UNIVERSITY OF SOUTH DAKOTA

A Gentle Introduction to Deep Learning

Dr. Rodrigue Rizk

Rodrigue.Rizk@usd.edu

Learning Objectives

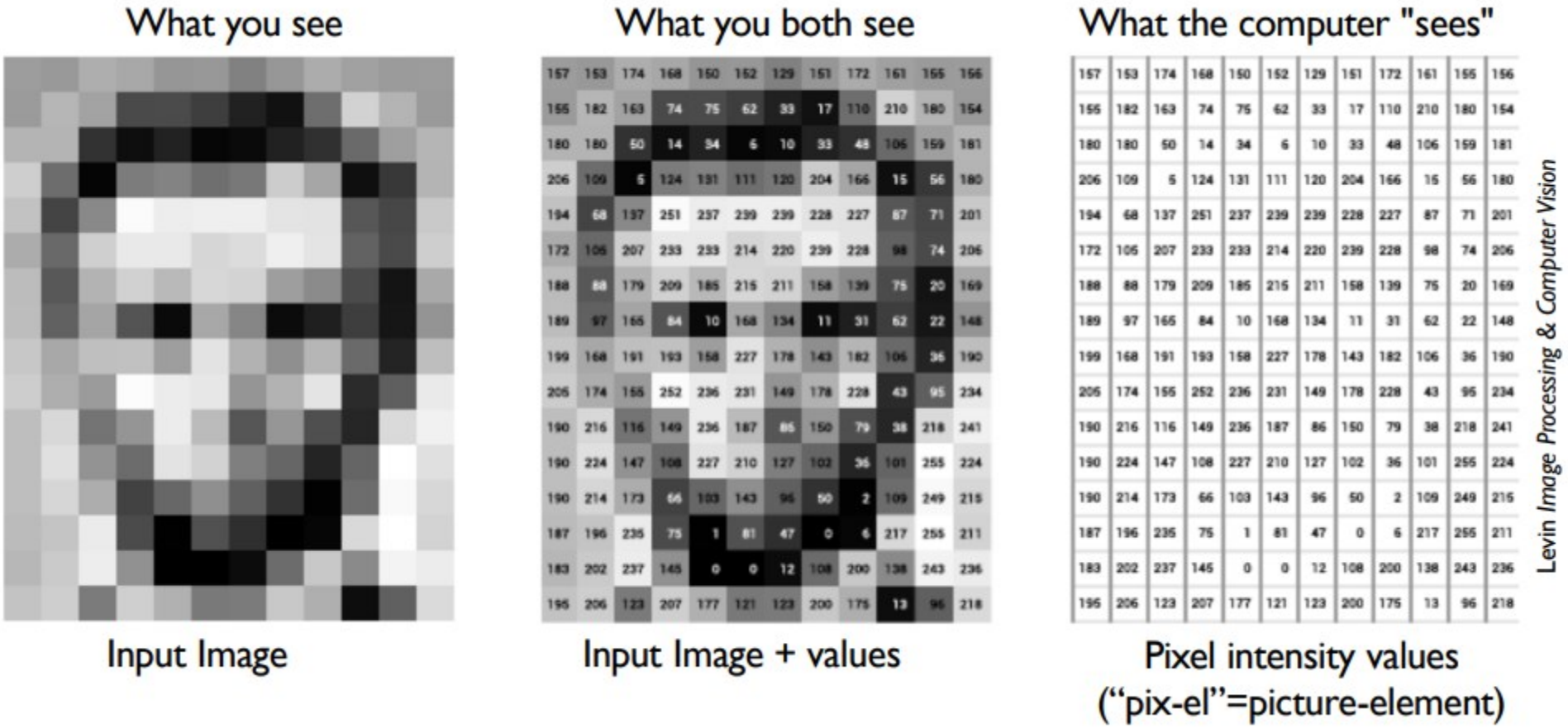
UNIVERSITY OF SOUTH DAKOTA

- Understand the intuition behind deep neural networks particularly convolutional neural networks (CNN)
- Learn the machine vision foundations
- Understand the building blocks of CNN
- Get hands-on experience with CNN

What Do You See?

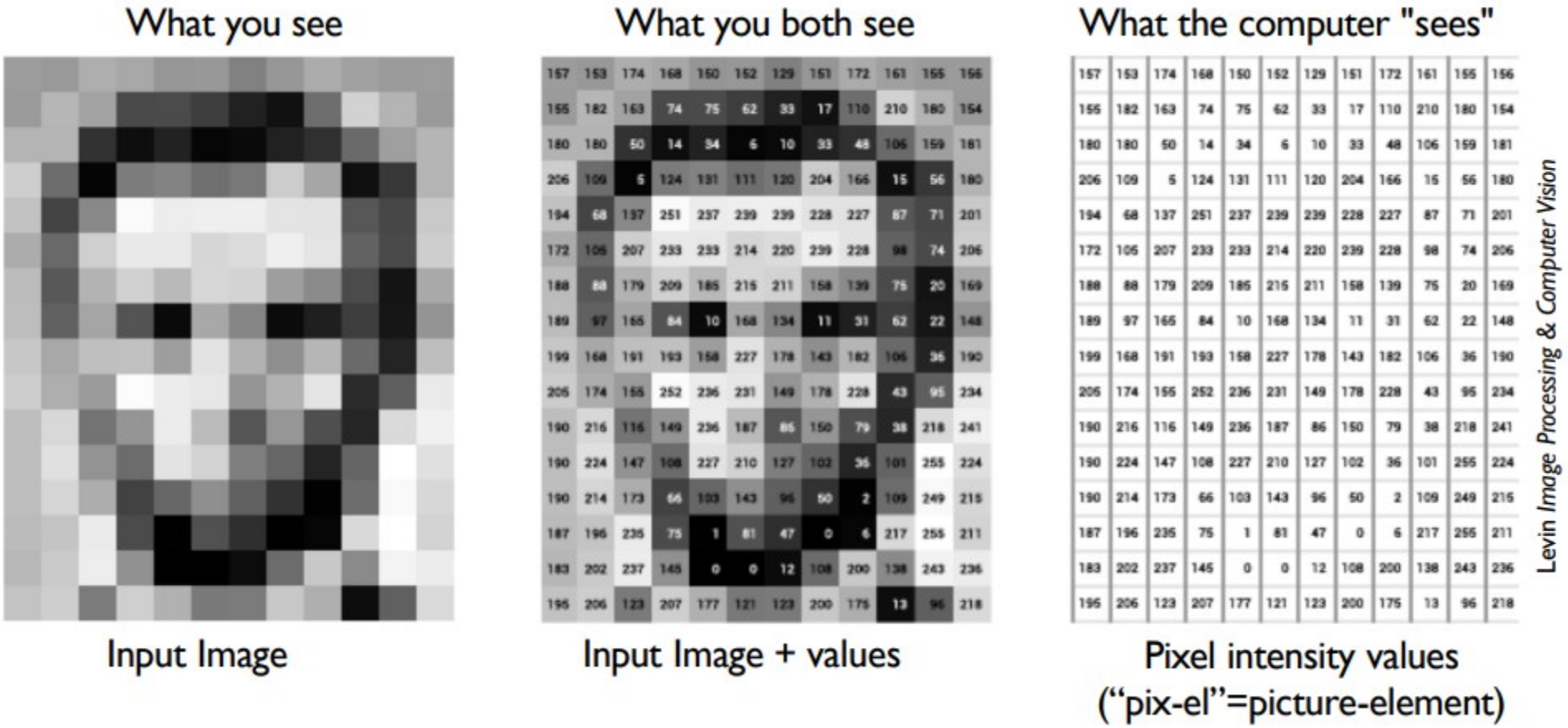
How can we help computers see?

What computers 'see': Images as Numbers



An image is just a matrix of numbers [0,255]

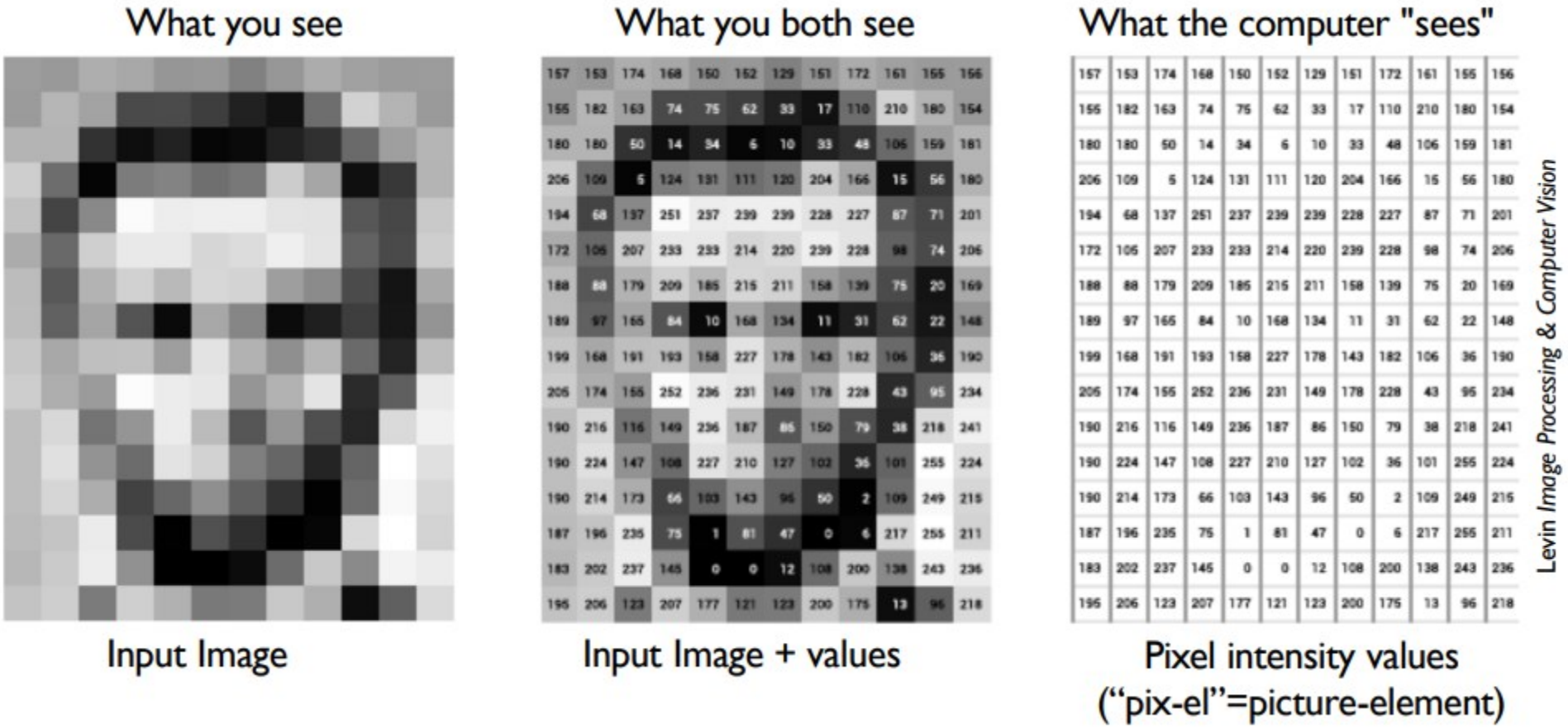
What computers 'see': Images as Numbers



An image is just a matrix of numbers [0,255]

Can I just do classification on this long image vector directly?

What computers 'see': Images as Numbers



An image is just a matrix of numbers [0,255]

Can I just do classification on this long image vector directly?

No. Instead: exploit image spatial structure. Learn patches. Build them up

High Level Feature Detection

UNIVERSITY OF SOUTH DAKOTA

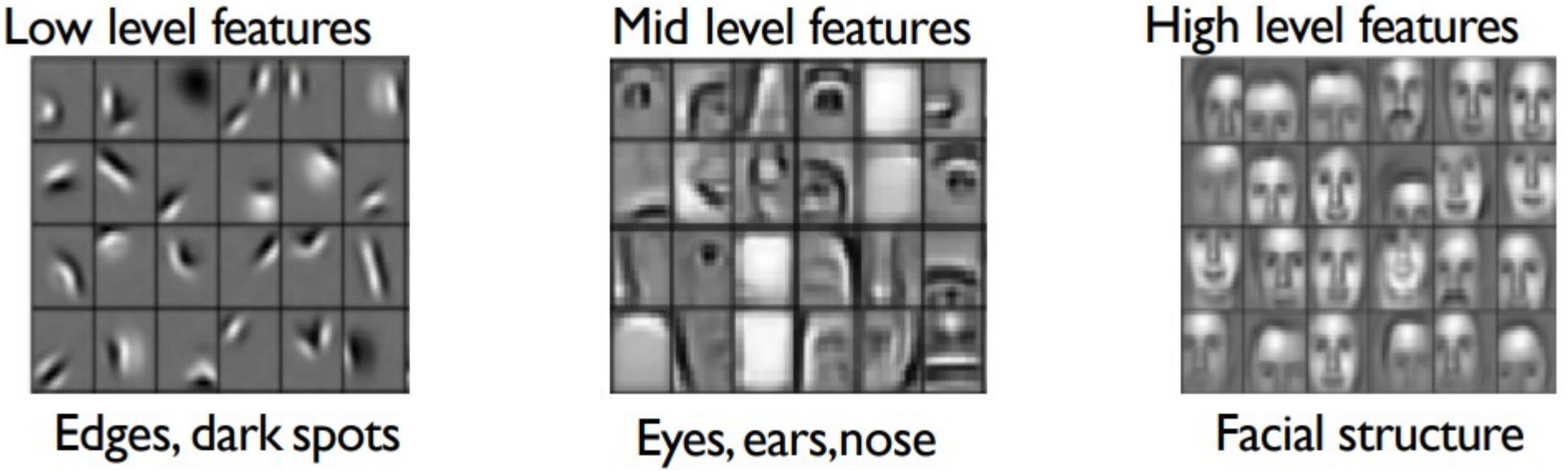
Let's identify key features in each image category

