



UNIVERSITY *of*
LOUISIANA
L A F A Y E T T E *

Lecture 3

Neural Network and Deep Learning

Xu Yuan

University of Louisiana at Lafayette

Machine Learning ~ Looking for a Function

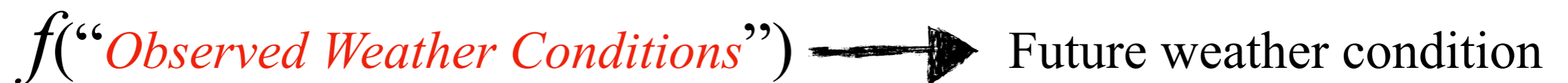
- Image recognition



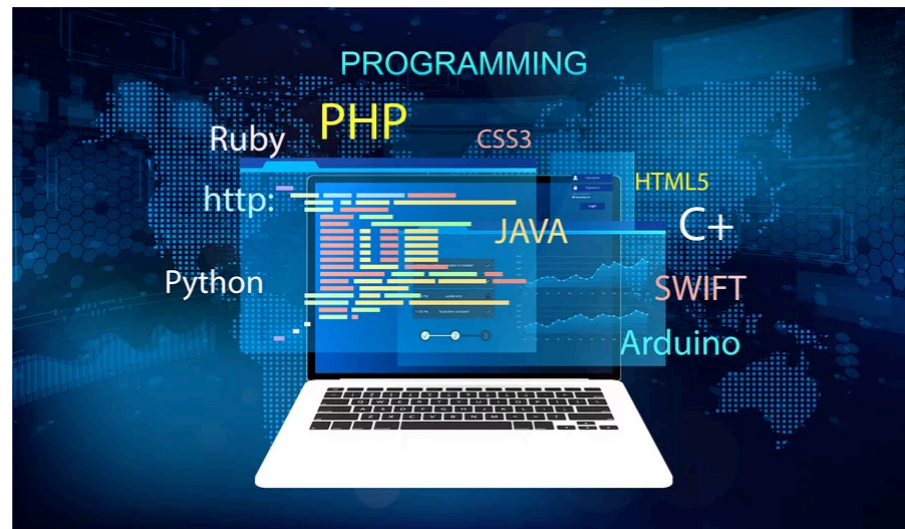
- Spam classification



- Weather prediction



Machine Learning ~ Training Framework



Training
Data



A set of functions
(models) f_1, f_2, \dots



Goodness of
function f



Pick the "best"
function f^*

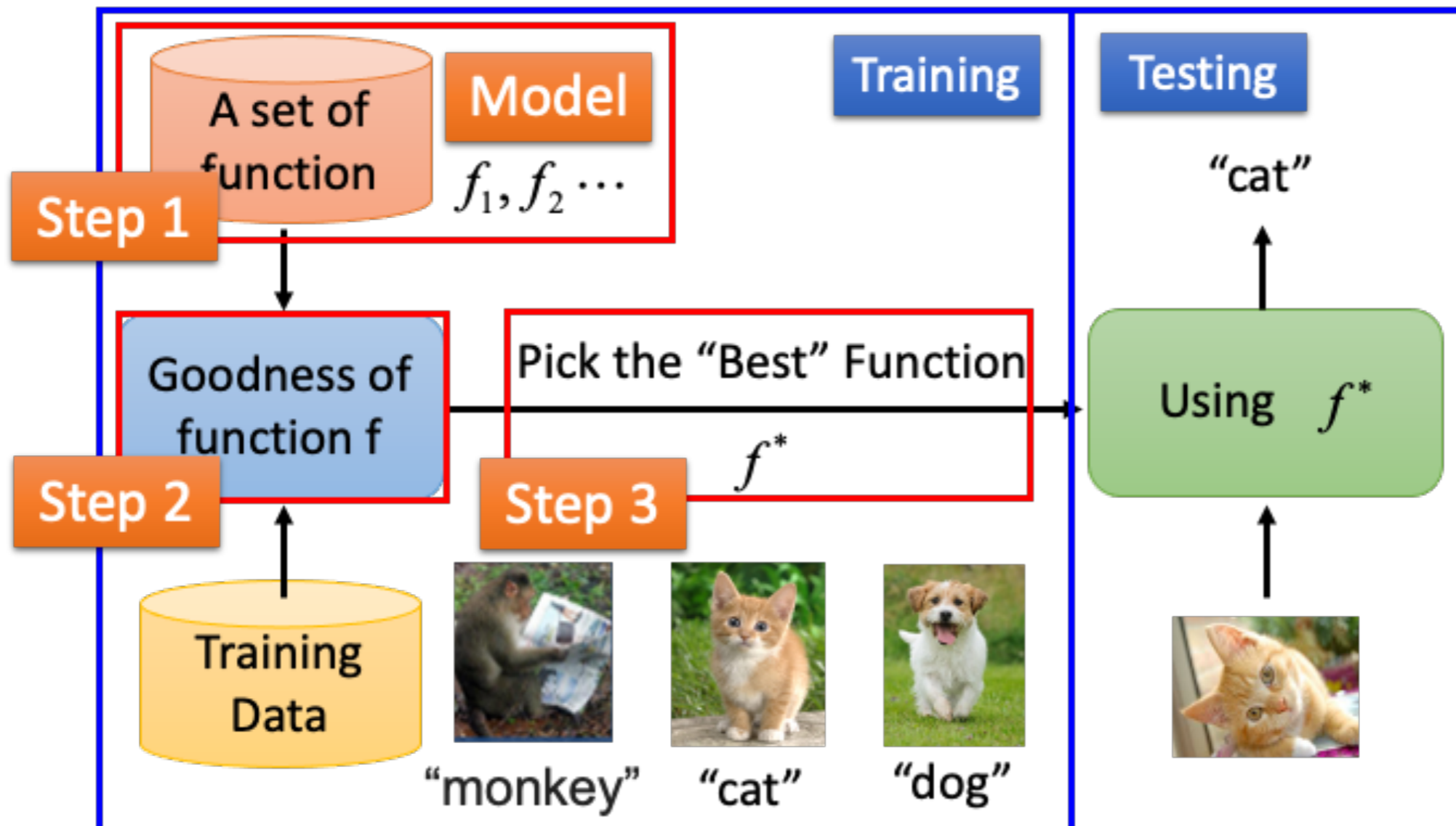
Trained Model

Framework

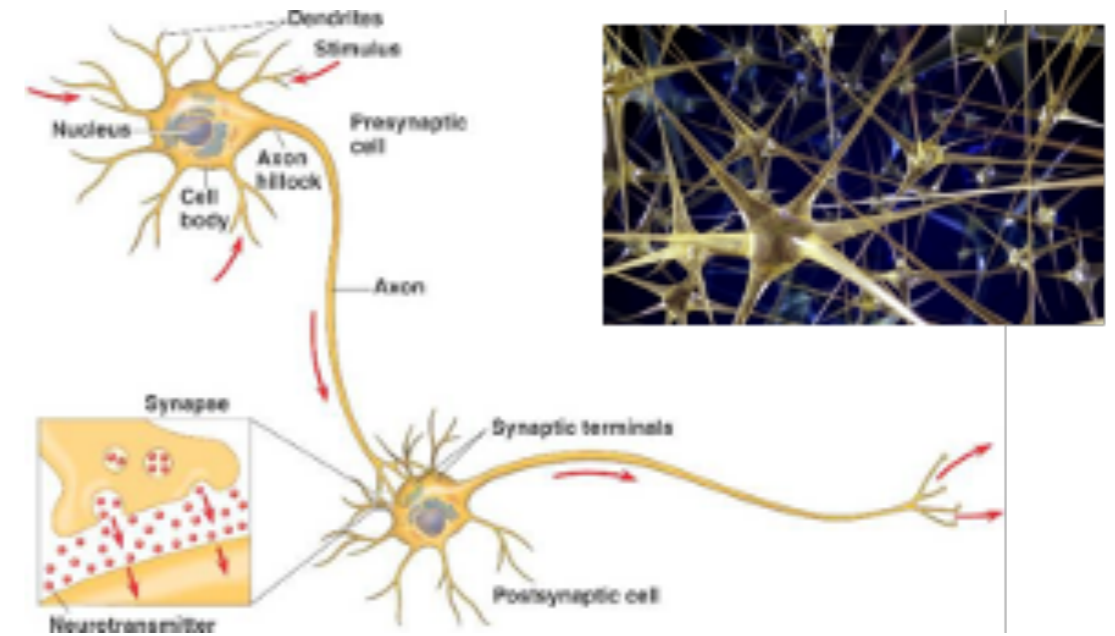
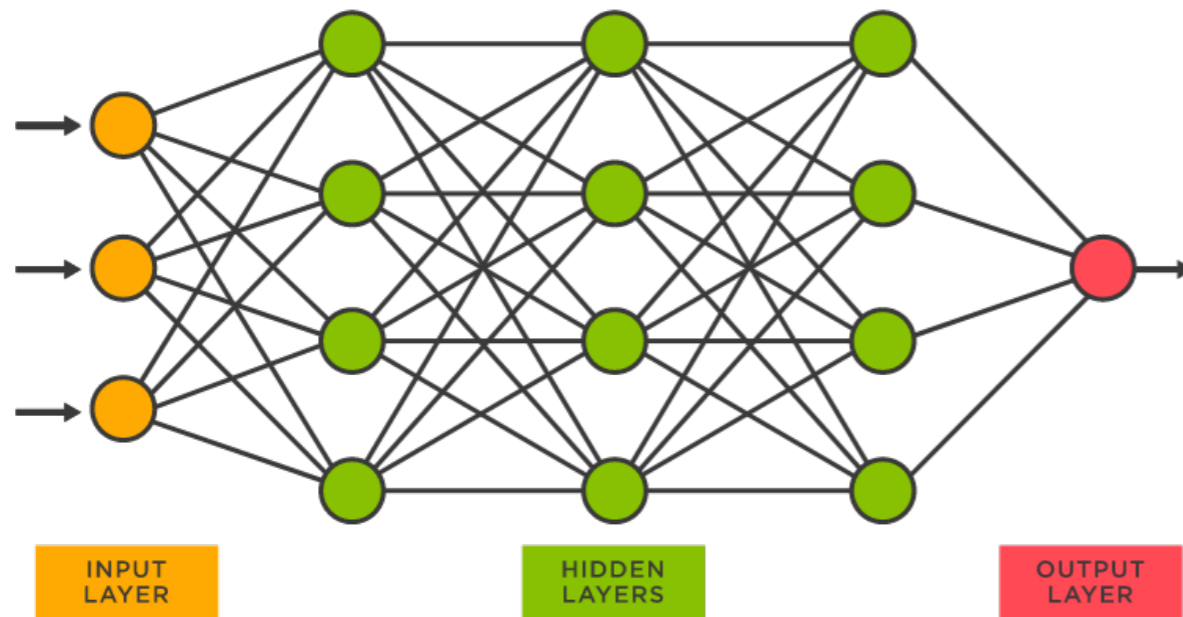
Image Recognition:

Framework

$$f(\text{Image of a cat}) = \text{"cat"}$$

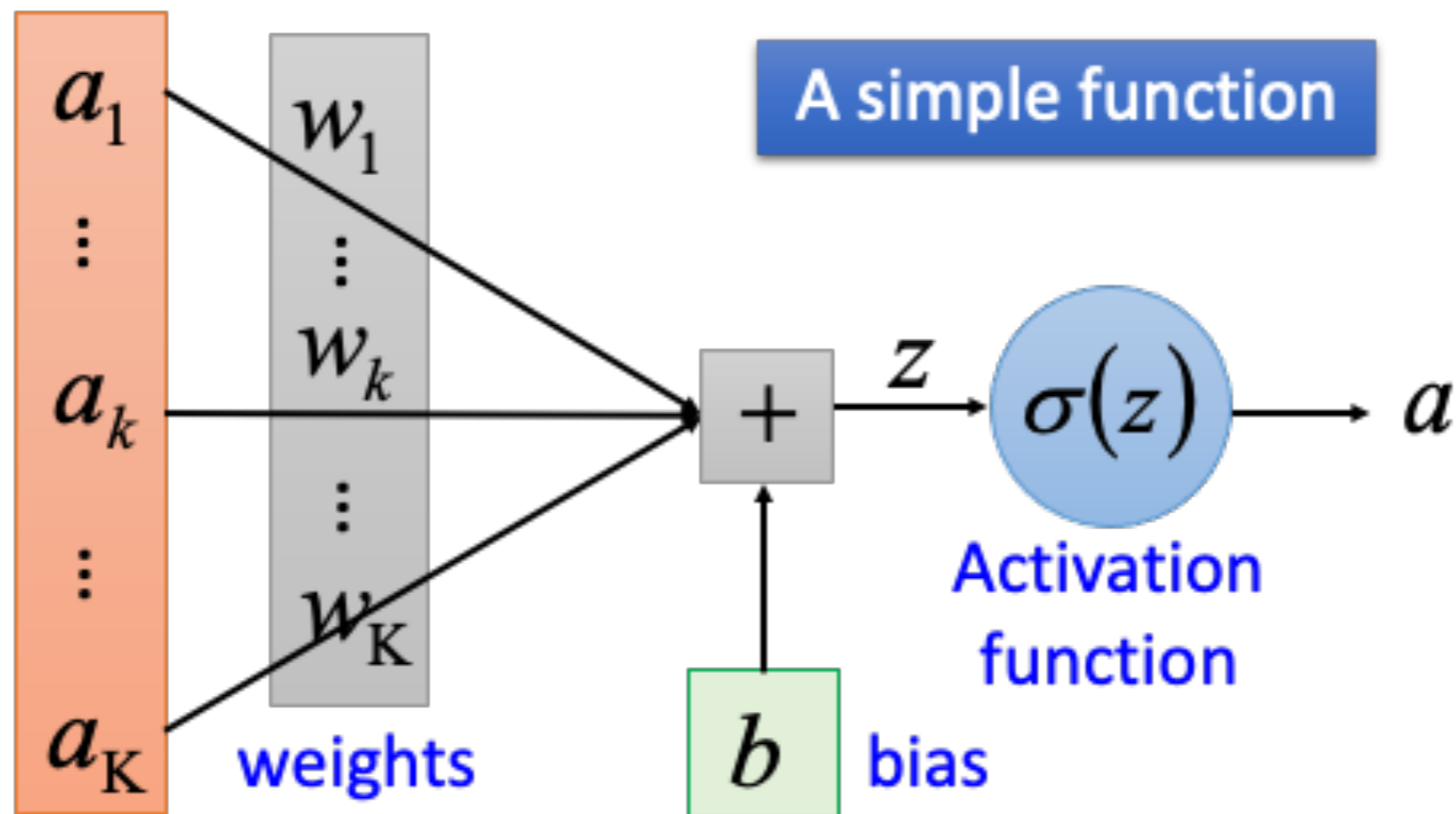


Neural Network

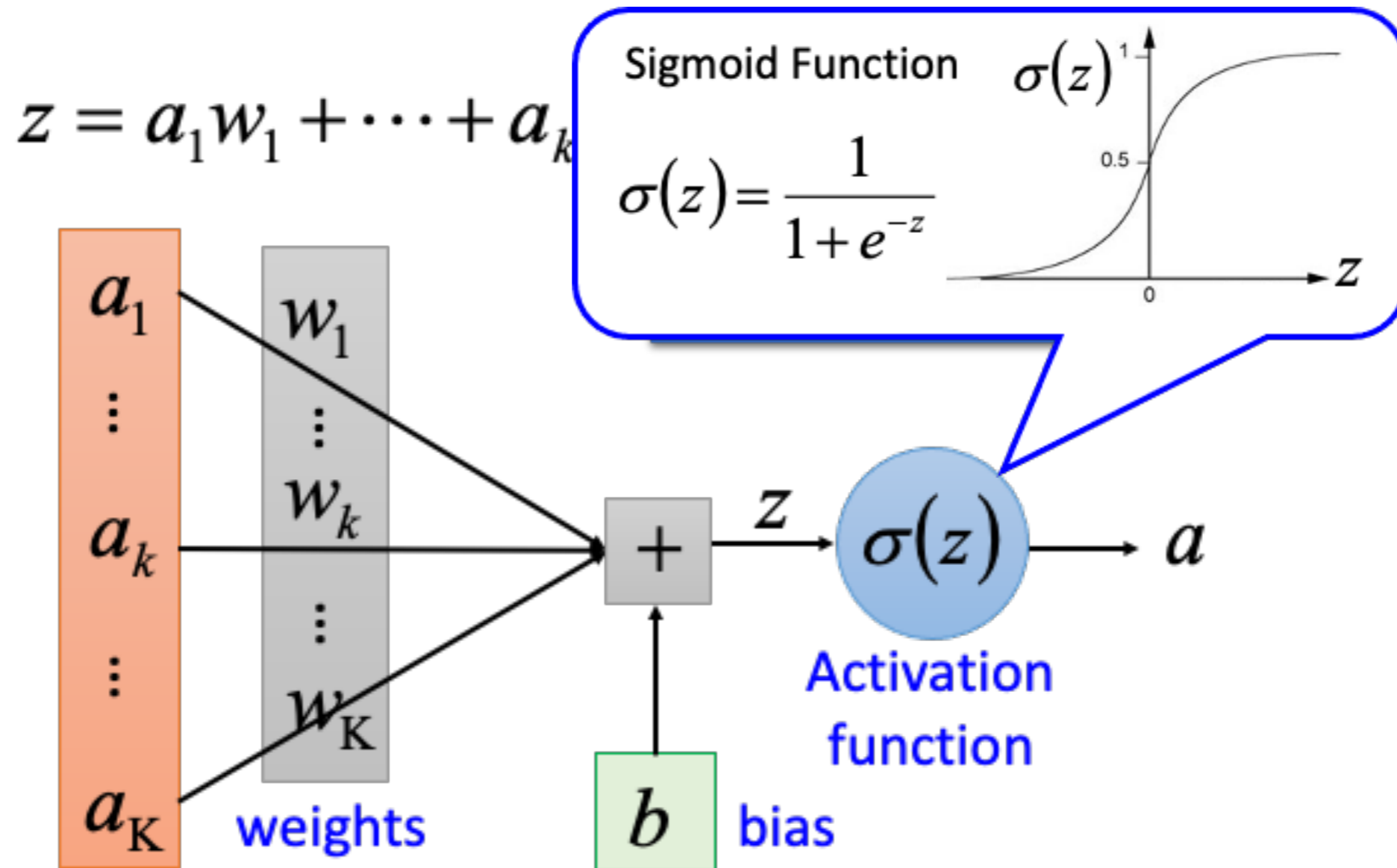


At Each Neuron

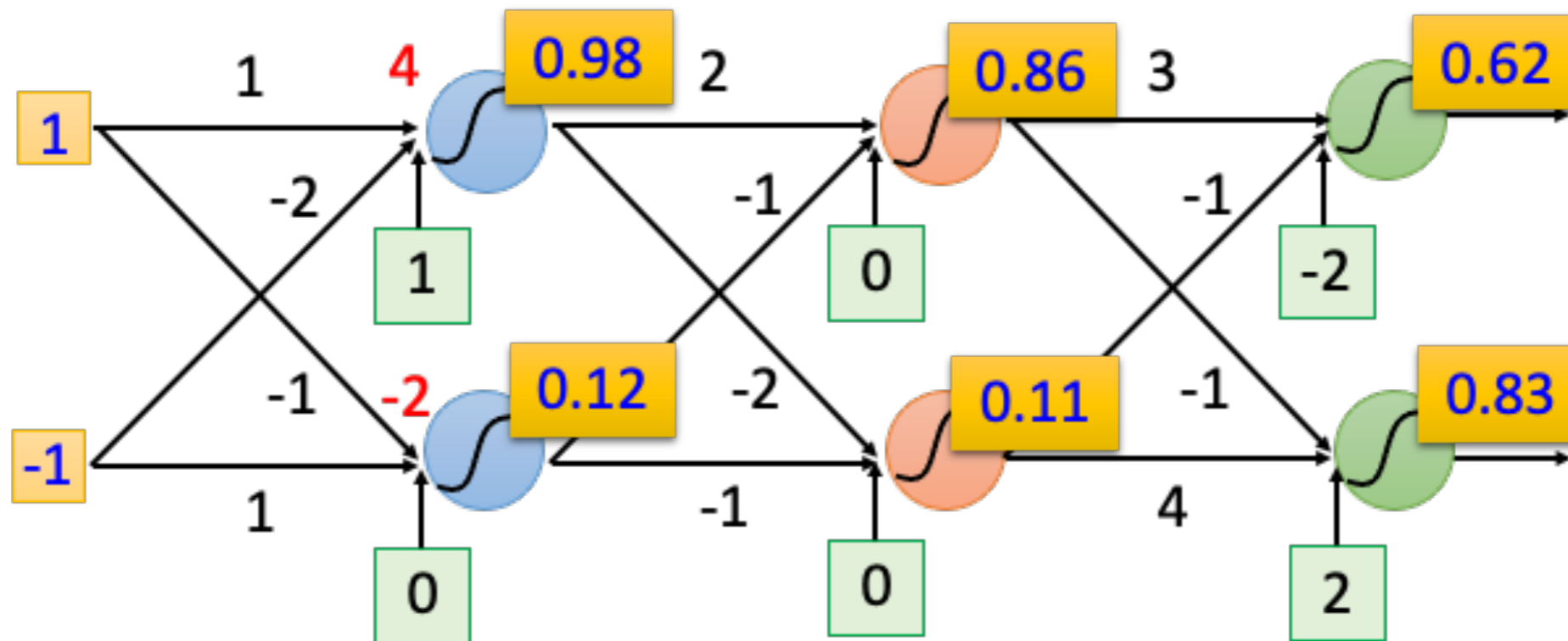
$$z = a_1 w_1 + \dots + a_k w_k + \dots + a_K w_K + b$$



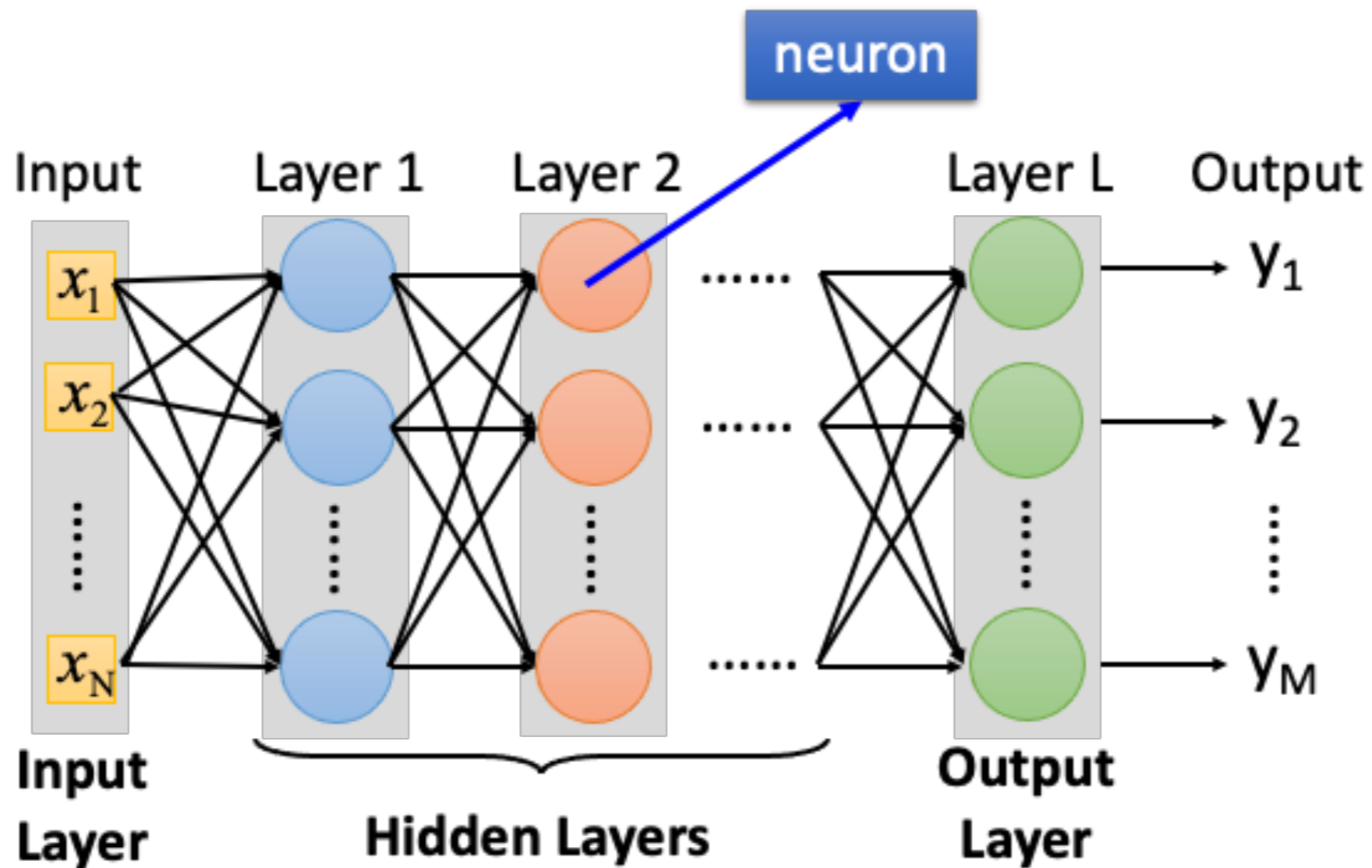
At Each Neuron



Fully Connected Neural Network



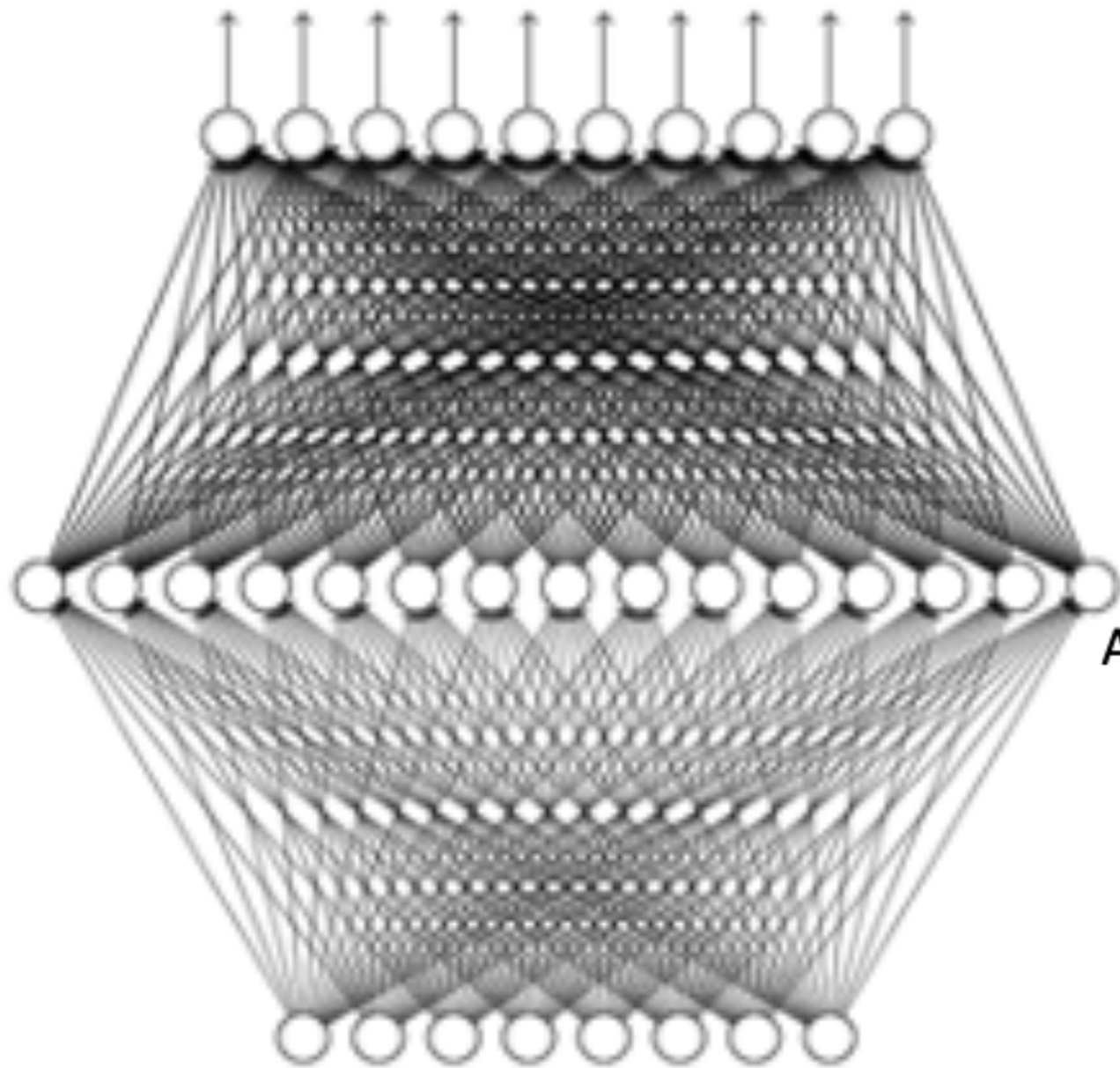
Fully Connected Neural Network



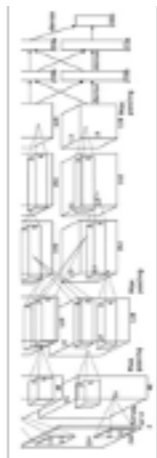
Deep means many hidden layers

Deep Neural Network

Deep = Many hidden layers



8 layers



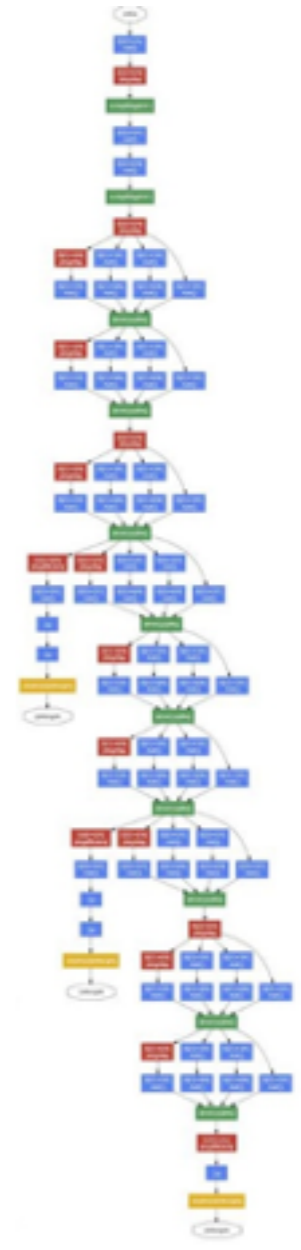
AlexNet (2012)

19 layers



VGG (2014)

22 layers



GoogleNet (2015)

