Digital Image Processing ECE 566

Ahmad Ghasemi

Department of Electrical and Computer Engineering

University of Massachusetts Amherst

Segmentation Evaluation

Can be considered to consist of two components:

(1) Theoretical

Study mathematical equivalence among algorithms.

(2) Empirical

Study practical performance of algorithms in specific application domains.

Segmentation Evaluation: Theoretical

Attributes commonly used by segmentation methods:

(1) Connectedness
(2) Texture
(3) Smoothness of boundary
(4) Gradient / homogeneity
(5) Shape information about object
(6) Noise handling
(7) Optimization employed
(8) Orientedness of boundary

Segmentation Evaluation: Theoretical

	Connected	Gradient	Texture	Smooth	Shape	Noise	Optimize
Chan-Vese	No	No	Yes	Yes	No	No	Yes
Mum-Shah	No	No	Yes	Yes	No	Yes	Yes
Live wire	Boundary	Yes	Yes	Yes	User	No	Yes
Act. shape	Yes	No	No	No	Yes	No	Yes
Act. app	Yes	No	Yes	No	Yes	No	Yes
Graph cut	Usly not	Yes	Possible	No	No	No	Yes
Clustering	No	No	Yes	No	No	No	Yes

Segmentation Evaluation: Theoretical

	Connected	Gradient	Texture	Smooth	Shape	Noise	Optimize
Chan-Vese	No	No	Yes	Yes	No	No	Yes
Mum-Shah	No	No	Yes	Yes	No	Yes	Yes
Live wire	Boundary	Yes	Yes	Yes	User	No	Yes
Act. shape	Yes	No	No	No	Yes	No	Yes
Act. app	Yes	No	Yes	No	Yes	No	Yes
Graph cut	Usly not	Yes	Possible	No	No	No	Yes
Clustering	No	No	Yes	No	No	No	Yes
Deep Learning	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Segmentation Evaluation: Empirical

Precision (Reliability)	:	Repeatability taking into account all subjective actions influencing the result.
Accuracy (Validity)	:	Degree to which the result agrees with truth.
Efficiency (Viability)	:	Practical viability of the method.

University of Massachusetts Amherst

Segmentation Evaluation: Empirical

Precision (Reliability)

A measure of precision for method M in a trial that produces $C_M^{O_1}$ and $C_M^{O_2}$

$$PR_M^{T_i} = \frac{|C_M^{O_1} \cap C_M^{O_2}|}{|C_M^{O_1} \cup C_M^{O_2}|},$$

Segmentation Evaluation: Empirical

Accuracy (Validity)

The degree to which segmentations agree with true segmentation.

Surrogates of truth are needed.

For any image C acquired for application domain

C_M^O - segmentation of O in C by method M,

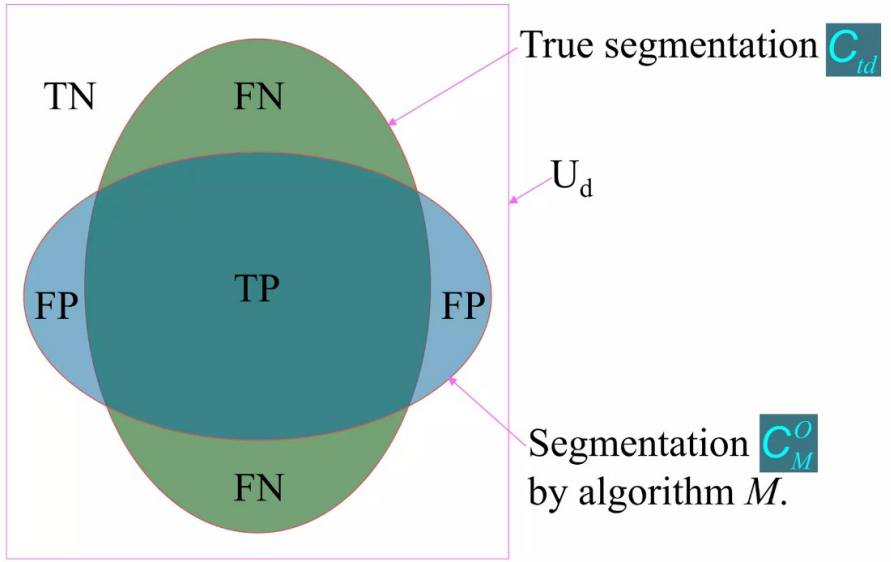
C_{td} - surrogate of true delineation of O in C.

Segmentation Evaluation: Empirical

Accuracy (Validity)

$$FNVF_M^d = \frac{\left|C_{td} - C_M^O\right|}{\left|C_{td}\right|}, \qquad TPVF_M^d = \frac{\left|C_{td} \cap C_M^O\right|}{\left|C_{td}\right|}$$

$$FPVF_M^d = \frac{\left|C_M^O - C_{td}\right|}{\left|U_d - C_{td}\right|}, \qquad TNVF_M^d = \frac{\left|U_d - C_M^O - C_{td}\right|}{\left|U_d - C_{td}\right|},$$



U_d : A binary scene representing a reference super set (for example, this may be the body region that is imaged).

$FNVF_M^d$: Amount of tissue truly in O that is missed by M.

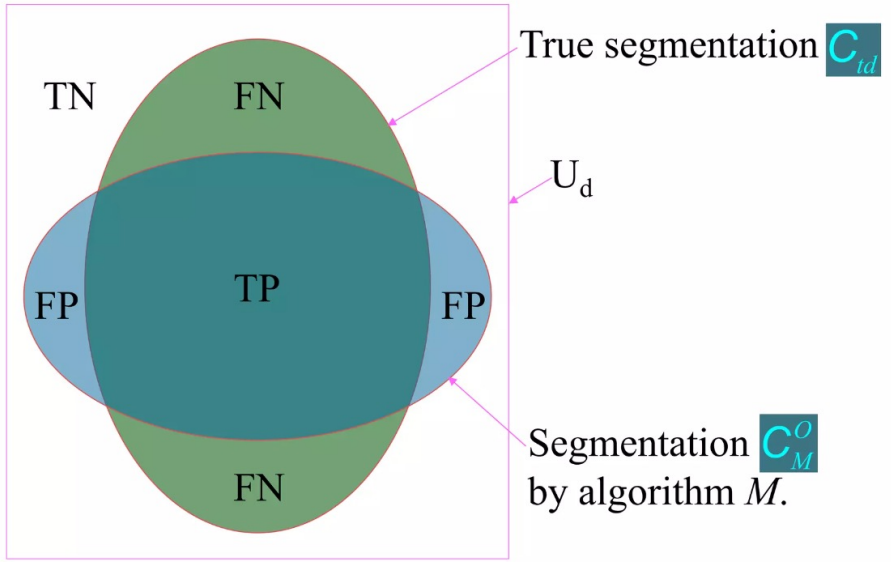
$FPVF_M^d$: Amount of tissue falsely delineated by M.

Segmentation Evaluation: Empirical

Accuracy (Validity)

$$FNVF_M^d = \frac{\left|C_{td} - C_M^O\right|}{\left|C_{td}\right|}, \qquad TPVF_M^d = \frac{\left|C_{td} \cap C_M^O\right|}{\left|C_{td}\right|}$$

$$FPVF_M^d = \frac{\left|C_M^O - C_{td}\right|}{\left|U_d - C_{td}\right|}, \qquad TNVF_M^d = \frac{\left|U_d - C_M^O - C_{td}\right|}{\left|U_d - C_{td}\right|},$$



U_d : A binary scene representing a reference super set (for example, this may be the body region that is imaged).

$FNVF_M^d$: Amount of tissue truly in O that is missed by M.

$FPVF_M^d$: Amount of tissue falsely delineated by M.

$$FNVF_M^d = 1 - TPVF_M^d$$

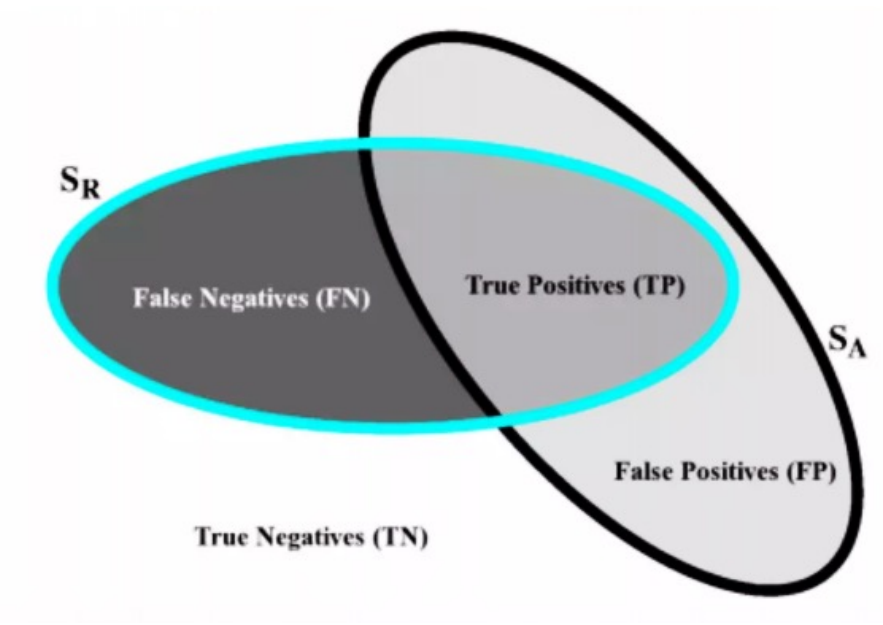
$$FPVF_M^d = 1 - TNVF_M^d$$

Segmentation Evaluation: Empirical

Overlap Measures

- Sensitivity: rate of detection of the structure

$$\text{Sens} = \frac{\text{TP}}{\text{TP}+\text{FN}}$$



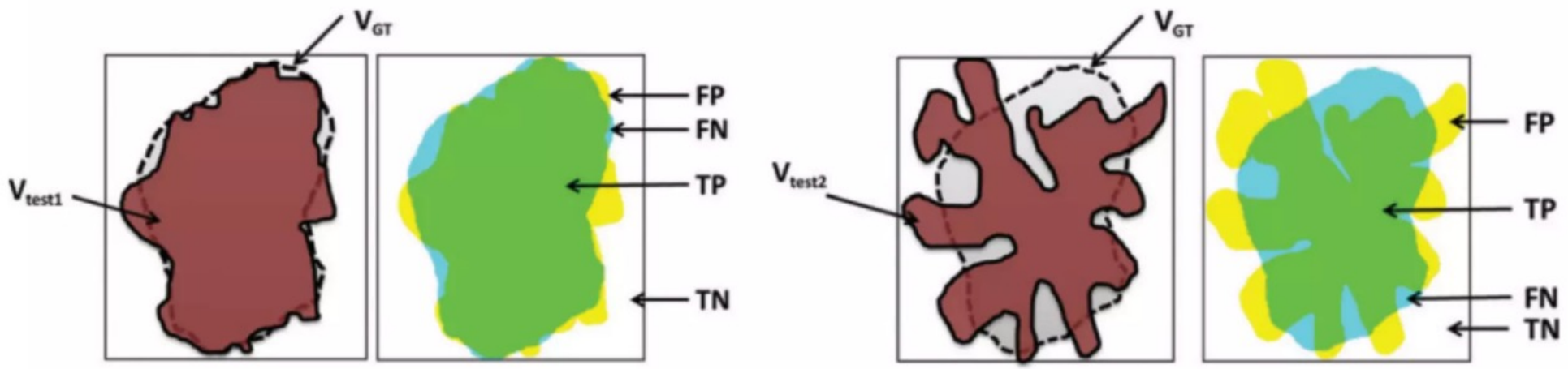
- Specificity: rate of detection of the structure

$$\text{Spec} = \frac{\text{TN}}{\text{TN}+\text{FP}}$$

University of Massachusetts Amherst

Segmentation Evaluation: Empirical

Overlap Measures: example



Sensitivity=94.69%
Specificity=94.19%

Sensitivity=72.99%
Specificity=78.16%

University of Massachusetts Amherst

Segmentation Evaluation: Empirical

Efficiency (Viability)

Describes practical viability of a method.

Four factors should be considered:

(1) Computational time – for one time training of M $\left(t_M^{c_1}\right)$

(2) Computational time – for segmenting each scene $\left(t_M^{c_2}\right)$

(3) Human time – for one-time training of M $\left(t_M^{h_1}\right)$

(4) Human time – for segmenting each scene $\left(t_M^{h_2}\right)$

(2) and (4) are crucial. (4) determines the degree of automation of M.

Segmentation Evaluation: Empirical

Remark

Precision, accuracy, efficiency are interdependent.

↑accuracy → ↓efficiency.

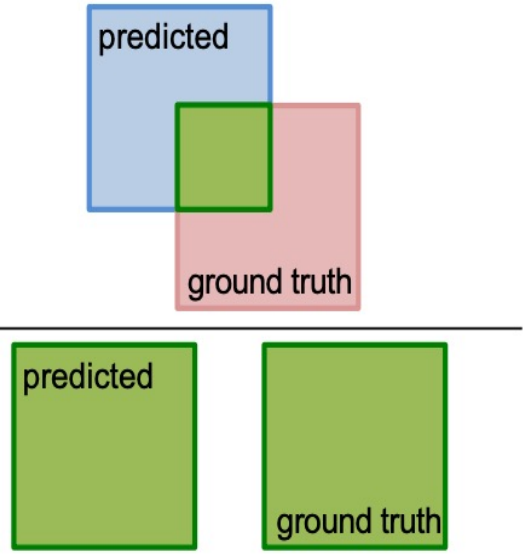
↑precision and ↑accuracy → difficult.

Segmentation Evaluation: Empirical

Dice index

$$Dice(A,B) = 2\frac{|A \cap B|}{|A| + |B|}$$

$$\text{Dice Coefficient} = \frac{\text{2×area of overlapped (green)}}{\text{total area (green)}} =$$



Jaccard index(IoU)

$$Jaccard(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Image Segmentation

Traditional Methods

- Threshold-based
- Edge-based
- Region-based
- Energy-based
- Graph-based
- …

University of Massachusetts Amherst

Deep Learning based Image Segmentation

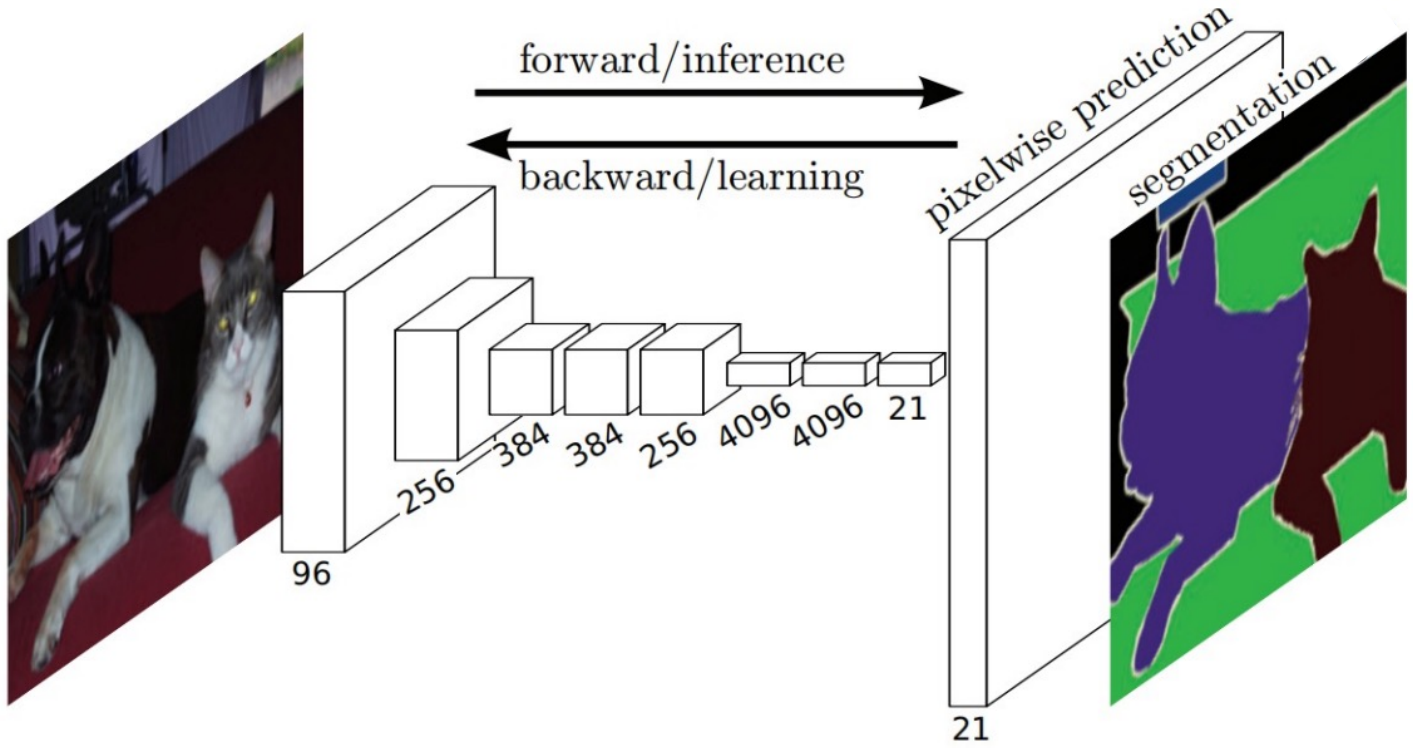
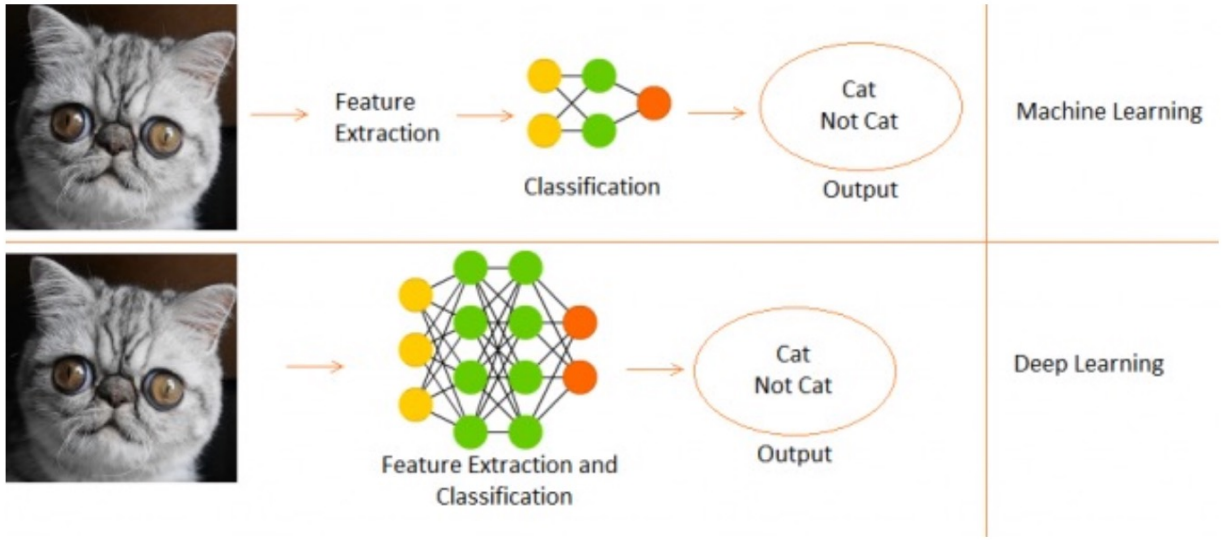


Figure 1. Fully convolutional networks can efficiently learn to make dense predictions for per-pixel tasks like semantic segmentation.

