Introduction to Deep Learning and Tensorflow

Veerasak Kritsanapraphan
Software Park Thailand

Credit Slide : Hung-yi Lee

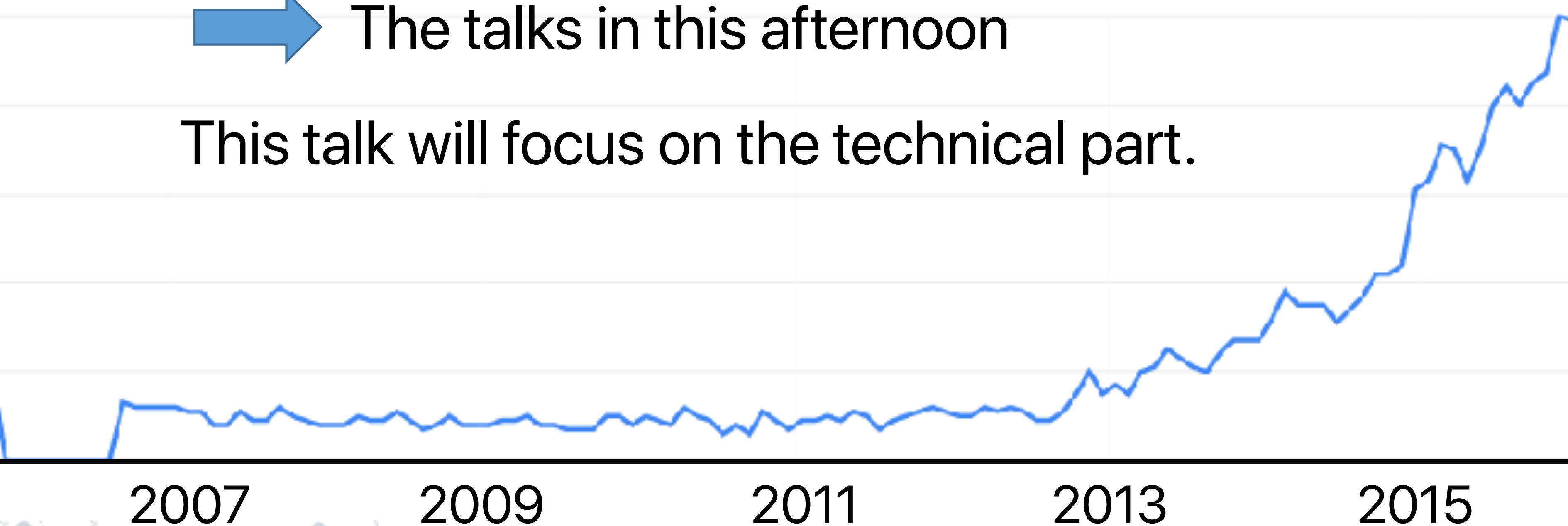
Deep learning
attracts lots of attention.

- Google Trends

Deep learning obtains many exciting results.

→ The talks in this afternoon

This talk will focus on the technical part.



Outline

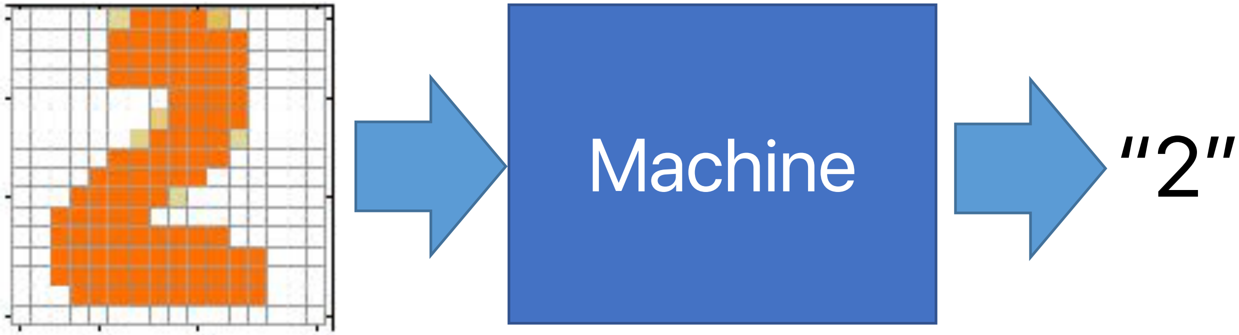
- Part I: Introduction of Deep Learning
- Part II: Why Deep?
- Part III: Tips for Training Deep Neural Network
- Part IV: Neural Network with Memory

Part I:

Introduction of Deep Learning

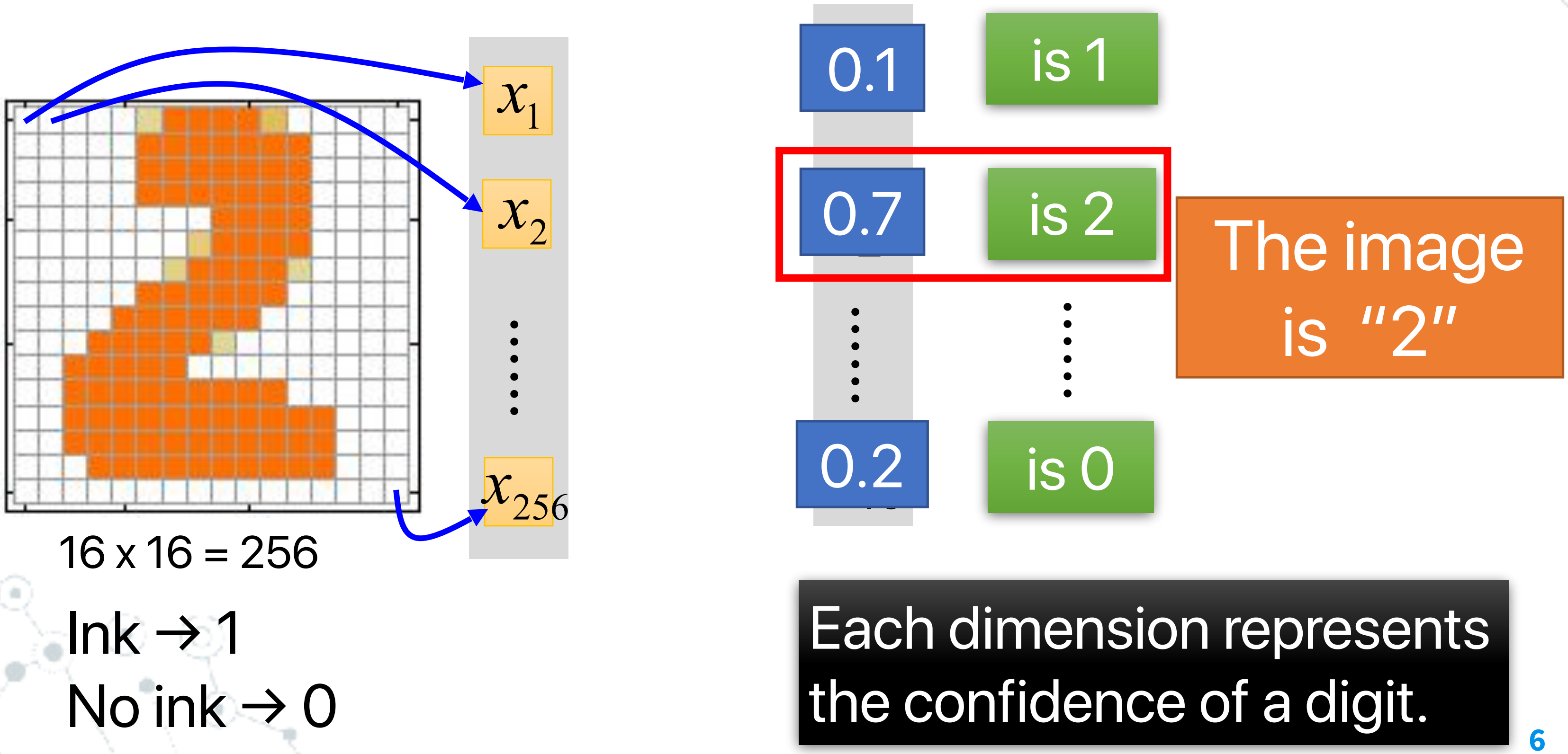
Example Application

◎ Handwriting Digit Recognition



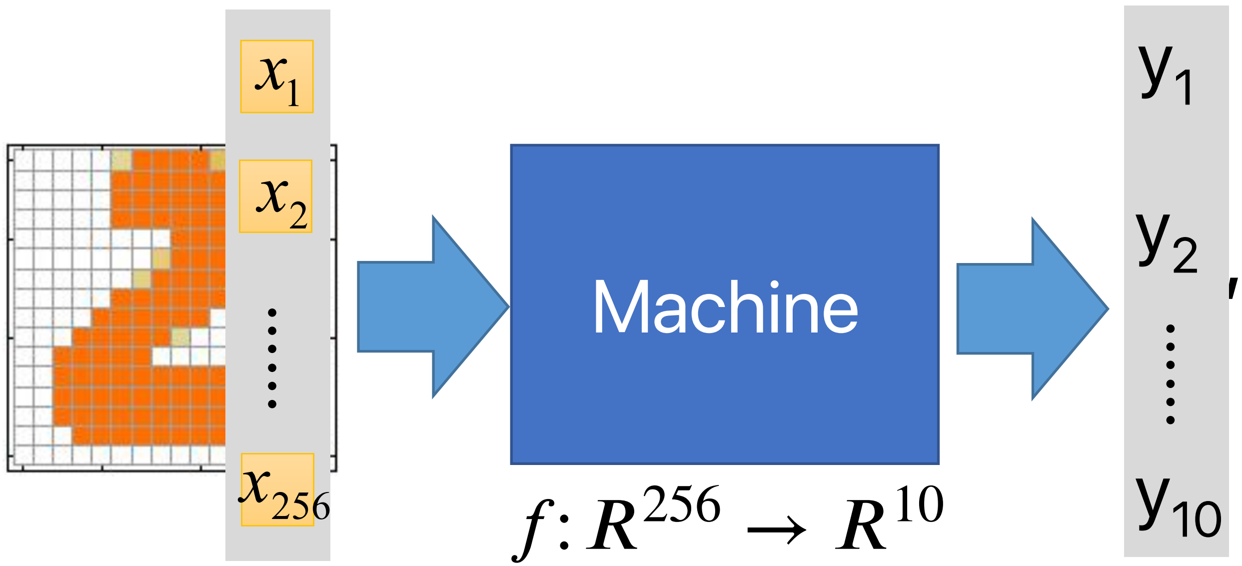
Handwriting Digit Recognition

◎ Input



Example Application

◎ Handwriting Digit Recognition

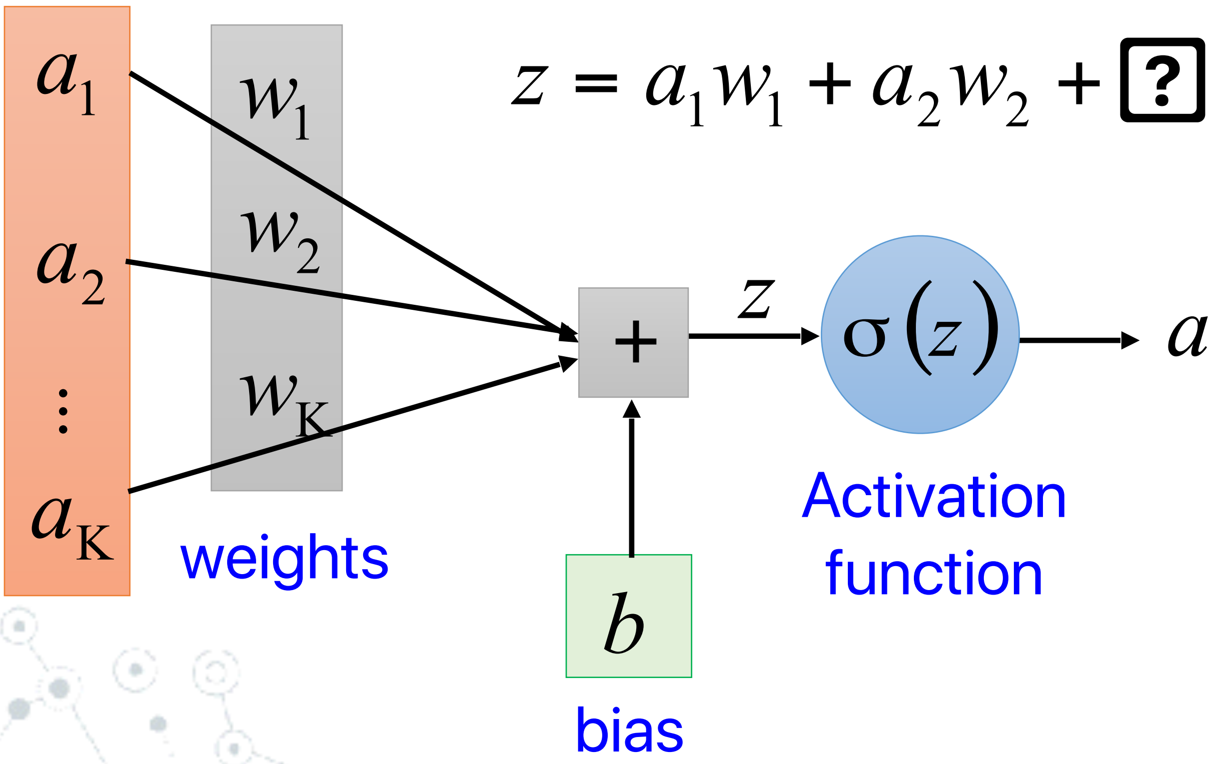


In deep learning, the function is represented by neural network

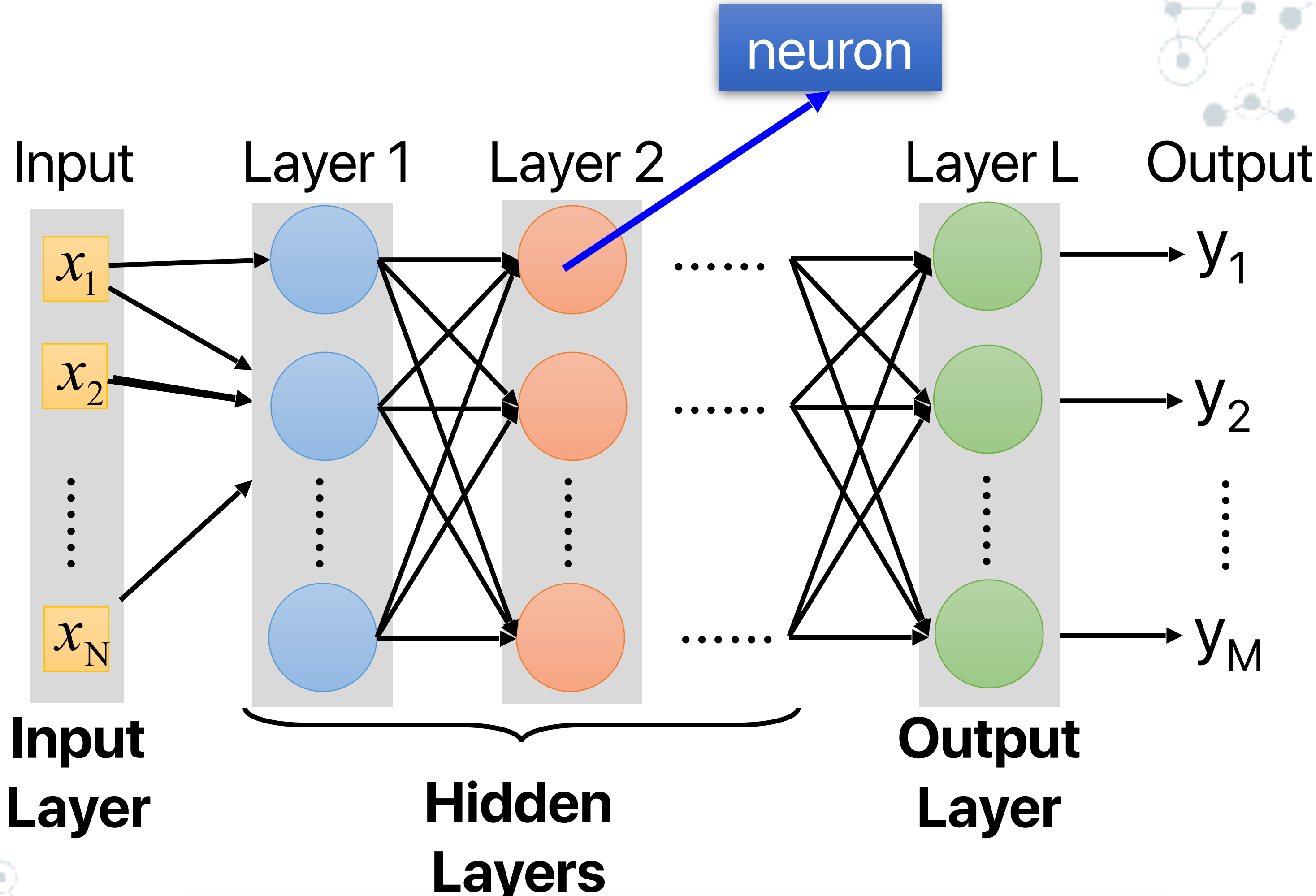
Element of Neural Network

Neuron $f: R^K \rightarrow R$

$$z = a_1 w_1 + a_2 w_2 + \boxed{?} + a_K w_K + b$$

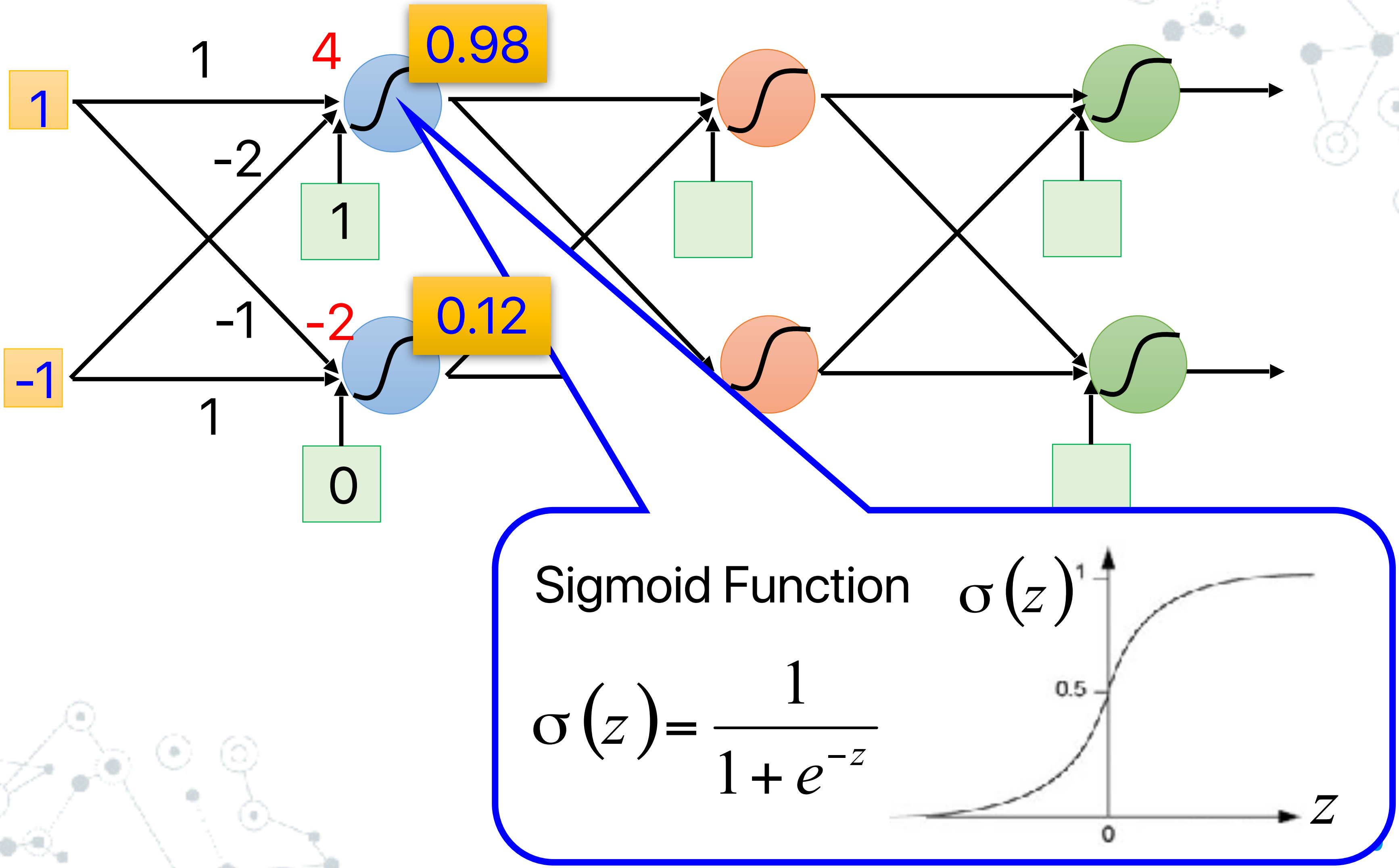


Neural Network



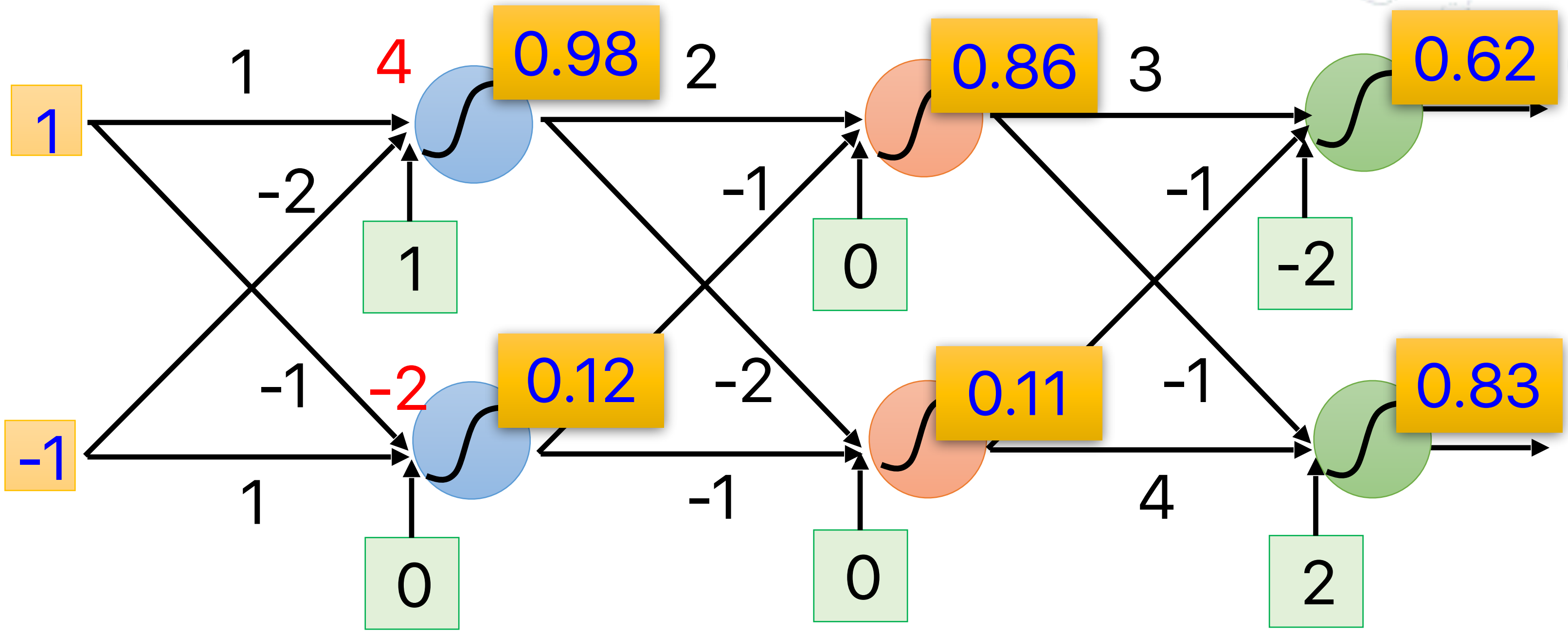
Deep means many hidden layers

Example of Neural Network

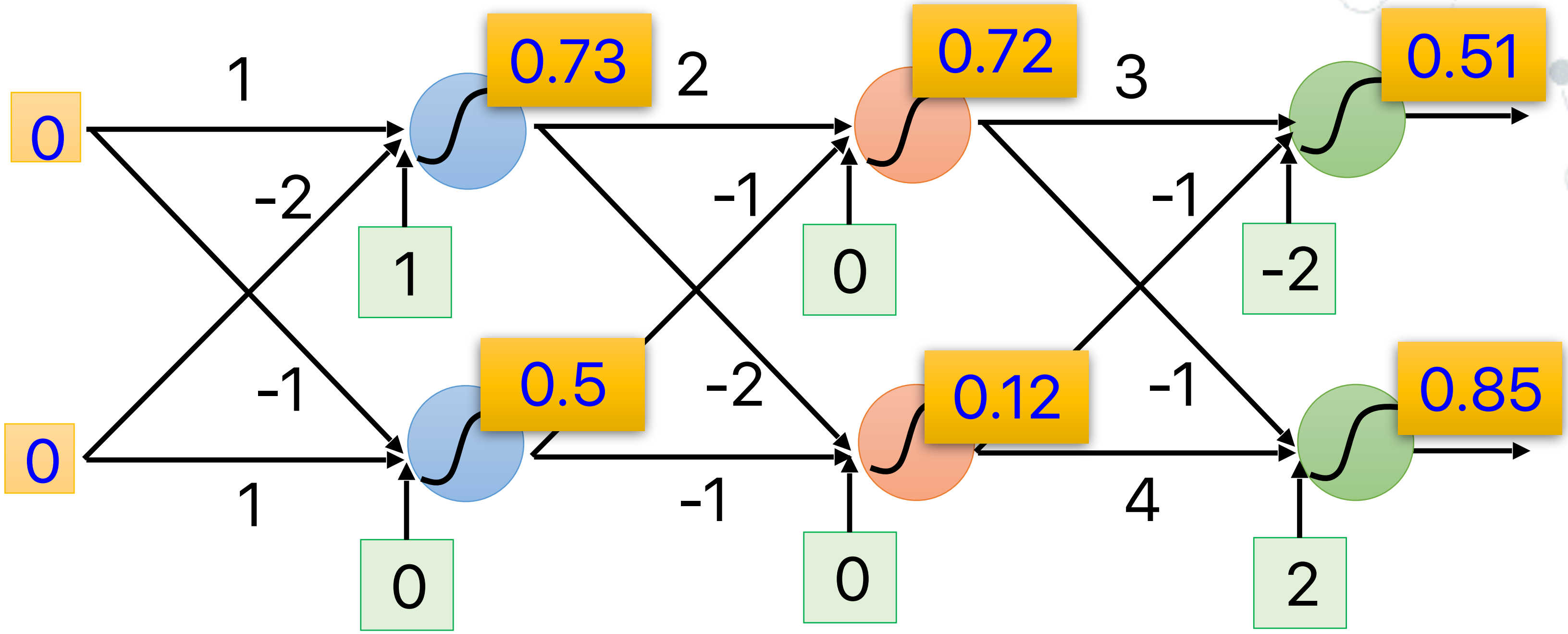


$$\sigma(z) = \frac{1}{1+e^{-z}}$$

Example of Neural Network



Example of Neural Network

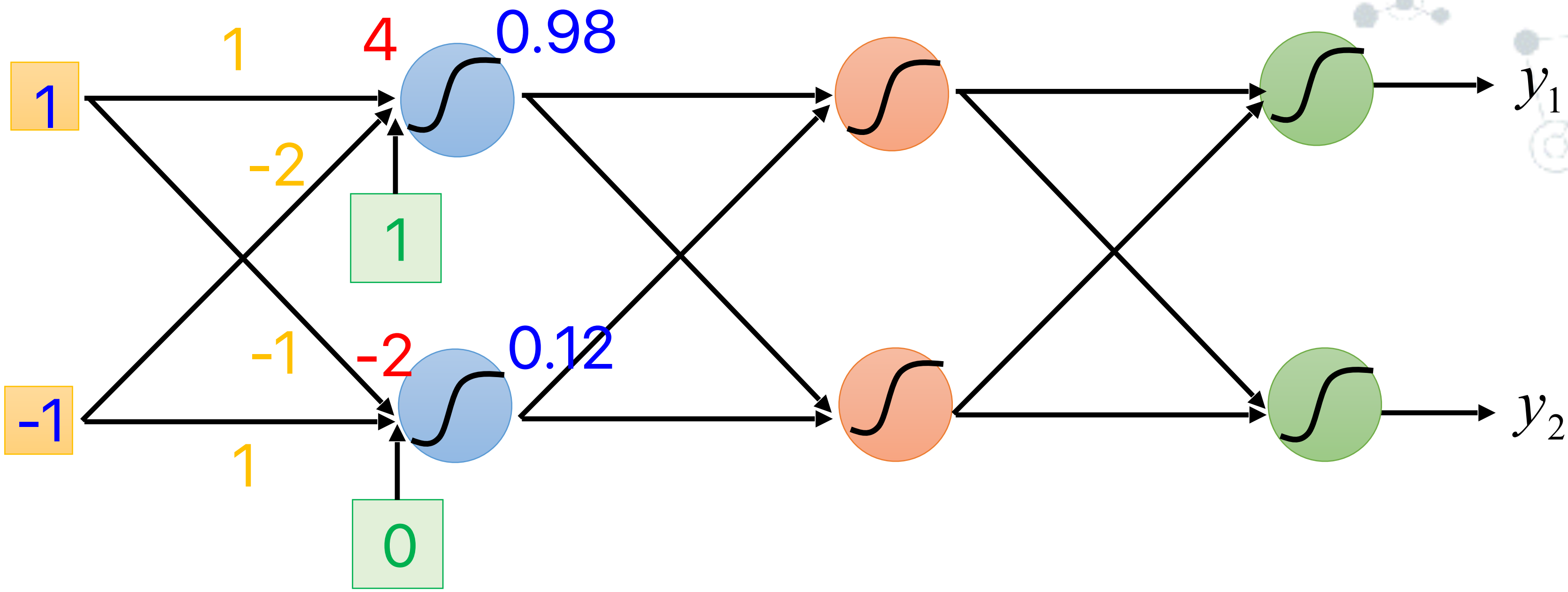


$f: R^2 \rightarrow R^2$

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

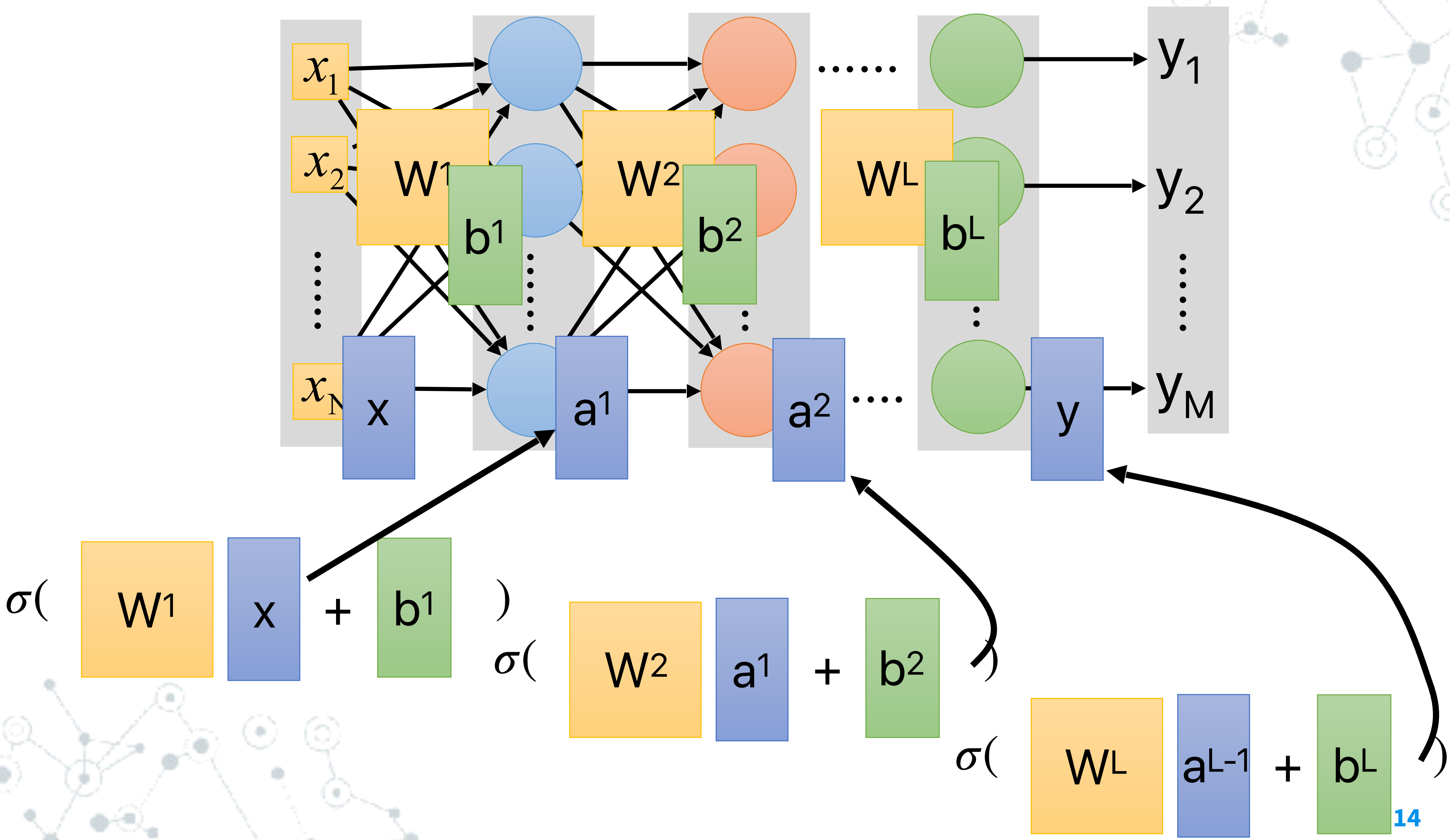
Different parameters define different function

Matrix Operation

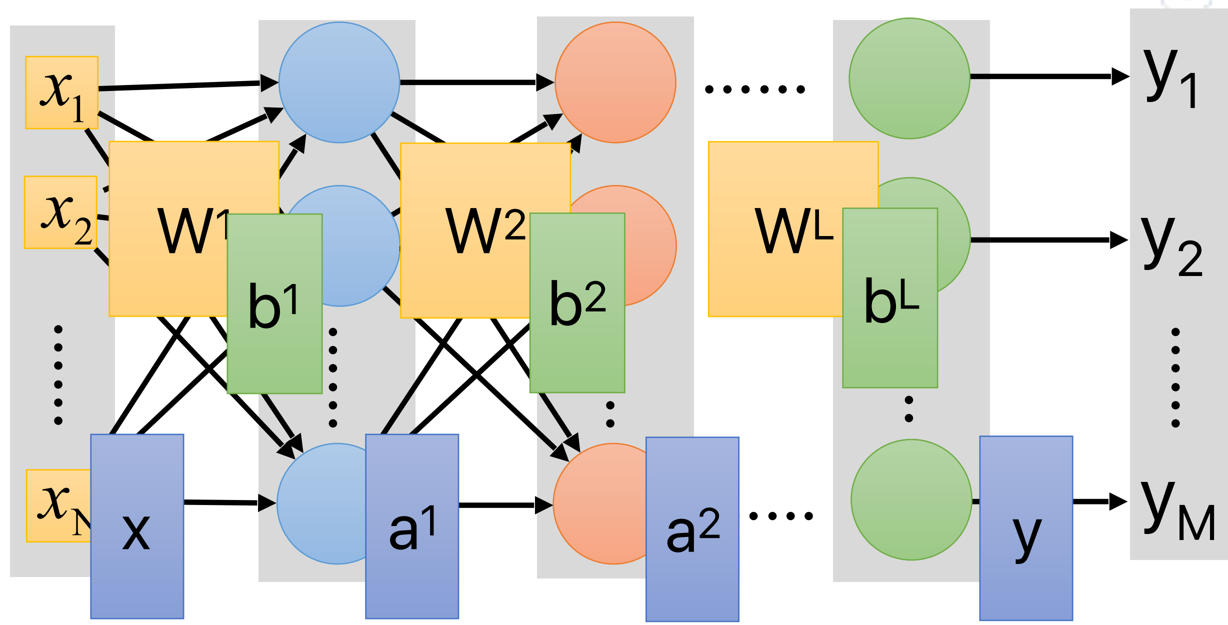


$$\sigma\Big(\underbrace{\begin{bmatrix} 1 & -2 \\ -1 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ -1 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix}}_{\begin{bmatrix} 4 \\ -2 \end{bmatrix}}\Big) = \begin{bmatrix} 0.98 \\ 0.12 \end{bmatrix}$$

Neural Network



Neural Network



$$y = f(x)$$

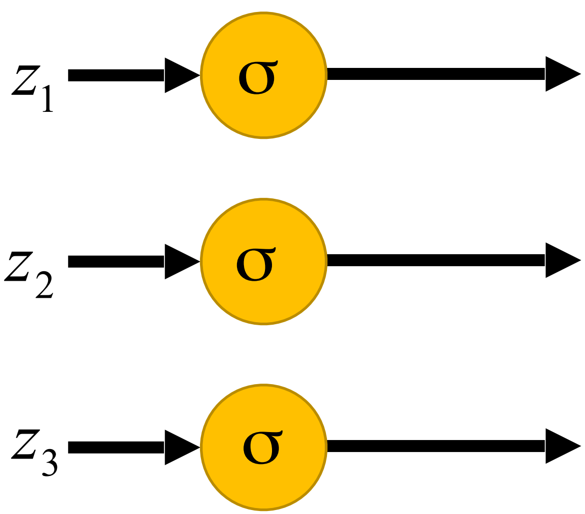
Using parallel computing techniques to speed up matrix operation

