# Knowledge Discovery & Data Mining
# — Recurrent Neural Networks —

**Instructor: Yong Zhuang**

*yong.zhuang@gvsu.edu*

Based on the original version at https://colah.github.io/posts/2015-08-Understanding-LSTMs/

# Outline

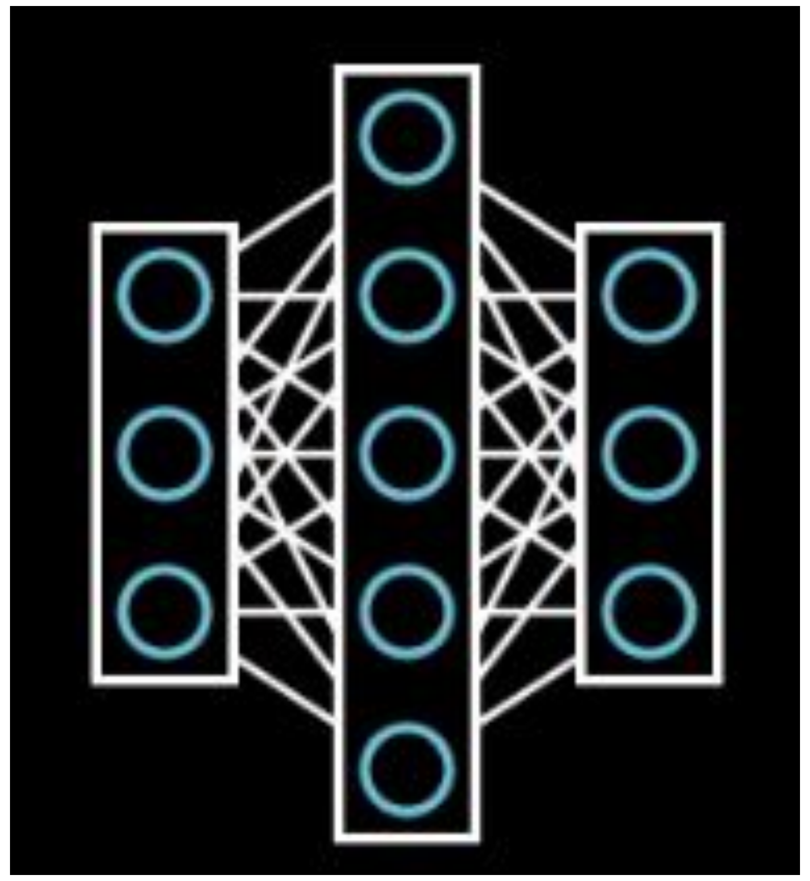
- Recurrent Neural Networks
  - Why do we need RNN?
  - RNNs:
    - Standard RNN
    - LSTM
    - GRU

# What are Sequences?

- Definition: is a stream of data (finite or infinite) which are interdependent.
- Examples: Time series data, strings of text, conversations.
- Importance: In sequences, individual elements have meaning, but the overall sequence provides context (e.g., a day's worth of stock market data).

# Limitations of Traditional Neural Networks

- Traditional neural networks process inputs independently, without considering the sequence or order.
- Lack of correlation between previous and subsequent inputs.
- Inefficient in capturing the context or sequence patterns in data (e.g., in a conversation or time series).



# Why RNNs?

- Memory Capability: RNNs have a form of 'memory' about previous inputs in a sequence.
- Context Awareness: They excel in tasks where context from earlier data is crucial.
- Just as humans base decisions on prior knowledge and context, RNNs use previous inputs to inform current outputs.

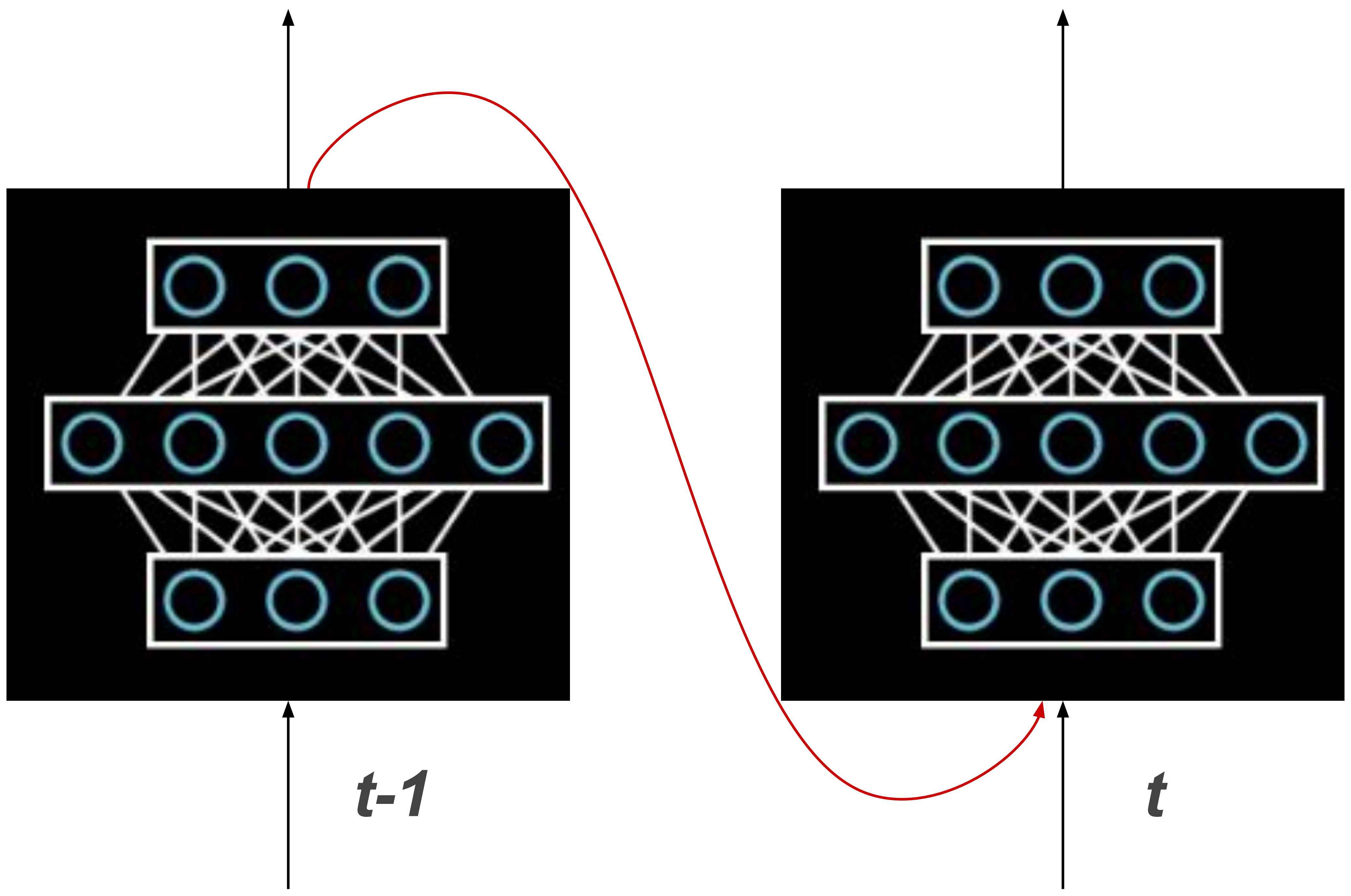
# Applications of RNNs

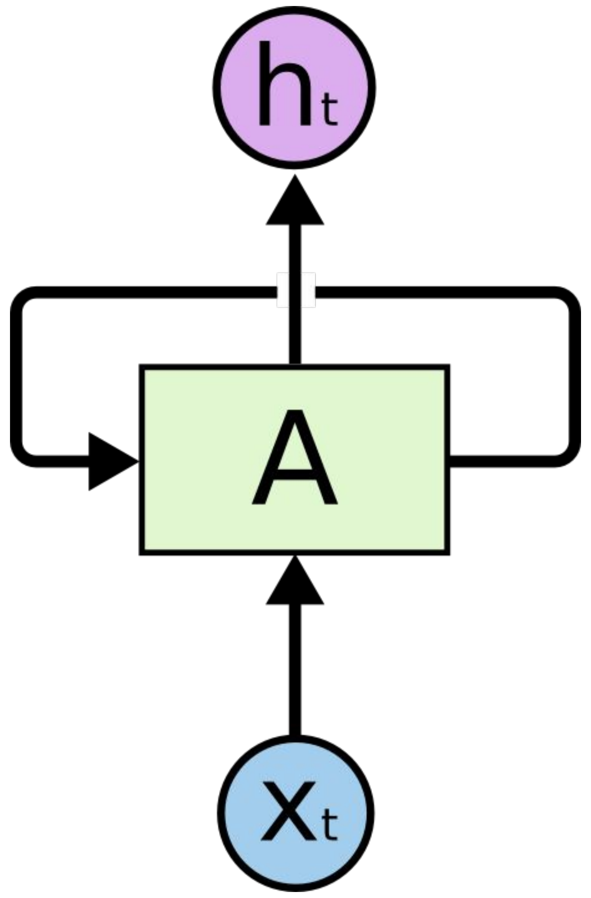
- Language Modeling and Text Generation: Predicting the likelihood of the next word in a sequence.
  - Use Case: Useful in auto-completion, content creation, and more.
- Machine Translation: Translating text from one language to another.
  - Implementation: Advanced RNN models are widely used in real-world translation systems.
- Speech Recognition: Converting spoken words into text by predicting phonetic segments.
  - Application: Used in voice assistants, dictation software, etc.
- Generating Image Descriptions: Combination of CNN (Convolutional Neural Network) for image segmentation and RNN for generating descriptions.
  - Use Case: Understanding and describing the content of images.
- Video Tagging: Frame-by-frame image description for videos.
  - Application: Facilitates video search and content analysis.

