Lecture 3
Neural Network and Deep Learning

Xu Yuan
University of Louisiana at Lafayette

Machine Learning ~ Looking for a Function

- Image recognition

$f($  $)$ → "Cat"

- Spam classification

$f($ "Tweets" $)$ → a spam message

- Weather prediction

$f($"*Observed Weather Conditions*"$)$ → Future weather condition

Machine Learning ~ Training Framework




Training Data

A set of functions (models) $f_1, f_2, \ldots$

Goodness of function f

Pick the "best" function f^*

Trained Model

Framework

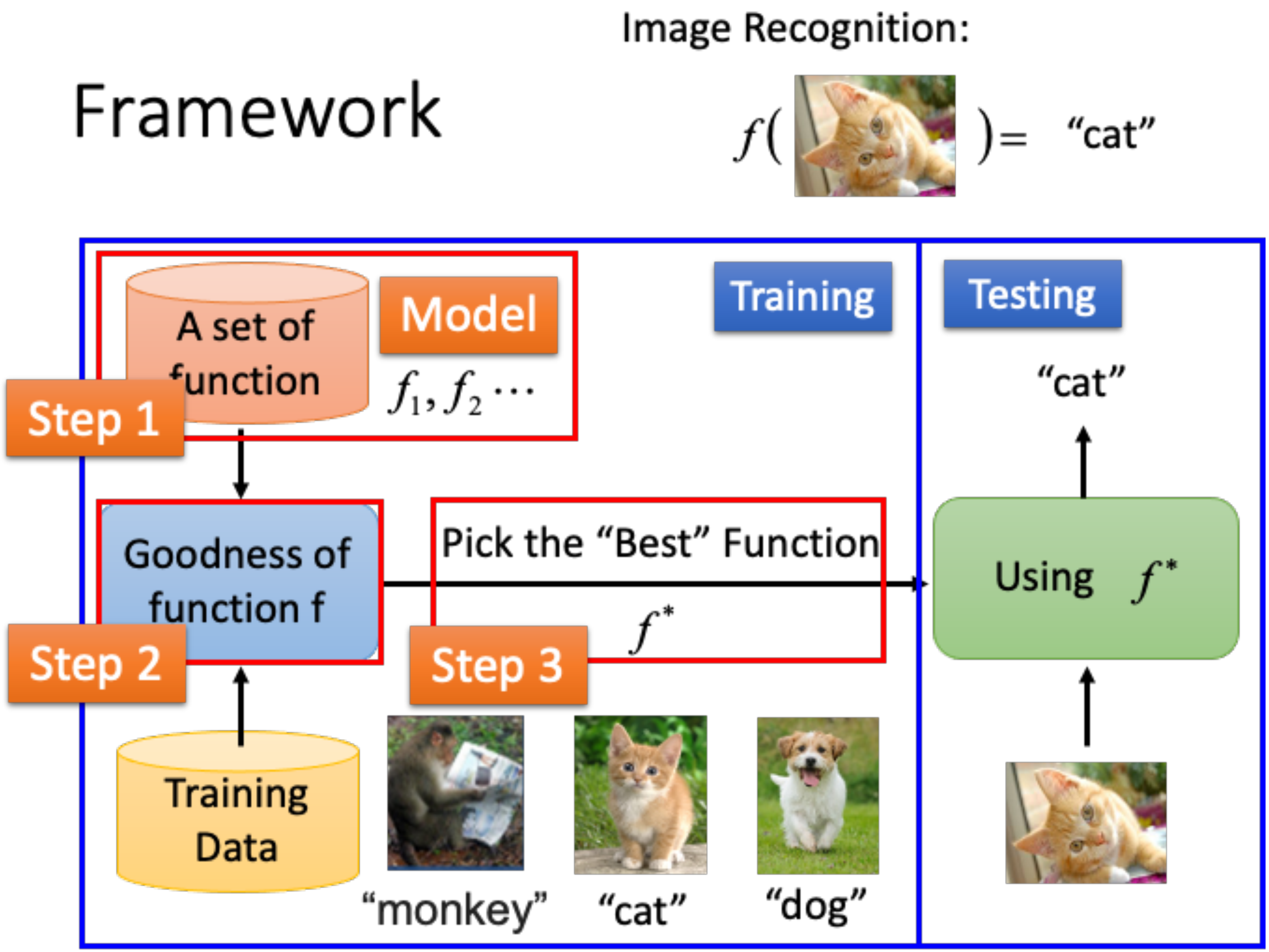
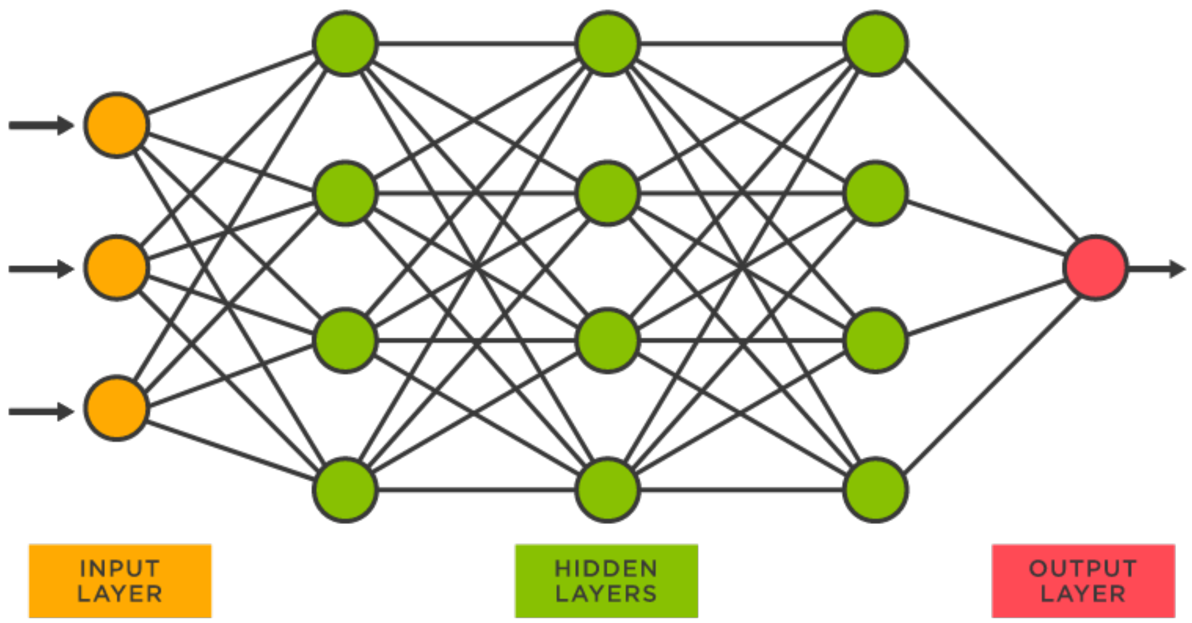
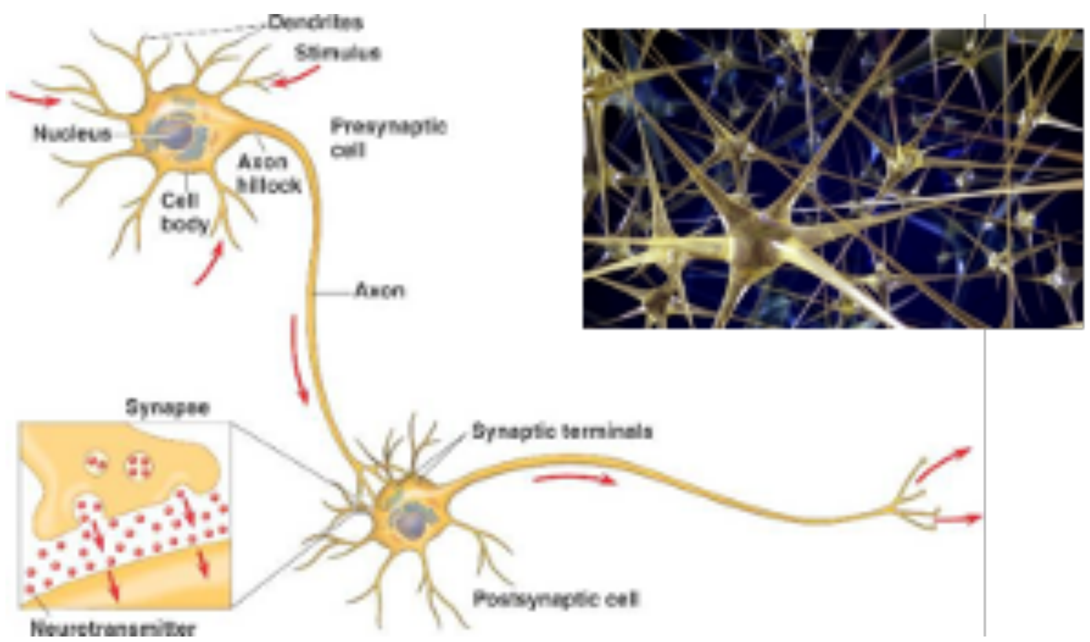


Image Recognition:
Framework
$f($ $)=$ “cat”
Training
Testing
A set of function
Model
$f_1, f_2 \cdots$
Step 1
Goodness of function f
Step 2
Pick the “Best” Function
f^*
Step 3
Using f^*
“cat”
Training Data
“monkey”
“cat”
“dog”

