Lecture 2: Recurrent Neural Networks

COMP 5801H/4900A: Generative AI and LLMs

2026-01-08

Sriram Subramanian
Assistant Professor & Canada Research Chair, Carleton University
Faculty Affiliate, Vector Institute for Artificial Intelligence
Faculty Affiliate, Schwartz Reisman Institute for Technology and Society

Outline

- Foundations of Sequence Modelling
- Recurrent Neural Networks (RNNs)
- The Scaling Wall: Vanishing & Exploding Gradients
- Advanced Solutions: LSTMs and GRUs
- Information and Computational Bottleneck

Why Sequences? The Challenge of Variable-Length Data

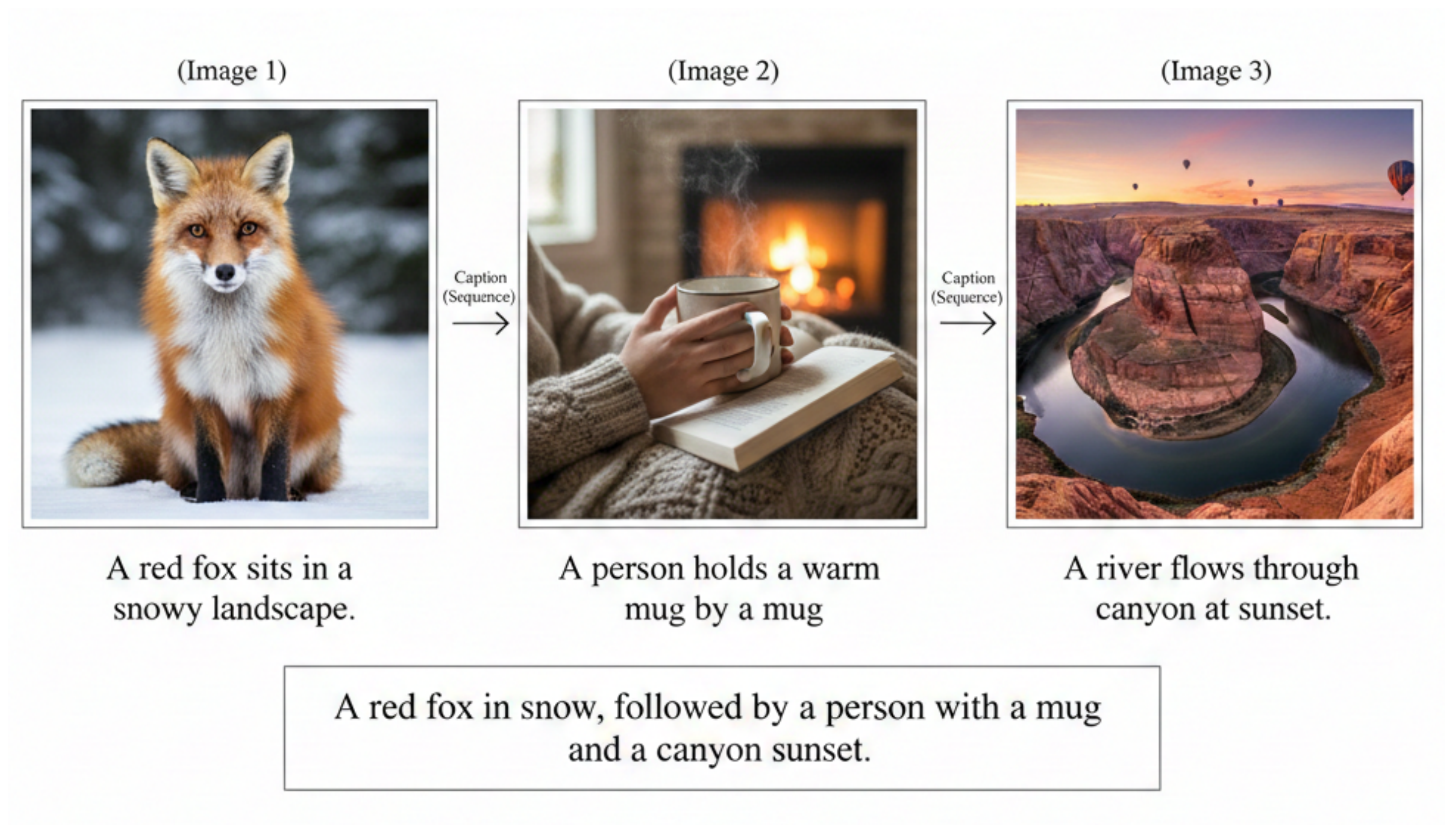
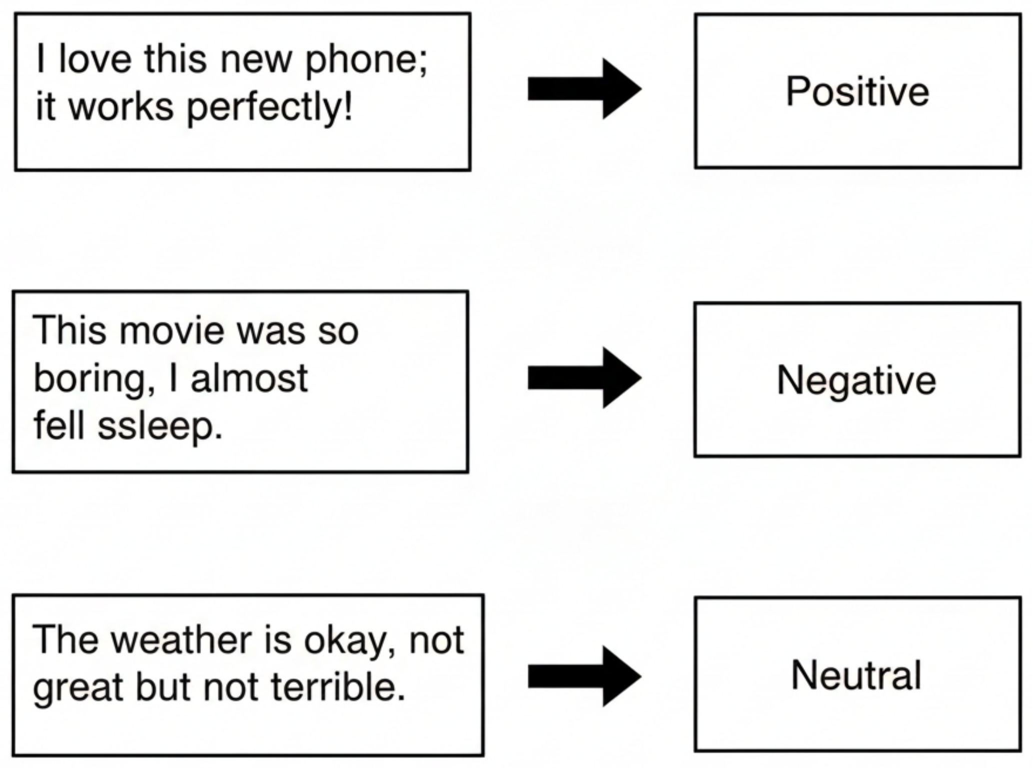
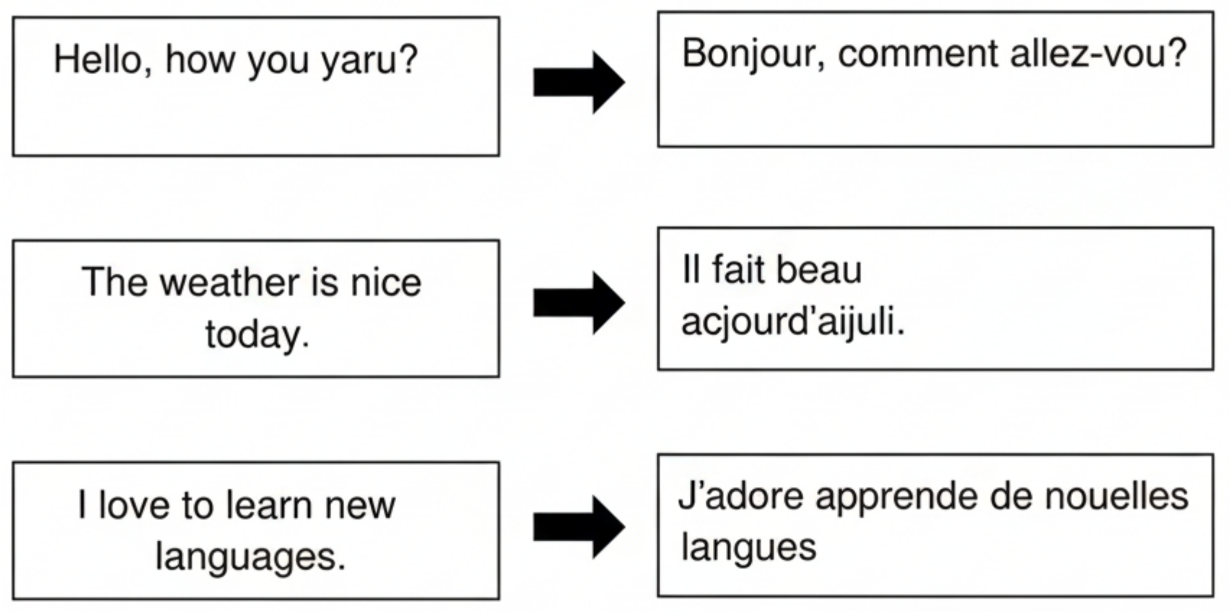

