Recurrent Neural Networks

Scratching the Surface

Youssef Beauferris
Teaching Assistant
Electrical and Computer Engineering
Schulich School of Engineering

March 2021

UNIVERSITY OF CALGARY

Outline

- Sequence Modeling
- Traditional Recurrent Neural Networks (RNNs)
- Back Propagation Through Time (BPTT)
- Problems with Long-term Dependencies
- Long short-term memory (LSTMs)

Additional Resources

- Stanford cheatsheet:
  - https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks
- Colah's blog:
  - https://colah.github.io/posts/2015-08-Understanding-LSTMs/
- Illustrated guide of LSTM:
  - https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

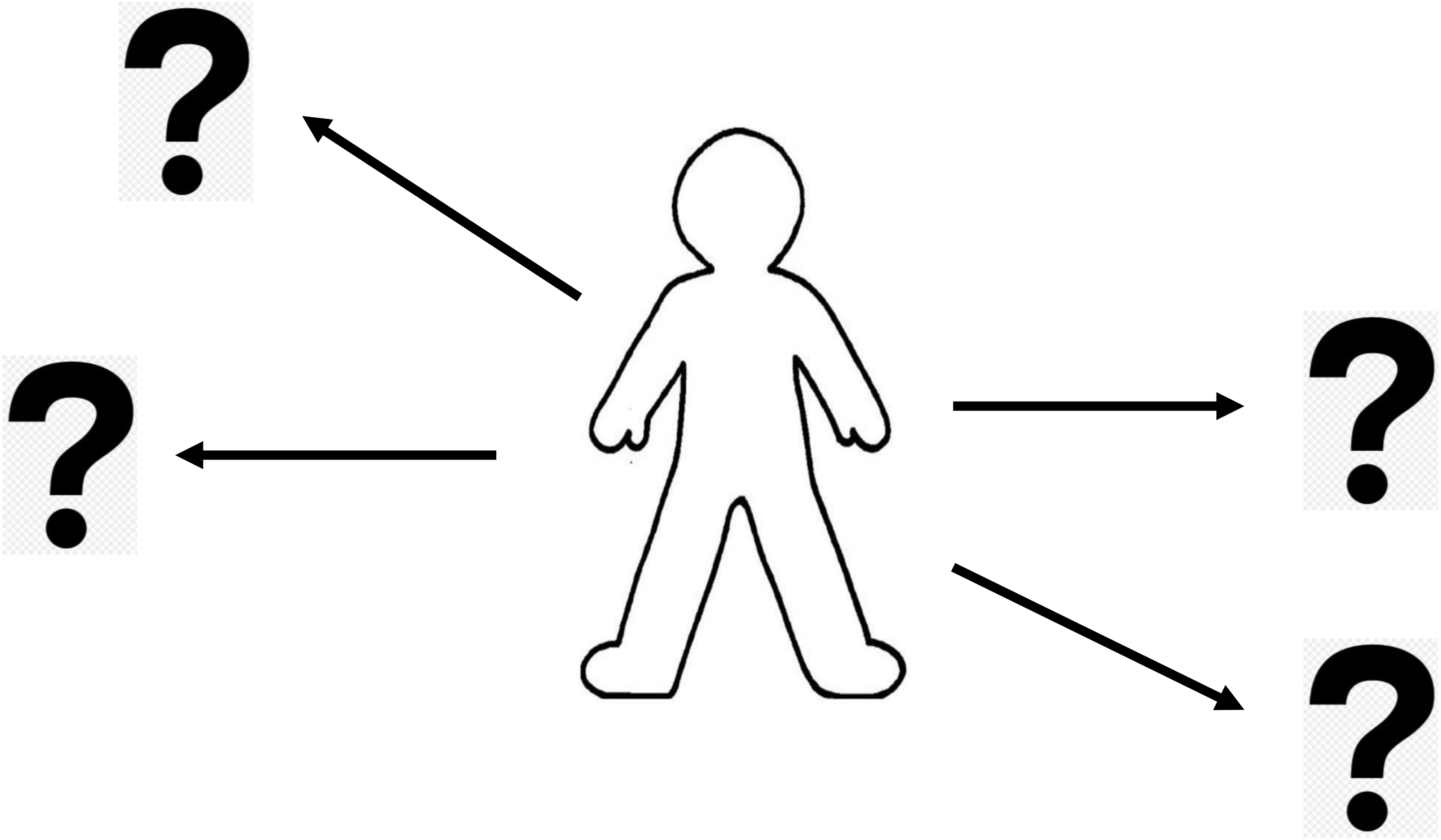
Sequential Modelling

Introduction

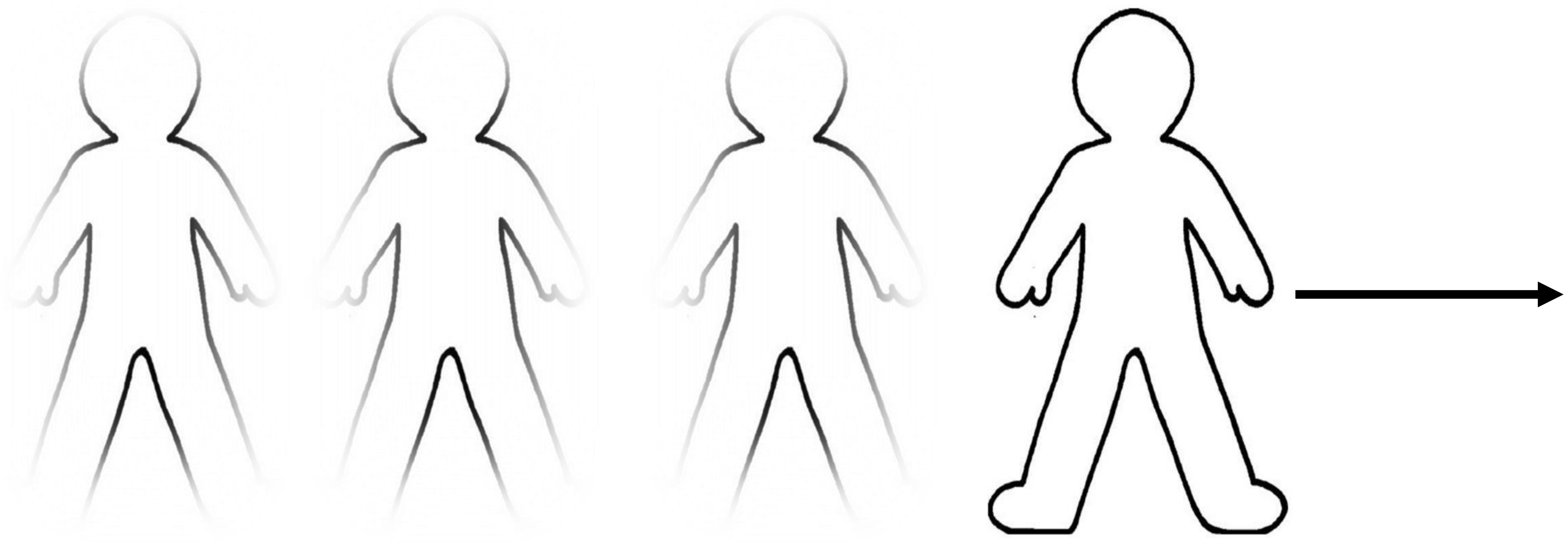
- Building models of sequential data is important: automatic speech recognition, machine translation, natural language, …

- Recurrent neural networks (RNNs) are a simple and general framework for this type of tasks

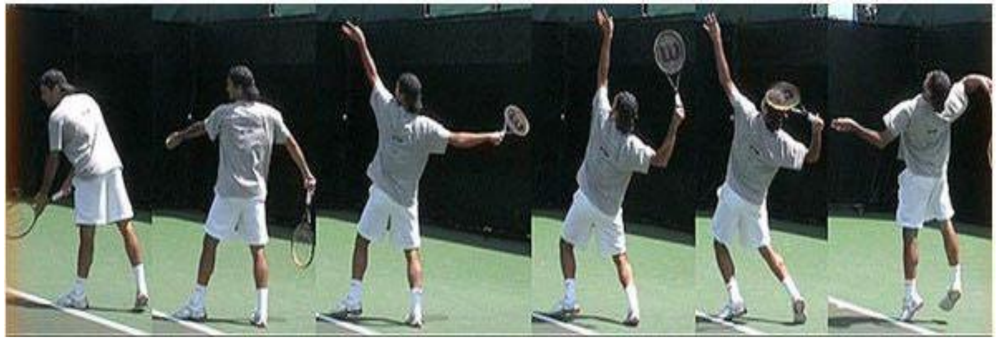
Can you predict where this person is going?

