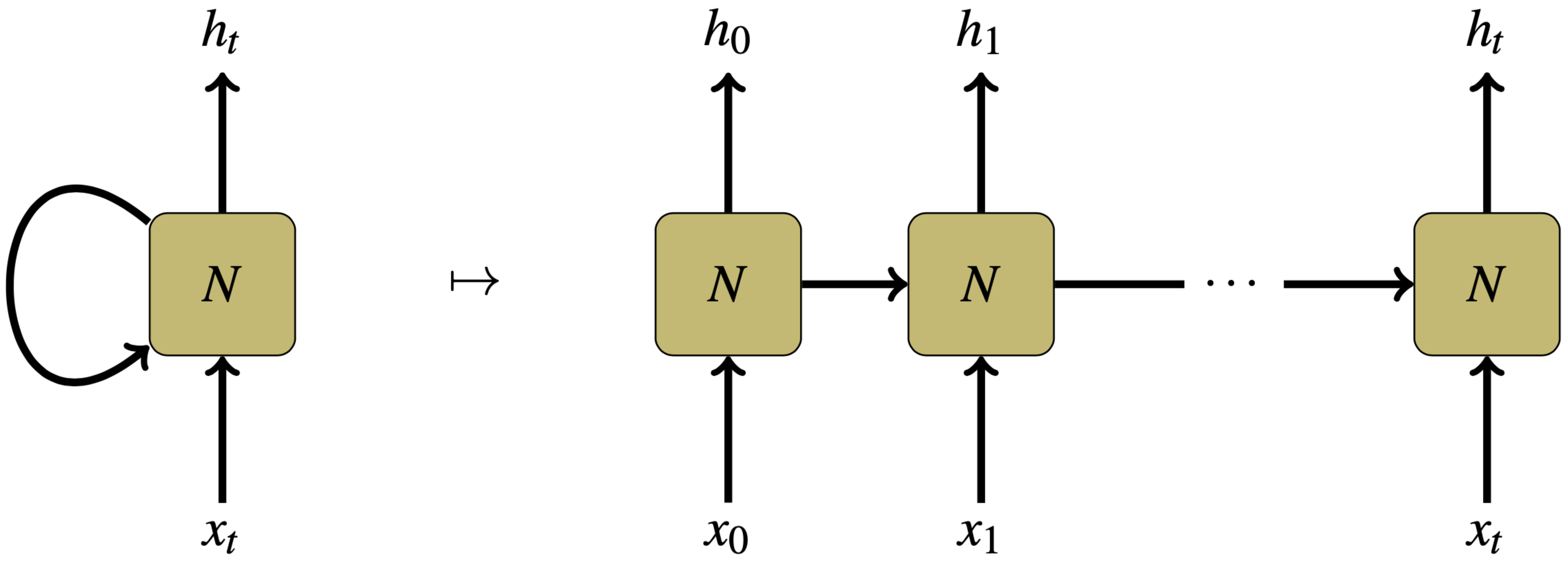
- Feed-forward networks: fixed-size input, fixed-size output
  - Previous LM: fixed sized window of previous words
- RNNs process *sequences* of vectors
  - Maintaining “hidden” state
  - Applying the same operation at each step

RNNs



Steinert-Threlkeld and Szymanik 2019; Olah 2015

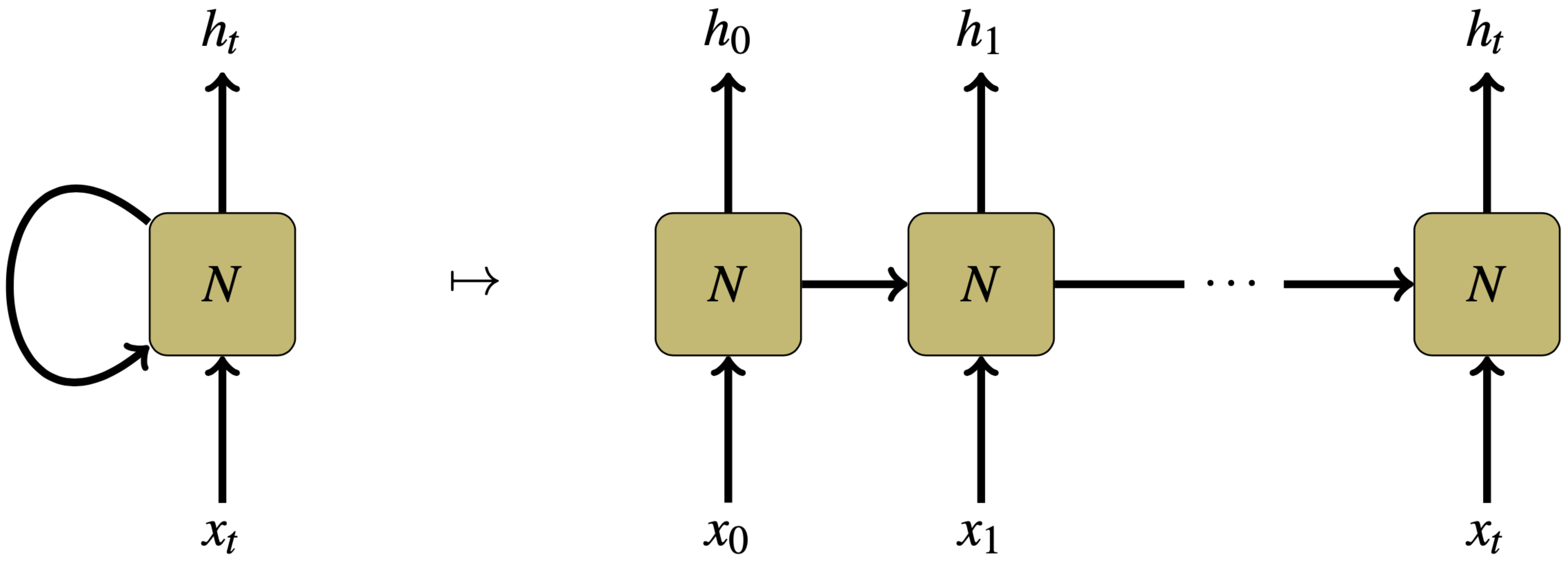
RNNs



$$h_t = f(x_t, h_{t-1})$$

Steinert-Threlkeld and Szymanik 2019; Olah 2015

RNNs

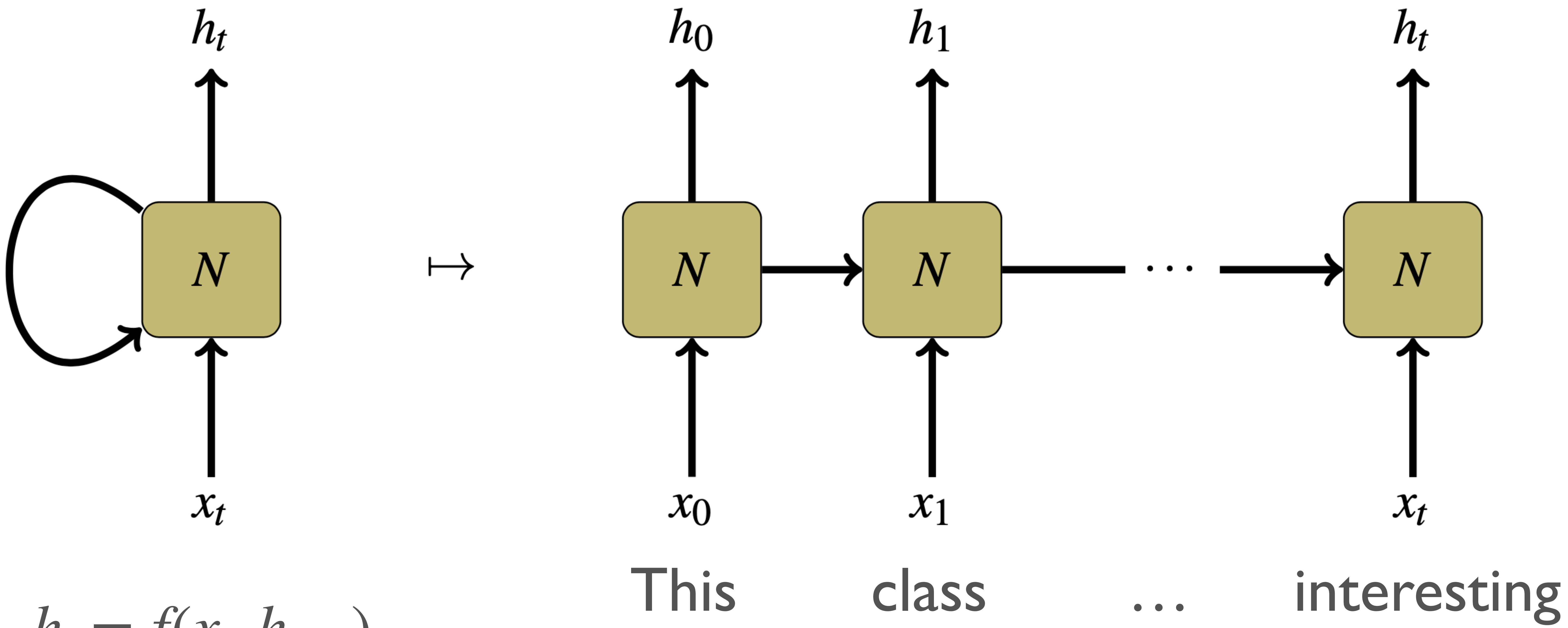


$$h_t = f(x_t, h_{t-1})$$

Simple/"Vanilla" RNN: $h_t = \tanh(W_x x_t + W_h h_{t-1} + b)$

Steinert-Threlkeld and Szymanik 2019; Olah 2015

RNNs

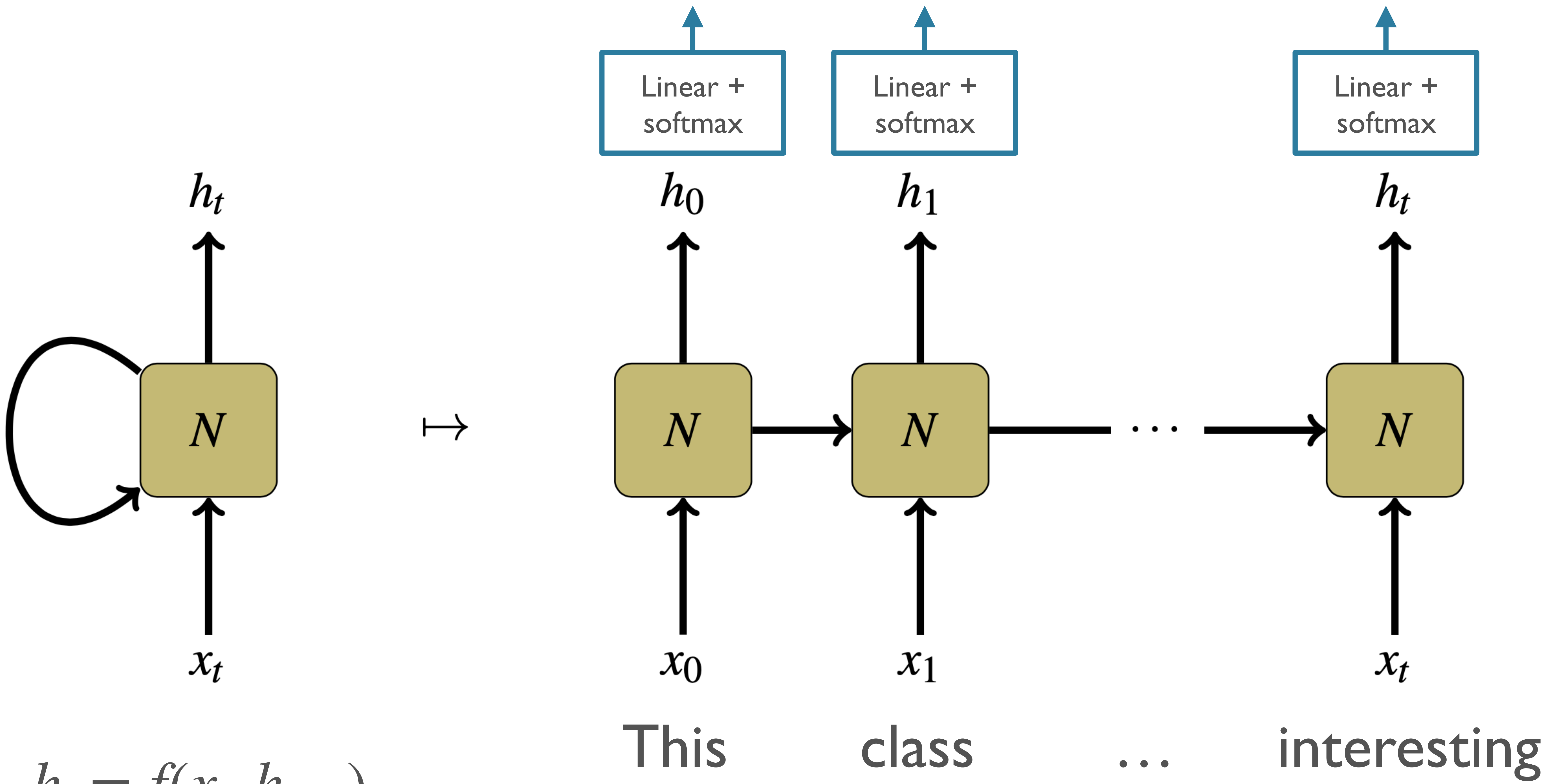


$$h_t = f(x_t, h_{t-1})$$

Simple/“Vanilla” RNN: $h_t = \tanh(W_x x_t + W_h h_{t-1} + b)$

Steinert-Threlkeld and Szymanik 2019; Olah 2015

RNNs



$$h_t = f(x_t, h_{t-1})$$

Simple/"Vanilla" RNN: $h_t = \tanh(W_x x_t + W_h h_{t-1} + b)$

Steinert-Threlkeld and Szymanik 2019; Olah 2015

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

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

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

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

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

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

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

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

🤔🤔🤷‍♀️🤮

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

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

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

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

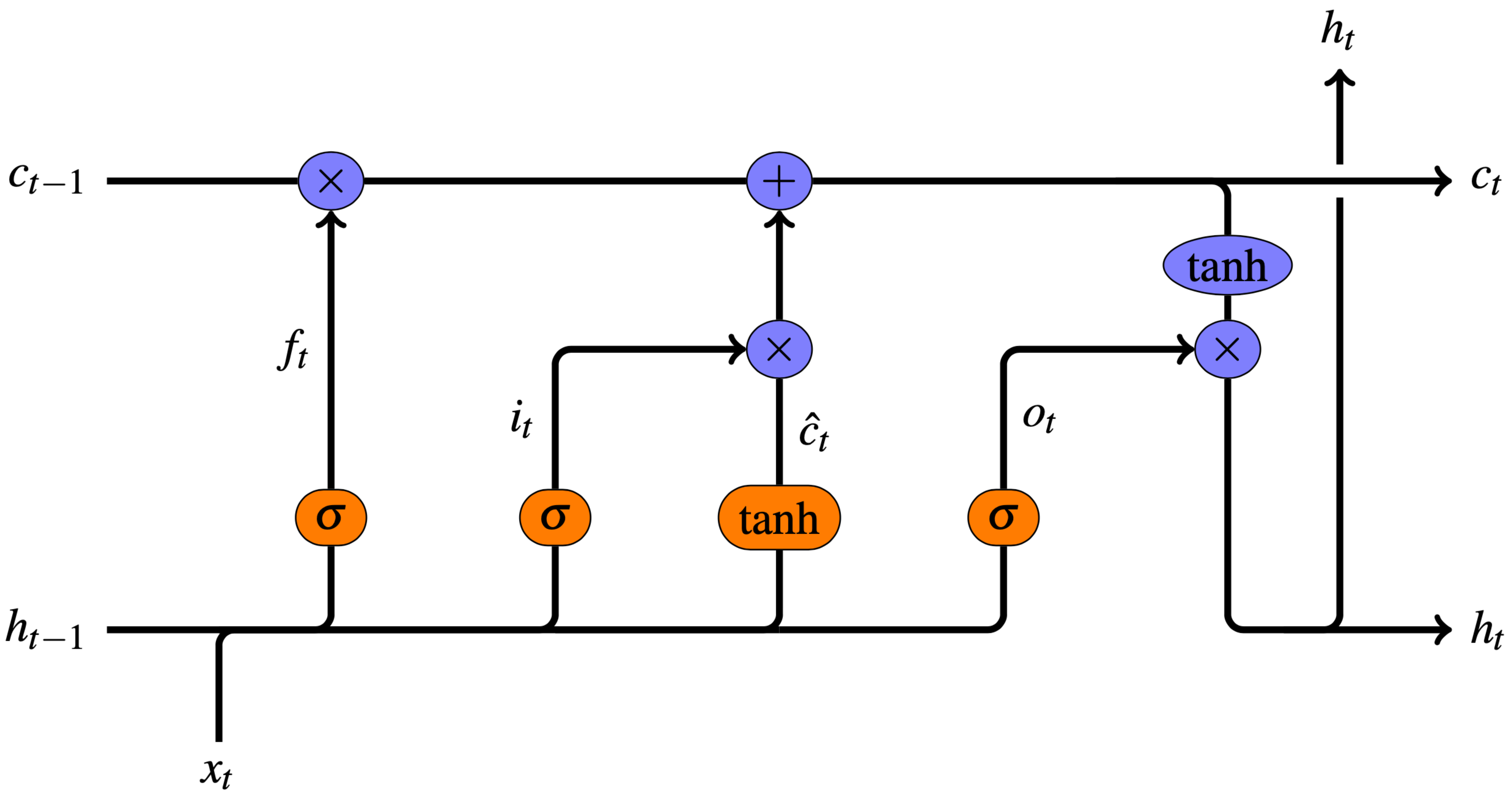
$$c_t = f_t \odot c_{t-1} + i_t \odot \hat{c}_t$$

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

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

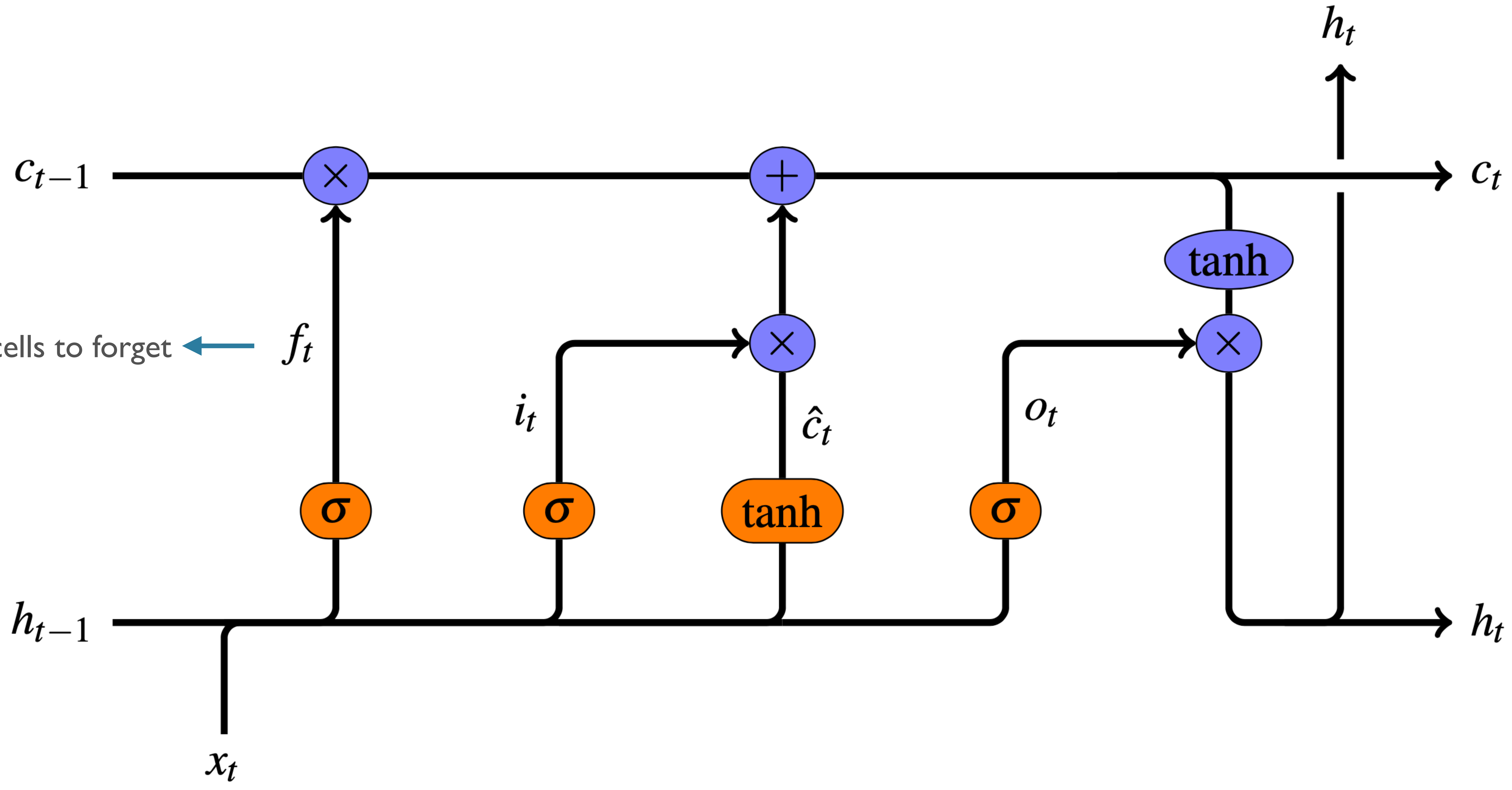
🤔🤔🤷‍♀️🤮

LSTMs

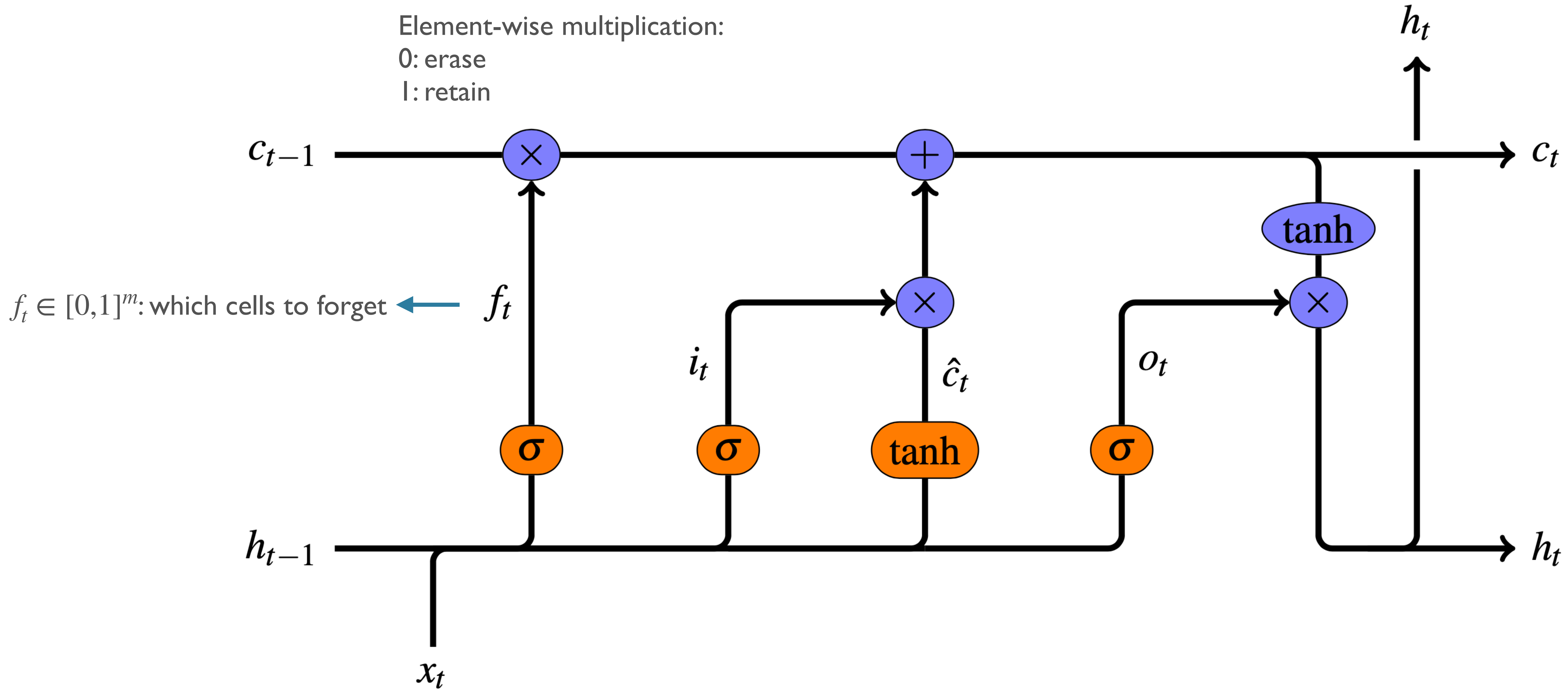


LSTMs

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

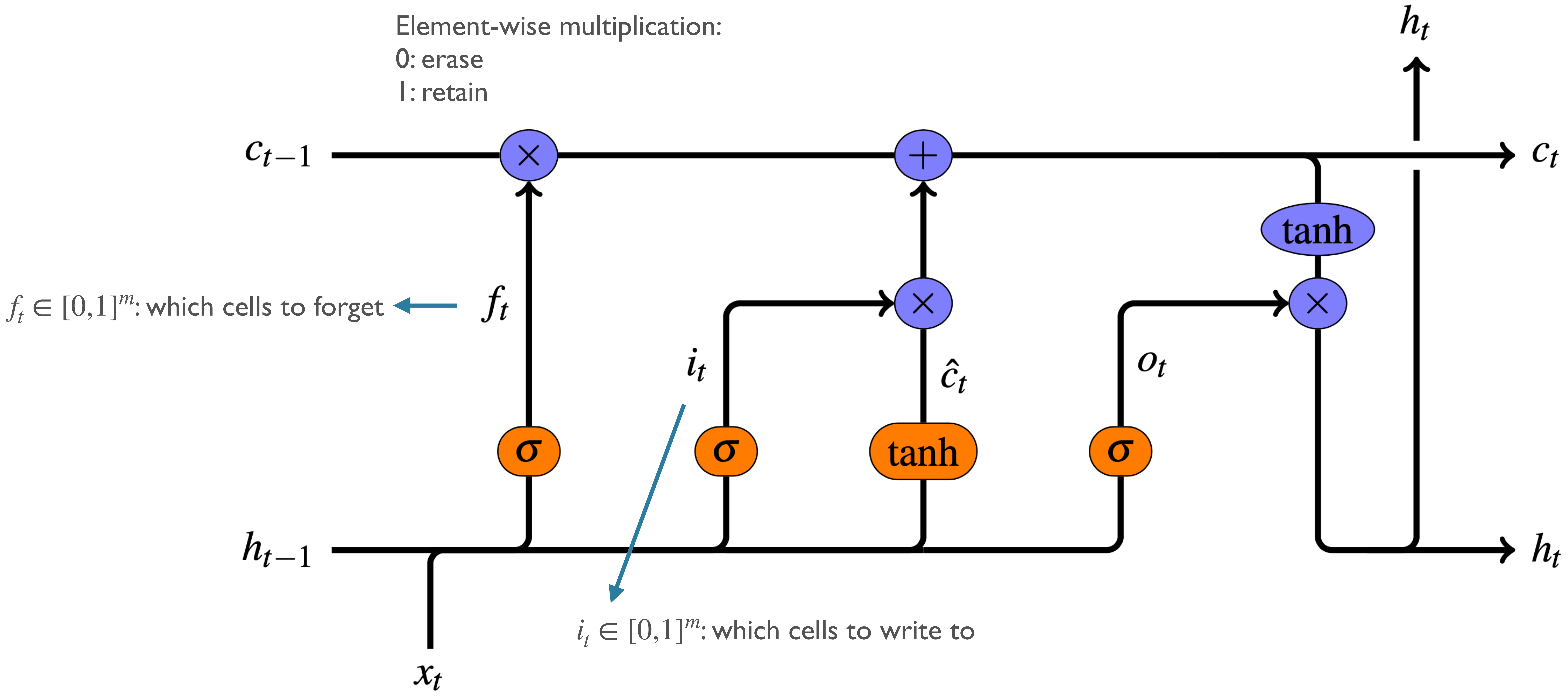


LSTMs



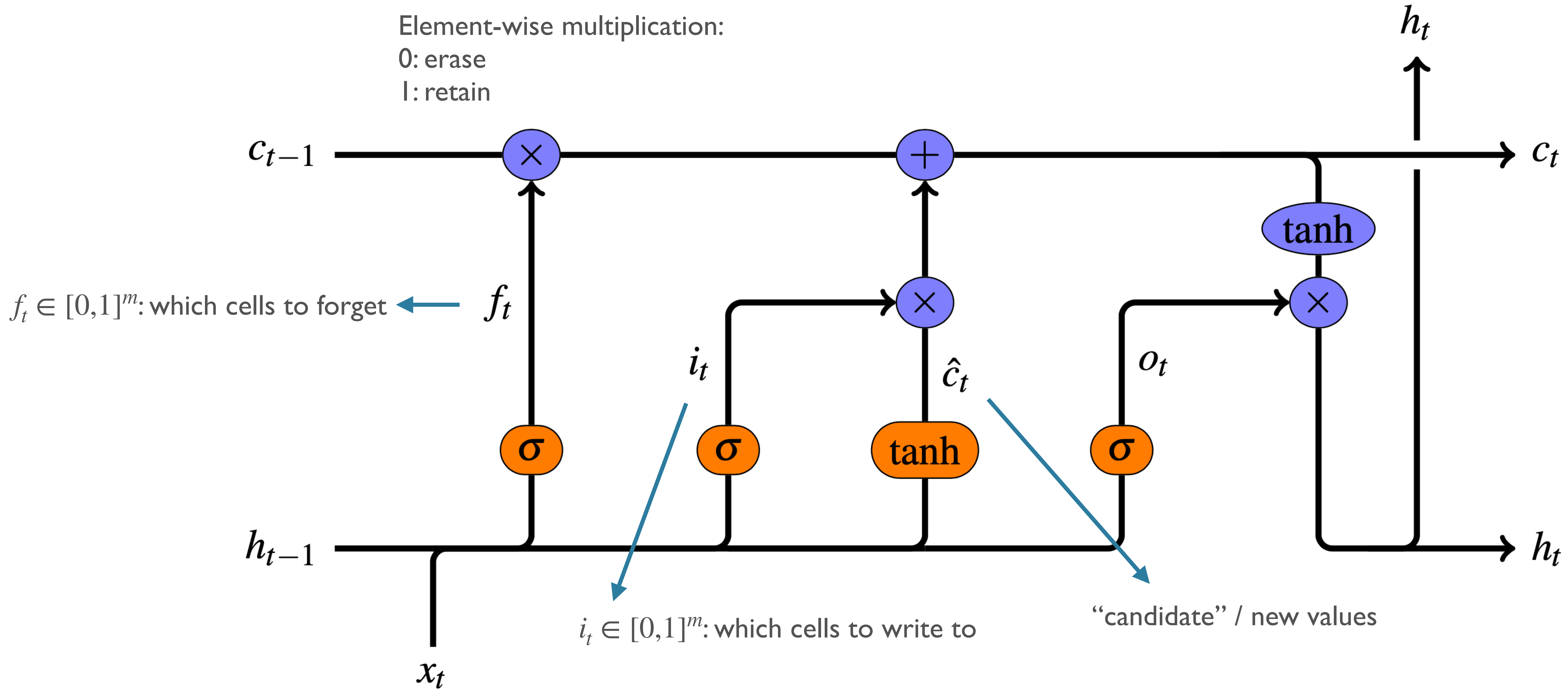
Element-wise multiplication:
0: erase
1: retain
$f_t \in [0,1]^m$: which cells to forget
c_{t-1}
h_{t-1}
x_t
f_t
i_t
$\hat{c}_t$
o_t
σ
σ
tanh
σ
tanh
h_t
c_t
h_t

LSTMs



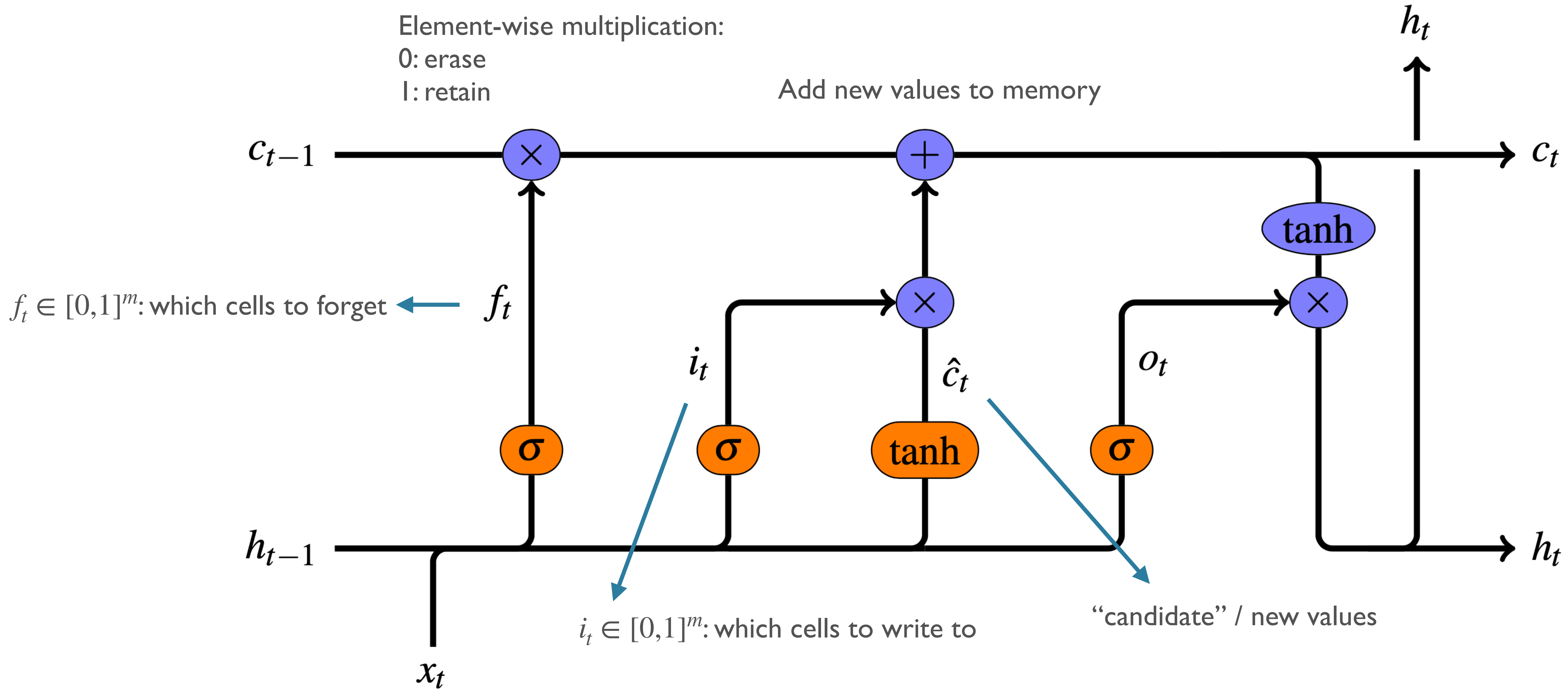
Element-wise multiplication:
0: erase
1: retain
h_t
c_{t-1}
c_t
tanh
$f_t \in [0,1]^m$: which cells to forget
f_t
i_t
$\hat{c}_t$
o_t
σ
σ
tanh
σ
h_{t-1}
h_t
x_t
$i_t \in [0,1]^m$: which cells to write to

