

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

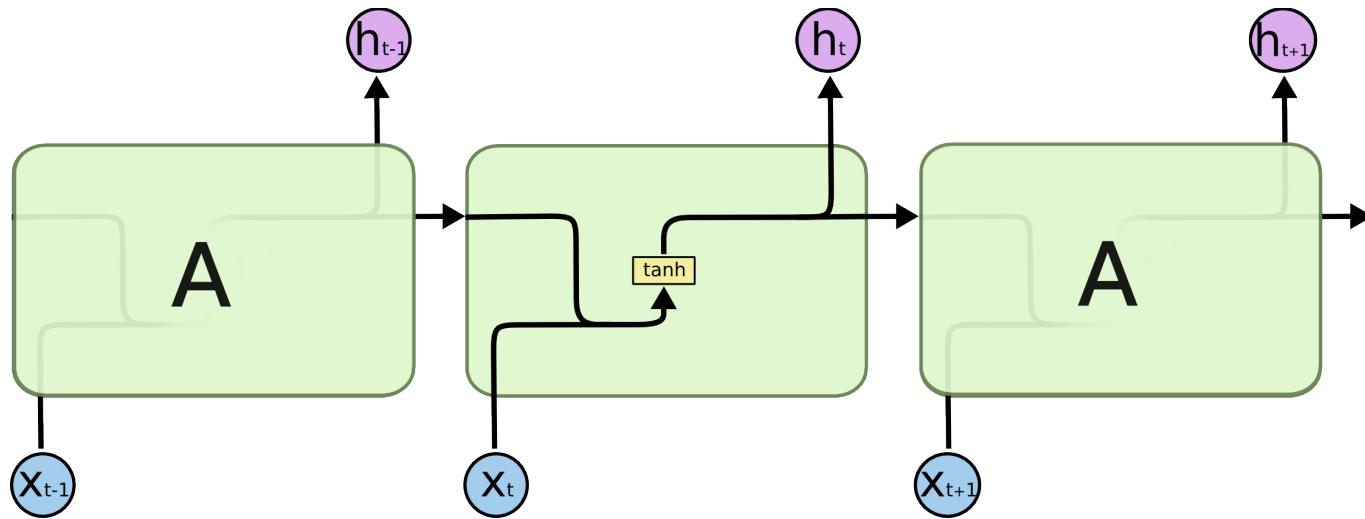
**Shusen Wang**

# LSTM Model

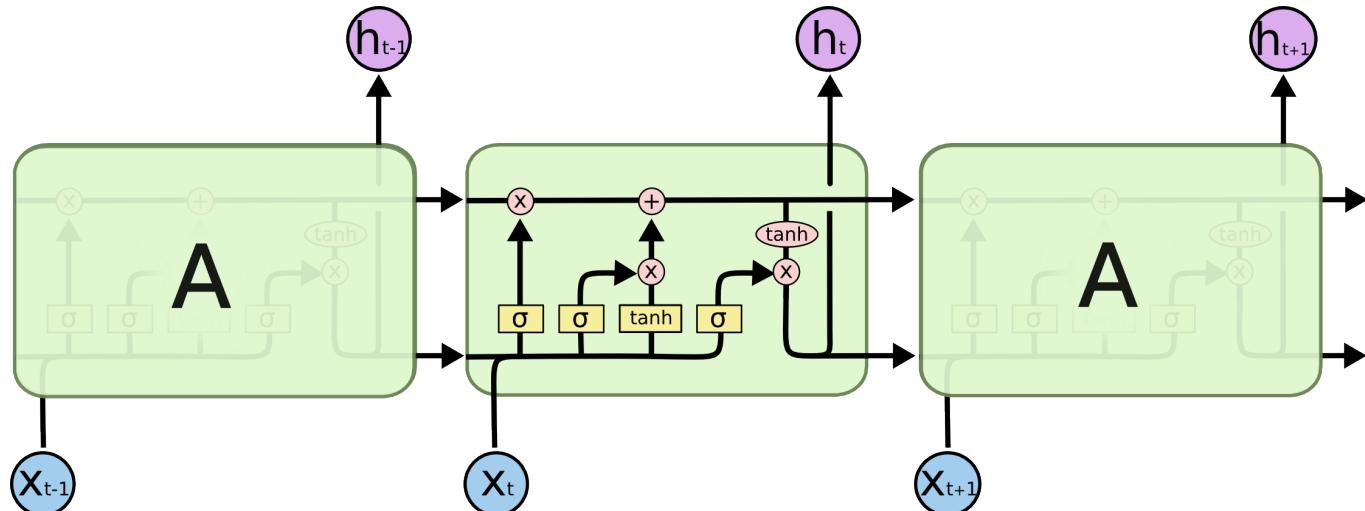
## Reference

- Hochreiter and Schmidhuber. [Long short-term memory](#). *Neural computation*, 1997.