(Image 1)
(Image 2)
(Image 3)
Caption (Sequence)
Caption (Sequence)
A red fox sits in a snowy landscape.
A person holds a warm mug by a mug
A river flows through canyon at sunset.
A red fox in snow, followed by a person with a mug and a canyon sunset.

Image Captioning (many to one)


I love this new phone; it works perfectly!
Positive
This movie was so boring, I almost fell ssleep.
Negative
The weather is okay, not great but not terrible.
Neutral

Sentiment Analysis (many to one)


Hello, how you yaru?
Bonjour, comment allez-vou?
The weather is nice today.
Il fait beau acjourd'aijuli.
I love to learn new languages.
J'adore apprende de nouelles langues

Translation (many to many)

- Standard MLPs
  - Require a **fixed input size** (e.g., a 224×224 image)
  - Only solution: **Padding** (ineffective)
- Sequential Data
  - **Variable length** (use the same logic for 3-word or 30-word)
  - Need a model where **computation scales with data** (not a model that forces data to scale to architecture)

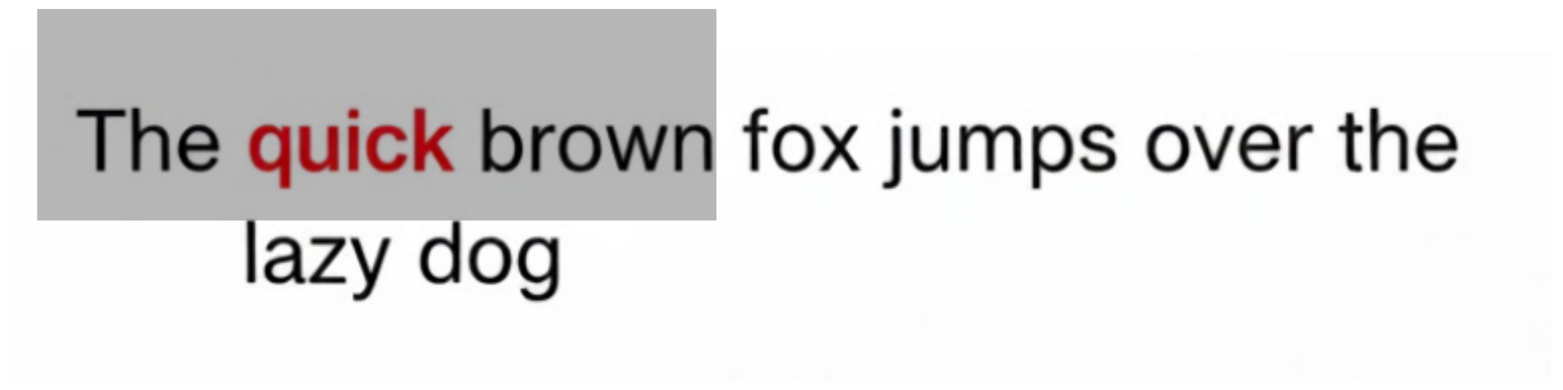
The MLP Limitation: Why Standard Networks Fail at Text

- **Fixed Input Size**
    - MLP: Requires fixed dimension input
    - Sequences are not fixed
    - MLP: Forced to perform padding or truncating
- **Lack of parameter sharing**
    - MLP: Each input has own weight (word 1 different from word 5)
    - Need to share word-meaning weights across every position
- **Ignores Structural Topology**
    - MLP: Cannot capture relational meaning
    - Example: “The dog bit the man” vs. “The man bit the dog”

The Markov Assumption & The N-Gram Era

- **The Markov Assumption**
  - The probability of a future state depends only on the current state (or a very small window of past states)
  - Formula: $P(w_i | w_1, \ldots, w_{i-1}) \approx P(w_i | w_{i-k}, \ldots, w_{i-1})$
- **N-Grams: Modelling local Context**
  - Unigram (n=1): "I", "want", "to", "eat". (No context)
  - Bigram (n=2): "I want", "want to", "to eat". (1 word of history)
  - Trigram (n=3): "I want to", "want to eat". (2 words of history)
  - The Trade-off: As n increases, the model's "context" grows, but the data becomes sparse

N-gram limitations


The quick brown fox jumps over the lazy dog

- **Long-Range Dependencies**:
  - Language is hierarchical, not just linear
  - Could miss the major parts of a sentence (noun, verb etc.)
- **Global Context**: N-grams can’t track the topic
- **No persistent state**

Discussion: "The Markov Assumption suggests that we can predict the future based only on a small window of the present. RNNs try to break this by maintaining a persistent state. If you had to model a human life, is it more 'Markovian' (we are the product of our current circumstances) or 'Recurrent' (we are the sum of everything that has ever happened to us)? Where do you think the limit of 'useful' history lies?"