Long J, Shelhamer E, Darrell T. Fully convolutional networks for semantic segmentation[C]// Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 3431-3440.

Why Deep Learning?



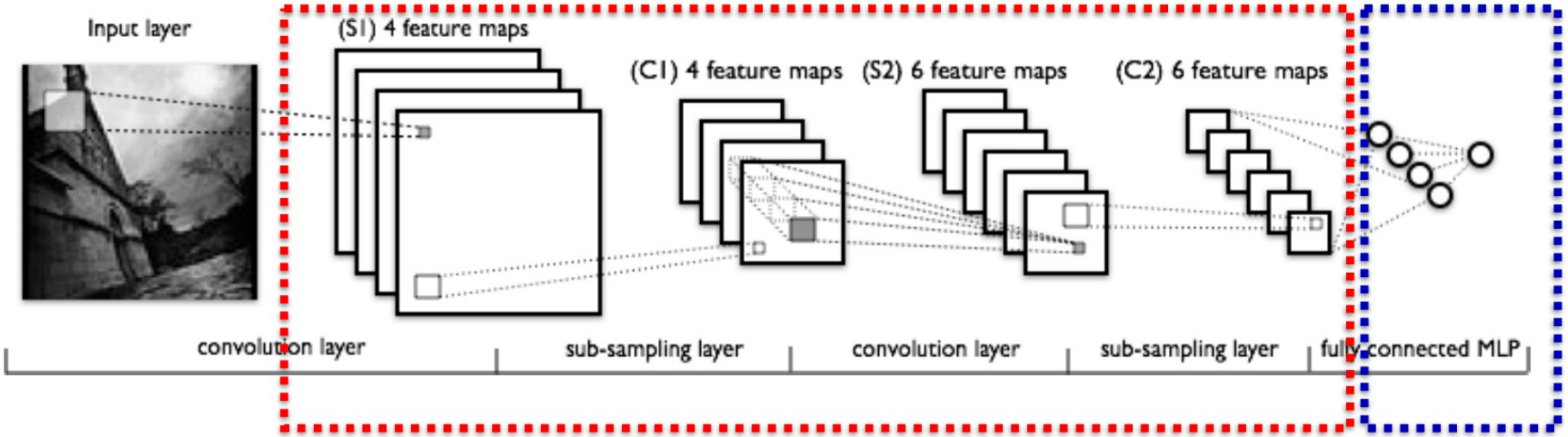
Hand engineered features are time consuming, brittle and not scalable in practice

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

Credit: Alexander Amini, MIT

University of Massachusetts Amherst

Deep Convolutional Neural Networks (DCNN)



Feature extraction using convolutional layers

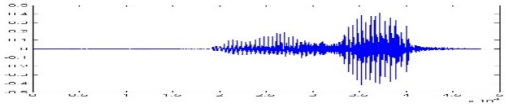
Classification using multilayer perceptron (MLP)

Credit: Prof. Sinisa Todorovic

University of Massachusetts Amherst

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

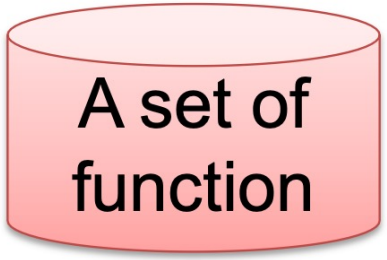
$f($ “Hi” $)=$ “Hello”

(what the user said) (system response)

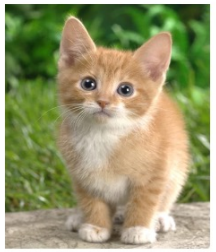
Credit: Hung-yi Lee, Andrew, Stanford course

Framework

Image Recognition



$f($  $)=$ "Cat"

$f_1($  $)=$ "cat"

$f_2($  $)=$ "money"

$f_1($  $)=$ "dog"

$f_2($  $)=$ "snake"

Credit: 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"

Better!

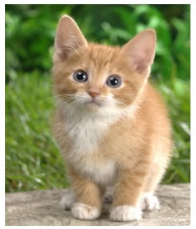
$f_1($  $)=$ "dog"

$f_2($  $)=$ "snake"

Goodness of function f

Training Data

Supervised Learning

function input:   

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

Credit: Hung-yi Lee, Andrew, Stanford course

Framework

Image Recognition

$f($  $) =$ "Cat"

Training

Testing

Step 1: A set of function; Model $f_1, f_2 \cdots$

Step 2: Goodness of function f

Step 3: Pick the "Best" Function f^*

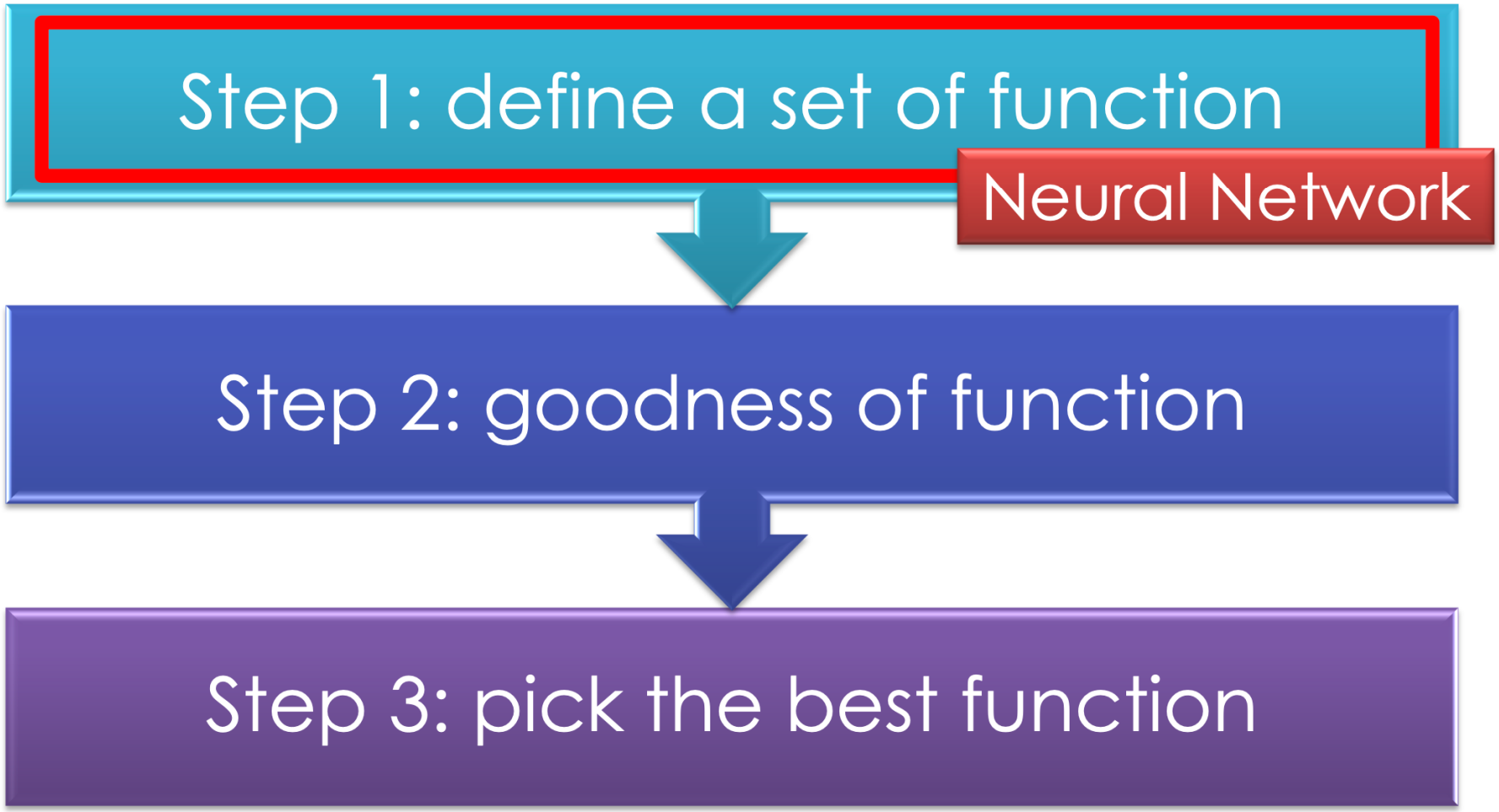
Training Data:  "monkey"  "cat"  "dog"

Using f^*: "cat"

Credit: Hung-yi Lee, Andrew, Stanford course

University of Massachusetts Amherst

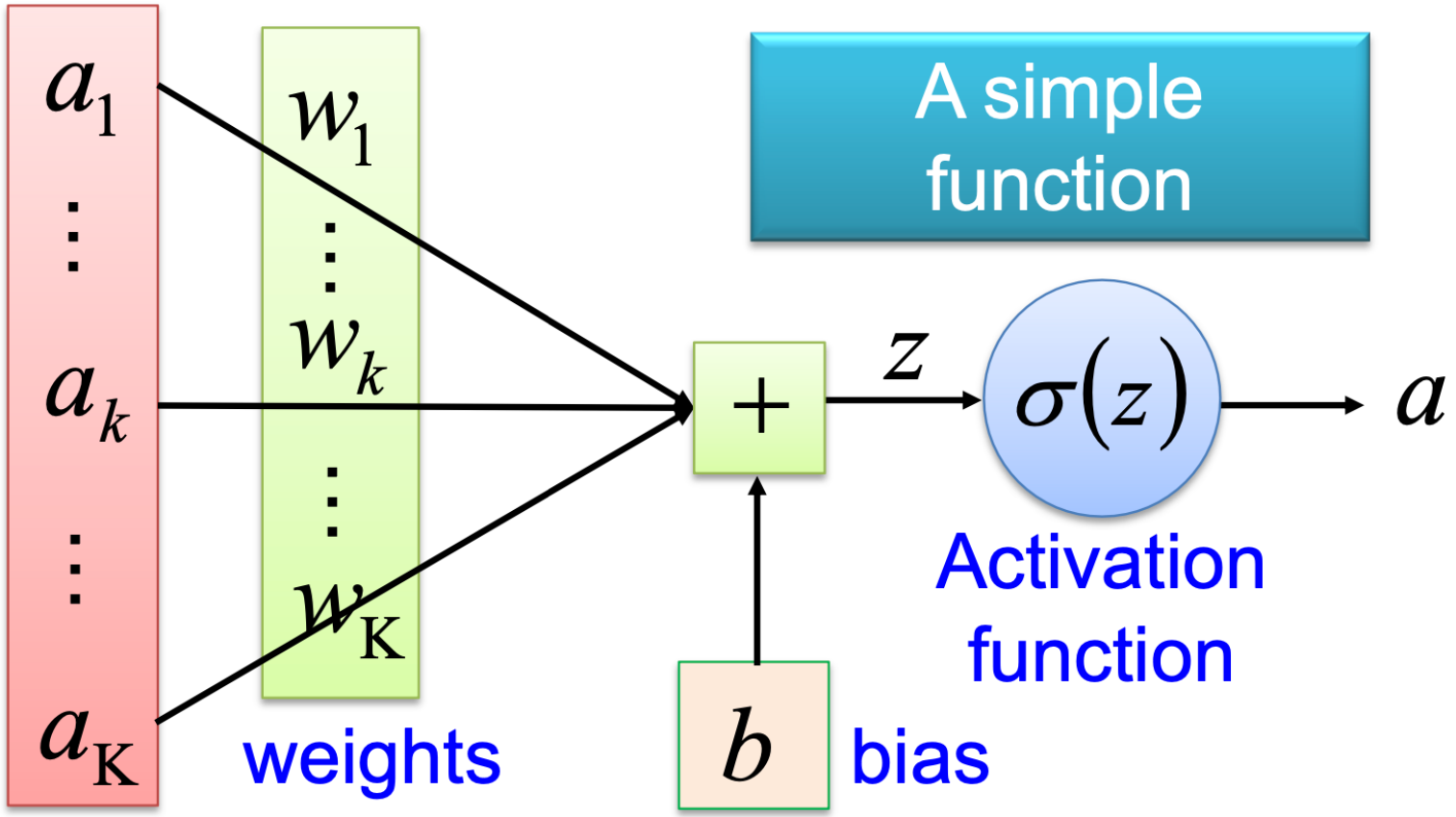
Three steps for Deep Learning



Credit: Hung-yi Lee, Andrew, Stanford course

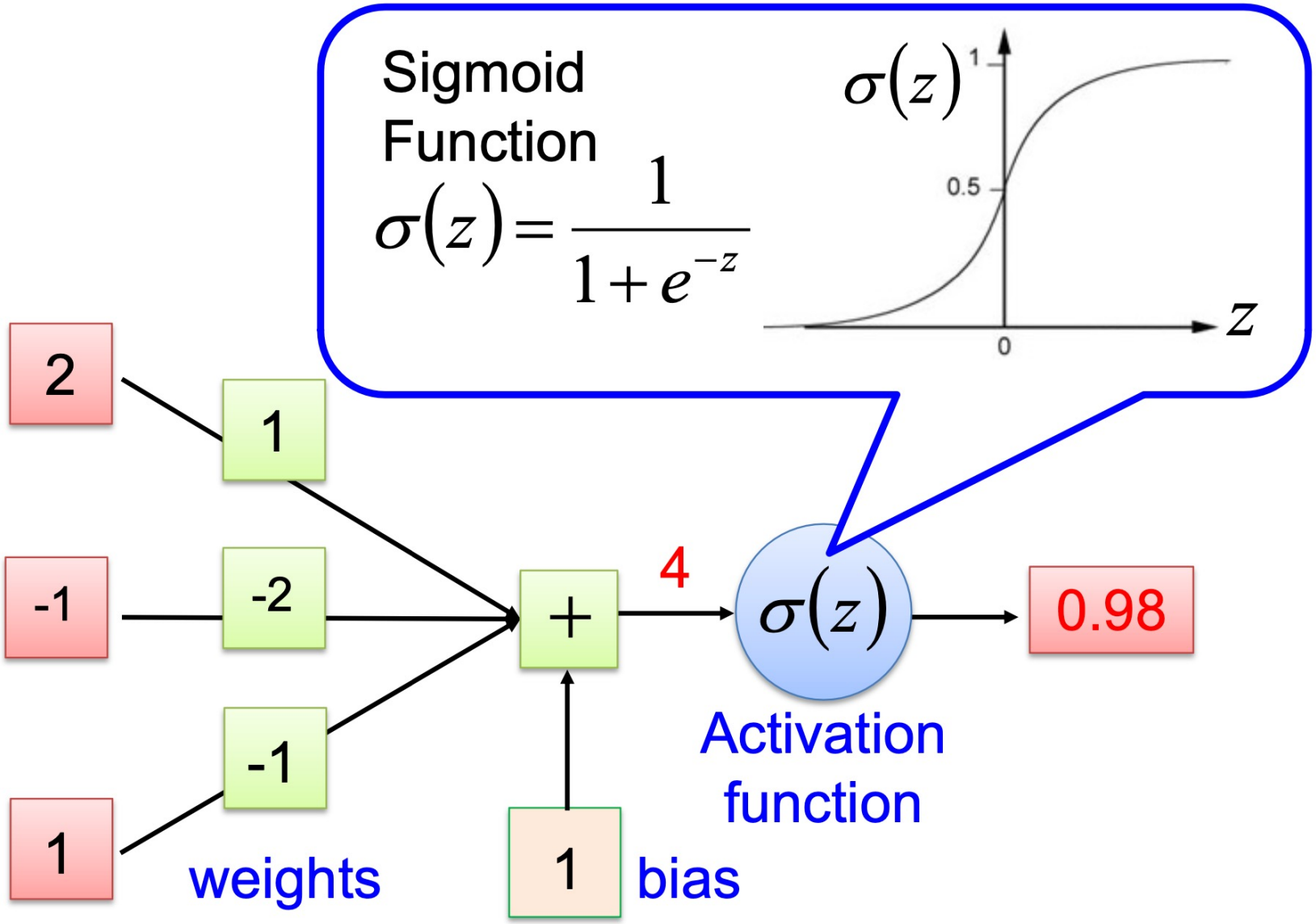
Neural Network: Neuron

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



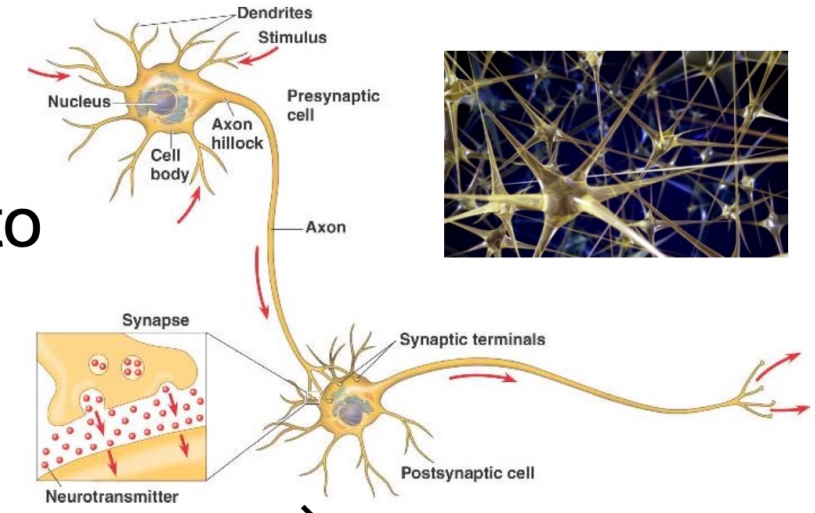
Credit: Hung-yi Lee, Andrew, Stanford course