*RNNs are fundamental in processing and generating sequential data, offering remarkable capabilities in diverse fields.*

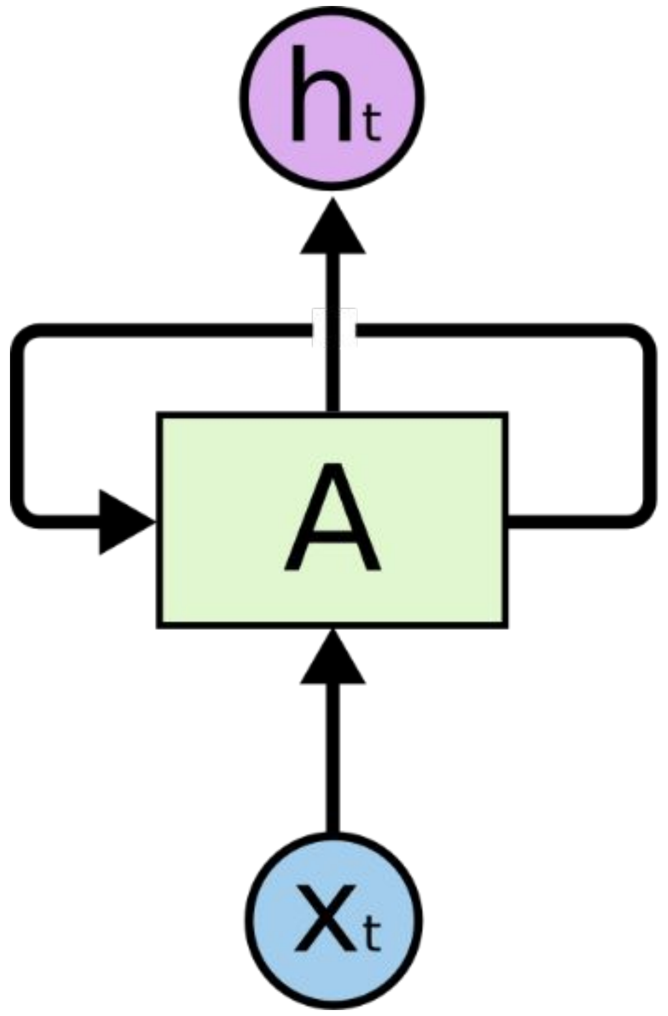
# Recurrent Neural Networks (RNN)

Recurrent neural networks are a type of neural network where the outputs from previous time steps are fed as input to the current time step.

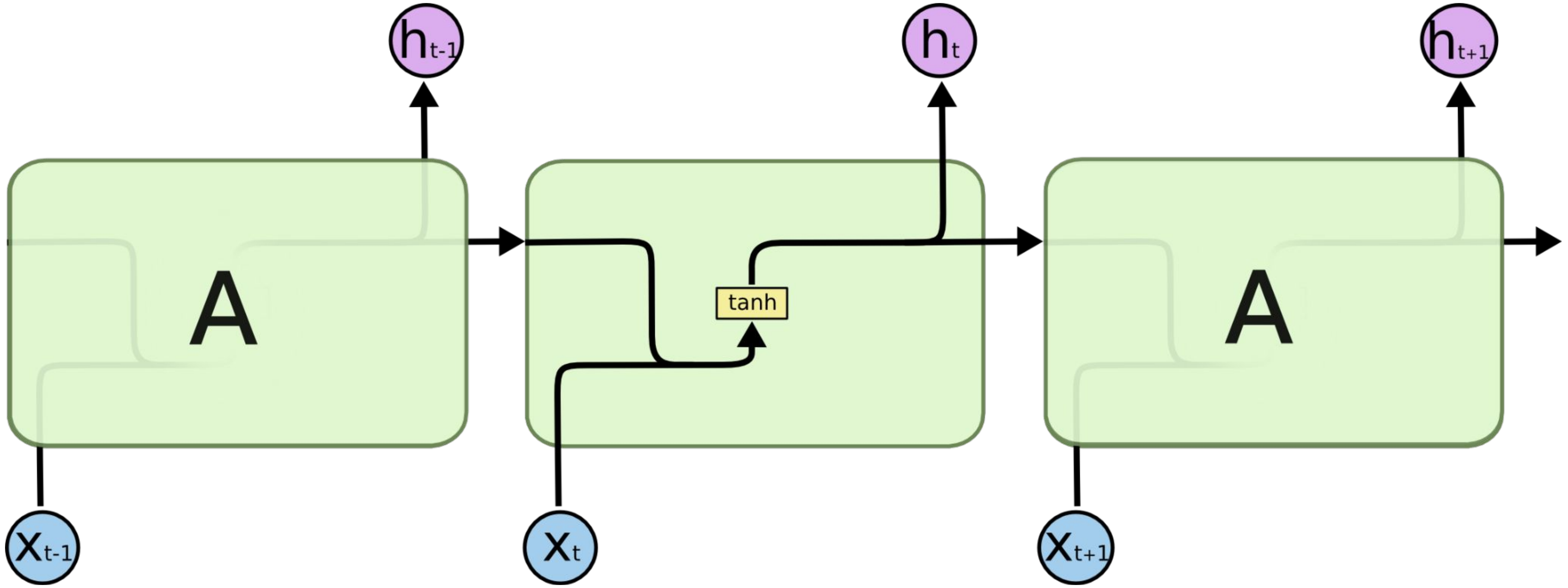


# An unrolled recurrent neural network.



# Standard RNN




$$h_t = \tanh(W \cdot [h_{t-1}, x_t] + b)$$

where $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

**Activation Function**

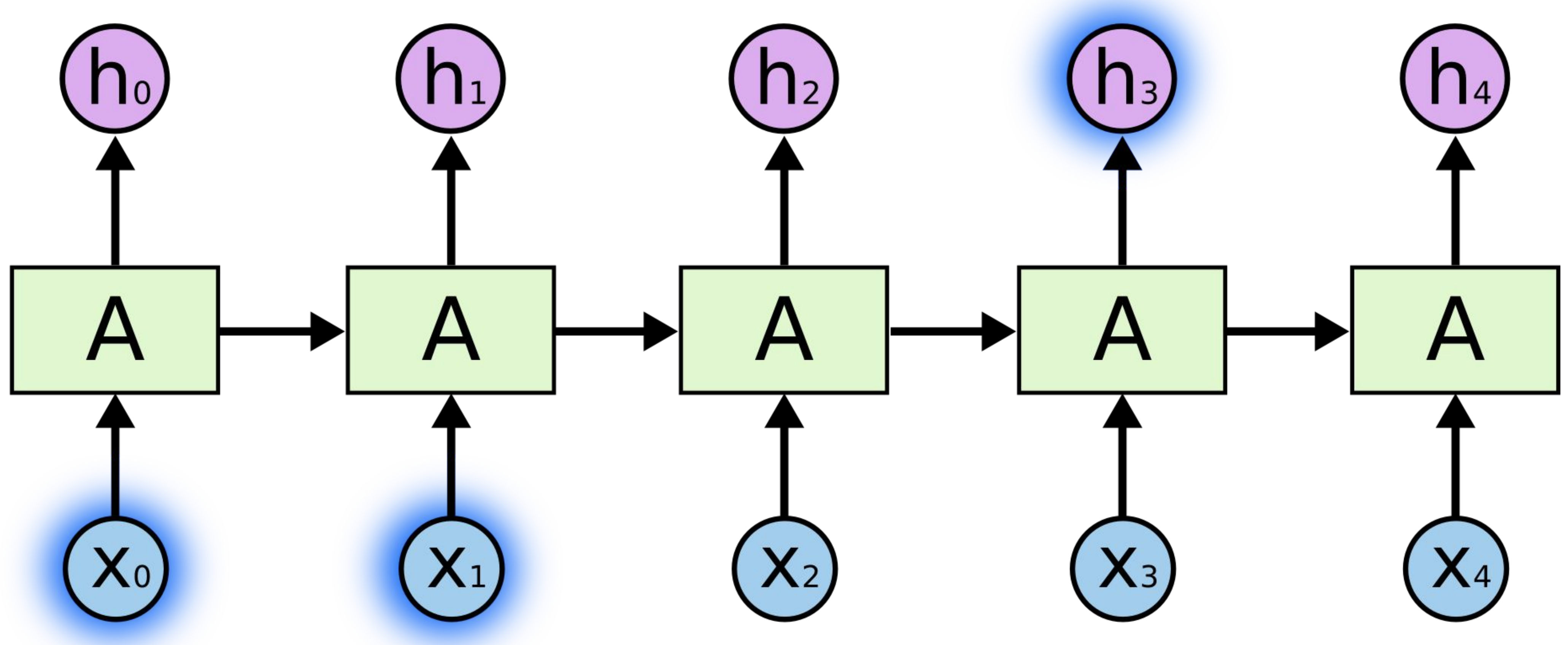
more activation functions

# Short-Term Dependencies in RNNs

RNNs perform well when only recent information is needed.

