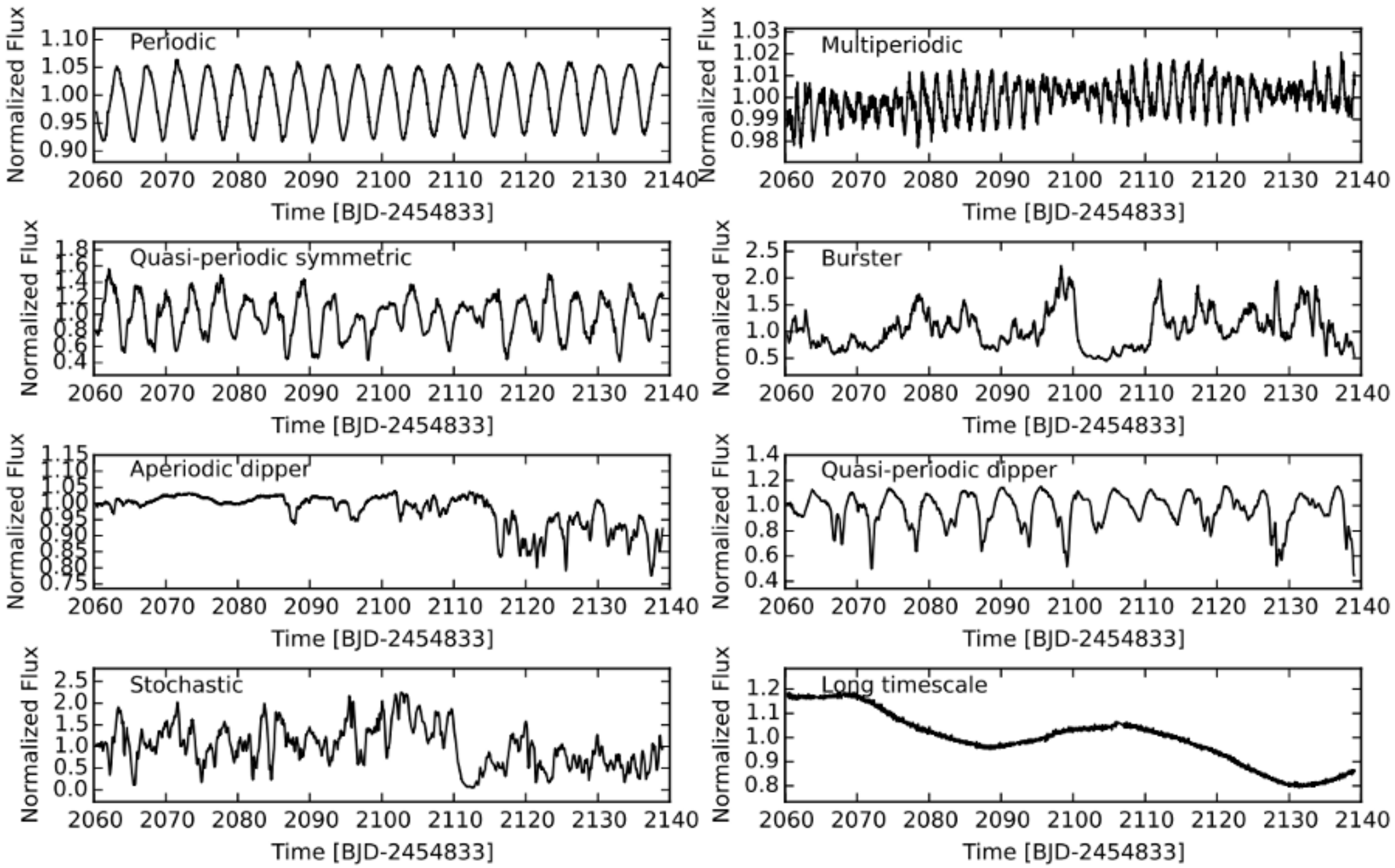
PHYS 139/239: Machine Learning in Physics

Lecture 9: Time-series data and recurrent neural networks

Javier Duarte — February 7, 2023

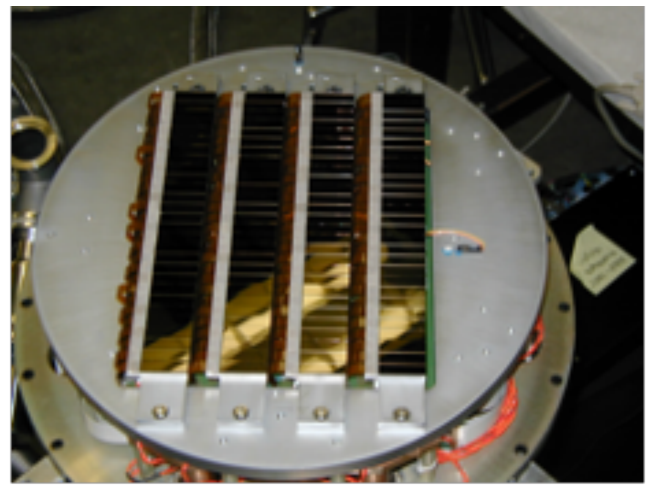
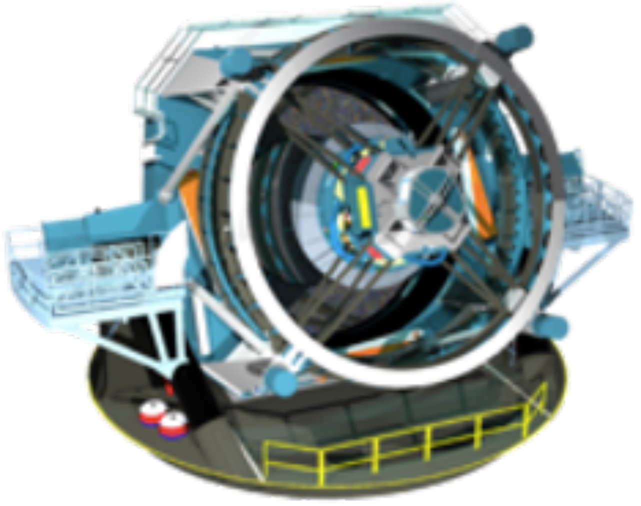
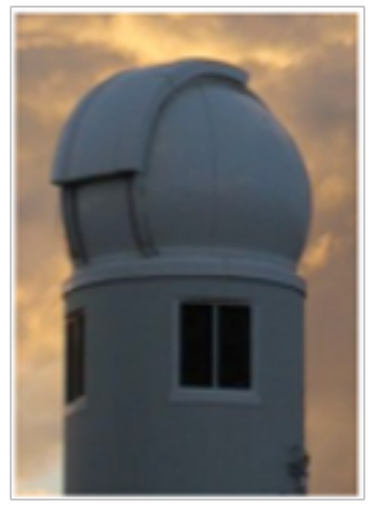
Time-series data tasks

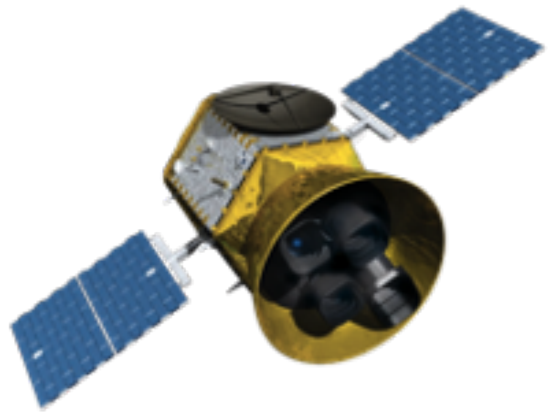
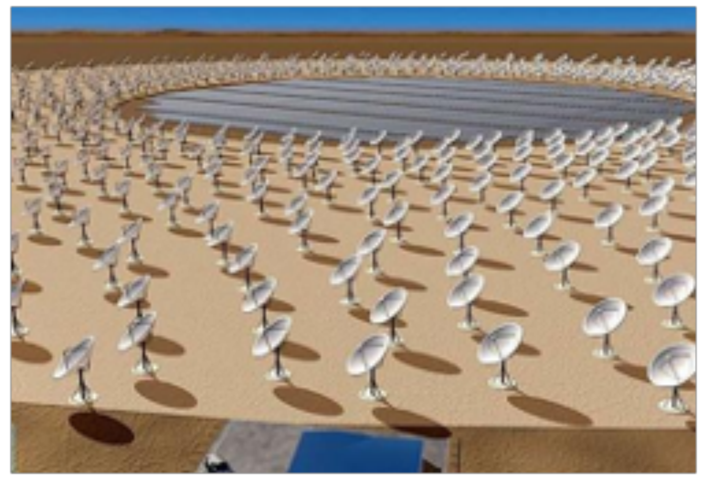
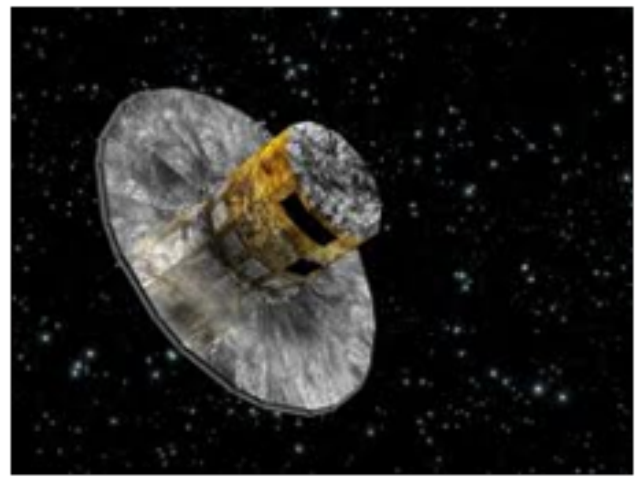
- Population behaviors
  - Characterize, categorize, classify
- Outliers
  - Extreme sources
- Physical models
  - Predictions

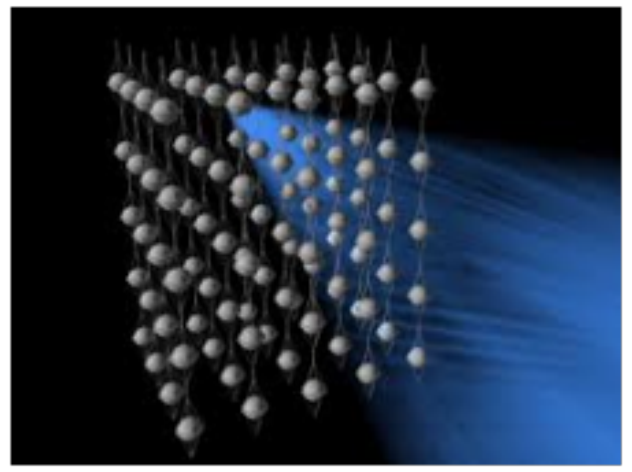


Time-series datasets

- Palomar-Quest Synoptic Sky Survey
- SDSS (Stripe 82)
- Catalina Real-time Transient Survey
- Palomar Transient Factory
- Zwicky Transient Factory
- KEPLER
- GAIA
- LIGO
- …