LSTMs

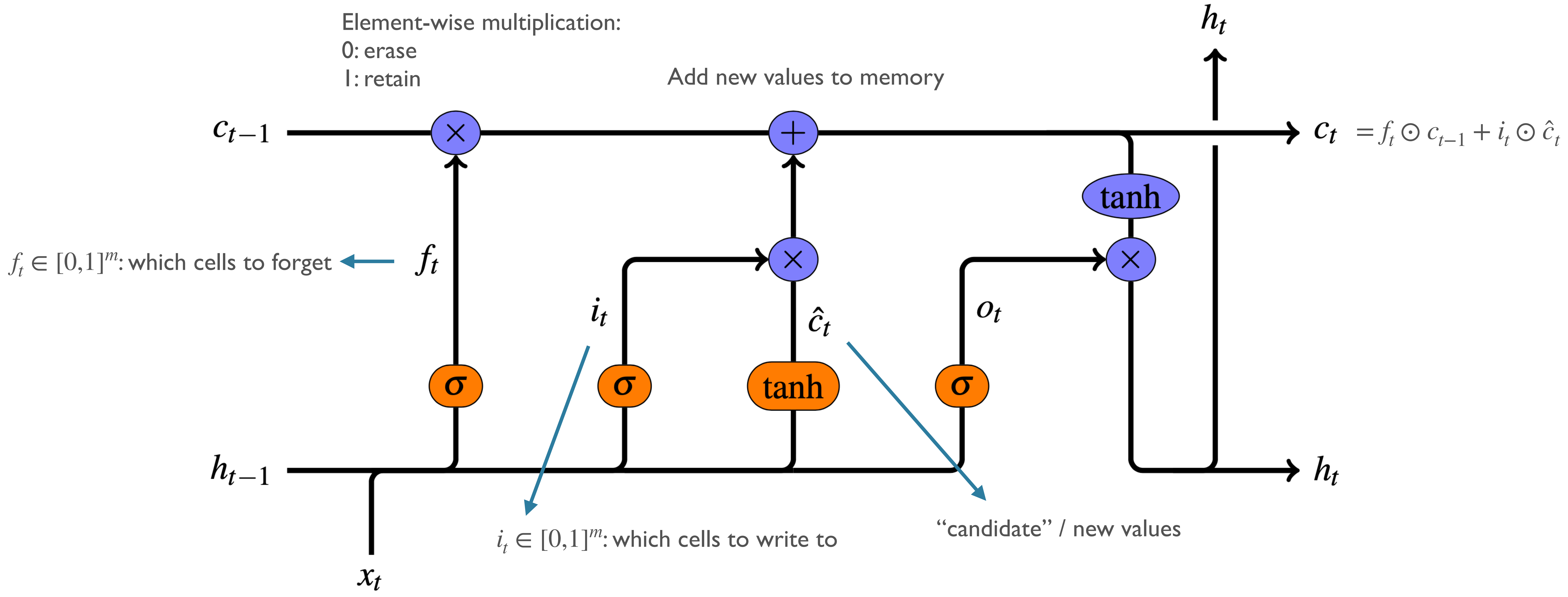


Element-wise multiplication:
0: erase
1: retain
$f_t \in [0,1]^m$: which cells to forget
$i_t \in [0,1]^m$: which cells to write to
"candidate" / new values
c_{t-1}
c_t
h_{t-1}
h_t
h_t
x_t
f_t
i_t
$\hat{c}_t$
o_t
σ
σ
tanh
σ
tanh

LSTMs

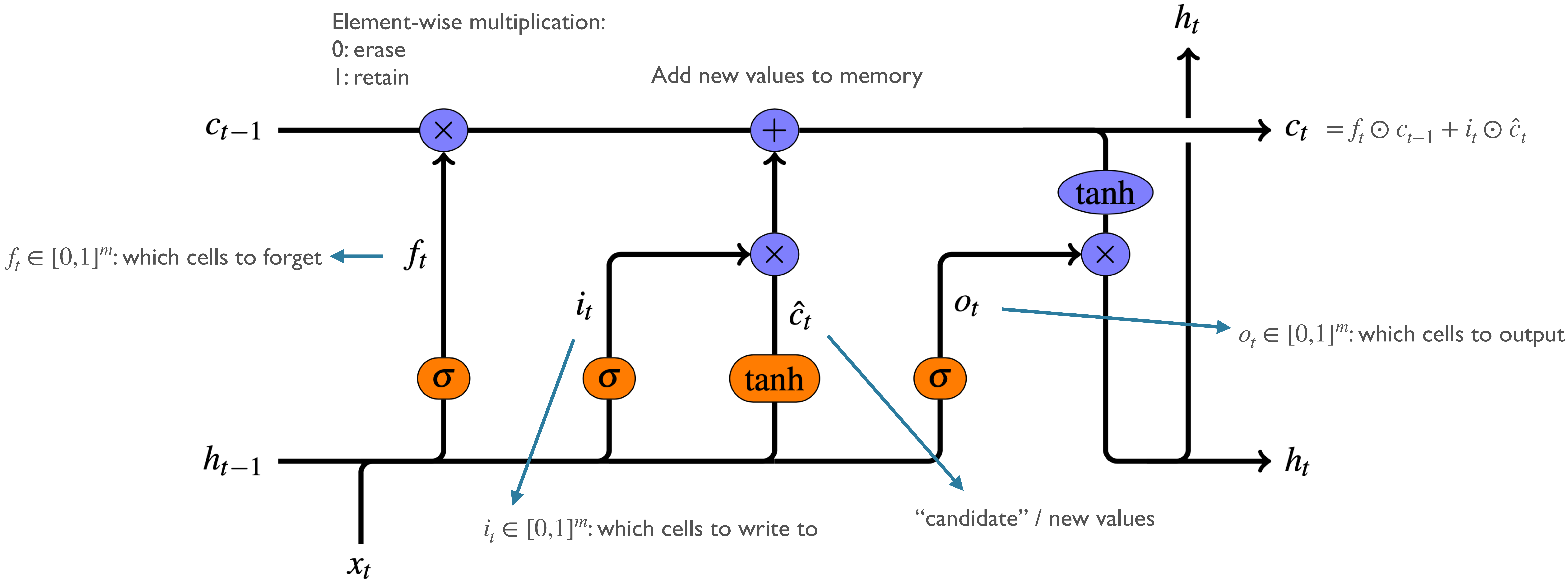

Element-wise multiplication:
0: erase
1: retain
Add new values to memory
h_t
c_{t-1}
c_t
tanh
$f_t \in [0,1]^m$: which cells to forget
f_t
i_t
$\hat{c}_t$
o_t
σ
σ
tanh
σ
h_{t-1}
h_t
$i_t \in [0,1]^m$: which cells to write to
"candidate" / new values
x_t

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
x_t
$i_t \in [0,1]^m$: which cells to write to
"candidate" / new values

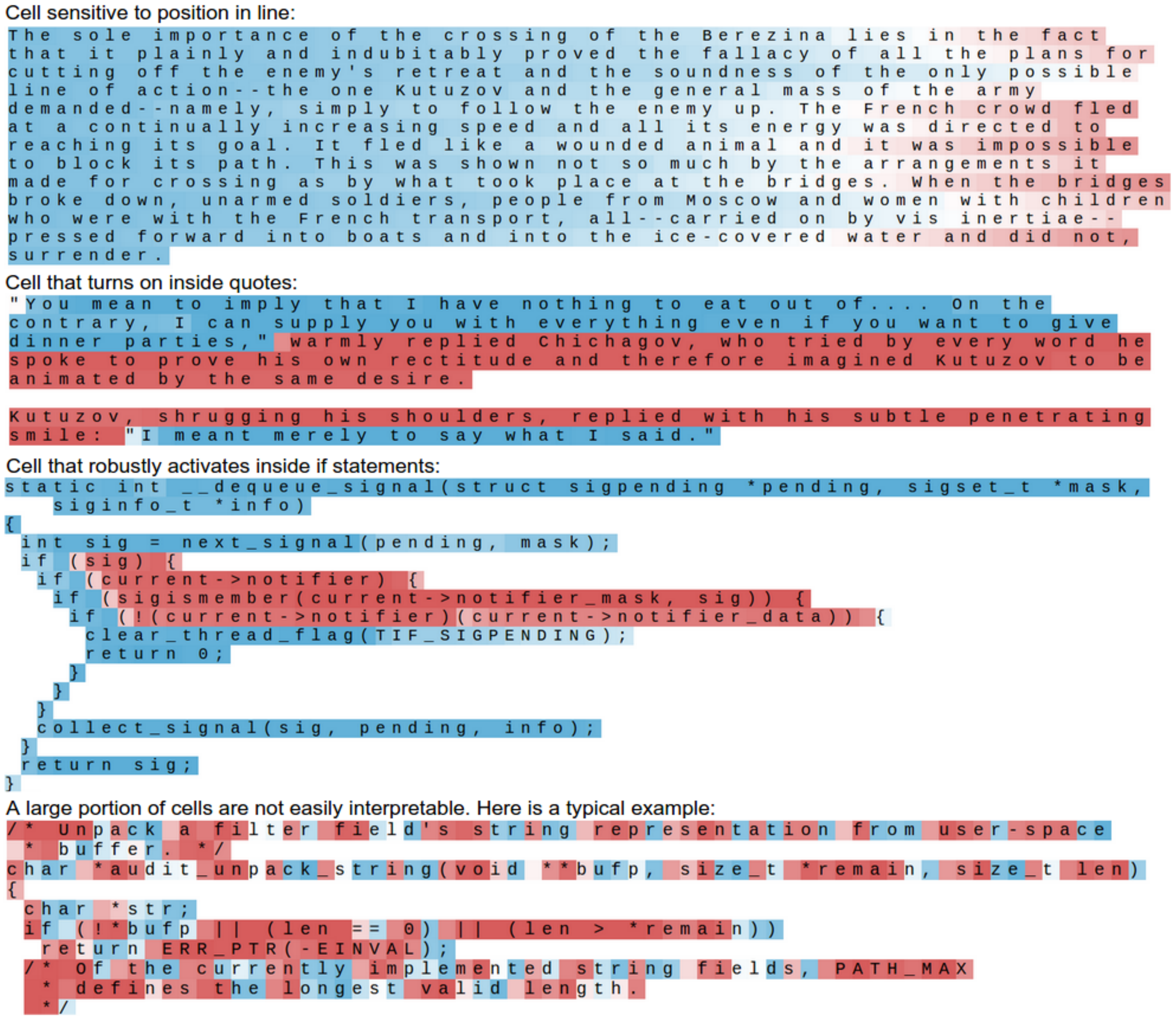
LSTMs



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

Fun with LSTM (character) LMs

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

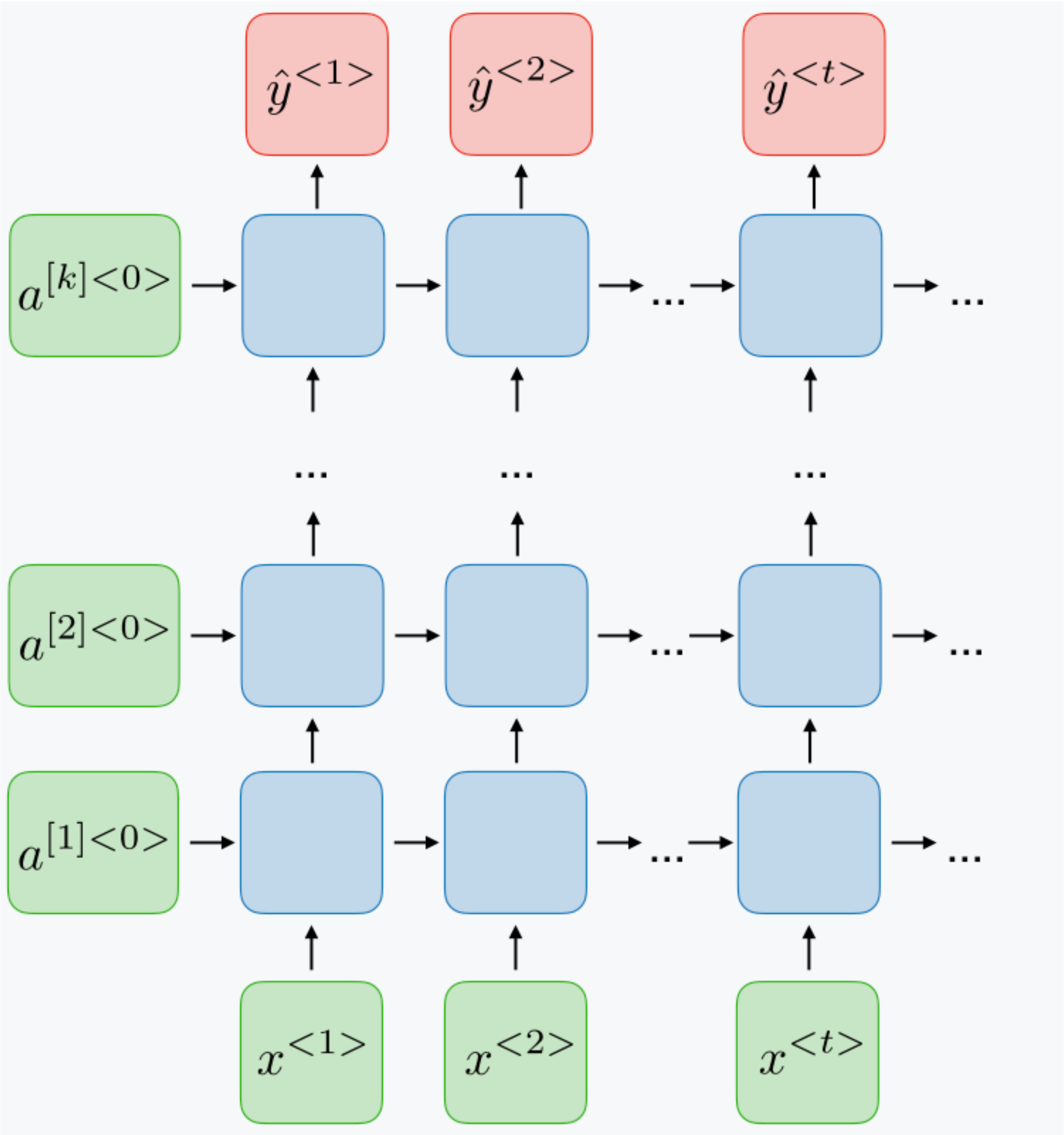


Some LSTM LMs

- Jozefowicz et al 2016 ("Exploring the Limits of Language Modeling")
  - https://github.com/tensorflow/models/tree/master/research/lm_1b
- Gulordava et al 2018 ("Colorless Recurrent Neural Networks Dream Hierarchically")
  - Fairly easy to use, lots of analysis work using either their pre-trained LM and/or their protocol
  - https://github.com/facebookresearch/colorlessgreenRNNs

Two Extensions

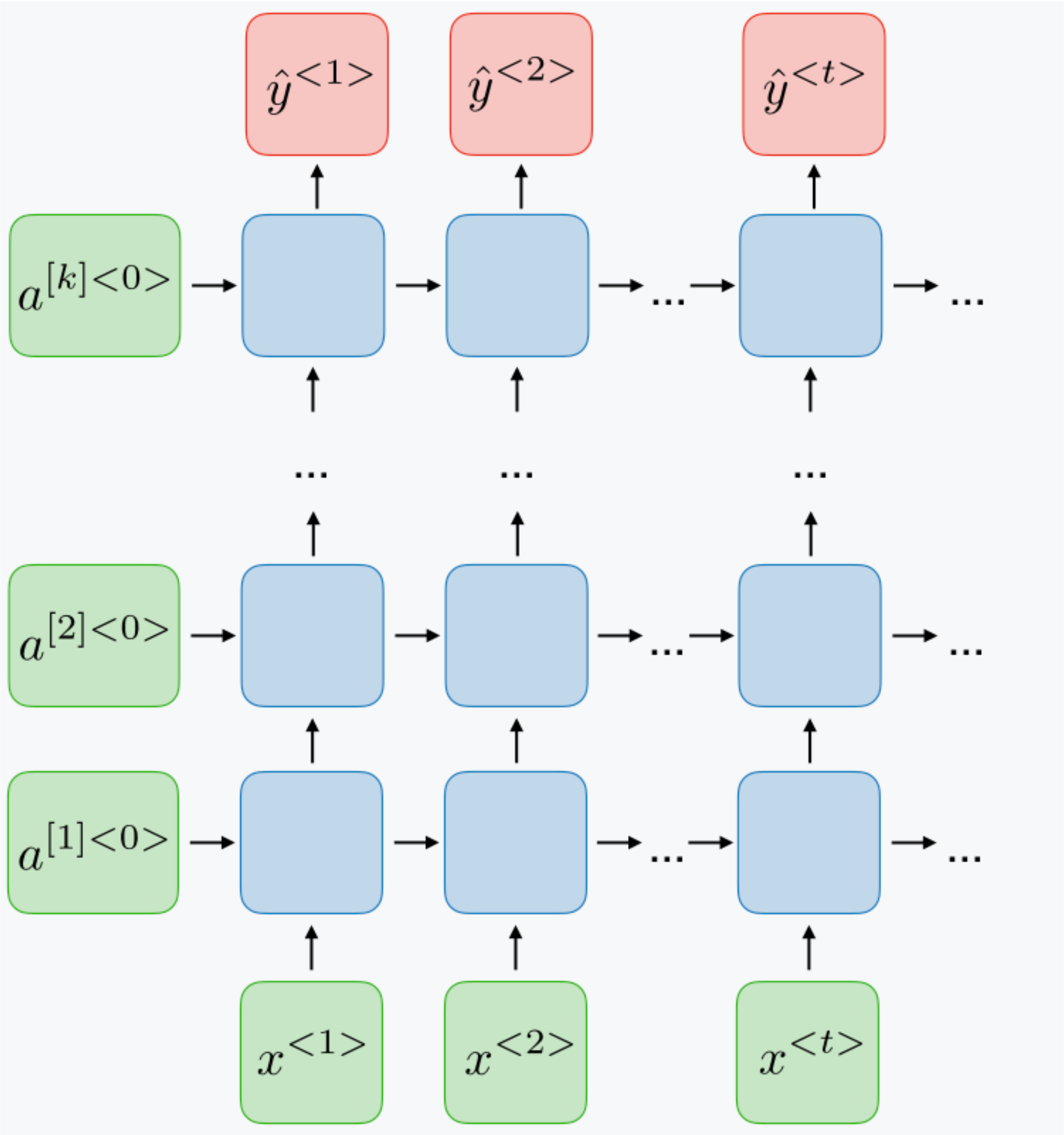
- Deep RNNs:



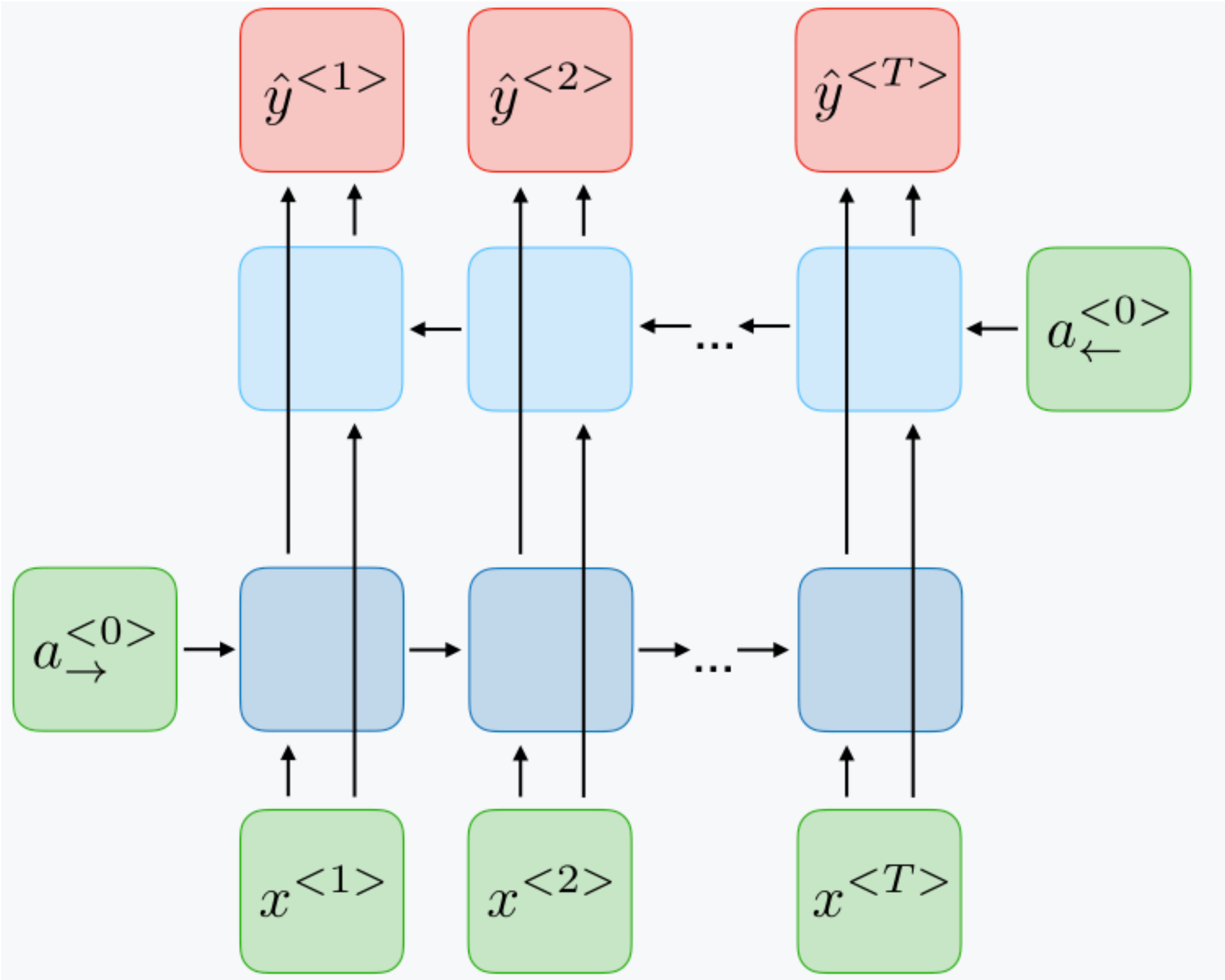
Source: RNN cheat sheet

Two Extensions

- Deep RNNs:



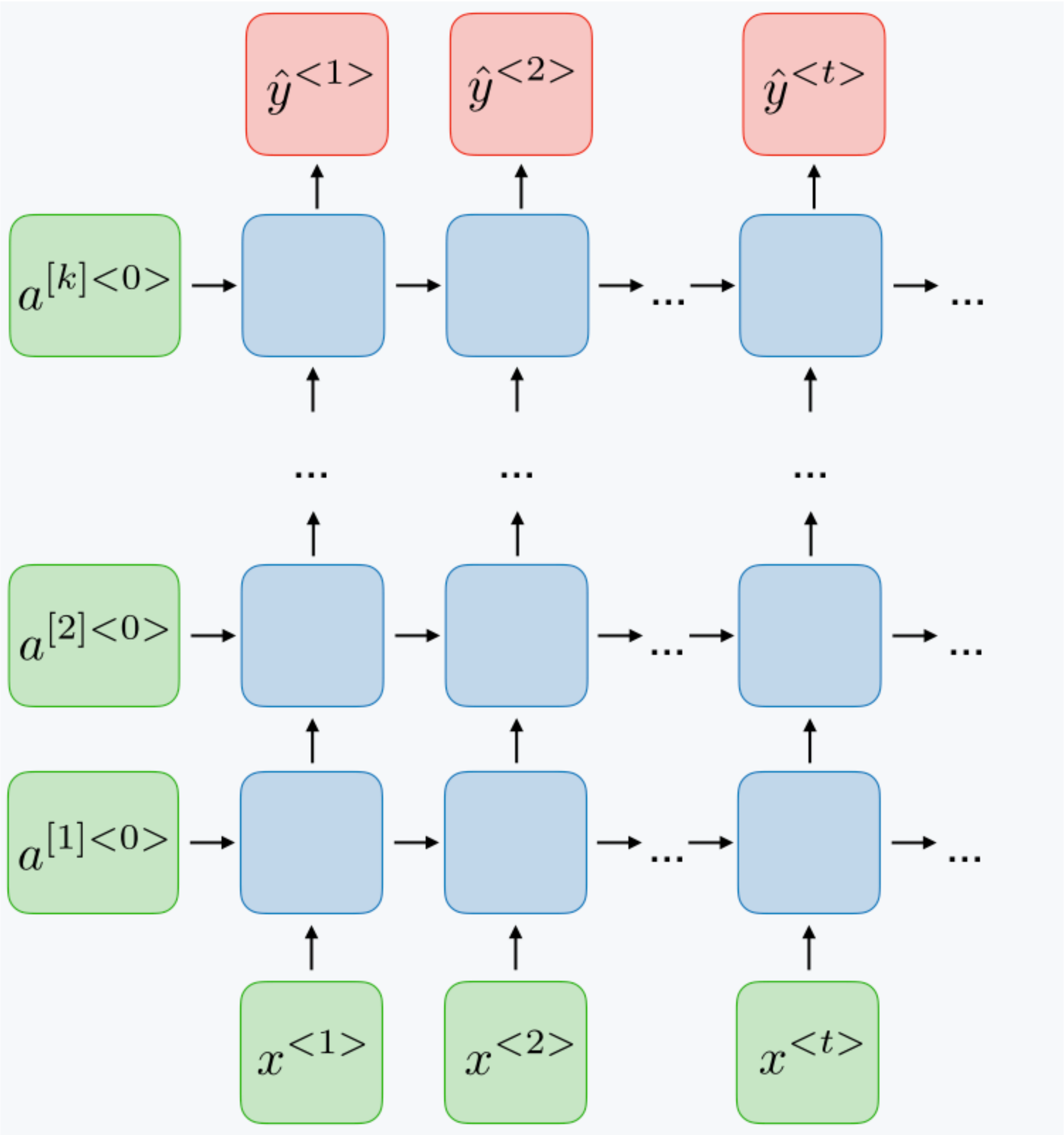
- Bidirectional RNNs:



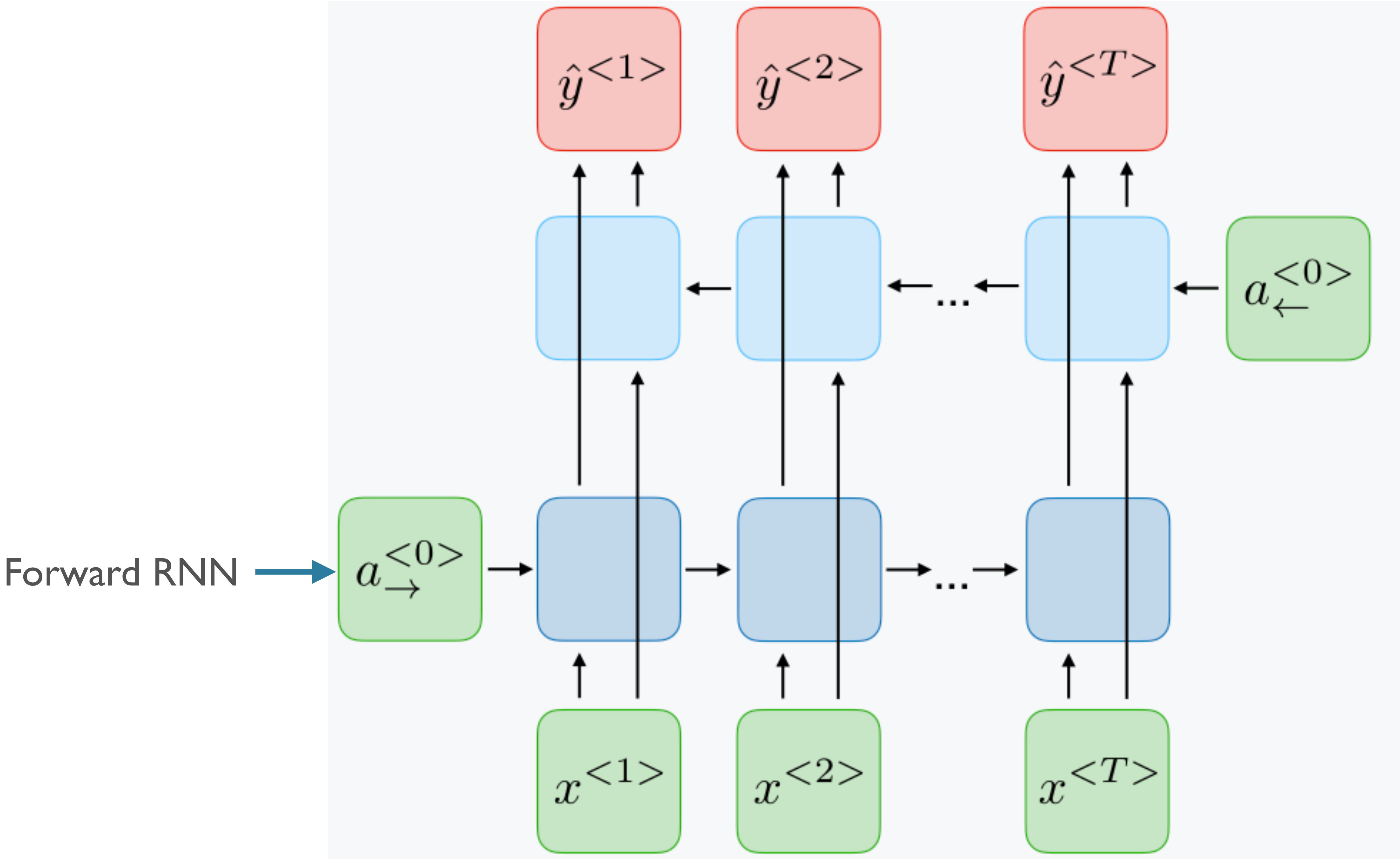
Source: RNN cheat sheet

Two Extensions

- Deep RNNs:



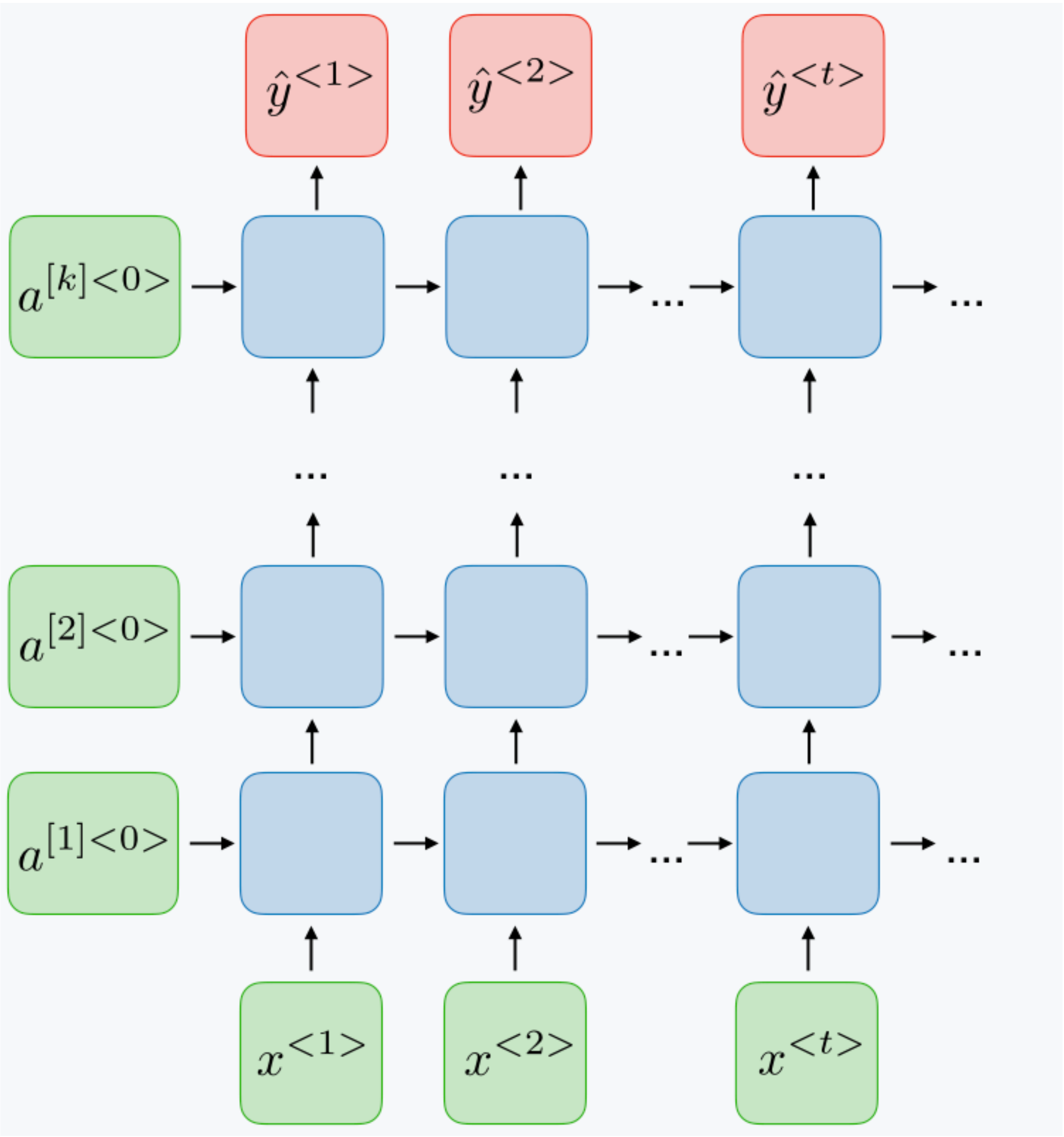
- Bidirectional RNNs:



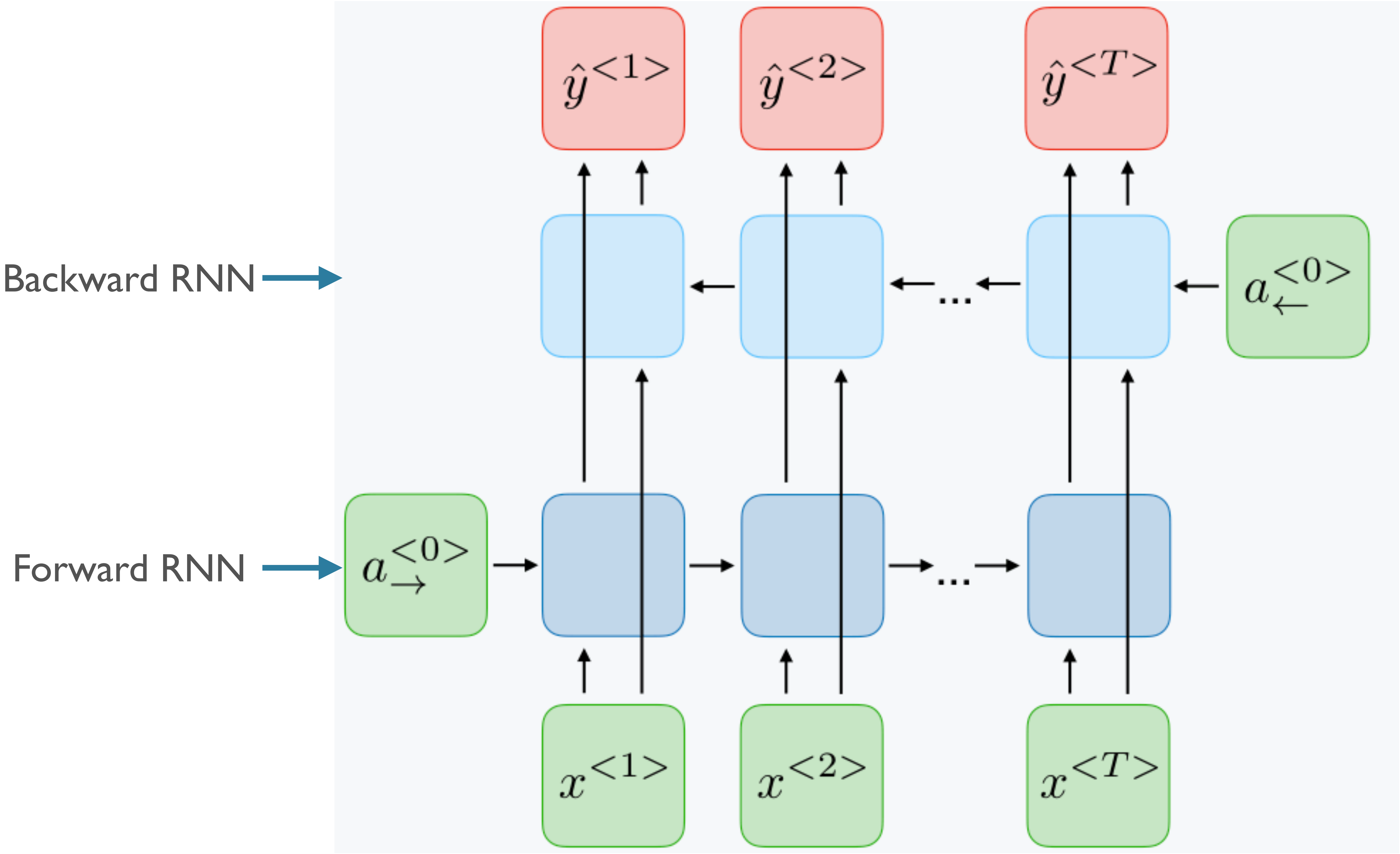
Source: RNN cheat sheet

Two Extensions

- Deep RNNs:



