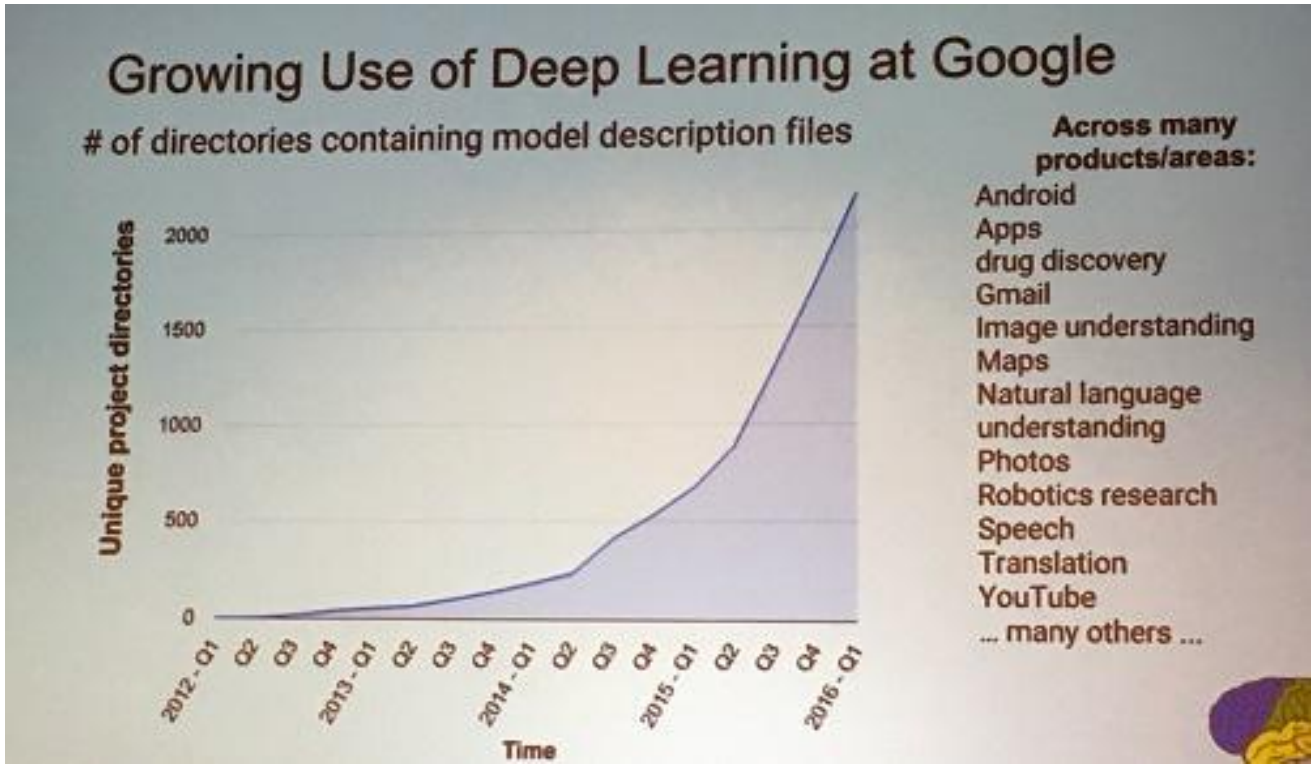
Deep Learning for Beginners

BUDT 758B

Credit to Hung-yi Lee

Deep learning attracts lots of attention.

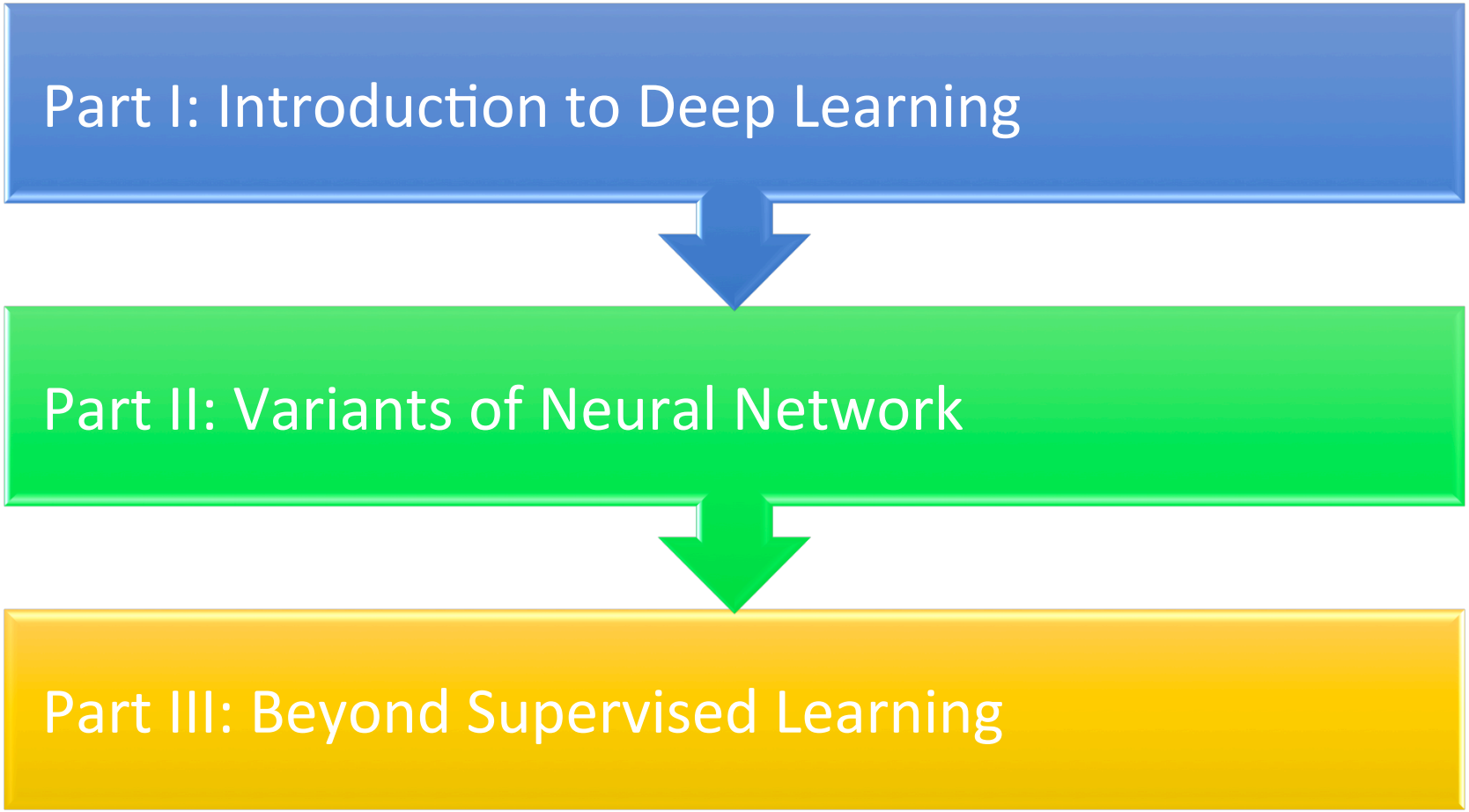
- I believe you have seen lots of exciting results before.



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

This lecture focuses on the basic techniques.

Outline


Part I: Introduction to Deep Learning
Part II: Variants of Neural Network
Part III: Beyond Supervised Learning

Part I:
Introduction to Deep Learning

Outline

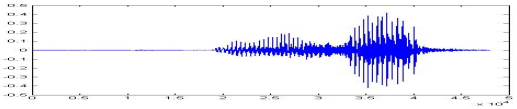
Introduction to Deep Learning

"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" $)=$ "Hello"

(what the user said) (system response)

Framework

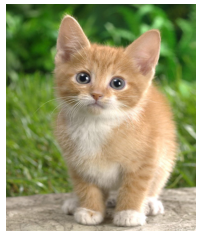
Image Recognition:

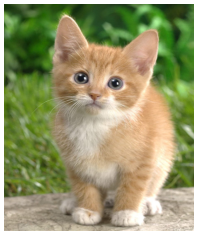
$f($  $)=$ "cat"

A set of functions

Model

$f_1, f_2 \cdots$

$f_1($  $)=$ "cat"

$f_2($  $)=$ "money"

$f_1($  $)=$ "dog"

$f_2($  $)=$ "snake"

Framework

Image Recognition:

$f($  $) =$ "cat"

A set of function

Goodness of function f

Training Data

Model

$f_1, f_2 \cdots$

$f_1($  $) =$ "cat"

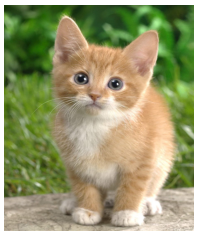
$f_2($  $) =$ "money"

Better!

$f_1($  $) =$ "dog"

$f_2($  $) =$ "snake"

Supervised Learning

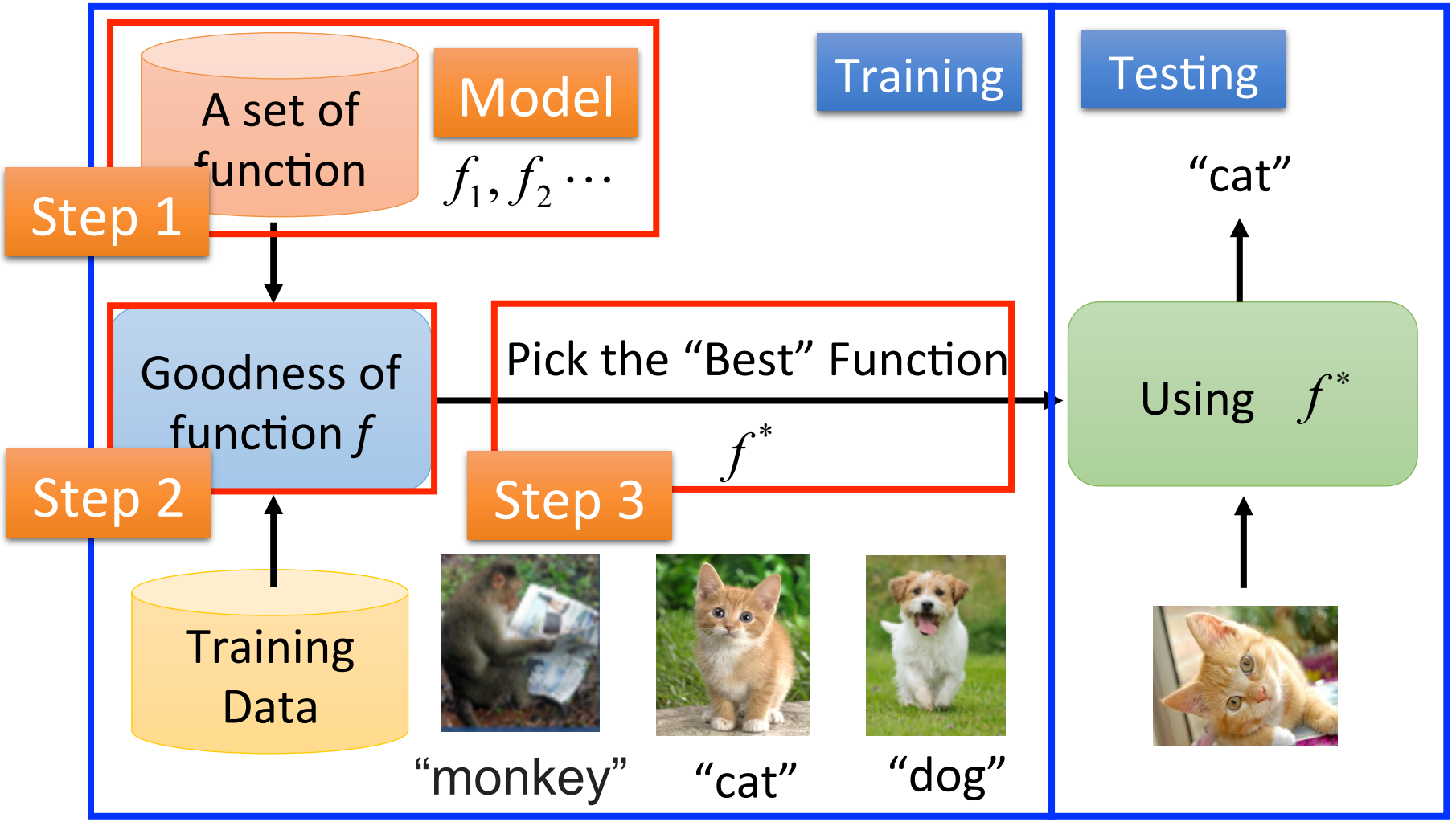
function input:   

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

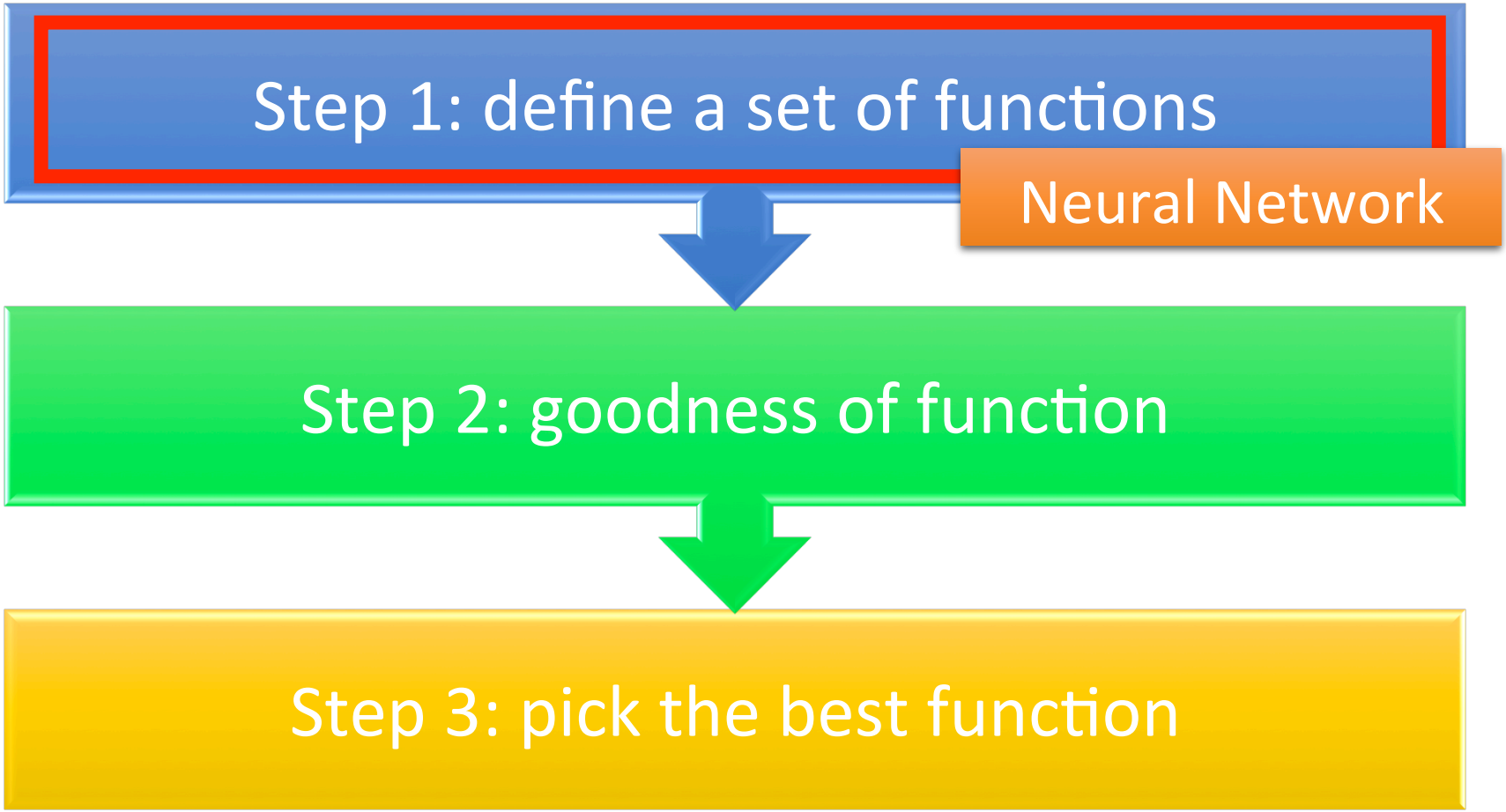
Framework

Image Recognition:

$f($  $) =$ "cat"



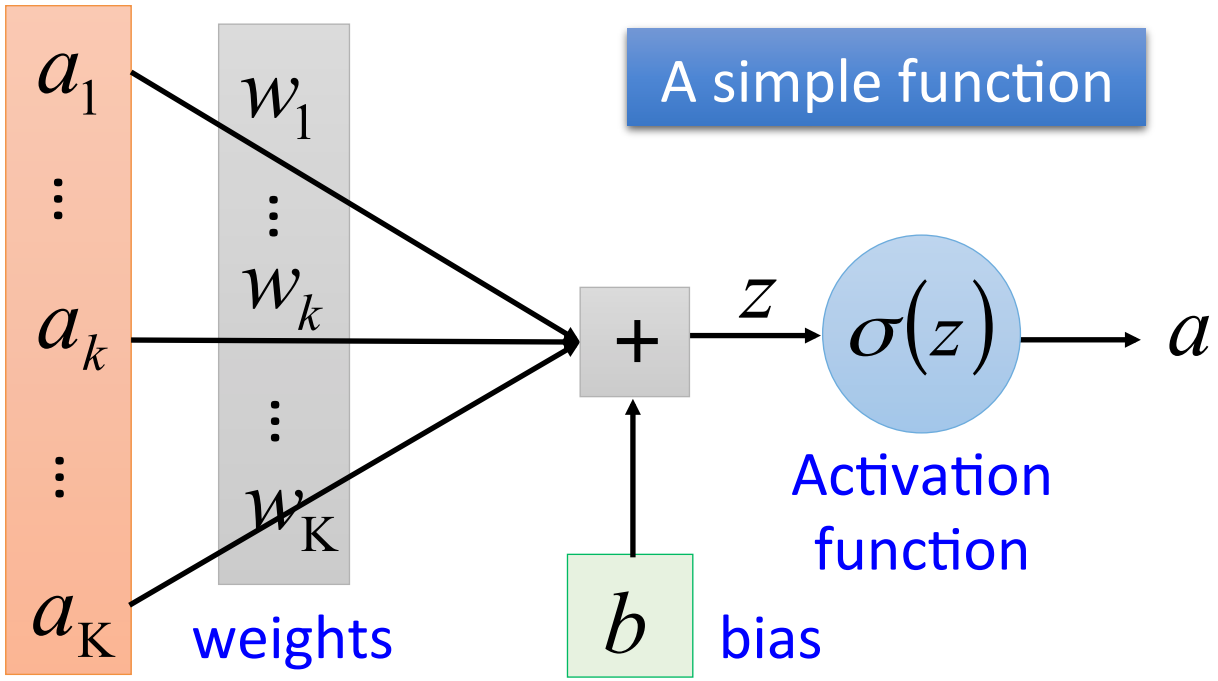
Three Steps for Deep Learning



Neural Network

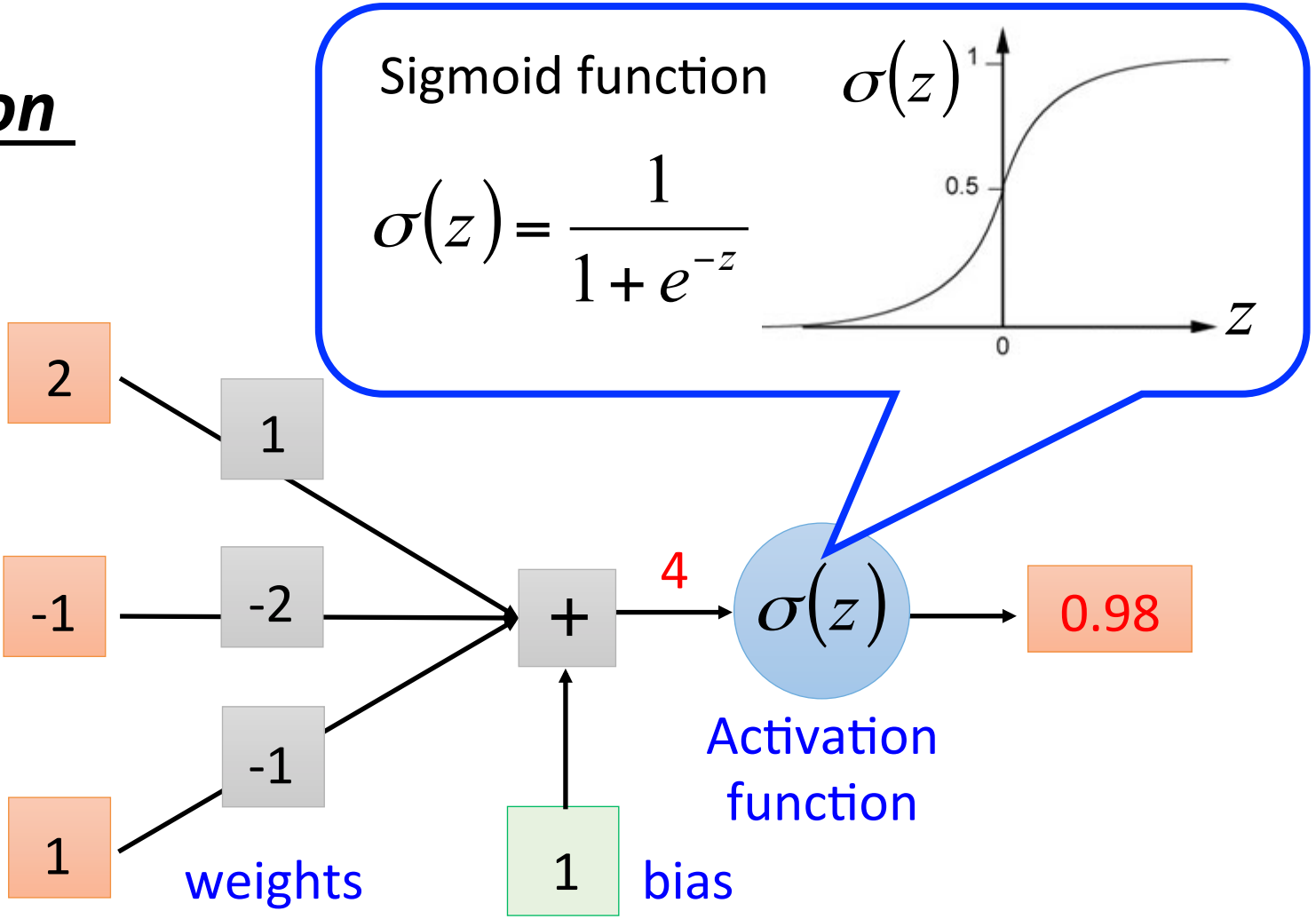
Neuron

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



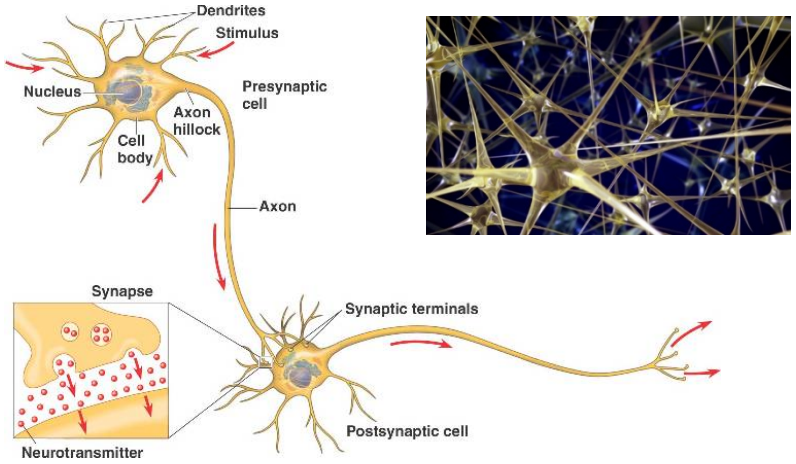
Neural Network

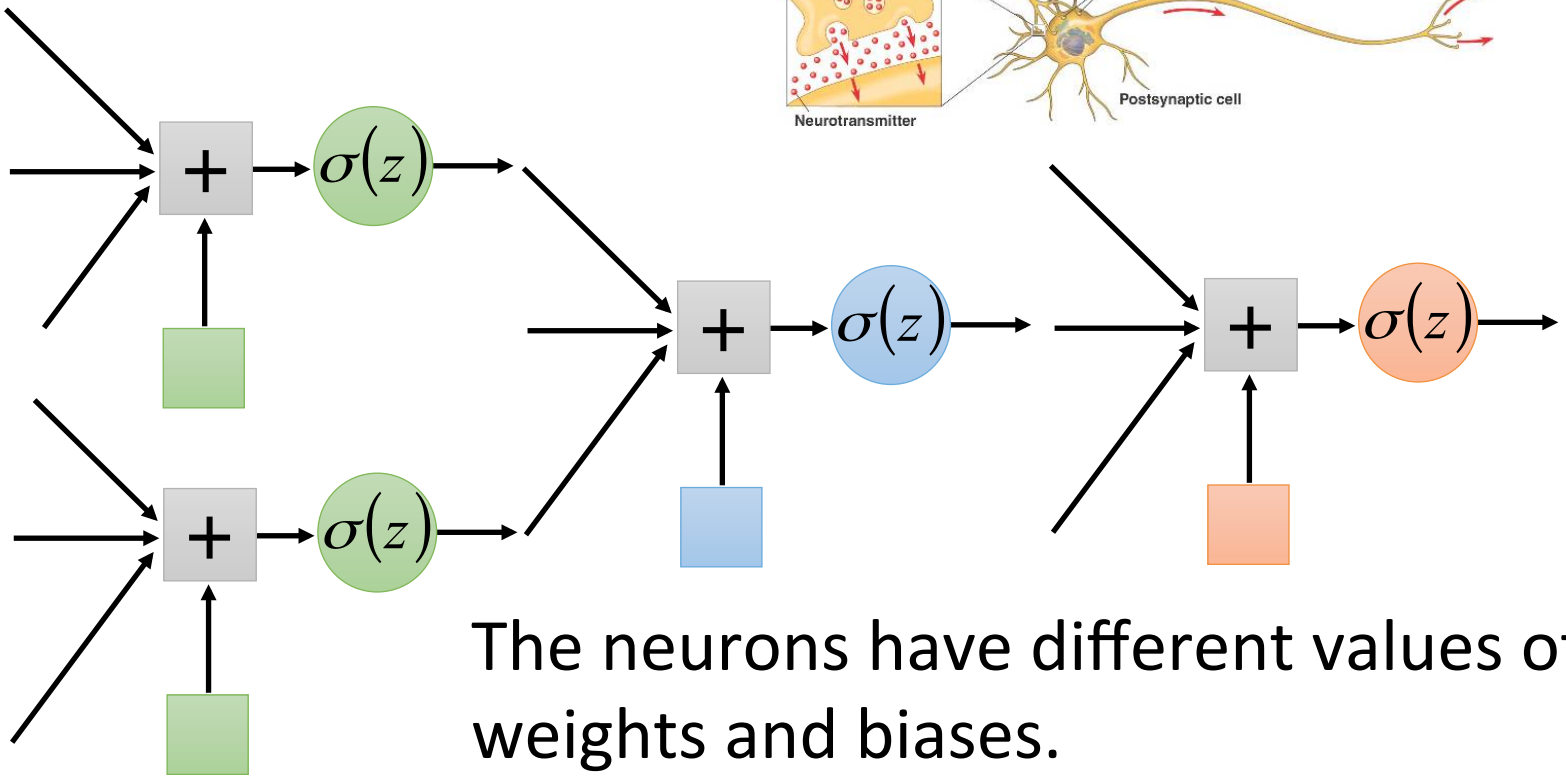
Neuron



Neural Network

Different connections lead to different network structures

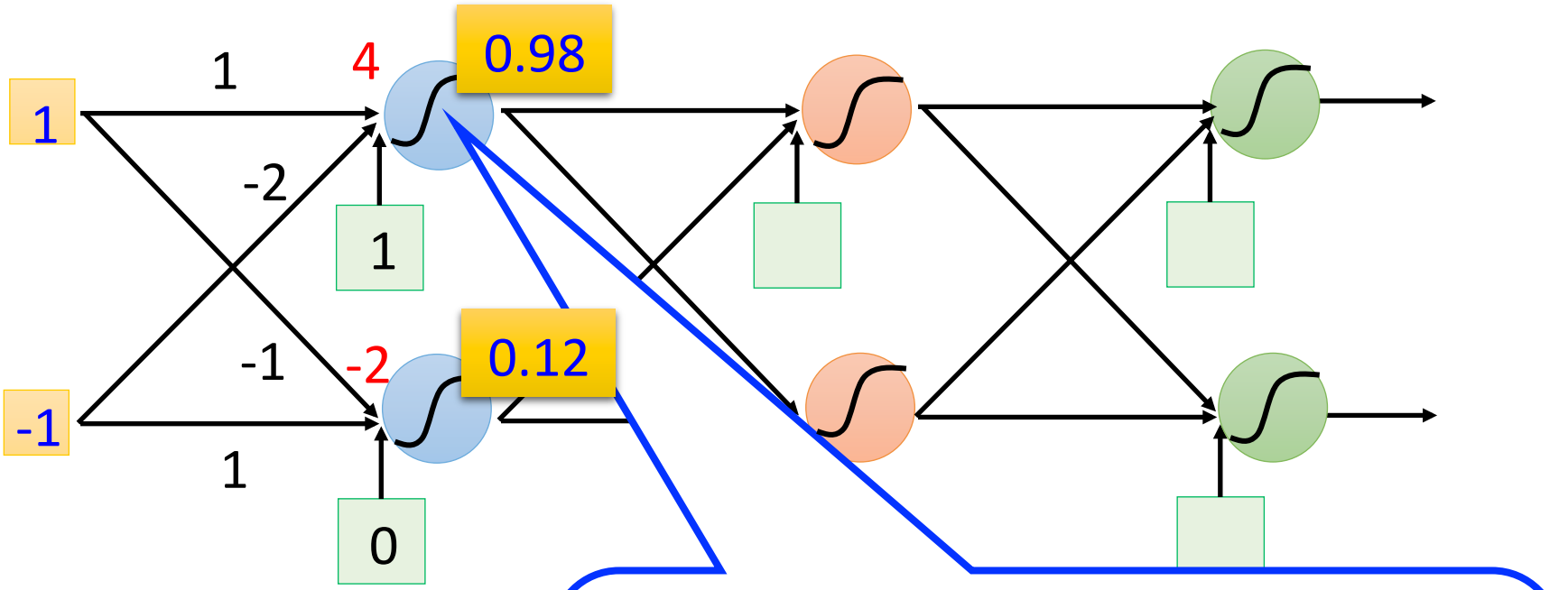




The neurons have different values of weights and biases.

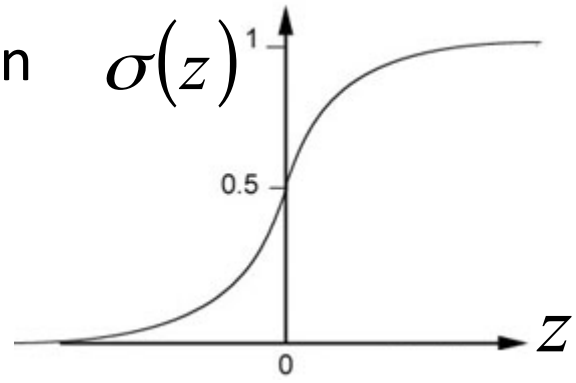
Weights and biases are network parameters θ

Fully Connect Feedforward Network

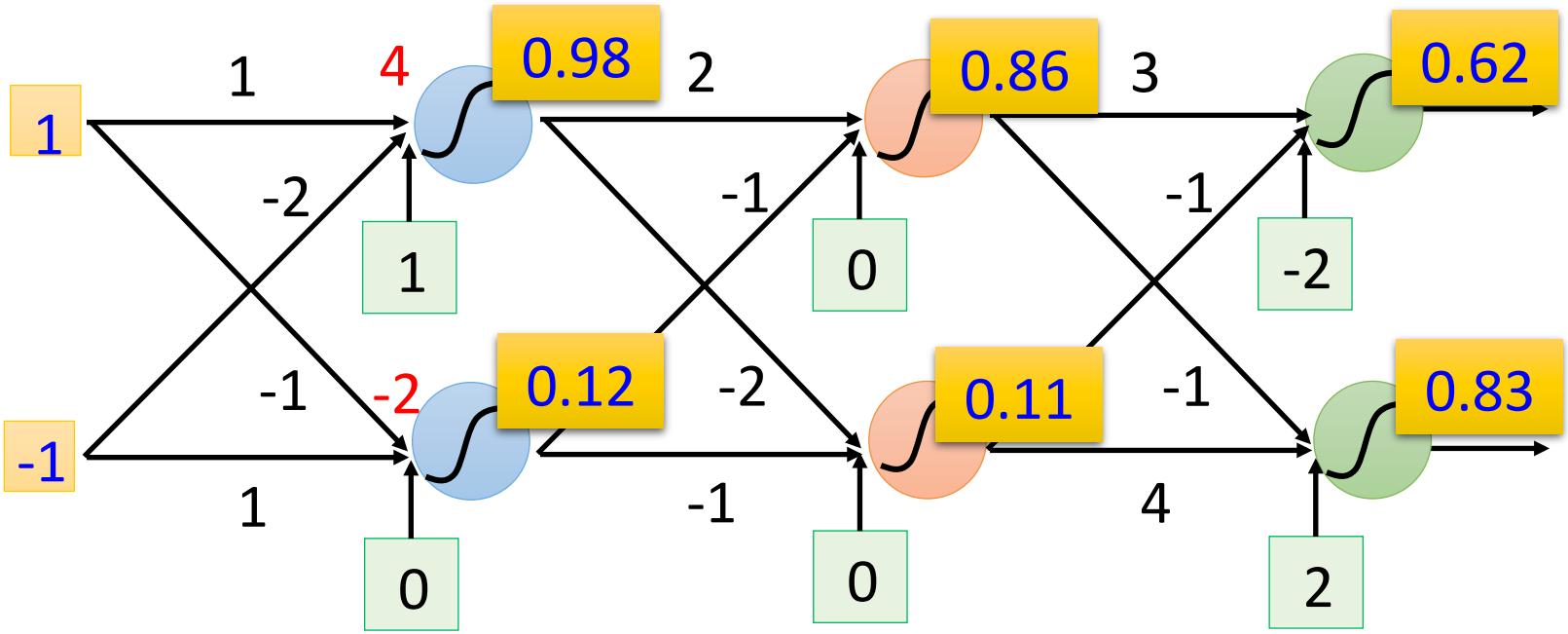


Sigmoid Function

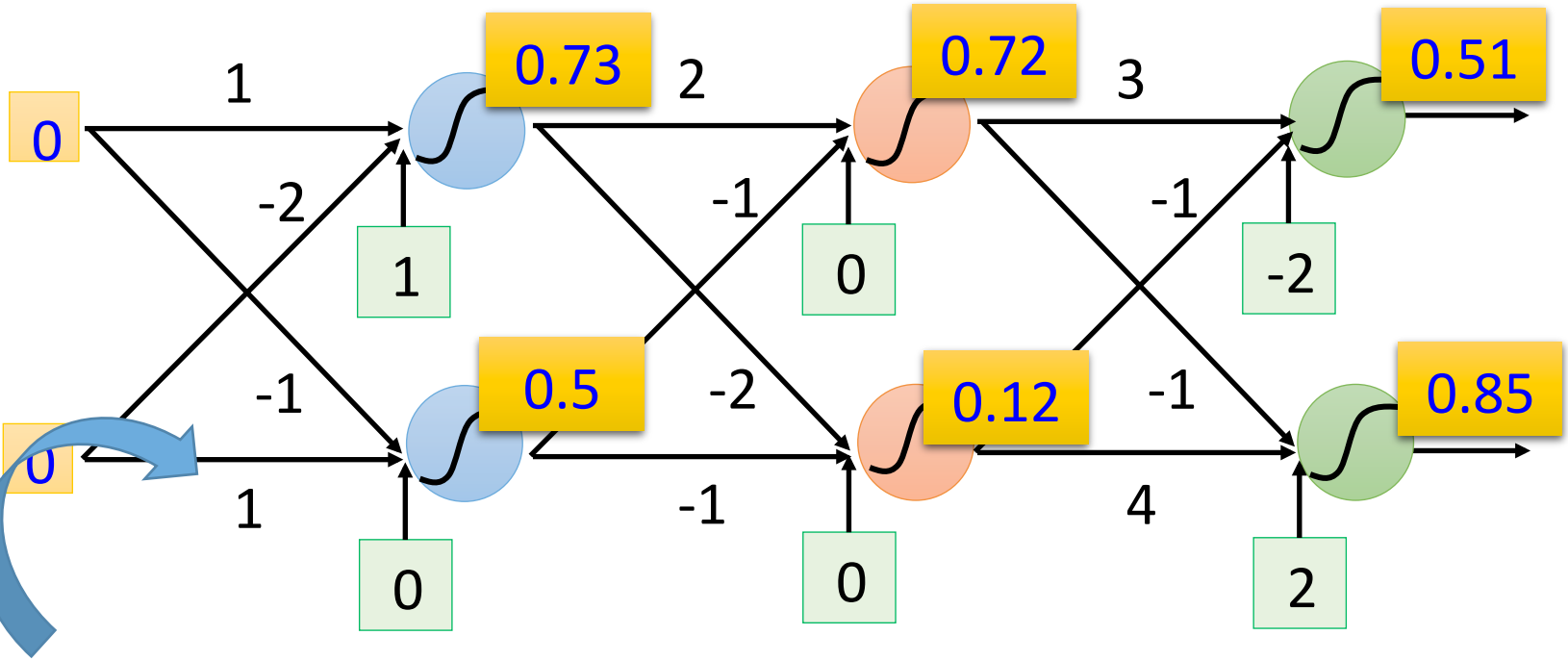
$$\sigma(z) = \frac{1}{1+e^{-z}}$$



Fully Connect Feedforward Network



Fully Connect Feedforward Network



This is a function.

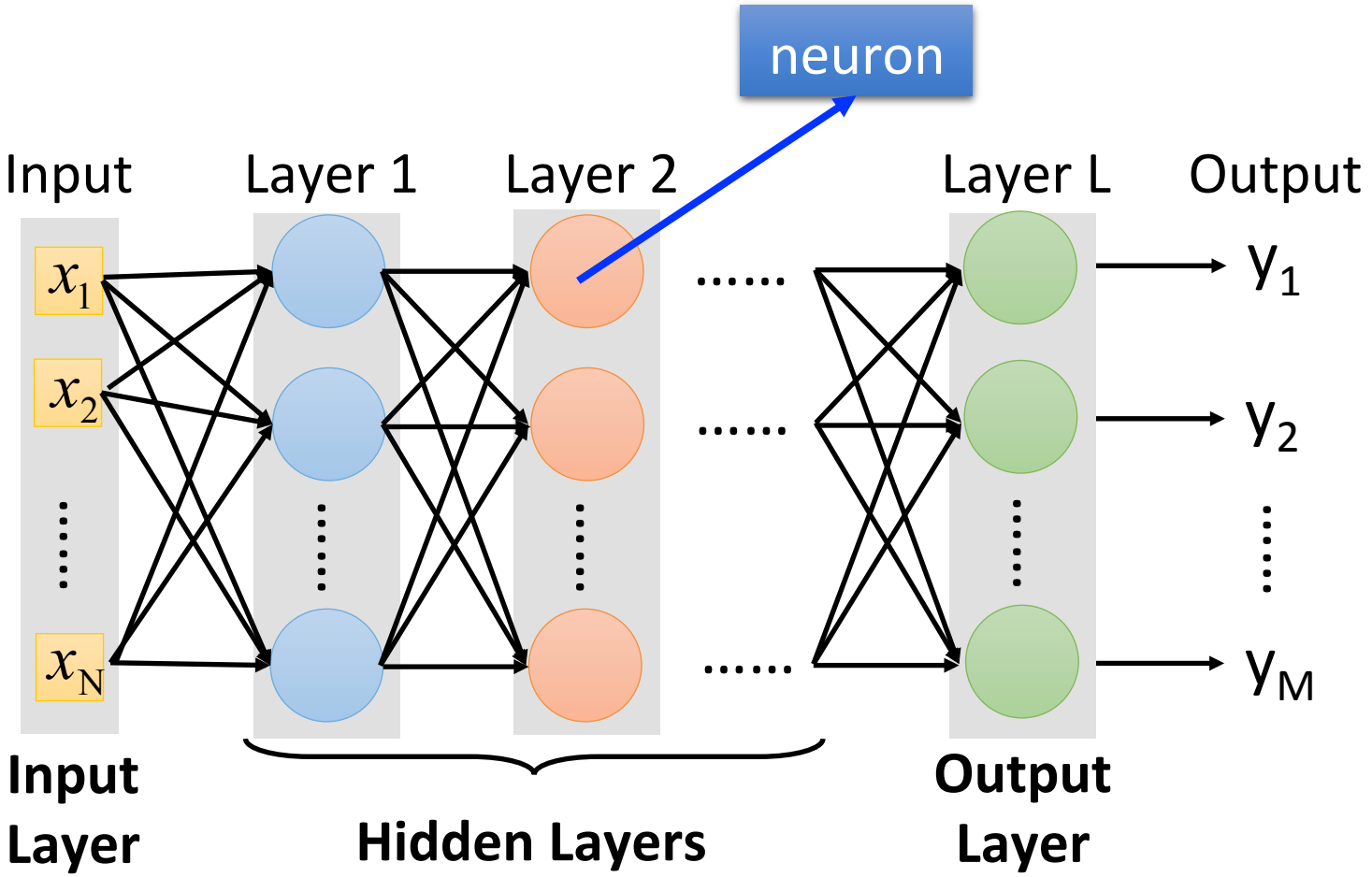
Input vector, output vector

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

Given parameters θ, define a function

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

Fully Connect Feedforward Network



Deep means many hidden layers

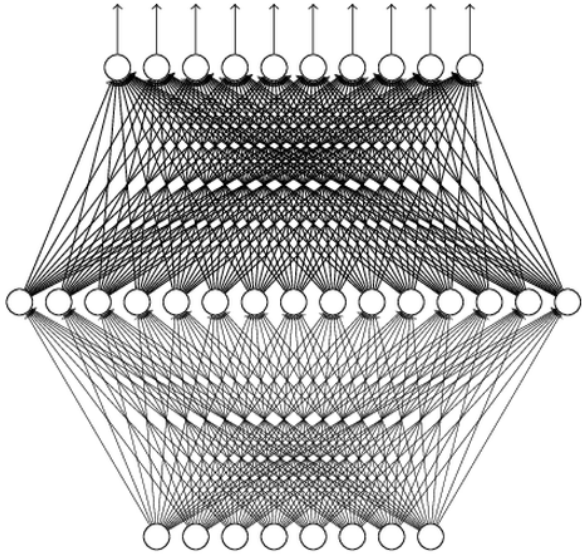
Why Deep? Universality Theorem

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

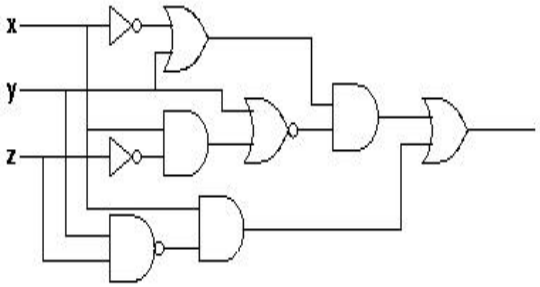
Why “Deep” neural network instead “Fat” neural network?

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

→ less gates needed



Neural network

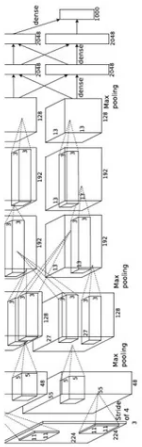
- Neural network consists of **neurons**
- **A hidden layer network** can represent **any continuous function.**
- Using multiple layers of neurons to represent some functions are much simpler

→ less parameters → less data?

Deep = Many hidden layers

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

8 layers



16.4%

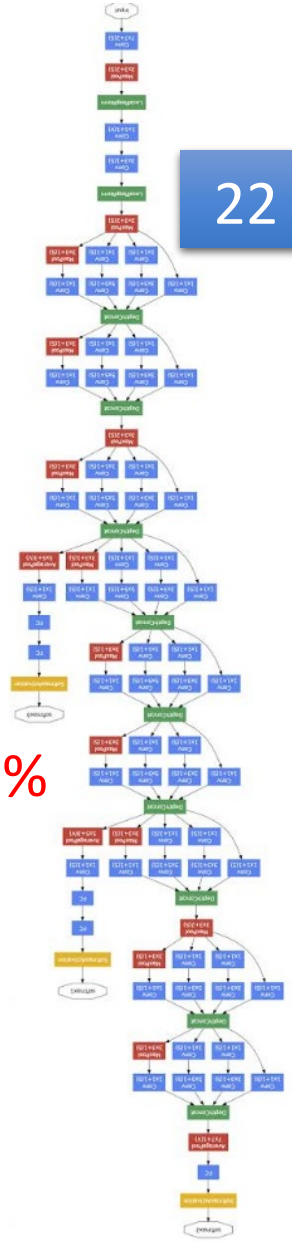
AlexNet (2012)

19 layers

7.3%

VGG (2014)

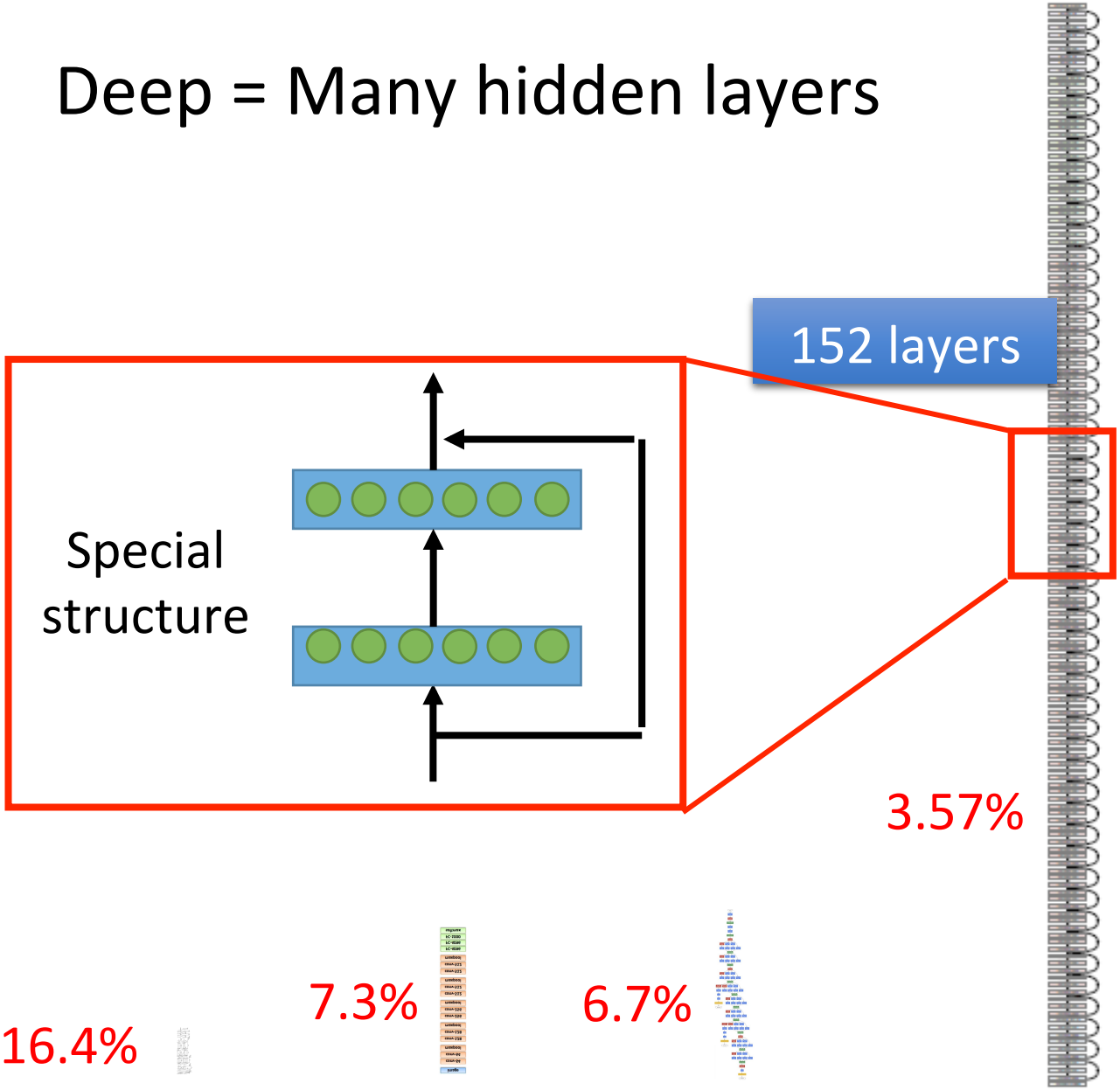
22 layers



6.7%

GoogleNet (2014)

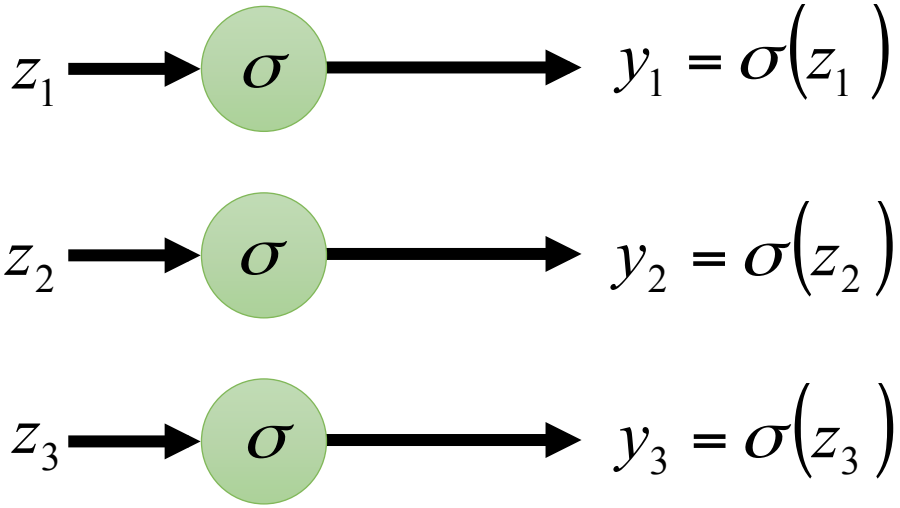
Deep = Many hidden layers



Output Layer

- Softmax layer as the output layer

Ordinary Layer



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

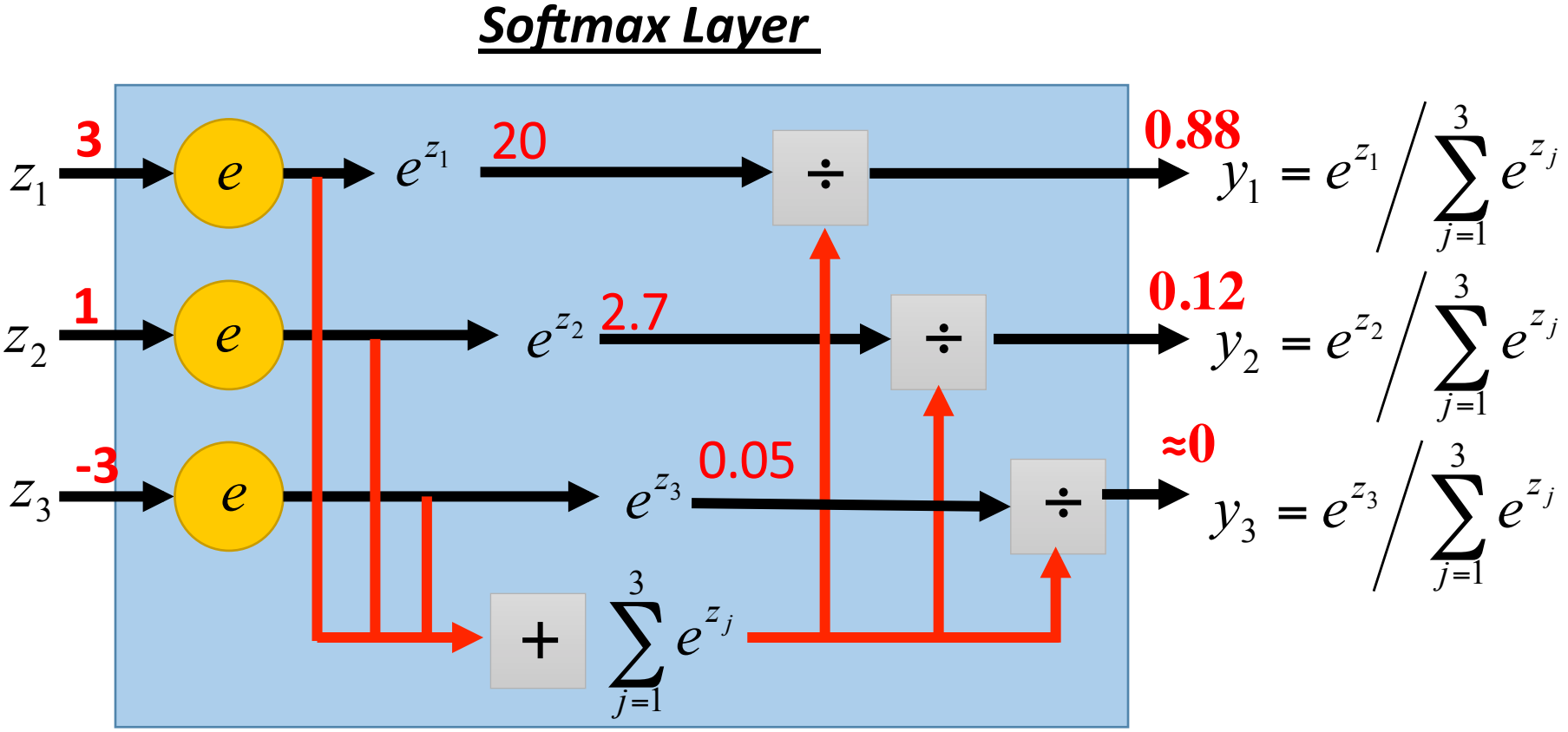
May not be easy to interpret

Output Layer

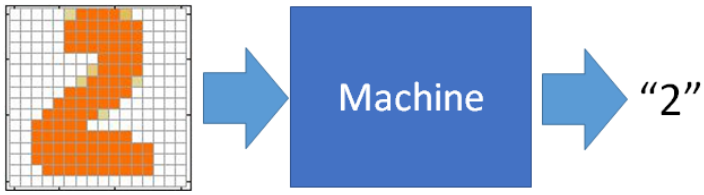
Probability:

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

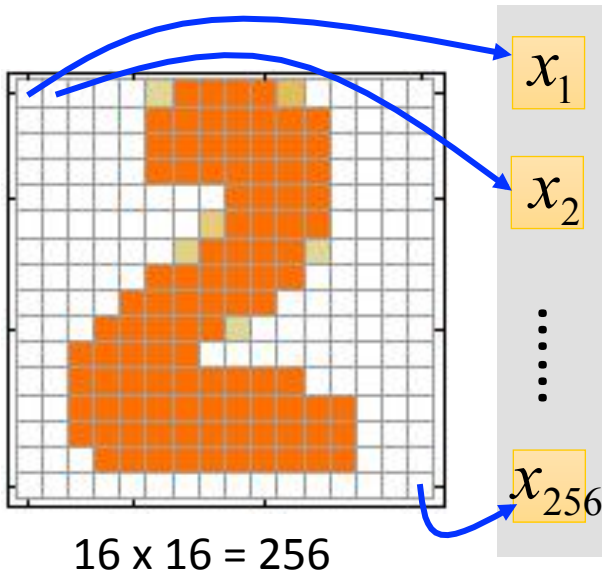
- Softmax layer as the output layer



Example Application



Input

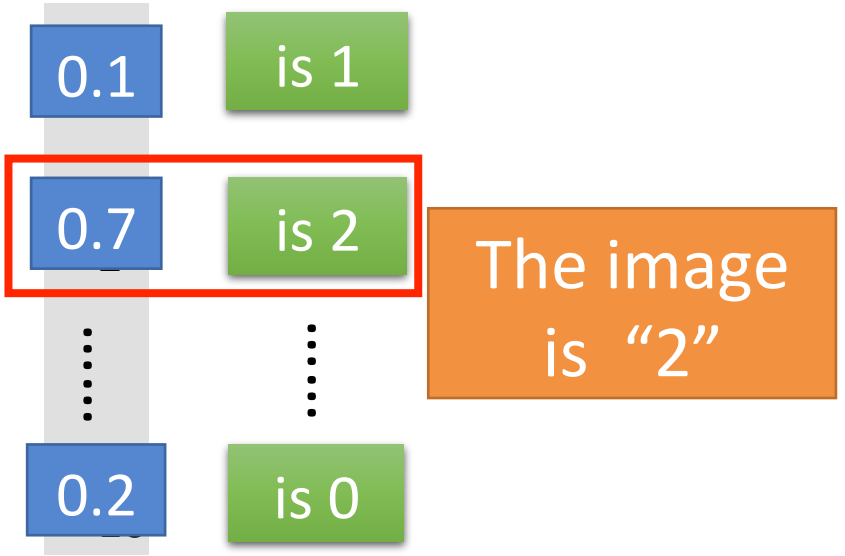


16 x 16 = 256

Ink → 1

No ink → 0

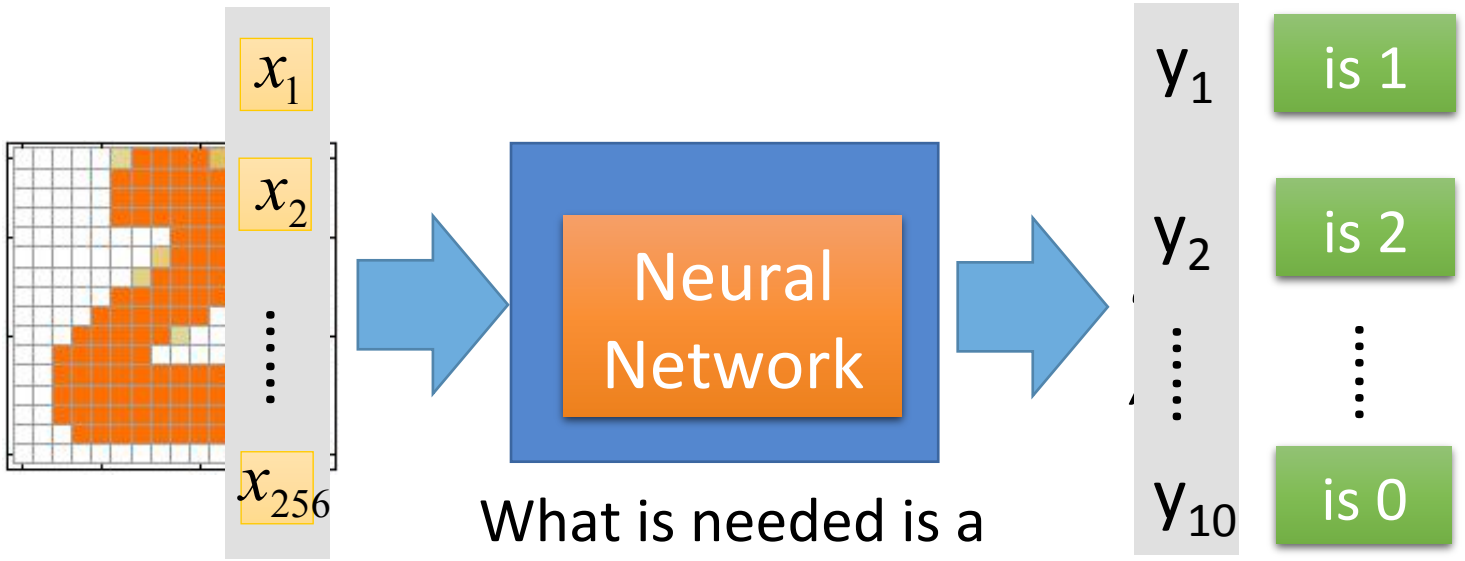
Output



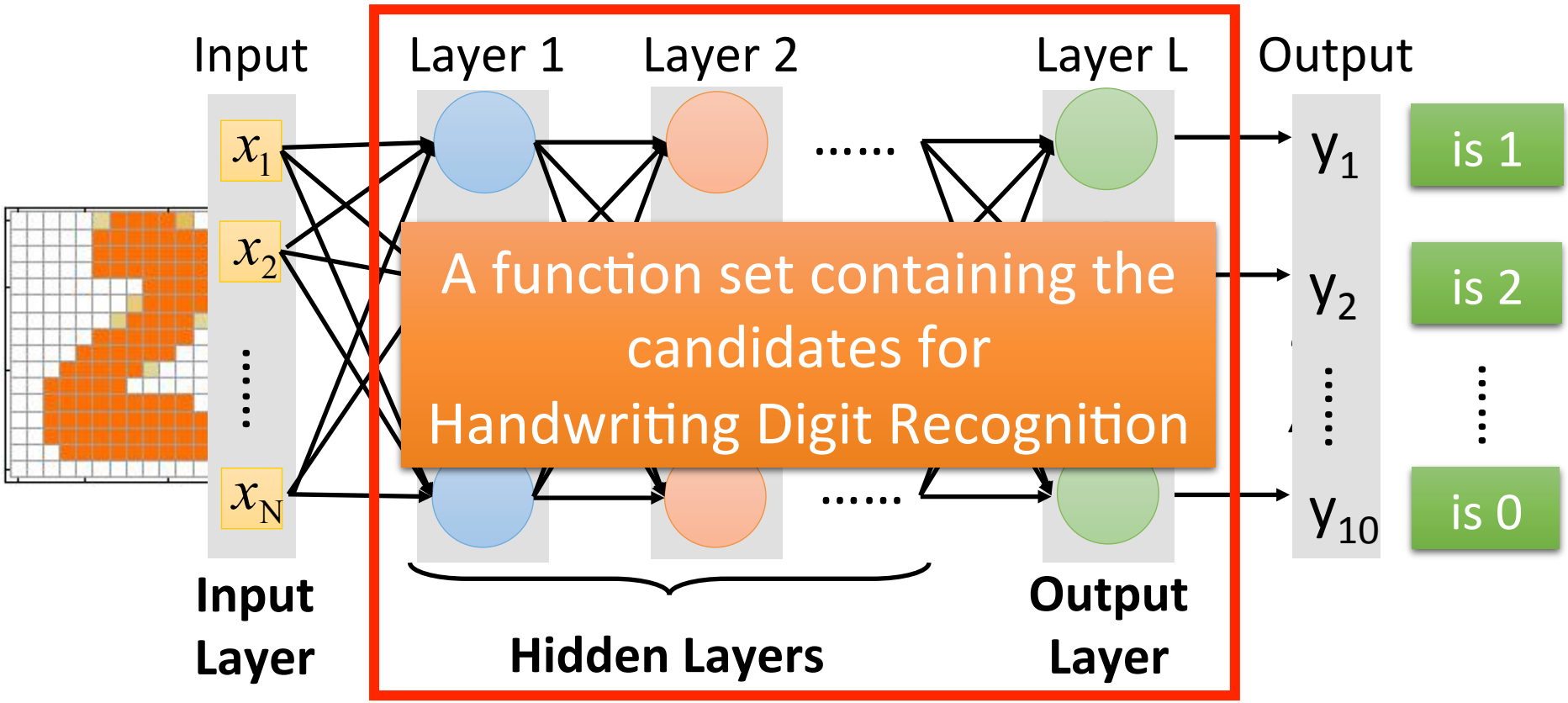
Each dimension represents the confidence of a digit.

Example Application

- Handwriting Digit Recognition

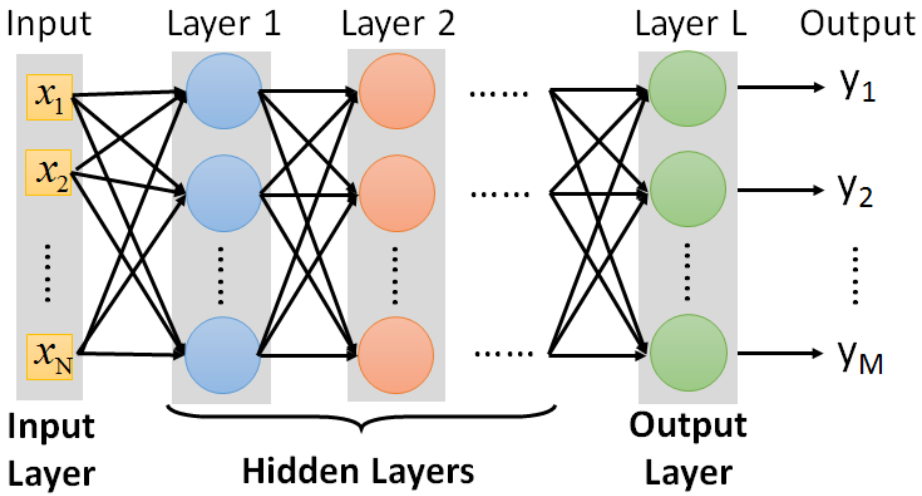


Example Application



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

FAQ



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

Trial and Error + Intuition

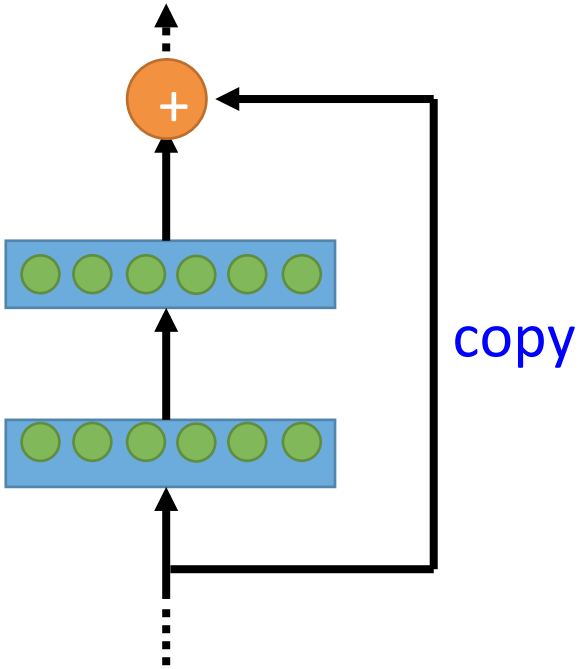
- Q: Can we design the network structure?