Example in Language Modeling:

- Predicting the next word in a sentence.
- Sentence: “The clouds are in the ***sky***...”
- The next word is easily predicted without needing much context.



# The Challenge of Long-Term Dependencies

Long-term dependencies remain a significant challenge for Standard RNNs, affecting their effectiveness in more complex scenarios.

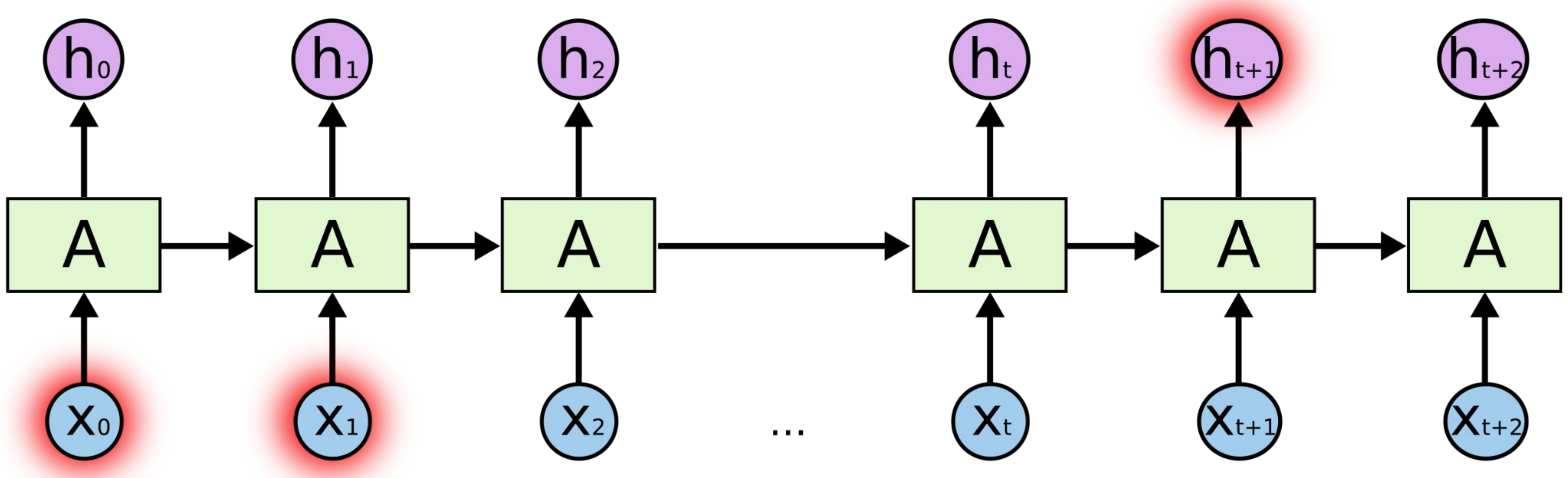
Example in Language Modeling:

- Sentence: Predicting the last word in “I grew up in ***France***… I speak fluent ____.”
- Challenge: The necessity to recall 'France' to accurately predict 'French' highlights the need for long-term context.

# The Challenge of Long-Term Dependencies

Predicting the last word in “I grew up in ***France***… I speak fluent ____.””

- Problem: As the gap between relevant information and its usage grows, RNNs struggle to make the connection. Difficulty in connecting '***France***' mentioned earlier in a sentence to predict 'French' much later.

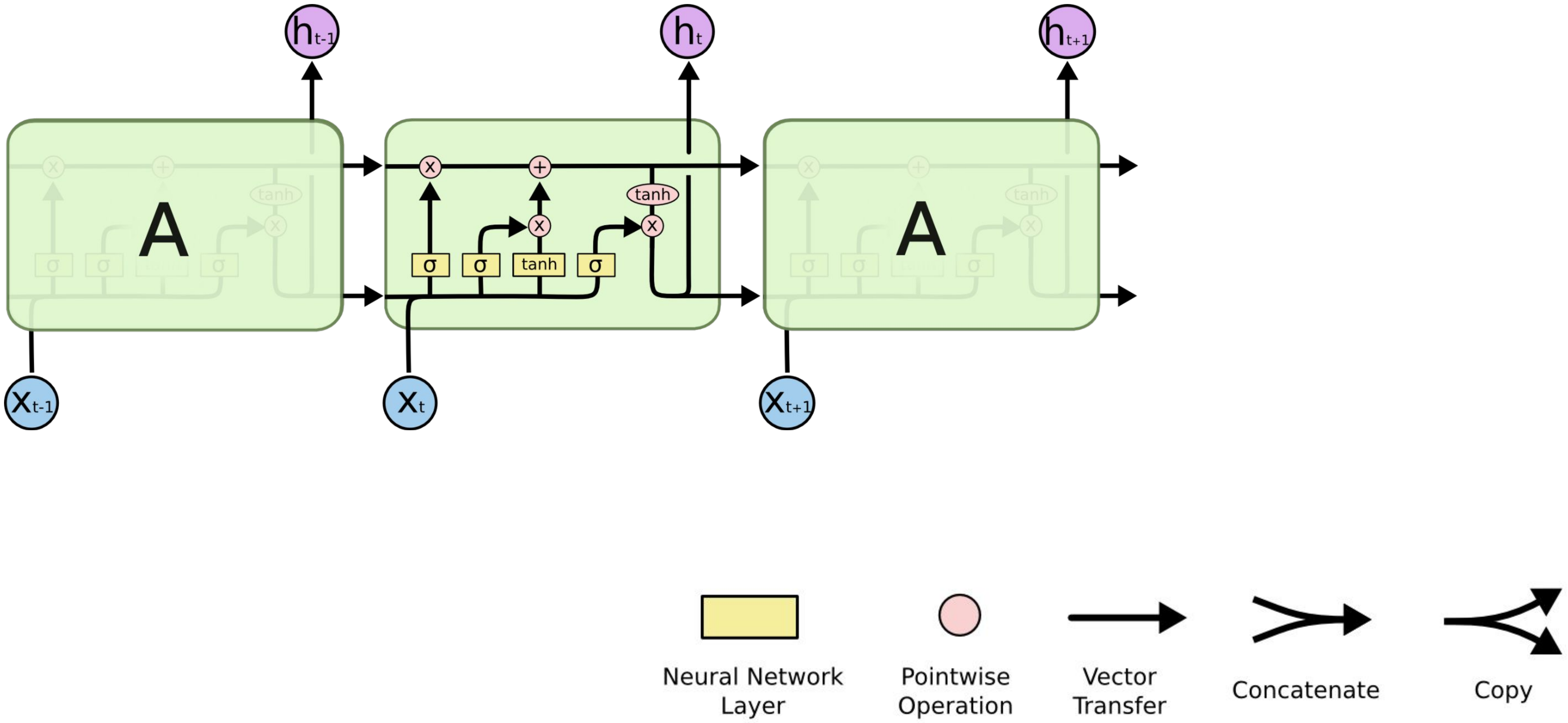


# Long Short Term Memory Networks (LSTMs)

- A special type of RNN designed to learn long-term dependencies.
- Origin: Introduced by Hochreiter & Schmidhuber in 1997 and refined in subsequent research.
- Learning Long-Term Dependencies: LSTMs are adept at handling tasks where it's crucial to remember information over extended periods.
- Wide Applicability: Proven effective across a diverse range of problems.

