Machine Learning

Deep Learning Introduction

Python tutorial: http://learnpython.org/

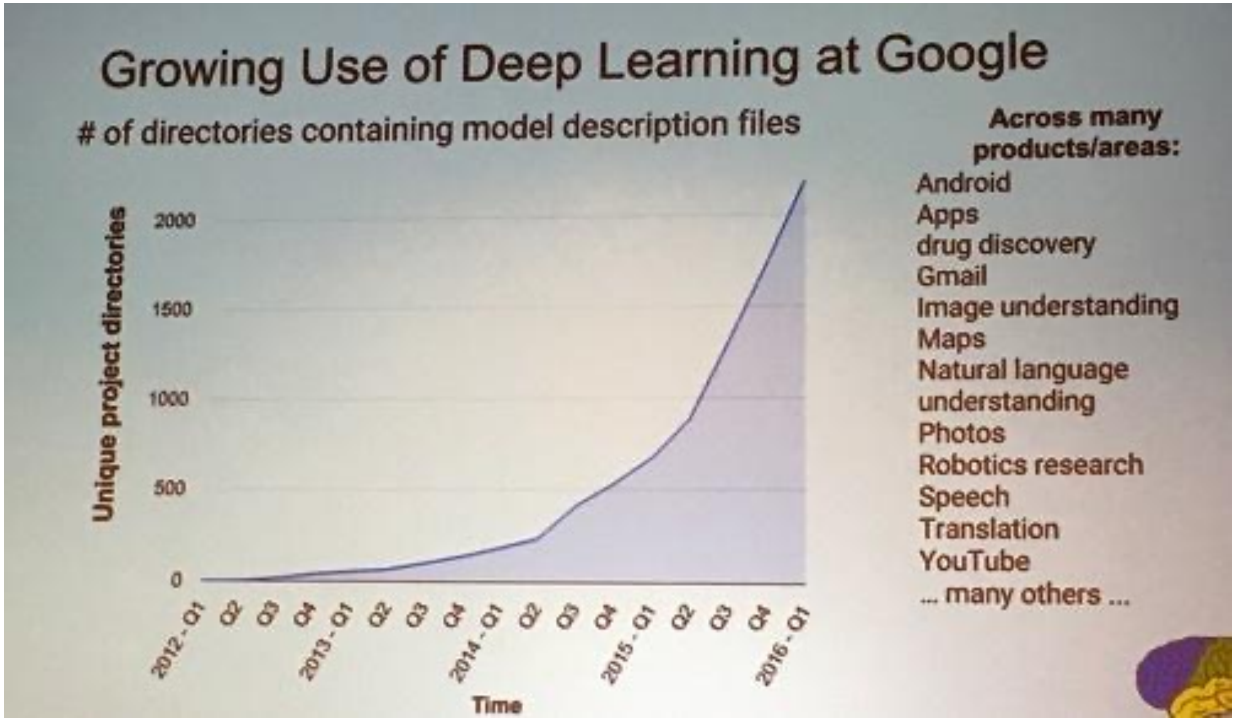
TensorFlow tutorial: https://www.tensorflow.org/tutorials/

PyTorch tutorial: https://pytorch.org/tutorials/

Acknowledge: The slides are partially referred to coursera online machine learning course by Prof. Andrew Ng, BYU Data Mining Lab, CS-478-Machine learning, and NYU machine learning course. All copyrights owned by original authors. (i.e. http://u.cs.biu.ac.il/~89-570/)

Deep learning attracts lots of attention.

- I believe you have seen lots of exciting results before.



Growing Use of Deep Learning at Google

# of directories containing model description files

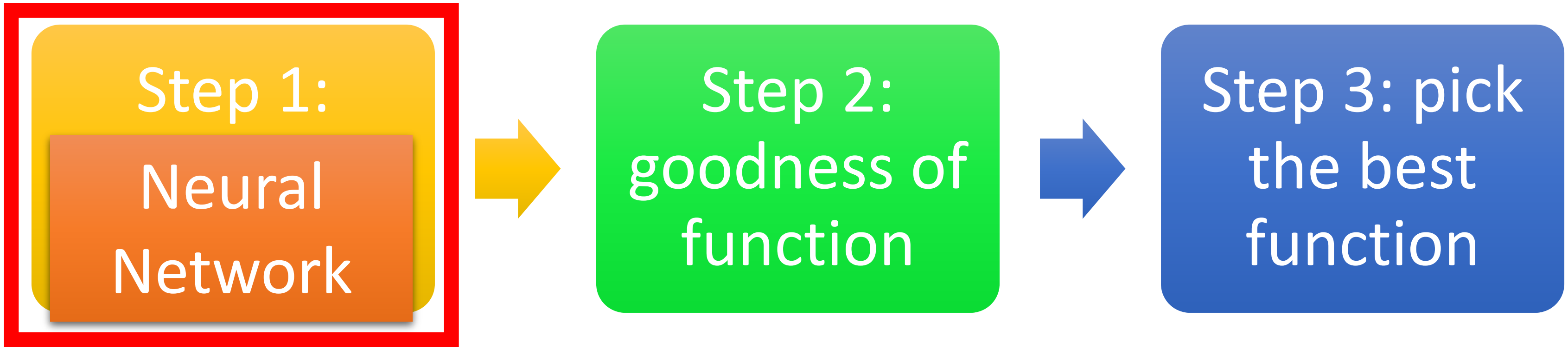
Across many products/areas:
Android
Apps
drug discovery
Gmail
Image understanding
Maps
Natural language understanding
Photos
Robotics research
Speech
Translation
YouTube
... many others ...

Deep learning trends at Google. Source: SIGMOD 2016/Jeff Dean

Ups and downs of Deep Learning

- 1958: Perceptron (linear model)
- 1969: Perceptron has limitation
- 1980s: Multi-layer perceptron
  - Do not have significant difference from DNN today
- 1986: Backpropagation
  - Usually more than 3 hidden layers is not helpful
- 1989: 1 hidden layer is “good enough”, why deep?
- 2006: RBM initialization
- 2009: GPU
- 2011: Start to be popular in speech recognition
- 2012: win ILSVRC image competition
- 2015.2: Image recognition surpassing human-level performance
- 2016.3: Alpha GO beats Lee Sedol
- 2016.10: Speech recognition system as good as humans

