BIG DATA and AI
for business

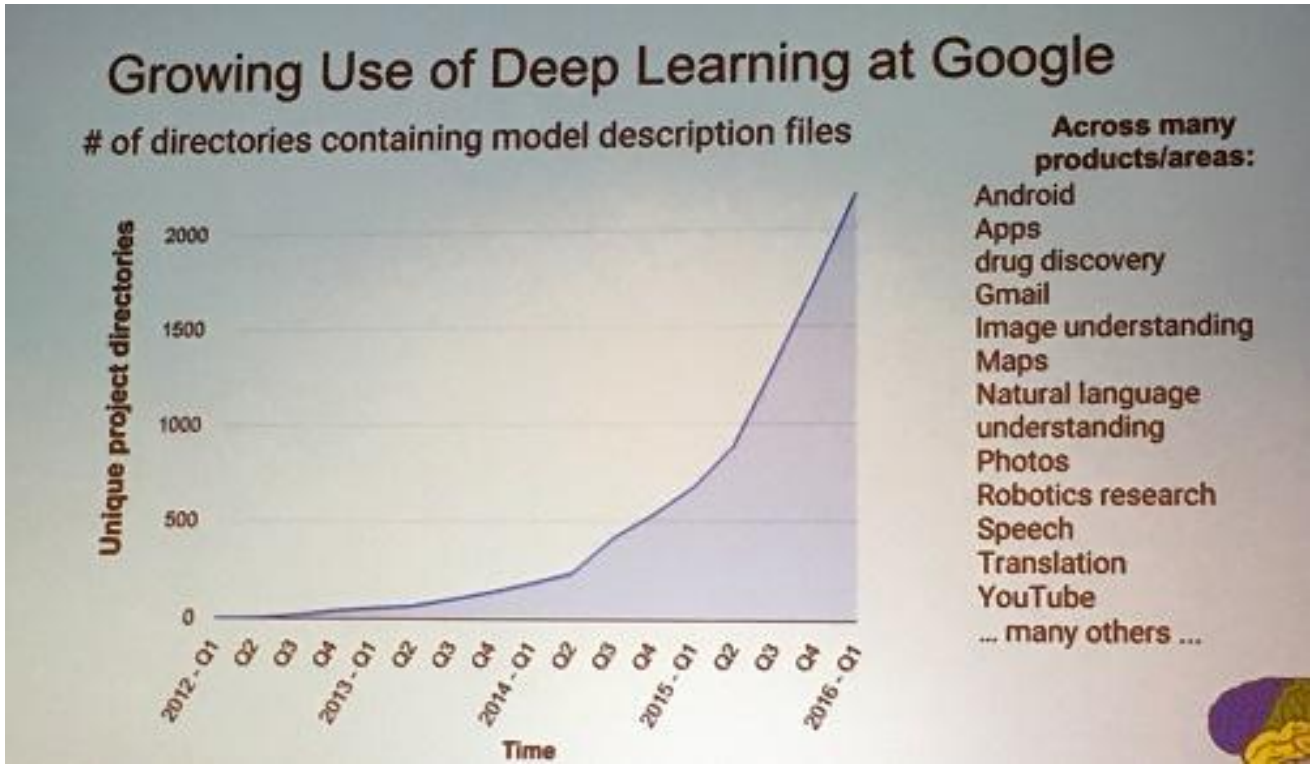
Lecture 3 (01/30, 02/04): Deep Learning (1)

Decisions, Operations & Information Technologies
Robert H. Smith School of Business
Spring, 2019

Credit to Hung-yi Lee

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

We mainly focus on the basic techniques.

Outline

Lecture I: Introduction of Deep Learning

Lecture II: Variants of Neural Network

Lecture III: Beyond Supervised Learning

Lecture I:
Introduction of Deep Learning

Outline

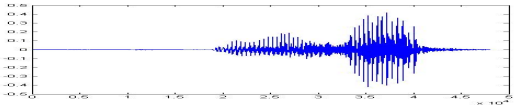
Introduction of Deep Learning

"Hello World" for Deep Learning

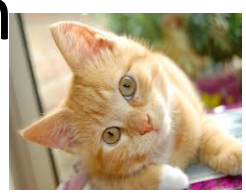
Tips for Deep Learning

Machine Learning ≈ Looking for a Function

- Speech Recognition

$f($  $)=$ “How are you”

- Image Recognition

$f($  $)=$ “Cat”

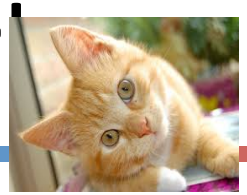
- Playing Go

$f($  $)=$ “5-5” (next move)

- Dialogue System

$f($ “Hi” (what the user said) $)=$ “Hello” (system response)

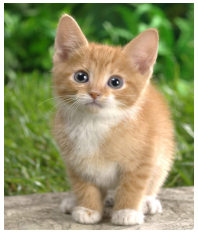
Image Recognition: Framework

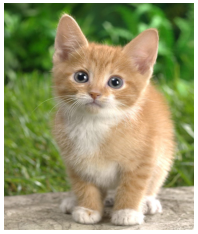
$f($  $) =$ "cat"

A set of function

Model

$f_1, f_2 \cdots$

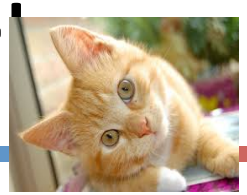
$f_1($  $) =$ "cat"

$f_2($  $) =$ "money"

$f_1($  $) =$ "dog"

$f_2($  $) =$ "snake"

Image Recognition: Framework

$f($  $) =$ "cat"

A set of function

Model

$f_1, f_2 \cdots$

Goodness of function f

Training Data

$f_1($  $) =$ "cat"

$f_2($  $) =$ "money"

Better!

$f_1($  $) =$ "dog"

$f_2($  $) =$ "snake"

Supervised Learning

function input:   

function output: "monkey" "cat" "dog"

Framework

Image Recognition:

$f($  $) =$ "cat"

Training

Testing

Step 1

A set of function

Model

$f_1, f_2 \cdots$

Step 2

Goodness of function f

Step 3

Pick the "Best" Function

f^*

Training Data

"monkey"

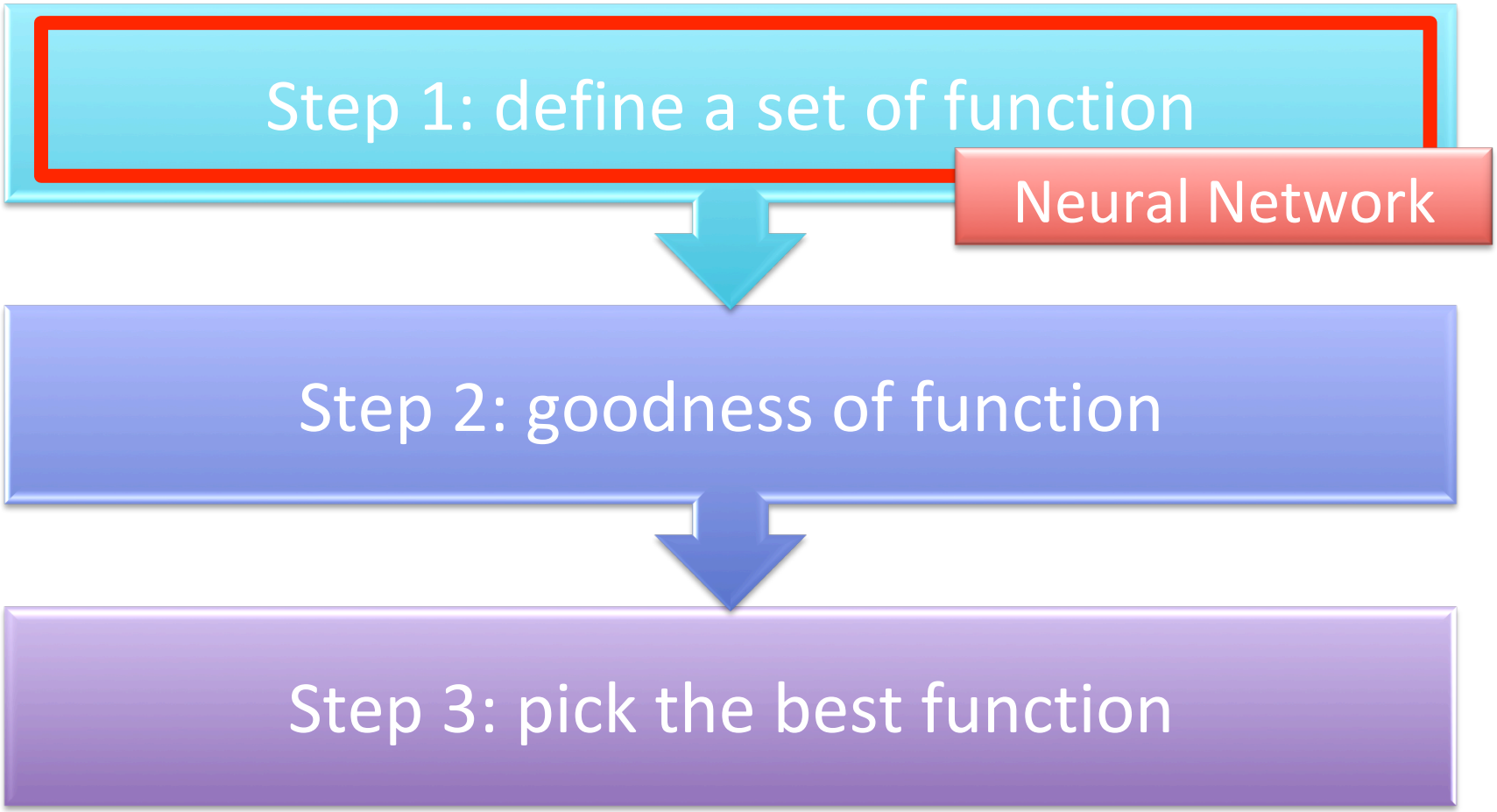
"cat"

"dog"

"cat"

Using f^*

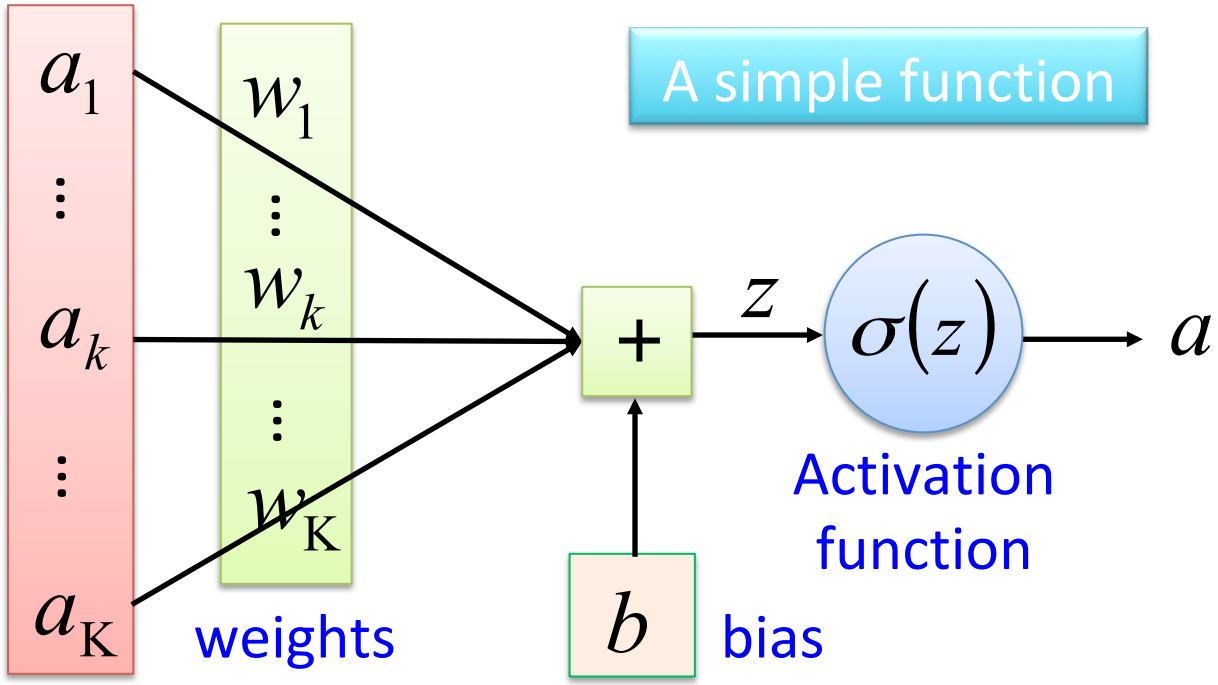
Three Steps for Deep Learning



Neural Network

Neuron

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

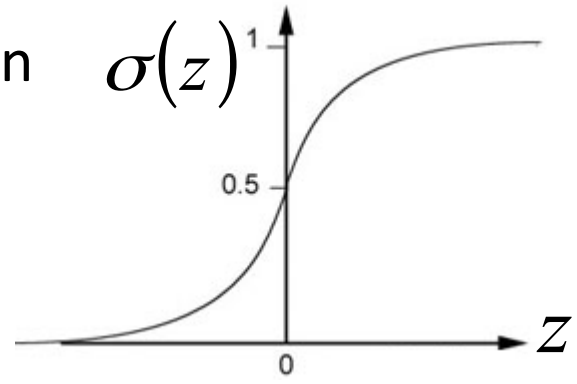


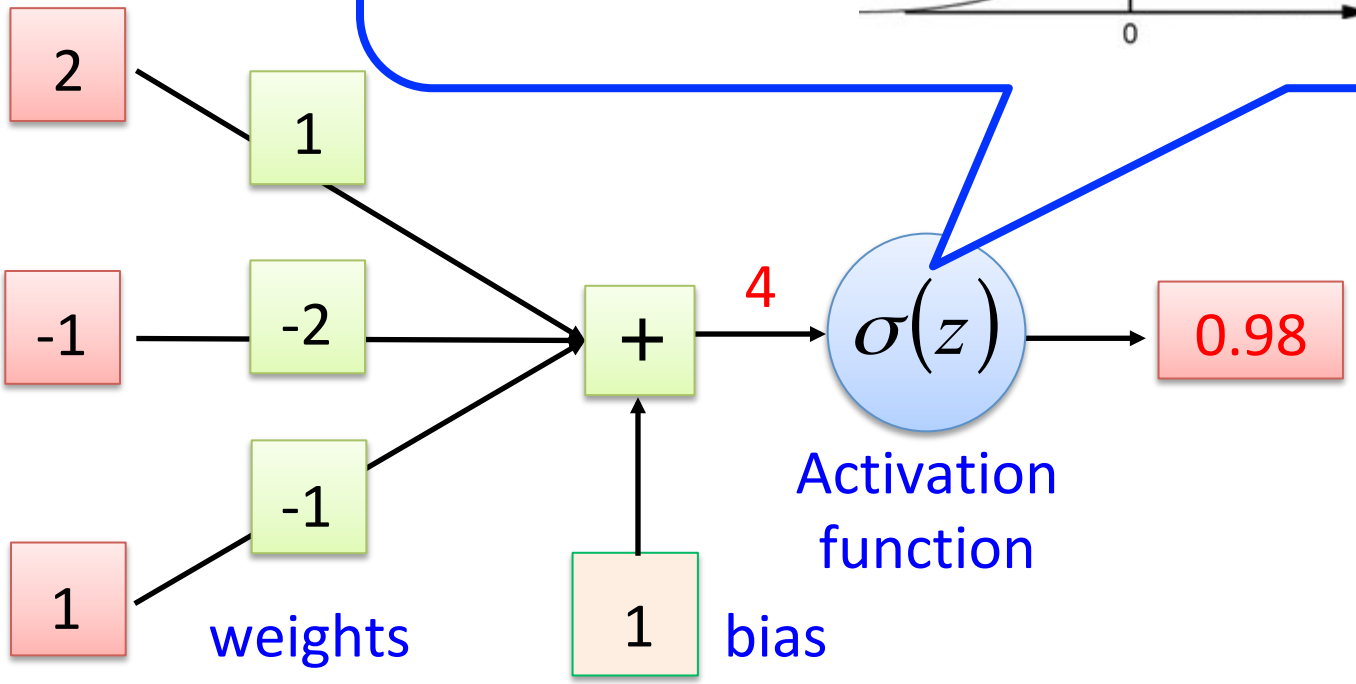
Neural Network

Neuron

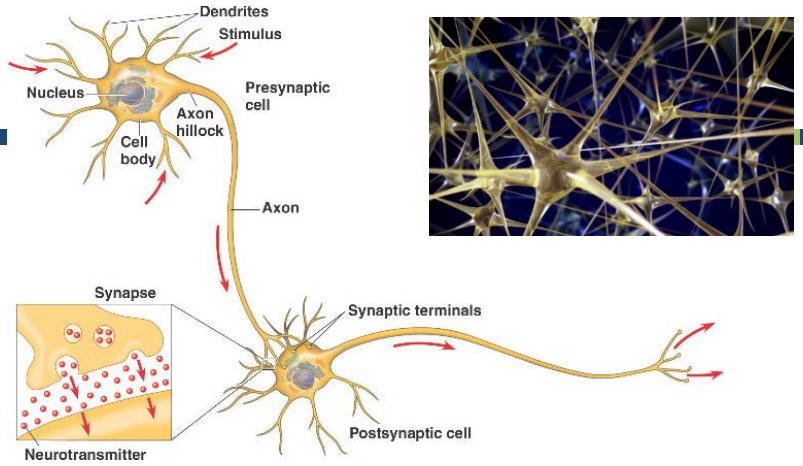
Sigmoid Function

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

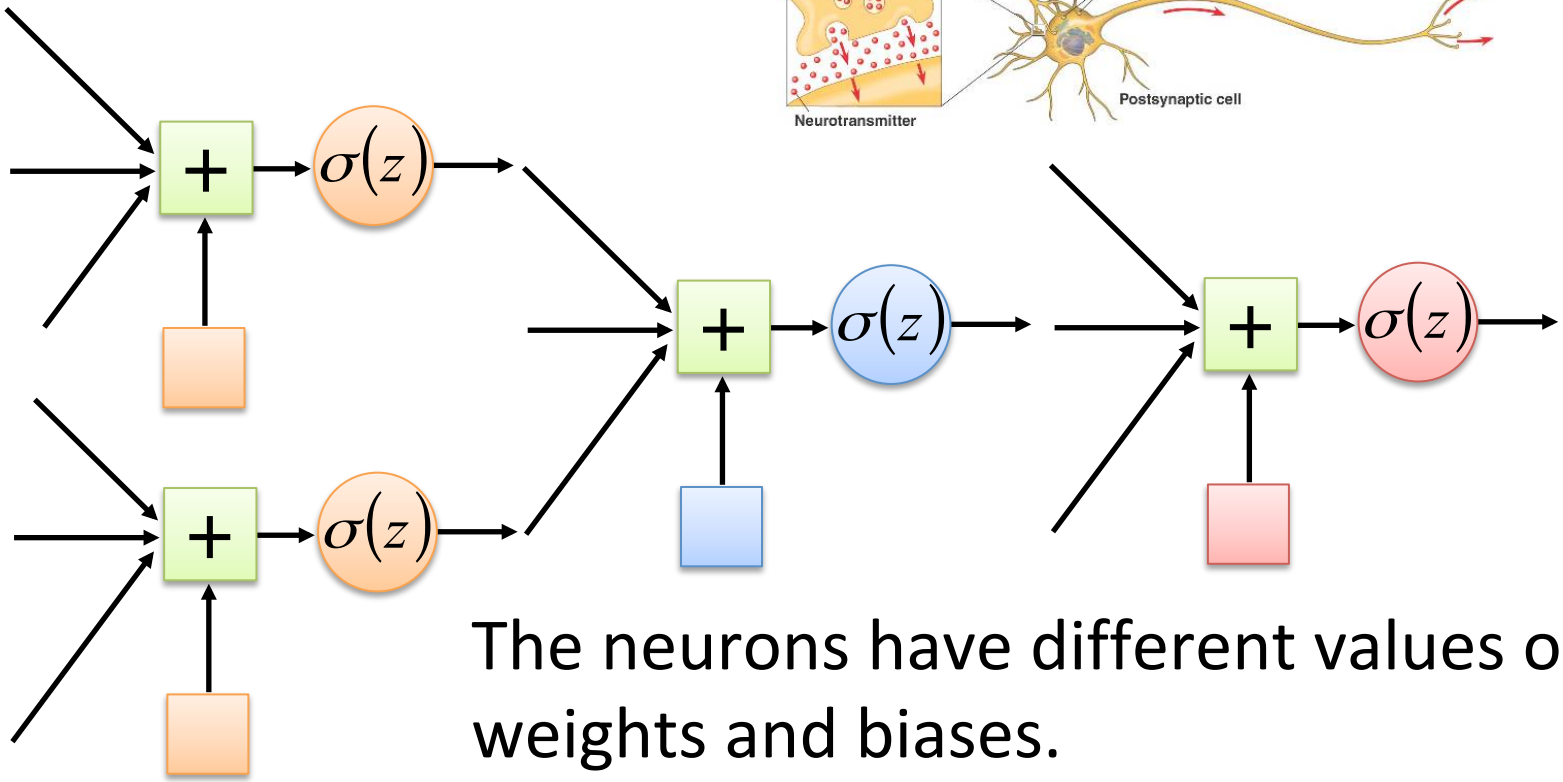




Neural Network



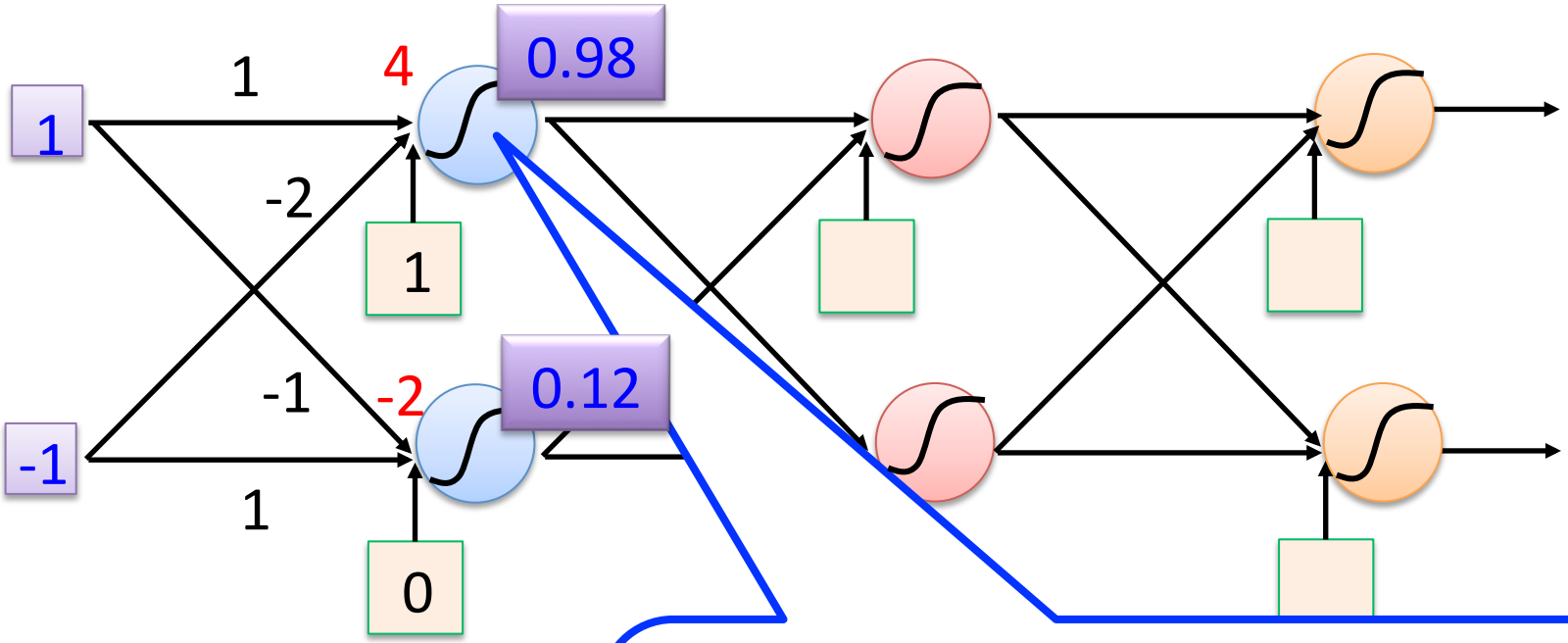
Different connections lead to different network structures



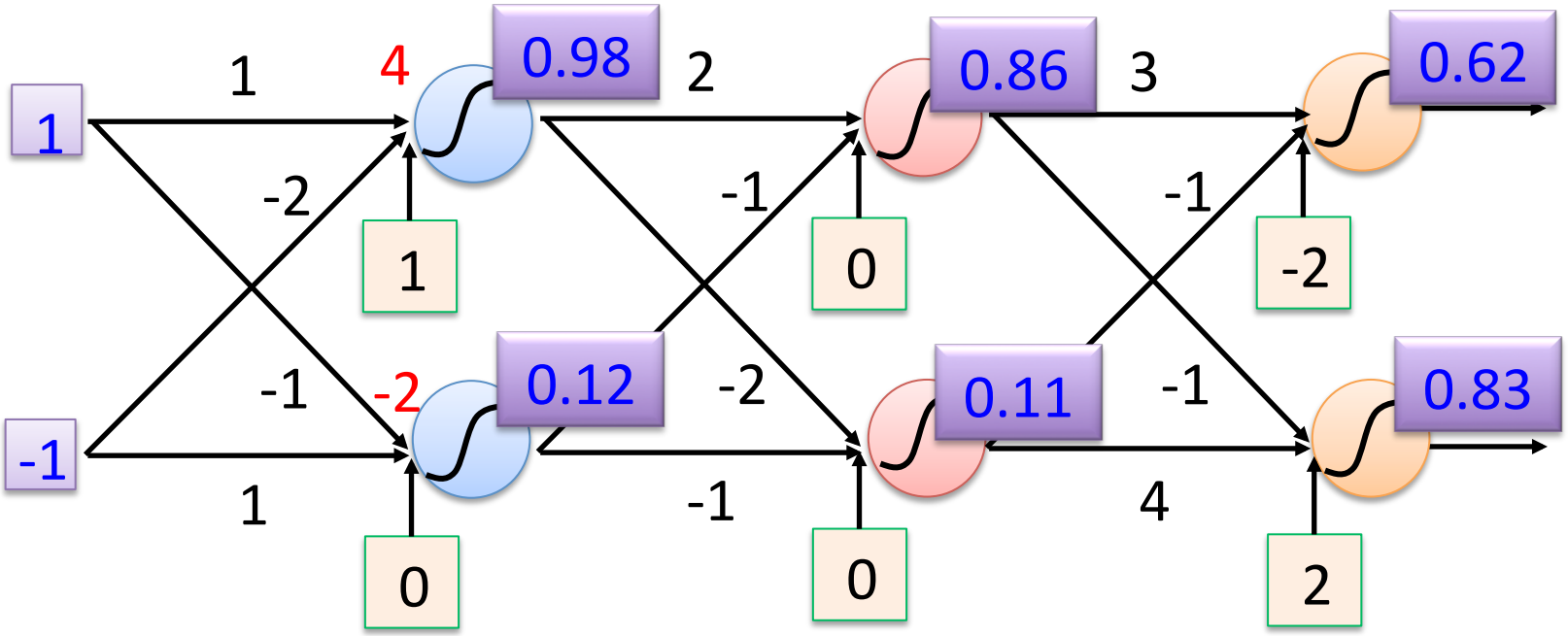
The neurons have different values of weights and biases.

Weights and biases are network parameters θ

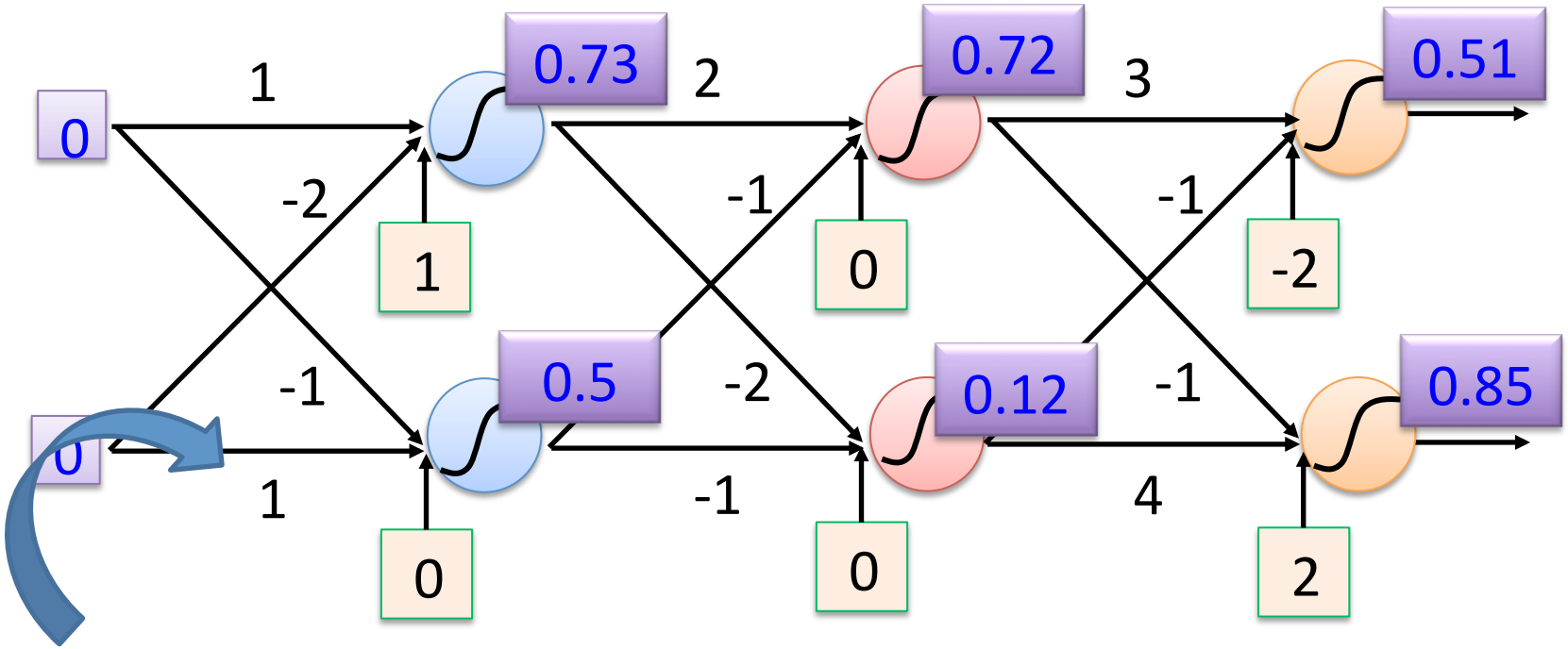
Fully Connect Feedforward Network



Fully Connect Feedforward Network



Fully Connect Feedforward Network



This is a function.

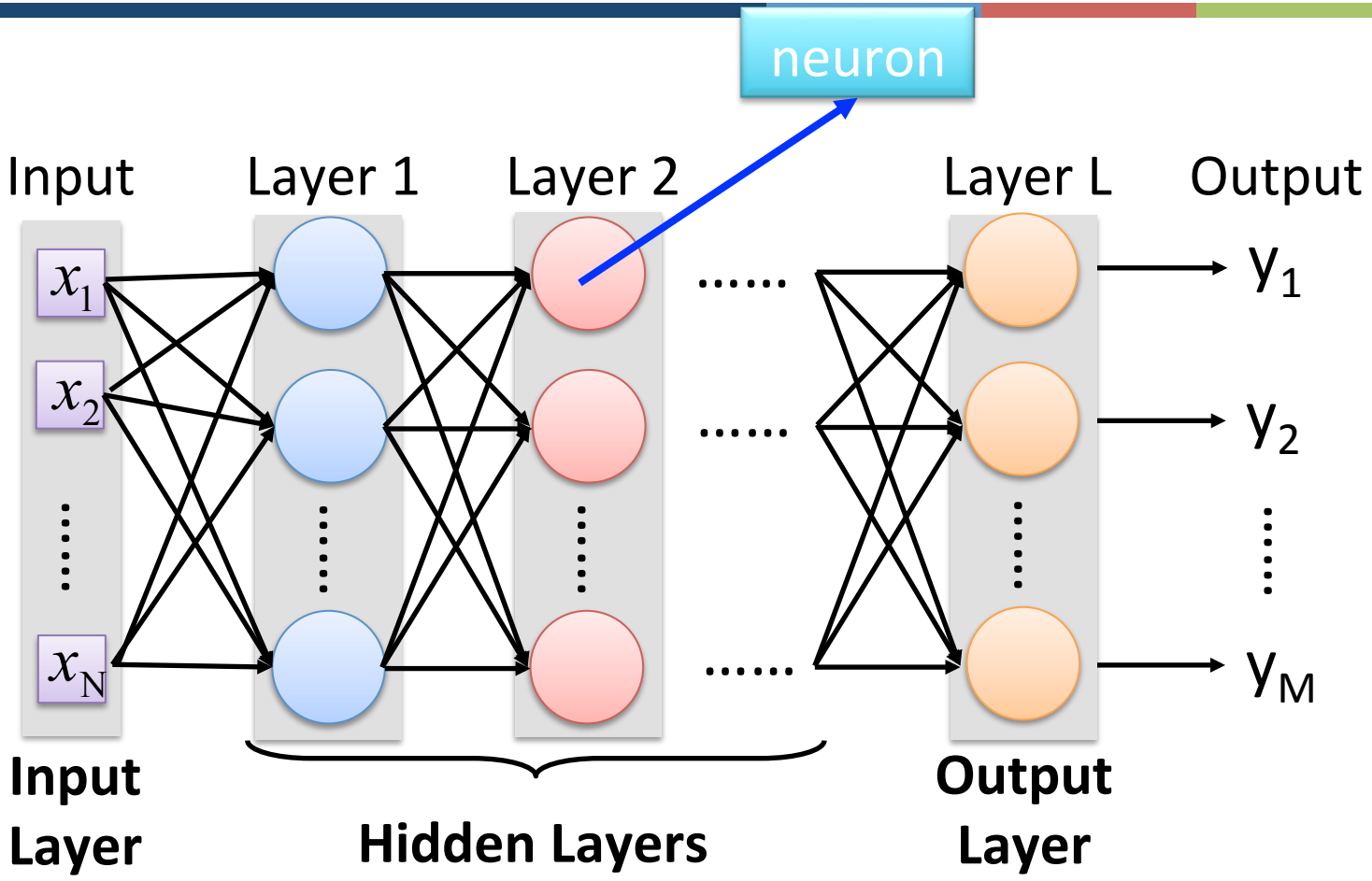
$f([\blacksquare 1@-1\ ])=[\blacksquare 0.62@0.83[\blacksquare 0@0\ ])=[\blacksquare 0.51@0.8$

Input vector, output vector

Given parameters θ, define a function

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

Fully Connect Feedforward Network



Deep means many hidden layers

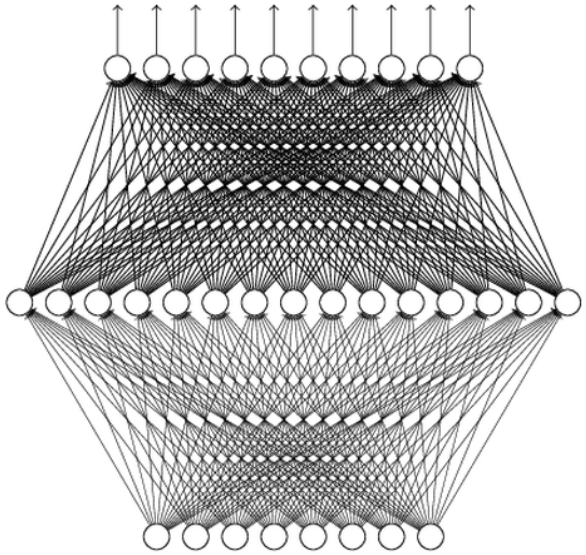
Why Deep? Universality Theorem

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?

