Introduction to Deep Learning

Recurrent Neural Networks

Prof. Songhwai Oh

ECE, SNU

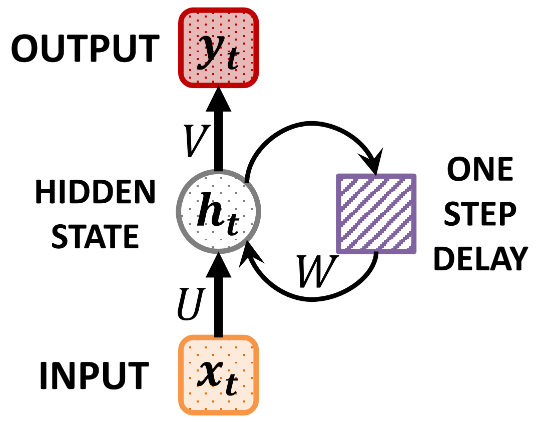
Adapted from slides by Hyemin Ahn

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



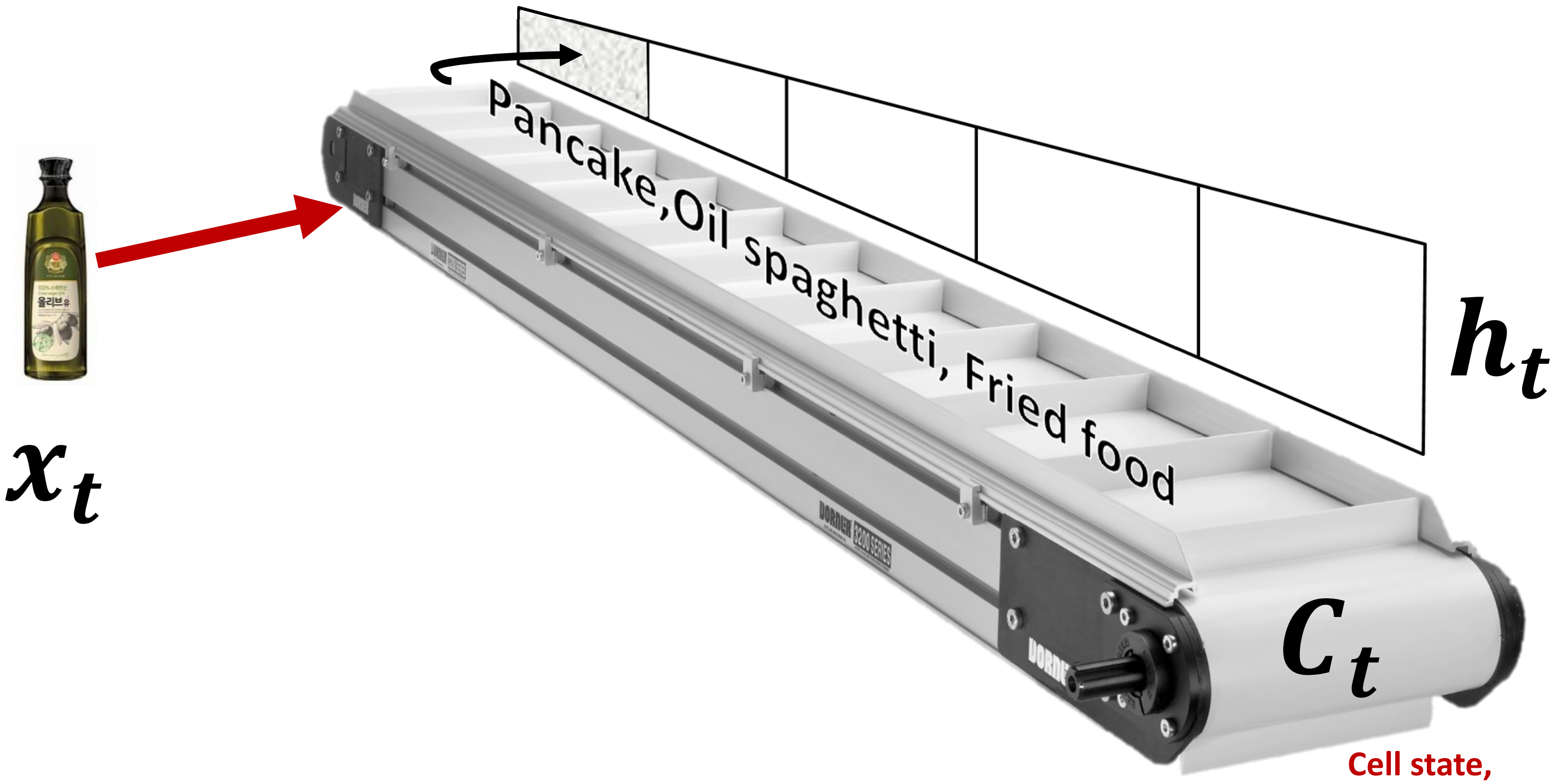
$$h_t = f(Ux_t + Wh_{t-1} + b)$$
$$y_t = Vh_t + c$$

- RNNs are called **RECURRENT** because they **perform the same task for every element of a sequence**, with the output being **depended 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 : LSTM