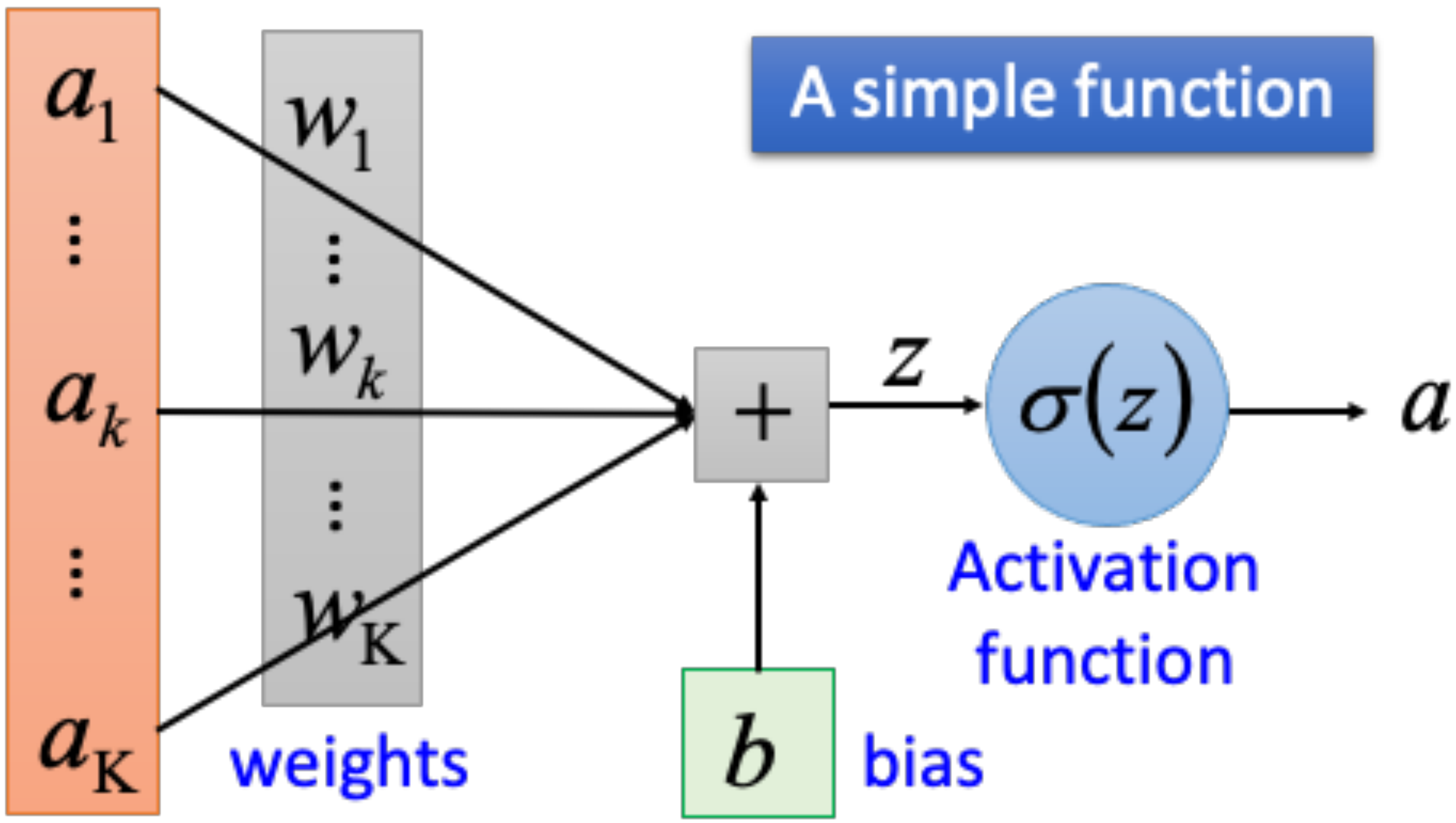
Neural Network



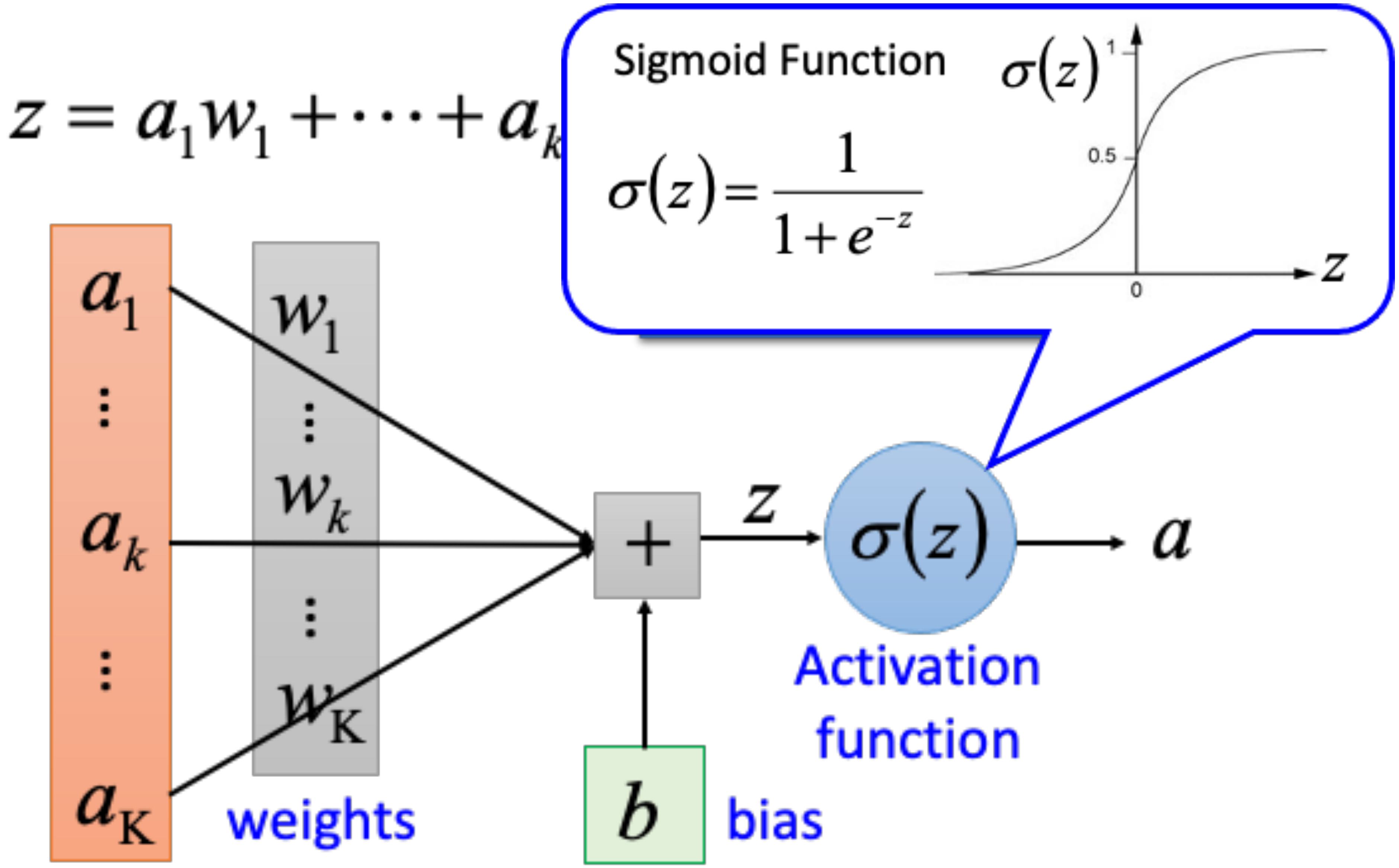


At Each Neuron

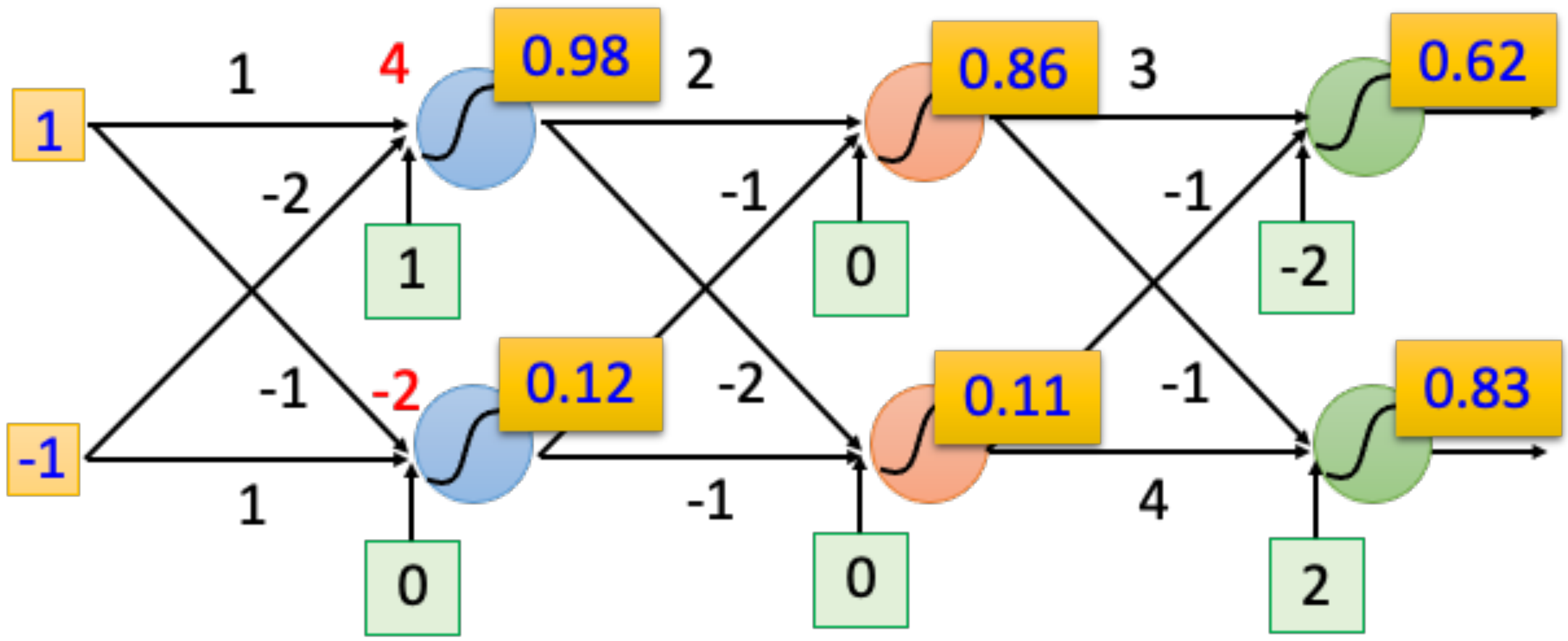
$$z = a_1 w_1 + \cdots + a_k w_k + \cdots + a_K w_K + b$$



