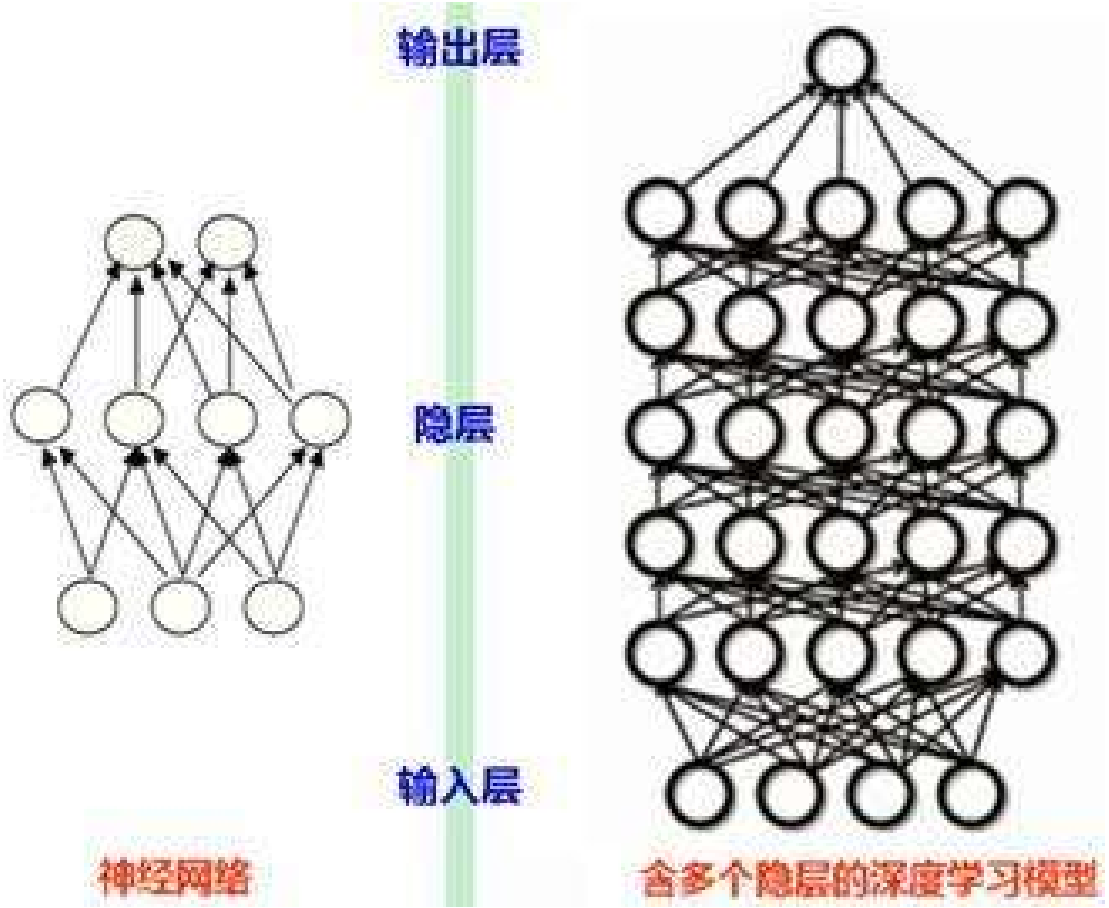
深度学习导论
(Deep learning)

代启国

大连民族大学
计算机科学与技术系

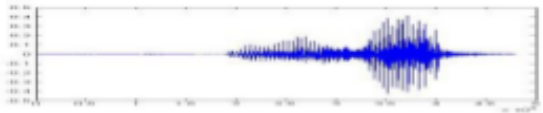
2018 年 12 月 17 日

什么是深度学习?



机器学习任务

- 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)

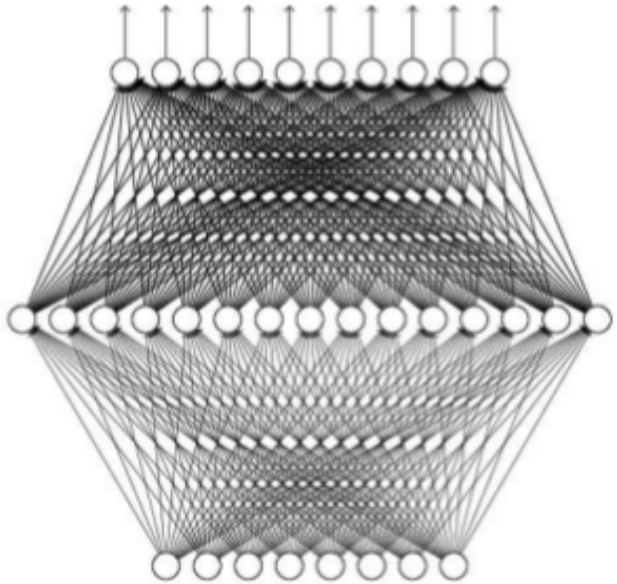
单隐层神经网络

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://neuralnetworksanddeplearning.com/chap4.html

只要隐层节点数目足够，单隐层网络可以拟合任何复杂连续函数

为什么要有“深度”？

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.

为什么要有“深度”？

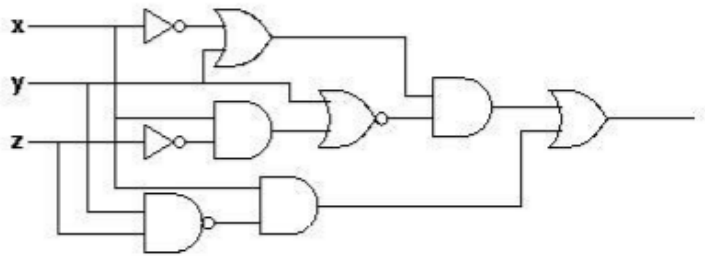
Layer X Size	Word Error Rate (%)	Layer X Size	Word Error Rate (%)
1 X 2k	24.2		
2 X 2k	20.4		
3 X 2k	18.4		
4 X 2k	17.8		
5 X 2k	17.2	1 X 3772	22.5
7 X 2k	17.1	1 X 4634	22.6
		1 X 16k	22.1

Why?

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

为什么要有“深度”？

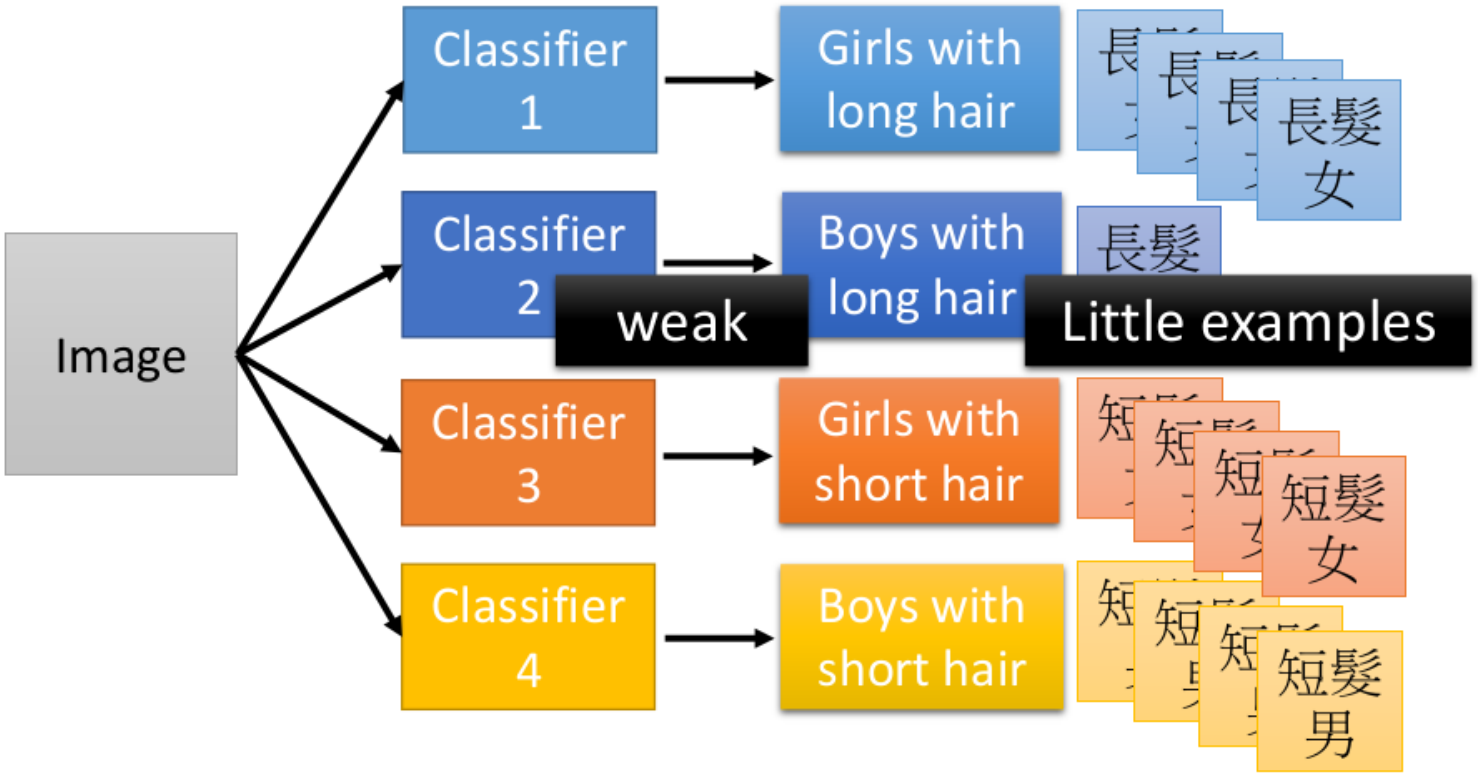
Analogy



Logic circuits	Neural network
• Logic circuits consists of **gates**	• Neural network consists of **neurons**
• **A two layers of logic gates** can represent **any Boolean function.**	• **A hidden layer network** can represent **any continuous function.**
• Using multiple layers of logic gates to build some functions are much simpler	• Using multiple layers of neurons to represent some functions are much simpler
→ less gates needed	→ less parameters → less data?