152 layers

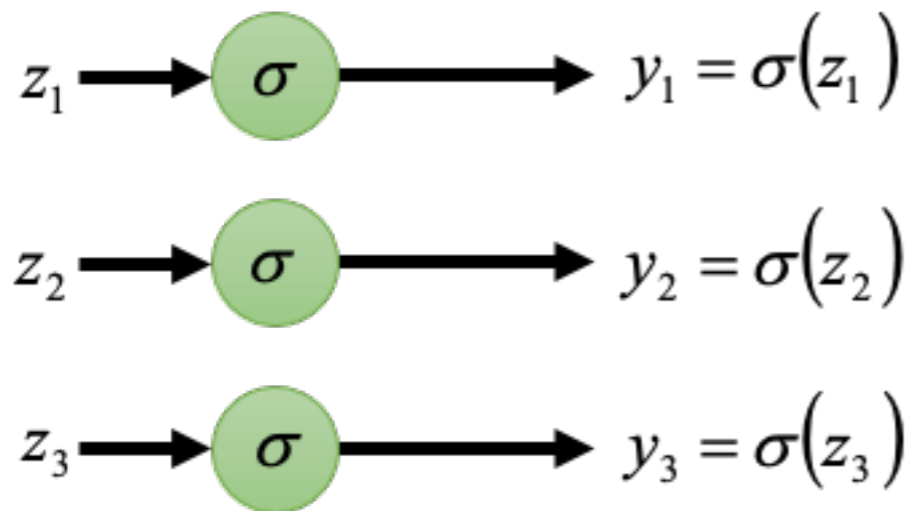


ResidualNet (2015)

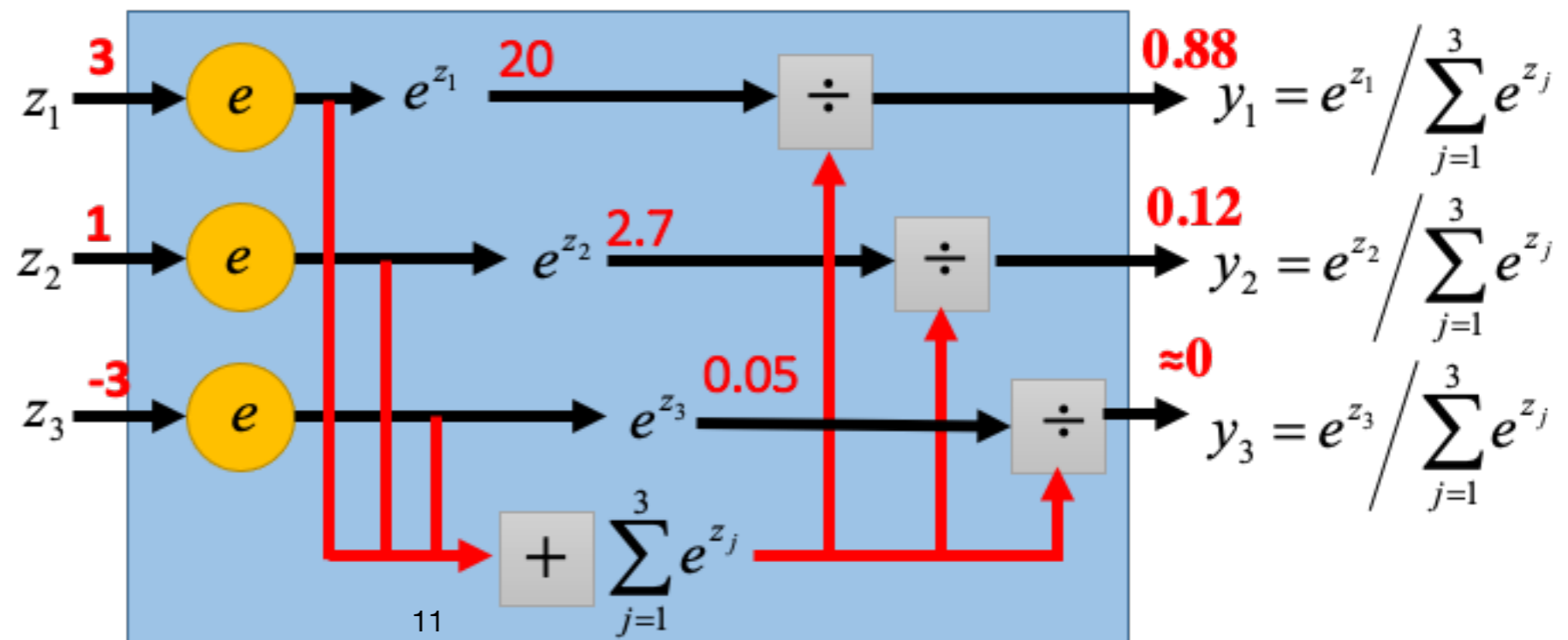
Using multiple layers of neurons to represent some functions

Output Layer

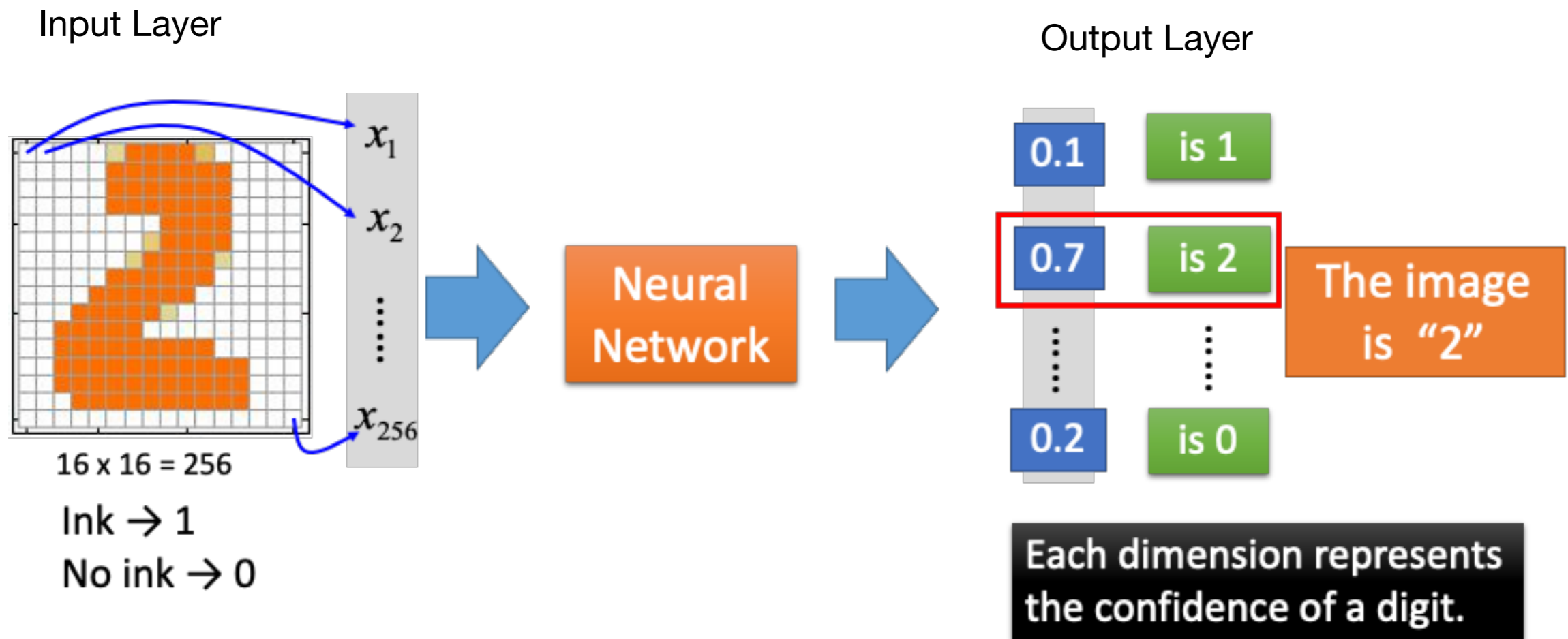
Ordinary Layer



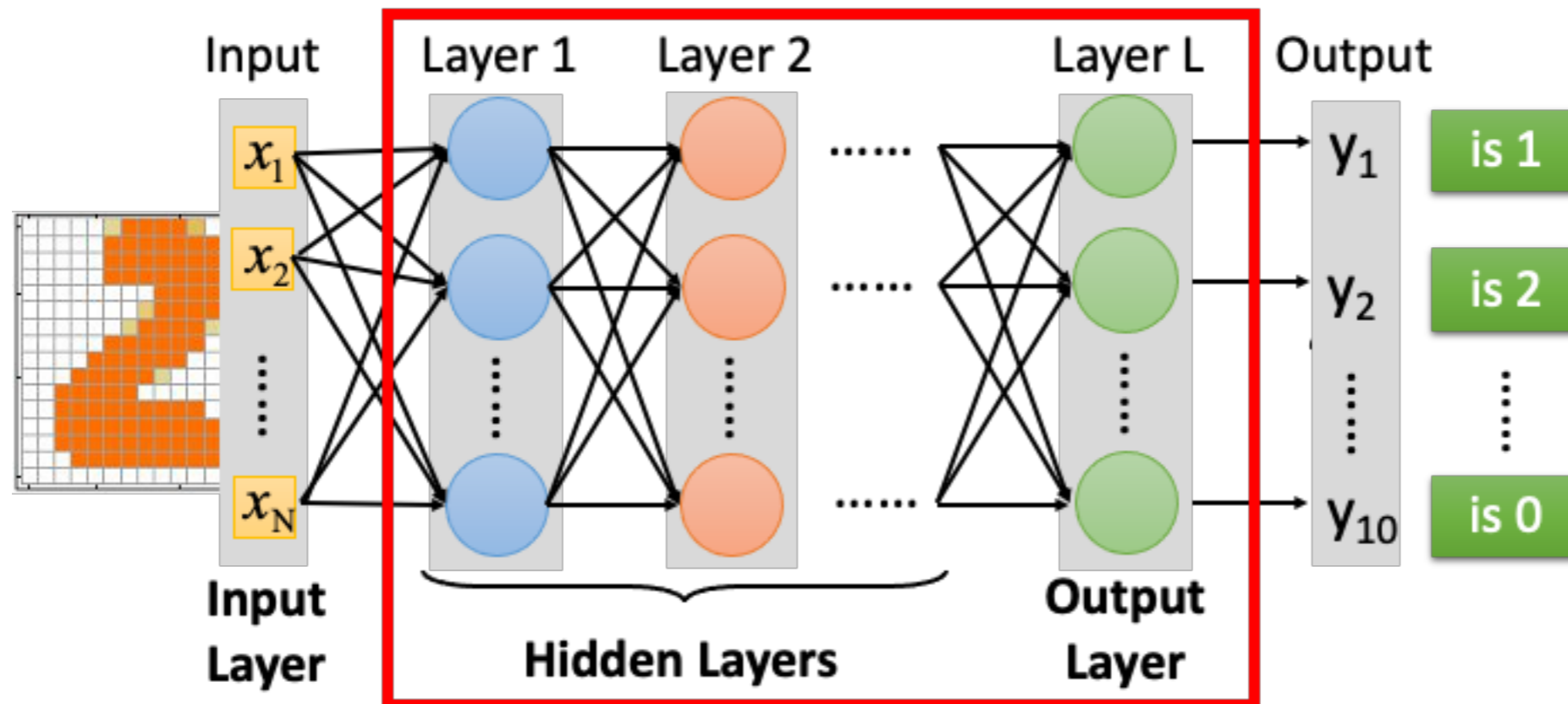
Softmax Layer



An Example



An Example

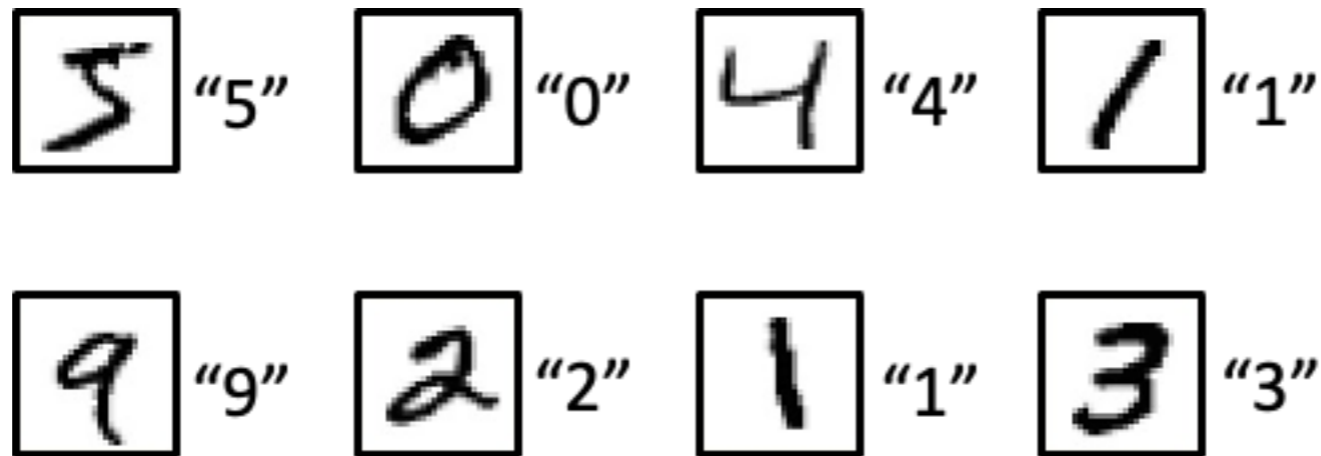


Needs to determine the network structure

How many layers?

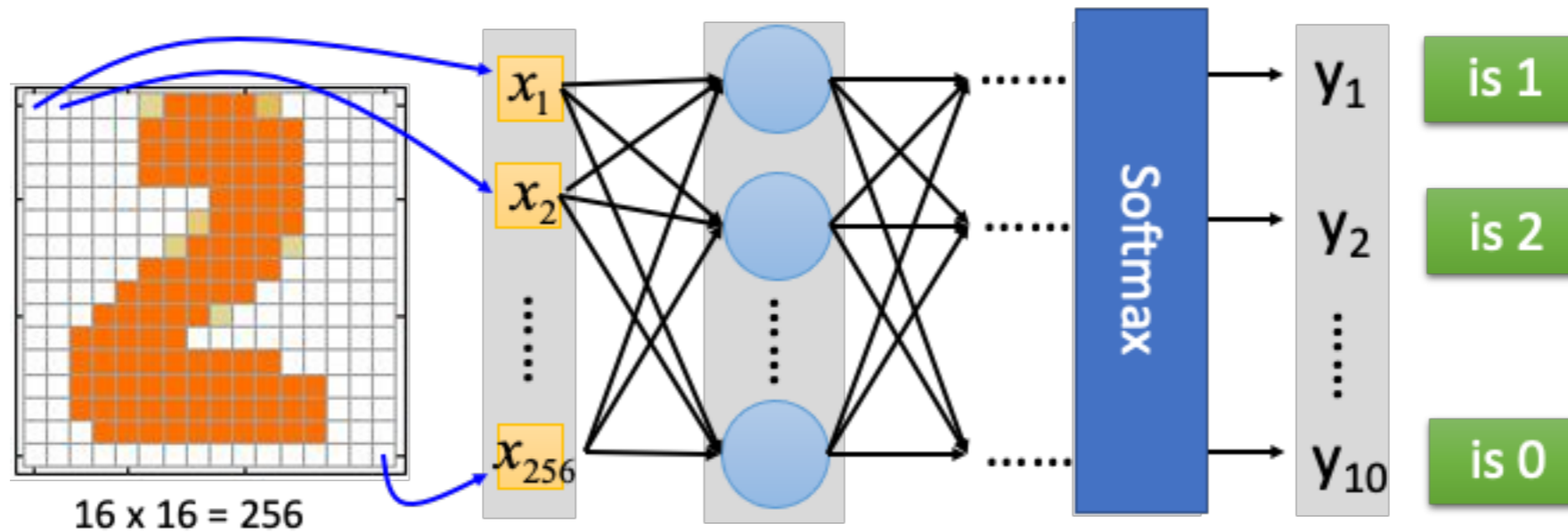
How many neurons for each layer?