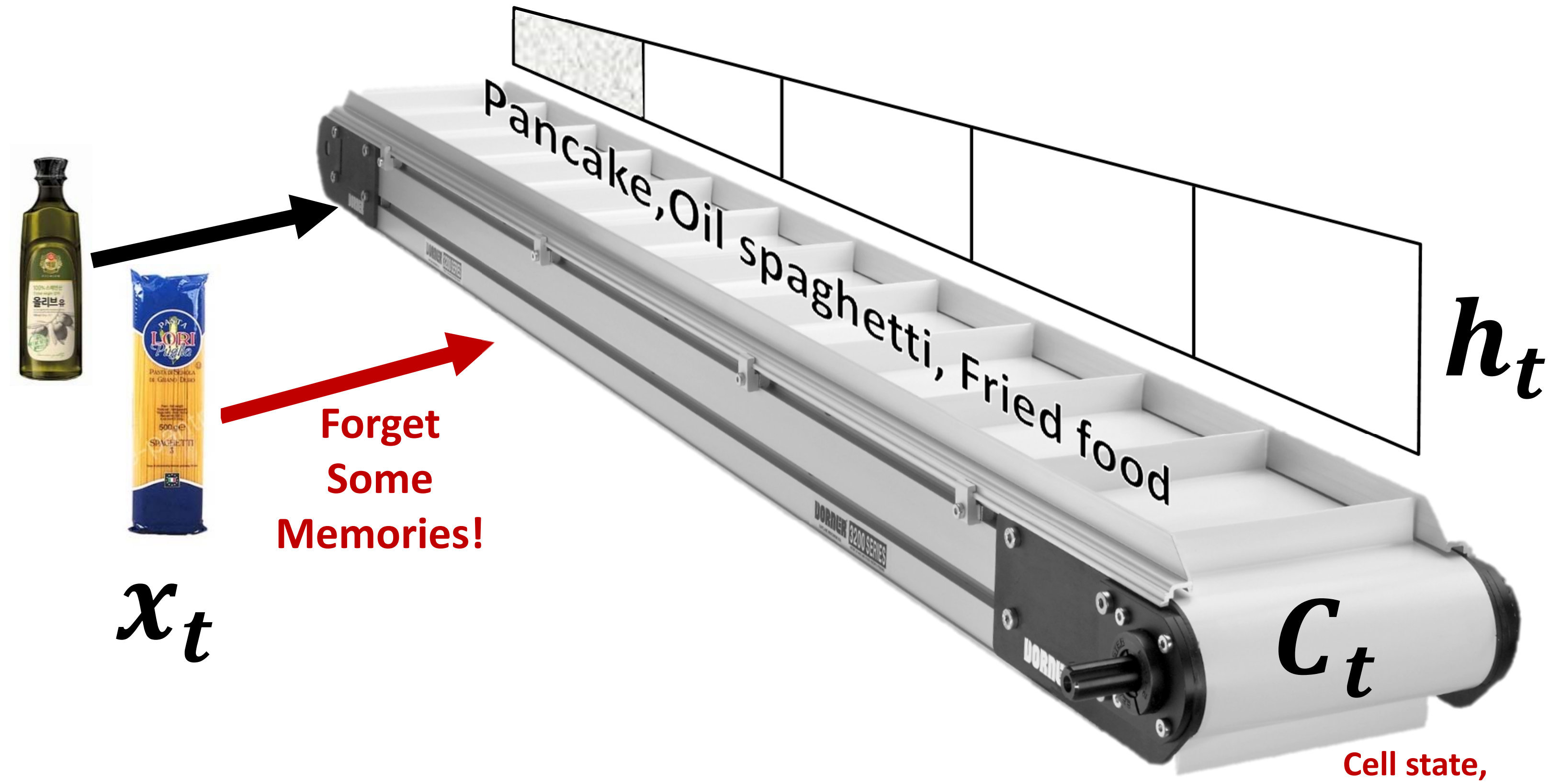
- Let’s think about the machine, which guesses the dinner menu from stuffs in a shopping bag.

Recurrent Neural Networks : LSTM



Cell state,
Internal memory unit,
Like a conveyor belt!

