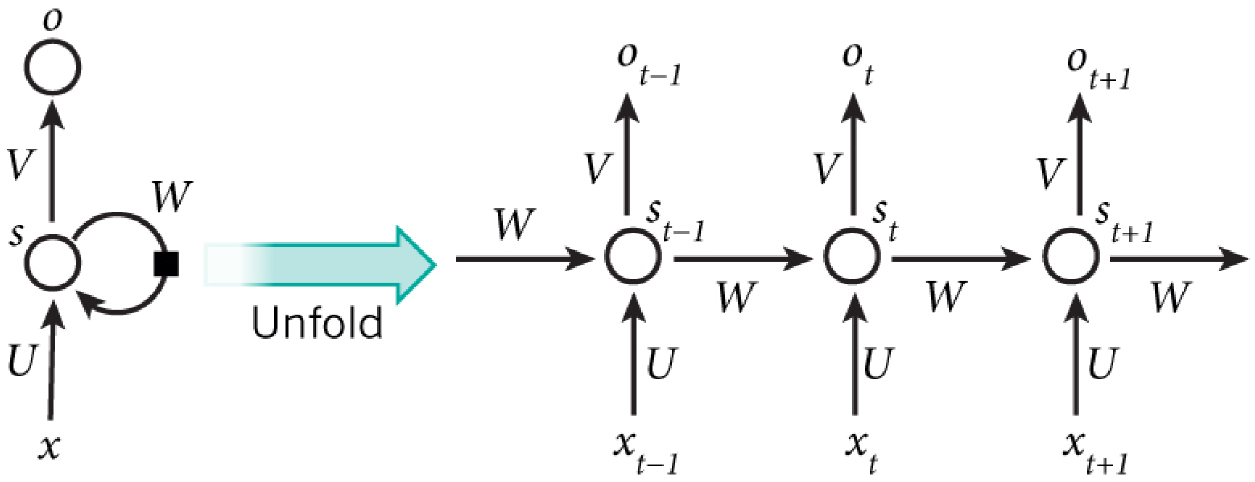
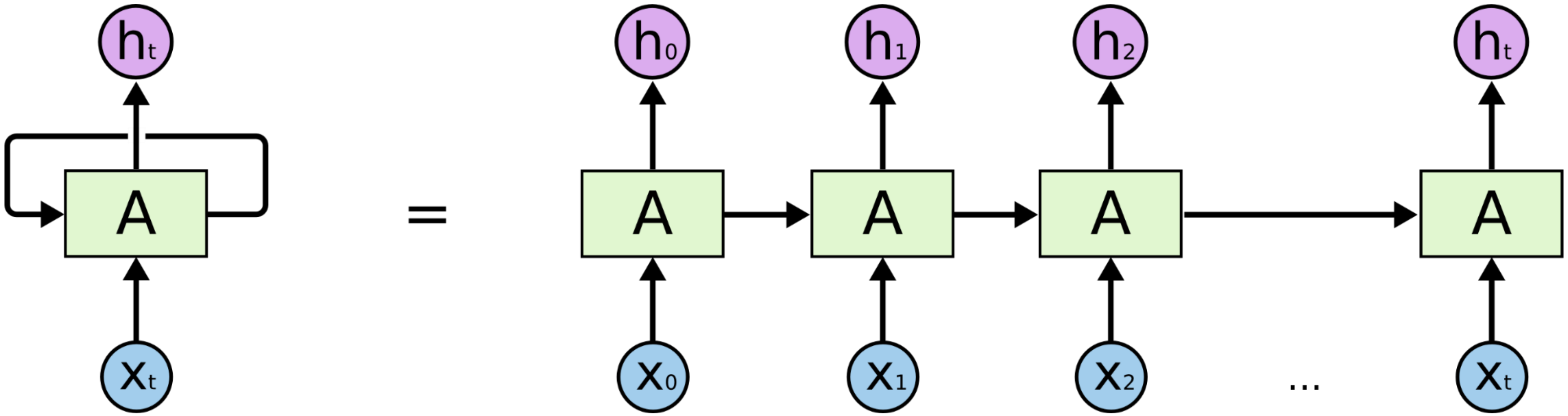
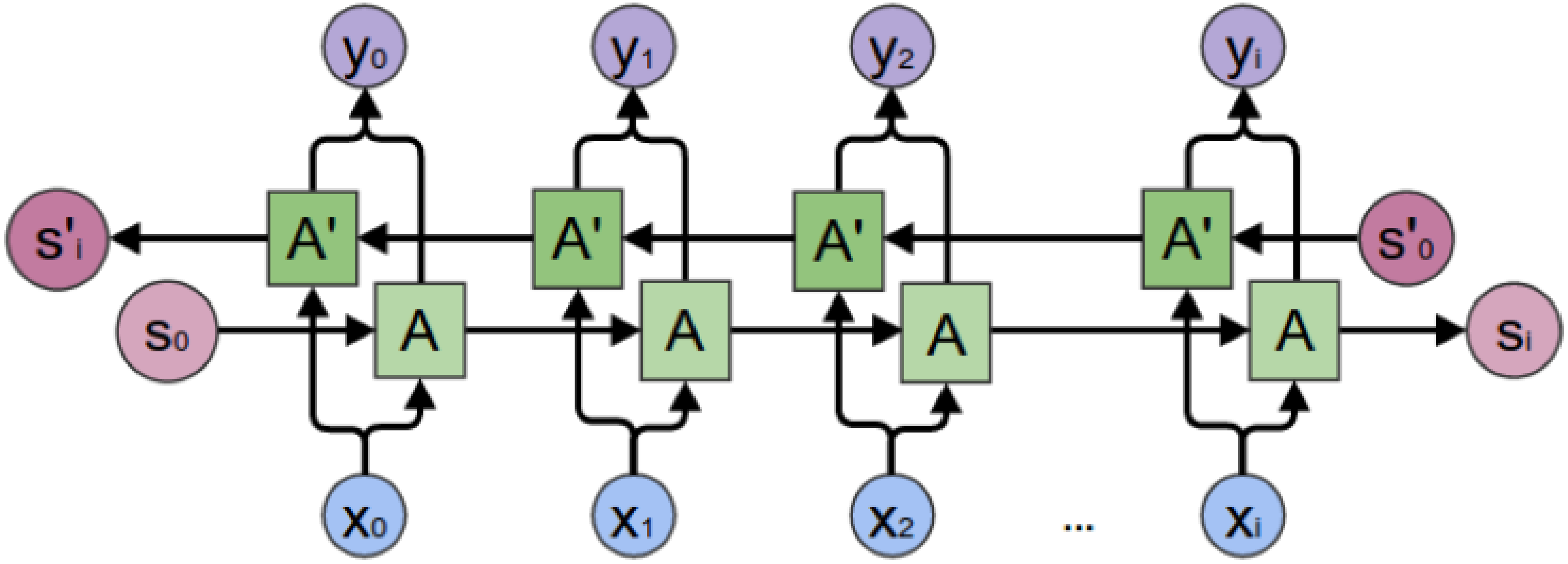
Deep Recurrent neural networks and LSTM

