

430.457

Introduction to Intelligent Systems

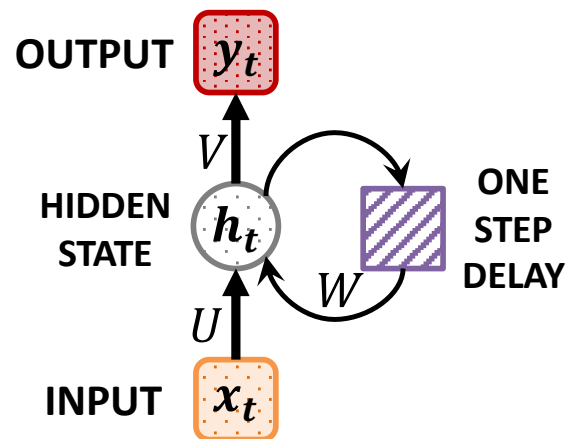
Prof. Songhwai Oh
ECE, SNU

RECURRENT NEURAL NETWORKS (RNN)

Recurrent Neural Networks : For what?

- Human remembers and uses the pattern of a sequence.
 - Try 'a b c d e f g...'
 - But how about 'z y x w v u t s...' ?
- The idea behind RNN is to **make use of sequential information.**
- Let's learn a pattern of a sequence and utilize it (estimate, generate, etc.)
- But **HOW?**

Recurrent Neural Networks : Typical RNNs



$$h_t = f(Ux_t + Wh_{t-1} + b)$$

$$y_t = Vh_t + c$$

- RNNs are called **RECURRENT** because they **perform the same task for every element of a sequence**, with the output being **dependent on the previous computations**.
- RNNs have a **memory** which captures information about what has been calculated so far.
- The hidden state h_t captures some information about the sequence.
- If we use $f = \tanh$, Vanishing or exploding gradient problem happens.
 - To overcome this, we use LSTM or GRU.
 - **LSTM**: Long short-term memory
 - **GRU**: gated recurrent unit

Recurrent Neural Networks : Typical RNNs

Vanishing or exploding gradient problem

$$h_t = f(x_t, h_{t-1})$$

$$f(x_t, h_{t-1}) = Ux_t + Wh_{t-1} + b$$

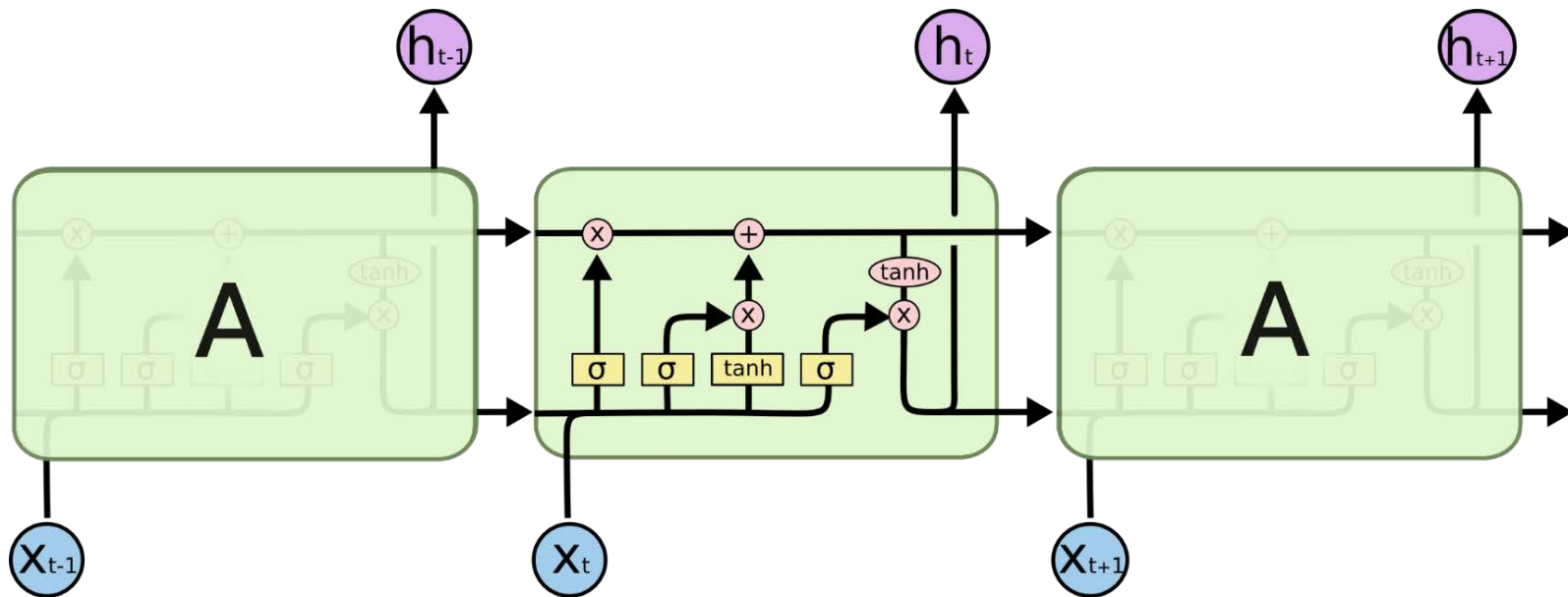
$$h_3 = f(x_3, h_2) = f(x_3, f(x_2, h_1)) = f(x_3, f(x_2, f(x_1, h_0)))$$