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

Softmax

◎ Softmax layer as the output layer

Ordinary Layer



$y_1 = \sigma(z_1)$

$y_2 = \sigma(z_2)$

$y_3 = \sigma(z_3)$

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

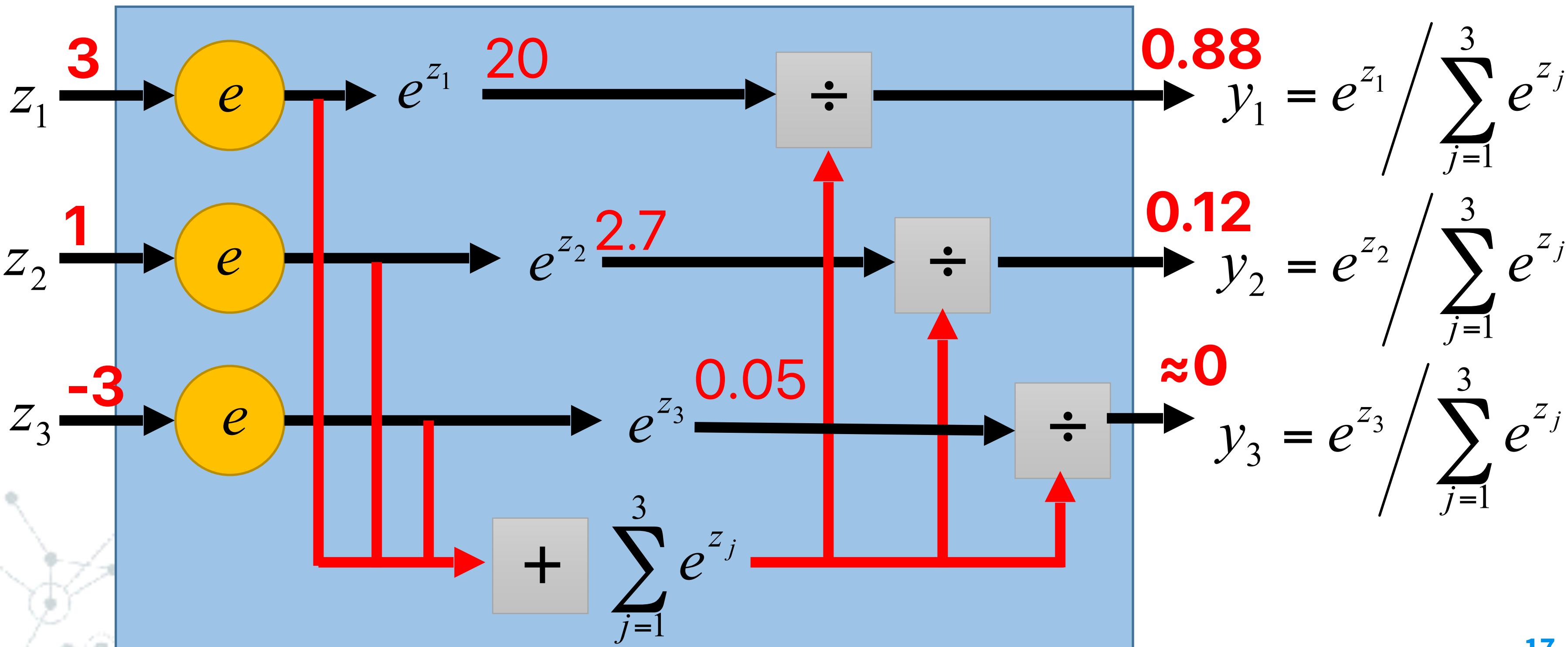
May not be easy to interpret

Softmax

- Softmax layer as the output layer

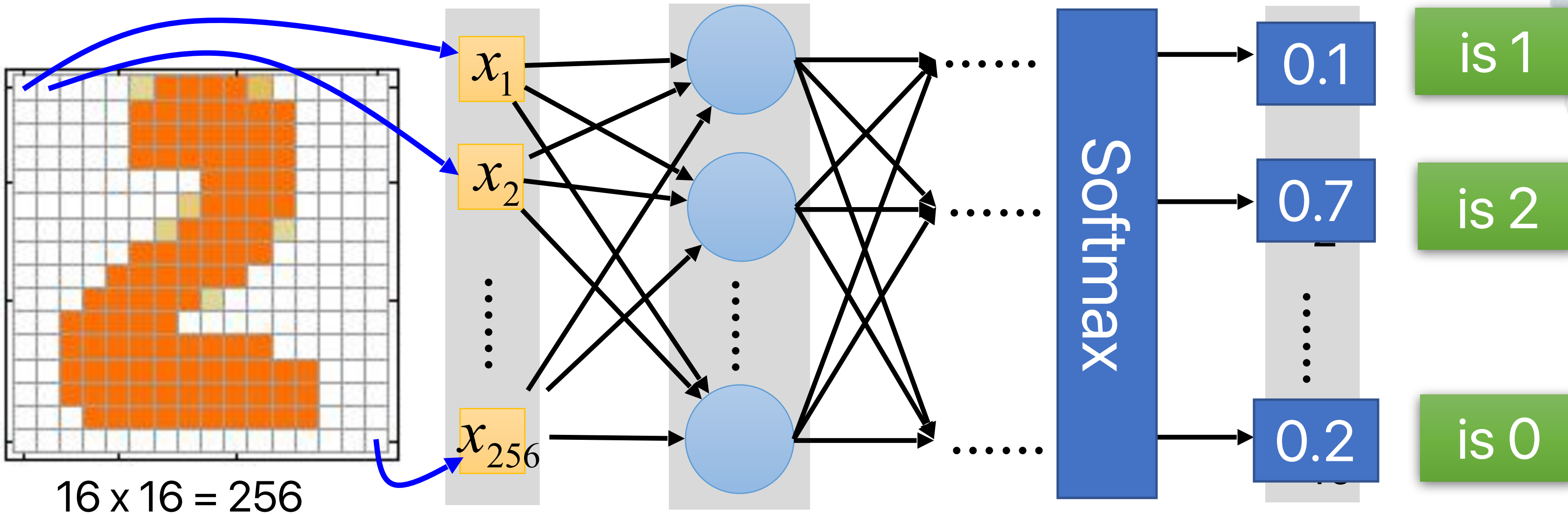
Softmax Layer

***Probability*:**



How to set network parameters

$$\theta = \{W^1, b^1, W^2, b^2, \cdots W^L, b^L\}$$



16 x 16 = 256

Ink → 1

No ink → 0

Set the network parameters such that ......

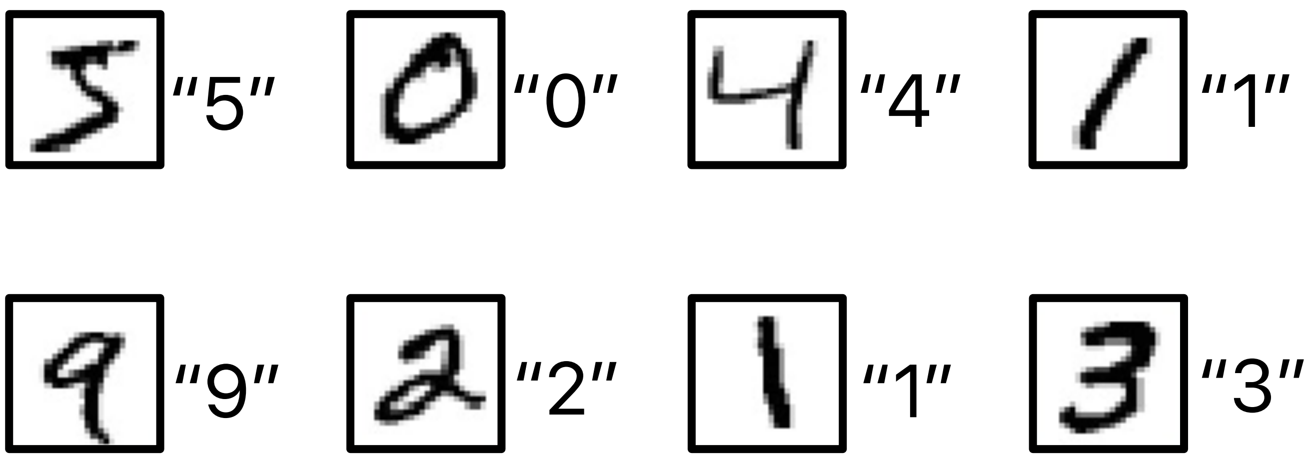
Input ... m value

How to let the neural network achieve this

Input:  y_2 has the maximum value

Training Data

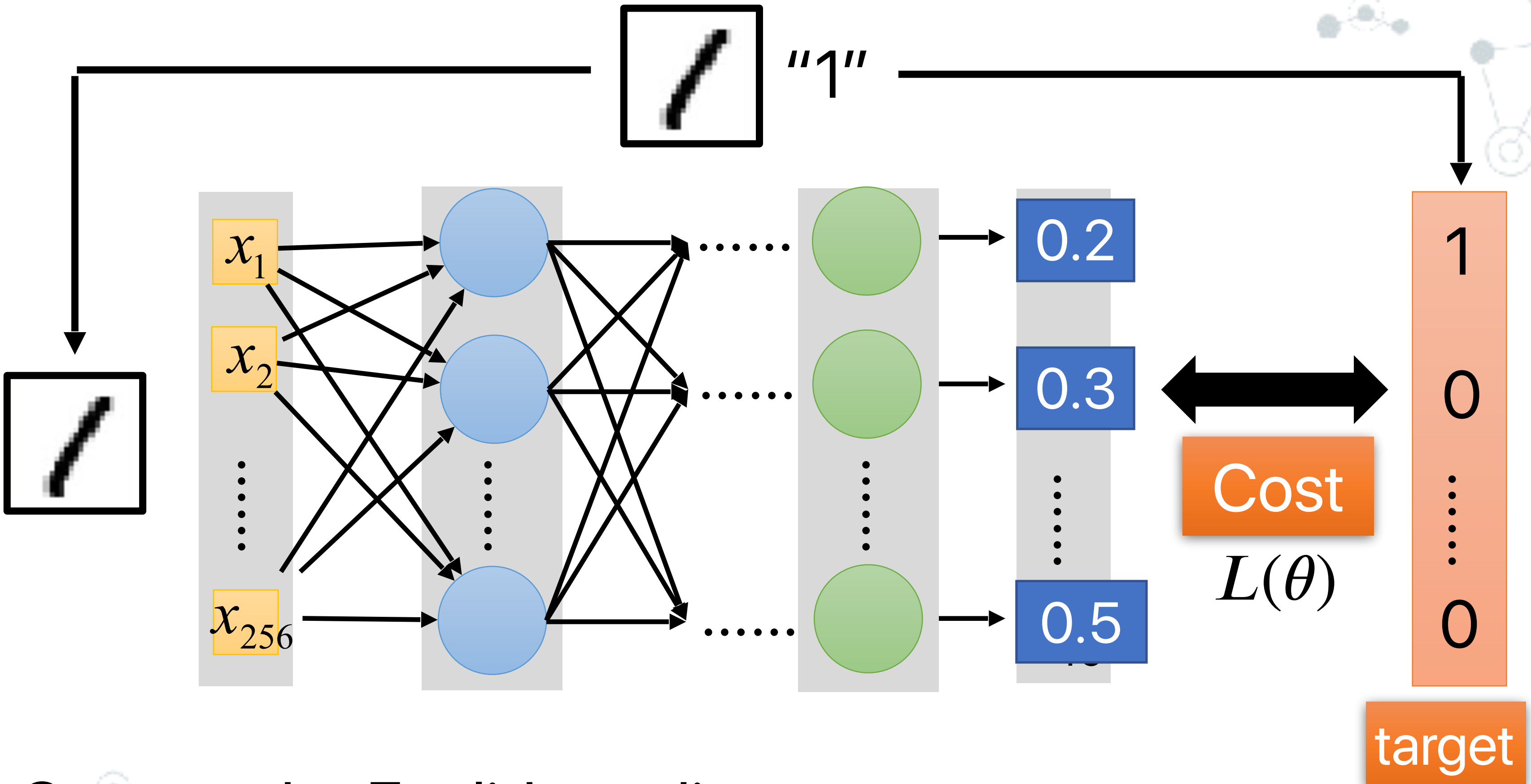
◎ Preparing training data: images and their labels



Using the training data to find the network parameters.

Cost

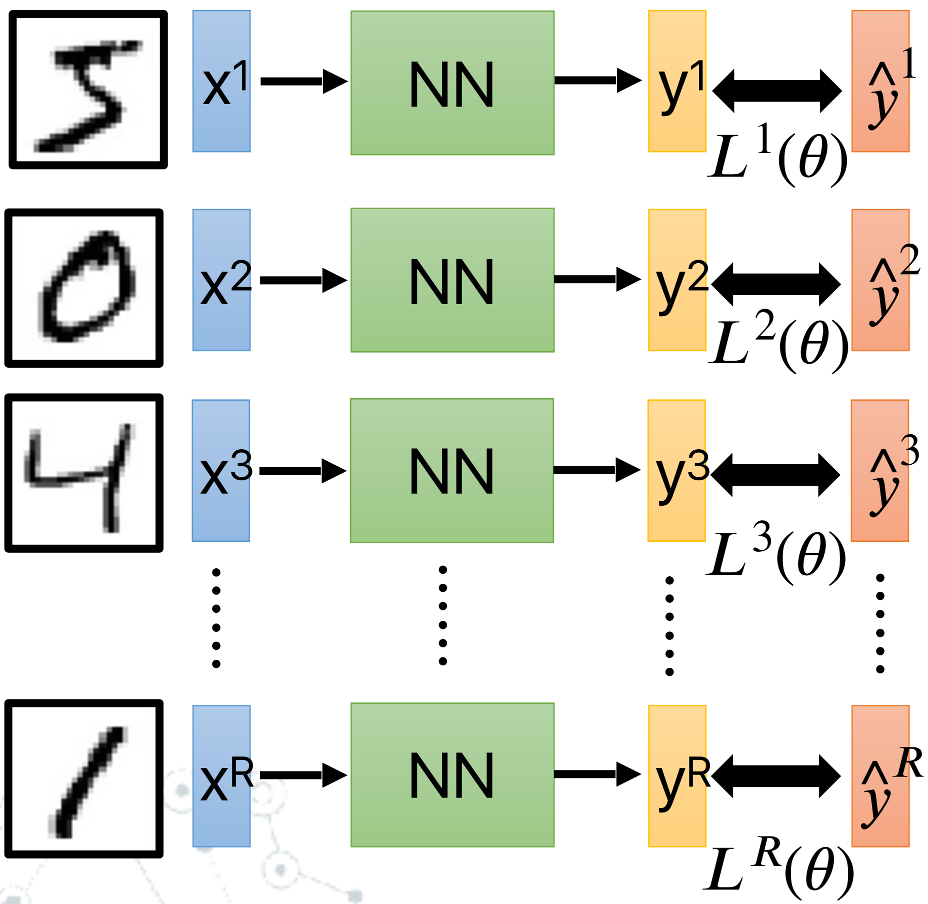
Given a set of network parameters , each example has a cost value.



Cost can be Euclidean distance or cross entropy of the network output and target

Total Cost

For all training data …



Total Cost:

$$C(\theta) = \sum_{r=1}^{R} L^r(\theta)$$

How bad the network parameters is on this task

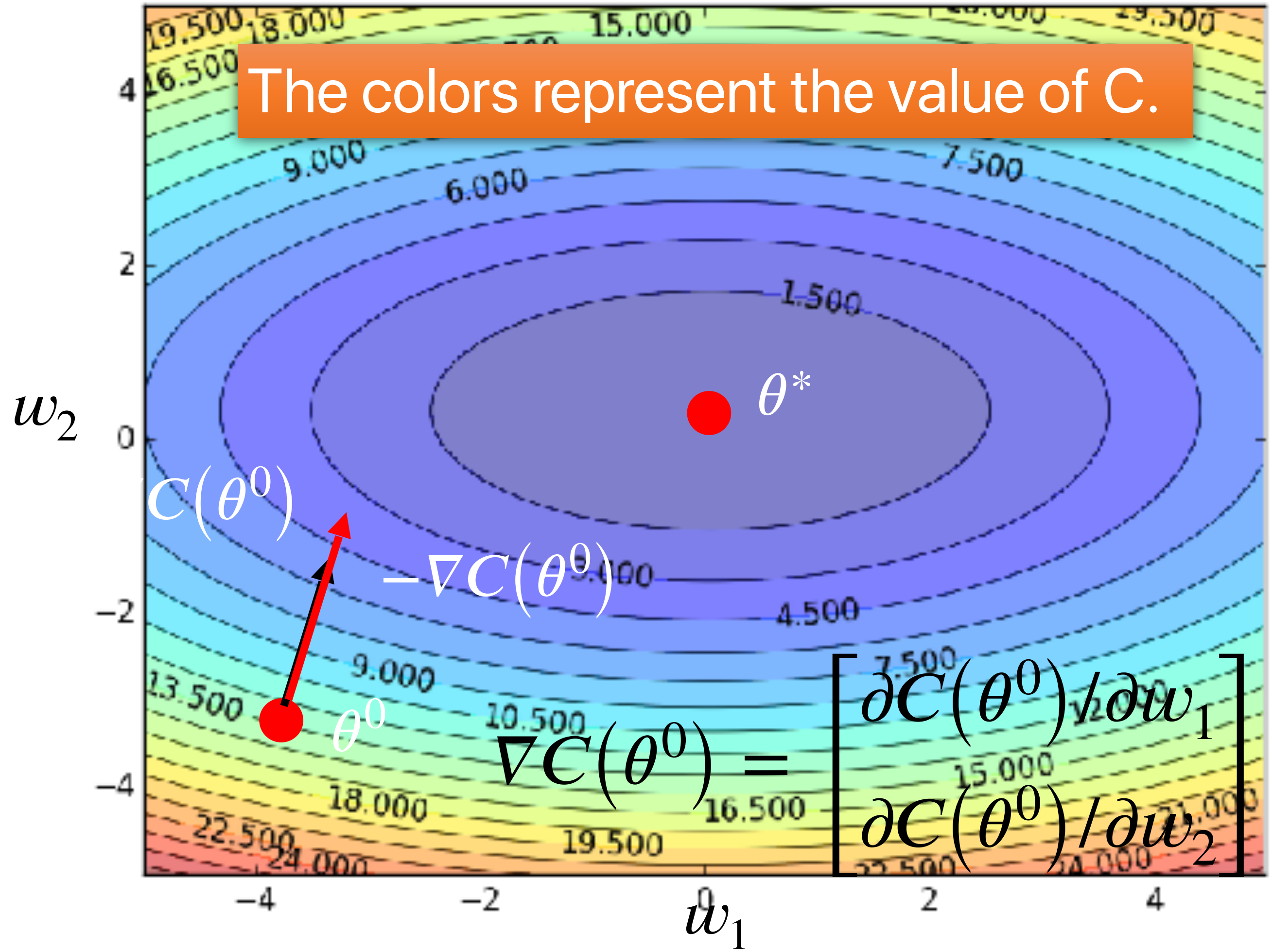
Find the network parameters that minimize this value

Gradient Descent

Assume there are only two parameters w_1 and w_2 in a network.

$$\theta = \{w_1, w_2\}$$

Error Surface



Randomly pick a starting point

Compute the negative gradient at

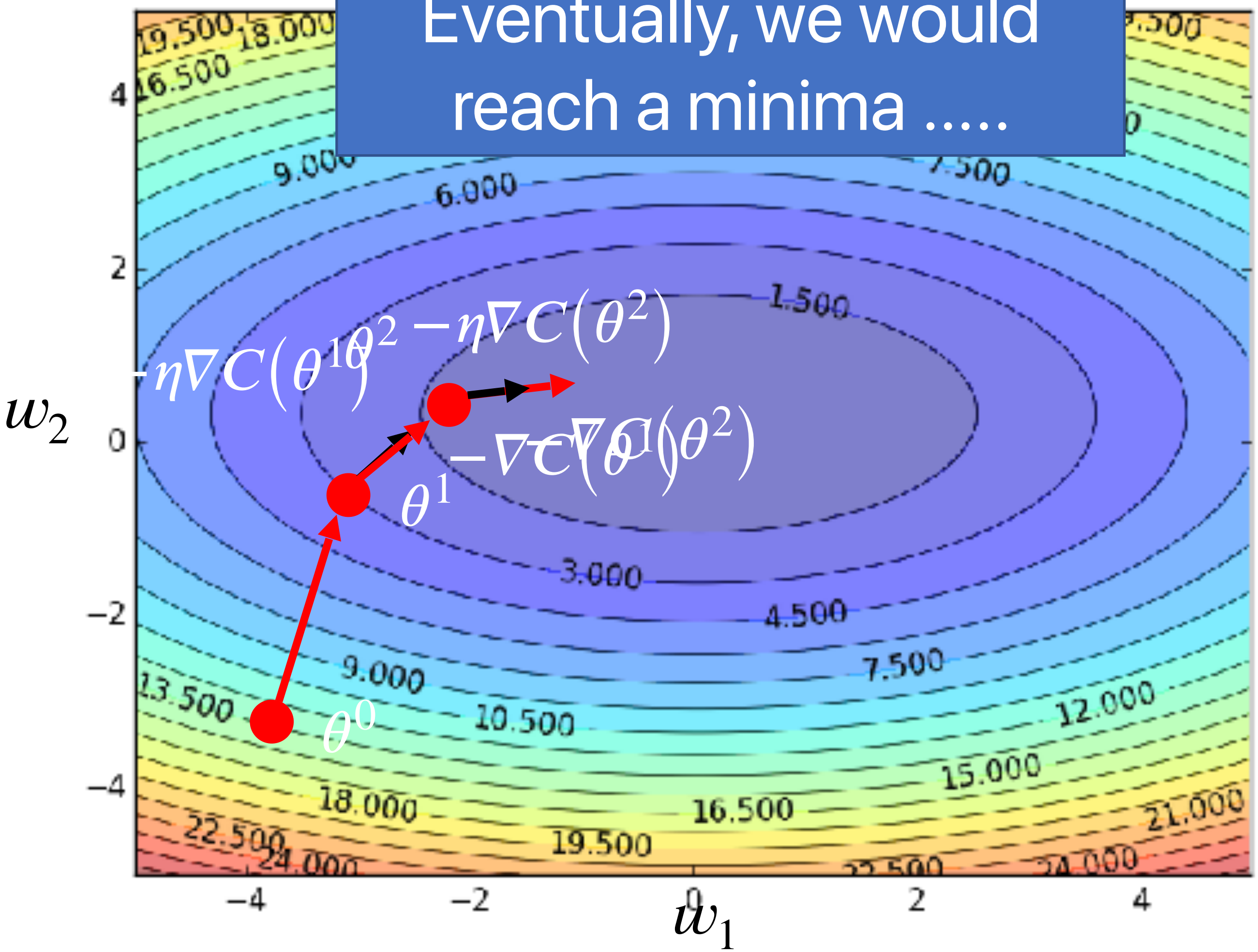
$\Rightarrow -\nabla C(\theta^0)$

Times the learning rate

$\Rightarrow -\eta \nabla C(\theta^0)$

Gradient Descent

Eventually, we would reach a minima .....



Randomly pick a starting point

Compute the negative gradient at

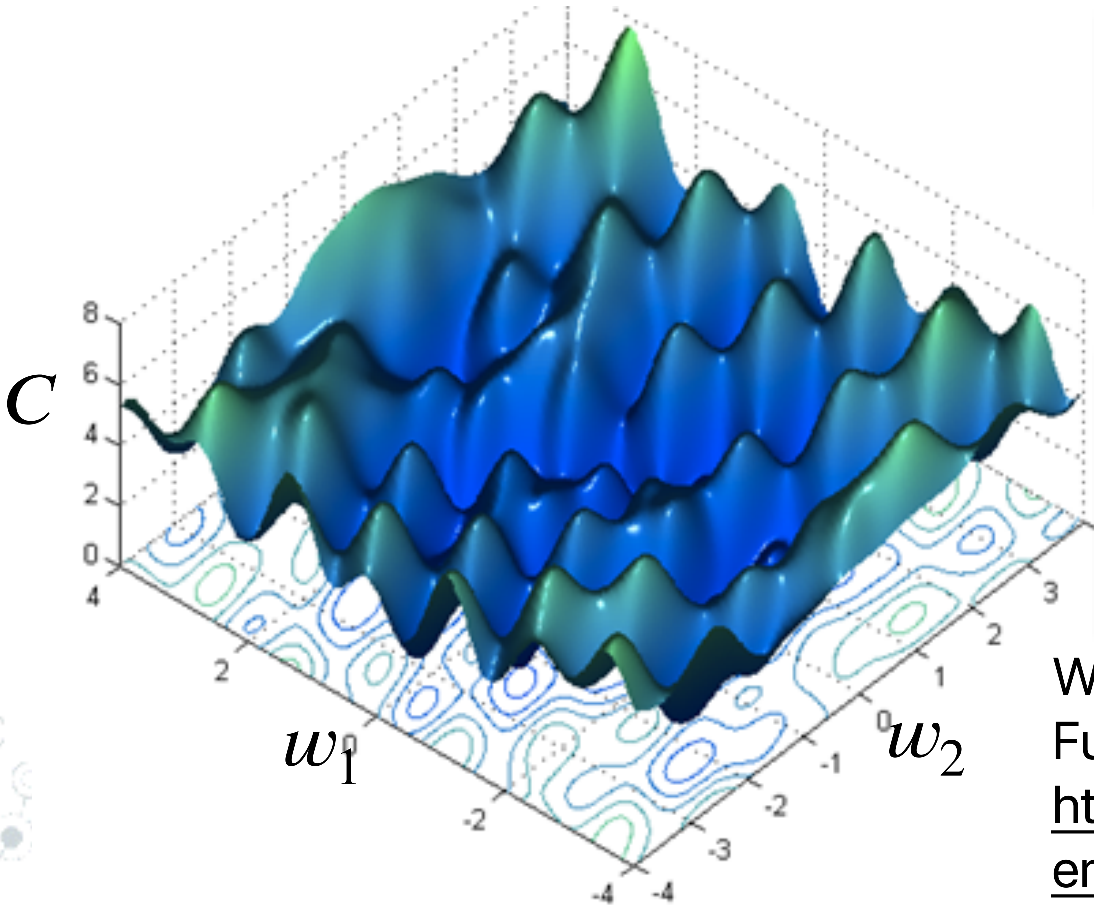
$$\Rightarrow -\nabla C(\theta^0)$$

Times the learning rate

$$\Rightarrow -\eta\nabla C(\theta^0)$$

Local Minima

◎ Gradient descent never guarantee global

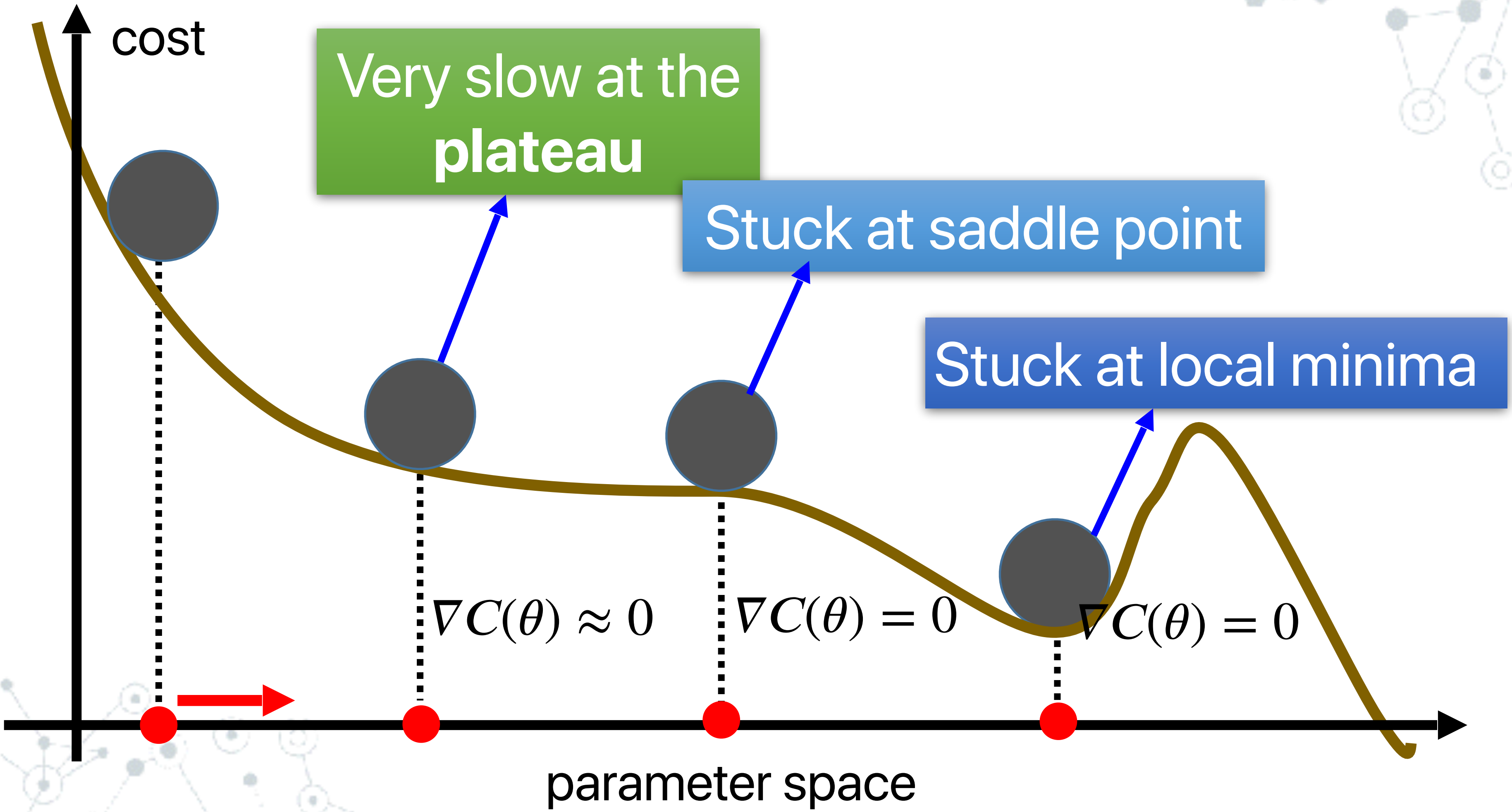


Different initial point

Reach different minima, so different results

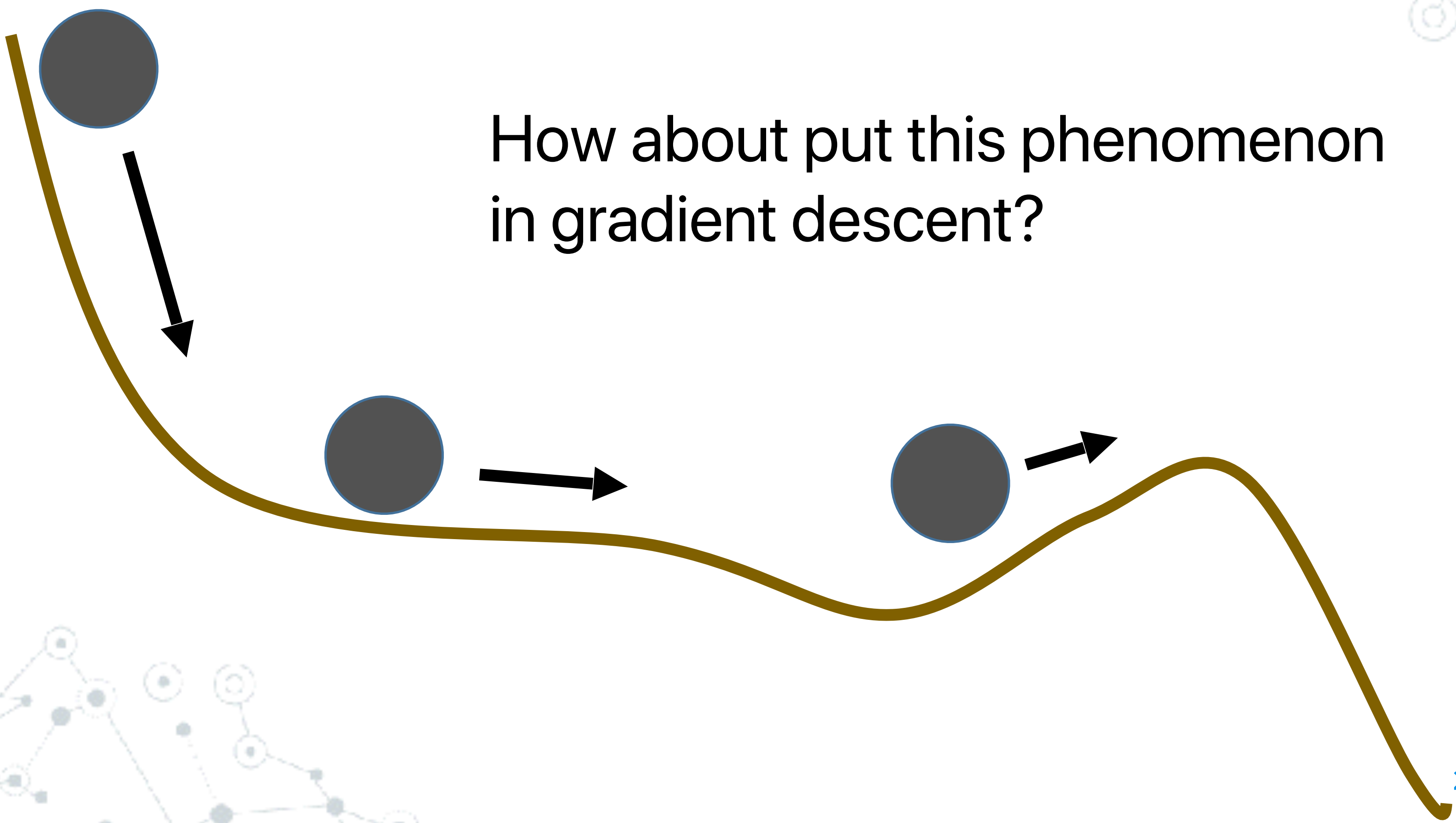
Who is Afraid of Non-Convex Loss Functions?
http://videolectures.net/eml07_lecun_wia/

Besides local minima ……



In physical world ……

◎ Momentum

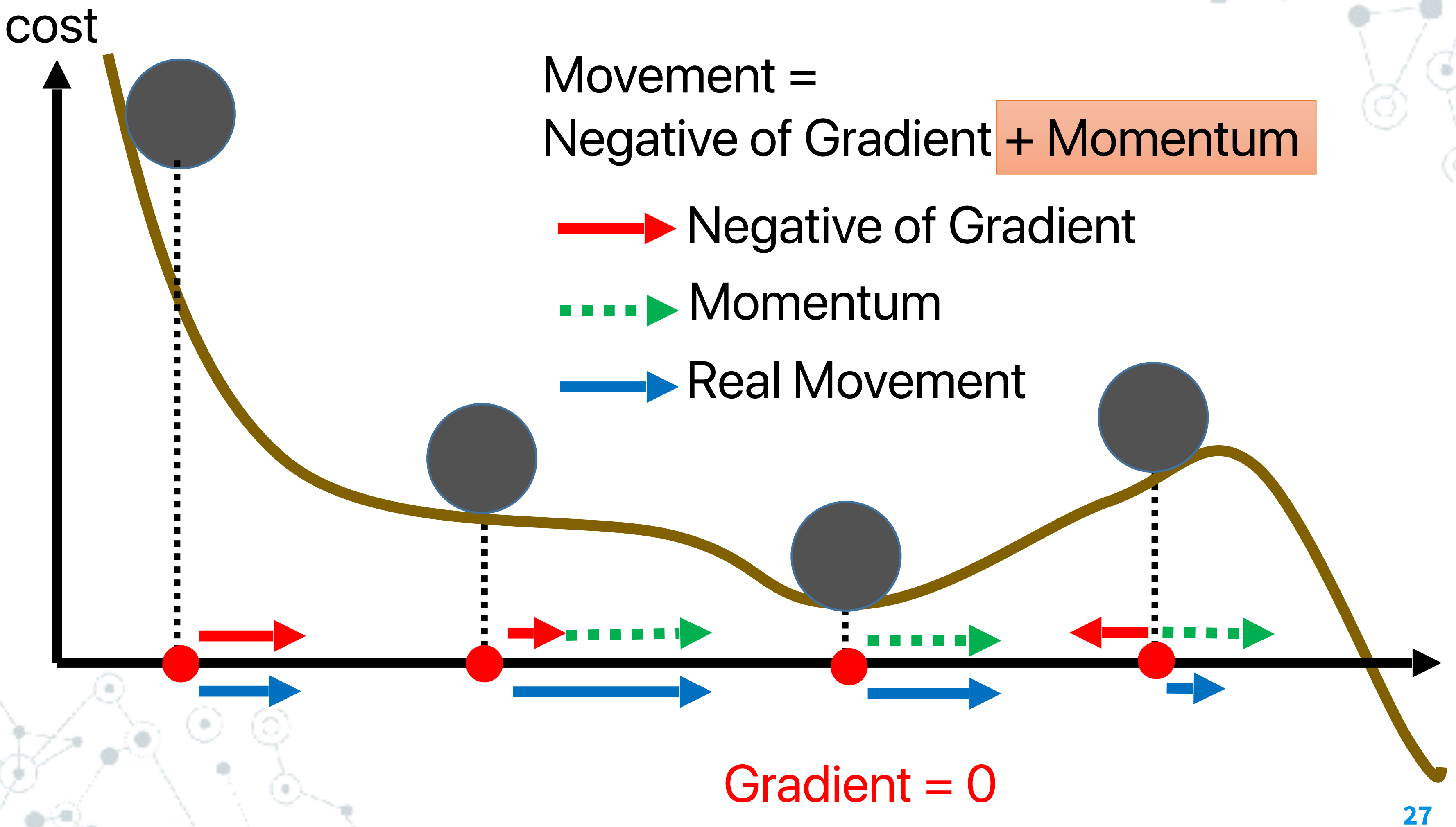


How about put this phenomenon in gradient descent?