# LSTM Networks



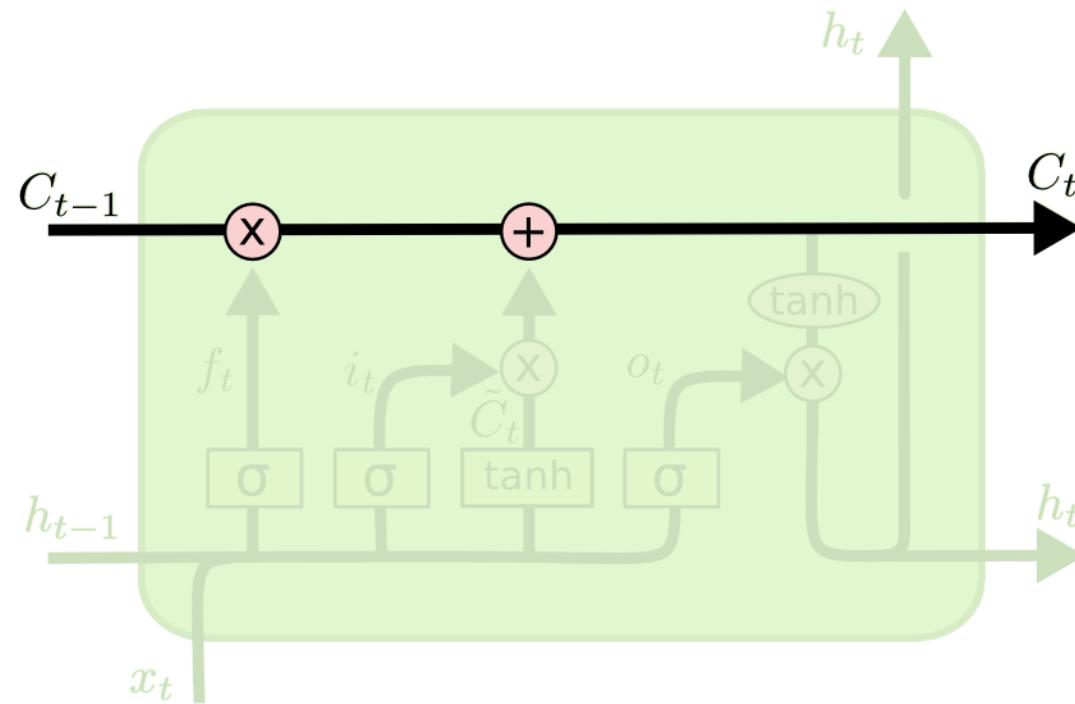
Simple RNN



LSTM

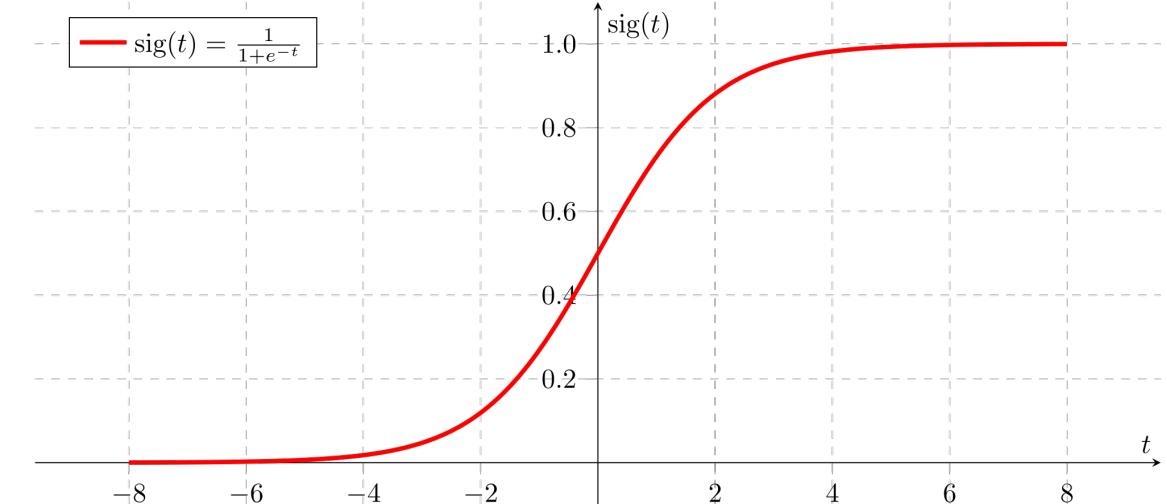
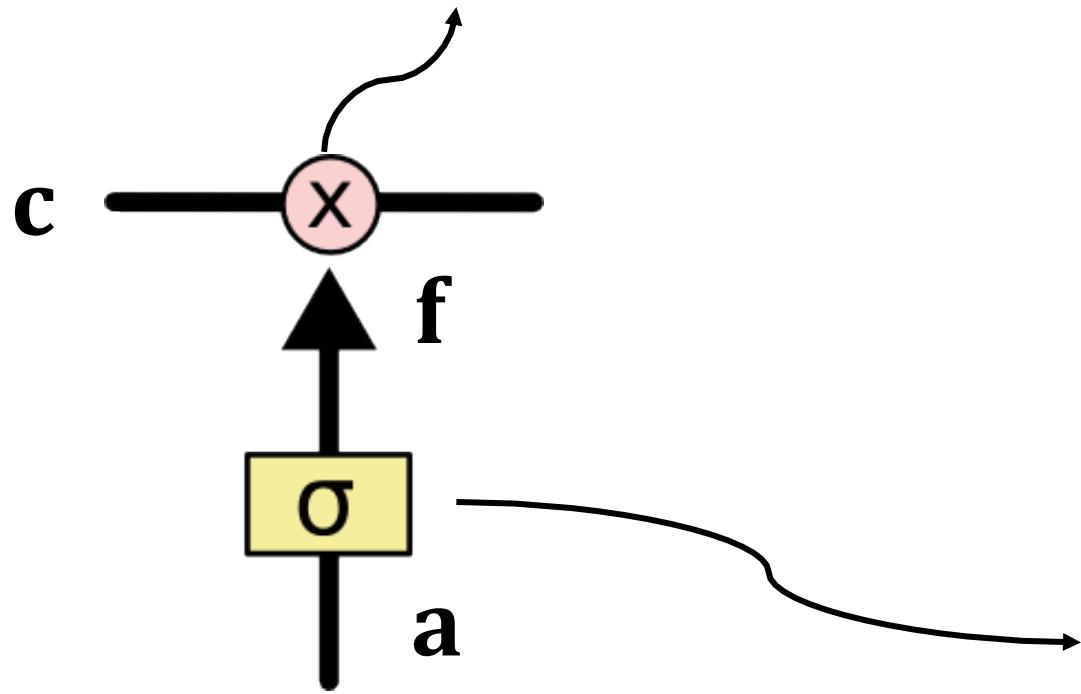
# LSTM: Conveyor Belt

- Conveyor belt: the past information directly flows to the future.



# LSTM: Forget Gate

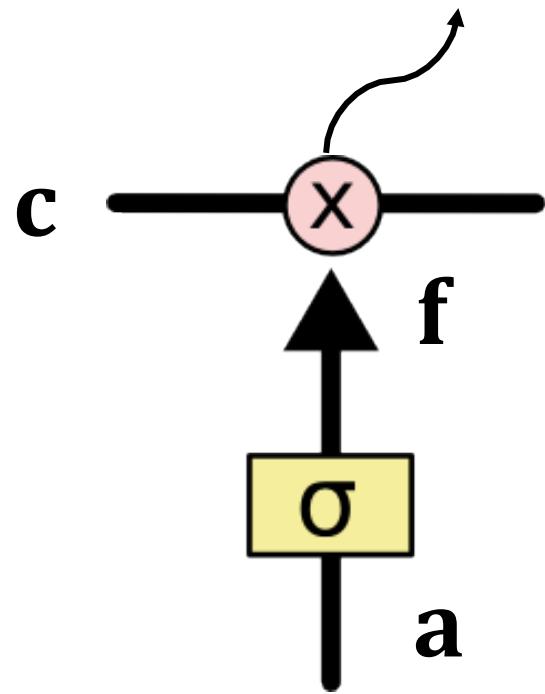
Elementwise multiplication of 2 vectors.