Nose, Eyes, Mouth

Wheels, License Plate, Headlights

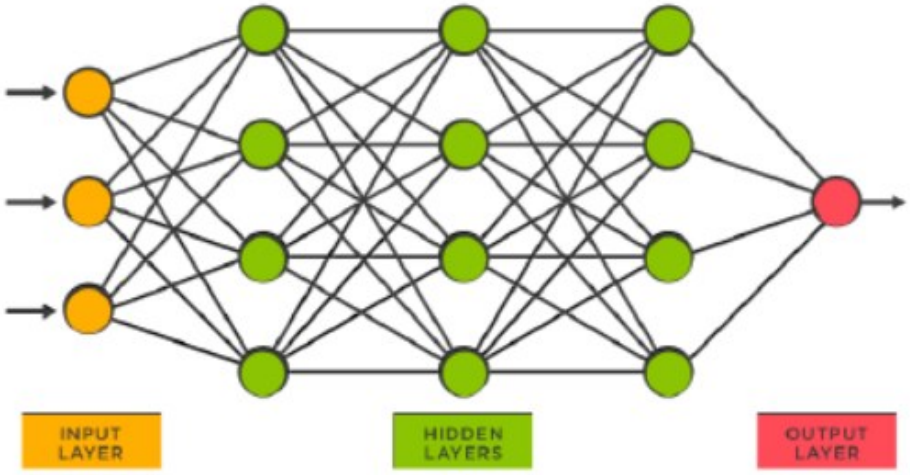
Door, Windows, Steps

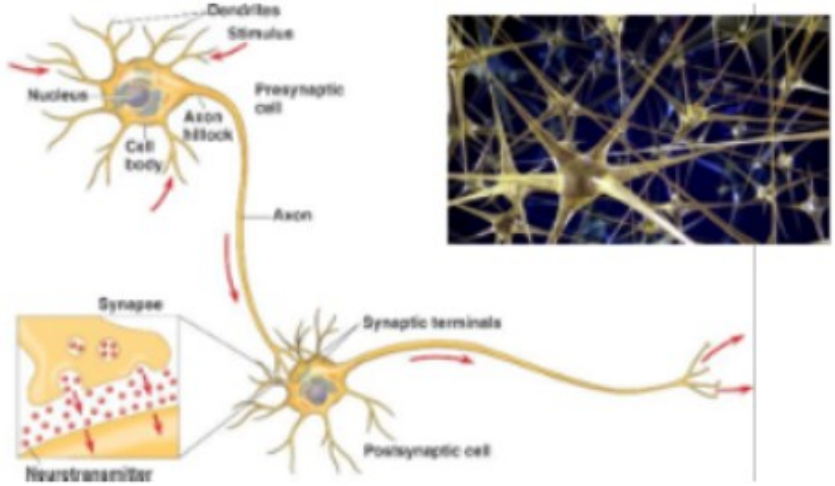
High Level Feature Detection



Learn the hierarchy of features directly from the data

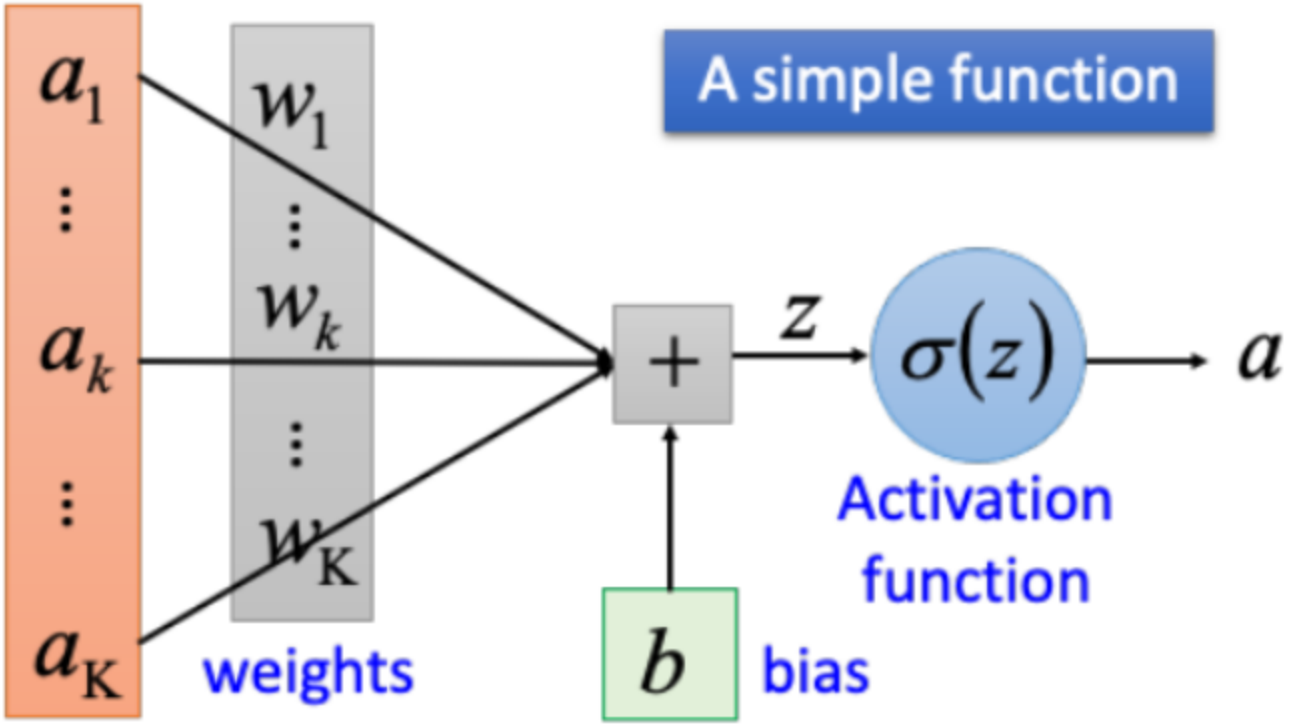
Neural Network



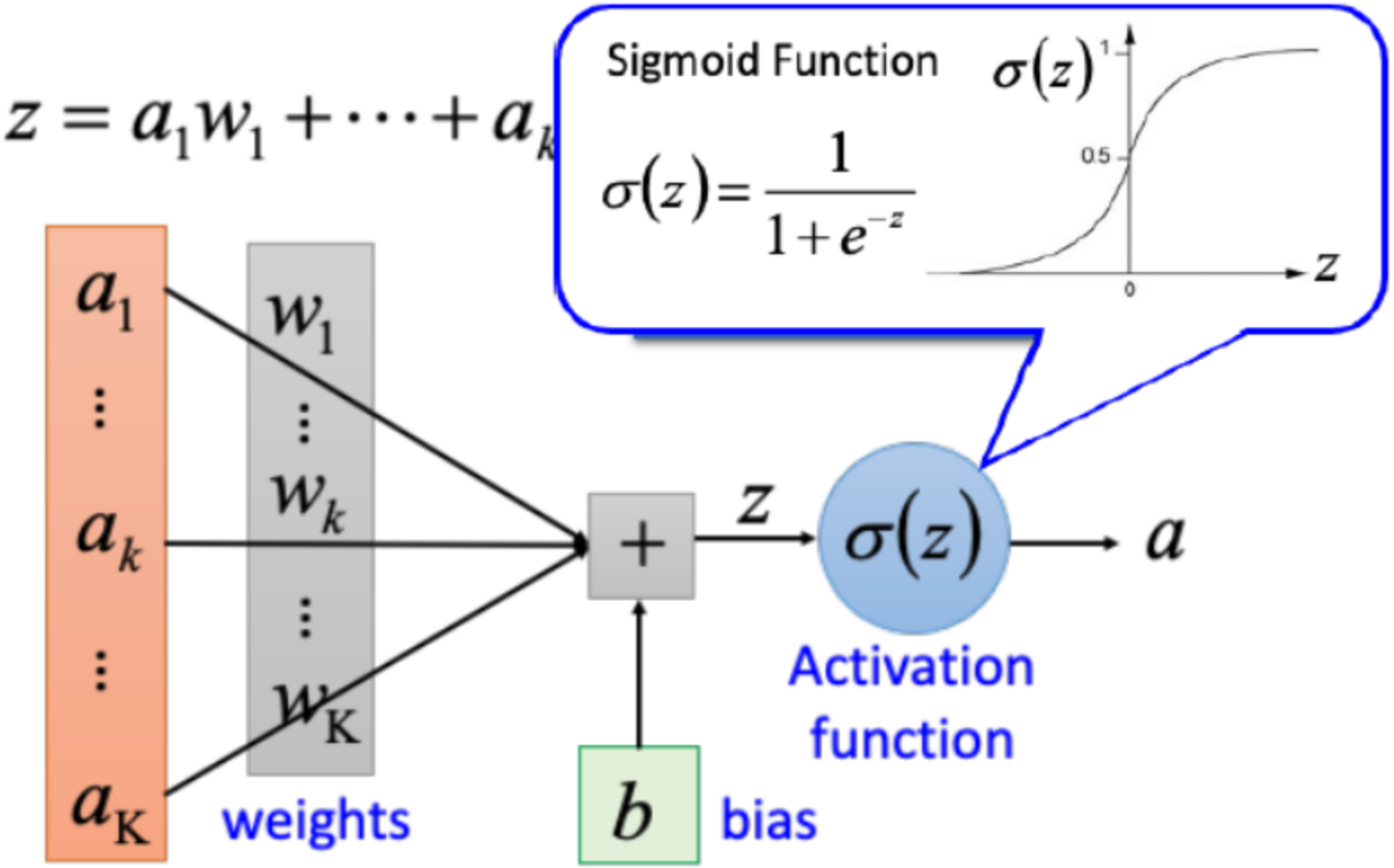


At Each Neuron

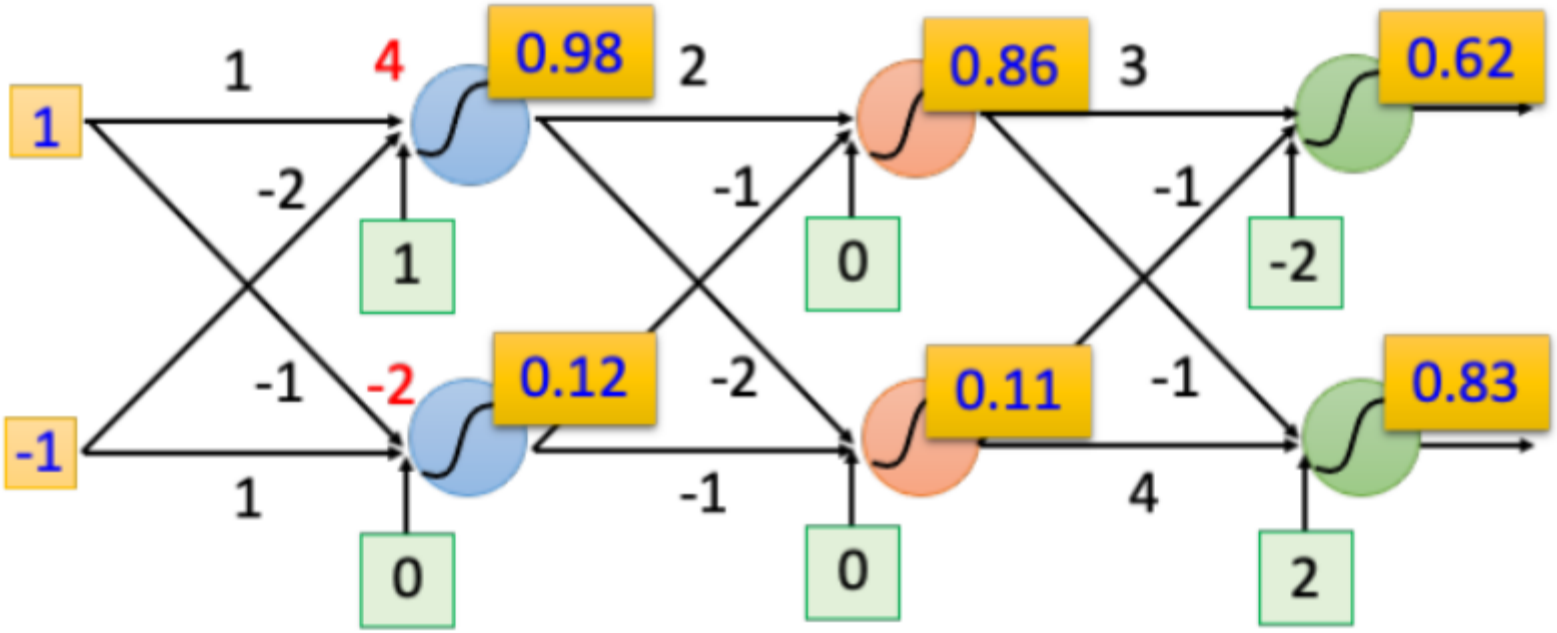
$$z = a_1 w_1 + \cdots + a_k w_k + \cdots + a_K w_K + b$$