Three Steps for Deep Learning



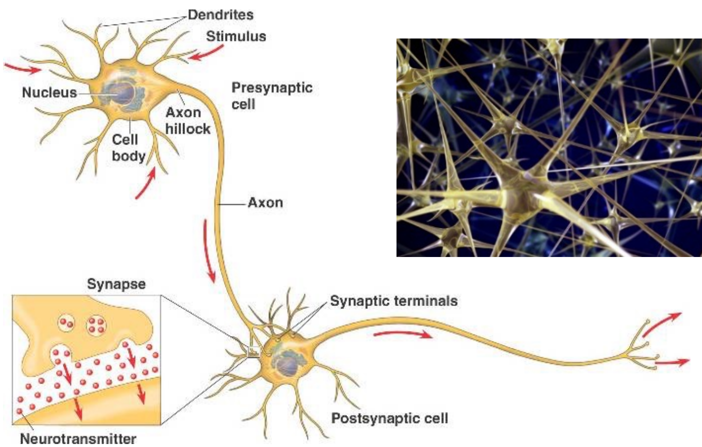
Step 1: Neural Network

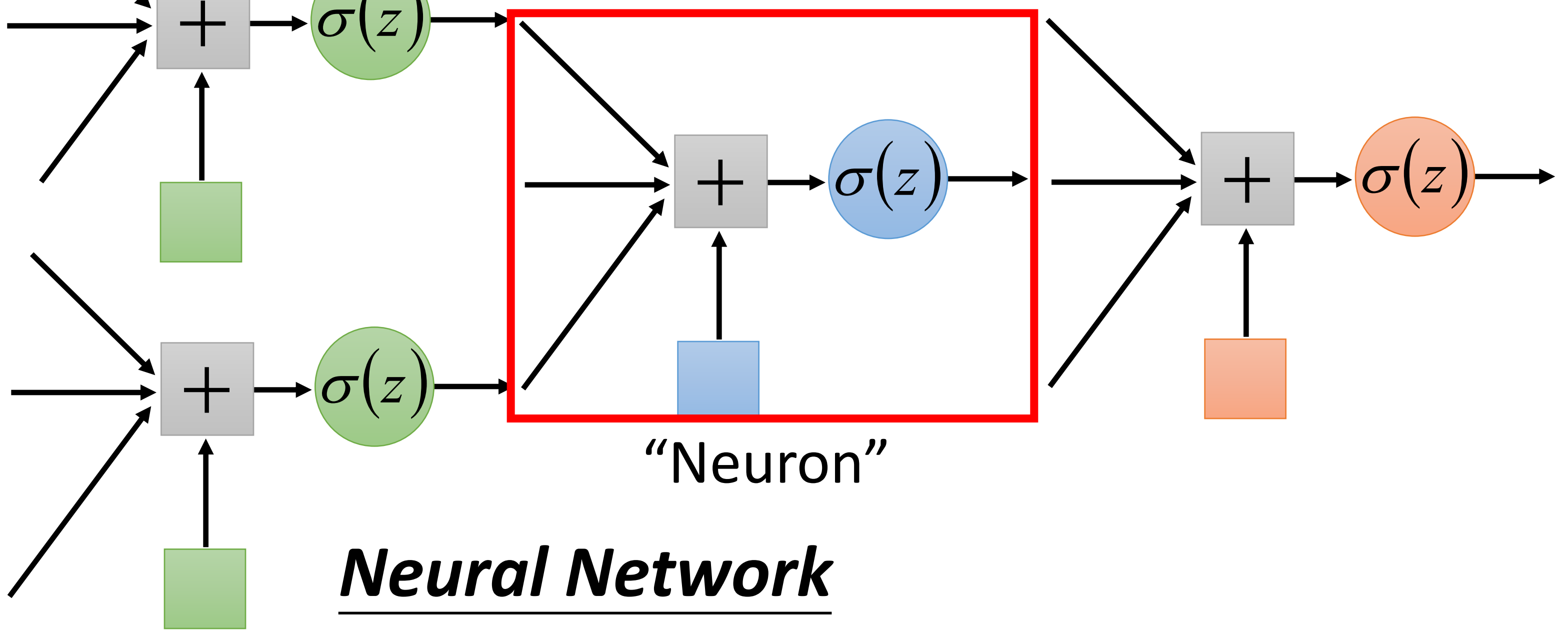
Step 2: goodness of function

Step 3: pick the best function

Deep Learning is so simple ……

Neural Network





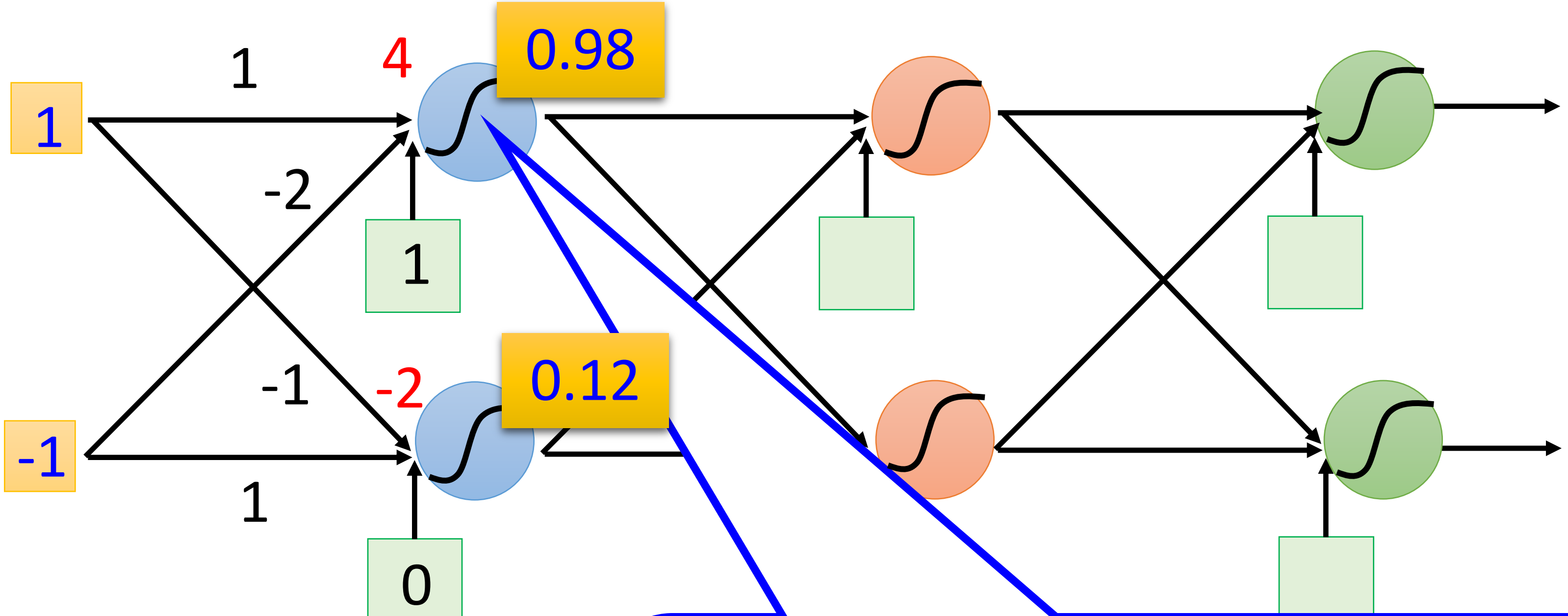
"Neuron"

Neural Network

Different connection leads to different network structures

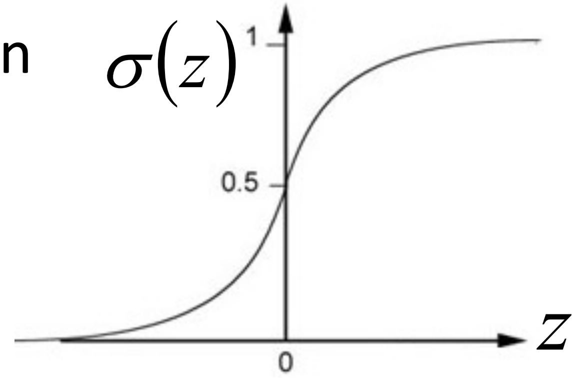
Network parameter θ: all the weights and biases in the "neurons"

Fully Connect Feedforward Network

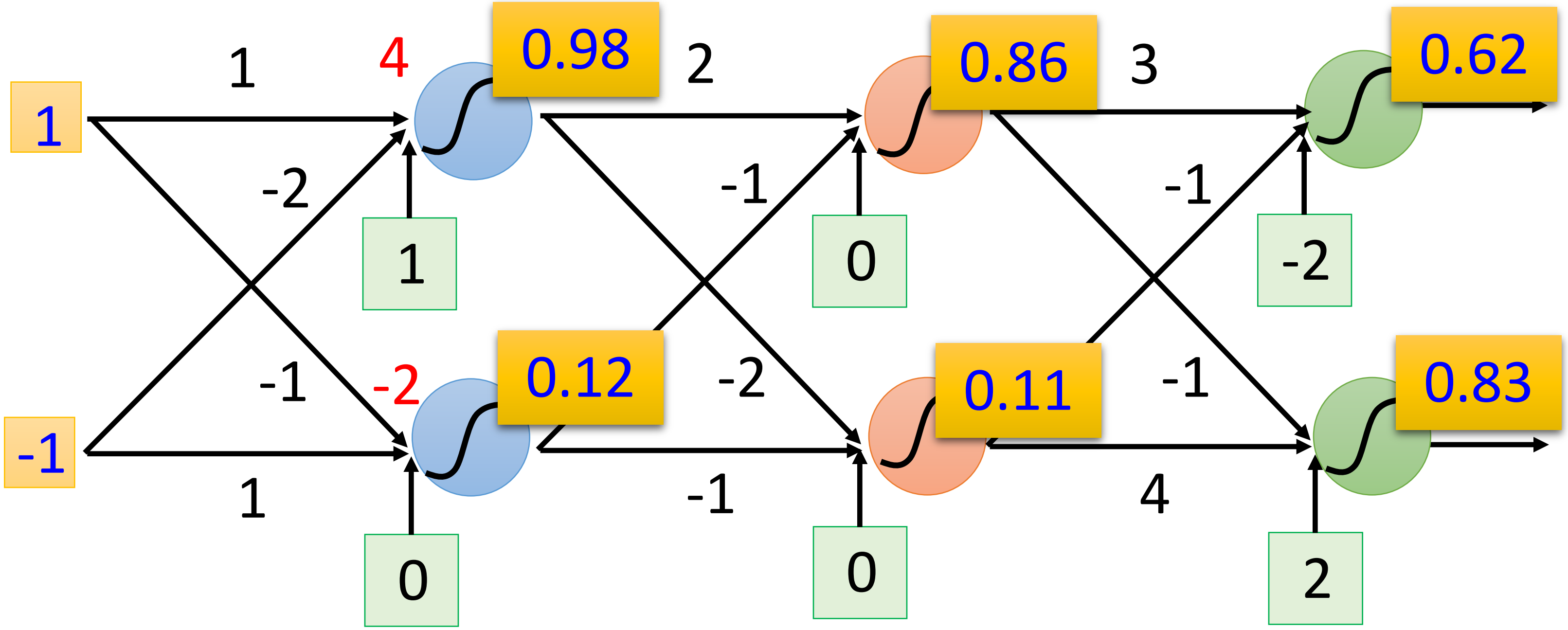


Sigmoid Function

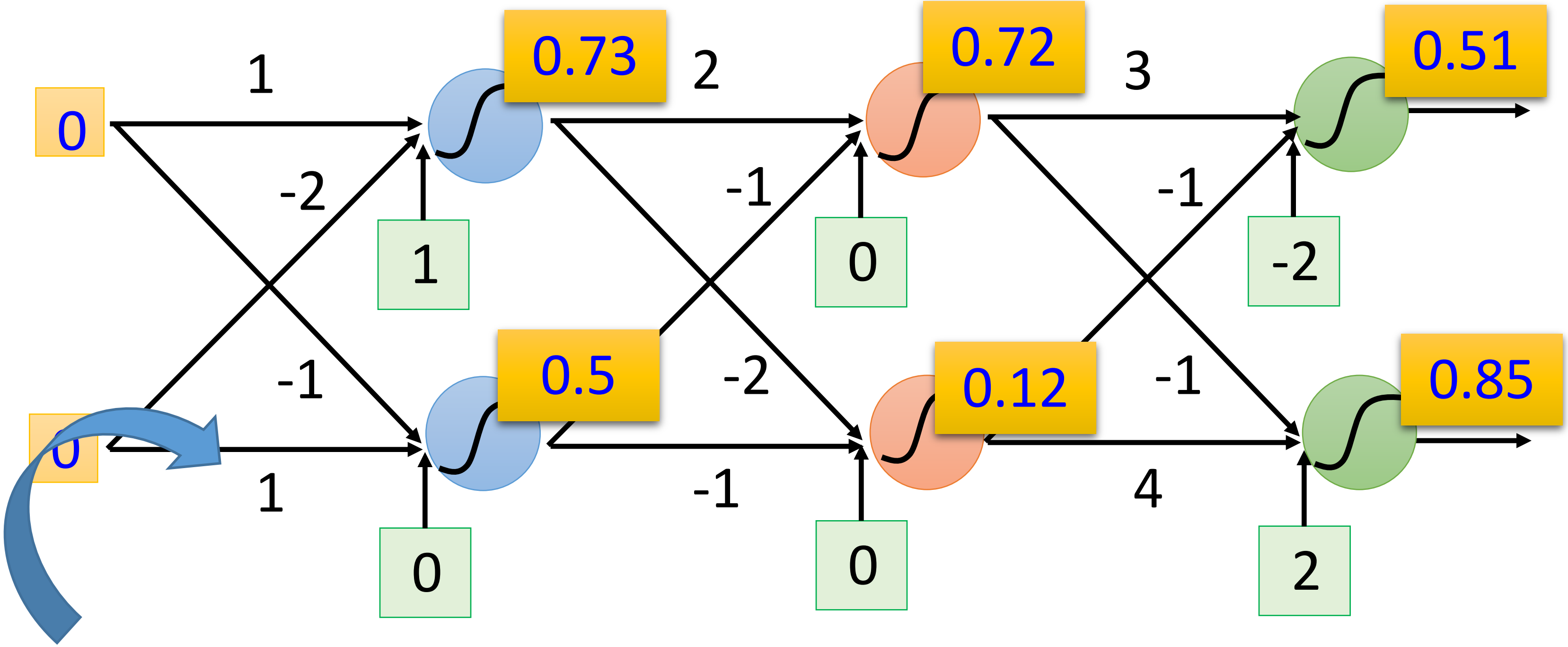
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$