Neural Network: Neuron

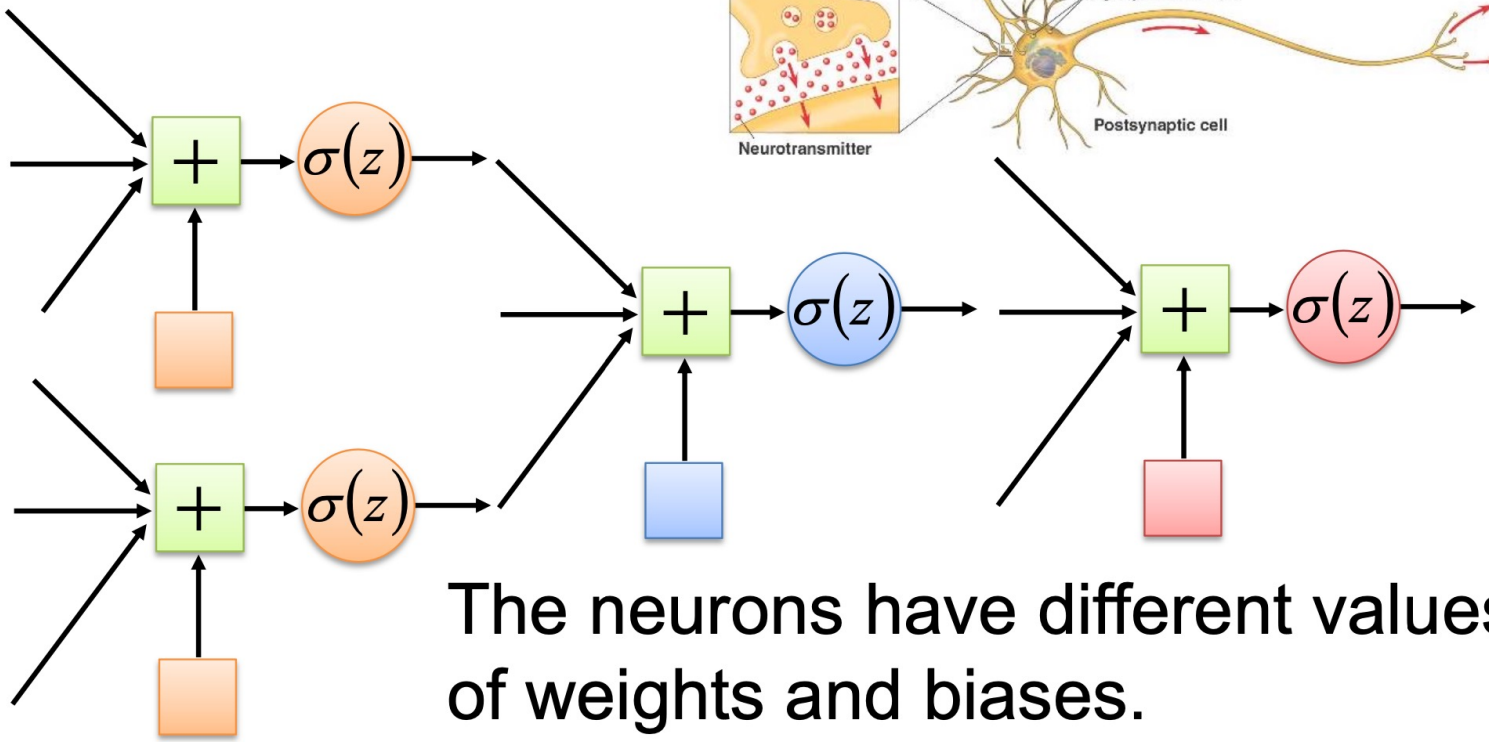


Credit: Hung-yi Lee, Andrew, Stanford course

Neural Network: Neuron



Different connections lead to different network structures

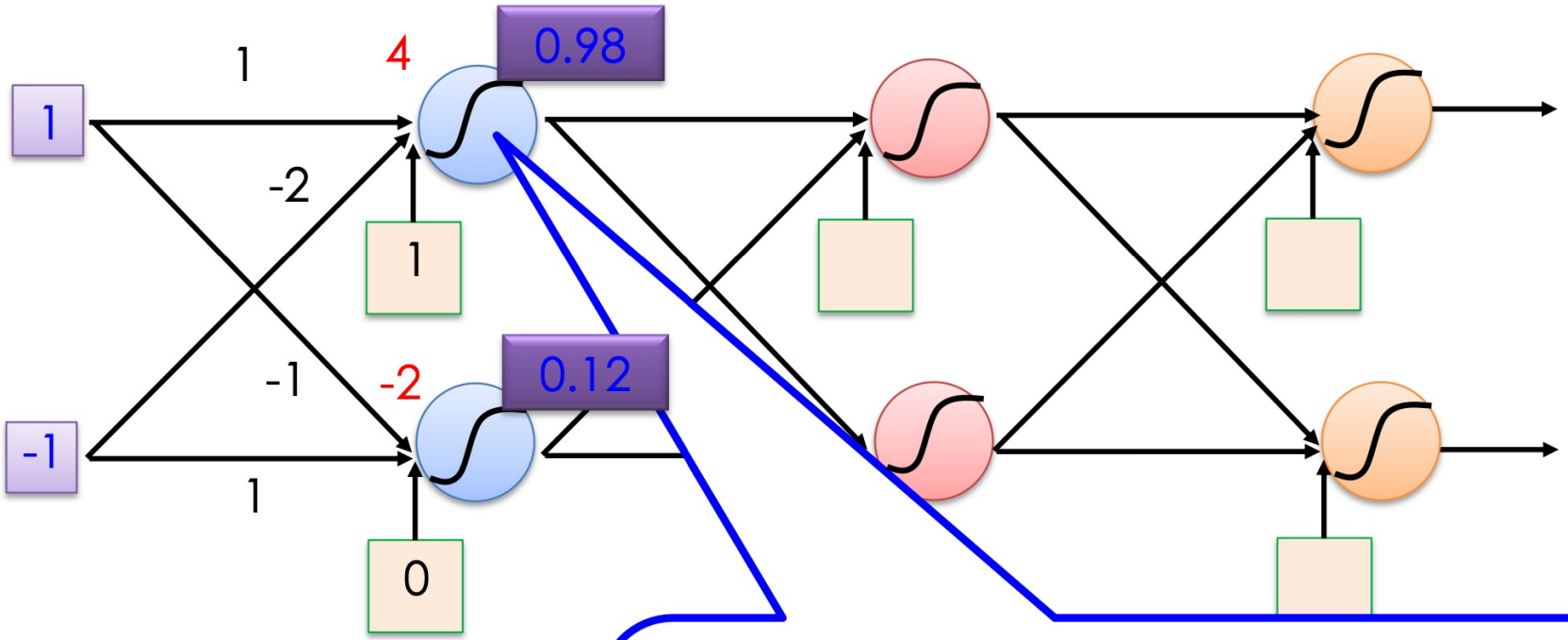


The neurons have different values of weights and biases.

Weights and biases are network parameters θ

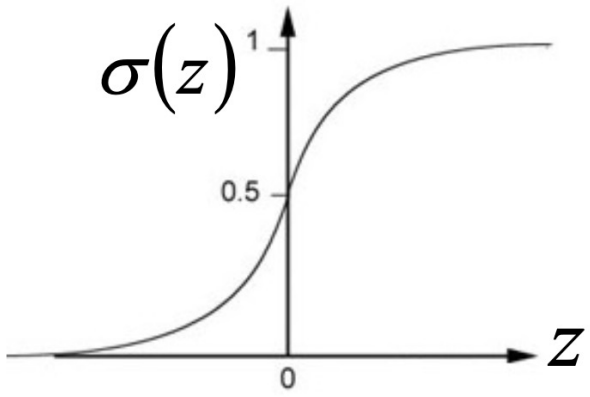
Credit: Hung-yi Lee, Andrew, Stanford course

Fully Connected Feedforward Network



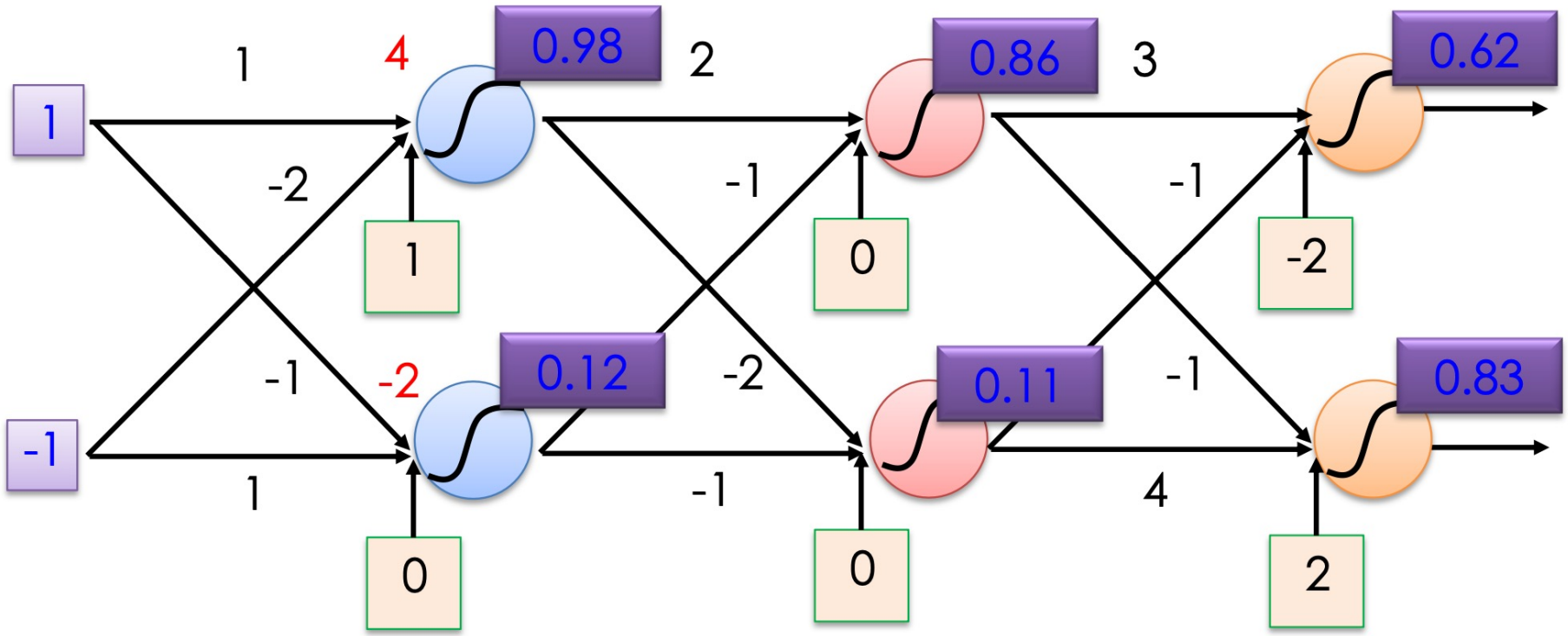
Sigmoid Function

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



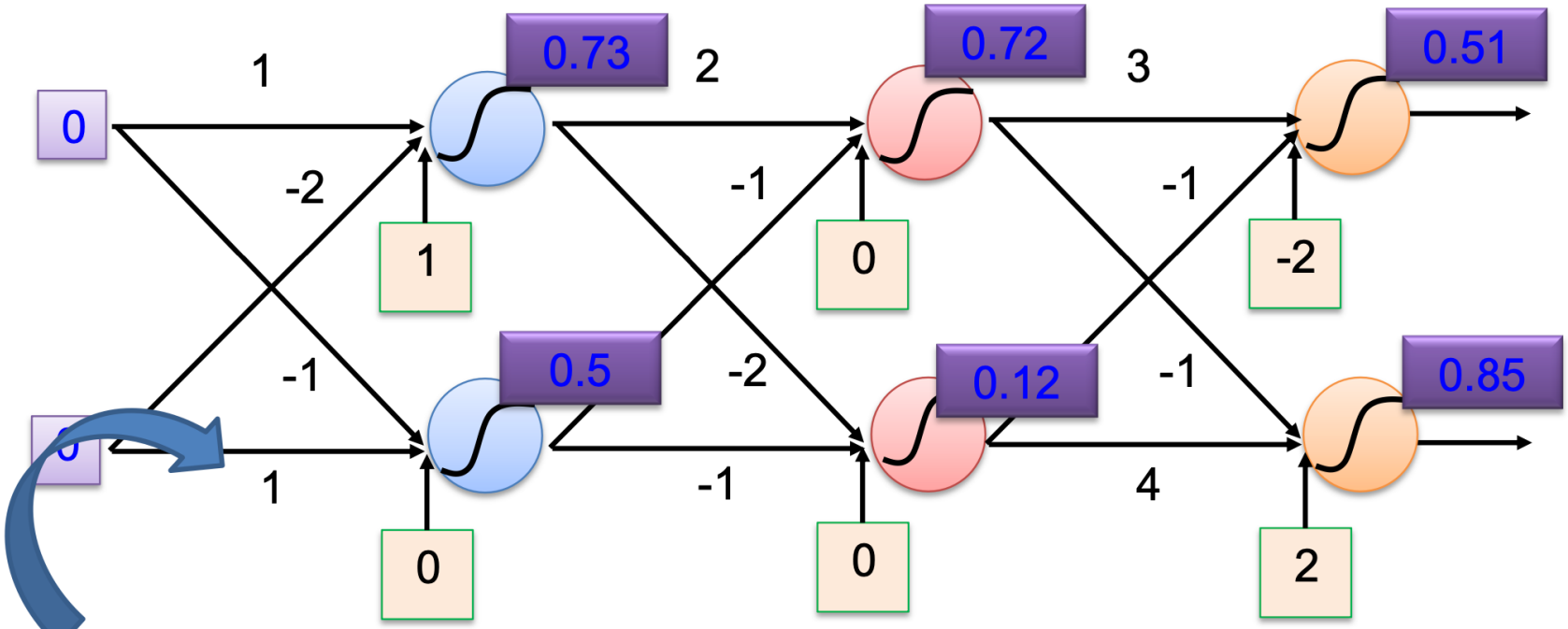
Credit: Hung-yi Lee, Andrew, Stanford course

Fully Connected Feedforward Network



Credit: Hung-yi Lee, Andrew, Stanford course

Fully Connected Feedforward Network



This is a function.