Time-series definition

- A time series is a set of time-tagged measurements: $\{X_i(t_i)\}$ possibly with observation errors σ_i
- Not i.i.d.
  - Data is sequential (i.e. next point depends on previous point)
- Homoskedasticity
  - All errors drawn from same process
- Ergodicity
  - The time average for one sequence is the same as the ensemble average:

$$\hat{f}(x) = \lim_{n \to \infty} \frac{1}{n} \sum_{k=0}^{n-1} f(T^k x)$$

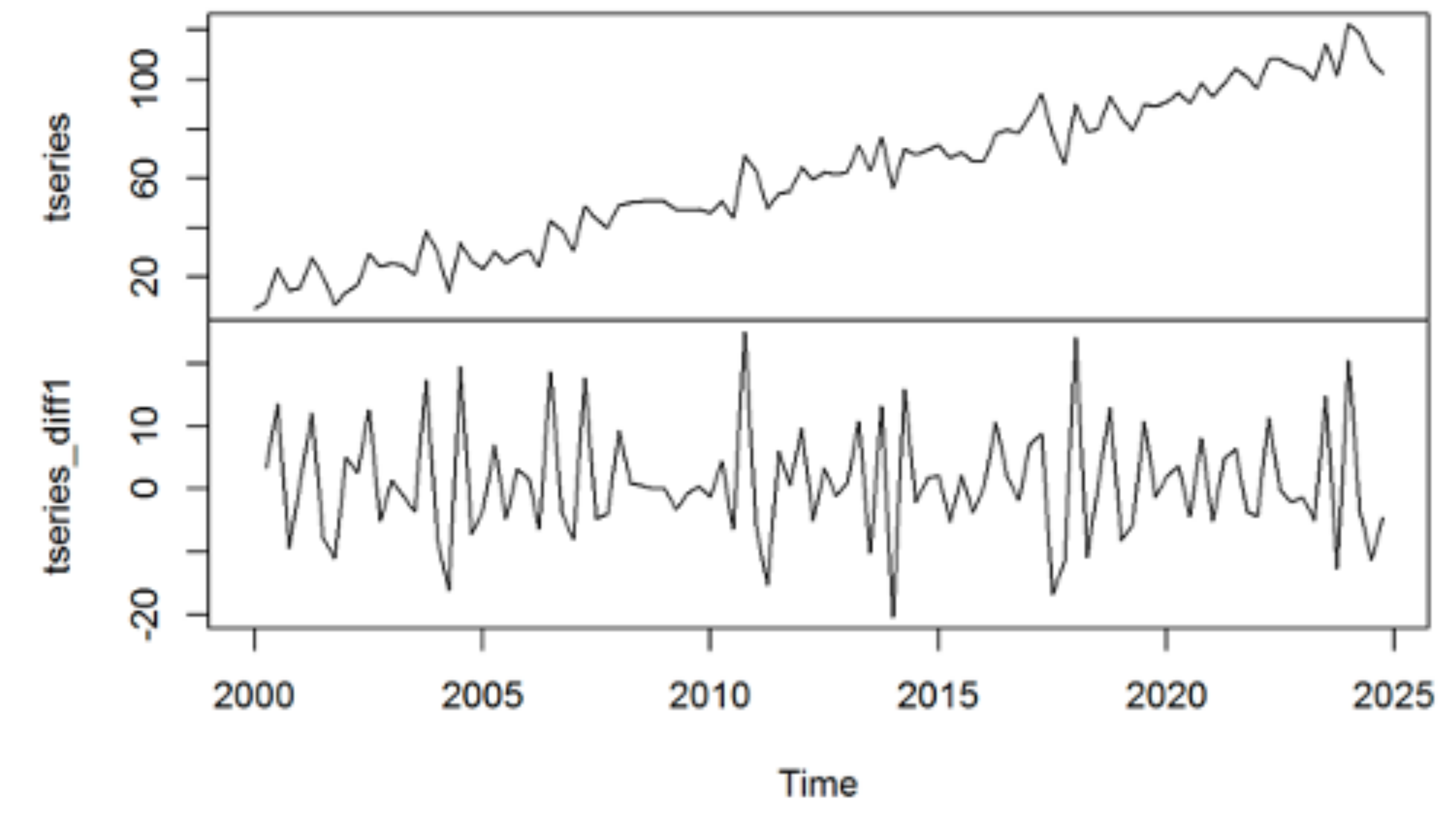
Stationarity

- The generating process is time independent:
  - Joint probability distribution is translationally invariant (strong)
  - Mean, variance, autocorrelation are constant (weak)

Stationarity

- Transformations to achieve stationarity (constant location and scale)
  - Difference the data: $Z_i = X_i - X_{i-1}$
  - Detrend the data: $Z(t) = X(t) - f(t)$
  - Stabilize the variance:

$$Z(t) = \sqrt{X(t) + A} \text{ or } \log(X(t) + A)$$



- Test with autocorrelation function (ACF):

$$\rho_k = \frac{\mathrm{cov}(X_t, X_{t+k})}{\mathrm{var}(X_t)\mathrm{var}(X_{t+k})}$$

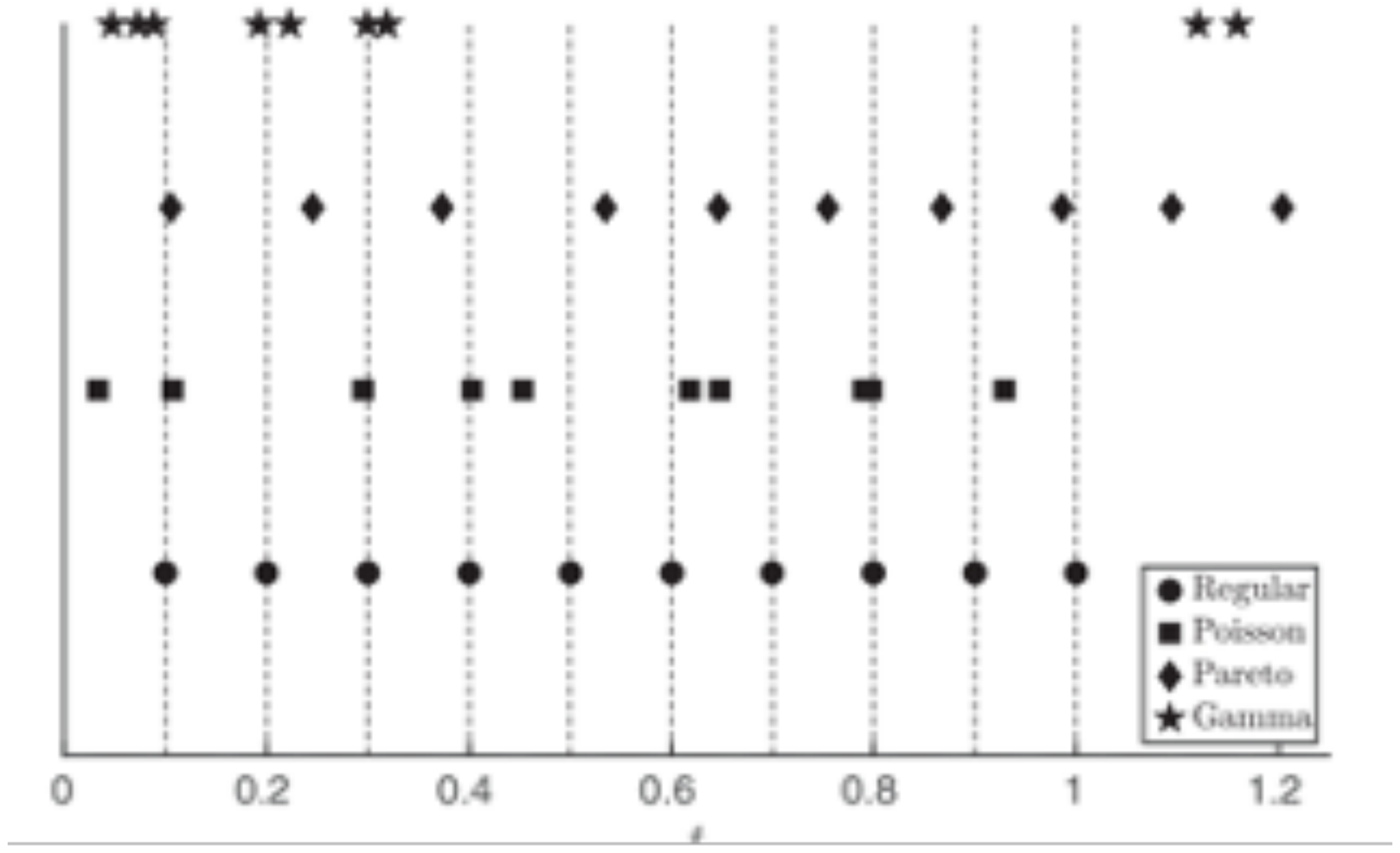
(should be time-independent if stationary)

Sampling

- Even or regular sampling: $y(t) = x(t_0 + n\Delta t)$ where $n = 0,1,\ldots,m$
- Uneven or irregular sampling: $y(t) = x(t_0), \ldots, x(t_m)$
- Regularization/resampling
  - Bin data onto regular grid:

$$y(t) = \frac{\sum_i w_i x_i}{\sum_i w_i} \text{ for } t_i \in [t_a, t_b]$$

  - Interpolate: linear, spline, Gaussian process



- Continuous time process:
  - Observations are a random sample drawn from a continuous process described by some differential equation

Autoregressive models

- Autoregressive models use observations from previous time steps as input to predict the value at the next time step

- Purely random:
  $x_t = z_t$ where $\{z_t\}$ are i.i.d.

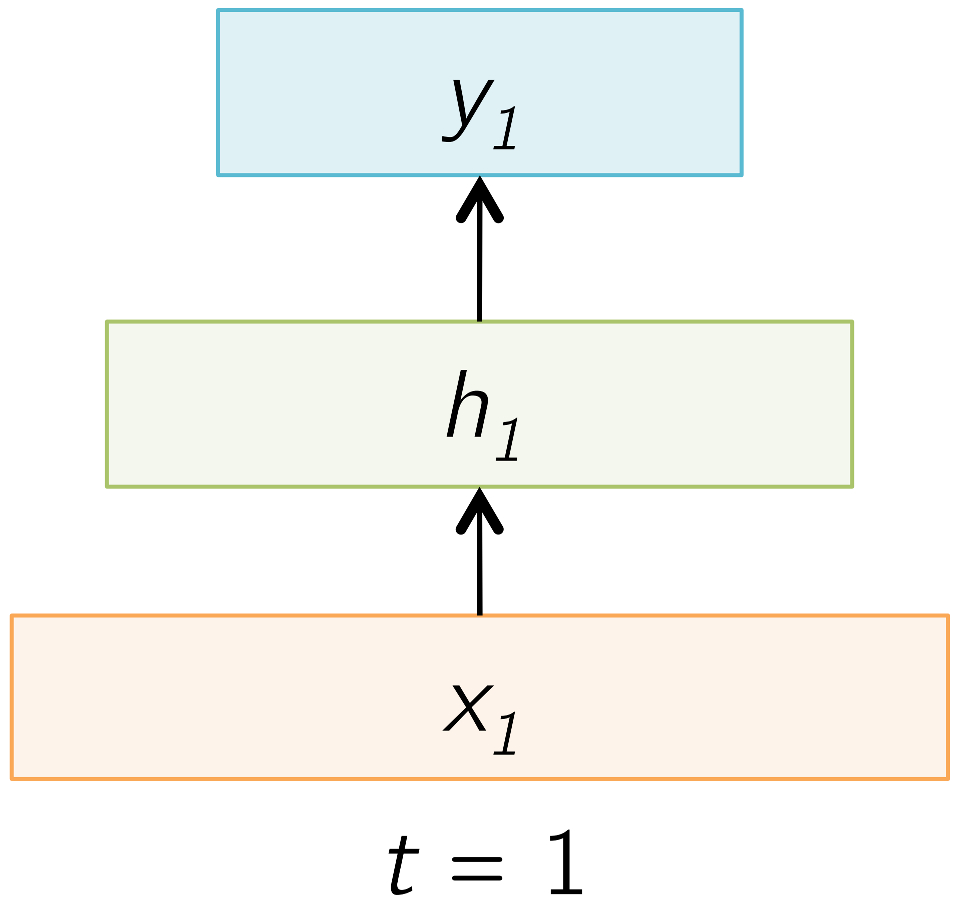
- Random walk (Brownian motion):
  $x_t = x_{t-1} + z_t$

- General autoregressive:
  $x_t = \alpha_1 x_{t-1} + \alpha_2 x_{t-2} + \cdots + z_t$