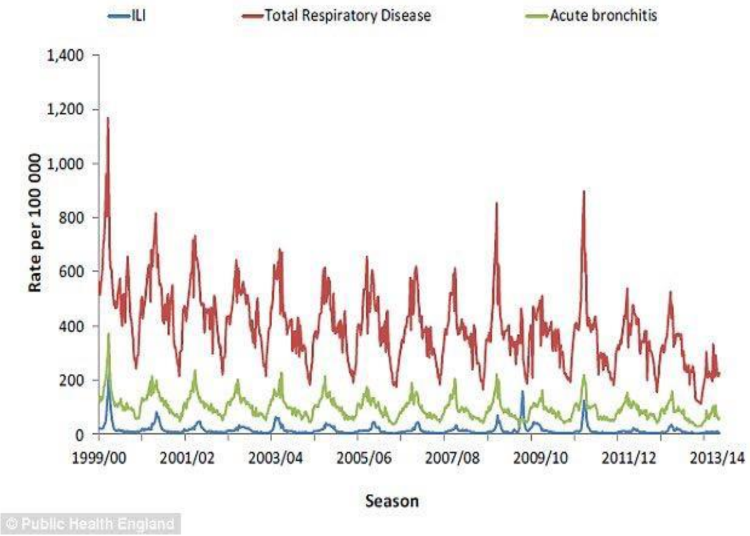


What if you know the persons previous location?

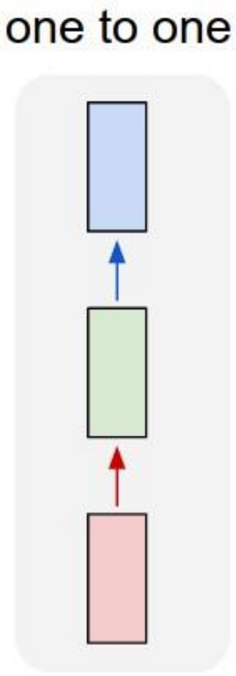


Data is often sequential in nature





Sequence Modeling Applications



Binary Classification

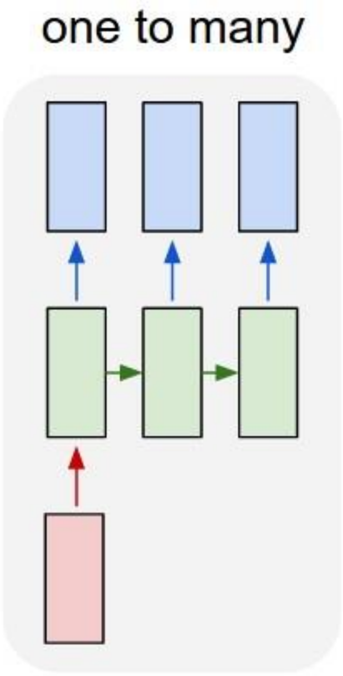
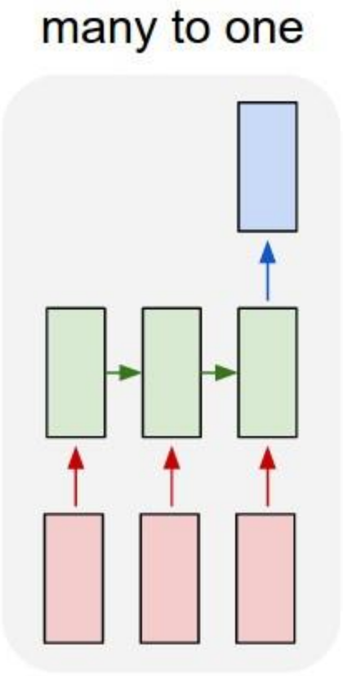
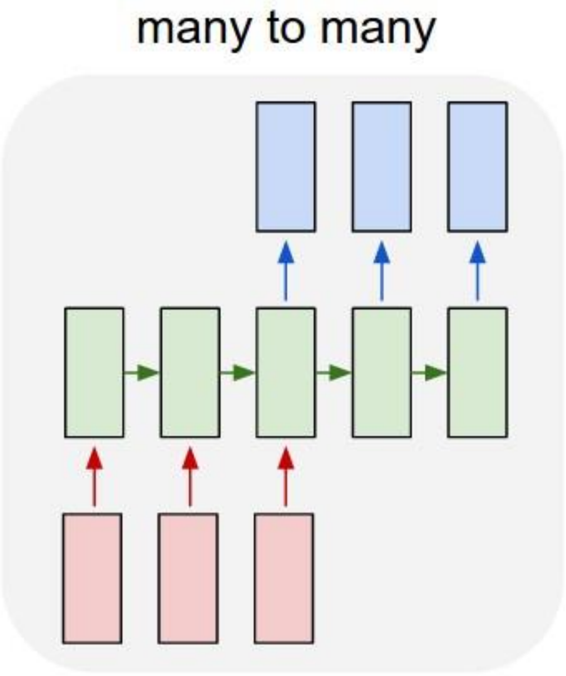


