

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

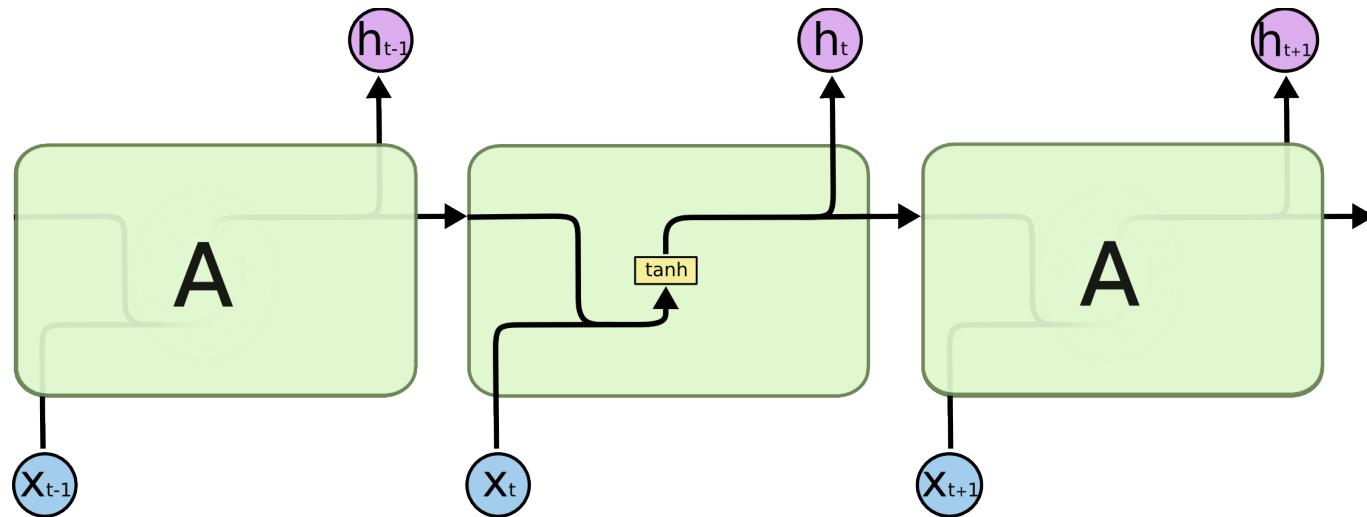
**Shusen Wang**

# LSTM Model

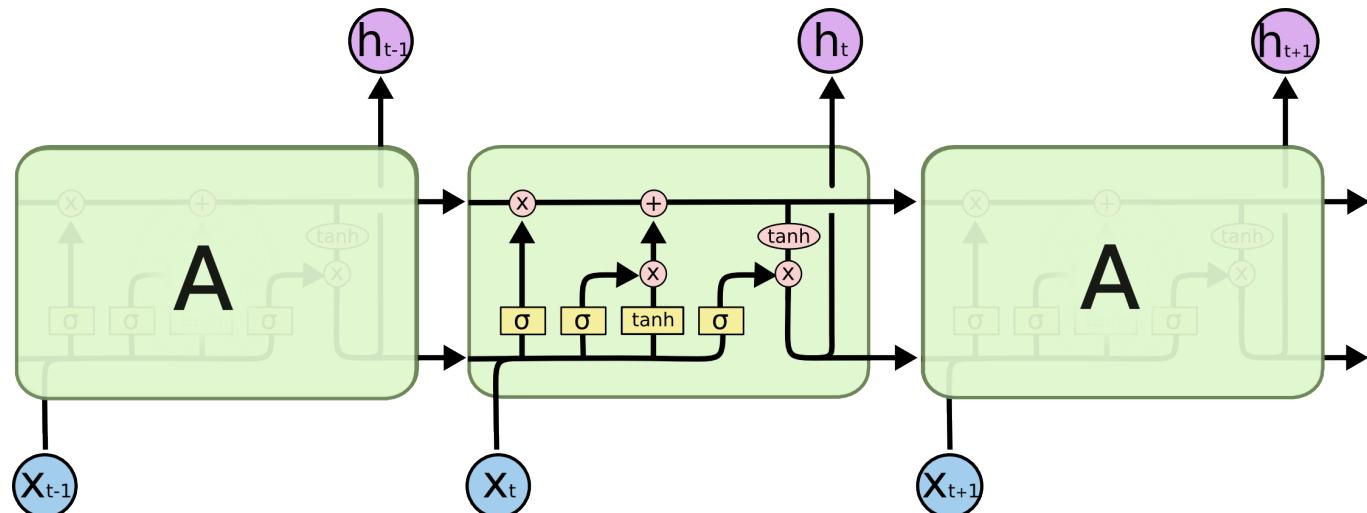
## Reference

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