Sigmoid function: between 0 and 1.

# LSTM: Forget Gate

Elementwise multiplication of 2 vectors.

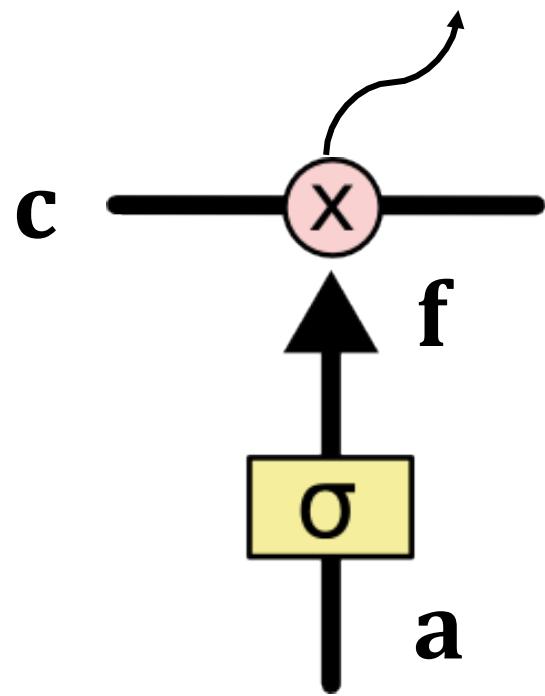


$$\sigma \begin{pmatrix} 1 \\ 3 \\ 0 \\ -2 \end{pmatrix} = \begin{bmatrix} 0.73 \\ 0.95 \\ 0.5 \\ 0.12 \end{bmatrix}$$

The diagram shows the sigmoid function  $\sigma$  applied to a vector  $a$  (indicated by a curly brace below the first two entries) to produce a vector  $f$  (indicated by a curly brace below the last two entries).

# LSTM: Forget Gate

Elementwise multiplication of 2 vectors.



$$\begin{bmatrix} 0.9 \\ 0.2 \\ -0.5 \\ -0.1 \end{bmatrix} \circ \begin{bmatrix} 0.5 \\ 0 \\ 1 \\ 0.8 \end{bmatrix} = \begin{bmatrix} 0.45 \\ 0 \\ -0.5 \\ -0.08 \end{bmatrix}$$

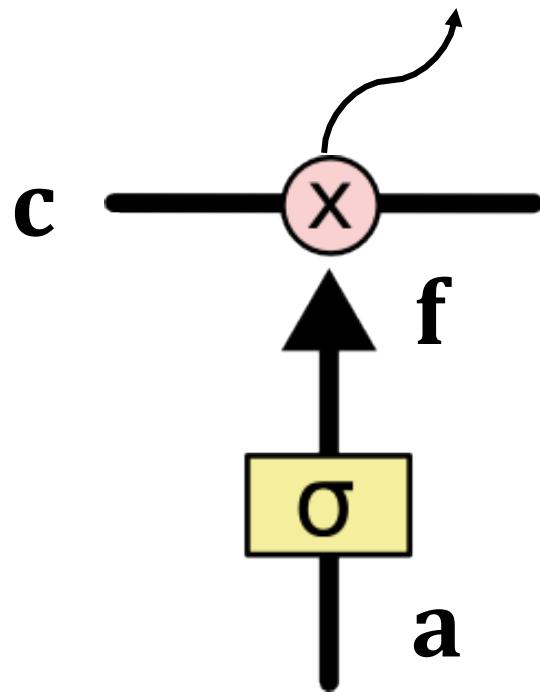
Diagram illustrating the elementwise multiplication of two vectors:

- Input vector  $c$ :  $\begin{bmatrix} 0.9 \\ 0.2 \\ -0.5 \\ -0.1 \end{bmatrix}$
- Input vector  $f$ :  $\begin{bmatrix} 0.5 \\ 0 \\ 1 \\ 0.8 \end{bmatrix}$
- Output vector:  $\begin{bmatrix} 0.45 \\ 0 \\ -0.5 \\ -0.08 \end{bmatrix}$ , labeled as "output"

# LSTM: Forget Gate

- Forget gate ( $f$ ): a vector (the same shape as  $c$  and  $h$ ).
  - A value of **zero** means “let **nothing** through”.
  - A value of **one** means “let **everything** through!”

Elementwise multiplication of 2 vectors.



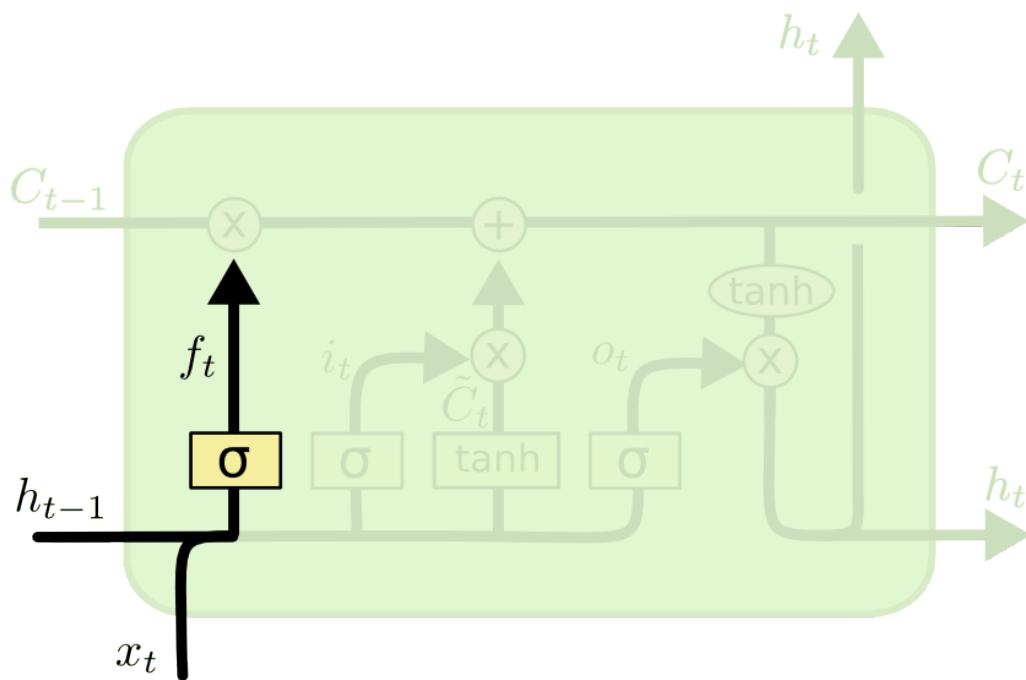
$$\begin{bmatrix} 0.9 \\ 0.2 \\ -0.5 \\ -0.1 \end{bmatrix} \circ \begin{bmatrix} 0.5 \\ 0 \\ 1 \\ 0.8 \end{bmatrix} = \begin{bmatrix} 0.45 \\ 0 \\ -0.5 \\ -0.08 \end{bmatrix}$$

