THE UNIVERSITY OF BRITISH COLUMBIA

Topics in AI (CPSC 532S):
Multimodal Learning with Vision, Language and Sound

Lecture 10: RNNs (part 3)

Course **Logistics**

— **Assignment 3** due date is Wednesday 11:59pm

Final **Project** – Reminder

- Group project (groups of 3 are encouraged, but fewer maybe possible)
- Groups are self-formed or random
- You need to come up with a **project proposal** and then work on the project as a group (each person in the group gets the same grade for the project)
- Project needs to be **research** oriented (not simply implementing an existing paper); you can use code of existing paper as a starting point though

Project proposal and class presentation

Presentation (~3-5 minutes irrespective of the group size)

1. Clear explanation of the **overall problem** you want to solve and relationship to the topics covered in class
2. What **model/algorithms** you planning to explore: this can be somewhat abstract (e.g., CNN+RNN)
3. The **dataset(s)** you will use and how will you **evaluate** performance
4. List of **papers** you plan to read as references
5. How will you **structure the project**, who will do what and a rough timeline

After proposal you will get the feedback from me

Project proposal and class presentation

Presentation (~3-5 minutes irrespective of the group size)

1. Clear explanation of the **overall problem** you want to solve and relationship to the topics covered in class
2. What **model/algorithms** you planning to explore: this can be somewhat abstract (e.g., CNN+RNN)
3. The **dataset(s)** you will use and how will you **evaluate** performance
4. List of **papers** you plan to read as references
5. How will you **structure the project**, who will do what and a rough timeline

After proposal you will get the feedback from me

Proposal

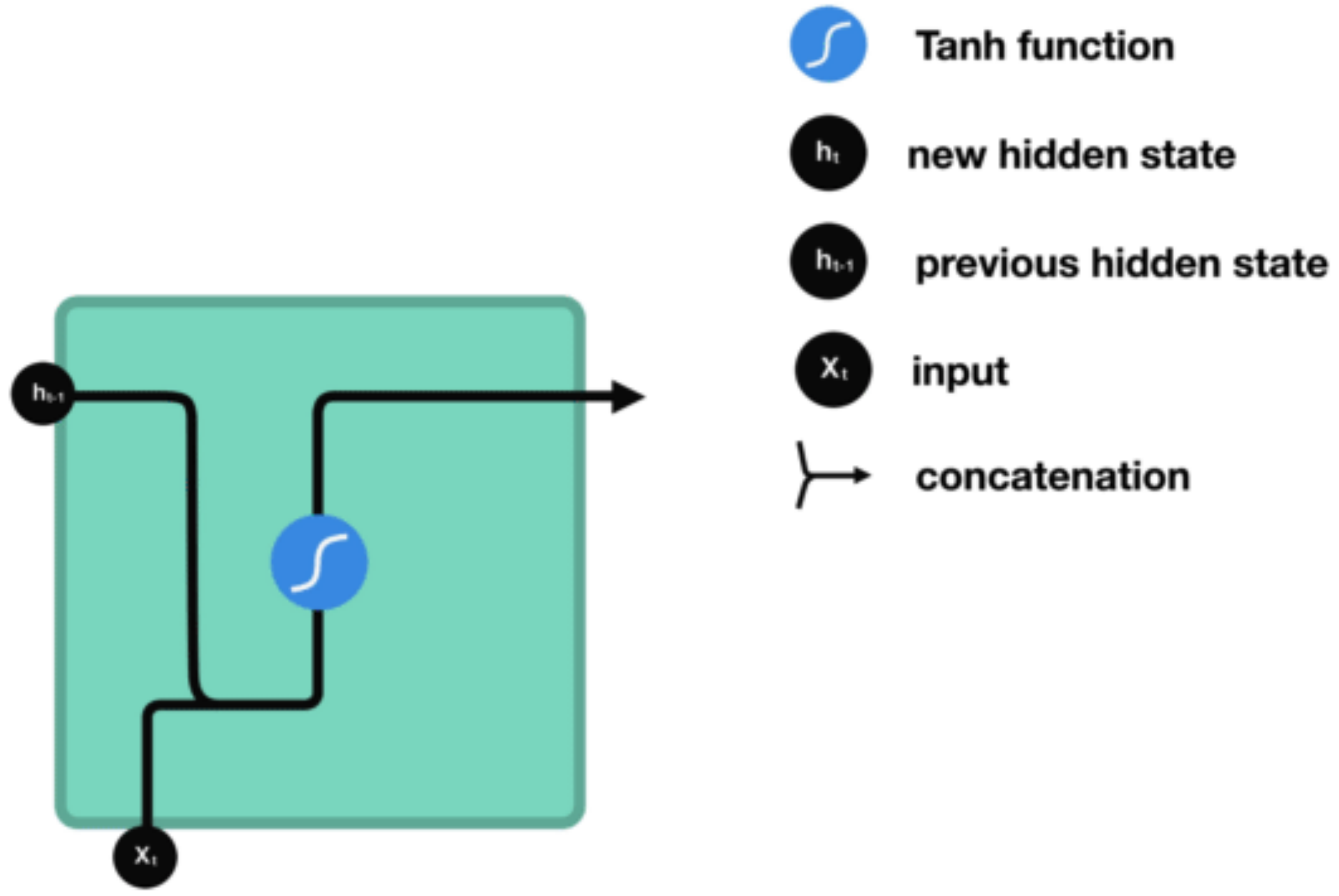
— Same as above but in more detail, with well defined algorithms and timeline

— Will be in the form of the **PDF** document (initial paper draft)

Long-Short Term Memory (**LSTM**)

Vanilla RNN

$$h_t = \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$



fully connected layer of size |h| x (|x| + |h|) with tanh activation function

LSTM

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

four fully connected layers of size |h| x (|x| + |h|) with sigmoid and tanh activation function

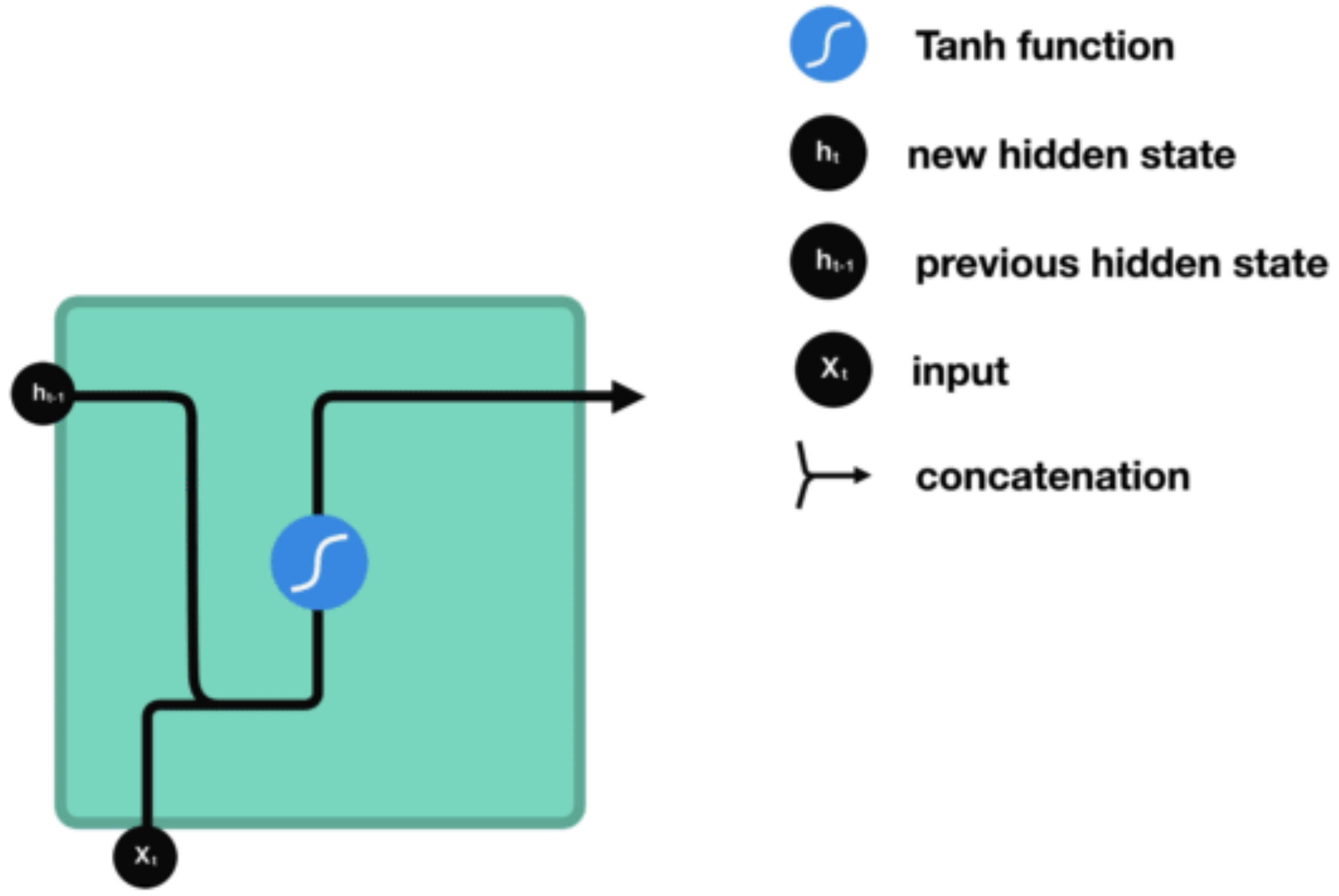
[Hochreiter and Schmidhuber, NC **1977**]

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Long-Short Term Memory (**LSTM**)

Vanilla RNN

$$h_t = \tanh\left(W\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$



fully connected layer of size |h| x (|x| + |h|) with tanh activation function

LSTM

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

four fully connected layers of size |h| x (|x| + |h|) with sigmoid and tanh activation function

[Hochreiter and Schmidhuber, NC **1977**]

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Long-Short Term Memory (**LSTM**)

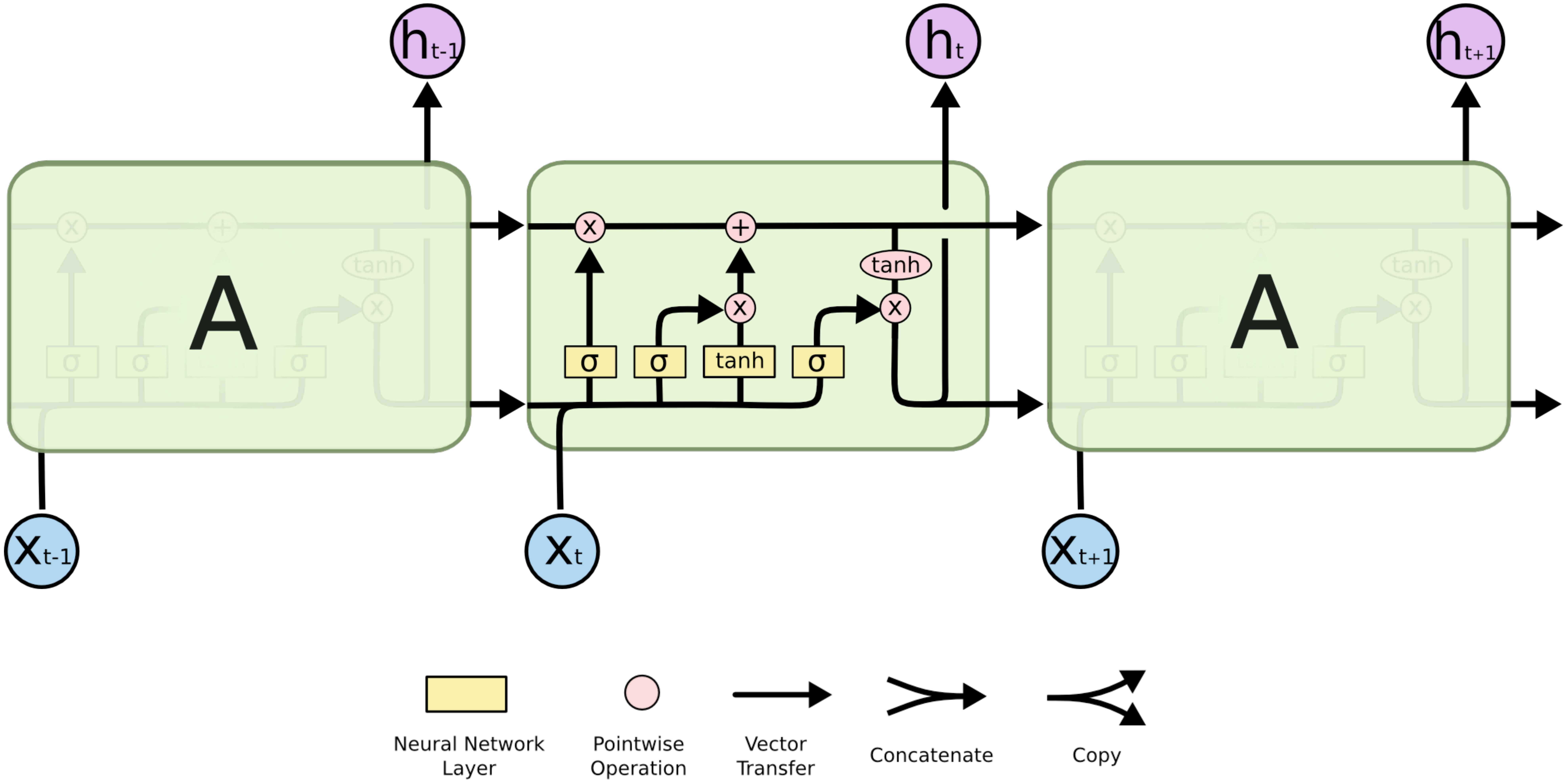


Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

* slide from Dhruv Batra

Long-Short Term Memory (**LSTM**)

Cell state / **memory**

0.1
-0.6
0.1
0.55
-0.67
0.4
0.01
0.7
…
0.9

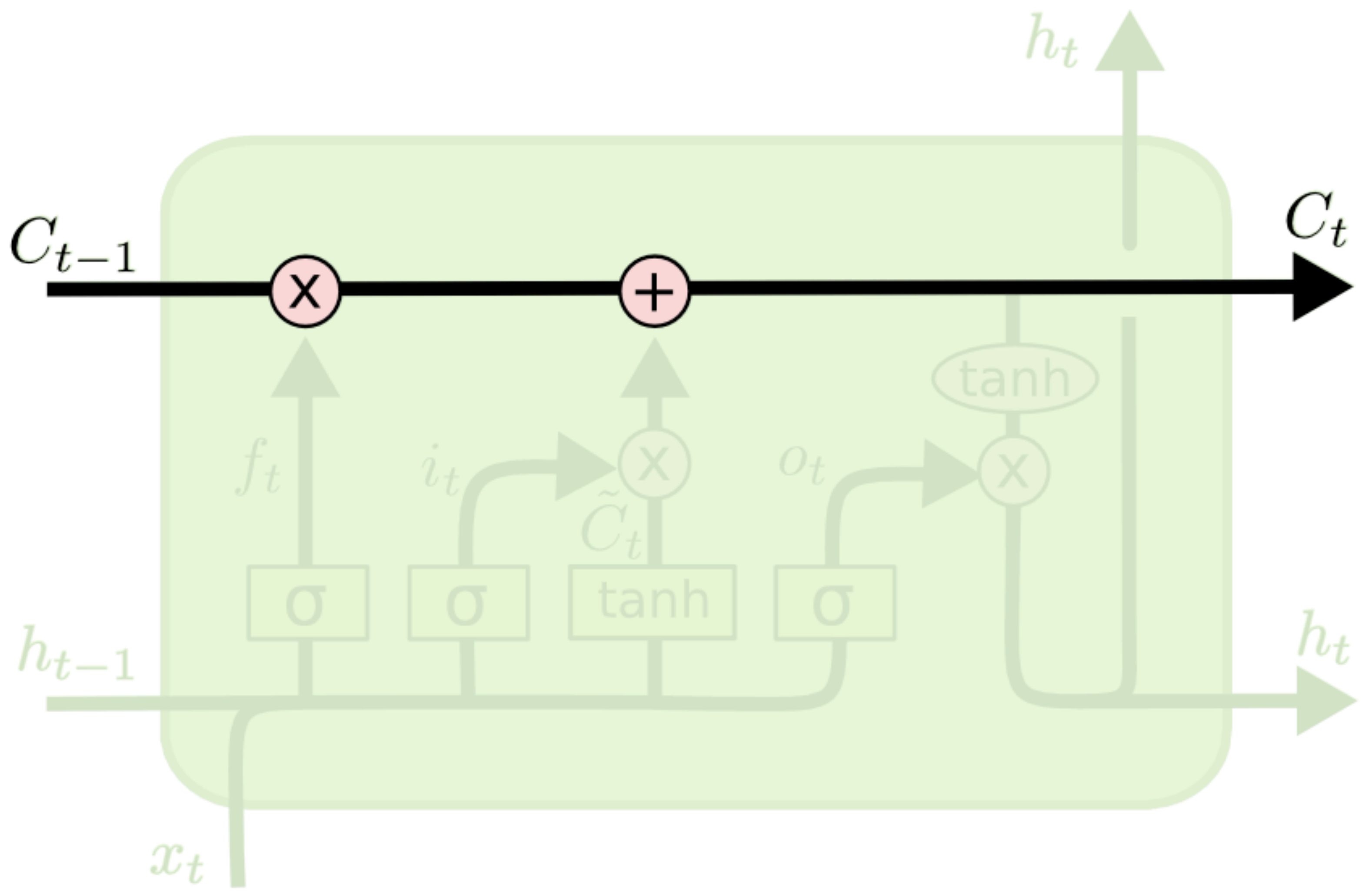


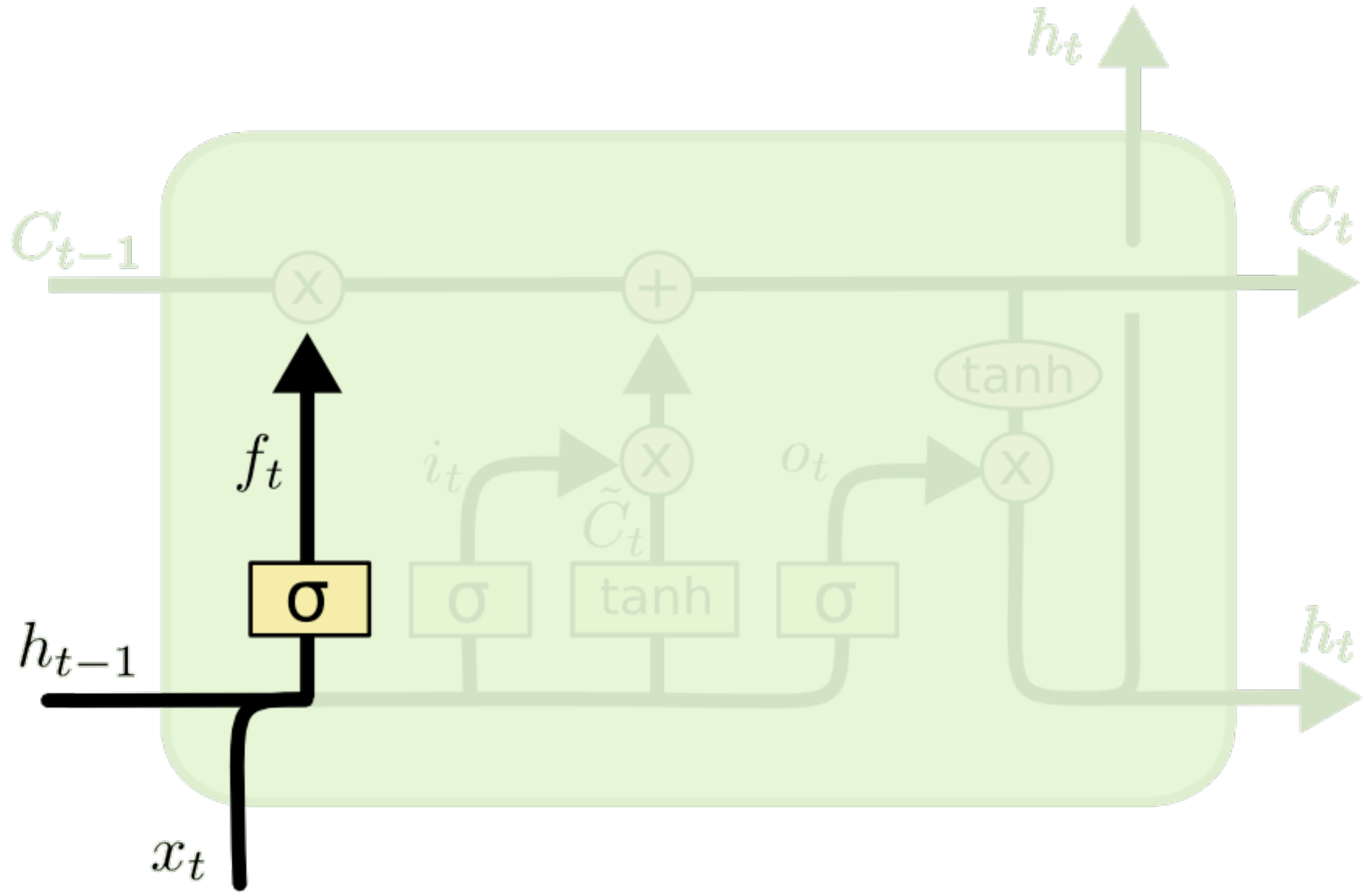
Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

* slide from Dhruv Batra

LSTM Intuition: Forget Gate

Should we continue to **remember** this “bit” of information or not?

0.1
-0.6
0.1
0.55
-0.67
0.4
0.01
0.7
…
0.9



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

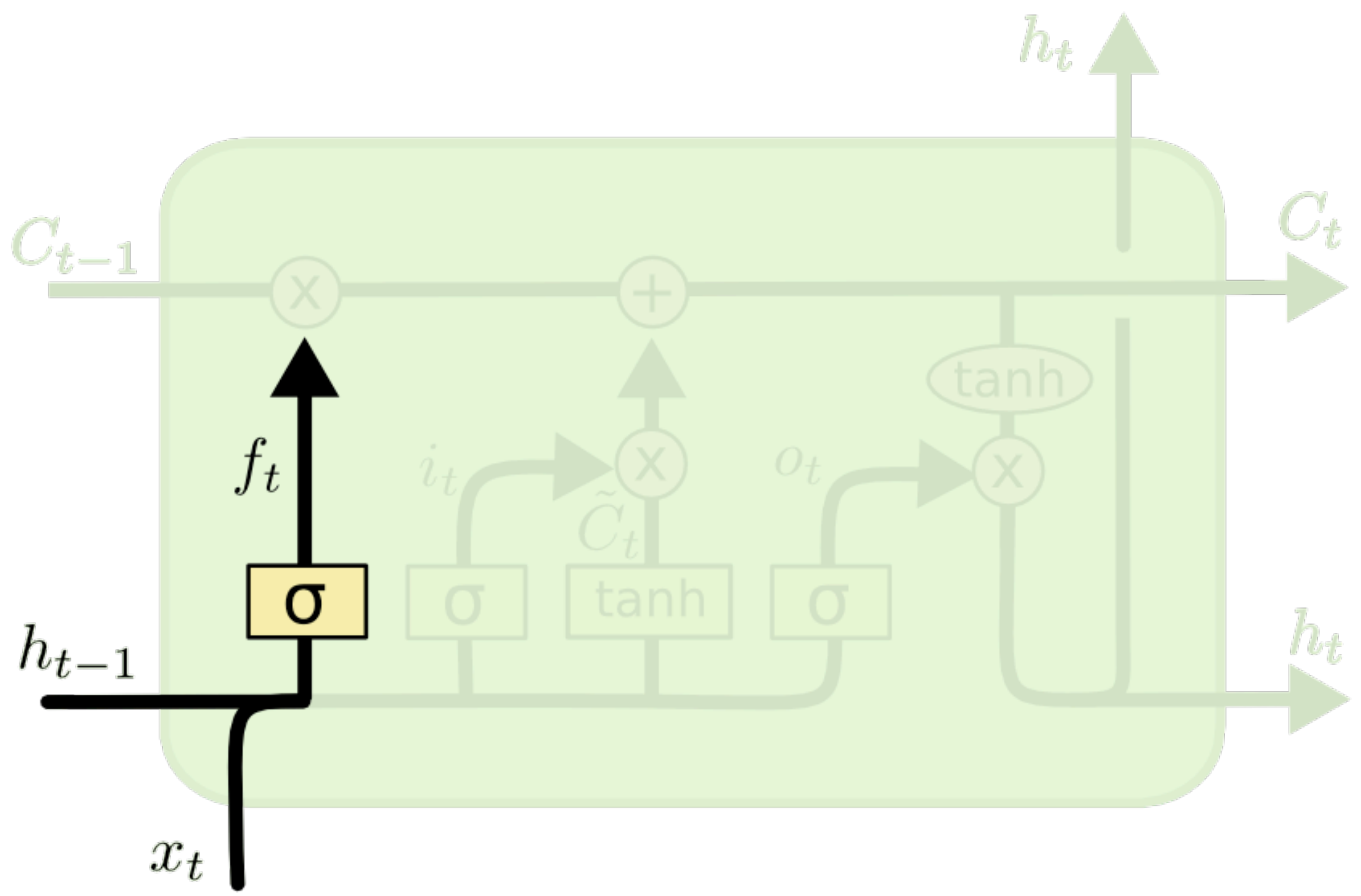
Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

* slide from Dhruv Batra

LSTM Intuition: Forget Gate

Should we continue to **remember** this “bit” of information or not?

