Deep Learning (DL)

Ups and downs of Deep Learning

- 1958: Perceptron
- 1980s: Multi-layer perceptron (MLP)
- 1986: Backpropagation (BP)
- 1989: 1 hidden layer is “good enough”, why deep?
- 2006: Restricted Boltzmann Machine (RBM) initialization
- 2009: GPU
- 2011: Start to be popular in speech recognition
- 2012: win ILSVRC image competition
- 2015: Image recognition surpassing human-level performance
- 2016: Alpha GO
- 2016: Speech recognition system as good as humans
- 2019: Pretrained language models (PLMs) for NLP tasks
- 2023: Large language models (LLMs)
- ……

Three Steps for Deep Learning

Step 1. A neural network is a function composed of simple functions (neurons)

- Usually we design the network structure, and let machine find parameters from data

Step 2. Cost function evaluates how good a set of parameters is

- We design the cost function based on the task

Step 3. Find the best function (e.g., gradient descent)

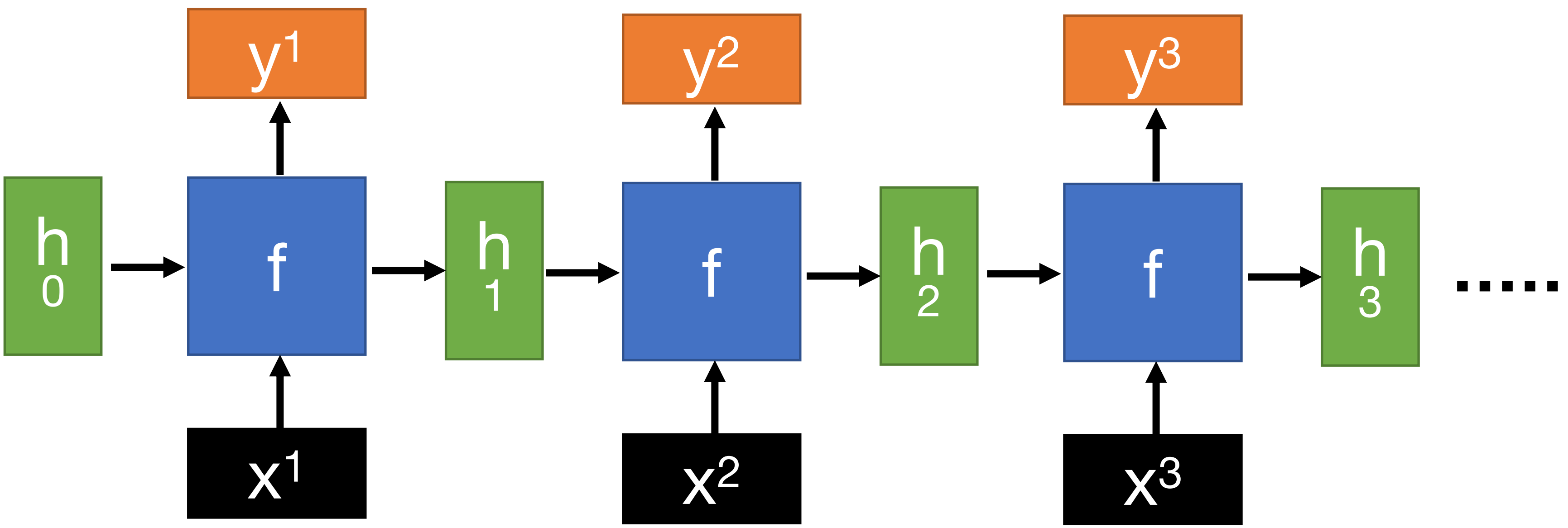
Basic Structure:
Recurrent Structure

Simplify the network
by using the same function again and again

Recurrent Neural Network

- Given function f: $h', y = f(h, x)$

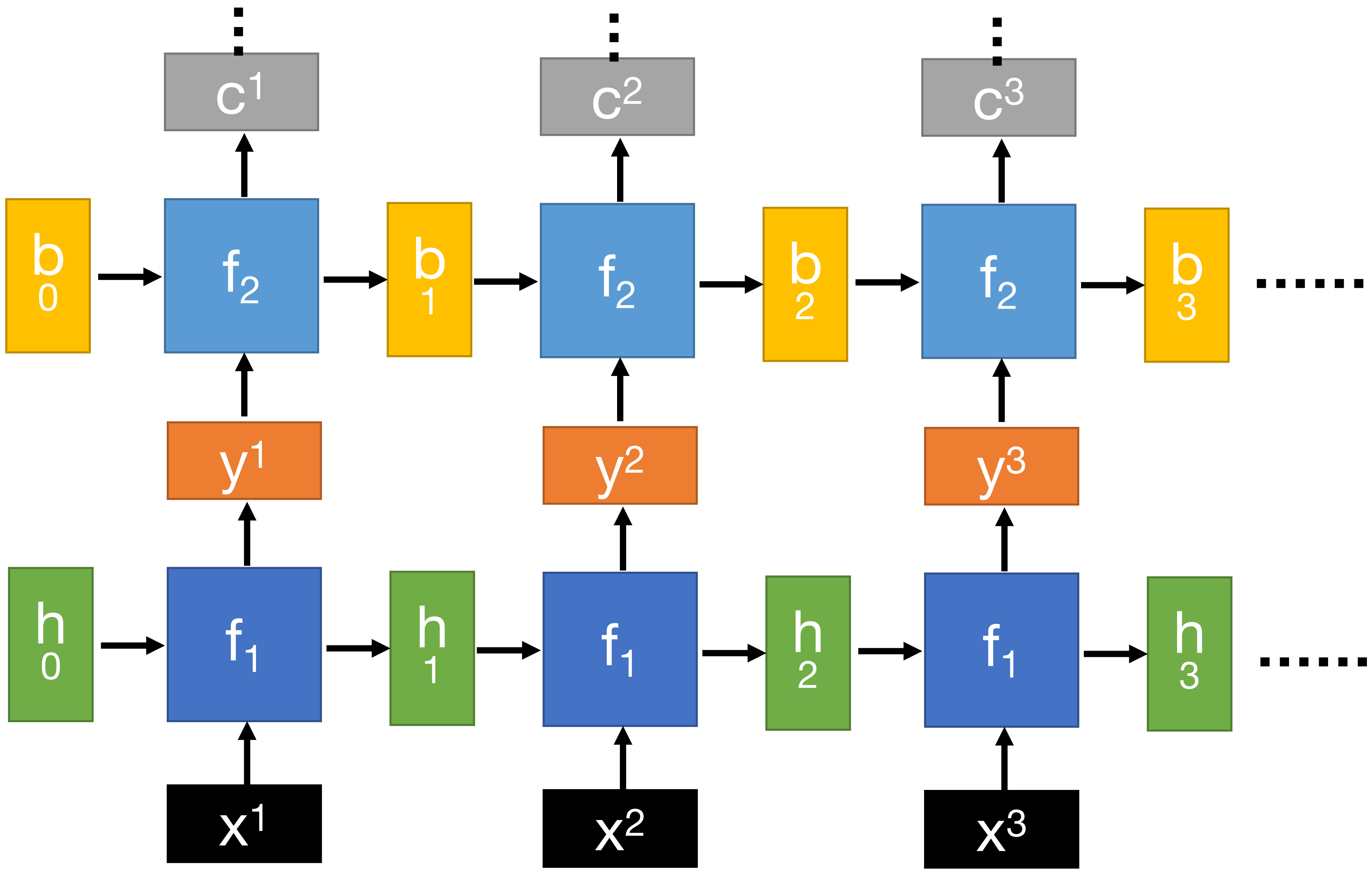
h and h' are vectors with the same dimension



No matter how long the input sequence is, we only need one function f

Deep RNN

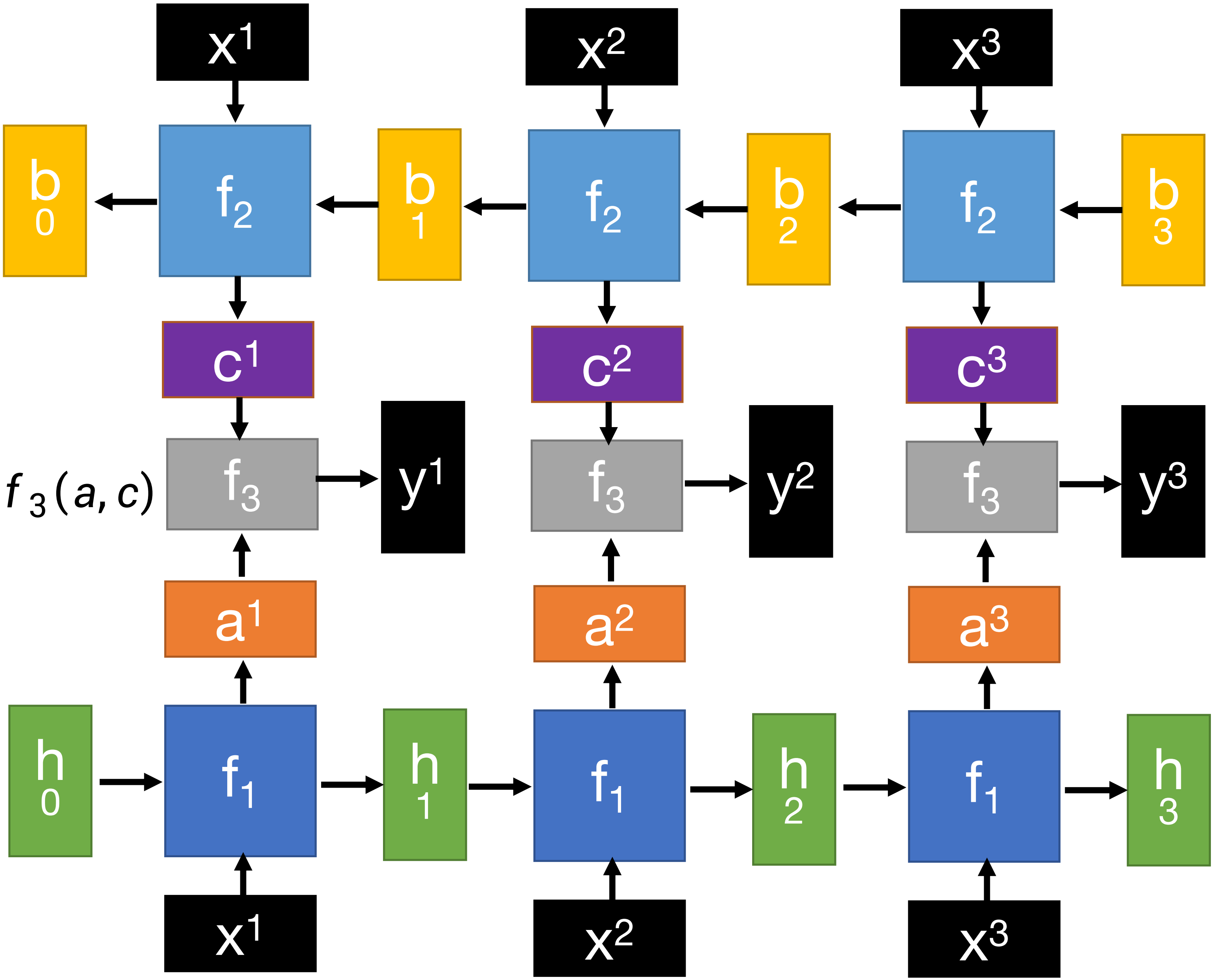
$$h', y = f_1(h, x) \quad b', c = f_2(b, y) \cdots$$



Bidirectional RNN

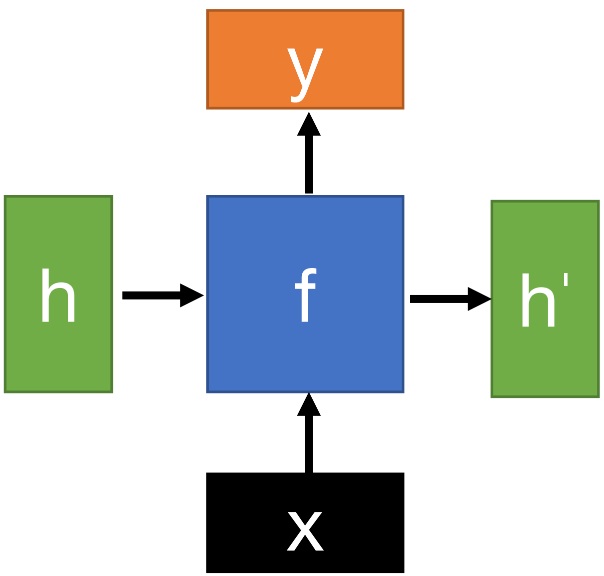
$h', a = f_1(h, x) \quad b', c = f_2(b, x)$