Recurrent Neural Network



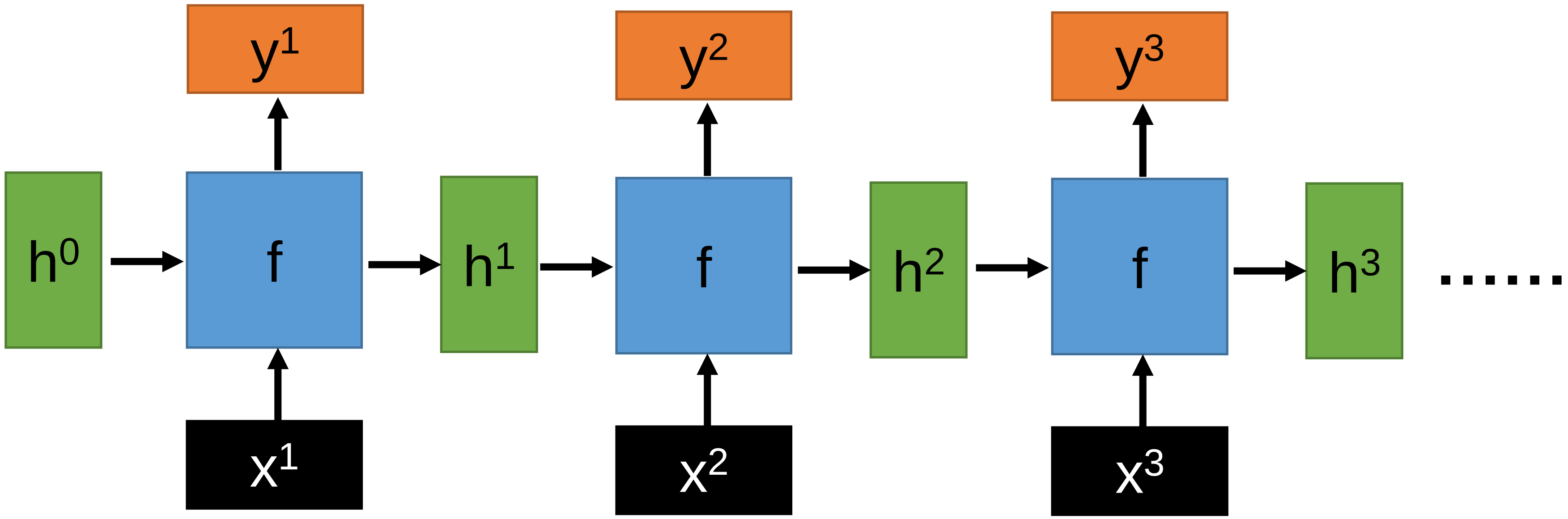


Bidirectional RNN

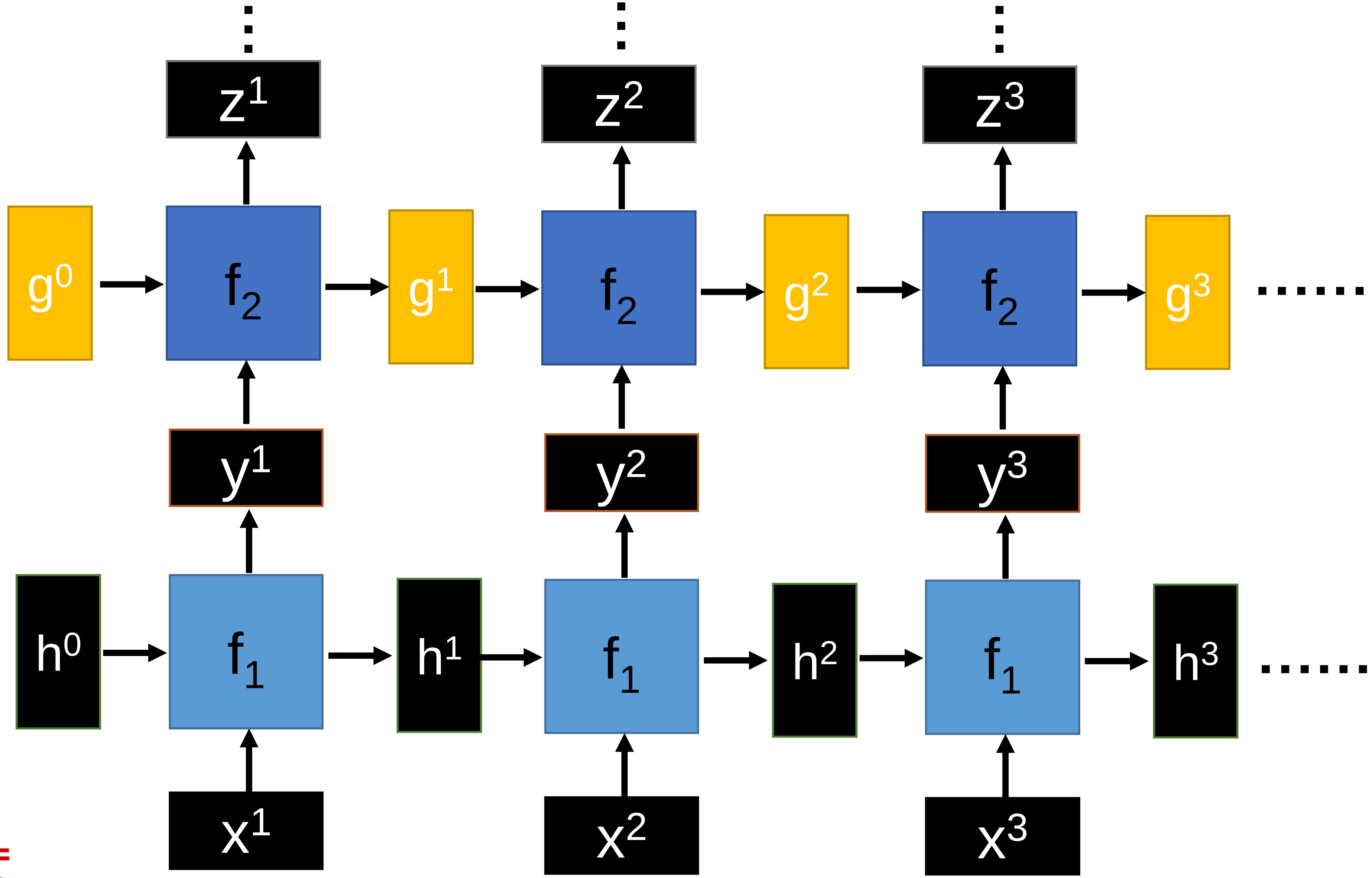


- 下一個字的出現決定於之前的字
- 前一個字的出現也決定於後面的字

- Given function f: $h^t, y^t = f(h^{t-1}, x)$



Deep RNN



$g^t, z^t = f_2(h^{t-1}, y^t)$

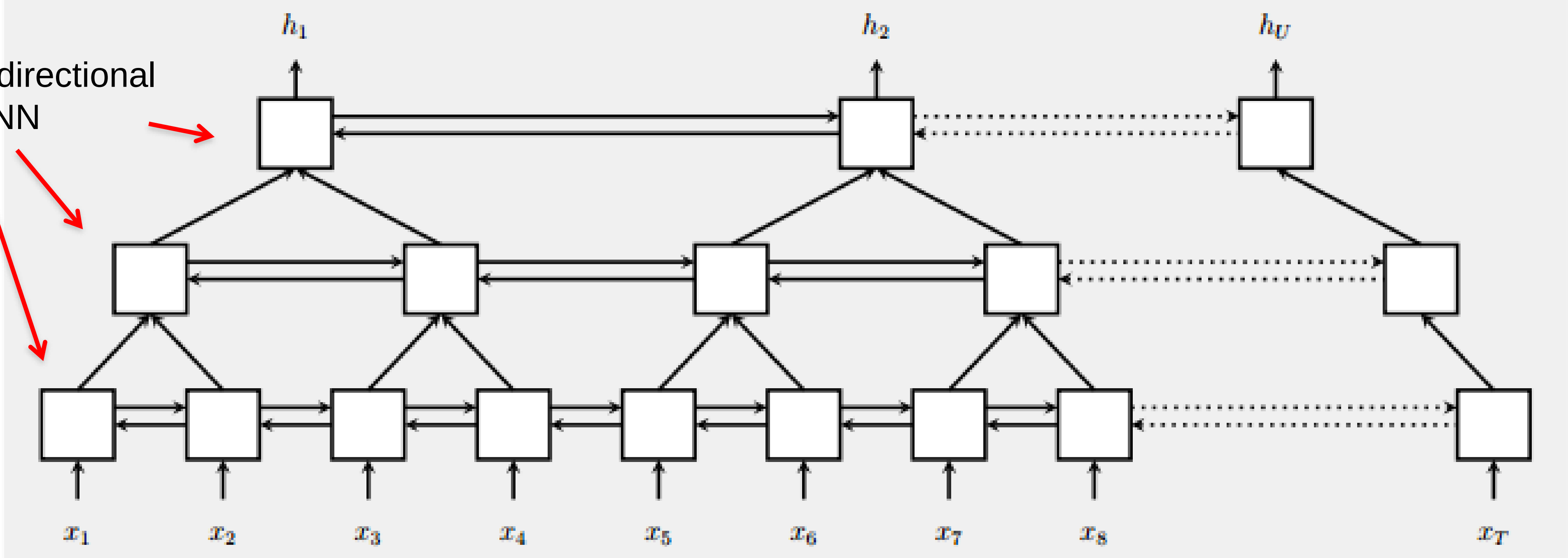
$h^t, y^t = f_1(h^{t-1}, x^t)$

Pyramid RNN

Significantly speed up training

- Reducing the number of time steps

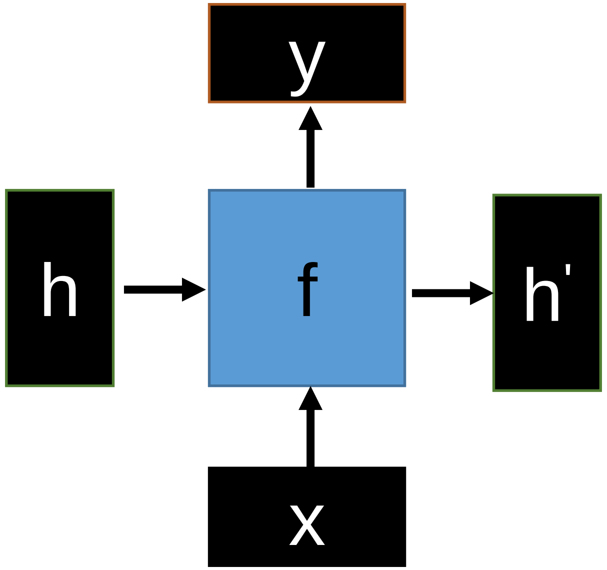
Bidirectional RNN



W. Chan, N. Jaitly, Q. Le and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," ICASSP, 2016

Naïve RNN

- Given function f: h’,y=f(h,x)



$$h' = \sigma(W^h h + W^i x)$$

$$y = \sigma(W^o h')$$

softmax

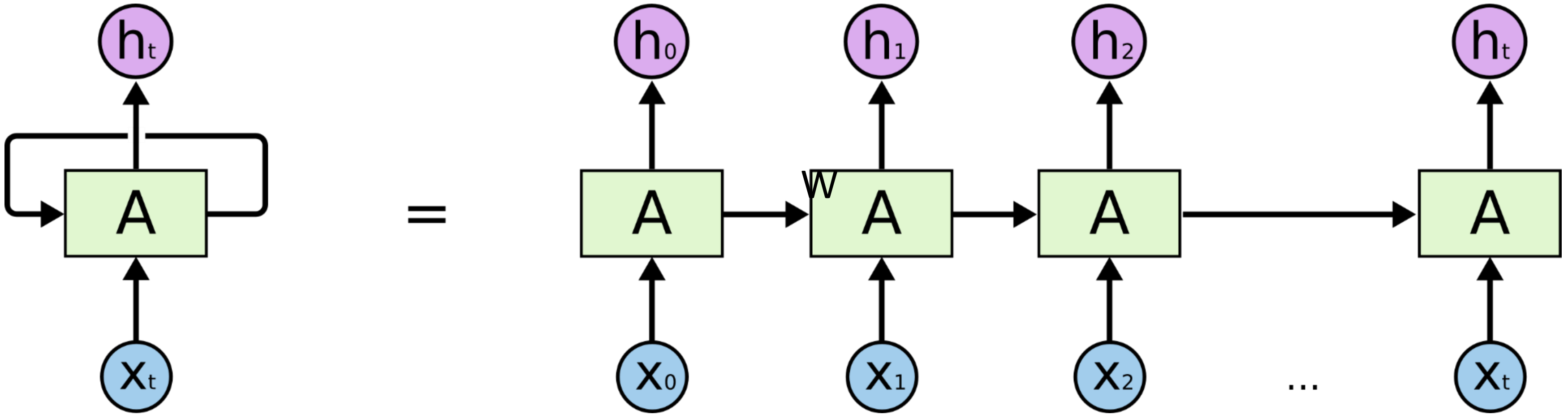
Note, y is computed from h’

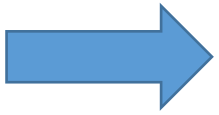
We have ignored the bias

Problems with traditional RNN

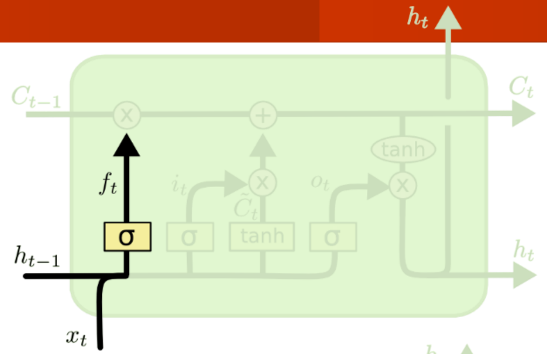
- When dealing with a time series, it tends to forget old information. When there is a distant relationship of unknown length, we wish to have a “memory” to it.
- Vanishing gradient problem.

W 有被連續乘多次之效果

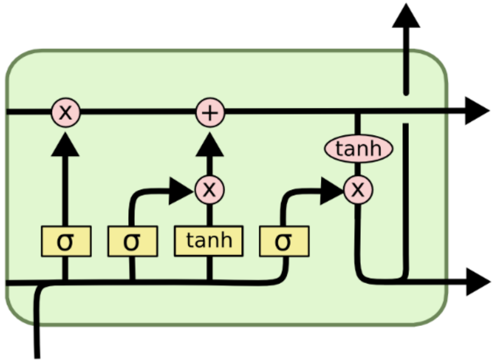


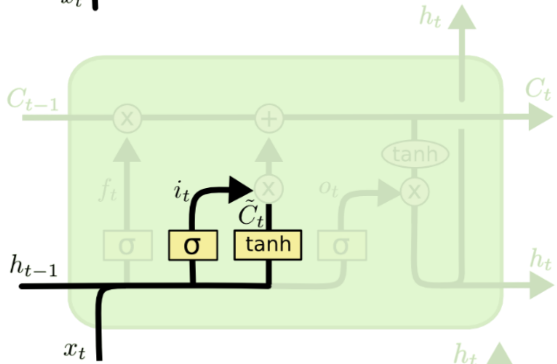


Use a storage for retaining the information, and use a gate deciding how much information being retained



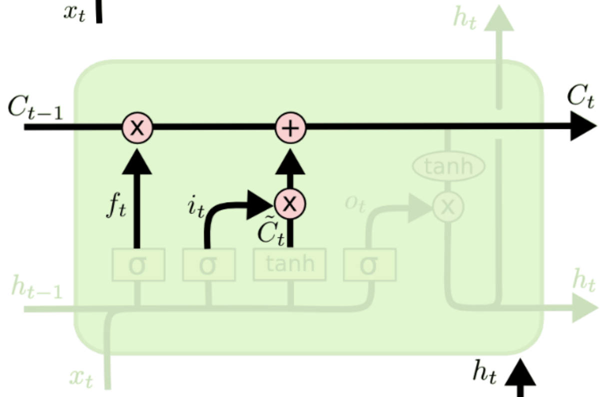
$$\boxed{f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \;+\; b_f\right)}$$





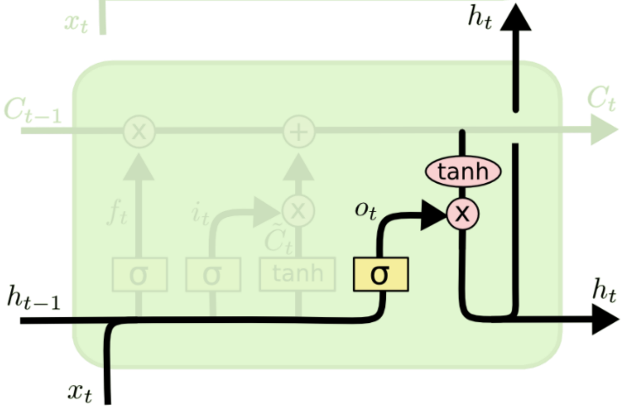
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \;+\; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \;+\; b_C)$$

i_t decides what component is to be updated.
C'_t provides change contents



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

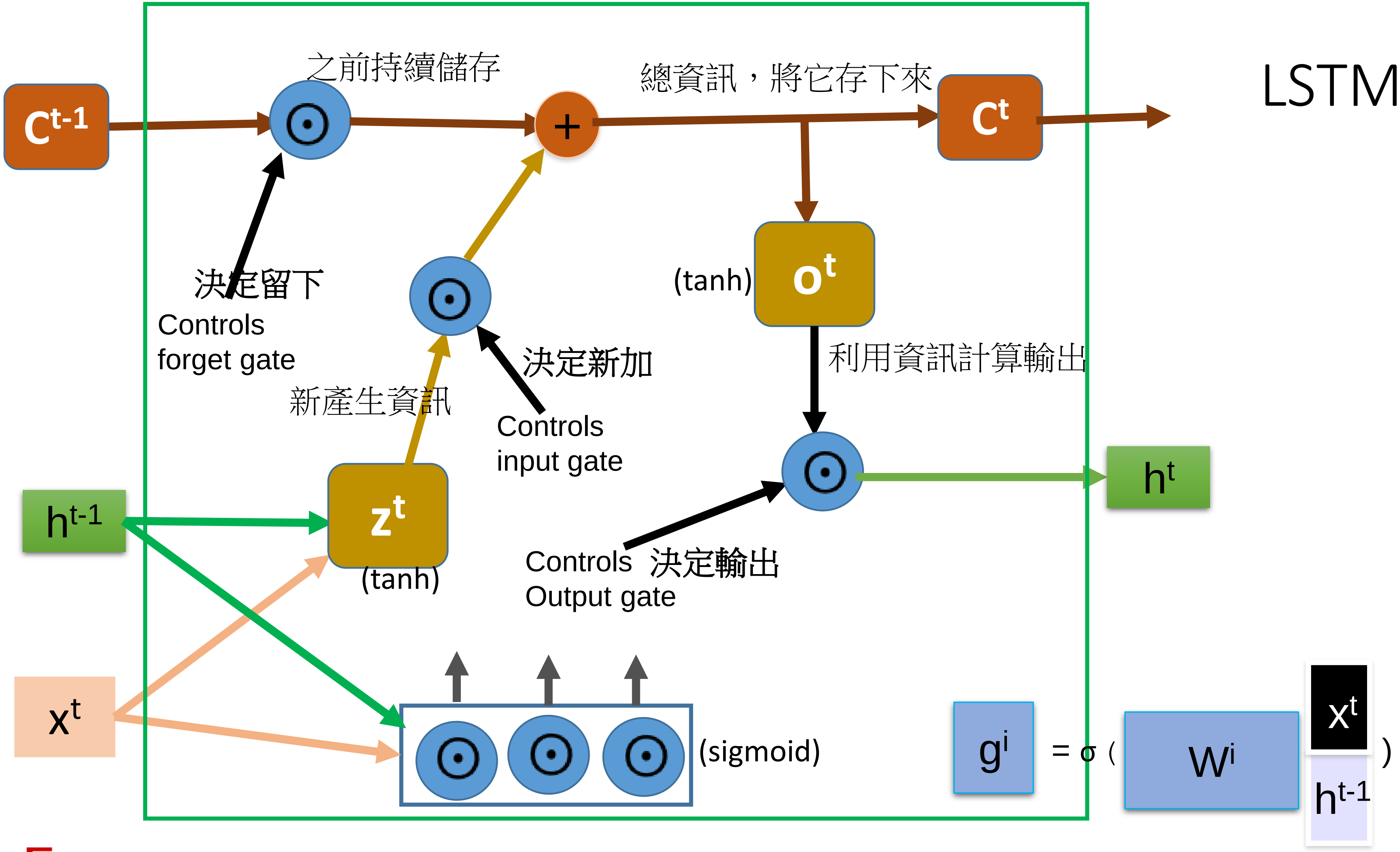
Updating the cell state



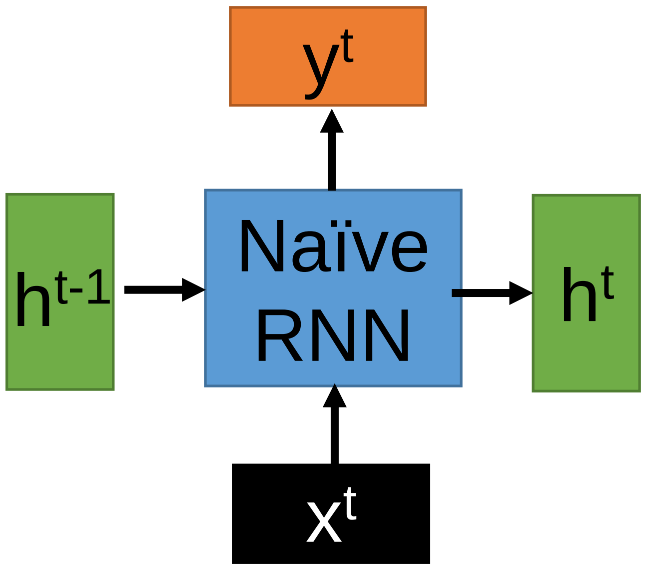
$$o_t = \sigma\left(W_o\; [h_{t-1}, x_t] \;+\; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