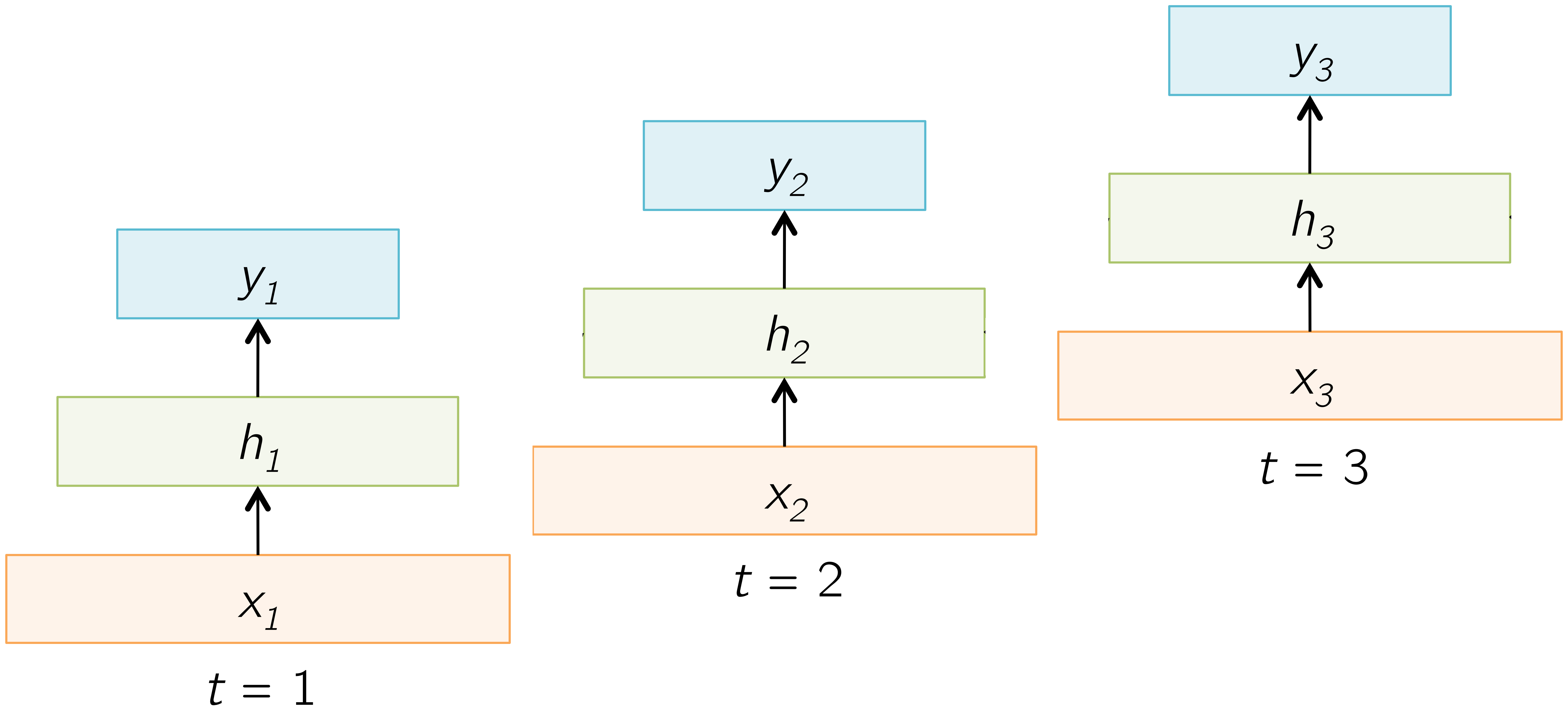
Neural networks for sequential data

- Not all problems can be handled with fixed-length inputs and outputs
- Speech recognition or time-series prediction require a system to store and use context information
  - Example: Output YES if the number of 1s is even, else NO
    - 1000010101 — YES, 100011 — NO, …
  - Hard/impossible to choose a fixed context window
    - There can always be a new sample longer than anything seen

Feed-forward NN

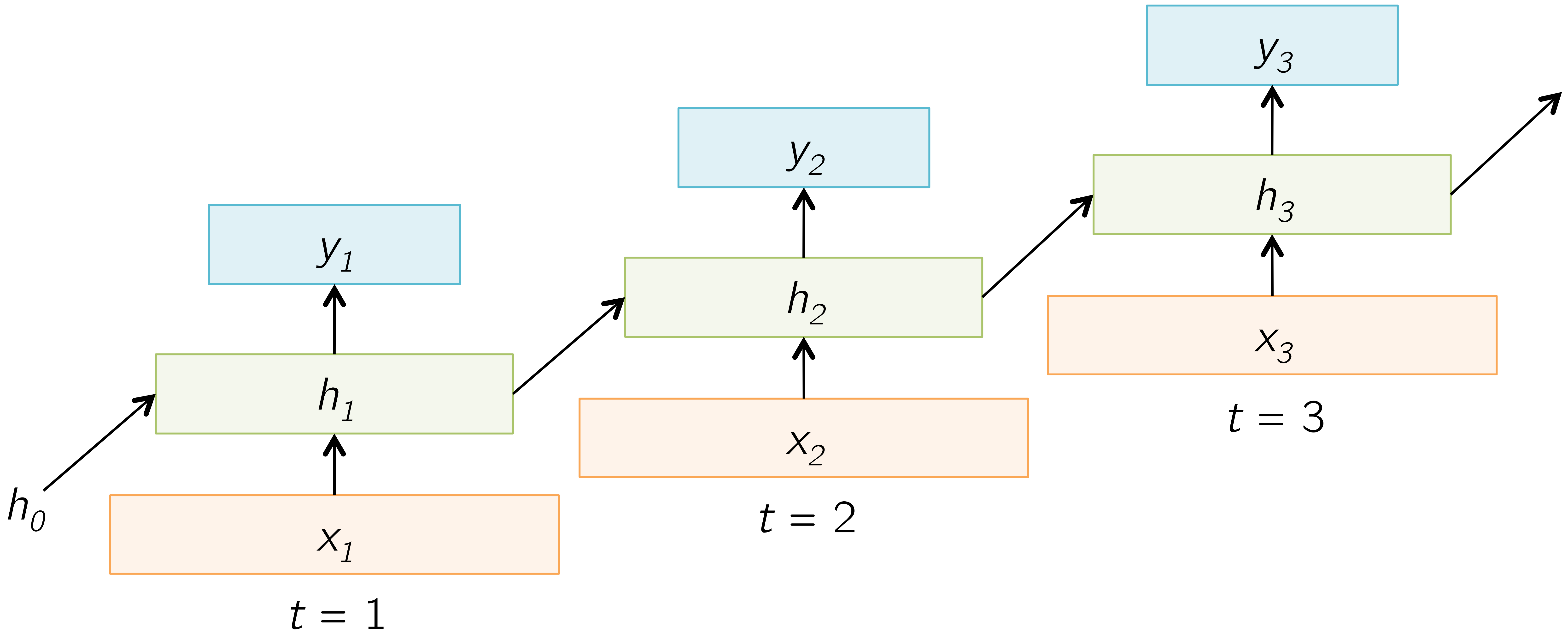


Time-distributed NN



- "Time-distributed" NN shares parameters across time steps
  - Equivalent to 1D CNN with a filter size of 1

Recurrent neural network

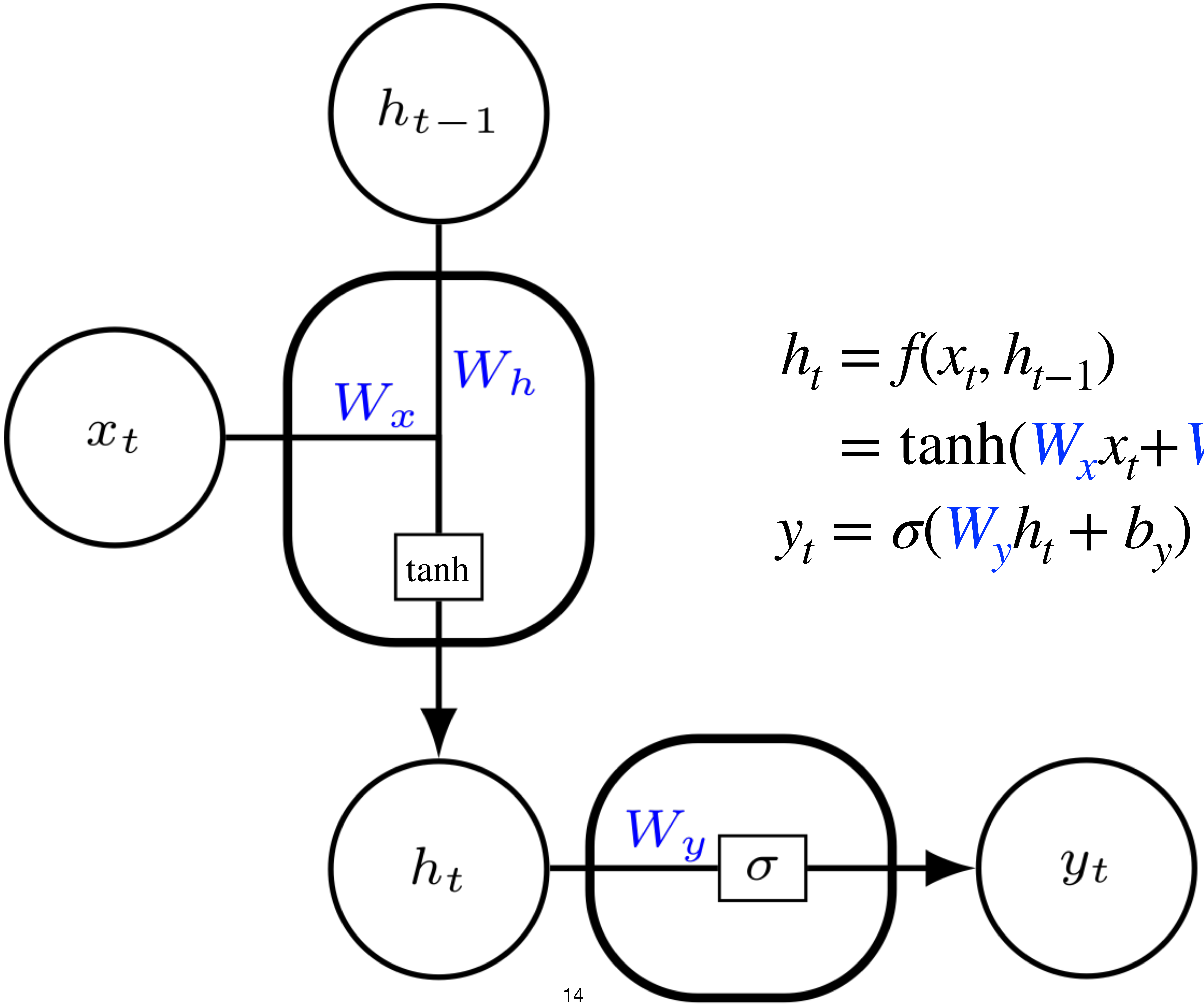


- Recurrent NN considers current input **and** previous hidden state

Recurrent neural network

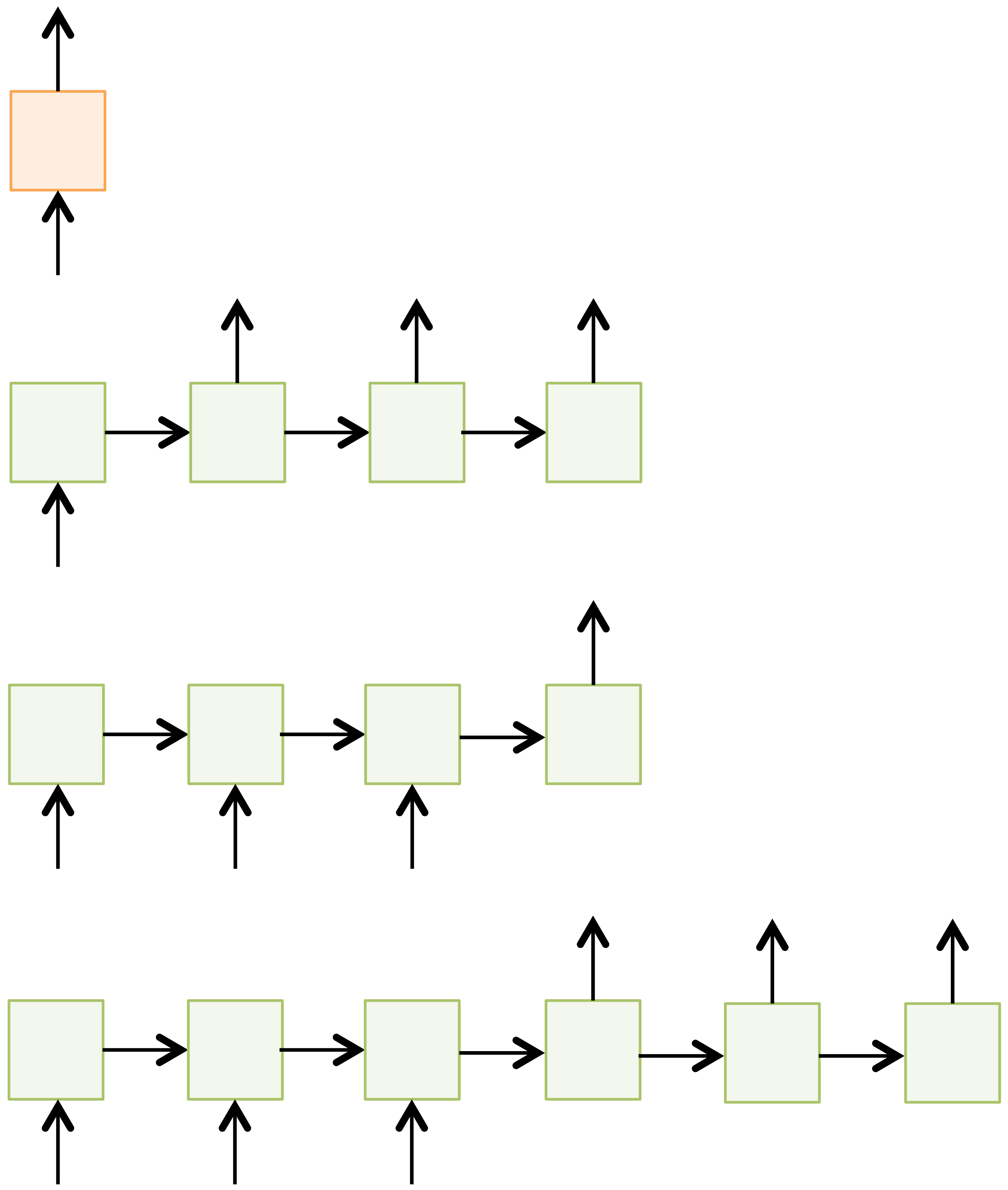
- Recurrent neural networks (RNNs) take the previous output or hidden states as inputs
    - The composite input at time t has some historical information about the happenings at time $T < t$
- RNNs are useful as their intermediate values (state) can store information about past inputs for a time that is not fixed a priori
- Parameters are shared over time steps
- Copies of the RNN cell are made over time (unrolling/unfolding), with different inputs at different time steps