Why Deep? Analogy

Logic circuits

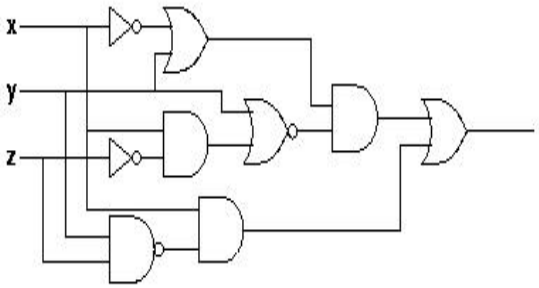
- Logic circuits consists of **gates**
- **A two layers of logic gates** can represent **any Boolean function.**
- Using multiple layers of logic gates to build some functions are much simpler

Neural network

- Neural network consists of **neurons**
- **A hidden layer network** can represent **any continuous function.**
- Using multiple layers of neurons to represent some functions are much simpler

→ less gates needed

→ less parameters → less data?



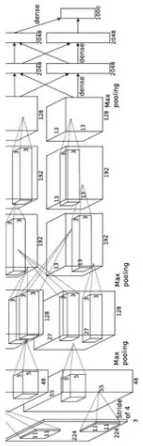
More reason: https://www.youtube.com/watch?v=XsC9byQkUH8&list=PLJV_el3uVTsPy9oCRY30oBPNLCo89yu49&index=13

Deep = Many hidden layers

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

8 layers

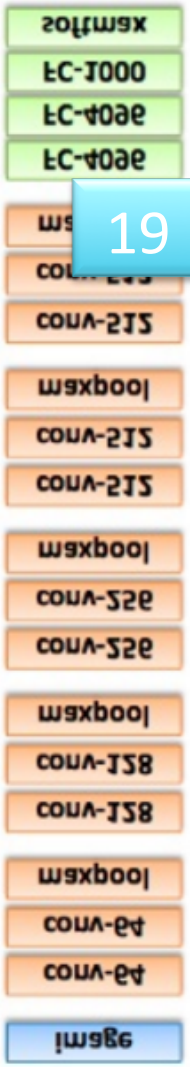
16.4%



AlexNet (2012)

19 layers

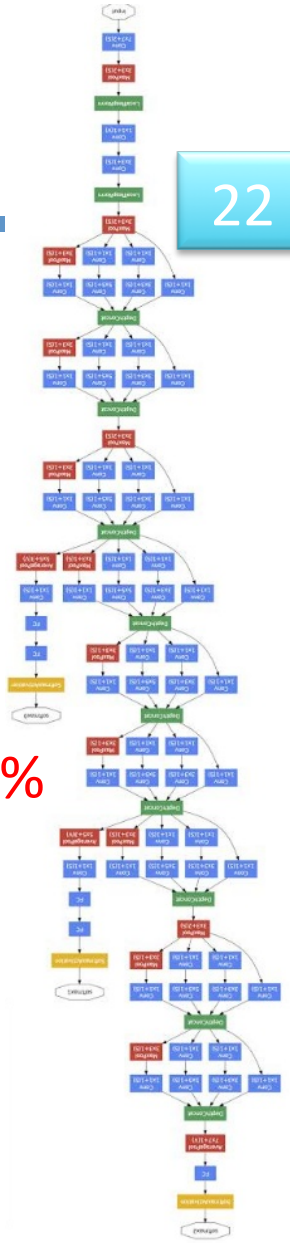
7.3%



VGG (2014)

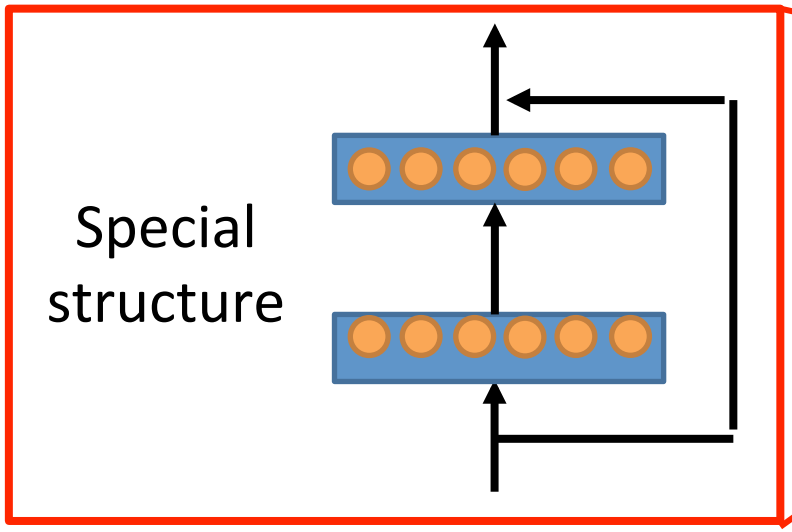
22 layers

6.7%



GoogleNet (2014)

Deep = Many hidden layers



152 layers

101 layers

3.57%

16.4%

AlexNet
(2012)

7.3%

VGG
(2014)

6.7%

GoogleNet
(2014)

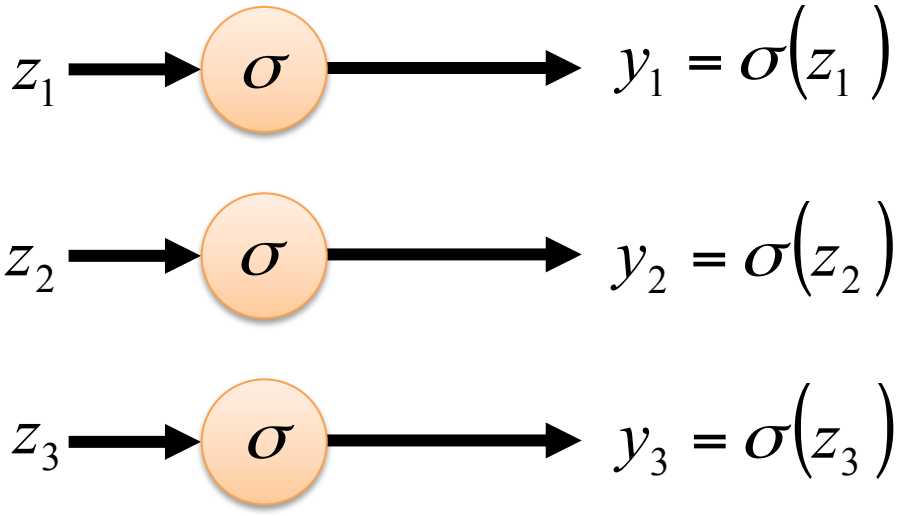
Residual Net
(2015)

Taipei
101

Output Layer

- Softmax layer as the output layer

Ordinary Layer



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

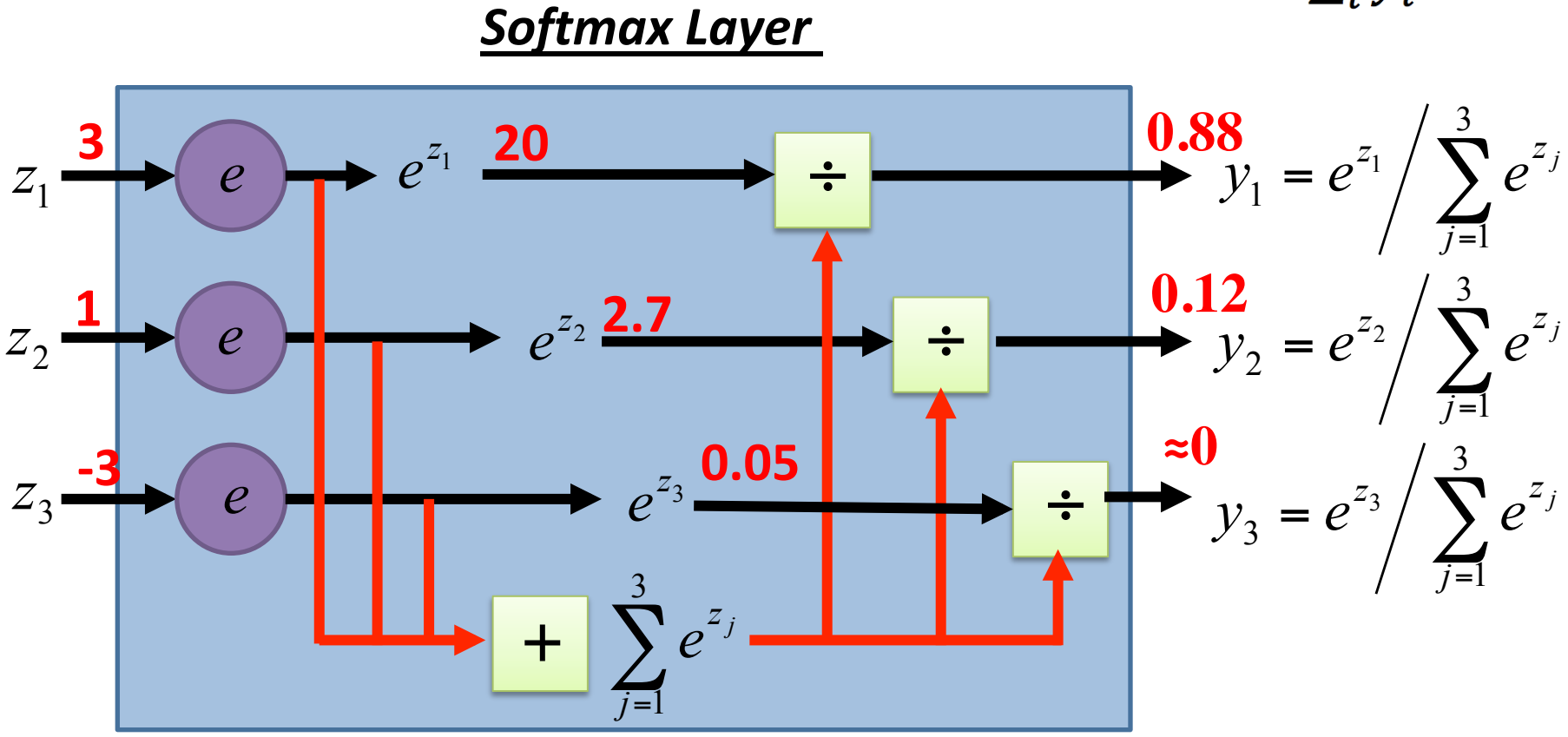
May not be easy to interpret

Output Layer

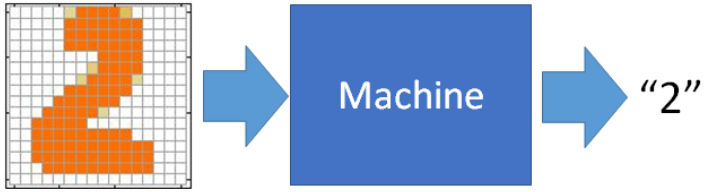
- Softmax layer as the output layer

Probability:

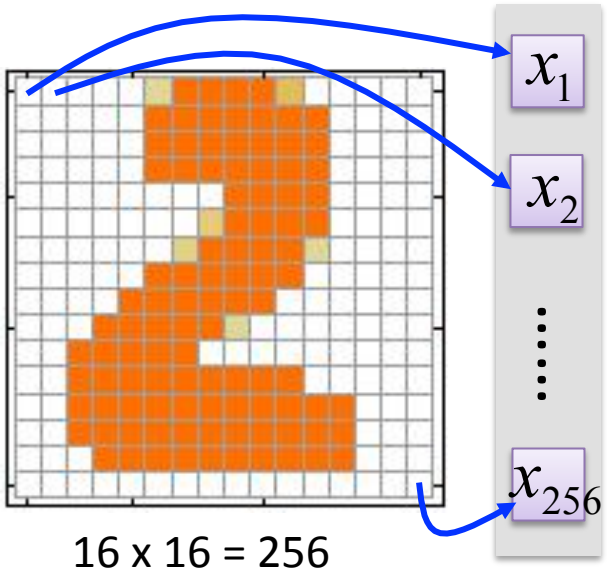
- $1 > y_i > 0$
- $\sum_i y_i = 1$



Example Application



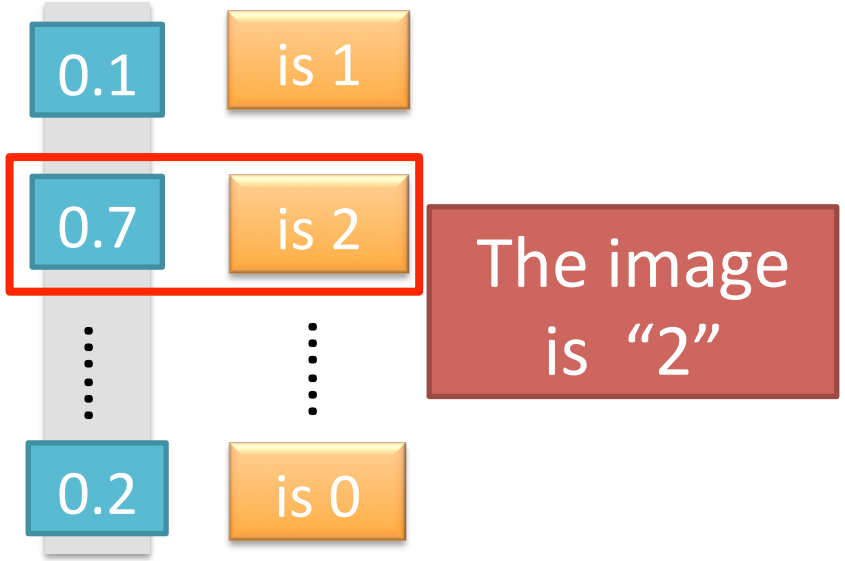
Input



Ink → 1

No ink → 0

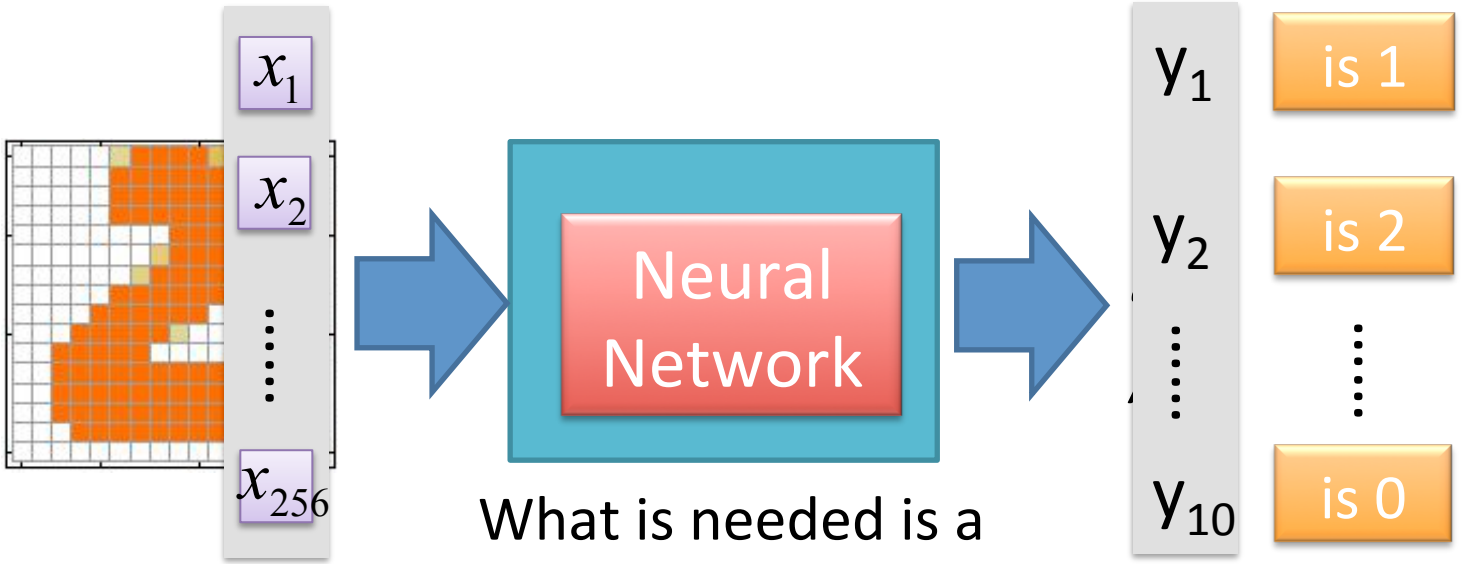
Output



Each dimension represents the confidence of a digit.

Example Application

- Handwriting Digit Recognition

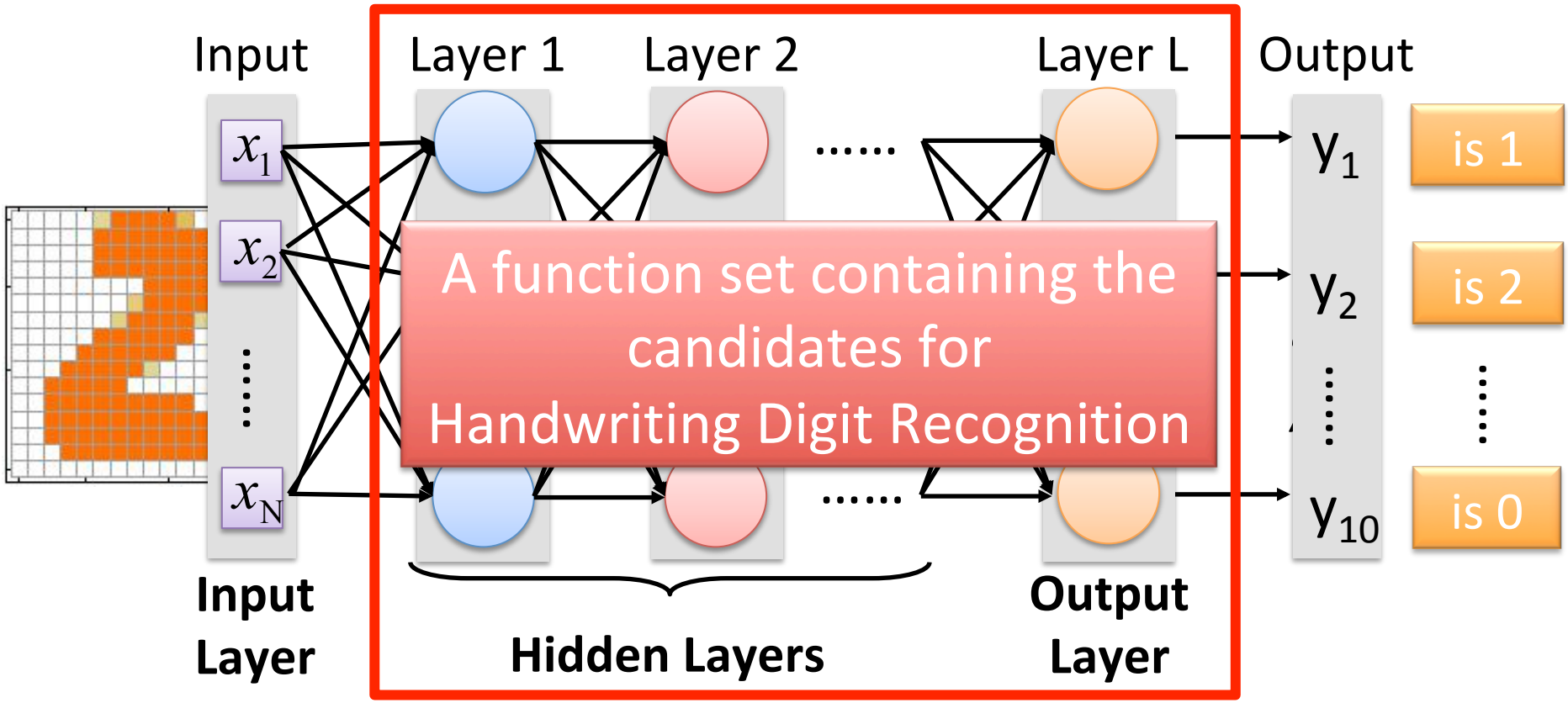


What is needed is a function ……

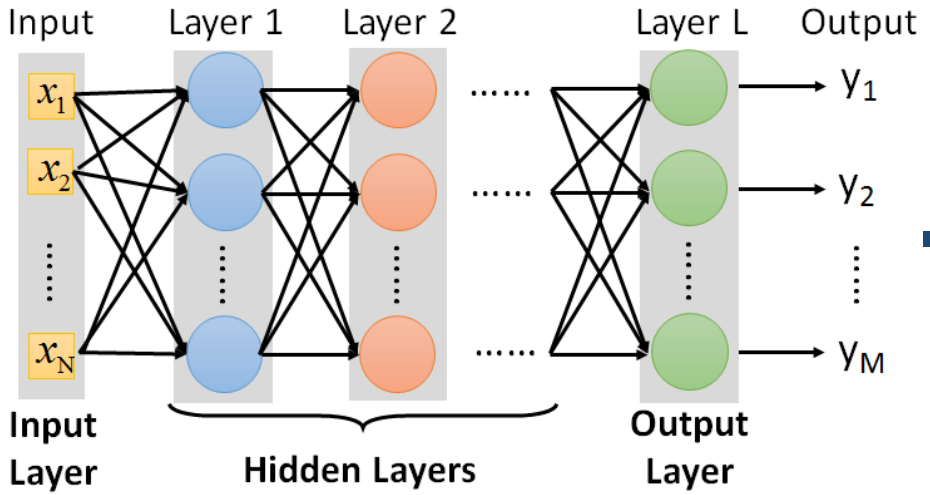
Input:
256-dim vector

output:
10-dim vector

Example Application



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



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

Trial and Error + Intuition

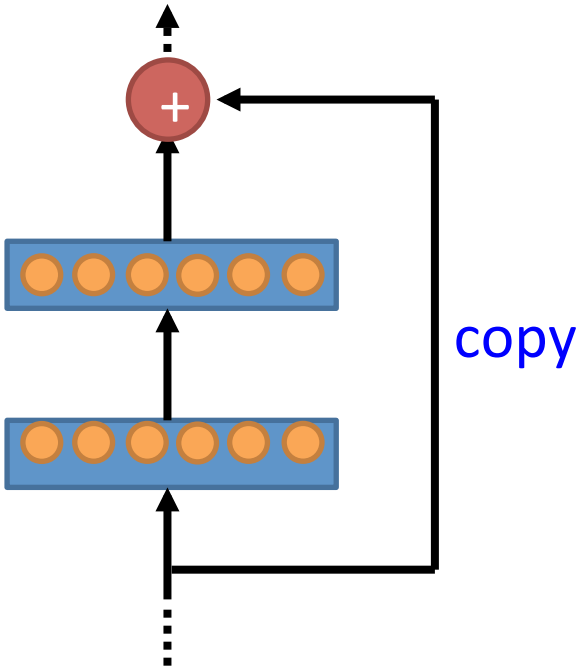
- Q: Can we design the network structure?

Convolutional Neural Network (CNN) in the next lecture

- Q: Can the structure be automatically determined?
  - Yes, but not widely studied yet.

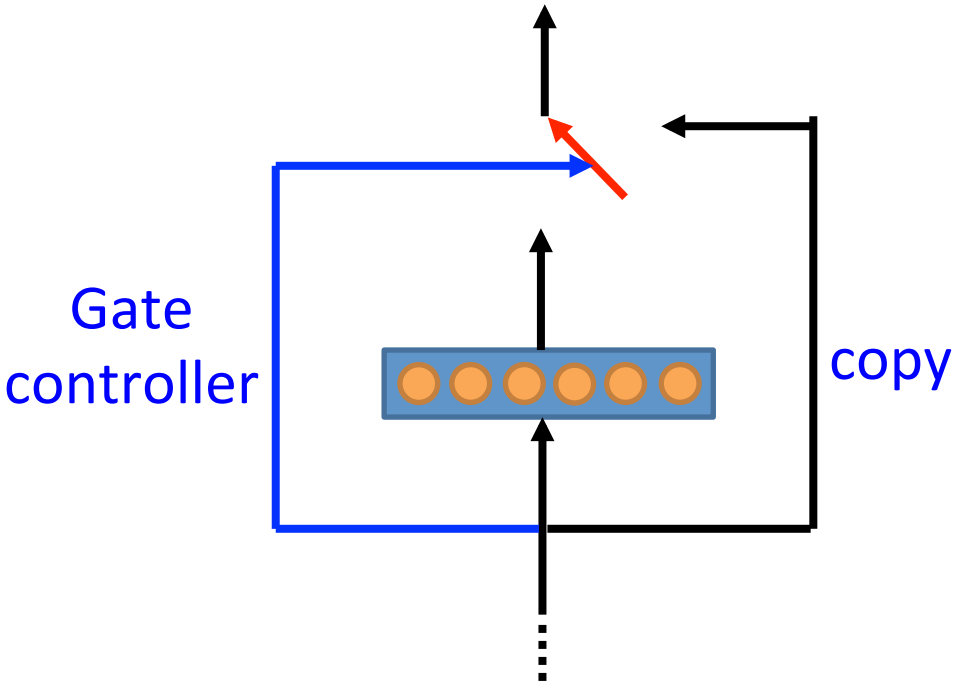
Highway Network

- **Residual Network**

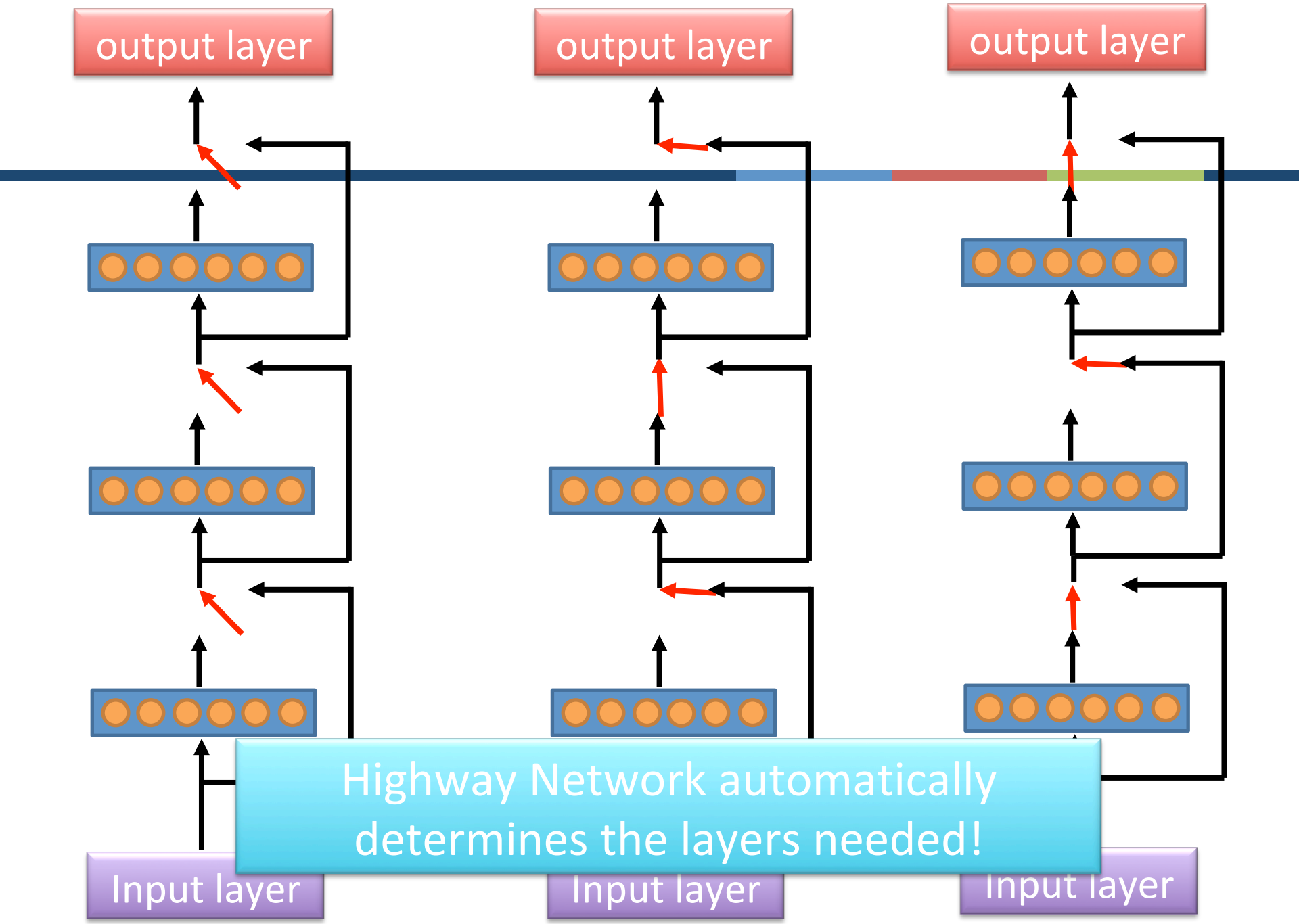


Deep Residual Learning for Image Recognition
http://arxiv.org/abs/1512.03385

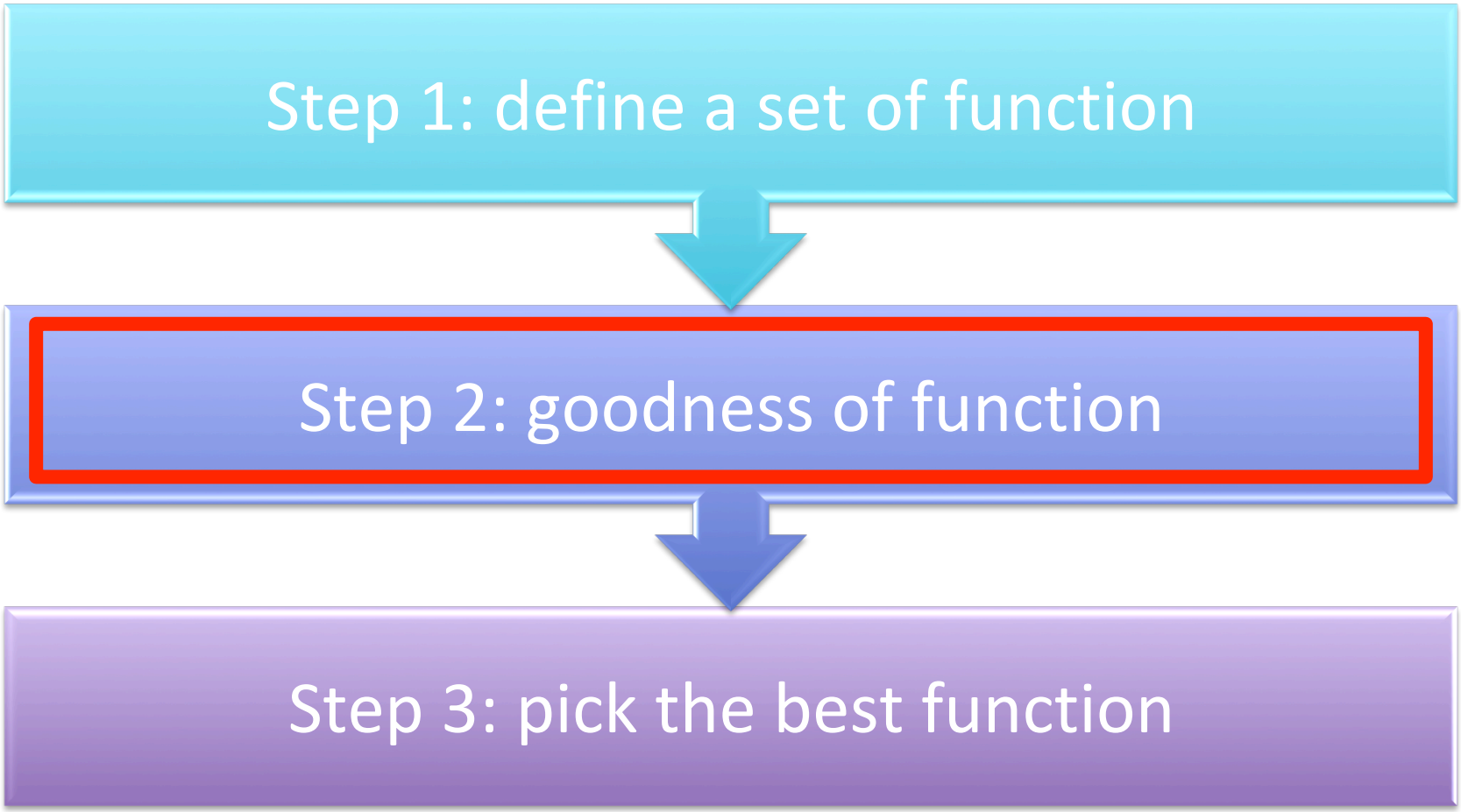
- **Highway Network**



Training Very Deep Networks
https://arxiv.org/pdf/1507.06228v2.pdf

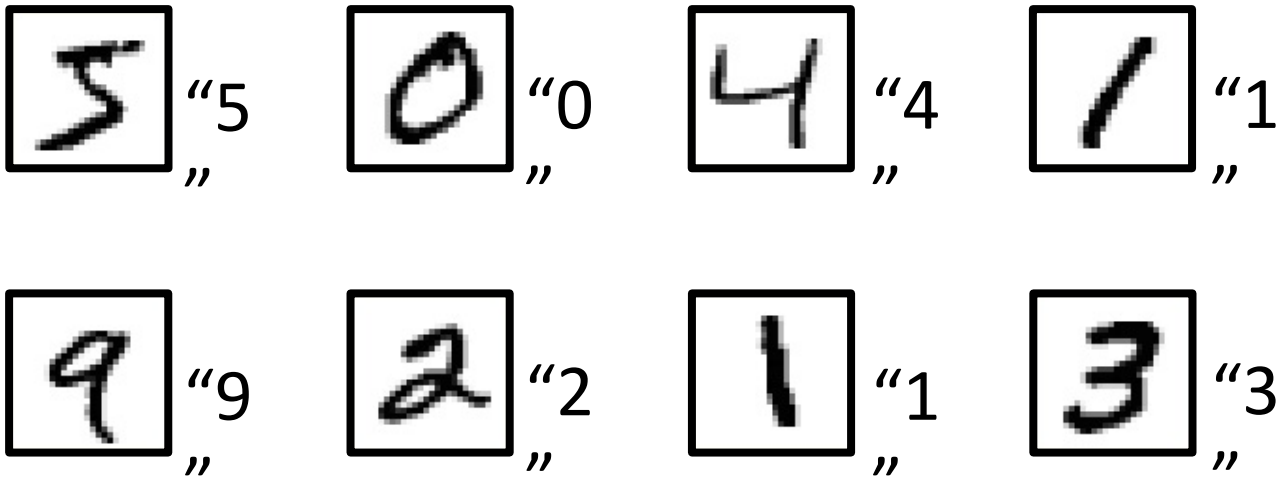

output layer
output layer
output layer
Highway Network automatically determines the layers needed!
Input layer
Input layer
Input layer

Three Steps for Deep Learning



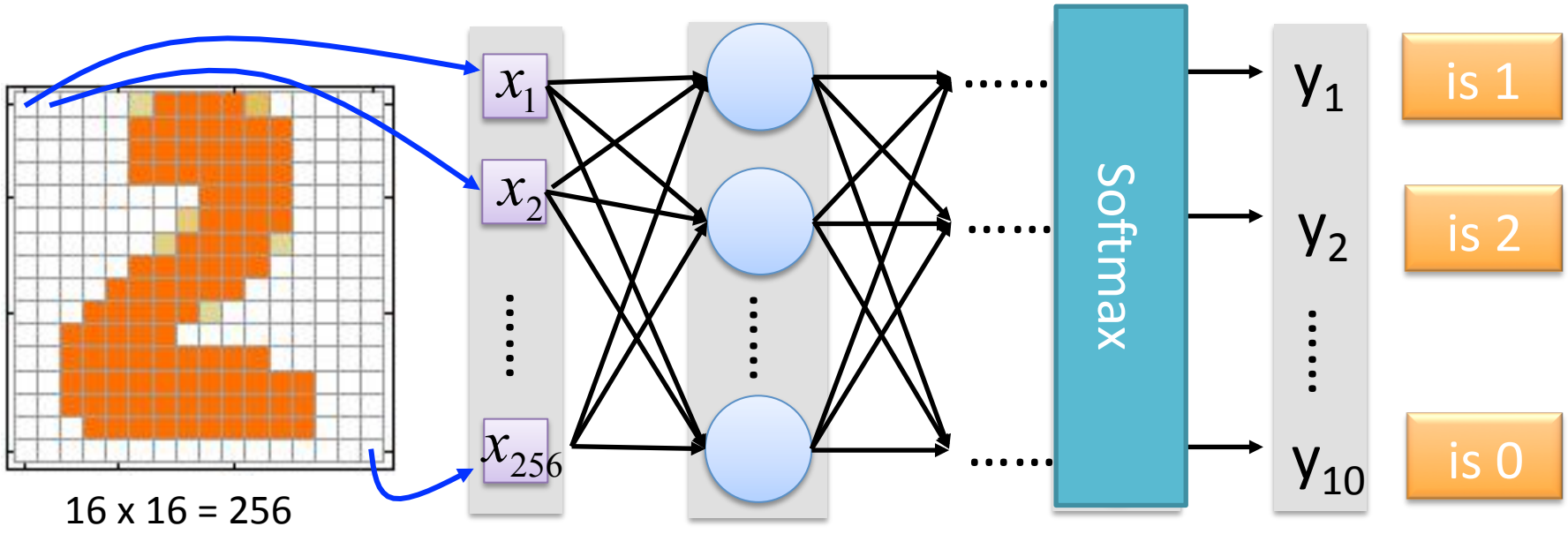
Training Data

- Preparing training data: images and their labels



The learning target is defined on the training data.