0.1
-0.6
0.1
0.55
-0.67
0.4
0.01
0.7
…
0.9



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Intuition: memory and forget gate output multiply, output of forget gate can be though of as binary (0 or 1)

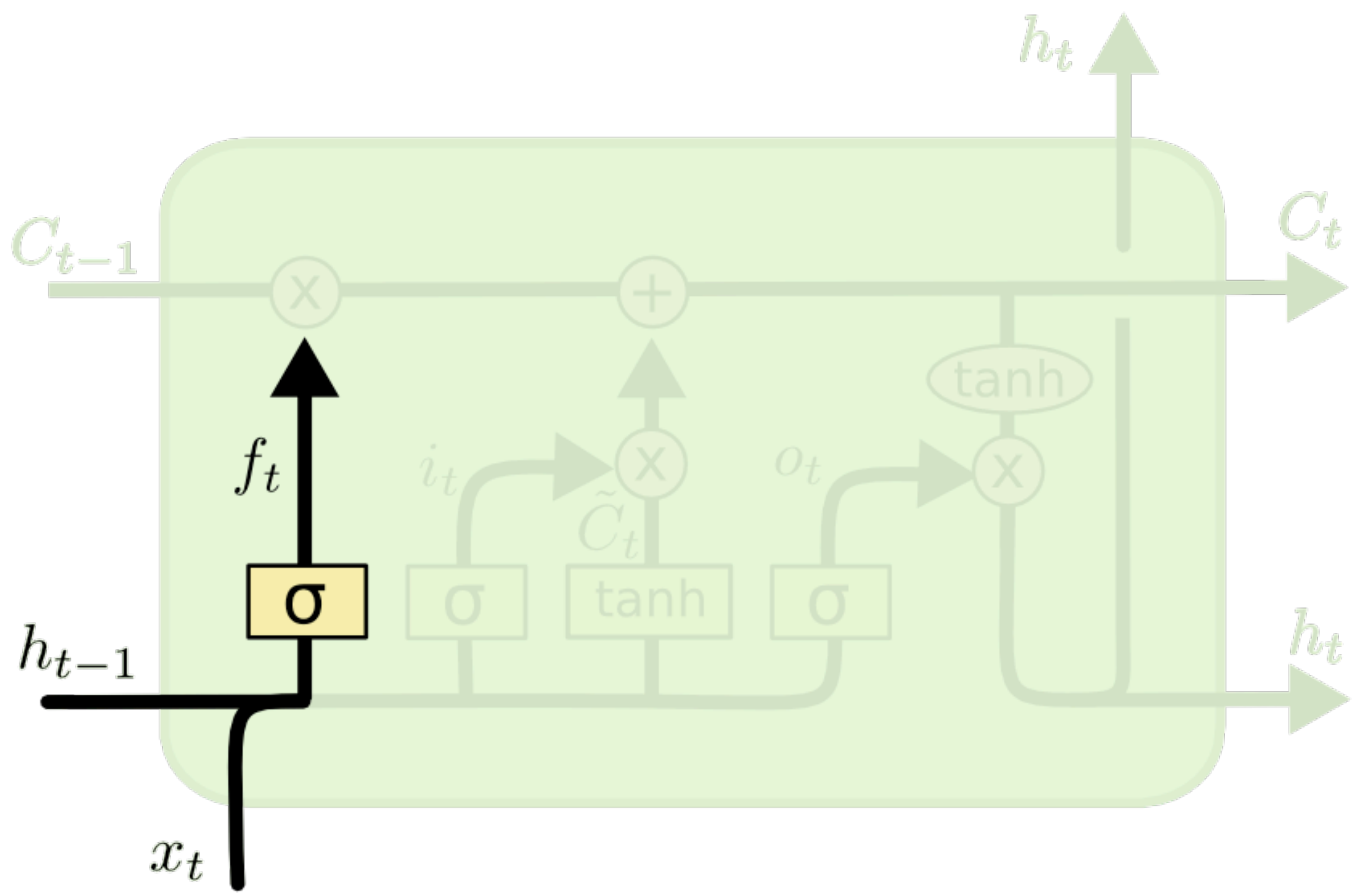
Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

* slide from Dhruv Batra

LSTM Intuition: Forget Gate

Should we continue to **remember** this “bit” of information or not?

0.1
-0.6
0.1
0.55
-0.67
0.4
0.01
0.7
…
0.9



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

0
0
0
0
1
1
1
1
…
1

Intuition: memory and forget gate output multiply, output of forget gate can be though of as binary (0 or 1)

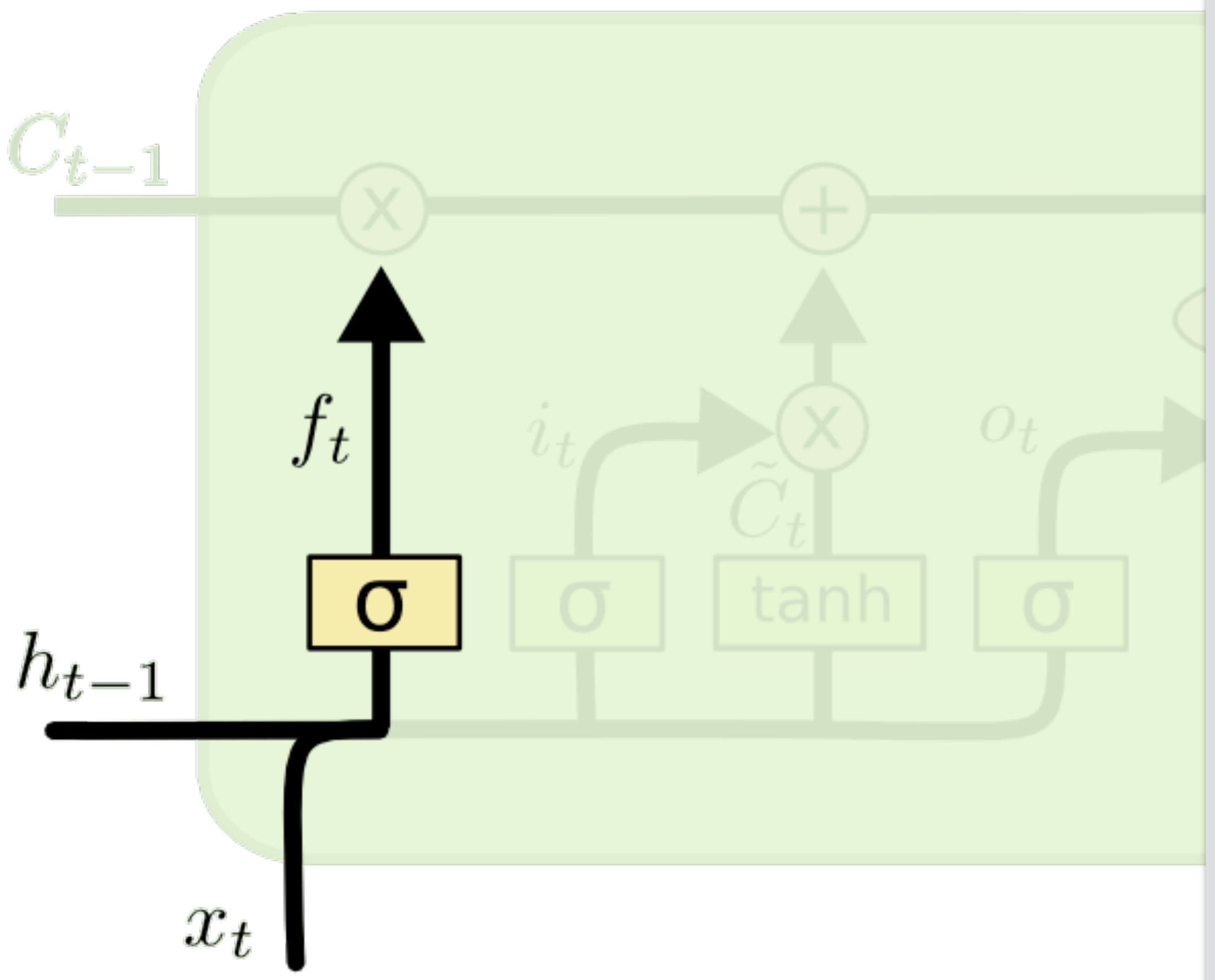
anything x 1 = anything (remember)
anything x 0 = 0 (forget)

Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

* slide from Dhruv Batra

LSTM Intuition: Forget Gate

Should we continue to **remember** this “bit” of information or not?



0.1		0		0
-0.6		0		0
0.1		0		0
0.55		0		0
-0.67		1		-0.67
0.4	X	1	=	0.4
0.01		1		0.01
0.7		1		0.7
…		…		…
0.9		1		0.9

selective read

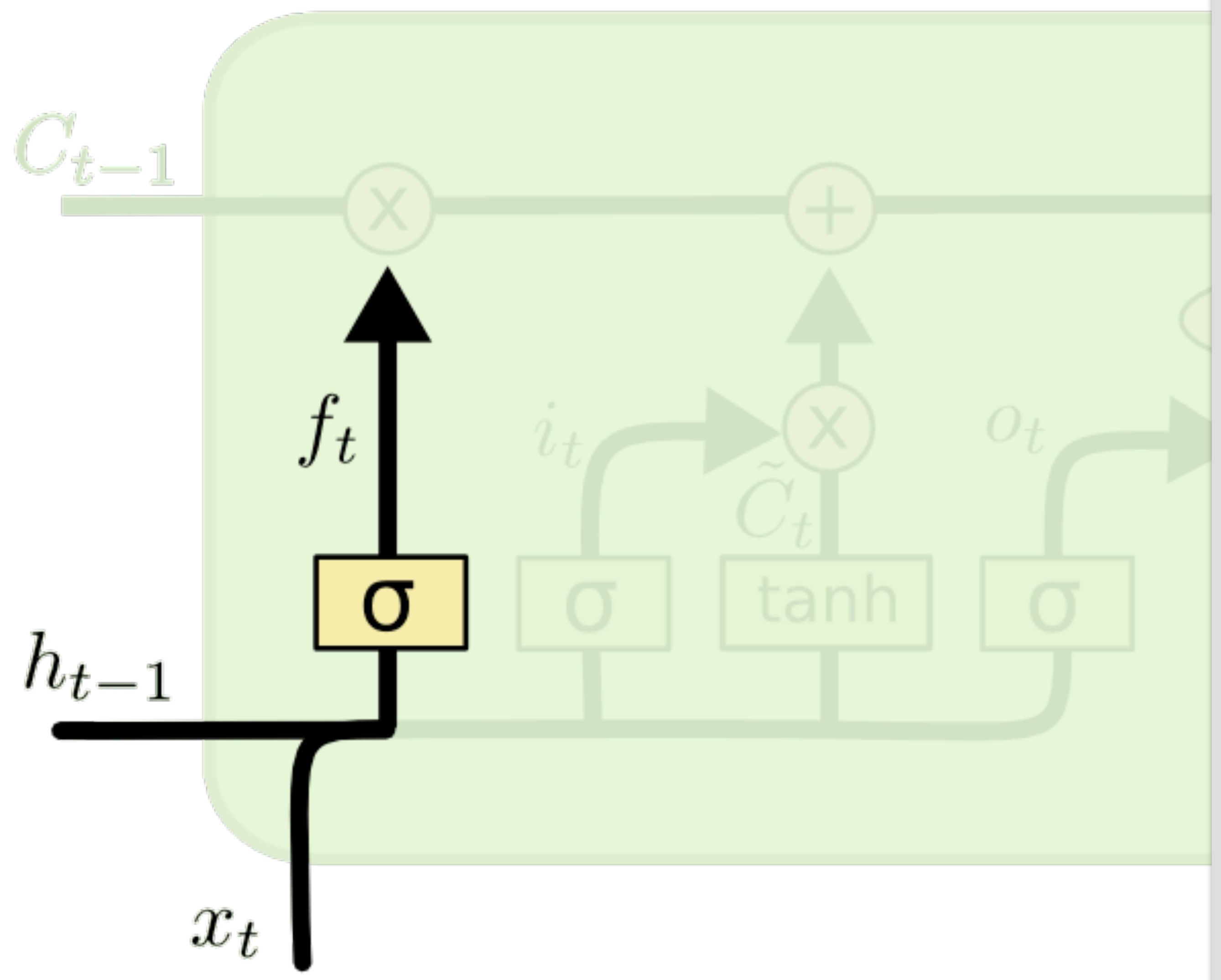
$h_{t-1}, x_t] + b_f)$

Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

* slide from Dhruv Batra

LSTM Intuition: Forget Gate

Should we continue to **remember** this "bit" of information or not?



0.1		0.1		..
-0.6		0.6		..
0.1		0.5		..
0.55		0.3		..
-0.67	X	0.8	=	..
0.4		0.2		..
0.01		0.1		..
0.7		1		..
…		…		…
0.9		0.9		..

selective read

$$h_{t-1}, x_t] \; + \; b_f)$$

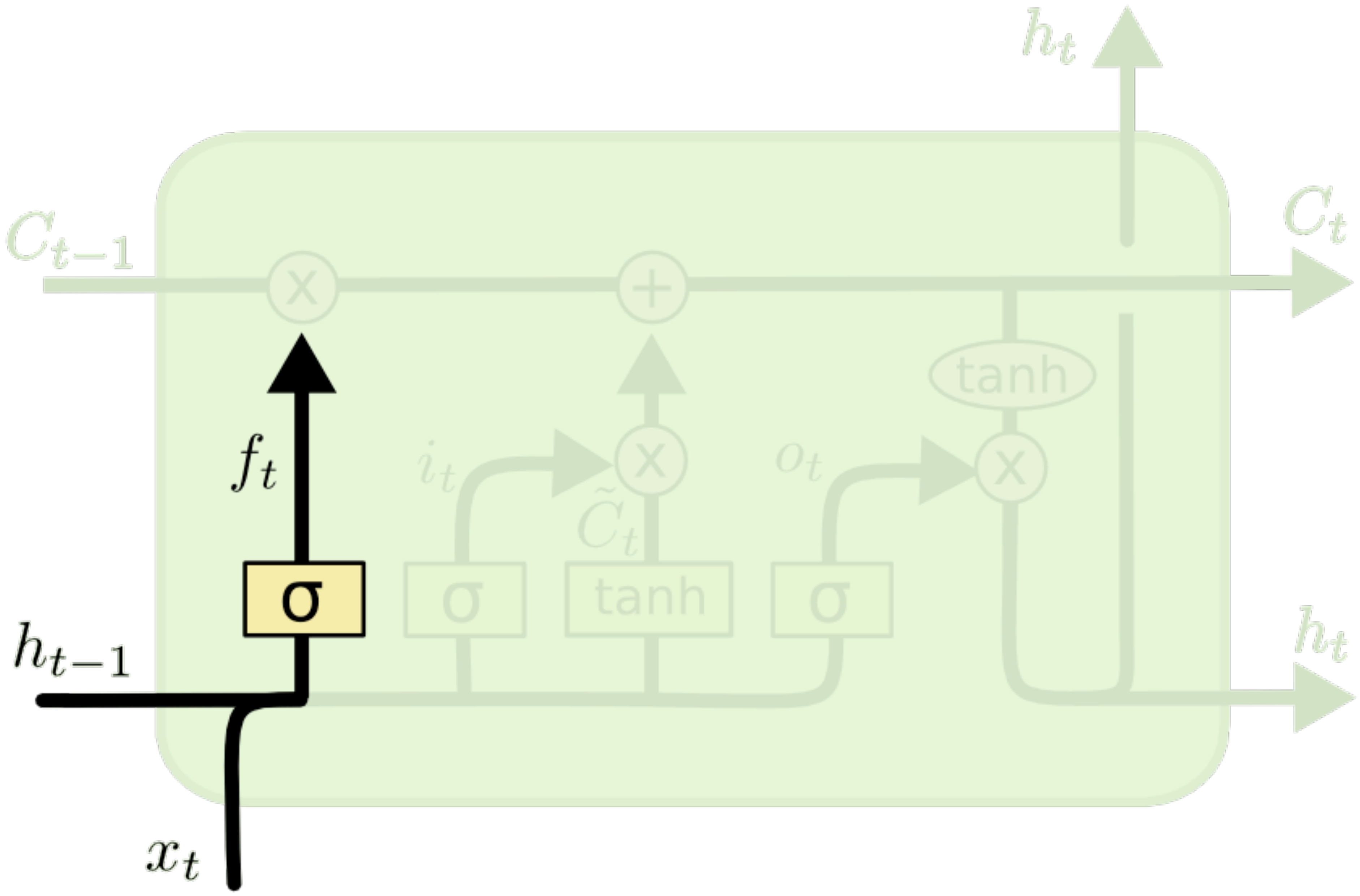
Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

* slide from Dhruv Batra

LSTM Intuition: Forget Gate

Should we continue to **remember** this “bit” of information or not?

0.1
-0.6
0.1
0.55
-0.67
0.4
0.01
0.7
…
0.9



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$

0.1
0.9
0.8
0.1
0
0.2
0
1
…
0.4

Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

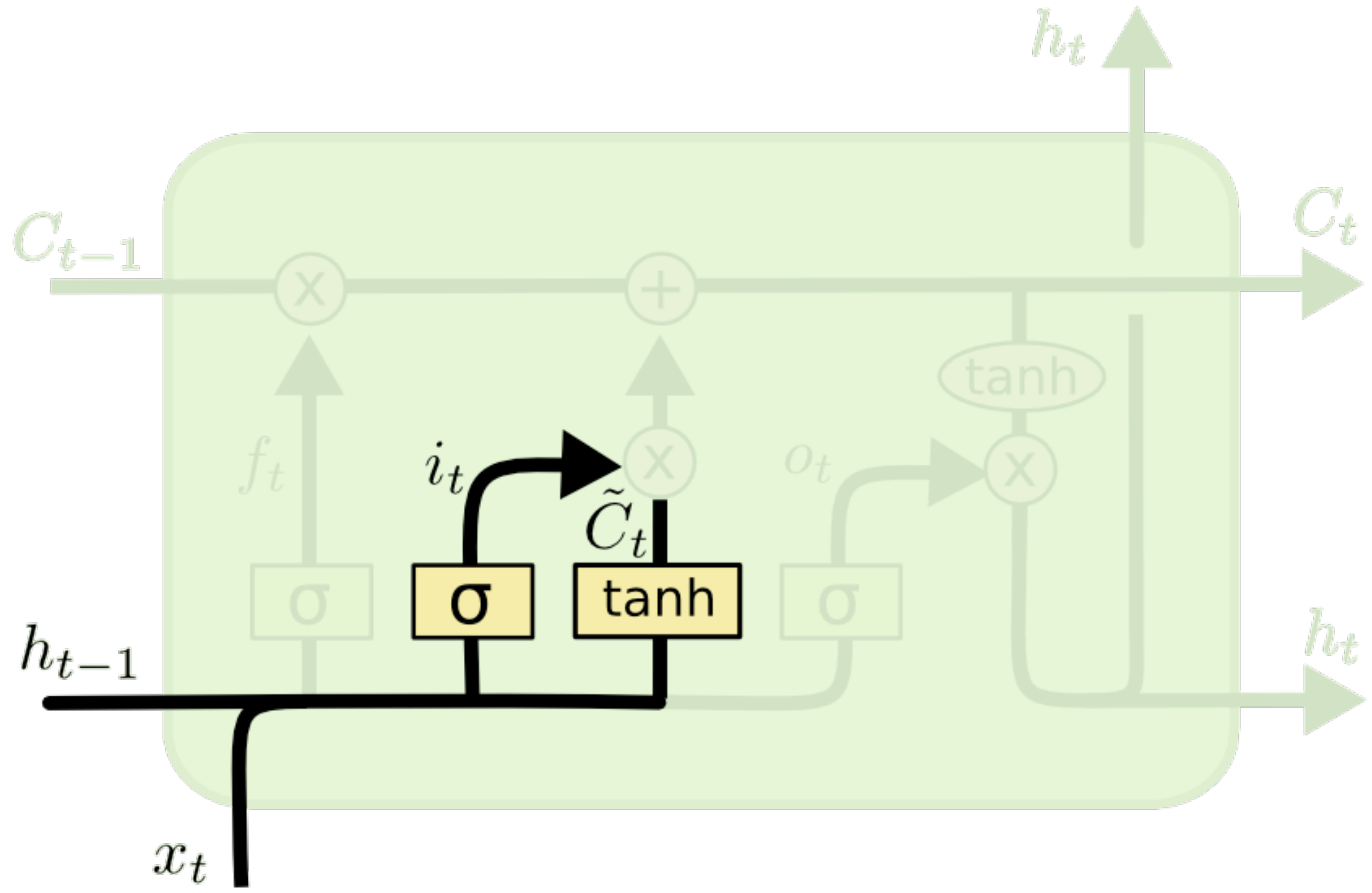
* slide from Dhruv Batra

LSTM Intuition: Input Gate

Should we **update** this “bit” of information or not?

If yes, then what should we **remember**?

0.1
-0.6
0.1
0.55
-0.67
0.4
0.01
0.7
…
0.9



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \;+\; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \;+\; b_C)$$

1
1
1
1
0
0
0
0
…
0

0.3
0.5
0.62
-0.34
0.43
-0.78
0.1
-0.45
…
0.9

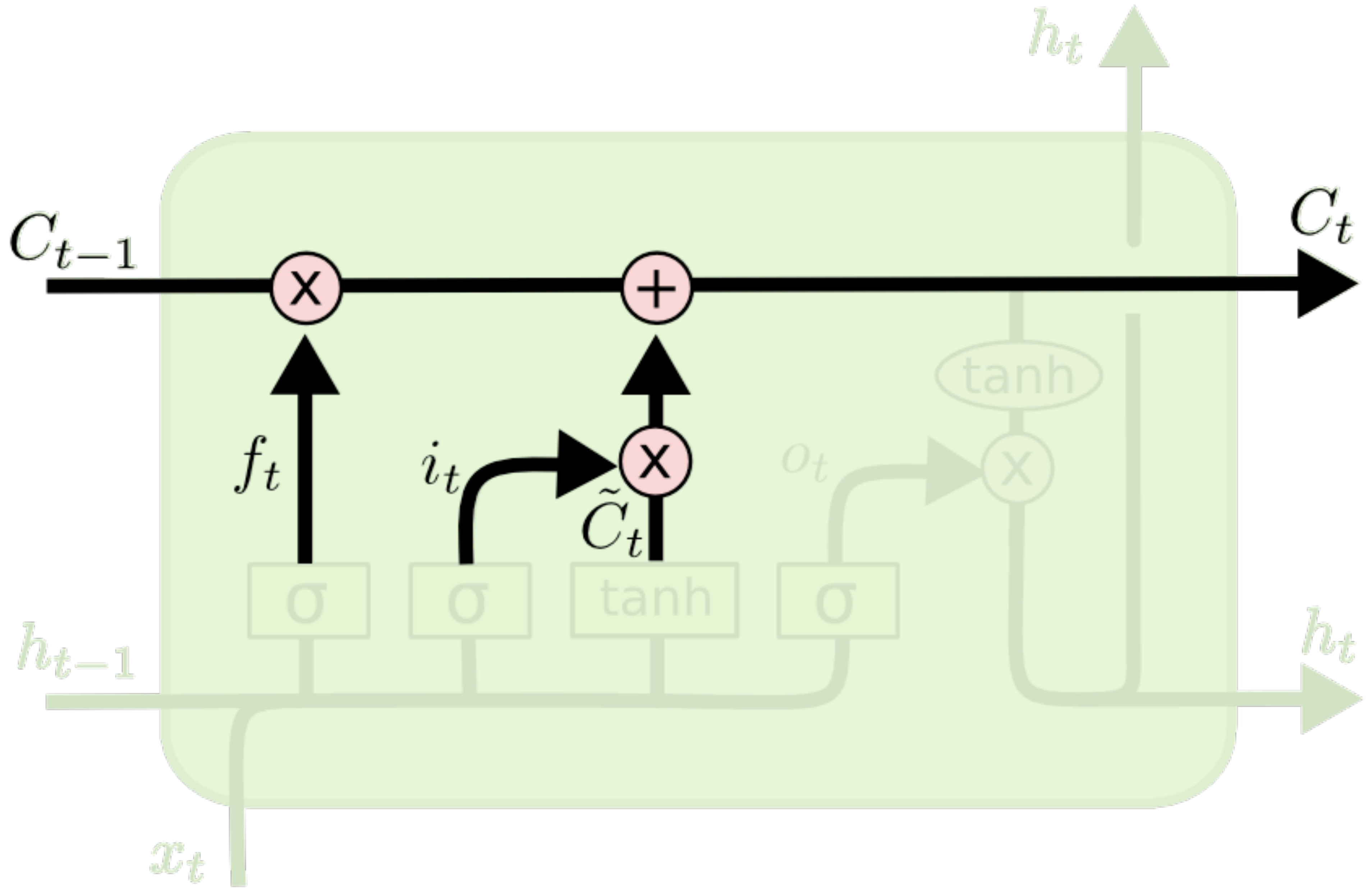
Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