$y = f_3(a, c)$



Naïve RNN

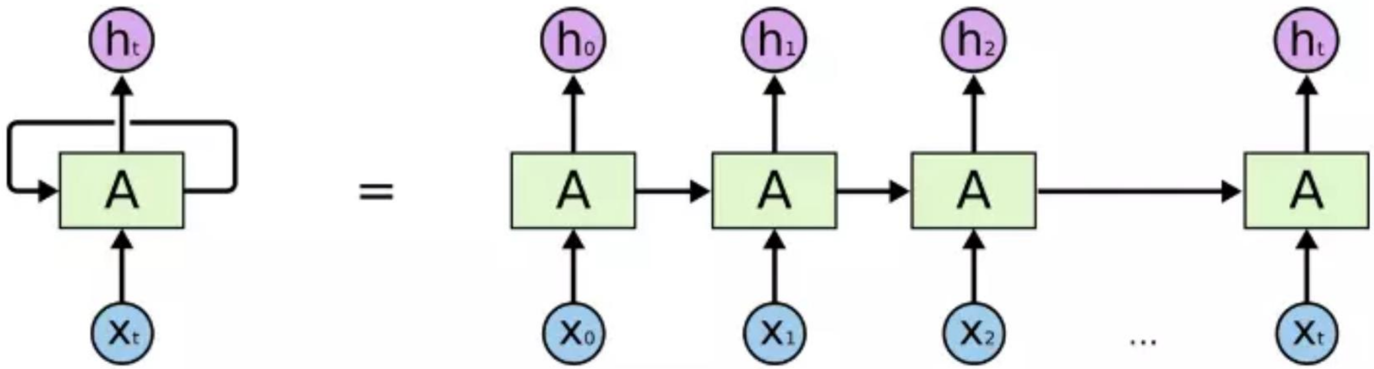
- Given function f: $h', y = f(h, x)$

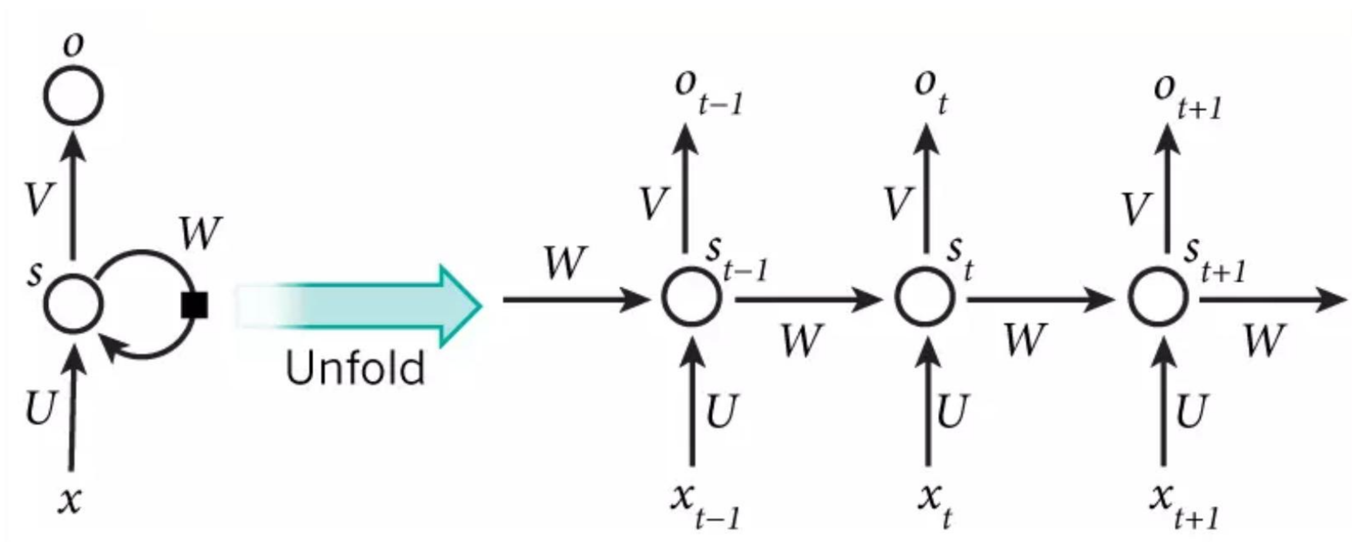


$$h' = \sigma(W^h h + W^i x)$$

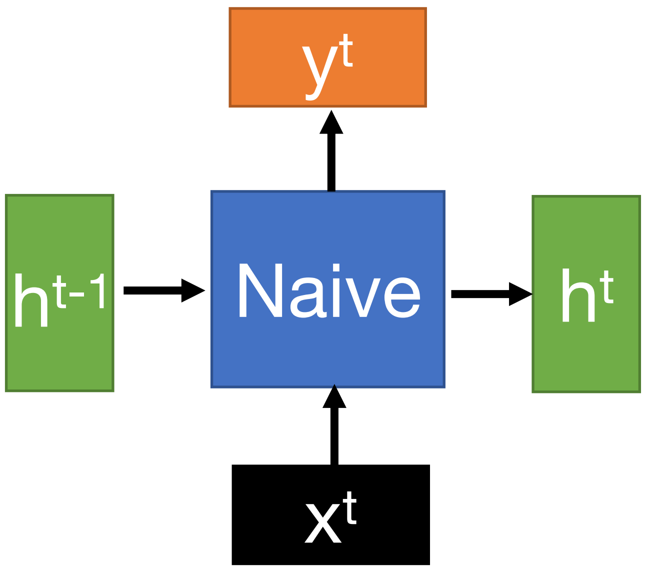
$$y = \sigma(W^o h')$$

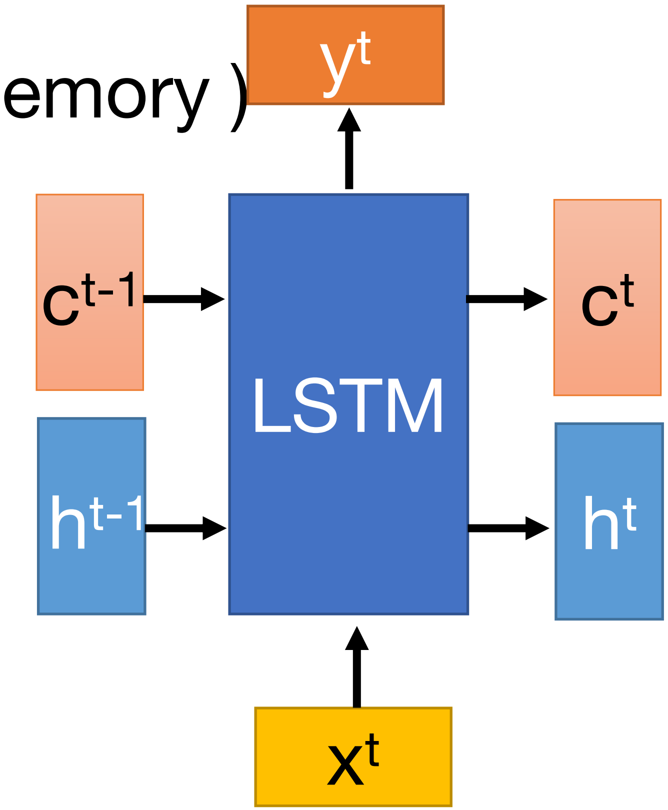
Ignore bias here


ht
A
xt
=
h0
h1
h2
ht
A
A
A
A
x0
x1
x2
...
xt


o
V
W
s
U
x
Unfold
W
o t−1
V
s t−1
W
U
x t−1
o t
V
s t
W
U
x t
o t+1
V
s t+1
W
U
x t+1

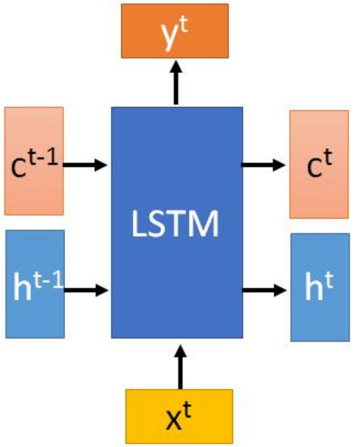
LSTM (long short-term memory)

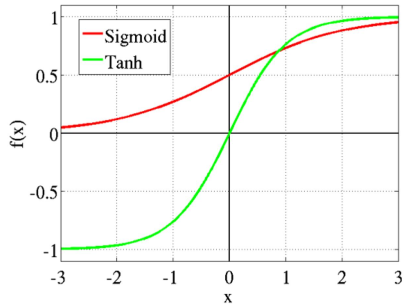




c changes slowly ➡ c^t is c^{t-1} added by something

h changes fast ➡ h^t and h^{t-1} can be very different





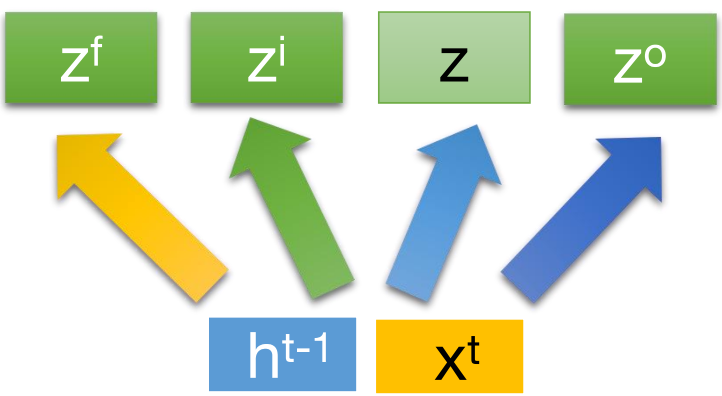
$$z = tanh\left(W \begin{bmatrix} x^t \\ h^{t-1} \end{bmatrix} \right)$$

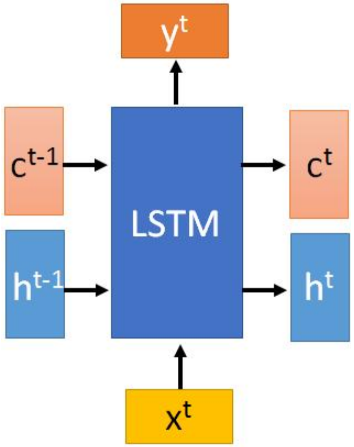
c^{t-1}

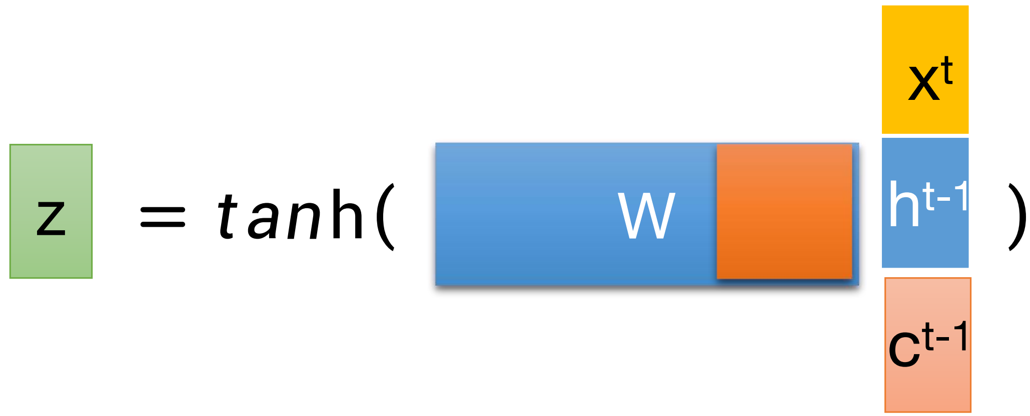
$$z^i = \sigma\left(W^i \begin{bmatrix} x^t \\ h^{t-1} \end{bmatrix} \right)$$