Simple RNN Cell



$$
\begin{aligned}
h_t &= f(x_t, h_{t-1}) \\
&= \tanh(W_x x_t + W_h h_{t-1} + b_h) \\
y_t &= \sigma(W_y h_t + b_y)
\end{aligned}
$$

Input-output scenarios



Single - Single — Feed-forward Network

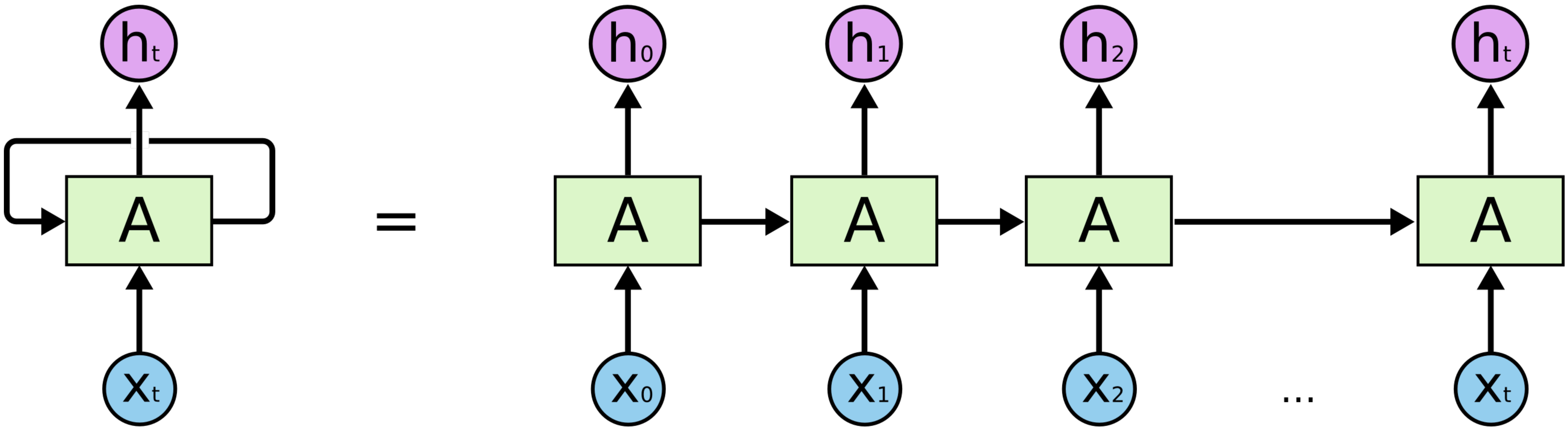
Single - Multiple — Image Captioning

Multiple - Single — Sentiment Classification

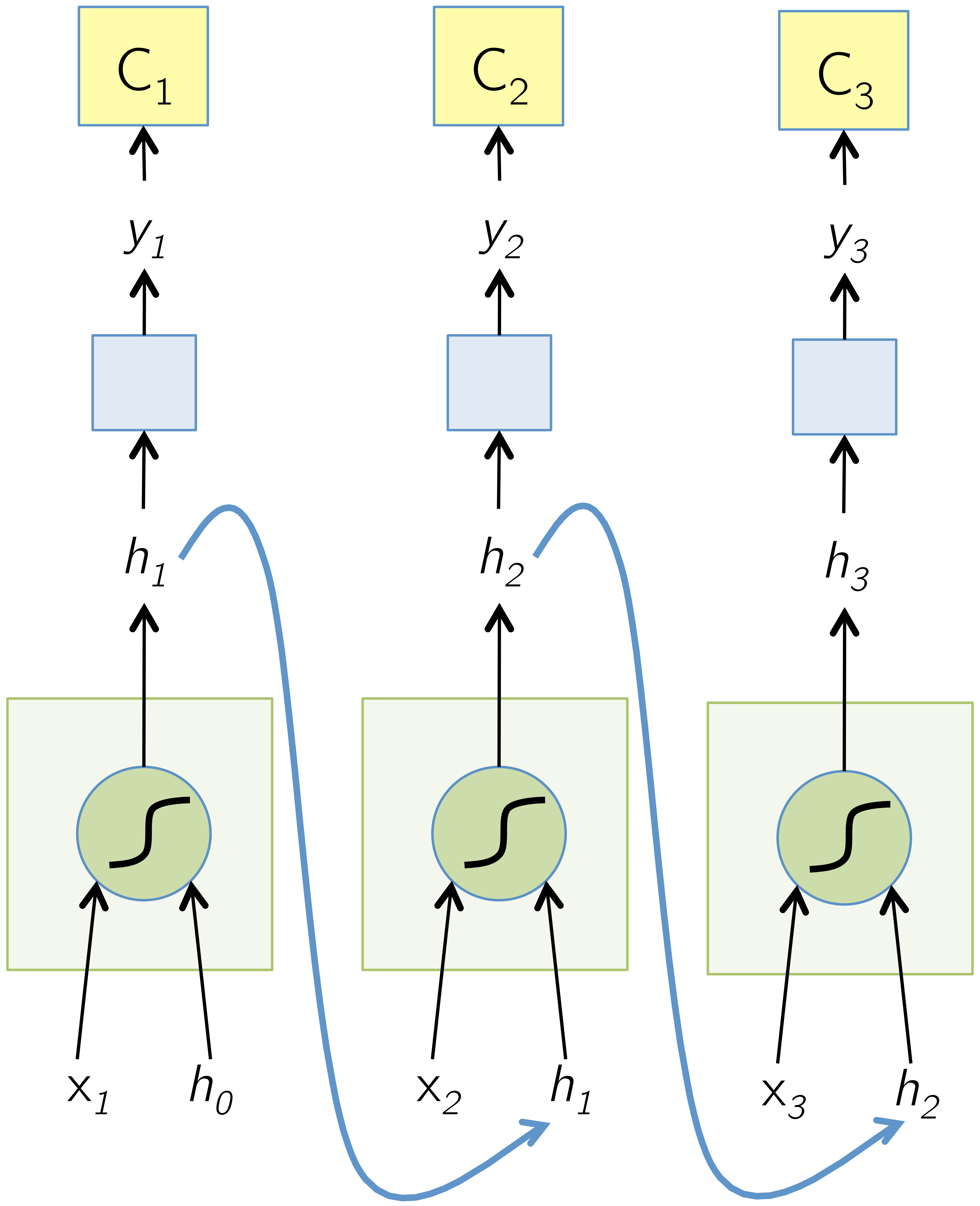
Multiple - Multiple — Translation

Backpropagation through time

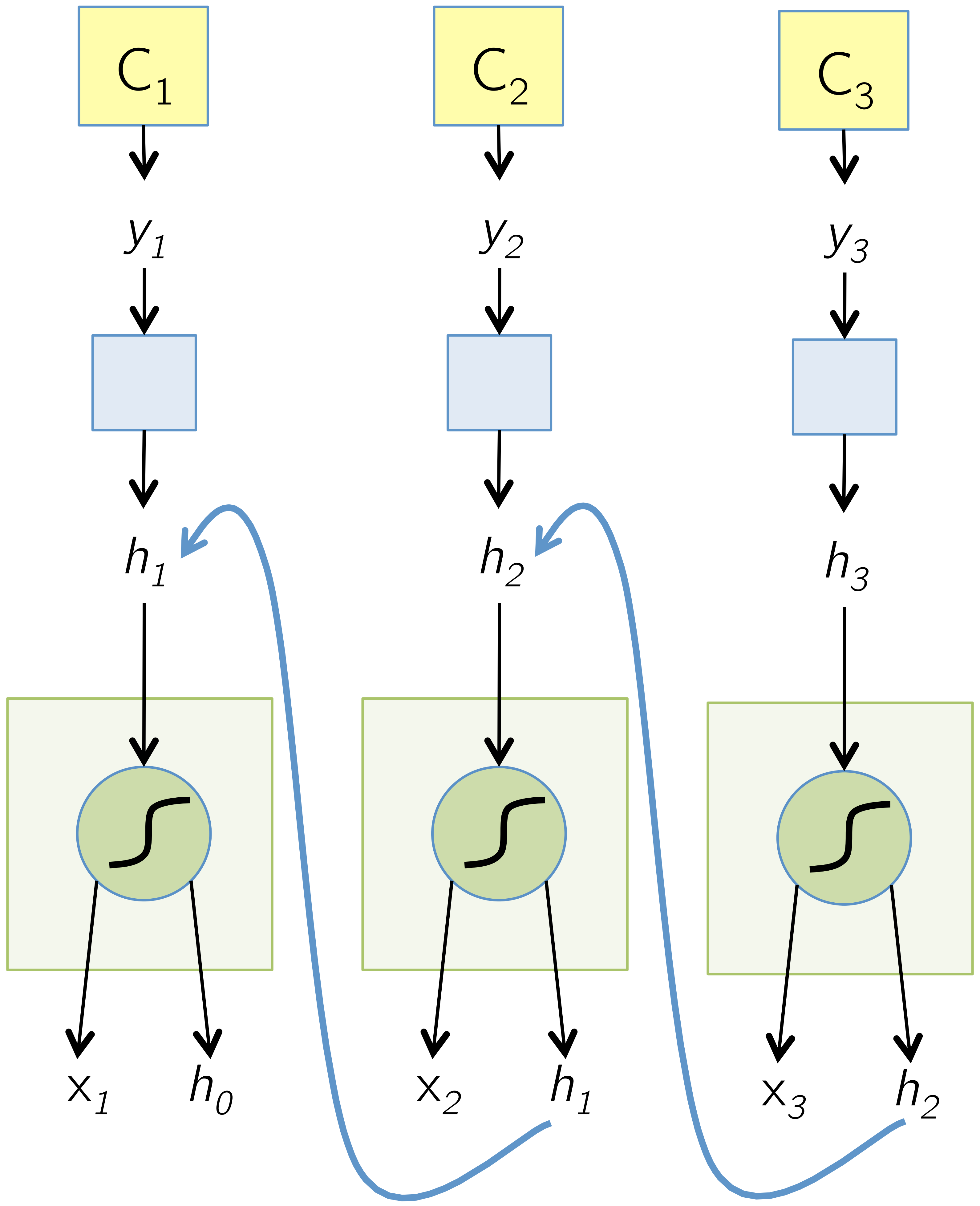
- Method used to train RNNs
- Unfolded network is treated as one big feed-forward network
- This unfolded network accepts the whole time series as input