* slide from Dhruv Batra

LSTM Intuition: Memory Update

Forget what needs to be forgotten + memorize what needs to be remembered

0.1
-0.6
0.1
0.55
-0.67
0.4
0.01
0.7
…
0.9



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

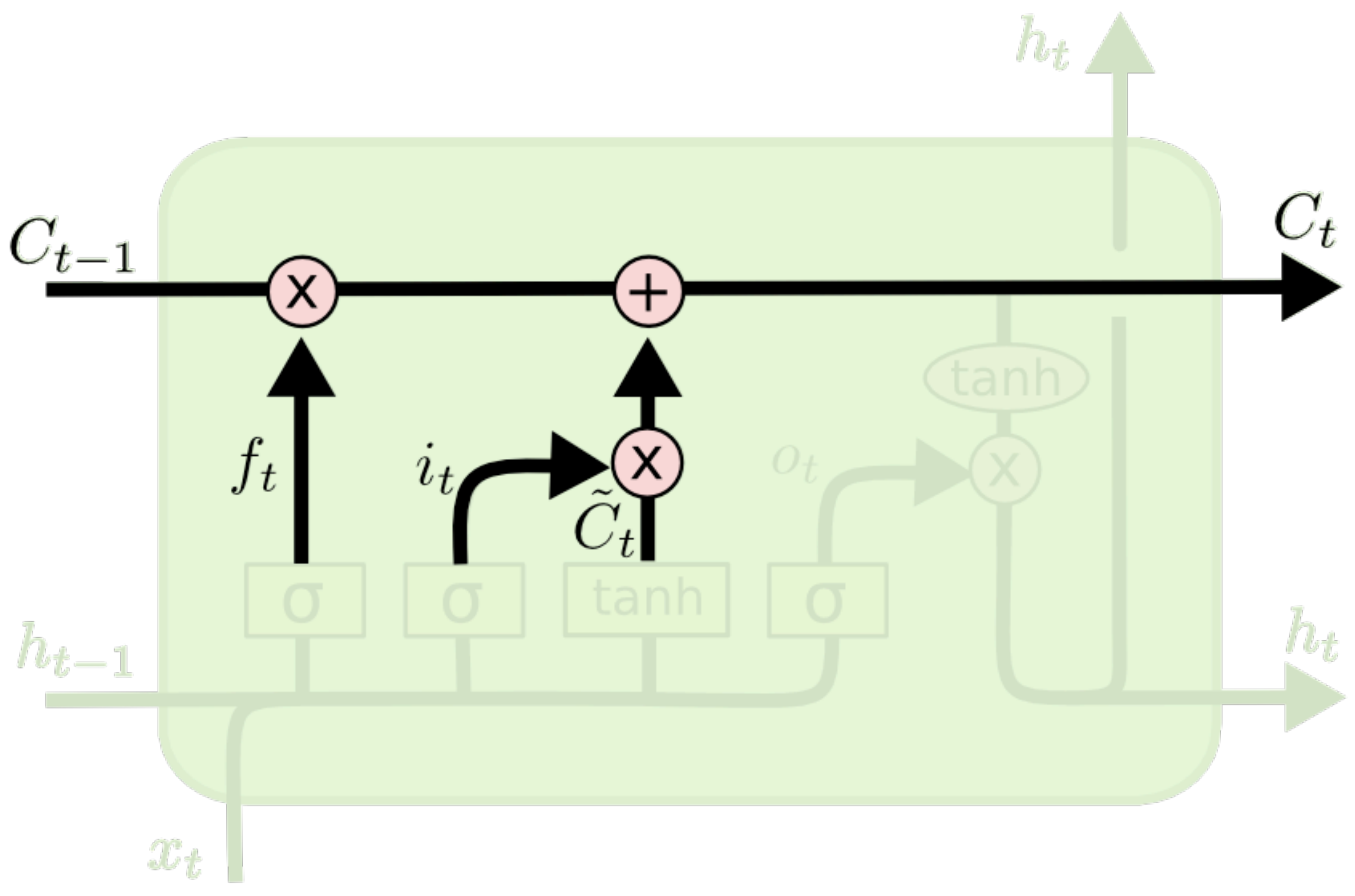
Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

* slide from Dhruv Batra

LSTM Intuition: Memory Update

Forget what needs to be forgotten + memorize what needs to be remembered

$$\begin{bmatrix} 0.1 \\ -0.6 \\ 0.1 \\ 0.55 \\ -0.67 \\ 0.4 \\ 0.01 \\ 0.7 \\ \ldots \\ 0.9 \end{bmatrix}$$



$$\begin{bmatrix} 0.3 \\ 0.5 \\ 0.62 \\ -0.34 \\ -0.67 \\ 0.4 \\ 0.01 \\ 0.7 \\ \ldots \\ 0.9 \end{bmatrix} = \begin{bmatrix} 0.1 \\ -0.6 \\ 0.1 \\ 0.55 \\ -0.67 \\ 0.4 \\ 0.01 \\ 0.7 \\ \ldots \\ 0.9 \end{bmatrix} \text{X} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ \ldots \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ \ldots \\ 0 \end{bmatrix} \text{X} \begin{bmatrix} 0.3 \\ 0.5 \\ 0.62 \\ -0.34 \\ 0.43 \\ -0.78 \\ 0.1 \\ -0.45 \\ \ldots \\ 0.9 \end{bmatrix}$$

selective write

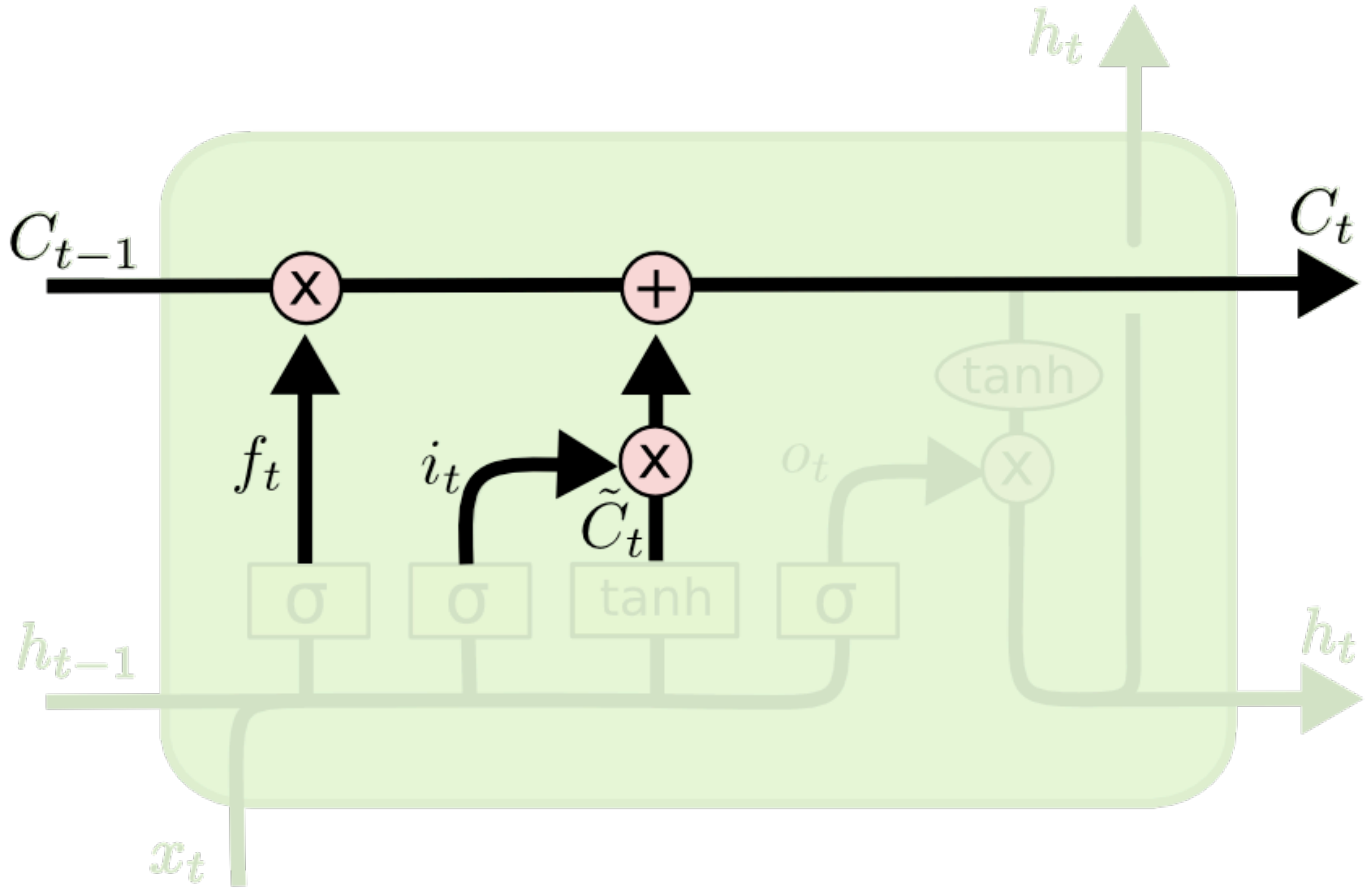
Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

* slide from Dhruv Batra

LSTM Intuition: Memory Update

Forget what needs to be forgotten + memorize what needs to be remembered

0.1
-0.6
0.1
0.55
-0.67
0.4
0.01
0.7
…
0.9



$$\begin{bmatrix} .. \\ .. \\ .. \\ .. \\ .. \\ .. \\ .. \\ .. \\ \dots \\ .. \end{bmatrix} = \begin{bmatrix} 0.1 \\ -0.6 \\ 0.1 \\ 0.55 \\ -0.67 \\ 0.4 \\ 0.01 \\ 0.7 \\ \dots \\ 0.9 \end{bmatrix} \text{X} \begin{bmatrix} 0.1 \\ 0.4 \\ 0.1 \\ 0.6 \\ 1 \\ 0.7 \\ 0.1 \\ 0 \\ 1 \\ \dots \\ 0.3 \end{bmatrix} + \begin{bmatrix} 0.4 \\ 0.6 \\ 0.5 \\ 1 \\ 0.1 \\ 0.4 \\ 0.5 \\ 0.2 \\ \dots \\ 0.6 \end{bmatrix} \text{X} \begin{bmatrix} 0.3 \\ 0.5 \\ 0.62 \\ -0.34 \\ 0.43 \\ -0.78 \\ 0.1 \\ -0.45 \\ \dots \\ 0.9 \end{bmatrix}$$

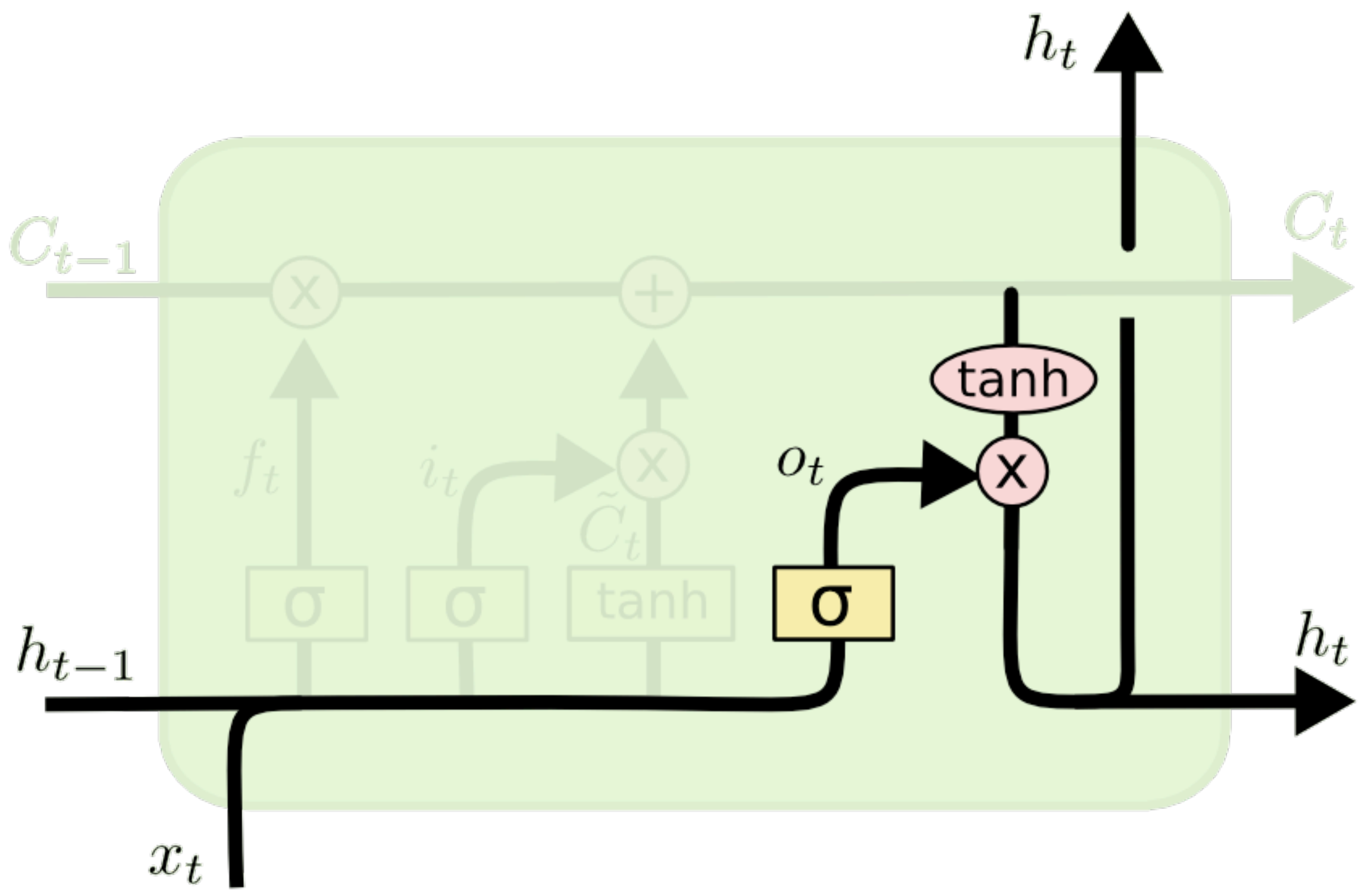
selective write

Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

* slide from Dhruv Batra

LSTM Intuition: Output Gate

Should we output this bit of information (e.g., to "deeper" LSTM layers)?



$$o_t = \sigma\left(W_o \left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

* slide from Dhruv Batra

LSTM Intuition: Additive Updates

Backpropagation from c_t to c_{t-1} only elementwise multiplication by f, no matrix multiply by W

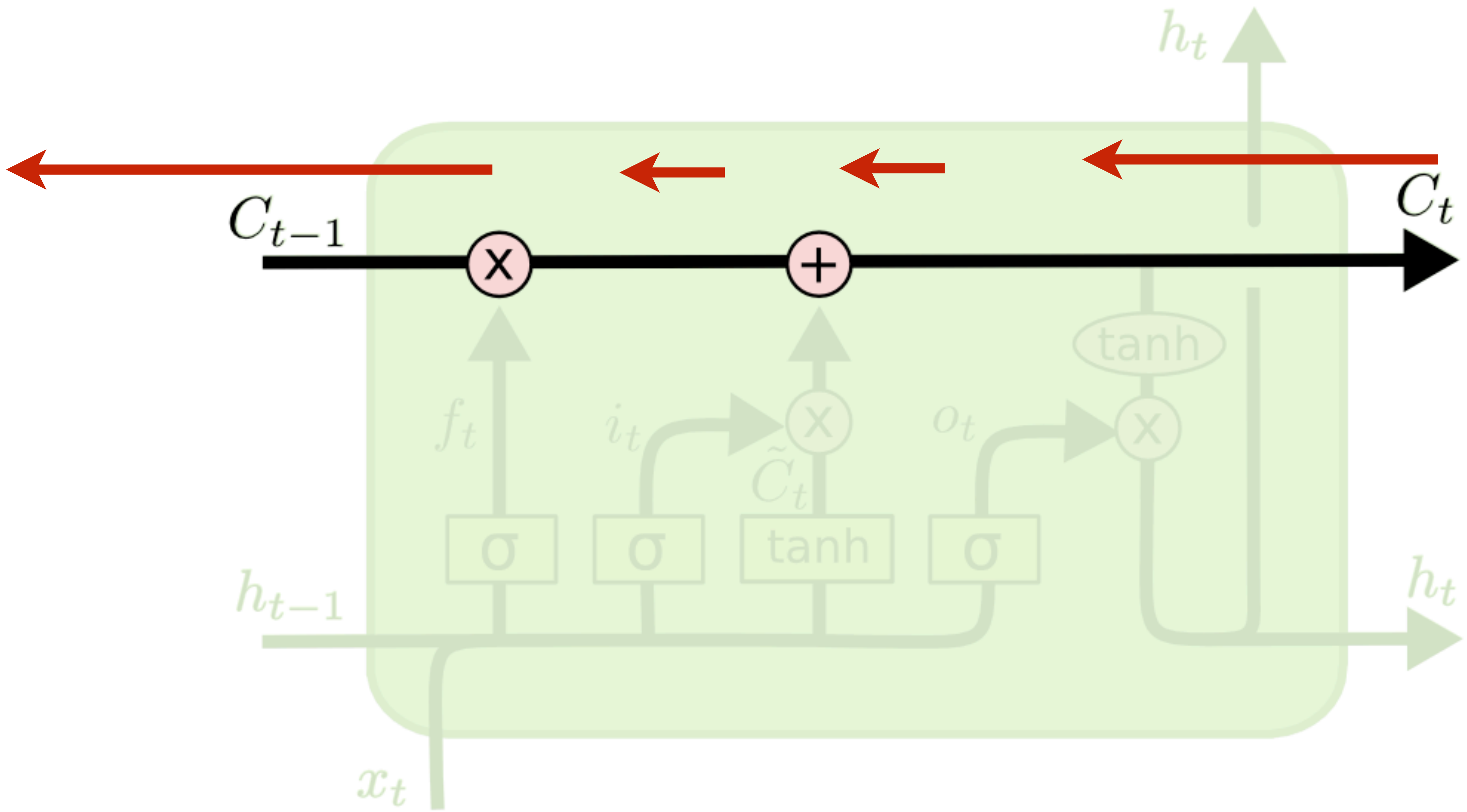
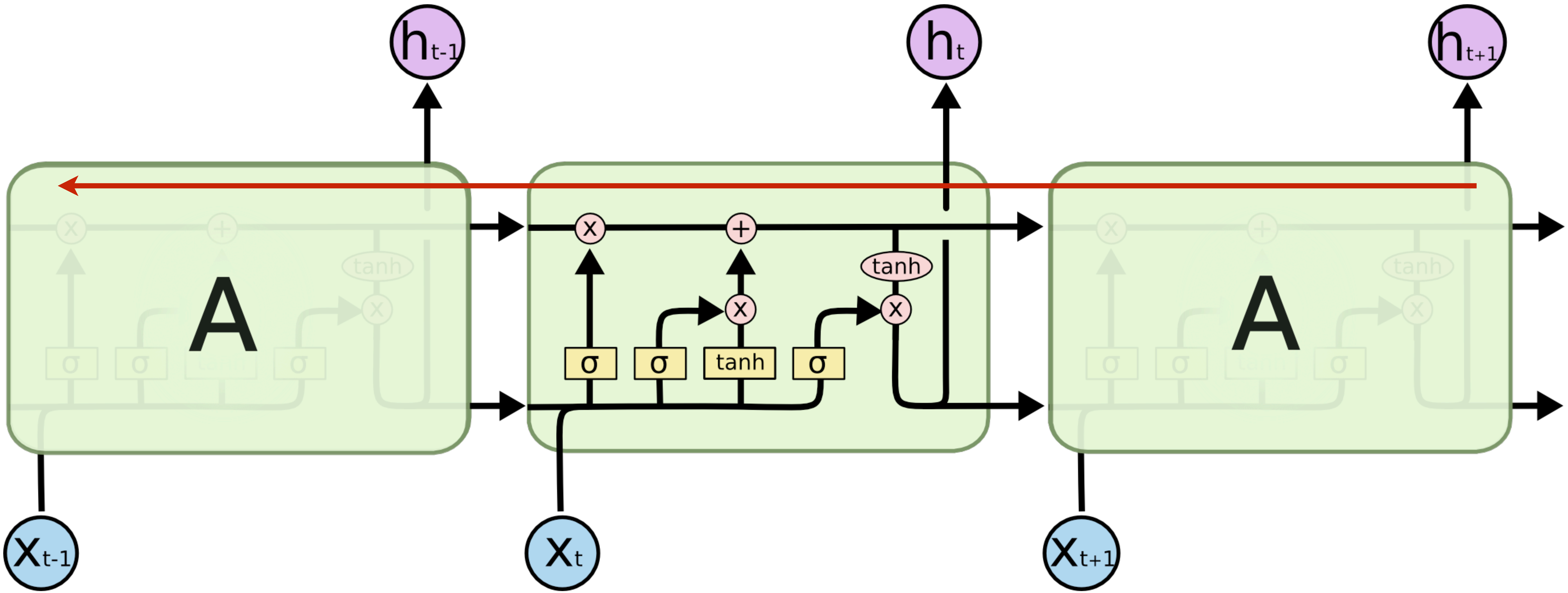


Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

* slide from Dhruv Batra

LSTM Intuition: Additive Updates



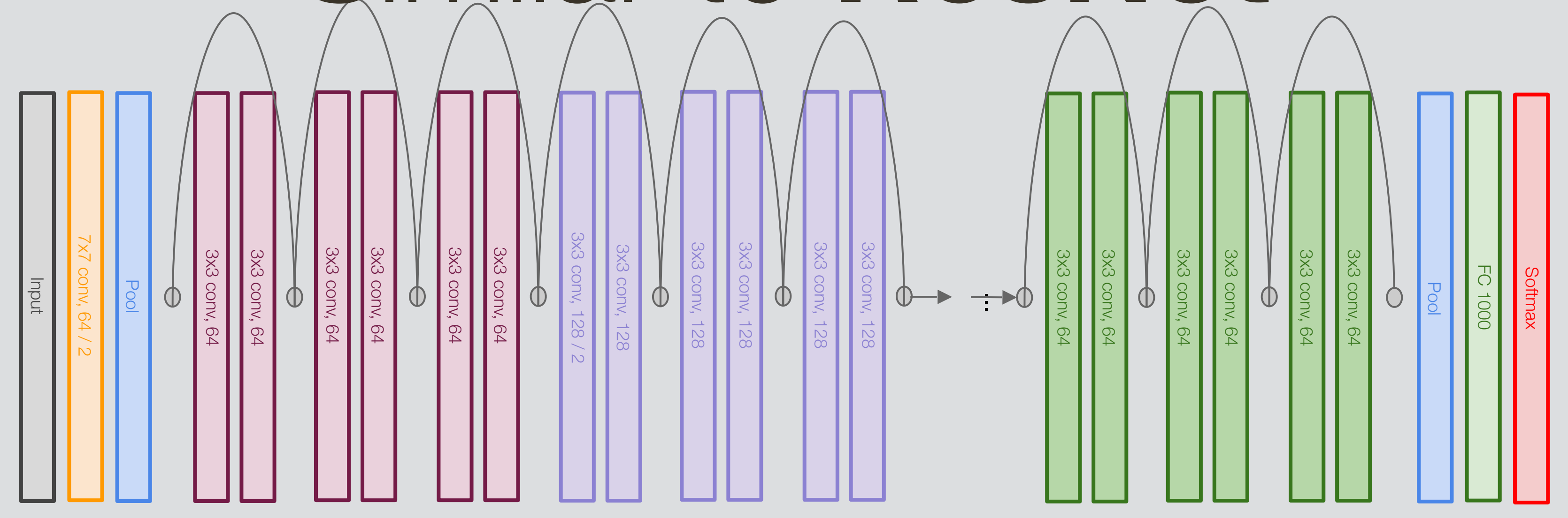
Uninterrupted gradient flow!