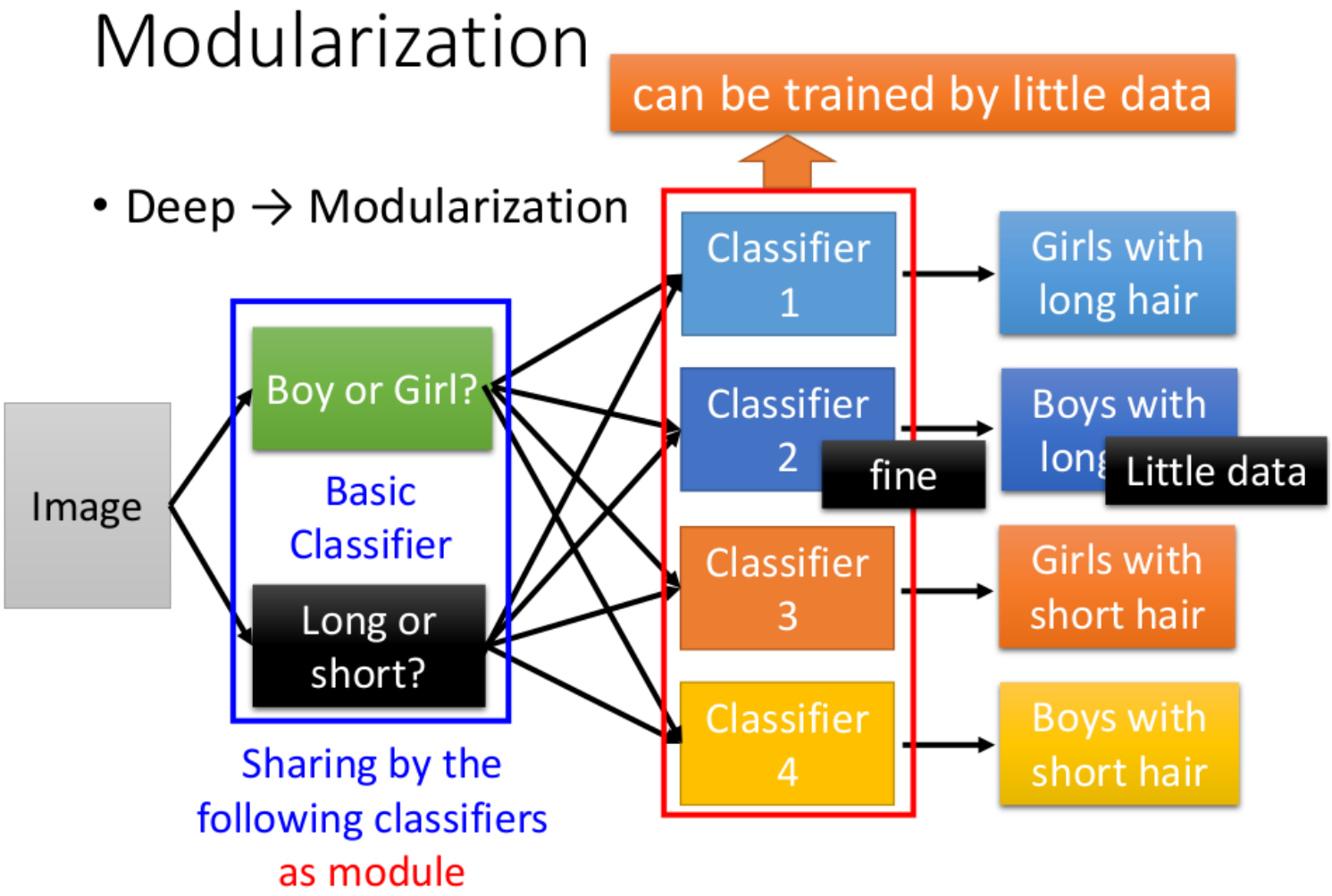
为什么要有“深度”？

深度可以有利于“模块化”



为什么要有“深度”？

深度可以有利于“模块化”

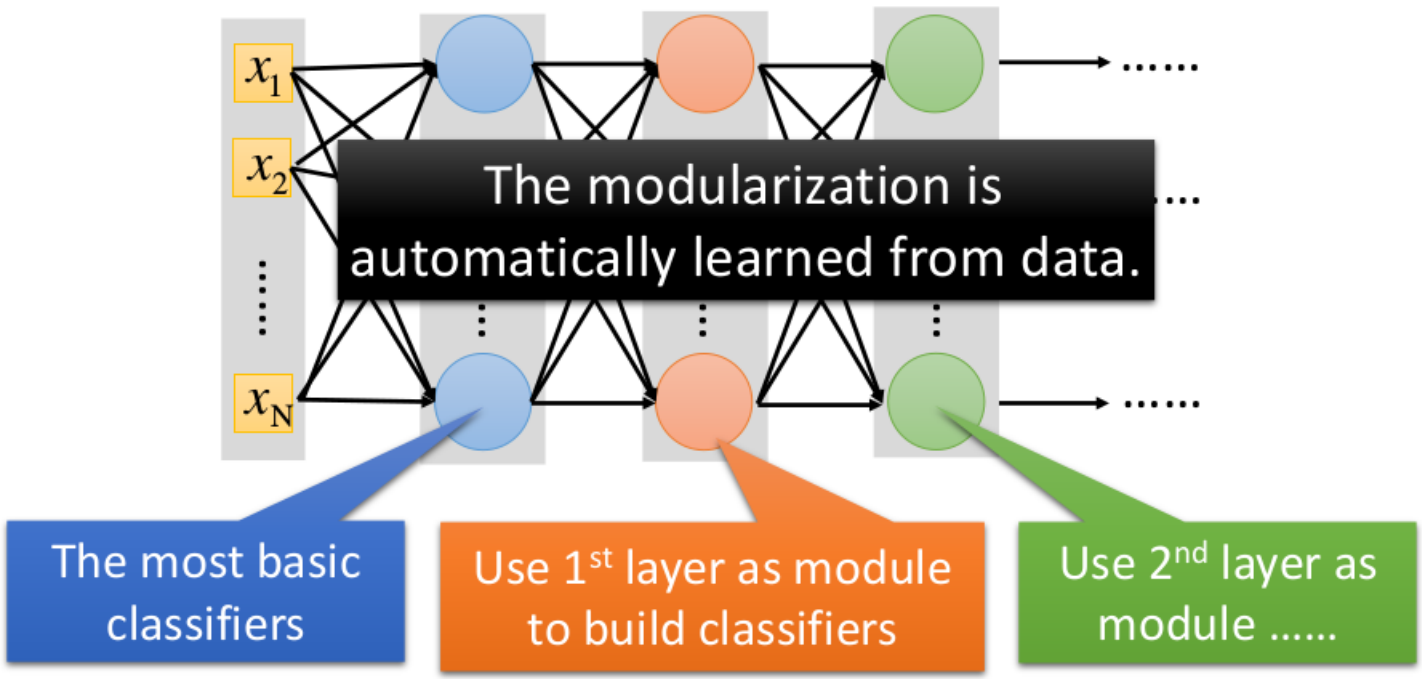


为什么要有“深度”？

深度可以有利于“模块化”

Modularization

- Deep → Modularization → Less training data?