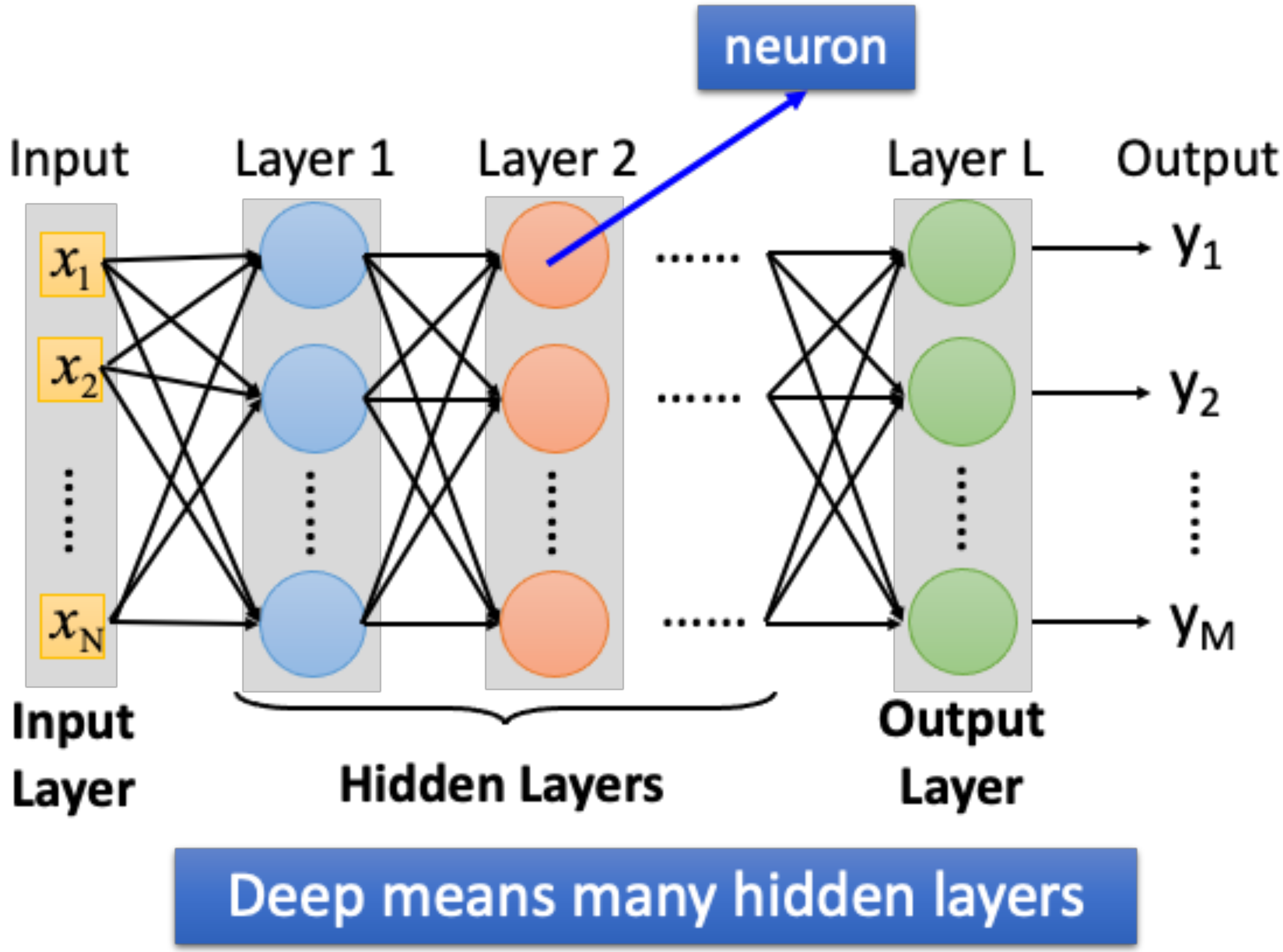
At Each Neuron



Fully Connected Neural Network

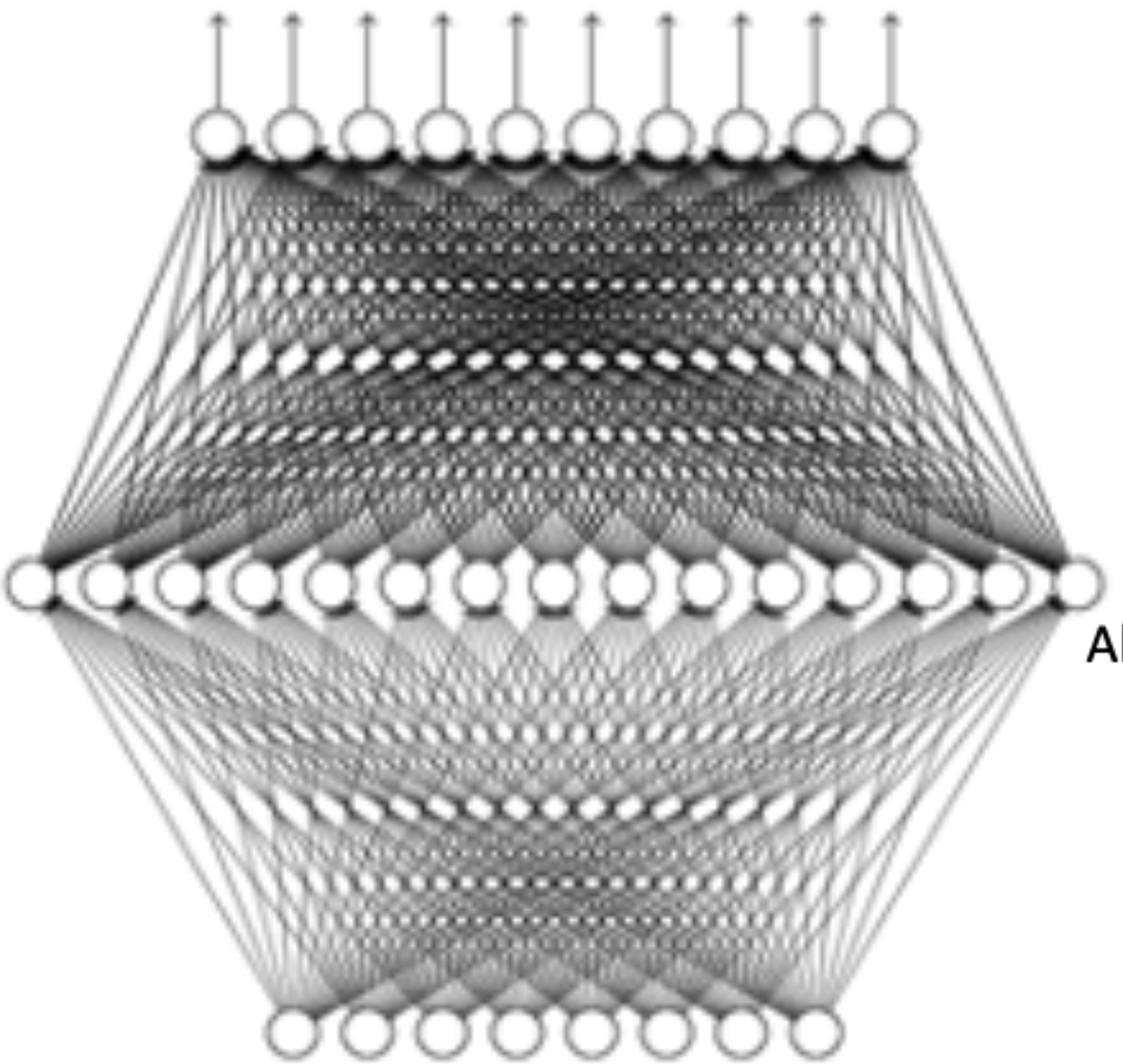


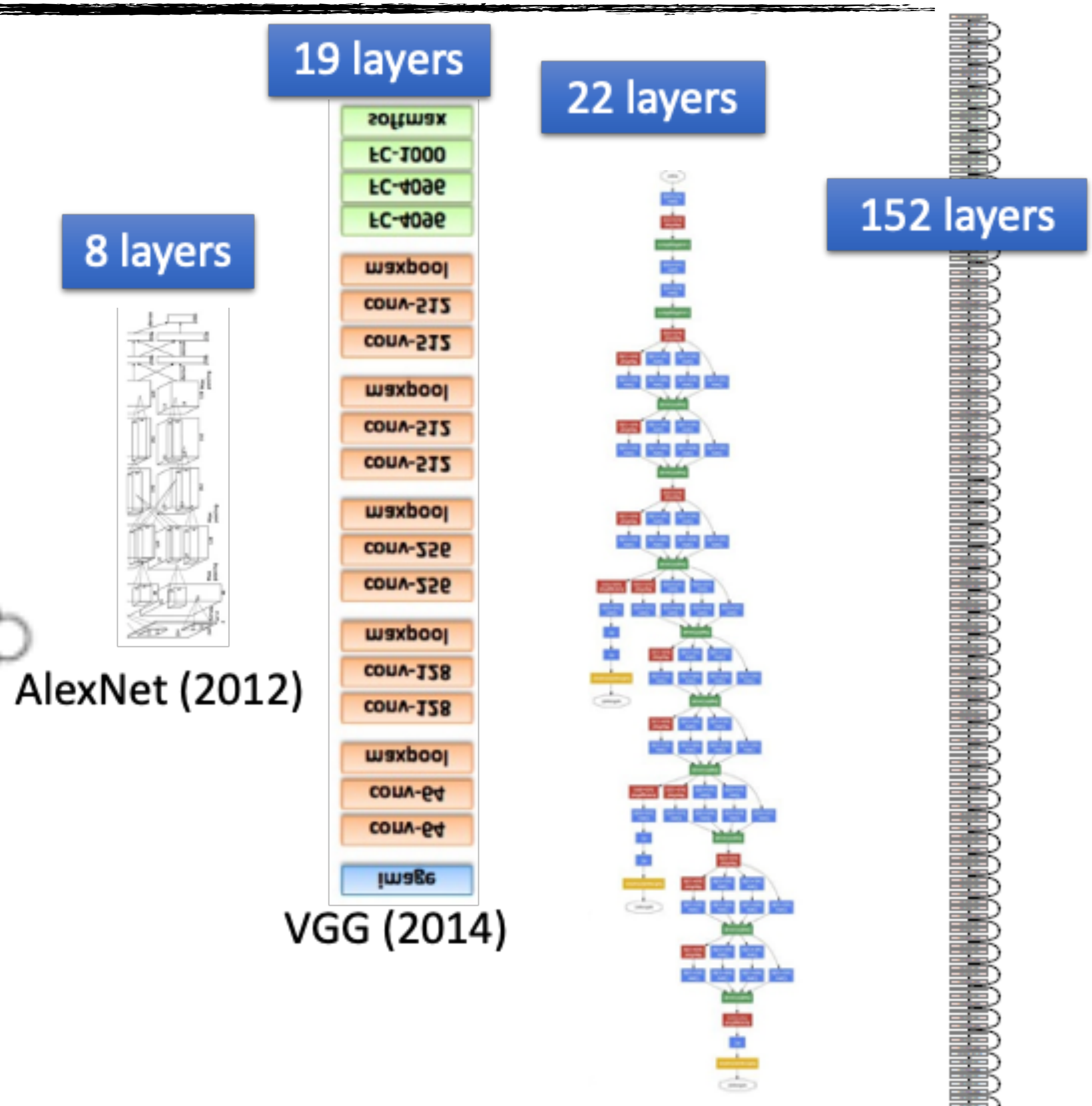
Fully Connected Neural Network



Deep Neural Network

Deep = Many hidden layers





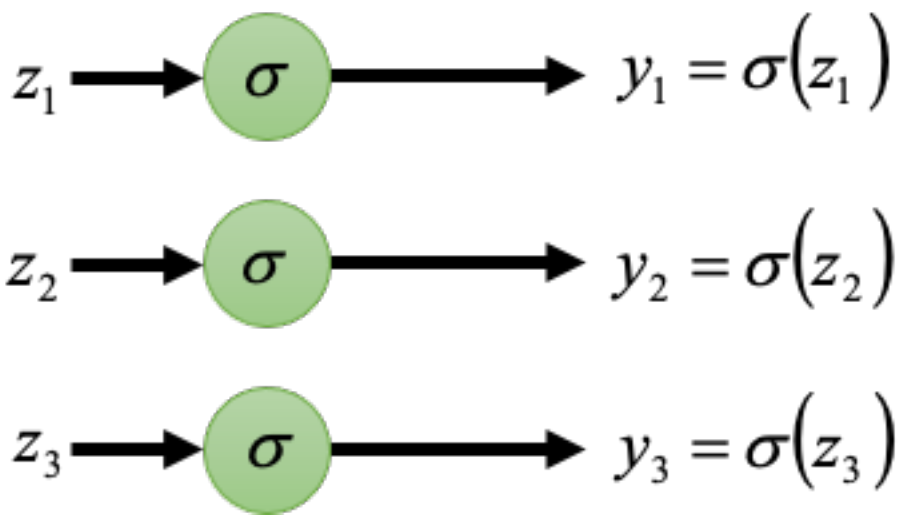
GoogleNet (2015)

ResidualNet (2015)

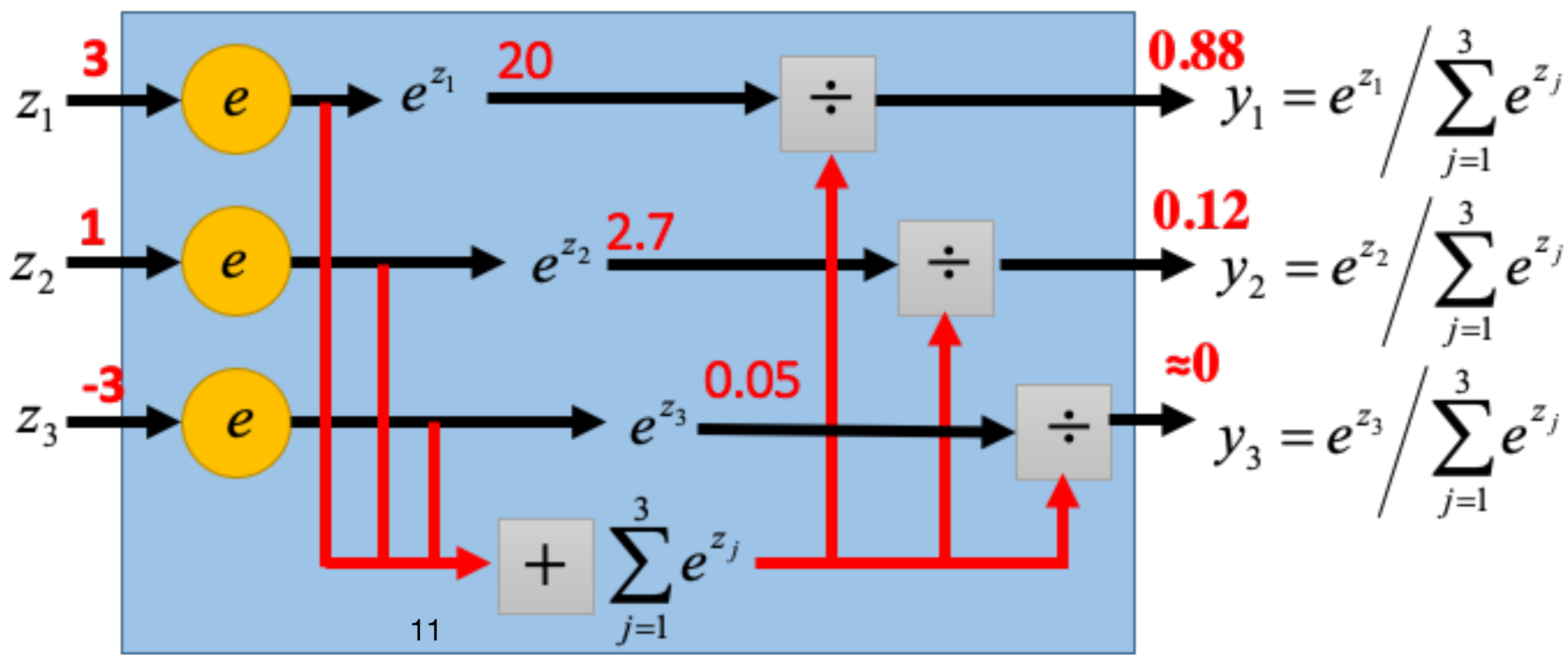
Using multiple layers of neurons to represent some functions

Output Layer

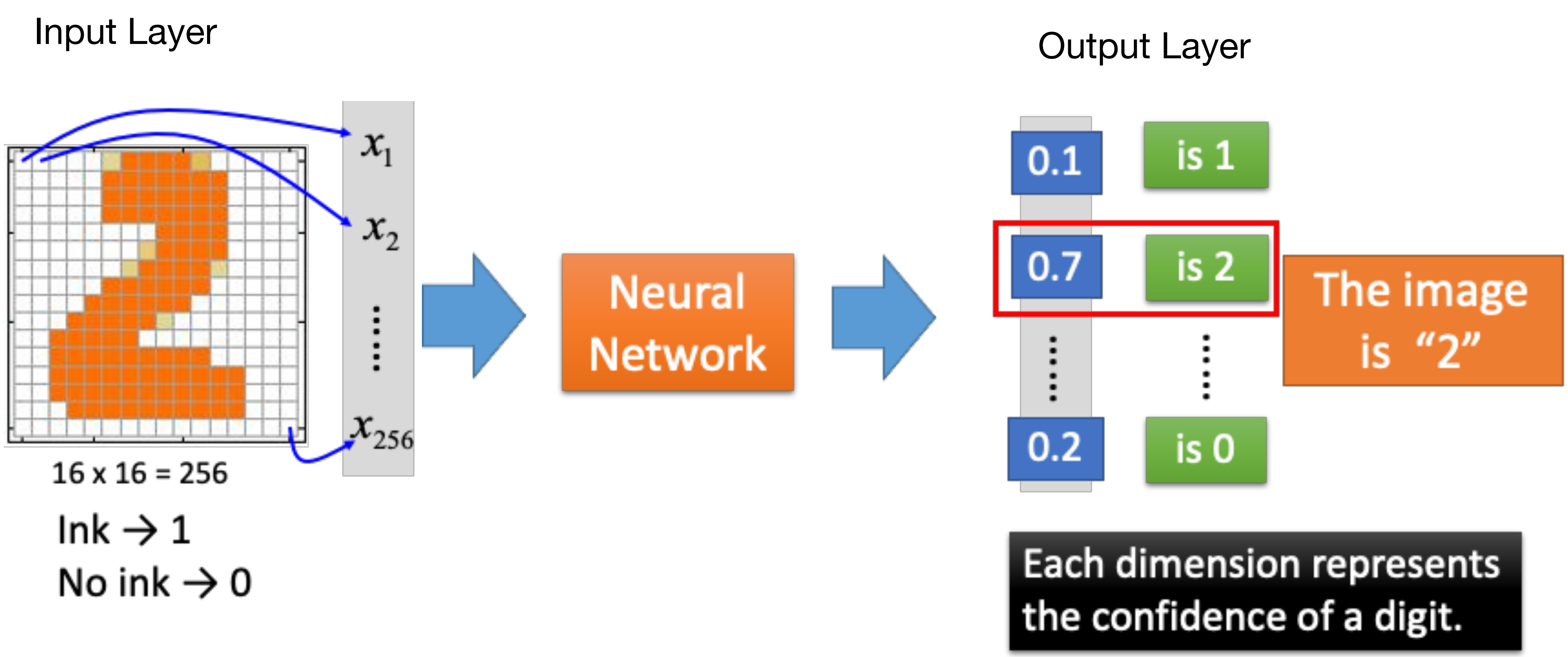
Ordinary Layer



Softmax Layer

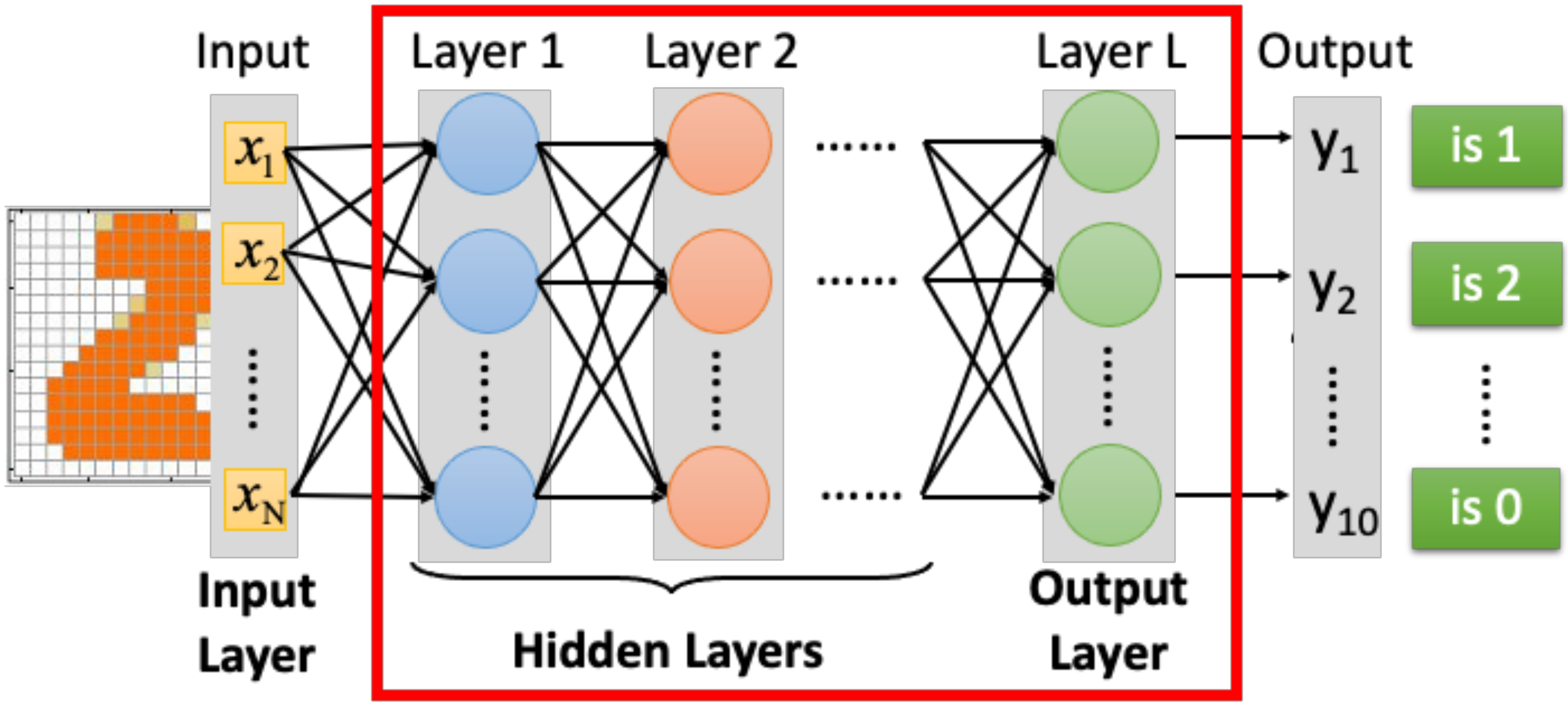


An Example



Input Layer
Output Layer
x_1
x_2
x_{256}
16 x 16 = 256
Ink → 1
No ink → 0
Neural Network
0.1
is 1
0.7
is 2
0.2
is 0
The image is "2"
Each dimension represents the confidence of a digit.