Learning Target



16 x 16 = 256

Ink → 1

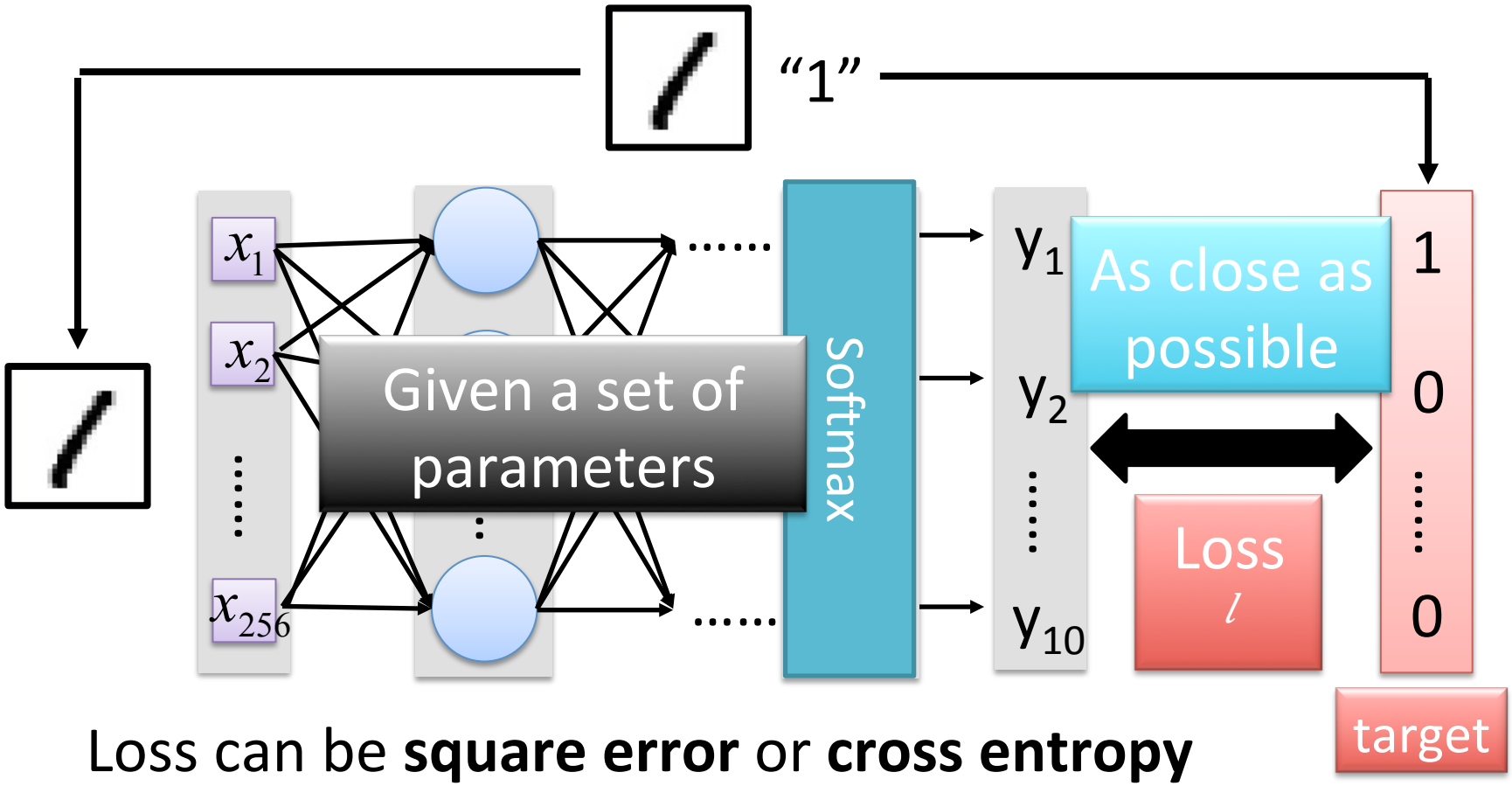
No ink → 0

The learning target is ……

Input:  → y_1 has the maximum value

Input:  → y_2 has the maximum value

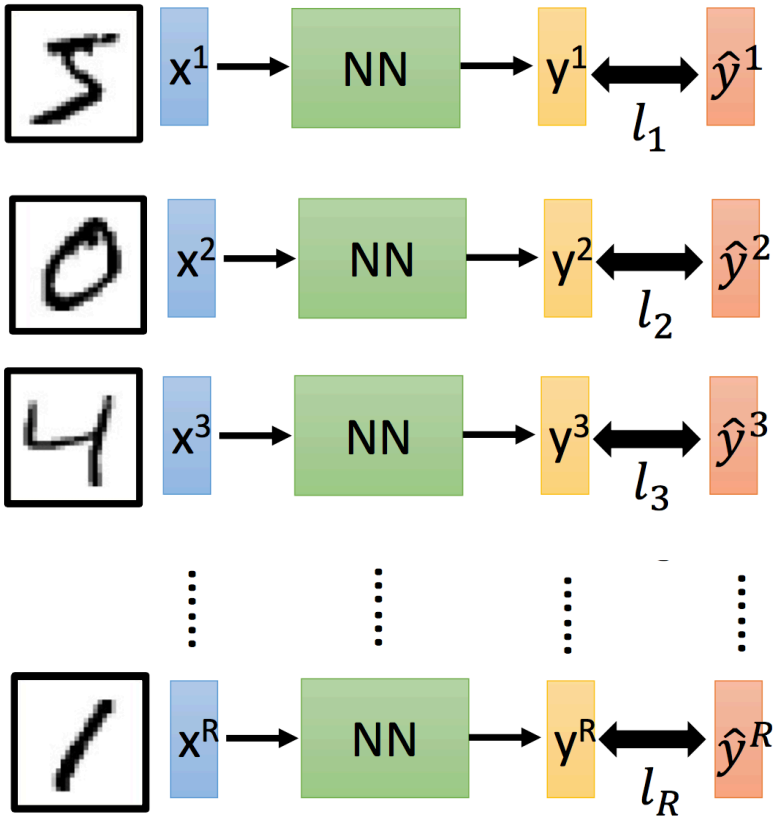
A good function should make the loss of all examples as small as possible.



Loss can be **square error** or **cross entropy** between the network output and target

Total Loss

For all training data …



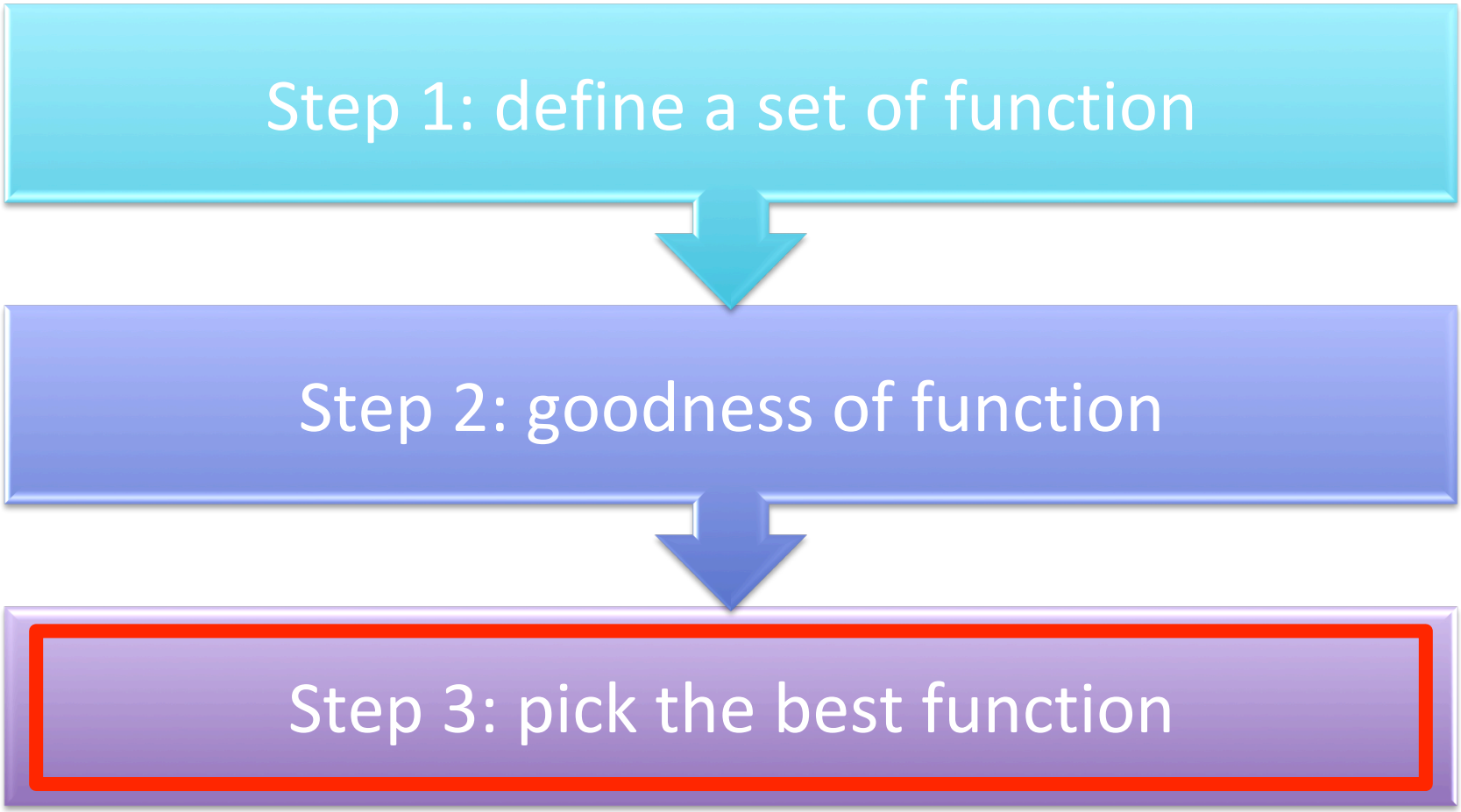
Total Loss: $L = \sum_{r=1}^{R} l_r$

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

Three Steps for Deep Learning



How to pick the best function

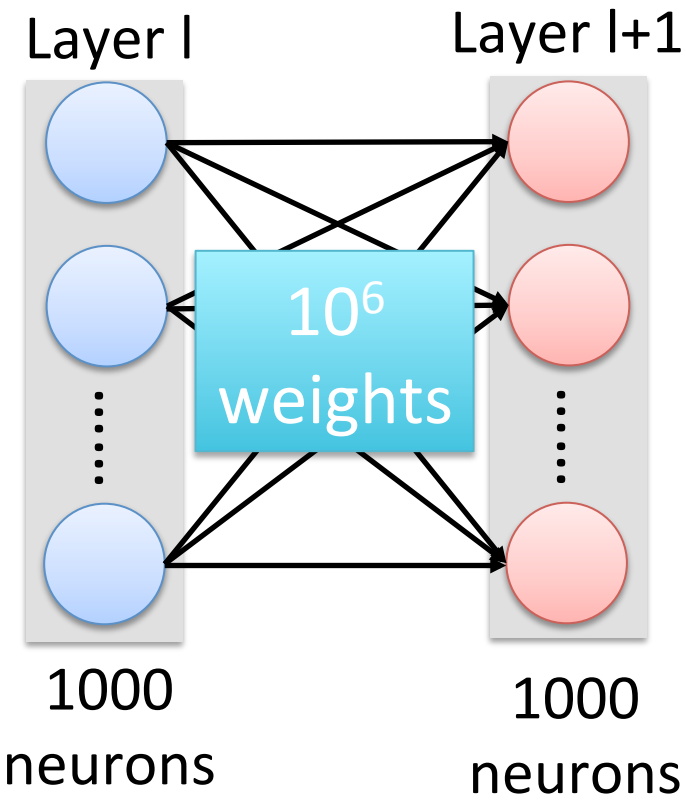
Find ***network parameters*** $\theta^* = \{w_1, w_2, \cdots, b_1, b_2, \cdots\}$ that minimize total loss L

Enumerate all possible values

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

Millions of parameters

E.g. speech recognition: 8 layers and 1000 neurons each layer



Gradient Descent

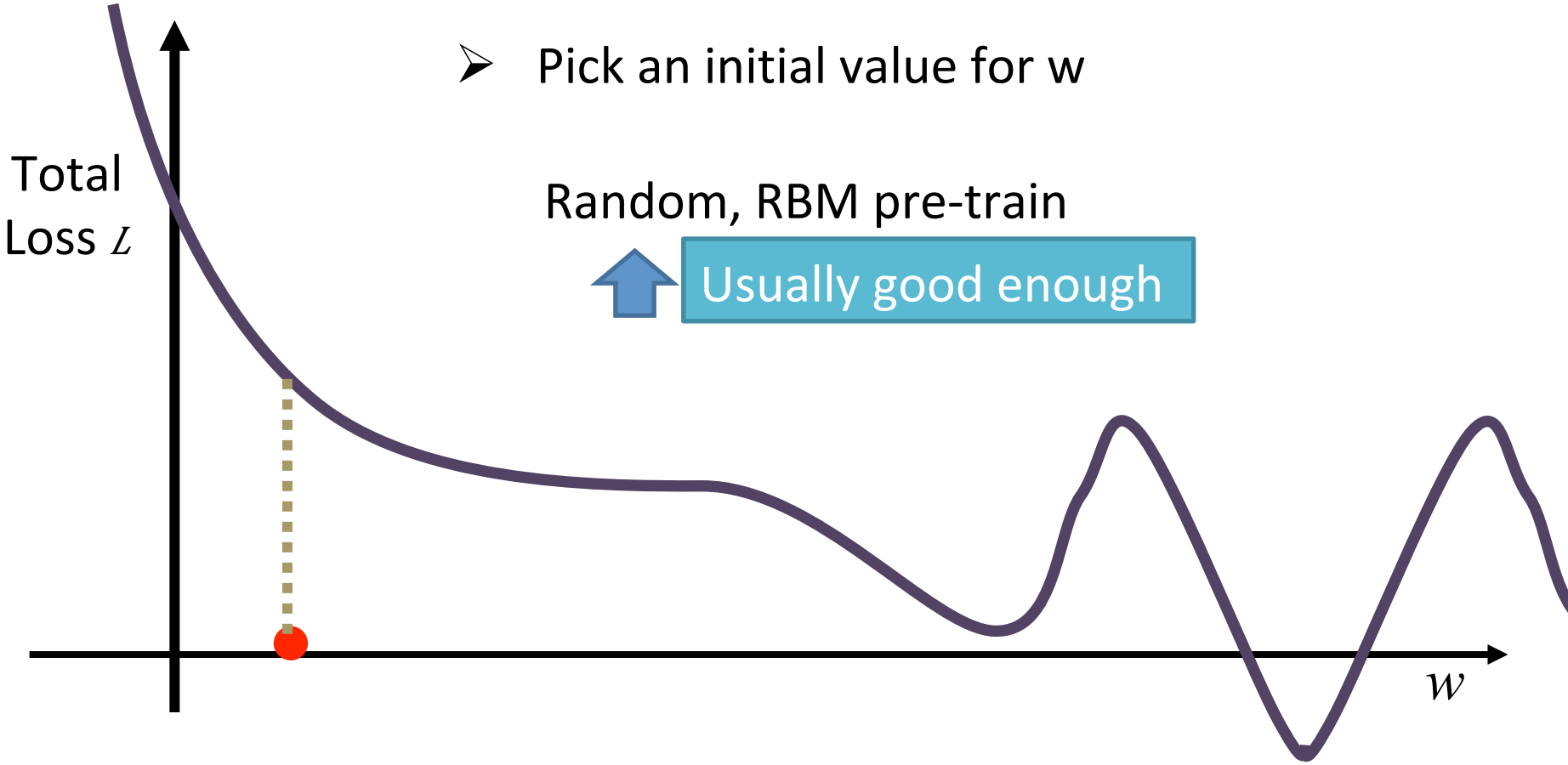
Network parameters $\theta=\{w_1, w_2, \ldots, b_1, b_2, \ldots\}$

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

- Pick an initial value for w

Random, RBM pre-train

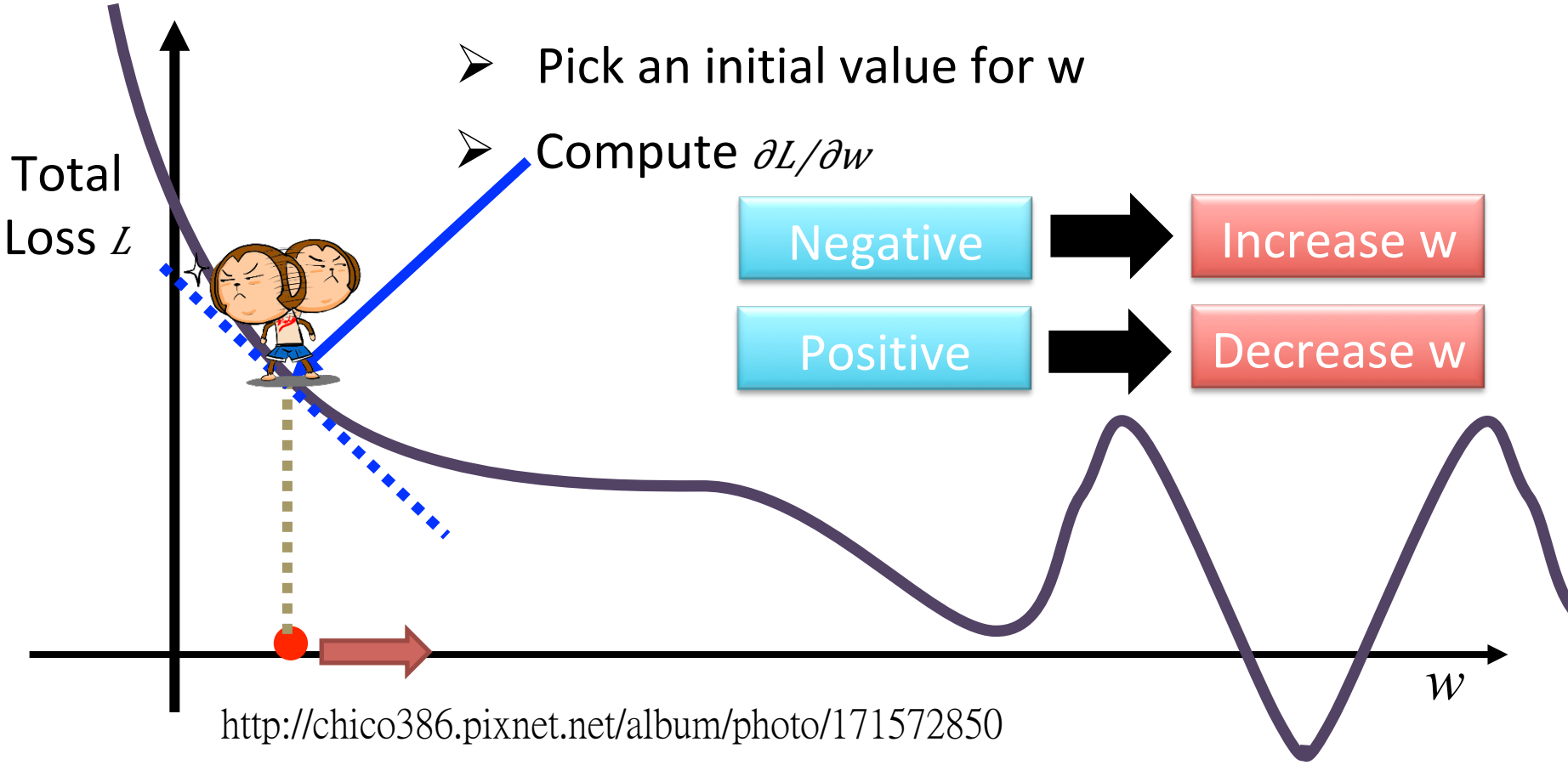
Usually good enough



Gradient Descent

Network parameters $\theta=\{w_1, w_2, \ldots, b_1, b_2, \ldots\}$

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

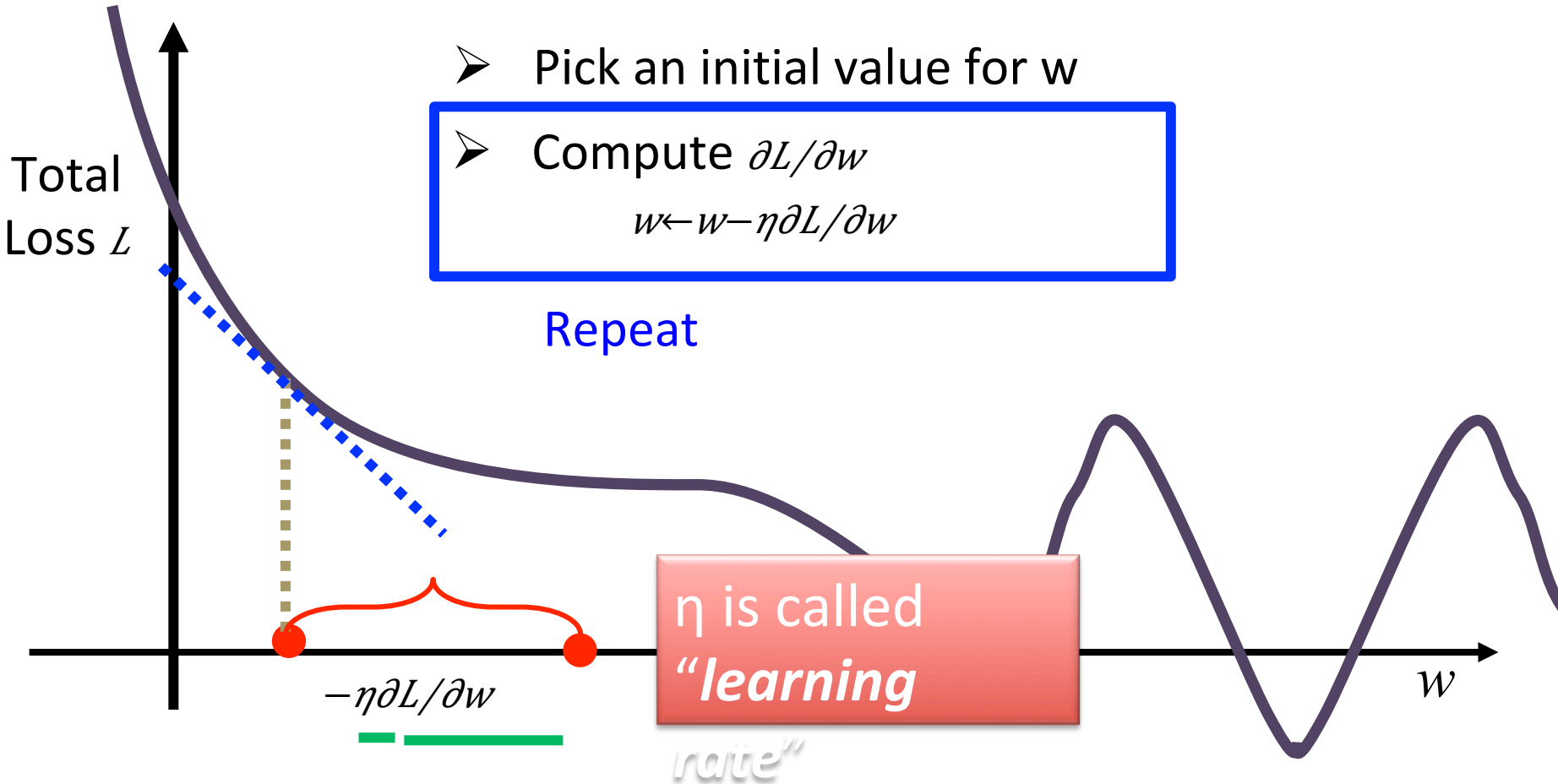


http://chico386.pixnet.net/album/photo/171572850

Gradient Descent

Network parameters $\theta=\{w_1, w_2, \ldots, b_1, b_2, \ldots\}$

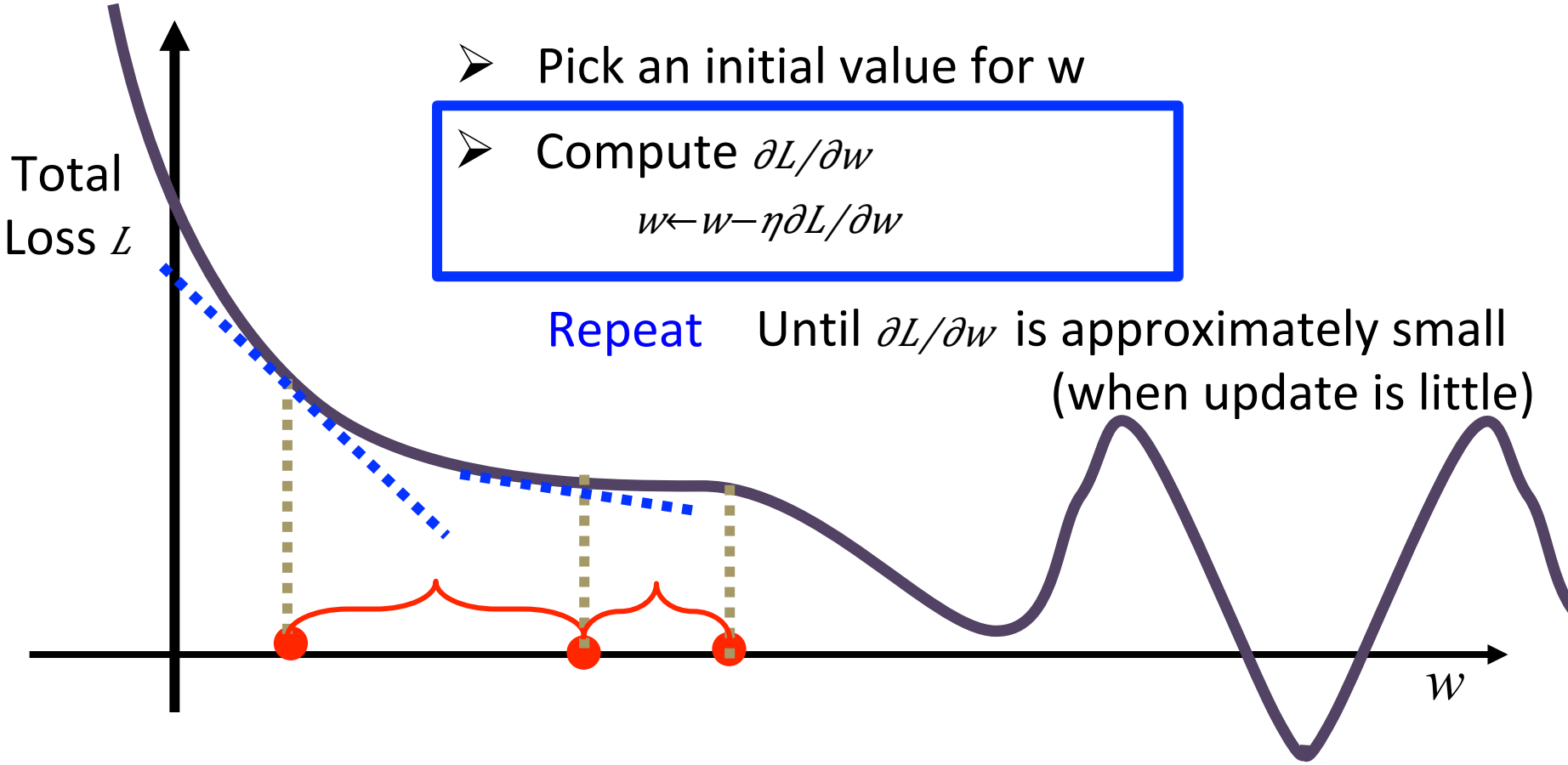
Find ***network parameters θ^**** that minimize total loss L