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

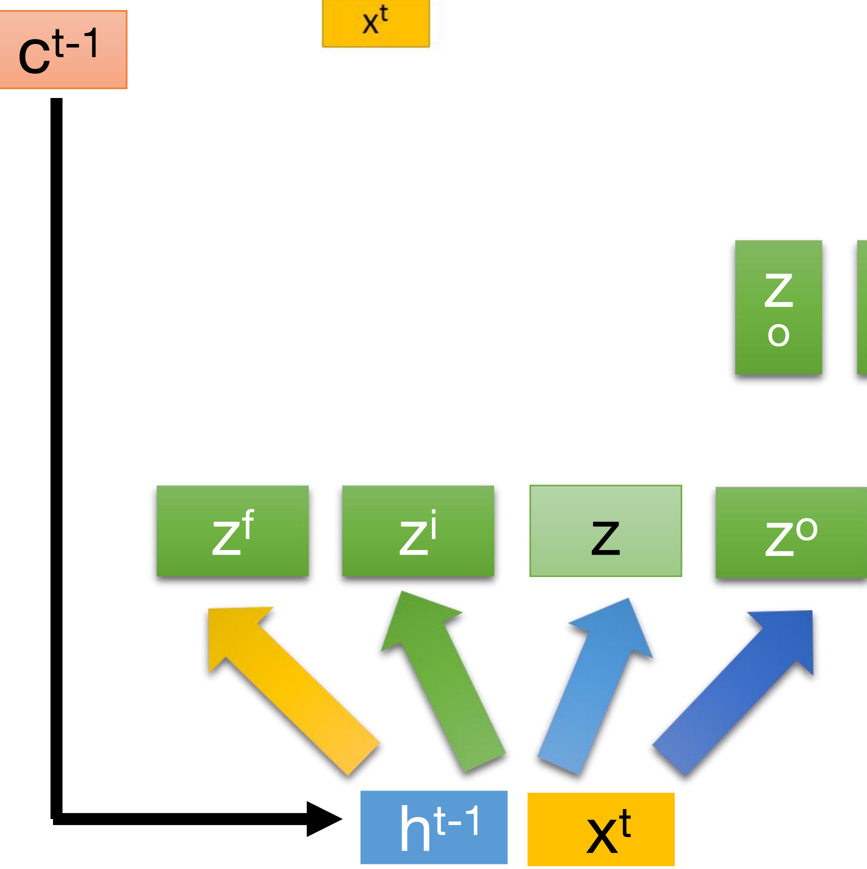
$$z^o = \sigma\left(W^o \begin{bmatrix} x^t \\ h^{t-1} \end{bmatrix} \right)$$

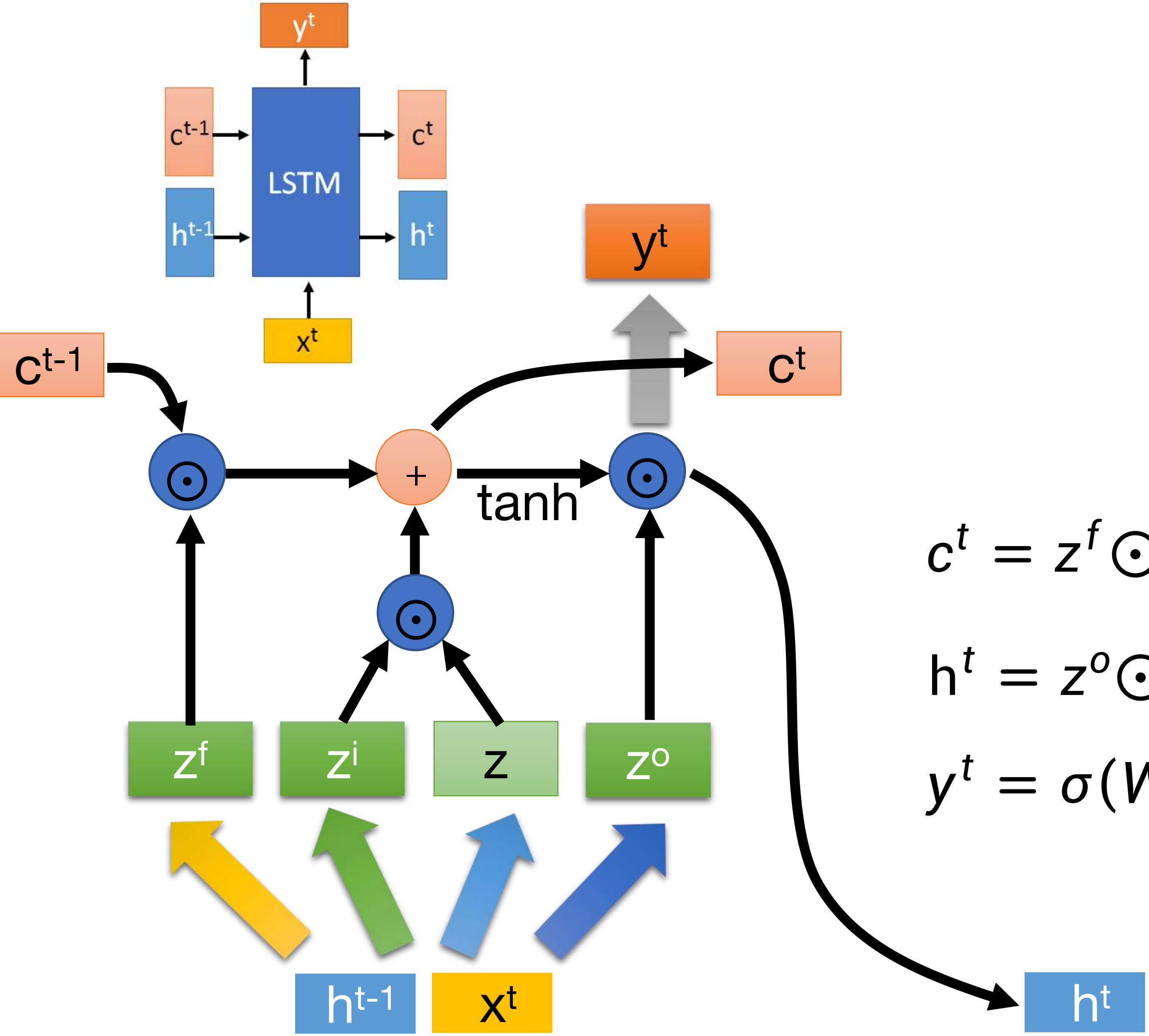






z^o z^f z^i obtained by the same way



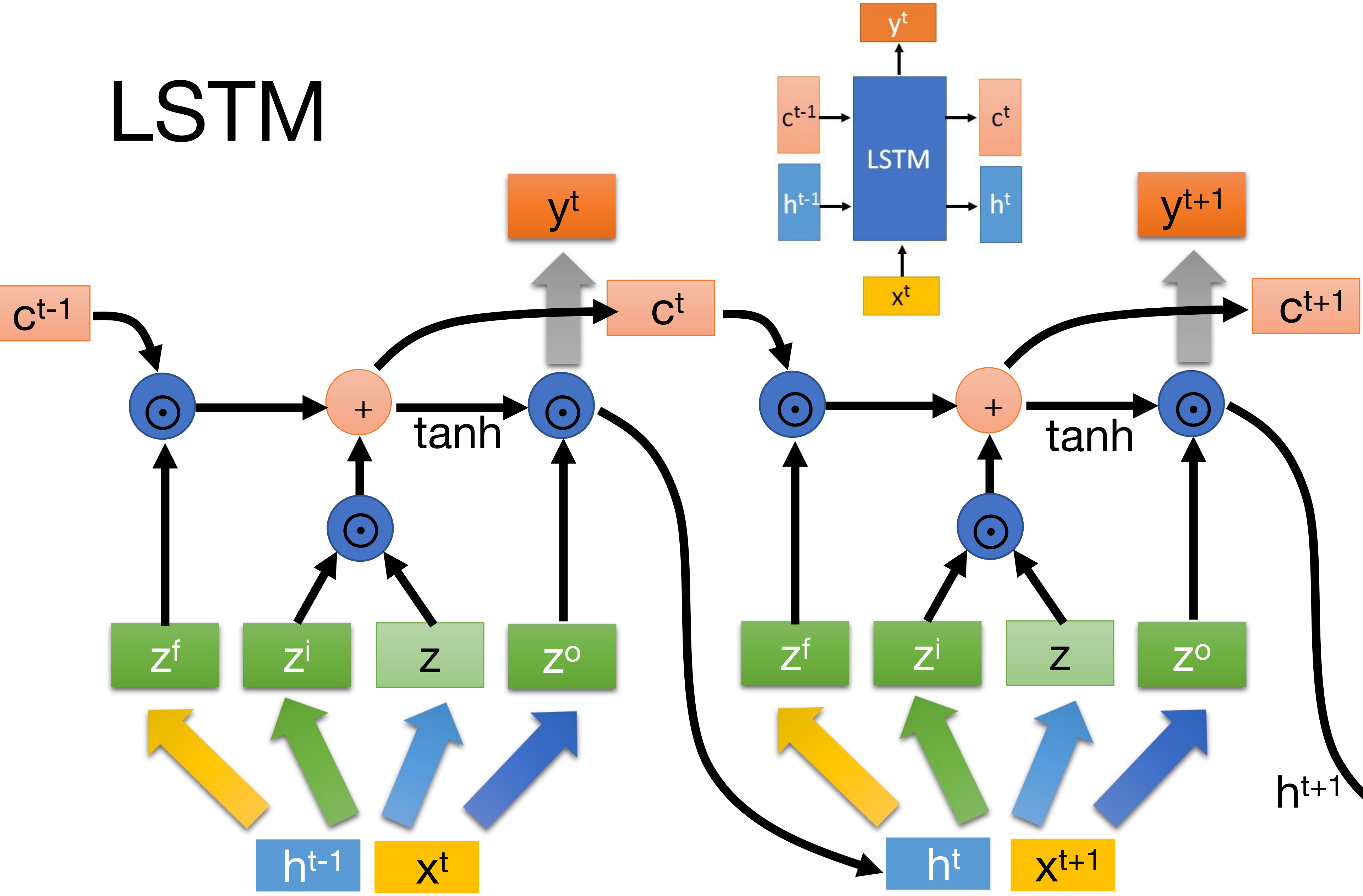


$$c^t = z^f \odot c^{t-1} + z^i \odot z$$

$$h^t = z^o \odot tanh(c^t)$$

$$y^t = \sigma(W^{'} h^t)$$

LSTM



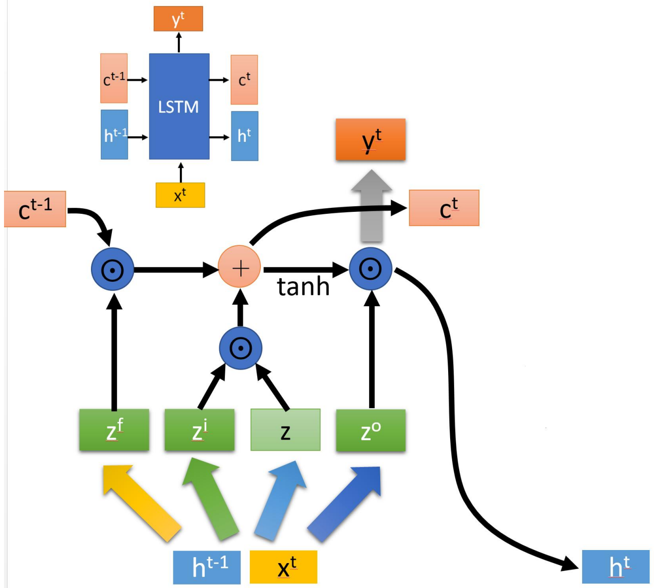
yt
ct-1
LSTM
ct
ht-1
ht
xt
yt+1
ct+1
ct
+
tanh
zf
zi
z
zo
ht-1
xt
ht
xt+1
ht+1

```python
def LSTMCELL(prev_ct, prev_ht, input):
    combine = prev_ht + input
    ft = forget_layer(combine)
    candidate = candidate_layer(combine)
    it = input_layer(combine)
    Ct = prev_ct * ft + candidate * it
    ot = output_layer(combine)
    ht = ot * tanh(Ct)
    return ht, Ct



ct = [0, 0, 0]
ht = [0, 0, 0]

for input in inputs:
    ct, ht = LSTMCELL(ct, ht, input)
```