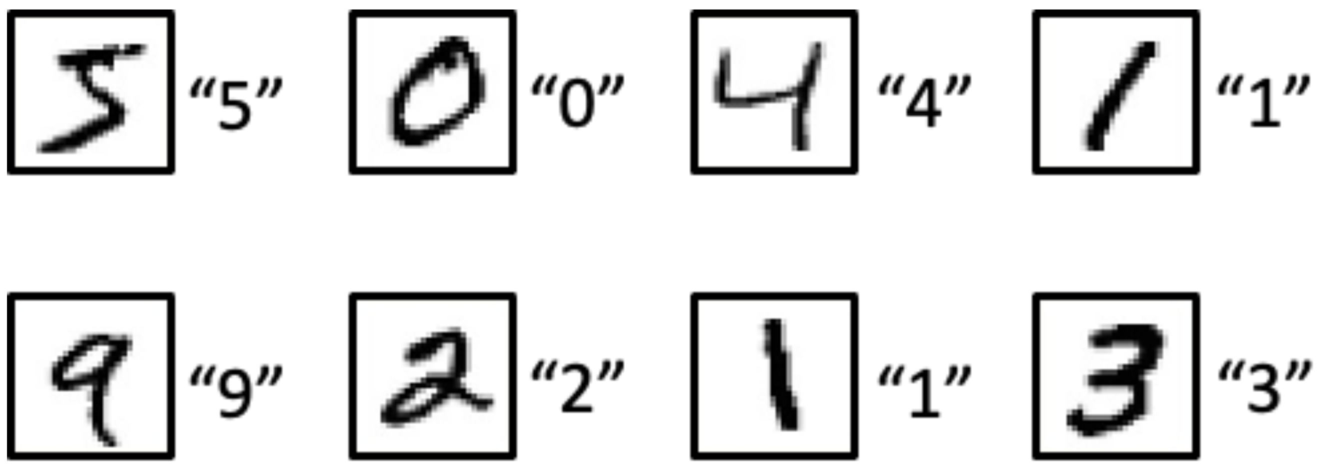
An Example



Needs to determine the network structure

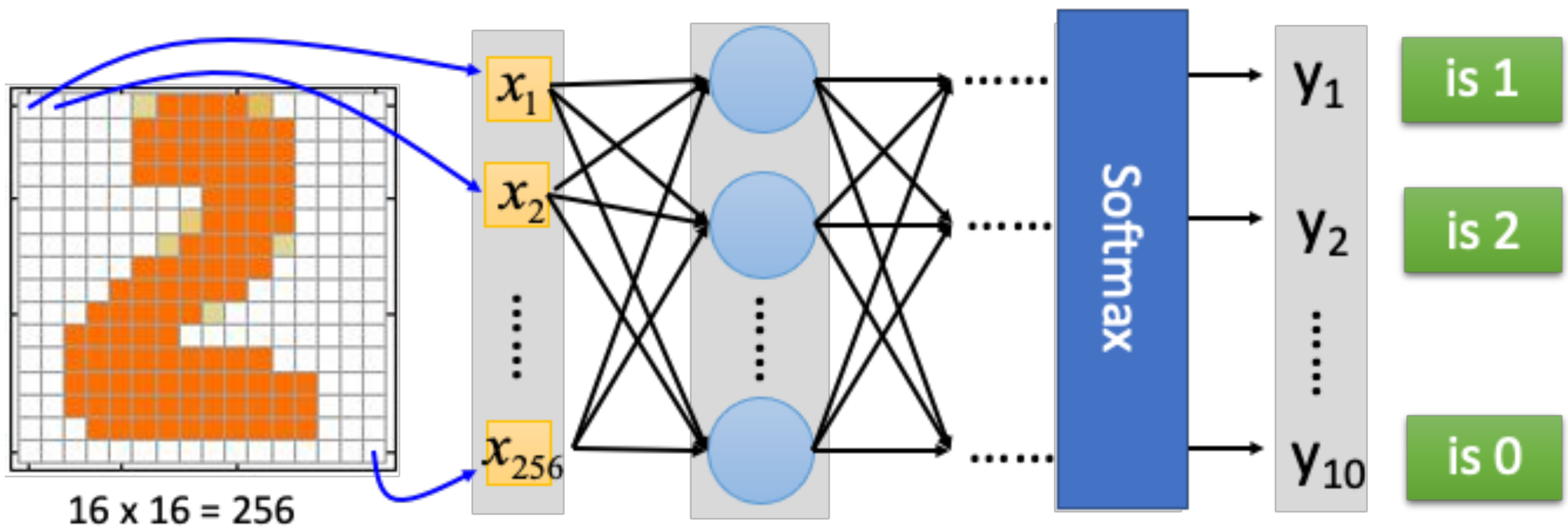
How many layers?
How many neurons for each layer?

Learning Target



The learning target is defined on the training data.

Learning Target



Ink → 1

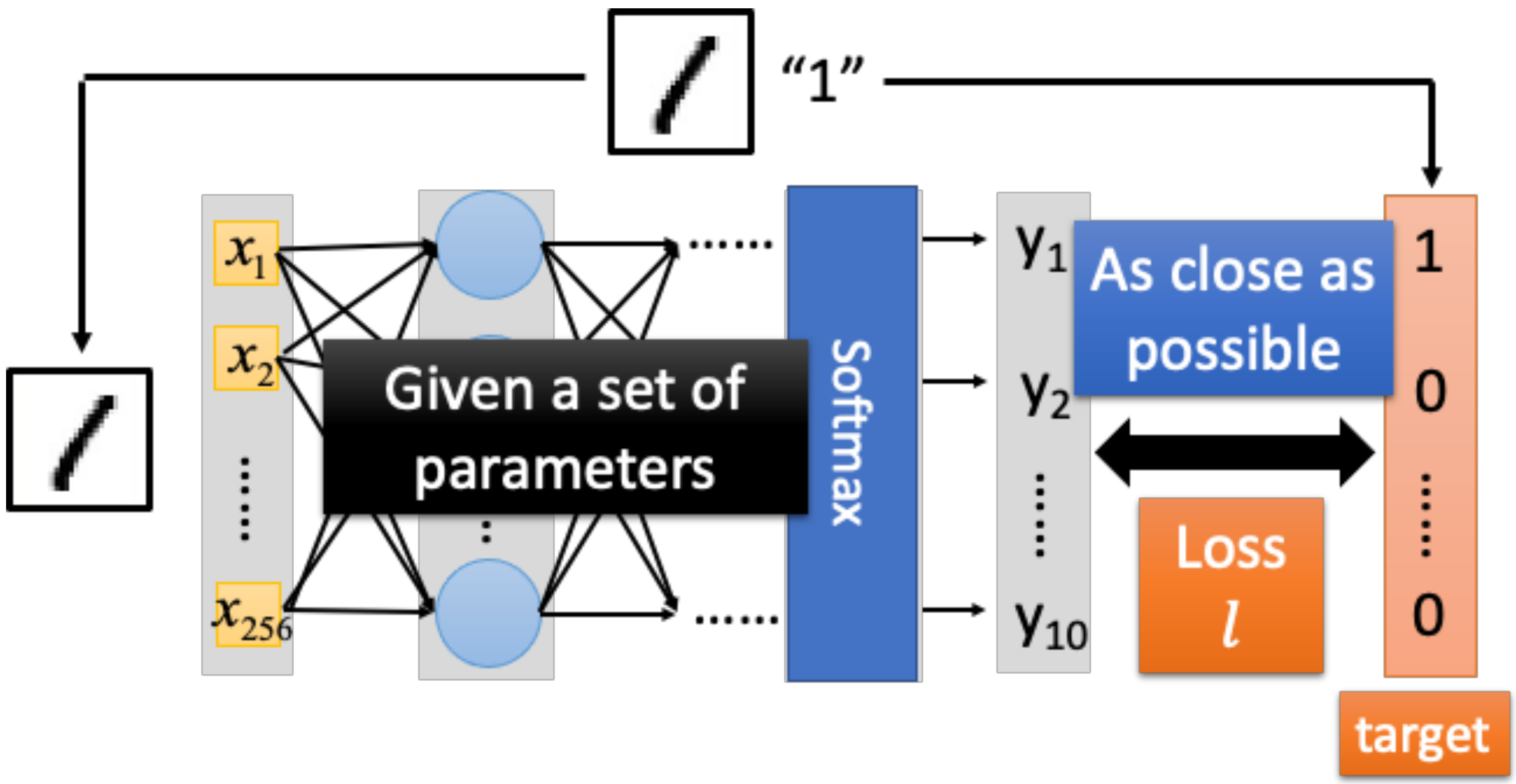
No ink → 0

The learning target is ......

Input:  ➡ y_1 has the maximum value

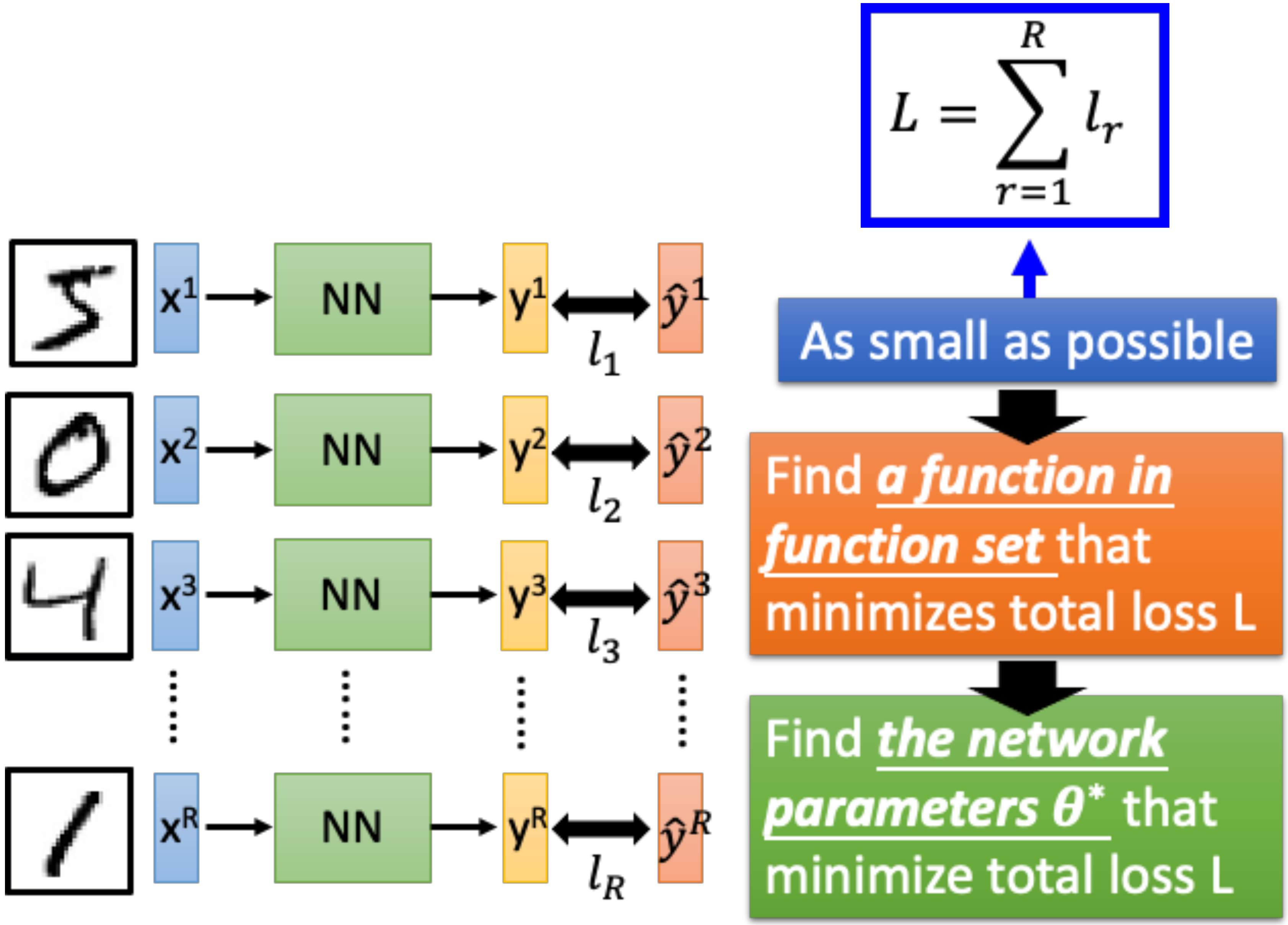
Input:  ➡ y_2 has the maximum value

Good Function = Loss as Small as Possible



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

Total Loss



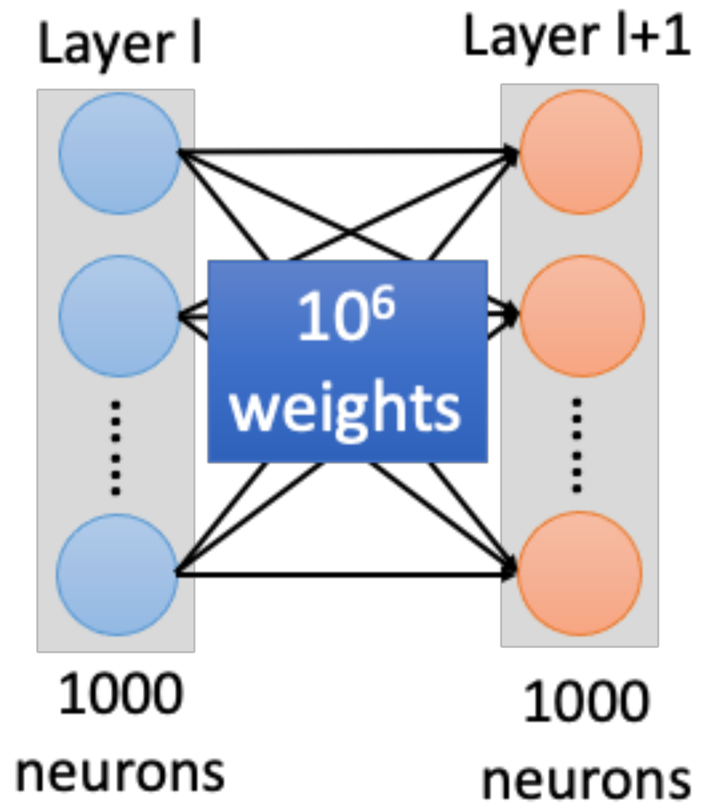
$L = \sum_{r=1}^{R} l_r$
As small as possible
Find a function in function set that minimizes total loss L
Find the network parameters θ^* that minimize total loss L
x^1 NN y^1 l_1 $\hat{y}^1$
x^2 NN y^2 l_2 $\hat{y}^2$
x^3 NN y^3 l_3 $\hat{y}^3$
x^R NN y^R l_R $\hat{y}^R$

Total Loss

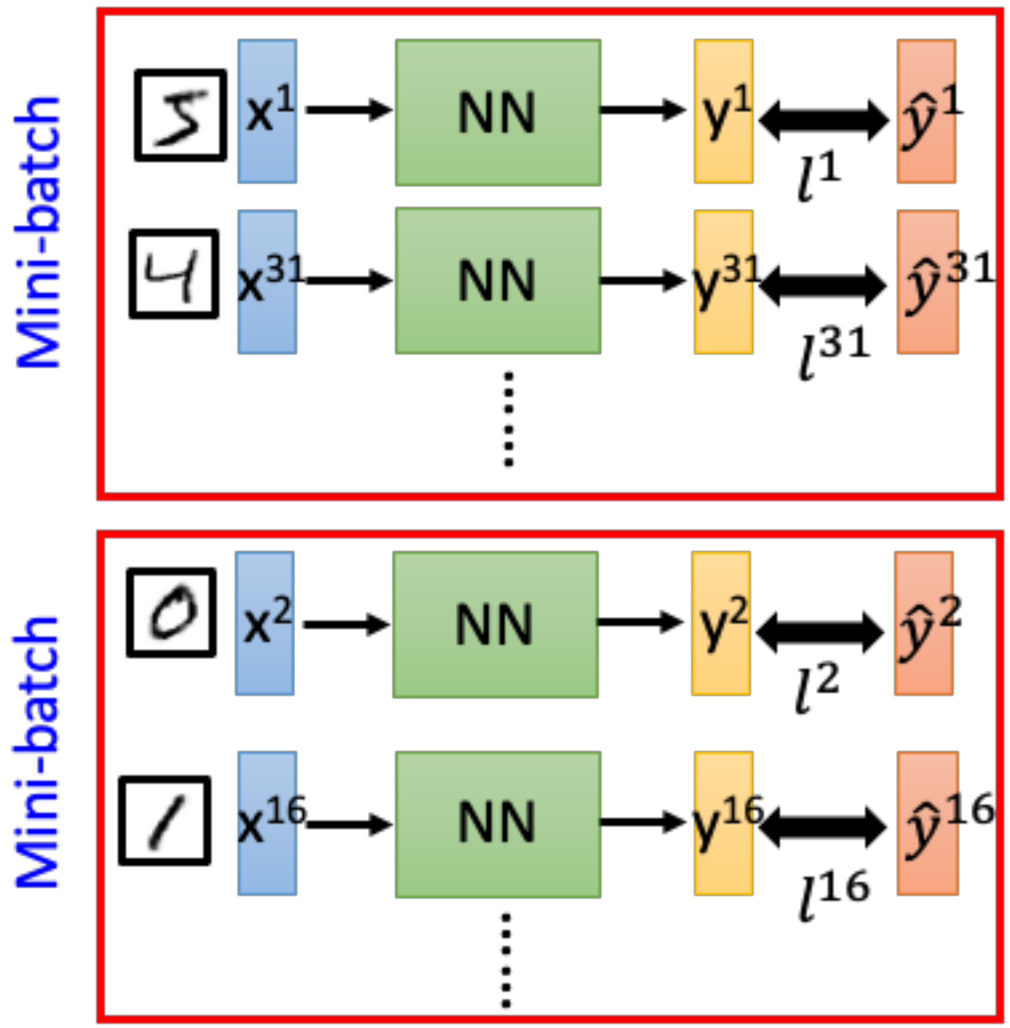
Find ***network parameters θ^**** that minimize total loss L