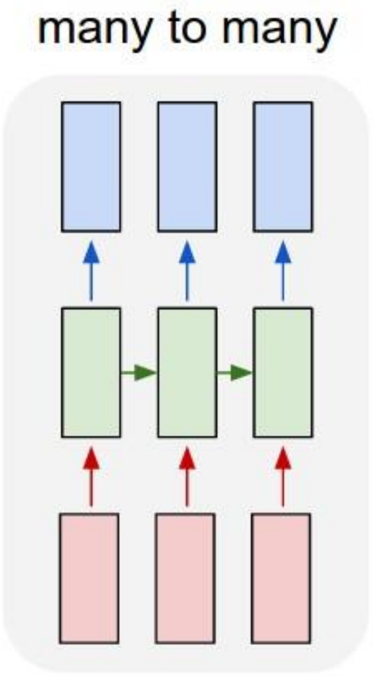
Image Caption



Sentiment Classification



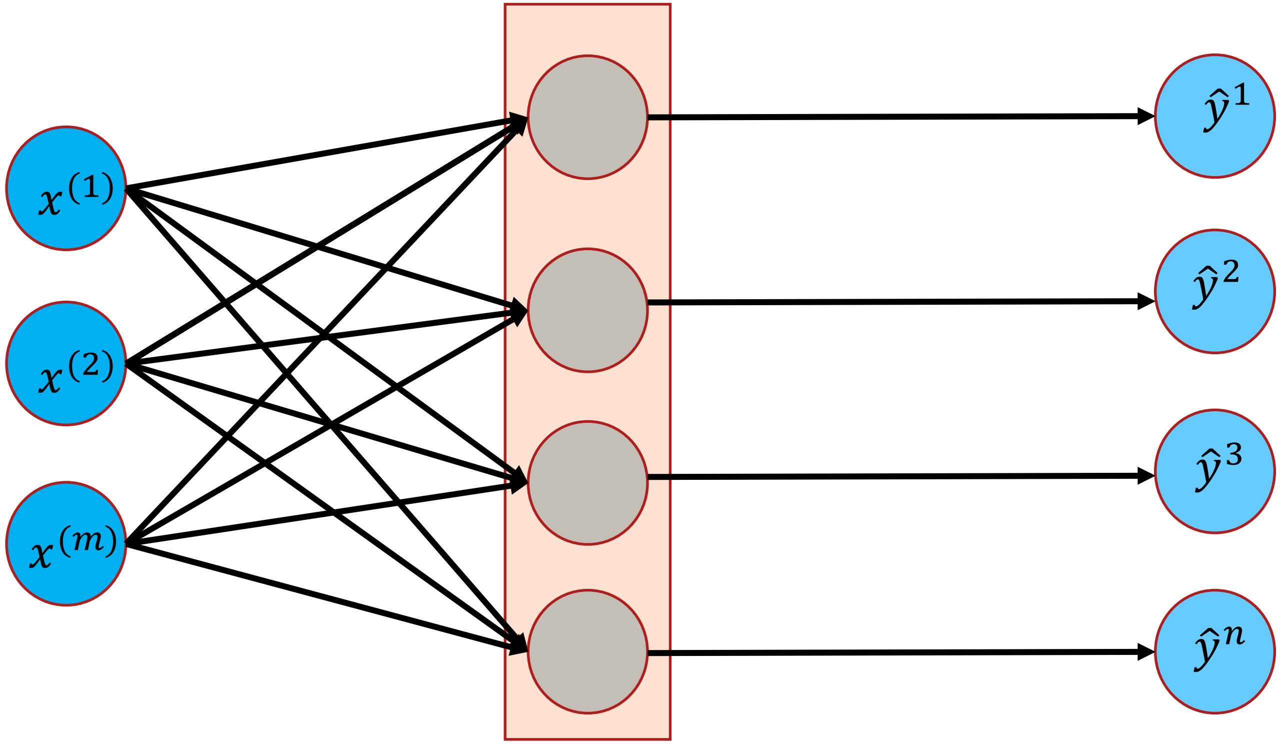
Machine Translation



Video Prediction

Traditional Recurrent Neural Networks

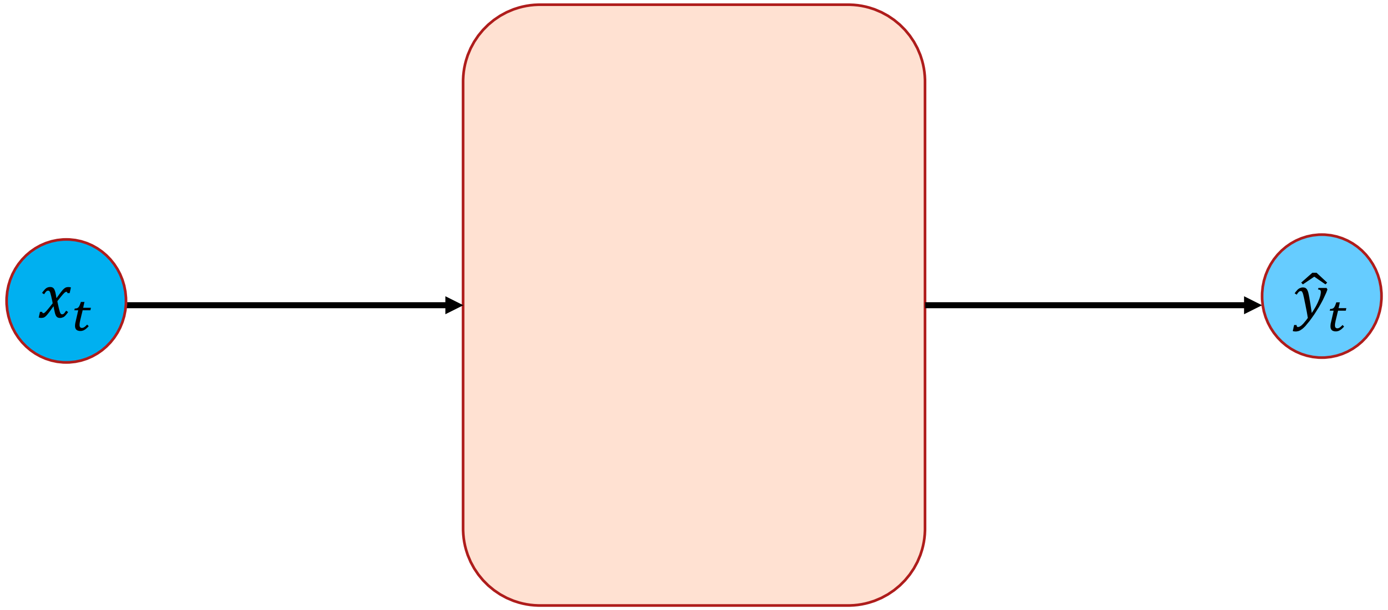
Feed Forward Networks



$\boldsymbol{x} \in \mathbb{R}^m$

$\hat{y} \in \mathbb{R}^n$

Feed Forward Networks

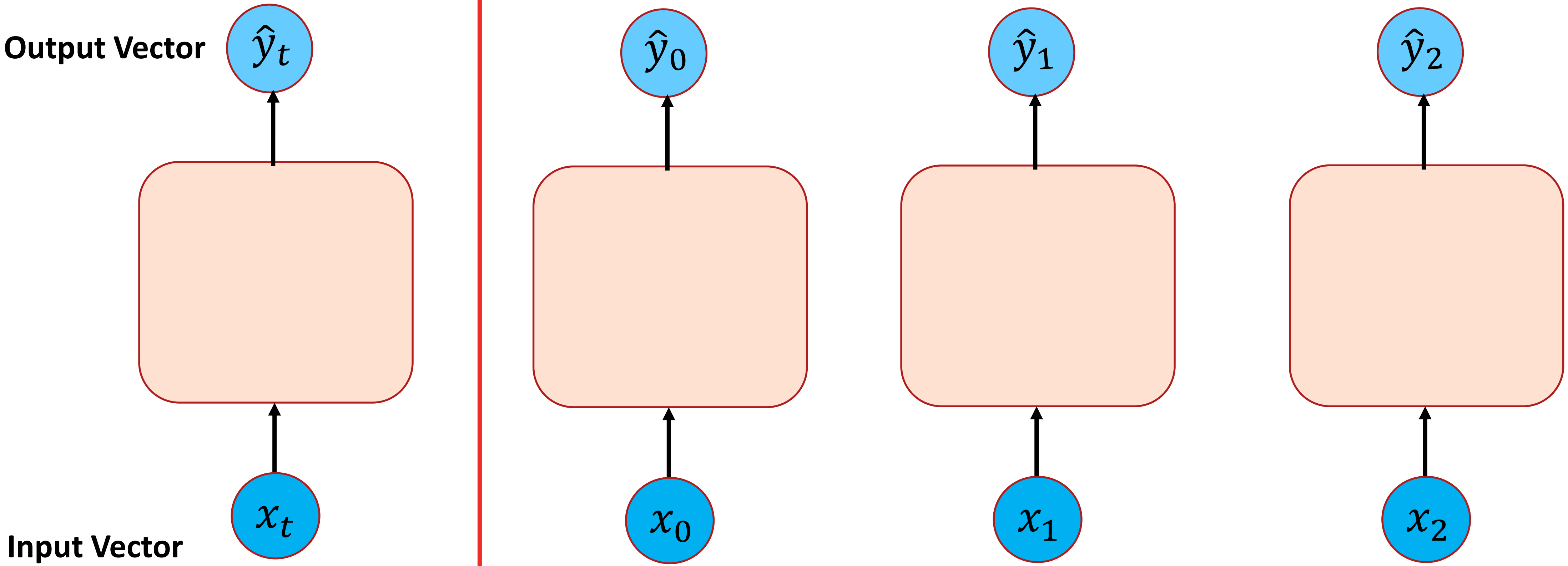


$\boldsymbol{x}_t \in \mathbb{R}^m$

$\hat{\boldsymbol{y}}_t \in \mathbb{R}^n$

Handling Different Time-points