Fully Connect Feedforward Network



Fully Connect Feedforward Network



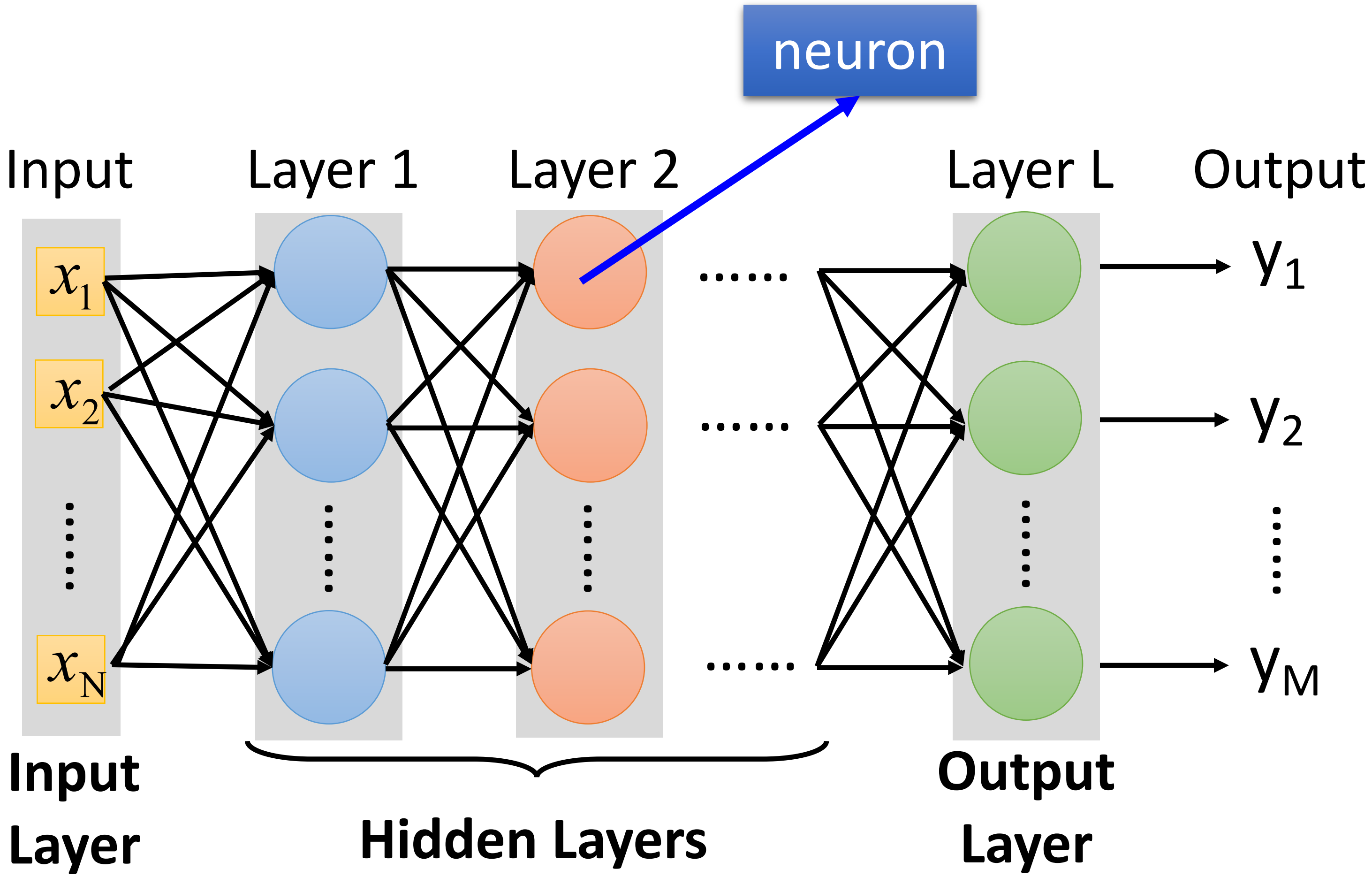
This is a function.

Input vector, output vector

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

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

Fully Connect Feedforward Network

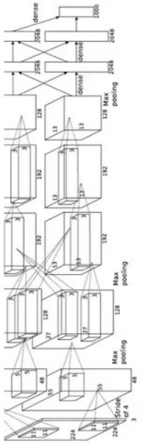


Deep = Many hidden layers

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

8 layers

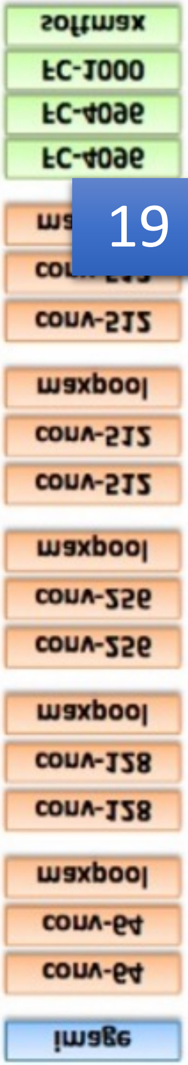
16.4%



AlexNet (2012)

19 layers

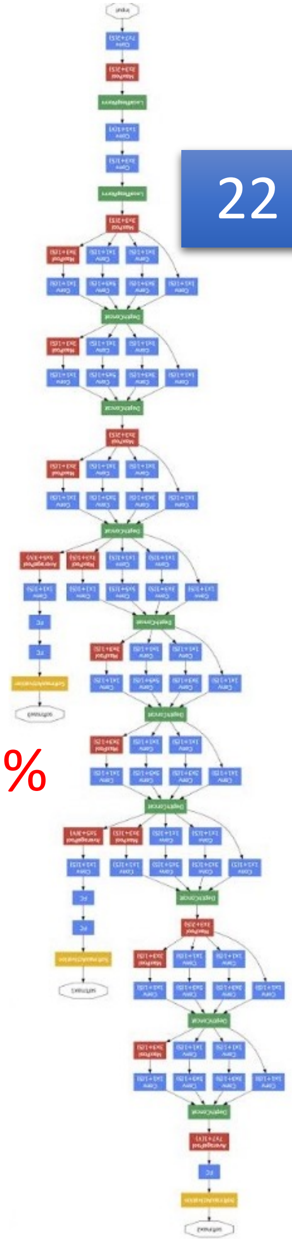
7.3%



VGG (2014)

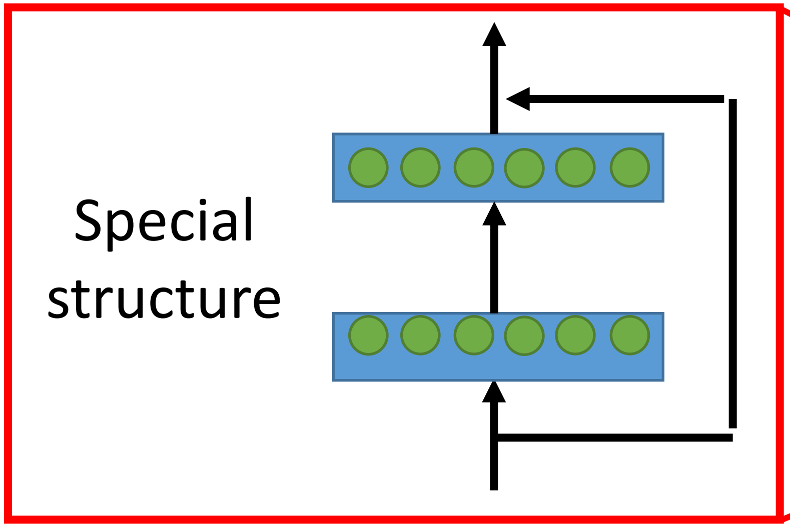
22 layers

6.7%



GoogleNet (2014)

Deep = Many hidden layers



Ref: https://www.youtube.com/watch?v=dxB6299gpvI

16.4%

AlexNet (2012)

7.3%

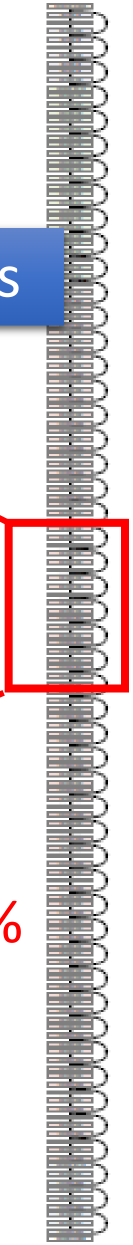
VGG (2014)

6.7%

GoogleNet (2014)