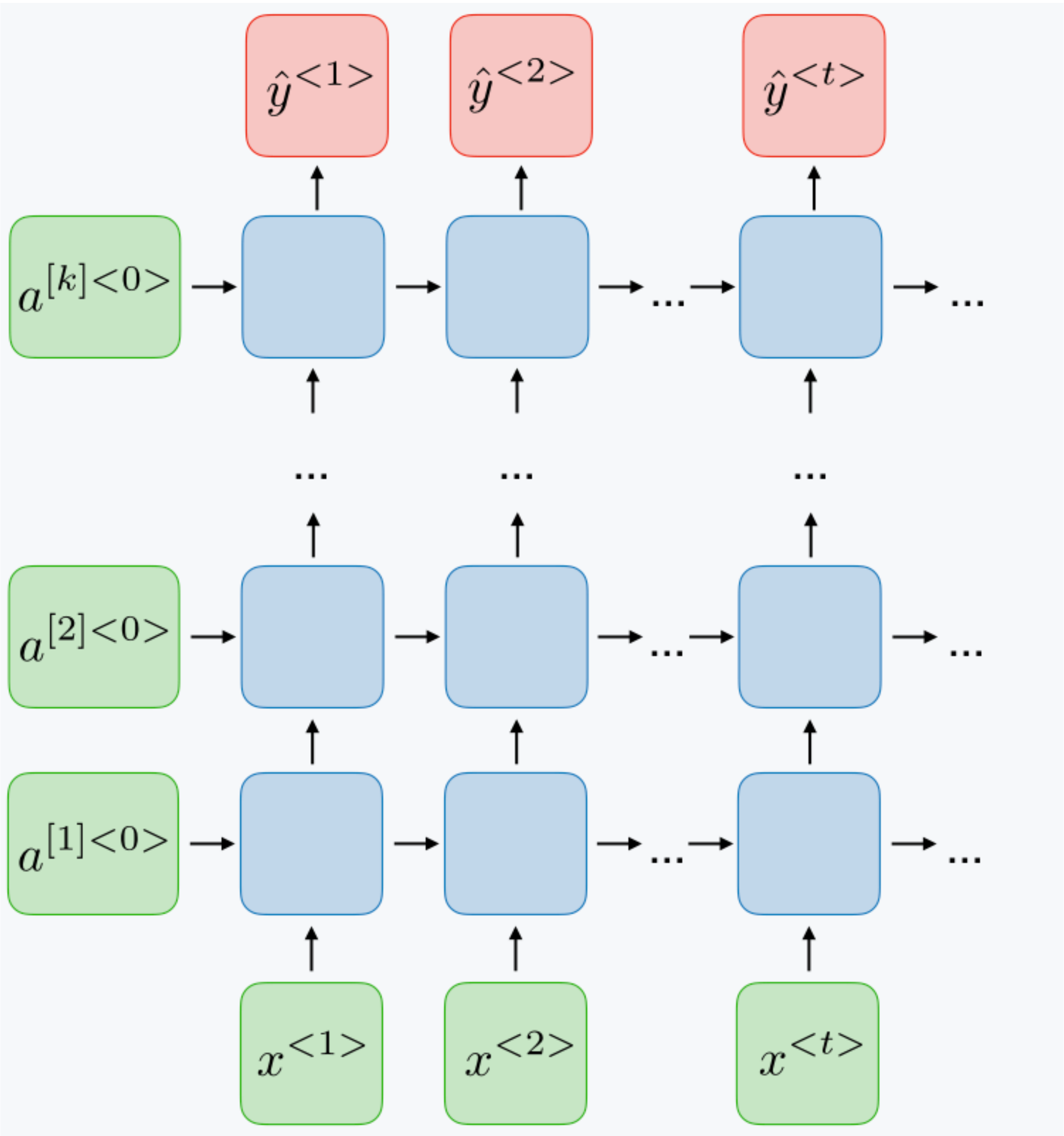
- Bidirectional RNNs:



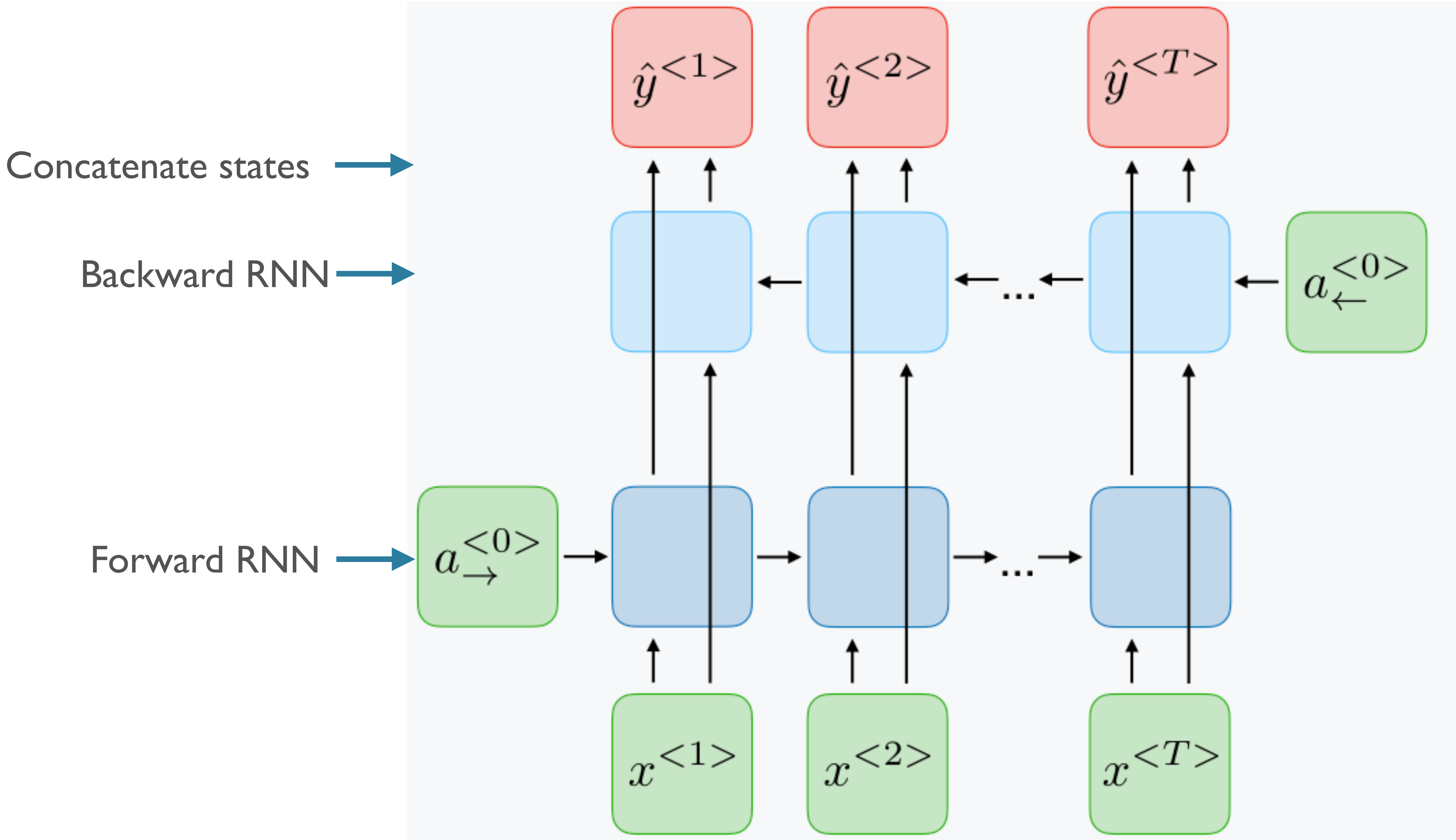
Source: RNN cheat sheet

Two Extensions

- Deep RNNs:



- Bidirectional RNNs:



Source: RNN cheat sheet

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

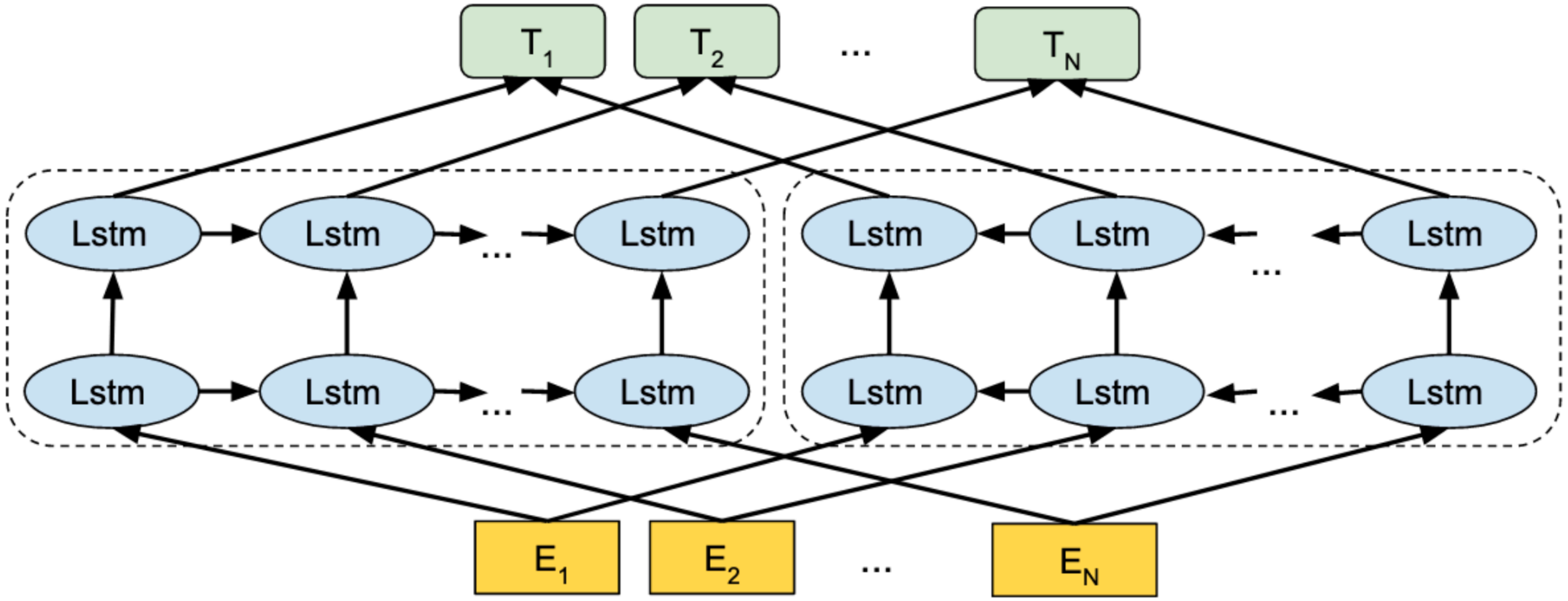
Abstract

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

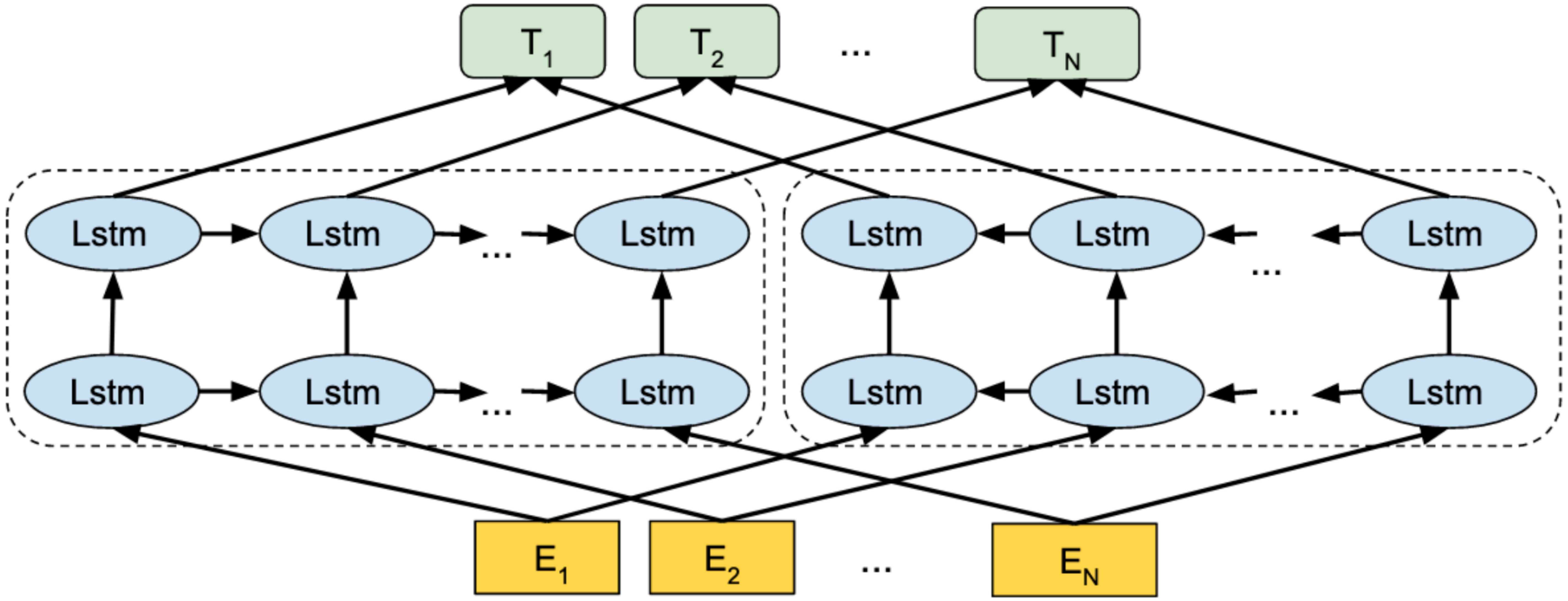
ELMo Model



Source: BERT paper

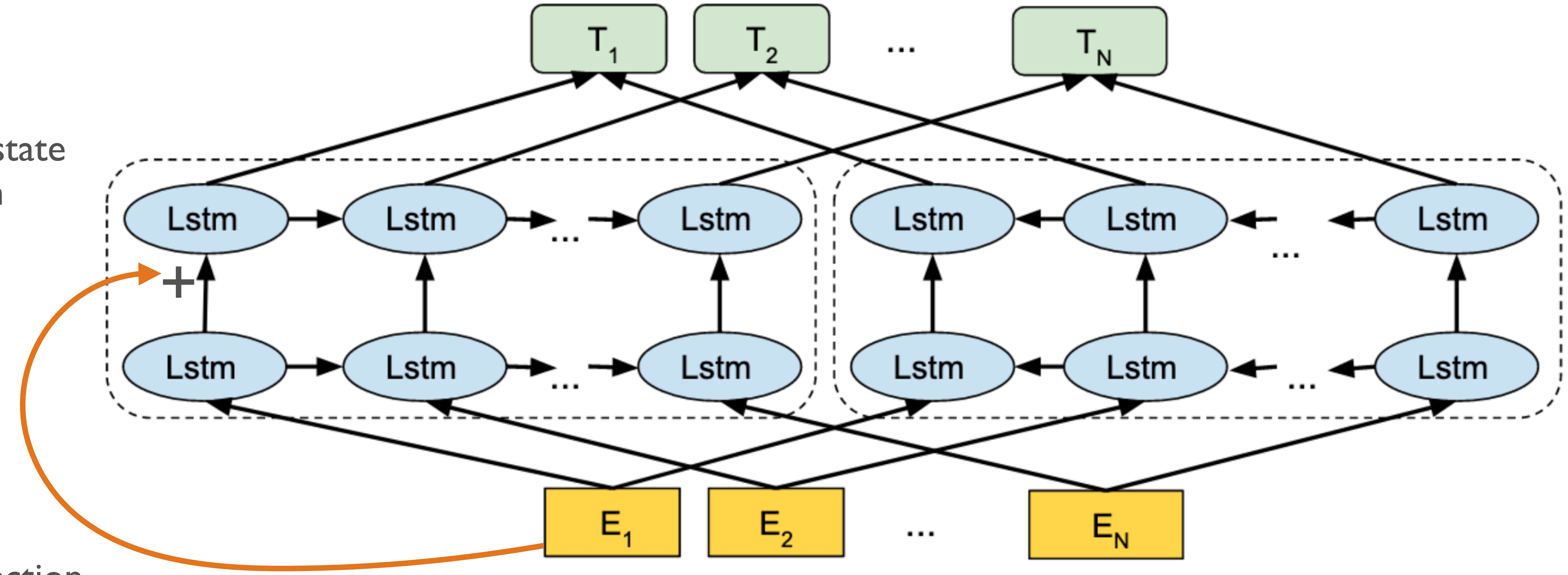
ELMo Model

4096-d hidden state
512d projection



ELMo Model

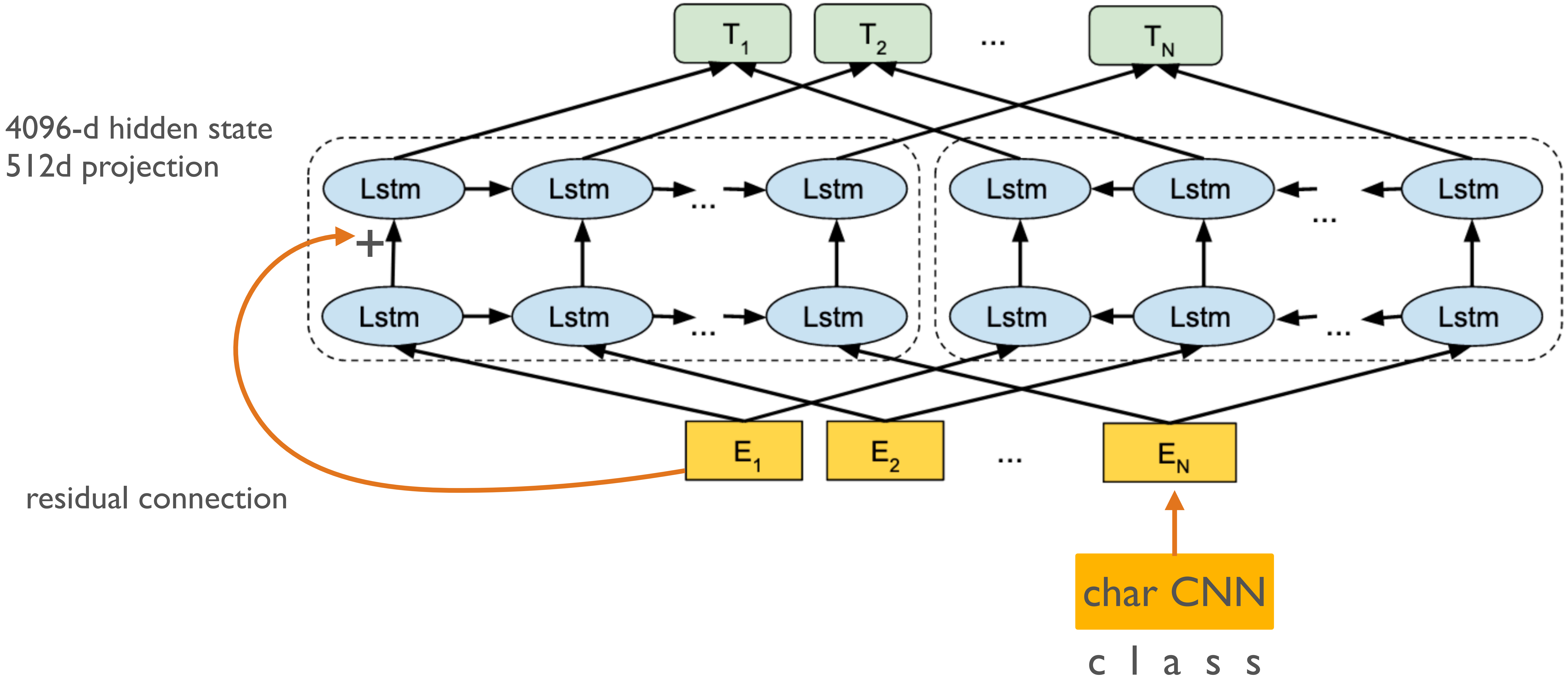
4096-d hidden state
512d projection



residual connection

Source: BERT paper

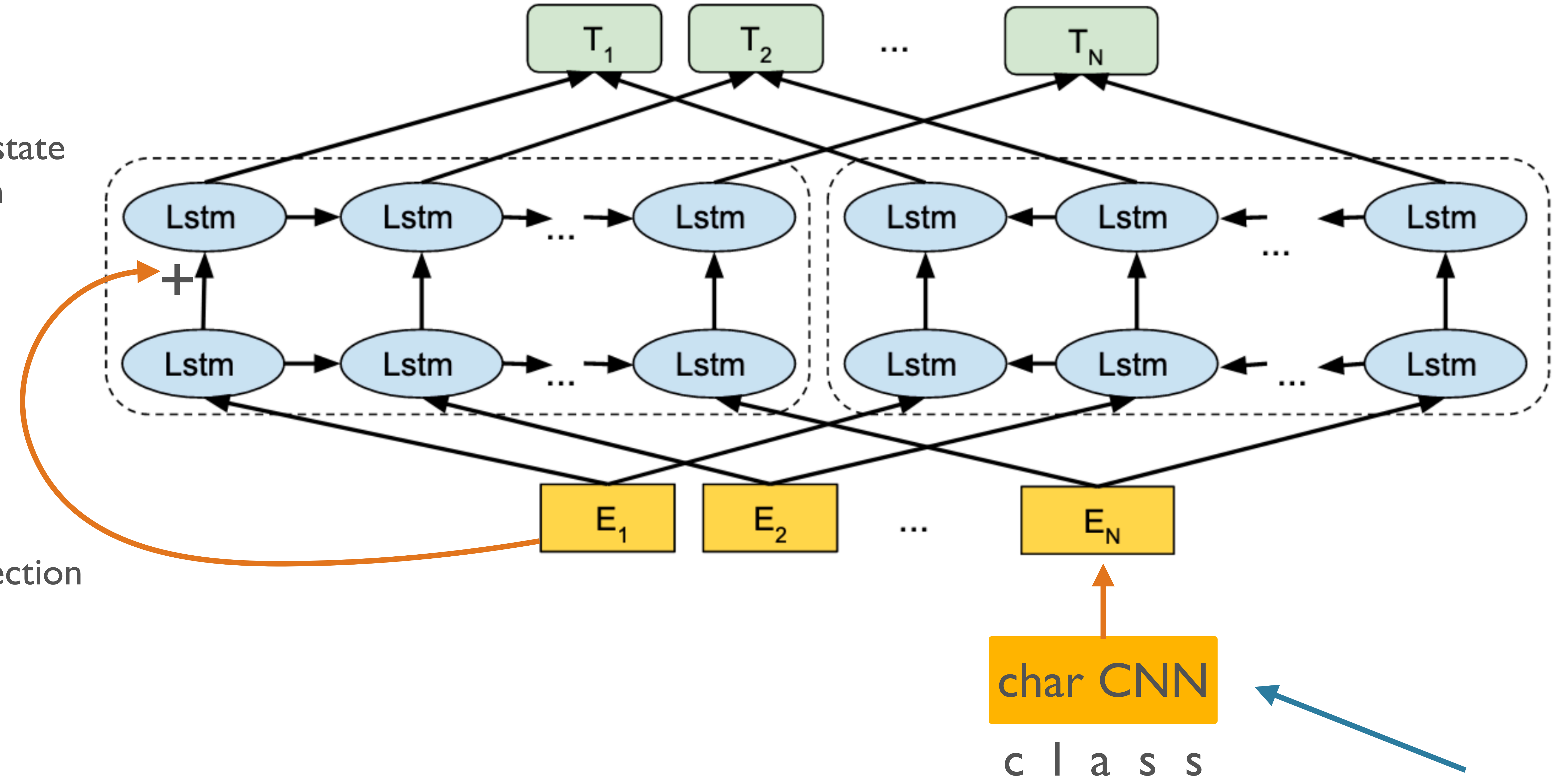
ELMo Model



T_1
T_2
...
T_N
4096-d hidden state
512d projection
Lstm
Lstm
...
Lstm
Lstm
Lstm
...
Lstm
Lstm
Lstm
...
Lstm
Lstm
Lstm
...
Lstm
+
E_1
E_2
...
E_N
residual connection
char CNN
c l a s s

ELMo Model

4096-d hidden state
512d projection



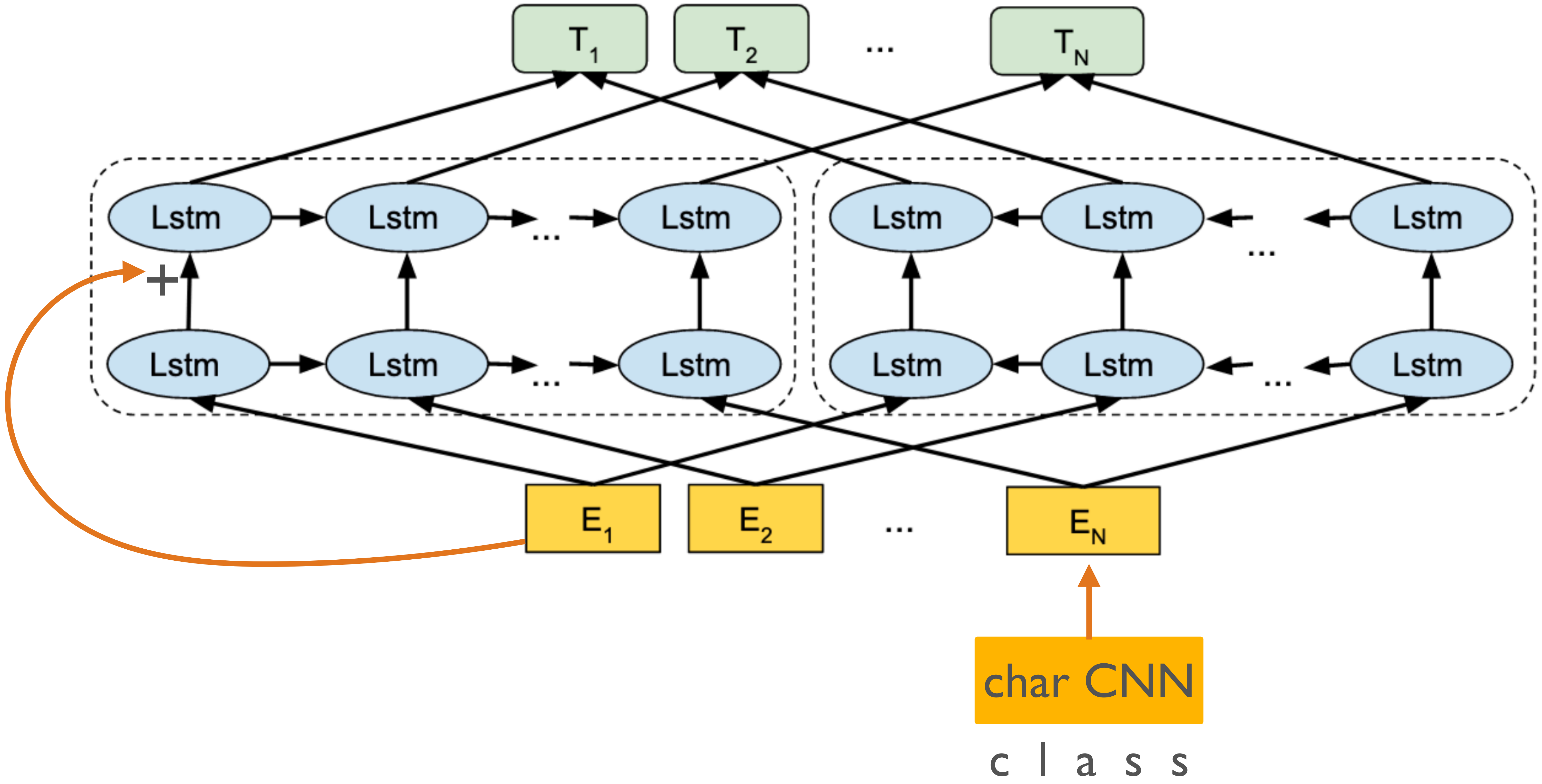
residual connection

Helps with rare / new words (no OOV)

ELMo Training

- 10 epochs on [1B Word Benchmark]
- NB: not SOTA perplexity even at time of publishing
  - See “Exploring the Limits of Language Modeling” paper
- Regularization:
  - [Dropout]
  - L2 norm

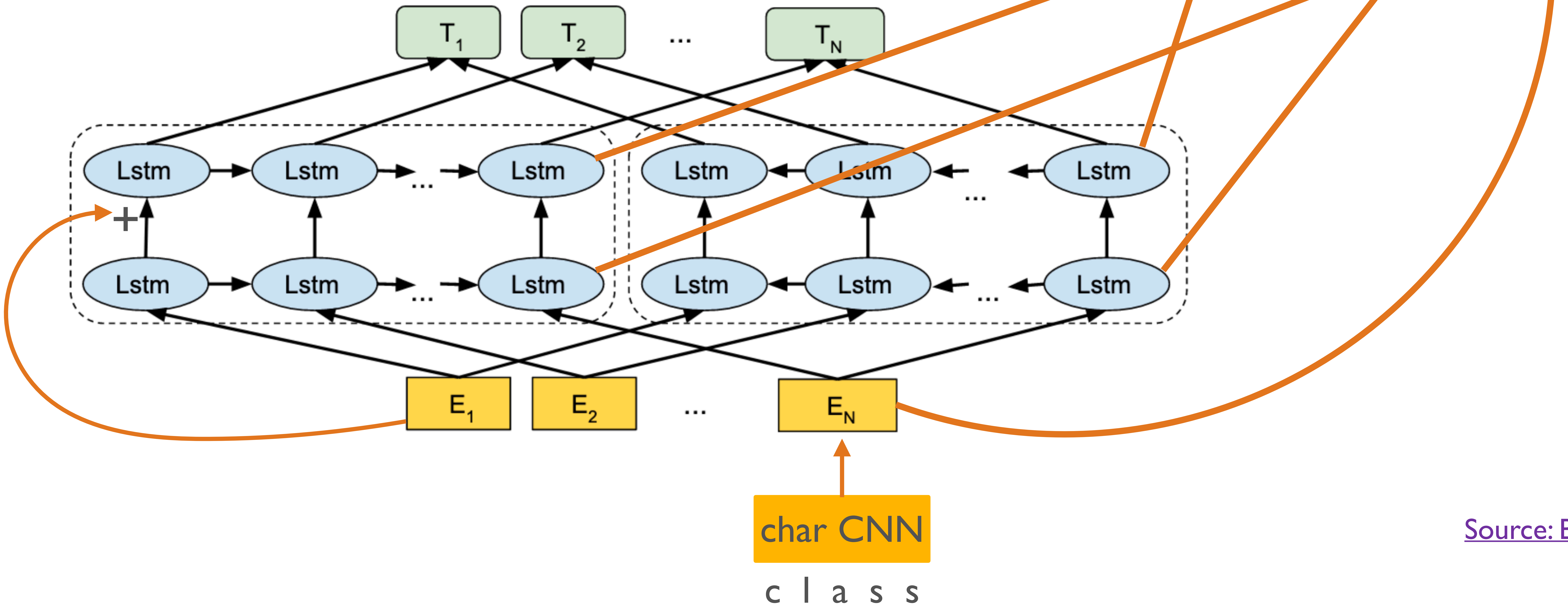
Transferring ELMo



Source: BERT paper

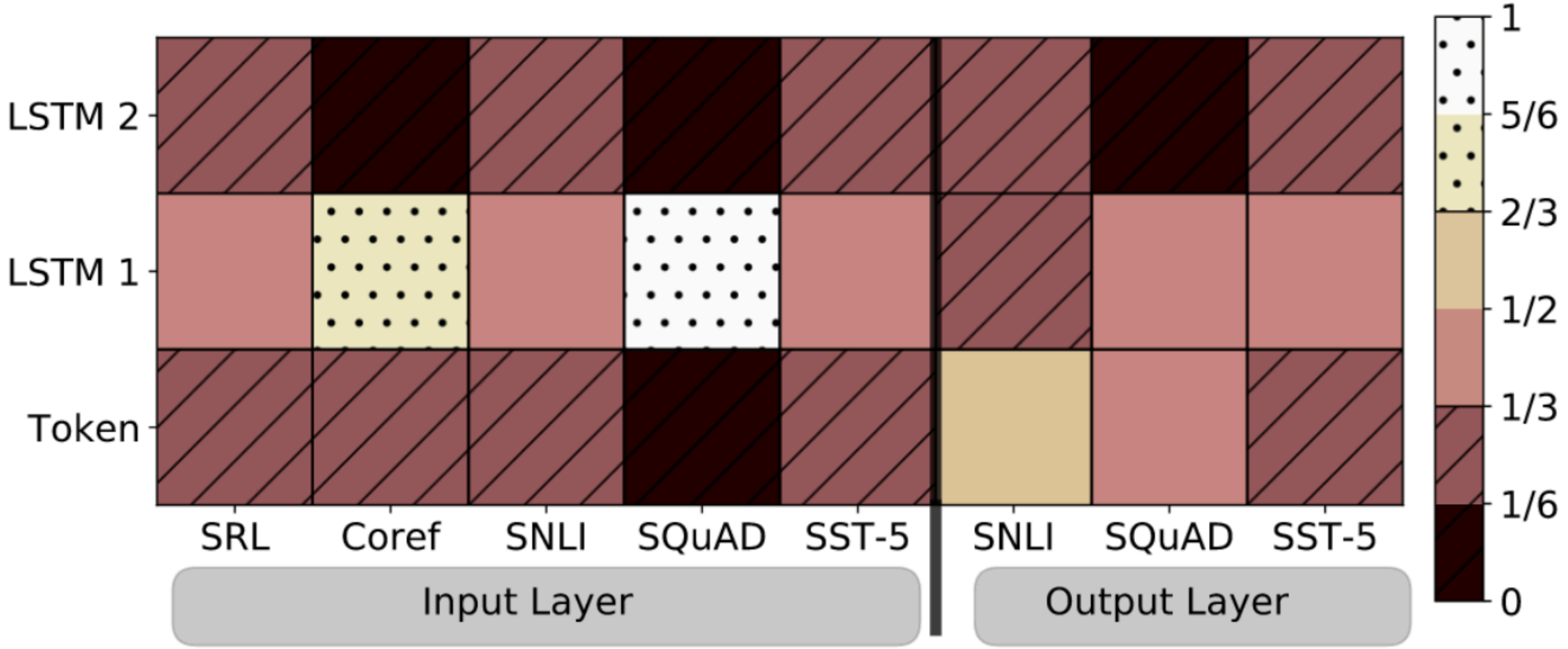
Transferring ELMo

$$\mathbf{ELMo}_k^{task} = E(R_k; \Theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} \mathbf{h}_{k,j}^{LM}.$$



