# Long Short Term Memory (LSTM)

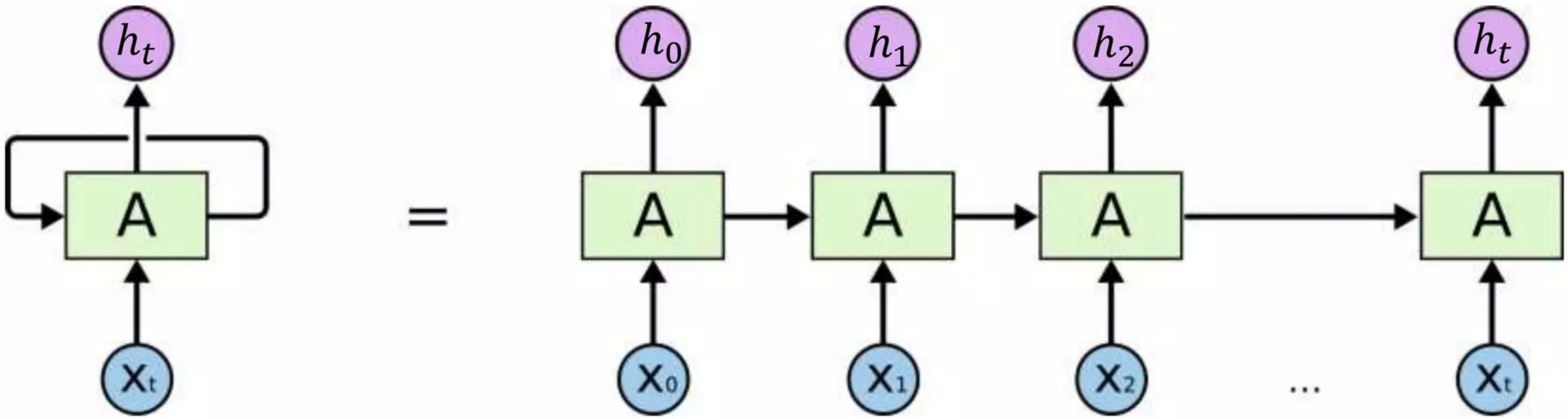
Anas Haddaoui
Maroine Amzil
Ayyoub Jaichi
Zakaria Lagouader
Hamza Bouktitiya

# Contents

- Introduction
- Vanishing/Exploding Gradient Problem
- Long Short Term Memory
- LSTM Variations

# Introduction

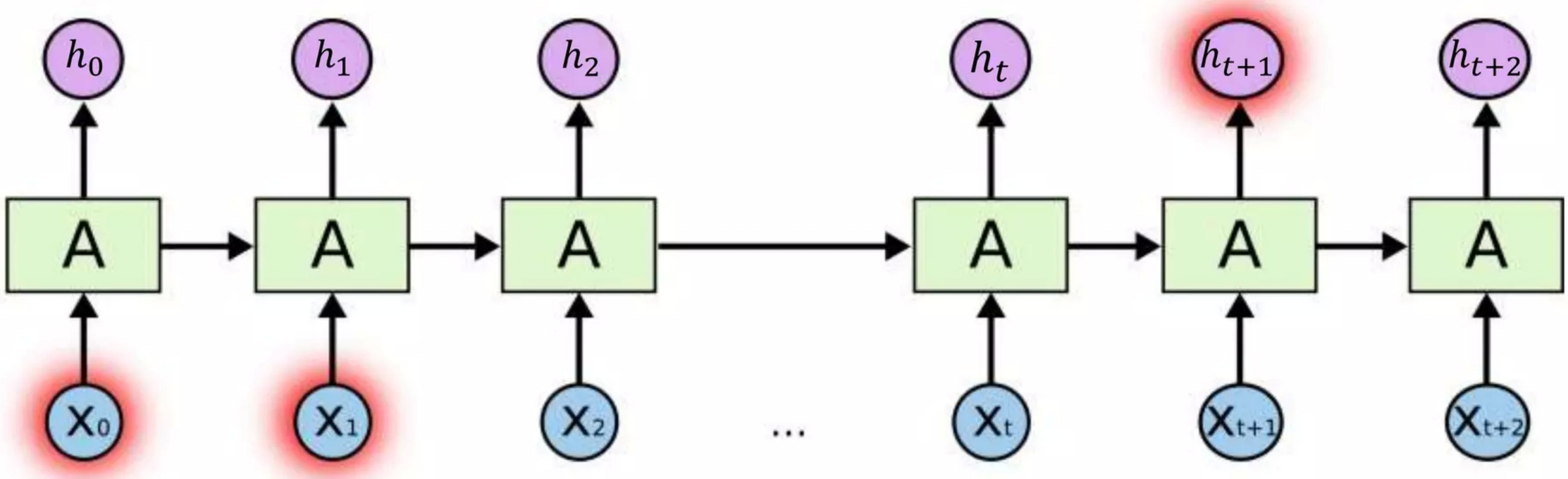
- LSTM is a kind of RNN.
- LSTM is capable of learning long term dependencies.




An unrolled recurrent neural network

# Introduction

- RNN is unable to learn to connect the information in large gap.
- LSTM don't have large gap problem.



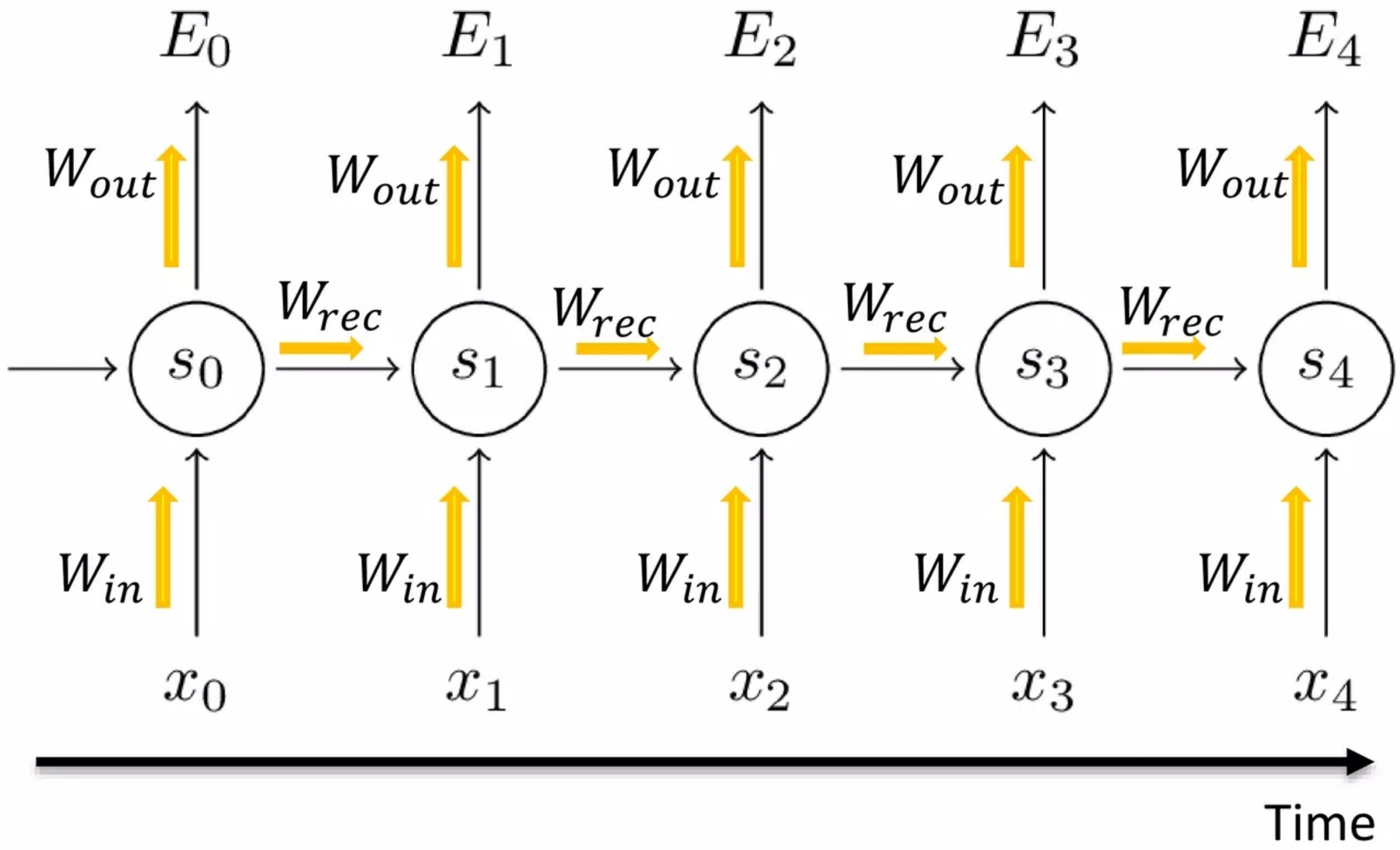

- The clouds are in the *sky*.
- I grew up in France. … I speak fluent *French*.

