CS229: Machine Learning

Deep Learning

part 3

Xiangliang Zhang

King Abdullah University of Science and Technology

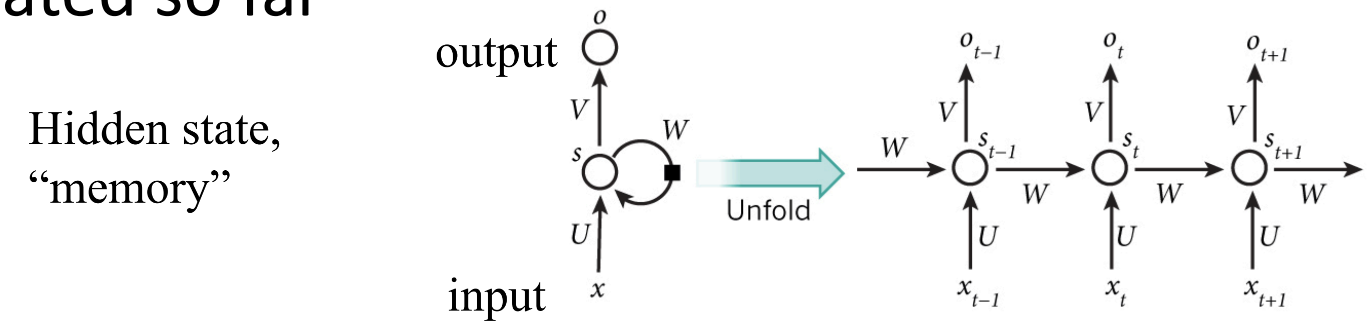
KAUST
King Abdullah University of
Science and Technology

Outline

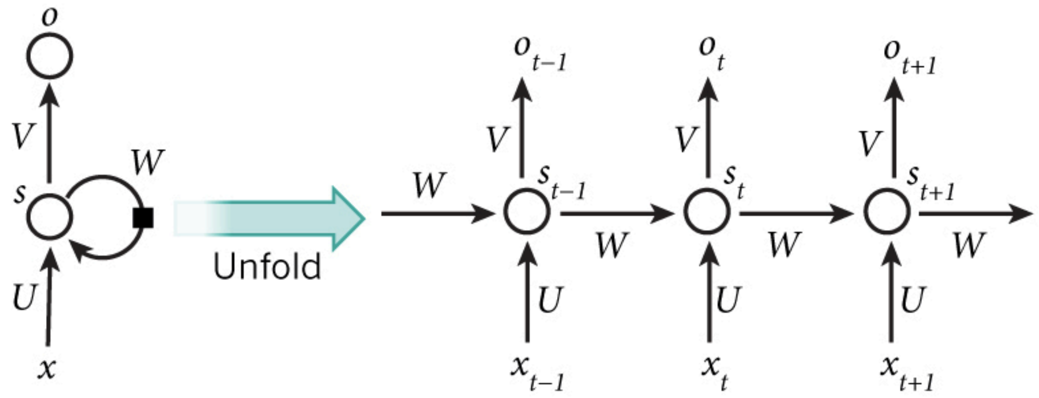
- Popular Deep Network Architectures
  - CNN
  - RNN, LSTM
  - Autoencoder, Attention
- Generative models and GAN
- Open Discussion

Recurrent Neural Networks, RNN

- Learning from sequential data
- Promising solutions for
  - Machine translation
  - Speech Recognition
  - Generating Image Descriptions (when used together with CNN)
  - NLP tasks, e.g., scoring arbitrary sentences based on how likely they are to occur (a measure of grammatical and semantic correctness), and generate new text
- Having a "memory" which captures information about what has been calculated so far



Recurrent Neural Networks, RNN



o_t: the next word in a sentence $o_t = \mathrm{softmax}(Vs_t)$

$$s_t = f(Ux_t + Ws_{t-1})$$

Activation function f can be tanh or ReLU

x_t: a one-hot vector corresponding to a word of a sentence

- RNN shares the same parameter (U,V,W) across all steps (performing the same task at each step, just with different inputs)
- Greatly reduces the total number of parameters to learn
- Can have one output at each time step, or only one final output