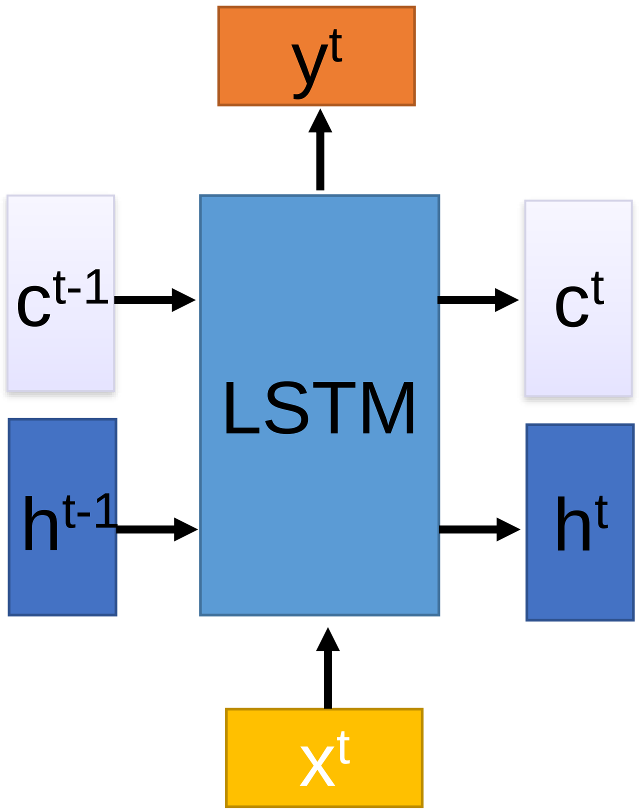
Decide what part of the cell state to output

LSTM


之前持續儲存
總資訊，將它存下來
C^{t-1}
C^t
決定留下
Controls
forget gate
新產生資訊
決定新加
Controls
input gate
(tanh)
o^t
利用資訊計算輸出
h^t
h^{t-1}
z^t
(tanh)
Controls 決定輸出
Output gate
x^t
(sigmoid)
$g^i = \sigma(W^i \begin{bmatrix} x^t \\ h^{t-1} \end{bmatrix})$

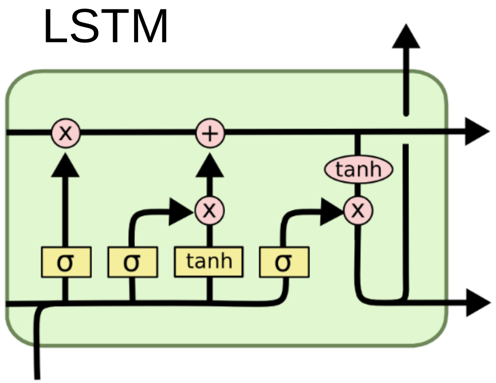
Naïve RNN(傳統) vs LSTM





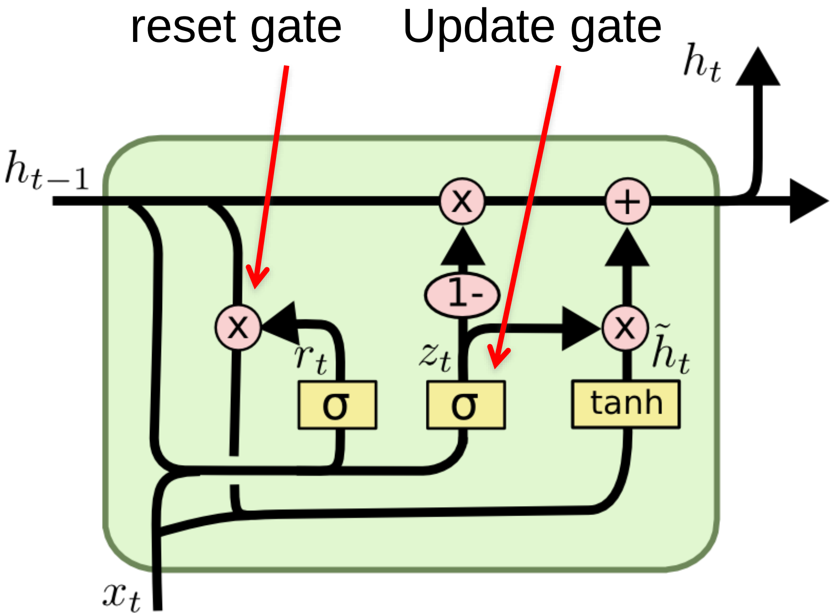
h changes faster, h^t and h^{t-1} can be very different

➡ Using storage c to control its change rate, c^t is c^{t-1} added by something



GRU – gated recurrent unit

(more compression)



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

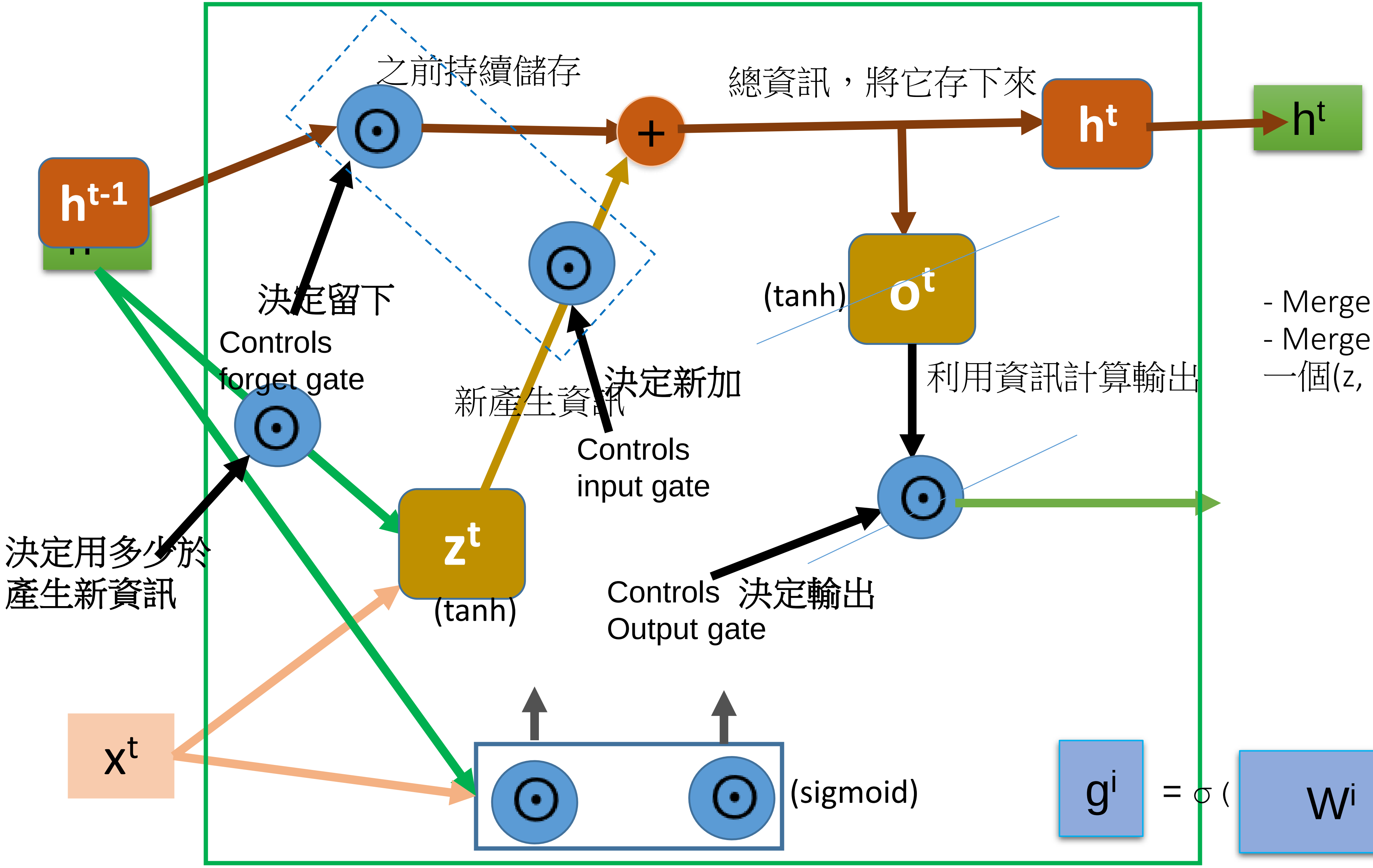
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

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

It combines the forget and input into a single update gate.
It also merges the cell state and hidden state. This is simpler than LSTM. There are many other variants too.

X,*: element-wise multiply

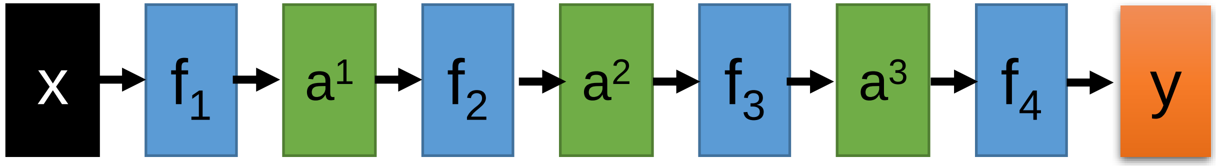


- Merge C and h
- Merge forget and input成為一個(z, 1-z)

$$g^i = \sigma\left(W^i \begin{bmatrix} x^t \\ h^{t-1} \end{bmatrix} \right)$$

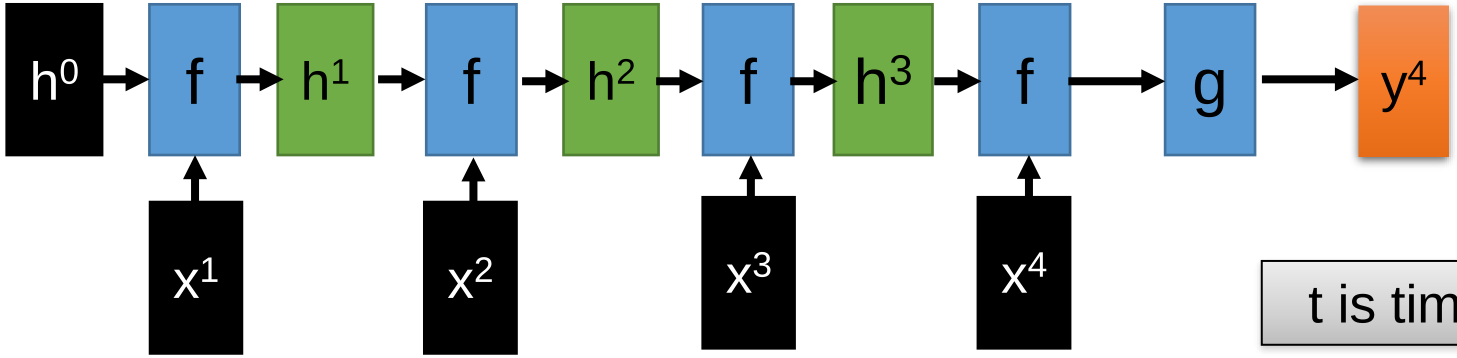
Feed-forward vs Recurrent Network

1. Feedforward network does not have input at each step
2. Feedforward network has different parameters for each layer



t is layer

$$a^t = f_t(a^{t-1}) = \sigma(W^t a^{t-1} + b^t)$$



t is time step

$$a^t = f(a^{t-1}, x^t) = \sigma(W^h a^{t-1} + W^i x^t + b^i)$$

We will turn the recurrent network 90 degrees.

