BIG DATA and AI
for business

Lecture 4 (02/14, 02/18): Deep Learning (2)

Decisions, Operations & Information Technologies
Robert H. Smith School of Business
Spring, 2019

Variants of Neural Networks

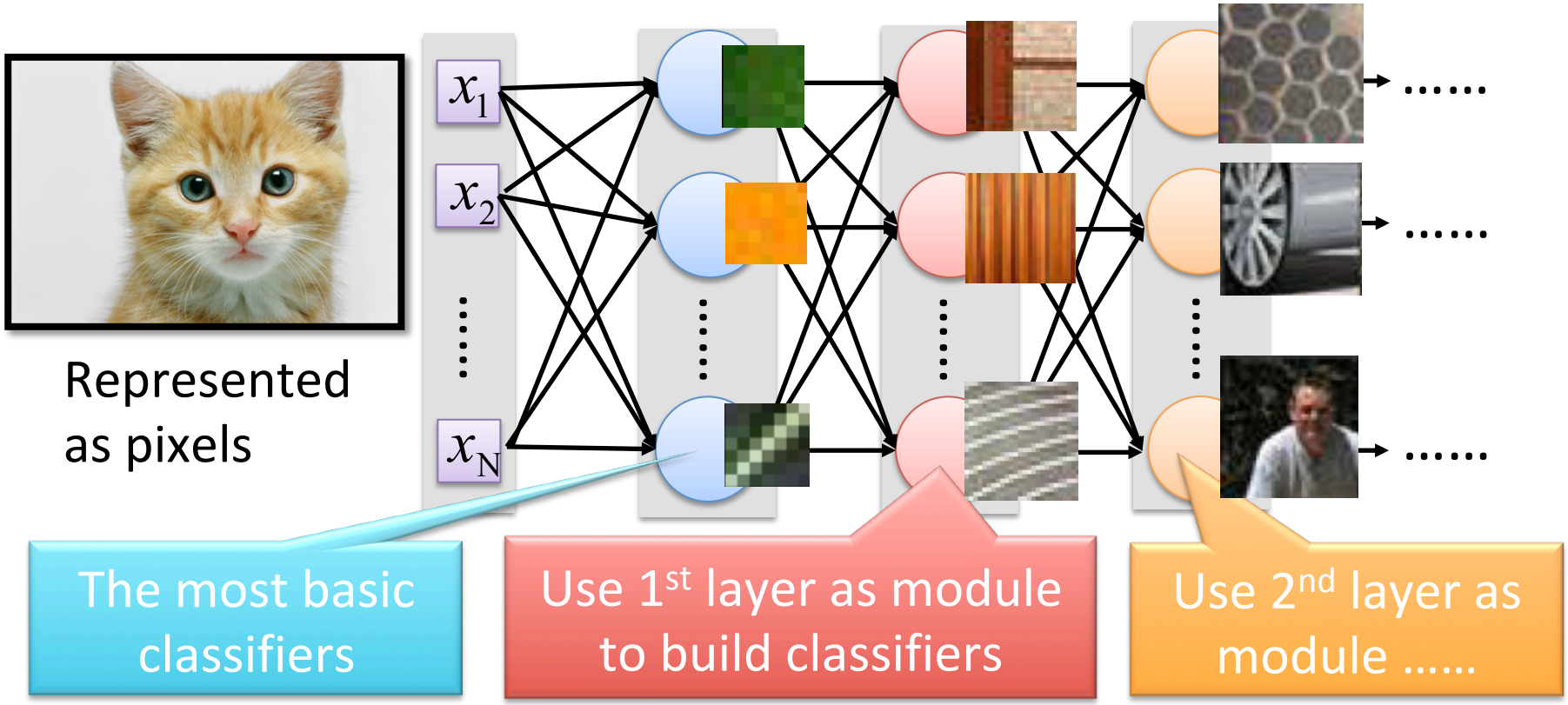
Convolutional Neural Network (CNN)

Widely used in image processing

Recurrent Neural Network (RNN)

Why CNN for Image?

[Zeiler, M. D., *ECCV 2014*]



Can the network be simplified by considering the properties of images?

Why CNN for Image

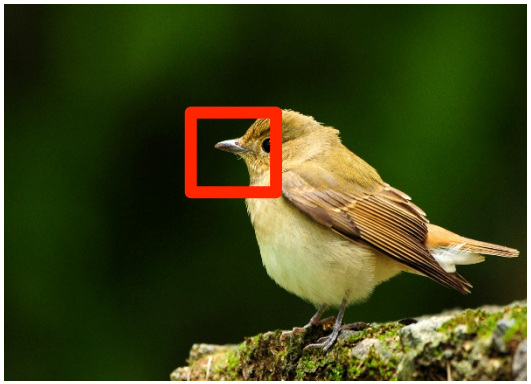
- Some patterns are much smaller than the whole image

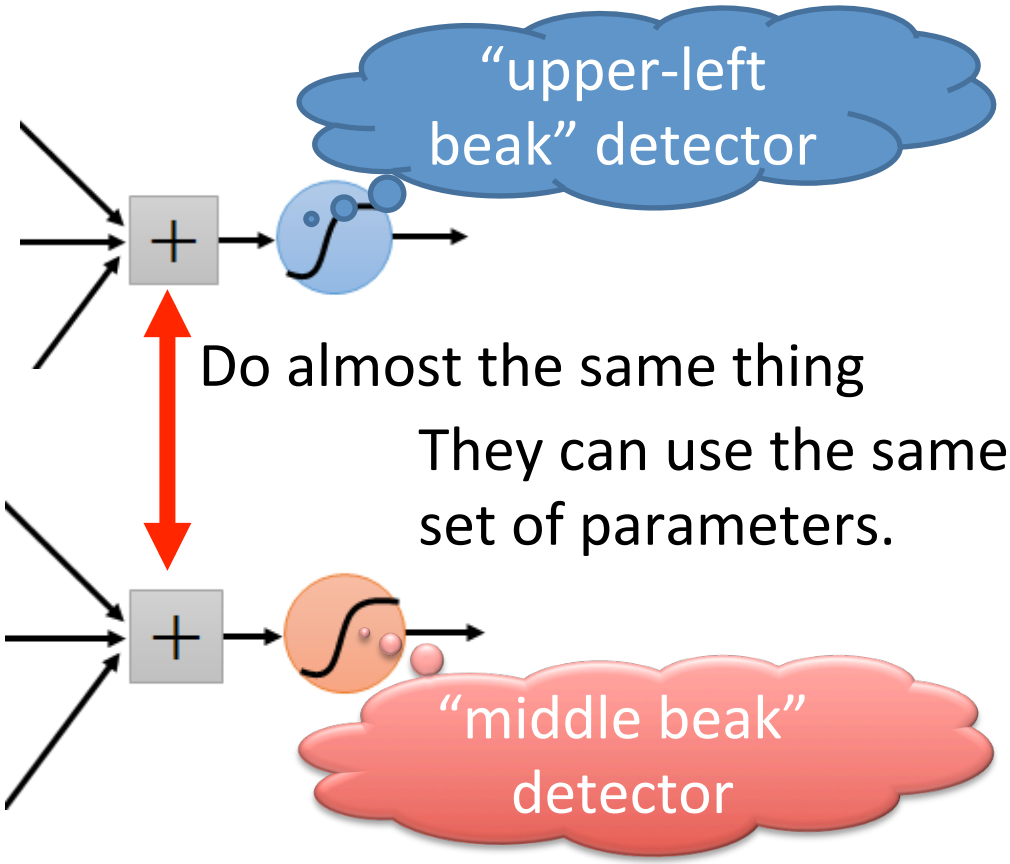
A neuron does not have to see the whole image to discover the pattern.

Connecting to small region with less parameters

Why CNN for Image

- The same patterns appear in different regions.





Why CNN for Image

- Subsampling the pixels will not change the object

bird

subsampling

bird

We can subsample the pixels to make image smaller

→ Less parameters for the network to process the image

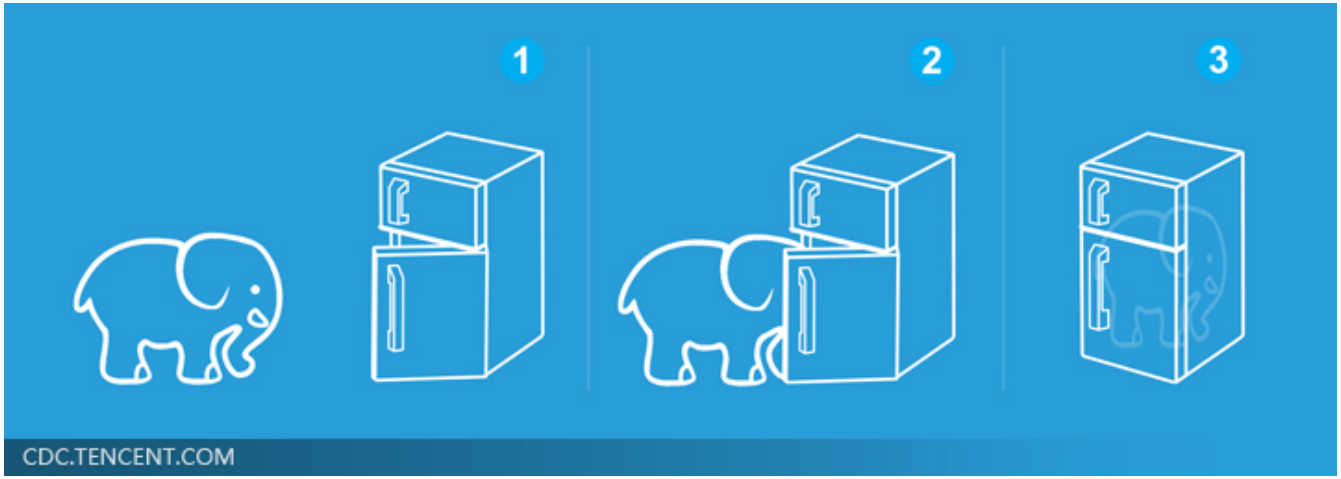
Three Steps for Deep Learning

Step 1: Convolutional Neural Network

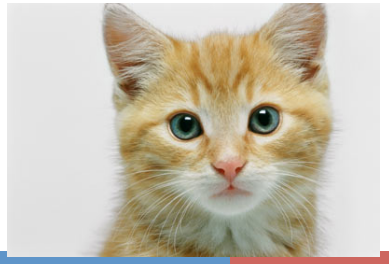
Step 2: goodness of function

Step 3: pick the best function

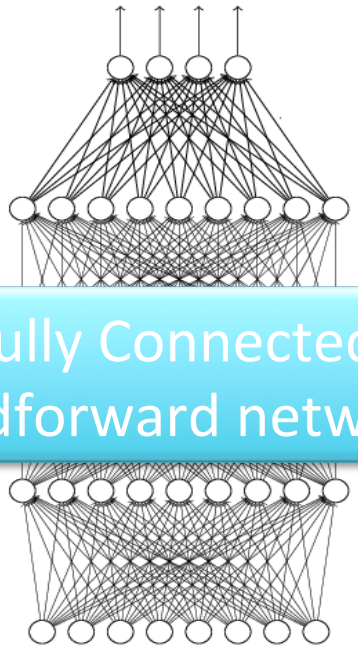
Deep Learning is so simple ……



The whole CNN



cat dog ……



Fully Connected Feedforward network

Convolution

Max Pooling

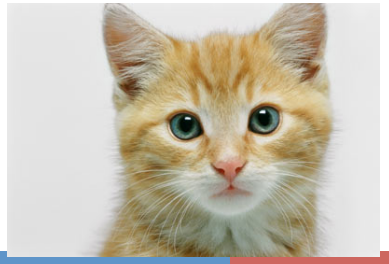
Convolution

Max Pooling

Can repeat many times

Flatten

The whole CNN



Property 1

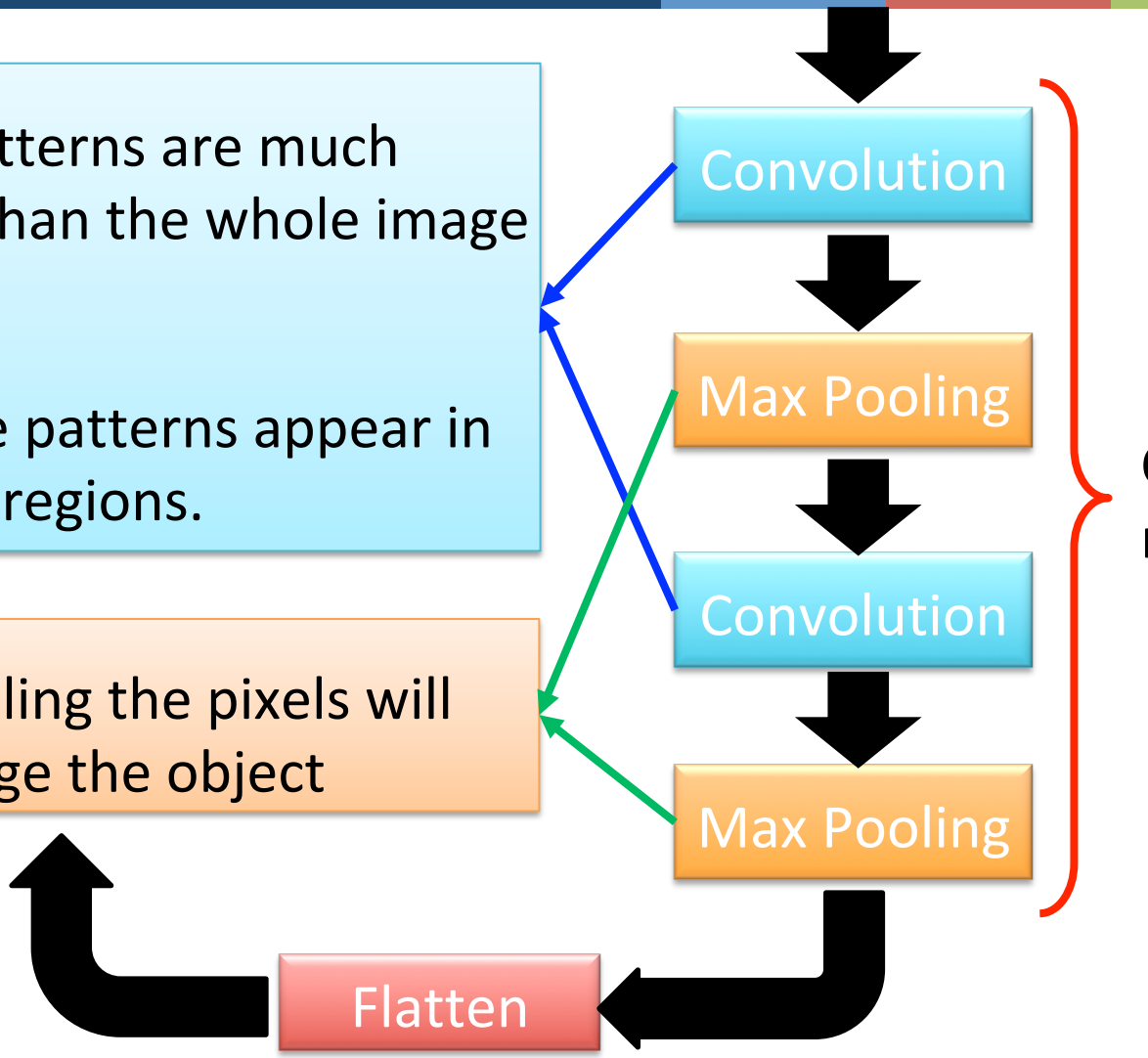
- Some patterns are much smaller than the whole image

Property 2

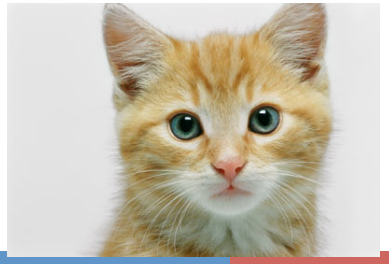
- The same patterns appear in different regions.

Property 3

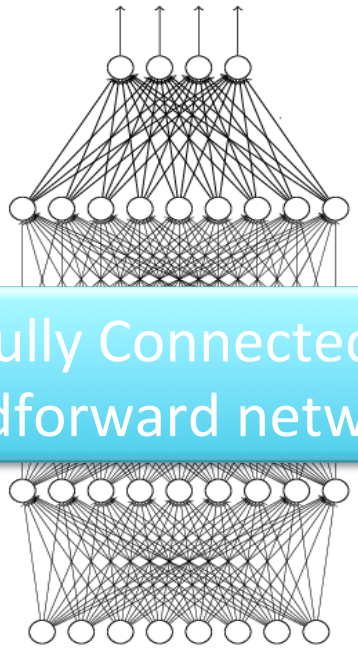
- Subsampling the pixels will not change the object



The whole CNN



cat dog ……



Fully Connected Feedforward network

Convolution

Max Pooling

Convolution

Max Pooling

Can repeat many times

Flatten

CNN – Convolution

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

Those are the network parameters to be learned.

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

Matrix

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

Matrix

⋮

Property 1

Each filter detects a small pattern (3 x 3).

CNN – Convolution

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

3 -1

CNN – Convolution

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

If stride=2

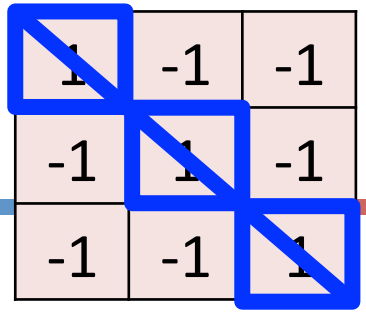
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

3 -3

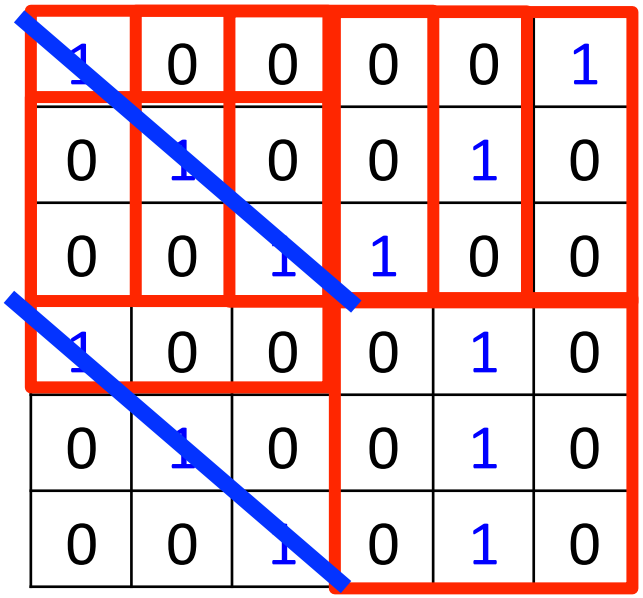
We set stride=1 below

CNN – Convolution

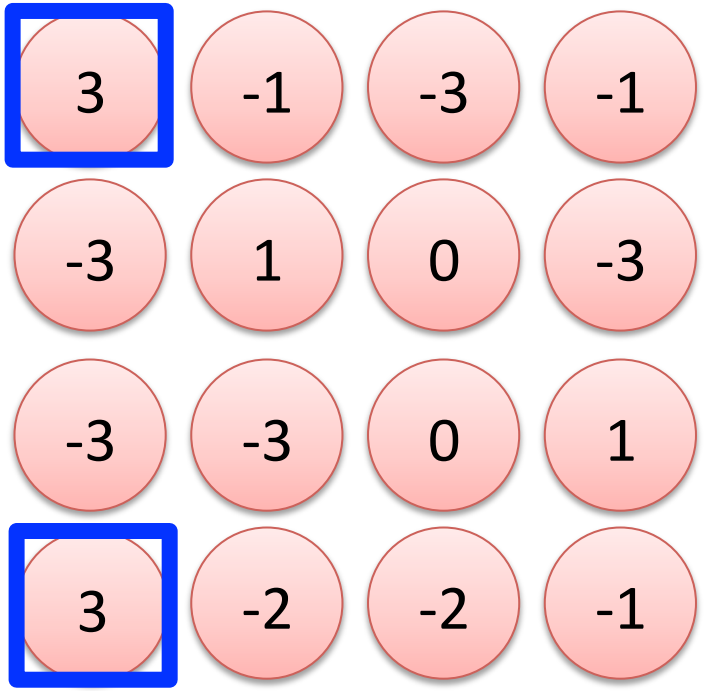


Filter 1

stride=1



6 x 6 image



Property 2

CNN – Convolution

-1	1	-1
-1	1	-1
-1	1	-1

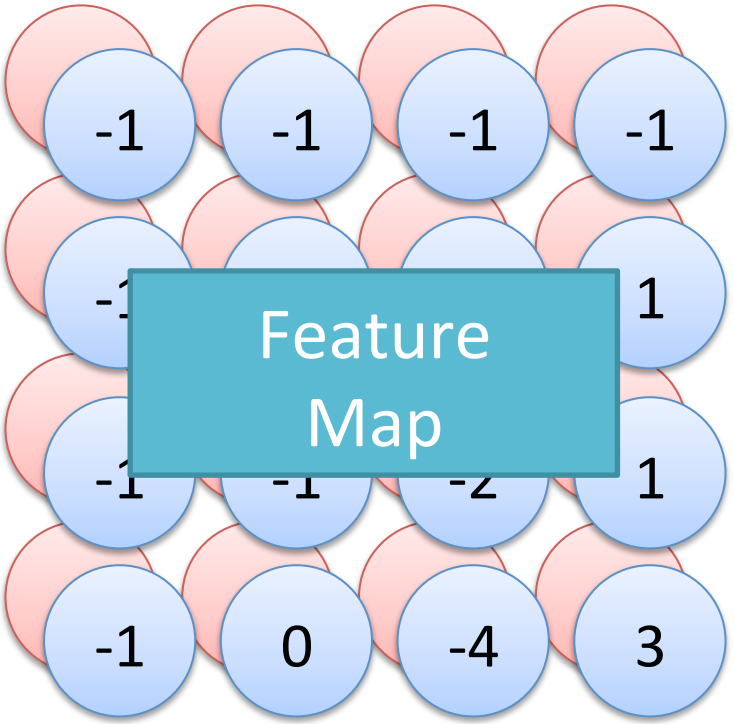
Filter 2

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

Do the same process for every filter

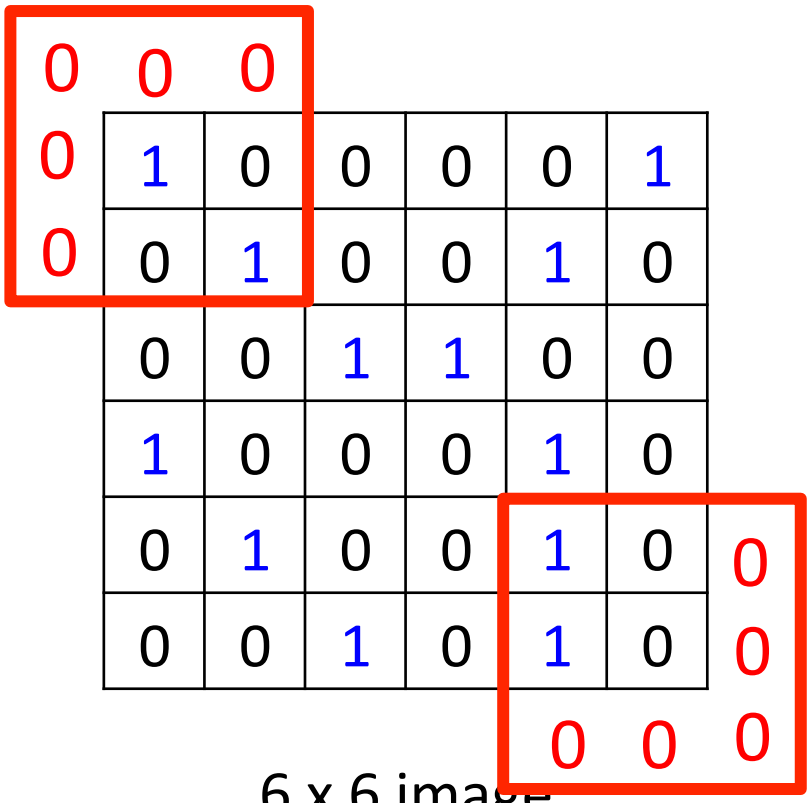


4 x 4 image

CNN – Zero Padding

1	-1	-1
-1	1	-1
-1	-1	1

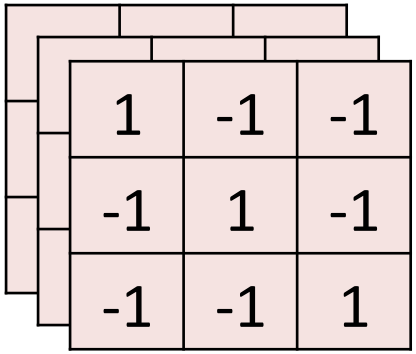
Filter 1



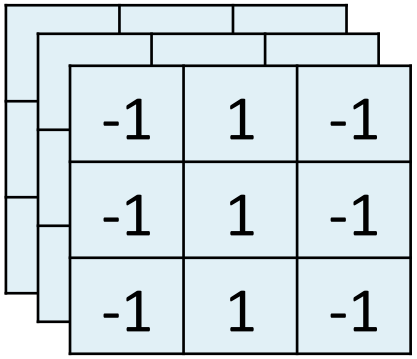
You will get another 6 x 6 images in this way