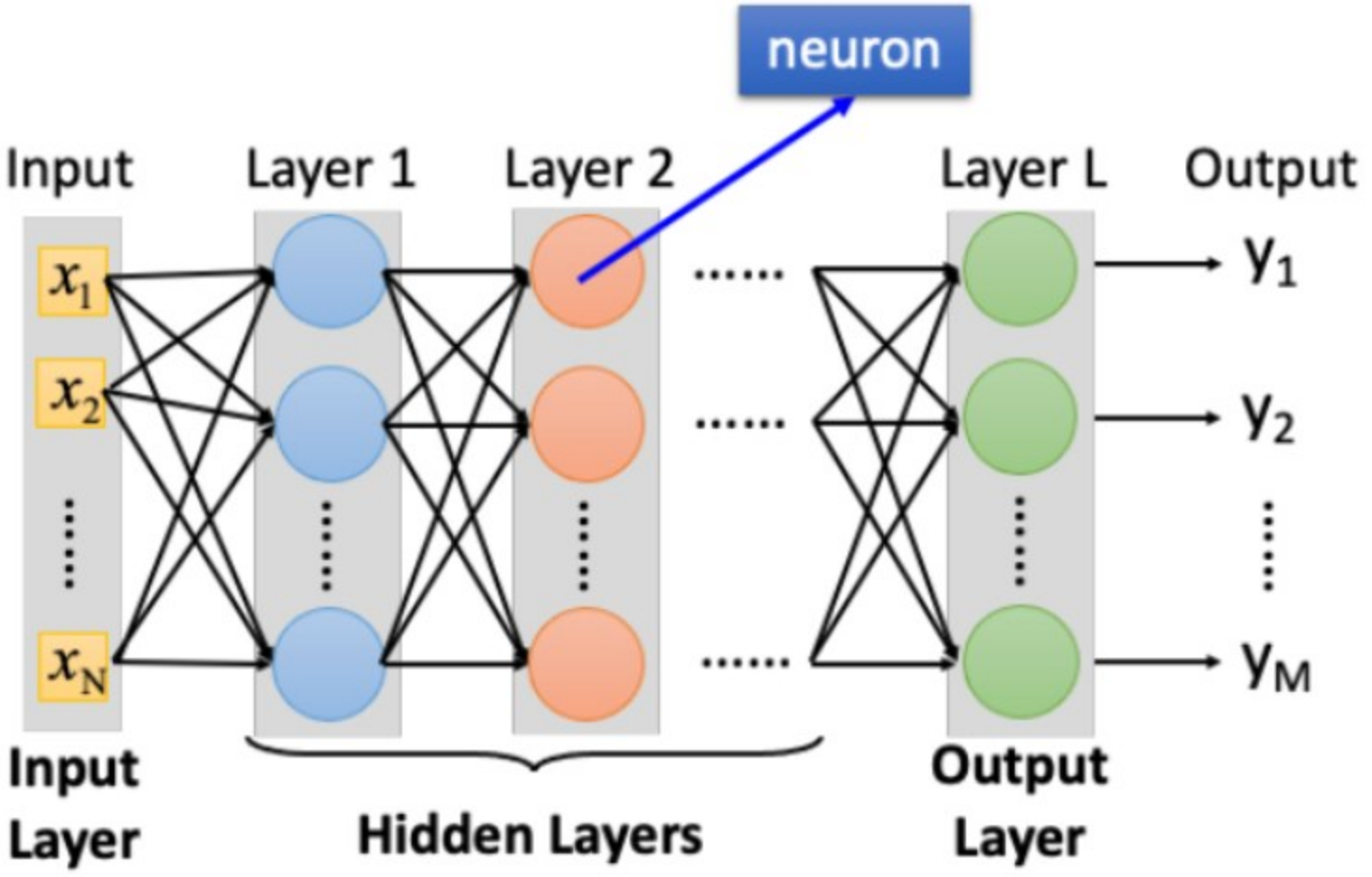
At Each Neuron



Fully Connected Neural Network

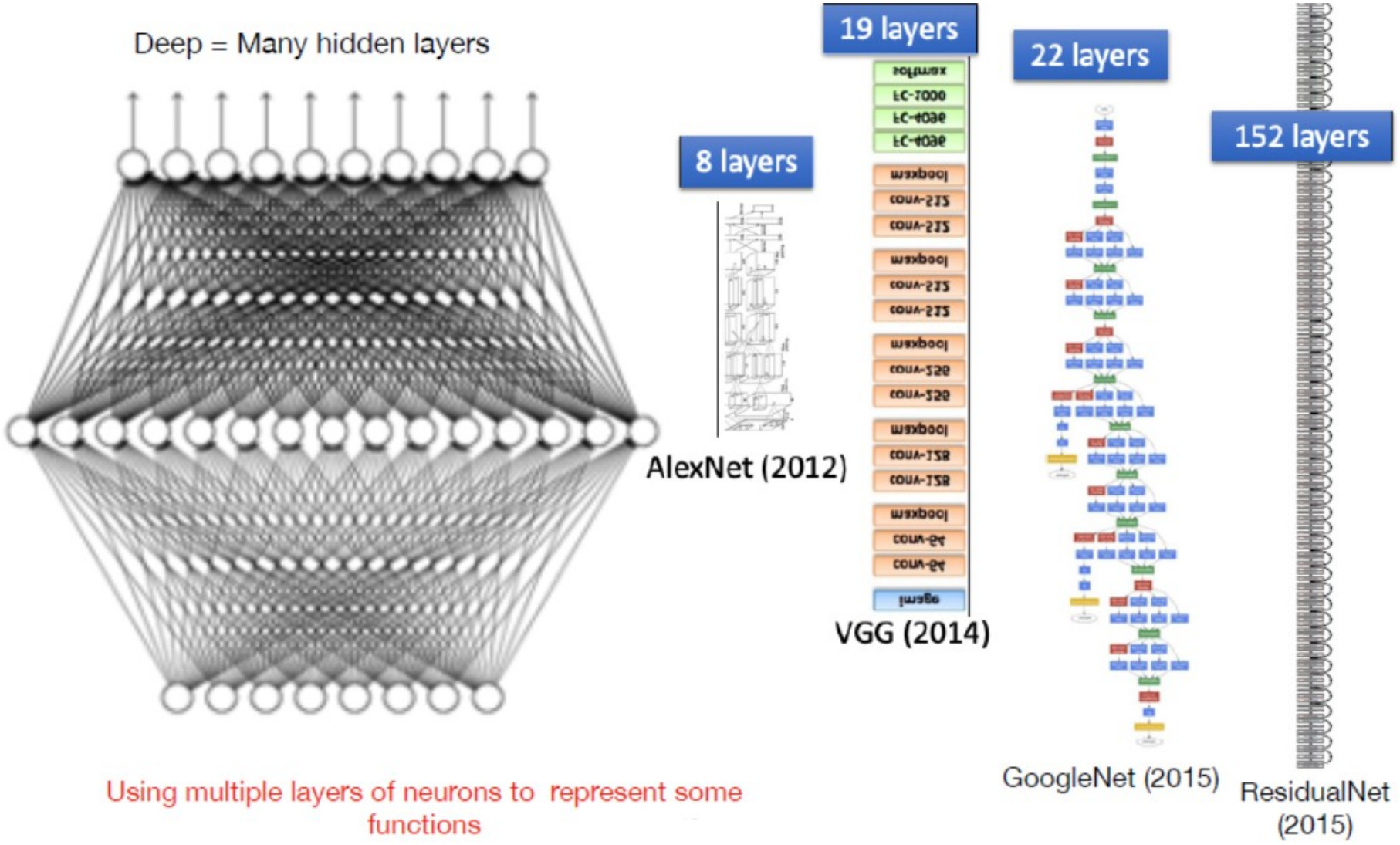


Fully Connected Neural Network



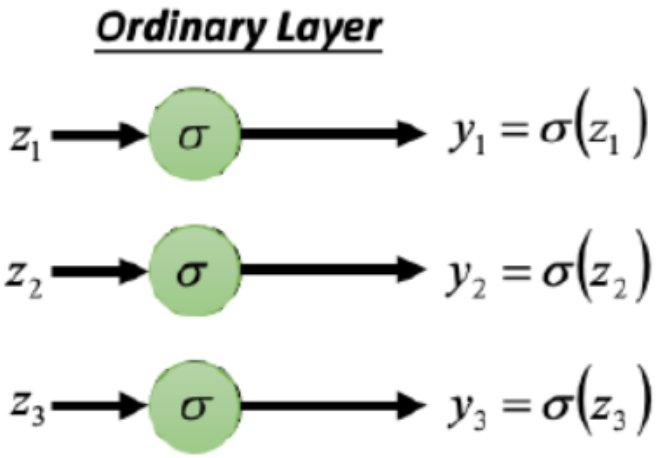
Deep means many hidden layers

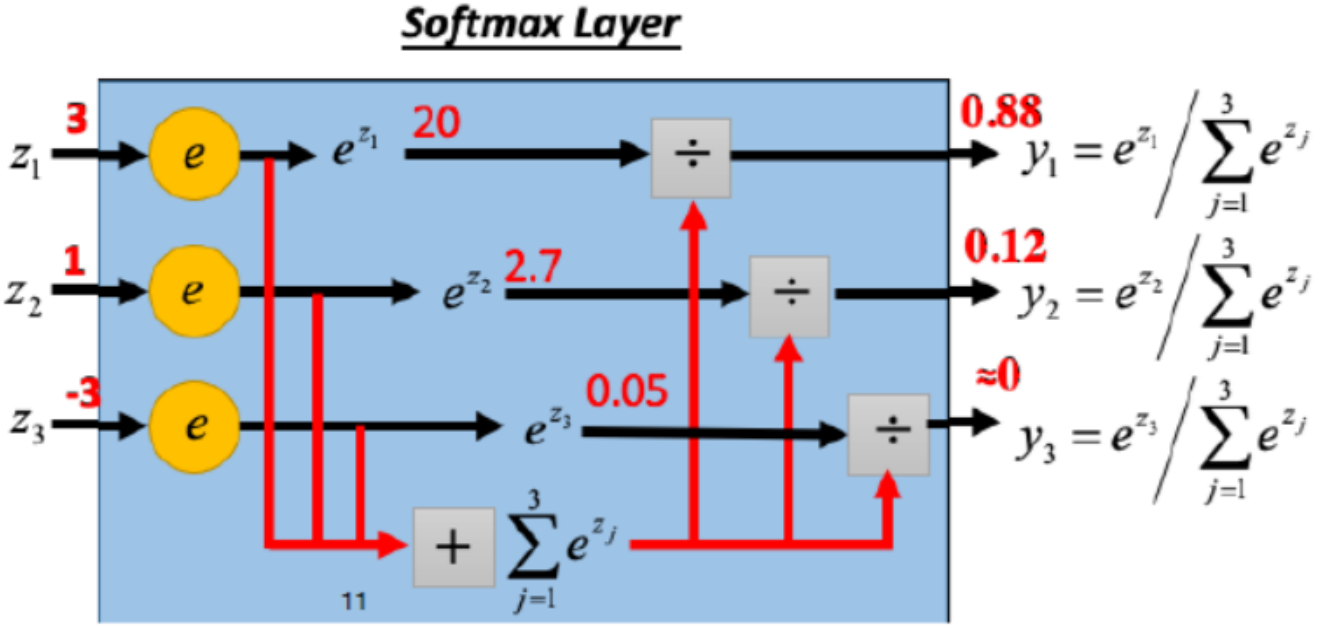
Deep Neural Network



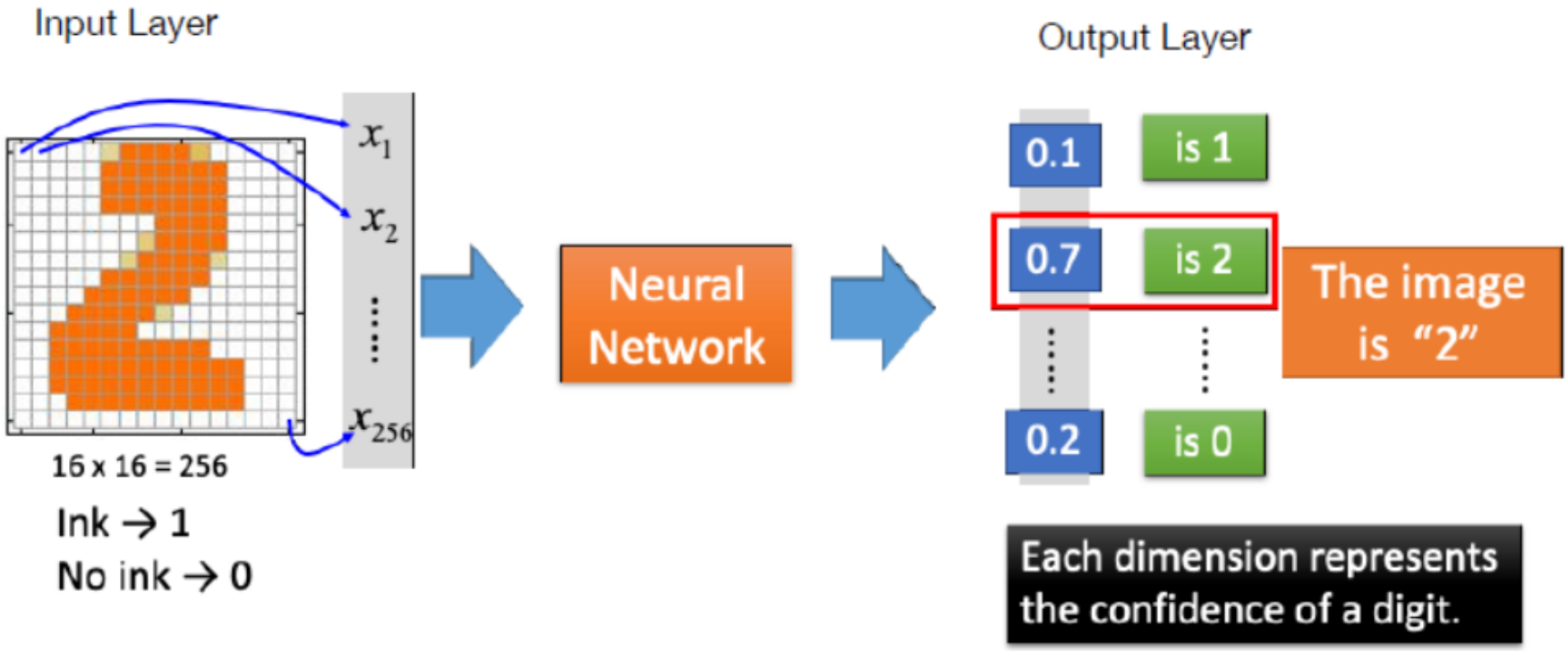
Using multiple layers of neurons to represent some functions