Recurrent Neural Networks, RNN

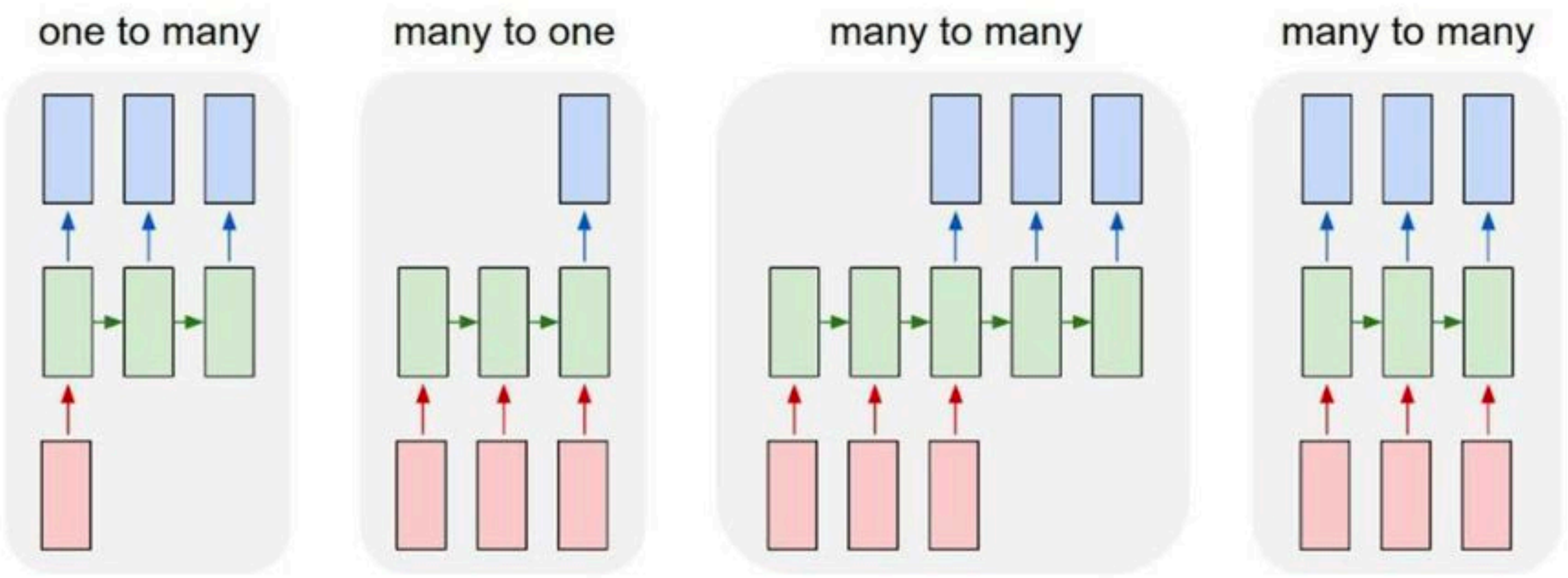
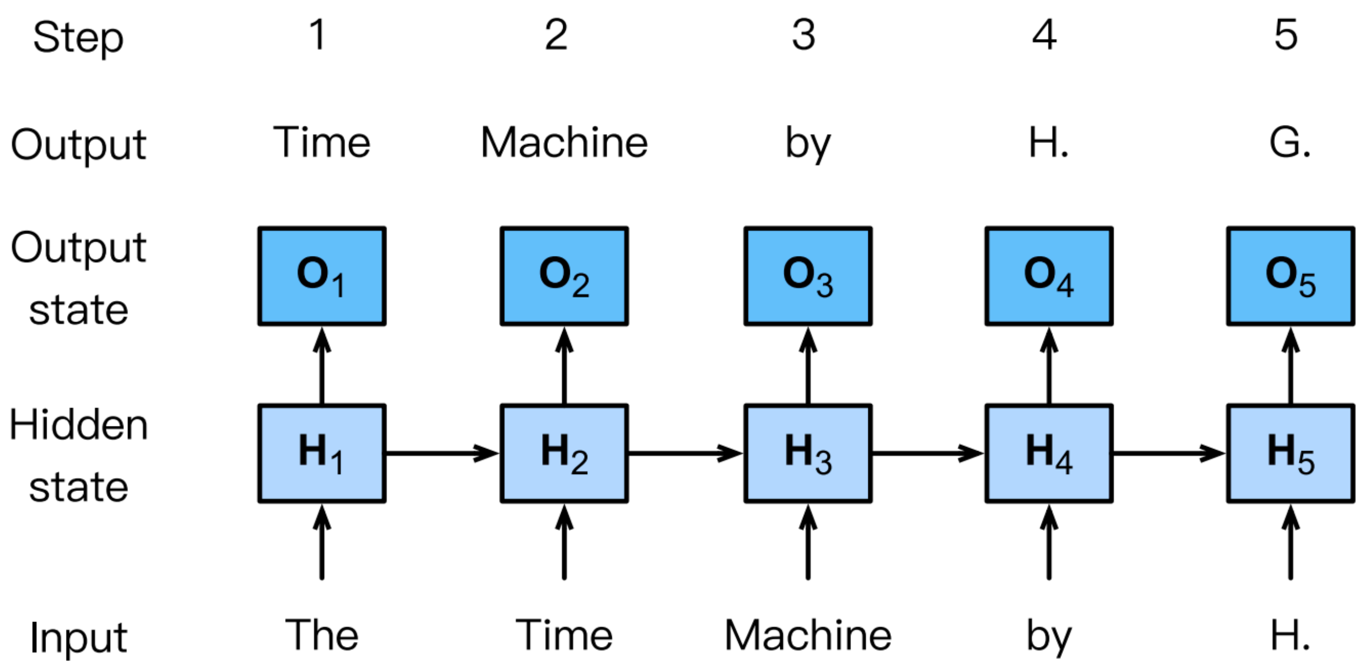