# LSTM Networks



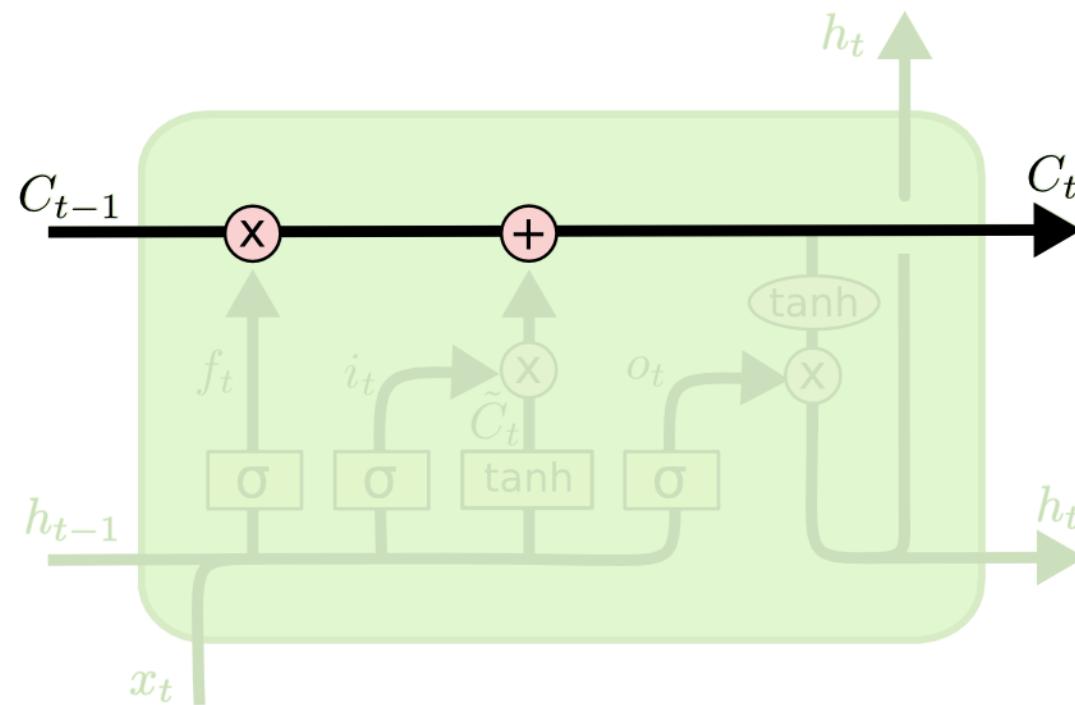
Simple RNN



LSTM

# LSTM: Conveyor Belt

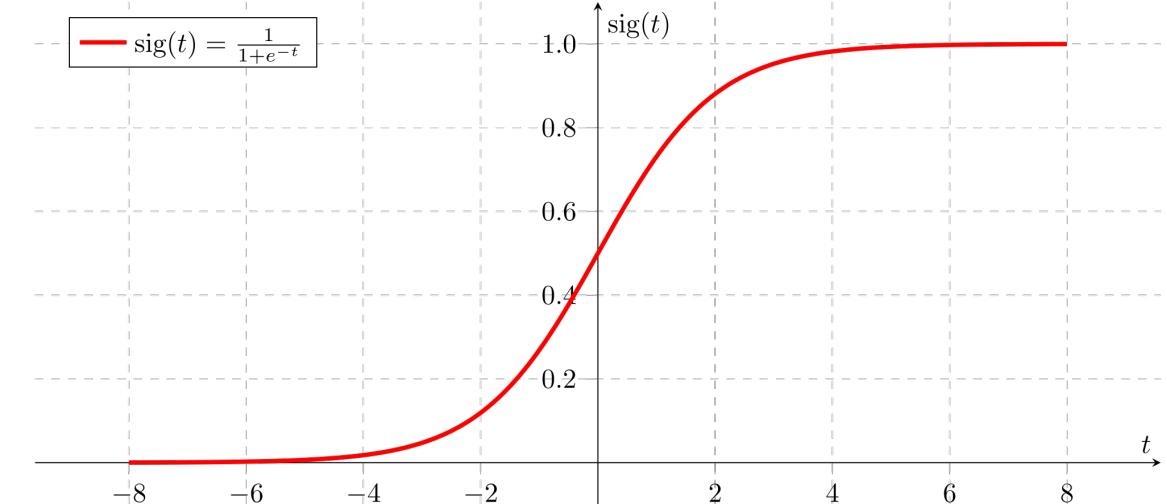
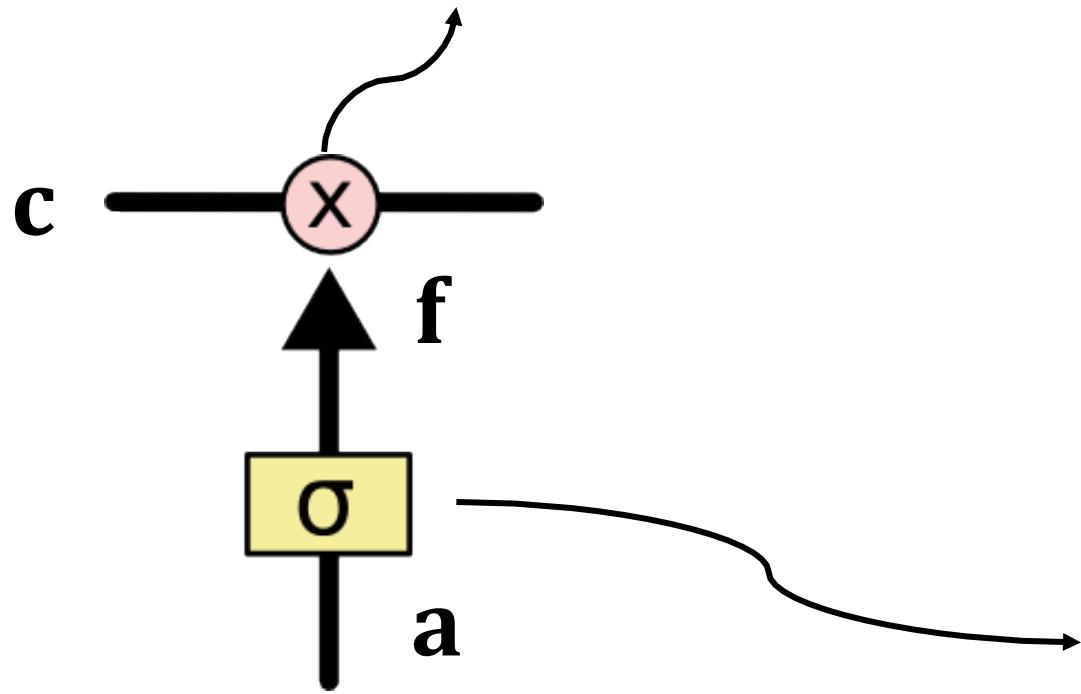
- Conveyor belt: the past information directly flows to the future.