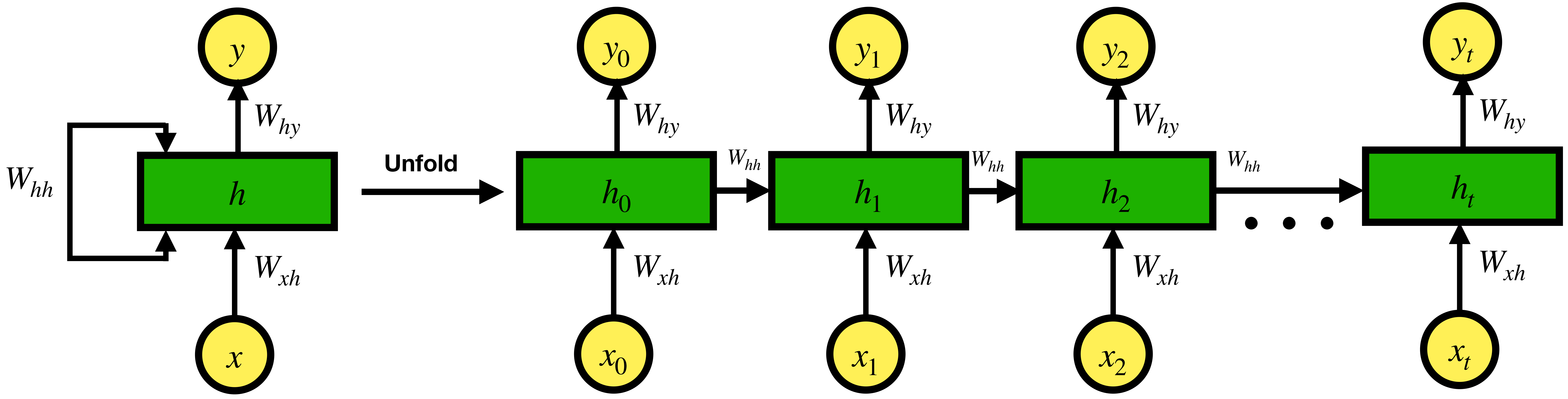
Introduction to Recurrence: The Hidden State

- **Memory as a Loop**
  - "Hidden State" (h_t):"mental state" or "summary" of past until time t
  - Persistence: Feeds own output back into itself as the next input
  - Formula for “now”:

$$h_t = \text{Weighted combination [Current Input (}x_t\text{) + Previous Memory (}h_{t-1}\text{)]}$$

- **Breaking the “snapshot”**
  - MLPs: Output = f(Input) - Each input is separate, isolated
  - RNNs: Output = f (Input, History) - Reaction based on memory

RNNs



The “Unrolled Perspective”

- Draw as a loop, but compute as a chain
- Each "step" in the chain uses the exact same weights (W), but the **Hidden State** evolves at every step

The Basic RNN Cell: Under the Hood

Fundamental equation to **calculate hidden state** at any t

$$h_t = \sigma(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$

- h_t: The new hidden state (Current Memory)
- h_{t-1}: The hidden state from the previous step (Past Context)
- x_t: The current input (The new word/data point)
- σ: Activation function (usually tanh or ReLU) to add non-linearity
- b_h: Bias for the hidden state

Fundamental equation to **calculate output** at any t

$$y_t = \sigma(W_{hy}h_t + c_y)$$

- y_t: Current output
- c_y: Bias for the output

Discussion: The Hidden State (h_t) is often called a 'summary' of the past. However, every summary involves a loss of detail. At what point does a summary stop being a 'representation' and start being a 'distortion'? If we compress a 500-word essay into a single 512-dimension vector, can we truly say the model 'knows' the essay, or does it just know a 'shadow' of it?

RNN Cell - Weight Matrices

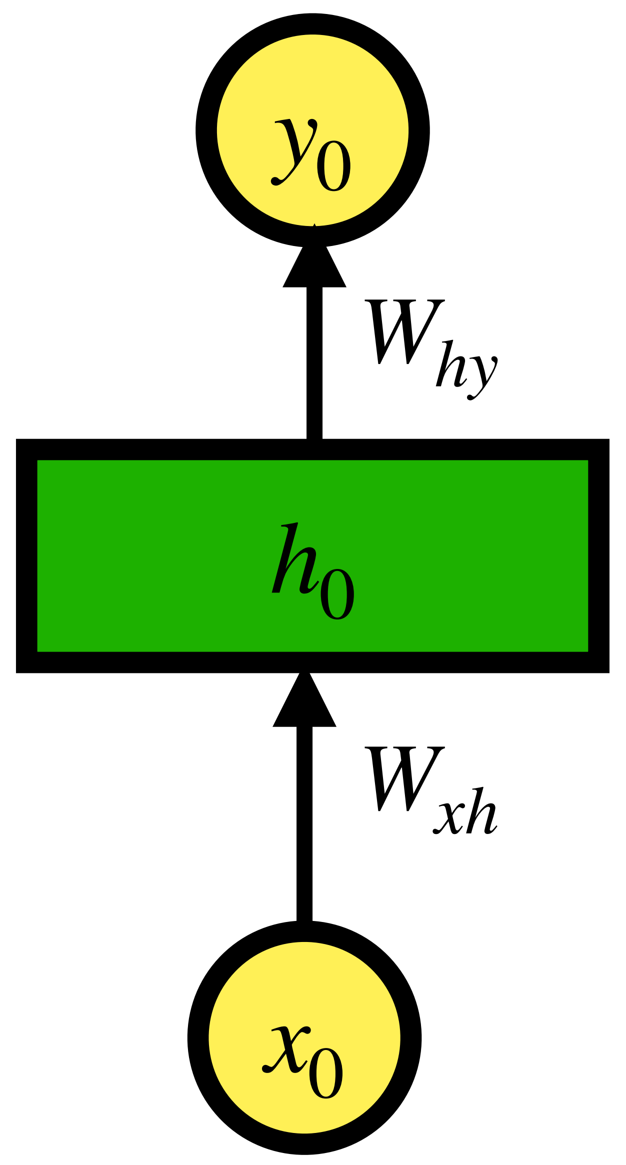
The **Three Weight Matrices**: The "intelligence" of the RNN is stored in these shared parameters

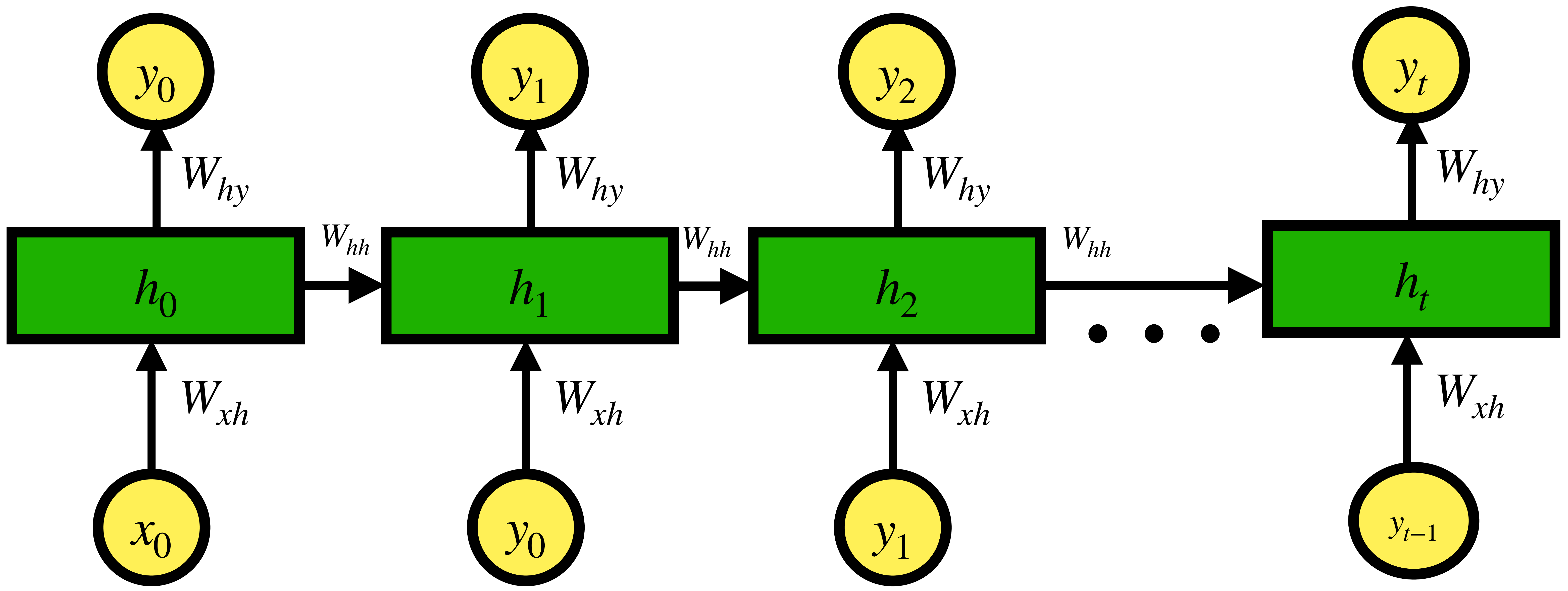
- W_{xh} (Input-to-Hidden): How much should the current word change our internal state?
- W_{hh} (Hidden-to-Hidden): How much of the past should we carry forward?
- W_{hy} (Hidden-to-Output): How much impact should the hidden state have on the output?