$$c^t = z^f \odot c^{t-1} + z^i \odot z$$

$$h^t = z^o \odot tanh(c^t)$$

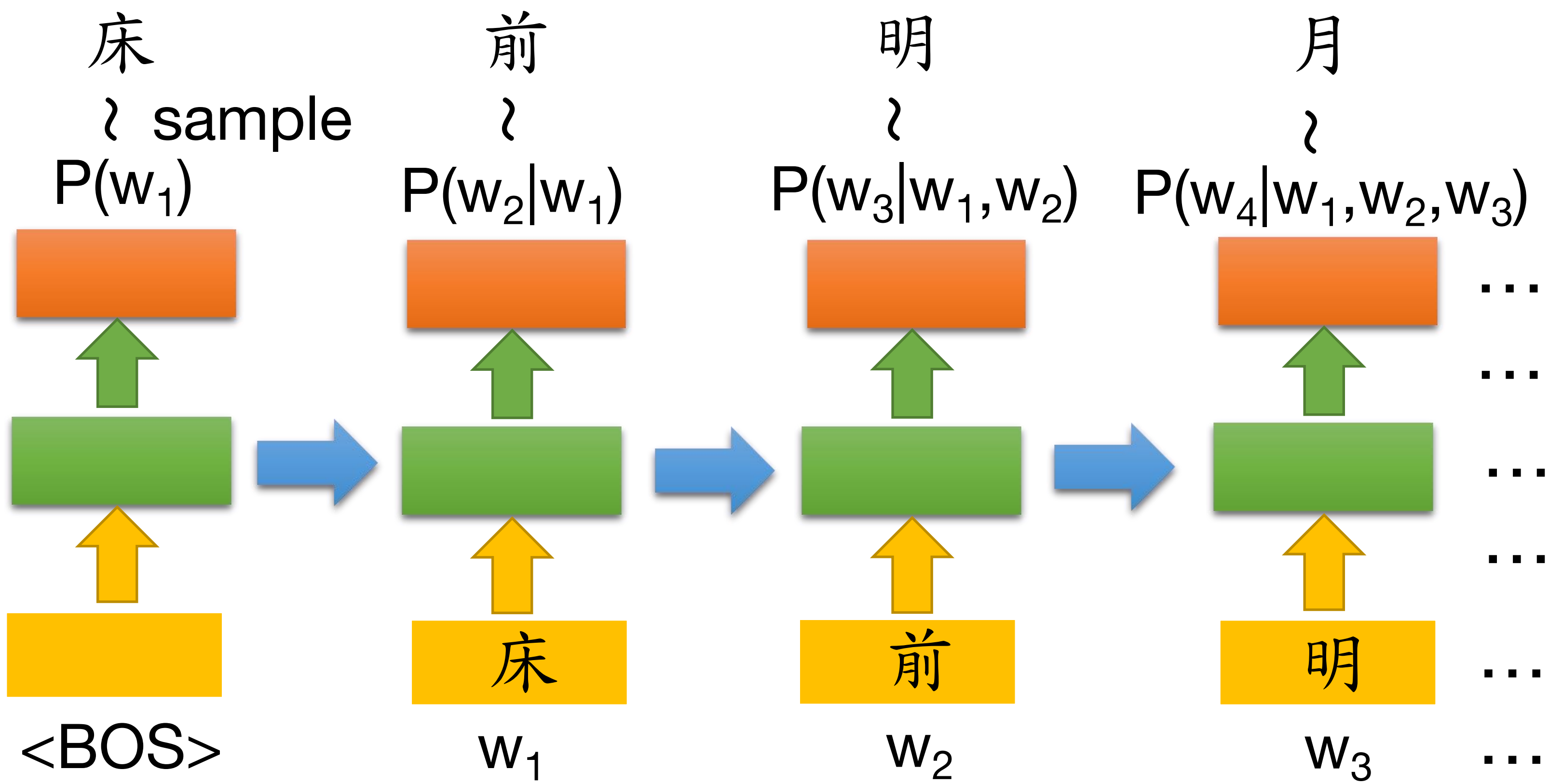
$$y^t = \sigma(W' h^t)$$

Conditional Generation by RNN & Attention

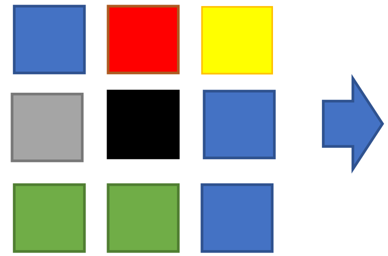
Generating a structured object component-by-component

Generation

- Sentences are composed of characters/words
    - Generating a character/word at each time by RNN



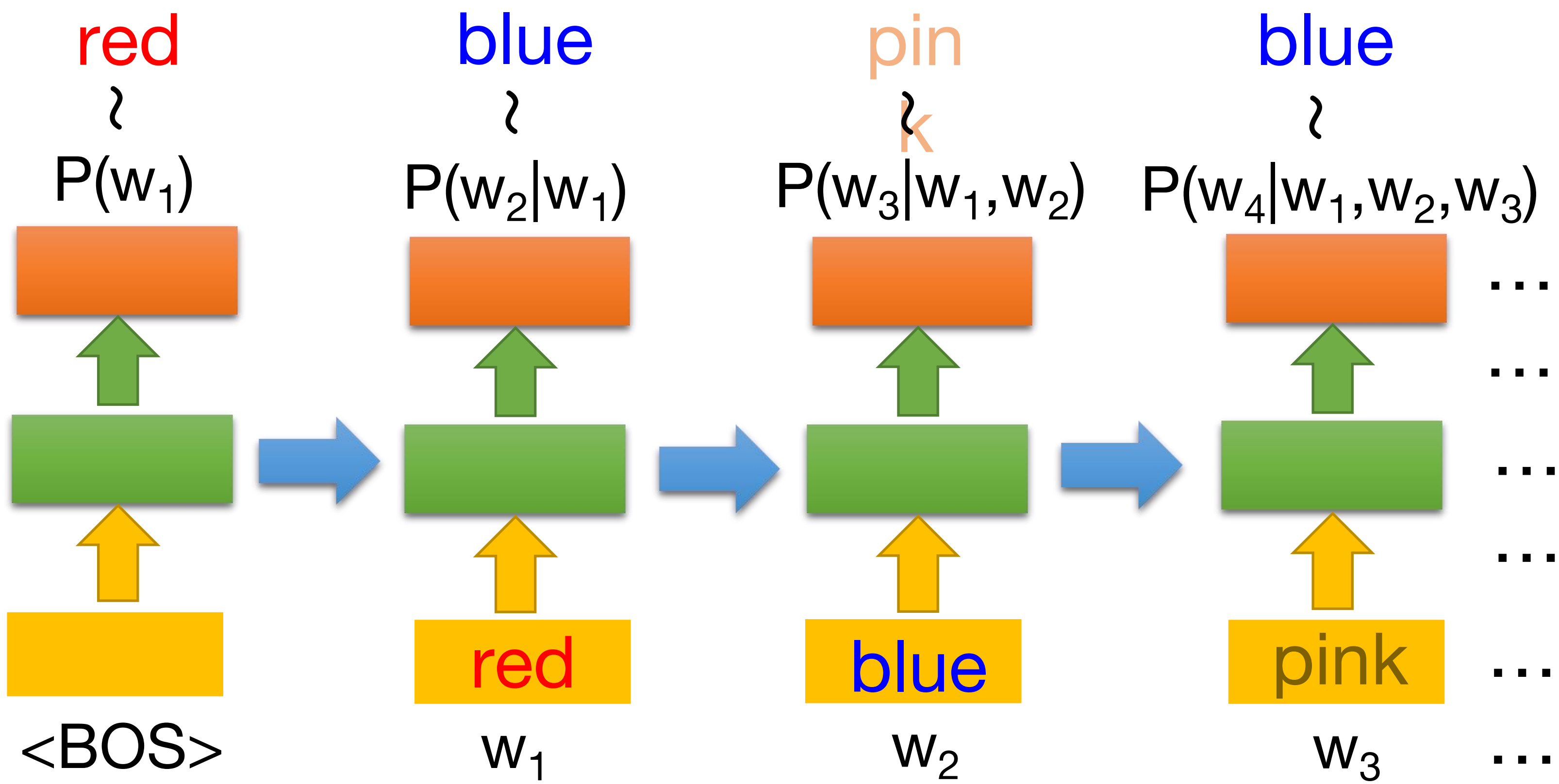
Generation



Consider as a sentence

blue red yellow gray……

- Images are composed of pixels
    - Generating a pixel at each time by RNN



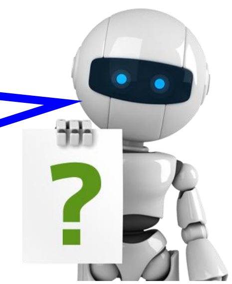
Conditional Generation

- We don't want to simply generate some random sentences.
- Generate sentences based on conditions:

Caption Generation

Given condition:

"A young girl is dancing."

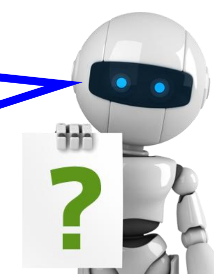


Chat-bot

Given condition:

"Hello"

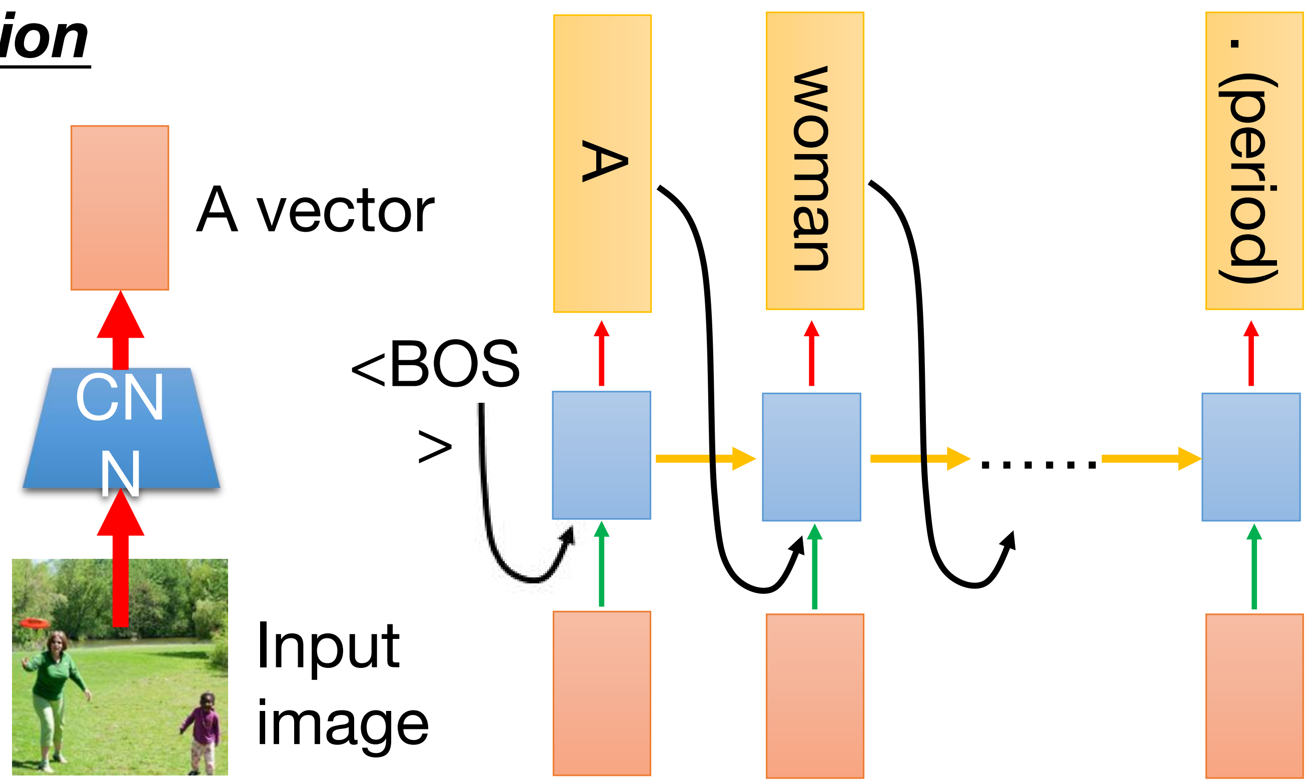
"Hello. Nice to see you."