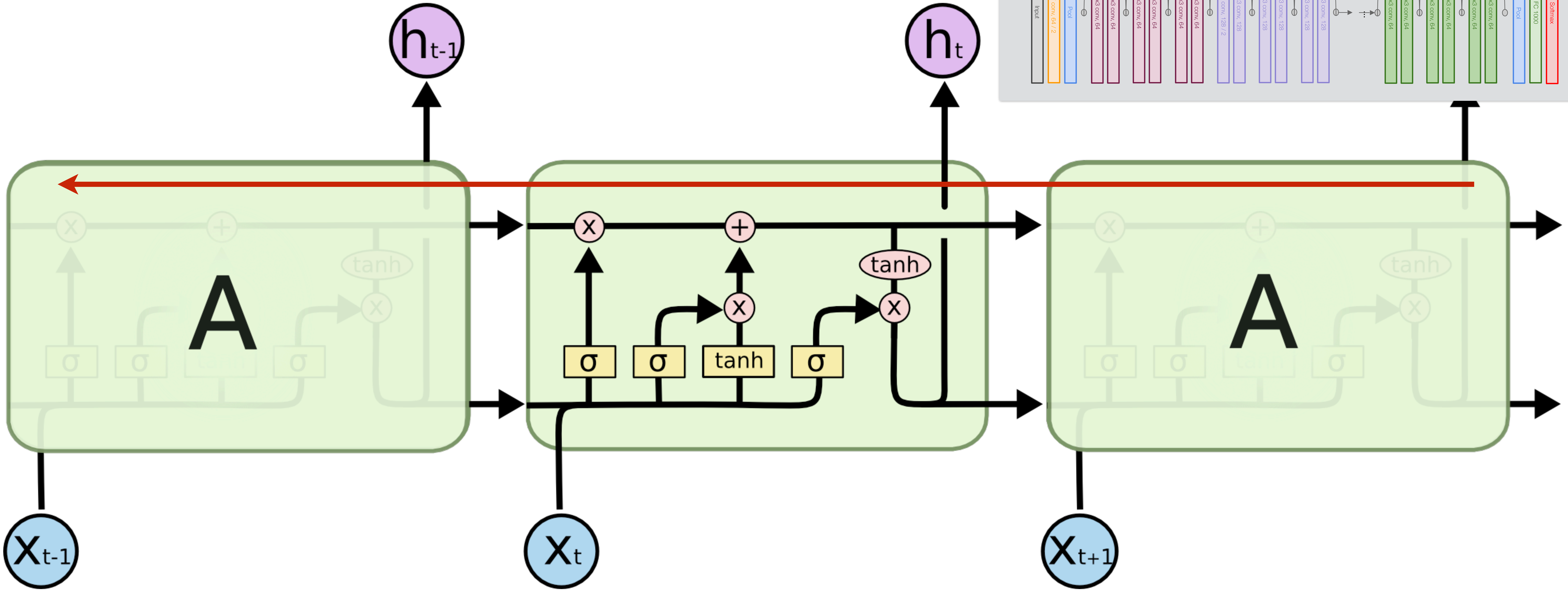
Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

* slide from Dhruv Batra

LSTM Intuition: Additive Updates

Similar to **ResNet**





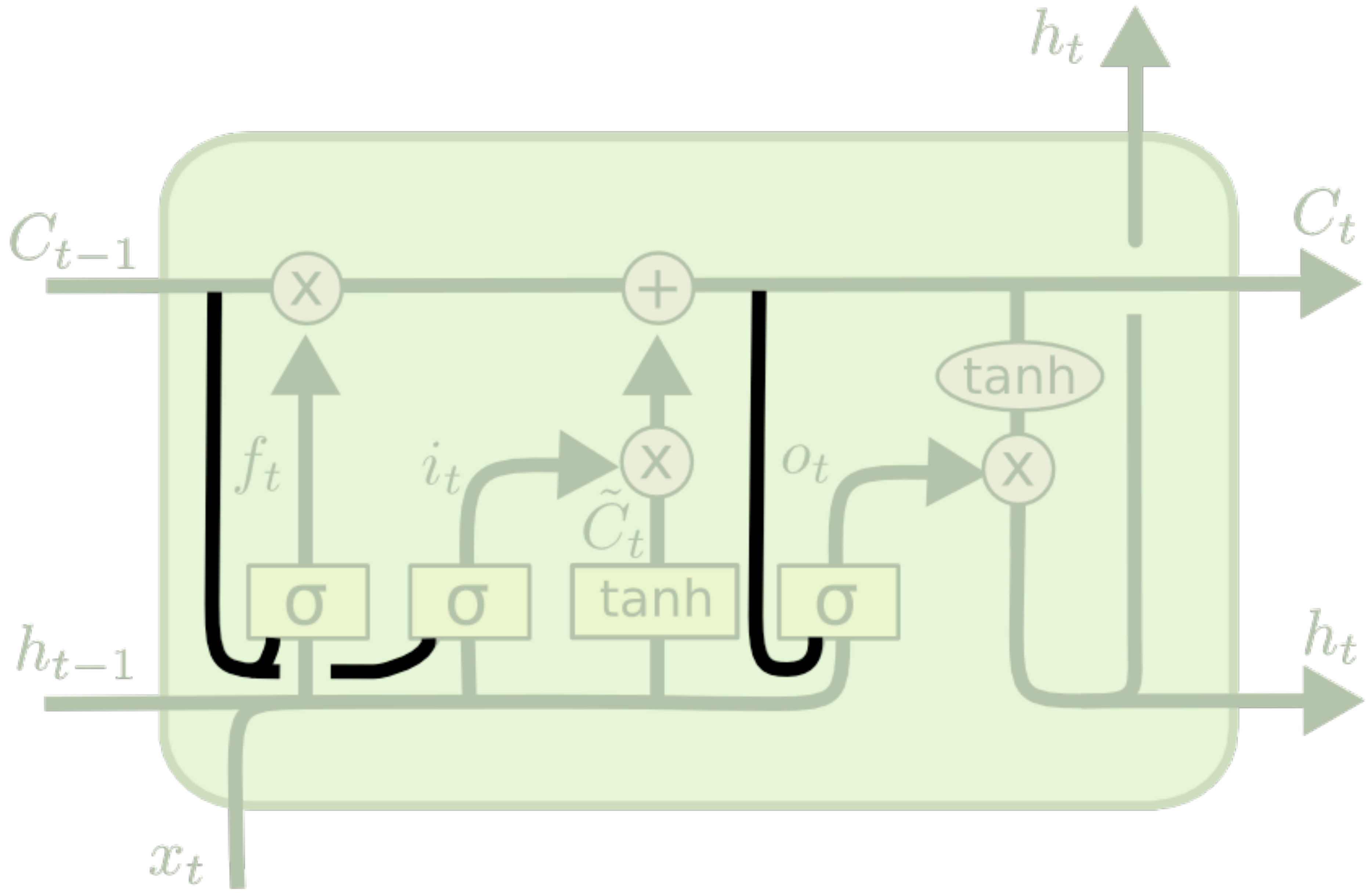
Uninterrupted gradient flow!

Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

* slide from Dhruv Batra

LSTM Variants: with Peephole Connections

Lets gates see the cell state / memory



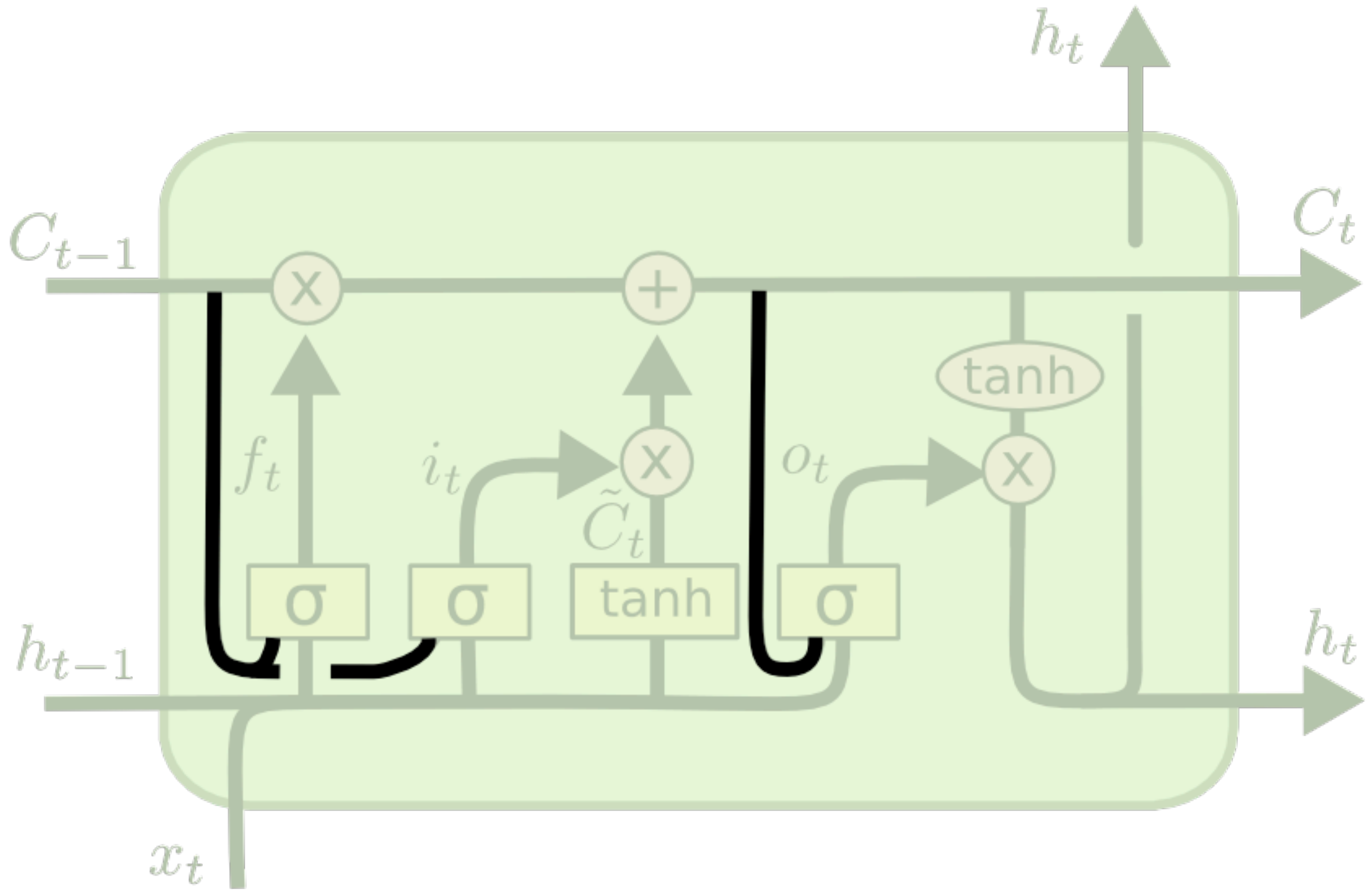
$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \;+\; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \;+\; b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \;+\; b_o\right)$$

Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

* slide from Dhruv Batra

LSTM Variants: with Peephole Connections

Lets gates see the cell state / memory



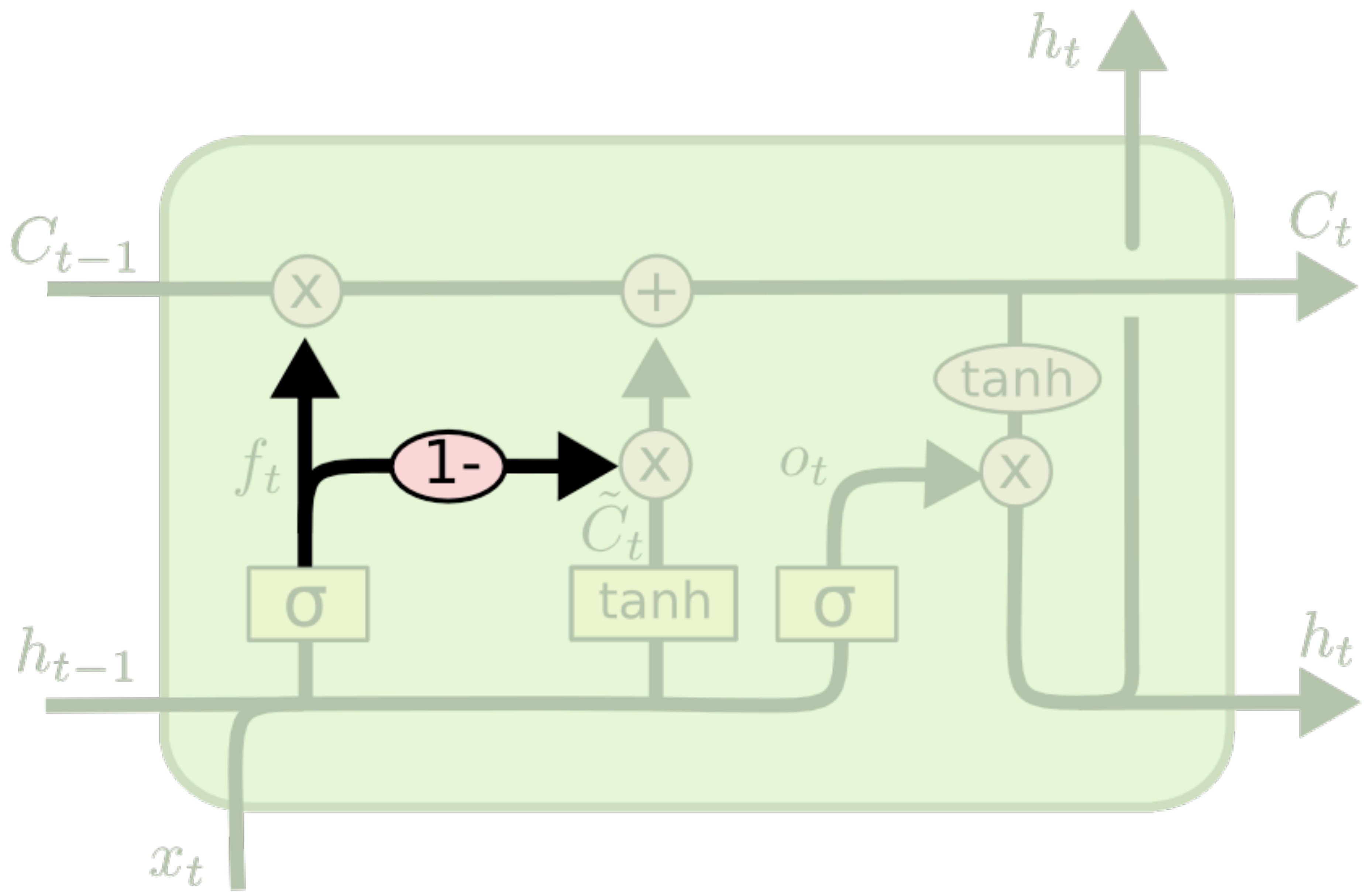
$$f_t = \sigma\left(W_f \cdot \left[\boldsymbol{C_{t-1}}, h_{t-1}, x_t\right] + b_f\right)$$
$$i_t = \sigma\left(W_i \cdot \left[\boldsymbol{C_{t-1}}, h_{t-1}, x_t\right] + b_i\right)$$
$$o_t = \sigma\left(W_o \cdot \left[\boldsymbol{C_t}, h_{t-1}, x_t\right] + b_o\right)$$

Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

* slide from Dhruv Batra

LSTM Variants: with Coupled Gates

Only memorize new information when you're forgetting old



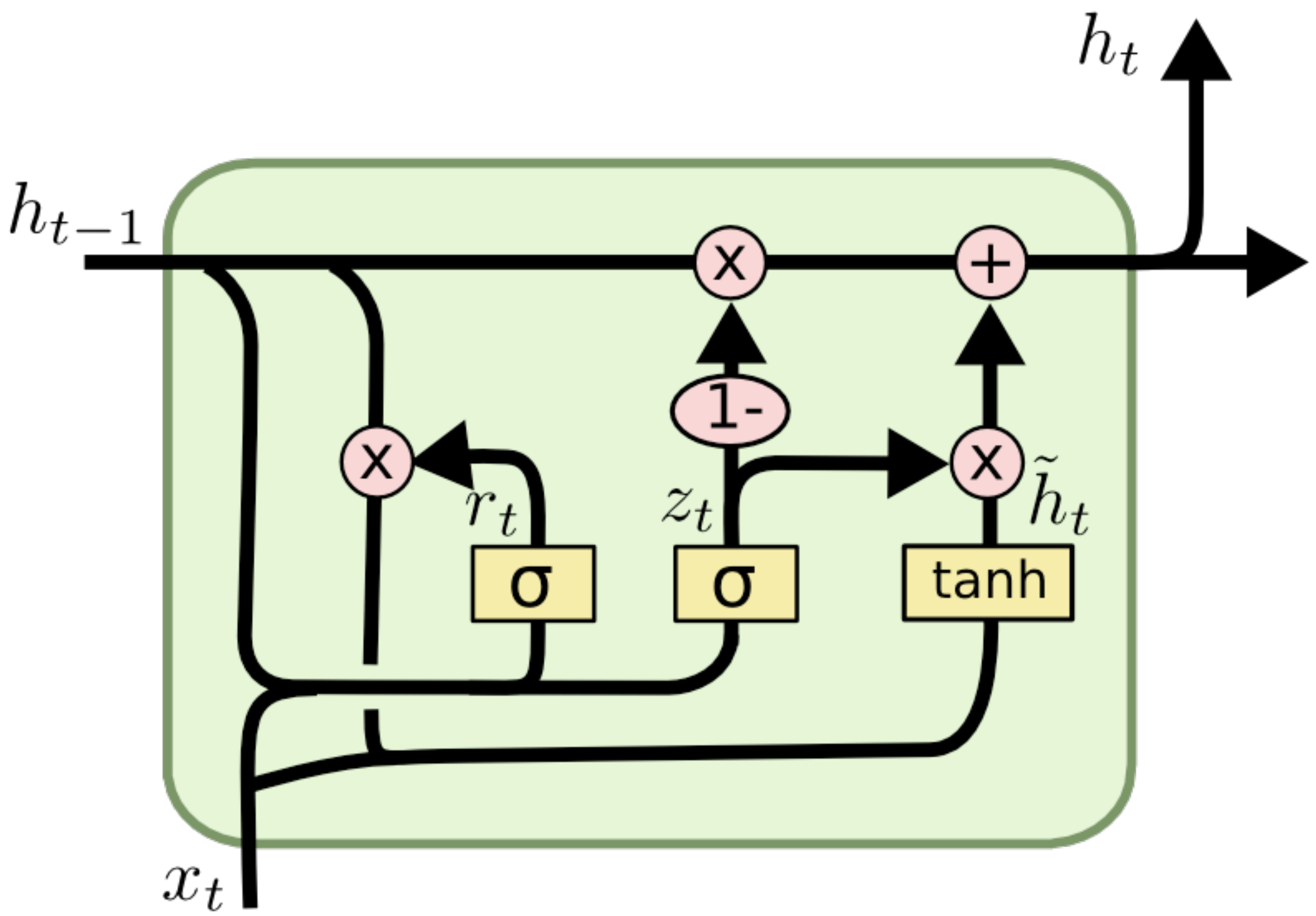
$$C_t = f_t * C_{t-1} + (1 - \boldsymbol{f_t}) * \tilde{C}_t$$

Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

* slide from Dhruv Batra

Gated Recurrent Unit (**GRU**)

No explicit memory; memory = hidden output



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

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

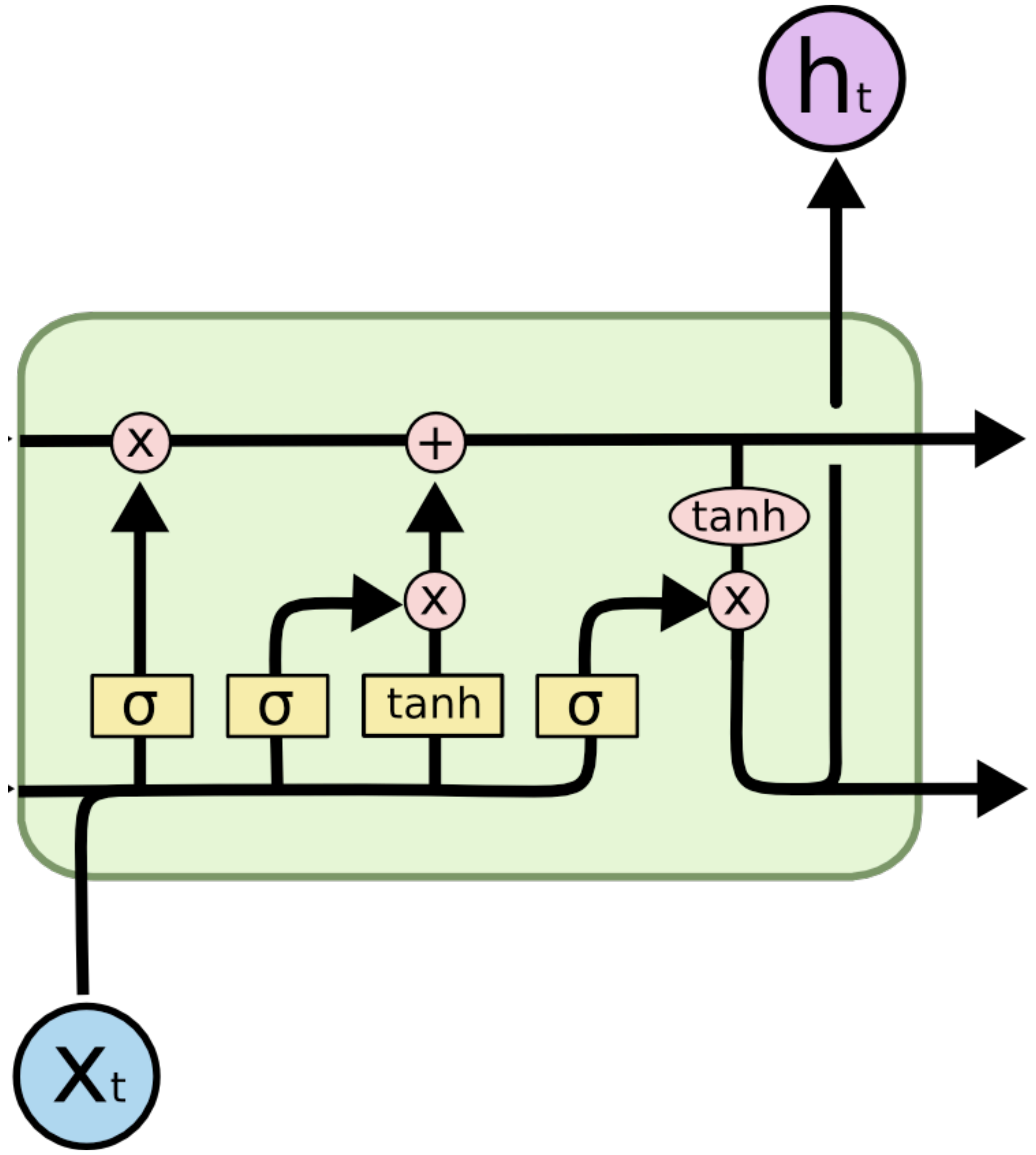
z = memorize new and forget old

Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

* slide from Dhruv Batra

LSTM/RNN **Challenges**

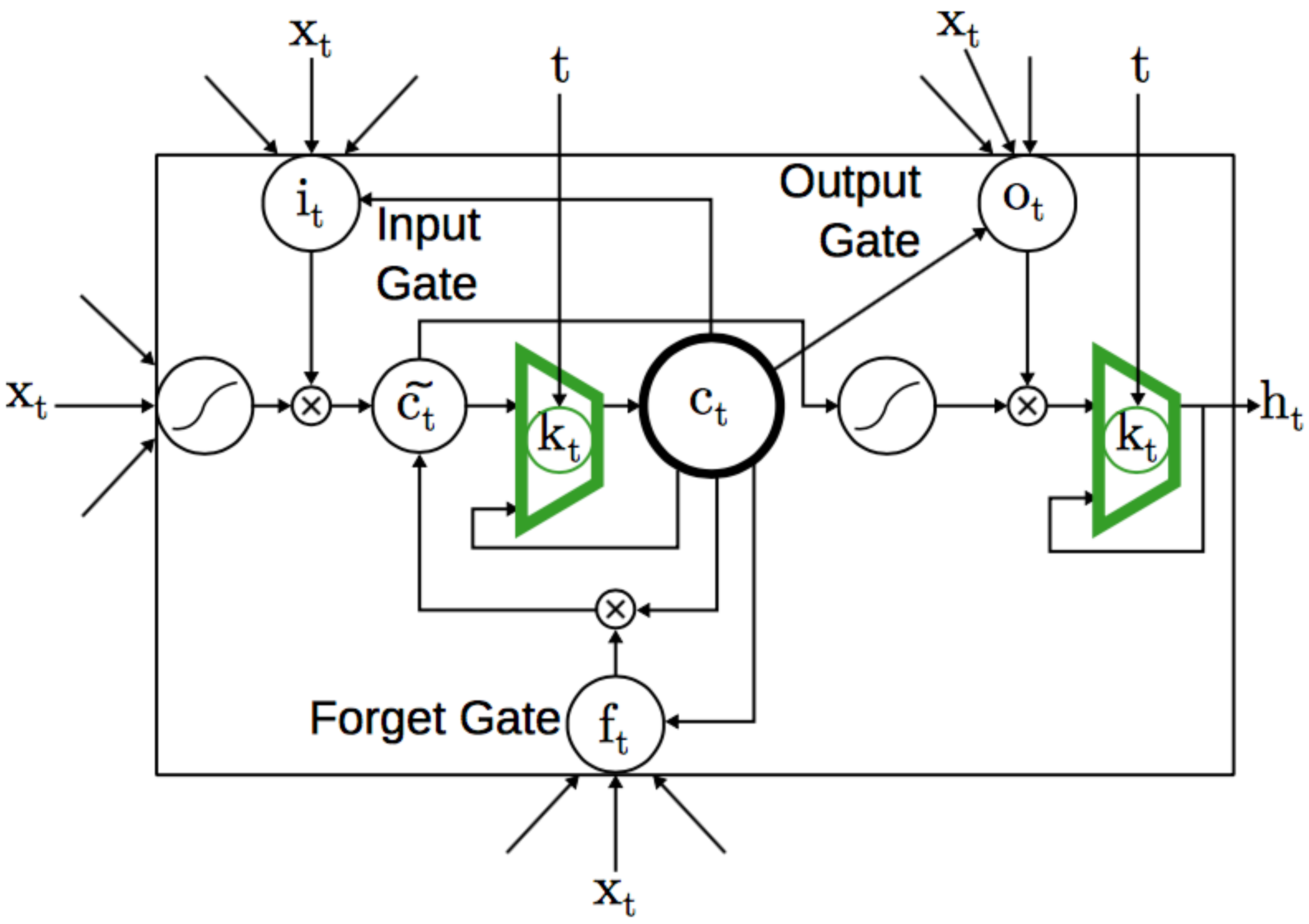
- LSTM can remember some history, but not too long
- LSTM assumes data is regularly sampled