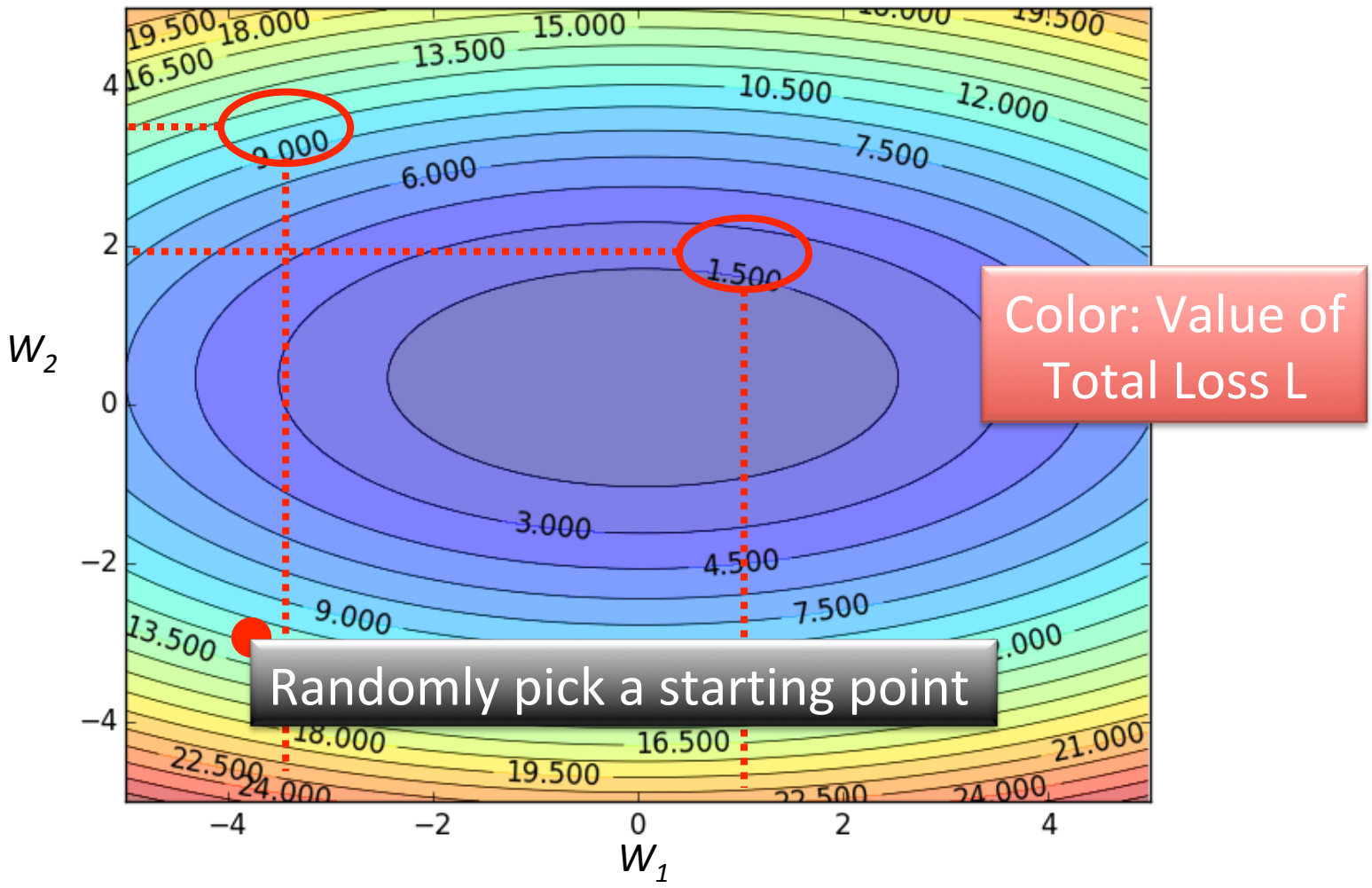
Gradient Descent

Network parameters $\theta=\{w_1, w_2, \ldots, b_1, b_2, \ldots\}$

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

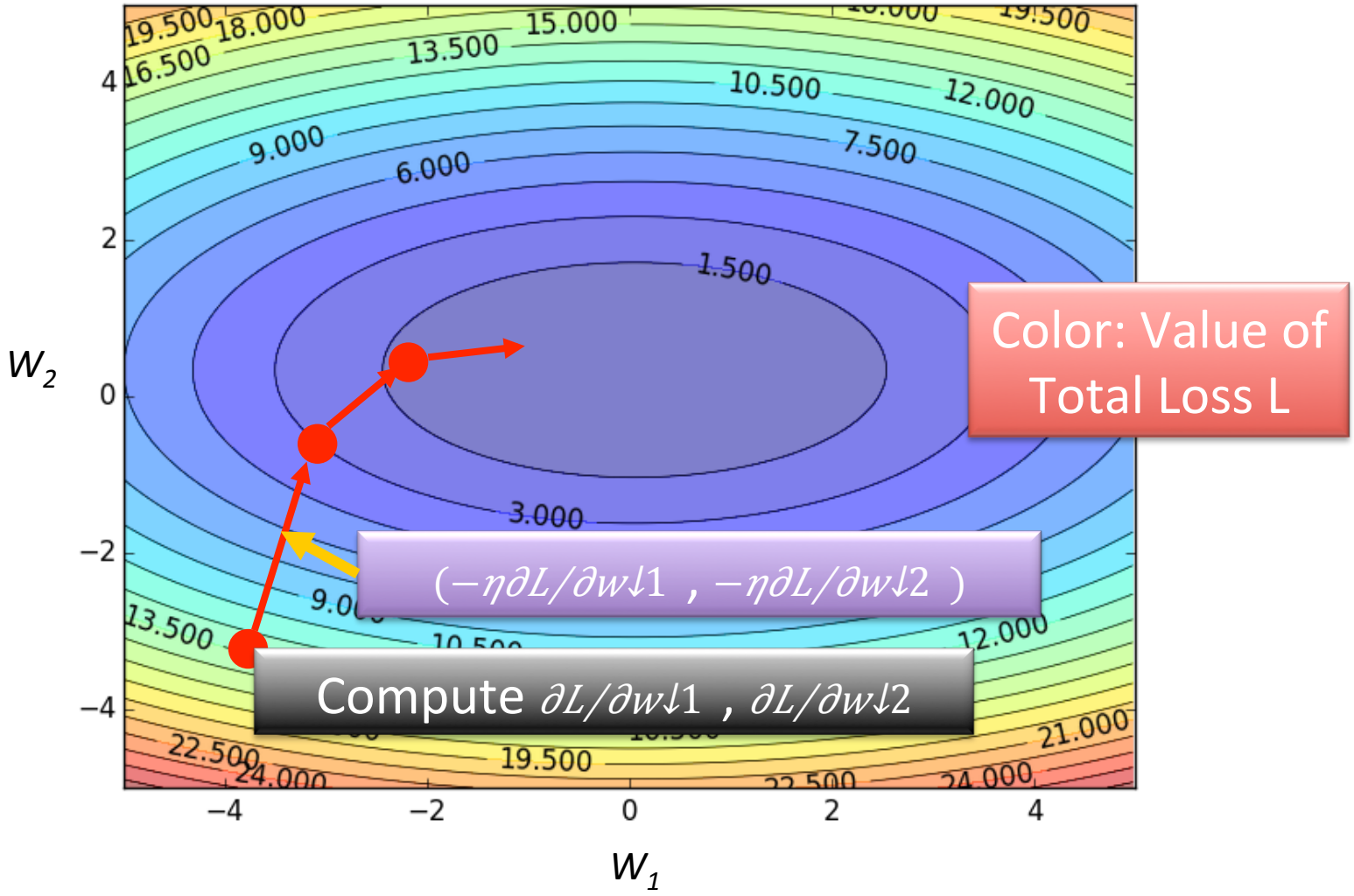


Gradient Descent

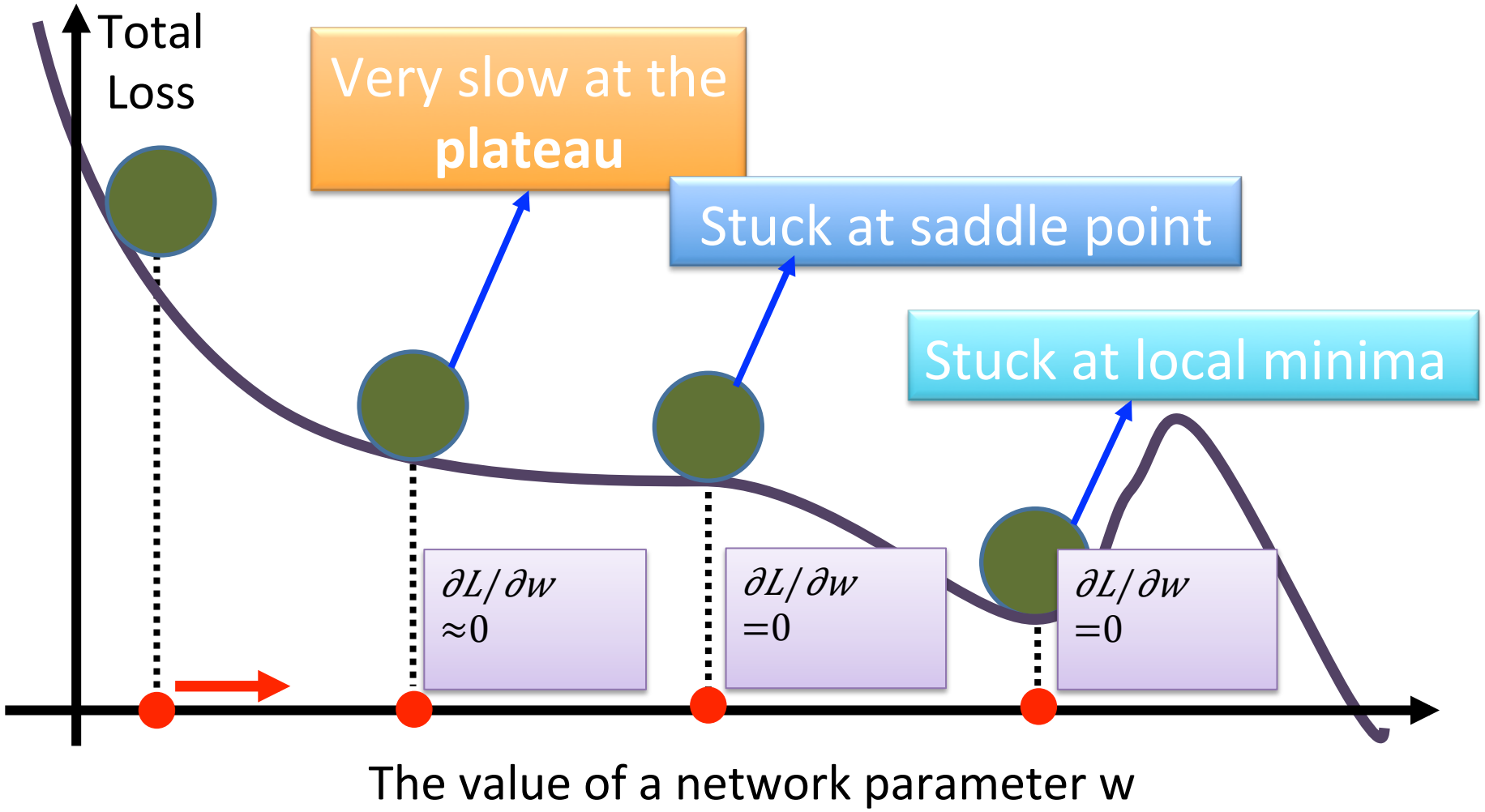


Gradient Descent

Hopfully, we would reach a minima .....

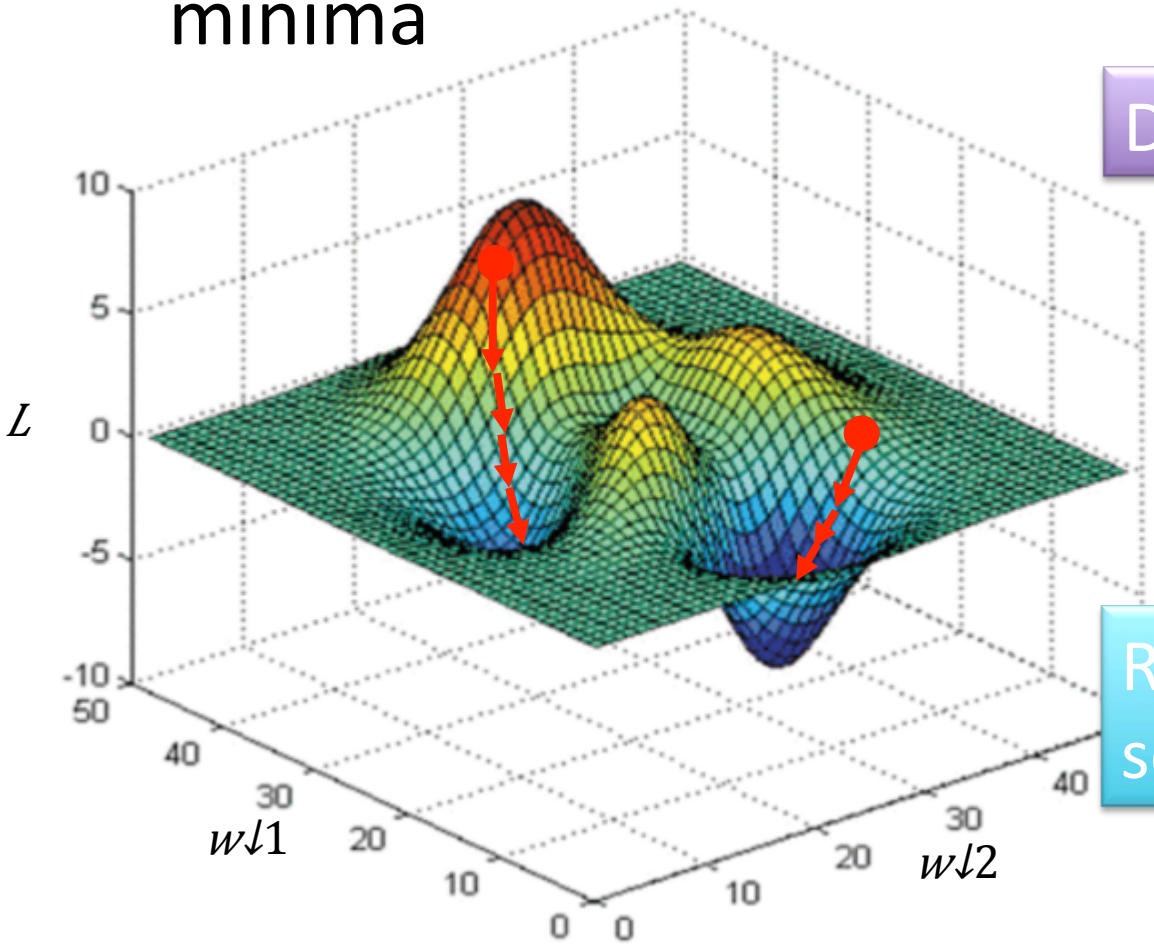


Local Minima



Local Minima

- Gradient descent never guarantee global minima



Different initial point

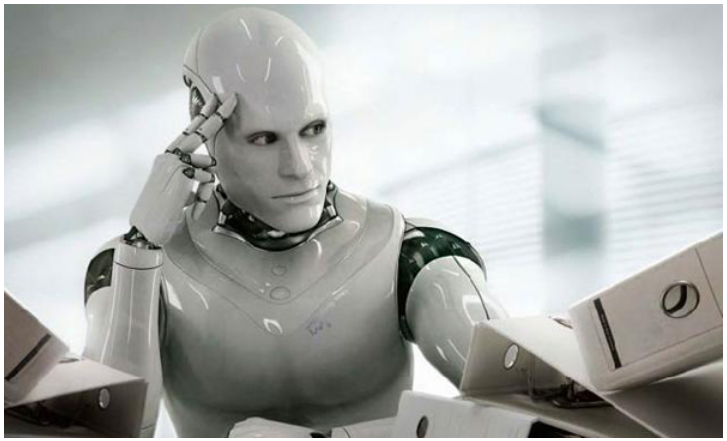
Reach different minima, so different results

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

Deep Learning library produced by Amazon
DSSTNE

Three Steps for Deep Learning

Deep Learning is so simple ……

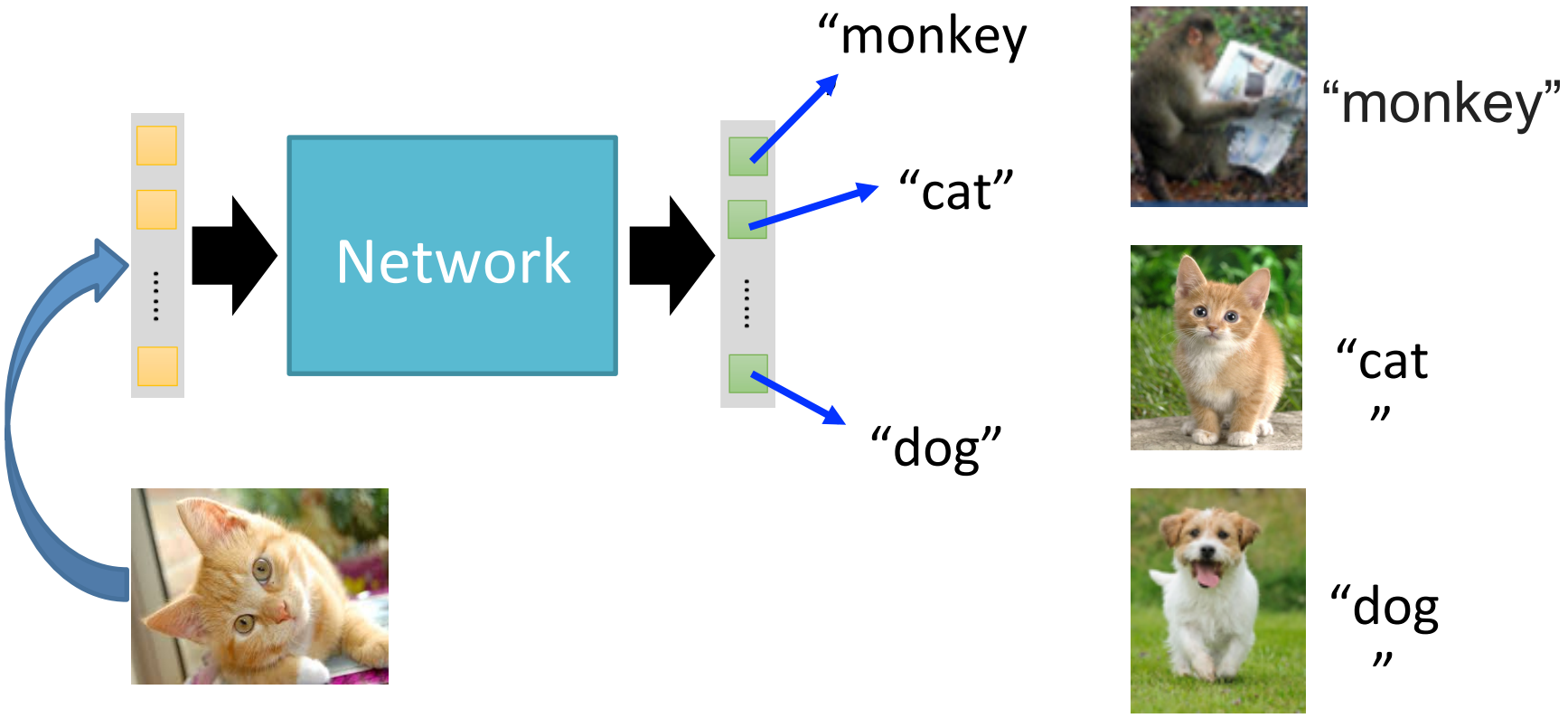
Now If you want to find a function

If you have lots of function input/output (?) as training data

→ You can use deep learning

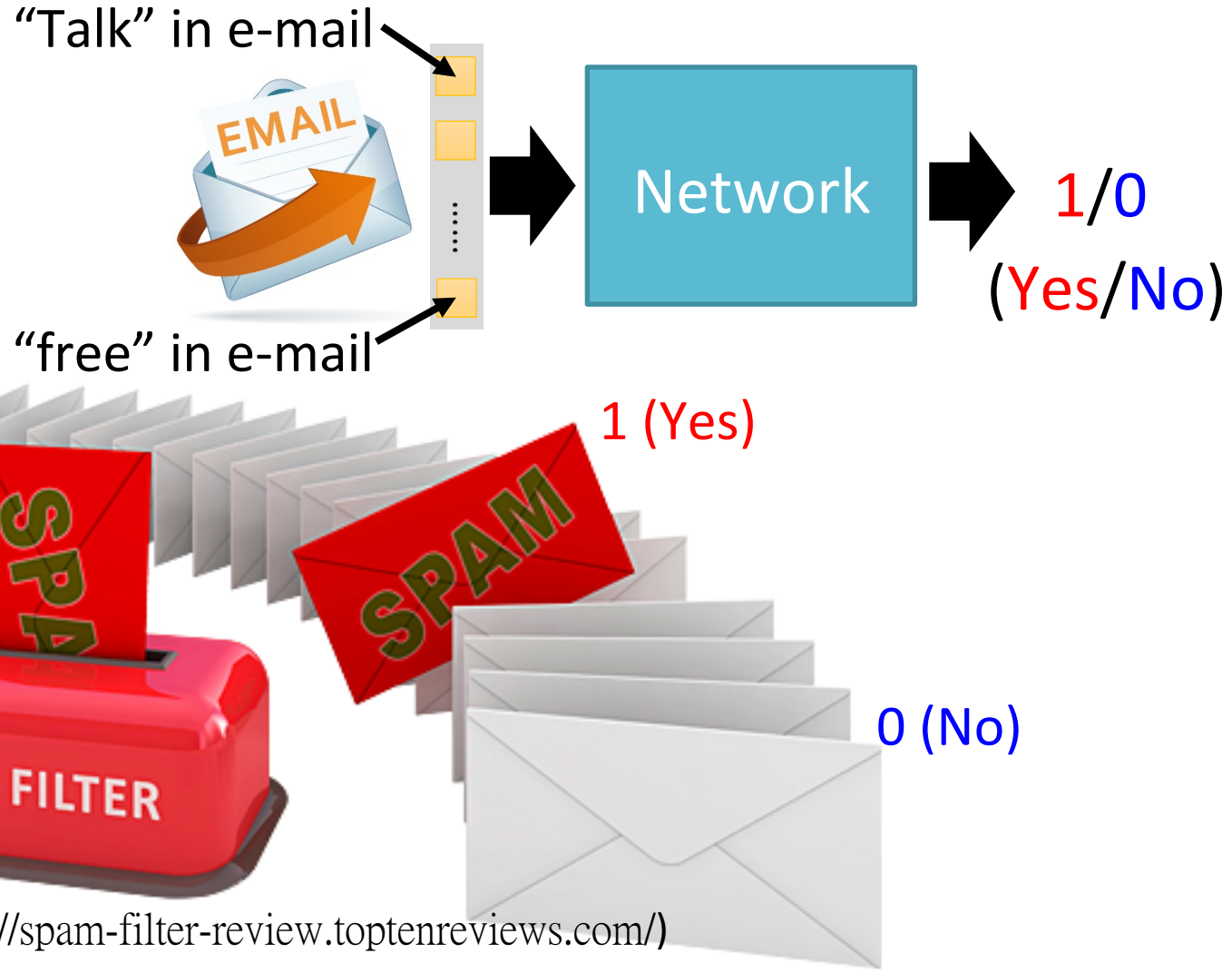
For example, you can do …….

- Image Recognition



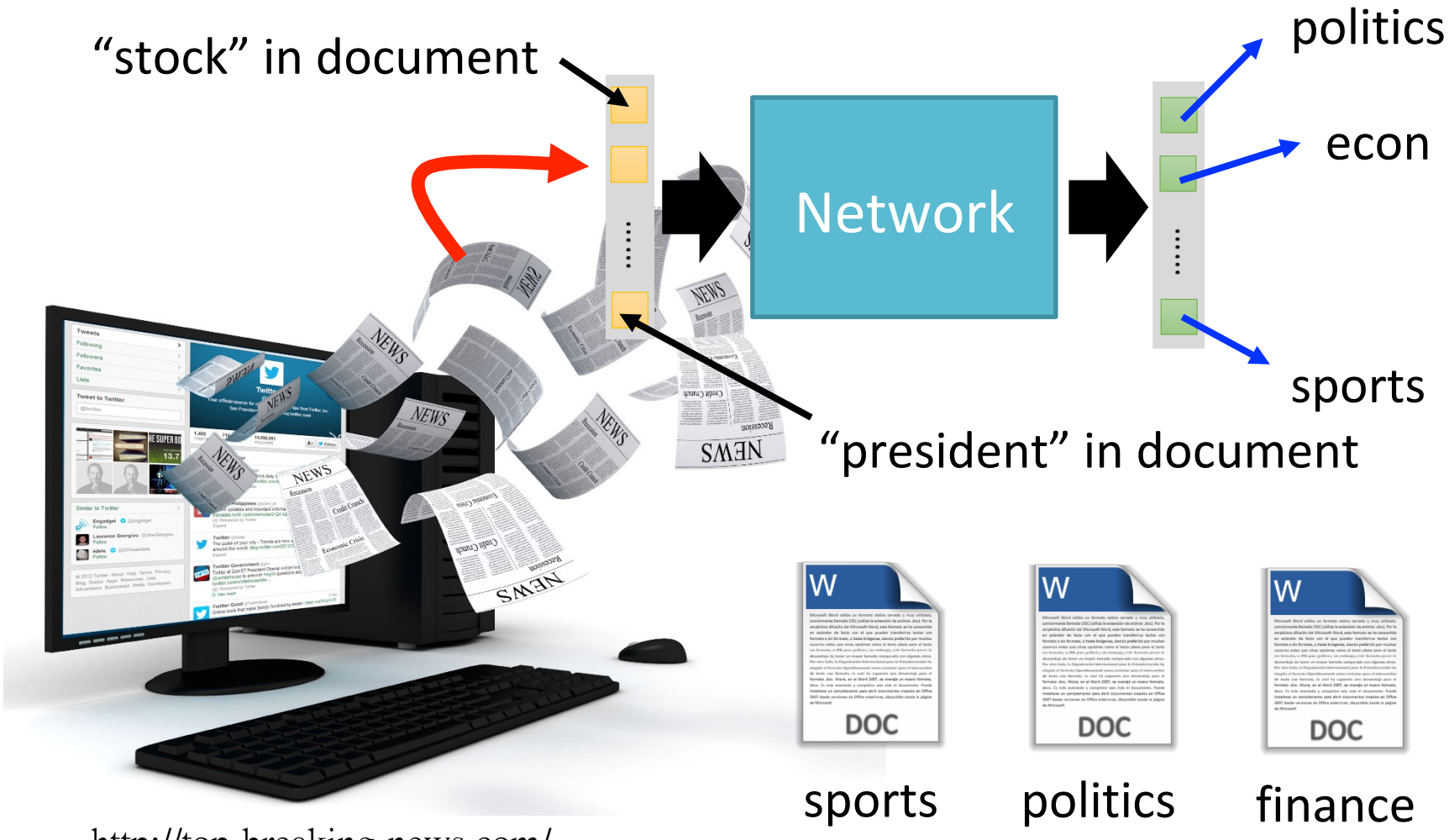
For example, you can do …….

Spam filtering



(http://spam-filter-review.toptenreviews.com/)

For example, you can do …….



http://top-breaking-news.com/

Outline

Introduction of Deep Learning

"Hello World" for Deep Learning

Tips for Deep Learning

Keras

or theano

Very flexible

Need some effort to learn

Interface of TensorFlow or Theano

keras

Easy to learn and use

(still have some flexibility)

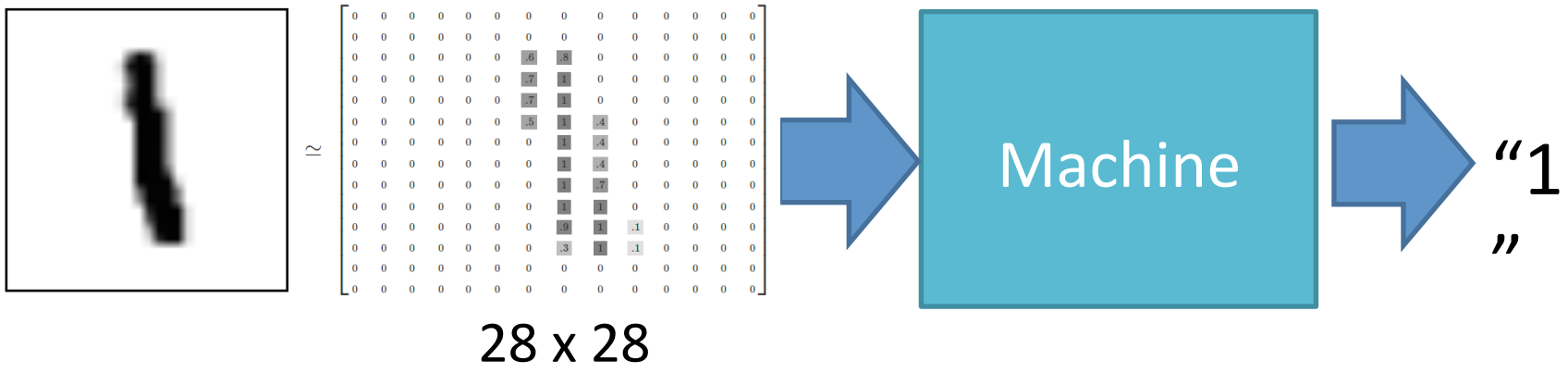
You can modify it if you can write TensorFlow or Theano

Keras

- François Chollet is the author of Keras.
  - He currently works for Google as a deep learning engineer and researcher.
- Keras means *horn* in Greek
- Documentation: http://keras.io/
- Example: https://github.com/fchollet/keras/tree/master/examples

Example Application

- Handwriting Digit Recognition



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

“Hello world” for deep learning

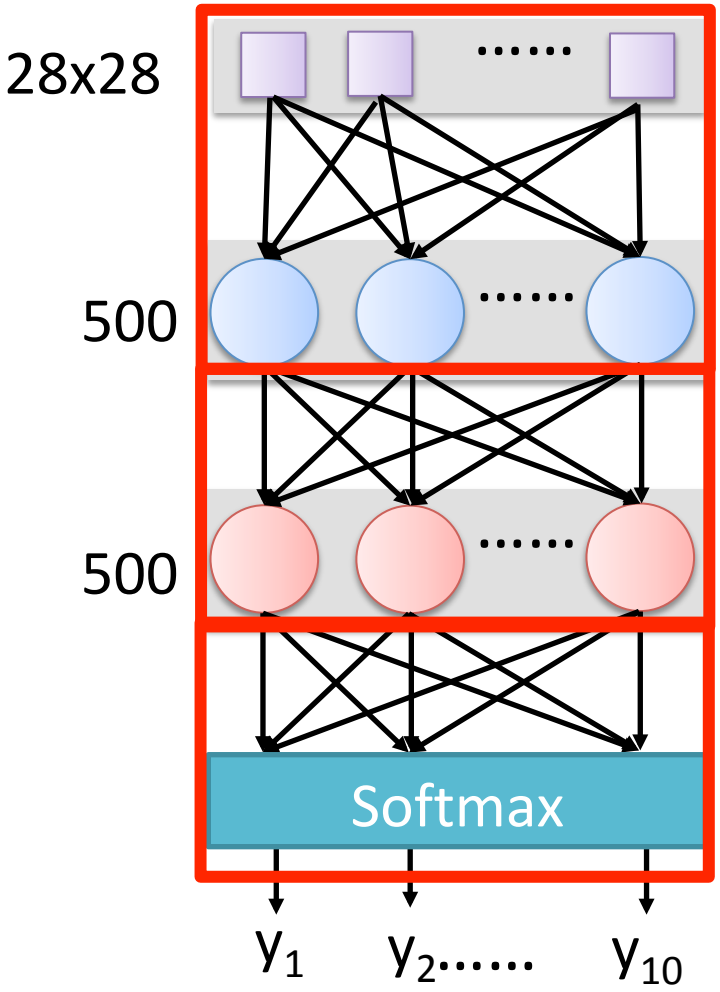
Keras provides data sets loading function: http://keras.io/datasets/

Keras

Step 1: define a set of function

Step 2: goodness of function

Step 3: pick the best function




```
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') )
```

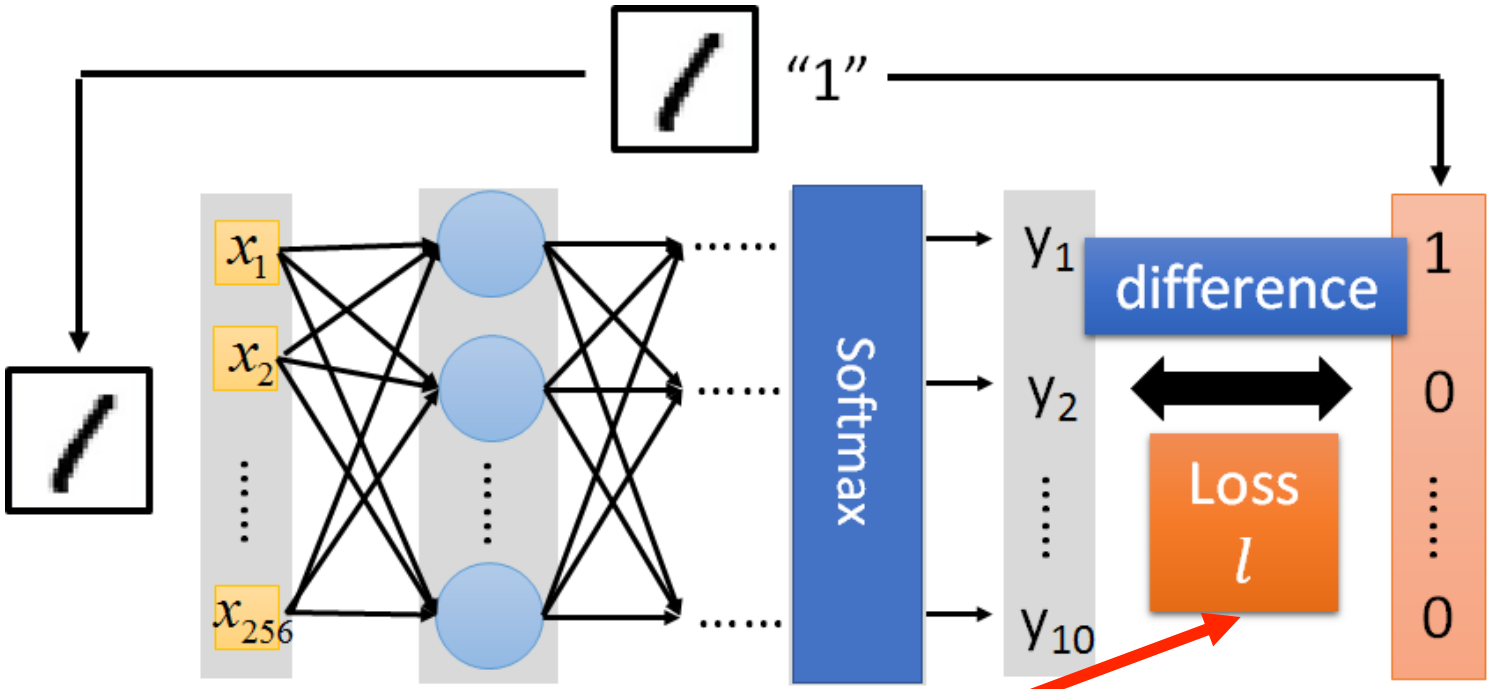
```
model.add( Dense(output_dim=10 ) )
model.add( Activation('softmax') )
```

Keras

Step 1: define a set of function

Step 2: goodness of function

Step 3: pick the best function




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

Keras

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)

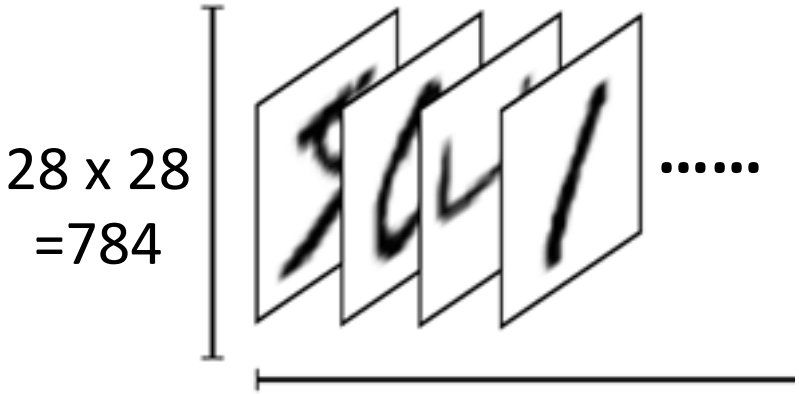
Keras

Step 1: define a set of function → Step 2: goodness of function → Step 3: pick the best function

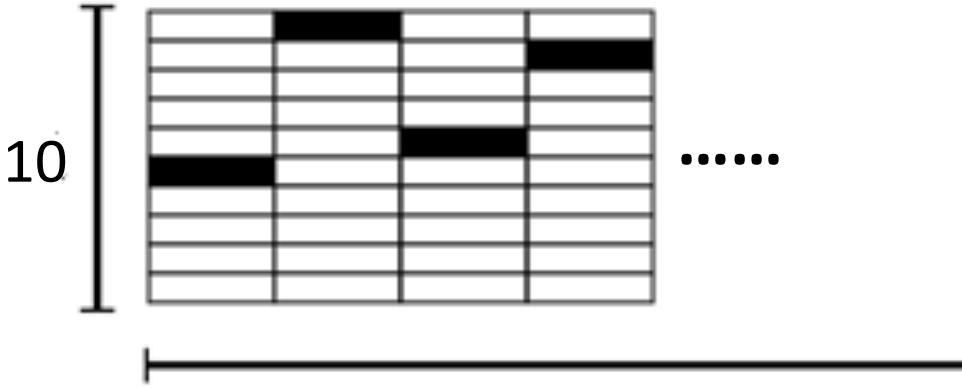
Step 3.2: Find the optimal network parameters