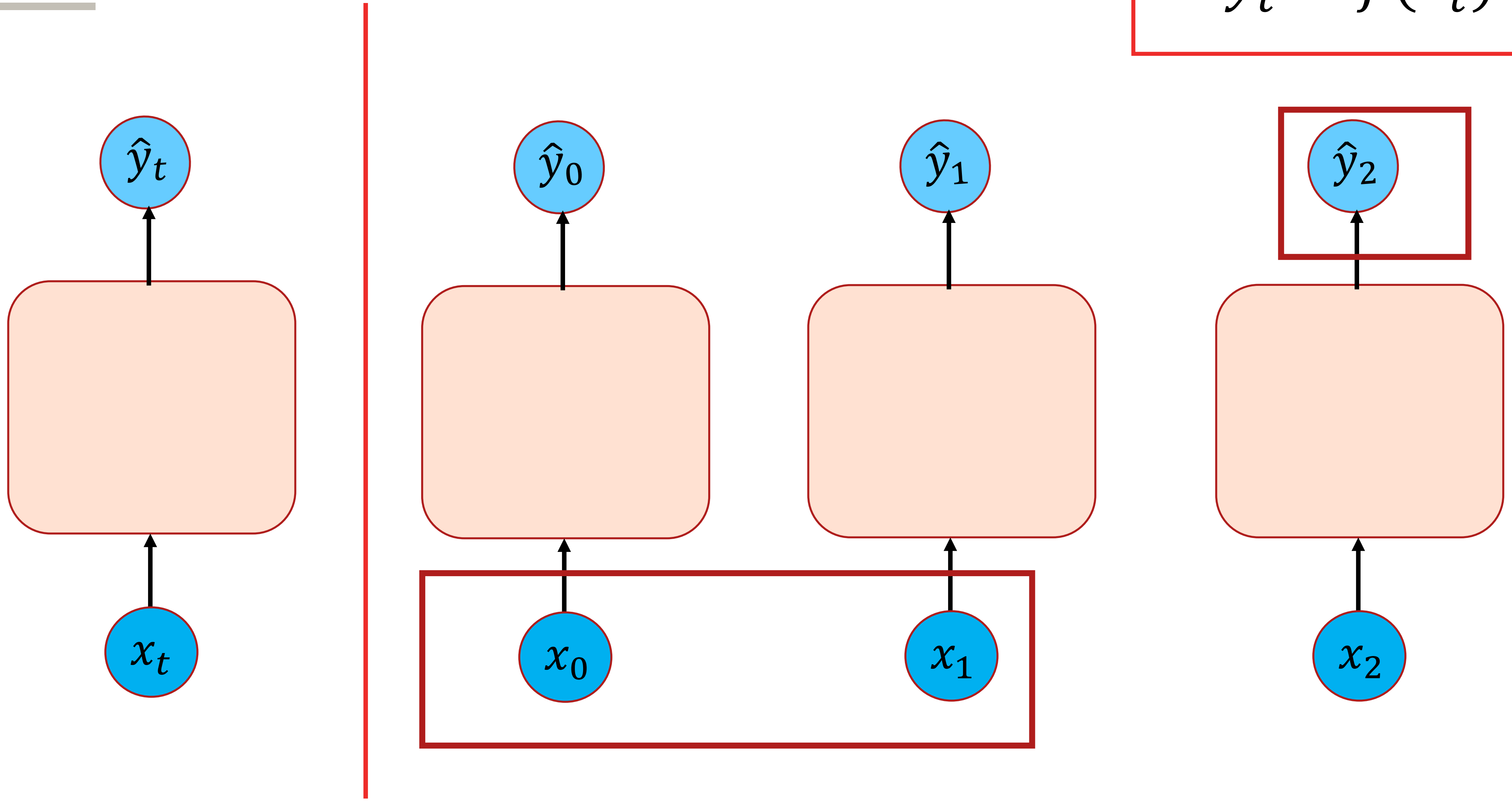
$$\hat{y}_t = f(x_t)$$



UNIVERSITY OF CALGARY

Temporal Dependency

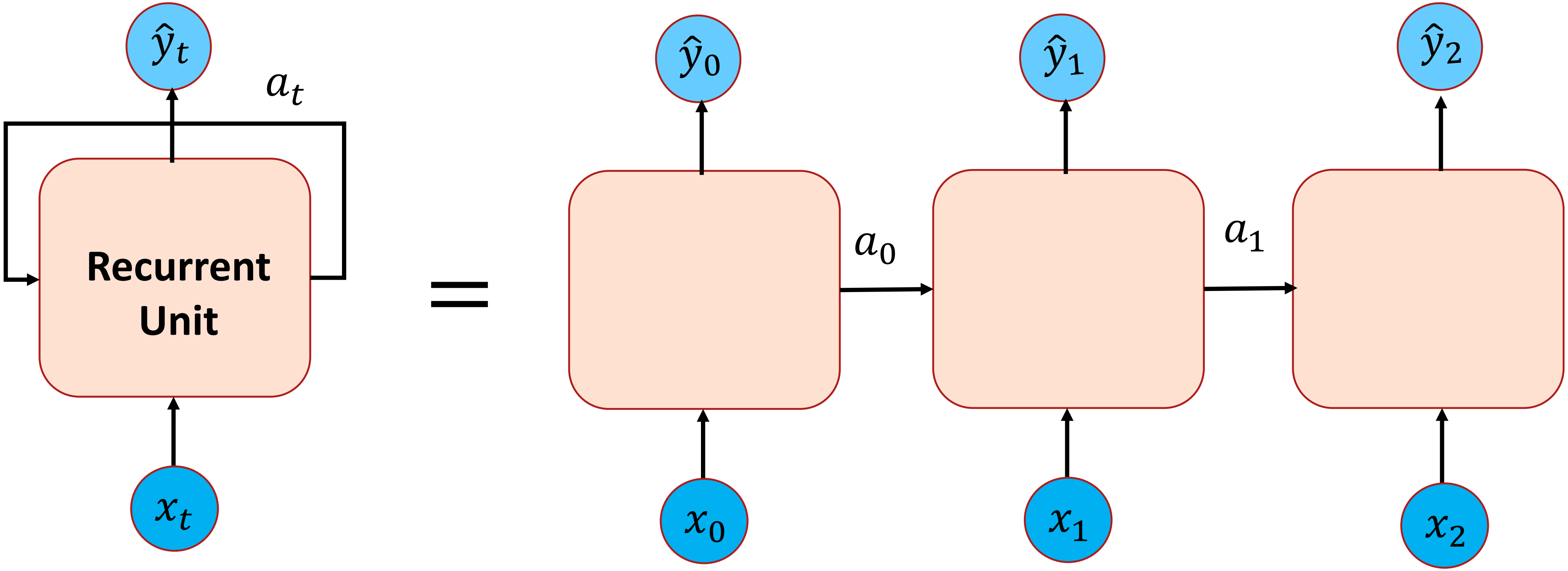
$$\hat{y}_t = f(x_t)$$



UNIVERSITY OF CALGARY

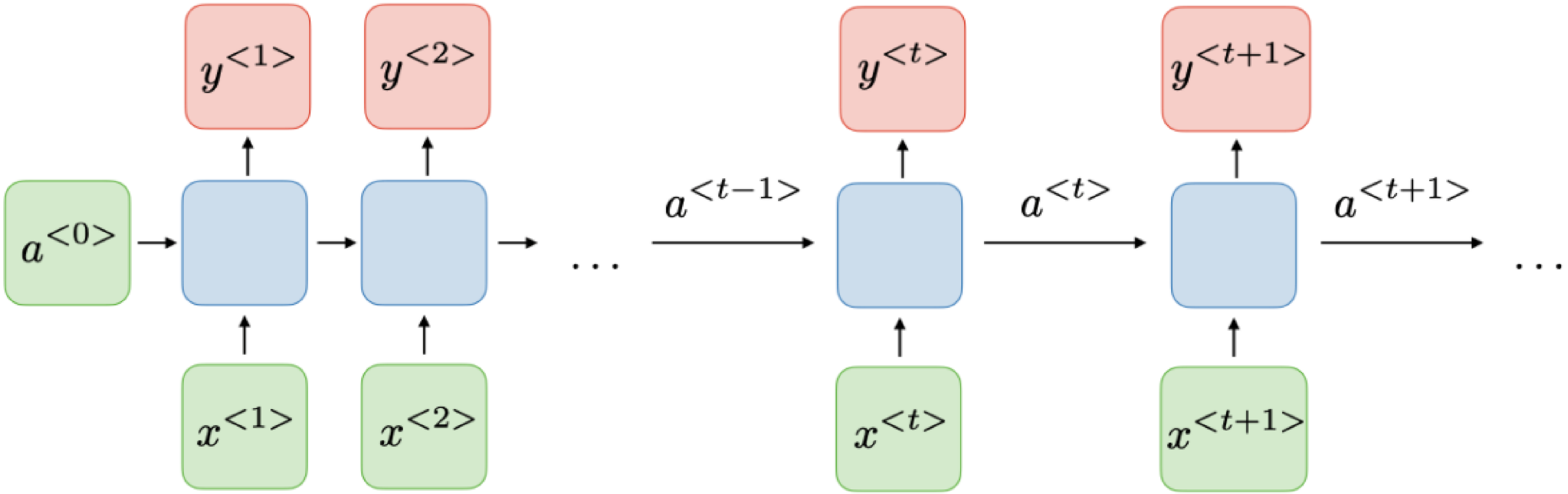
Traditional RNN

$$\boxed{y^t = f(x^t, a^{t-1})}$$



Traditional RNN

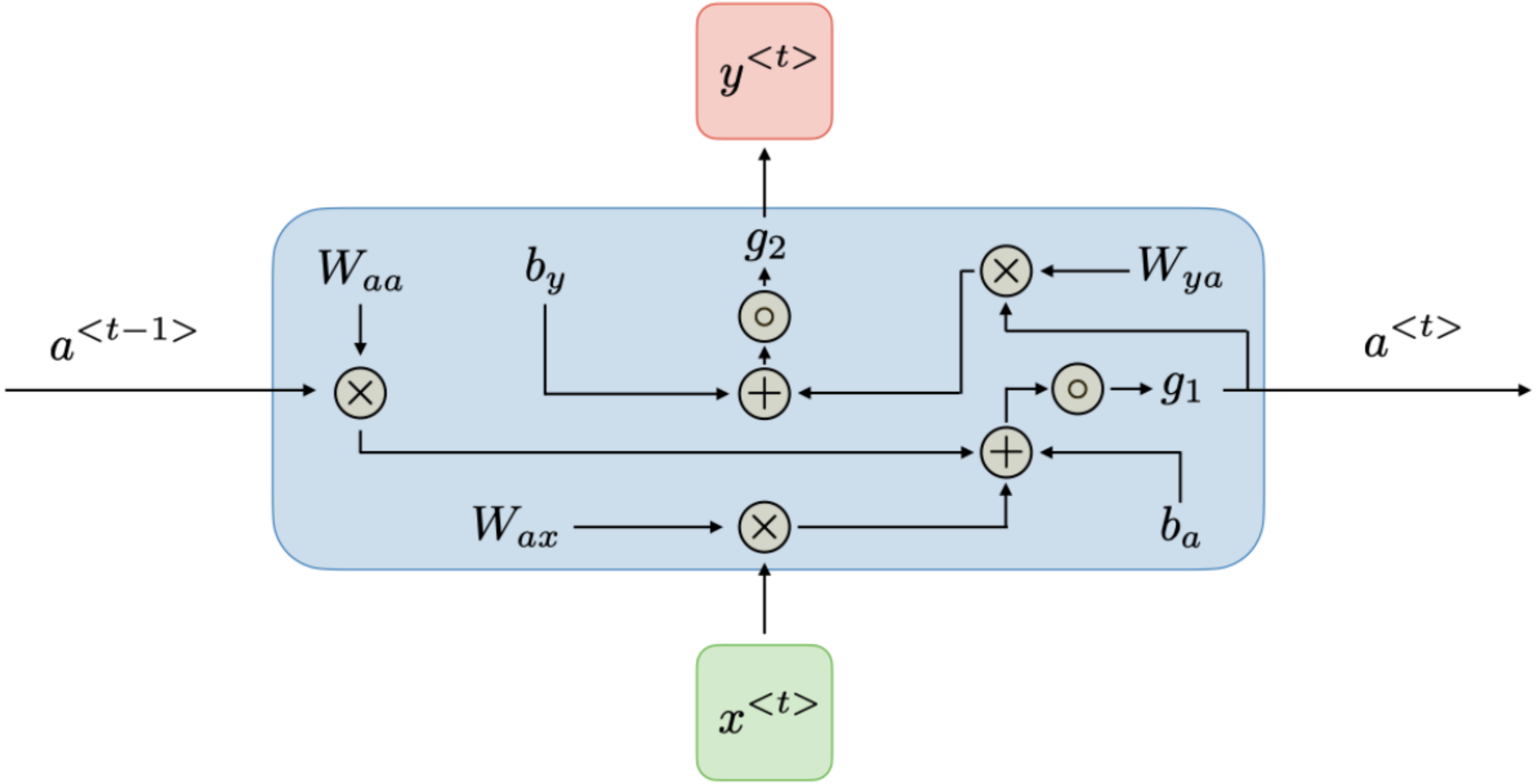
$$y^t = f(x^t, a^{t-1})$$



- RNNs can be seen as a (very deep) feedforward network with shared weights
- Model is trained using backpropagation through time