Image Captioning | Video Activity Recog / Text Classification | Video Captioning / Machine Translation | POS Tagging / Language Modeling

An example of next word prediction

- At each step, output is predicted with a softmax operation on the output layer
- Training by the cross-entropy loss function computed with the error between each output prediction and the ground truth word.



If predicting the next character in a text sequence, how many output dimensions?

RNN

Unstable gradient problem of RNN:

Vanishing or Exploding

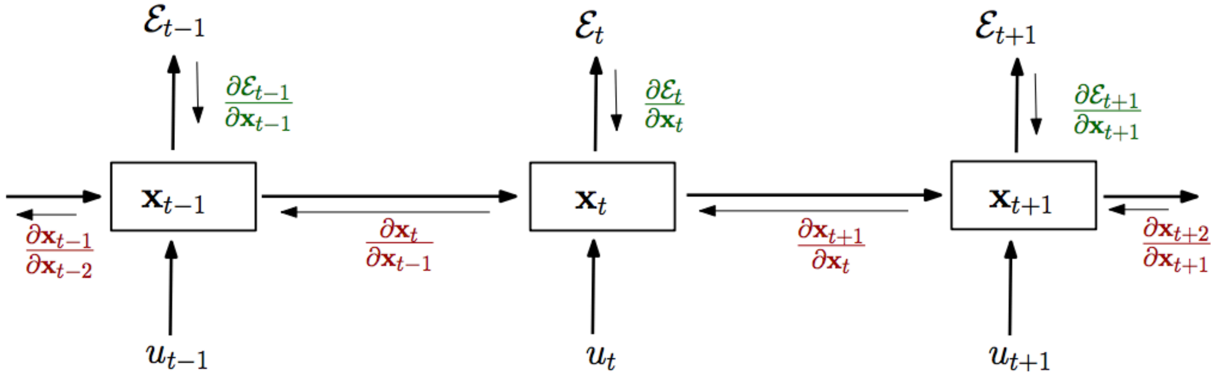


Figure 2. Unrolling recurrent neural networks in time by creating a copy of the model for each time step. We denote by $\mathbf{x}_t$ the hidden state of the network at time t, by $\mathbf{u}_t$ the input of the network at time t and by $\mathcal{E}_t$ the error obtained from the output at time t.

highlight the exploding gradients problem:

$$\frac{\partial \mathcal{E}}{\partial \theta} = \sum_{1 \le t \le T} \frac{\partial \mathcal{E}_t}{\partial \theta} \tag{3}$$