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

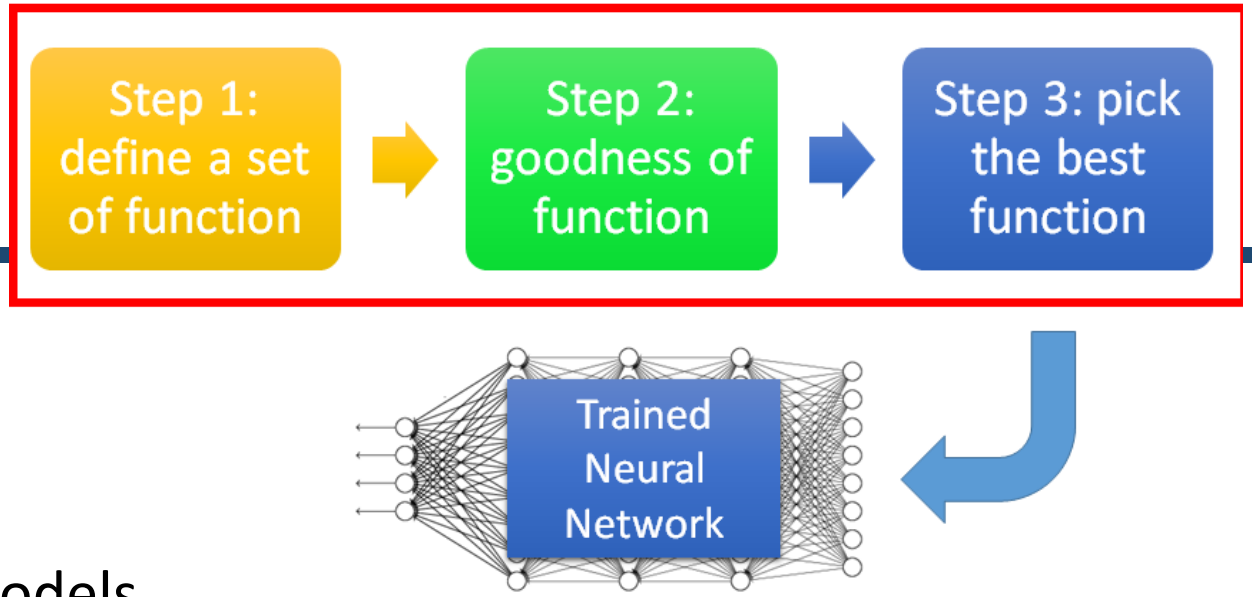
numpy array



numpy array



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



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)
```

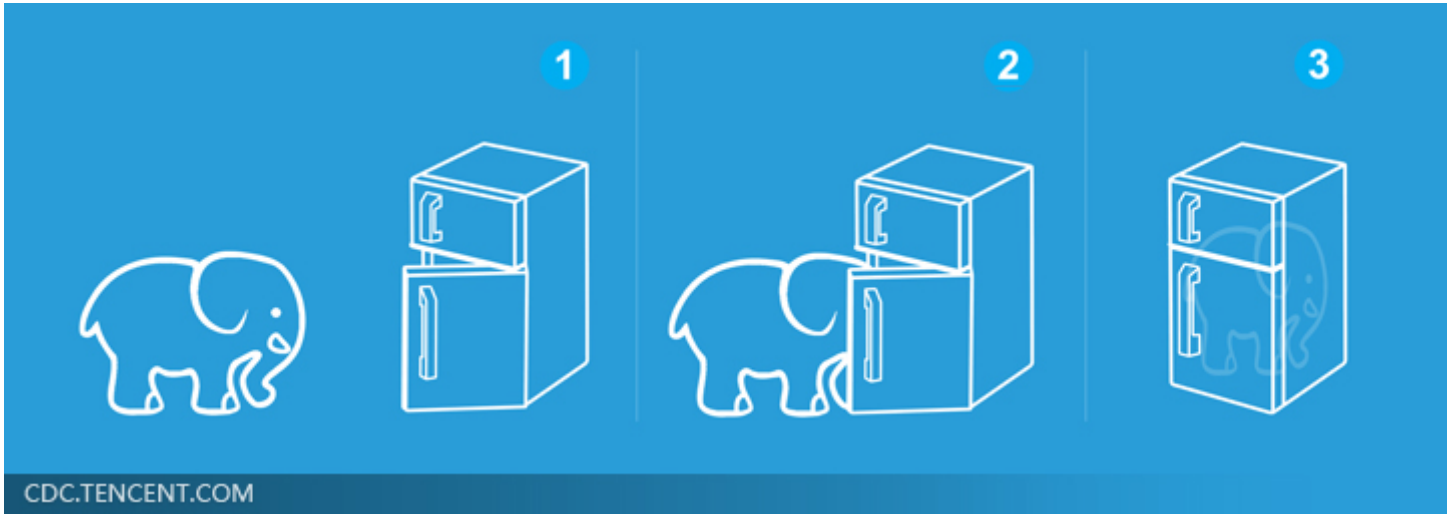
Keras

- Using GPU to speed training
  - Way 1
    - THEANO_FLAGS=device=gpu0 python YourCode.py
  - Way 2 (in your code)
    - import os
    - os.environ["THEANO_FLAGS"] = "device=gpu0"

Three Steps for Deep Learning

Step 1: define a set of function → Step 2: goodness of function → Step 3: pick the best function

Deep Learning is so simple ……



Outline

Introduction of Deep Learning

"Hello World" for Deep Learning

Tips for Deep Learning

Recipe of Deep Learning

YES

Step 1: define a set of function

Step 2: goodness of function

Step 3: pick the best function

NO

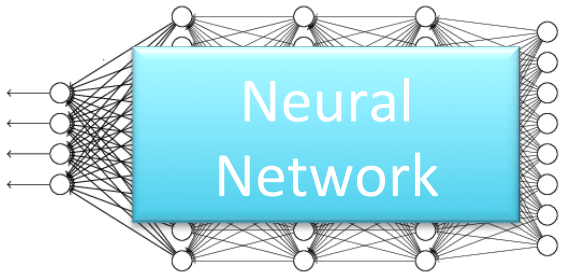
Overfitting!

Good Results on Testing Data?

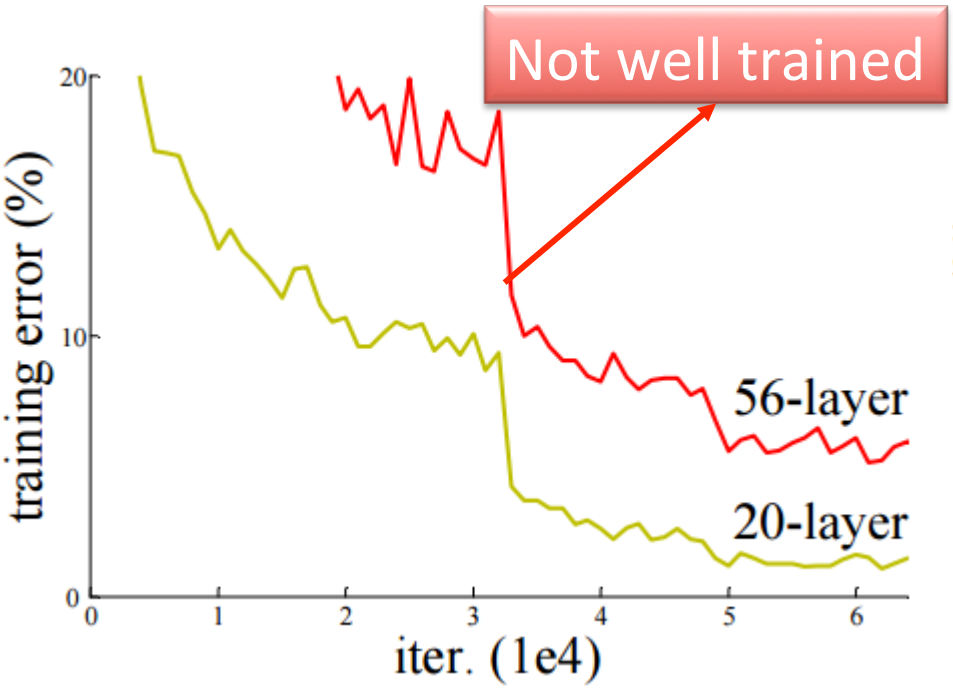
YES

NO

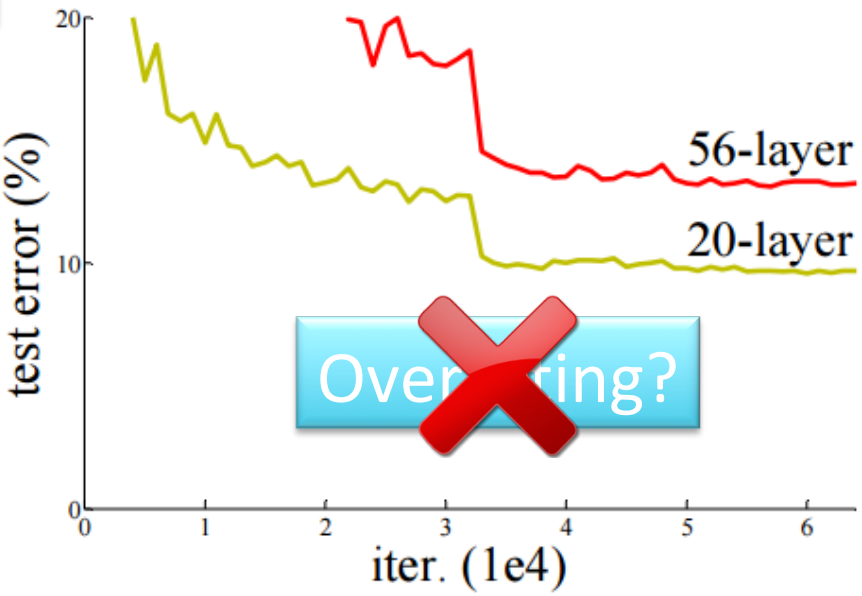
Good Results on Training Data?



Do not always blame Overfitting



Training Data



Testing Data

Deep Residual Learning for Image Recognition
http://arxiv.org/abs/1512.03385

Recipe of Deep Learning

YES

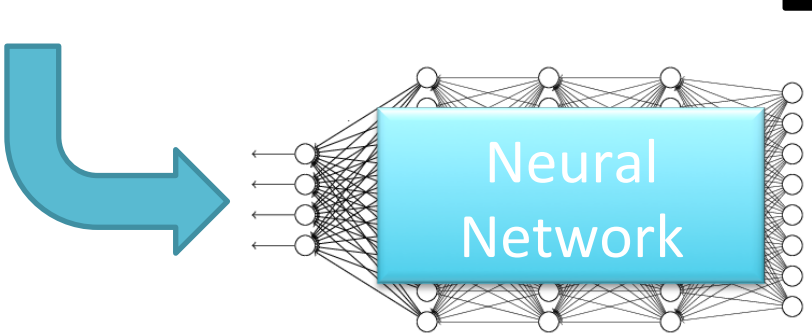
Good Results on Testing Data?

Different approaches for different problems.

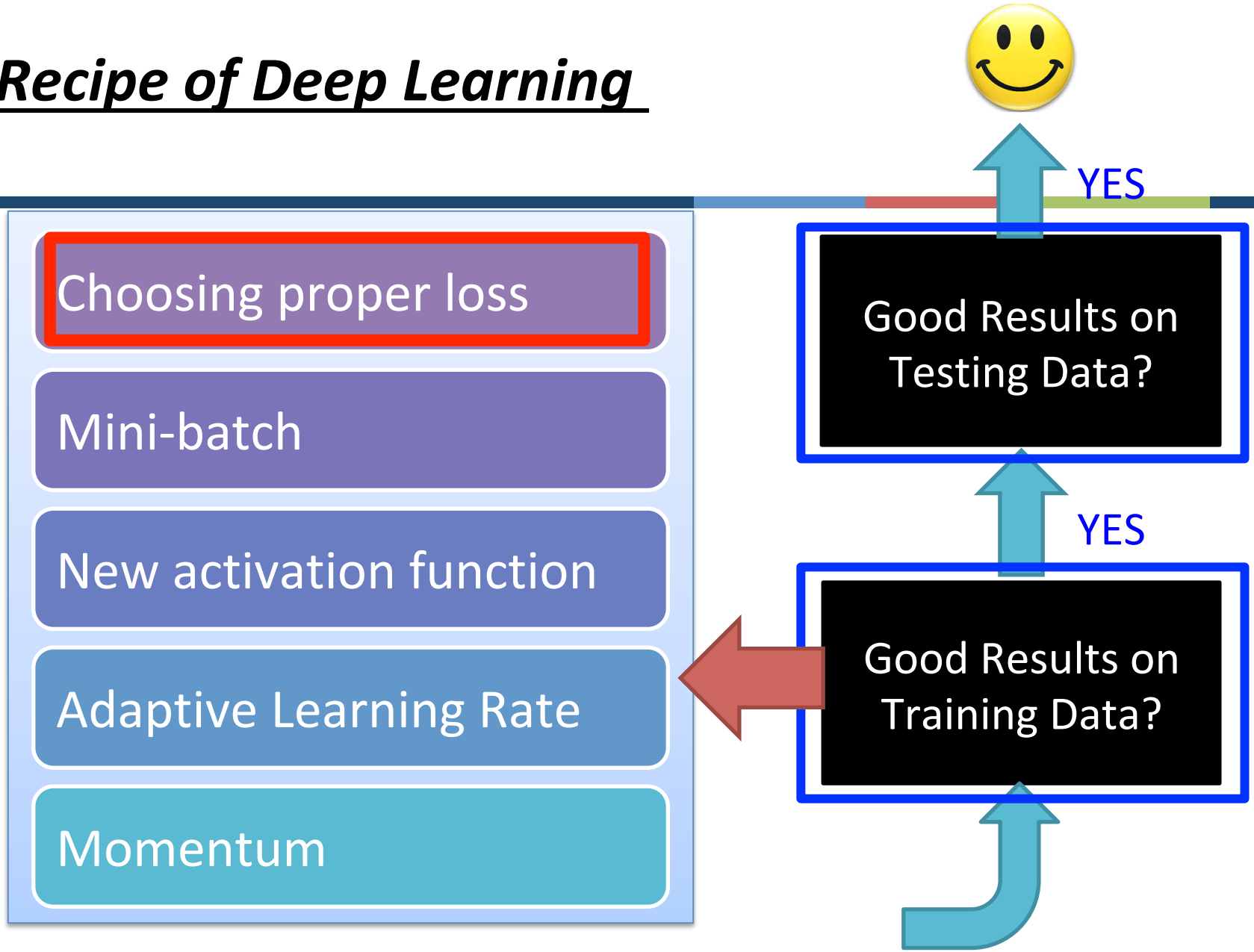
e.g. dropout for good results on testing data

YES

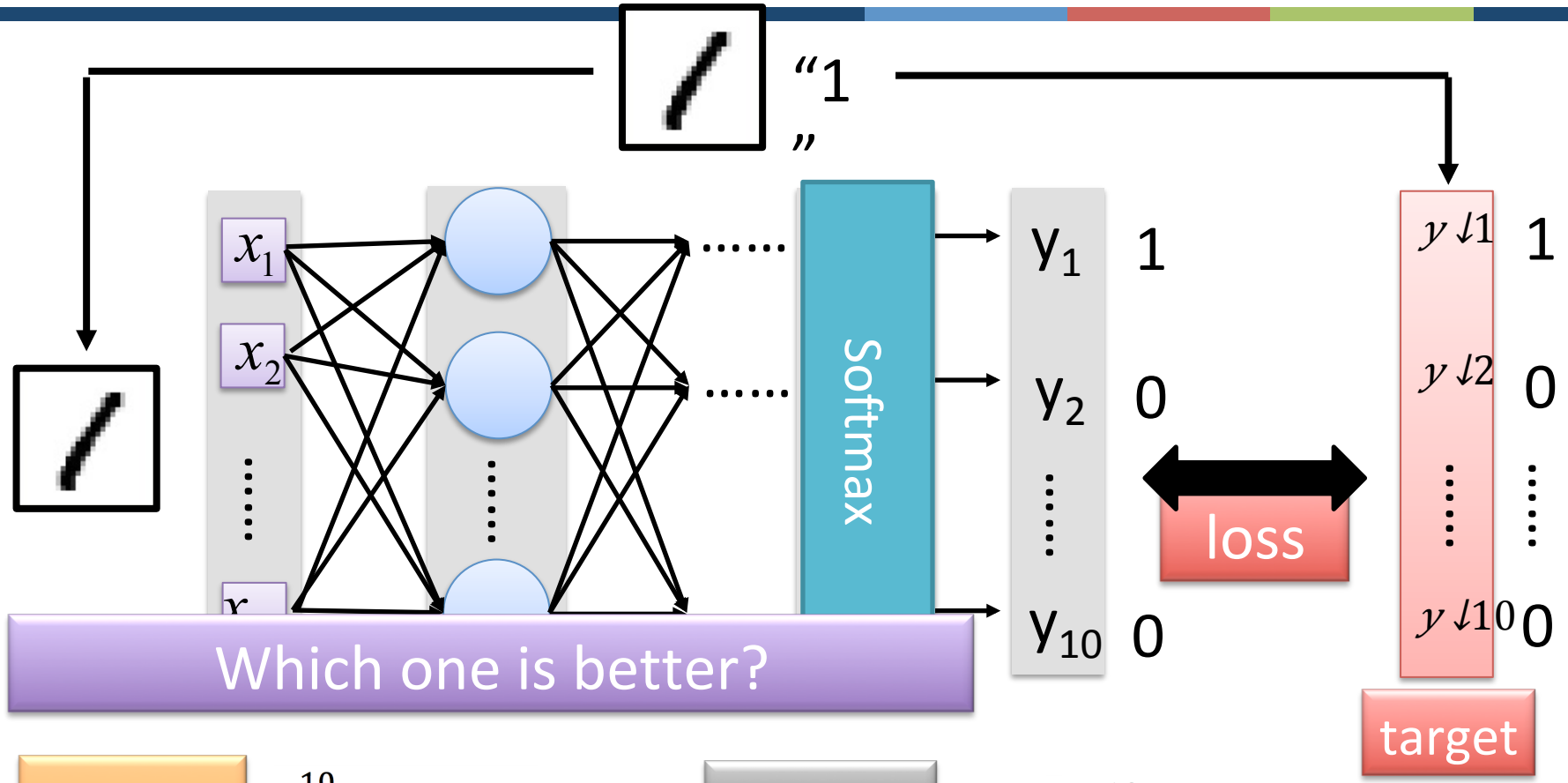
Good Results on Training Data?



Recipe of Deep Learning



Choosing Proper Loss



Square Error

$$\sum_{i=1}^{10}(y_i-\hat{y}_i)^2$$

=0

Cross Entropy

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

=0

Demo

Square Error

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

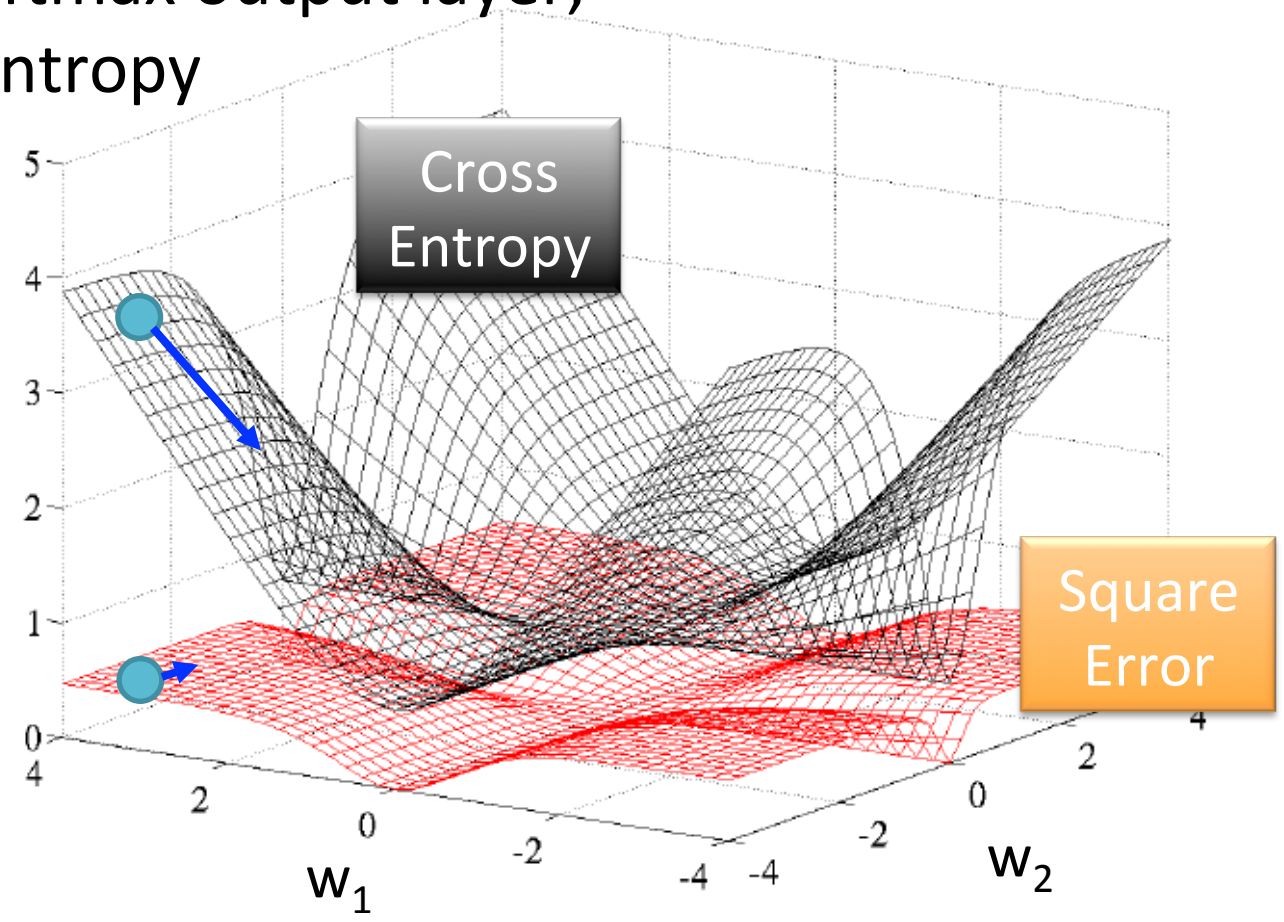
Cross Entropy

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

Several alternatives: https://keras.io/objectives/

Choosing Proper Loss

When using softmax output layer, choose cross entropy



http://jmlr.org/proceedings/papers/v9/glorot10a/glorot10a.pdf

Recipe of Deep Learning

YES

Good Results on Testing Data?

YES

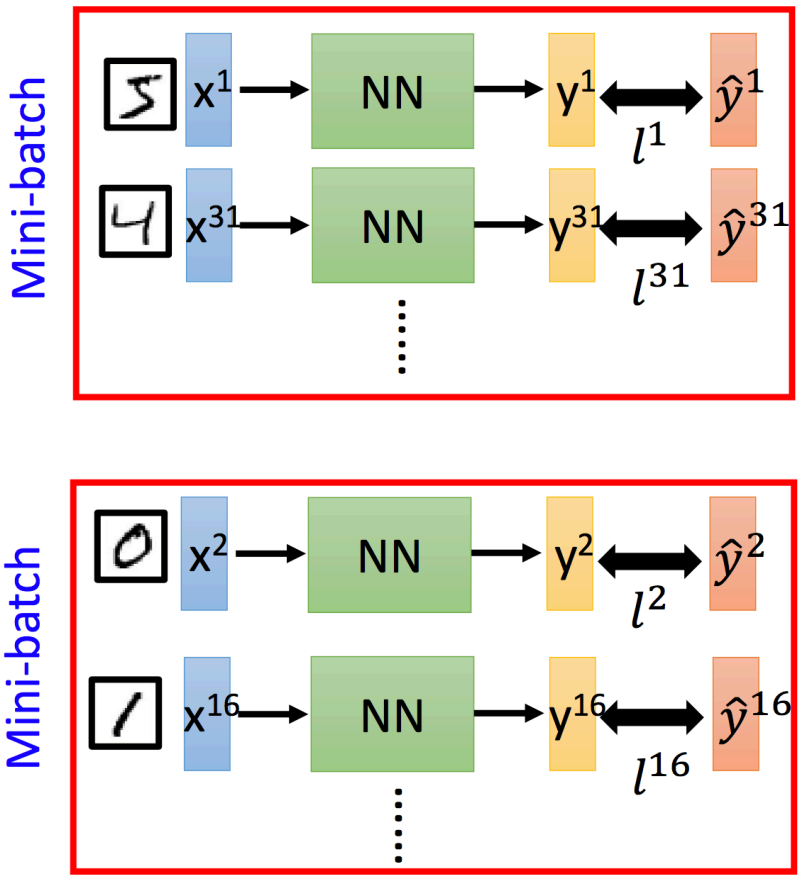
Good Results on Training Data?

- Choosing proper loss
- Mini-batch
- New activation function
- Adaptive Learning Rate
- Momentum

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

We do not really minimize total loss!

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

  ⋮

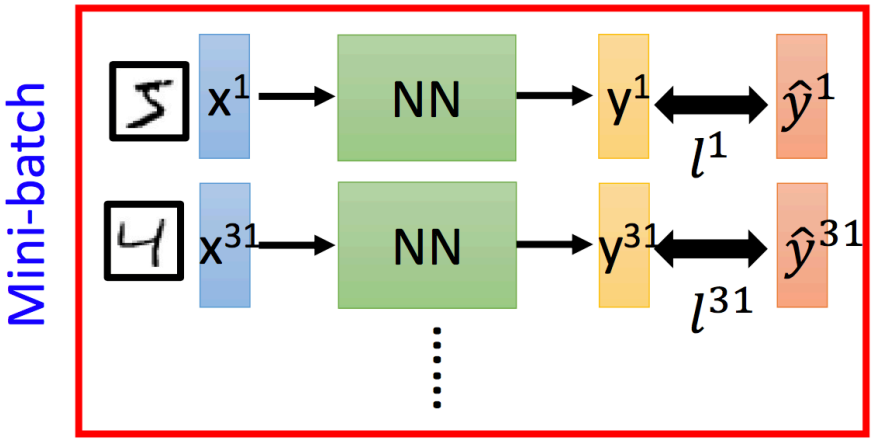
- Until all mini-batches have been picked

one epoch

Repeat the above process

Mini-batch

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



100 examples in a mini-batch

- 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

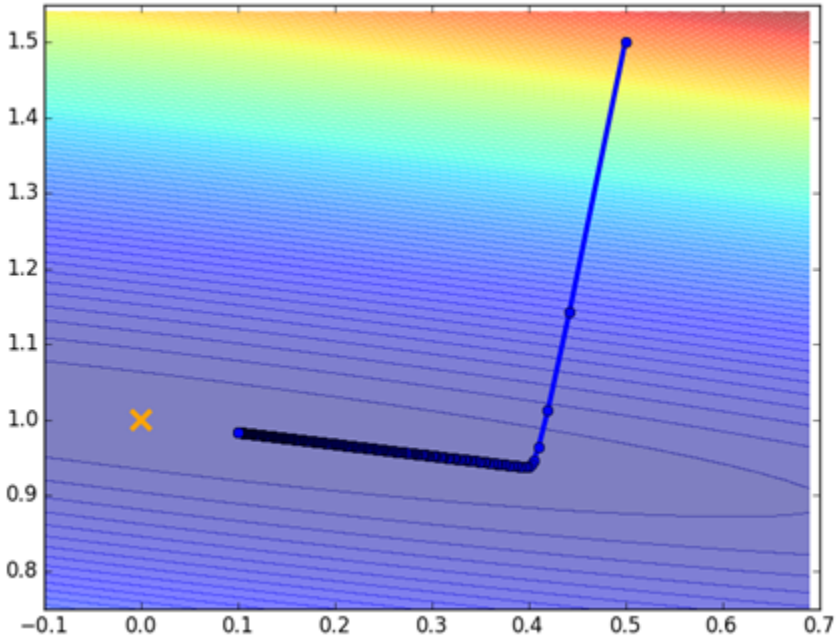
  ⋮

- Until all mini-batches have been picked

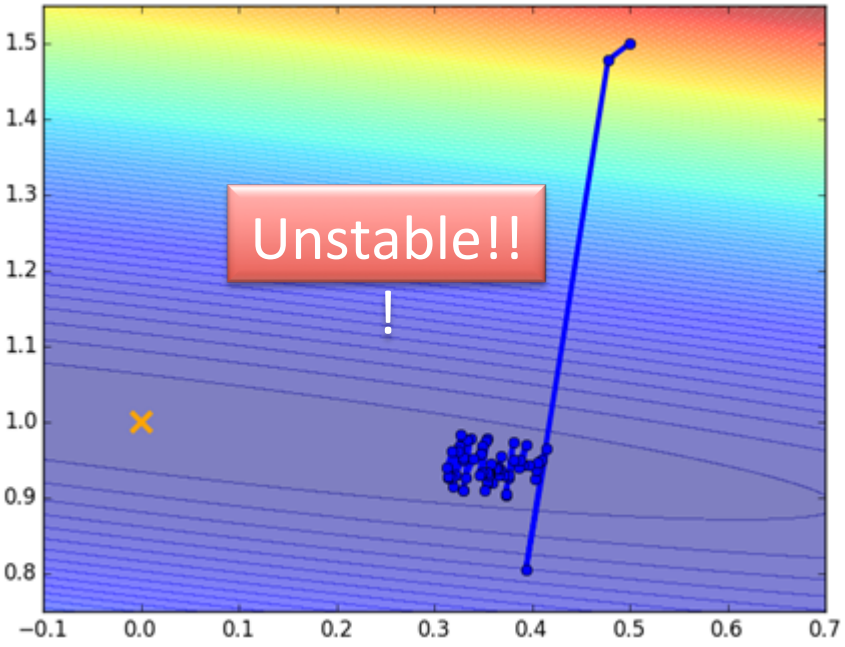
Repeat 20 times

Mini-batch

Original Gradient Descent



With Mini-batch



The colors represent the total loss.

Mini-batch is Faster

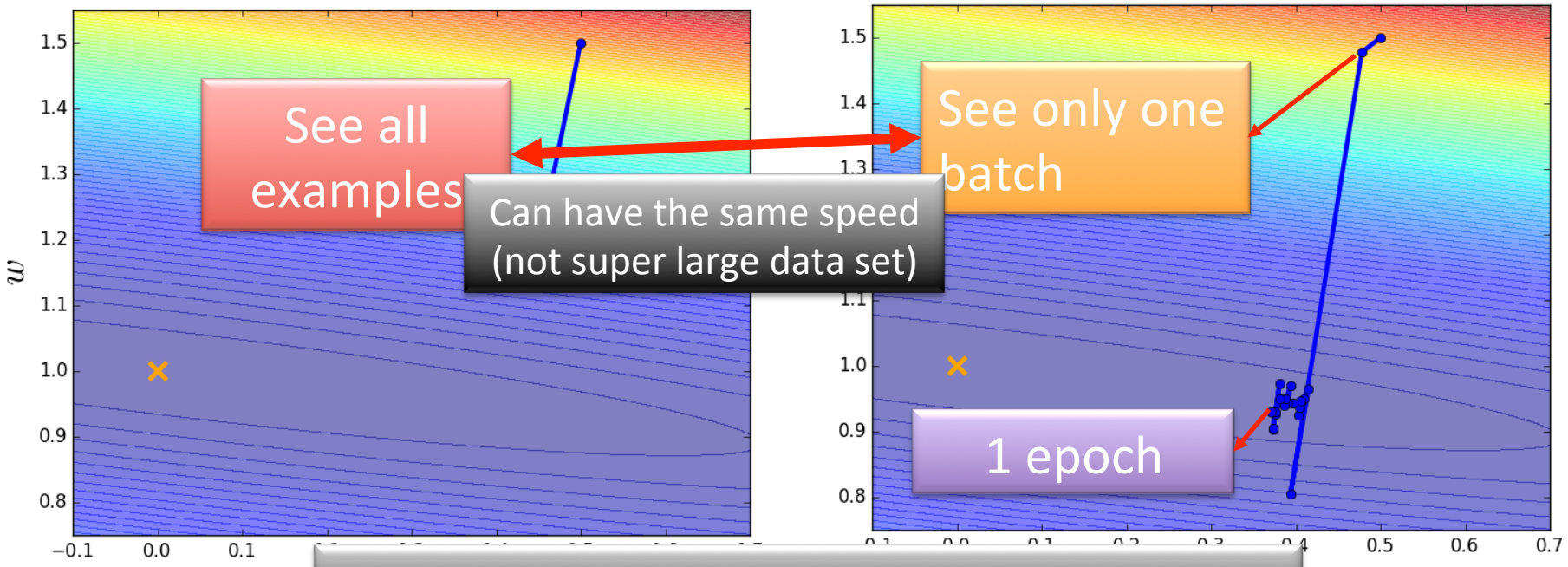
Not always true with parallel computing.

Original Gradient Descent

Update after seeing all examples

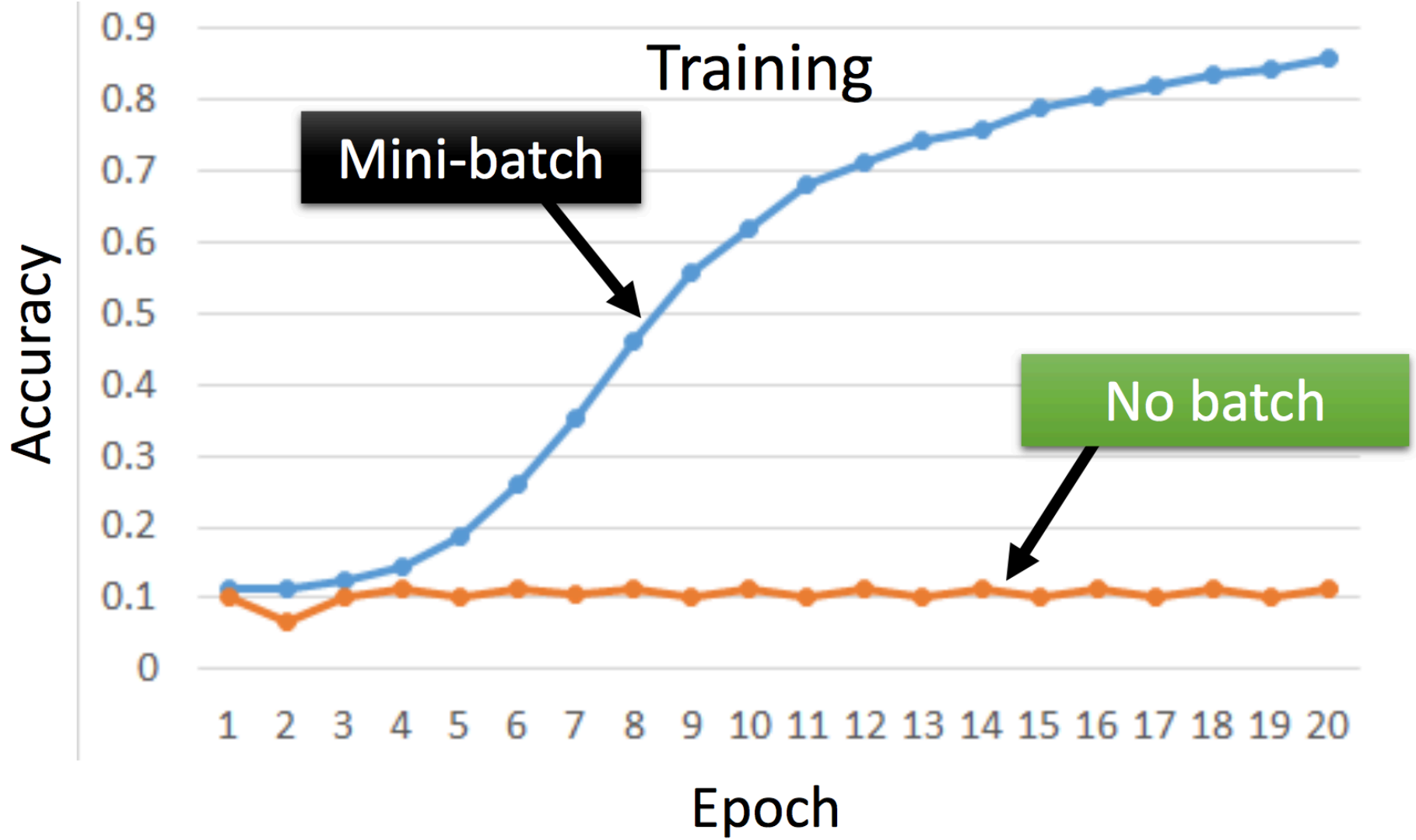
With Mini-batch

If there are 20 batches, update 20 times in one epoch.

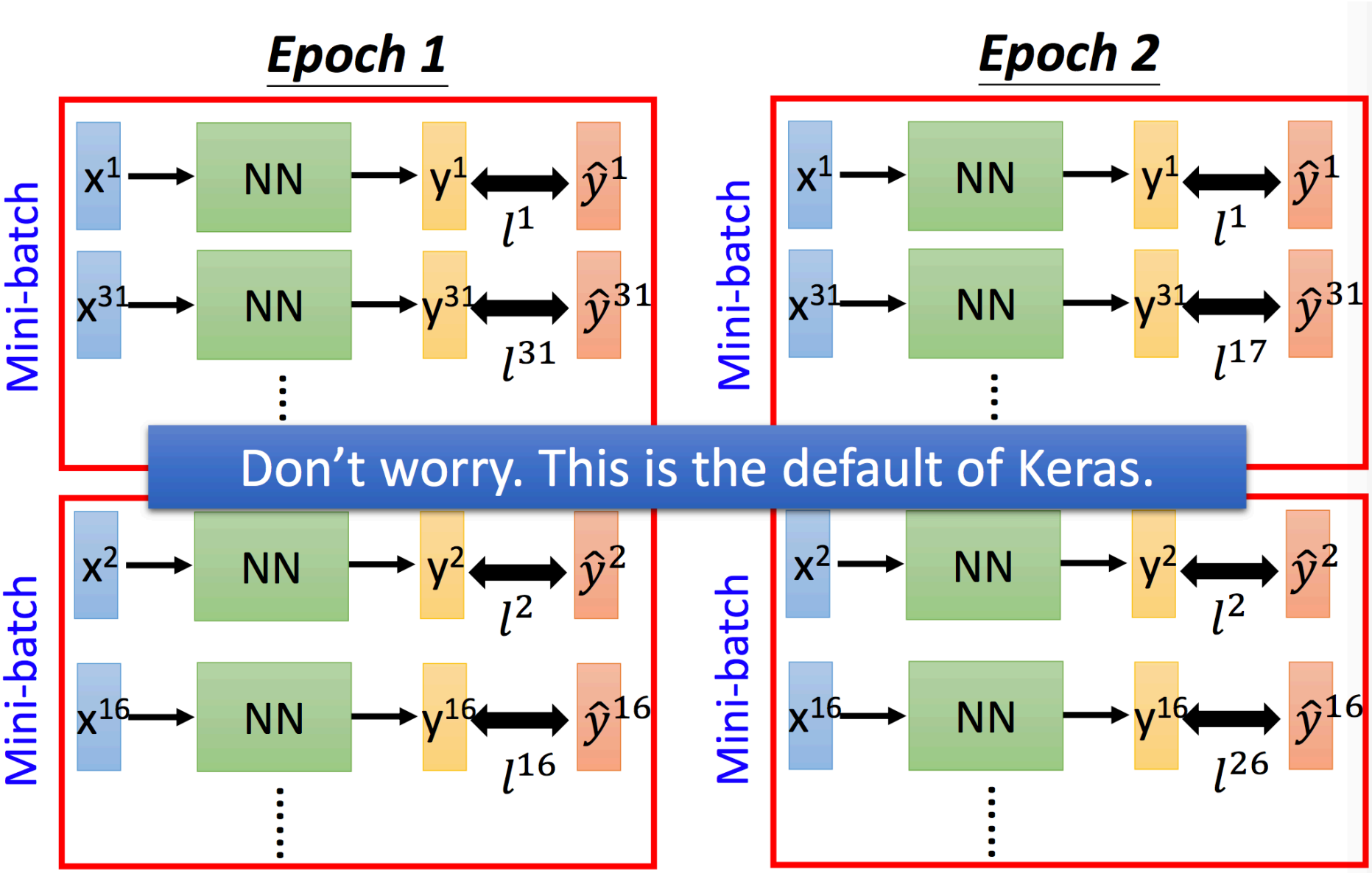


Mini-batch has better performance!

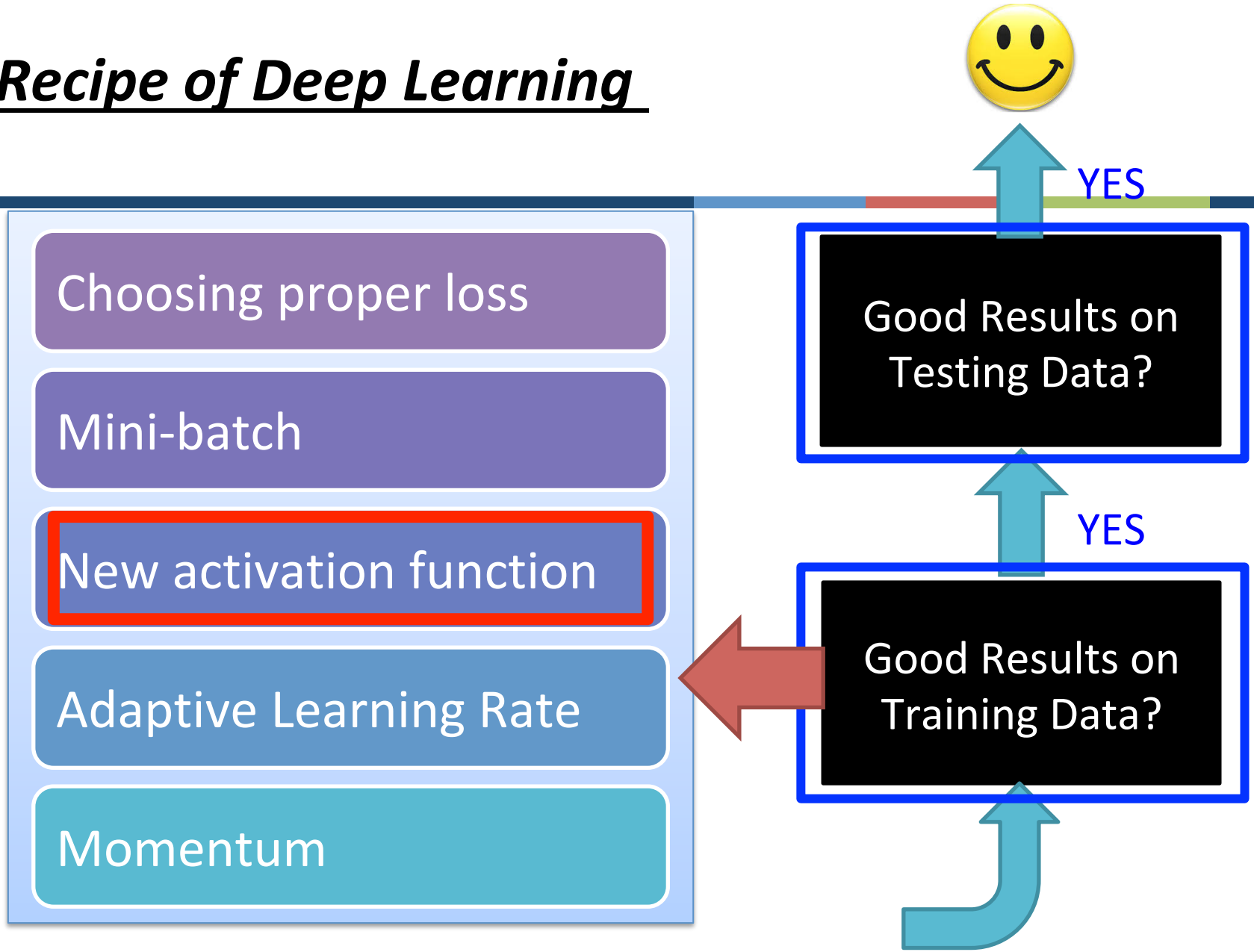
Mini-batch is Faster



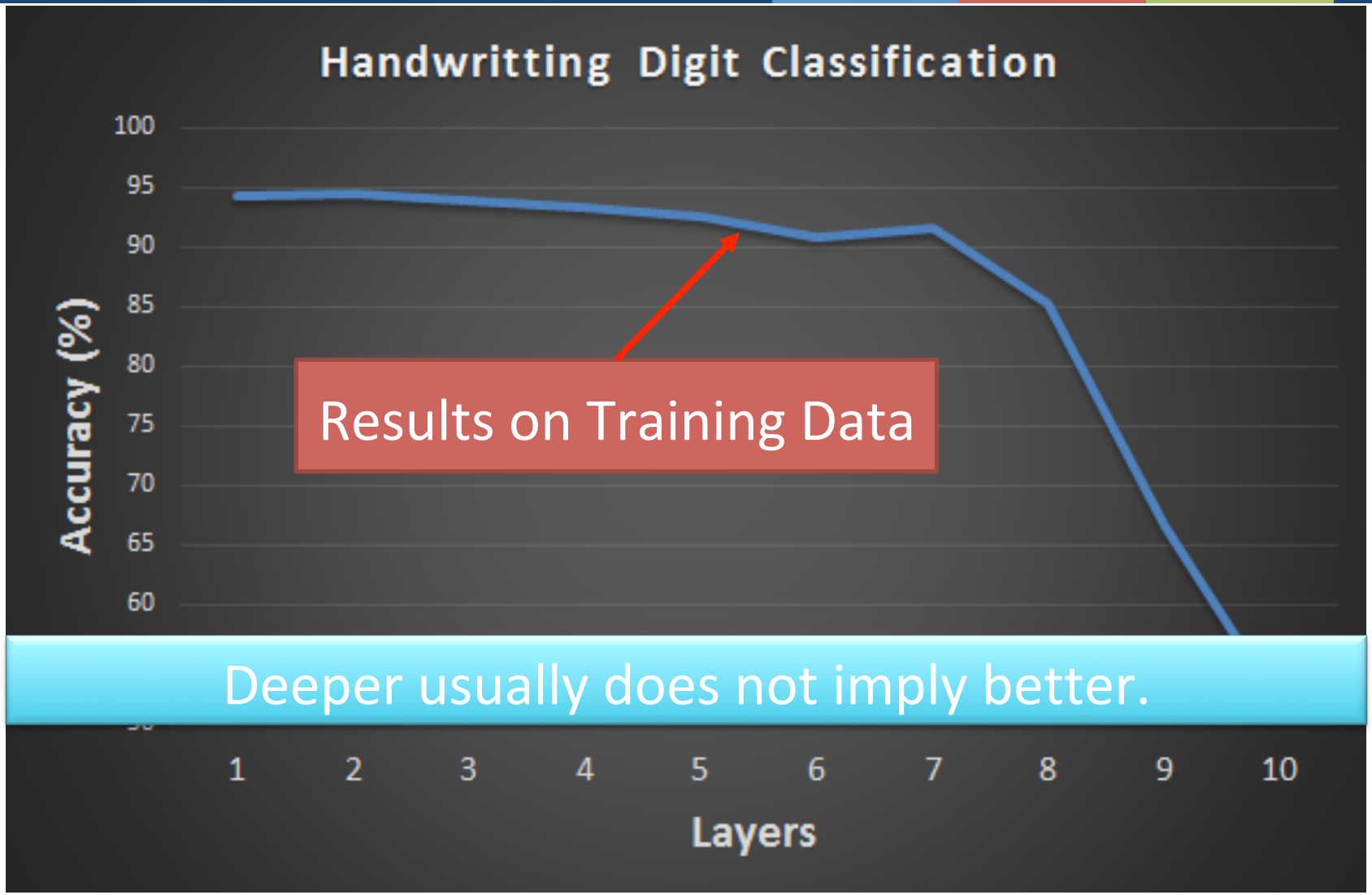
Shuffle the training examples for each epoch



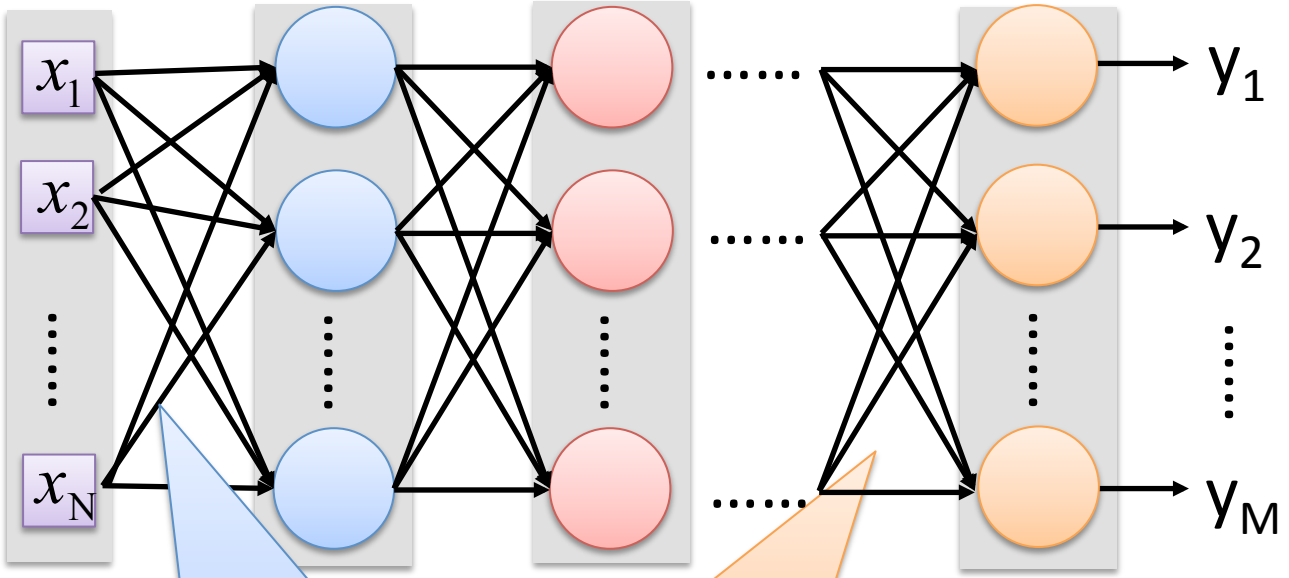
Recipe of Deep Learning



Hard to get the power of Deep …



Vanishing Gradient Problem



Smaller gradients

Learn very slow

Almost random

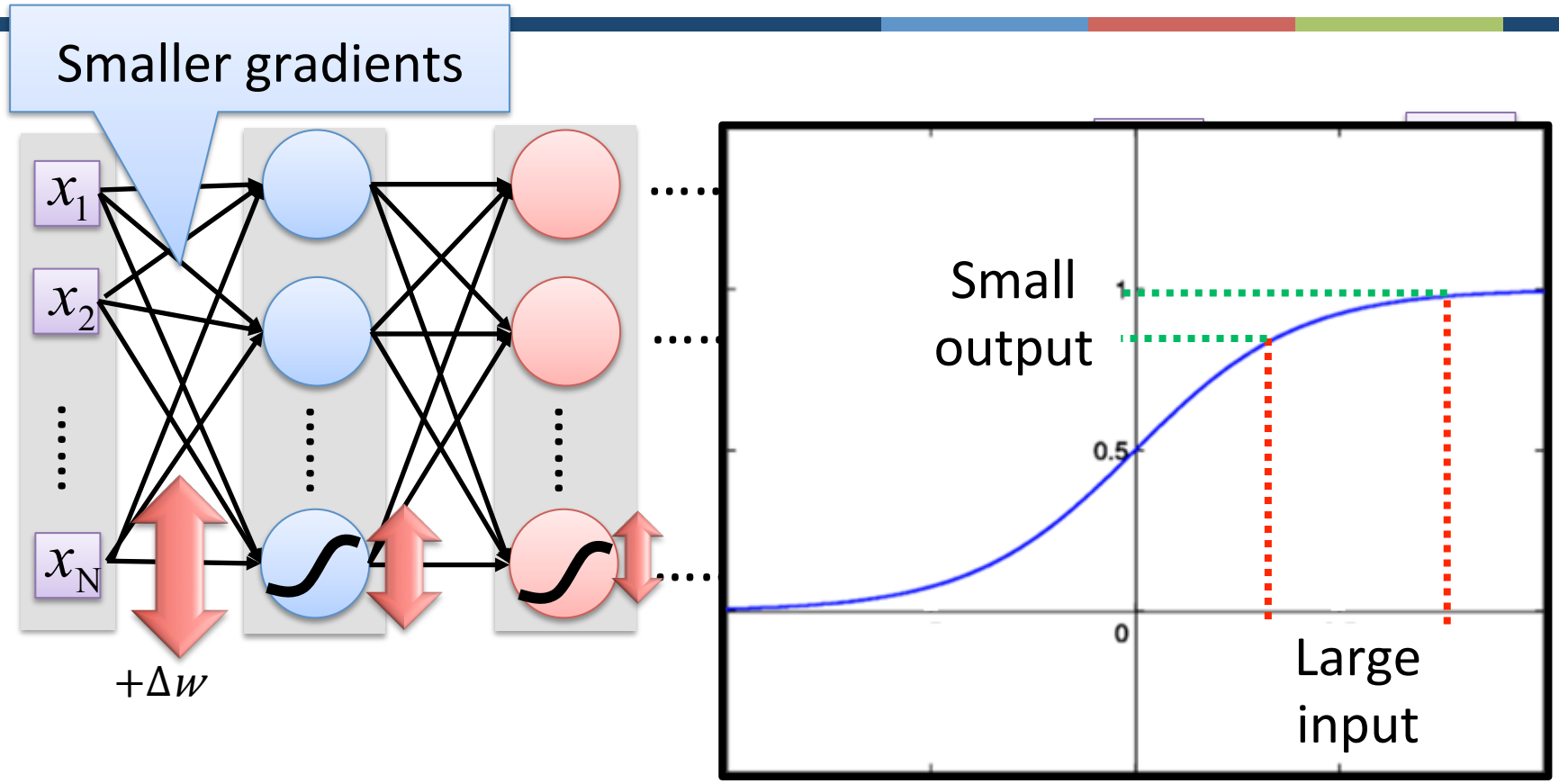
Larger gradients

Learn very fast

Already converge

based on random!?

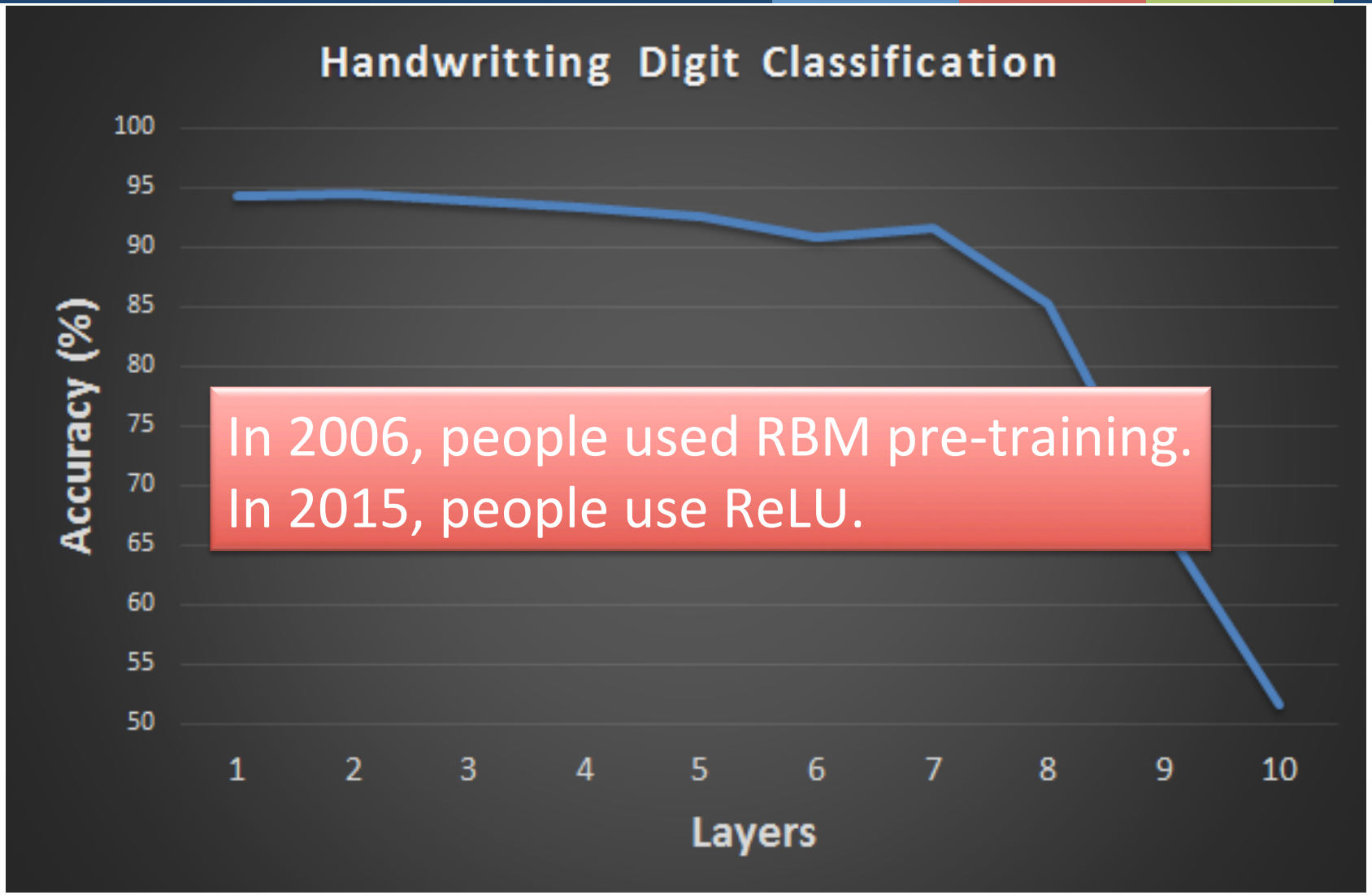
Vanishing Gradient Problem