Zero padding

CNN – Colorful image

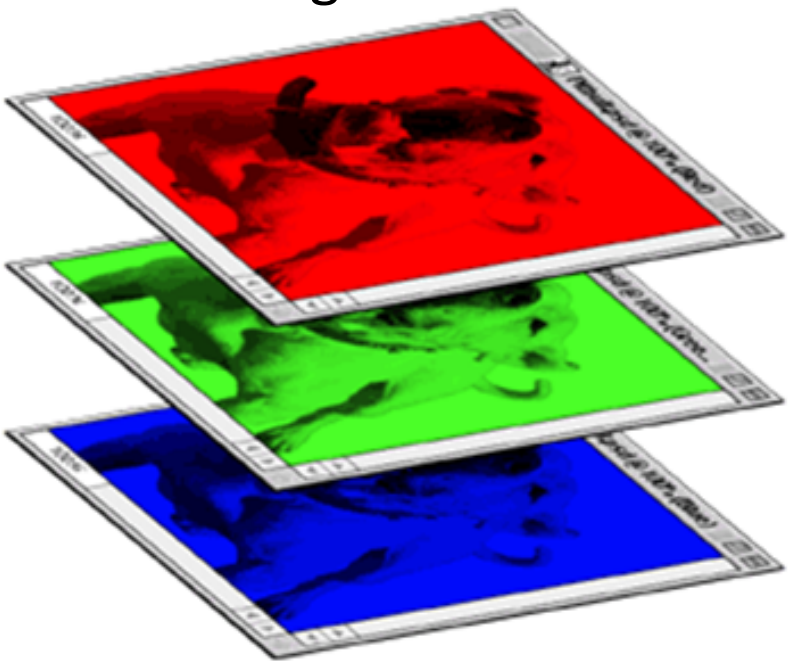


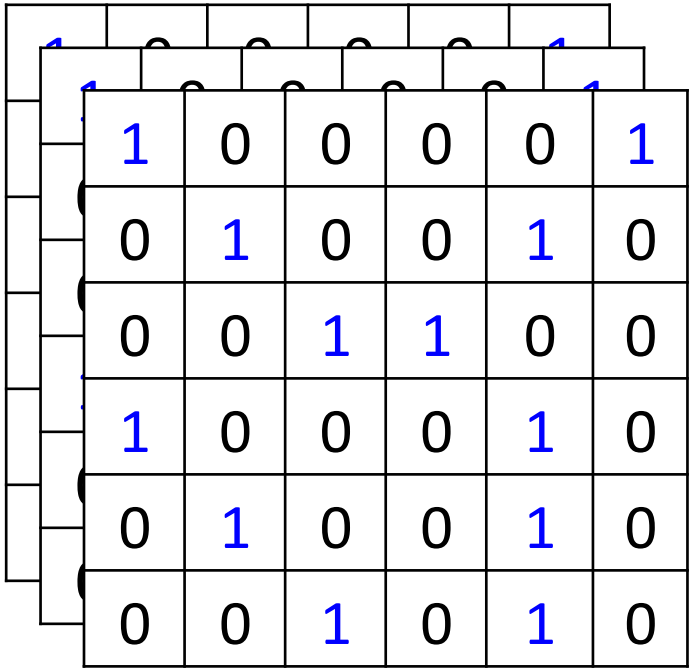
Filter 1



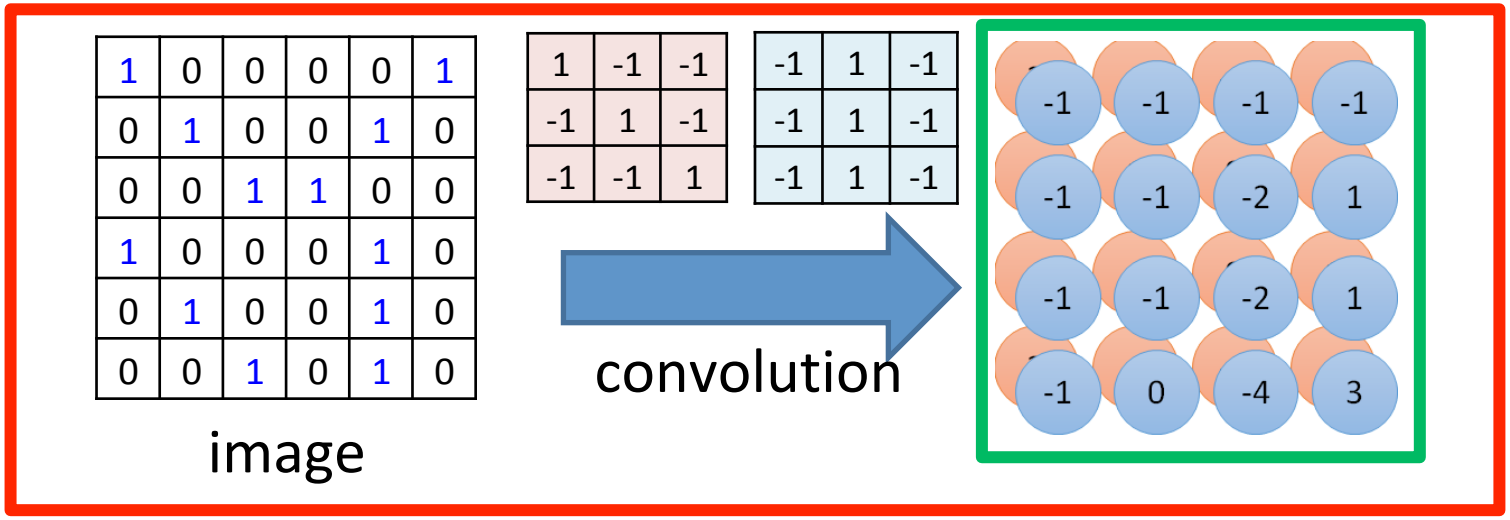
Filter 2

Colorful image



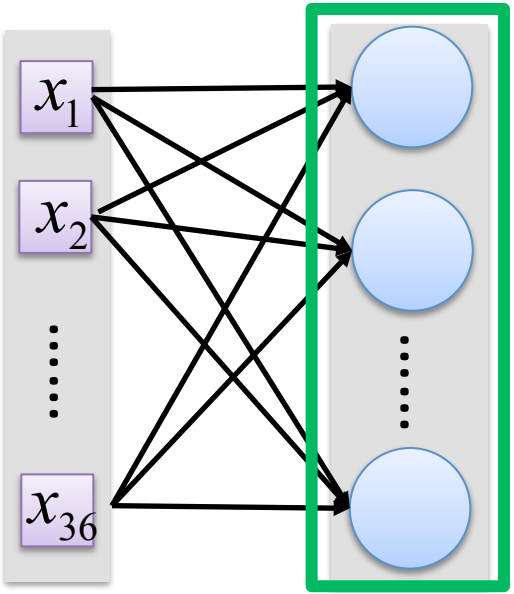


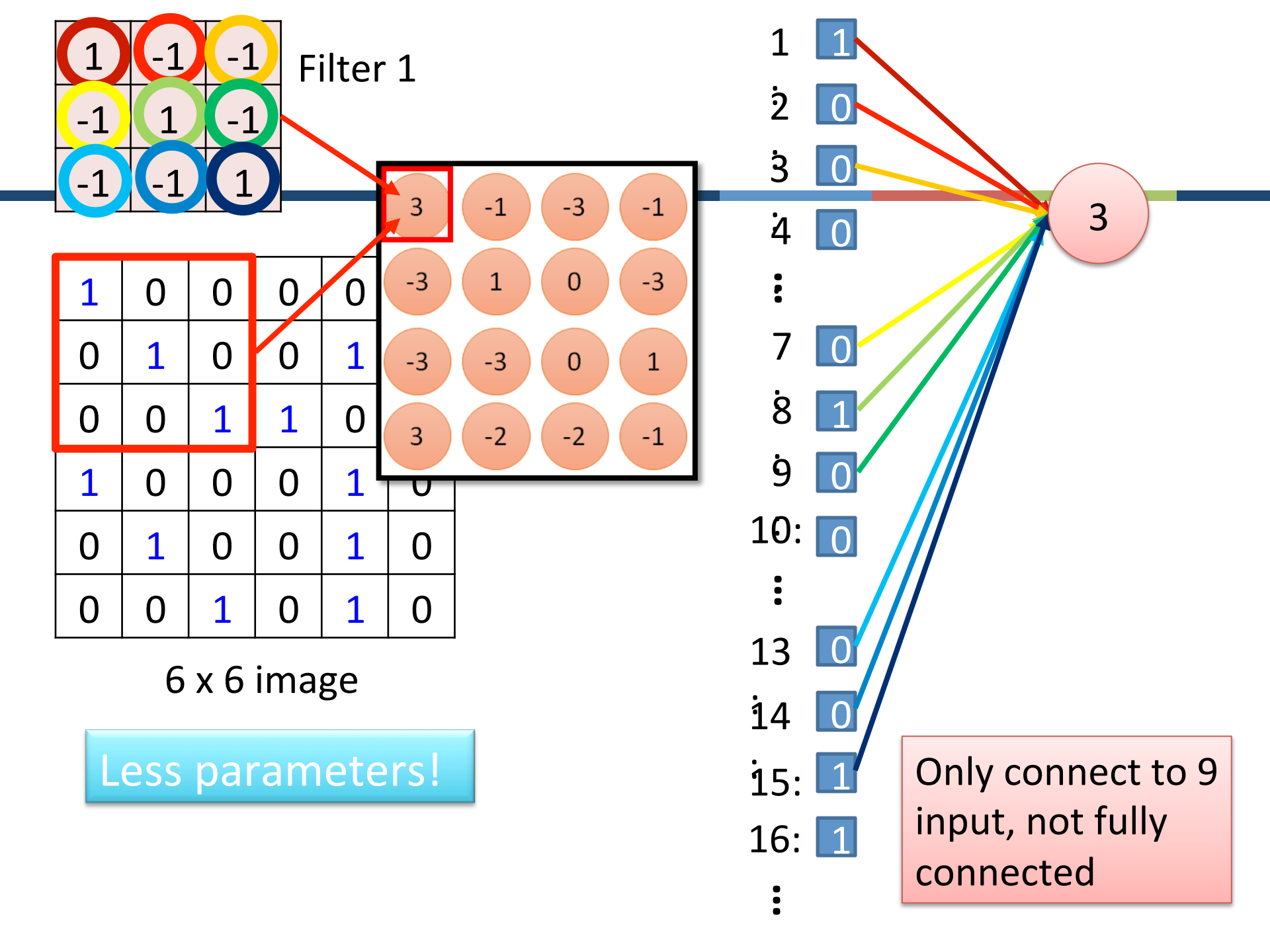
Convolution v.s. Fully Connected

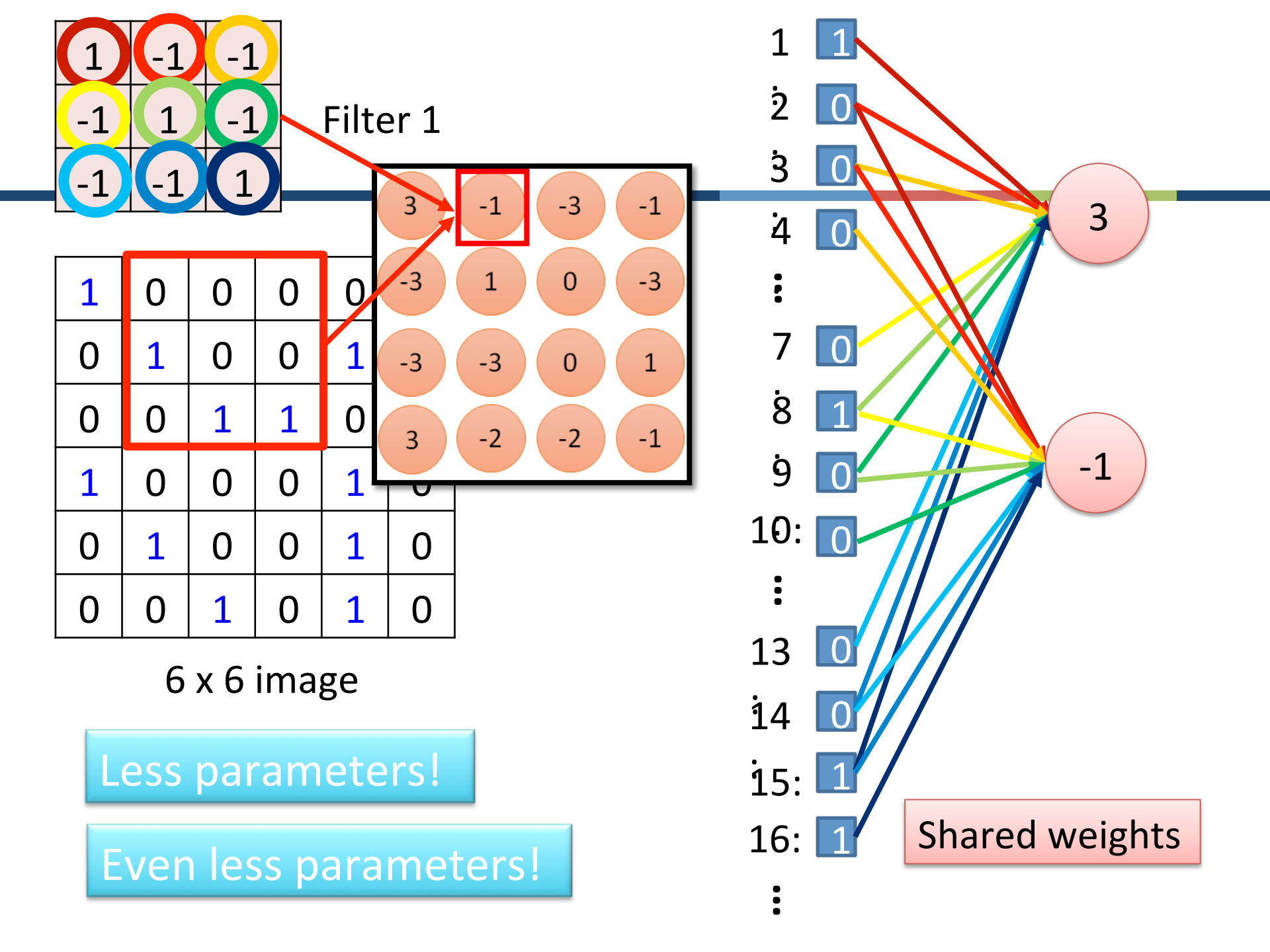


Fully-connected

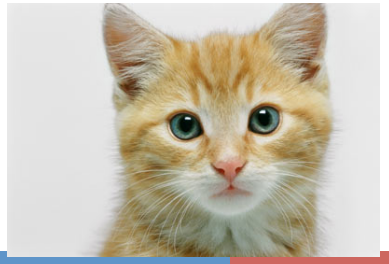
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0



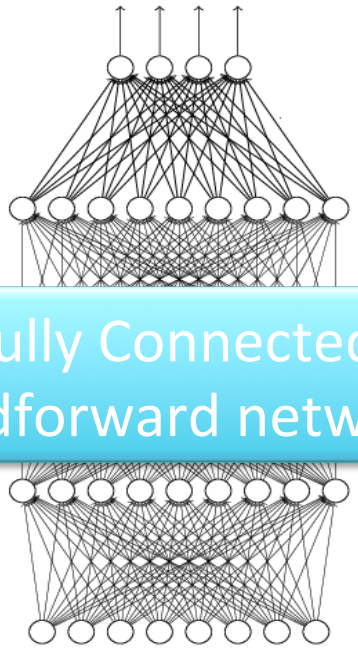

1 -1 -1
-1 1 -1
-1 -1 1
Filter 1
1 0 0 0 0
0 1 0 0 1
0 0 1 1 0
1 0 0 0 1 0
0 1 0 0 1 0
0 0 1 0 1 0
6 x 6 image
3 -1 -3 -1
-3 1 0 -3
-3 -3 0 1
3 -2 -2 -1
1
2
3
4
7
8
9
10:
13
14
15:
16:
1 0 0 0 1 0 0 1 0
3
Less parameters!
Only connect to 9 input, not fully connected


1 -1 -1
-1 1 -1
-1 -1 1
Filter 1
3 -1 -3 -1
-3 1 0 -3
-3 -3 0 1
3 -2 -2 -1
1 0 0 0 0
0 1 0 0 1
0 0 1 1 0
1 0 0 0 1
0 1 0 0 1 0
0 0 1 0 1 0
6 x 6 image
1
2
3
4
7
8
9
10:
13
14
15:
16:
1 0 0 0 1 0 0 0 1 0 0 1
3
-1
Shared weights
Less parameters!
Even less parameters!

The whole CNN



cat dog ……



Fully Connected Feedforward network

Convolution

Max Pooling

Convolution

Max Pooling

Can repeat many times

Flatten

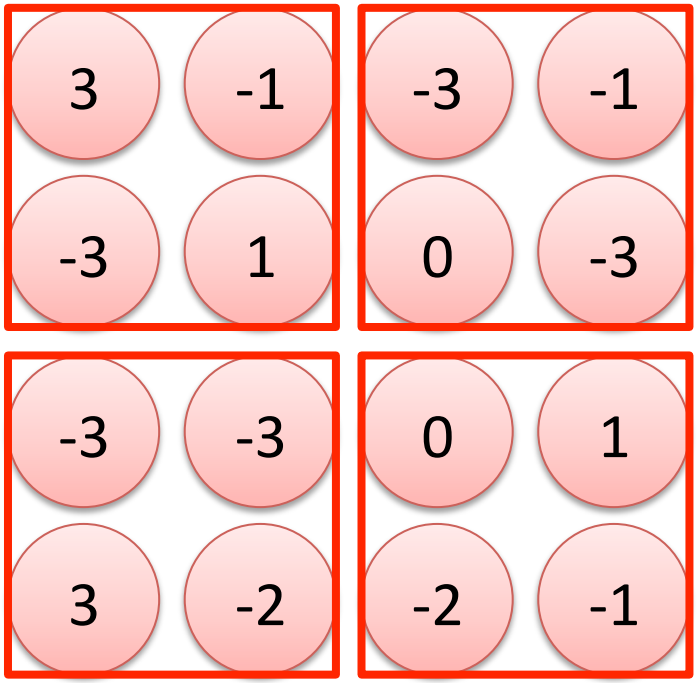
CNN – Max Pooling

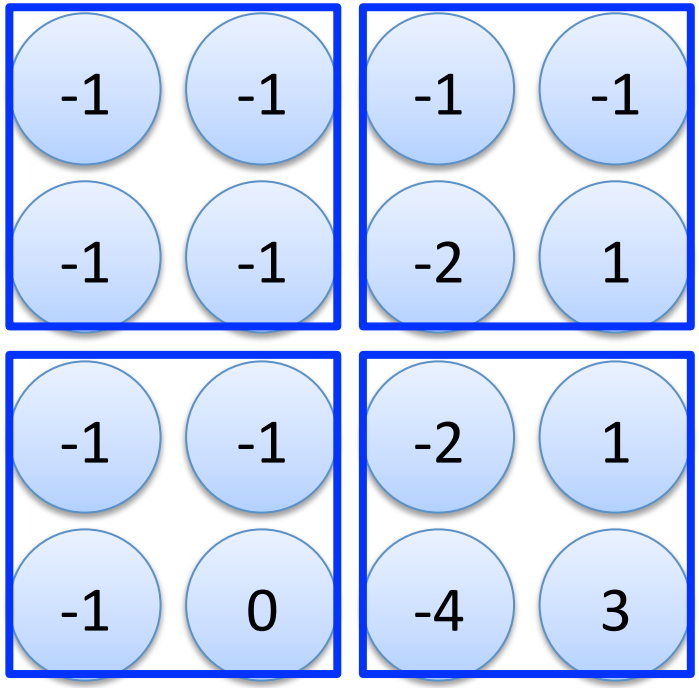
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

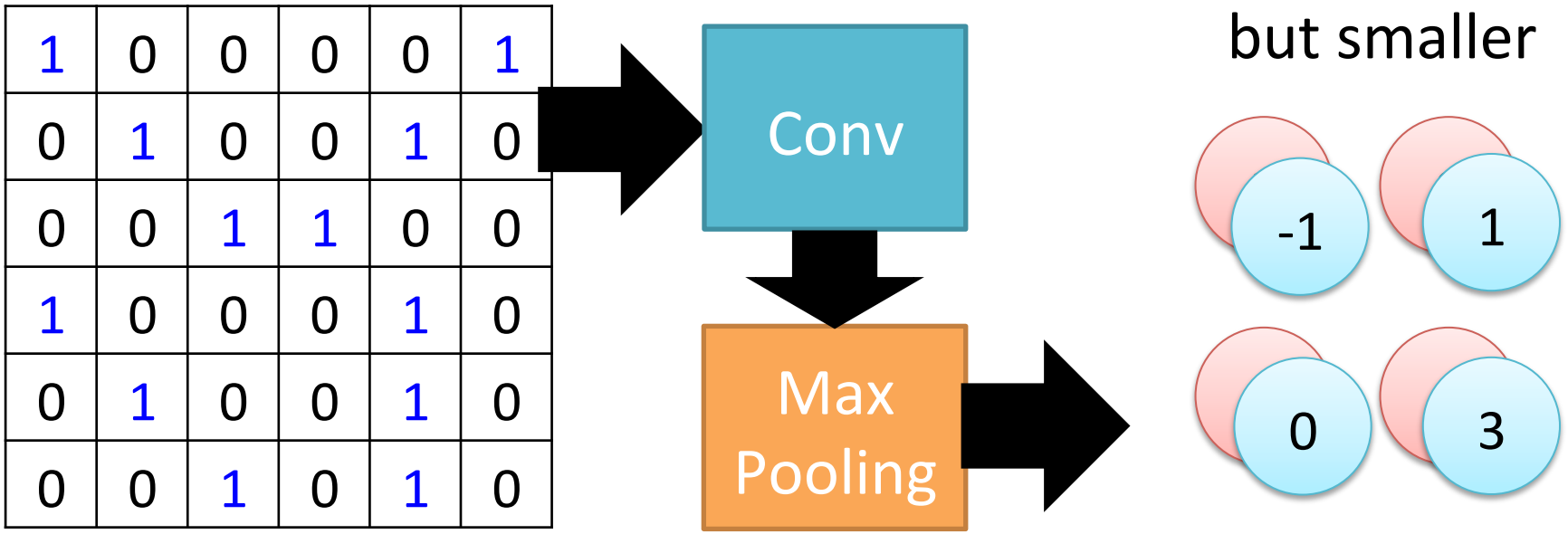
-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