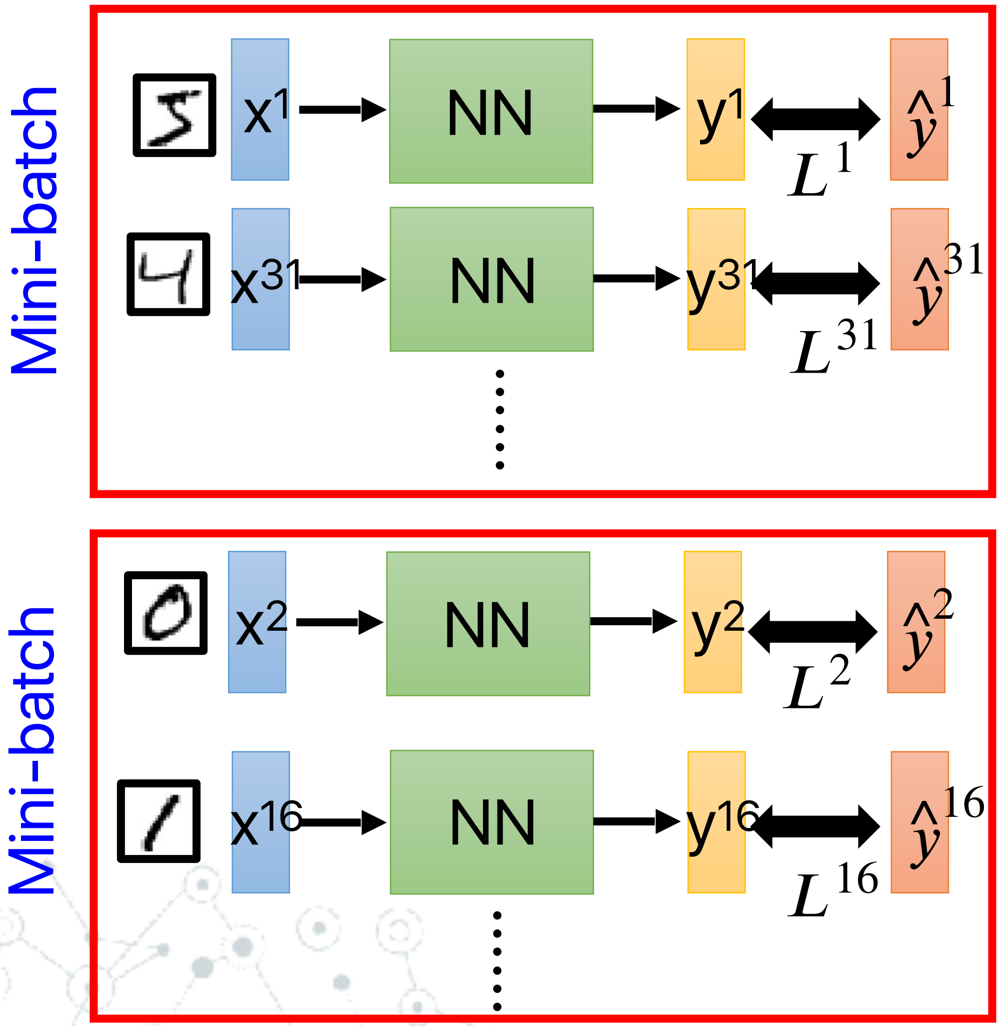
Momentum

Still not guarantee reaching global minima, but give some hope ……

Movement = Negative of Gradient + Momentum



Mini-batch



- Randomly initialize
- Pick the 1st batch

$$C = L^1 + L^{31} + \cdots$$

$$\theta^1 \leftarrow \theta^0 - \eta \nabla C(\theta^0)$$

- Pick the 2nd batch

$$C = L^2 + L^{16} + \cdots$$

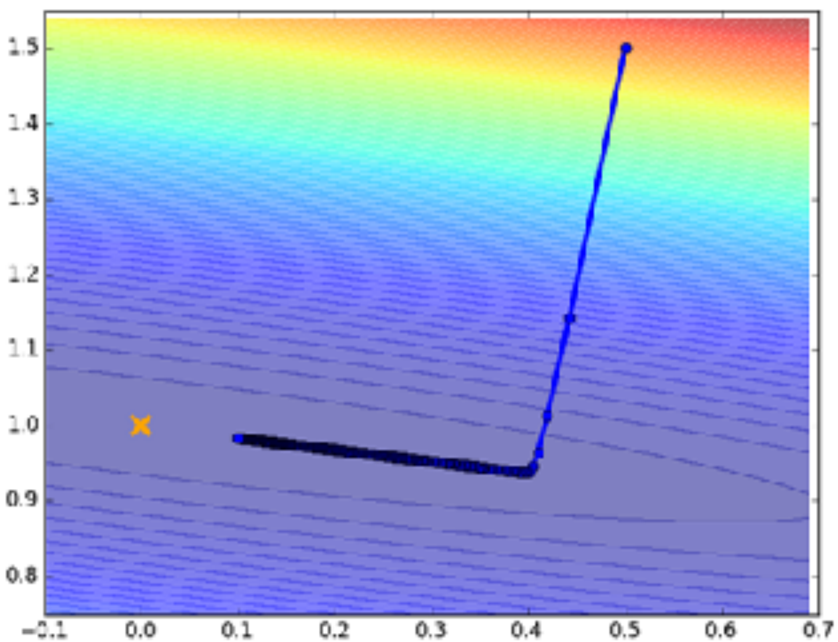
$$\theta^2 \leftarrow \theta^1 - \eta \nabla C(\theta^1)$$

$$\vdots$$

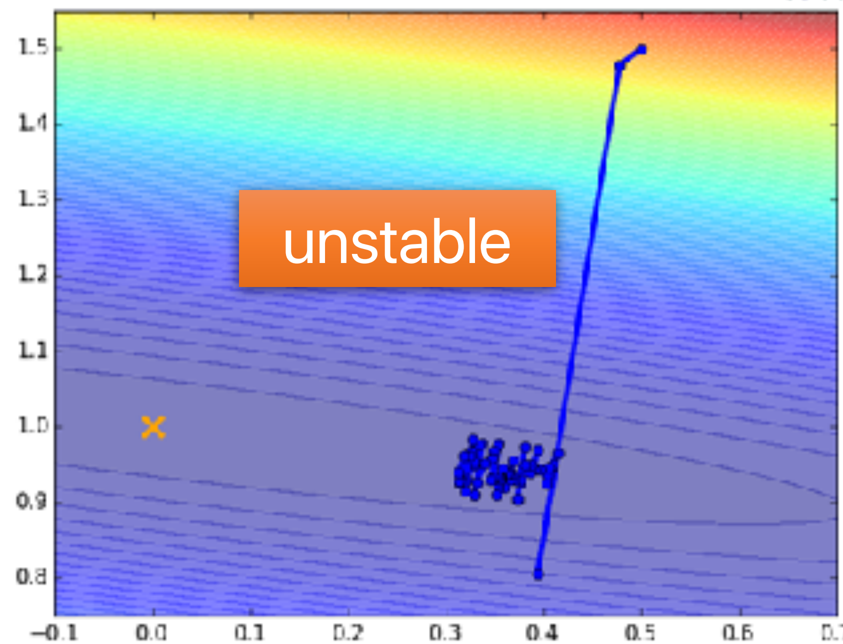
C is different each time when we update parameters!

Mini-batch

Original Gradient Descent



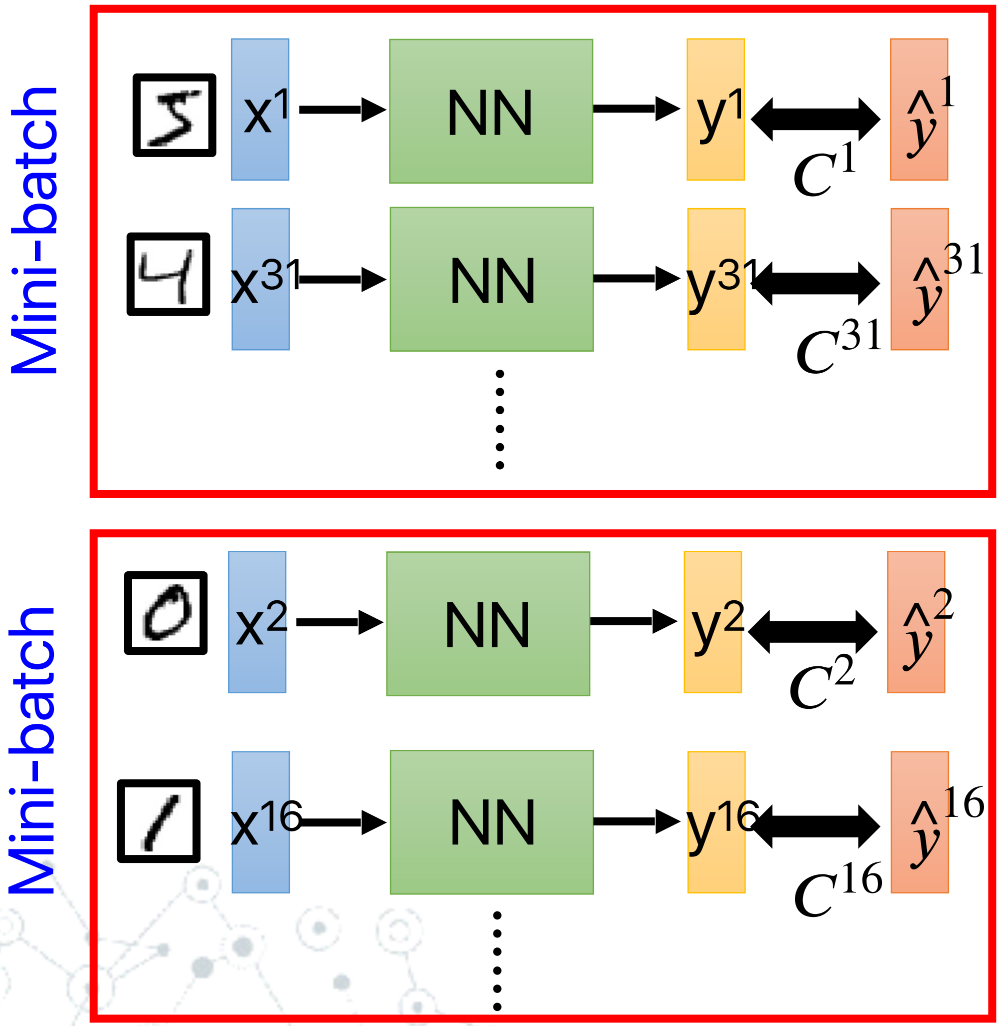
With Mini-batch



The colors represent the total C on all training data.

Mini-batch

Faster Better!



- Randomly initialize
- Pick the 1st batch

$$C = C^1 + C^{31} + \cdots$$

$$\theta^1 \leftarrow \theta^0 - \eta \nabla C(\theta^0)$$

- Pick the 2nd batch

$$C = C^2 + C^{16} + \cdots$$

$$\theta^2 \leftarrow \theta^1 - \eta \nabla C(\theta^1)$$

$$\vdots$$

- Until all mini-batches have been picked

one epoch

Repeat the above process

Backpropagation

- A network can have millions of parameters.
  - Backpropagation is the way to compute the gradients efficiently (not today)
  - Ref: http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2/Lecture/DNN%20backprop.ecm.mp4/index.html
- Many toolkits can compute the gradients automatically

theano torch TensorFlow

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

Part II:
Why Deep?

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.

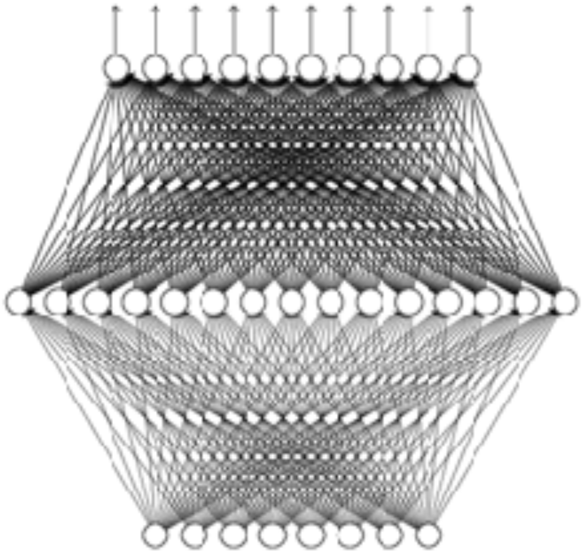
Universality Theorem

Any continuous function f

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

Can be realized by a network with one hidden layer

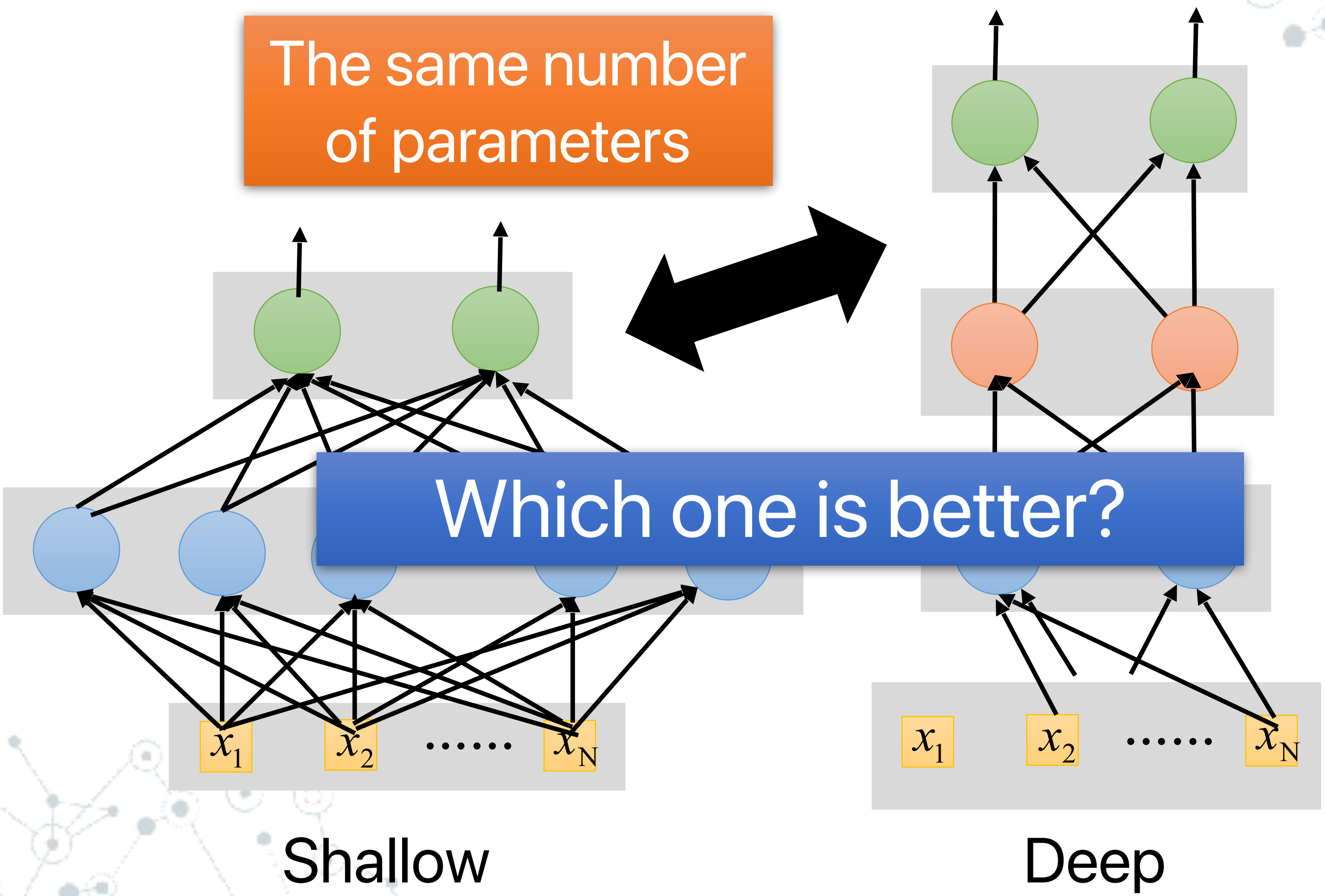
(given **enough** hidden neurons)



Reference for the reason: http://neuralnetworksanddeeplearning.com/chap4.html

Why "Deep" neural network not "Fat" neural network?

Fat + Short v.s. Thin + Tall



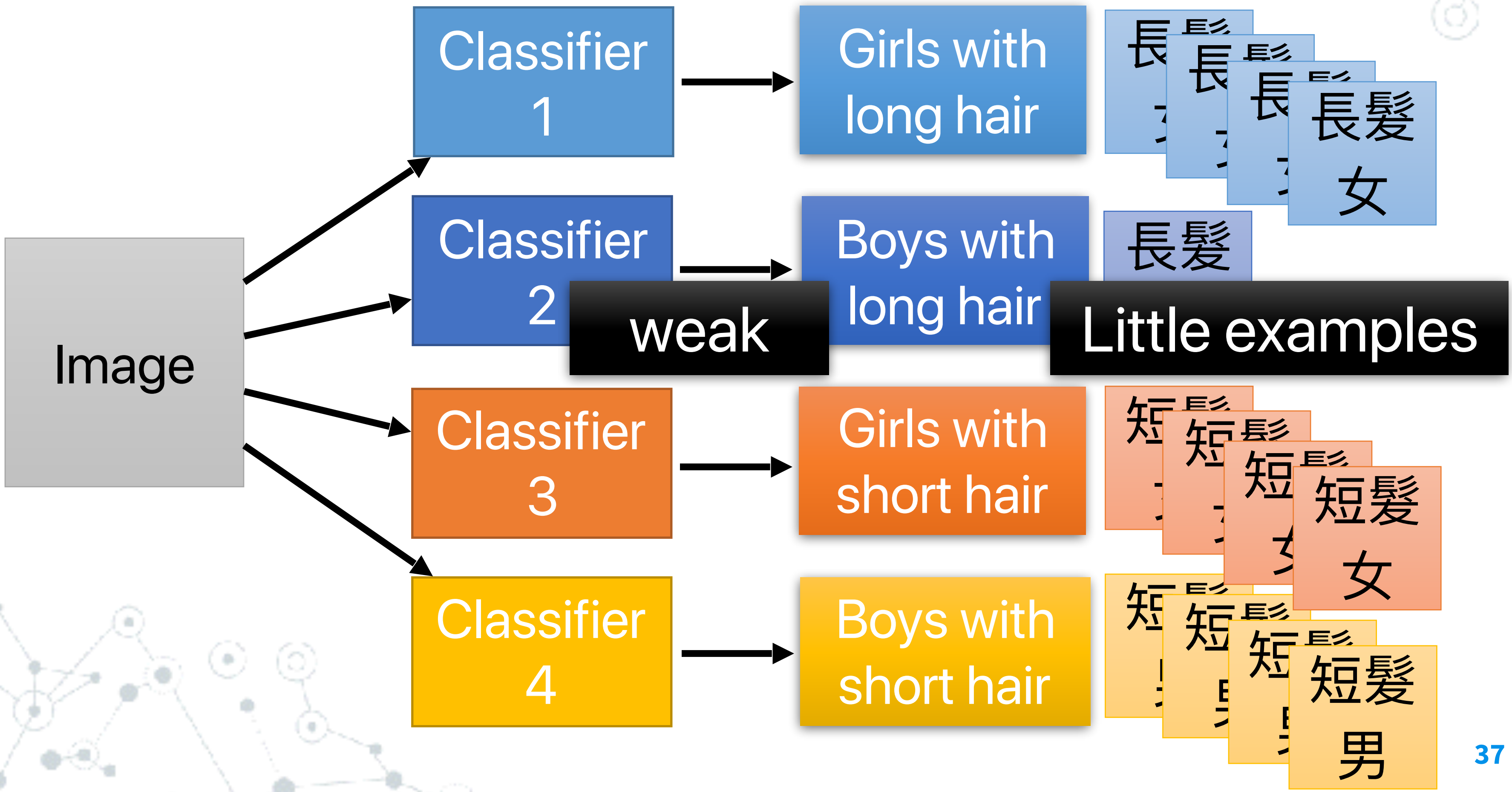
Fat + Short v.s. Thin + Tall

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

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

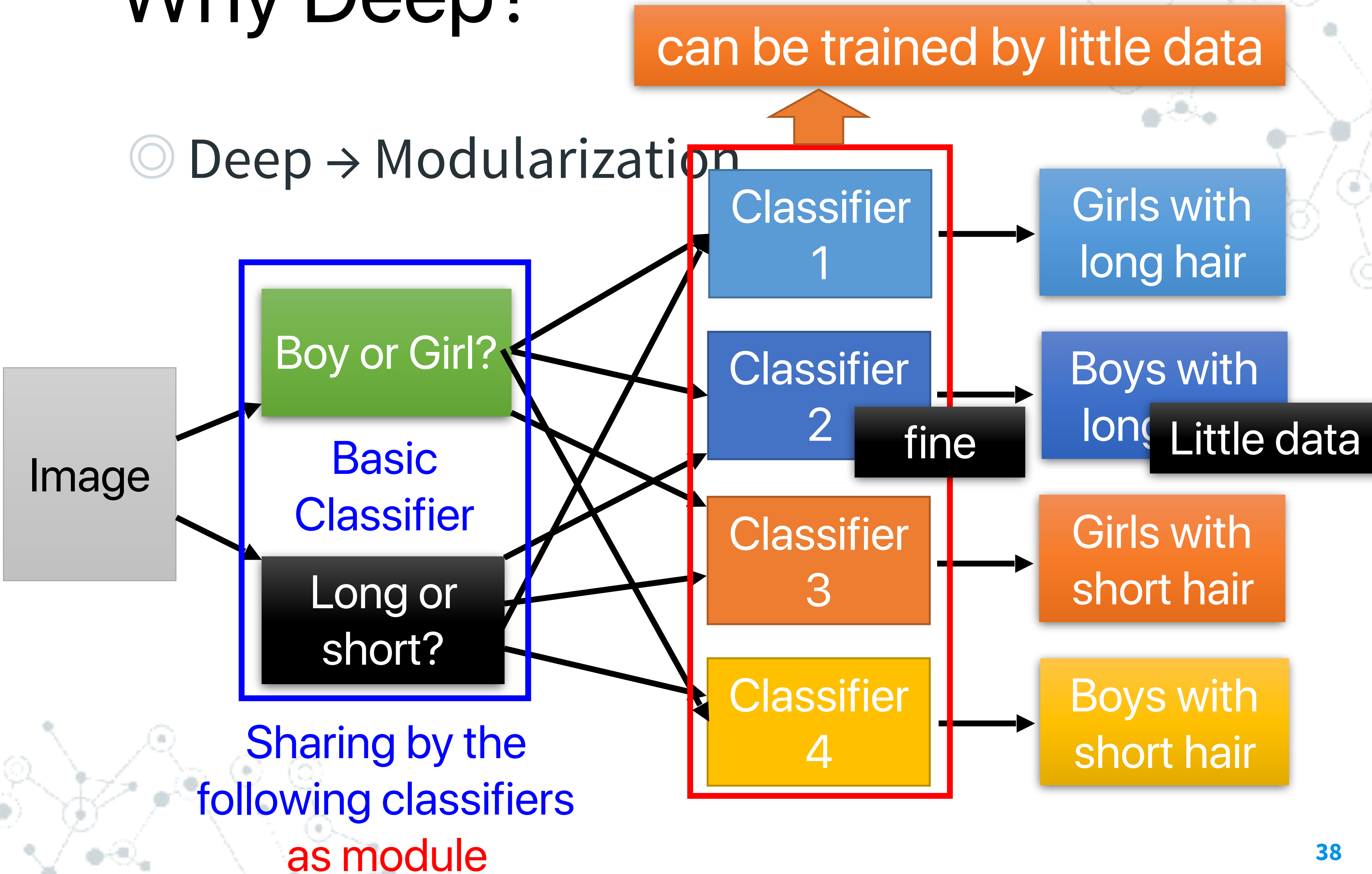
Why Deep?

◎ Deep → Modularization



Why Deep?

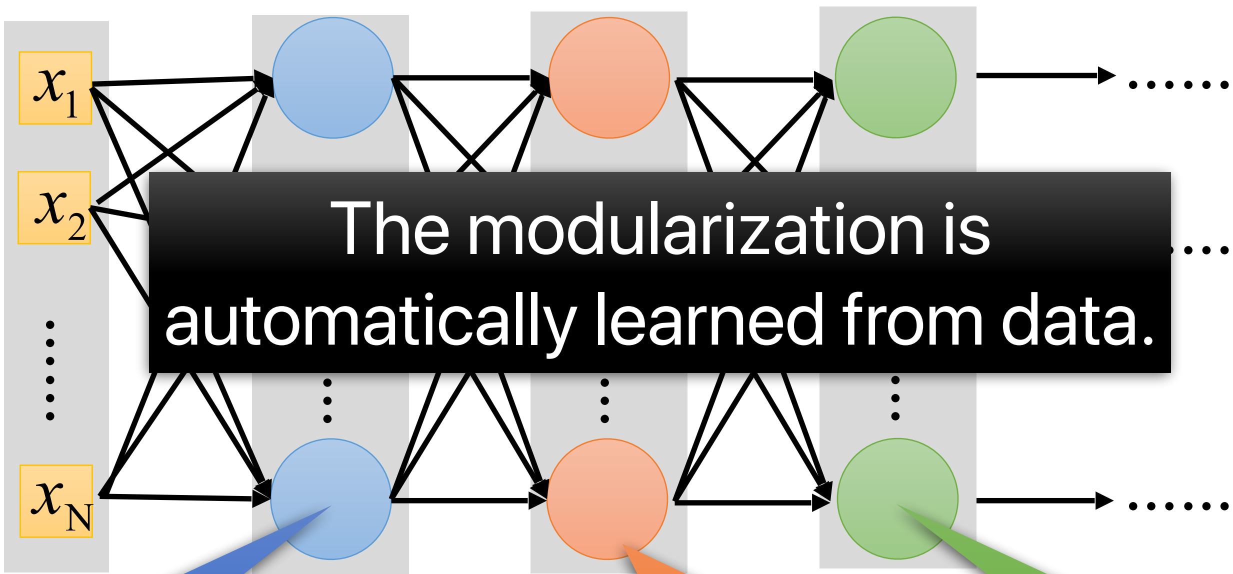
◎ Deep → Modularization



Why Deep?

Deep Learning also works on small data set like TIMIT.

◎ Deep → Modularizat → Less training data?



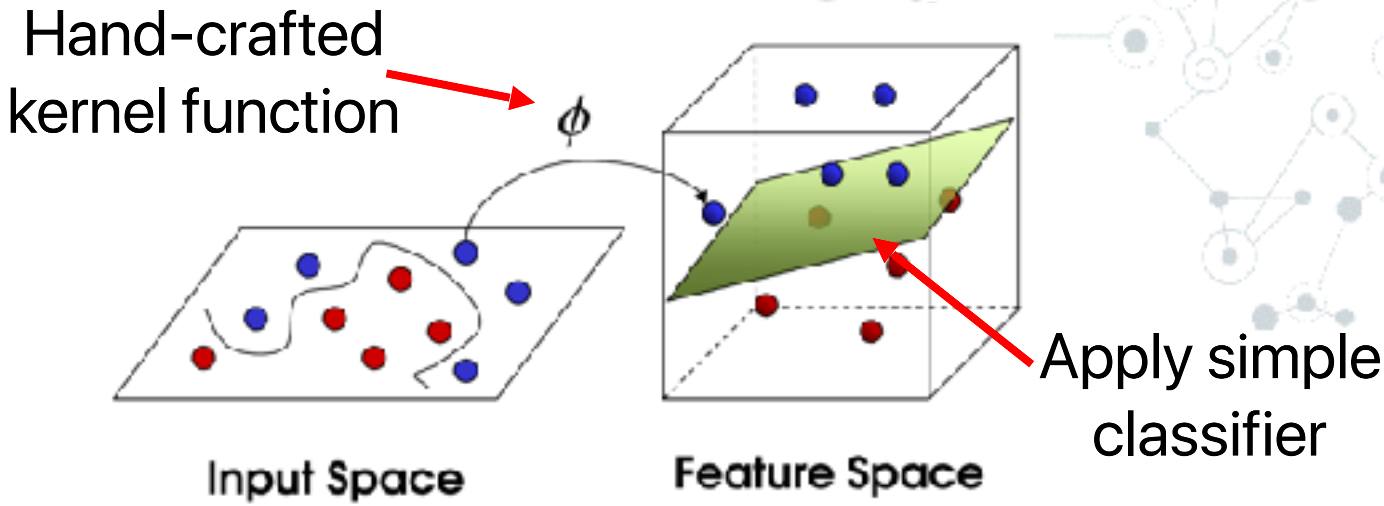
The modularization is automatically learned from data.

The most basic classifiers

Use 1st layer as module to build classifiers

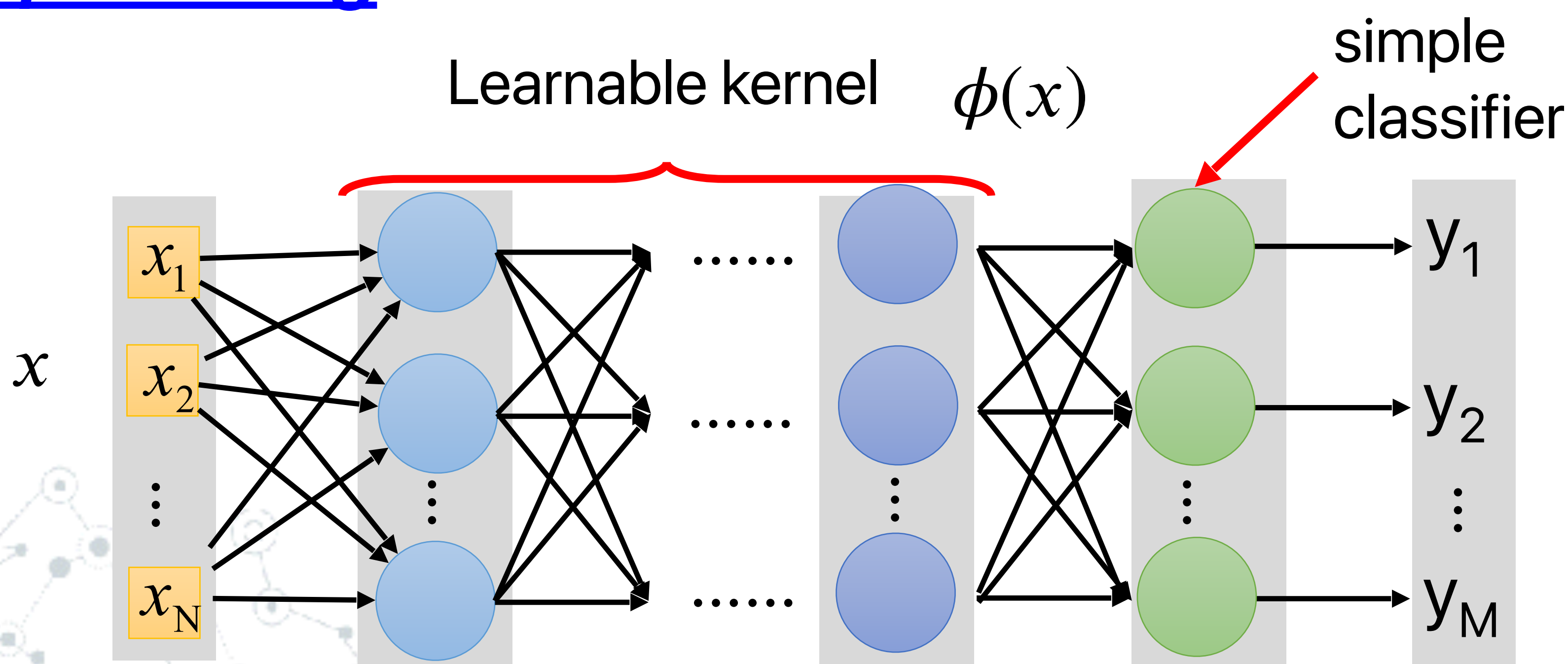
Use 2nd layer as module ......

SVM

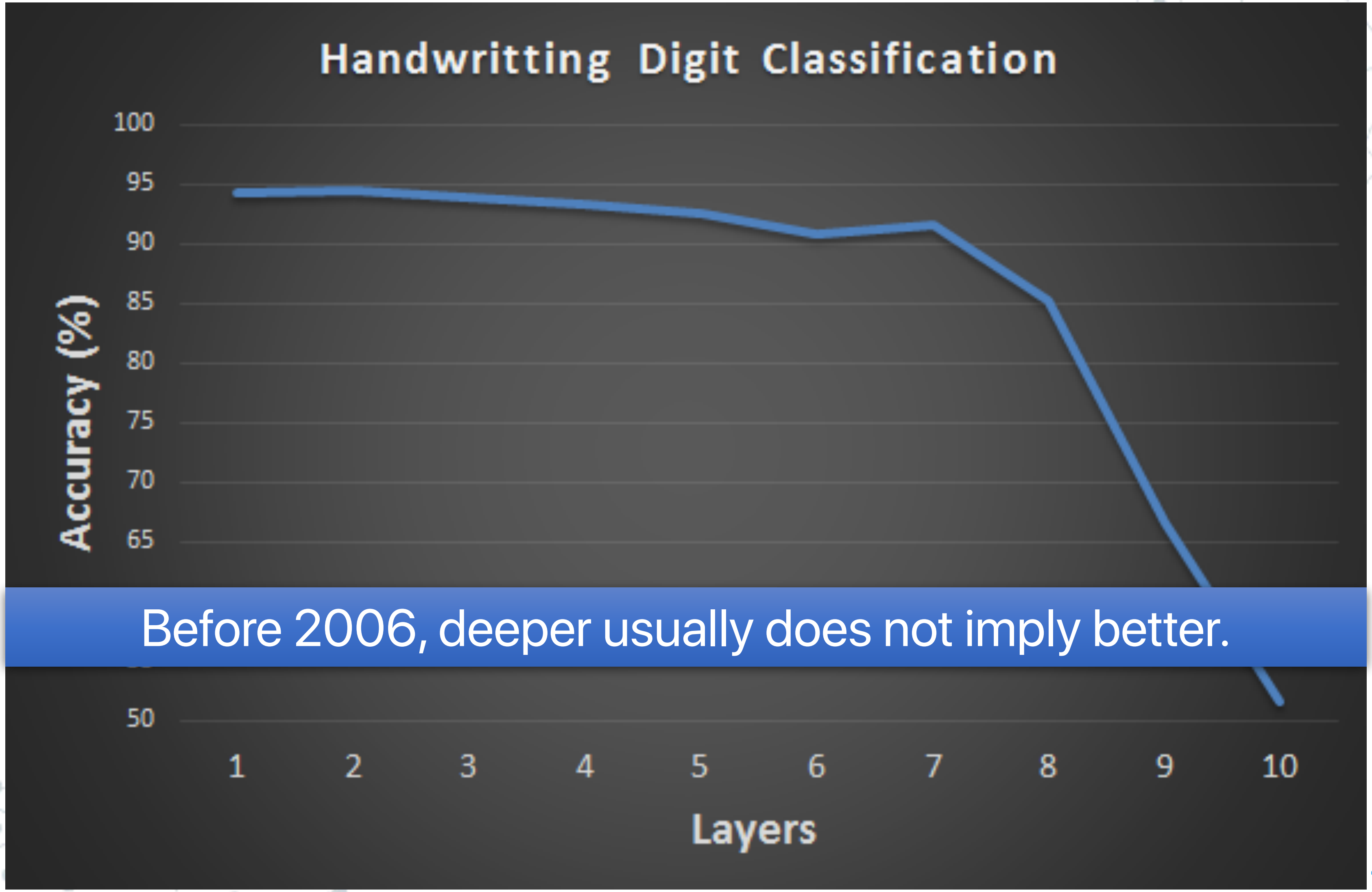


Source of image: http://www.gipsa-lab.grenoble-inp.fr/transfert/seminaire/455_Kadri2013Gipsa-lab.pdf

Deep Learning



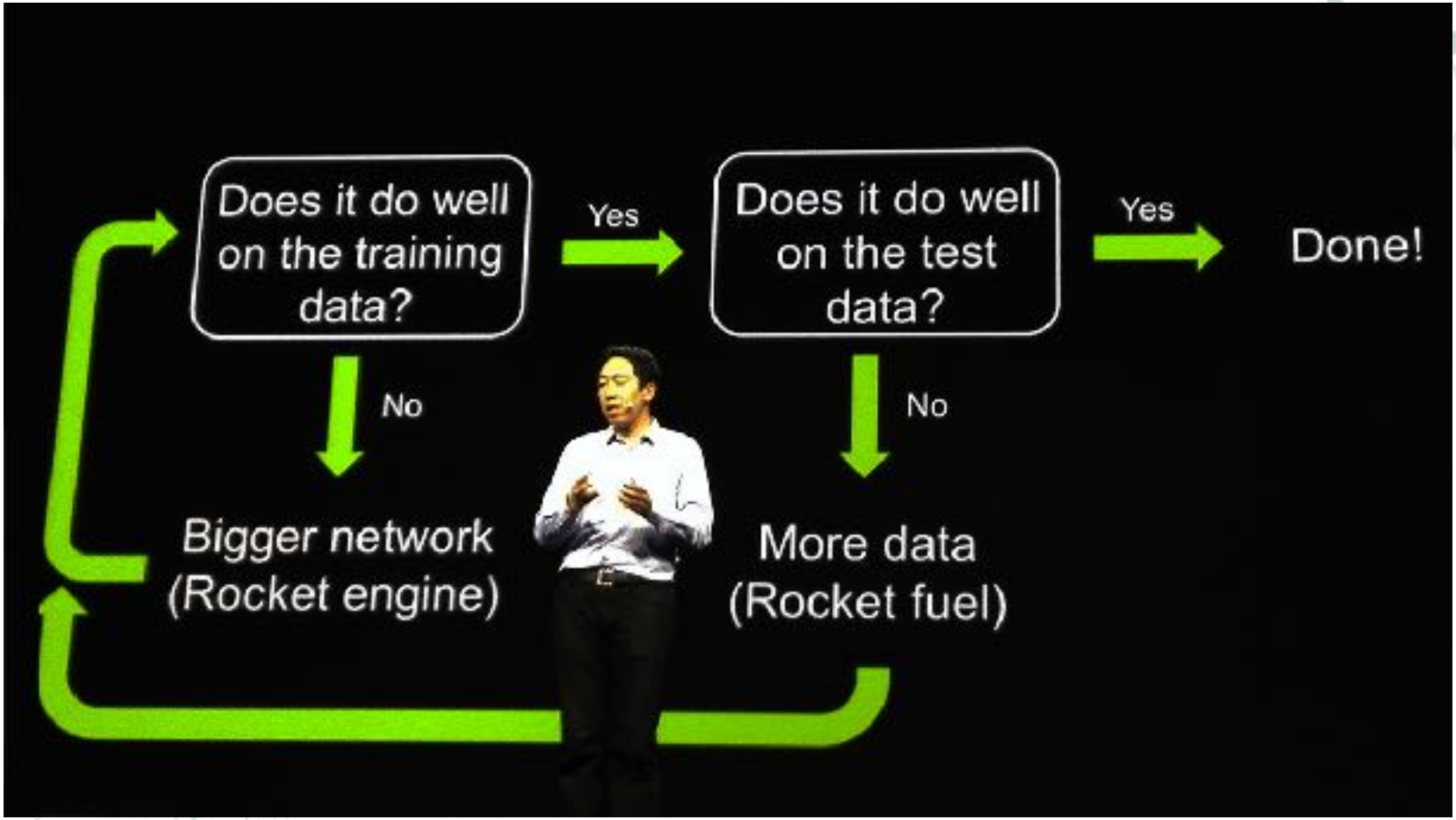
Hard to get the power of Deep …



Before 2006, deeper usually does not imply better.

