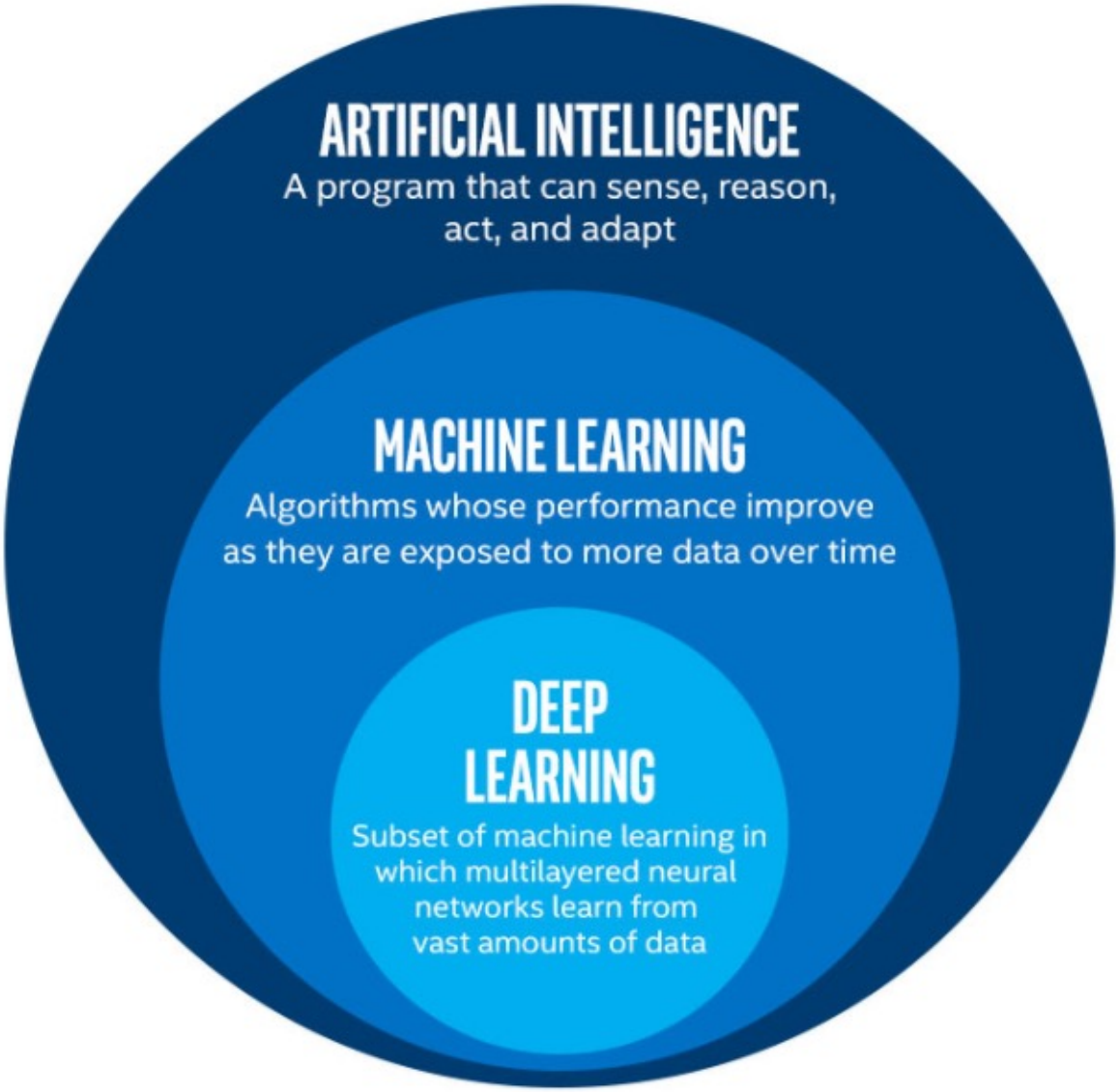
CAP 5516
Medical Image Computing
(Spring 2022)

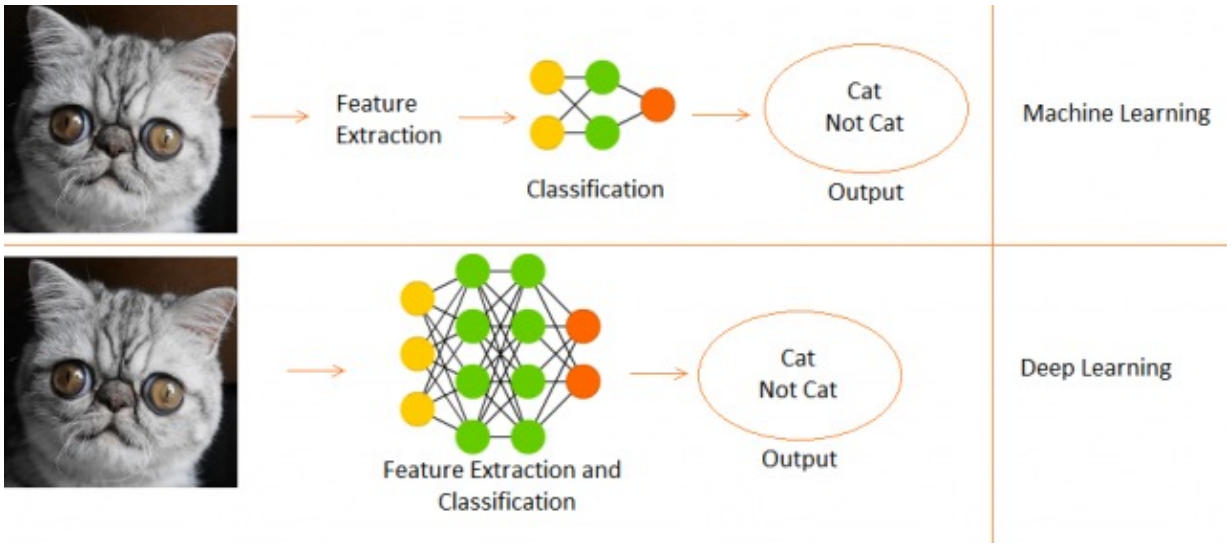
Dr. Chen Chen
Center for Research in Computer Vision (CRCV)
University of Central Florida
Office: HEC 221
Address: 4328 Scorpius St., Orlando, FL 32816-2365
Email: chen.chen@crcv.ucf.edu
Web: https://www.crcv.ucf.edu/chenchen/

Lecture 5: Introduction to Deep Learning (1)



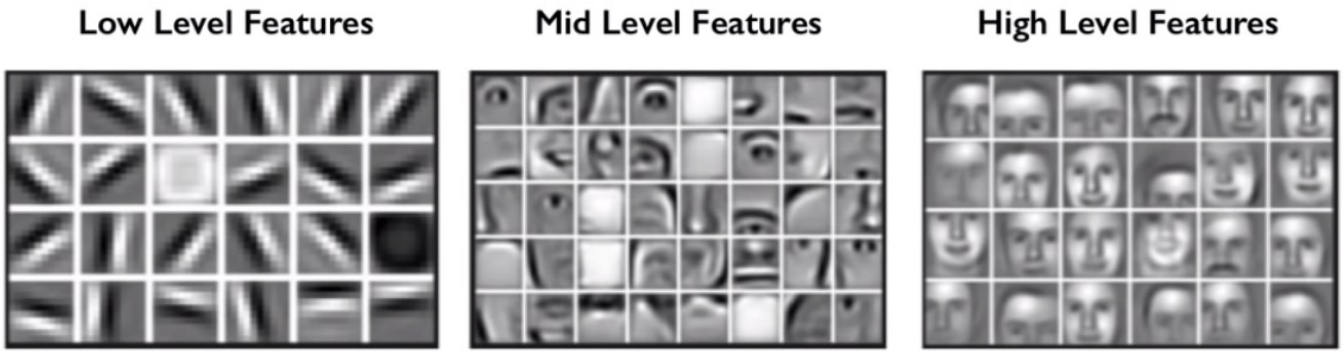
https://towardsdatascience.com/cousins-of-artificial-intelligence-dda4edc27b55

Why Deep Learning?



Hand engineered features are time consuming, brittle and not scalable in practice

Can we learn the **underlying features** directly from data?



Credit: Alexander Amini

Why Now?

1952 Stochastic Gradient Descent

1958 Perceptron
- Learnable Weights

1986 Backpropagation
- Multi-Layer Perceptron

1995 Deep Convolutional NN
- Digit Recognition

Neural Networks date back decades, so why the resurgence?

1. Big Data
- Larger Datasets
- Easier Collection & Storage

IMAGENET

WIKIPEDIA The Free Encyclopedia

2. Hardware
- Graphics Processing Units (GPUs)
- Massively Parallelizable

3. Software
- Improved Techniques
- New Models
- Toolboxes

TensorFlow

PYTORCH

Caffe2

Chainer

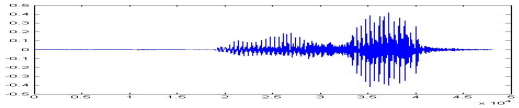
mxnet

4. Open resource (GitHub)

Credit: Lex Fridman

Machine Learning ≈ Looking for a Function

- Speech Recognition

$f($  $)=$ “How are you”

- Image Recognition

$f($  $)=$ “Cat”

- Playing Go

$f($  $)=$ “5-5” (next move)

- Dialogue System

$f($ “Hi” $)=$ “Hello”

(what the user said) (system response)

Credit: The following slides are adapted from Hung-yi Lee, Andrew, Stanford course

Framework

Image Recognition:

$f($  $) =$ "cat"

A set of function

Model

$f_1, f_2 \cdots$

$f_1($  $) =$ "cat"

$f_2($  $) =$ "money"

$f_1($  $) =$ "dog"

$f_2($  $) =$ "snake"

Framework

Image Recognition:

$f($  $) =$ "cat"

A set of function

Model

$f_1, f_2 \cdots$

Goodness of function f

Training Data

$f_1($  $) =$ "cat"

$f_2($  $) =$ "money"

Better!

$f_1($  $) =$ "dog"

$f_2($  $) =$ "snake"

Supervised Learning

function input:   

function output: "monkey" "cat" "dog"

Framework

Image Recognition:

$f($  $) =$ "cat"

Training

Testing

Model

A set of function

$f_1, f_2 \cdots$

Step 1

Goodness of function f

Step 2

Pick the "Best" Function

f^*

Step 3

Using f^*

"cat"

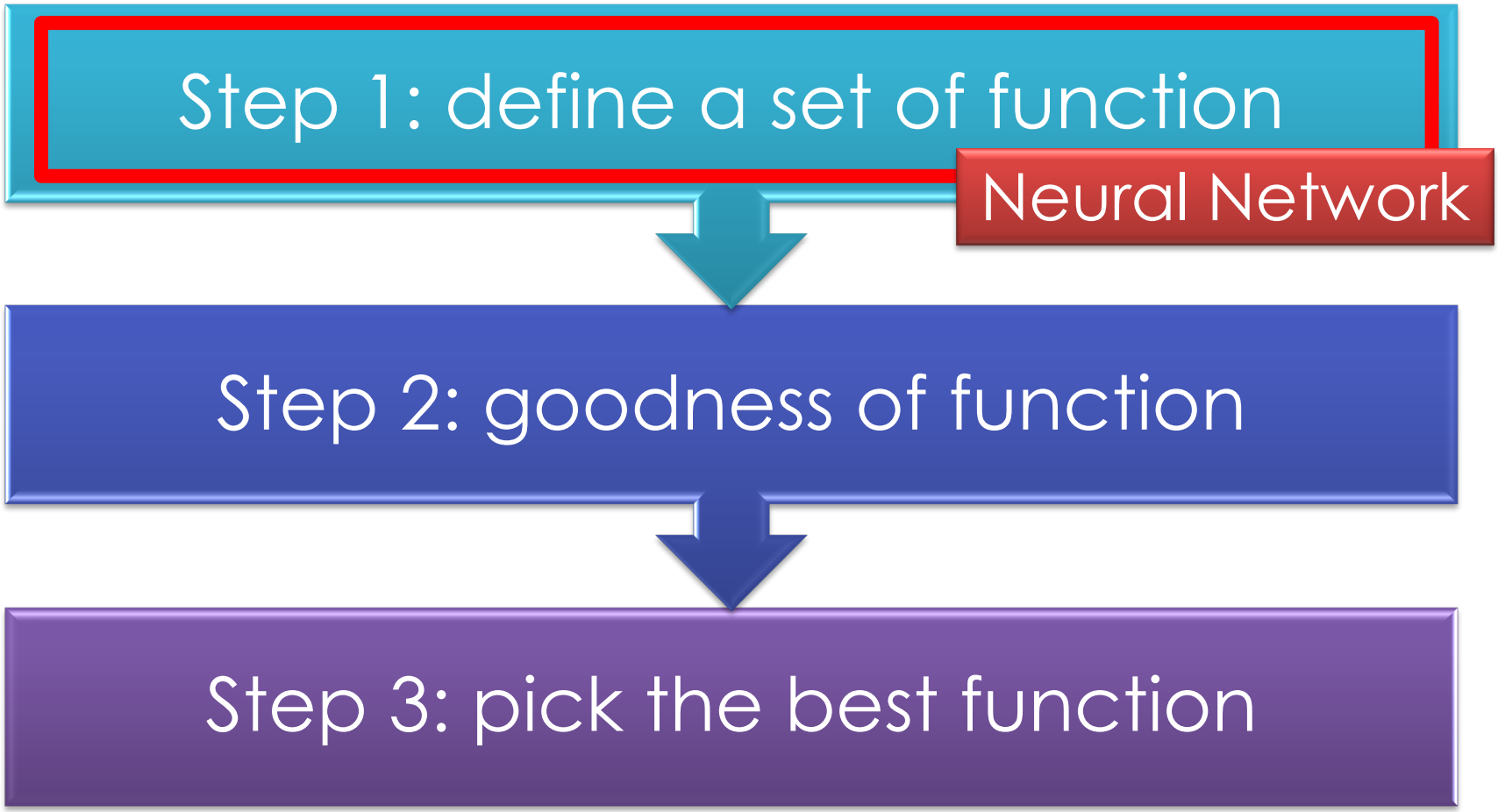
Training Data

"monkey"

"cat"

"dog"