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

# LSTM: Forget Gate

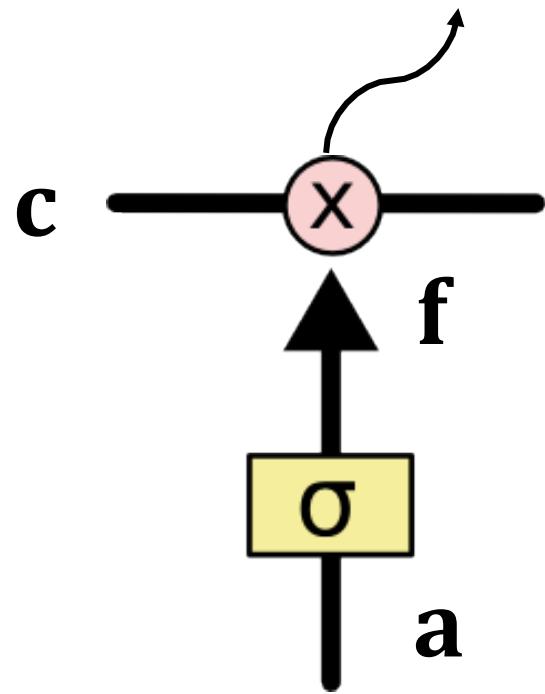
Elementwise multiplication of 2 vectors.



Sigmoid function: between 0 and 1.

# LSTM: Forget Gate

Elementwise multiplication of 2 vectors.

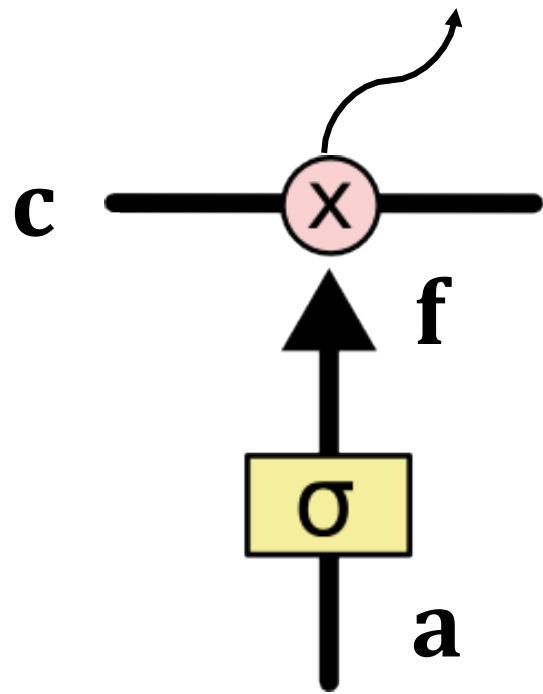


$$\sigma \begin{pmatrix} 1 \\ 3 \\ 0 \\ -2 \end{pmatrix} = \begin{bmatrix} 0.73 \\ 0.95 \\ 0.5 \\ 0.12 \end{bmatrix}$$

The diagram shows the sigmoid function  $\sigma$  applied to a vector  $a$  (represented by a curly brace) to produce a vector  $f$  (represented by another curly brace).

# LSTM: Forget Gate

Elementwise multiplication of 2 vectors.



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

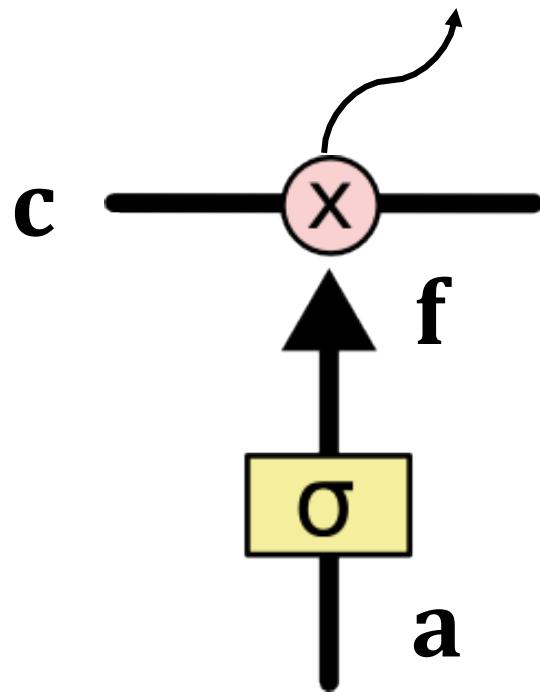
Diagram illustrating the elementwise multiplication of two vectors:

- 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.



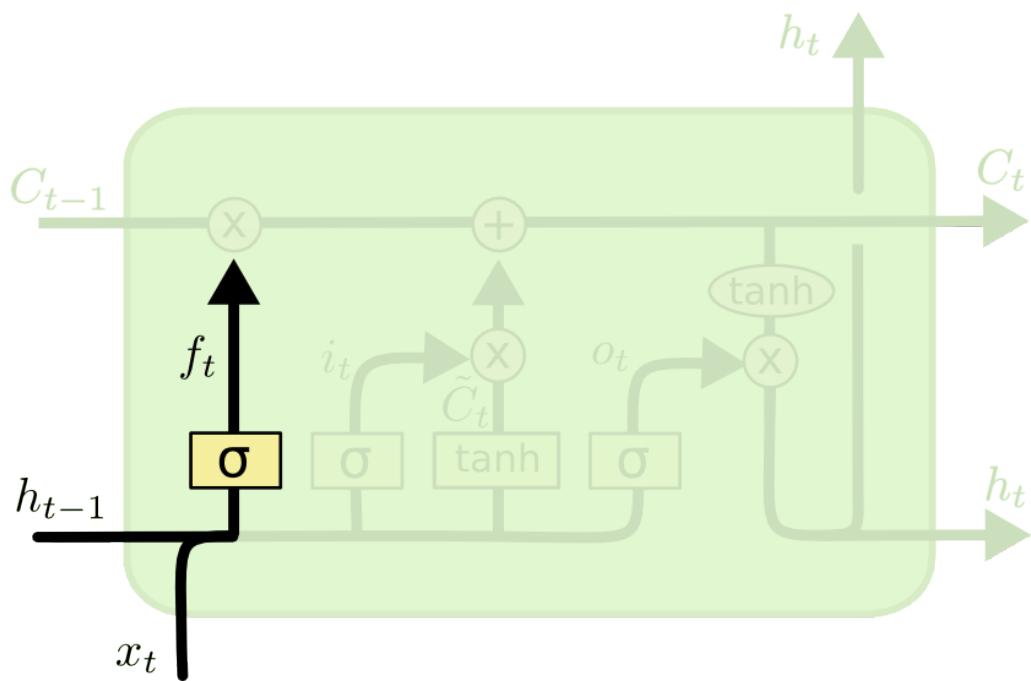
$$\begin{bmatrix} 0.9 \\ 0.2 \\ -0.5 \\ -0.1 \end{bmatrix} \circ \begin{bmatrix} 0.5 \\ 0 \\ 1 \\ 0.8 \end{bmatrix} = \begin{bmatrix} 0.45 \\ 0 \\ -0.5 \\ -0.08 \end{bmatrix}$$

