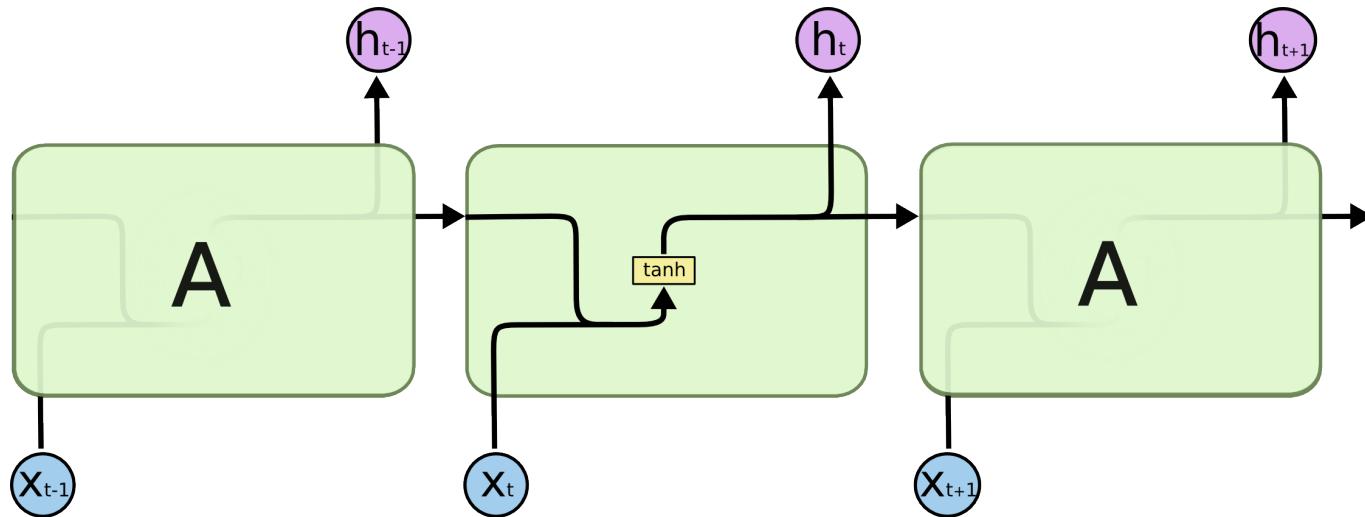


LSTM Model

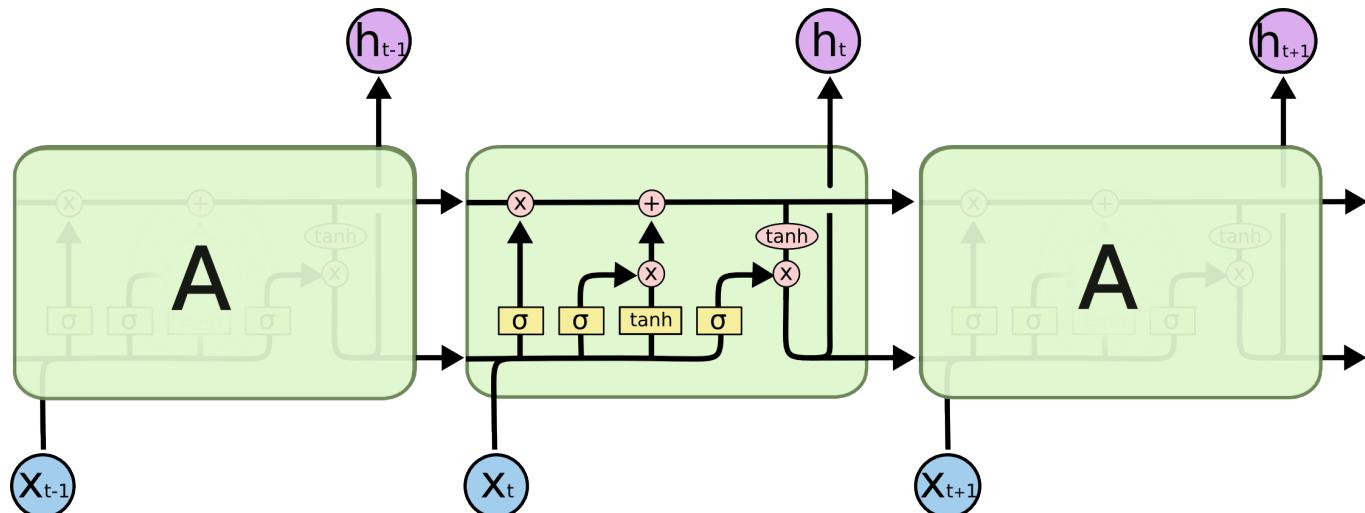
Reference

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

LSTM Networks



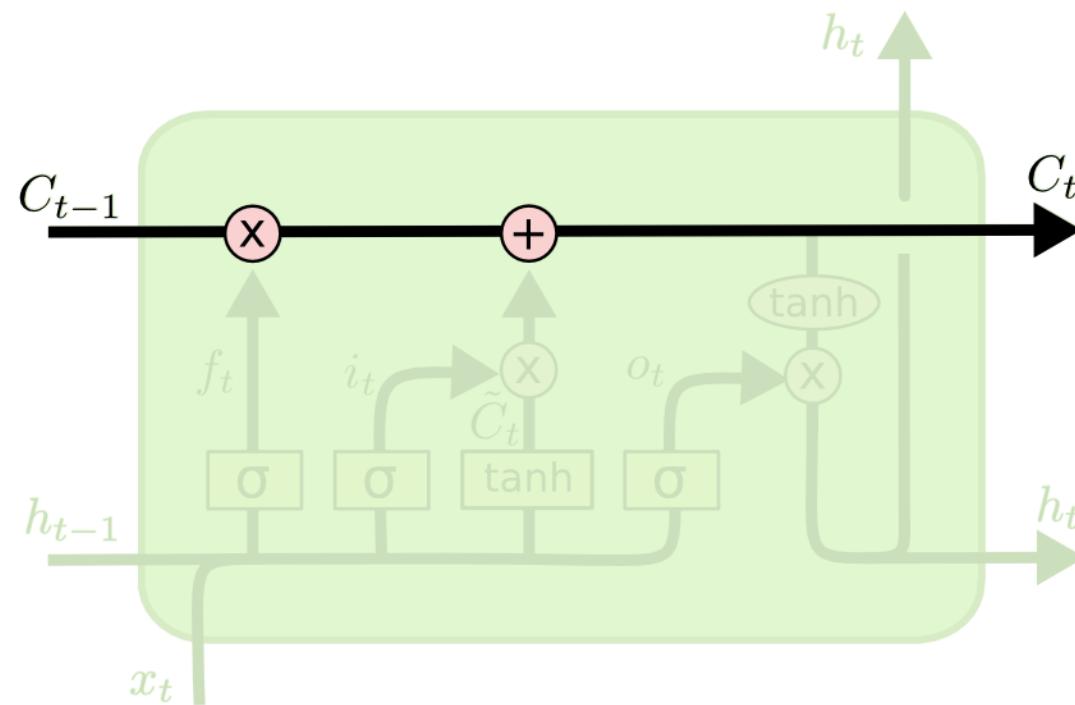