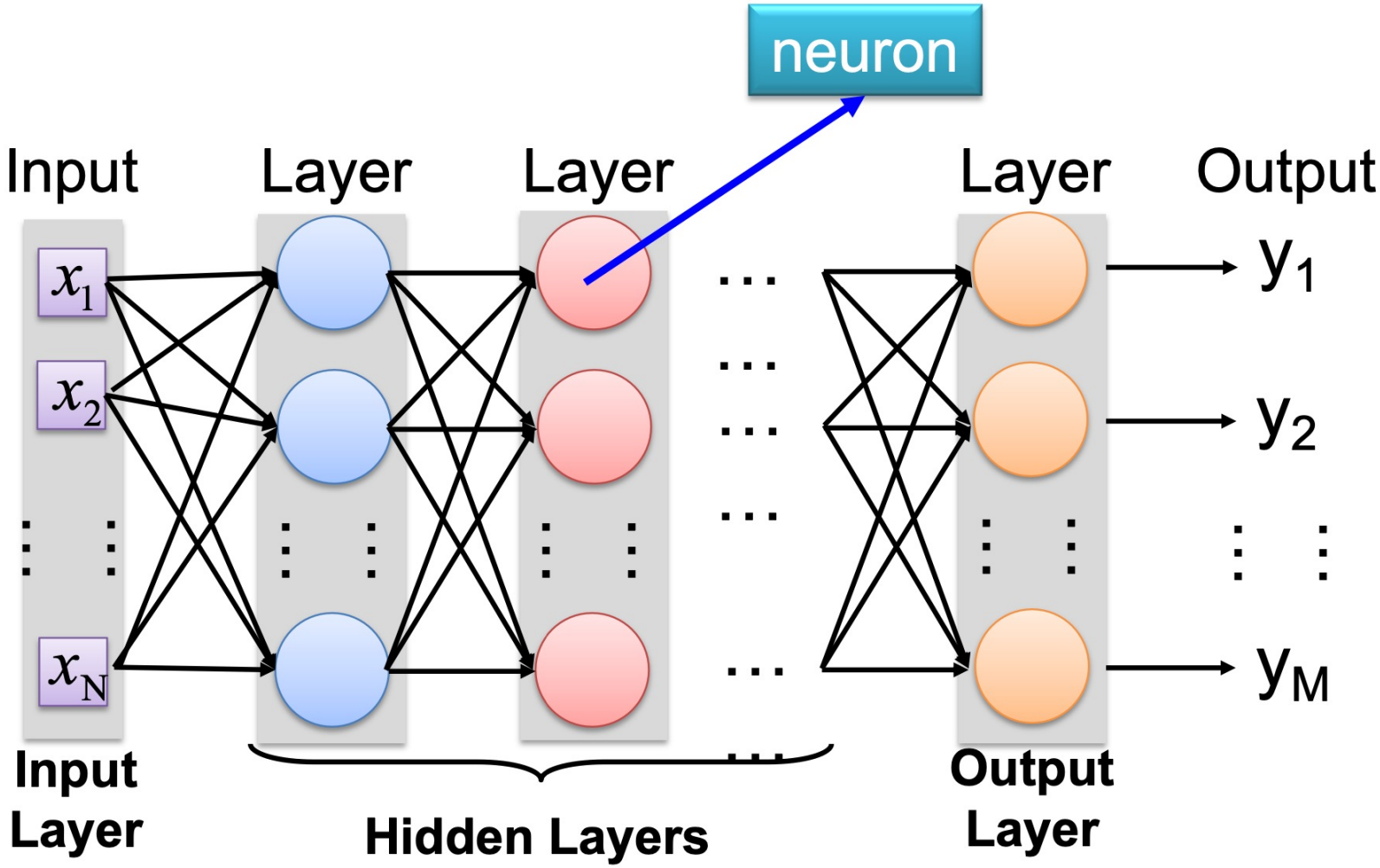
Input vector, output vector

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

Given parameters θ, define a function

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

Fully Connected Feedforward Network



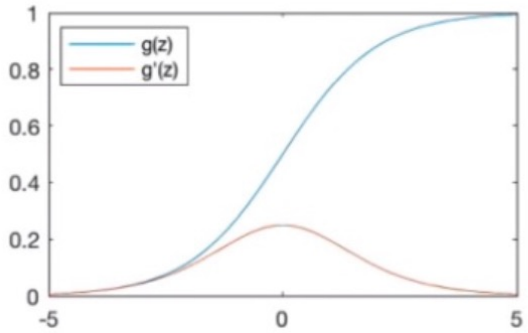
Deep means many hidden layers

Credit: Hung-yi Lee, Andrew, Stanford course

University of Massachusetts Amherst

Activation Function

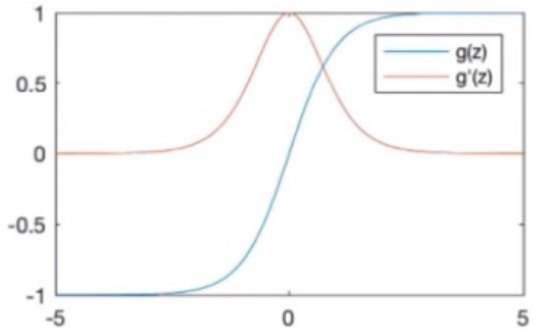
Sigmoid Function



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

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

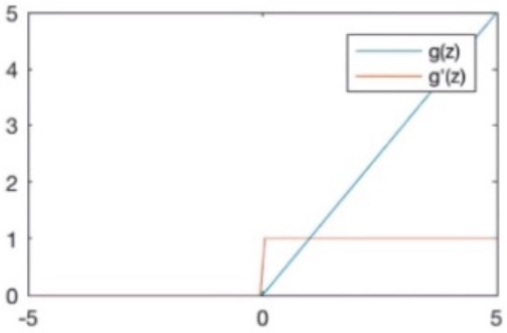
Hyperbolic Tangent



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

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

Rectified Linear Unit (ReLU)



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

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

NOTE: All activation functions are non-linear

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

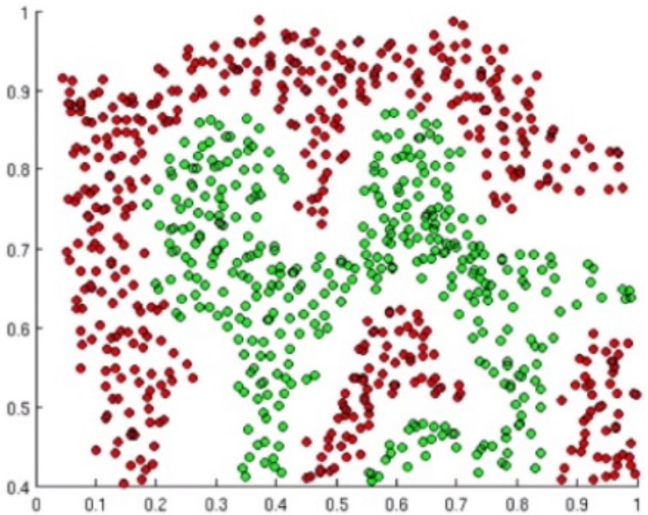
Activation Function

- 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*



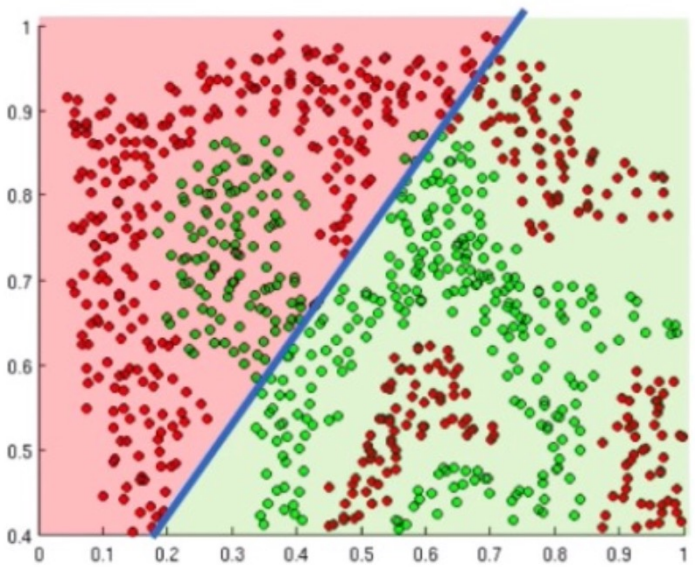
What if we wanted to build a Neural Network to distinguish green vs red points?

University of Massachusetts Amherst

Activation Function

- Why non-linear activation function?

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



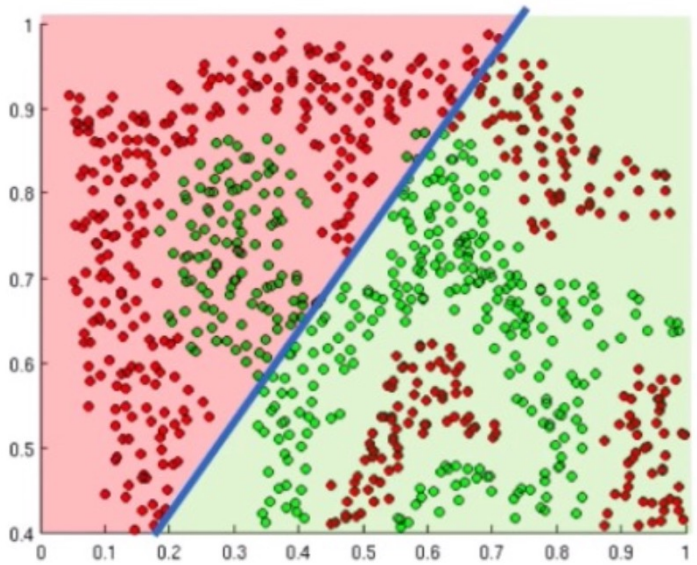
Linear Activation functions produce linear decisions no matter the network size

University of Massachusetts Amherst

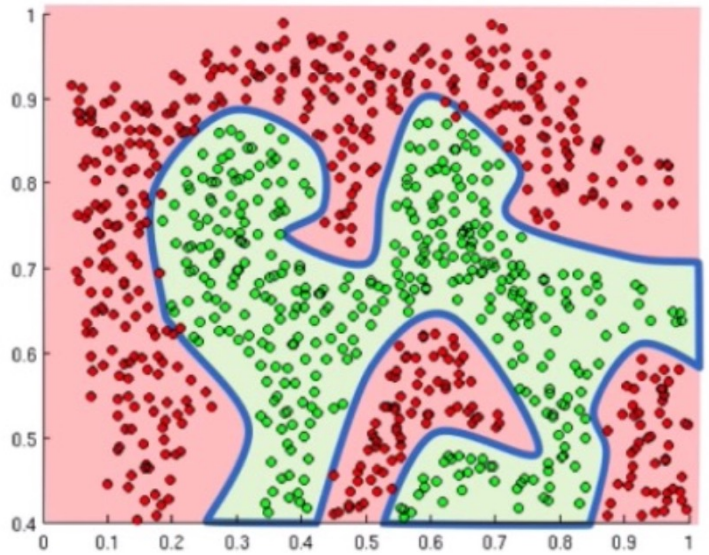
Activation Function

- 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



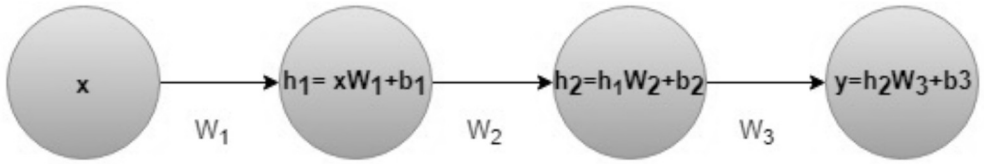
Non-linearities allow us to approximate arbitrarily complex functions

University of Massachusetts Amherst

Activation Function

- Why non-linear activation function?

- With linear activation functions, no matter how many layers in the NN, the last layer will be a linear function of the first layer (because a linear combination of linear functions is still a linear function) ➔ a linear activation function turns the NN into just one layer.

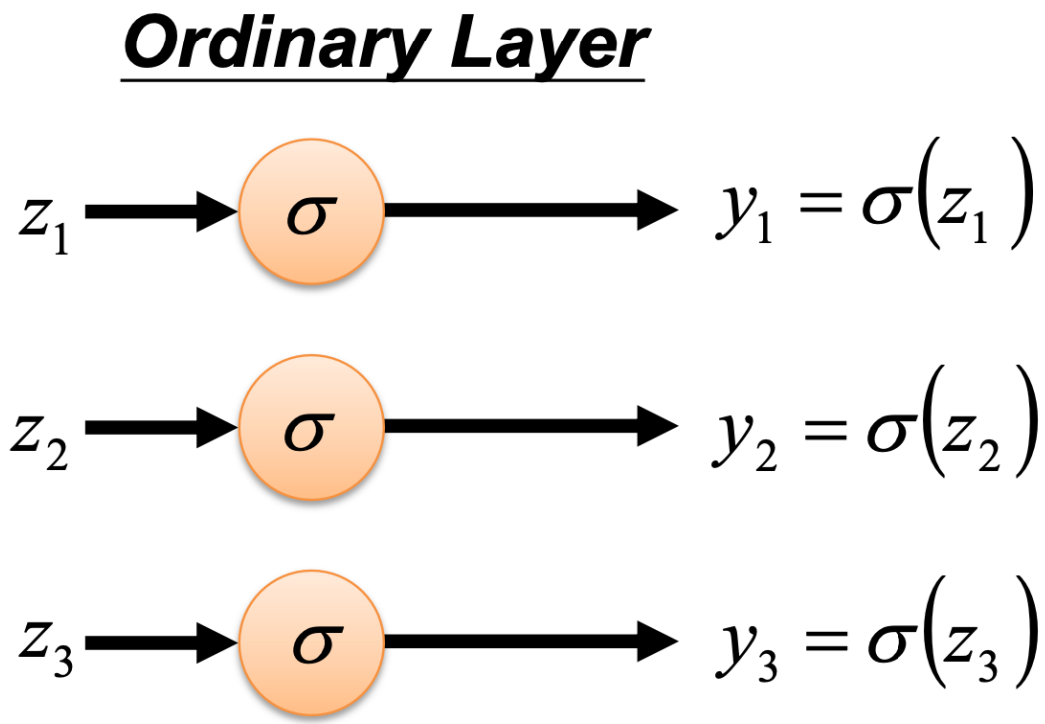



```
y = h2 * W3 + b3
  = (h1 * W2 + b2) * W3 + b3
  = h1 * W2 * W3 + b2 * W3 + b3
  = (x * W1 + b1) * W2 * W3 + b2 * W3 + b3
  = x * W1 * W2 * W3 + b1 * W2 * W3 + b2 * W3 + b3
  = x * W' + b'
```

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

- A NN 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.