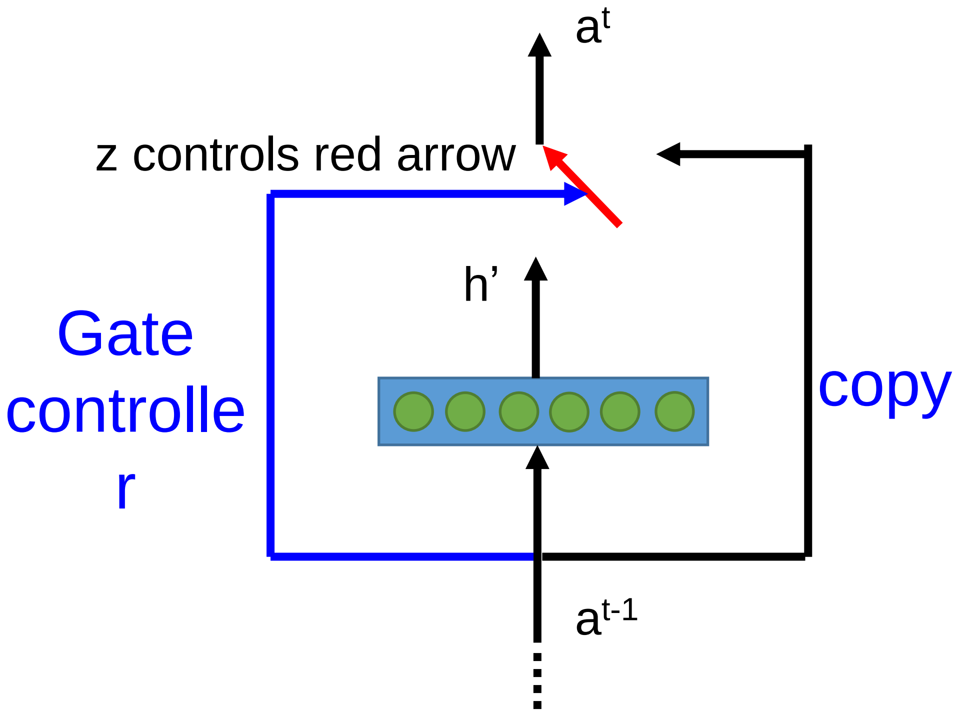
Highway Network

$h' = \sigma(W a^{t-1})$

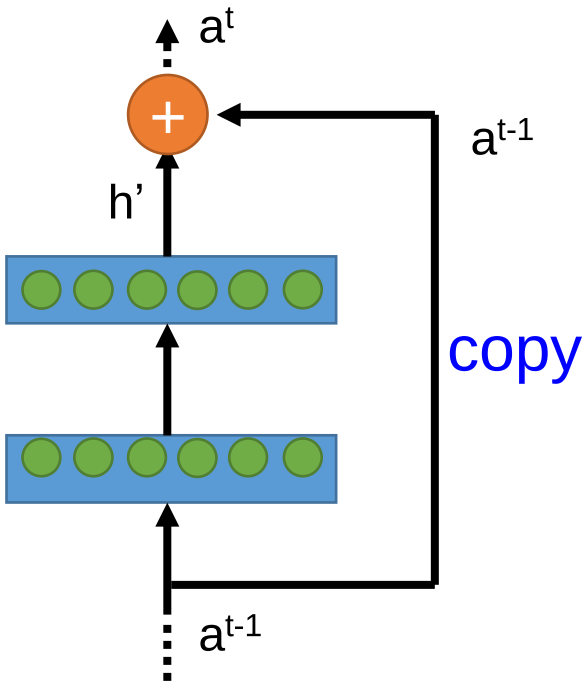
$z = \sigma(W' a^{t-1})$

$a^t = z \odot a^{t-1} + (1-z) \odot h$

- **Highway Network**

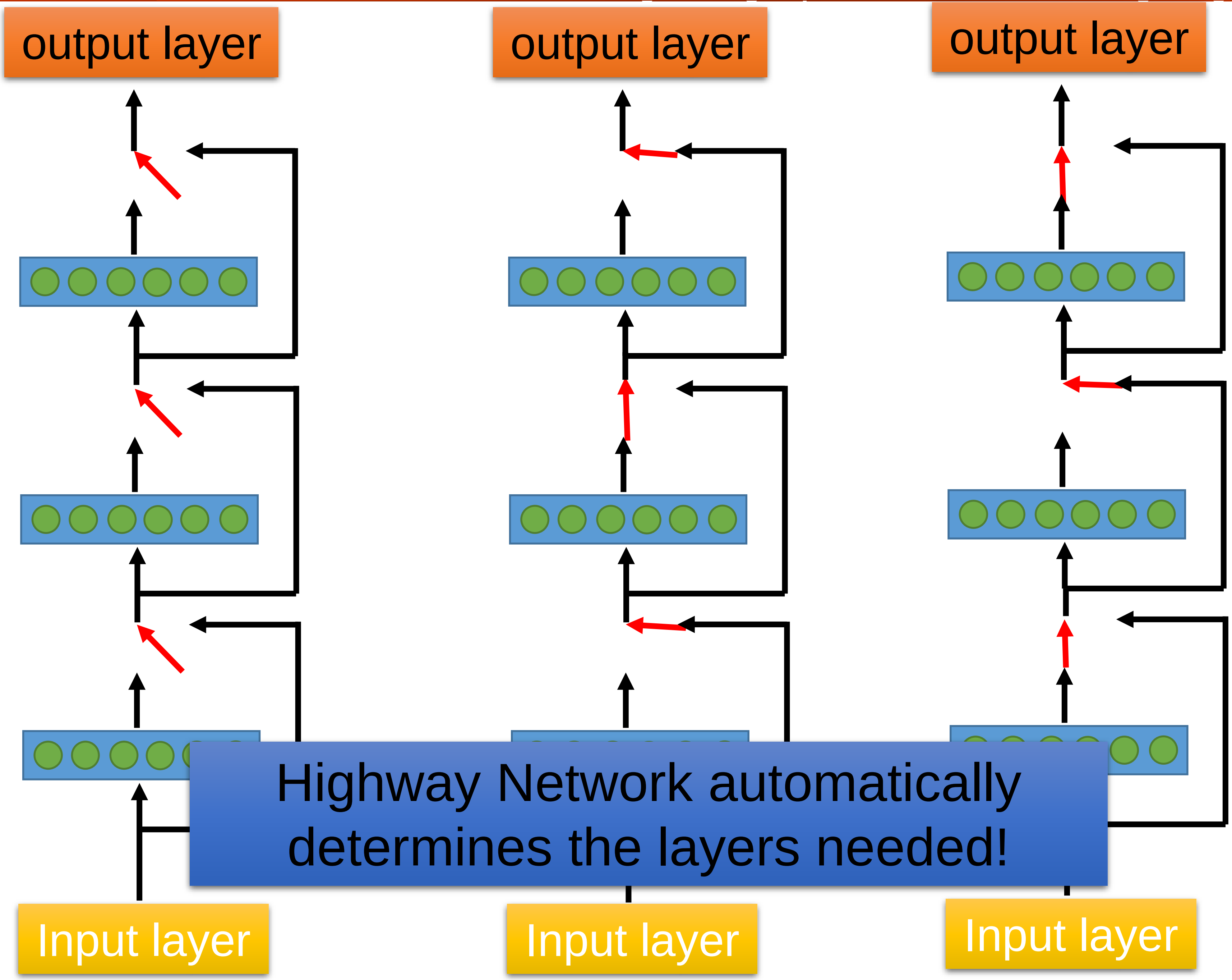


- **Residual Network**



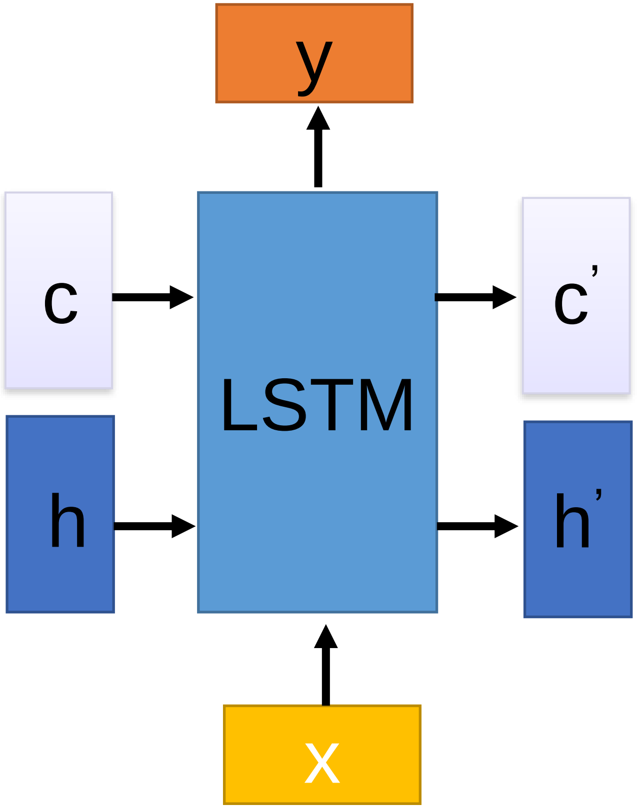
Training Very Deep Networks https://arxiv.org/pdf/1507.06228v2.pdf

Deep Residual Learning for Image Recognition http://arxiv.org/abs/1512.03385

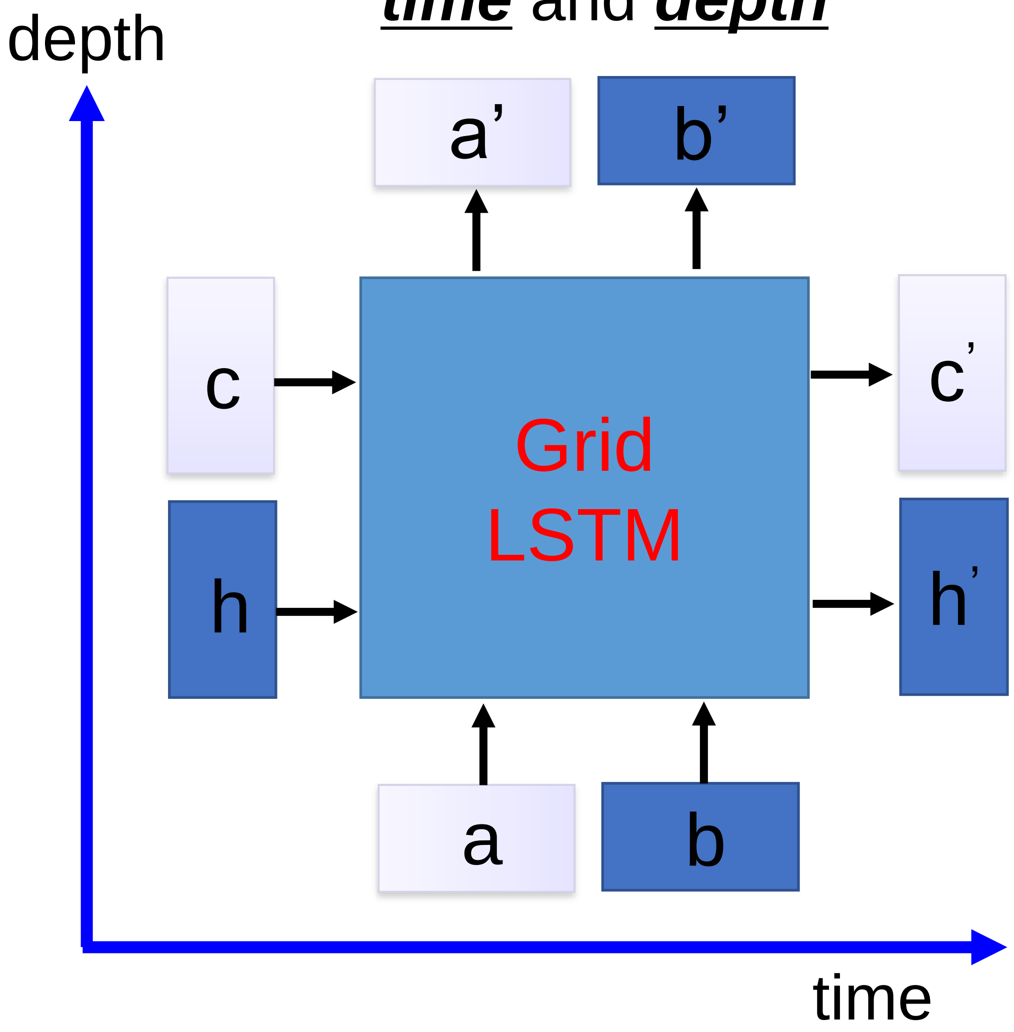


Highway Network automatically determines the layers needed!

Grid LSTM

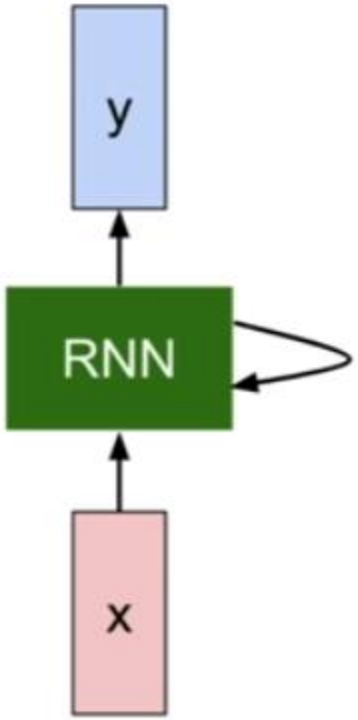


Memory for both
time and ***depth***



(Vanilla) Recurrent Neural Network

The state consists of a single *"hidden"* vector **h**:



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

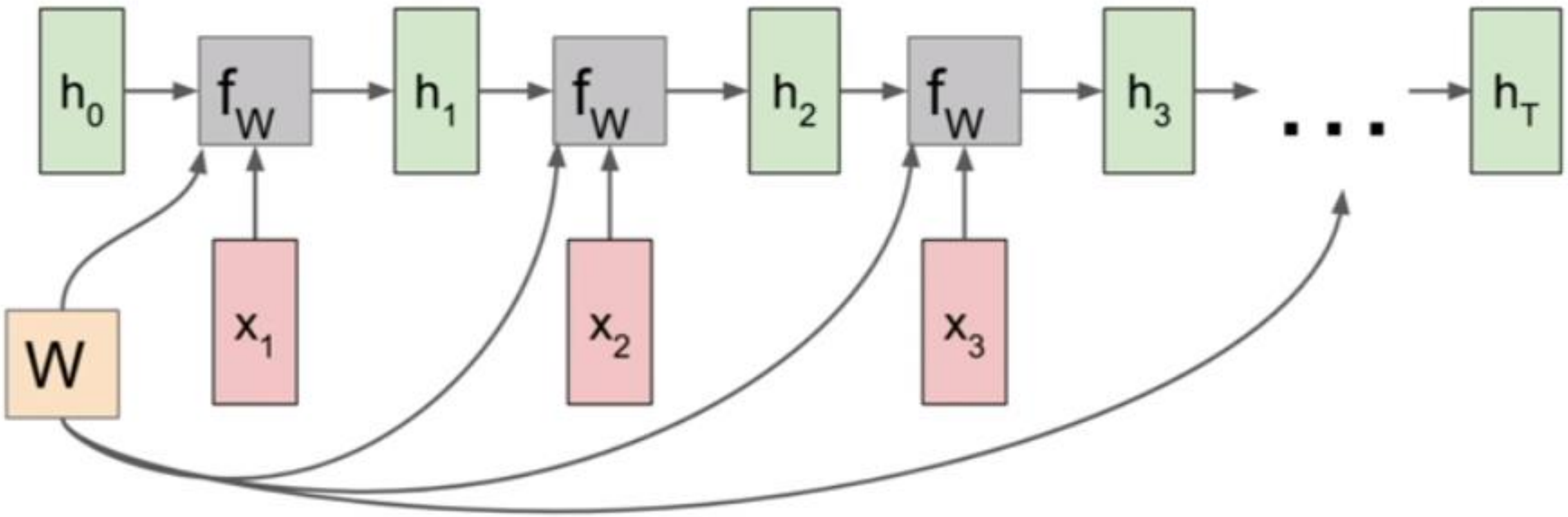
$$\downarrow$$

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

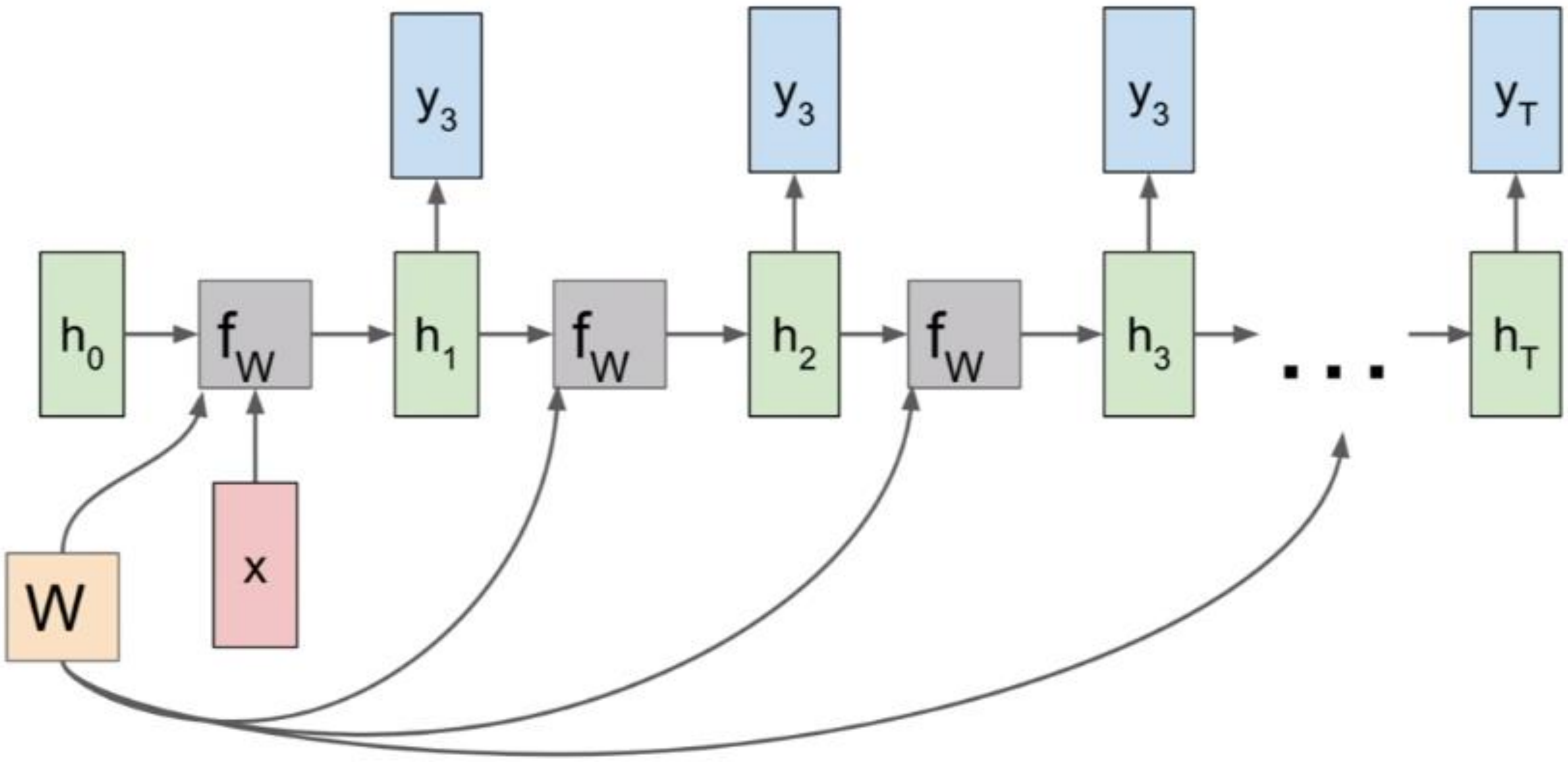
$$y_t = W_{hy}h_t$$

RNN: Computational Graph

Re-use the same weight matrix at every time-step

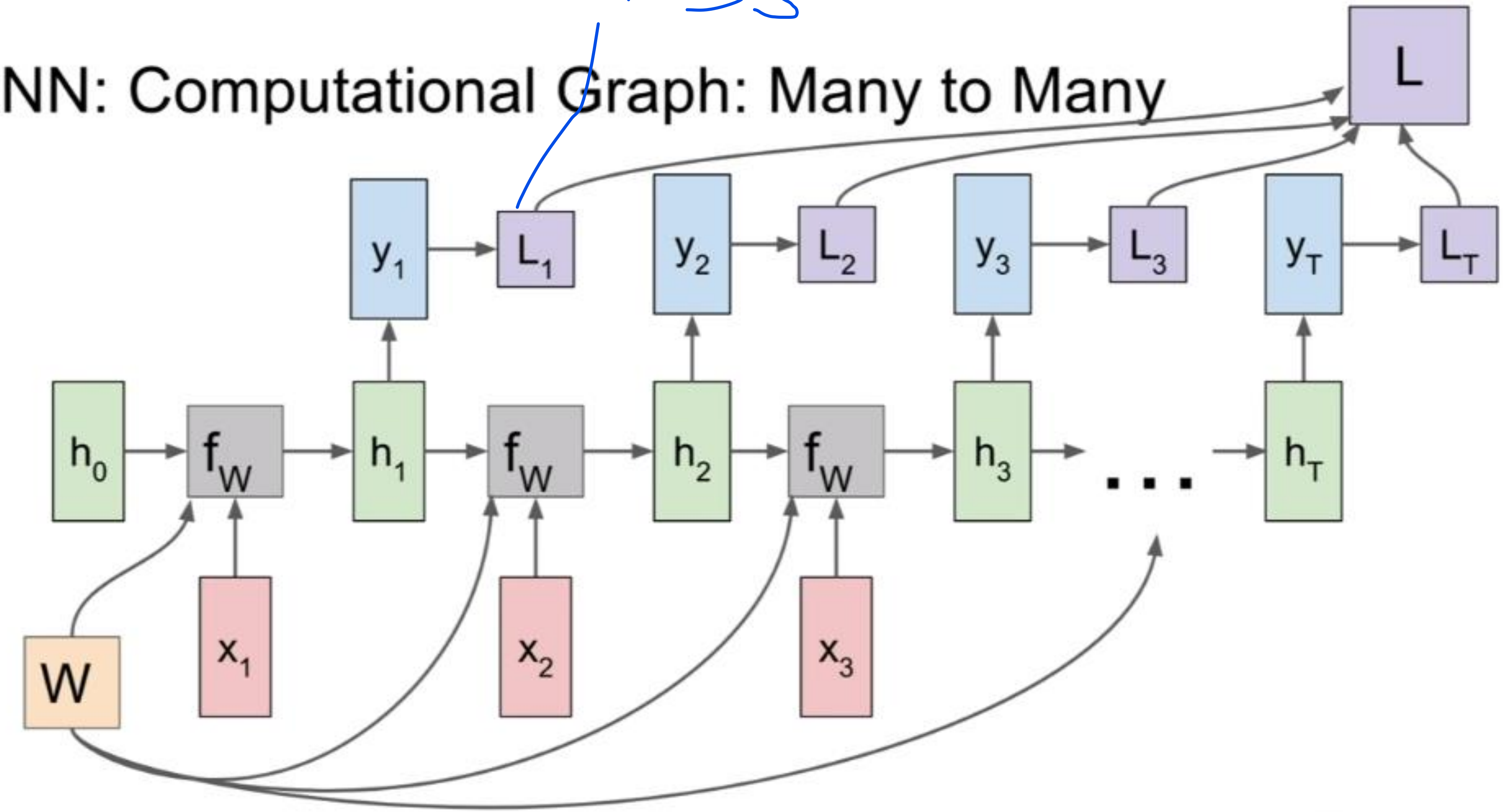


How do we train the network?