Convolutional Neural Network (CNN) in the next lecture

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

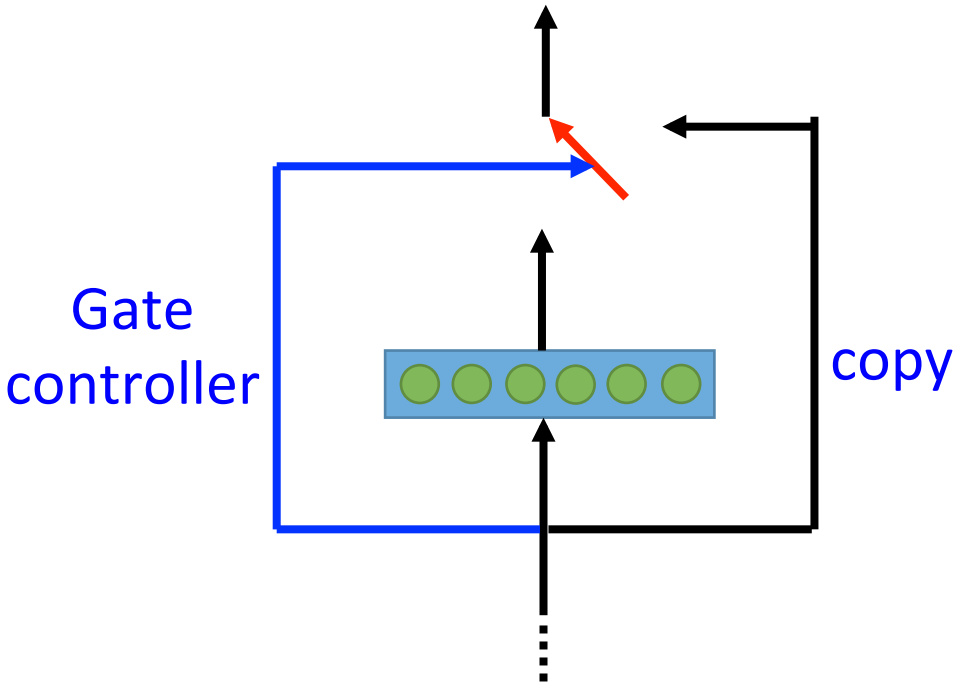
Highway Network

- **Residual Network**

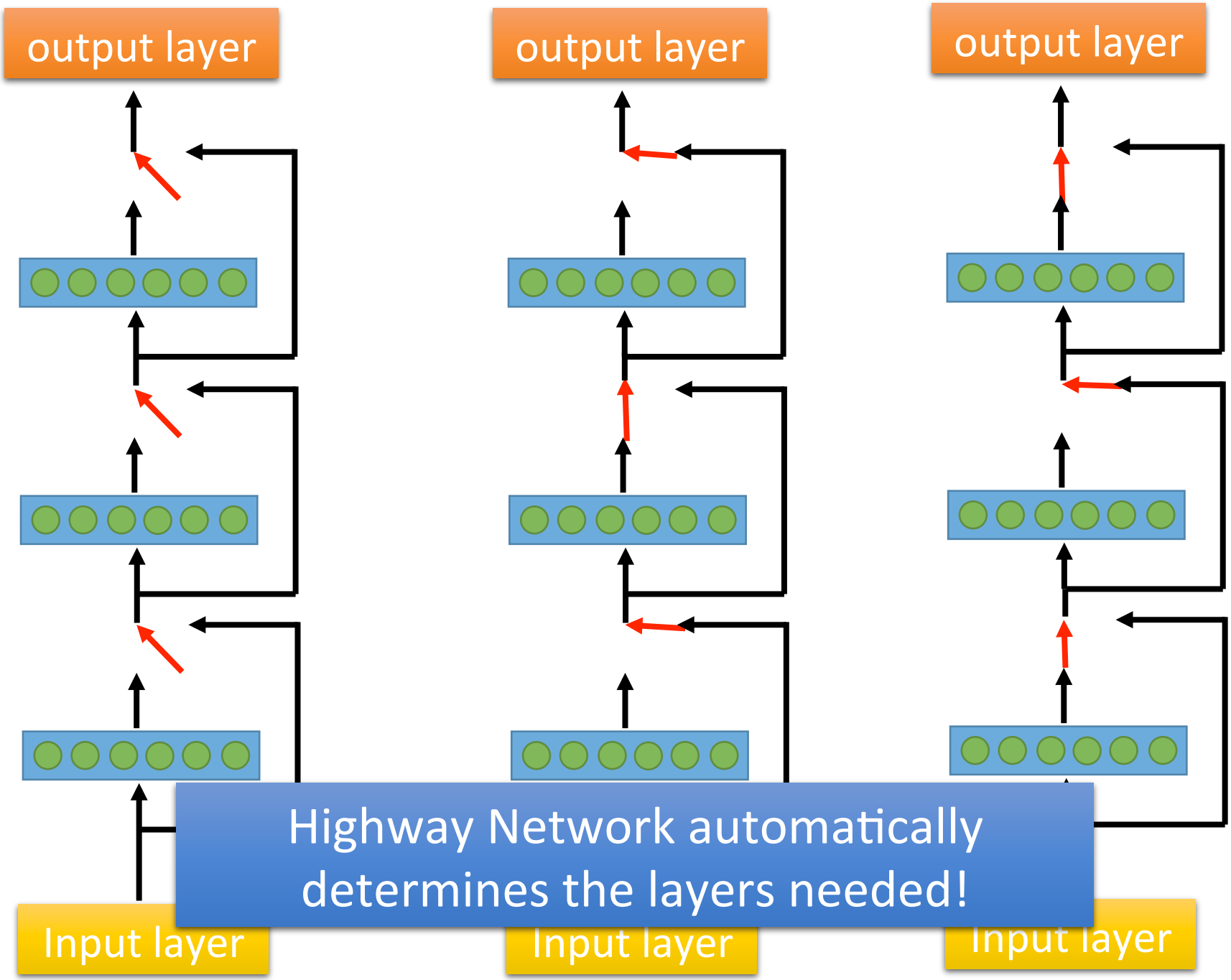


Deep Residual Learning for Image Recognition
http://arxiv.org/abs/1512.03385

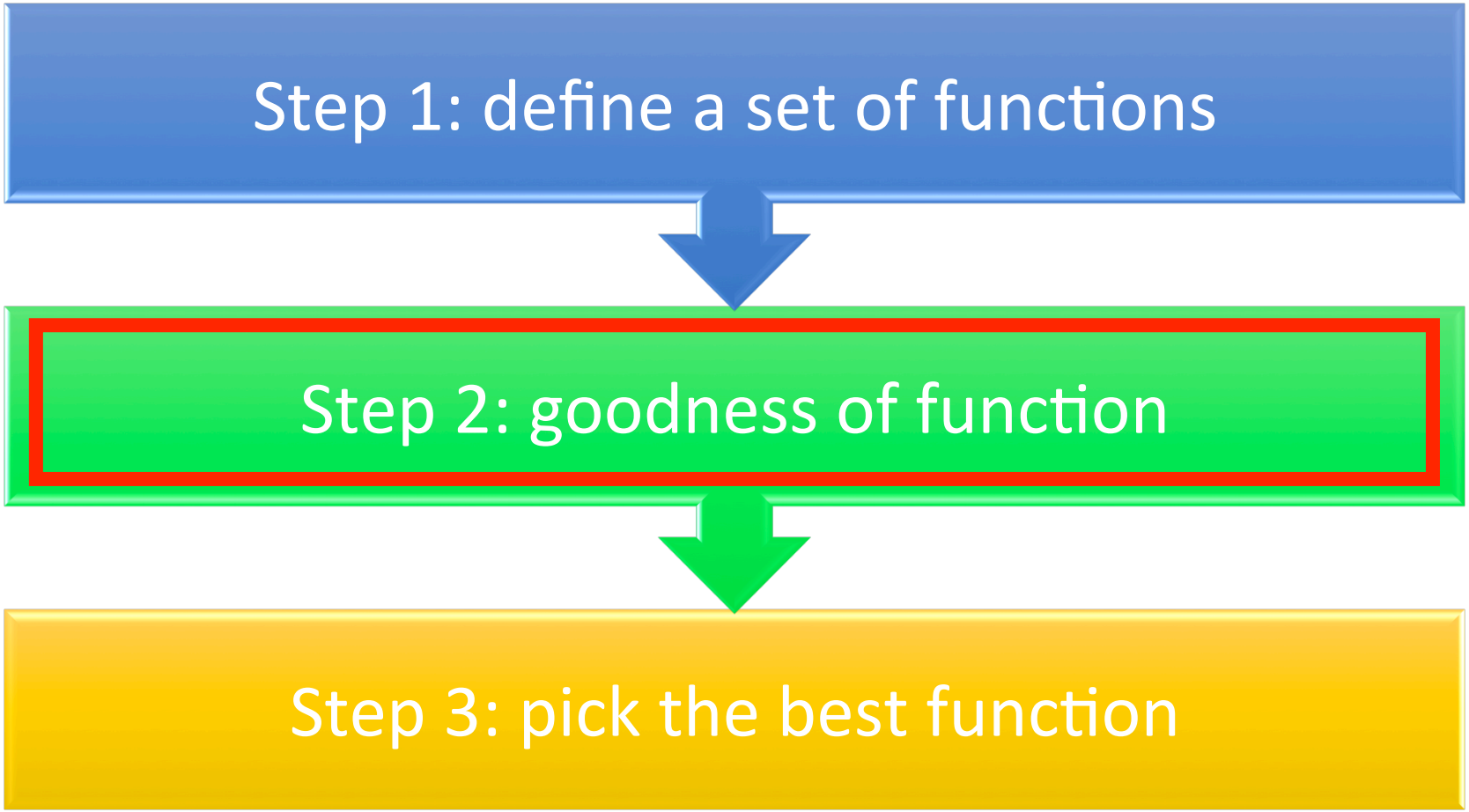
- **Highway Network**



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

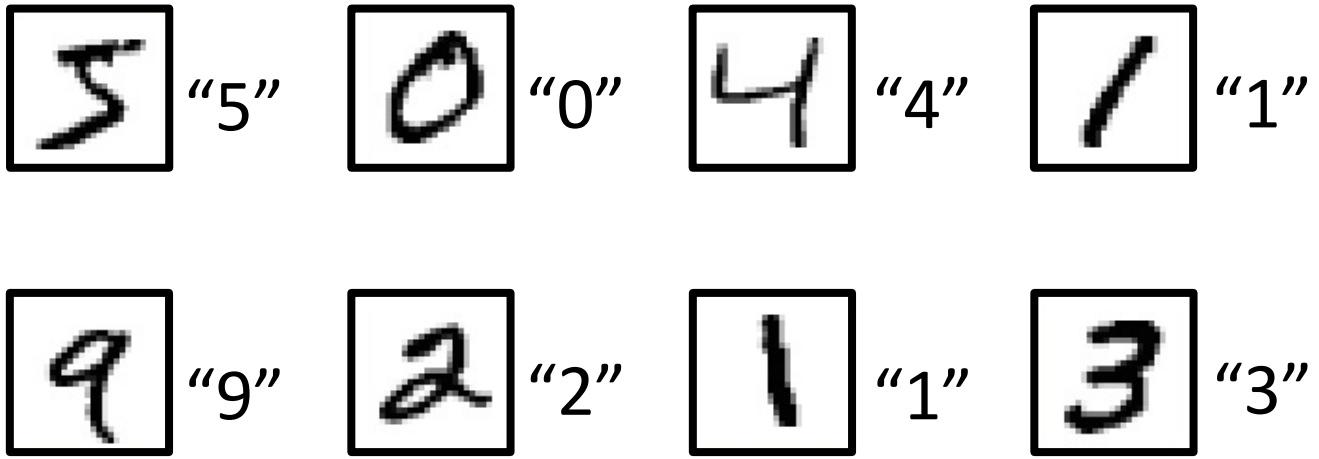

output layer
output layer
output layer
Highway Network automatically
determines the layers needed!
Input layer
Input layer
Input layer

Three Steps for Deep Learning



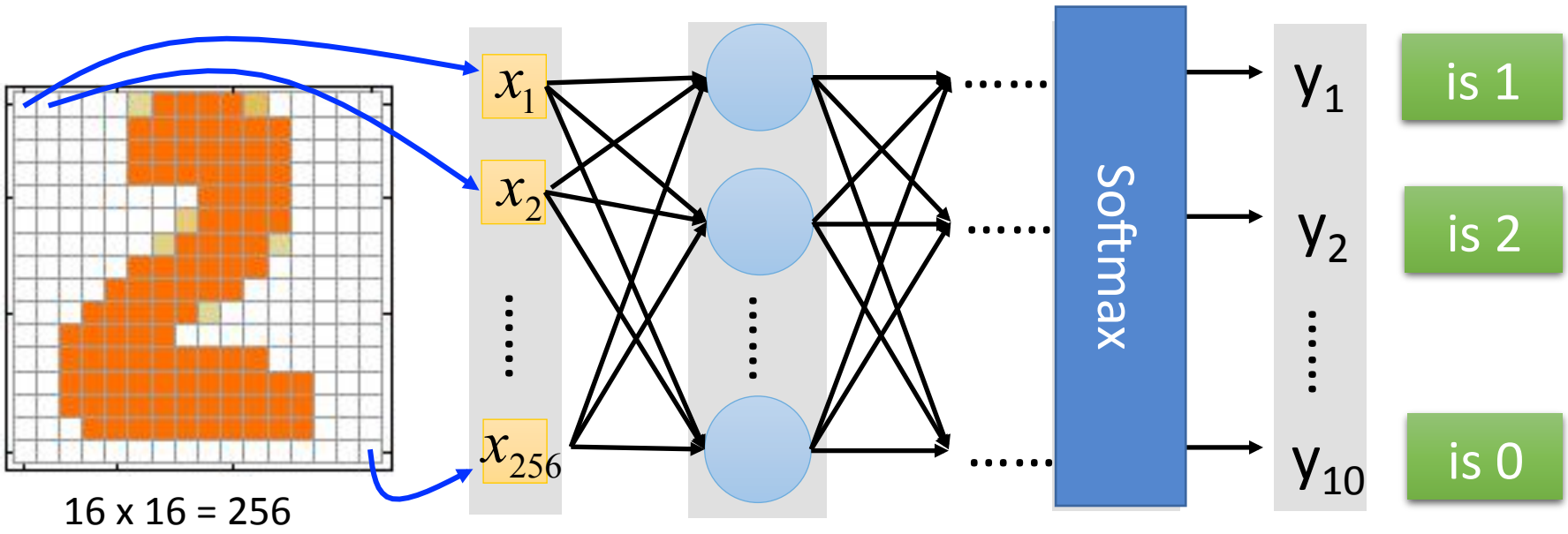
Training Data

- Preparing training data: images and their labels



The learning target is defined in the training data.

Learning Target



Ink → 1

No ink → 0

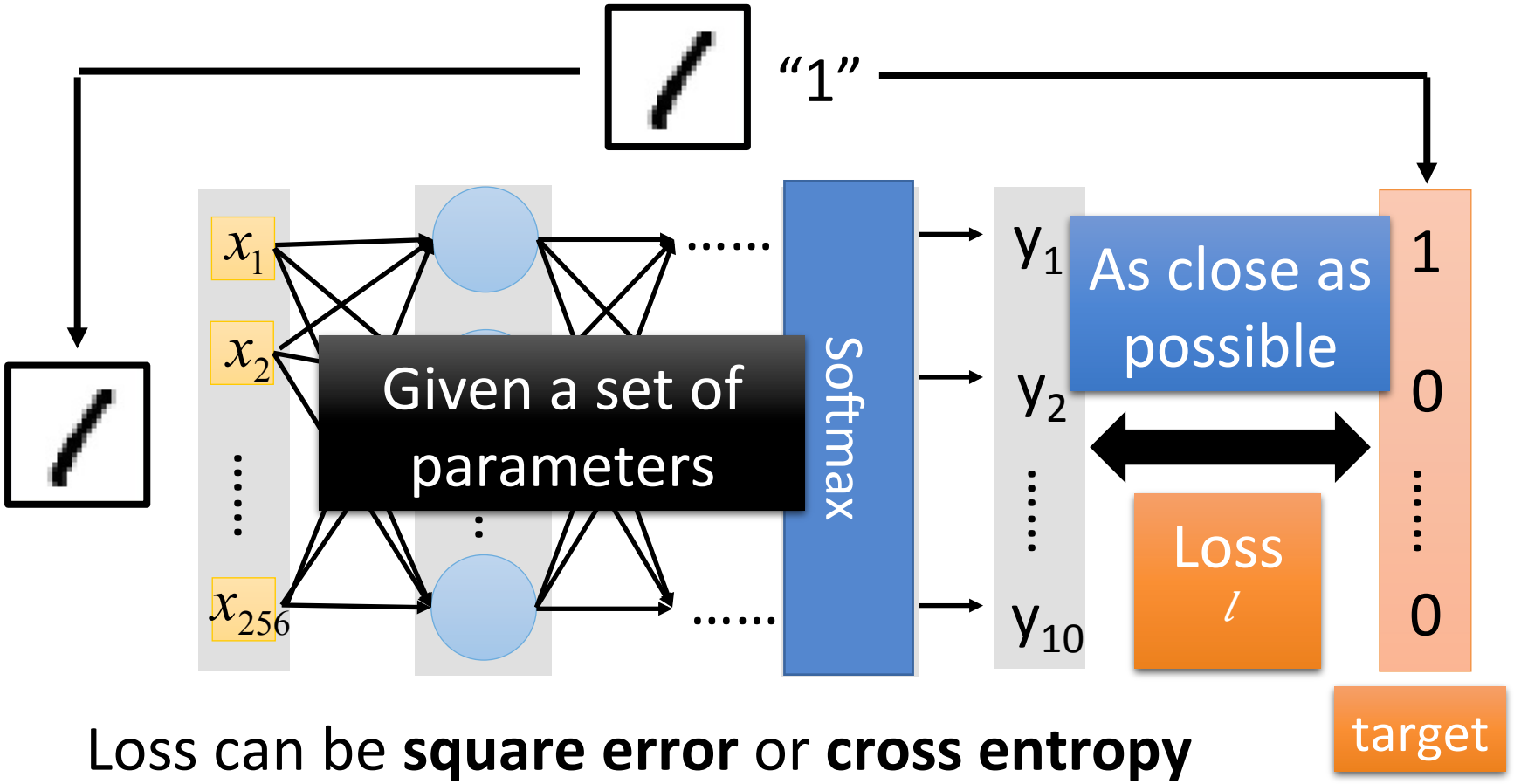
The learning target is ……

Input:  → y_1 has the maximum value

Input:  → y_2 has the maximum value

Loss

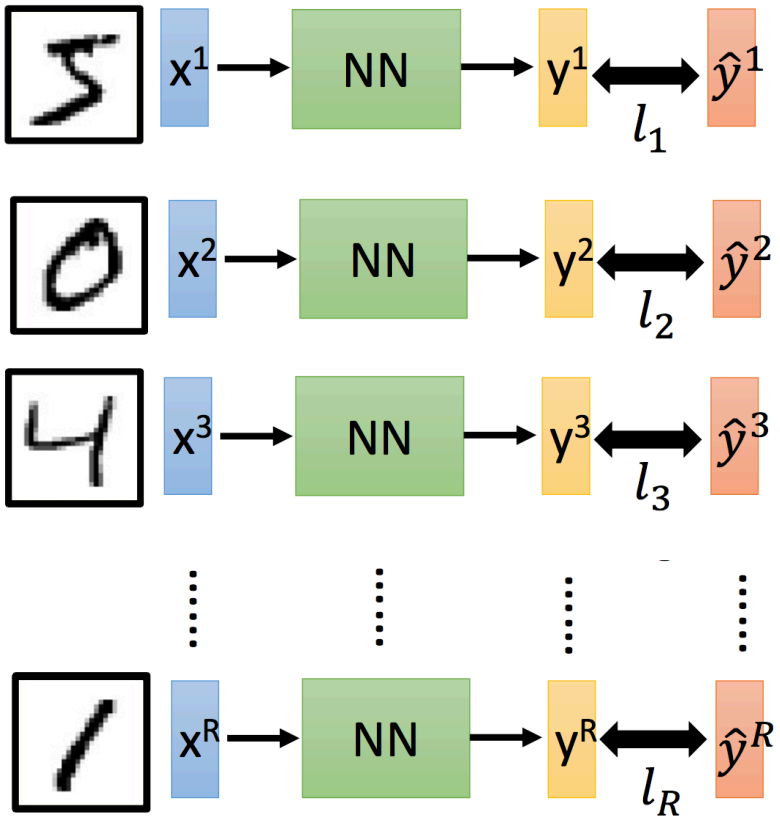
A good function should make the loss of all examples as small as possible.



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

Total Loss

For all training data …



Total Loss:

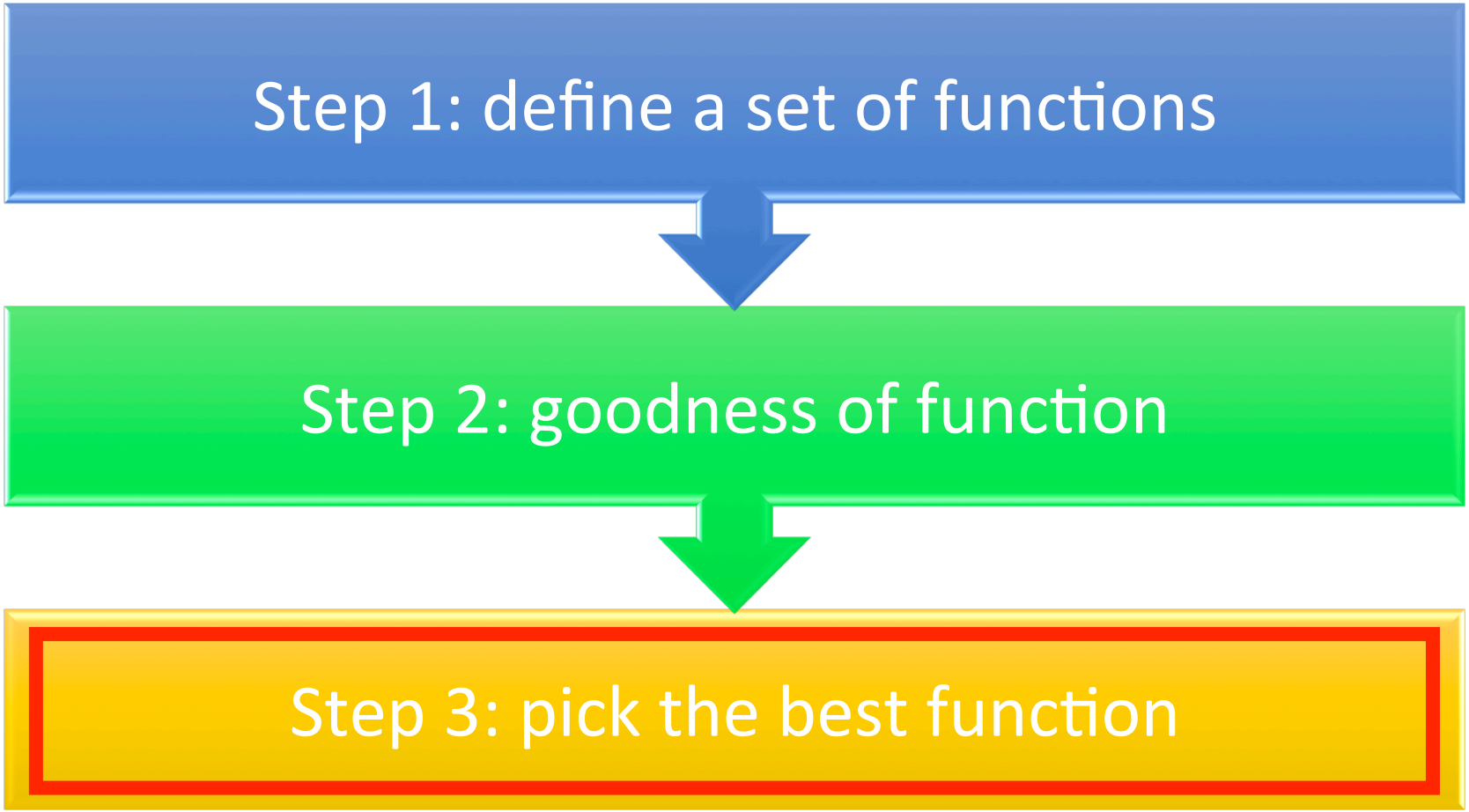
$$L = \sum_{r=1}^{R} l_r$$

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

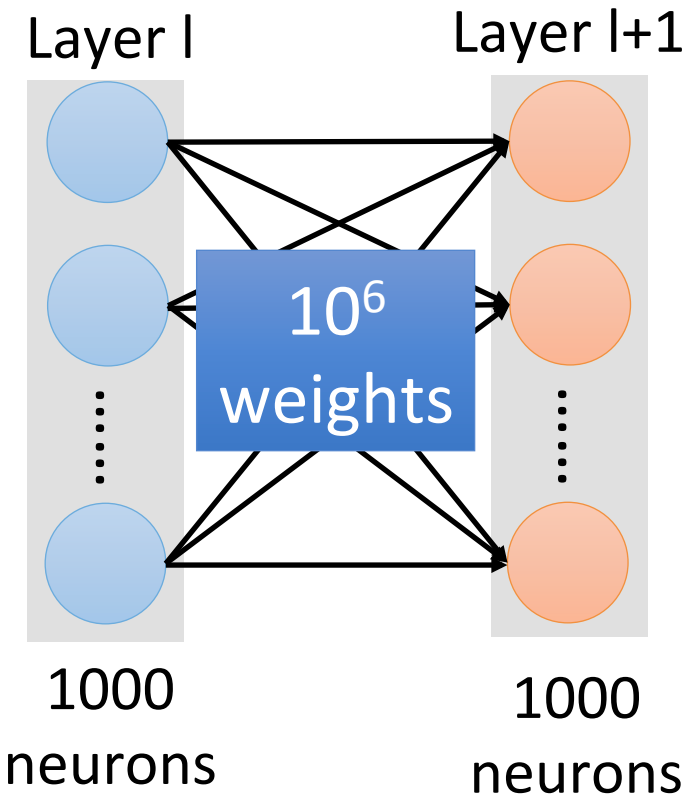
Find ***network parameters*** θ^* that minimize total loss L

Enumerate all possible values

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

Millions of parameters

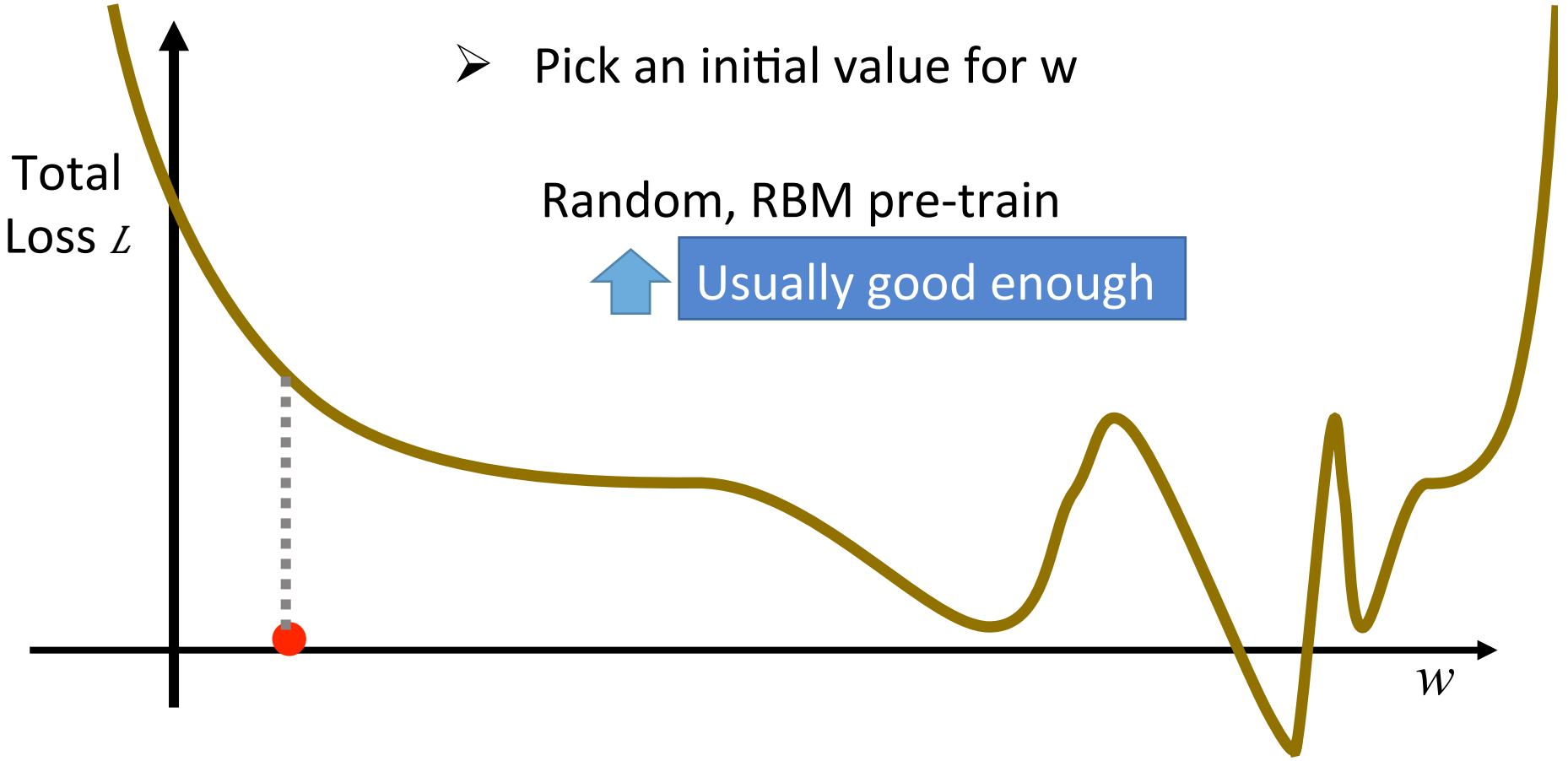
E.g. speech recognition: 8 layers and 1000 neurons each layer



Gradient Descent

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

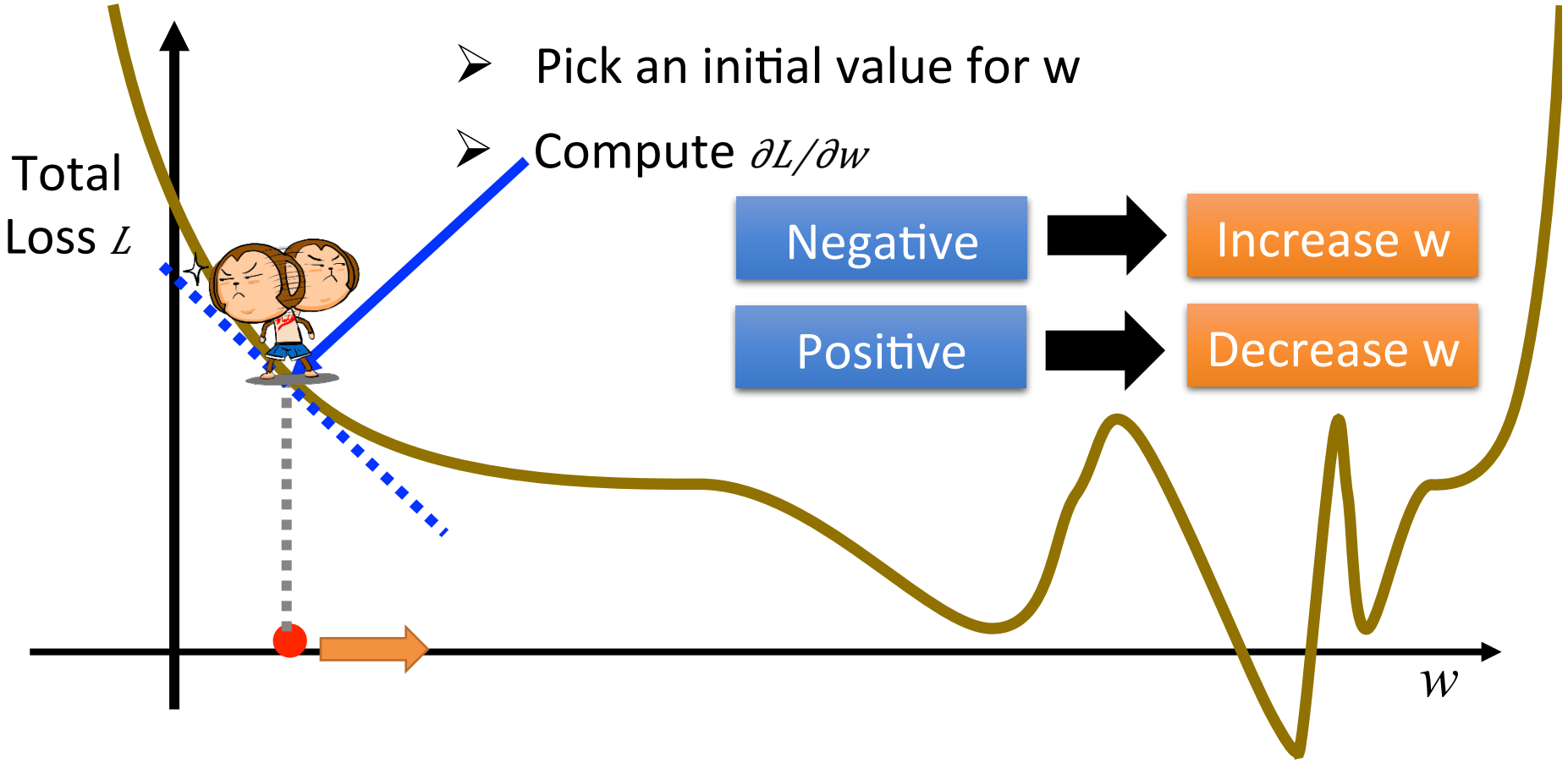
Find ***network parameters θ^**** that minimize total loss L



Gradient Descent

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

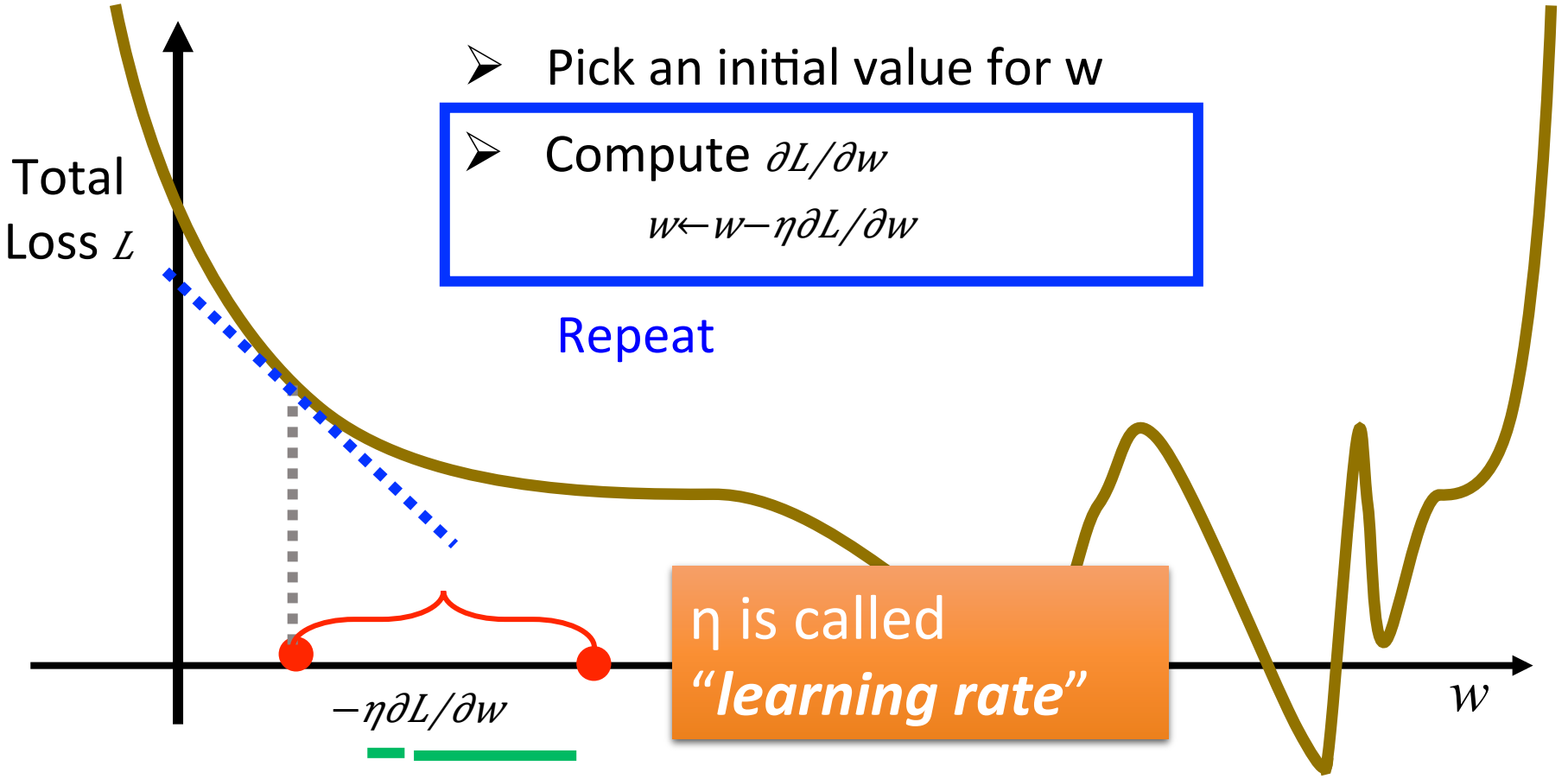
Find ***network parameters θ^**** that minimize total loss L



Gradient Descent

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

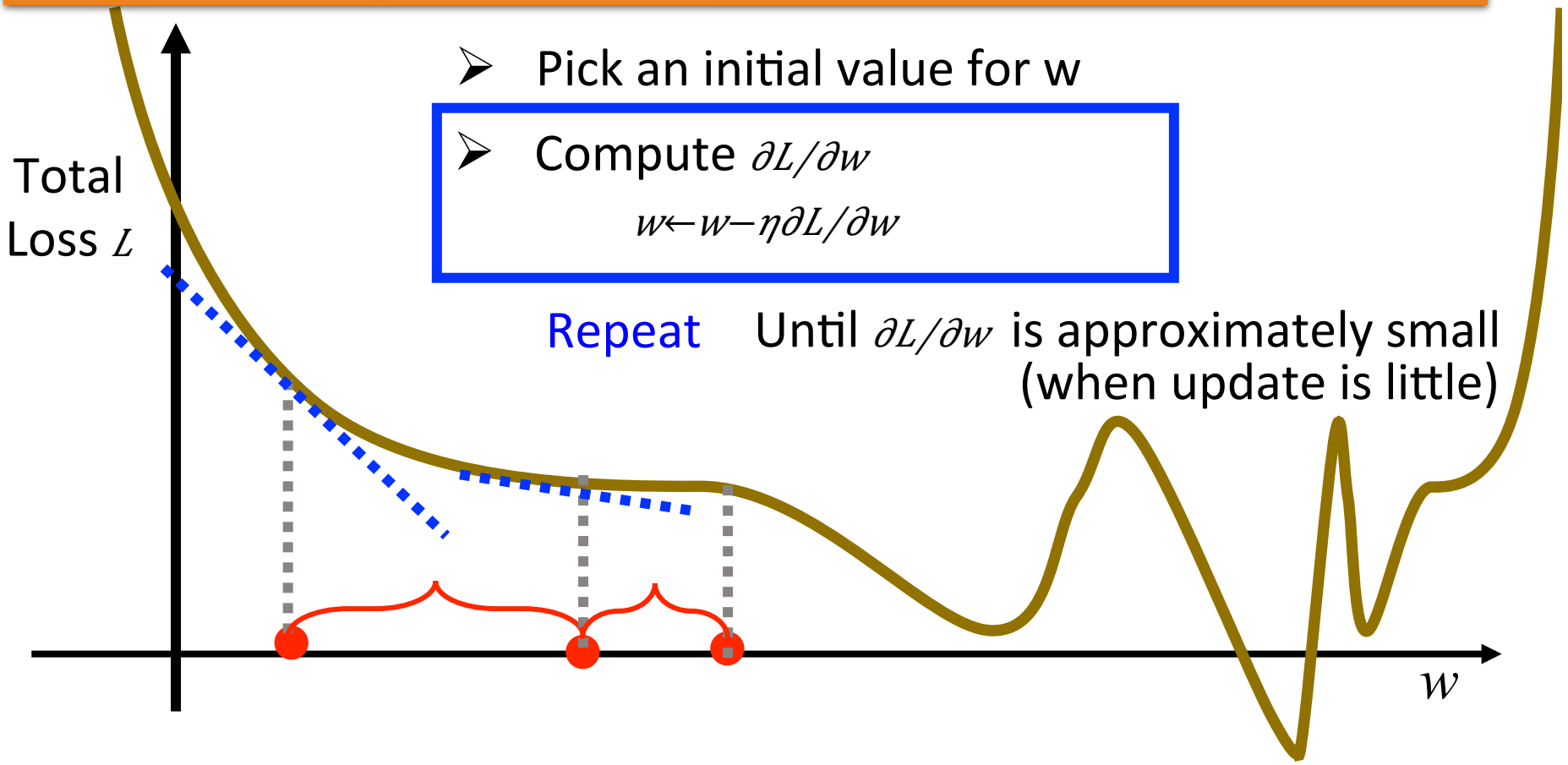
Find ***network parameters θ^**** that minimize total loss L



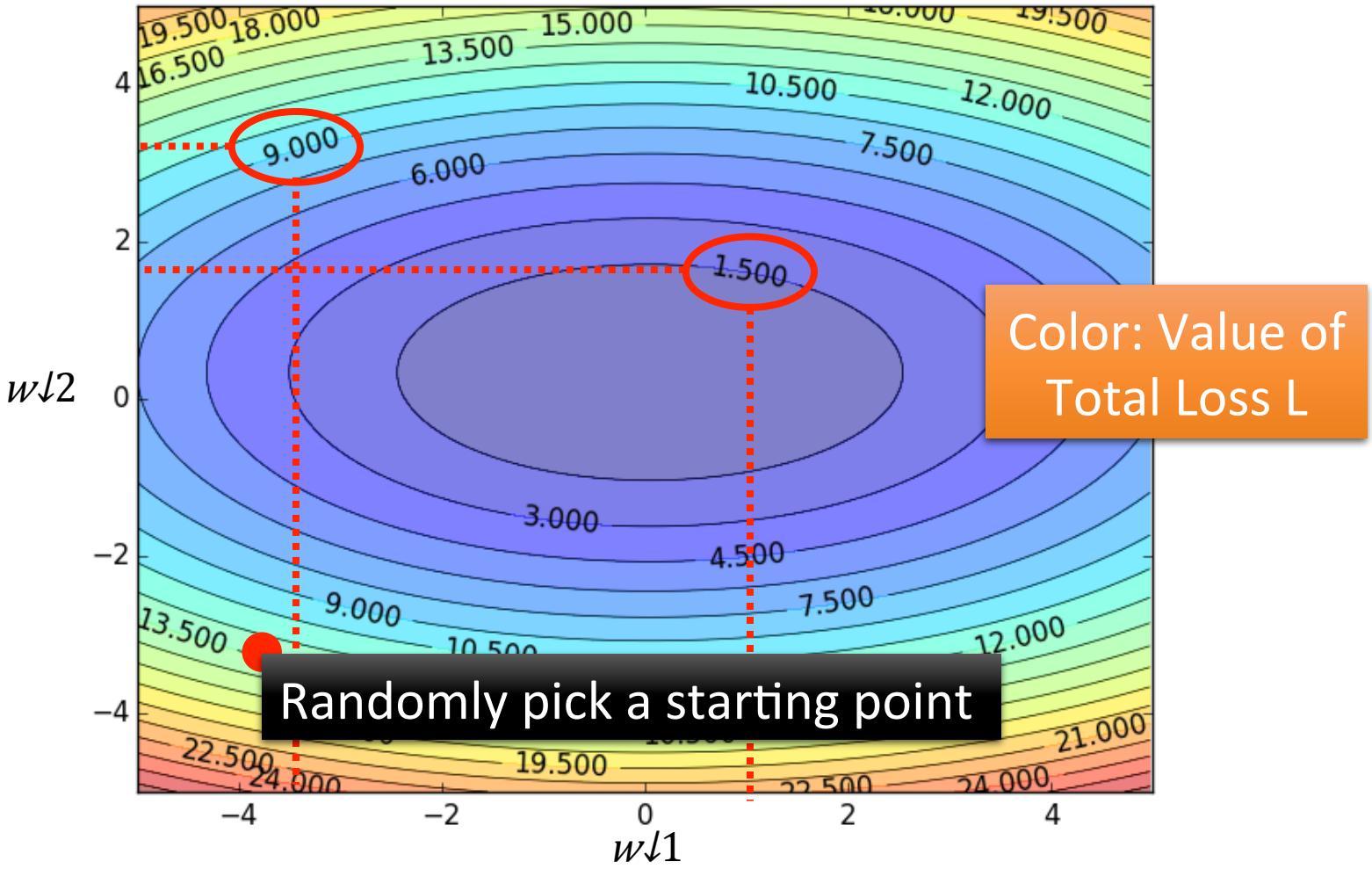
Gradient Descent

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

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

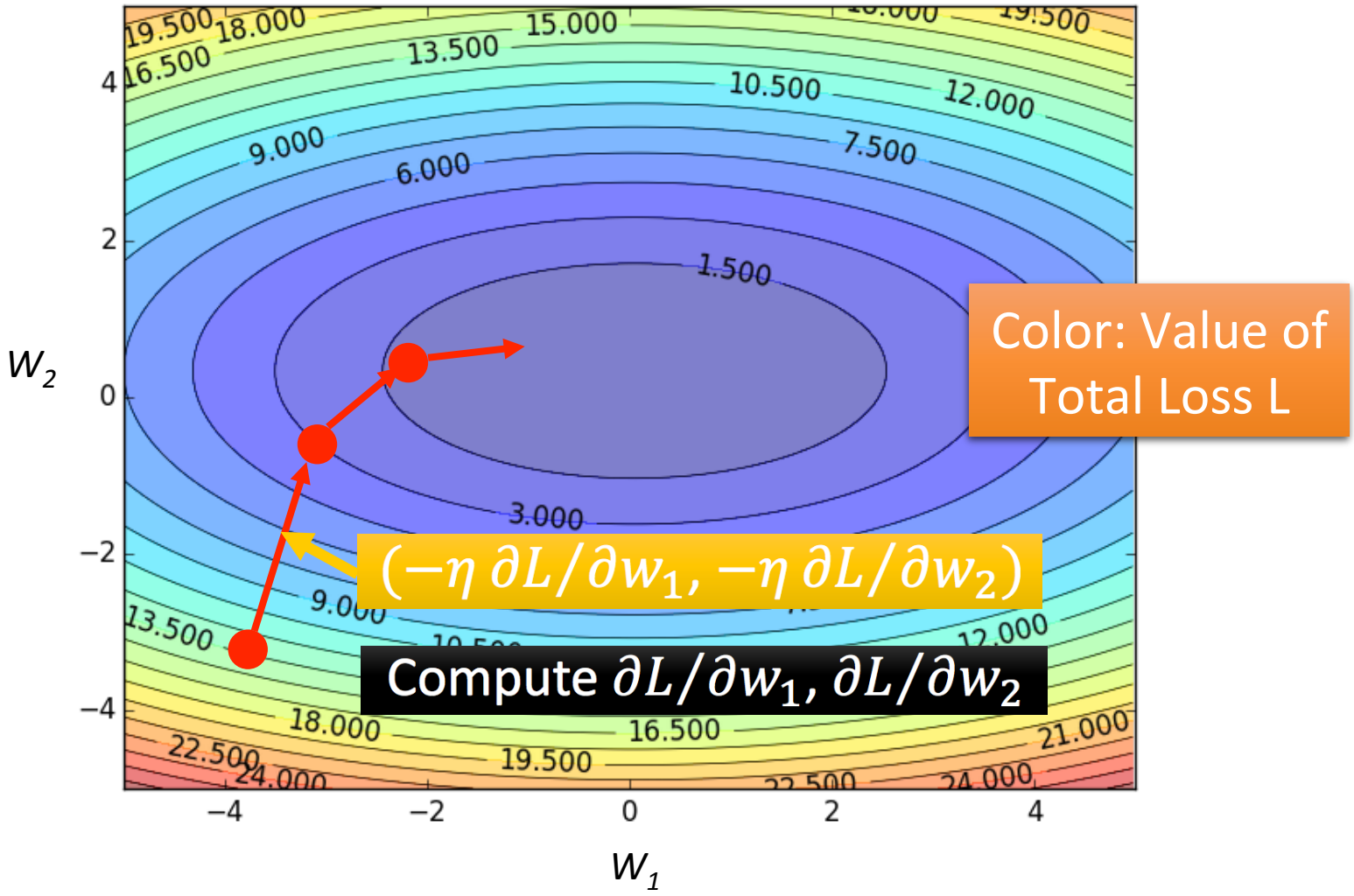


Gradient Descent

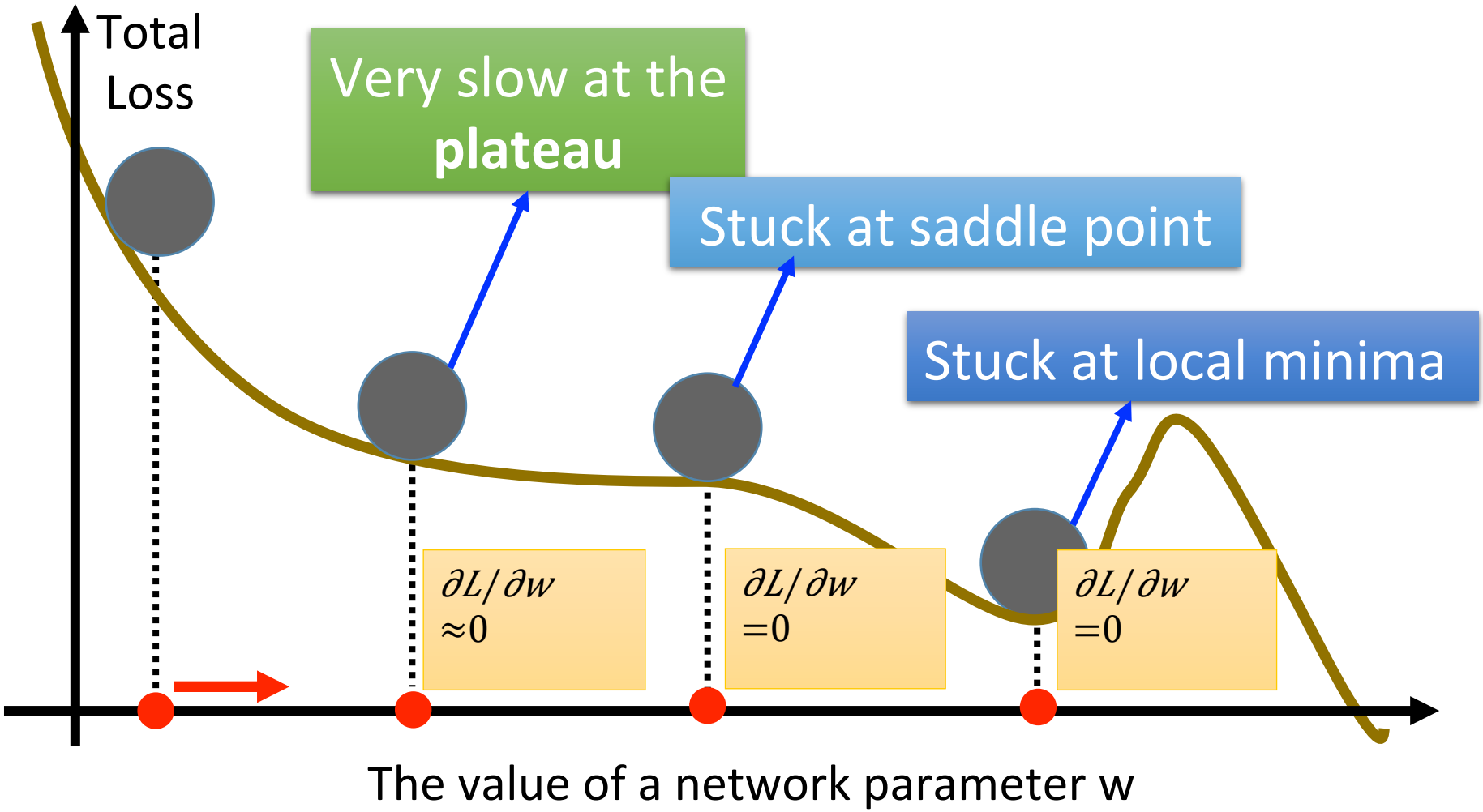


Gradient Descent

Hopefully, we would reach a minima .....

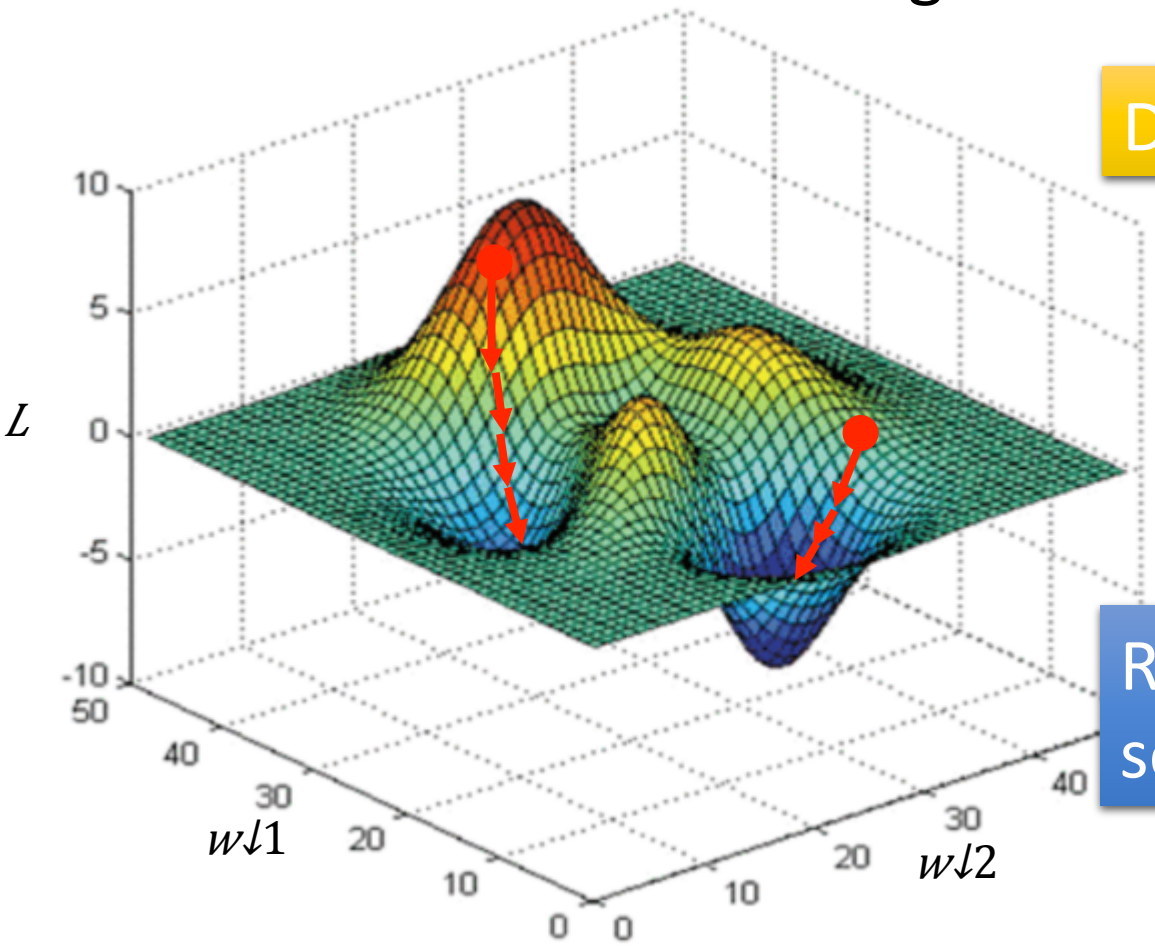


Local Minima



Local Minima

- Gradient descent never guarantee global minima



Different initial point

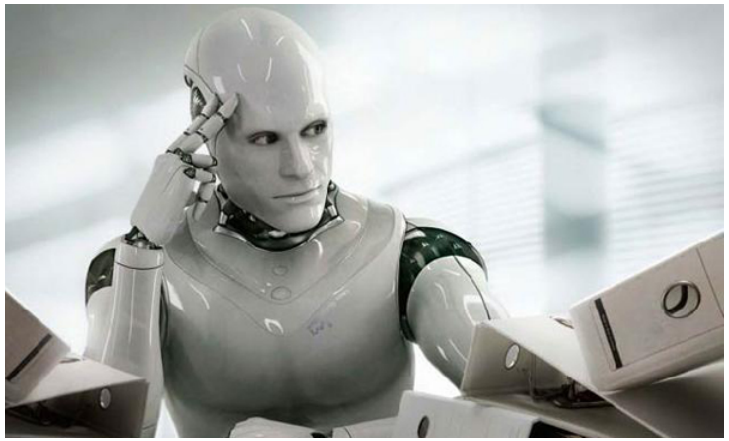
Reach different minima, so different results

Gradient Descent

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

➡ Even alpha go using this approach.

People image ……

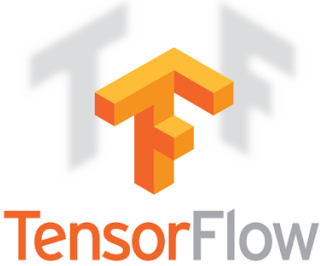


Actually …..

I hope you are not too disappointed :p

Backpropagation

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



Three Steps for Deep Learning

Deep Learning is so simple ……

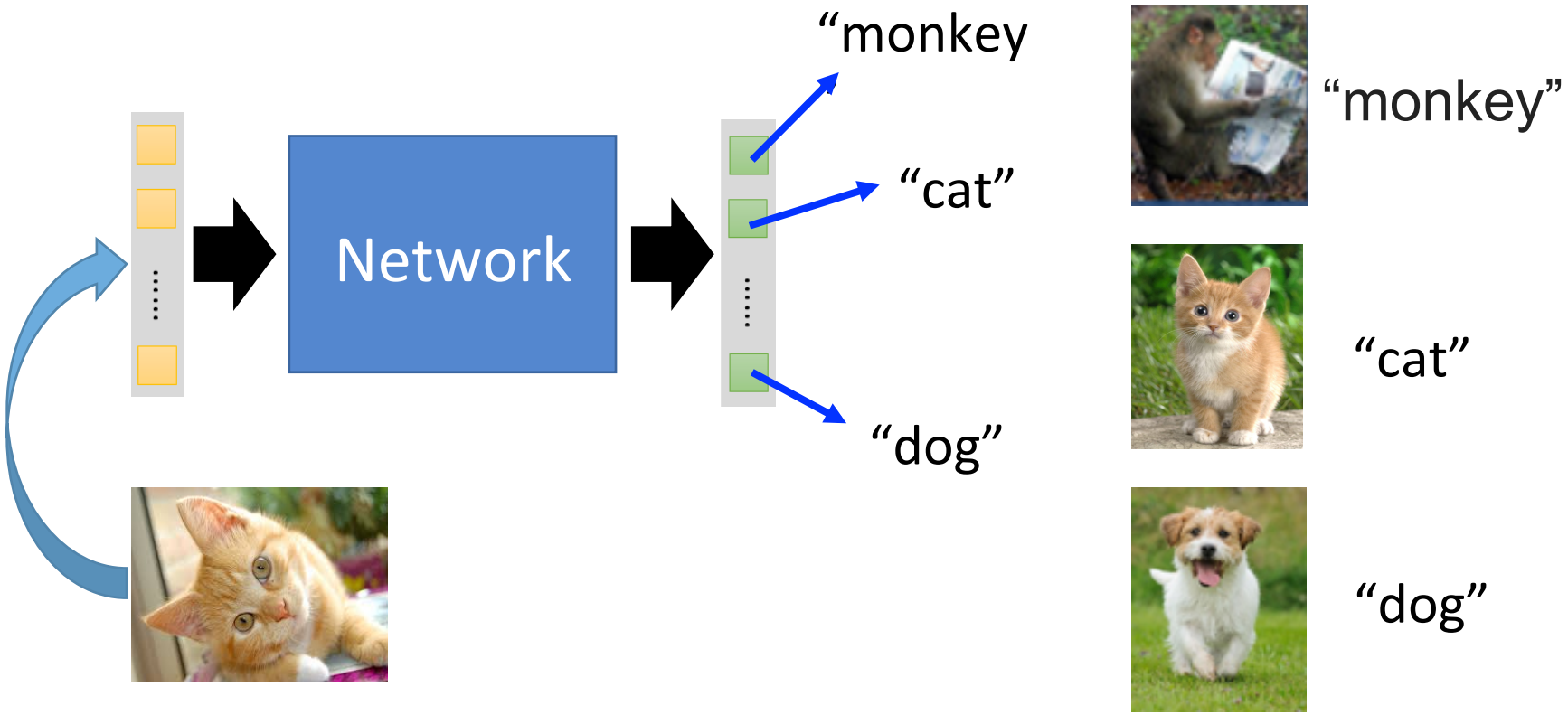
Now If you want to find a function

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

 You can use deep learning

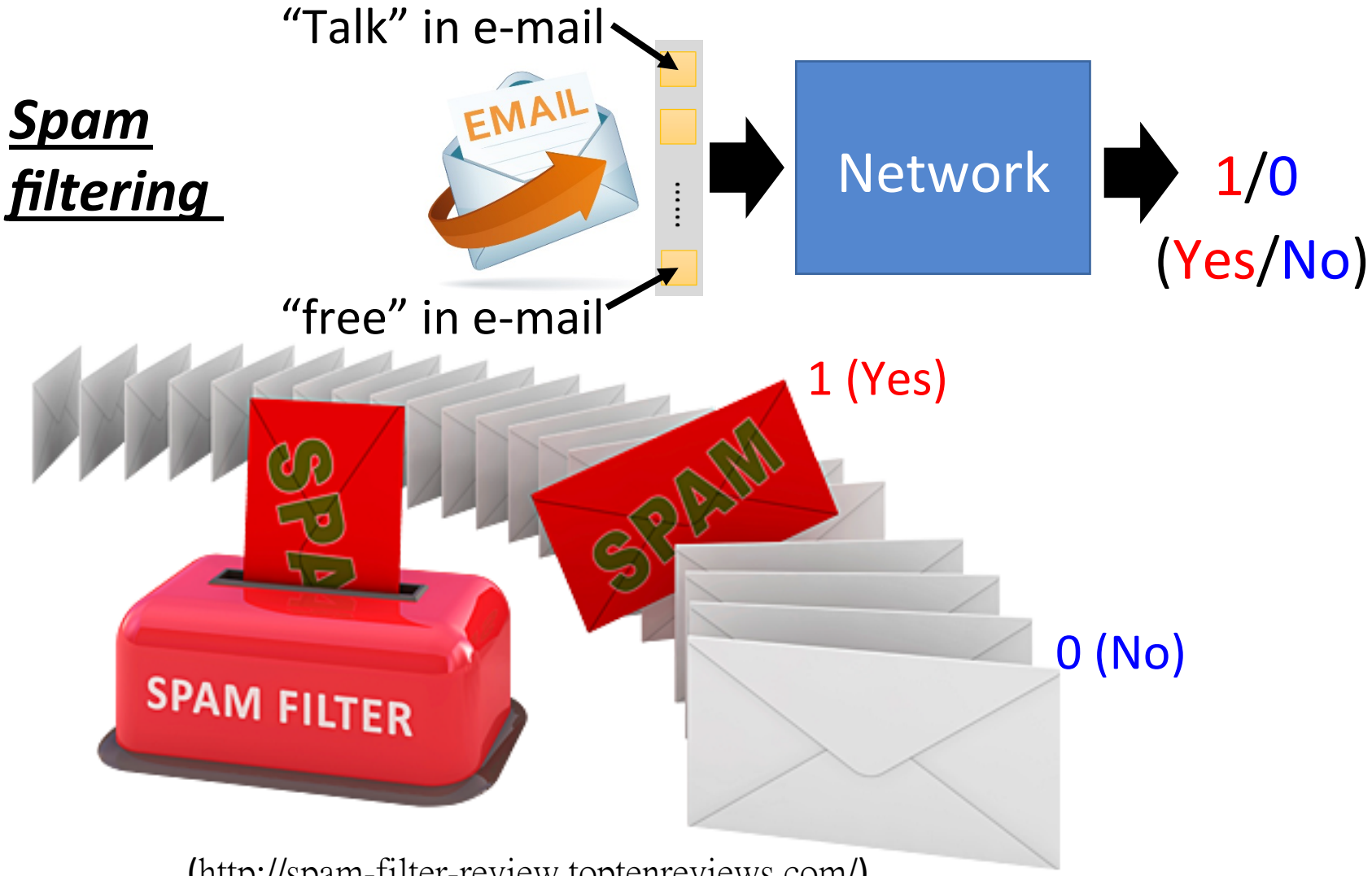
For example, you can do …….

- Image Recognition



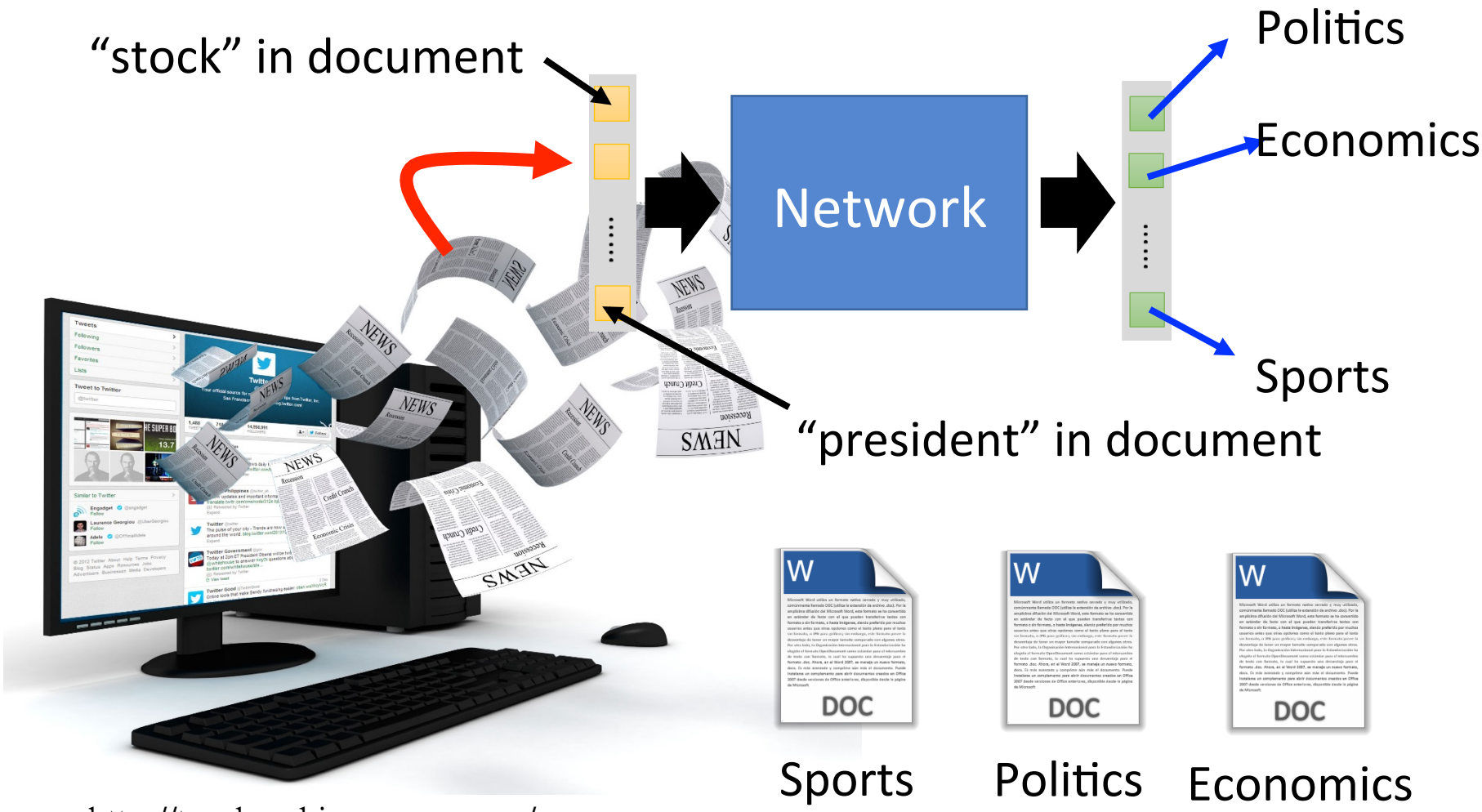
For example, you can do …….

Spam filtering



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

For example, you can do …….