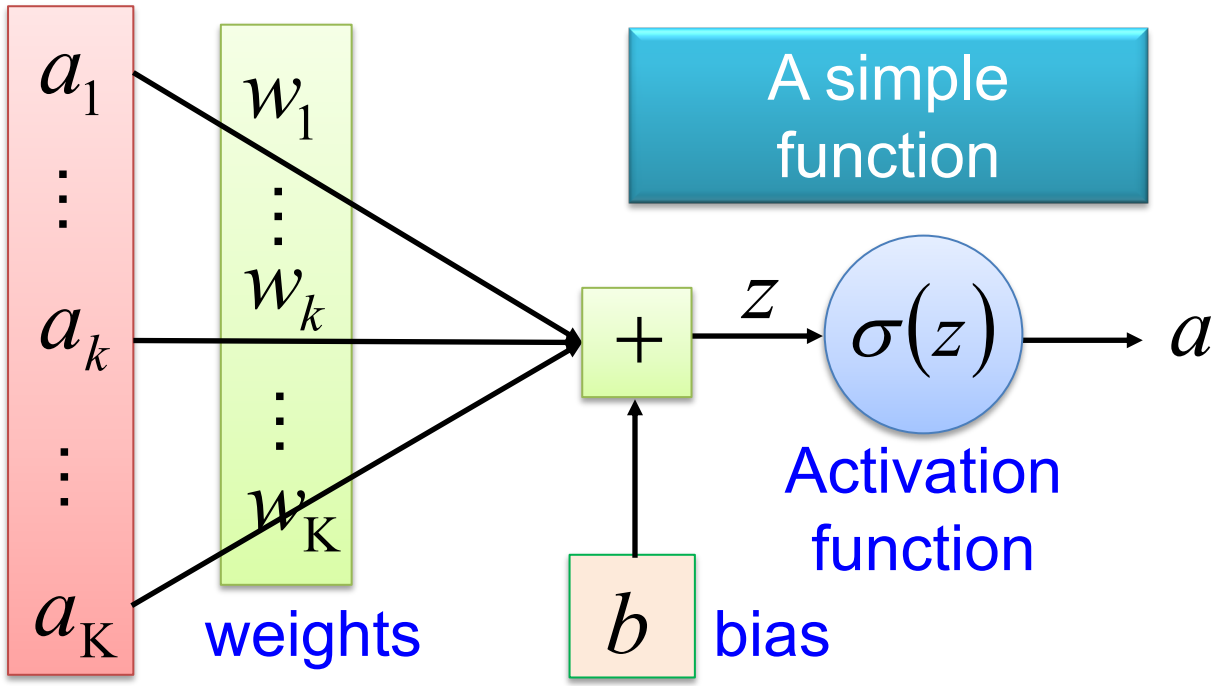
Three Steps for Deep Learning



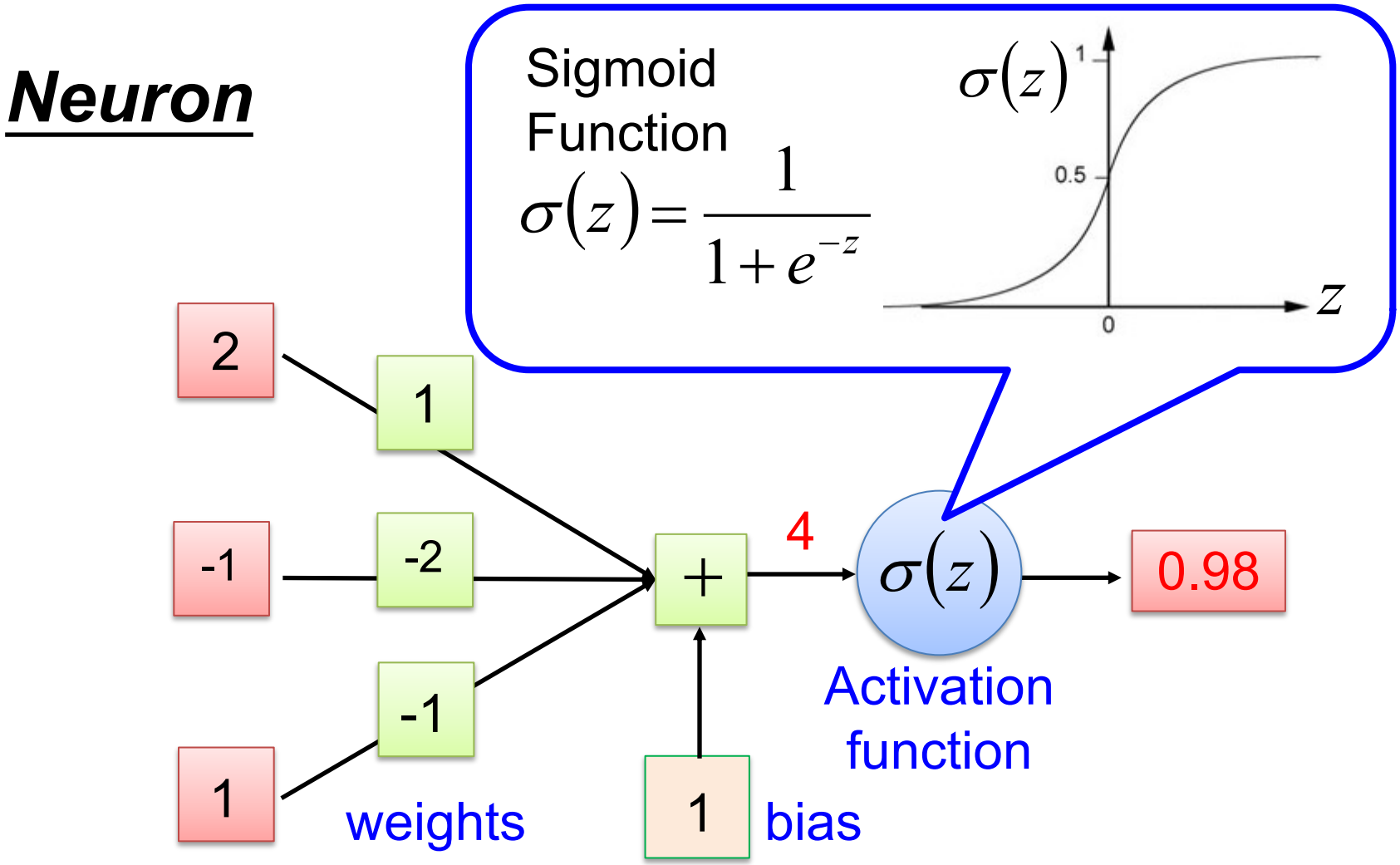
Neural Network

Neuron

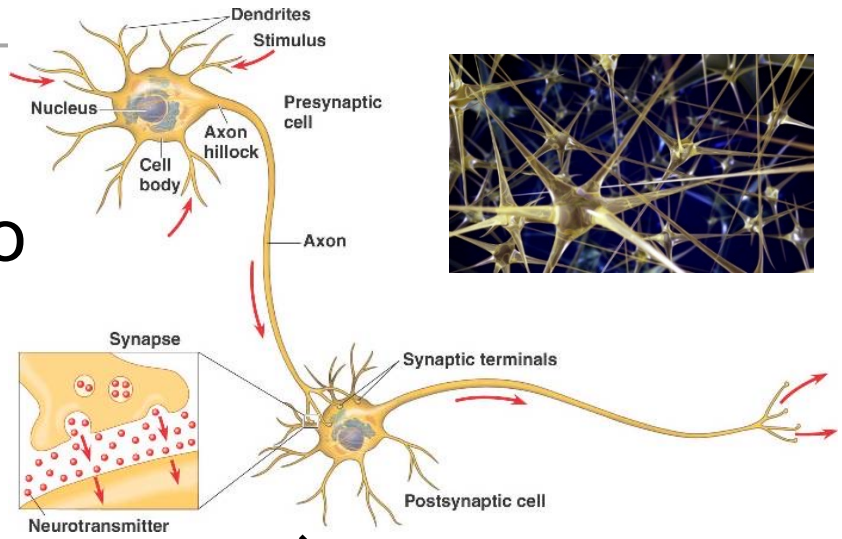
$$z = a_1 w_1 + \cdots + a_k w_k + \cdots + a_K w_K + b$$



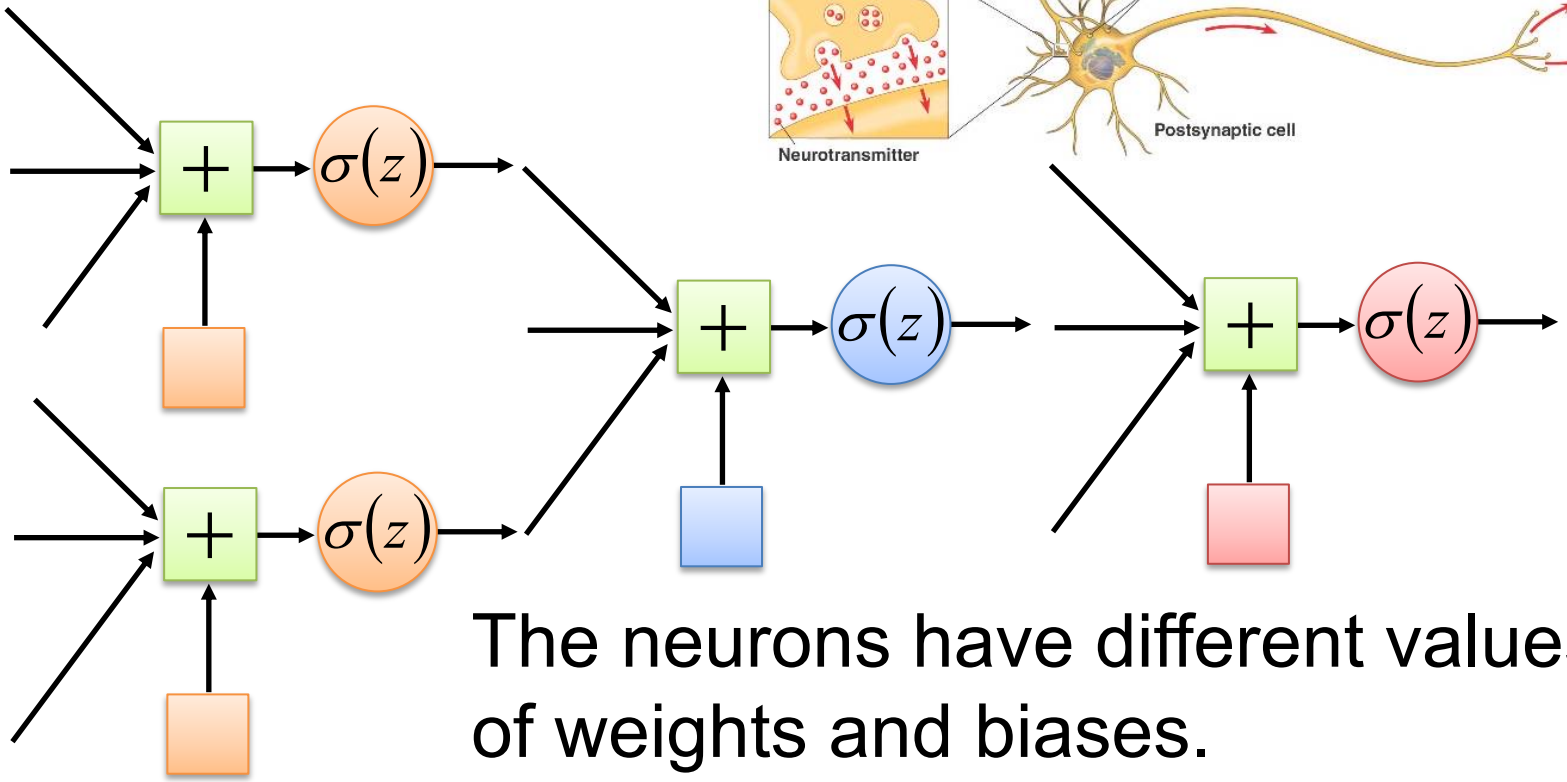
Neural Network



Neural Network



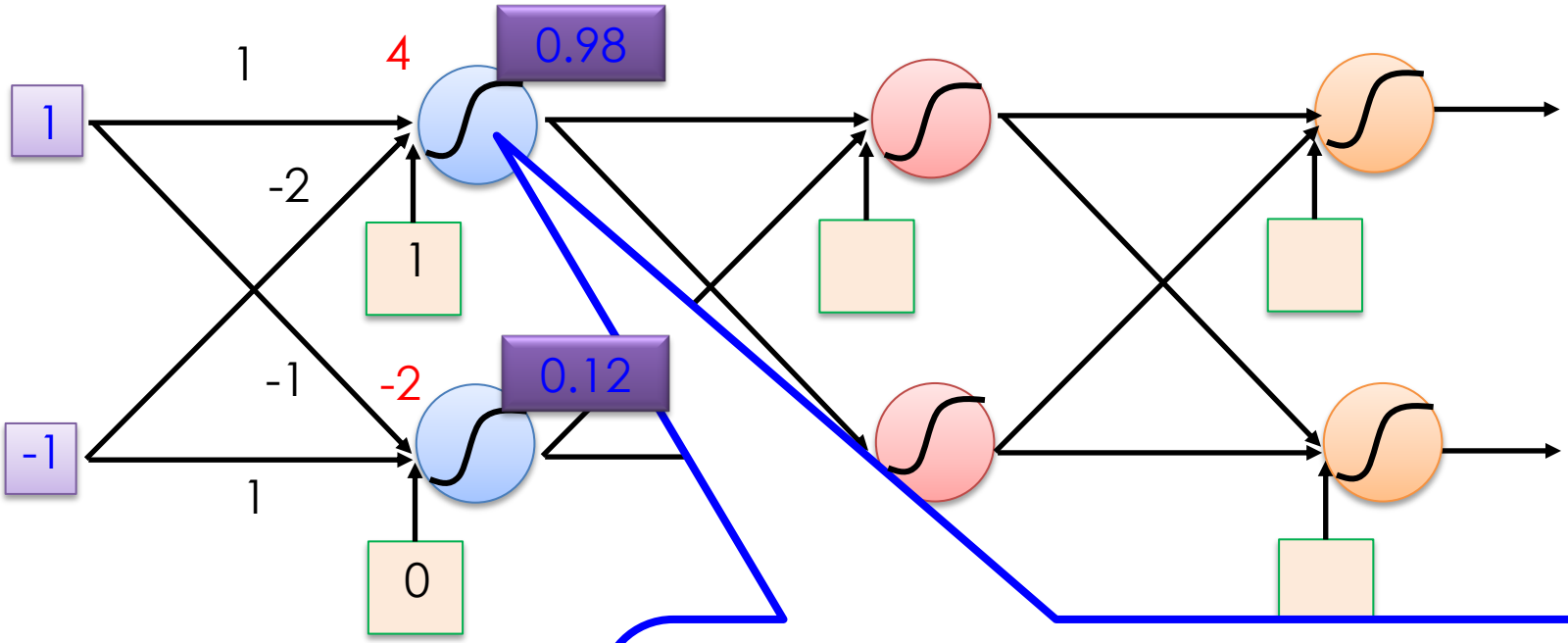
Different connections lead to different network structures



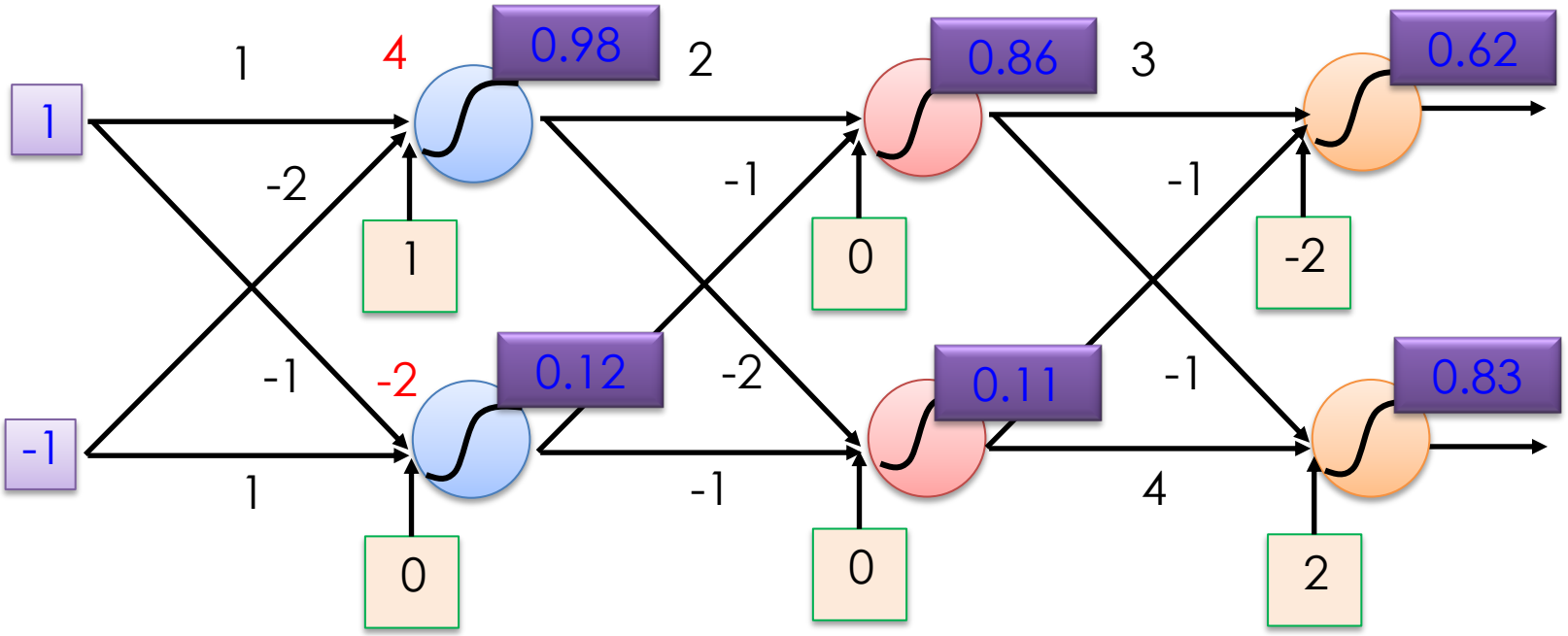
The neurons have different values of weights and biases.

Weights and biases are network parameters θ

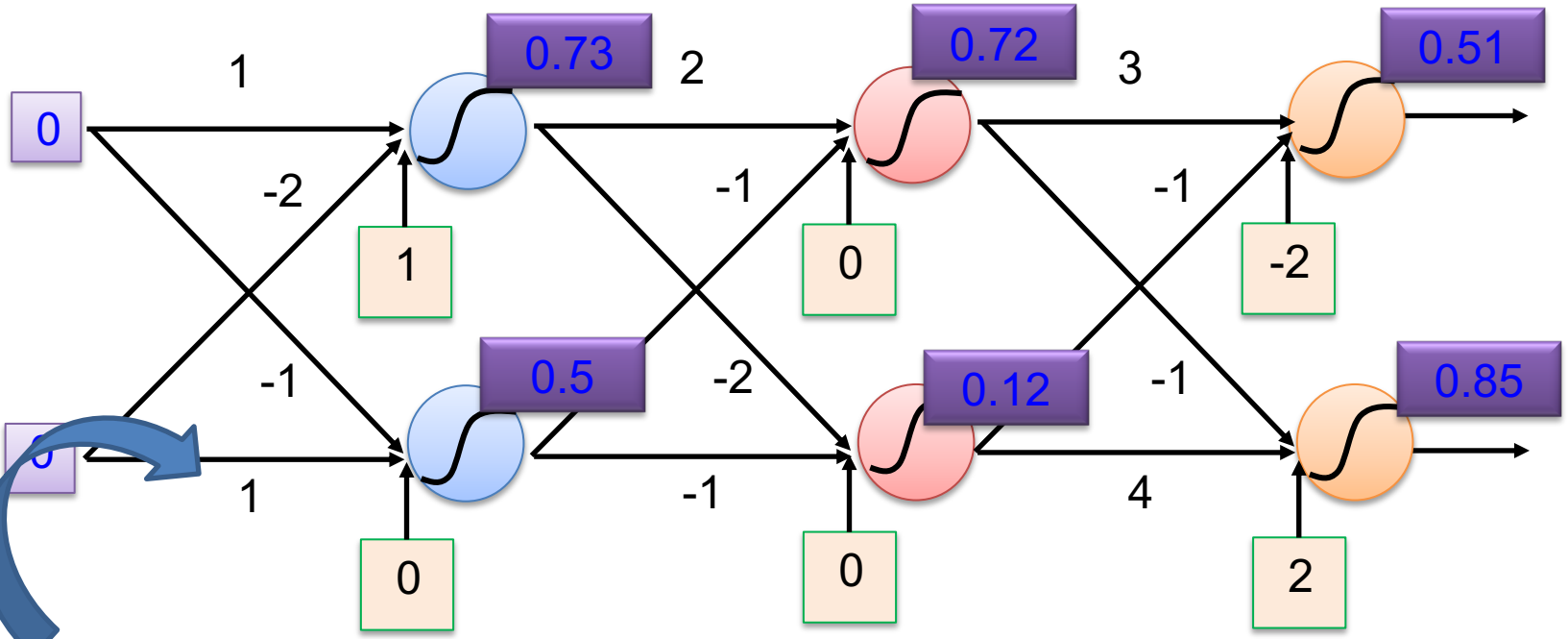
Fully Connected Feedforward Network



Fully Connected Feedforward Network



Fully Connected Feedforward Network



This is a function.