Output Layer

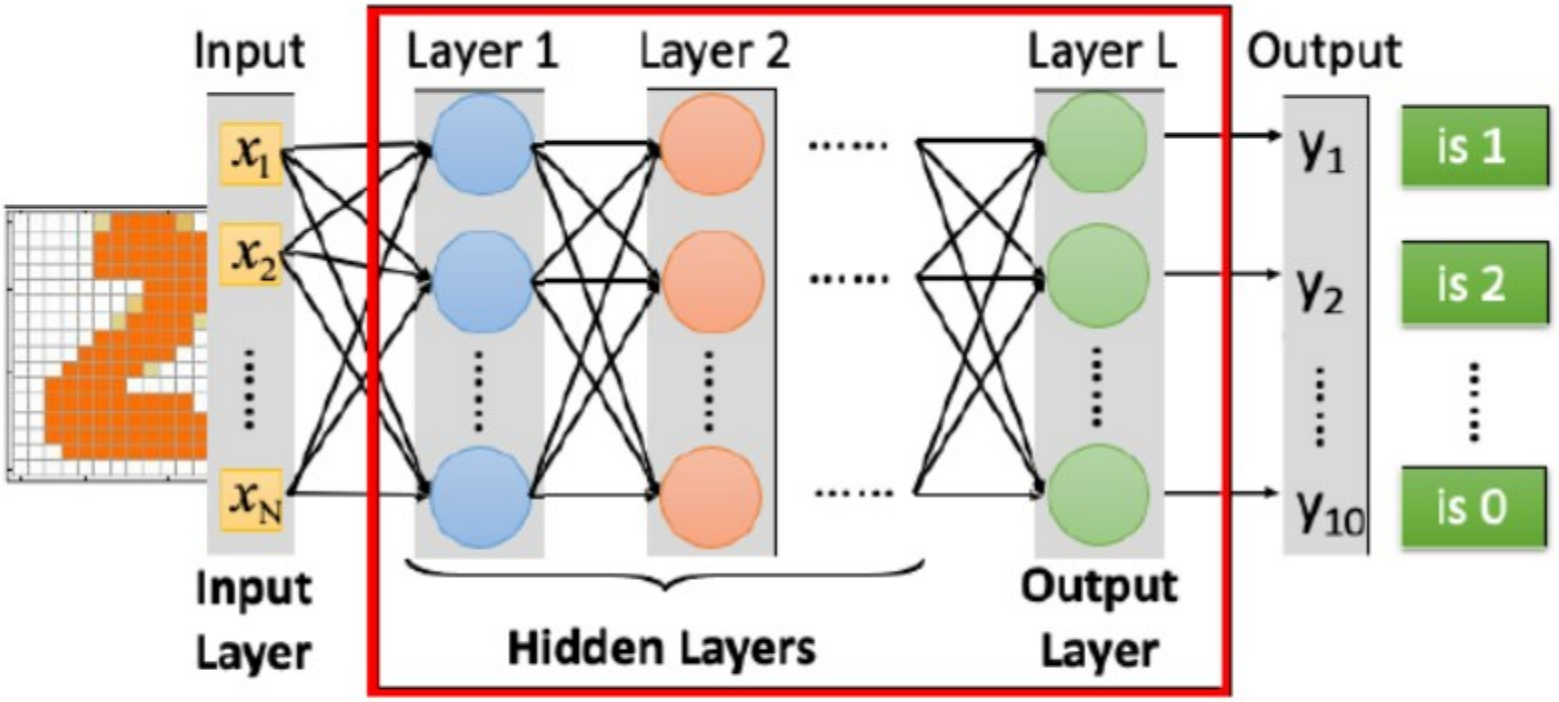




An Example


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

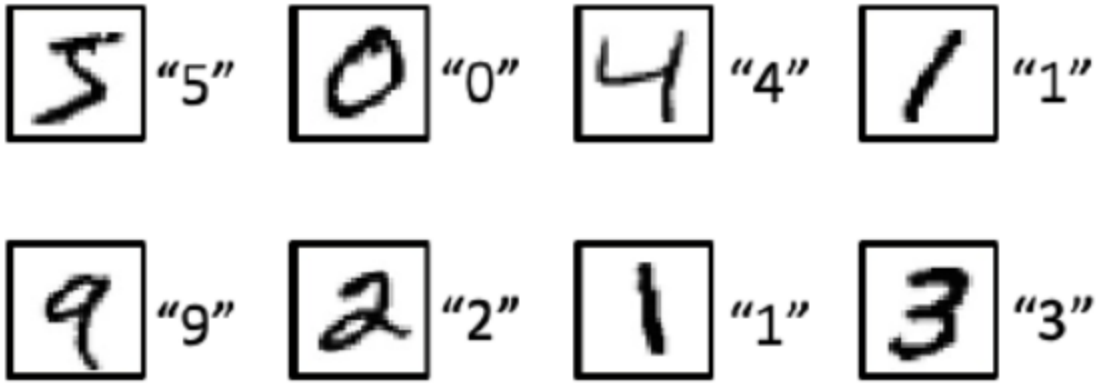
An Example



Needs to determine the network structure

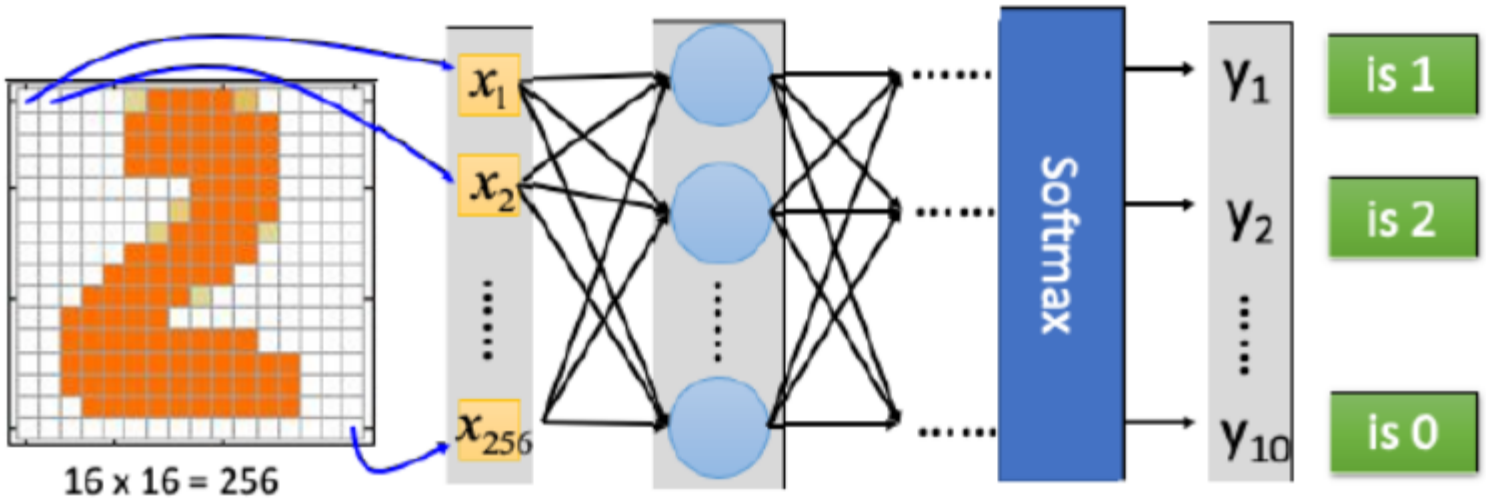
How many layers?
How many neurons for each layer?

Learning Target



The learning target is defined on the training data.

Learning Target



Ink → 1

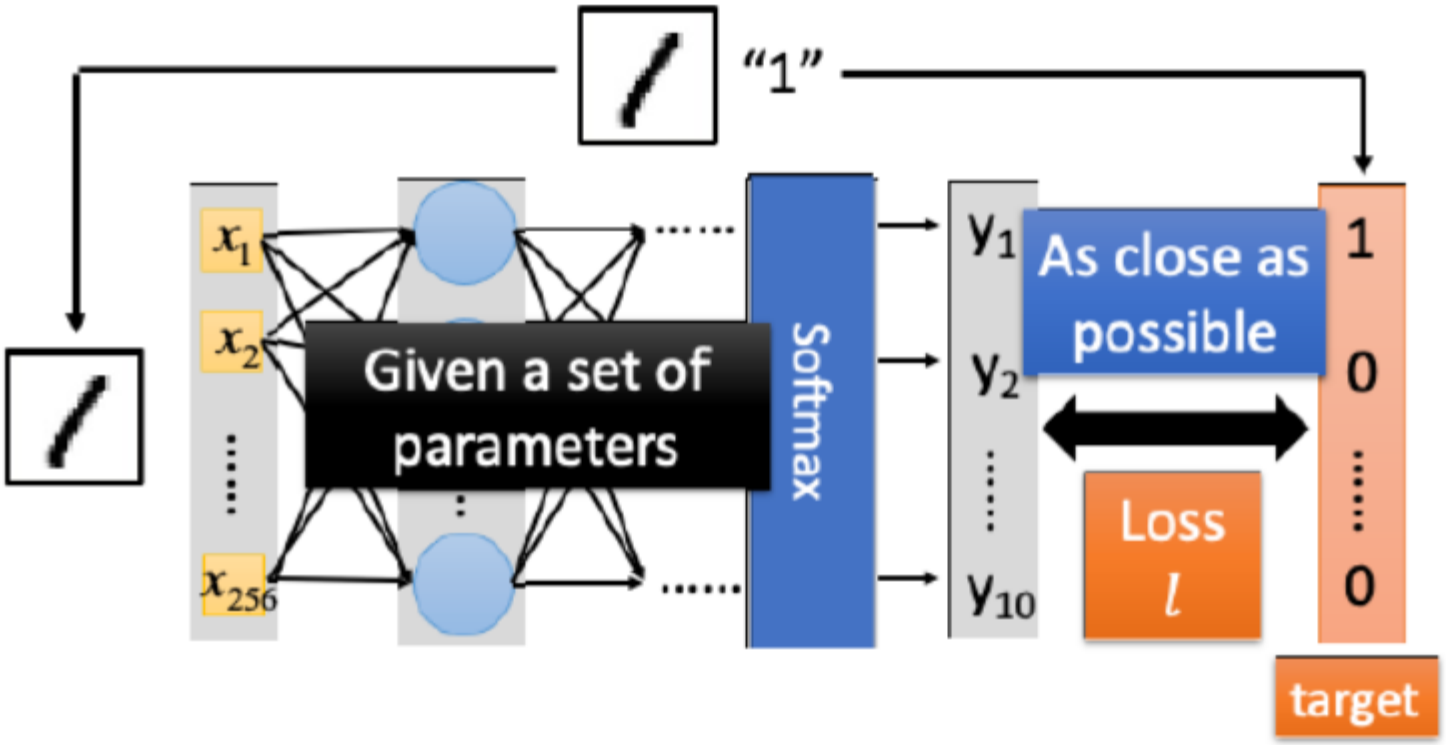
No ink → 0

The learning target is .....

Input:  → y_1 has the maximum value

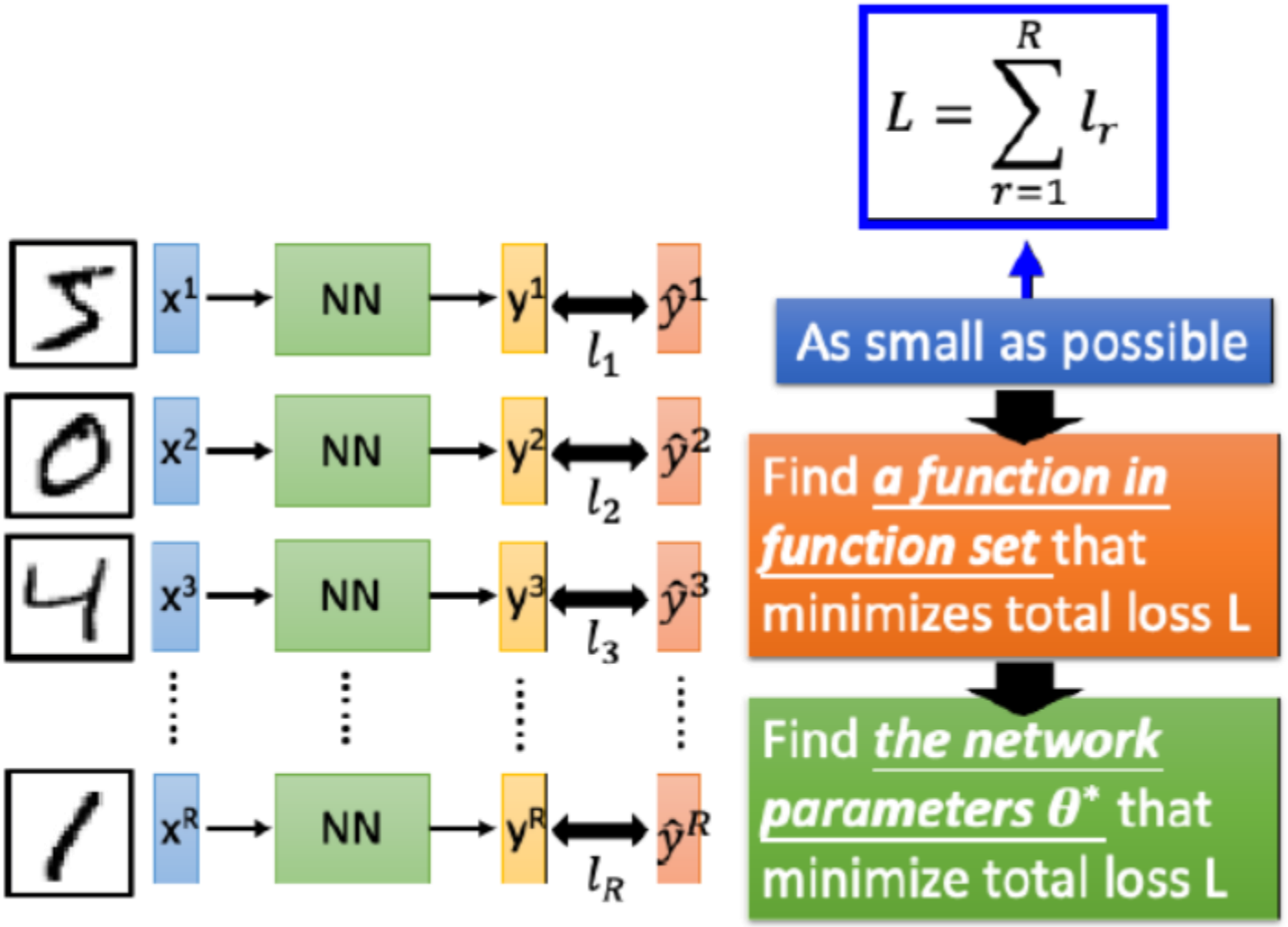
Input:  → y_2 has the maximum value

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
$\hat{y}^1$
l_1
x^2
NN
y^2
$\hat{y}^2$
l_2
x^3
NN
y^3
$\hat{y}^3$
l_3
x^R
NN
y^R
$\hat{y}^R$
l_R