Simple RNN



LSTM

LSTM: Conveyor Belt

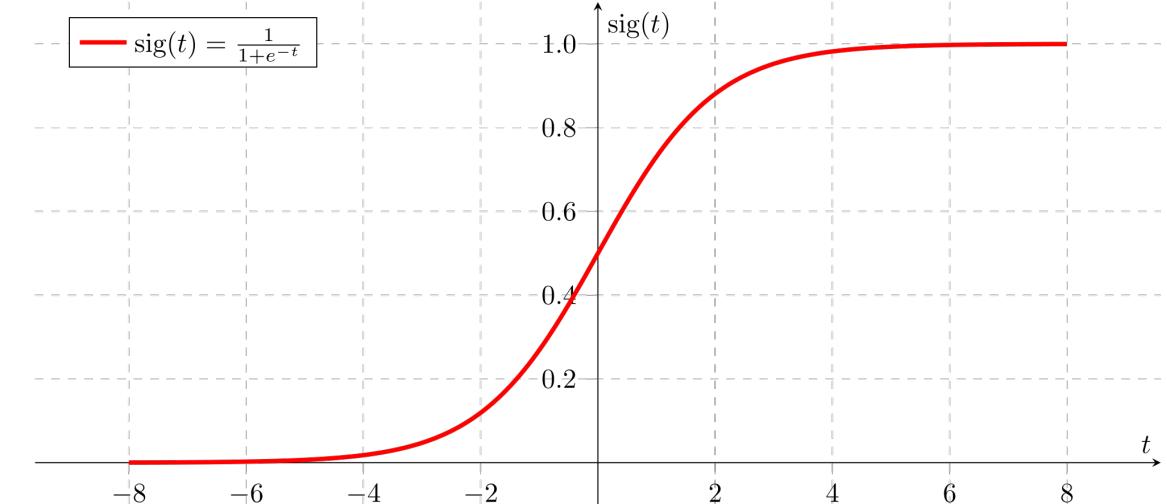
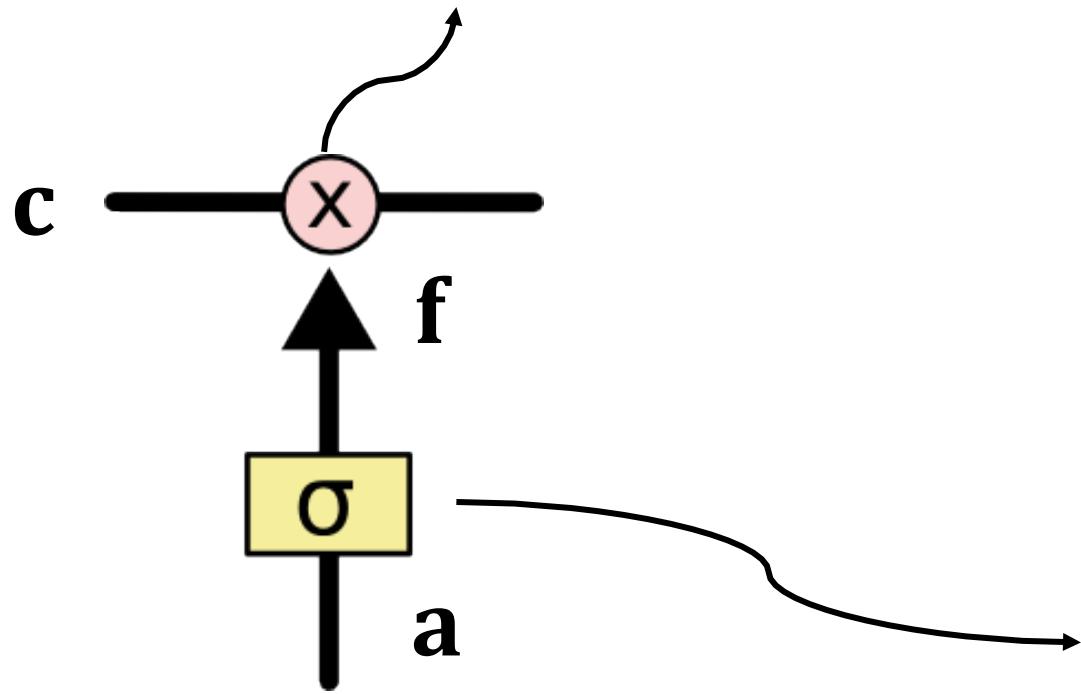
- Conveyor belt: the past information directly flows to the future.