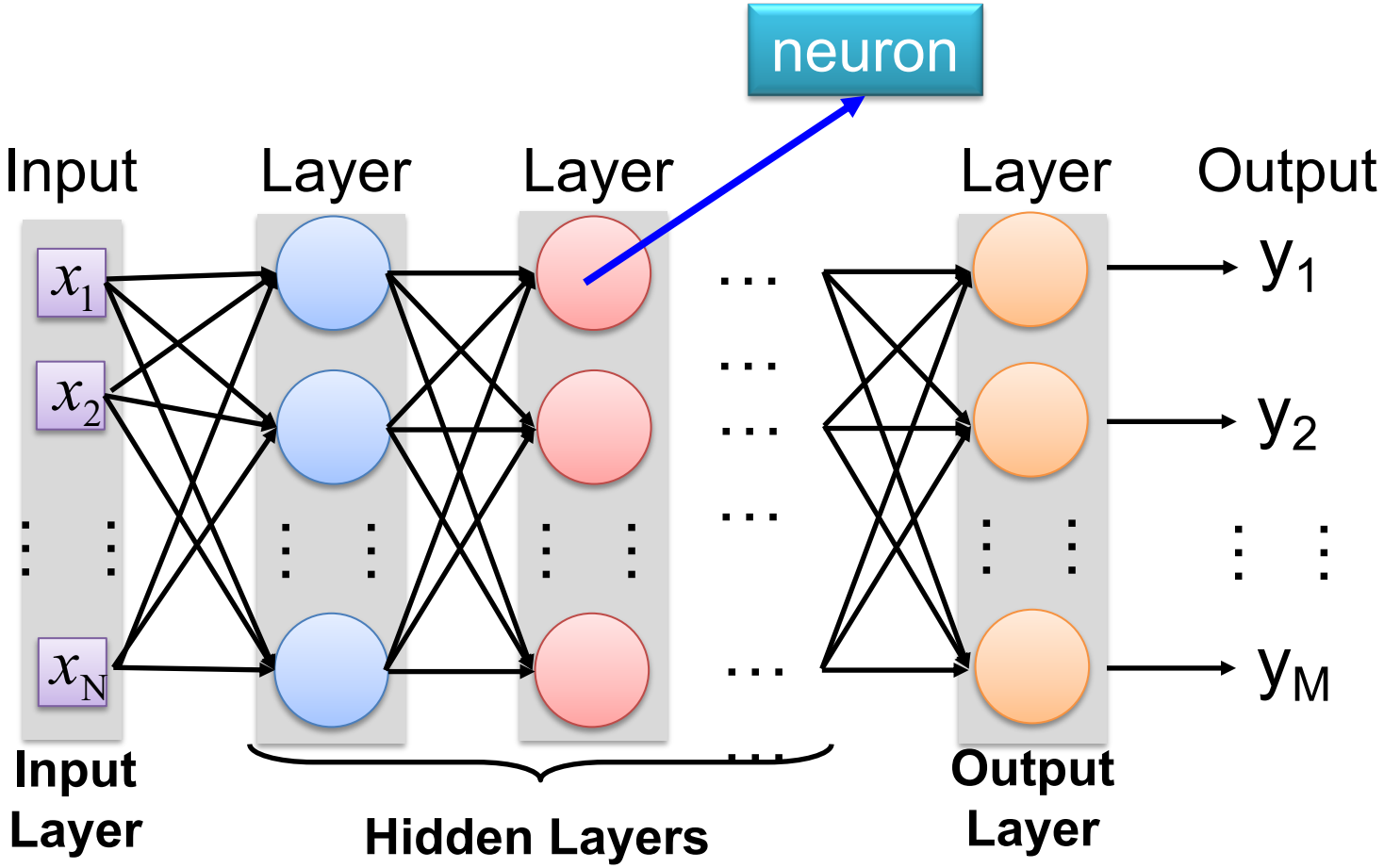
Input vector, output vector

$$f\left(\begin{bmatrix}1\\-1\end{bmatrix}\right)=\begin{bmatrix}0.62\\0.83\end{bmatrix} \quad f\left(\begin{bmatrix}0\\0\end{bmatrix}\right)=\begin{bmatrix}0.51\\0.85\end{bmatrix}$$

Given parameters θ, define a function

Given network structure, define ***a function set***

Fully Connected Feedforward Network

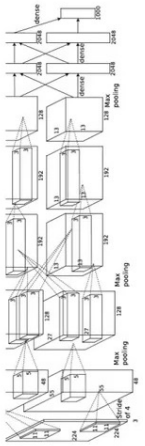


Deep means many hidden layers

Deep = Many hidden layers

http://cs231n.stanford.edu/slides/winter1516_lecture8.pdf

8 layers

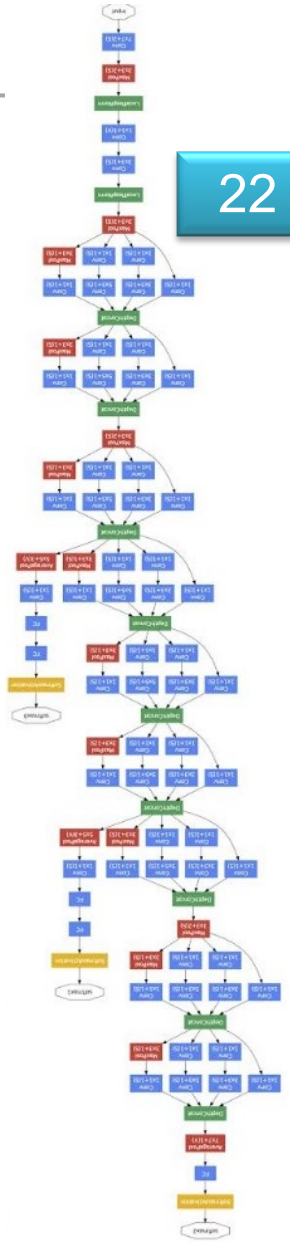


AlexNet (2012)

19 layers

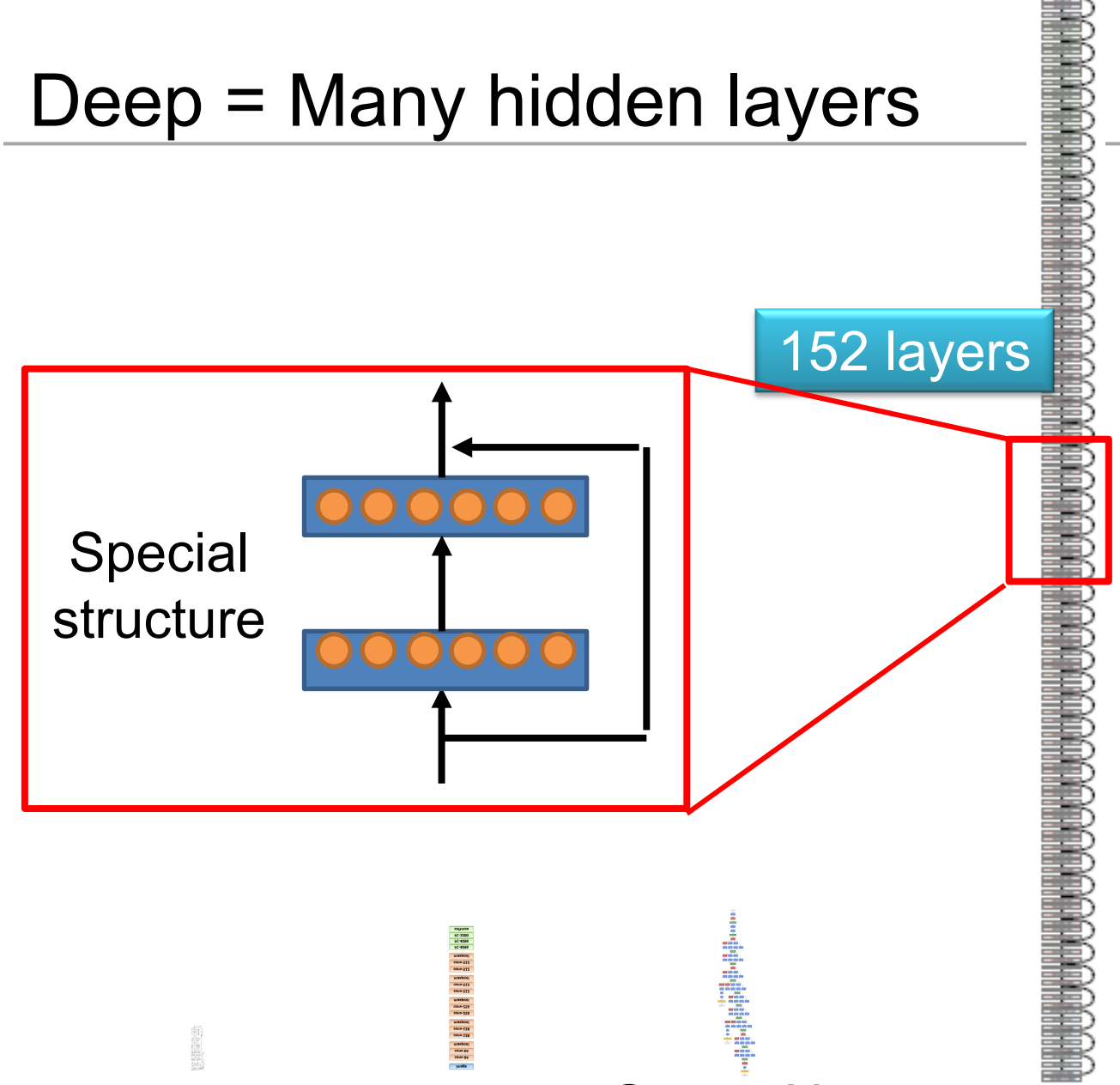
VGG (2014)

22 layers



GoogleNet (2014)

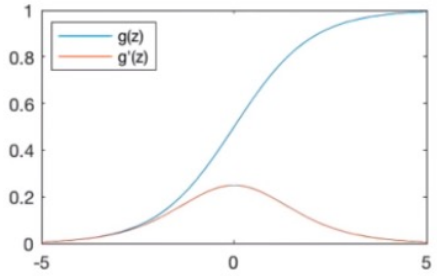
Deep = Many hidden layers



Activation Function

Common Activation Functions

Sigmoid Function

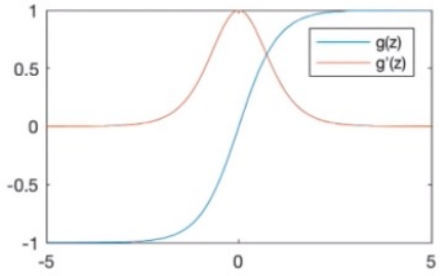


$$g(z) = \frac{1}{1 + e^{-z}}$$

$$g'(z) = g(z)(1 - g(z))$$

tf.nn.sigmoid(z)

Hyperbolic Tangent

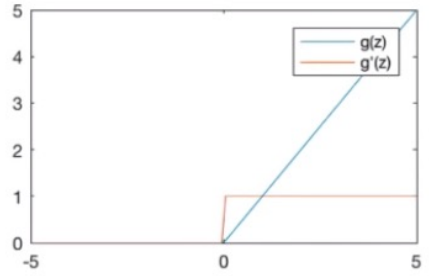


$$g(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

$$g'(z) = 1 - g(z)^2$$

tf.nn.tanh(z)

Rectified Linear Unit (ReLU)



$$g(z) = \max(0, z)$$

$$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases}$$

tf.nn.relu(z)

NOTE: All activation functions are non-linear

https://ml-cheatsheet.readthedocs.io/en/latest/activation_functions.html

More details about the activation functions: https://www.v7labs.com/blog/neural-networks-activation-functions

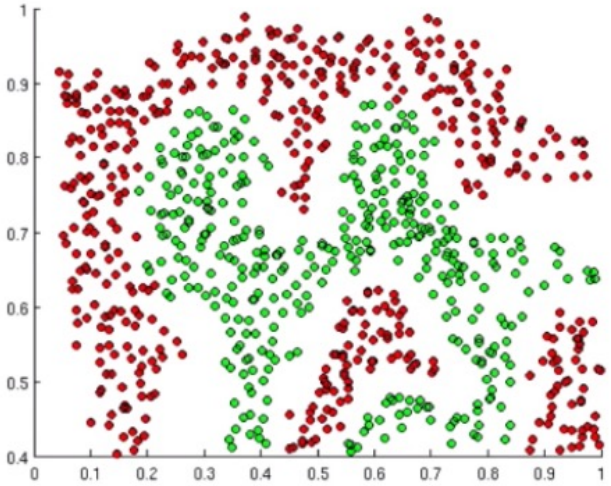
Activation Function

- Why non-linear activation function?

Activation Function

- Why non-linear activation function?

The purpose of activation functions is to ***introduce non-linearities*** *into the network*

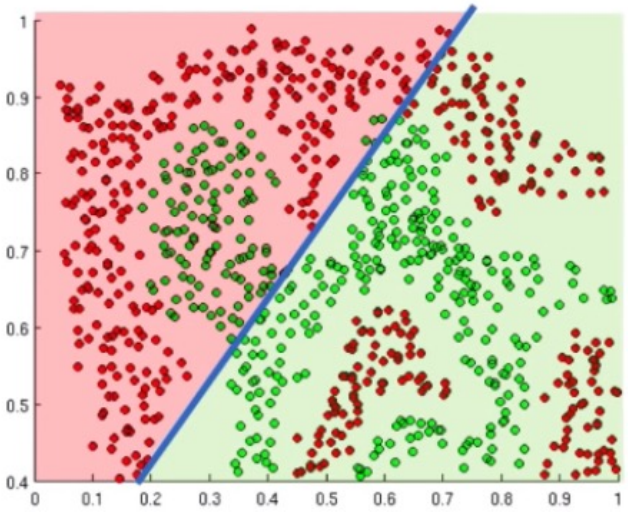


What if we wanted to build a Neural Network to distinguish green vs red points?

Activation Function

- Why non-linear activation function?

The purpose of activation functions is to ***introduce non-linearities*** *into the network*

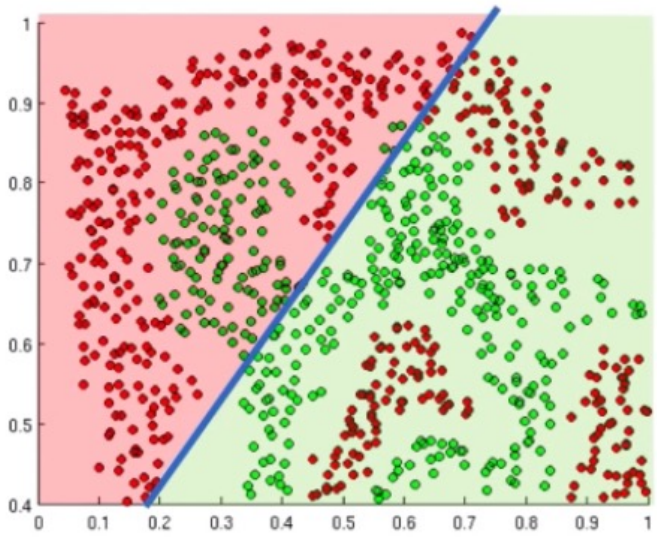


Linear Activation functions produce linear decisions no matter the network size

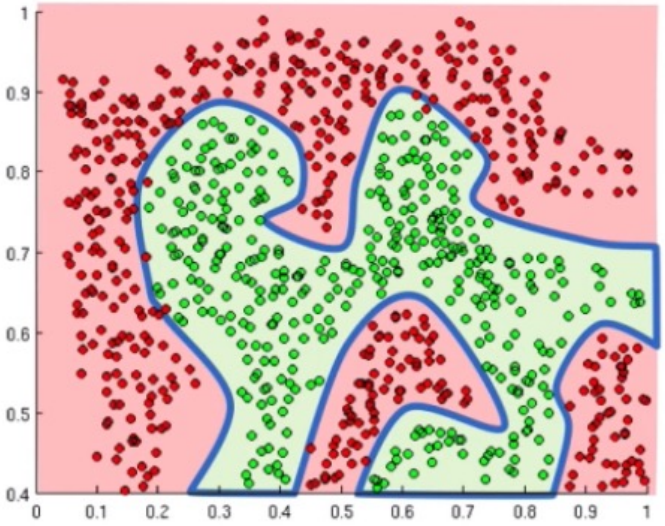
Activation Function

- Why non-linear activation function?