152 layers

3.57%

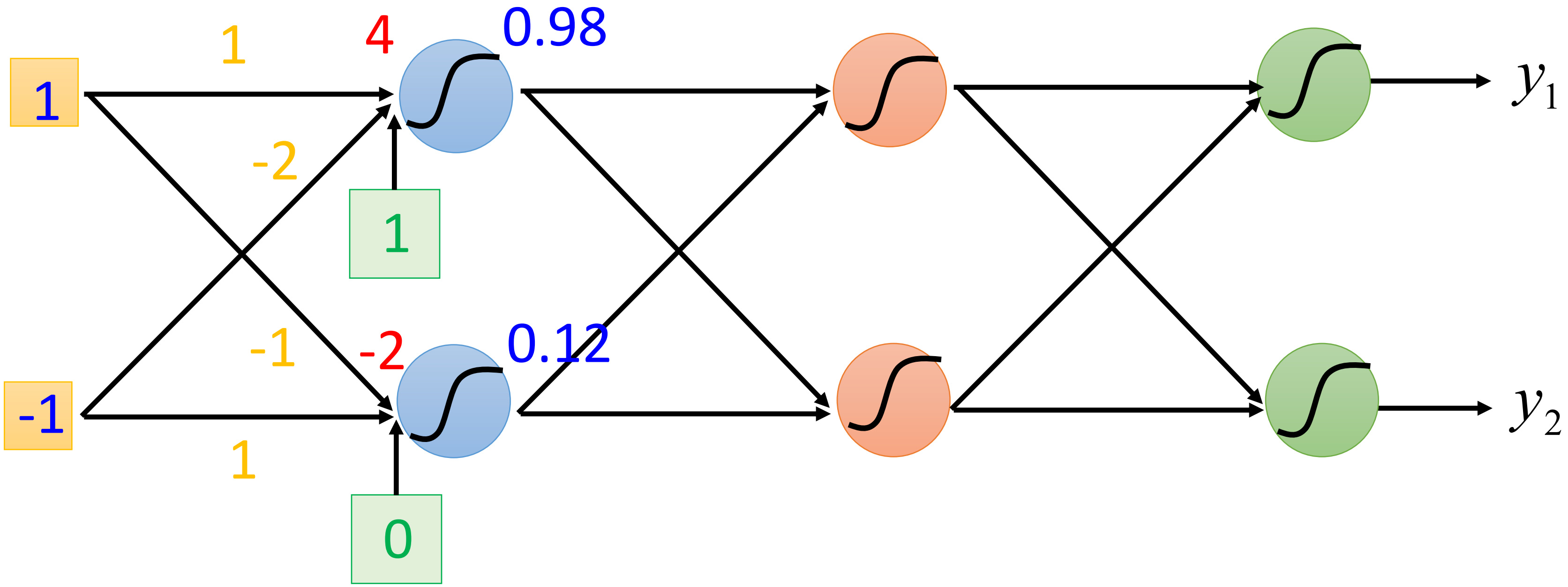


Residual Net (2015)

101 layers

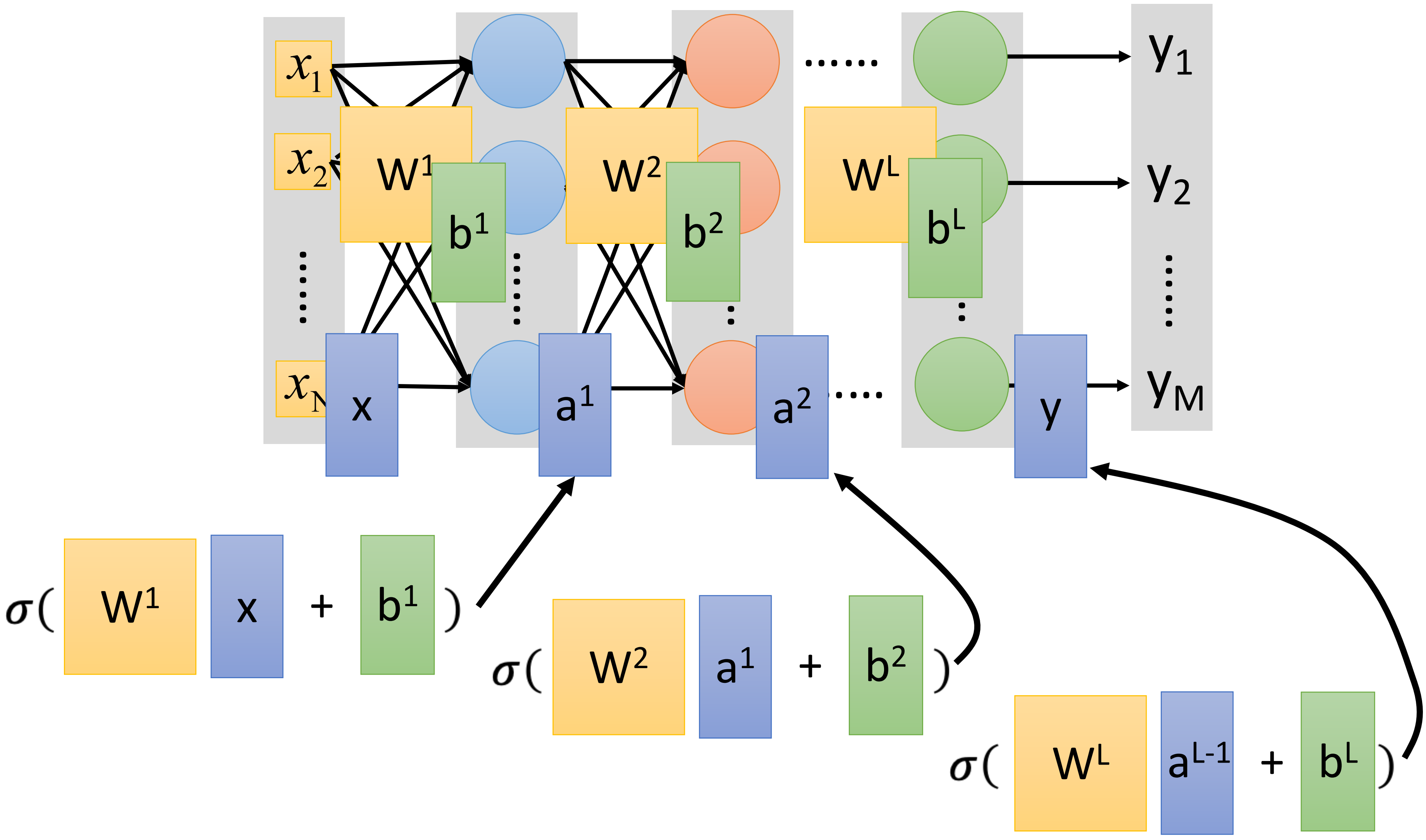
Taipei 101

Matrix Operation

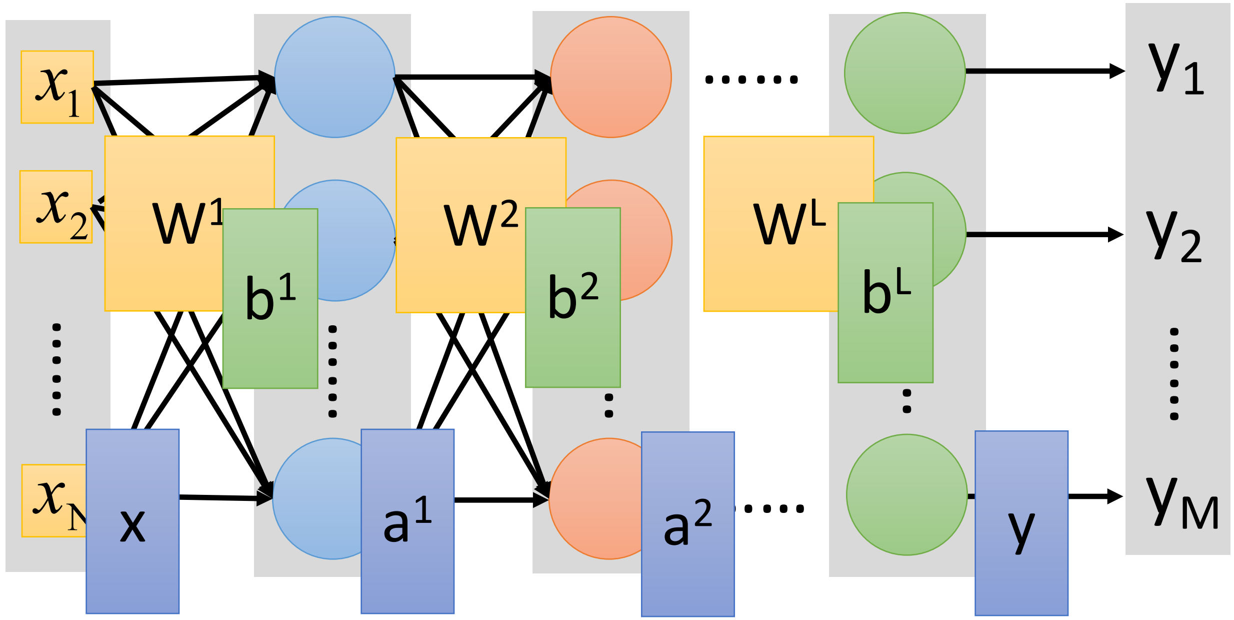


$$\sigma\Big(\underbrace{\begin{bmatrix}1 & -2\\ -1 & 1\end{bmatrix}\begin{bmatrix}1\\ -1\end{bmatrix}+\begin{bmatrix}1\\ 0\end{bmatrix}}_{\begin{bmatrix}4\\ -2\end{bmatrix}}\Big)=\begin{bmatrix}0.98\\ 0.12\end{bmatrix}$$

Neural Network


x_1
x_2
x_N
x
W^1
b^1
a^1
W^2
b^2
a^2
W^L
b^L
y
y_1
y_2
y_M
$\sigma($ W^1 x + b^1 $)$
$\sigma($ W^2 a^1 + b^2 $)$
$\sigma($ W^L a^{L-1} + b^L $)$