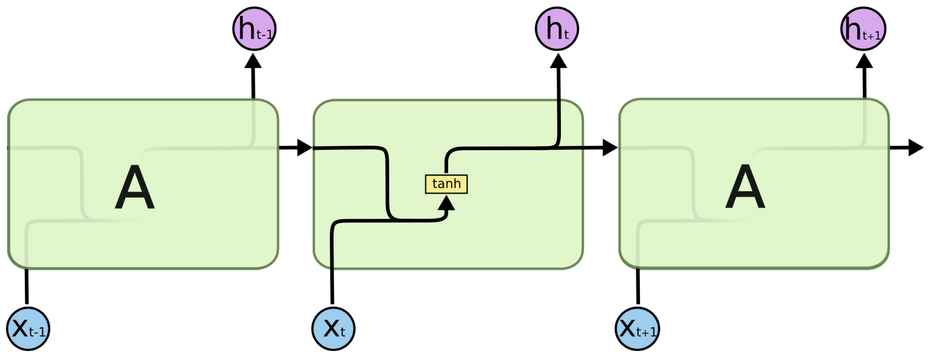
$$\frac{\partial \mathcal{E}_t}{\partial \theta} = \sum_{1 \le k \le t} \left(\frac{\partial \mathcal{E}_t}{\partial \mathbf{x}_t} \frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k} \frac{\partial^+ \mathbf{x}_k}{\partial \theta} \right) \tag{4}$$

$$\frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k} = \prod_{t \ge i > k} \frac{\partial \mathbf{x}_i}{\partial \mathbf{x}_{i-1}} = \prod_{t \ge i > k} \mathbf{W}_{rec}^T diag(\sigma'(\mathbf{x}_{i-1})) \tag{5}$$

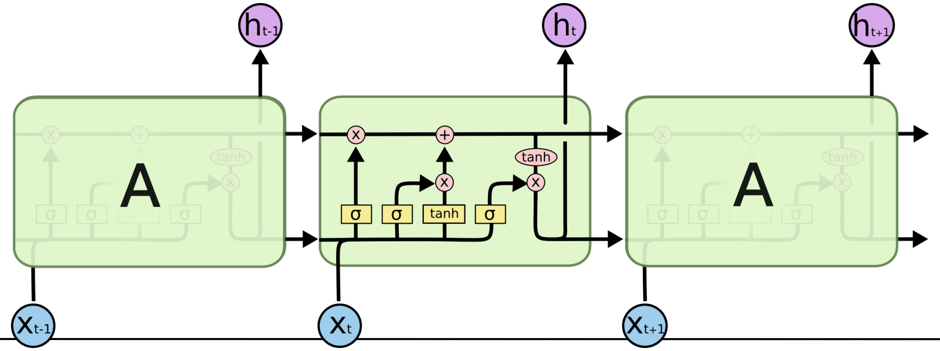
On the difficulty of training recurrent neural networks
By Pascanu, Mikolov, Bengio
In ICML'13

Long Short Term Memory: LSTM networks

- LSTM is designed to address the unstable gradient problem of RNN through a **gating** mechanism
- Also, takes long term memory, e.g., for predicting "I grew up in France...I speak fluent ?", we need the context of France, from further back.



A standard RNN contains a single layer

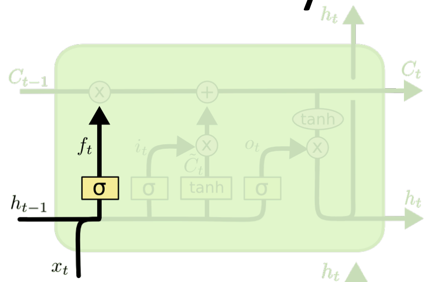


LSTMs also have this chain like structure, but the repeating module has **four** neural network layers, interacting in a very special way.

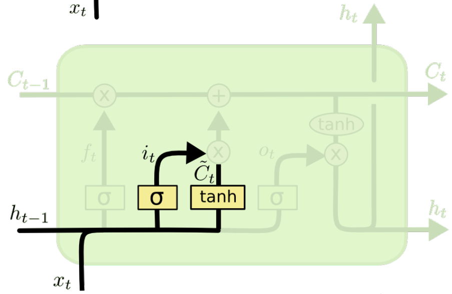
introduced by Hochreiter & Schmidhuber in 1997

Long Short Term Memory: LSTM networks

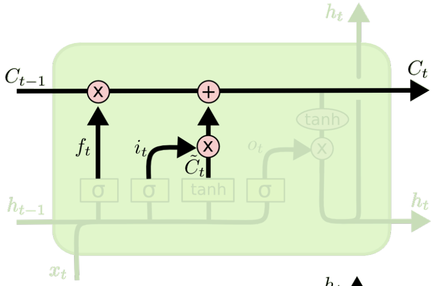
Just another way to compute a hidden state, replacing $s_t = f(Ux_t + Ws_{t-1})$