Learning Target




The learning target is defined on
the training data.


Learning Target



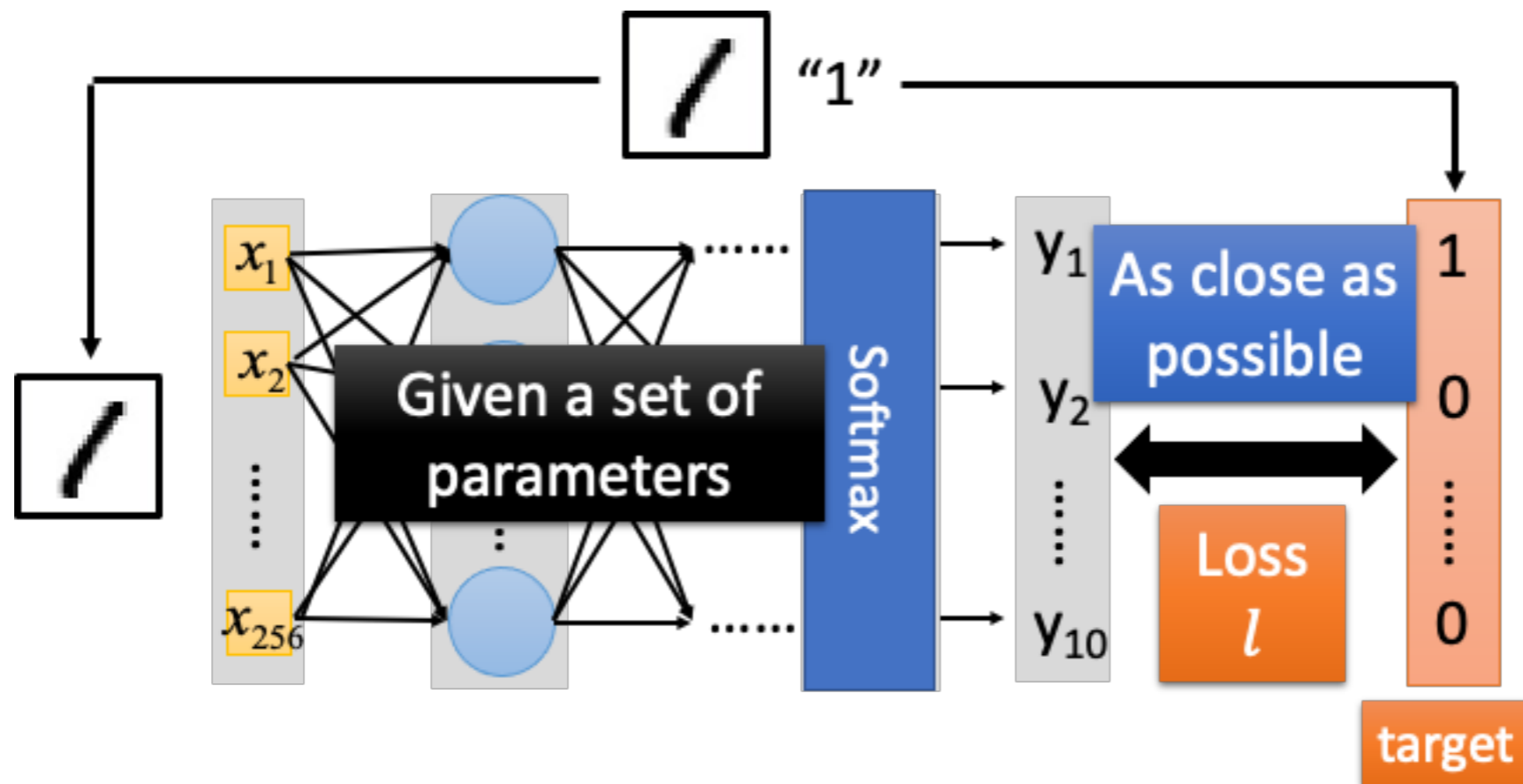
Ink \rightarrow 1
No ink \rightarrow 0

The learning target is

Input:  \rightarrow y_1 has the maximum value

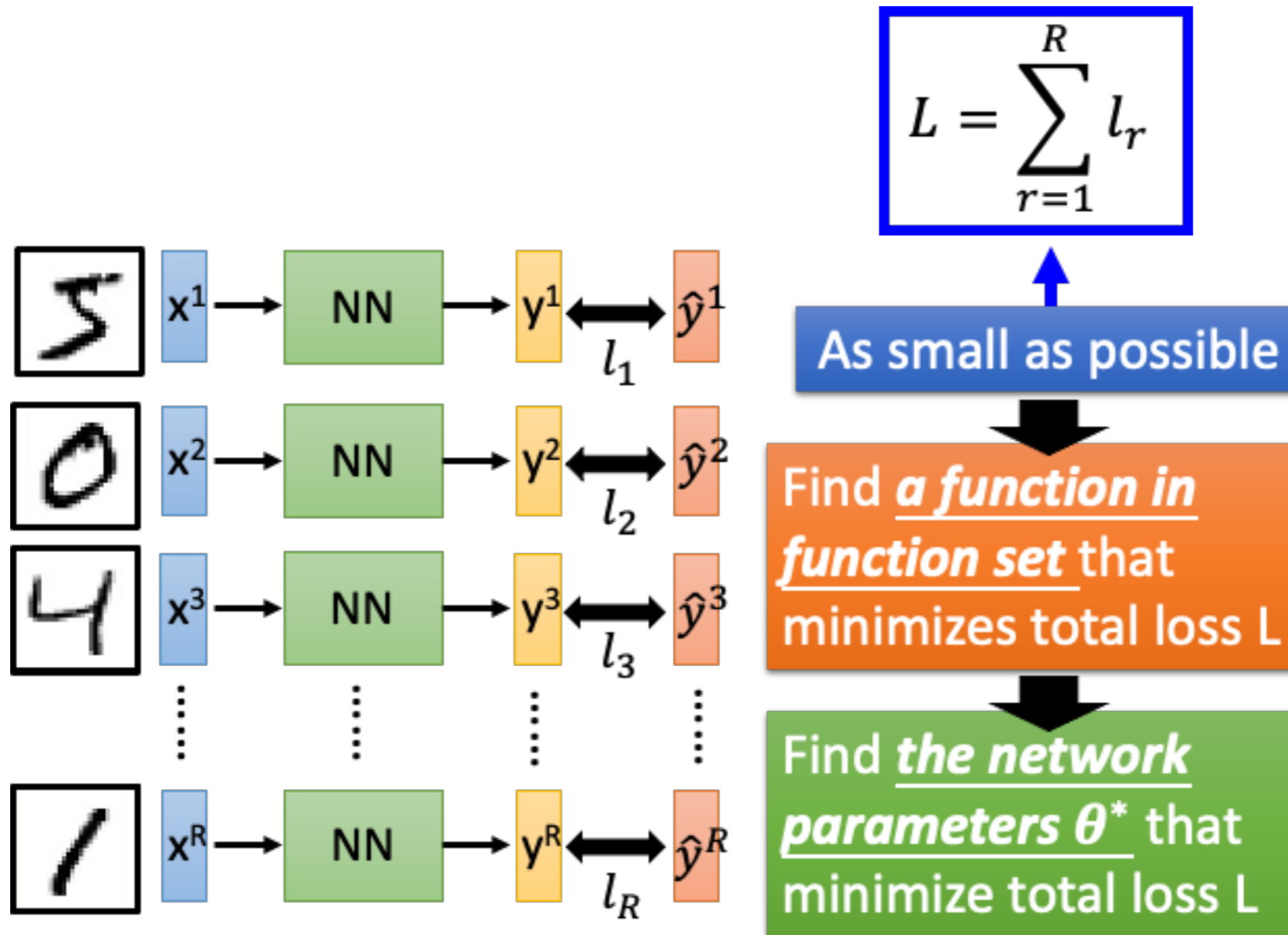
Input:  \rightarrow y_2 has the maximum value

Good Function = Loss as Small as Possible



Loss can be square error or cross entropy between the output and the target

Total Loss



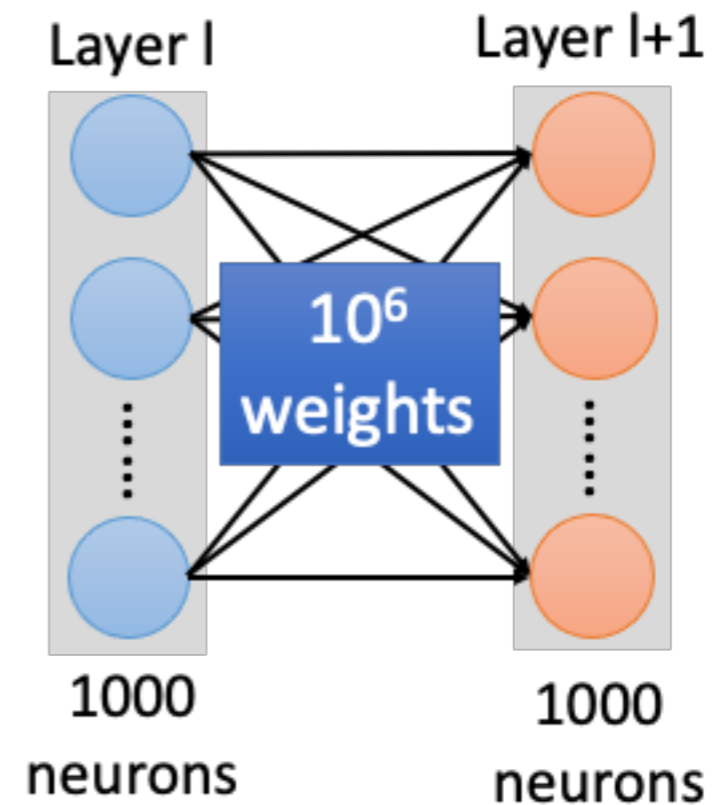
Total Loss

Find network parameters θ^* that minimize total loss L

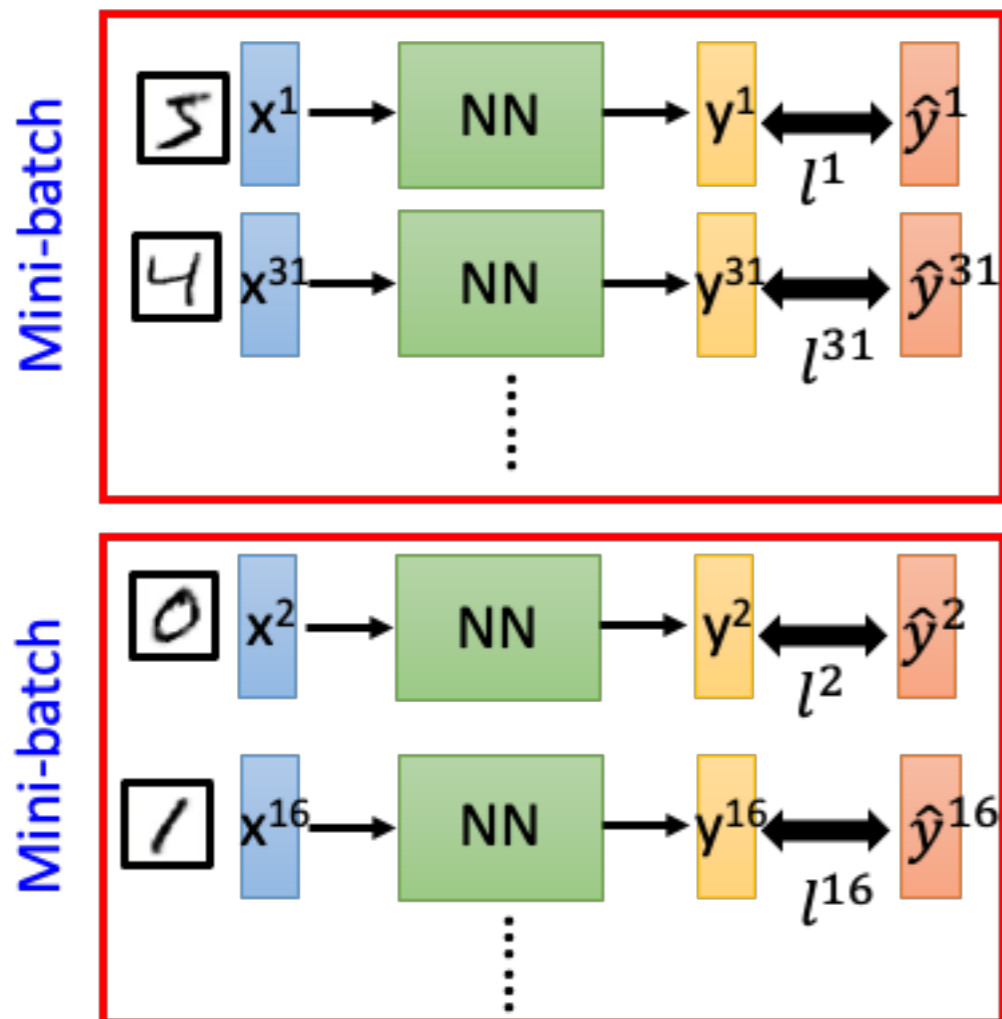
Enumerate all possible values

Network parameters $\theta =$
 $\{w_1, w_2, w_3, \dots, b_1, b_2, b_3, \dots\}$

Millions of parameters



Mini-batch



➤ Randomly initialize network parameters

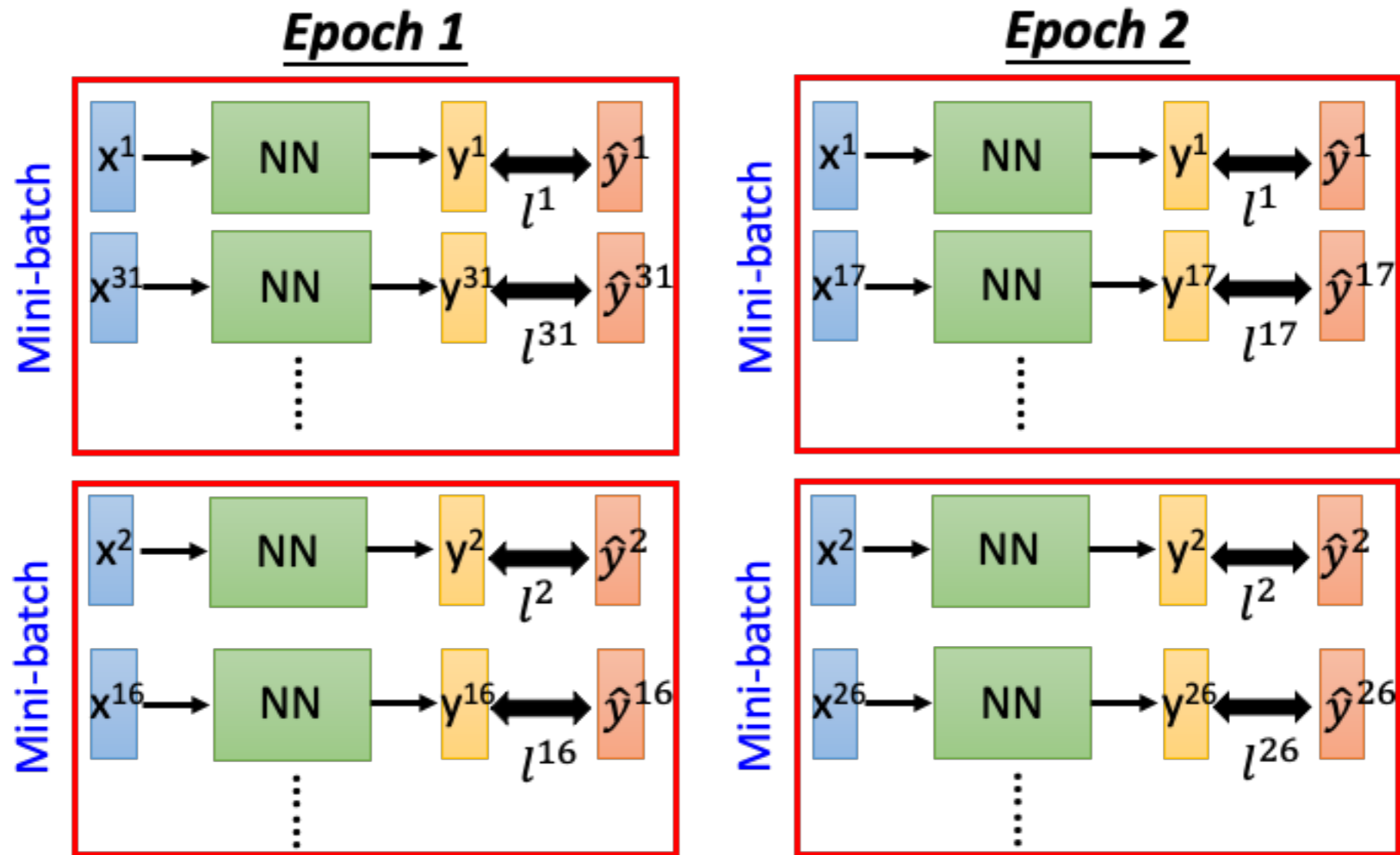
- Pick the 1st batch
 $L' = l^1 + l^{31} + \dots$
Update parameters once
- Pick the 2nd batch
 $L'' = l^2 + l^{16} + \dots$
Update parameters once
- ⋮
- Until all mini-batches have been picked