Neural Network



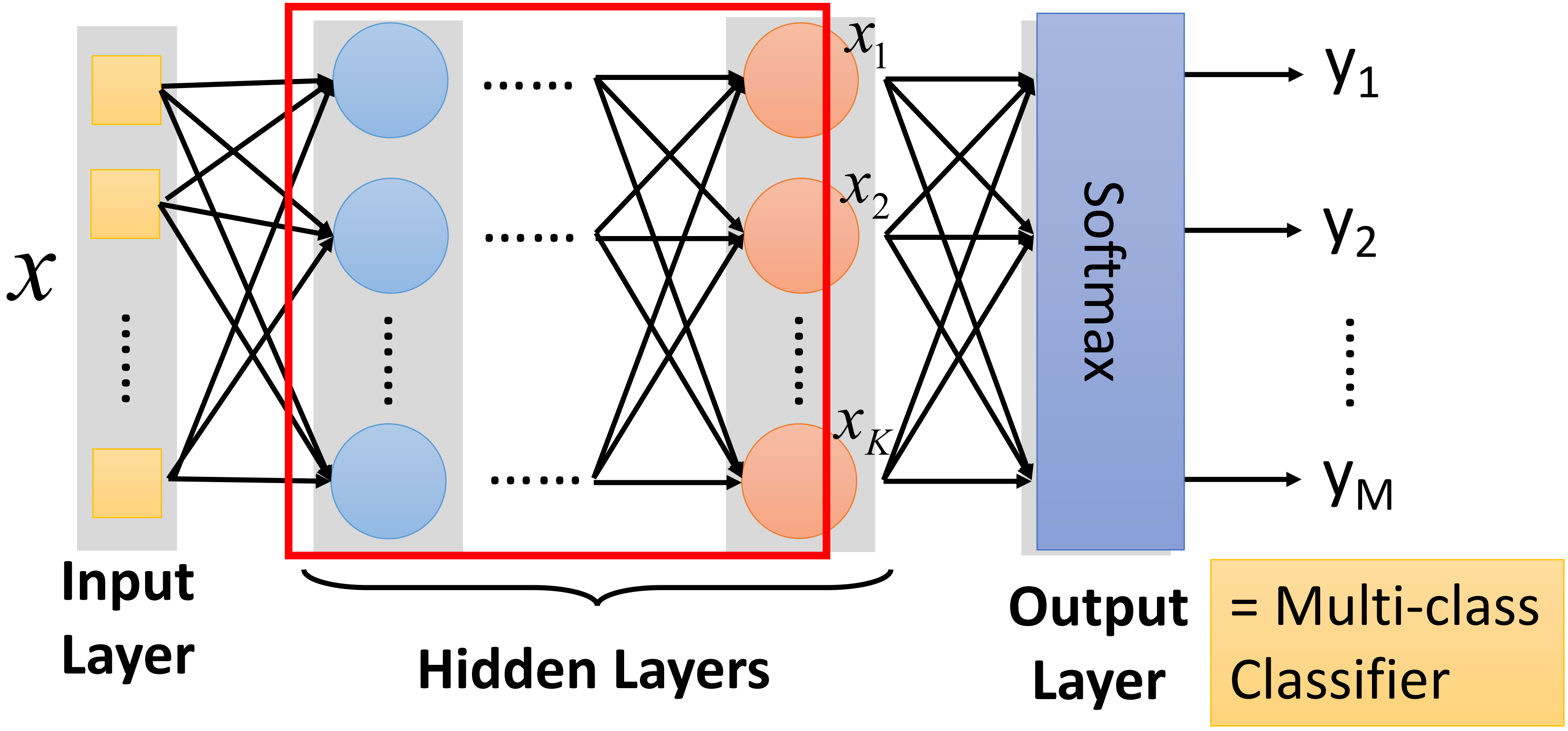
$$y = f(x)$$

Using parallel computing techniques to speed up matrix operation

$$= \sigma(W^L \cdots \sigma(W^2 \; \sigma(W^1 \; x + b^1) + b^2) \cdots + b^L)$$

Output Layer
as Multi-Class Classifier

Feature extractor replacing feature engineering



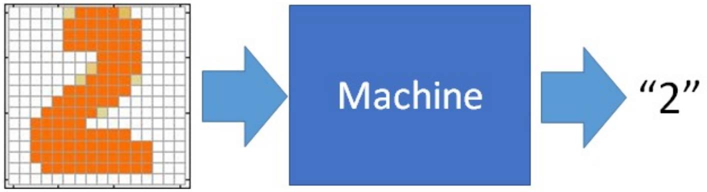
Input Layer

Hidden Layers

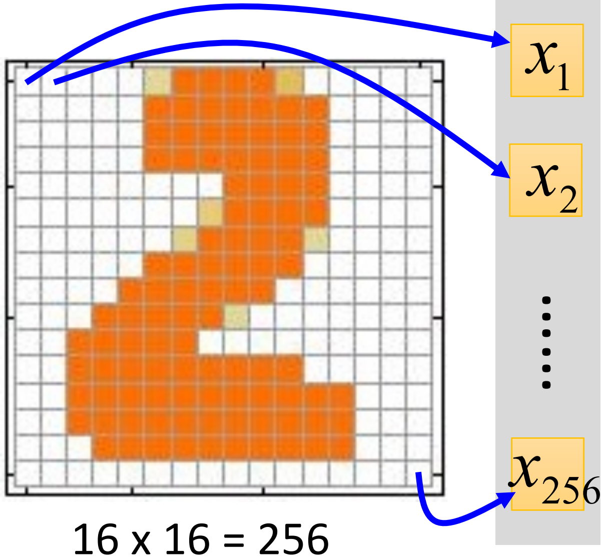
Output Layer

= Multi-class Classifier

Example Application



Input

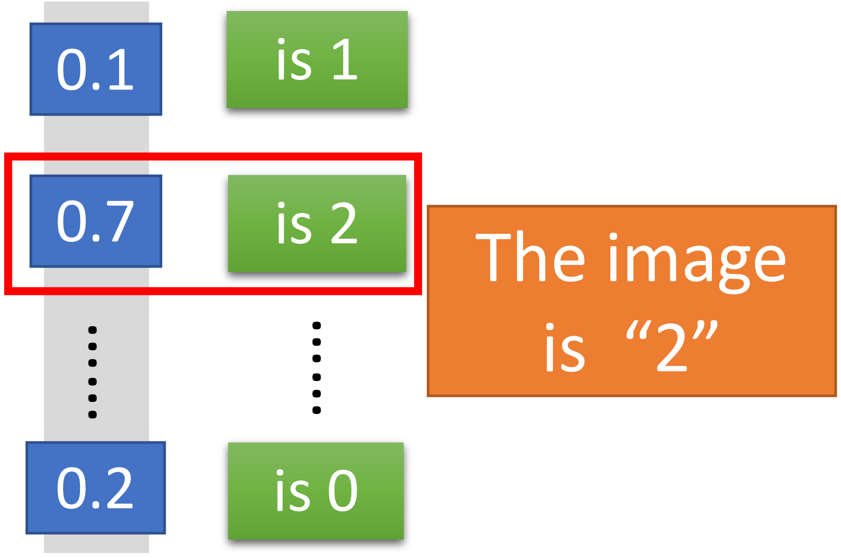


16 x 16 = 256

Ink → 1

No ink → 0

Output



0.1 is 1

0.7 is 2

……

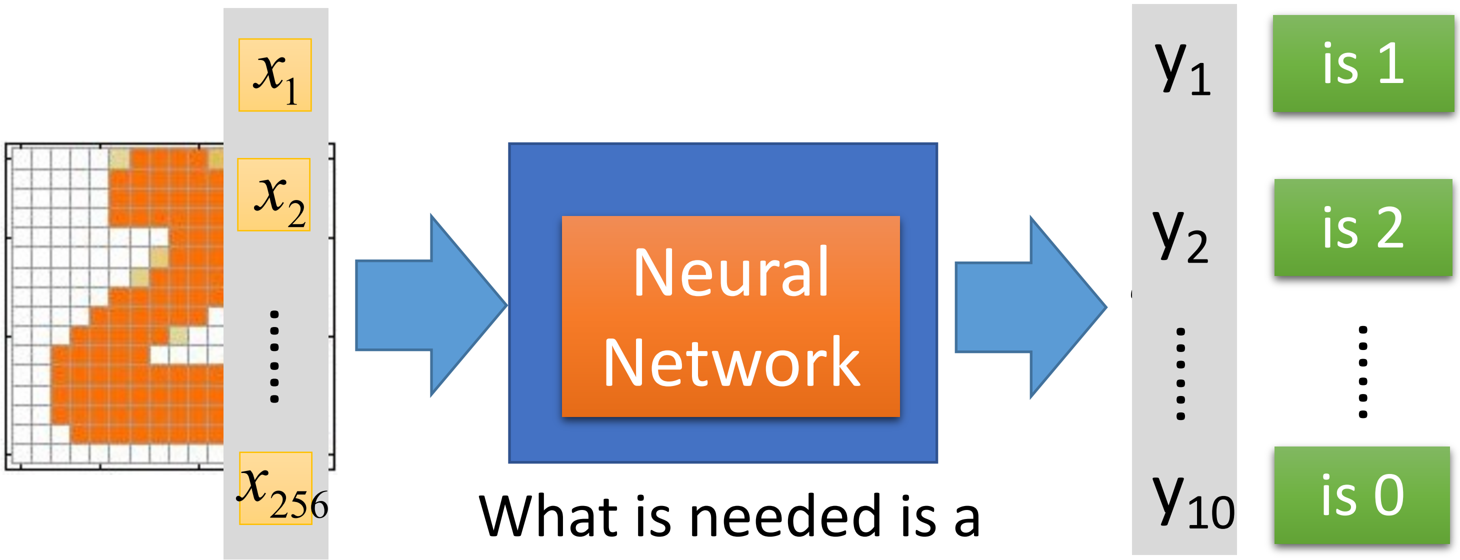
0.2 is 0

The image is “2”

Each dimension represents the confidence of a digit.