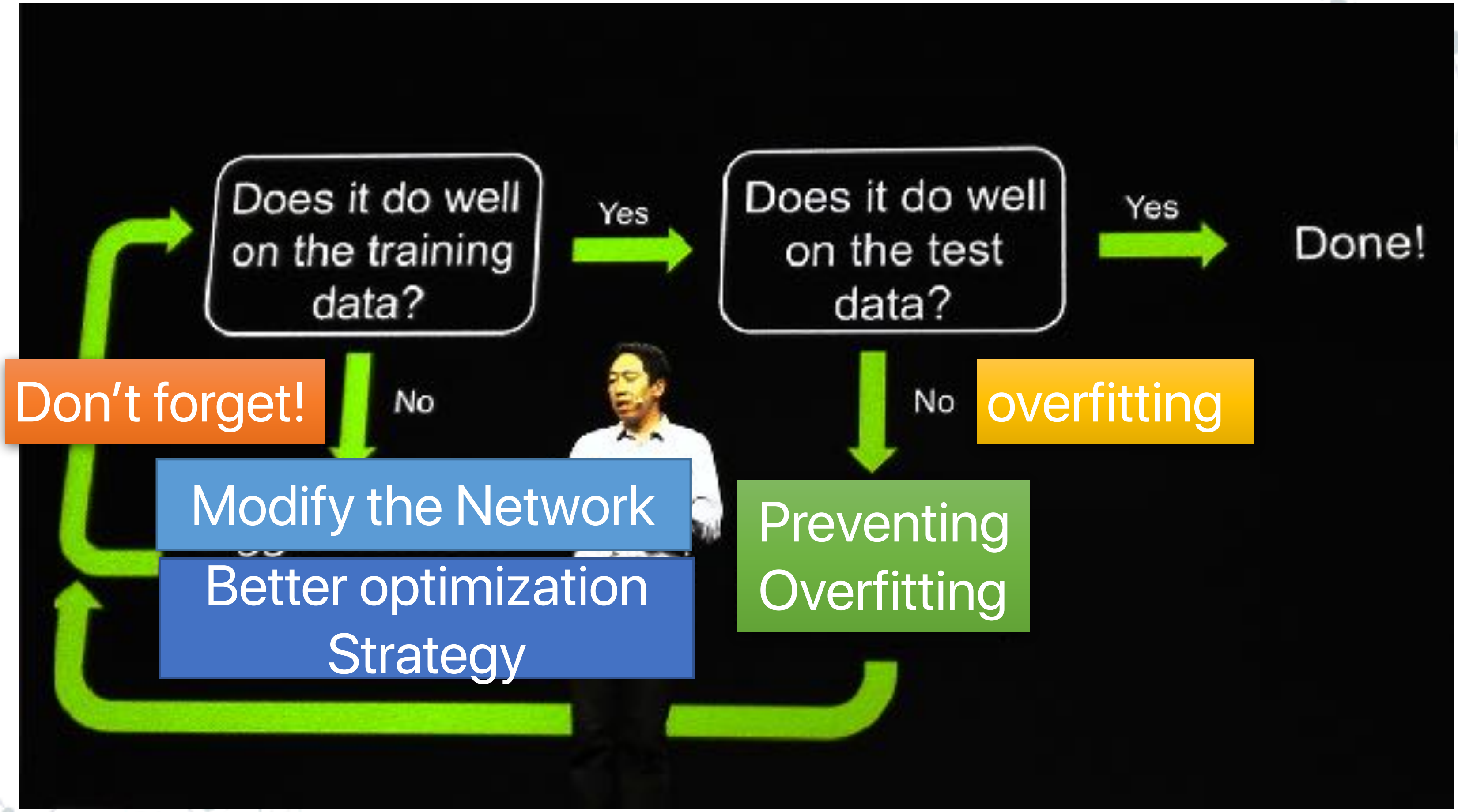
Part III:
Tips for Training DNN

Recipe for Learning



http://www.gizmodo.com.au/2015/04/the-basic-recipe-for-machine-learning-explained-in-a-single-powerpoint-slide/

Recipe for Learning



http://www.gizmodo.com.au/2015/04/the-basic-recipe-for-machine-learning-explained-in-a-single-powerpoint-slide/

Recipe for Learning

Modify the Network

- New activation functions, for

Better optimization Strategy

- Adaptive learning rates

Prevent Overfitting

- Dropout

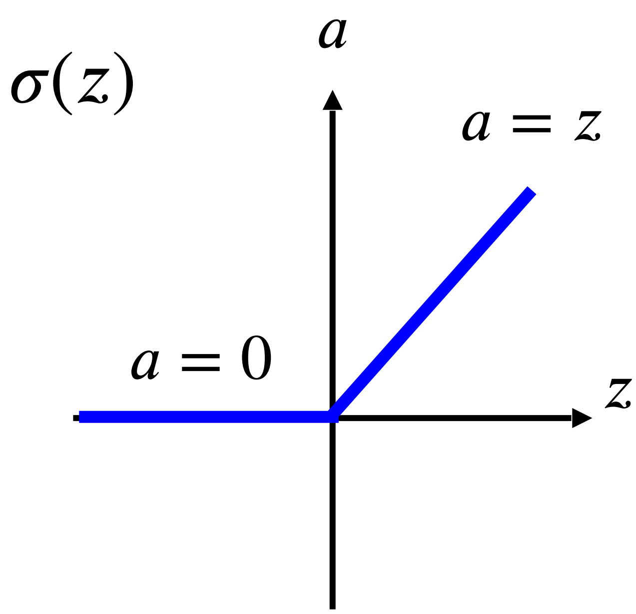
Only use this approach when you already obtained good results on the training data.

Part III:
Tips for Training DNN

New Activation Function

ReLU

◎ Rectified Linear Unit (ReLU)

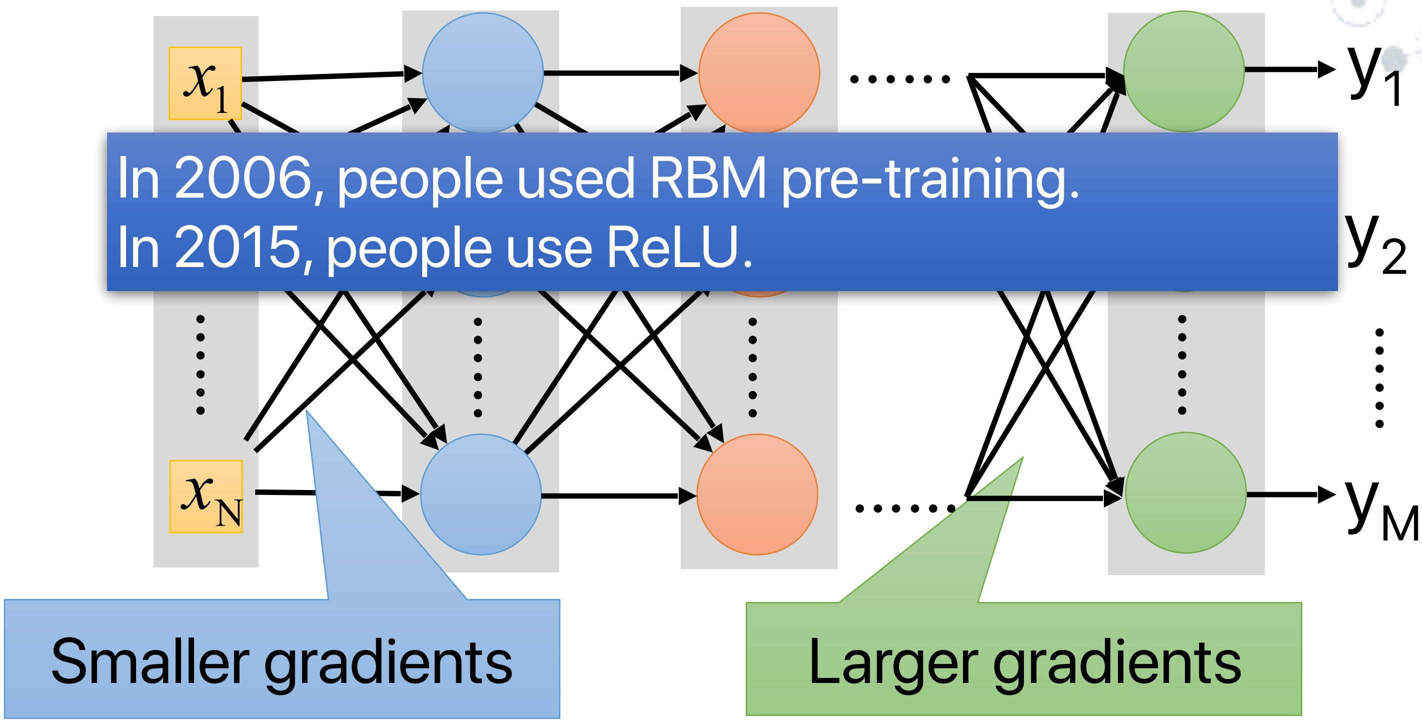


[Xavier Glorot, AISTATS'11]
[Andrew L. Maas, ICML'13]
[Kaiming He, arXiv'15]

Reason:

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

Vanishing Gradient Problem

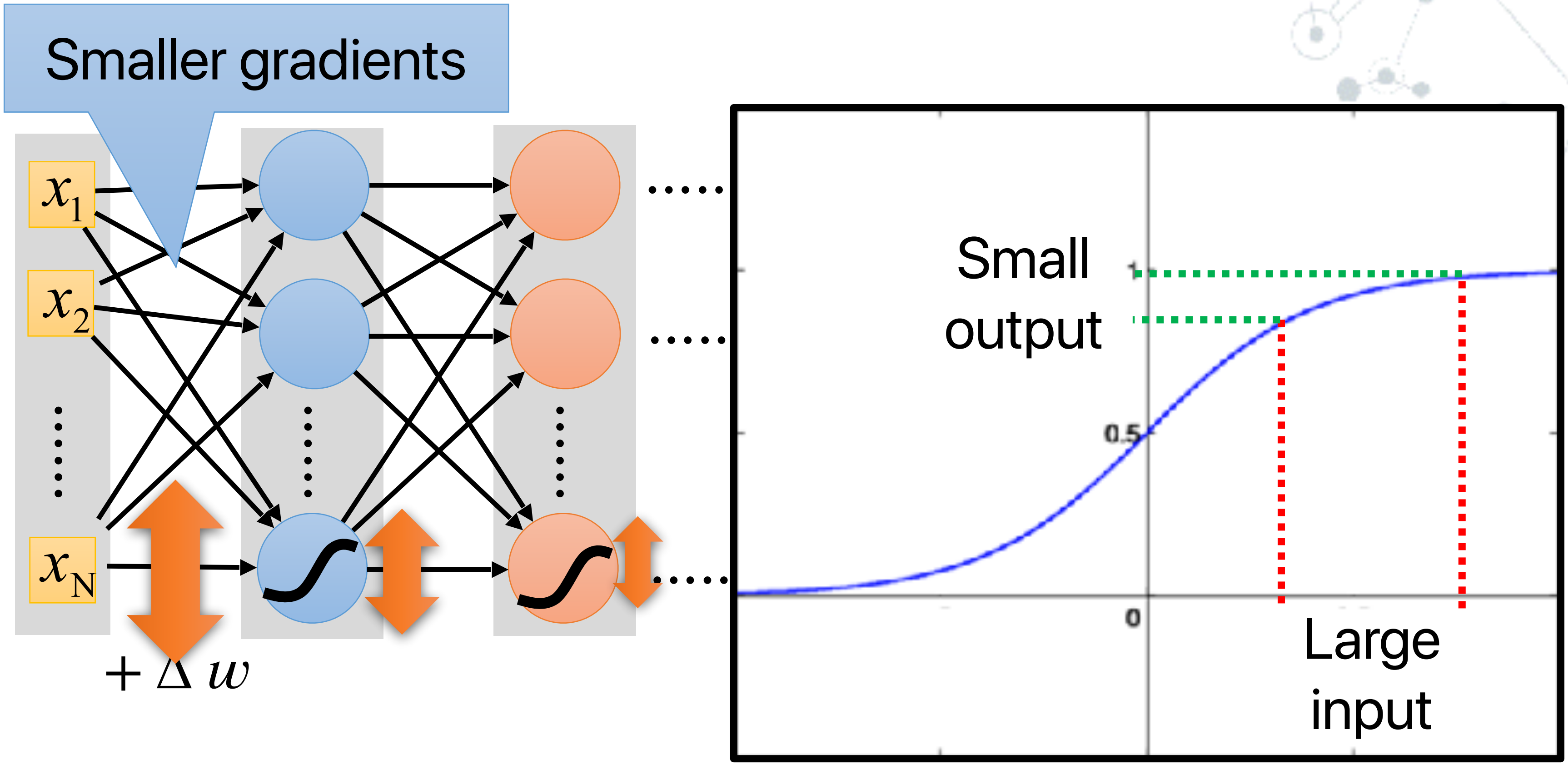


In 2006, people used RBM pre-training.
In 2015, people use ReLU.

Learn very slow	Learn very fast
Almost random	Already converge

based on random!?

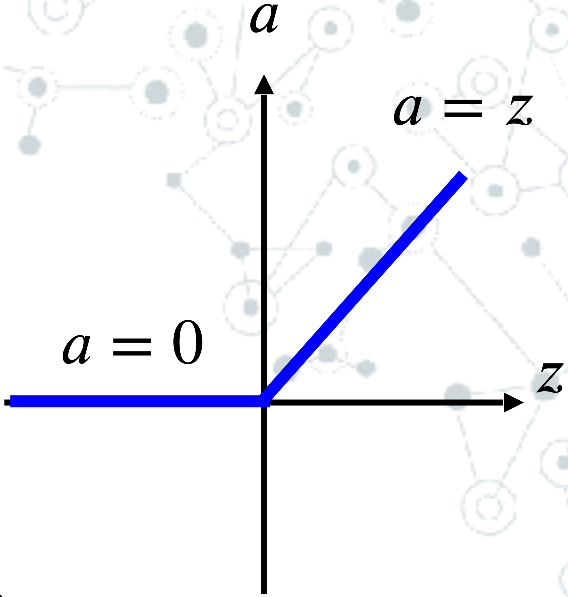
Vanishing Gradient Problem

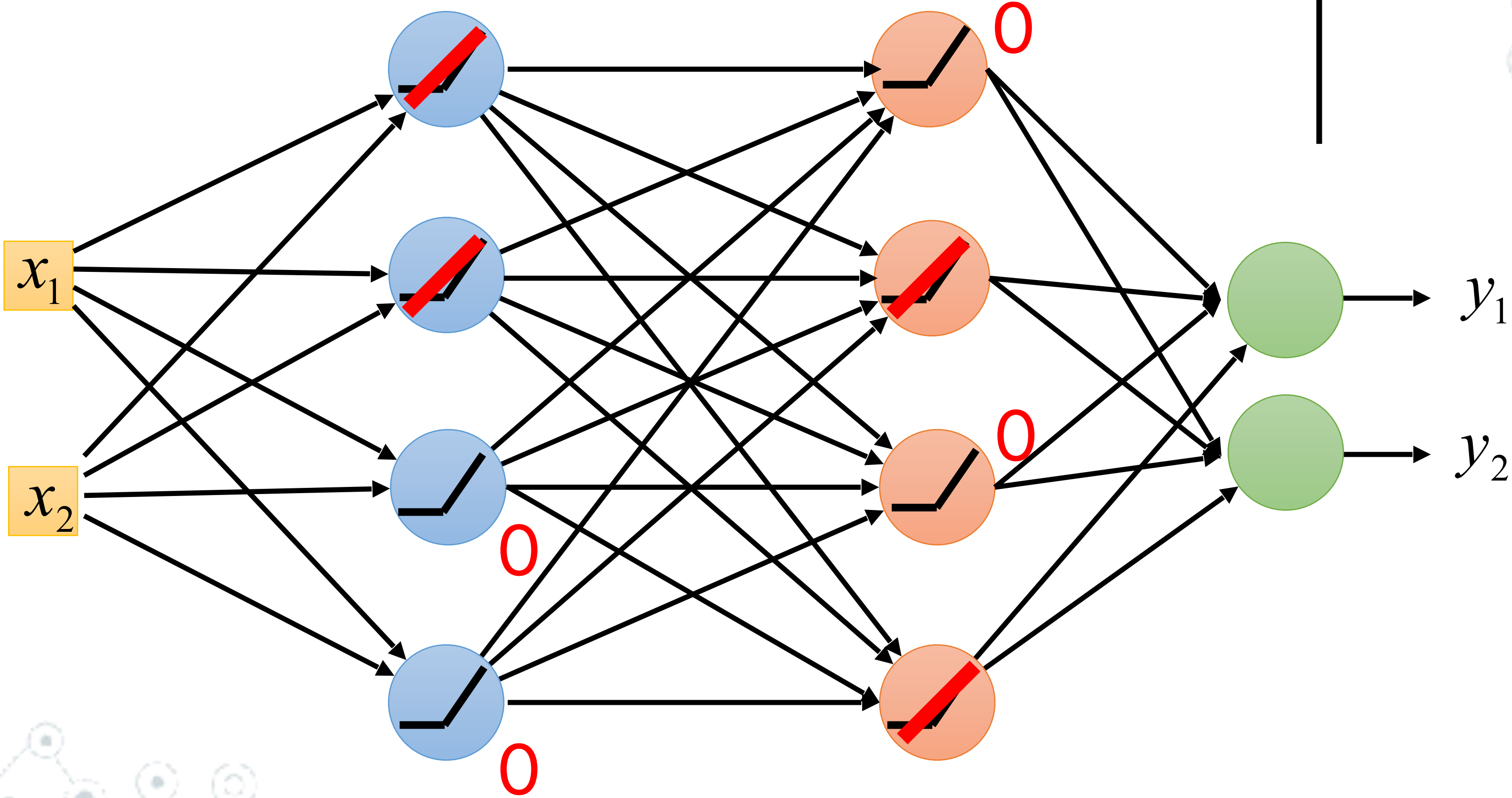


Intuitive way to compute the gradient ...

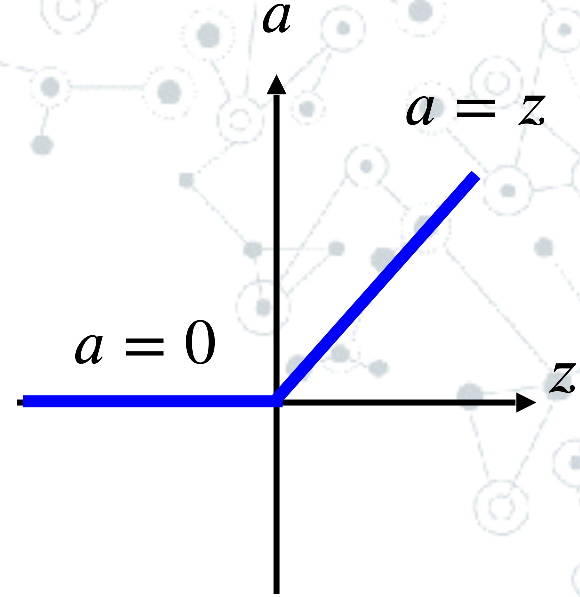
$$\frac{\partial C}{\partial w} = ? \frac{\Delta C}{\Delta w}$$

ReLU

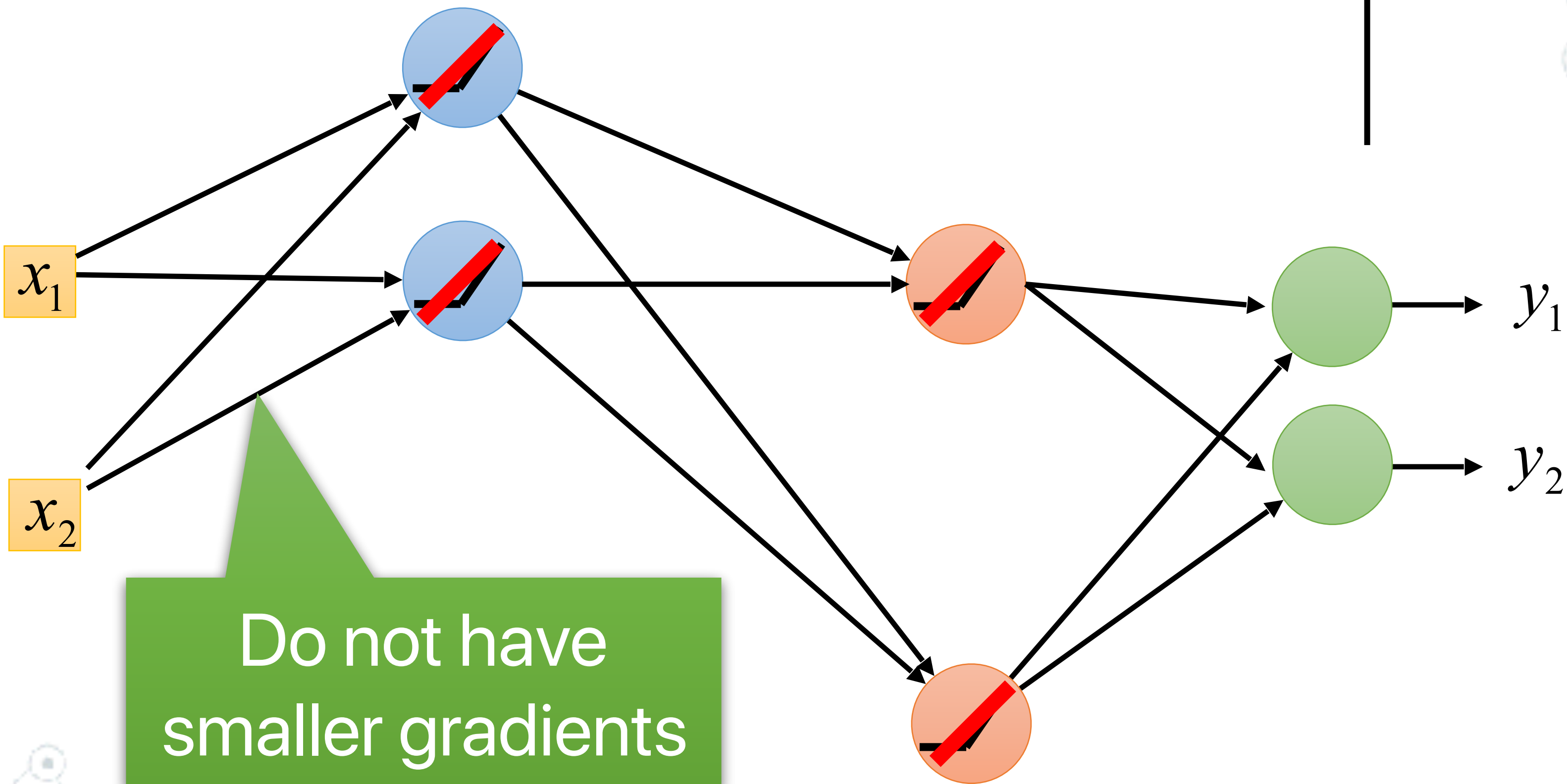




ReLU



A Thinner linear network

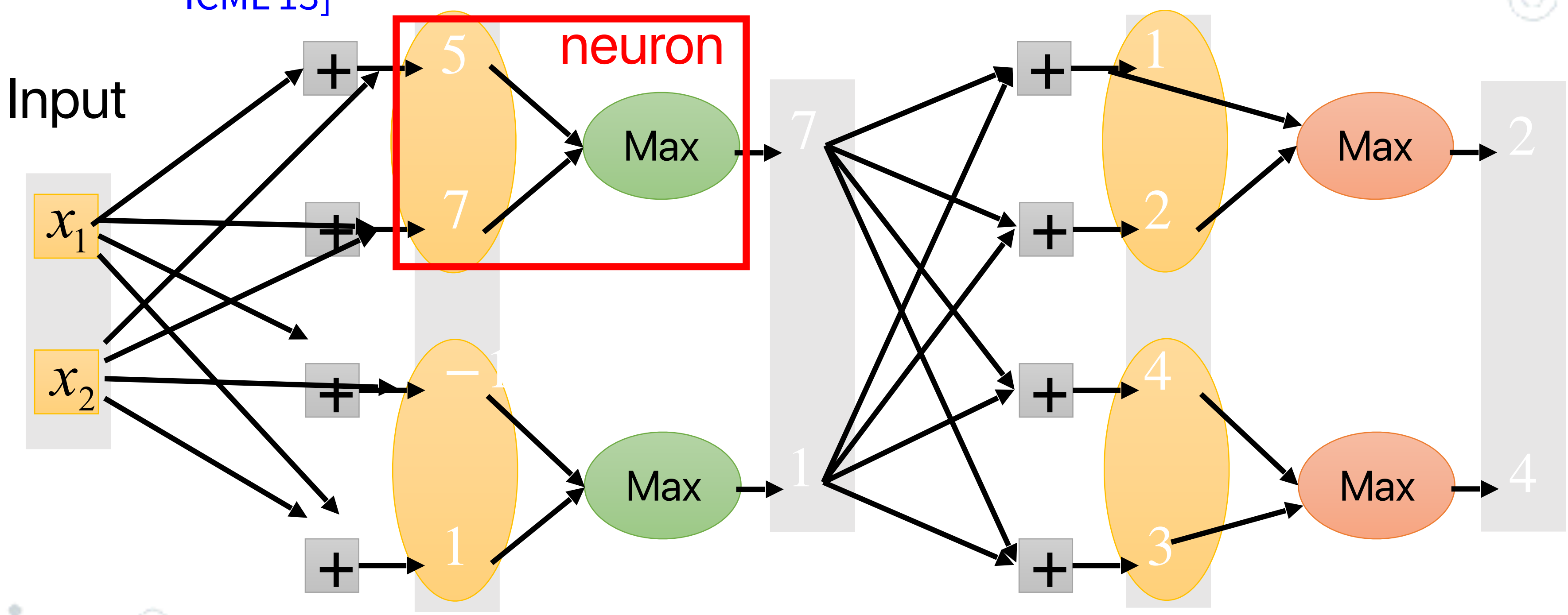


Do not have smaller gradients

Maxout

ReLU is a special cases of Maxout

◎ Learnable activation function [Ian J. Goodfellow, ICML’13]



You can have more than 2 elements in a group.

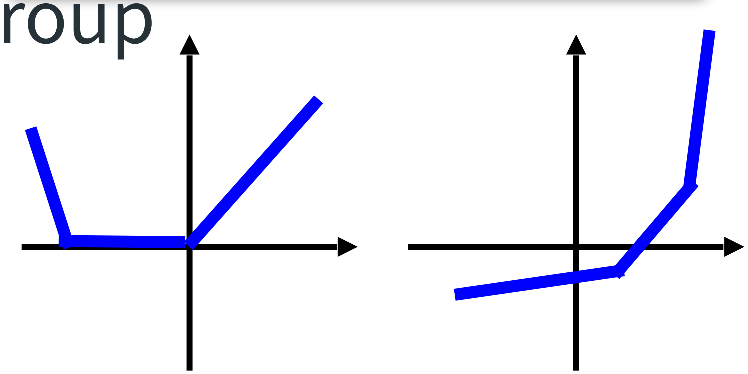
Maxout

ReLU is a special cases of Maxout

- Learnable activation function [Ian J. Goodfellow, ICML'13]
  - Activation function in maxout network can be any piecewise linear convex function
  - ...s dep... many elements in a group

2 elements in a group

3 elements in a group

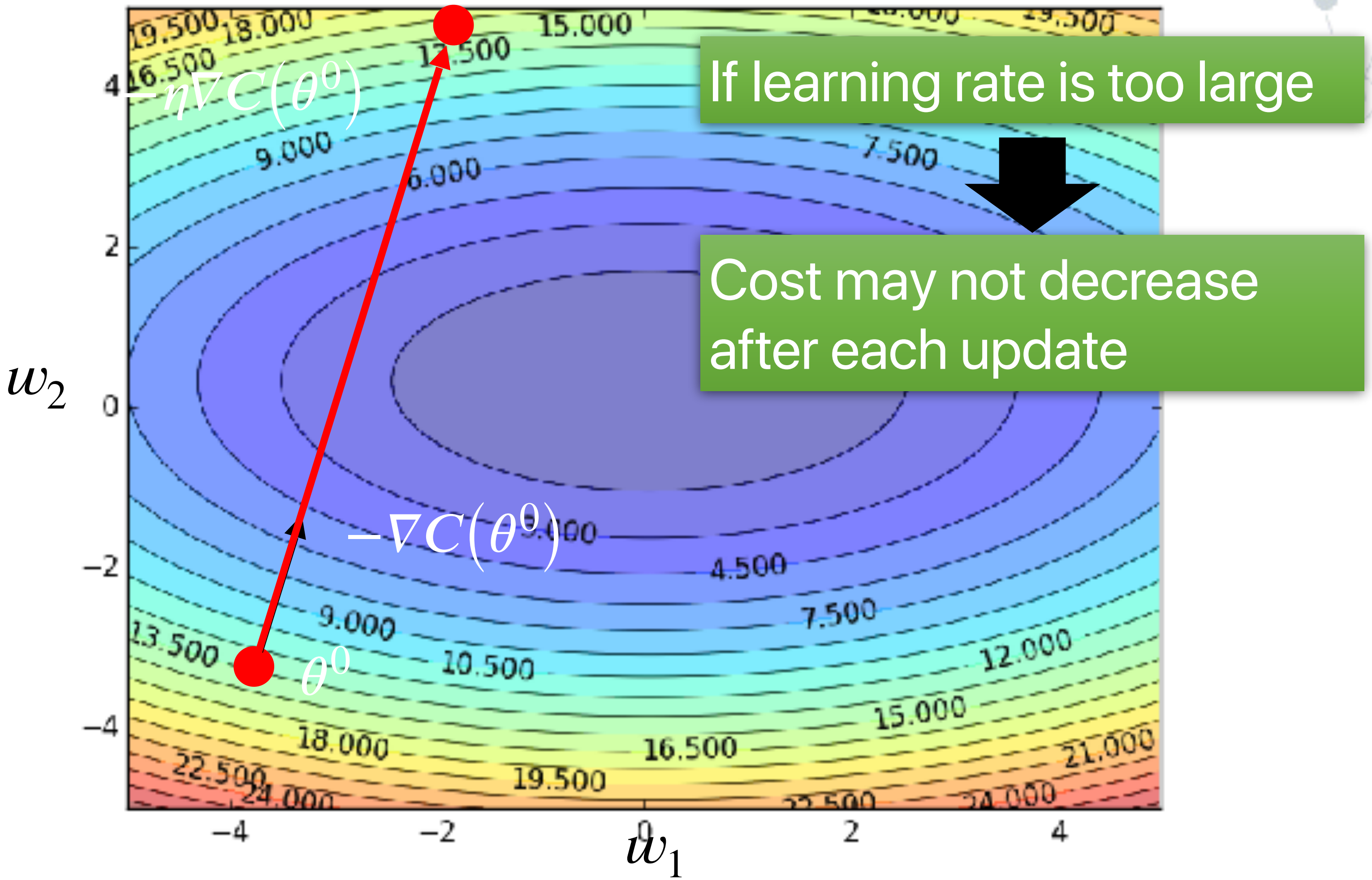


Part III:
Tips for Training DNN

Adaptive Learning Rate

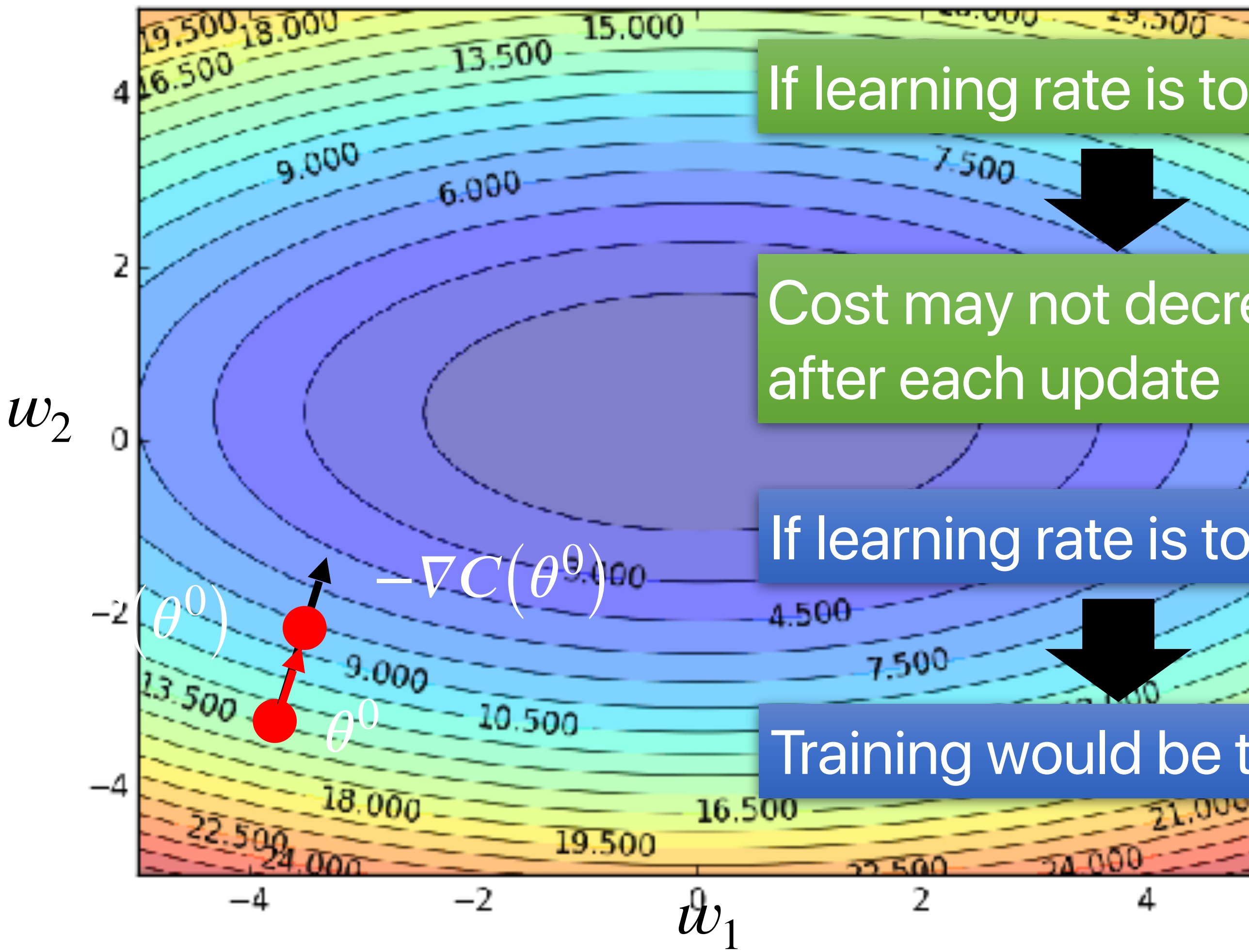
Learning Rate

Set the learning rate η carefully



Learning Rate

Can we give different parameters different learning rates?



If learning rate is too large

Cost may not decrease after each update

If learning rate is too small

Training would be too slow

Adagrad

Original Gradient Descent

$$\theta^t \leftarrow \theta^{t-1} - \eta \nabla C(\theta^{t-1})$$

Each parameter w are considered separately

$$w^{t+1} \leftarrow w^t - \boxed{\eta_w} g^t \qquad g^t = \frac{\partial C(\theta^t)}{\partial w}$$

Parameter dependent learning rate

$$\eta_w = \frac{\eta}{\sqrt{\boxed{\sum_{i=0}^{t}\left(g^i\right)^2}}}$$

constant

Summation of the square of the previous derivatives

Adagrad

$$\eta_w = \boxed{\frac{\eta}{\sqrt{\sum_{i=0}^{t}\left(g^i\right)^2}}}$$

w_1

g^0
0.1

Learning rate:

$$\frac{\eta}{\sqrt{0.1^2}} = \frac{\eta}{0.1}$$

$$\frac{\eta}{\sqrt{0.1^2 + 0.2^2}} = \frac{\eta}{0.22}$$

w_2

g^0
20.0

Learning rate:

$$\frac{\eta}{\sqrt{20^2}} = \frac{\eta}{20}$$

$$\frac{\eta}{\sqrt{20^2 + 10^2}} = \frac{\eta}{22}$$

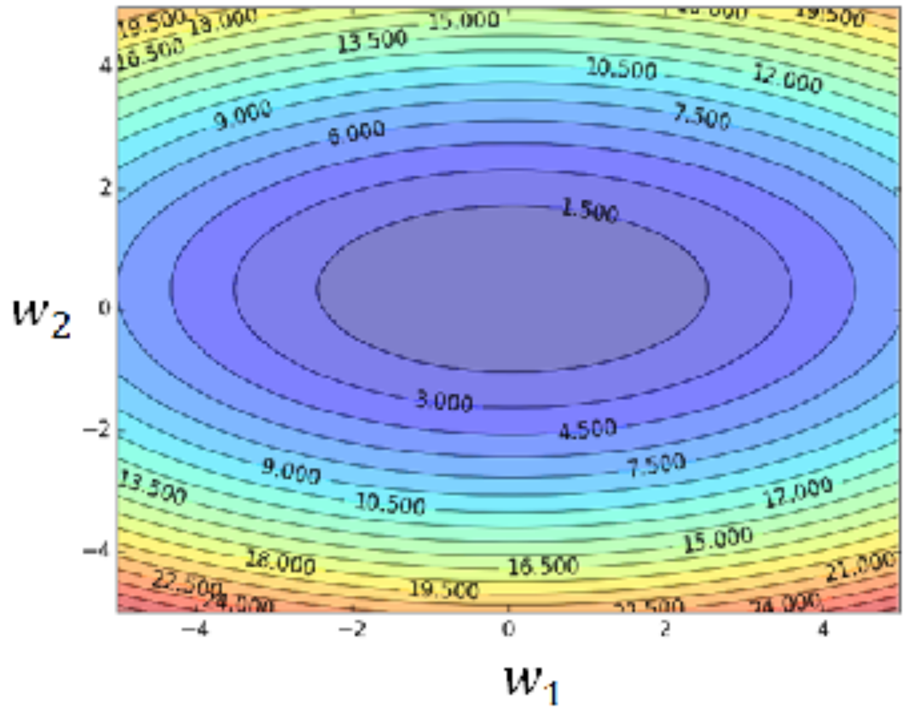
Observation: 1. Learning rate is smaller and smaller for all parameters

2. Smaller derivatives, larger learning rate, and vice versa

Why?