Enumerate all possible values

Network parameters $\theta = \underbrace{\{w_1, w_2, w_3, \cdots, b_1, b_2, b_3, \cdots\}}_{\text{Millions of parameters}}$



Mini-batch



- Randomly initialize network parameters
- Pick the 1st batch
  $L' = l^1 + l^{31} + \cdots$
  Update parameters once
- Pick the 2nd batch
  $L'' = l^2 + l^{16} + \cdots$
  Update parameters once

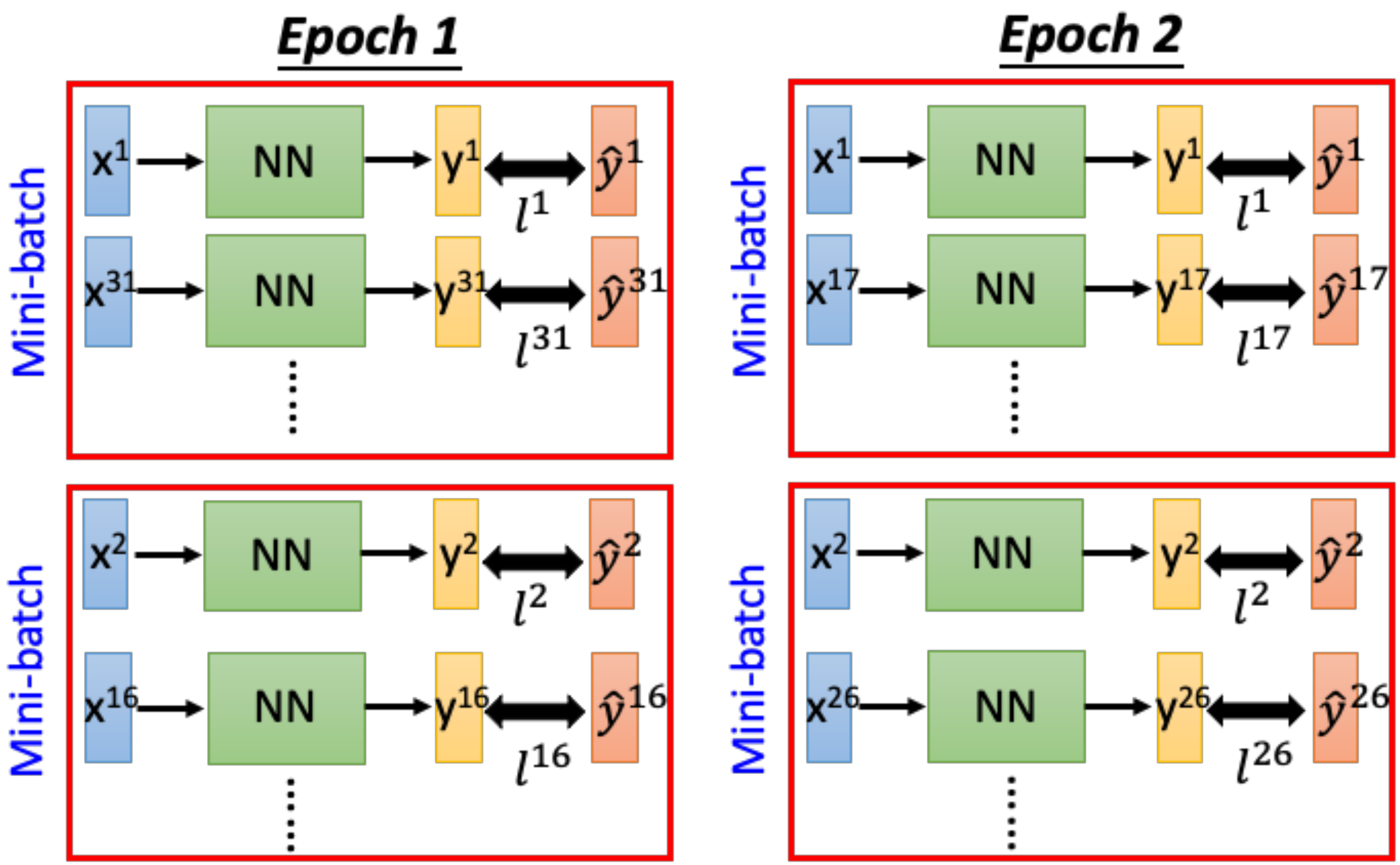
  ⋮
- Until all mini-batches have been picked

one epoch

Repeat the above process

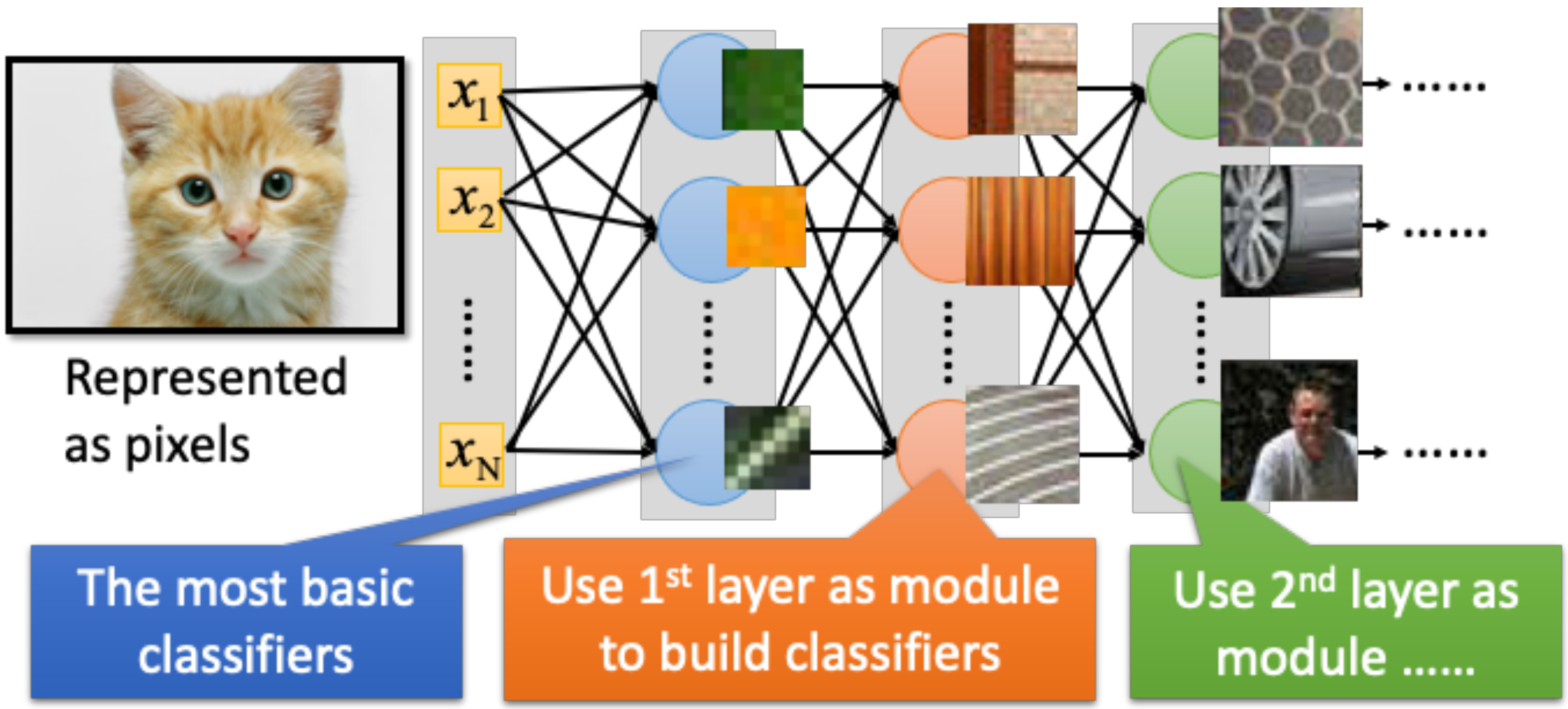
We do not really minimize total loss!

Mini-batch



Epoch 1
Epoch 2
Mini-batch
Mini-batch
Mini-batch
Mini-batch
x1
NN
y1
ŷ1
l1
x31
NN
y31
ŷ31
l31
x2
NN
y2
ŷ2
l2
x16
NN
y16
ŷ16
l16
x1
NN
y1
ŷ1
l1
x17
NN
y17
ŷ17
l17
x2
NN
y2
ŷ2
l2
x26
NN
y26
ŷ26
l26

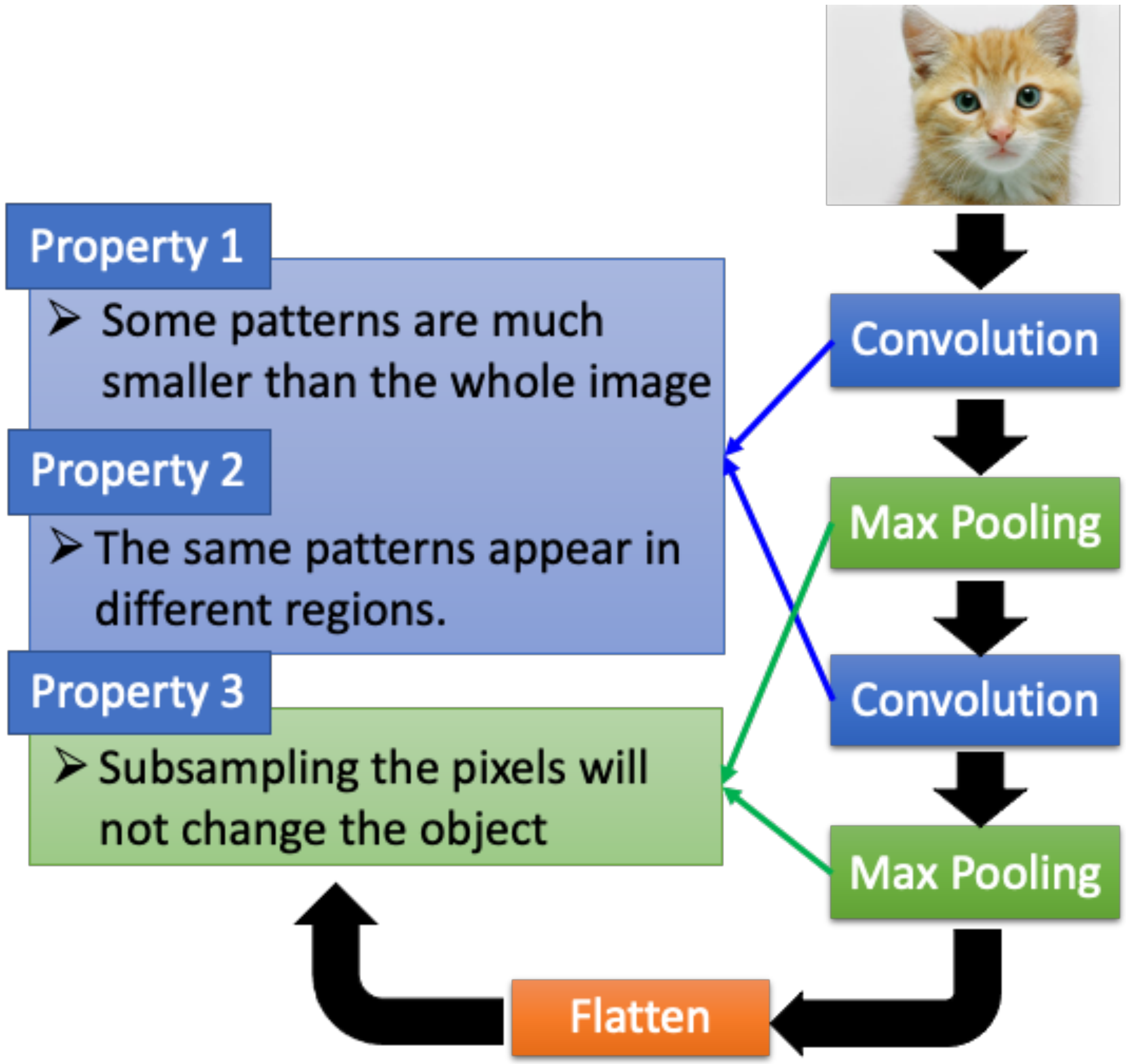
Convolutional Neural Network (CNN)



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

3

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

Filter 1

3

-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

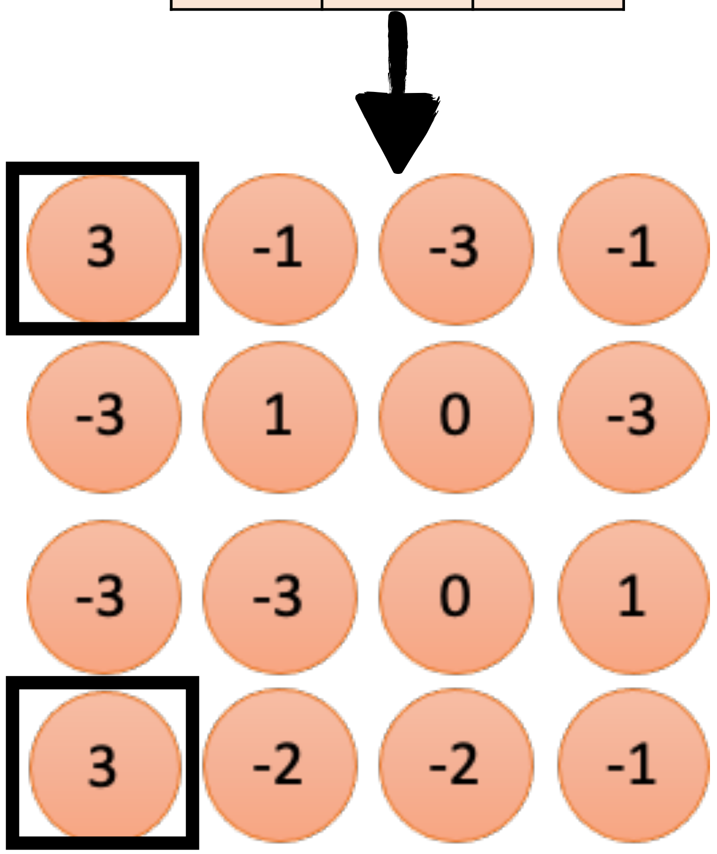
-3

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



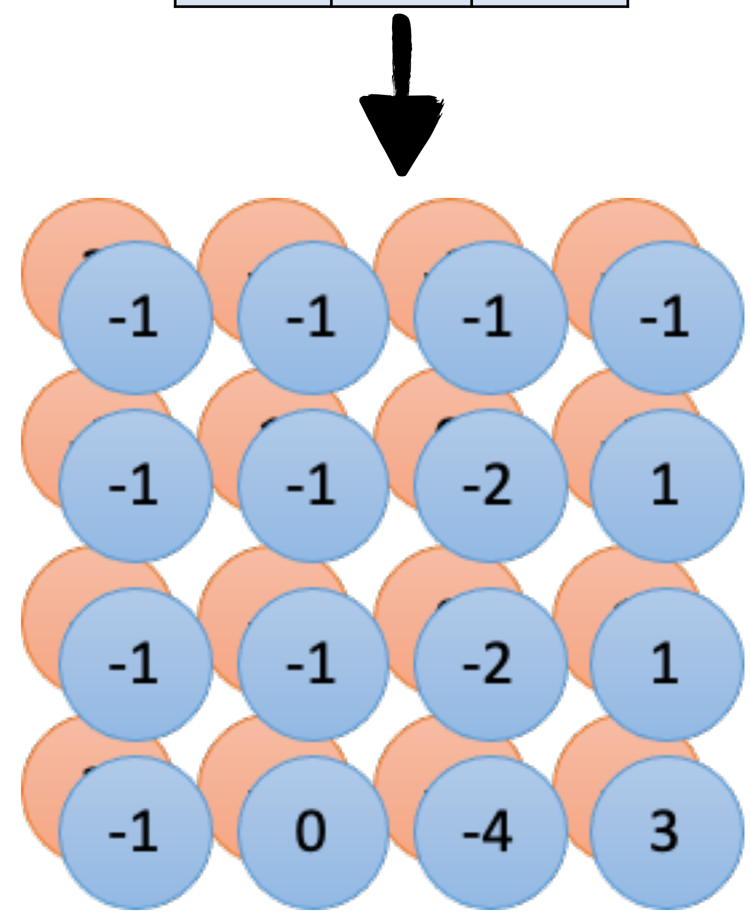
Property 2: Same patterns appears at different region!