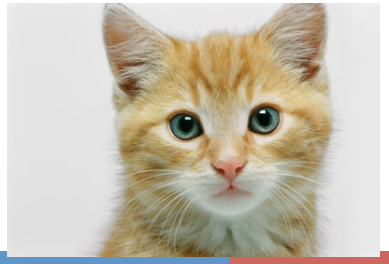


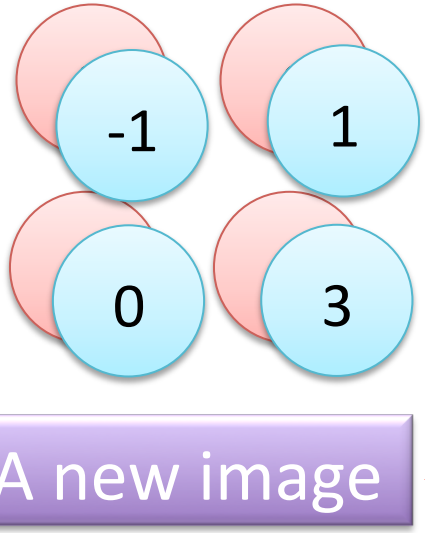


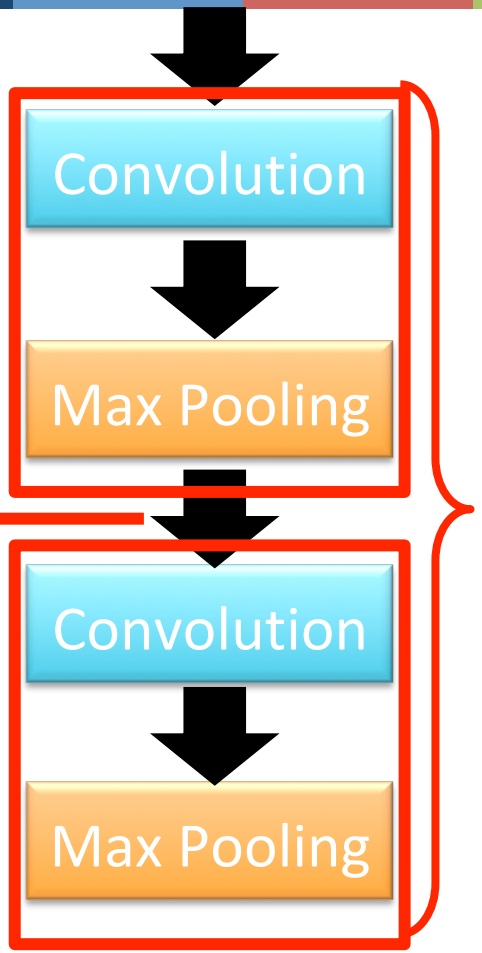
CNN – Max Pooling



The whole CNN



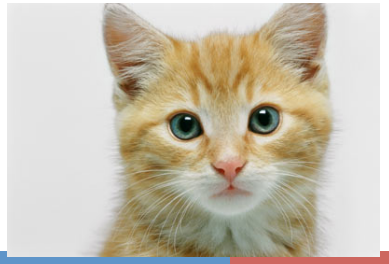




Smaller than the original image

The number of the channel is the number of filters

The whole CNN



cat dog ……

Convolution

Max Pooling

A new image

Convolution

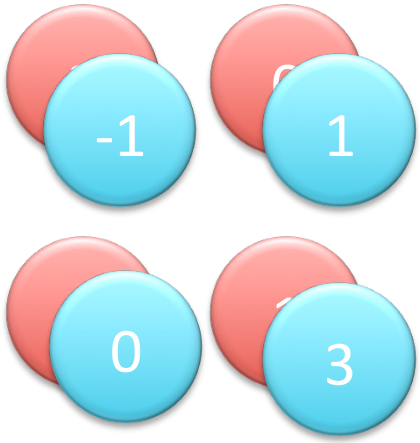
Max Pooling

A new image

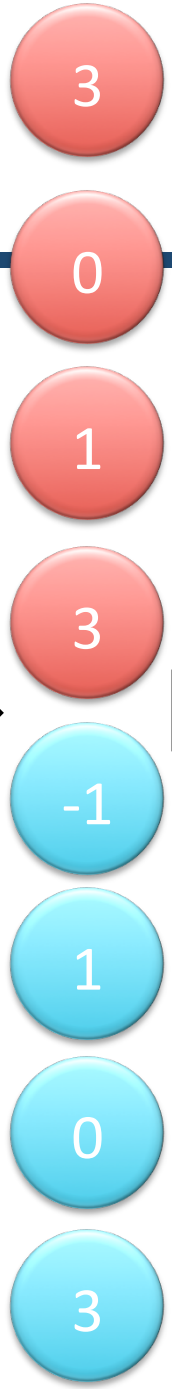
Flatten

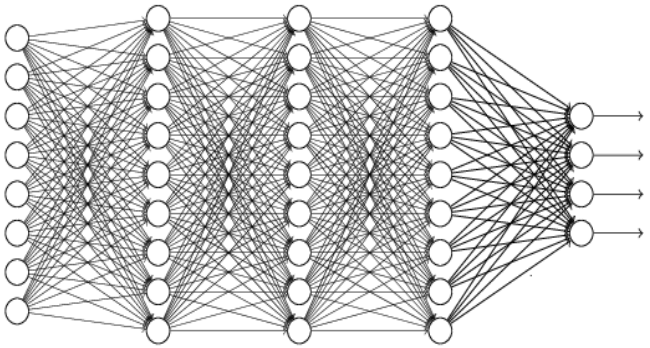
Fully Connected Feedforward network

Flatten



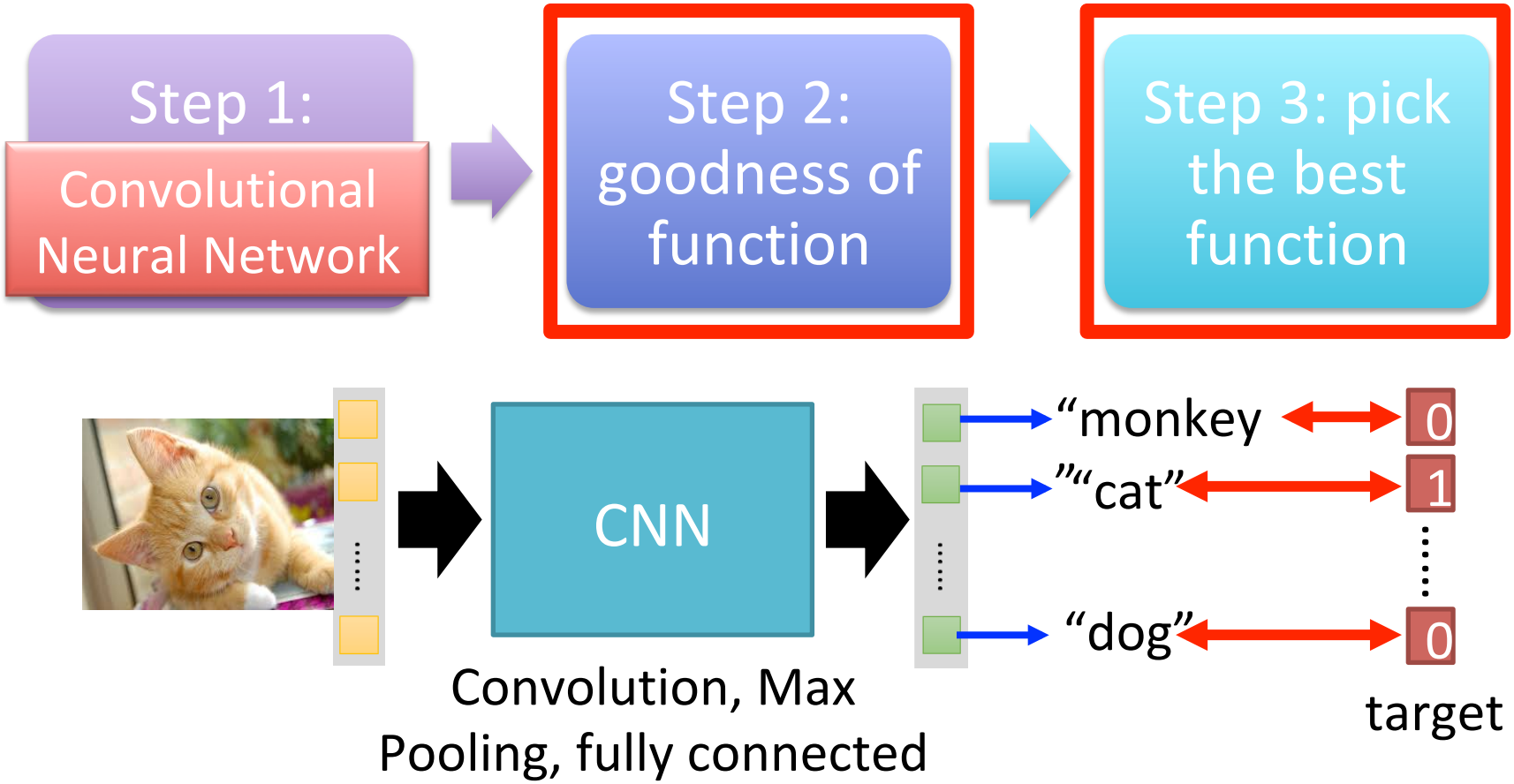
Flatten





Fully Connected Feedforward network

Convolutional Neural Network

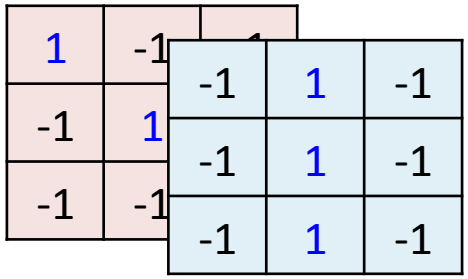


Learning: Nothing special, just gradient descent ……

CNN in Keras

Only modified the ***network structure*** and ***input format (vector -> 3-D tensor)***

```
model2.add( Convolution2D( 25,3,3,
            input_shape=(1,28,28) ) )
```



.....

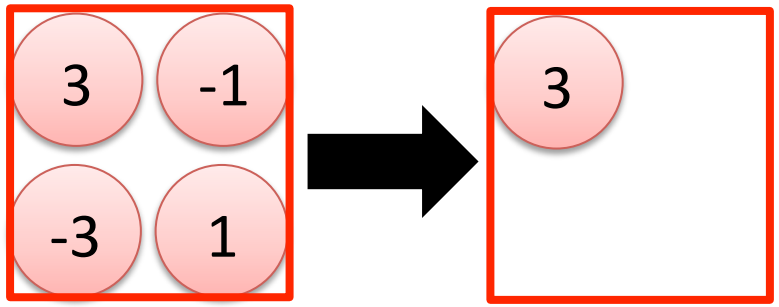
There are 25 3x3 filters.

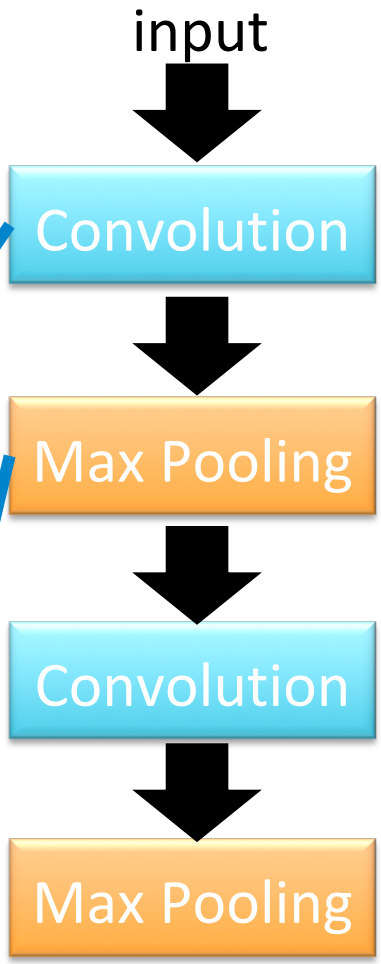
Input_shape = (1 , 28 , 28)

1: black/weight, 3: RGB

28 x 28 pixels

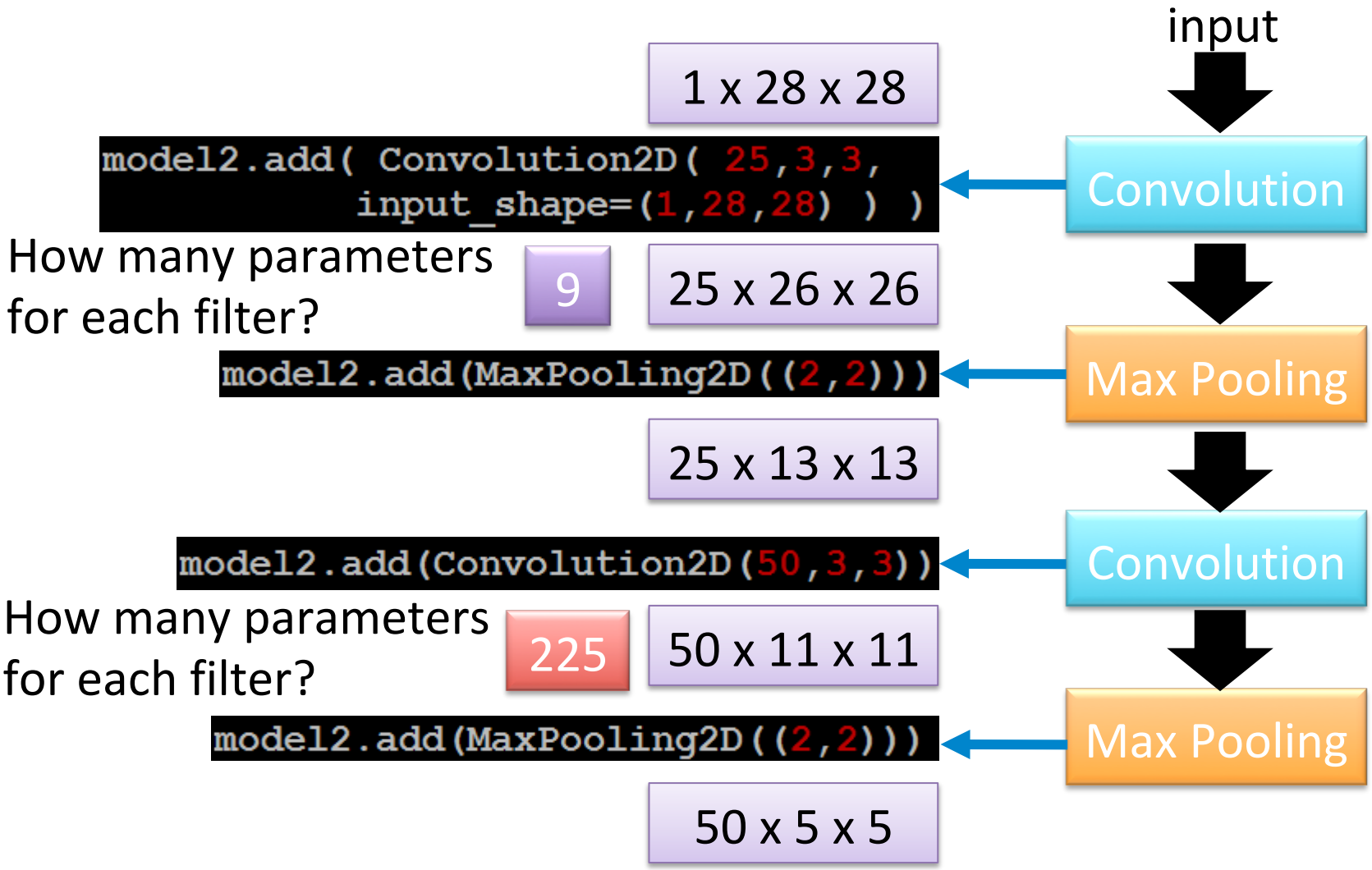
```
model2.add(MaxPooling2D((2,2)))
```





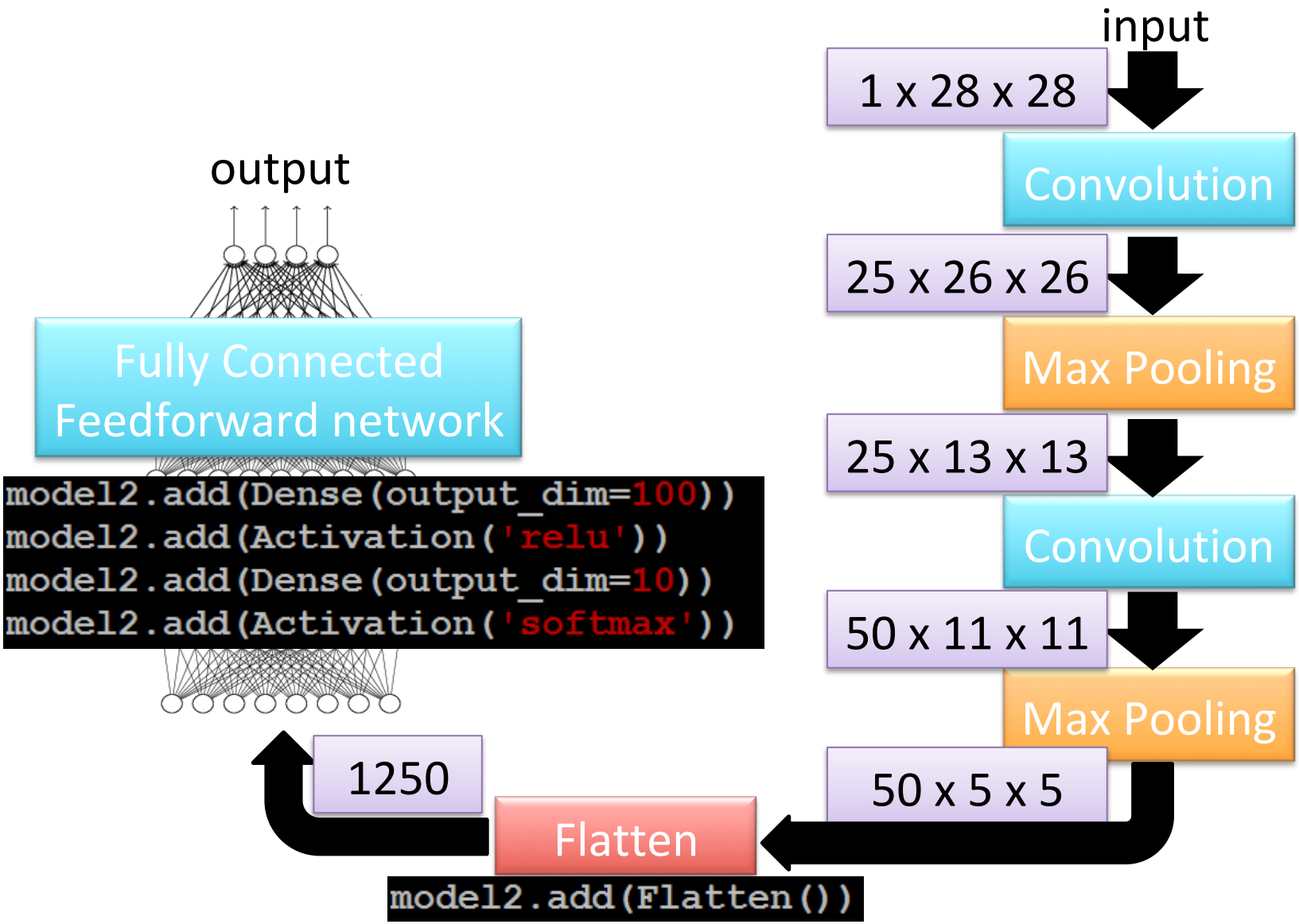
CNN in Keras

Only modified the ***network structure*** and ***input format (vector -> 3-D tensor)***



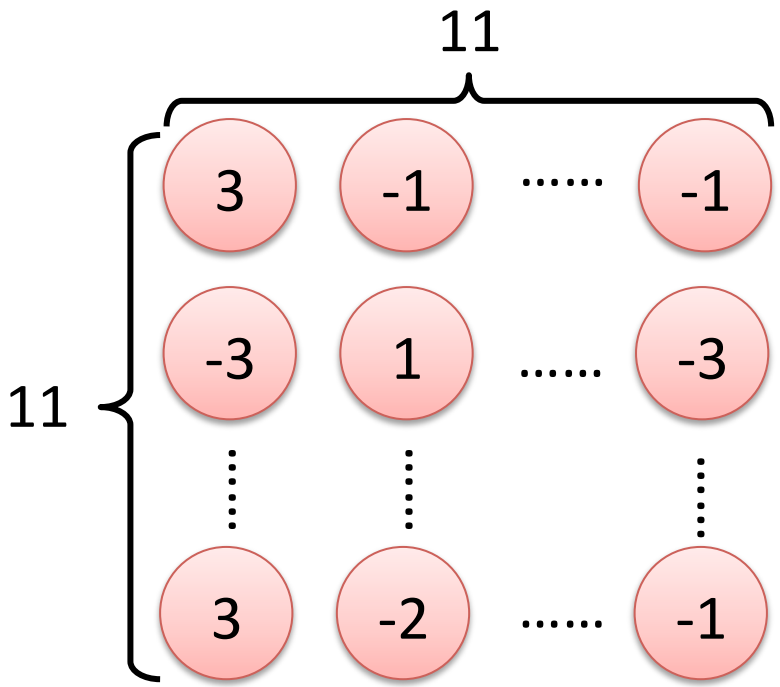
CNN in Keras

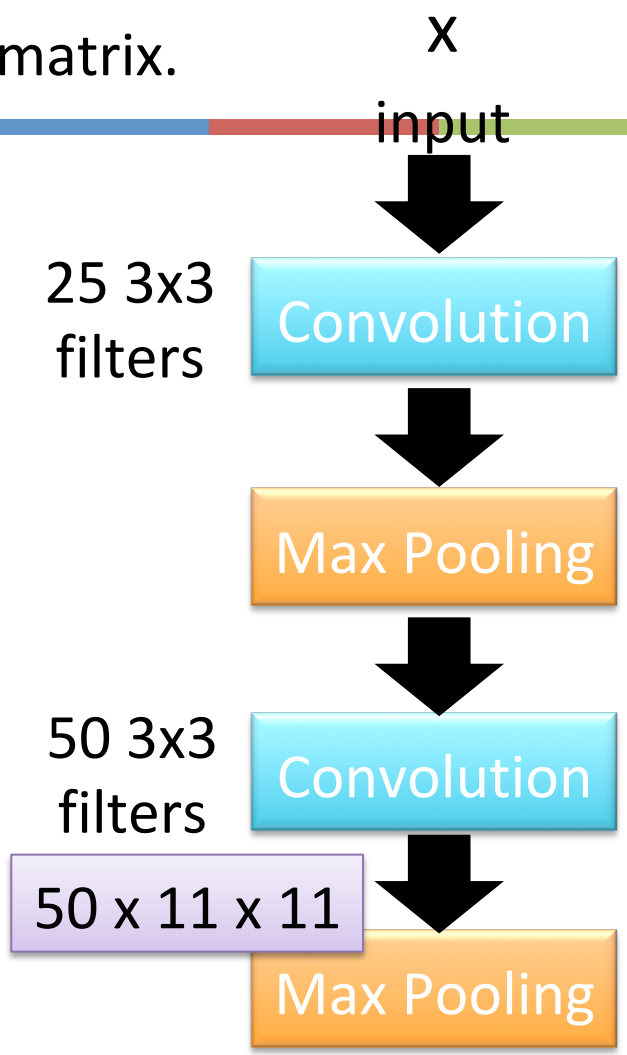
Only modified the ***network structure*** and ***input format (vector -> 3-D tensor)***



What does CNN learn?

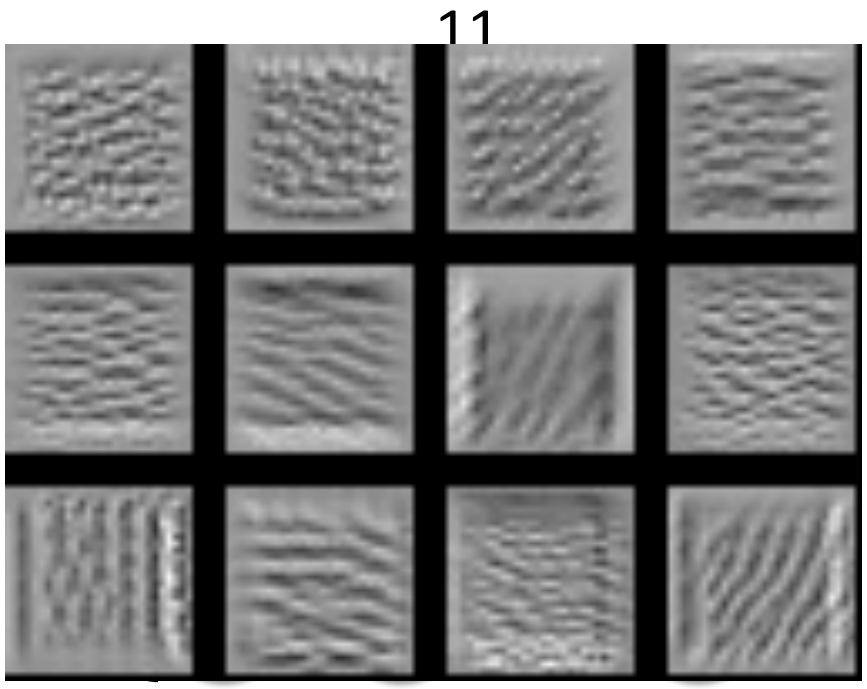
The output of the k-th filter is a 11 x 11 matrix.

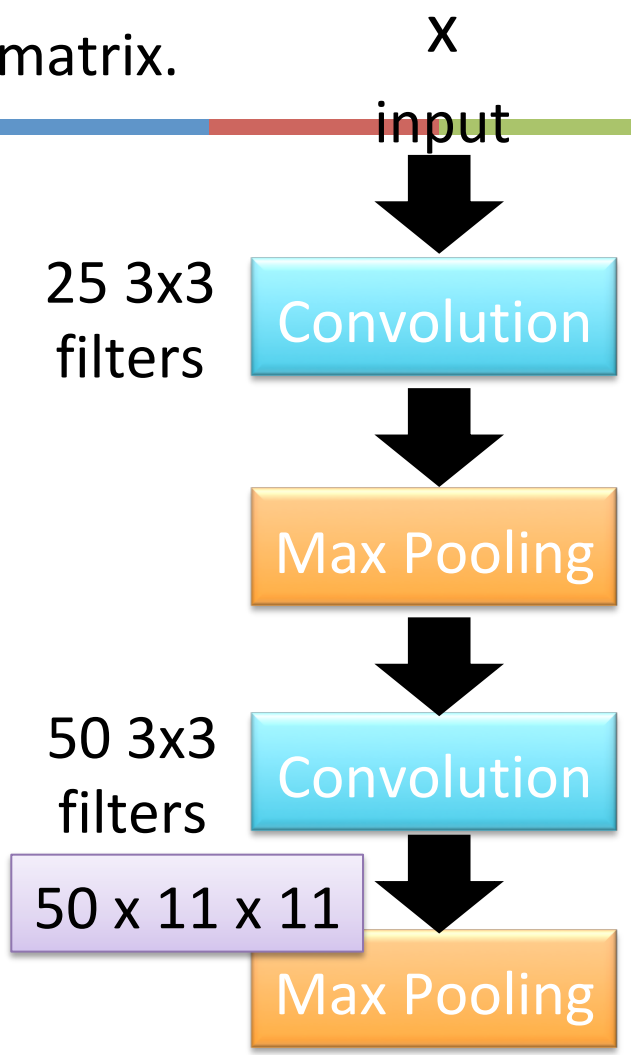




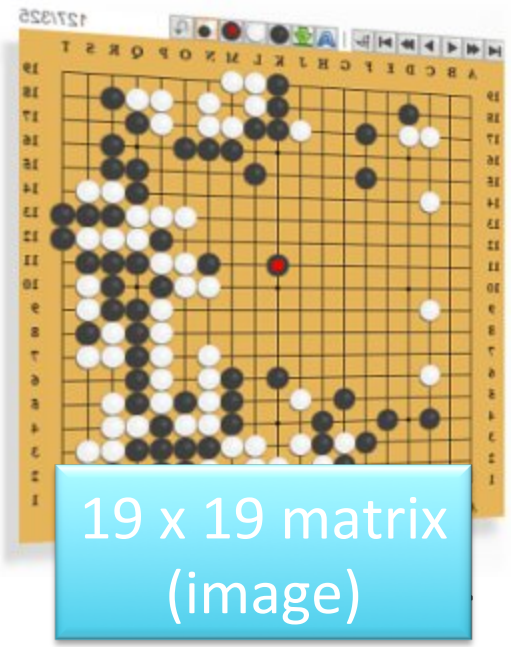
What does CNN learn?

The output of the k-th filter is a 11 x 11 matrix.





More Application: Playing Go



19 x 19 matrix (image)

Network

Next move (19 x 19 positions)

19 x 19 vector

Black: 1

white: -1

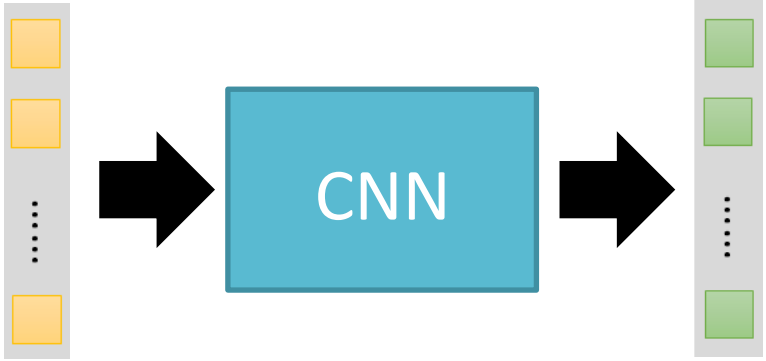
none: 0

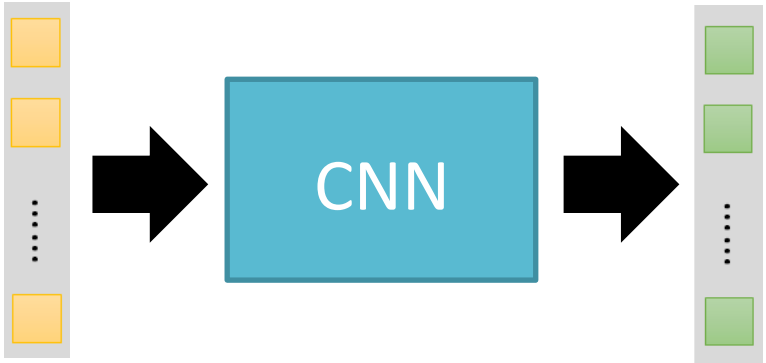
Fully-connected feedforward network can be used

But CNN performs much better.

More Application: Playing Go

Training: record of previous plays 黑: 5之五→白: 天元→黑: 五之5 …





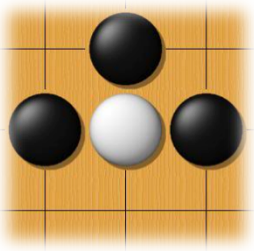
Target:
“天元” = 1
else = 0

Target:
“五之 5” = 1
else = 0

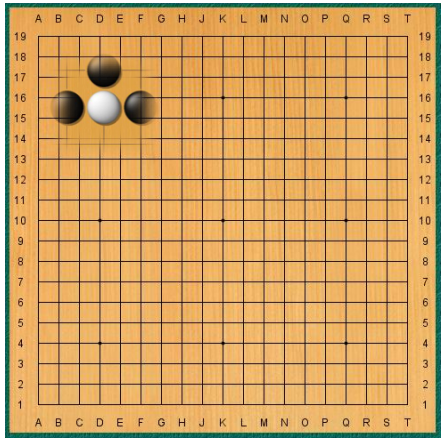
Why CNN for playing Go?

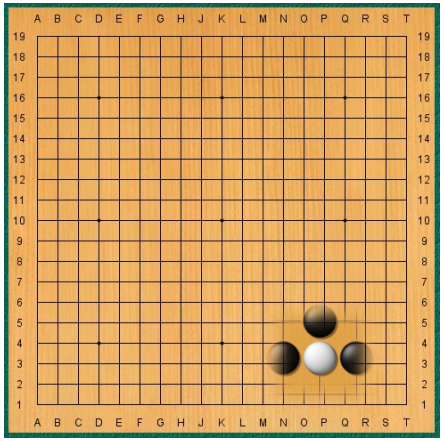
- Some patterns are much smaller than the whole image

  Alpha Go uses 5 x 5 for first layer



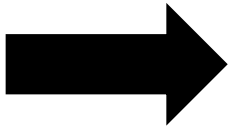
- The same patterns appear in different regions.