Parameter Sharing

- Unlike an MLP where every word position has **different weights**, the RNN uses the **exact same** W_{xh} and W_{hh} for **every word in the sentence**
- Benefit: Same **number of parameters** for a sentence with 5 words or 500 words
- Why does this matter?
    - Position Invariance
    - Unlimited Sequence Length Management
    - Efficiency

RNN Architectures





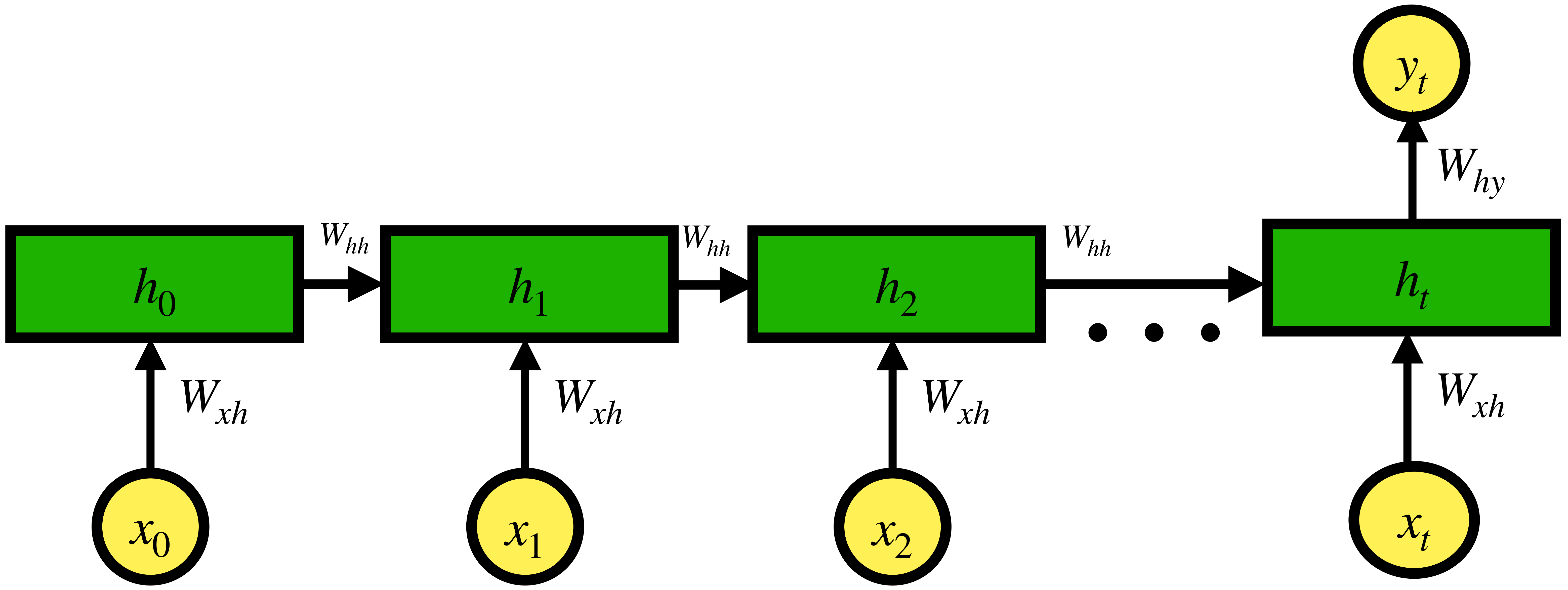
One-to-One

Traditional Neural Network
(Classification/Clustering)

One-to-Many

Music Generation

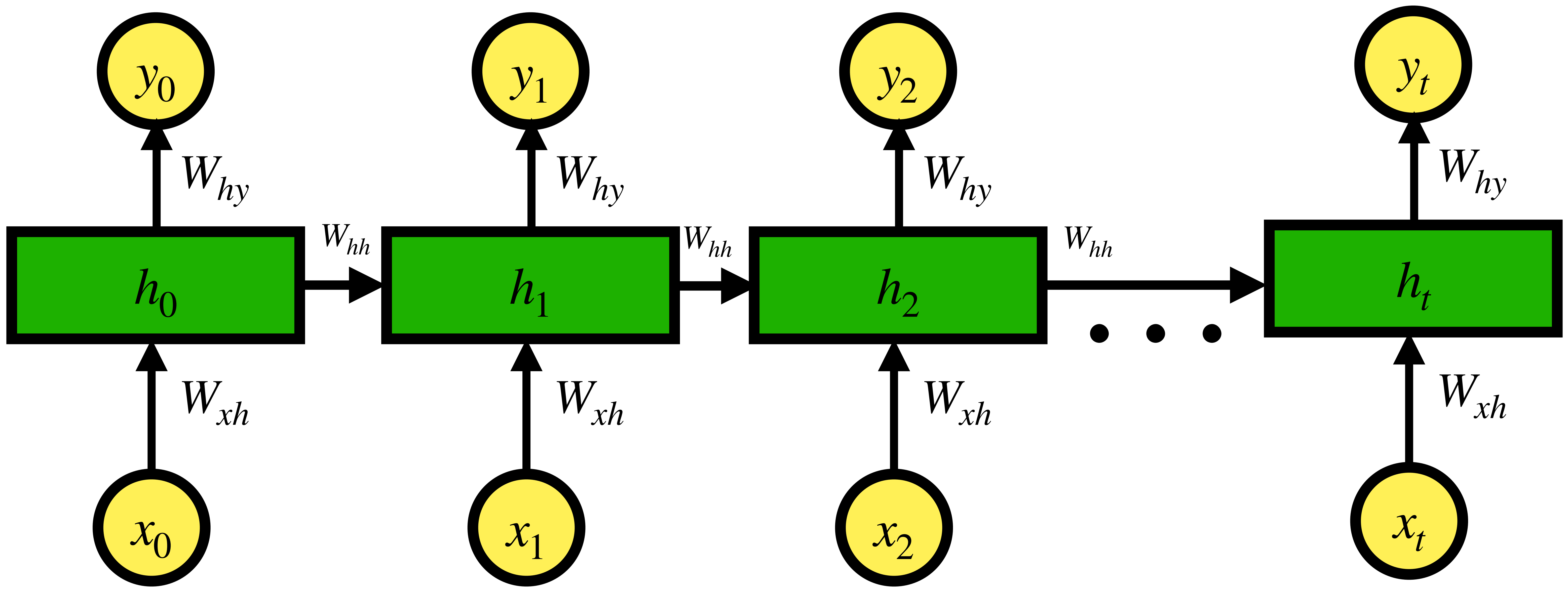
RNN Architectures - II



Many-to-One

Named Entity Recognition

RNN Architectures - III



Many-to-Many

Machine Translation

Backpropagation Through Time (BPTT)