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

LSTM: Forget Gate

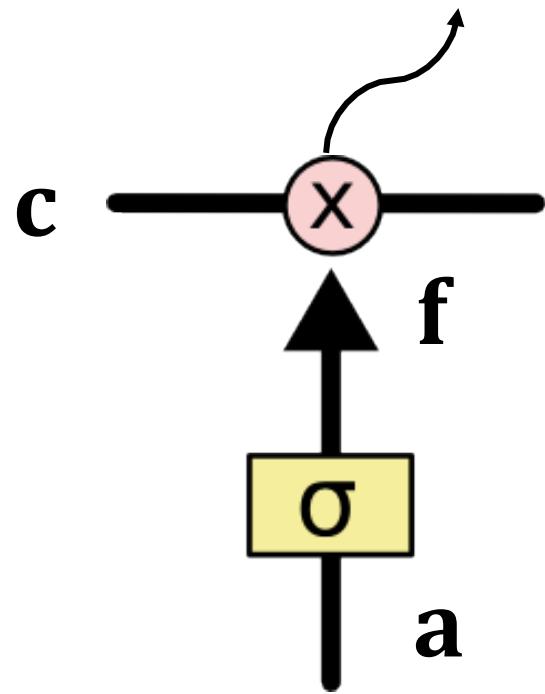
Elementwise multiplication of 2 vectors.



Sigmoid function: between 0 and 1.

LSTM: Forget Gate

Elementwise multiplication of 2 vectors.

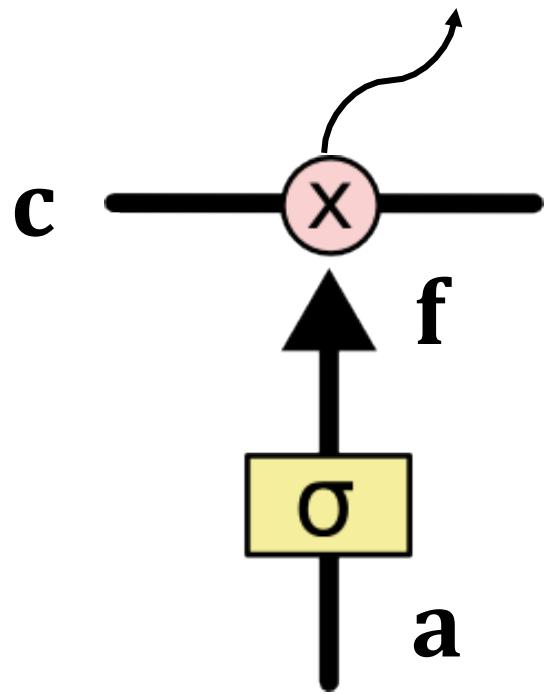


$$\sigma \begin{pmatrix} 1 \\ 3 \\ 0 \\ -2 \end{pmatrix} = \begin{bmatrix} 0.73 \\ 0.95 \\ 0.5 \\ 0.12 \end{bmatrix}$$

The diagram shows the elementwise multiplication of the vector a (bottom left) and the vector f (bottom right) to produce the output vector. Arrows indicate the mapping from the input vectors to the output vector.

LSTM: Forget Gate

Elementwise multiplication of 2 vectors.



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

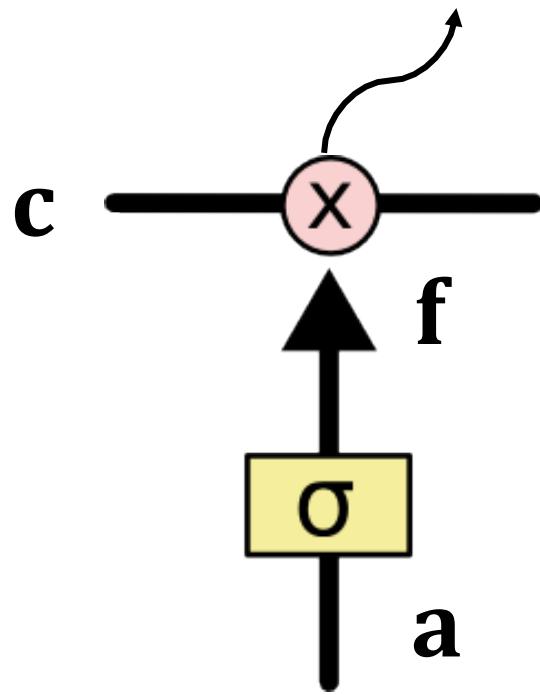
Diagram illustrating the elementwise multiplication of two vectors:

- 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.