# Introduction

- **Using LSTM:**
  - Robot control
  - Time series prediction
  - Speech recognition
  - Rhythm learning
  - Music composition
  - Grammar learning
  - Handwriting recognition
  - Human action recognition
  - End to end translation

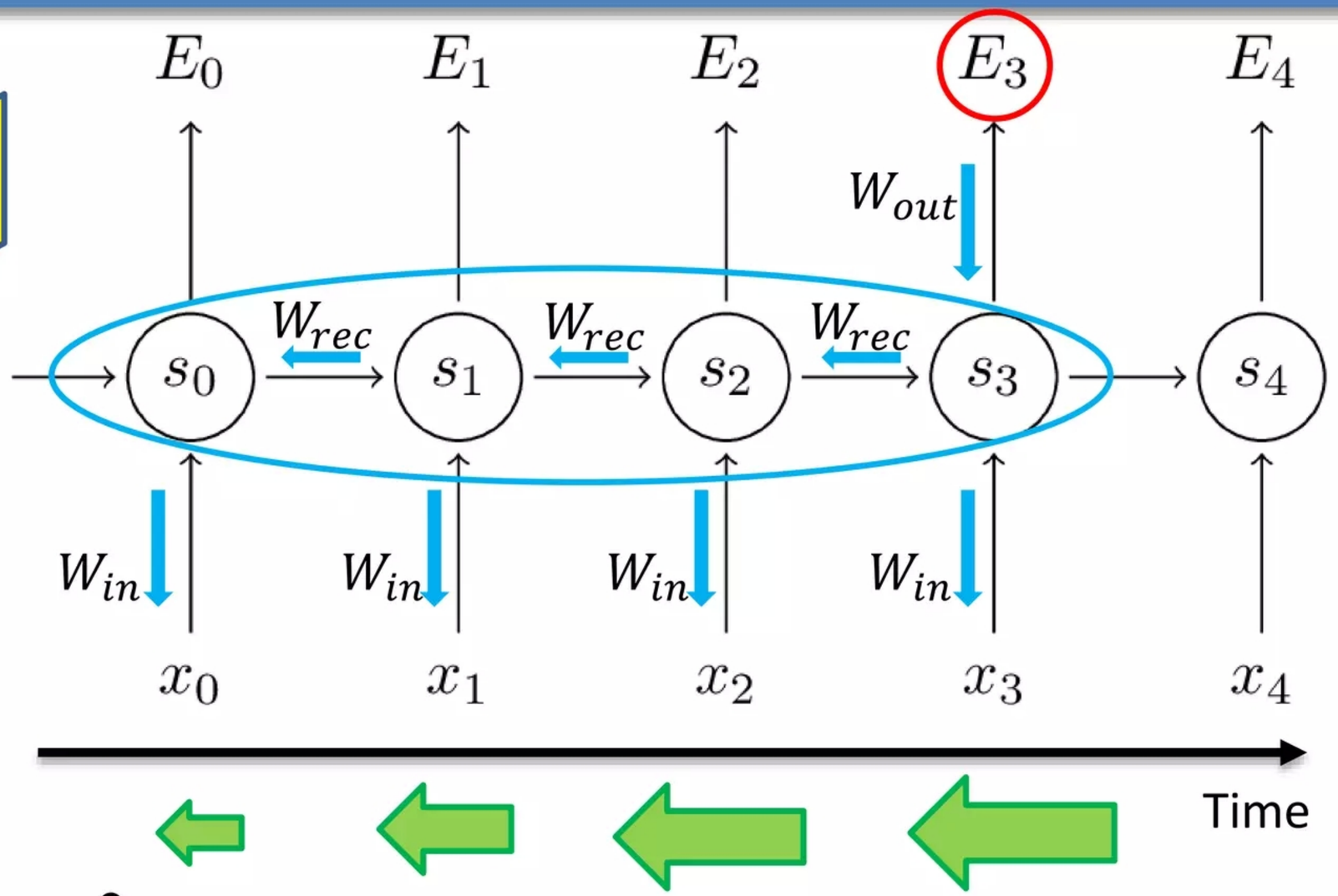
# Vanishing Gradient

- **RNN:**
- Sharing the same parameters at all time steps
- Occurs in time-series with long-term dependencies

# Vanishing Gradient

**Back propagation**




$$\frac{\partial E_3}{\partial w_{rec}} = \frac{\partial E_3}{\partial \hat{y}_3} \times \frac{\partial \hat{y}_3}{\partial s_3} \times \frac{\partial s_3}{\partial w_{rec}}$$

$$s_3 = \tanh\left(w_{in}\, x_3 + w_{rec}\, s_2\right)$$

$$\frac{\partial E_3}{\partial w_{rec}} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3} \times \frac{\partial \hat{y}_3}{\partial s_3} \times \frac{\partial s_3}{\partial s_k} \times \frac{\partial s_k}{\partial w_{rec}}$$

# Vanishing Gradient

$$s_3 = \tanh\left(w_{in}\, x_3 + w_{rec}\, s_2\right) \qquad (\tanh)' \in [0,1]$$

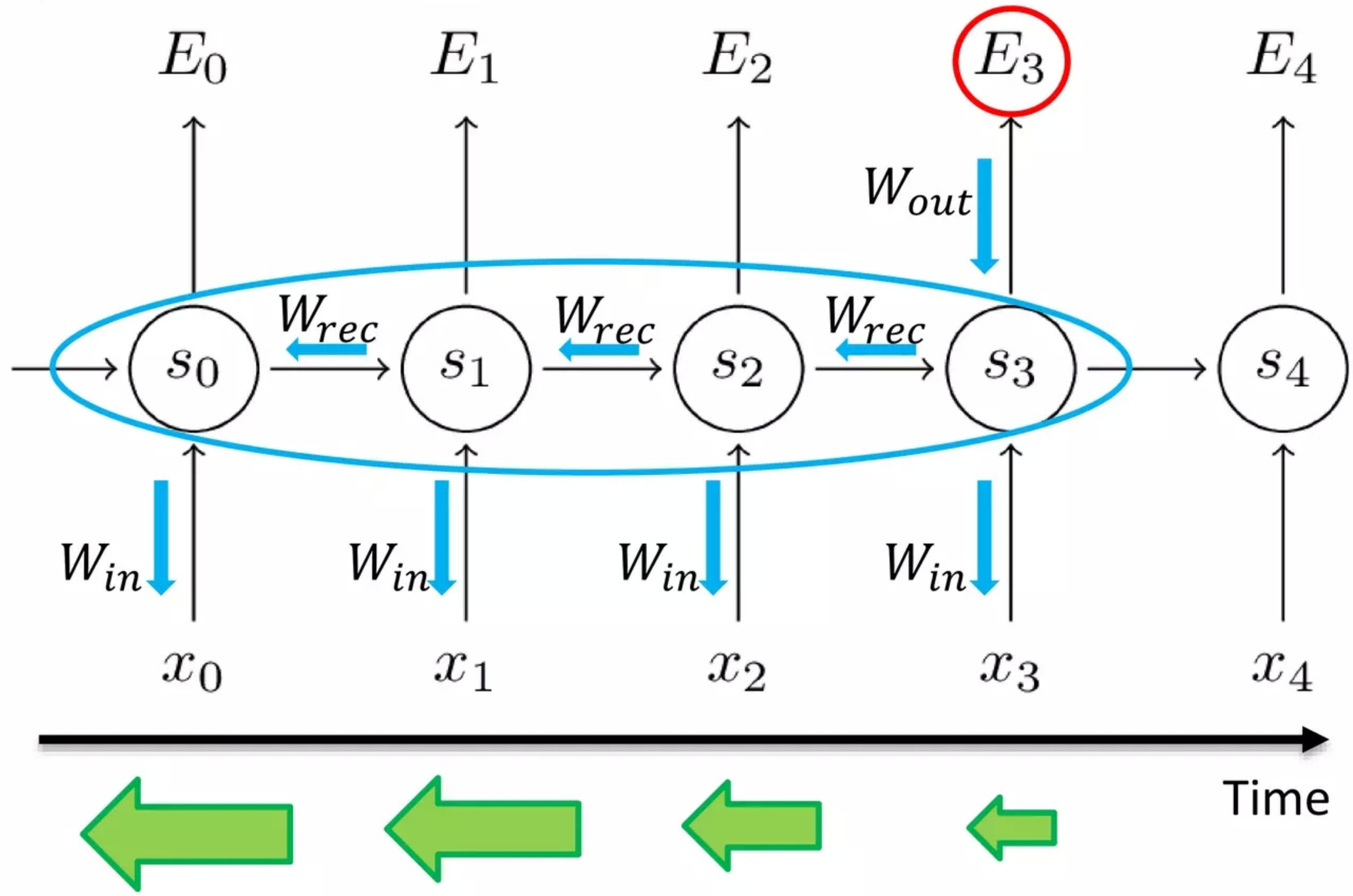
$$\frac{\partial E_3}{\partial w_{rec}} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3} \times \frac{\partial \hat{y}_3}{\partial s_3} \times \frac{\partial s_3}{\partial s_k} \times \frac{\partial s_k}{\partial w_{rec}}$$