- Let L represent the total loss which depends on the final hidden state (say h_t)
- This hidden state depends on the previous hidden state (h_{t-1}) which depends on the previous hidden state h_{t-2} and so on till h_0
- Hence, gradients are **backpropagated through time** (through the chain rule)
- Gradient of the total loss
  - The total loss L is the sum of losses at each time step
  - To update our weights (W_{hh}), we need the derivative of the total loss
  - $\dfrac{\partial L}{\partial W_{hh}} = \sum_{t=1}^{T} \dfrac{\partial L_t}{\partial W_{hh}}$

Backpropagation Through Time (BPTT)

- **Chain rule for a single time step**
  - Consider specific loss L_t at time t
  - Gradient must account for every previous state W_{hh} influenced (Multivariate Chain Rule)

$$\frac{\partial L_t}{\partial W_{hh}} = \sum_{k=1}^{t} \frac{\partial L_t}{\partial h_t} \times \frac{\partial h_t}{\partial h_k} \times \frac{\partial h_k}{\partial W_{hh}}$$

- The term $\frac{\partial h_t}{\partial h_k}$ is the chain that carries the error back through time

$$\frac{\partial h_t}{\partial h_k} = \Pi_{i=k+1}^{t} \frac{\partial h_i}{\partial h_{i-1}}$$

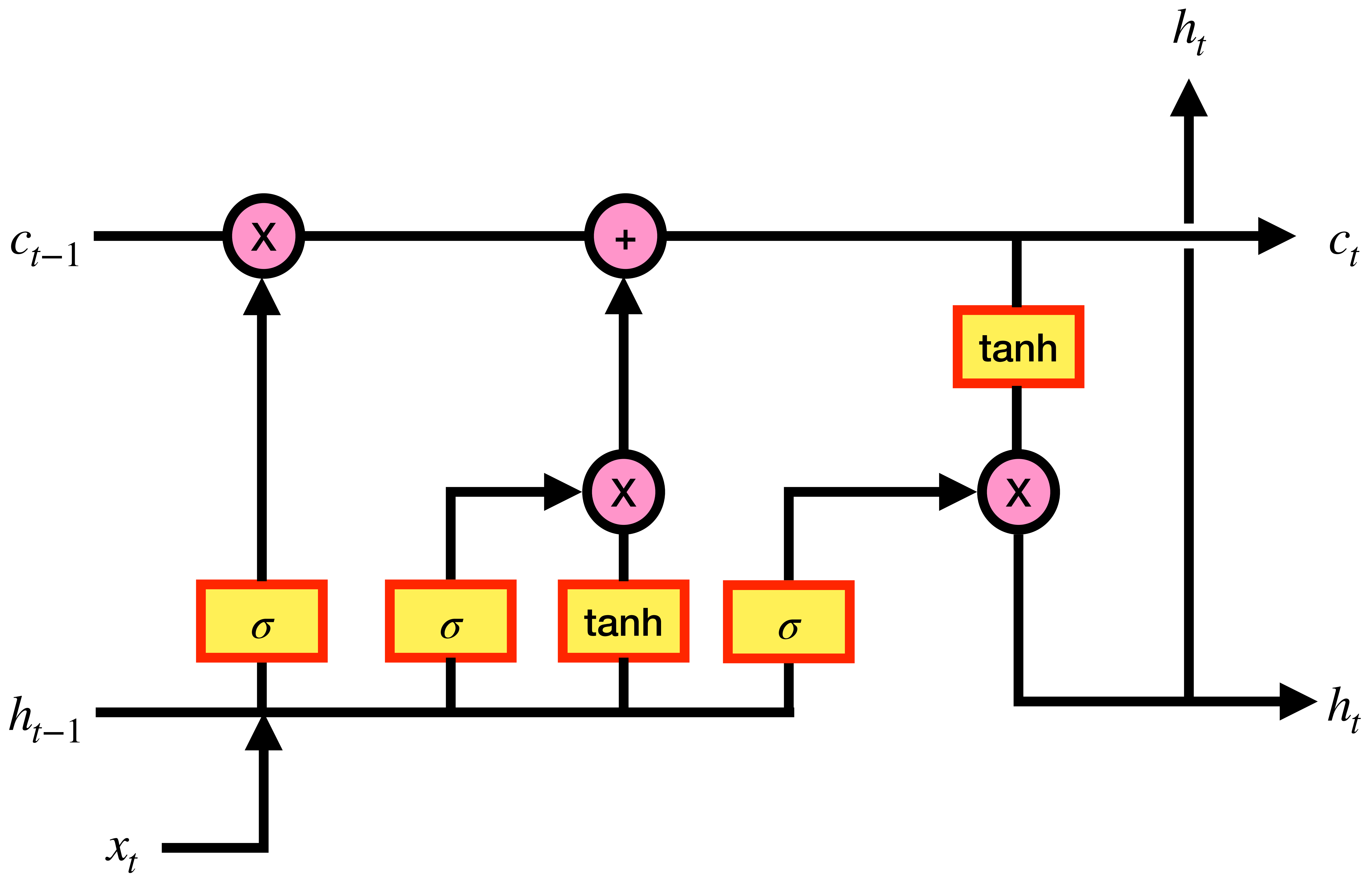
- Recall, $h_i = \sigma(W_{hh}h_{i-1} + W_{xh}x_i + b)$
- The product involves multiplying W_{hh} by itself many number of times

Why this math matters

- **Product of gradients**: If your sequence is 50 words long, you are multiplying W_{hh} by itself nearly 50 times
- **Vanishing Gradient:** If the largest eigenvalue of W_{hh} is < 1, the product will shrink towards 0 (forget the start of sentence)
- **Exploding Gradient:** If the largest eigenvalue of W_{hh} is > 1, the product will grow toward ∞ (model blows up and training crashes)
  - $0.9^{10} \approx 0.34$
  - $0.9^{50} \approx 0.005$ (vanishing)
  - $1.1^{50} \approx 117.39$ (exploding)

Discussion: "Because RNNs use Parameter Sharing, the 'rules' for processing a word never change. However, the 'context' (the hidden state) is always different. If the rules are fixed but the context is infinite, is the model's output truly 'creative,' or is it just a deterministic reaction to a long chain of events? Can a machine ever 'surprise' us if we are the ones who defined its recurrence?"