Diagram illustrating the elementwise multiplication of two vectors:

- c**: Input vector
- f**: Forget gate vector
- output**: Result of elementwise multiplication

# LSTM: Forget Gate

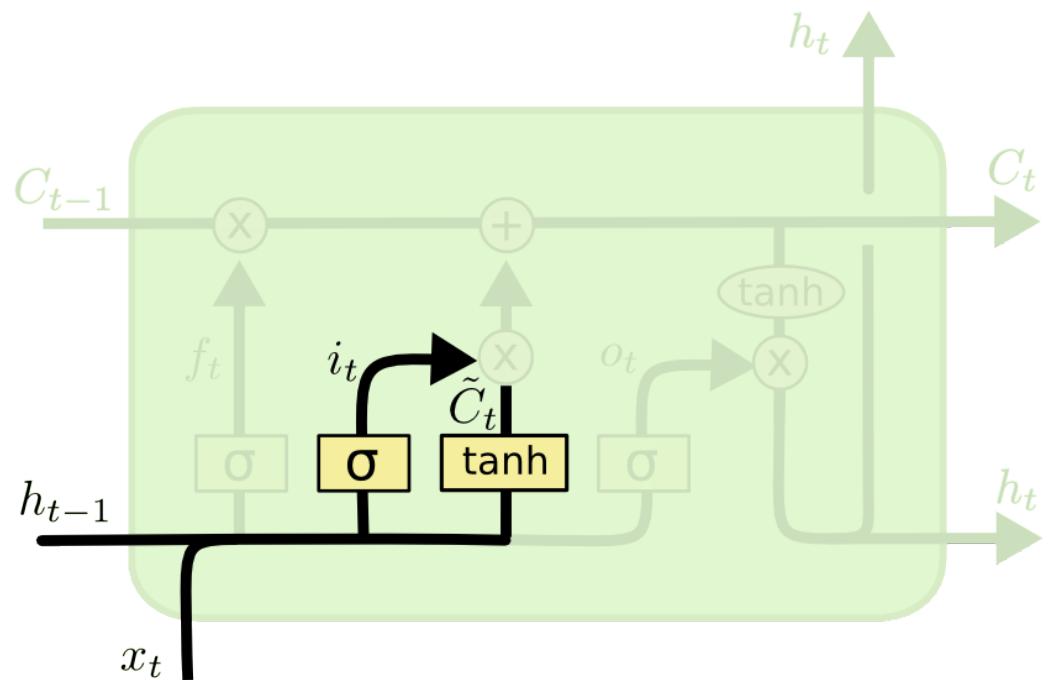
- Forget gate ( $f$ ): a vector (the same shape as  $\mathbf{c}$  and  $\mathbf{h}$ ).
  - A value of zero means “let nothing through”.
  - A value of one means “let everything through!”



$$f_t = \sigma \begin{bmatrix} \text{purple row} & \text{blue row} \end{bmatrix} \cdot \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix}$$

# LSTM: Input Gate

- Input gate ( $i_t$ ): decides which values of the conveyor belt we'll update.

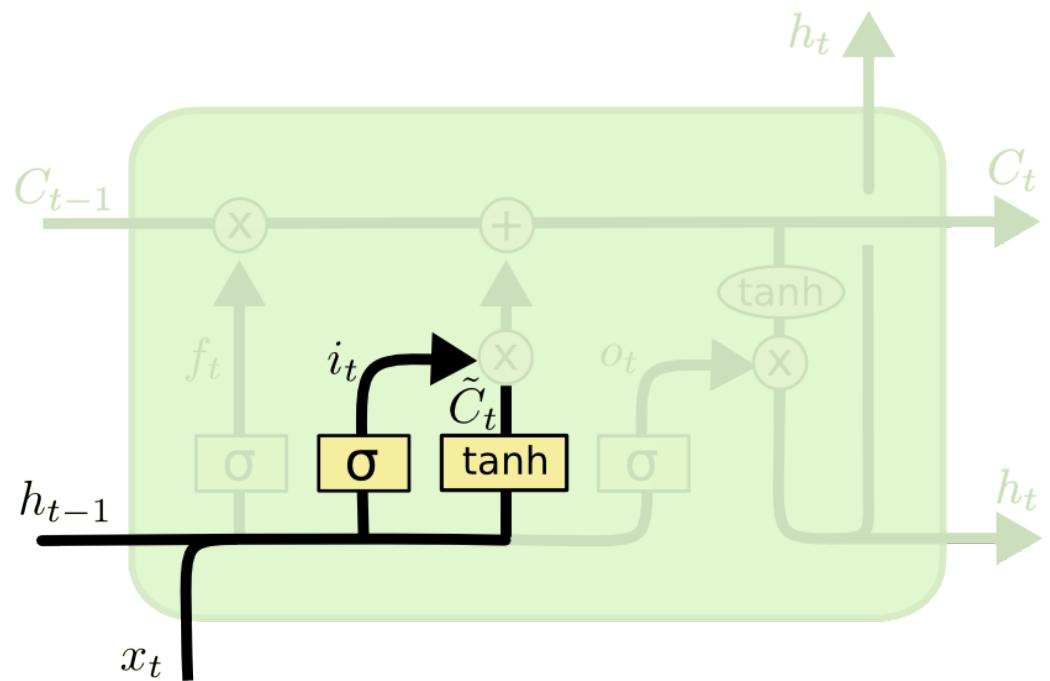


The diagram illustrates the computation of the hidden state  $h_{t-1}$  and input  $x_t$  from the previous hidden state  $h_{t-1}$  and input  $i_t$ . The input  $i_t$  is passed through a weight matrix  $W_i$  (represented by a grid of purple and blue squares) and a bias vector (represented by a vertical column of purple squares). The result is then passed through a sigmoid function ( $\sigma$ ) to produce the hidden state  $h_{t-1}$ . A dot product operation is shown between the hidden state  $h_{t-1}$  and the input  $i_t$ , resulting in the input  $x_t$ .