$$\frac{\partial E_3}{\partial w_{rec}} = \frac{\partial E_3}{\partial \hat{y}_3} \times \frac{\partial \hat{y}_3}{\partial s_3} \times \frac{\partial s_3}{\partial s_2} \times \frac{\partial s_2}{\partial w_{rec}} + \frac{\partial E_3}{\partial \hat{y}_3} \times \frac{\partial \hat{y}_3}{\partial s_3} \times \underbrace{\frac{\partial s_3}{\partial s_1}}_{\frac{\partial s_3}{\partial s_2} \times \frac{\partial s_2}{\partial s_1}} \times \frac{\partial s_1}{\partial w_{rec}} + \frac{\partial E_3}{\partial \hat{y}_3} \times \frac{\partial \hat{y}_3}{\partial s_3} \times \underbrace{\frac{\partial s_3}{\partial s_0}}_{\frac{\partial s_3}{\partial s_2} \times \frac{\partial s_2}{\partial s_1} \times \frac{\partial s_1}{\partial s_0}} \times \frac{\partial s_0}{\partial w_{rec}}$$

$$\frac{\partial E_3}{\partial w_{rec}} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3} \times \frac{\partial \hat{y}_3}{\partial s_3} \times \left( \prod_{j=k+1}^{3} \frac{\partial s_j}{\partial s_{j-1}} \right) \times \frac{\partial s_k}{\partial w_{rec}}$$