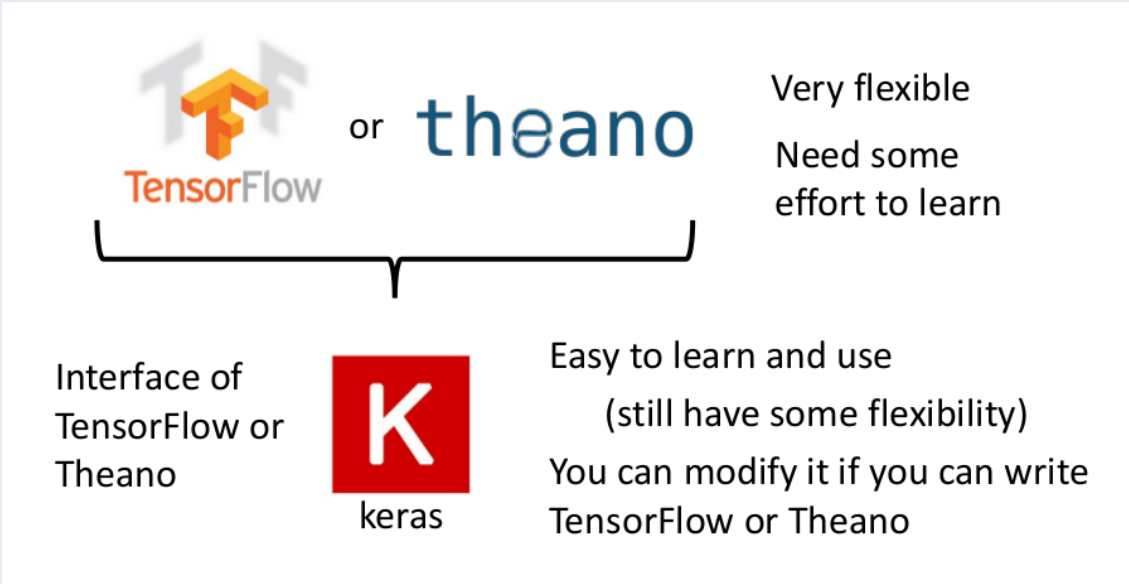


深度学习框架

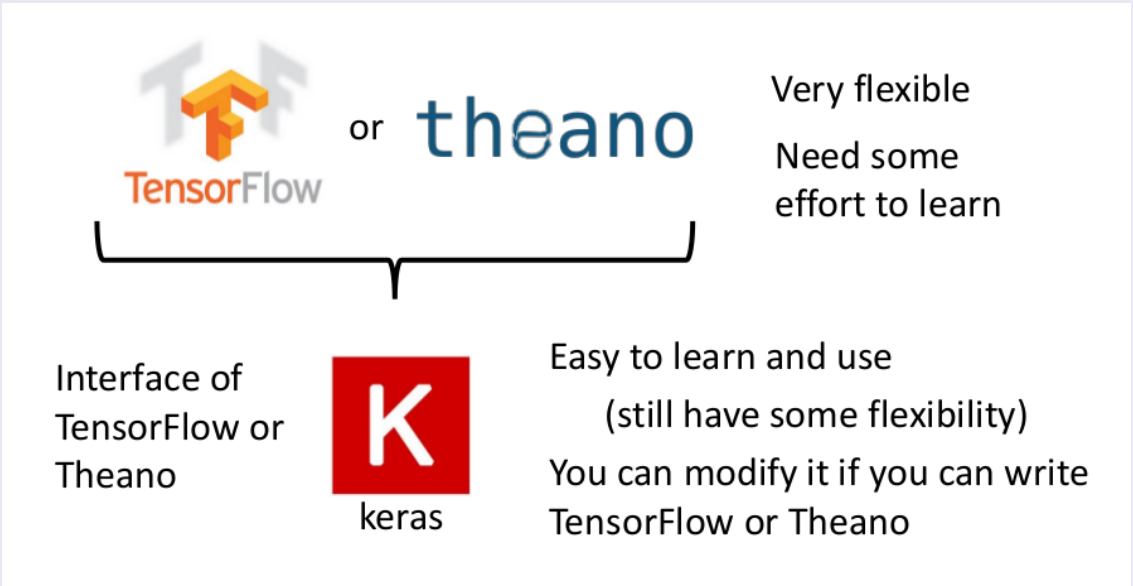
用于快速构建和训练深度学习模型的框架

Keras: 基于 Python 的深度学习库

- Keras 是一个高层神经网络 API, Keras 由纯 Python 编写而成并基 Tensorflow、 Theano 以及 CNTK 后端
- Keras 为支持快速实验而生，能够把你的 idea 迅速转换为结果
- 可用于构建多种神经网络 DNN、CNN 等
- 无缝支持 CPU 和 GPU 切换



Keras: 基于 Python 的深度学习库



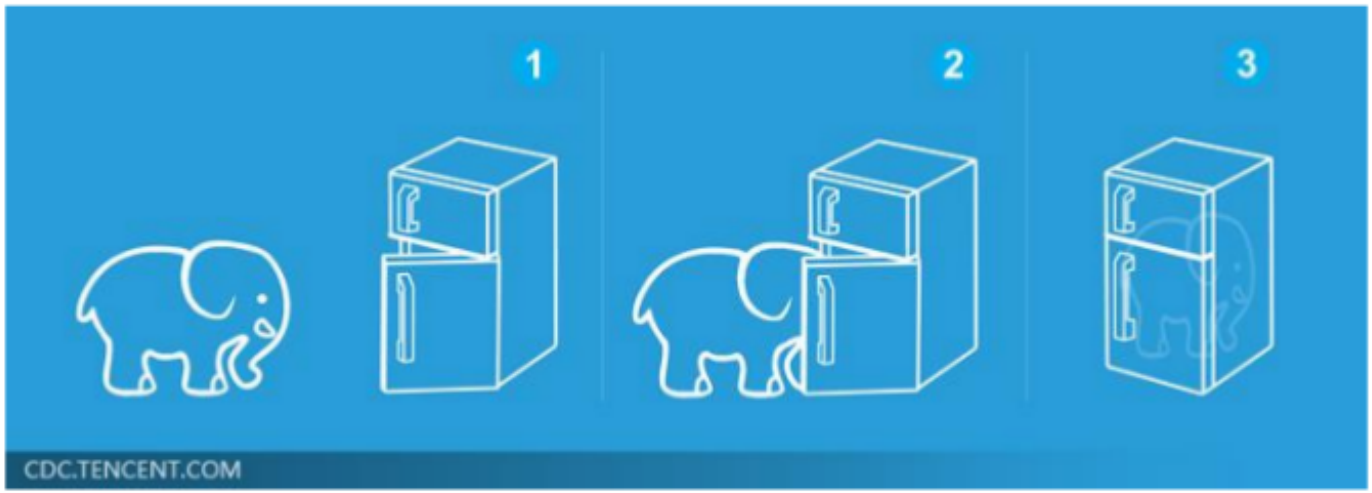
- keras 中文文档：http://keras-cn.readthedocs.io/en/latest/
- keras 英文文档：http://keras.io/

Keras: 基于 Python 的深度学习库

使用 Keras 心得

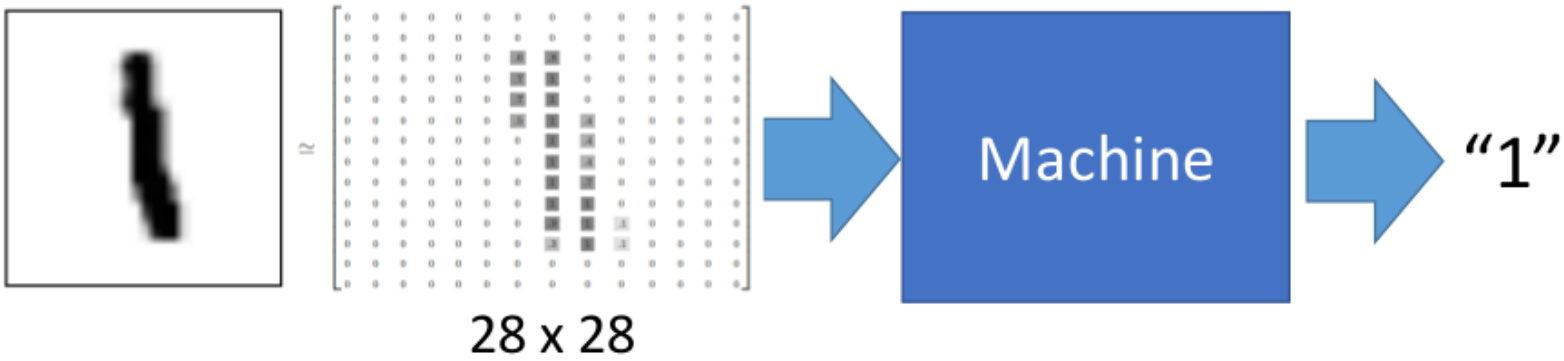
Keras 的使用方法

Deep Learning is so simple ……



Keras 的使用方法

- Handwriting Digit Recognition



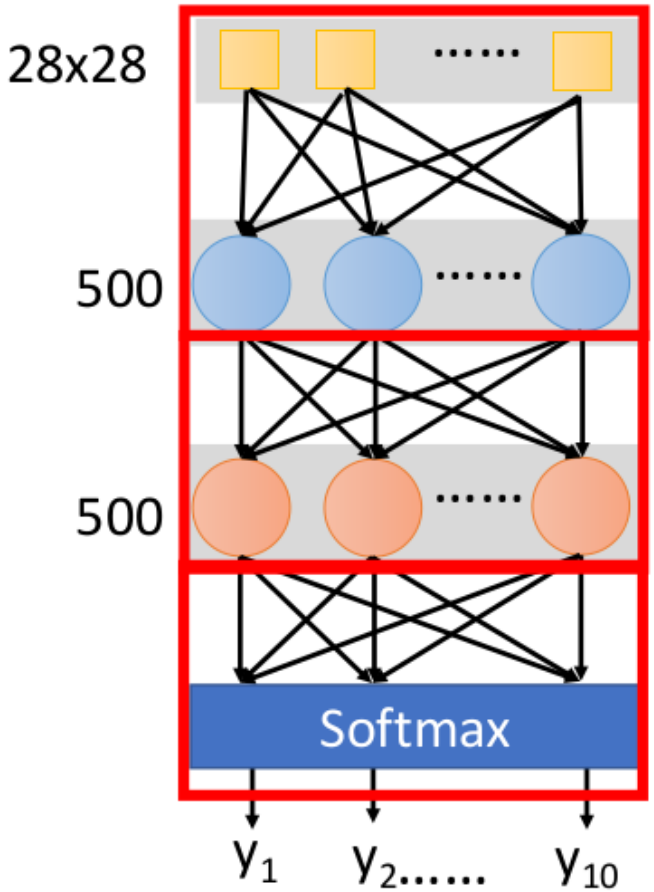
MNIST Data: http://yann.lecun.com/exdb/mnist/