The left figure is from Christopher Olah's blog: Understanding LSTM Networks.

# LSTM: New Value

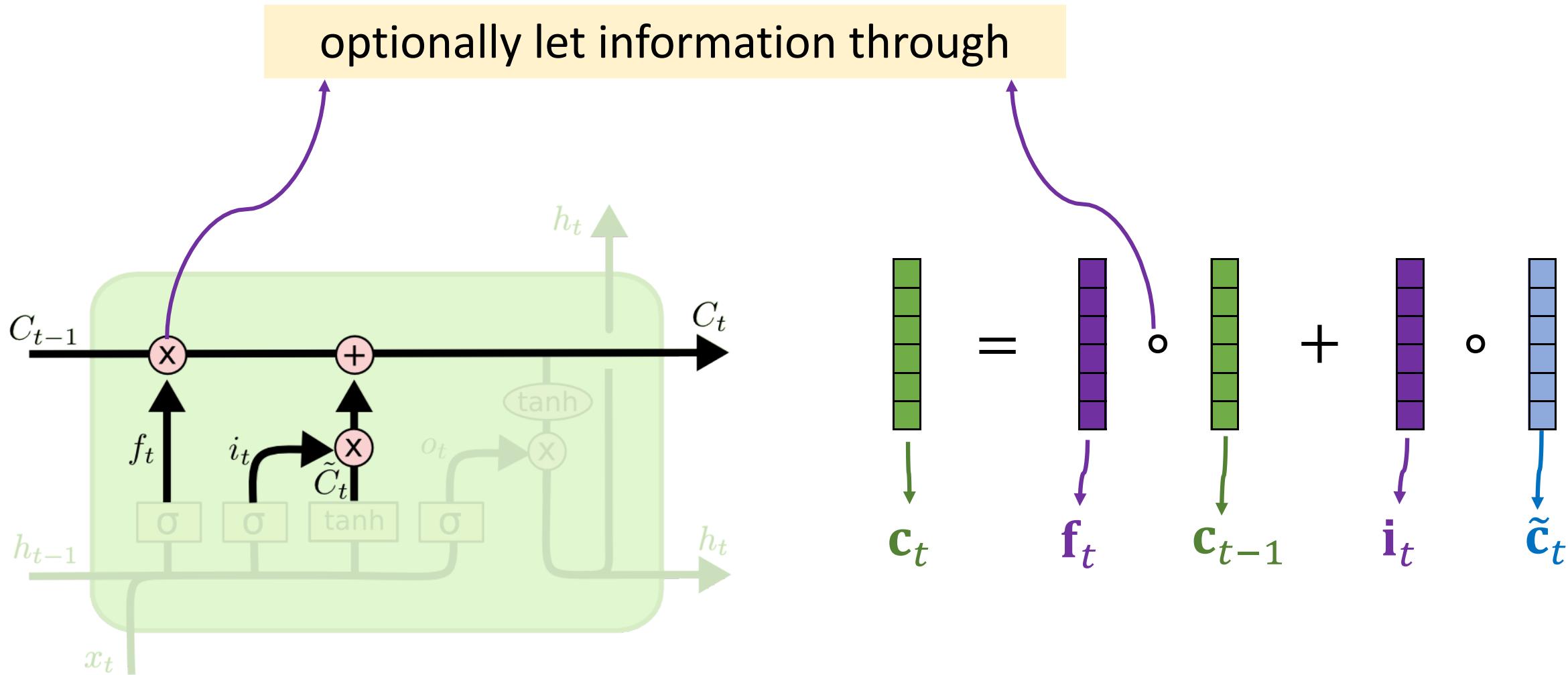
- New value ( $\tilde{c}_t$ ): to be added to the conveyor belt.



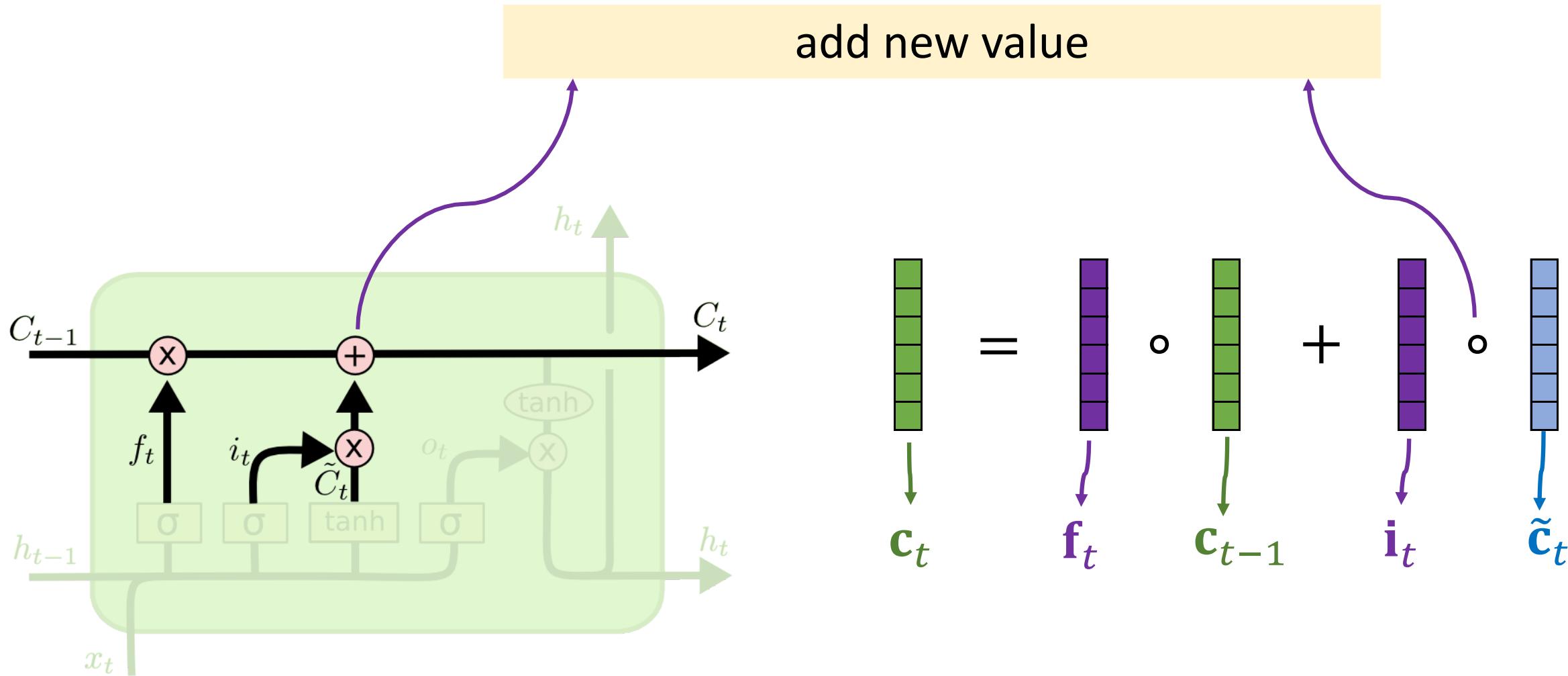
$$\tilde{c}_t = \tanh \left[ \begin{array}{c|c} \text{purple} & \text{blue} \\ \hline \text{purple} & \text{blue} \end{array} \right] \cdot W_c$$