Diagram illustrating the elementwise multiplication of two vectors:

- c**:  $\begin{bmatrix} 0.9 \\ 0.2 \\ -0.5 \\ -0.1 \end{bmatrix}$
- f**:  $\begin{bmatrix} 0.5 \\ 0 \\ 1 \\ 0.8 \end{bmatrix}$
- output**:  $\begin{bmatrix} 0.45 \\ 0 \\ -0.5 \\ -0.08 \end{bmatrix}$

# LSTM: Forget Gate

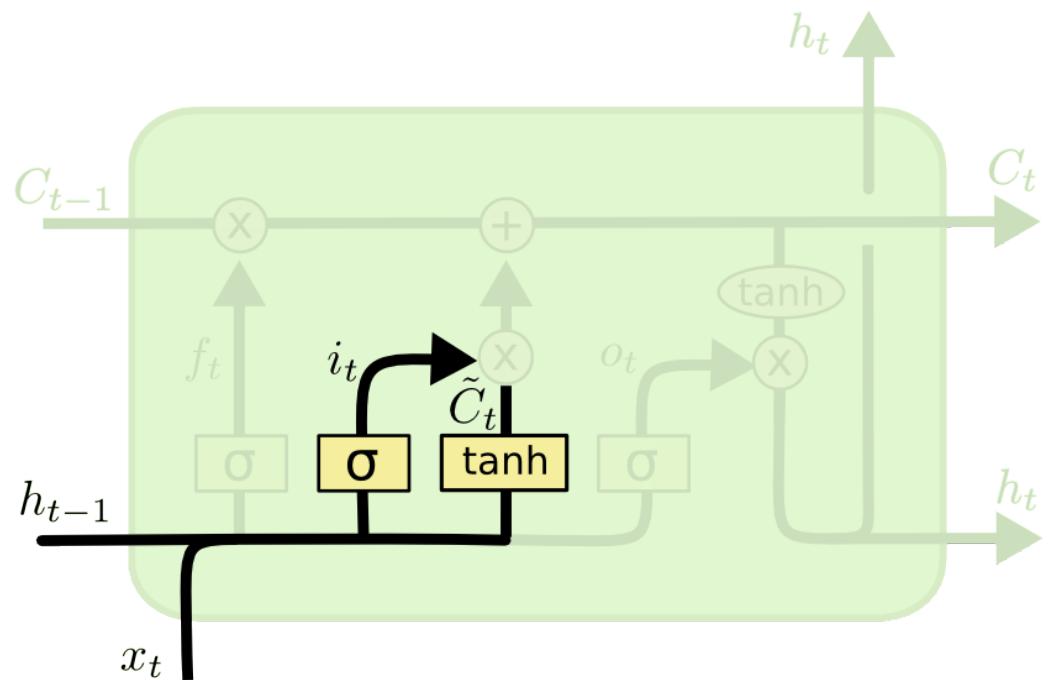
- Forget gate ( $f$ ): a vector (the same shape as  $\mathbf{c}$  and  $\mathbf{h}$ ).
  - A value of zero means “let nothing through”.
  - A value of one means “let everything through!”