Conditional Generation

- Represent the input condition as a vector, and consider the vector as the input of RNN generator

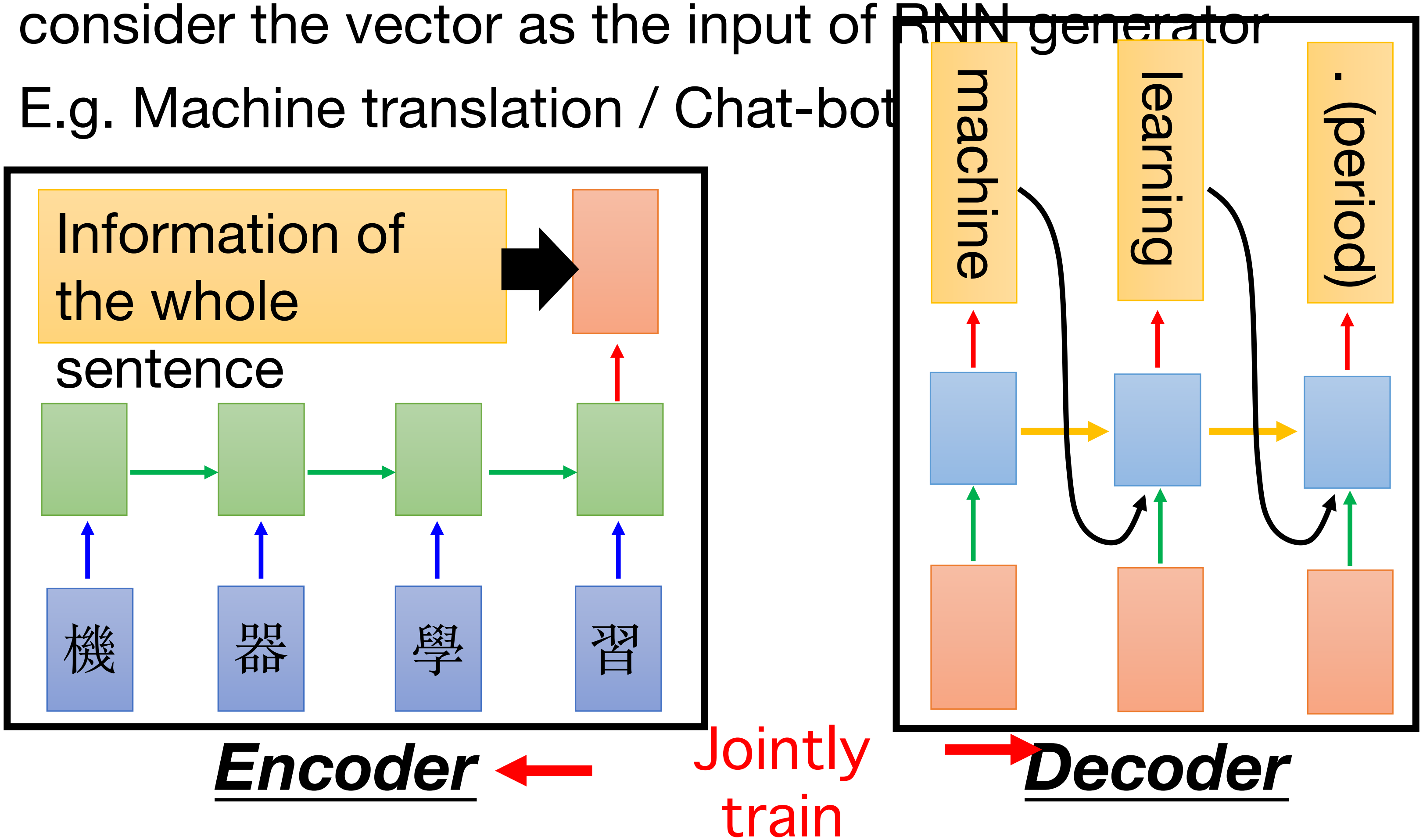
Image Caption Generation



Conditional Generation

Sequence-to-sequence learning

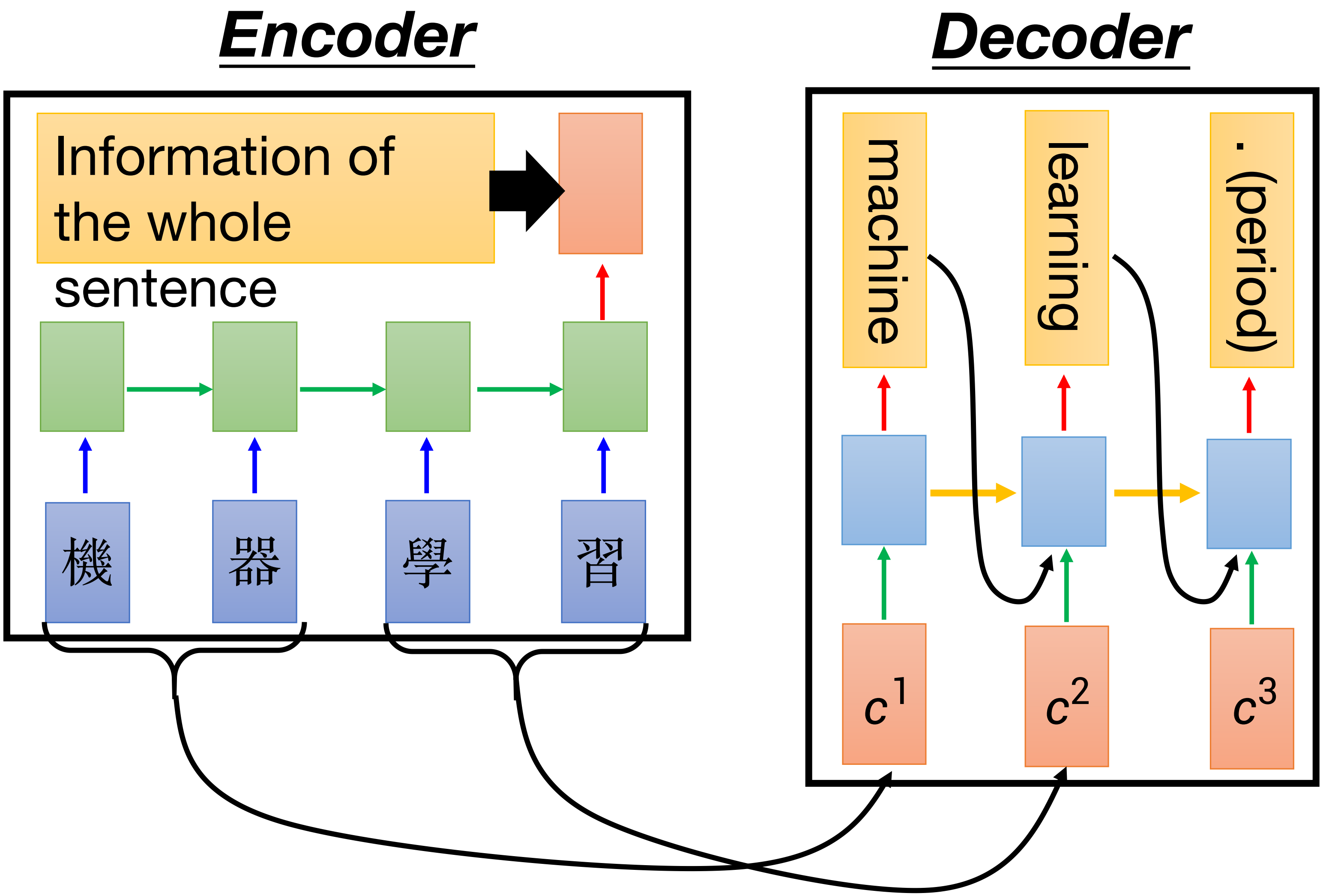
- Represent the input condition as a vector, and consider the vector as the input of RNN generator
- E.g. Machine translation / Chat-bot



Attention

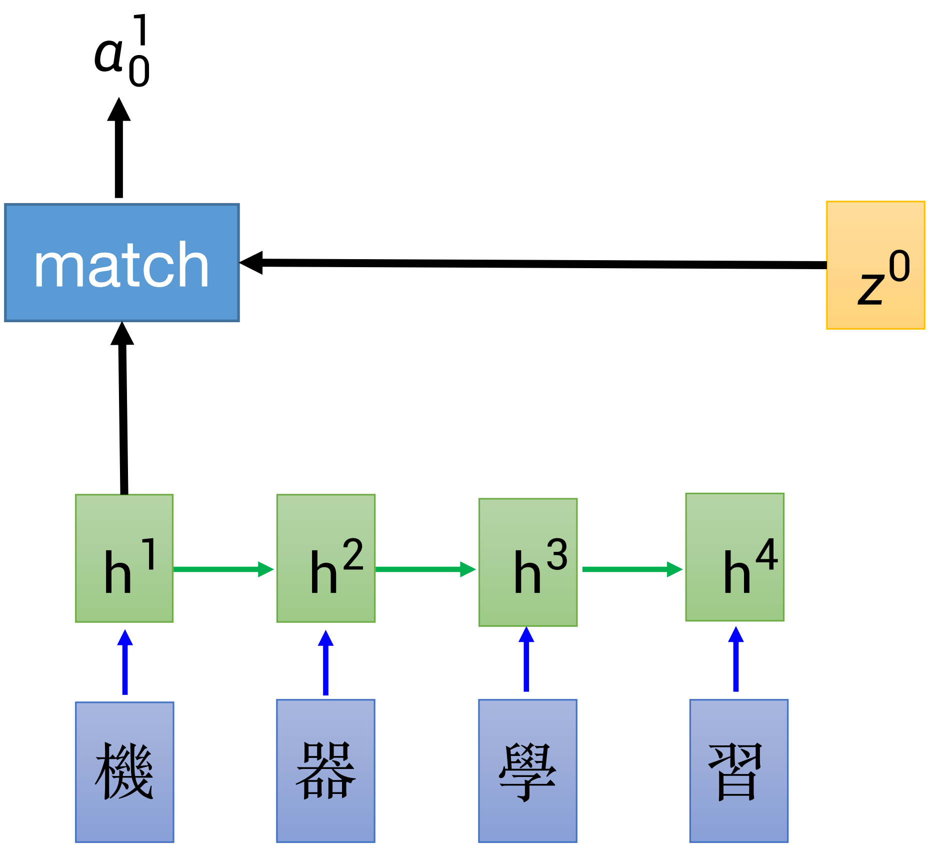
Dynamic Conditional Generation

Dynamic Conditional Generation

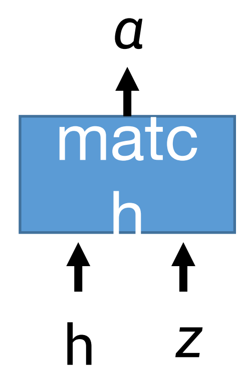


Machine Translation

- Attention-based model



Jointly learned with other part of the network



What is match?

Design by yourself

- Cosine similarity of z and h
- Small NN whose input is z and h, output a scalar
- $\alpha = h^T W z$

Definition [edit]

The cosine of two non-zero vectors can be derived by using the Euclidean dot product formula:

$$\mathbf{A} \cdot \mathbf{B} = \|\mathbf{A}\| \|\mathbf{B}\| \cos\theta$$

Given two vectors of attributes, *A* and *B*, the cosine similarity, cos(θ), is represented using a dot product and magnitude as

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum\limits_{i=1}^{n} A_i B_i}{\sqrt{\sum\limits_{i=1}^{n} A_i^2} \sqrt{\sum\limits_{i=1}^{n} B_i^2}},$$

where A_i and B_i are components of vector A and B respectively.

The resulting similarity ranges from −1 meaning exactly opposite, to 1 meaning exactly the same, with 0 indicating orthogonality or decorrelation, while in-between values indicate intermediate similarity or dissimilarity.

For text matching, the attribute vectors *A* and *B* are usually the term frequency vectors of the documents. Cosine similarity can be seen as a method of normalizing document length during comparison.

In the case of information retrieval, the cosine similarity of two documents will range from 0 to 1, since the term frequencies (using tf–idf weights) cannot be negative. The angle between two term frequency vectors cannot be greater than 90°.

NIPS17 Attention Is All You Need

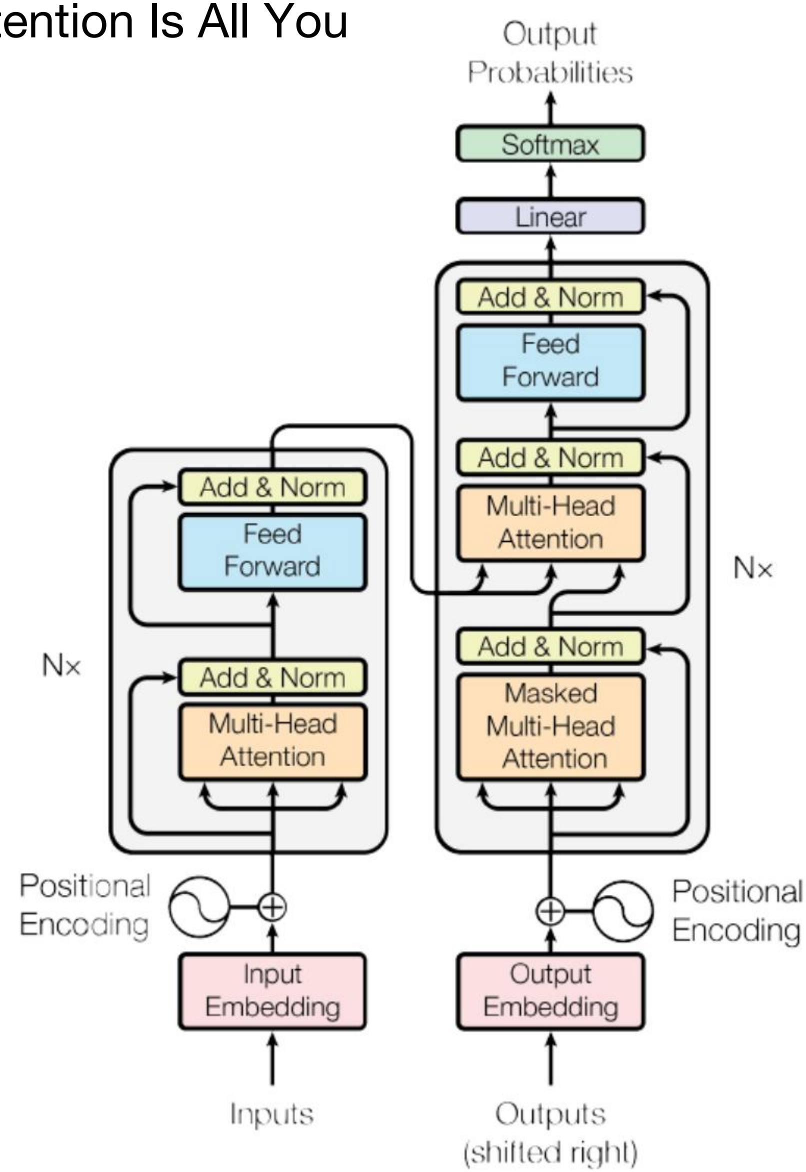
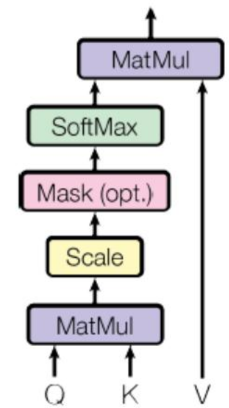
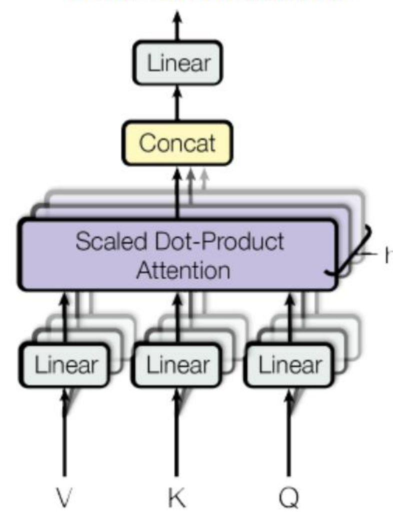


Figure 1: The Transformer - model architecture.

