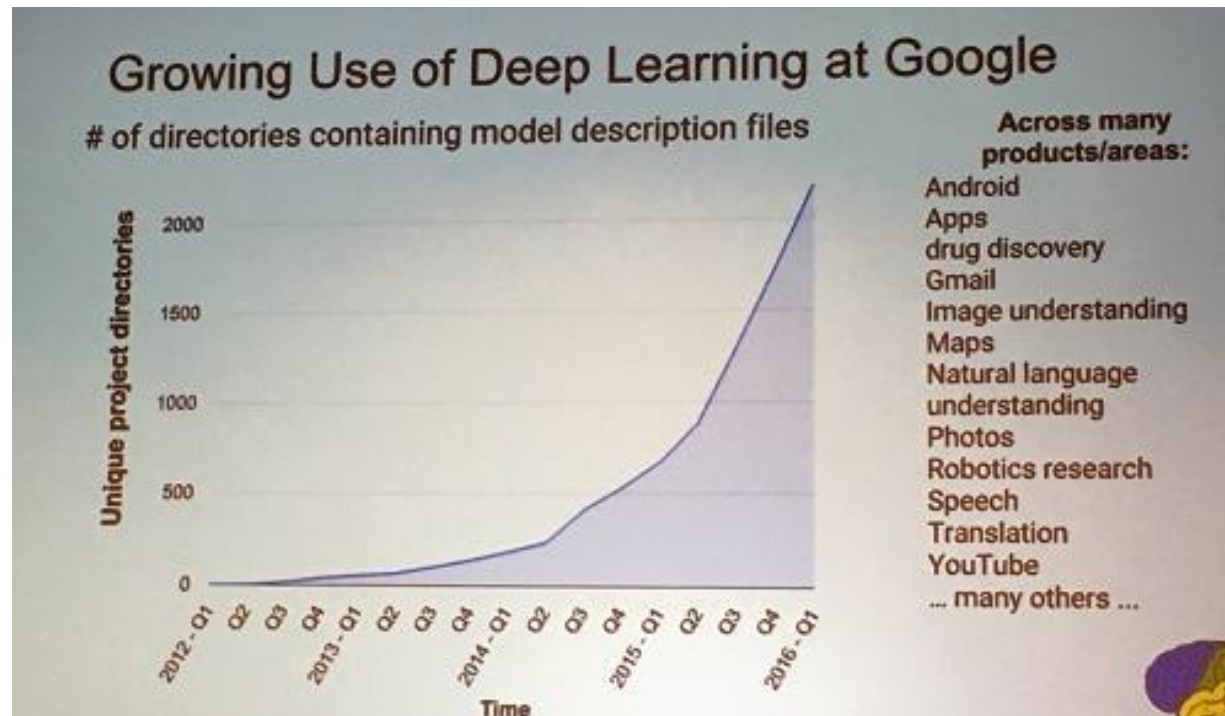


# Deep Learning

# Deep learning attracts lots of attention.

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



Deep learning trends at Google. Source: SIGMOD/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 (breakthrough)
- 2009: GPU
- 2011: Start to be popular in speech recognition
- 2012: win ILSVRC image competition

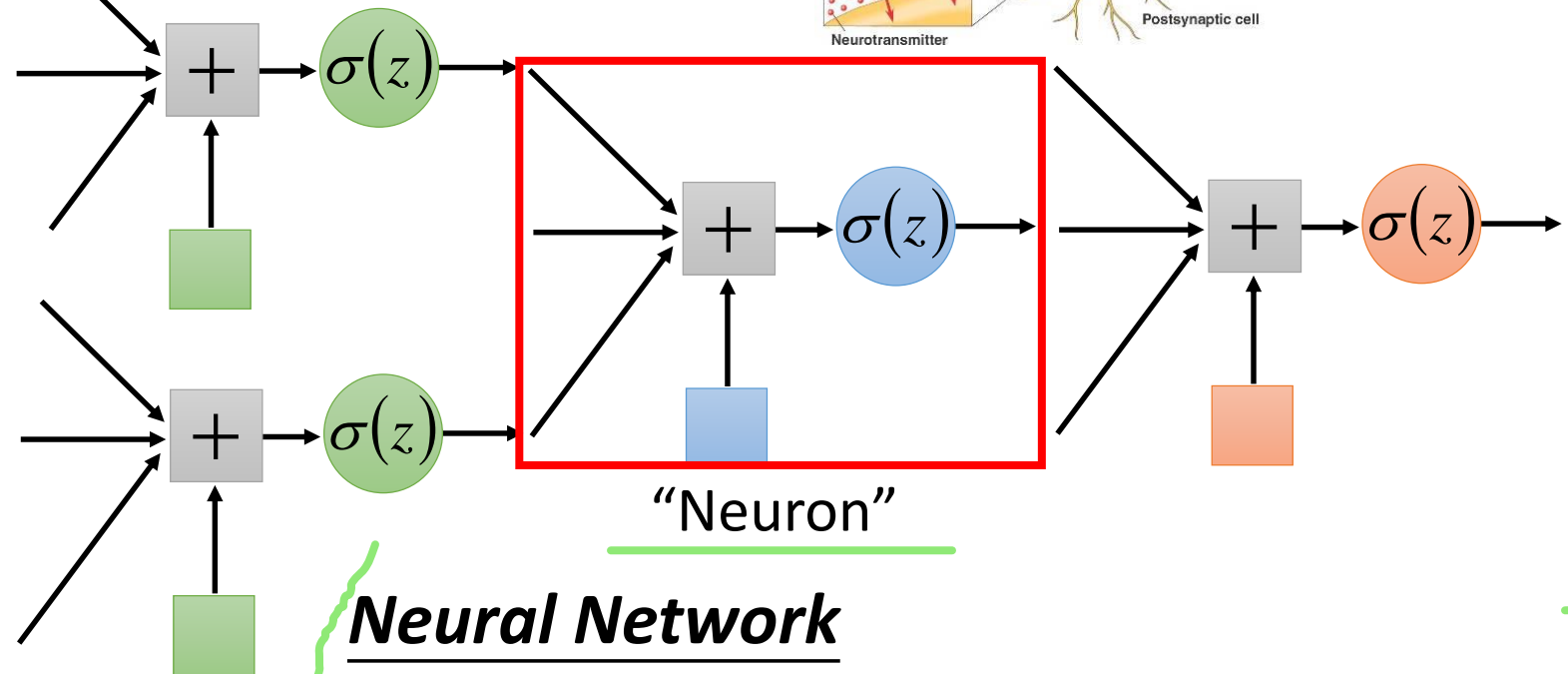
# Three Steps for Deep Learning



Deep Learning is so simple .....



# Neural Network

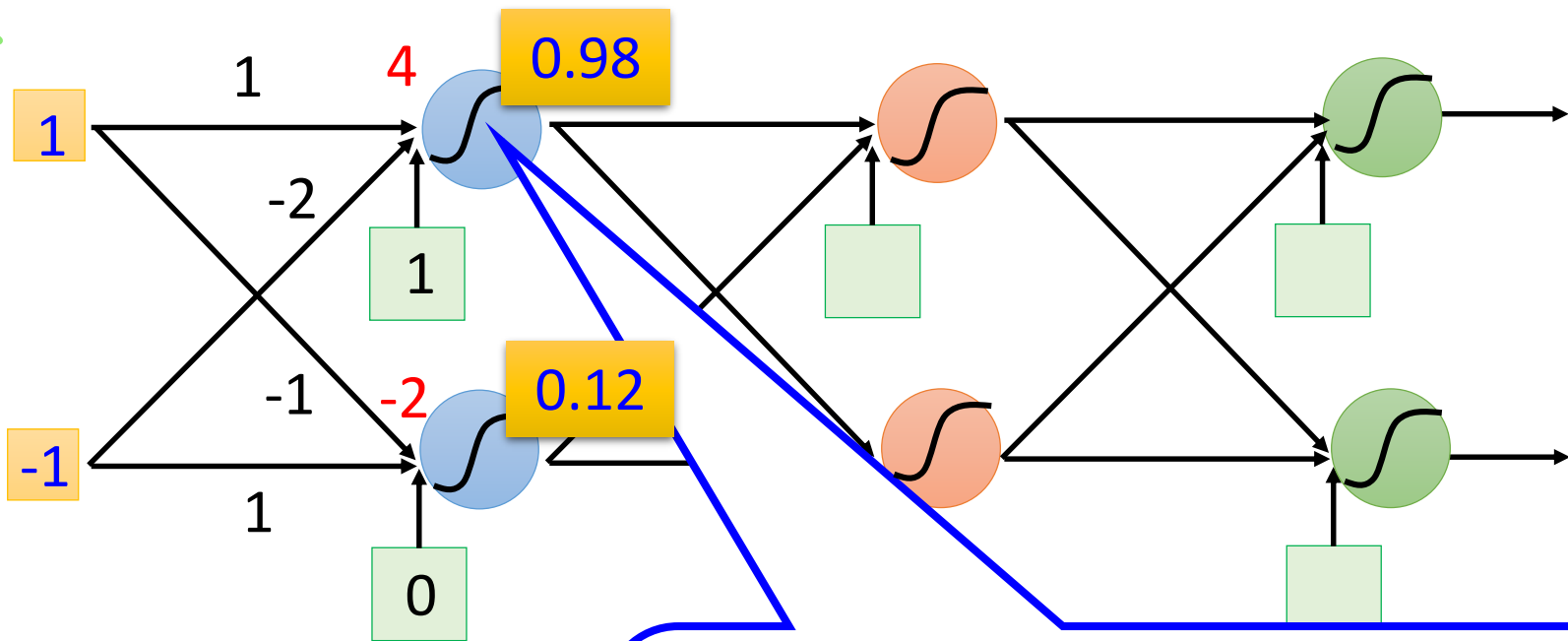


## Neural Network

Different connection leads to different network structures

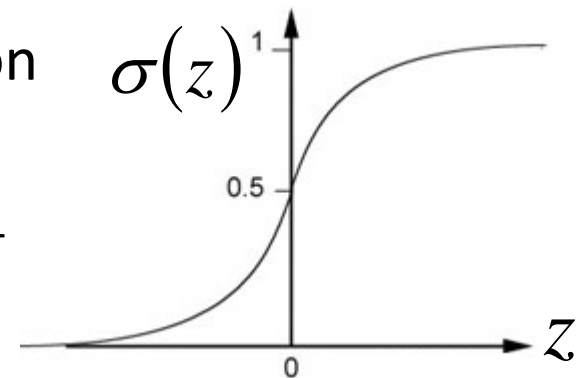
Network parameter  $\theta$ : all the weights and biases in the "neurons"