The diagram shows a matrix multiplication operation where a vertical vector of purple and blue blocks is multiplied by a weight matrix  $W_c$ . The result is a single purple block, labeled  $\tilde{c}_t$ . This represents the candidate hidden state  $\tilde{C}_t$  being passed through a tanh function. To the right, another vertical vector of purple and blue blocks is shown, with arrows pointing to  $h_{t-1}$  and  $x_t$ , indicating the inputs to the LSTM cell.

# LSTM: Update the Conveyor Belt

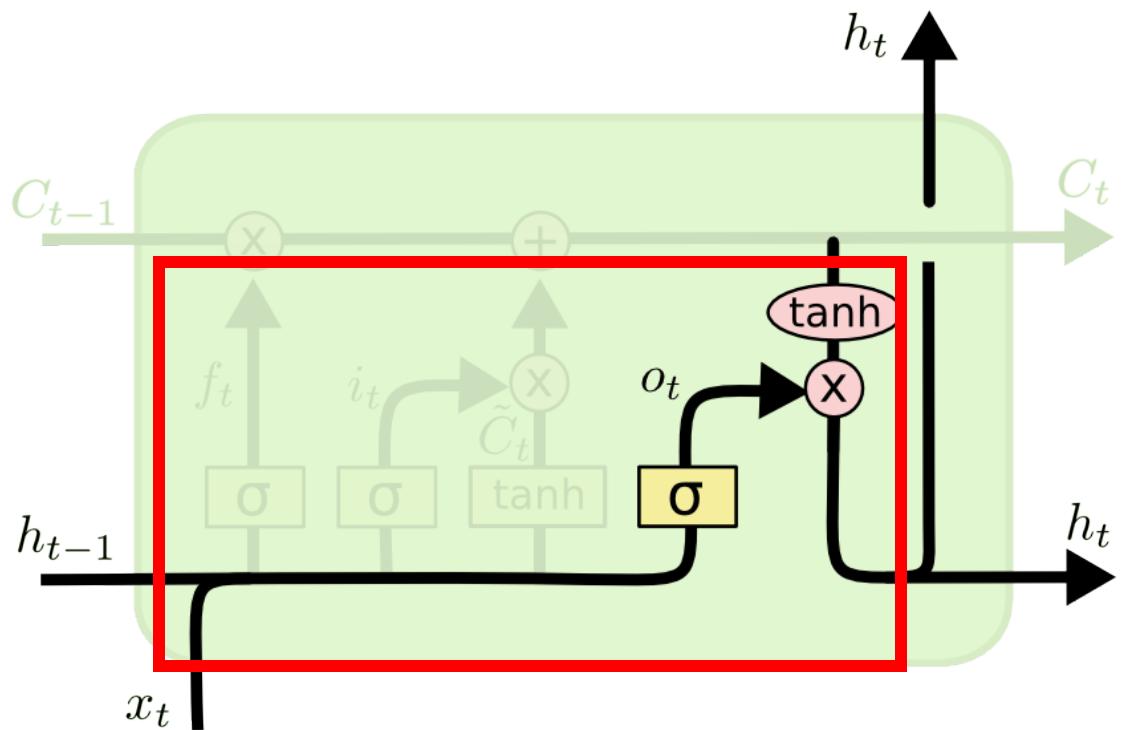


# LSTM: Update the Conveyor Belt



# LSTM: Output Gate

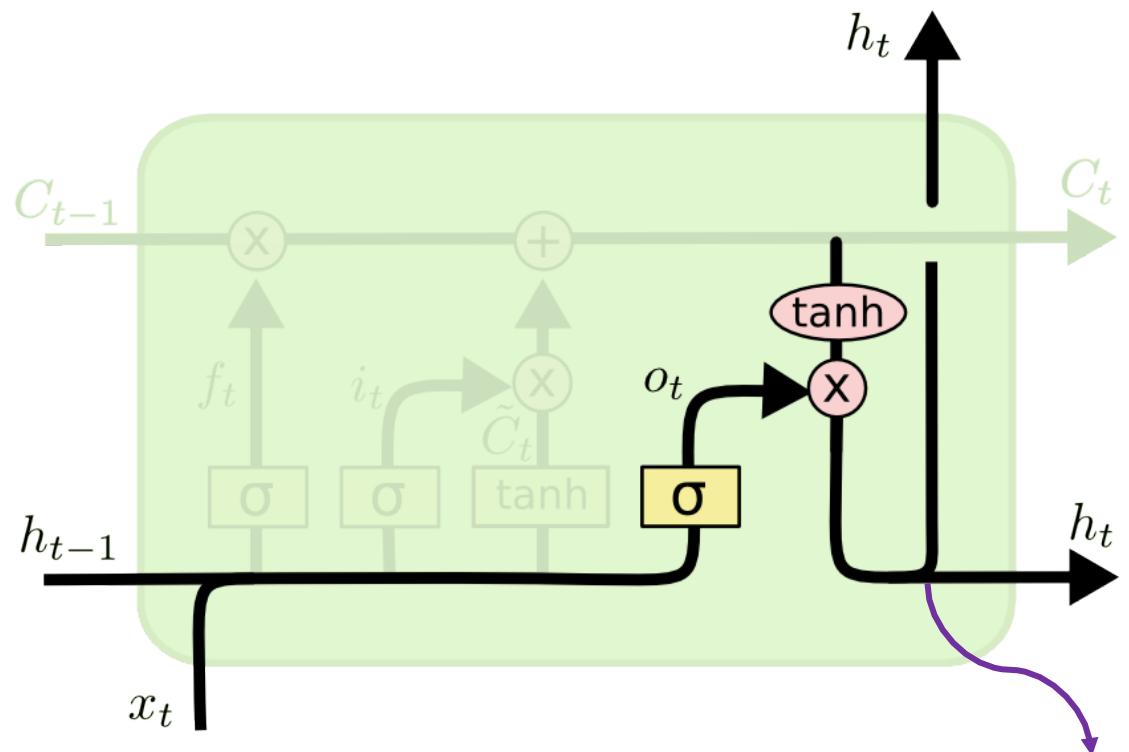