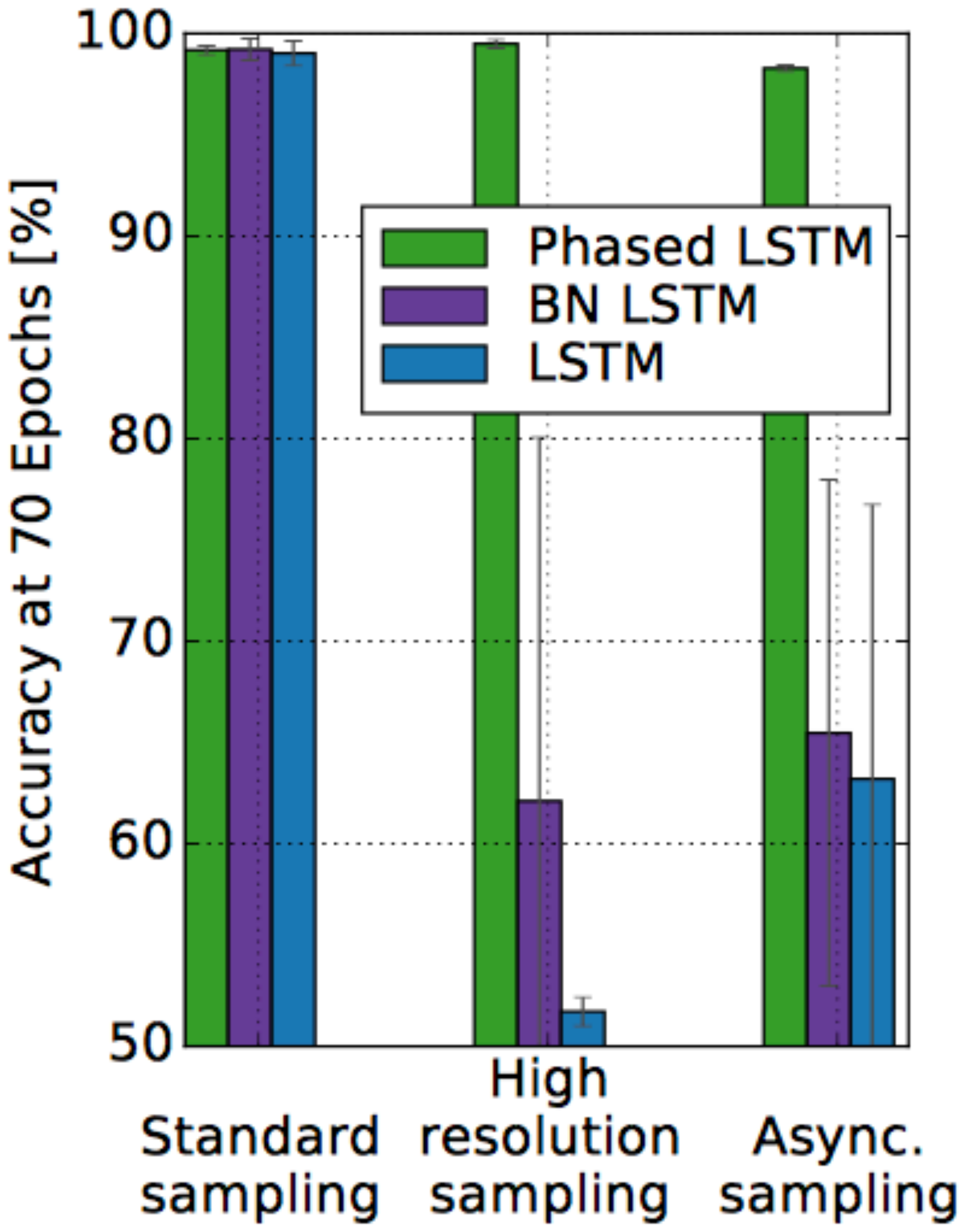


Phased LSTM

[Neil et al., 2016]

Gates are controlled by **phased** (periodic) **oscillations**



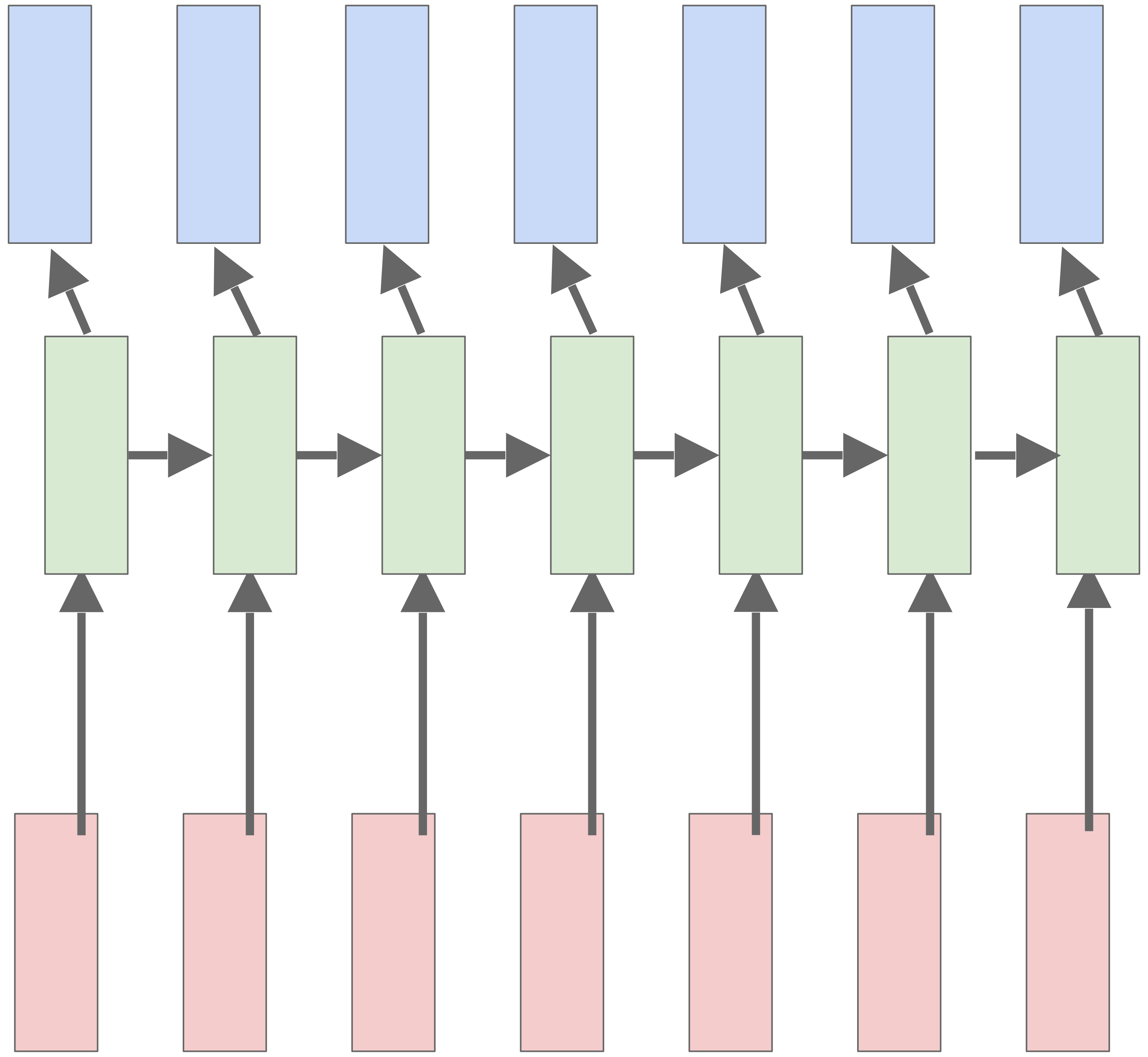


Bi-directional RNNs/LSTMs

$$y_t = W_{hy}h_t + b_y$$

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

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$



Bi-directional RNNs/LSTMs

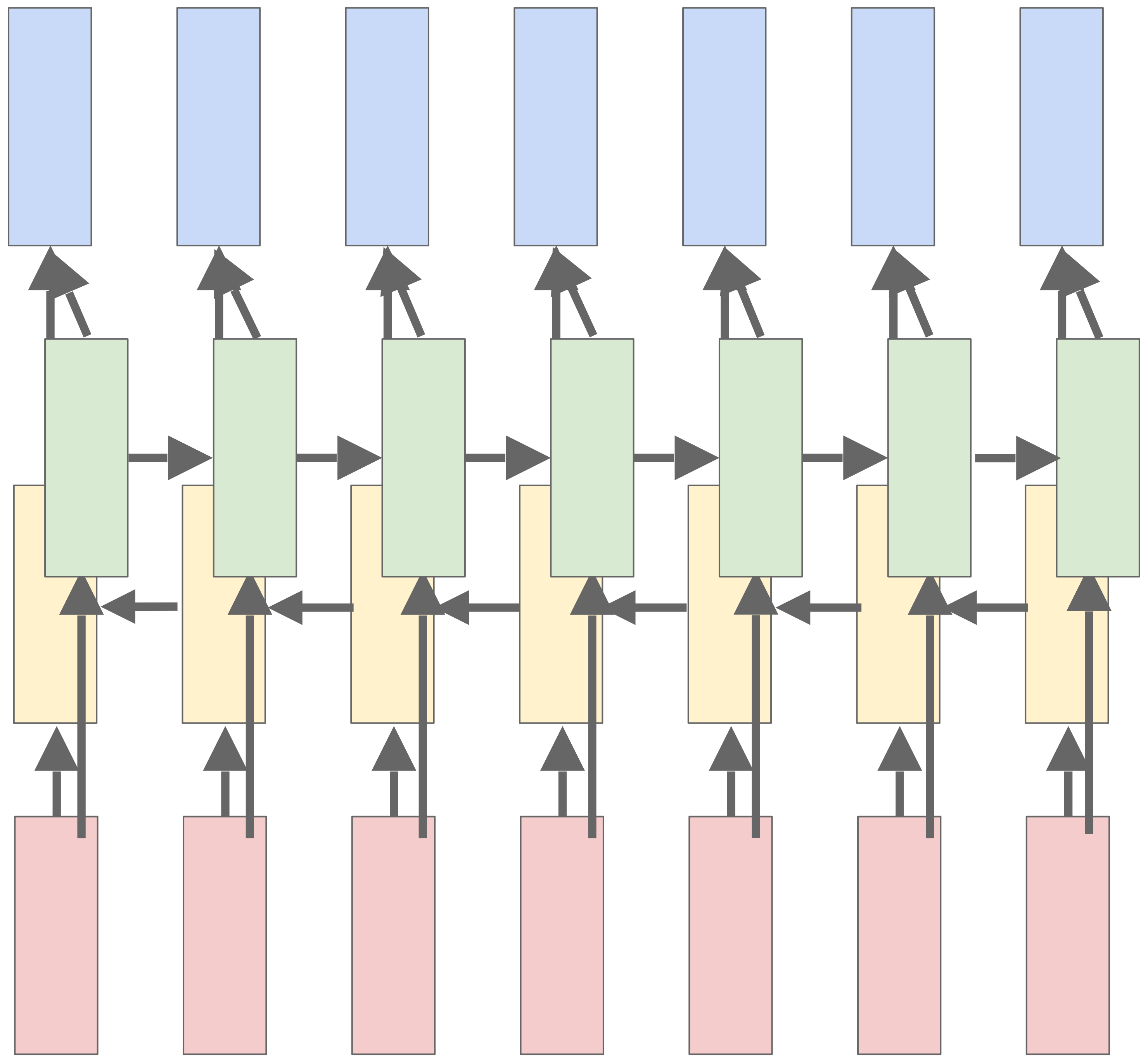
$$y_t = W_{hy}[\overrightarrow{h}_t, \overleftarrow{h}_t]^T + b_y$$

$$\overrightarrow{h}_t = f_{\overrightarrow{W}}(\overrightarrow{h}_{t-1}, x_t)$$

$$\overleftarrow{h}_t = f_{\overleftarrow{W}}(\overleftarrow{h}_{t+1}, x_t)$$

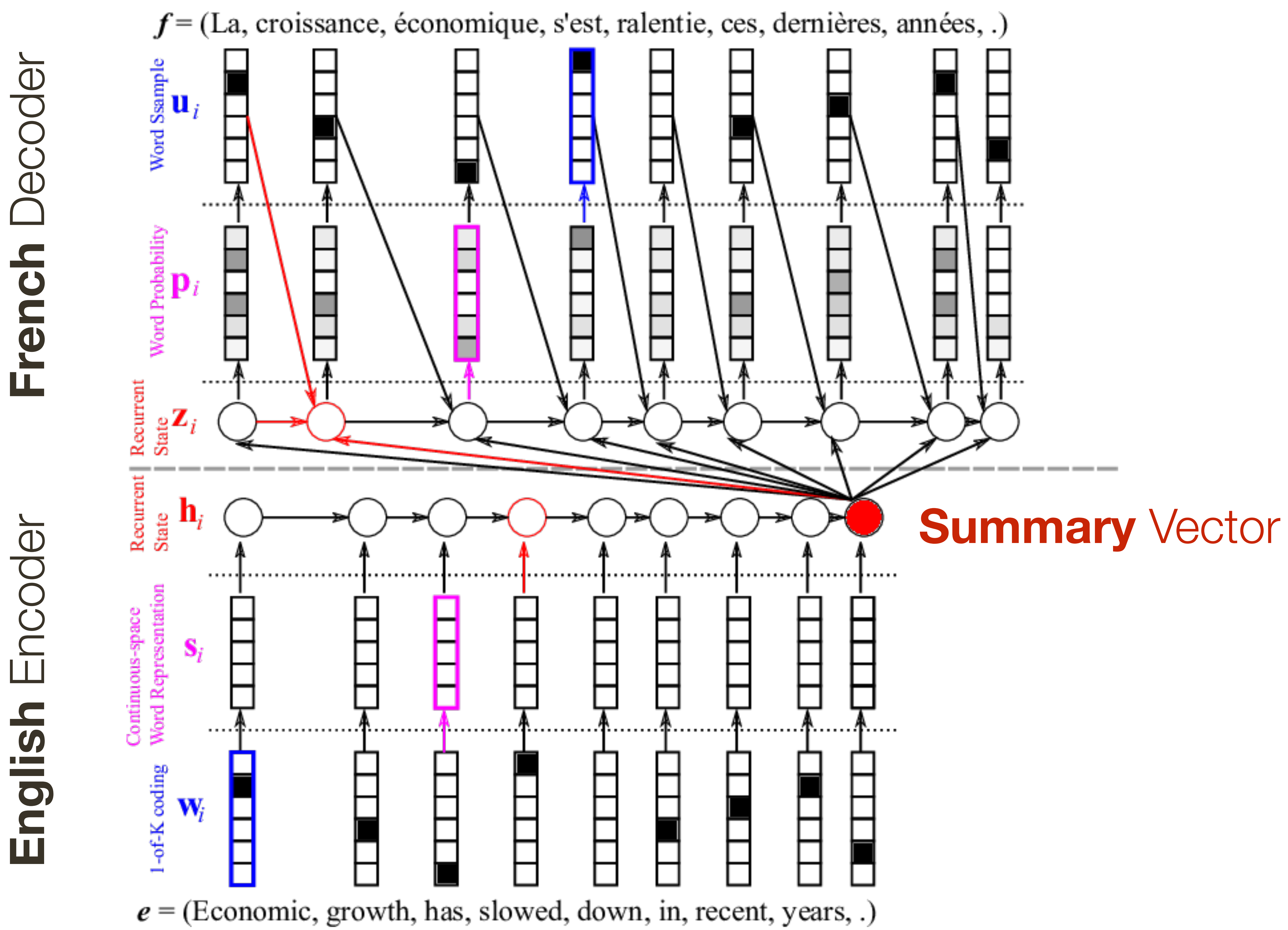
$$\overrightarrow{h}_t = \tanh(\overrightarrow{W}_{hh}\overrightarrow{h}_{t-1} + \overrightarrow{W}_{xh}x_t + \overrightarrow{b}_h)$$

$$\overleftarrow{h}_t = \tanh(\overleftarrow{W}_{hh}\overleftarrow{h}_{t+1} + \overleftarrow{W}_{xh}x_t + \overleftarrow{b}_h)$$



Attention Mechanisms and RNNs

Consider a **translation task**: This is one of the first neural translation models

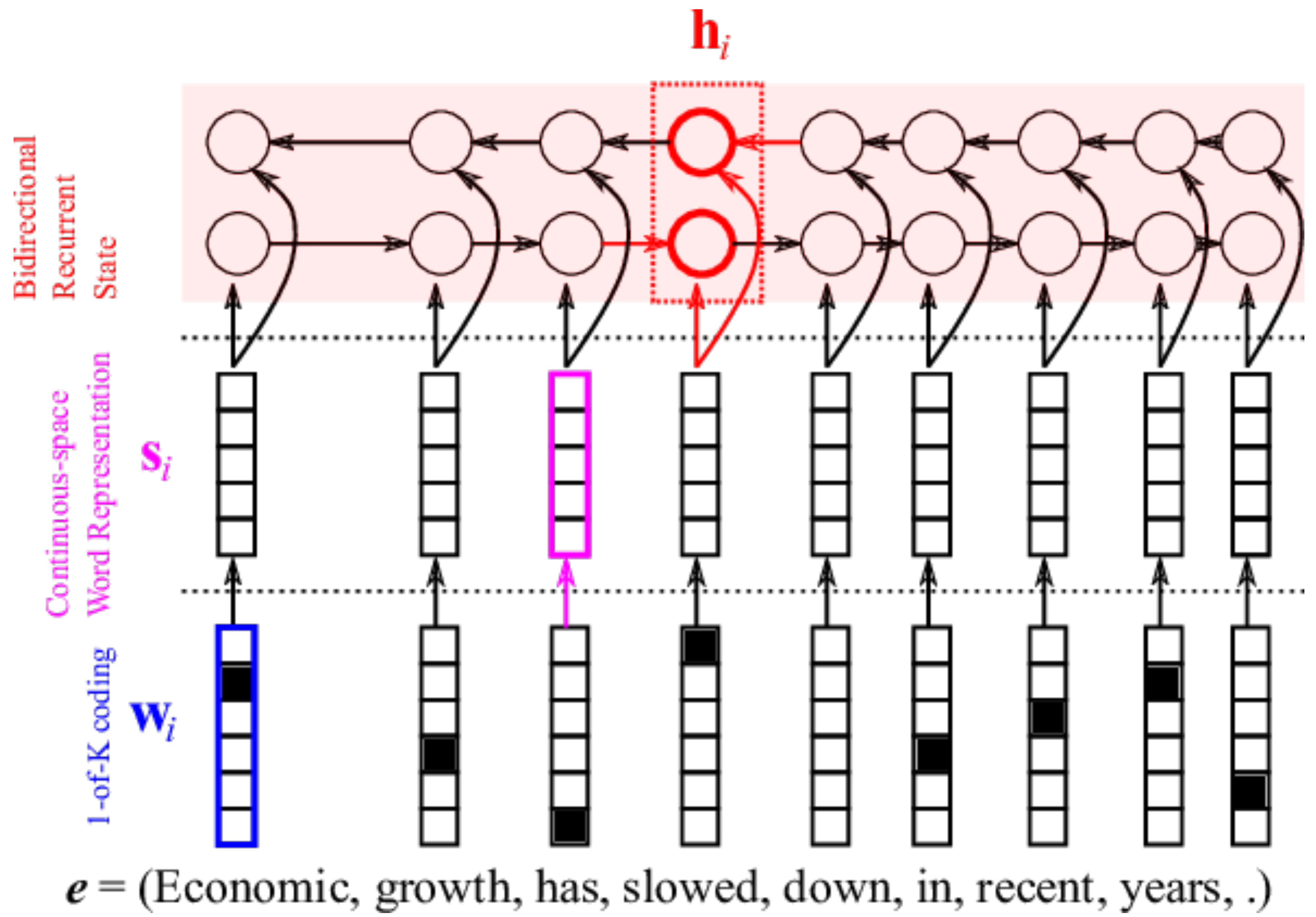


https://devblogs.nvidia.com/introduction-neural-machine-translation-gpus-part-3/

Attention Mechanisms and RNNs

Consider a **translation task** with a bi-directional encoder of the source language

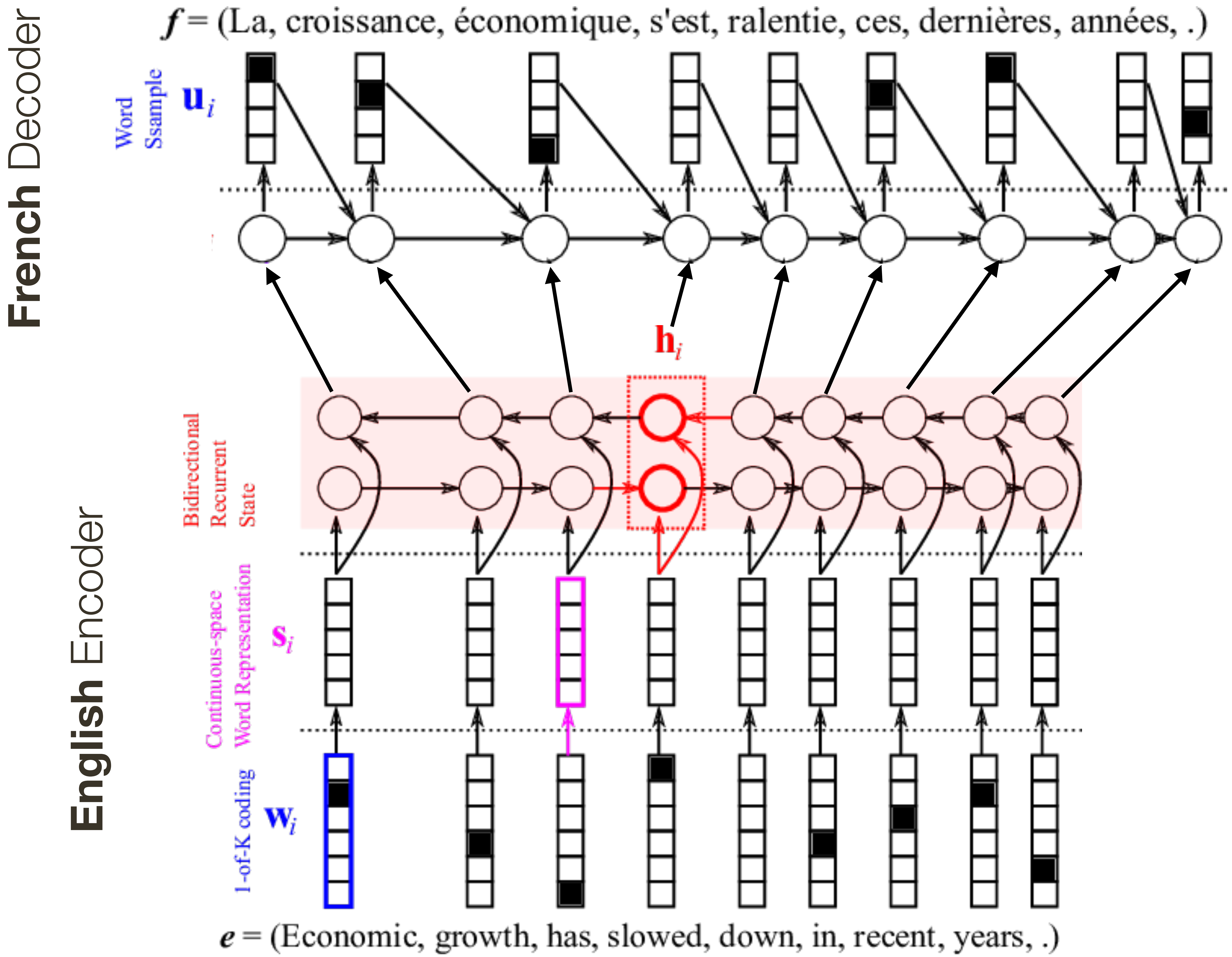
English Encoder



https://devblogs.nvidia.com/introduction-neural-machine-translation-gpus-part-3/

Attention Mechanisms and RNNs

Consider a **translation task** with a bi-directional encoder of the source language