Output Layer: Softmax Function



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

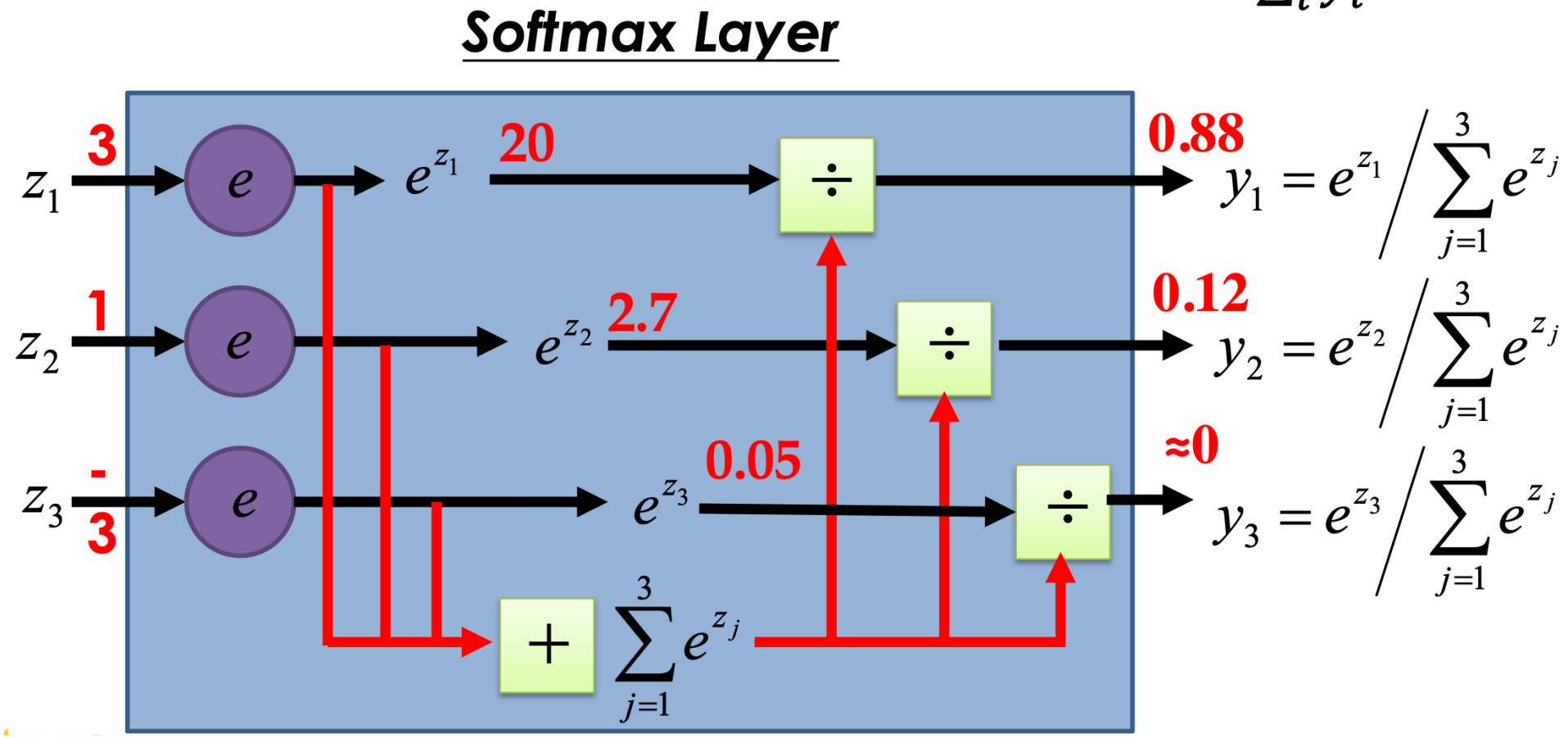
May not be easy to interpret

Credit: Hung-yi Lee, Andrew, Stanford course

Output Layer: Softmax Function

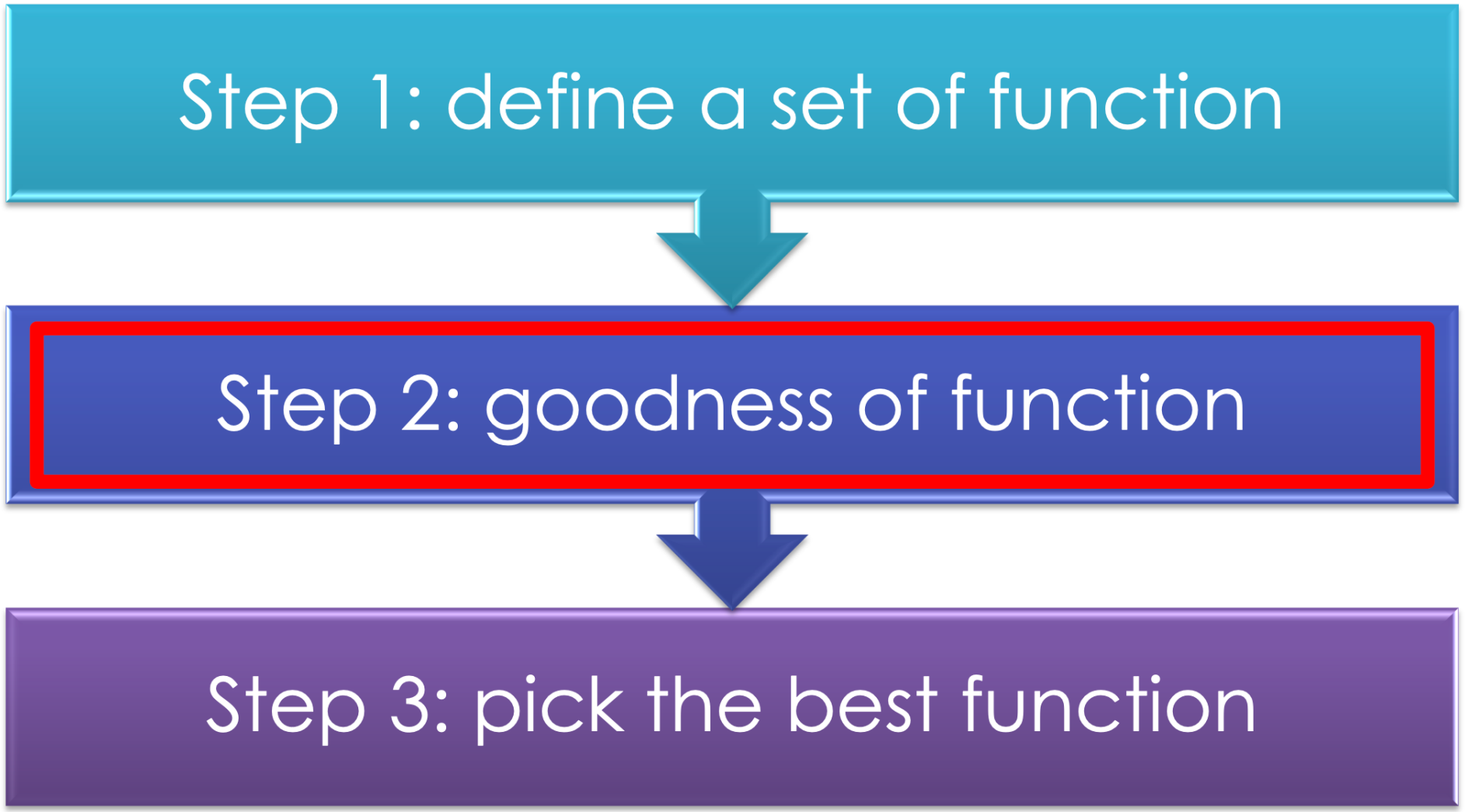
Probability:

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



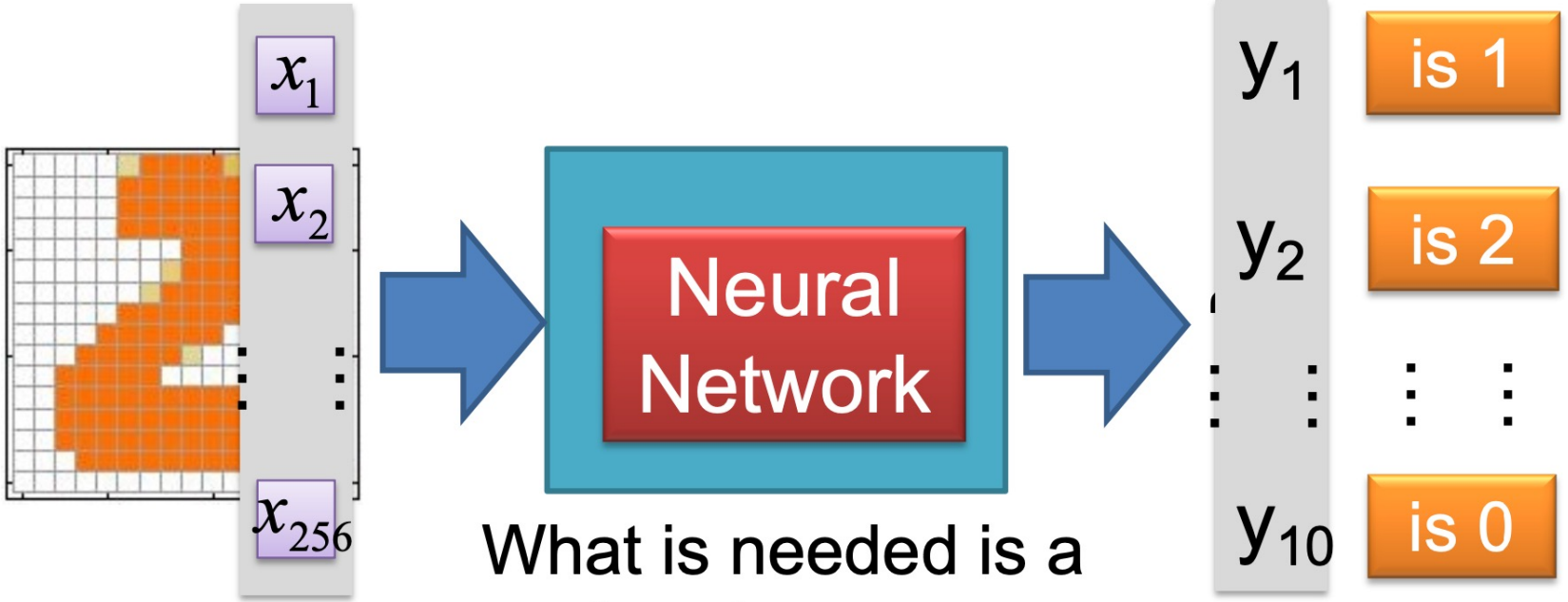
Credit: Hung-yi Lee, Andrew, Stanford course

Three steps for Deep Learning



Credit: Hung-yi Lee, Andrew, Stanford course

Application Example: Handwriting Digit Recognition



What is needed is a function ……

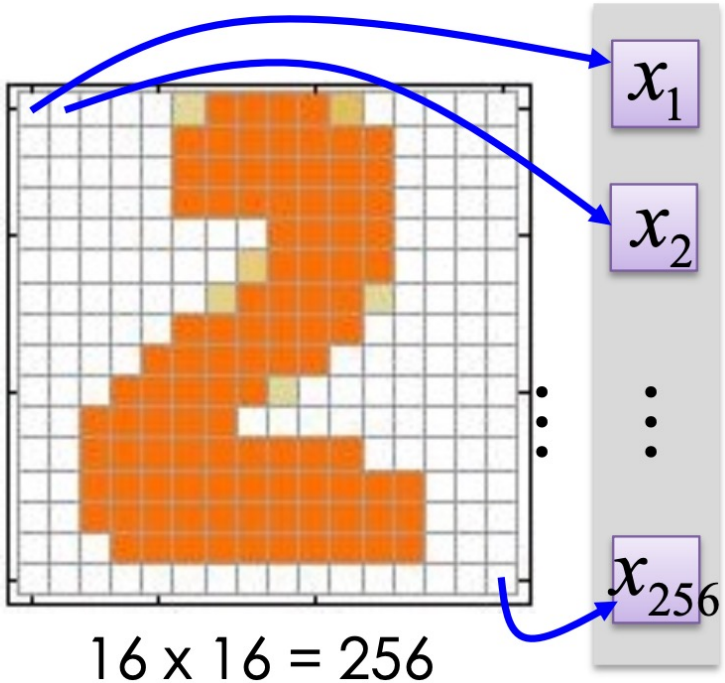
Input:
256-dim vector

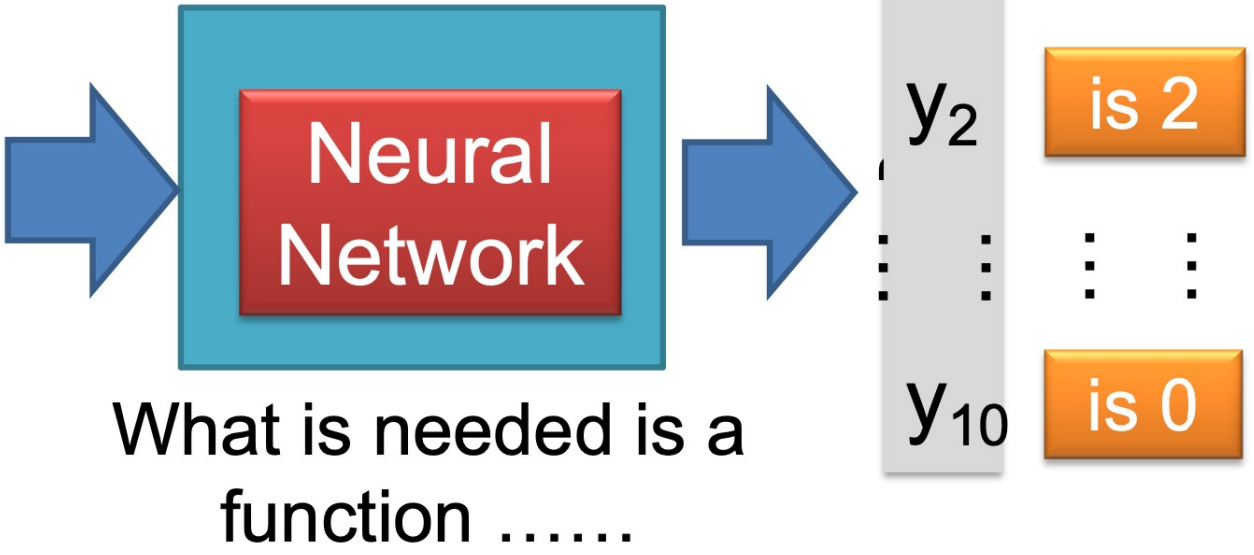
output:
10-dim vector

Credit: Hung-yi Lee, Andrew, Stanford course

Application Example: Handwriting Digit Recognition

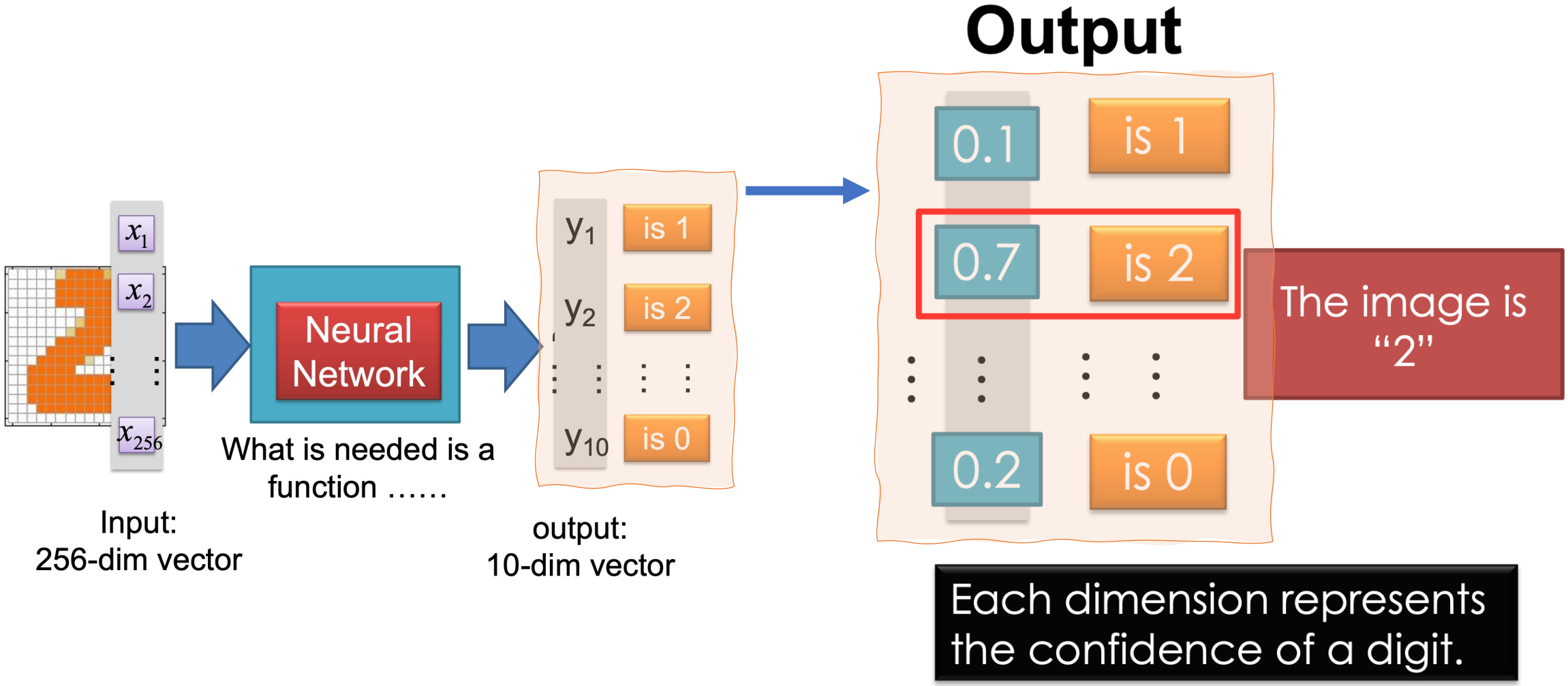
Input





Credit: Hung-yi Lee, Andrew, Stanford course

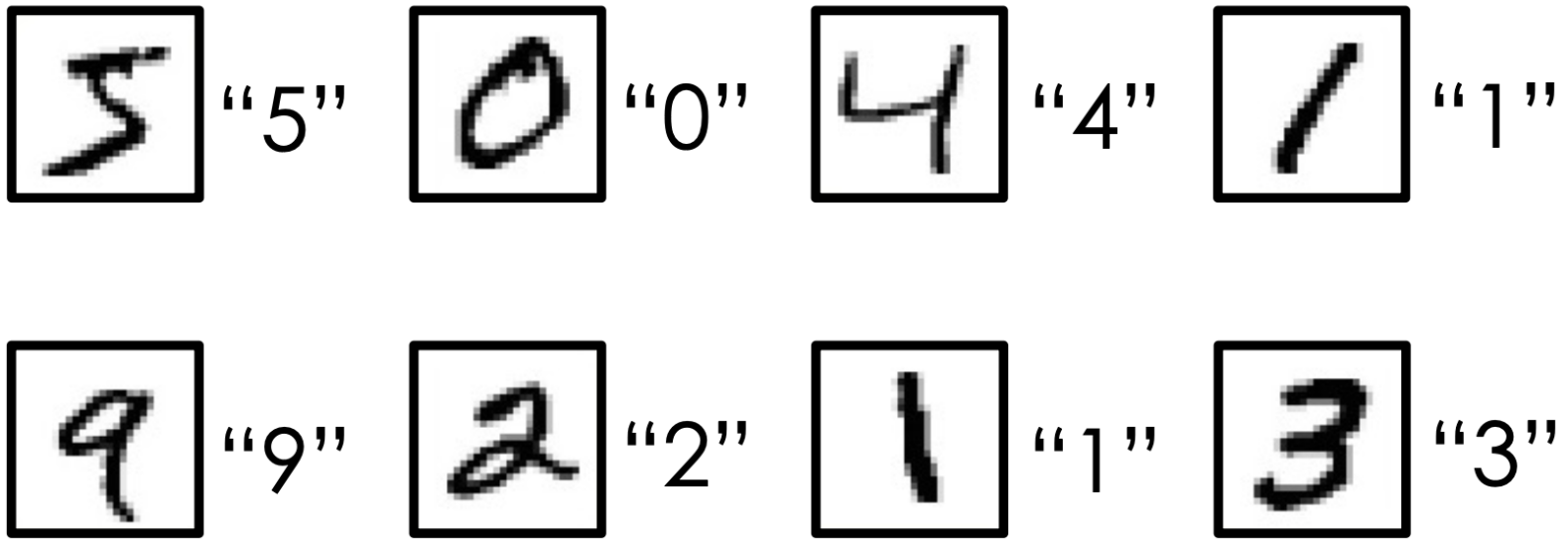
Application Example: Handwriting Digit Recognition