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

# LSTM: Input Gate

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

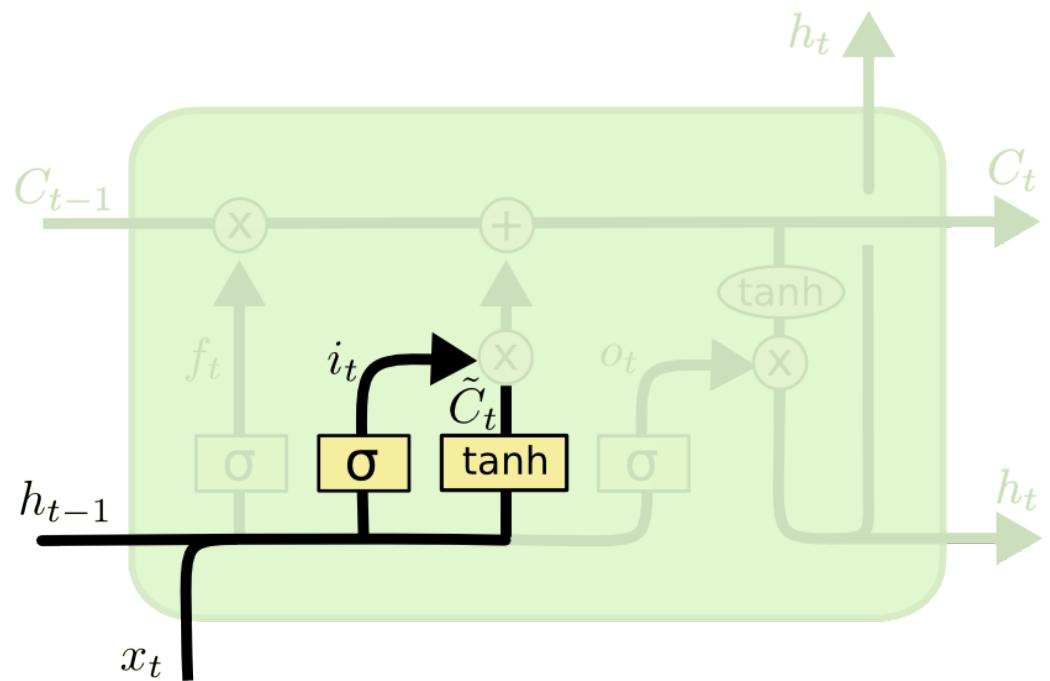


The diagram illustrates the computation of hidden states and outputs in a neural network layer. On the left, the input  $i_t$  is shown as a vertical vector of purple squares. An arrow points from  $i_t$  to the right, indicating its role in the computation. In the center, the input  $i_t$  is multiplied by a weight matrix  $W_i$ . The weight matrix  $W_i$  is represented as a grid of 16 squares, divided into two columns: a pink column on the left and a blue column on the right. A large bracket labeled  $\sigma$  indicates that the result of this multiplication is passed through an activation function (sigmoid). Below the matrix, a downward-pointing arrow is labeled  $W_i$ , identifying the weight matrix. To the right of the activation step, a dot indicates the result is multiplied by a bias vector  $b$ . This result is then passed through another activation function (sigmoid), represented by a vertical vector of squares. Two arrows point from this vector to the right: one labeled  $h_{t-1}$  representing the hidden state, and another labeled  $x_t$  representing the output.

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

# LSTM: New Value

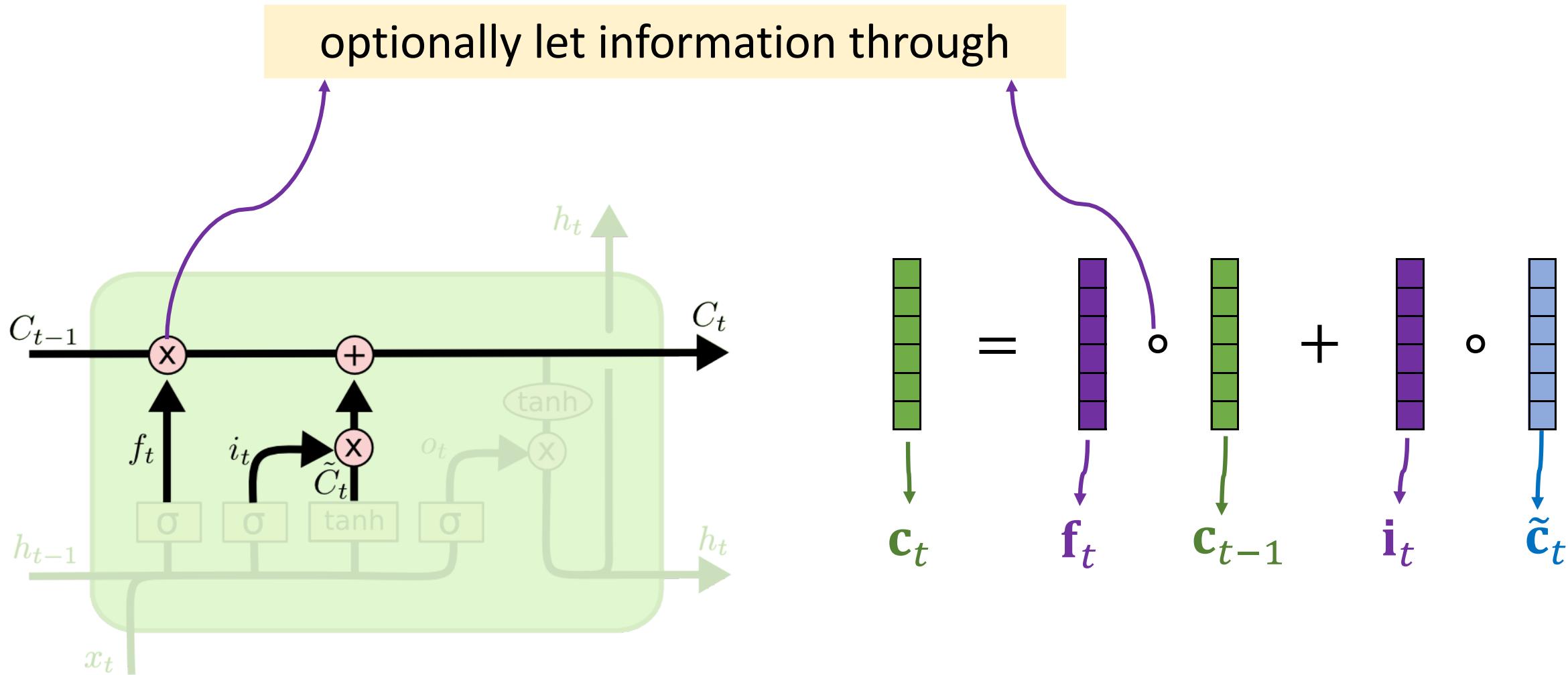
- New value ( $\tilde{c}_t$ ): to be added to the conveyor belt.