# Fully Connect Feedforward Network

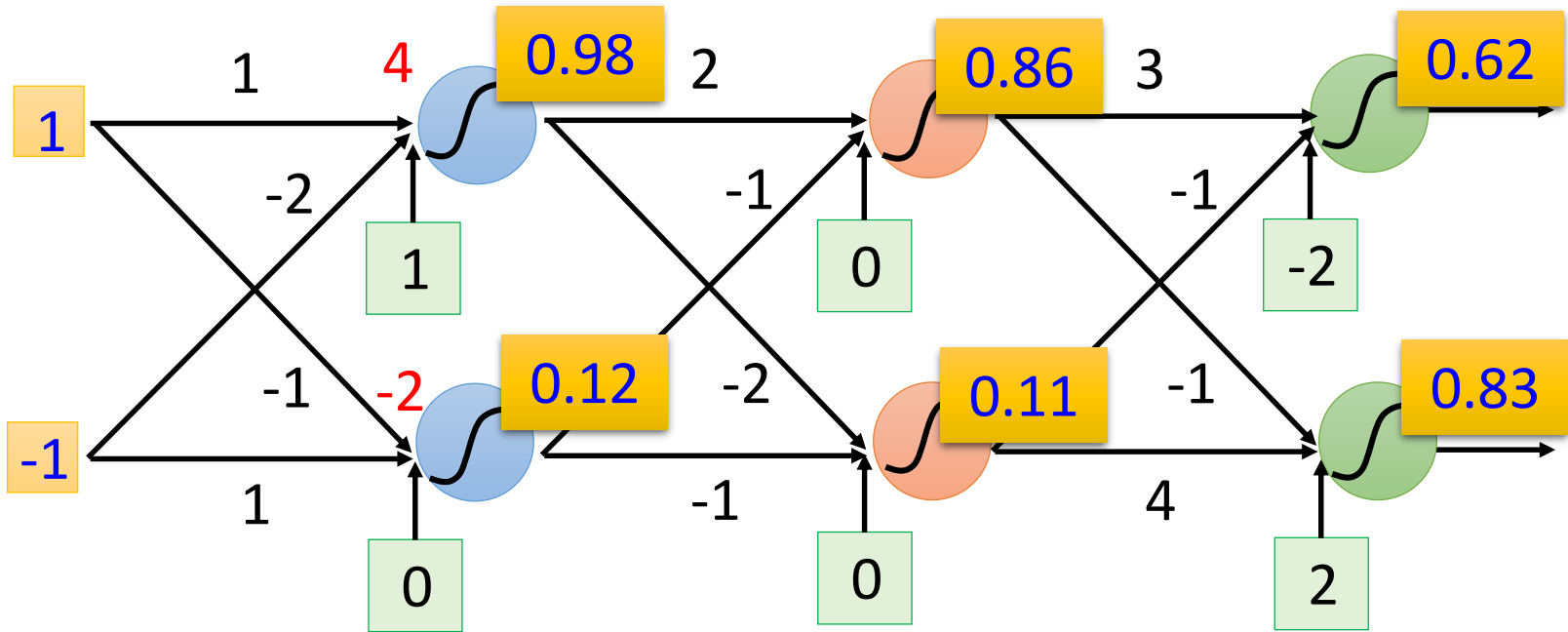


Sigmoid Function

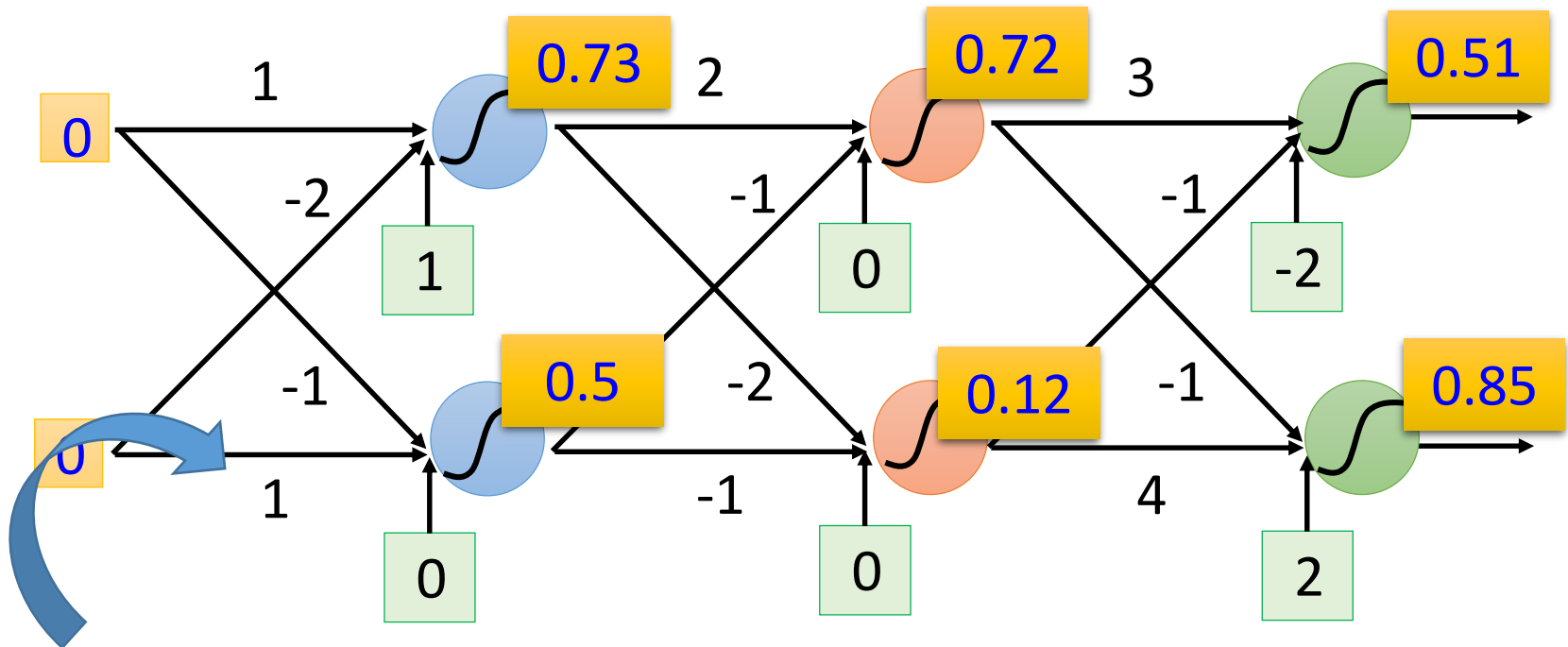
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$



# Fully Connect Feedforward Network



# Fully Connect Feedforward Network



This is a function.