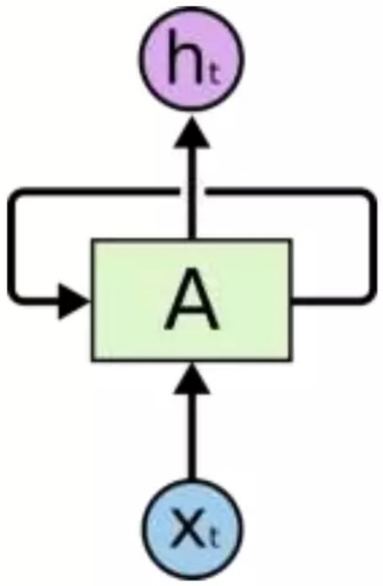
# Exploding Gradient

- Increasing weights
- Large errors
- Unstability in the network



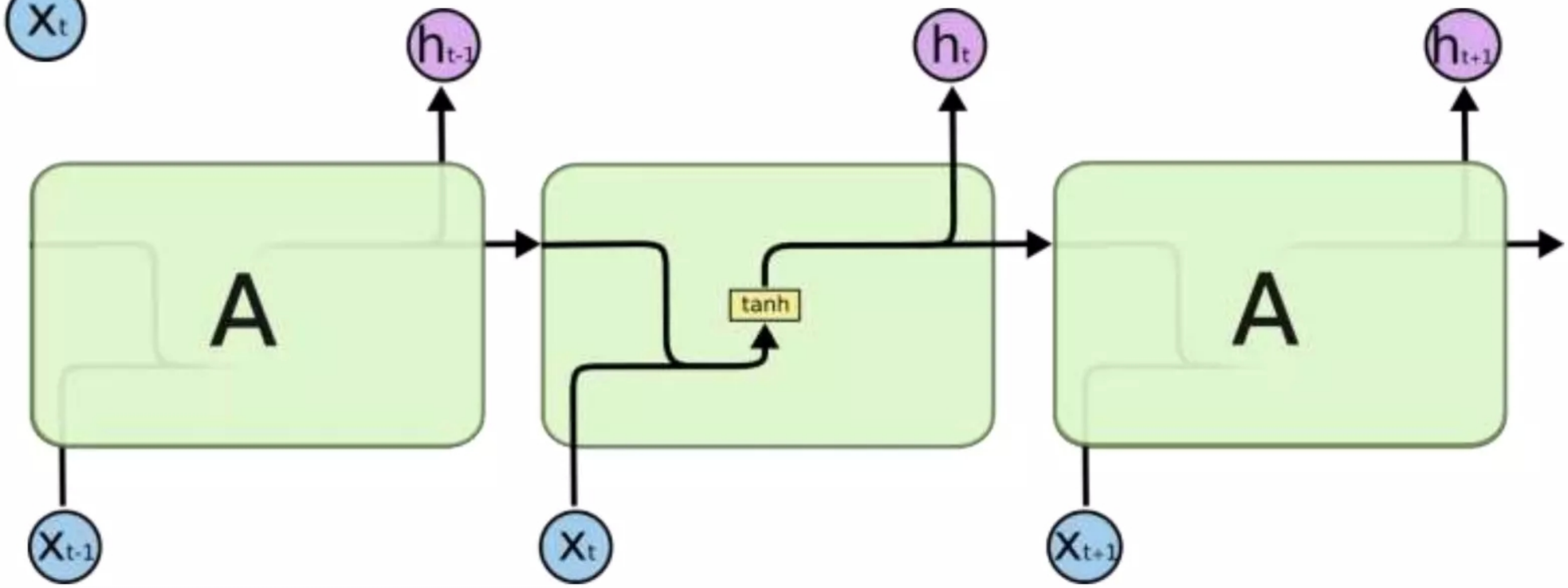
# Long Short Term Memory

– **Vanilla RNN:**



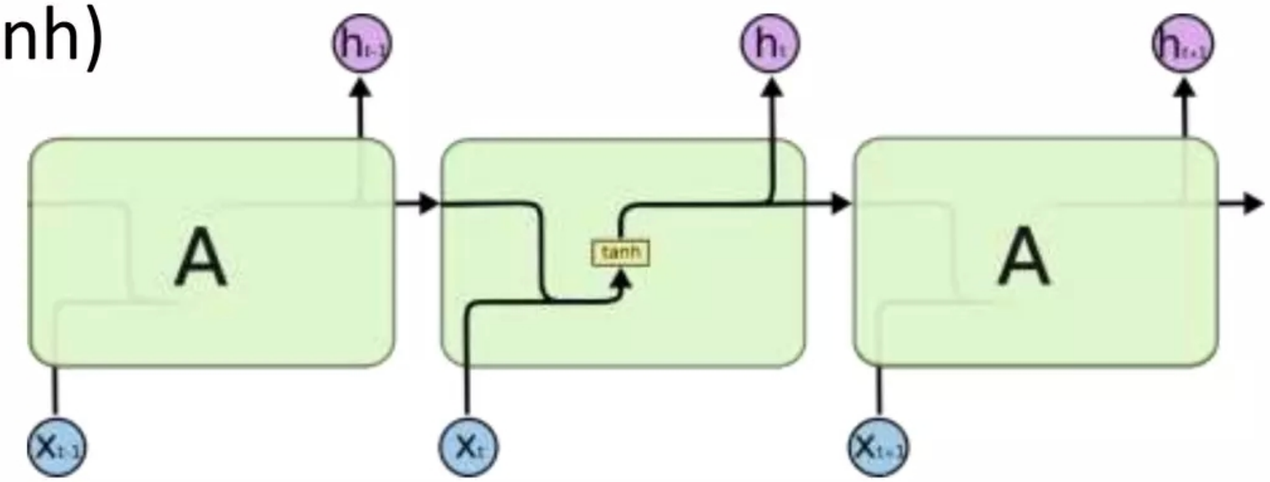

$$h_t = fw\left(h_{t-1}, x_t\right)$$

**Example:** $h_t = \tanh\left(w_{hh}\, h_{t-1} + w_{xh}\, x_t\right)$

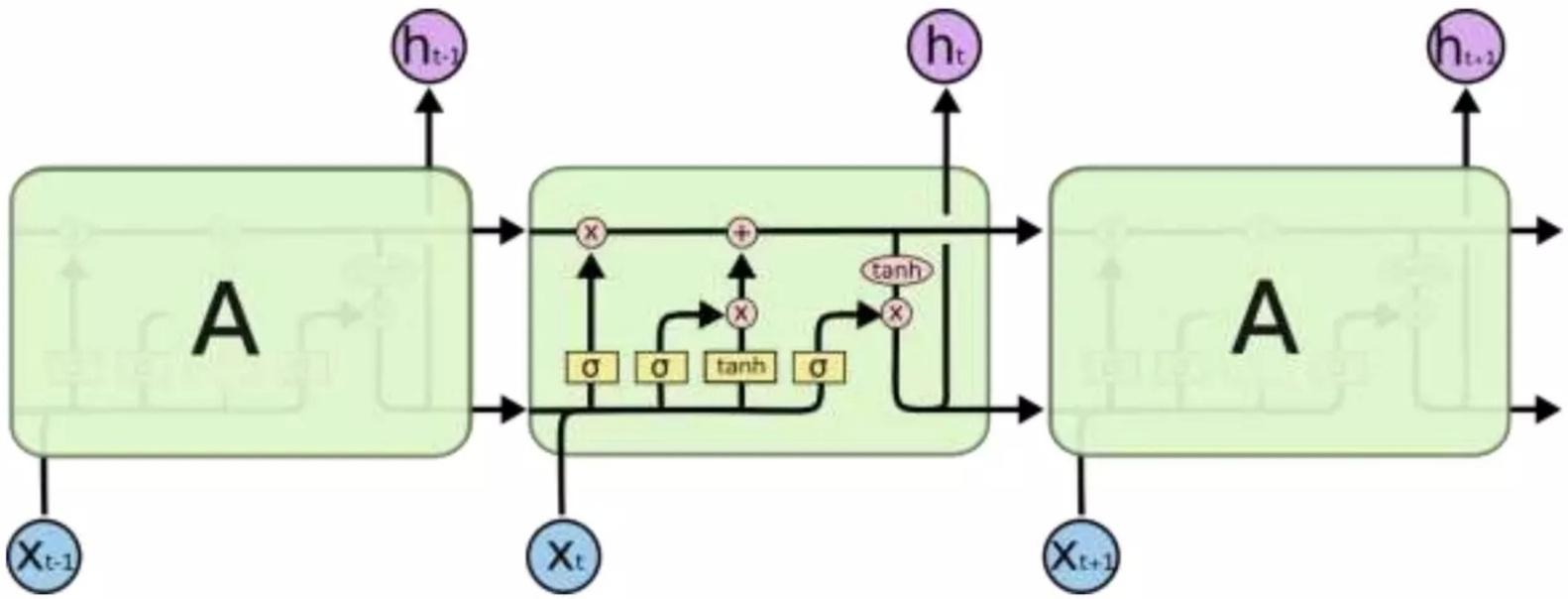


# Long Short Term Memory

- **Difference between RNN and LSTM:**
  - **RNN:** single layer (tanh)