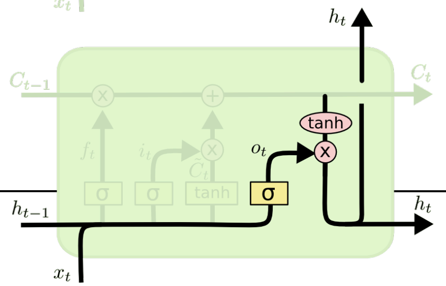
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$



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



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



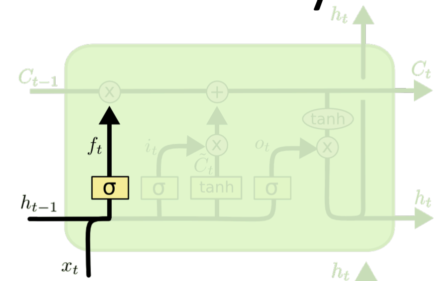
$$o_t = \sigma\left(W_o \ [h_{t-1}, x_t] \ + \ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

- Use **Gates** to control how to let information go through.
- A Gate is composed out of a sigmoid neural net layer and a pointwise multiplication operation.
- There are three Gates in one module.
- The sigmoid layer outputs numbers between zero and one, a value of zero means "let nothing through," while a value of one means "let everything through!"

introduced by Hochreiter & Schmidhuber in 1997

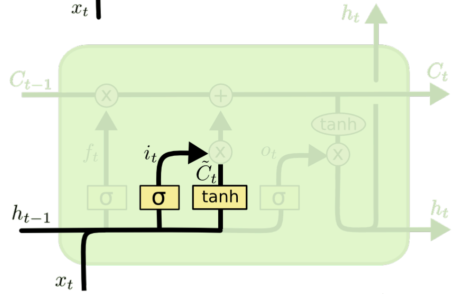
Long Short Term Memory: LSTM networks

Just another way to compute a hidden state, replacing $s_t = f(Ux_t + Ws_{t-1})$



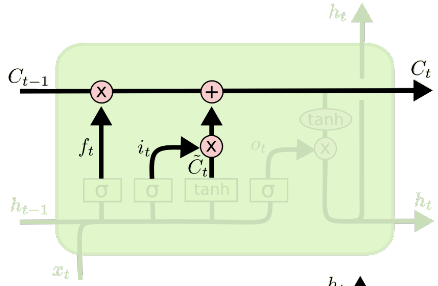
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

forget gate layer: what to forget



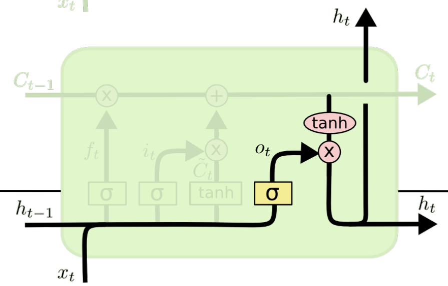
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

what new information to store:
By combining the “input gate layer” and “a tanh layer”