- Output gate ( $\mathbf{o}_t$ ): decide what flows from the conveyor belt  $C_{t-1}$  to the state  $h_t$ .



$$\mathbf{o}_t = \sigma \begin{bmatrix} \text{red vertical vector} \\ \vdots \\ \text{red vertical vector} \end{bmatrix} = \sigma \begin{bmatrix} \text{purple grid} & \text{blue grid} \\ \vdots & \downarrow \\ \text{purple grid} & \text{blue grid} \end{bmatrix} \cdot \begin{bmatrix} \text{purple vertical vector} \\ \vdots \\ \text{blue vertical vector} \end{bmatrix} = \mathbf{h}_{t-1} \cdot \mathbf{x}_t$$

# LSTM: Update State

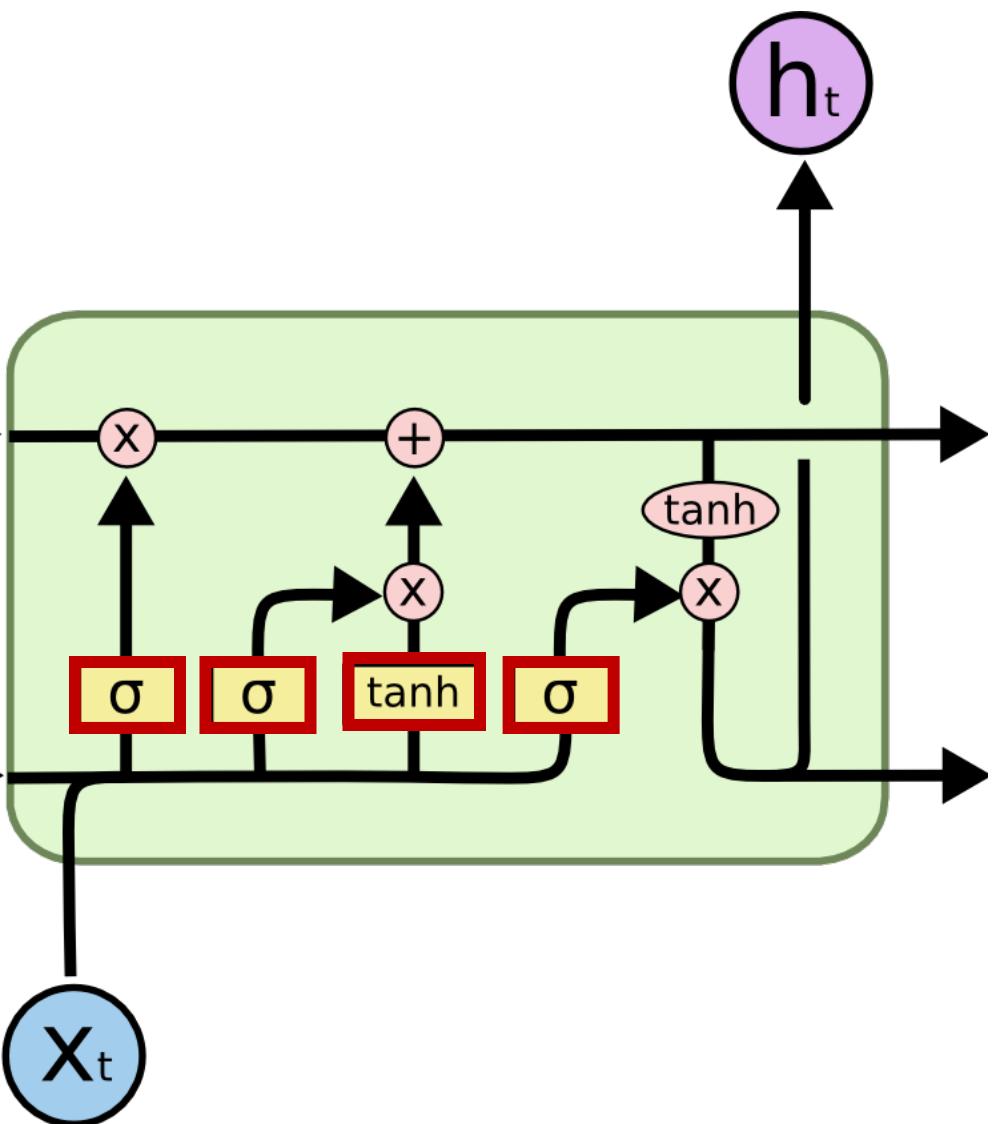
- State ( $\mathbf{h}_t$ ): the output of LSTM.