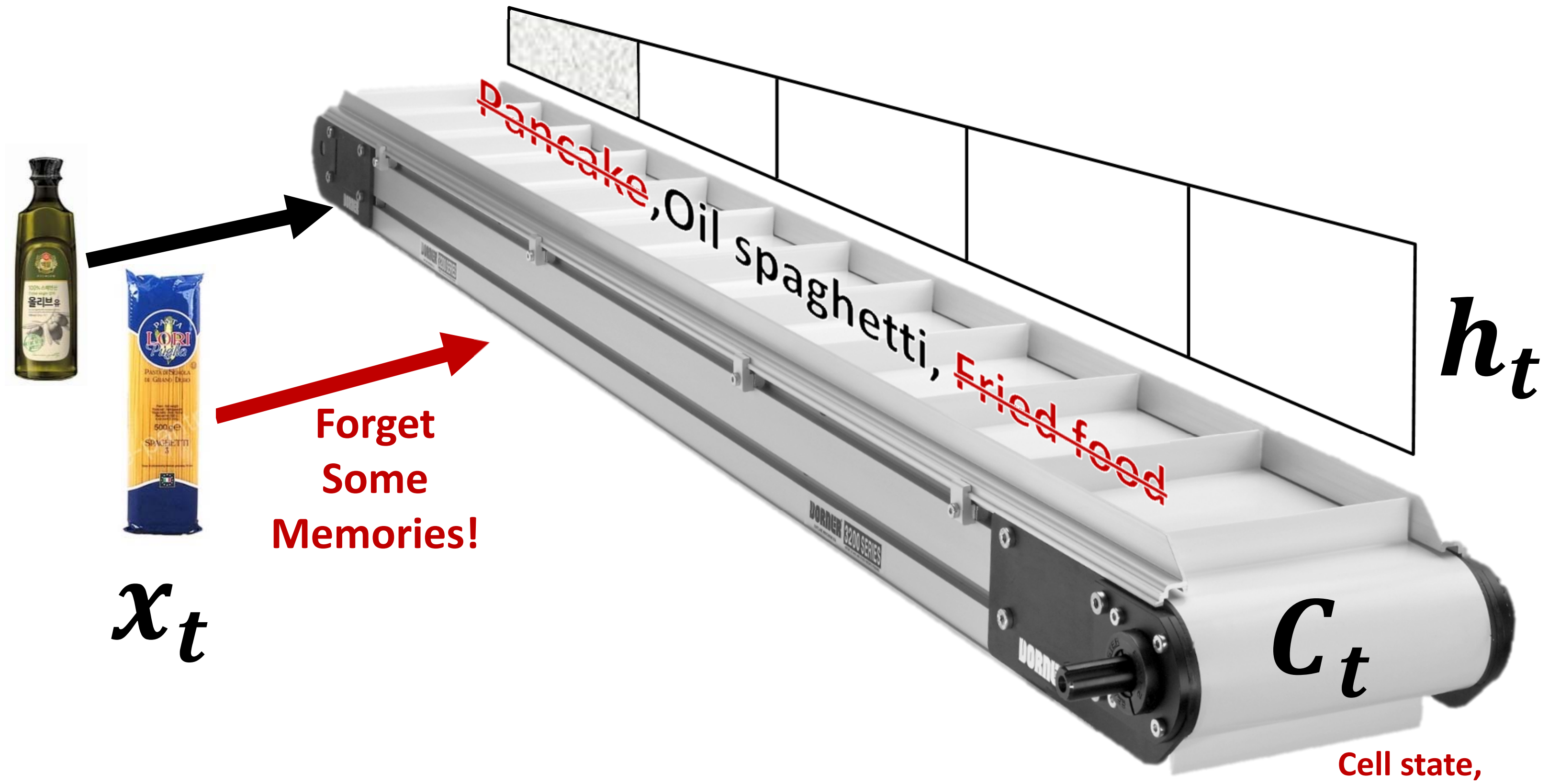
Recurrent Neural Networks : LSTM



Cell state,
Internal memory unit,
Like a conveyor belt!

Recurrent Neural Networks : LSTM

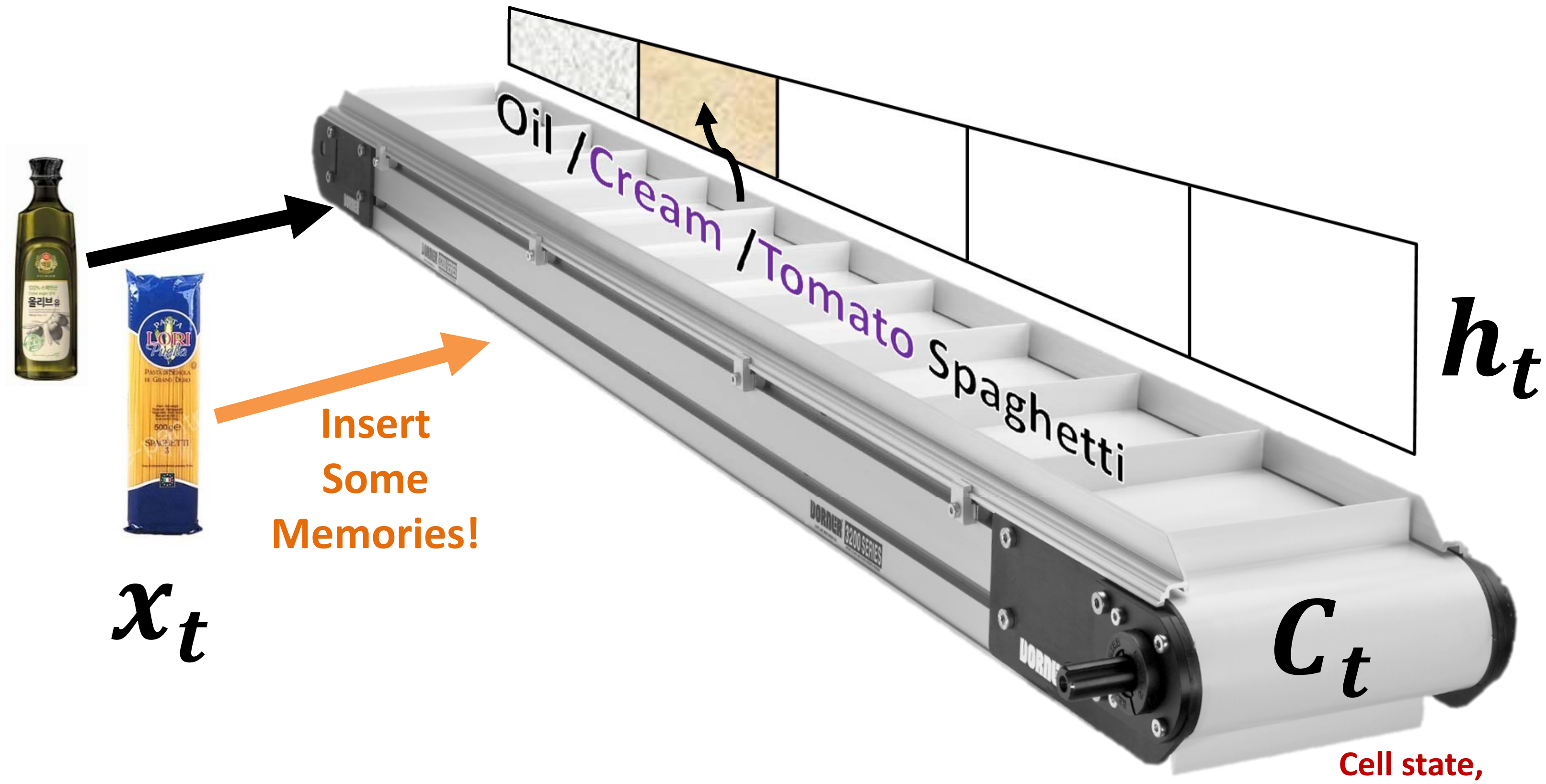
LSTM learns (1) **How to forget a memory** when the h_{t-1} and new input x_t is given,
(2) Then **how to add the new memory** from given h_{t-1} and x_t.



Cell state,
Internal memory unit,
Like a conveyor belt!