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

update the old memory cell state, C_{t-1}, into the new memory cell state C_t



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

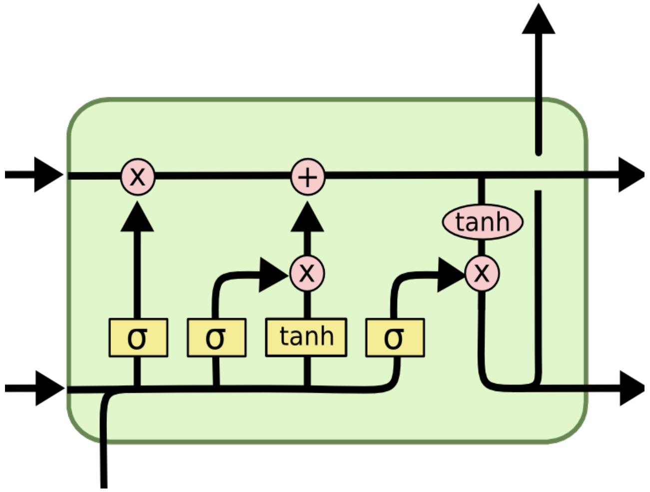
what to output

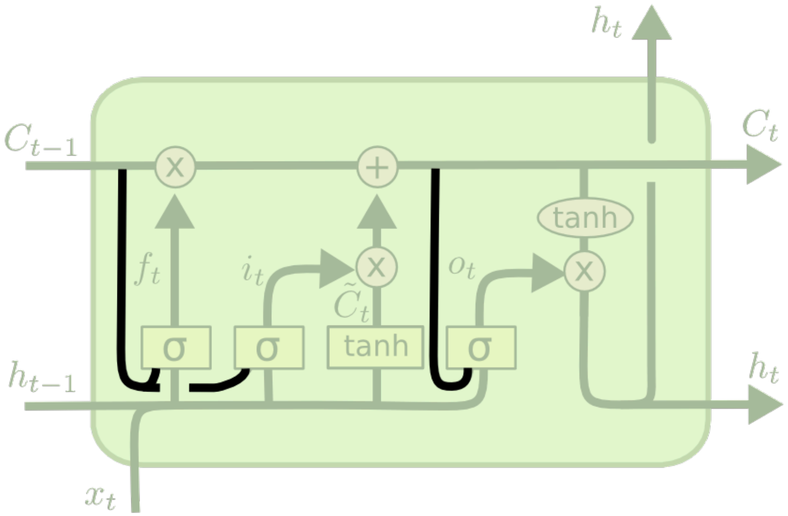
LSTM Forward and Backward Pass

http://arunmallya.github.io/writeups/nn/lstm/index.html#/

A nice illustration made by Arun, a graduate student at UIUC