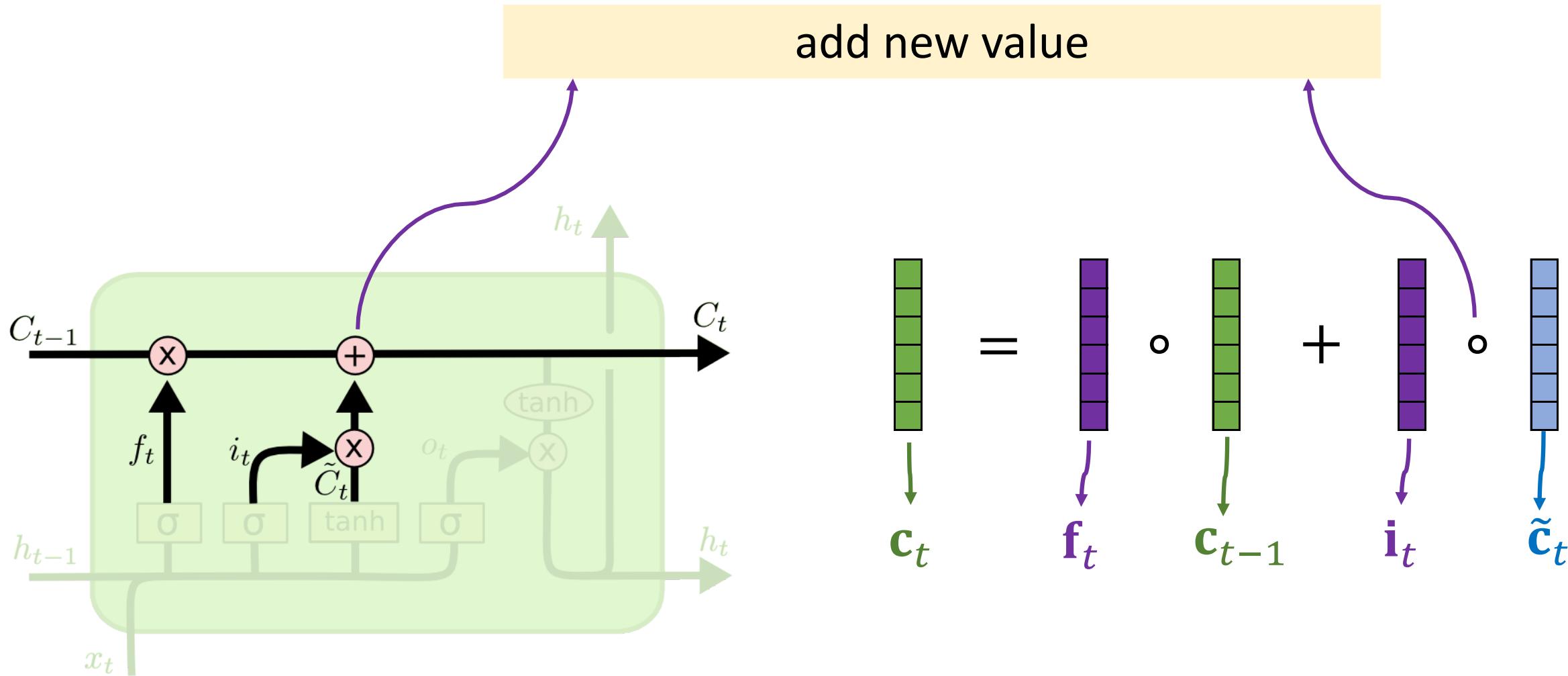
$$\tilde{c}_t = \tanh \left[ \begin{array}{c|c} \text{purple vertical bar} & \cdot \\ \hline \text{purple grid} & \cdot \\ \text{blue grid} & \cdot \\ \hline \text{blue vertical bar} & \cdot \\ \end{array} \right] \cdot \mathbf{W}_c$$

The diagram shows the computation of the new cell value  $\tilde{c}_t$ . It consists of a matrix multiplication of a vertical vector (containing the previous hidden state  $h_{t-1}$  and input  $x_t$ ) with a weight matrix  $\mathbf{W}_c$ , followed by a tanh activation function. The resulting vector is then multiplied by the output of the forget gate to produce  $\tilde{C}_t$ .