Example Application

- Handwriting Digit Recognition

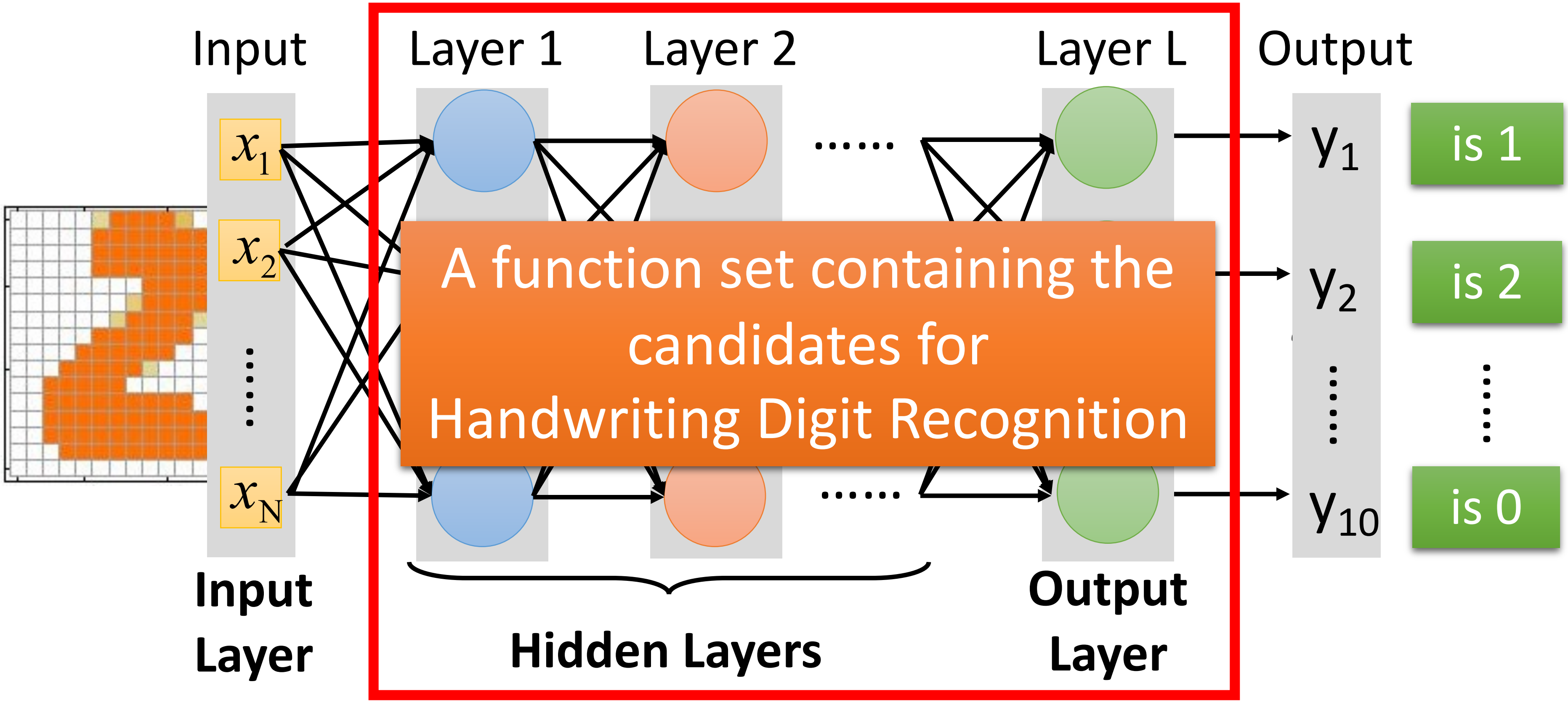


What is needed is a function ……

Input:
256-dim vector

output:
10-dim vector

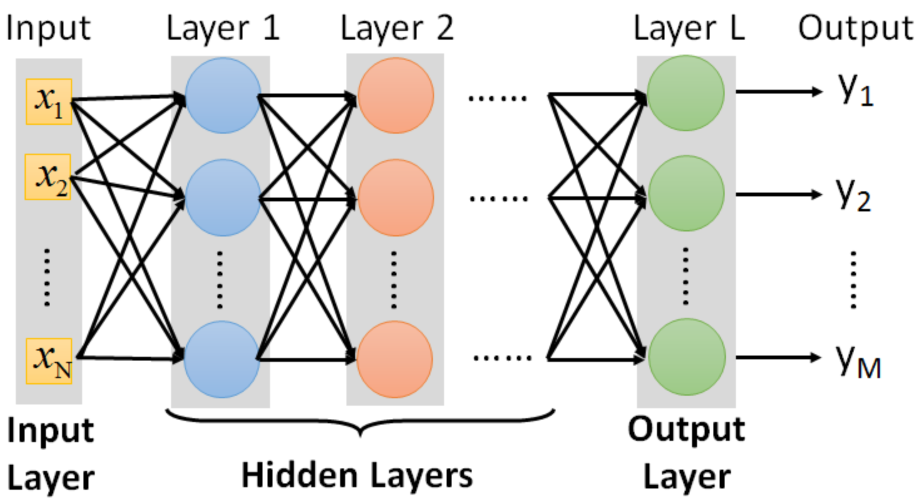
Example Application



A function set containing the candidates for Handwriting Digit Recognition

You need to decide the network structure to let a good function in your function set.

FAQ



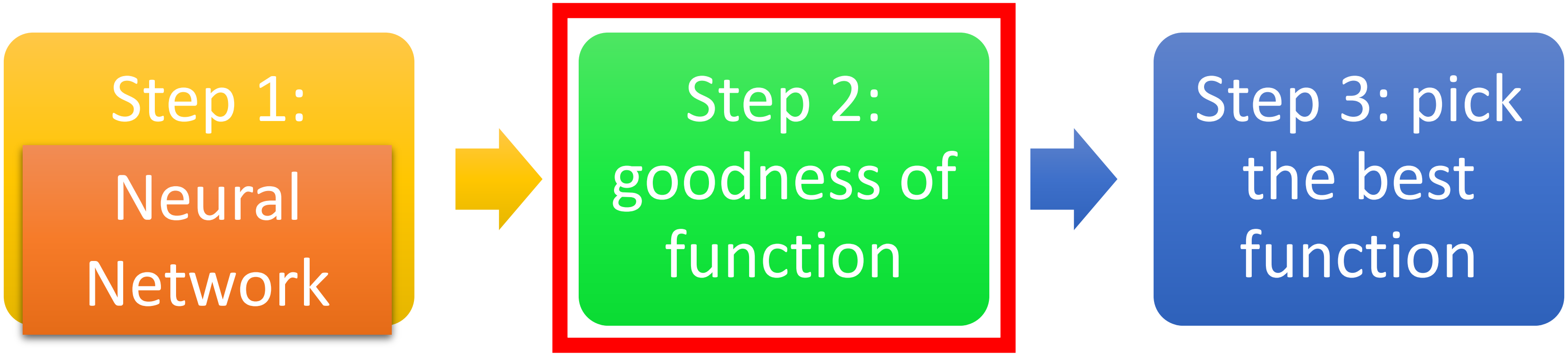
- Q: How many layers? How many neurons for each layer?

Trial and Error + Intuition

- Q: Can the structure be automatically determined?
  - E.g. Evolutionary Artificial Neural Networks
- Q: Can we design the network structure?

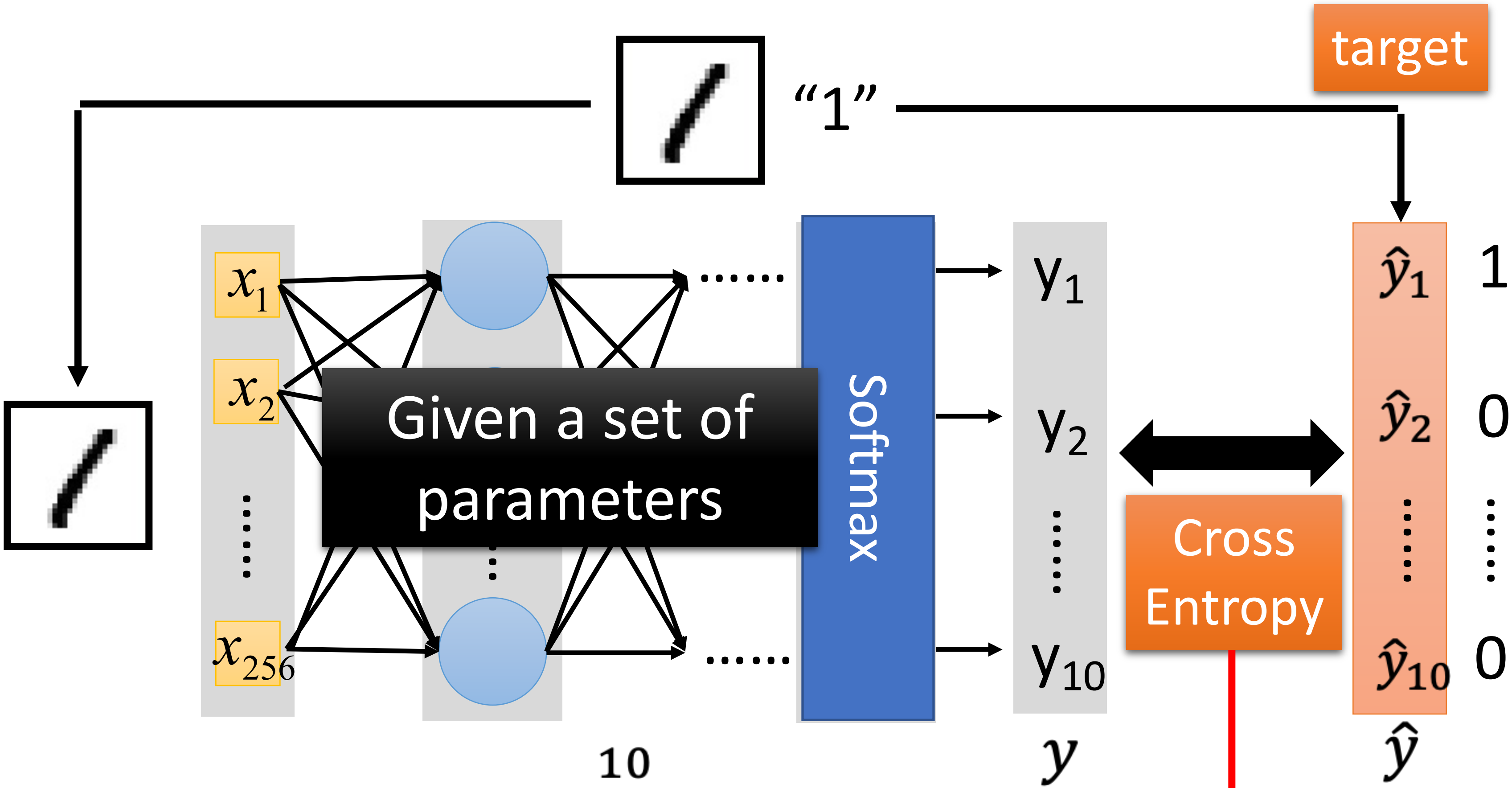
Convolutional Neural Network (CNN)