one epoch

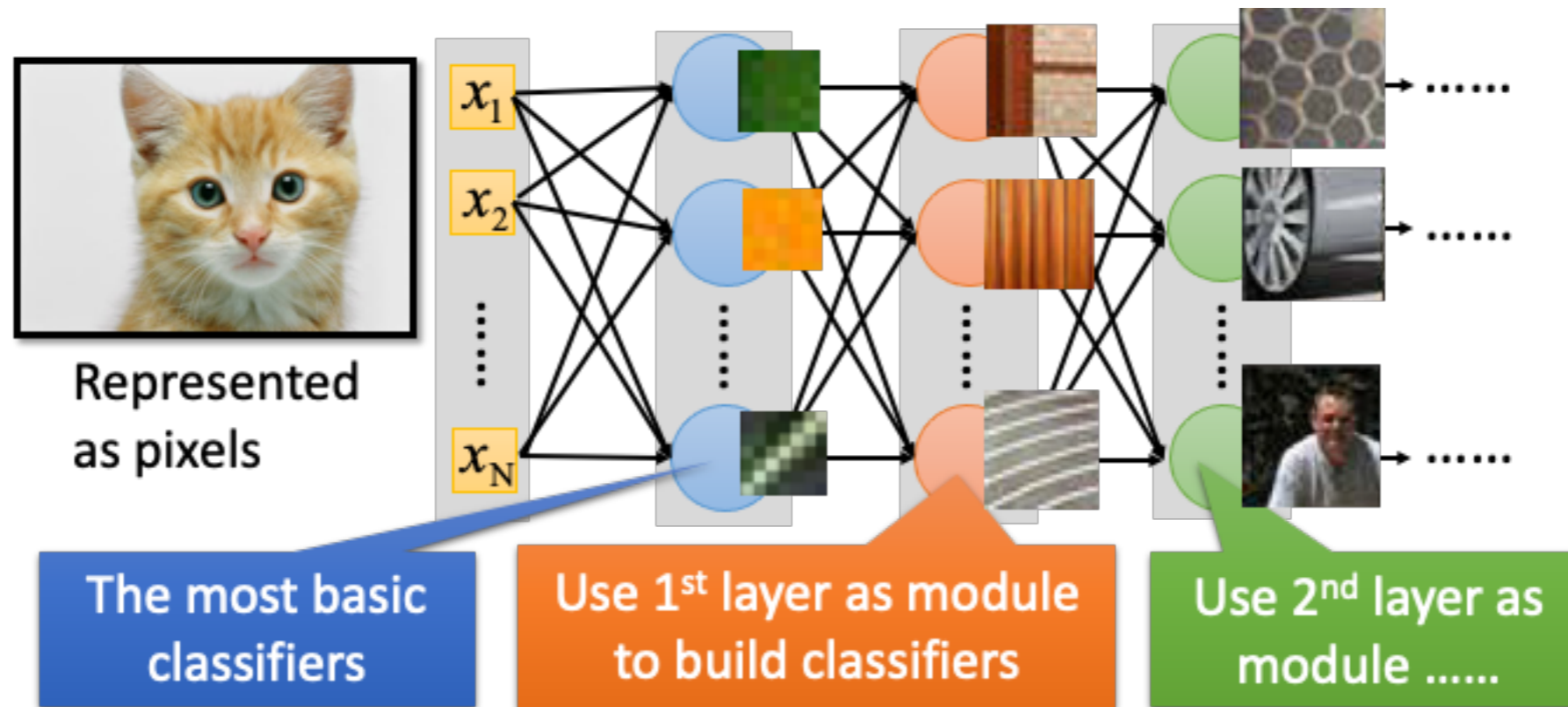
Repeat the above process

We do not really minimize total loss!

Mini-batch



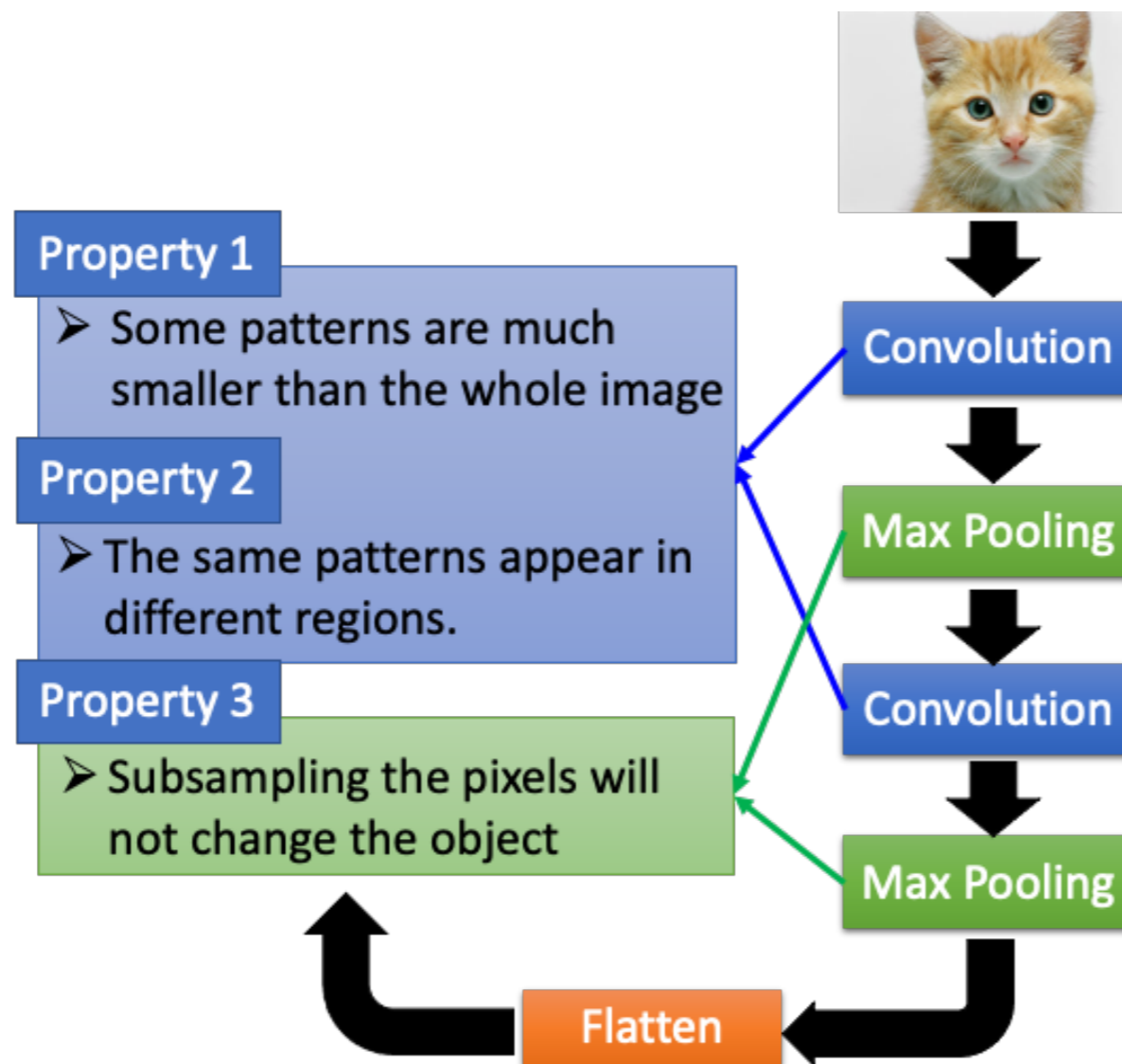
Convolutional Neural Network (CNN)



Each pixel as one input

Can we simplify the network by considering the properties of Images?

Convolutional Neural Network (CNN)



We can subsample the pixels to make image smaller

Less parameters for the network to process the image

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

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

Filter1
Matrix

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

Filter 2
Matrix

...
...
...
...

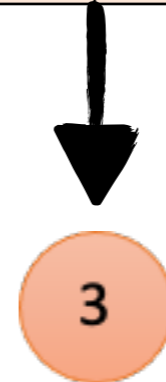
Each filter detects a small region (3x3) **Property 1**

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

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

Filter 1



CNN -Convolution

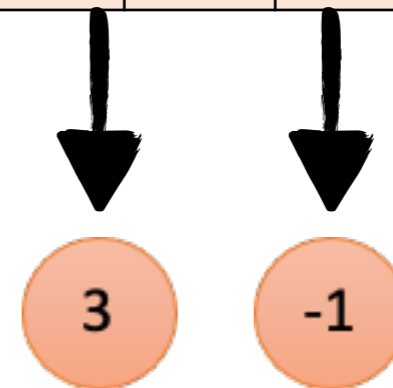
Stride =1

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

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



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

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

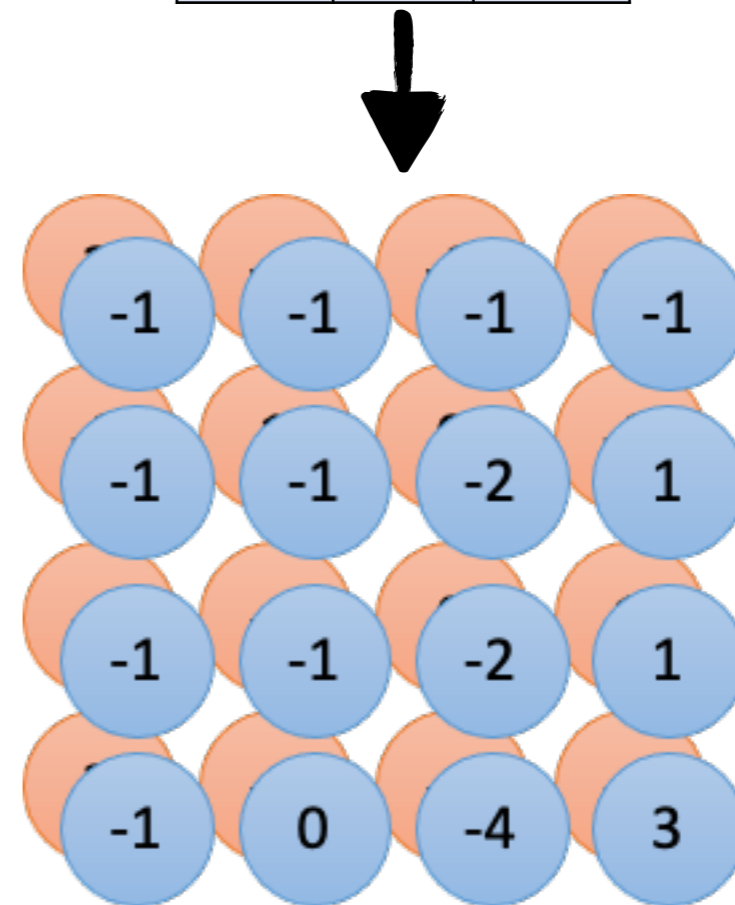
Property 2: Same patterns appears at different region!