# LSTM: Update the Conveyor Belt



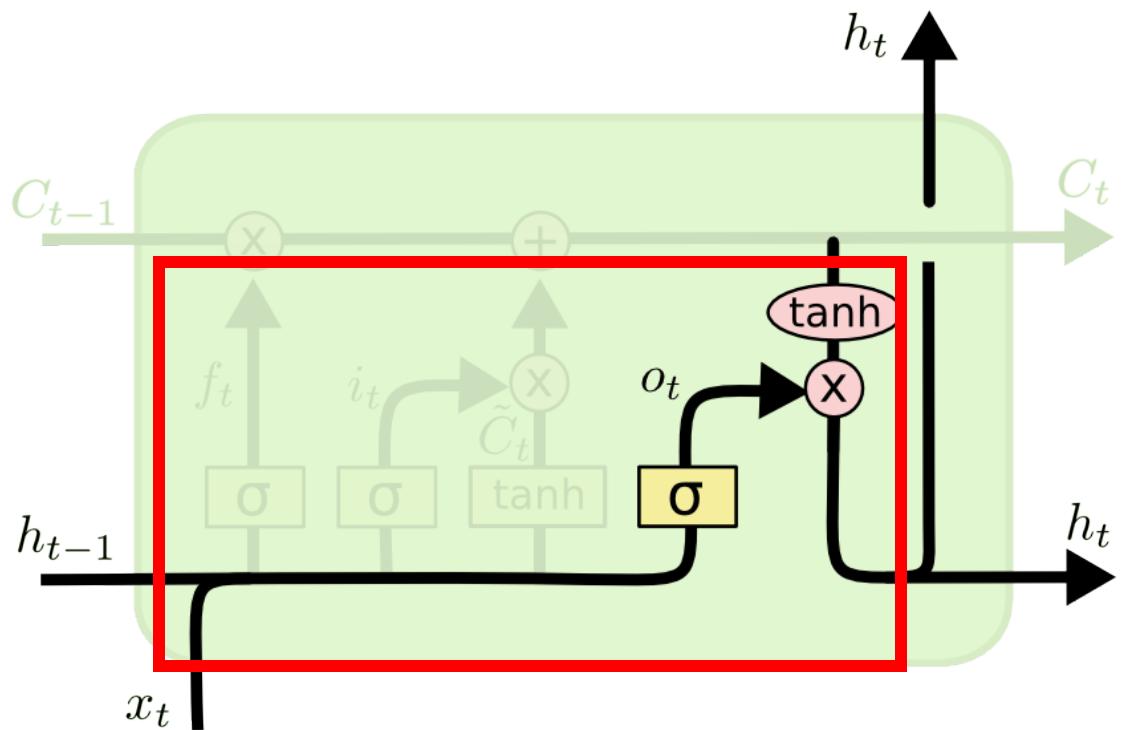
# LSTM: Update the Conveyor Belt



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

# LSTM: Output Gate

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

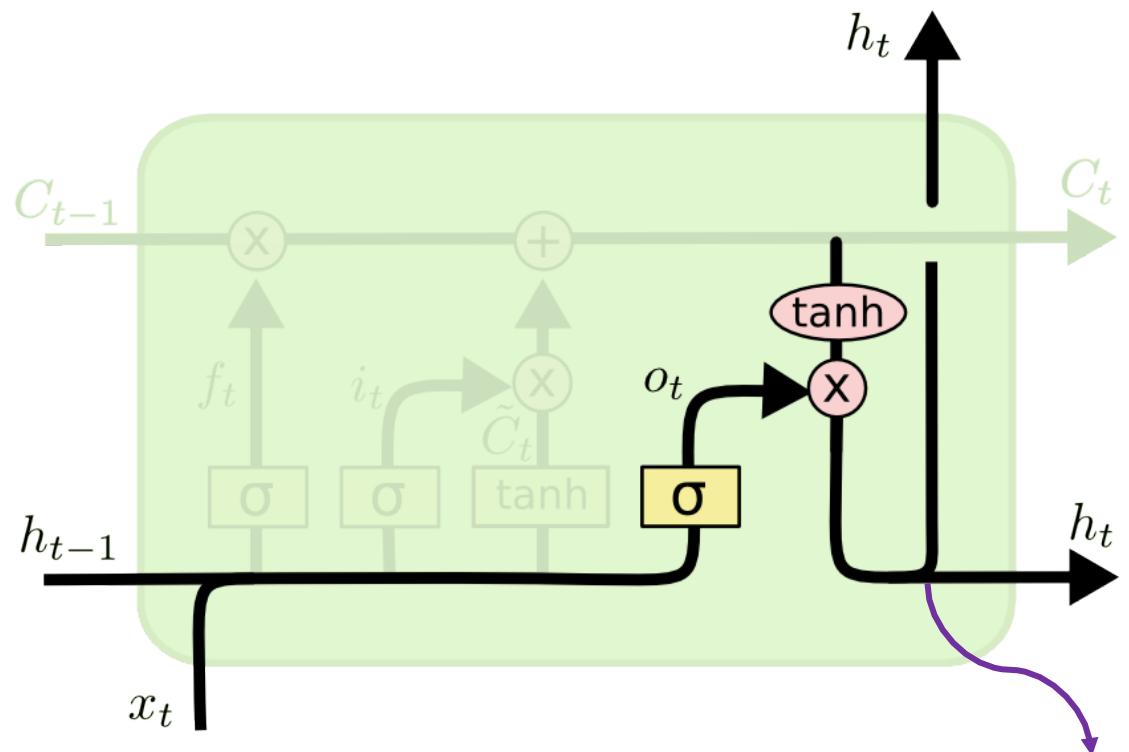


$$\mathbf{o}_t = \sigma \begin{bmatrix} \text{red vector} \\ \text{matrix} \end{bmatrix} \cdot \mathbf{W}_o$$

The diagram shows the mathematical representation of the output gate. The output  $\mathbf{o}_t$  is calculated as the sigmoid function ( $\sigma$ ) applied to a matrix multiplication of a vector and a weight matrix  $\mathbf{W}_o$ . The vector consists of the previous hidden state  $h_{t-1}$  (represented by a purple block) and the current cell state  $C_t$  (represented by a blue block).

# LSTM: Update State

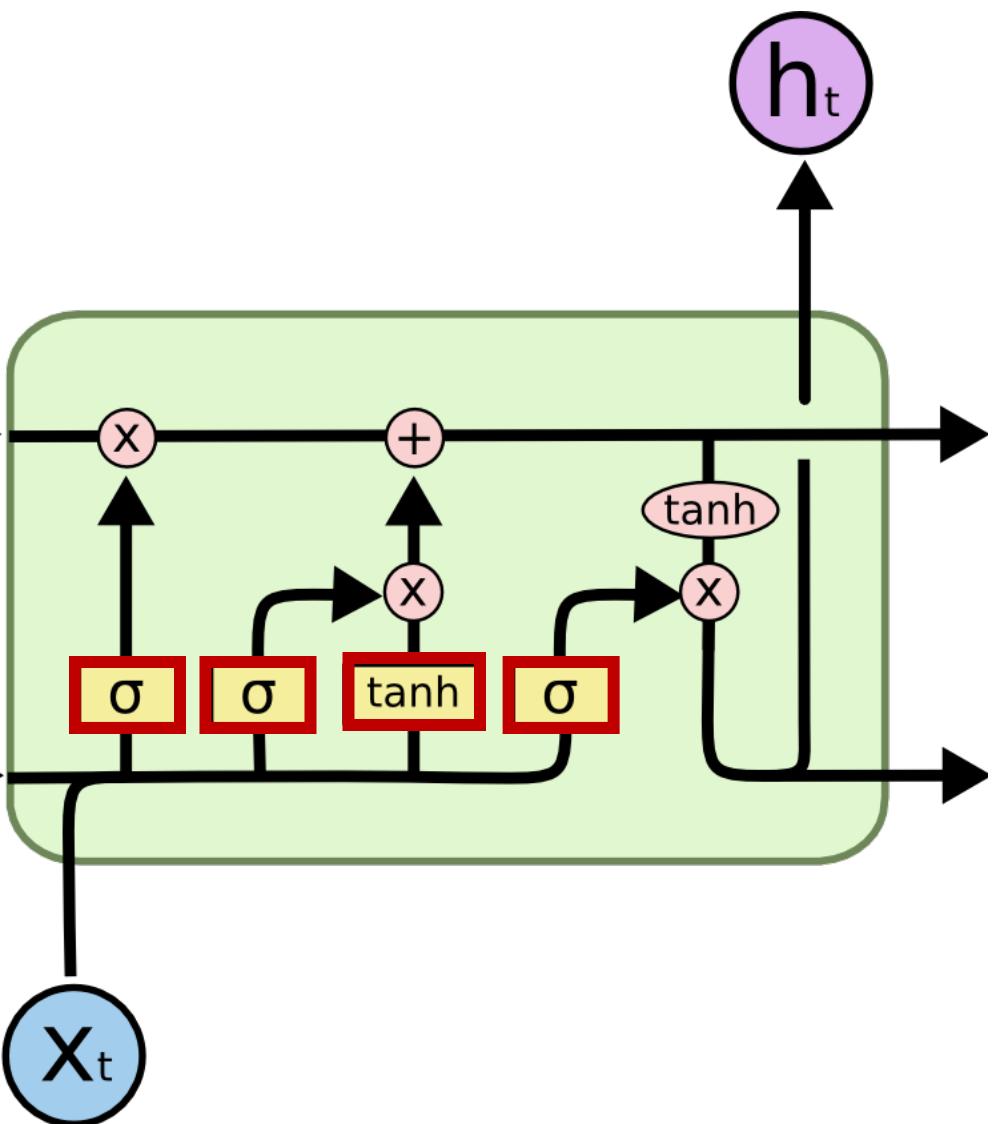
- State ( $\mathbf{h}_t$ ): the output of LSTM.