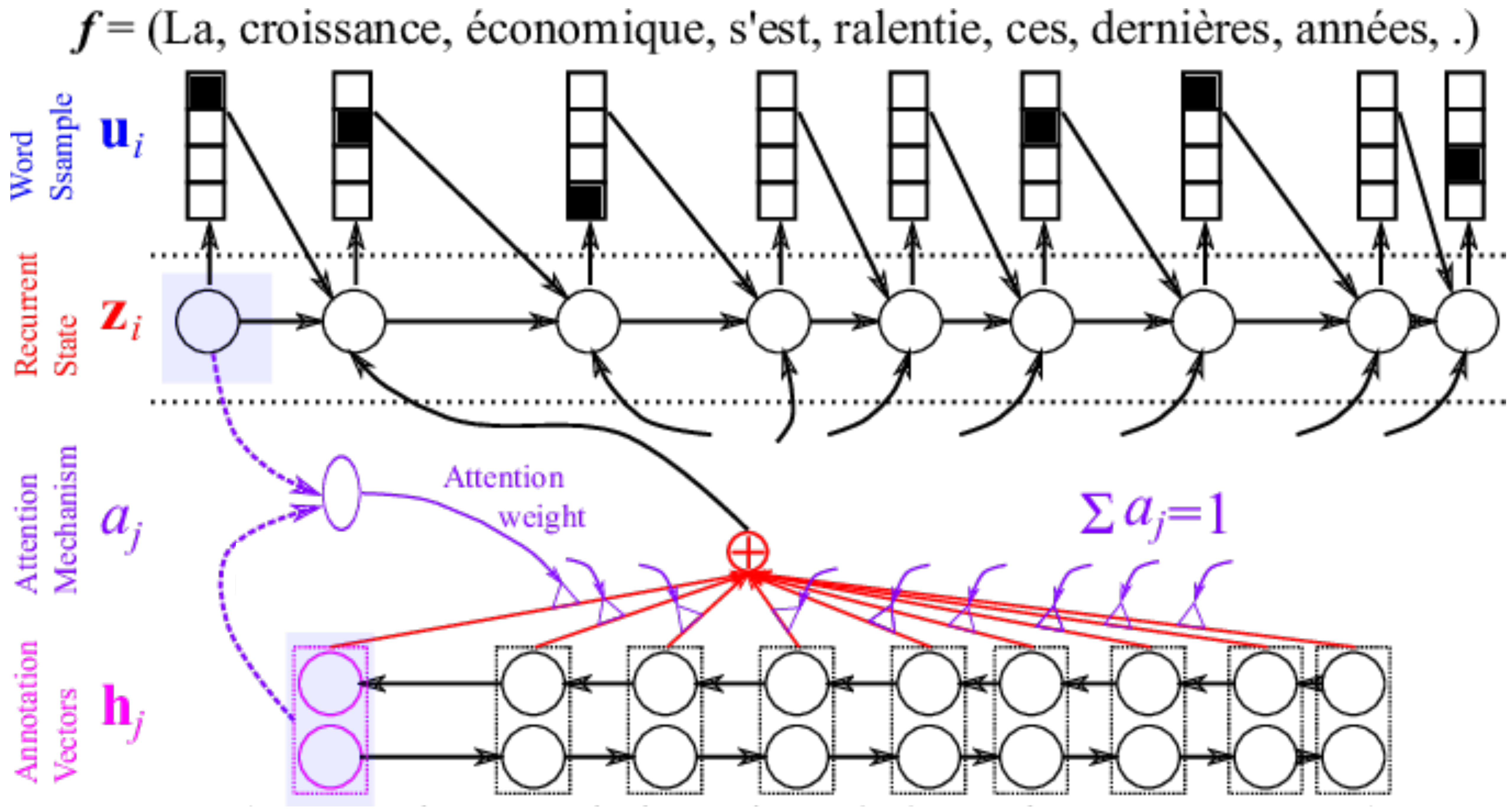
https://devblogs.nvidia.com/introduction-neural-machine-translation-gpus-part-3/

Attention Mechanisms and RNNs

Consider a **translation task** with a bi-directional encoder of the source language

[Cho et al., 2015]

French Decoder



Build a **small neural network** (one layer) with softmax output that takes

(1) everything decoded so far and (encoded by previous decoder state Zi)

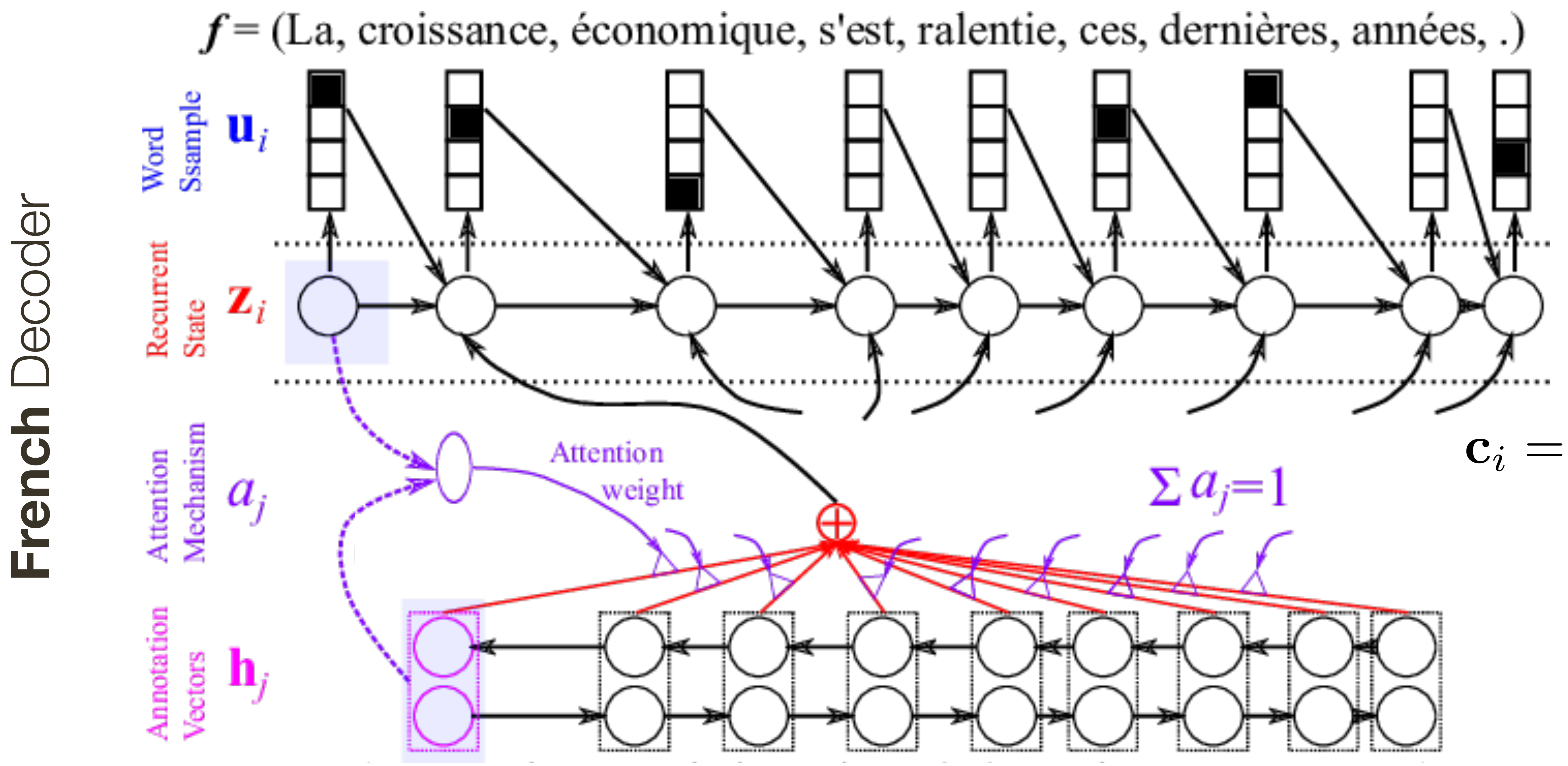
(2) encoding of the current word (encoded by the hidden state of encoder hj)

and predicts **relevance of every source word** towards next translation

https://devblogs.nvidia.com/introduction-neural-machine-translation-gpus-part-3/

Attention Mechanisms and RNNs

Consider a **translation task** with a bi-directional encoder of the source language



[Cho et al., 2015]

$$\mathbf{c}_i = \sum_{j=1}^{T} \alpha_j \mathbf{h}_j$$

Build a **small neural network** (one layer) with softmax output that takes

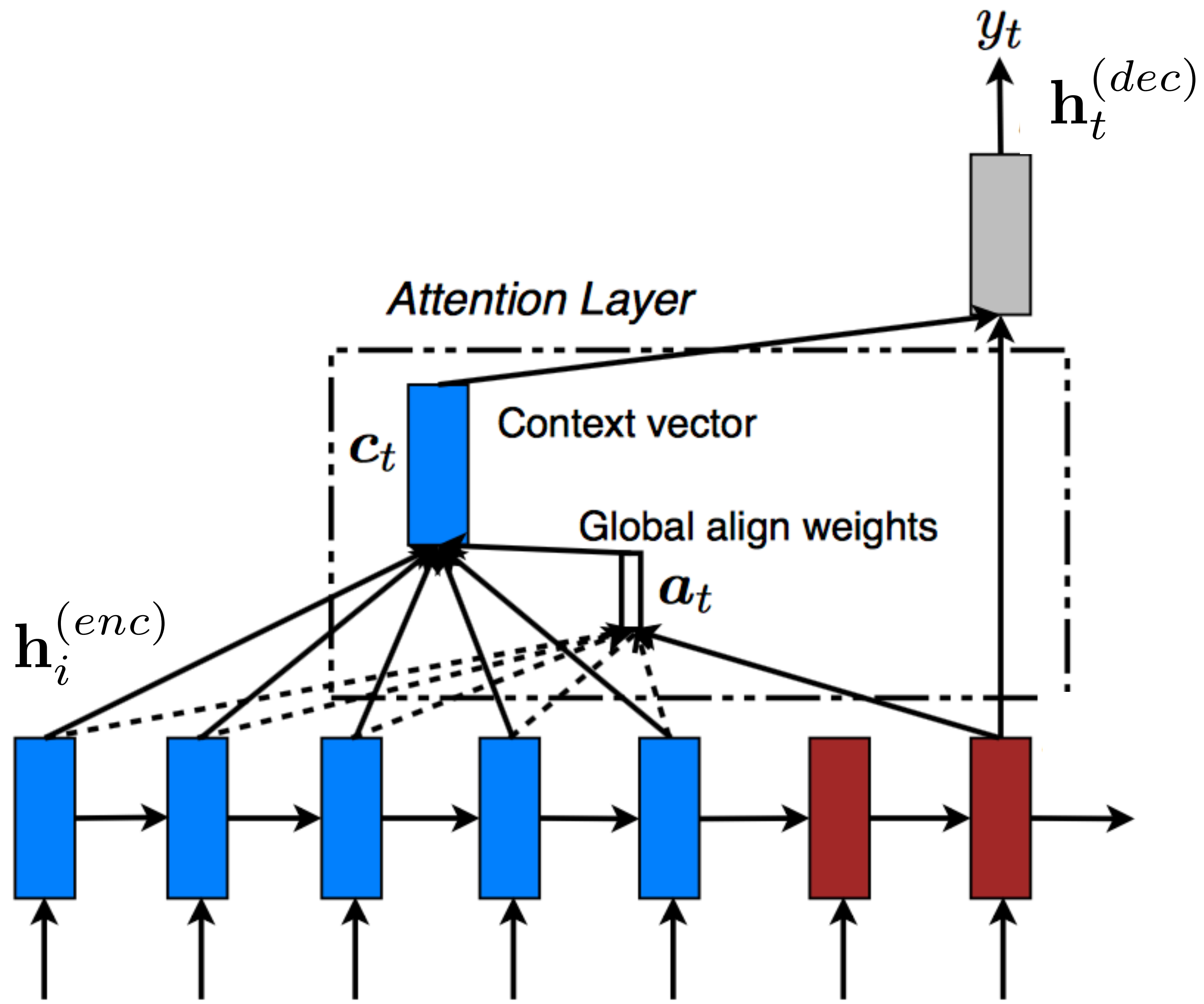
(1) everything decoded so far and (encoded by previous decoder state Zi)

(2) encoding of the current word (encoded by the hidden state of encoder hj)

and predicts **relevance of every source word** towards next translation

https://devblogs.nvidia.com/introduction-neural-machine-translation-gpus-part-3/

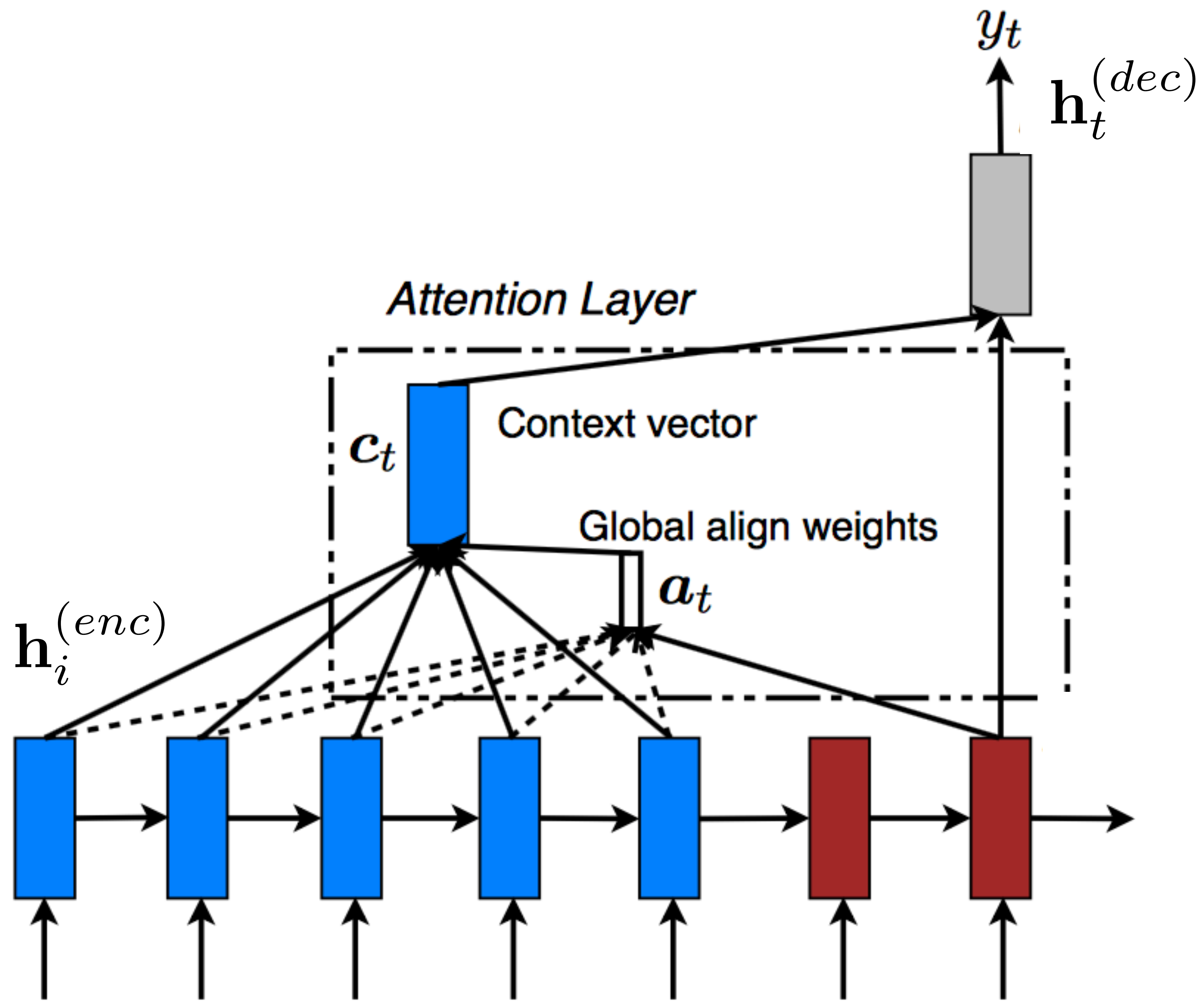
Soft **Attention** in details



Soft **Attention** in details

$$\beta_{i,t} = score(\mathbf{h}_i^{(enc)}, \mathbf{h}_t^{(dec)})$$

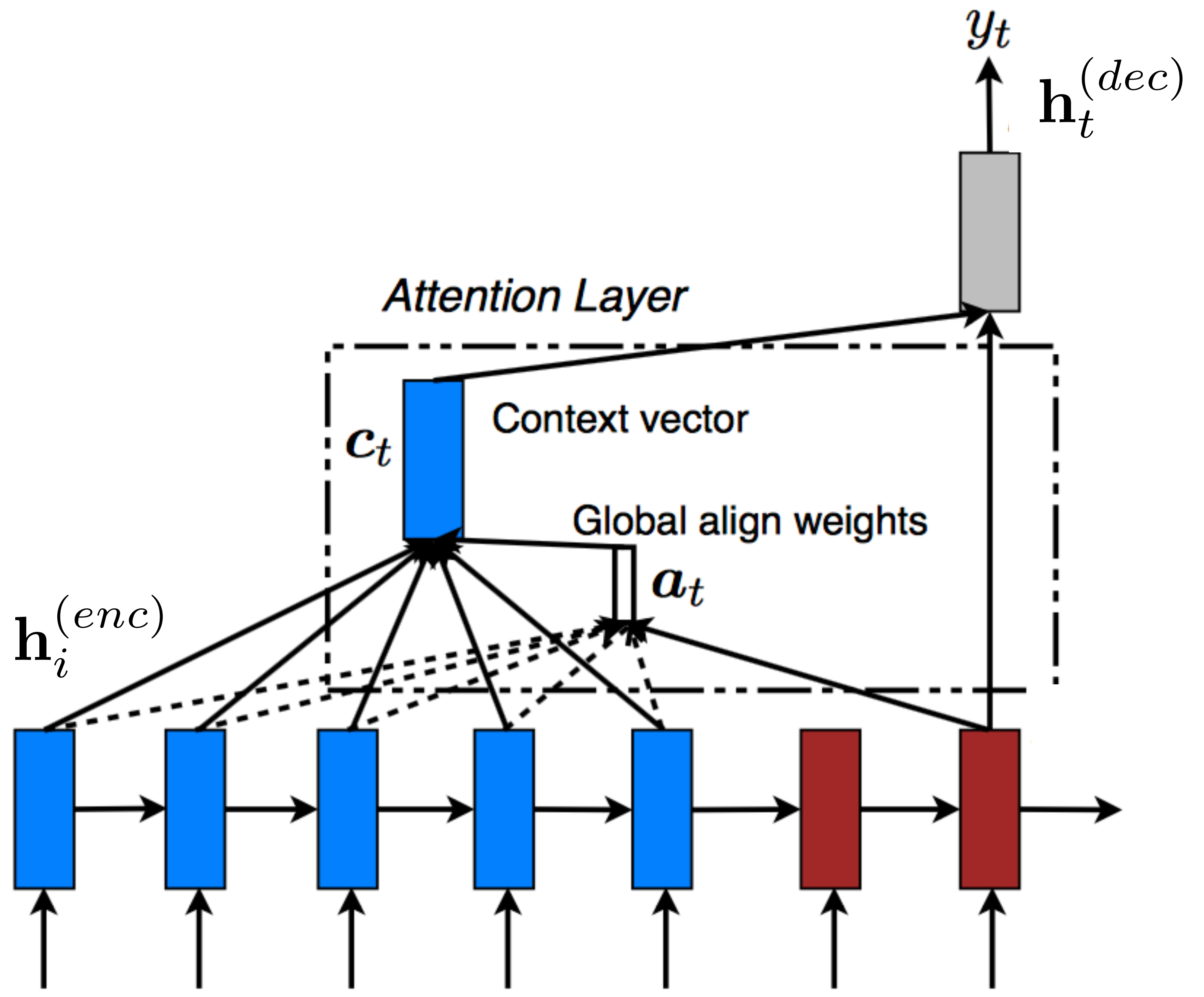
Relevance of encoding at token i for decoding token t



Soft **Attention** in details

$$\beta_{i,t} = score(\mathbf{h}_i^{(enc)}, \mathbf{h}_t^{(dec)})$$

Relevance of encoding at token i for decoding token t



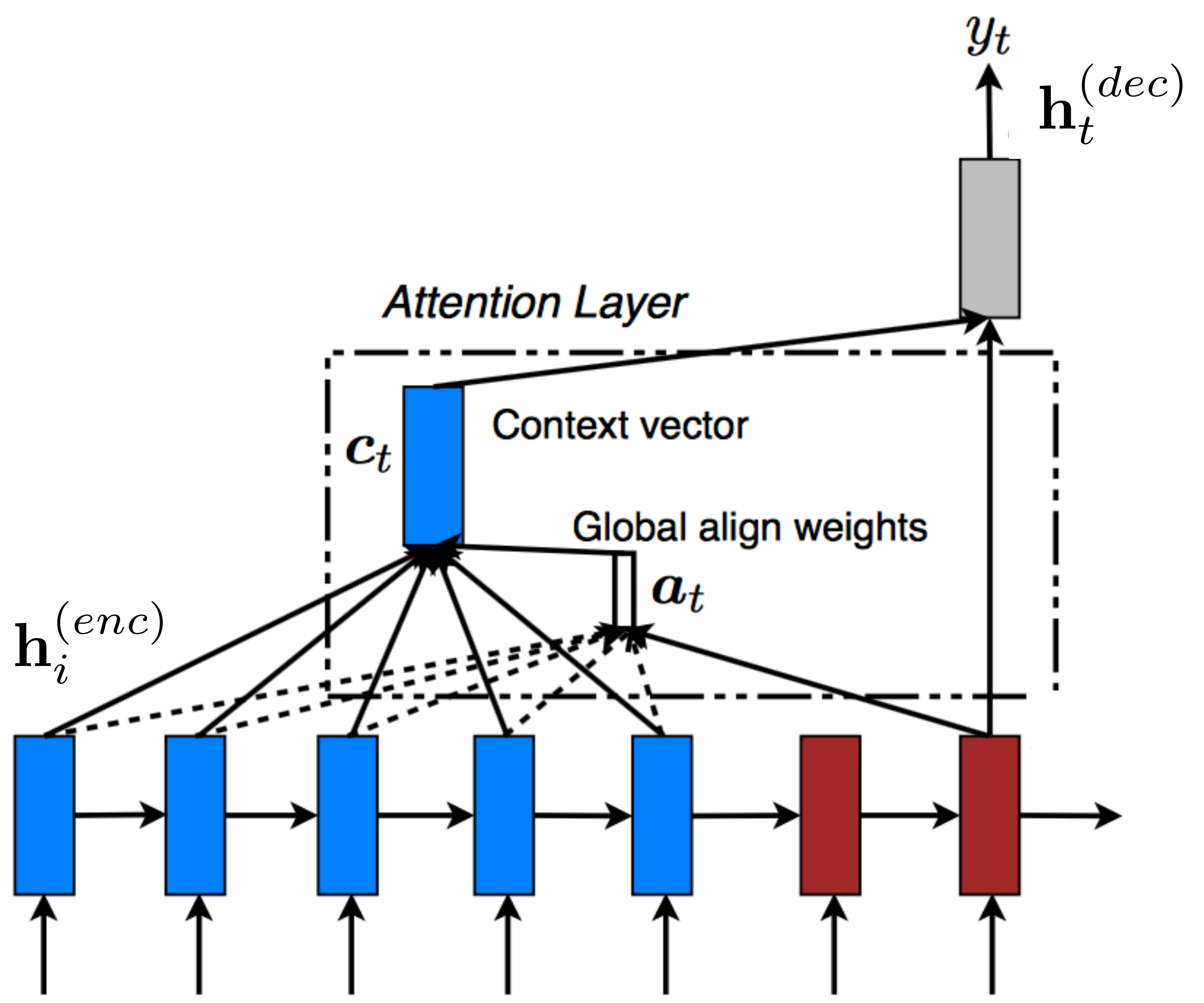
$$\alpha_{i,t} = \text{Softmax}(\beta_{i,t})$$

Normalize the weights to sum to 1

Soft **Attention** in details

$$\beta_{i,t} = score(\mathbf{h}_i^{(enc)}, \mathbf{h}_t^{(dec)})$$

Relevance of encoding at token i for decoding token t



$$\alpha_{i,t} = \text{Softmax}(\beta_{i,t})$$

Normalize the weights to sum to 1

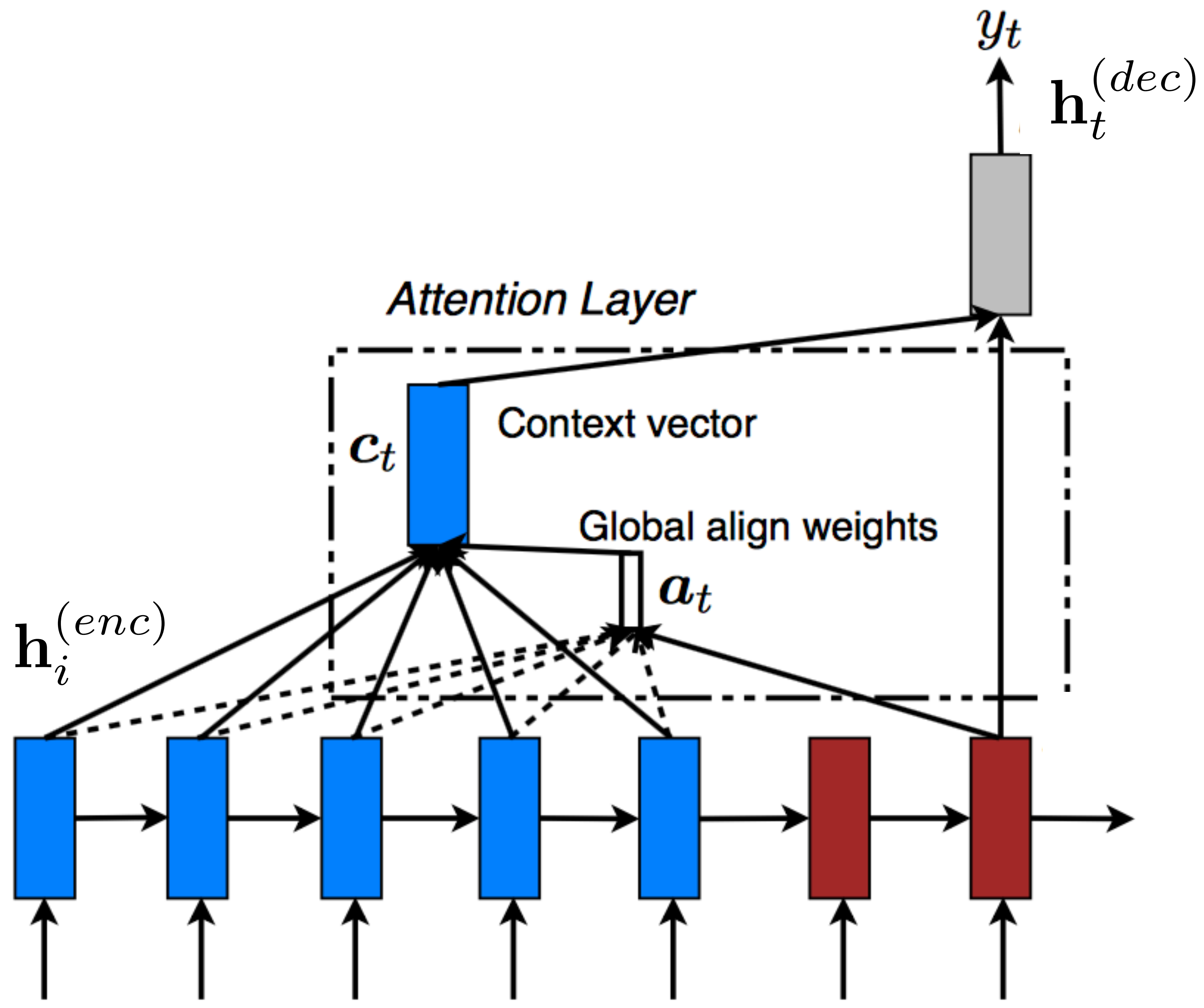
$$\mathbf{c}_t = \sum_i \alpha_{i,t} \mathbf{h}_i^{(enc)}$$