Why CNN for playing Go?

- Subsampling the pixels will not change the object



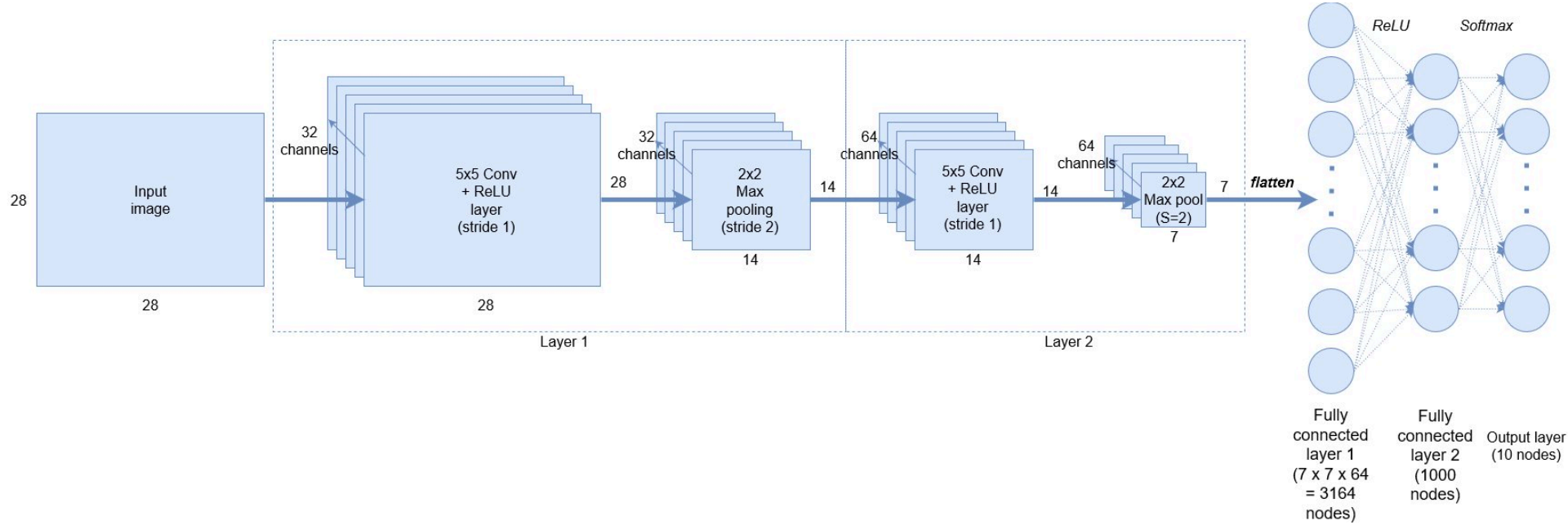
Max Pooling

How to explain this???

Neural network architecture. The input to the policy network is a $19 \times 19 \times 48$ image stack consisting of 48 feature planes. The first hidden layer zero pads the input into a 23×23 image, then convolves k filters of kernel size 5×5 with stride 1 with the input image and applies a rectifier nonlinearity. Each of the subsequent hidden layers 2 to 12 zero pads the respective previous hidden layer into a 21×21 image, then convolves k filters of kernel size 3×3 with stride 1, again followed by a rectifier nonlinearity. The final layer convolves 1 filter of kernel size 1×1 with stride 1, with a different bias for each position, and applies a softmax function. The Extended Data Table 3 additionally show the results of training with $k = 128$, 256 and 384 filters.

Alpha Go does not use Max Pooling .....