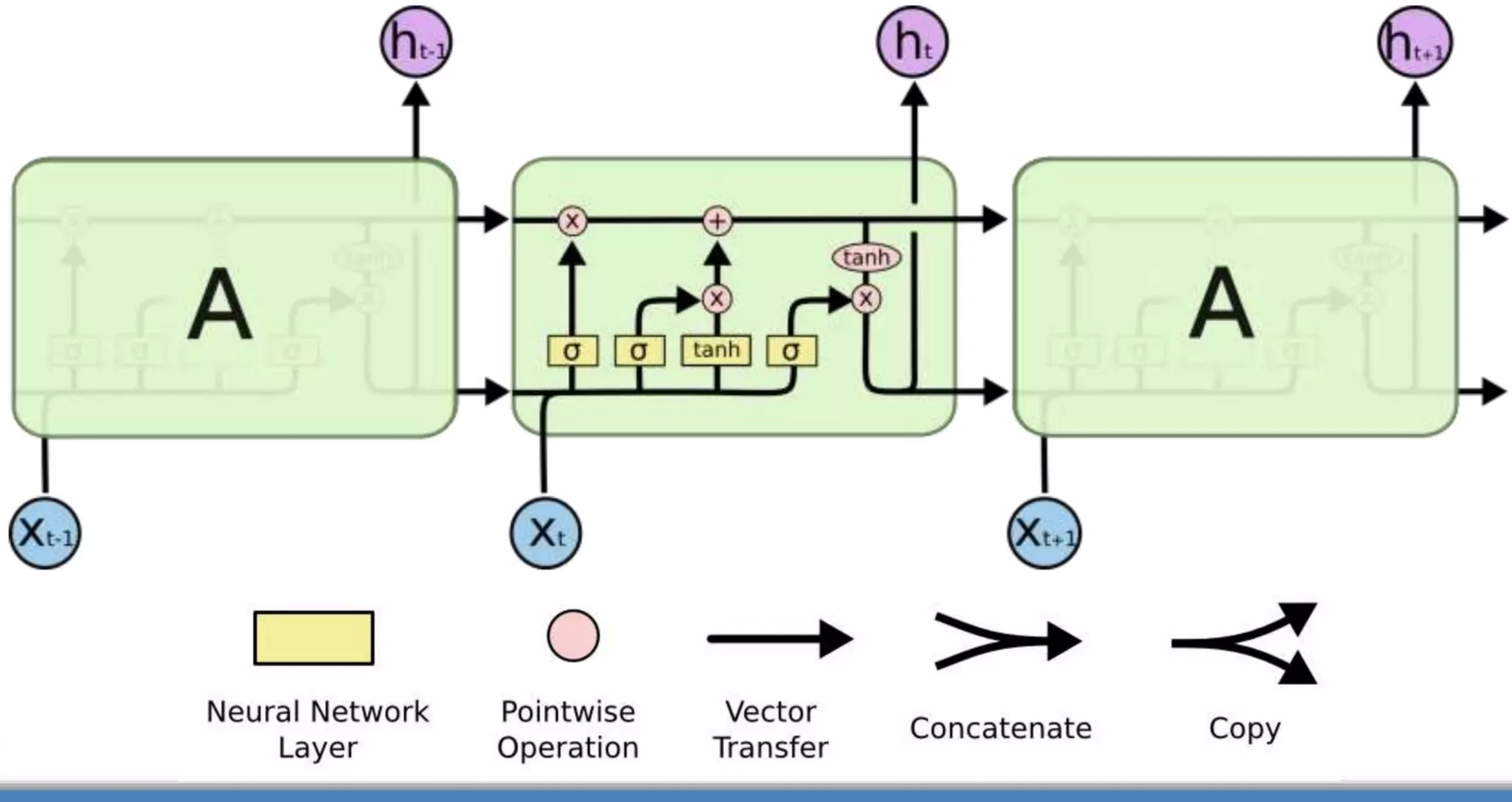

  - **LSTM:** 4 interactive layers



# Long Short Term Memory

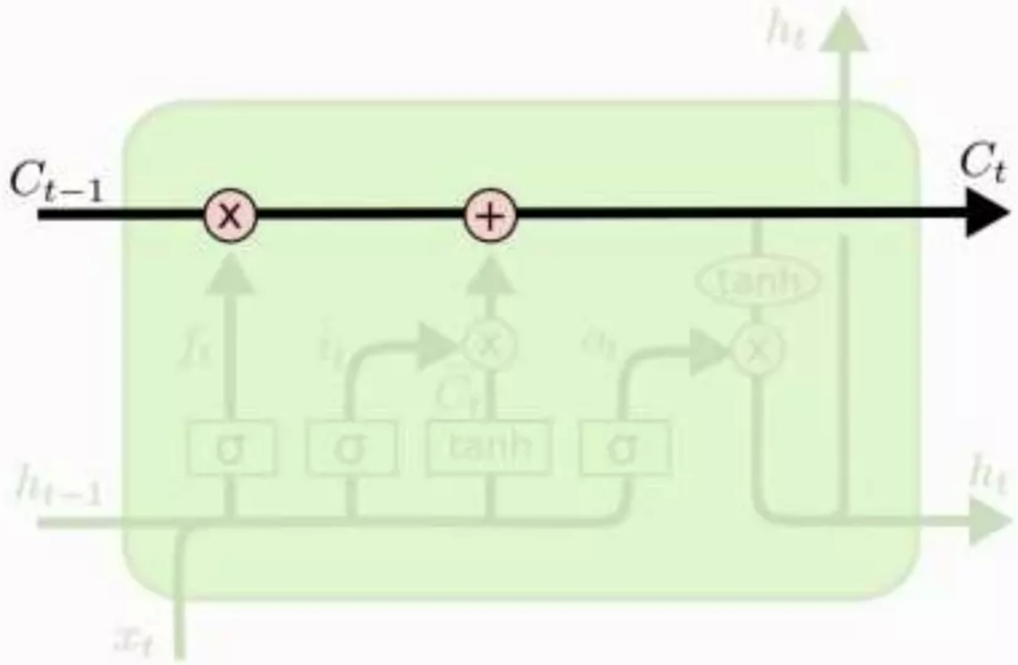
- **Vanilla LSTM:**

  Weights are all the same, only inputs change



# Long Short Term Memory

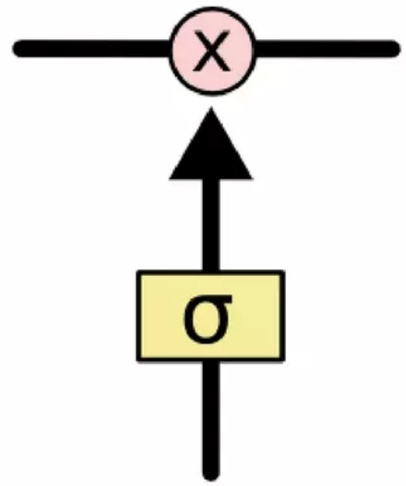
- **Cell state:**
  - Like a conveyor belt
  - It runs straight down the entire chain
  - LSTM can remove or add information to the cell state

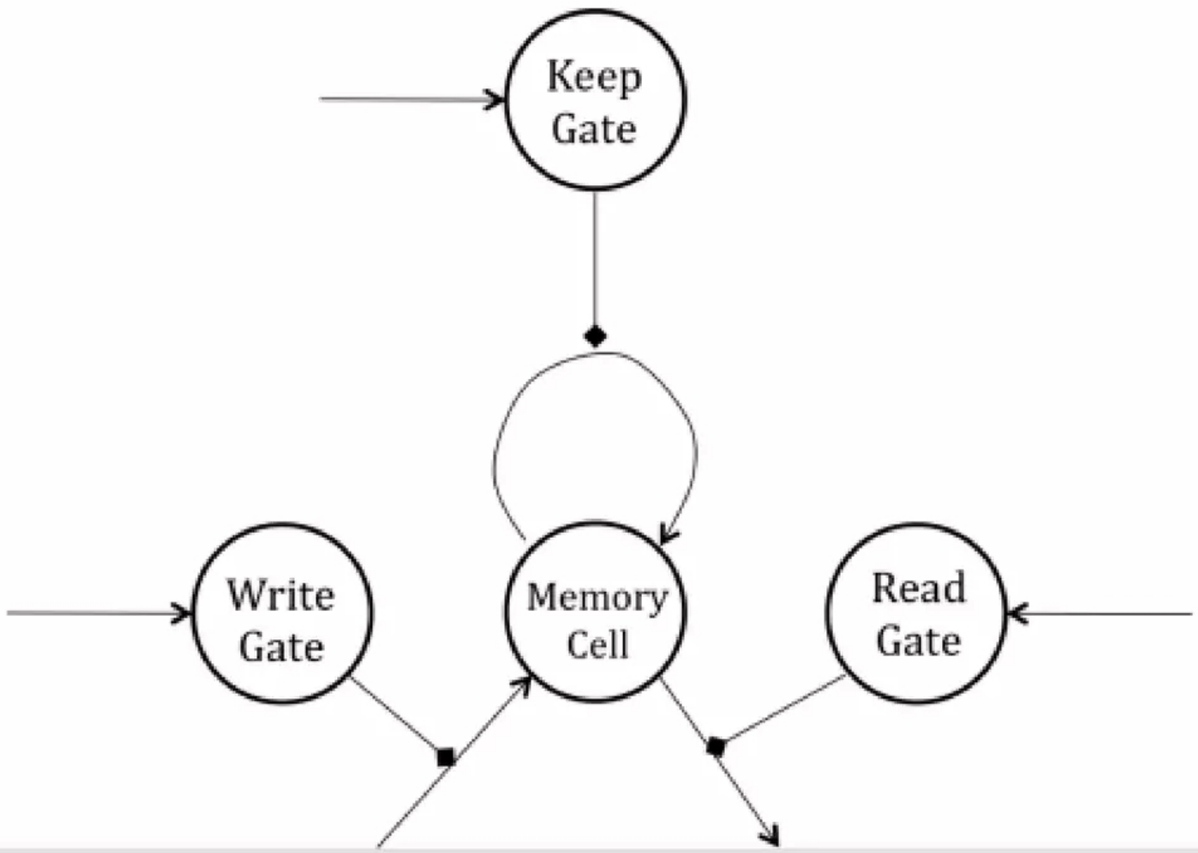



Long memory

# Long Short Term Memory

- **Gates:**
  - A way to optionally let information through
  - They are composed out of:
    - A sigmoid neural net layer
    - A pointwise multiplication operation