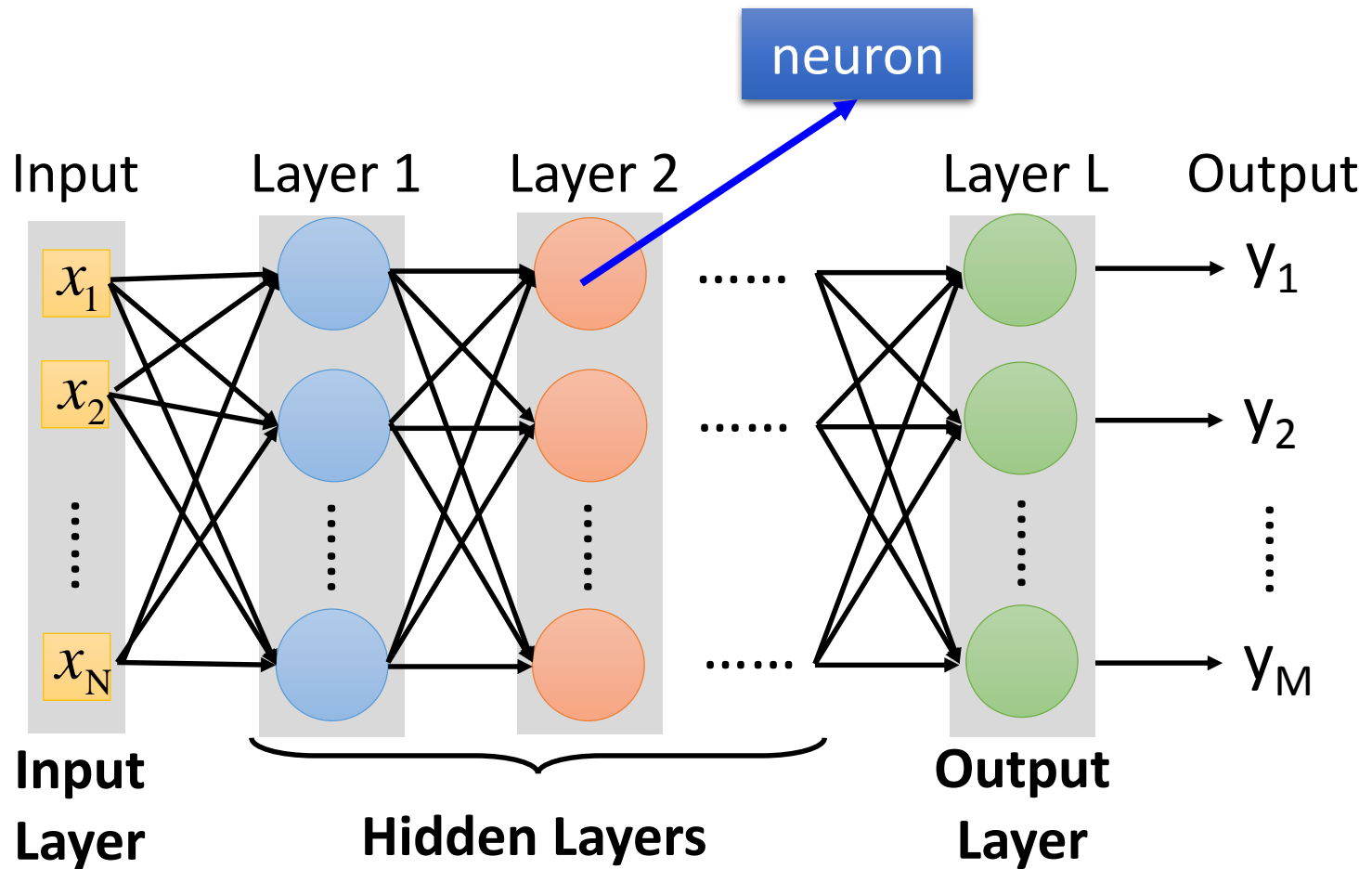
Input vector, output vector

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

Given network structure, define a function set



# Fully Connect Feedforward Network

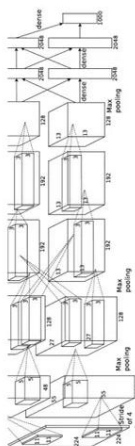


# Deep = Many hidden layers

[http://cs231n.stanford.edu/slides/winter1516\\_lecture8.pdf](http://cs231n.stanford.edu/slides/winter1516_lecture8.pdf)

8 layers

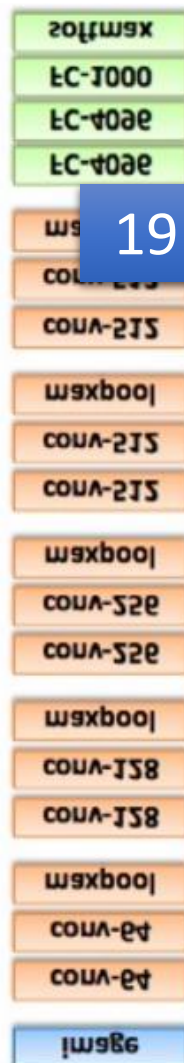
16.4%



AlexNet (2012)

19 layers

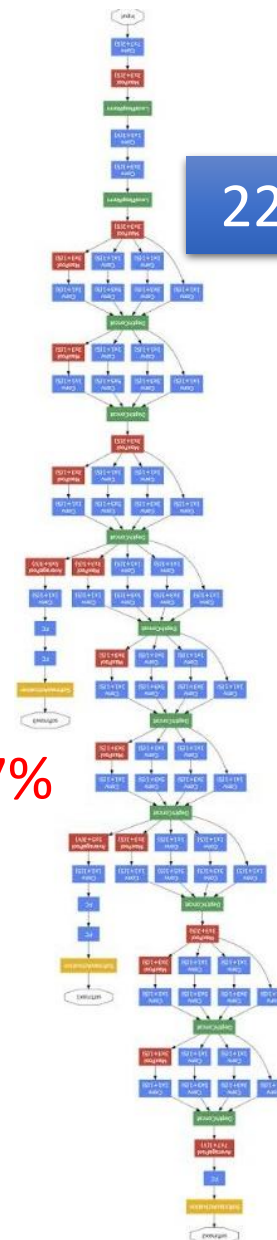
7.3%



VGG (2014)

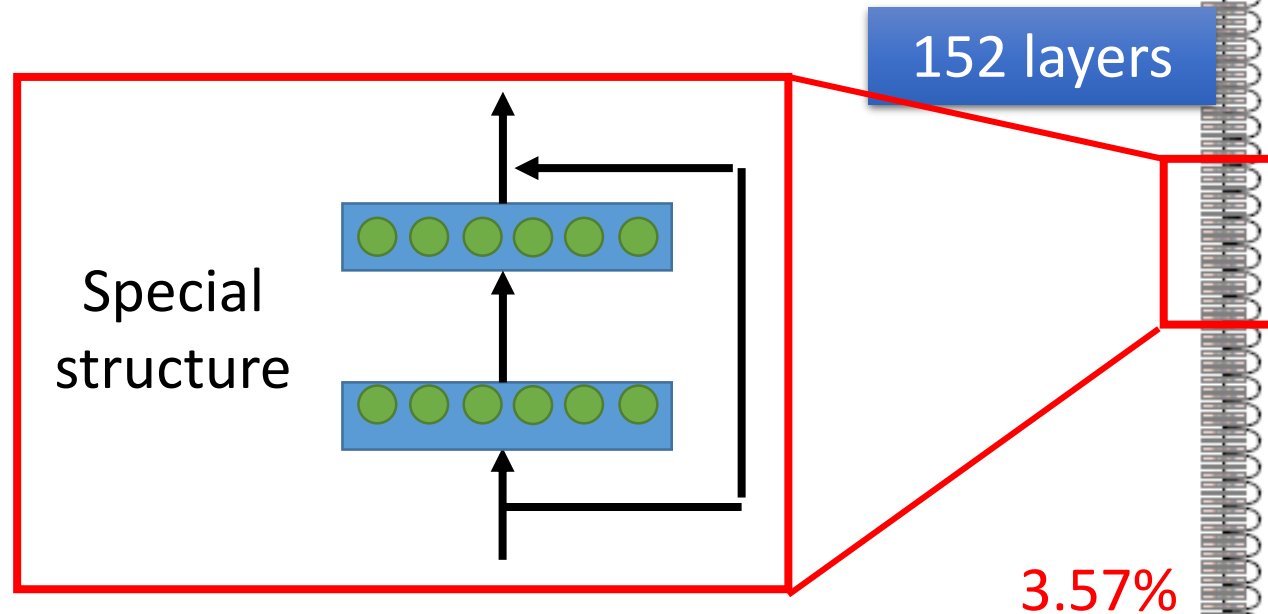
22 layers

6.7%



GoogleNet (2014)

# Deep = Many hidden layers



16.4%

AlexNet  
(2012)

7.3%

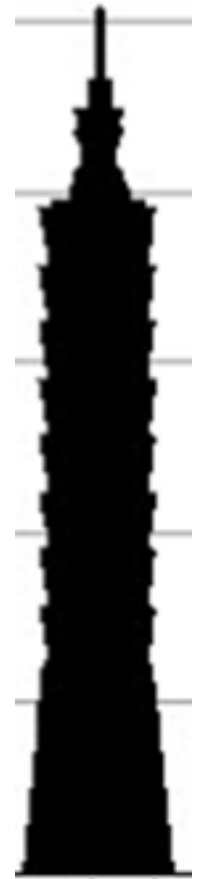
VGG  
(2014)

6.7%

GoogleNet  
(2014)