Example


28
Input image
28
32 channels
5x5 Conv + ReLU layer (stride 1)
28
Layer 1
32 channels
28
2x2 Max pooling (stride 2)
14
64 channels
14
5x5 Conv + ReLU layer (stride 1)
14
Layer 2
64 channels
14
2x2 Max pool (S=2)
7
7
flatten
ReLU
Softmax
Fully connected layer 1 (7 x 7 x 64 = 3164 nodes)
Fully connected layer 2 (1000 nodes)
Output layer (10 nodes)

```
model = Sequential()

model.add(Conv2D(32, kernel_size=(5, 5), strides=(1, 1), activation='relu’,
                 input_shape = (28,28,1) ))

model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

model.add(Conv2D(64, (5, 5), activation='relu'))

model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())

model.add(Dense(1000, activation='relu'))

model.add(Dense(10, activation='softmax'))
```

```
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.SGD(lr=0.01),
              metrics=['accuracy'])
```

```
model.fit(x_train, y_train,
          batch_size=64,
          epochs=10,
          verbose=1,
          validation_data=(x_test, y_test)
         )
```

```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

Variants of Neural Networks

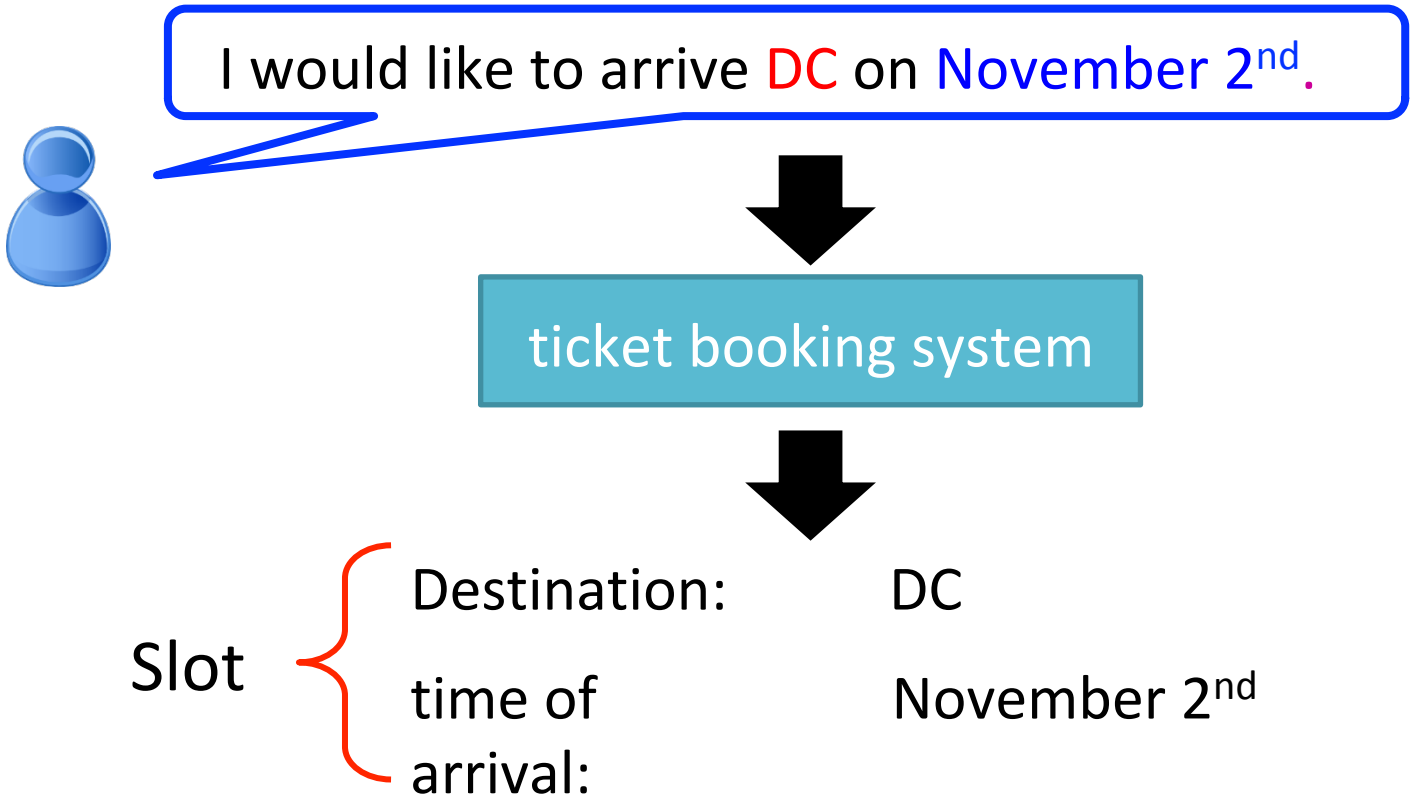
Convolutional Neural Network (CNN)

Recurrent Neural Network (RNN)

Neural Network with Memory

Example Application

- Slot Filling

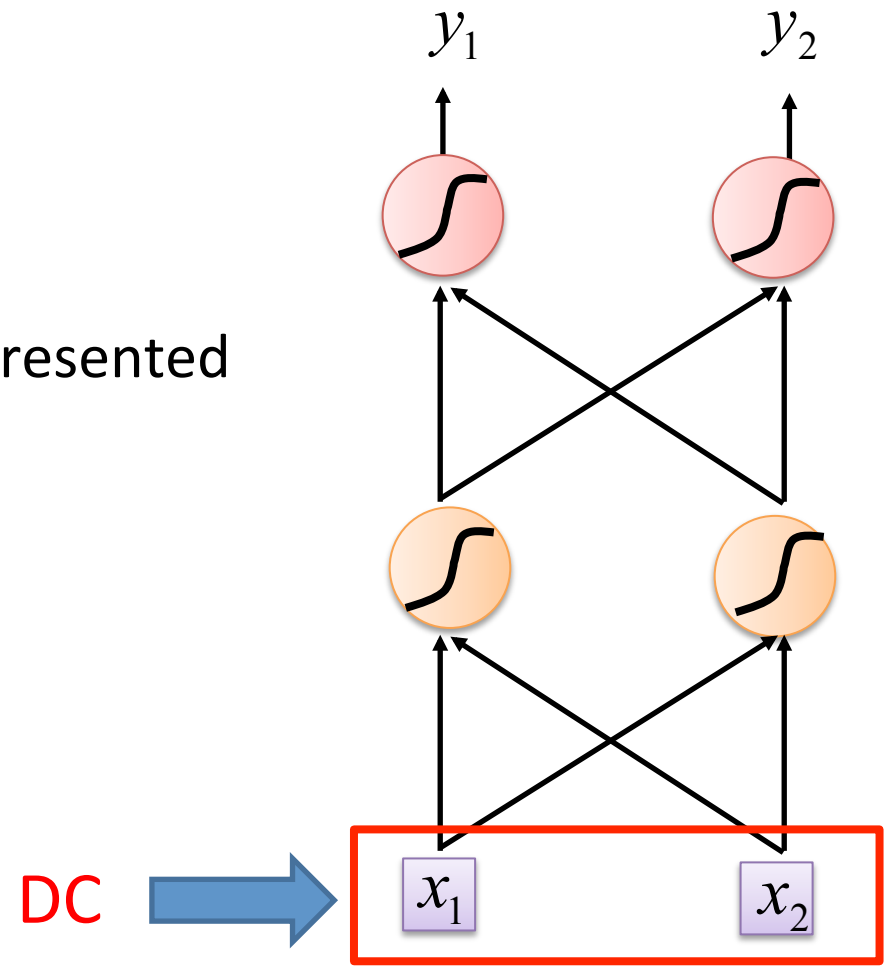


Example Application

Solving slot filling by Feedforward network?

Input: a word

(Each word is represented as a vector)



1-of-N encoding

How to represent each word as a vector?

1-of-N Encoding lexicon = {apple, bag, cat, dog, elephant}

The vector is lexicon size.

Each dimension corresponds to a word in the lexicon

The dimension for the word is 1, and others are 0

apple = [1 0 0 0 0]

bag = [0 1 0 0 0]

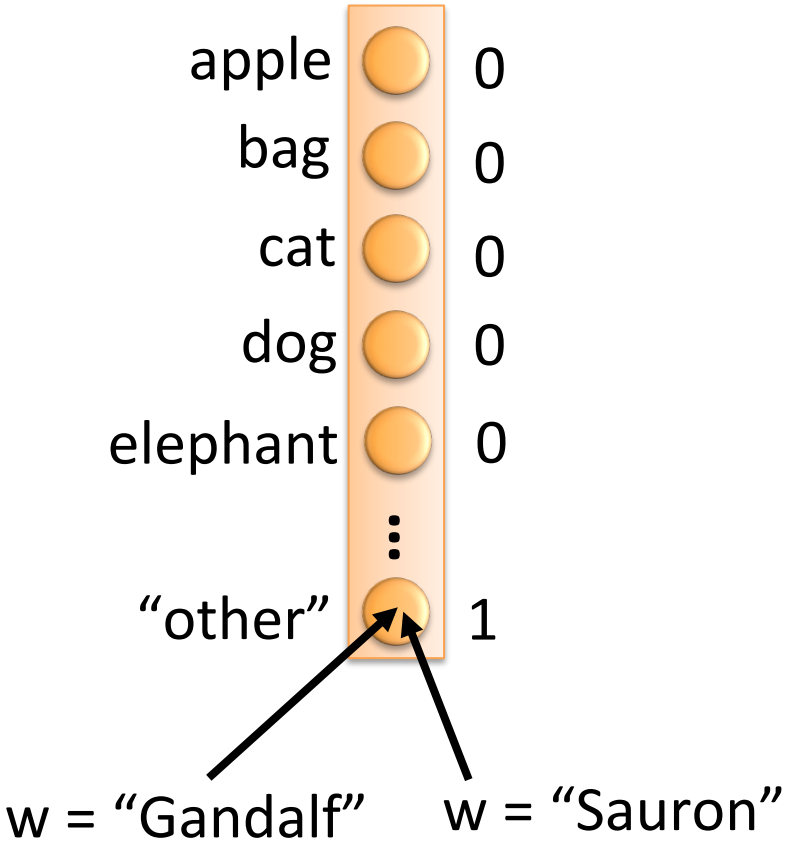
cat = [0 0 1 0 0]

dog = [0 0 0 1 0]

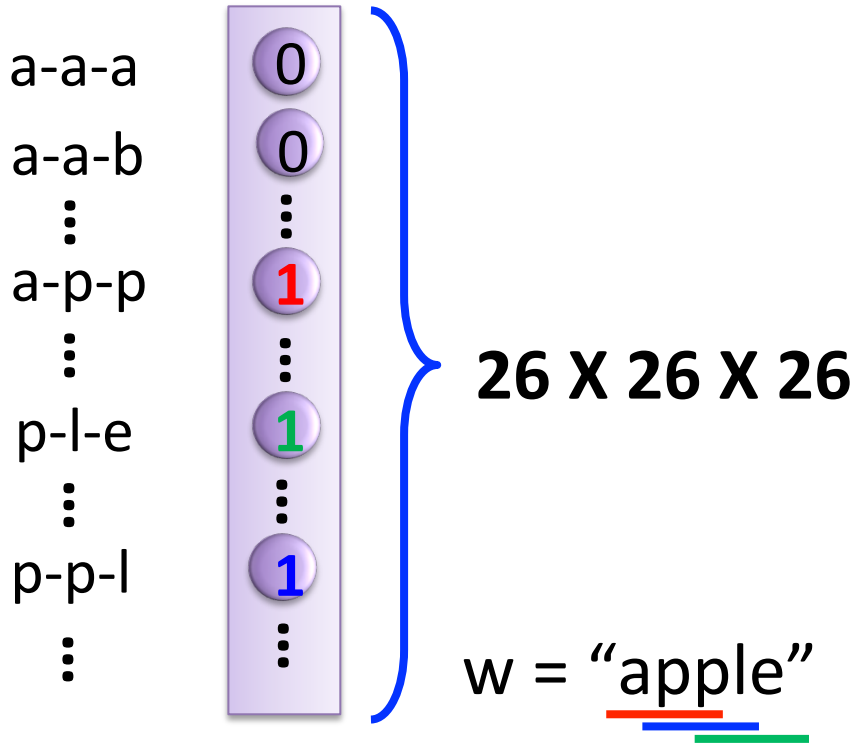
elephant = [0 0 0 0 1]

Beyond 1-of-N encoding

Dimension for "Other"



Word hashing



Example Application

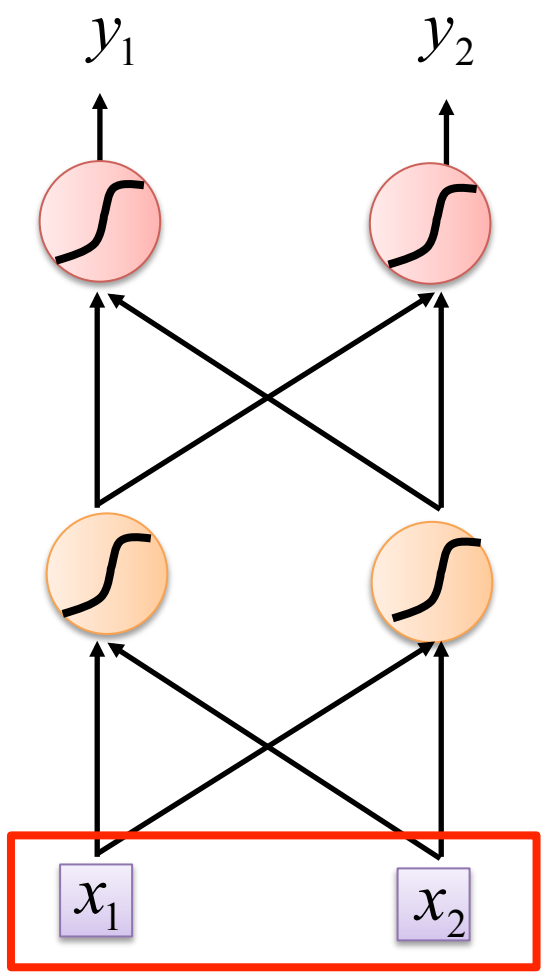
Solving slot filling by Feedforward network?