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

Outline

Introduction to 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)

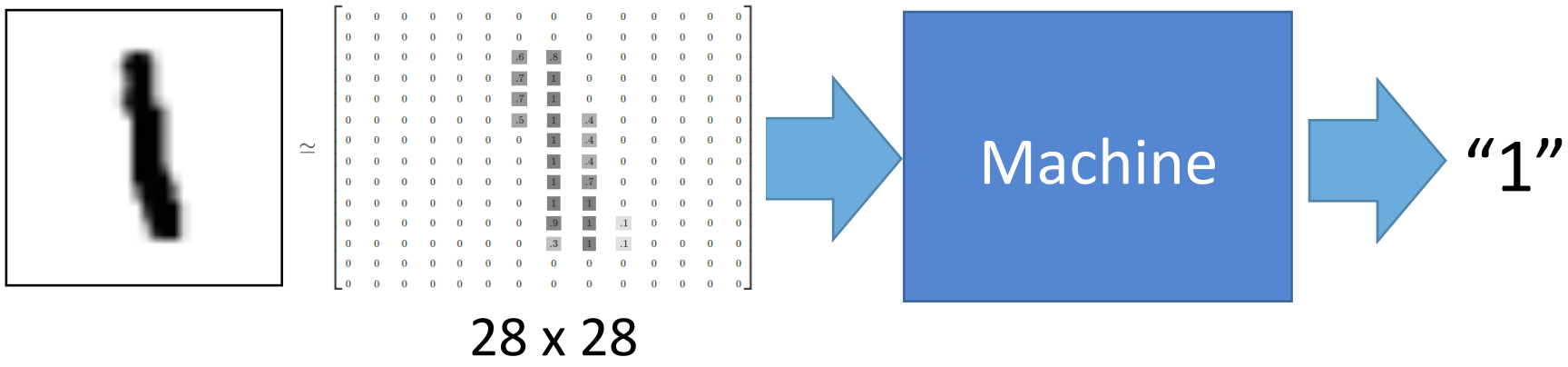
You can modify it if you can write TensorFlow or Theano

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

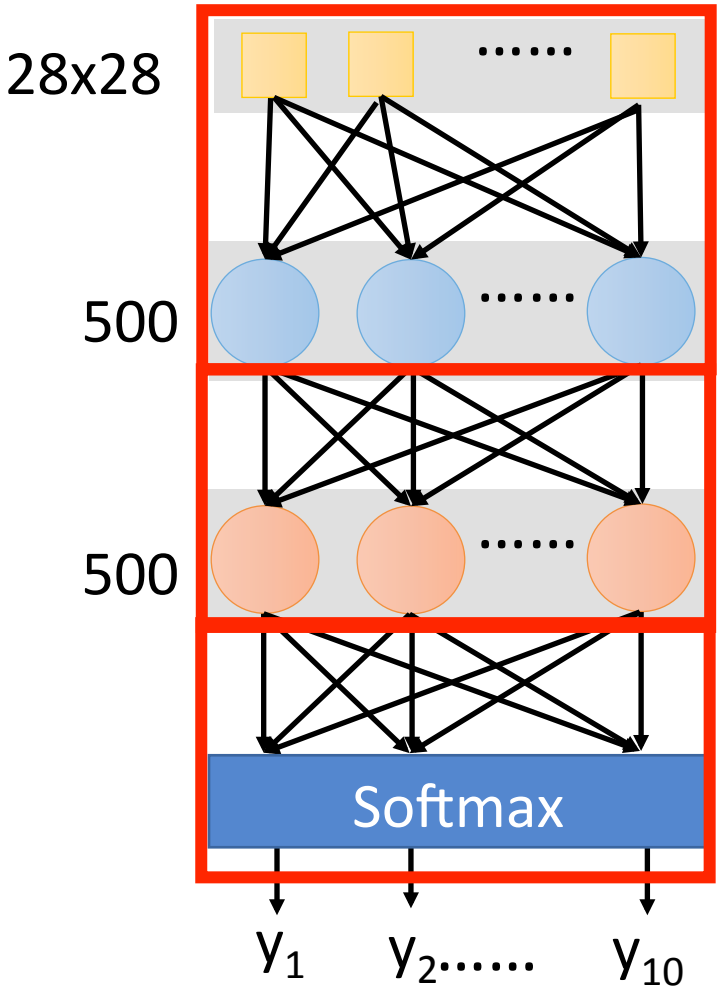
Keras provides data sets loading function: http://keras.io/datasets/