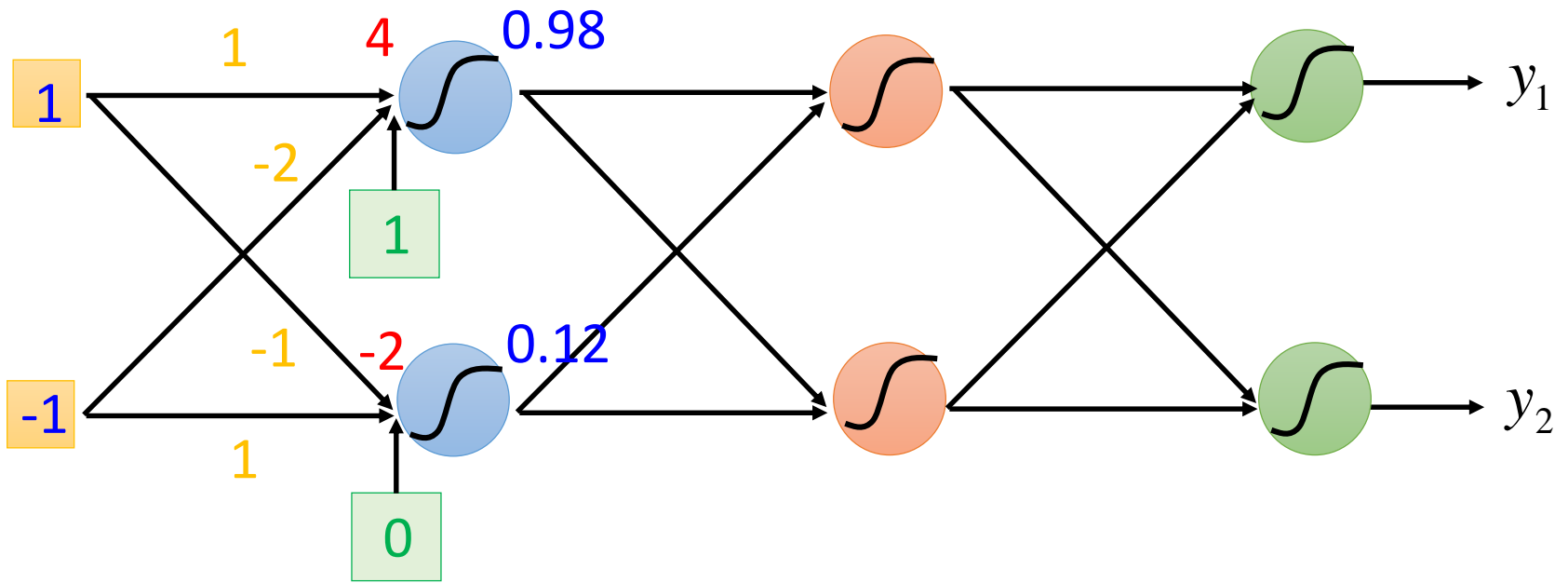
Residual Net  
(2015)

101 layers



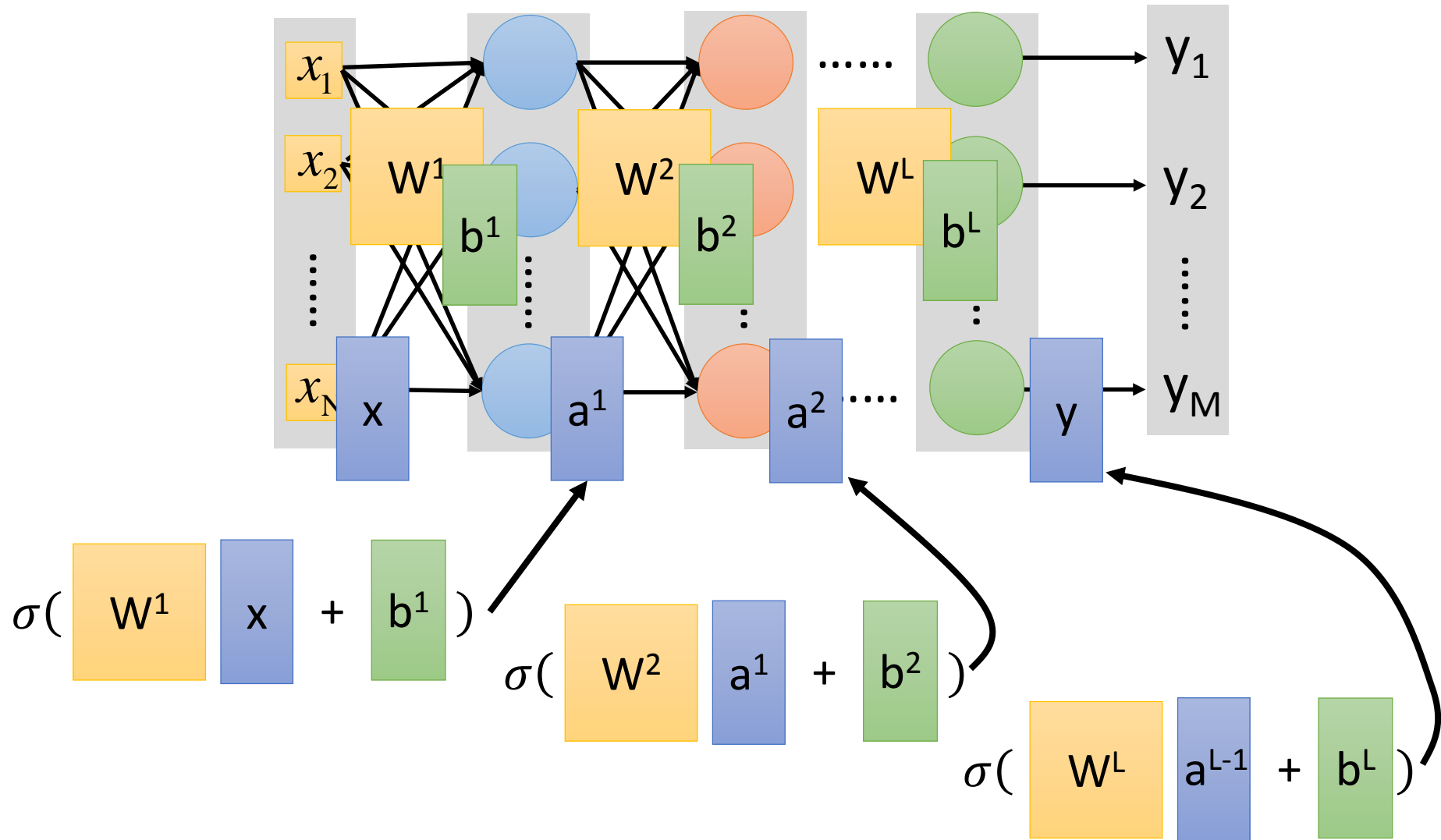
Taipei  
101

# Matrix Operation

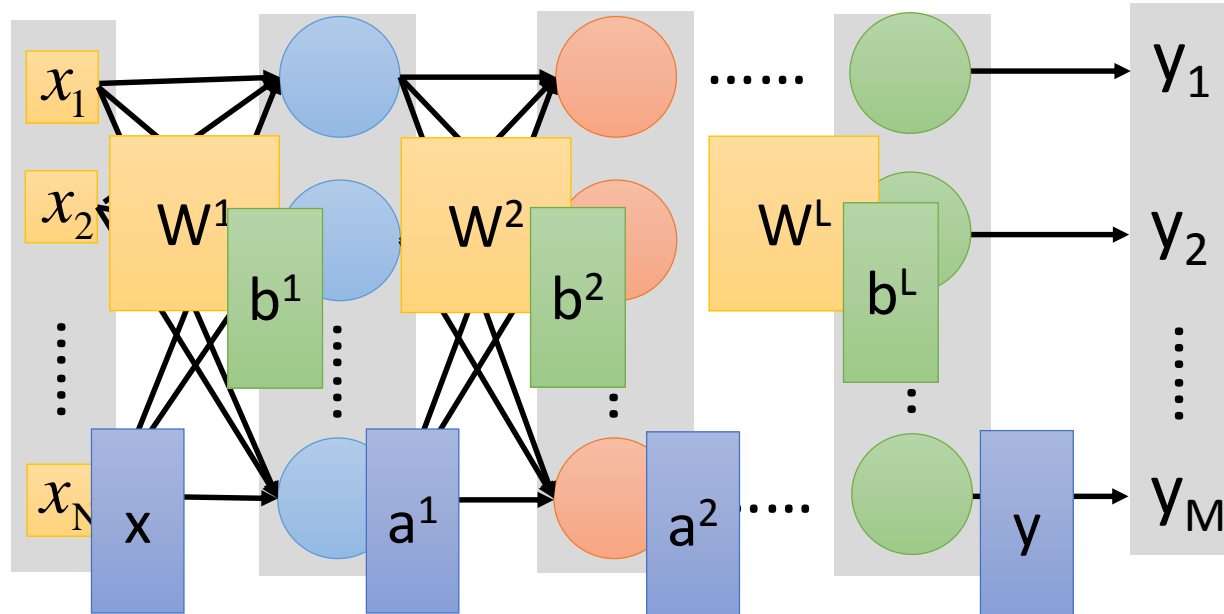


$$\sigma\left(\underbrace{\begin{bmatrix} 1 & -2 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix}}_{\begin{bmatrix} 4 \\ -2 \end{bmatrix}}\right) = \begin{bmatrix} 0.98 \\ 0.12 \end{bmatrix}$$