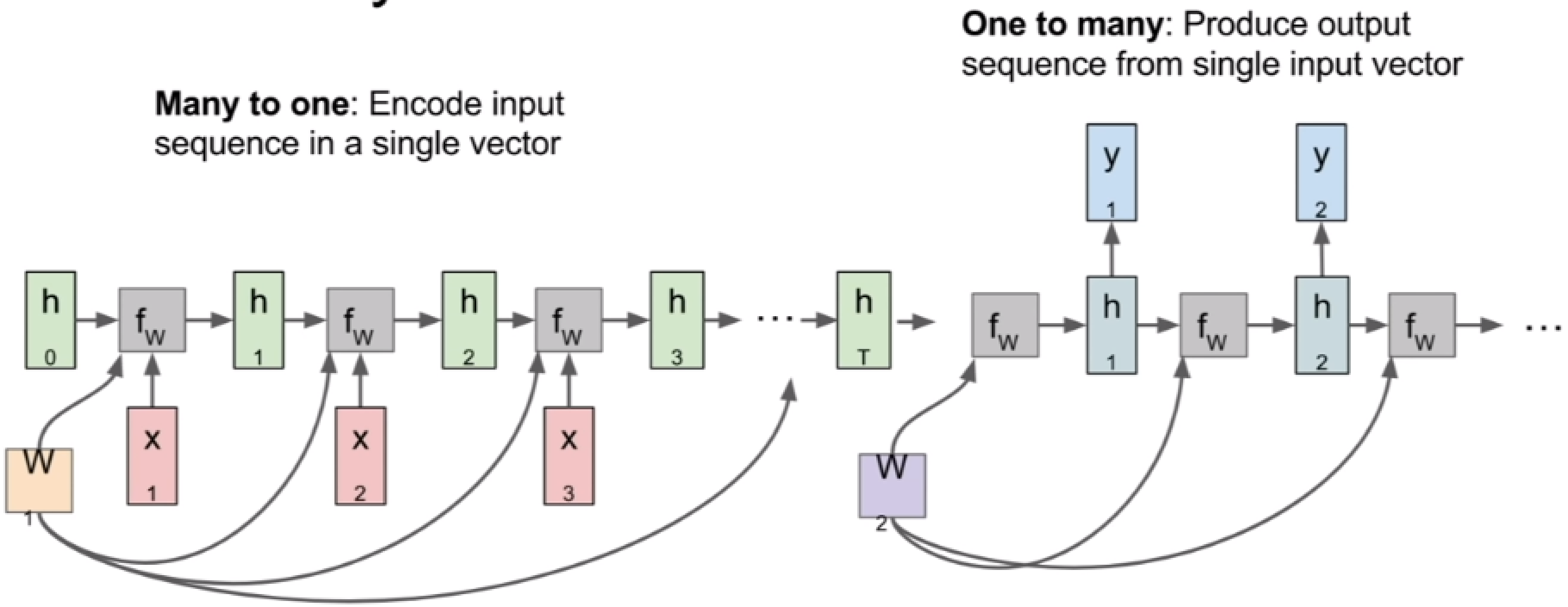


loss

RNN: Computational Graph: Many to Many

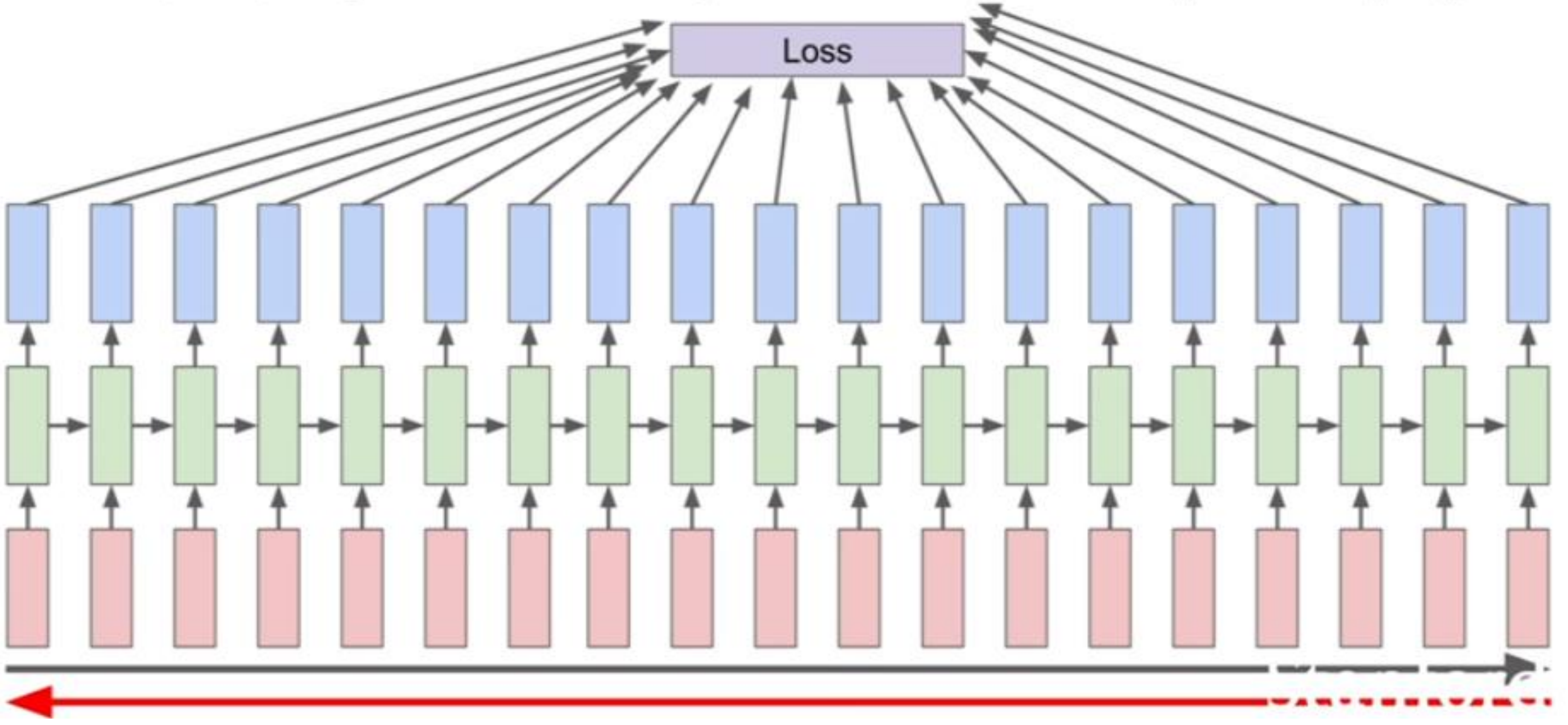


Sequence to Sequence: Many-to-one + one-to-many

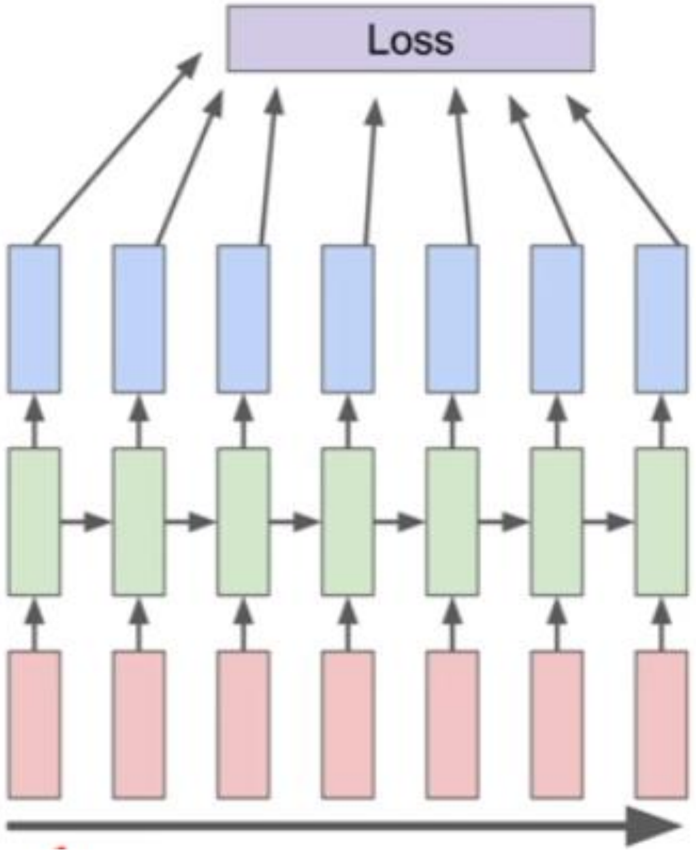


Backpropagation through time

Forward through entire sequence to compute loss, then backward through entire sequence to compute gradient



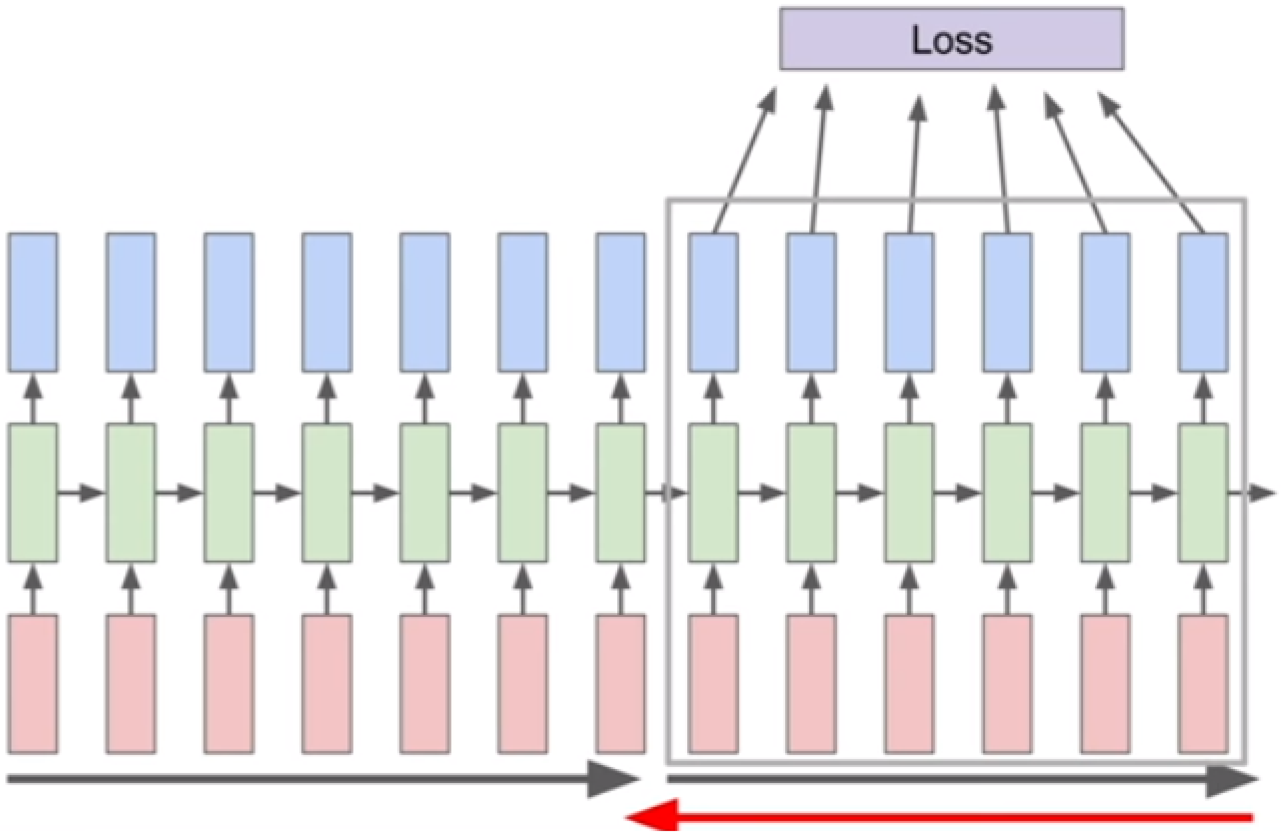
Truncated Backpropagation through time



Run forward and backward through chunks of the sequence instead of whole sequence

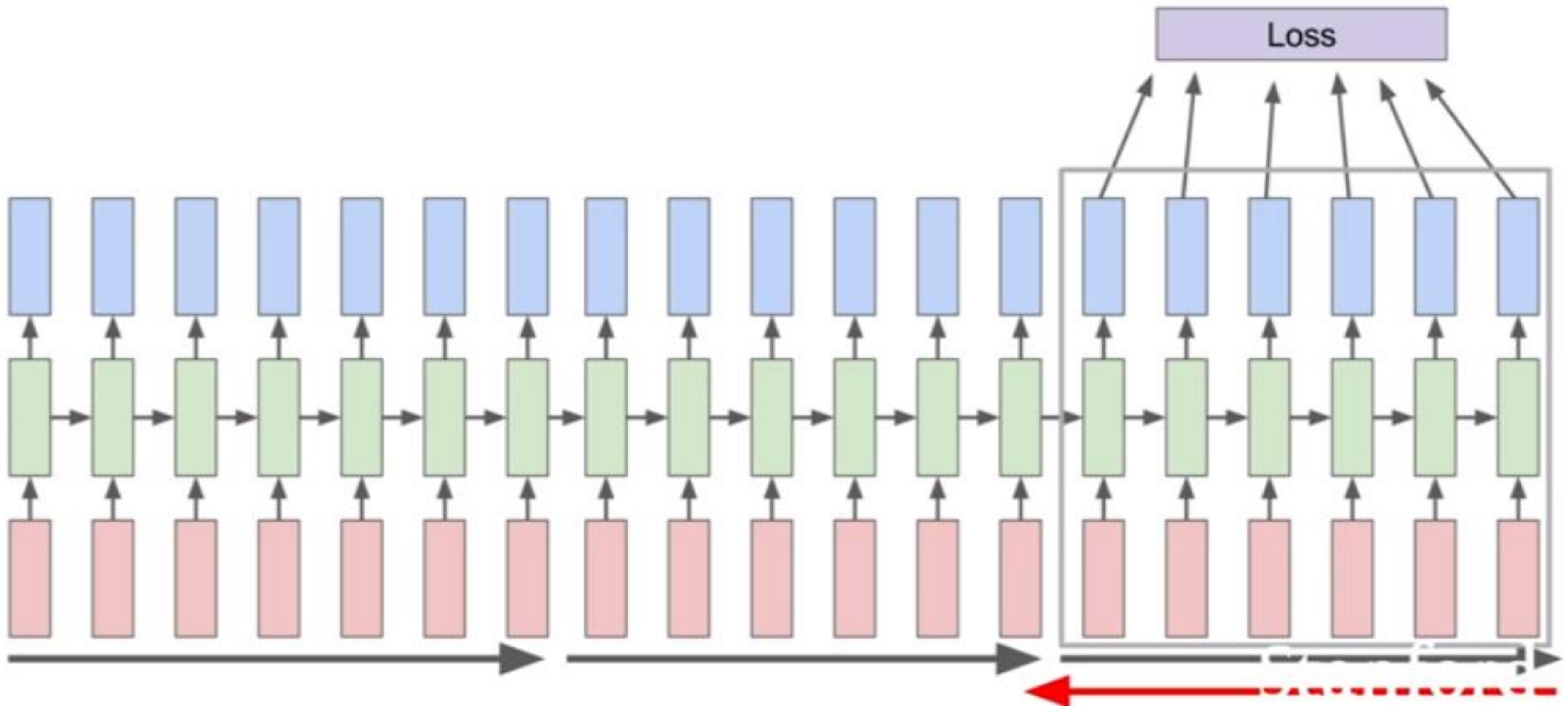
https://www.youtube.com/watch?v=6niqTuYFZLQ

Truncated Backpropagation through time

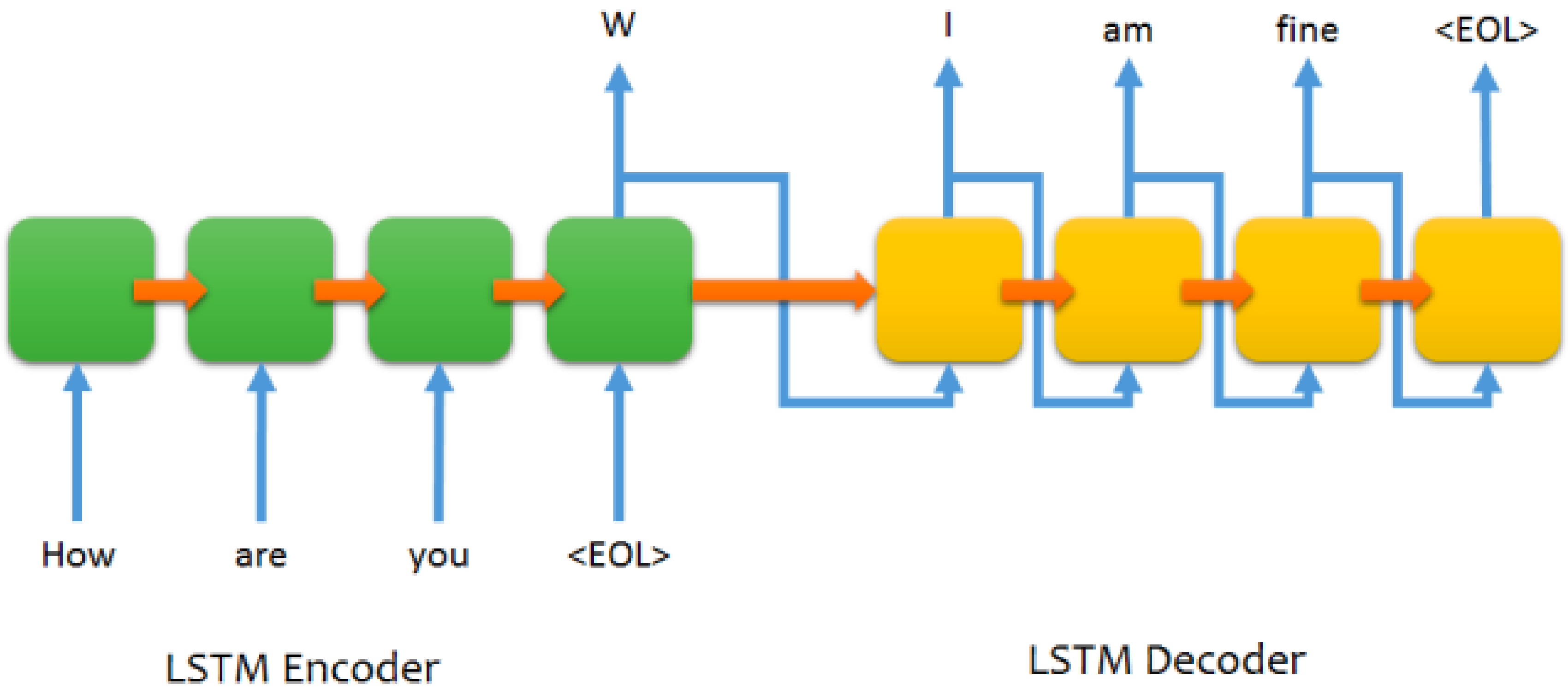


Carry hidden states forward in time forever, but only backpropagate for some smaller number of steps