Recurrent Neural Networks : LSTM

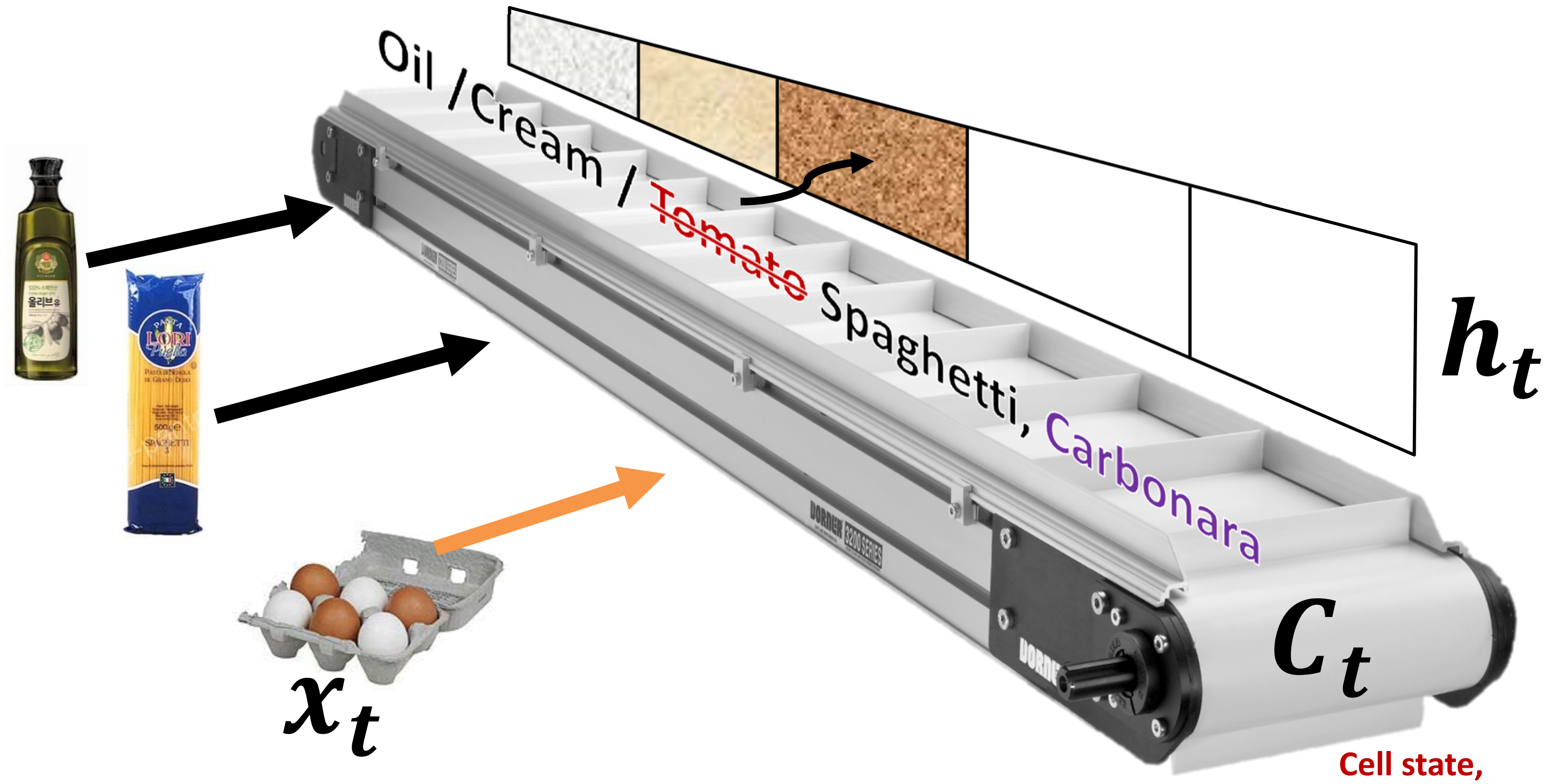
LSTM learns (1) **How to forget a memory** when the h_{t-1} and new input x_t is given,
(2) Then **how to add the new memory** from given h_{t-1} and x_t.



Cell state,
Internal memory unit,
Like a conveyor belt!

Recurrent Neural Networks : LSTM

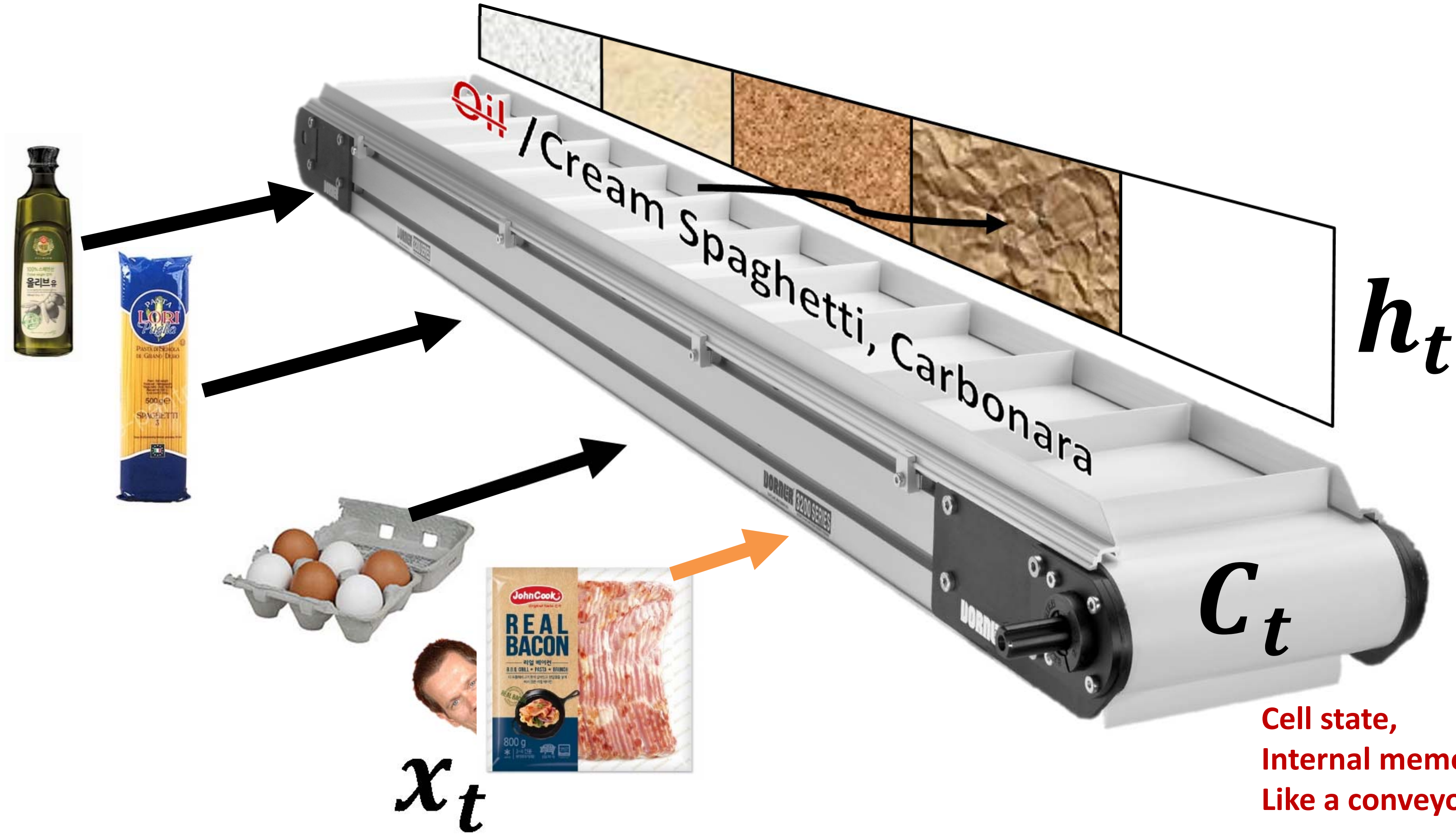
LSTM learns (1) **How to forget a memory** when the h_{t-1} and new input x_t is given,
(2) Then **how to add the new memory** from given h_{t-1} and x_t.