# Neural Network



# Neural Network



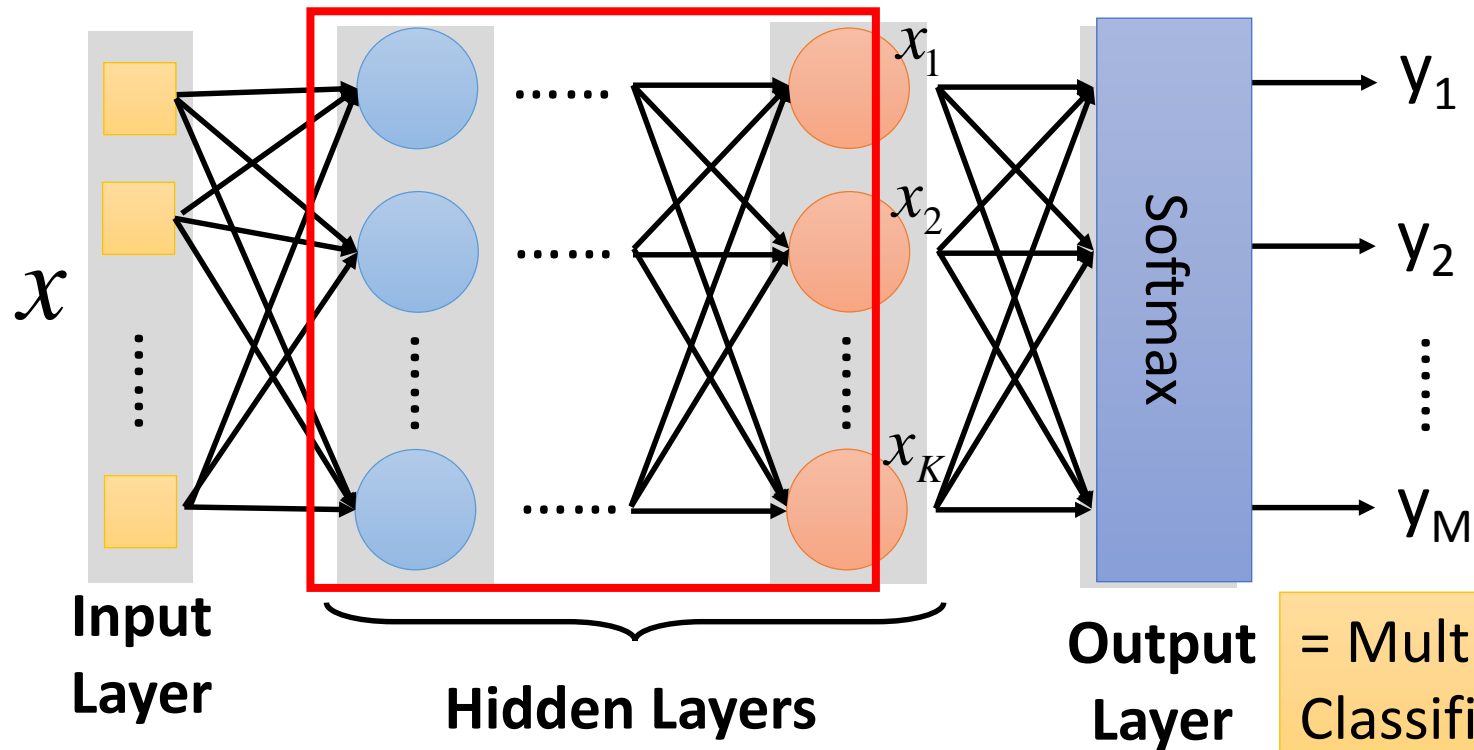
$$y = f(x)$$

Using parallel computing techniques to speed up matrix operation

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

# Output Layer

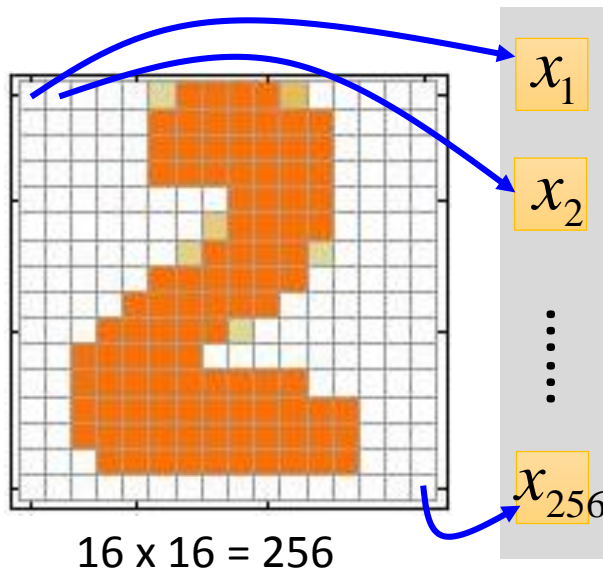
Feature extractor replacing  
feature engineering



# Example Application



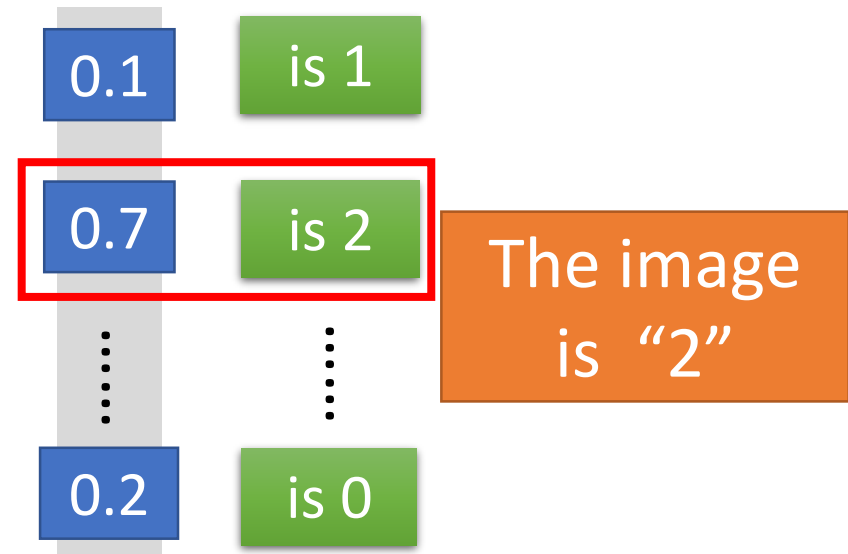
## Input



Ink  $\rightarrow$  1

No ink  $\rightarrow$  0

## Output

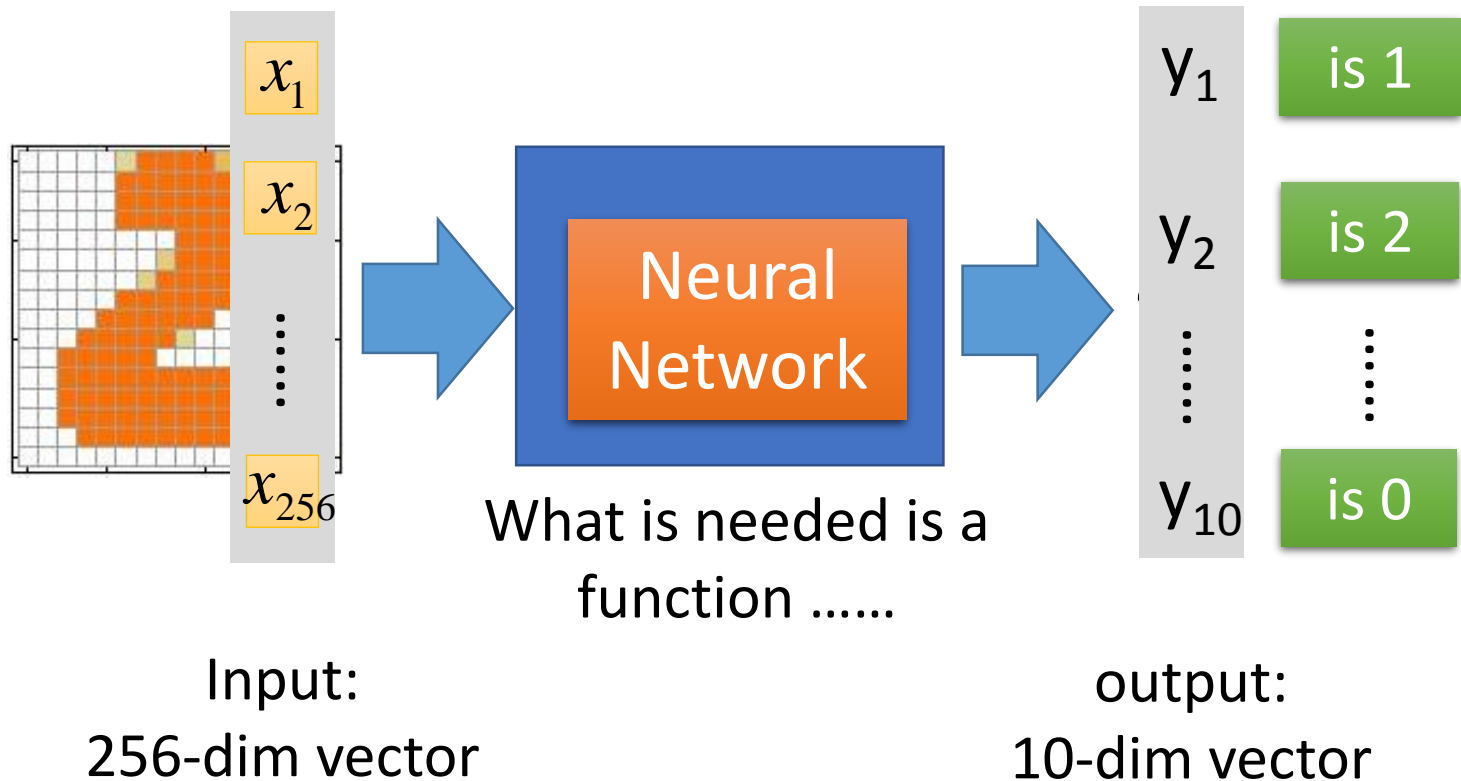


Each dimension represents the confidence of a digit.