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

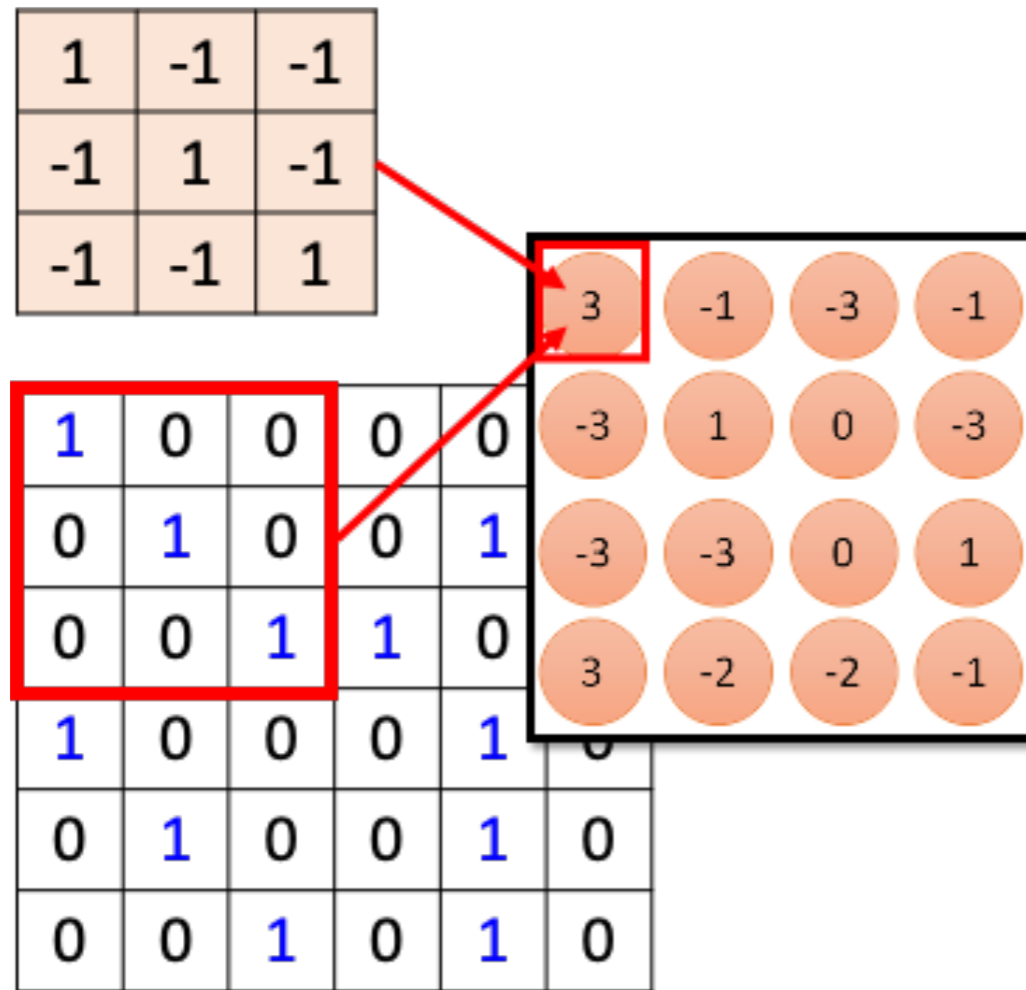
-1	1	-1
-1	1	-1
-1	1	-1

Filter2



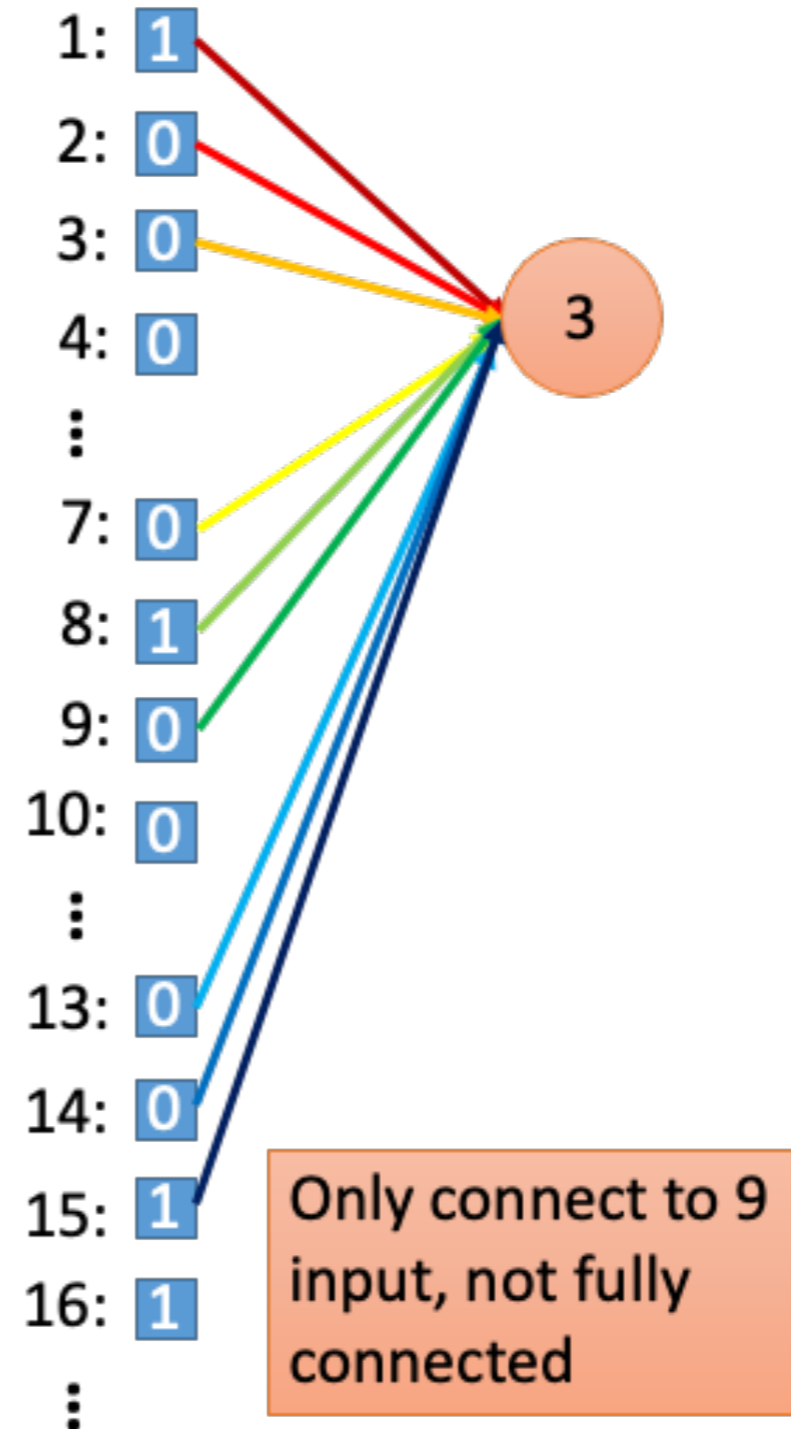
Property 2: Same patterns appears at different region!

CNN -Convolution

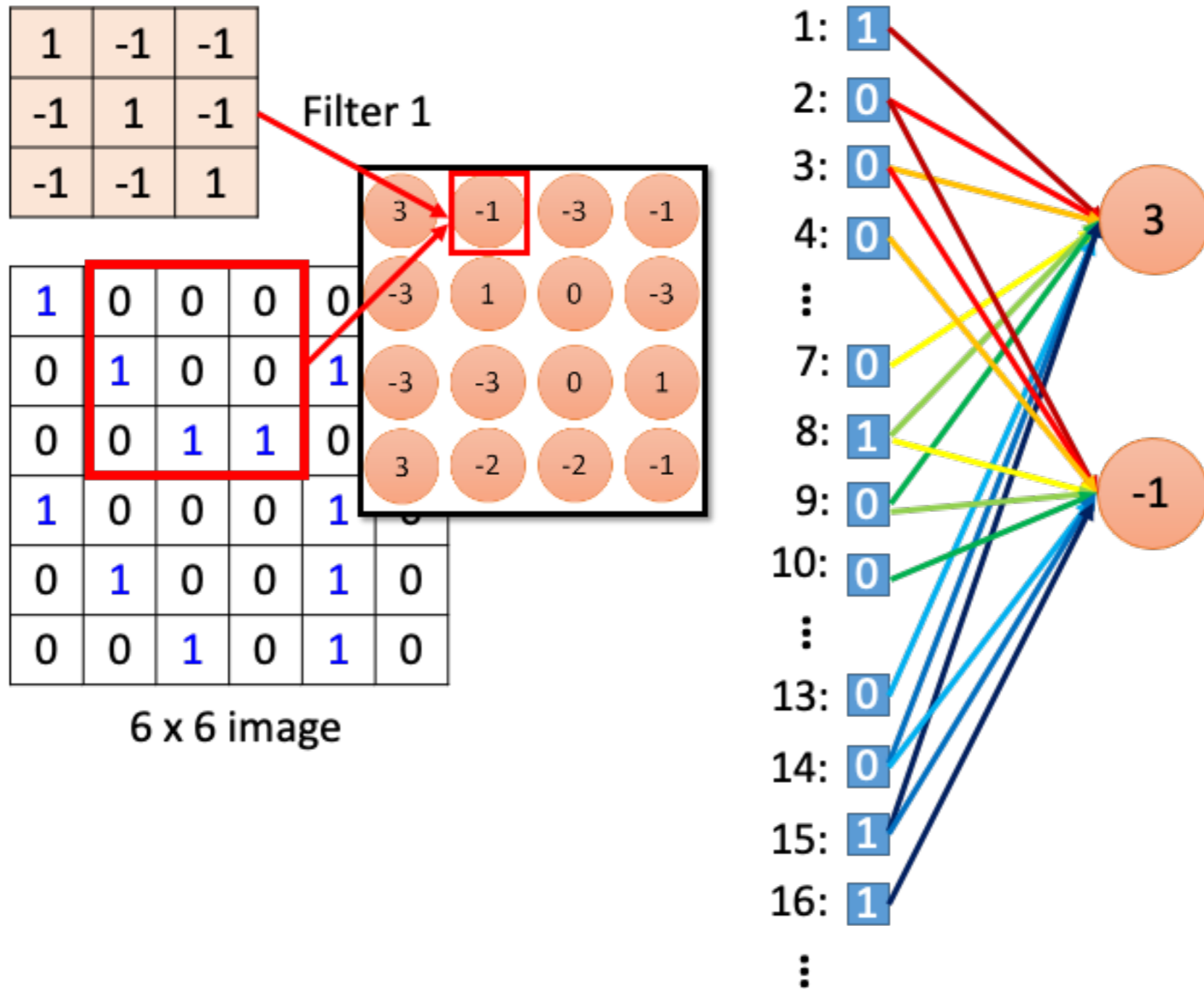


6 x 6 image

Less parameters!



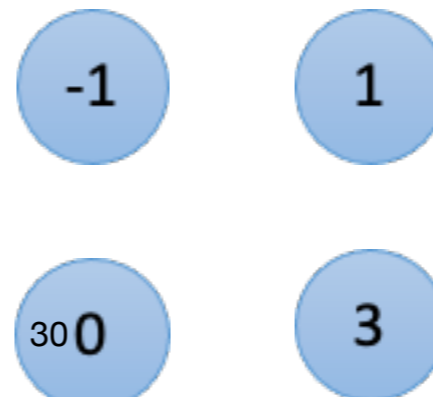
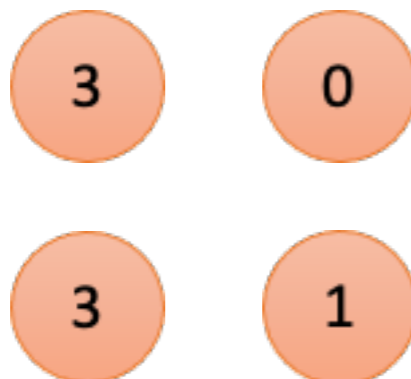
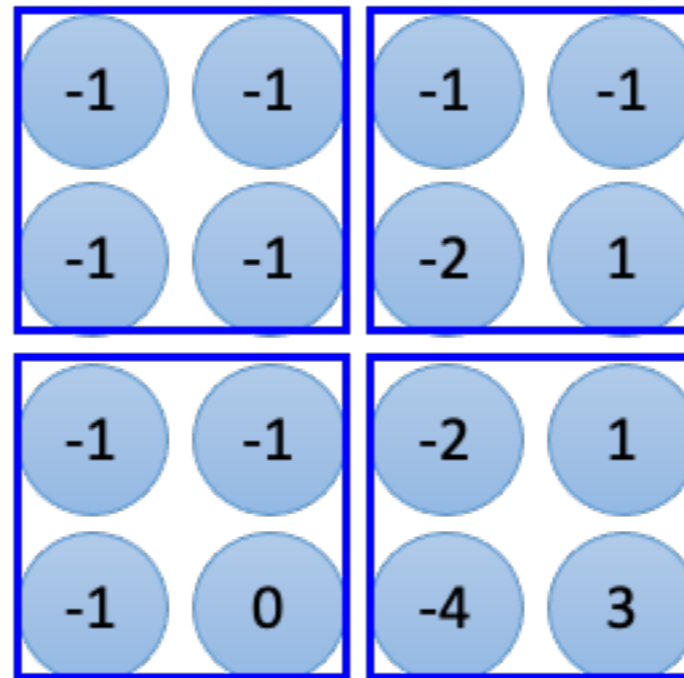
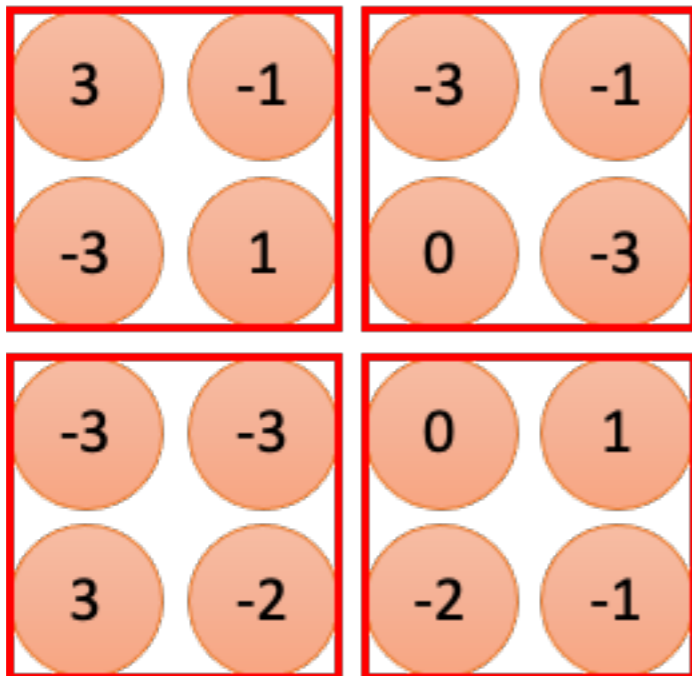
CNN -Convolution