Cell state,
Internal memory unit,
Like a conveyor belt!

Recurrent Neural Networks : LSTM

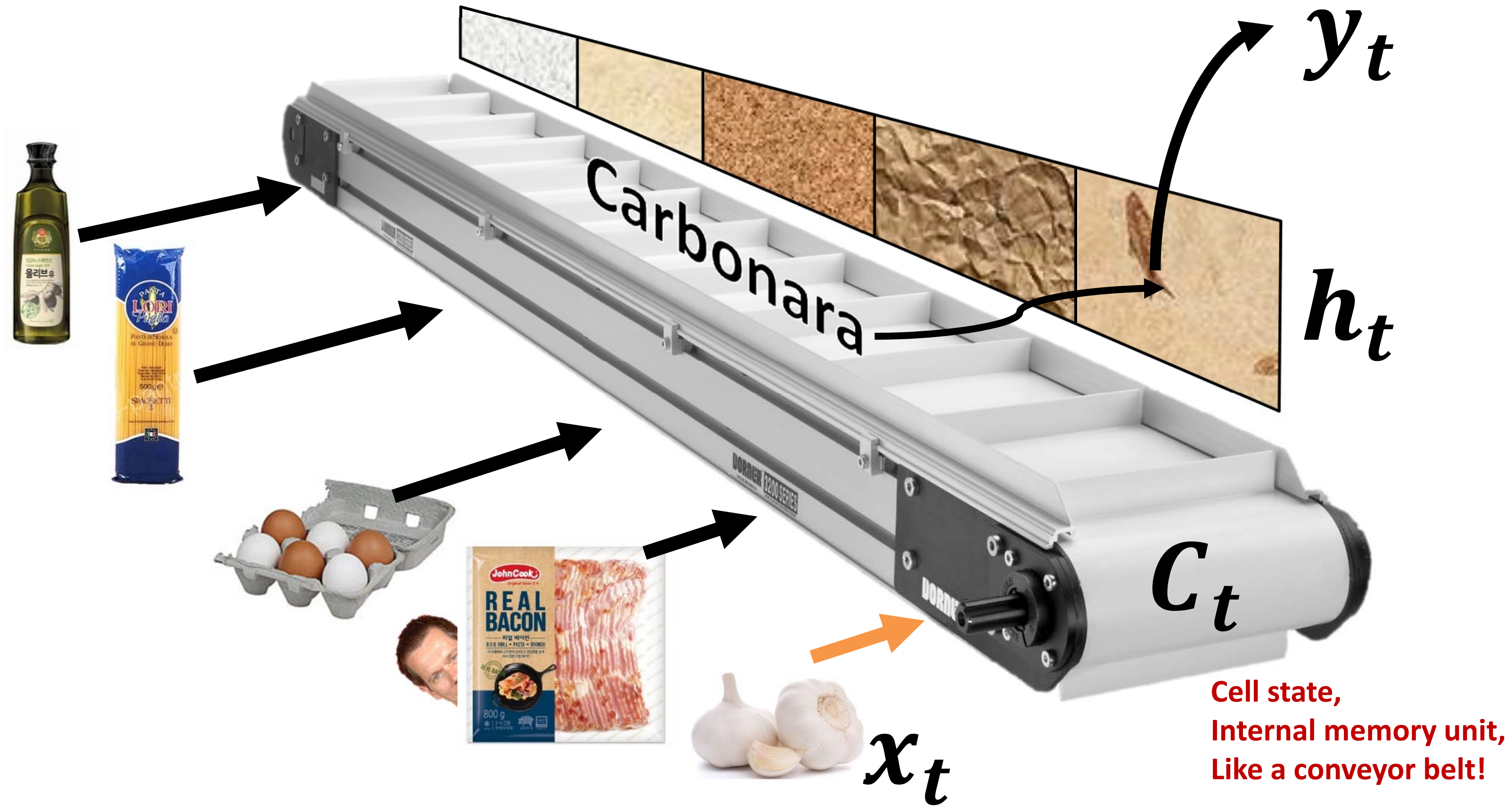
LSTM learns (1) **How to forget a memory** when the h_{t-1} and new input x_t is given,
(2) Then **how to add the new memory** from given h_{t-1} and x_t.



Cell state,
Internal memory unit,
Like a conveyor belt!