“Hello world” for deep learning

Keras provides data sets loading function: http://keras.io/datasets/

Keras 的使用方法

Keras



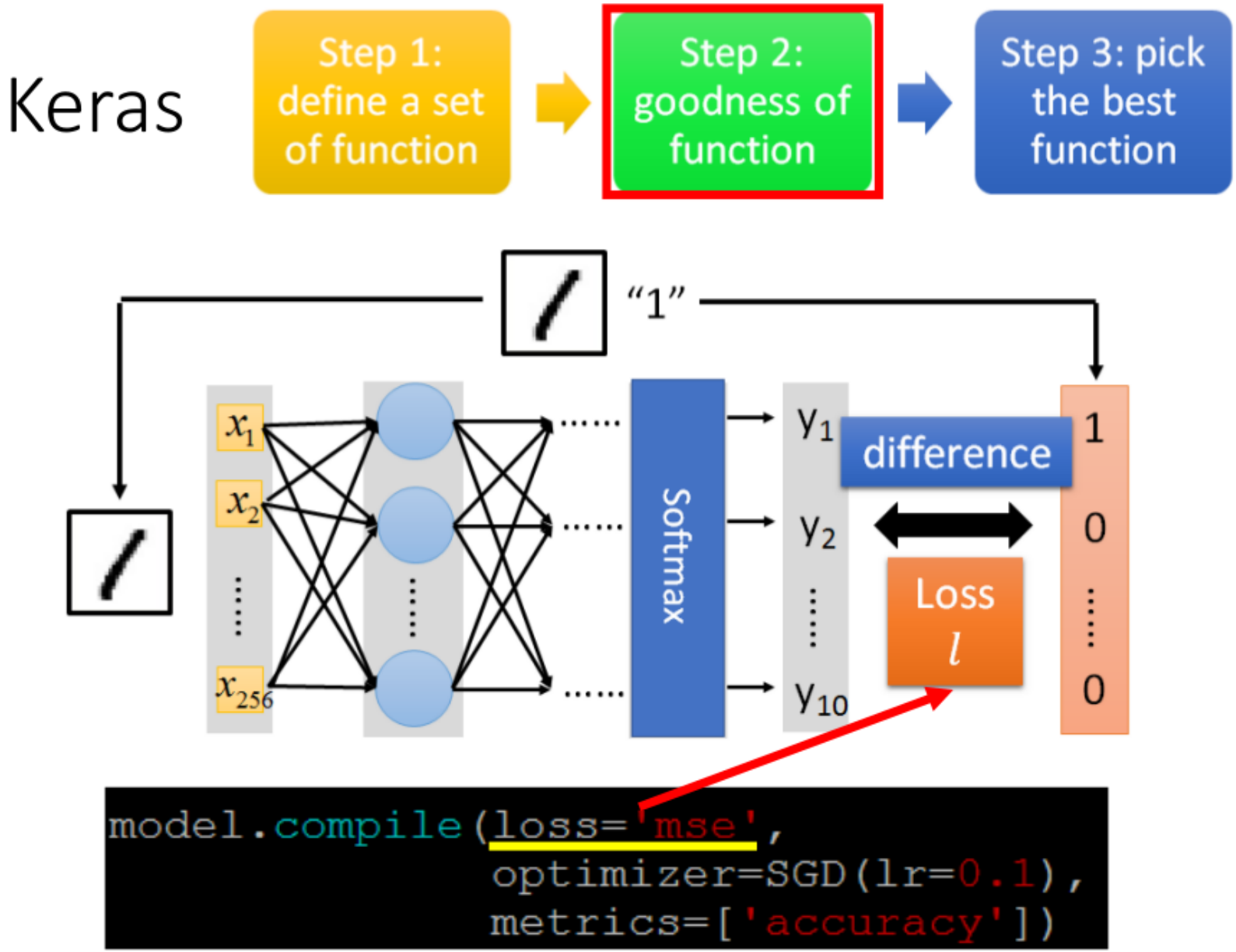

```
model = Sequential()
```

```
model.add( Dense( input_dim=28*28,
                  output_dim=500 ))
model.add( Activation('sigmoid') )
```

```
model.add( Dense( output_dim=500 ) )
model.add( Activation('sigmoid') )
```

```
model.add( Dense(output_dim=10 ) )
model.add( Activation('softmax') )
```

Keras 的使用方法


Keras
Step 1: define a set of function
Step 2: goodness of function
Step 3: pick the best function
"1"
x_1
x_2
x_{256}
Softmax
y_1
y_2
y_{10}
difference
Loss l
1
0
0
model.compile(loss='mse',
optimizer=SGD(lr=0.1),
metrics=['accuracy'])

Keras 的使用方法

Step 3.1: Configuration

```
model.compile(loss='mse',
              optimizer=SGD(lr=0.1),
              metrics=['accuracy'])
```

$$w \leftarrow w - \eta \partial L / \partial w$$

0.1

Step 3.2: Find the optimal network parameters

```
model.fit(x_train, y_train, batch_size=100, nb_epoch=20)
```

Training data (Images)

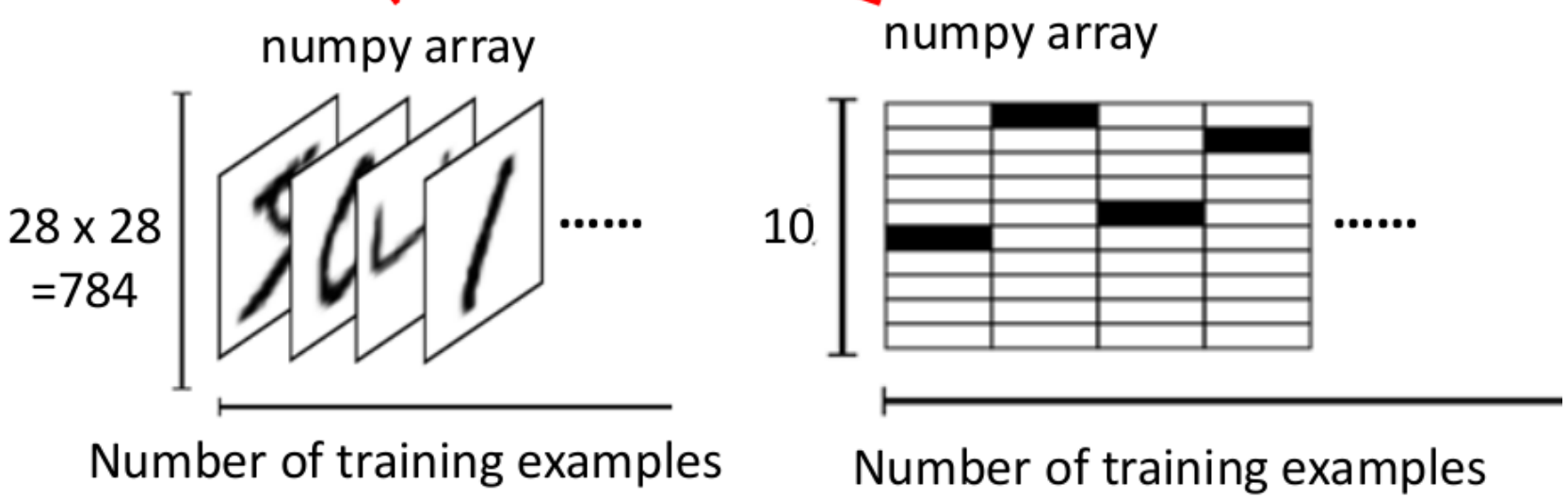
Labels (digits)