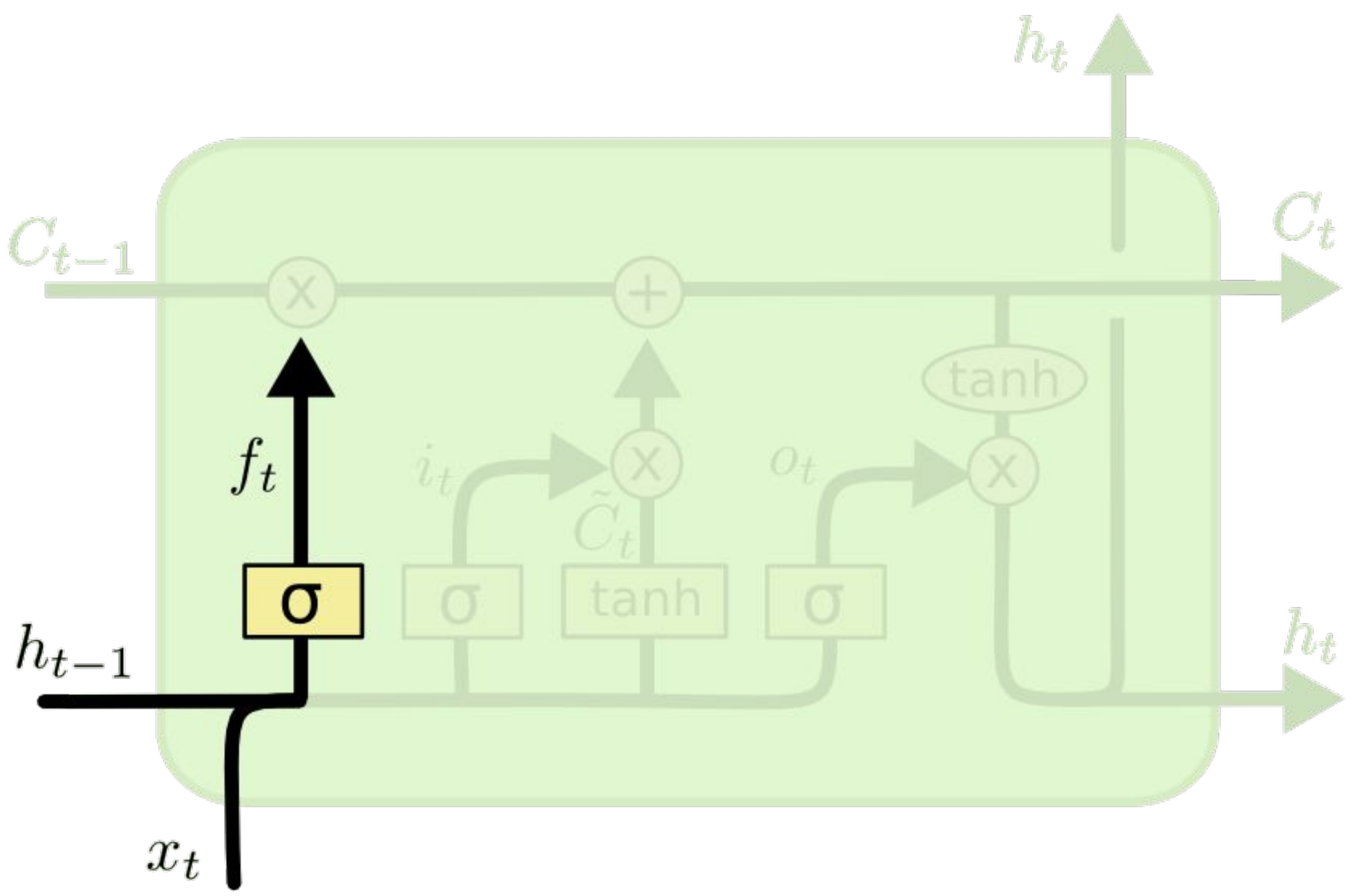
*LSTMs represent a significant advancement in neural networks, offering a robust solution to the limitations of traditional RNNs in processing sequential data with long-term dependencies.*

# Four interactive layers of LSTM units



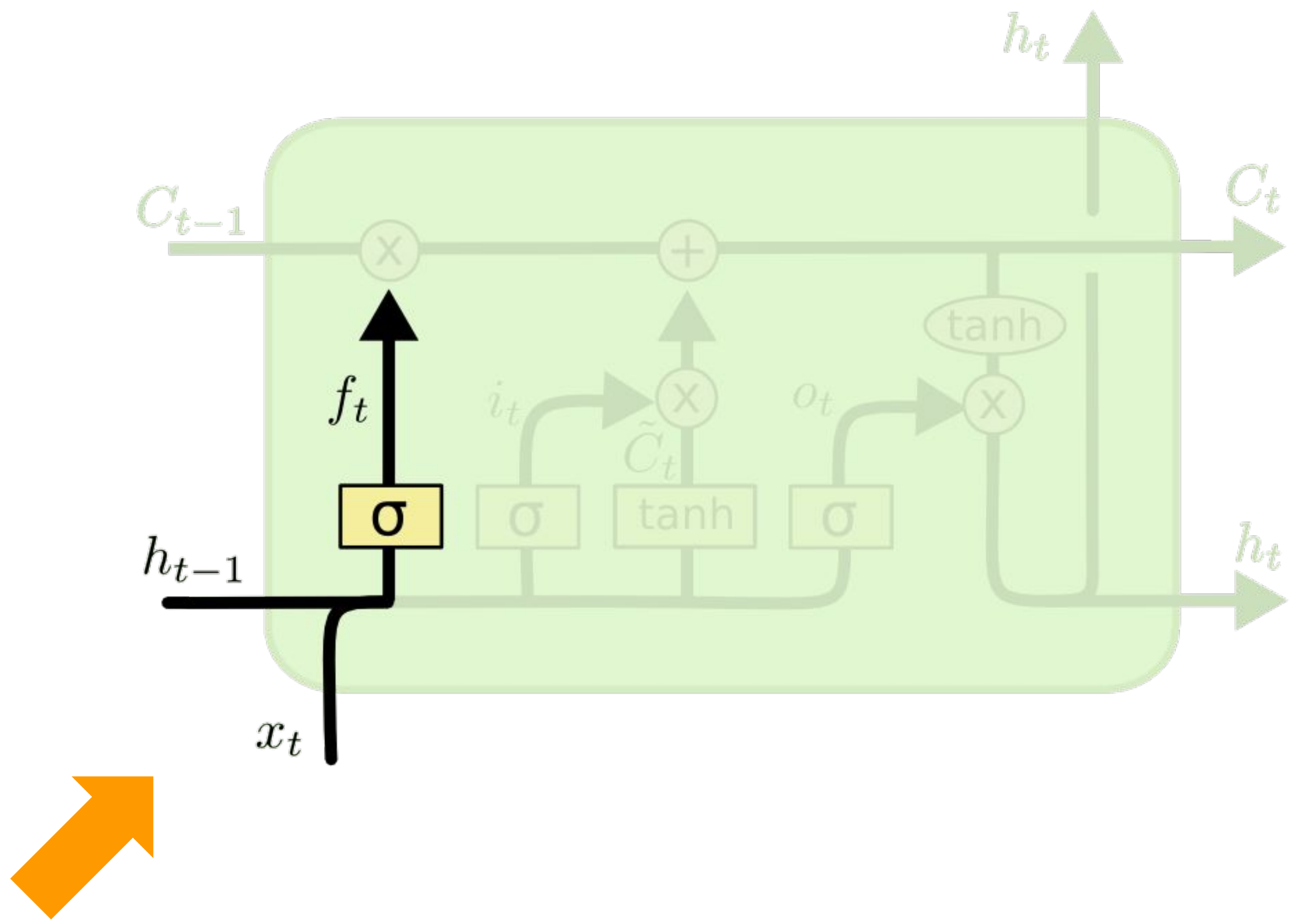
# Forget gate layer

- Primary Role: Determines what information to discard from the cell state.
- Example in Language Modeling:
  - Sentence: "***He*** is French. ***She*** is…"
  - On encountering "***She***", the forget gate decides to discard the information related to "***He***" to update the subject context.