Keras

Step 1: define a set of function

Step 2: goodness of function

Step 3: pick the best function



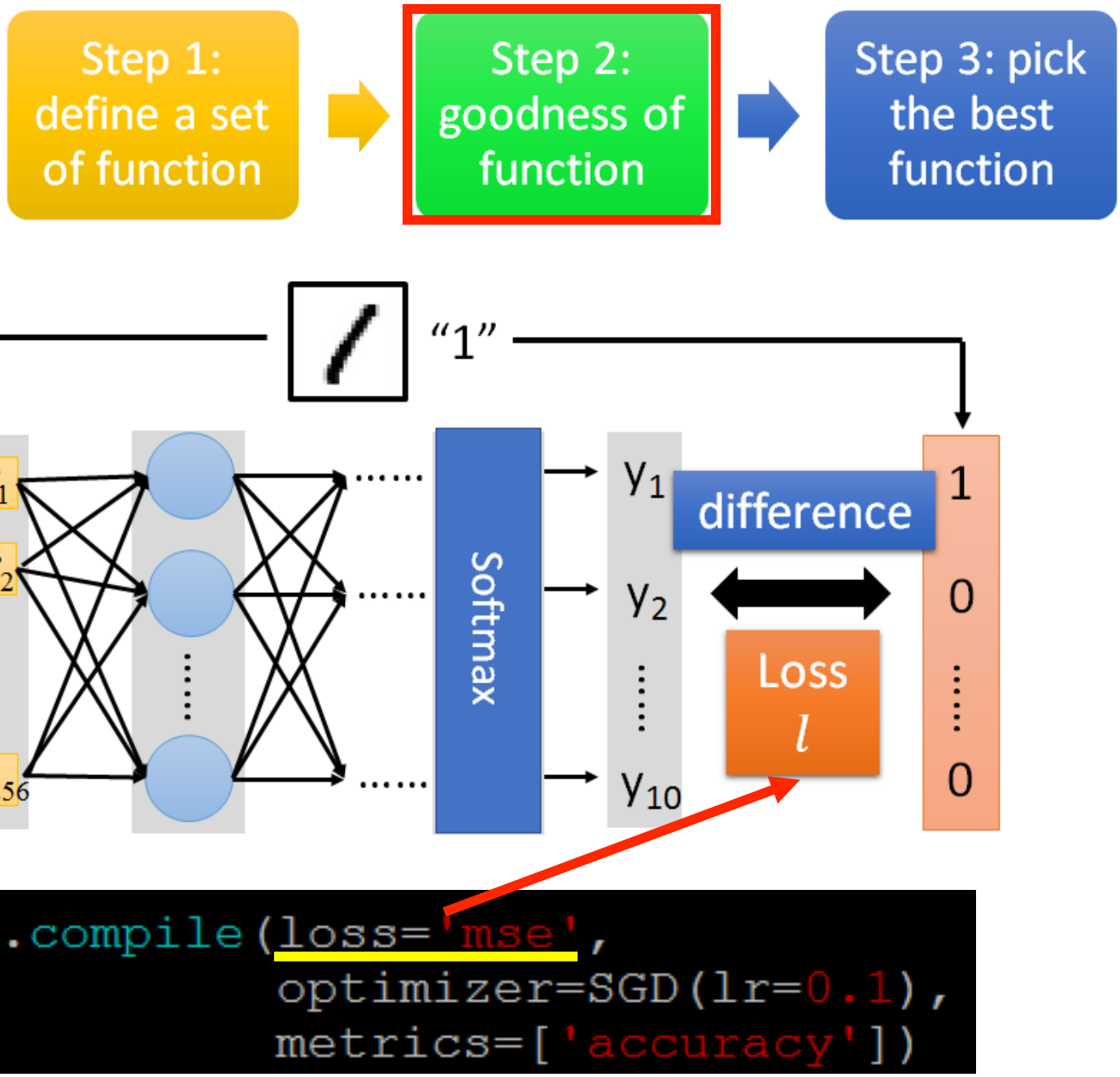

```
model = Sequential()
```

```
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
"1"
x_1
x_2
x_{256}
Softmax
y_1
y_2
y_{10}
difference
Loss l
1
0
0
model.compile(loss='mse',
optimizer=SGD(lr=0.1),
metrics=['accuracy'])

Keras

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)

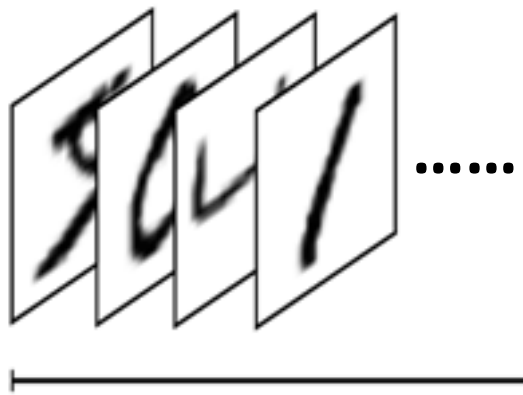
Keras

Step 3.2: Find the optimal network parameters

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

numpy array

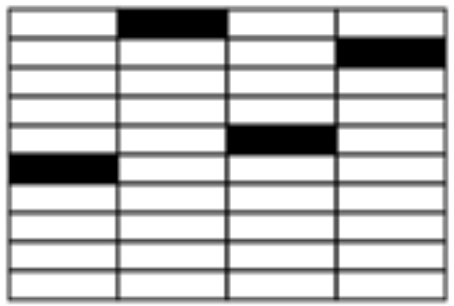
28 x 28 =784



Number of training examples

numpy array

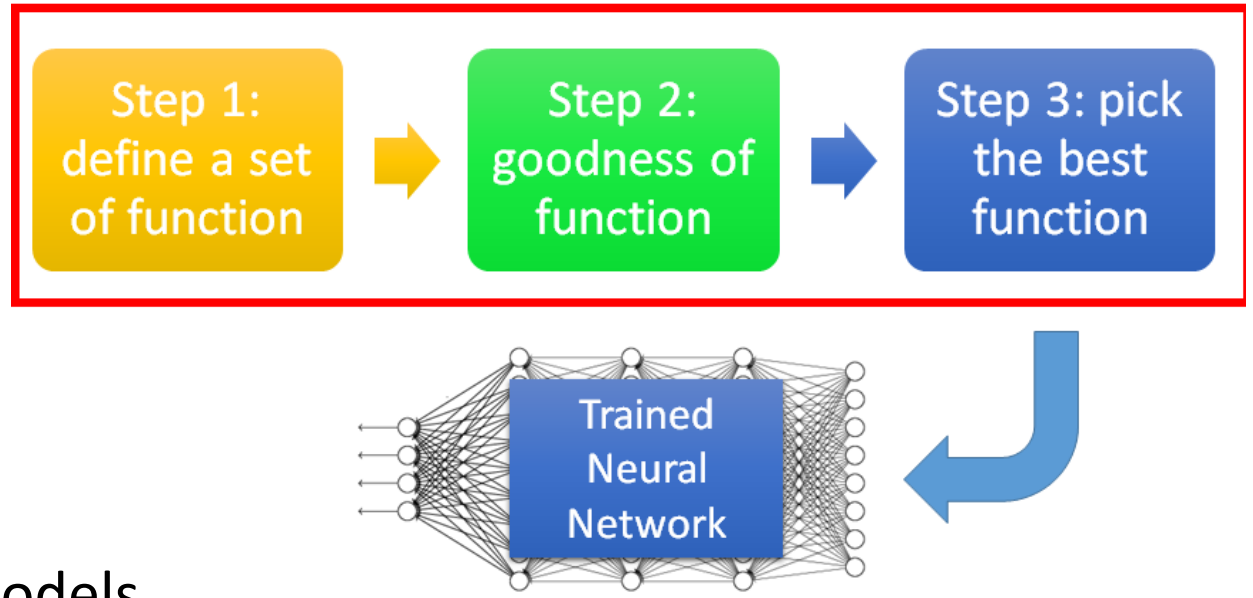
10



Number of training examples

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

Keras



Save and load models

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

How to use the neural network (testing):

case 1:

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

case 2:

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

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"

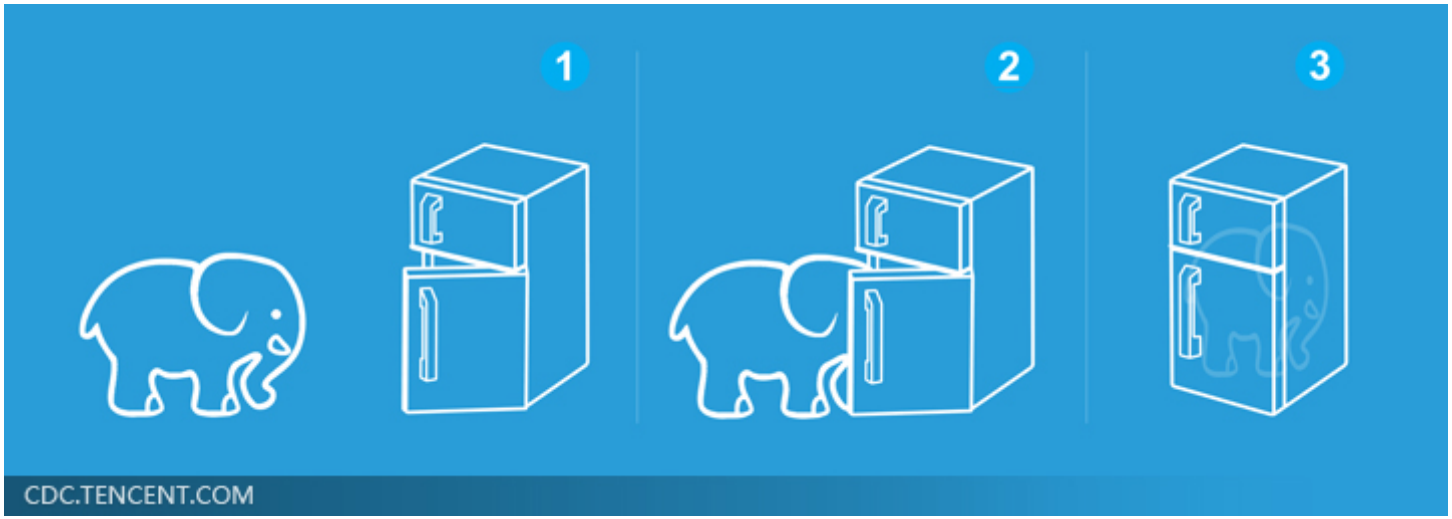
Three Steps for Deep Learning

Step 1: define a set of functions

Step 2: goodness of function

Step 3: pick the best function

Deep Learning is so simple ……



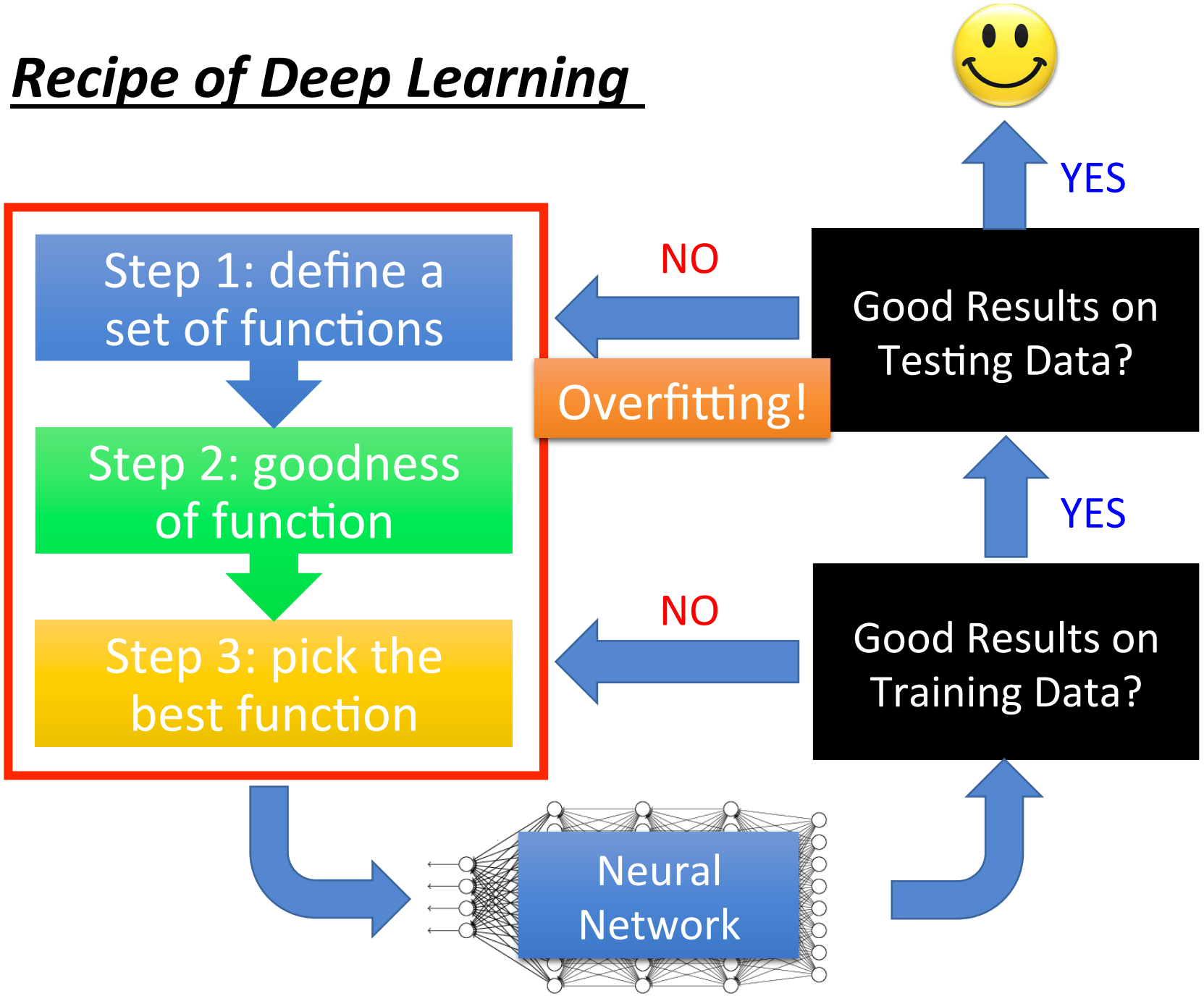
Outline

Introduction to Deep Learning

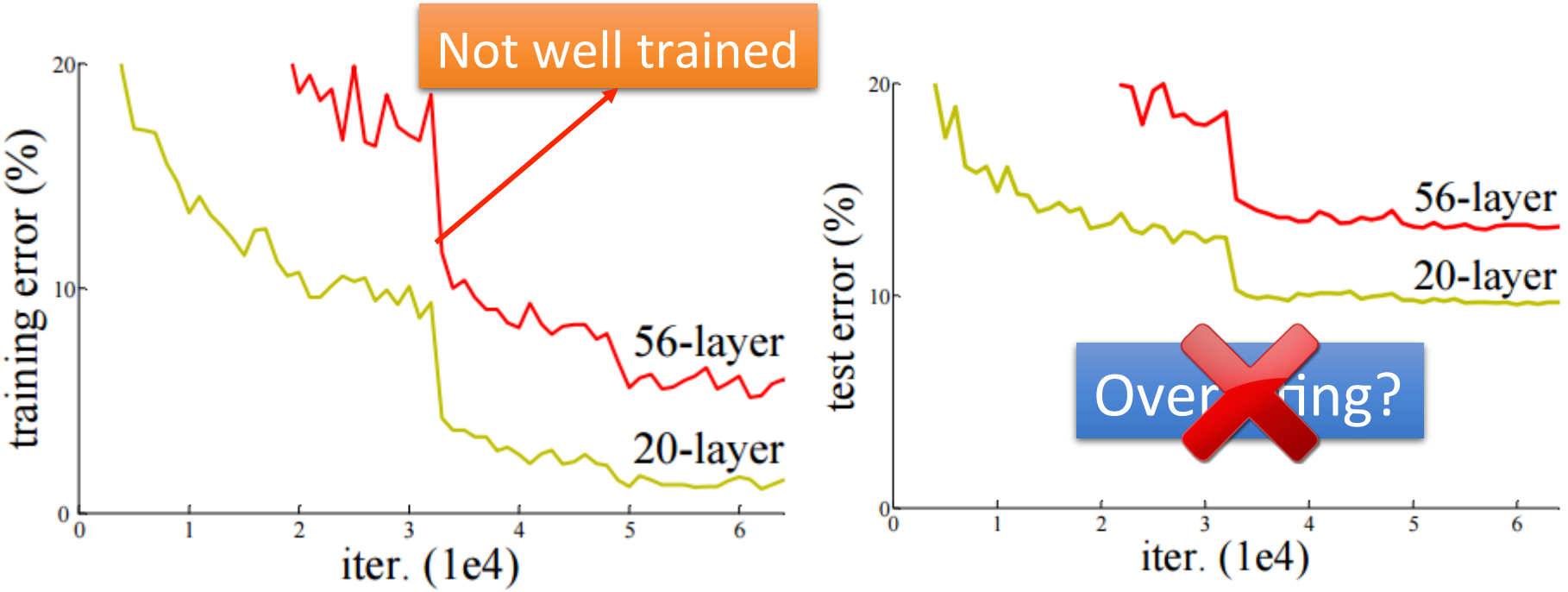
"Hello World" for Deep Learning

Tips for Deep Learning

Recipe of Deep Learning



Do not always blame Overfitting

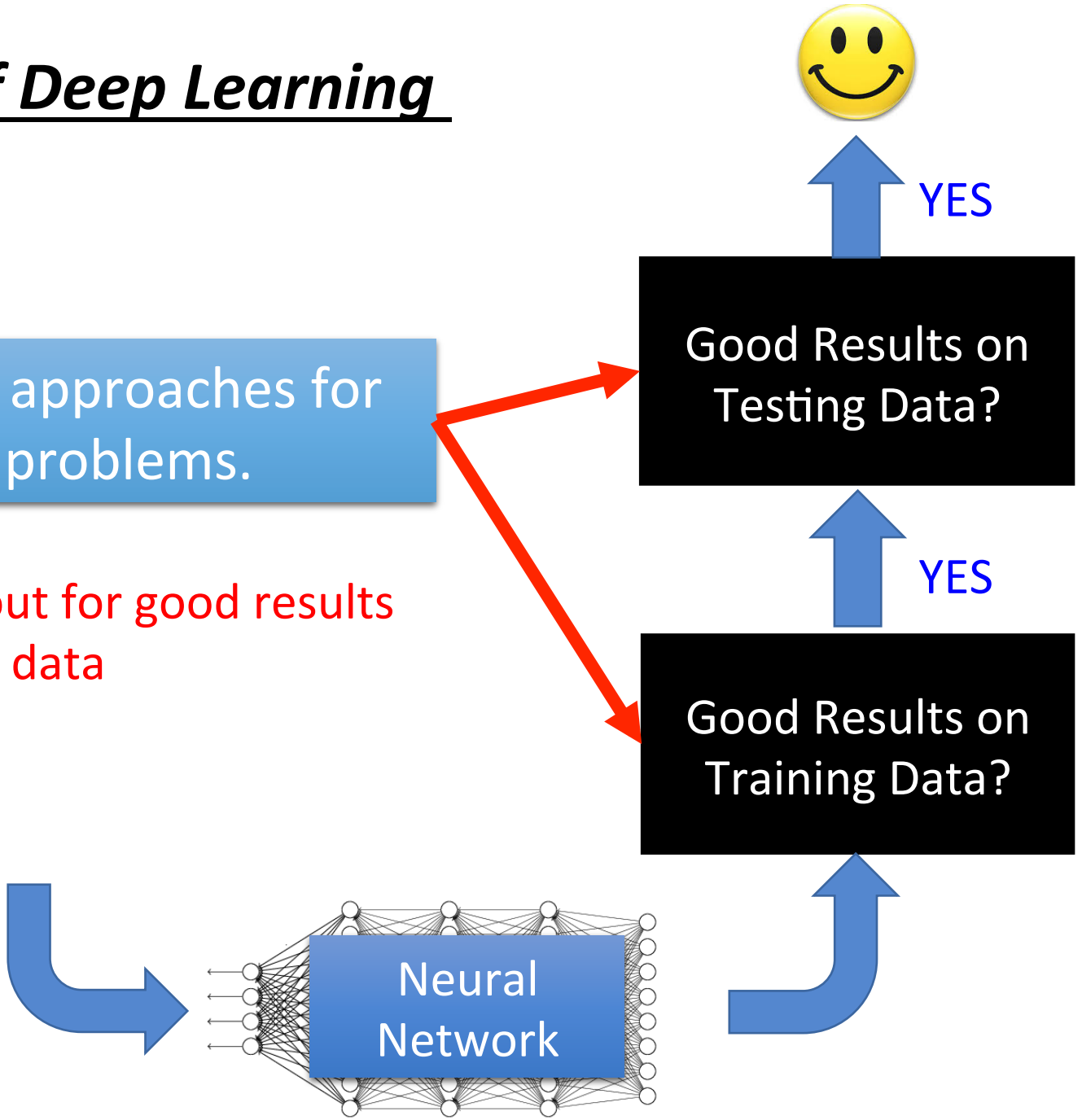


Training Data

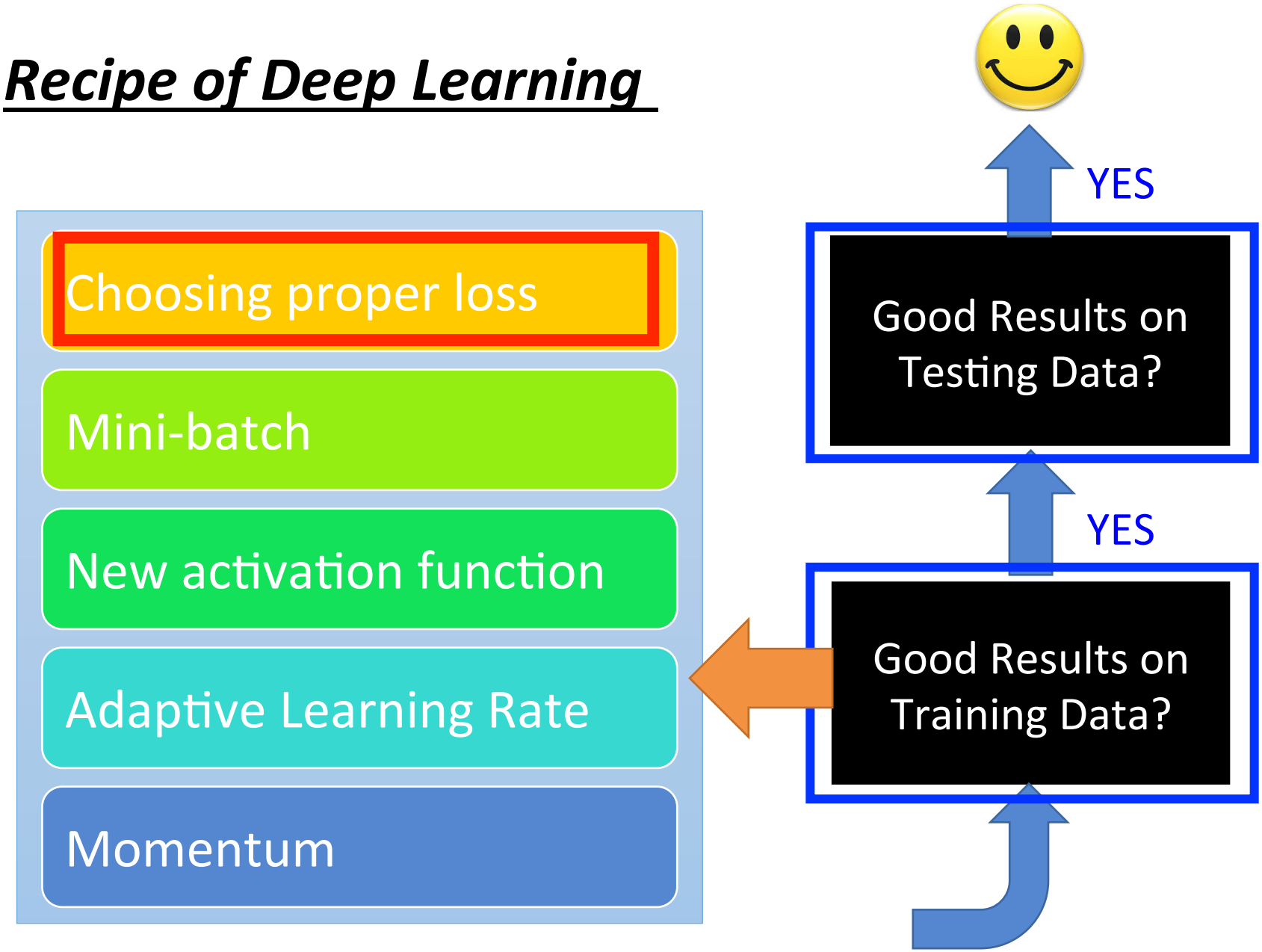
Testing Data

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

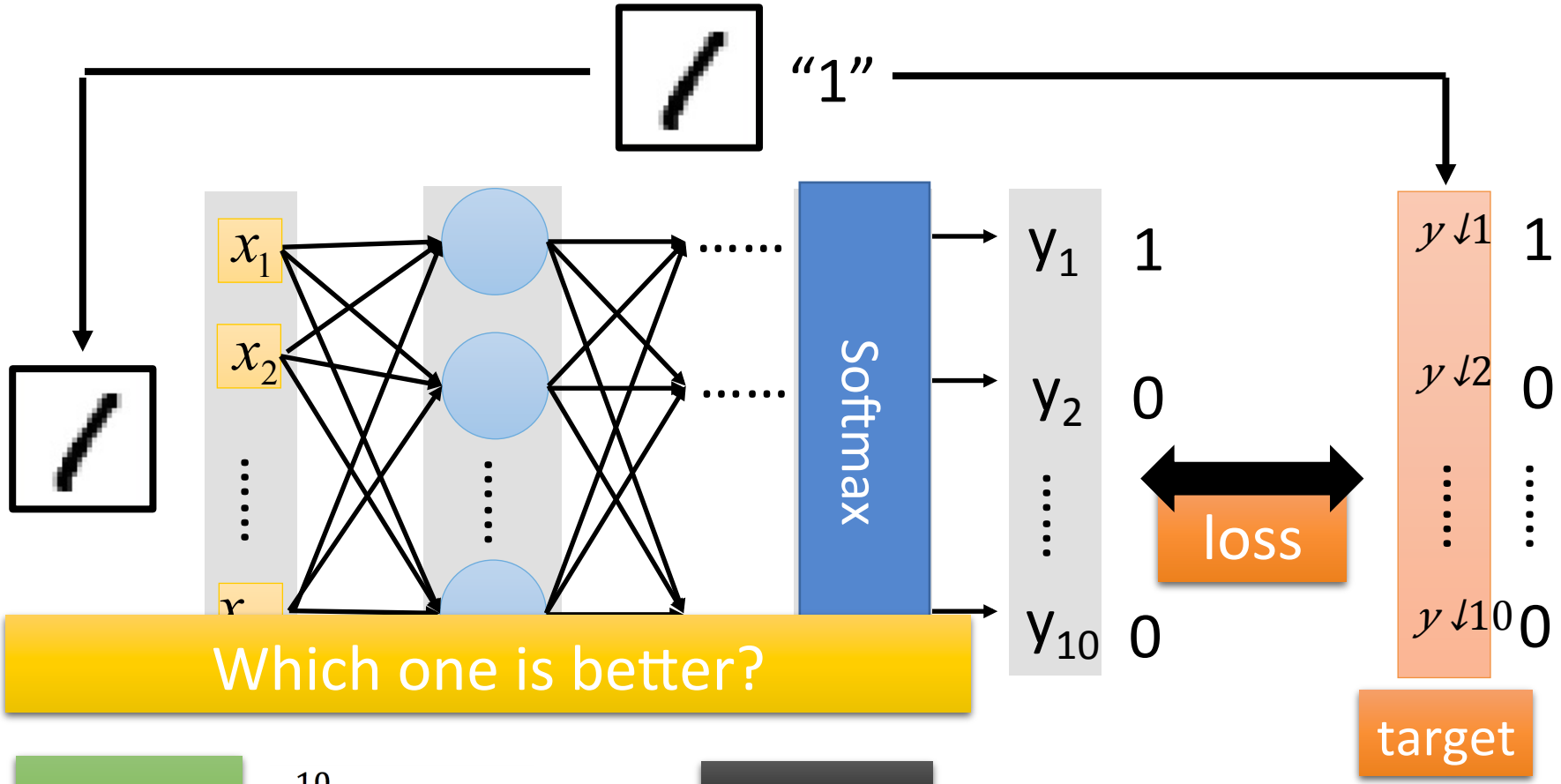
Recipe of Deep Learning



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

Square Error

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

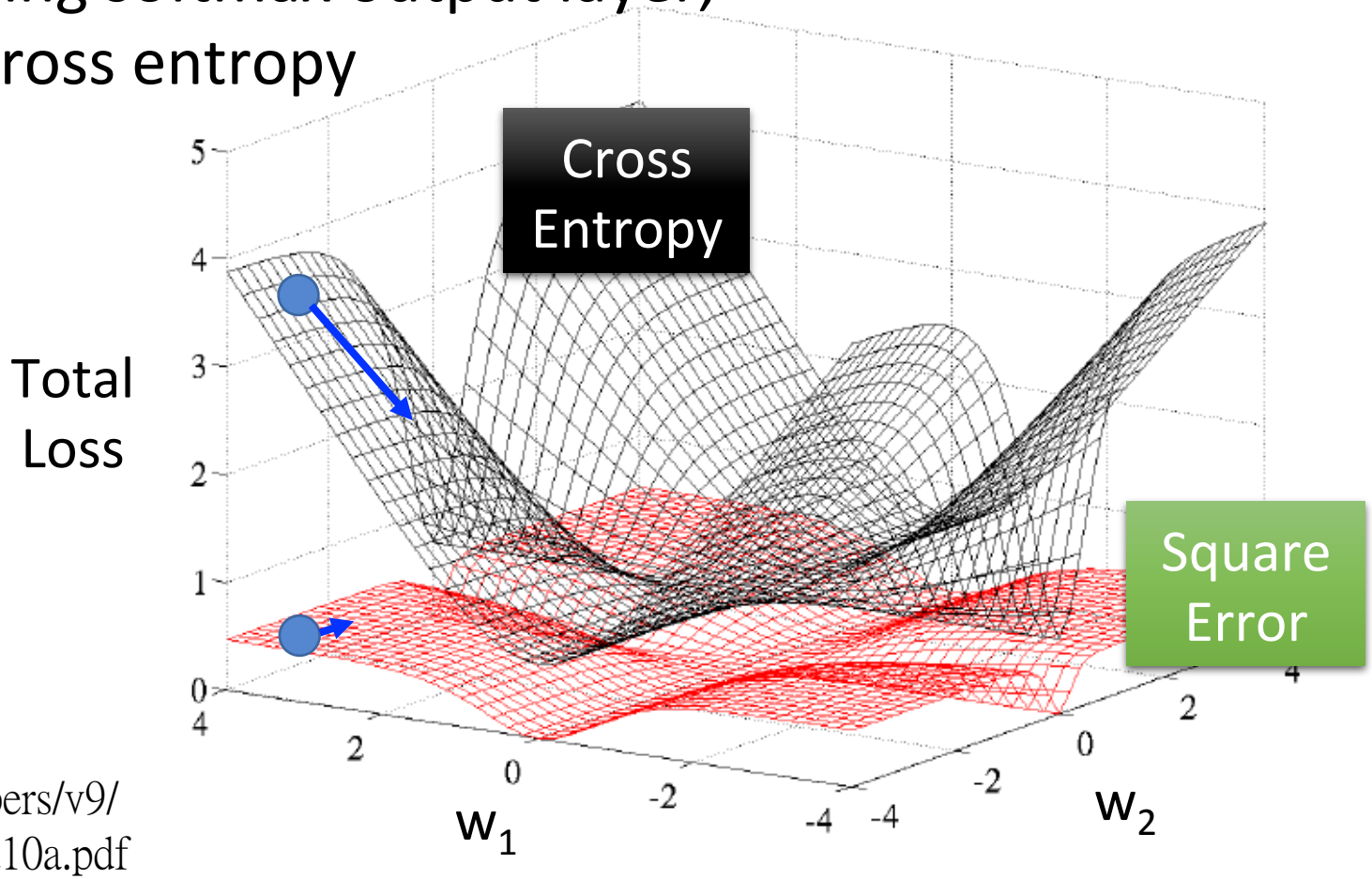
Cross Entropy

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

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

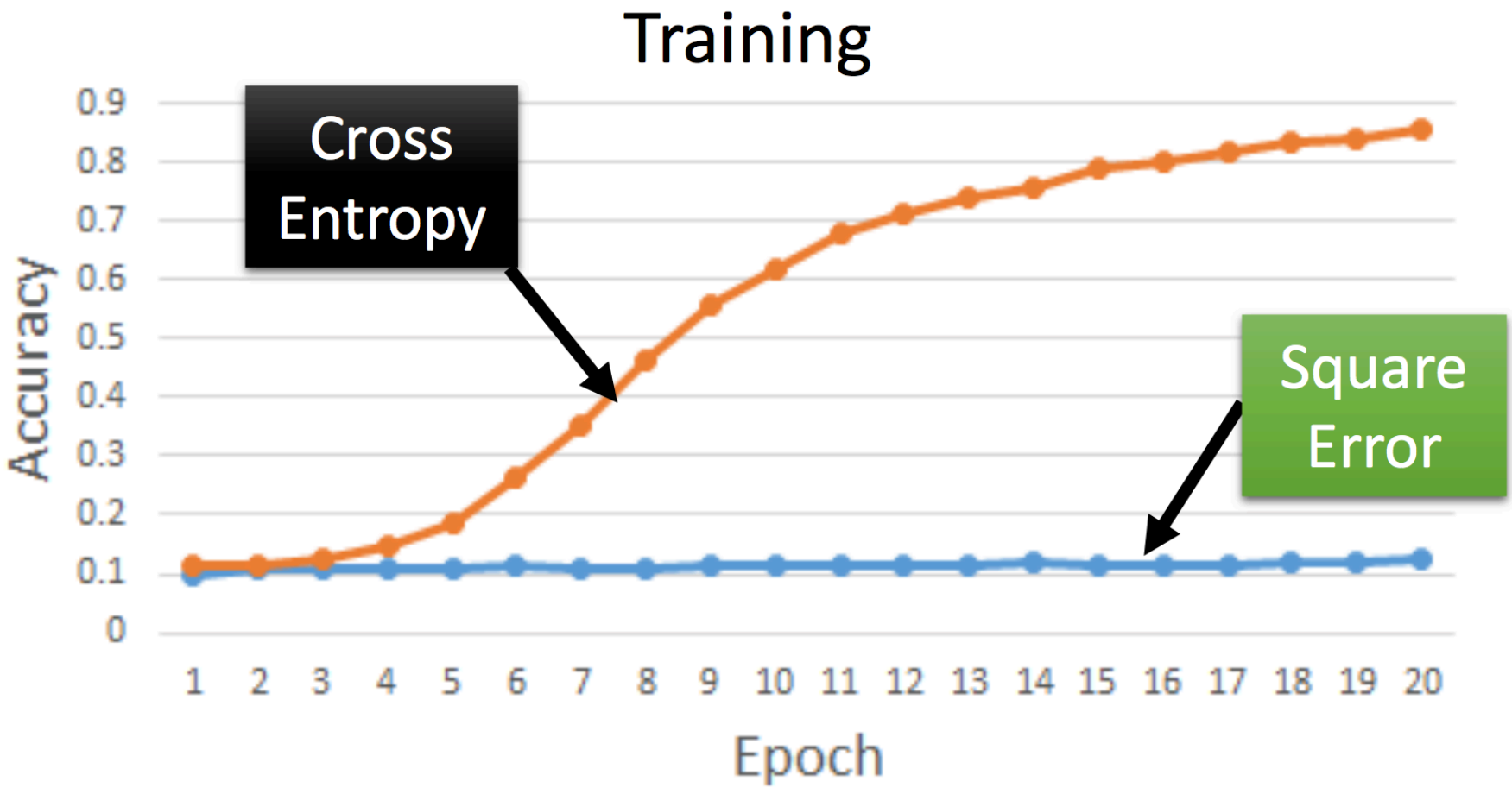
Choosing Proper Loss

When using softmax output layer, choose cross entropy

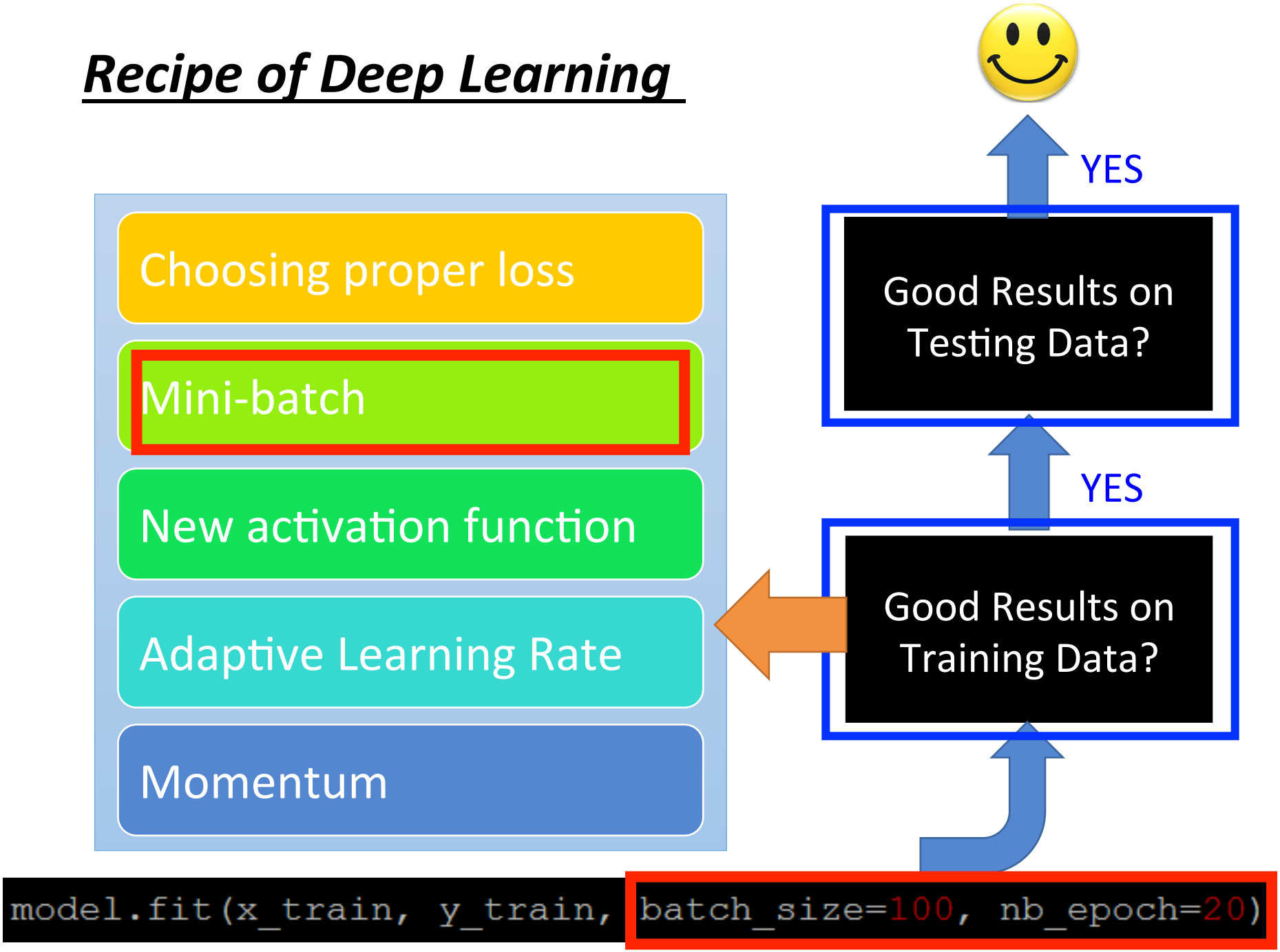


http://jmlr.org/proceedings/papers/v9/glorot10a/glorot10a.pdf

Choosing Proper Loss

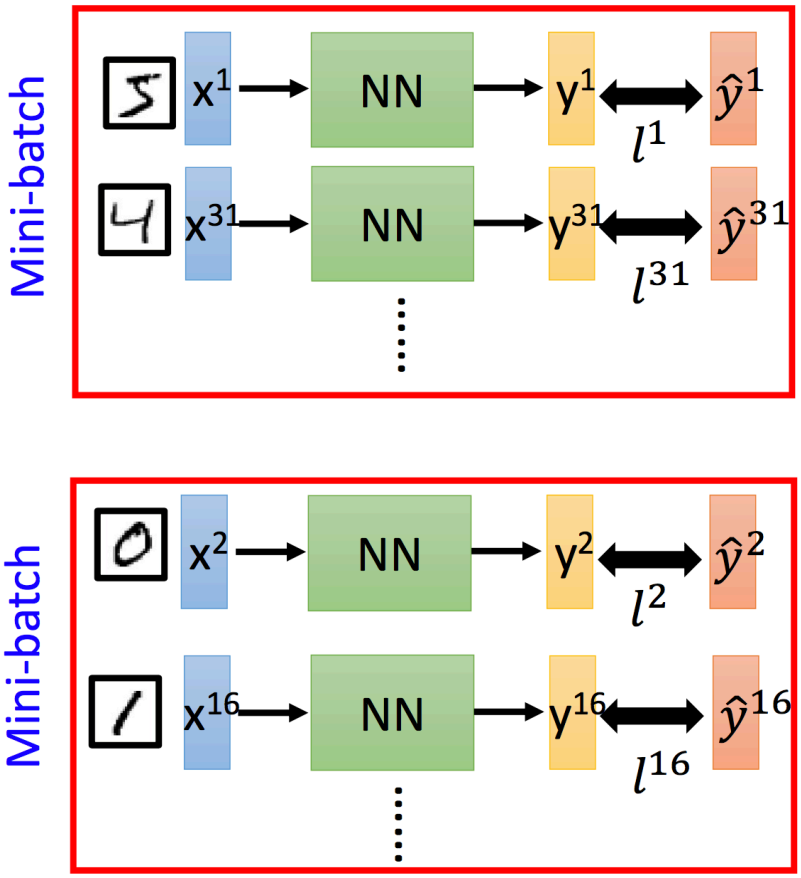


Recipe of Deep Learning



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

  ⋮

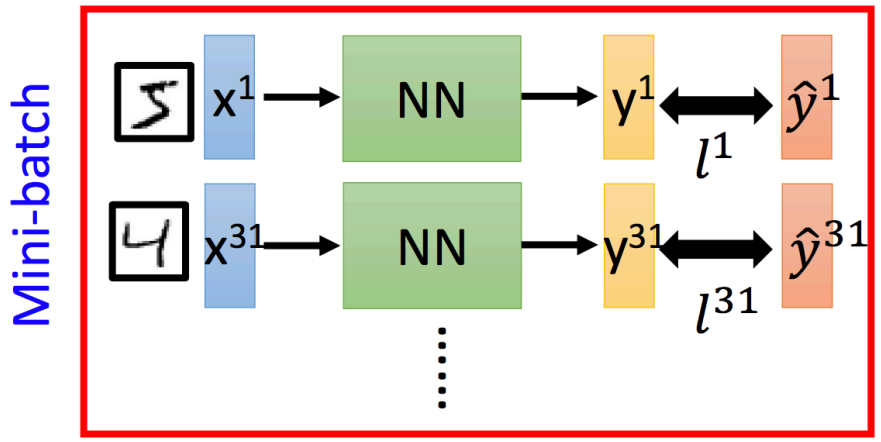
- Until all mini-batches have been picked

one epoch

Repeat the above process

Mini-batch

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



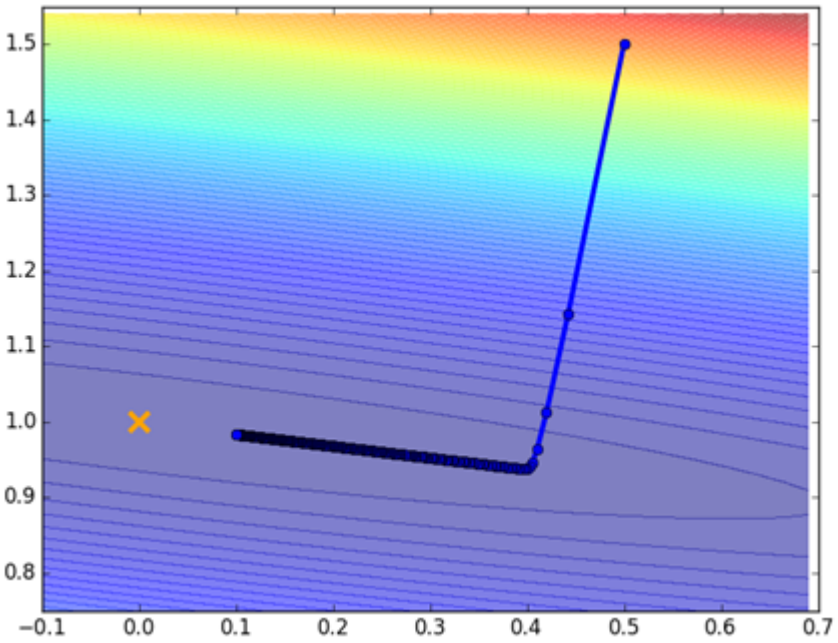
100 examples in a mini-batch

- 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

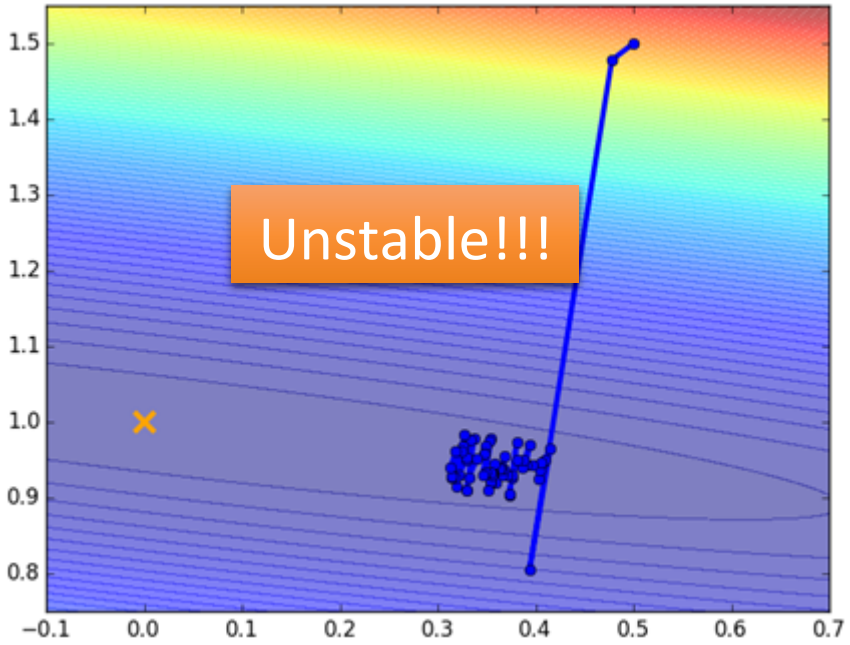
Repeat 20 times

Mini-batch

Original Gradient Descent



With Mini-batch



The colors represent the total loss.

Mini-batch is Faster

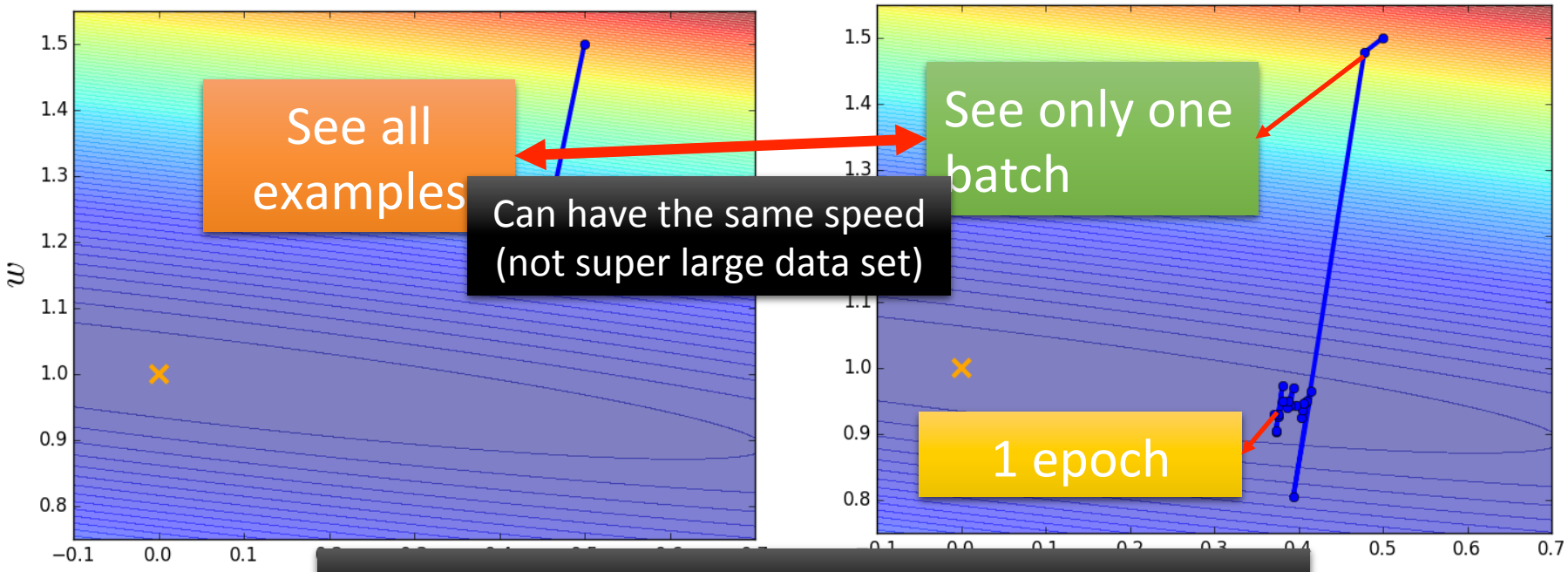
Not always true with parallel computing.

Original Gradient Descent

Update after seeing all examples

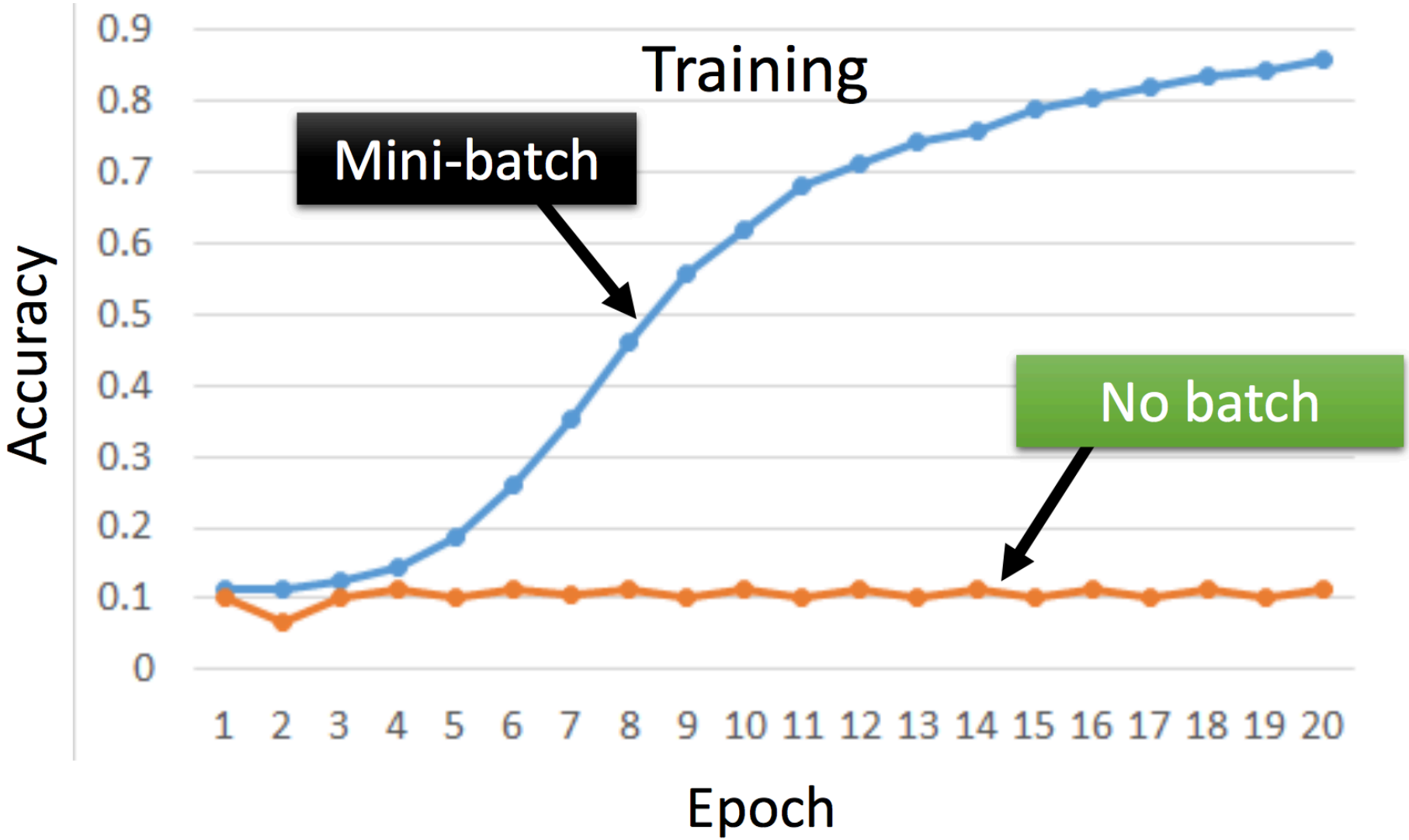
With Mini-batch

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

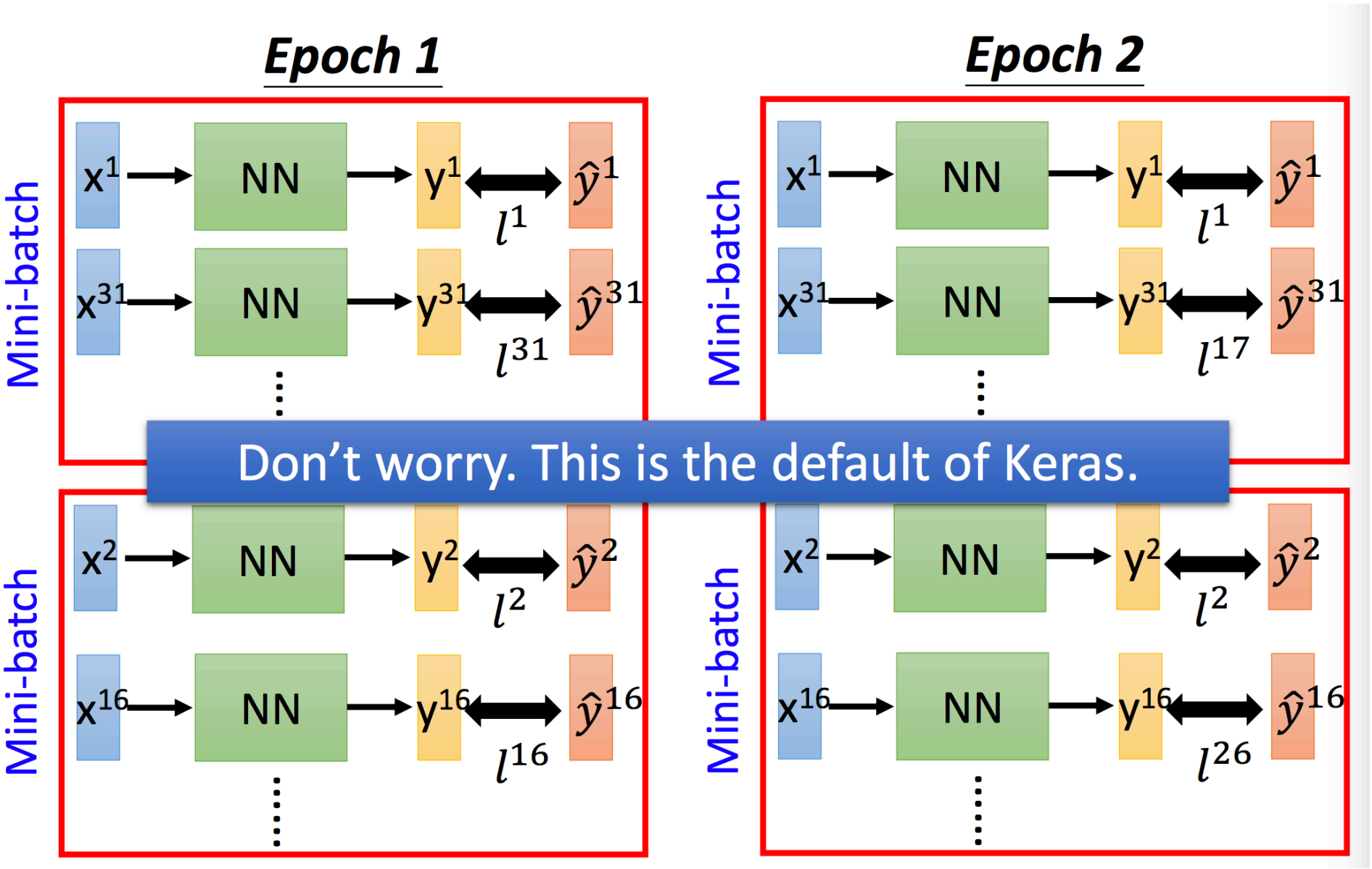


Mini-batch has better performance!

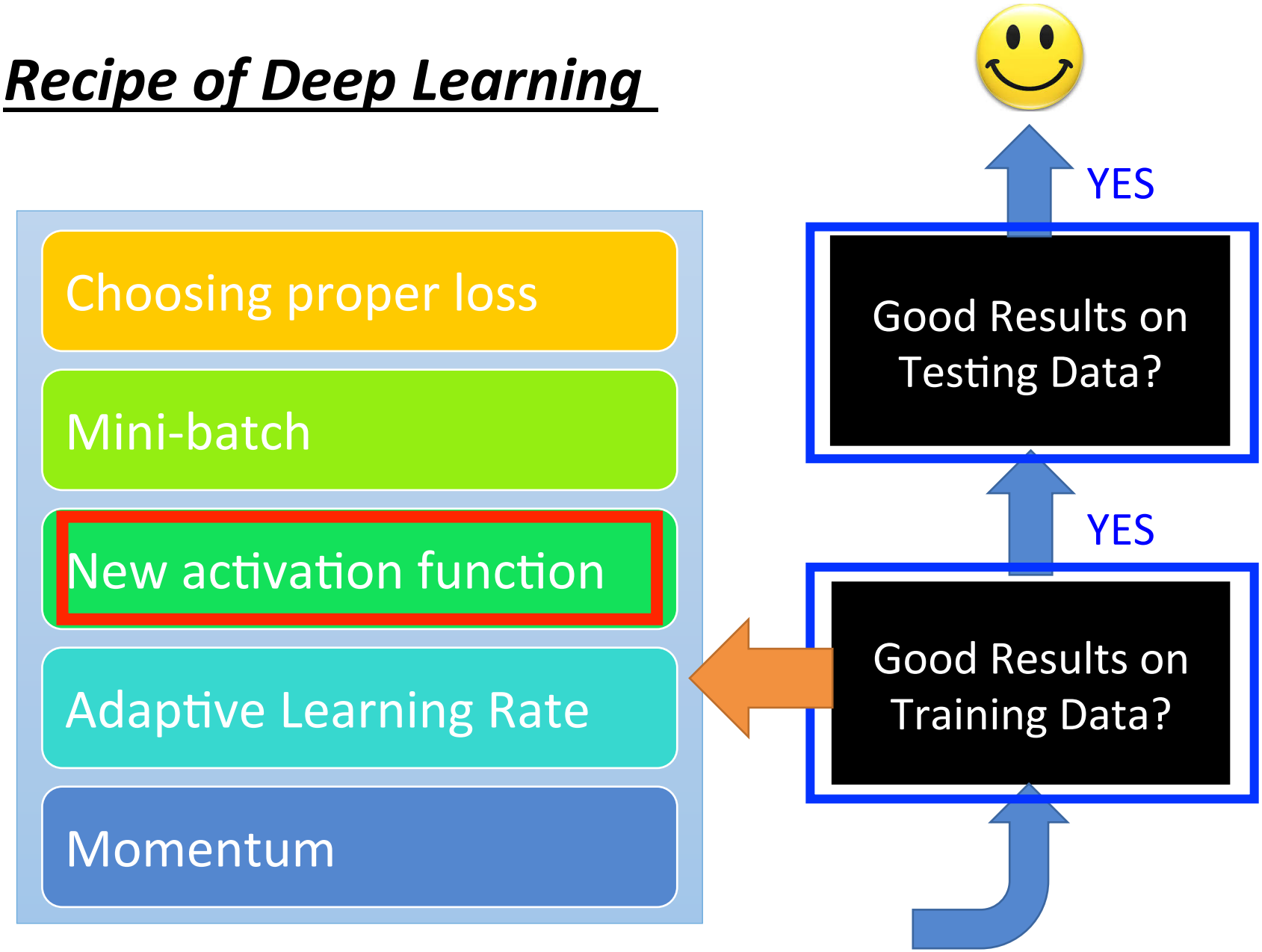
Mini-batch is Faster



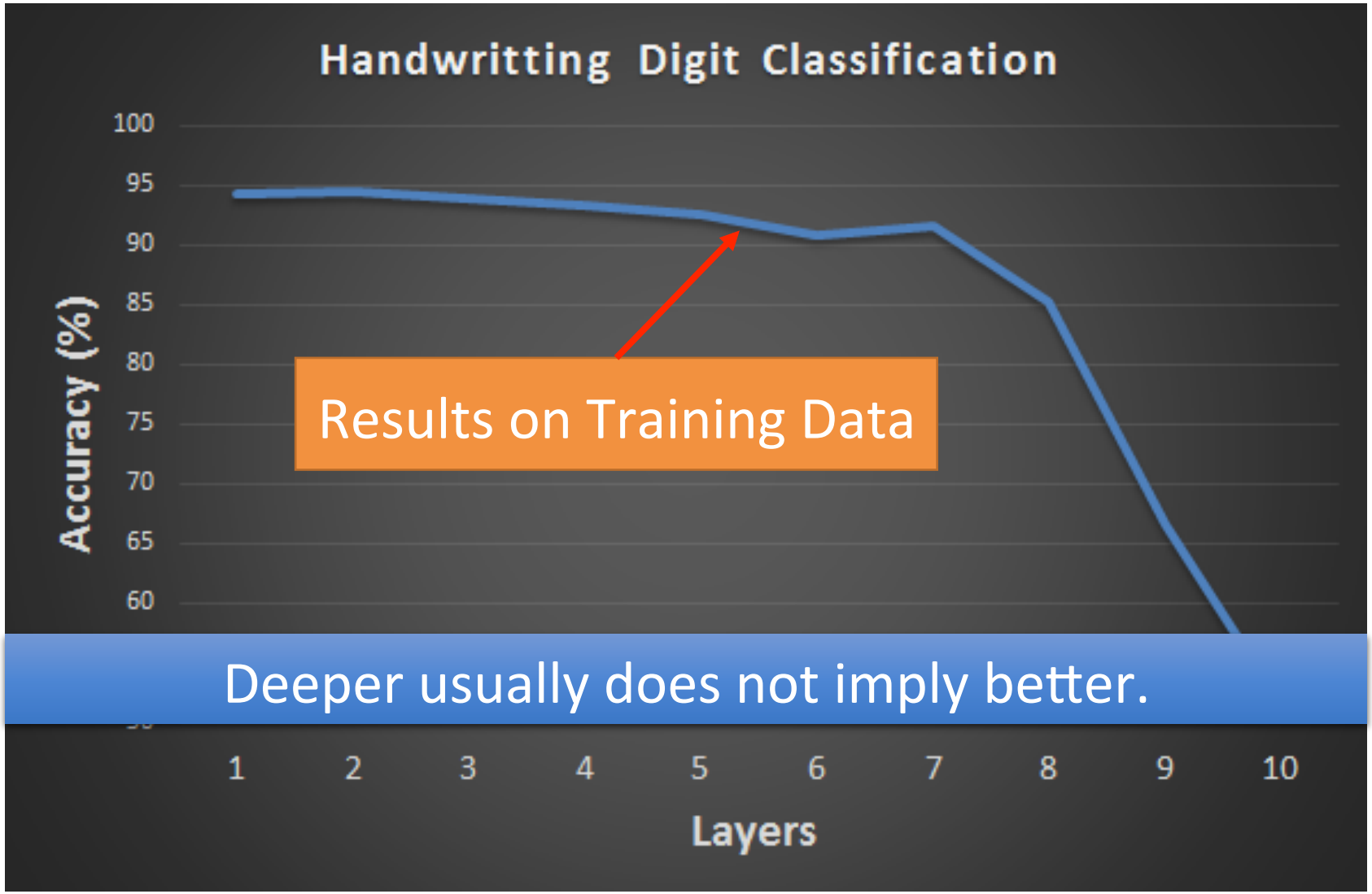
Shuffle the training examples for each epoch



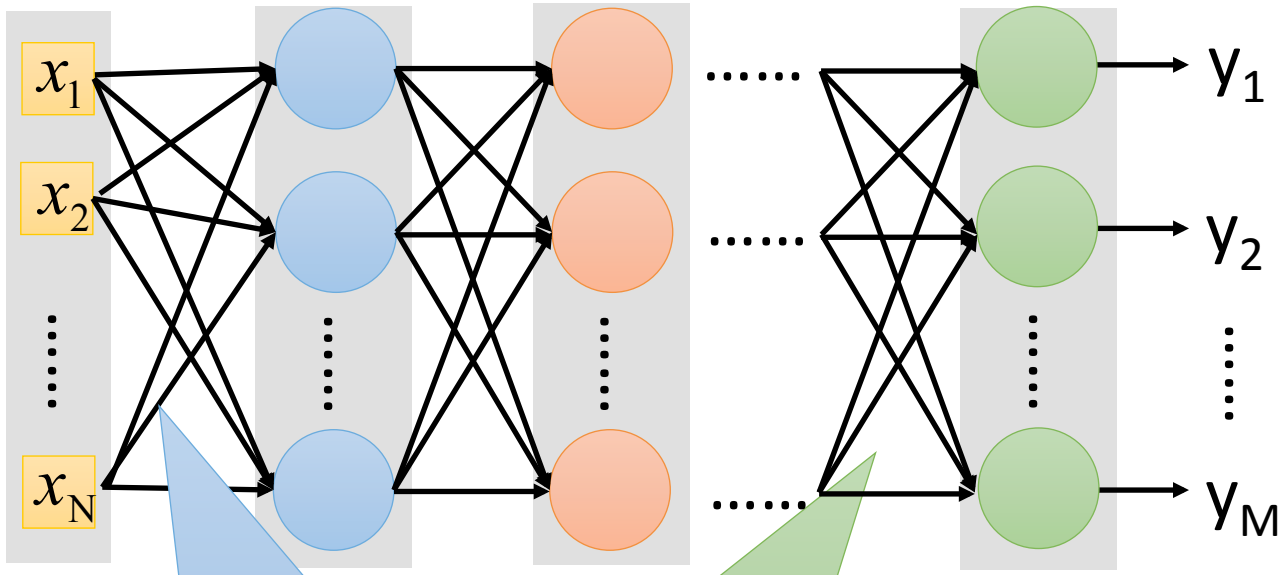
Recipe of Deep Learning



Hard to get the power of Deep …



Vanishing Gradient Problem



Smaller gradients

Learn very slow

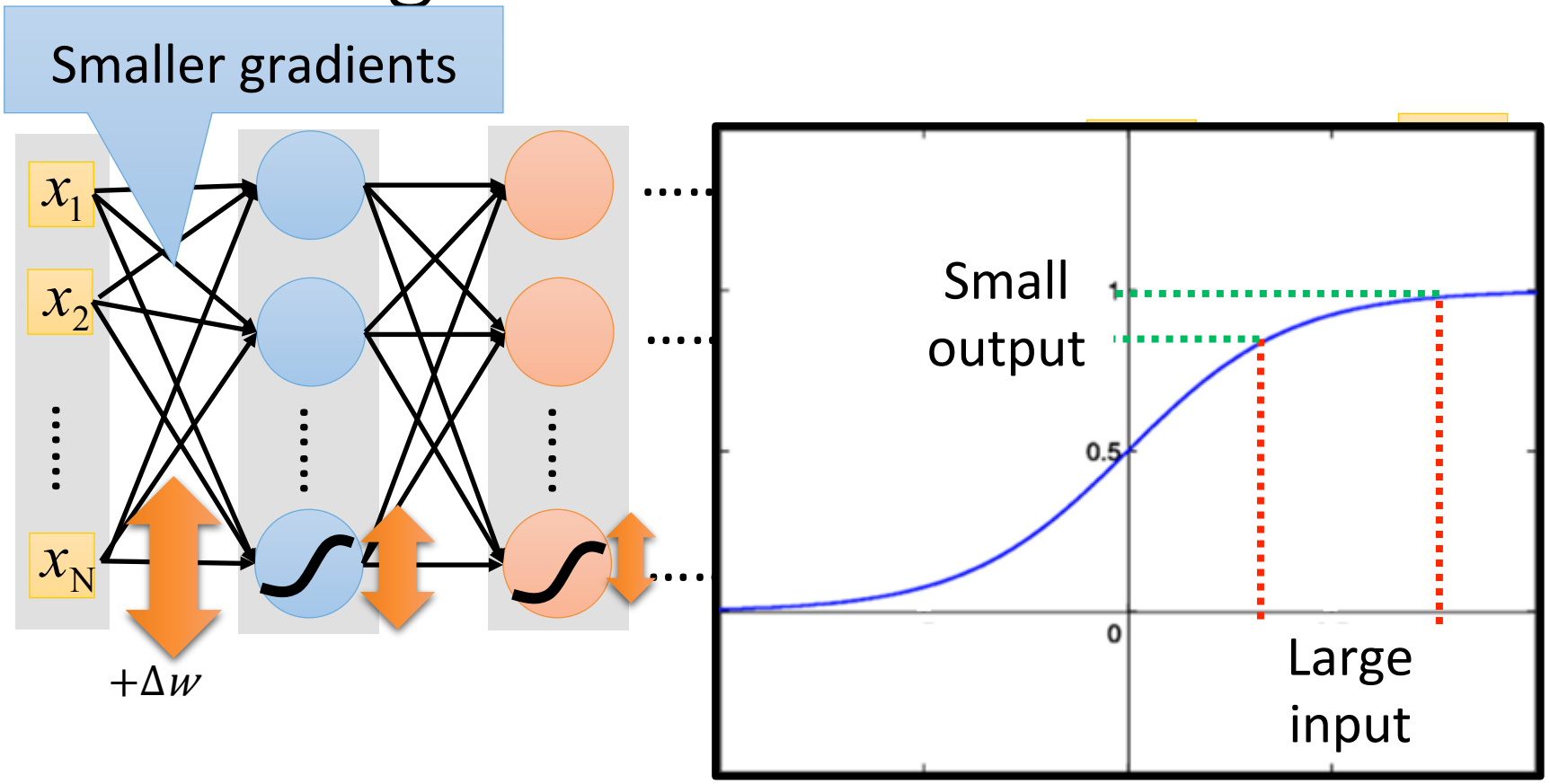
Almost random

Larger gradients

Learn very fast

Already converge

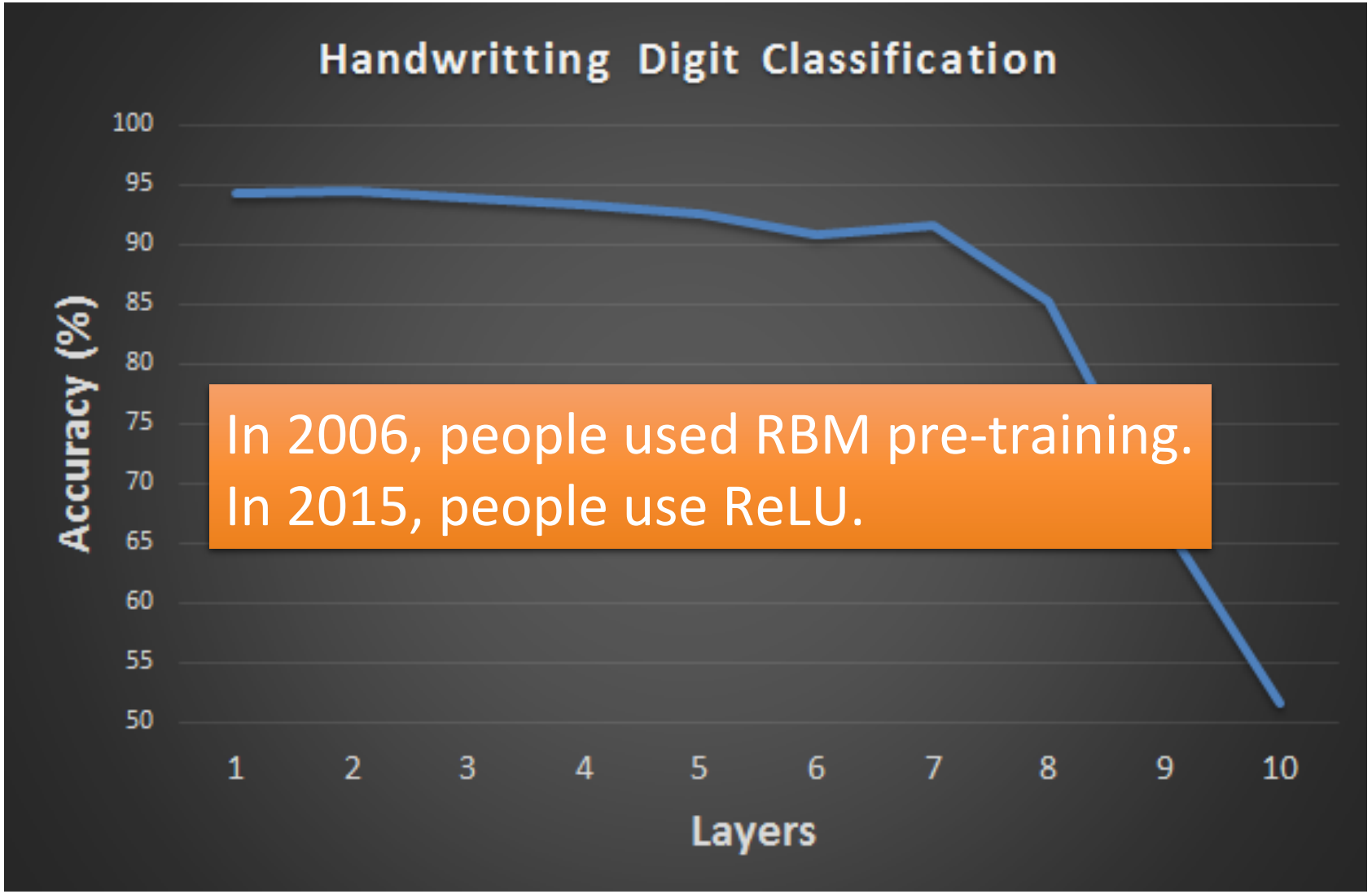
Vanishing Gradient Problem



Intuitive way to compute the derivatives …

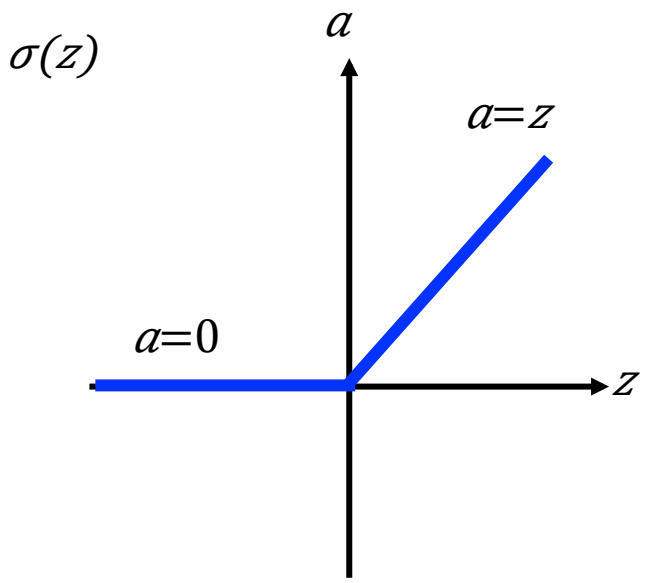
$\partial l/\partial w = ?\ \Delta l/\Delta w$

Hard to get the power of Deep …



ReLU

- Rectified Linear Unit (ReLU)

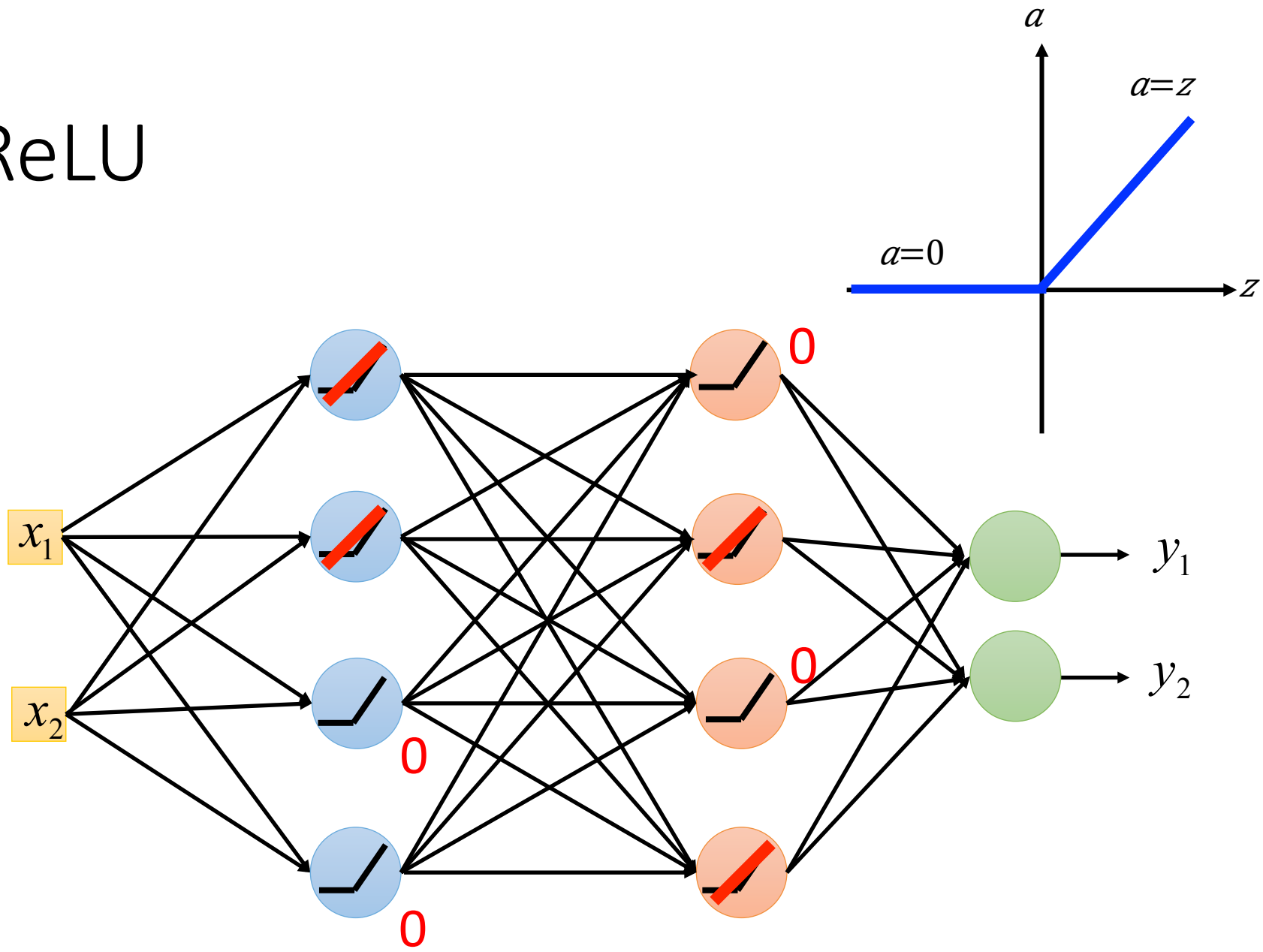


[Xavier Glorot, AISTATS'11]
[Andrew L. Maas, ICML'13]
[Kaiming He, arXiv'15]

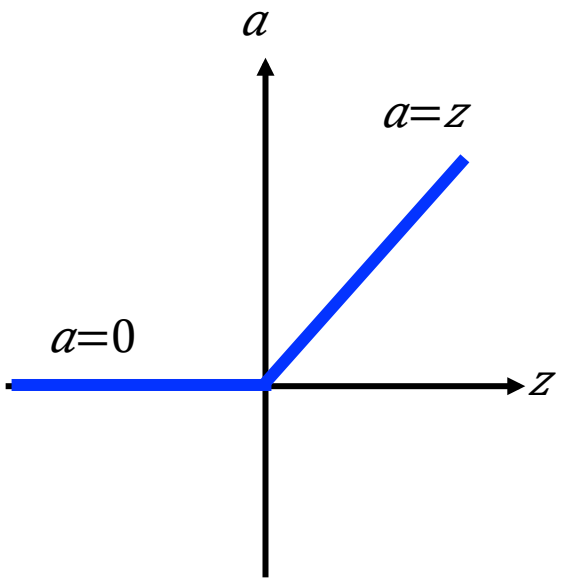
Reason:

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

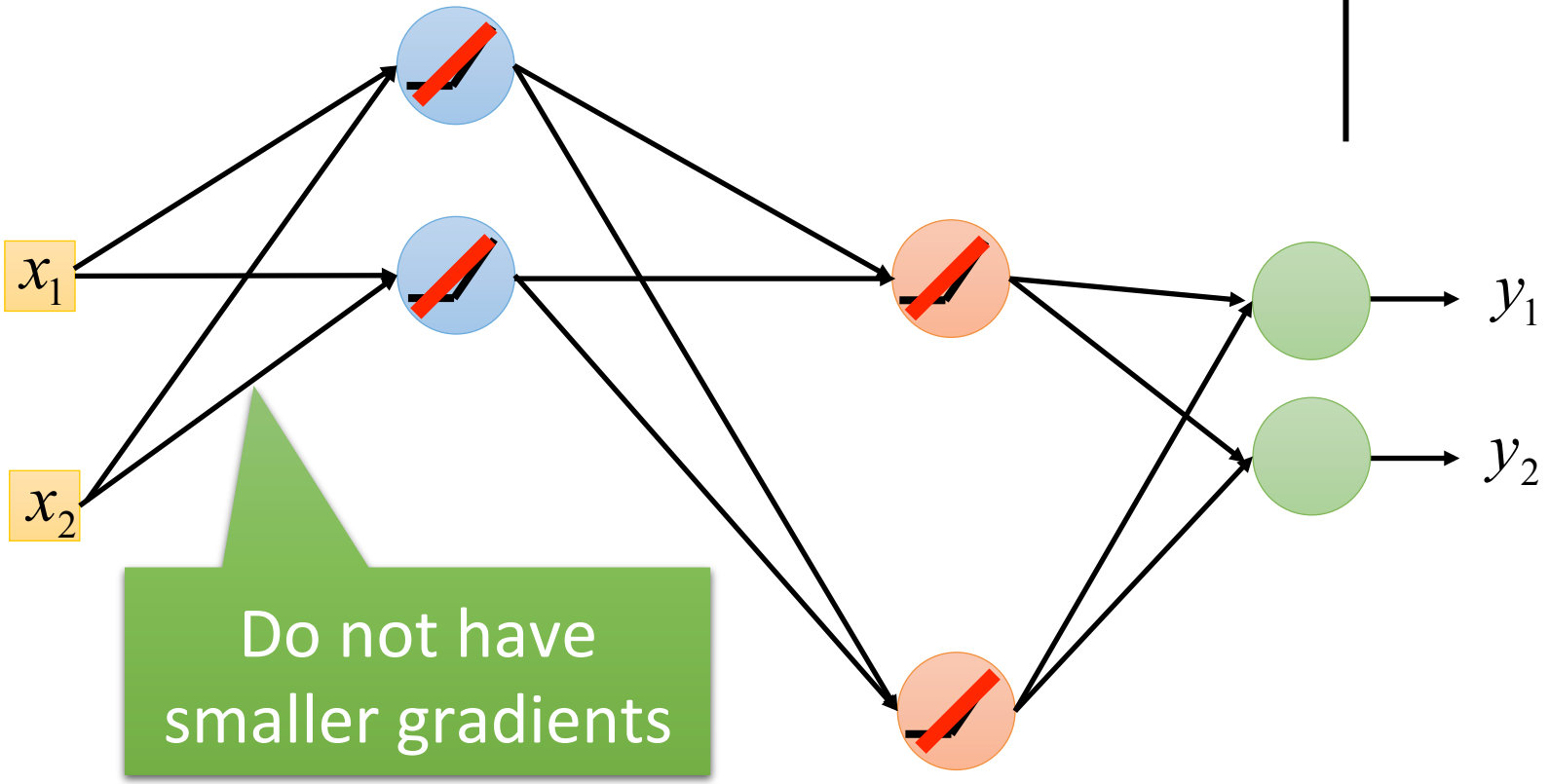
ReLU



ReLU

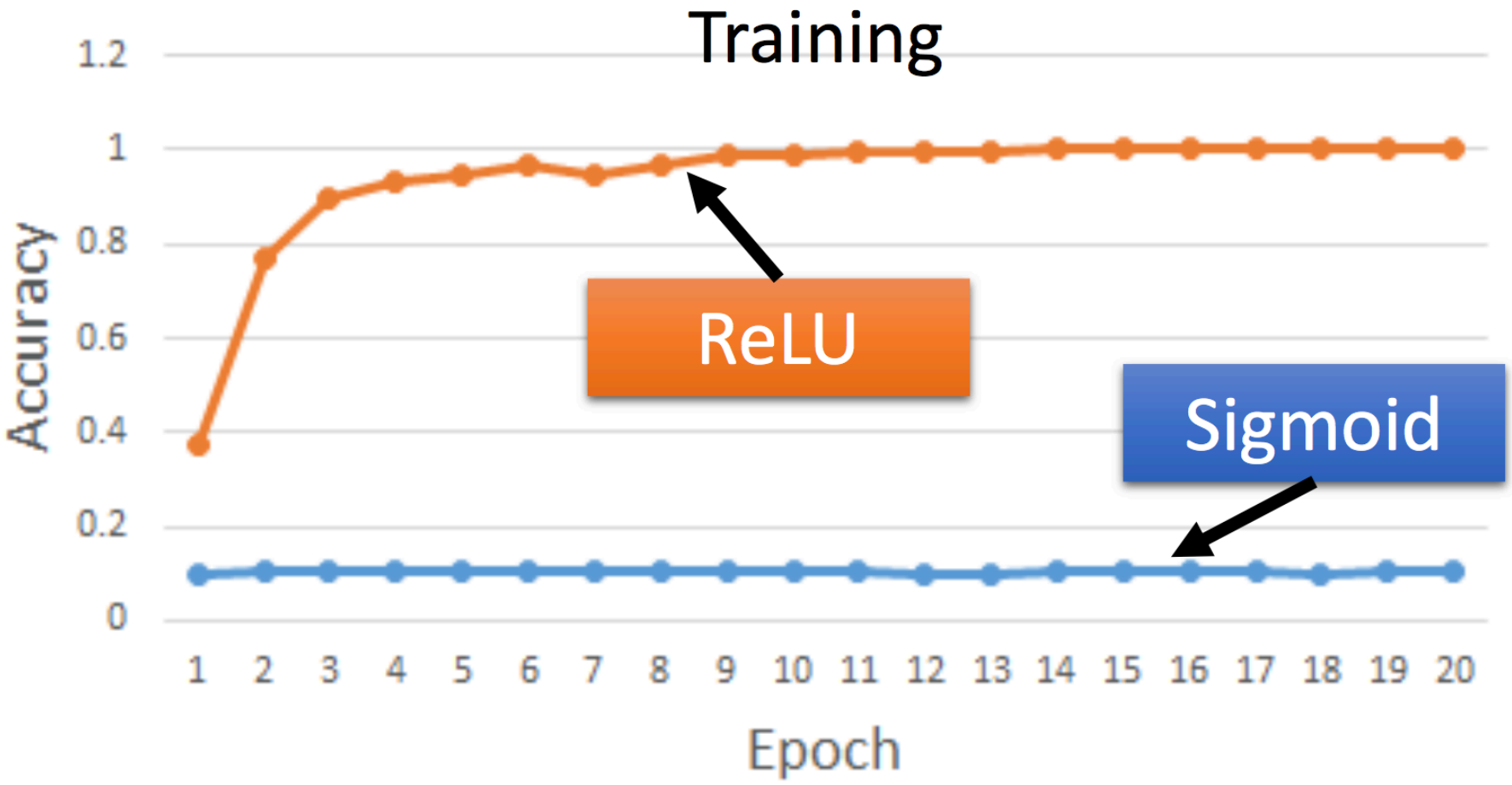


A Thinner linear network



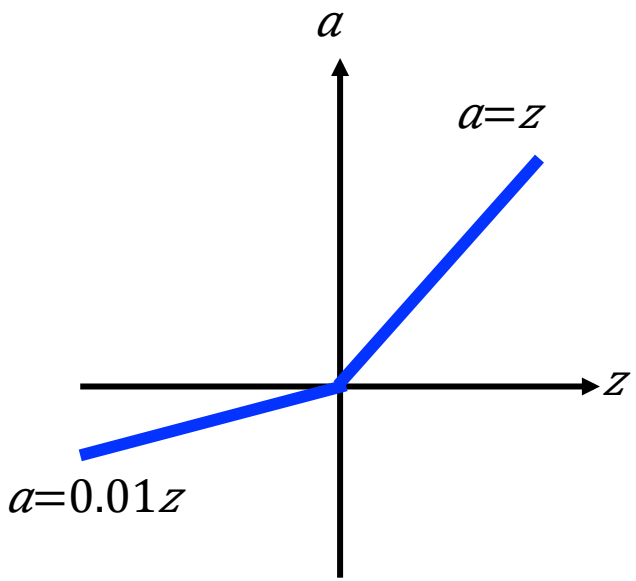
Do not have smaller gradients

ReLU