LSTM variants

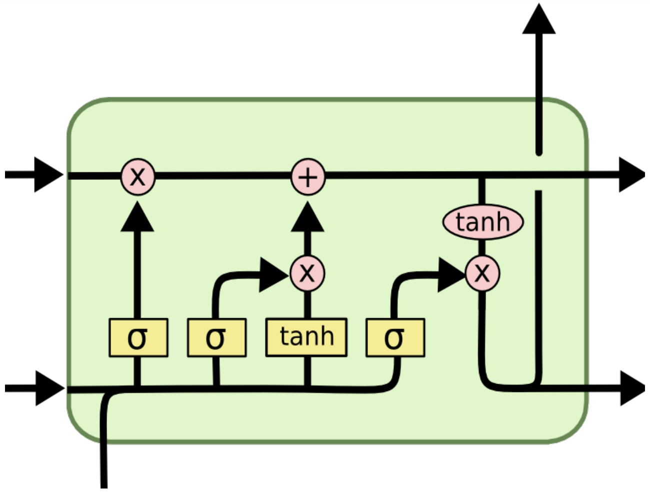


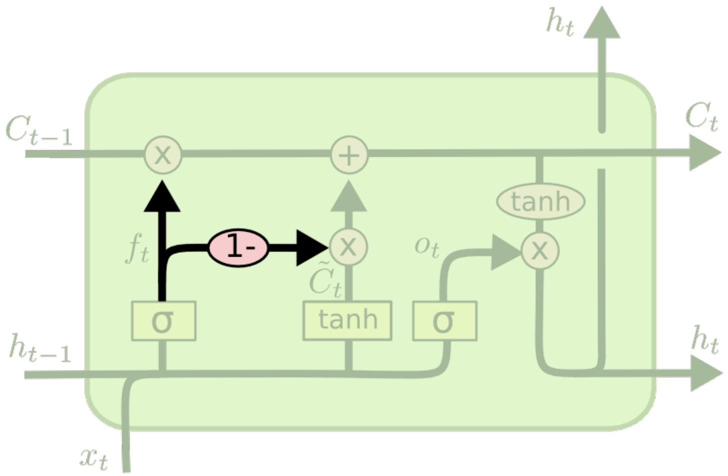


let the gate layers look at the cell state

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

LSTM variants



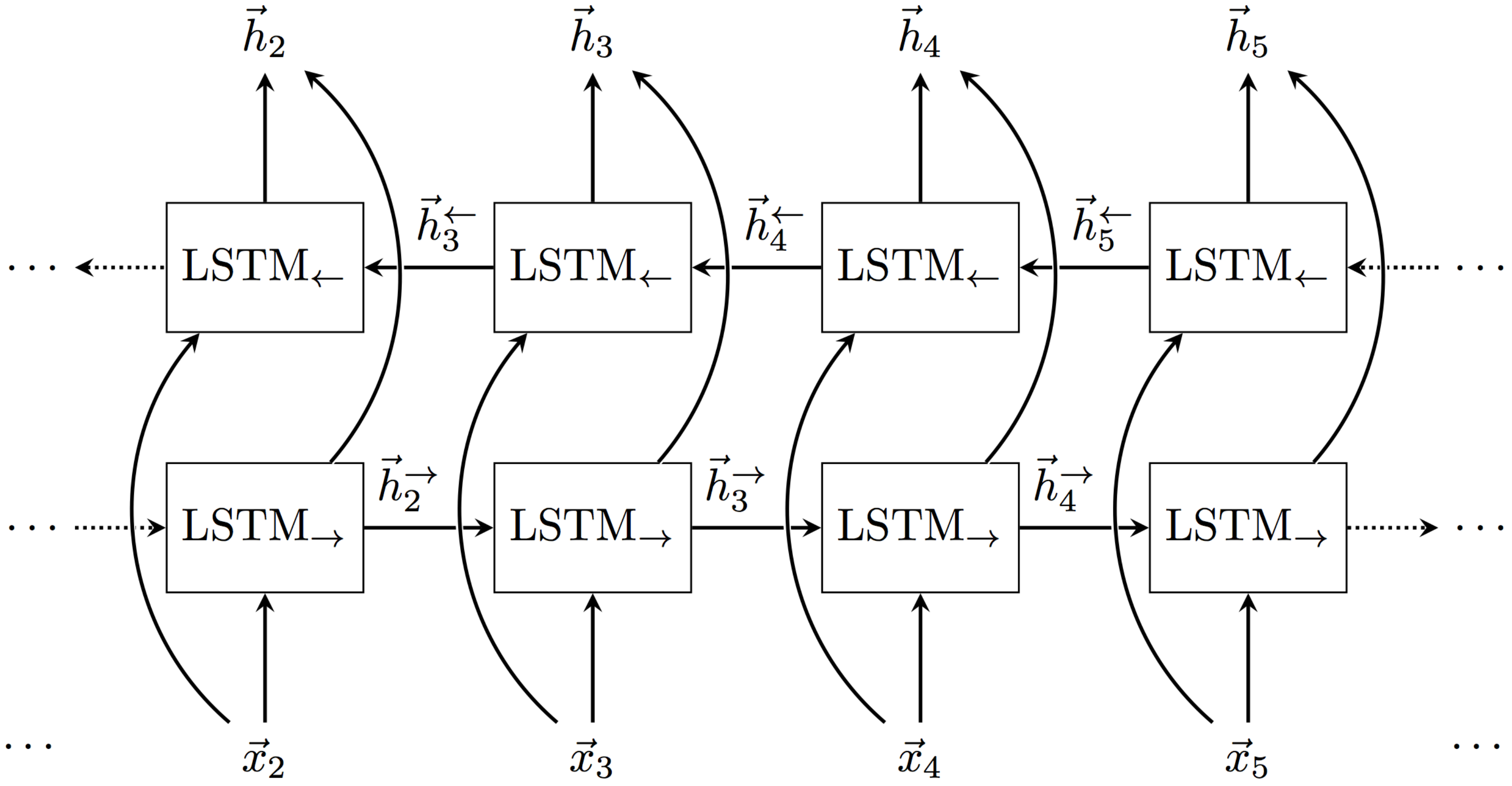


use coupled forget and input gates:

- only forget when we're going to input something in its place
- only input new values to the state when we forget something older