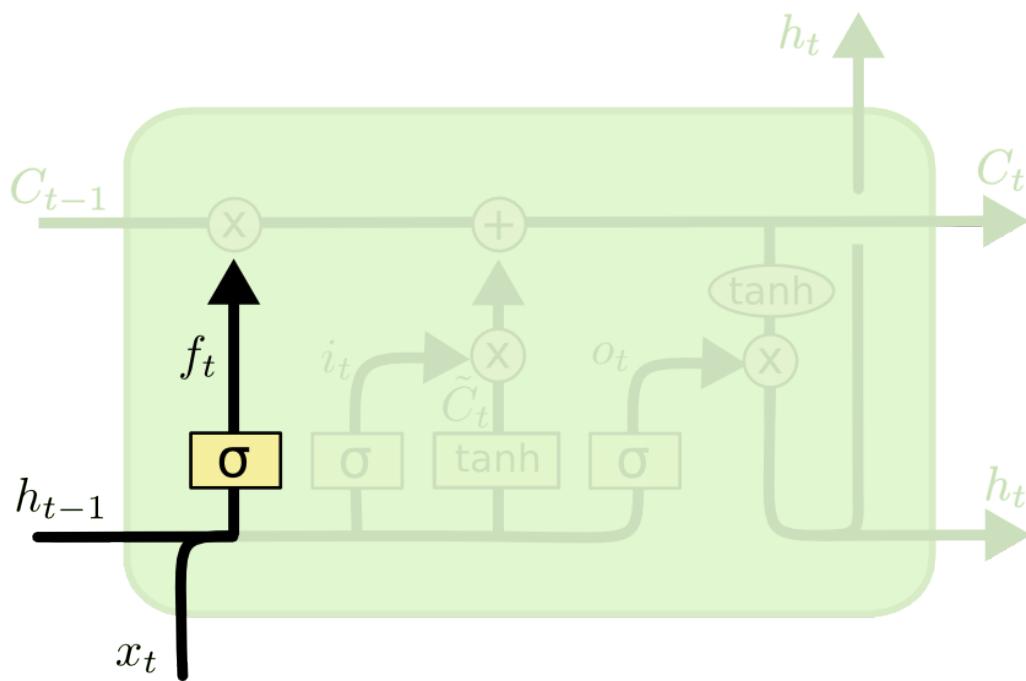
$$\begin{bmatrix} 0.9 \\ 0.2 \\ -0.5 \\ -0.1 \end{bmatrix} \circ \begin{bmatrix} 0.5 \\ 0 \\ 1 \\ 0.8 \end{bmatrix} = \begin{bmatrix} 0.45 \\ 0 \\ -0.5 \\ -0.08 \end{bmatrix}$$

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

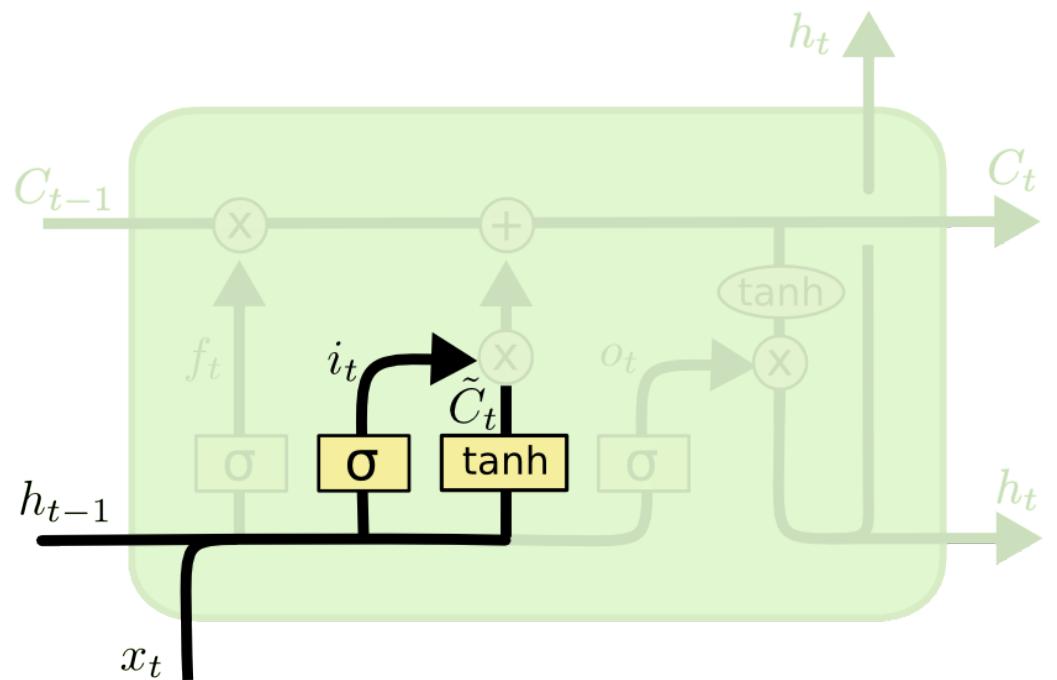
- Forget gate (f): a vector (the same shape as \mathbf{c} and \mathbf{h}).
 - A value of zero means “let nothing through”.
 - A value of one means “let everything through!”