Two copies of  $h_t$

$$\begin{matrix} \textcolor{purple}{h}_t \\ = \\ \textcolor{red}{o}_t \\ \circ \quad \tanh \quad \left[ \begin{array}{c} \textcolor{green}{C}_t \\ \vdots \\ \textcolor{green}{C}_t \end{array} \right] \end{matrix}$$

# LSTM: Number of Parameters

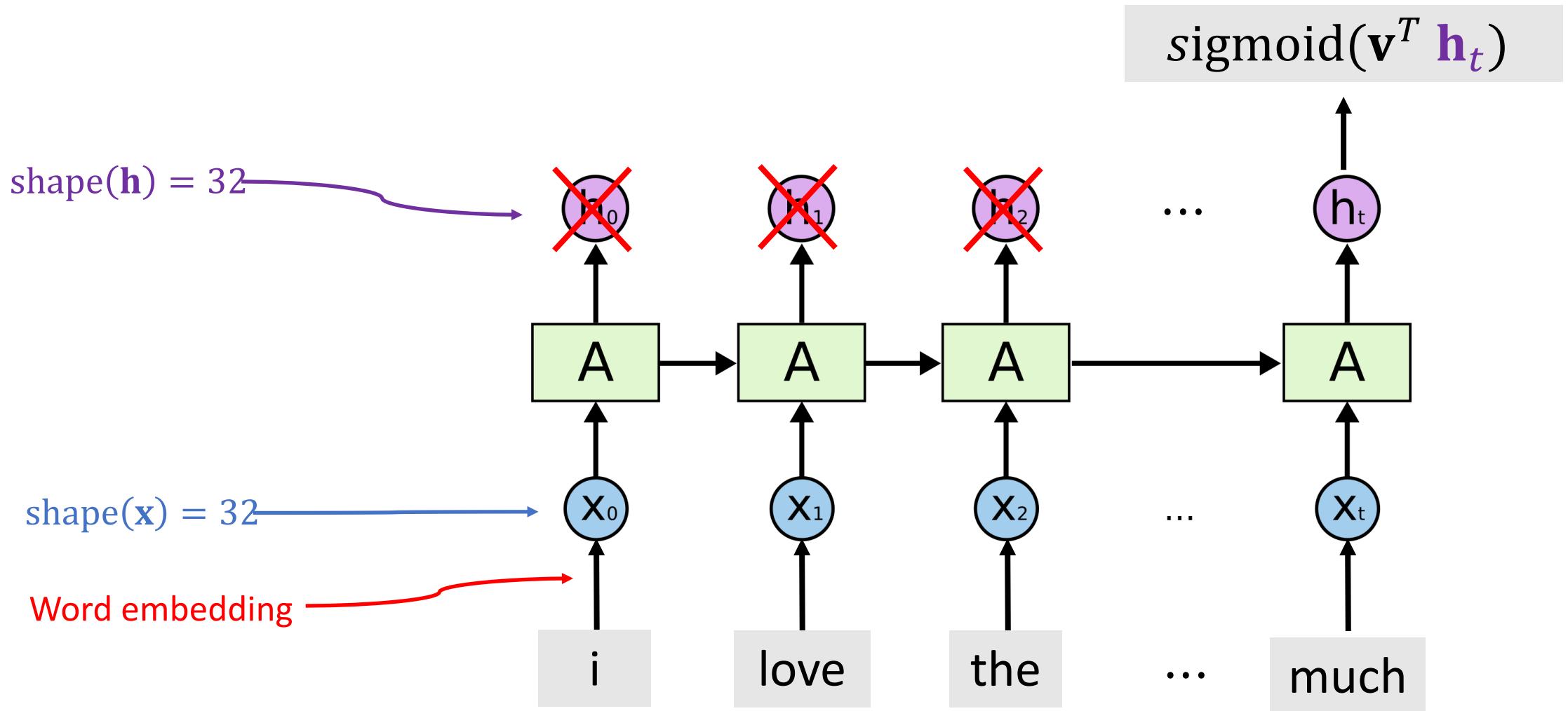


#parameters: **4 times as many as SimpleRNN**

- **4** parameter matrices, each of which has
  - #rows: shape ( $h$ )
  - #cols: shape ( $h$ ) + shape ( $x$ )
- #parameter (do not count intercept):  
$$4 \times \text{shape}(\mathbf{h}) \times [\text{shape}(\mathbf{h}) + \text{shape}(\mathbf{x})]$$

# LSTM Using Keras

# LSTM for IMDB Review



# LSTM for IMDB Review

```
from keras.models import Sequential
from keras.layers import LSTM, Embedding, Dense, Flatten

vocabulary = 10000
embedding_dim = 32
word_num = 500
state_dim = 32

model = Sequential()
model.add(Embedding(vocabulary, embedding_dim, input_length=word_num))
model.add(LSTM(state_dim, return_sequences=False))
model.add(Dense(1, activation='sigmoid'))

Replace "SimpleRNN" by "LSTM".
model.summary()
```

# LSTM for IMDB Review

Layer (type)	Output Shape	Param #
=====		
embedding_1 (Embedding)	(None, 500, 32)	320000
lstm_1 (LSTM)	(None, 32)	8320
dense_1 (Dense)	(None, 1)	33
=====		
Total params:	328,353	
Trainable params:	328,353	
Non-trainable params:	0	

# LSTM for IMDB Review

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 500, 32)	320000
lstm_1 (LSTM)	(None, 32)	8320
dense_1 (Dense)	(None, 1)	33
Total params: 328,353	#parameters in LSTM:	
Trainable params: 328,353	• $8320 = 2080 \times 4$	
Non-trainable params: 0	• $2080 = 32 \times (32 + 32) + 32$	

shape( $h$ ) = 32

shape( $x$ )

# LSTM for IMDB Review

- Training Accuracy: 91.8%
- Validation Accuracy: 88.7%
- Test Accuracy: 88.6%

Substantial improvement over SimpleRNN (whose test accuracy is 84%).

# LSTM Dropout

```
from keras.models import Sequential
from keras.layers import LSTM, Embedding, Dense, Flatten

vocabulary = 10000
embedding_dim = 32
word_num = 500
state_dim = 32

model = Sequential()
model.add(Embedding(vocabulary, embedding_dim, input_length=word_num))
model.add(LSTM(state_dim, return_sequences=False, dropout=0.2))
model.add(Dense(1, activation='sigmoid'))

model.summary()
```

# Summary

- LSTM uses a “**conveyor belt**” to get longer memory than SimpleRNN.

# Summary

- LSTM uses a “conveyor belt” to get longer memory than SimpleRNN.
- Each of the following blocks has a parameter matrix:
  - Forget gate.
  - Input gate.
  - New values.
  - Output gate.
- Number of parameters:
$$4 \times \text{shape}(\mathbf{h}) \times [\text{shape}(\mathbf{h}) + \text{shape}(\mathbf{x})].$$

**Thank You!**