Credit: Hung-yi Lee, Andrew, Stanford course

University of Massachusetts Amherst

Preparing Training Data: Images and their labels

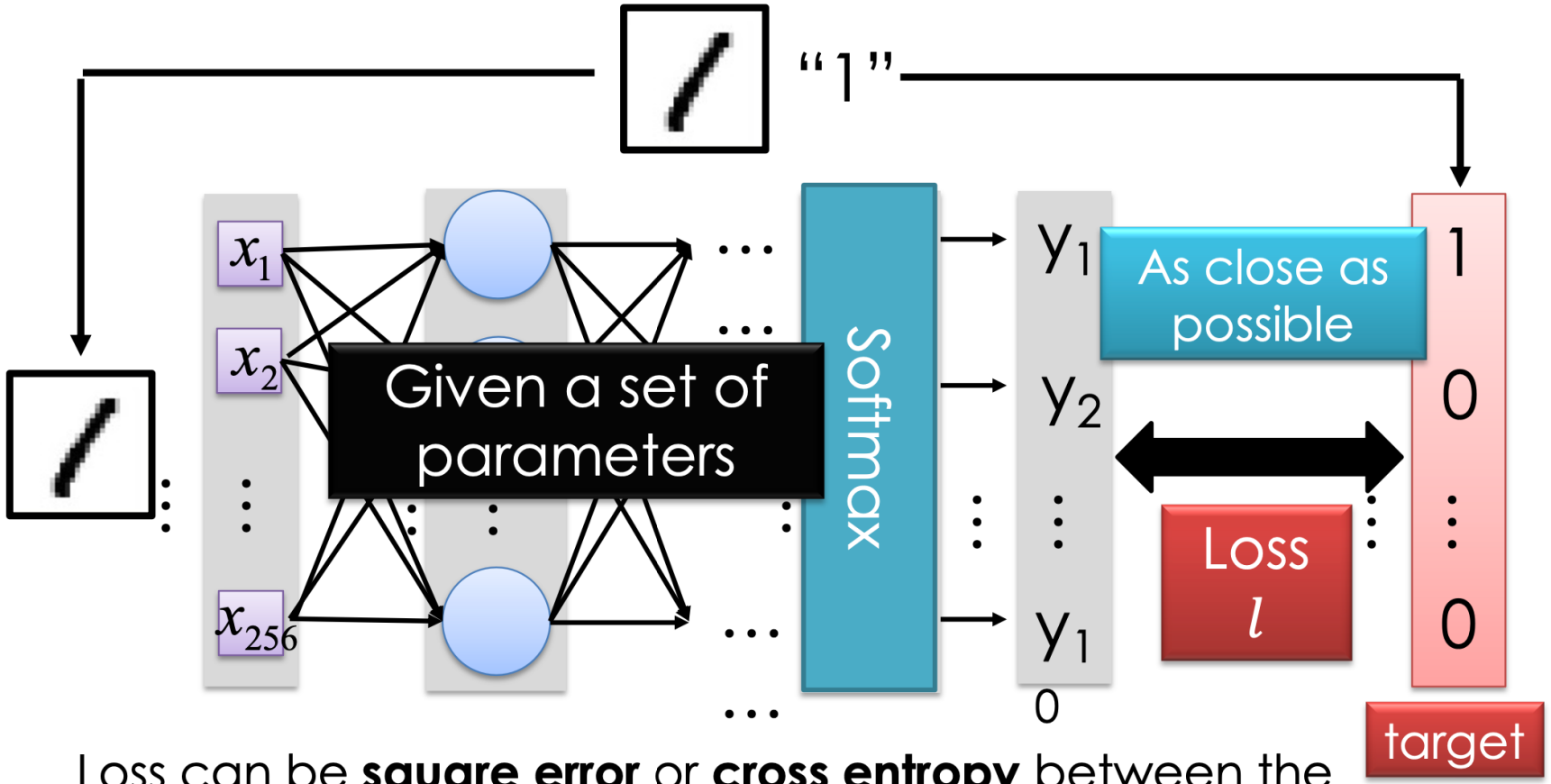


The learning target is defined on the training data.

Credit: Hung-yi Lee, Andrew, Stanford course

Loss

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

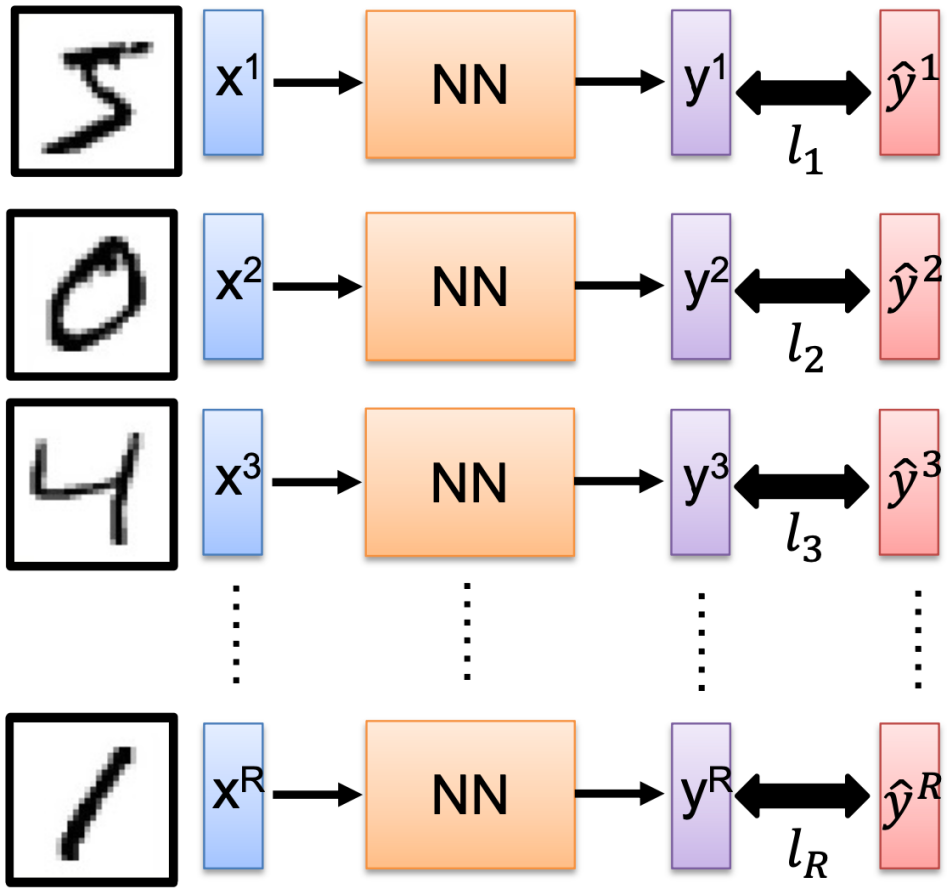


Loss can be **square error** or **cross entropy** between the network output and target

Credit: Hung-yi Lee, Andrew, Stanford course

Total Loss

For all training data …



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

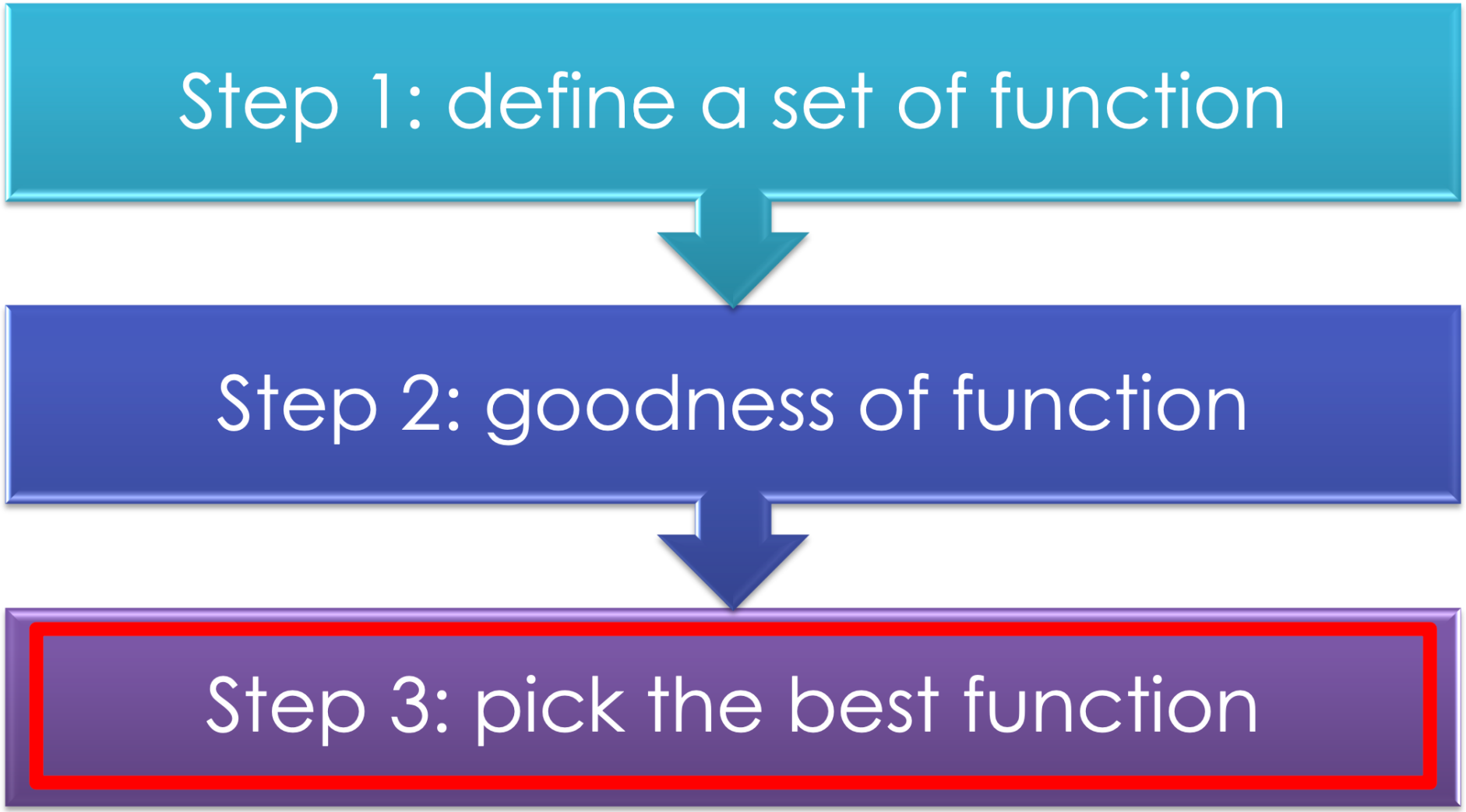
As small as possible

Find ***a function in function set*** that minimizes total loss L

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

Credit: Hung-yi Lee, Andrew, Stanford course

Three steps for Deep Learning



Testing or Inference

Credit: Hung-yi Lee, Andrew, Stanford course

Convolutional Neural Network (CNN)

Widely used in image processing and computer vision

Credit: Hung-yi Lee, Andrew, Stanford course

Why CNN for Images?

- Some patterns are much smaller than the whole image

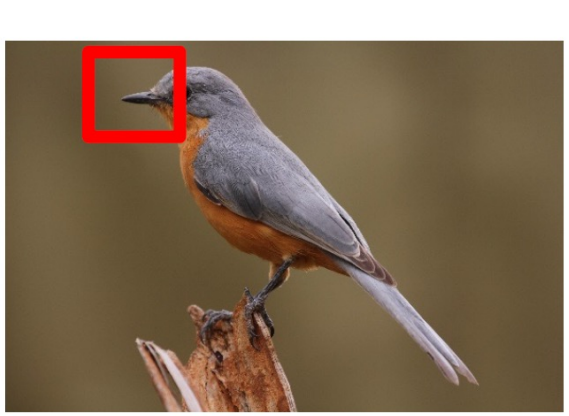
A neuron does not have to see the whole image to discover the pattern.

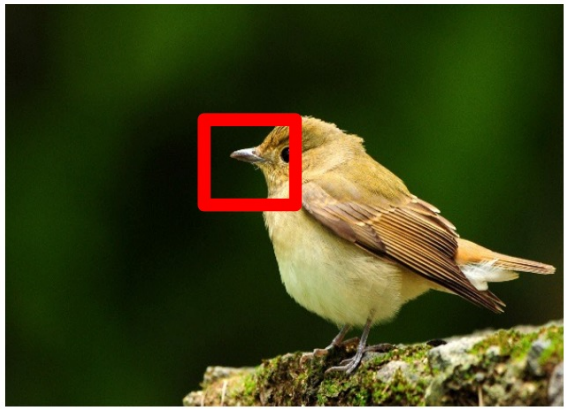
Connecting to small region with less parameters

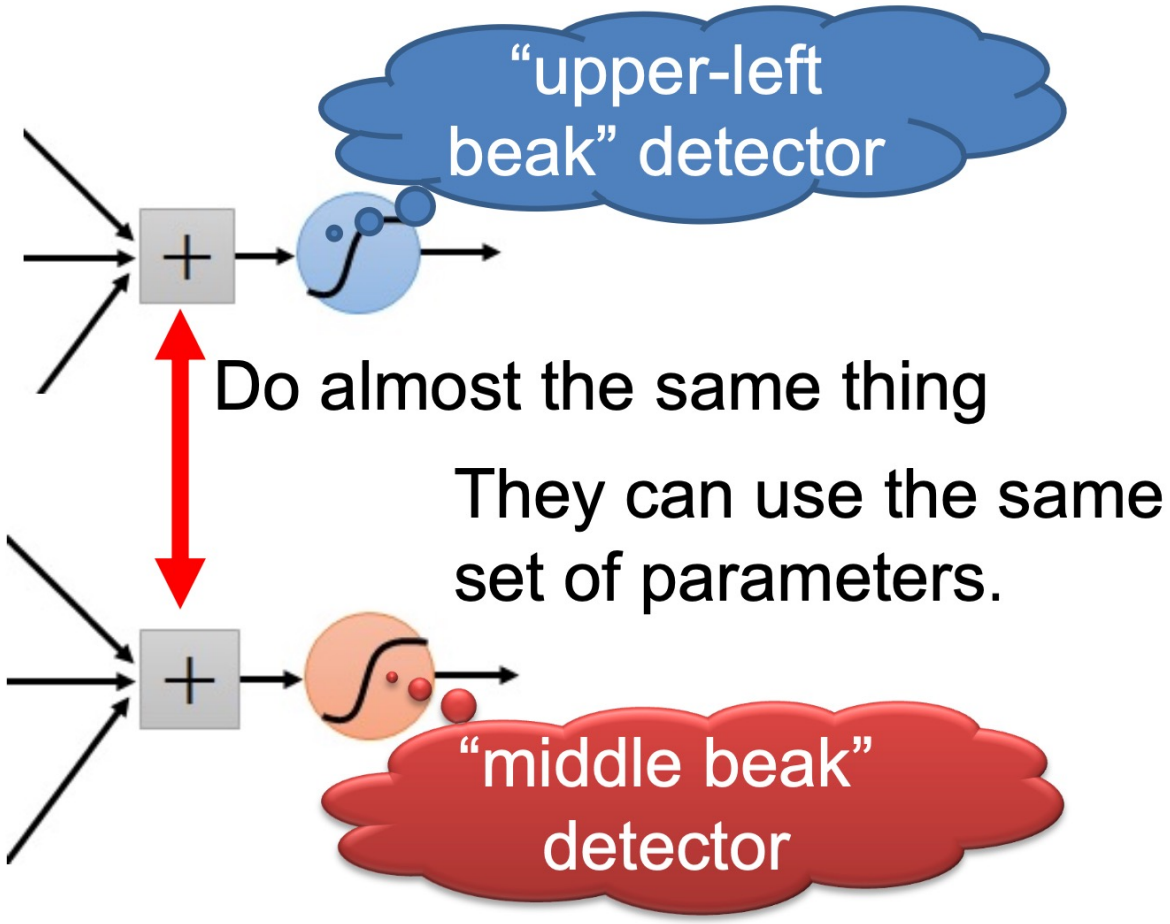
Credit: Hung-yi Lee, Andrew, Stanford course

Why CNN for Images?

- The same patterns appear in different regions.







Credit: Hung-yi Lee, Andrew, Stanford course

Why CNN for Images?

- 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

Credit: Hung-yi Lee, Andrew, Stanford course

University of Massachusetts Amherst

Why CNN for Images?