*The purpose of activation functions is to **introduce non-linearities** into the network*



Linear Activation functions produce linear decisions no matter the network size



Non-linearities allow us to approximate arbitrarily complex functions

Activation Function

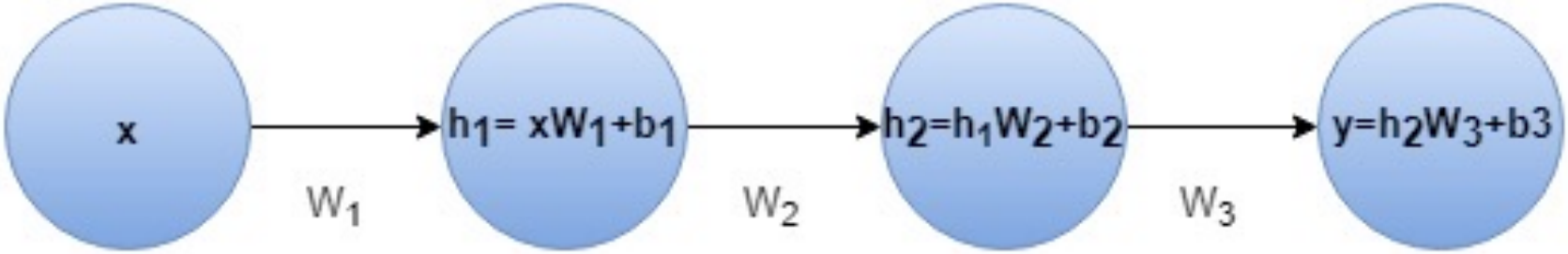
- Why non-linear activation function?

- **All layers of the neural network collapse into one**—with linear activation functions, no matter how many layers in the neural network, the last layer will be a linear function of the first layer (because a linear combination of linear functions is still a linear function). So a linear activation function turns the neural network into just one layer.

- A neural network with a linear activation function is simply a linear regression model. It has limited power and ability to handle complexity varying parameters of input data.

Activation Function

- Why non-linear activation function?



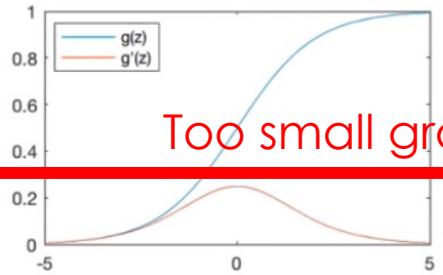

```
y = h2 * W3 + b3
  = (h1 * W2 + b2) * W3 + b3
  = h1 * W2 * W3 + b2 * W3 + b3
  = (x * W1 + b1) * W2 * W3 + b2 * W3 + b3
  = x * W1 * W2 * W3 + b1 * W2 * W3 + b2 * W3 + b3
  = x * W' + b'
```

A linear activation function turns the neural network into just one layer.

Activation Function

Common Activation Functions

Sigmoid Function



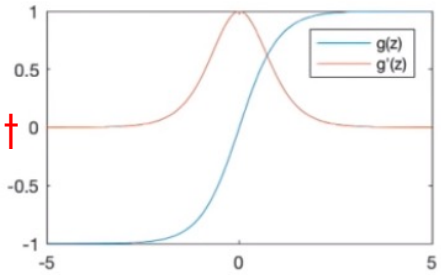
Too small gradient

$$g(z) = \frac{1}{1 + e^{-z}}$$

$$g'(z) = g(z)(1 - g(z))$$

`tf.nn.sigmoid(z)`

Hyperbolic Tangent

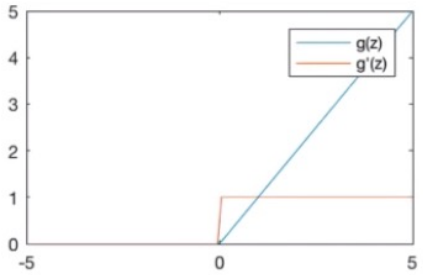


$$g(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

$$g'(z) = 1 - g(z)^2$$

`tf.nn.tanh(z)`

Rectified Linear Unit (ReLU)



$$g(z) = \max(0, z)$$

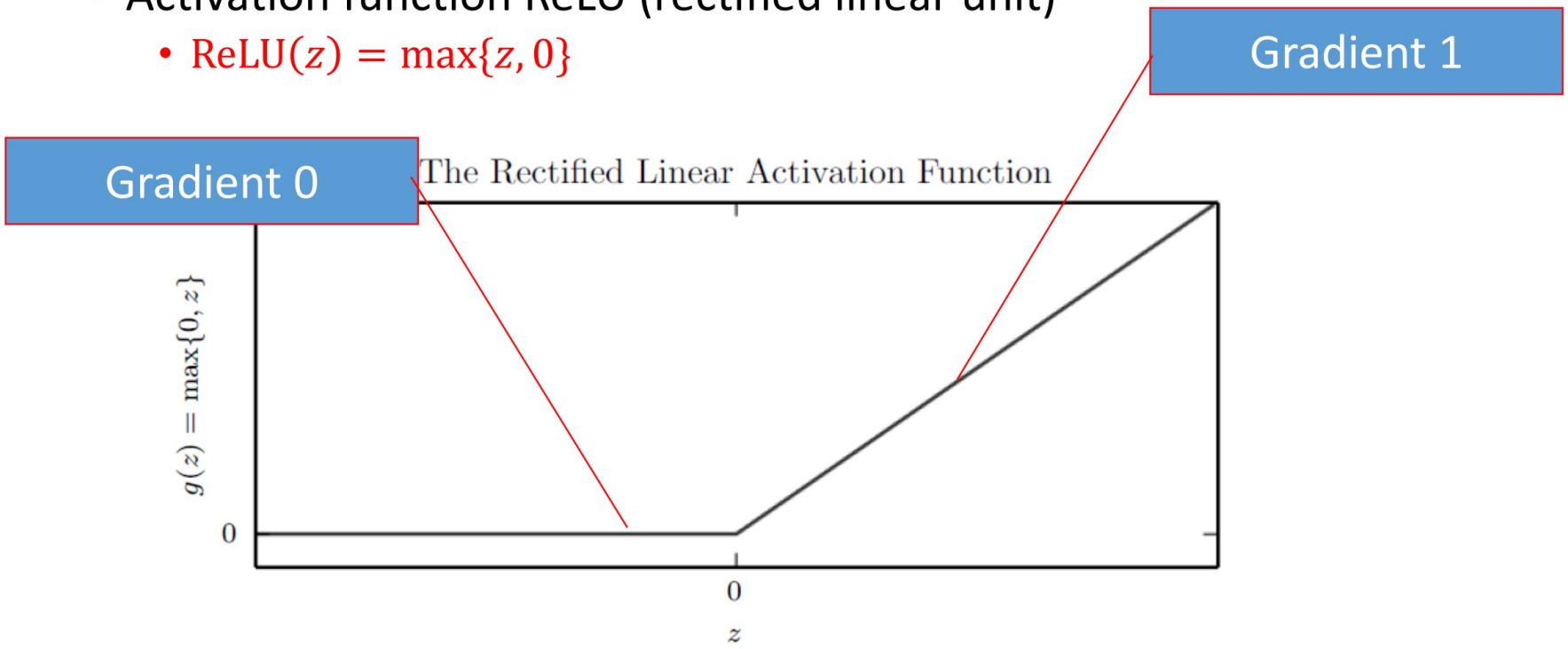
$$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases}$$

`tf.nn.relu(z)`

NOTE: All activation functions are non-linear

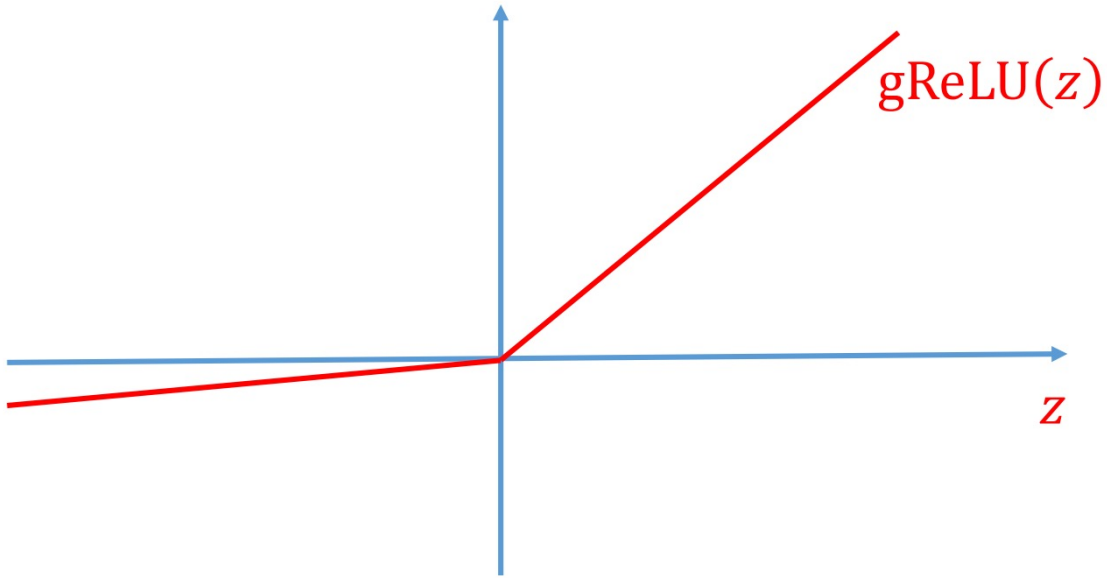
Activation Function

- Activation function ReLU (rectified linear unit)
  - $\mathrm{ReLU}(z) = \max\{z, 0\}$



Activation Function

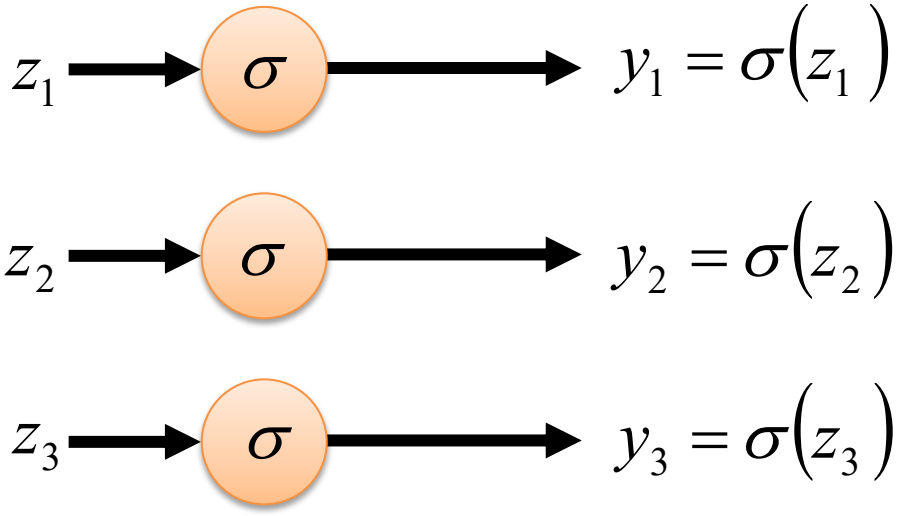
- Generalizations of ReLU $\mathrm{gReLU}(z) = \max\{z, 0\} + \alpha \min\{z, 0\}$
  - Leaky-ReLU$(z) = \max\{z, 0\} + 0.01 \min\{z, 0\}$
  - Parametric-ReLU(z): α learnable



Output Layer

- Softmax layer as the output layer

Ordinary Layer



In general, the output of network can be any value.

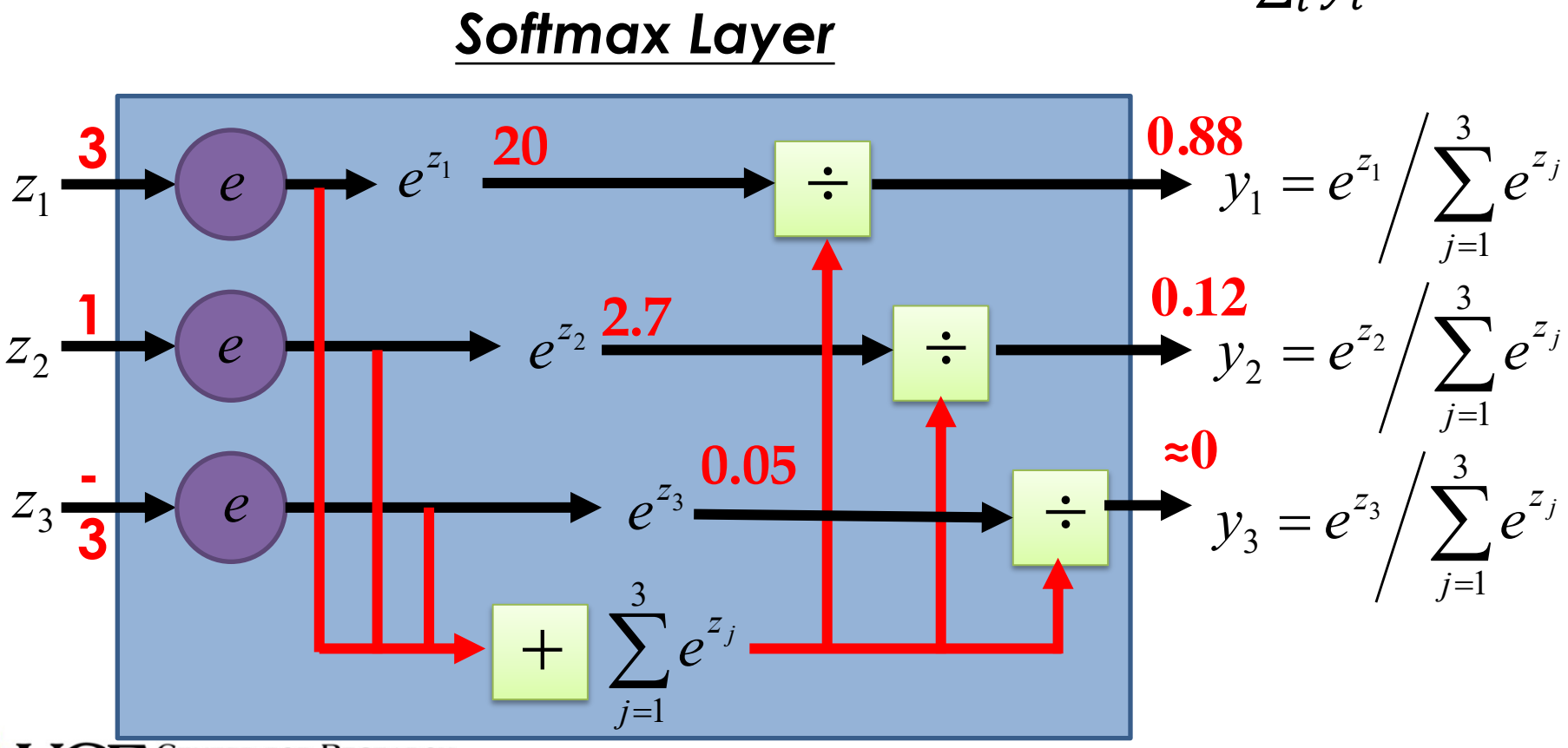
May not be easy to interpret

Output Layer

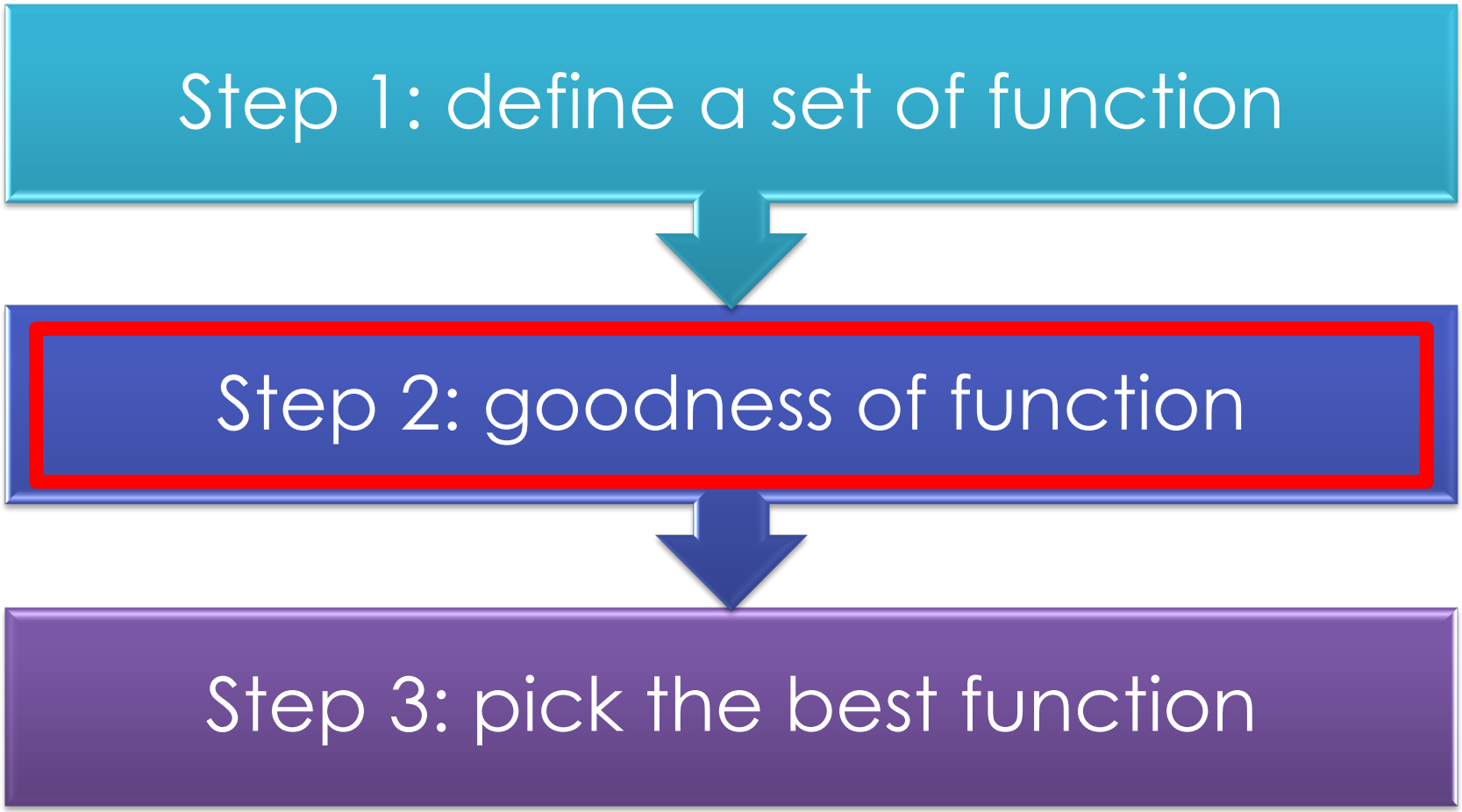
- Softmax layer as the output layer

Probability:

- $1 > y_i > 0$
- $\sum_i y_i = 1$

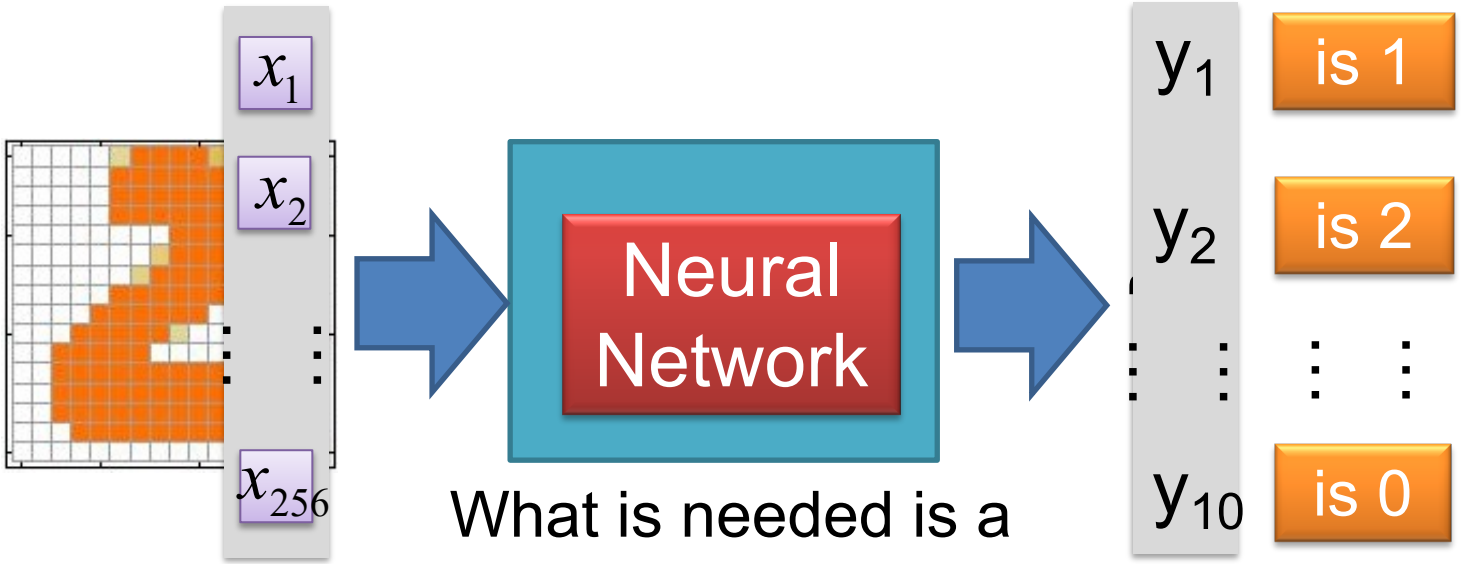


Three Steps for Deep Learning



Example Application

- Handwriting Digit Recognition



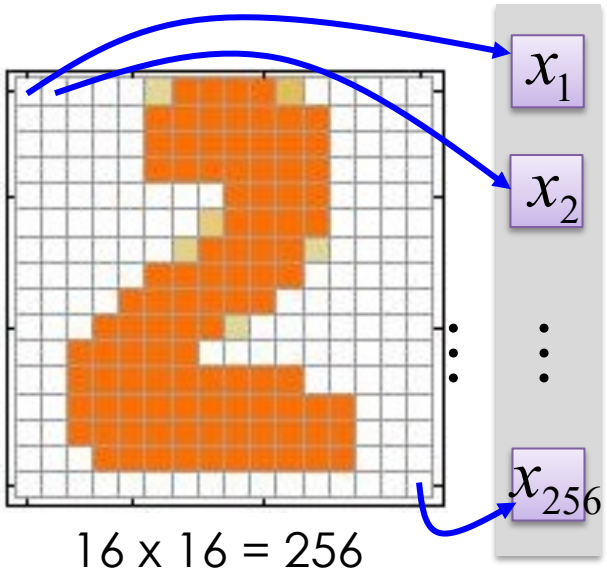
What is needed is a function ……

Input:
256-dim vector

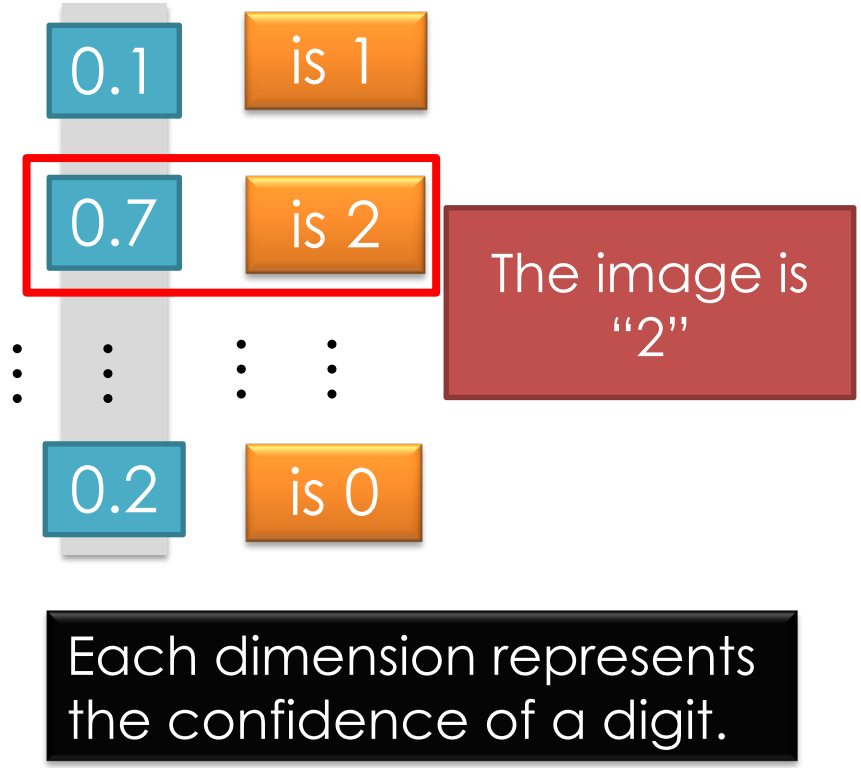
output:
10-dim vector

Example Application

Input

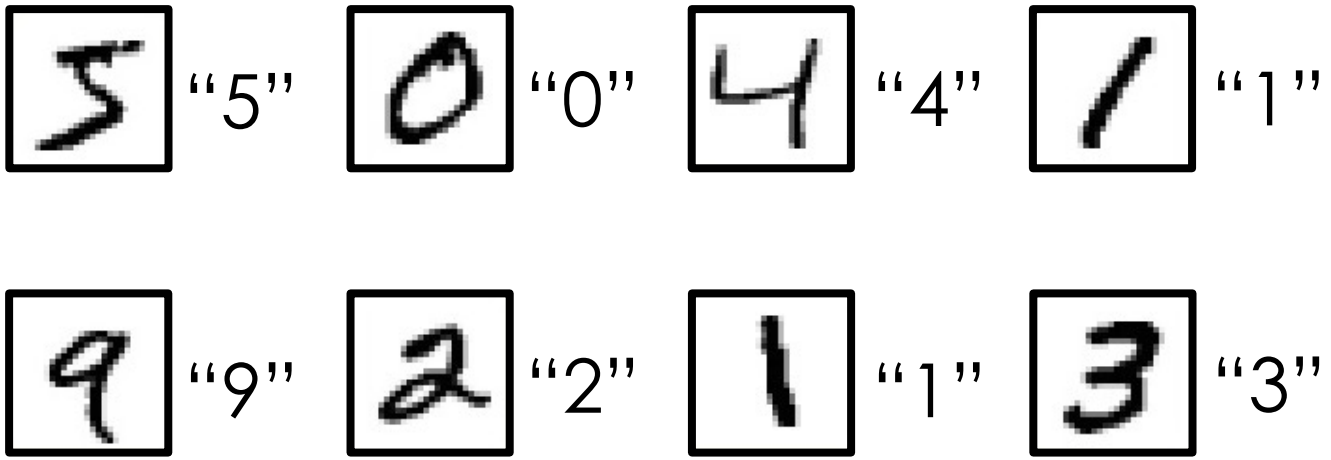


Output



Training Data

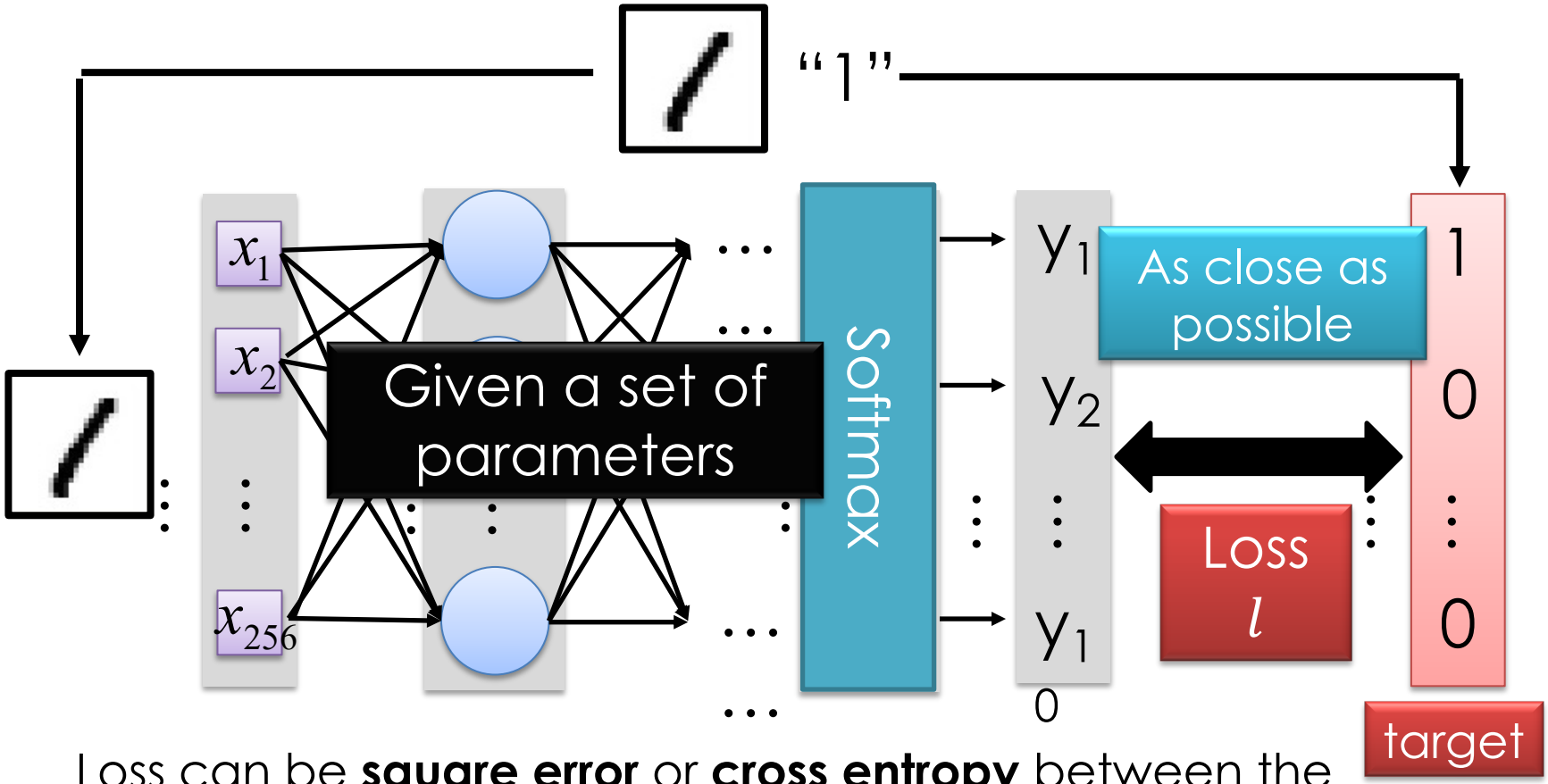
- Preparing training data: images and their labels



The learning target is defined on the training data.

Loss

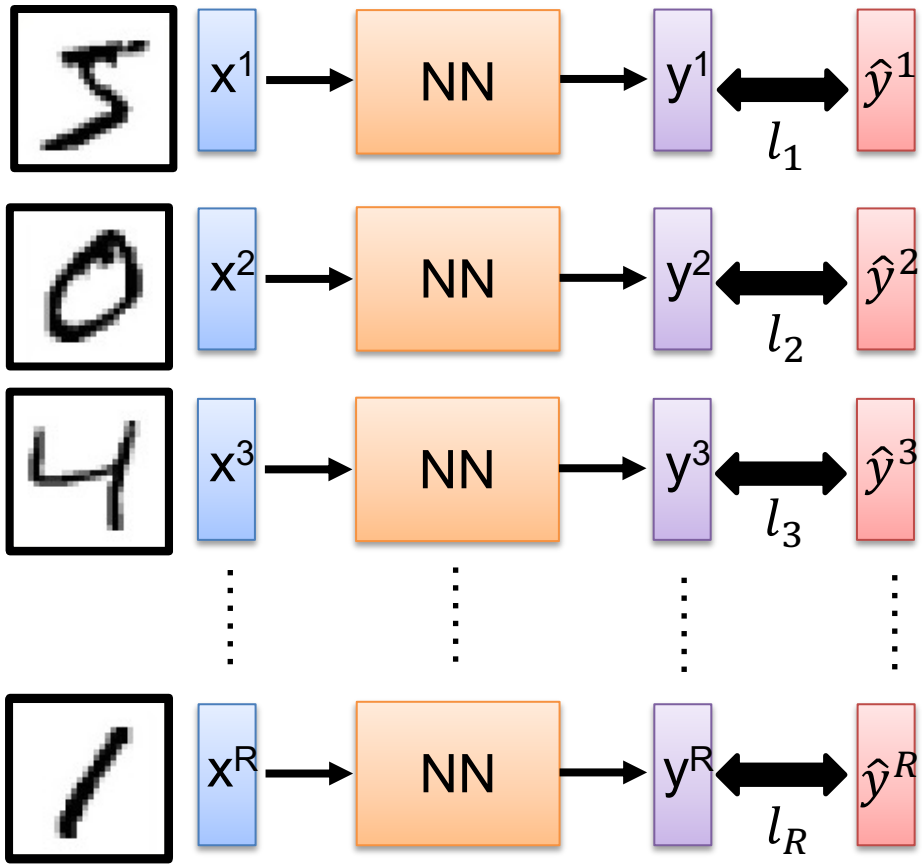
A good function should make the loss of all examples as small as possible.