Intuitive way to compute the derivatives …

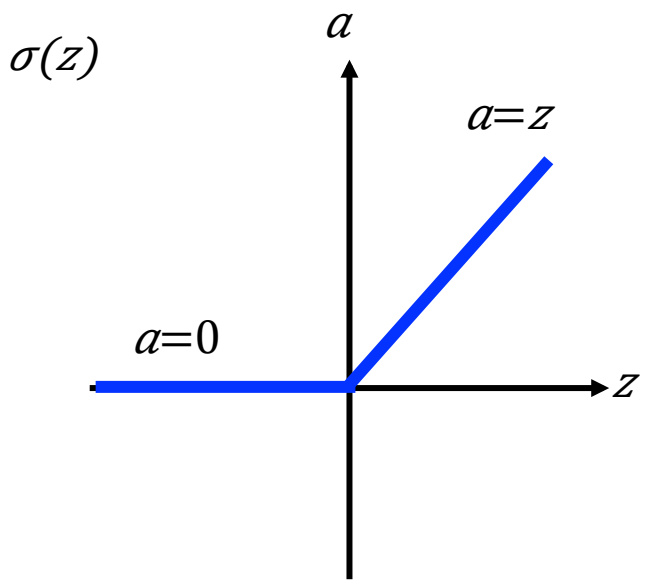
$\partial l/\partial w = ?\ \Delta l/\Delta w$

Hard to get the power of Deep …



ReLU

- Rectified Linear Unit (ReLU)

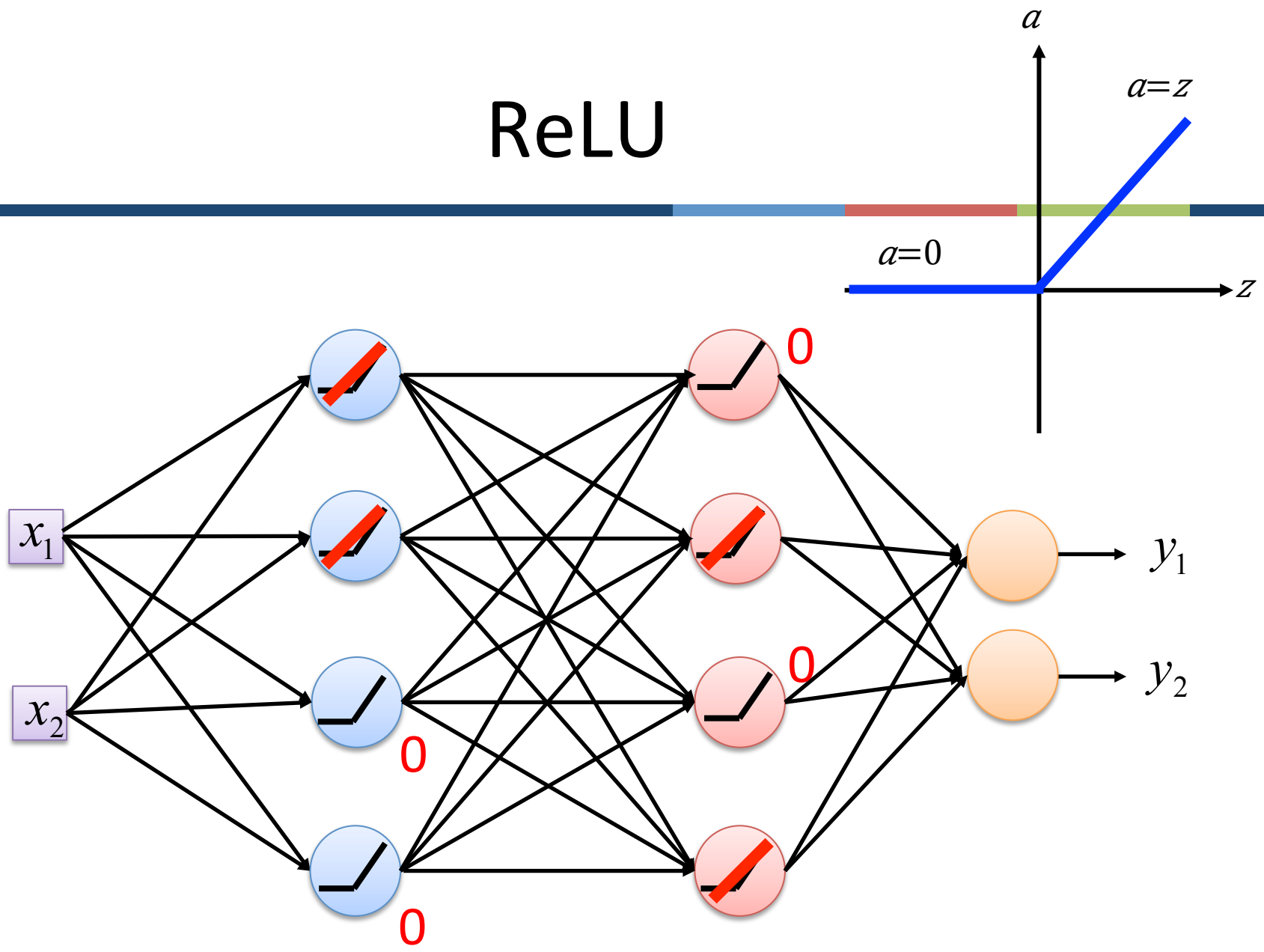


[Xavier Glorot, AISTATS'11]
[Andrew L. Maas, ICML'13]
[Kaiming He, arXiv'15]

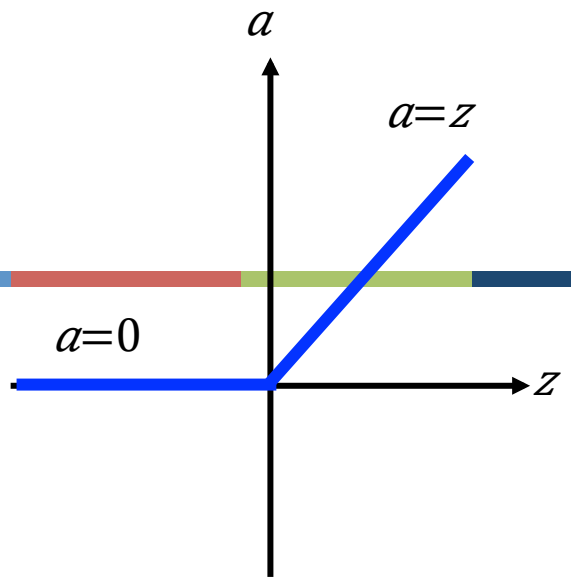
Reason:

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

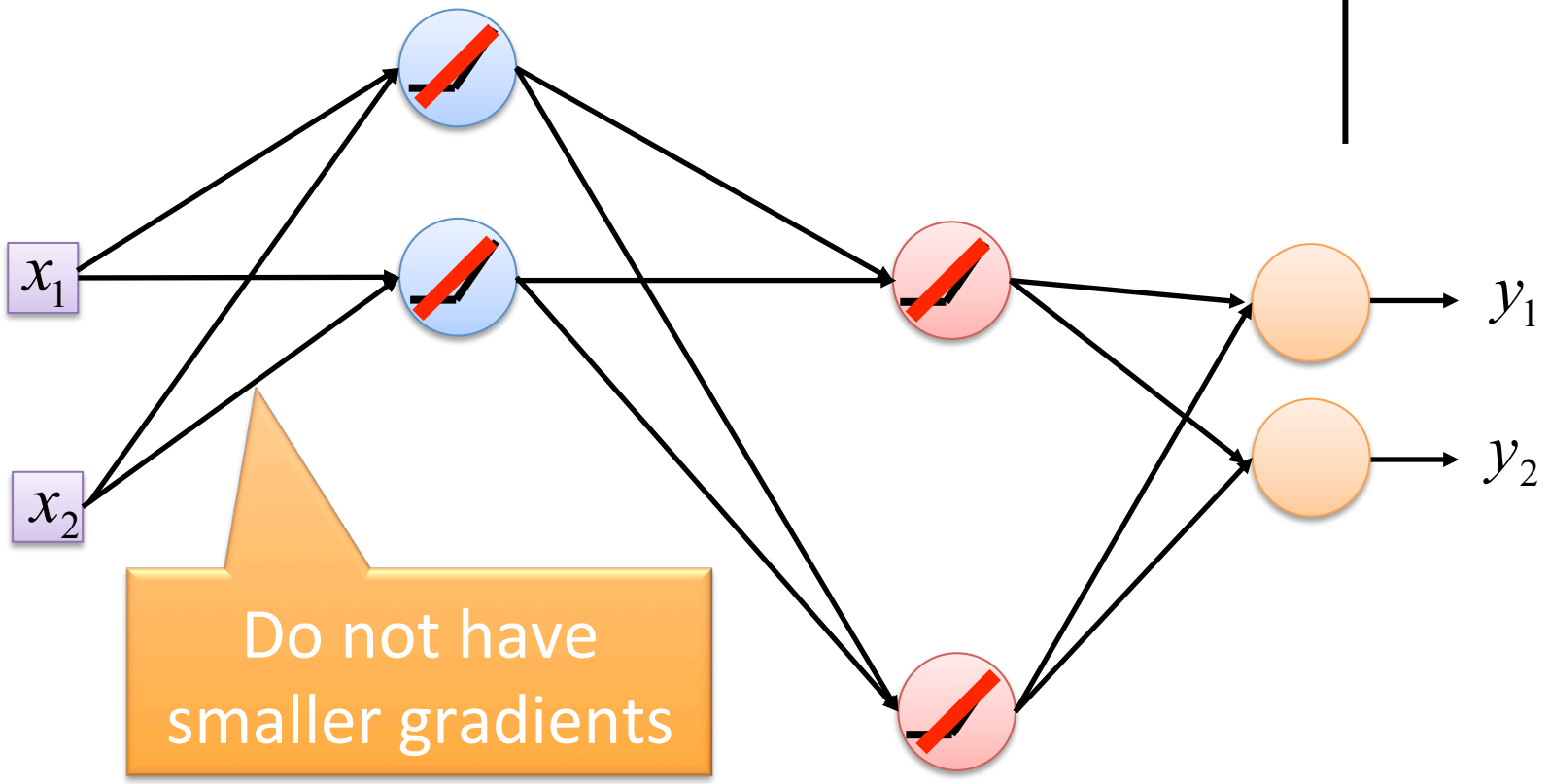
ReLU



ReLU



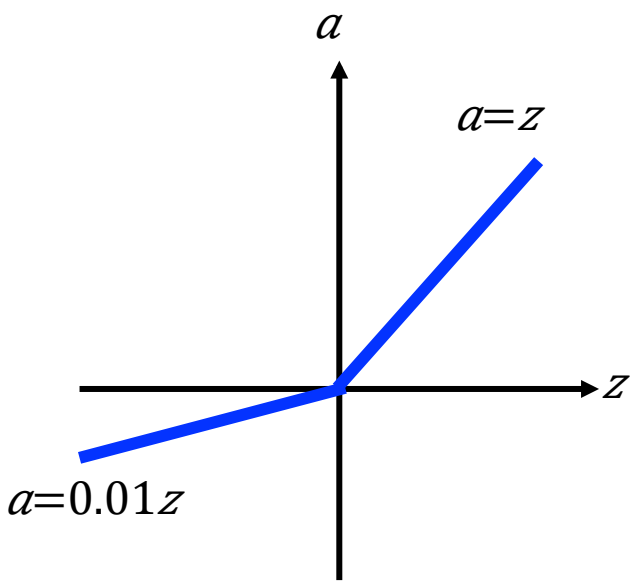
A Thinner linear network



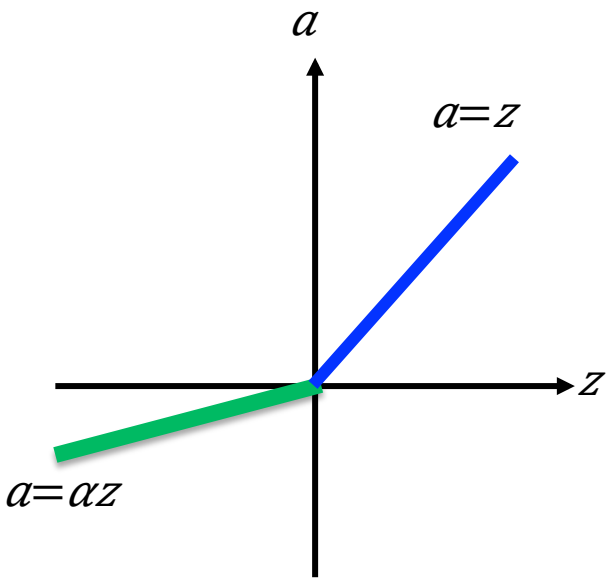
Do not have smaller gradients

ReLU - variant

Leaky ReLU



Parametric ReLU

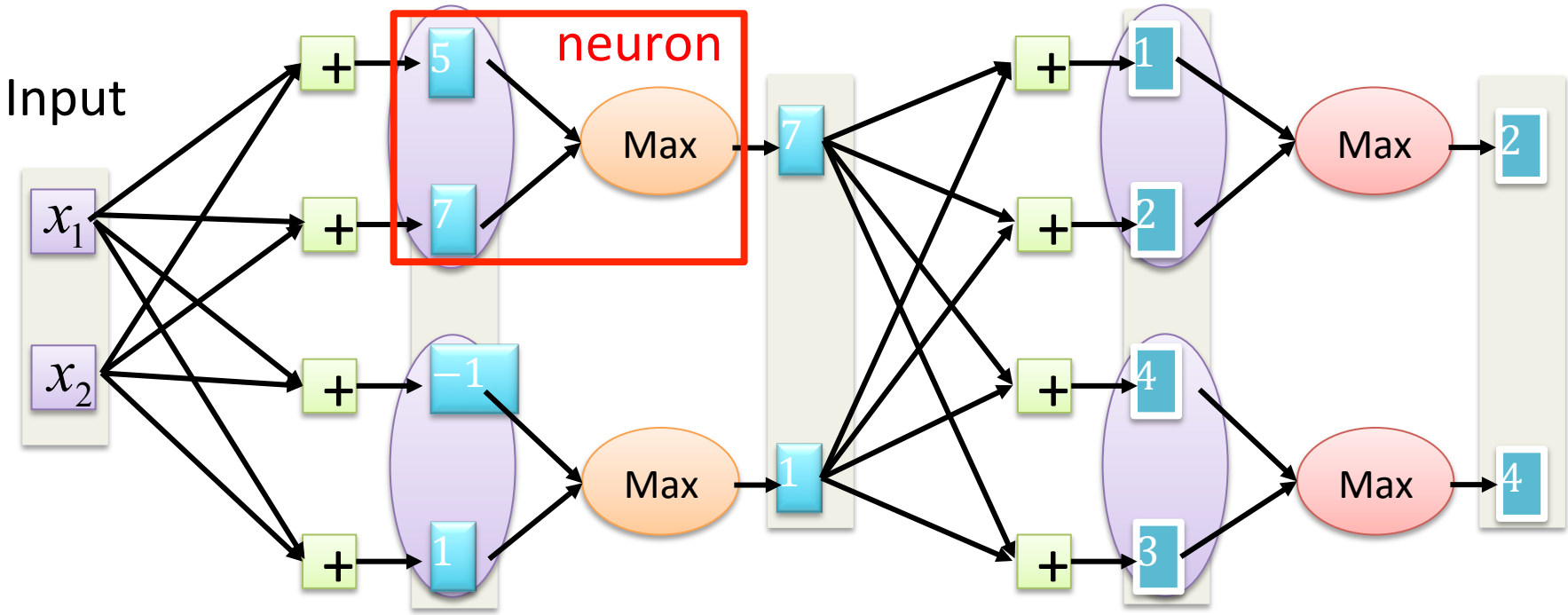


α also learned by gradient descent

Maxout

ReLU is a special cases of Maxout

- Learnable activation function [Ian J. Goodfellow, ICML'13]



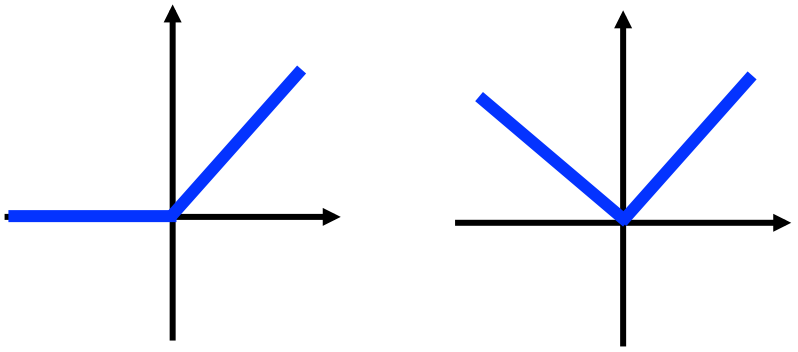
You can have more than 2 elements in a group.

Maxout

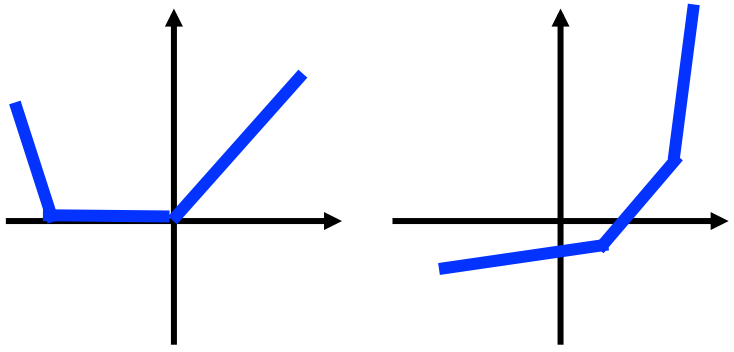
ReLU is a special cases of Maxout

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

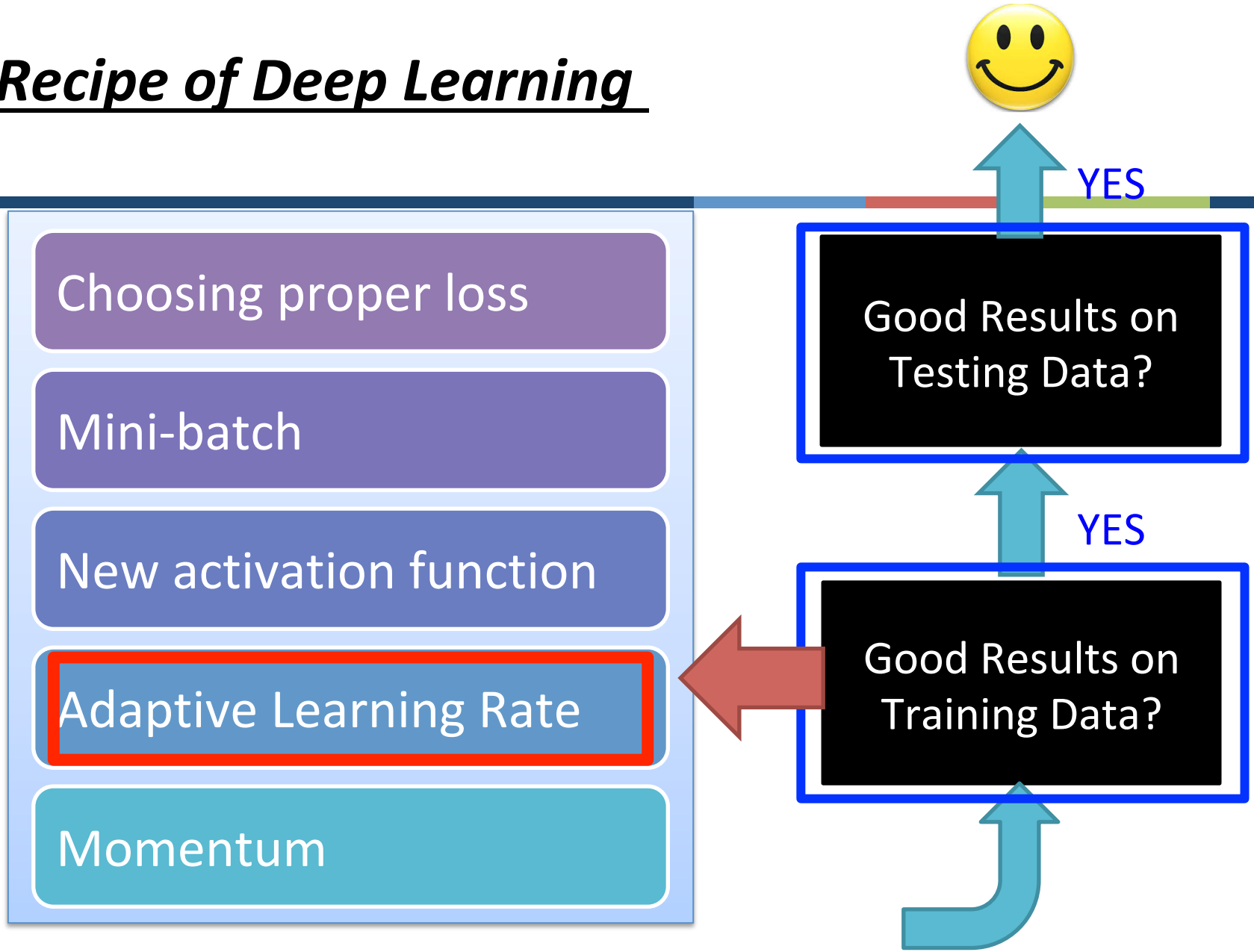
2 elements in a group



3 elements in a group

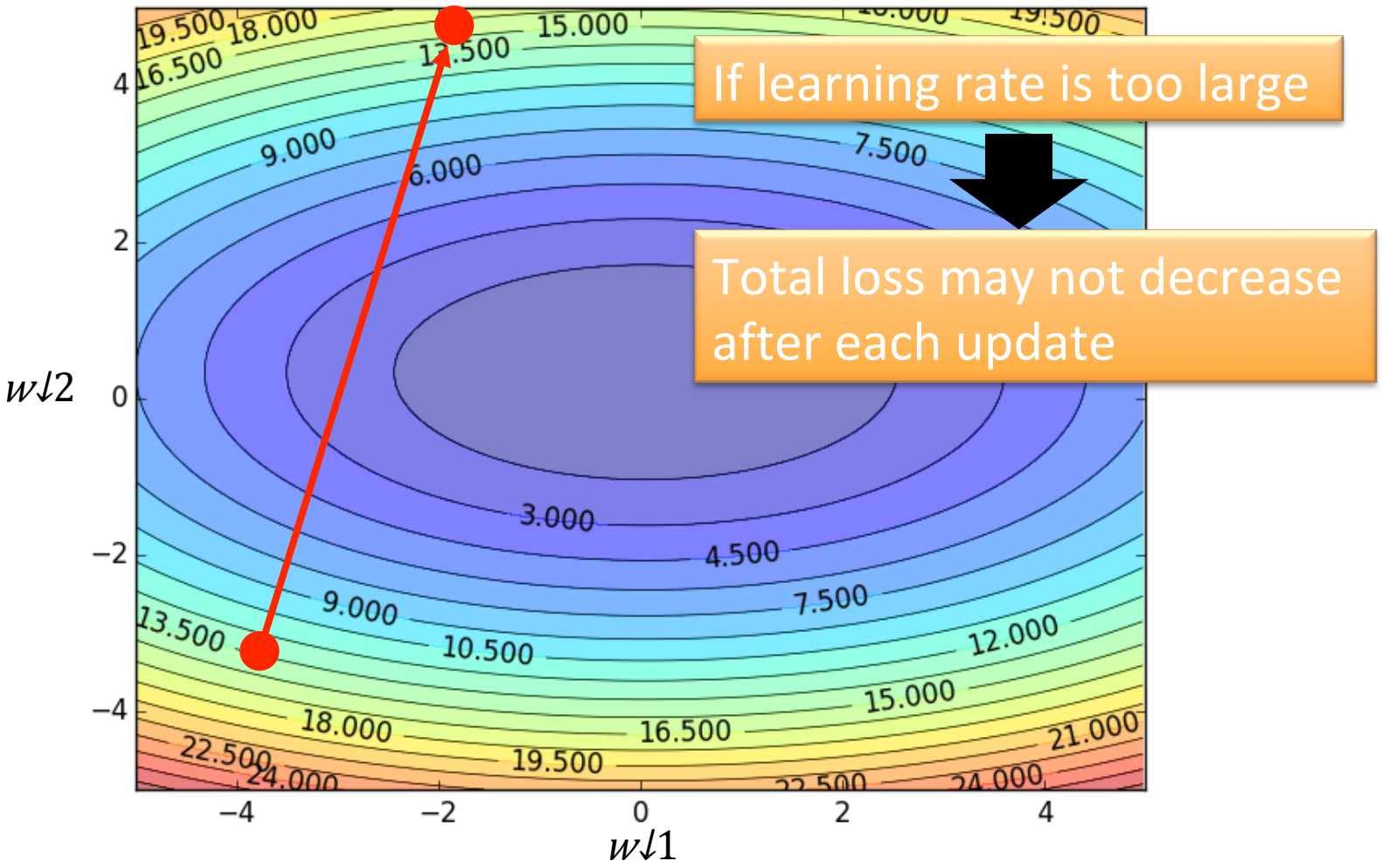


Recipe of Deep Learning



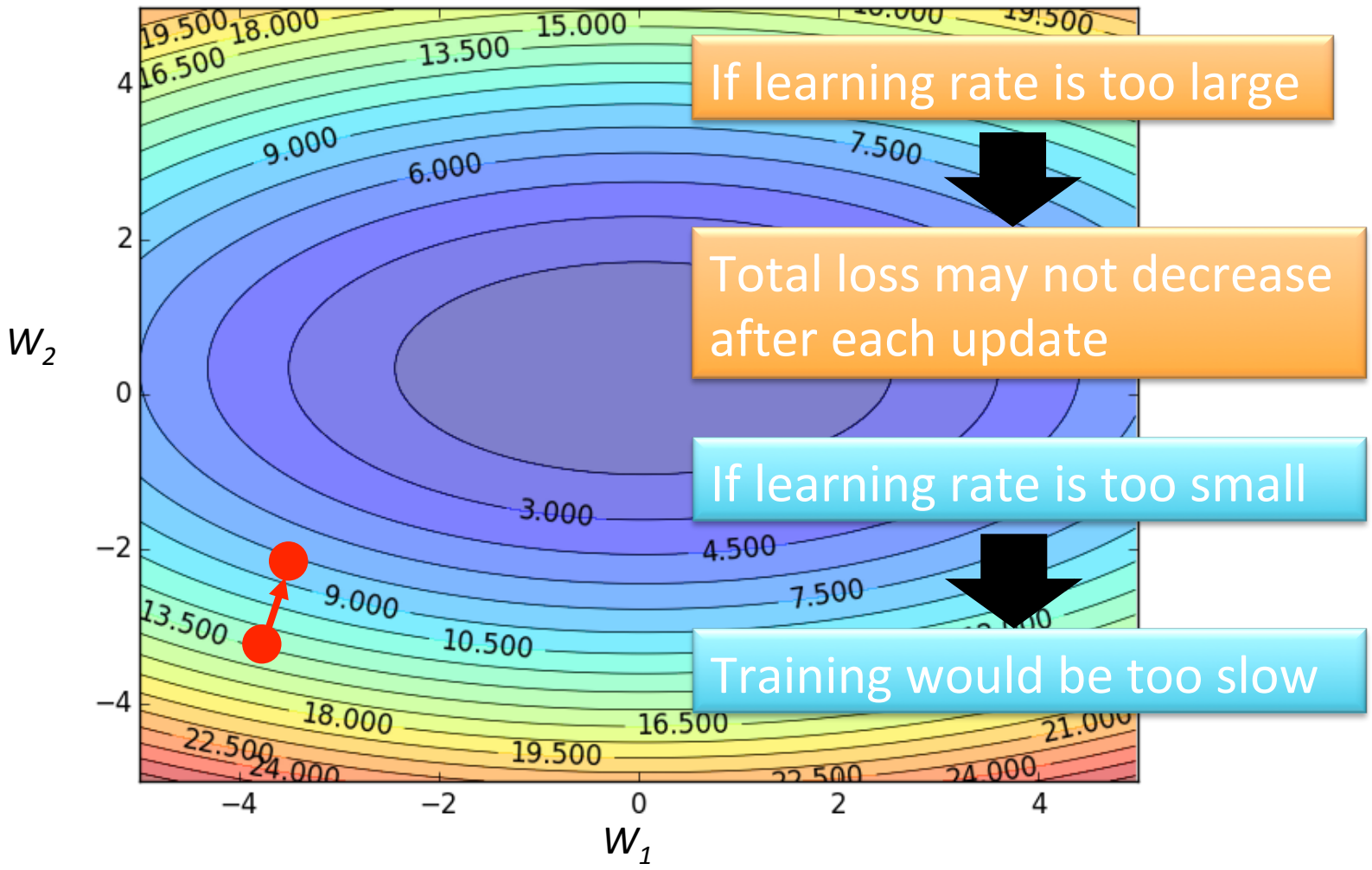
Learning Rates

Set the learning rate η carefully



Learning Rates

Set the learning rate η carefully



If learning rate is too large

Total loss may not decrease after each update

If learning rate is too small

Training would be too slow

Learning Rates

- Popular & Simple Idea: Reduce the learning rate by some factor every few epochs.
  - At the beginning, we are far from the destination, so we use larger learning rate
  - After several epochs, we are close to the destination, so we reduce the learning rate
  - E.g. 1/t decay: $\eta^t = \eta/\sqrt{t+1}$
- Learning rate cannot be one-size-fits-all
  - Giving different parameters different learning rates

Adagrad

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

Adagrad: $w \leftarrow w - \eta_w \partial L / \partial w$

Parameter dependent learning rate

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

η: constant

g^i is $\partial L / \partial w$ obtained at the i-th update

Summation of the square of the previous derivatives

Adagrad

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

	g^0
w_1	0.1

	g^0
w_2	20.0

Learning rate:

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

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

Learning rate:

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

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

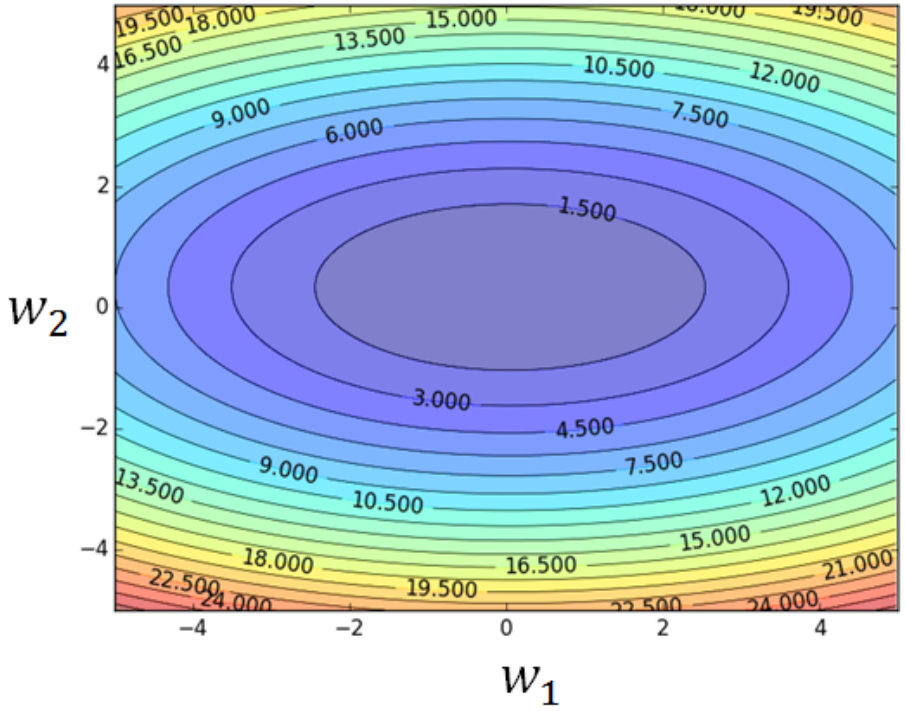
Observation: 1. Learning rate is smaller and smaller for all parameters

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

Why?

Larger derivatives

Smaller Learning Rate



Smaller Derivatives

Larger Learning Rate

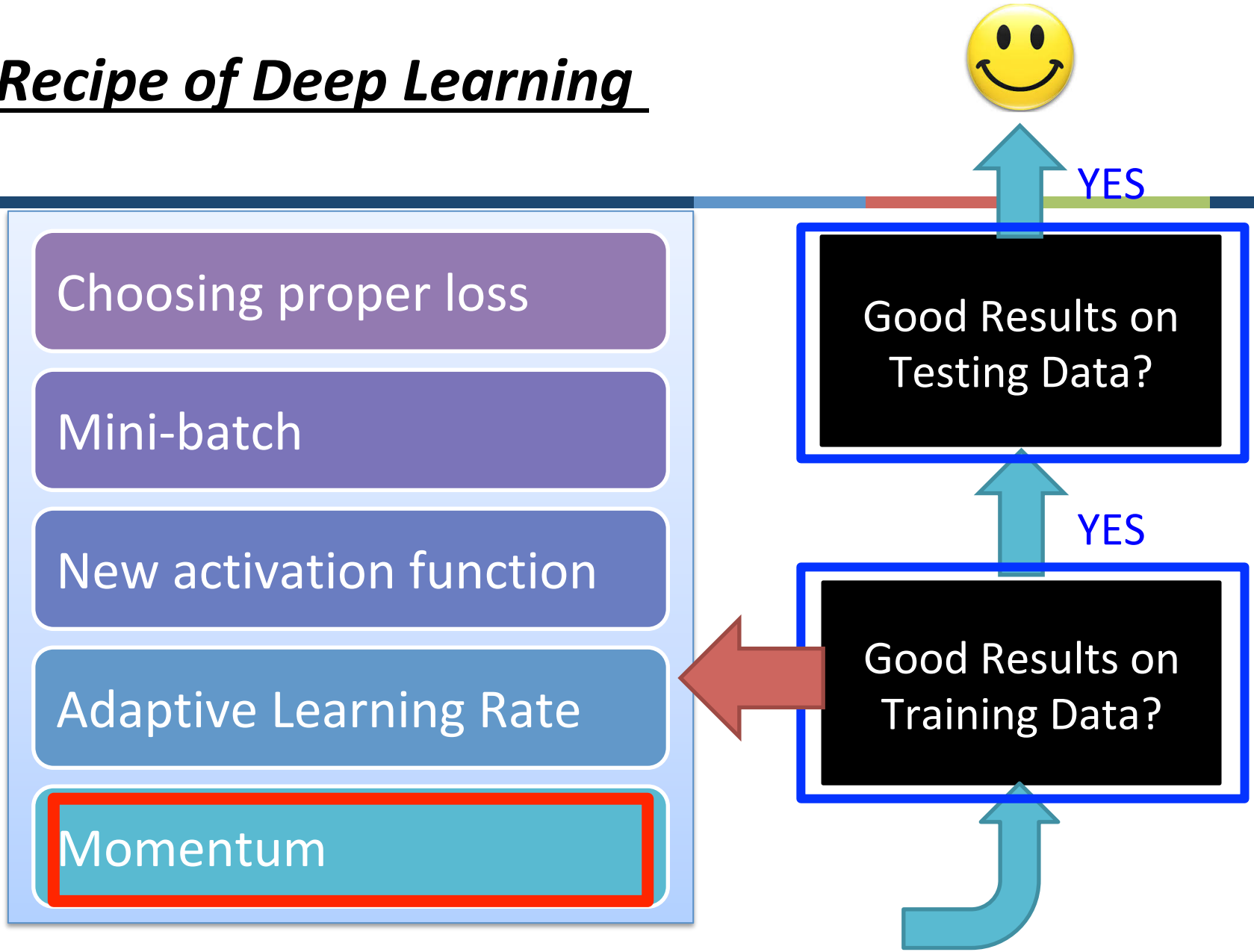
2. Smaller derivatives, larger learning rate, and vice versa

Why?

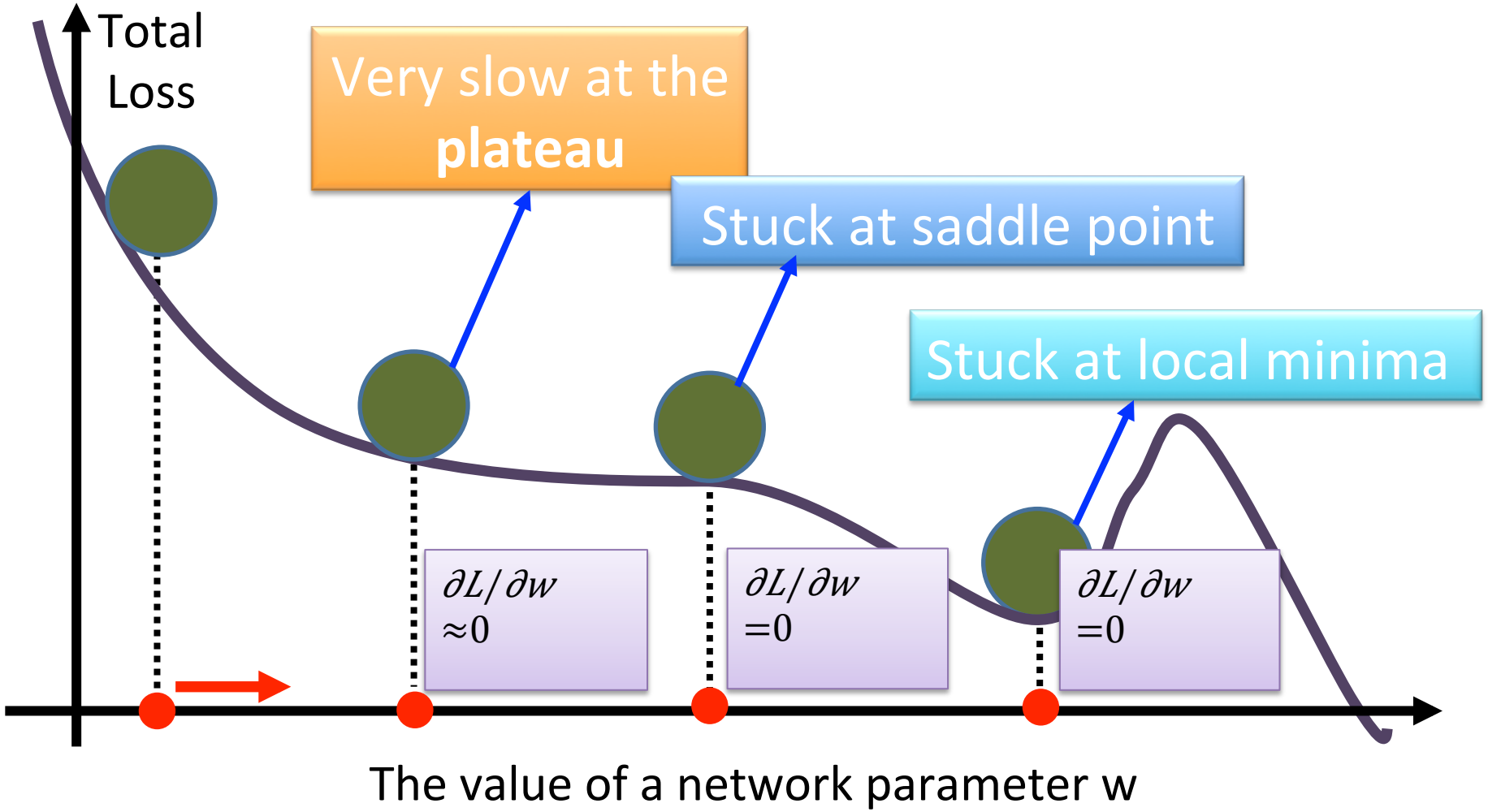
Not the whole story ……

- Adagrad [John Duchi, JMLR'11]
- RMSprop
  - https://www.youtube.com/watch?v=O3sxAc4hxZU
- Adadelta [Matthew D. Zeiler, arXiv'12]
- "No more pesky learning rates" [Tom Schaul, arXiv'12]
- AdaSecant [Caglar Gulcehre, arXiv'14]
- Adam [Diederik P. Kingma, ICLR'15]
- Nadam
  - http://cs229.stanford.edu/proj2015/054_report.pdf

Recipe of Deep Learning

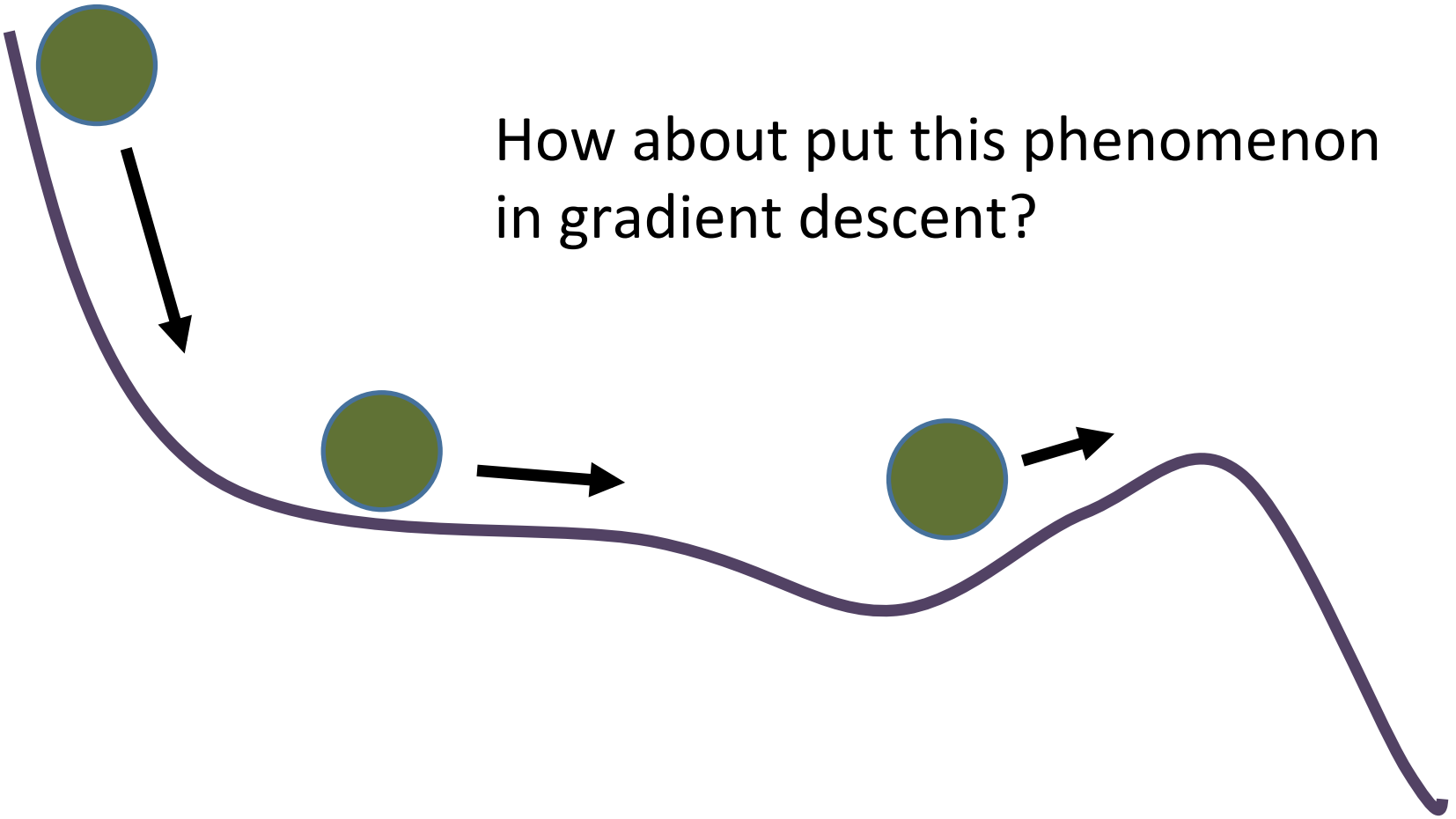


Hard to find optimal network parameters



In physical world ……

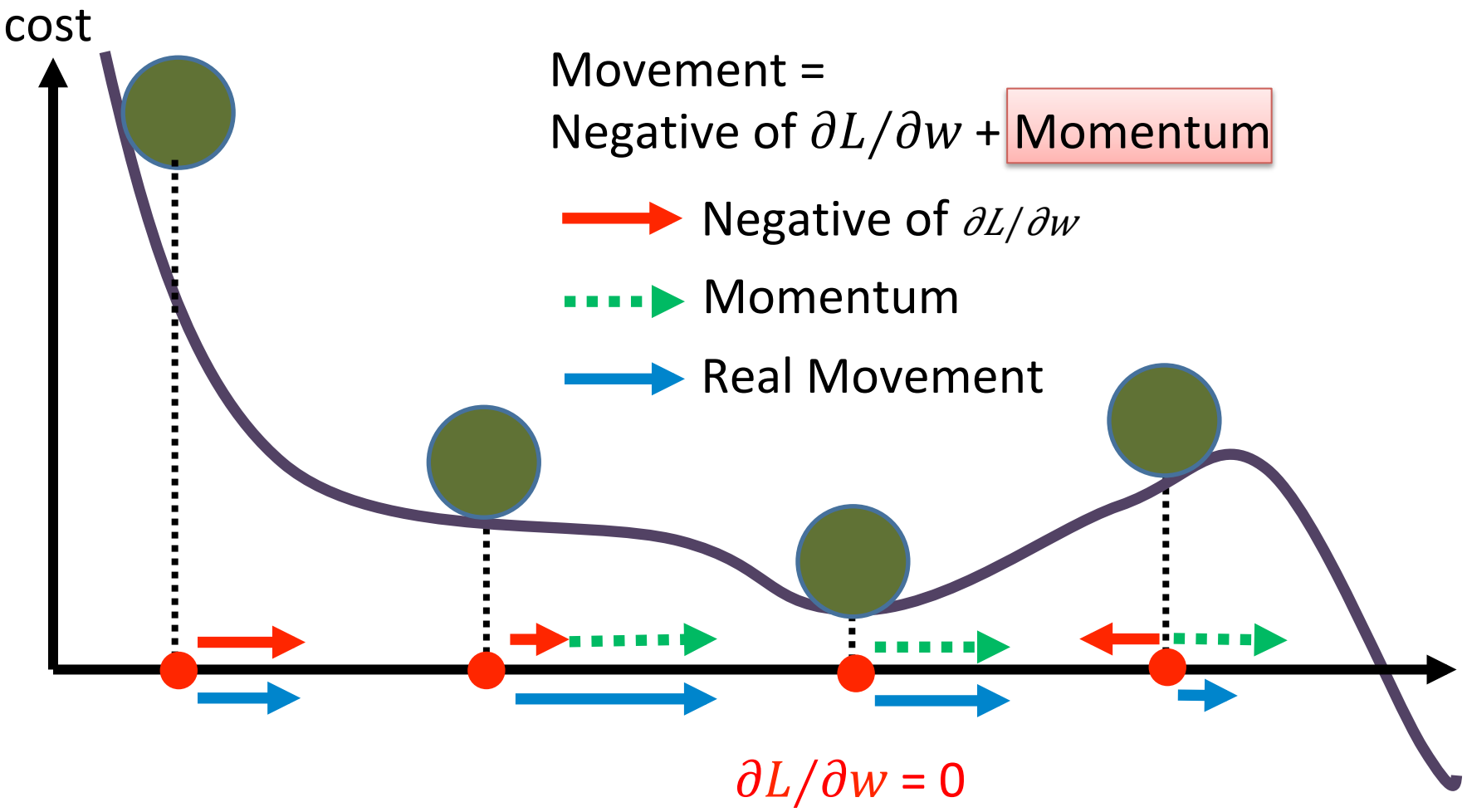
- Momentum



How about put this phenomenon in gradient descent?

Momentum

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



RMSProp (Advanced Adagrad) + Momentum

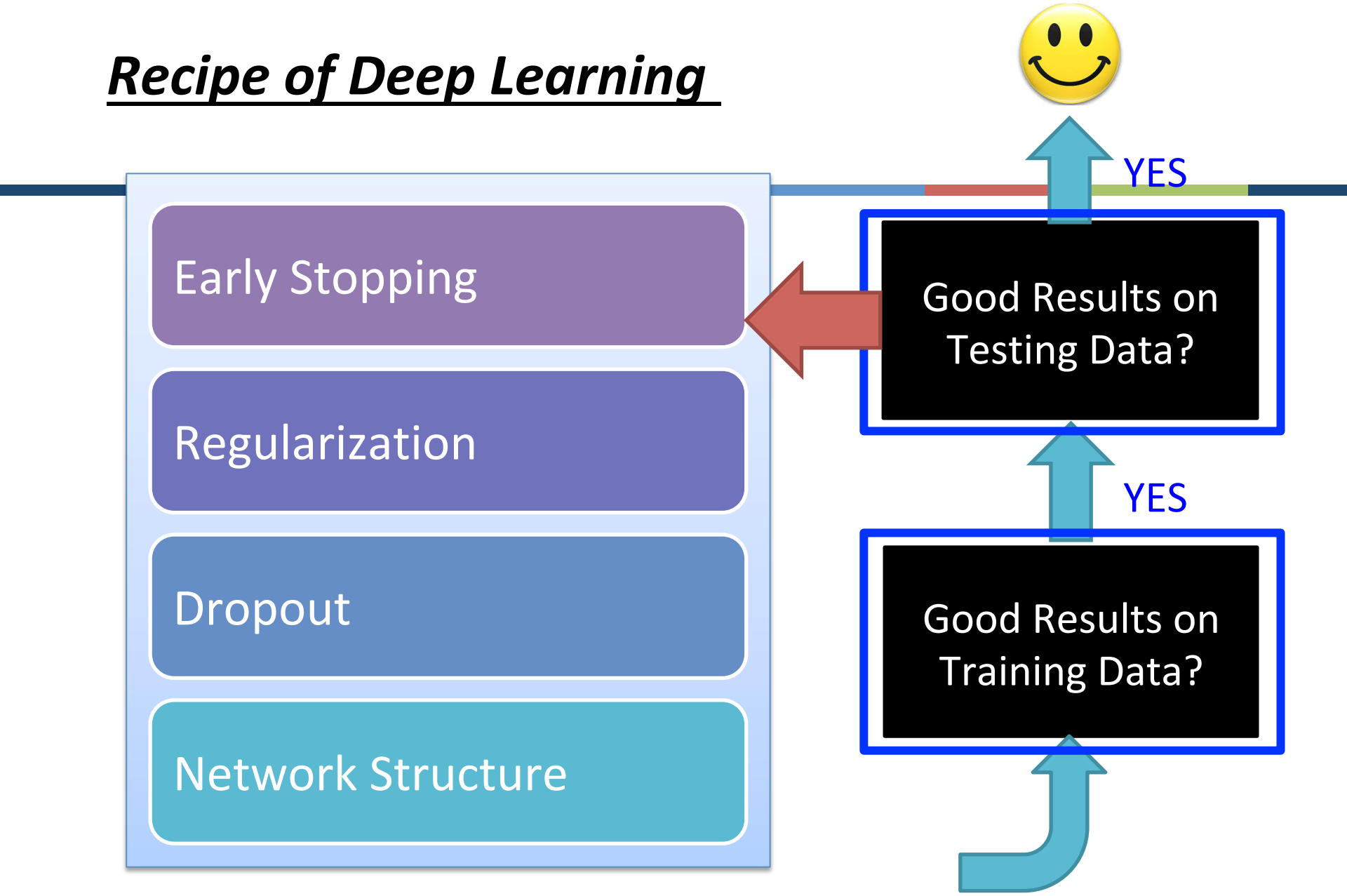
```
model.compile(loss='categorical_crossentropy',
              optimizer=SGD(lr=0.1),
              metrics=['accuracy'])
```