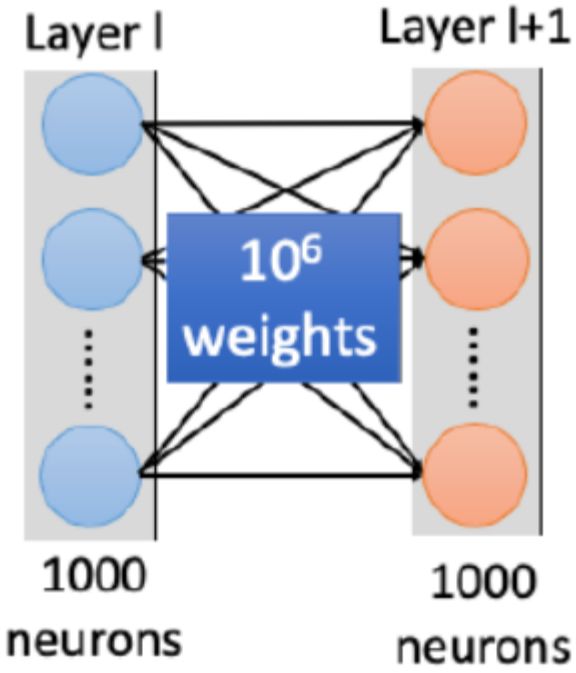
Total Loss

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

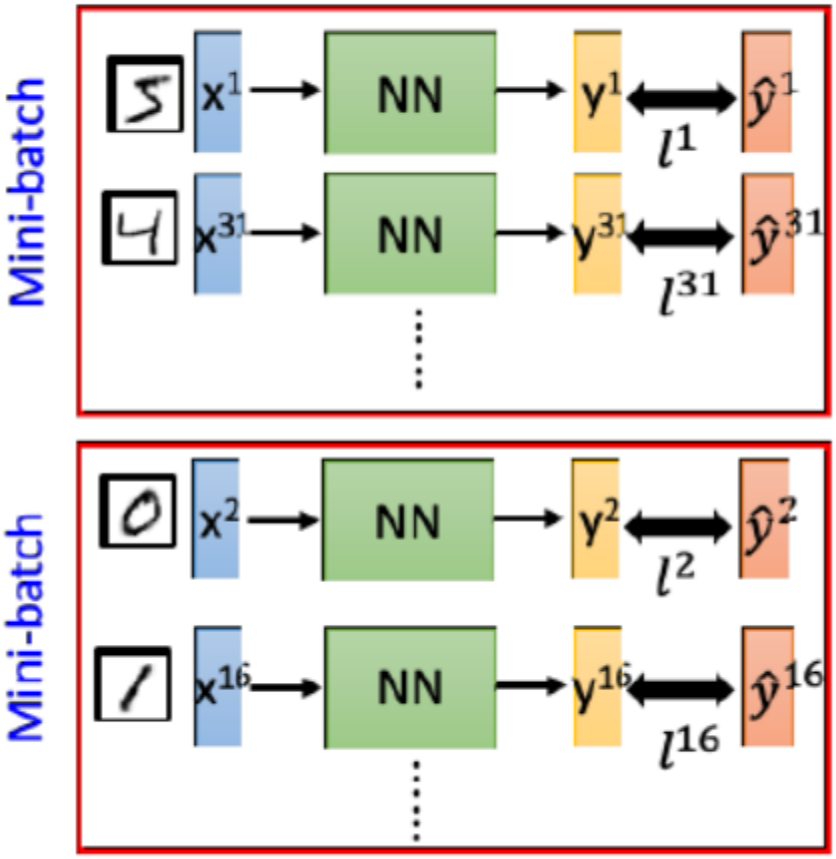
Enumerate all possible values

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

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

  $L'' = l^2 + l^{16} + \cdots$

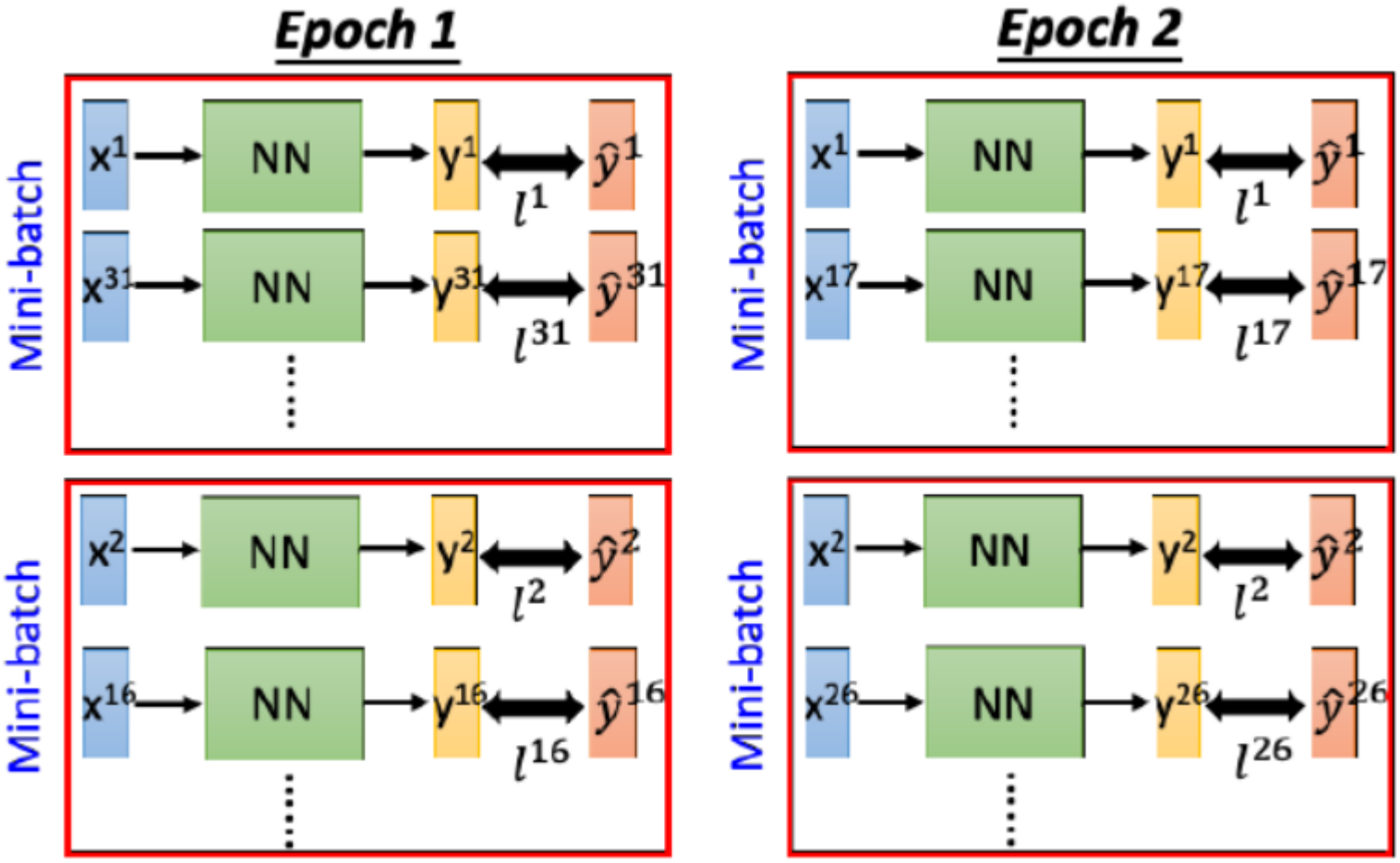
  Update parameters once

  ⋮
- Until all mini-batches have been picked

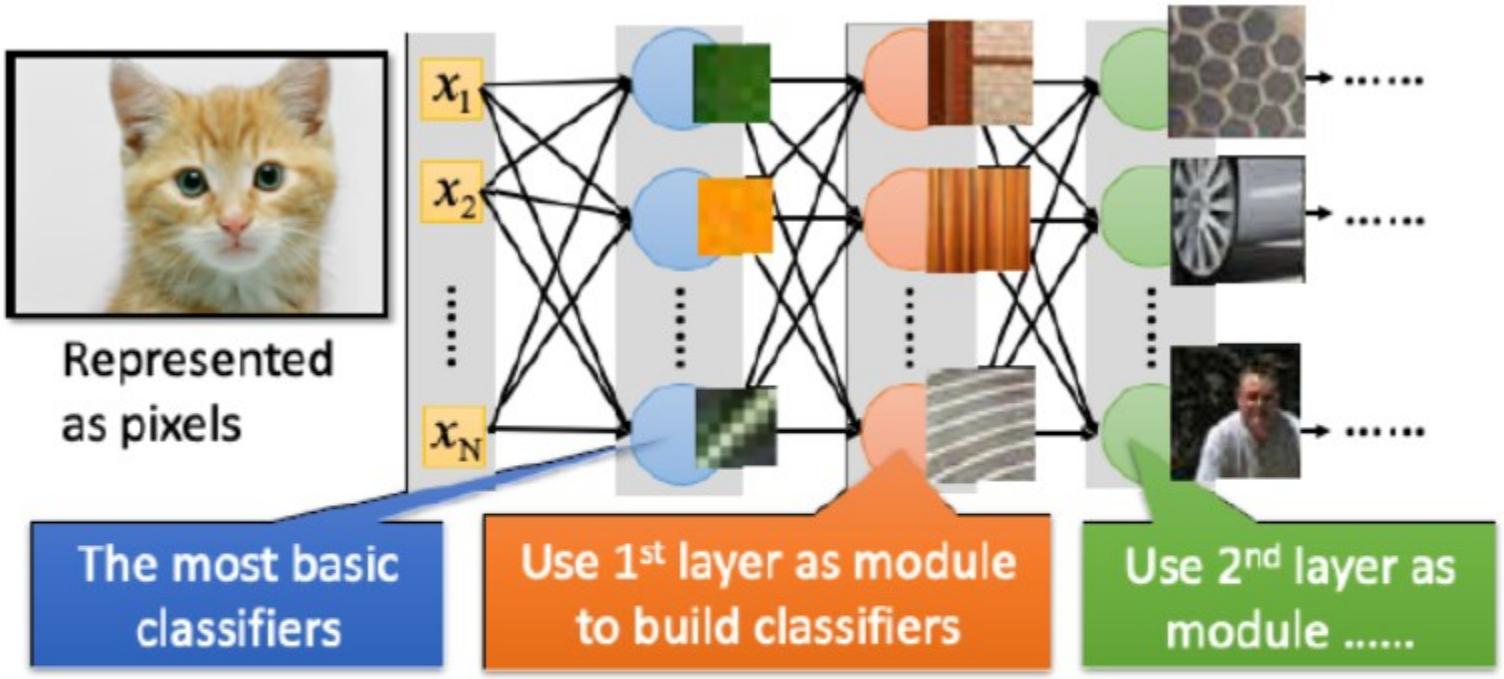
one epoch

Repeat the above process

Mini-batch


Epoch 1
Epoch 2
Mini-batch
Mini-batch
Mini-batch
Mini-batch
NN
NN
NN
NN
NN
NN
NN
NN

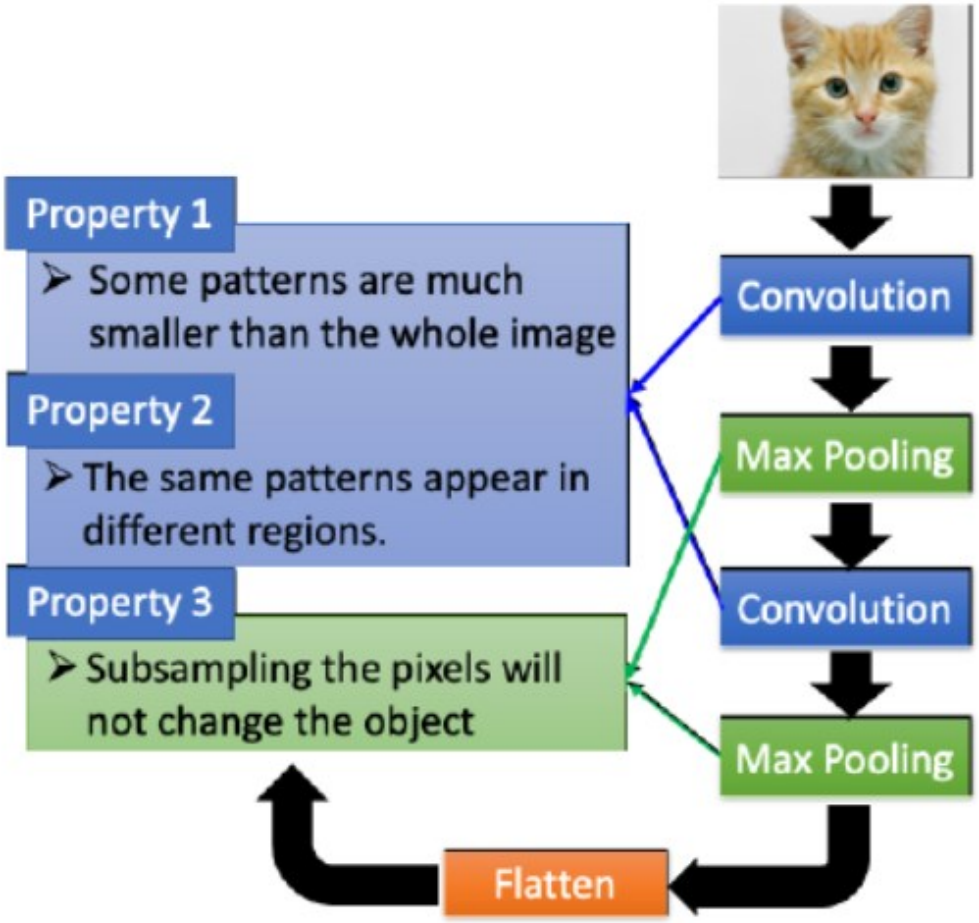
Convolutional Neural Network (CNN)