Three Steps for Deep Learning



Deep Learning is so simple ……

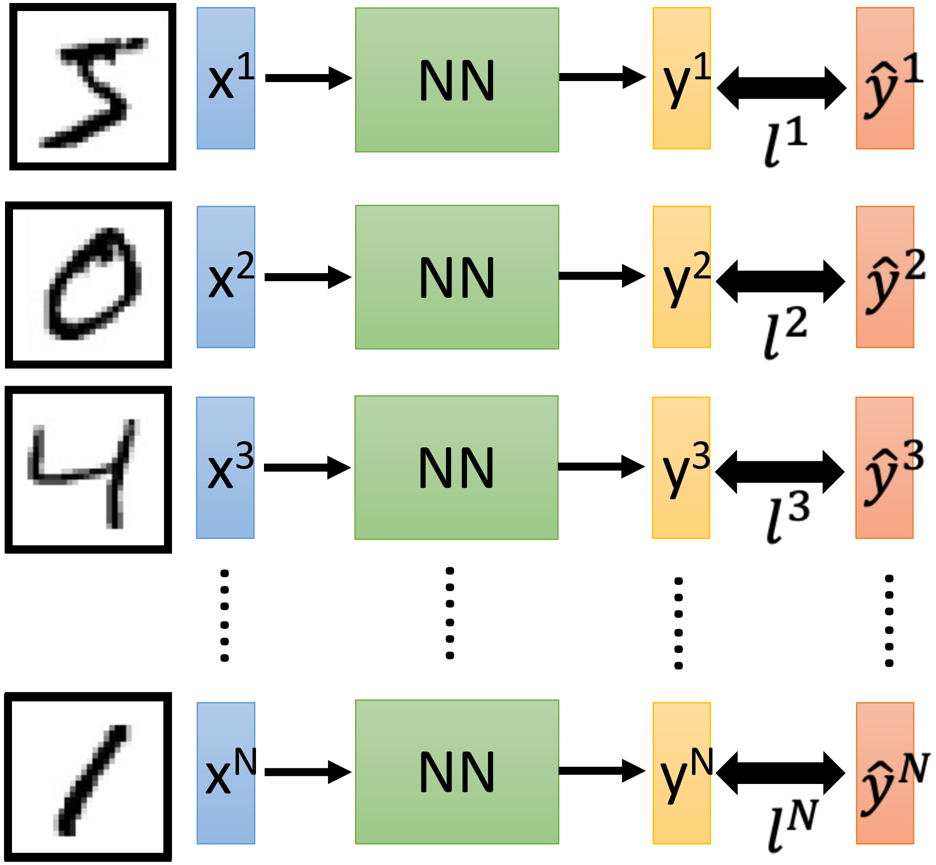
Loss for an Example



$$l(y, \hat{y}) = -\sum_{i=1}^{10} \hat{y}_i \ln y_i$$

Total Loss

For all training data …



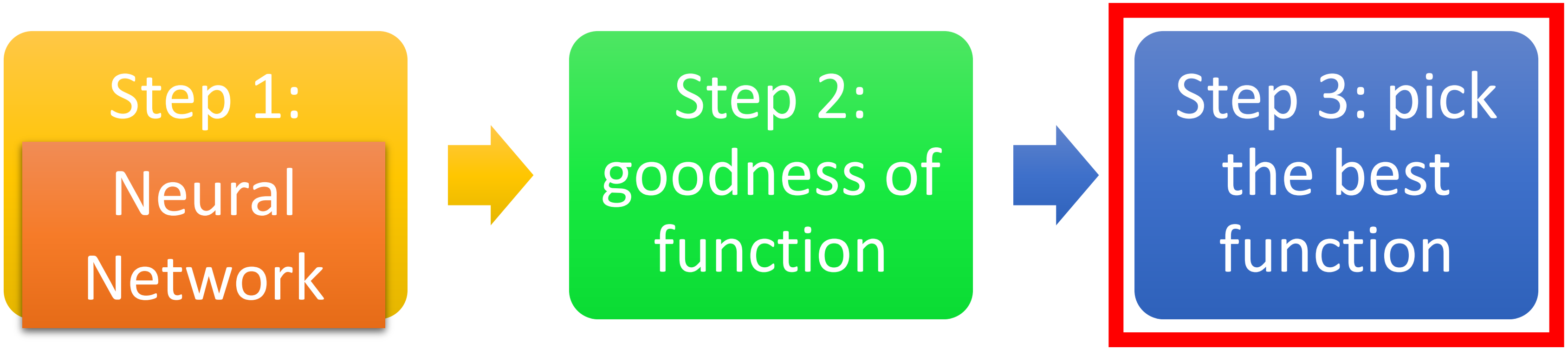
Total Loss:

$$\boxed{L = \sum_{n=1}^{N} l^n}$$

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

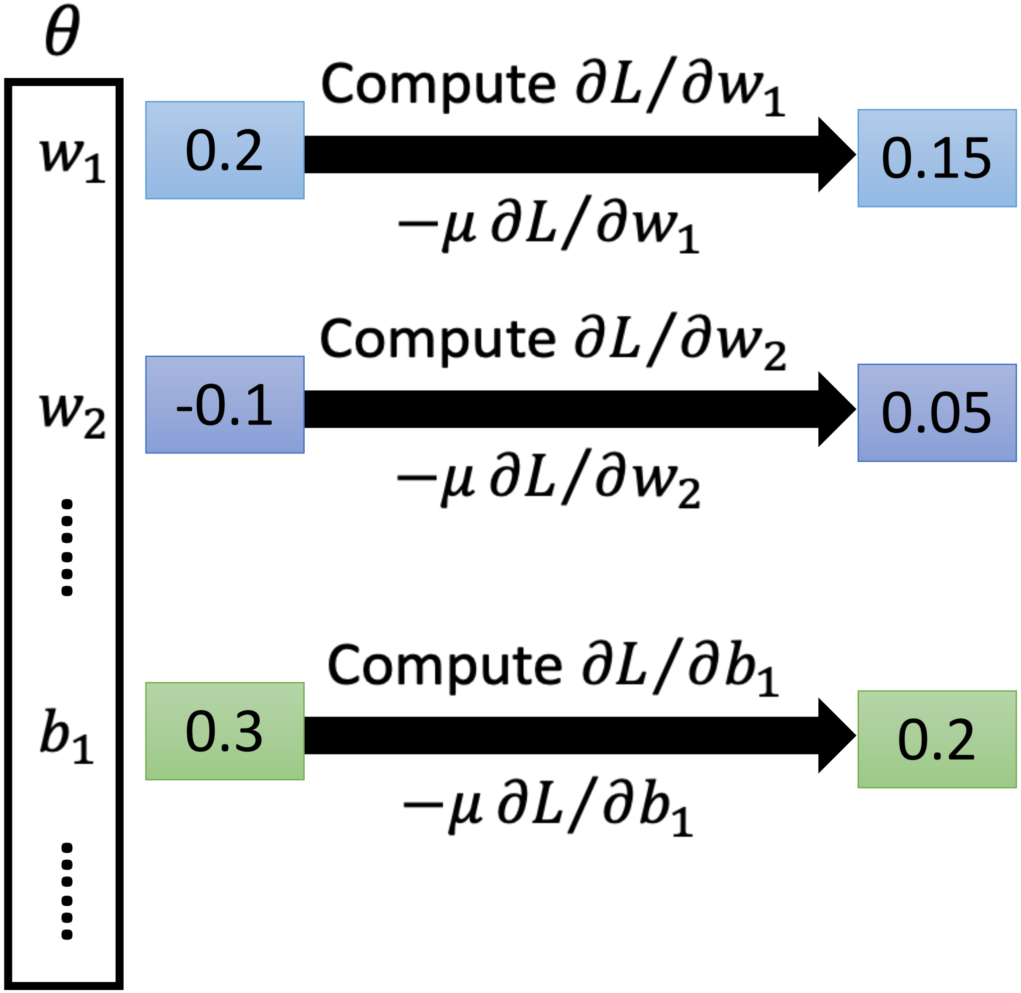
Find ***the network parameters θ^**** that minimize total loss L