Form a context vector that would simply be added to the standard decoder input

Soft **Attention** in details

$$\beta_{i,t} = score(\mathbf{h}_i^{(enc)}, \cancel{\mathbf{h}_t^{(dec)}})$$

Relevance of encoding at token i for decoding token t



$$\alpha_{i,t} = \text{Softmax}(\beta_{i,t})$$

Normalize the weights to sum to 1

$$\mathbf{c}_t = \sum_i \alpha_{i,t}\mathbf{h}_i^{(enc)}$$

Form a context vector that would simply be added to the standard decoder input

Soft **Attention** in details

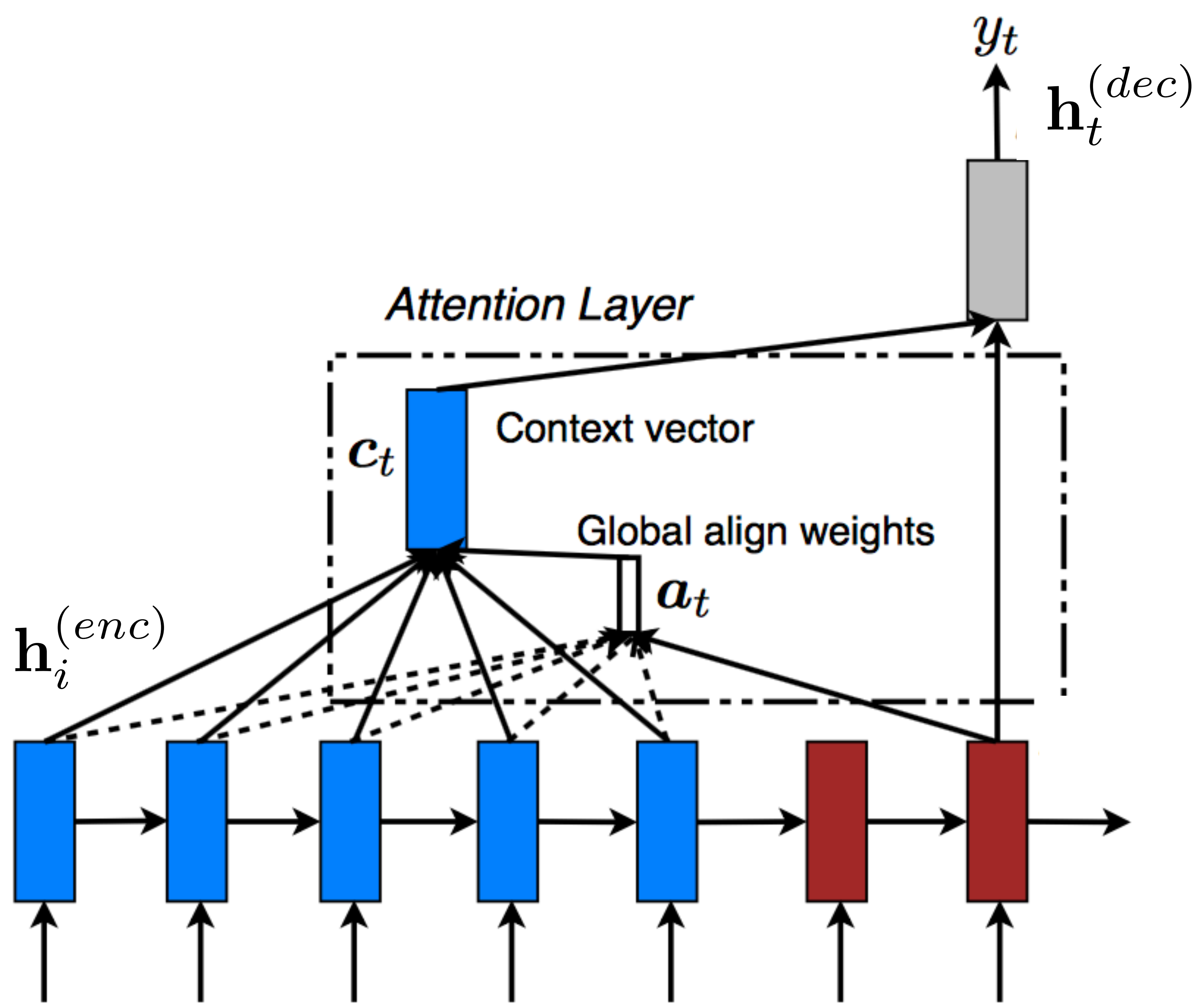
$$\beta_{i,t} = score(\mathbf{h}_i^{(enc)}, \cancel{\mathbf{h}_t^{(dec)}})$$

Relevance of encoding at
token i for decoding token t

$$\beta_{i,t} = score(\mathbf{h}_i^{(enc)}, \mathbf{x}_t^{(dec)})$$

$$\beta_{i,t} = score(\mathbf{h}_i^{(enc)}, \mathbf{h}_{t-1}^{(dec)})$$

$$\beta_{i,t} = score(\mathbf{h}_i^{(enc)}, [\mathbf{x}_t^{(dec)}, \mathbf{h}_{t-1}^{(dec)}])$$



$$\alpha_{i,t} = \text{Softmax}(\beta_{i,t})$$

Normalize the weights
to sum to 1

$$\mathbf{c}_t = \sum_i \alpha_{i,t} \mathbf{h}_i^{(enc)}$$

Form a context vector that would simply be added to the standard decoder input

Soft **Attention** in details

$$\beta_{i,t} = score(\mathbf{h}_i^{(enc)}, \cancel{\mathbf{h}_t^{(dec)}})$$

Relevance of encoding at token i for decoding token t

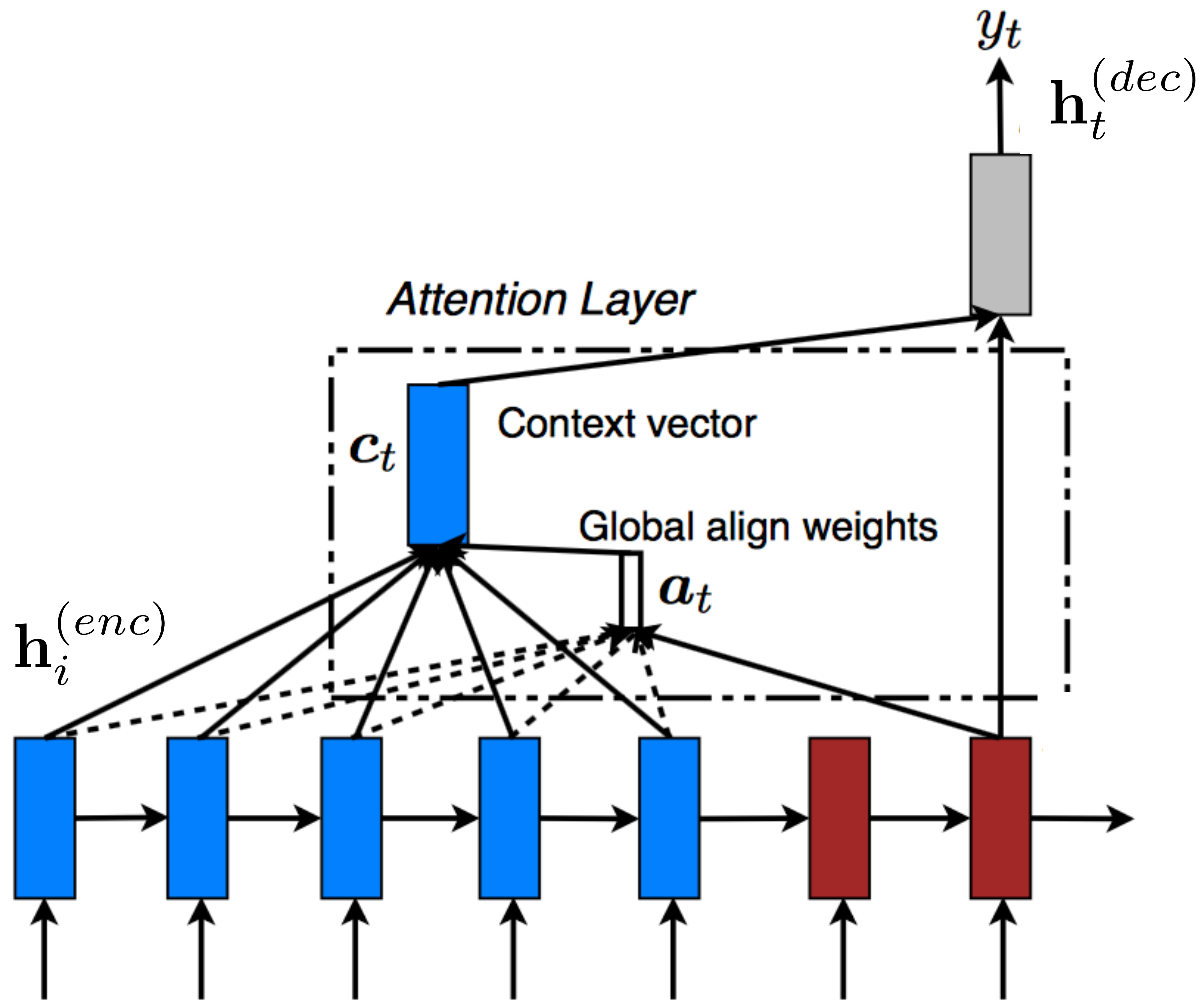
Query: $\mathbf{Q}_t$

$$\beta_{i,t} = score(\mathbf{h}_i^{(enc)}, \mathbf{x}_t^{(dec)})$$

$$\beta_{i,t} = score(\mathbf{h}_i^{(enc)}, \mathbf{h}_{t-1}^{(dec)})$$

$$\beta_{i,t} = score(\mathbf{h}_i^{(enc)}, [\mathbf{x}_t^{(dec)}, \mathbf{h}_{t-1}^{(dec)}])$$

Key: $\mathbf{K}_i$



$$\alpha_{i,t} = \text{Softmax}(\beta_{i,t})$$

Normalize the weights to sum to 1

$$\mathbf{c}_t = \sum_i \alpha_{i,t} \mathbf{h}_i^{(enc)}$$

Value: $\mathbf{V}_i$

Form a context vector that would simply be added to the standard decoder input

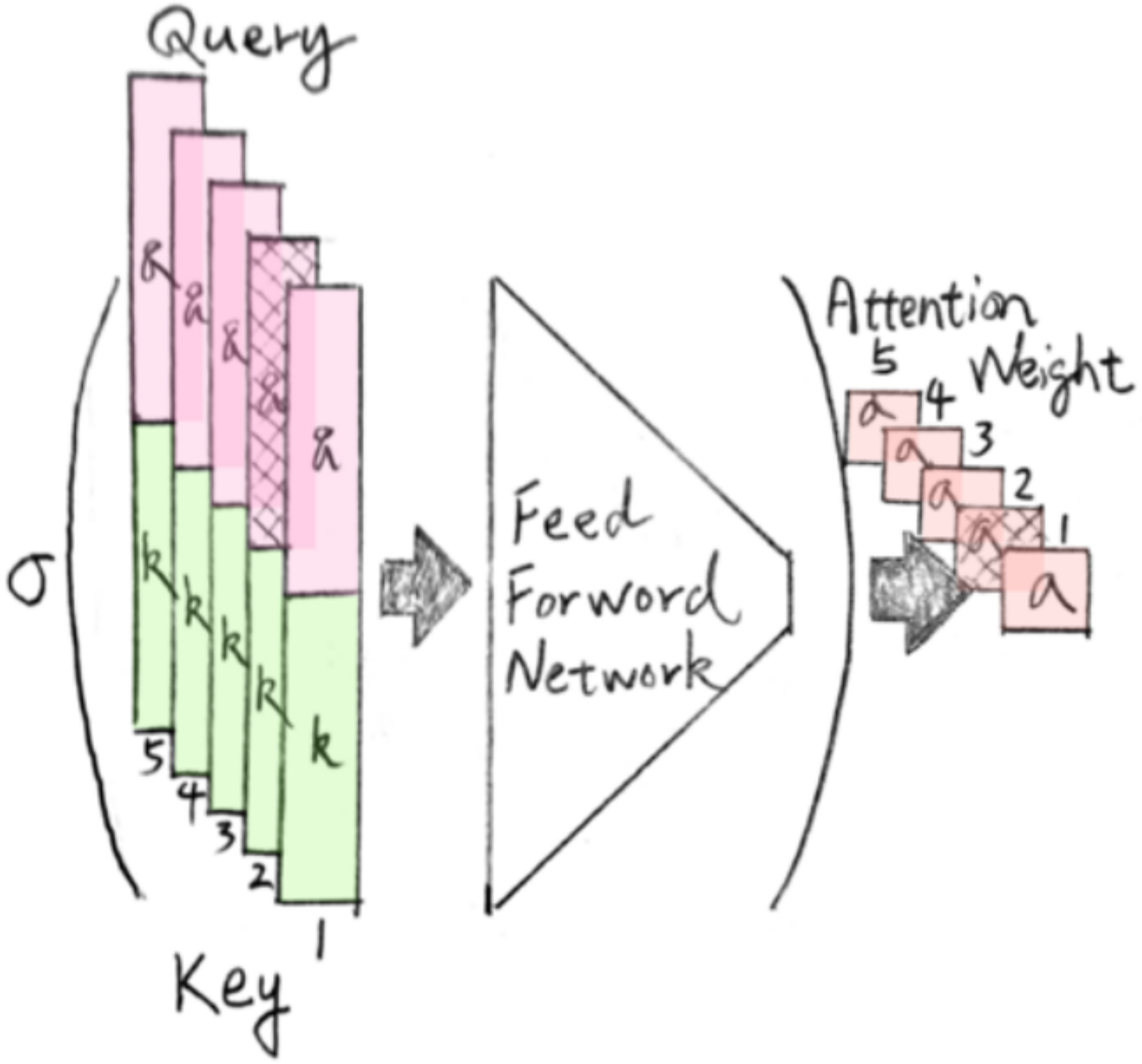
Soft **Attention** in details

Name	Alignment score function	Citation
Content-base attention	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \text{cosine}[\boldsymbol{s}_t, \boldsymbol{h}_i]$	Graves2014
Additive(*)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\boldsymbol{s}_t; \boldsymbol{h}_i])$	Bahdanau2015
Location-Base	$\alpha_{t,i} = \text{softmax}(\mathbf{W}_a \boldsymbol{s}_t)$ Note: This simplifies the softmax alignment to only depend on the target position.	Luong2015
General	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \mathbf{W}_a \boldsymbol{h}_i$ where $\mathbf{W}_a$ is a trainable weight matrix in the attention layer.	Luong2015
Dot-Product	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \boldsymbol{h}_i$	Luong2015
Scaled Dot-Product(^)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \frac{\boldsymbol{s}_t^\top \boldsymbol{h}_i}{\sqrt{n}}$ Note: very similar to the dot-product attention except for a scaling factor; where n is the dimension of the source hidden state.	Vaswani2017

Soft **Attention** in details

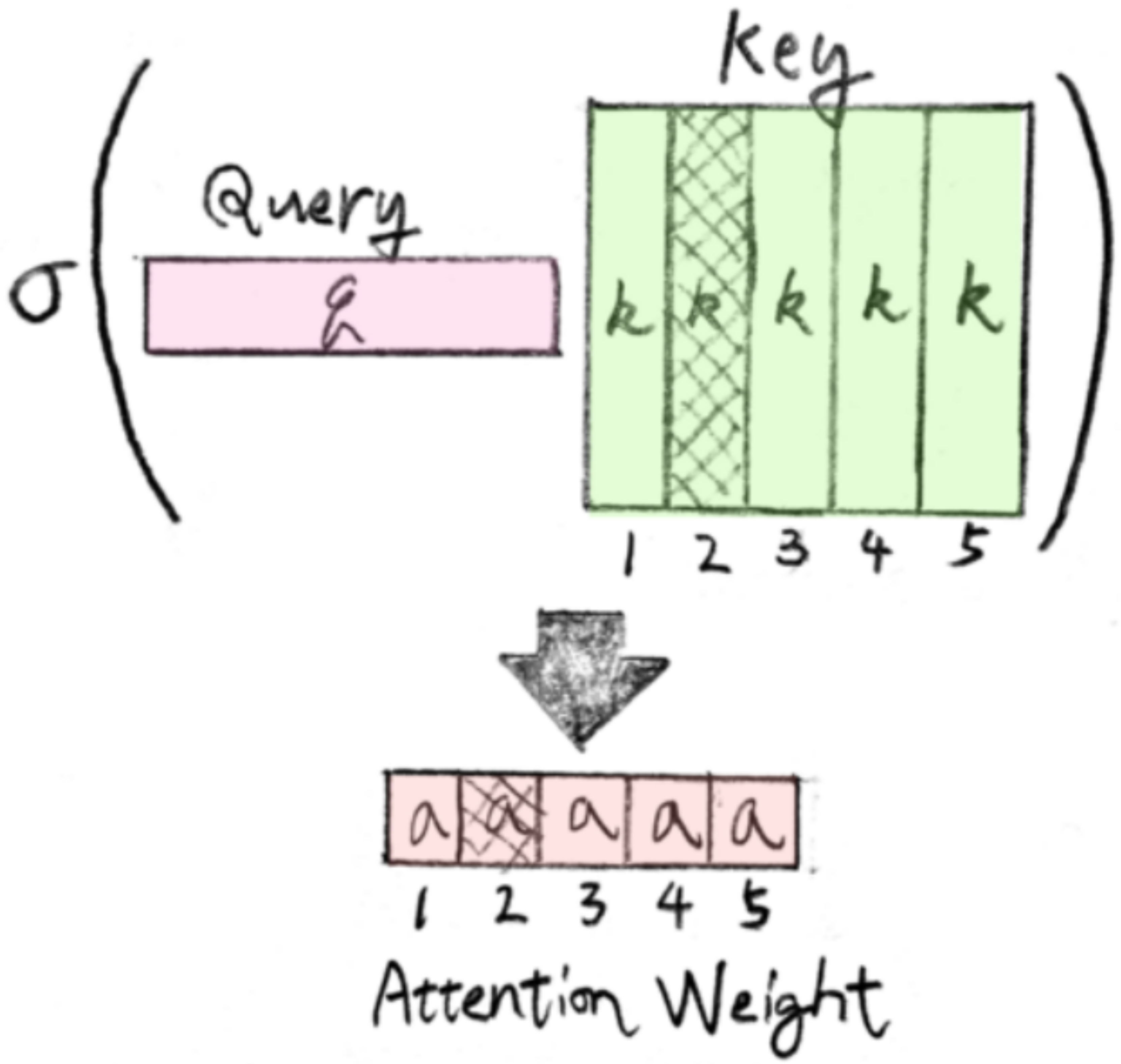
(Additive Attention)

$$softmax(FFN([Q;K]))$$



(Dot-Product Attention)

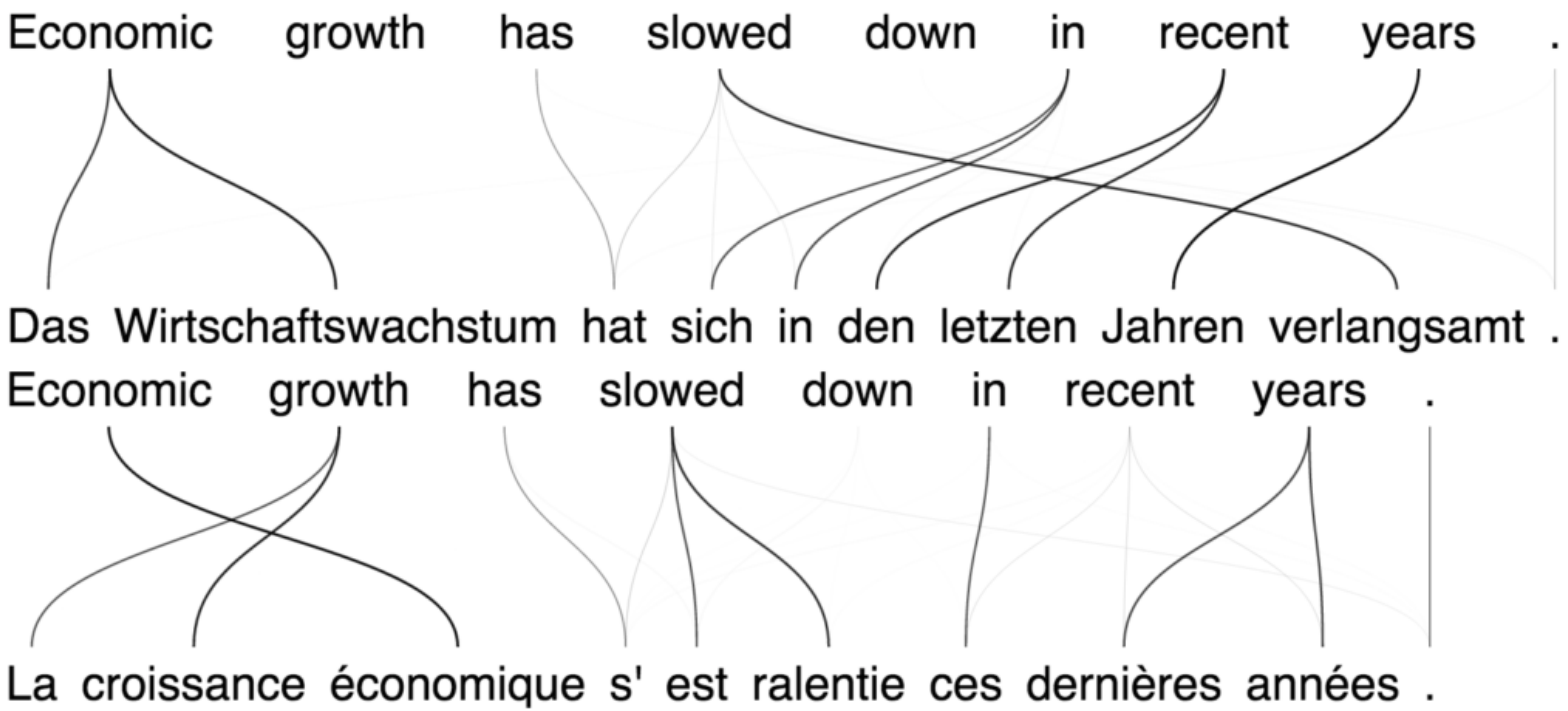
$$softmax(QK^T)$$



https://tzuruey.medium.com/attention-is-all-you-need-98d26aeb3517

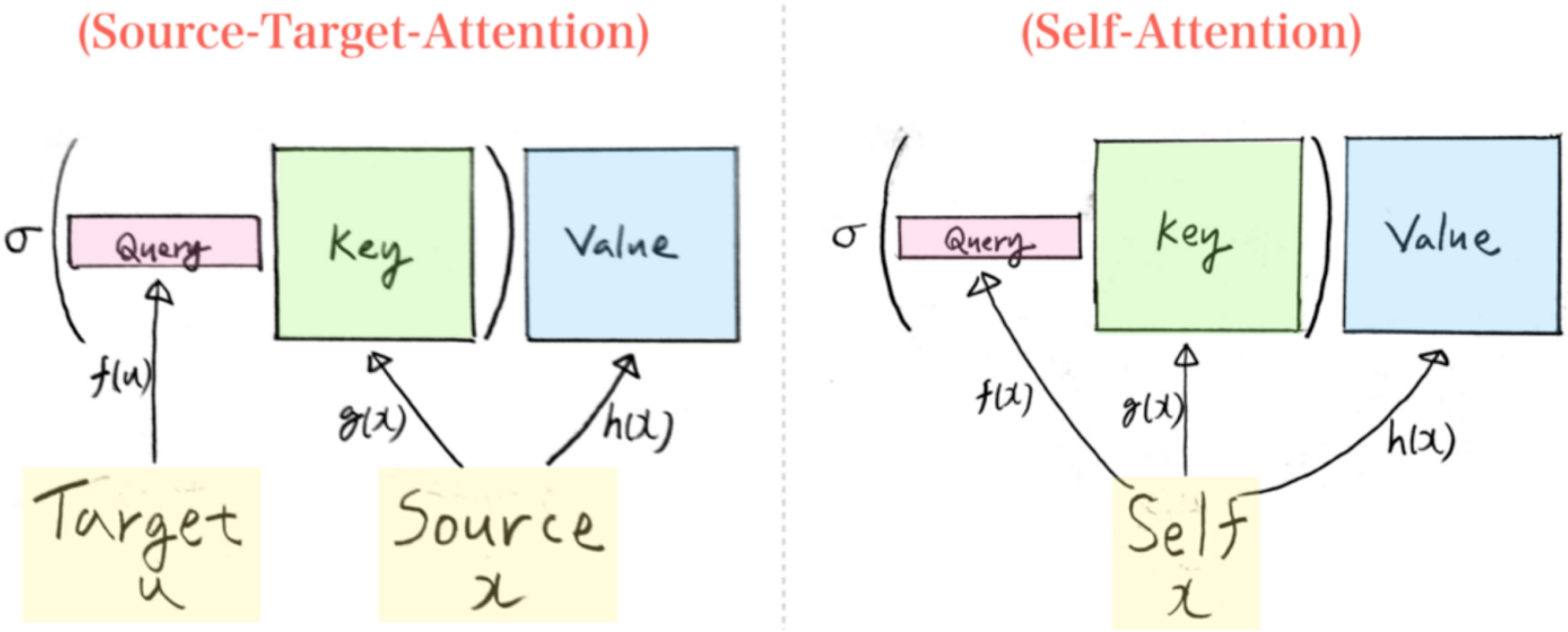
Attention Mechanisms and RNNs

[Cho et al., 2015]