Next lecture

Keras 的使用方法

Step 3.2: Find the optimal network parameters

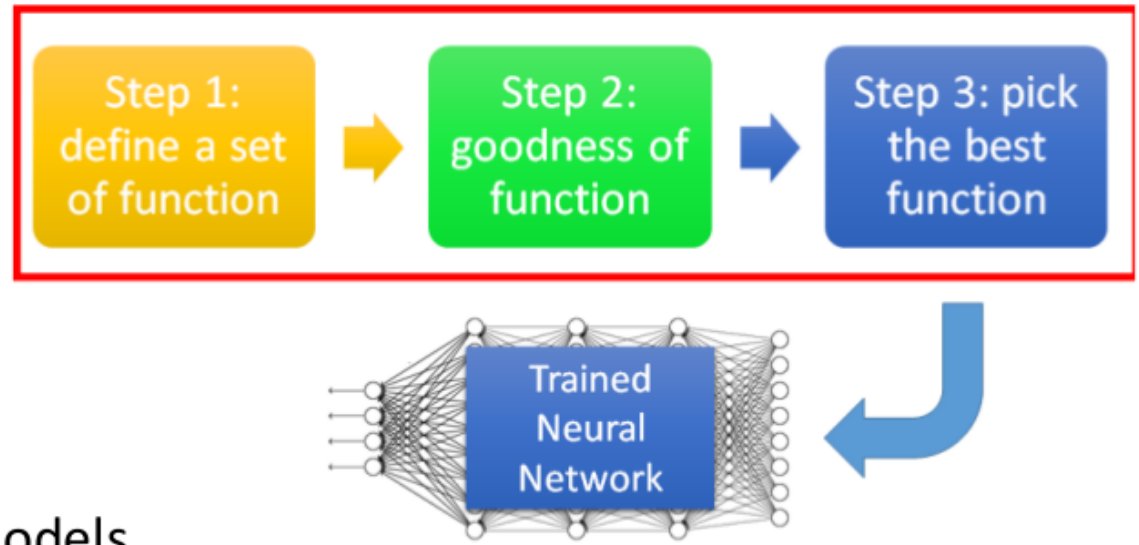
```
model.fit(x_train, y_train, batch_size=100, nb_epoch=20)
```



https://www.tensorflow.org/versions/r0.8/tutorials/mnist/beginners/index.html

Keras 的使用方法

Keras



Save and load models

http://keras.io/getting-started/faq/#how-can-i-save-a-keras-model

How to use the neural network (testing):

case 1:

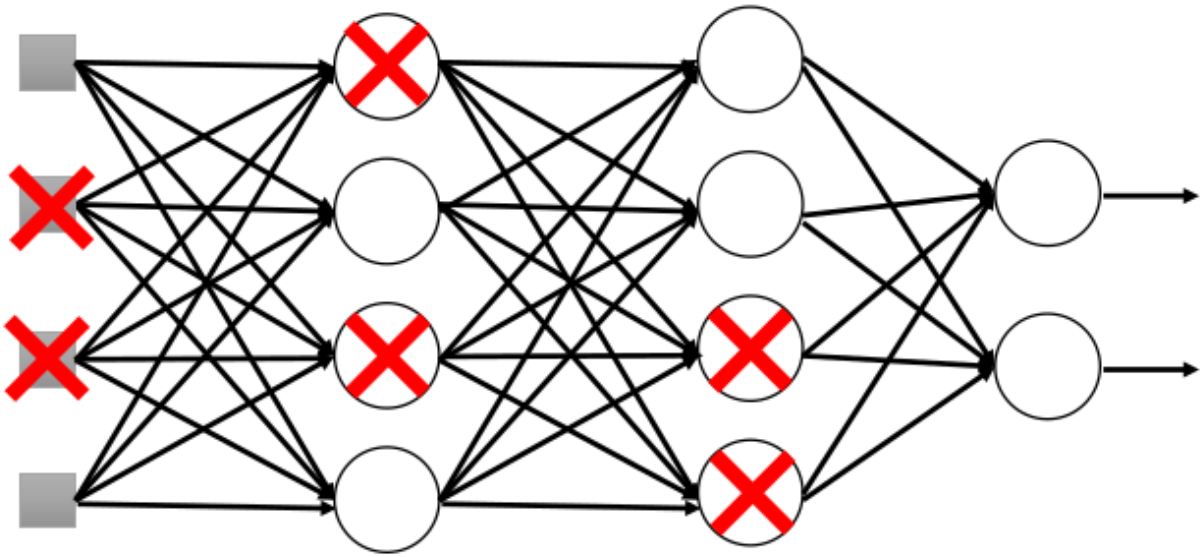
```
score = model.evaluate(x_test,y_test)
print('Total loss on Testing Set:', score[0])
print('Accuracy of Testing Set:', score[1])
```

case 2:

```
result = model.predict(x_test)
```

Dropout

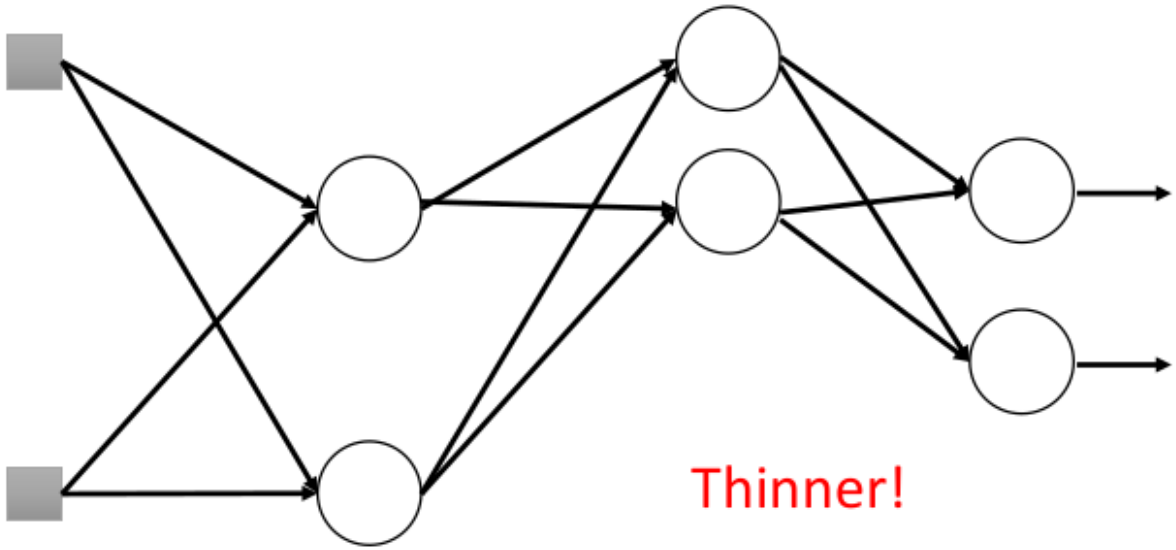
Training:



- **Each time before updating the parameters**
  - Each neuron has p% to dropout

Dropout

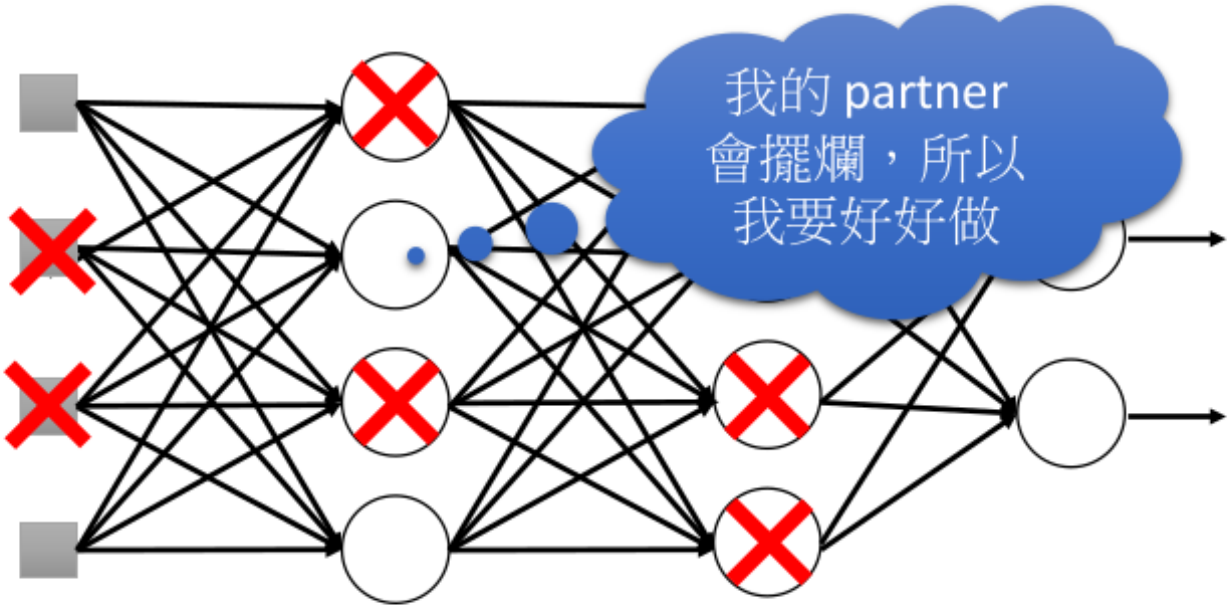
Training:



- **Each time before updating the parameters**
  - Each neuron has p% to dropout
    - **The structure of the network is changed.**
  - Using the new network for training