Scaled Dot-Product Attention

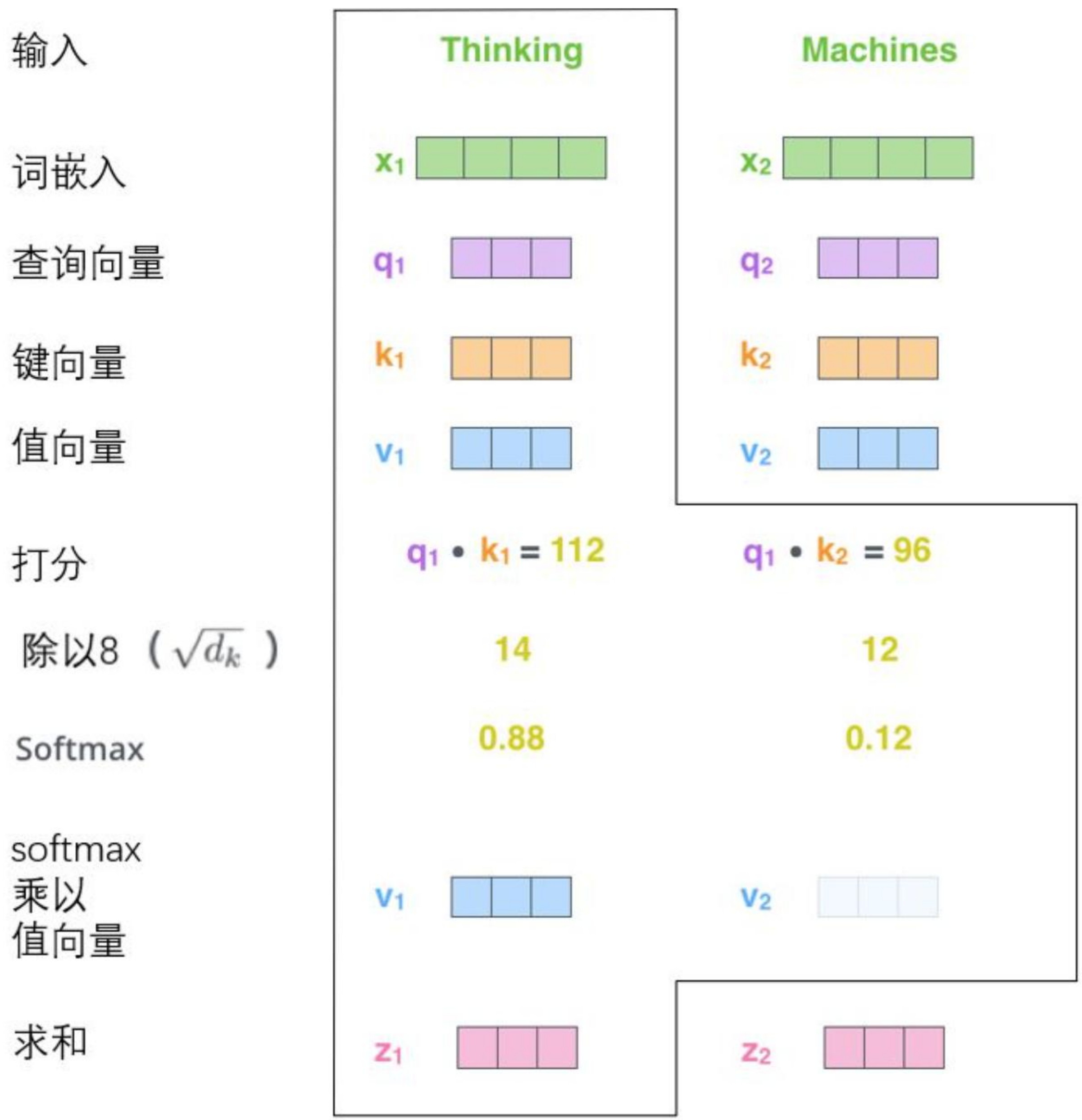


Multi-Head Attention



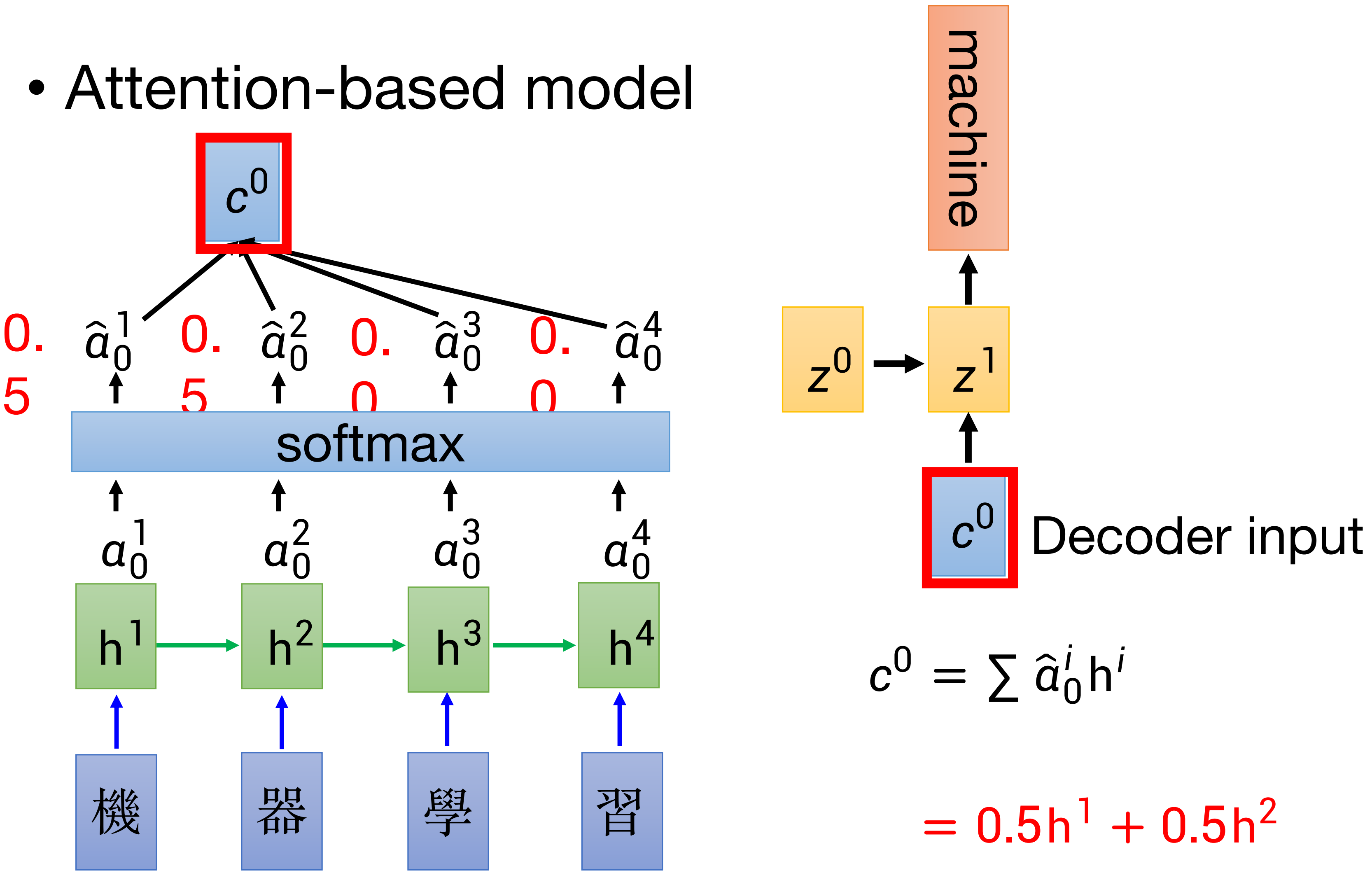
$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