Larger derivatives

Smaller Learning Rate



Smaller Derivatives

Larger Learning Rate

2. Smaller derivatives, larger le

Why?

rate, and vice versa

Not the whole story ......

- Adagrad [John Duchi, JMLR'11]
- RMSprop
  - https://www.youtube.com/watch?v=O3sxAc4hxZU
- Adadelta [Matthew D. Zeiler, arXiv'12]
- Adam [Diederik P. Kingma, ICLR'15]
- AdaSecant [Caglar Gulcehre, arXiv'14]
- "No more pesky learning rates" [Tom Schaul, arXiv'12]

Part III:
Tips for Training DNN

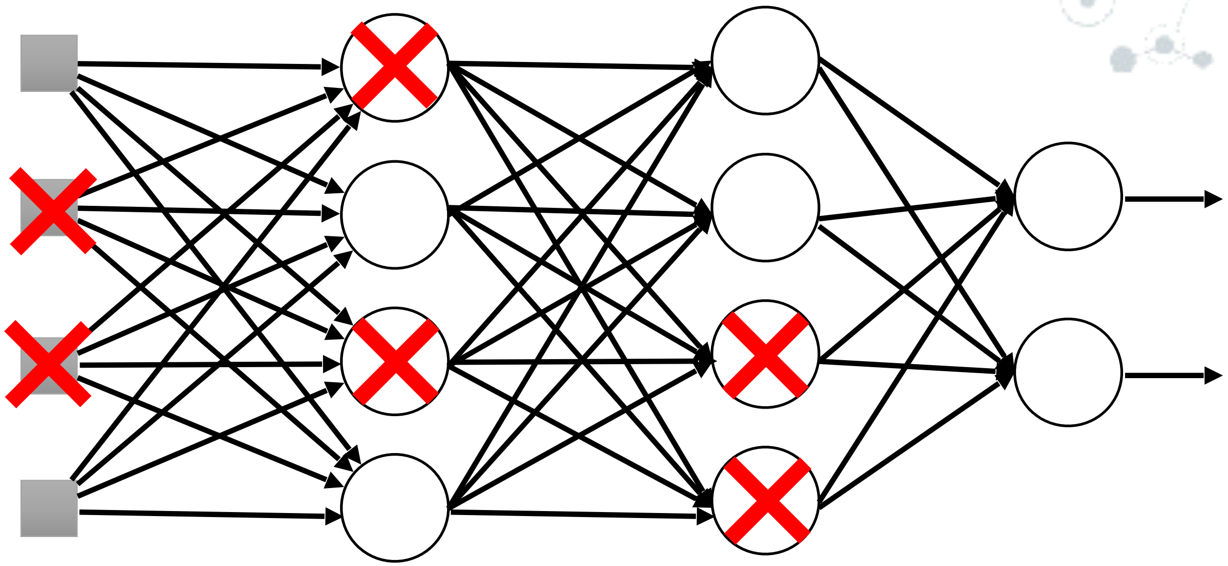
Dropout

Dropout

Pick a mini-batch

$$\theta^t \leftarrow \theta^{t-1} - \eta \nabla C\left(\theta^{t-1}\right)$$

Training:



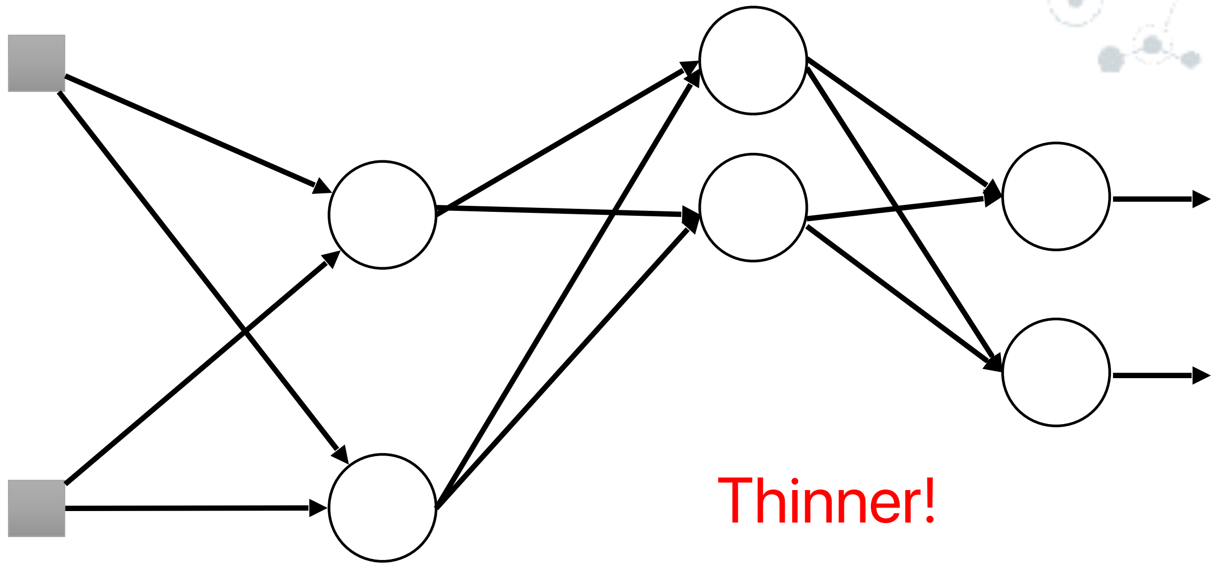
- **Each time before computing the gradients**
  - Each neuron has p% to dropout

Dropout

Pick a mini-batch

$$\theta^t \leftarrow \theta^{t-1} - \eta \nabla C\left(\theta^{t-1}\right)$$

Training:



Thinner!

➢ **Each time before computing the gradients**

- Each neuron has p% to dropout

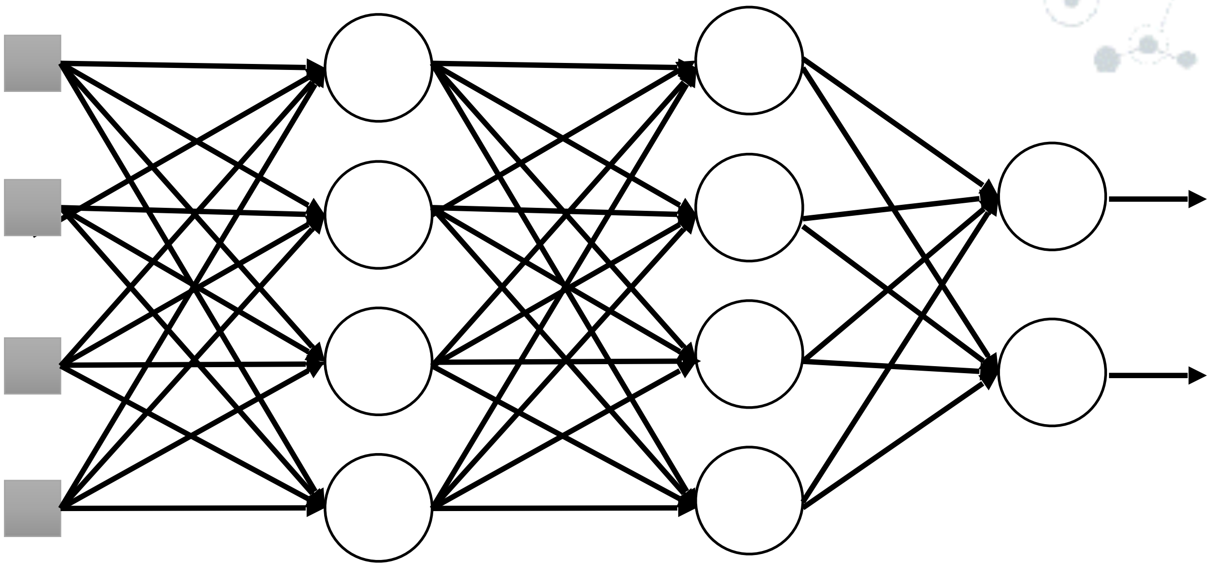
The structure of the network is changed.

- Using the new network for training

For each mini-batch, we resample the dropout neurons

Dropout

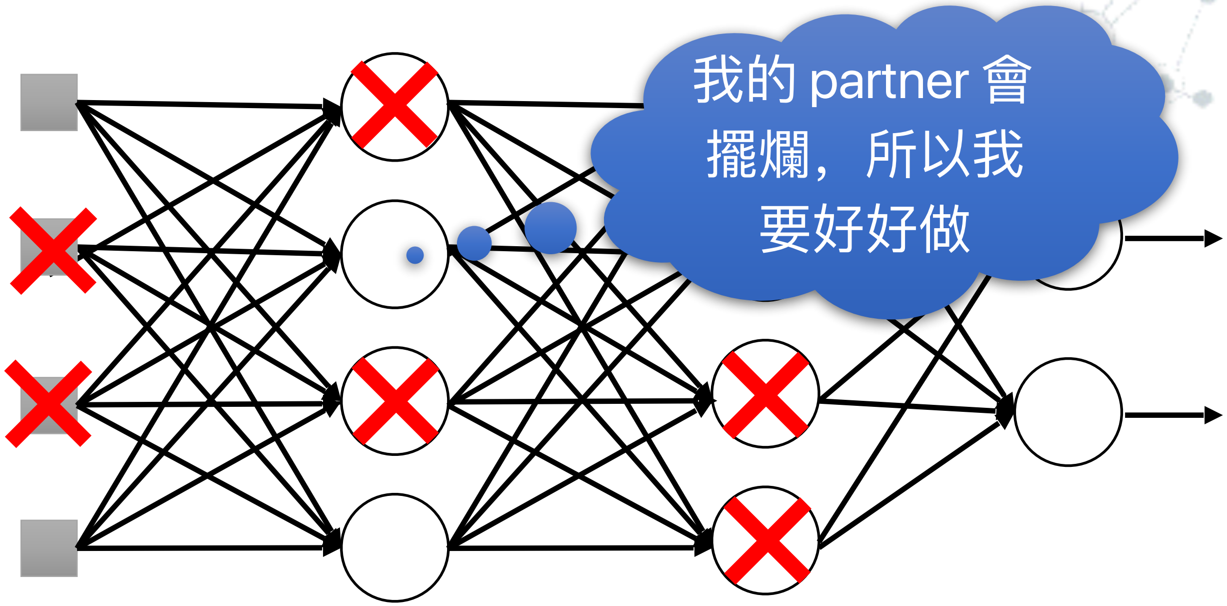
Testing:



➢ **No dropout**

- If the dropout rate at training is p%, all the weights times (1-p)%
- Assume that the dropout rate is 50%. If a weight   by training, set for testing.

Dropout - Intuitive Reason



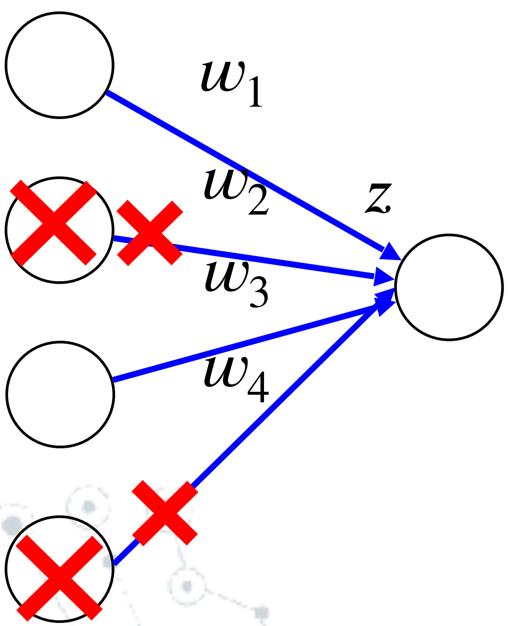
- When teams up, if everyone expect the partner will do the work, nothing will be done finally.
- However, if you know your partner will dropout, you will do better.
- When testing, no one dropout actually, so obtaining good results eventually.

Dropout - Intuitive Reason

◎ Why the weights should multiply (1-p)% (dropout rate) when testing?

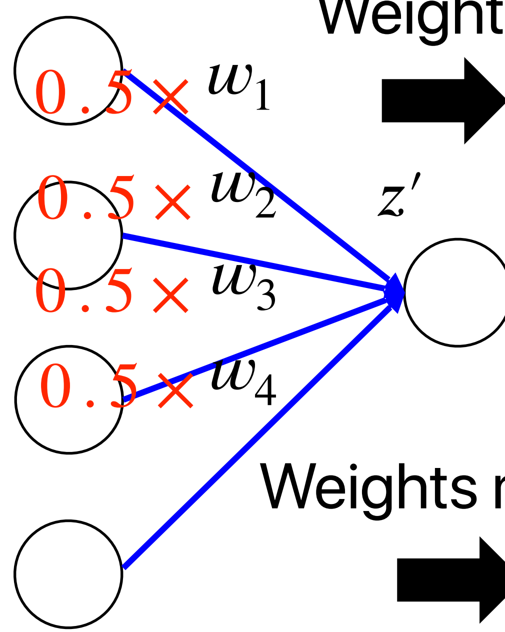
Training of Dropout

Assume dropout rate is 50%



Testing of Dropout

No dropout



Weights from training

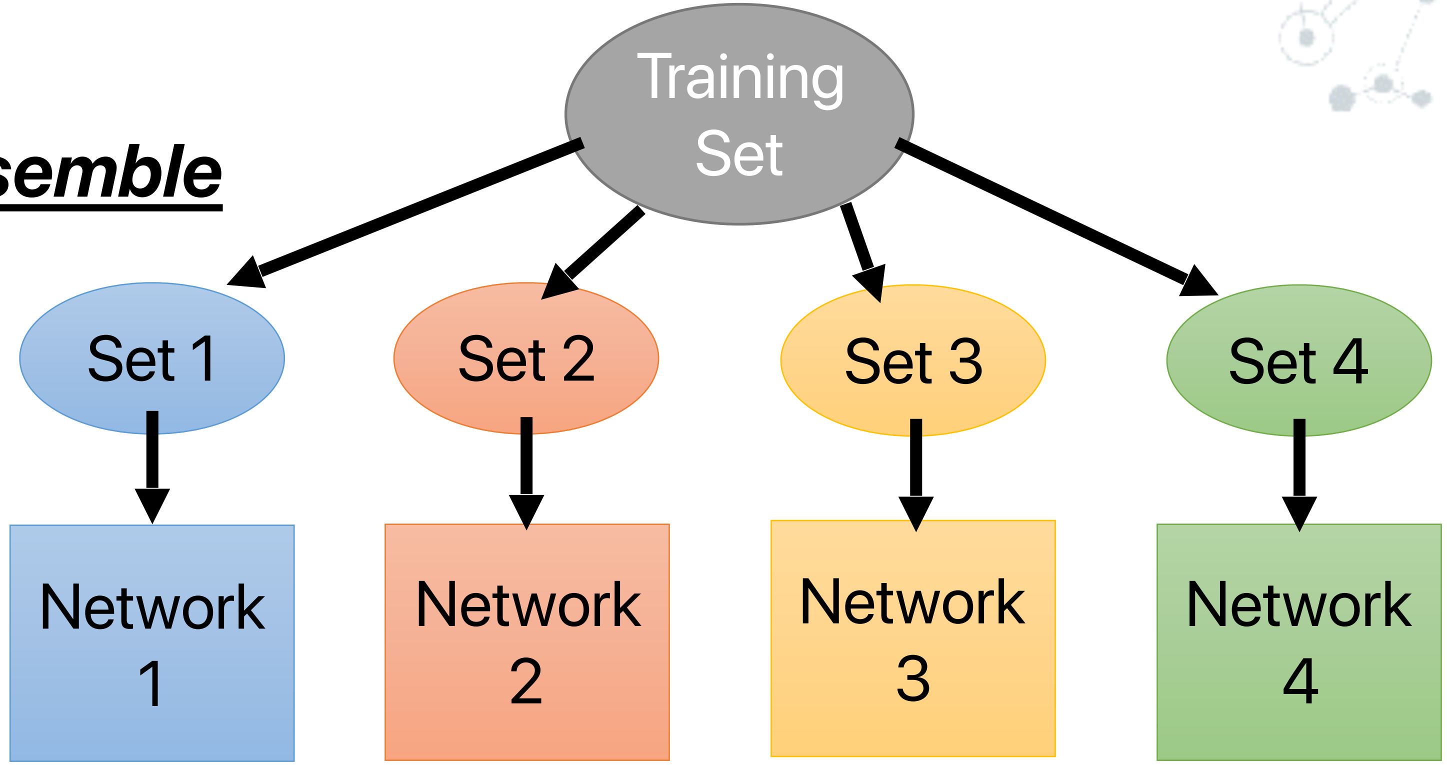
$z' \approx 2z$

Weights multiply (1-p)%

$z' \approx z$

Dropout is a kind of ensemble.

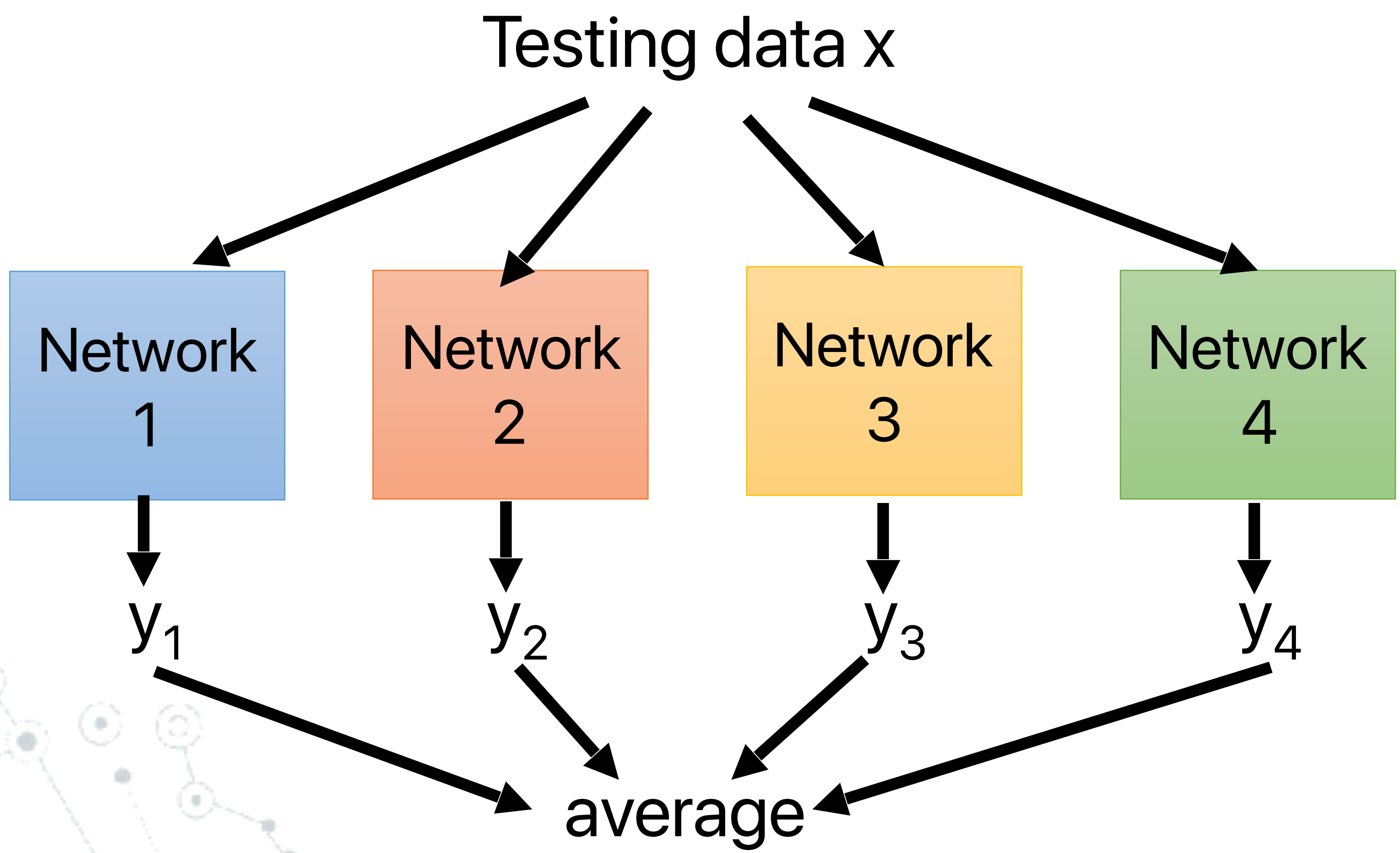
Ensemble



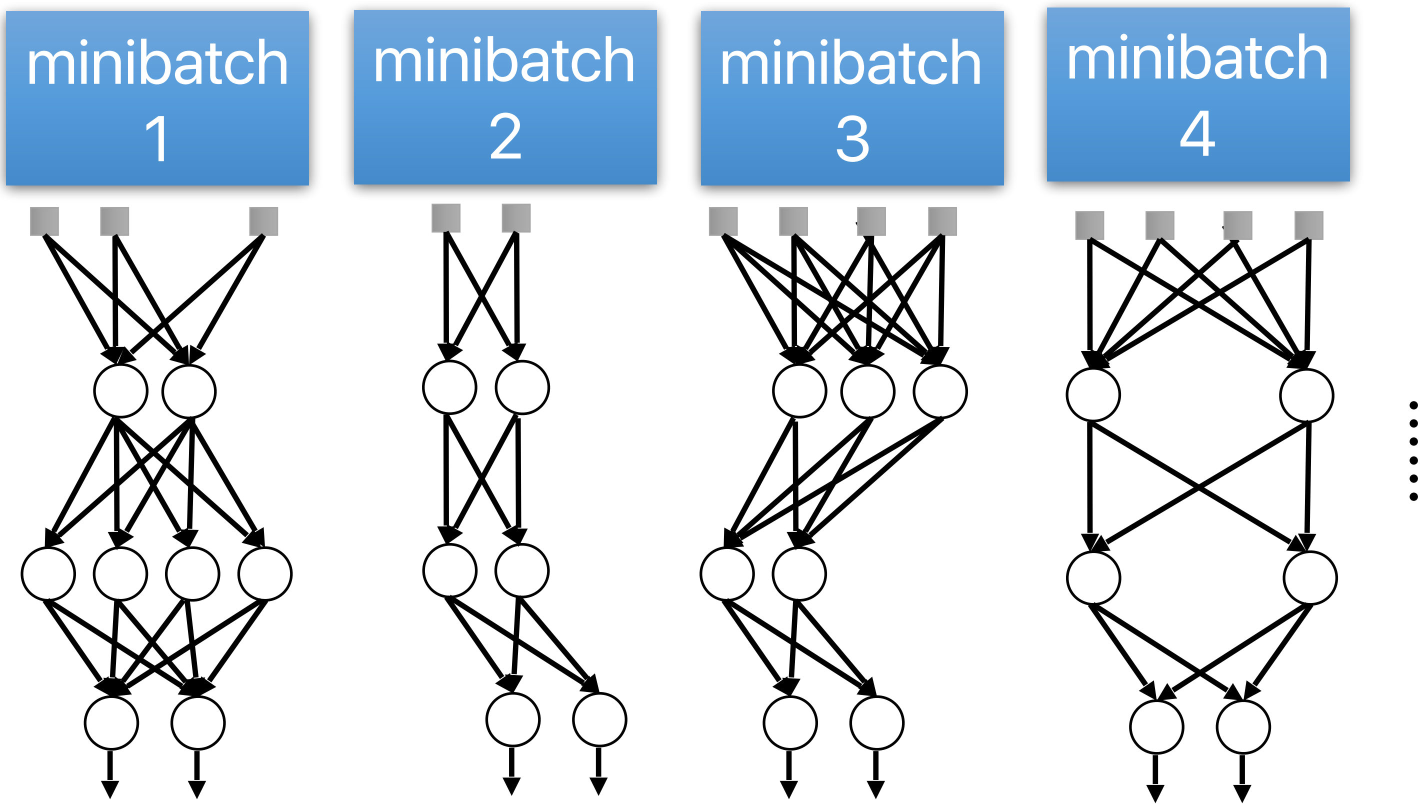
Train a bunch of networks with different structures

Dropout is a kind of ensemble.

Ensemble



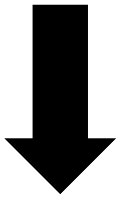
Dropout is a kind of ensemble.



Training of Dropout

M neurons

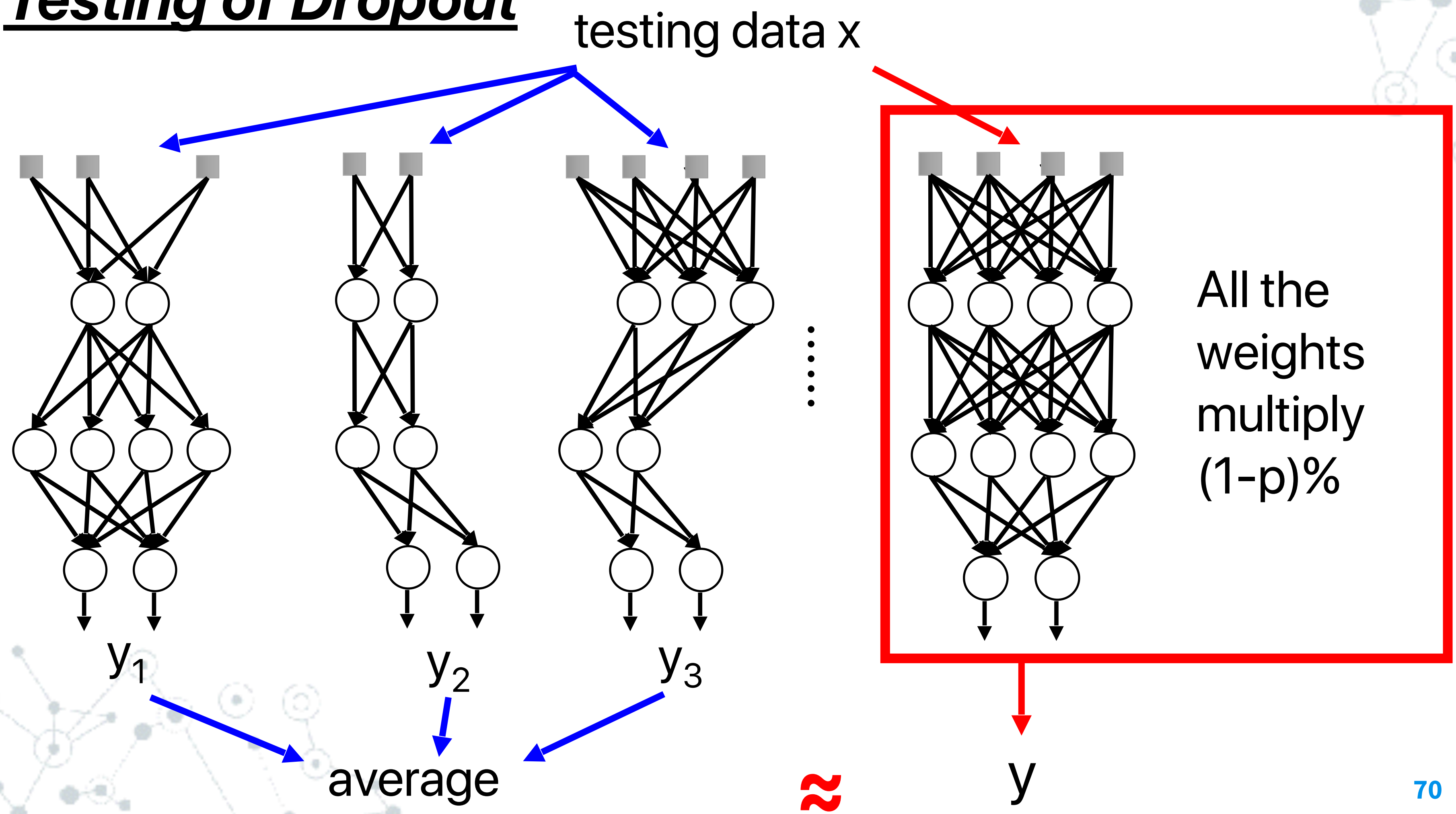
2^M possible networks



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

Dropout is a kind of ensemble.

Testing of Dropout



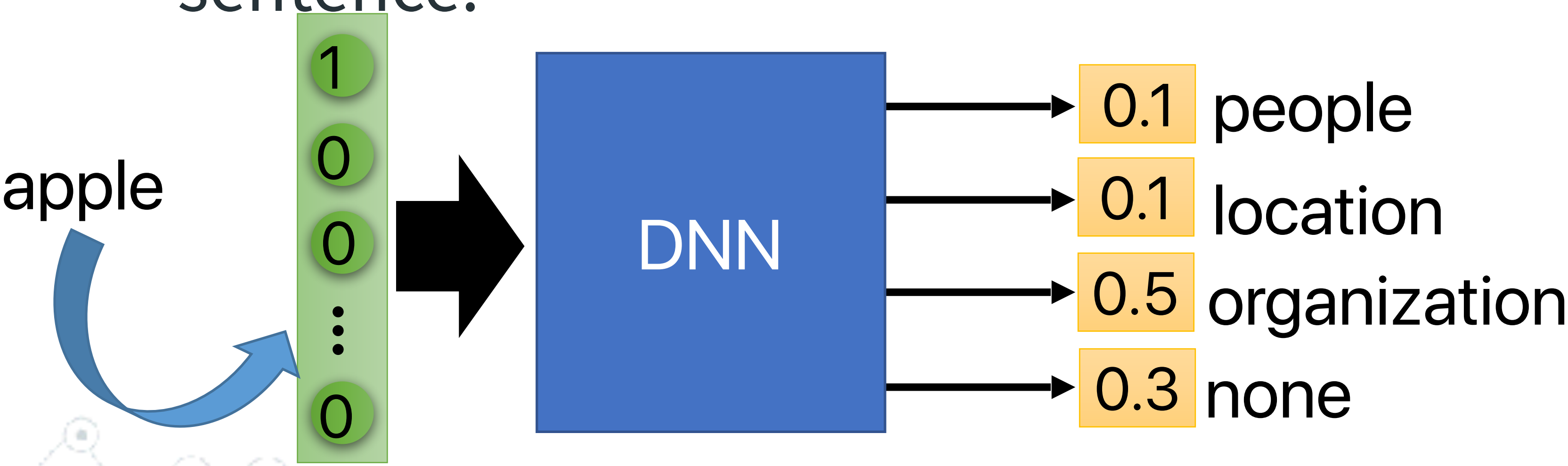
More about dropout

- More reference for dropout [Nitish Srivastava, JMLR'14] [Pierre Baldi, NIPS'13][Geoffrey E. Hinton, arXiv'12]
- Dropout works better with Maxout [Ian J. Goodfellow, ICML'13]
- Dropconnect [Li Wan, *ICML'13*]
  - Dropout delete neurons
  - Dropconnect deletes the connection between neurons
- Annealed dropout [S.J. Rennie, SLT'14]
  - Dropout rate decreases by epochs
- Standout [J. Ba, NISP'13]
  - Each neural has different dropout rate

Part IV:
Neural Network with Memory

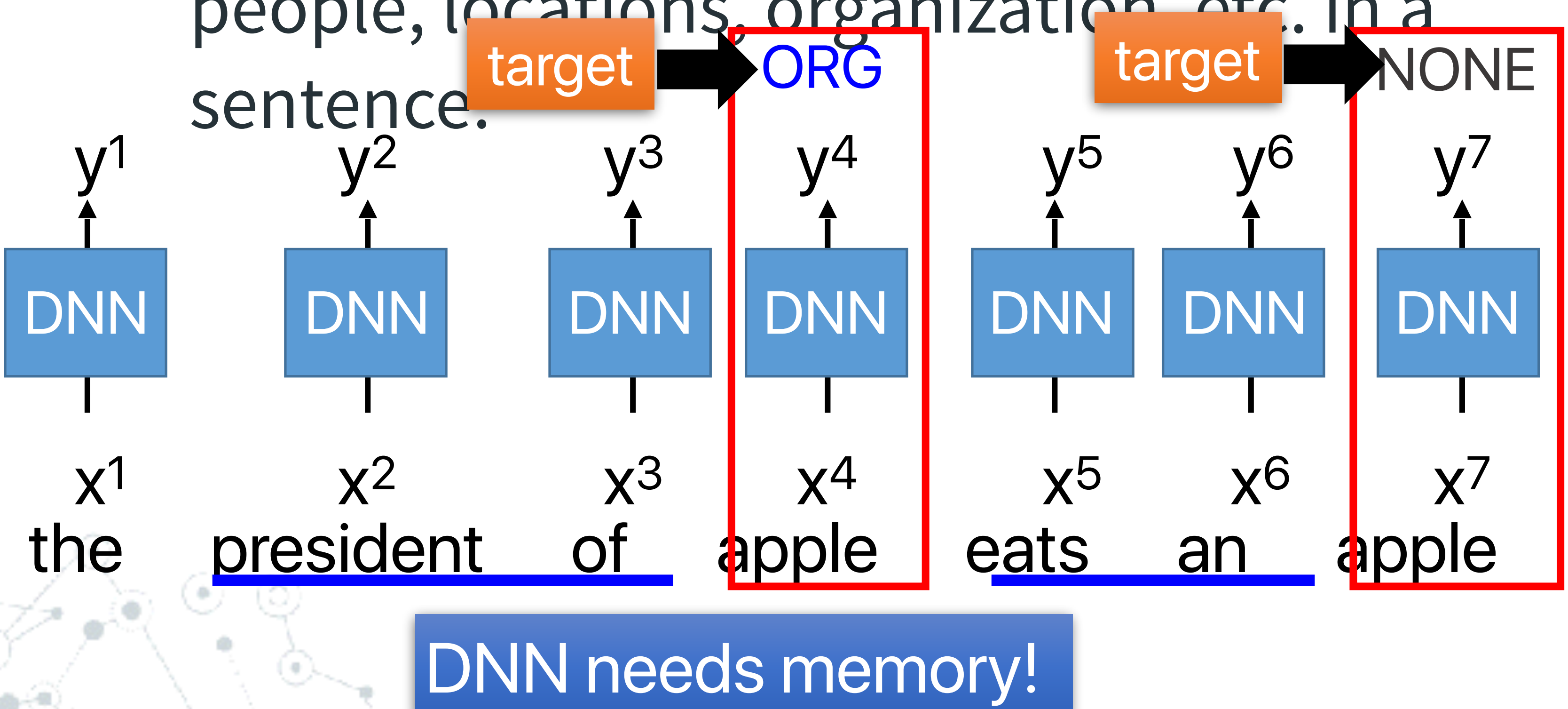
Neural Network needs Memory

- Name Entity Recognition
  - Detecting named entities like name of people, locations, organization, etc. in a sentence.



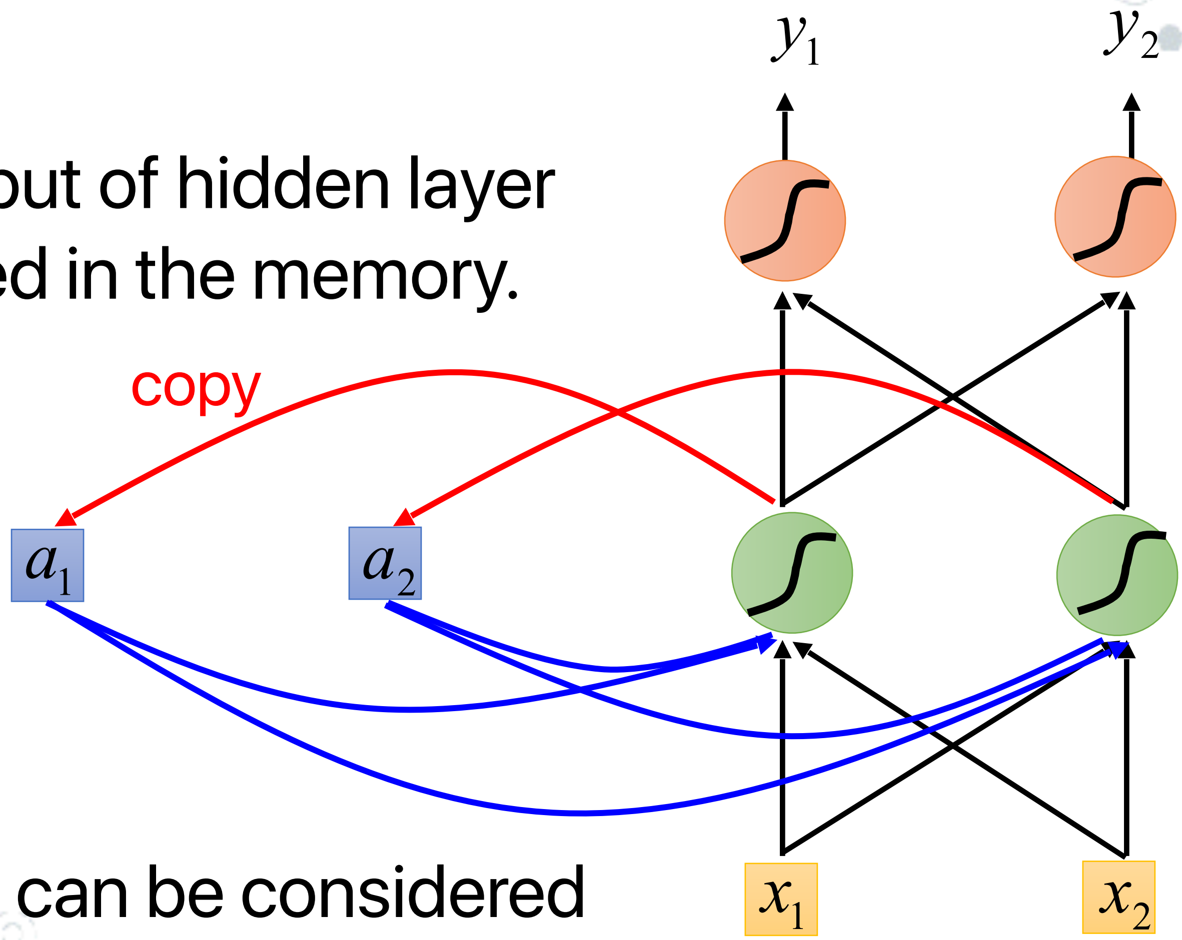
Neural Network needs Memory

- Name Entity Recognition
  - Detecting named entities like name of people, locations, organization, etc. in a sentence.