Two copies of  $h_t$

$$\mathbf{h}_t = \sigma \circ \tanh \left[ \begin{array}{c} \text{red} \\ \text{red} \\ \text{red} \end{array} \right] = \mathbf{o}_t \circ \tanh \left[ \begin{array}{c} \text{red} \\ \text{red} \\ \text{red} \end{array} \right] + \mathbf{c}_t$$

# LSTM: Number of Parameters

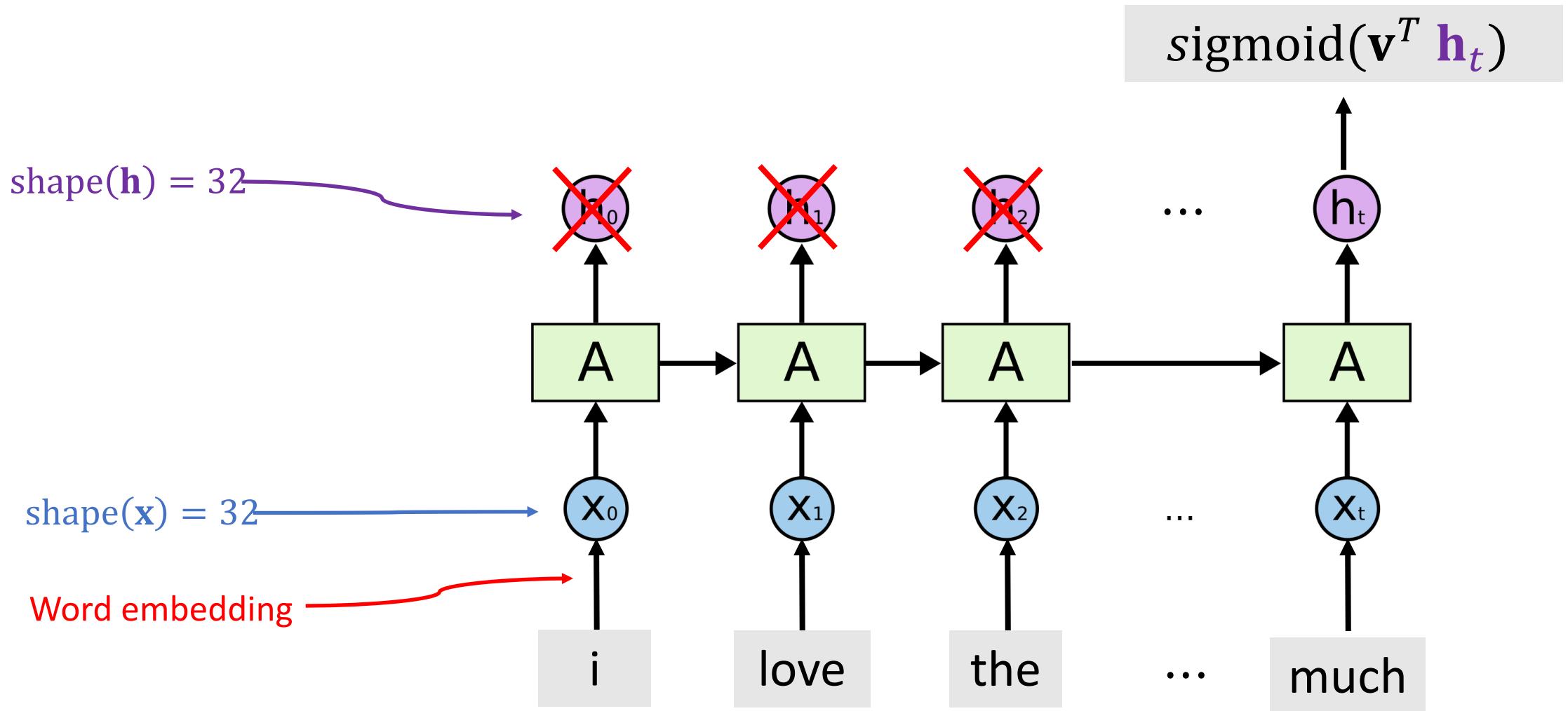


#parameters: **4 times as many as SimpleRNN**

- **4** parameter matrices, each of which has
  - #rows: shape ( $\mathbf{h}$ )
  - #cols: shape ( $\mathbf{h}$ ) + shape ( $\mathbf{x}$ )
- #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) number of words	320000 vocabulary × embedding_dim
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%).

Zhiqiu:

# LSTM Dropout

Dropout does not improve the accuracy because the overfitting is caused by the number of parameters in the embedding layer.

```
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!**