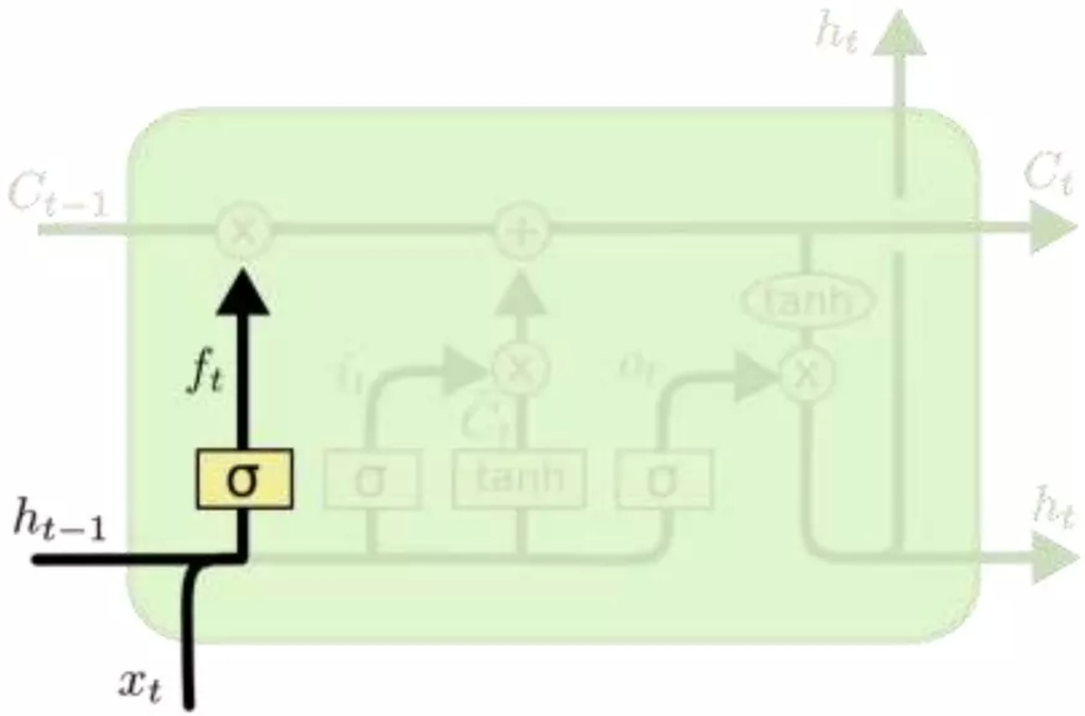



# Long Short Term Memory

- An LSTM has three of these gates, to protect and control the cell state:
  - Forget gate layer ⟹ Keep gate
  - Input gate layer ⟹ Write gate
  - Output gate layer ⟹ Read gate

# Long Short Term Memory

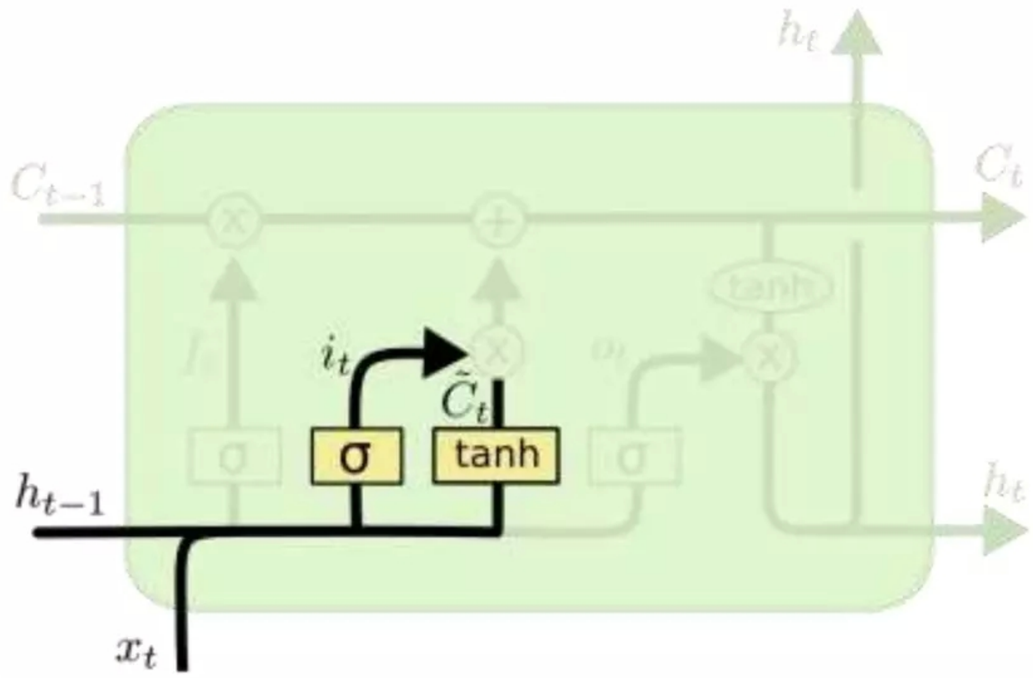
- **Forget information:**
  - Decide what information throw away from the cell state
  - **Forget gate layer:**
    - Output a number between 0 and 1




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

# Long Short Term Memory

- **Add new information:**
  - Decide what new information store in the cell state
  - **Input gate layer:**
    - Decides which values we'll update
  - **Tanh layer:**
    - creates a vector of new candidate values, $\widetilde{C}_t$

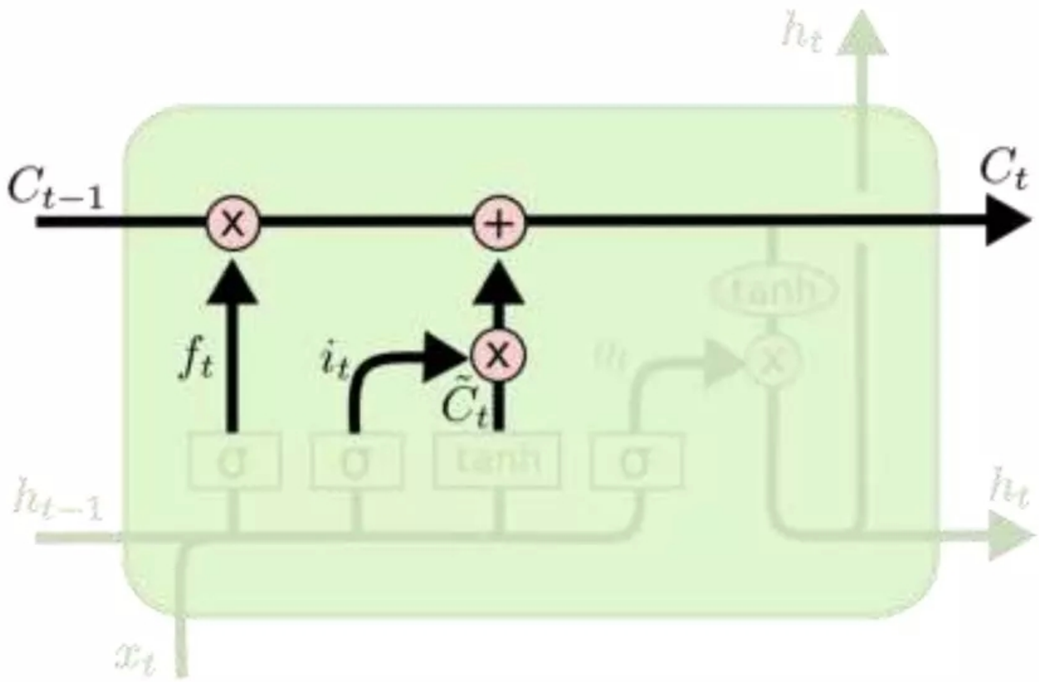



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

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

# Long Short Term Memory

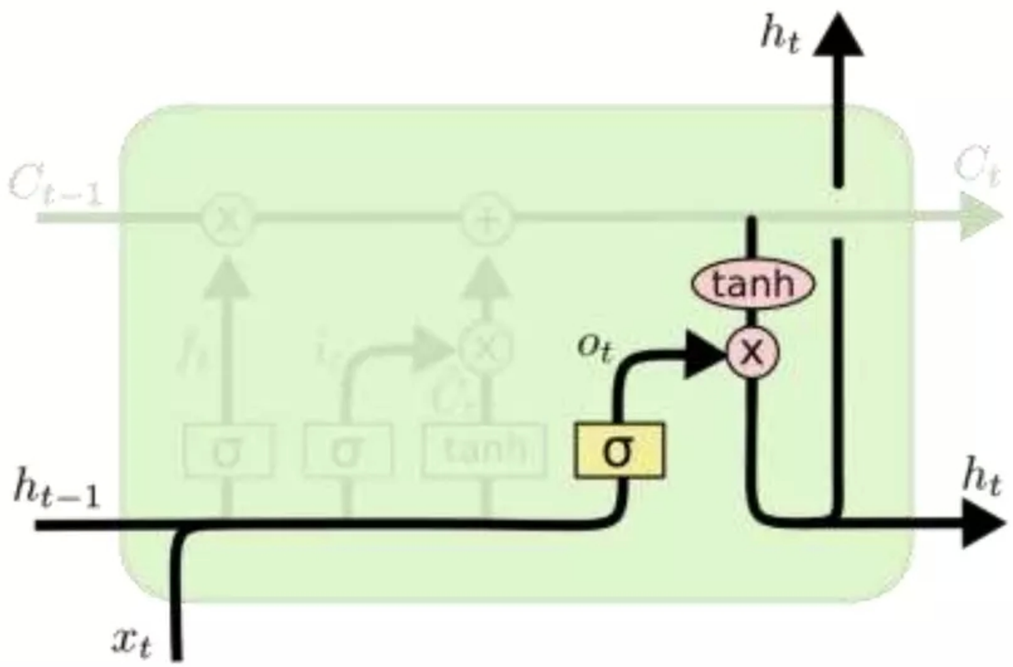
- **Update cell state:**
  - Forgetting the things we decided to forget earlier: $f_t * C_{t-1}$
  - Adding information we decide to add: $i_t * \tilde{C}_t$