Recurrent Neural Networks : LSTM

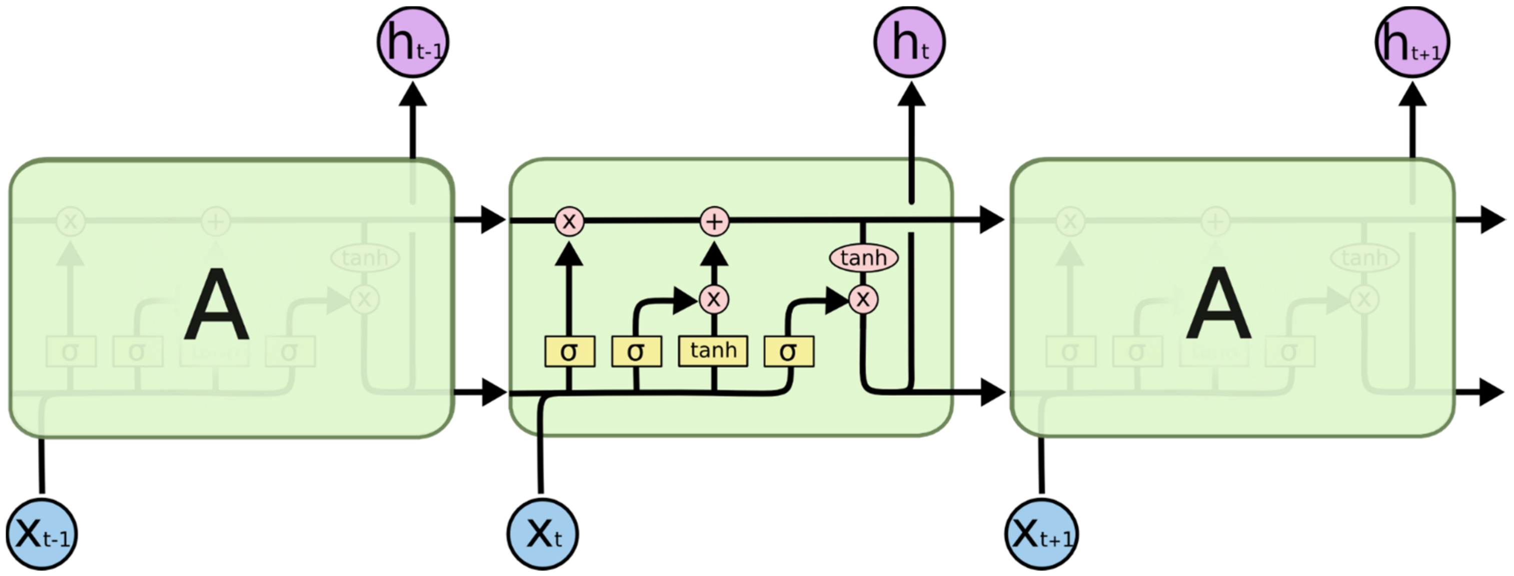
LSTM learns (1) **How to forget a memory** when the h_{t-1} and new input x_t is given,
(2) Then **how to add the new memory** from given h_{t-1} and x_t.



Cell state,
Internal memory unit,
Like a conveyor belt!

Recurrent Neural Networks : LSTM

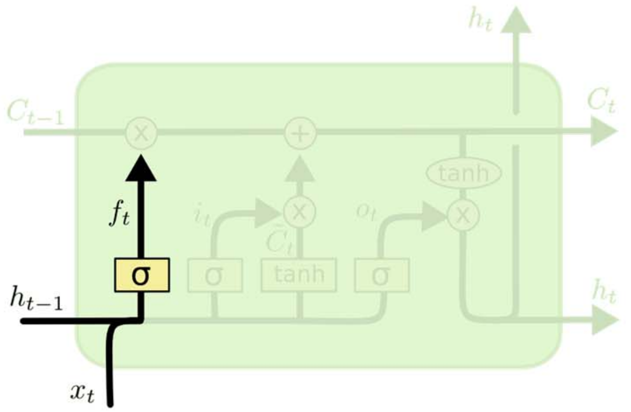
LSTM learns (1) **How to forget a memory** when the h_{t-1} and new input x_t is given,
(2) Then **how to add the new memory** from given h_{t-1} and x_t.



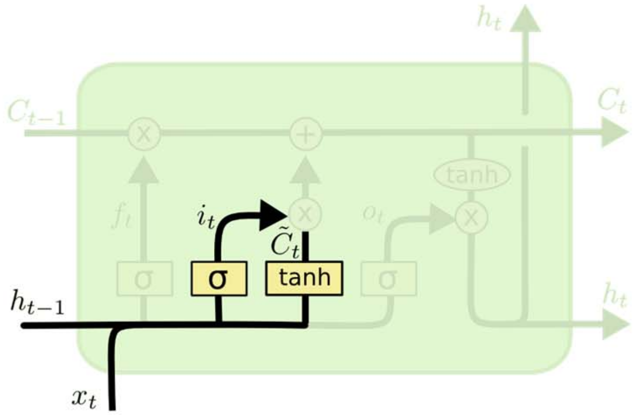
Figures from http://colah.github.io/posts/2015-08-Understanding-LSTMs/

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.



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



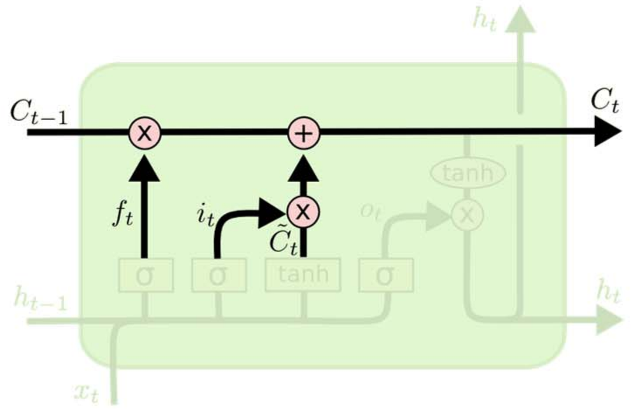
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$ Input gate