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

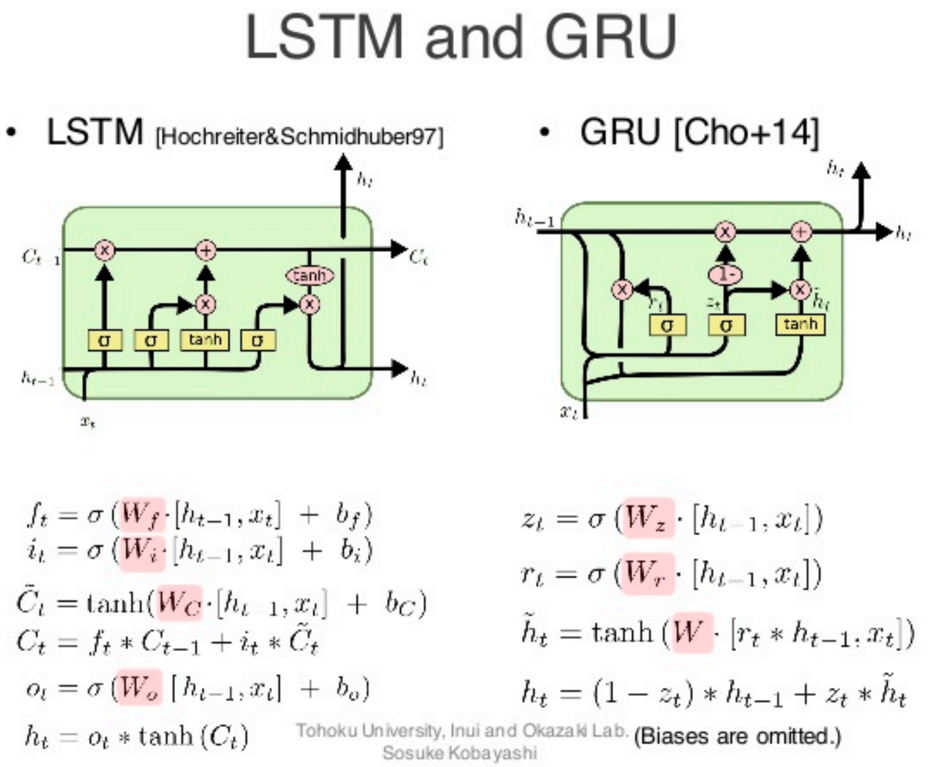
Bidirectional LSTM



LSTM vs GRU

And many other variants. Which is the **best**?

- See a paper: An Empirical Exploration of Recurrent Network Architectures. ICML 2015
- Authors evaluated **over ten thousand** different RNN architectures, and identified an architecture that outperforms both the LSTM and the recently-introduced Gated Recurrent Unit (GRU) **on some but not all** tasks



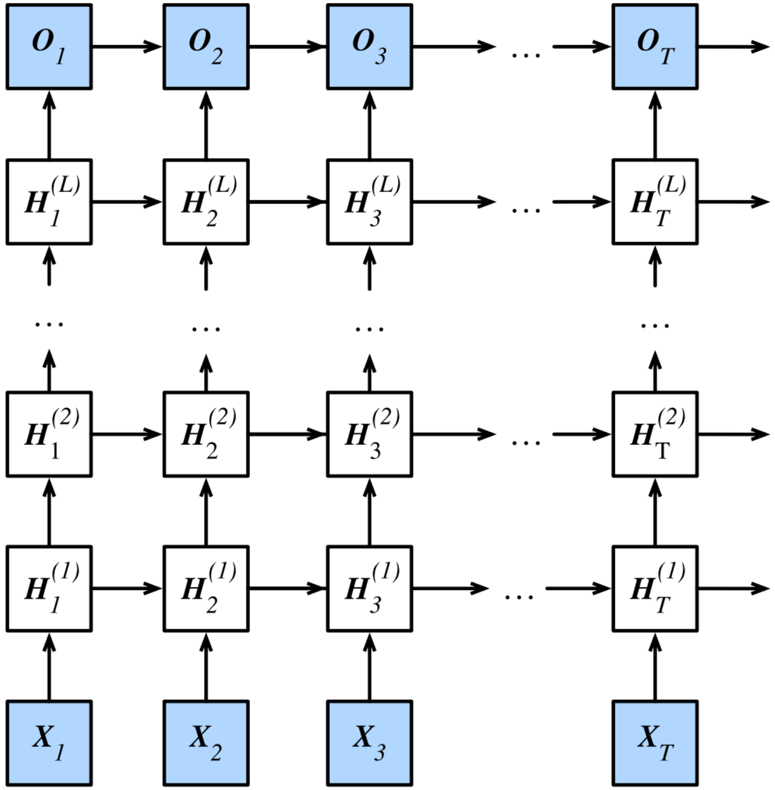
- Adding a **positive bias to the forget gate** greatly improves the performance of the LSTM. Given that this technique the simplest to implement, we recommend it for every LSTM implementation.

Deep RNN (LSTM) in Practice

$$\mathbf{O}_t = g\left(\mathbf{H}_t^{(L)}\right)$$

$$\mathbf{H}_t^{(l)} = f_l\left(\mathbf{H}_t^{(l-1)}, \mathbf{H}_{t-1}^{(l)}\right).$$

$$\mathbf{H}_t^{(1)} = f_1\left(\mathbf{X}_t, \mathbf{H}_{t-1}^{(1)}\right),$$



Seq2Seq for translation