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

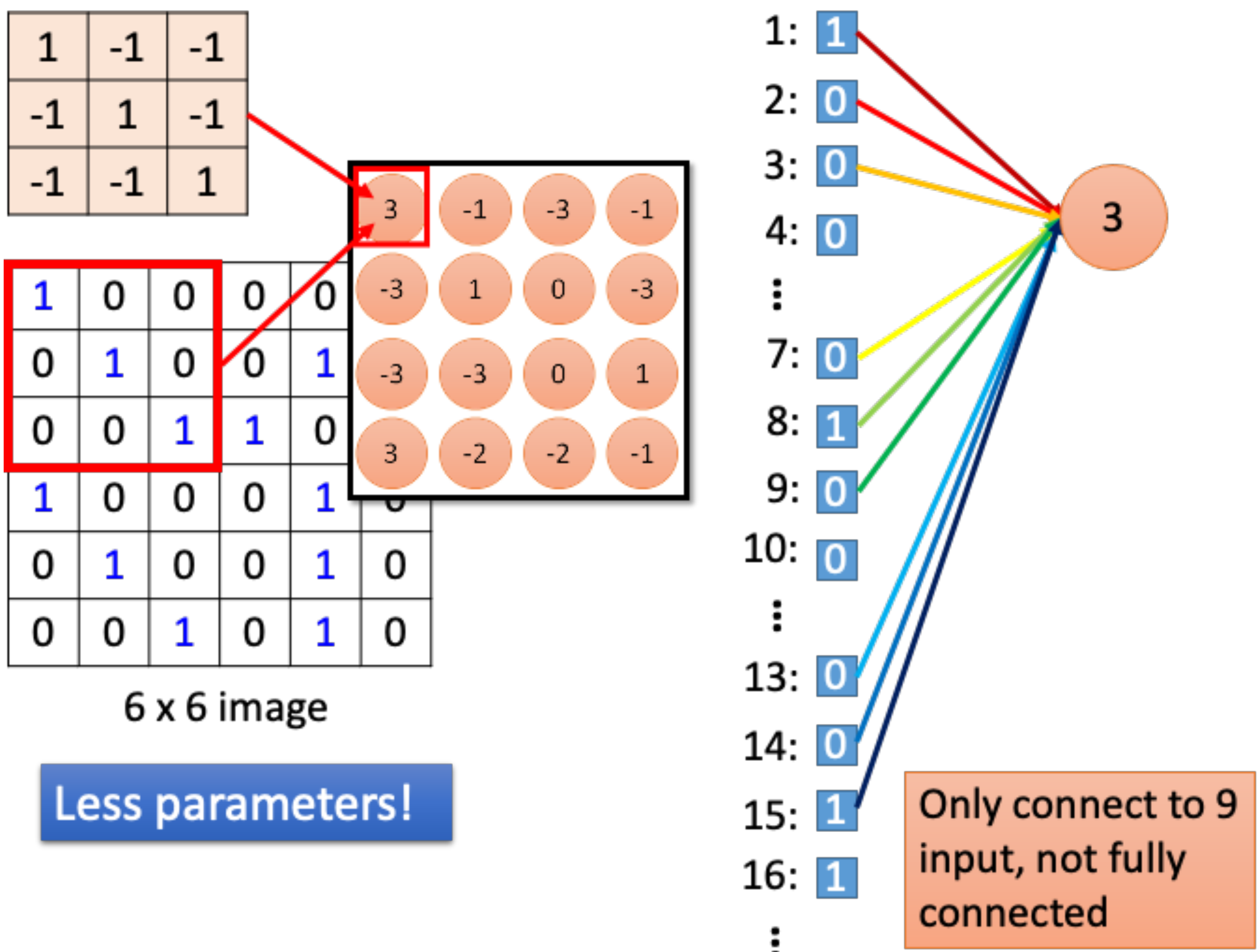
-1	1	-1
-1	1	-1
-1	1	-1

Filter2

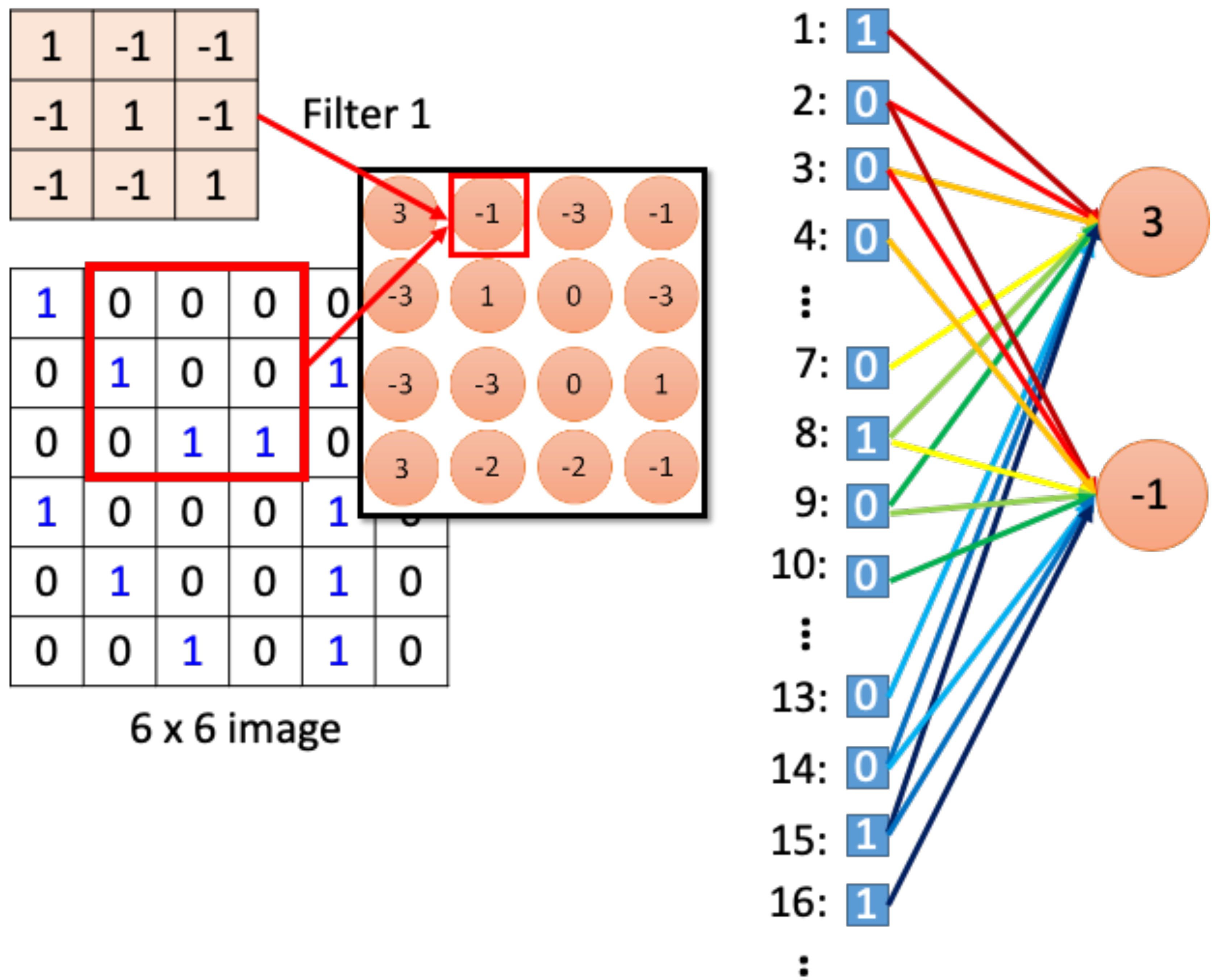


Property 2: Same patterns appears at different region!