```
model.compile(loss='categorical_crossentropy',
              optimizer=Adam(),
              metrics=['accuracy'])
```

Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1st moment vector)
  $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
  $t \leftarrow 0$ (Initialize timestep)
  while θ_t not converged **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
  end while
  return θ_t (Resulting parameters)

Recipe of Deep Learning

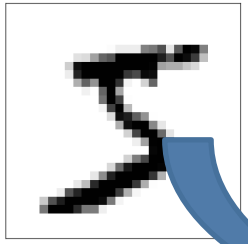


Panacea for Overfitting

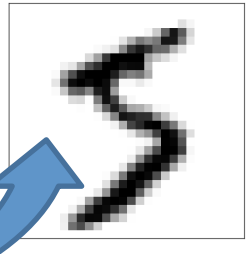
- Have more training data
- ***Create*** more training data (?)

Handwriting recognition:

Original
Training Data:

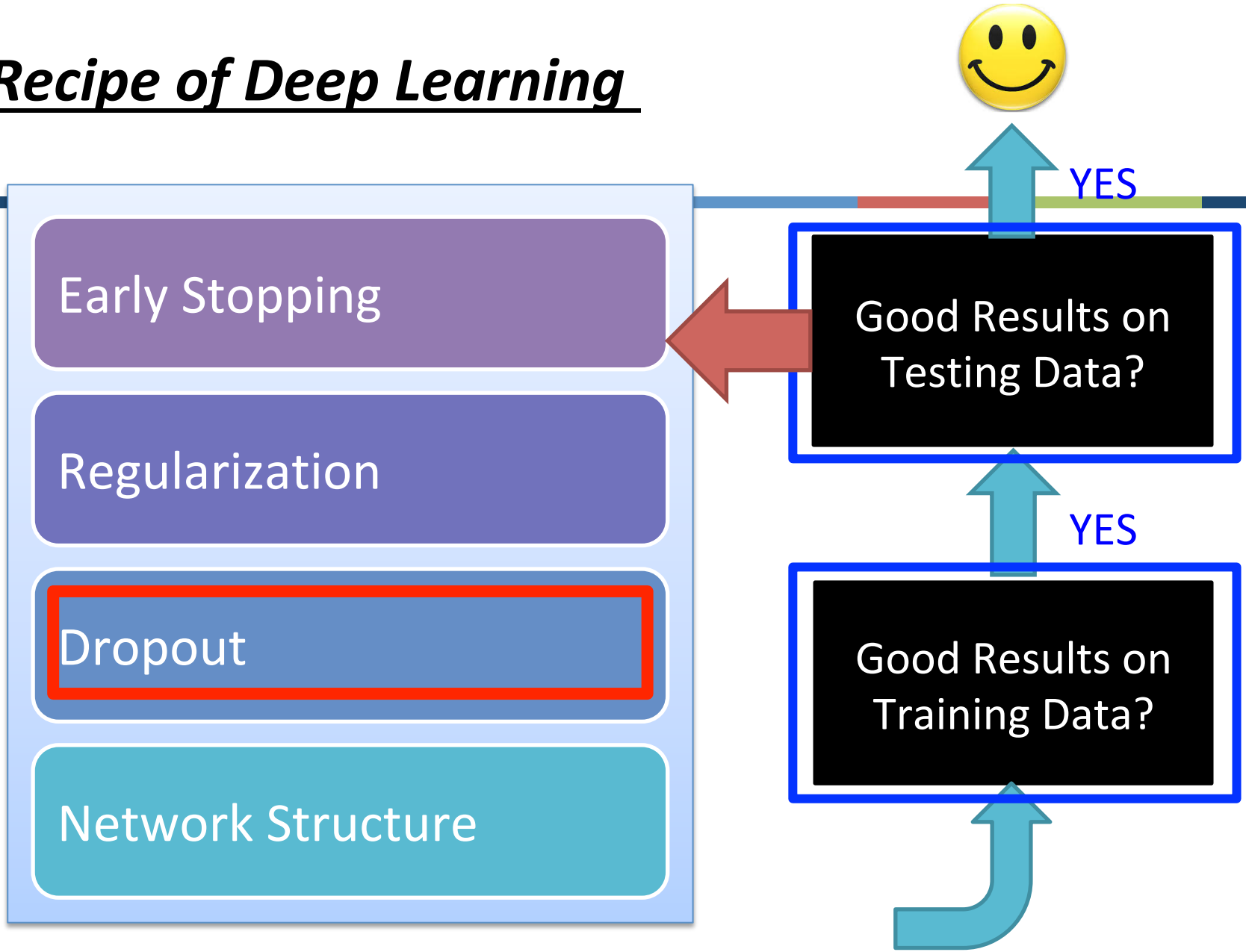


Created
Training Data:



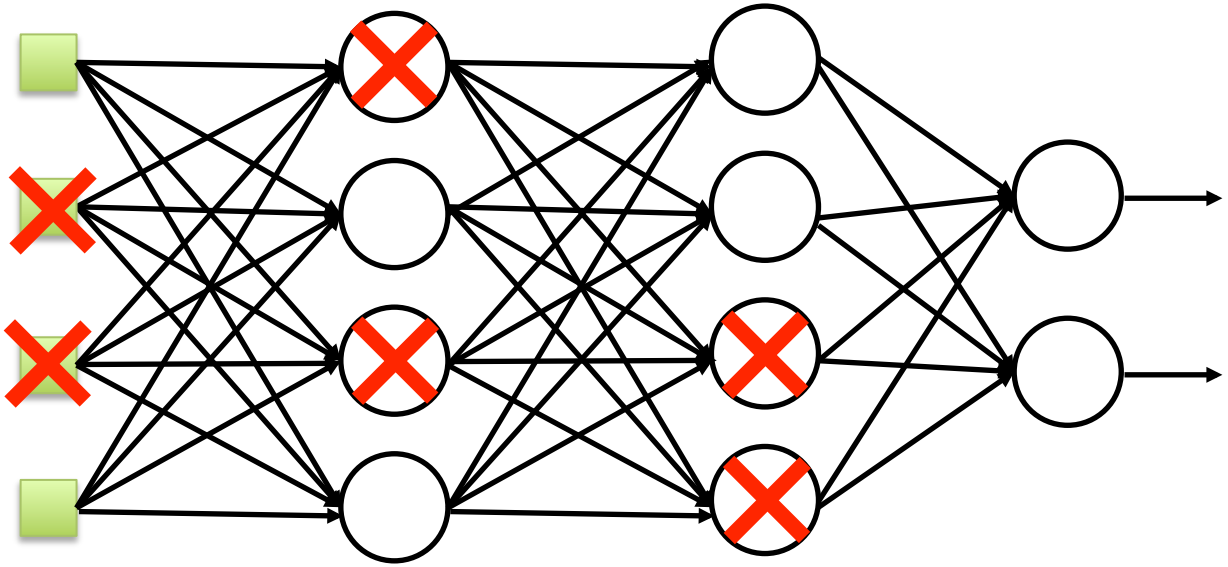
Shift 15°

Recipe of Deep Learning



Dropout

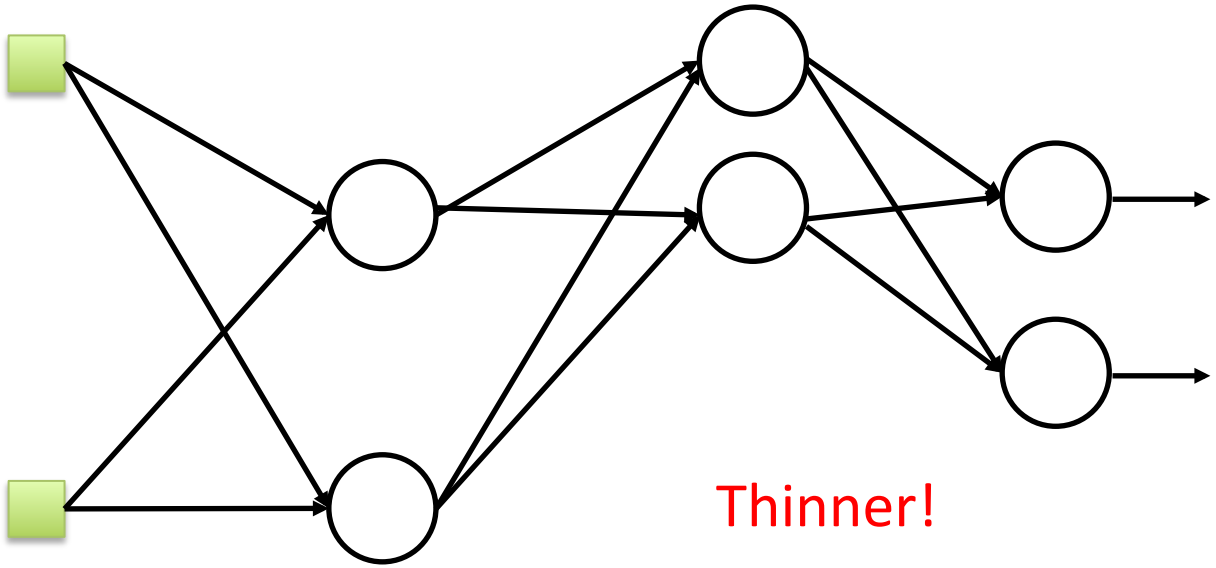
Training:



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

Dropout

Training:



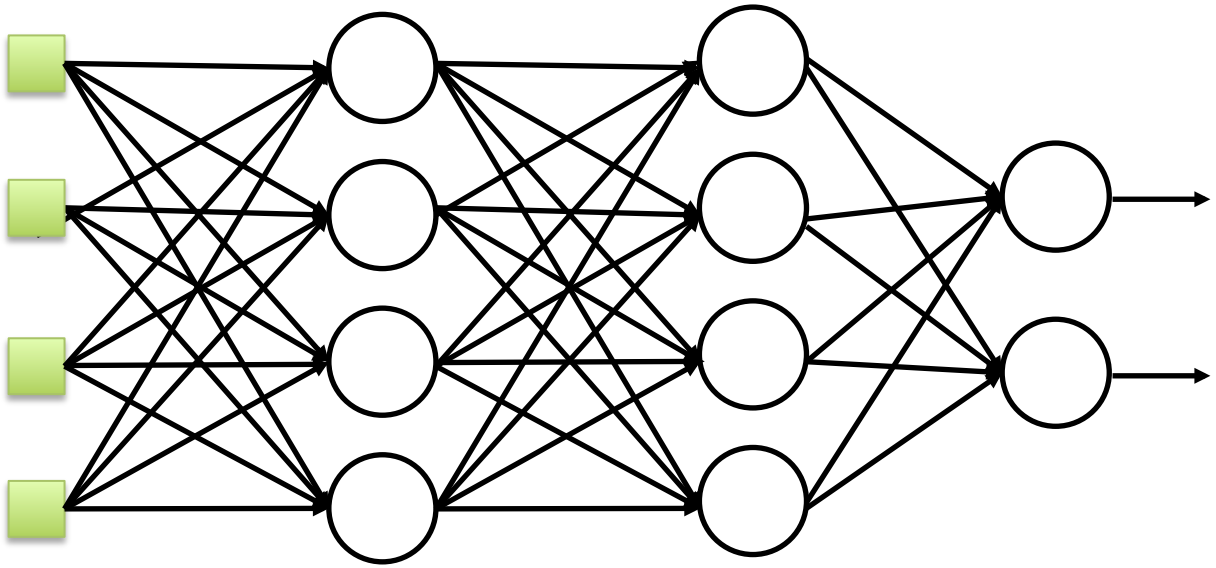
- **Each time before updating the parameters**
  - Each neuron has p% to dropout

    The structure of the network is changed.
  - Using the new network for training

For each mini-batch, we resample the dropout neurons

Dropout

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 $w=1$ by training, set $w=0.5$ for testing.

Dropout - Intuitive Reason

Testing

No dropout

Training

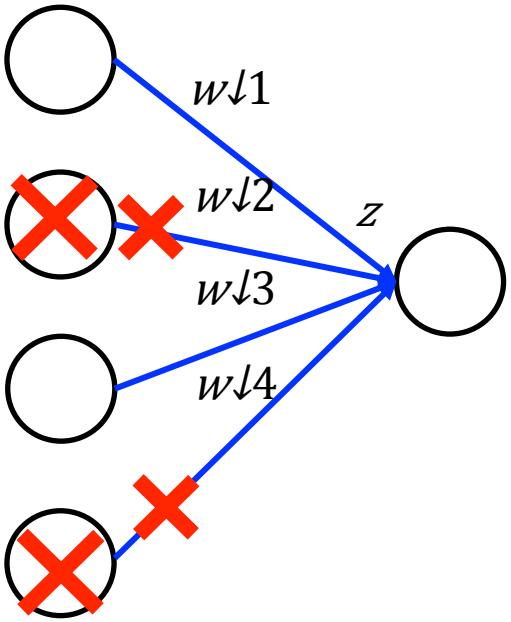
Dropout

Dropout - Intuitive Reason

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

Training of Dropout

Assume dropout rate is 50%

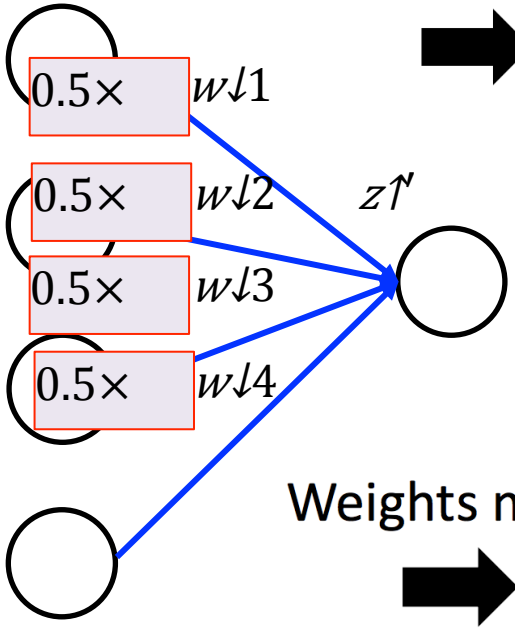


Testing of Dropout

No dropout

Weights from training

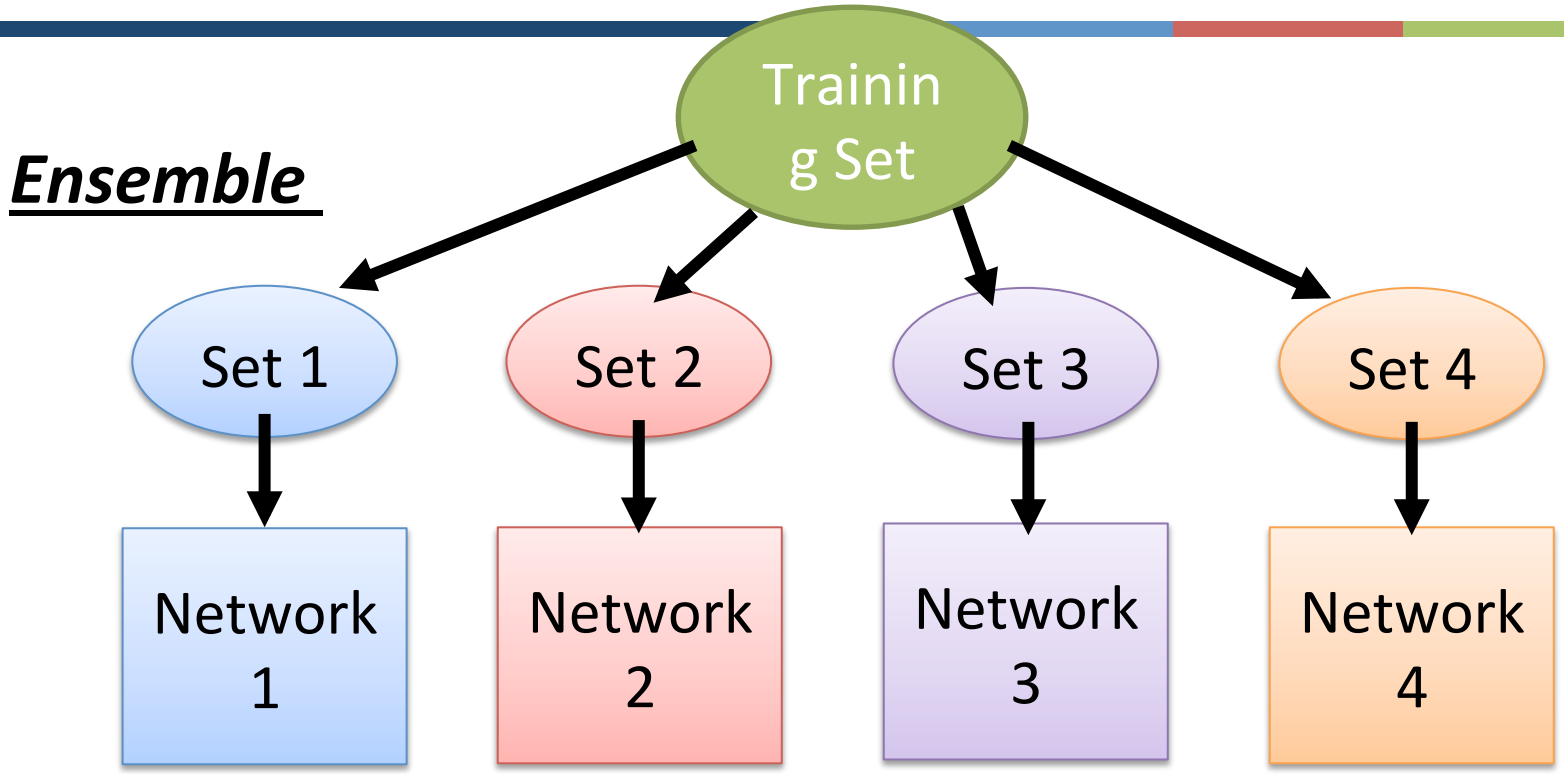
$z' \approx 2z$



Weights multiply (1-p)%

$z' \approx z$

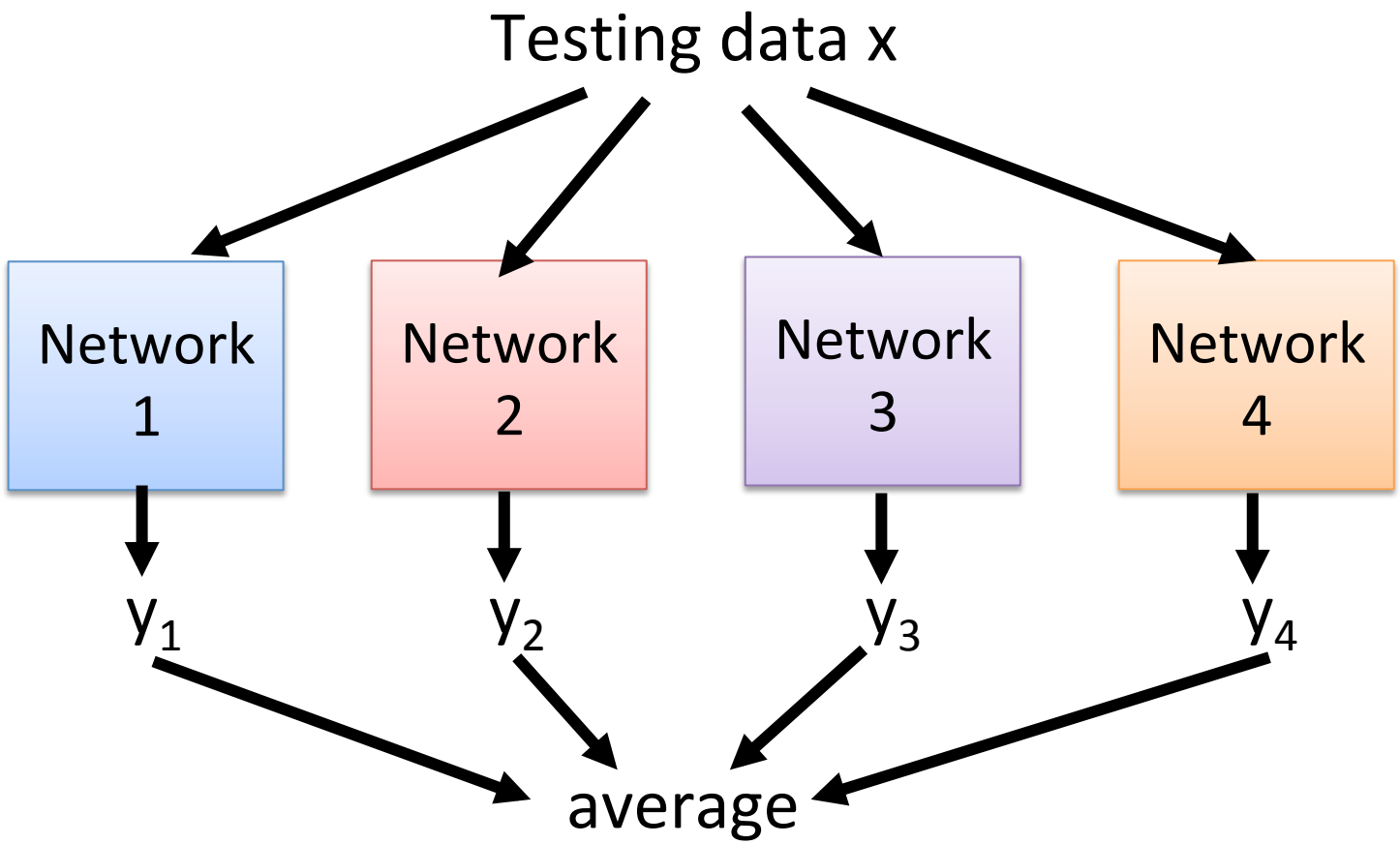
Dropout is a kind of ensemble.



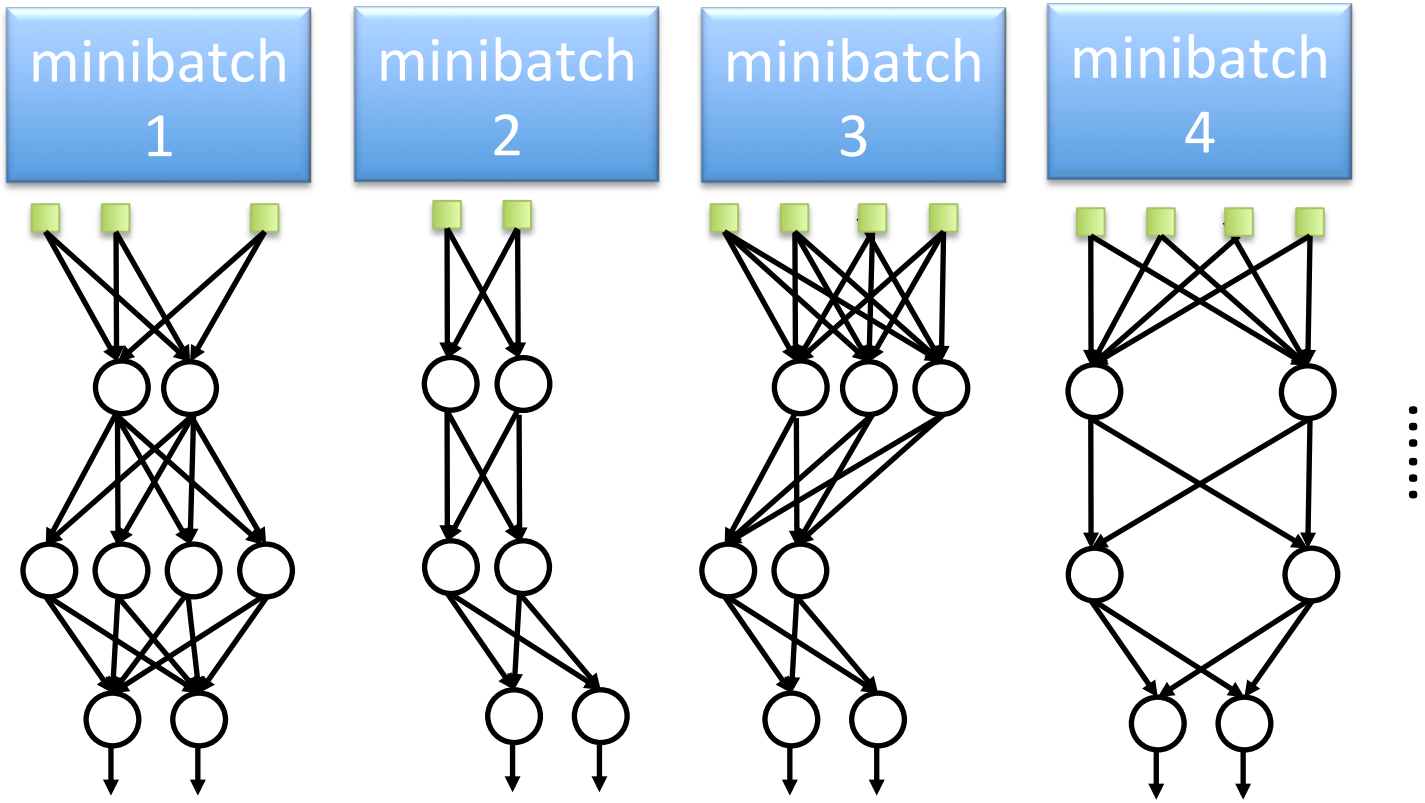
Train a bunch of networks with different structures

Dropout is a kind of ensemble.

Ensemble



Dropout is a kind of ensemble.



Training of Dropout

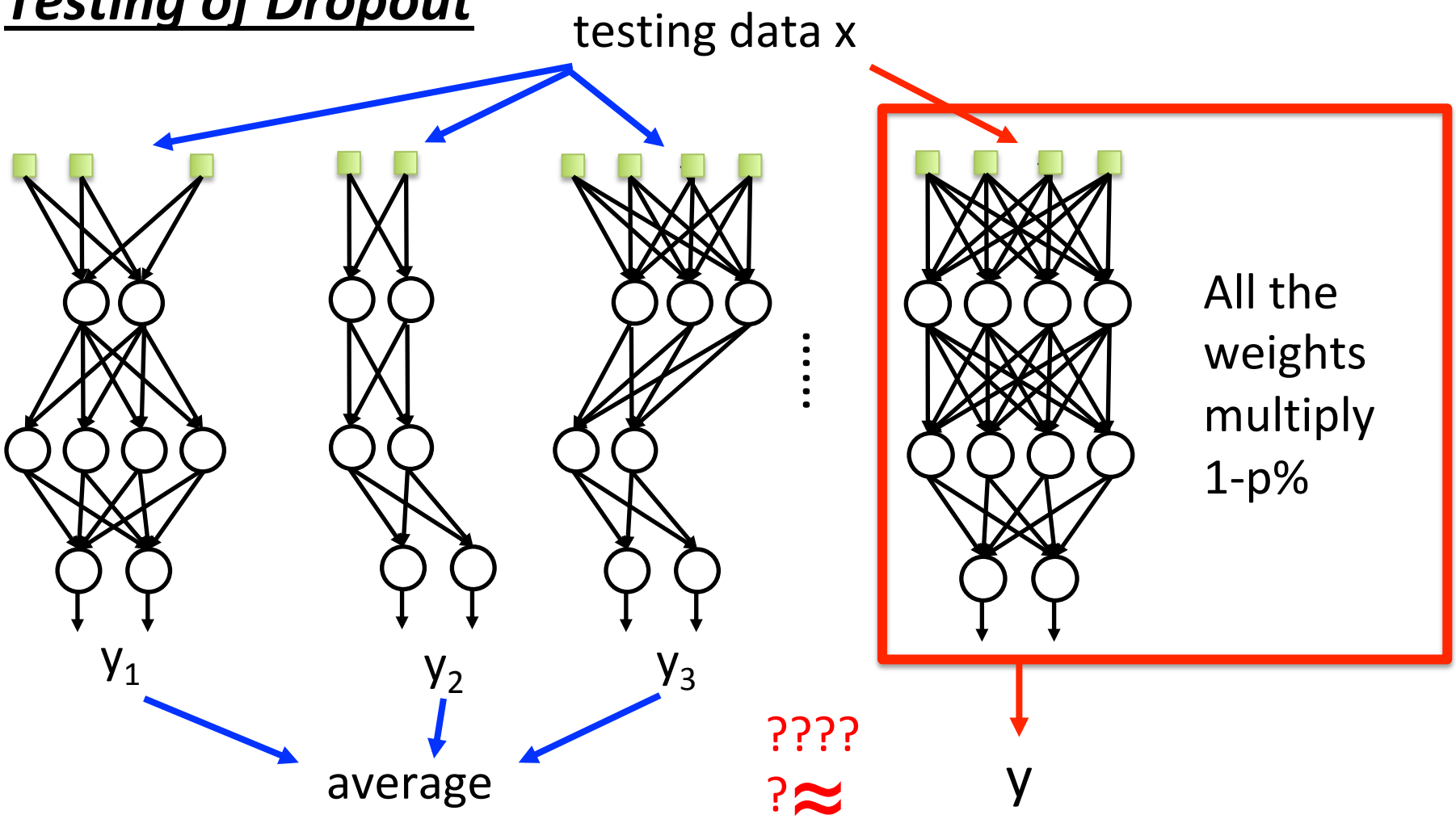
M neurons

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.

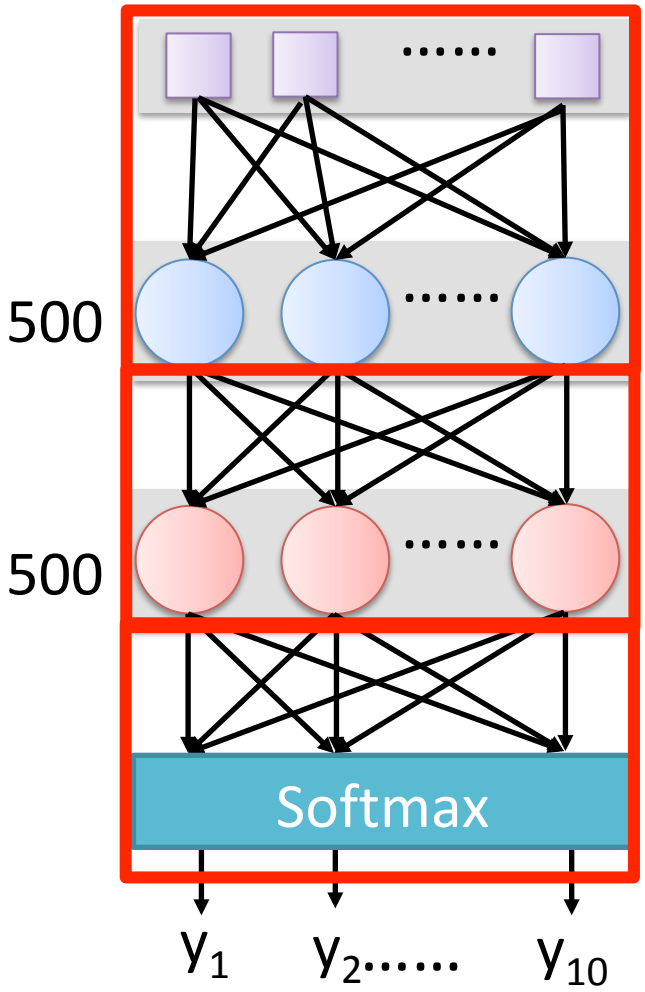
Testing of Dropout



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

Demo