Three Steps for Deep Learning



Deep Learning is so simple ……

Gradient Descent



θ

w_1 0.2 Compute $\partial L/\partial w_1$ $-\mu\, \partial L/\partial w_1$ 0.15

w_2 -0.1 Compute $\partial L/\partial w_2$ $-\mu\, \partial L/\partial w_2$ 0.05

$\vdots$

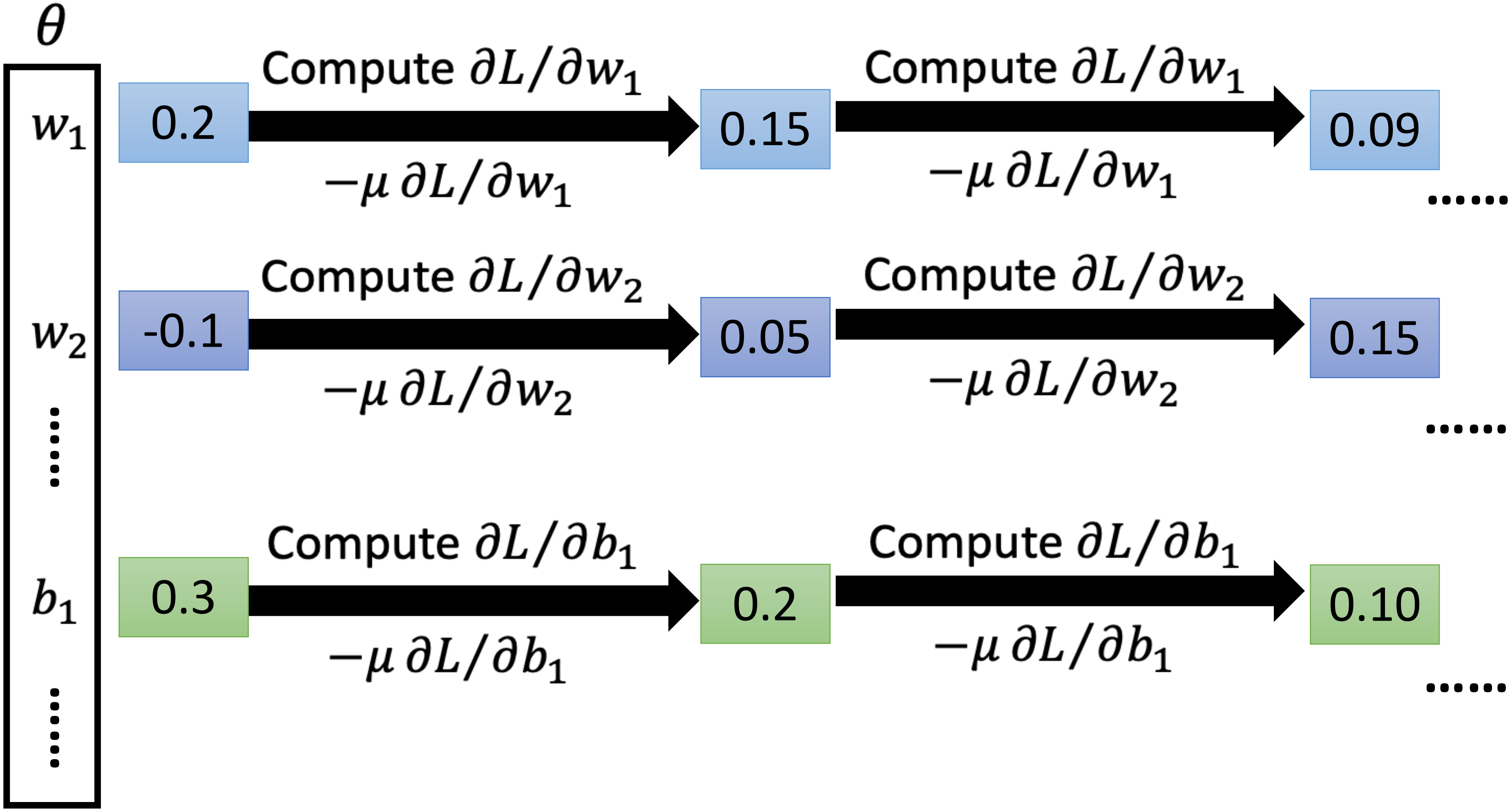
b_1 0.3 Compute $\partial L/\partial b_1$ $-\mu\, \partial L/\partial b_1$ 0.2

$\vdots$

$$\nabla L = \begin{bmatrix} \dfrac{\partial L}{\partial w_1} \\ \dfrac{\partial L}{\partial w_2} \\ \vdots \\ \dfrac{\partial L}{\partial b_1} \\ \vdots \end{bmatrix}$$

gradient

Gradient Descent



θ

w_1 0.2 Compute $\partial L/\partial w_1$ $-\mu\,\partial L/\partial w_1$ 0.15 Compute $\partial L/\partial w_1$ $-\mu\,\partial L/\partial w_1$ 0.09 ……

w_2 -0.1 Compute $\partial L/\partial w_2$ $-\mu\,\partial L/\partial w_2$ 0.05 Compute $\partial L/\partial w_2$ $-\mu\,\partial L/\partial w_2$ 0.15 ……

⋮

b_1 0.3 Compute $\partial L/\partial b_1$ $-\mu\,\partial L/\partial b_1$ 0.2 Compute $\partial L/\partial b_1$ $-\mu\,\partial L/\partial b_1$ 0.10 ……

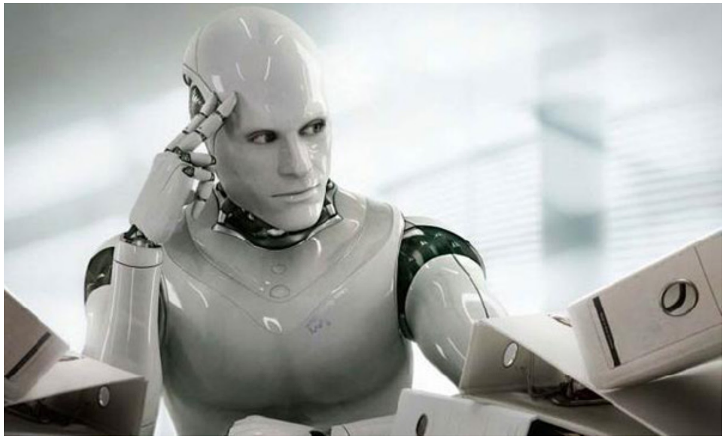
⋮

Gradient Descent

This is the “learning” of machines in deep learning ……

➡ Even alpha go using this approach.

People image ……



Actually …..

I hope you are not too disappointed :p

Backpropagation

- Backpropagation: an efficient way to compute $\partial L / \partial w$ in neural network

TensorFlow

torch

theano

Caffe

Microsoft CNTK

Chainer

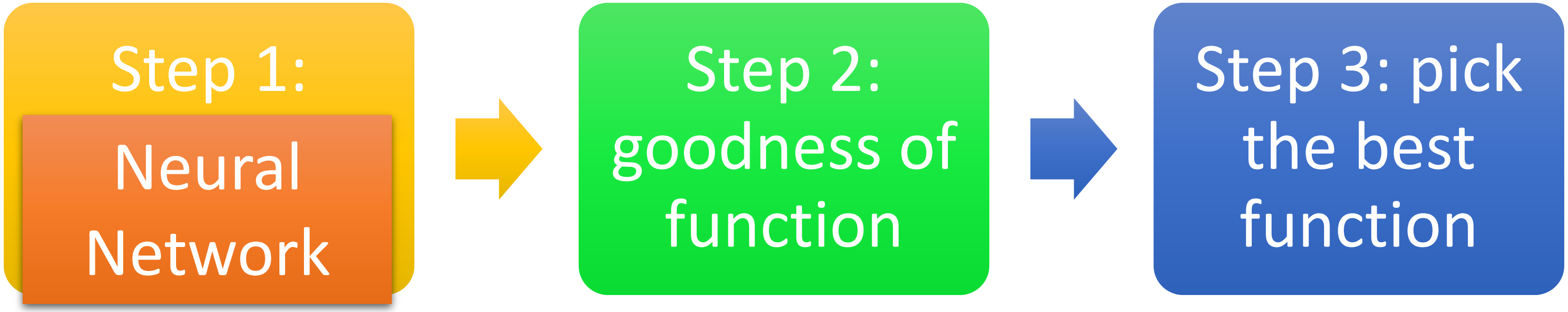
Deep Learning library produced by Amazon
DSSTNE

mxnet

libdnn
台大周伯威
同學開發

Ref: http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2/Lecture/DNN%20backprop.ecm.mp4/index.html

Three Steps for Deep Learning

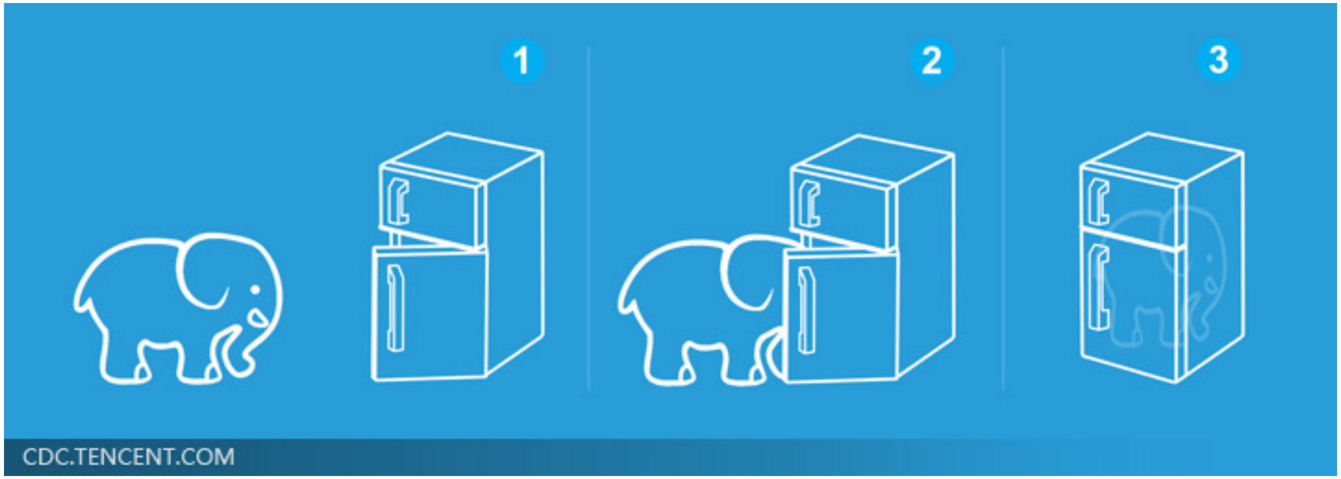


Step 1: Neural Network

Step 2: goodness of function

Step 3: pick the best function

Deep Learning is so simple ……



CDC.TENCENT.COM