Simple RNN forward



Simple RNN backward



$$h_t = \tanh(W_x x_t + W_h h_{t-1} + b_h)$$

$$y_t = \sigma(W_y h_t + b_y)$$

$$C_t = \text{Loss}(\bar{y}_t, y_t)$$

$$\frac{\partial C_t}{\partial h_1} = \left(\frac{\partial C_t}{\partial y_t}\right)\left(\frac{\partial y_t}{\partial h_1}\right)$$

$$= \left(\frac{\partial C_t}{\partial y_t}\right)\left(\frac{\partial y_t}{\partial h_t}\right)\left(\frac{\partial h_t}{\partial h_{t-1}}\right)\cdots\left(\frac{\partial h_2}{\partial h_1}\right)$$

Vanishing/exploding gradients

- In the same way a product of k real numbers can shrink to zero or explode to infinity, so can a product of matrices

- It is sufficient for $\lambda_1 < 1/\gamma$, where λ_1 is the largest singular value of W, for the **vanishing gradients** problem to occur and it is necessary for **exploding gradients** that $\lambda_1 > 1/\gamma$, where $\gamma = 1$ for tanh and $\gamma = 1/4$ for sigmoid nonlinearity

- Exploding gradients are often controlled with gradient element-wise or norm clipping

Identity relationship

- Recall

$$\frac{\partial C_t}{\partial h_1} = \left(\frac{\partial C_t}{\partial y_t}\right)\left(\frac{\partial y_t}{\partial h_1}\right)$$

$$= \left(\frac{\partial C_t}{\partial y_t}\right)\left(\frac{\partial y_t}{\partial h_t}\right)\left(\frac{\partial h_t}{\partial h_{t-1}}\right)\cdots\left(\frac{\partial h_2}{\partial h_1}\right)$$

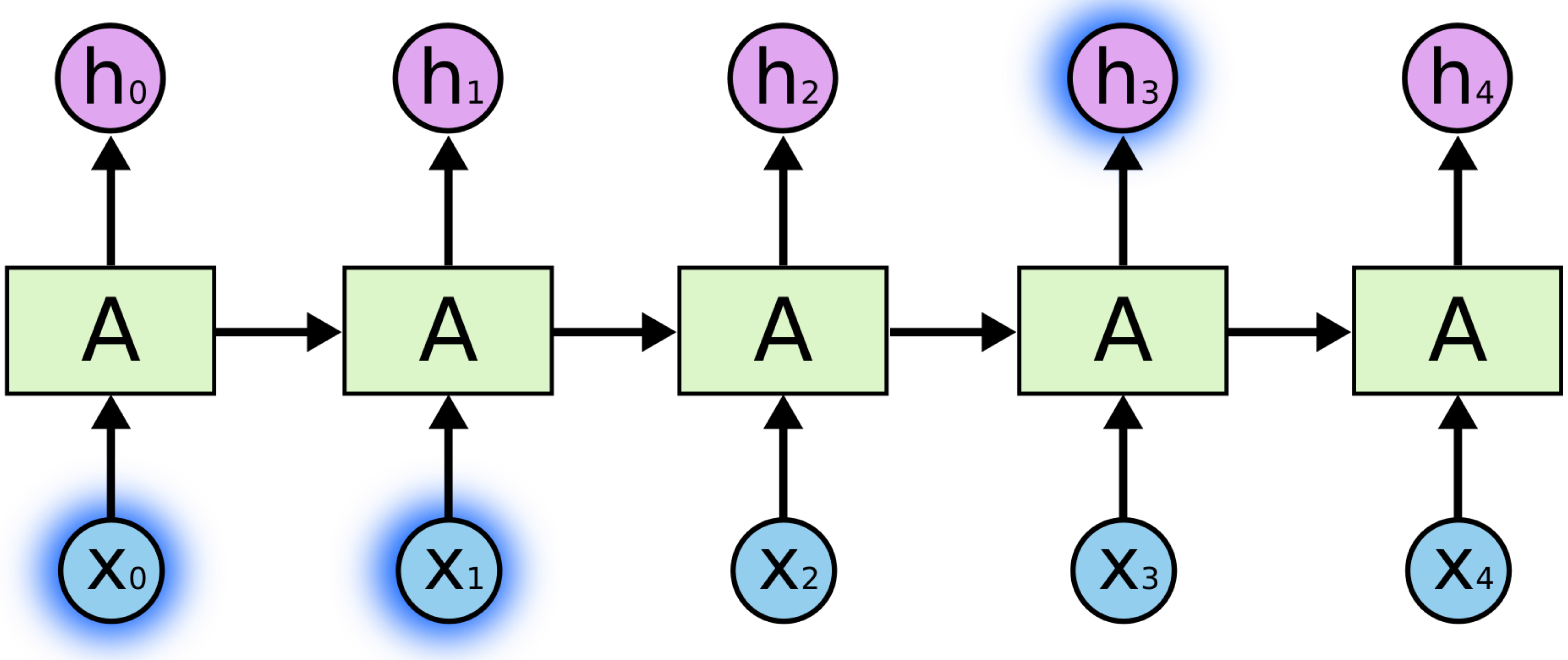
- Suppose we had an identity relationship between hidden states

$$h_t = h_{t-1} + f(x_t) \implies \frac{\partial h_t}{\partial h_{t-1}} = 1$$

Similar to ResNets

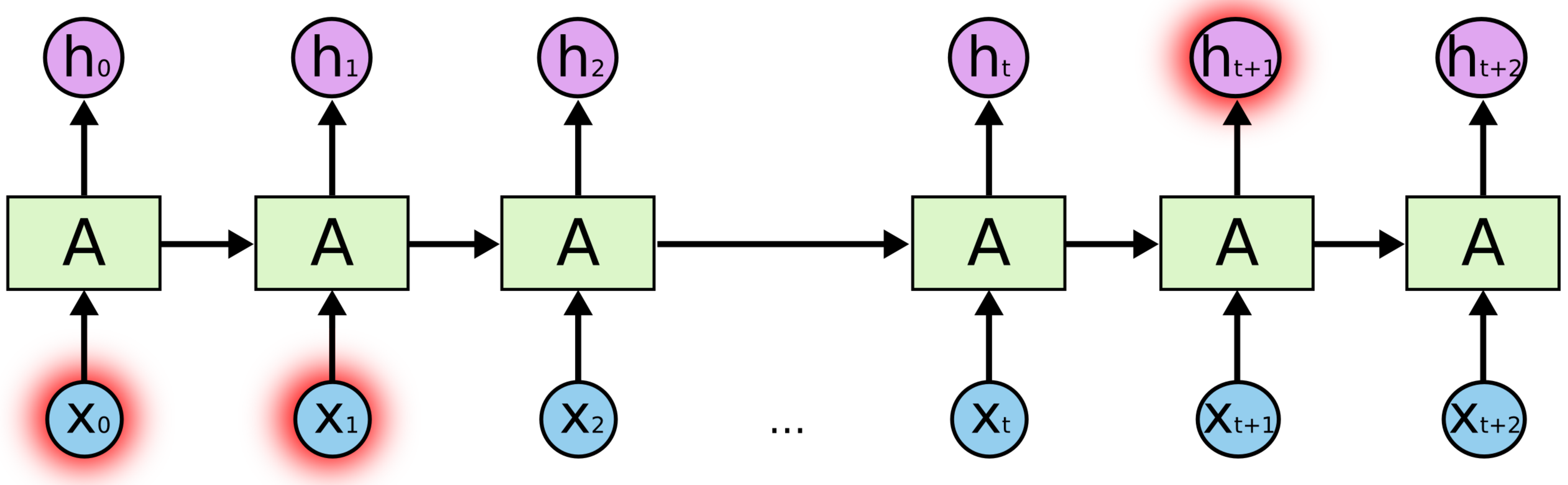
- Gradient does not decay as error is propagated all the way back (“constant error flow”)

Problem of long-term dependencies



- For small gaps, simple RNNs can learn to use past information, e.g. predicting the last word in “the clouds are in the *sky*”