Truncated Backpropagation through time

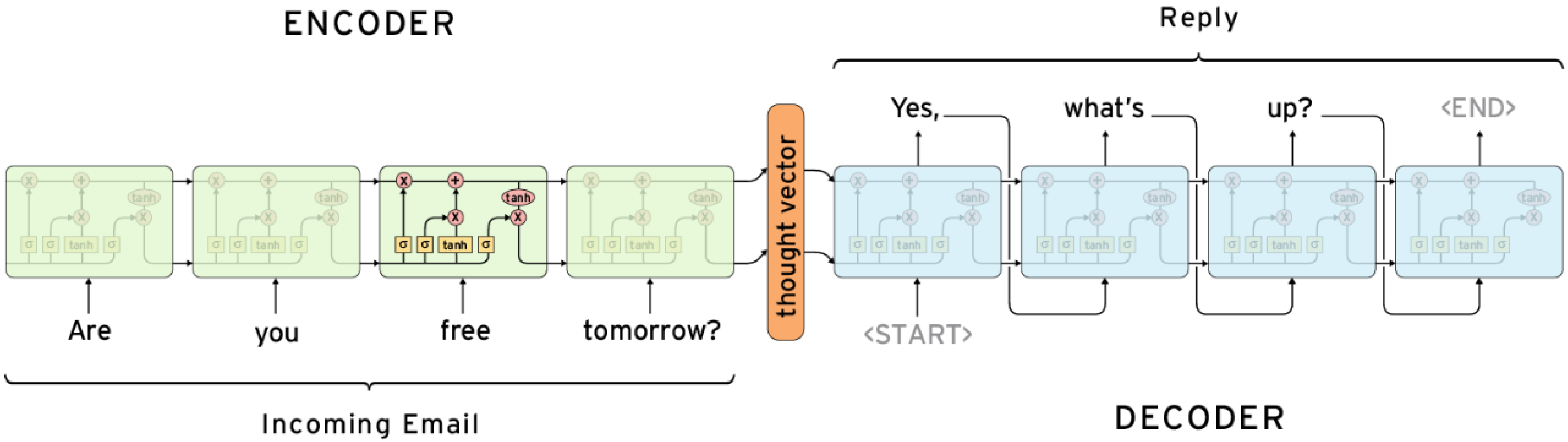


Sequence to sequence chat model



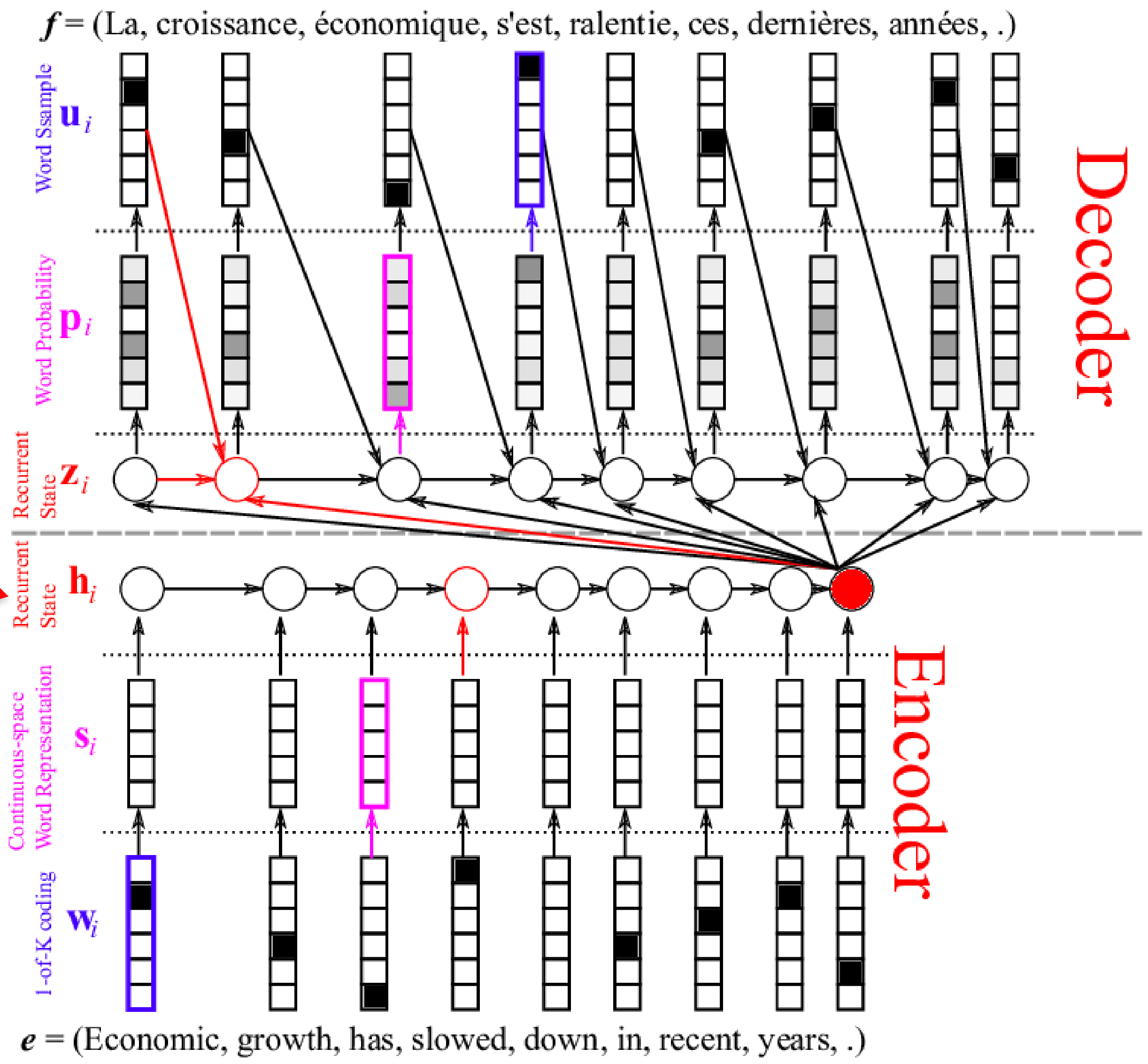
Applications of LSTM / RNN

Seq2Seq: 解決output 與input 須等長之困擾
using encoder and decoder

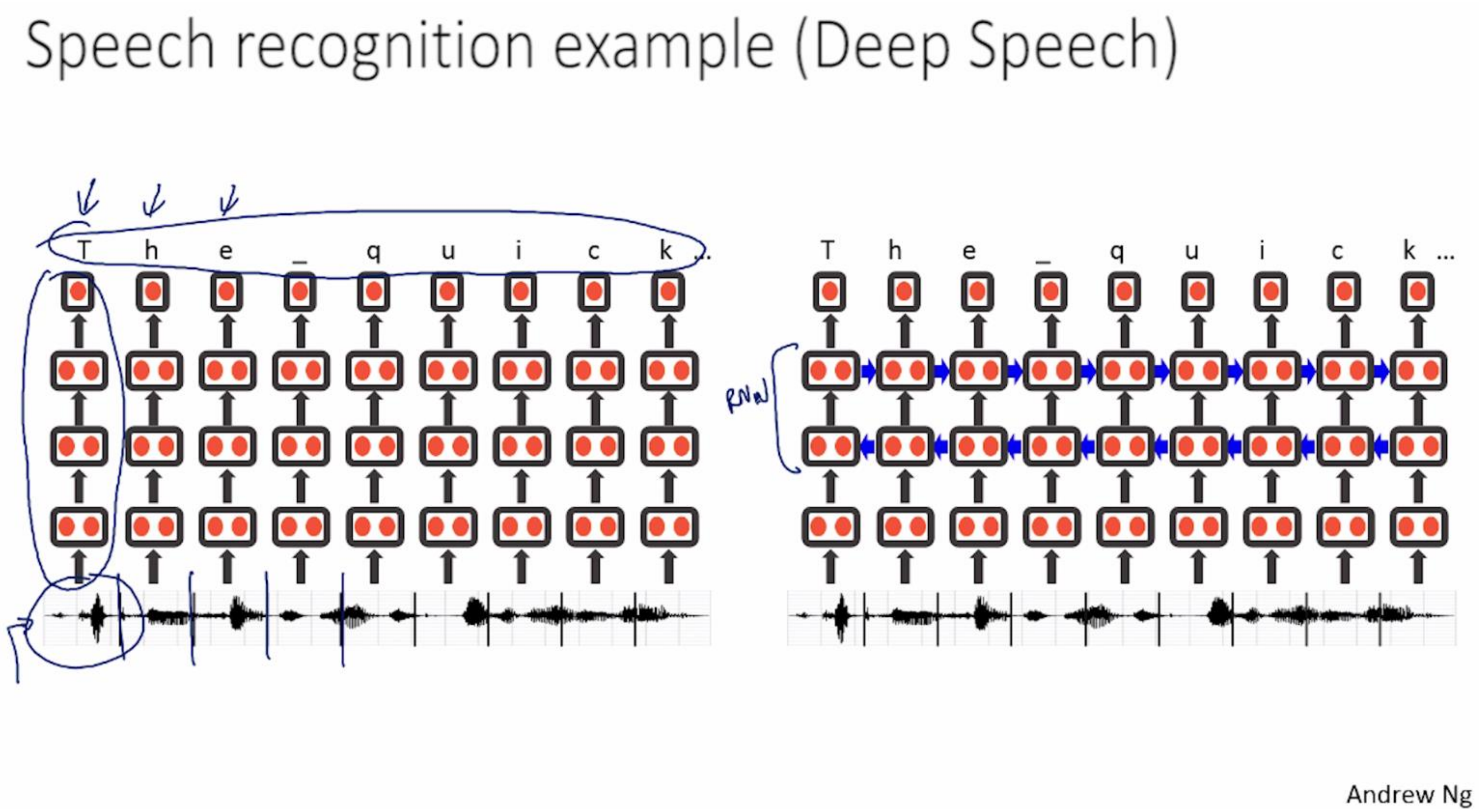


Neural machine translation

LSTM



Baidu’s speech recognition using RNN



Attention

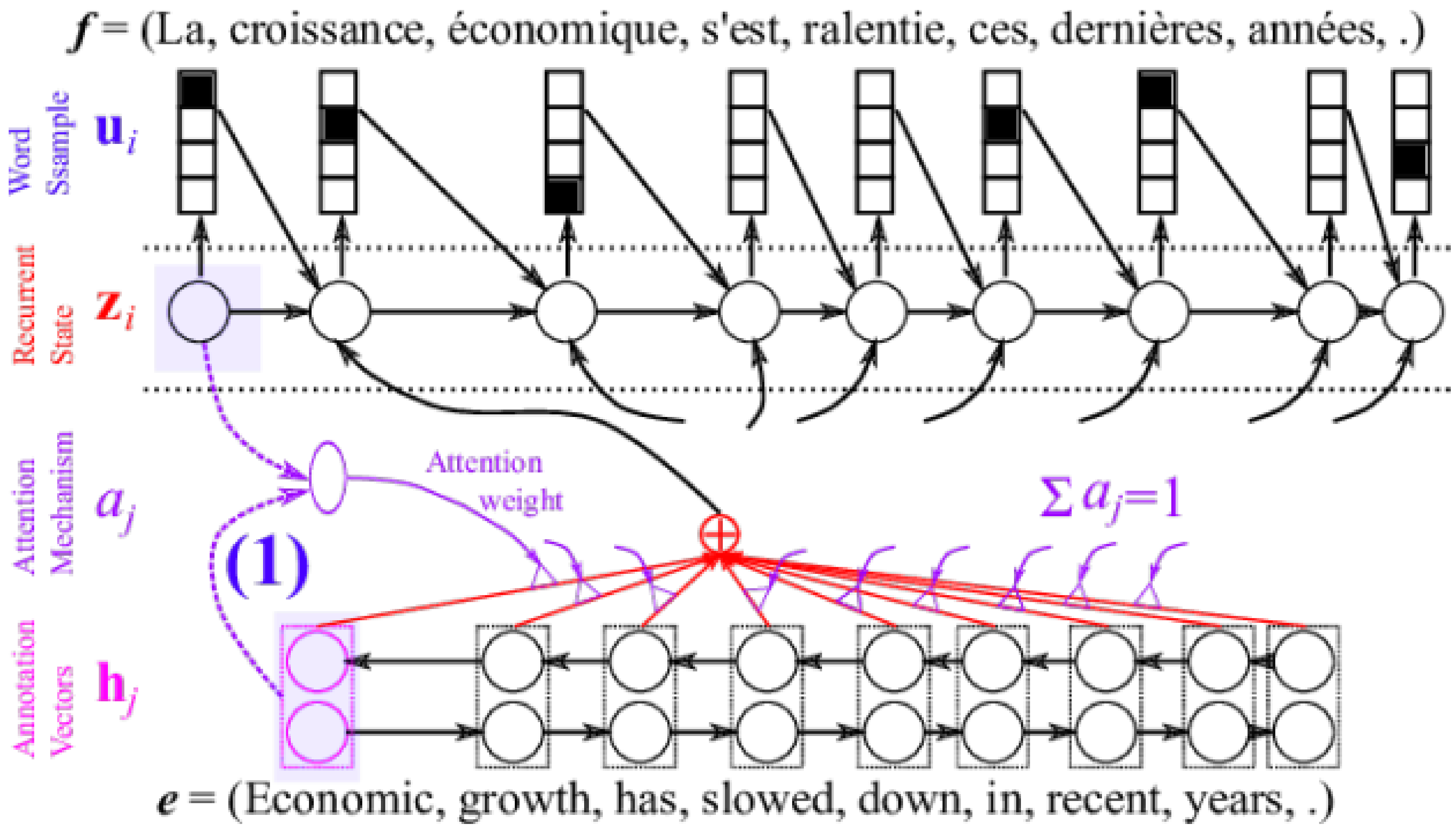
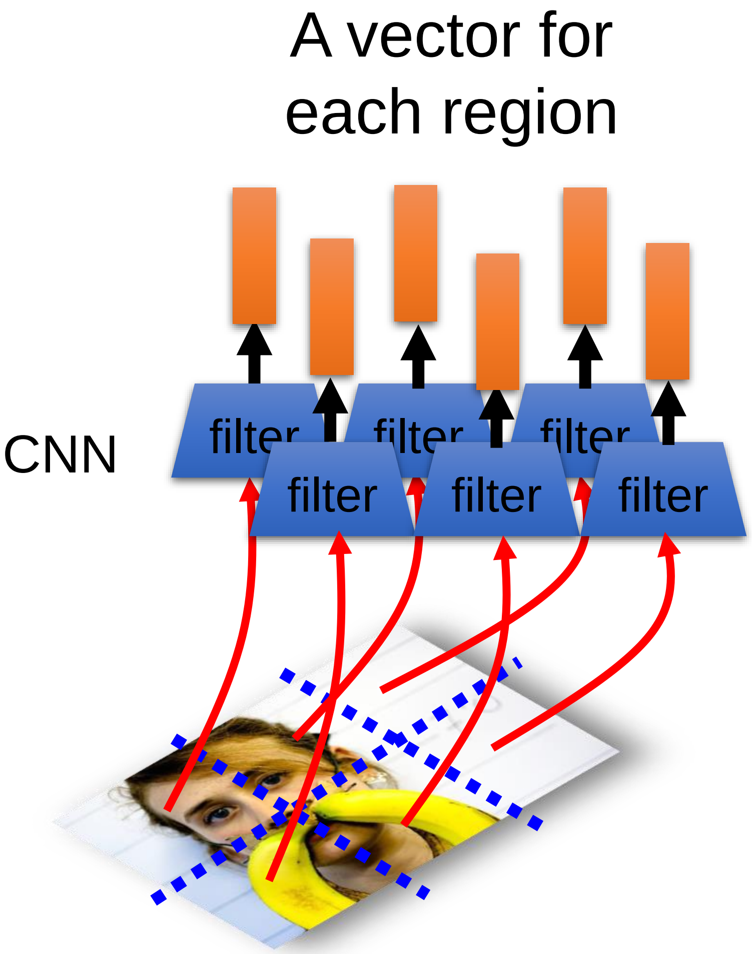