For each mini-batch, we resample the dropout neurons

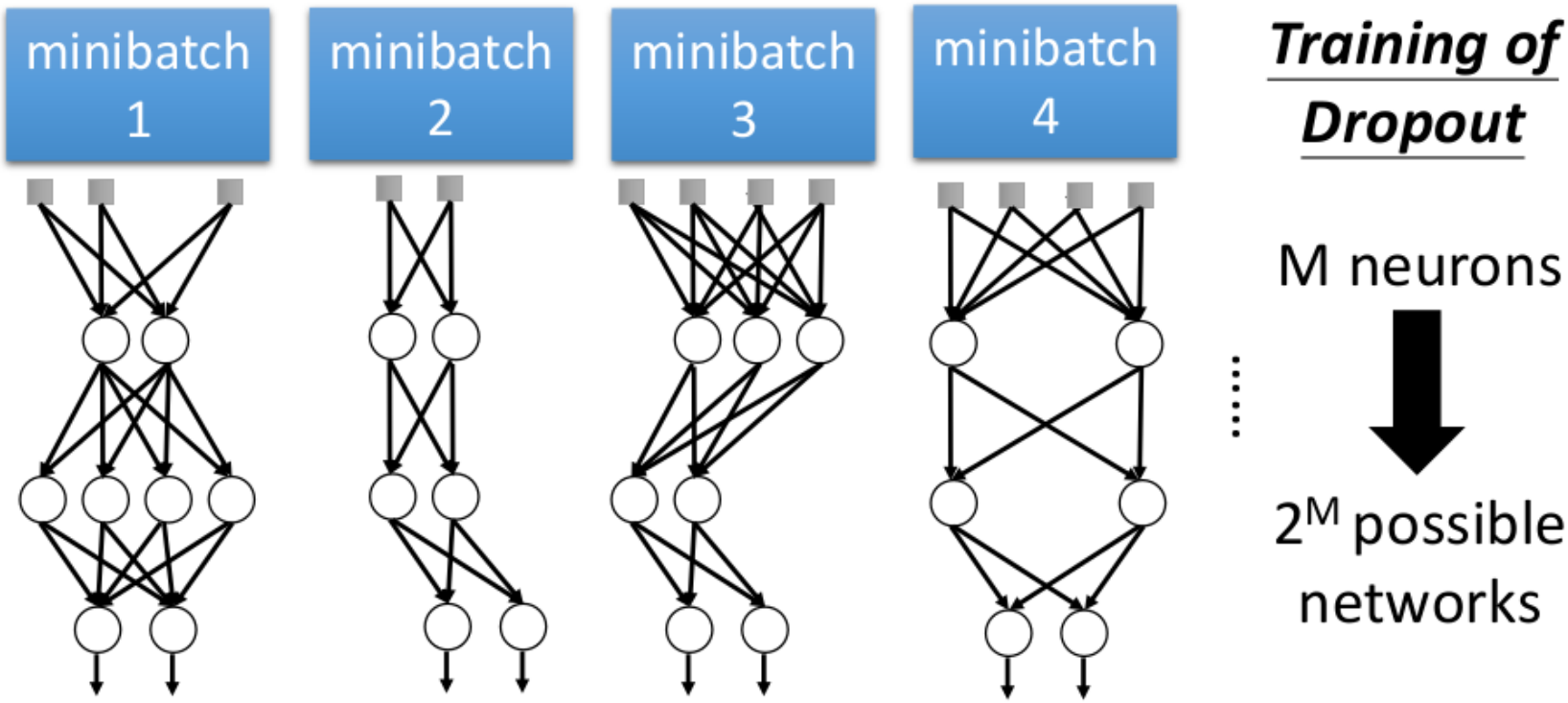
Dropout - Intuitive Reason



- When teams up, if everyone expect the partner will do the work, nothing will be done finally.
- However, if you know your partner will dropout, you will do better.
- When testing, no one dropout actually, so obtaining good results eventually.

Dropout 技术

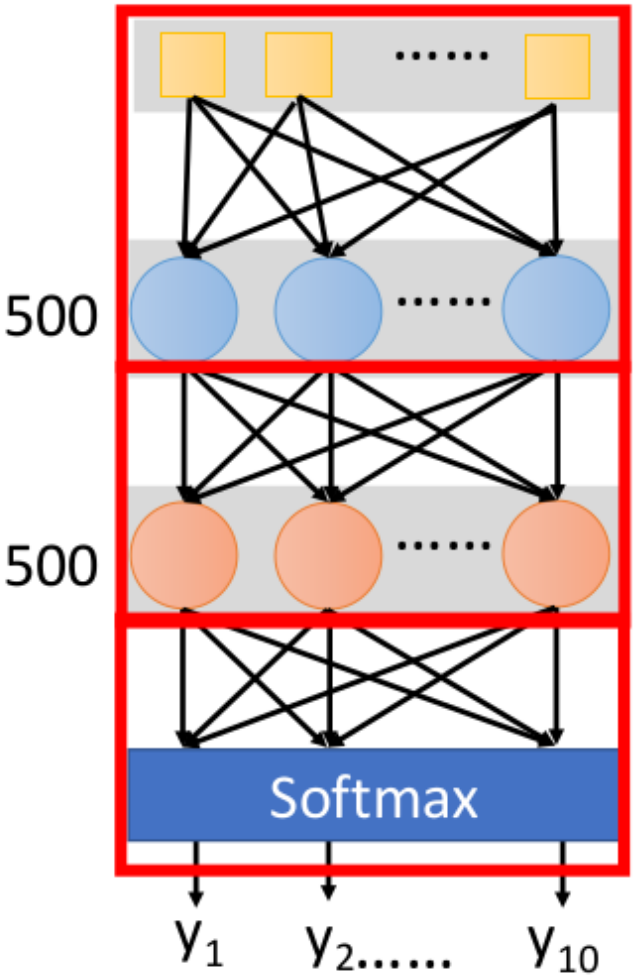
Dropout is a kind of ensemble.



- Using one mini-batch to train one network
- Some parameters in the network are shared

Dropout 技术

Let's try it




```
model = Sequential()
```

```
model.add( Dense( input_dim=28*28,
                  output_dim=500 ))
model.add( Activation('sigmoid') )
```

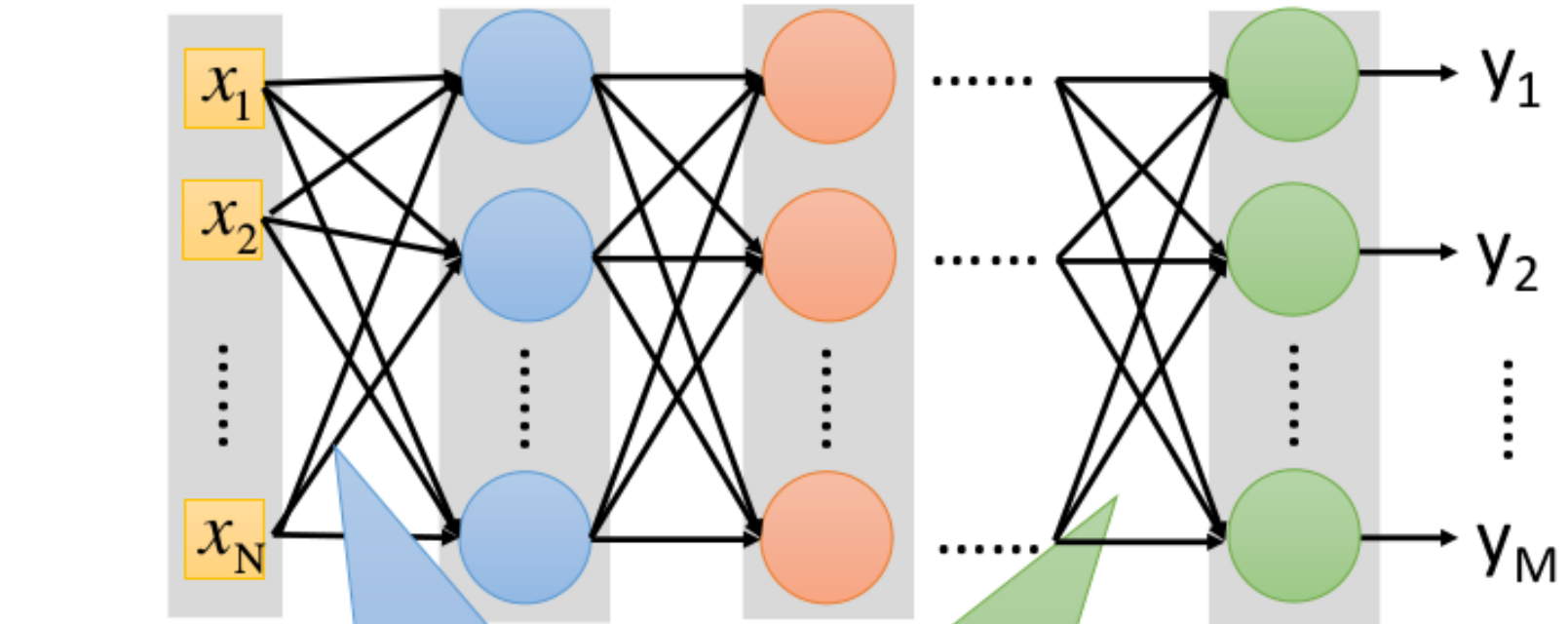
model.add(dropout(0.8))

```
model.add( Dense( output_dim=500 ) )
model.add( Activation('sigmoid') )
```