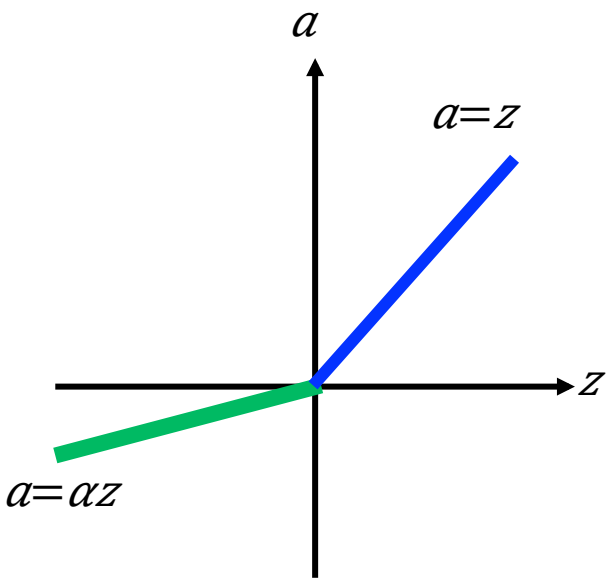

Training
1.2
1
0.8
0.6
0.4
0.2
0
Accuracy
ReLU
Sigmoid
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Epoch

ReLU - variant

Leaky ReLU



Parametric ReLU

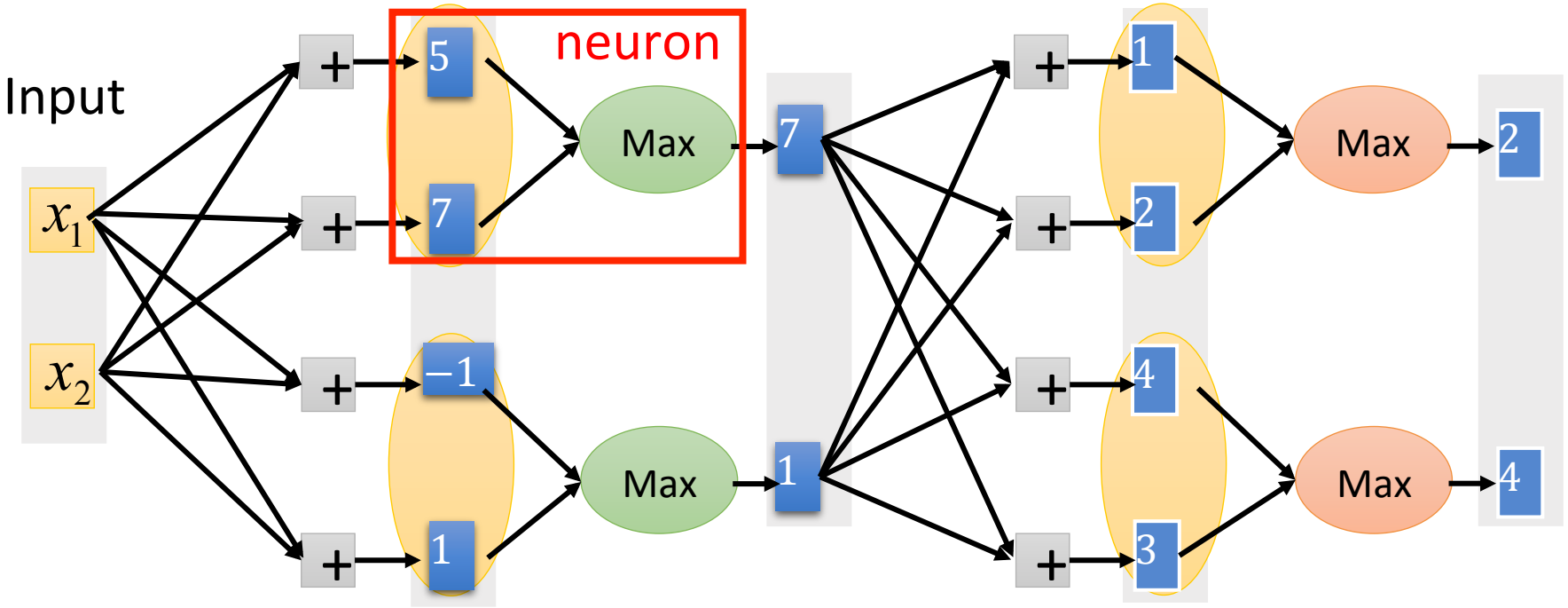


α also learned by gradient descent

Maxout

ReLU is a special cases of Maxout

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



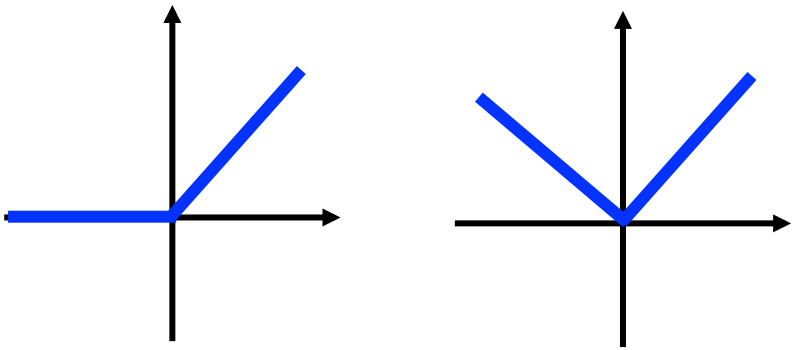
You can have more than 2 elements in a group.

Maxout

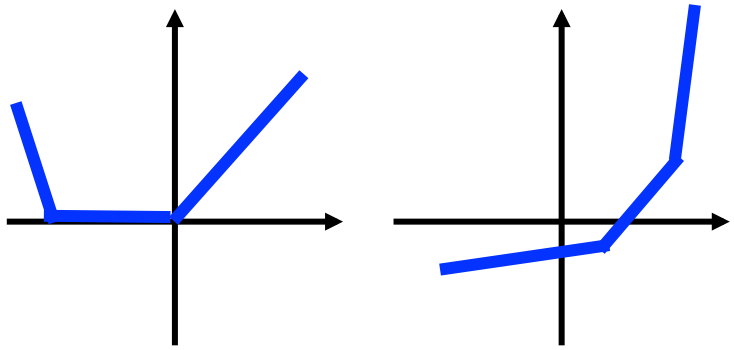
ReLU is a special cases of Maxout

- 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

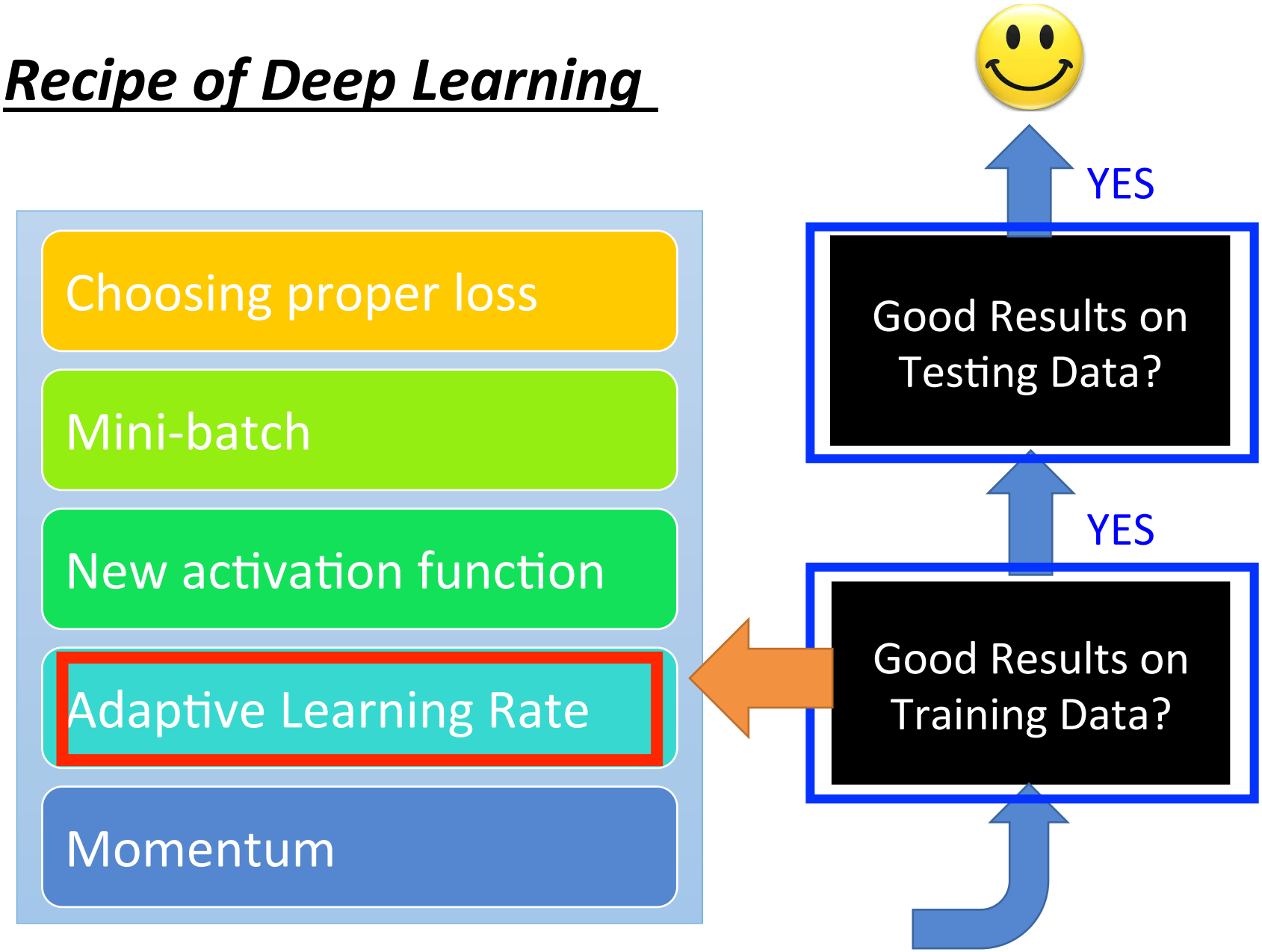
2 elements in a group



3 elements in a group

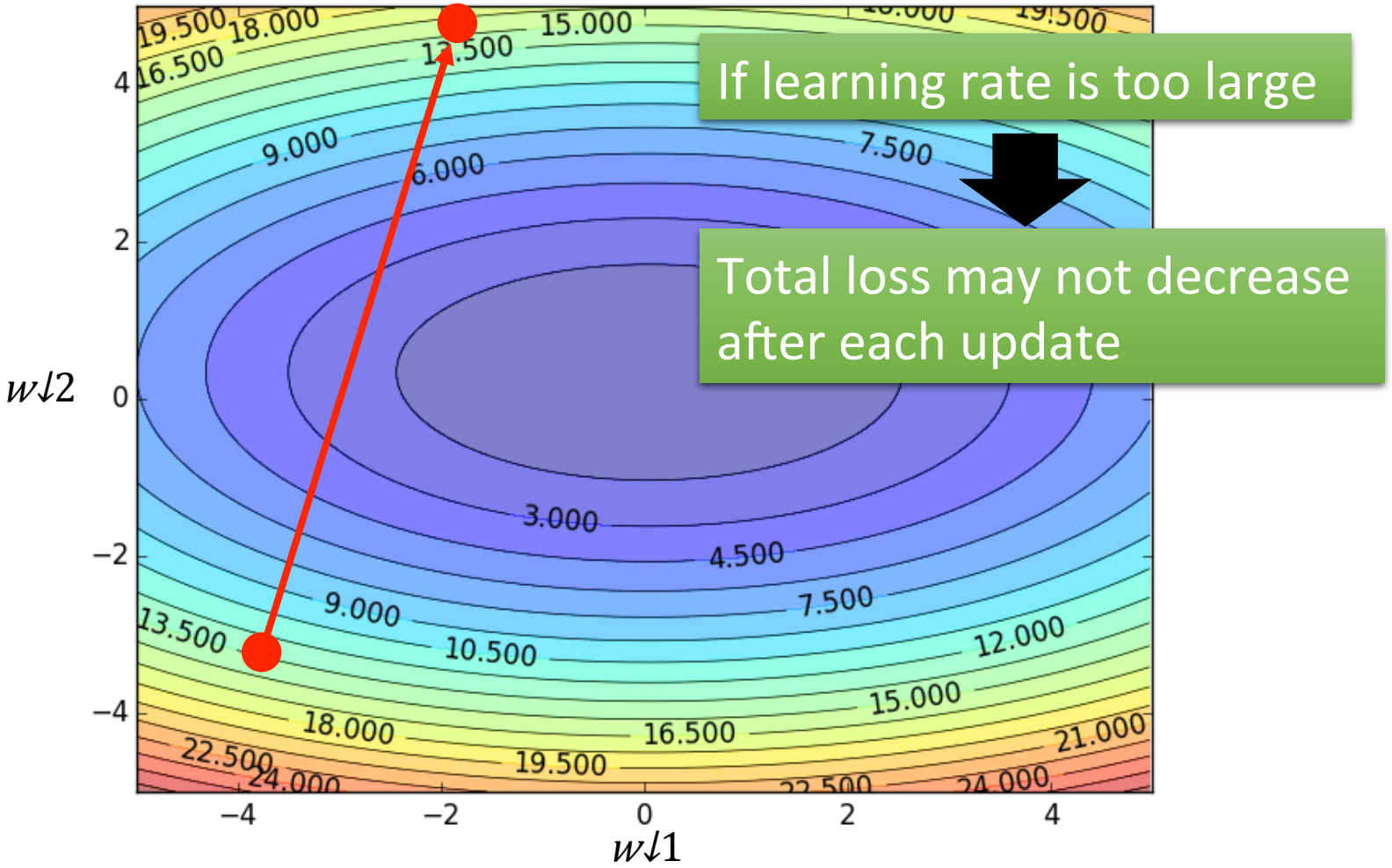


Recipe of Deep Learning



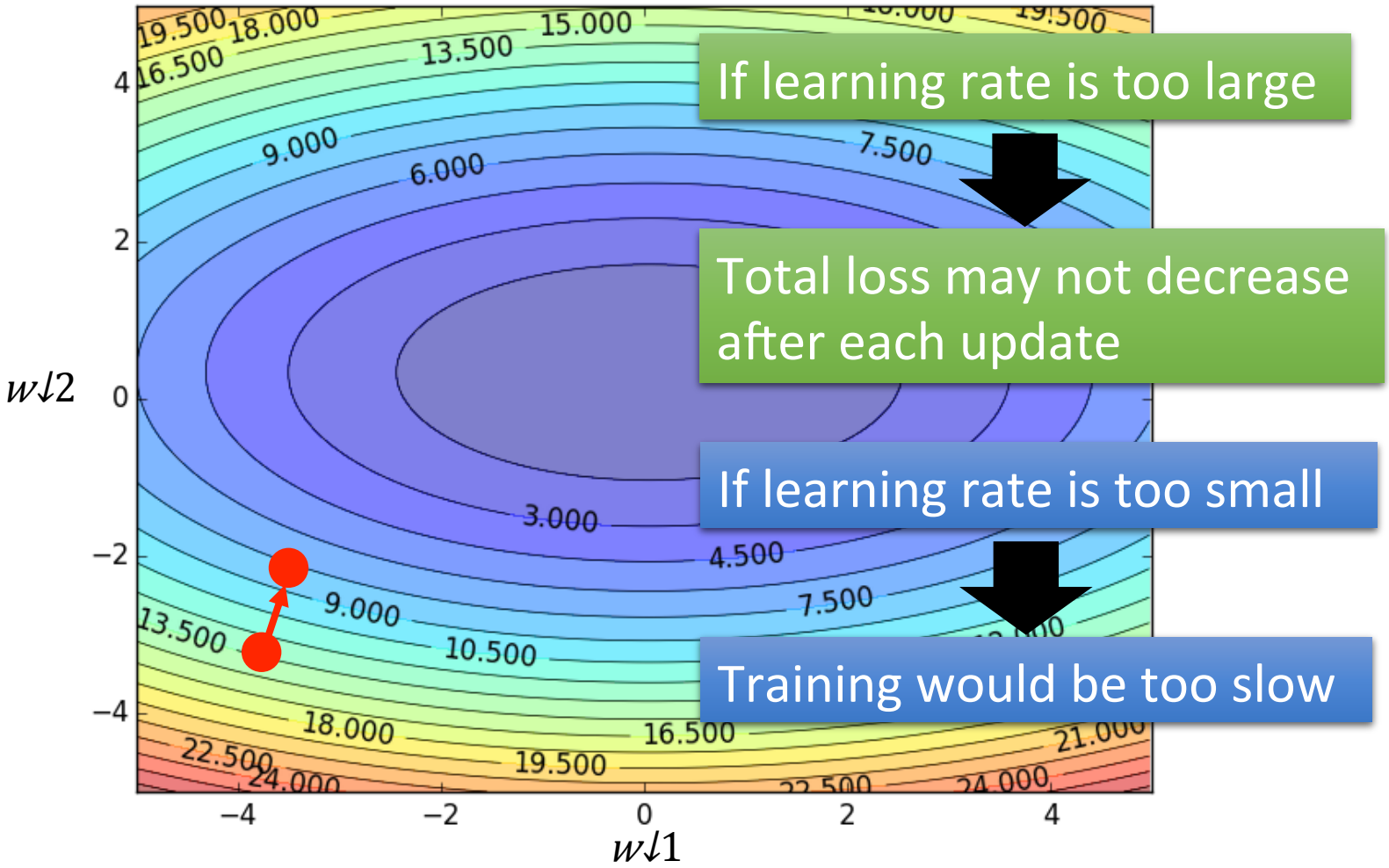
Learning Rates

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

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

w_1	g^0
	0.1

w_2	g^0
	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:

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

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

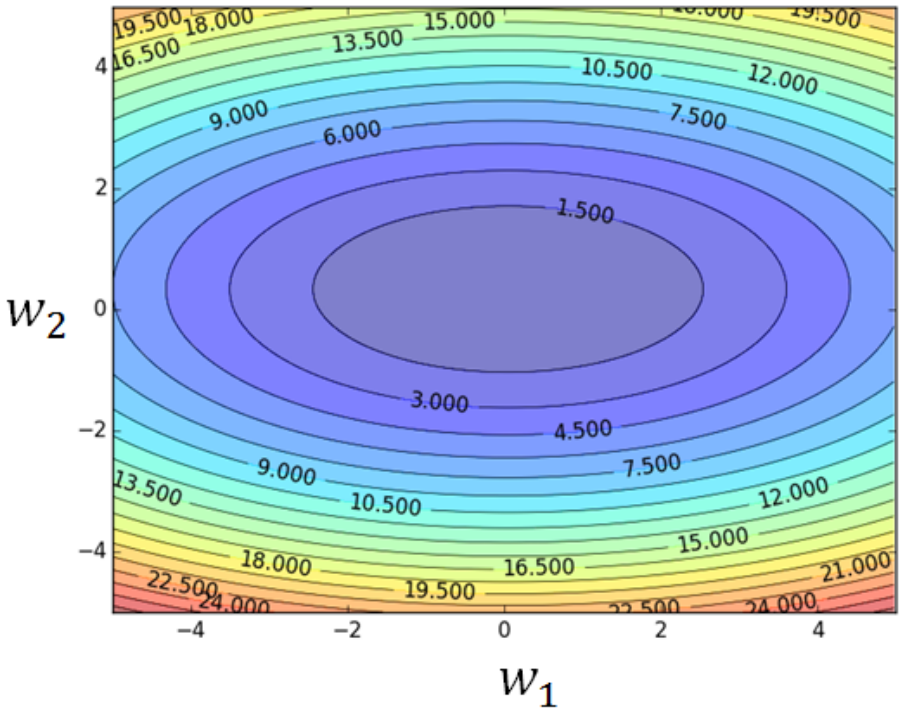
Observation: 1. Learning rate is smaller and smaller for all parameters

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

Why?

Larger derivatives

Smaller Learning Rate



Smaller Derivatives

Larger Learning Rate

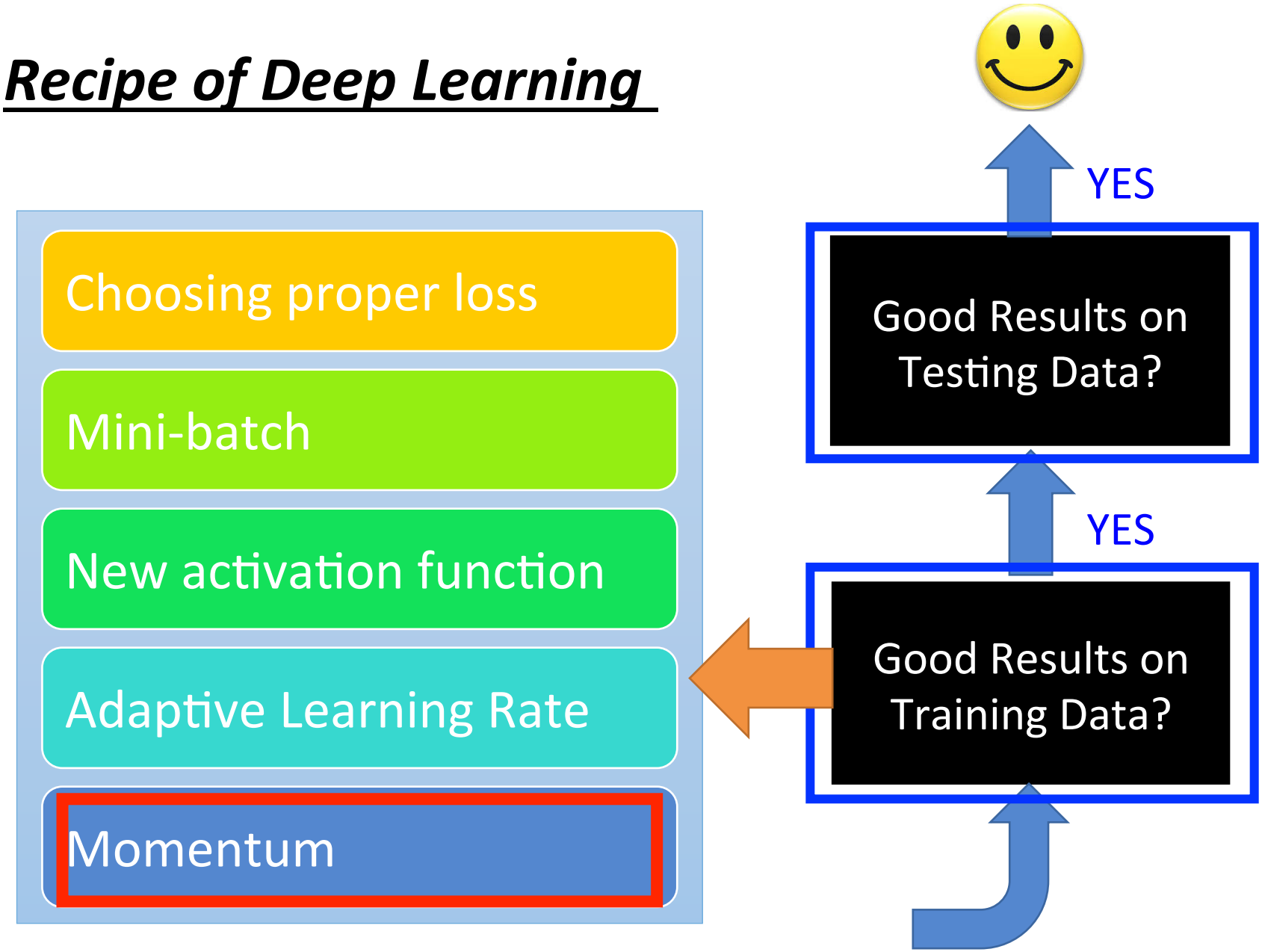
2. Smaller derivatives, larger learning rate, and vice versa

Why?

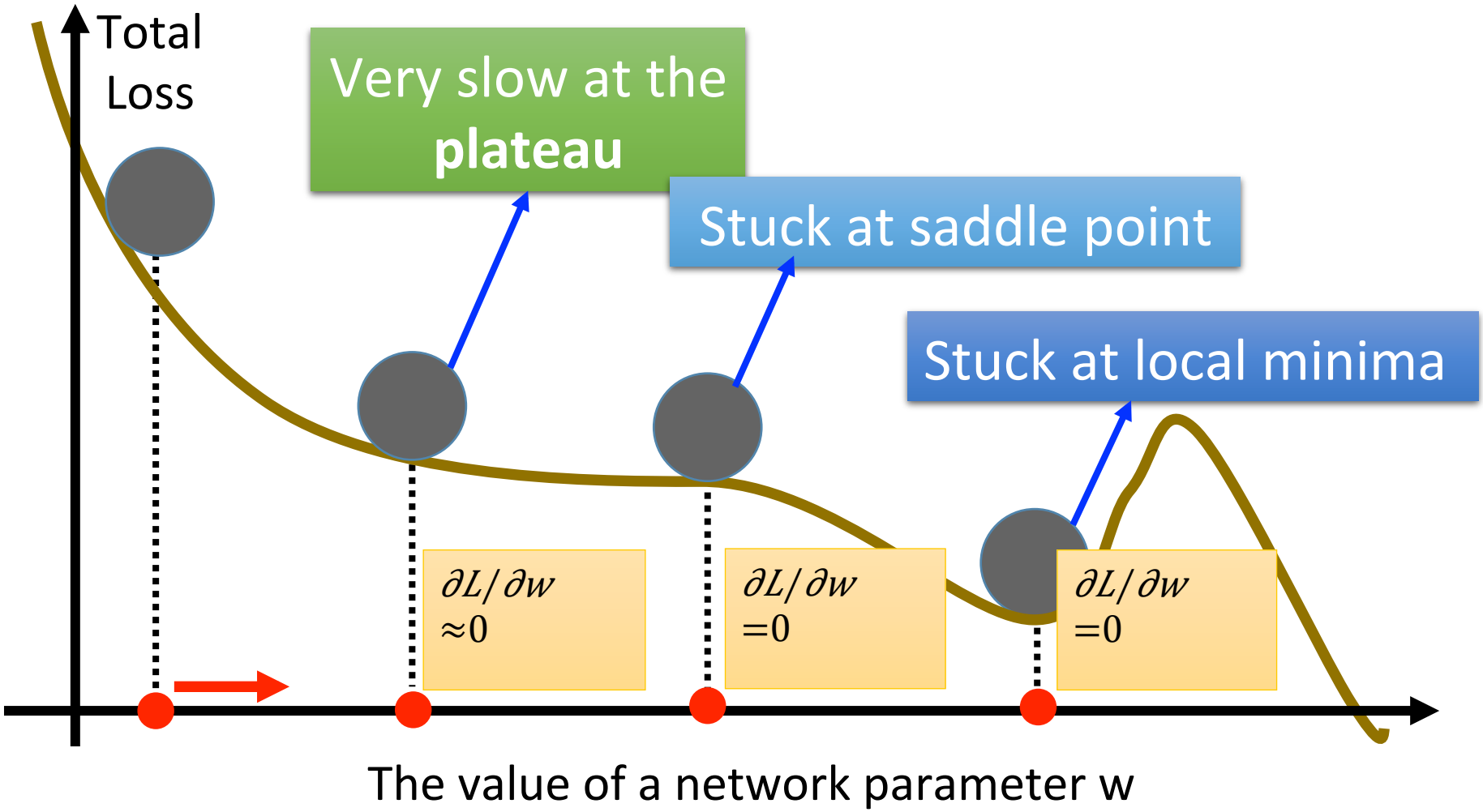
Not the whole story ……

- 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

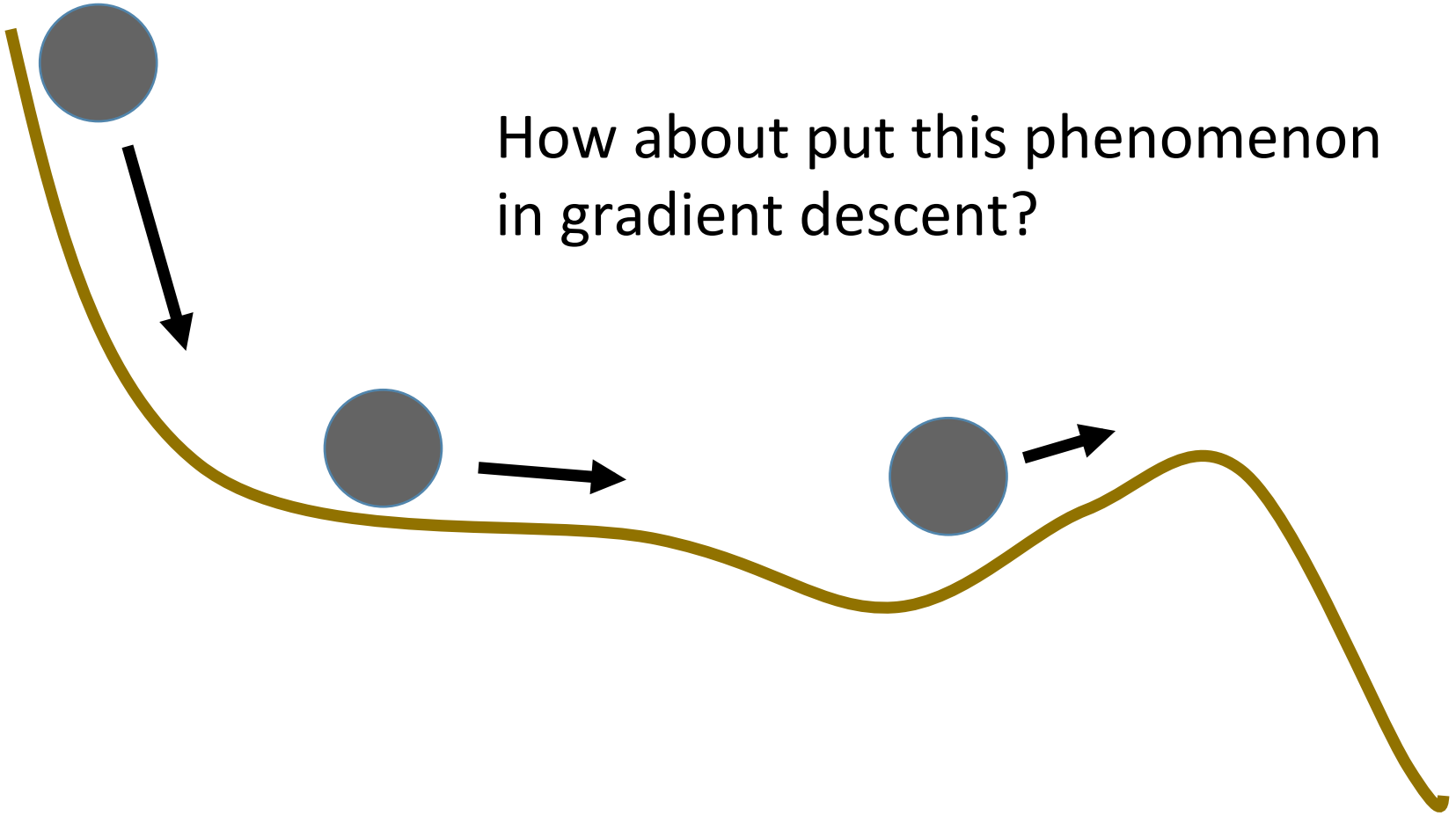


Hard to find optimal network parameters



In physical world ……

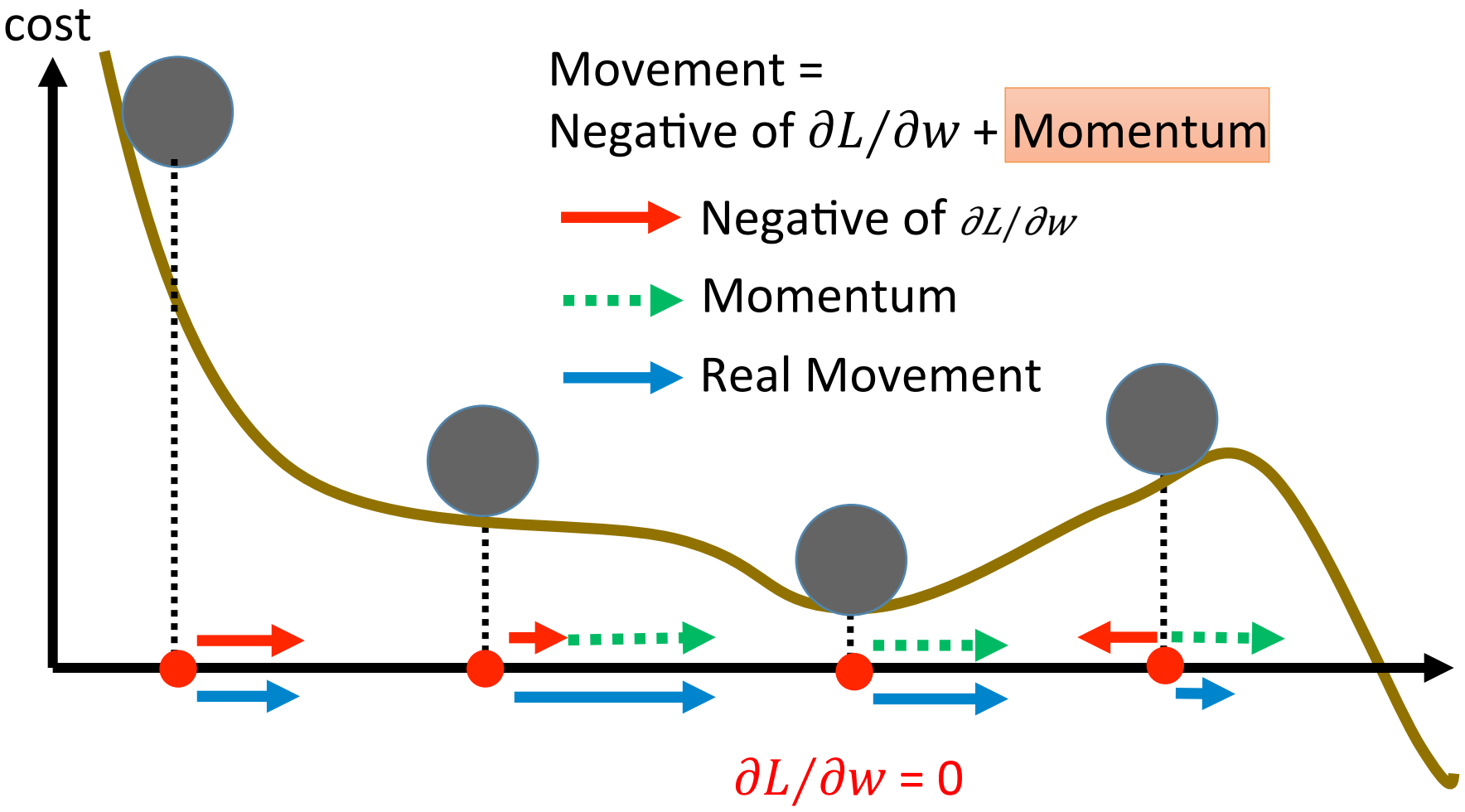
- Momentum



How about put this phenomenon in gradient descent?

Momentum

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



Adam

RMSProp (Advanced Adagrad) + Momentum

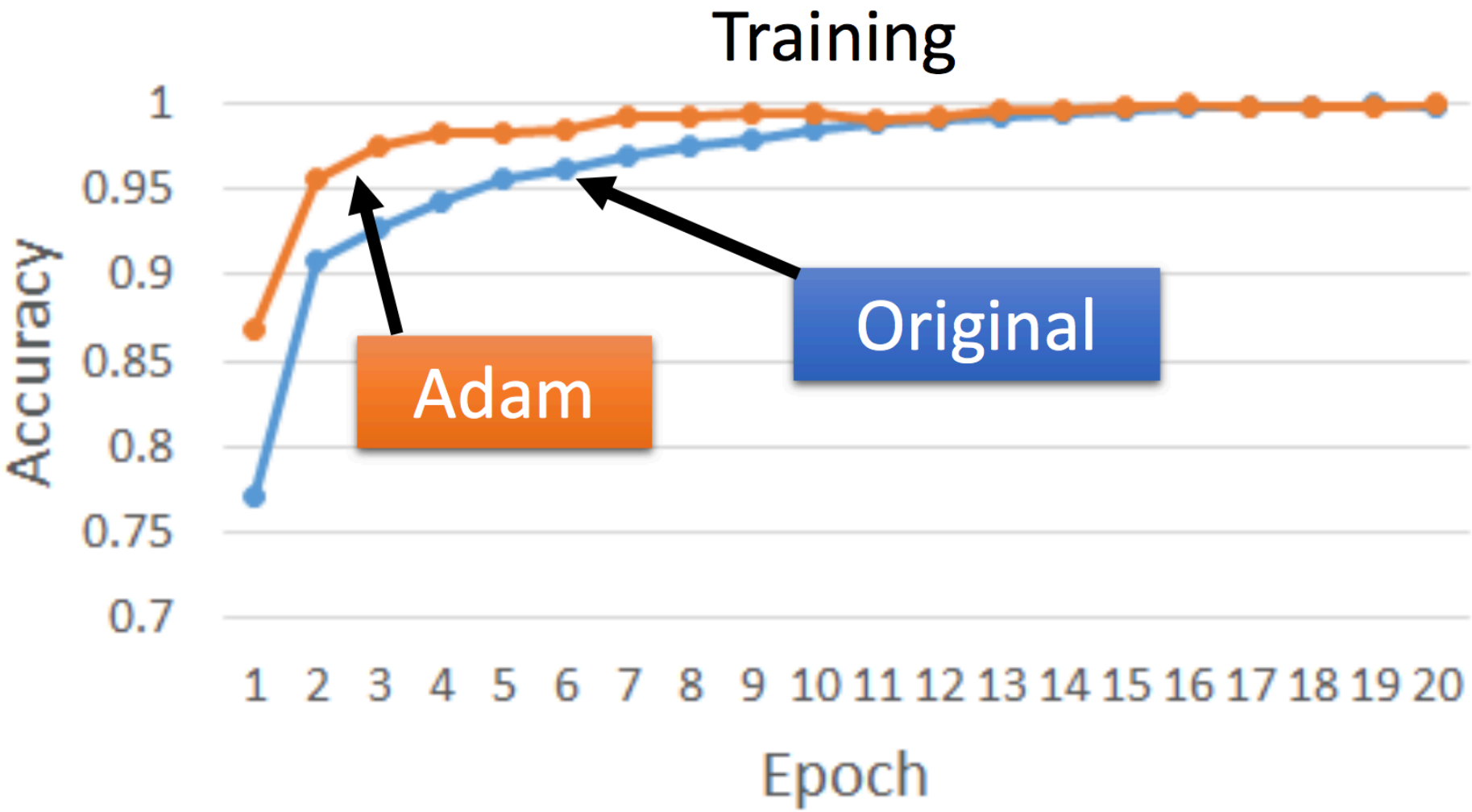
```
model.compile(loss='categorical_crossentropy',
              optimizer=SGD(lr=0.1),
              metrics=['accuracy'])
```

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

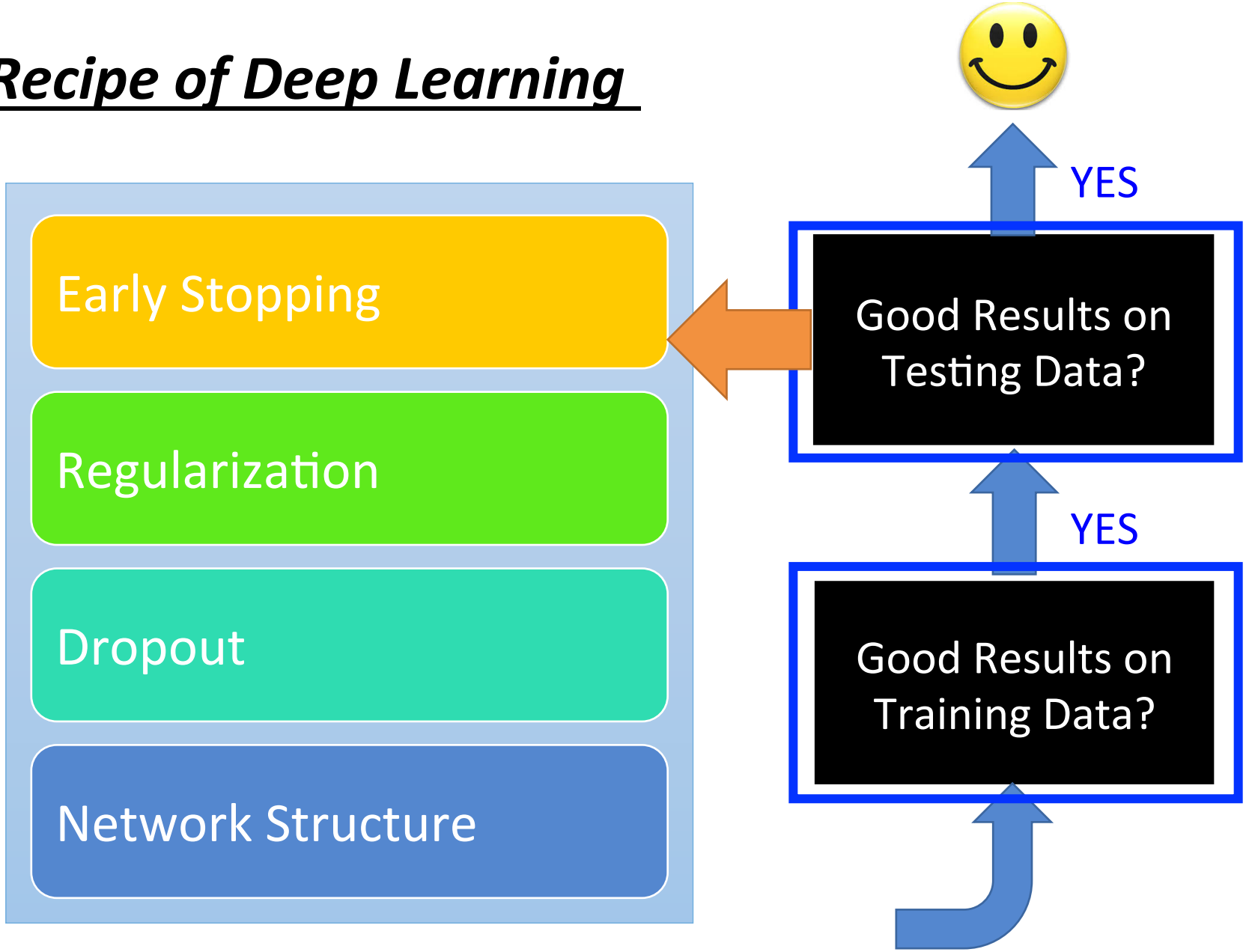
Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ 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)

ReLU, 3 layers



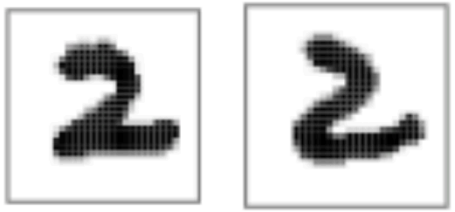
Recipe of Deep Learning



Why Overfitting ?

- Training data and testing data can be different.

Training Data:



Testing Data:

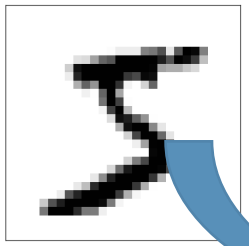
- Learning target is defined by the training data.
- The parameters achieving the learning target do not necessary have good results on the testing data.

Panacea for Overfitting

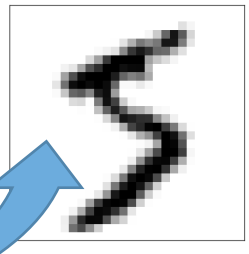
- Have more training data
- ***Create*** more training data (?)

Handwriting recognition:

Original
Training Data:

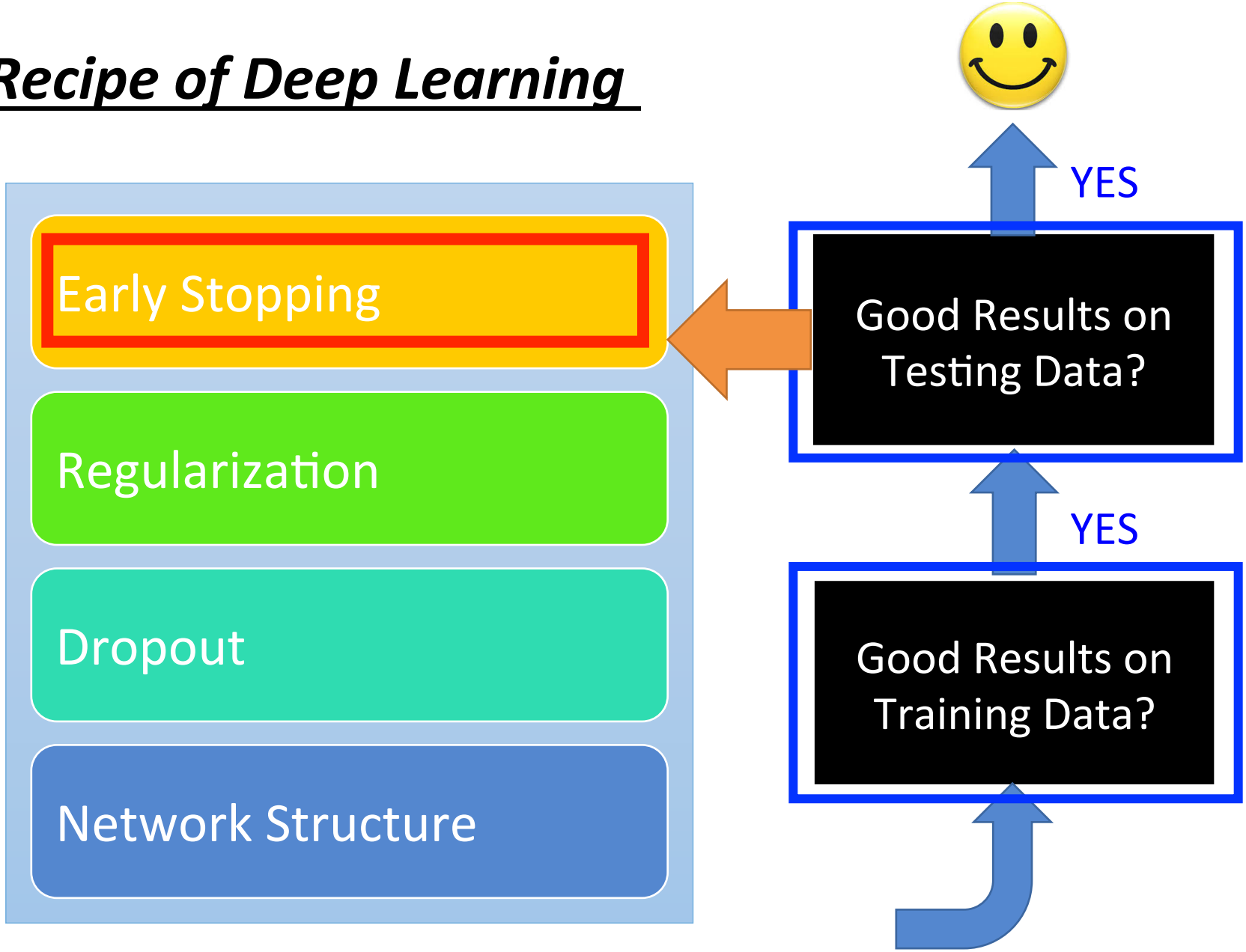


Created
Training Data:

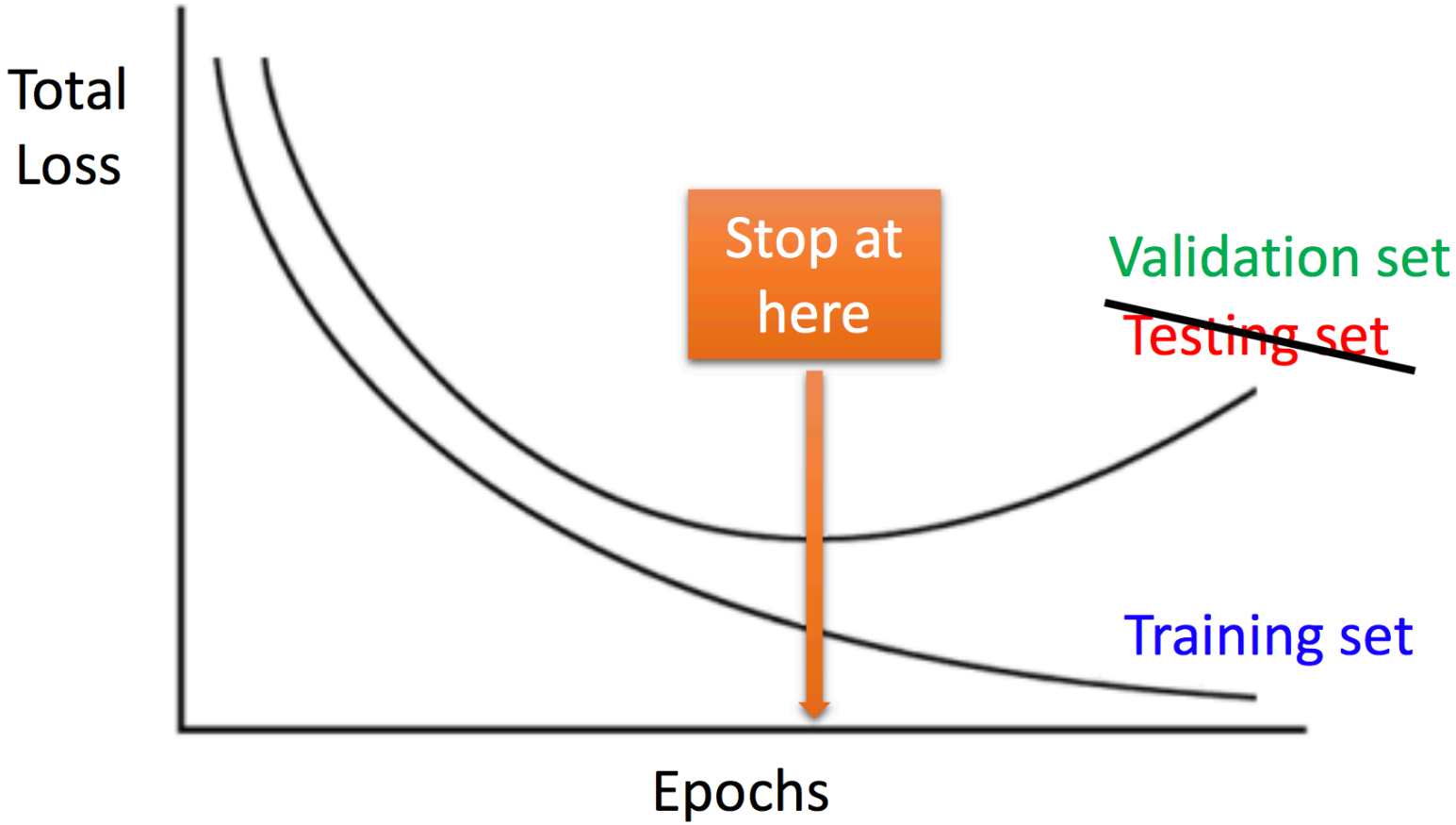


Shift 15 ◦

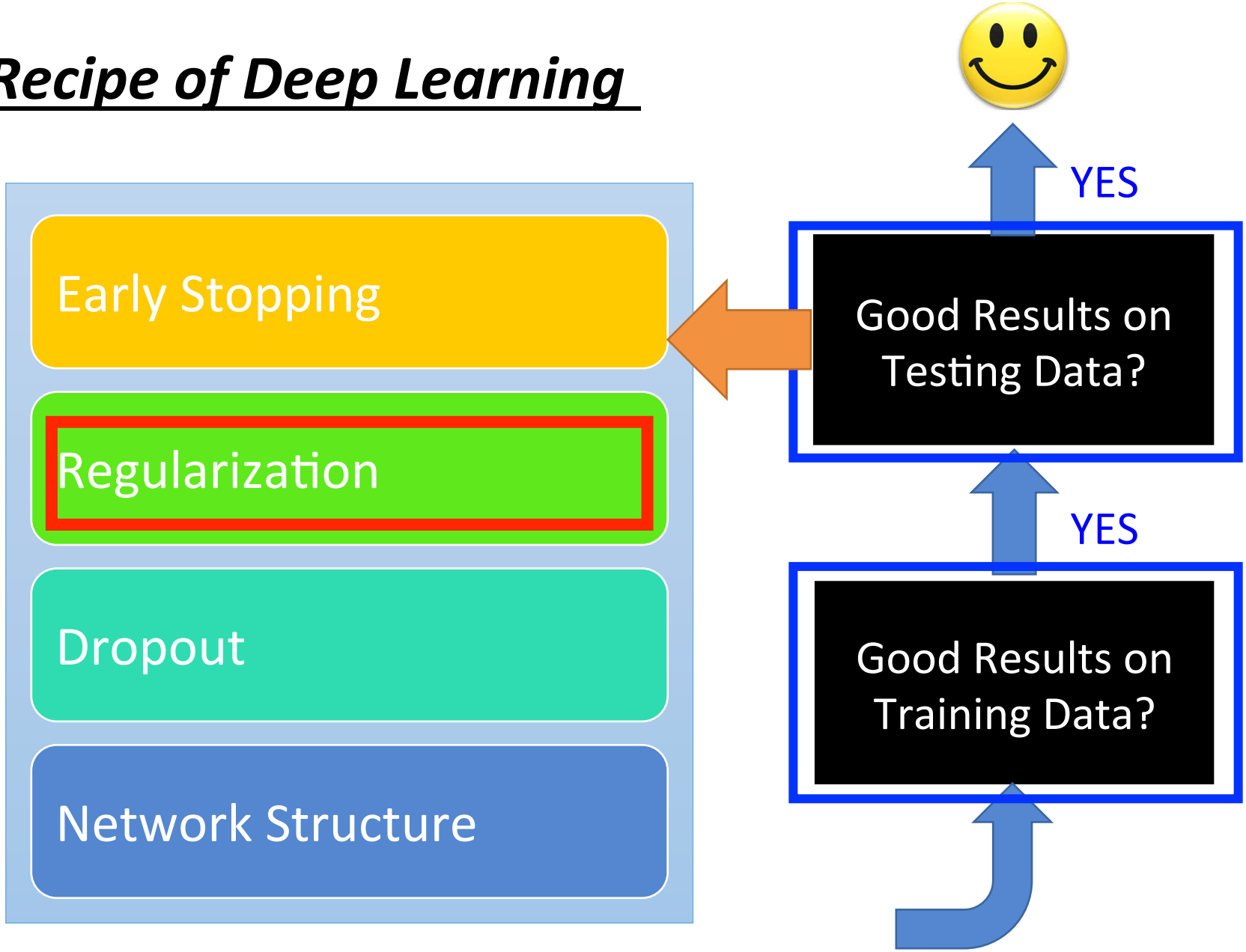
Recipe of Deep Learning



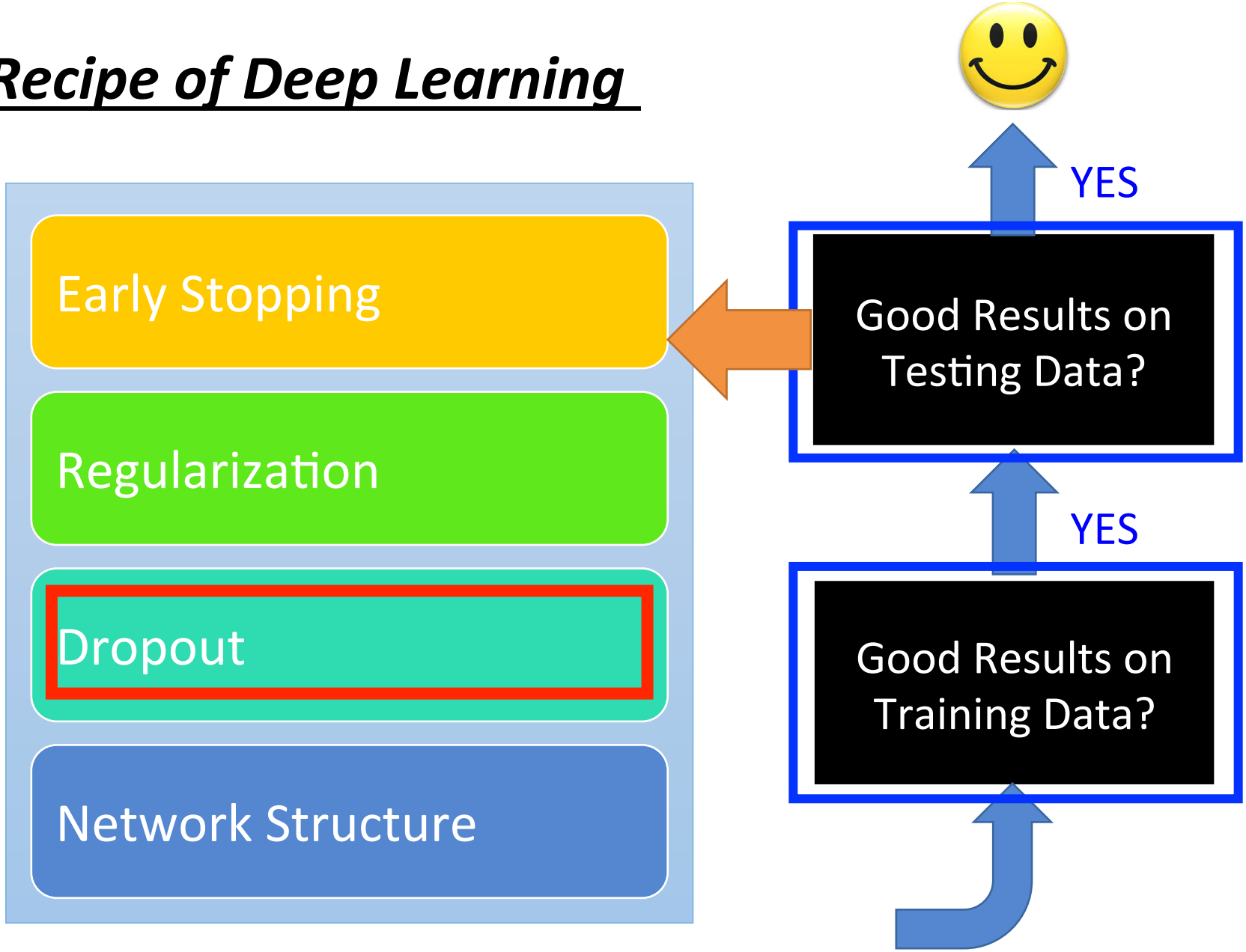
Early Stopping



Recipe of Deep Learning

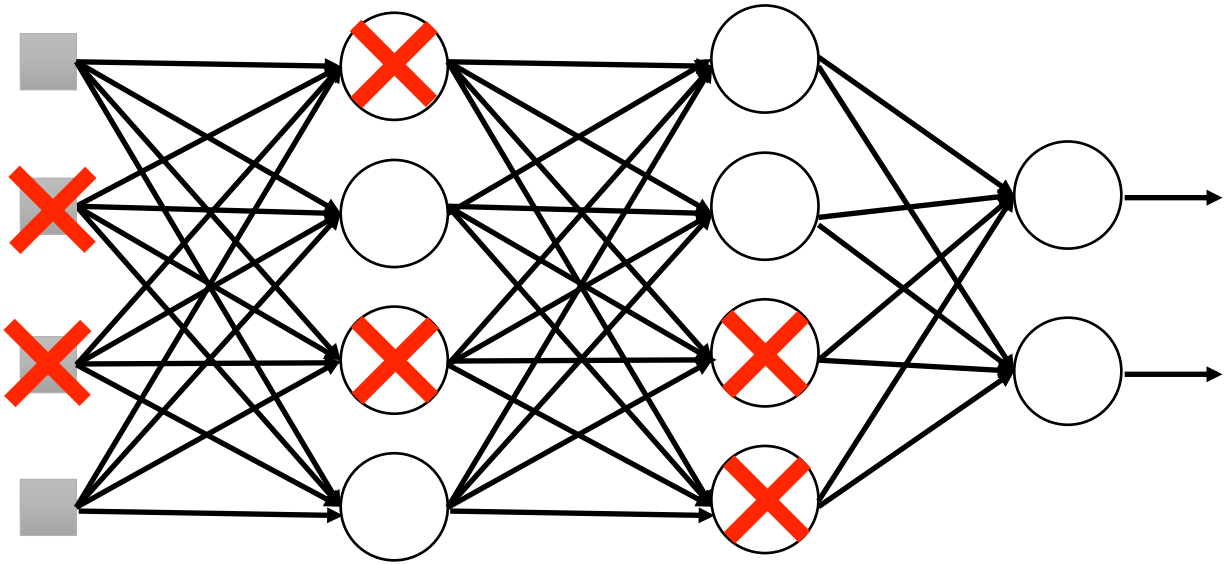


Recipe of Deep Learning



Dropout

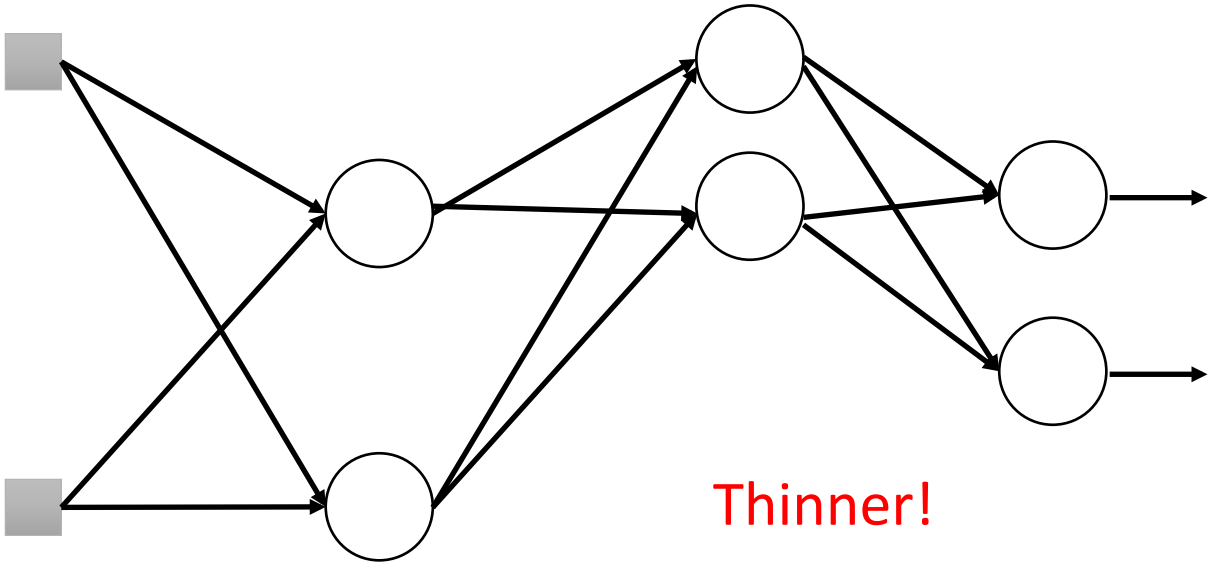
Training:



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

Dropout

Training:



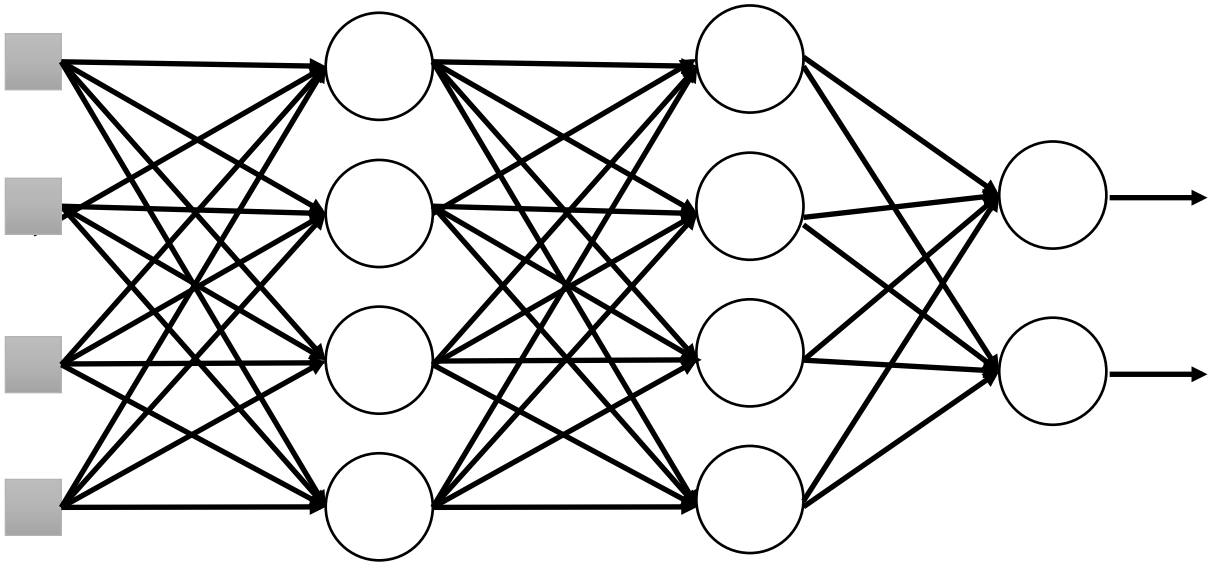
- **Each time before updating the parameters**
  - Each neuron has p% to dropout

    → **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

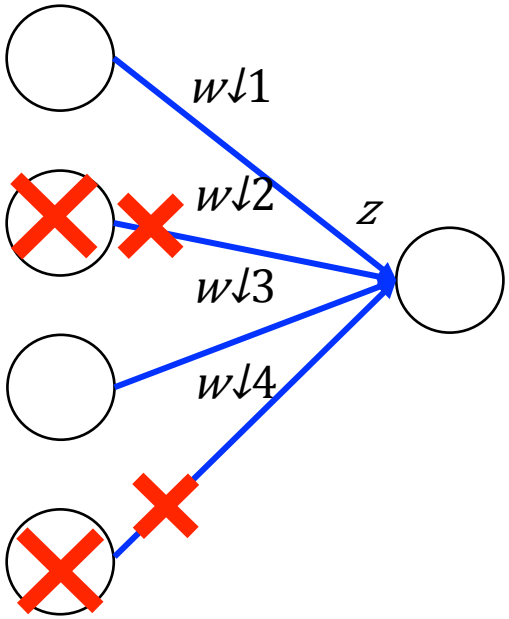
Dropout

Dropout - Intuitive Reason

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

Training of Dropout

Assume dropout rate is 50%

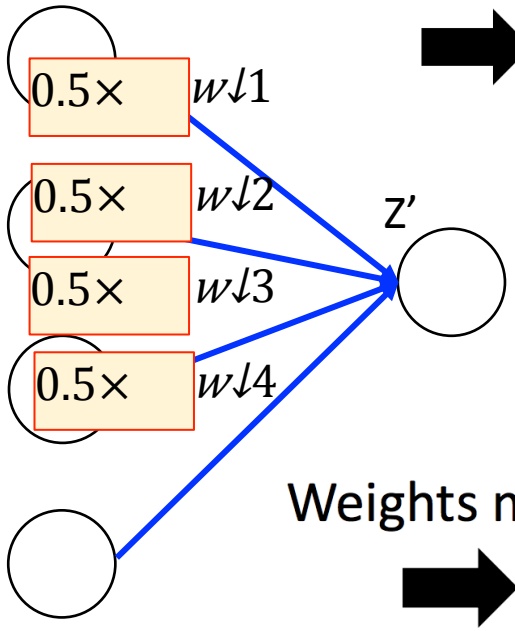


Testing of Dropout

No dropout

Weights from training

$z' \approx 2z$

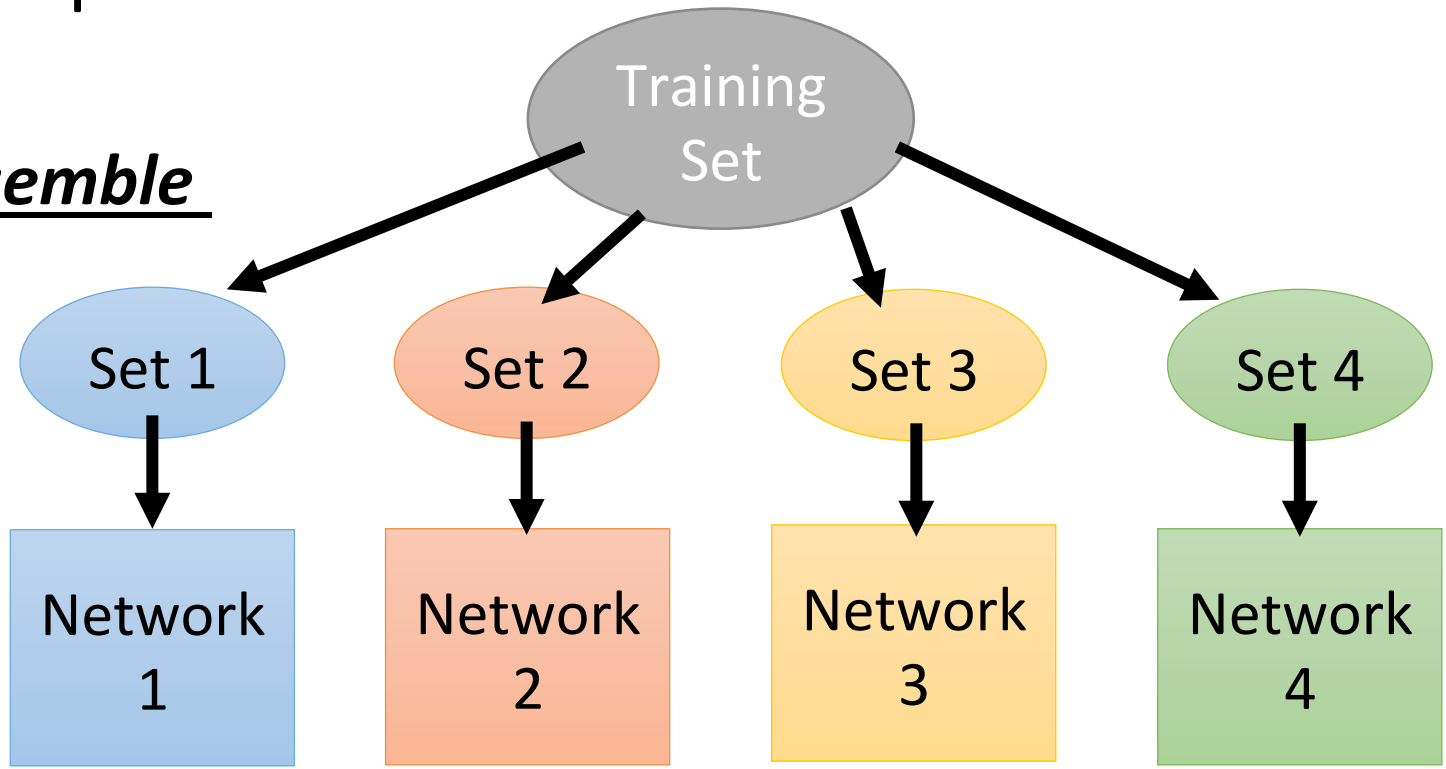


Weights multiply (1-p)%

$z' \approx z$

Dropout is a kind of ensemble.

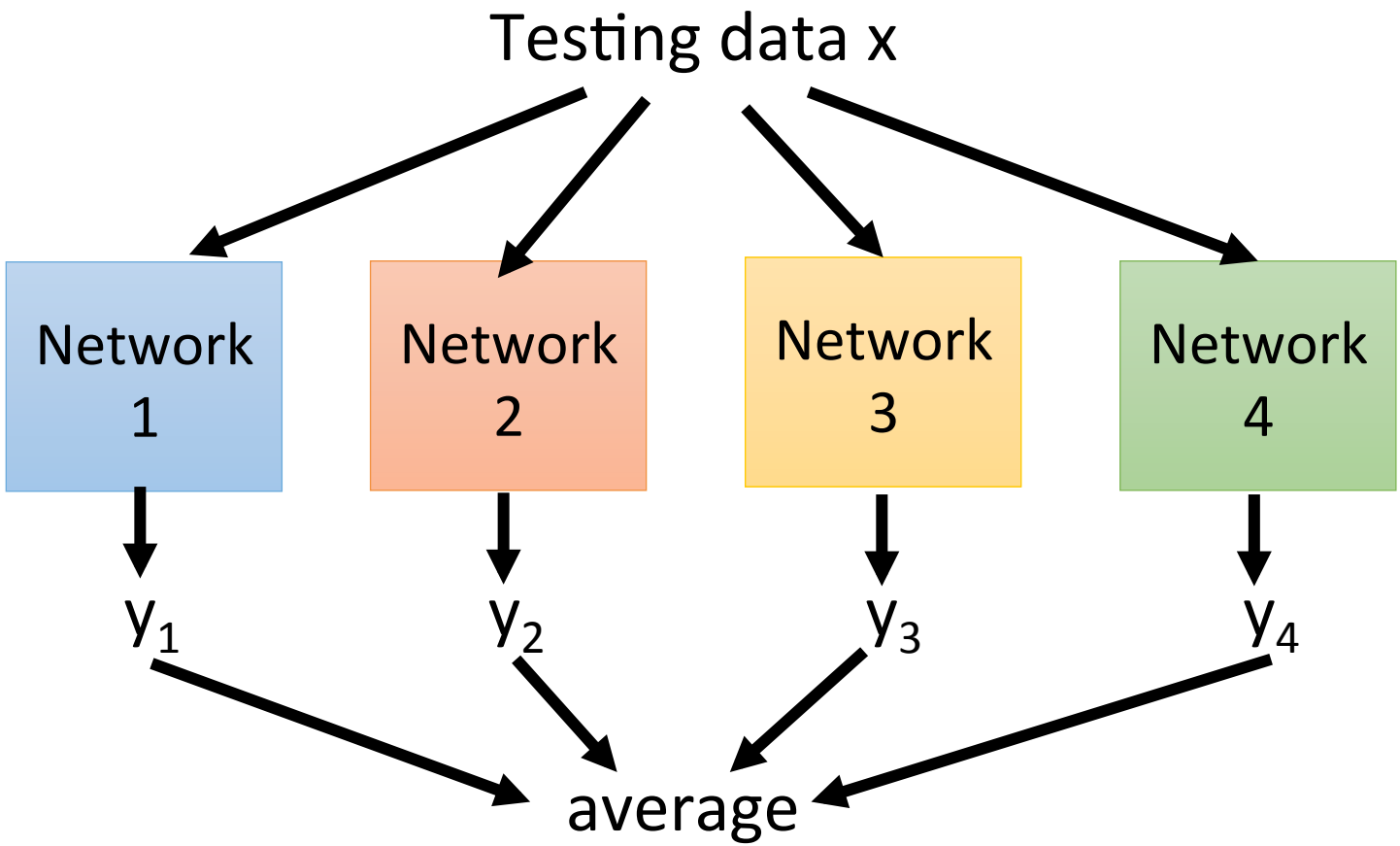
Ensemble



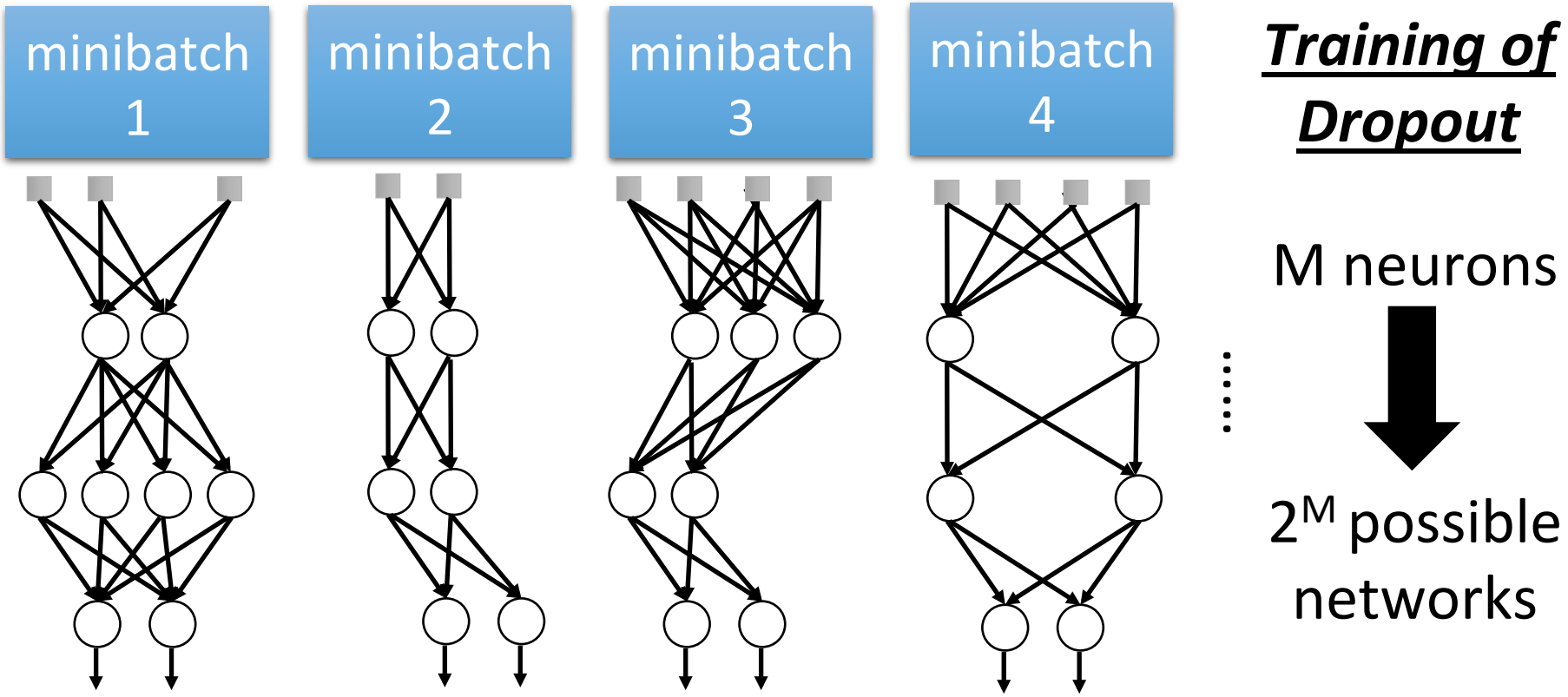
Train a bunch of networks with different structures

Dropout is a kind of ensemble.

Ensemble

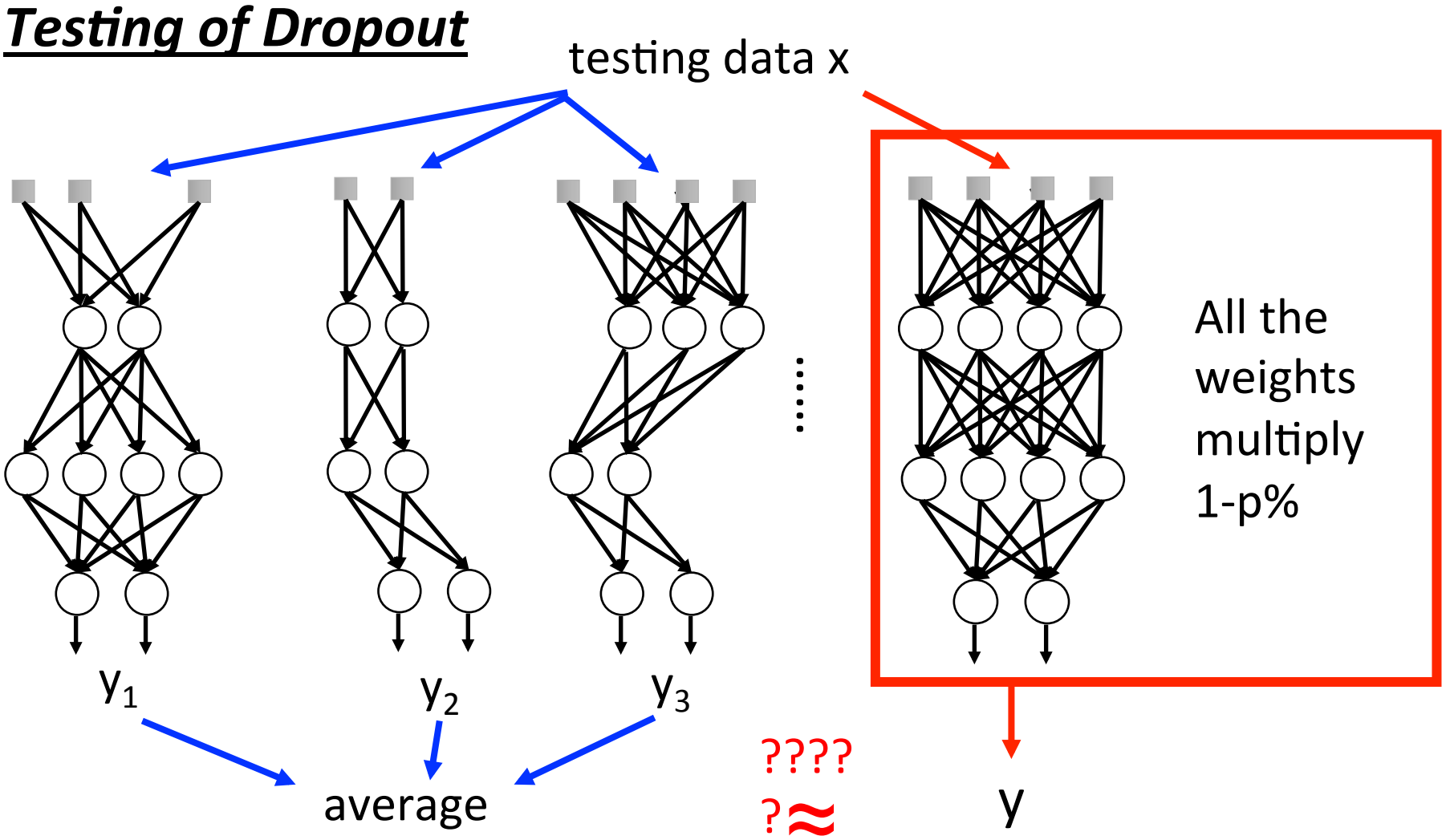


Dropout is a kind of ensemble.



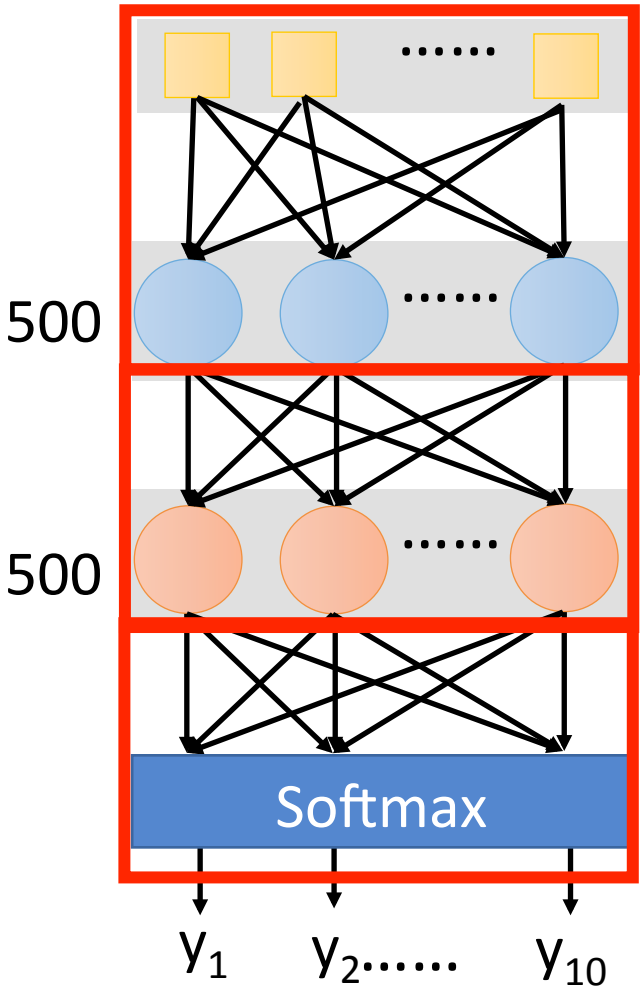
- Using one mini-batch to train one network
- Some parameters in the network are shared

Dropout is a kind of ensemble.



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