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



Machine Translation

- Attention-based model

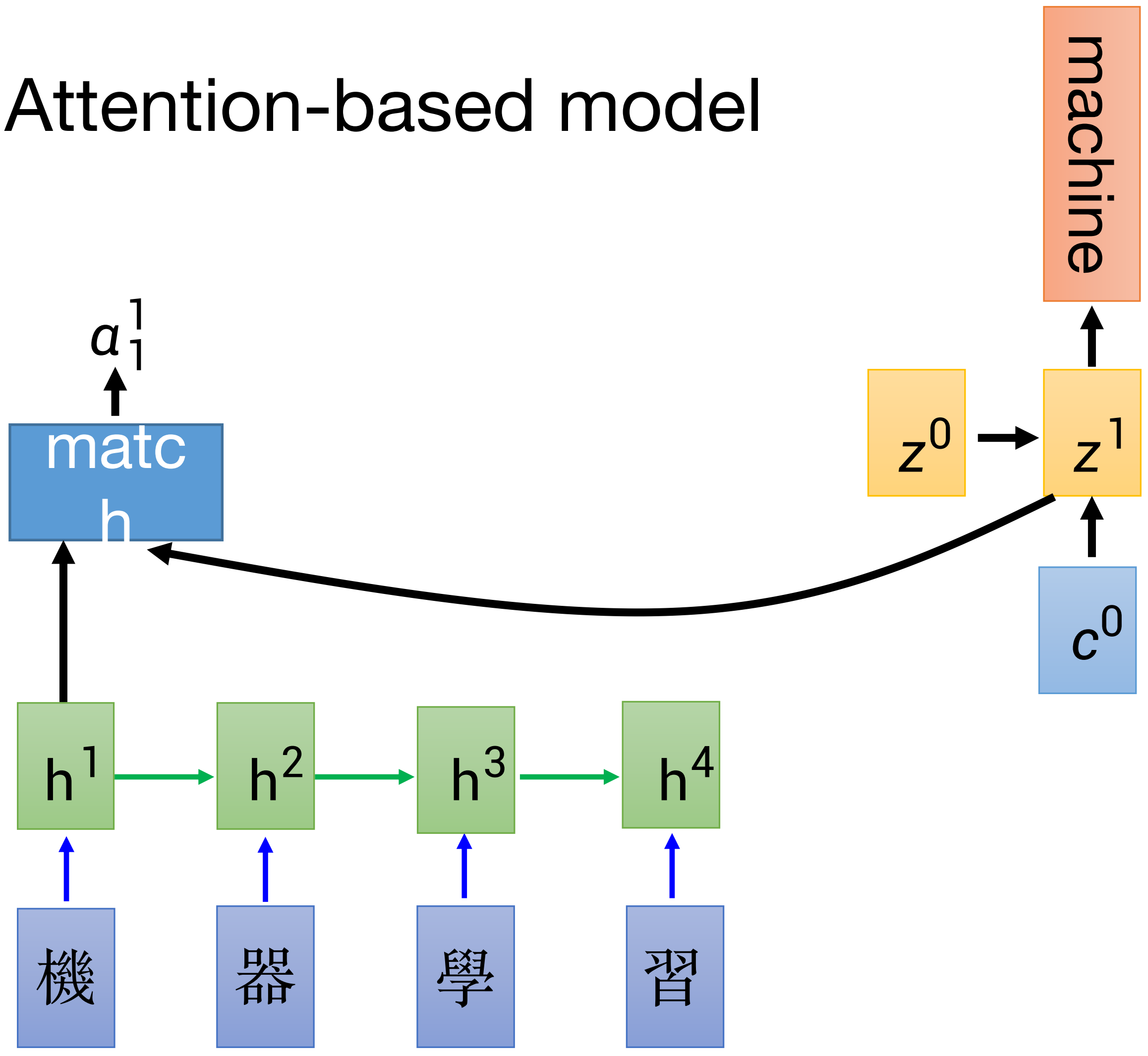


$$c^0 = \sum \hat{a}_0^i h^i$$

$$= 0.5h^1 + 0.5h^2$$

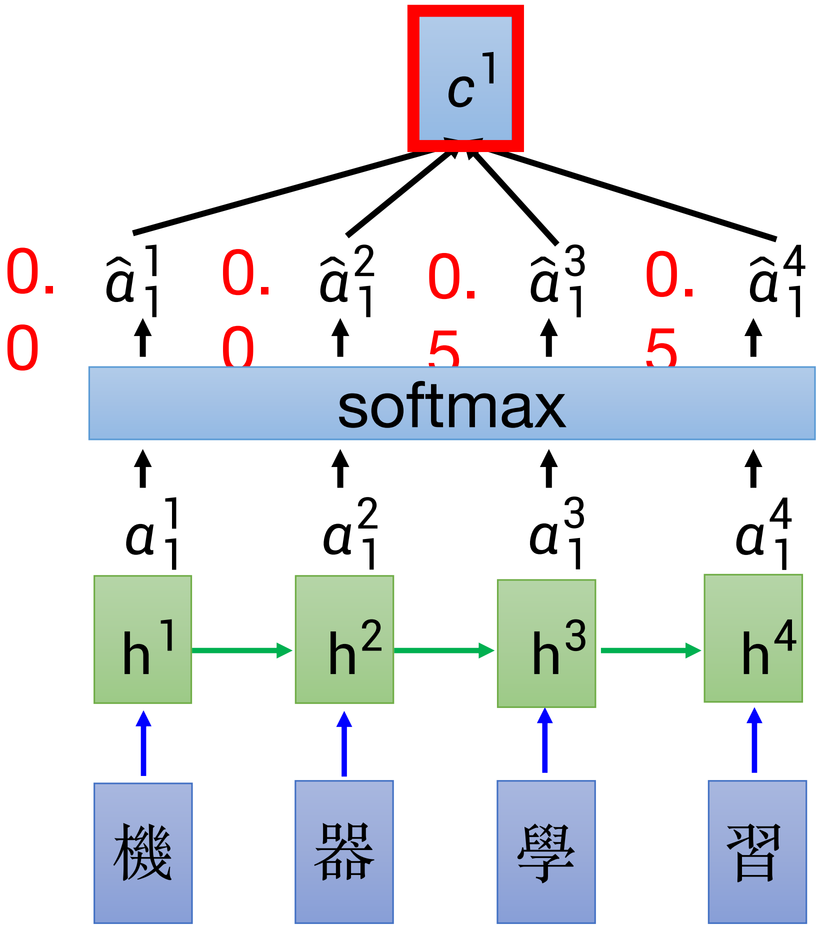
Machine Translation

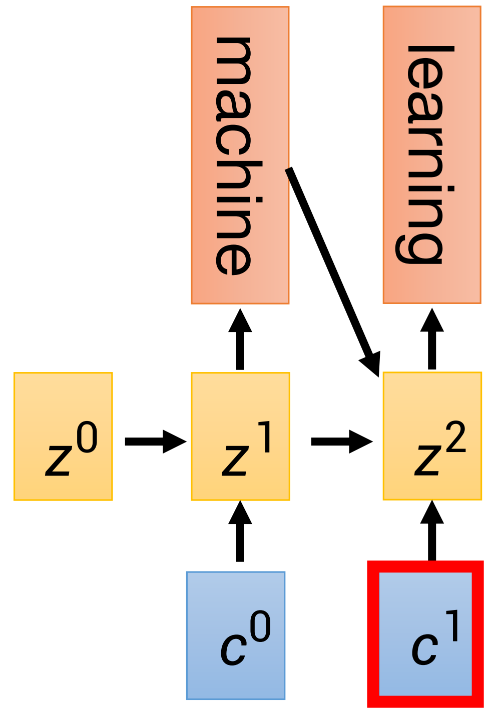
- Attention-based model



Machine Translation

- Attention-based model



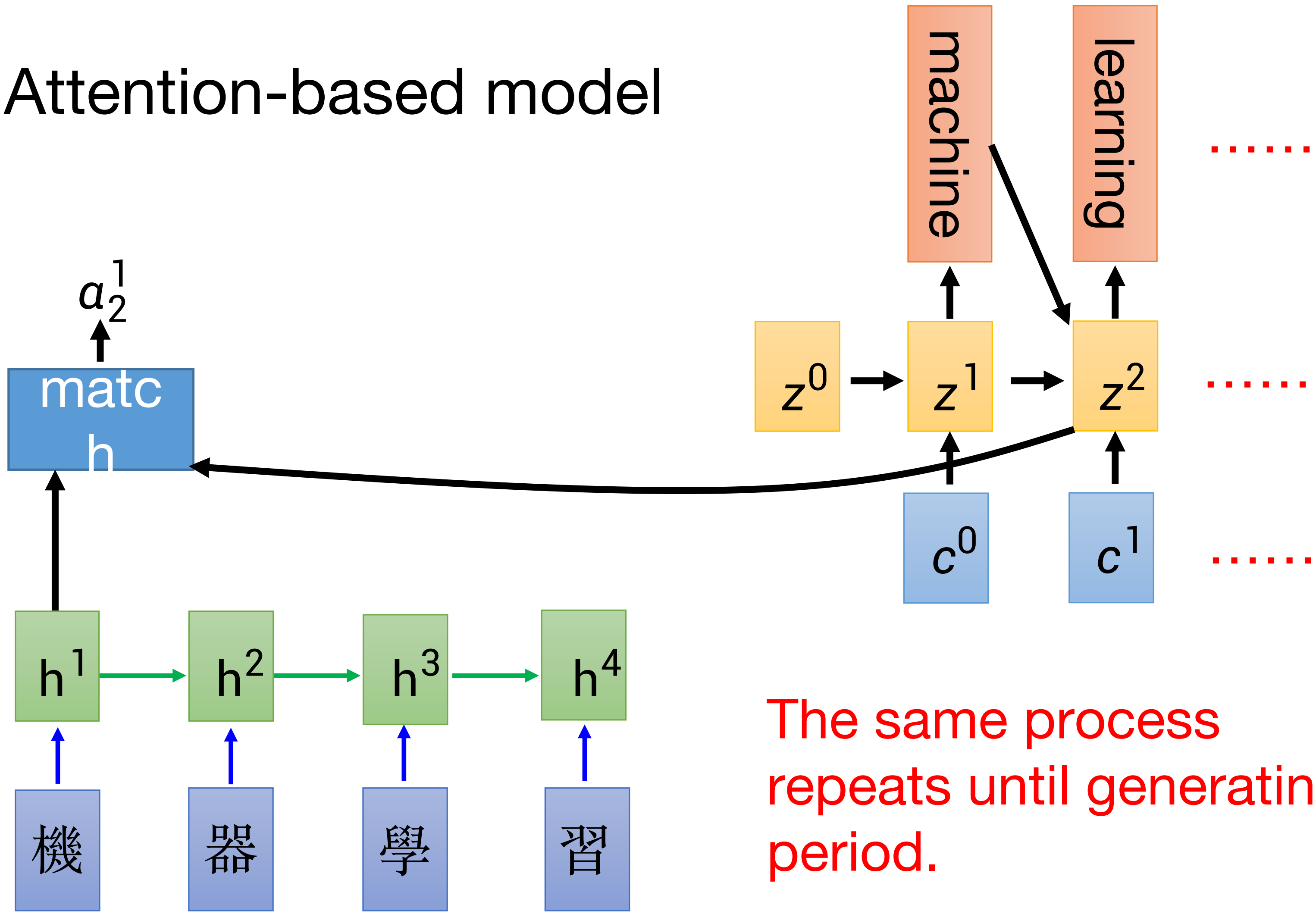


$$c^1 = \sum \hat{\alpha}_1^i h^i$$

$$= 0.5h^3 + 0.5h^4$$

Machine Translation

- Attention-based model



The same process repeats until generating period.