UNIVERSITY OF CALGARY

Traditional RNN

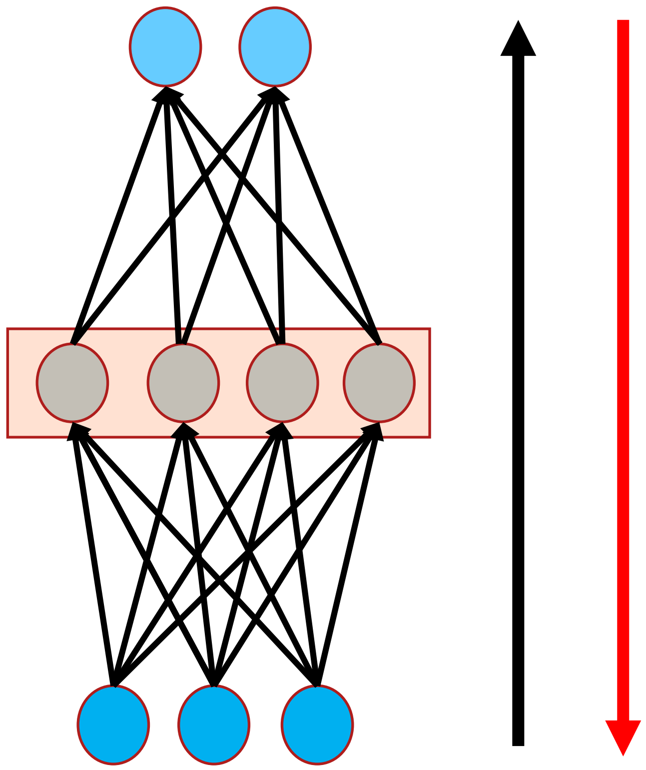


$$a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a)$$

$$y^{<t>} = g_2(W_{ya}a^{<t>} + b_y)$$

Back Propagation Through Time (BPTT)

Recall: Feed Forward Back Propagation

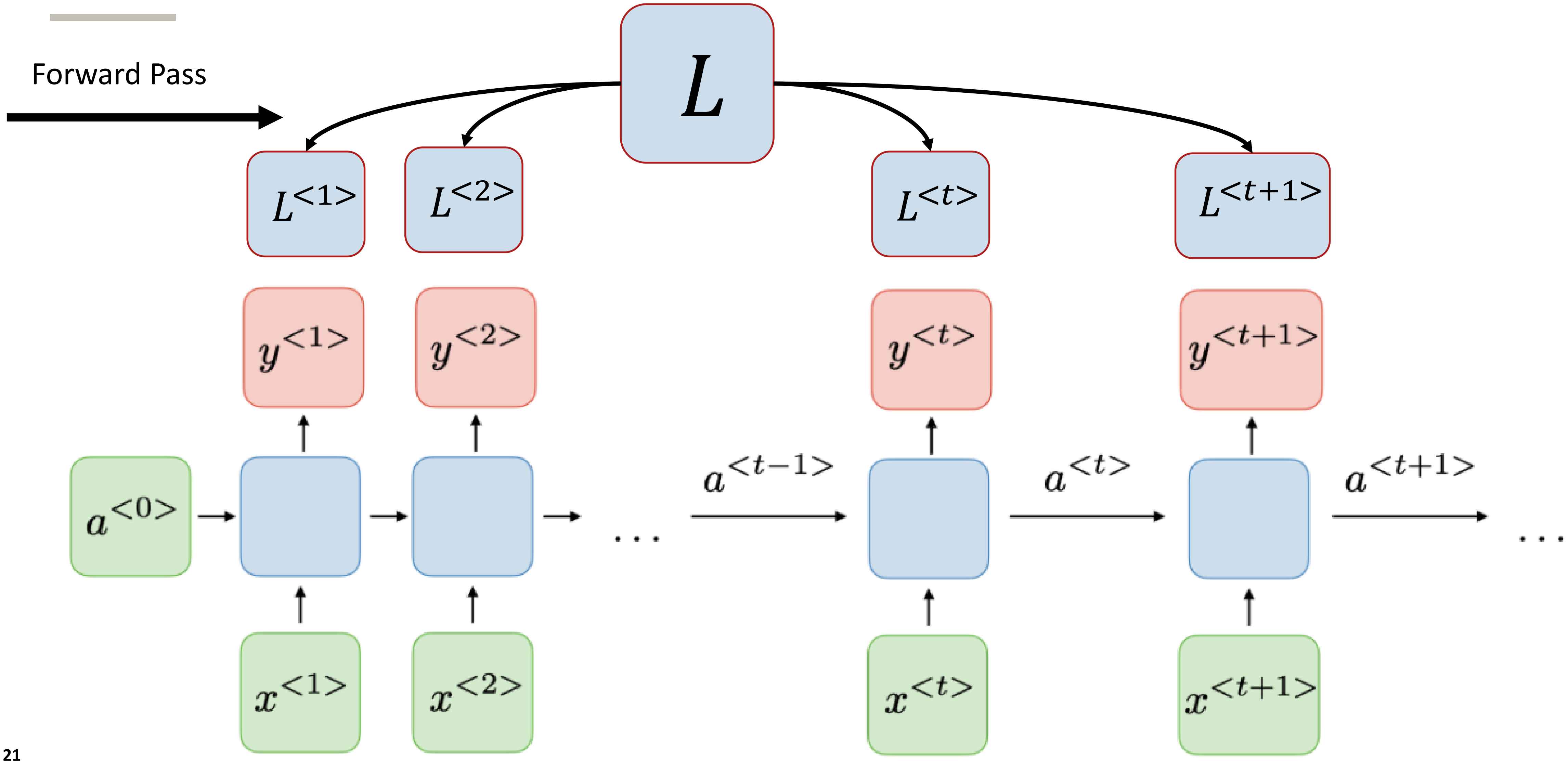


1. Perform forward pass to compute loss

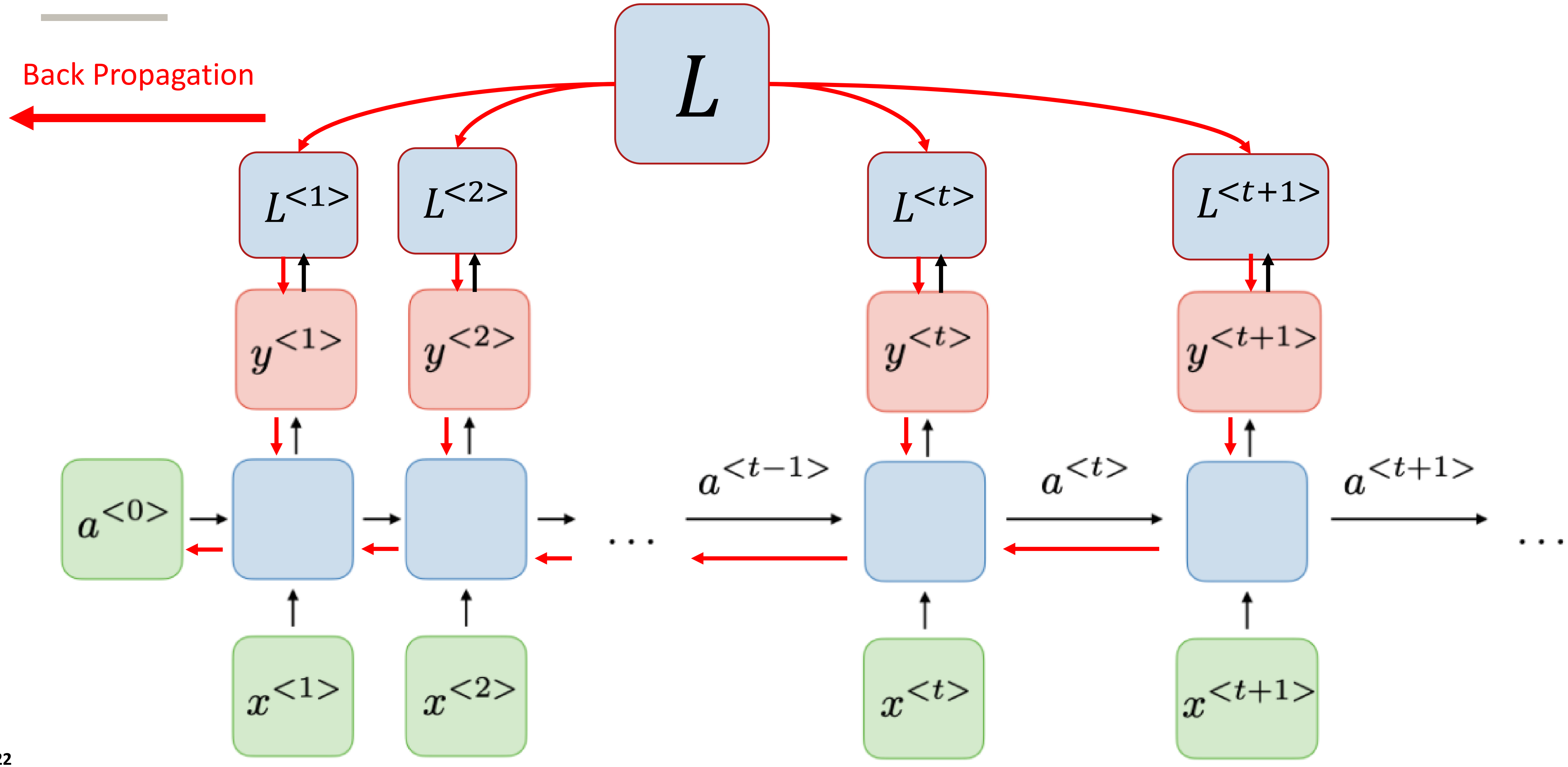
2. Compute gradient of loss w.r.t each parameter