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

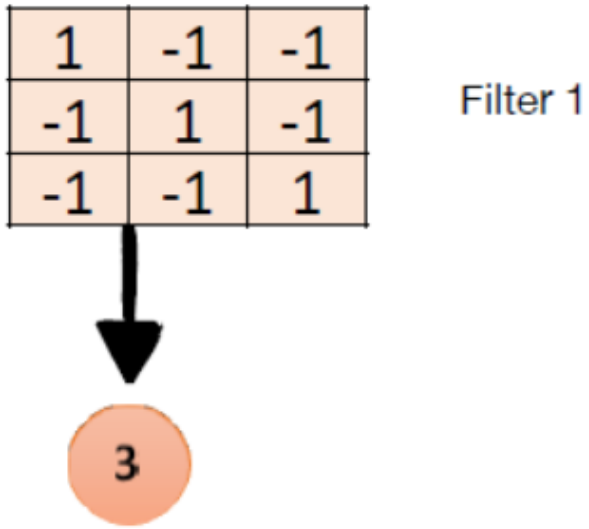
Filter 1
Matrix

…
…
…
…

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

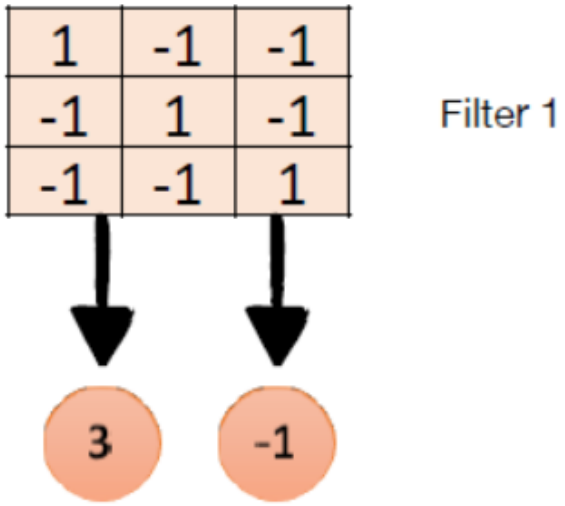


Filter 1

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

-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

3	-1	-3	-1
-3	1	0	-3
-3	-3	0	1
3	-2	-2	-1

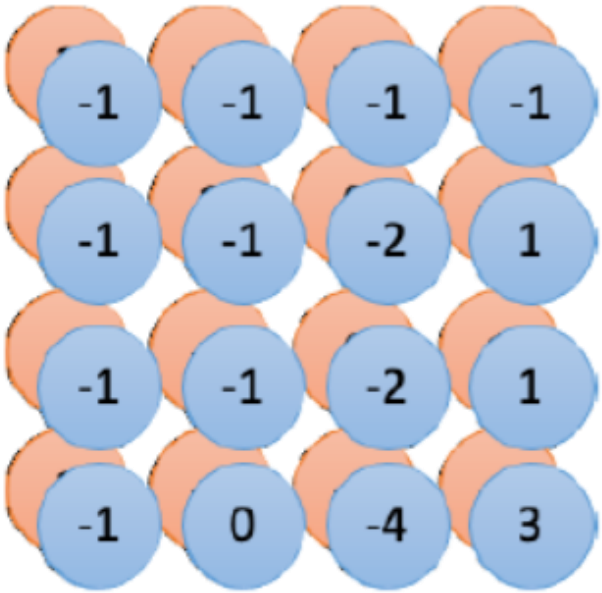
Property 2: Same patterns appears at different region!

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

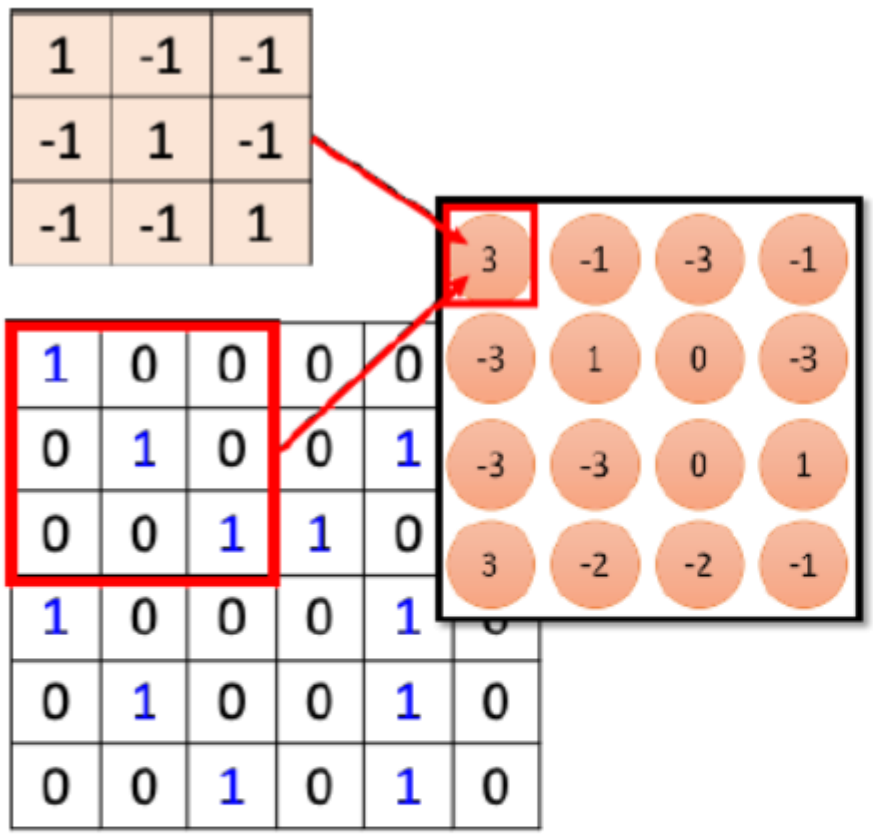
-1	1	-1
-1	1	-1
-1	1	-1

Filter2

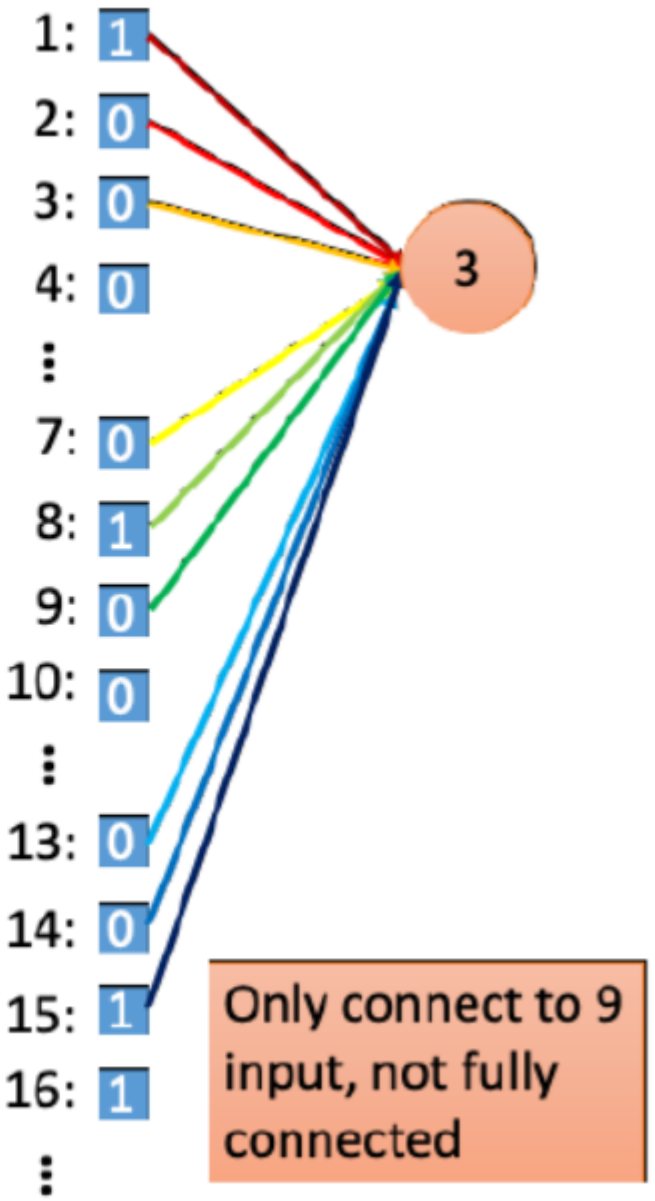


Property 2: Same patterns appears at different region!

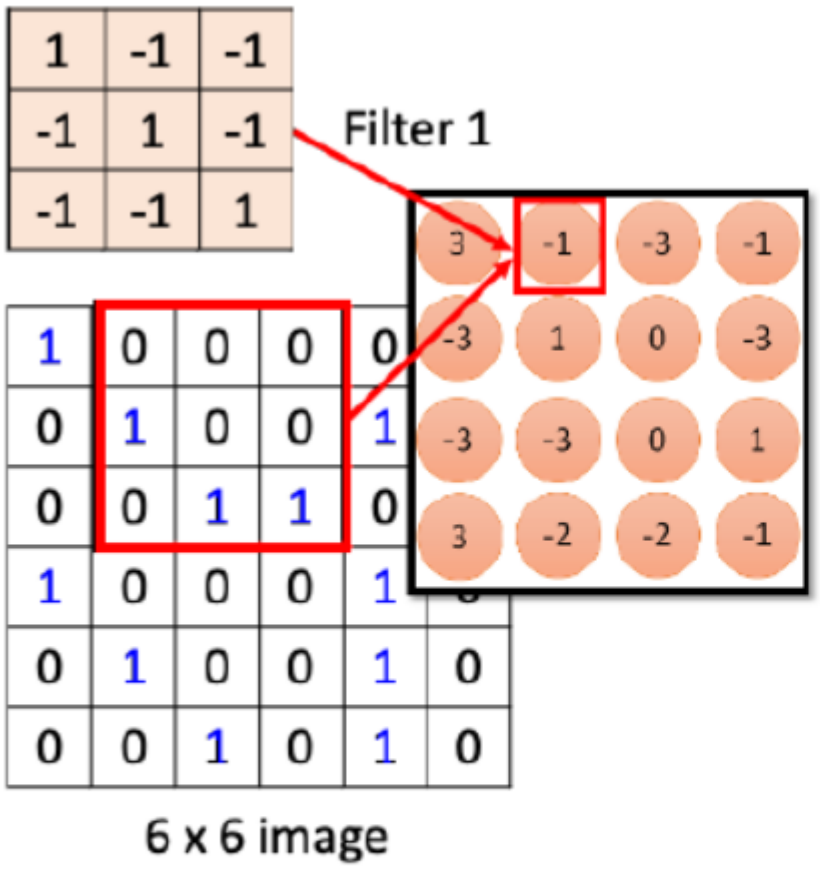
CNN - Convolution

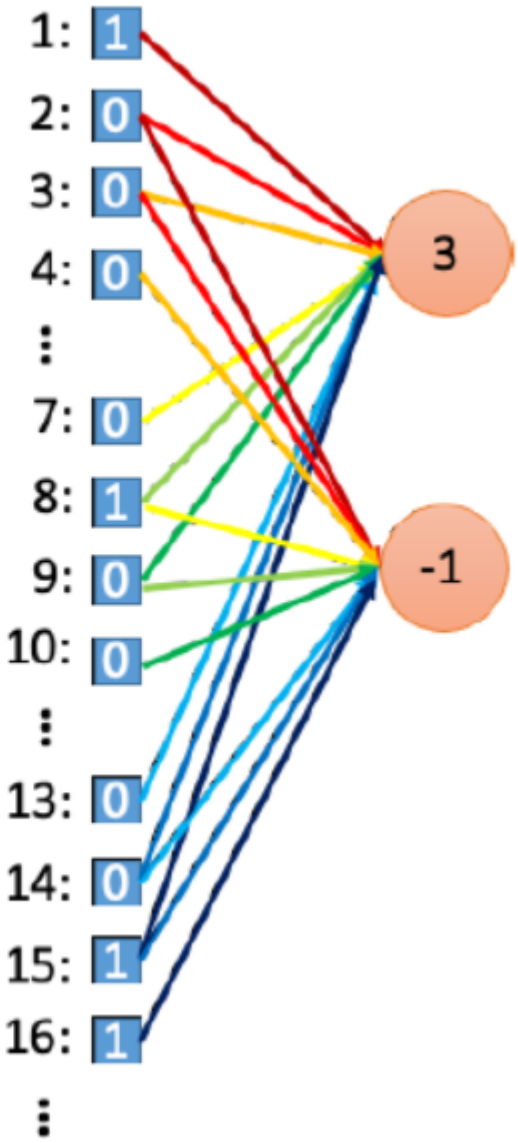


Less parameters!