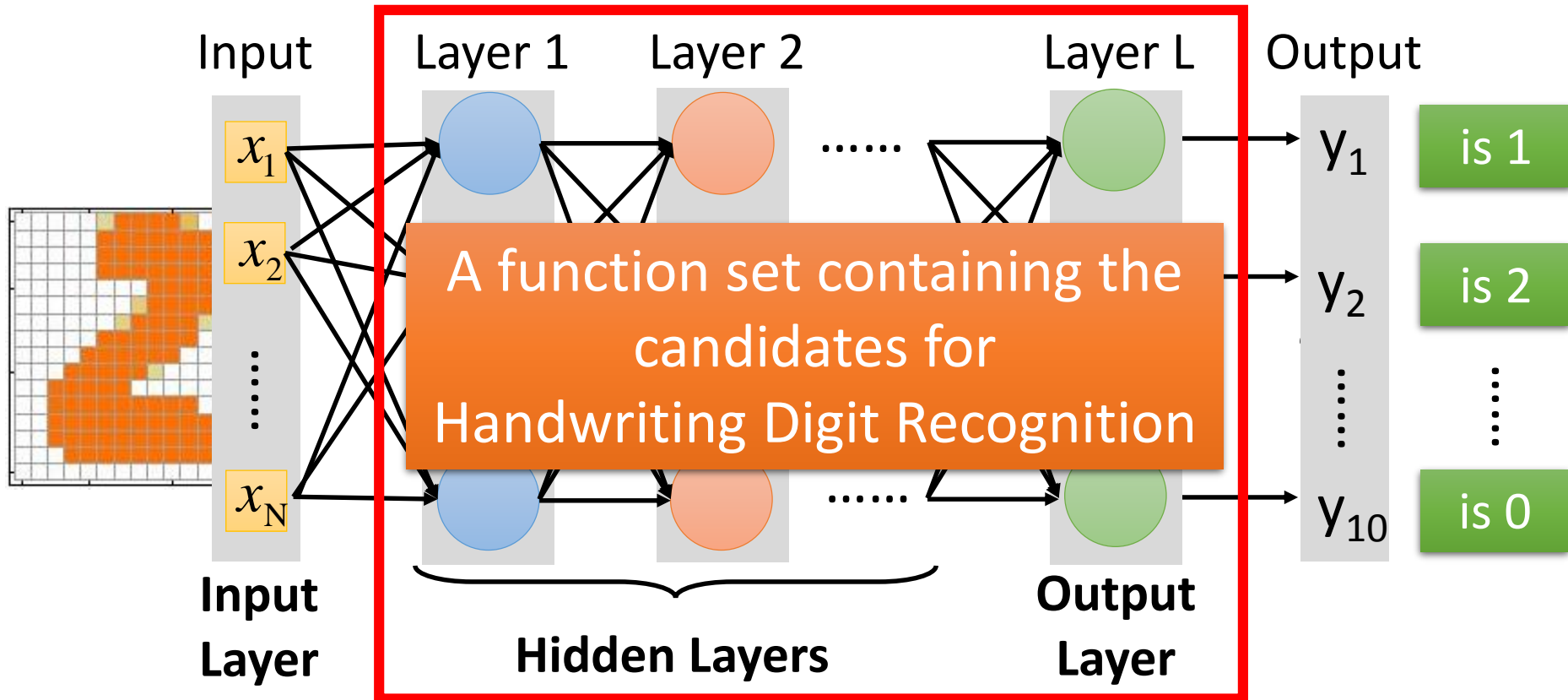


# Example Application

- Handwriting Digit Recognition

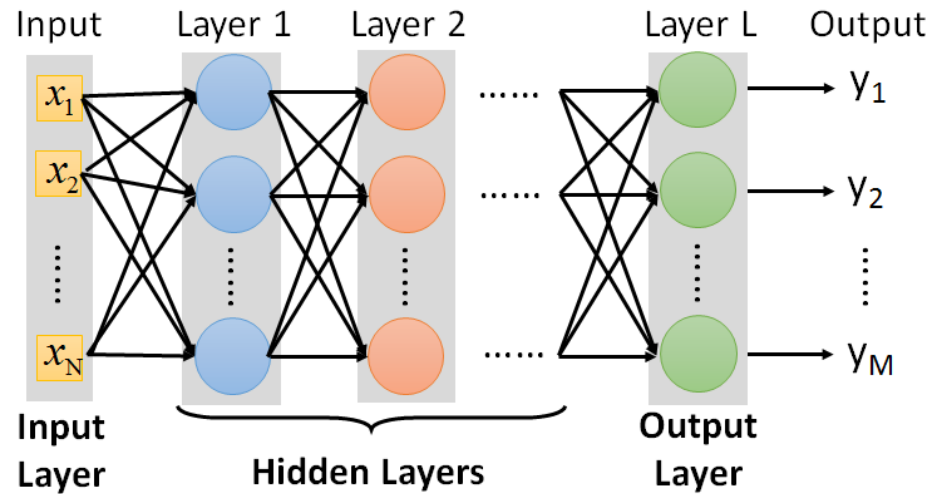


# Example Application



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)

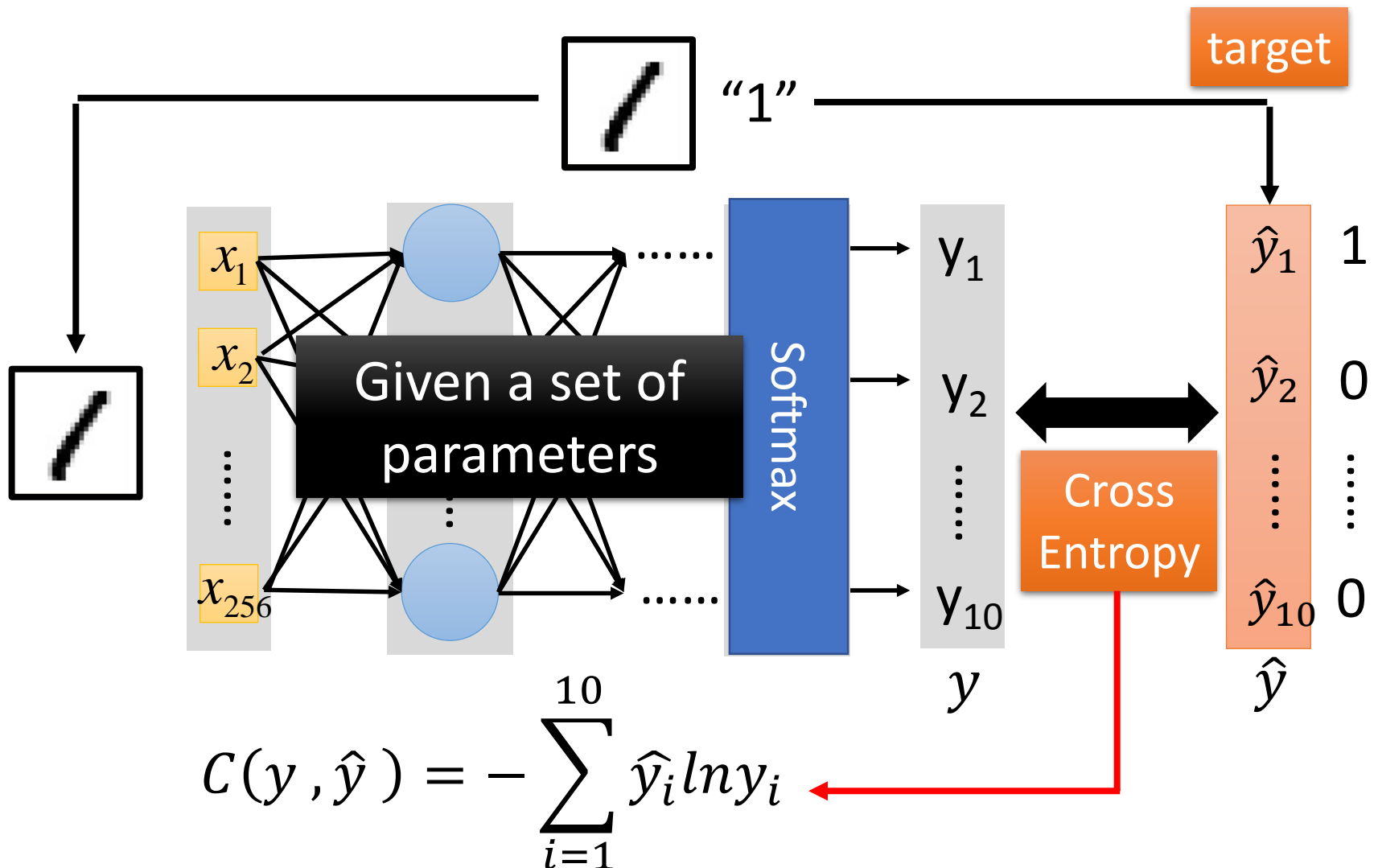
# Three Steps for Deep Learning



Deep Learning is so simple .....