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

LSTM: Input Gate

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

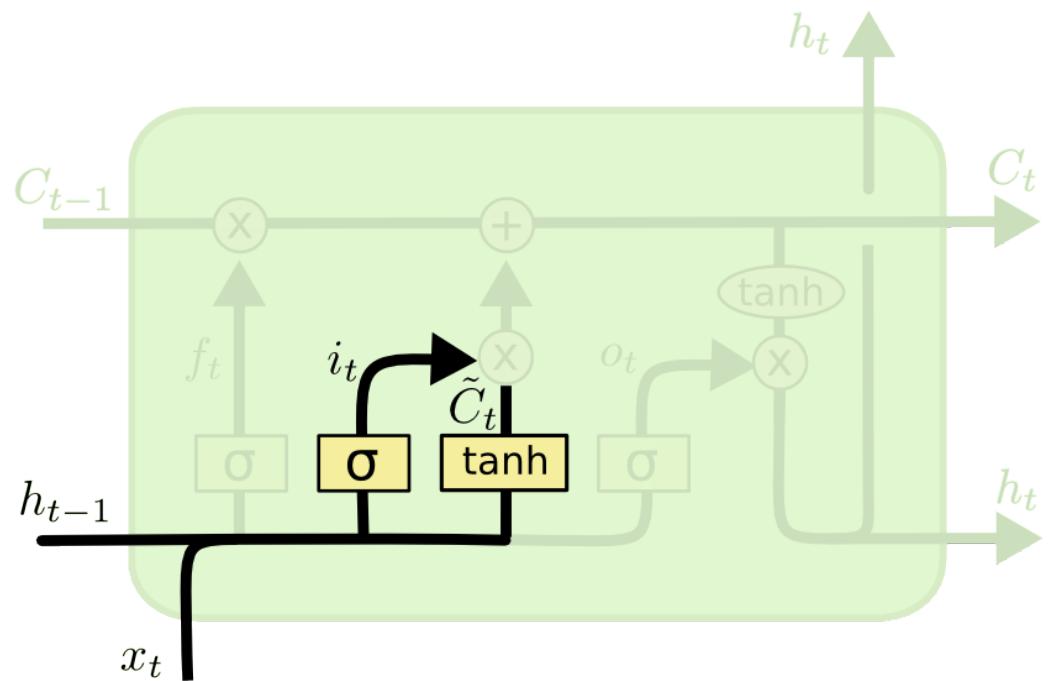


The diagram illustrates the computation of the hidden state h_{t-1} and output x_t from the previous hidden state h_{t-1} and input i_t . The input i_t is passed through a weight matrix W_i (represented as a grid of pink and blue squares) and a bias vector (represented as a vertical column of purple squares). The result is passed through a sigmoid function (σ) to produce the hidden state h_{t-1} . The output x_t is produced by applying the weight matrix W_i to the hidden state h_{t-1} .

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

LSTM: New Value

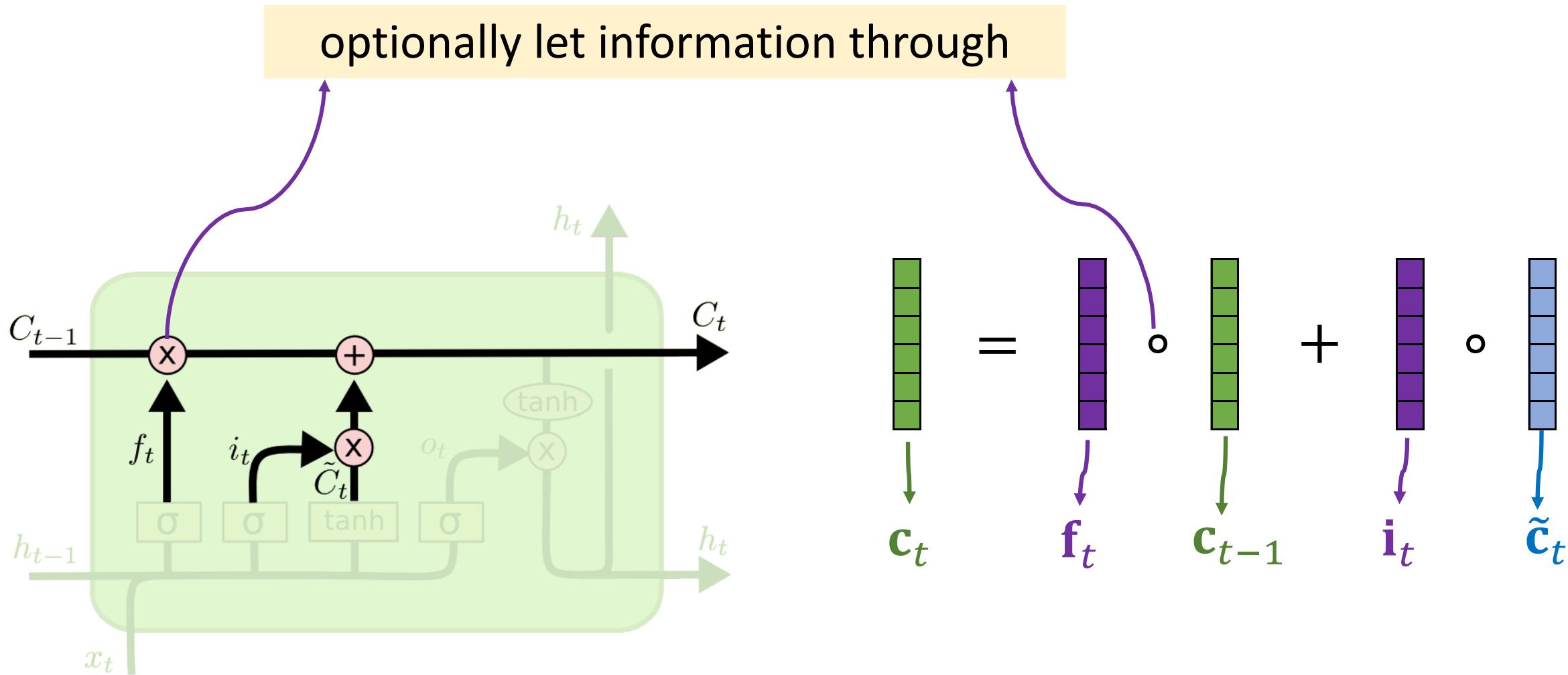
- New value (\tilde{c}_t): to be added to the conveyor belt.