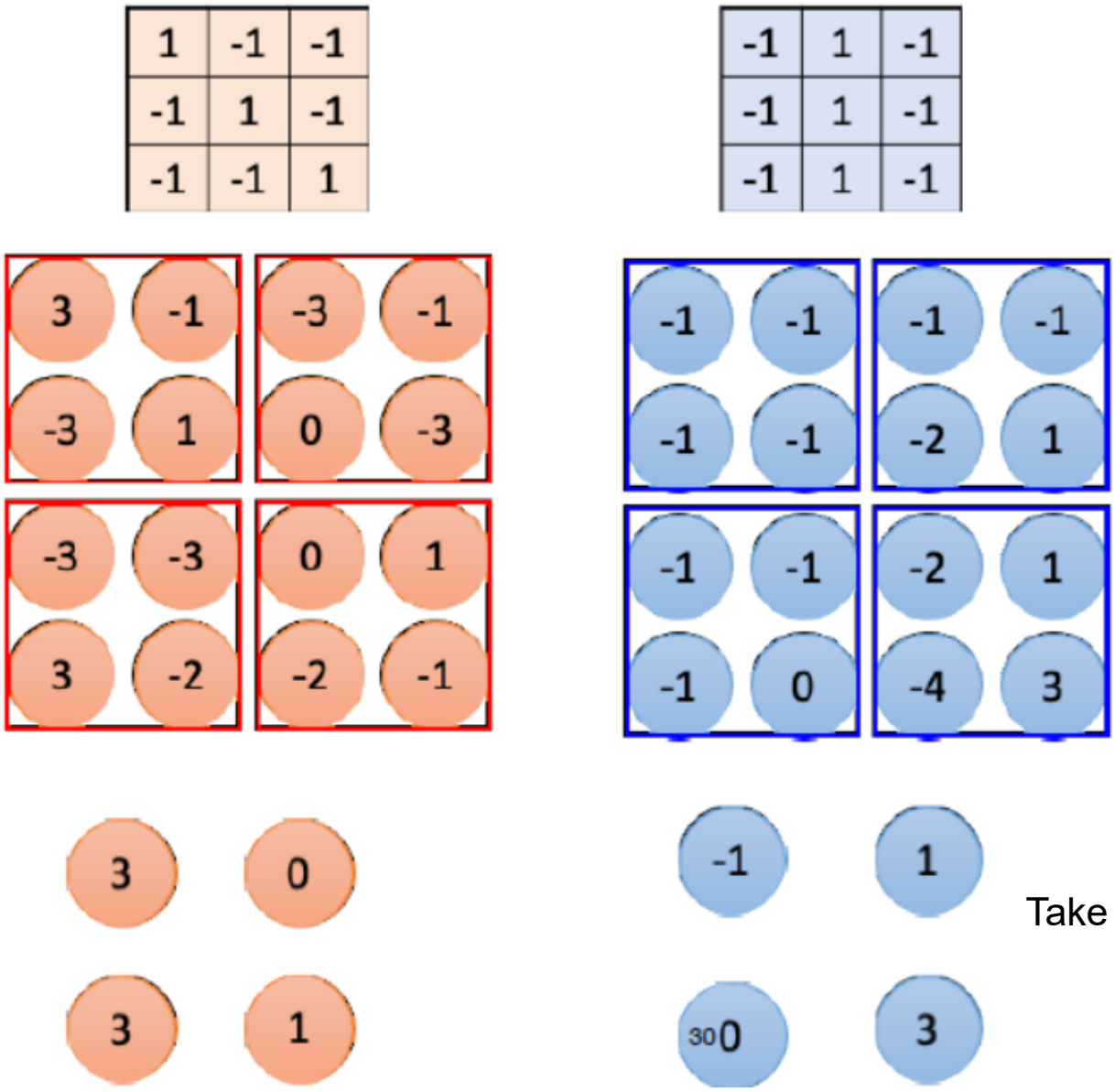


CNN - Convolution



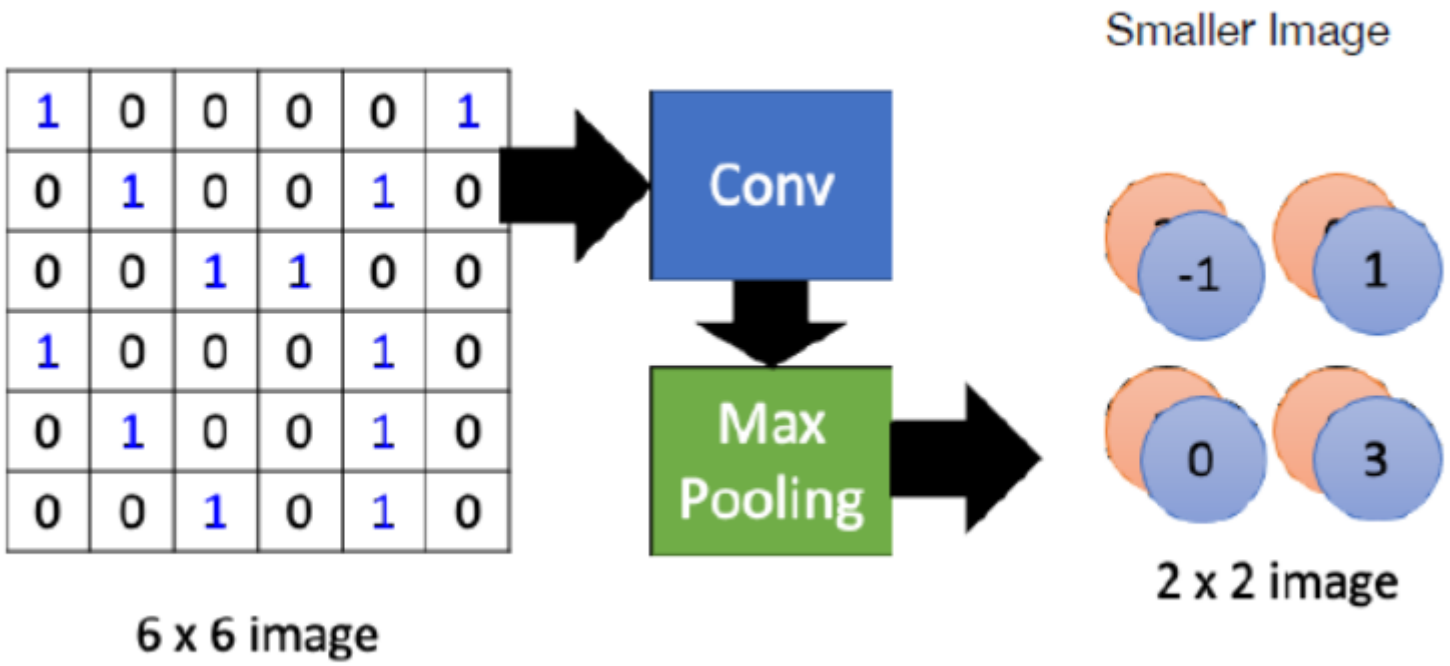


CNN - Max Pooling

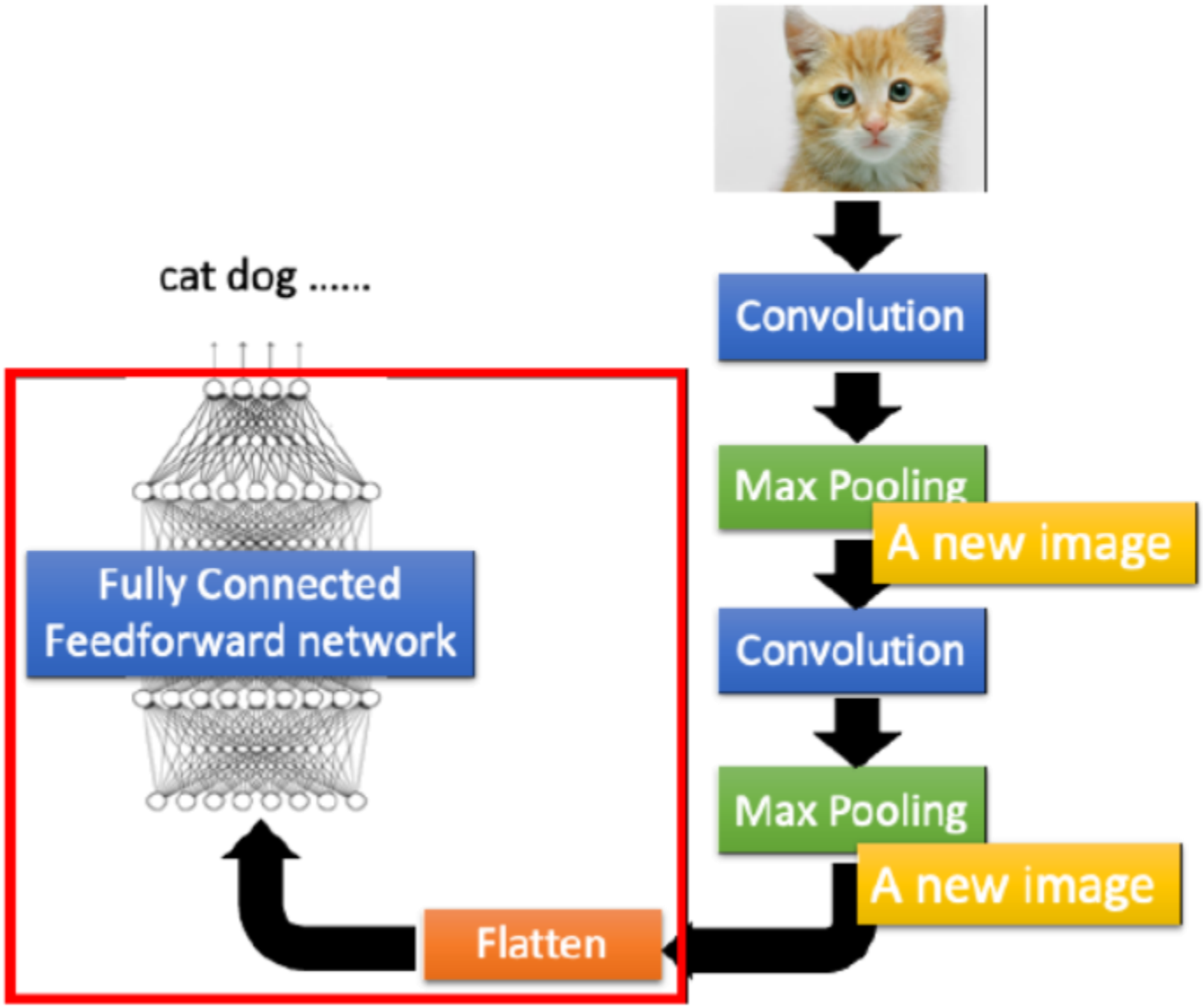


Take the dominant Features Feature Dimensionality Reduction

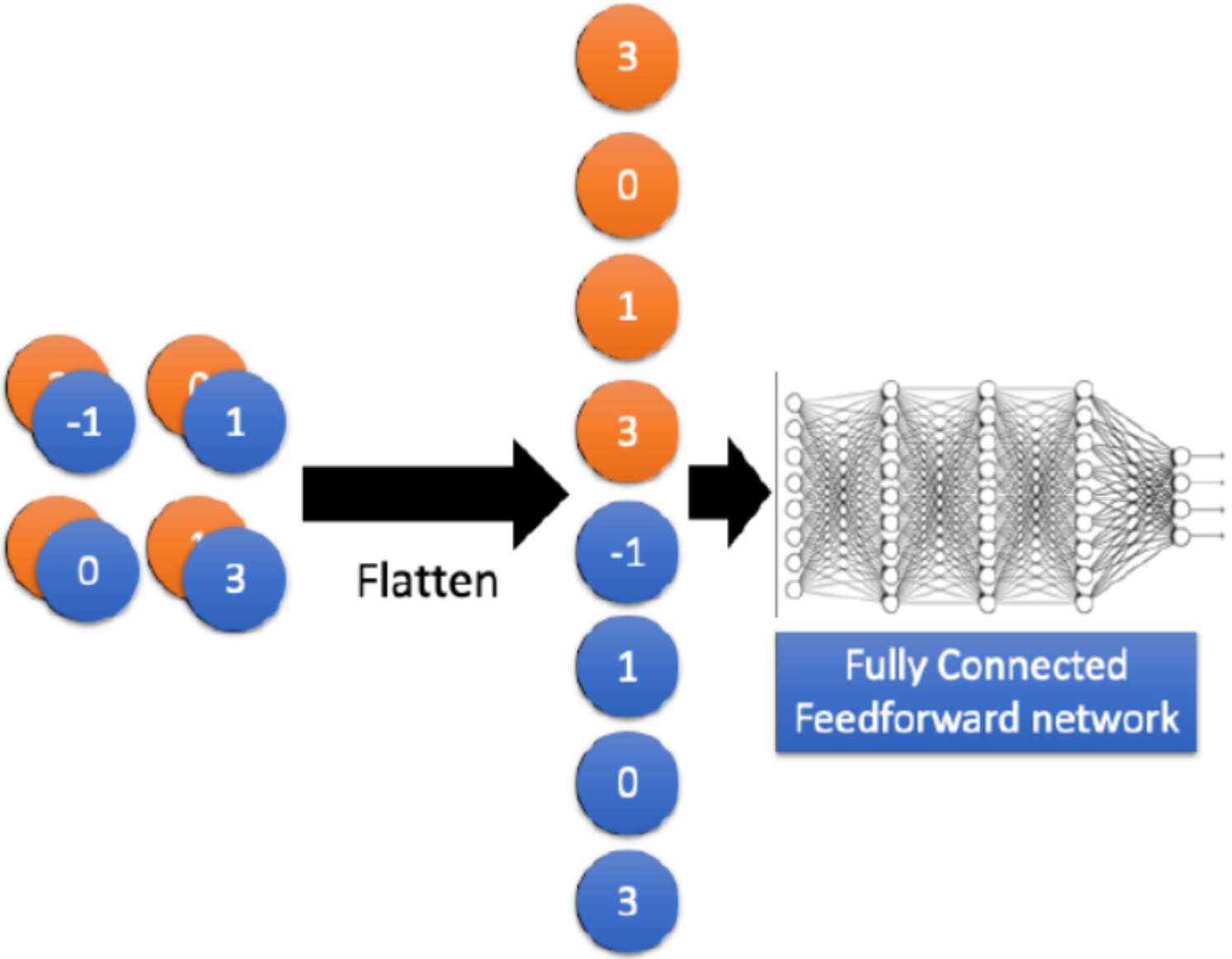
CNN - Convolution



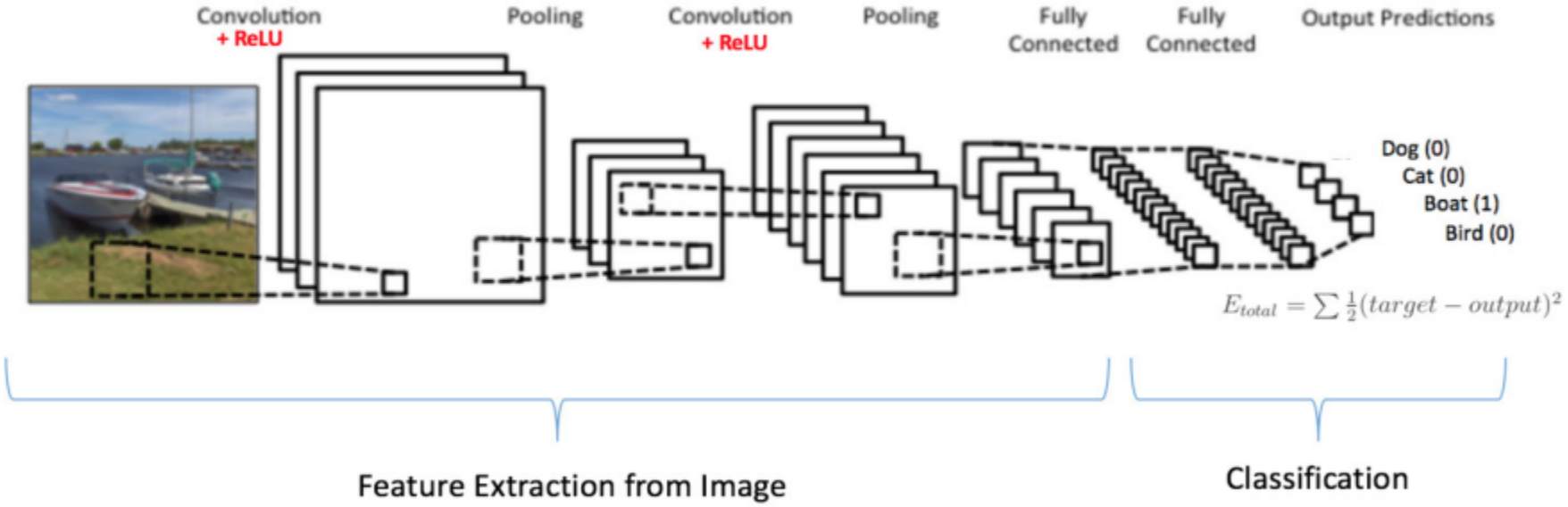
The Whole CNN



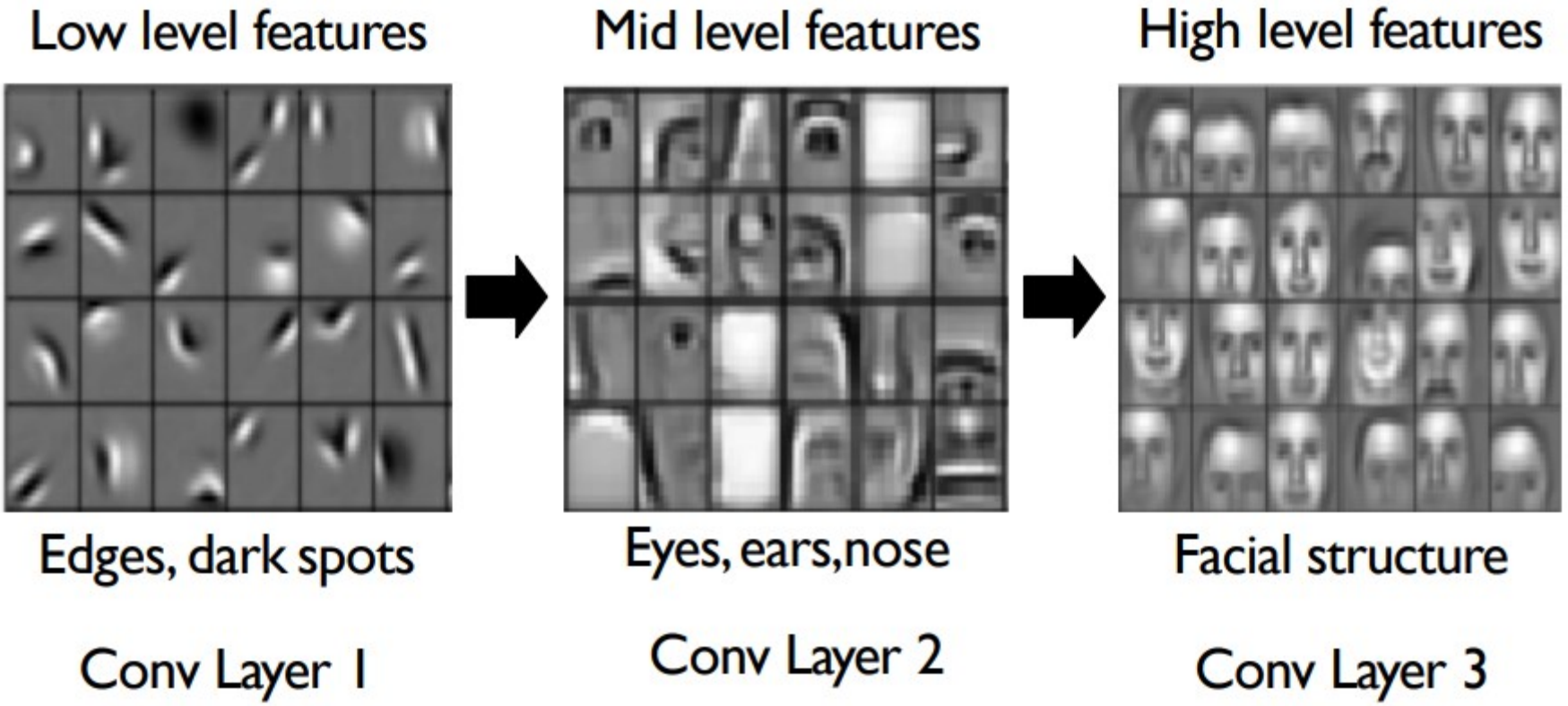
Flatten


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

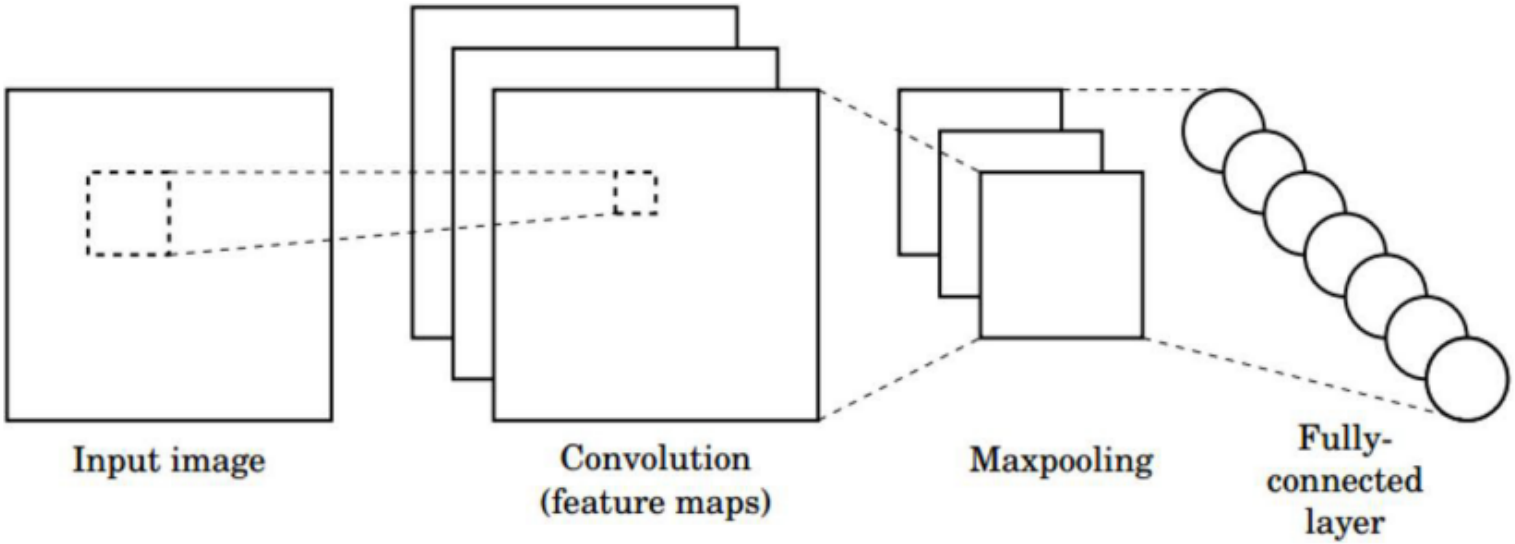
An image classification CNN



Representation Learning in Deep CNNs



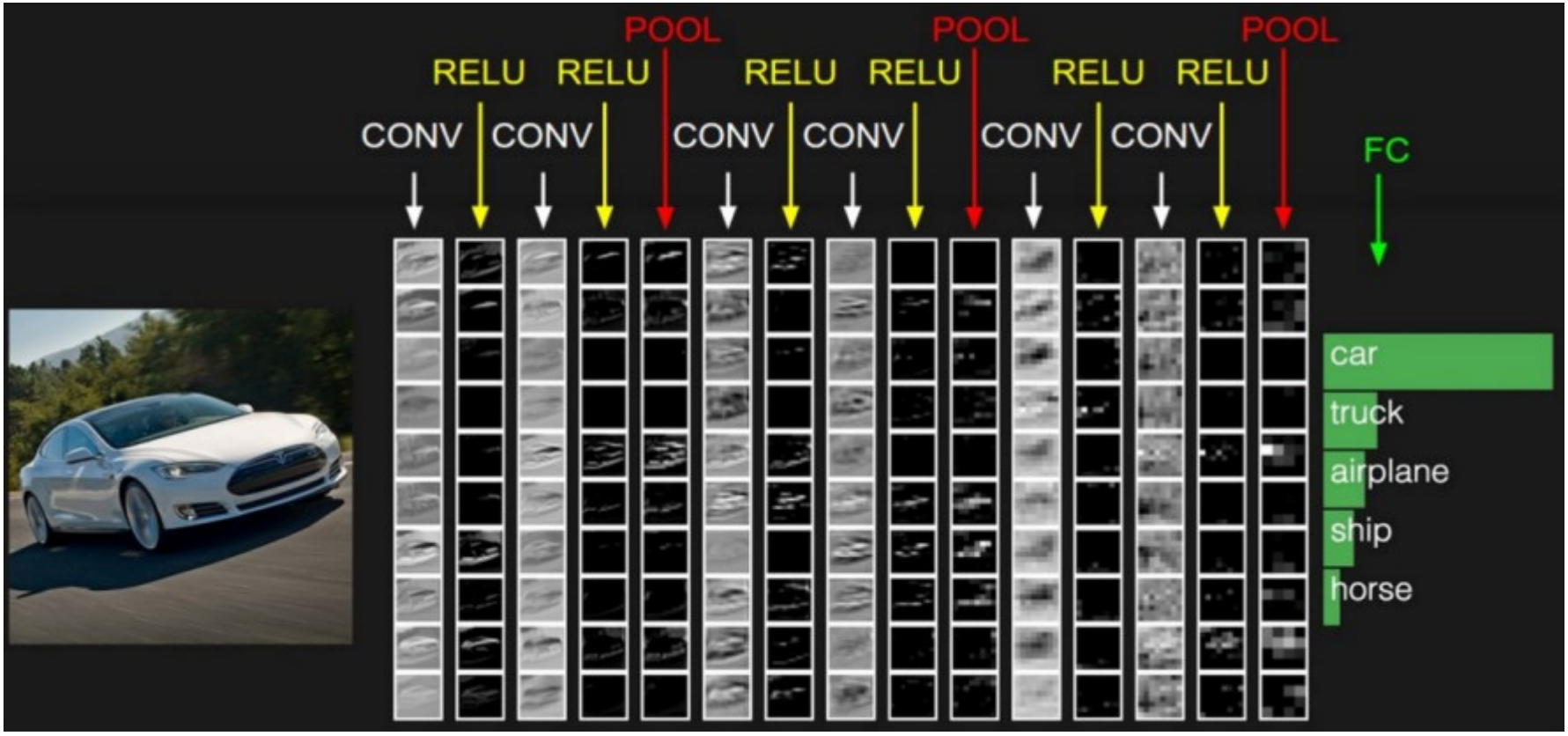
CNNs for Classification



1. **Convolution**: Apply filters to generate feature maps.
2. **Non-linearity**: Often ReLU.
3. **Pooling**: Downsampling operation on each feature map.

Train model with image data.
Learn weights of filters in convolutional layers.

Example – Six convolutional layers



Hands-On Example