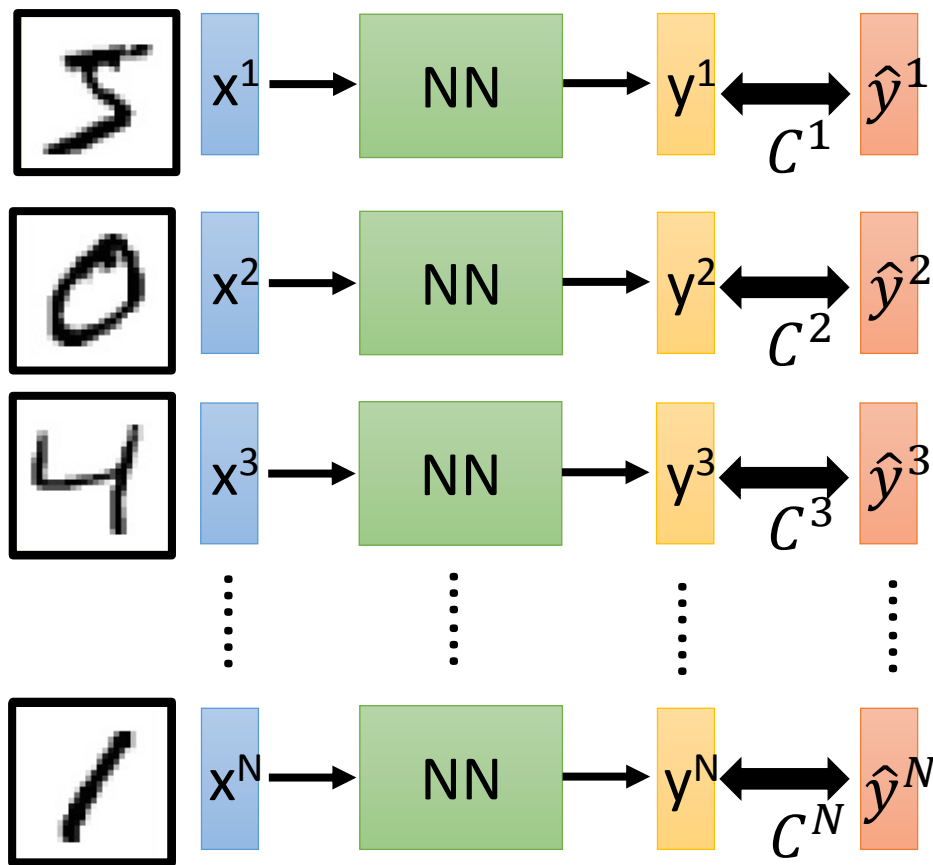


# Loss for an Example



# Total Loss

For all training data ...



Total Loss:

$$L = \sum_{n=1}^N C^n$$

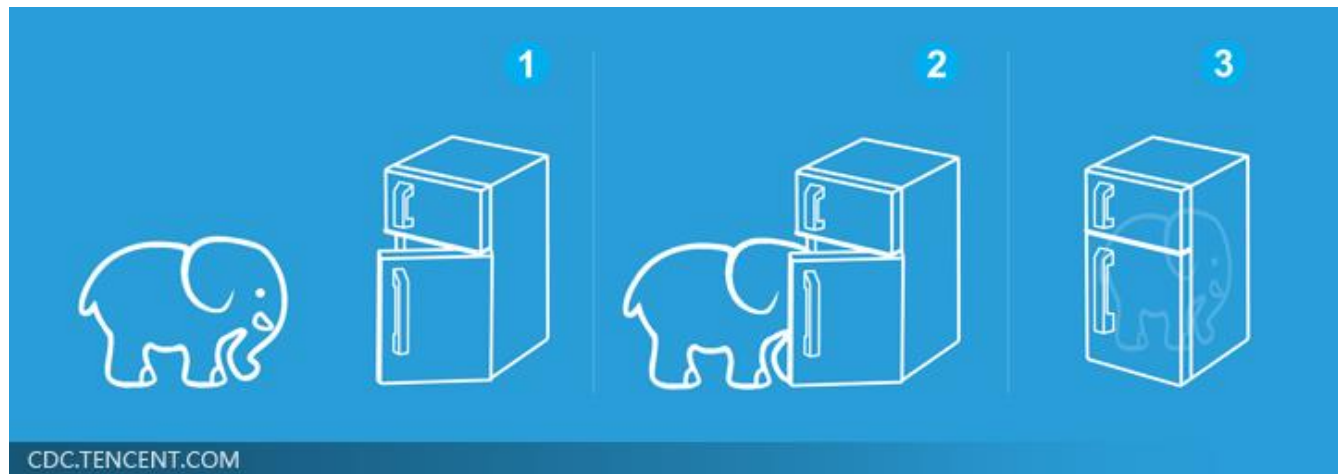
Find a function in function set that minimizes total loss  $L$

Find the network parameters  $\theta^*$  that minimize total loss  $L$

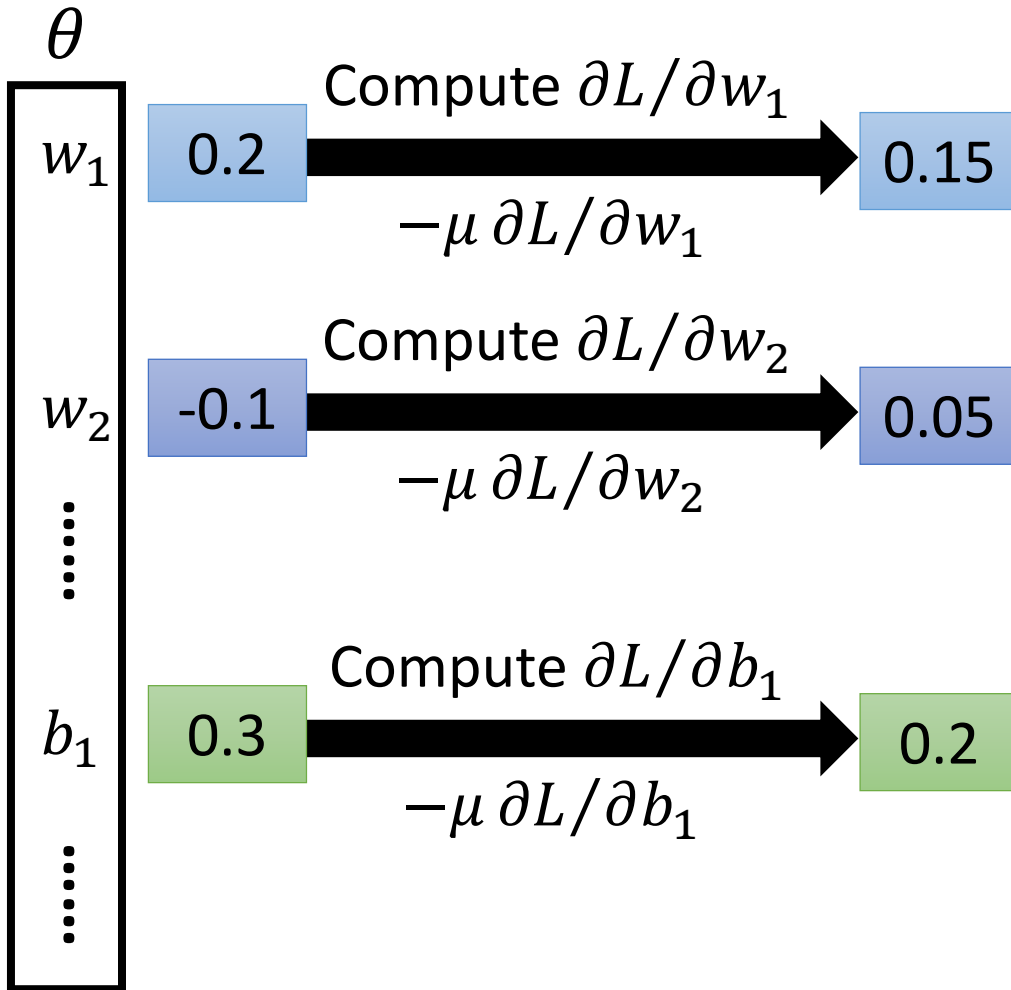
# Three Steps for Deep Learning



Deep Learning is so simple .....