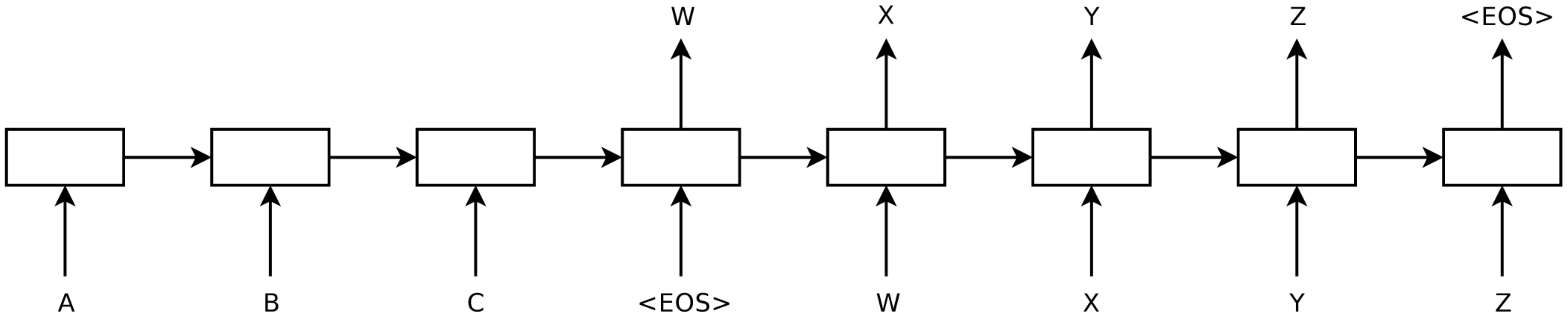
Source: BERT paper

Layer Weights by Transfer Task



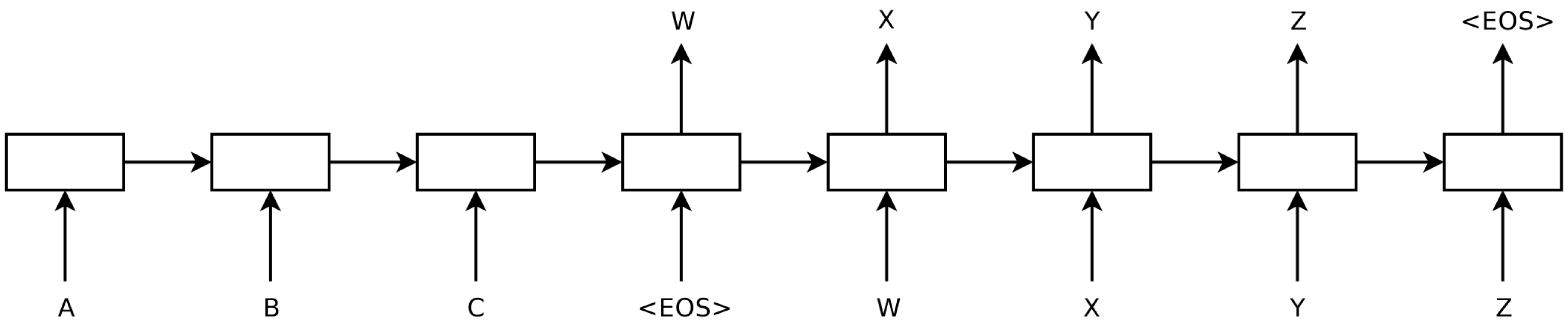
Attention

seq2seq architecture [e.g. NMT]



Sutskever et al 2013

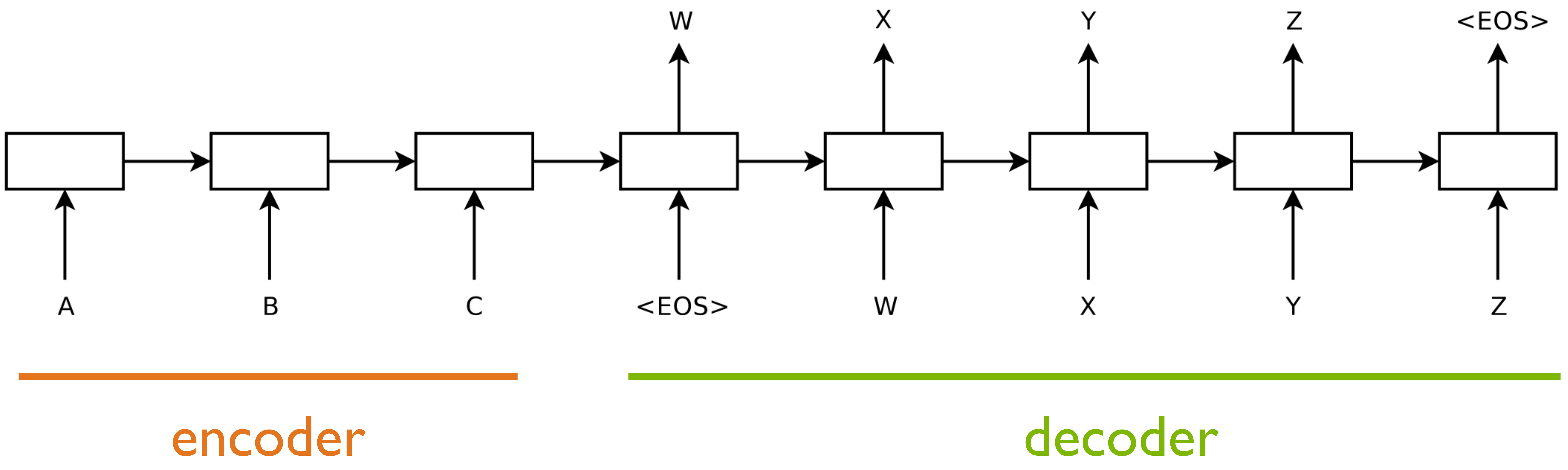
seq2seq architecture [e.g. NMT]



encoder

Sutskever et al 2013

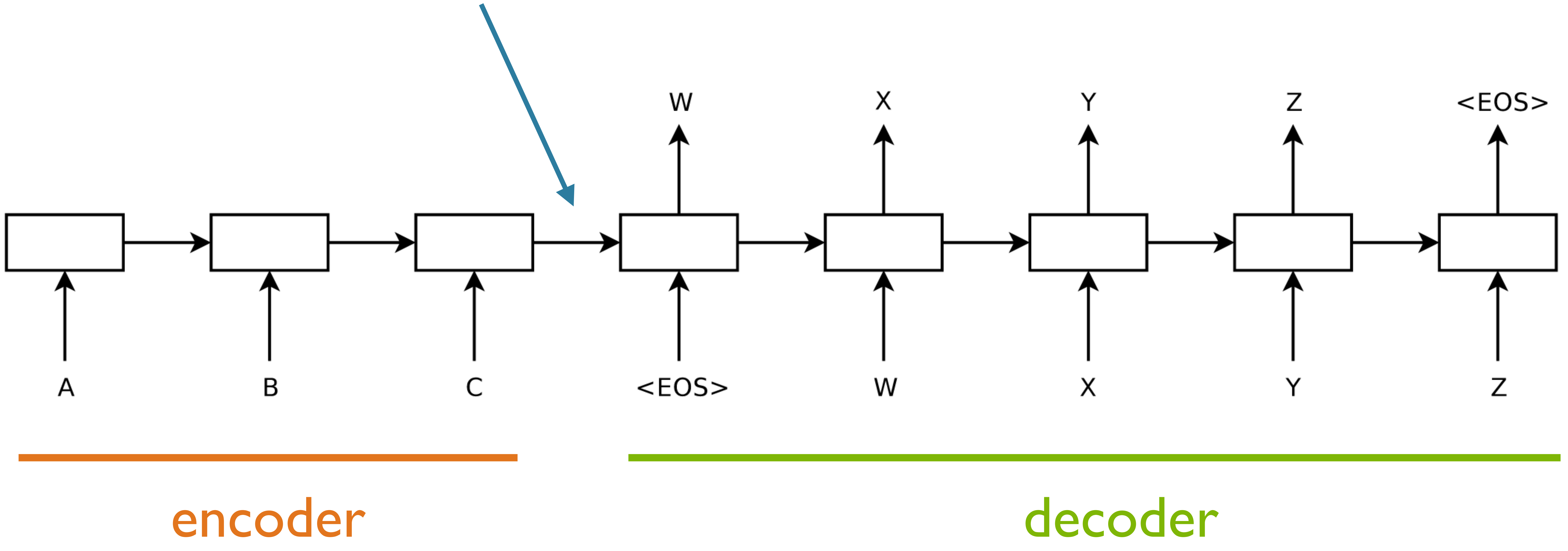
seq2seq architecture [e.g. NMT]



Sutskever et al 2013

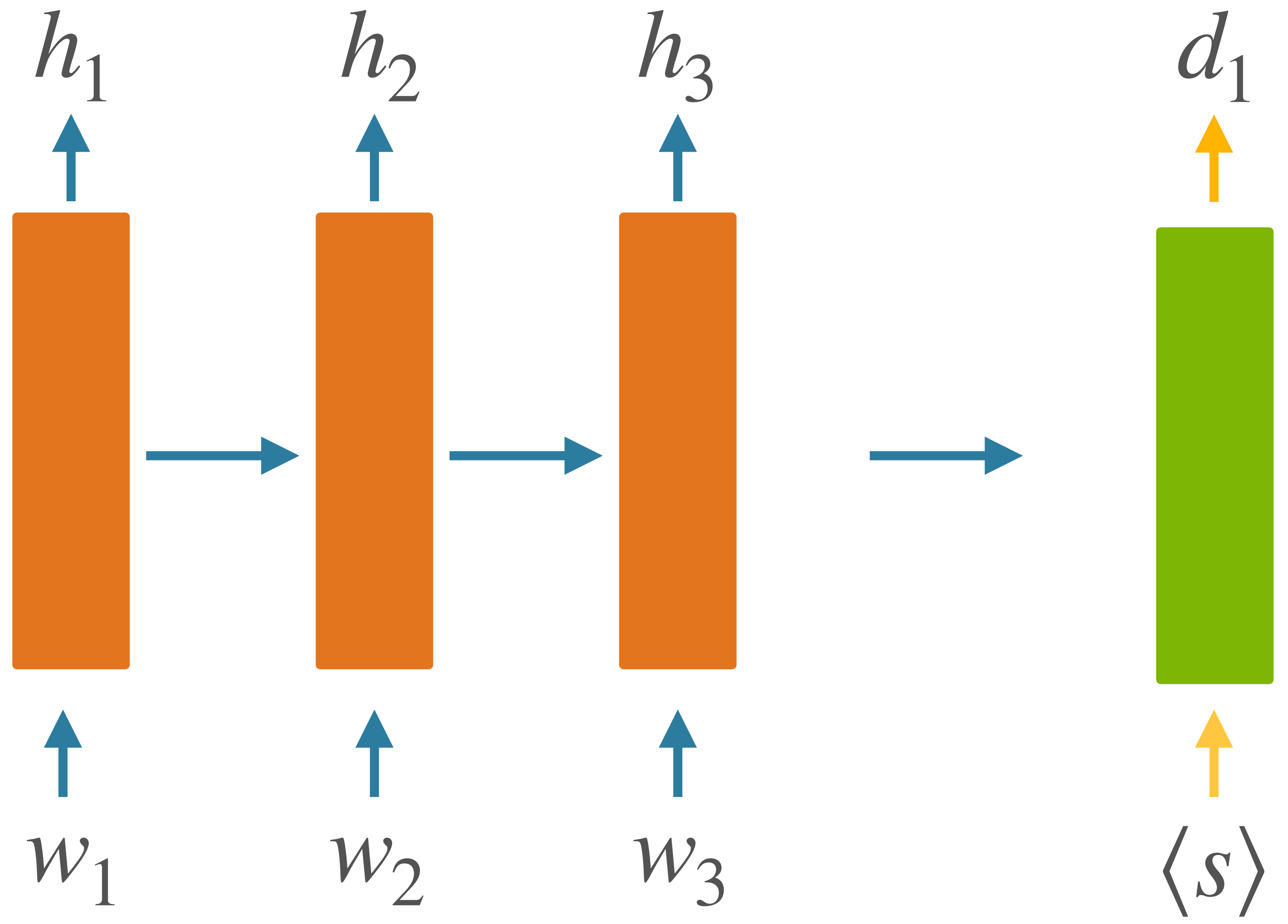
seq2seq architecture [e.g. NMT]

Decoder can only see info in this one vector
all info about source must be "crammed" into here



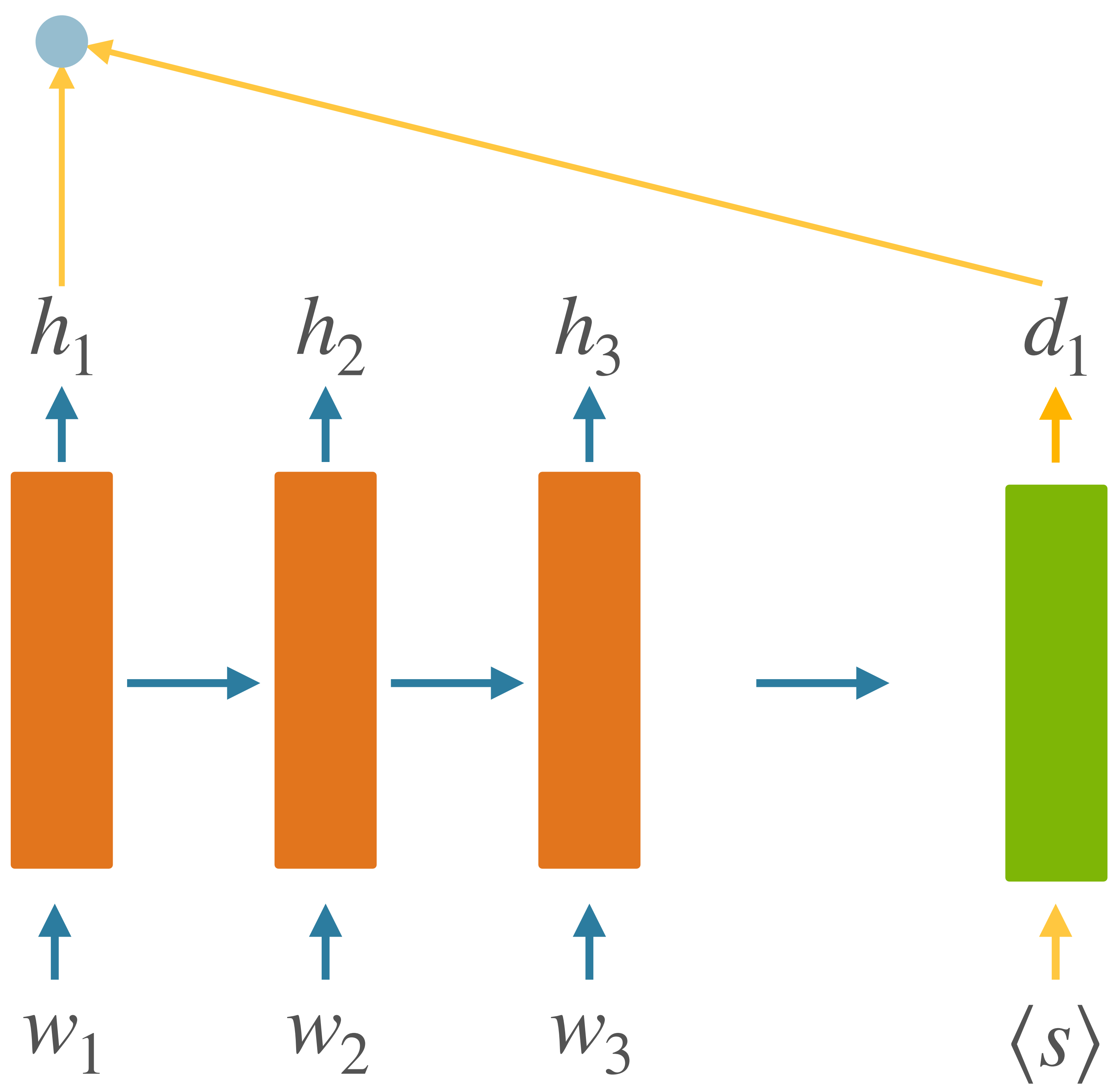
Sutskever et al 2013

Adding Attention



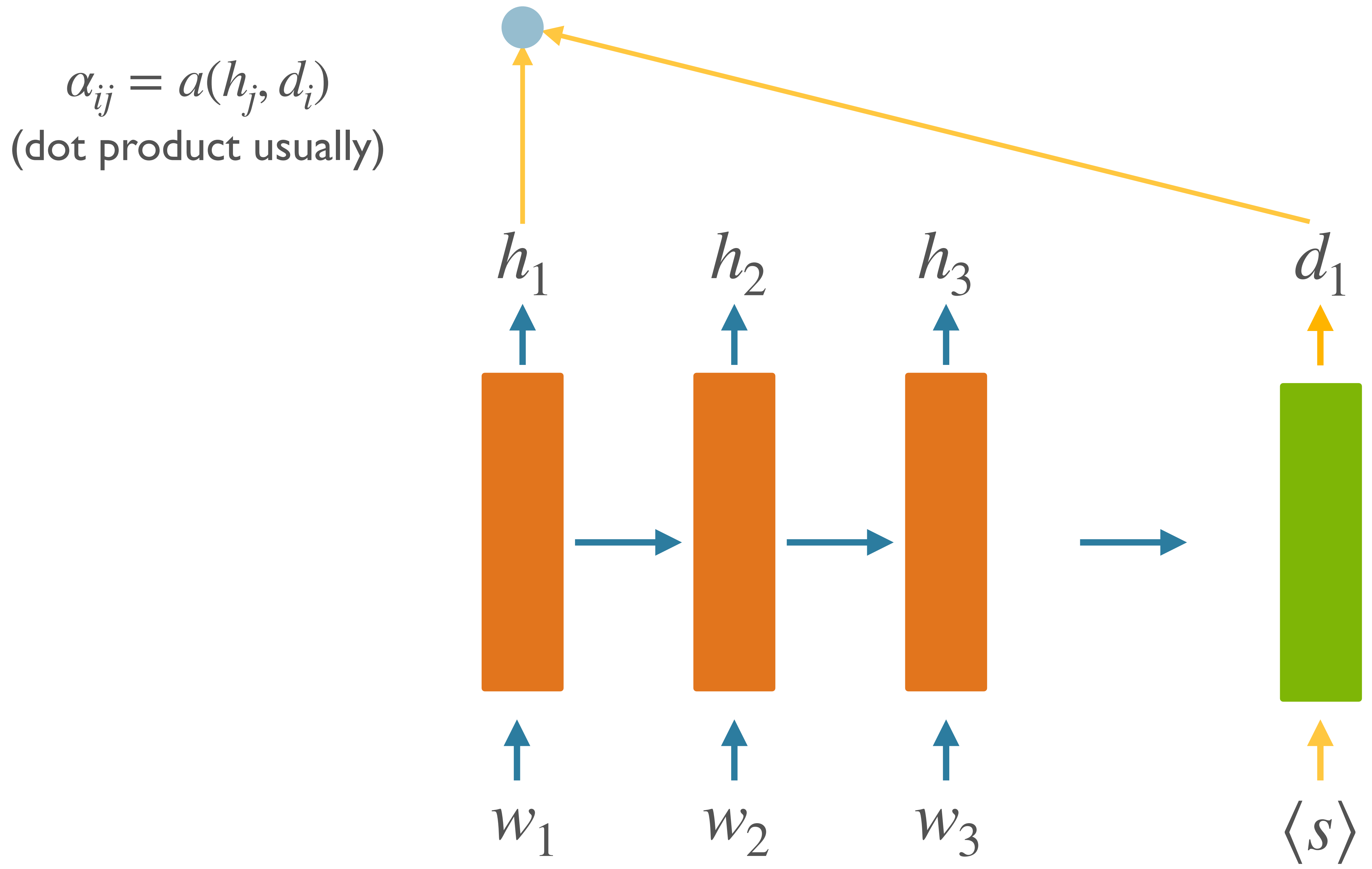
Badhanau et al 2014

Adding Attention



Badhanau et al 2014

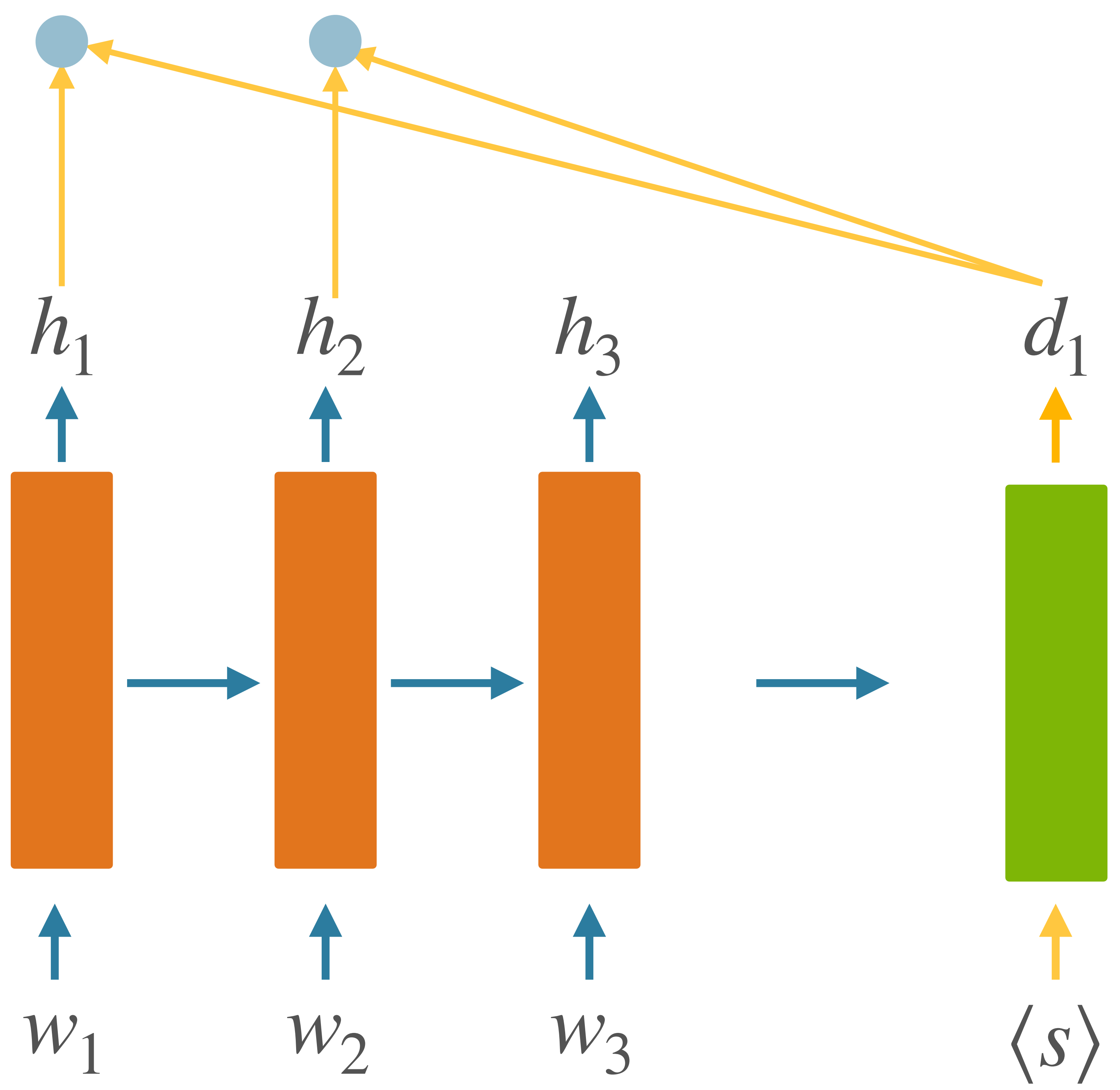
Adding Attention



Badhanau et al 2014

Adding Attention

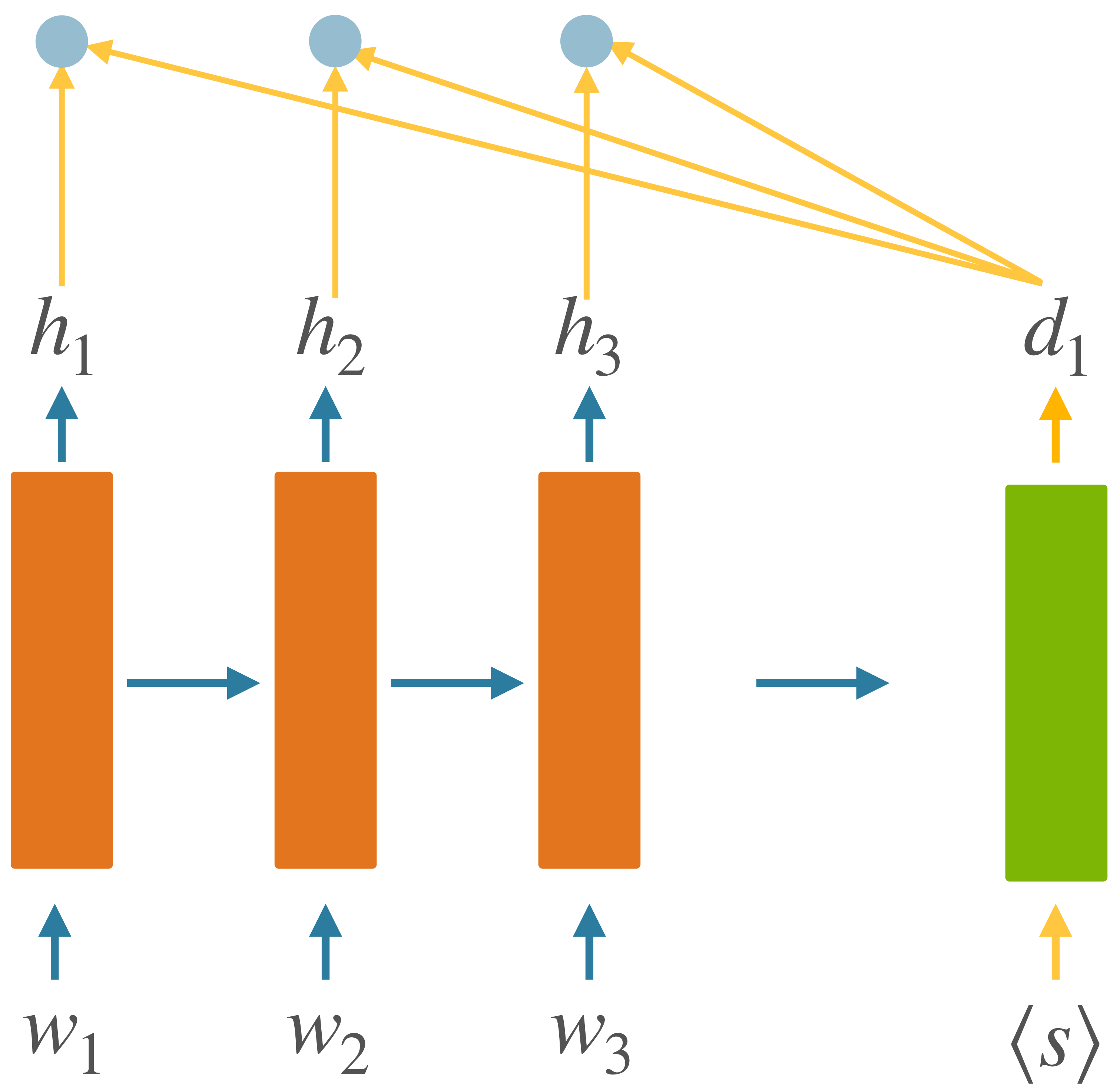
$\alpha_{ij} = a(h_j, d_i)$
(dot product usually)



Badhanau et al 2014

Adding Attention

$\alpha_{ij} = a(h_j, d_i)$
(dot product usually)

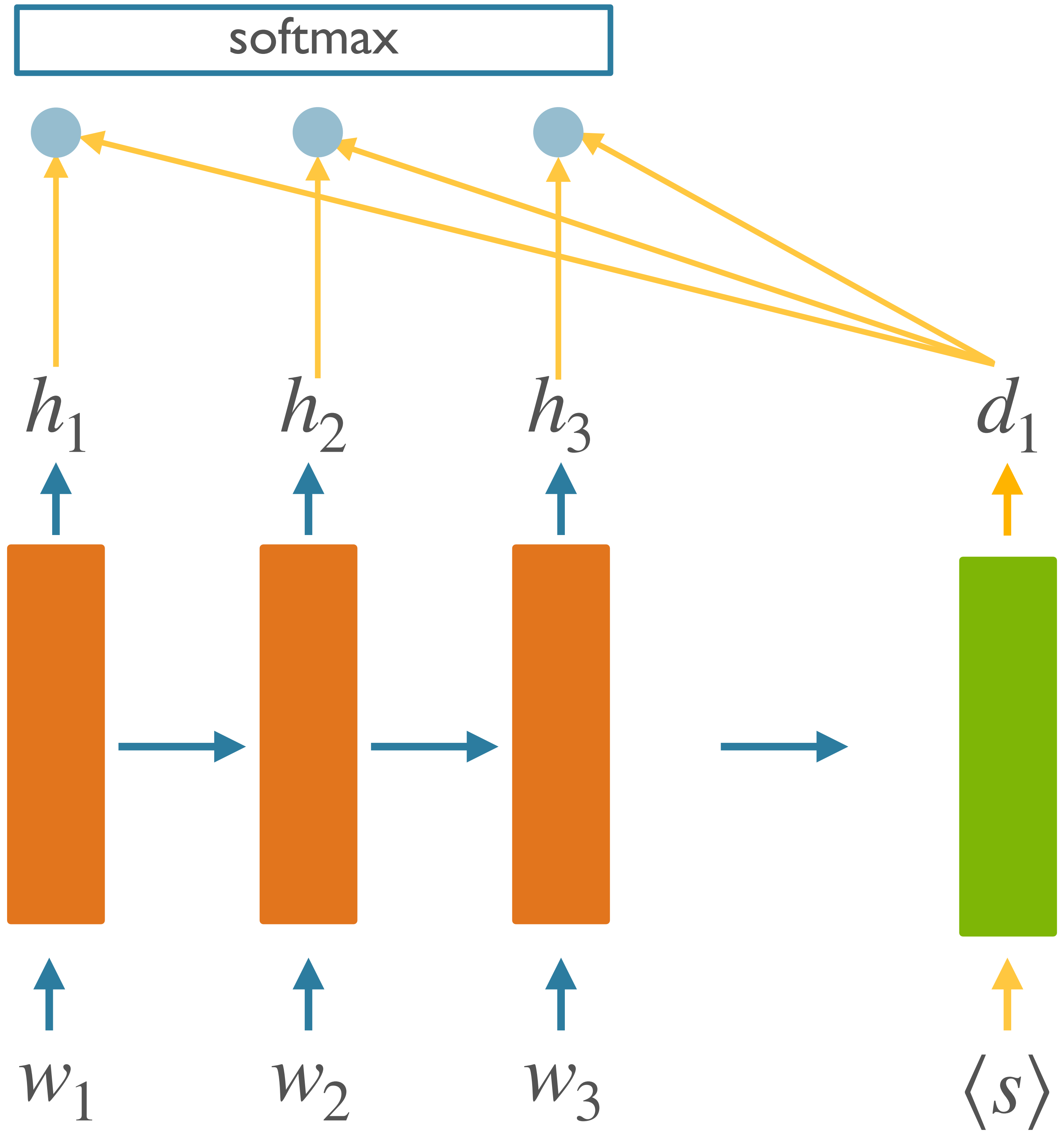


Badhanau et al 2014

Adding Attention

$e_{ij} = \text{softmax}(\alpha)_j$

$\alpha_{ij} = a(h_j, d_i)$
(dot product usually)



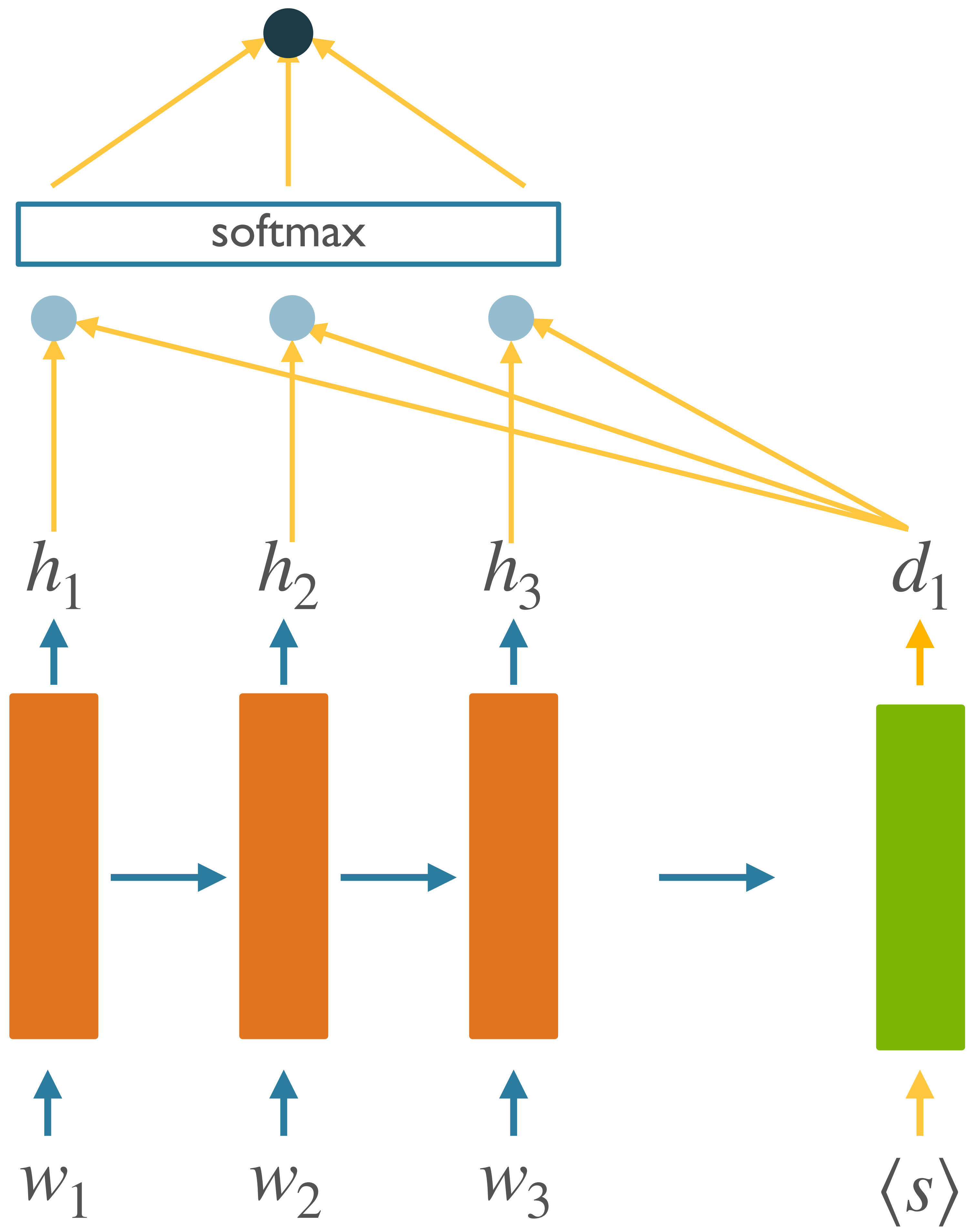
Badhanau et al 2014

Adding Attention

$c_i = \Sigma_j e_{ij} h_j$

$e_{ij} = \text{softmax}(\alpha)_j$

$\alpha_{ij} = a(h_j, d_i)$
(dot product usually)