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

# Long Short Term Memory

- **Create output:**
  - Decide what we're going to output
  - **Output gate layer:**
    - Decides what parts of the cell state we're going to output
  - **Tanh layer:**
    - Push the values between -1 and +1




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

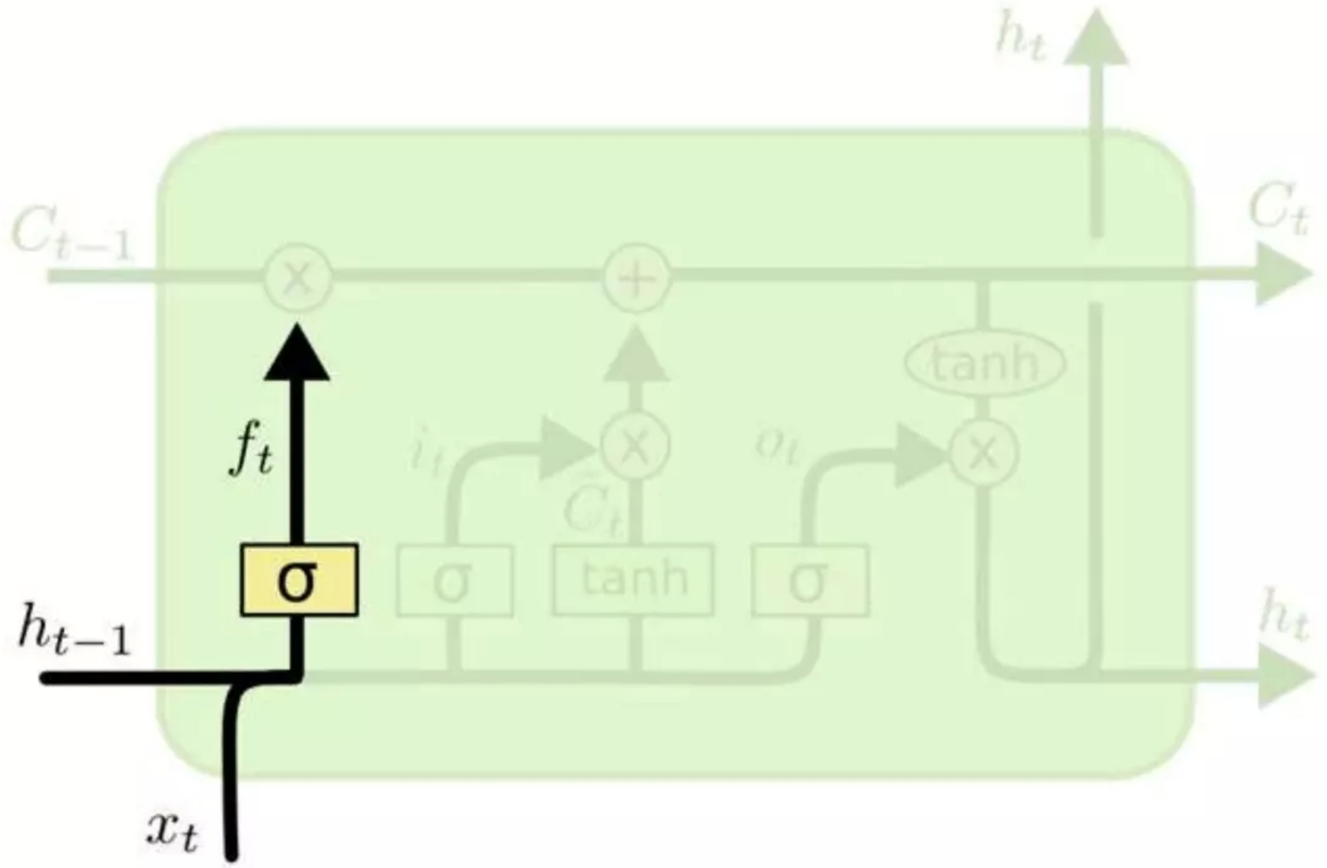
$$h_t = o_t * \tanh\left(C_t\right)$$

↓

Shadow state/ Short memory

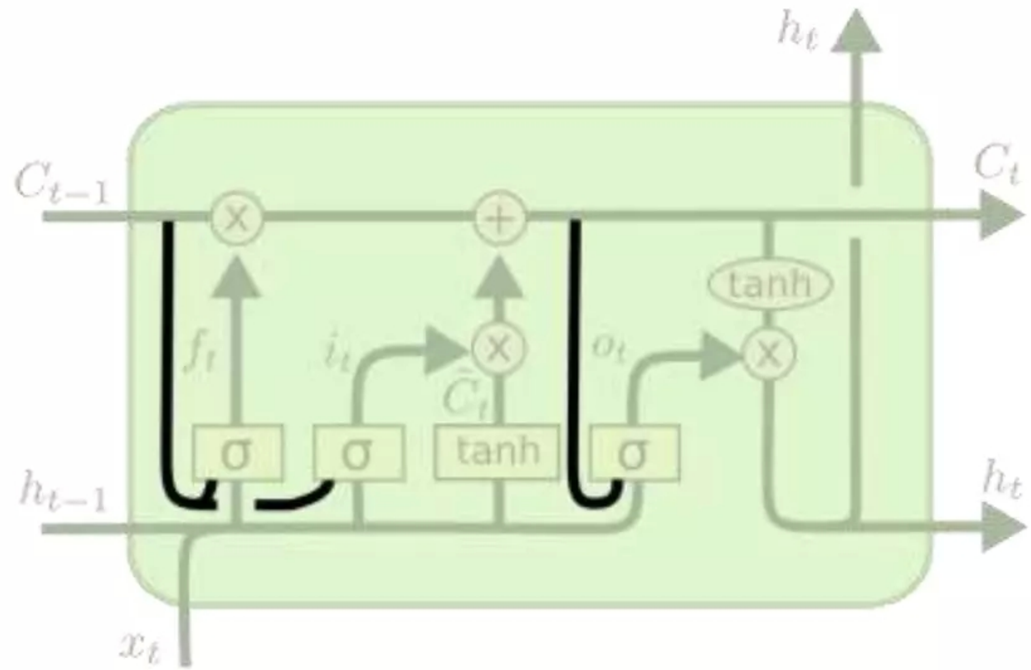
# Long Short Term Memory

- **Conclusion:**
  - **Step 1:** Forget gate layer
  - **Step 2:** Input gate layer
  - **Step 3:** Combine step 1 & 2
  - **Step 4:** Output the cell state