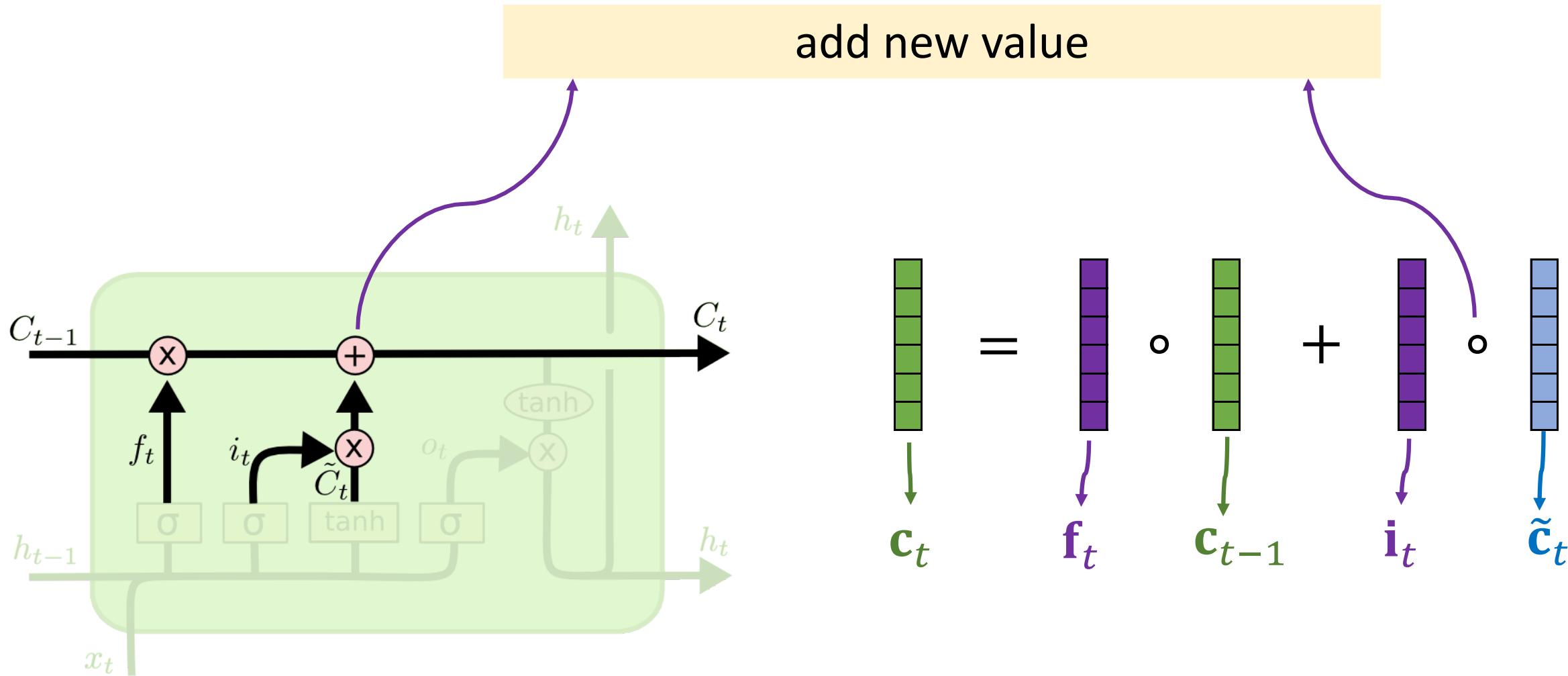
$$\tilde{c}_t = \tanh \left[\begin{array}{c|c} \text{purple} & \text{blue} \\ \hline \text{purple} & \text{blue} \\ \text{purple} & \text{blue} \\ \text{purple} & \text{blue} \\ \text{purple} & \text{blue} \end{array} \right] \cdot W_c$$

The diagram shows a vertical stack of four purple rectangles representing the hidden state h_{t-1} and a vertical stack of four blue rectangles representing the input x_t . A bracket groups these two stacks, with a dot indicating multiplication by the weight matrix W_c .