Problem of long-term dependencies

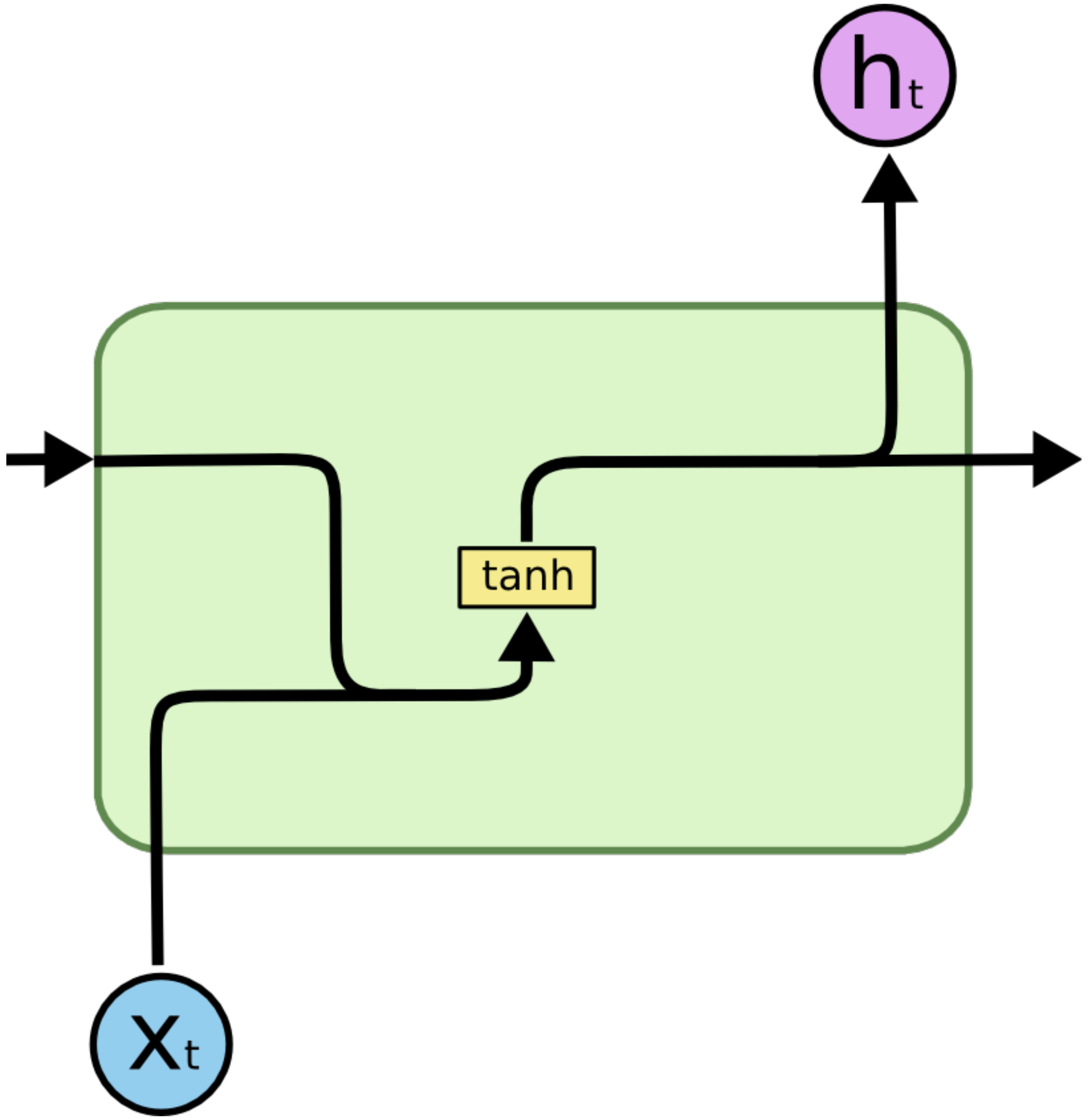


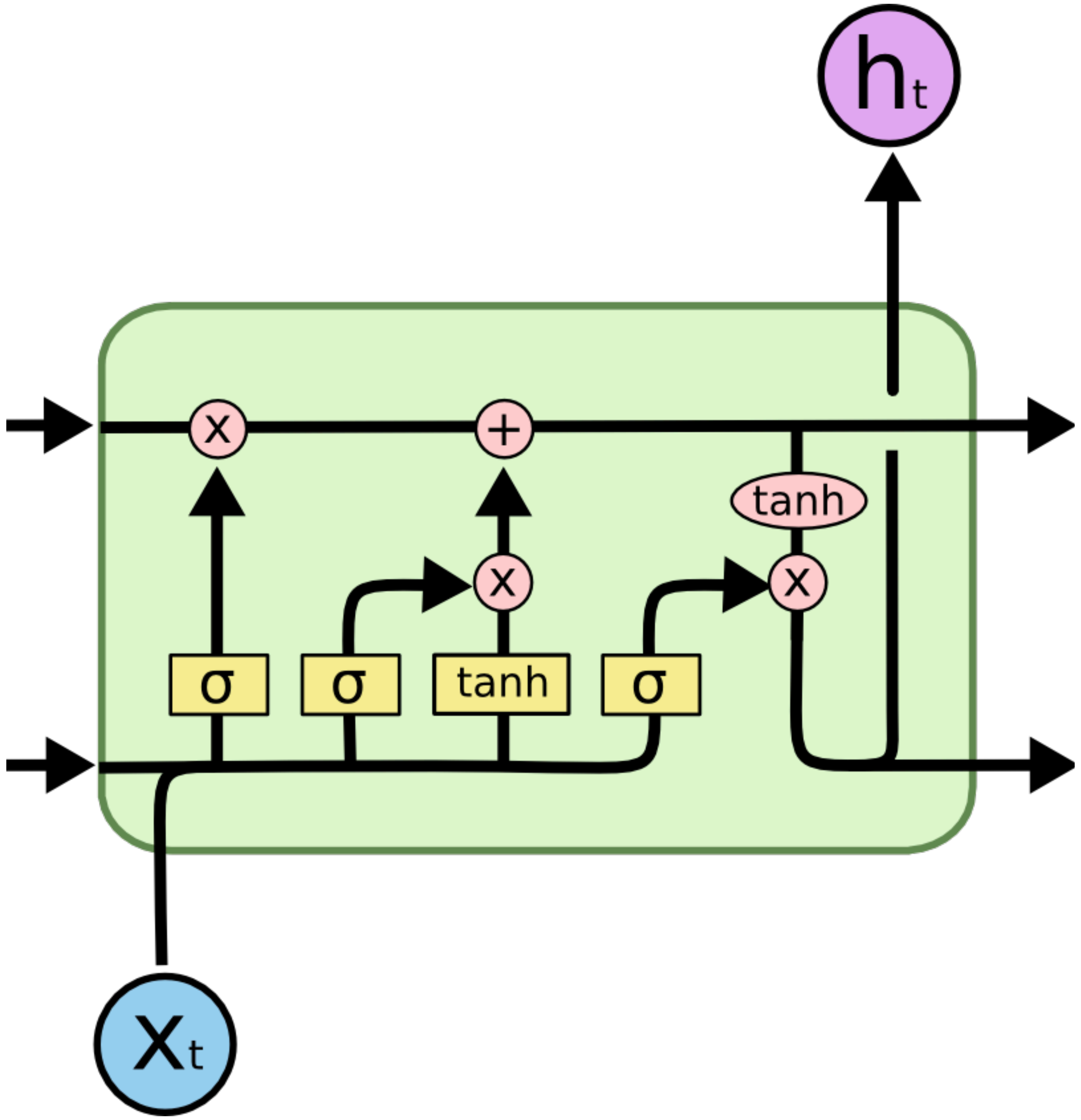
- As the gap grows, RNNs become unable to learn to connect the information in practice

Long short-term memory (LSTM)

- Developed to cope with the issue of long-term dependencies [10.1162/neco.1997.9.8.1735]
- LSTM uses this idea of “constant error flow” to ensure that gradients don’t decay
- Key components are
  - an internal memory (“cell state”)
  - gates that control the cell state actively

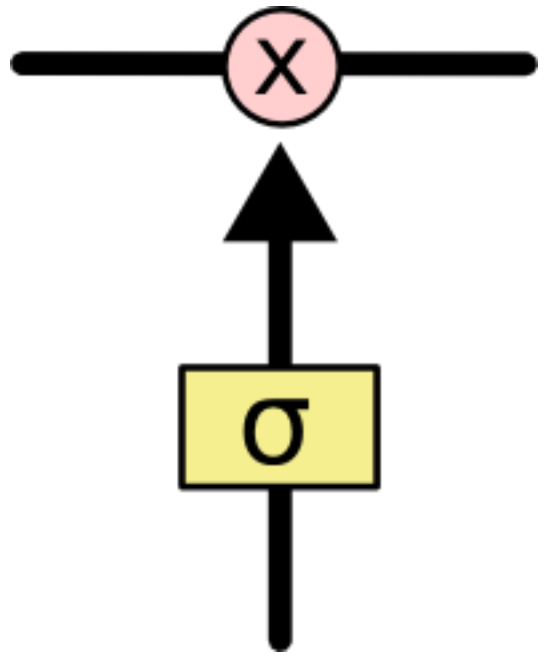
Simple RNN vs. LSTM

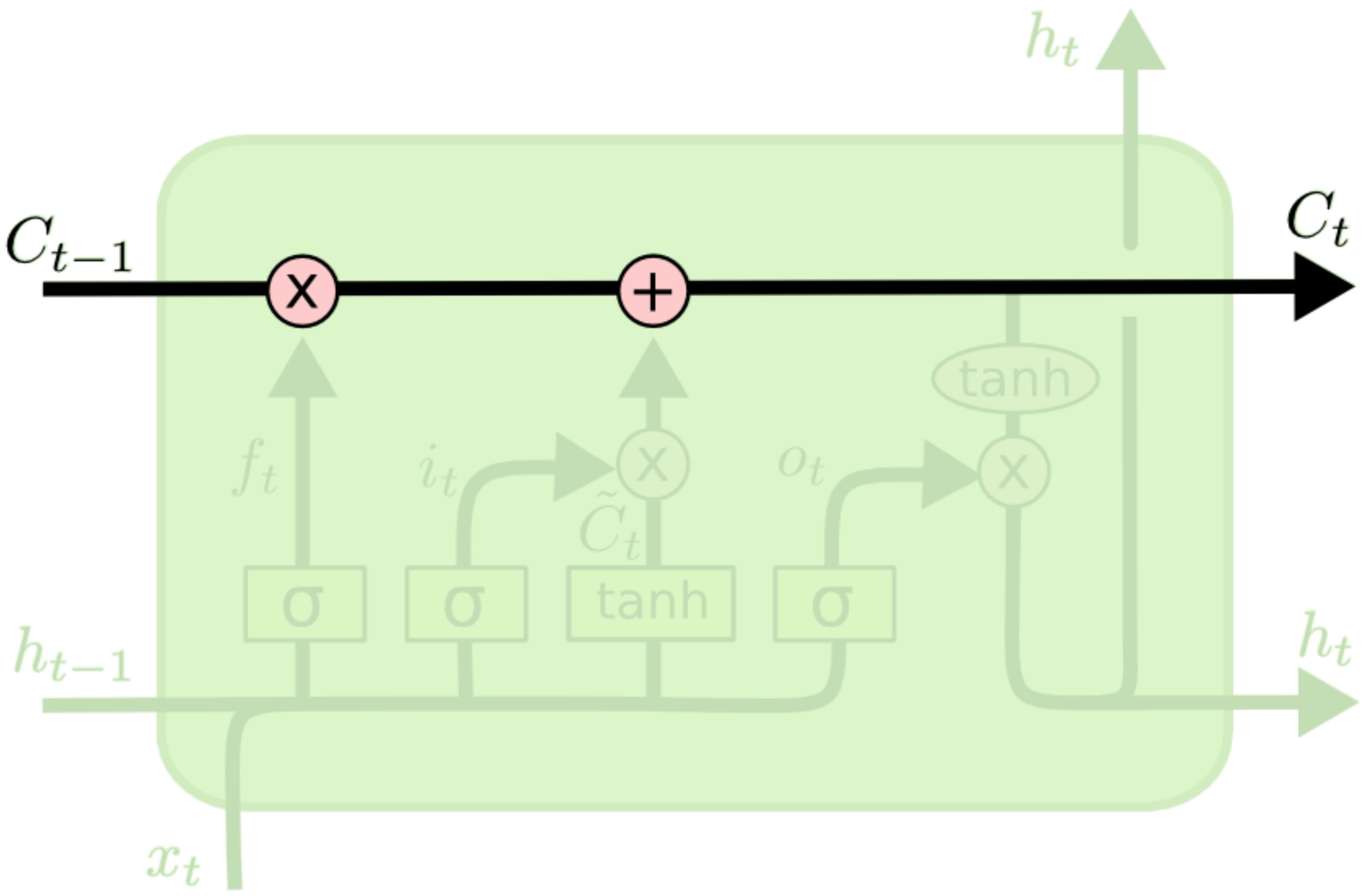




Cell state

- Cell state is like a conveyer belt: runs straight down the entire chain, with only some minor linear interactions
- Gates optionally let information through: sigmoid outputs numbers between 0 and 1, describing how much should be let through

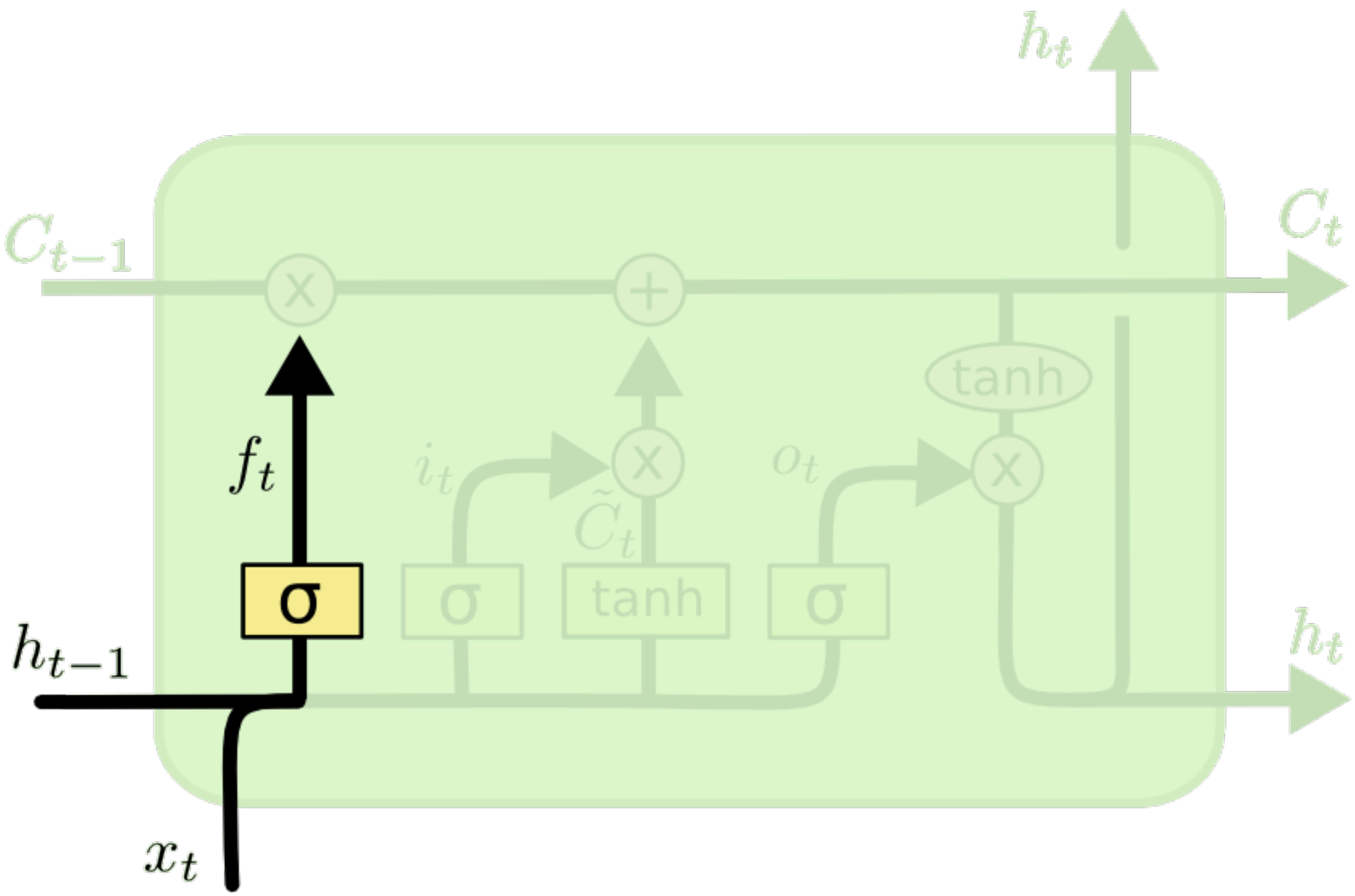




Forget gate layer

- Forget gate layer controls how much information to throw away from the cell state