3. Update parameters to minimize loss

Back Propagation Through Time (BPTT)



Back Propagation Through Time (BPTT)



RNNs Advantages and Disadvantages

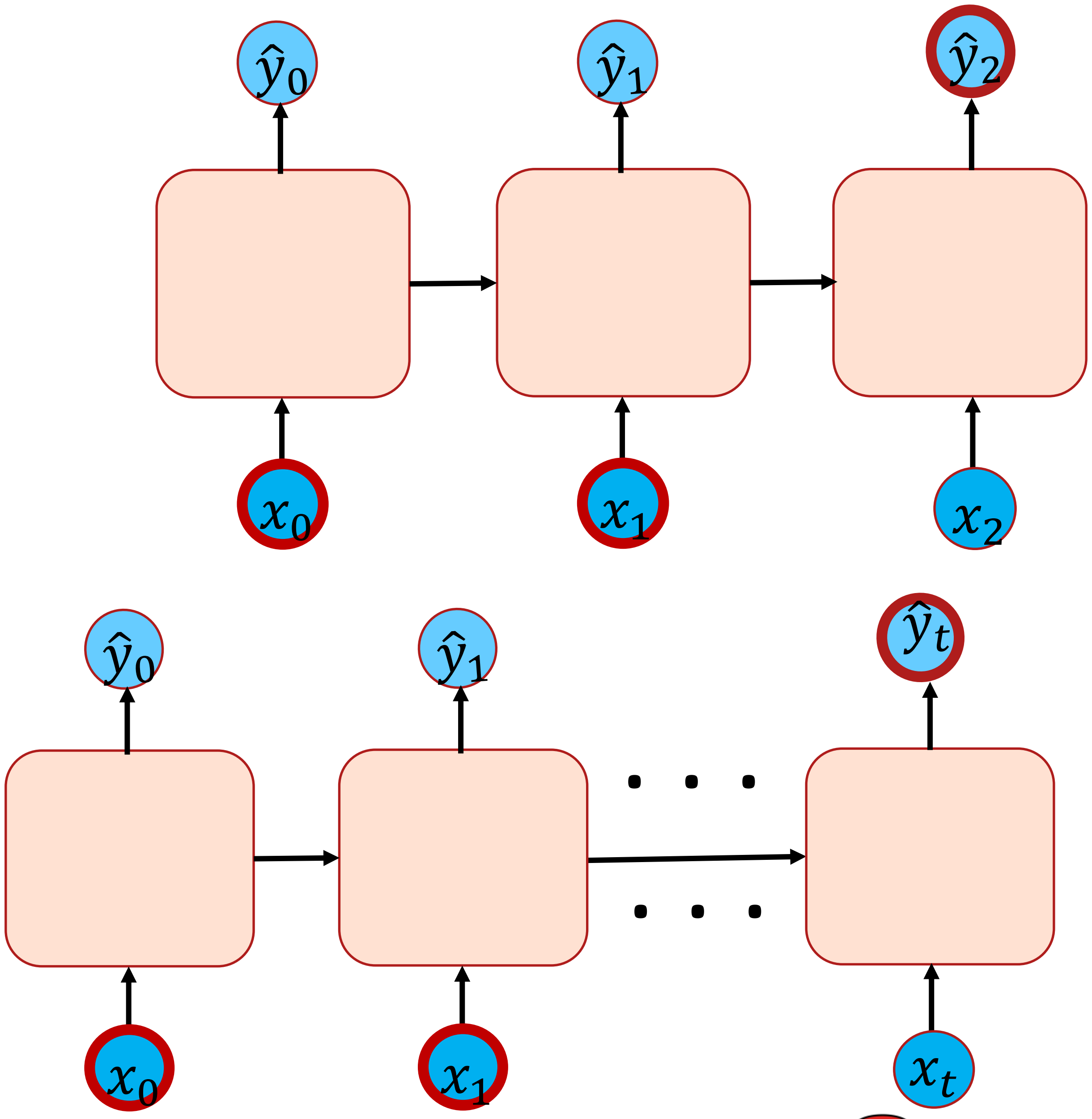
Advantages	Disadvantages
Possibility of processing input of any length	Computation being slow
Model size not increasing with size of input	Difficulty of accessing information from a long time ago
Computations take into account historical information	Cannot consider any future input for the current state
Weights are shared across time	

Problems with Long Term Dependency

Long Term Dependencies

"Today is Wednesday, tomorrow is _____"

"I was born in Canada, … I am fluent in _____"



Vanishing gradients

- As we propagate the gradients back in time, usually their magnitude quickly decreases this is called “vanishing gradient problem”

- In practice this means that learning long term dependencies in data is difficult for simple RNN architecture

- Special RNN architectures address this problem:
  - *Exponential trace memory* (Jordan 1987, Mozer 1989)
  - *Long Short-term Memory* (Hochreiter & Schmidhuber, 1997))