LSTM: Update the Conveyor Belt

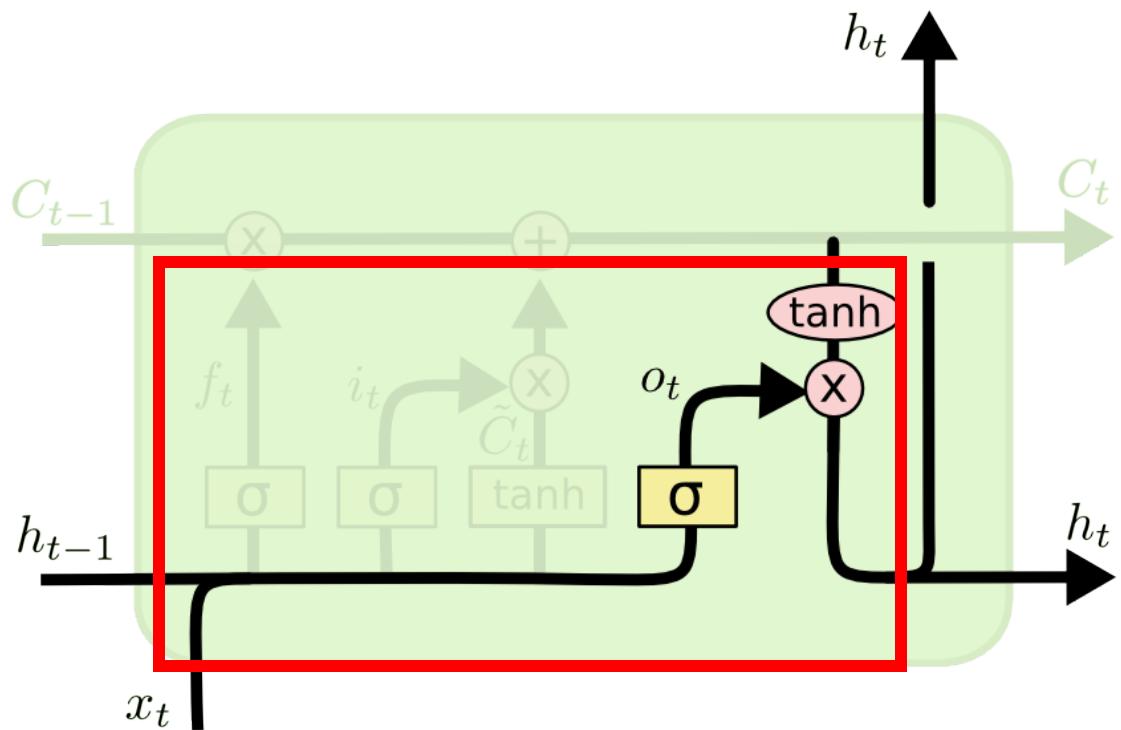


LSTM: Update the Conveyor Belt



LSTM: Output Gate

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

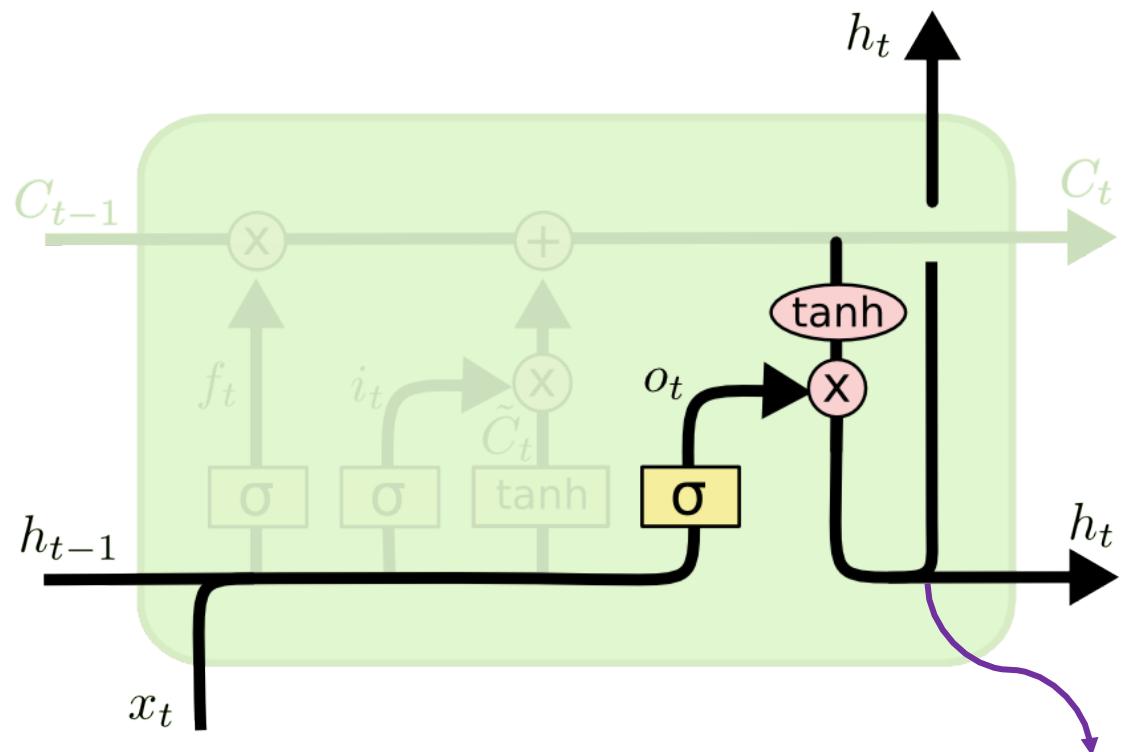


$$\mathbf{o}_t = \sigma \begin{bmatrix} \text{red vertical vector} \\ \text{matrix} \end{bmatrix} \cdot \mathbf{W}_o + \mathbf{b}_o$$

The equation shows the computation of the output gate vector \mathbf{o}_t . It consists of two parts: a scalar σ (red arrow) and a matrix multiplication (black bracket) between a vertical vector (red) and a weight matrix \mathbf{W}_o . The result is then added to a bias vector \mathbf{b}_o (blue arrow).

LSTM: Update State

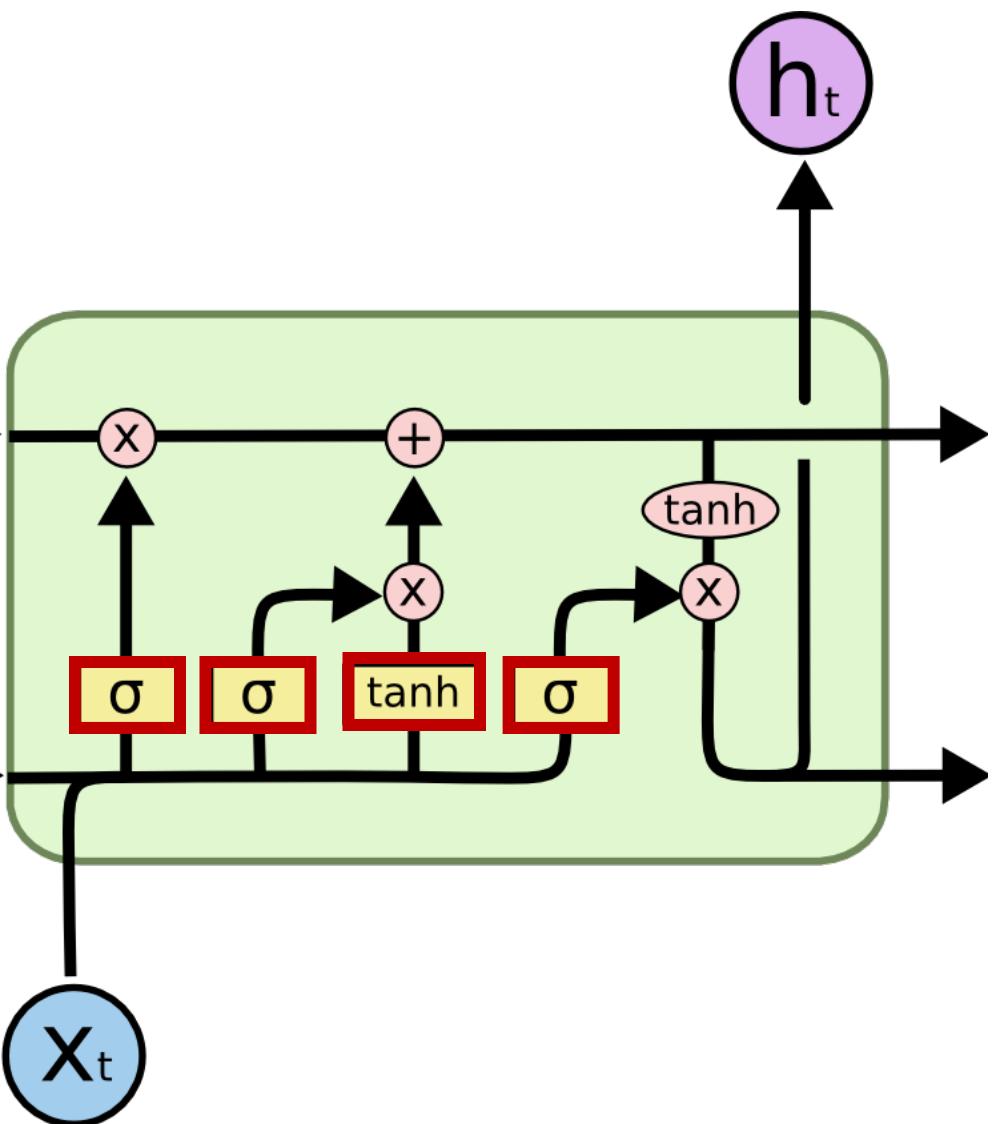
- State (\mathbf{h}_t): the output of LSTM.