LSTM - Repeating Module



LSTM - Components

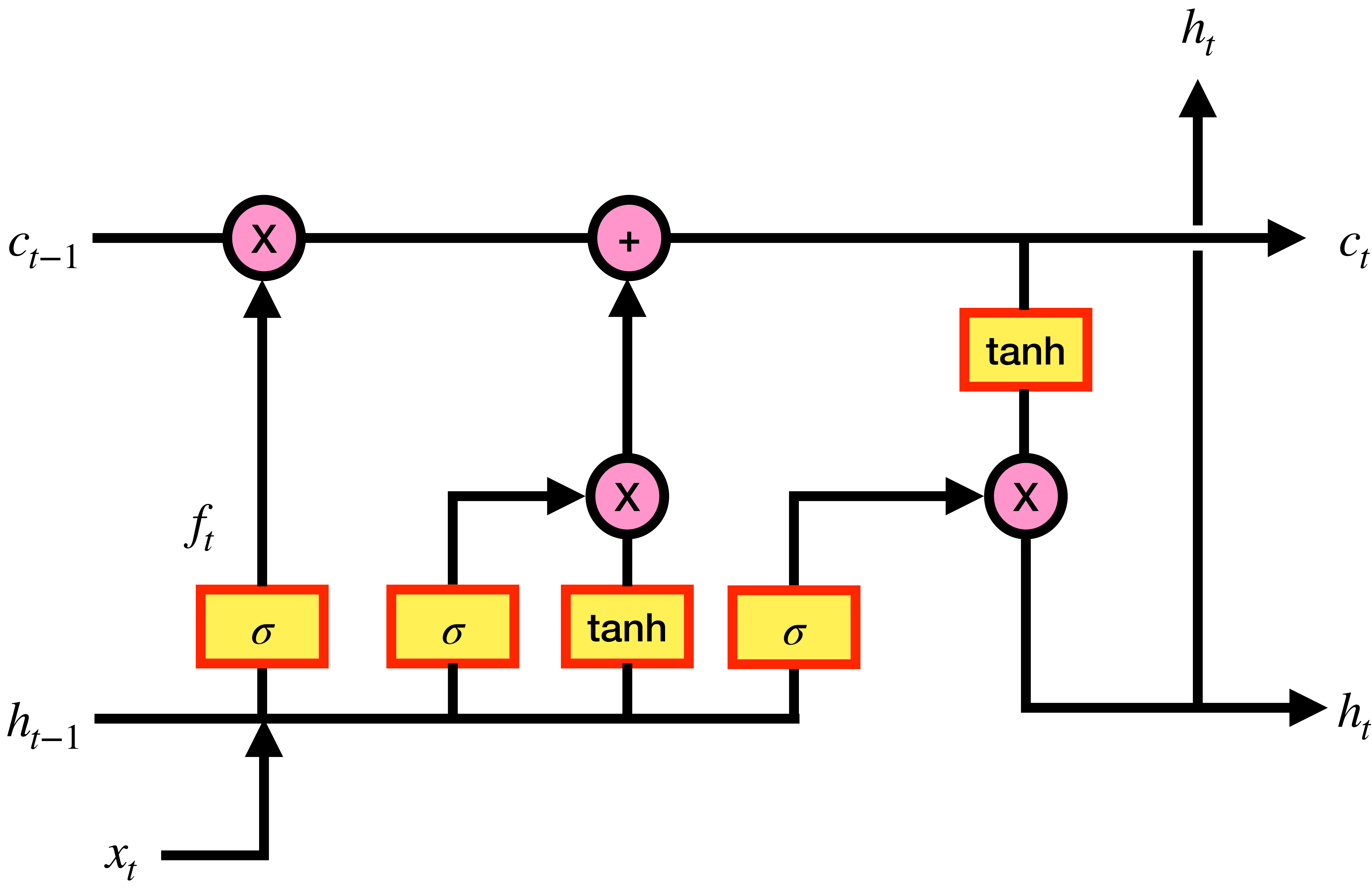
- **Cell state (c_t)**
    - Hidden state is subject to heavy multiplication and non-linearities
    - Cell state stays virtually the same (like a conveyor belt)
    - Information flows down with very little interference, allowing gradients to stay stable
- **Gates: Controls the flow of information**
    - Forget Gate: "What should we throw away from the past?"
    - Input Gate: "What new information is worth keeping?"
    - Output Gate: "What parts of our internal memory should we show the world right now?”

Inside the LSTM

Forget Gate

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

- Looks at previous hidden state and new input
- Decides what is relevant
- Example: If the sentence moves from a singular subject to a plural one, the forget gate "erases" the singular grammar rules



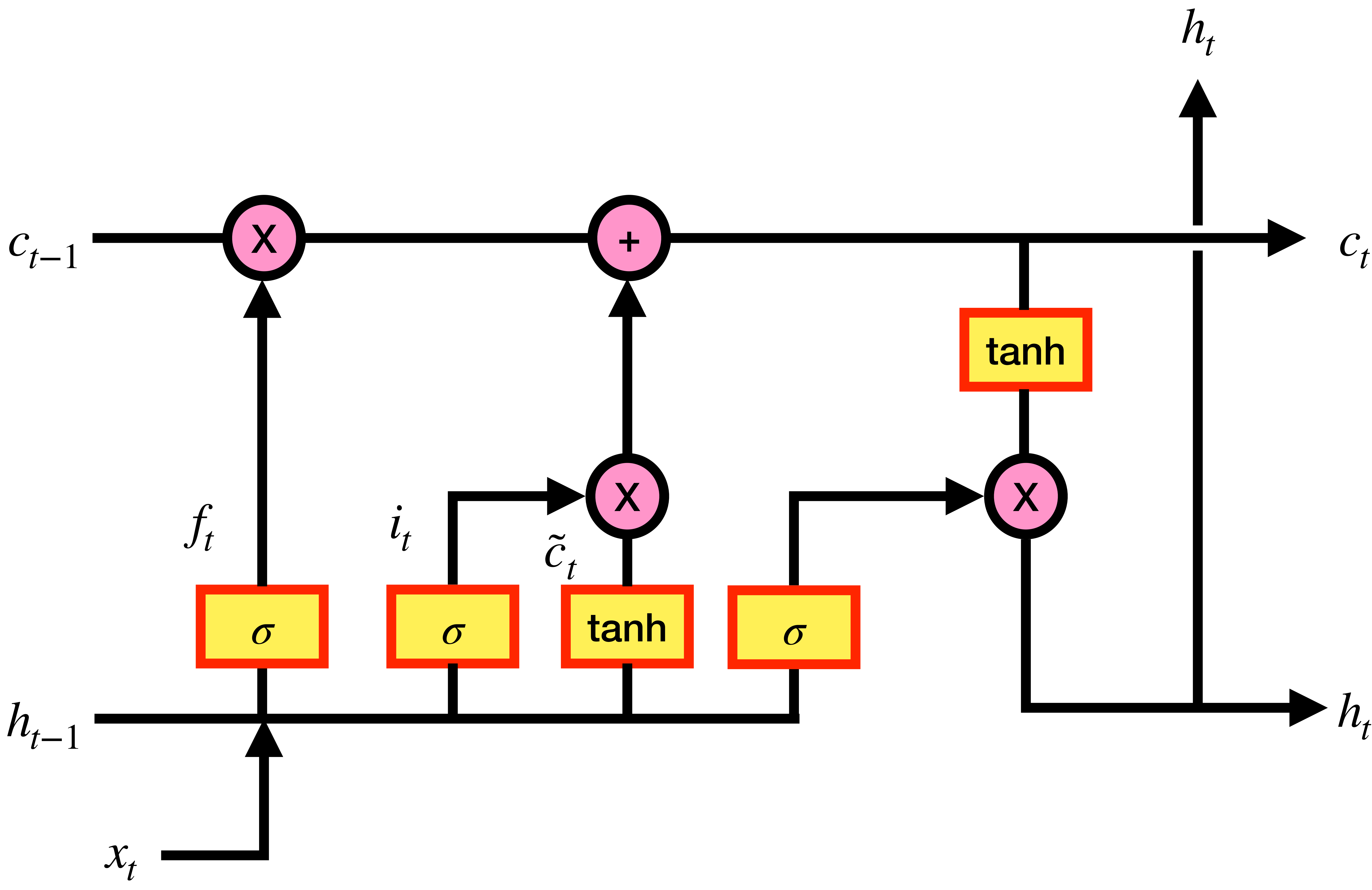
Inside the LSTM- II

Input gate & Candidate state - New information to store in the cell state

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

$$\tilde{c}_t = tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$

- Input gate: decides how much of the new information should be allowed into the long-term memory
- Candidate state: Creates new candidate values to add to the cell state