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

Total Loss

For all training data …



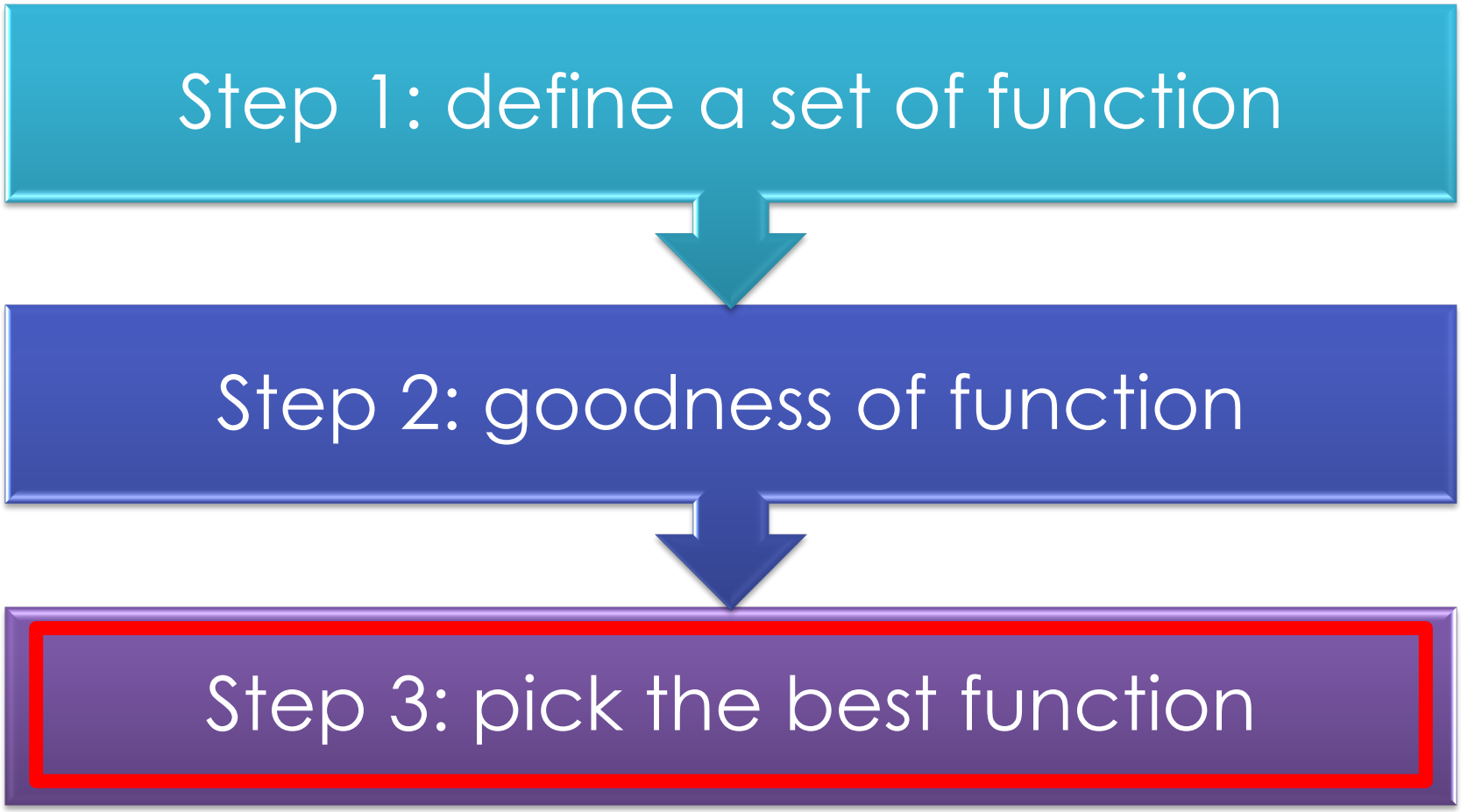
Total Loss: $L = \sum_{r=1}^{R} l_r$

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

Three Steps for Deep Learning



Convolutional Neural Network (CNN)

Widely used in image processing and computer vision

Why CNN for Image

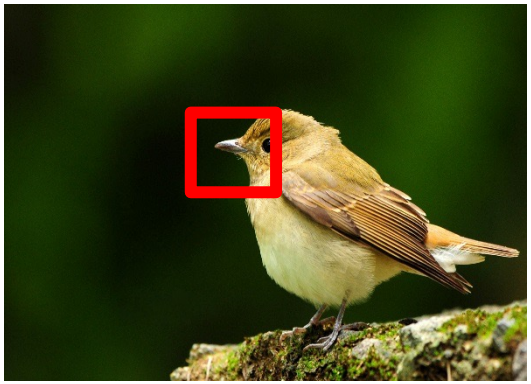
- Some patterns are much smaller than the whole image

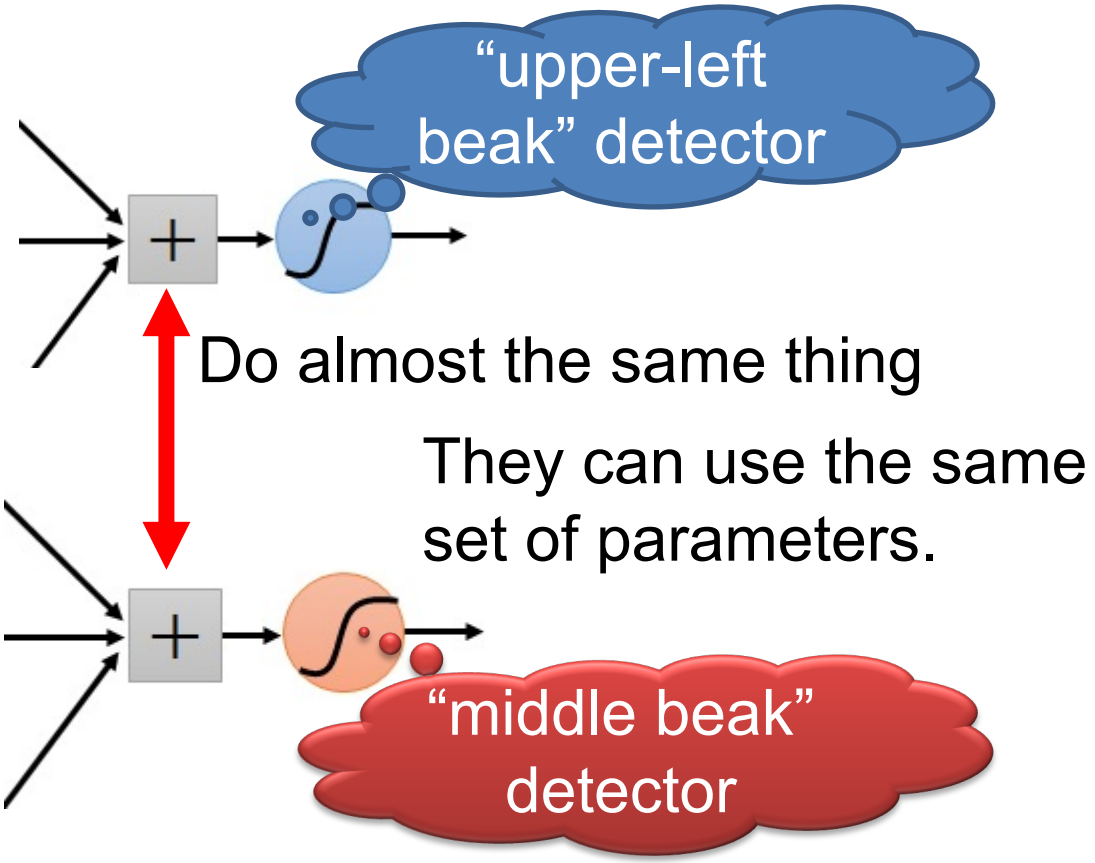
A neuron does not have to see the whole image to discover the pattern.

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

CNN for Image

Property 1

- Some patterns are much smaller than the whole

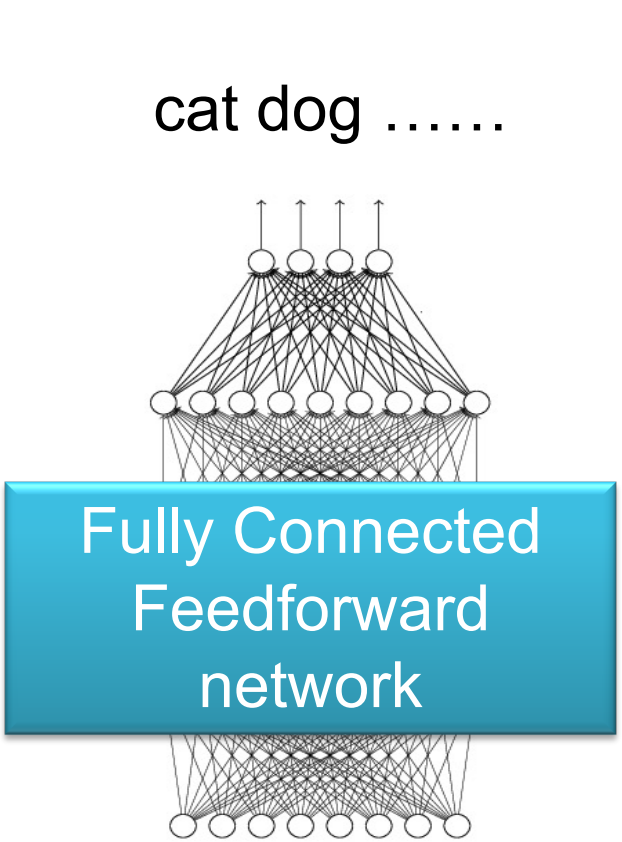
Property 2

- The same patterns appear in different regions.

Property 3

- Subsampling the pixels will not change the object

The whole CNN



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

Convolution

Max Pooling

Convolution

Max Pooling

Can repeat many times

Flatten

CNN – Convolution

Those are the network parameters to be learned.

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

Filter/kernel 1

Matrix

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

Filter 2

Matrix

⋮ ⋮

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

CNN – Convolution

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

Filter/Kernel 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

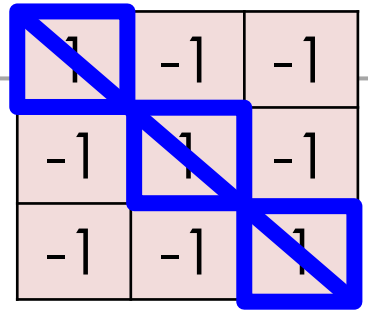
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

3 -3

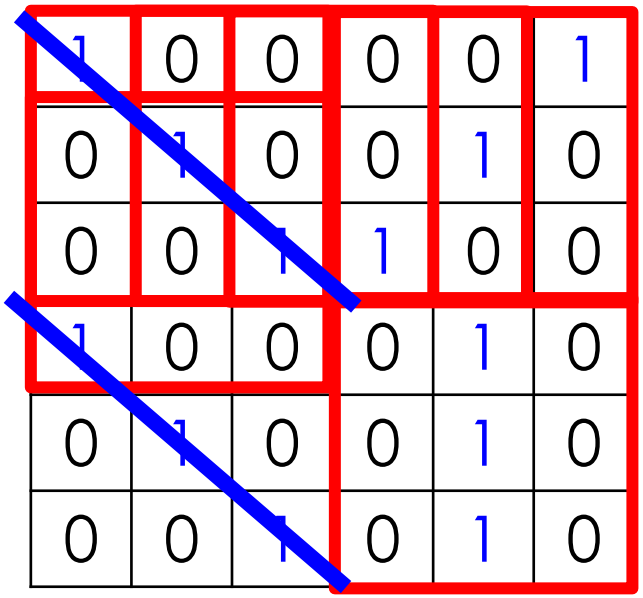
We set stride=1 below

CNN – Convolution

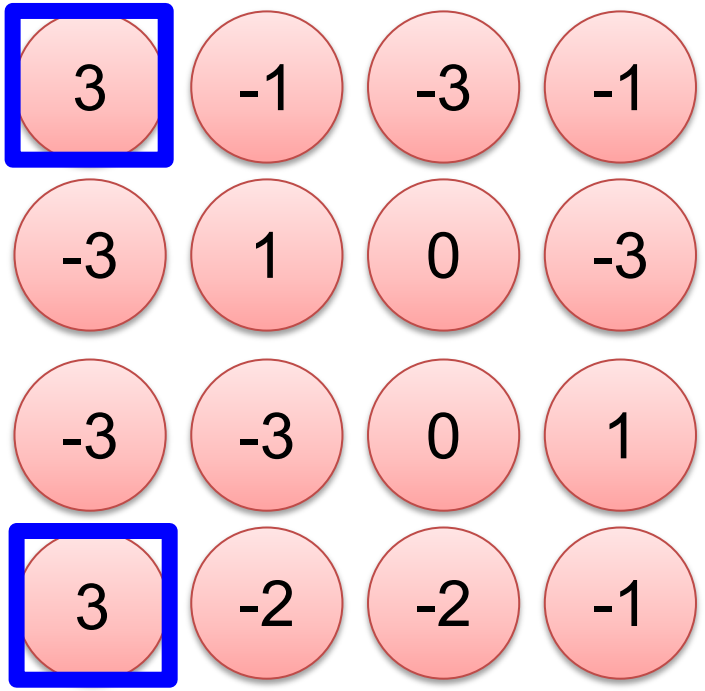


Filter 1

stride=1



6 x 6 image



Property 2

CNN – Convolution

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

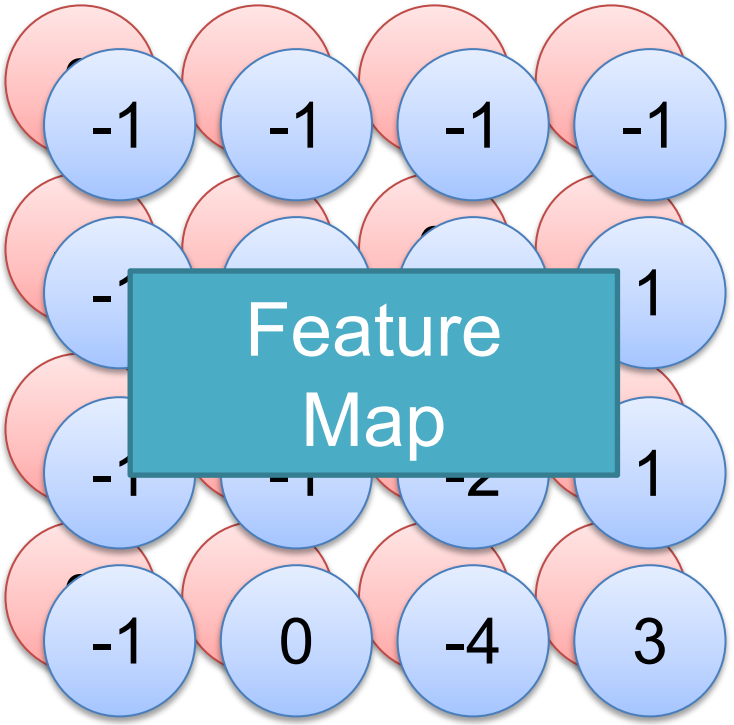
Filter 2

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

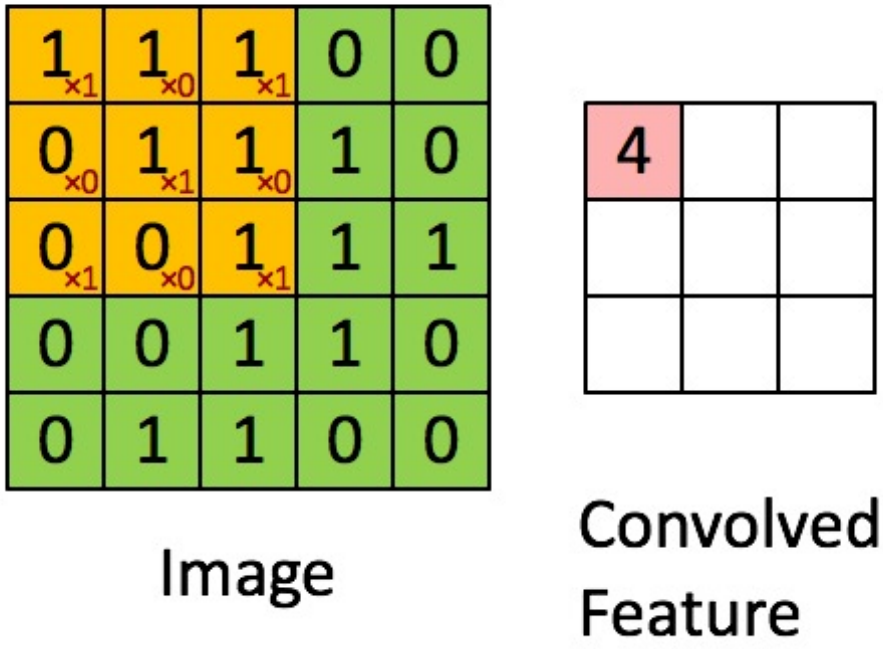


4 x 4 image (reduced size)

Demo

1	0	1
0	1	0
1	0	1

Filter/kernel

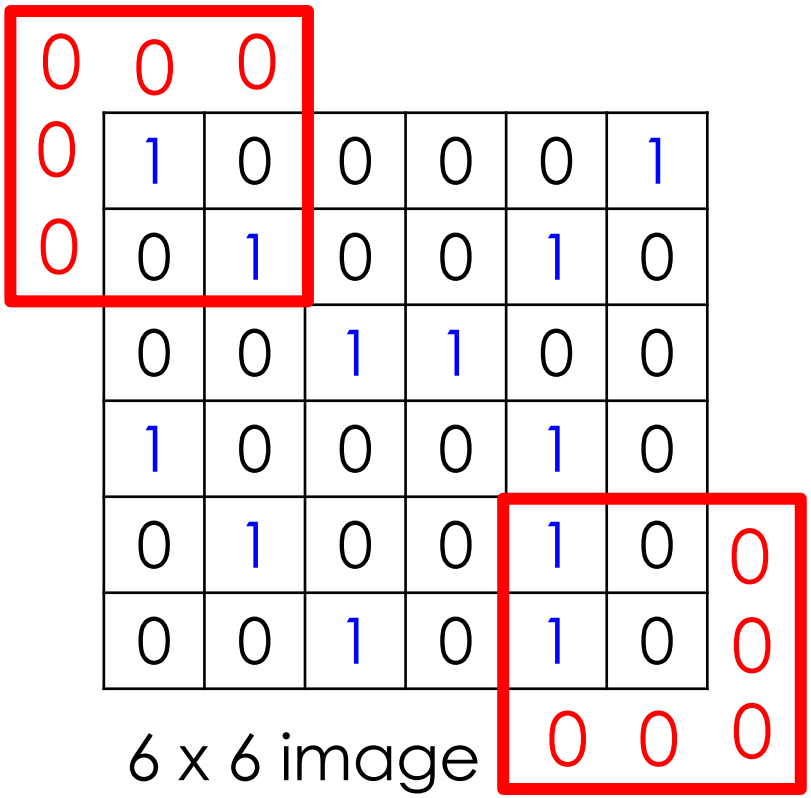


https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

CNN – Zero Padding

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

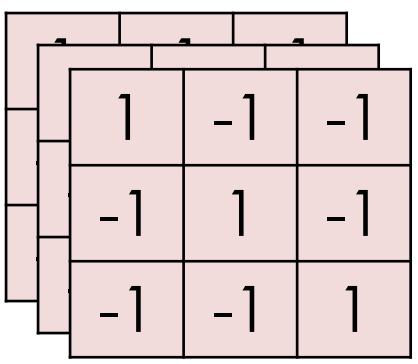
Filter 1



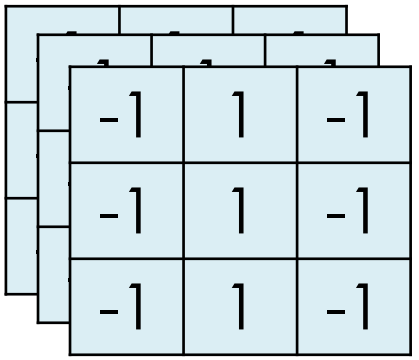
You will get another 6 x 6 image in this way