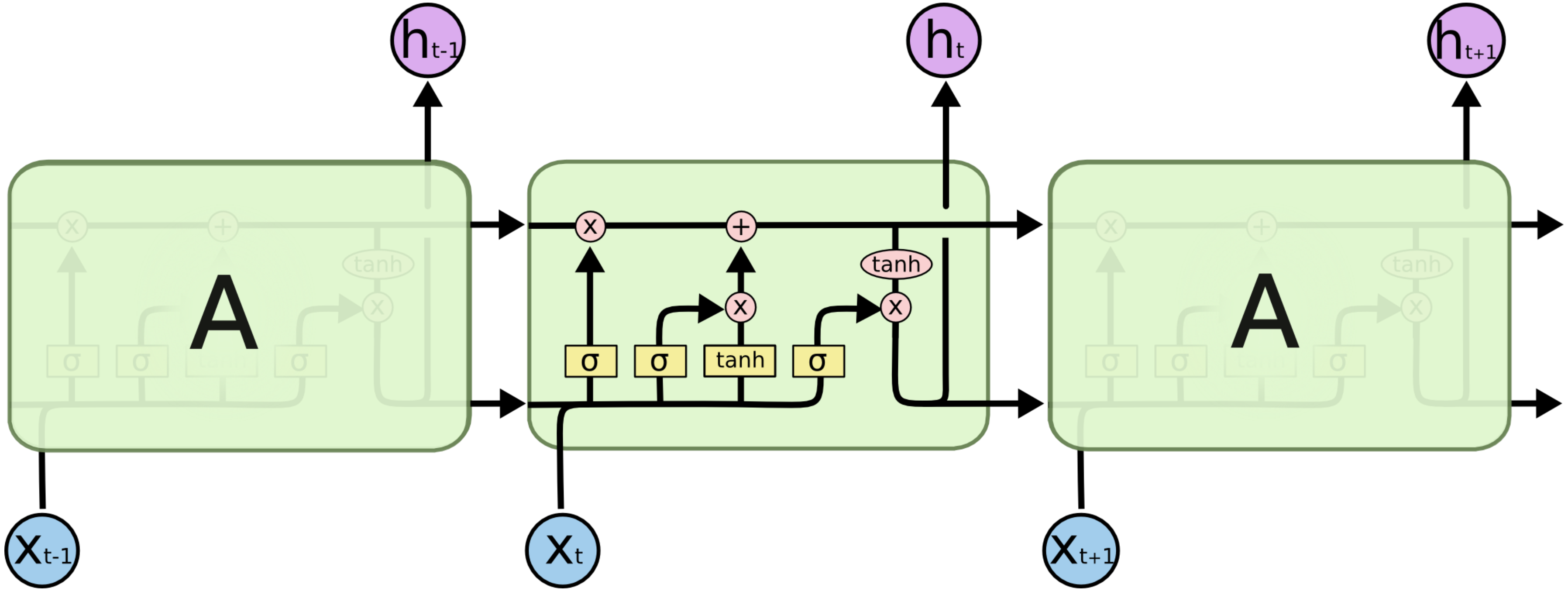
Exploding gradients

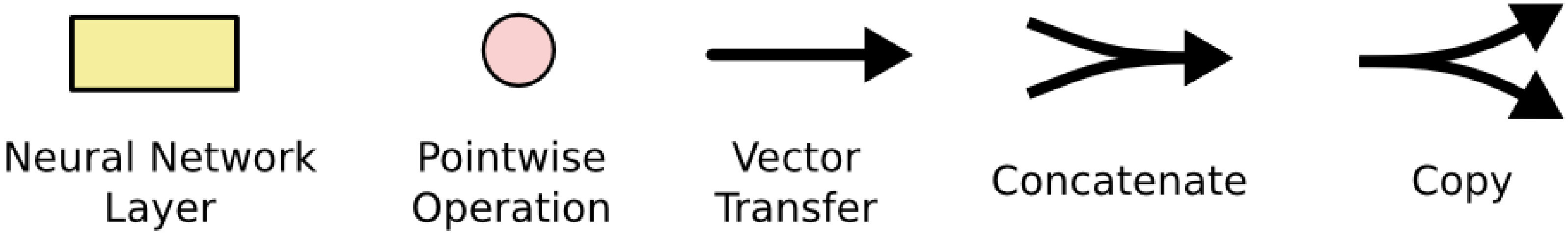
- Sometimes, the gradients start to increase exponentially during backpropagation through the recurrent weights
- Happens rarely, but the effect can be catastrophic: huge gradients will lead to big change of weights, and thus destroy what has been learned so far
- One of the main reasons why RNNs were supposed to be unstable
- Simple solution: clip or normalize values of the gradients to avoid huge changes of weights

Solution to Long-term Dependency (LSTMs)

Long Short-Term Memory (LSTM)

- LSTM is a type of RNN capable of learning long-term dependencies



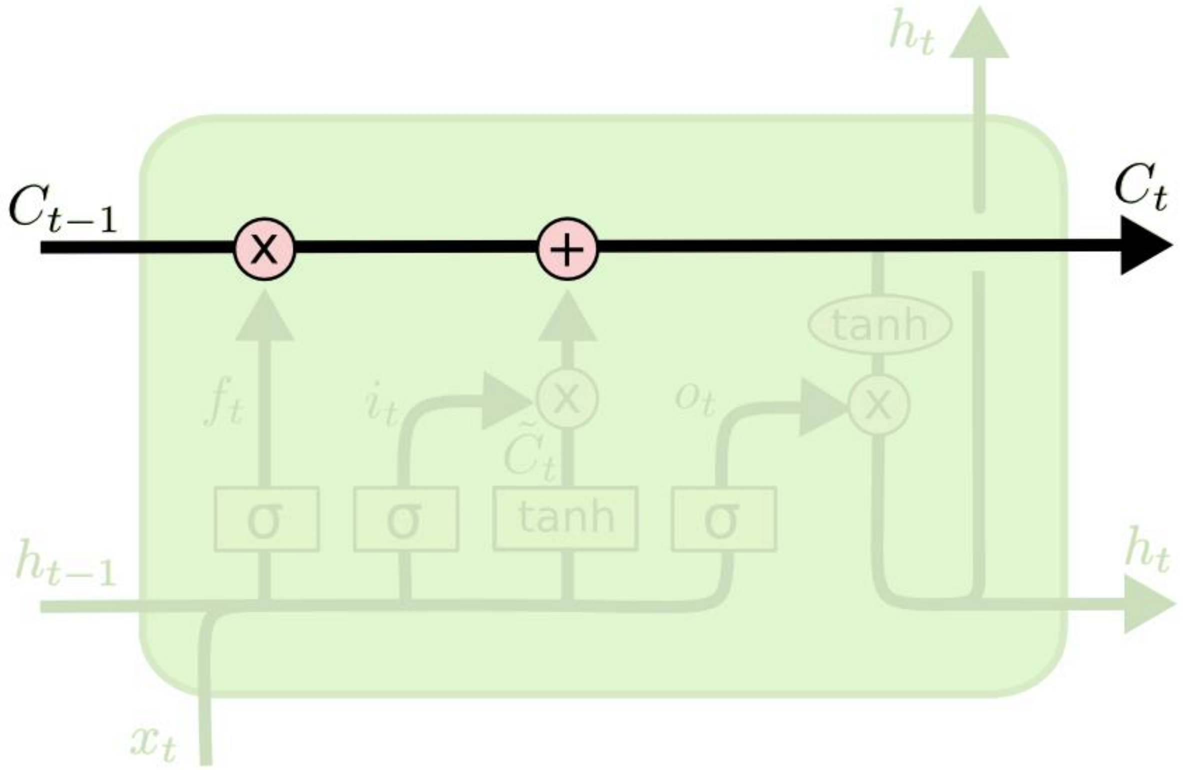


UNIVERSITY OF CALGARY

The Cell State

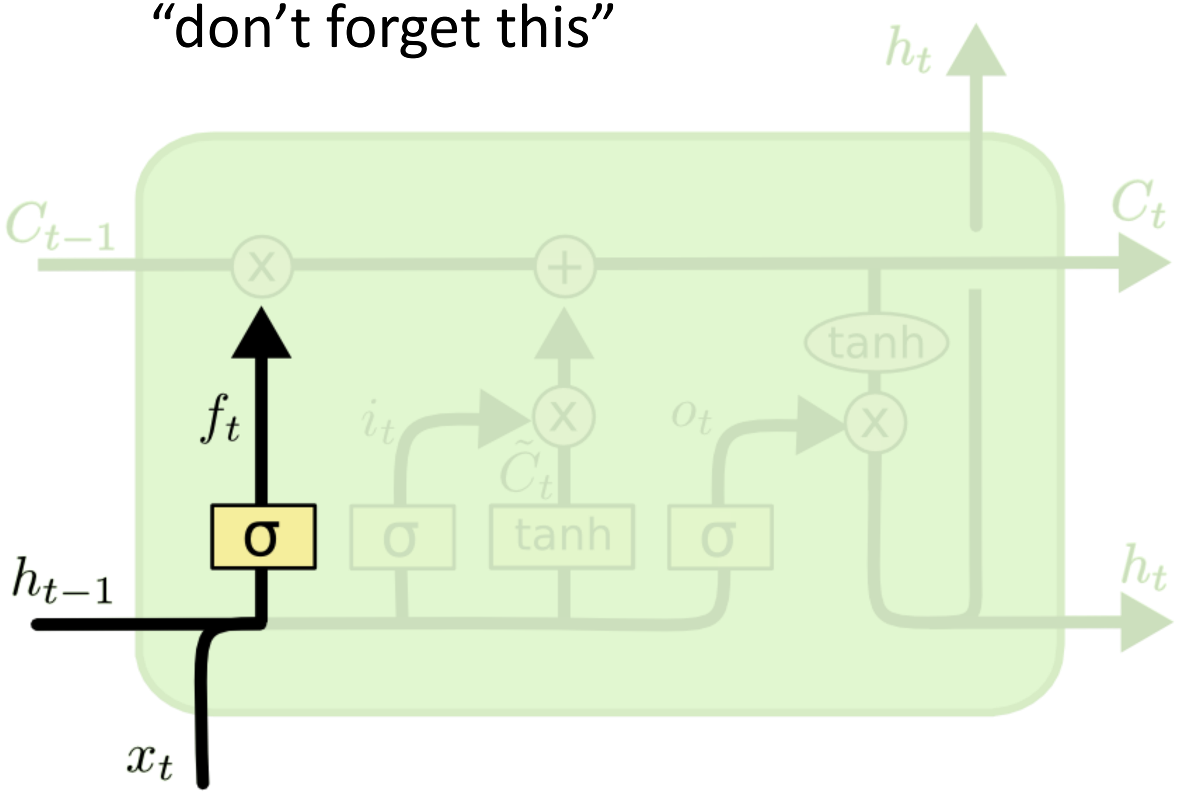
- Information flows along the cell state unchanged

- However, LSTM can add or remove info from cell state through controlled “gates”