f = (La, croissance, économique, s'est, ralentie, ces, dernières, années, .)
Word Ssample
$\mathbf{u}_i$
Recurrent State
$\mathbf{z}_i$
Attention Mechanism
a_j
Attention weight
(1)
$\Sigma a_j=1$
Annotation Vectors
$\mathbf{h}_j$
e = (Economic, growth, has, slowed, down, in, recent, years, .)

Image caption generation using attention (From CY Lee lecture)



z^0 is initial parameter, it is also learned

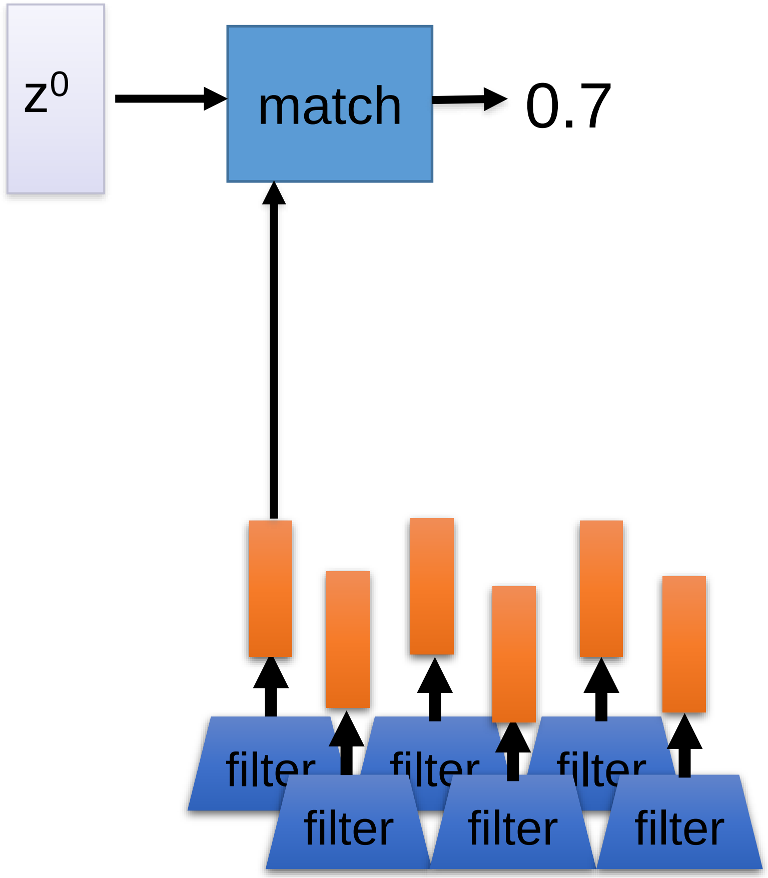


Image Caption Generation

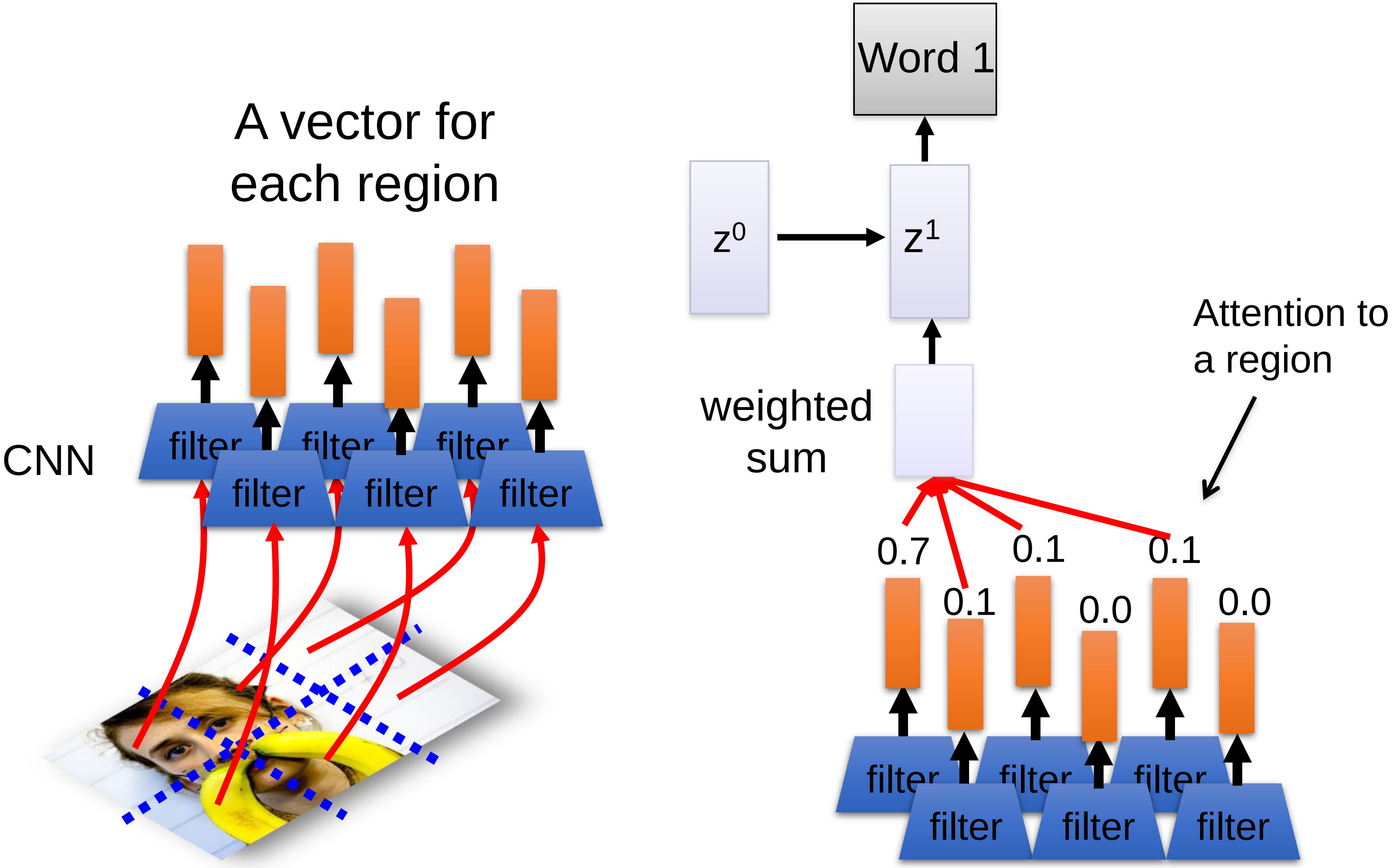
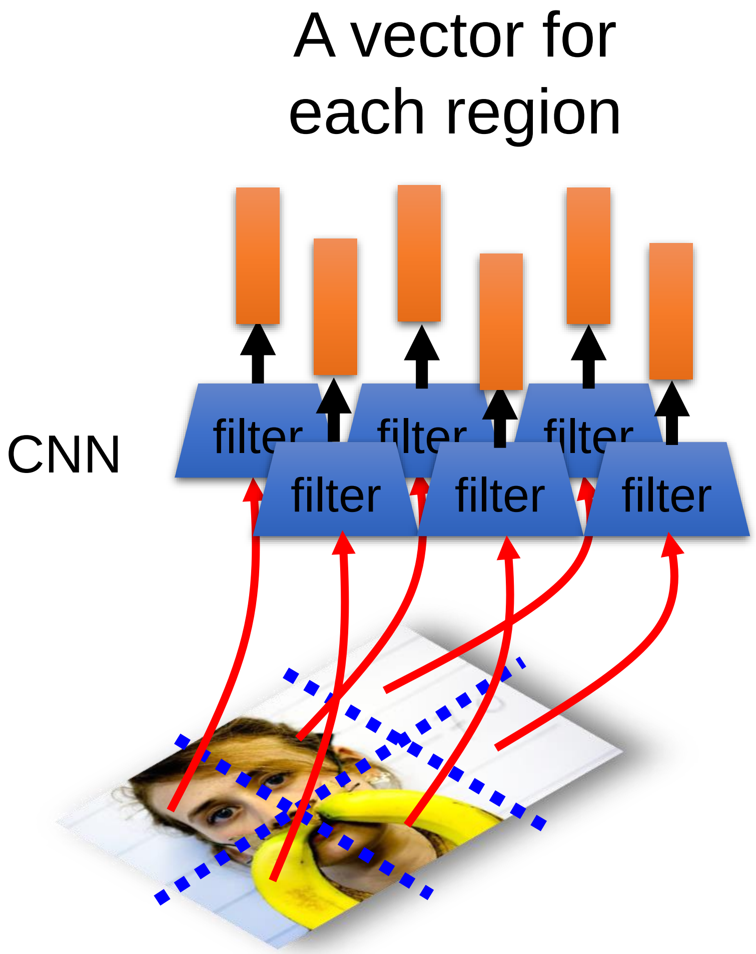


Image Caption Generation



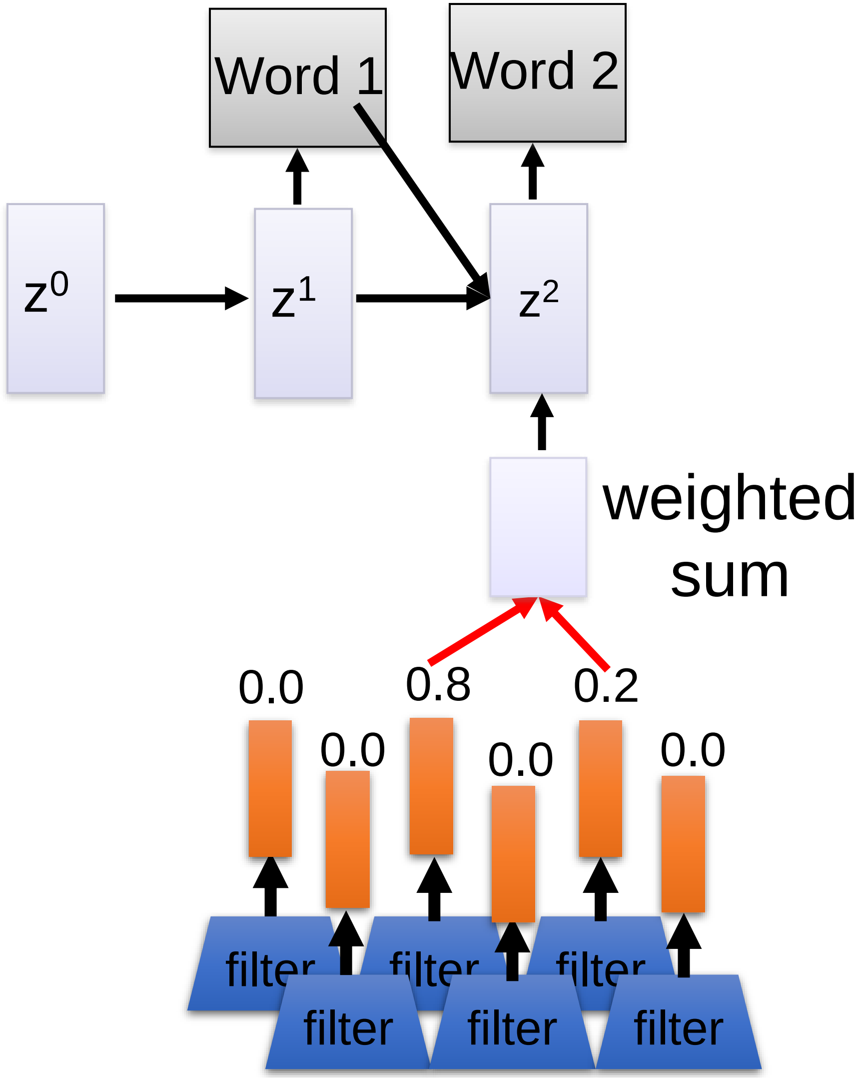


Image Caption Generation

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, Yoshua Bengio, “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention”, ICML, 2015