```
model = Sequential()
```

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

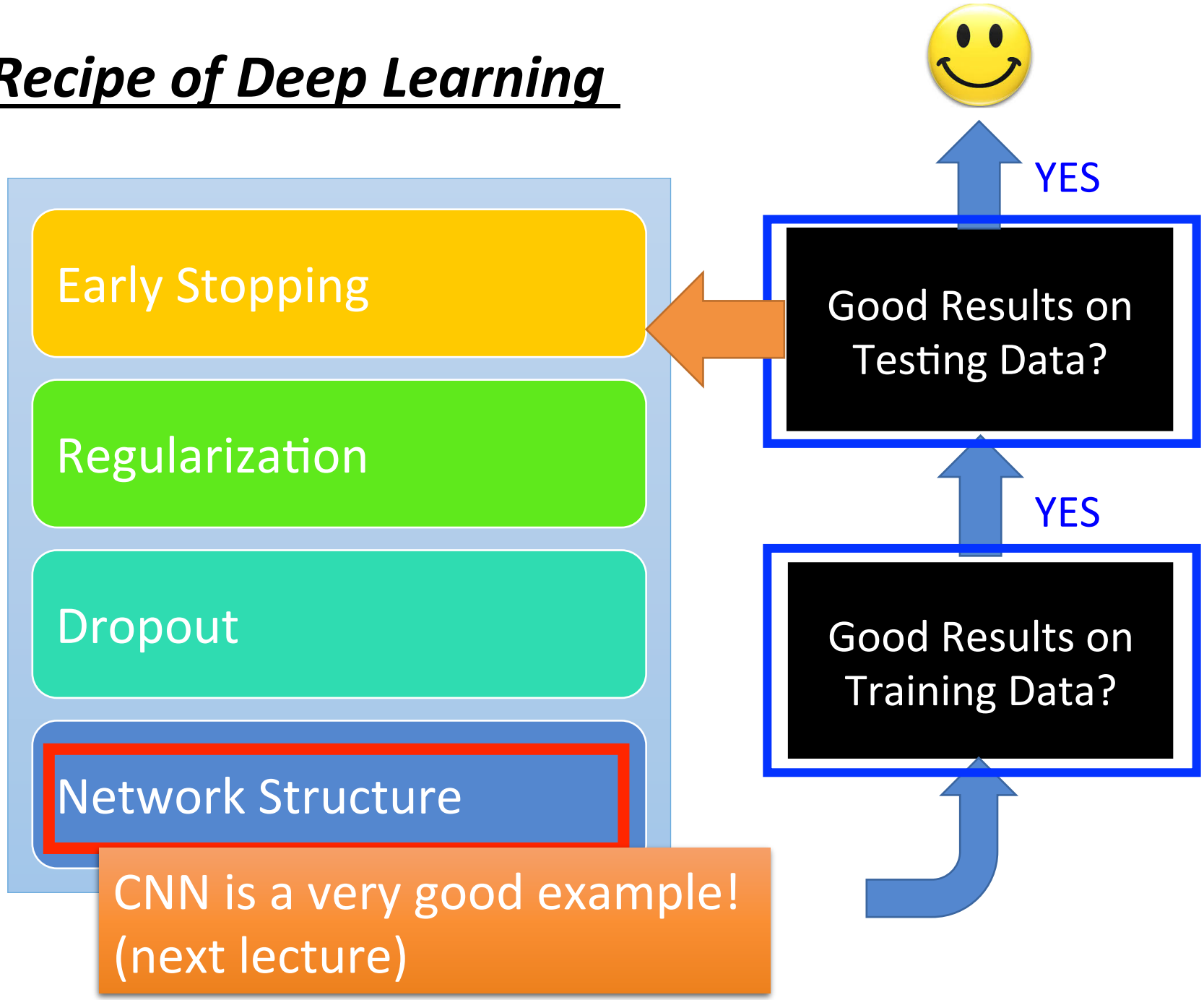
model.add(dropout(0.8))

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

model.add(dropout(0.8))

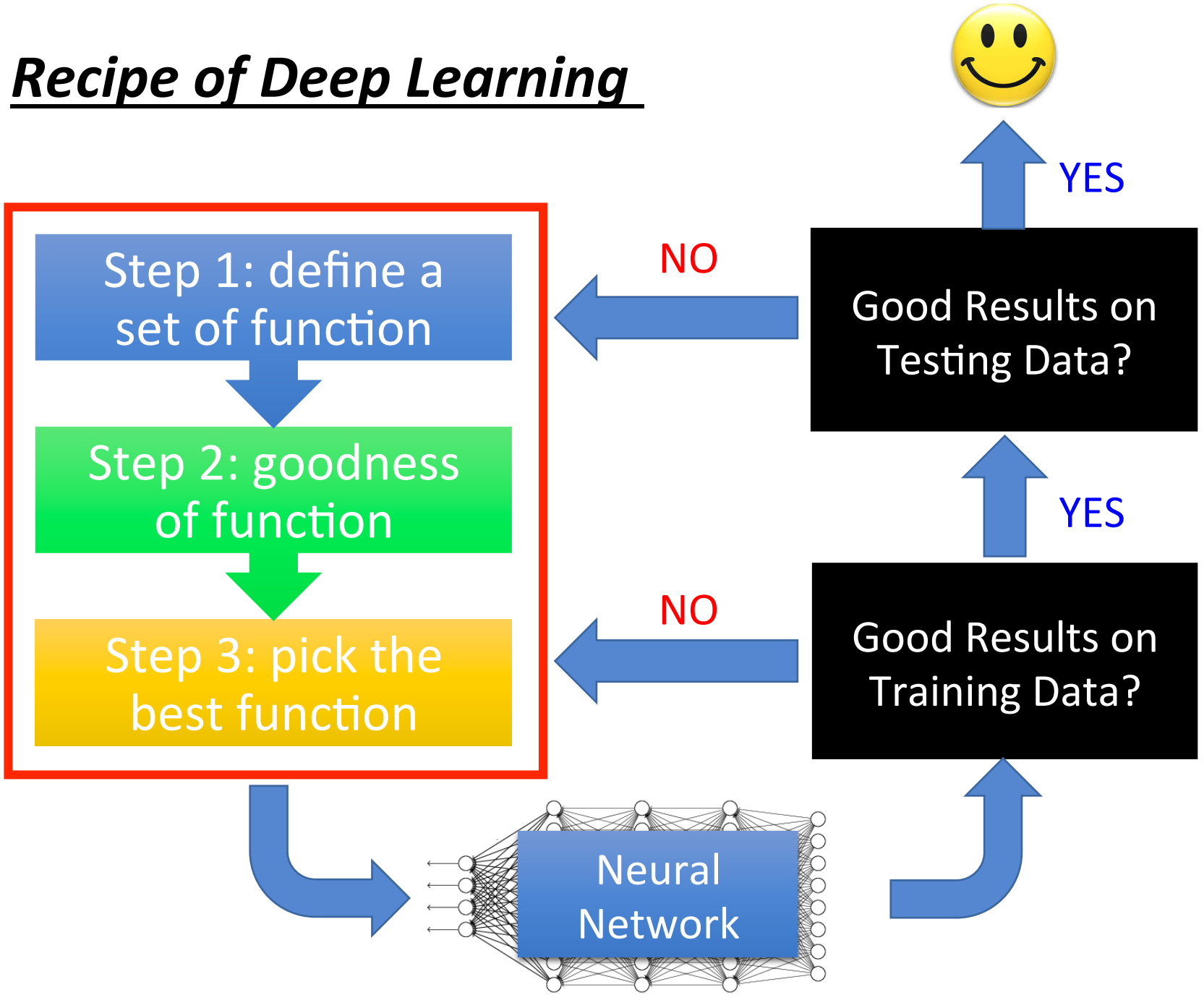
```
model.add( Dense(output_dim=10 ) )
model.add( Activation('softmax') )
```

Recipe of Deep Learning



Concluding Remarks of Part I

Recipe of Deep Learning



Part II:
Variants of Neural Networks

Variants of Neural Networks

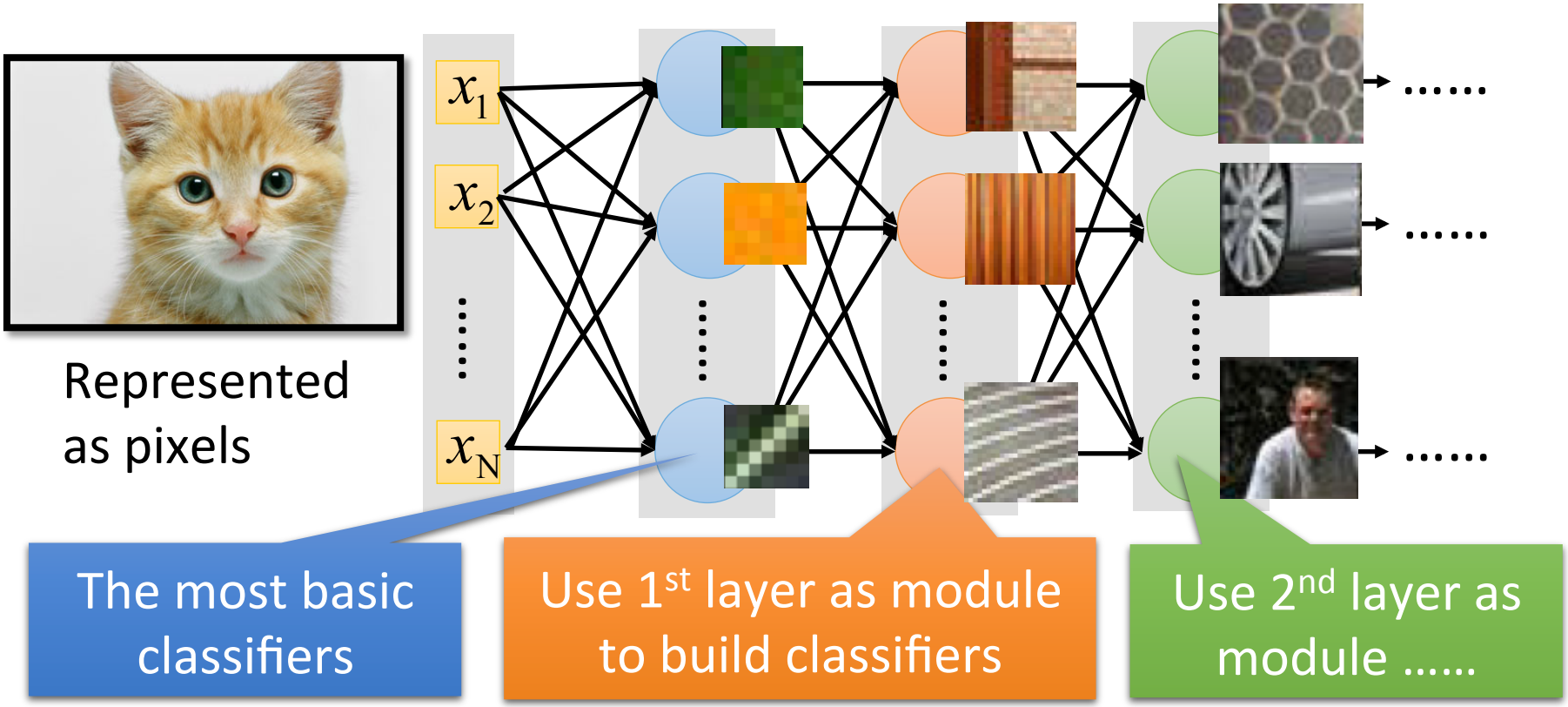
Convolutional Neural Network (CNN)

Widely used in image processing

Recurrent Neural Network (RNN)

Why CNN for Image?

[Zeiler, M. D., *ECCV 2014*]



Can the network be simplified by considering the properties of images?

Why CNN for Image

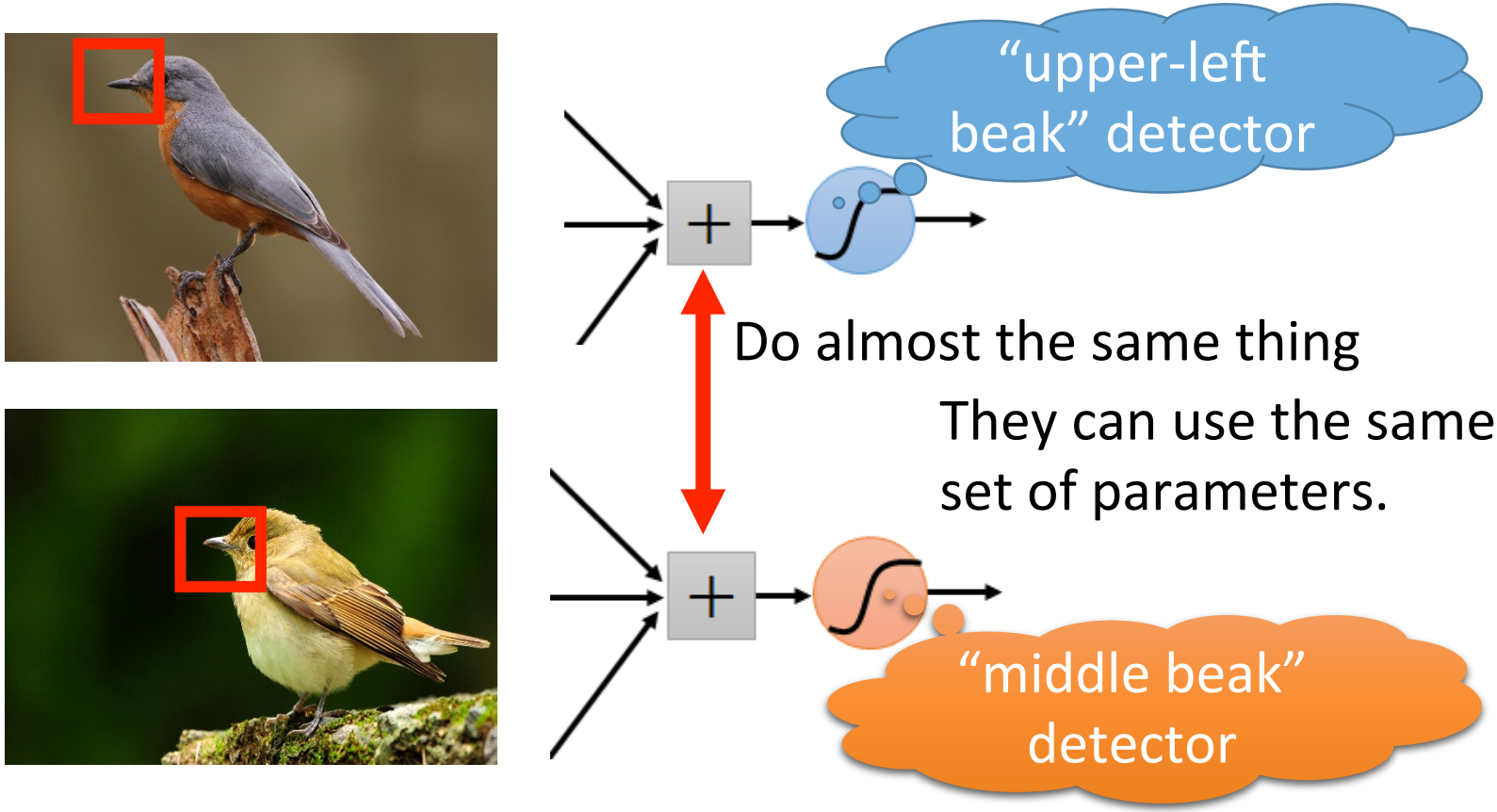
- Some patterns are much smaller than the whole image

A neuron does not have to see the whole image to discover the pattern.

Connecting to small region with less parameters

Why CNN for Image

- The same patterns appear in different regions.



Why CNN for Image

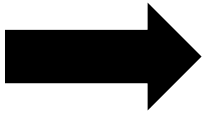
- Subsampling the pixels will not change the object

bird

subsampling

bird

We can subsample the pixels to make image smaller



Less parameters for the network to process the image

Three Steps for Deep Learning

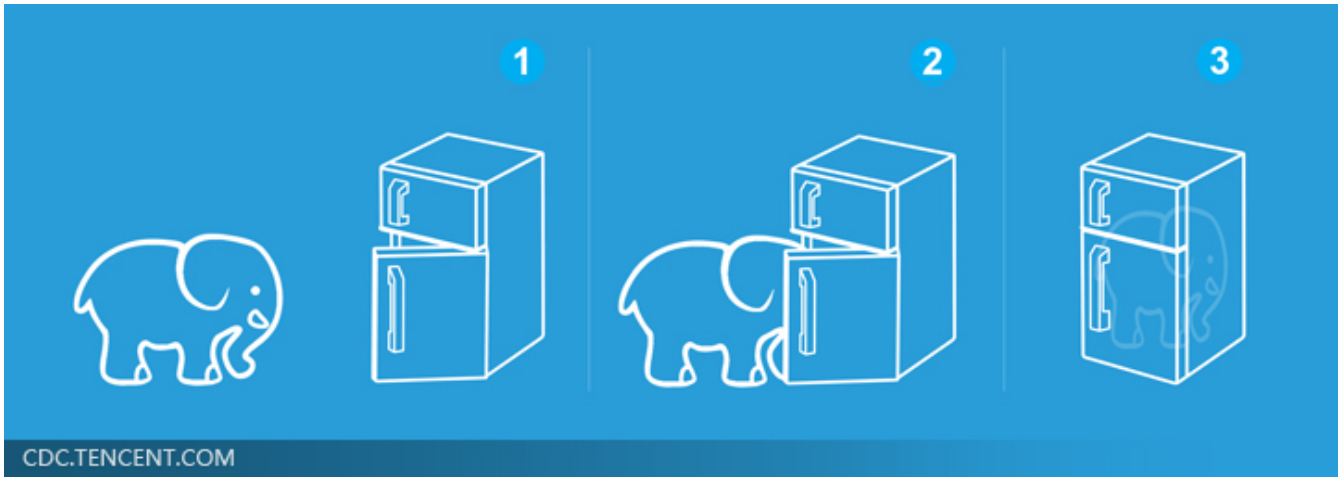
Step 1:

Convolutional Neural Network

Step 2: goodness of function

Step 3: pick the best function

Deep Learning is so simple ……



The whole CNN

Convolution

Max Pooling

Convolution

Max Pooling

Can repeat many times

Flatten

Fully Connected Feedforward network

cat dog ……

The whole CNN

Property 1

- Some patterns are much smaller than the whole image

Property 2

- The same patterns appear in different regions.

Property 3

- Subsampling the pixels will not change the object

Convolution

Max Pooling

Convolution

Max Pooling

Can repeat many times

Flatten

The whole CNN

Convolution

Max Pooling

Convolution

Max Pooling

Can repeat many times

Flatten

Fully Connected Feedforward network

cat dog ……

CNN – Convolution

Those are the network parameters to be learned.

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

Matrix

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

Matrix

Property 1

Each filter detects a small pattern (3 x 3).

CNN – Convolution

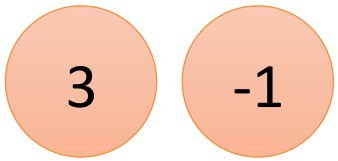
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image



CNN – Convolution

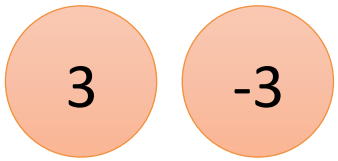
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

If stride=2

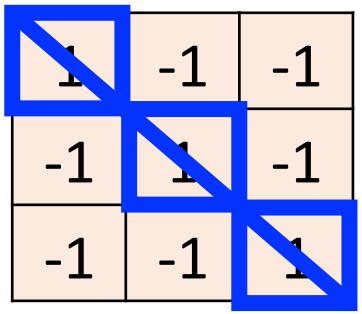
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image



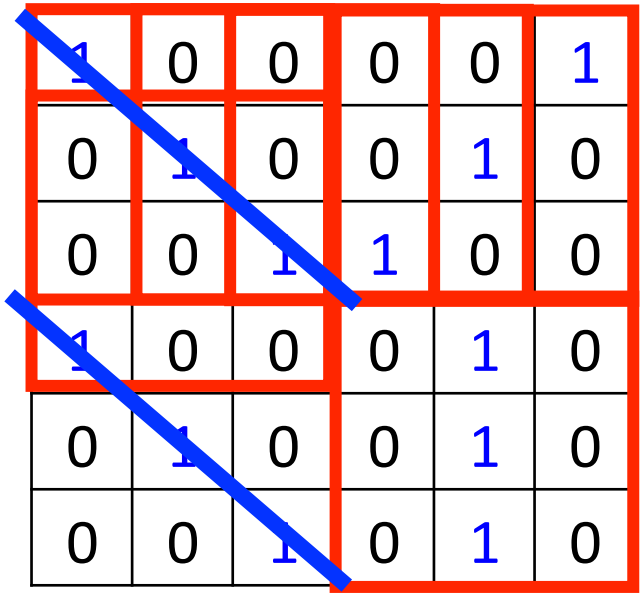
We set stride=1 below

CNN – Convolution

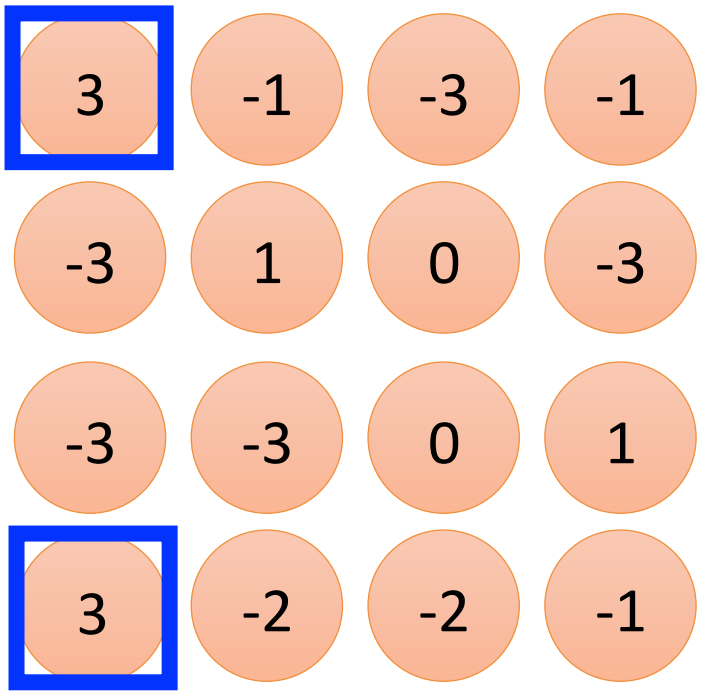


Filter 1

stride=1



6 x 6 image



Property 2

CNN – Convolution

-1	1	-1
-1	1	-1
-1	1	-1

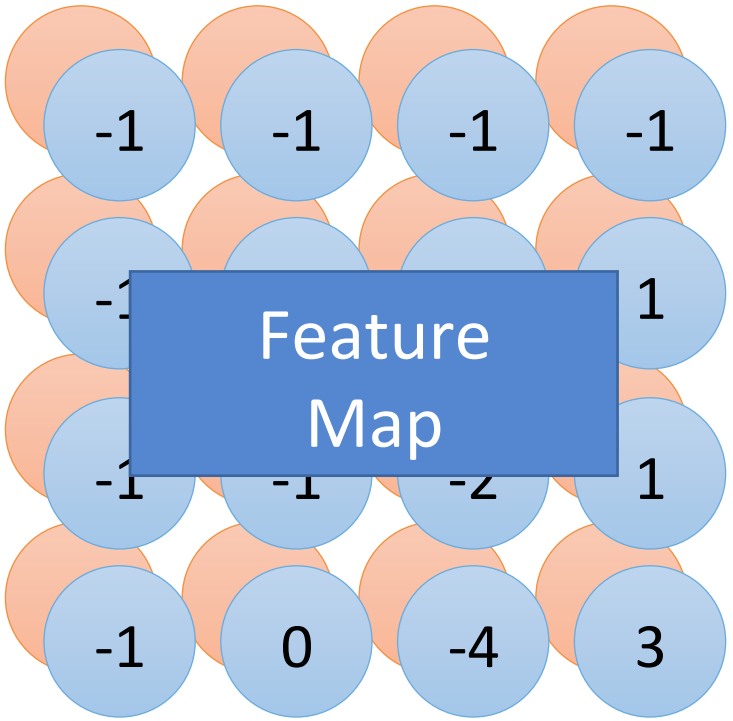
Filter 2

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

Do the same process for every filter

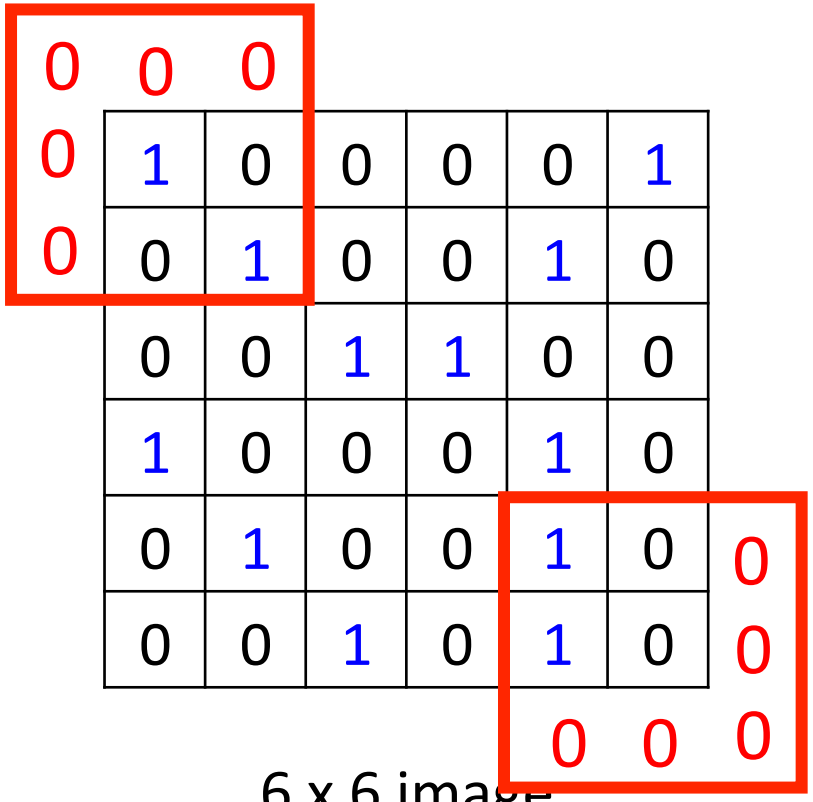


4 x 4 image

CNN – Zero Padding

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

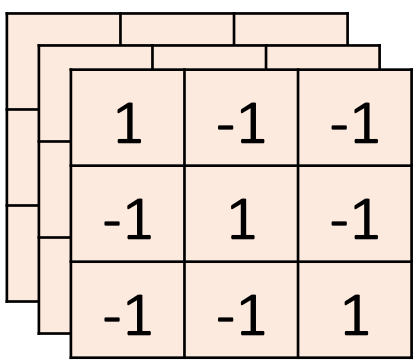


6 x 6 image

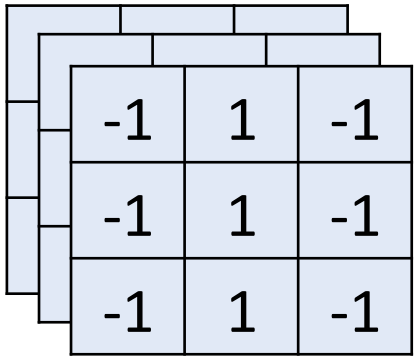
You will get another 6 x 6 images in this way

Zero padding

CNN – Colorful image

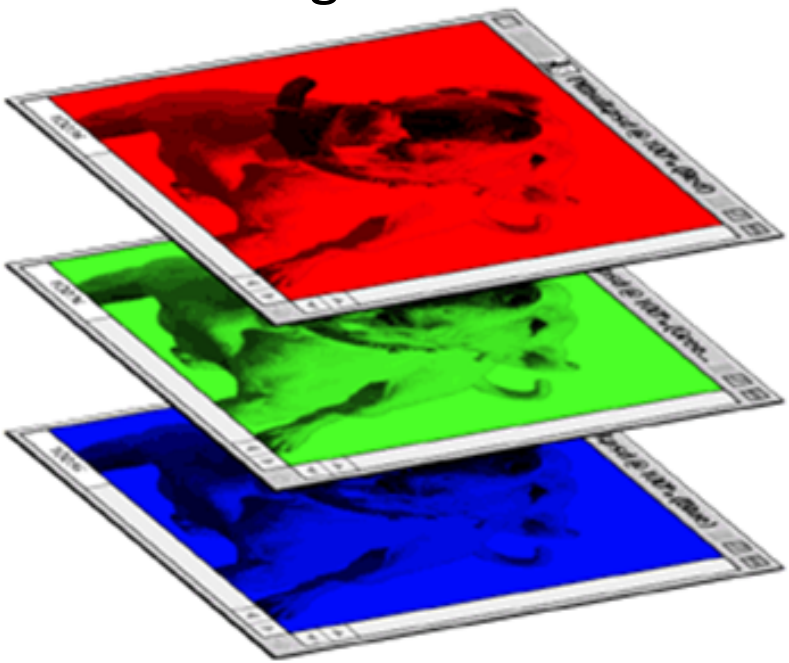


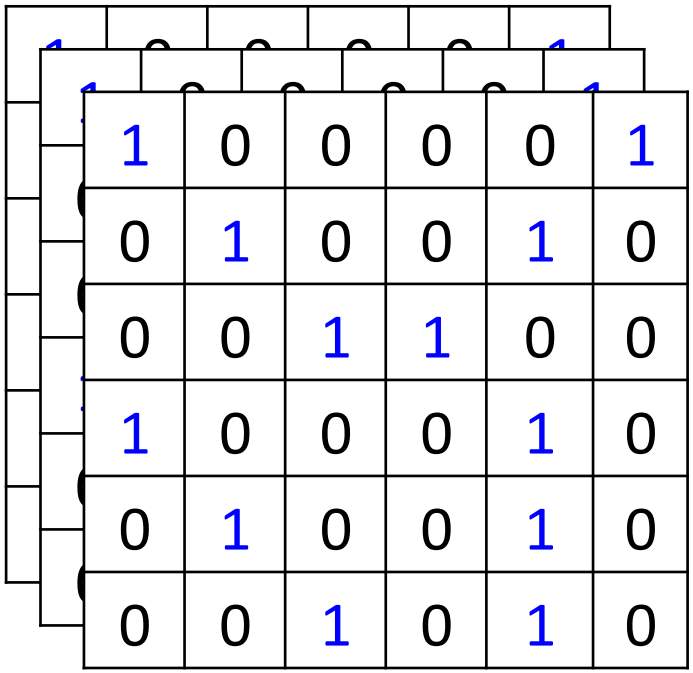
Filter 1



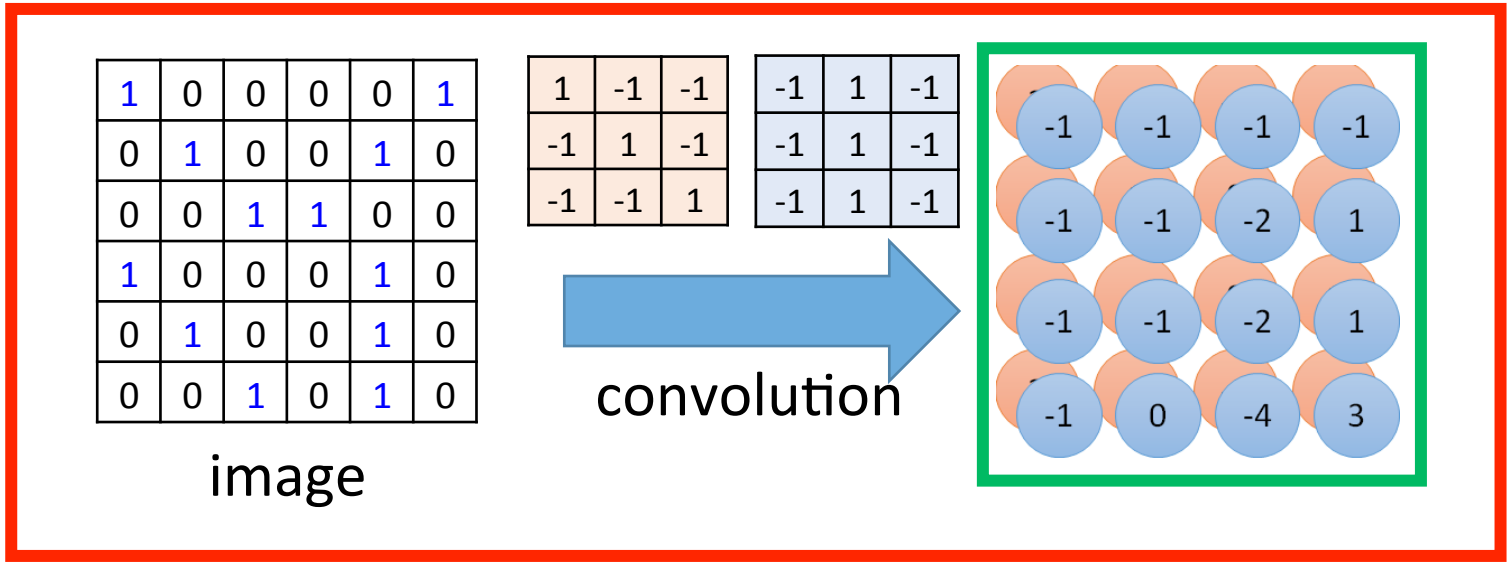
Filter 2

Colorful image



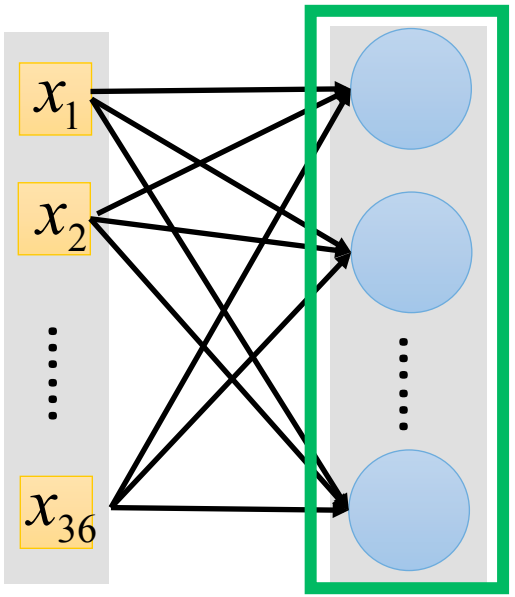


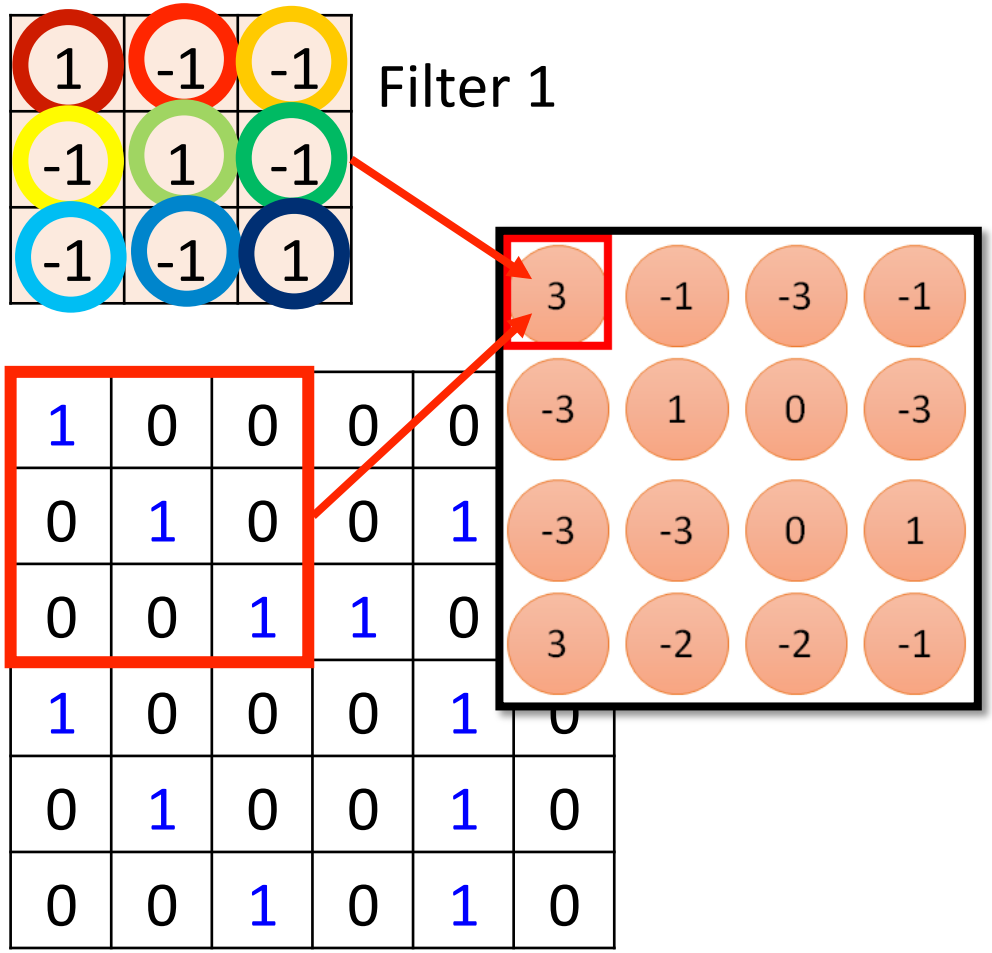
Convolution v.s. Fully Connected



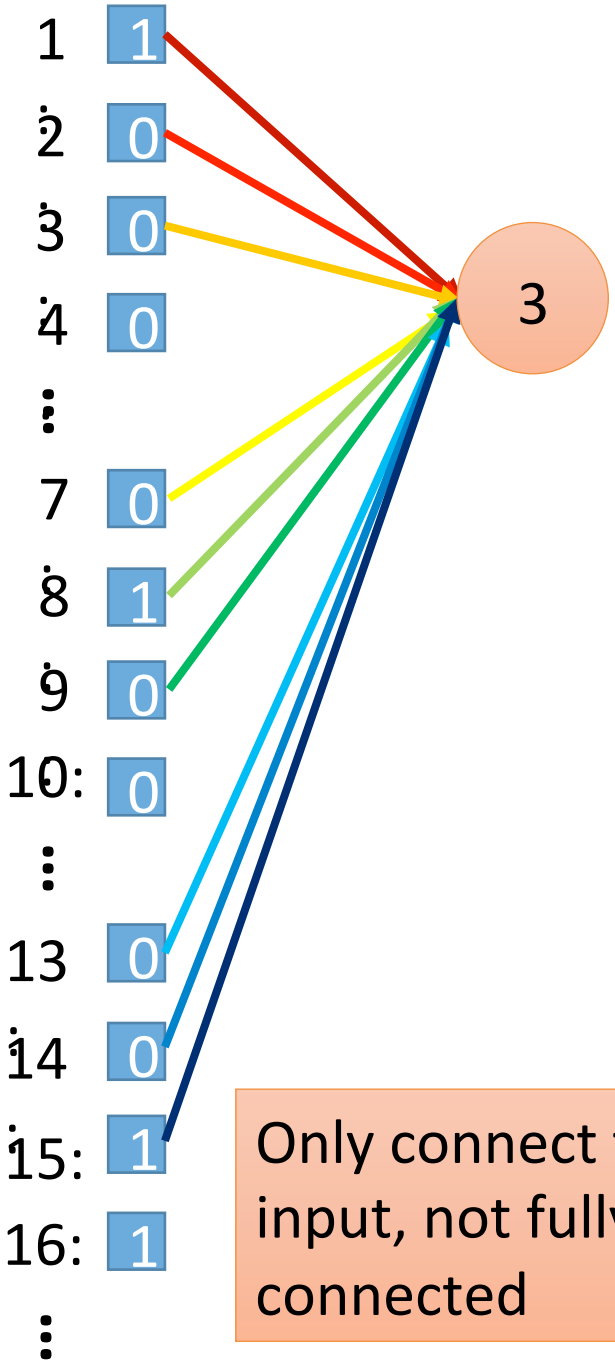
Fully-connected

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

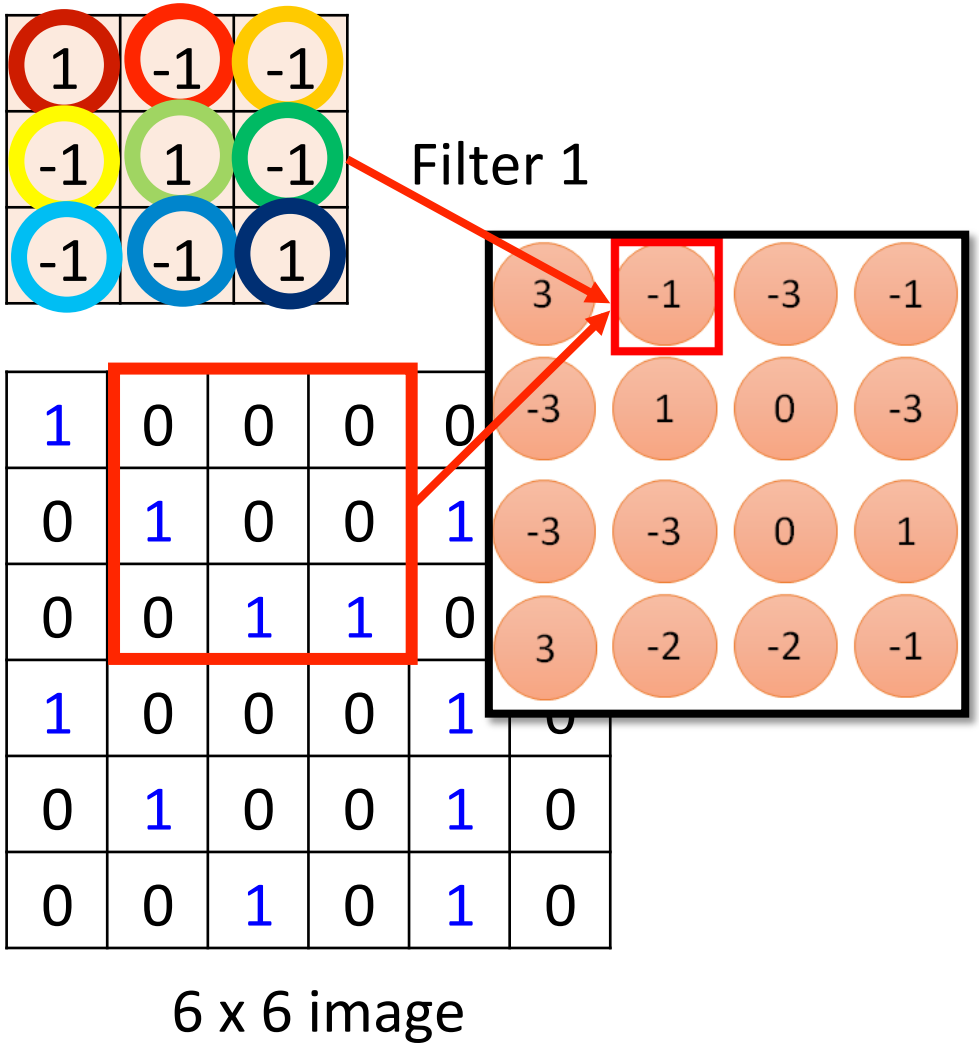




Less parameters!

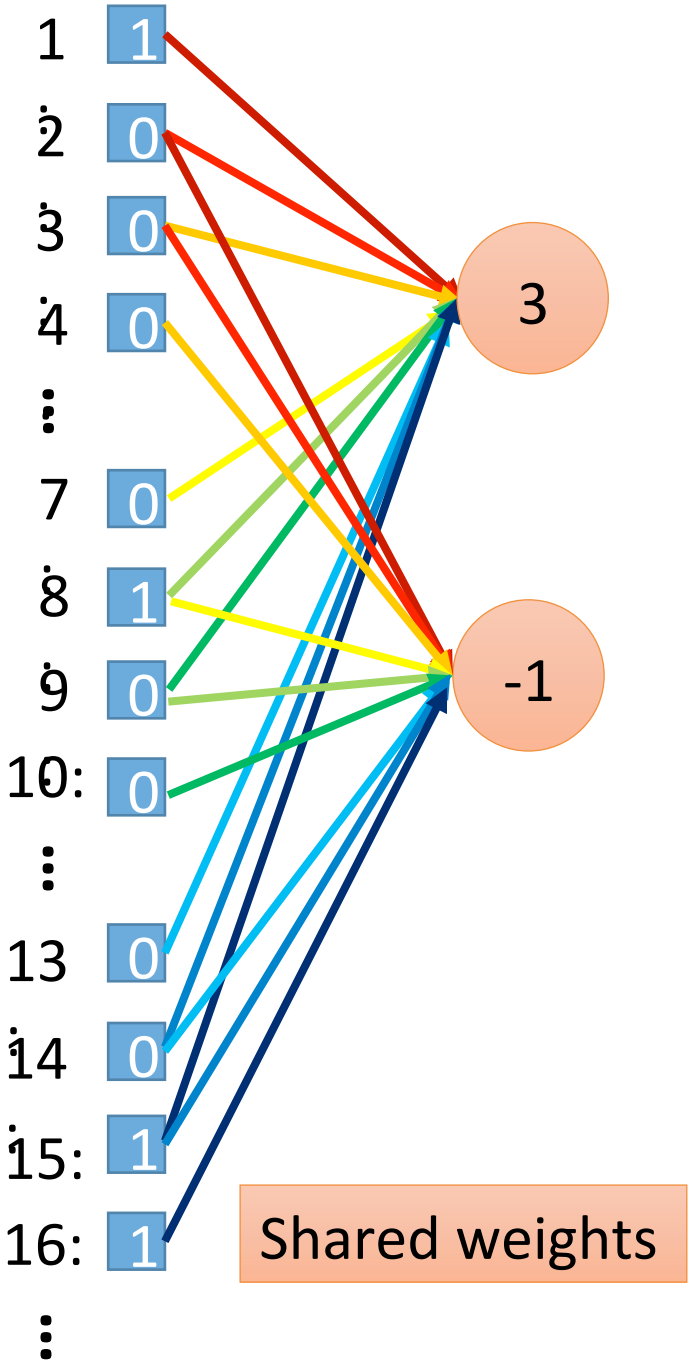


Only connect to 9 input, not fully connected

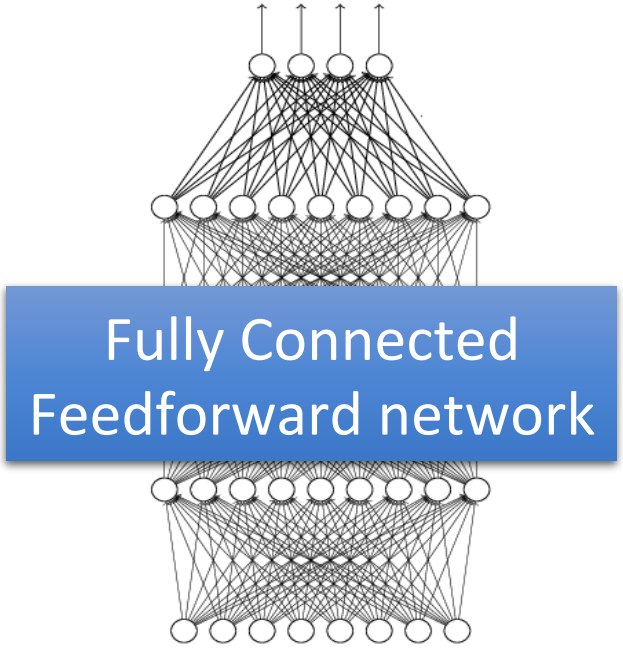


Less parameters!

Even less parameters!



The whole CNN



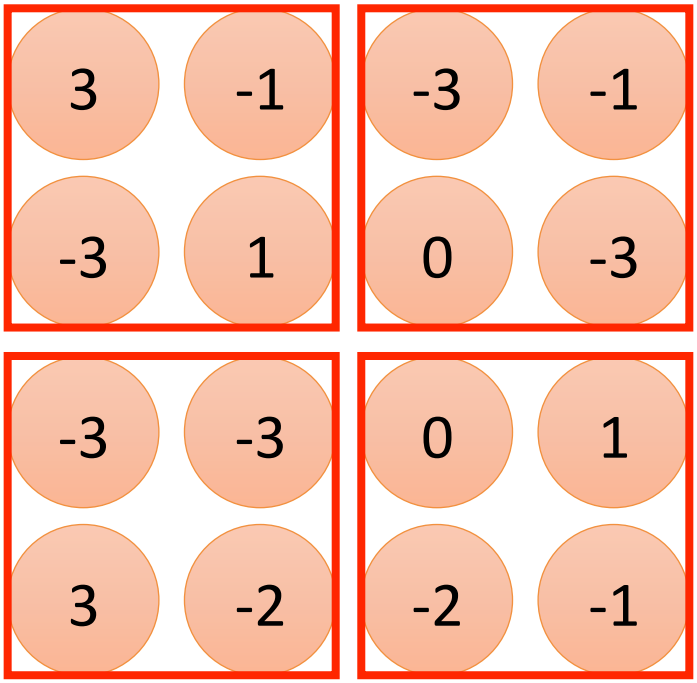
CNN – Max Pooling

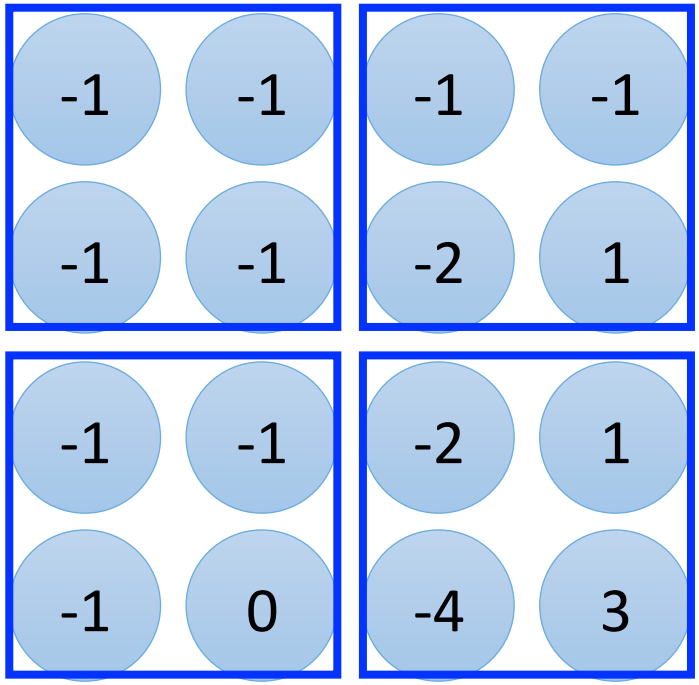
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

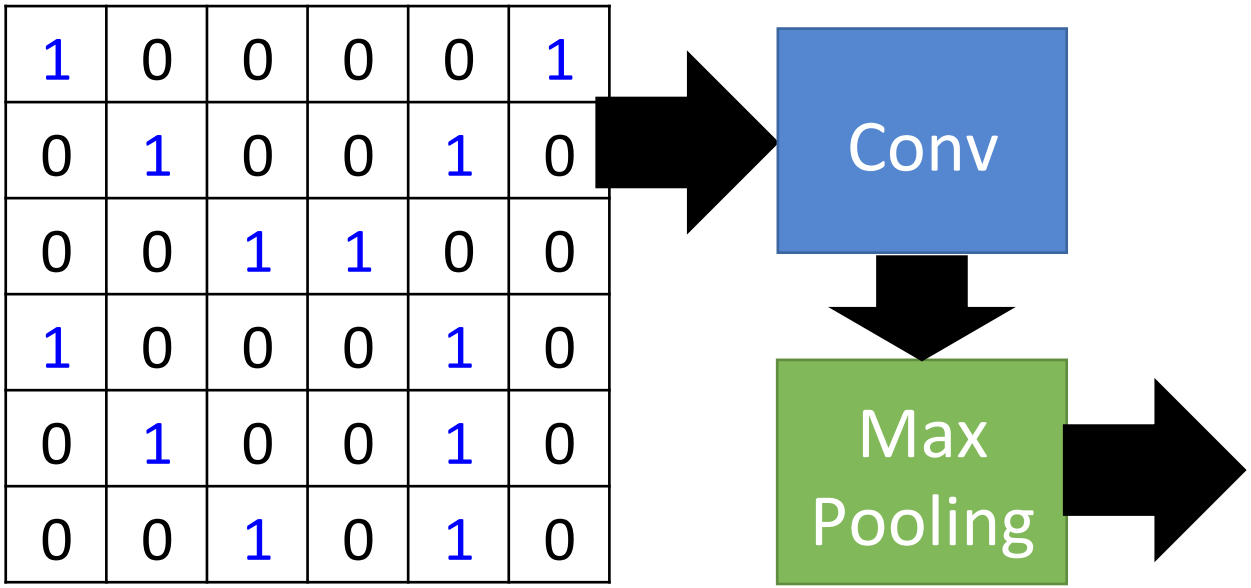
-1	1	-1
-1	1	-1
-1	1	-1

Filter 2



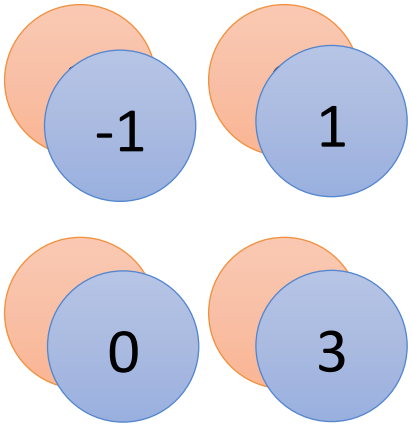


CNN – Max Pooling



6 x 6 image

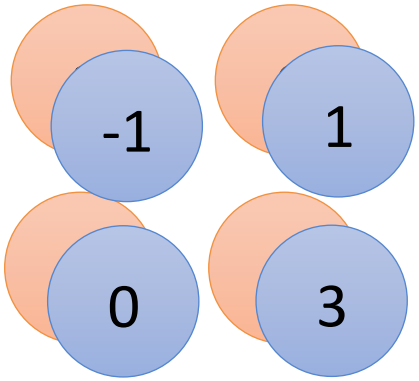
New “image” but smaller



2 x 2 image

Each filter is a channel

The whole CNN



A new “image”

Convolution

Max Pooling

Convolution

Max Pooling

Can repeat many times

Smaller than the original image

The number of the channel is the number of filters

The whole CNN

Convolution

Max Pooling

A new image

Convolution

Max Pooling

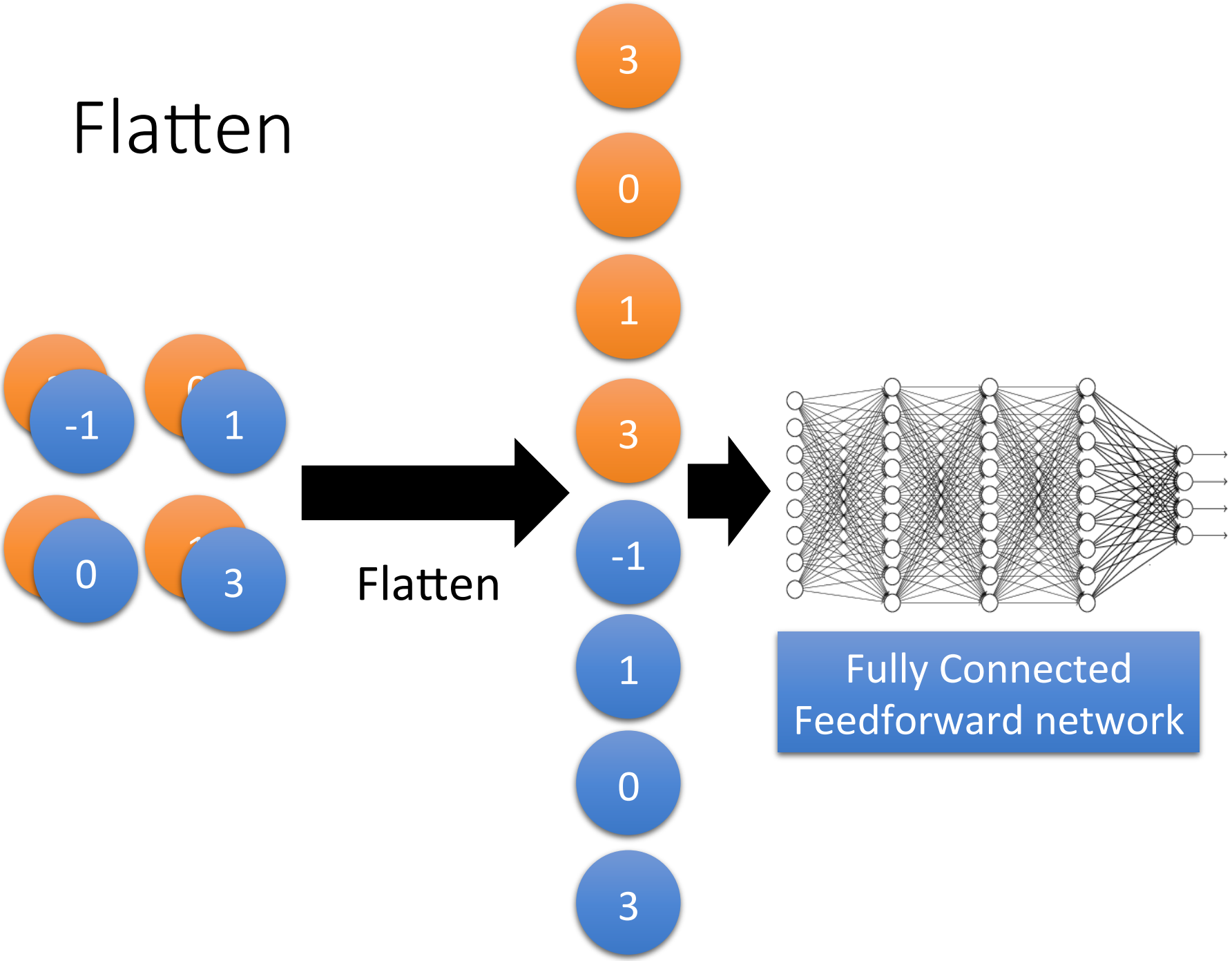
A new image

Flatten

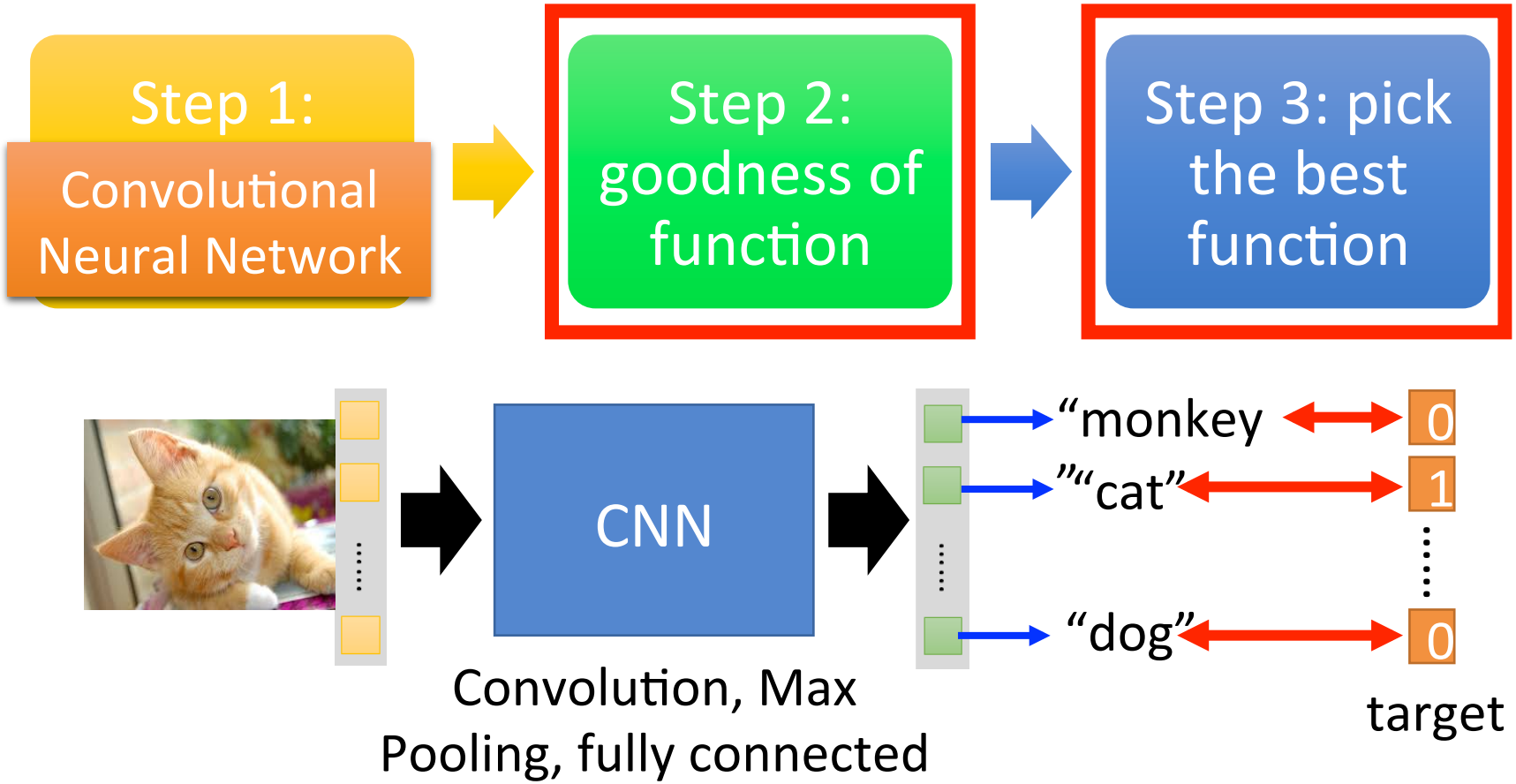
Fully Connected Feedforward network

cat dog ……

Flatten


-1
1
0
3
Flatten
3
0
1
3
-1
1
0
3
Fully Connected
Feedforward network

Convolutional Neural Network

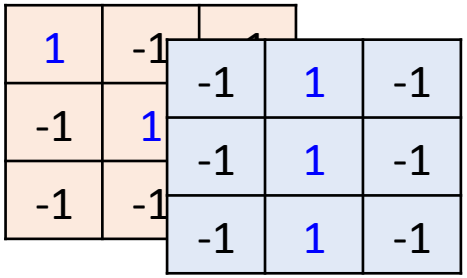


Learning: Nothing special, just gradient descent ……

CNN in Keras

Only modified the ***network structure*** and ***input format (vector -> 3-D tensor)***