# LSTM Variations (1)

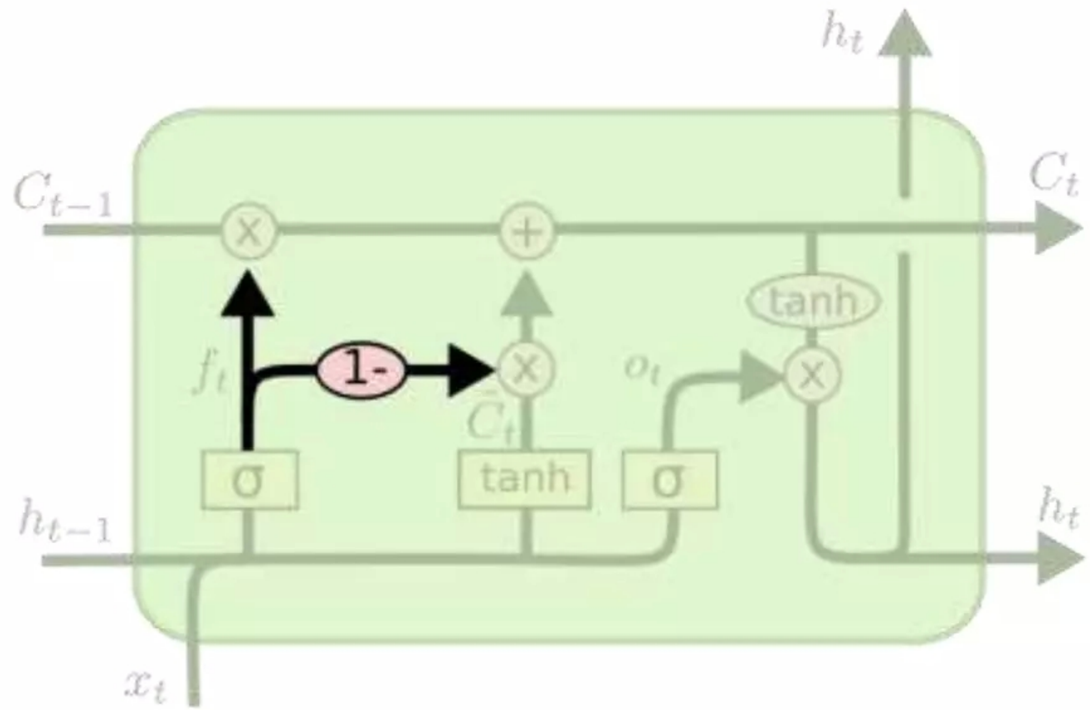
- **Peephole:**
  - Let the gate layer look at the cell state (entire/ partial)




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

# LSTM Variations (2)

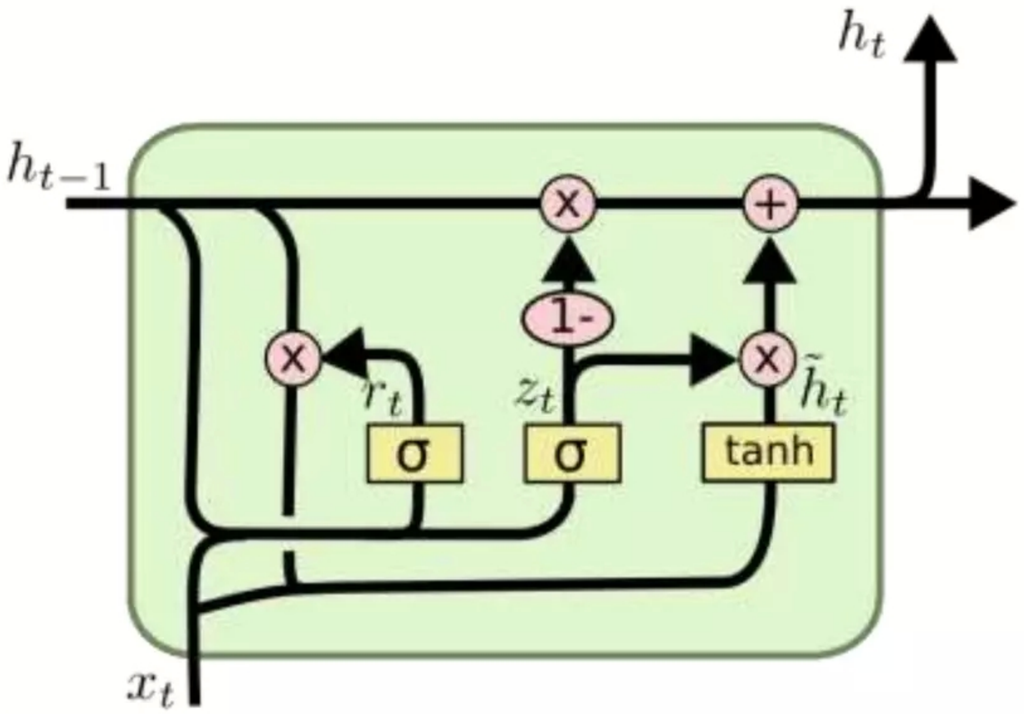
- **Coupled forgot and input gates:**
  - Not deciding separately




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

# LSTM Variations (3)

- **Gated Recurrent Unit (GRU):**
  - Combine the forget and input layer into a single “update gate”
  - Merge the cell state and the hidden state
  - Simpler and popular




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

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

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

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

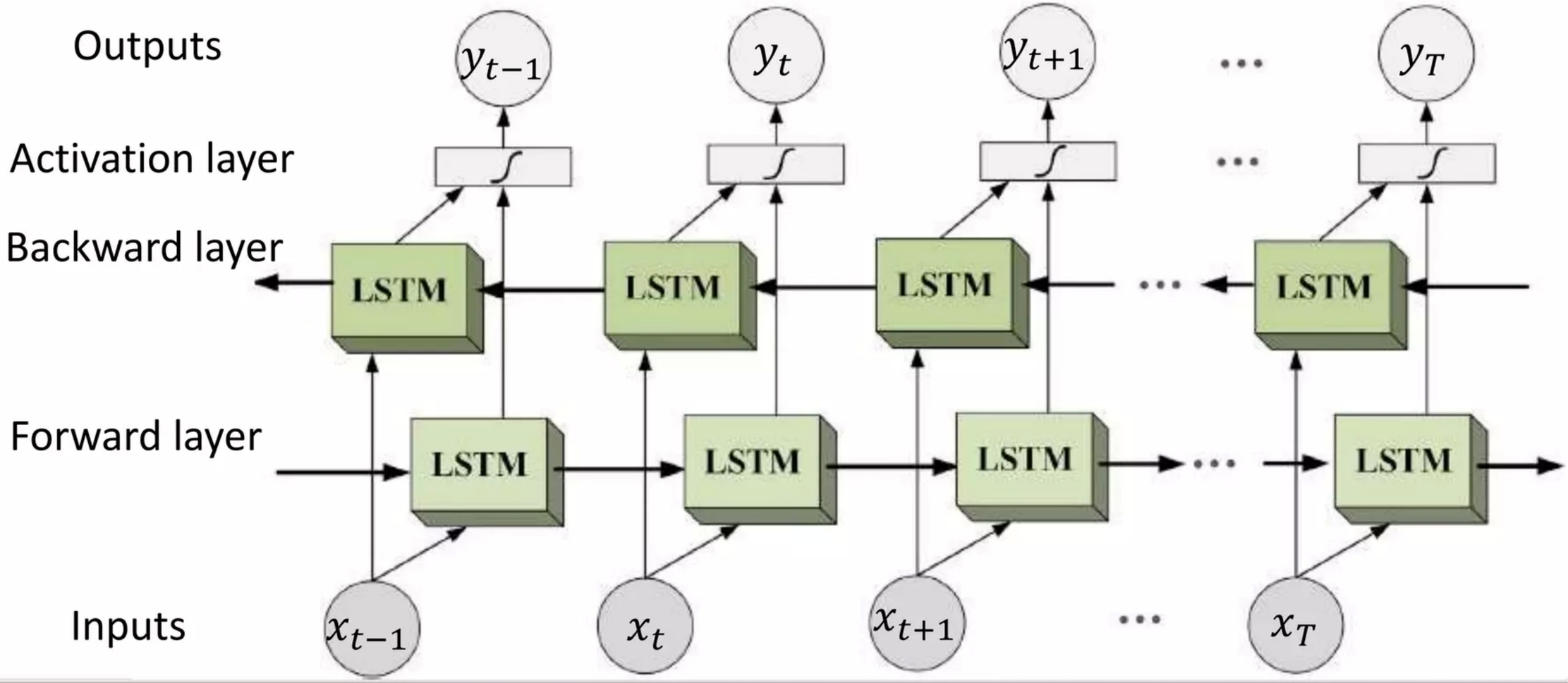
# LSTM Variations Comparison

- They're all about the same in performance
- We can reduce the number of parameters and the computational cost by:
  - Coupling the input and forget gates (GRU, Variation #2)
  - Removing peephole connections (Vanilla LSTM)

Greff, K., et al. (2017). LSTM: A search space odyssey. *IEEE transactions on neural networks and learning systems*.

# Bidirectional LSTM

- Training Information travels in both forward and backward directions
- Remembers complex long term dependencies better.
- **Using BiLSTM:**



# Thank you