Property 1

- Some patterns are much smaller than the whole

Property 2

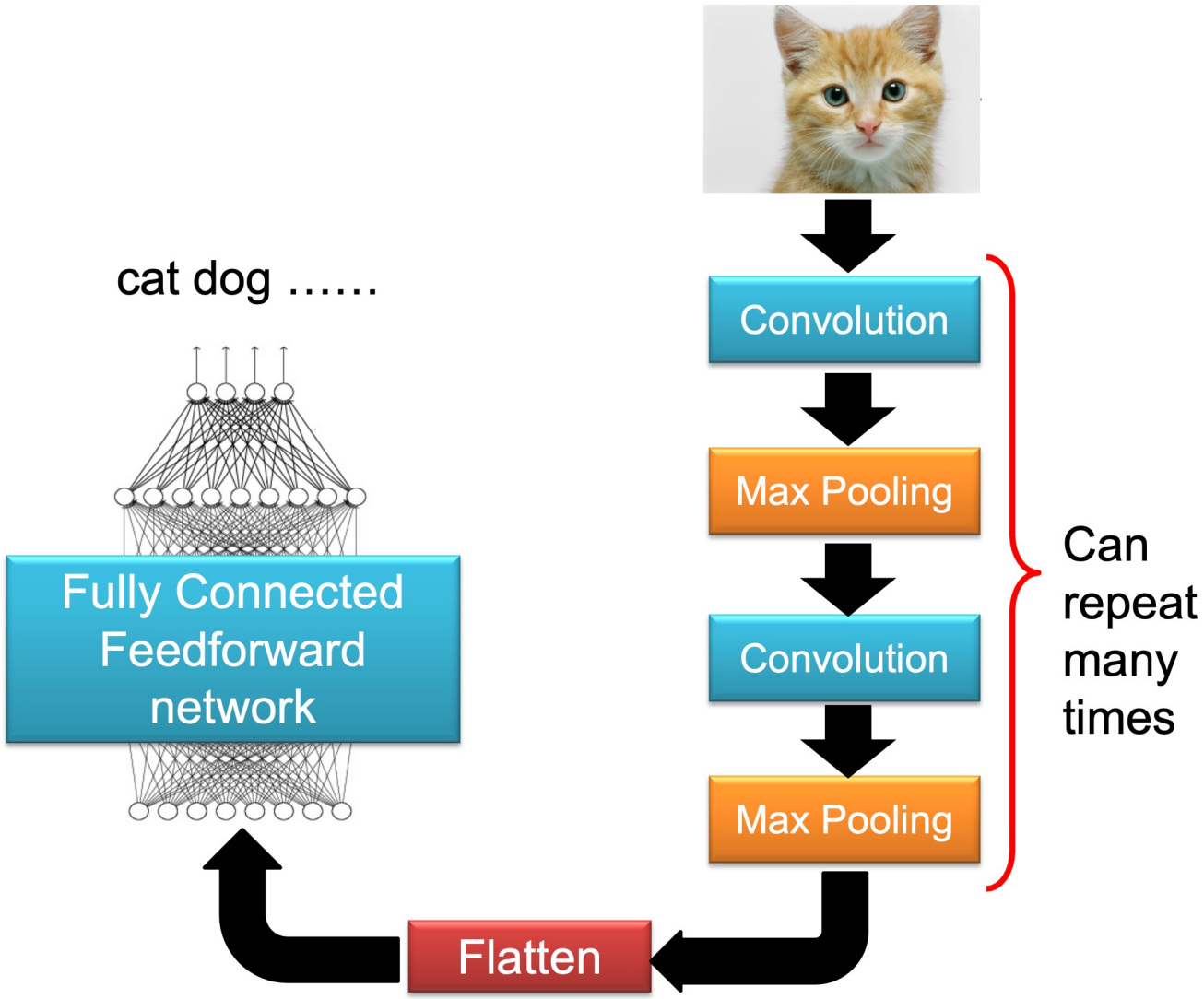
- The same patterns appear in different regions.

Property 3

- Subsampling the pixels will not change the object

Credit: Hung-yi Lee, Andrew, Stanford course

The Whole CNN



Credit: Hung-yi Lee, Andrew, Stanford course

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

Credit: Hung-yi Lee, Andrew, Stanford course

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

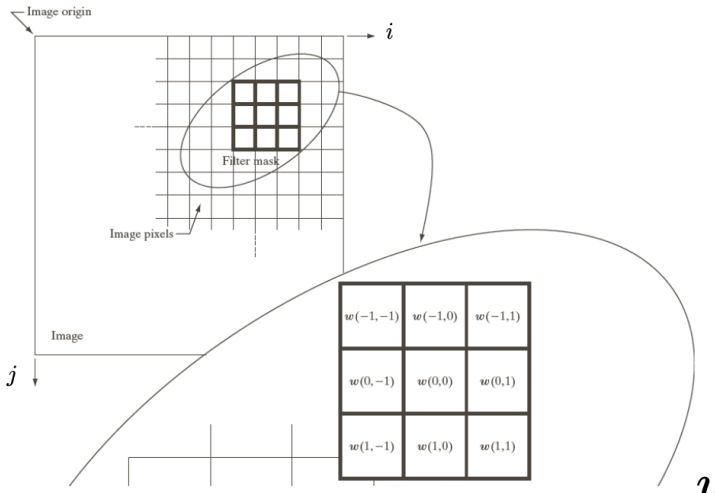
Matrix

⋮ ⋮

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

University of Massachusetts Amherst

CNN: Convolution



Filter/Kernel 1

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

filter previous level

$$u_{i,j}^{L} = \sum_{m,n} w_{i,j,m,n}^{L} a_{i-m,j-n}^{L-1}$$

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

3 -1

If stride=2

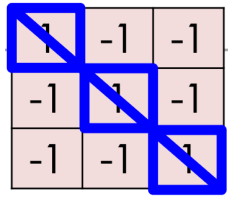
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

3 -3

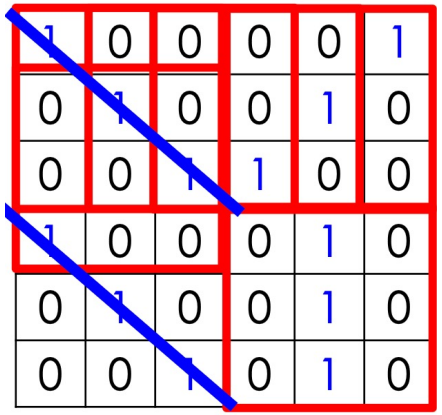
We set stride=1 below

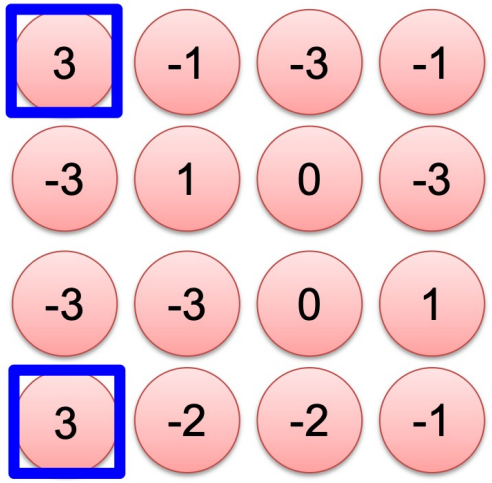
Credit: Hung-yi Lee, Andrew, Stanford course

CNN: Convolution



stride=1





Credit: Hung-yi Lee, Andrew, Stanford course

CNN: Convolution

Filter 2

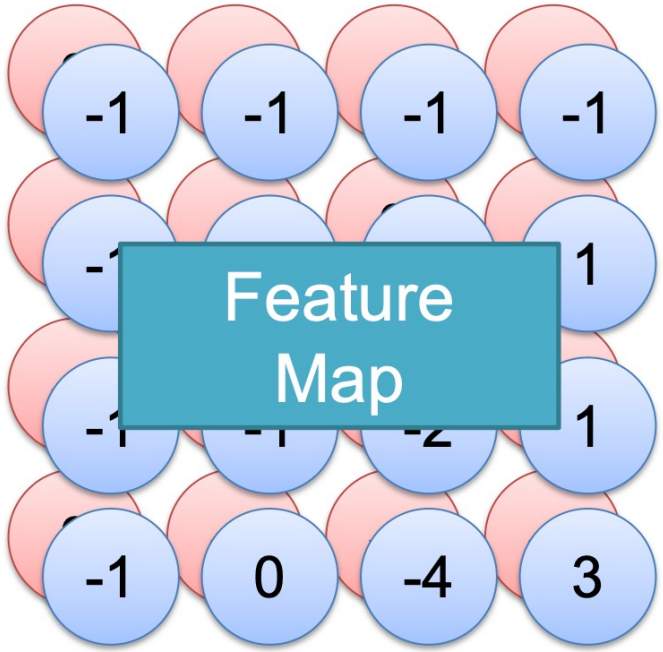
-1	1	-1
-1	1	-1
-1	1	-1

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

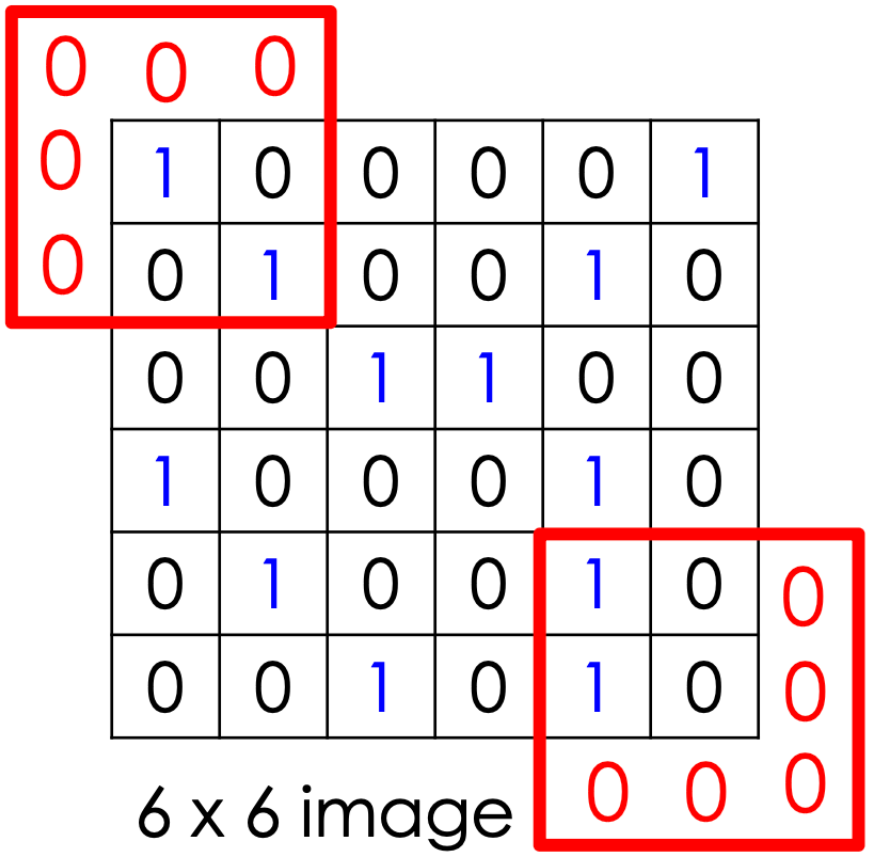
Do the same process for every filter



4 x 4 image (reduced size)