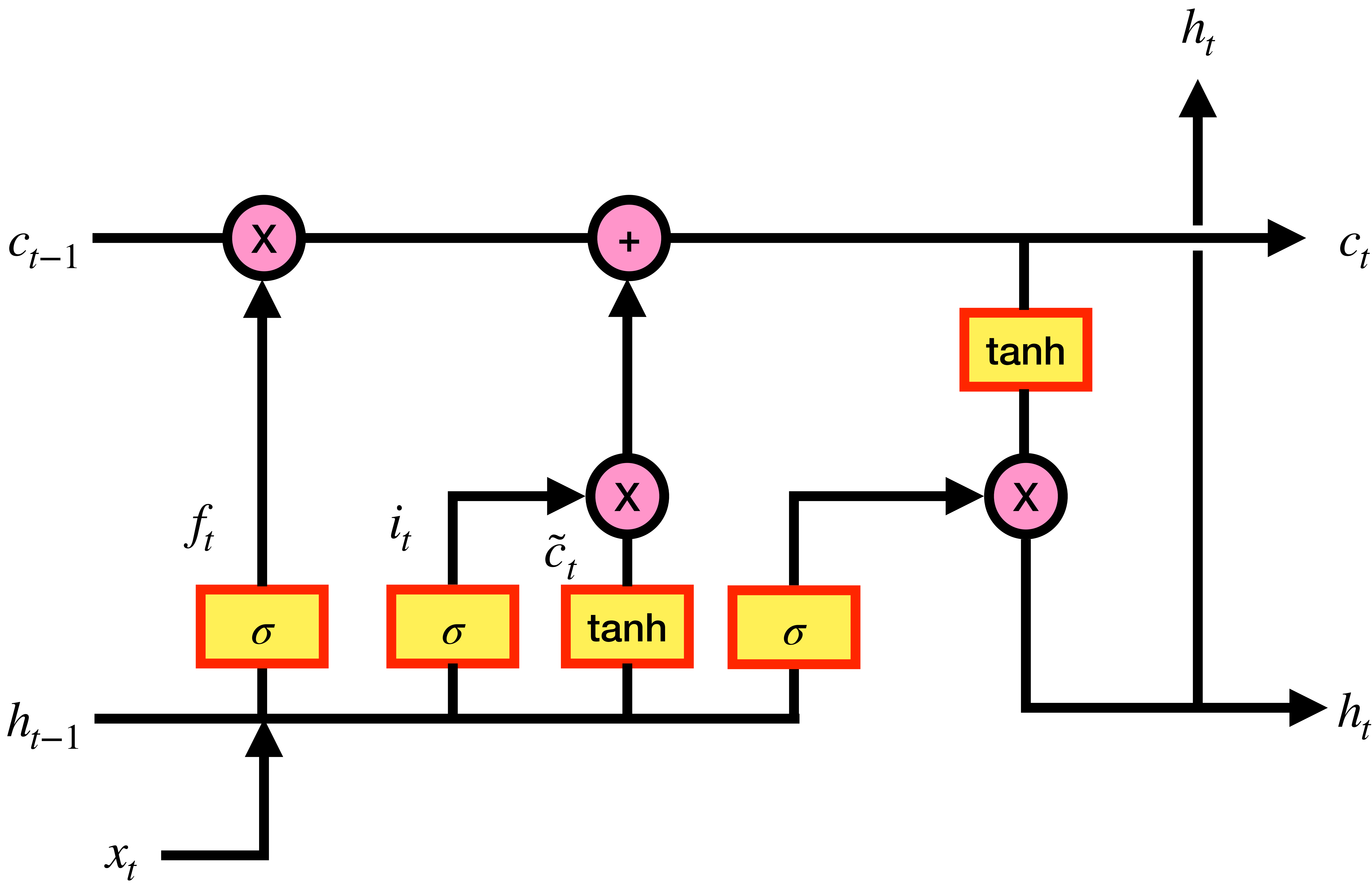


Inside the LSTM - III

Updating the cell state

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

- We combine the old state and the new candidate state
- Additive update: gradient can flow through the “+” sign and not be diminished by weight