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$$

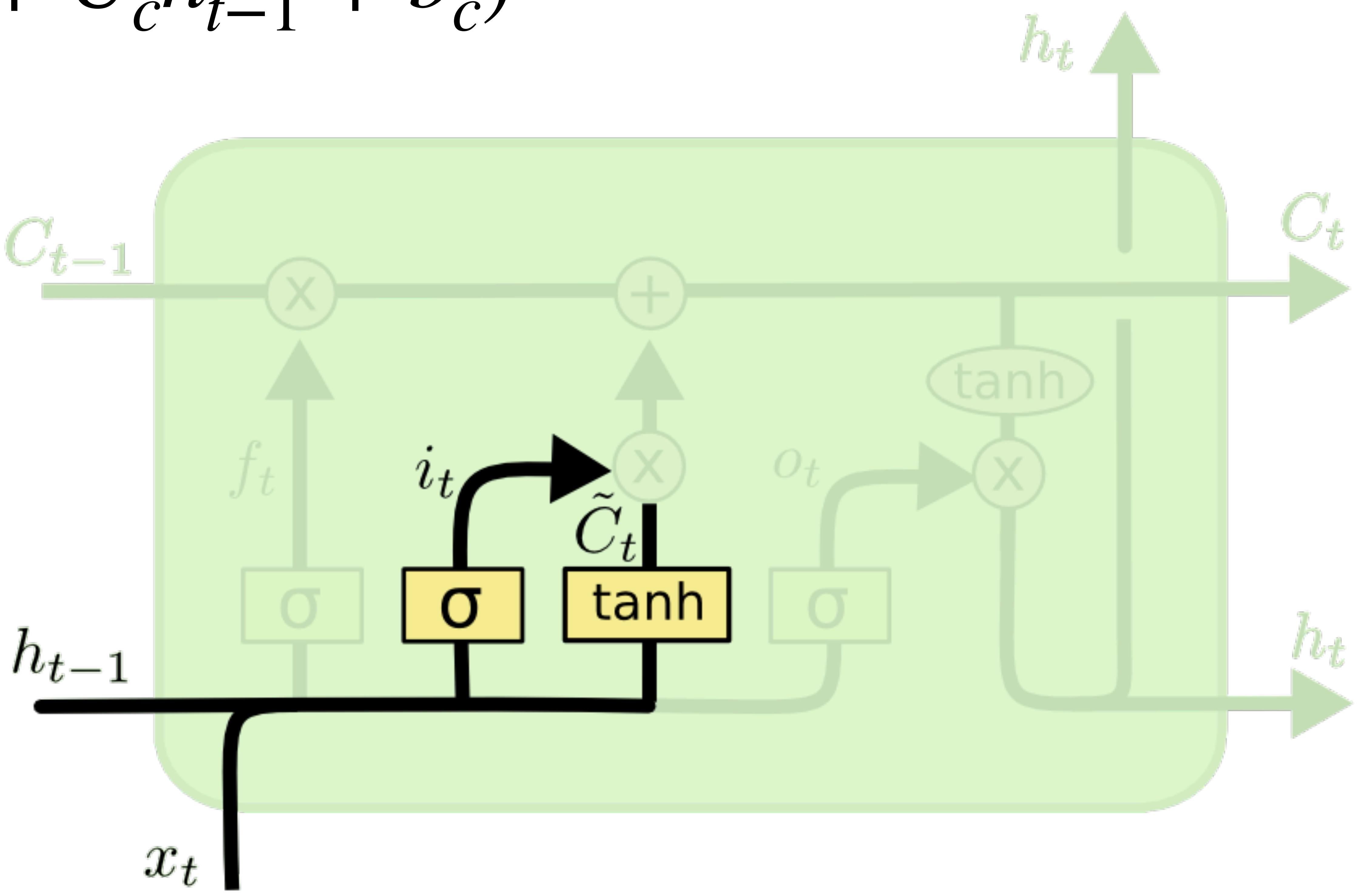


Input gate layer

- Input gate layer controls whether a new candidate value $\tilde{C}_t$ flows into the cell state

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$

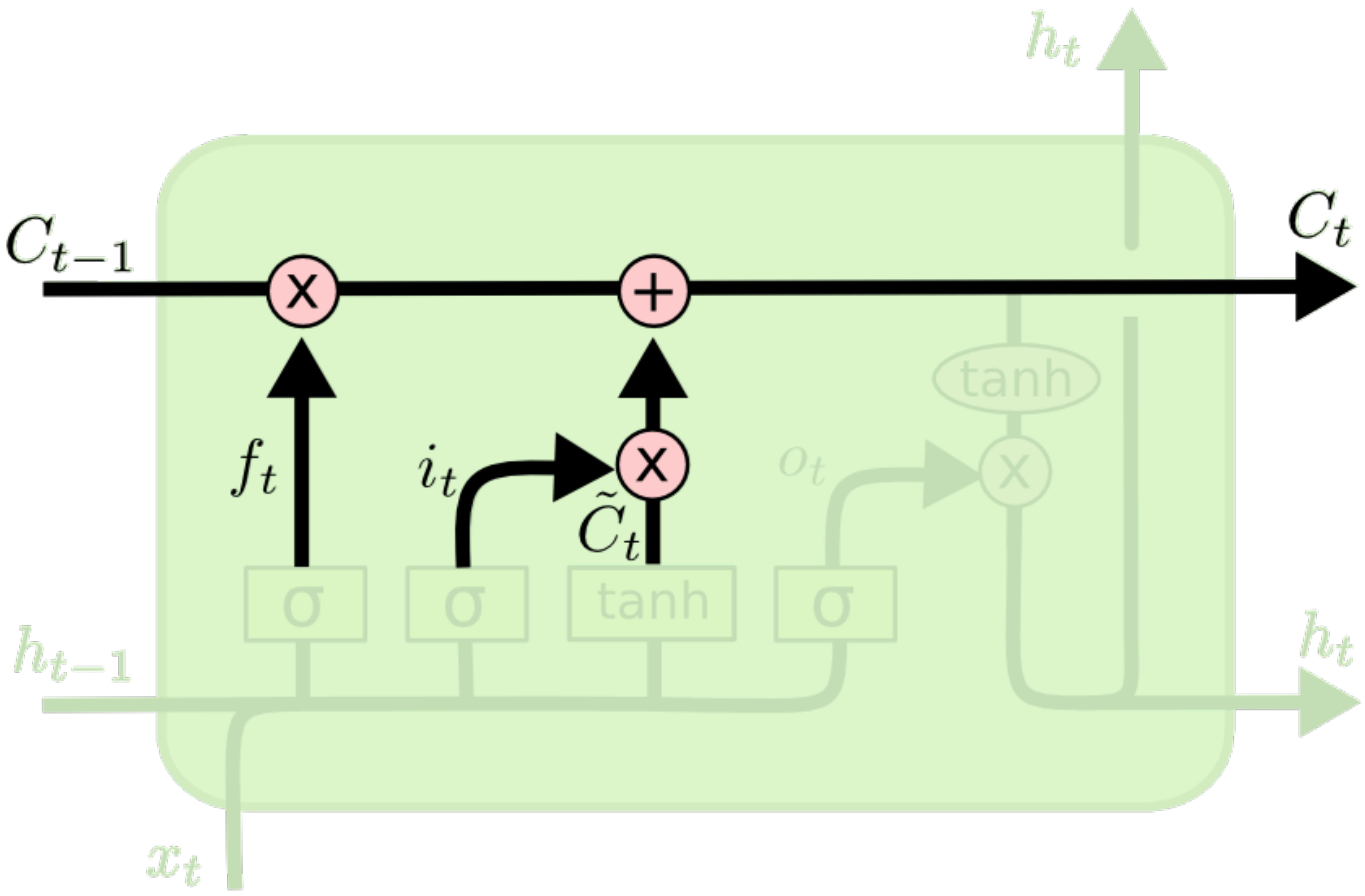
$$\tilde{C}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c)$$