Input: a word

(Each word is represented as a vector)

Output:

Probability distribution that the input word belonging to the slots



Example Application

arrive DC on November 2nd

other dest other time time

dest time of departure

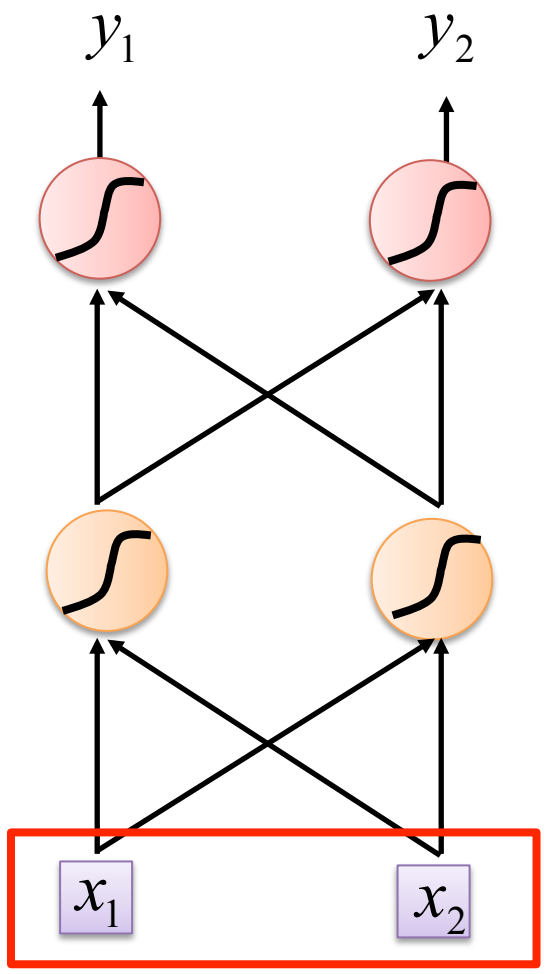
Problem?

leave DC on November 2nd

place of departure

Neural network needs memory!

DC



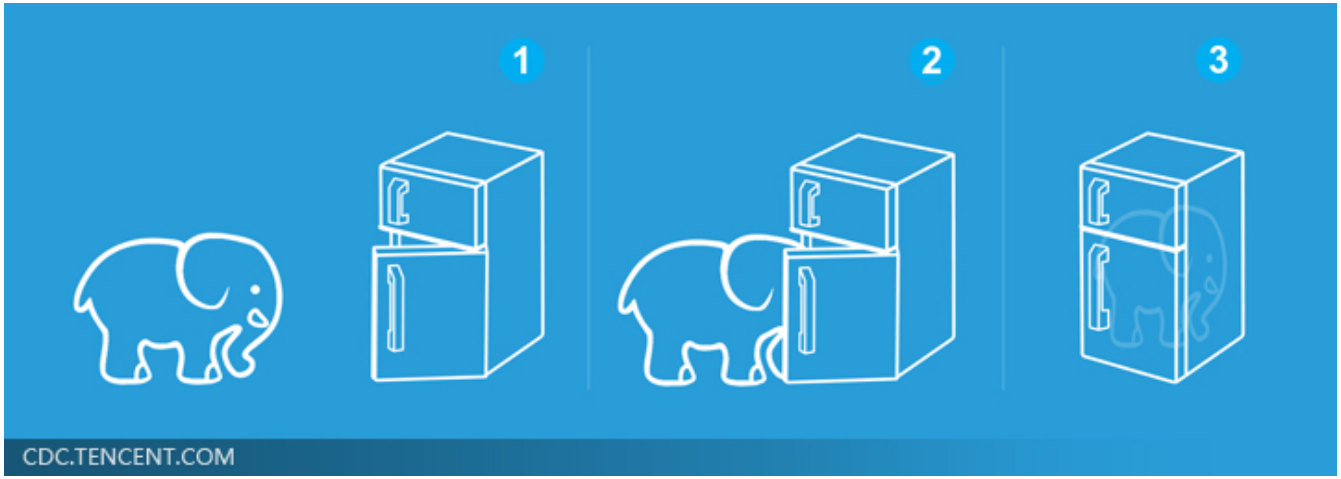
Three Steps for Deep Learning

Step 1: Recurrent Neural Network

Step 2: goodness of function

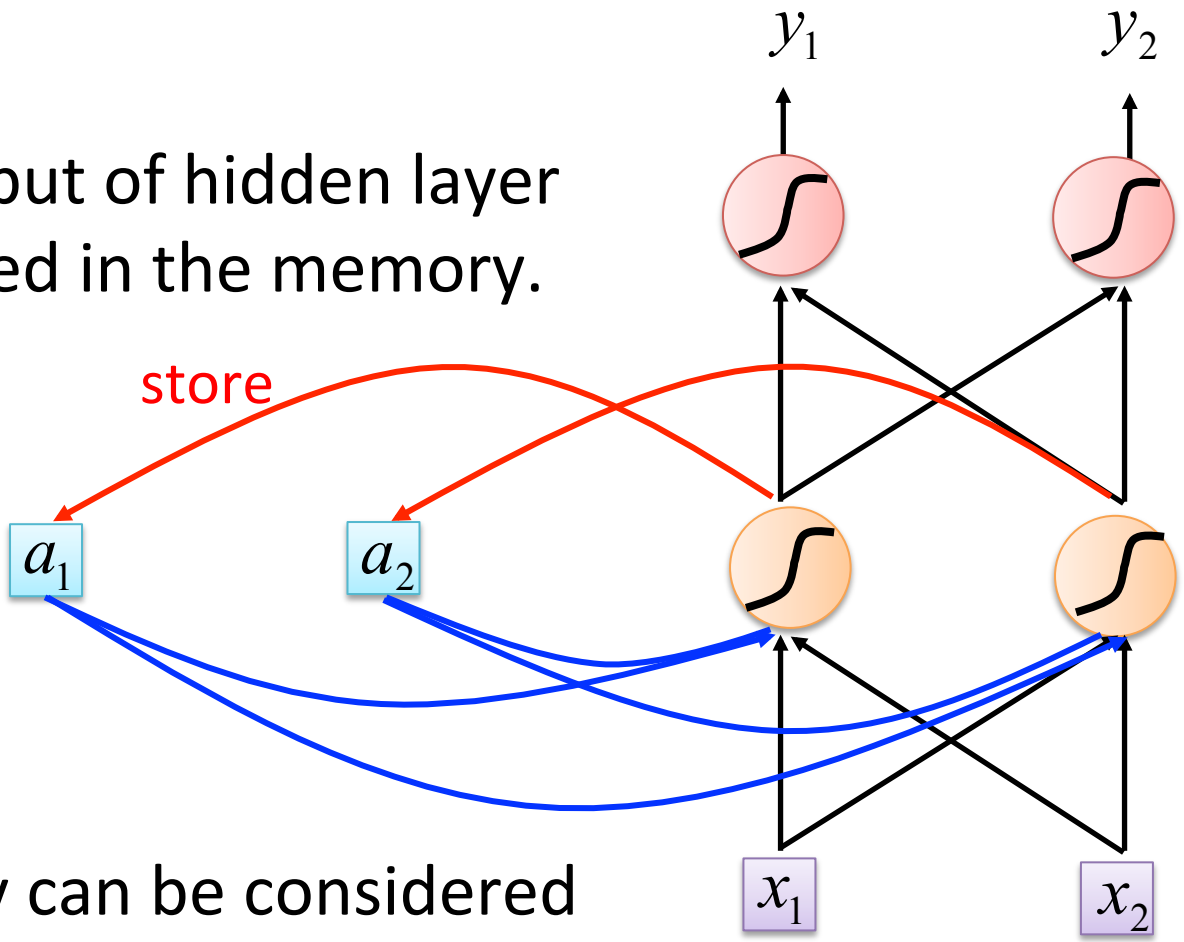
Step 3: pick the best function

Deep Learning is so simple ……



Recurrent Neural Network (RNN)

The output of hidden layer are stored in the memory.



Memory can be considered as another input.

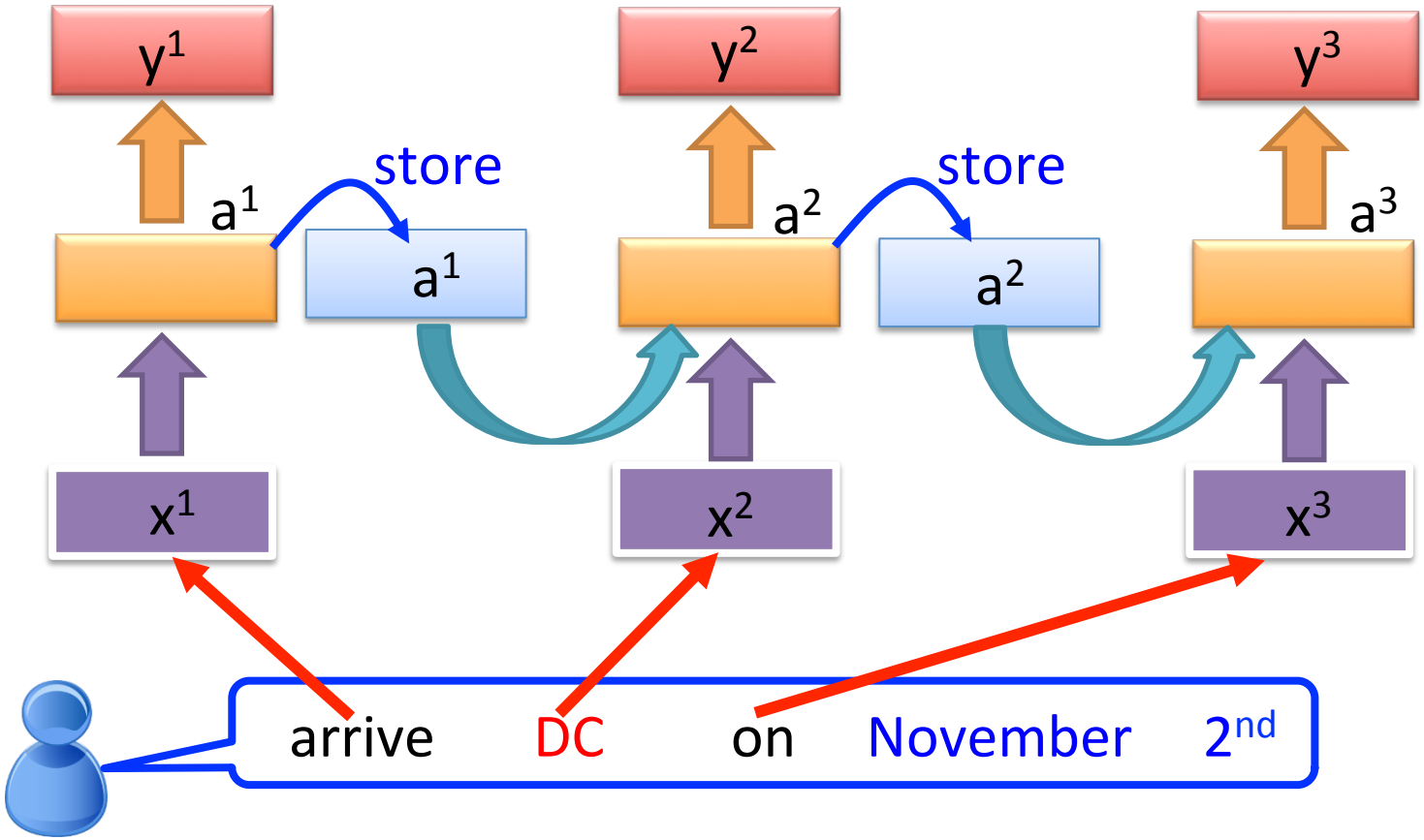
RNN

The same network is used again and again.

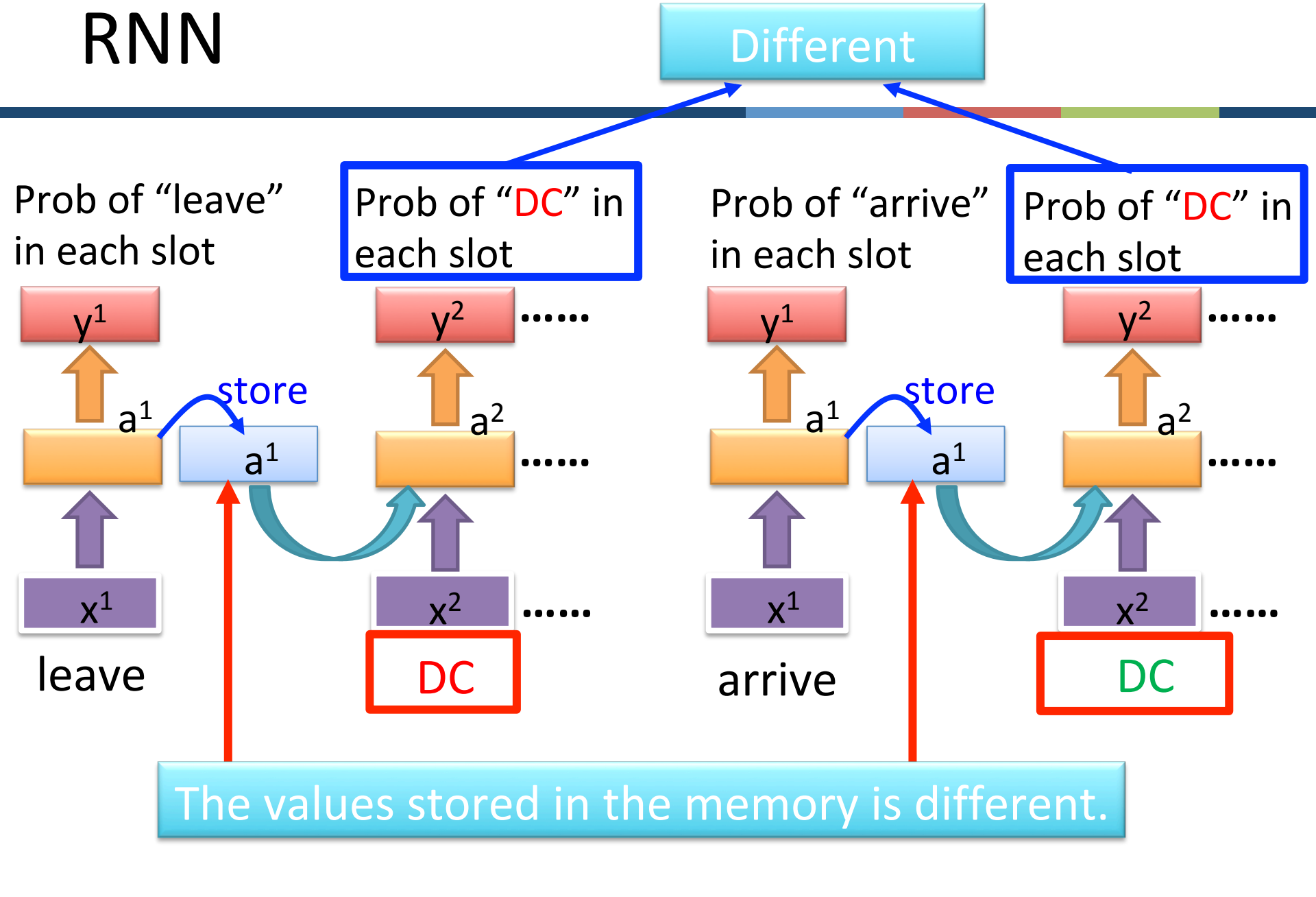
Probability of "arrive" in each slot

Probability of "DC" in each slot

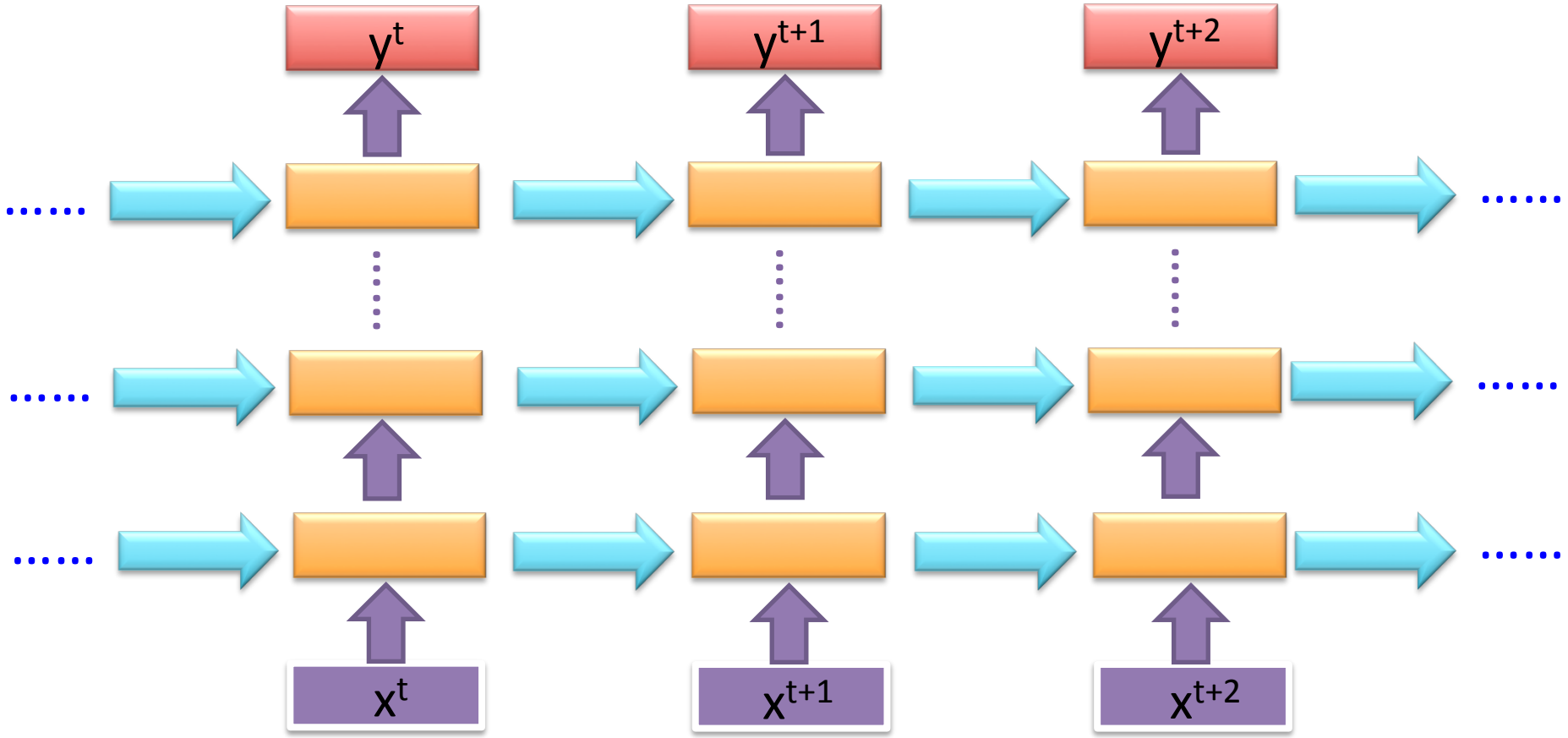
Probability of "on" in each slot



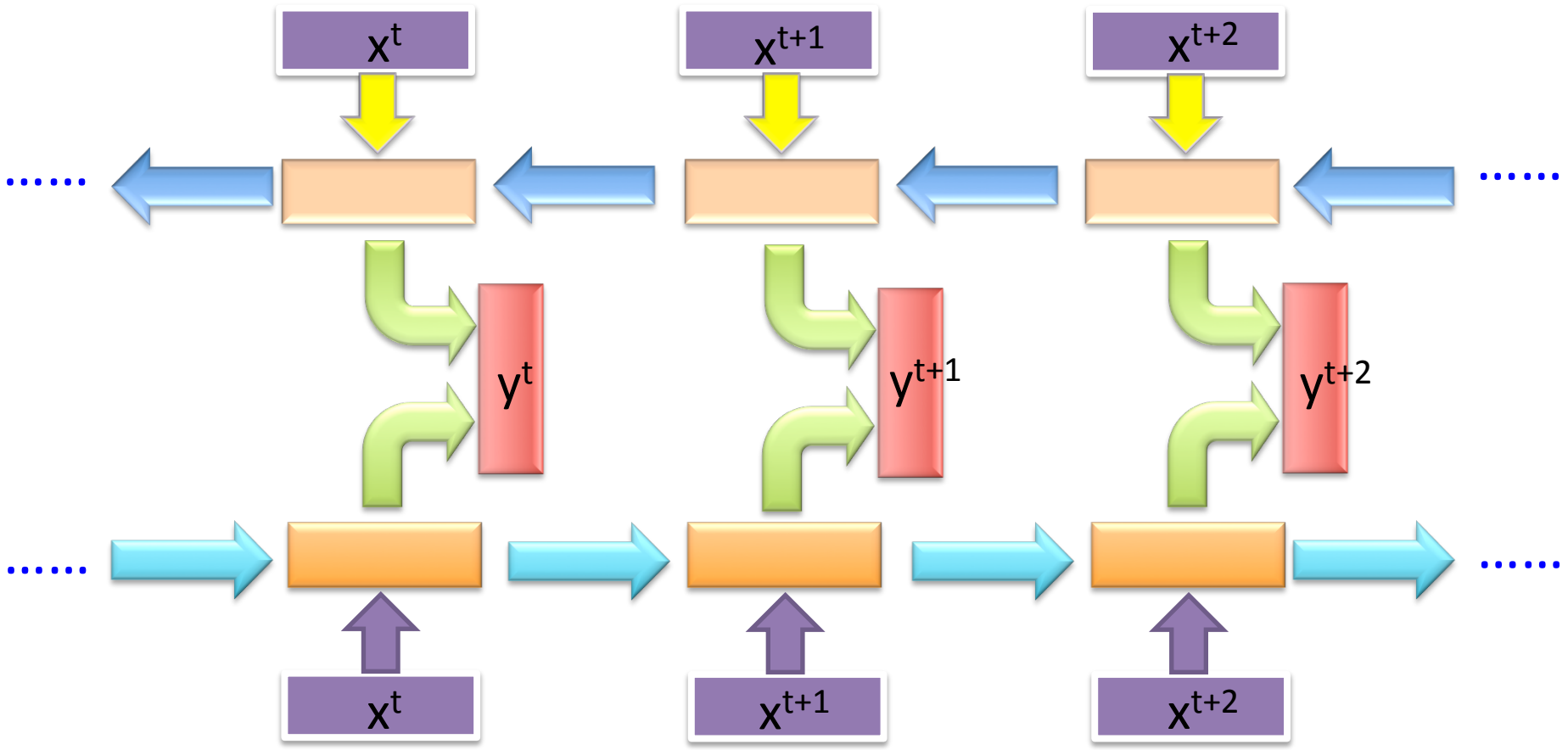
RNN


Different
Prob of "leave" in each slot
Prob of "DC" in each slot
Prob of "arrive" in each slot
Prob of "DC" in each slot
y1
y2
a1
a2
store
x1
x2
leave
DC
arrive
DC
The values stored in the memory is different.

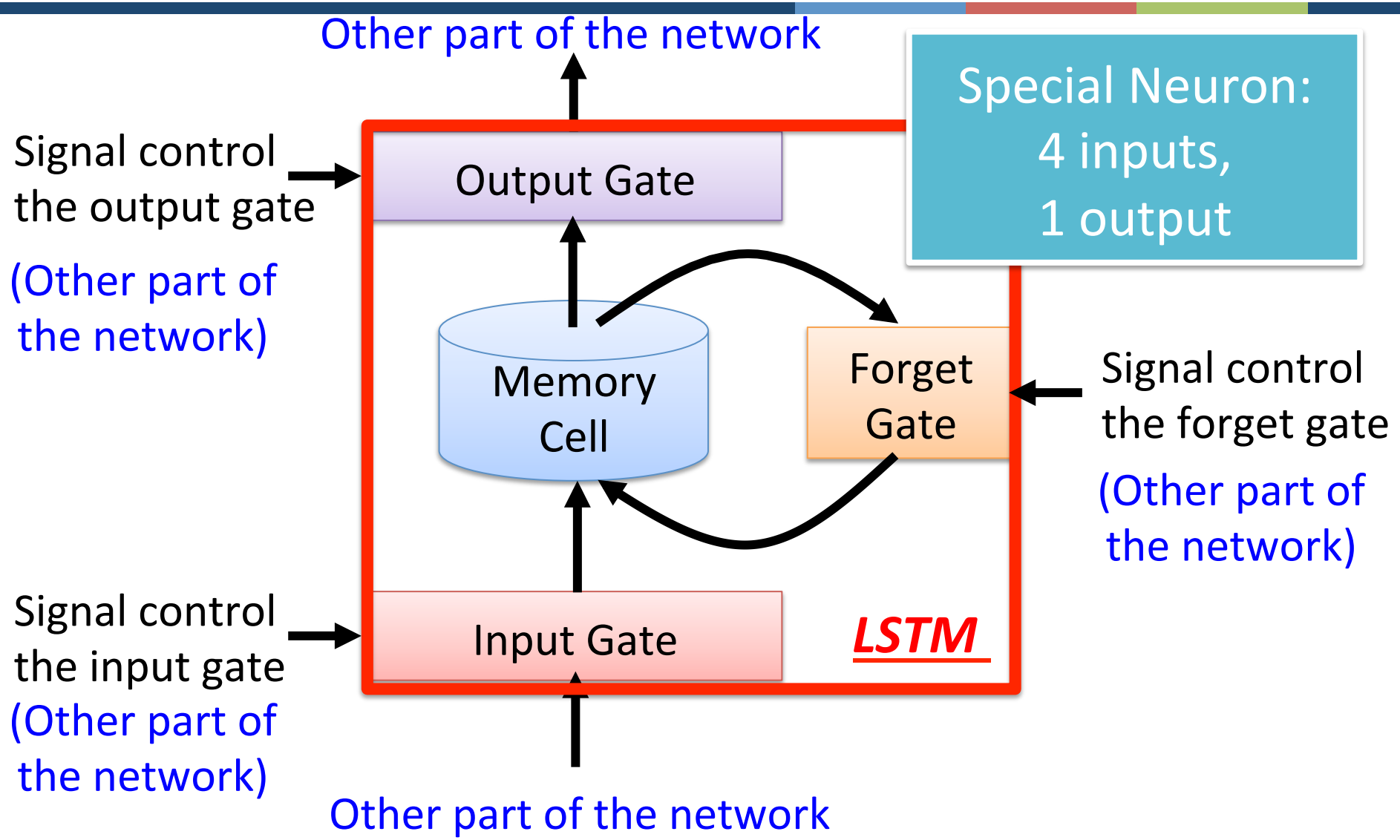
Of course it can be deep …

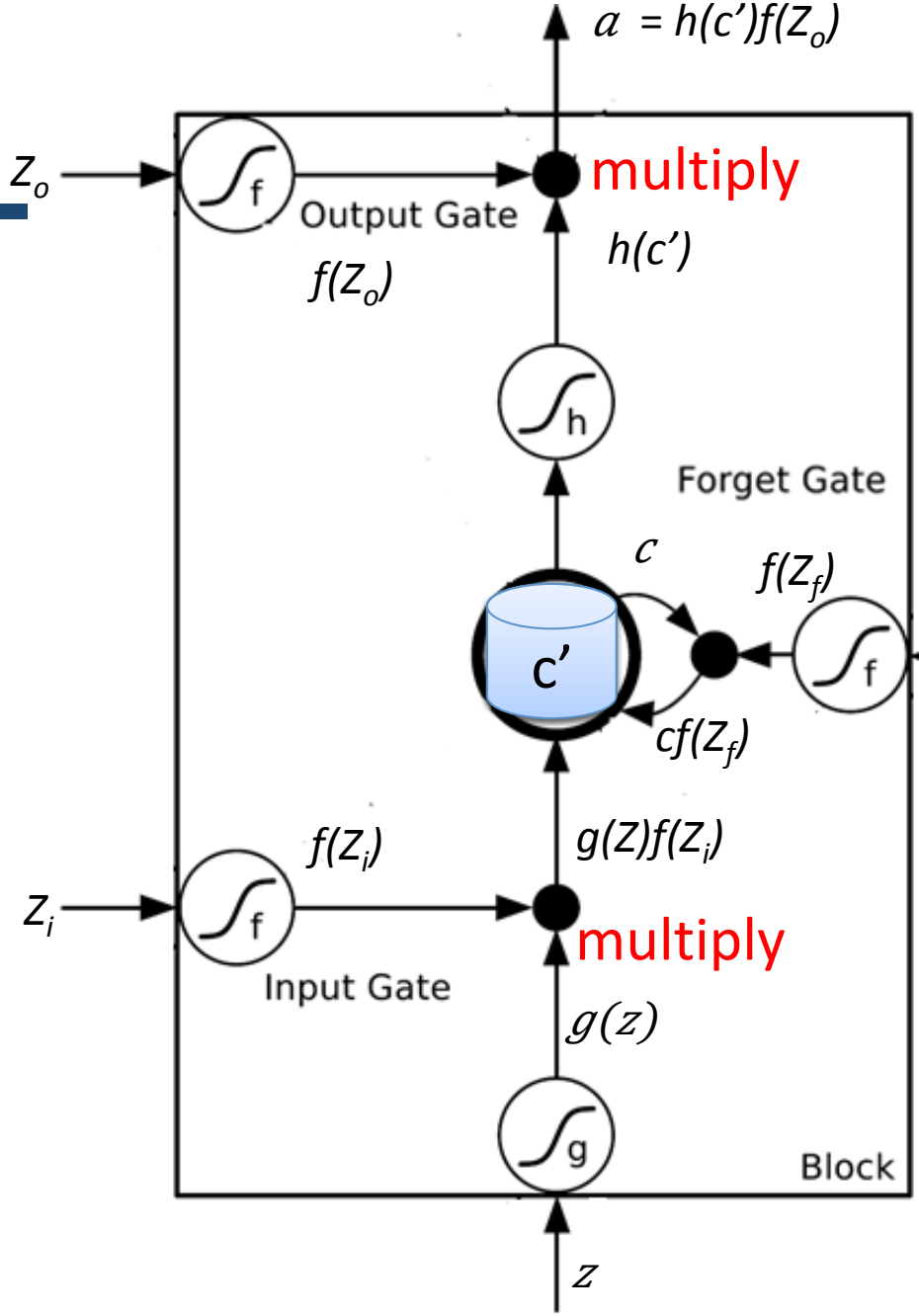


Bidirectional RNN



Long Short-term Memory (LSTM)



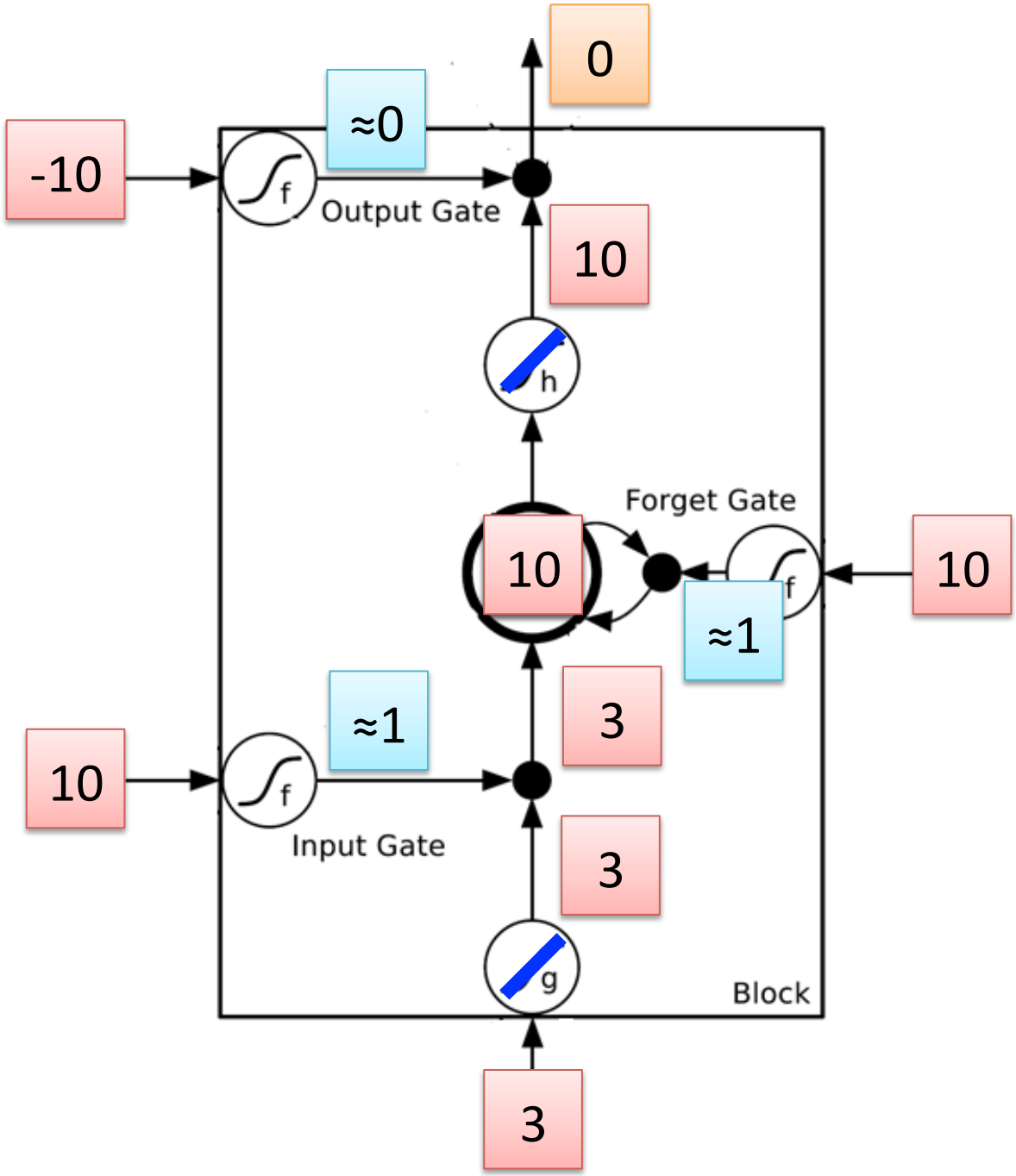


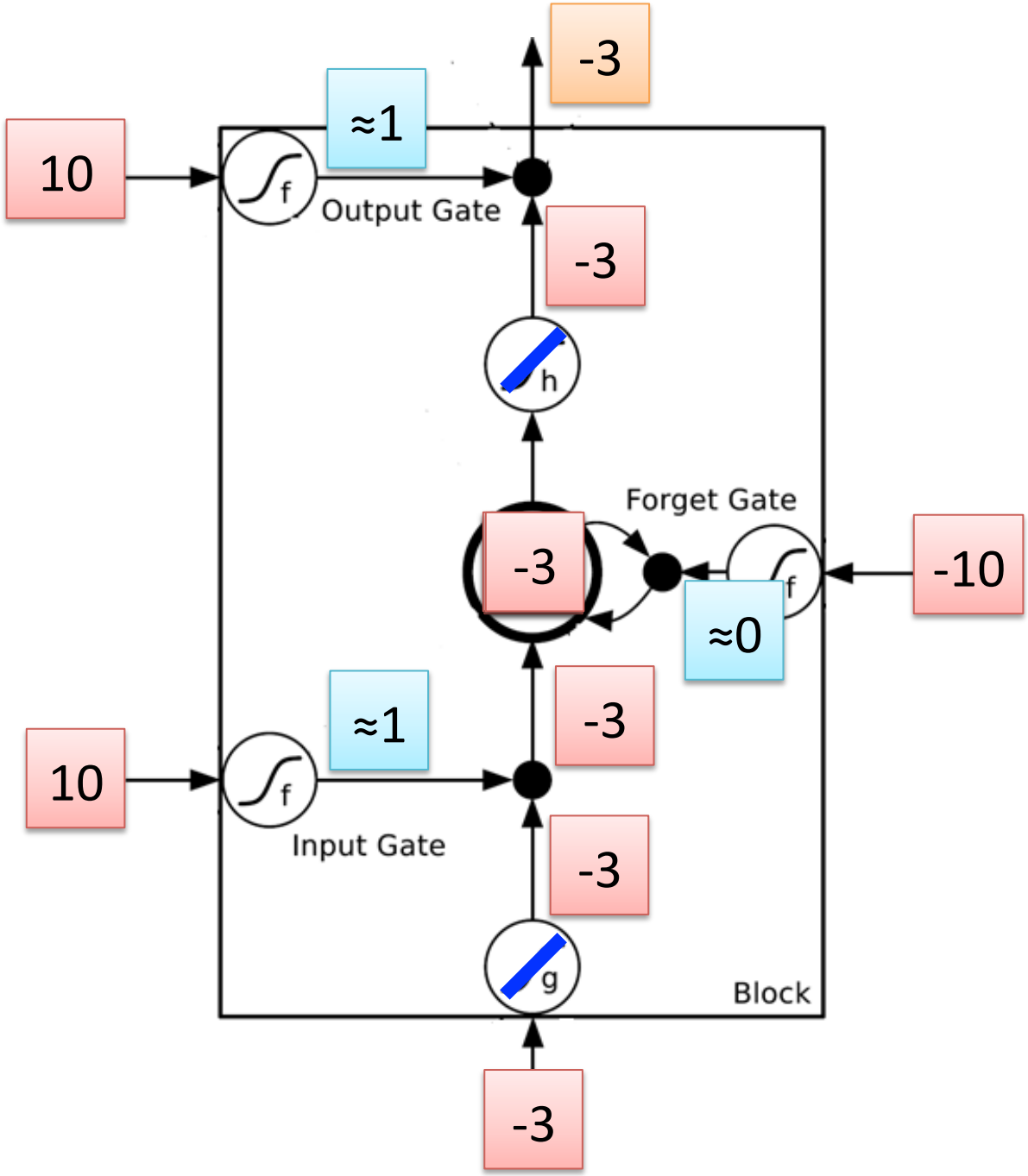
Activation function f is usually a sigmoid function

Between 0 and 1

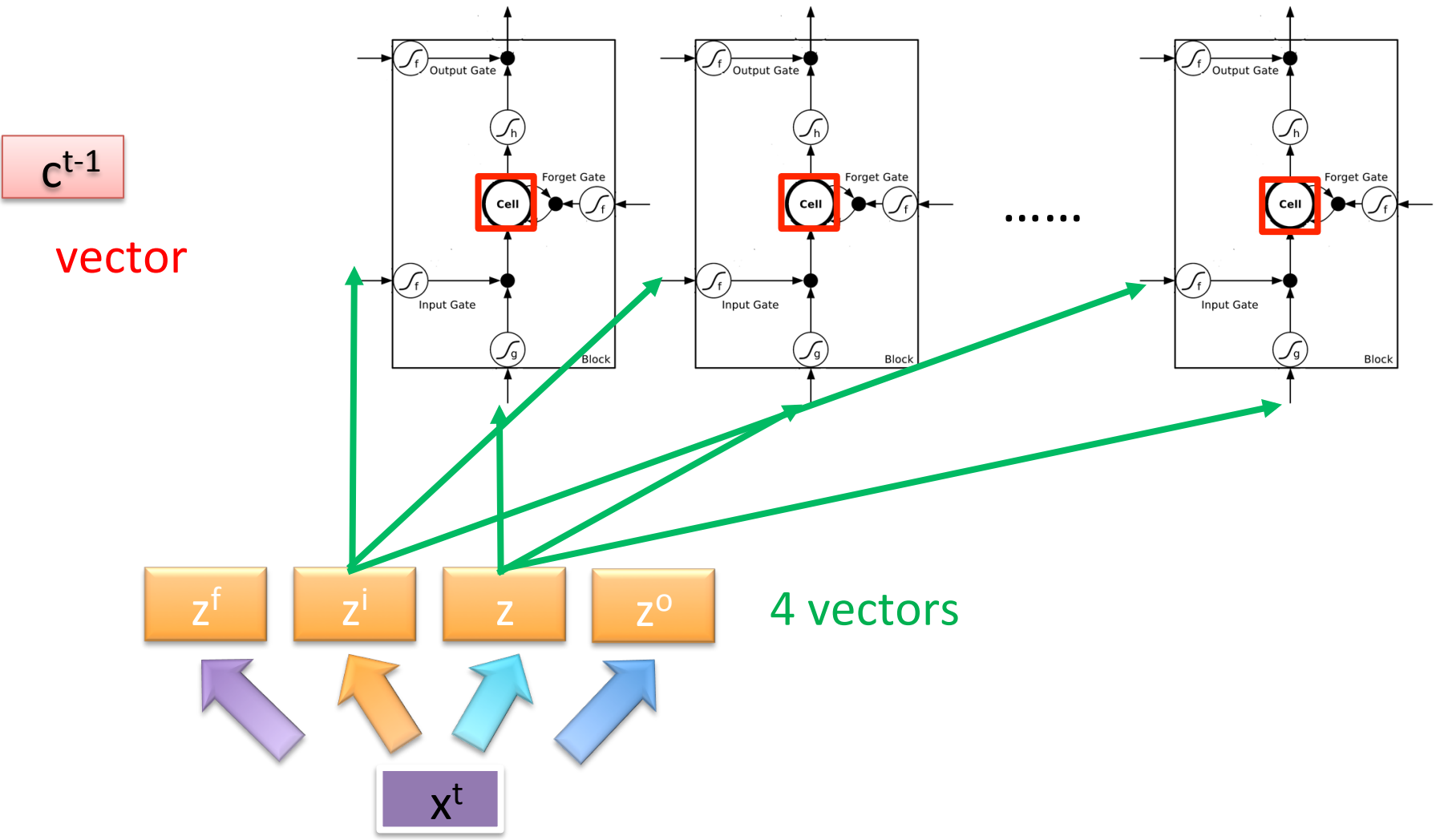
Mimic open and close gate

$c' = g(z)f(Z_i) + cf(Z_f)$

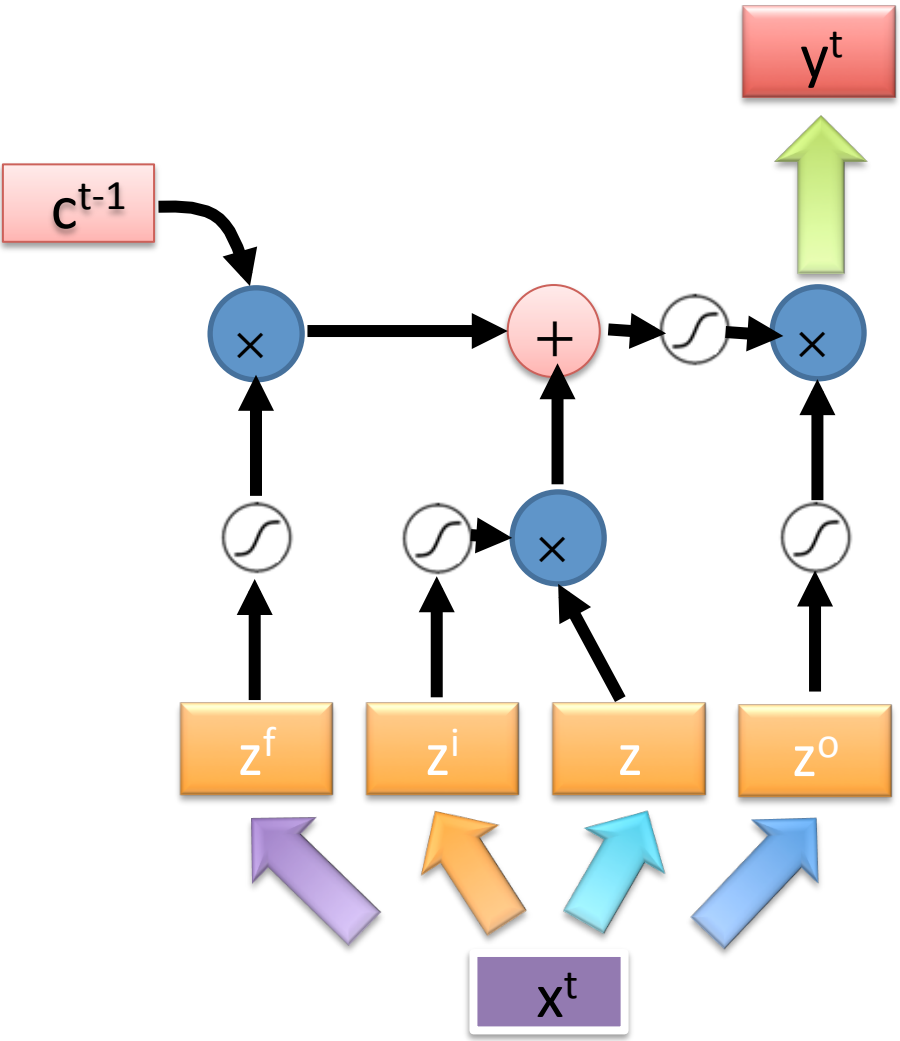

0
≈0
-10
f
Output Gate
10
h
Forget Gate
10
10
f
≈1
≈1
3
10
f
Input Gate
3
g
Block
3

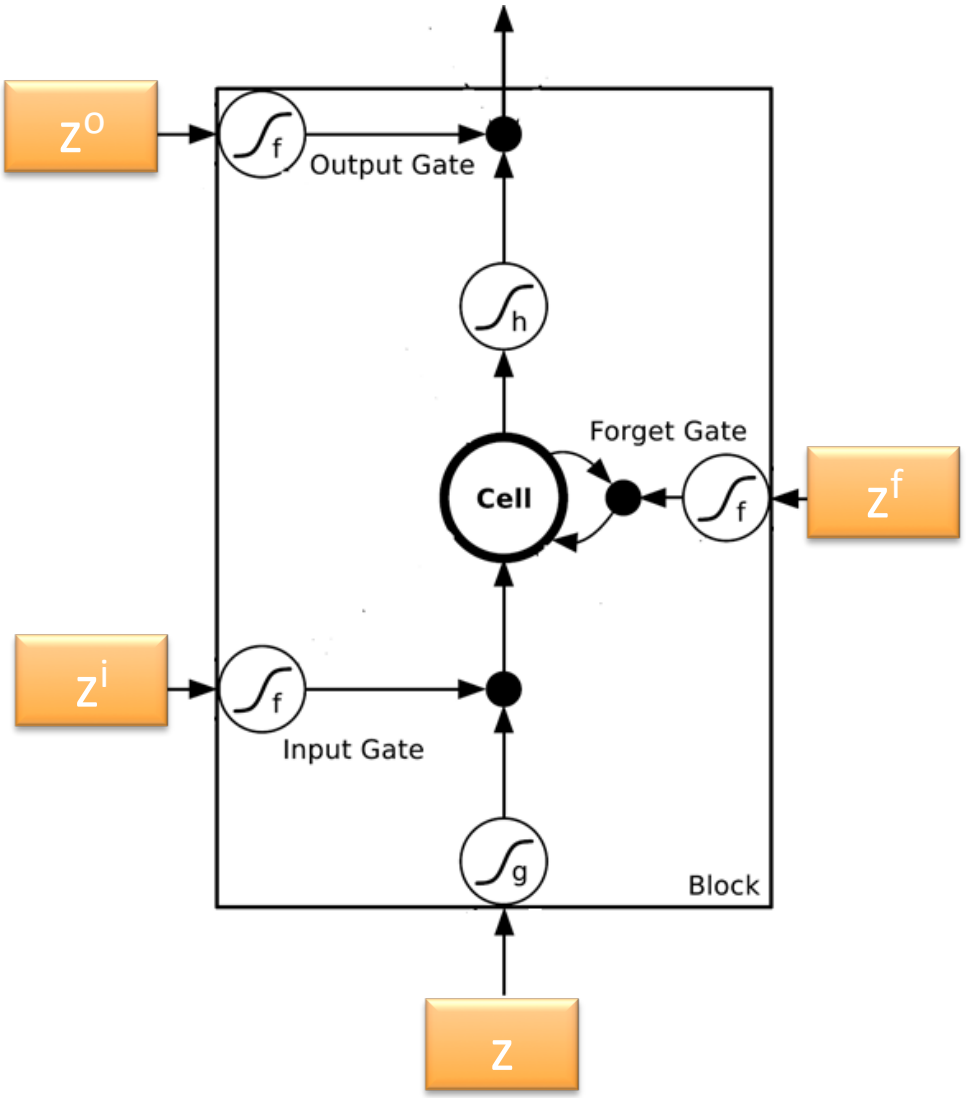

-3
≈1
10
f
Output Gate
-3
h
Forget Gate
-3
-10
f
≈0
≈1
-3
10
f
Input Gate
-3
g
Block
-3

LSTM


c^{t-1}
vector
Output Gate
Forget Gate
Cell
Input Gate
Block
.....
z^f
z^i
z
z^o
4 vectors
x^t

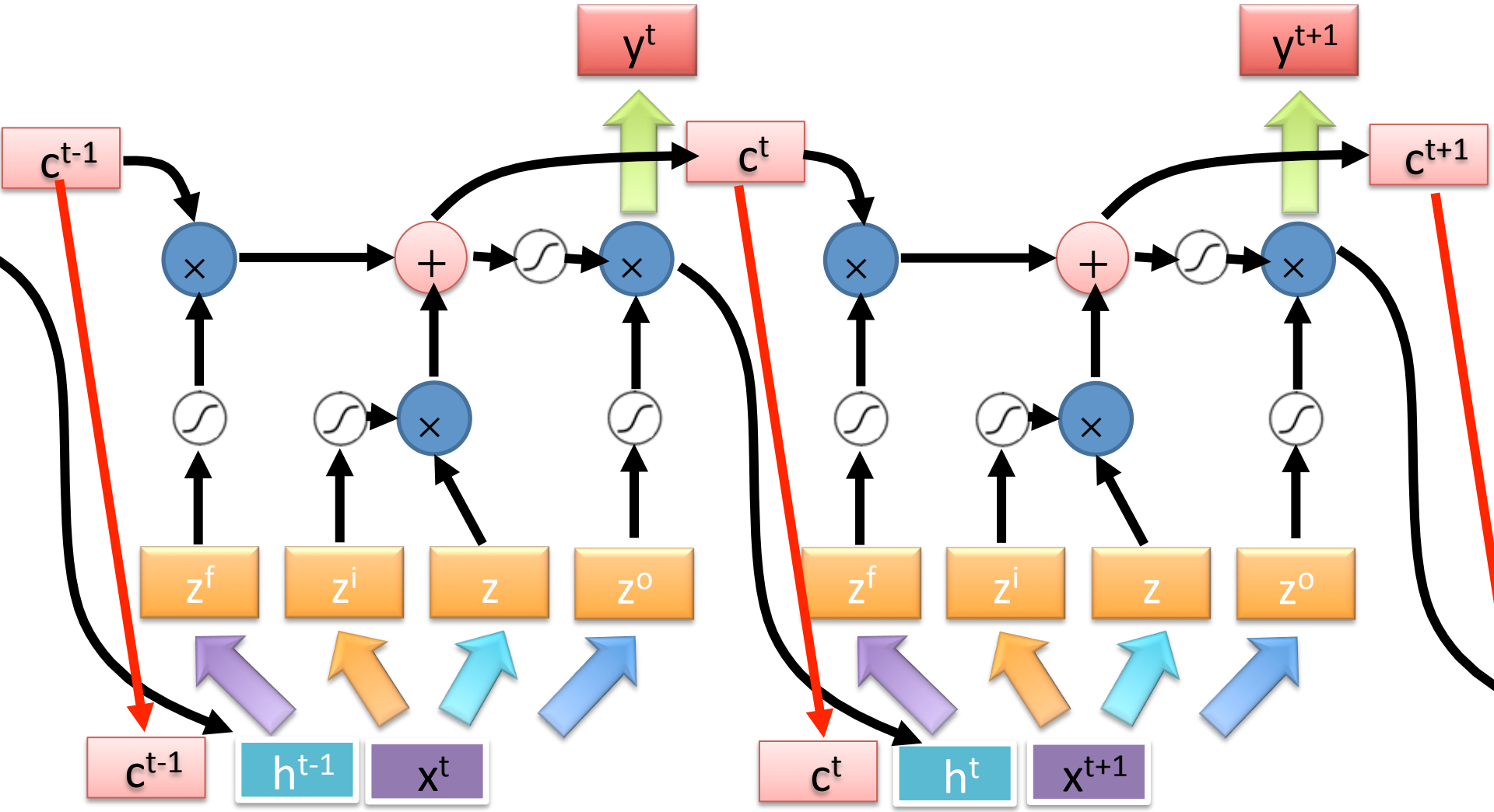