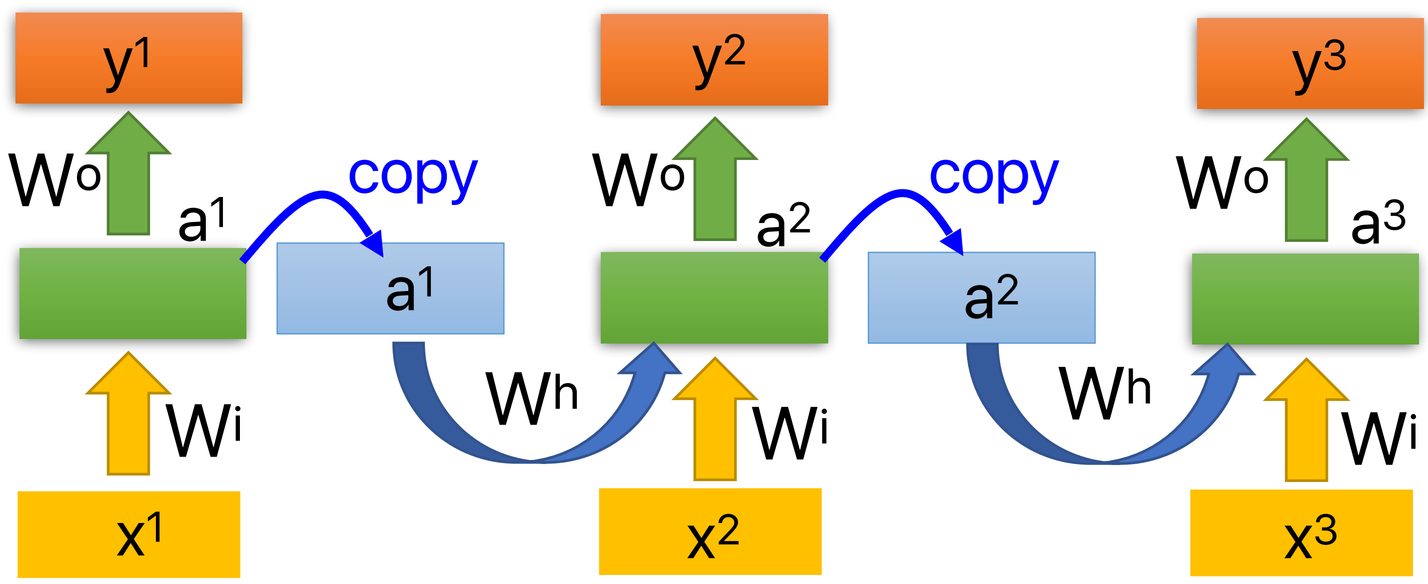
Recurrent Neural Network (RNN)

The output of hidden layer are stored in the memory.



Memory can be considered as another input.

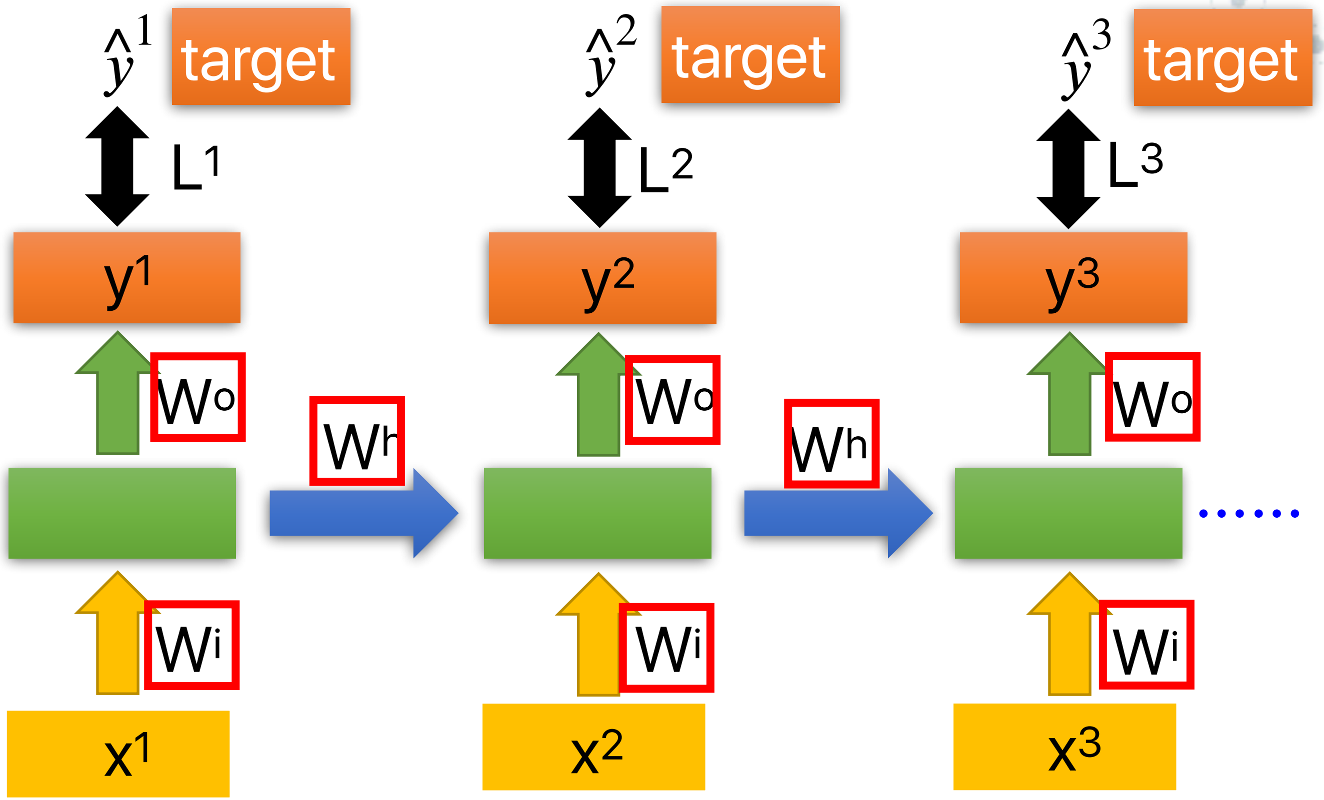
RNN



The same network is used again and again.

Output y^i depends on $x^1, x^2, \ldots\ldots x^i$

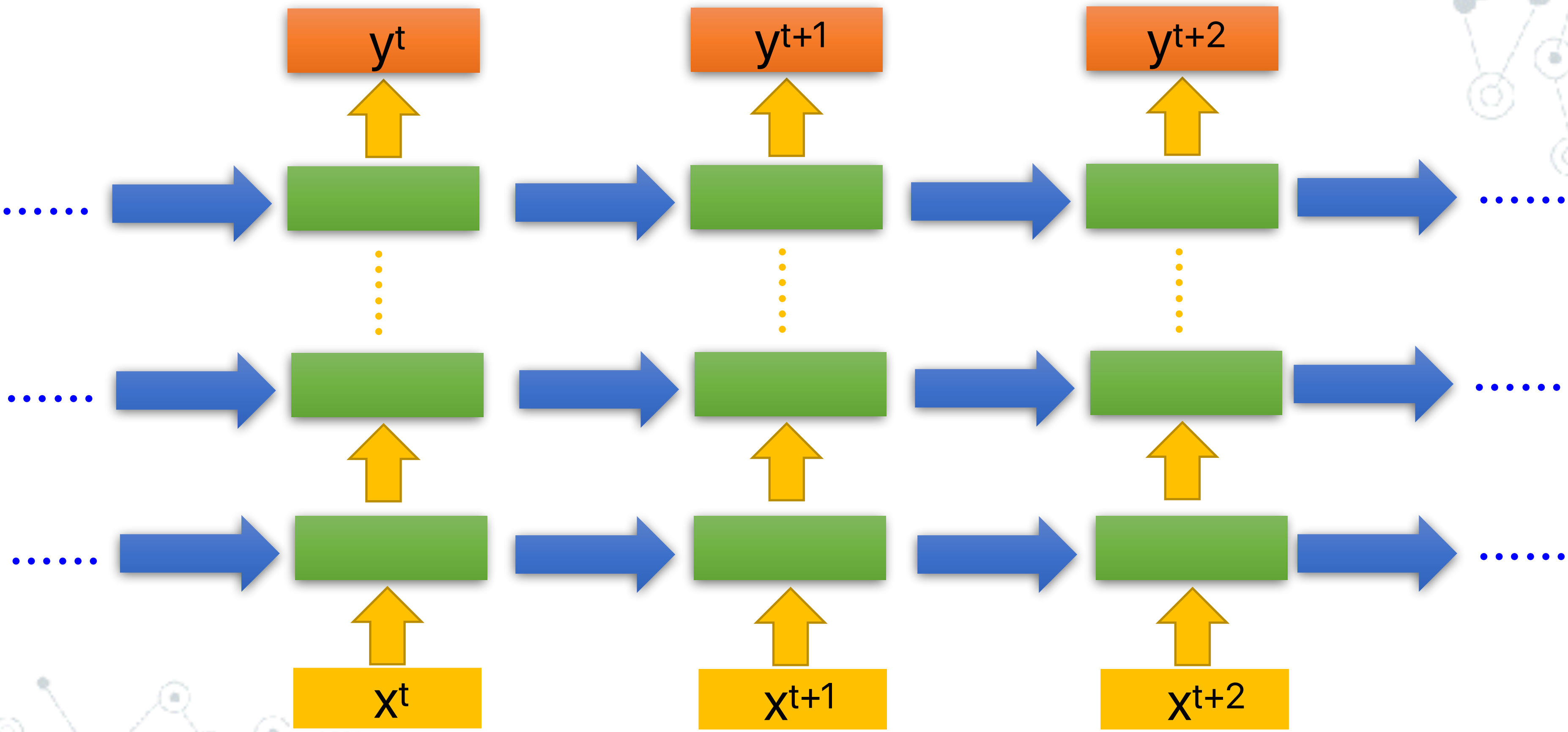
RNN How to train?



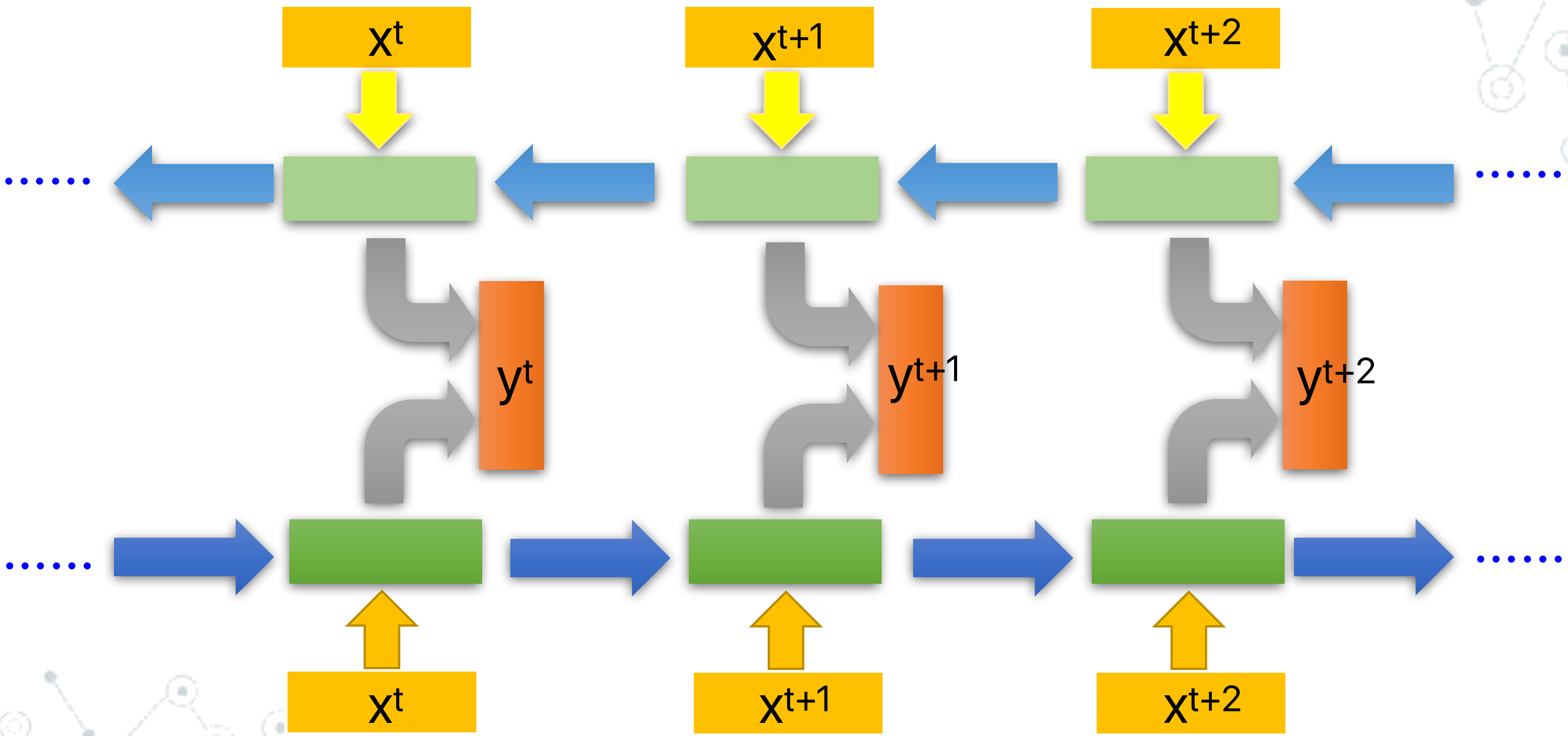
Find the network parameters to minimize the total cost:

Backpropagation through time (BPTT)

Of course it can be deep ...

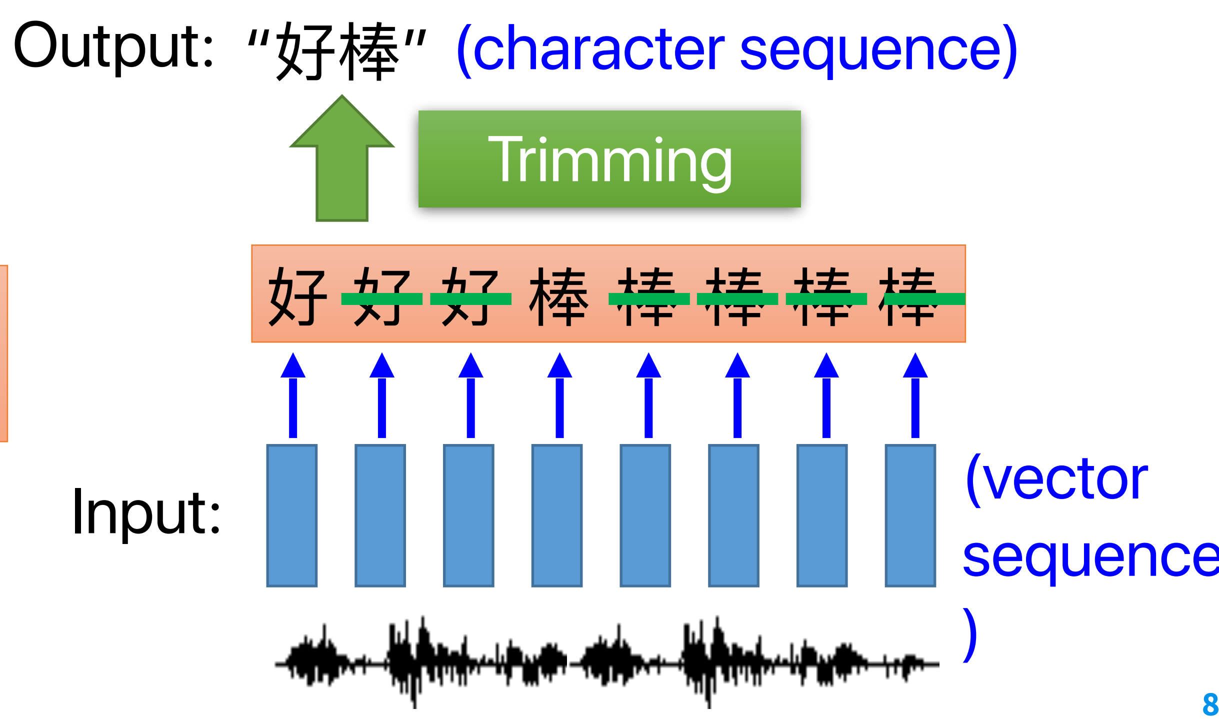


Bidirectional RNN



Many to Many (Output is shorter)

- Both input and output are both sequences, ***<u>but the output is shorter.</u>***
  - E.g. ***<u>Speech Recognition</u>***



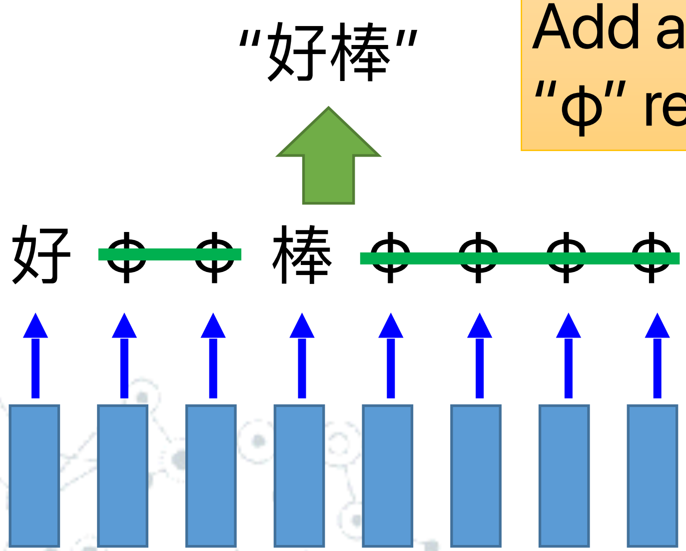
Problem?

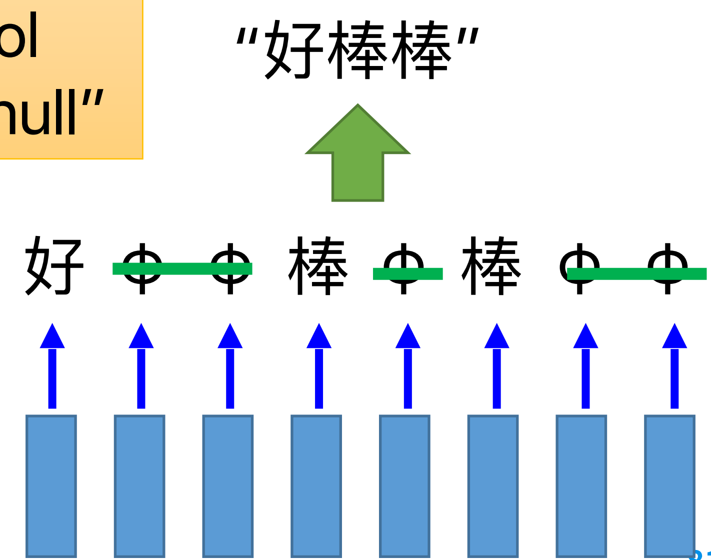
Why can’t it be “好棒棒”

Many to Many (Output is shorter)

- Both input and output are both sequences, ***but the output is shorter.***
  - Connectionist Temporal Classification (CTC) [Alex Graves, ICML’06][Alex Graves, ICML’14][Haşim Sak, Interspeech’15][Jie Li, Interspeech’15][Andrew Senior, ASRU’15]

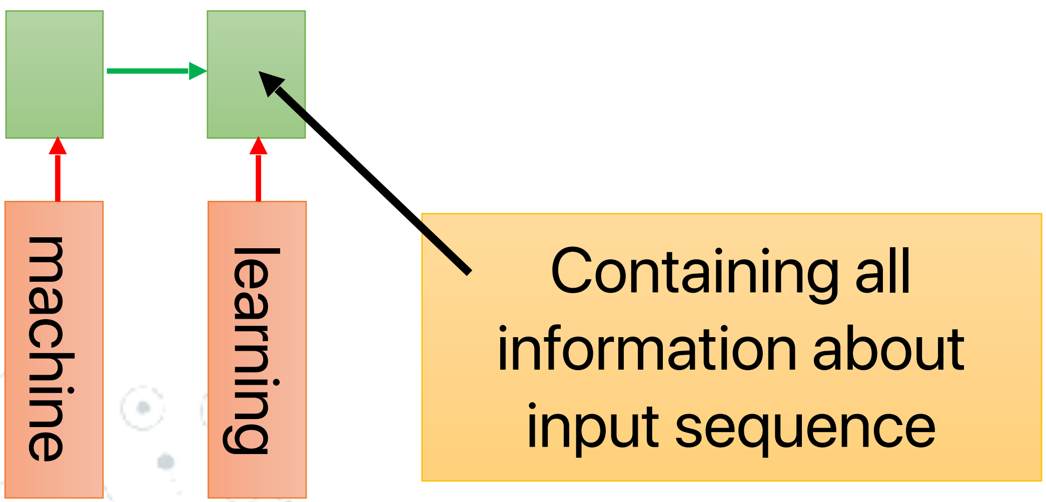
Add an extra symbol “φ” representing “null”





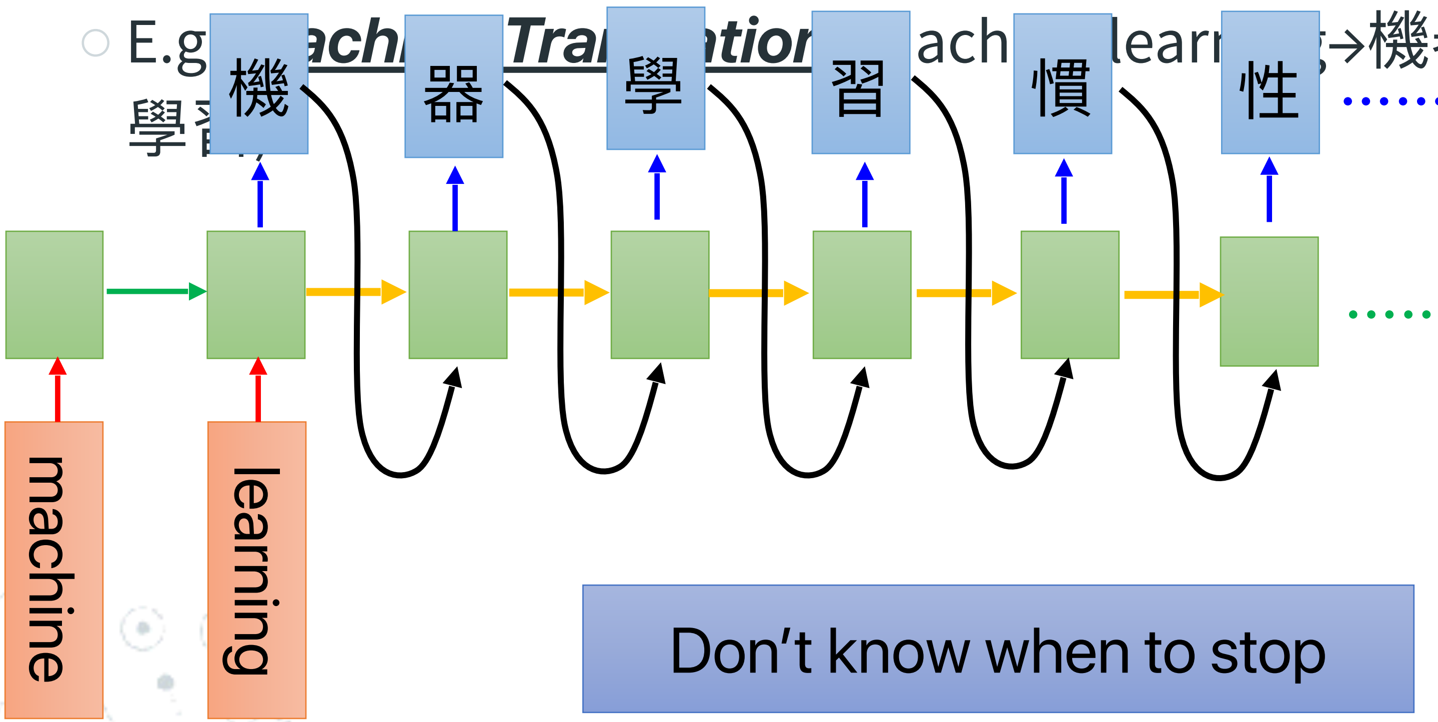
Many to Many (No Limitation)

- Both input and output are both sequences ***with different lengths***. → ***Sequence to sequence learning***
  - E.g. ***Machine Translation*** (machine learning→機器學習)



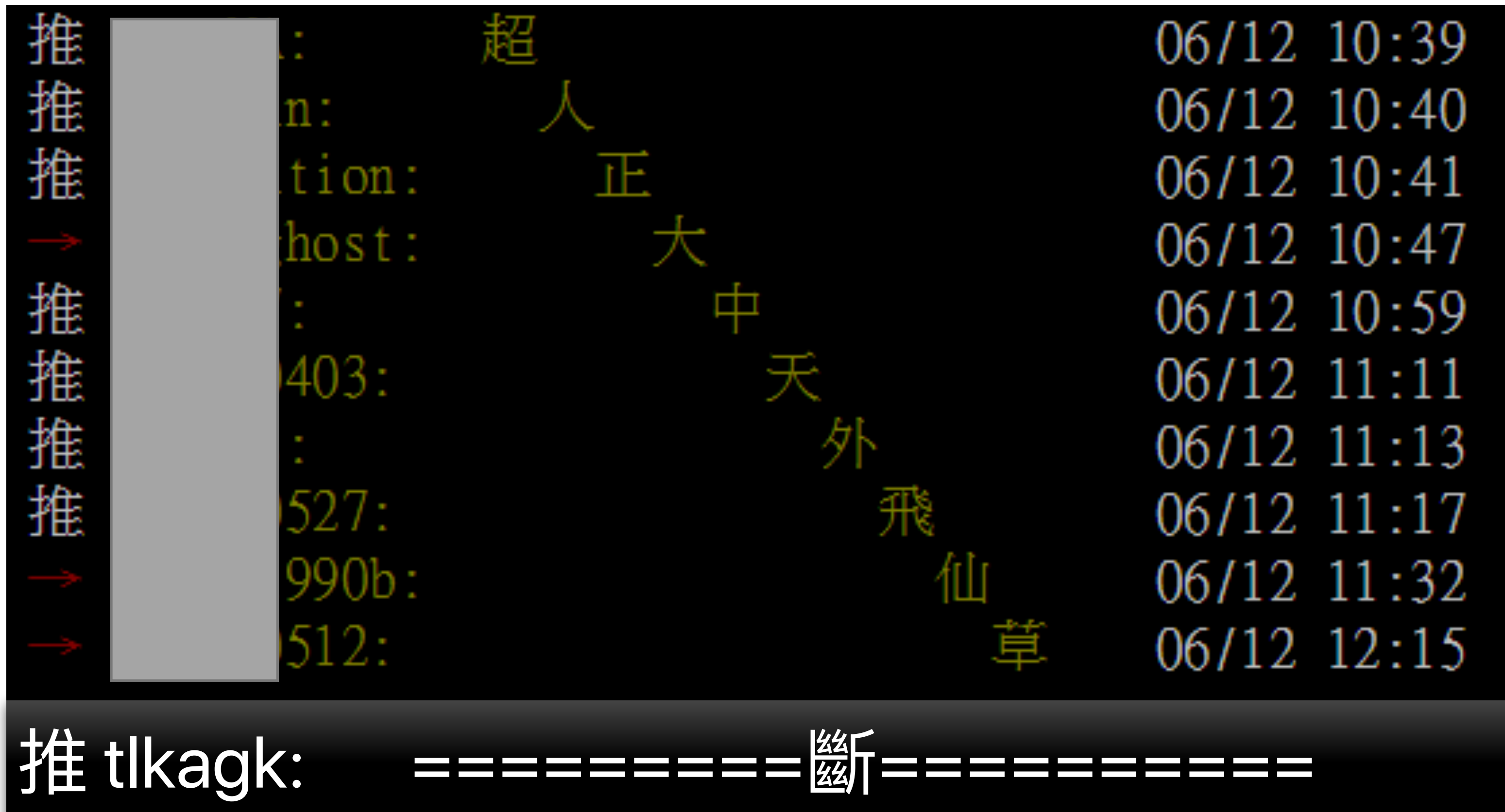
Many to Many (No Limitation)

- Both input and output are both sequences ***with different lengths***. → ***Sequence to sequence learning***
  - E.g. ***ach Tra tion*** (ach lear →機器學習)



Don’t know when to stop

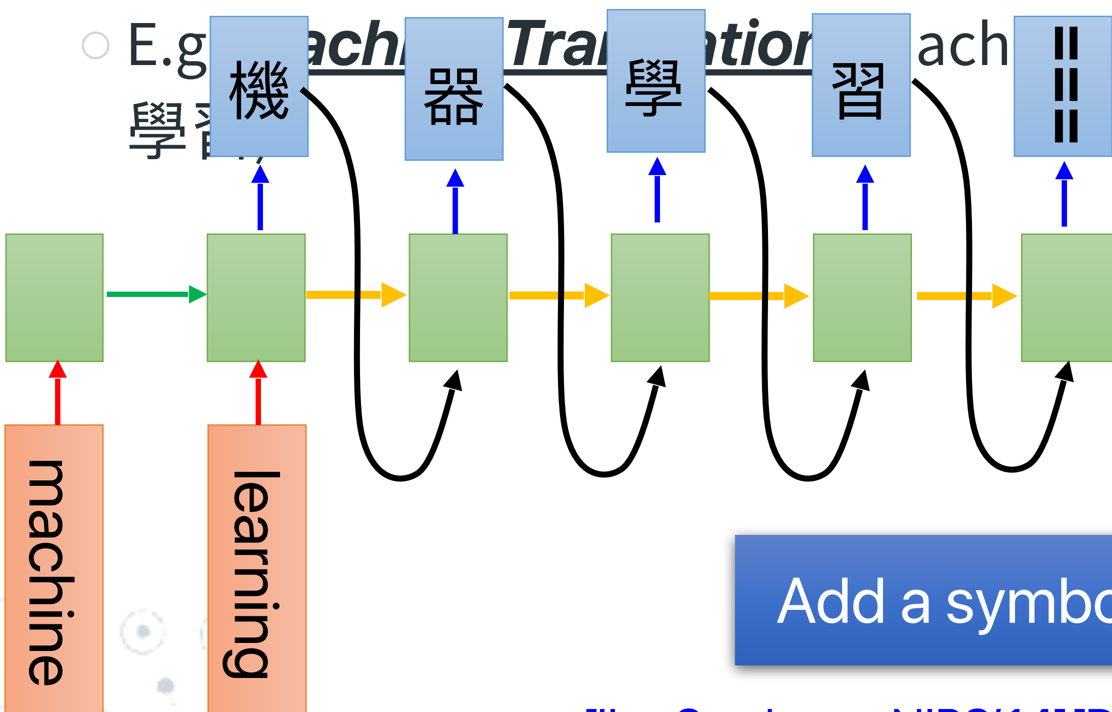
Many to Many (No Limitation)



Ref:http://zh.pttpedia.wikia.com/wiki/%E6%8E%A5%E9%BE%8D%E6%8E%A8%E6%96%87 (鄉民百科)

Many to Many (No Limitation)

- Both input and output are both sequences ***with different lengths***. → ***Sequence to sequence learning***
  - E.g. ***ach*** ***Tra*** ***tion*** ach learning→機器學習



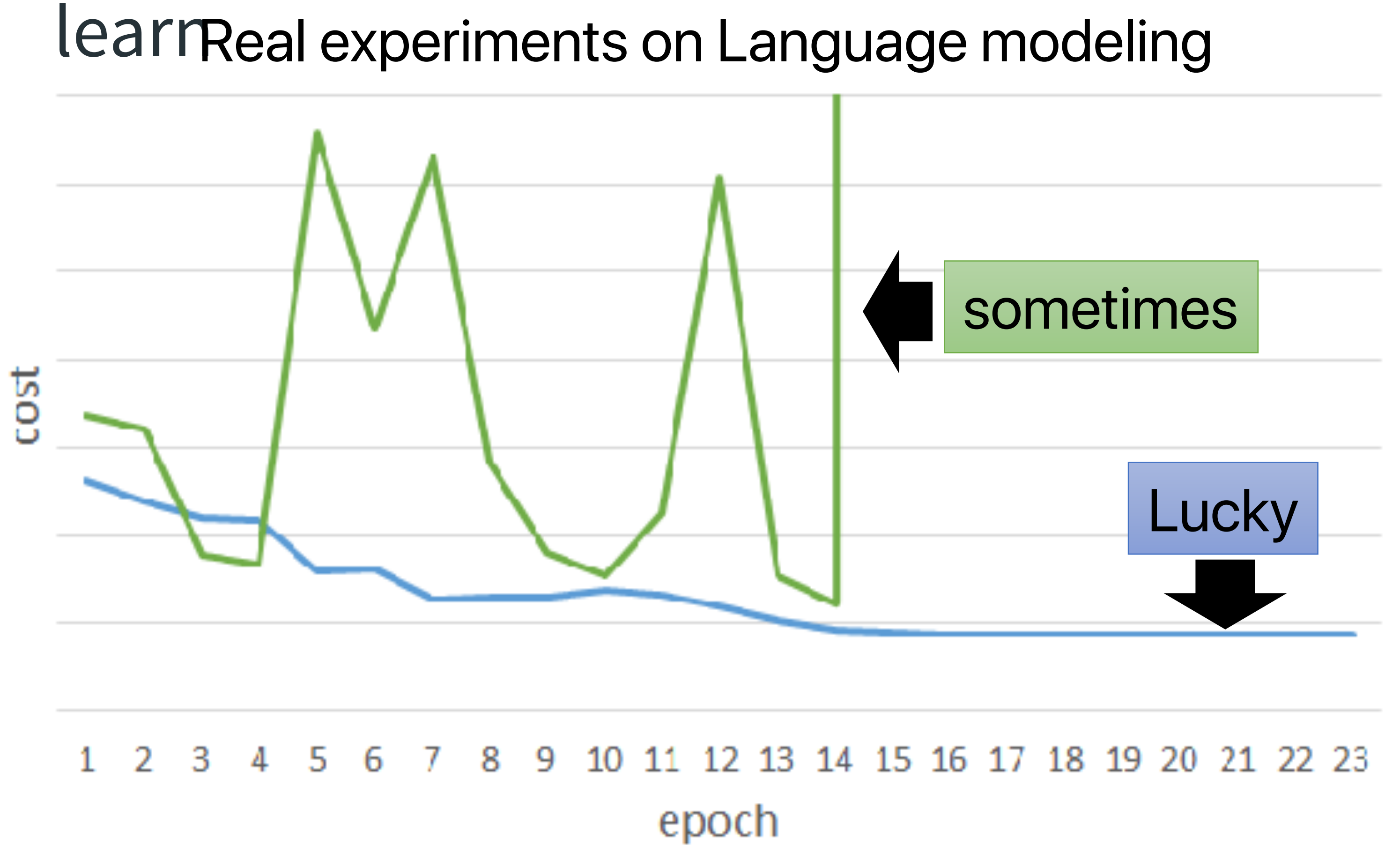
Add a symbol "===" (斷)

[Ilya Sutskever, NIPS'14][Dzmitry Bahdanau, arXiv'15]

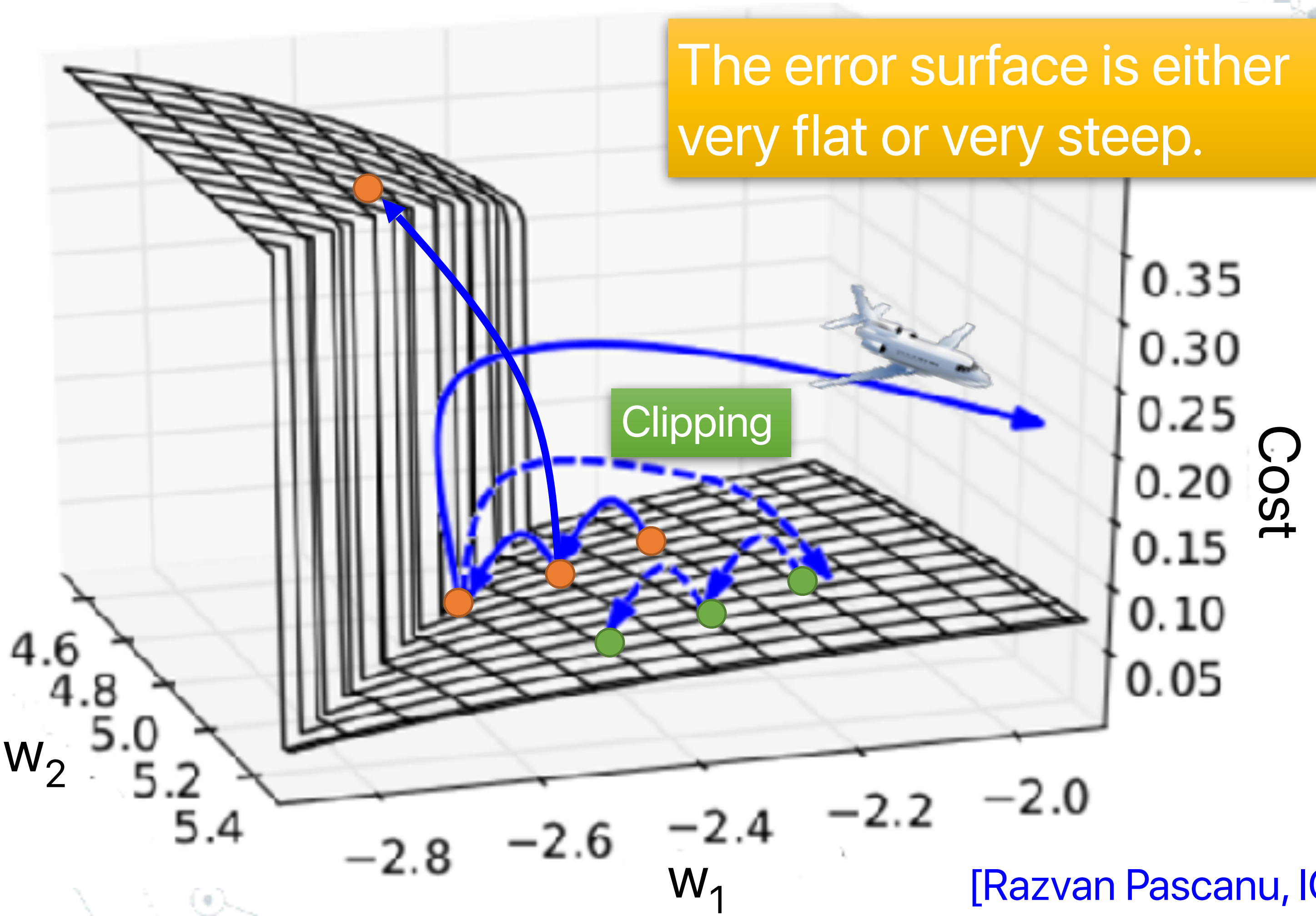
感謝 曾柏翔 同學提供實驗結果

Unfortunately ……

◎ RNN-based network is not always easy to learn



The error surface is rough.