Zero padding

CNN – Color image

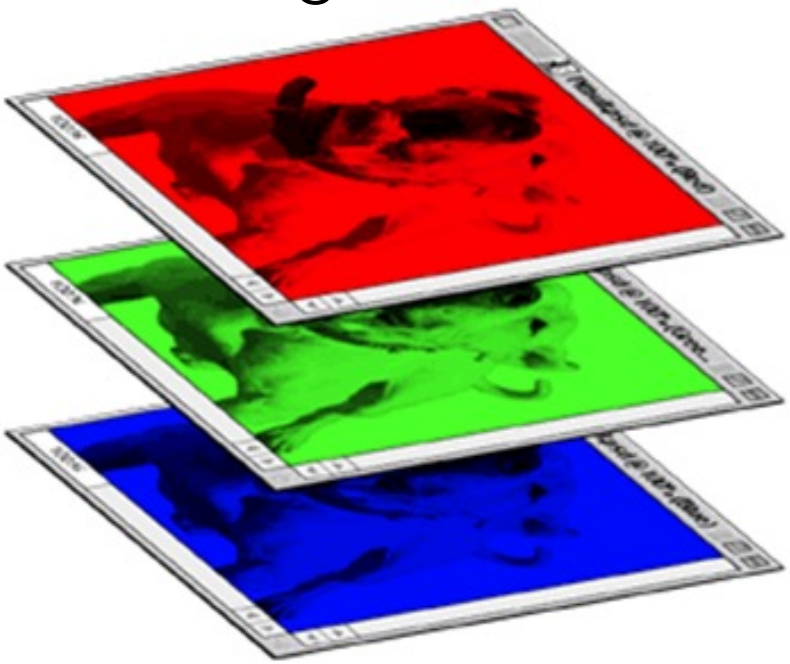


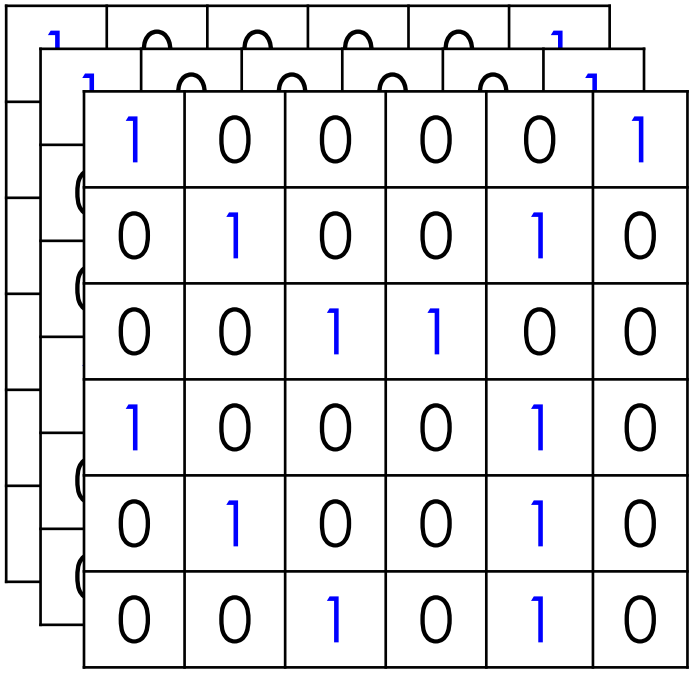
Filter 1

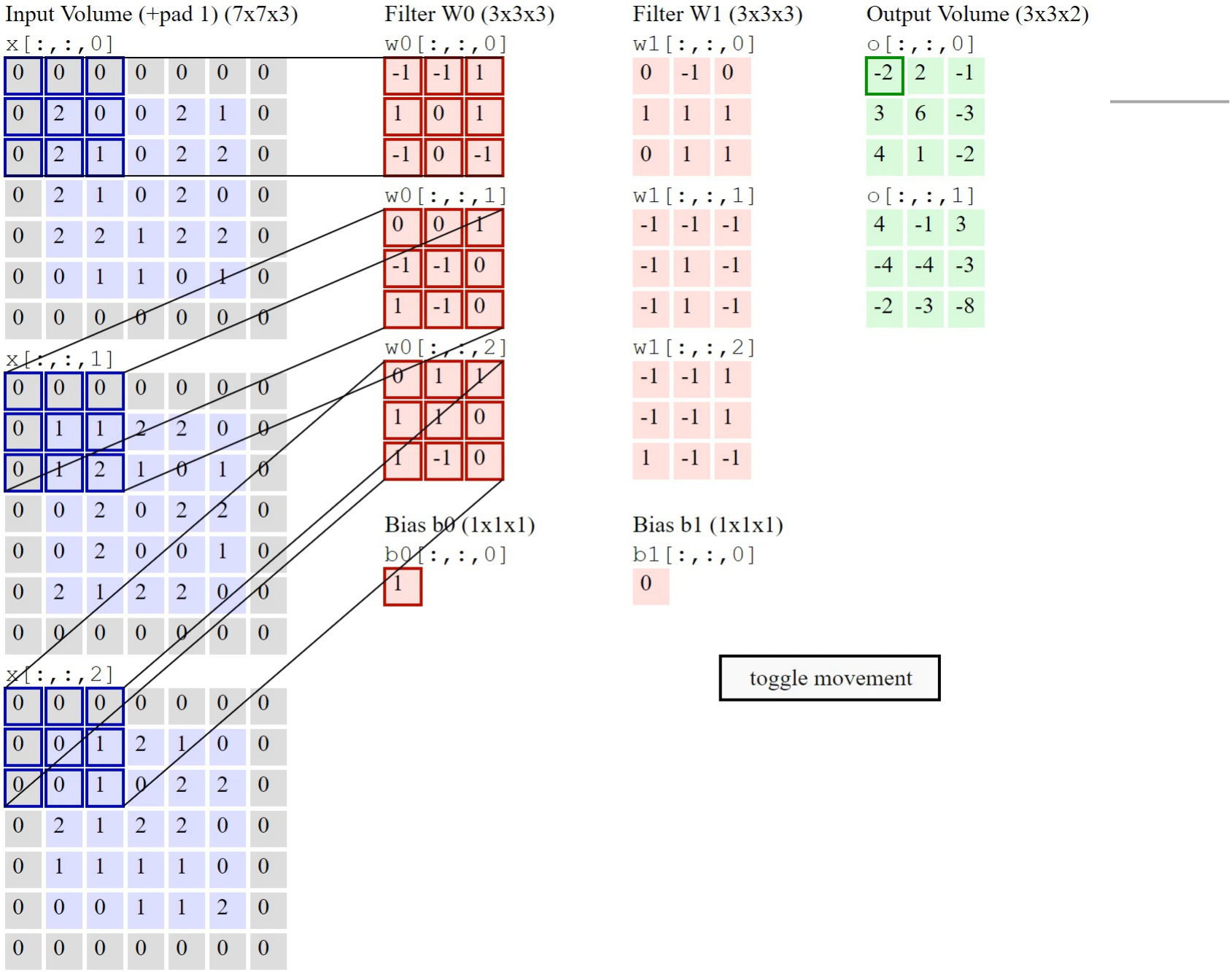


Filter 2

Color image



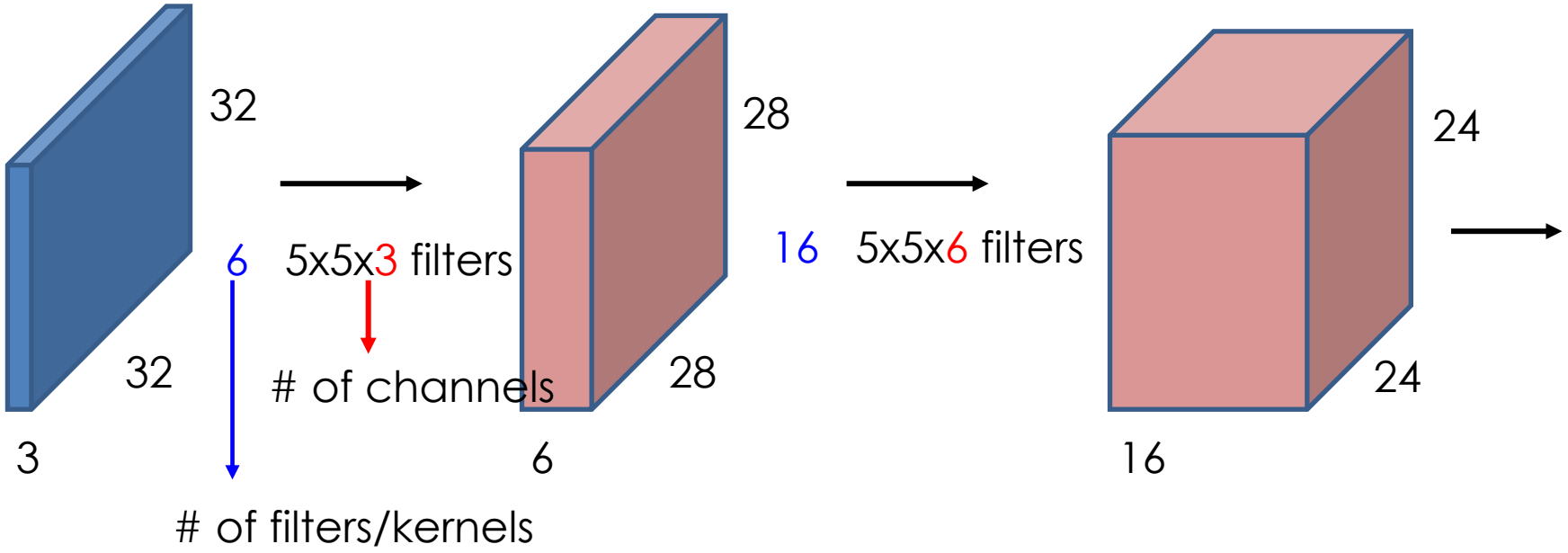




http://cs231n.github.io/convolutional-networks/

Convolutional Network

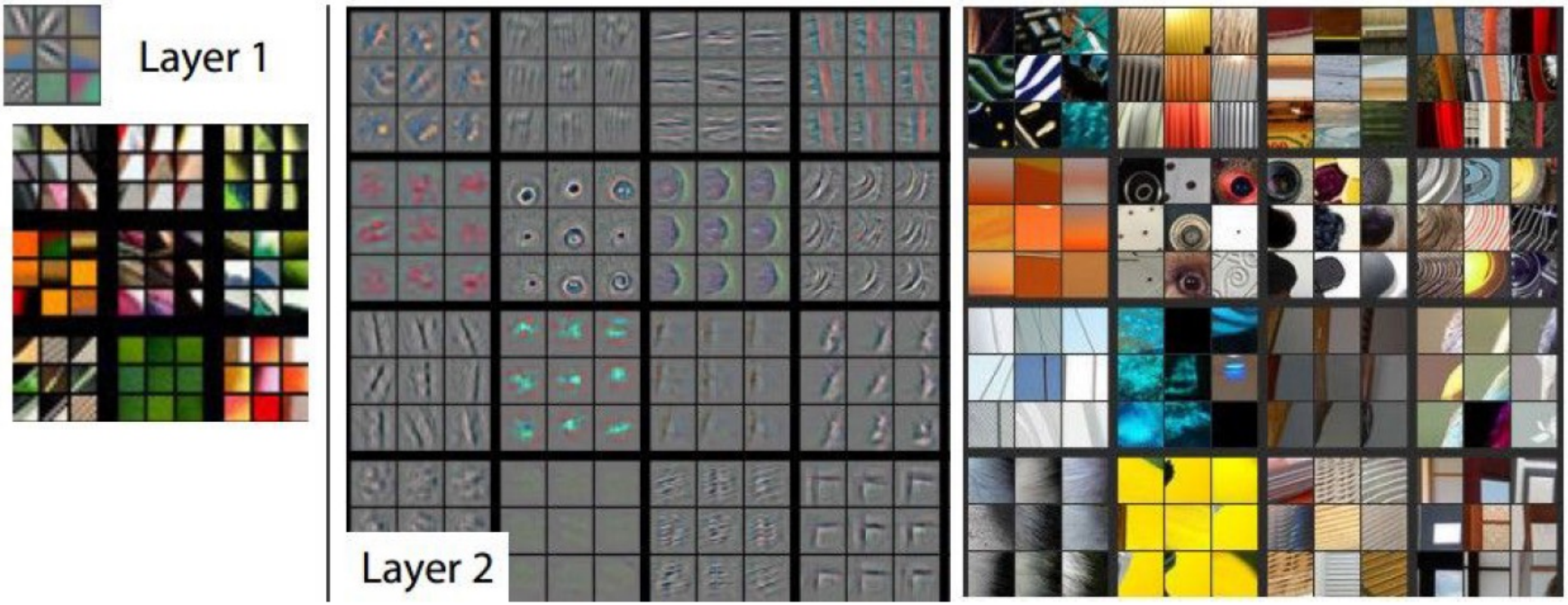
- Convolution network is a sequence of these layers