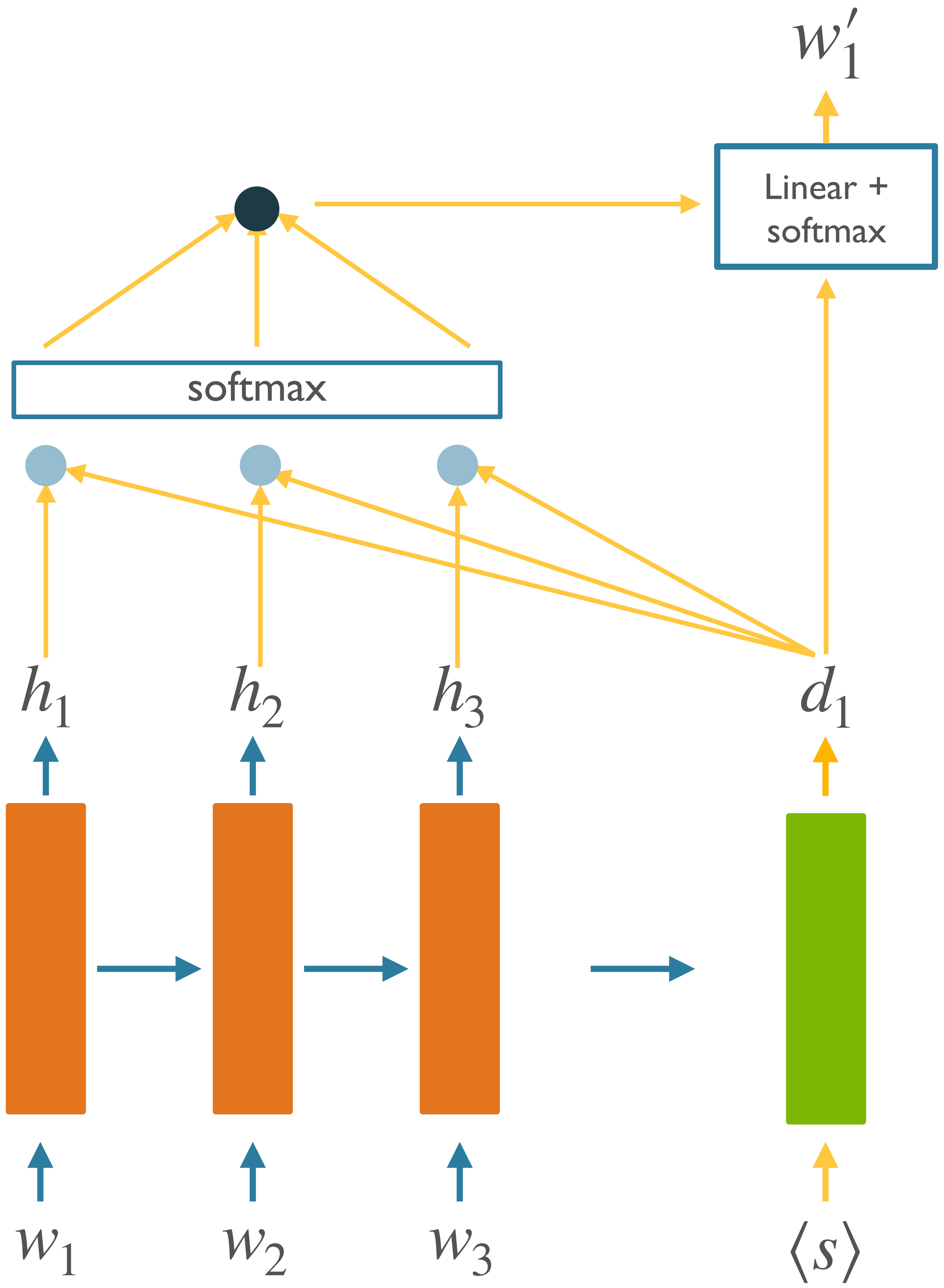
Badhanau et al 2014

Adding Attention

$c_i = \Sigma_j e_{ij} h_j$

$e_{ij} = \text{softmax}(\alpha)_j$

$\alpha_{ij} = a(h_j, d_i)$
(dot product usually)



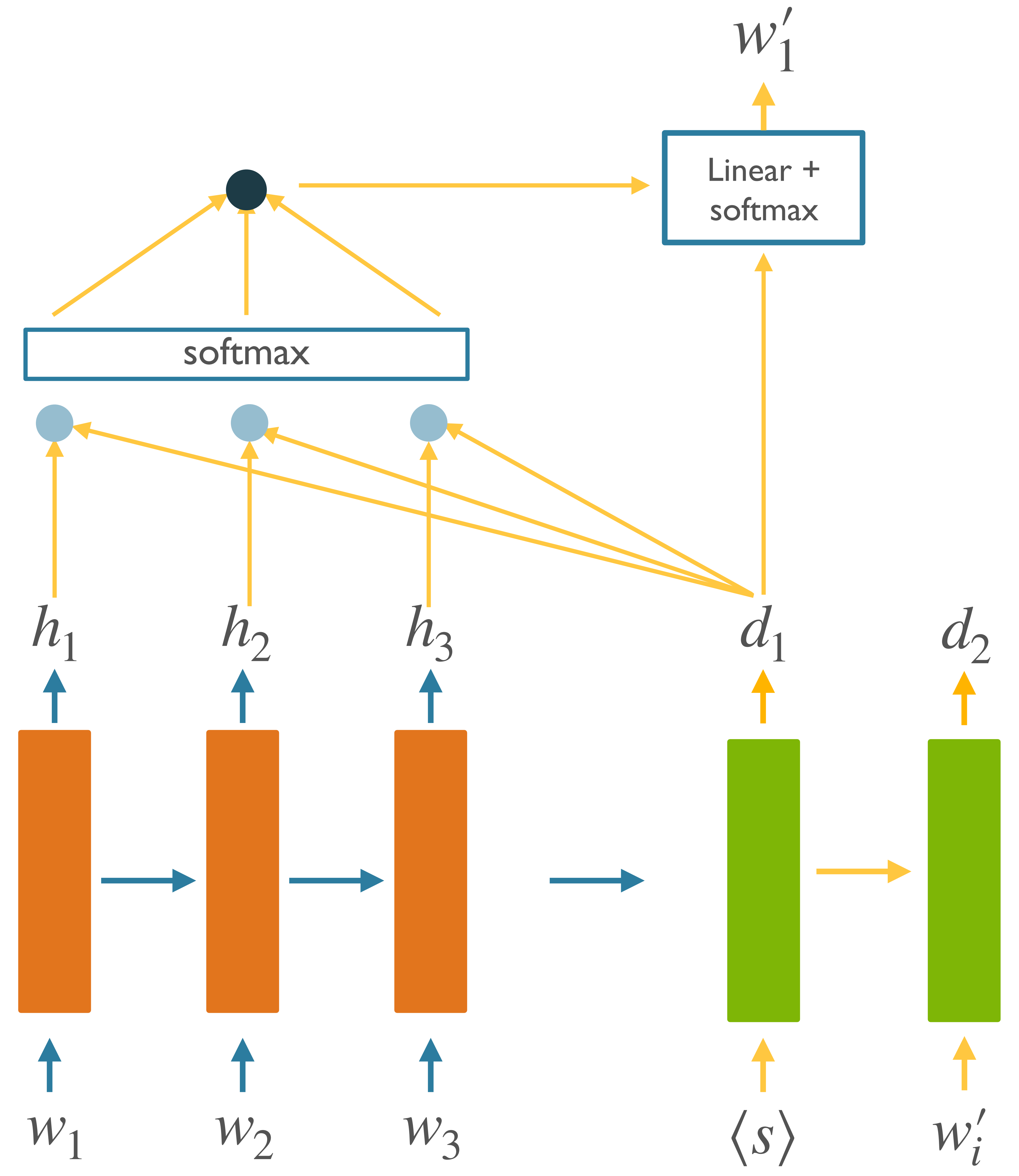
Badhanau et al 2014

Adding Attention

$c_i = \Sigma_j e_{ij} h_j$

$e_{ij} = \text{softmax}(\alpha)_j$

$\alpha_{ij} = a(h_j, d_i)$
(dot product usually)



Badhanau et al 2014

Attention, Generally

Attention, Generally

- A *query* q pays attention to some *values* $\{v_k\}$ based on similarity with some *keys* $\{k_v\}$.

Attention, Generally

- A *query* q pays attention to some *values* $\{v_k\}$ based on similarity with some *keys* $\{k_v\}$.
- Dot-product attention:

$$\alpha_j = q \cdot k_j$$

$$e_j = e^{\alpha_j} / \Sigma_j e^{\alpha_j}$$

$$c = \Sigma_j e_j v_j$$

Attention, Generally

- A *query* q pays attention to some *values* $\{v_k\}$ based on similarity with some *keys* $\{k_v\}$.
- Dot-product attention:

$$\alpha_j = q \cdot k_j$$

$$e_j = e^{\alpha_j} / \Sigma_j e^{\alpha_j}$$

$$c = \Sigma_j e_j v_j$$

- In the previous example: encoder hidden states played *both* the keys and the values roles.

Why attention?

Why attention?

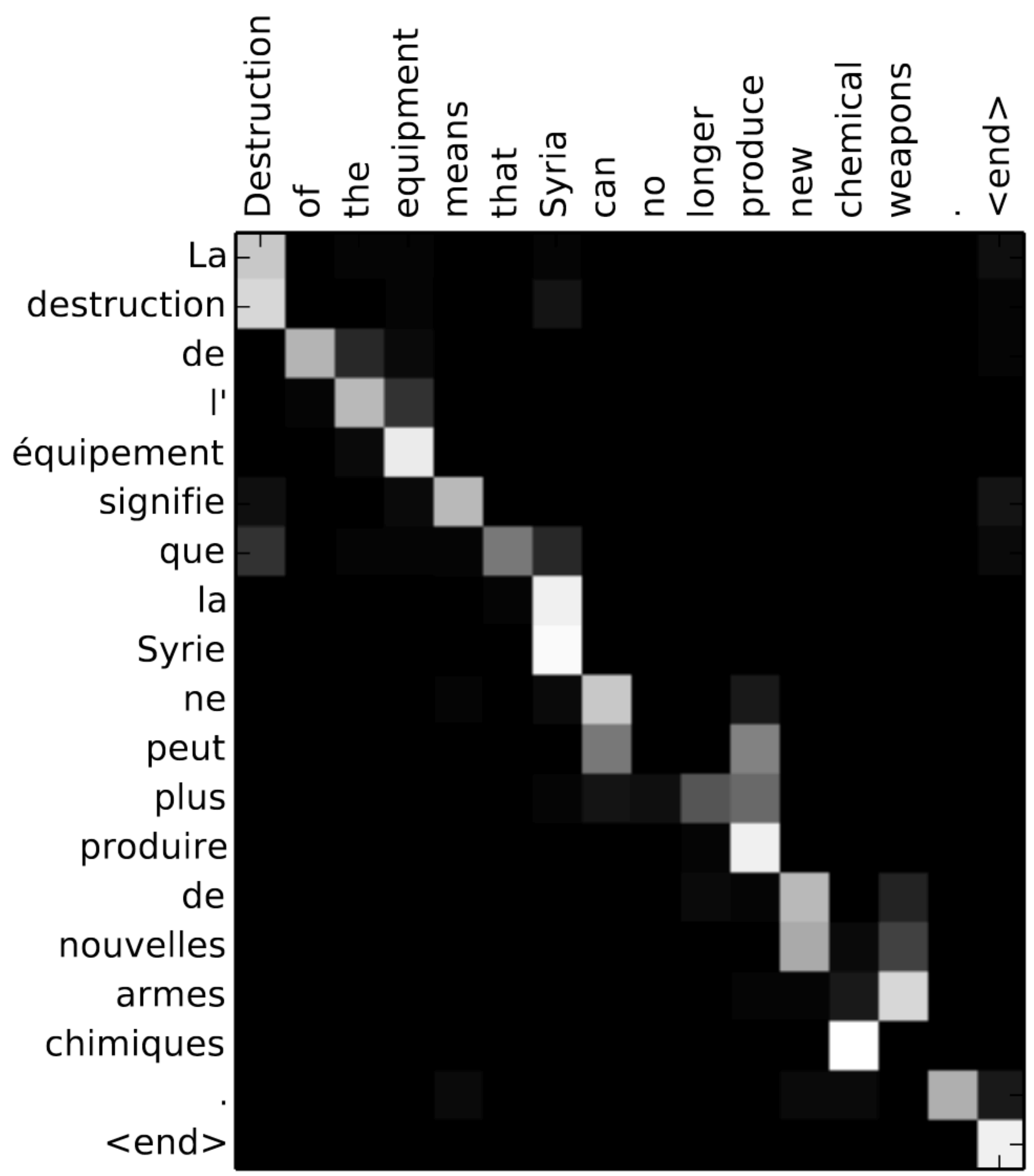
- Incredibly useful (for performance)
  - By “solving” the bottleneck issue

Why attention?

- Incredibly useful (for performance)
  - By “solving” the bottleneck issue
- Aids interpretability:*
  - * some debate; more next week

Why attention?

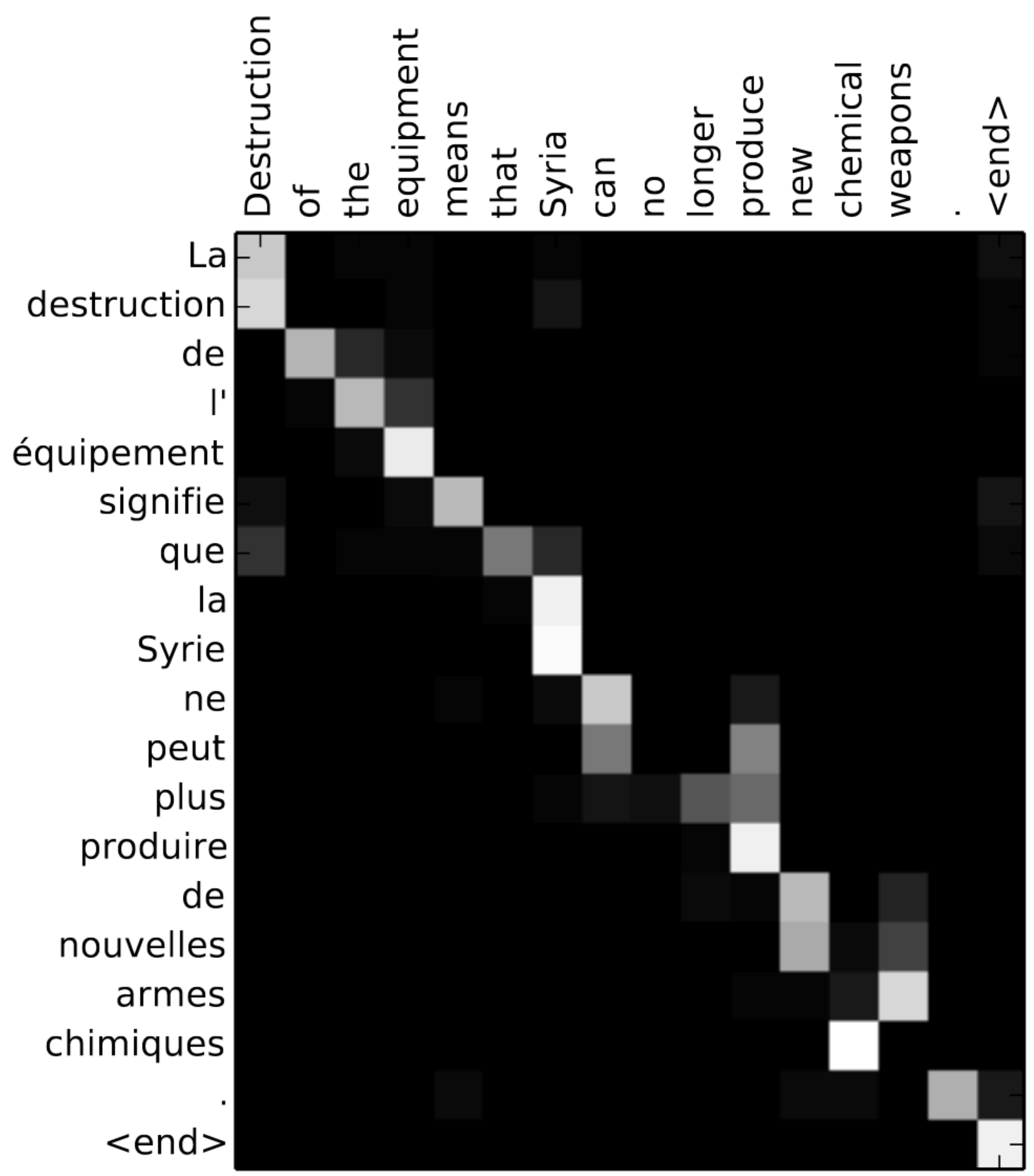
- Incredibly useful (for performance)
  - By “solving” the bottleneck issue
- Aids interpretability:*
  - * some debate; more next week



Badhanau et al 2014

Why attention?

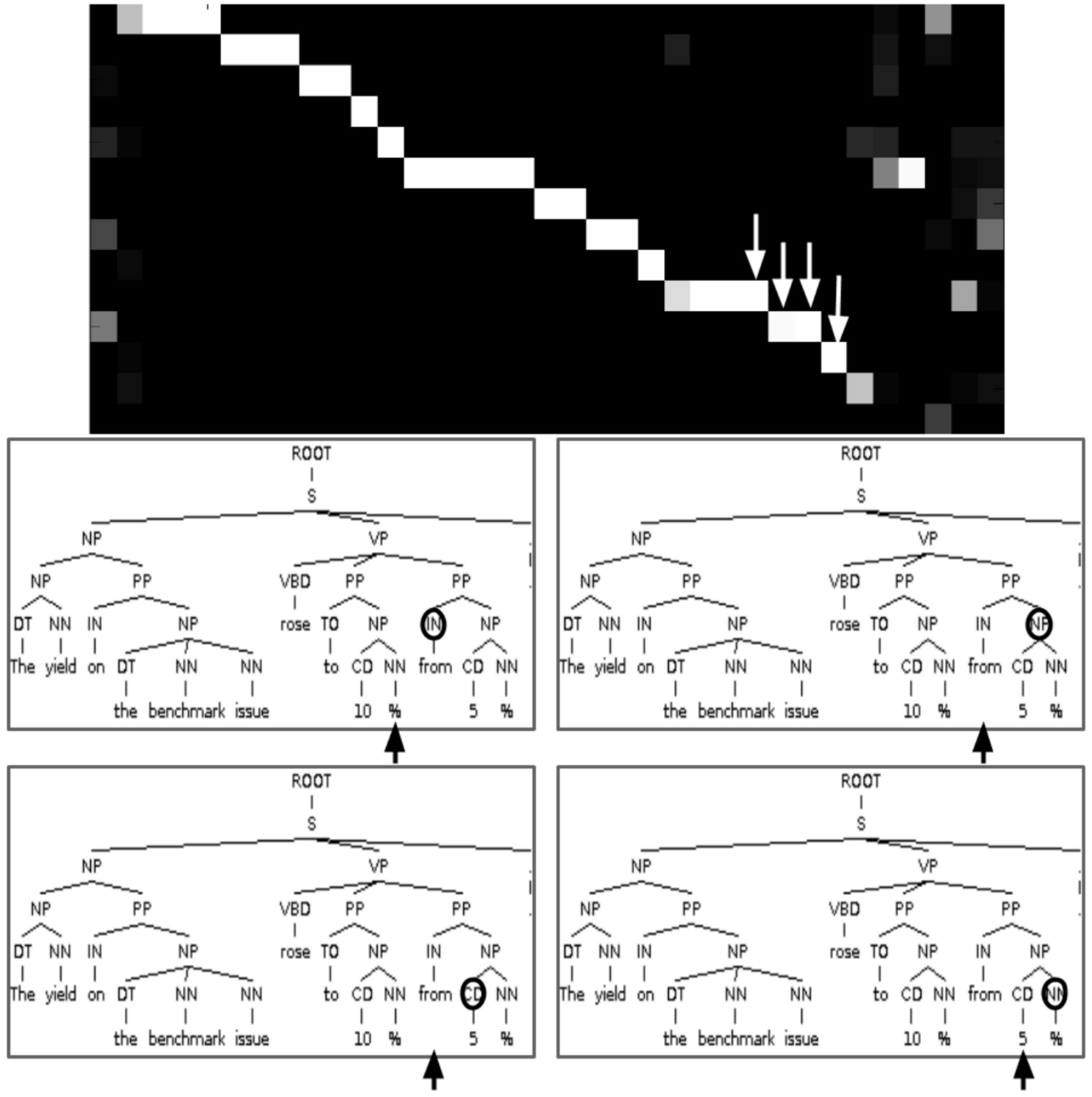
- Incredibly useful (for performance)
  - By “solving” the bottleneck issue
- Aids interpretability:*
  - * some debate; more next week
- A general technique for combining representations, applications in:
  - NMT, parsing, image/video captioning, …



Badhanau et al 2014

Why attention?

- Incredibly useful (for performance)
  - By “solving” the bottleneck issue
- Aids interpretability:*
  - * some debate; more next week
- A general technique for combining representations, applications in:
  - NMT, parsing, image/video captioning, …



[Vinyals et al 2015](#)

Outline

- Background
- Recurrent Neural Networks (LSTMs in particular)
  - ELMo
  - seq2seq + *attention*
- **Transformers**
  - BERT
- Snapshot of the current landscape

Transformer Architecture

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.0 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature.

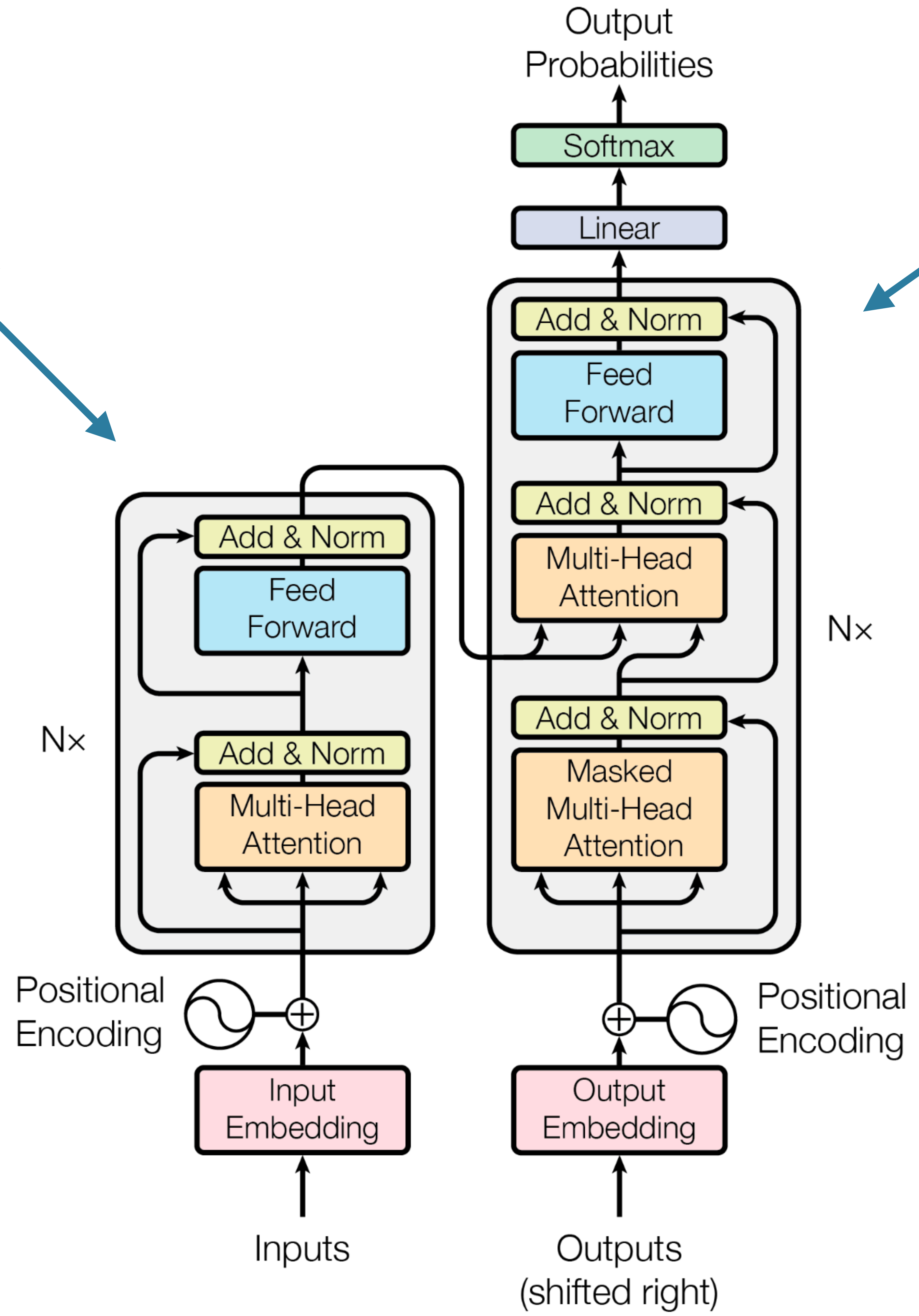
Paper link

(but see Annotated and Illustrated Transformer)

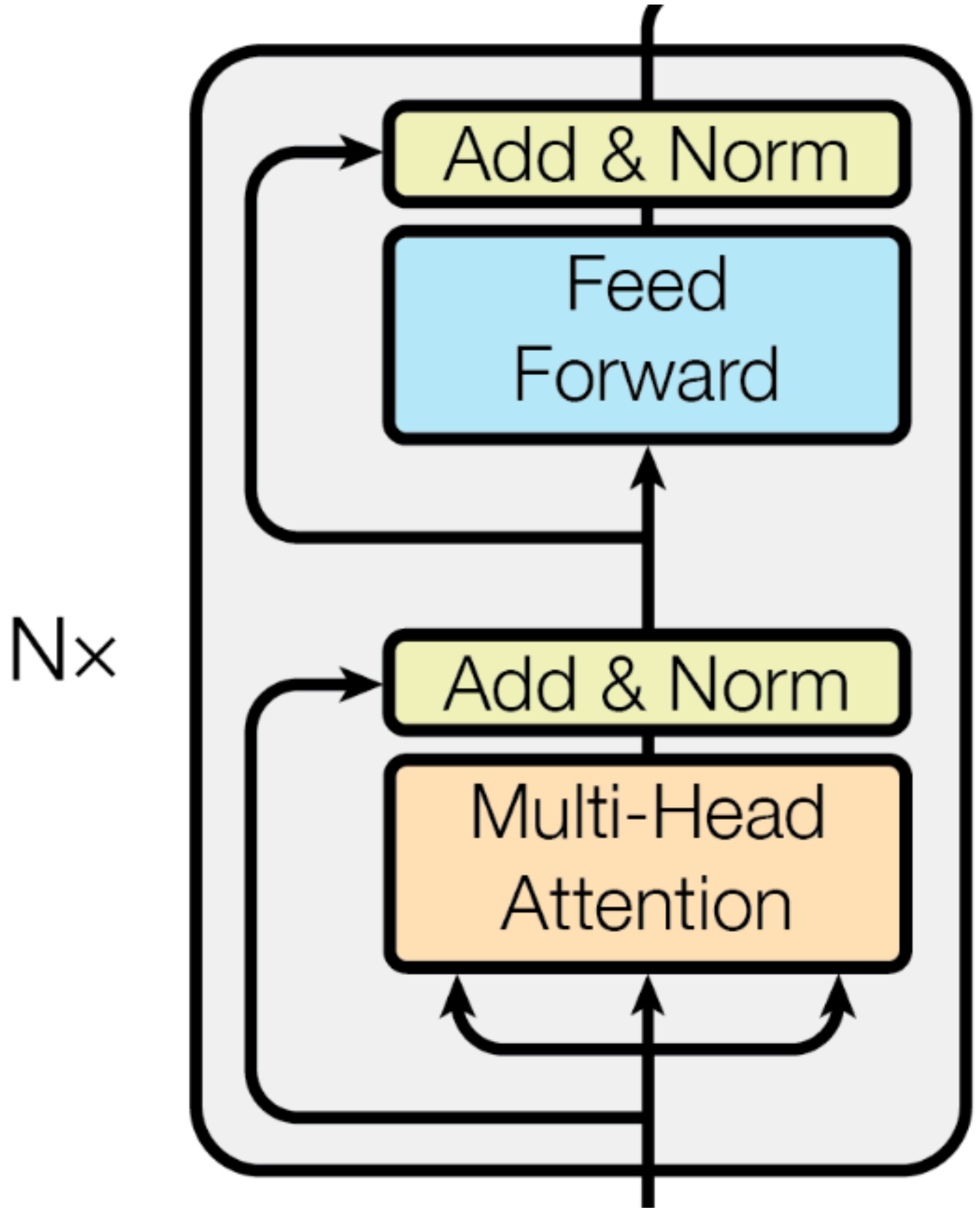
Full Model

encoder

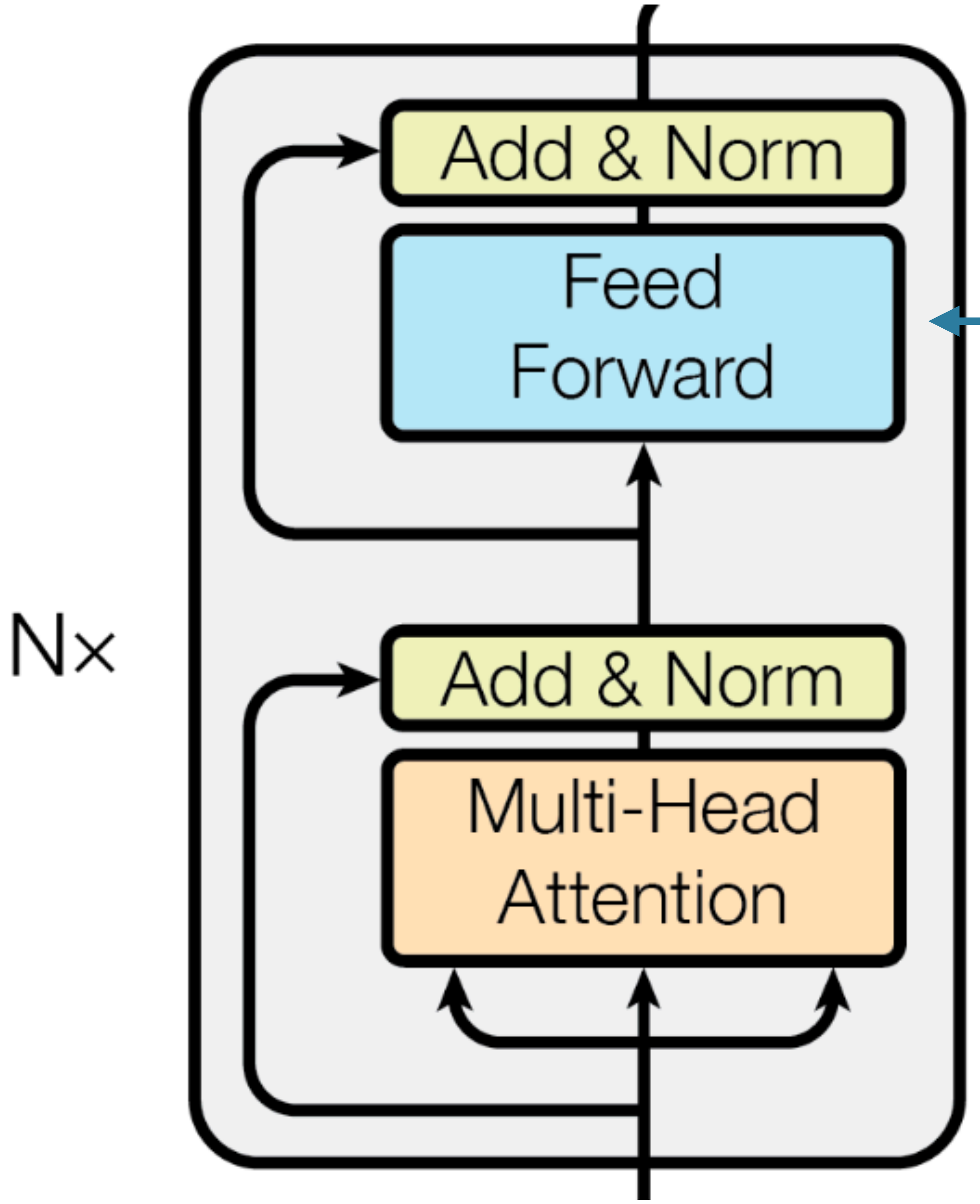
decoder



Transformer Block

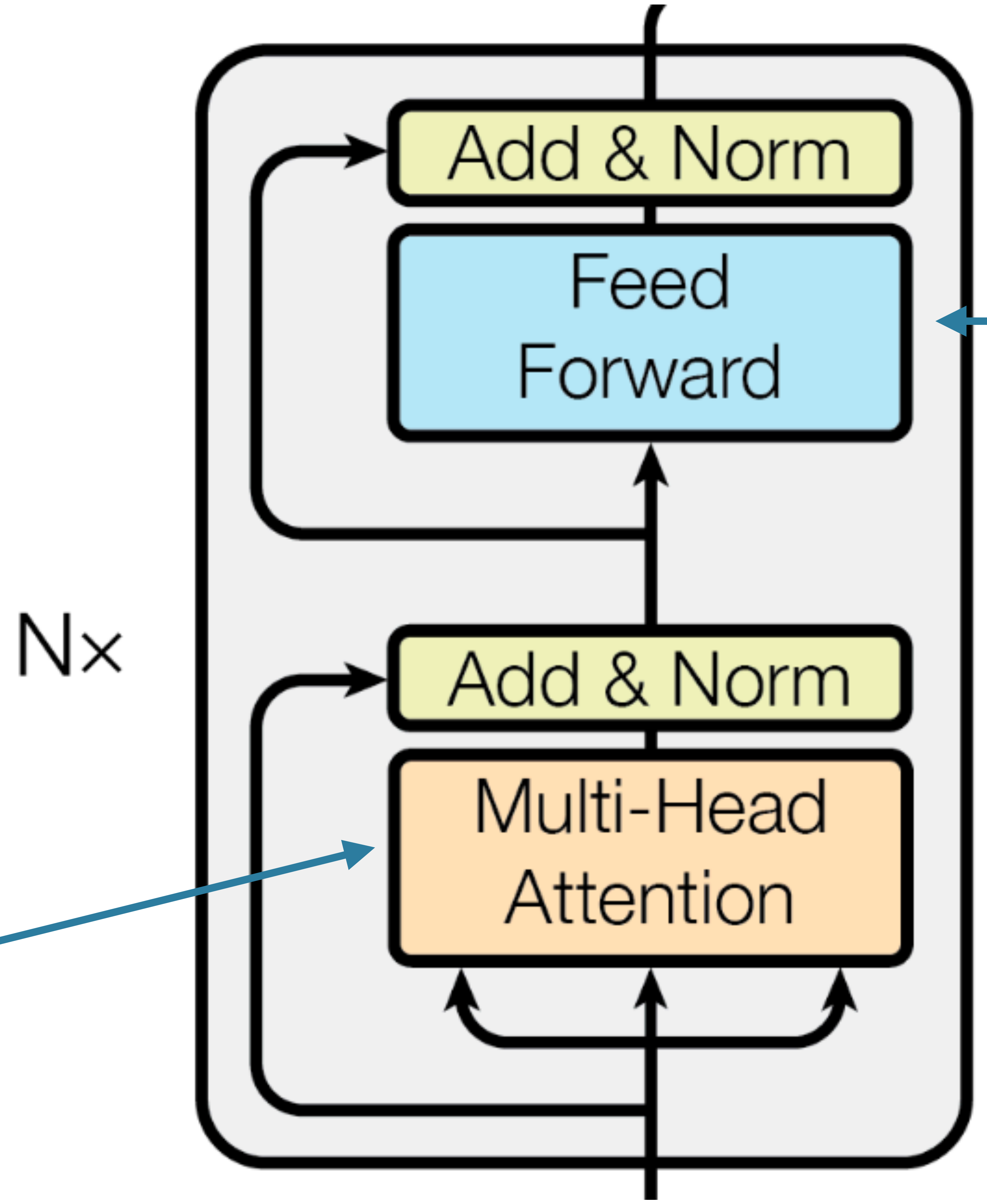


Transformer Block



Single layer, applied to each position

Transformer Block



What's this?