- Python provides a set of libraries including different deep learning packets
- Standard libraries provide the ready-to-use implementation of algorithm
- **Tensorflow** is the one we will use in this lecture

Hands-On Example

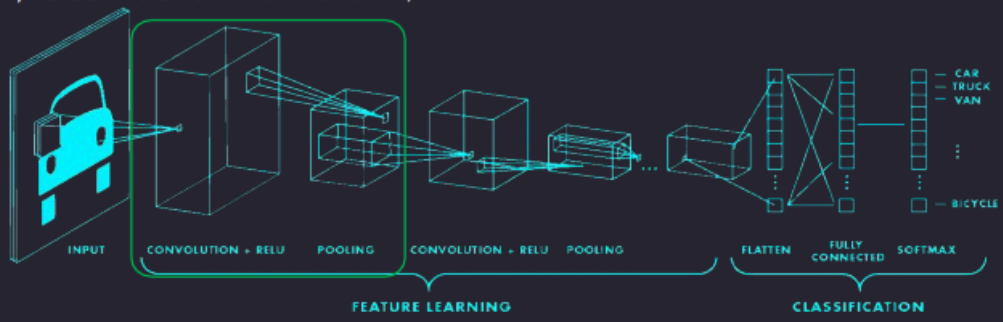
```
import tensorflow as tf

def generate_model():
    model = tf.keras.Sequential([
        # first convolutional layer
        tf.keras.layers.Conv2D(32, filter_size=3, activation='relu'),
        tf.keras.layers.MaxPool2D(pool_size=2, strides=2),

        # second convolutional layer
        tf.keras.layers.Conv2D(64, filter_size=3, activation='relu'),
        tf.keras.layers.MaxPool2D(pool_size=2, strides=2),

        # fully connected classifier
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(1024, activation='relu'),
        tf.keras.layers.Dense(10, activation='softmax')
        # 10 outputs

    ])
    return model
```



Q&A

The only stupid question is the one you were afraid to ask but never did.

-Rich Sutton