Cell state update

- Cell state is updated using forget and input gates

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

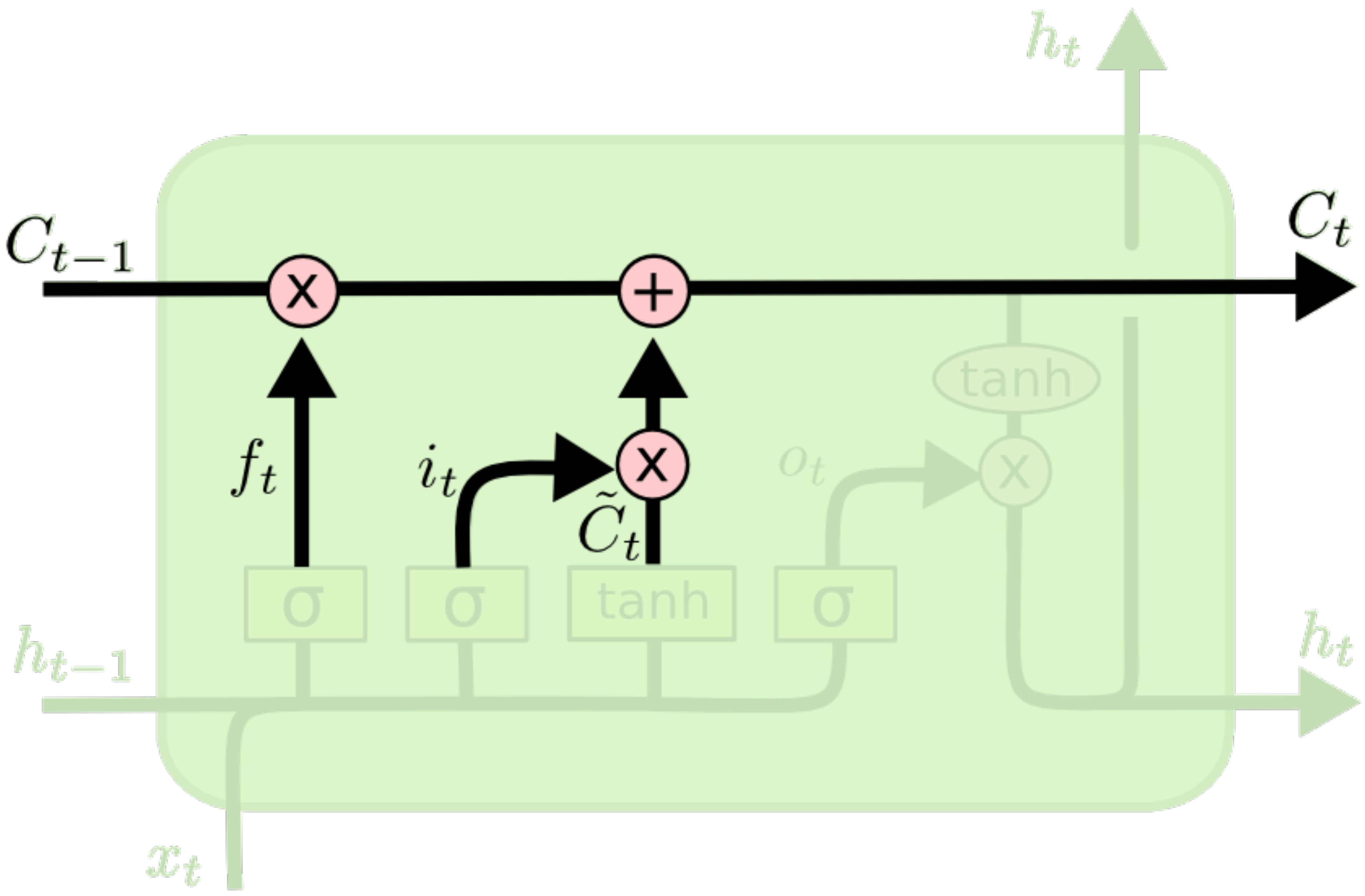


Output gate layer

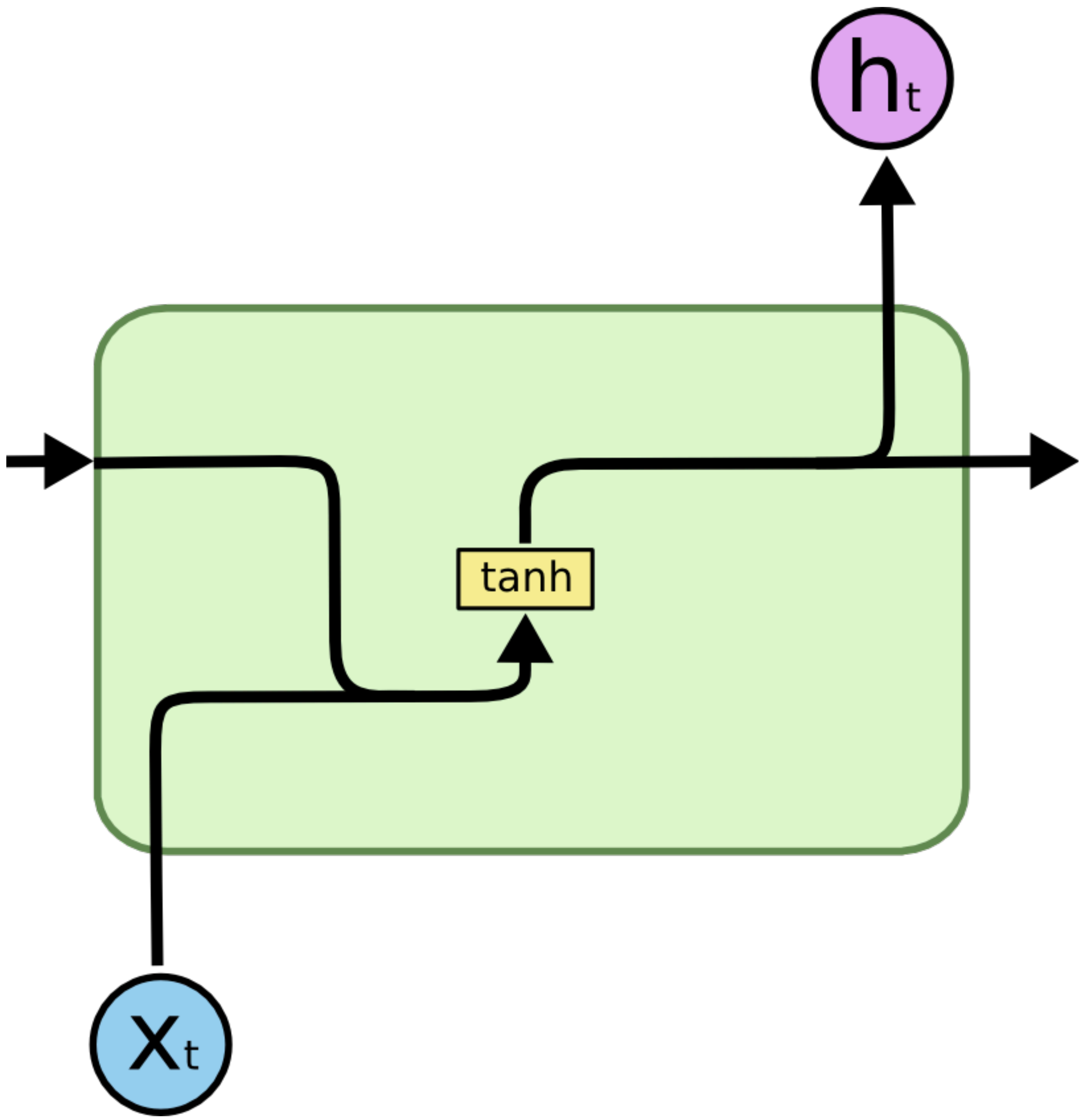
- Finally, the output gate layer controls how the updated cell value contributes to the hidden state

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$$

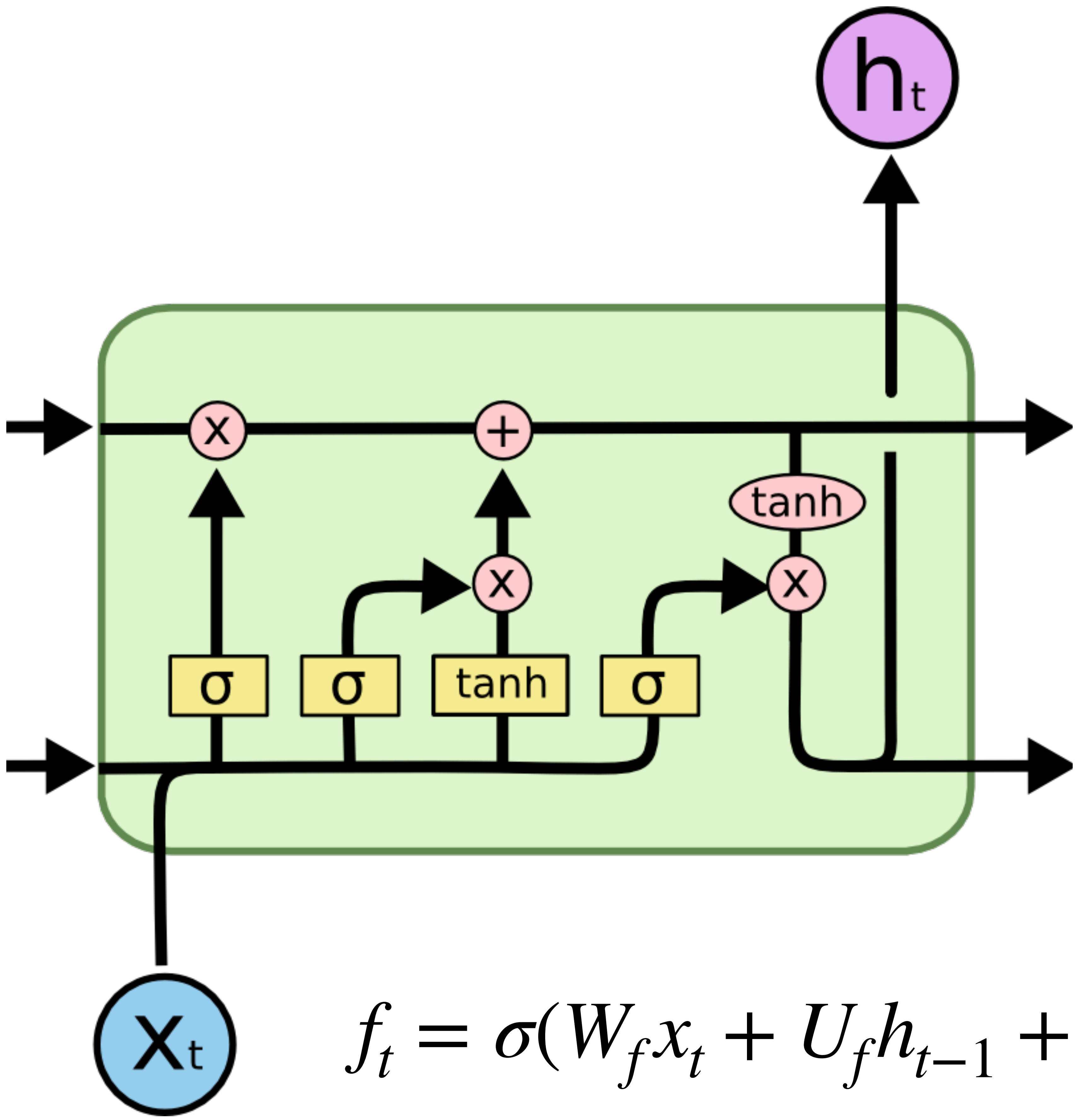
$$h_t = o_t \cdot \tanh(C_t)$$



Simple RNN vs. LSTM



$$h_t = \tanh(W_f x_t + U_f h_{t-1} + b_f)$$



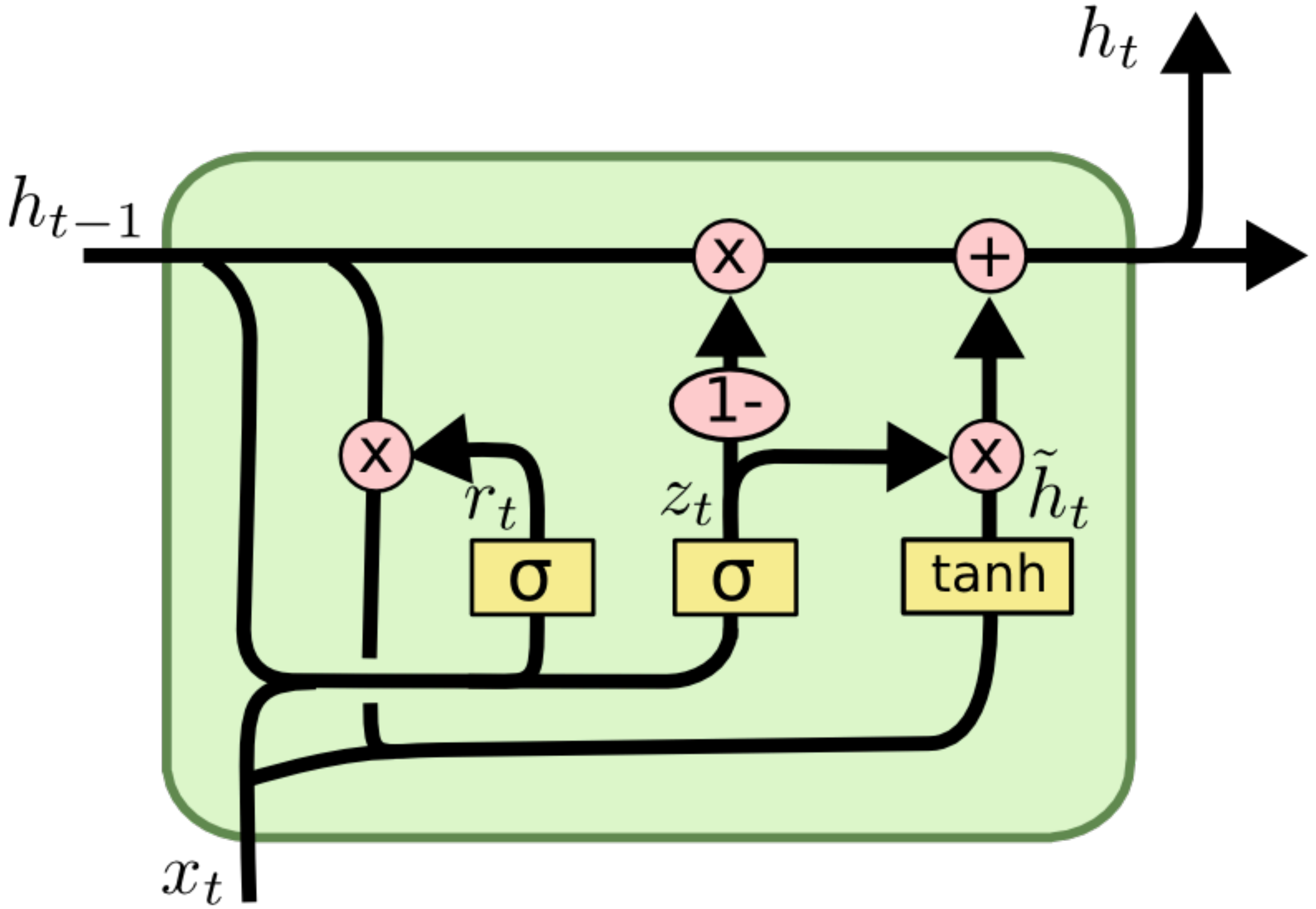
$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$$
$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$
$$\tilde{C}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c)$$
$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$$
$$h_t = o_t \cdot \tanh(C_t)$$

LSTM Variants

- Many variants of LSTM spurred by questions of the architecture
  - Does it need to be this complicated? Can it be simplified?
  - Should forget and input gates be related somehow?
  - What is the point of having separate cell and hidden states?

Gated recurrent unit

- Gated recurrent unit (GRU) [arXiv:1406.1078]
  - Combines forget and input gates into a single “update gate”
  - Merges cell and hidden state



$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r)$$
$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z)$$
$$\tilde{h}_t = \tanh(W_z x_t + U_h(r_t \cdot h_{t-1}) + b_h)$$
$$h_t = (1 - z_t) \cdot h_{t-1} + z_t \cdot \tilde{h}_t$$

Next time

- More on recurrent neural networks
- Applications
- Hands-on