Single layer, applied to each position

Scaled Dot-Product Attention

- Recall:

- Putting it together: (keys/values in matrices) $$\text{Attention}(q, K, V) = \sum_j \frac{e^{q \cdot k_j}}{\sum_i e^{q \cdot k_i}} v_j$$

- Stacking *multiple* queries: (and scaling) $$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V$$

Scaled Dot-Product Attention

- Recall:

$$\alpha_j = q \cdot k_j$$

$$e_j = e^{\alpha_j} / \Sigma_j e^{\alpha_j}$$

$$c = \Sigma_j e_j v_j$$

- Putting it together: (keys/values in matrices)

$$\text{Attention}(q, K, V) = \sum_j \frac{e^{q \cdot k_j}}{\sum_i e^{q \cdot k_i}} v_j$$

- Stacking *multiple* queries: (and scaling)

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V$$

Scaled Dot-Product Attention

- Recall:

$$\alpha_j = q \cdot k_j$$

$$e_j = e^{\alpha_j} / \sum_j e^{\alpha_j}$$

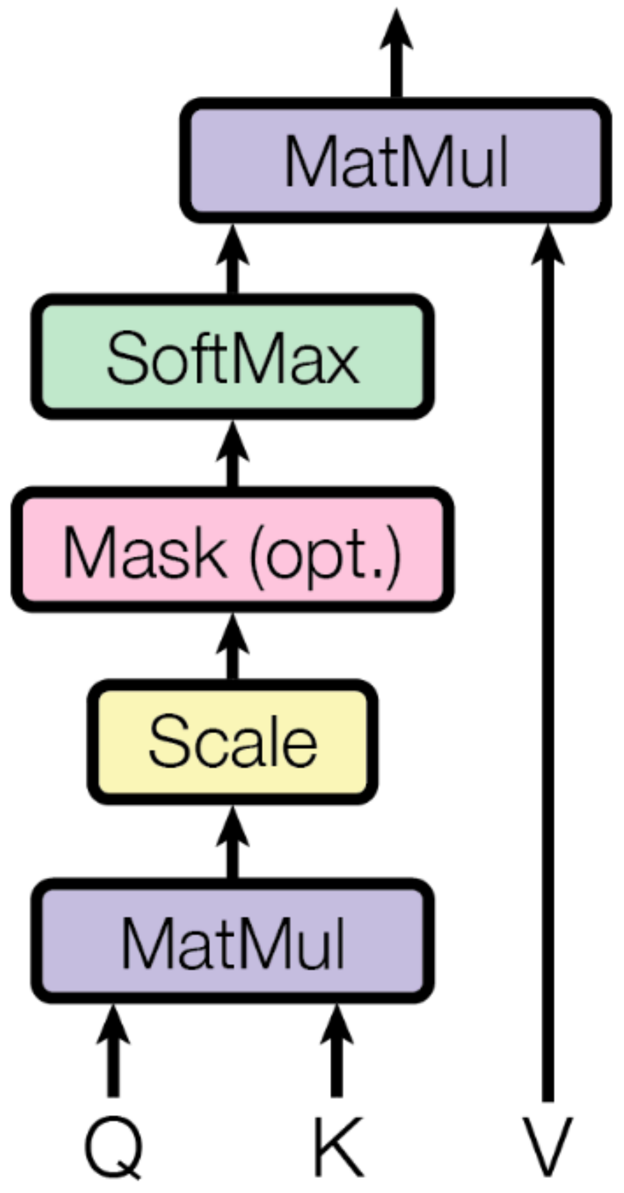
$$c = \sum_j e_j v_j$$

- Putting it together: (keys/values in matrices)

$$\text{Attention}(q, K, V) = \sum_j \frac{e^{q \cdot k_j}}{\sum_i e^{q \cdot k_i}} v_j$$

- Stacking *multiple* queries: (and scaling)

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V$$



Why multiple queries?

Why multiple queries?

- seq2seq: single decoder token attends to *all* encoder states

Why multiple queries?

- seq2seq: single decoder token attends to *all* encoder states
- Transformer: *self*-attention
  - Every (token) position attends to every other position [including self!]
  - Caveat: in the encoder, and only by default
    - Mask in decoder to attend only to previous positions
    - Masking technique applied in some Transformer-based LMs

Why multiple queries?

- seq2seq: single decoder token attends to *all* encoder states
- Transformer: *self*-attention
  - Every (token) position attends to every other position [including self!]
  - Caveat: in the encoder, and only by default
    - Mask in decoder to attend only to previous positions
    - Masking technique applied in some Transformer-based LMs
- So vector at each position is a query
  - And a key, and a value

Multi-headed Attention

- So far: a *single* attention mechanism.
- Could be a bottleneck: need to pay attention to different vectors *for different reasons*
- Multi-headed: several attention mechanisms in parallel

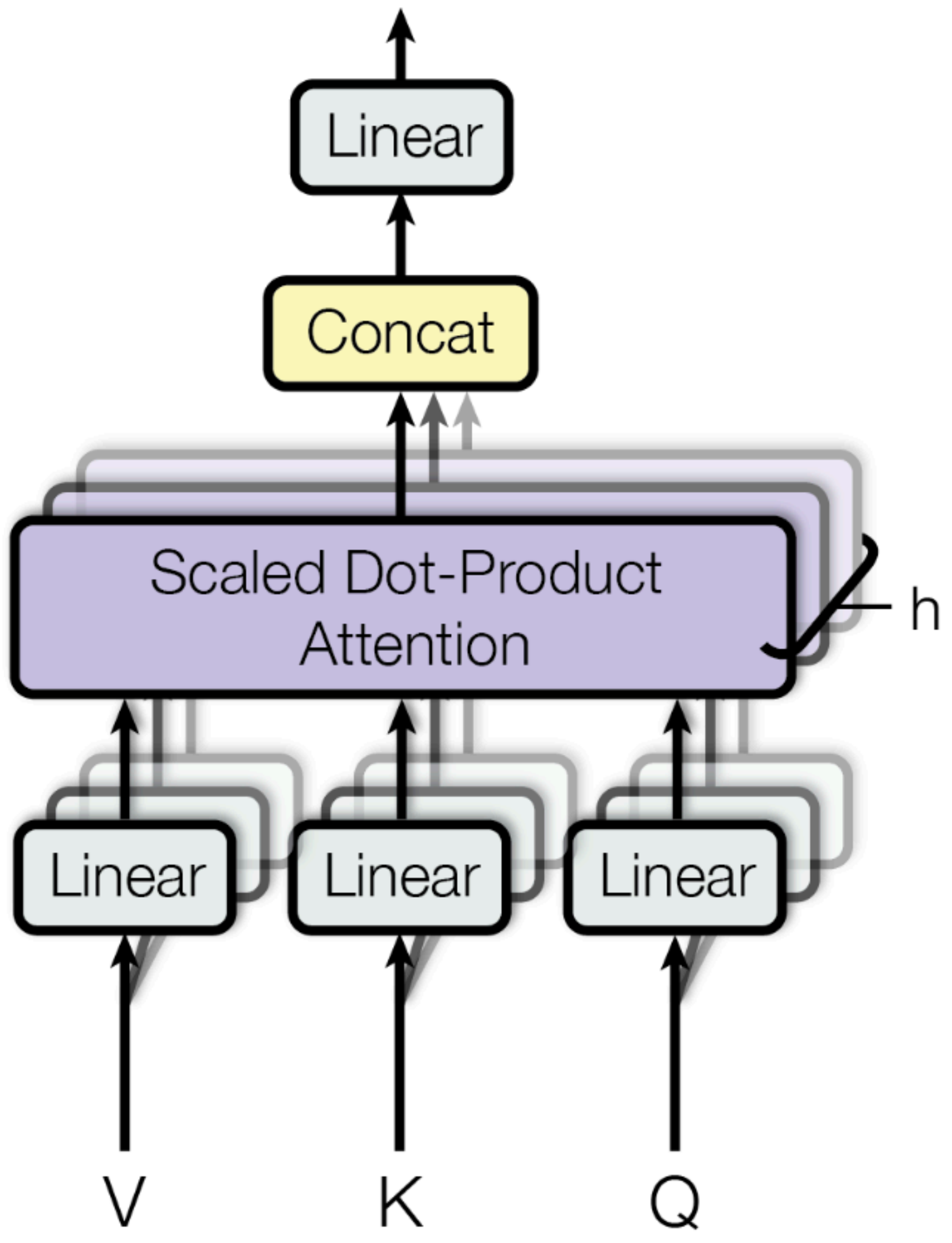
Multi-headed Attention

- So far: a *single* attention mechanism.
- Could be a bottleneck: need to pay attention to different vectors *for different reasons*
- Multi-headed: several attention mechanisms in parallel

$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head_1}, ..., \mathrm{head_h})W^O$$
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Multi-headed Attention

- So far: a *single* attention mechanism.
- Could be a bottleneck: need to pay attention to different vectors *for different reasons*
- Multi-headed: several attention mechanisms in parallel



$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head_1}, \ldots, \mathrm{head_h})W^O$$
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

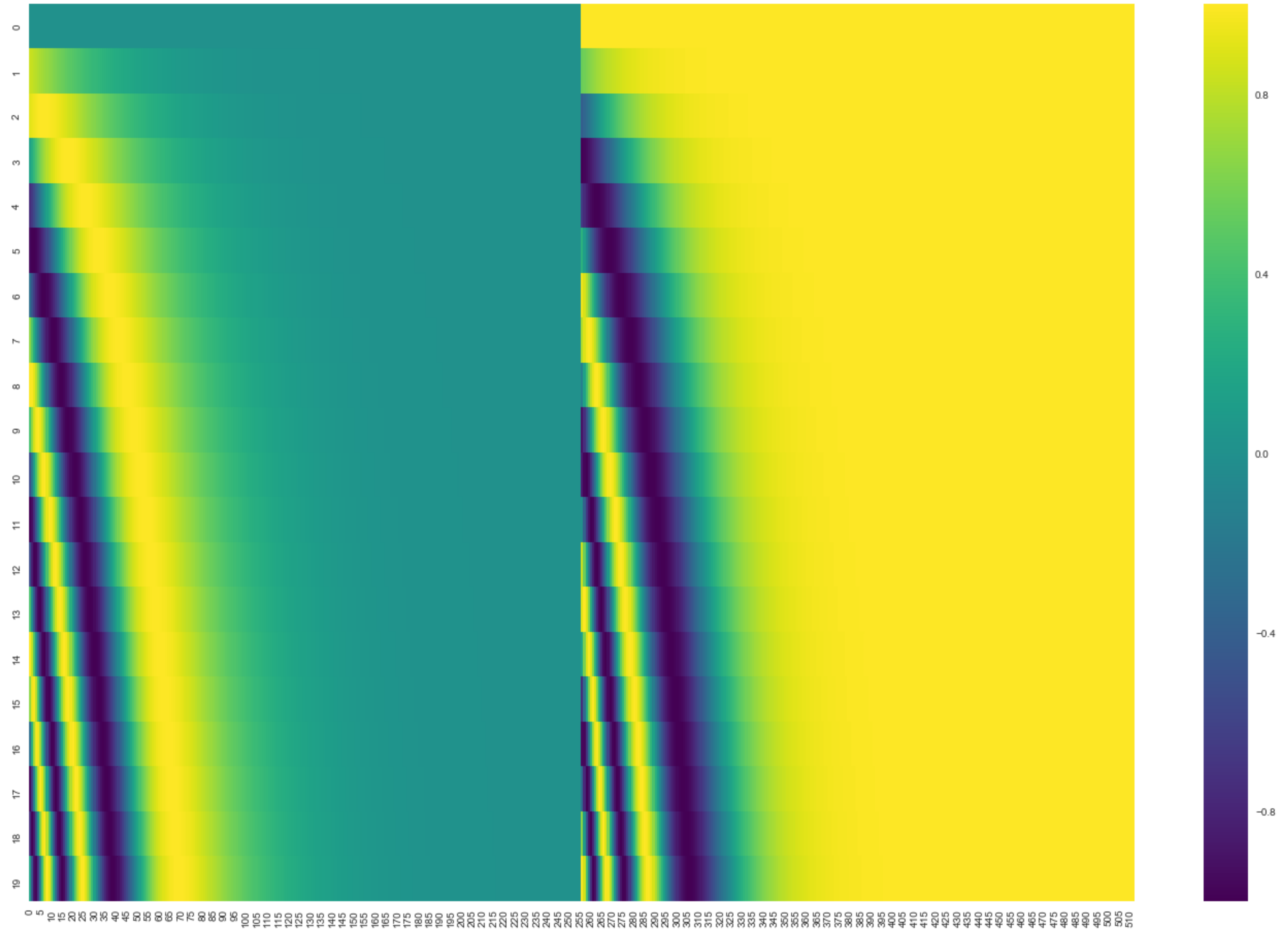
Representing Order

Representing Order

- No notion of order in Transformer. Represented via *positional* encodings.

Representing Order

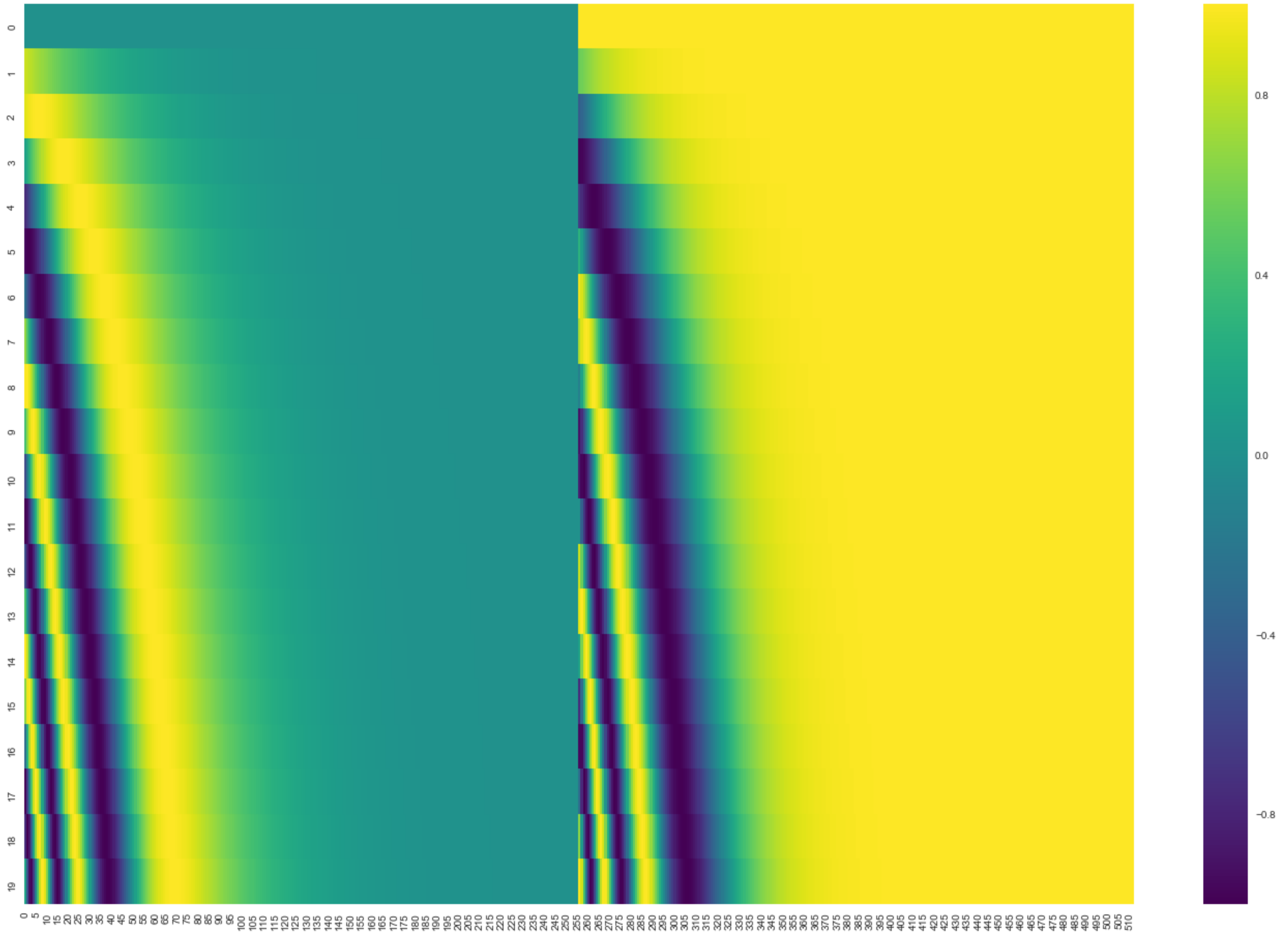
- No notion of order in Transformer. Represented via *positional* encodings.



source

Representing Order

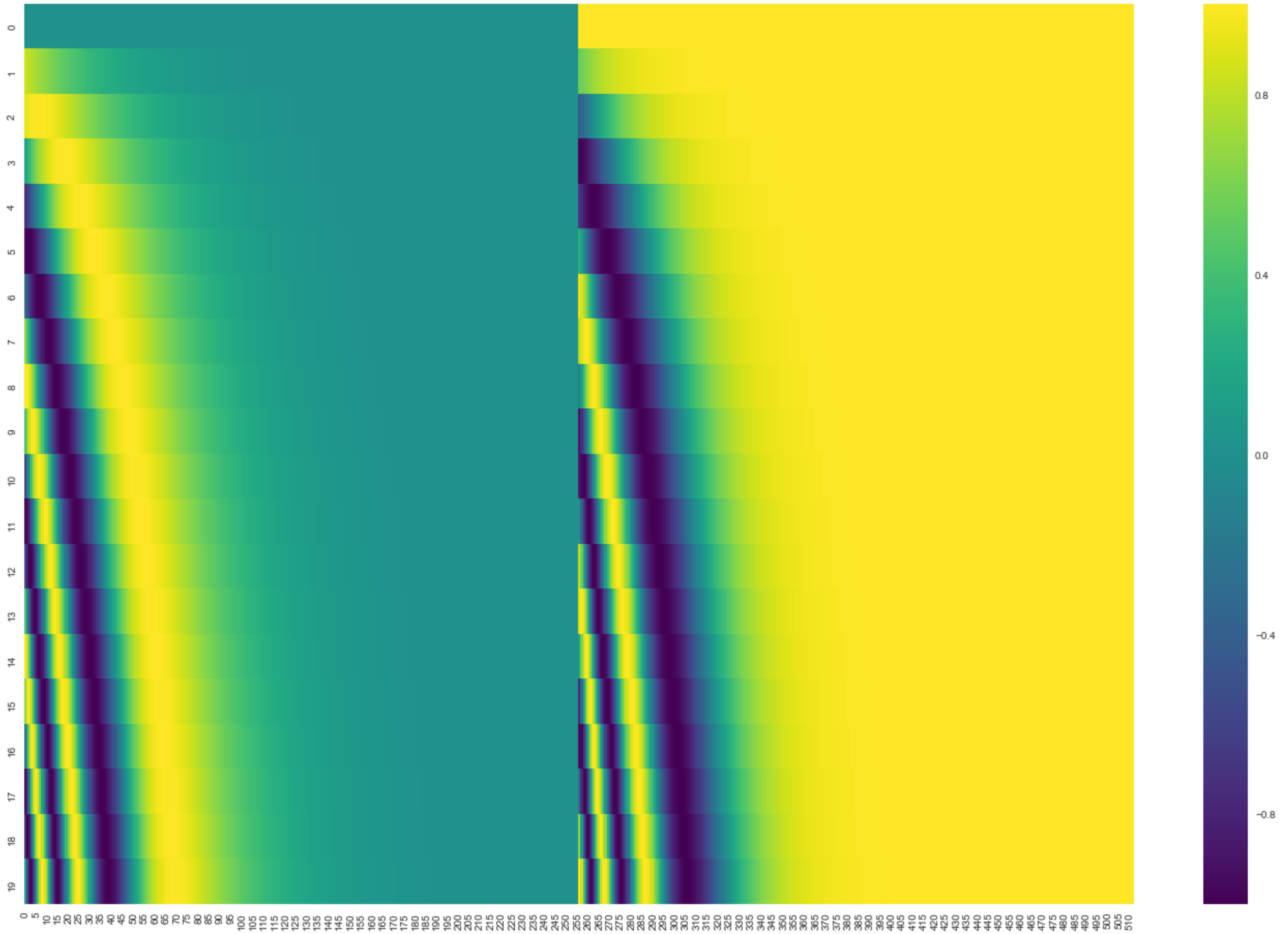
- No notion of order in Transformer. Represented via *positional* encodings.
- Usually fixed, though can be learned.



source

Representing Order

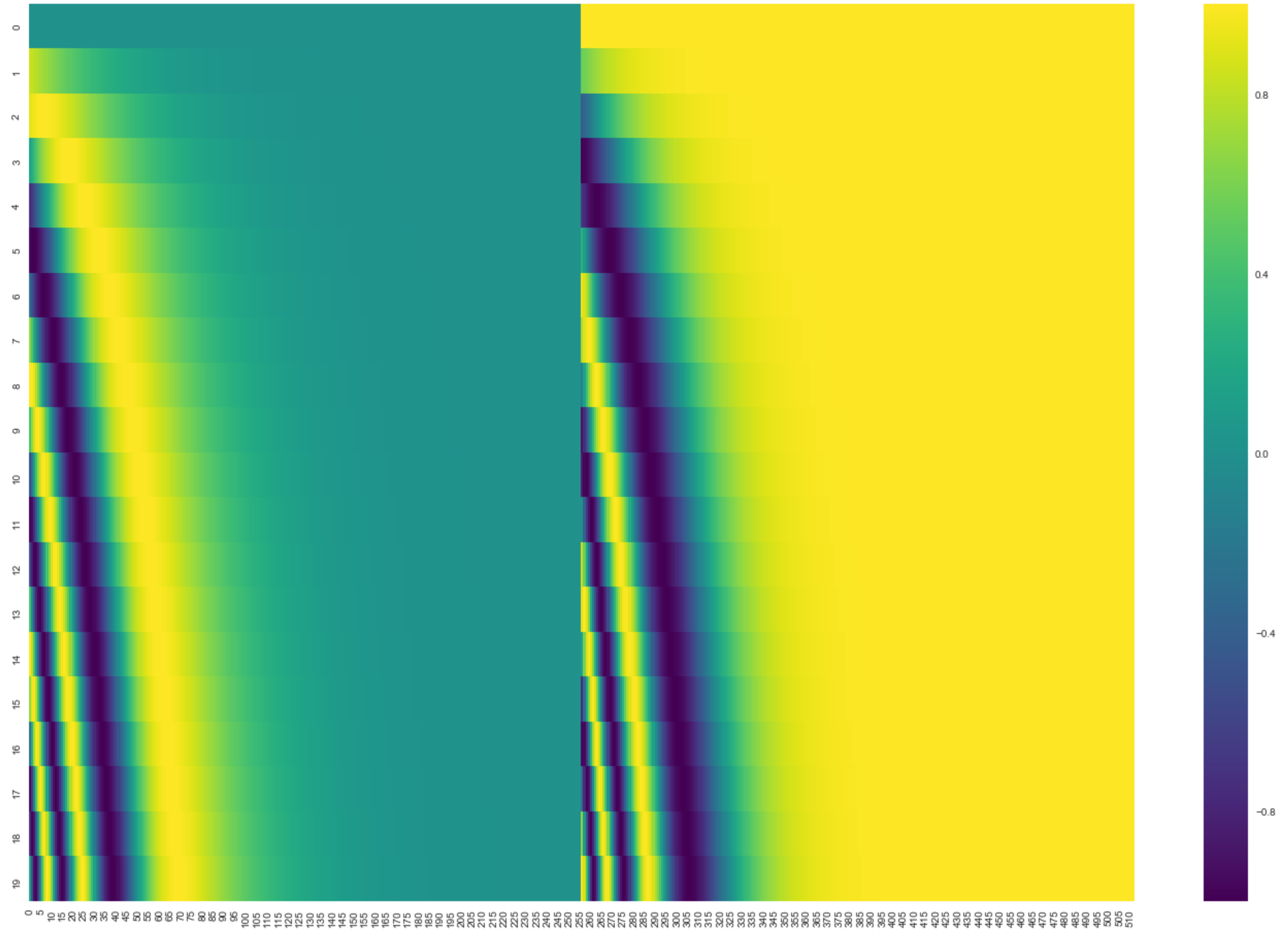
- No notion of order in Transformer. Represented via *positional* encodings.
- Usually fixed, though can be learned.
  - No significant improvement; less generalization.



source

Representing Order

- No notion of order in Transformer. Represented via *positional* encodings.
- Usually fixed, though can be learned.
  - No significant improvement; less generalization.
- [Not necessary in certain Transformer LM contexts 😱; more on this later]



source

Initial WMT Results

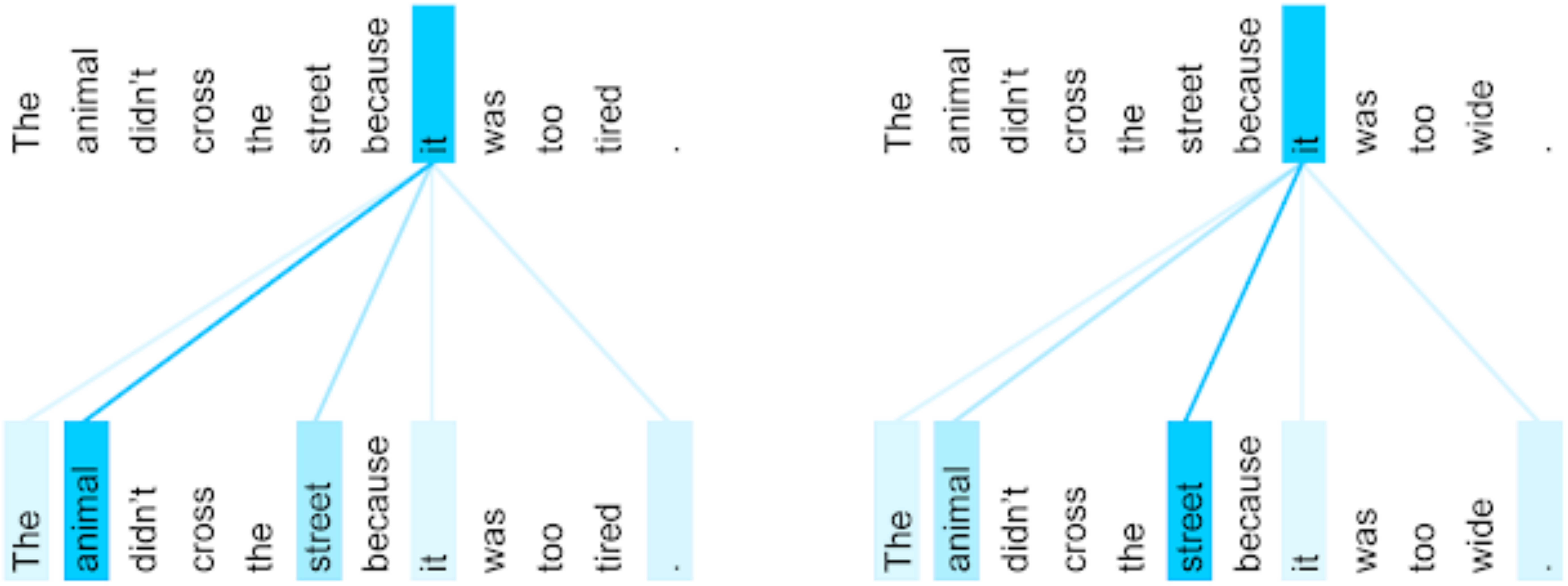
Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [15]	23.75			
Deep-Att + PosUnk [32]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [31]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [8]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [26]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [32]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [31]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [8]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.0**	$2.3 \cdot 10^{19}$	

Initial WMT Results

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [15]	23.75			
Deep-Att + PosUnk [32]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [31]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [8]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [26]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [32]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [31]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [8]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.0**	$2.3 \cdot 10^{19}$	

More on why important later

Attention Visualization: Coreference?

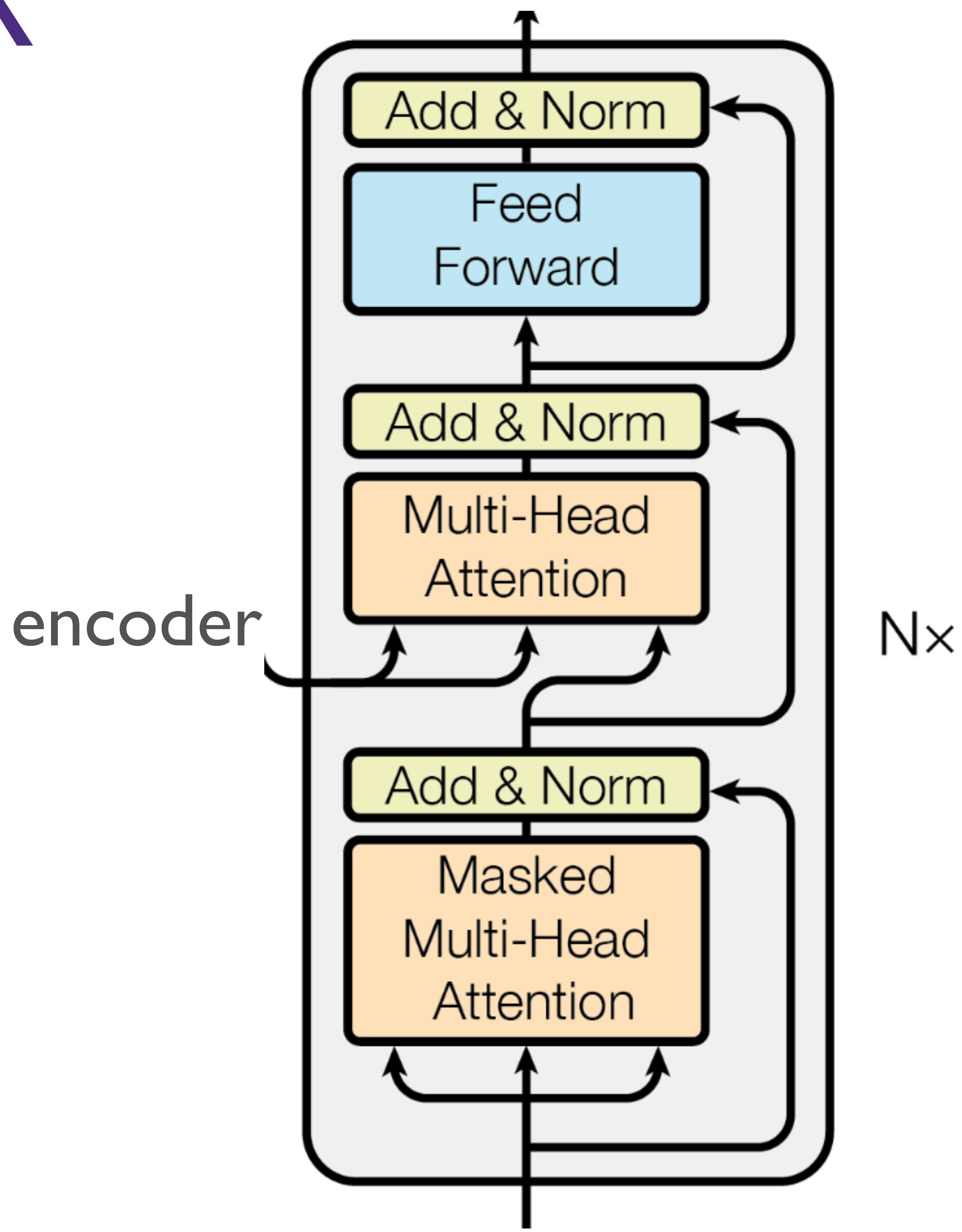


source

Transformer Decoder

Decoder Block

- Like the encoder, the decoder is many *blocks* stacked vertically
- Two slightly different ingredients:
  - *Masked* self-attention
  - Cross (encoder-decoder) attention

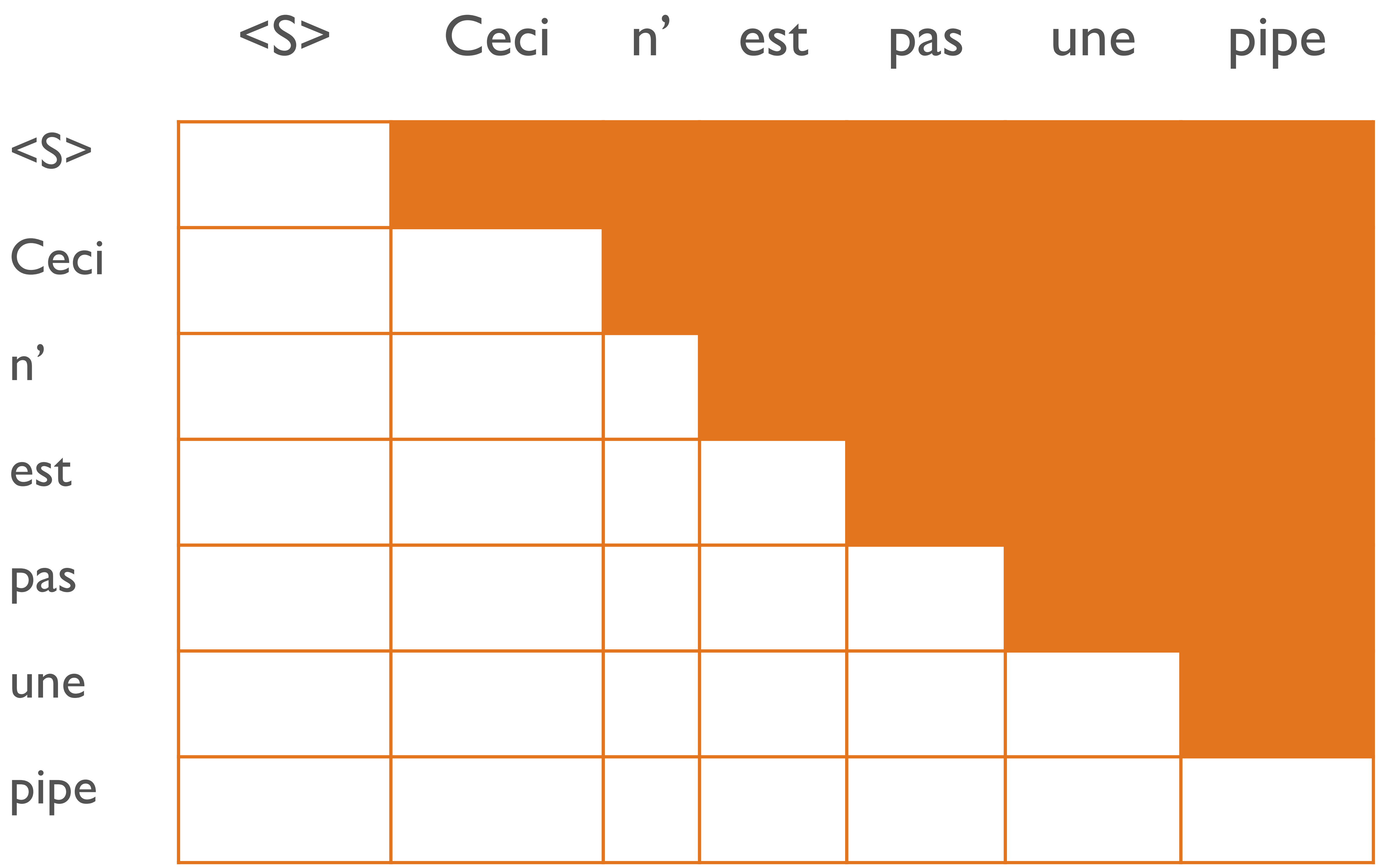


Masked Self-Attention

- Recall from seq2seq:
  - Decoder a kind of *conditional* language model
  - Predicts next tokens in output sequence, *given* the encoder representations
  - [Can also be used on its own as an unconditional LM; more later]
- Problem: self-attention "looks to the future"
  - Decoders should only be able to pay attention to *previous* positions

Masking Out the Future

- Key idea:
  - Use a “mask” to block out certain attention scores
- On the left:
  - Tokens in the rows (as queries) can *not* pay attention to the tokens in the columns (values) that are shaded in



Masking Out the Future

QK^T: total attention scores

$$\text{mask}_{ij} = \begin{cases} -\infty & j > i \\ 0 & \text{otherwise} \end{cases}$$

$$\text{MaskedAttention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}} + \text{mask}\right)V$$

	<S>	Ceci	n’	est	pas	une	pipe
<S>	0	-inf	-inf	-inf	-inf	-inf	-inf
Ceci	0	0	-inf	-inf	-inf	-inf	-inf
n’	0	0	0	-inf	-inf	-inf	-inf
est	0	0	0	0	-inf	-inf	-inf
pas	0	0	0	0	0	-inf	-inf
une	0	0	0	0	0	0	-inf
pipe	0	0	0	0	0	0	0

Masked Self-Attention

- In a nutshell:
  - Compute “raw” attention scores as before
  - Add a mask to “zero out” the future positions in a sequence
- As in the encoder:
  - This is one attention *head*, several used for multi-headed attention
  - Q, K, V are generated by applying learned matrices for each head

Cross-Attention

- Recall the original application of attention: allowing a decoder to attend to *all* of an encoder's representations, instead of just the final one
- How can we apply this form in Transformer-land?
  - What are the queries, keys, and values?