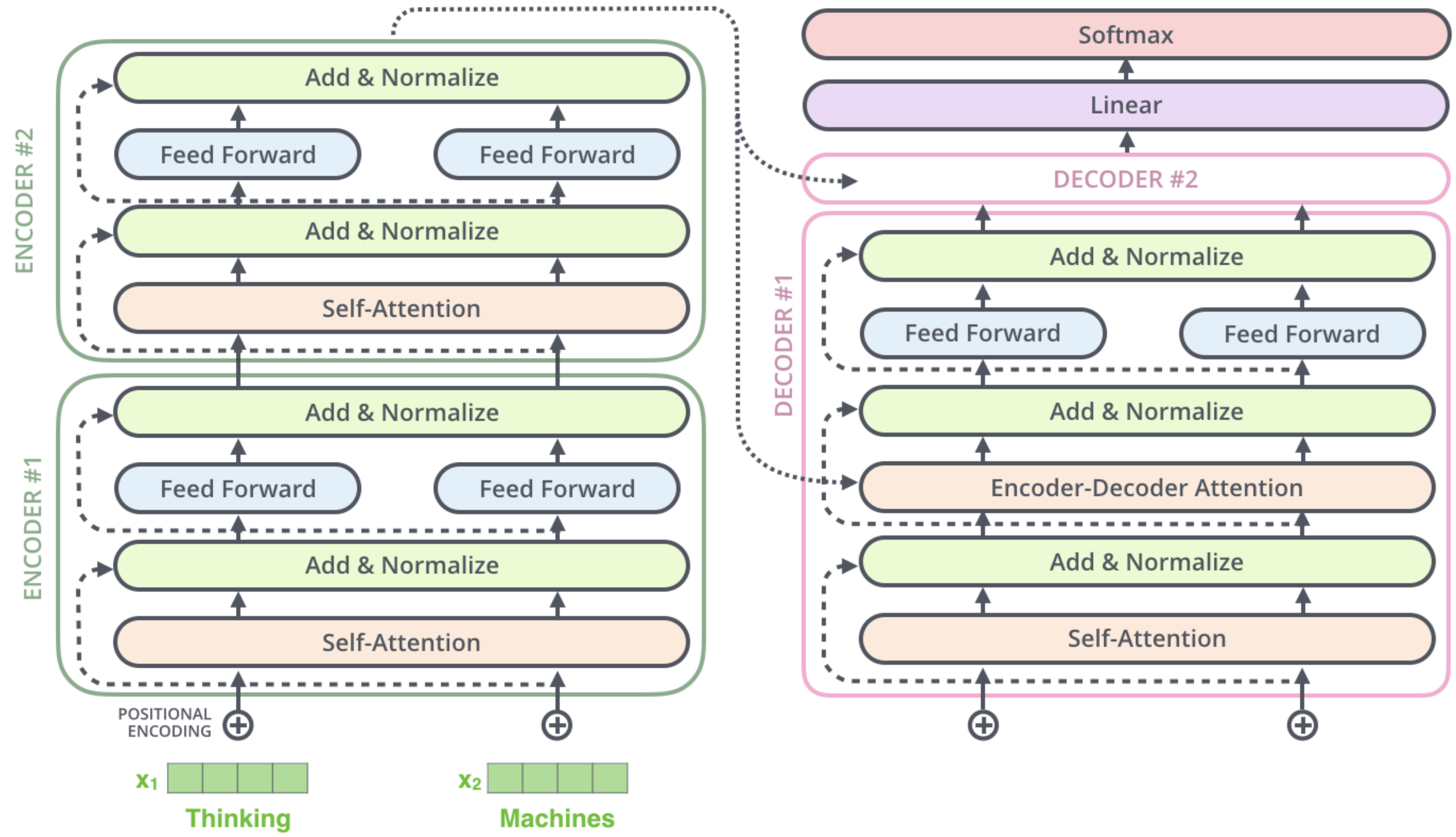
https://devblogs.nvidia.com/introduction-neural-machine-translation-gpus-part-3/

Self Attention

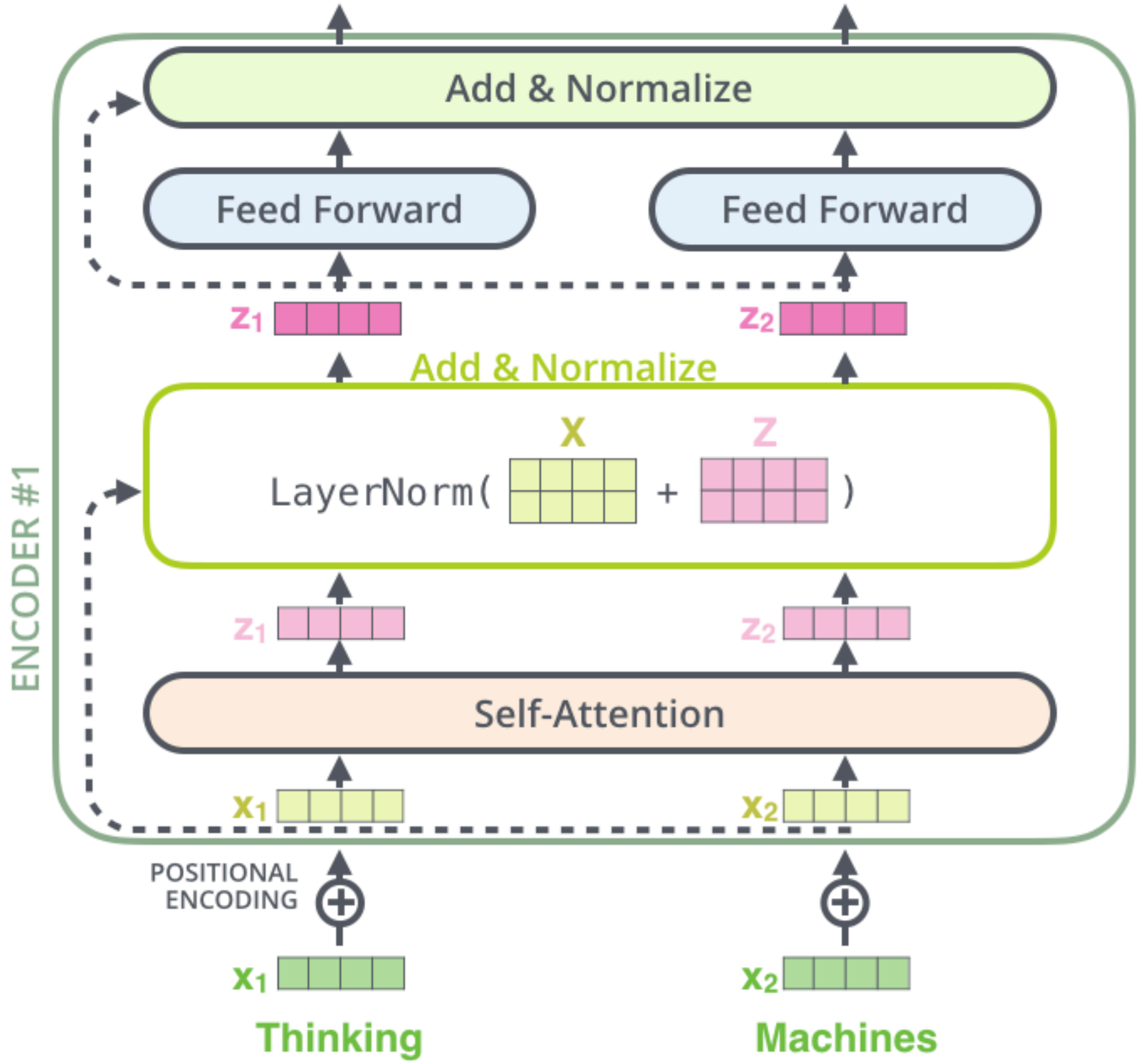


https://tzuruey.medium.com/attention-is-all-you-need-98d26aeb3517

Transformers: Attention is all you need

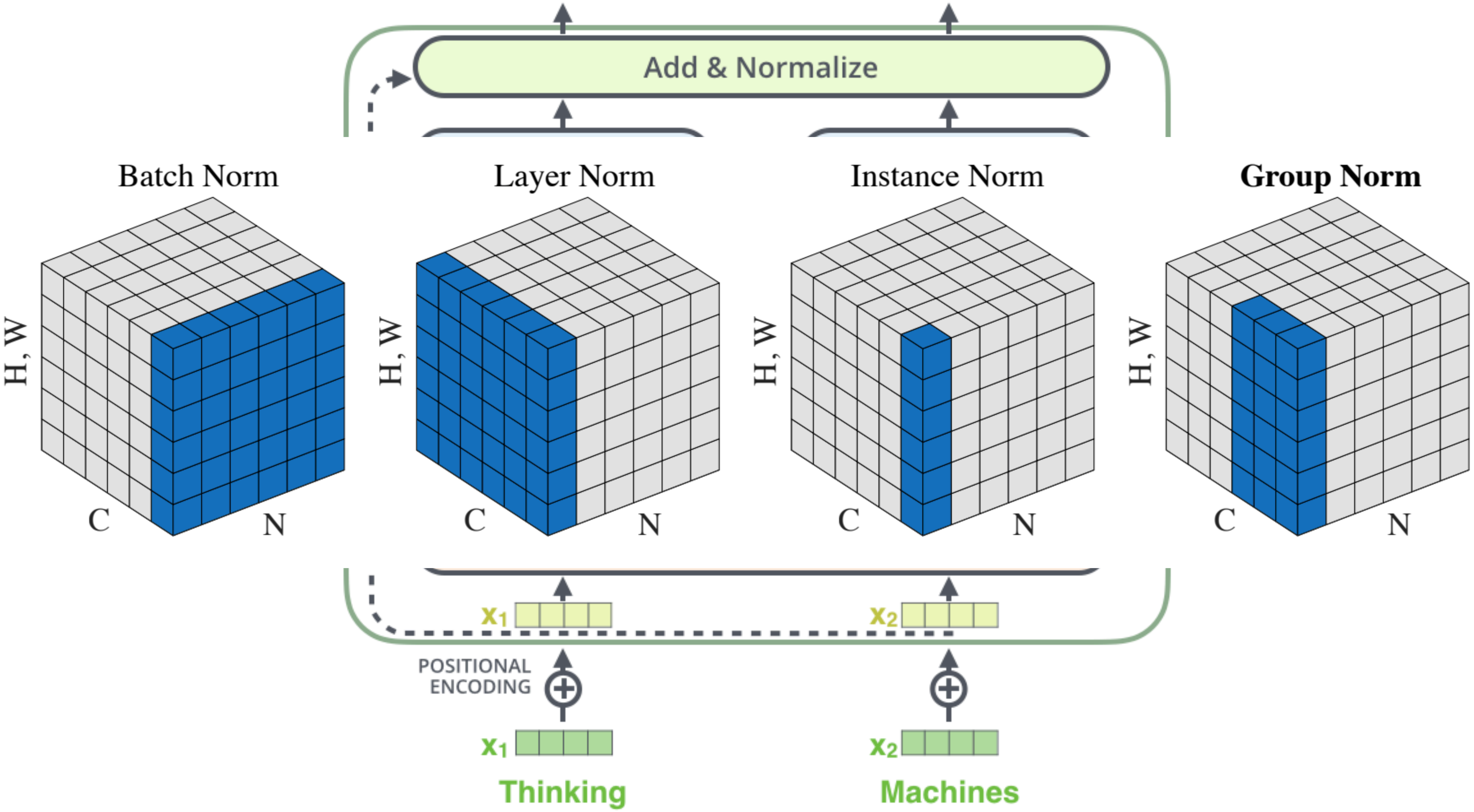


Transformers: Attention is all you need (Encoder)



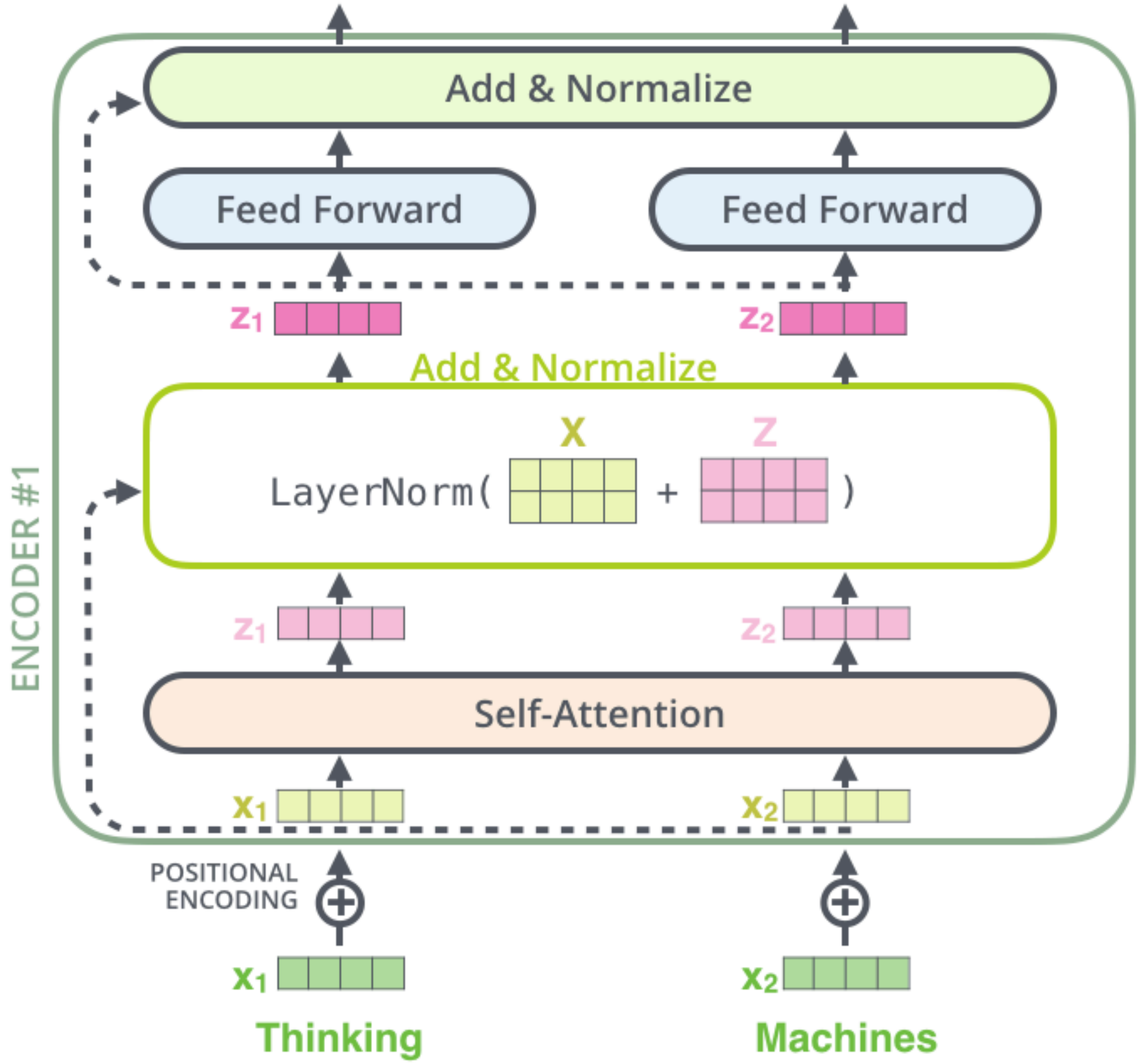
Note: for assignment you are <u>not</u> implementing transformer encoder

Transformers: Attention is all you need (Encoder)



Note: for assignment you are <u>not</u> implementing transformer encoder

Transformers: Attention is all you need (Encoder)

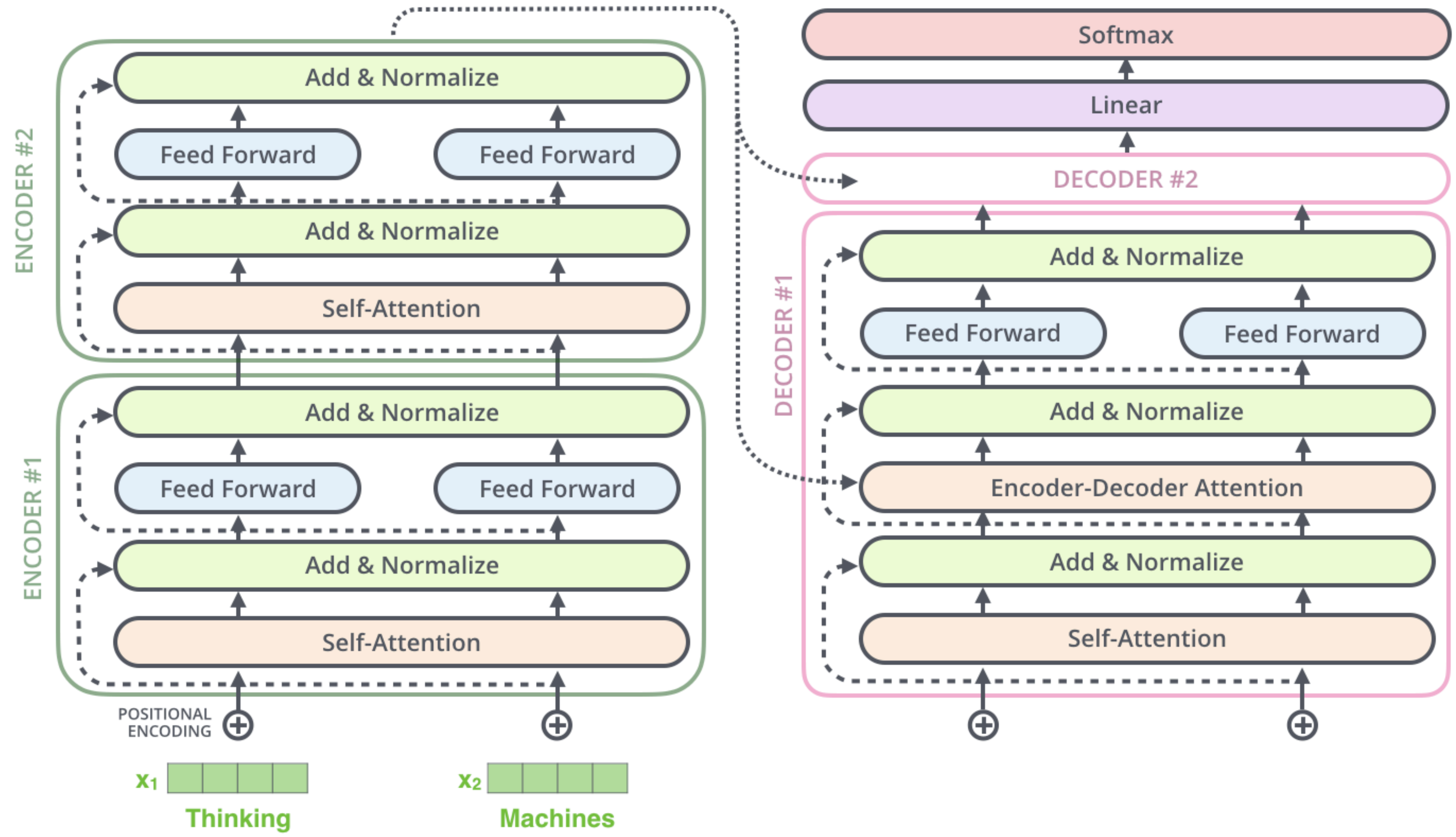


$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}})$$

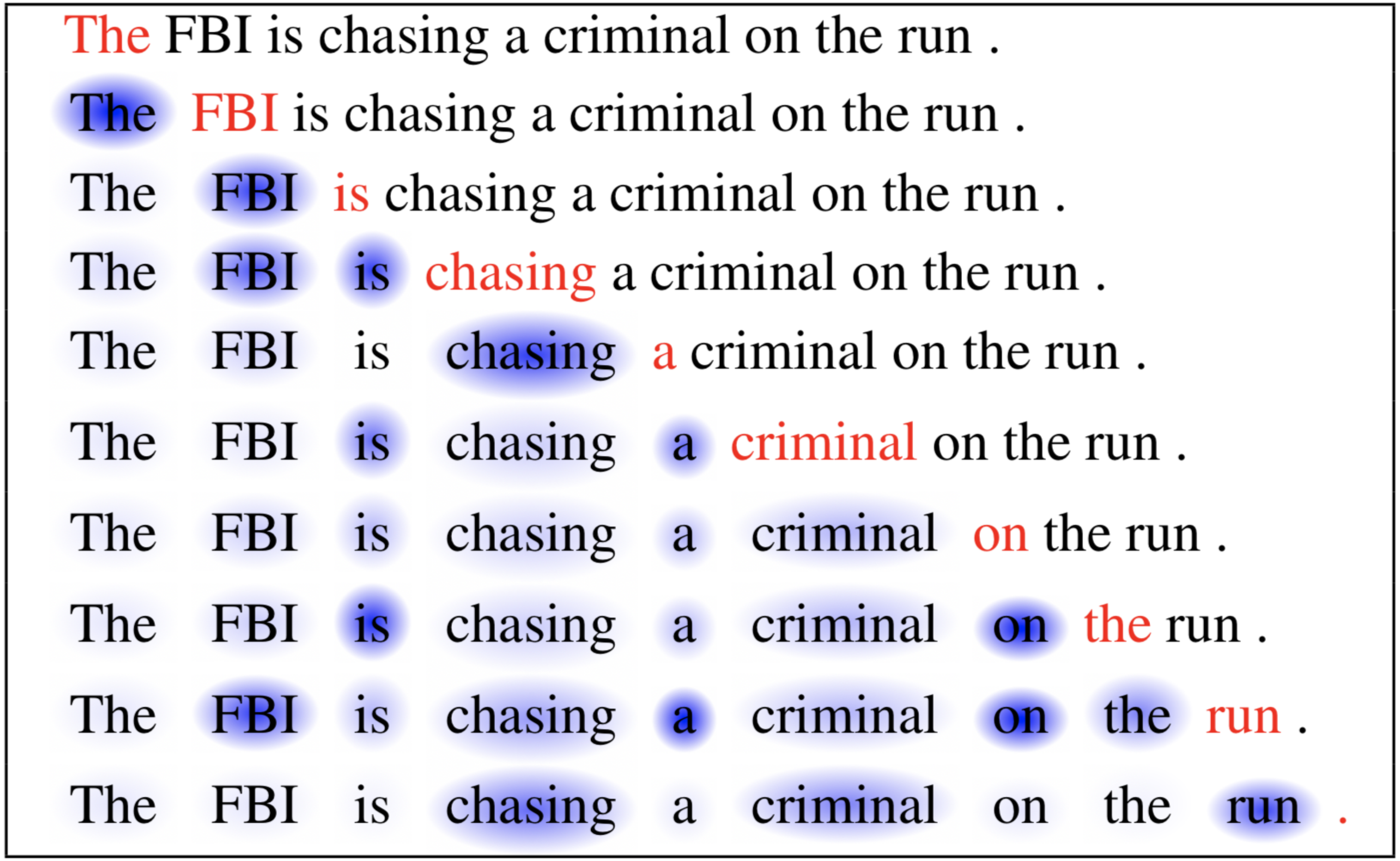
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}})$$

Note: for assignment you are not implementing transformer encoder

Transformers: Attention is all you need



Self Attention



Regularization in RNNs

Standard dropout in recurrent layers does not work because it causes **loss of long term memory**!

* slide from Marco Pedersoli and Thomas Lucas

Regularization in RNNs

Standard dropout in recurrent layers does not work because it causes **loss of long term memory**!

— Dropout in input-to-hidden or hidden-to-output layers [Zaremba et al., 2014]

— Apply dropout at sequence level (same zeroed units for the entire sequence) [Gal, 2016]

— Dropout only at the cell update (for LSTM and GRU units) [Semeniuta et al., 2016]

— Enforcing norm of the hidden state to be similar along time [Krueger & Memisevic, 2016]

— Zoneout some hidden units (copy their state to the next tilmestep) [Krueger et al., 2016]

* slide from Marco Pedersoli and Thomas Lucas