CNN -Max Pooling

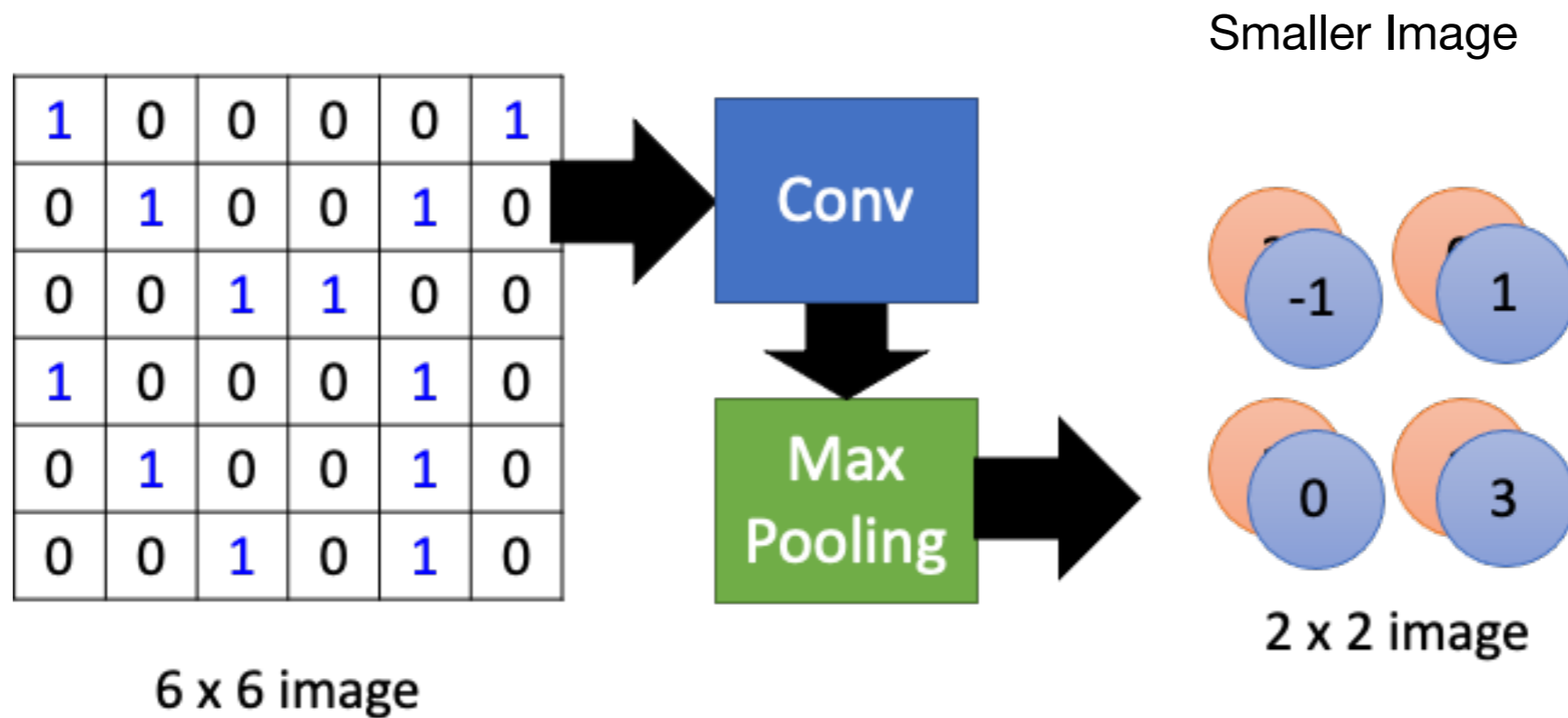
1	-1	-1
-1	1	-1
-1	-1	1

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

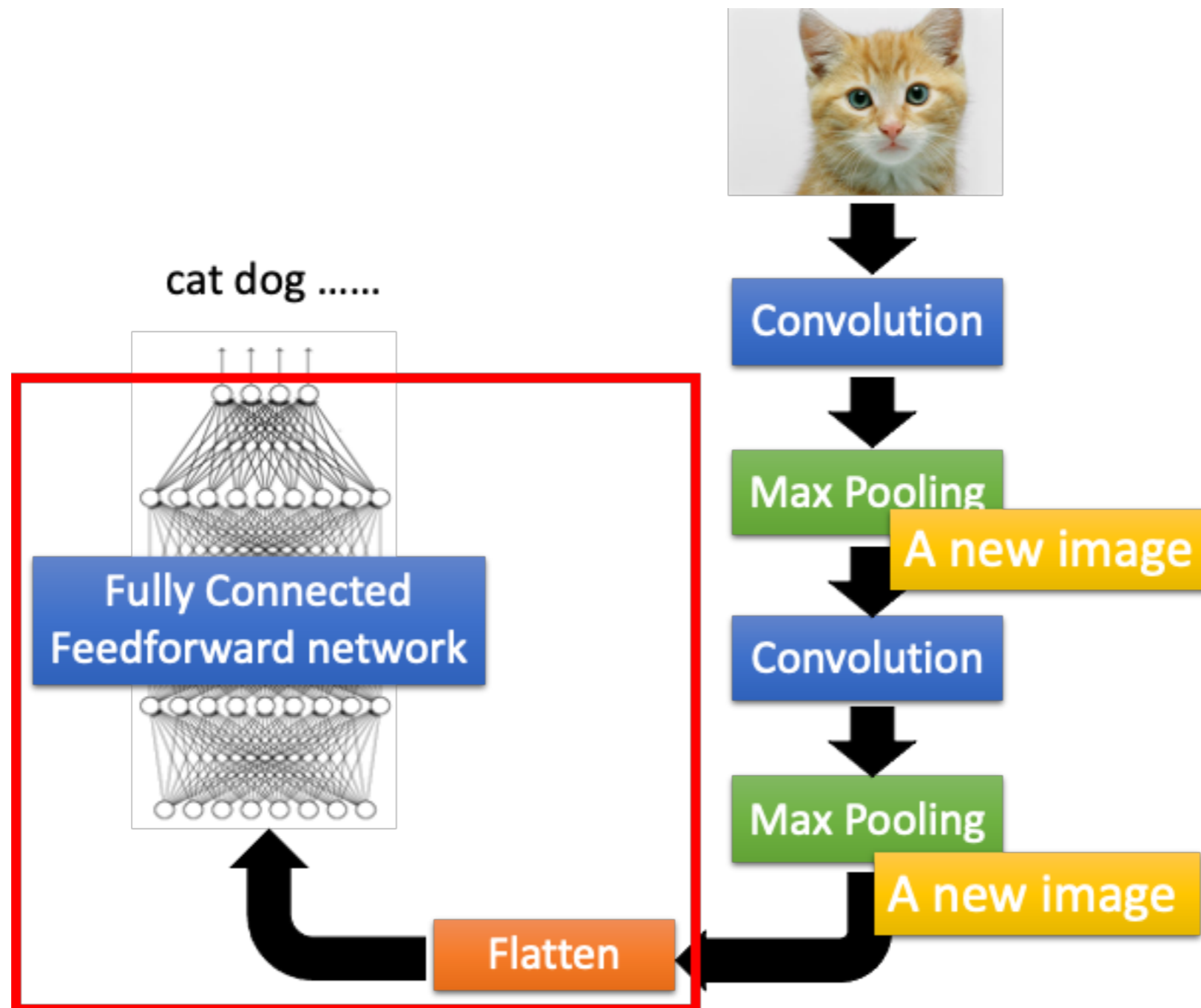


Take the dominant Features
Feature Dimensionality Reduction

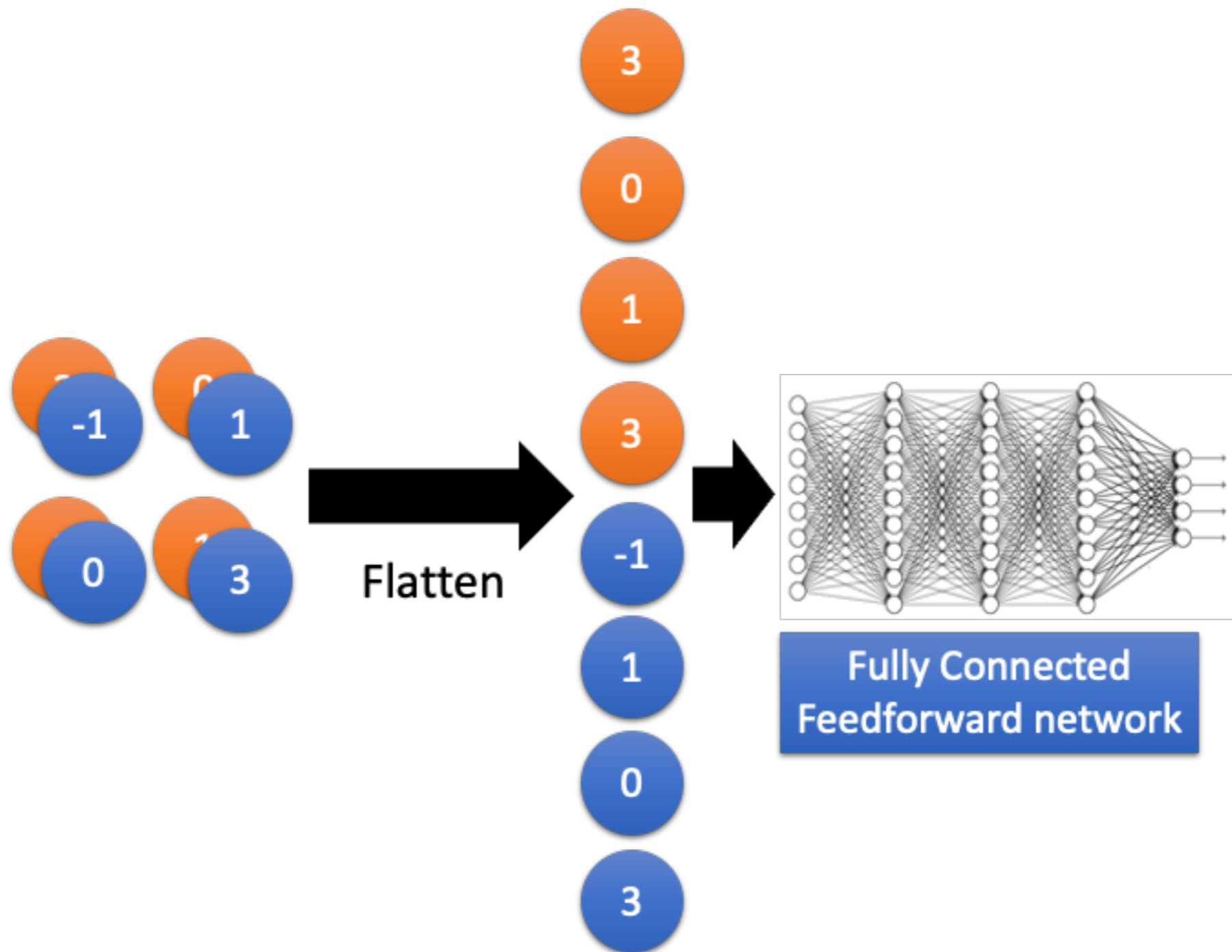
CNN -Convolution



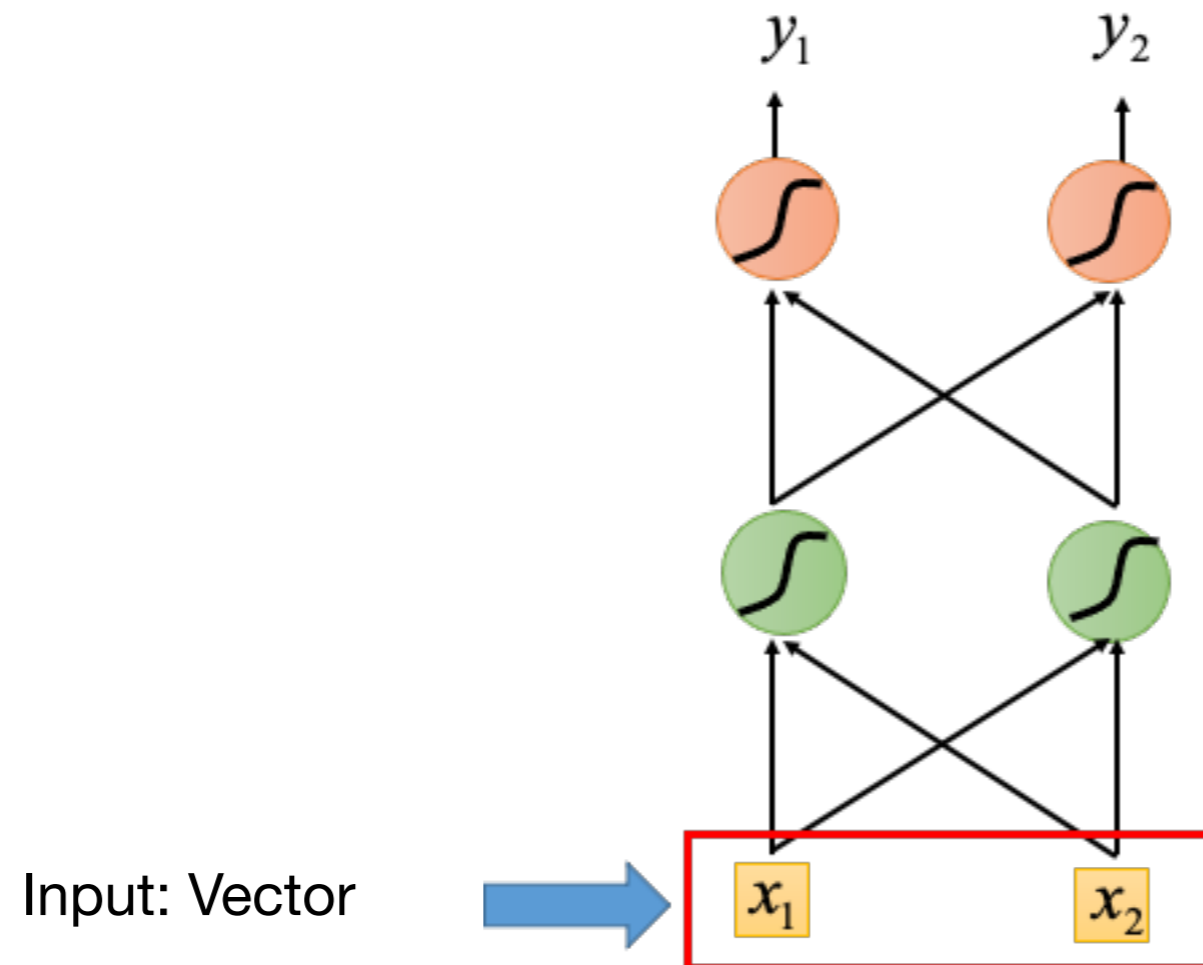
The Whole CNN



Flatten



Recurrent Neural Network



Word to Vector

1to N encoding  Embedding to high dimensional space

lexicon = {apple, bag, cat, dog, elephant}

apple = [1 0 0 0 0]

bag = [0 1 0 0 0]

cat = [0 0 1 0 0]

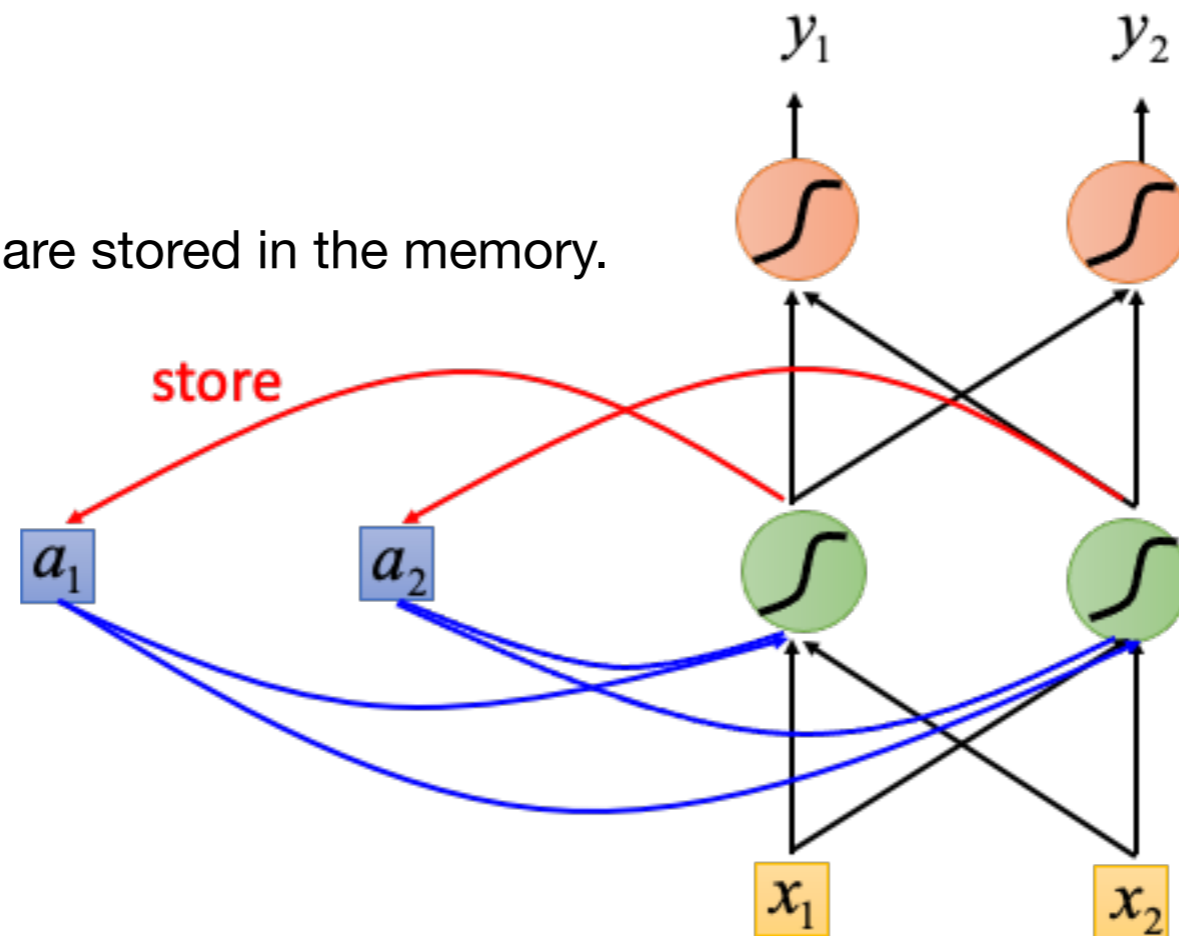
dog = [0 0 0 1 0]

elephant = [0 0 0 0 1]