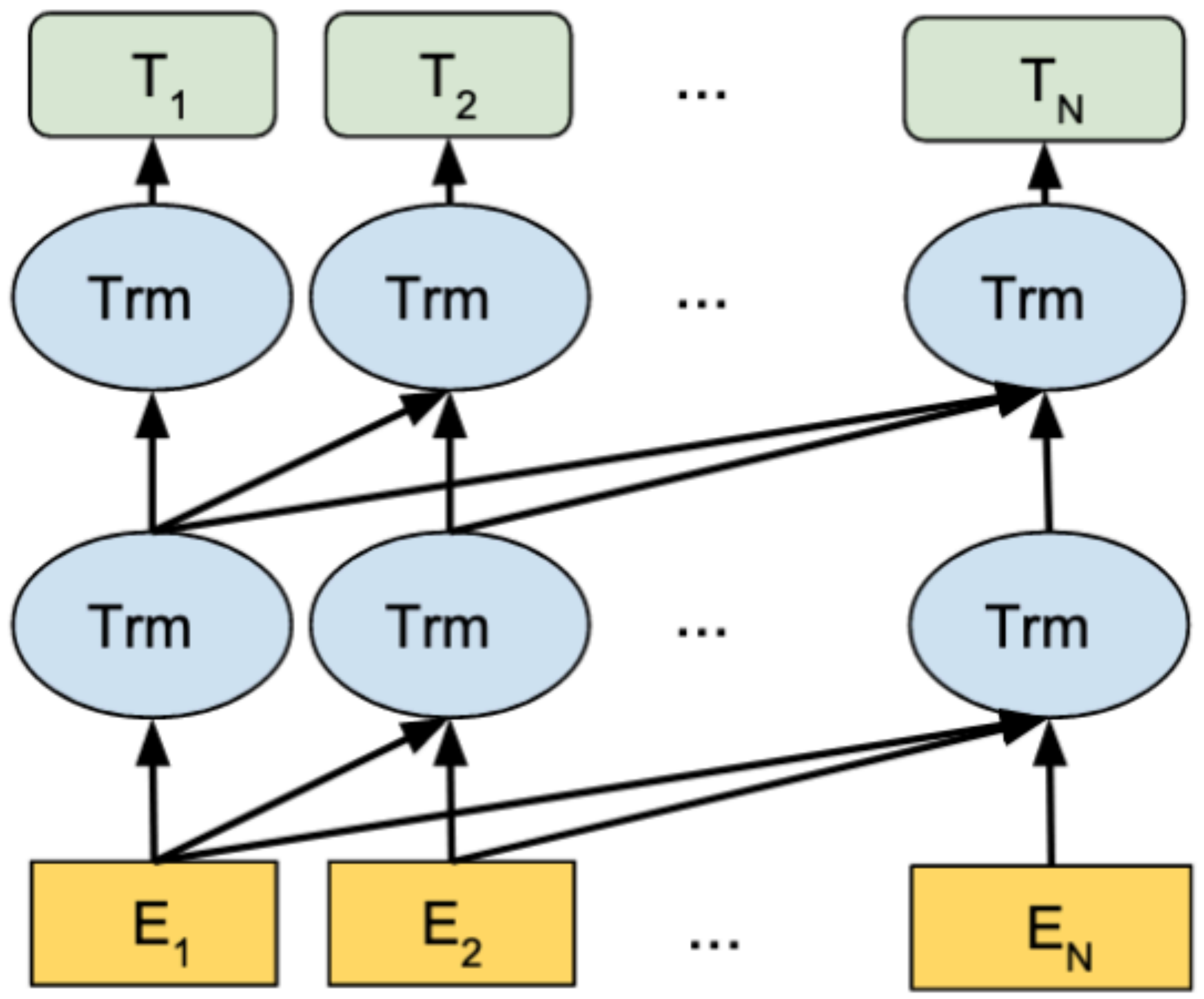
Cross-Attention

- Queries: decoder representations *X*
- Keys and values: top-layer encoder representations *Z*
- Learned weight matrices W_q, W_k, W_v as before

$$\text{CrossAttention} = \text{Attention}\left(XW_q, ZW_k, ZW_v\right)$$

Transformer Decoders

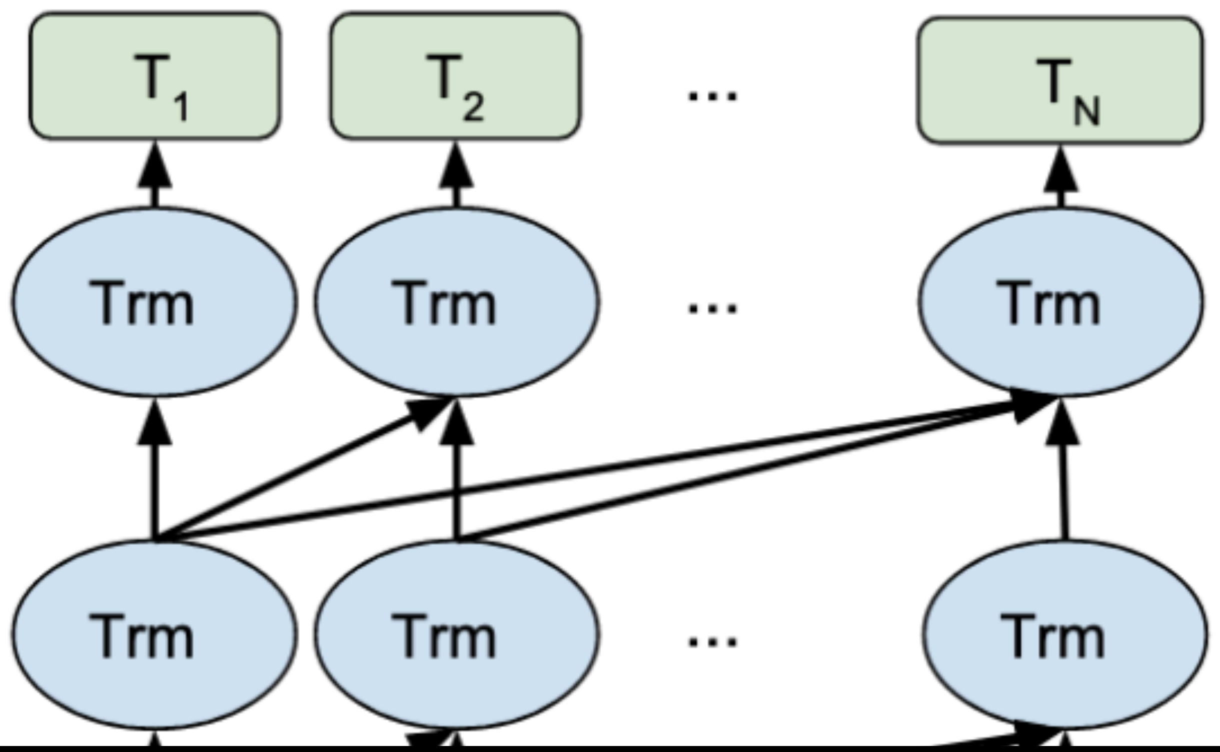
- Can be used any place you would use a decoder
- Masked attention prevents “peeking into the future”
- In seq2seq, for conditional language modeling, e.g.
  - Translation
  - Summarization
- On its own, as a “pure” language model
  - [NB: people now call this “causal language modeling” sometimes]



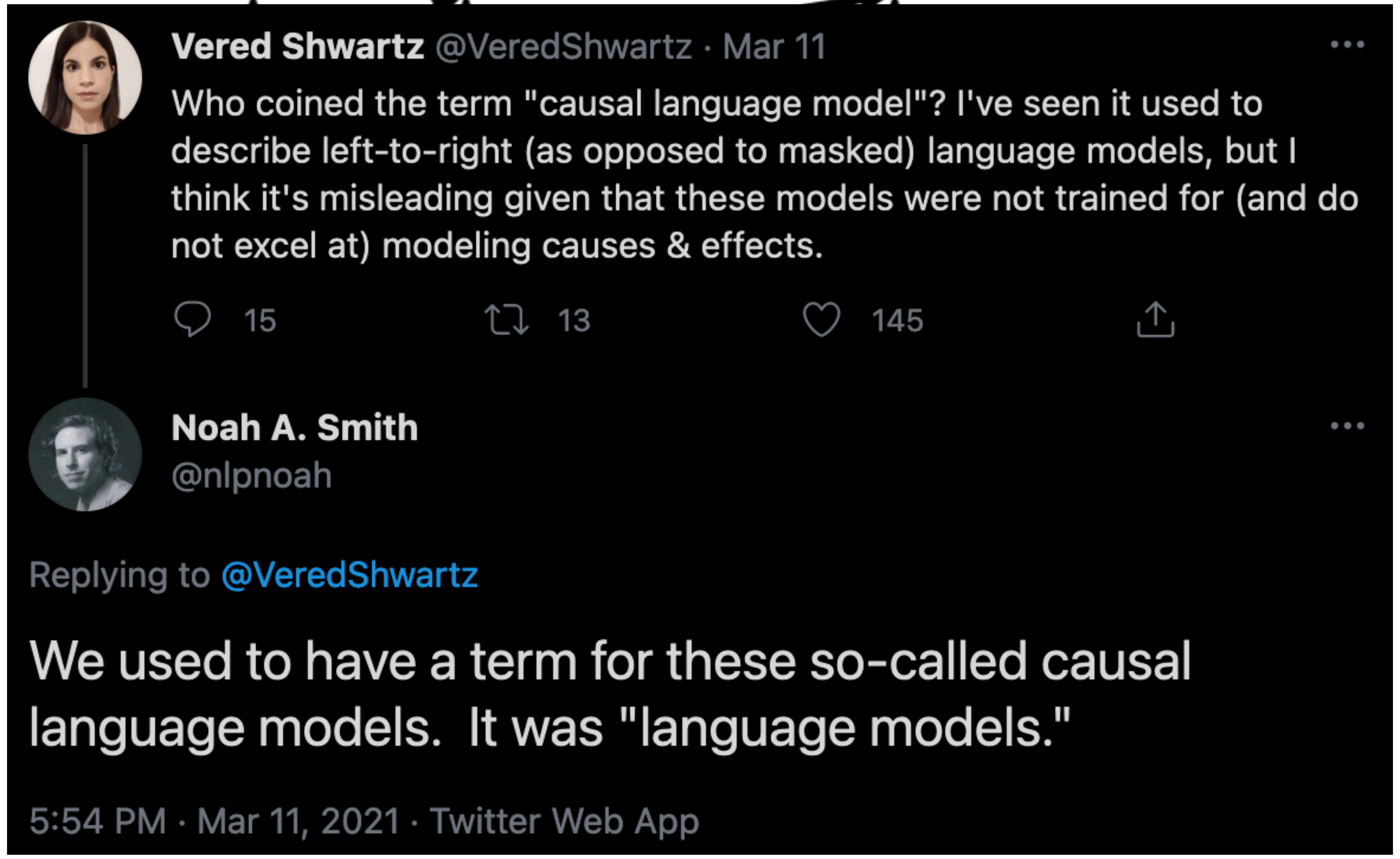
source

Transformer Decoders

- Can be used any place you would use a decoder
- Masked attention prevents "peeking into the future"
- In seq2seq, for conditional language modeling, e.g.
  - Translation
  - Summarization
- On its own, as a "pure" language model
  - [NB: people now call this "causal language modeling" sometimes]



source



Transformer LM (Decoder-only) Results

- Character-level:
  - NB: used several auxiliary losses

Model	Parameters ($\times 10^6$) train	inference	bpc
LSTM (Cooijmans et al. 2016)	-	-	1.43
BN-LSTM (Cooijmans et al. 2016)	-	-	1.36
HM-LSTM (Chung, Ahn, and Bengio 2016)	35	35	1.29
Recurrent Highway (Zilly et al. 2016)	45	45	1.27
mLSTM (Krause et al. 2016)	45	45	1.27
T12 (ours)	44	41	**1.18**
T64 (ours)	235	219	**1.13**
mLSTM + dynamic eval (Krause et al. 2017)	45	-	1.19

- GPT2 results
  - Zero-shot evaluation: trained on very large corpus, evaluated on standard benchmarks

	WikiText2 (PPL)	PTB (PPL)	enwik8 (BPB)	text8 (BPC)	WikiText103 (PPL)	1BW (PPL)
SOTA	39.14	46.54	0.99	1.08	18.3	**21.8**
117M	**29.41**	65.85	1.16	1.17	37.50	75.20
345M	**22.76**	47.33	1.01	**1.06**	26.37	55.72
762M	**19.93**	**40.31**	**0.97**	**1.02**	22.05	44.575
1542M	**18.34**	**35.76**	**0.93**	**0.98**	**17.48**	42.16

Transformer: Summary

- *Entirely* feed-forward
  - Therefore massively parallelizable
  - RNNs are inherently sequential, a parallelization bottleneck
- (Self-)attention everywhere
- Long-term dependencies:
  - LSTM: has to maintain representation of early item
  - Transformer: very short "path-lengths"

BERT: Bidirectional Encoder Representations from Transformers

Devlin et al NAACL 2019

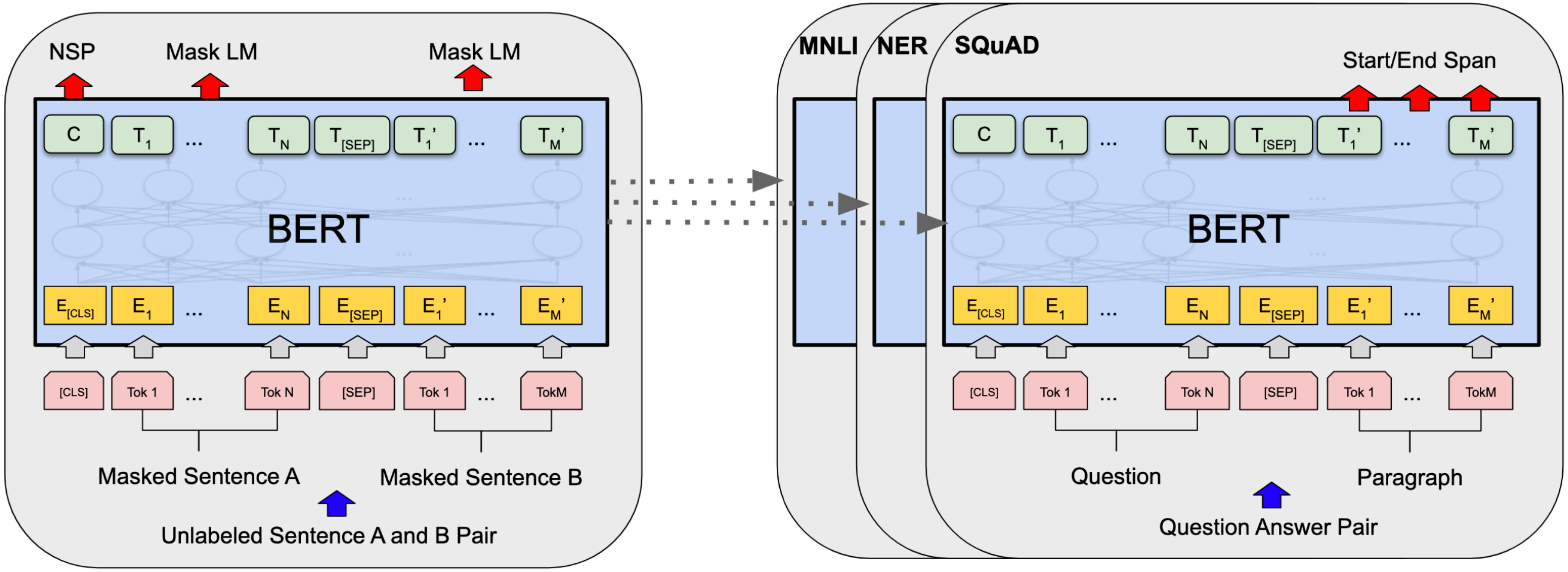
Overview

- Encoder Representations from Transformers: ✓
- Bidirectional: ………?
  - BiLSTM (ELMo): left-to-right and right-to-left
  - Self-attention: every token can see every other
- How do you treat the encoder as an LM (as computing $P(w_t \mid w_{t-1}, w_{t-2}, \ldots, w_1)$)?
  - Don't: modify the task

Masked Language Modeling

- Language modeling: next word prediction
- *Masked* Language Modeling (a.k.a. cloze task): fill-in-the-blank
  - Nancy Pelosi sent the articles of ____ to the Senate.
  - Seattle ____ some snow, so UW was delayed due to ____ roads.
- I.e. $P(w_t \mid w_{t+k}, w_{t+(k-1)}, \ldots, w_{t+1}, w_{t-1}, \ldots, w_{t-(m+1)}, w_{t-m})$
  - (very similar to CBOW: continuous bag of words from word2vec)
- Auxiliary training task: next sentence prediction.
  - Given sentences A and B, binary classification: did B follow A in the corpus or not?

Schematically



Pre-training

Fine-Tuning

Some details

Some details

- BASE model:
  - 12 Transformer Blocks
  - Hidden vector size: 768
  - Attention heads / layer: 12
  - Total parameters: 110M

Some details

- BASE model:
  - 12 Transformer Blocks
  - Hidden vector size: 768
  - Attention heads / layer: 12
  - Total parameters: 110M
- LARGE model:
  - 24 Transformer Blocks
  - Hidden vector size: 1024
  - Attention heads / layer: 16
  - Total parameters: 340M

Some details

- BASE model:
  - 12 Transformer Blocks
  - Hidden vector size: 768
  - Attention heads / layer: 12
  - Total parameters: 110M
- LARGE model:
  - 24 Transformer Blocks
  - Hidden vector size: 1024
  - Attention heads / layer: 16
  - Total parameters: 340M

this is the first work to demonstrate convincingly that scaling to extreme model sizes also leads to large improvements on very small scale tasks, provided that the model has been sufficiently pre-trained. Peters et al. (2018b) presented

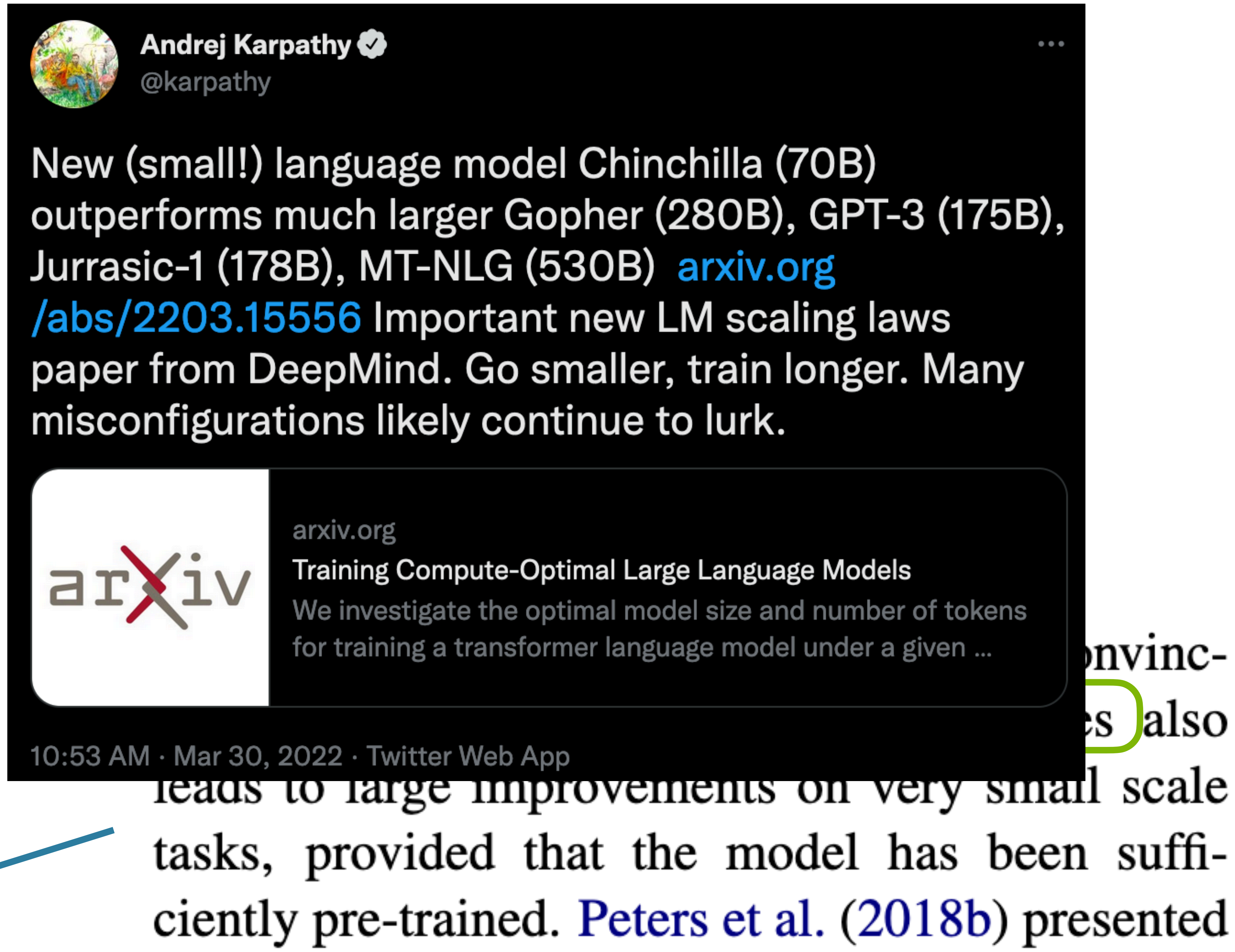
Some details

- BASE model:
  - 12 Transformer Blocks
  - Hidden vector size: 768
  - Attention heads / layer: 12
  - Total parameters: 110M
- LARGE model:
  - 24 Transformer Blocks
  - Hidden vector size: 1024
  - Attention heads / layer: 16
  - Total parameters: 340M

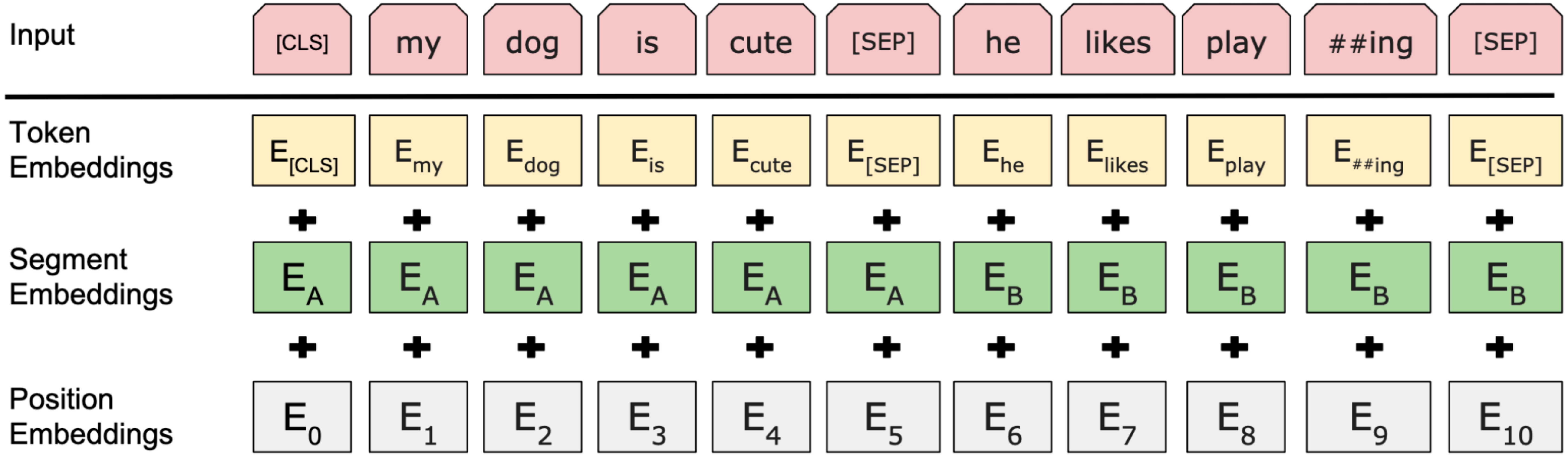
this is the first work to demonstrate convincingly that scaling to extreme model sizes also leads to large improvements on very small scale tasks, provided that the model has been sufficiently pre-trained. Peters et al. (2018b) presented

Some details

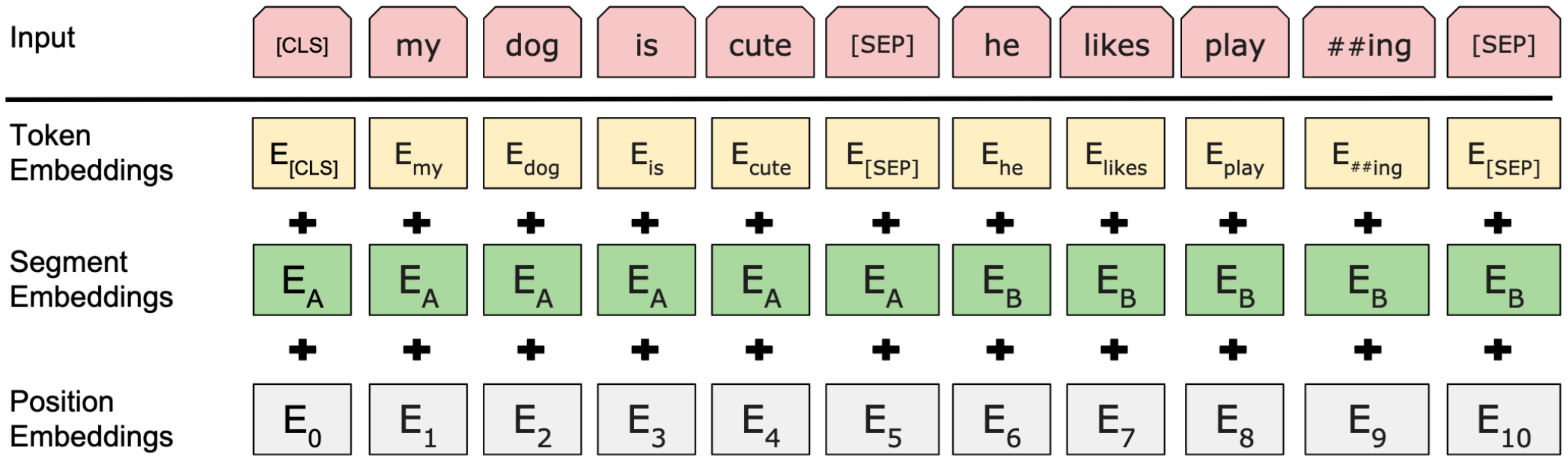
- BASE model:
  - 12 Transformer Blocks
  - Hidden vector size: 768
  - Attention heads / layer: 12
  - Total parameters: 110M
- LARGE model:
  - 24 Transformer Blocks
  - Hidden vector size: 1024
  - Attention heads / layer: 16
  - Total parameters: 340M



Input Representation

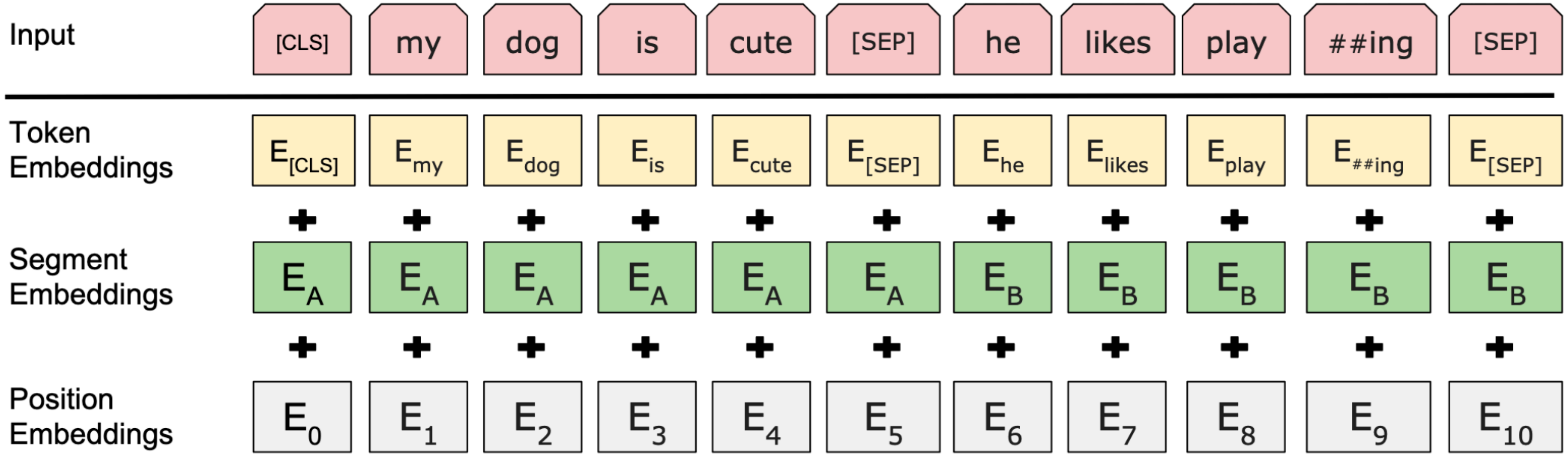


Input Representation



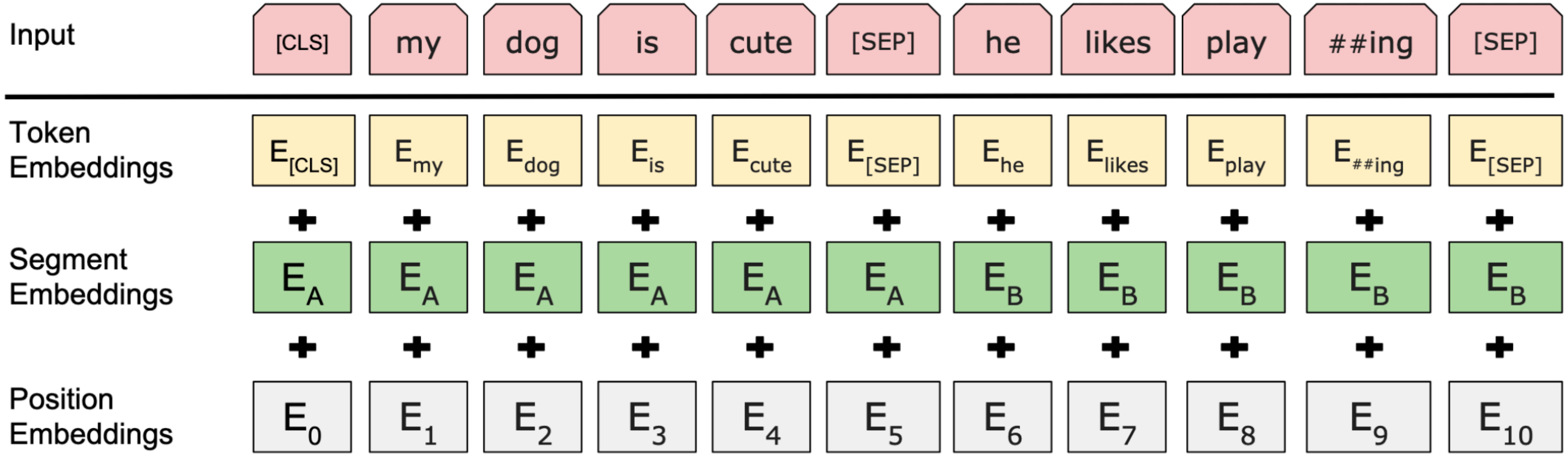
- [CLS], [SEP]: special tokens

Input Representation



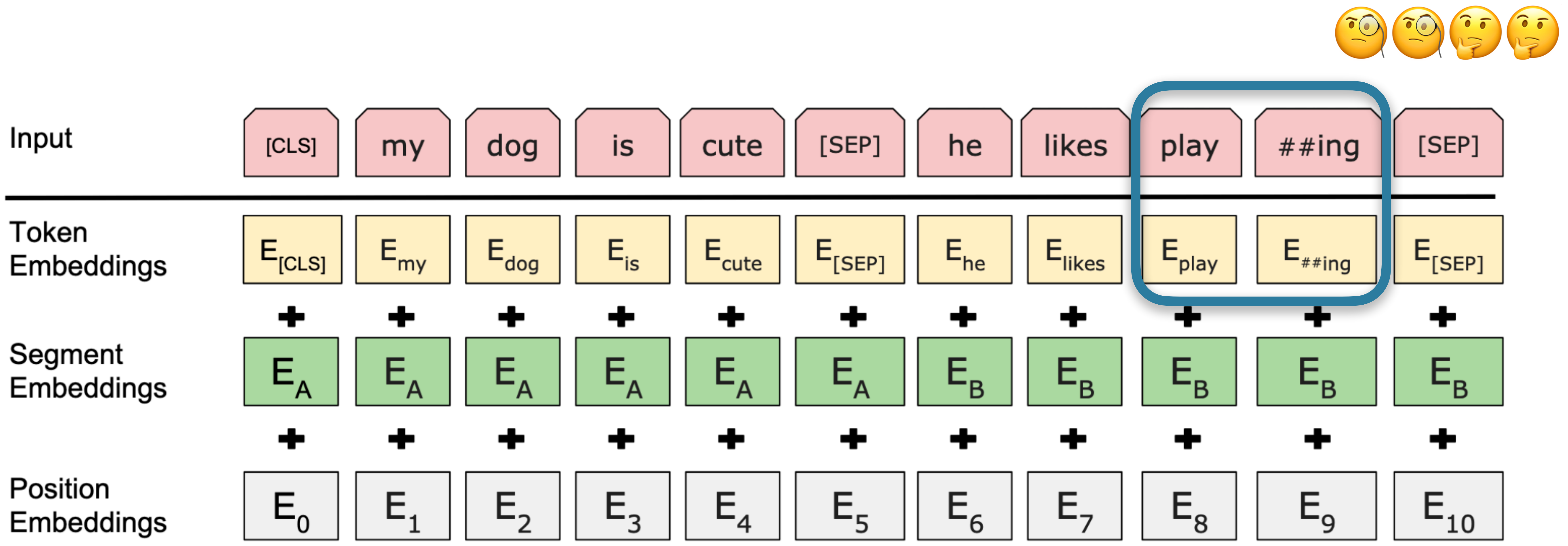
- [CLS], [SEP]: special tokens
- Segment: is this a token from sentence A or B?