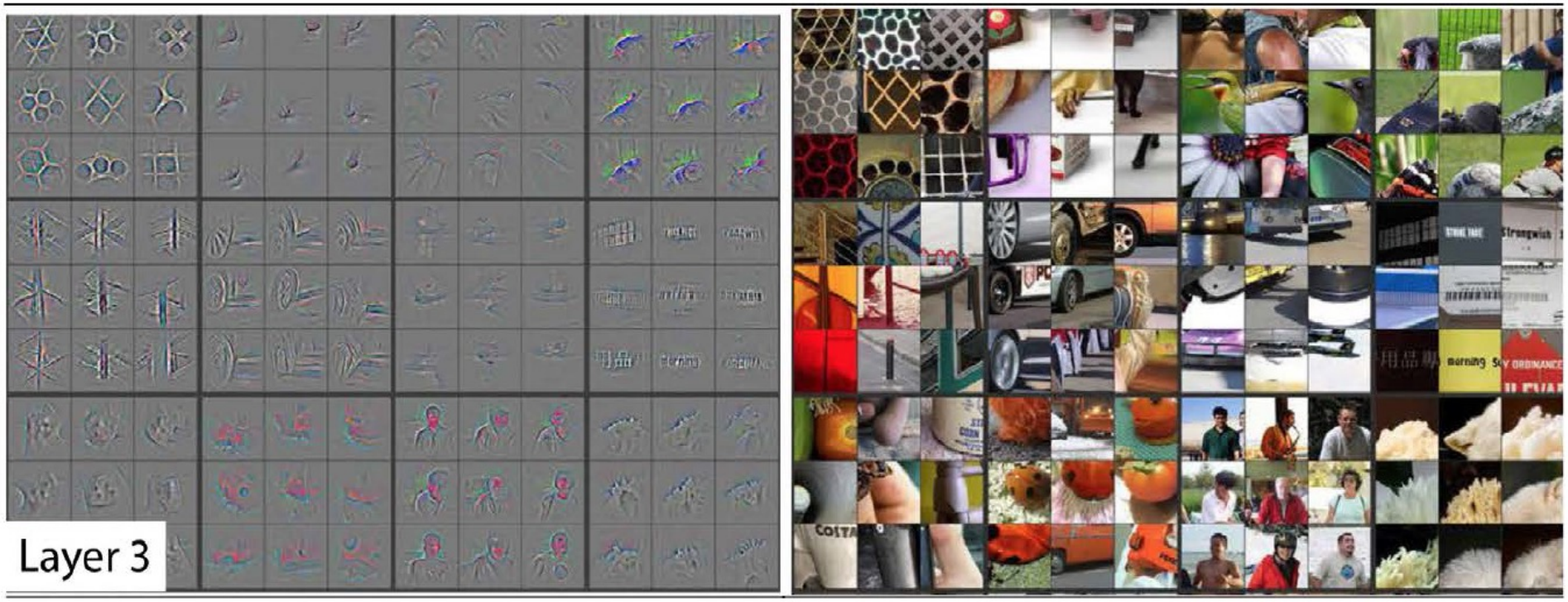
Convolution - Intuition

Source : https://cs.nyu.edu/~fergus/tutorials/deep_learning_cvpr12/

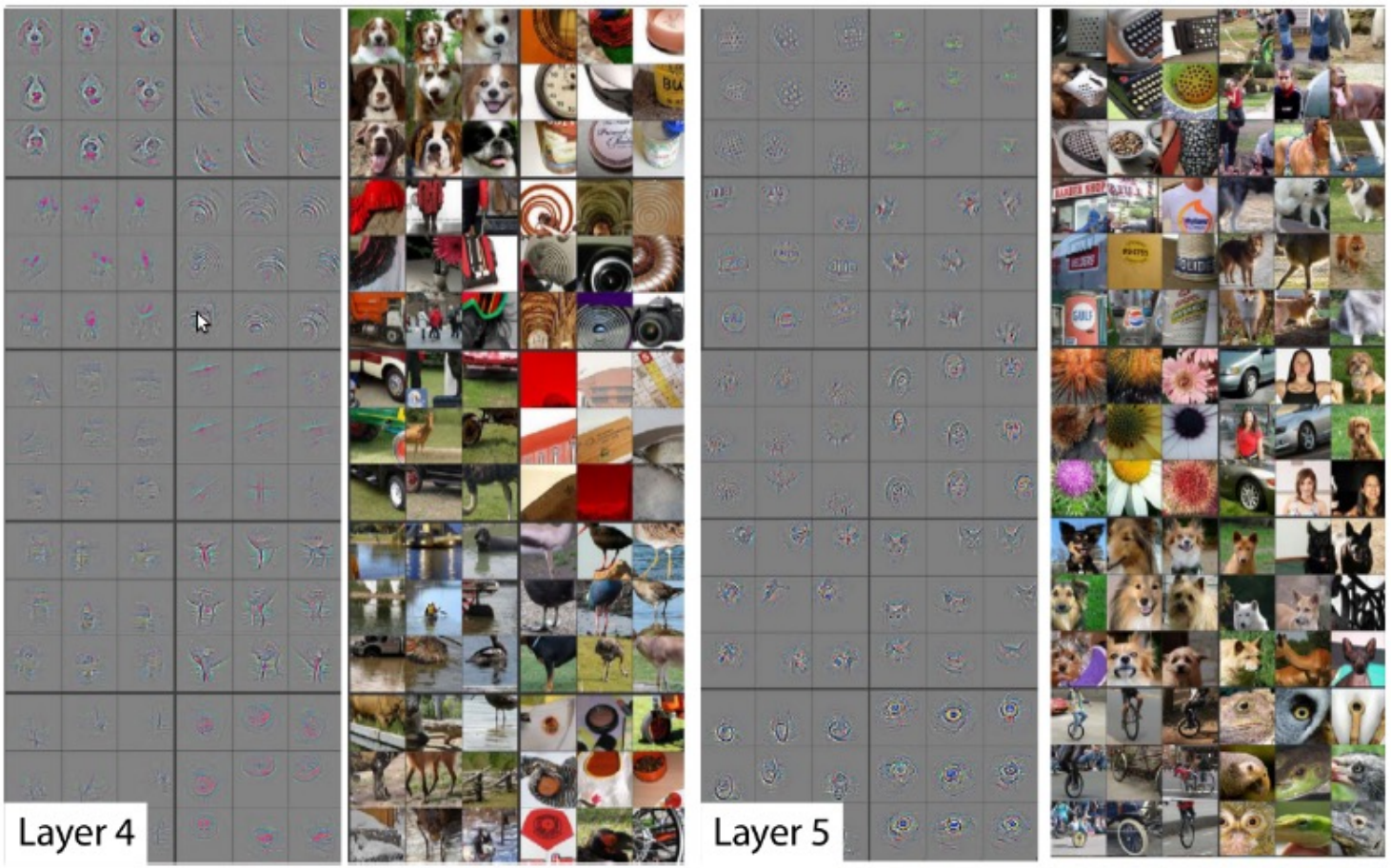
Visualizing CNN



Zeiler, Matthew D., and Rob Fergus. "Visualizing and understanding convolutional networks." *European conference on computer vision*. Springer, Cham, 2014.

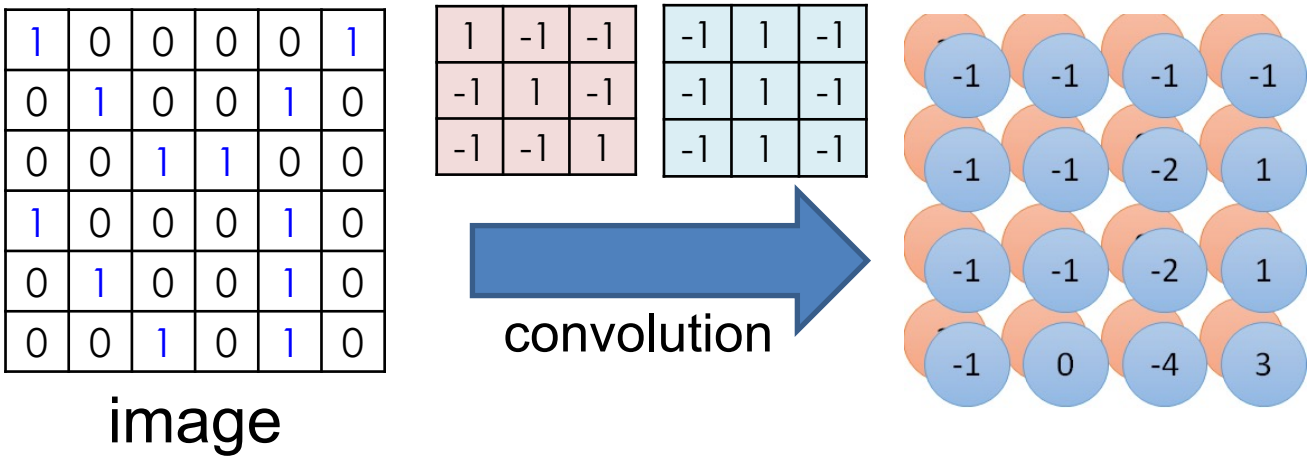


Zeiler, Matthew D., and Rob Fergus. "Visualizing and understanding convolutional networks." *European conference on computer vision*. Springer, Cham, 2014.

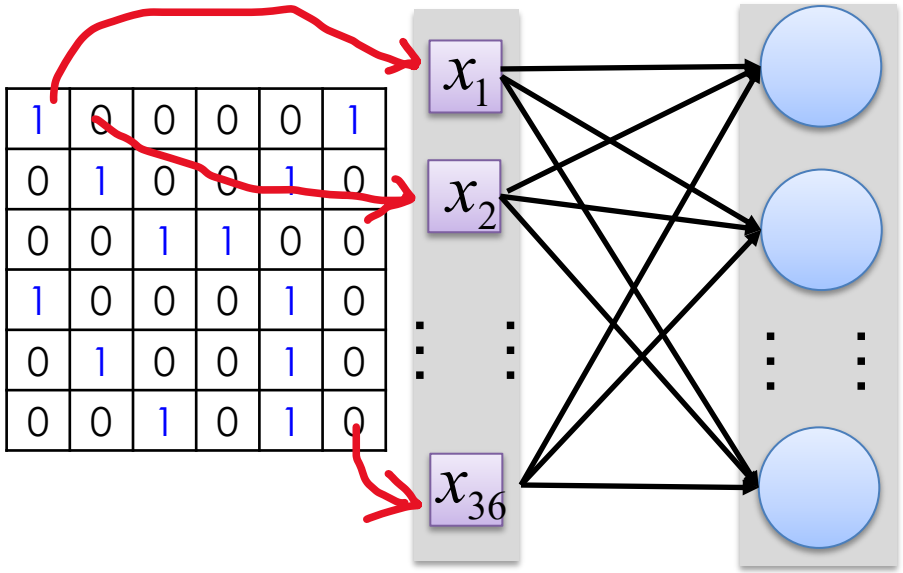


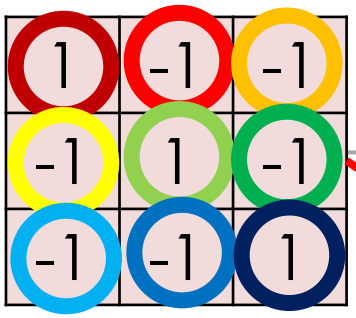
Zeiler, Matthew D., and Rob Fergus. "Visualizing and understanding convolutional networks." *European conference on computer vision*. Springer, Cham, 2014.

Convolution v.s. Fully Connected

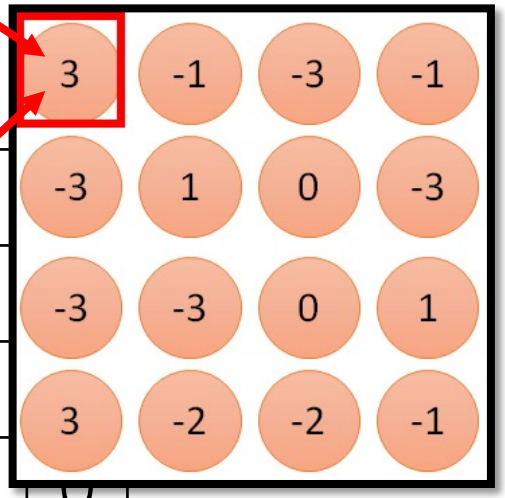


Fully-connected





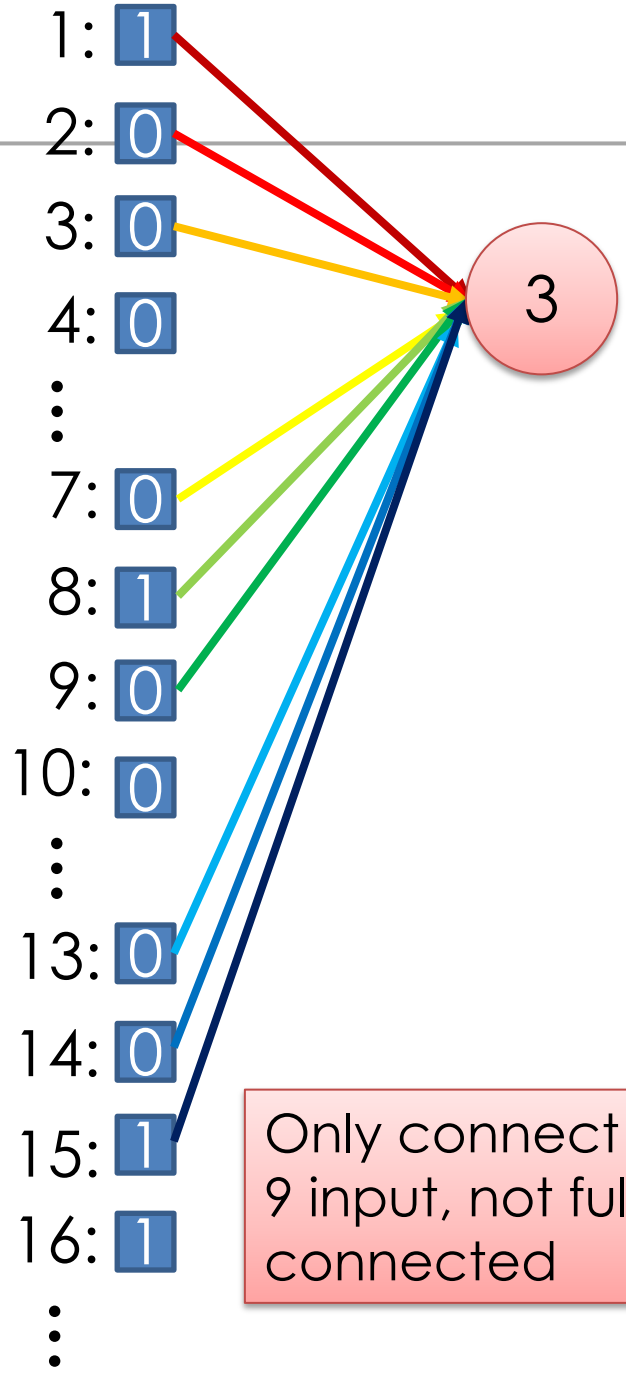
Filter 1



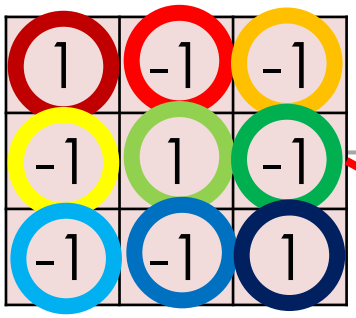
1	0	0	0	0	
0	1	0	0	1	
0	0	1	1	0	
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

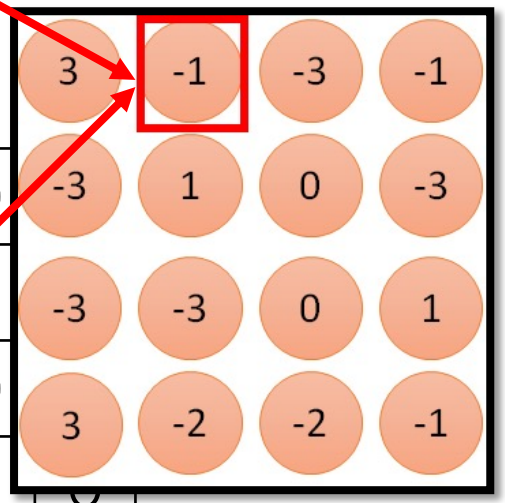
Less parameters!



Only connect to 9 input, not fully connected



Filter 1

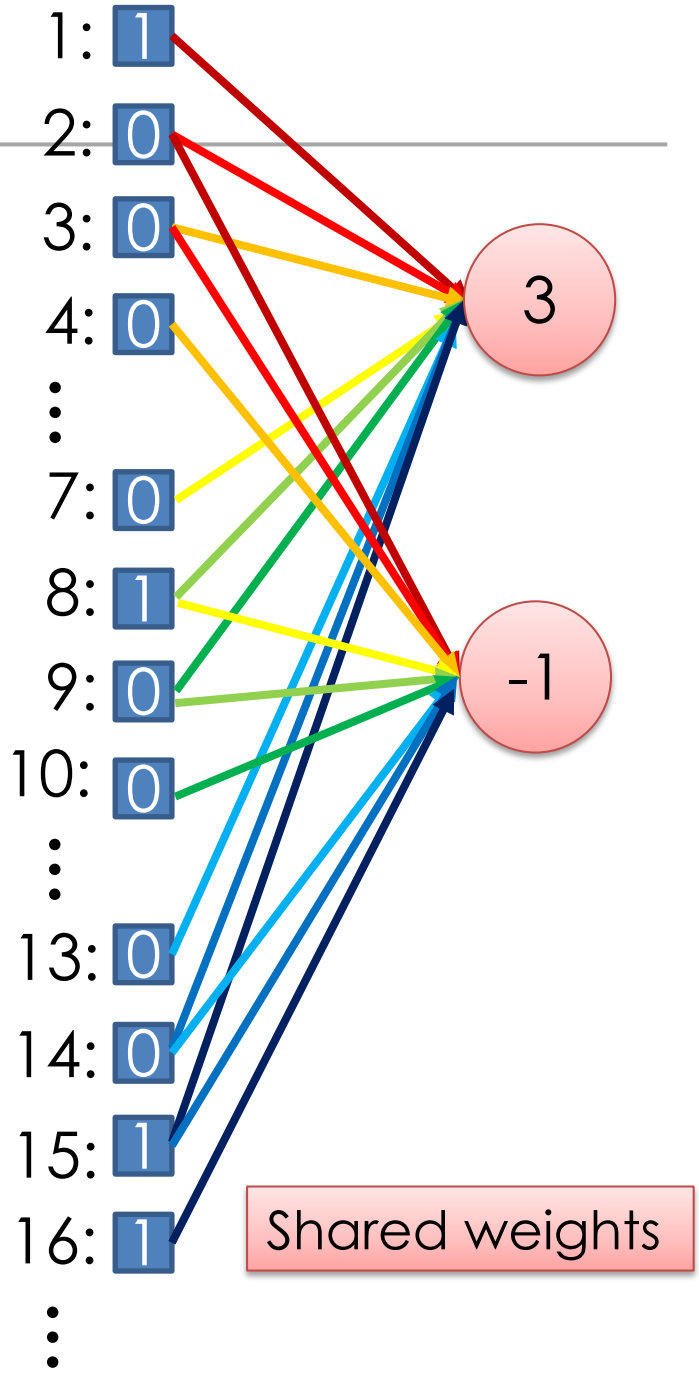


1	0	0	0	0	
0	1	0	0	1	
0	0	1	1	0	
1	0	0	0	1	
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

Less parameters!

Even less parameters!



Shared weights

Thank you!

Question?

References and Slide Credits

- Many slides are adapted from the existing teaching or tutorial slides by Hung-yi Lee, Andrew Ng, Alexander Amini, Lex Fridman, and Stanford course - CS231n: Convolutional Neural Networks for Visual Recognition
- Special thanks to Dr. Hung-yi Lee for making his machine learning course slides and materials available
- Alexander Amini, MIT 6.S191 Introduction to Deep Learning: http://introtodeeplearning.com/
  - Youtube videos: https://www.youtube.com/watch?v=5tvmMX8r_OM&list=PLtBw6njQRU-rwp5__7C0oIVt26ZgjG9NI&index=1
- Lex Fridman, MIT Deep Learning and Artificial Intelligence Lectures: https://deeplearning.mit.edu/ https://www.youtube.com/watch?v=O5xeyoRL95U