Step 1: Forget Gate

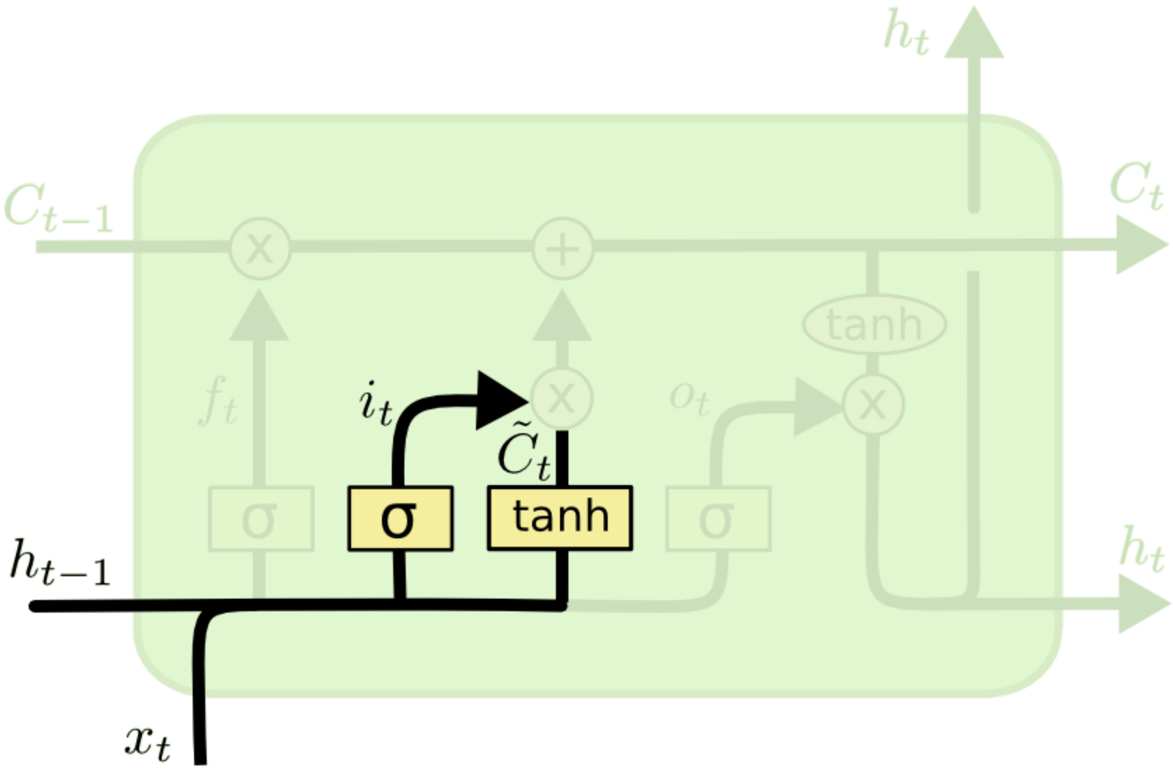
- Determine how much information should be forgotten from cell state
- Outputs number values from 0 to 1, 0 means “completely forget” and 1 means “don’t forget this”



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

Step 2: Input Gate Layer

- What new information will be stored in the cell state



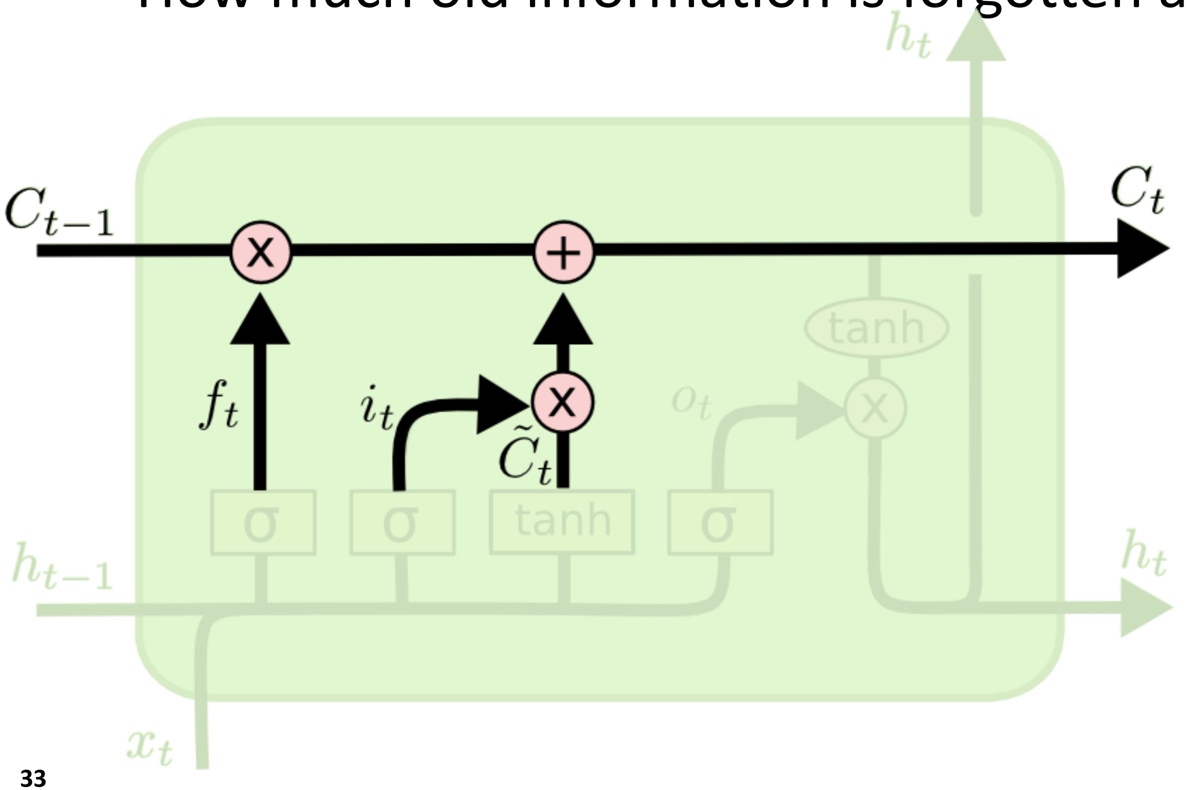
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

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

Step 3: Update the Old Cell State (C_{t-1})

- Previous steps inform how the previous cell state will be updated

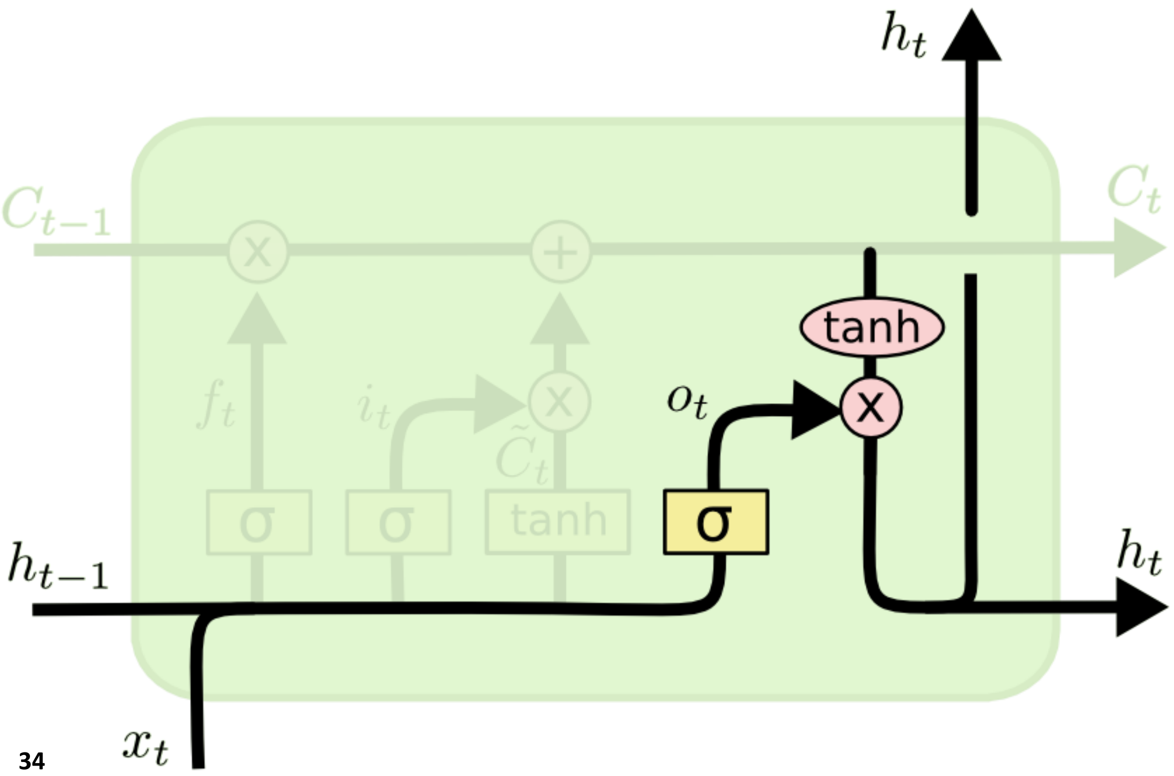
- How much old information is forgotten and how much new info will be added



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

Step 4: Output Layer

- Calculating new hidden state (h_t)



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

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

Summary

- RNNs are capable of handling sequences of arbitrary lengths
- Traditional RNNs are not capable in practice to model long-term dependencies in data
- The LSTM model allows you to model these long-term dependencies
- More details in the tutorials…

Thank you!