LSTM


y^t
c^{t-1}
z^f
z^i
z
z^o
x^t

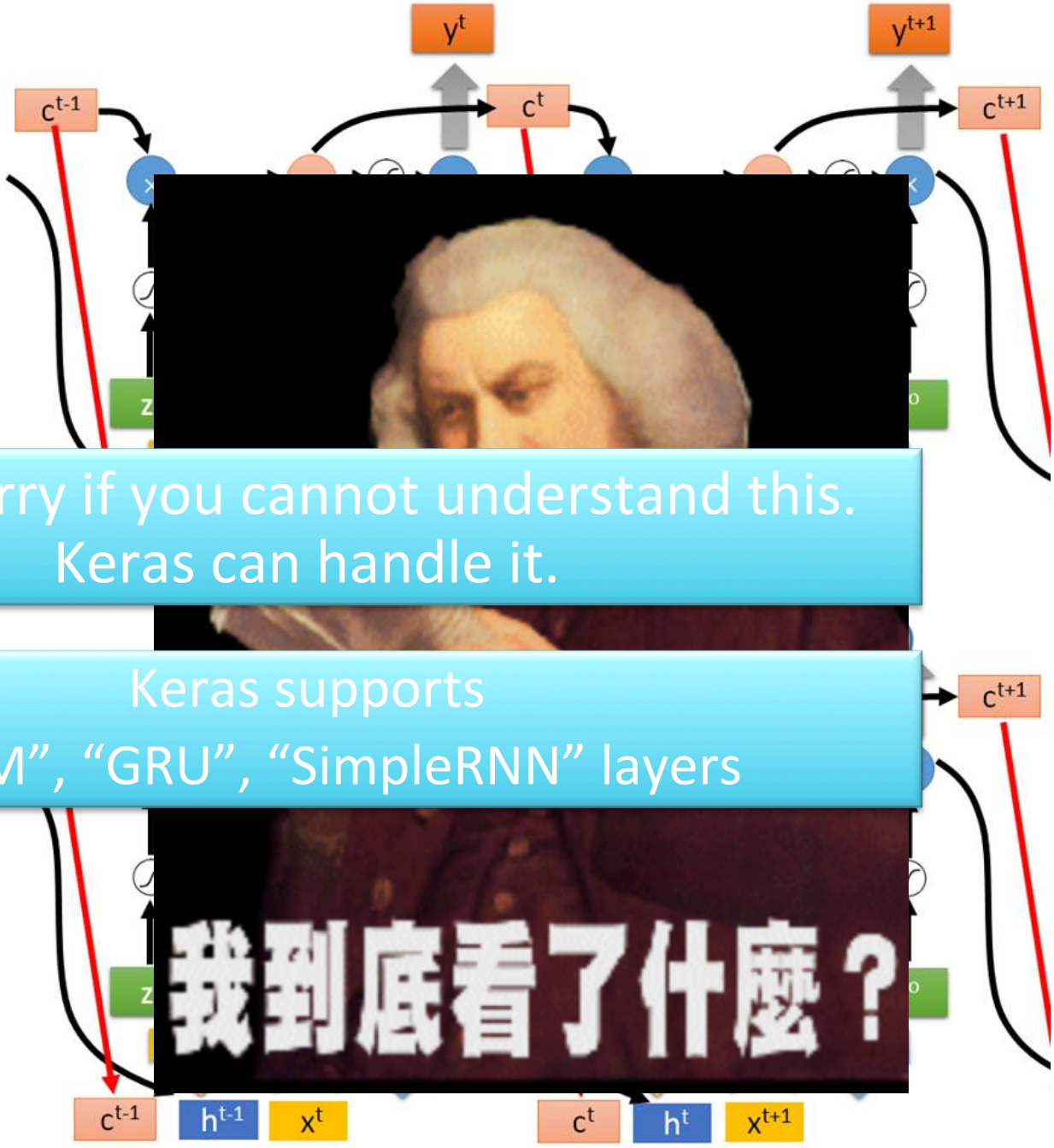

z^o
Output Gate
h
Forget Gate
Cell
z^f
z^i
Input Gate
g
Block
z

LSTM

Extension: “peephole”



Multiple-layer LSTM



This is quite standard now.

https://img.komicolle.org/2015-09-20/src/14426967627131.gif

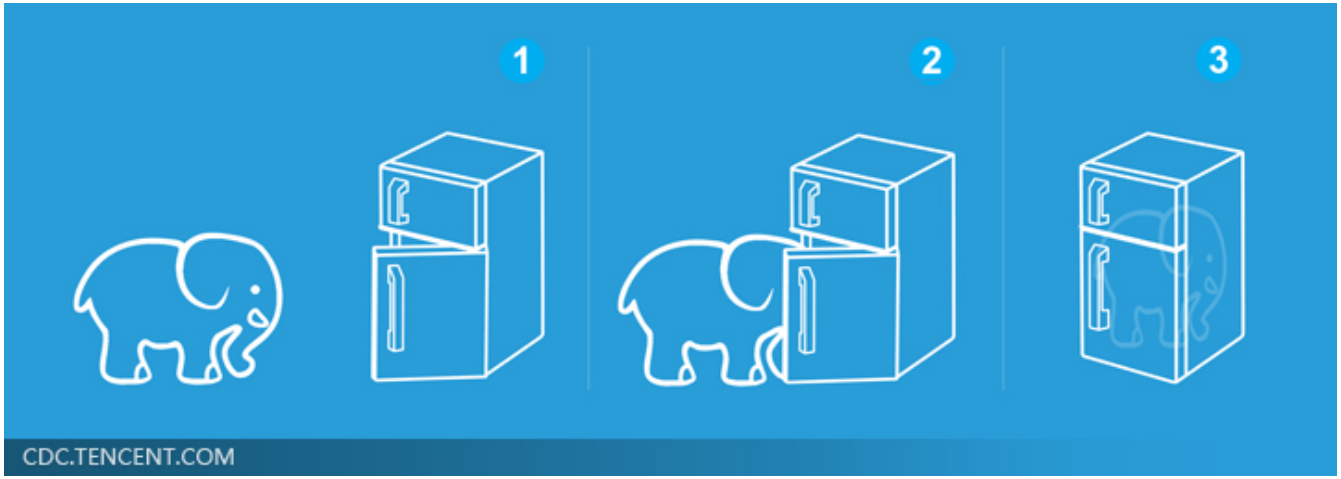
Three Steps for Deep Learning

Step 1: define a set of function

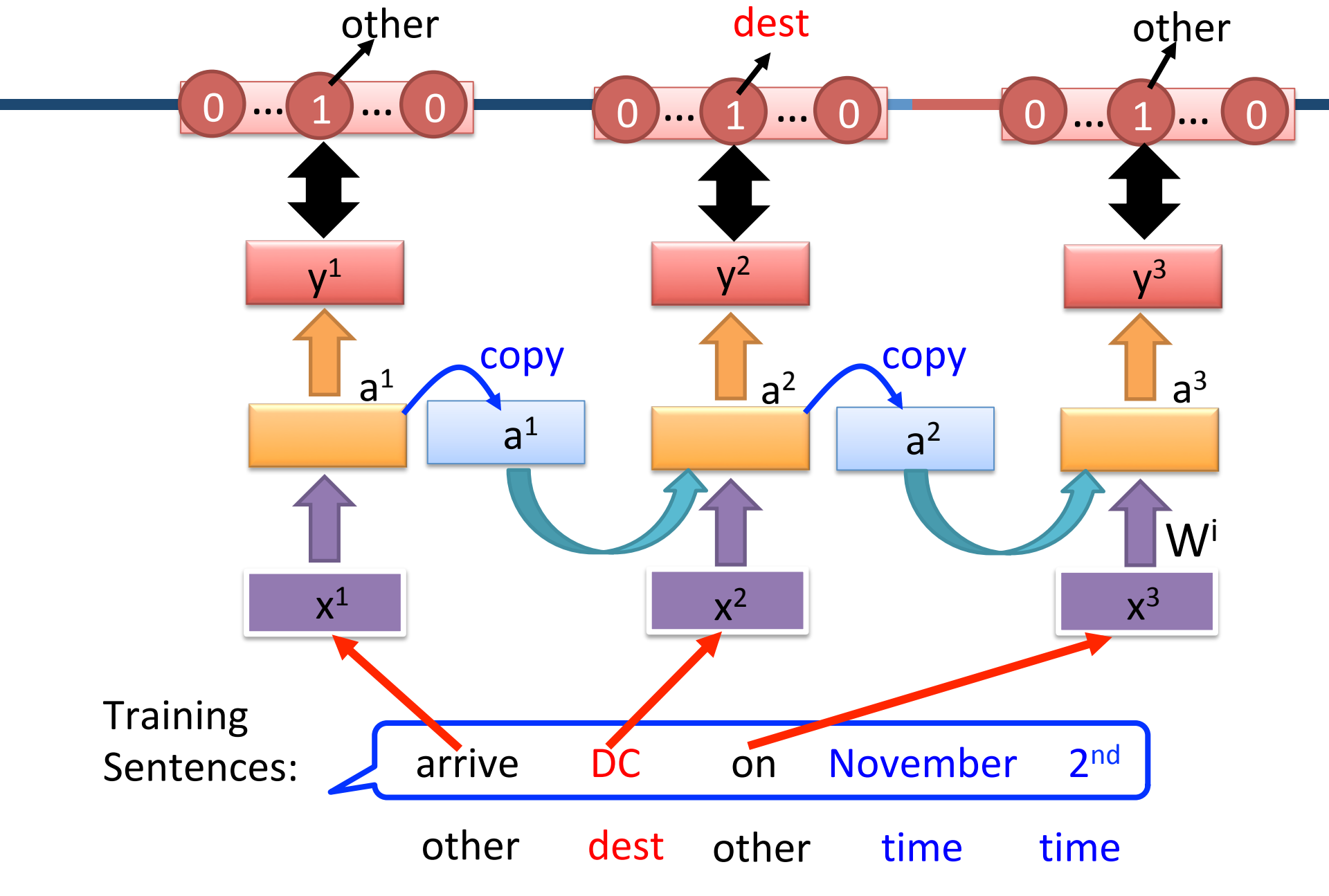
Step 2: goodness of function

Step 3: pick the best function

Deep Learning is so simple ……



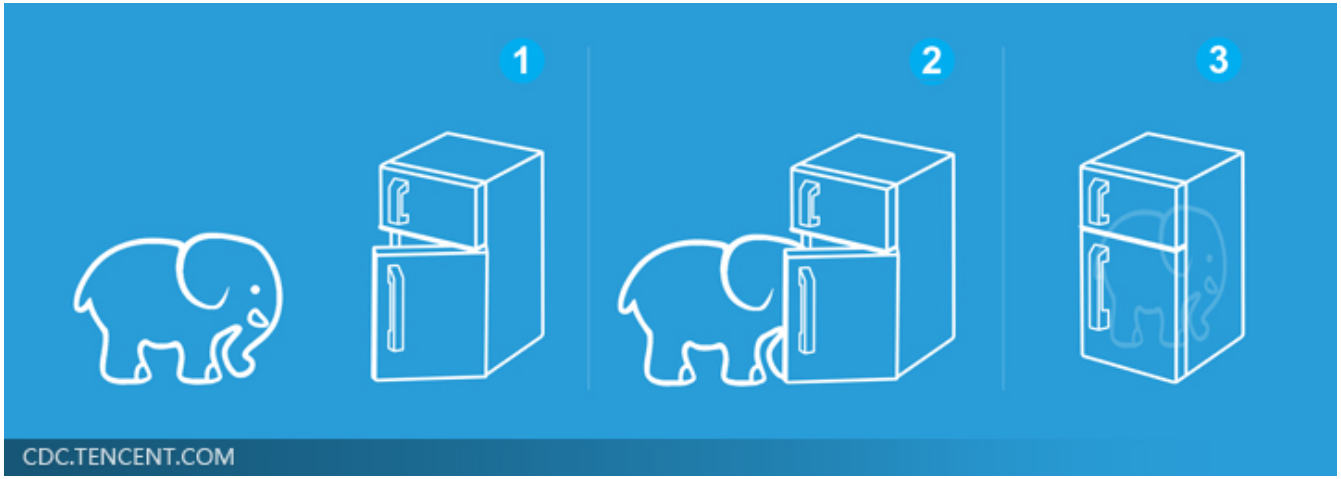
Learning Target



Three Steps for Deep Learning

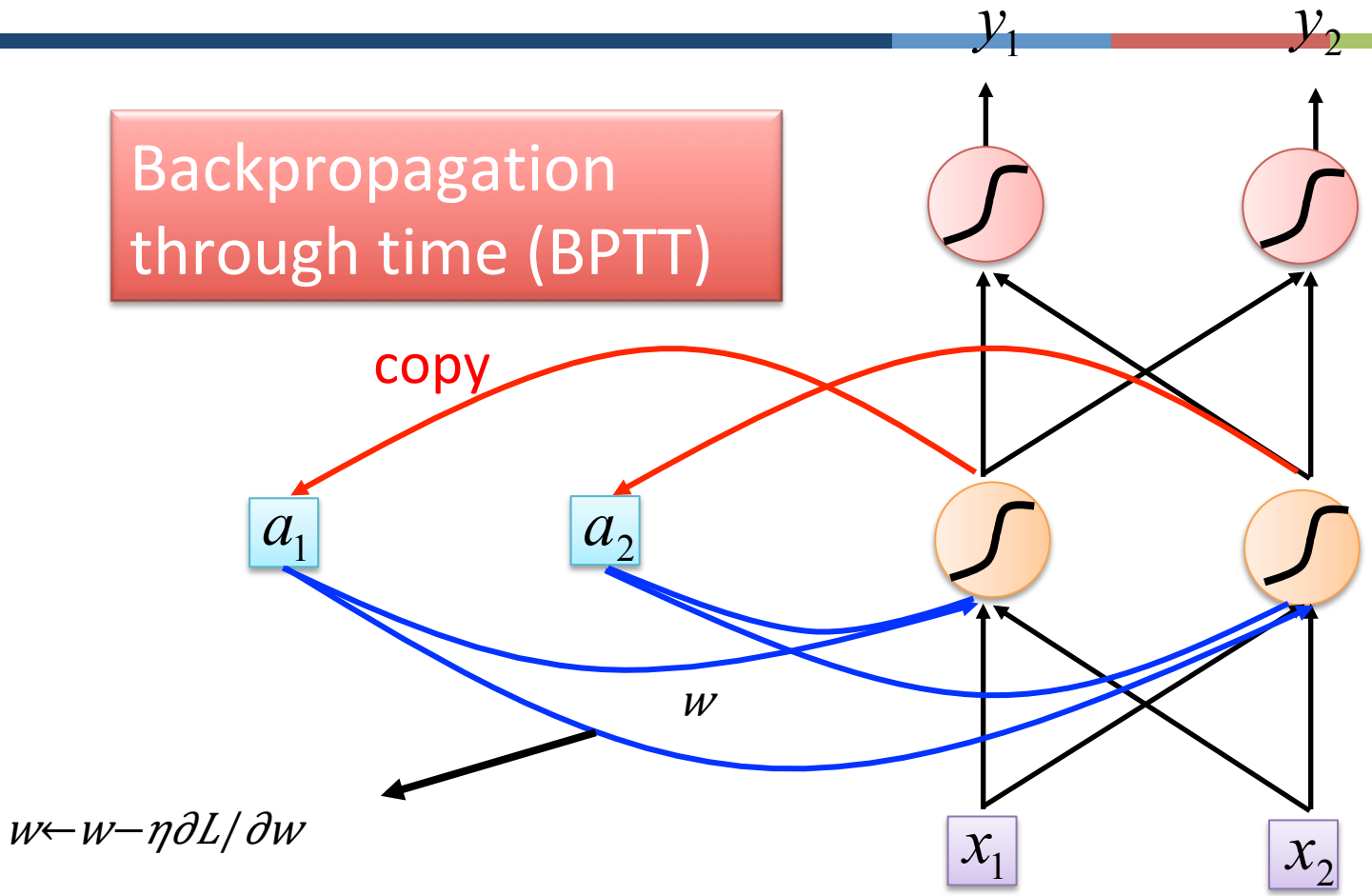
Step 1: define a set of function → Step 2: goodness of function → Step 3: pick the best function

Deep Learning is so simple ……



Learning

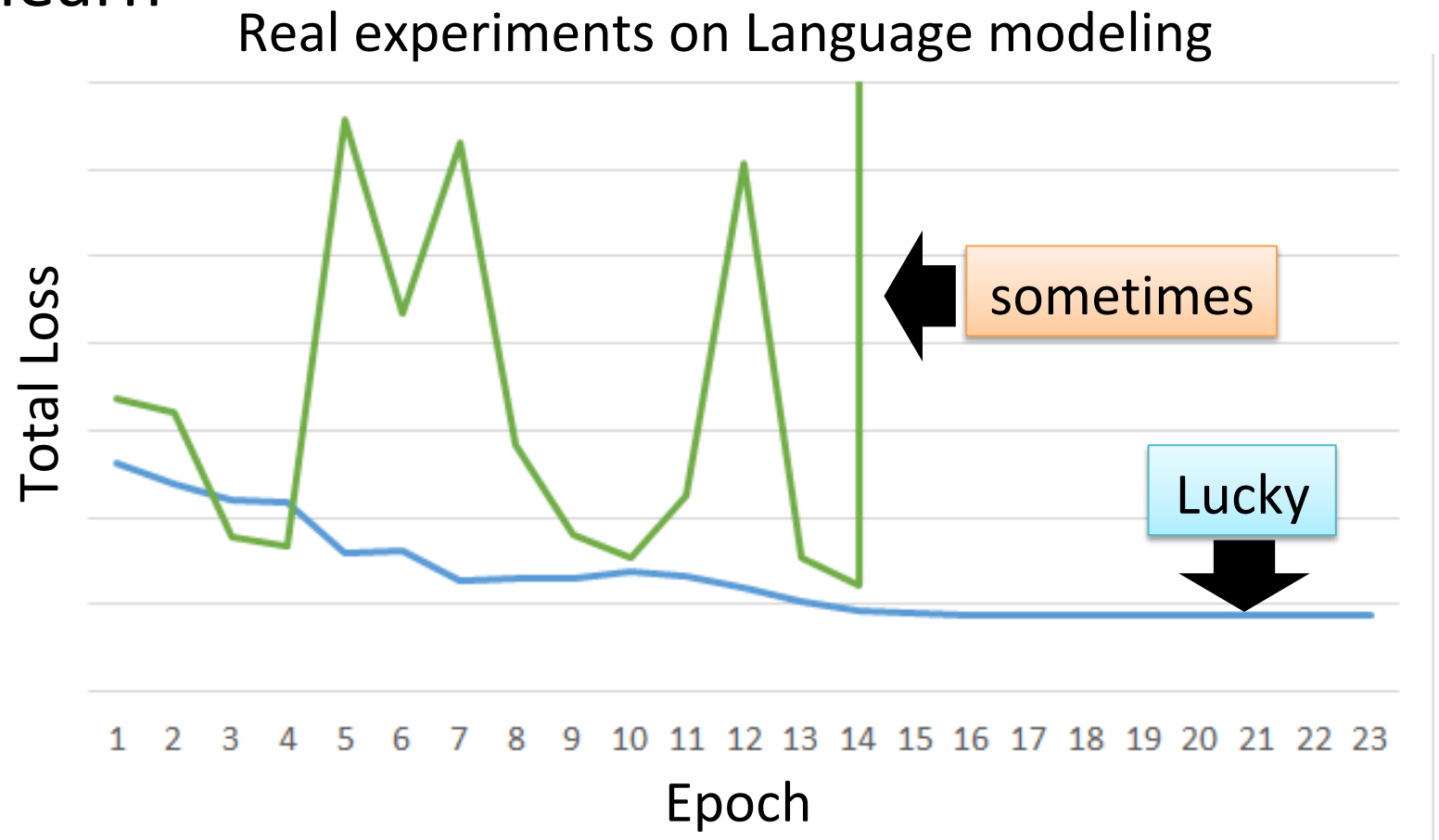
Backpropagation through time (BPTT)



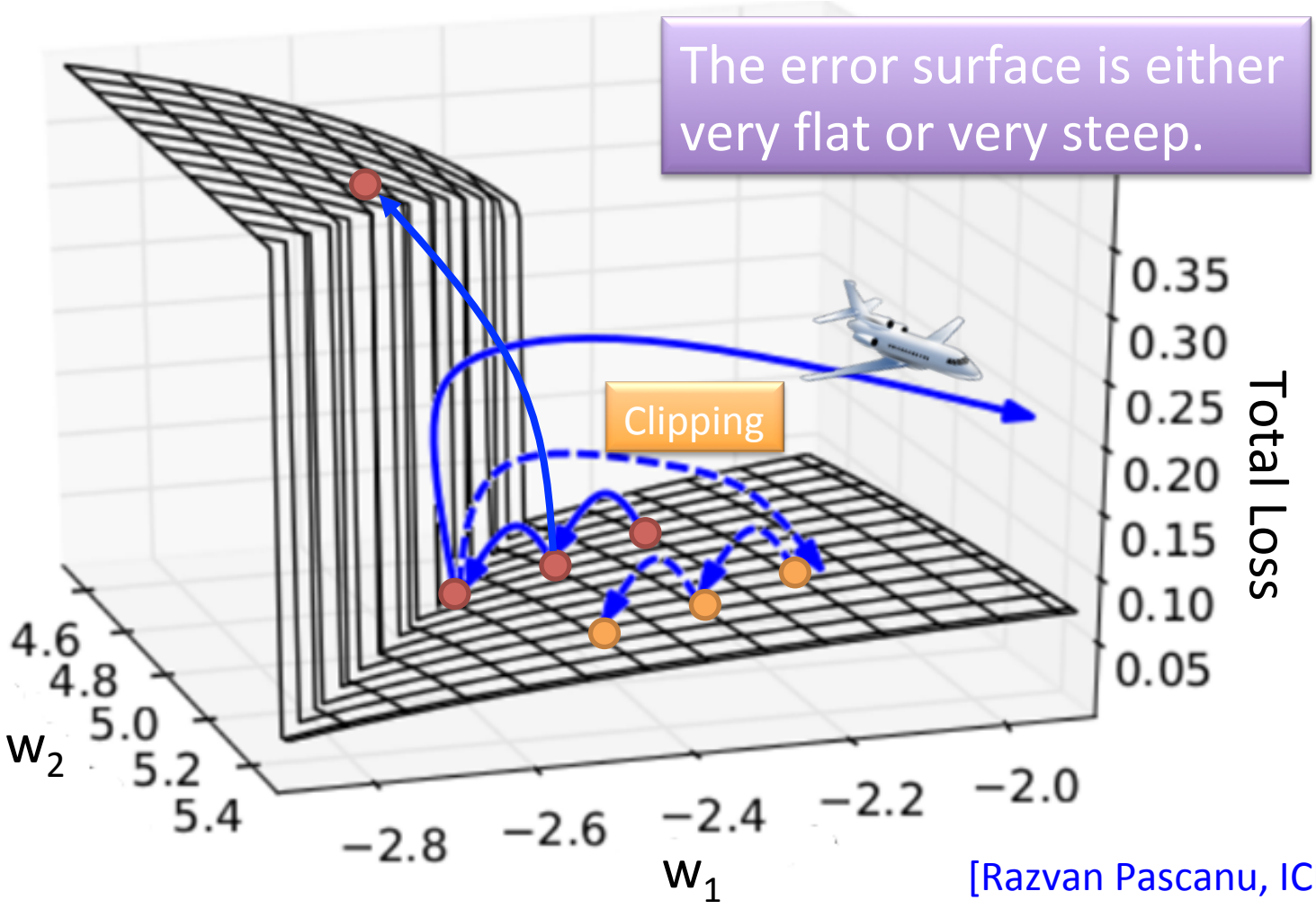
RNN Learning is very difficult in practice.

Unfortunately ……

- RNN-based network is not always easy to learn



The error surface is rough.



[Razvan Pascanu, ICML'13]

Why?

$w=1$ → $y^{1000} = 1$

$w=1.01$ → $y^{1000} \approx 20000$

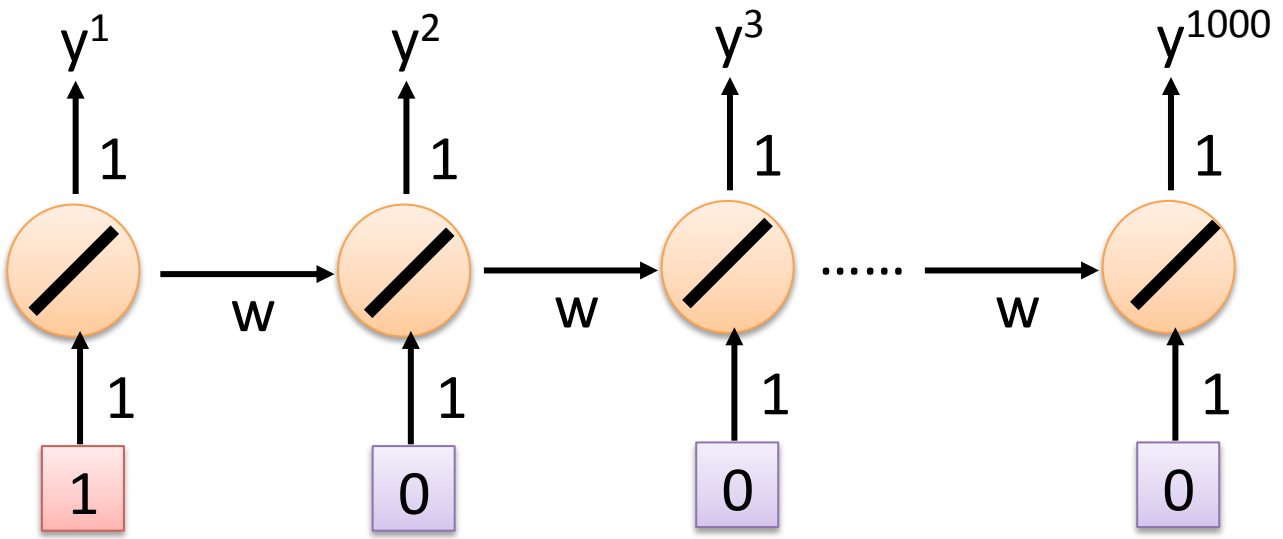
Large $\partial L/\partial w$ → Small Learning rate?

$w=0.99$ → $y^{1000} \approx 0$

$w=0.01$ → $y^{1000} \approx 0$

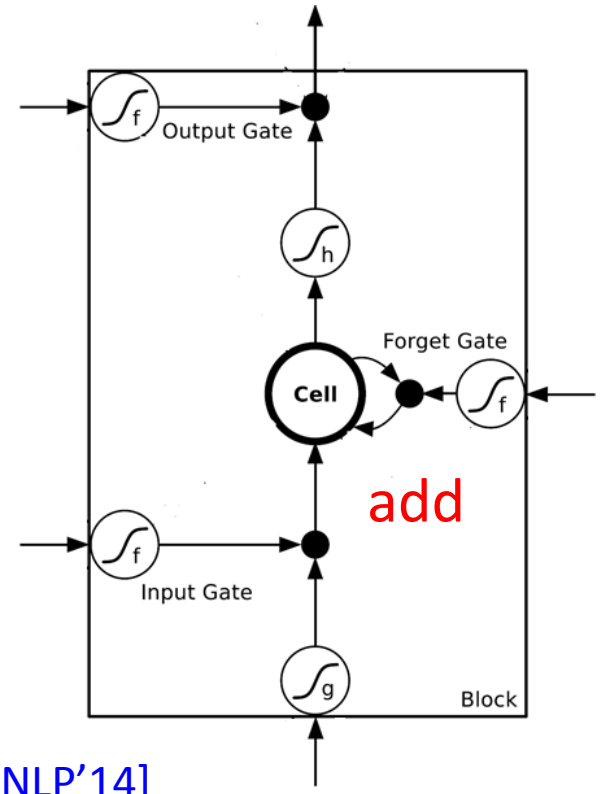
small $\partial L/\partial w$ → Large Learning rate?

Toy Example



Helpful Techniques

- Long Short-term Memory (LSTM)
  - Can deal with gradient vanishing (not gradient explode)
    - Memory and input are ***added***
    - The influence never disappears unless forget gate is closed



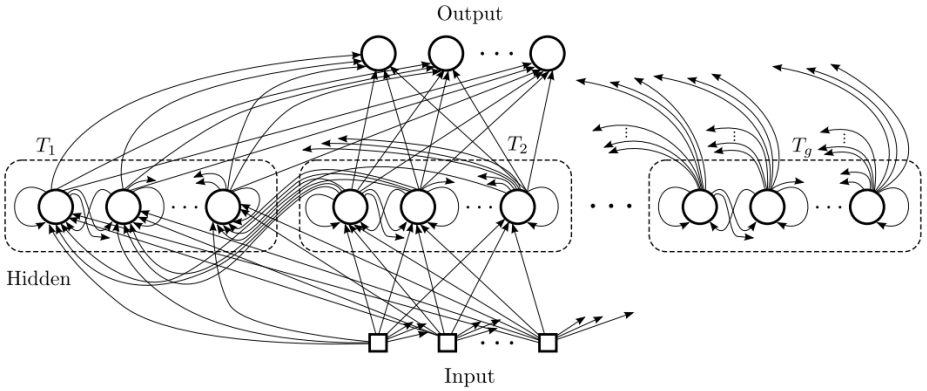
No Gradient vanishing
(If forget gate is

Gated Recurrent Unit (GRU): simpler than LSTM

[Cho, EMNLP'14]

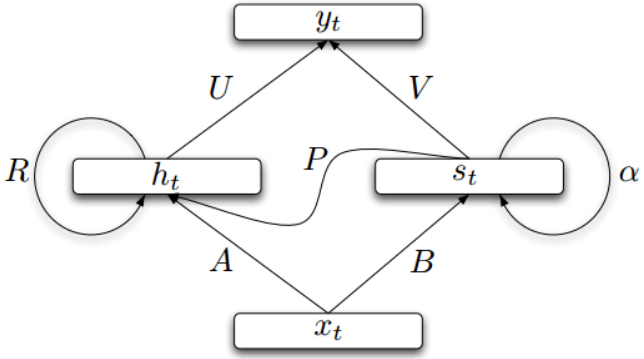
Helpful Techniques

Clockwise RNN



[Jan Koutnik, JMLR'14]

Structurally Constrained Recurrent Network (SCRN)



[Tomas Mikolov, ICLR'15]

Vanilla RNN Initialized with Identity matrix + ReLU activation function [Quoc V. Le, arXiv'15]