CNN -Convolution



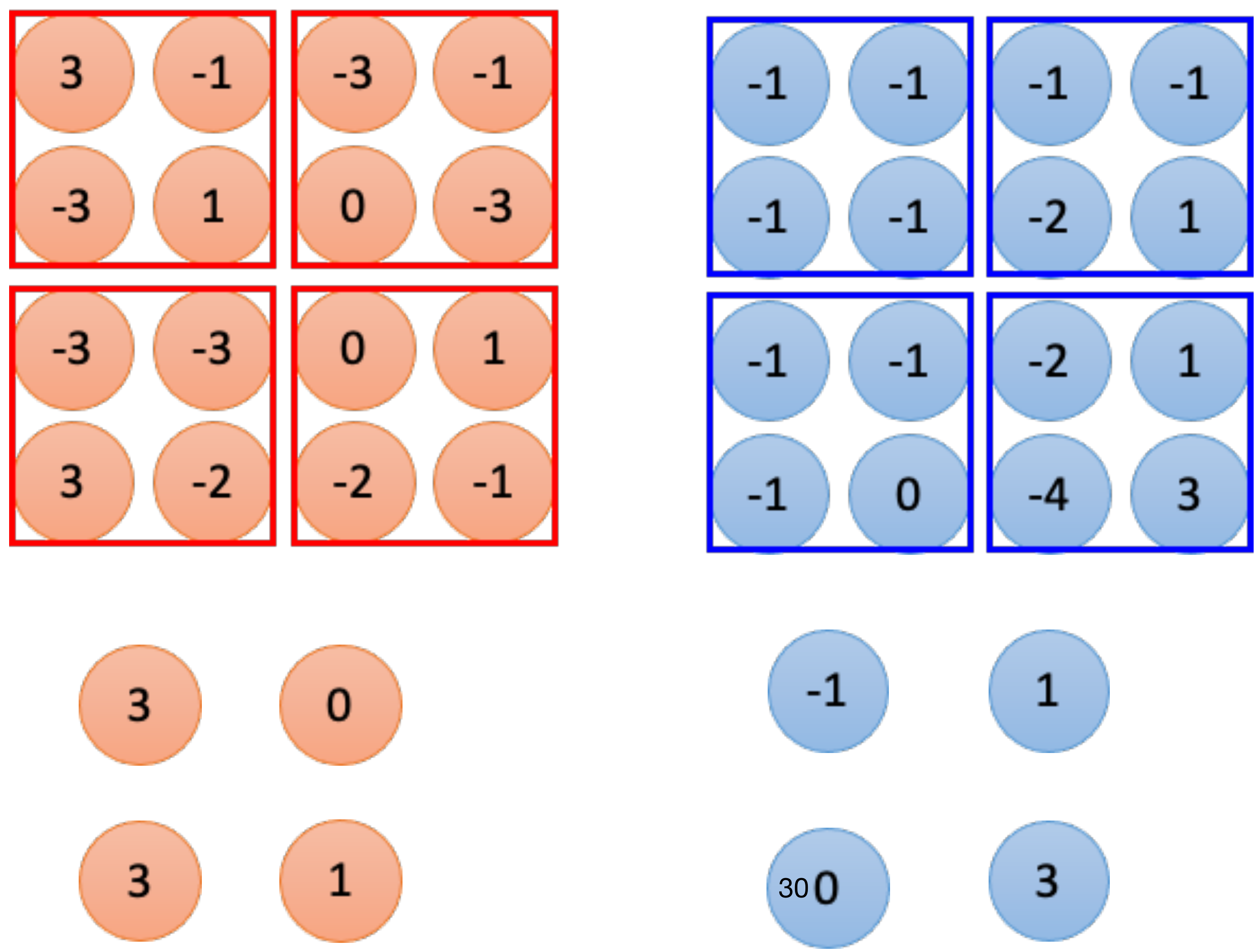
CNN -Convolution



CNN -Max Pooling

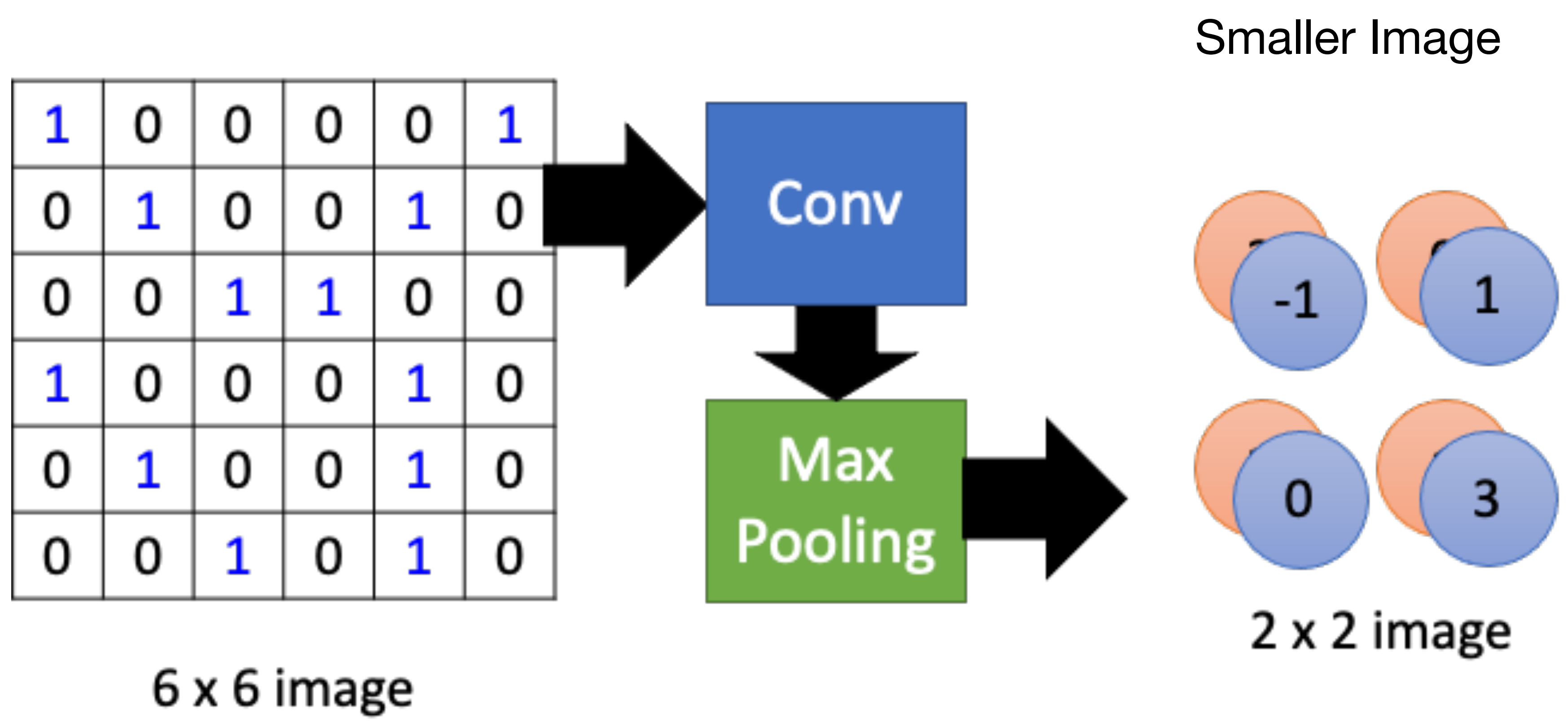
1	-1	-1
-1	1	-1
-1	-1	1

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

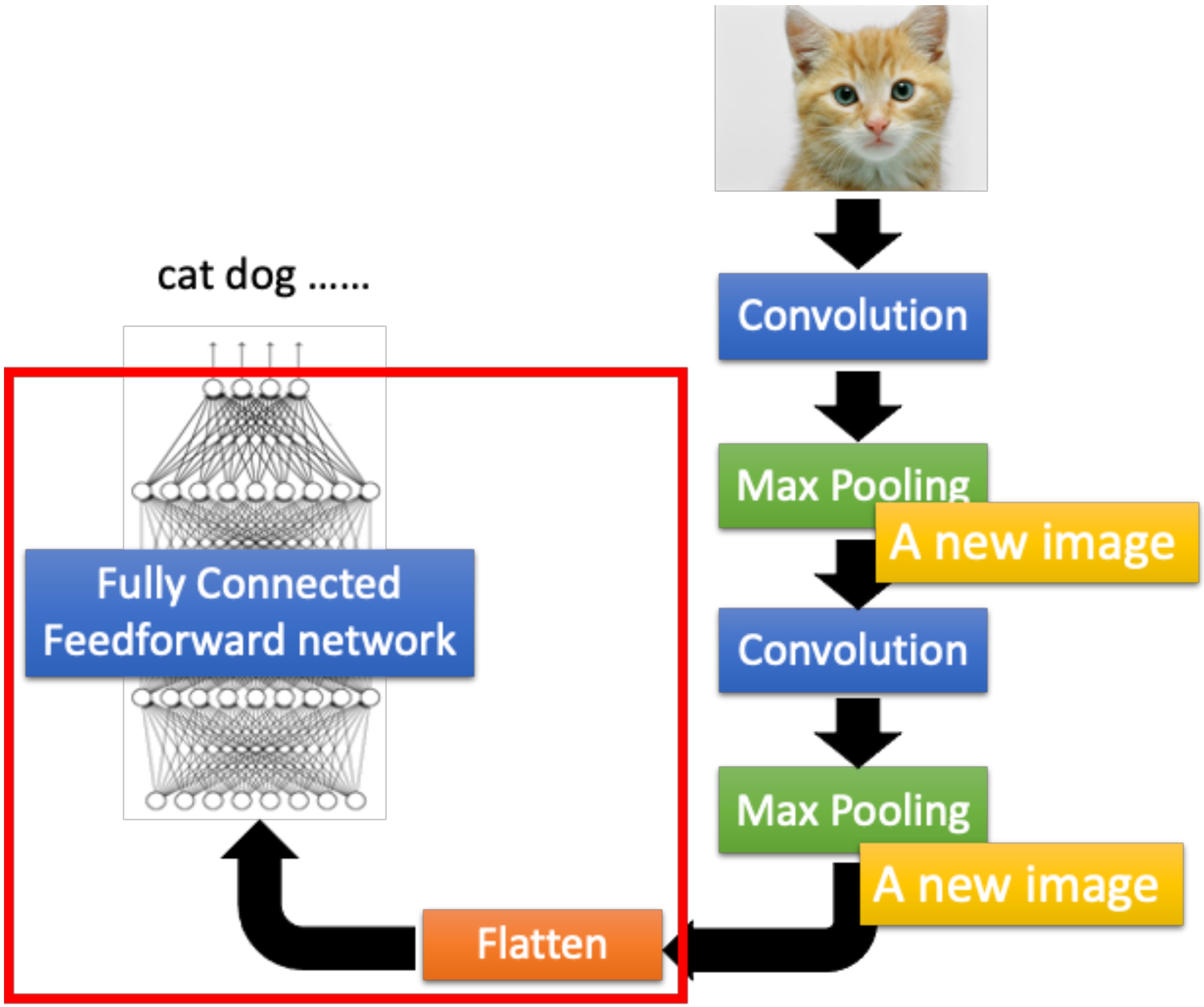


Take the dominant Features
Feature Dimensionality Reductio

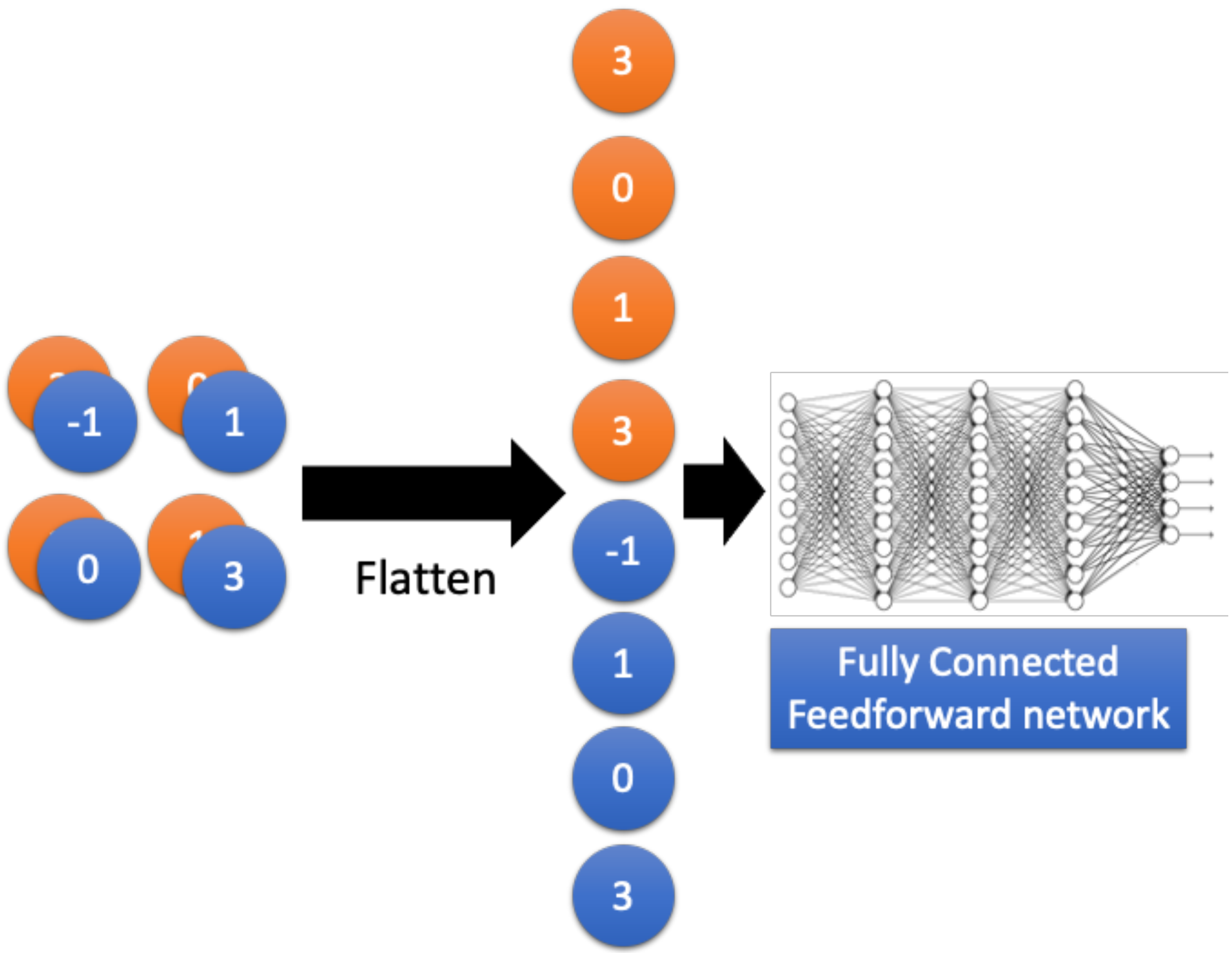
CNN -Convolution



The Whole CNN

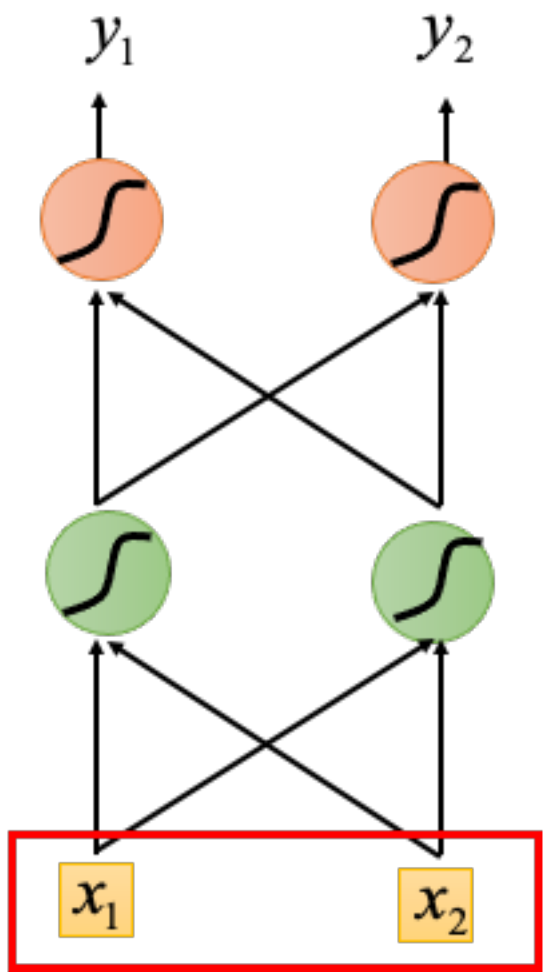


Flatten


-1
1
0
3
Flatten
3
0
1
3
-1
1
0
3
Fully Connected
Feedforward network

Recurrent Neural Network

Input: Vector



Word to Vector

1to N encoding → Embedding to high dimensional space

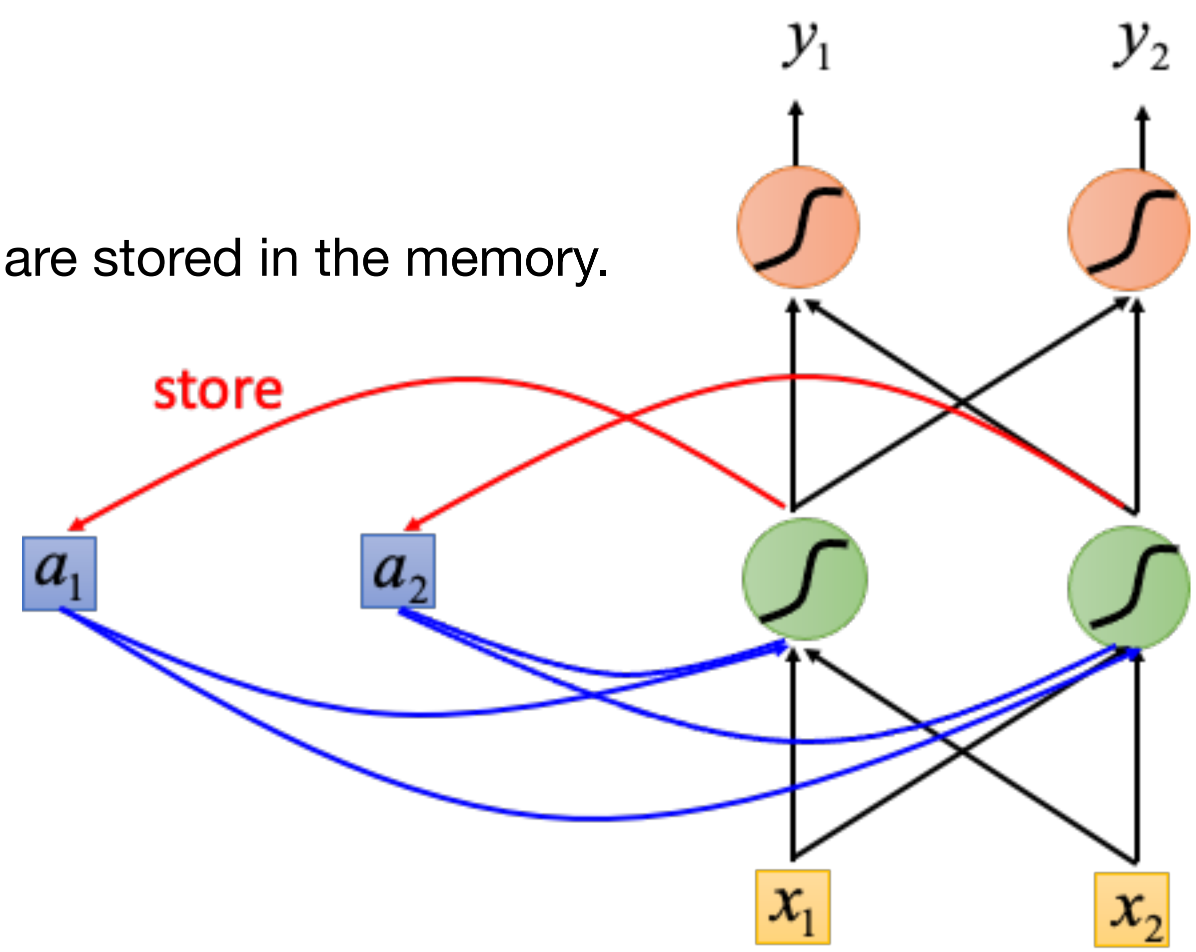
lexicon = {apple, bag, cat, dog, elephant}

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]