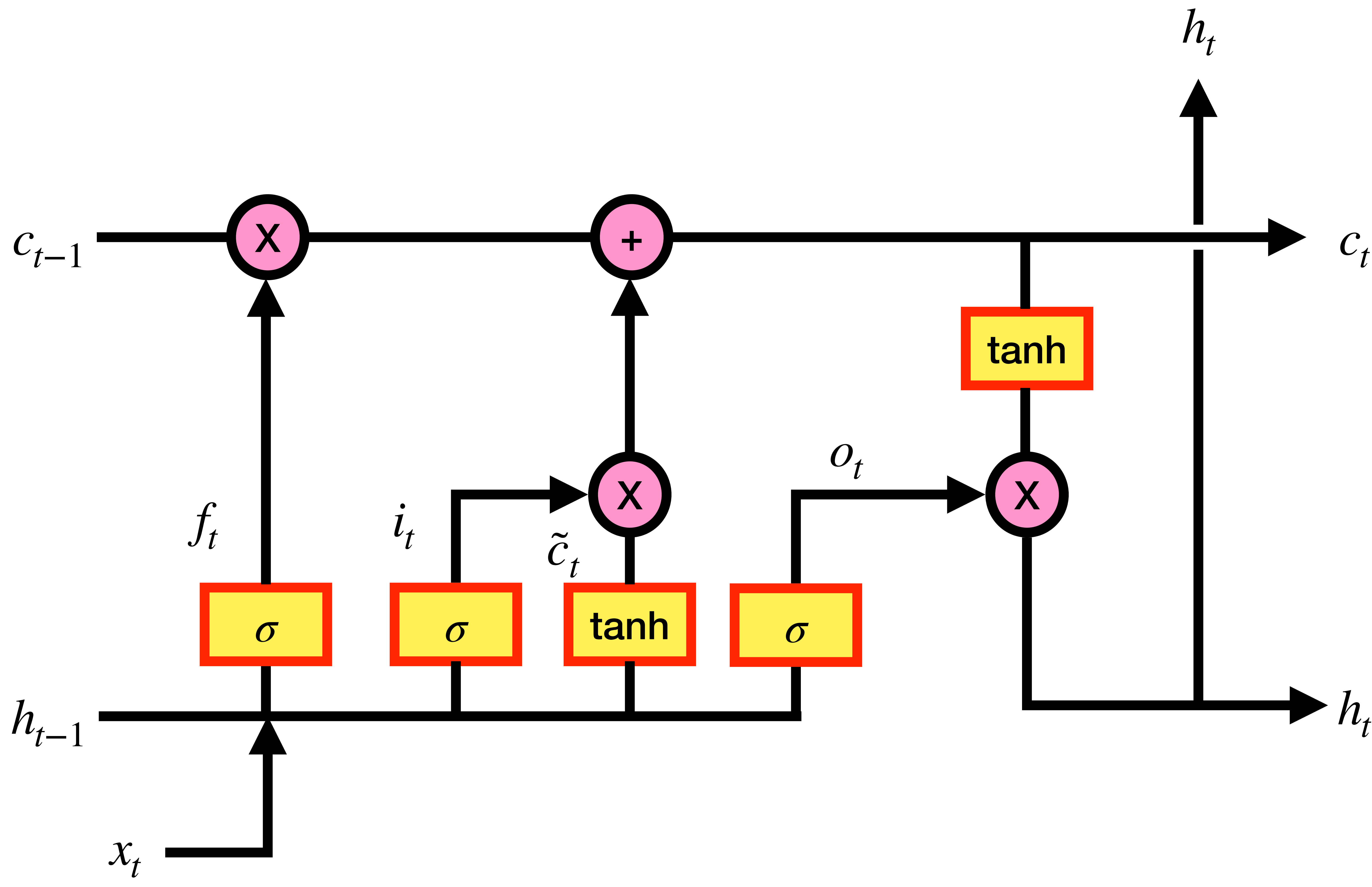
Inside the LSTM - IV

Output Gate

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

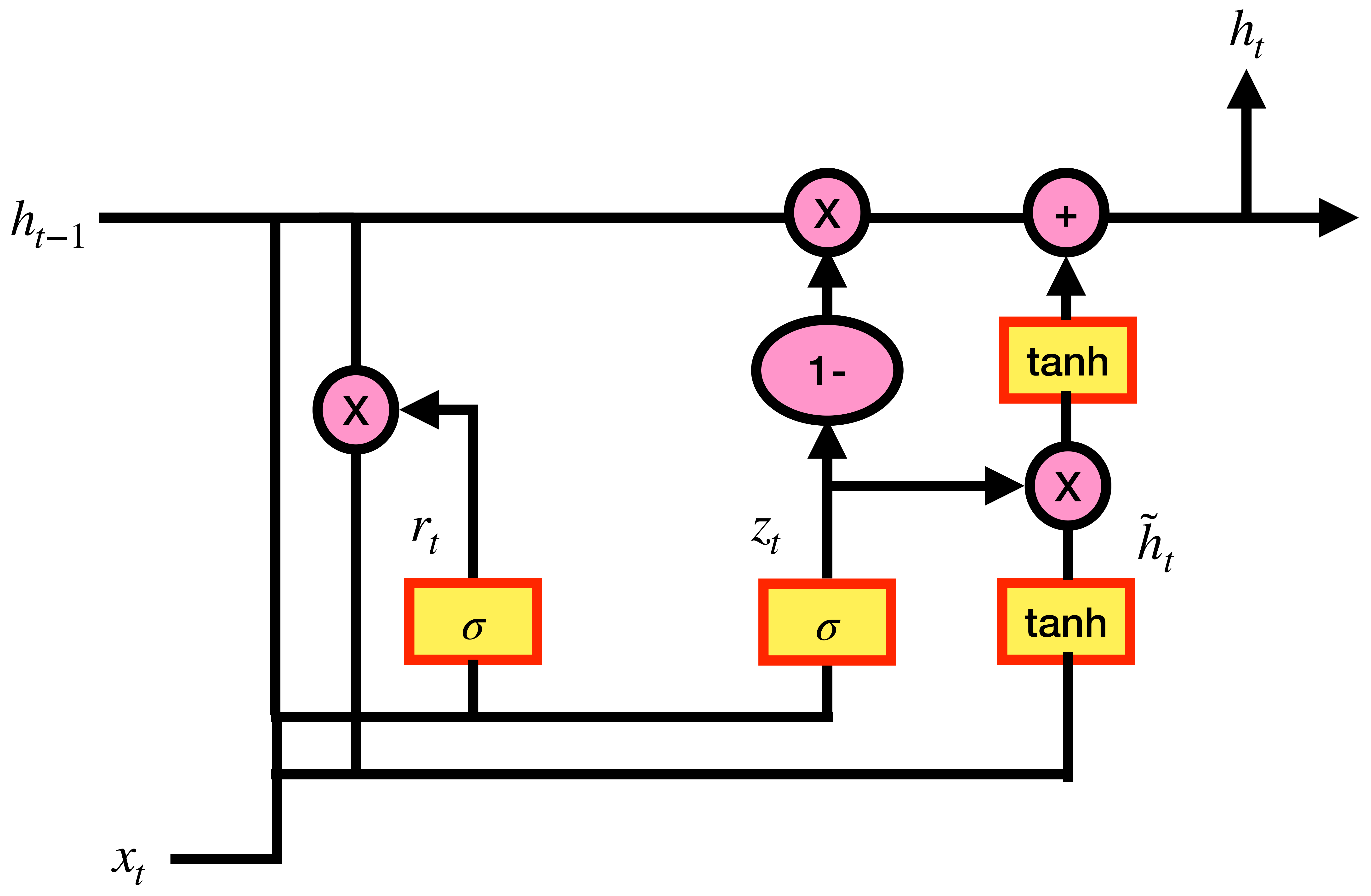
$$h_t = o_t \cdot tanh(c_t)$$

- Finally, we decide what the next hidden state (h_t) will be
- Pass the cell state through a tanh and multiply it by the output gate



Discussion: In an LSTM, we explicitly design a Forget Gate to discard information. In human cognition, is 'forgetting' a bug or a feature of intelligence? If a model had 'perfect memory' (a Forget Gate always set to 1), would it actually become less capable of understanding complex, evolving ideas?

GRU



Gated Recurrent Unit (GRU) - Simplified LSTM

- **Elements**
  - Only two gates: Update Gate and Reset Gate
  - Merges "Cell State" and "Hidden State" into a single vector h_t
- **The Update Gate (z_t):**
  - Combines forget and input gates
  - The Update Gate decides how much of the past to keep and how much of the new candidate to add at the same time
- **The Reset Gate (r_t):**
  - This gate determines how much of the previous hidden state to ignore when calculating the new candidate state
  - Capturing short-term dependencies or "starting fresh"

GRU - Update equations

- **Reset Gate (r_t)** - Decides how much of the past to ignore when calculating the new potential state

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

- **Update Gate (z_t)** - Decides how much of the past state to keep versus how much of the new state to adopt

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

- **Candidate Hidden State ($\tilde{h}_t$)** - Calculates the "new information"

$$\tilde{h}_t = tanh(W \cdot [r_t \odot h_{t-1}, x_t] + b)$$

- **Final Hidden State (h_t)**

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

LSTM vs GRU

Feature	LSTM	GRU
States	Two	One
Gates	3 (Forget, Input, Output)	2 (Update, Reset)
Complexity	High (more parameters)	Lower (faster, less memory)
Performance	Often better on large datasets	Often better on smaller datasets

The Information & Computational Bottleneck

- **The Sequential Constraint**
  - The Problem: In an RNN, h_t cannot be computed until h_{t-1} is finished
  - No Parallelization: RNNs are "locked" into a serial time-chain
  - The Cost: Hardware has to wait for each word to finish before moving to the next
- **The State Compression (Information)**
  - We use the final hidden state (h_t) as the "mental state" or summary of the entire past
  - Fixed Capacity: Whether the sentence is 5 or 500 words, the vector size stays the same
  - Loss of Detail: Forcing a massive amount of history into a small, fixed-size vector
- **The Path of the Gradient**
  - Even with LSTM's cell state, the gradient must travel through every time step
  - As t increases, the probability of the signal being corrupted increases (regardless of the gating architecture)