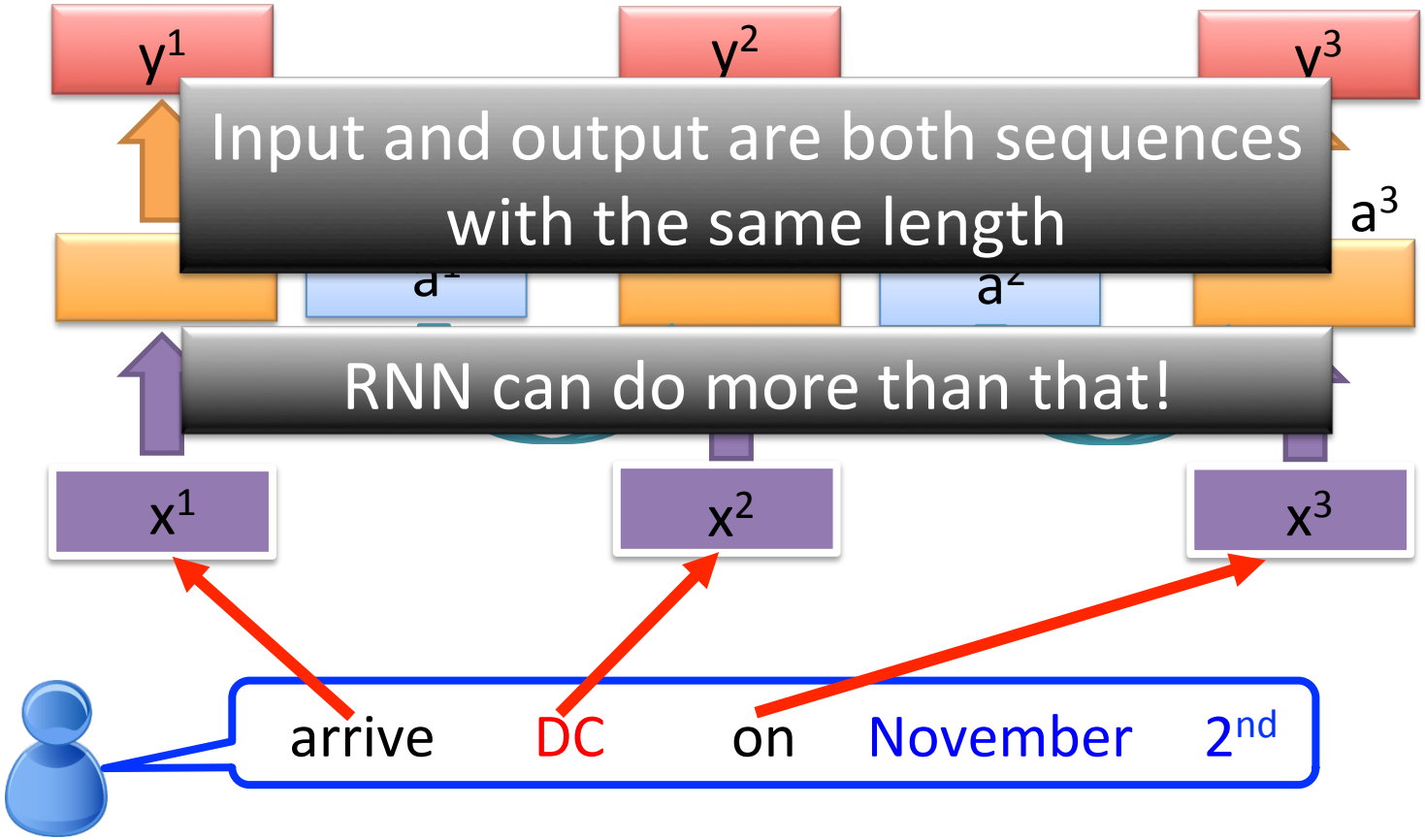
- Outperform or be comparable with LSTM in 4 different tasks

More Applications ……

Probability of "arrive" in each slot

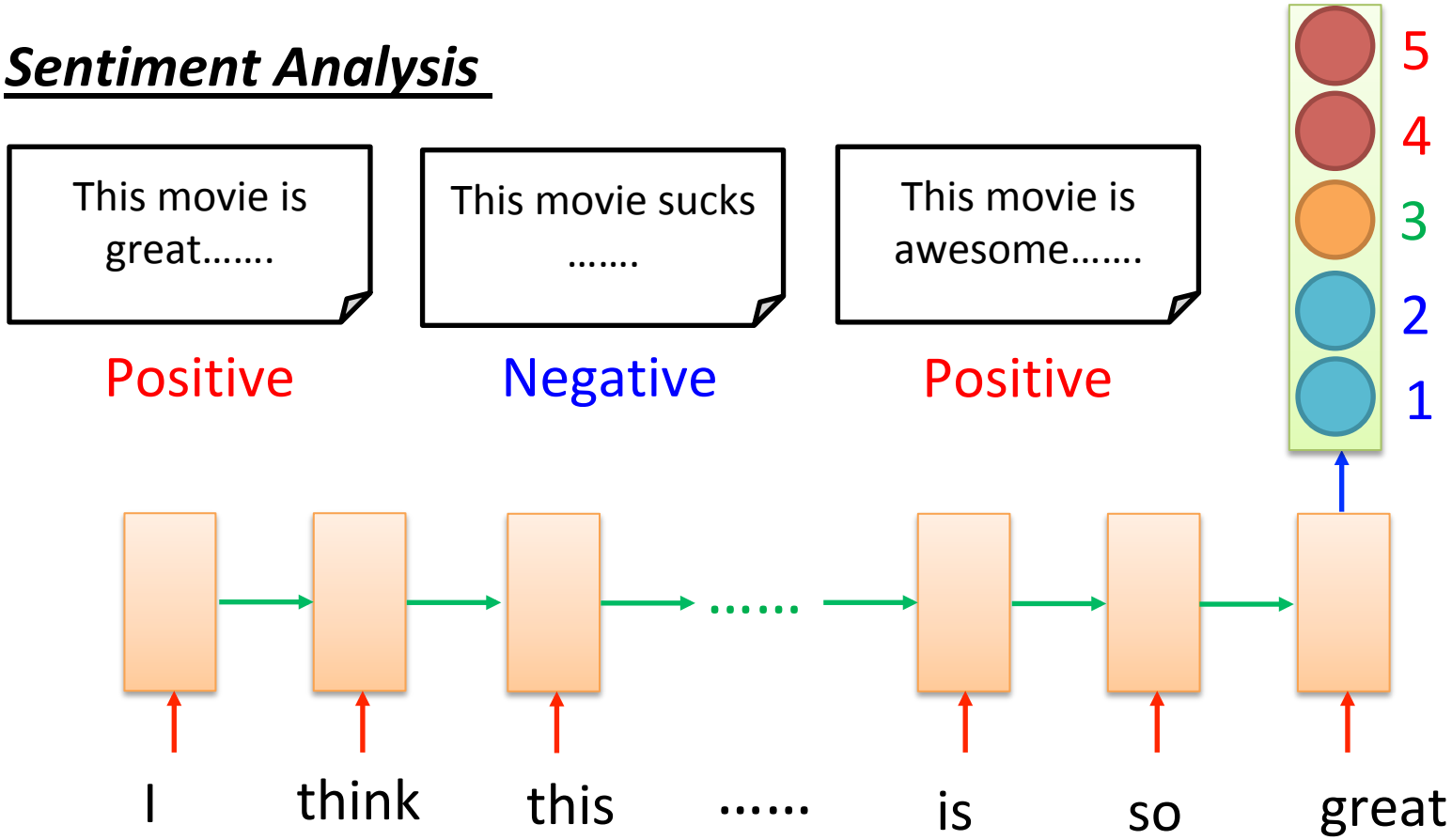
Probability of "DC" in each slot

Probability of "on" in each slot



Many to one

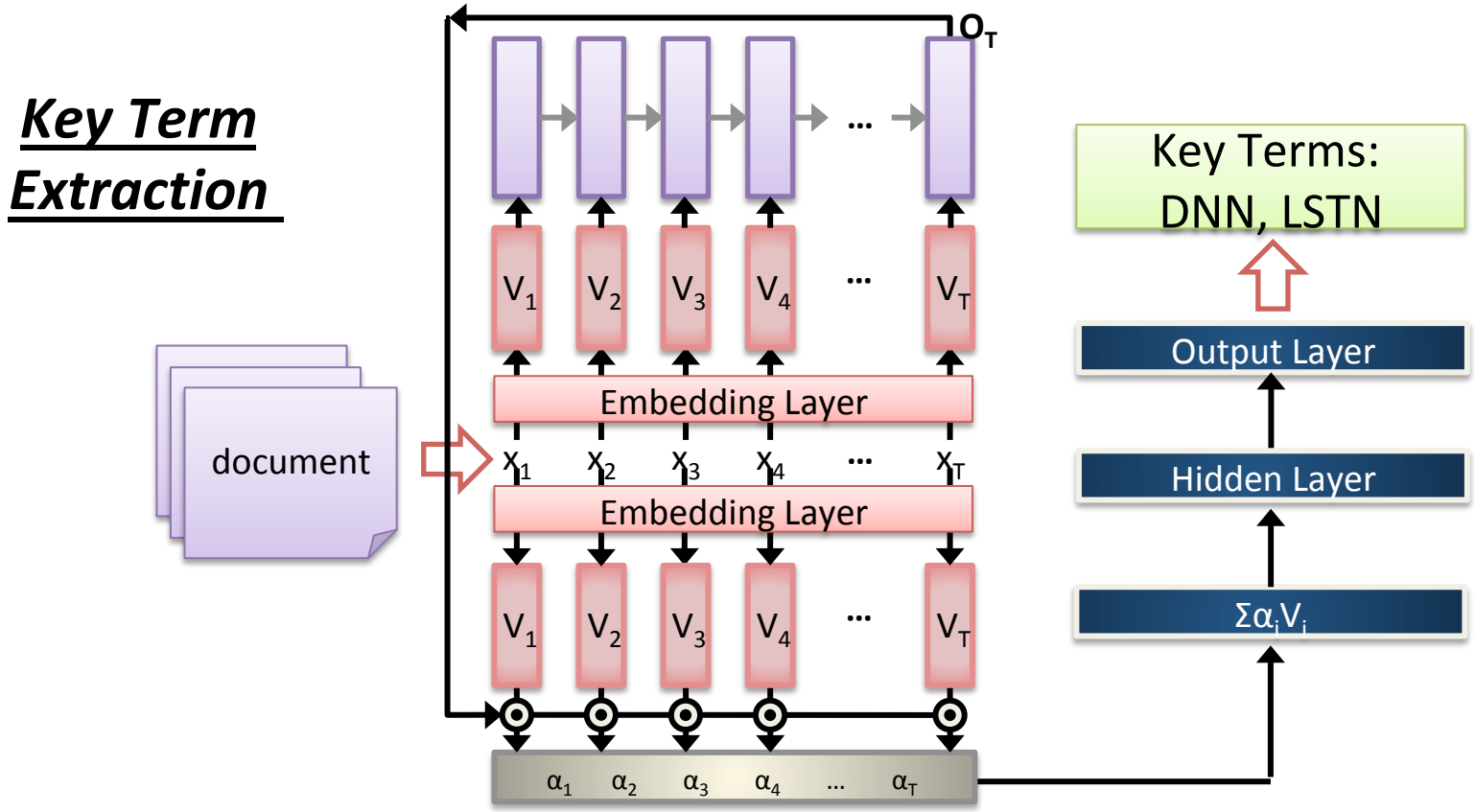
- Input is a vector sequence, but output is only one vector



Many to one

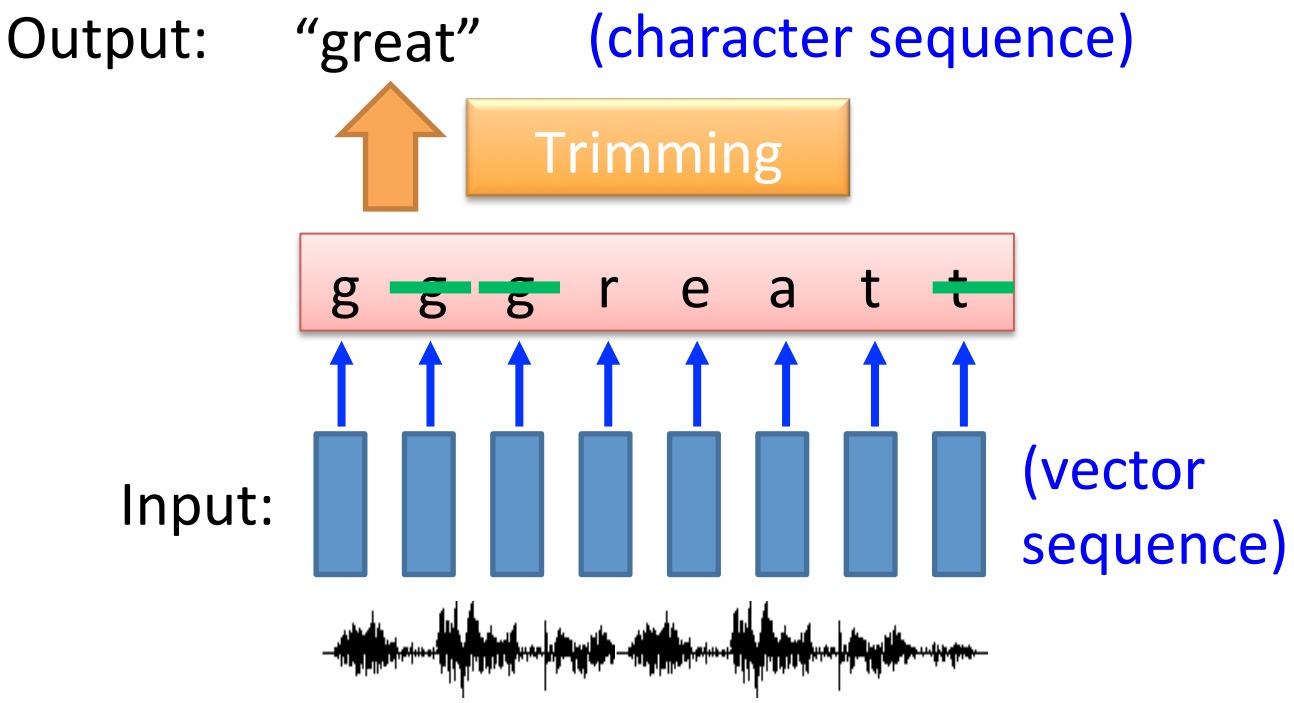
[Shen & Lee, Interspeech 16]

- Input is a vector sequence, but output is only one vector



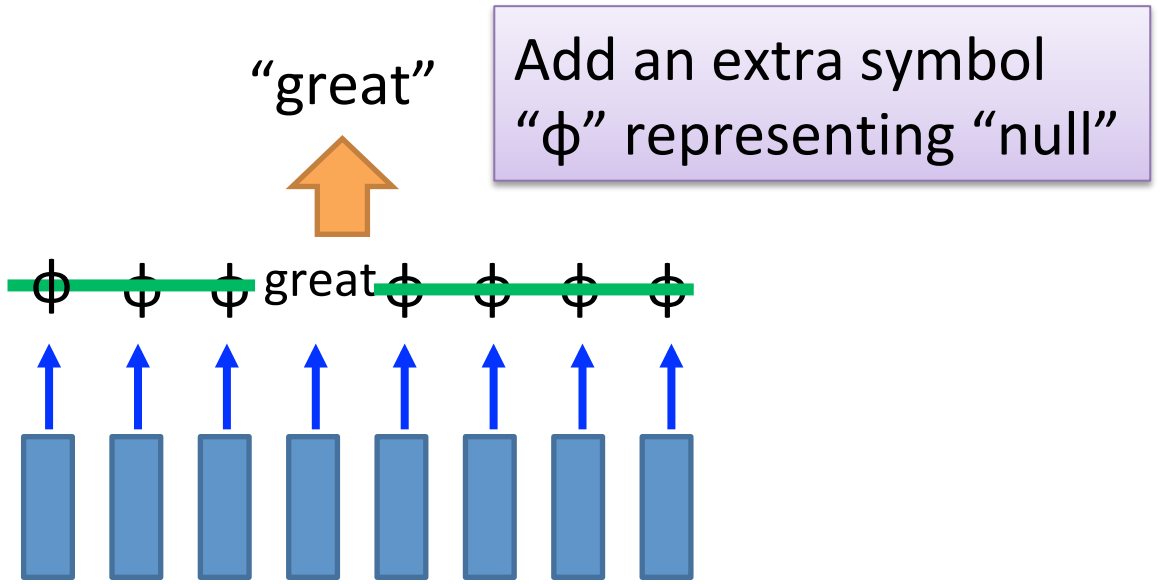
Many to Many (Output is shorter)

- Both input and output are both sequences, ***but the output is shorter.***
  - E.g. ***Speech Recognition***



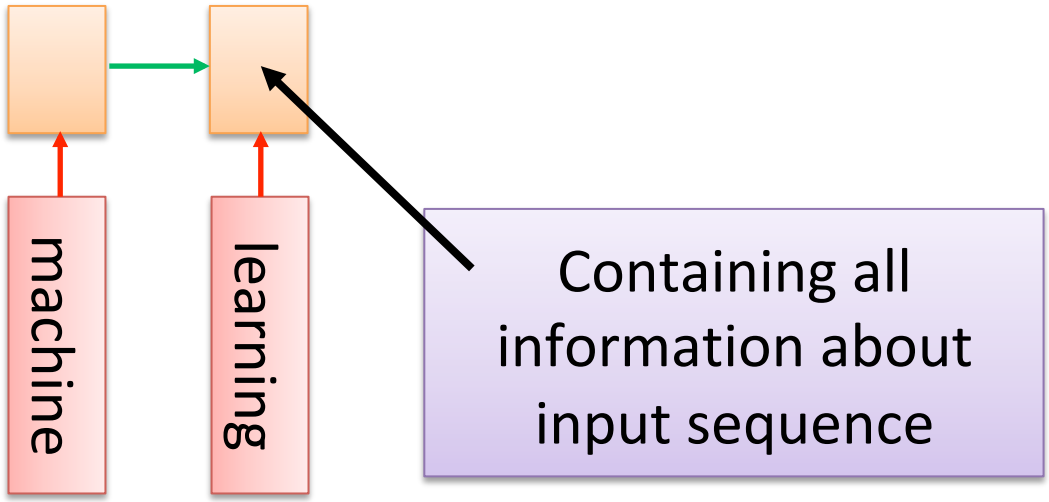
Many to Many (Output is shorter)

- Both input and output are both sequences, ***but the output is shorter.***
  - Connectionist Temporal Classification (CTC) [Alex Graves, ICML'06] [Alex Graves, ICML'14][Haşim Sak, Interspeech'15][Jie Li, Interspeech'15][Andrew Senior, ASRU'15]



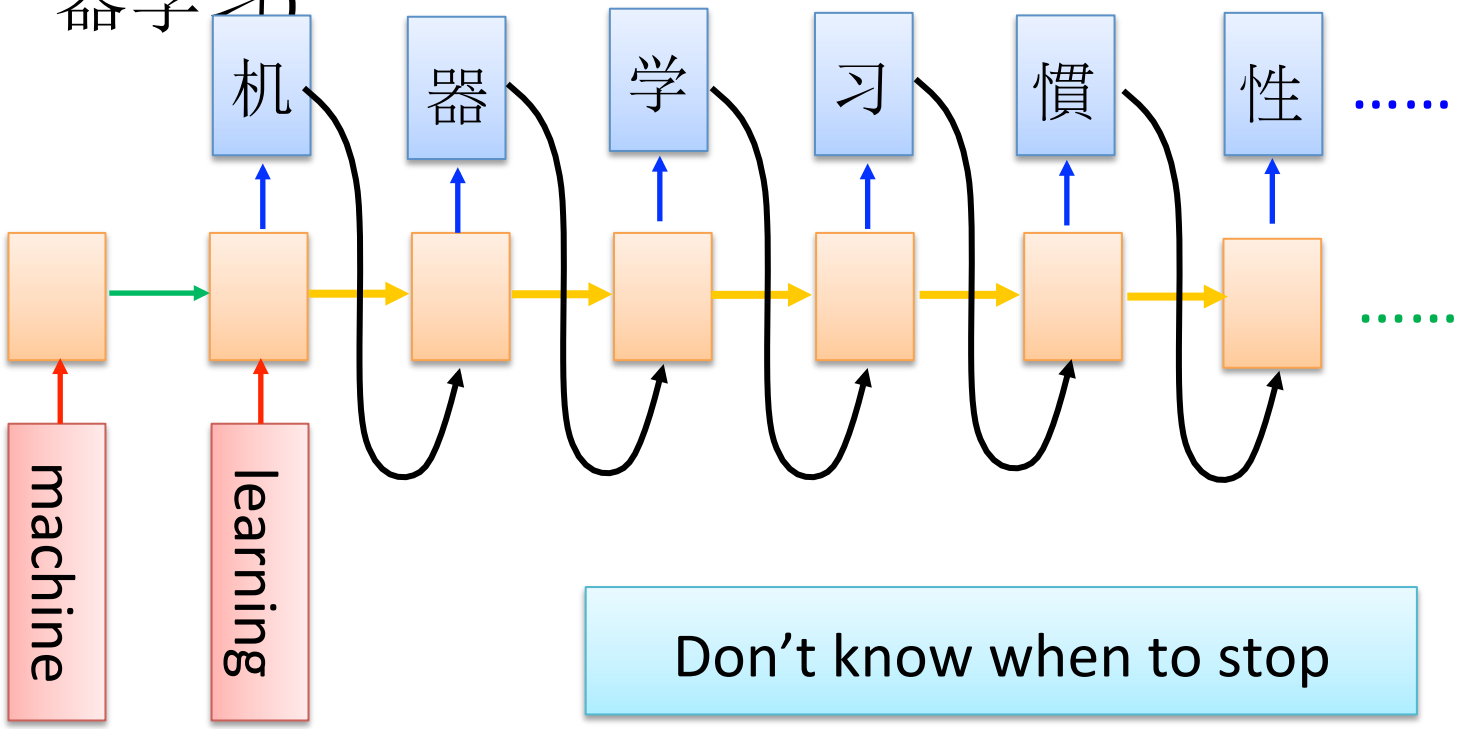
Many to Many (No Limitation)

- Both input and output are both sequences ***with different lengths***. → ***Sequence to sequence learning***
  - E.g. ***Machine Translation*** (machine learning→机器学习)



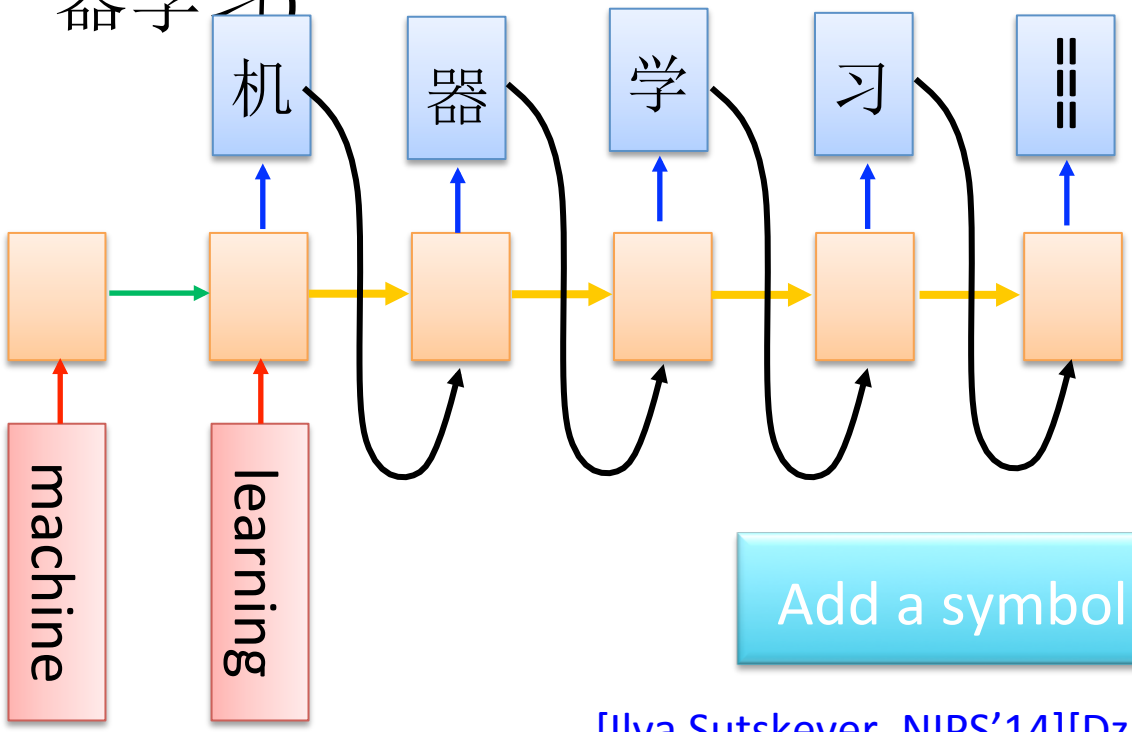
Many to Many (No Limitation)

- Both input and output are both sequences ***with different lengths***. → ***Sequence to sequence learning***
  - E.g. ***Machine Translation*** (machine learning→机器学习)



Many to Many (No Limitation)

- Both input and output are both sequences ***with different lengths***. → ***Sequence to sequence learning***
  - E.g. ***Machine Translation*** (machine learning→机器学习)



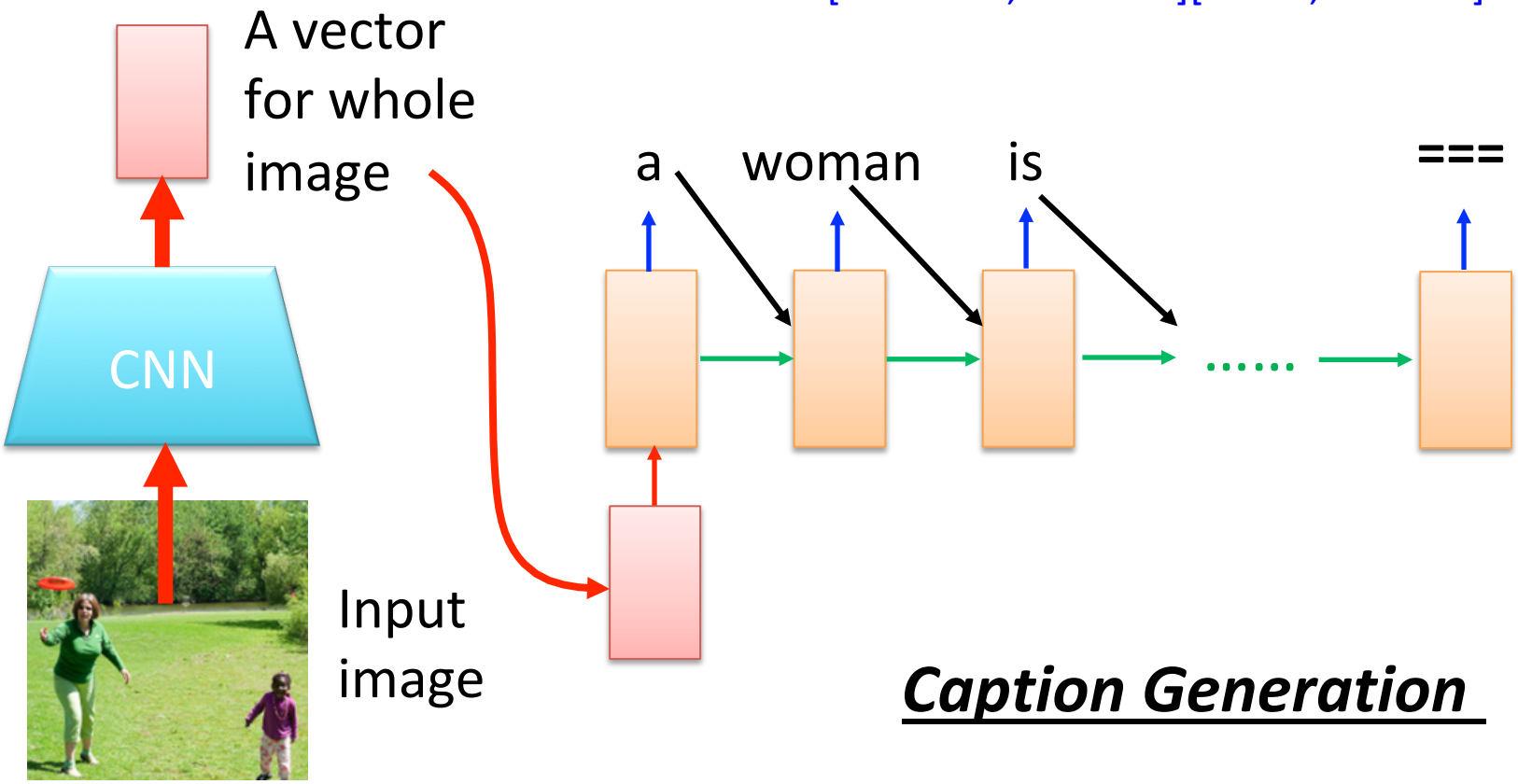
Add a symbol "===" (stop)

[Ilya Sutskever, NIPS'14][Dzmitry Bahdanau, arXiv'15]

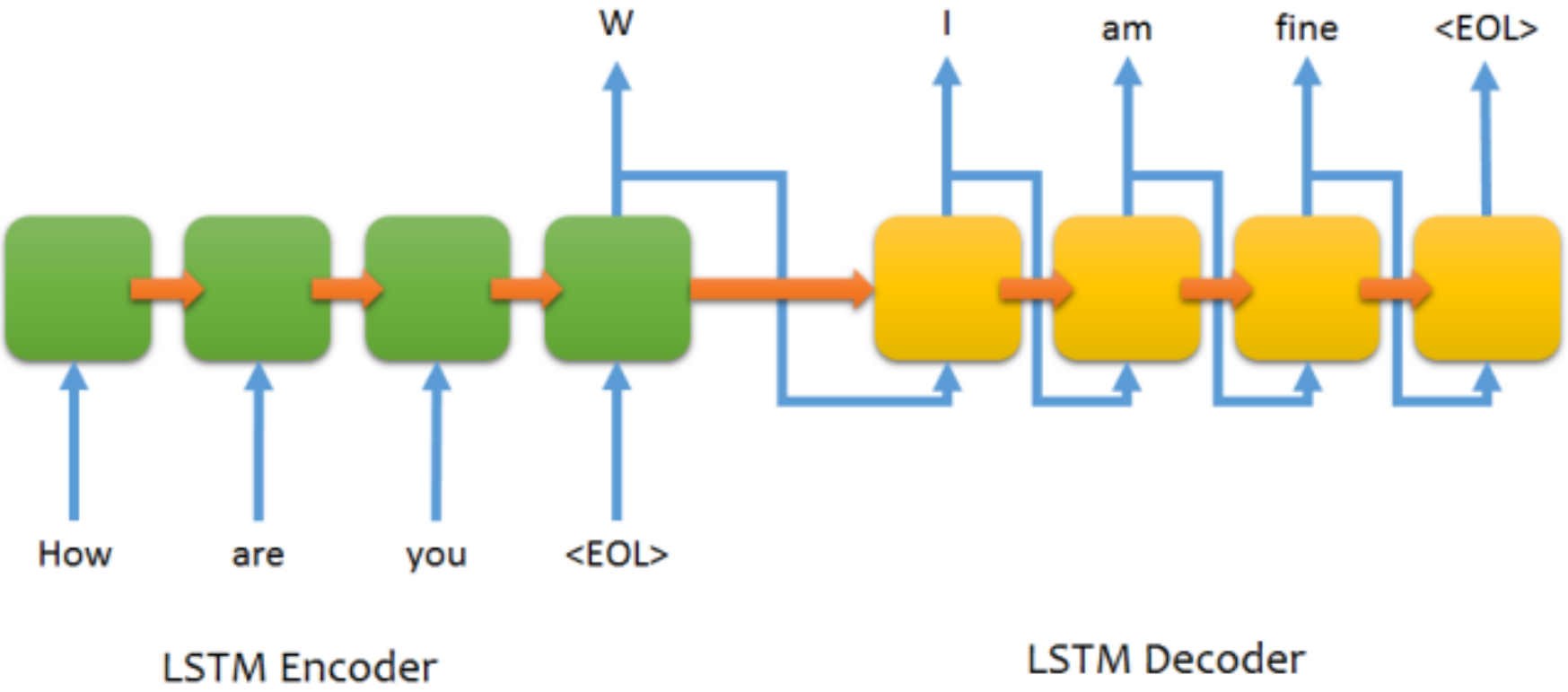
Image Caption Generation

- Input an image, but output a sequence of words

[Kelvin Xu, arXiv'15][Li Yao, ICCV'15]



Chat-bot



movie (~40,000 sentences), presidential debate…