Image Caption Generation

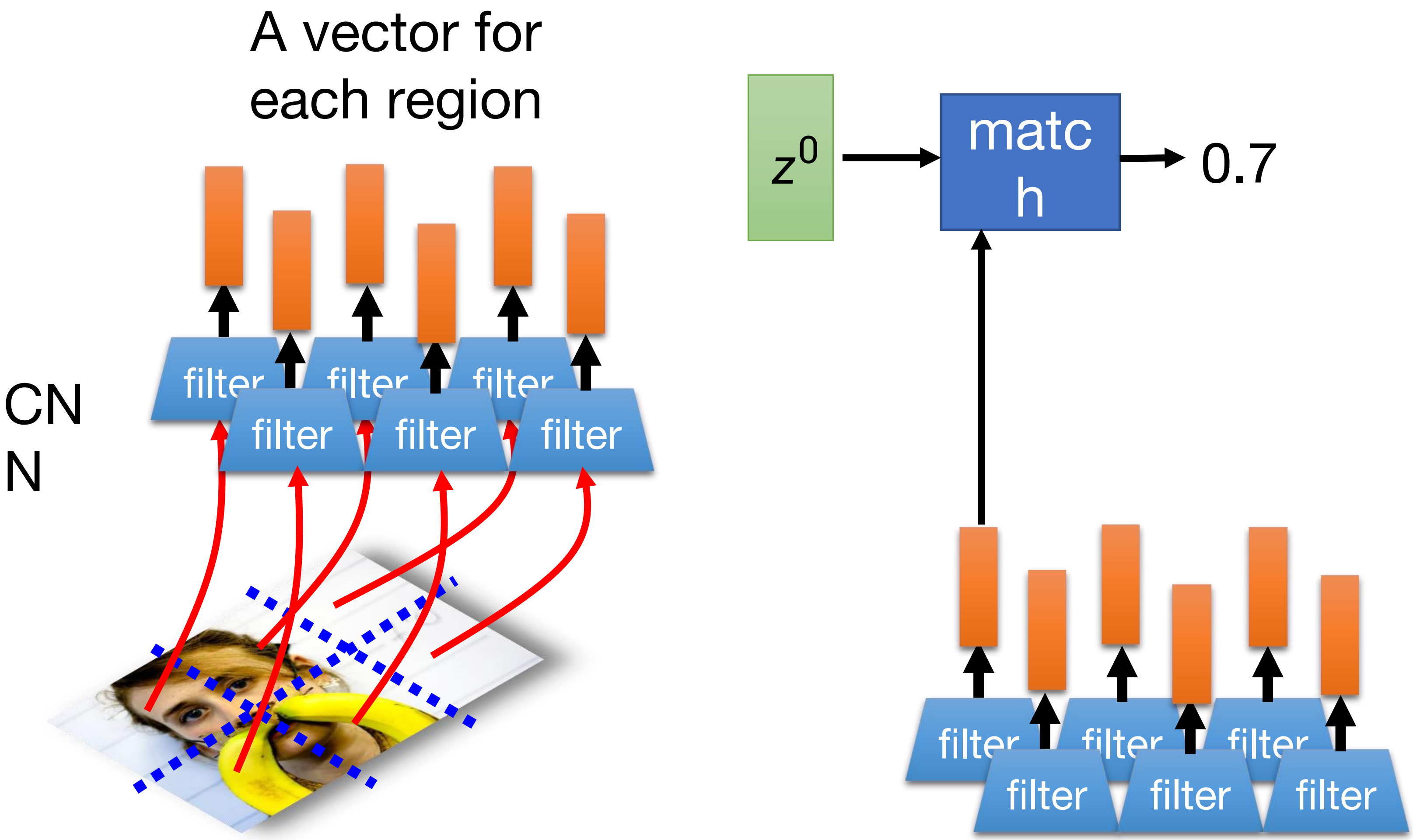


Image Caption Generation

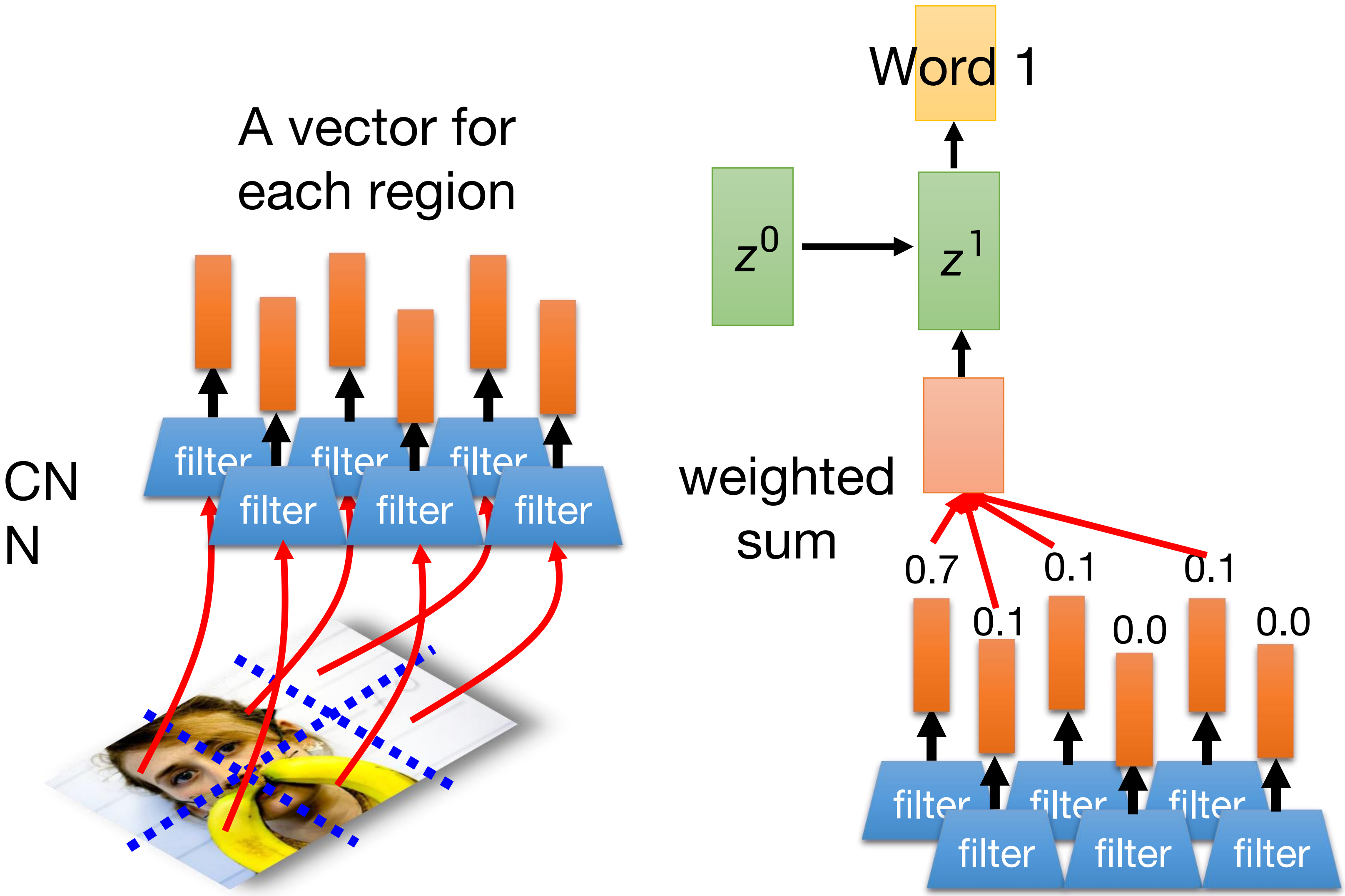


Image Caption Generation

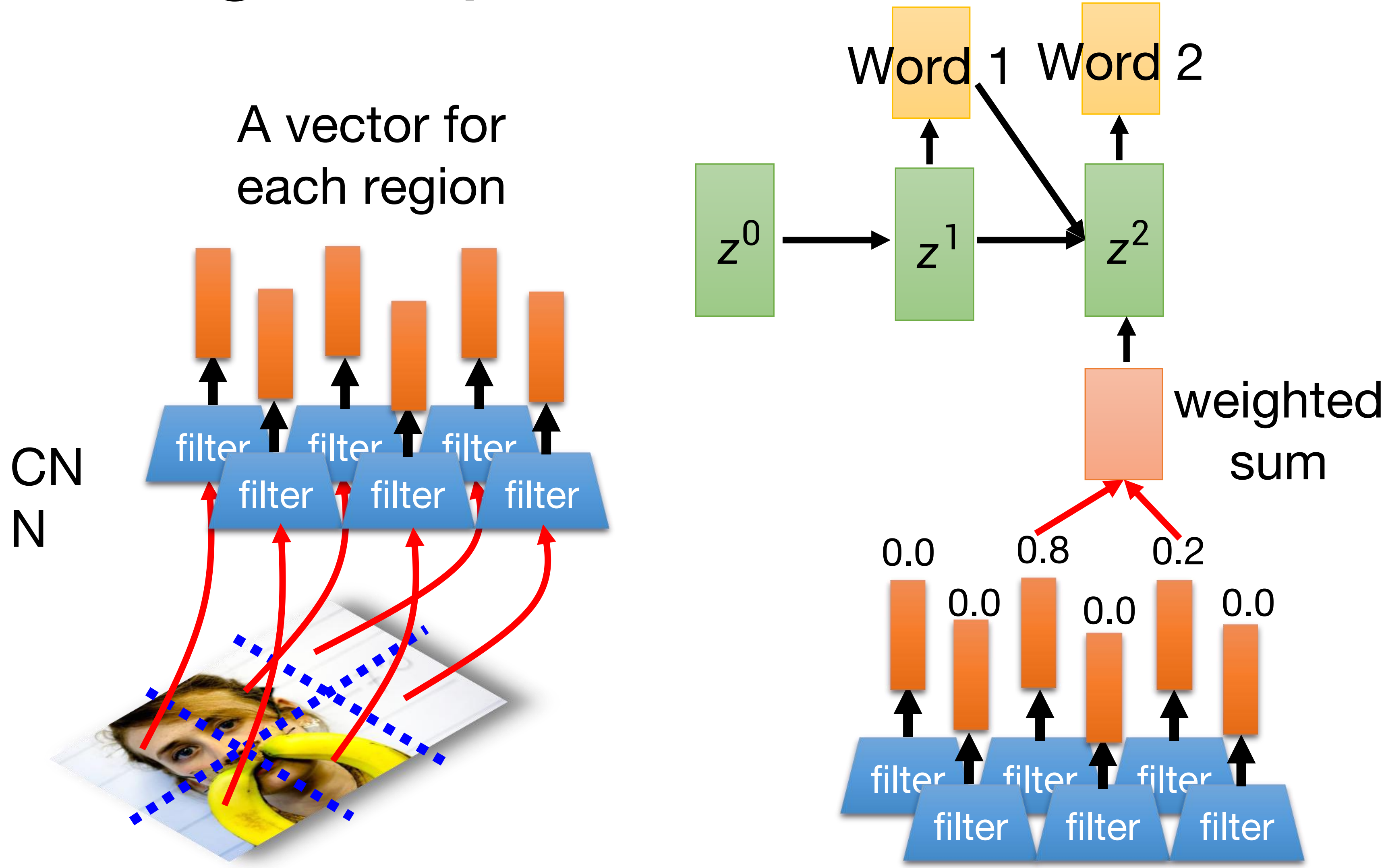


Image Caption Generation (positive samples)

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.

Image Caption Generation (negative samples)

A large white bird standing in a forest.

A woman holding a clock in her hand.

A man wearing a hat and
a hat on a skateboard.

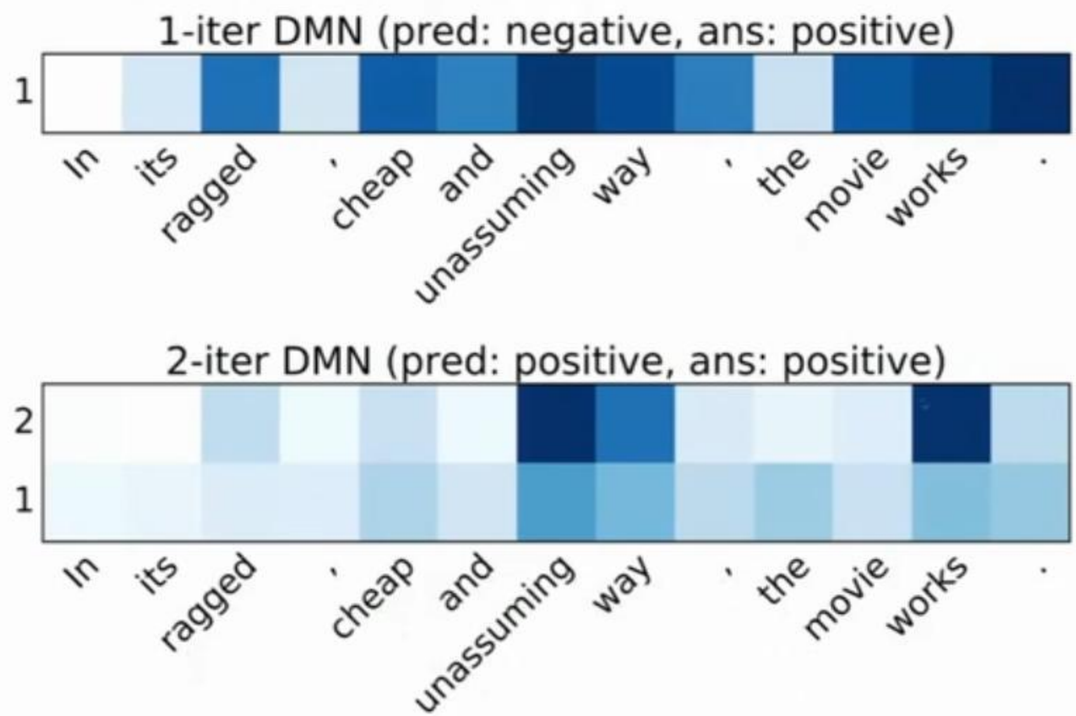
A person is standing on a beach
with a surfboard.

A woman is sitting at a table
with a large pizza.

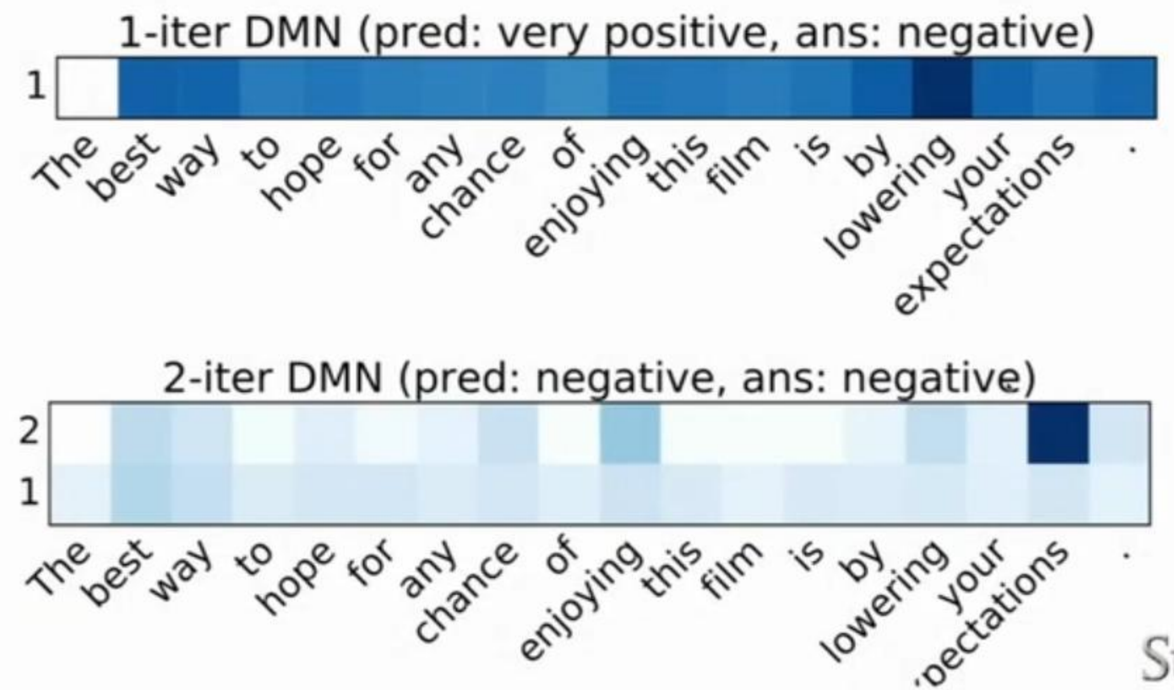
A man is talking on his cell phone
while another man watches.

Analysis of Attention for Sentiment

- Sharper attention when 2 passes are allowed.
- Examples that are wrong with just one pass



Analysis of Attention for Sentiment



Analysis of Attention for Sentiment

- Examples where full sentence context from first pass changes attention to words more relevant for final prediction