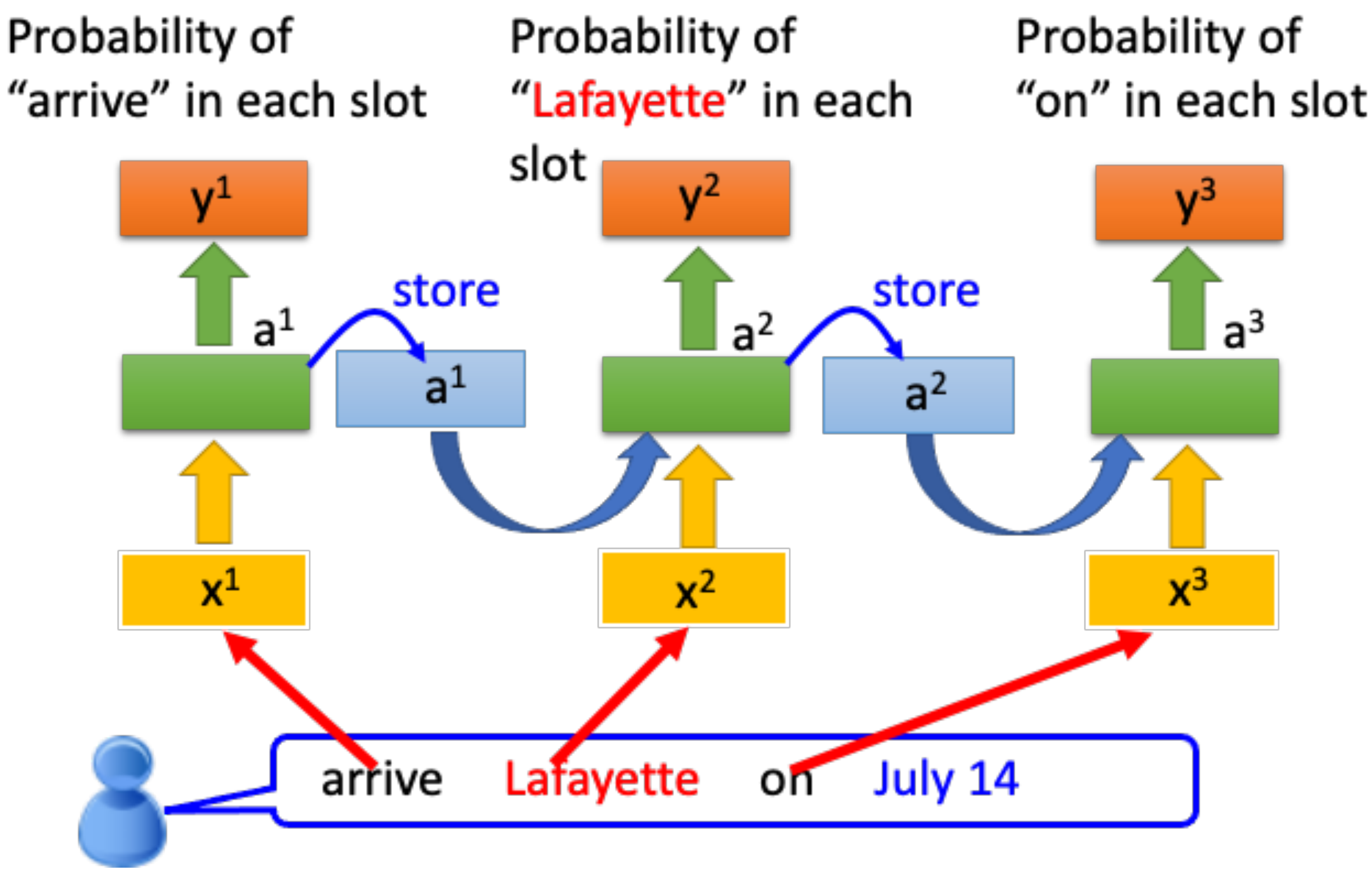
Popular techniques: Word2Vector, Node2Vector

RNN

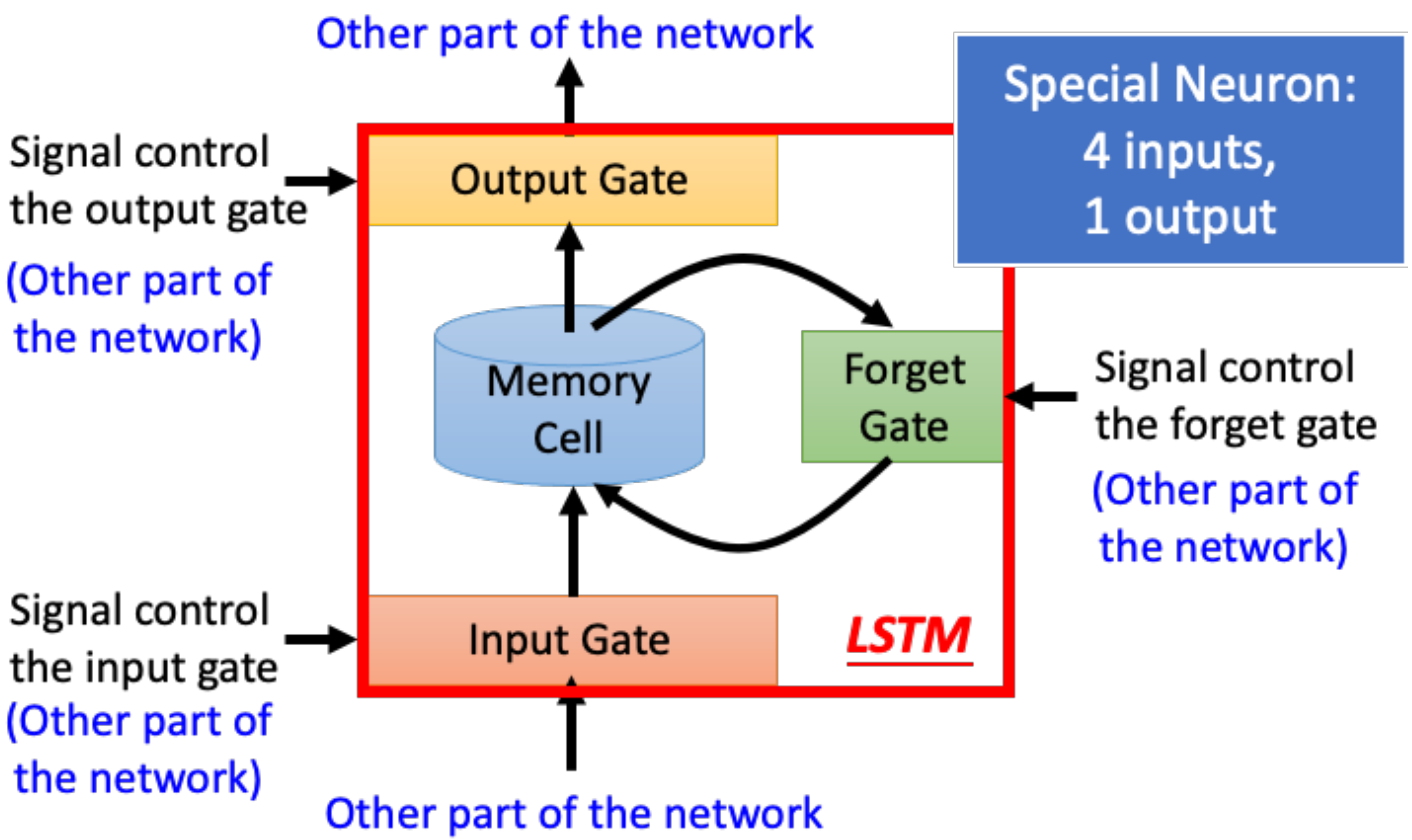
The output of hidden layer are stored in the memory.



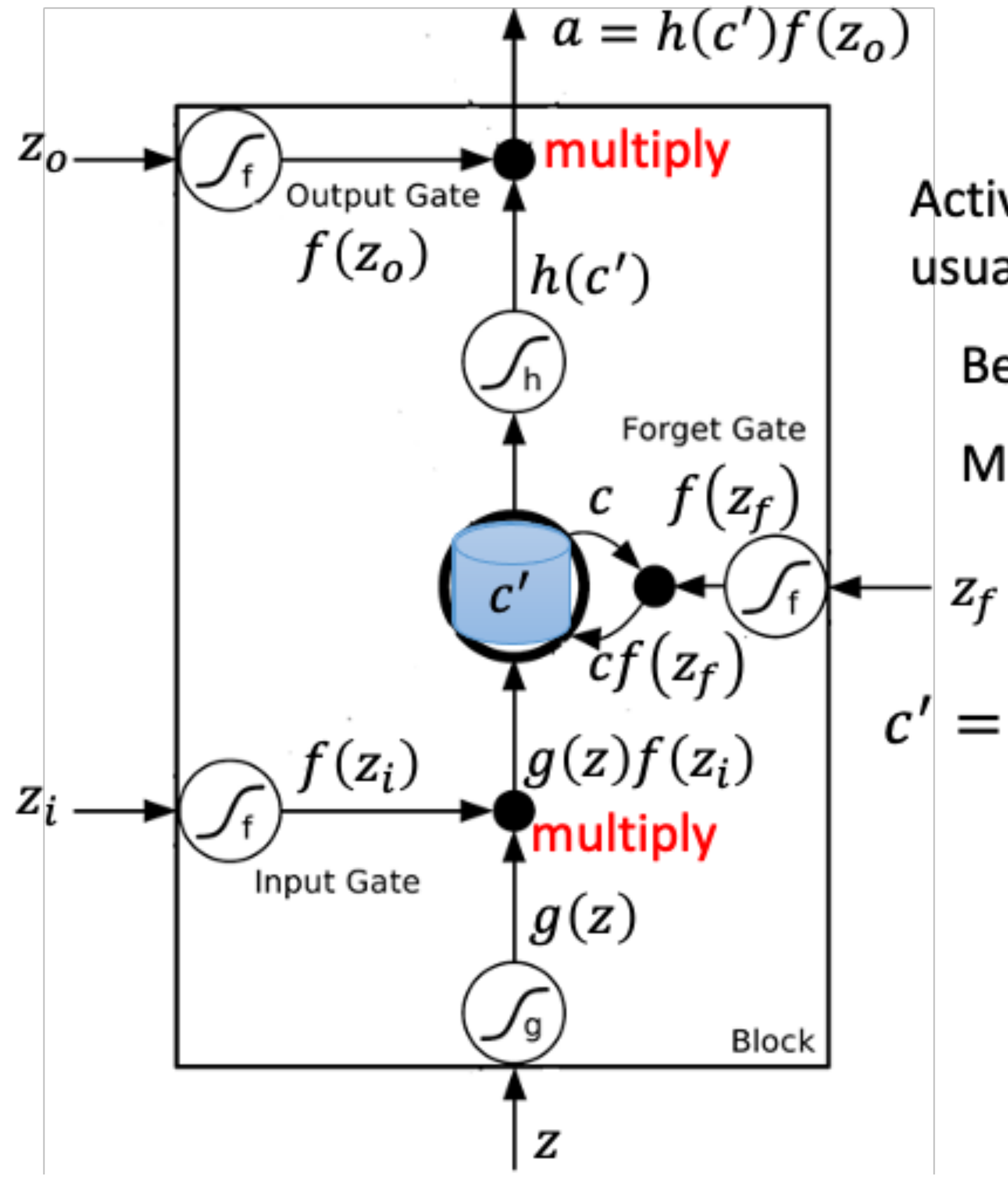
RNN


Probability of "arrive" in each slot
Probability of "Lafayette" in each slot
Probability of "on" in each slot
y1
y2
y3
a1
a2
a3
store
store
a1
a2
x1
x2
x3
arrive Lafayette on July 14

Long Short-term Memory (LSTM)



Long Short-term Memory (LSTM)

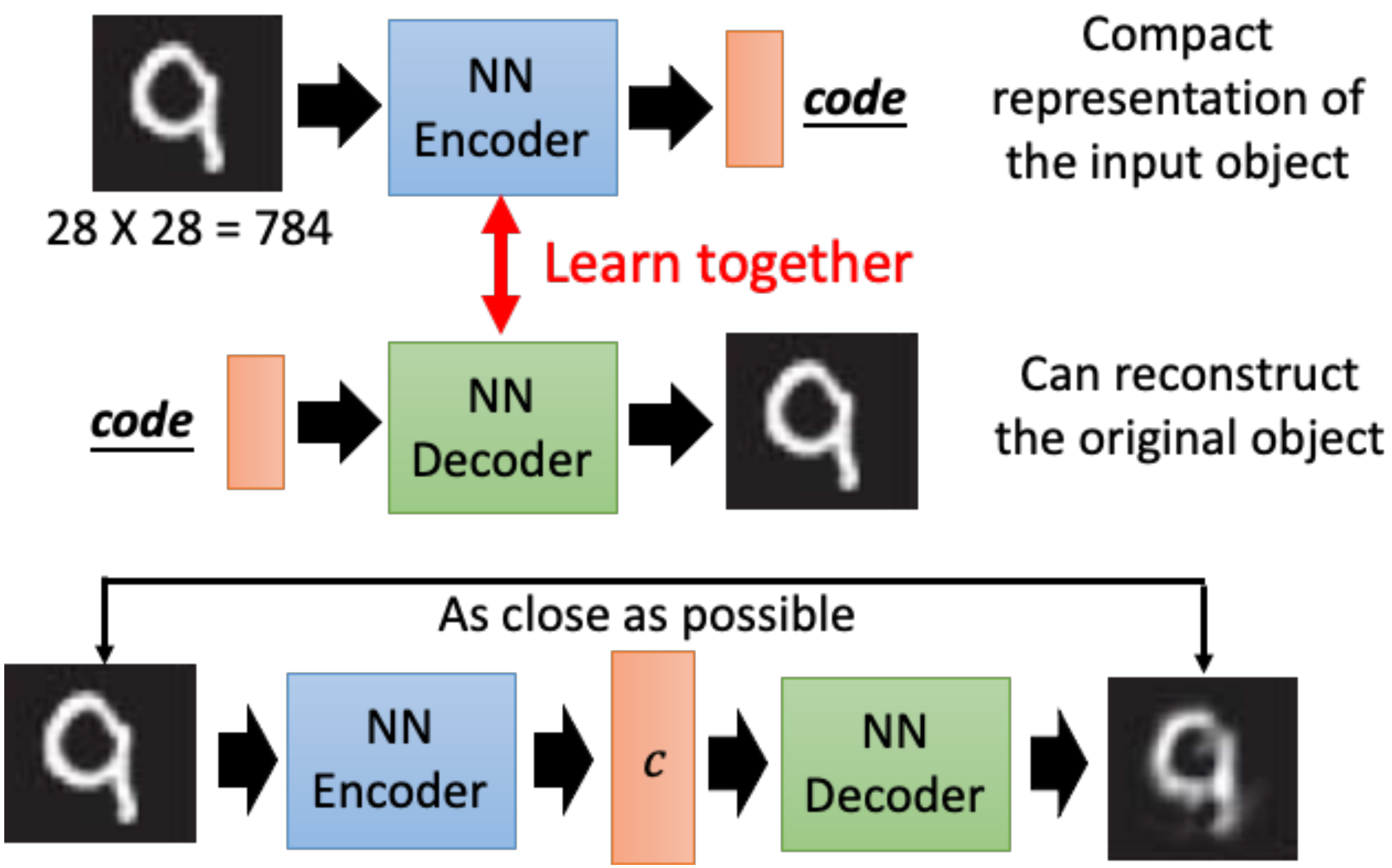


Activation function f is usually a sigmoid function

- Between 0 and 1
- Mimic open and close gate

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

Auto-Encoder



Deep Auto-Encoder