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

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

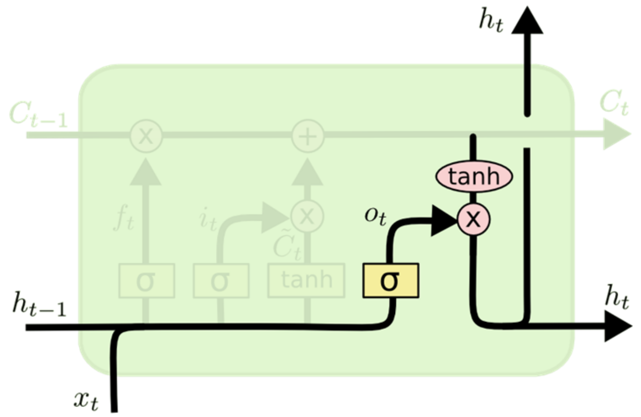
Recurrent Neural Networks : LSTM

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



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

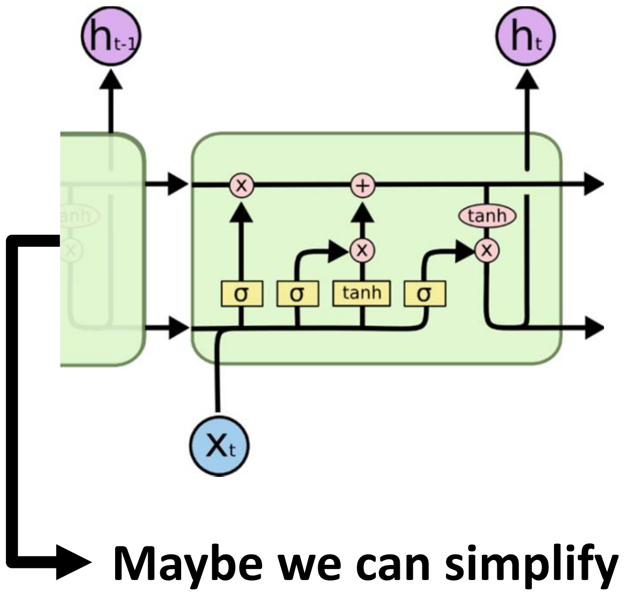
Output gate

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

Hidden state

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

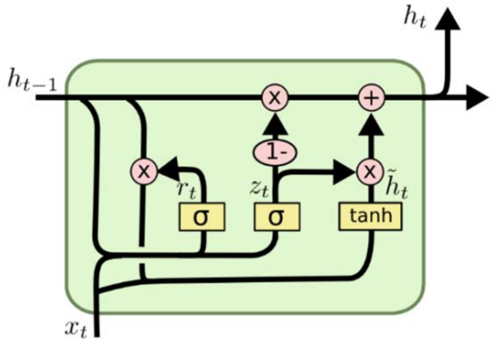
Recurrent Neural Networks : GRU



$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$
$$h_t = o_t * \tanh(C_t)$$

Maybe we can simplify this structure, efficiently!

GRU



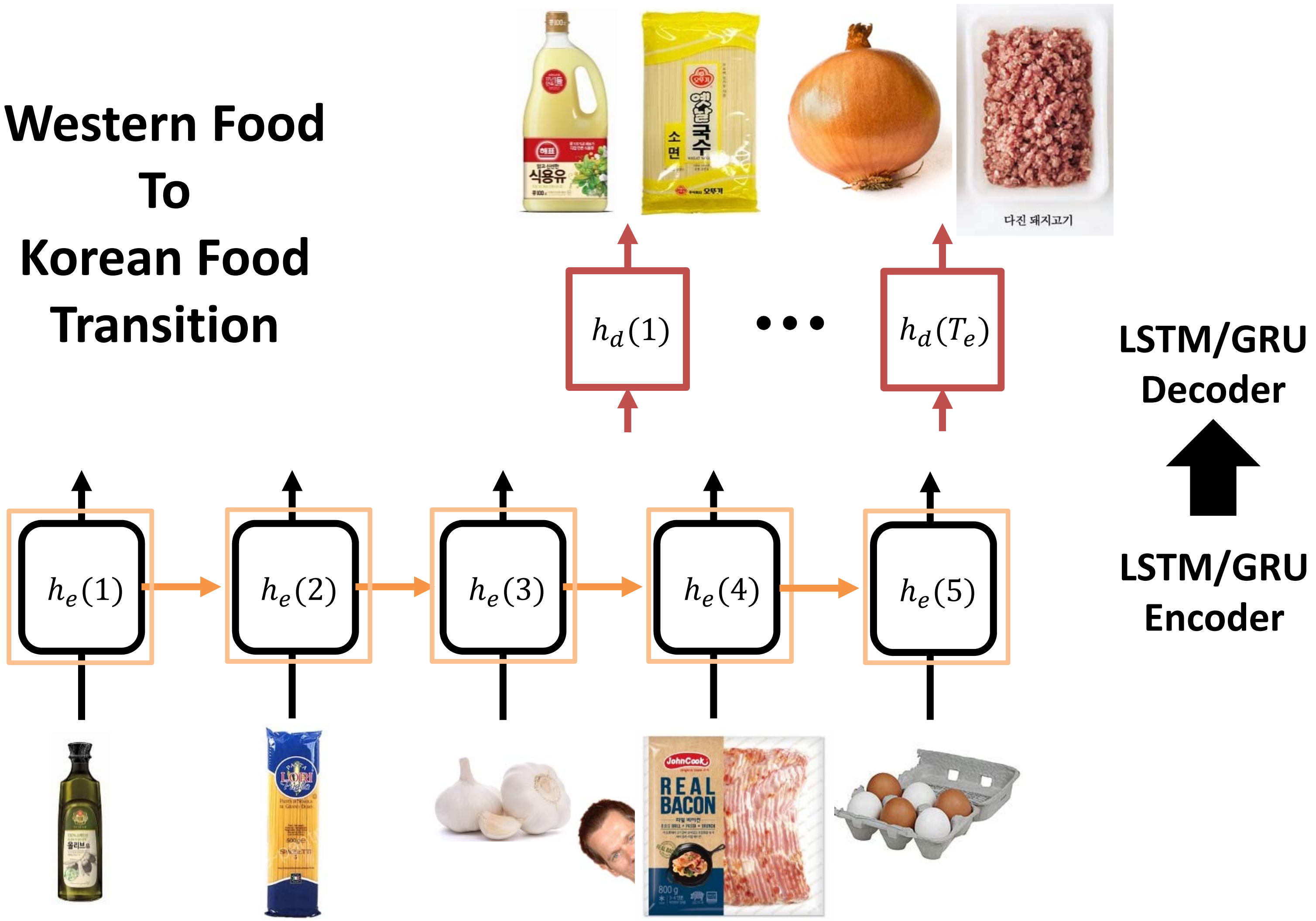
$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z)$$
$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r)$$
$$\tilde{h}_t = \tanh(W_h \cdot [r_t * h_{t-1}, x_t] + b_C)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