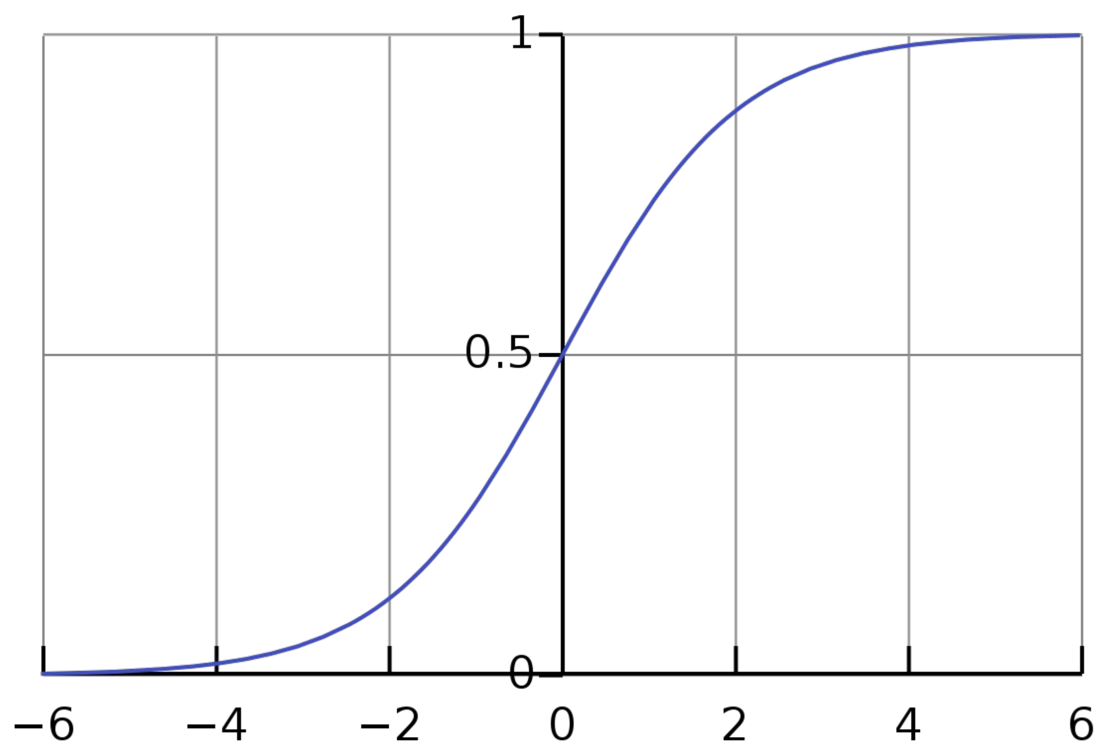

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

# Forget gate layer




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

***sigmoid function***

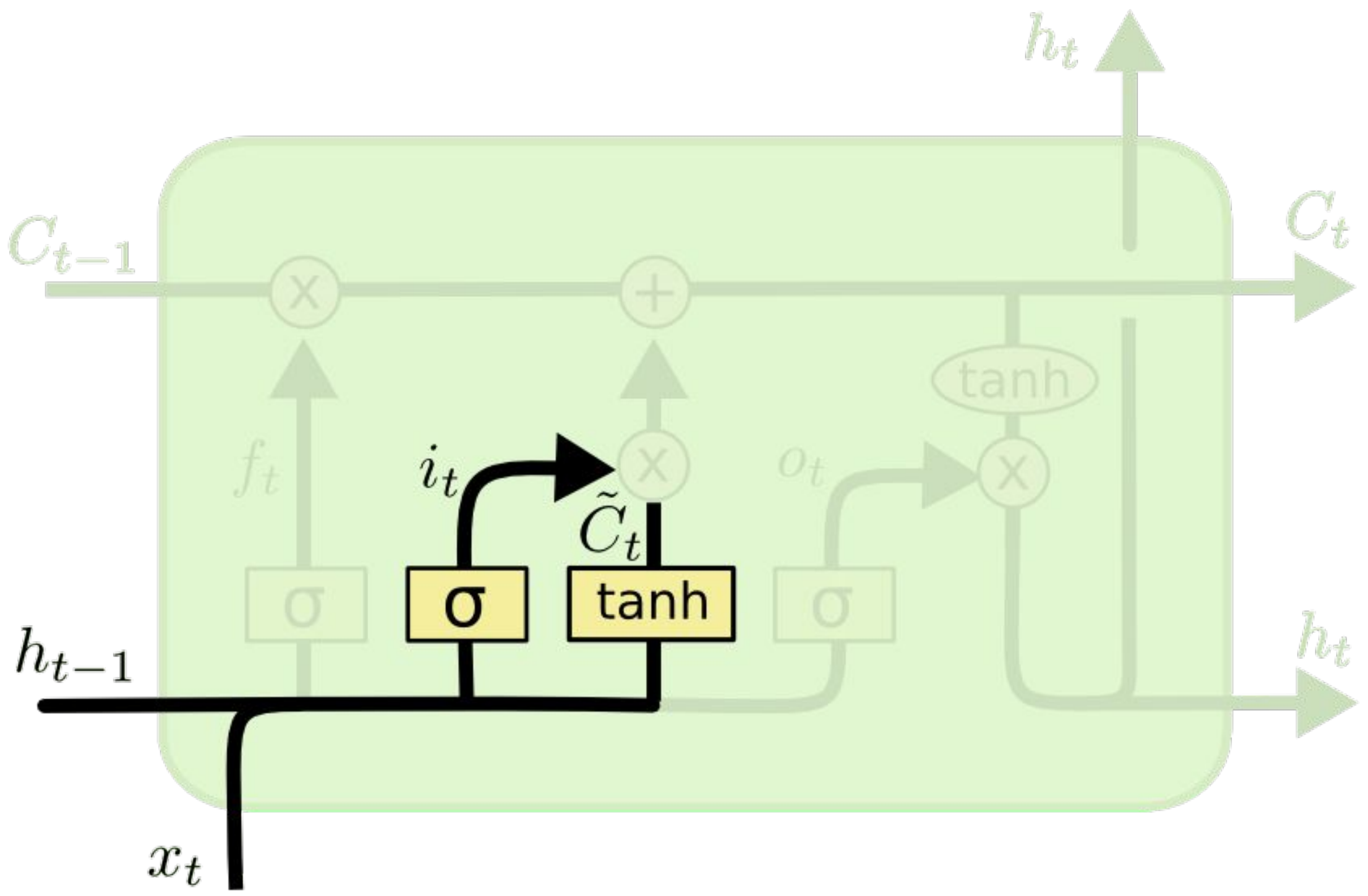



- **Inputs**: Takes in $h_{t-1}$ (previous hidden state) and $x_t$ (current input).

- **Output**: Generates values between 0 and 1 for each number in the cell state $C_{t-1}$.
  - **1**: Retain the information completely.
  - **0**: Completely discard the information.

# Input gate layer

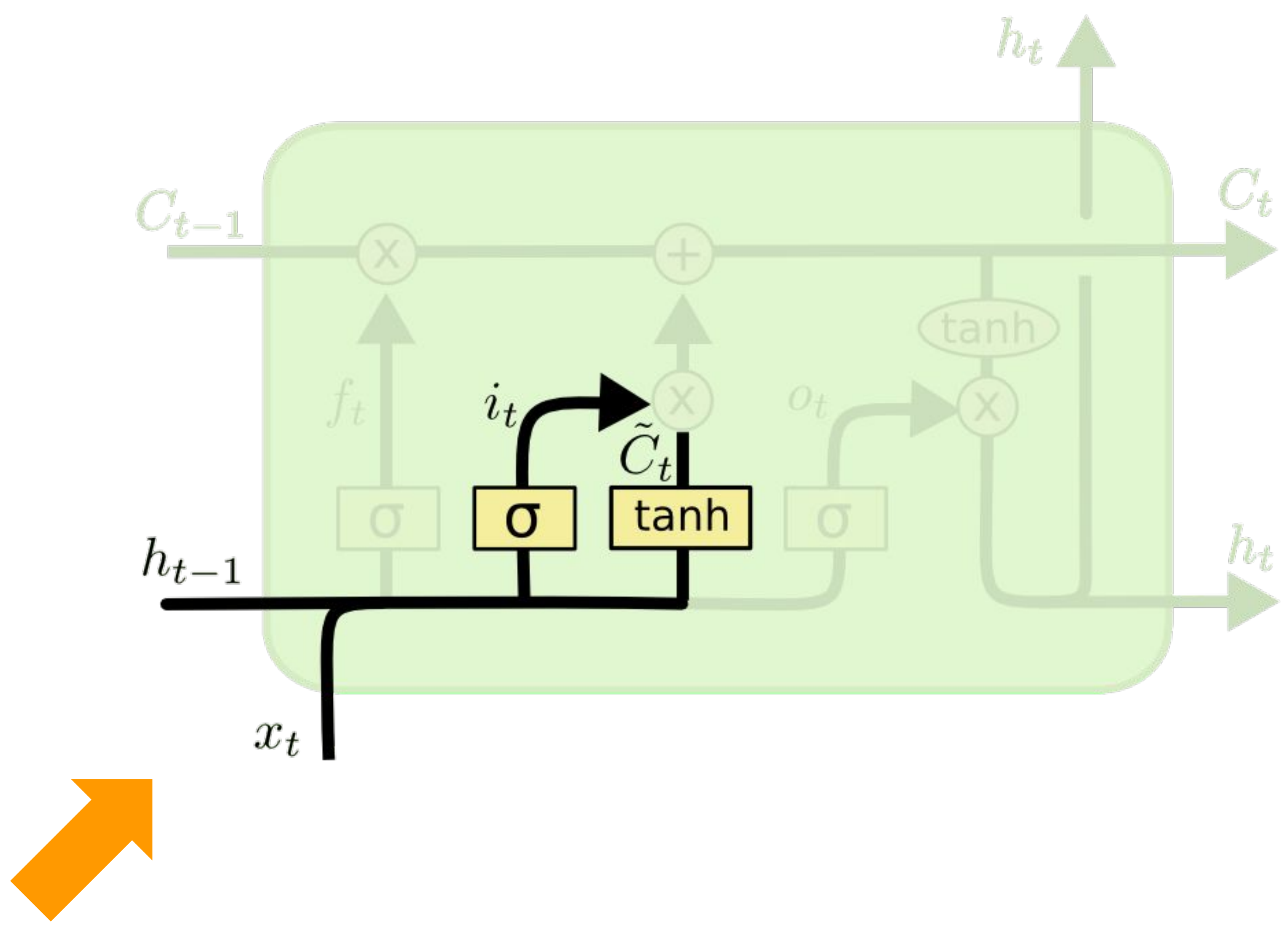
- Primary Role: Determines which values in the cell state to update.
- Example in Language Modeling:
  - Sentence: "***He*** is French. ***She*** is…"
  - The model decides to add "***She***" to the cell state, replacing the information about "***He***" that was previously forgotten.