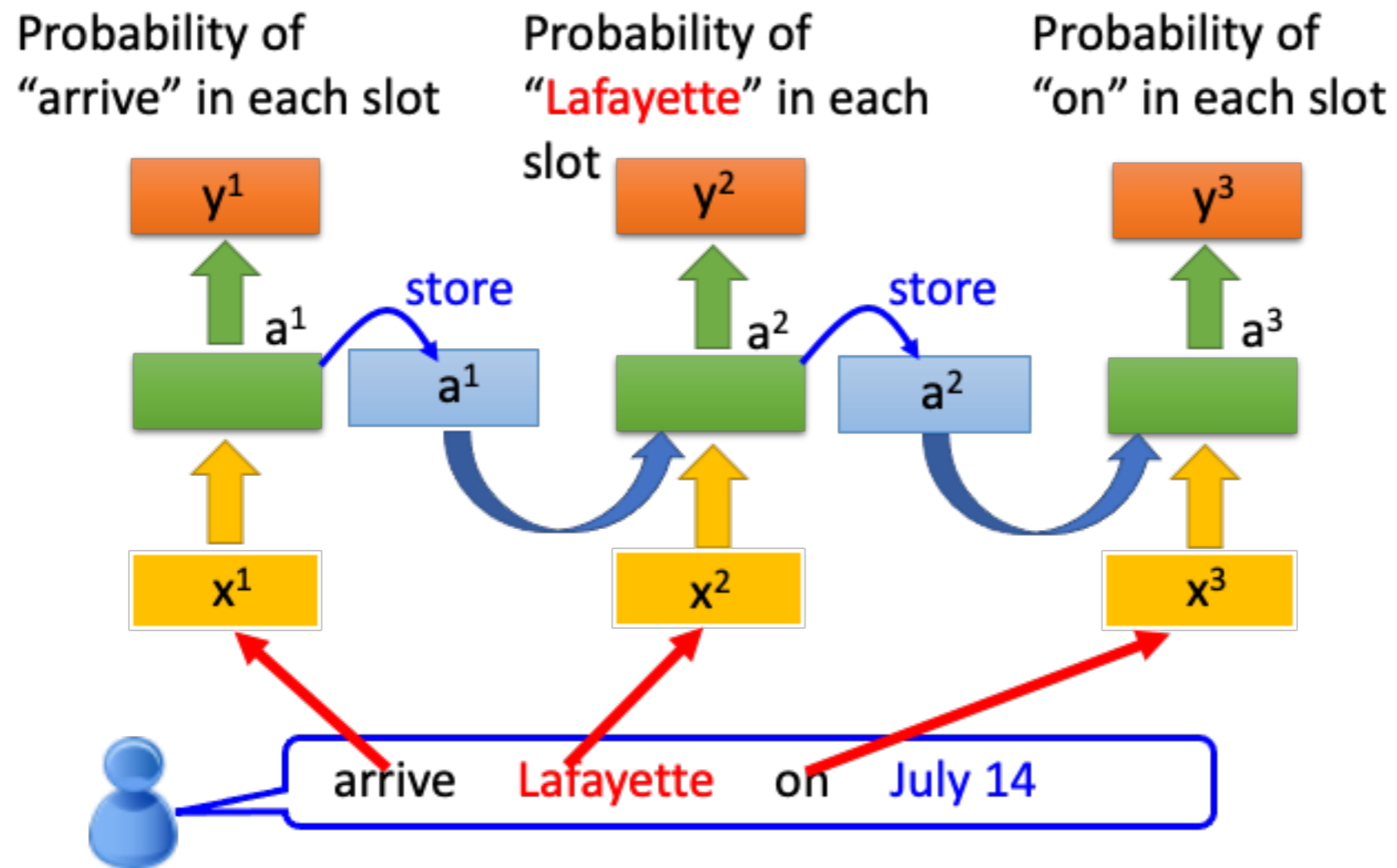
Popular techniques: Word2Vector, Node2Vector

RNN

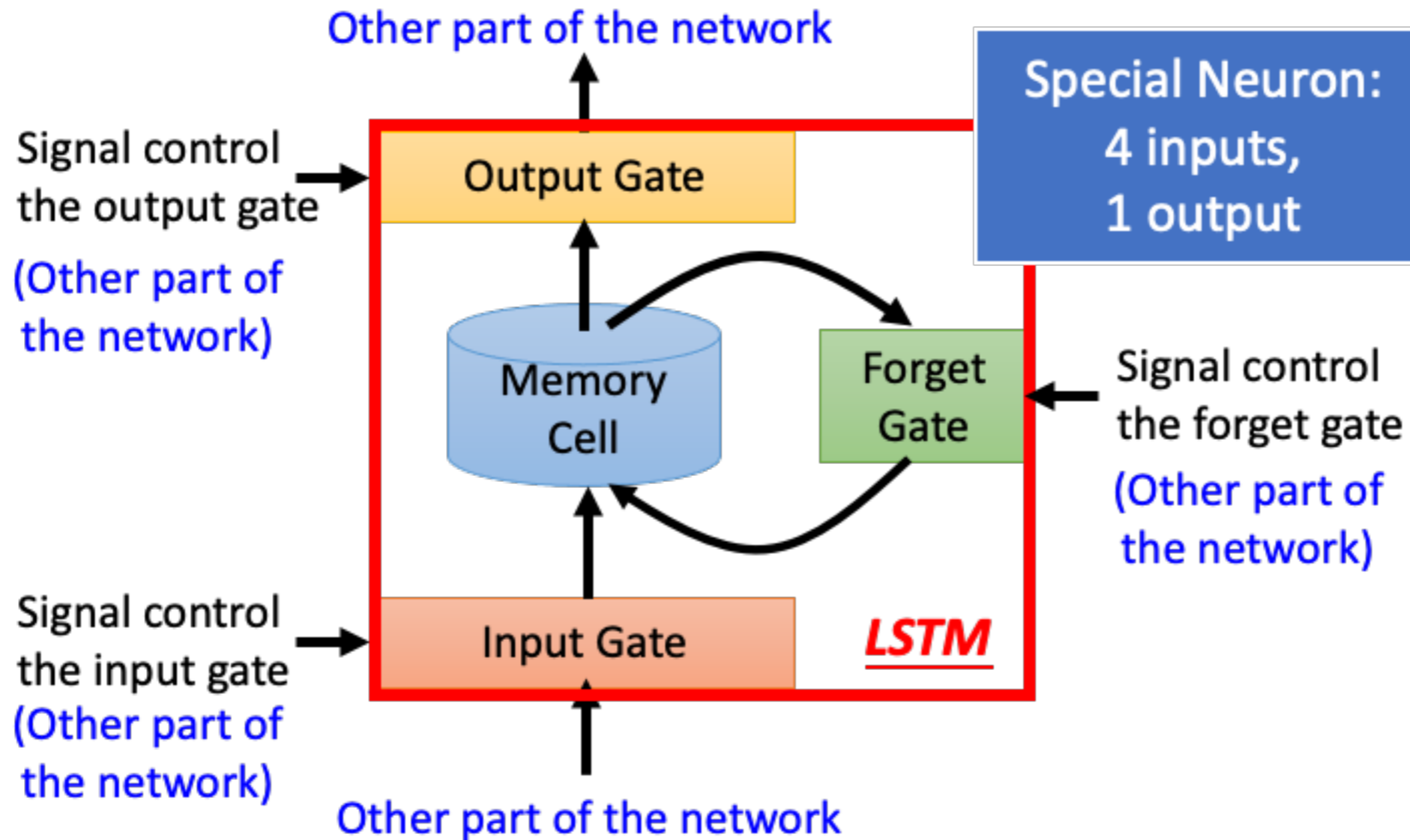
The output of hidden layer are stored in the memory.



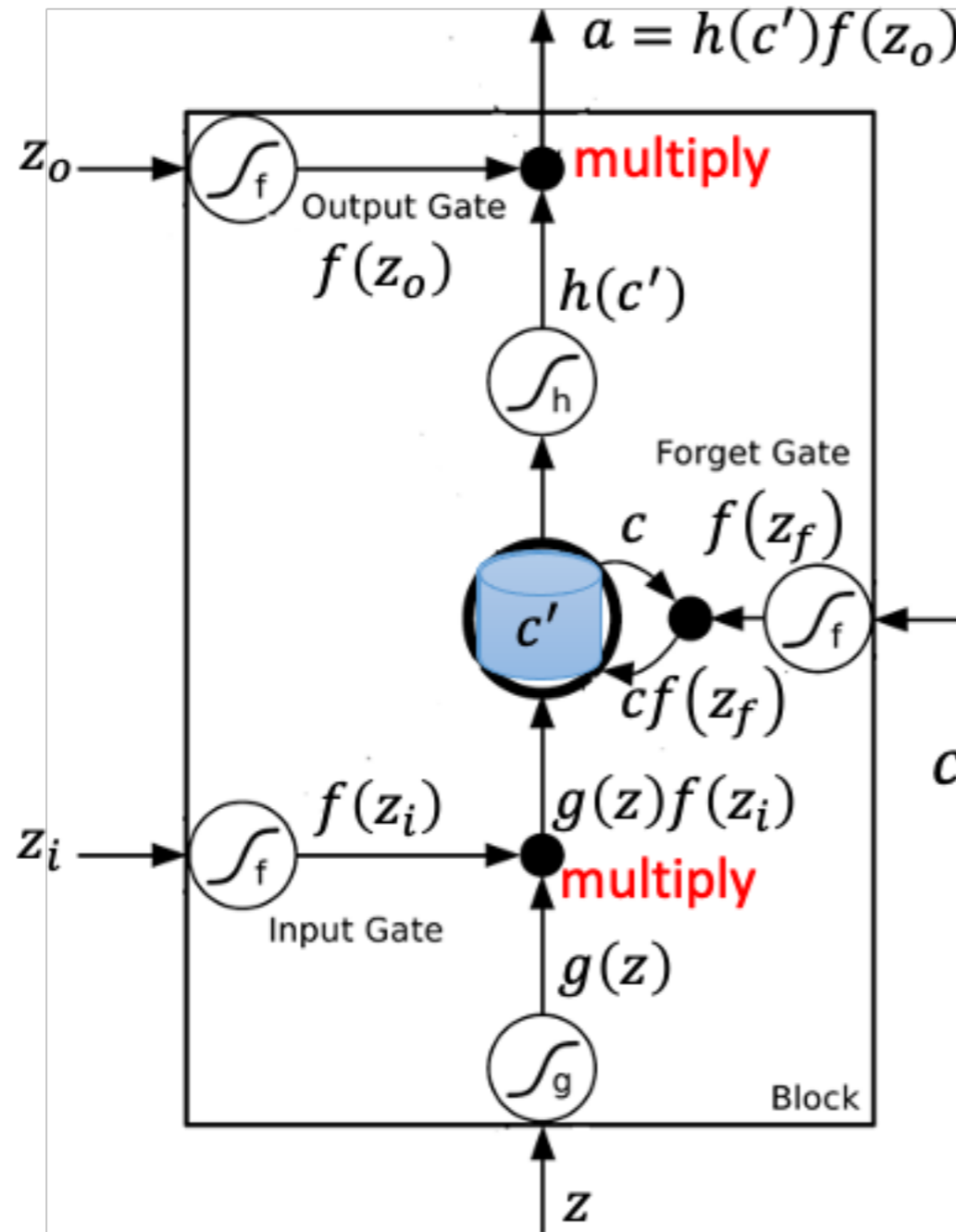
RNN



Long Short-term Memory (LSTM)



Long Short-term Memory (LSTM)



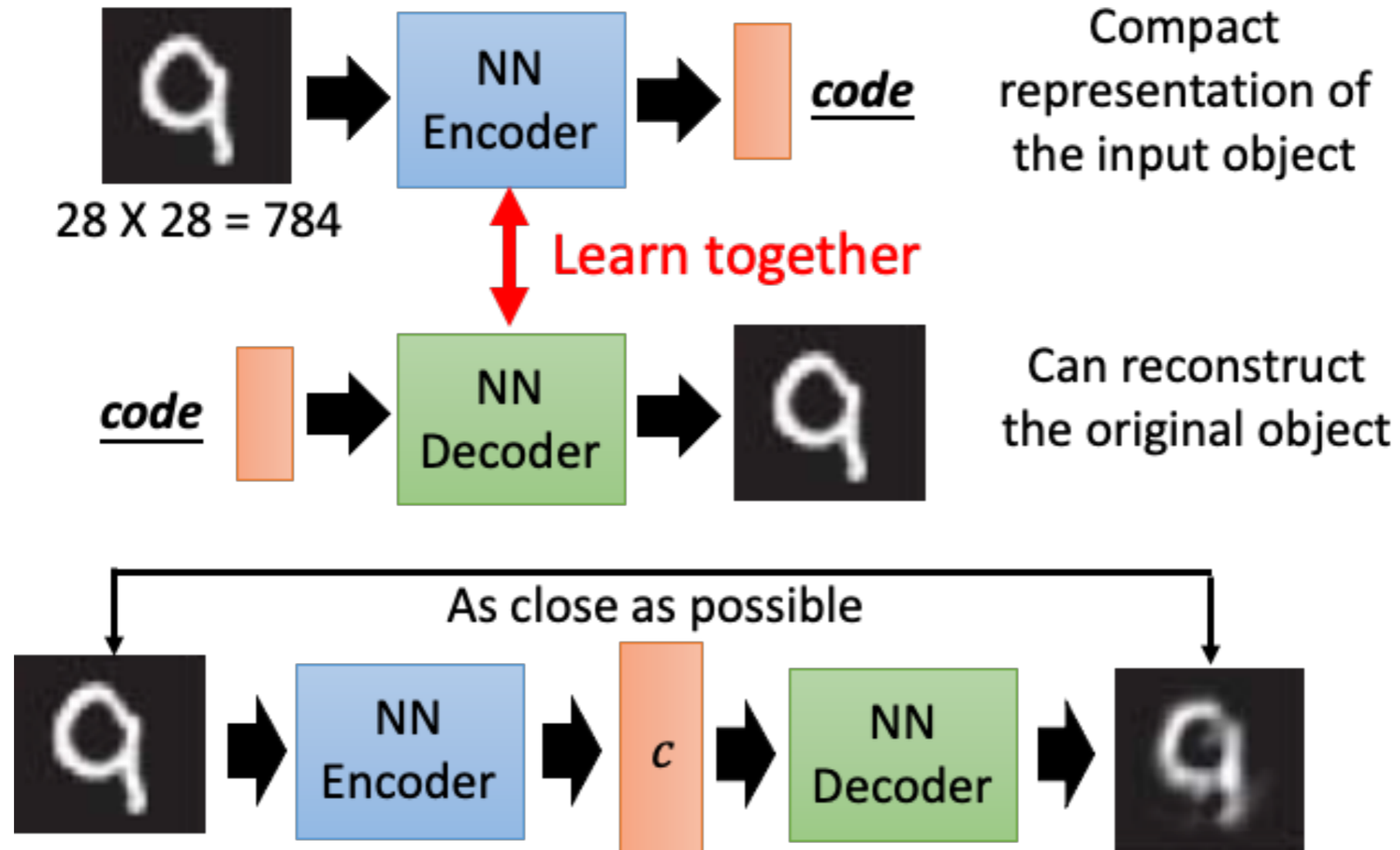
Activation function f is usually a sigmoid function

Between 0 and 1

Mimic open and close gate

$$c' = g(z)f(z_i) + cf(z_f)$$

Auto-Encoder



Deep Auto-Encoder