# Gradient Descent

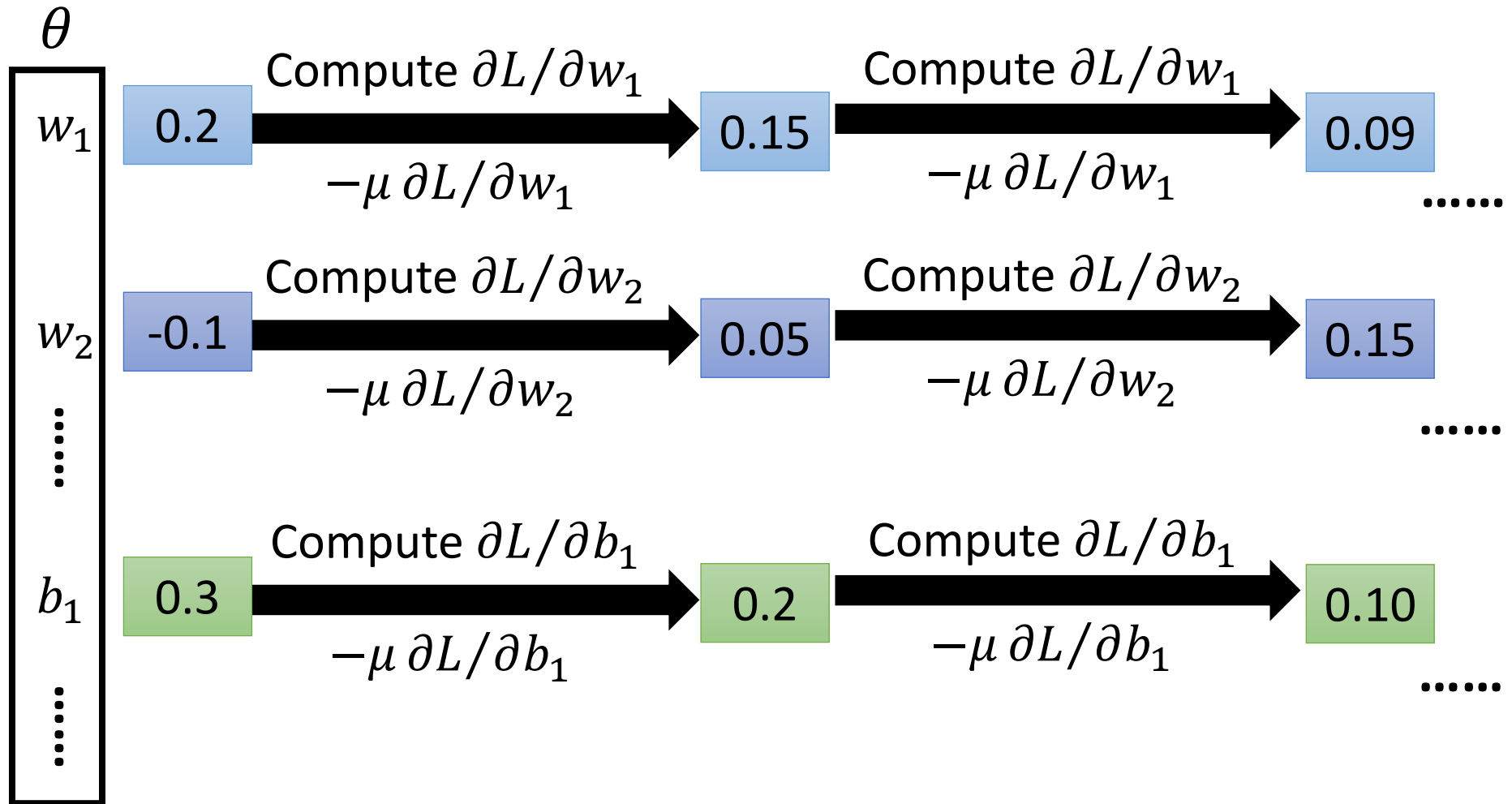


$$\nabla L = \begin{bmatrix} \frac{\partial L}{\partial w_1} \\ \frac{\partial L}{\partial w_2} \\ \vdots \\ \frac{\partial L}{\partial b_1} \\ \vdots \end{bmatrix}$$

gradient



# Gradient Descent

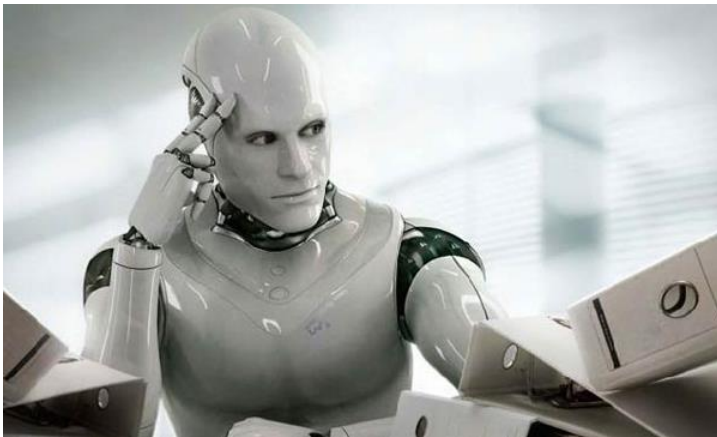


# Gradient Descent

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



theano

Caffe



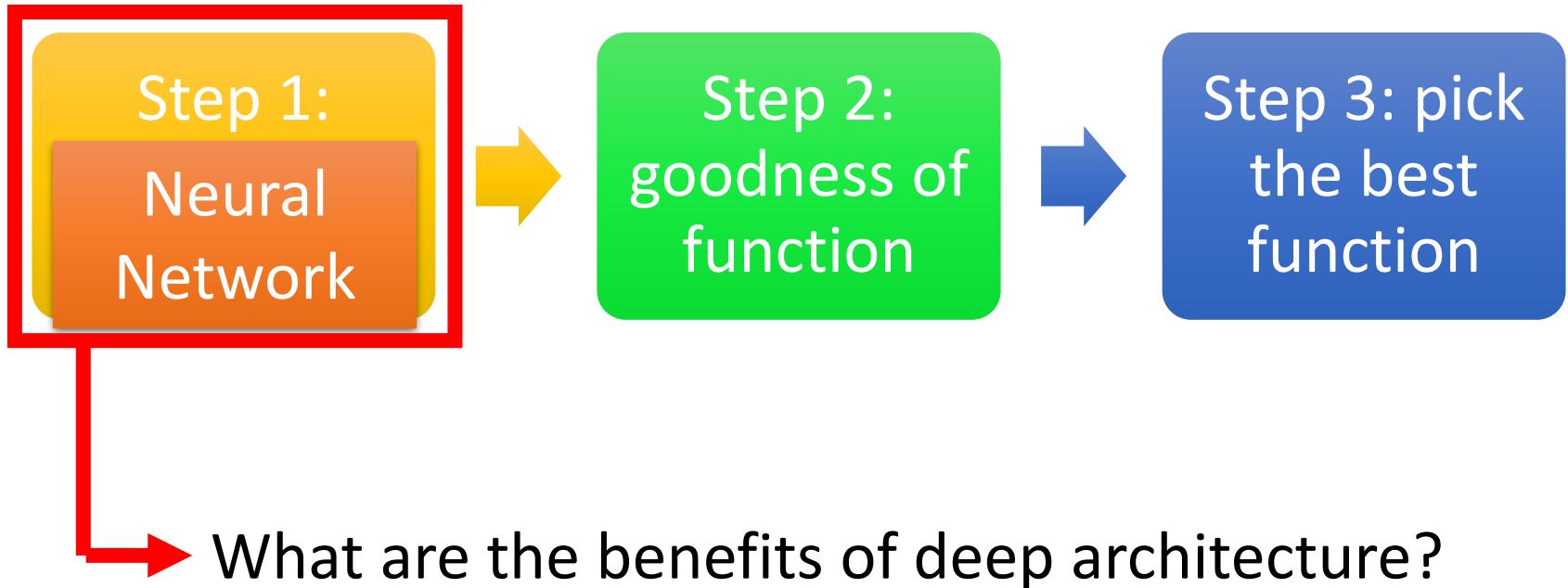
libdnn

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

# Concluding Remarks



# Deeper is Better?

Layer X Size	Word Error Rate (%)
1 X 2k	24.2
2 X 2k	20.4
3 X 2k	18.4
4 X 2k	17.8
5 X 2k	17.2
7 X 2k	17.1

Not surprised, more parameters, better performance

Seide, Frank, Gang Li, and Dong Yu. "Conversational Speech Transcription Using Context-Dependent Deep Neural Networks." *Interspeech*. 2011.

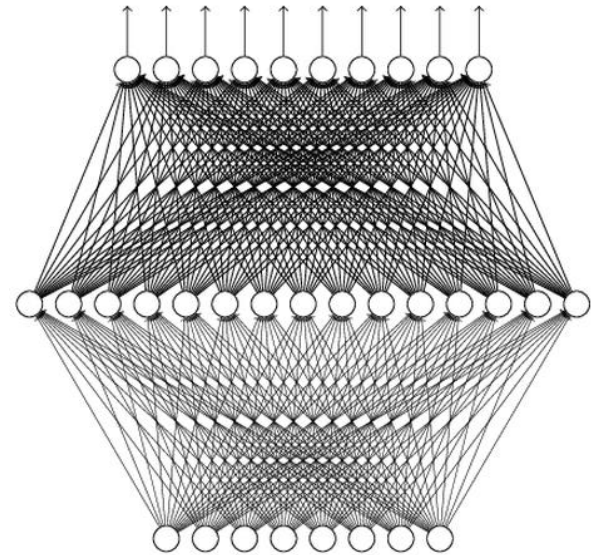
# Universality Theorem

Any continuous function  $f$

$$f : R^N \rightarrow R^M$$

Can be realized by a network  
with one hidden layer

(given **enough** hidden  
neurons)



Reference for the reason:

<http://neuralnetworksanddeeplearning.com/chap4.html>

Why “Deep” neural network not “Fat” neural network?

(next lecture)

# “深度學習深度學習”

- My Course: Machine learning and having it deep and structured
  - [http://speech.ee.ntu.edu.tw/~tlkagk/courses\\_MLSD15\\_2.html](http://speech.ee.ntu.edu.tw/~tlkagk/courses_MLSD15_2.html)
  - 6 hour version: [http://www.slideshare.net/tw\\_dsconf/ss-62245351](http://www.slideshare.net/tw_dsconf/ss-62245351)
- “Neural Networks and Deep Learning”
  - written by Michael Nielsen
  - <http://neuralnetworksanddeeplearning.com/>
- “Deep Learning”
  - written by Yoshua Bengio, Ian J. Goodfellow and Aaron Courville
  - <http://www.deeplearningbook.org>

# Acknowledgment

- 感謝 Victor Chen 發現投影片上的打字錯誤