update
reset

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

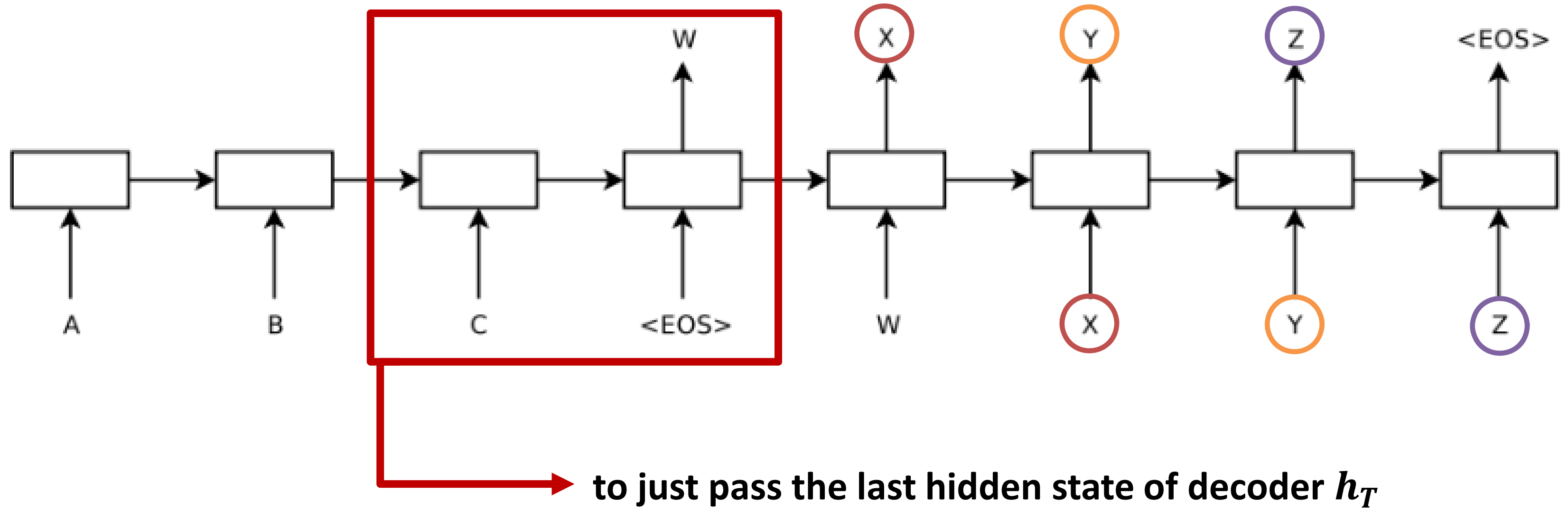
Sequence to Sequence Model: What is it?

Western Food
To
Korean Food
Transition



Sequence to Sequence Model: Implementation

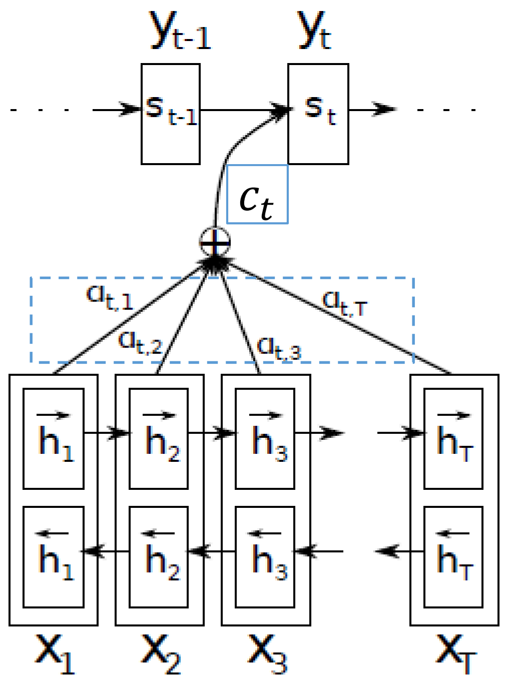
- The simplest way to implement sequence to sequence model is



to just pass the last hidden state of decoder h_T to the first GRU cell of encoder!

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

Sequence to Sequence Model: Attention Decoder



Attention GRU Decoder

Bidirectional GRU Encoder

- For each GRU cell consisting the decoder, let's **pass the encoder's information differently**!

$$h_i = \begin{bmatrix} \vec{h}_i \\ \overleftarrow{h}_i \end{bmatrix} \qquad c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

$$\begin{aligned} s_i &= f(s_{i-1}, y_{i-1}, c_i) \\ &= (1 - z_i) * s_{i-1} + z_i * \tilde{s}_i \\ z_i &= \sigma(W_z y_{i-1} + U_z s_{i-1}) \\ r_i &= \sigma(W_r y_{i-1} + U_r s_{i-1}) \\ \tilde{s}_i &= \tanh(y_{i-1} + U[r_i * s_{i-1}] + C c_i) \end{aligned}$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$$

$$e_{ij} = v_a^T \tanh\left(W_a s_{i-1} + U_a h_j\right)$$

Wrap Up

- Recurrent neural networks
- LSTM
- GRU
- Seq2Seq