```
model2.add( Convolution2D( 25,3,3,
            input_shape=(1,28,28) ) )
```



……

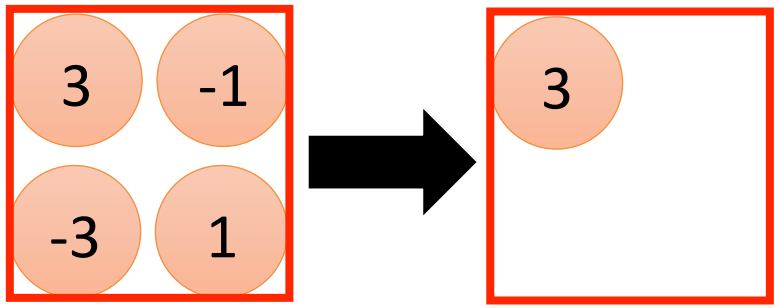
There are 25 3x3 filters.

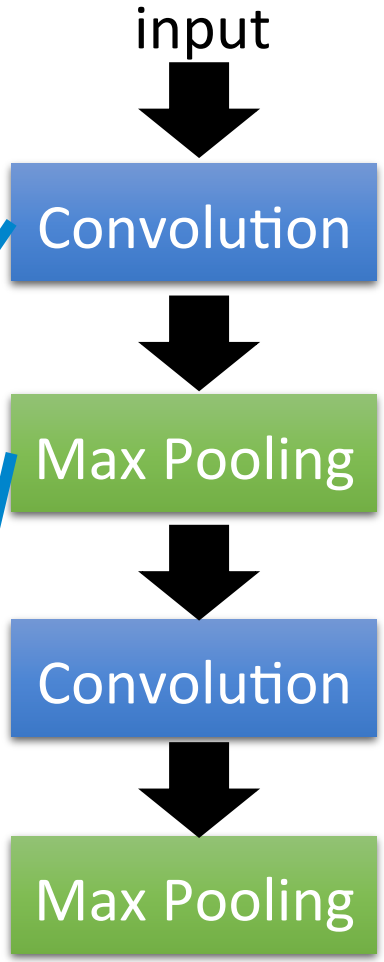
Input_shape = (1 , 28 , 28)

1: black/weight, 3: RGB

28 x 28 pixels

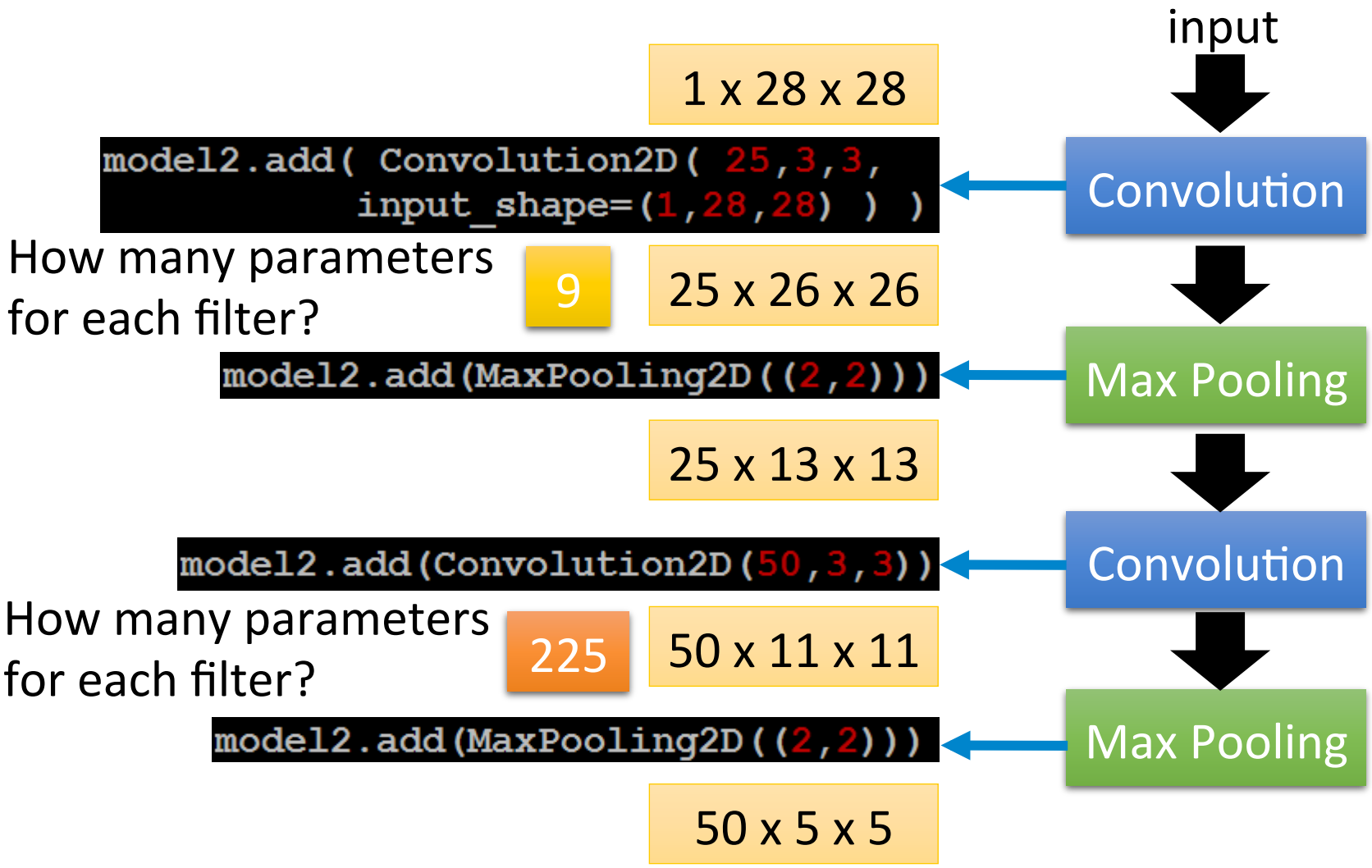
```
model2.add(MaxPooling2D((2,2)))
```





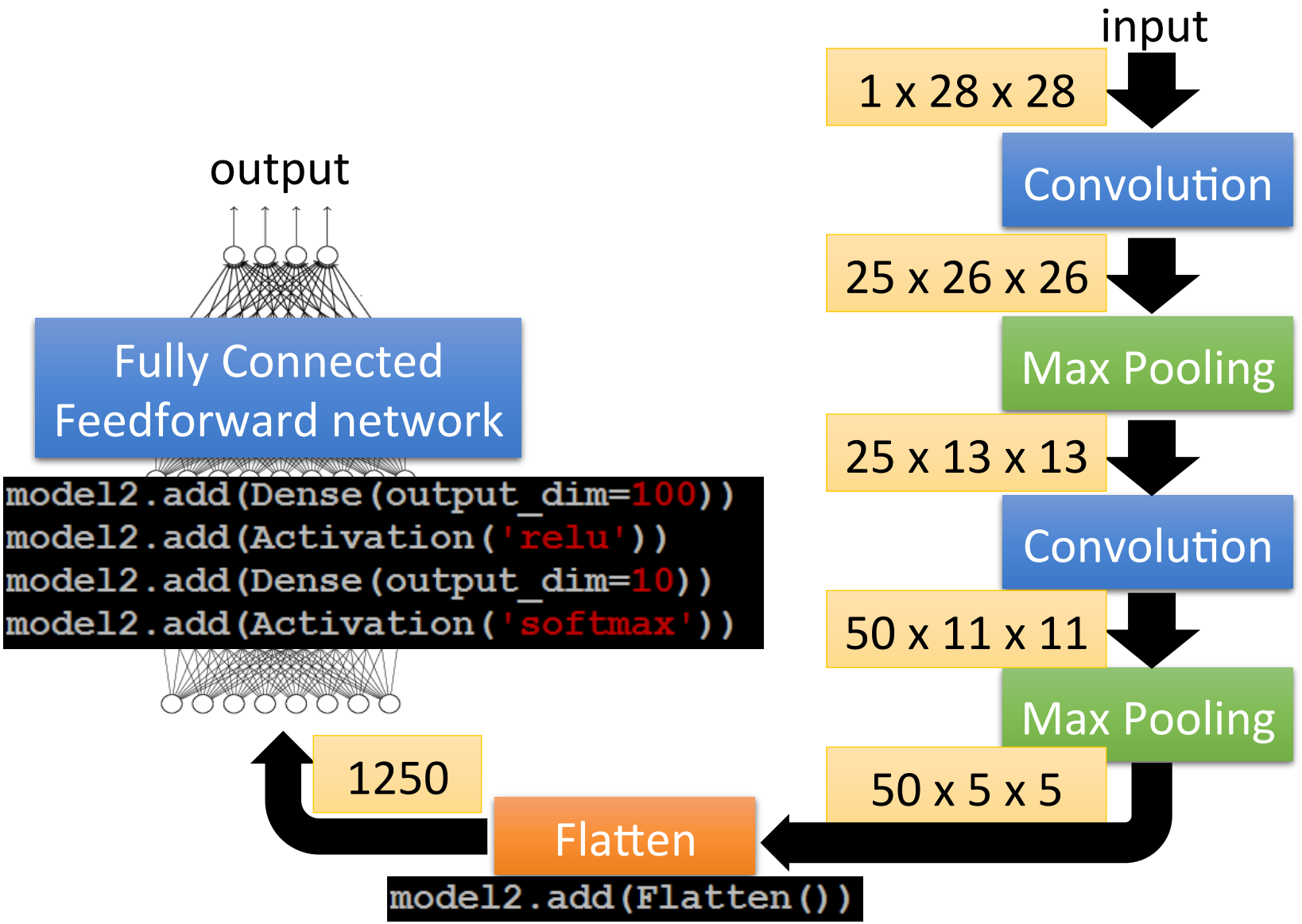
CNN in Keras

Only modified the ***network structure*** and ***input format (vector -> 3-D tensor)***

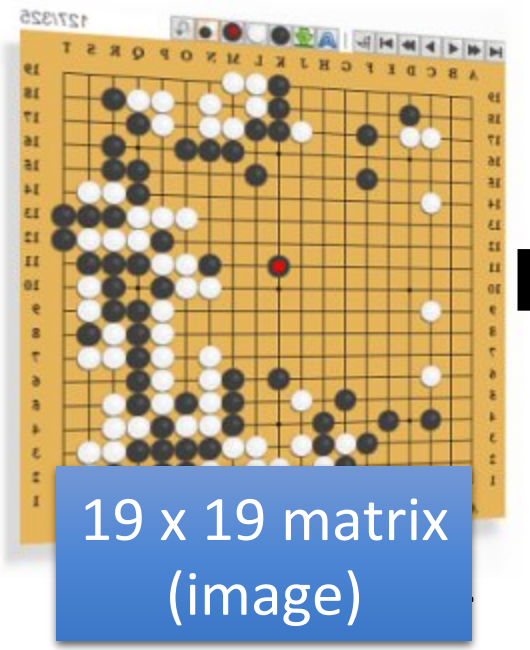


CNN in Keras

Only modified the ***network structure*** and ***input format (vector -> 3-D tensor)***



Application: Playing Go



19 x 19 matrix (image)

Network

Next move (19 x 19 positions)

19 x 19 vector

Black: 1

white: -1

none: 0

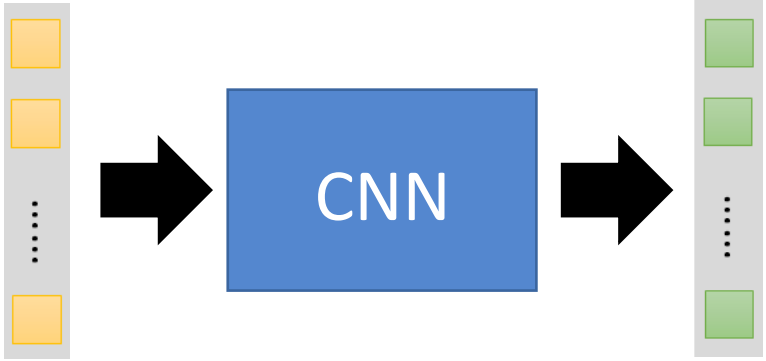
Fully-connected feedforward network can be used

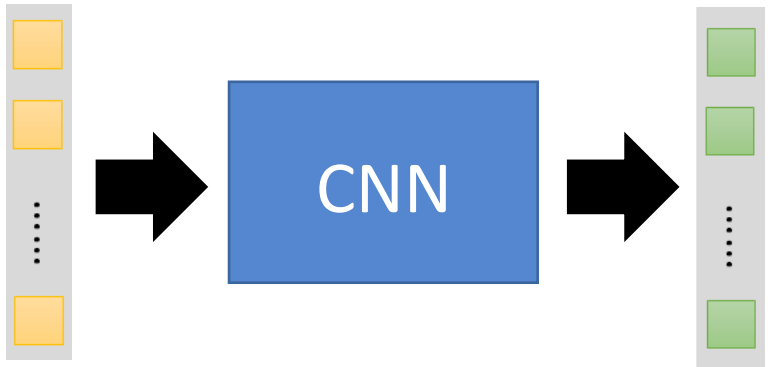
But CNN performs much better.

Application: Playing Go

Training: record of previous plays

black: 5之五 → white: 天元 → black: 五之5 …





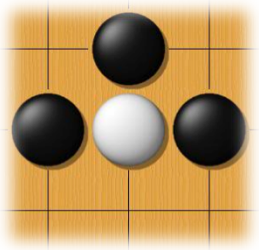
Target:
“天元” = 1
else = 0

Target:
“五之 5” = 1
else = 0

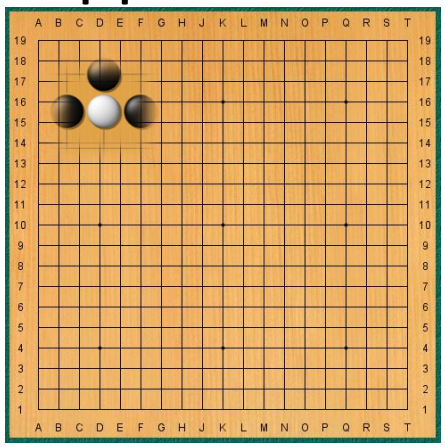
Why CNN for playing Go?

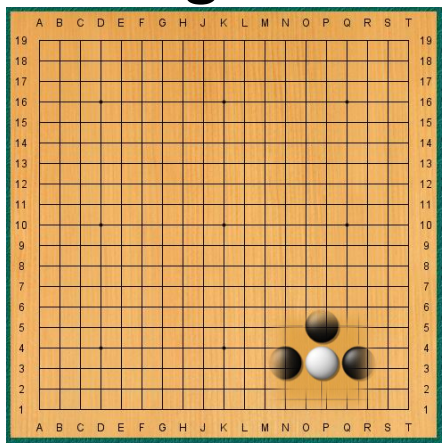
- Some patterns are much smaller than the whole image

Alpha Go uses 5 x 5 for first layer



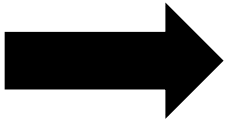
- The same patterns appear in different regions.





Why CNN for playing Go?

- Subsampling the pixels will not change the object



Max Pooling

How to explain this???

Neural network architecture. The input to the policy network is a $19 \times 19 \times 48$ image stack consisting of 48 feature planes. The first hidden layer zero pads the input into a 23×23 image, then convolves k filters of kernel size 5×5 with stride 1 with the input image and applies a rectifier nonlinearity. Each of the subsequent hidden layers 2 to 12 zero pads the respective previous hidden layer into a 21×21 image, then convolves k filters of kernel size 3×3 with stride 1, again followed by a rectifier nonlinearity. The final layer convolves 1 filter of kernel size 1×1 with stride 1

Alpha Go does not use Max Pooling .....

x function. The match version of AlphaGo used $k = 192$ filters; Fig. 2b and Extended Data Table 3 additionally show the results of training with $k = 128$, 256 and 384 filters.

Variants of Neural Networks

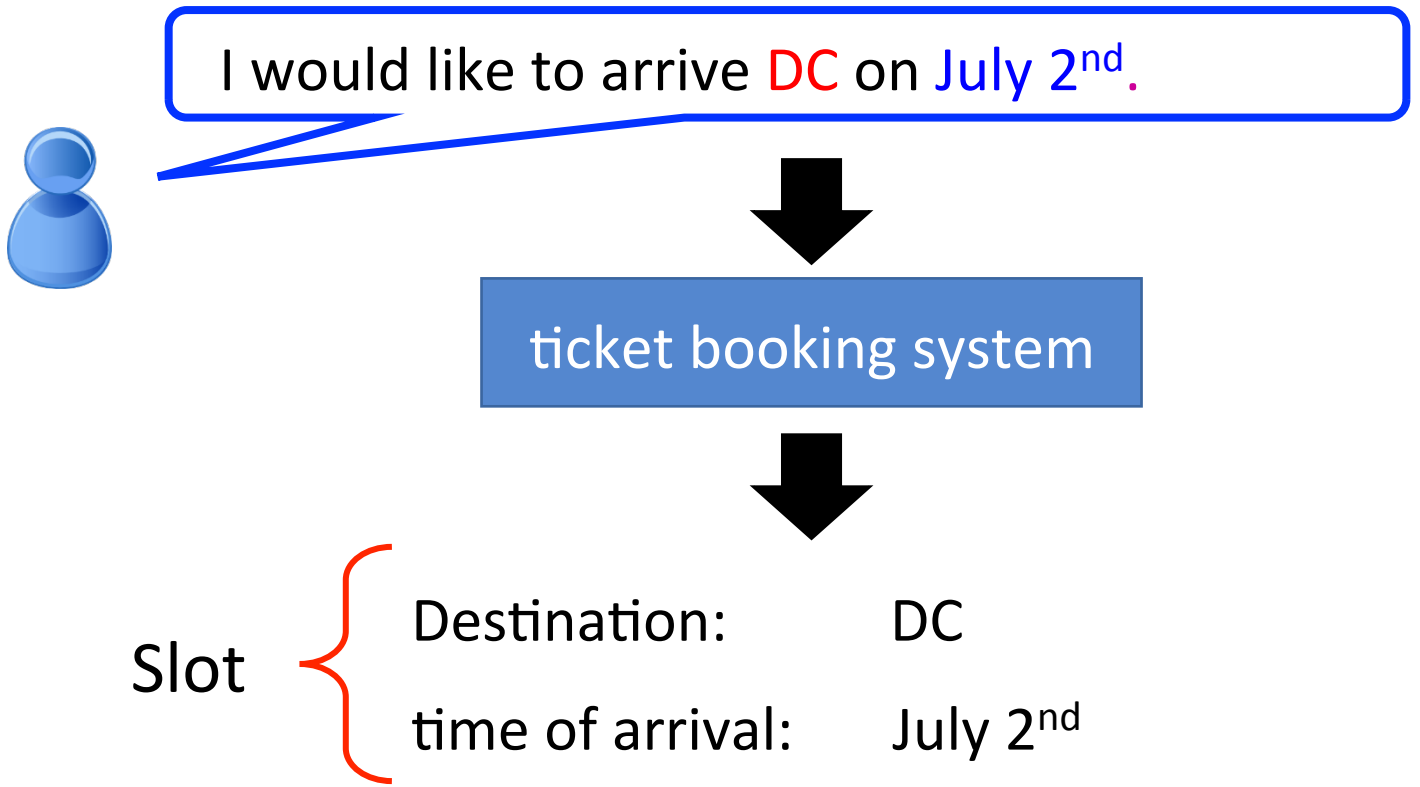
Convolutional Neural Network (CNN)

Recurrent Neural Network (RNN)

Neural Network with Memory

Example Application

- Slot Filling

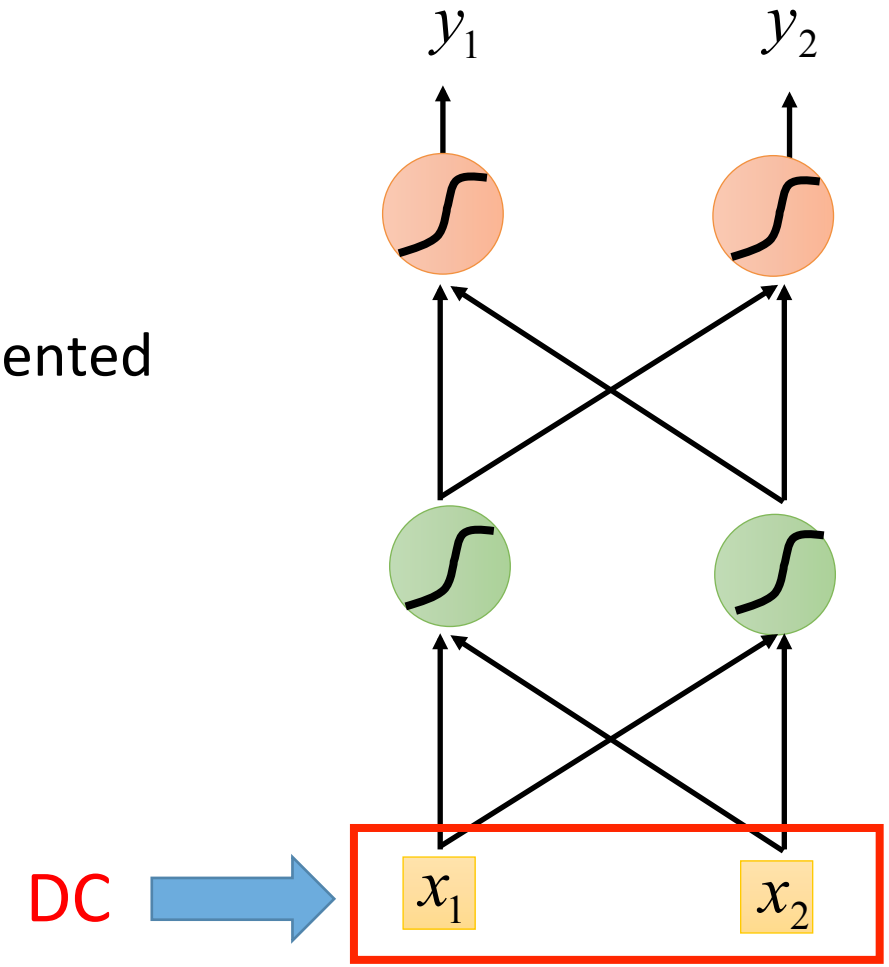


Example Application

Solving slot filling by Feedforward network?

Input: a word

(Each word is represented as a vector)



1-of-N encoding

How to represent each word as a vector?

1-of-N Encoding lexicon = {apple, bag, cat, dog, elephant}

The vector is lexicon size.

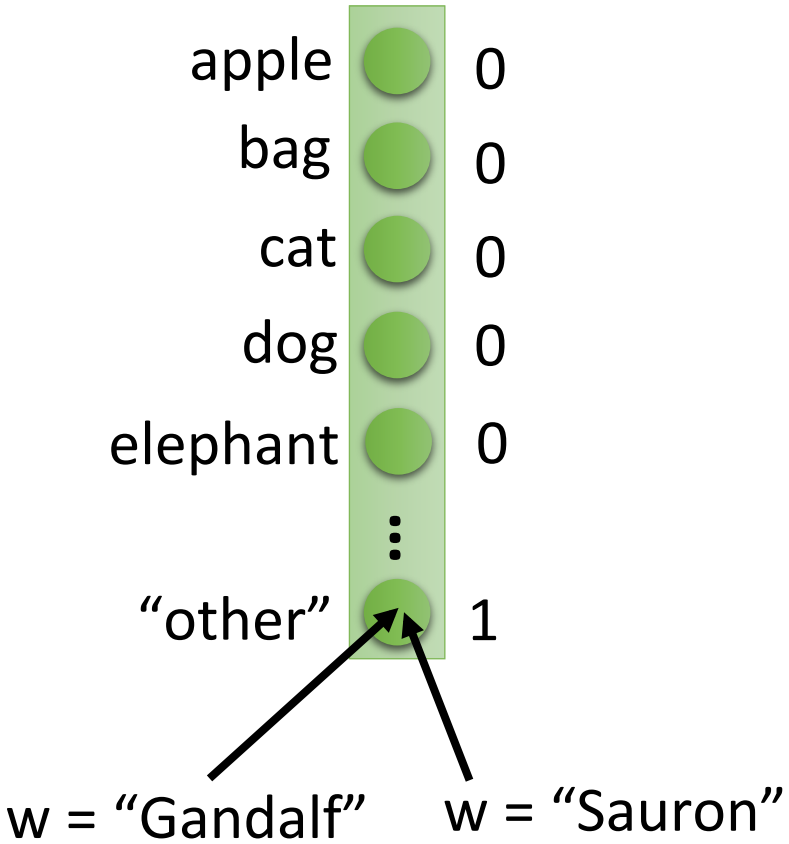
Each dimension corresponds to a word in the lexicon

The dimension for the word is 1, and others are 0

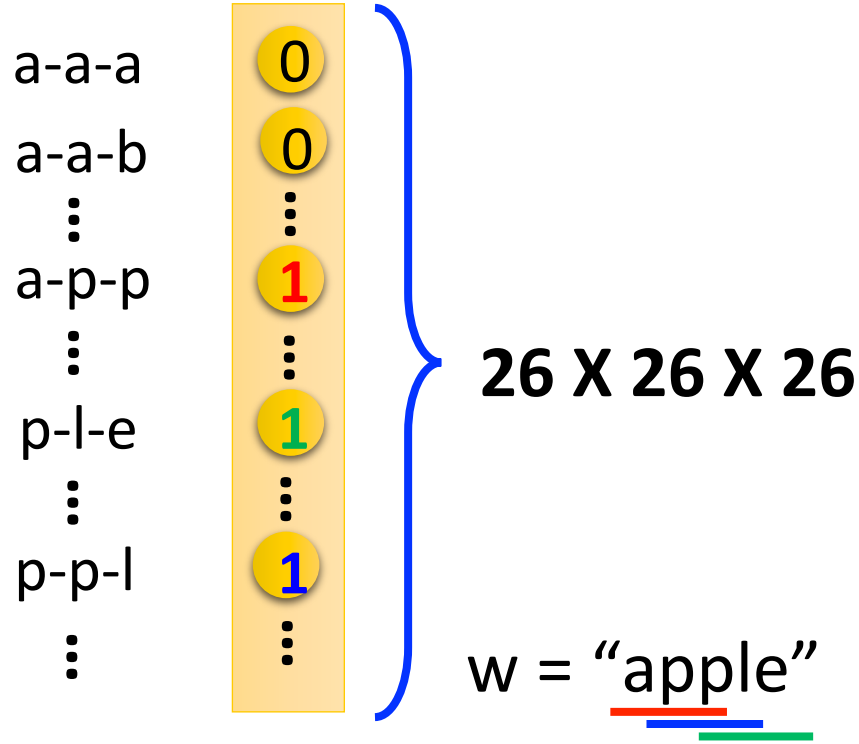
apple = [1 0 0 0 0]
bag = [0 1 0 0 0]
cat = [0 0 1 0 0]
dog = [0 0 0 1 0]
elephant = [0 0 0 0 1]

Beyond 1-of-N encoding

Dimension for "Other"



Word hashing



Example Application

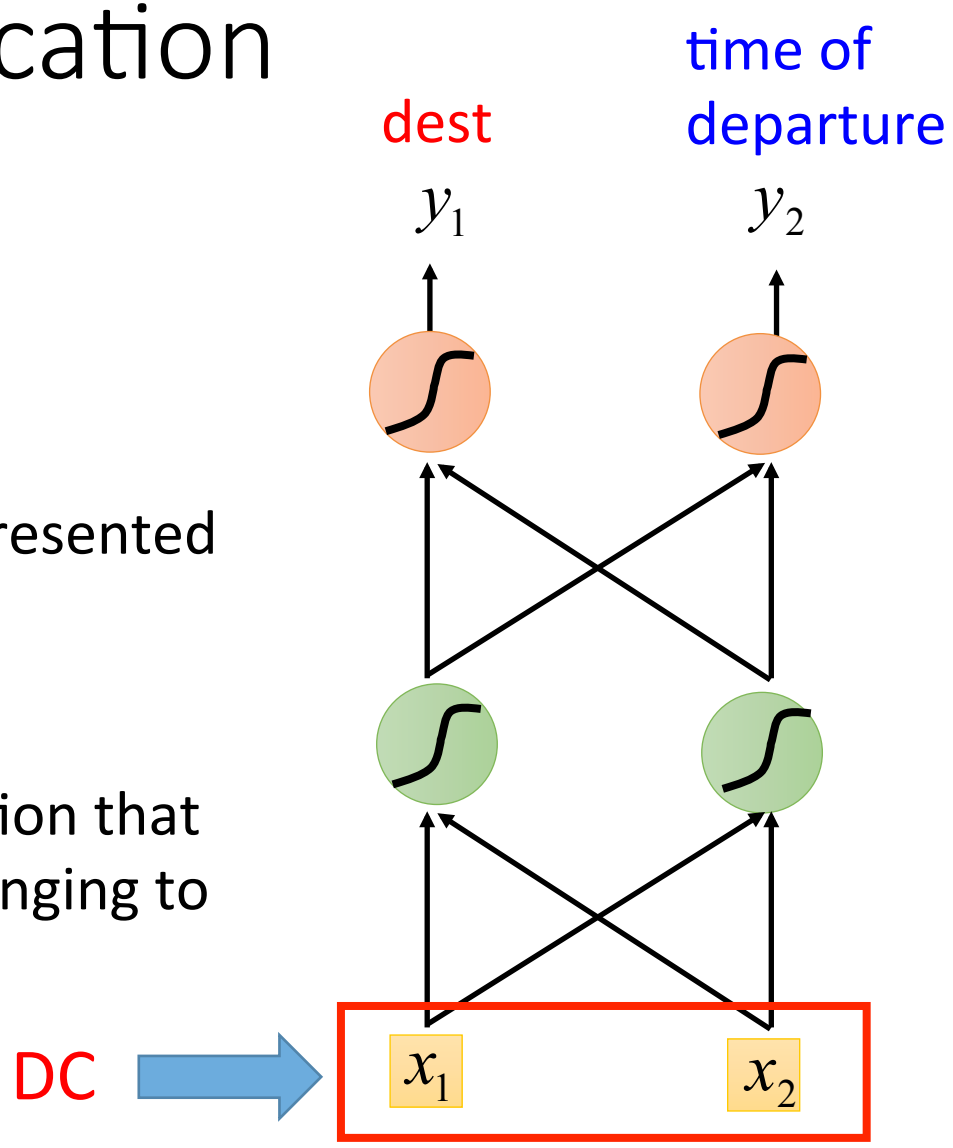
Solving slot filling by Feedforward network?

Input: a word

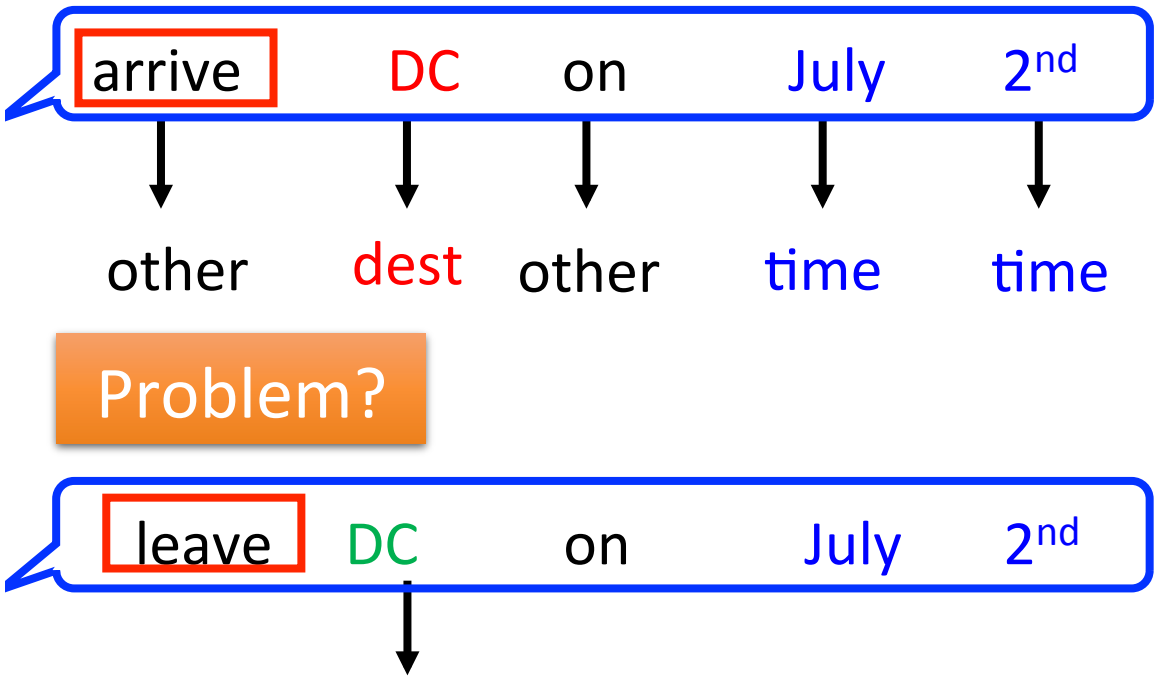
(Each word is represented as a vector)

Output:

Probability distribution that the input word belonging to the slots

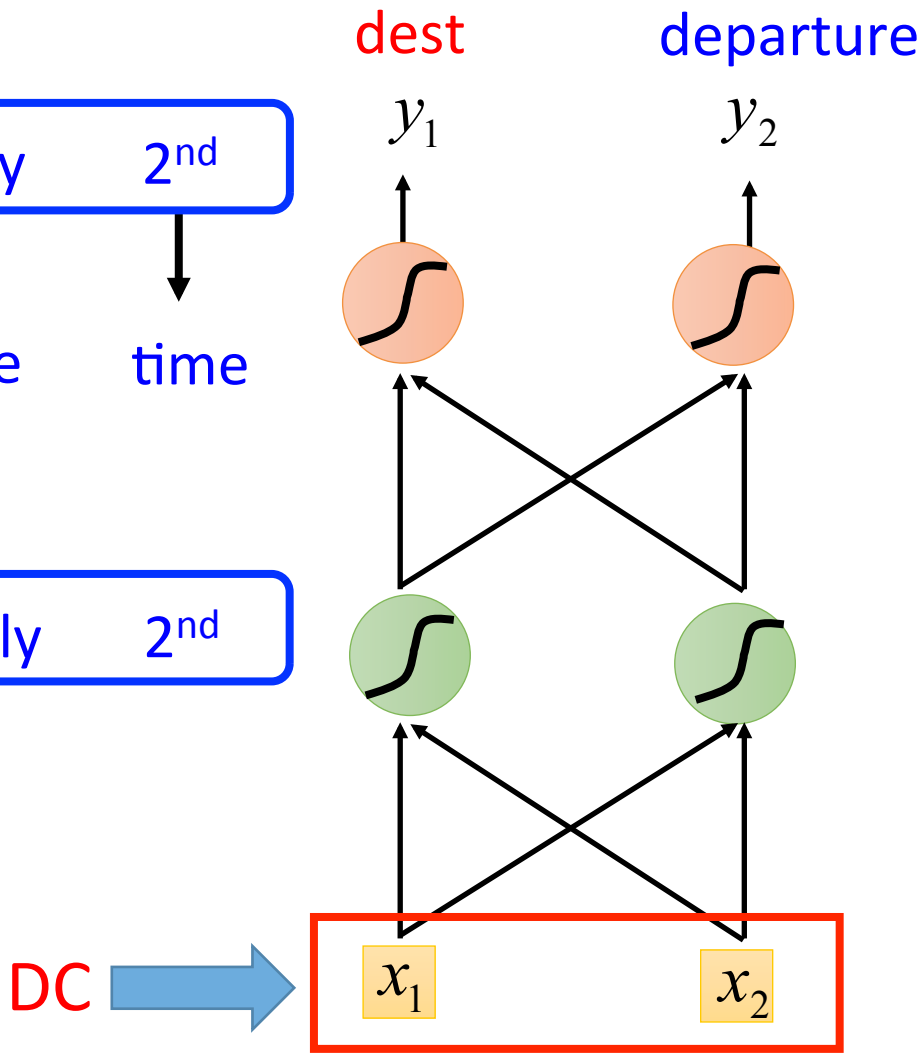


Example Application



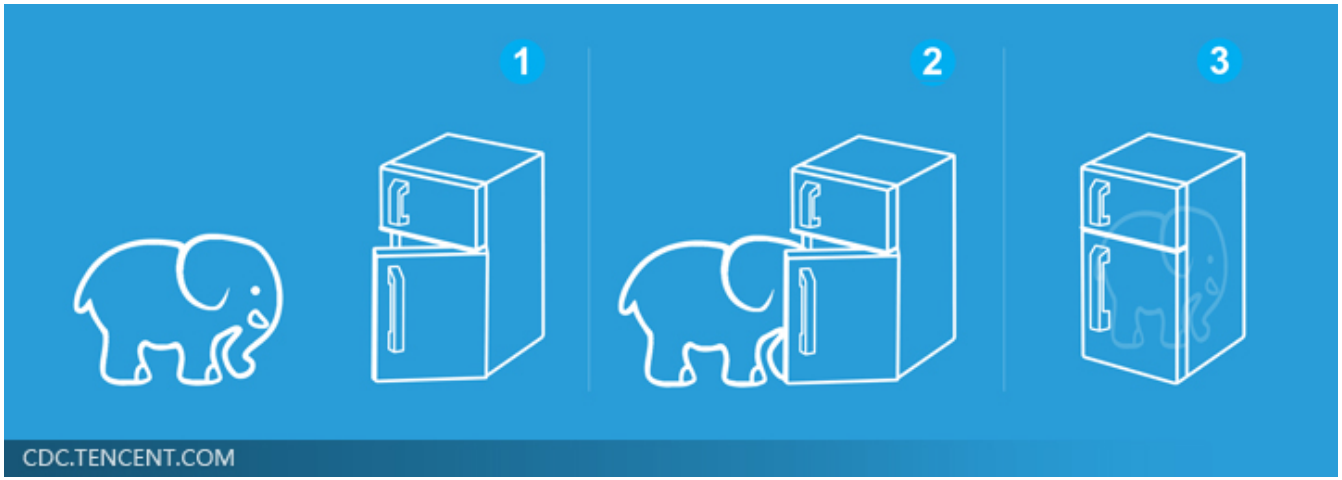
Problem?

Neural network needs memory!



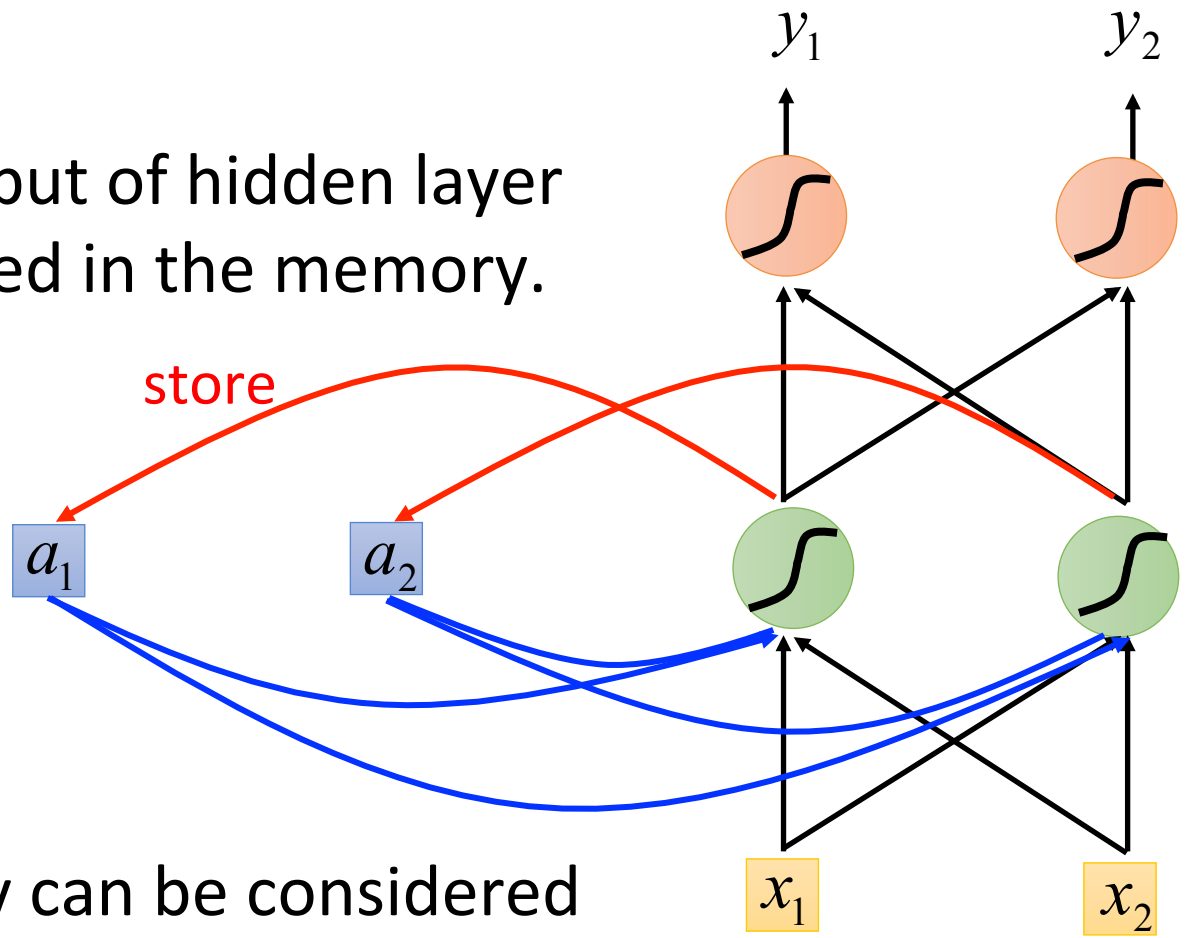
Three Steps for Deep Learning

Deep Learning is so simple ……



Recurrent Neural Network (RNN)

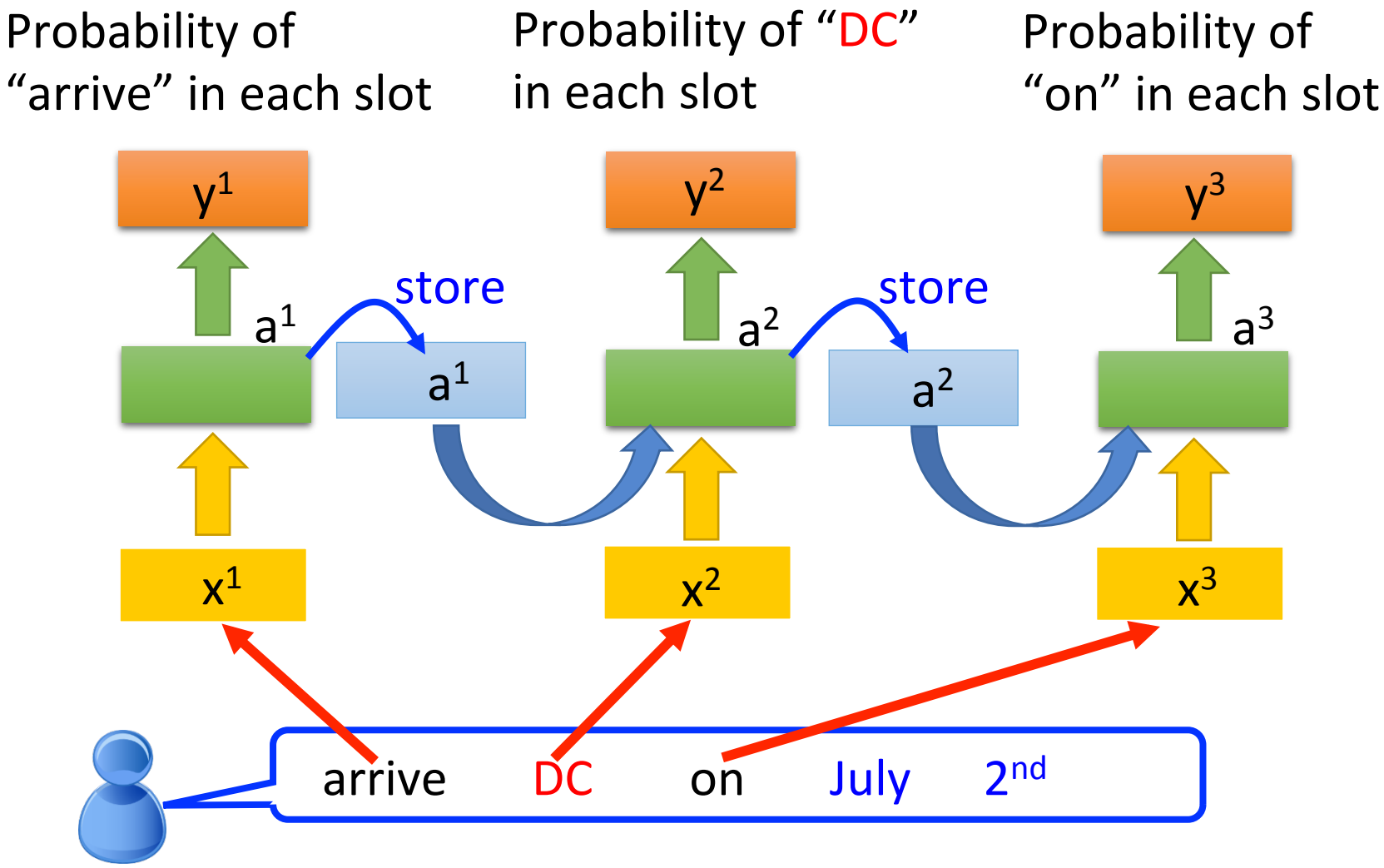
The output of hidden layer are stored in the memory.



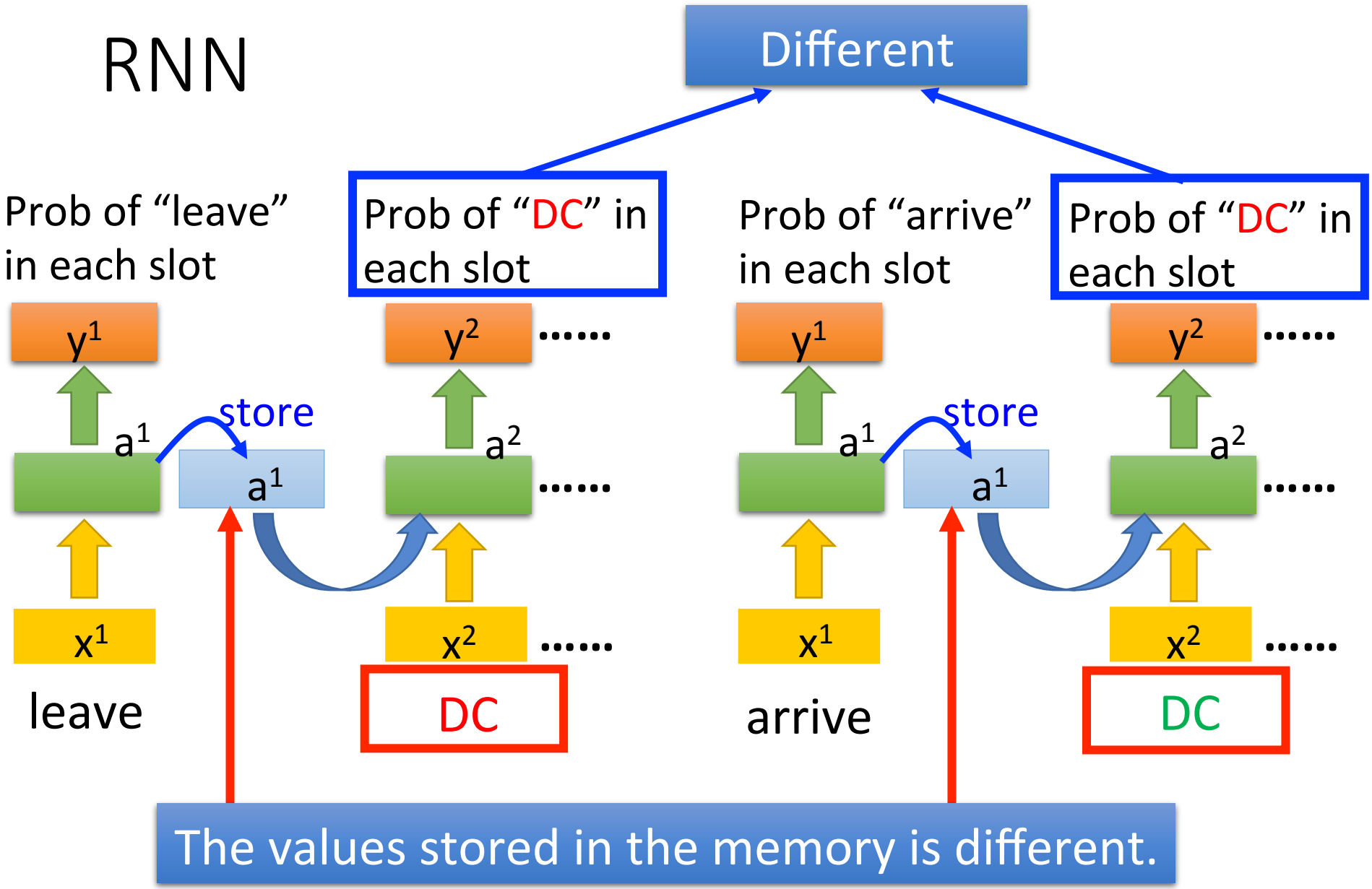
Memory can be considered as another input.

RNN

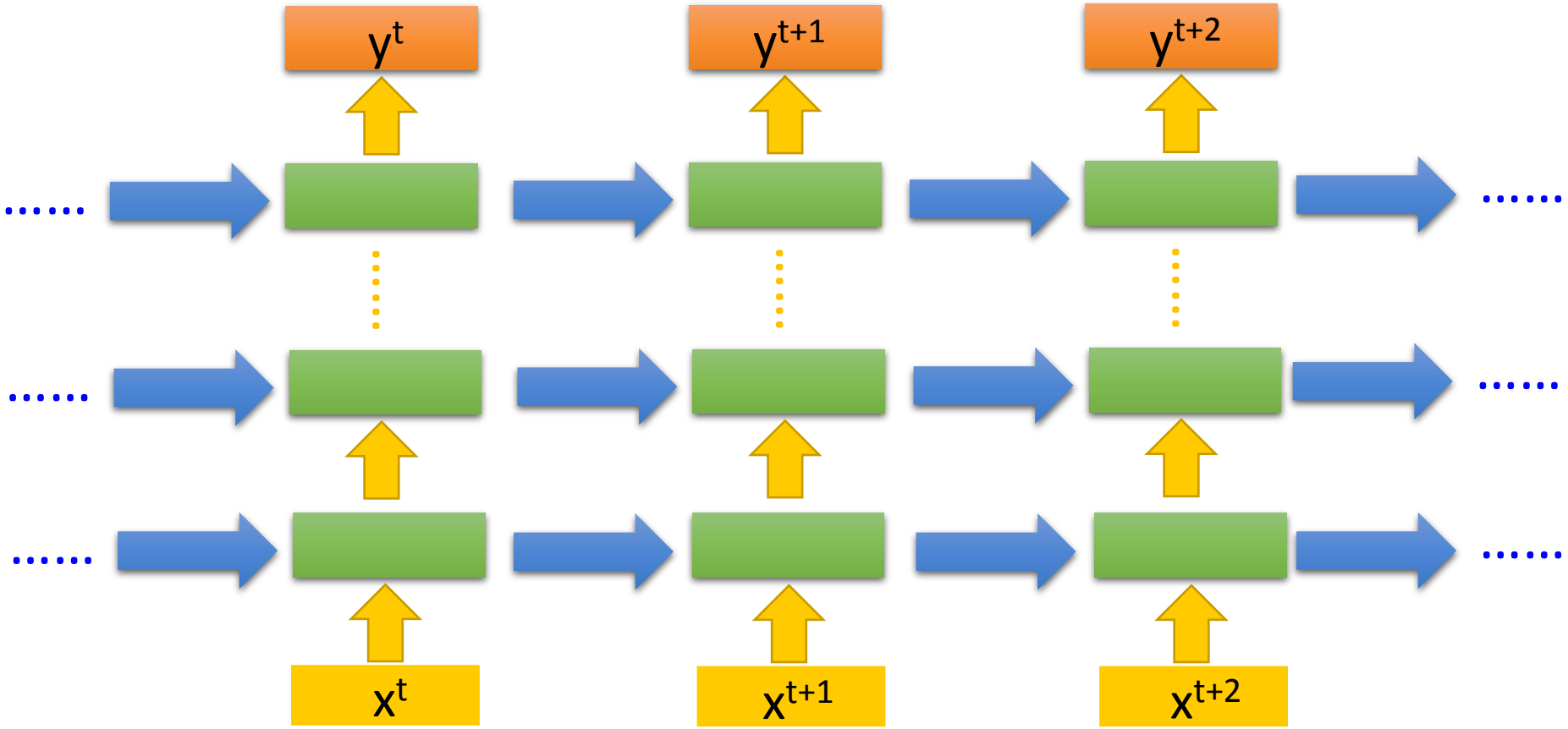
The same network is used again and again.



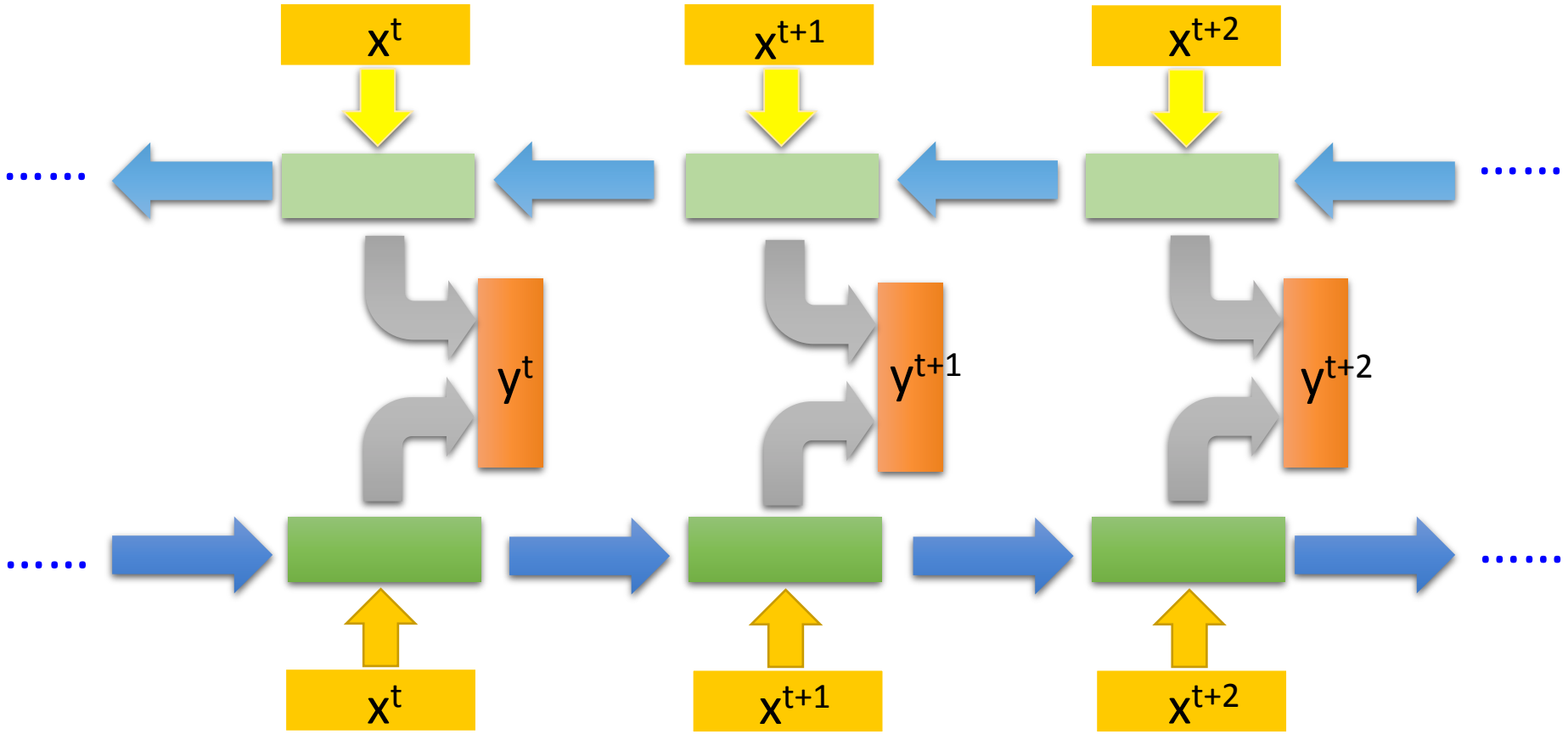
RNN


Different
Prob of "leave" in each slot
Prob of "DC" in each slot
Prob of "arrive" in each slot
Prob of "DC" in each slot
y^1
y^2
.....
a^1
store
a^2
x^1
x^2
leave
DC
arrive
DC
The values stored in the memory is different.

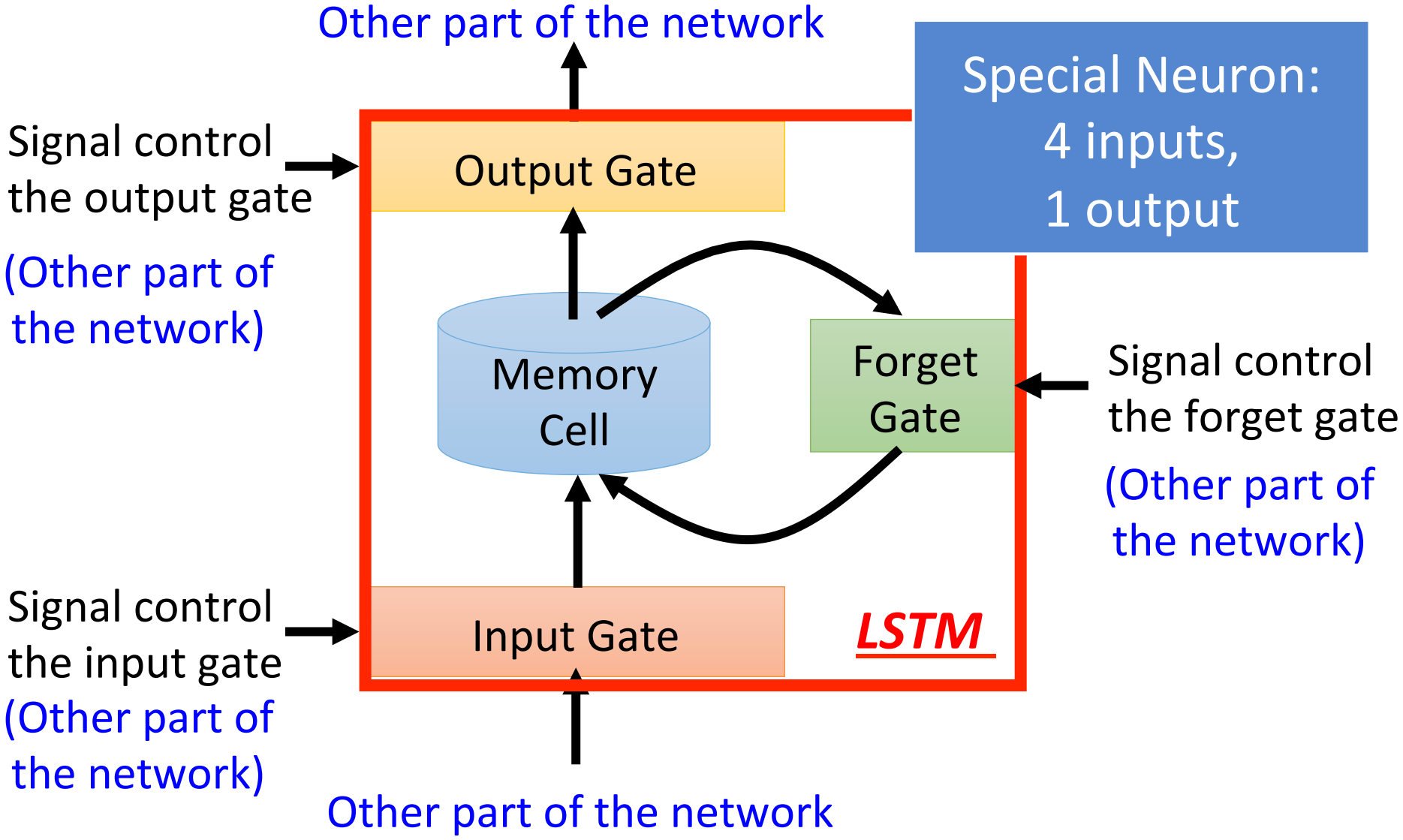
Of course it can be deep …

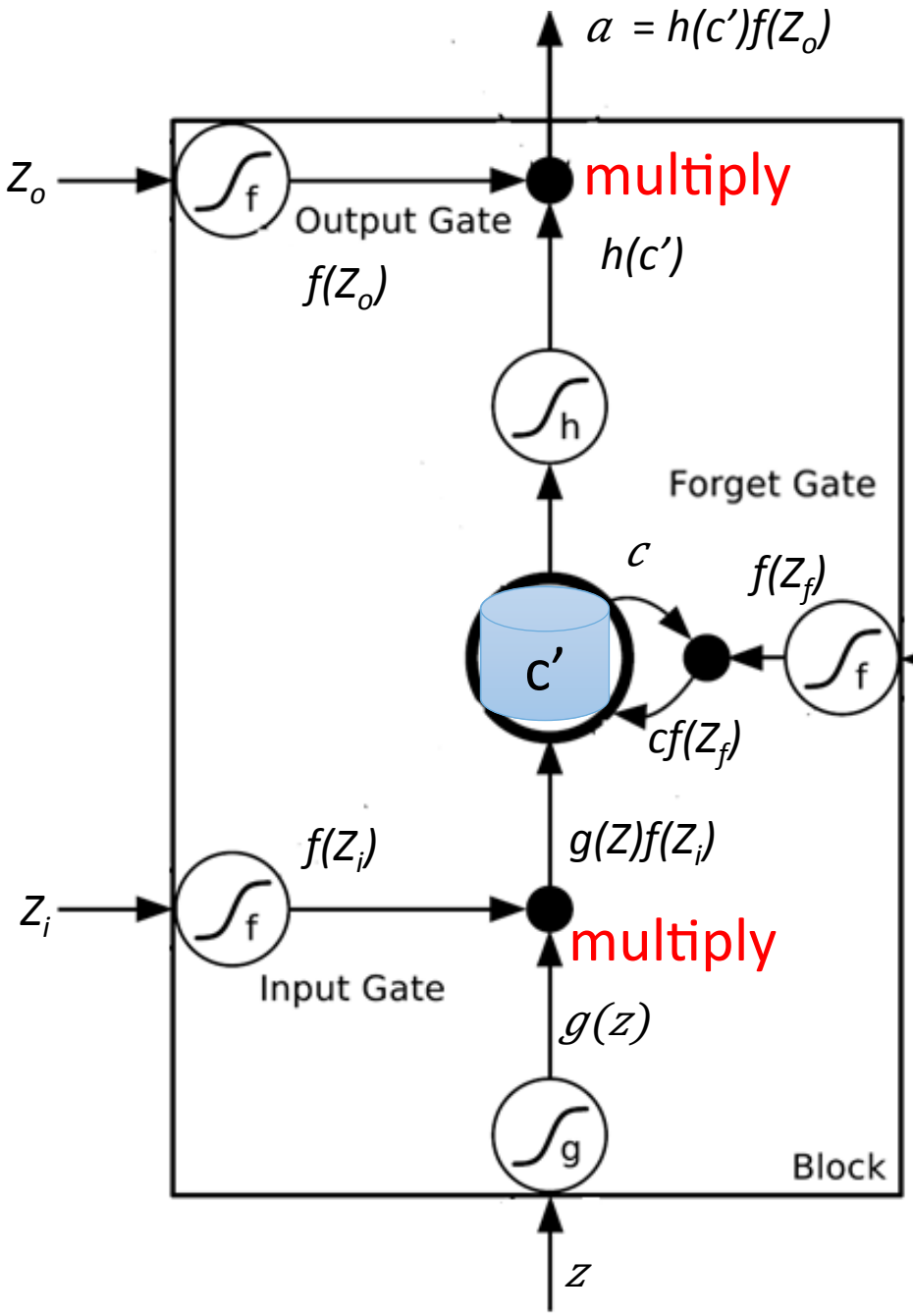


Bidirectional RNN



Long Short-term Memory (LSTM)



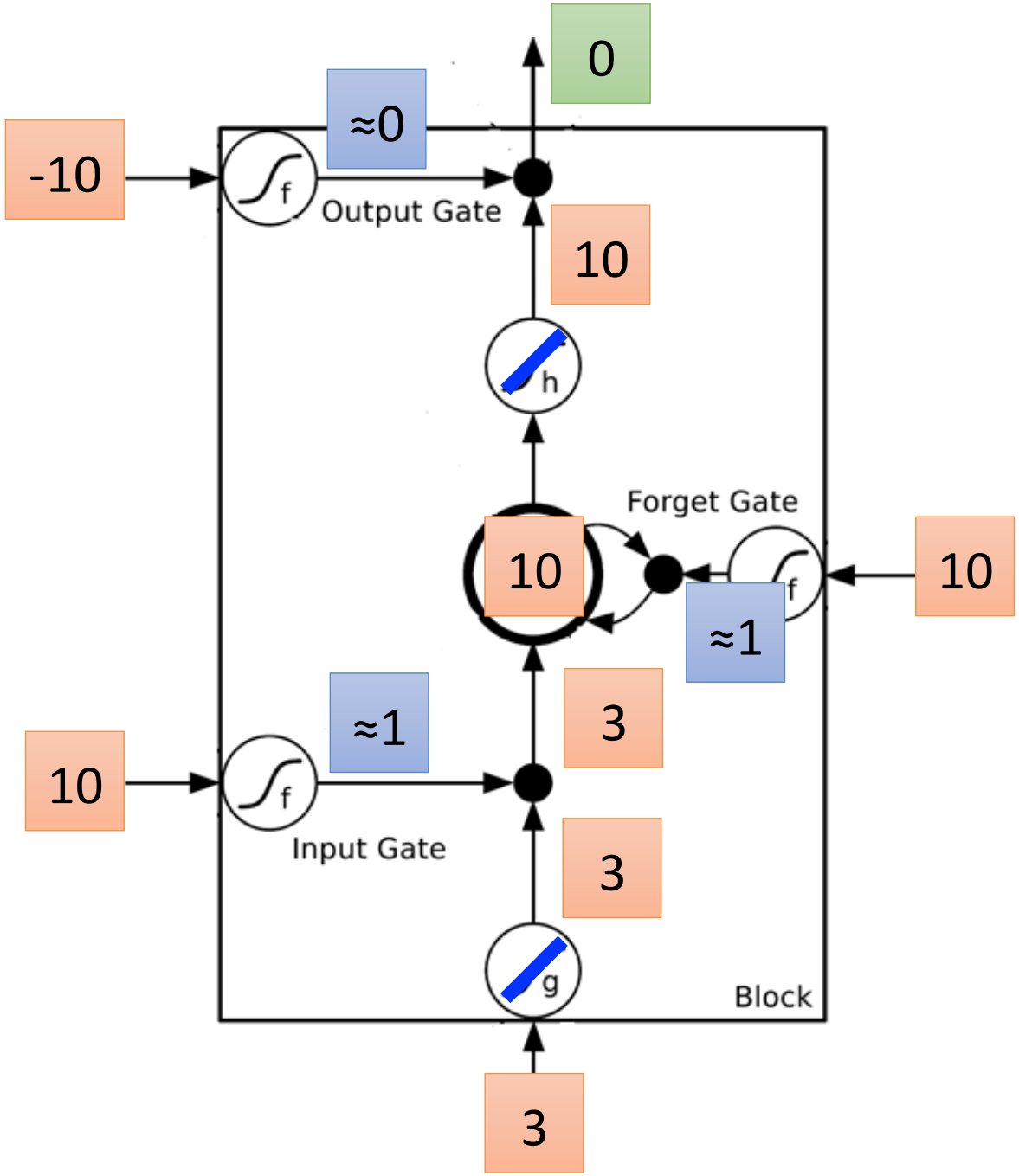


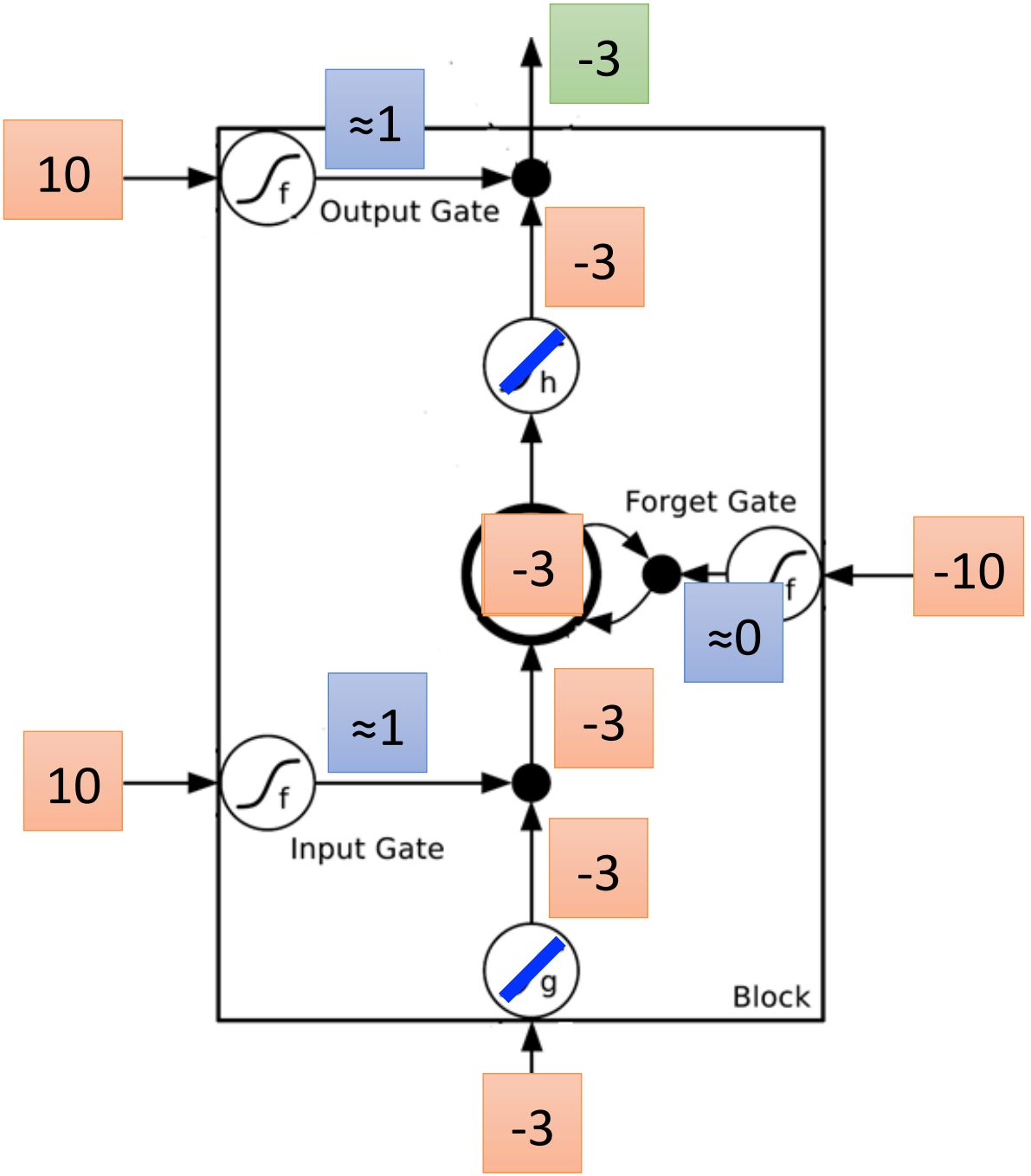
Activation function f is usually a sigmoid function

Between 0 and 1

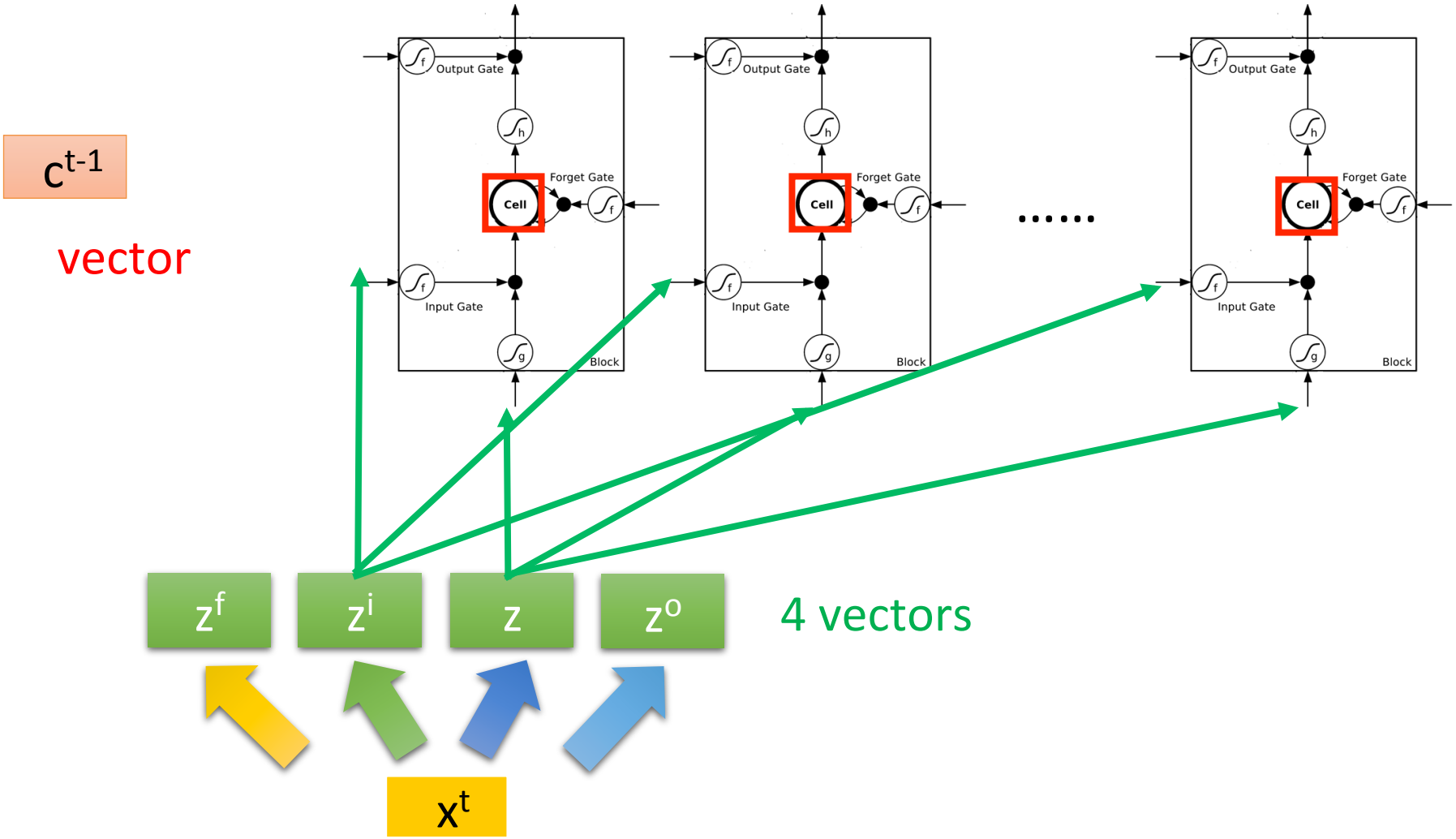
Mimic open and close gate

$c' = g(z)f(Z_i) + cf(Z_f)$

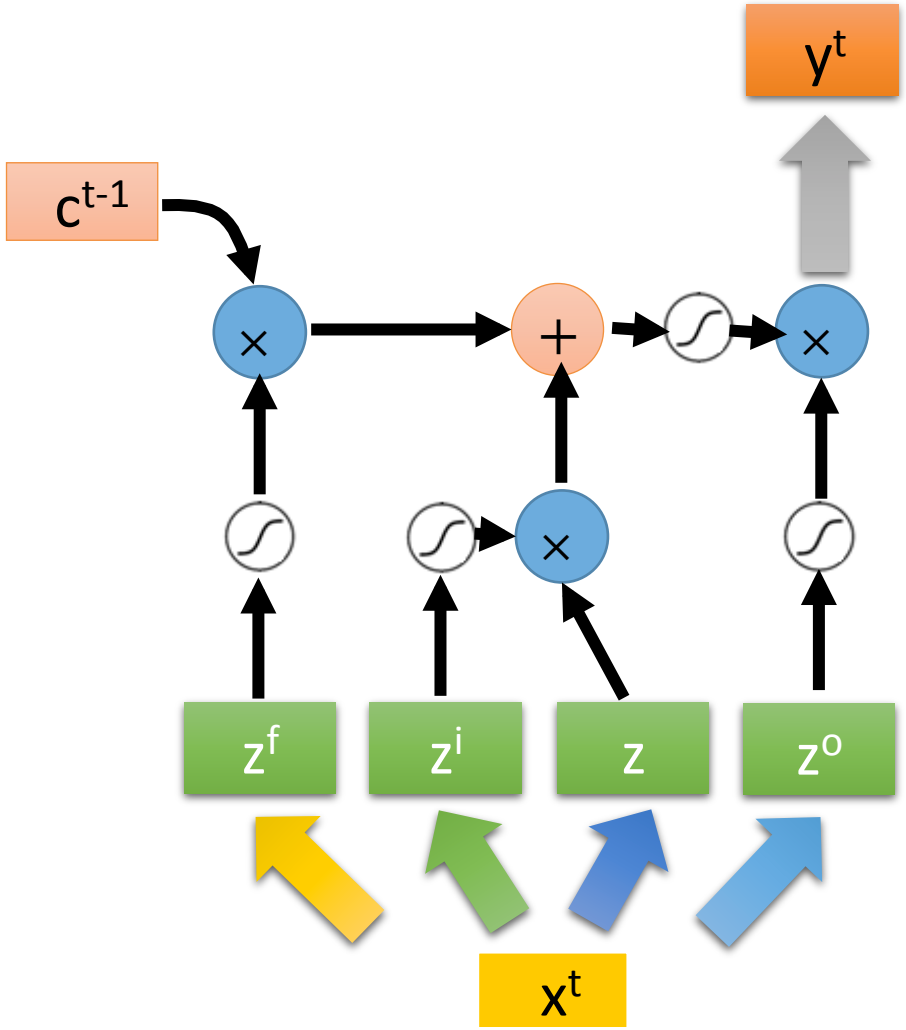

0
≈0
-10
Output Gate
10
h
Forget Gate
10
10
≈1
10
≈1
3
10
Input Gate
3
g
Block
3

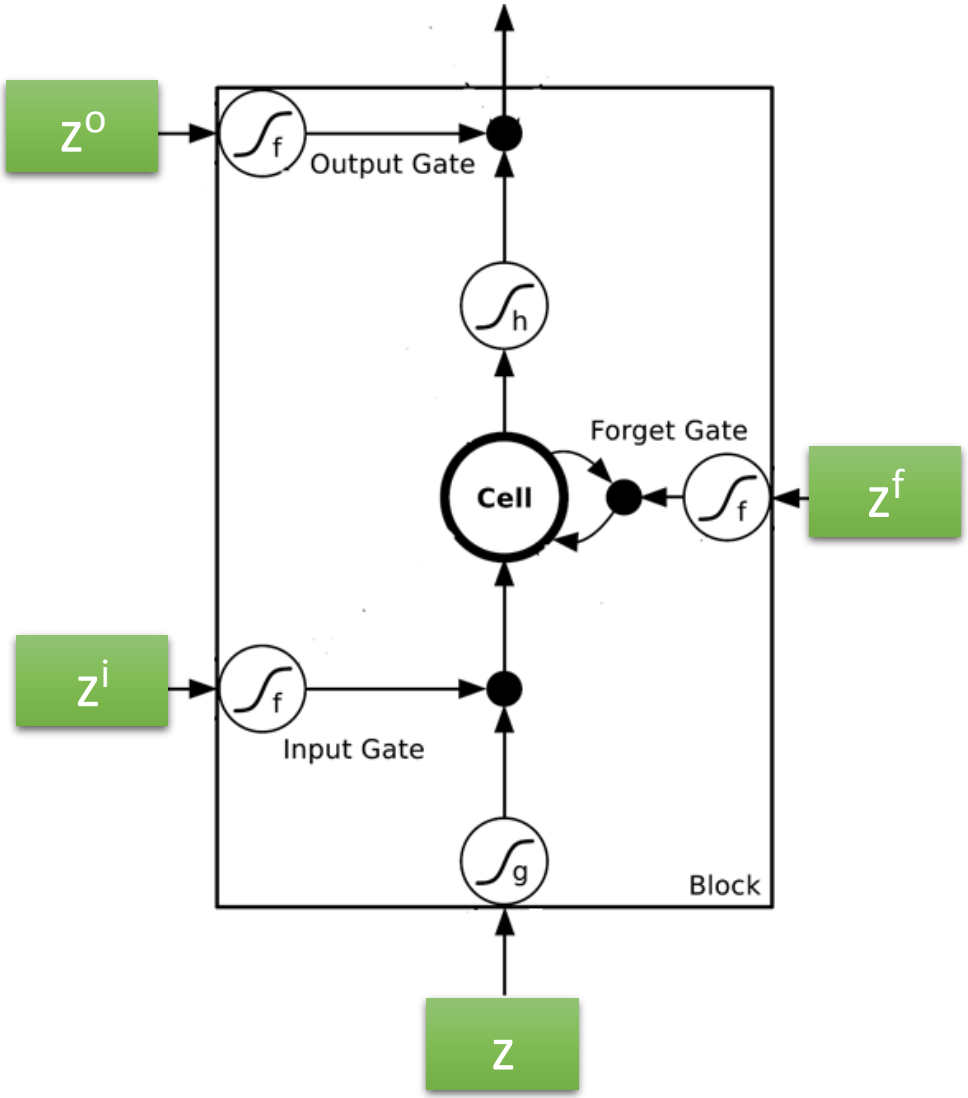

-3
≈1
10
Output Gate
-3
h
Forget Gate
-3
-10
≈0
≈1
-3
10
Input Gate
-3
g
Block
-3

LSTM


c^{t-1}
vector
Output Gate
Forget Gate
Cell
Input Gate
Block
.....
z^f
z^i
z
z^o
4 vectors
x^t

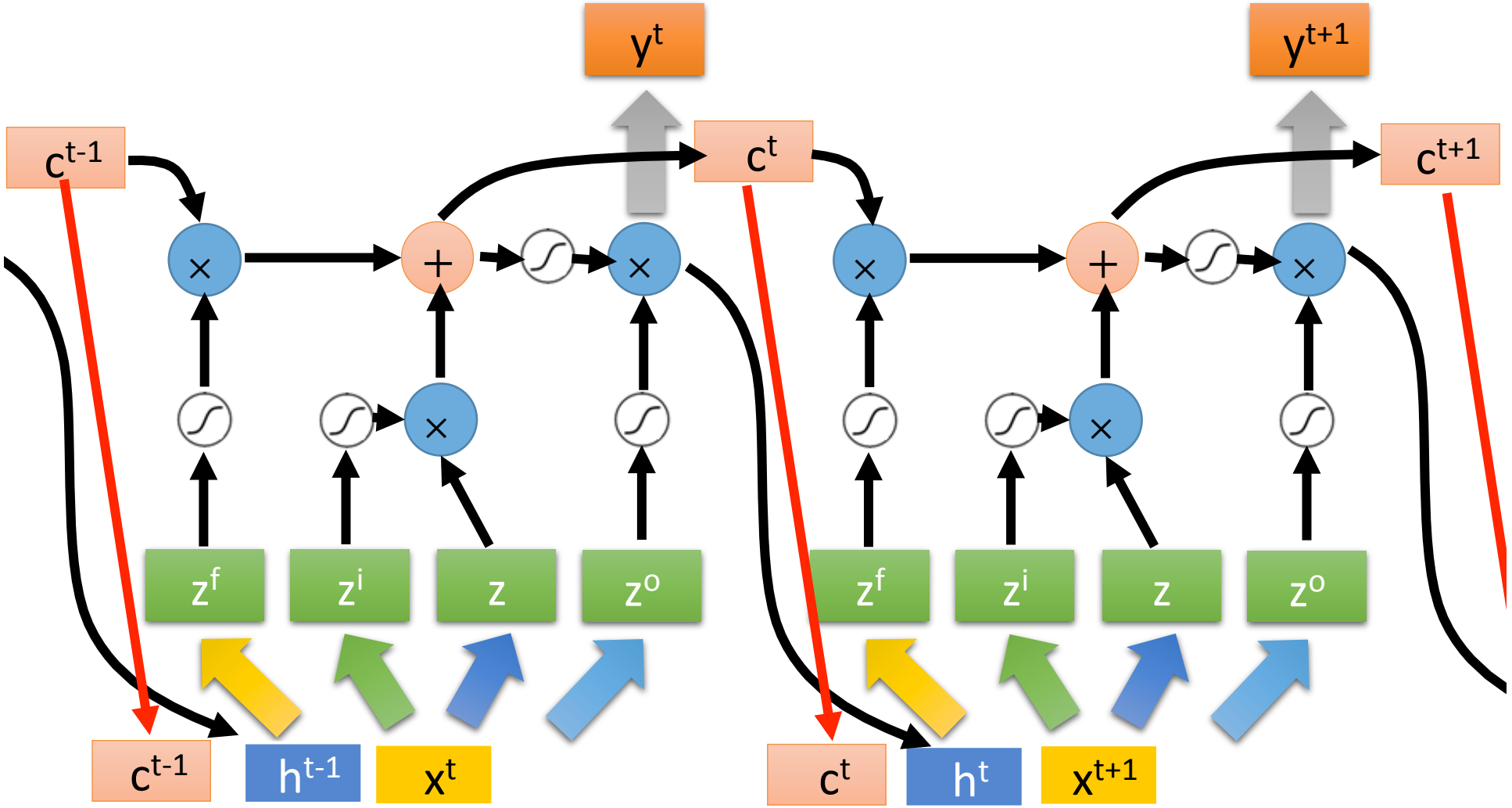
LSTM


y^t
c^{t-1}
z^f
z^i
z
z^o
x^t

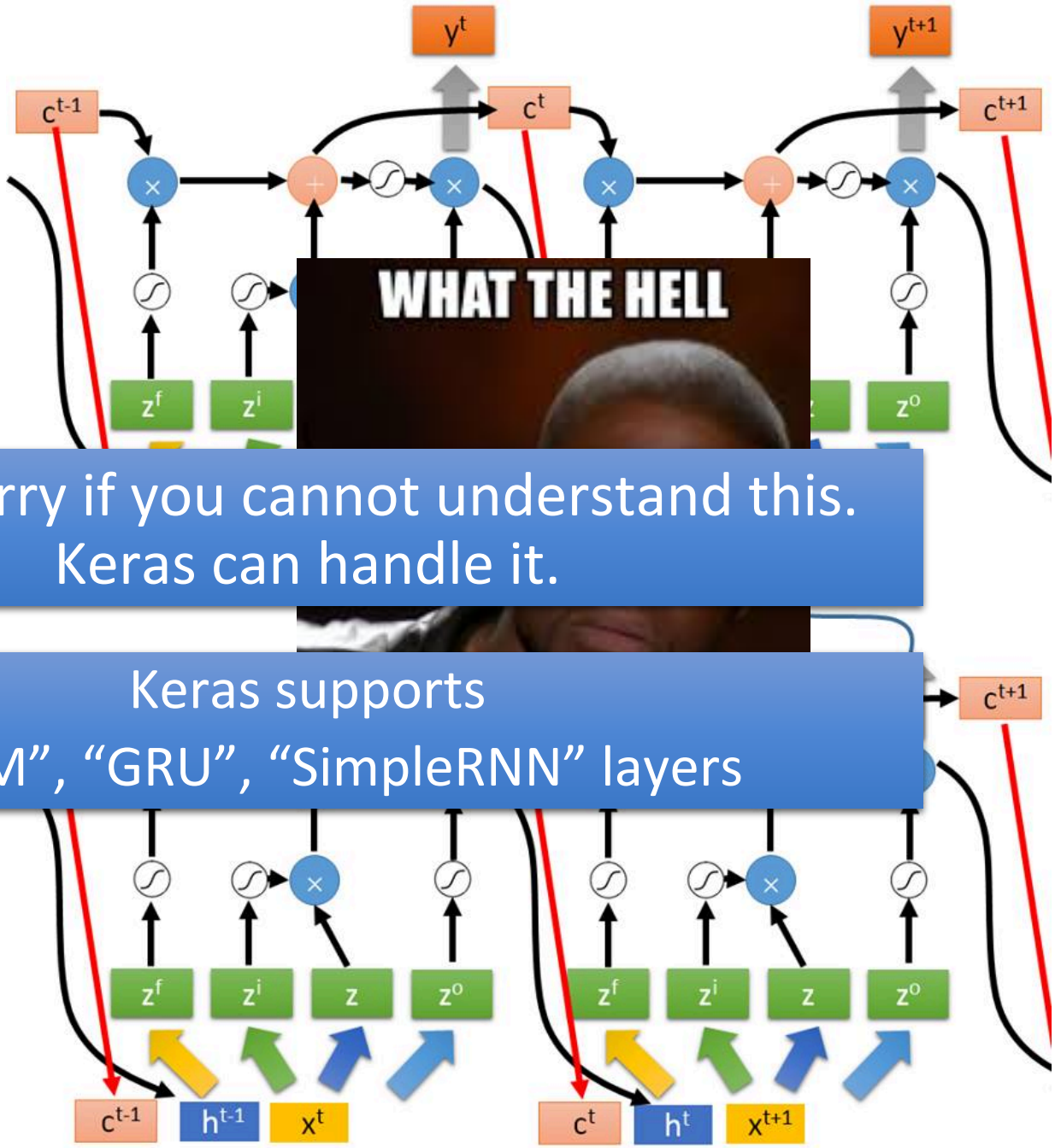

z^o
Output Gate
h
Forget Gate
Cell
z^f
z^i
Input Gate
g
Block
z

LSTM

Extension: “peephole”



Multiple-layer LSTM



Don't worry if you cannot understand this.
Keras can handle it.

Keras supports
"LSTM", "GRU", "SimpleRNN" layers

This is quite standard now.

https://img.komicolle.org/2015-09-20/src/14426967627131.gif

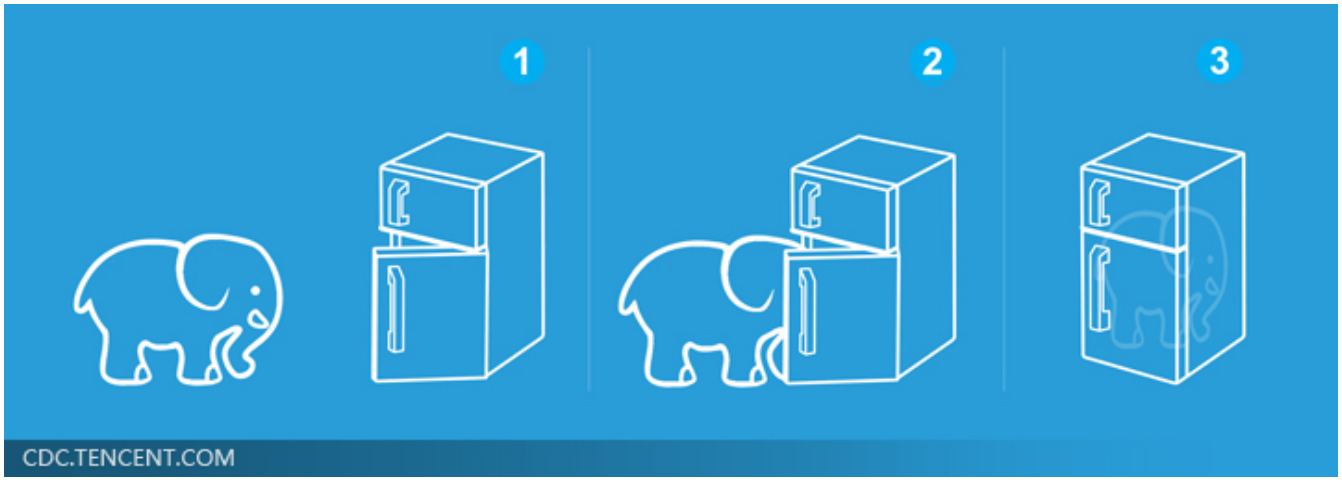
Three Steps for Deep Learning

Step 1: define a set of function

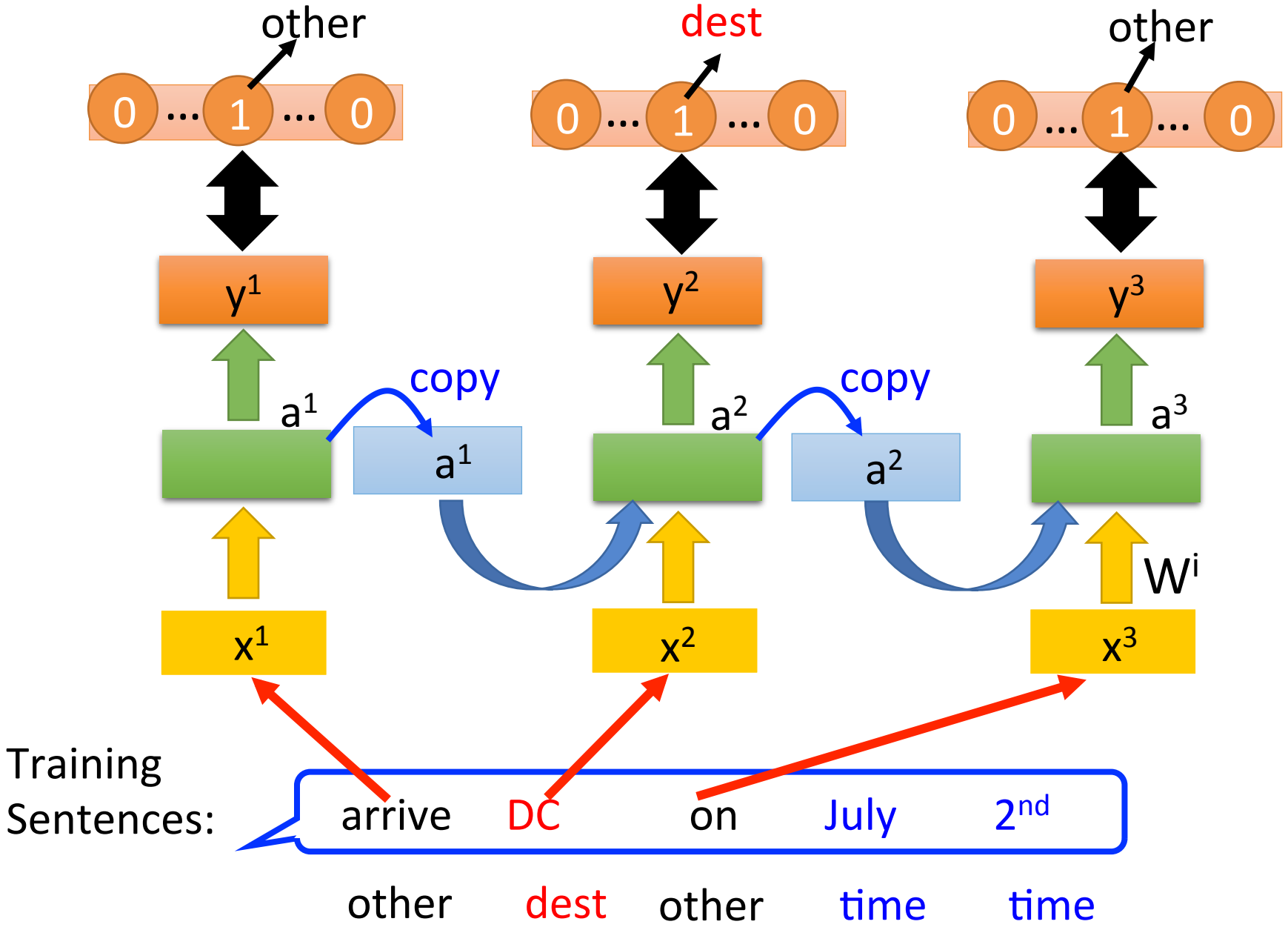
Step 2: goodness of function

Step 3: pick the best function

Deep Learning is so simple ……



Learning Target



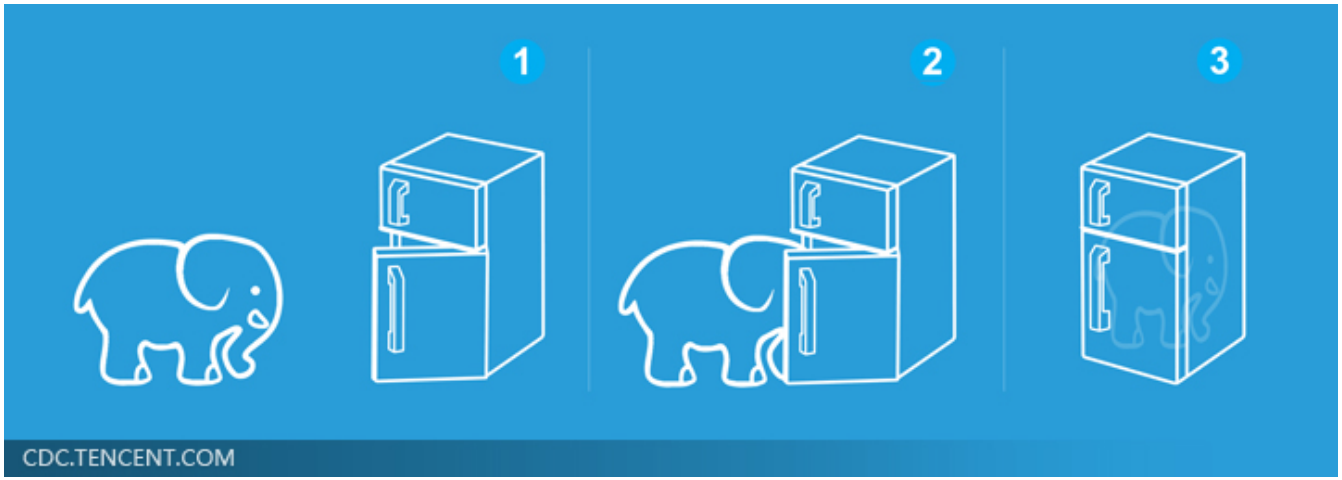
Three Steps for Deep Learning

Step 1: define a set of function

Step 2: goodness of function

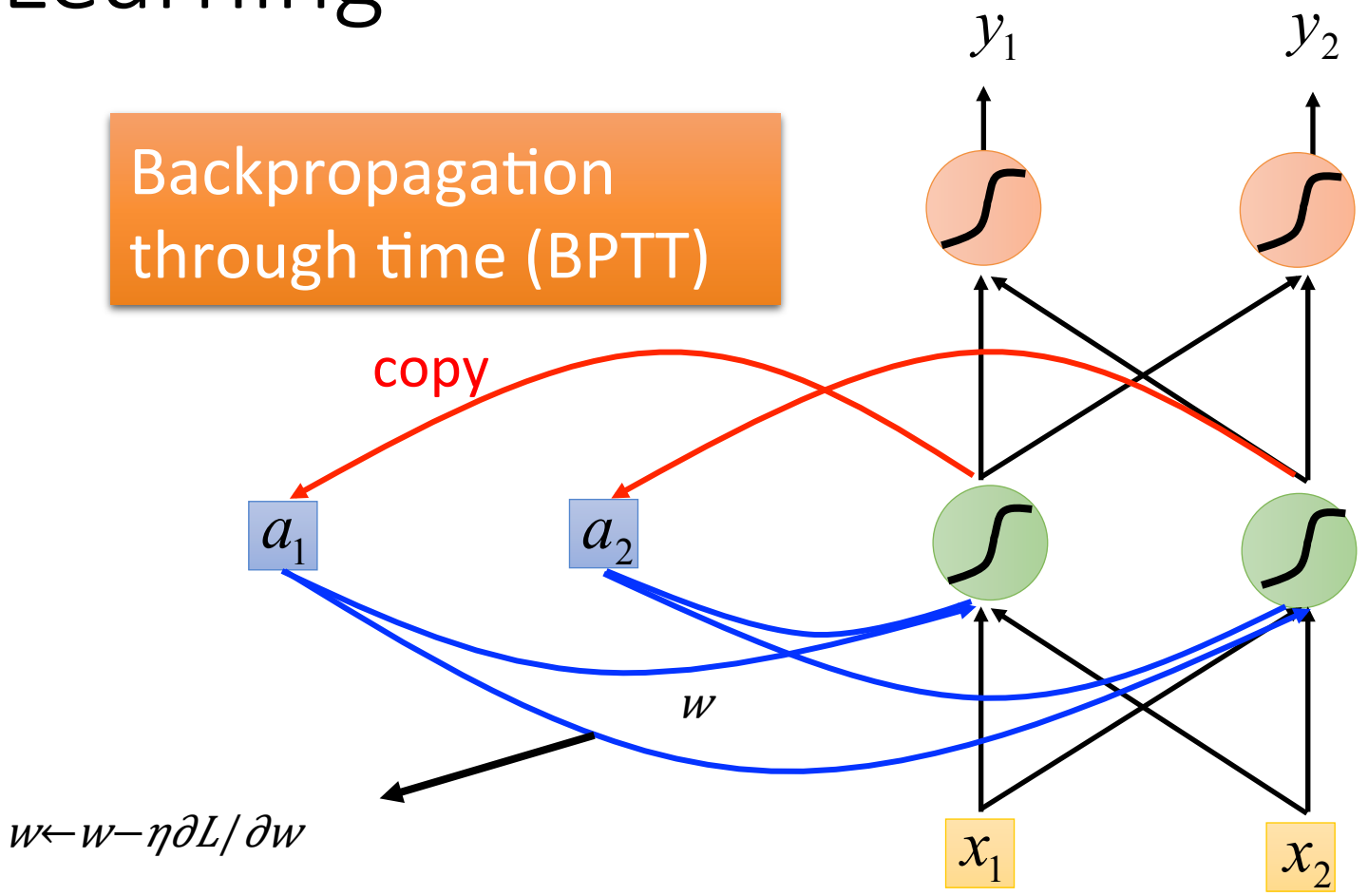
Step 3: pick the best function

Deep Learning is so simple ……



Learning

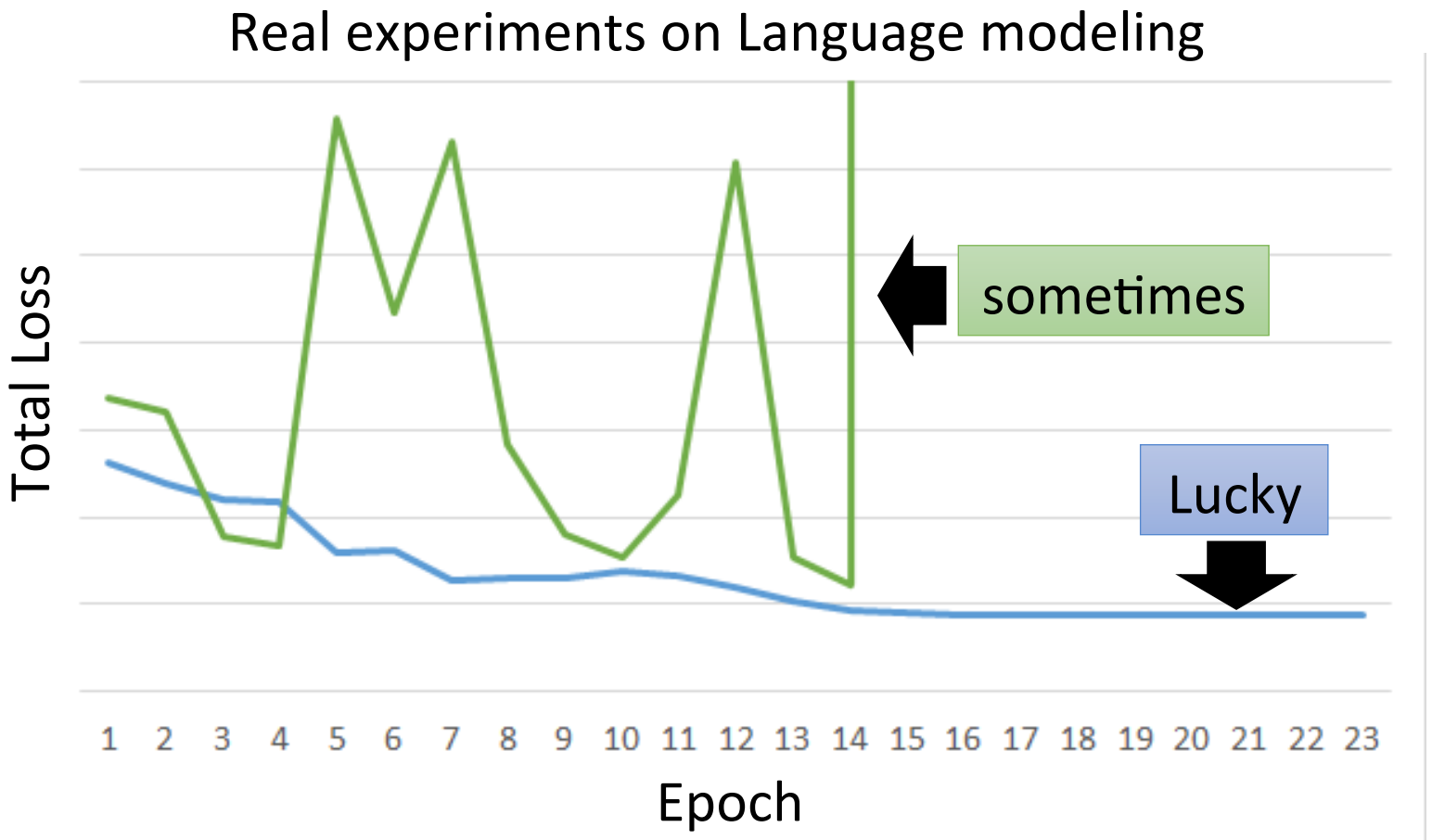
Backpropagation through time (BPTT)



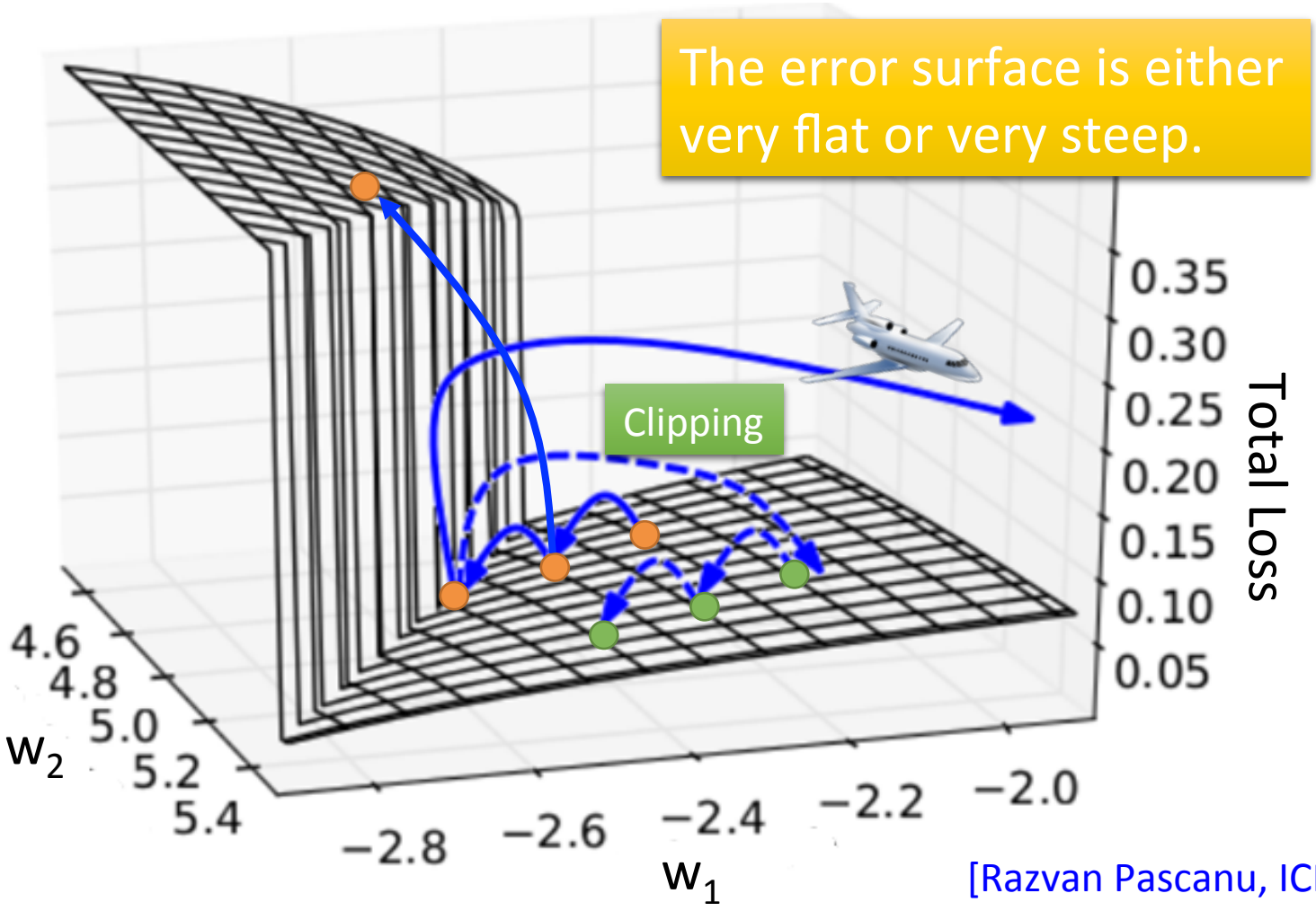
RNN Learning is very difficult in practice.