[Razvan Pascanu, ICML'13]

Why?

$w = 1 \Rightarrow y^{1000} = 1$

$w = 1.01 \Rightarrow y^{1000} \approx 20000$

Large gradient → Small Learning rate?

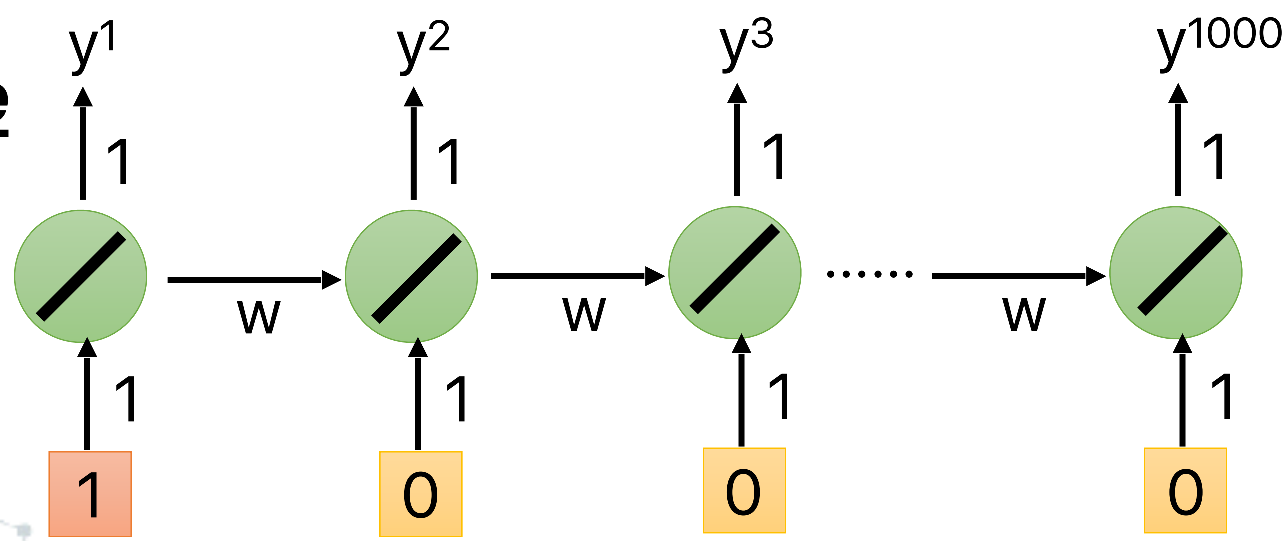
$w = 0.99 \Rightarrow y^{1000} \approx 0$

$w = 0.01 \Rightarrow y^{1000} \approx 0$

small gradient → Large Learning rate?

$=w^{999}$

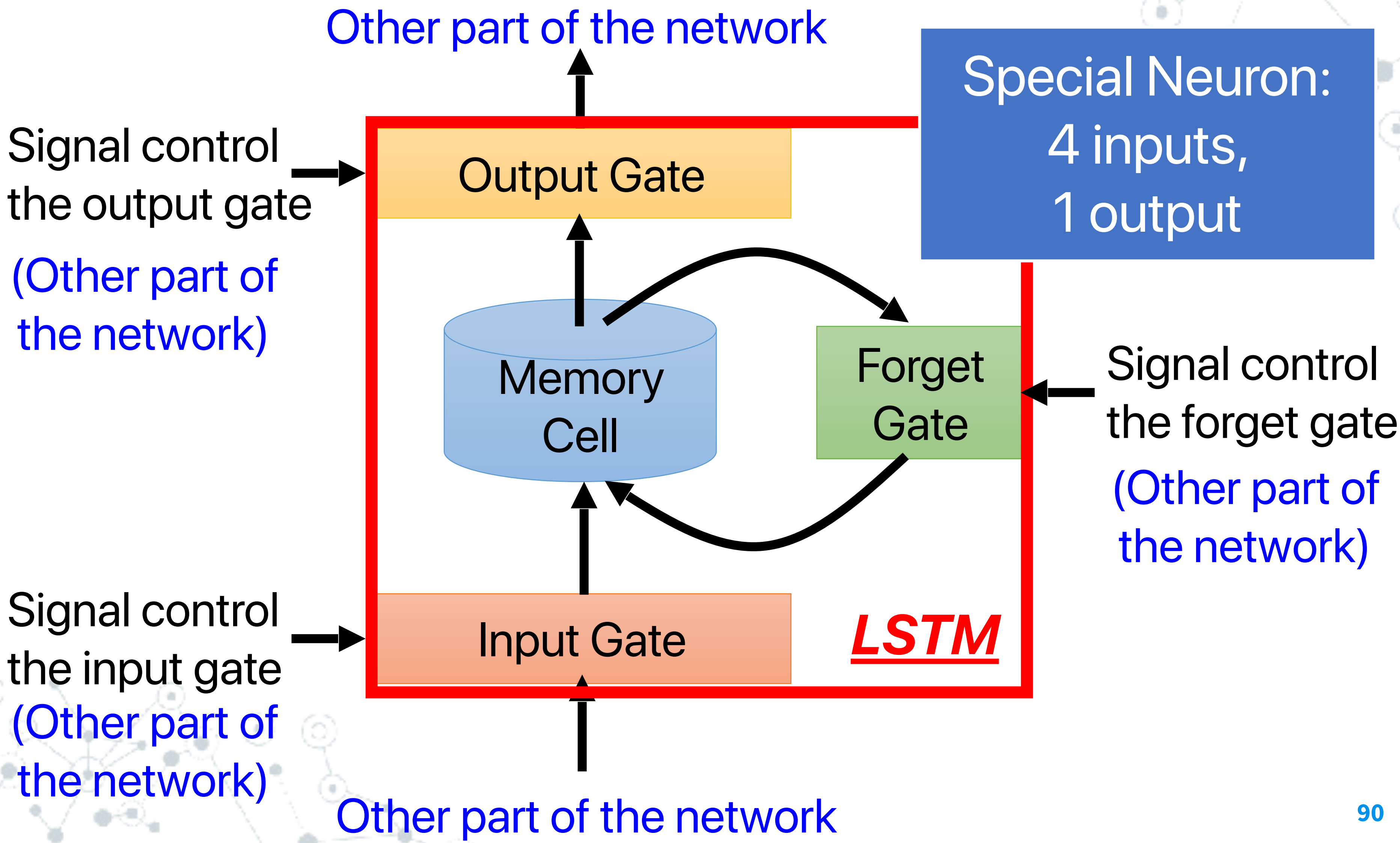
Toy Example

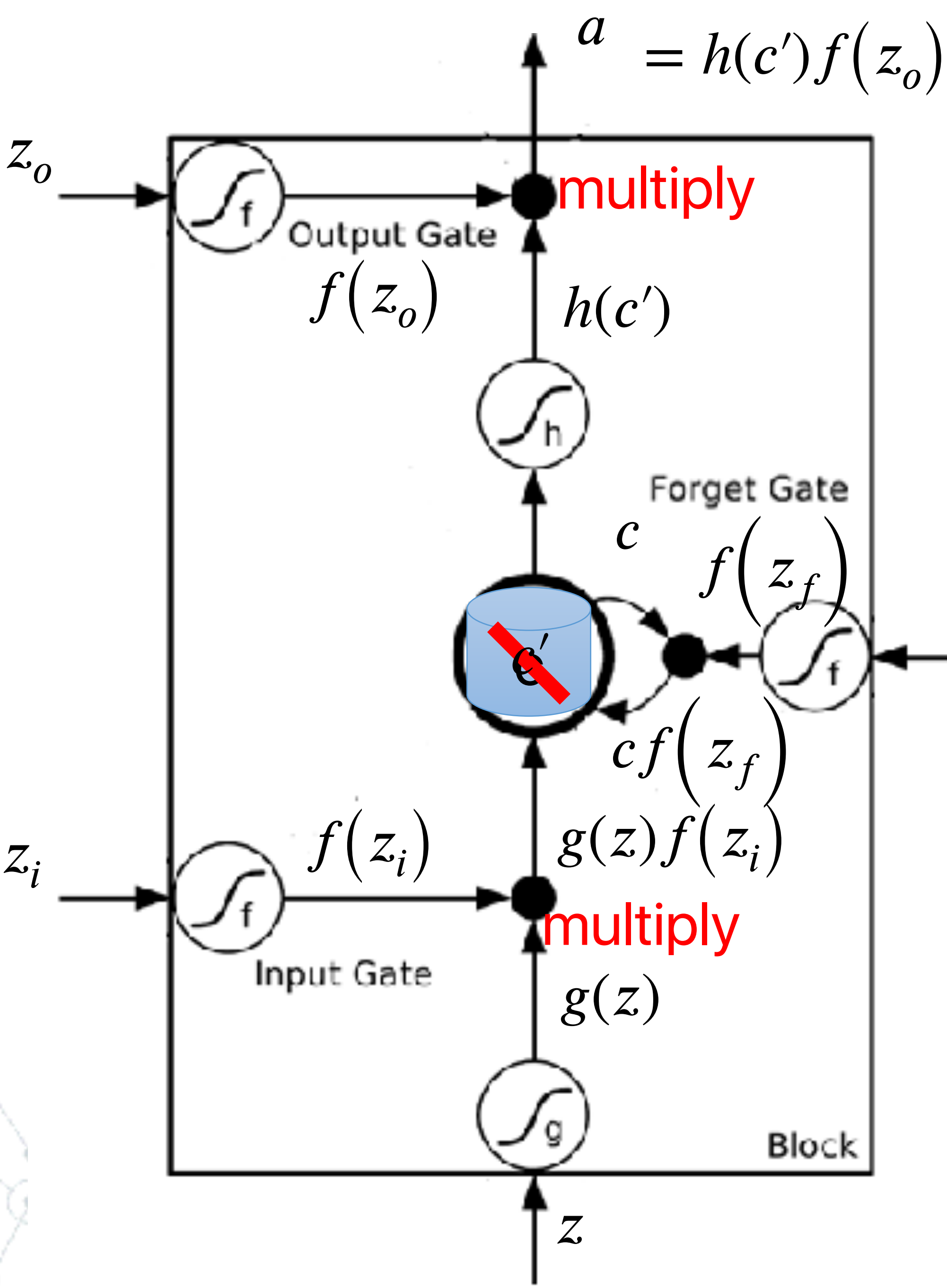


Helpful Techniques

- Nesterov's Accelerated Gradient (NAG):
  - Advance momentum method
- RMS Prop
  - Advanced approach to give each parameter different learning rates
  - Considering the change of Second derivatives
- Long Short-term Memory (LSTM)
  - Can deal with gradient vanishing (not gradient explode)

Long Short-term Memory (LSTM)





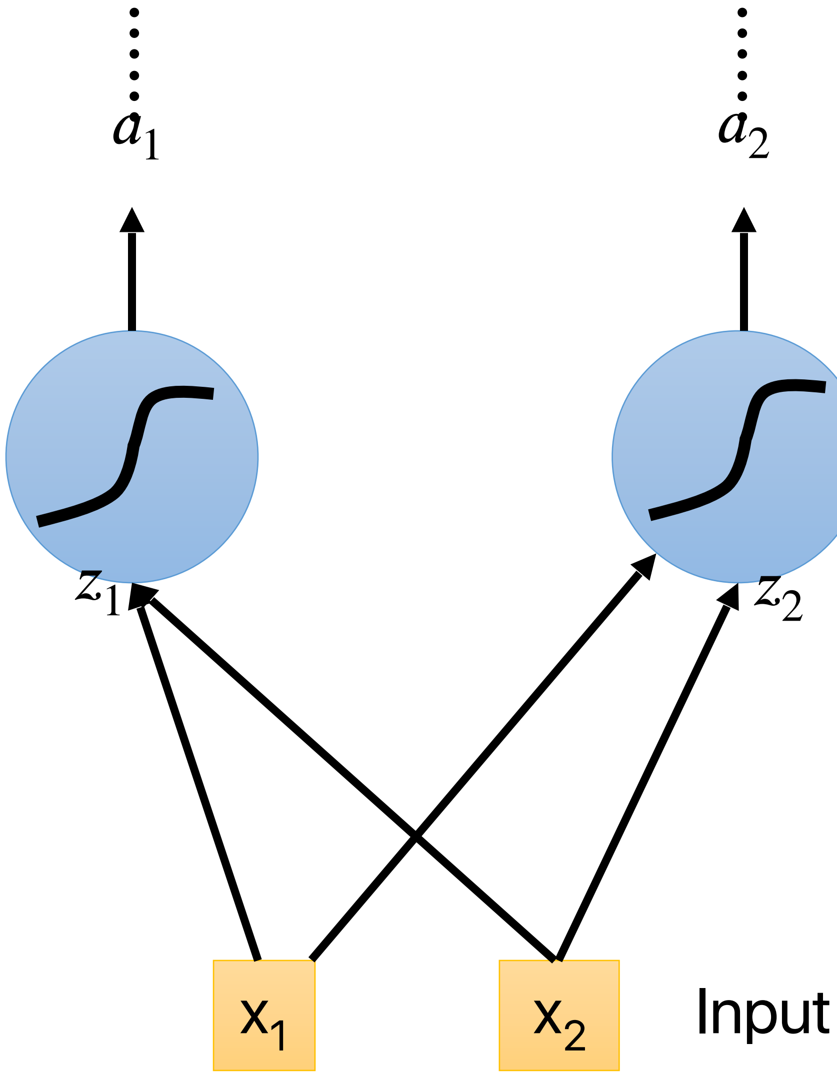
Activation function f is usually a sigmoid function

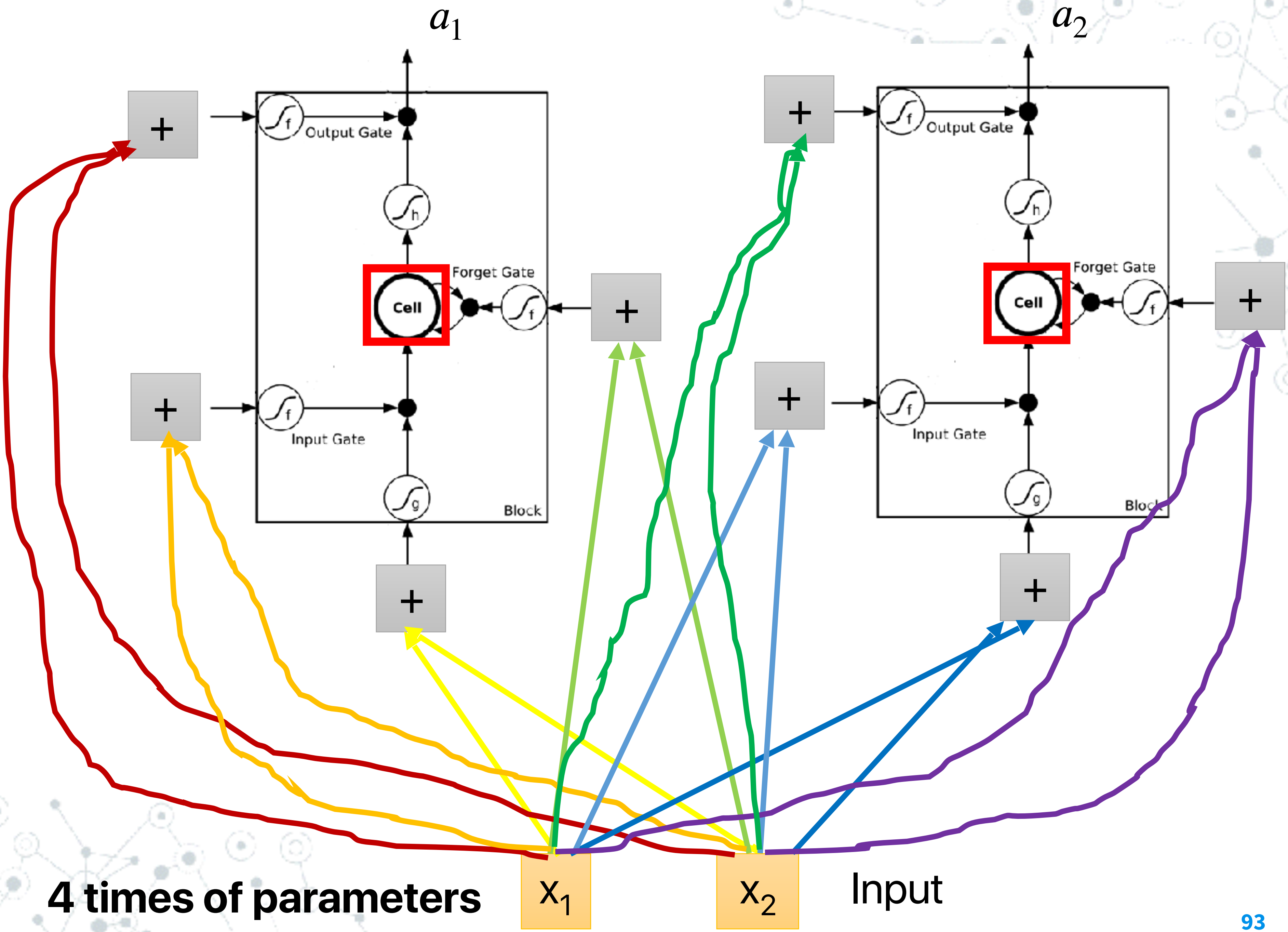
- Between 0 and 1
- Mimic open and close gate

$$c' = g(z)f(z_i) + cf(z_f)$$

Original Network:

➢Simply replace the neurons with LSTM

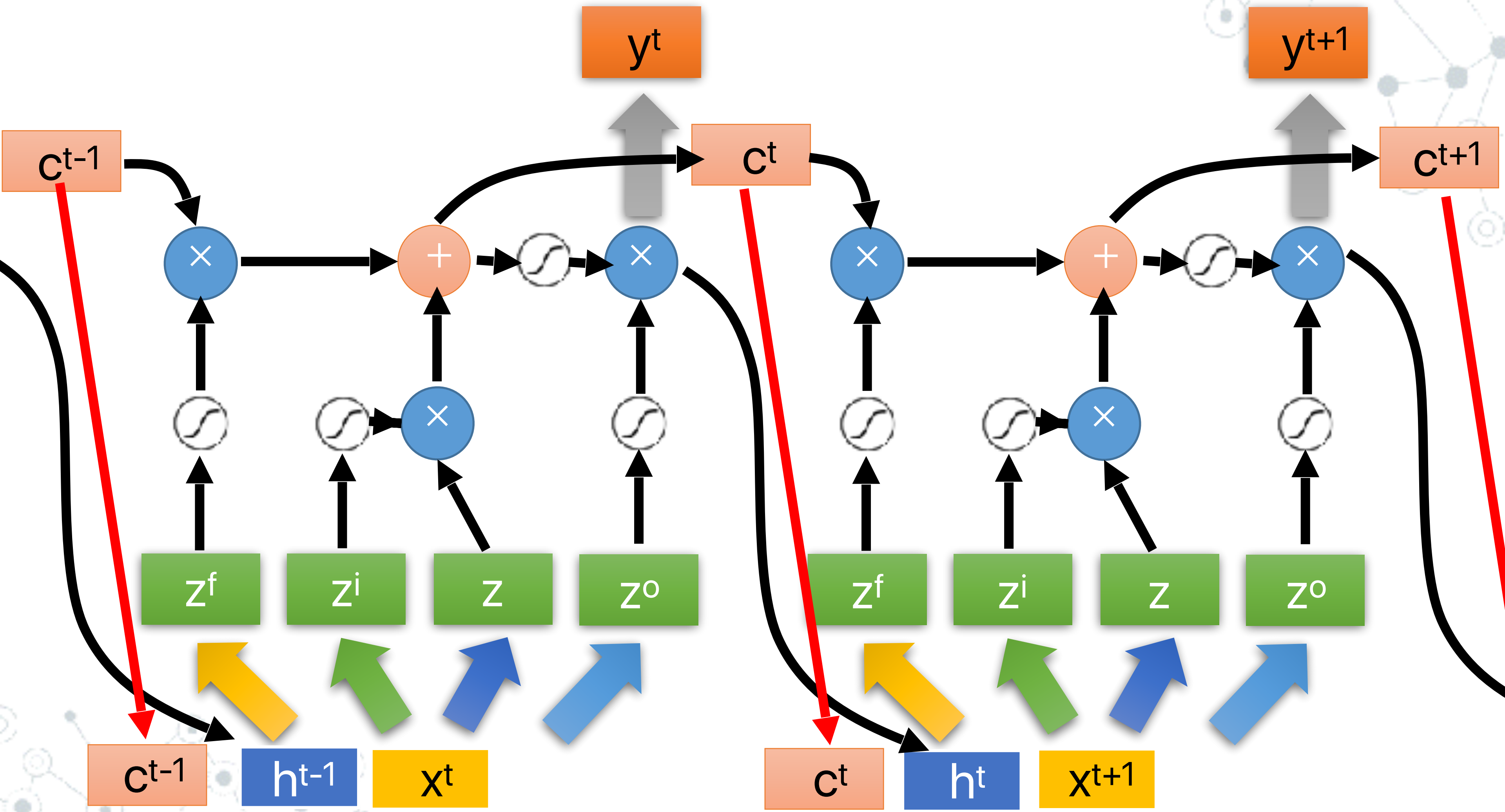




4 times of parameters

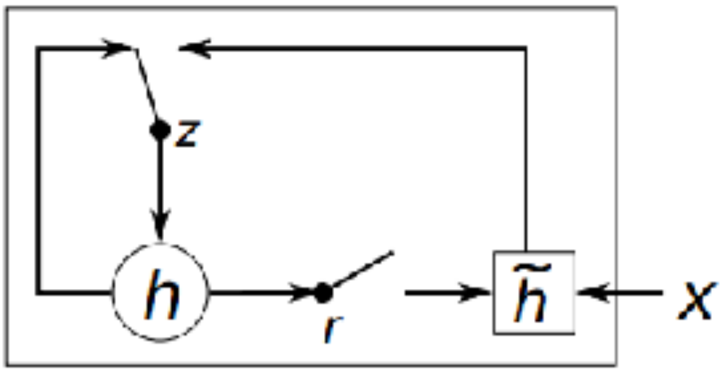
LSTM

Extension: "peephole"



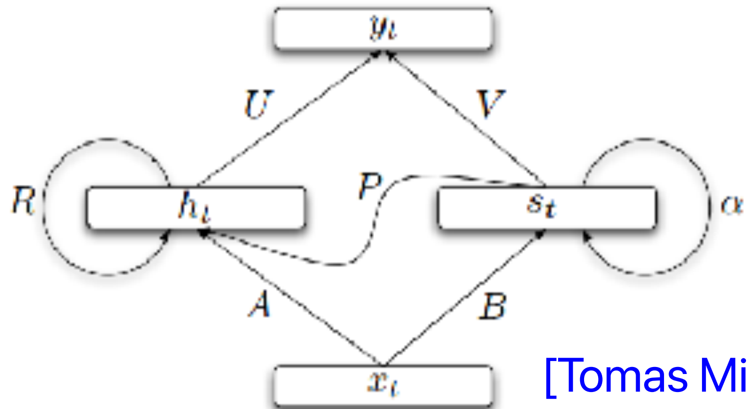
Other Simpler Alternatives

Gated Recurrent Unit (GRU)



[Cho, EMNLP'14]

Structurally Constrained Recurrent Network (SCRN)



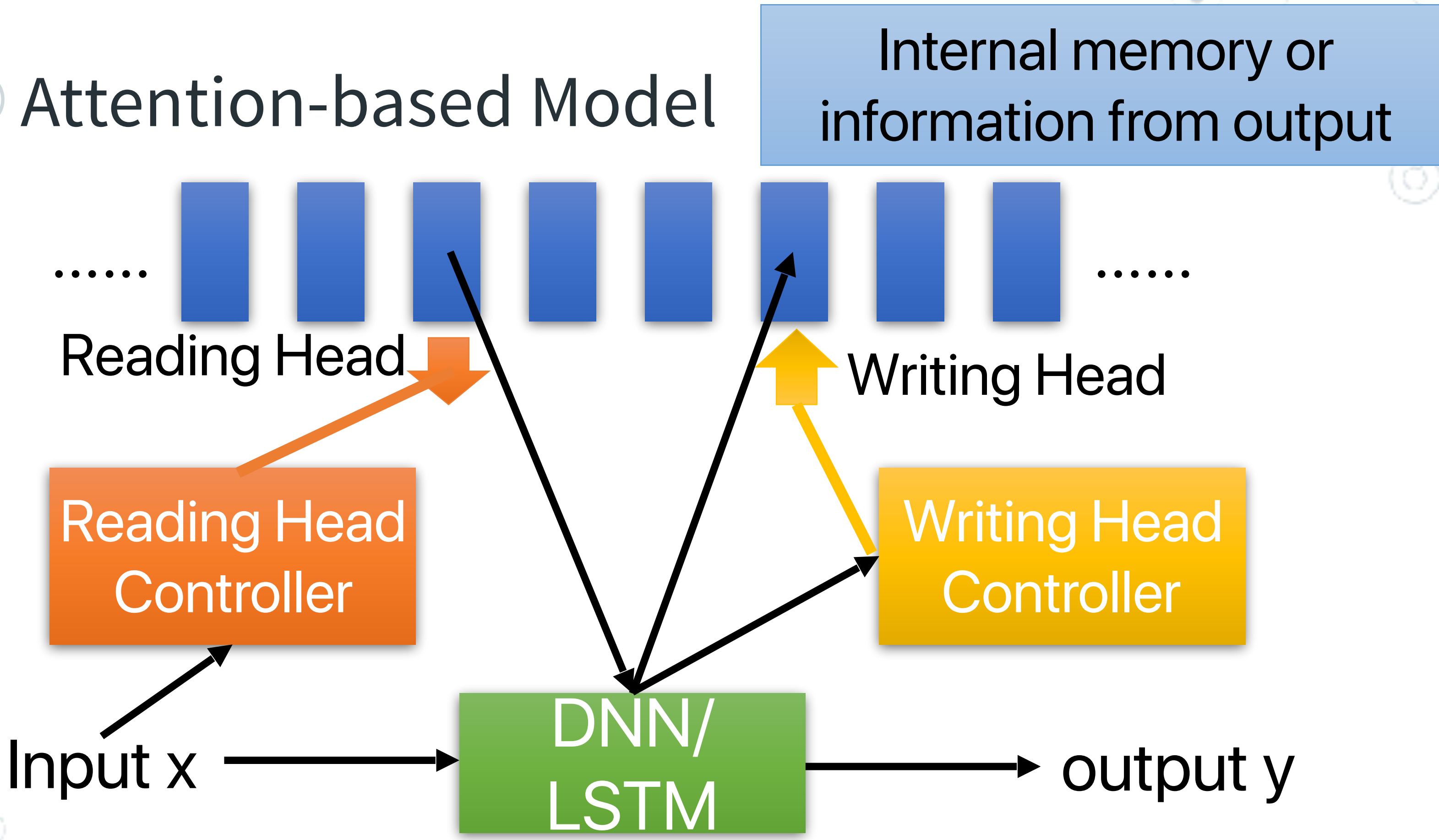
[Tomas Mikolov, ICLR'15]

Vanilla RNN Initialized with Identity matrix + ReLU activation function [Quoc V. Le, arXiv'15]

➢ Outperform or be comparable with LSTM in 4 different tasks

What is the next wave?

◎ Attention-based Model



Already applied on speech recognition, caption generation, QA, visual QA

What is the next wave?

- Attention-based Model
- End-To-End Memory Networks. S. Sukhbaatar, A. Szlam, J. Weston, R. Fergus. arXiv Pre-Print, 2015.
- Neural Turing Machines. Alex Graves, Greg Wayne, Ivo Danihelka. arXiv Pre-Print, 2014
- Ask Me Anything: Dynamic Memory Networks for Natural Language Processing. Kumar et al. arXiv Pre-Print, 2015
- Neural Machine Translation by Jointly Learning to Align and Translate. D. Bahdanau, K. Cho, Y. Bengio; International Conference on Representation Learning 2015.
- Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. Kelvin Xu et. al.. arXiv Pre-Print, 2015.
- Attention-Based Models for Speech Recognition. Jan Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, Yoshua Bengio. arXiv Pre-Print, 2015.
- Recurrent models of visual attention. V. Mnih, N. Hees, A. Graves and K. Kavukcuoglu. In NIPS, 2014.
- A Neural Attention Model for Abstractive Sentence Summarization. A. M. Rush, S. Chopra and J. Weston. EMNLP 2015.

Concluding Remarks

Concluding Remarks

- Introduction of deep learning
- Discussing some reasons using deep learning
- New techniques for deep learning
  - ReLU, Maxout
  - Giving all the parameters different learning rates
  - Dropout
- Network with memory
  - Recurrent neural network
  - Long short-term memory (LSTM)

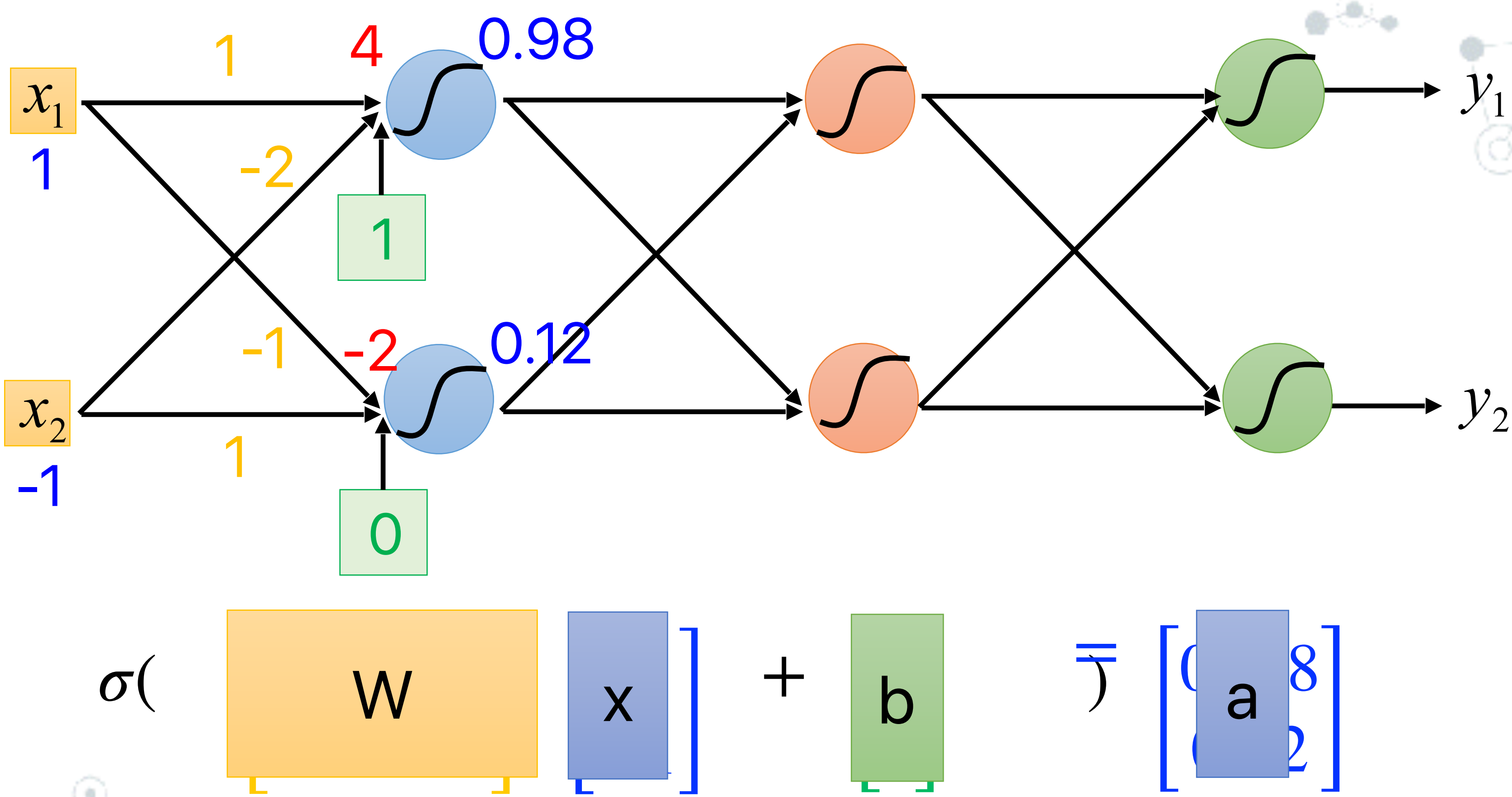
Reading Materials

- "Neural Networks and Deep Learning"
  - written by Michael Nielsen
  - http://neuralnetworksanddeeplearning.com/
- "Deep Learning" (not finished yet)
  - Written by Yoshua Bengio, Ian J. Goodfellow and Aaron Courville
  - http://www.iro.umontreal.ca/~bengioy/dlbook/

Thank you!