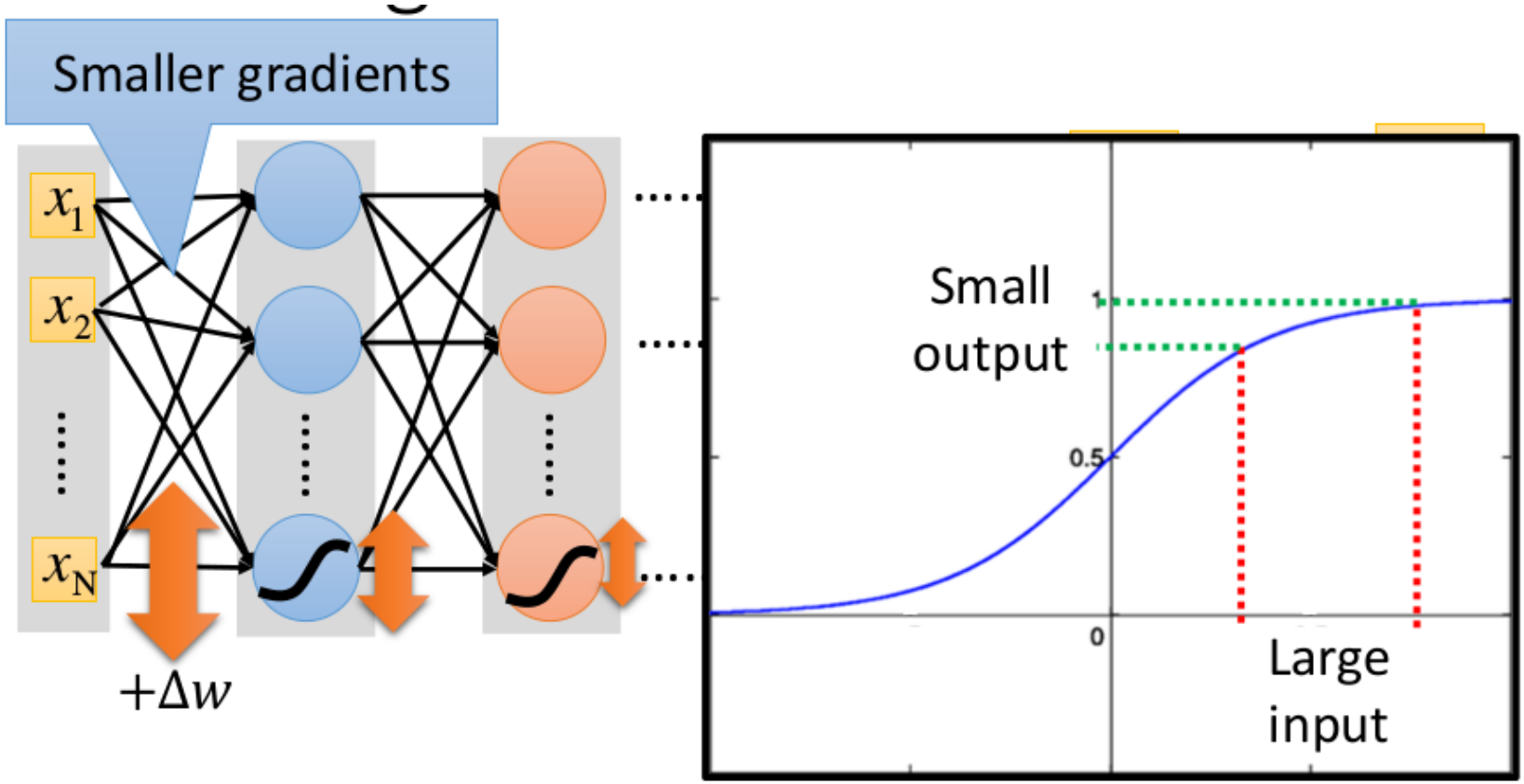
model.add(dropout(0.8))

```
model.add( Dense(output_dim=10 ) )
model.add( Activation('softmax') )
```

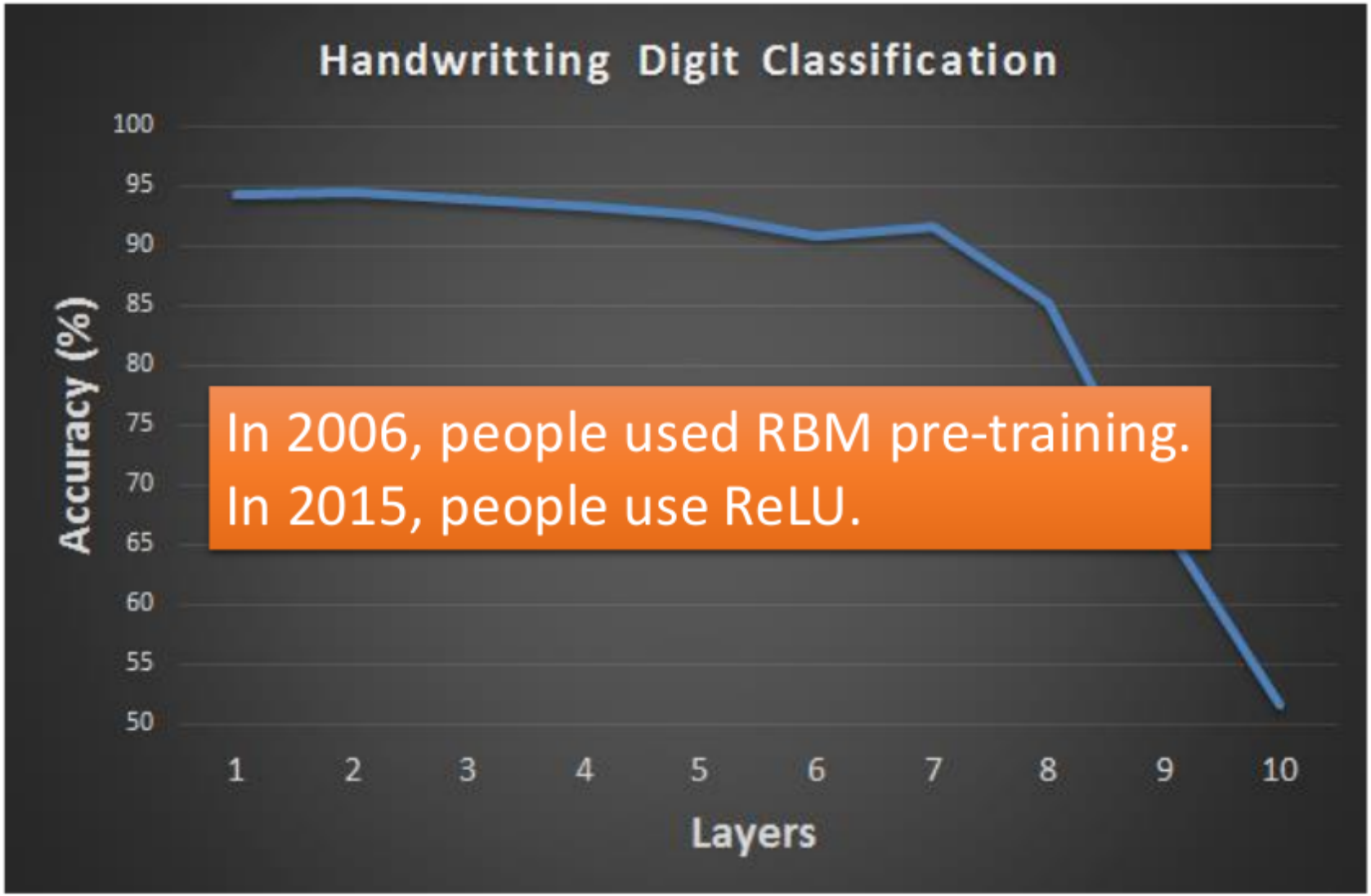
梯度消失


x_1
x_2
x_N
y_1
y_2
y_M
Smaller gradients
Learn very slow
Almost random
Larger gradients
Learn very fast
Already converge

梯度消失

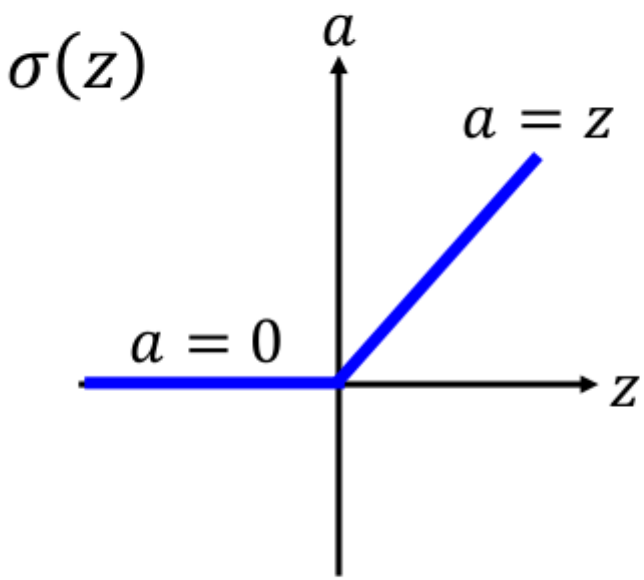

Smaller gradients
x_1
x_2
x_N
$+\Delta w$
Small output
Large input
1
0.5
0

梯度消失


Handwritting Digit Classification
Accuracy (%)
100
95
90
85
80
75
70
65
60
55
50
1
2
3
4
5
6
7
8
9
10
Layers
In 2006, people used RBM pre-training.
In 2015, people use ReLU.