$$h_3 = U^3x_1 + U^2x_2 + Ux_3 + \dots$$

$$h_T = U^Tx_1 + U^{T-1}x_2 + U^{T-2}x_3 + \dots$$

Recurrent Neural Networks : LSTM

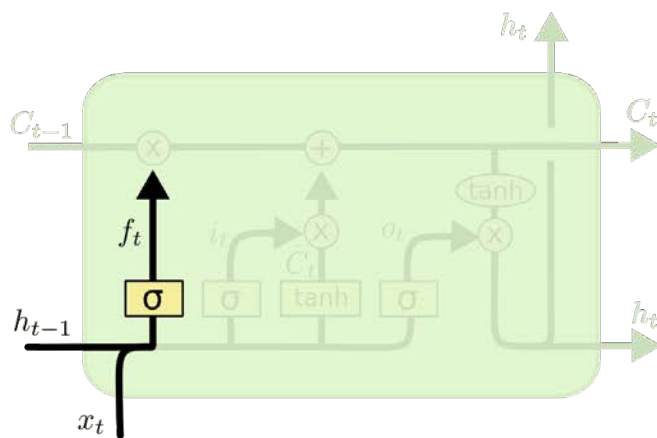
LSTM learns (1) **How to forget a memory** when the h_{t-1} and new input x_t is given,
(2) Then **how to add the new memory** from given h_{t-1} and x_t .



Figures from <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

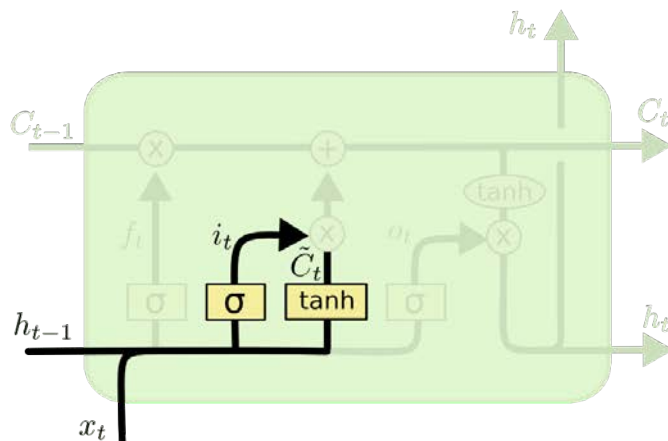
Recurrent Neural Networks : LSTM

LSTM learns (1) **How to forget a memory** when the h_{t-1} and new input x_t is given,
 (2) Then **how to add the new memory** from given h_{t-1} and x_t .