Two copies of h_t

$$\mathbf{h}_t = \sigma \circ \tanh \left[\begin{array}{c} \mathbf{h}_{t-1} \\ \mathbf{o}_t \end{array} \right] \quad \mathbf{C}_t = \mathbf{f}_t \circ \mathbf{C}_{t-1} + \mathbf{i}_t \circ \tanh \left[\begin{array}{c} \mathbf{h}_{t-1} \\ \mathbf{o}_t \end{array} \right]$$

LSTM: Number of Parameters

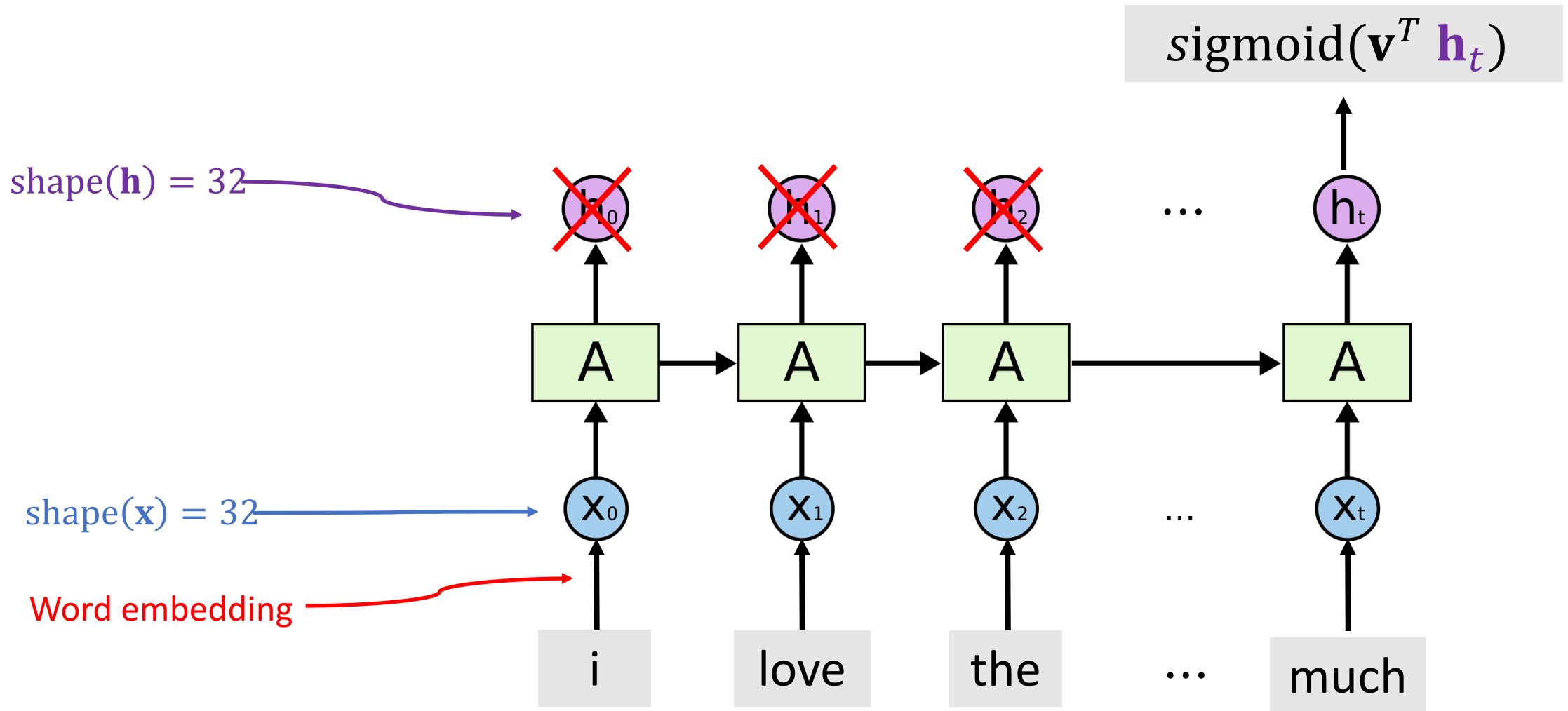


#parameters: **4 times as many as SimpleRNN**

- **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!