Input Representation



- [CLS], [SEP]: special tokens
- Segment: is this a token from sentence A or B?
- Position embeddings: provide position in sequence (*learned* in this case, not fixed)

Input Representation



- [CLS], [SEP]: special tokens
- Segment: is this a token from sentence A or B?
- Position embeddings: provide position in sequence (*learned* in this case, not fixed)

WordPiece Embeddings

- Another solution to OOV problem, from NMT context (see Wu et al 2016)
- Main idea:
  - Fix vocabulary size |V| in advance [for BERT: 30k]
  - Choose |V| wordpieces (subwords) such that total number of wordpieces in the corpus is minimized
- Frequent words aren't split, but rarer ones are
- NB: this is a small issue when you transfer to / evaluate on pre-existing tagging datasets with their own vocabularies. (More on that in week 5.)

Training Details

- BooksCorpus (800M words) + Wikipedia (2.5B)
- Masking the input text. 15% of all tokens are chosen. Then:
  - 80% of the time: replaced by designated '[MASK]' token
  - 10% of the time: replaced by random token
  - 10% of the time: unchanged
- Loss is cross-entropy of the prediction at the masked positions.
- Max seq length: 128 tokens for first 90%, 512 tokens for final 10%
- 1M training steps, batch size 256 = 4 days on 4 or 16 TPUs

Initial Results

System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

Ablations

Hyperparams				Dev Set Accuracy		
#L	#H	#A	LM (ppl)	MNLI-m	MRPC	SST-2
3	768	12	5.84	77.9	79.8	88.4
6	768	3	5.24	80.6	82.2	90.7
6	768	12	4.68	81.9	84.8	91.3
12	768	12	3.99	84.4	86.7	92.9
12	1024	16	3.54	85.7	86.9	93.3
24	1024	16	3.23	86.6	87.8	93.7

- Not a given (depth doesn't help ELMo); possibly a difference between fine-tuning vs. feature extraction

Tasks	Dev Set MNLI-m (Acc)	QNLI (Acc)	MRPC (Acc)	SST-2 (Acc)	SQuAD (F1)
$\text{BERT}_{\text{BASE}}$	84.4	88.4	86.7	92.7	88.5
No NSP	83.9	84.9	86.5	92.6	87.9
LTR & No NSP	82.1	84.3	77.5	92.1	77.8
+ BiLSTM	82.1	84.1	75.7	91.6	84.9

- Many more variations to explore

Outline

- Background
- Recurrent Neural Networks (LSTMs in particular)
  - ELMo
  - seq2seq + *attention*
- Transformers
  - BERT
- **Snapshot of the current landscape**

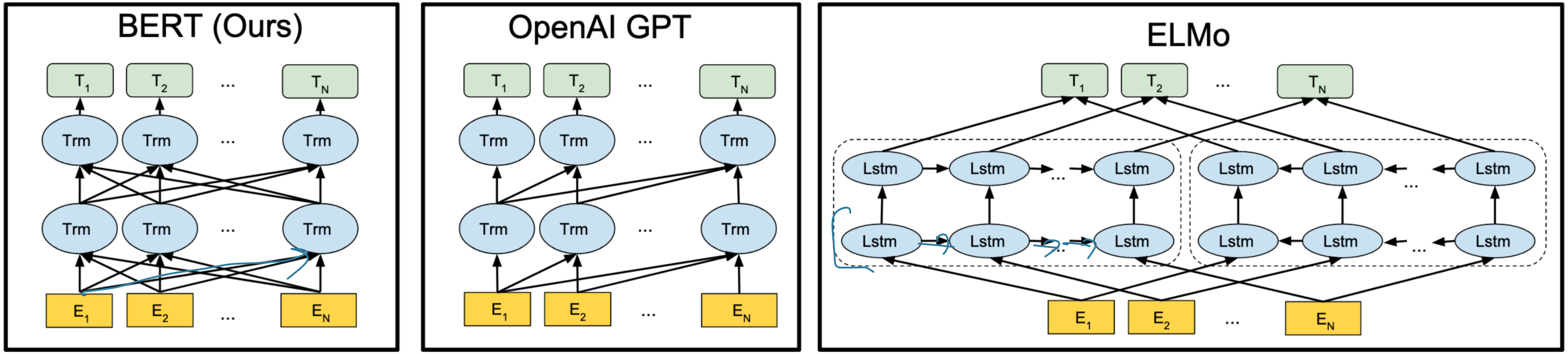
Whirlwind Tour

- Many LMs that have come out since
- Brief description of main changes/innovations
  - Can be useful for analysis projects, e.g. do those changes impact the nature of the representations learned?
- Pointers to multi-lingual and multi-modal models

GPT(2)

- Generative Pre-training
  - Radford et al 2018; 2019 (GPT2)
- Uses Transformer *decoder* instead of *encoder*
  - "Self"-attention: masked so that only can attend to *previous* tokens
  - Pure LM training objective
  - Can be used for text generation
- GPT: same params as BERT-BASE; GPT2 much bigger; GPT3 muuuuch bigger
- Training data: crawled from outbound Reddit links w/ >3 karma, not public

Comparison



Source: BERT paper

Generated Samples (Cherry-picked)

SYSTEM PROMPT (HUMAN-WRITTEN)	*A train carriage containing controlled nuclear materials was stolen in Cincinnati today. Its whereabouts are unknown.*
MODEL COMPLETION (MACHINE-WRITTEN, FIRST TRY)	The incident occurred on the downtown train line, which runs from Covington and Ashland stations. In an email to Ohio news outlets, the U.S. Department of Energy said it is working with the Federal Railroad Administration to find the thief. “The theft of this nuclear material will have significant negative consequences on public and environmental health, our workforce and the economy of our nation,” said Tom Hicks, the U.S. Energy Secretary, in a statement. “Our top priority is to secure the theft and ensure it doesn’t happen again.” The stolen material was taken from the University of Cincinnati’s Research Triangle Park nuclear research site, according to a news release from Department officials. The Nuclear Regulatory Commission did not immediately release any information.

Generated Samples (Cherry-picked)

SYSTEM PROMPT (HUMAN-WRITTEN)

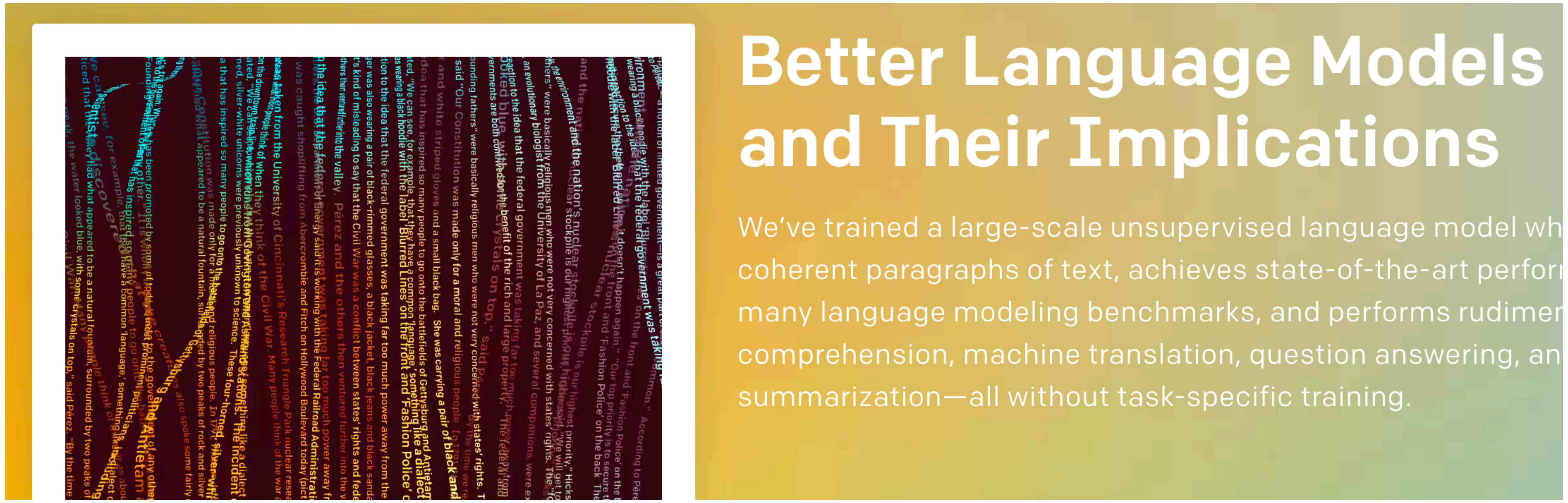
Recycling is good for the world.

NO! YOU COULD NOT BE MORE WRONG!!

MODEL COMPLETION (MACHINE-WRITTEN, 25 TRIES)

Recycling is NOT good for the world. It is bad for the environment, it is bad for our health, and it is bad for our economy. I'm not kidding. Recycling is not good for the environment. It is destructive to the earth and it is a major contributor to global warming. Recycling is not good for our health. It contributes to obesity and diseases like heart disease and cancer. Recycling is bad for our economy. It increases the cost of a product, and in turn, the price of everything that is made with that product. Recycling is not good for our nation. We pay a tremendous price for the privilege of having the world's most advanced and efficient recycling system. Recycling is a huge, colossal waste of time, energy, money, and resources. And THAT is why we need to get back to basics and get back to basics in our recycling efforts. One of the best ways to start is to look at the process of creating a paper product. When you make a paper product, it is basically a long chain of materials. Everything from the raw materials (wood, cardboard, paper, etc.), to the reagents (dyes, solvents, etc.) to the printing equipment (chemicals, glue, paper, ink, etc.), to the packaging, to the packaging materials (mercury, chemicals, etc.) to the processing equipment (heating, cooling, etc.), to the packaging materials, to the packaging materials that are shipped overseas and to the packaging materials that are used in the United States. Each step along the way creates tons of waste that we constantly have to clean up. The process

GPT2 Release Controversy



- Worry: it's so good, it could be used by malicious actors to make deep-fakes for text (incl. fake news, character assassination, …)
- So the model was not initially released (but has been now)
- Some cynical interpretations

https://openai.com/blog/better-language-models/

https://talktotransformer.com/

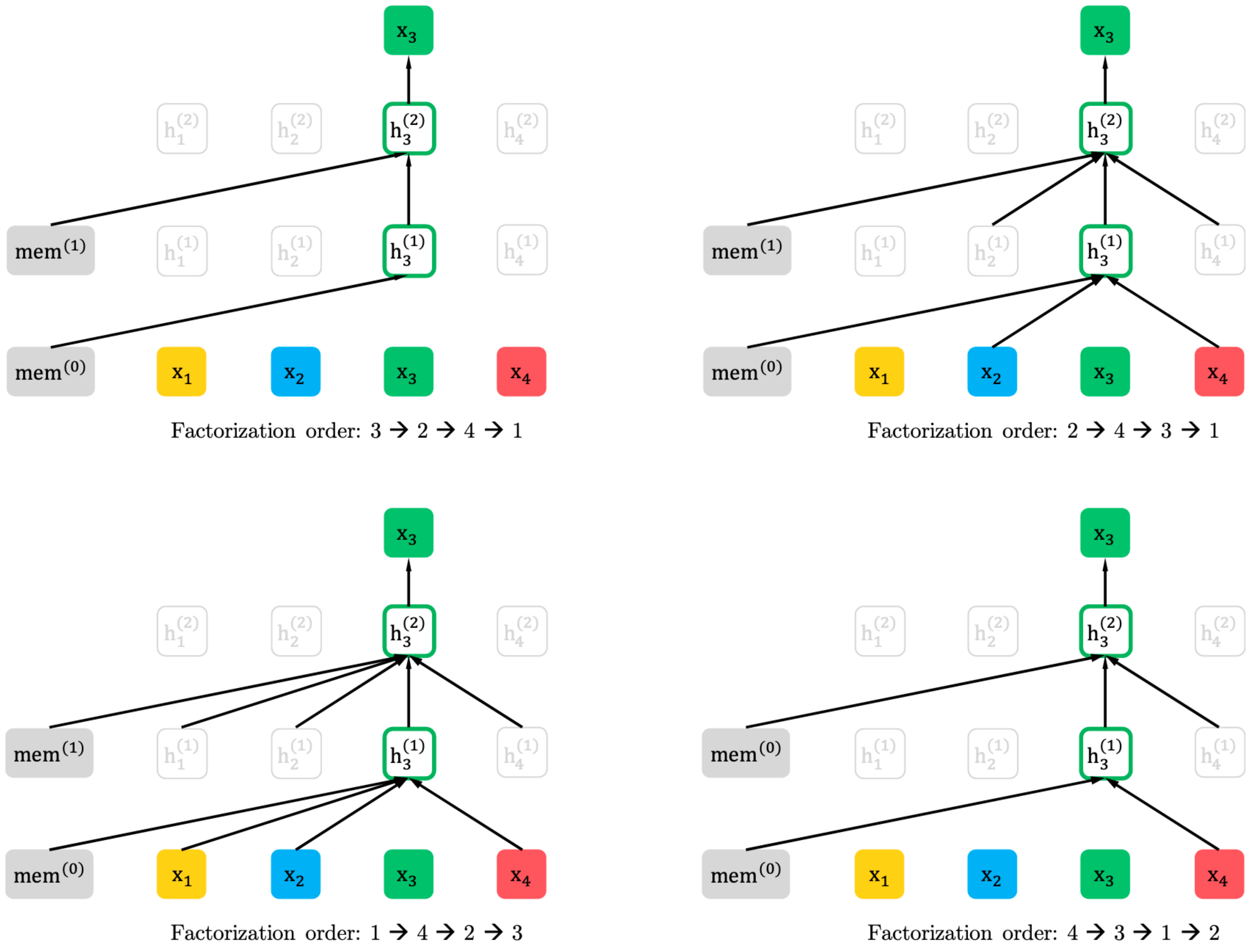
- Prompt: **Studying computational linguistics is a good idea. There are many interesting problems.**
- However, there are many bad problems. The research into computational linguistics has not helped very much in the development of better solutions. One problem with computer science and machine learning is that, until recently, the problems they were supposed to solve were insufficiently well studied. Most of the people who are doing research in computer science (at least in my profession) have no understanding of how to design a particularly natural problem, for example. In the case of the reasoning problem above, most of the people I know who work on computer science or machine learning have no understanding of how to make computer programs work in ways which do not have unavoidable adverse consequences. In the best cases, computational linguistics was little help.

https://talktotransformer.com/

- Prompt: **I didn't ride my bike to campus today because of the snowy weather.**
- There is a lot of construction and I wanted to have a few hours of skiing. All that snow is a real pain in the ass. Good thing there was an LRT planned for that area. See you guys on campus in a few days! Photo credit: Tapto@archivosos.gr – August 26, 2016: Windy Weather on the Mountain (photo credit: Diana Prat)
  From Diana Prat, via tapto@archivosos.gr:
  Ever since the camp that I organized to protest the modernisation of the Barranco, my beautiful alma mater, you know, going in for a press conference, I was asking why the Italian government

XLNet

- Main innovation: *permutation* language modeling.
  - Like LM, but across all possible orders for factorizing
- Significantly outperforms BERT-Large, with same hyper parameters and same training data
  - [NB: still not quite the exact same model]
- Full model: 512 TPUs for 6 days



RoBERTa

- Robustly optimized BERT approach
- Same BERT-large model, but try variations on the pre-training procedure

Model	data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**
$\text{BERT}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7
$\text{XLNet}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	94.0/87.8	88.4	94.4
+ additional data	126GB	2K	500K	94.5/88.8	89.8	95.6

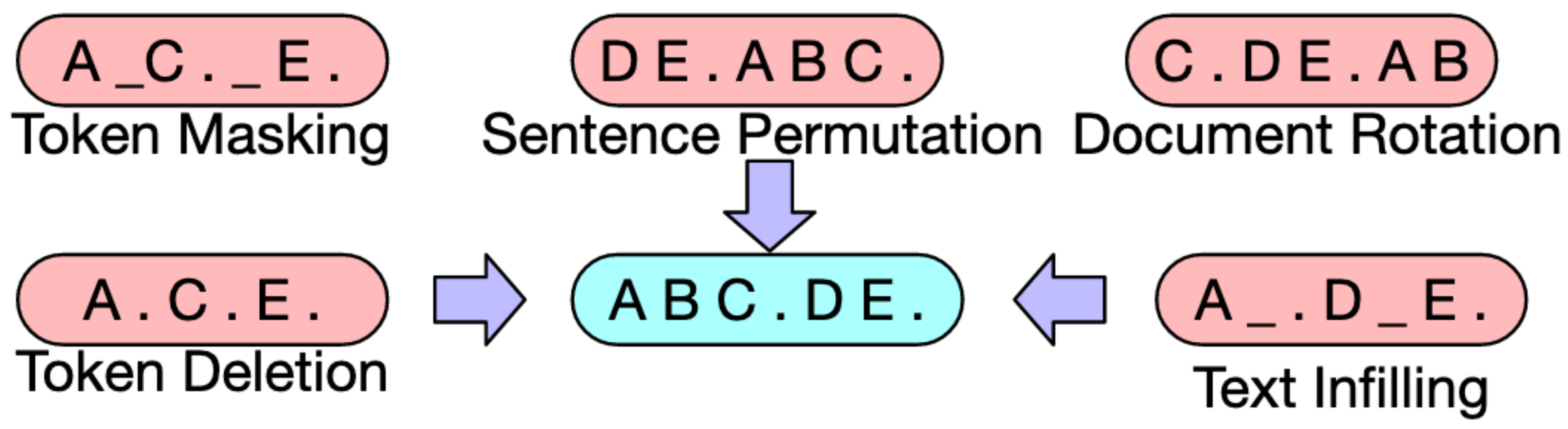
A Lite BERT (ALBERT)

- Reducing parameters while keeping overall architecture:
  - Smaller wordpiece embeddings (not same size as hidden layer)
  - Share parameters *across* transformer blocks
- Instead of NSP: AB+, BA- examples. (Harder task.)

Model		Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg	Speedup
BERT	base	108M	90.4/83.2	80.4/77.6	84.5	92.8	68.2	82.3	17.7x
	large	334M	92.2/85.5	85.0/82.2	86.6	93.0	73.9	85.2	3.8x
	xlarge	1270M	86.4/78.1	75.5/72.6	81.6	90.7	54.3	76.6	1.0
ALBERT	base	12M	89.3/82.3	80.0/77.1	81.6	90.3	64.0	80.1	21.1x
	large	18M	90.6/83.9	82.3/79.4	83.5	91.7	68.5	82.4	6.5x
	xlarge	60M	92.5/86.1	86.1/83.1	86.4	92.4	74.8	85.5	2.4x
	xxlarge	235M	**94.1/88.3**	**88.1/85.1**	**88.0**	**95.2**	**82.3**	**88.7**	1.2x

BART

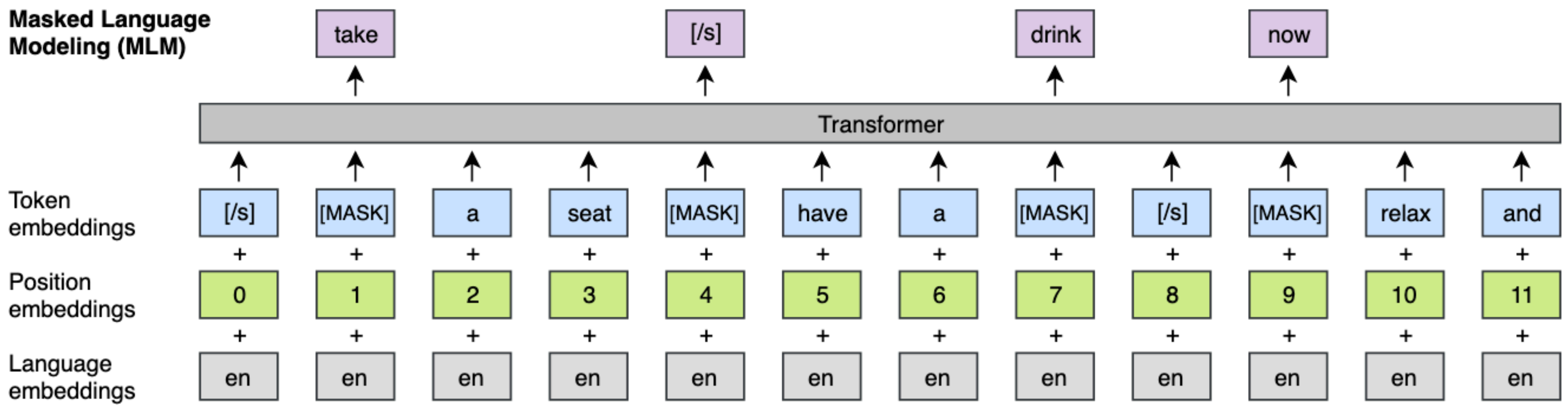
- Full Transformer, i.e. encoder-decoder transducer
  - Many composable transformations of raw text, presented to encoder
  - Goal of decoder is to reconstruct the original text



- Good for both discrimination and generation

Multilingual Models

- Common practice = variations on:
  - Concatenate monolingual corpora
    - Upsample less-frequent languages: $p(l_i) := \frac{n_i^\alpha}{\sum_j n_j^\alpha}$
  - *Shared* wordpiece/BPE vocabulary
  - Same LM-ish training tasks
- Very useful for:
  - Low-resource languages
  - Unsupervised tasks (e.g. unsup NMT)
  - Zero-shot transfer to new languages
- Some experiments on what it is that makes this kind of training work: https://aclanthology.org/2020.acl-main.536/



Some Pointers

- Multi-lingual models (train MLM on, e.g. 100 languages with largest Wikipedias):
  - mBERT: https://github.com/google-research/bert/blob/master/multilingual.md
  - XLM(-R):
    - https://arxiv.org/abs/1911.02116,
    - https://github.com/pytorch/fairseq/blob/master/examples/xlmr/README.md
  - mBART: https://direct.mit.edu/tacl/article/doi/10.1162/tacl_a_00343/96484/Multilingual-Denoising-Pre-training-for-Neural
- Multi-modal models (e.g. vision and language):
  - VisualBERT: https://arxiv.org/abs/1908.03557
  - ViLBERT: https://openreview.net/forum?id=S1eOXNHeUS
  - CLIP: https://arxiv.org/abs/2103.00020

Transformer LM Table

	Encoder-only	Decoder-only	Encoder-decoder
English-only *	BERT, RoBERTa, XLNet, ALBERT, …	GPT-n	BART
Multilingual	mBERT, XLM(-R), …	XGLM	mBART, MASS

Note on Smaller Models

Note on Smaller Models

- DistilBERT (similar variations):
  - Initialize new model, 1/2x the size of BERT-base
  - Train via "knowledge distillation", i.e. predict BERT-base's behavior
  - Great starting point b/c of smaller size

	H=128	H=256	H=512	H=768
L=2	4.4	9.7	22.8	39.2
L=4	4.8	11.3	29.1	53.4
L=6	5.2	12.8	35.4	67.5
L=8	5.6	14.4	41.7	81.7
L=10	6.0	16.0	48.0	95.9
L=12	6.4	17.6	54.3	110.1

(a) Millions of parameters

	H=128	H=256	H=512	H=768
L=2	65.24	31.25	14.44	7.46
L=4	32.37	15.96	7.27	3.75
L=6	21.87	10.67	4.85	2.50
L=8	16.42	8.01	3.64	1.88
L=10	13.05	6.37	2.90	1.50
L=12	11.02	5.35	2.43	1.25

(b) Relative speedup wrt $\text{BERT}_{\text{LARGE}}$ on TPU v2

Note on Smaller Models

- DistilBERT (similar variations):
  - Initialize new model, 1/2x the size of BERT-base
  - Train via "knowledge distillation", i.e. predict BERT-base's behavior
  - Great starting point b/c of smaller size
- Mini BERTs: systematically trained with BERT objective, but varying # layers and hidden dimension
  - Pretraining only vs. distillation vs. fine-tuning
  - https://huggingface.co/google/bert_uncased_L-2_H-128_A-2 [more in week 5]

Table 1: **DistilBERT retains 97% of BERT performance.** Comparison on the dev sets of the GLUE benchmark. ELMo results as reported by the authors. BERT and DistilBERT results are the medians of 5 runs with different seeds.

Model	**Score**	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	WNLI
ELMo	68.7	44.1	68.6	76.6	71.1	86.2	53.4	91.5	70.4	56.3
BERT-base	79.5	56.3	86.7	88.6	91.8	89.6	69.3	92.7	89.0	53.5
DistilBERT	77.0	51.3	82.2	87.5	89.2	88.5	59.9	91.3	86.9	56.3

Table 2: **DistilBERT yields to comparable performance on downstream tasks.** Comparison on downstream tasks: IMDb (test accuracy) and SQuAD 1.1 (EM/F1 on dev set). D: with a second step of distillation during fine-tuning.

Model	IMDb (acc.)	SQuAD (EM/F1)
BERT-base	93.46	81.2/88.5
DistilBERT	92.82	77.7/85.8
DistilBERT (D)	-	79.1/86.9

Table 3: **DistilBERT is significantly smaller while being constantly faster.** Inference time of a full pass of GLUE task STS-B (sentiment analysis) on CPU with a batch size of 1.

Model	# param. (Millions)	Inf. time (seconds)
ELMo	180	895
BERT-base	110	668
DistilBERT	66	410

	H=128	**H=256**	**H=512**	**H=768**
L=2	4.4	9.7	22.8	39.2
L=4	4.8	11.3	29.1	53.4
L=6	5.2	12.8	35.4	67.5
L=8	5.6	14.4	41.7	81.7
L=10	6.0	16.0	48.0	95.9
L=12	6.4	17.6	54.3	110.1

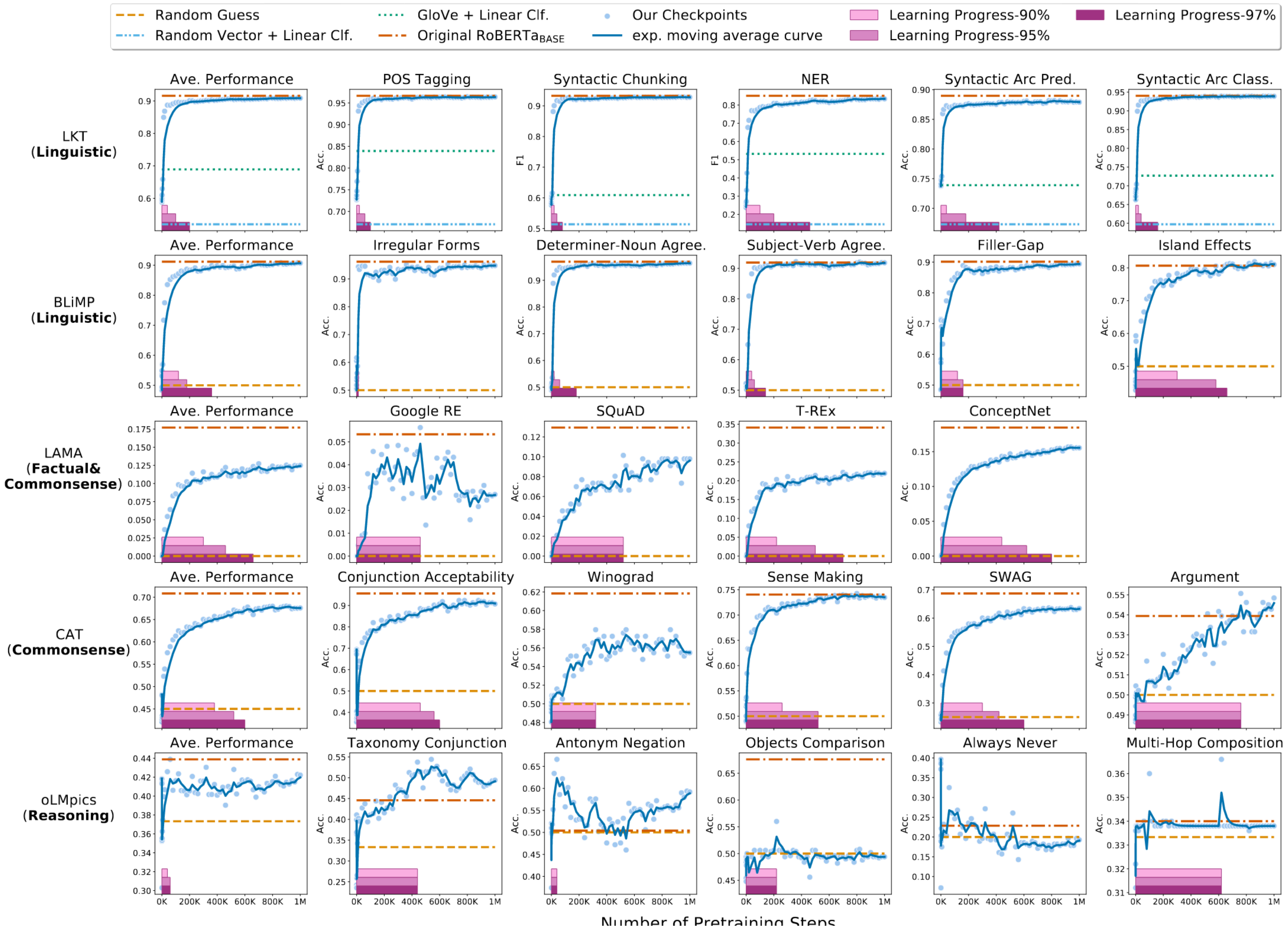
(a) Millions of parameters

	H=128	**H=256**	**H=512**	**H=768**
L=2	65.24	31.25	14.44	7.46
L=4	32.37	15.96	7.27	3.75
L=6	21.87	10.67	4.85	2.50
L=8	16.42	8.01	3.64	1.88
L=10	13.05	6.37	2.90	1.50
L=12	11.02	5.35	2.43	1.25

(b) Relative speedup wrt $\text{BERT}_{\text{LARGE}}$ on TPU v2

Other Model Variations

- MultiBERTs, robustness:
  - Multiple random seeds (25)
  - What happens *during* training?
    - MultiBERTs also releases intermediate checkpoints
    - "Probing Across Time" [more next time] provides RoBERTa checkpoints



Note on the costs of LMs

Note on the costs of LMs

- Currently something of an ‘arms race’ between e.g. Google, Facebook, OpenAI, MS, Baidu

Note on the costs of LMs

- Currently something of an ‘arms race’ between e.g. Google, Facebook, OpenAI, MS, Baidu
- Hugely expensive
  - Carbon emissions
  - Monetarily
    - Inequitable access

Note on the costs of LMs

- Currently something of an ‘arms race’ between e.g. Google, Facebook, OpenAI, MS, Baidu
- Hugely expensive
  - Carbon emissions
  - Monetarily
    - Inequitable access

Energy and Policy Considerations for Deep Learning in NLP

Emma Strubell **Ananya Ganesh** **Andrew McCallum**
College of Information and Computer Sciences
University of Massachusetts Amherst
{strubell, aganesh, mccallum}@cs.umass.edu

Abstract

Recent progress in hardware and methodology for training neural networks has ushered in a new generation of large networks trained on abundant data. These models have obtained notable gains in accuracy across many NLP tasks. However, these accuracy improvements depend on the availability of exceptionally large computational resources that necessitate similarly substantial energy consumption. As a result these models are costly to train and develop, both financially, due to the cost of hardware and electricity or cloud compute time, and environmentally, due to the carbon footprint required to fuel modern tensor

Consumption	**CO_2e (lbs)**
Air travel, 1 person, NY↔SF	1984
Human life, avg, 1 year	11,023
American life, avg, 1 year	36,156
Car, avg incl. fuel, 1 lifetime	126,000
Training one model (GPU)	
NLP pipeline (parsing, SRL)	39
w/ tuning & experiments	78,468
Transformer (big)	192
w/ neural arch. search	626,155

Table 1: Estimated CO_2 emissions from training common NLP models, compared to familiar consumption.[1]

Note on the costs of LMs

- Currently something of an ‘arms race’ between e.g. Google, Facebook, OpenAI, MS, Baidu
- Hugely expensive
  - Carbon emissions
  - Monetarily
    - Inequitable access

Note on the costs of LMs

- Currently something of an ‘arms race’ between e.g. Google, Facebook, OpenAI, MS, Baidu
- Hugely expensive
  - Carbon emissions
  - Monetarily
    - Inequitable access

Green AI

Roy Schwartz*◇ Jesse Dodge*◇♣ Noah A. Smith◇♡ Oren Etzioni◇

◇Allen Institute for AI, Seattle, Washington, USA
♣ Carnegie Mellon University, Pittsburgh, Pennsylvania, USA
♡ University of Washington, Seattle, Washington, USA

July 2019

Abstract

The computations required for deep learning research have been doubling every few months, resulting in an estimated 300,000x increase from 2012 to 2018 [2]. These computations have a surprisingly large carbon footprint [40]. Ironically, deep learning was inspired by the human brain, which is remarkably energy efficient. Moreover, the financial cost of the computations can make it difficult for academics, students, and researchers, in particular those from emerging economies, to engage in deep learning research.

This position paper advocates a practical solution by making **efficiency** an evaluation criterion for research alongside accuracy and related measures. In addition, we propose reporting the financial cost or “price tag” of developing, training, and running models to provide baselines for the investigation of increasingly efficient methods. Our goal is to make AI both greener and more inclusive—enabling any inspired undergraduate with a laptop to write high-quality research papers. Green AI is an emerging focus at the Allen Institute for AI.

Note on the costs of LMs

- Currently something of an ‘arms race’ between e.g. Google, Facebook, OpenAI, MS, Baidu
- Hugely expensive
  - Carbon emissions
  - Monetarily
    - Inequitable access
- A role for interpretability/analysis:
  - Bigger is better, but:
  - Which factors really matter

Green AI

Roy Schwartz[*◇] Jesse Dodge[*◇♣] Noah A. Smith[◇♡] Oren Etzioni[◇]

[◇]Allen Institute for AI, Seattle, Washington, USA
[♣] Carnegie Mellon University, Pittsburgh, Pennsylvania, USA
[♡] University of Washington, Seattle, Washington, USA

July 2019

Abstract

The computations required for deep learning research have been doubling every few months, resulting in an estimated 300,000x increase from 2012 to 2018 [2]. These computations have a surprisingly large carbon footprint [40]. Ironically, deep learning was inspired by the human brain, which is remarkably energy efficient. Moreover, the financial cost of the computations can make it difficult for academics, students, and researchers, in particular those from emerging economies, to engage in deep learning research.

This position paper advocates a practical solution by making **efficiency** an evaluation criterion for research alongside accuracy and related measures. In addition, we propose reporting the financial cost or “price tag” of developing, training, and running models to provide baselines for the investigation of increasingly efficient methods. Our goal is to make AI both greener and more inclusive—enabling any inspired undergraduate with a laptop to write high-quality research papers. Green AI is an emerging focus at the Allen Institute for AI.

More on the Costs of LMs

On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? 🦜

Emily M. Bender*
ebender@uw.edu
University of Washington
Seattle, WA, USA

Timnit Gebru*
timnit@blackinai.org
Black in AI
Palo Alto, CA, USA

Angelina McMillan-Major
aymm@uw.edu
University of Washington
Seattle, WA, USA

Shmargaret Shmitchell
shmargaret.shmitchell@gmail.com
The Aether

ABSTRACT

The past 3 years of work in NLP have been characterized by the development and deployment of ever larger language models, especially for English. BERT, its variants, GPT-2/3, and others, most recently Switch-C, have pushed the boundaries of the possible both through architectural innovations and through sheer size. Using these pretrained models and the methodology of fine-tuning them for specific tasks, researchers have extended the state of the art

alone, we have seen the emergence of BERT and its variants [39, 70, 74, 113, 146], GPT-2 [106], T-NLG [112], GPT-3 [25], and most recently Switch-C [43], with institutions seemingly competing to produce ever larger LMs. While investigating properties of LMs and how they change with size holds scientific interest, and large LMs have shown improvements on various tasks (§2), we ask whether enough thought has been put into the potential risks associated with developing them and strategies to mitigate these risks.

- For more on the reactions to this paper: https://faculty.washington.edu/ebender/stochasticparrots.html

Wrap-up

- The landscape of language models is huge.
- Today: basic building blocks
  - LSTMs
  - Transformers
  - Pointers to more models
- Next time: methods for analyzing these models.
  - That will help formulate research questions.
- Start thinking of questions you might want to ask!

That's all folks!