$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

# Input gate layer

*to decide the extent of updating each value.*




***sigmoid function***

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

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

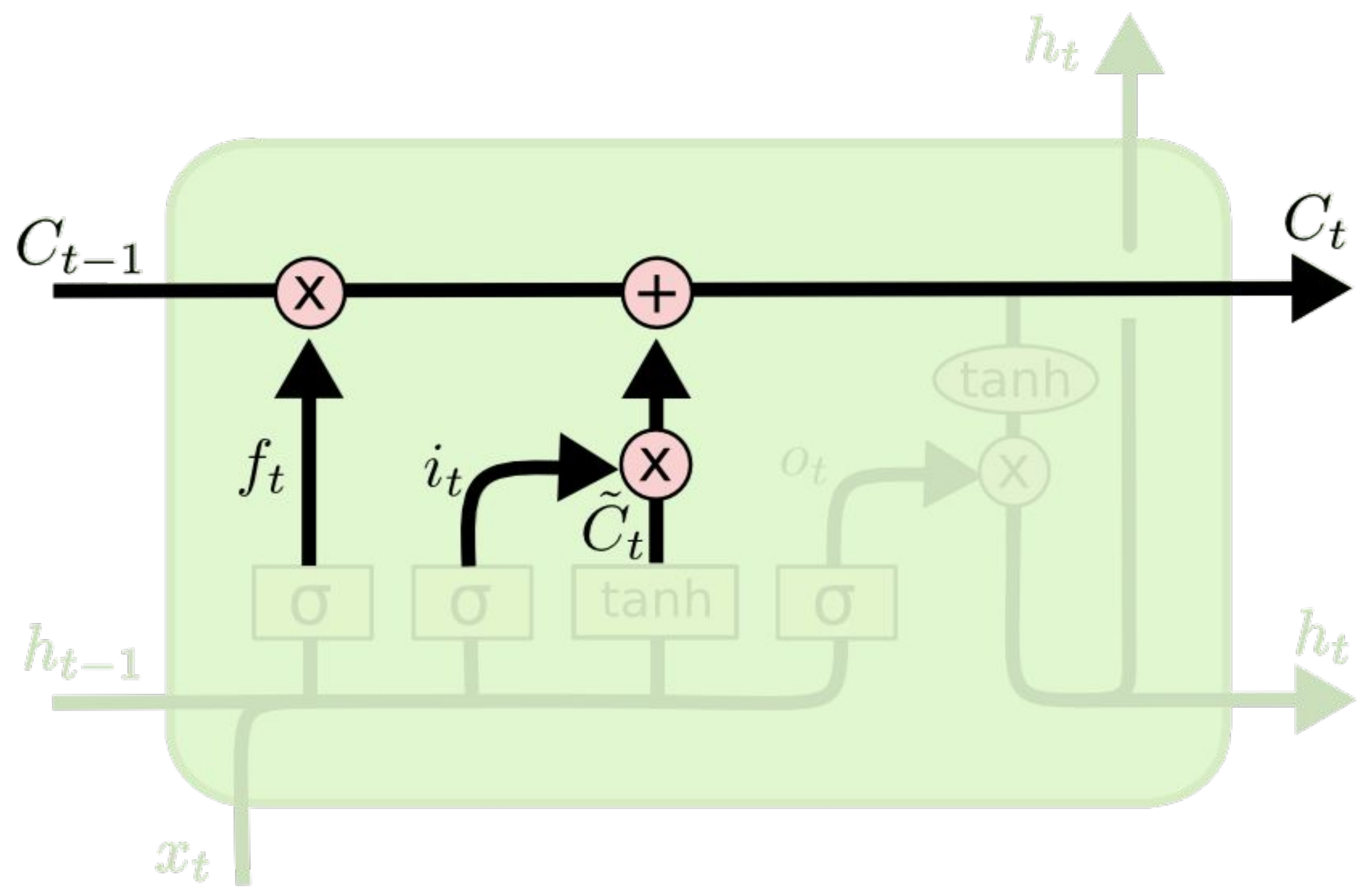
***tanh function***

- **Inputs**: Takes in $h_{t-1}$ (previous hidden state) and $x_t$ (current input).

*creates a vector of potential new values.*

# Updating Memory

- Example in Language Modeling:
  - Sentence: "***He*** is French. ***She*** is…"
  - Forgetting: The information about "***He***" is dropped.
  - Updating: New information "***She***" is added to the cell state.

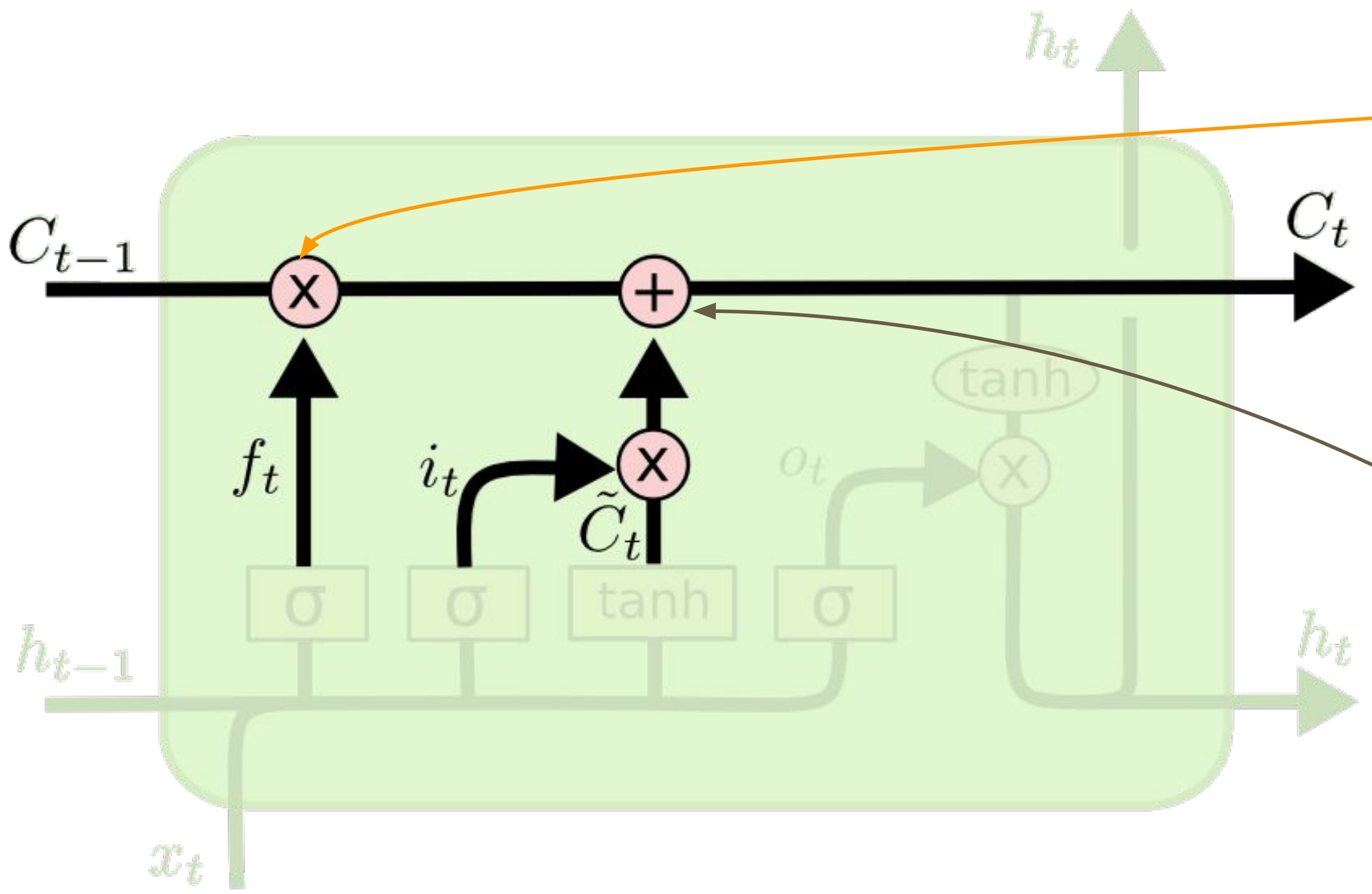



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

# Updating Memory

Forgetting: The old state is modified to forget certain details.