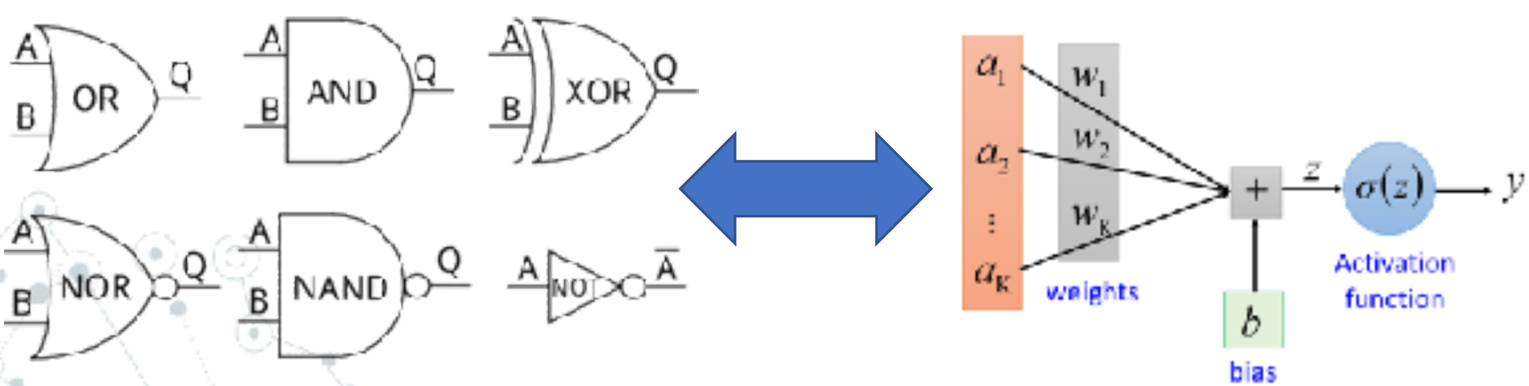
Appendix

Matrix Operation

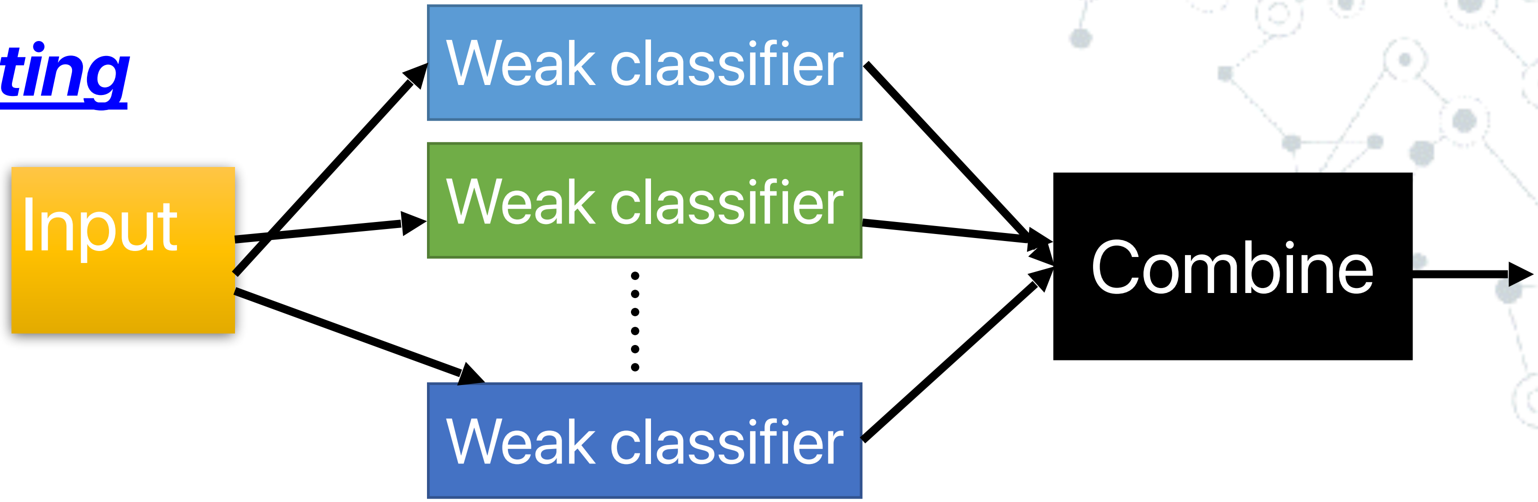


Why Deep? – Logic Circuits

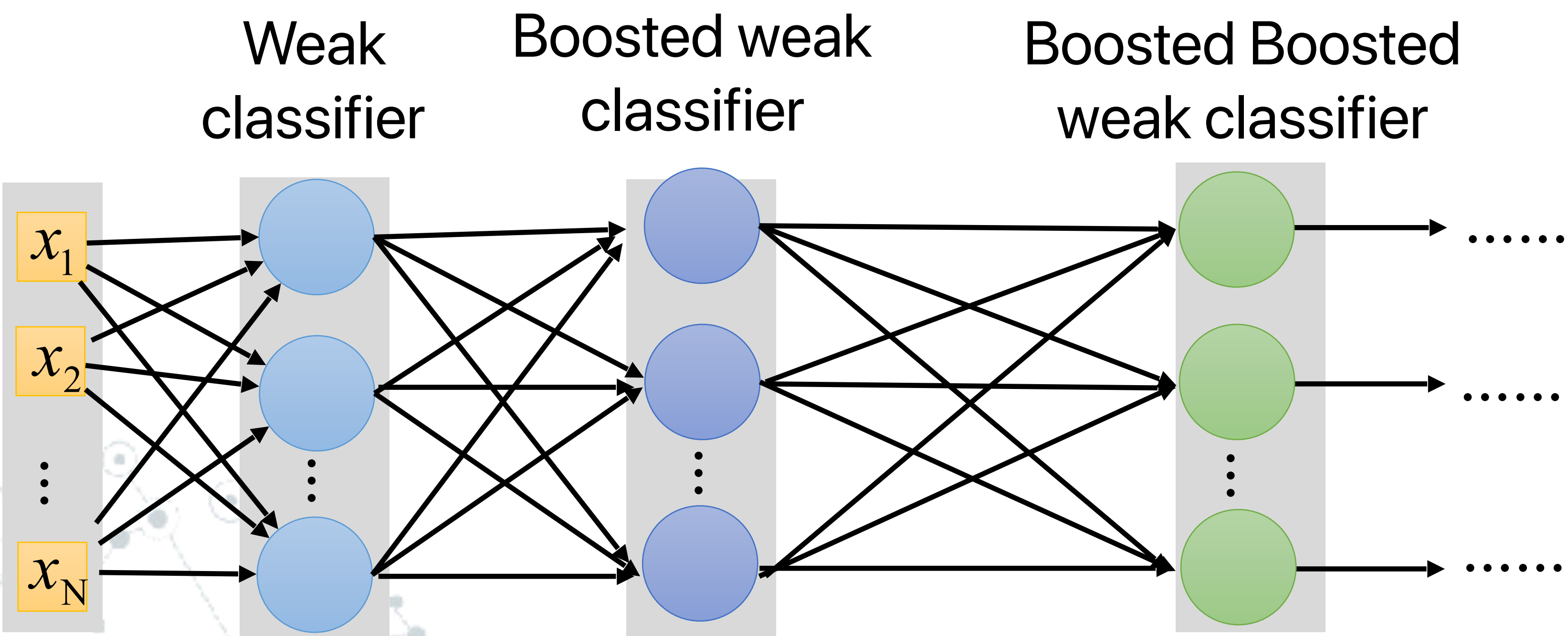
- A two levels of basic logic gates can represent any Boolean function.
- However, no one uses two levels of logic gates to build computers
- Using multiple layers of logic gates to build some functions are much simpler (less gates needed).



Boosting

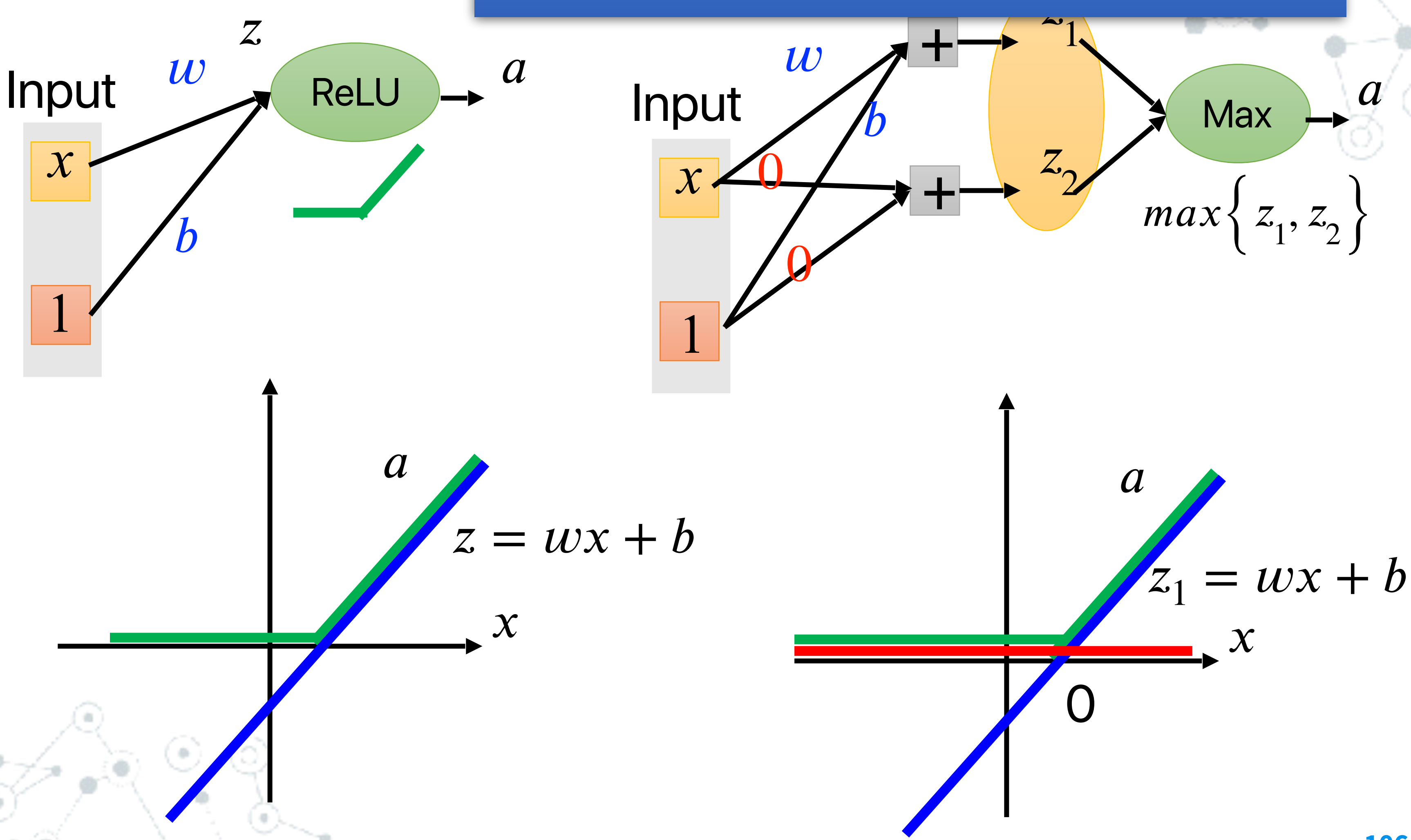


Deep Learning



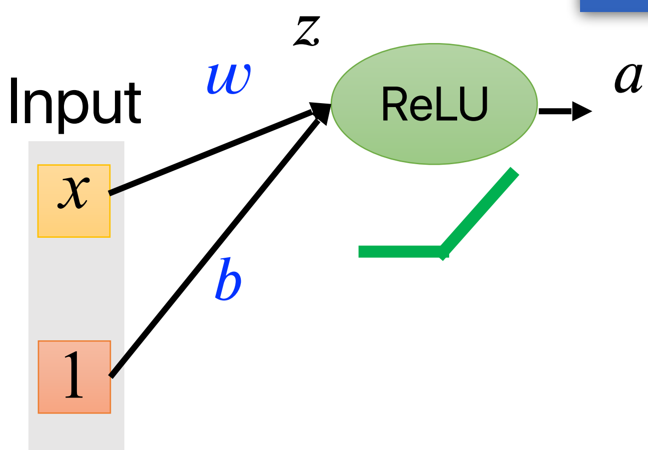
Maxout

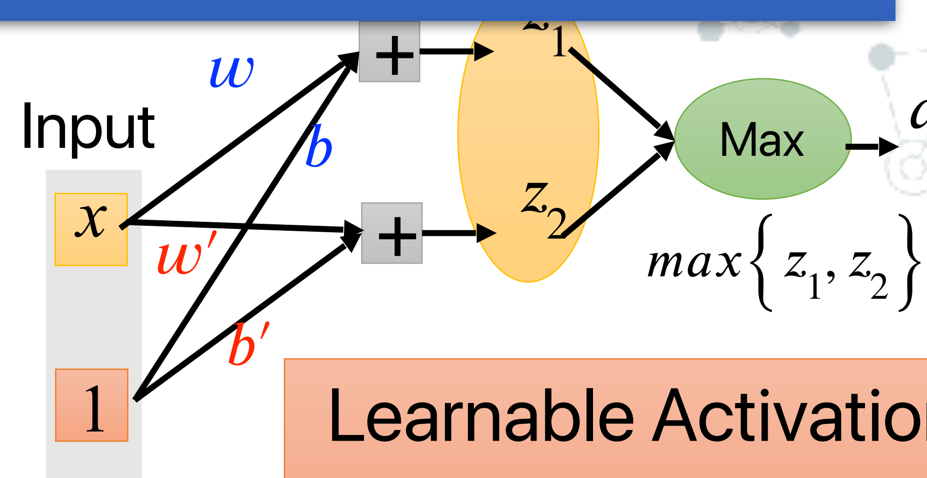
ReLU is a special cases of Maxout



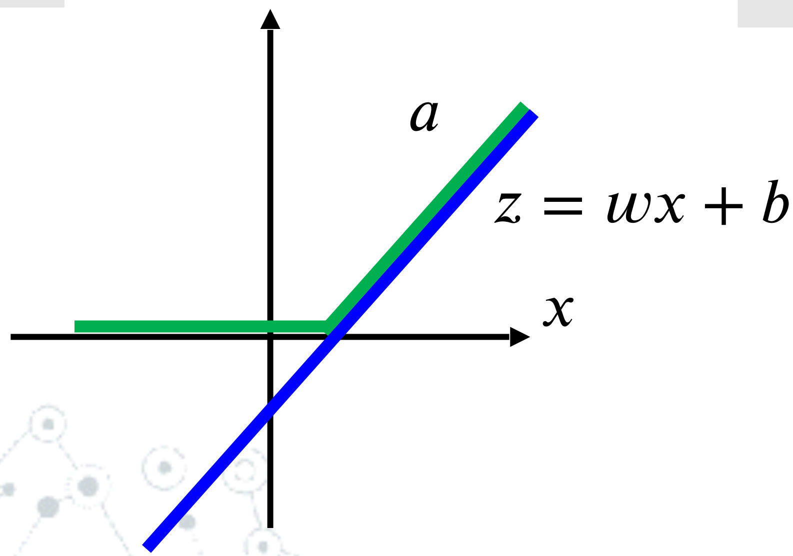
Maxout

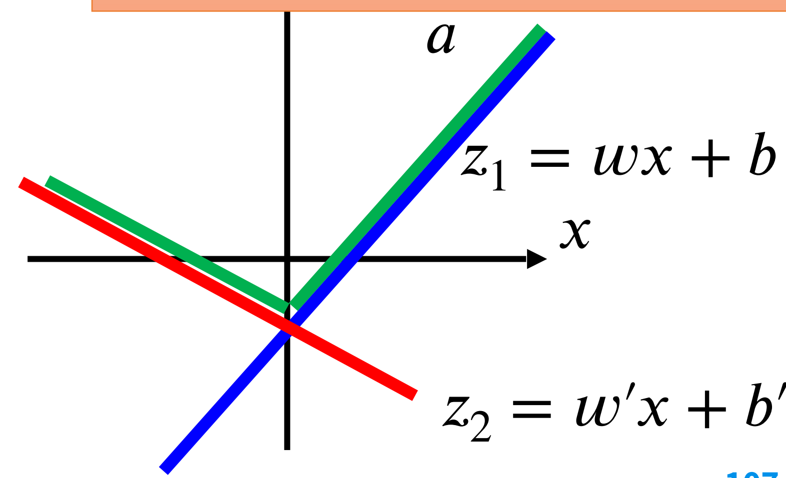
ReLU is a special cases of Maxout





Learnable Activation Function





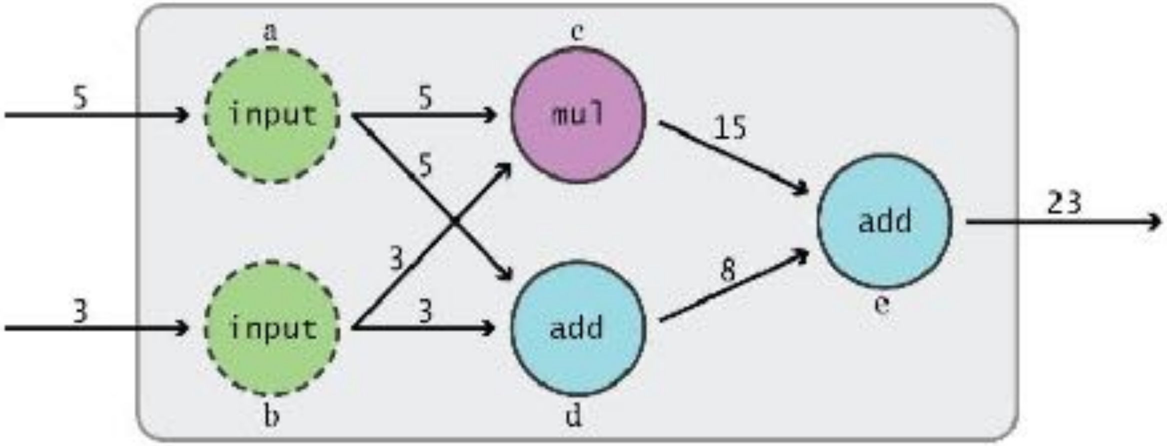
Getting Started

```
import
tensorflow as tf
```

Graphs and Sessions

Data Flow Graphs

TensorFlow separates definition of computations from their execution

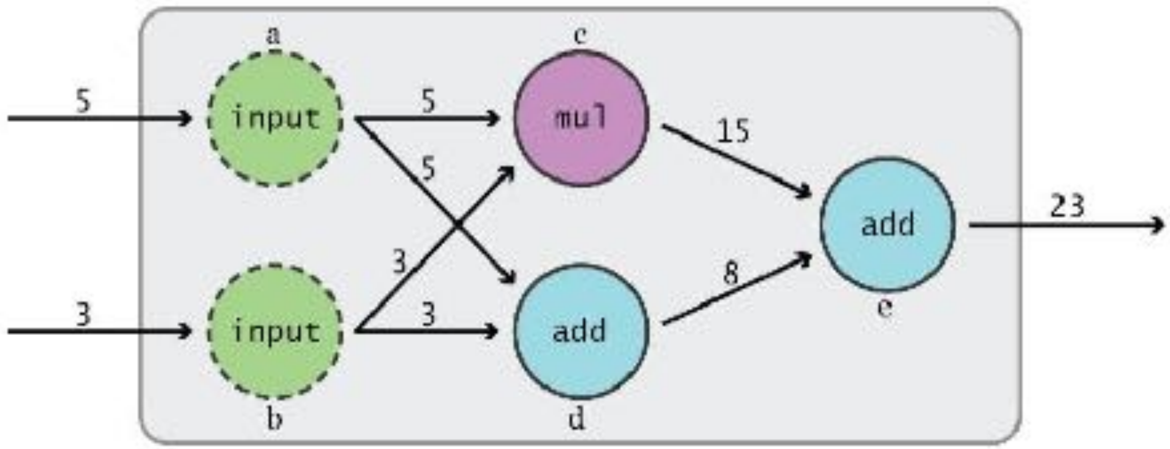


Graph from *TensorFlow for Machine Intelligence*

Data Flow Graphs

Phase 1: assemble a graph

Phase 2: use a session to execute operations in the graph



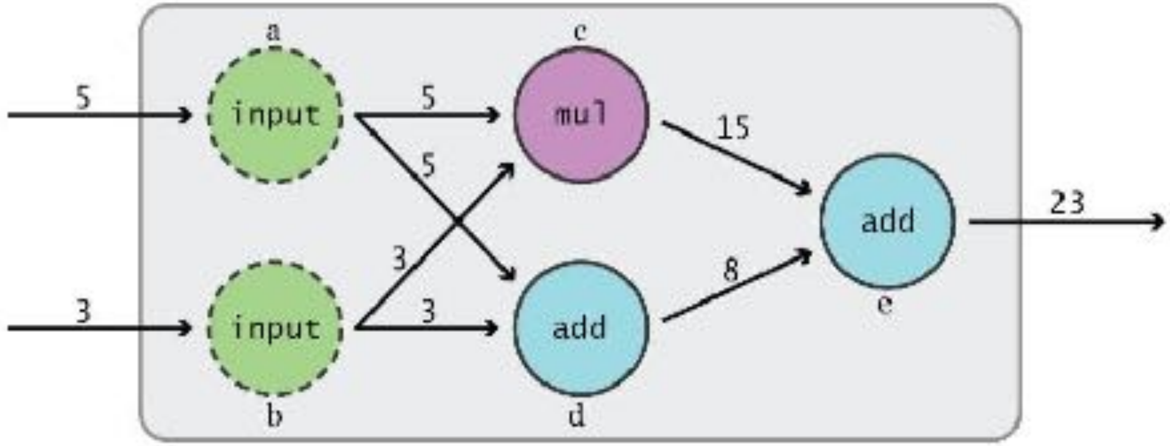
Graph from *TensorFlow for Machine Intelligence*

Data Flow Graphs

Phase 1: assemble a graph

This might change in the future with eager mode!!

Phase 2: use a session to execute operations in the graph



Graph from *TensorFlow for Machine Intelligence*

What’s a tensor?

What’s a tensor?

An n-dimensional array

0-d tensor: scalar (number)

1-d tensor: vector

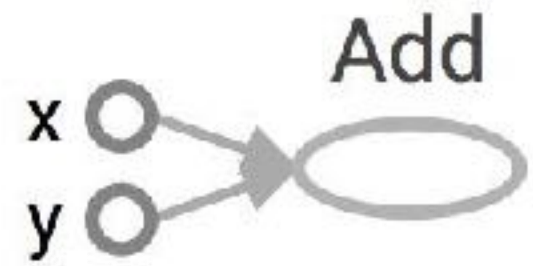
2-d tensor: matrix

and so on

Data Flow Graphs

```
import tensorflow as tf
a = tf.add(3, 5)
```

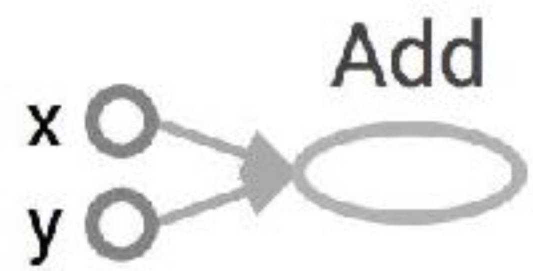
Visualized by TensorBoard



Data Flow Graphs

```
import tensorflow as tf
a = tf.add(3, 5)
```

Visualized by TensorBoard



Why x, y?

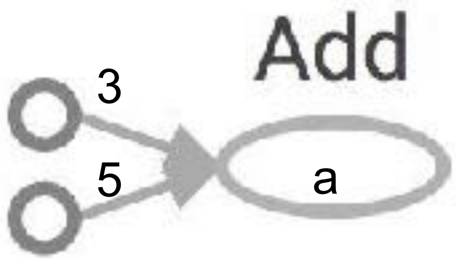
TF automatically names the nodes when you don't explicitly name them.
x = 3
y = 5

Data Flow Graphs

```
import tensorflow as tf
a = tf.add(3, 5)
```

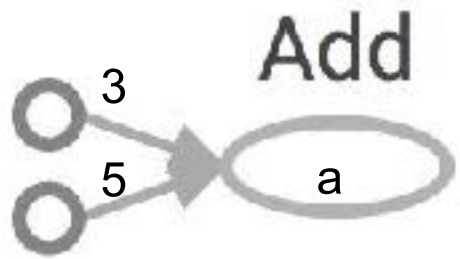
Interpreted?

Nodes: operators, variables, and constants
Edges: tensors



Data Flow Graphs

```
import tensorflow as tf
a = tf.add(3, 5)
```
Interpreted?

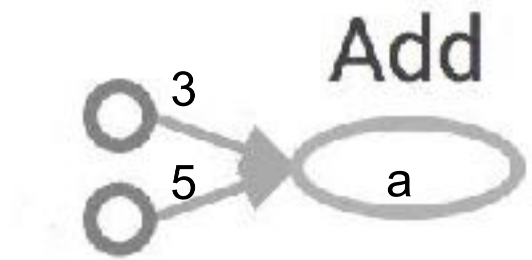


Nodes: operators, variables, and constants
Edges: tensors

Tensors are data.
TensorFlow = tensor + flow = data + flow

Data Flow Graphs

```
import tensorflow as tf
a = tf.add(3, 5)
print(a)
```




```
>> Tensor("Add:0", shape=(),
dtype=int32)
```

(Not 8)

How to get the value of a?

Create a **session**, assign it to variable sess so we can call it later

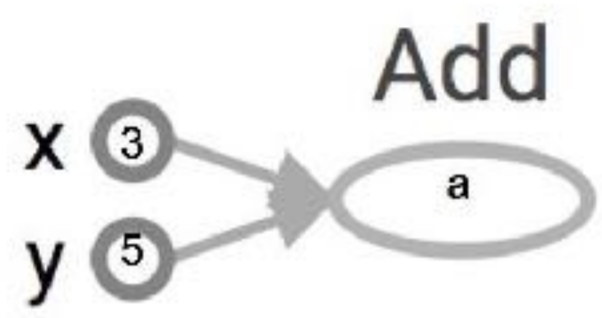
Within the session, evaluate the graph to fetch the value of a

How to get the value of a?

Create a **session**, assign it to variable sess so we can call it later

Within the session, evaluate the graph to fetch the value of a

```
import tensorflow as tf
a = tf.add(3, 5)
sess = tf.Session()
print(sess.run(a))
sess.close()
```

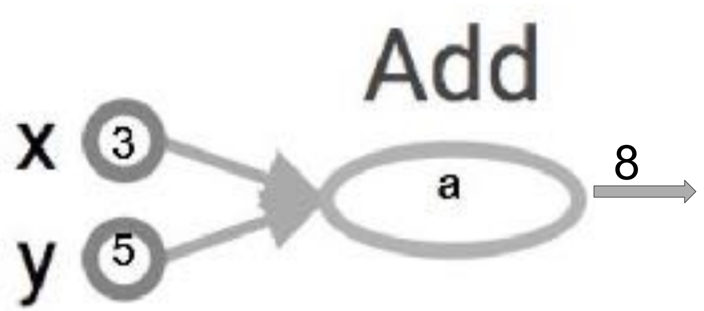


How to get the value of a?

Create a **session**, assign it to variable sess so we can call it later

Within the session, evaluate the graph to fetch the value of a

```
import tensorflow as tf
a = tf.add(3, 5)
sess = tf.Session()
print(sess.run(a))
sess.close()
```

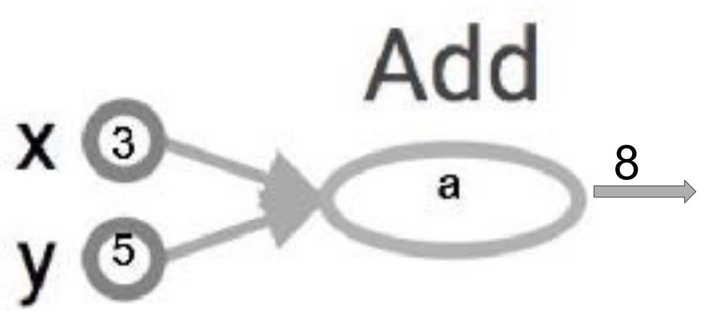


How to get the value of a?

Create a **session**, assign it to variable sess so we can call it later

Within the session, evaluate the graph to fetch the value of a

```
import tensorflow as tf
a = tf.add(3, 5)
sess = tf.Session()
with tf.Session() as sess:
        print(sess.run(a))
sess.close()
```



tf.Session()

A Session object encapsulates the environment in which Operation objects are executed, and Tensor objects are evaluated.

tf.Session()

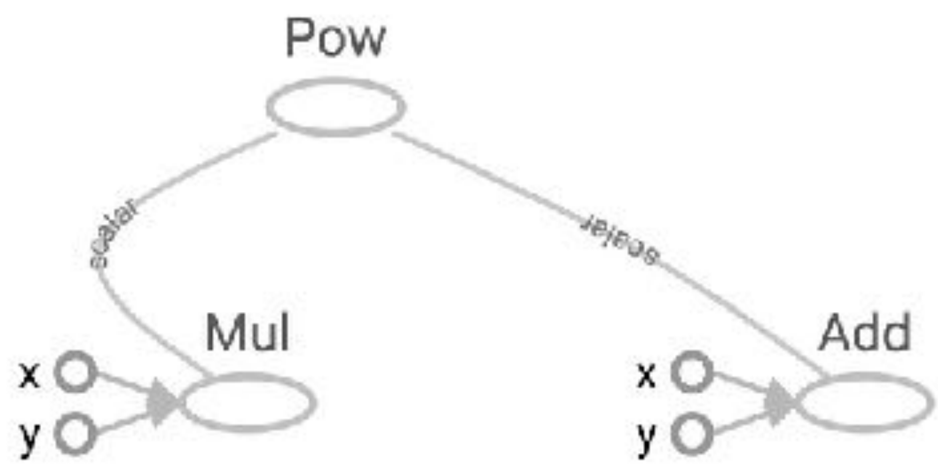
A Session object encapsulates the environment in which Operation objects are executed, and Tensor objects are evaluated.

Session will also allocate memory to store the current values of variables.

More graph

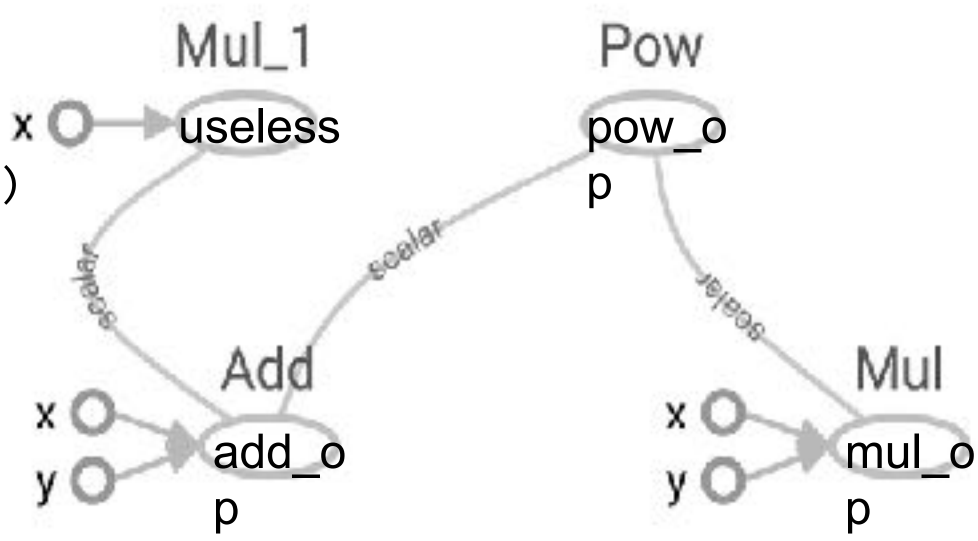
```
x = 2
y = 3
op1 = tf.add(x, y)
op2 = tf.multiply(x, y)
op3 = tf.pow(op2, op1)
with tf.Session() as sess
        op3 = sess.run(op
```

Visualized by TensorBoard



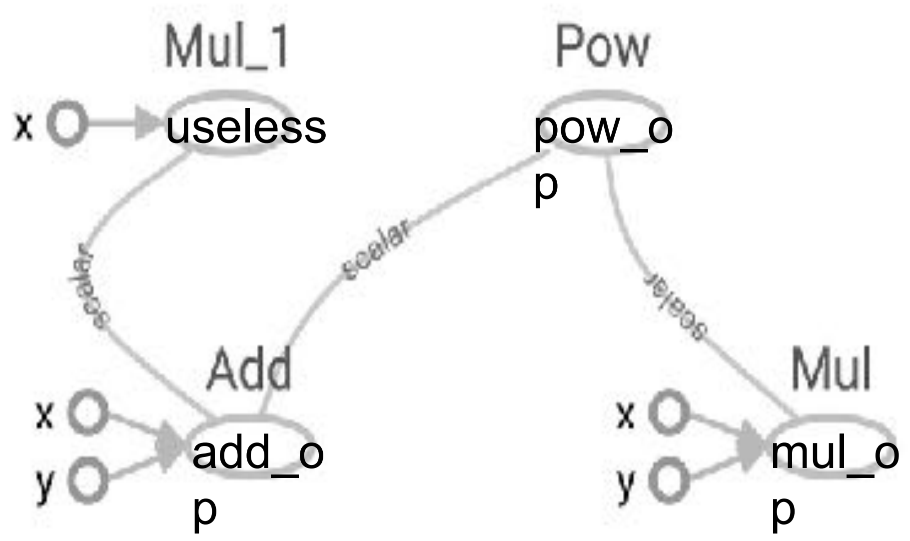
Subgraphs

```
x = 2
y = 3
add_op = tf.add(x, y)
mul_op = tf.multiply(x, y)
useless = tf.multiply(x, add_op)
pow_op = tf.pow(add_op, mul_op)
with tf.Session() as sess:
        z = sess.run(pow_op)
```



Subgraphs

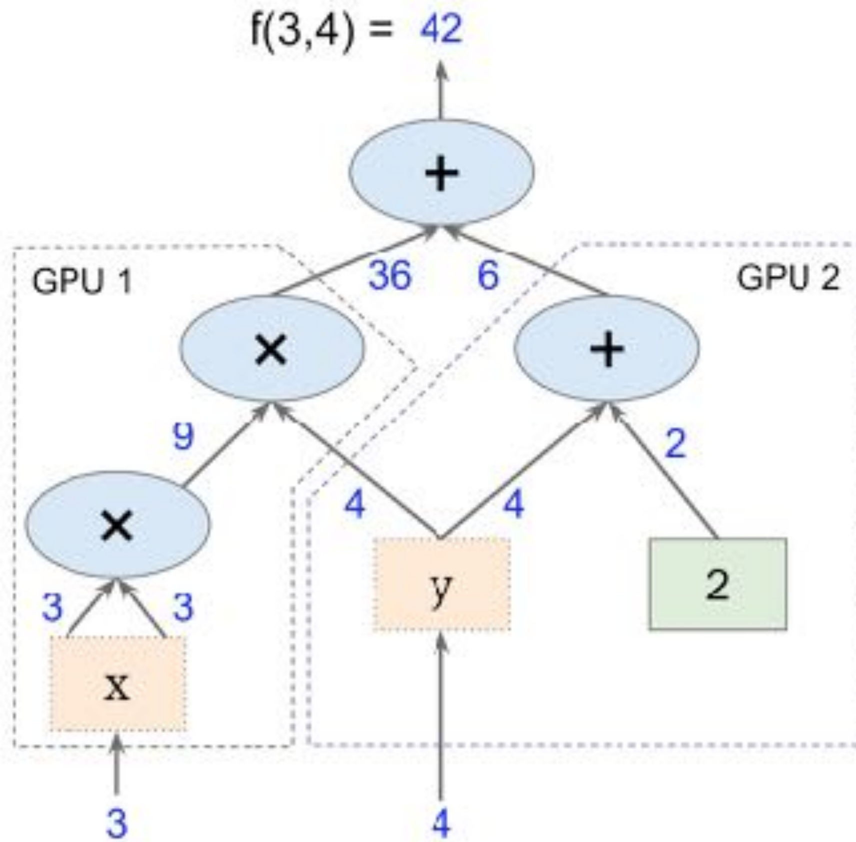
```
x = 2
y = 3
add_op = tf.add(x, y)
mul_op = tf.multiply(x, y)
useless = tf.multiply(x, add_op)
pow_op = tf.pow(add_op, mul_op)
with tf.Session() as sess:
        z, not_useless = sess.run
```



Subgraphs

Possible to break graphs into several chunks and run them parallelly across multiple CPUs, GPUs, TPUs, or other devices

Example: AlexNet



Graph from *Hands-On Machine Learning with Scikit-Learn and TensorFlow*

Distributed Computation

To put part of a graph on a specific CPU or GPU:

```
# Creates a graph.
with tf.device('/gpu:2'):
  a = tf.constant([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], name='a')
  b = tf.constant([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], name='b')
  c = tf.multiply(a, b)

# Creates a session with log_device_placement set to True.
sess =
tf.Session(config=tf.ConfigProto(log_device_placement=True))

# Runs the op.
print(sess.run(c))
```

tf.Graph()

create a graph:

```
g = tf.Graph()
```

tf.Graph()

to add operators to a graph, set it as default:

```
g = tf.Graph()
with g.as_default():
        x = tf.add(3, 5)
sess = tf.Session(graph=g)
with tf.Session() as sess:
        sess.run(x)
```

tf.Graph()

To handle the default graph:

```
g = tf.get_default_graph()
```

tf.Graph()

Do not mix default graph and user created graphs

```
g = tf.Graph()

# add ops to the default graph
a = tf.constant(3)

# add ops to the user created graph
with g.as_default():
        b = tf.constant(5)
```

tf.Graph()

Do not mix default graph and user created graphs

```
g1 = tf.get_default_graph()
g2 = tf.Graph()

# add ops to the default graph
with g1.as_default():
        a = tf.Constant(3)

# add ops to the user created graph
with g2.as_default():
        b = tf.Constant(5)
```

Install Tensorflow

Install Tensorflow

To install the library we will create an environment in Anaconda with **python 3.5** we name it **tensorflow**. However, you may choose your own desired name for it. Open command prompt (or terminal) and type:

```
conda create --name tensorflow python=3.5
```

Once the environment is created, we can activate the environment:

(for Windows):

```
activate tensorflow
```

(for Linux & Mac):

```
source activate tensorflow
```

(for Windows):

(CPU version):

```
pip install --upgrade tensorflow
```

(GPU version):

```
pip install --upgrade tensorflow-gpu
```

For Mac - CPU

```
pip install --ignore-installed --upgrade https://storage.googleapis.com/
tensorflow/mac/cpu/tensorflow-1.10.0-py3-none-any.whl
```

For Linux - CPU

```
pip install --ignore-installed --upgrade https://storage.googleapis.com/
tensorflow/linux/cpu/tensorflow-1.10.0-cp35-cp35m-linux_x86_64.whl
```

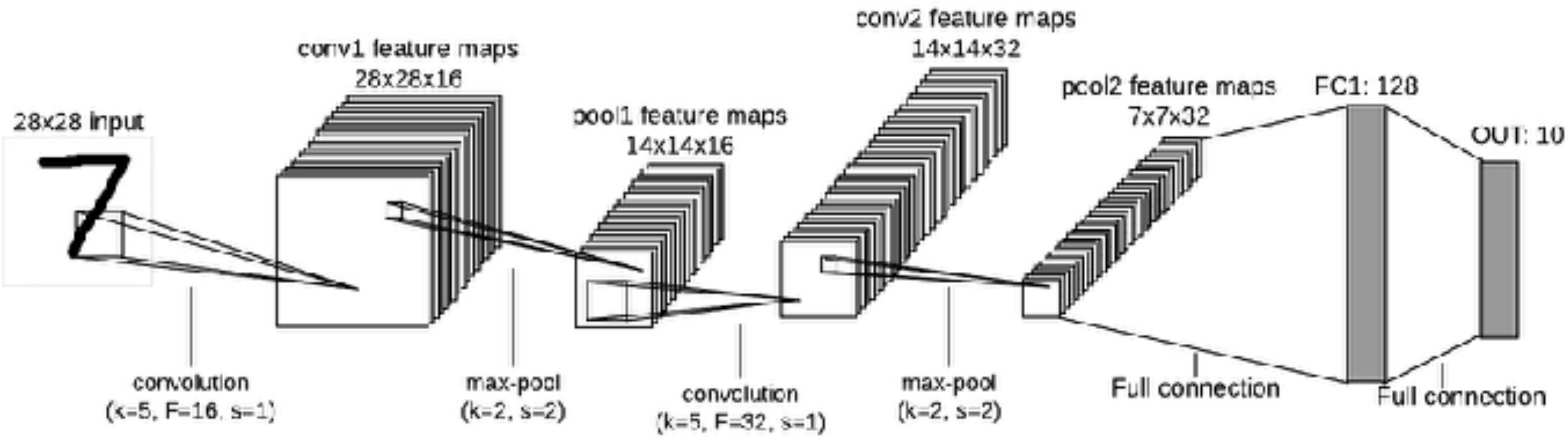
For Linux - GPU

```
pip install --ignore-installed --upgrade https://storage.googleapis.com/
tensorflow/linux/gpu/tensorflow_gpu-1.10.0-cp35-cp35m-linux_x86_64.whl
```

Install Jupiter

- conda install jupyter

CNN (Convolution Neural Networks)



See Jupiter Notebook

Thank you