梯度消失

- Rectified Linear Unit (ReLU)



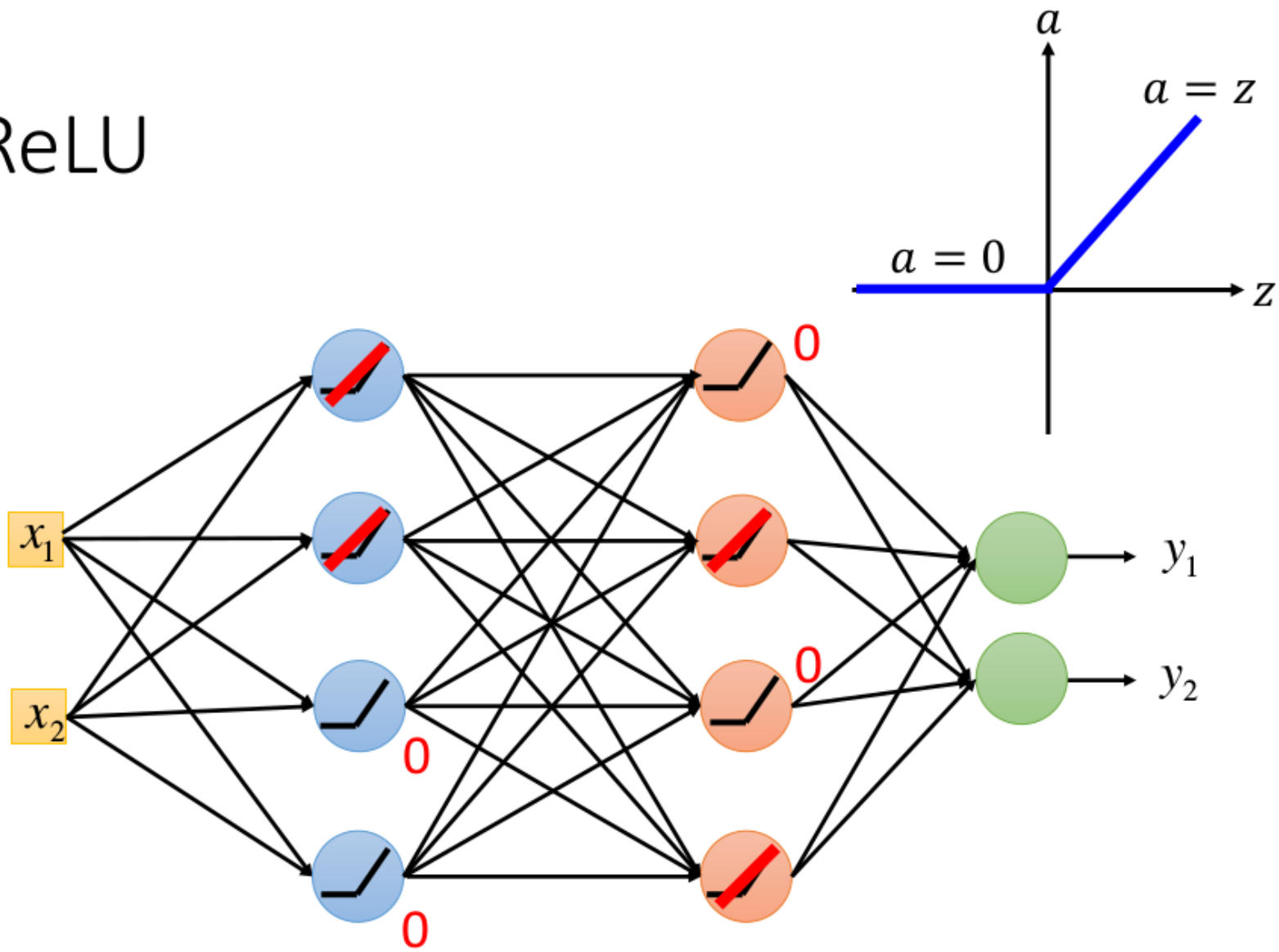
[Xavier Glorot, AISTATS'11]
[Andrew L. Maas, ICML'13]
[Kaiming He, arXiv'15]

Reason:

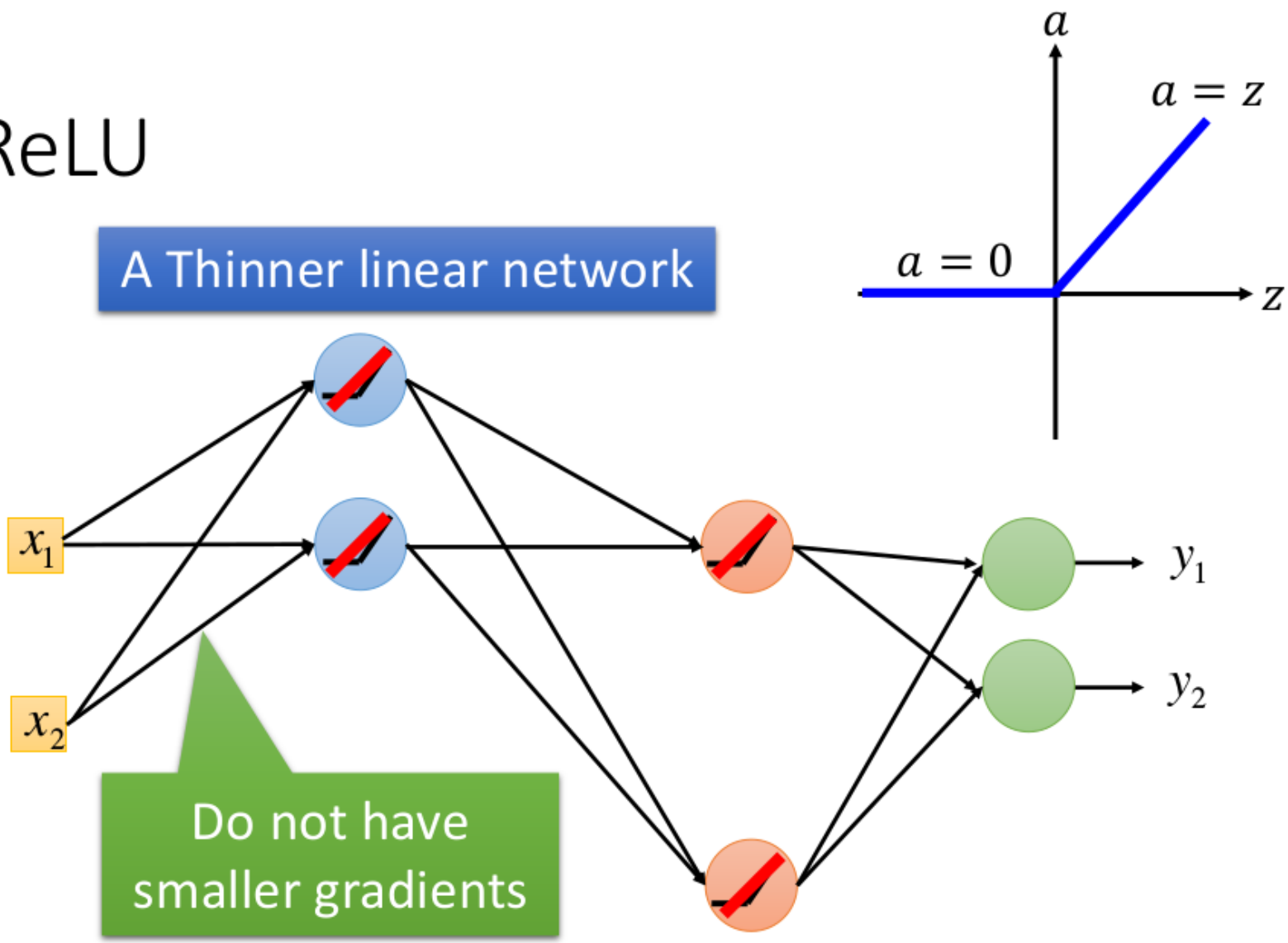
1. Fast to compute
2. Biological reason
3. Infinite sigmoid with different biases
4. Vanishing gradient problem

梯度消失

ReLU



梯度消失


ReLU
A Thinner linear network
a
a = z
a = 0
z
x1
x2
y1
y2
Do not have
smaller gradients

本 PPT 资料主要源于台湾大学

Deep Learning Tutorial

李宏毅

Hung-yi Lee