*Multiply the old cell state by the forget gate's output.*



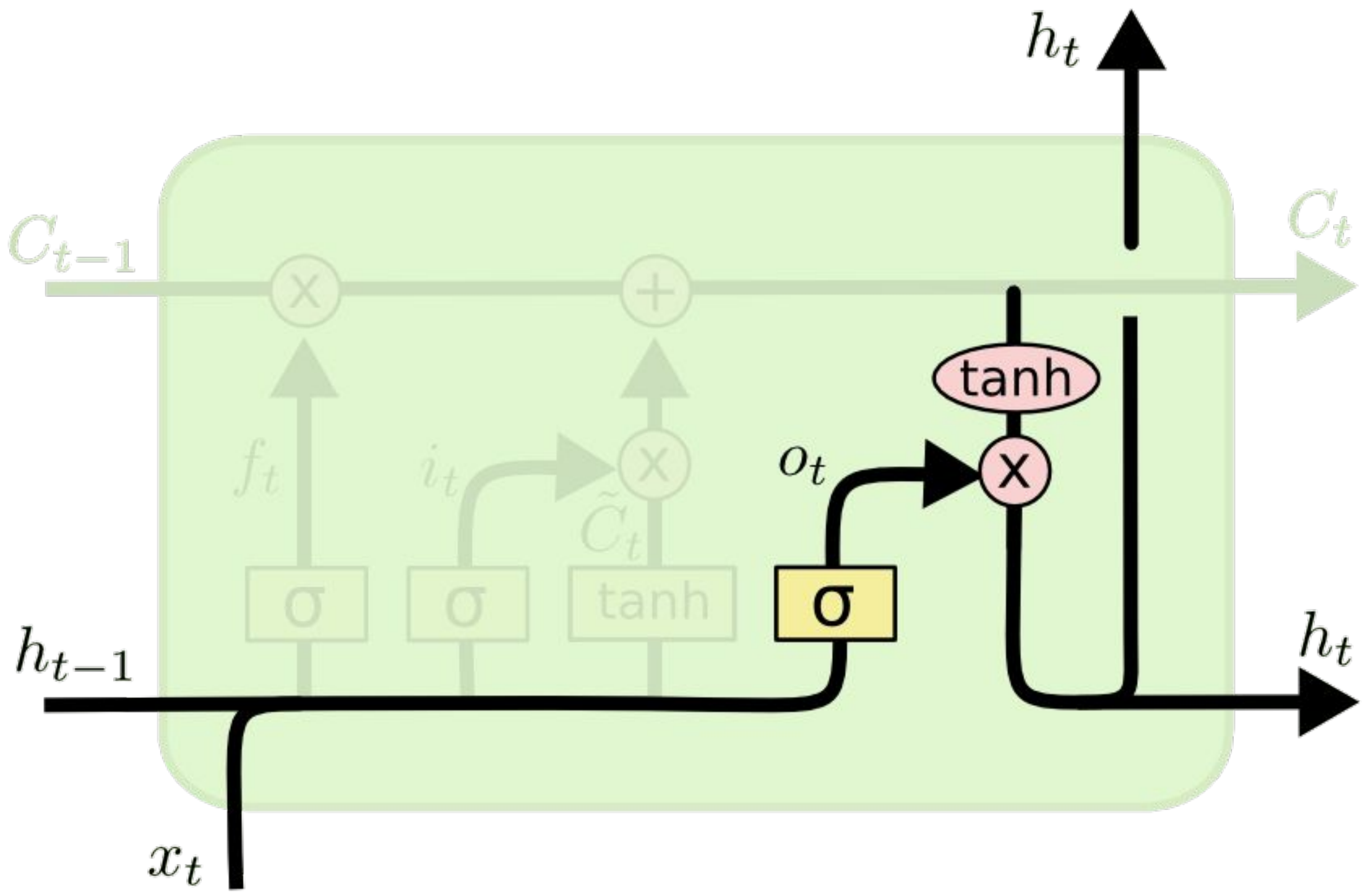

$$C_t = \overbrace{f_t * C_{t-1}}^{\textbf{forgetting}} + \underbrace{i_t * \tilde{C}_t}_{\textbf{updating}}$$

*Add scaled new candidate values (from the input gate).*

Updating: New candidate values are introduced to the state.

# Output

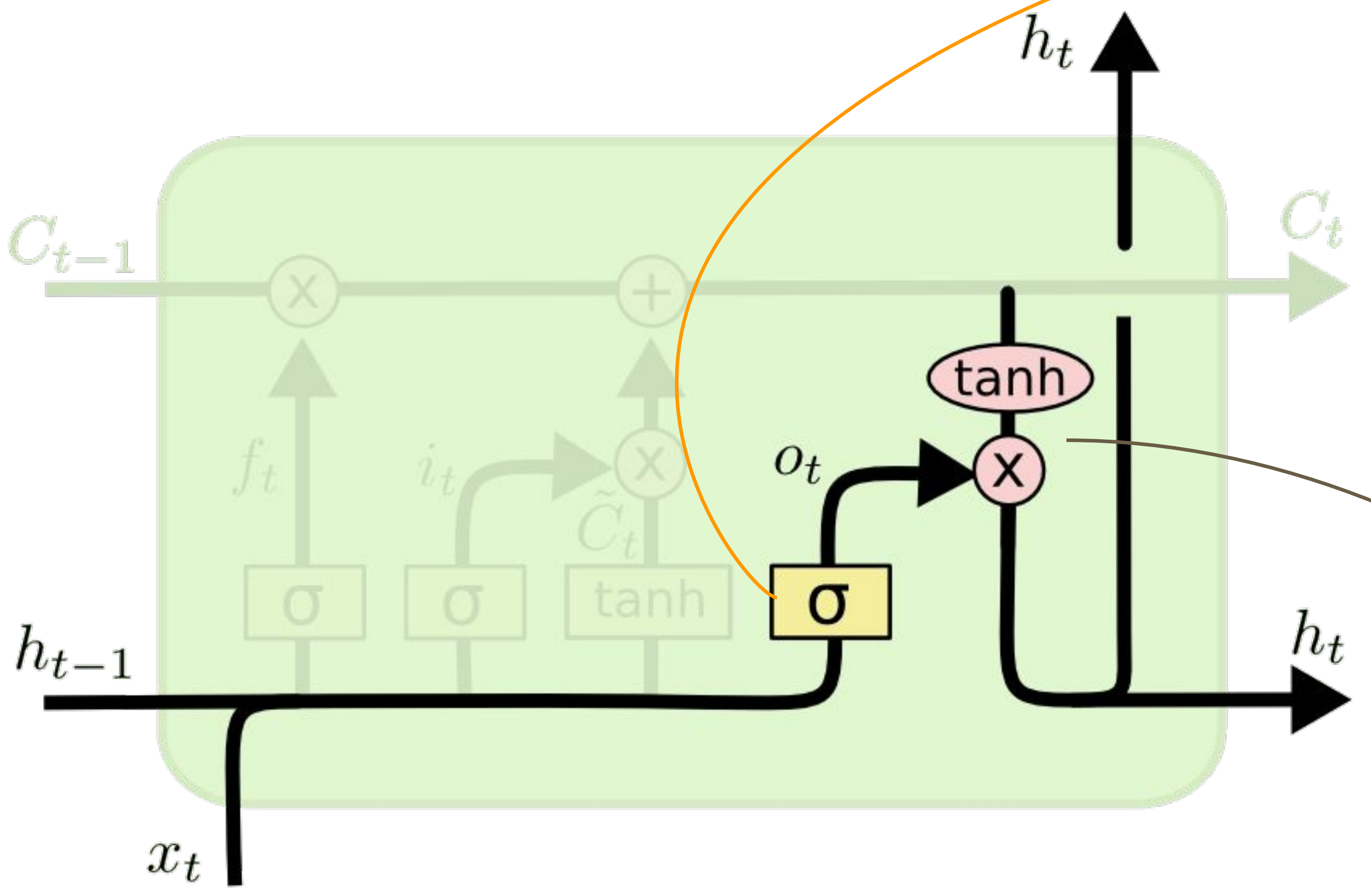
- Purpose: Ensures that only the most relevant information from the cell state is used in the output.
- Example in Language Modeling:
  - Sentence: "***He*** is French. ***She*** is…"
  - Based on the recent subject ("***She***"), the model might output features relevant for predicting a verb, such as the subject's number (singular/plural).