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

Forget gate



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

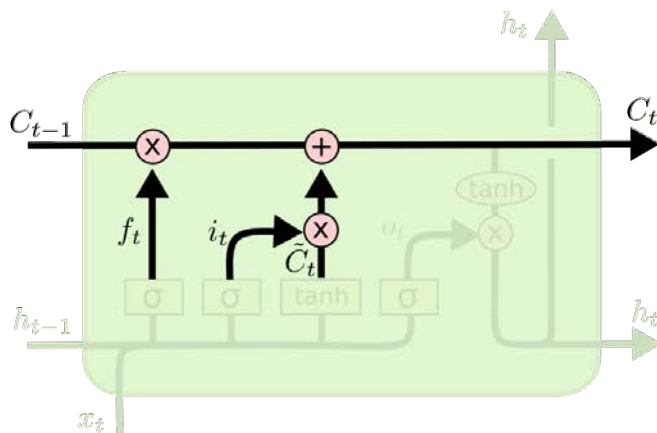
Input gate

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

Figures from <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

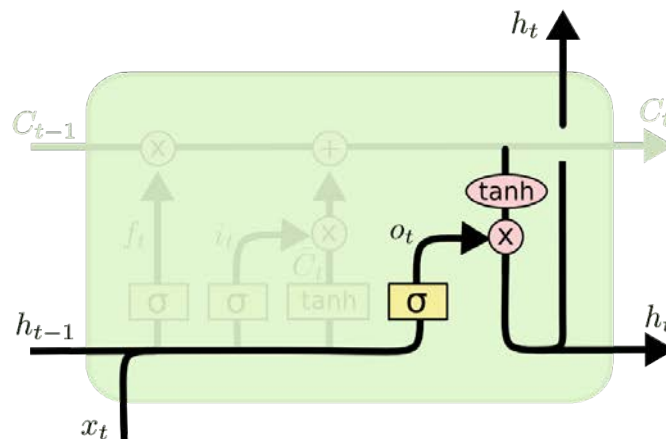
Recurrent Neural Networks : LSTM

LSTM learns (1) **How to forget a memory** when the h_{t-1} and new input x_t is given,
 (2) Then **how to add the new memory** from given h_{t-1} and x_t .



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

Cell state



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

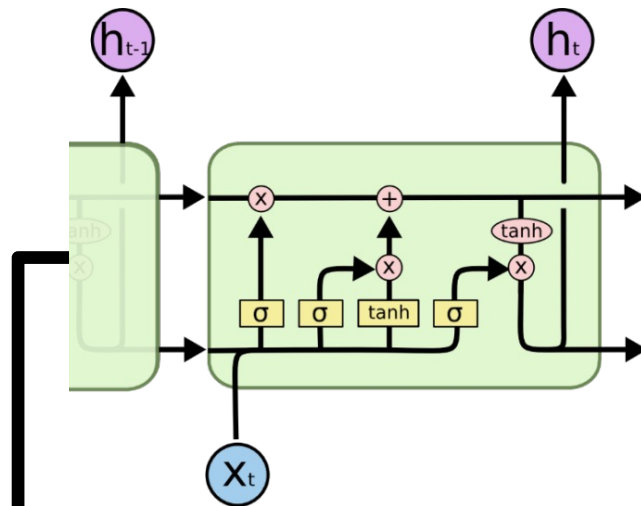
Output gate

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

Hidden state

Figures from <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

Recurrent Neural Networks : GRU



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

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

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

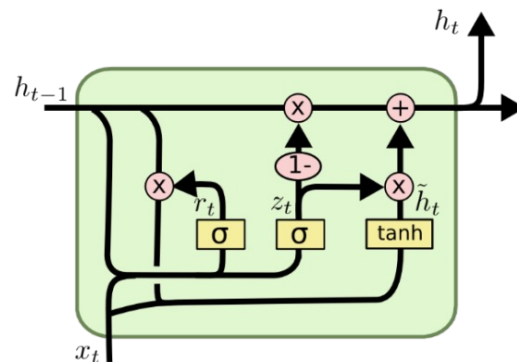
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

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

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

Maybe we can simplify this structure, efficiently!