Credit: Hung-yi Lee, Andrew, Stanford course

CNN: Zero Padding

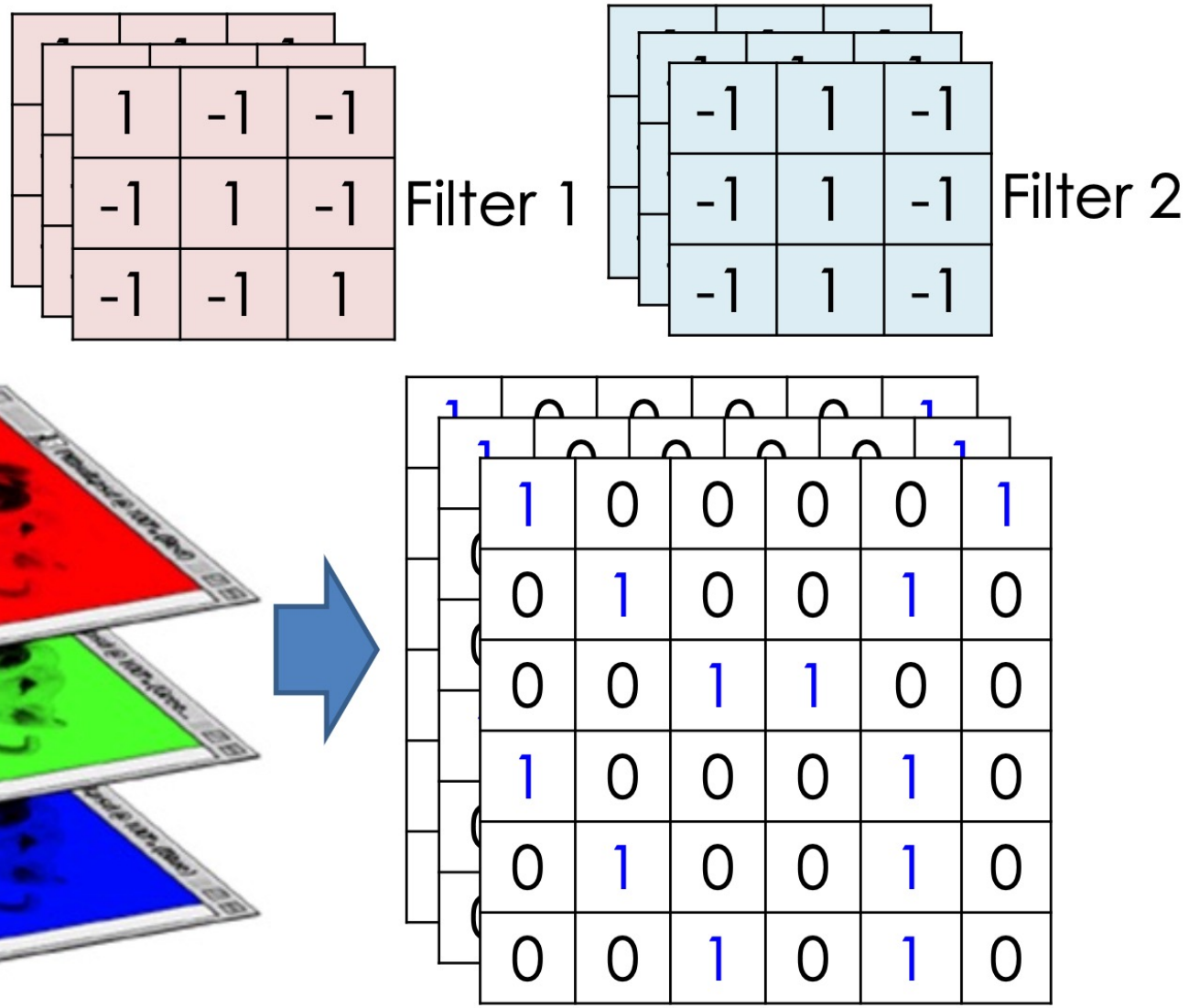


You will get another 6 x 6 image in this way

Zero padding

Credit: Hung-yi Lee, Andrew, Stanford course

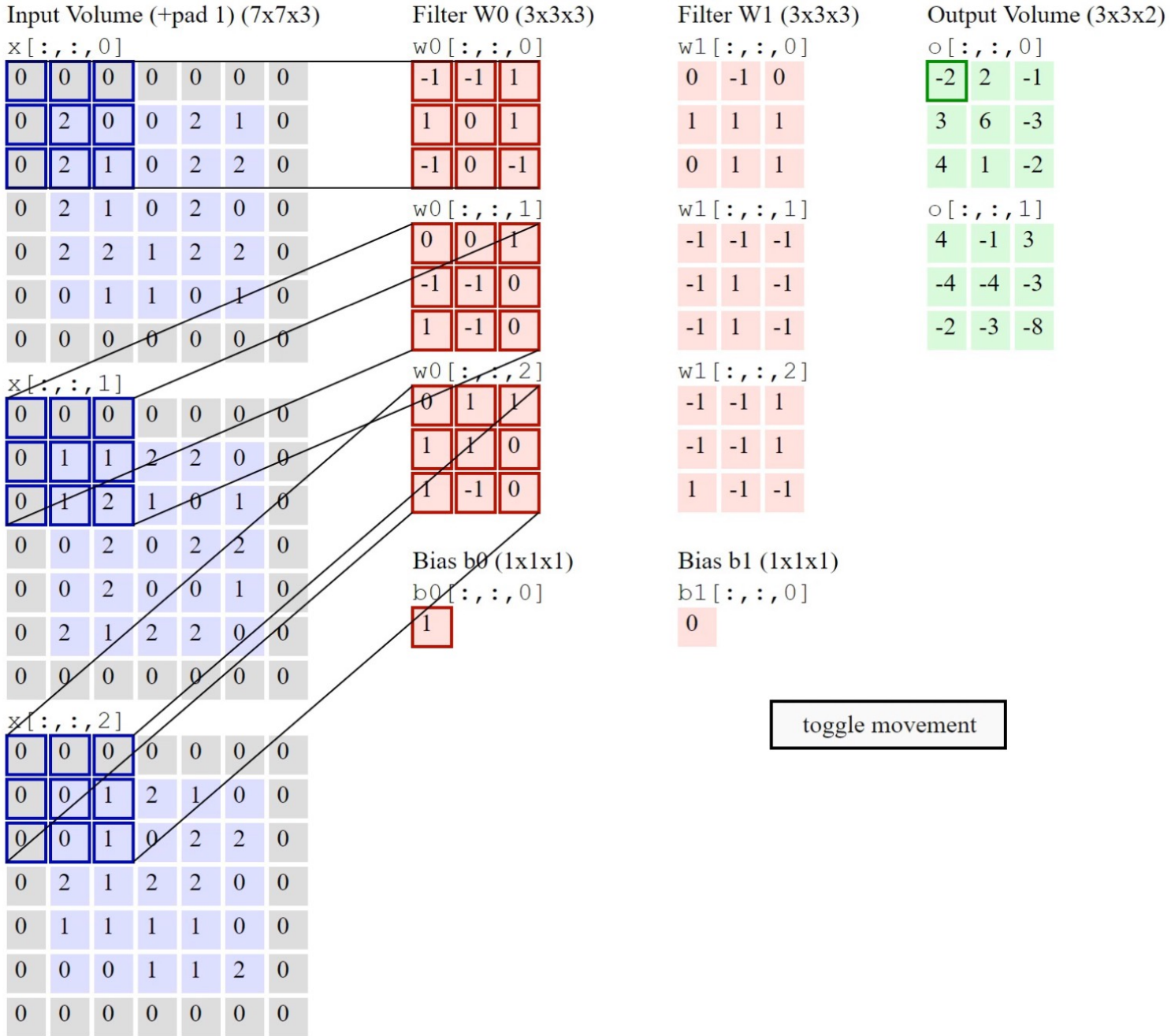
CNN: Color Image



Credit: Hung-yi Lee, Andrew, Stanford course

University of Massachusetts Amherst

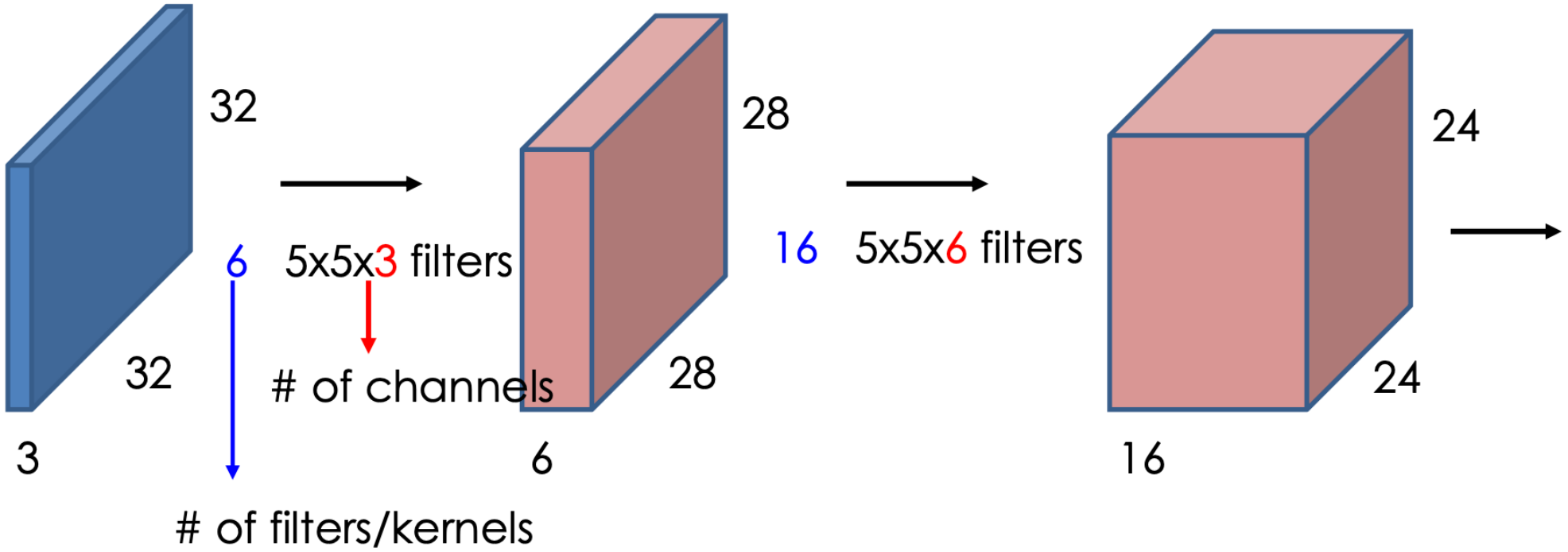
CNN: Color Image



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

Convolutional Network

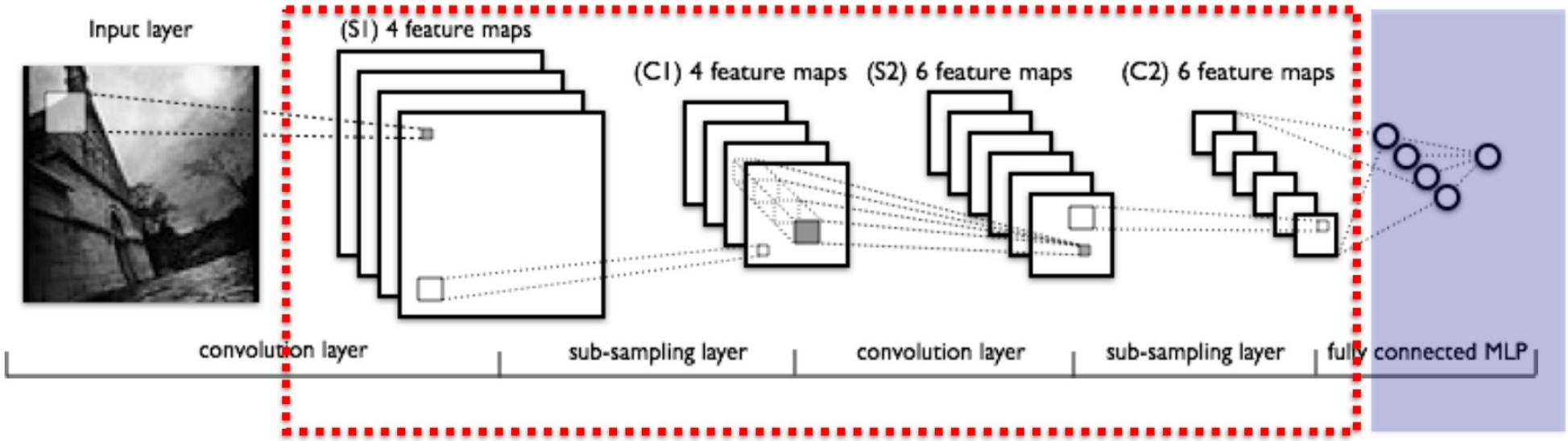
- Convolution network is a sequence of these layers



Credit: Hung-yi Lee, Andrew, Stanford course

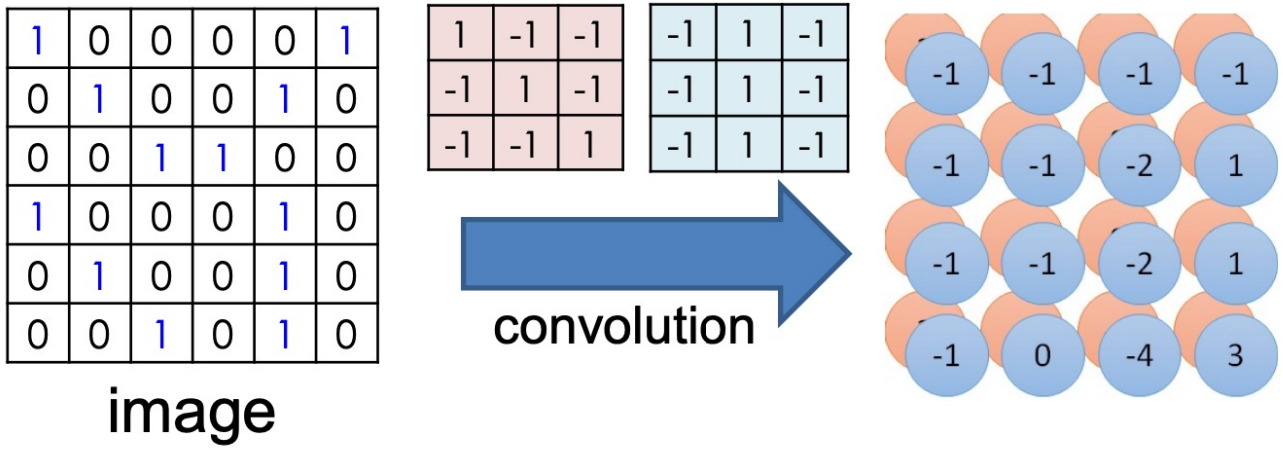
University of Massachusetts Amherst

Convolutional Network

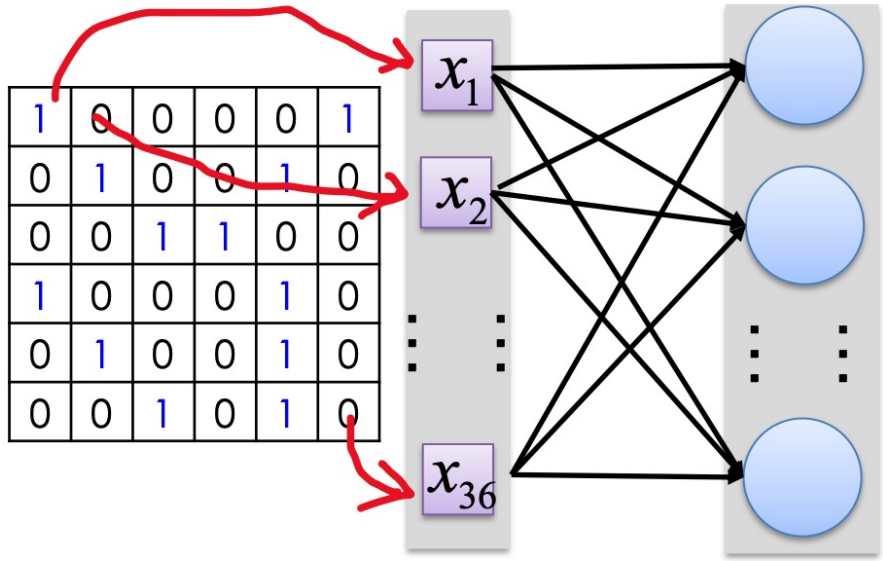


Feature extraction using convolutional layers

Convolution vs. Fully Connected

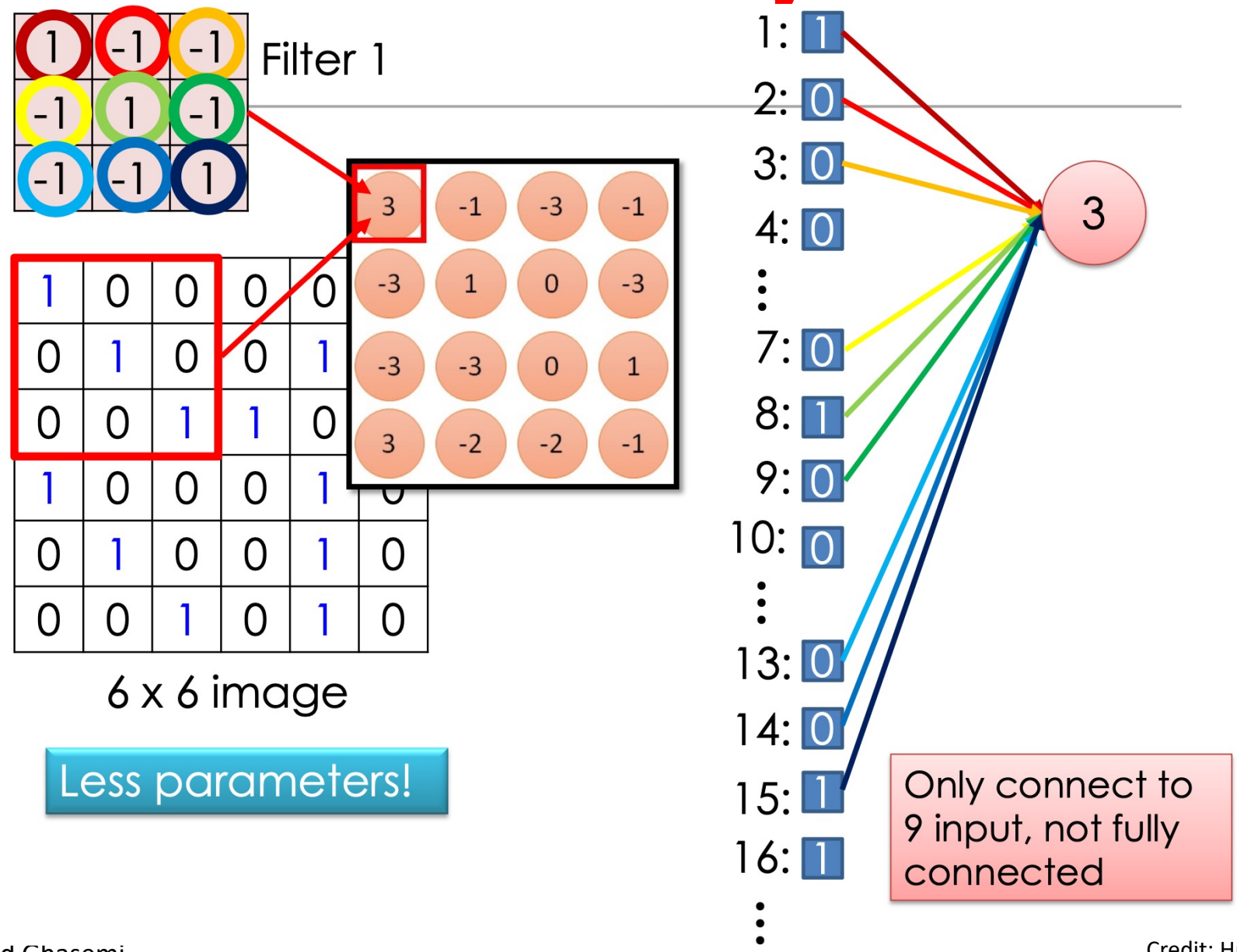


Fully-connected



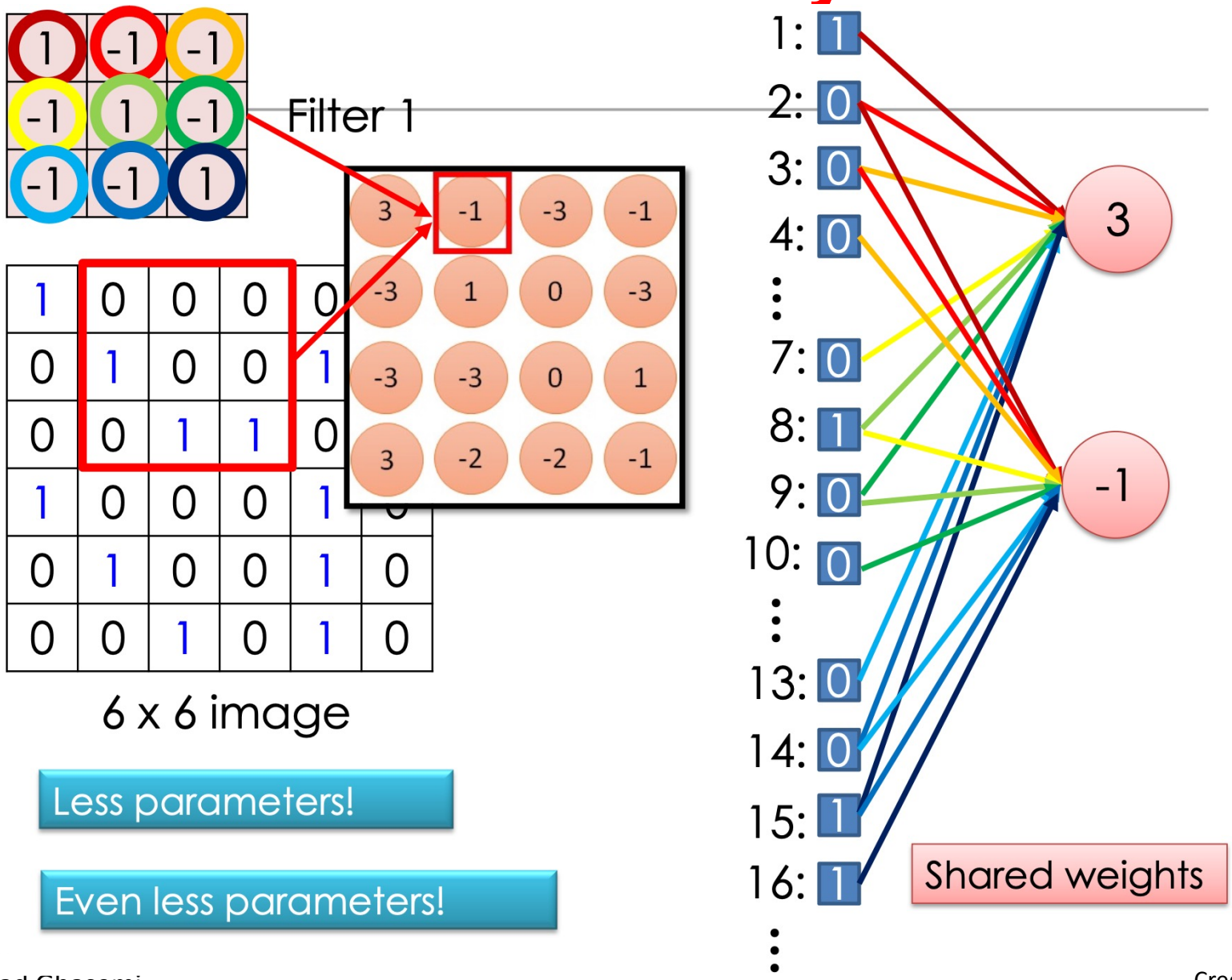
Credit: Hung-yi Lee, Andrew, Stanford course

Convolution vs. Fully Connected



Credit: Hung-yi Lee, Andrew, Stanford course

Convolution vs. Fully Connected



Less parameters!

Even less parameters!

Credit: Hung-yi Lee, Andrew, Stanford course