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

# Output

Decides which parts of the cell state will contribute to the output.




Output gate

**sigmoid function**

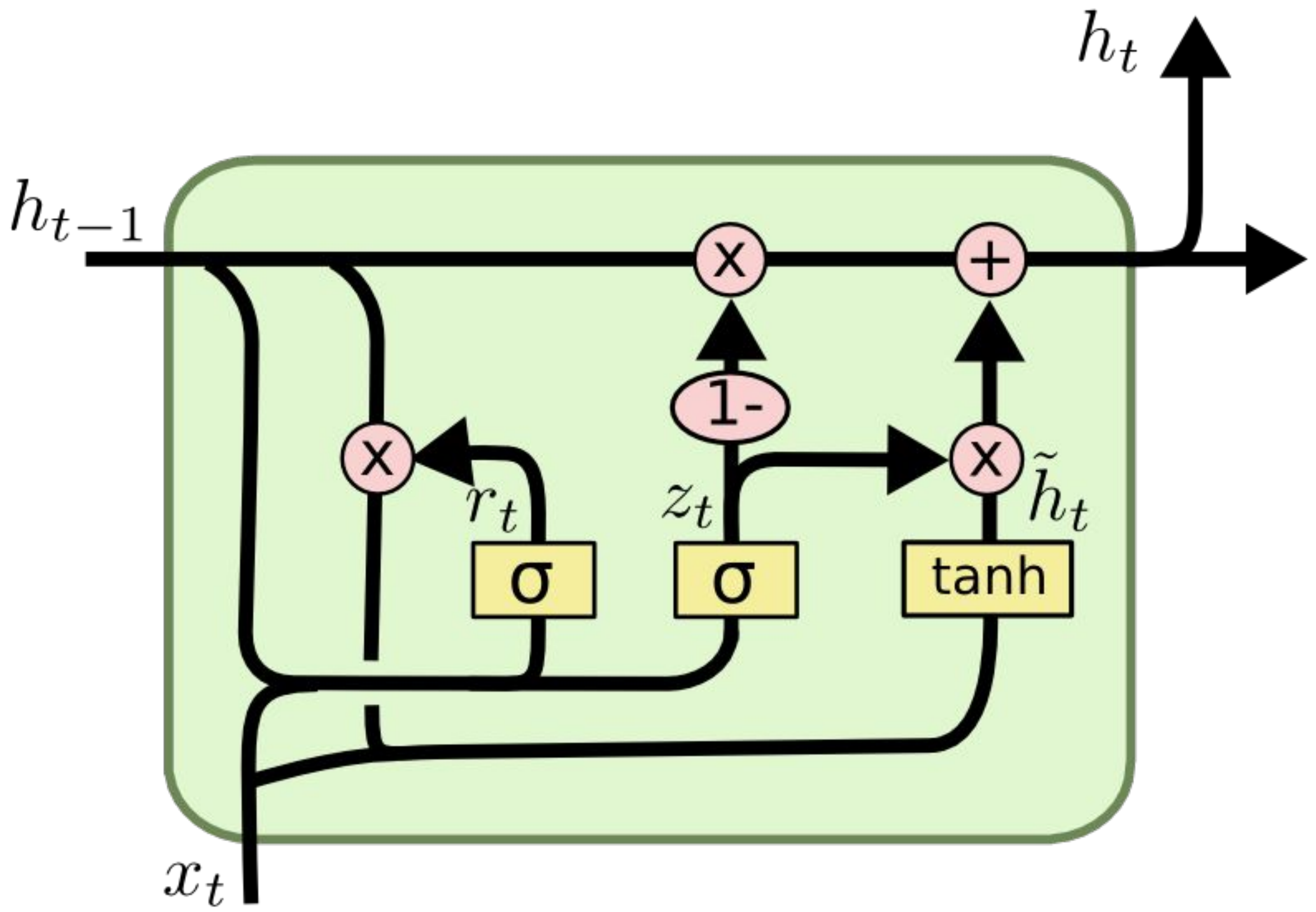
$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

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

**tanh & Multiplication**

Multiplies the normalized state with the sigmoid output to determine the final output.

# Variant on LSTM: Gated recurrent unit (GRU)

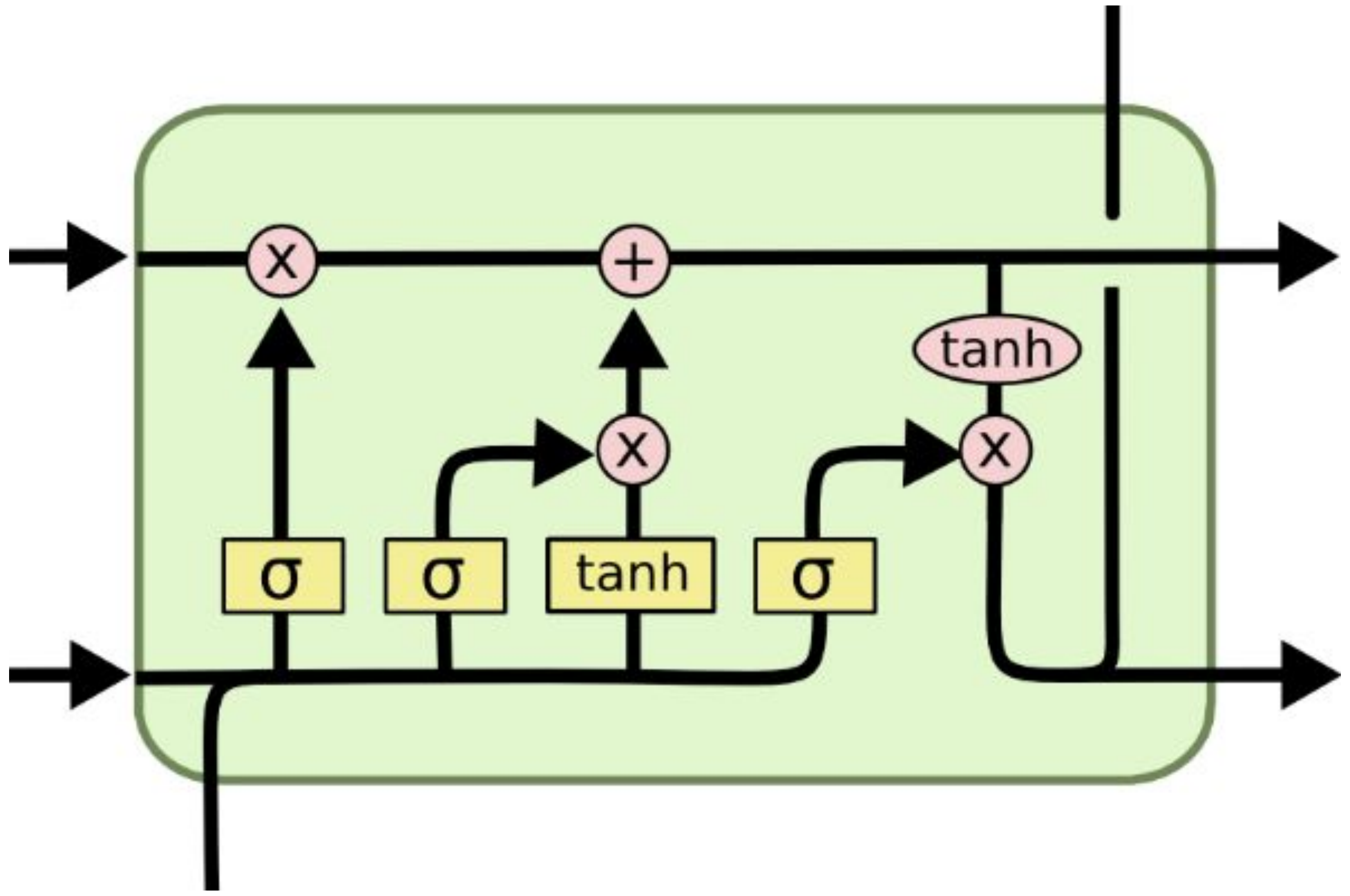



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

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

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

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




Introduced by **Cho, et al.**

- Number of Gates:
  - GRU: 2 gates (reset and update).
  - LSTM: 3 gates (input, output, forget).
- Memory Management:
  - GRU: Less complex, with no separate memory cell.
  - LSTM: Separate cell state and hidden state for refined memory control.
- Training and Performance:
  - GRU: Generally faster to train, suitable for smaller datasets.
  - LSTM: Potentially higher performance on complex tasks, especially with larger datasets.

# Summary

- Recurrent Neural Networks
  - Why do we need RNN?
  - RNNs:
    - Standard RNN
    - LSTM
    - GRU