GRU



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

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

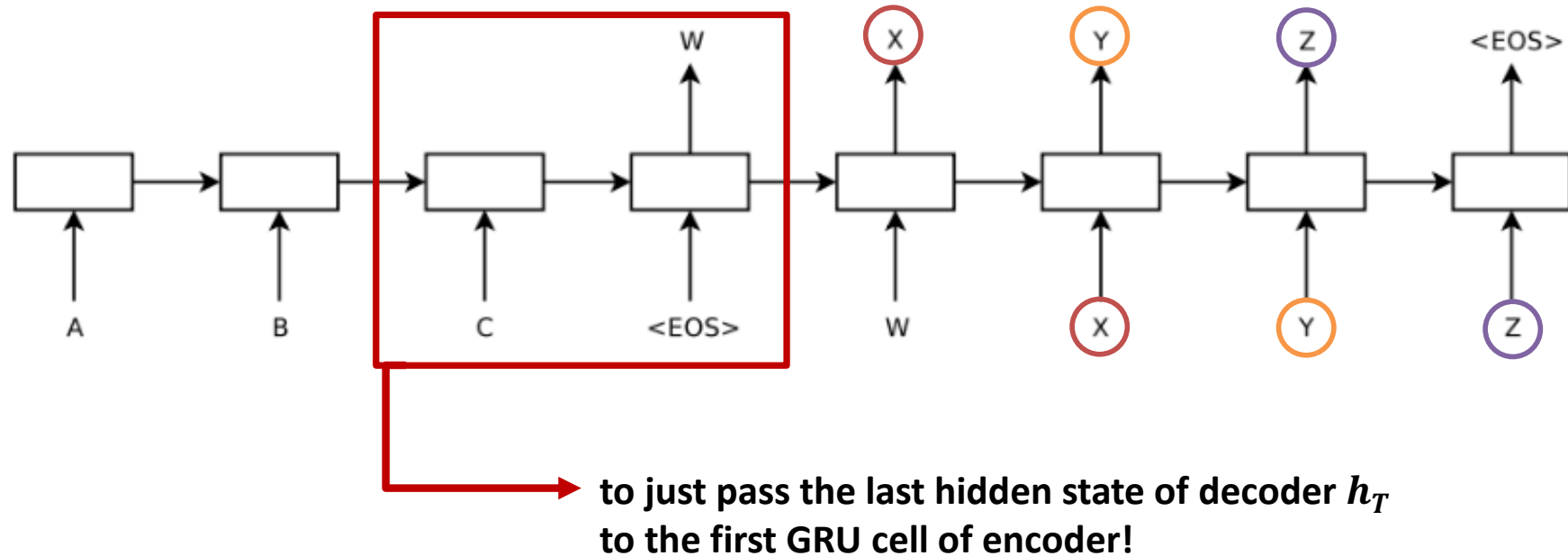
$$\tilde{h}_t = \tanh(W_h \cdot [r_t * h_{t-1}, x_t] + b_C)$$

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

update
reset

Sequence to Sequence Model: Implementation

- The simplest way to implement sequence to sequence model is



- However, this method's power gets weaker when the encoder need to generate a longer sequence.

Transformer

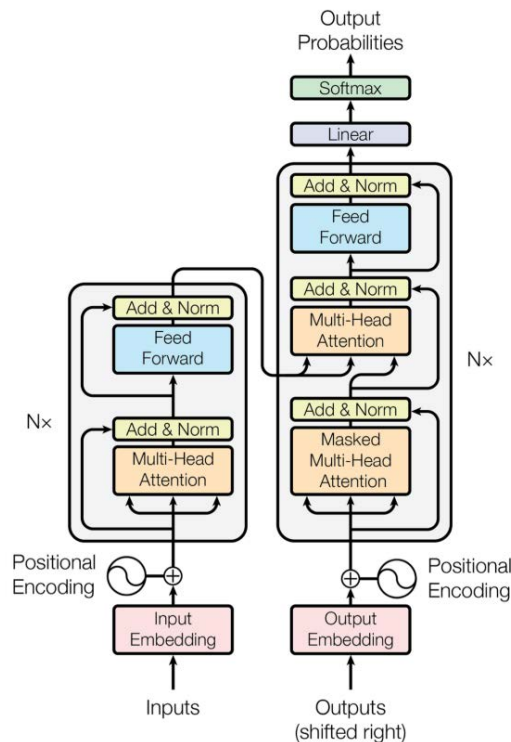
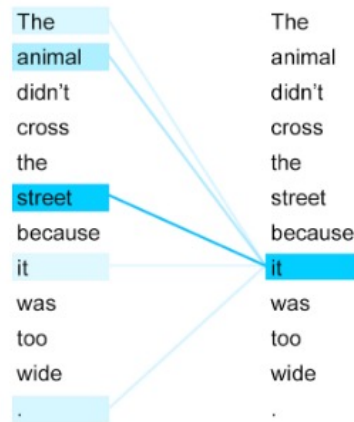


Figure 1: The Transformer - model architecture.



$$\text{Attention}(Q, K, V) = \text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) V$$

Other NLP Models:

- Bidirectional Encoder Representations from Transformers (BERT) [2]
- XLNet
- RoBERTa
- ALBERT
- StructBERT
- ...

[1] Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." In *Advances in neural information processing systems*, pp. 5998-6008. 2017.

[2] Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." In *NAACL-HLT (1)*. 2019.