```
model = Sequential()
```

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

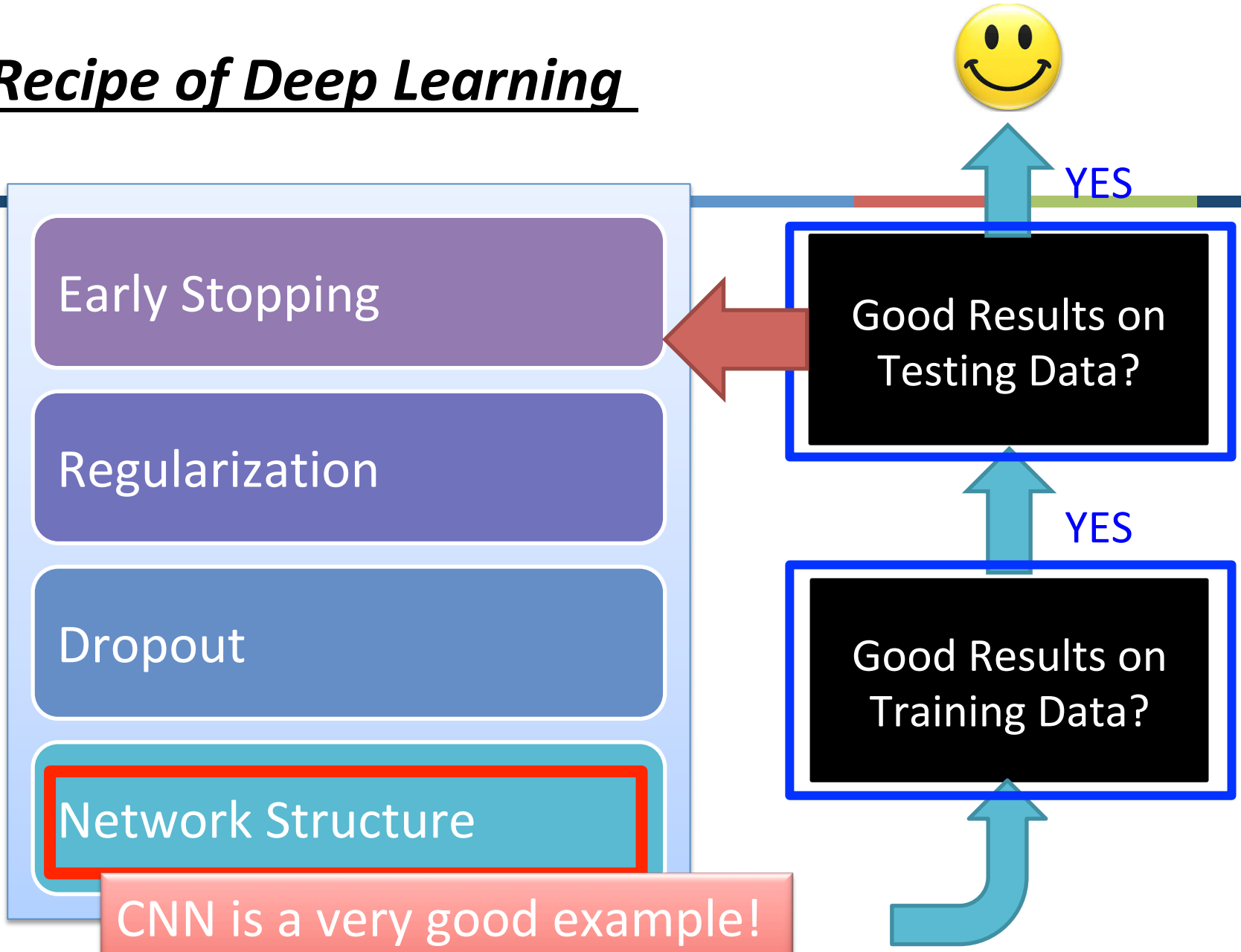
model.add(dropout(0.8))

```
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') )
```

Recipe of Deep Learning



Concluding Remarks

Recipe of Deep Learning

YES

Step 1: define a set of function

Step 2: goodness of function

Step 3: pick the best function

NO

Good Results on Testing Data?

YES

NO

Good Results on Training Data?