Unfortunately ……

- RNN-based network is not always easy to learn



The error surface is rough.



[Razvan Pascanu, ICML'13]

Why?

$w=1$ → $y^{1000}=1$

$w=1.01$ → $y^{1000}\approx 20000$

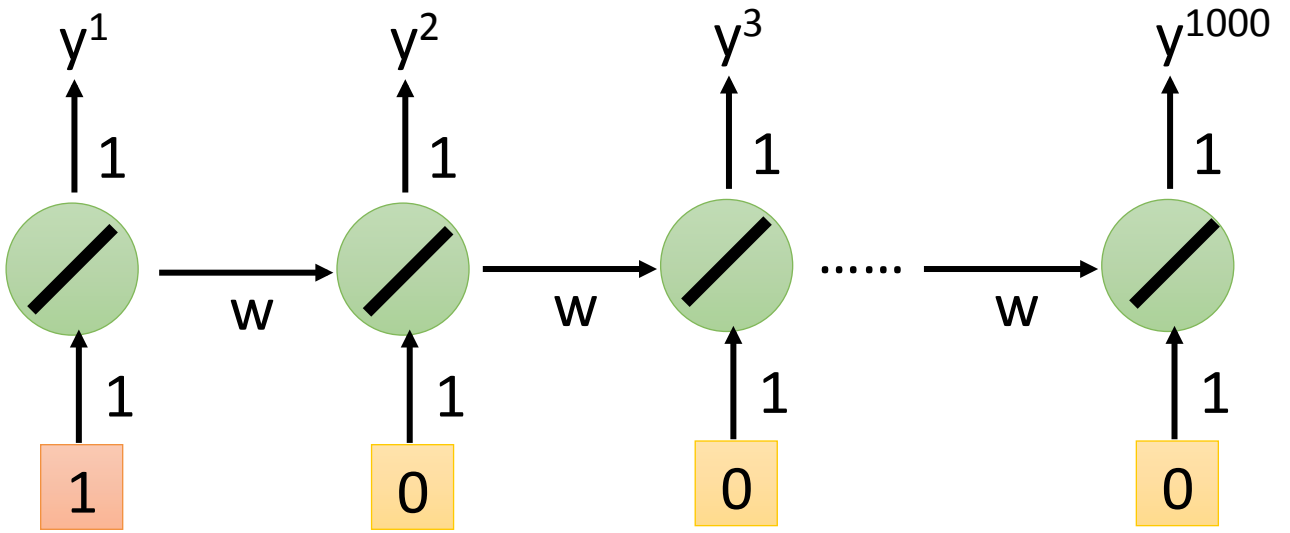
Large $\partial L/\partial w$ → Small Learning

$w=0.99$ → $y^{1000}\approx 0$

$w=0.01$ → $y^{1000}\approx 0$

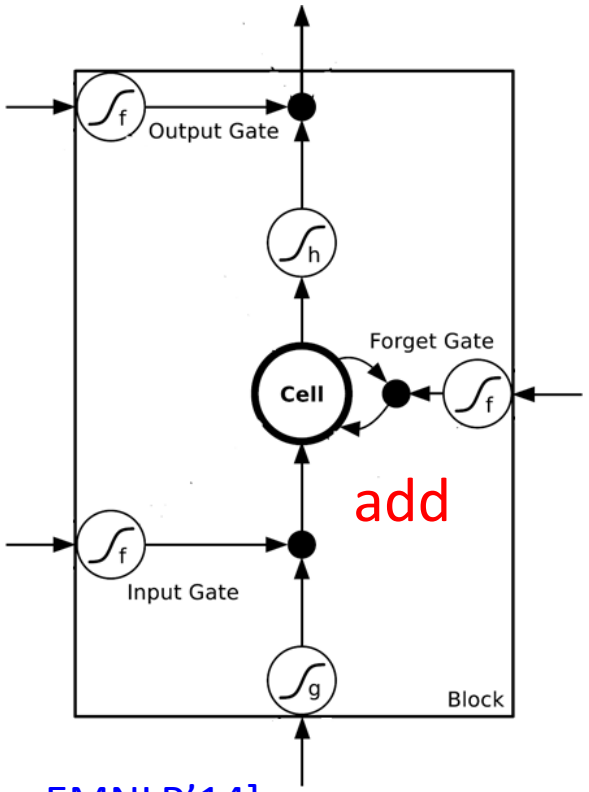
small $\partial L/\partial w$ → Large Learning

Toy Example



Helpful Techniques

- Long Short-term Memory (LSTM)
  - Can deal with gradient vanishing (not gradient explode)
    - Memory and input are ***added***
    - The influence never disappears unless forget gate is closed



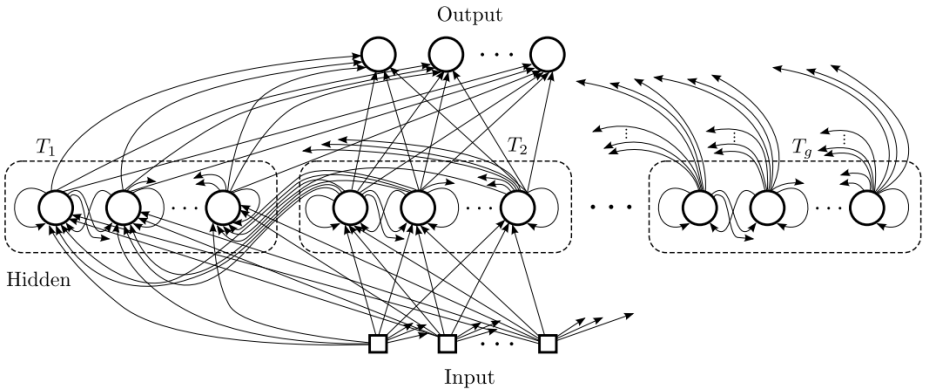
No Gradient vanishing
(If forget gate is opened.)

Gated Recurrent Unit (GRU): simpler than LSTM

[Cho, EMNLP'14]

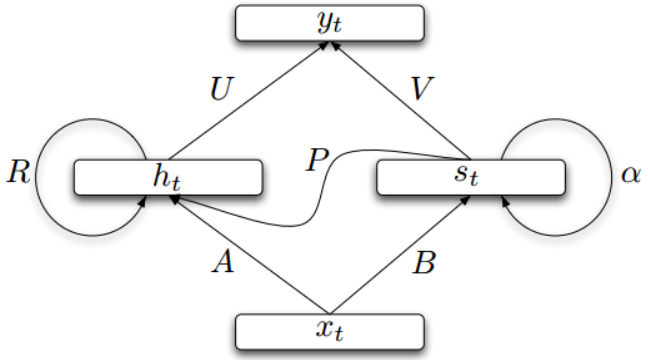
Helpful Techniques

Clockwise RNN



[Jan Koutnik, JMLR'14]

Structurally Constrained Recurrent Network (SCRN)



[Tomas Mikolov, ICLR'15]

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

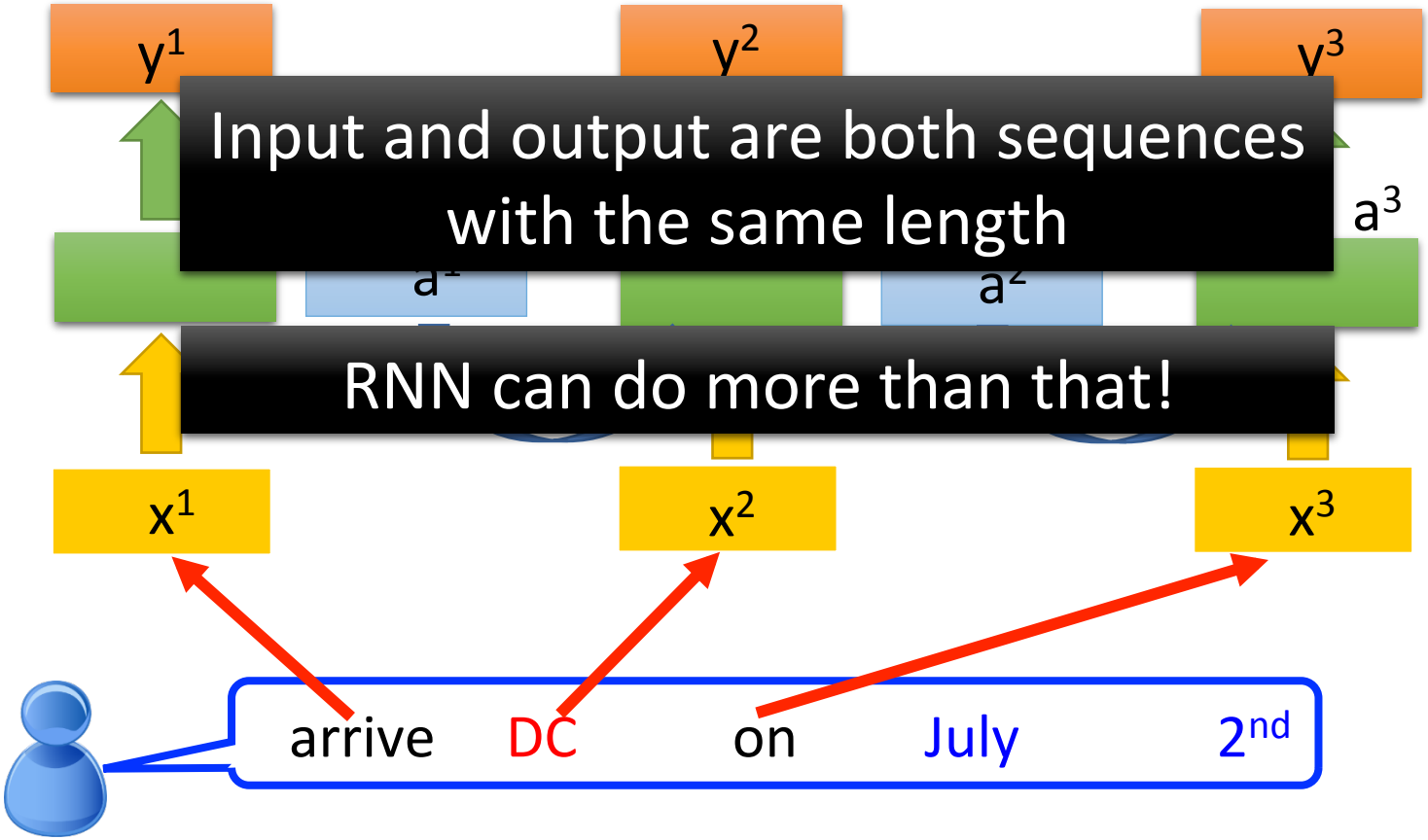
- Outperform or be comparable with LSTM in 4 different tasks

More Applications ……

Probability of "arrive" in each slot

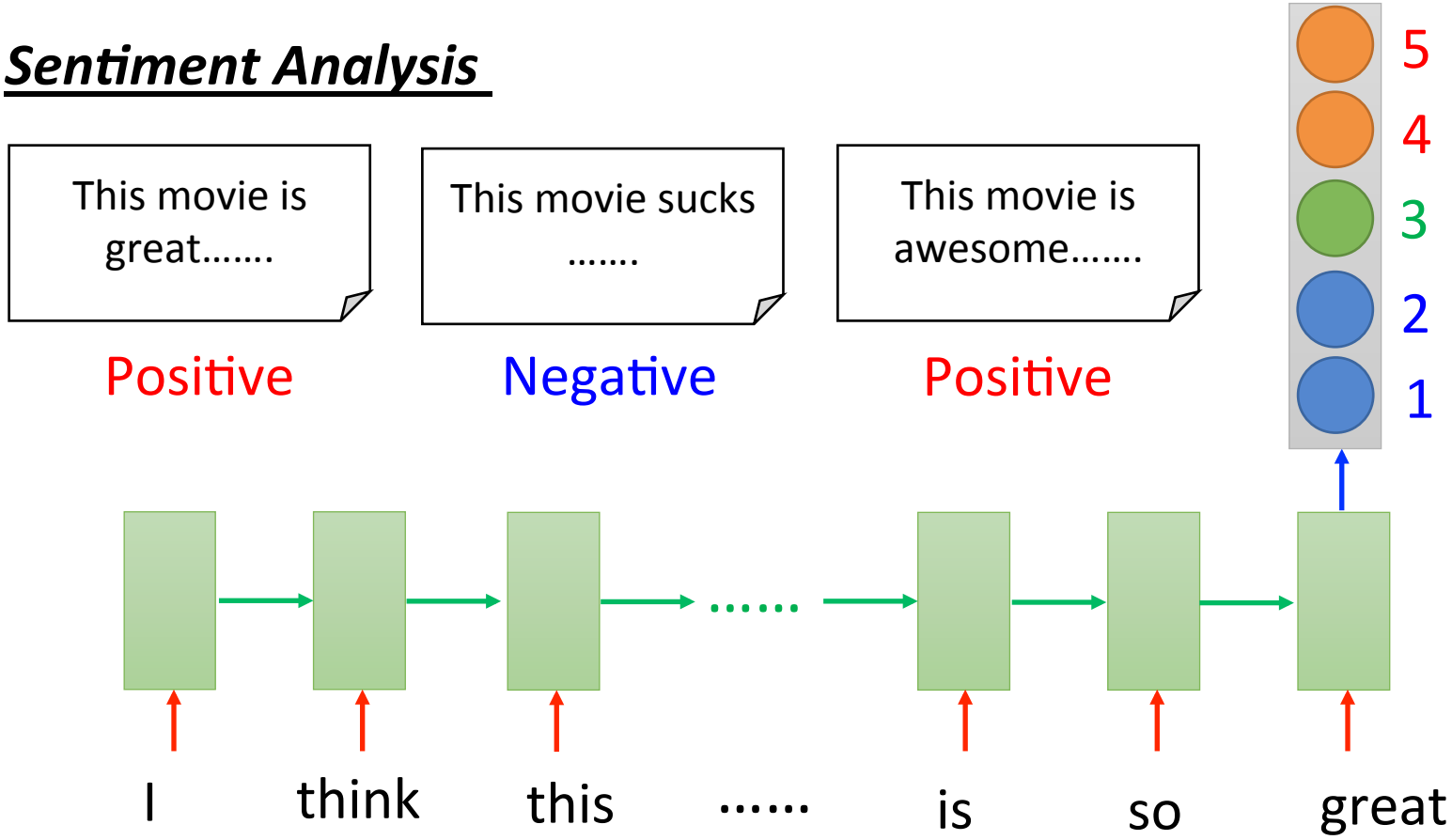
Probability of "DC" in each slot

Probability of "on" in each slot



Many to one

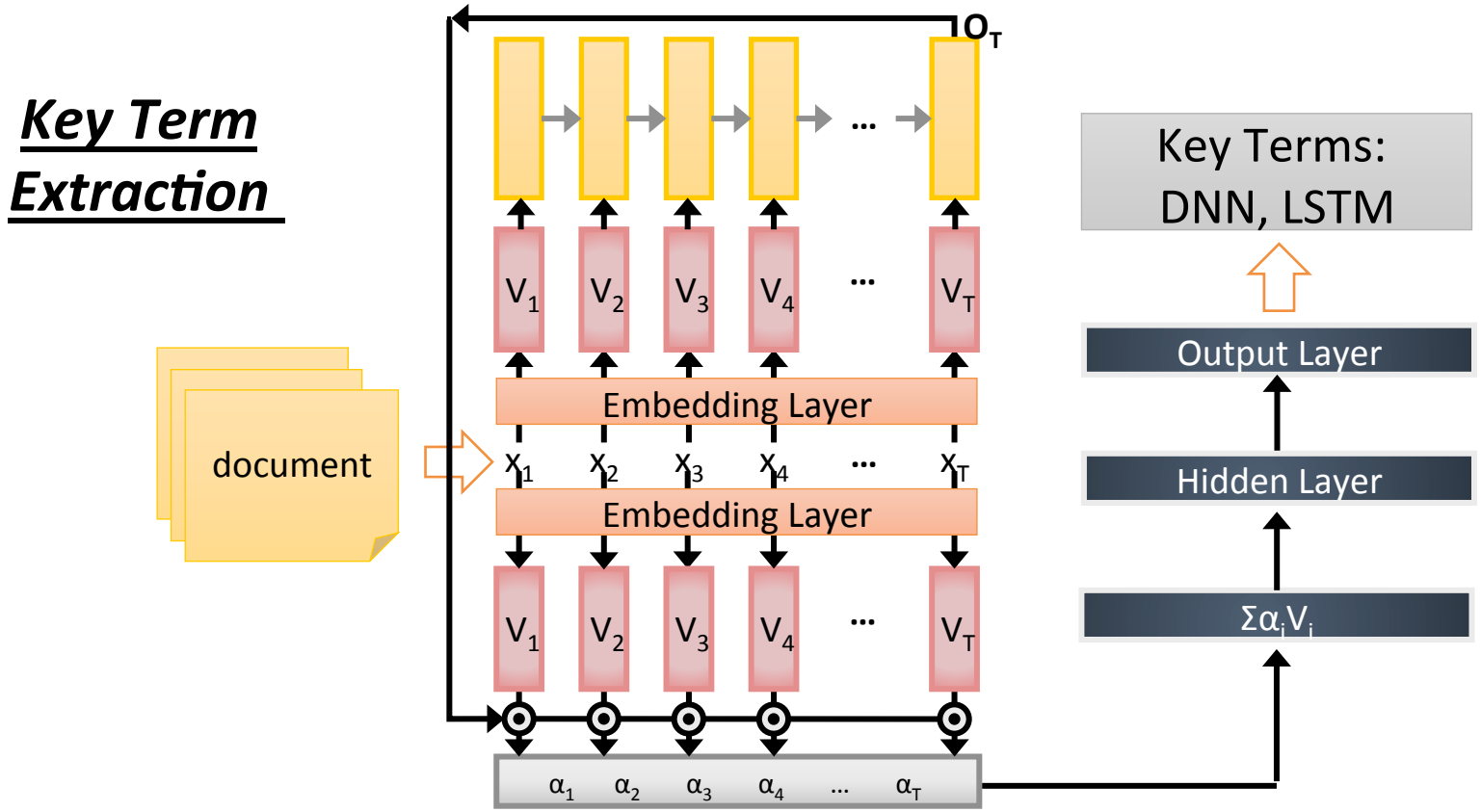
- Input is a vector sequence, but output is only one vector



Many to one

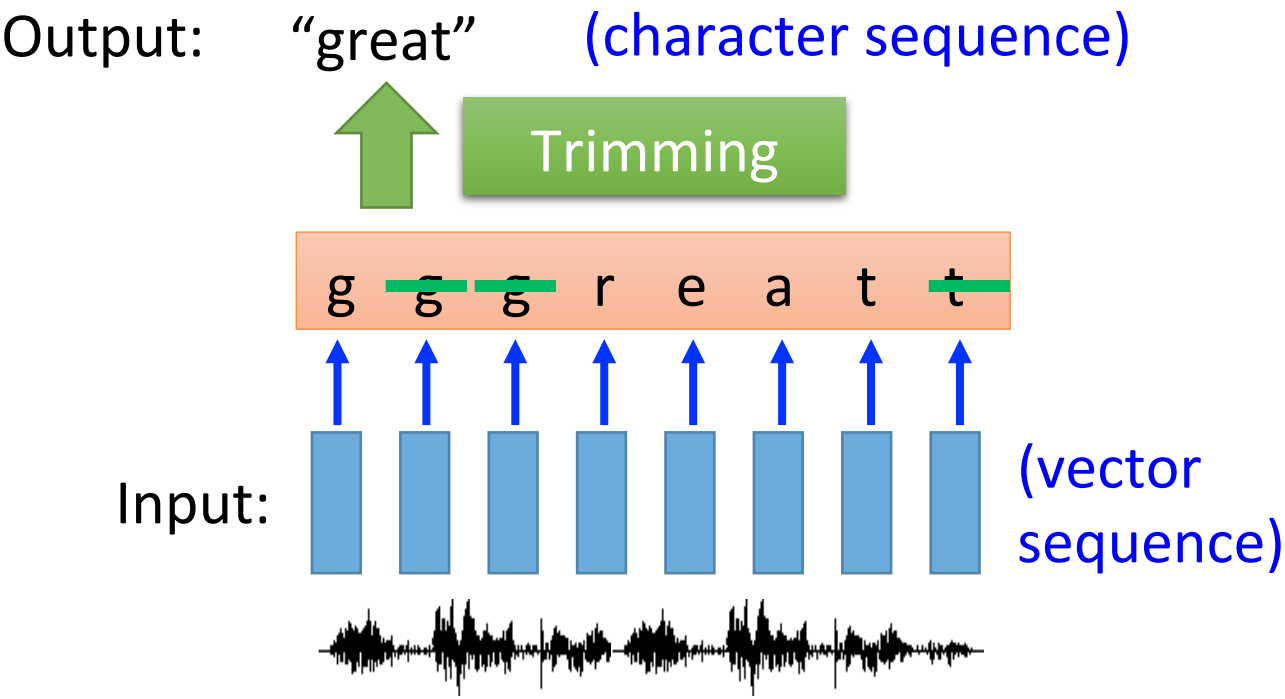
[Shen & Lee, Interspeech 16]

- Input is a vector sequence, but output is only one vector



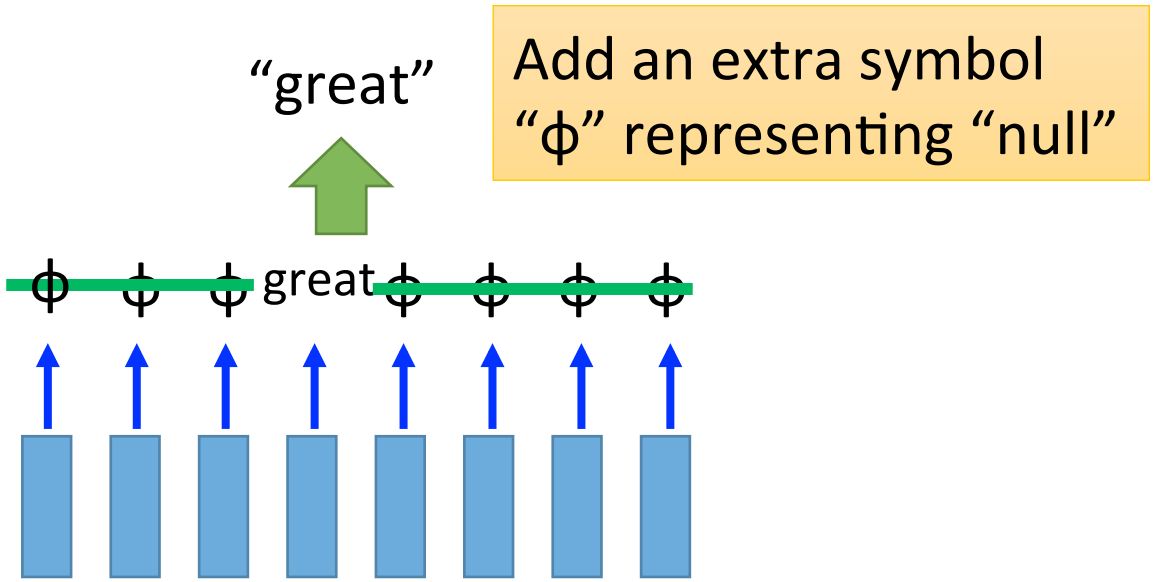
Many to Many (Output is shorter)

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



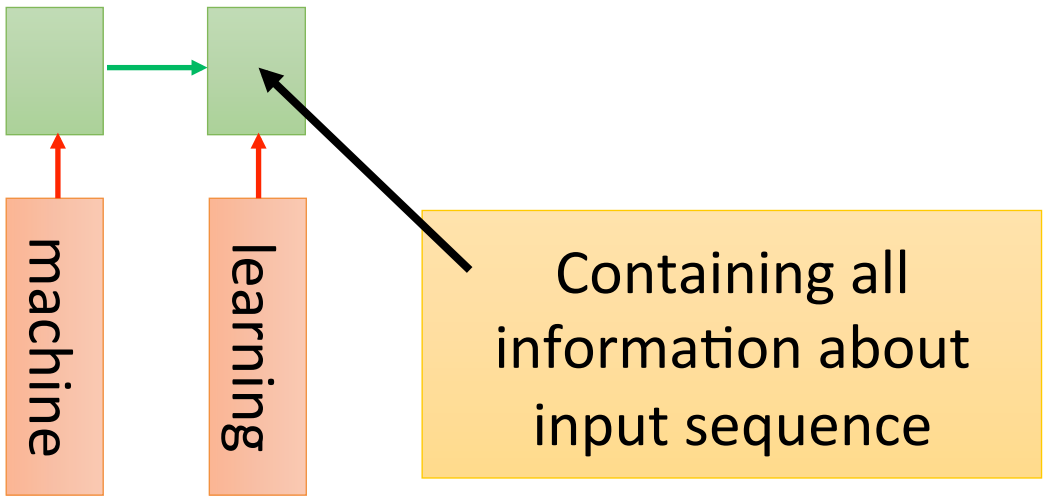
Many to Many (Output is shorter)

- 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]



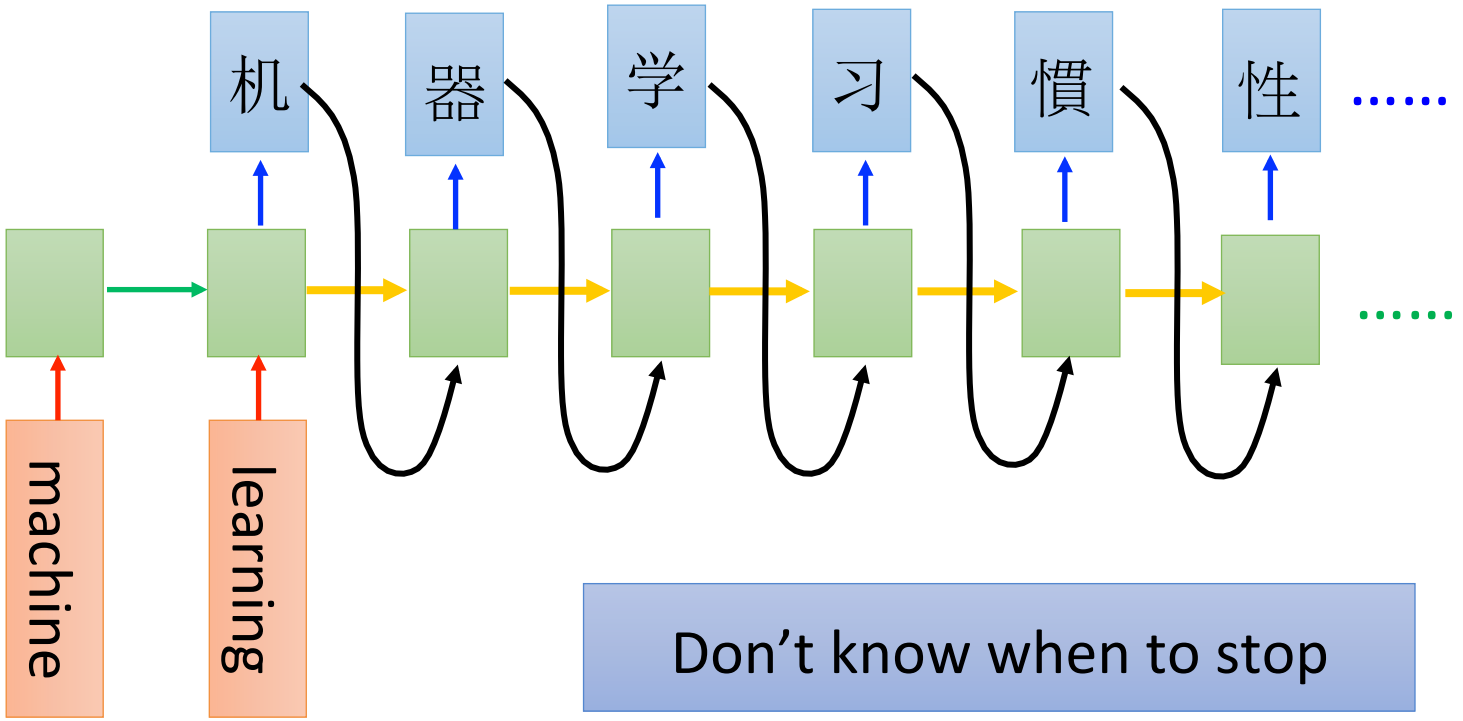
Many to Many (No Limitation)

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



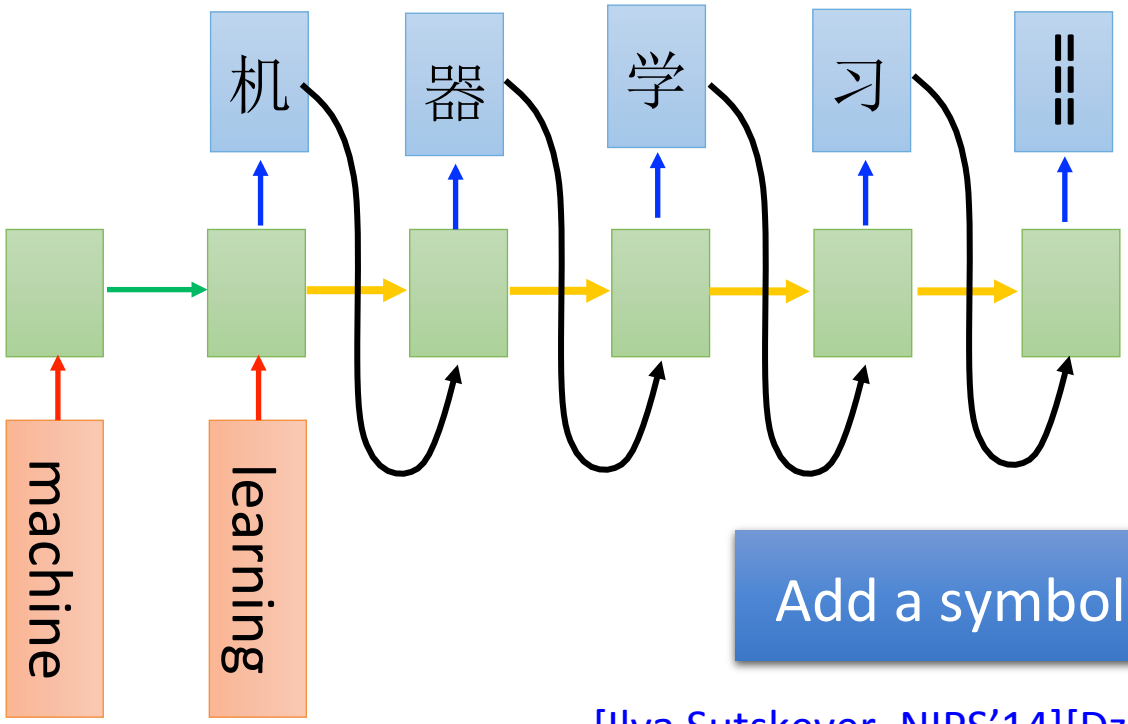
Many to Many (No Limitation)

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



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→机器学习)

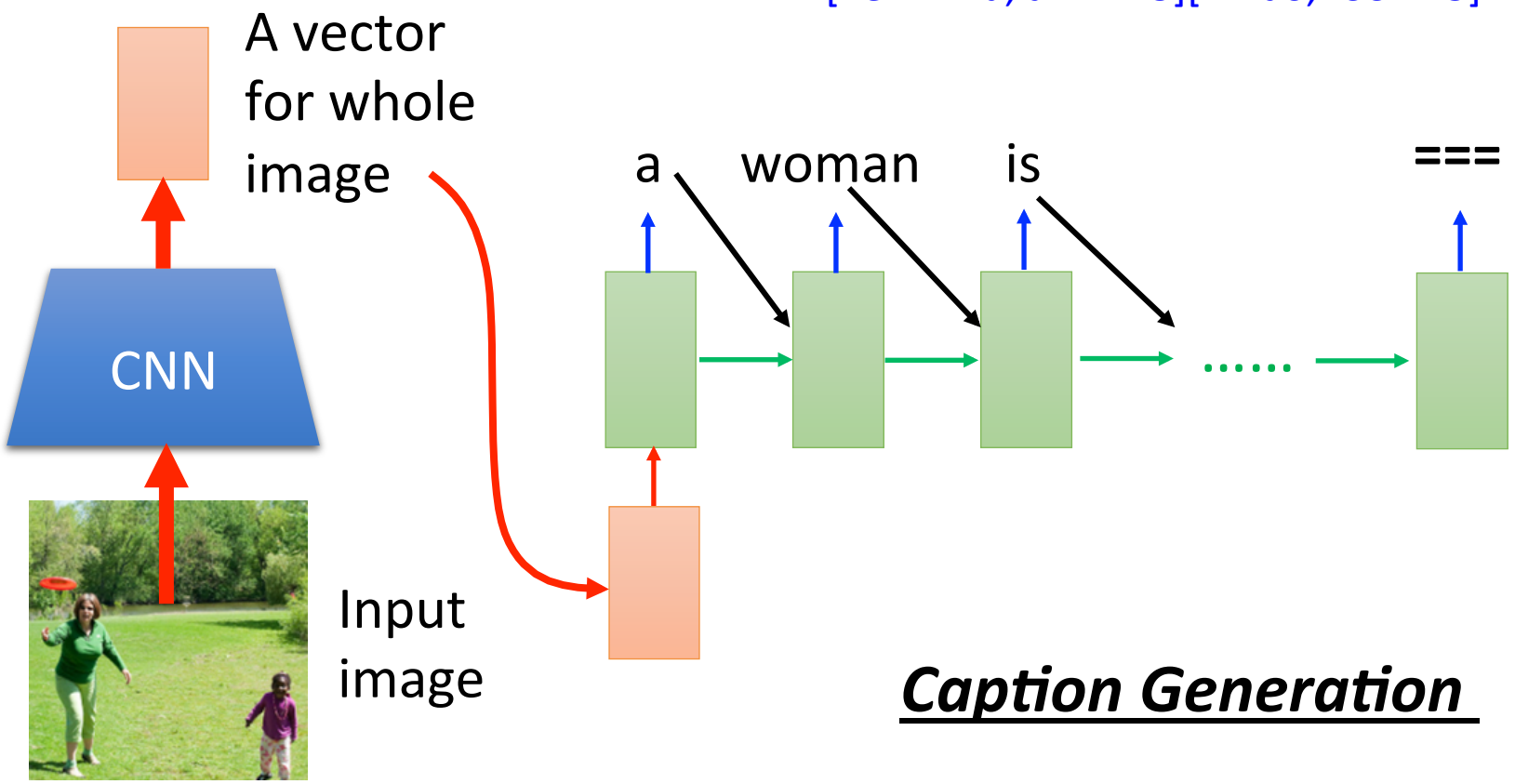


[Ilya Sutskever, NIPS'14][Dzmitry Bahdanau, arXiv'15]

Image Caption Generation

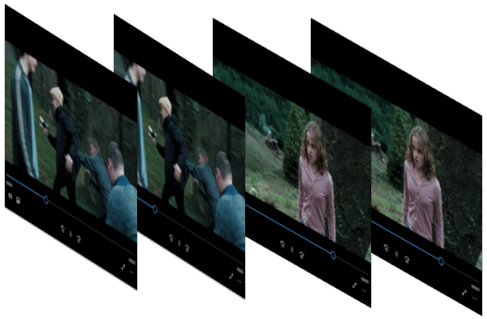
- Input an image, but output a sequence of words

[Kelvin Xu, arXiv'15][Li Yao, ICCV'15]



Caption Generation

Video Caption Generation



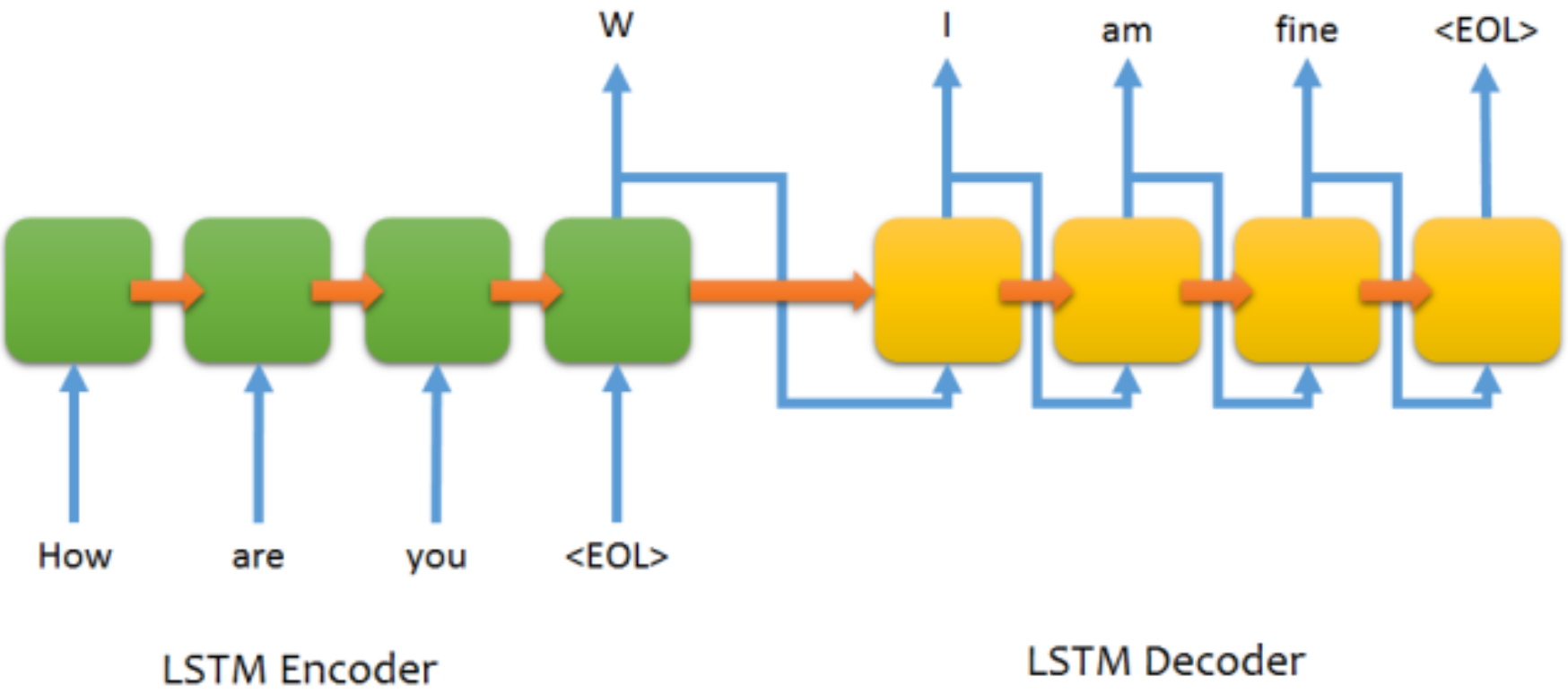
Video

A group of people is knocked by a tree.

A girl is running.

A group of people is walking in the forest.

Chat-bot



Movie (~40,000 sentences), presidential debate…

Concluding Remarks

Convolutional Neural Network (CNN)

Recurrent Neural Network (RNN)

Part III:
Beyond Supervised Learning

Outline

Unsupervised Learning

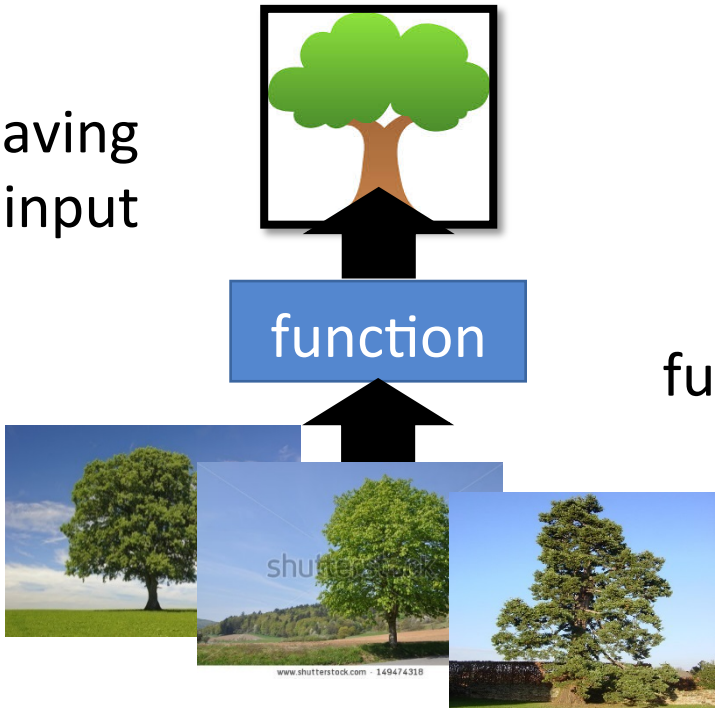
- Auto-encoder
- Word Vector and Audio Word Vector

Reinforcement Learning

Unsupervised Learning

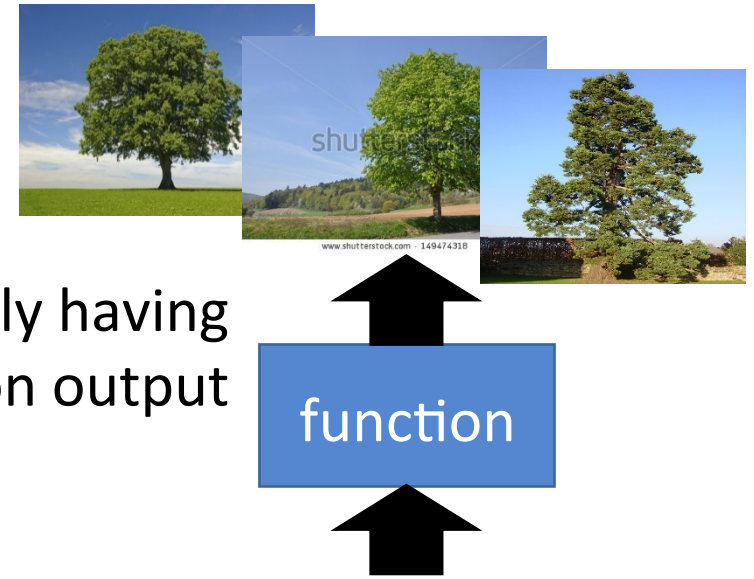
- abstraction

only having function input



- Out of nothing

only having function output



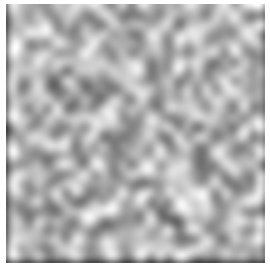
Outline

Unsupervised Learning

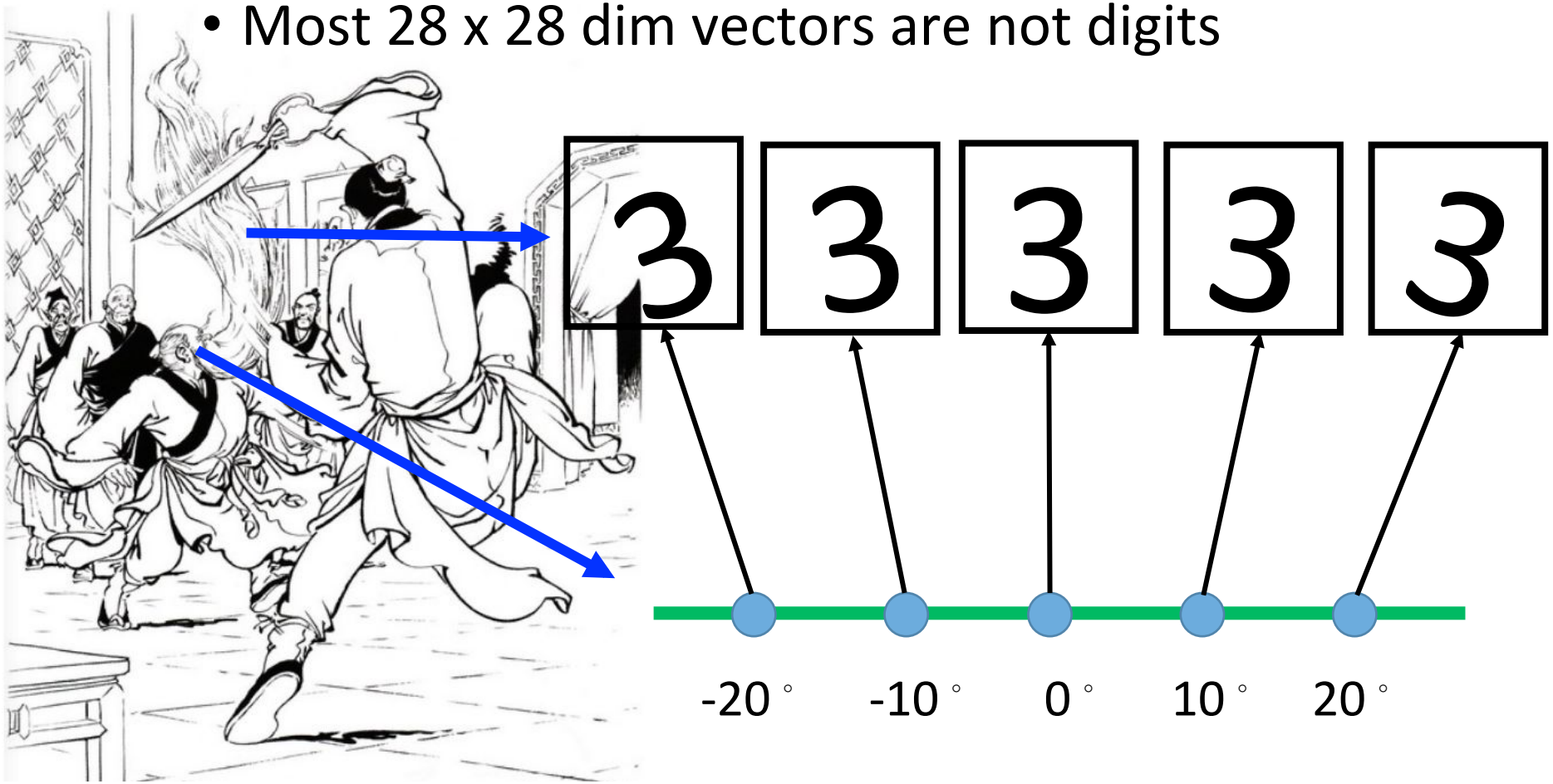
- abstraction
  - Auto-encoder
  - Word Vector and Audio Word Vector
- Out of nothing

Reinforcement Learning

Motivation



- In MNIST, a digit is 28 x 28 dims.
  - Most 28 x 28 dim vectors are not digits



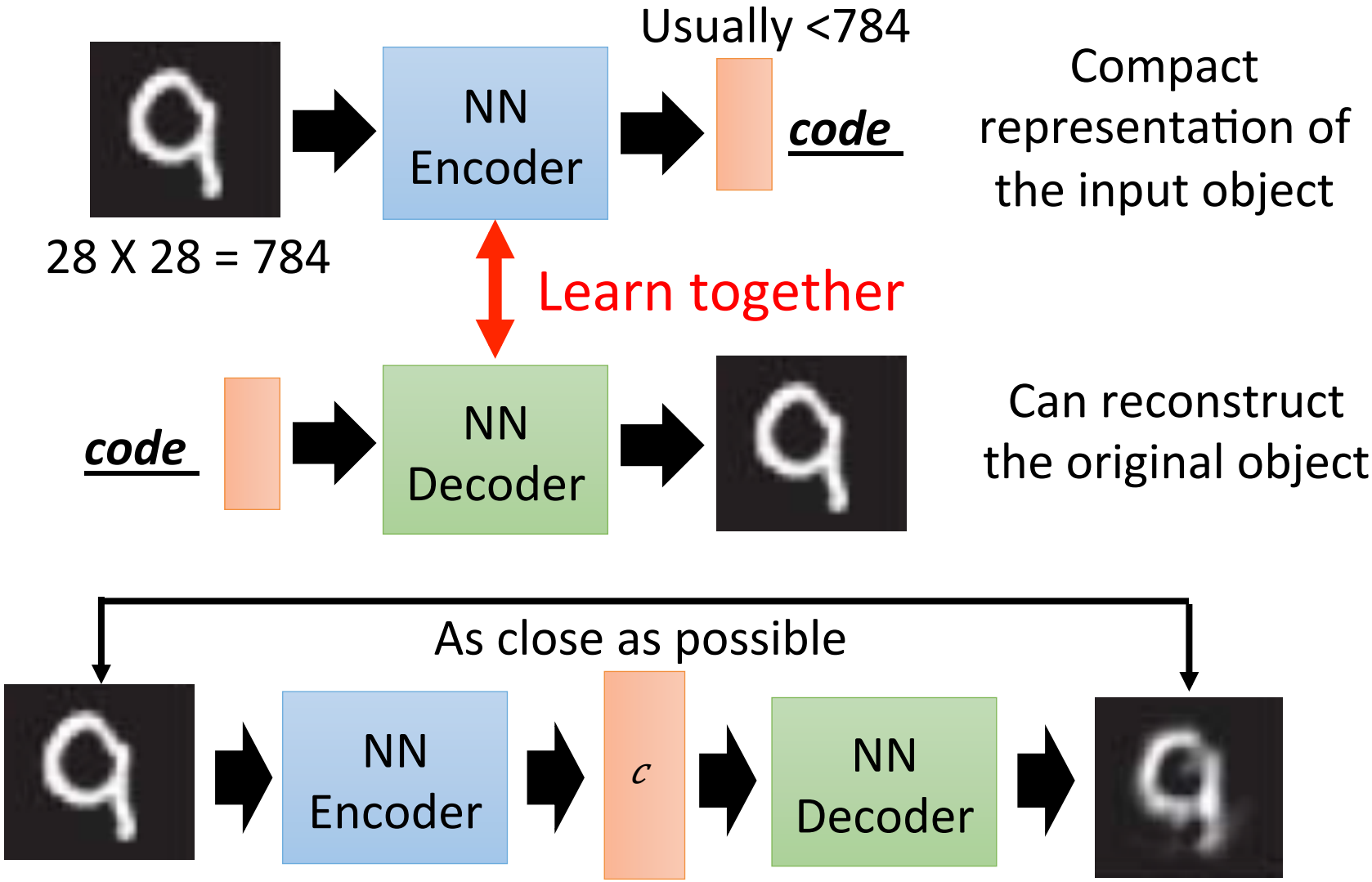
Outline

Unsupervised Learning

- Abstraction
  - Auto-encoder
  - Word Vector and Audio Word Vector
- Out of nothing

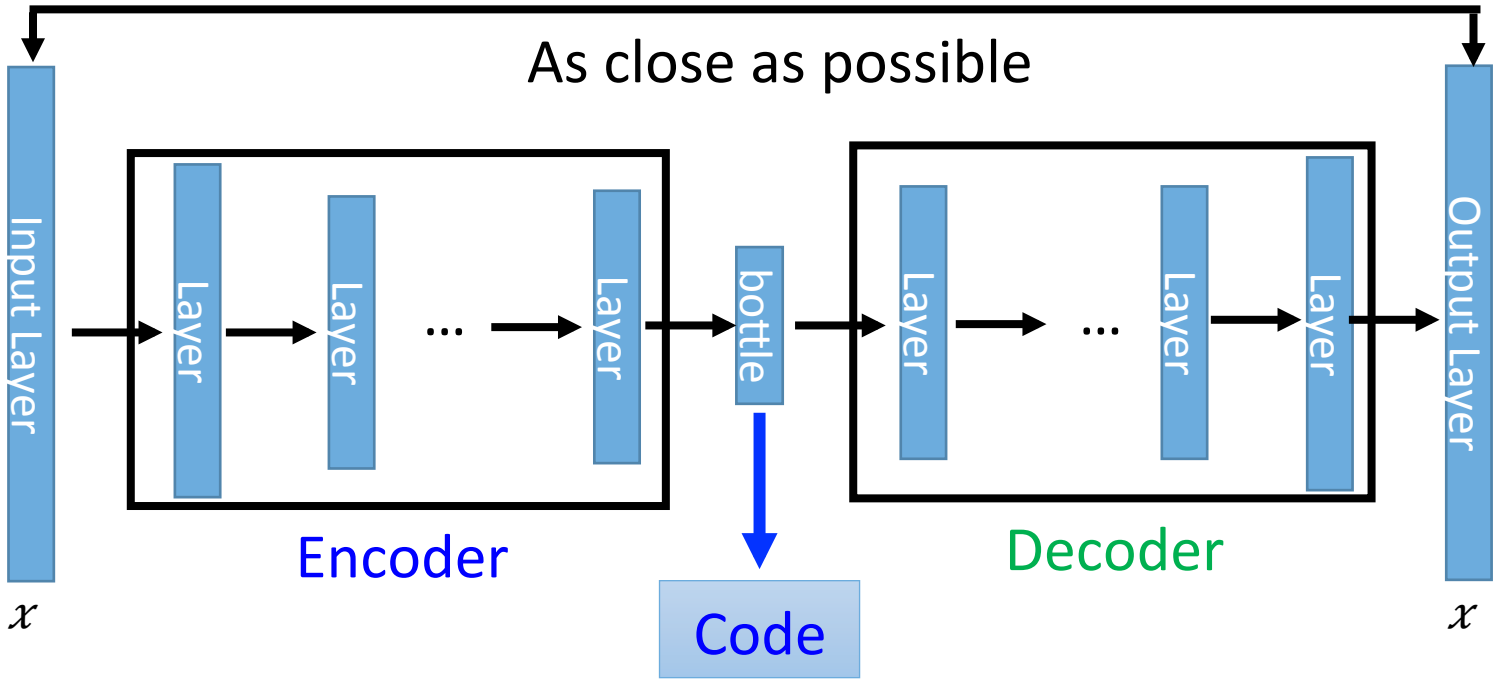
Reinforcement Learning

Auto-encoder



Deep Auto-encoder

- NN encoder + NN decoder = a deep network



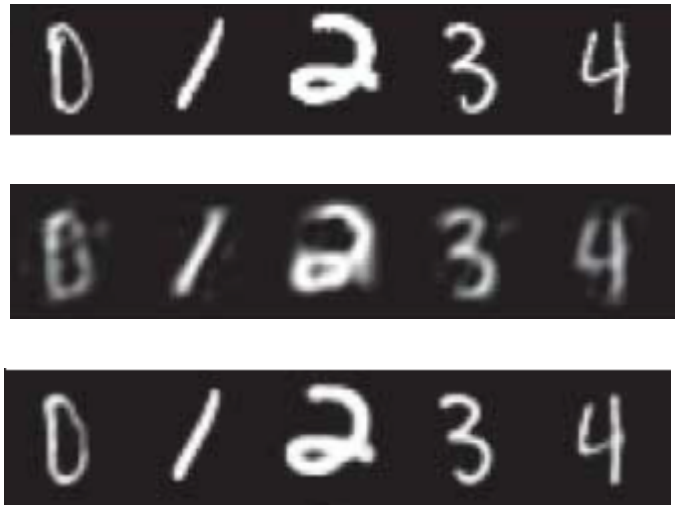
Reference: Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." *Science* 313.5786 (2006): 504-507

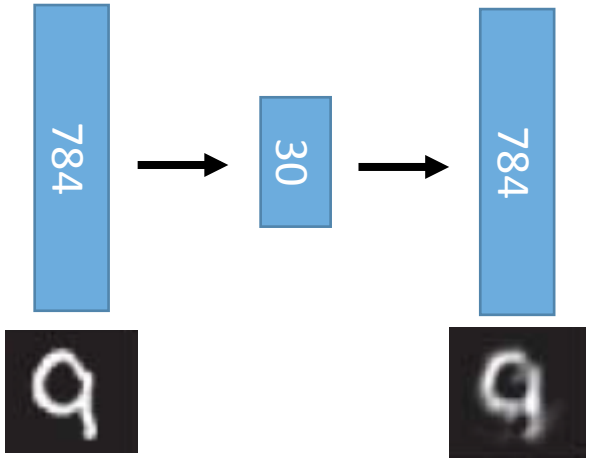
Deep Auto-encoder

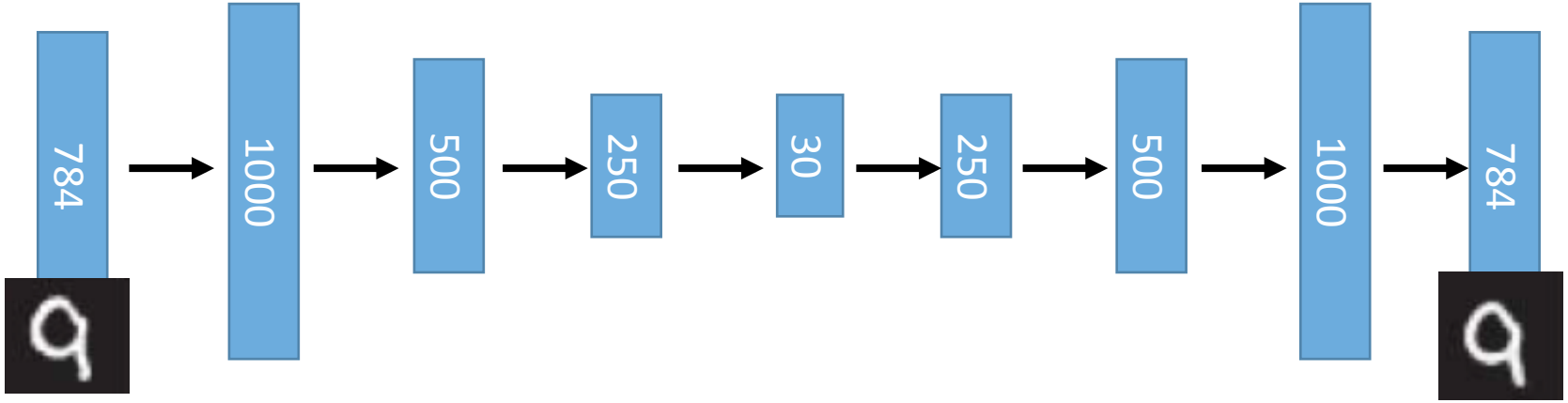
Original Image

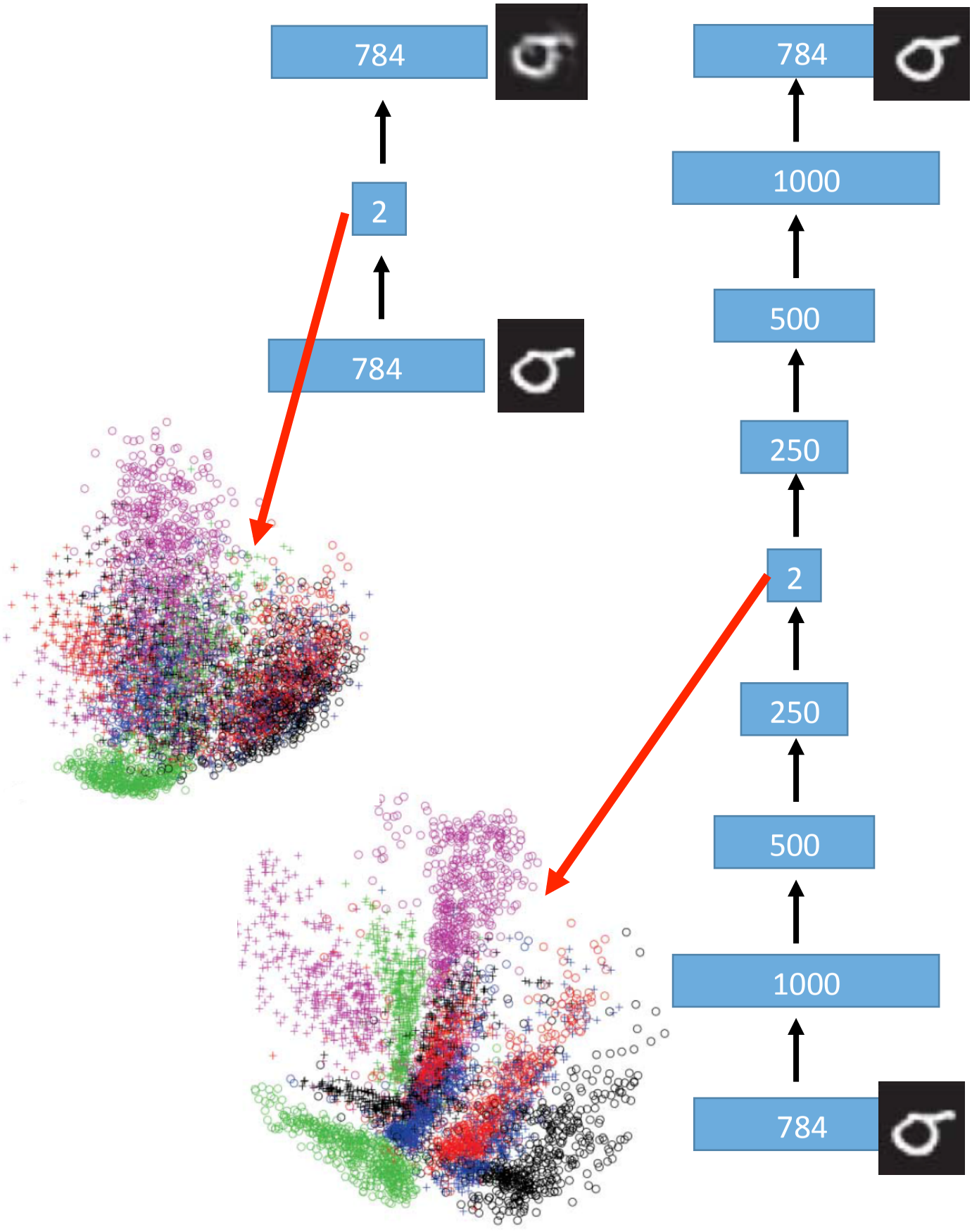
PCA

Deep Auto-encoder







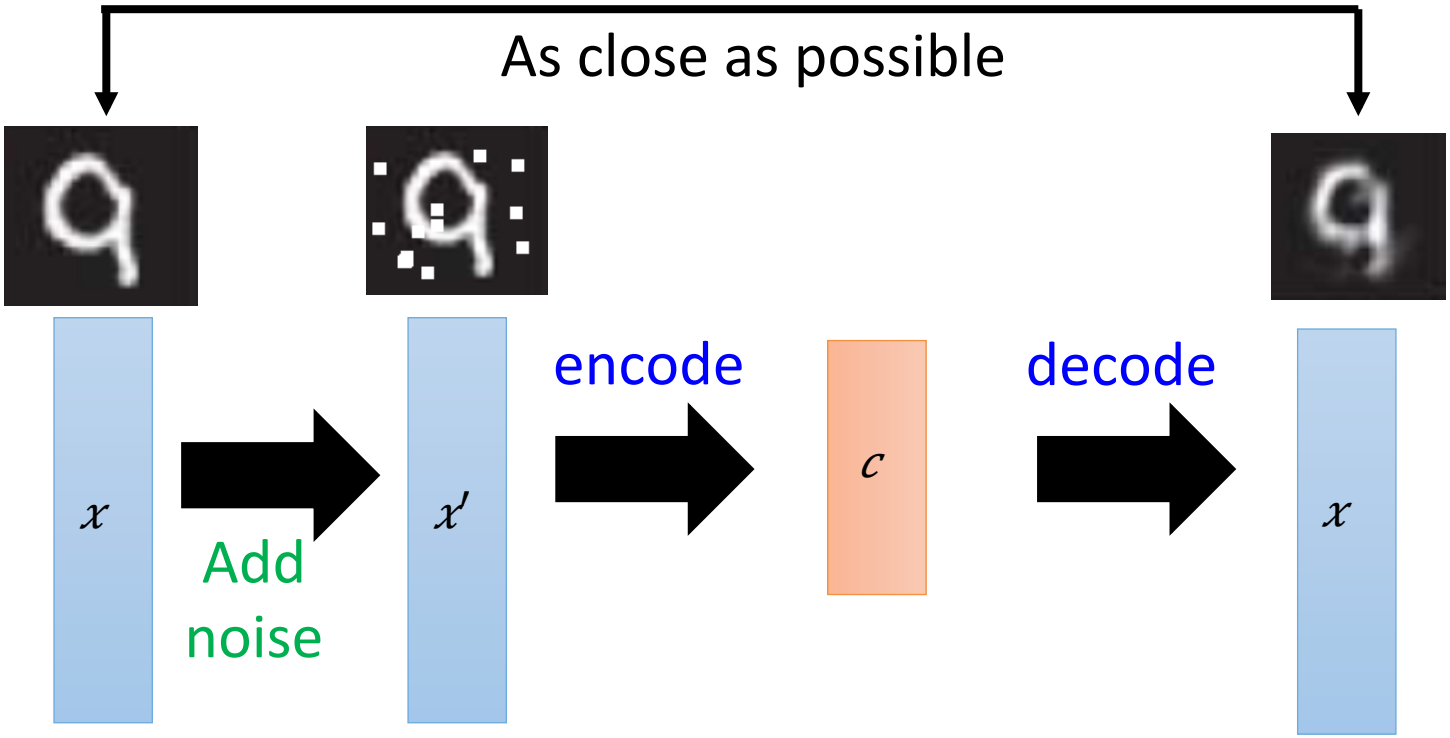

784
2
784
784
1000
500
250
2
250
500
1000
784

Auto-encoder

More: Contractive auto-encoder

Ref: Rifai, Salah, et al. "Contractive auto-encoders: Explicit invariance during feature extraction." *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*. 2011.

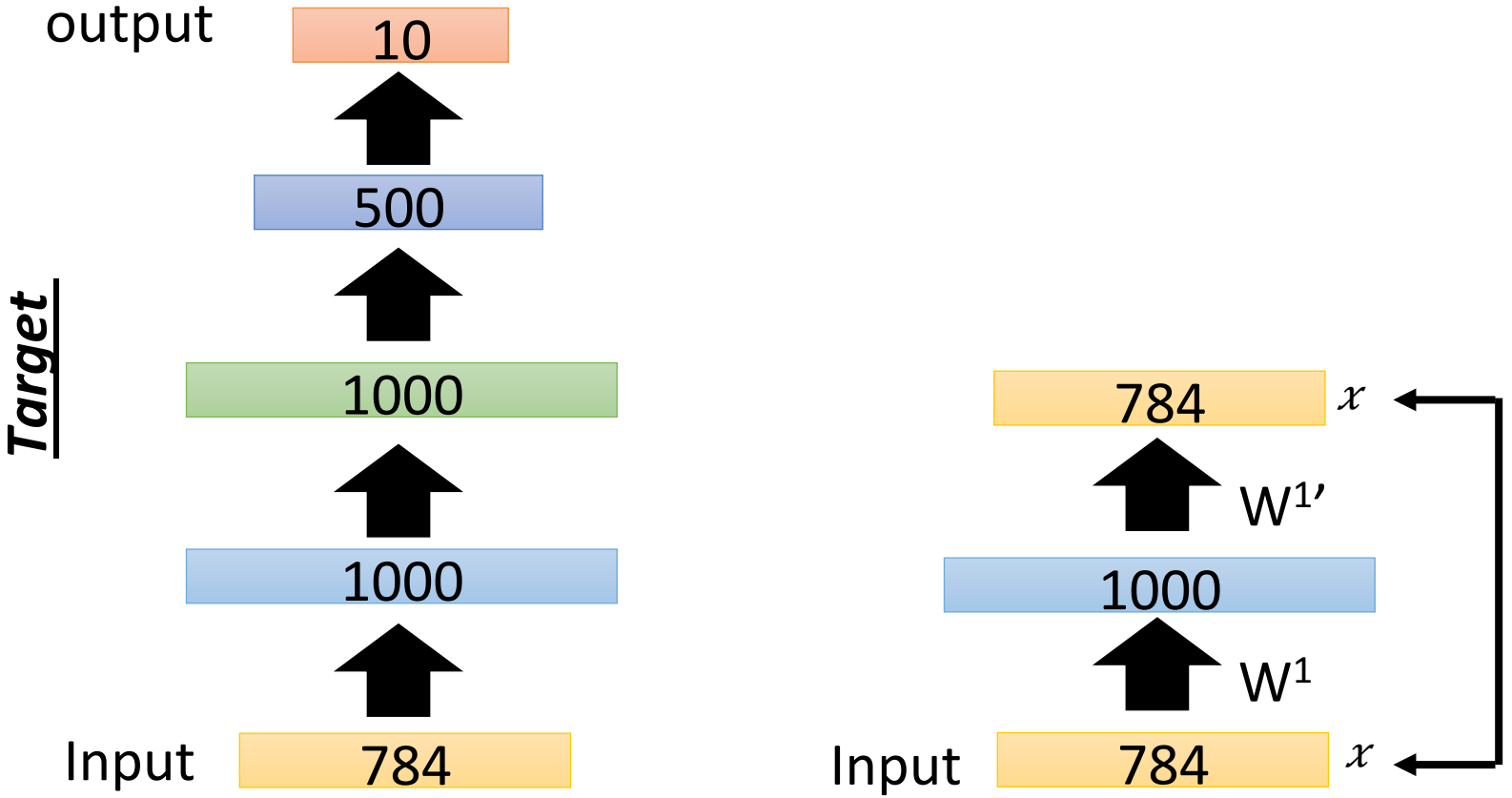
- De-noising auto-encoder



Vincent, Pascal, et al. "Extracting and composing robust features with denoising autoencoders." *ICML,* 2008.

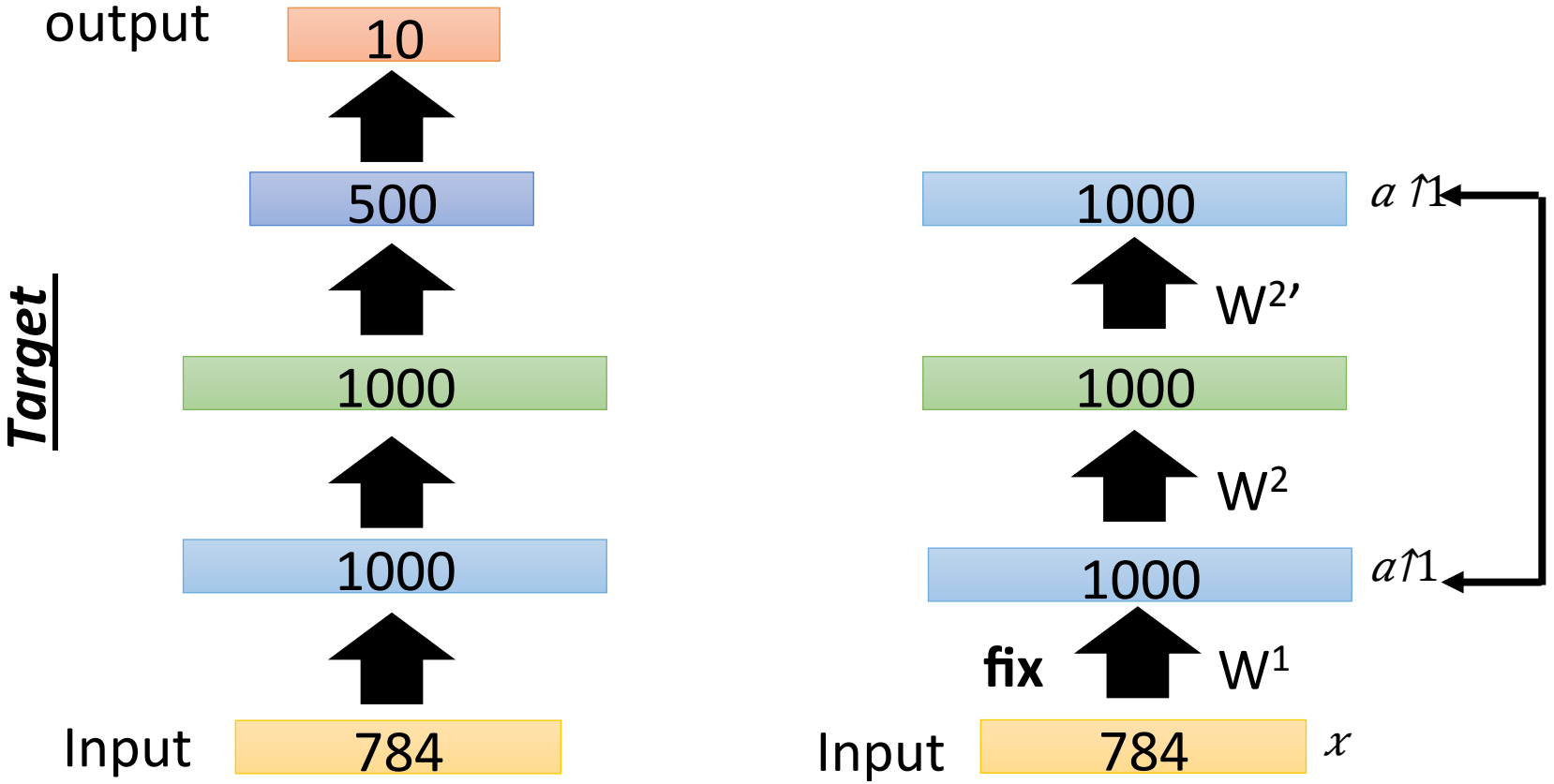
Auto-encoder – Pre-training DNN

- Greedy Layer-wise Pre-training *again*



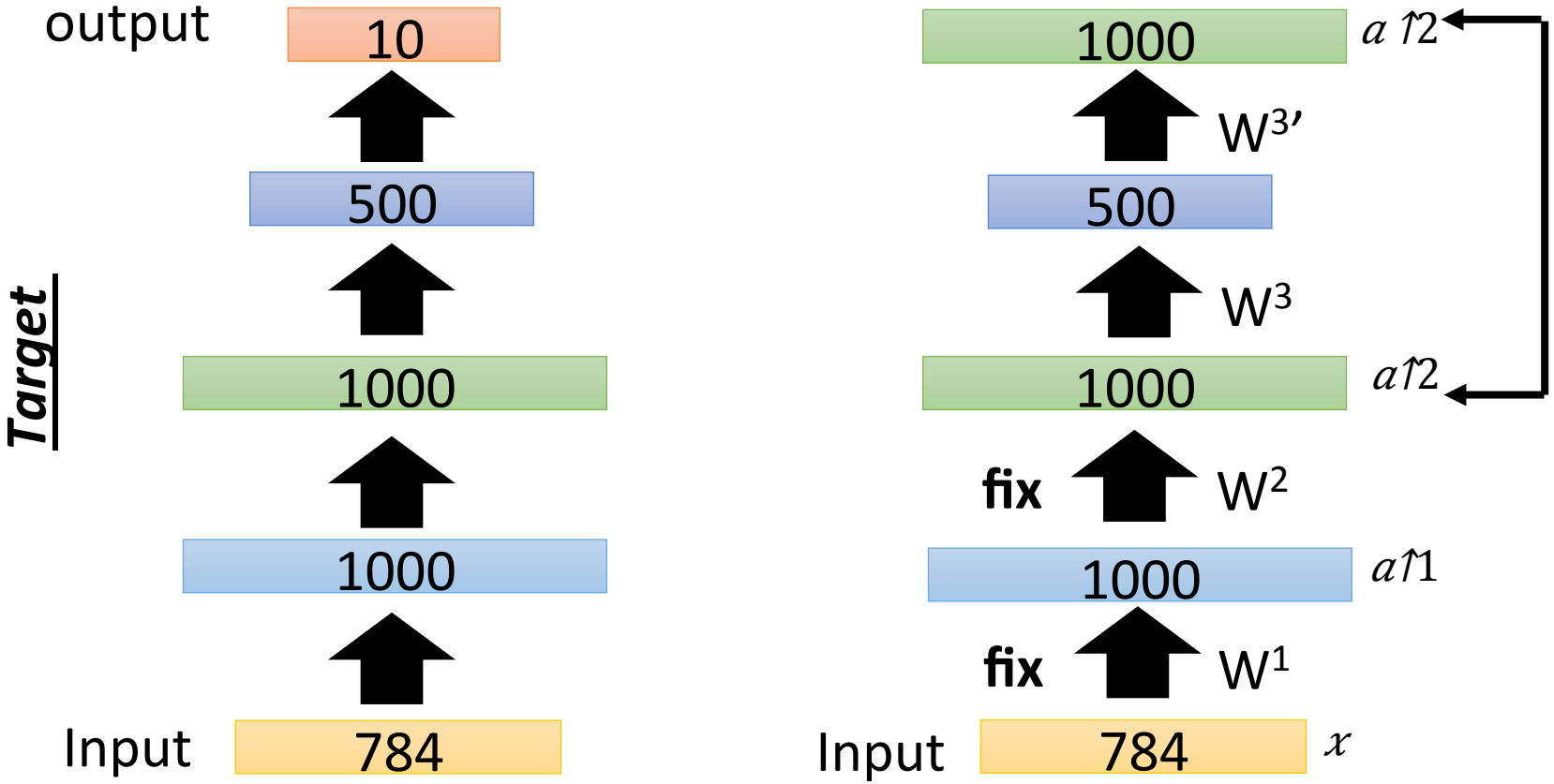
Auto-encoder – Pre-training DNN

- Greedy Layer-wise Pre-training *again*



Auto-encoder – Pre-training DNN

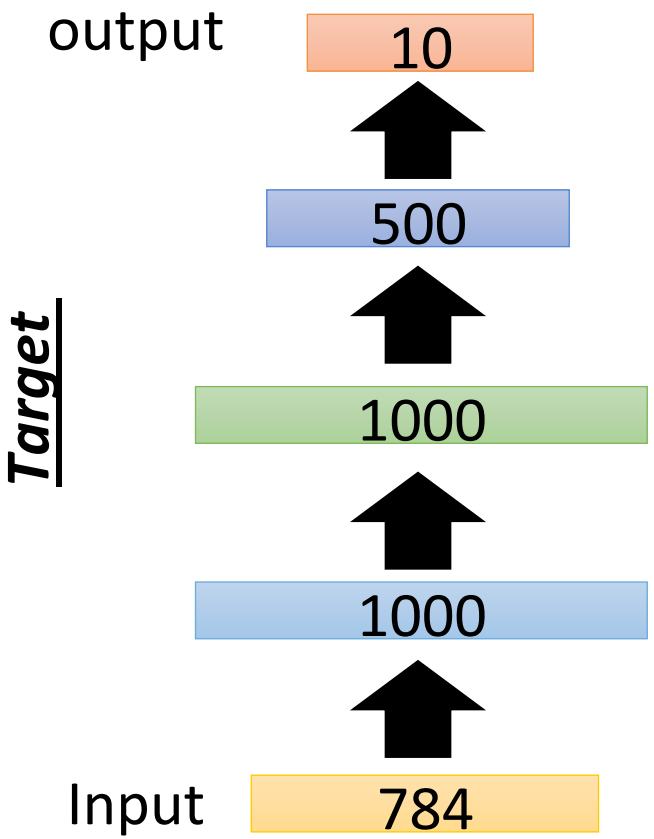
- Greedy Layer-wise Pre-training *again*

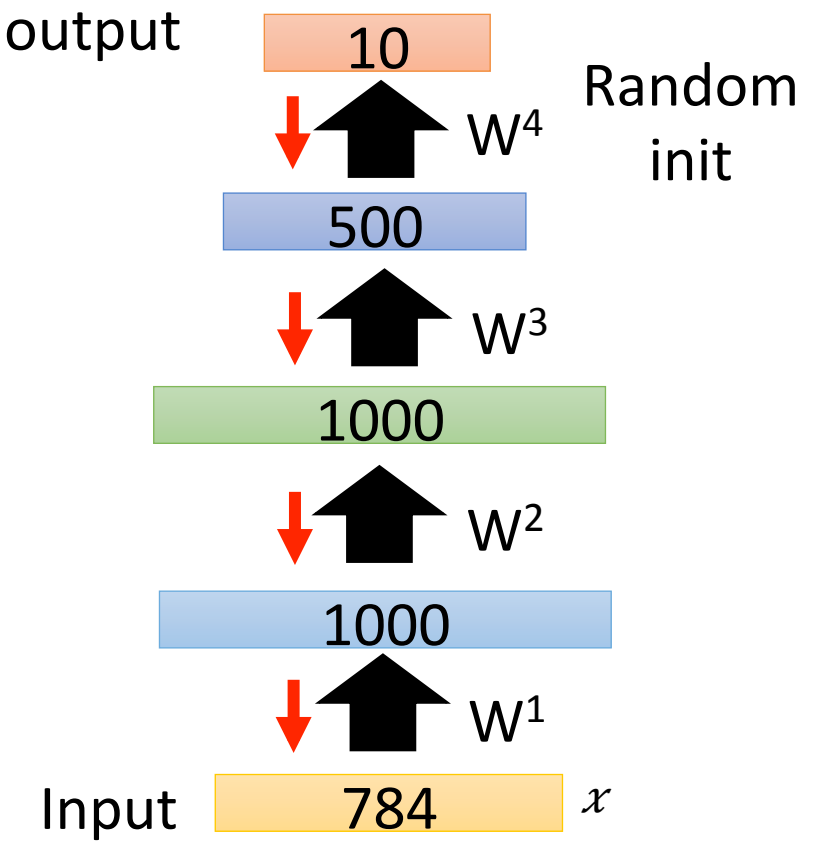


Auto-encoder – Pre-training DNN

- Greedy Layer-wise Pre-training *again*

Fine-tune by backpropagation





Outline

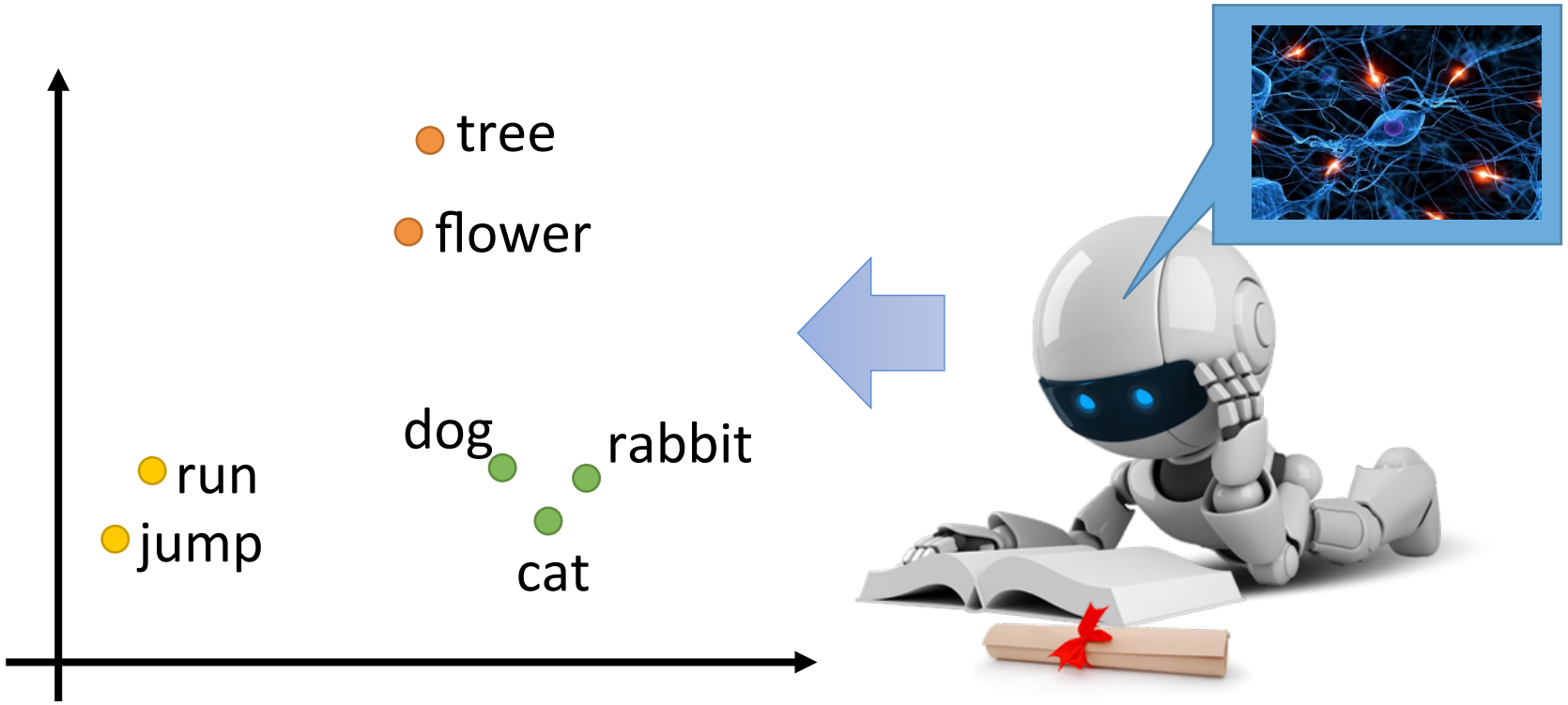
Unsupervised Learning

- Abstraction
  - Auto-encoder
  - Word Vector and Audio Word Vector
- Out of nothing

Reinforcement Learning

Word Vector/Embedding

- Machine learns the meaning of words from reading a lot of documents without supervision



Word Embedding

- Machine learns the meaning of words from reading a lot of documents without supervision
- A word can be understood by its context

D.C. , Beijing are something very similar

You shall know a word by its context

Beijing is the capital

D. C. is the capital

How to exploit the context?

- **Count based**
  - If two words w_i and w_j frequently co-occur, $V(w_i)$ and $V(w_j)$ would be close to each other
  - E.g. Glove Vector: http://nlp.stanford.edu/projects/glove/

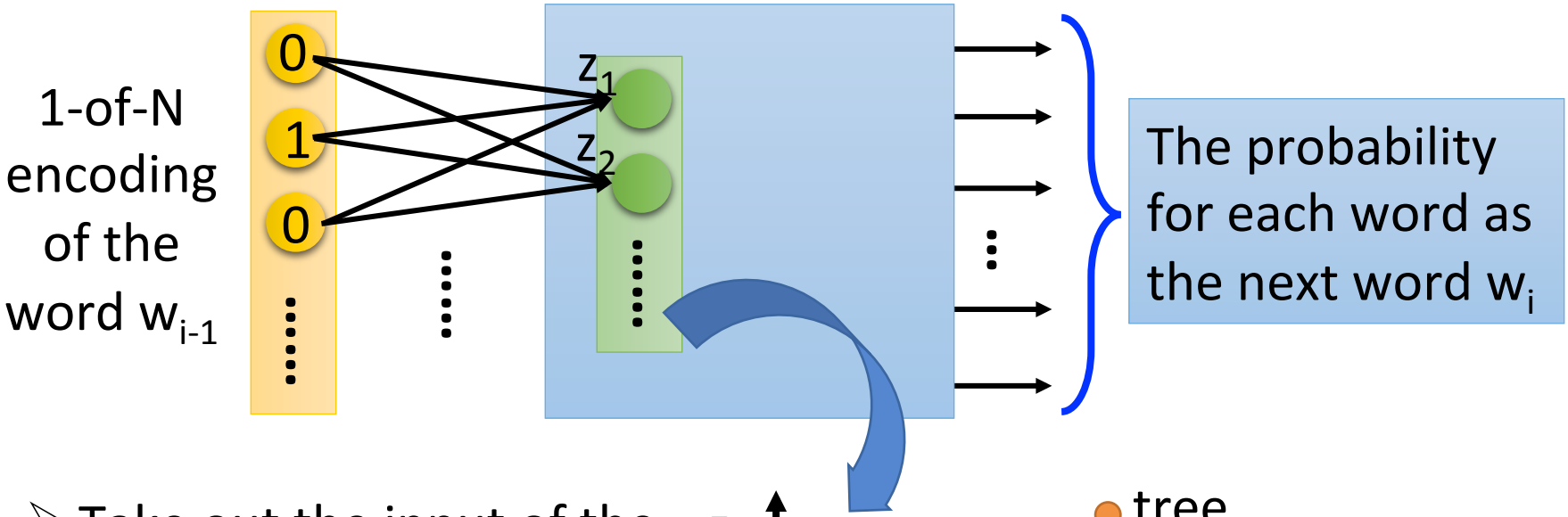
$V(w_i) \cdot V(w_j)$ ⟷ $N_{i,j}$

Inner product

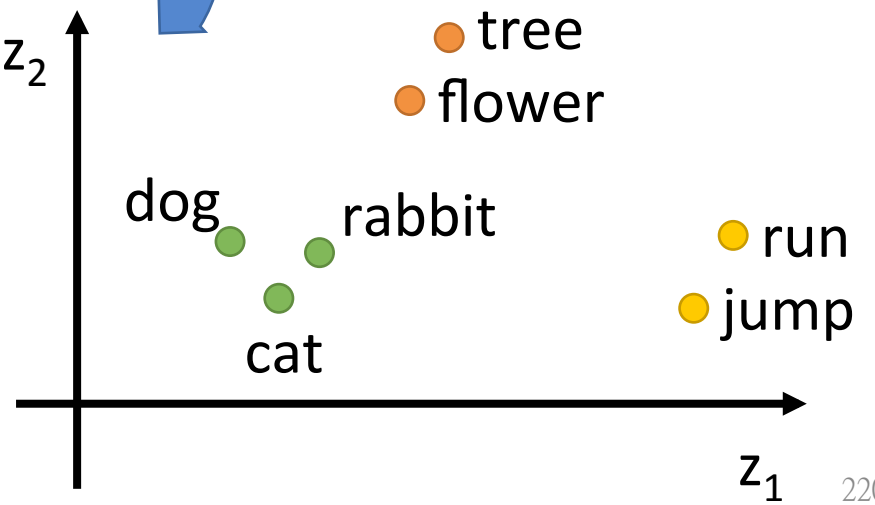
Number of times w_i and w_j in the same document

- **Prediction based**

Prediction-based



- Take out the input of the neurons in the first layer
- Use it to represent a word w
- Word vector, word embedding feature: V(w)

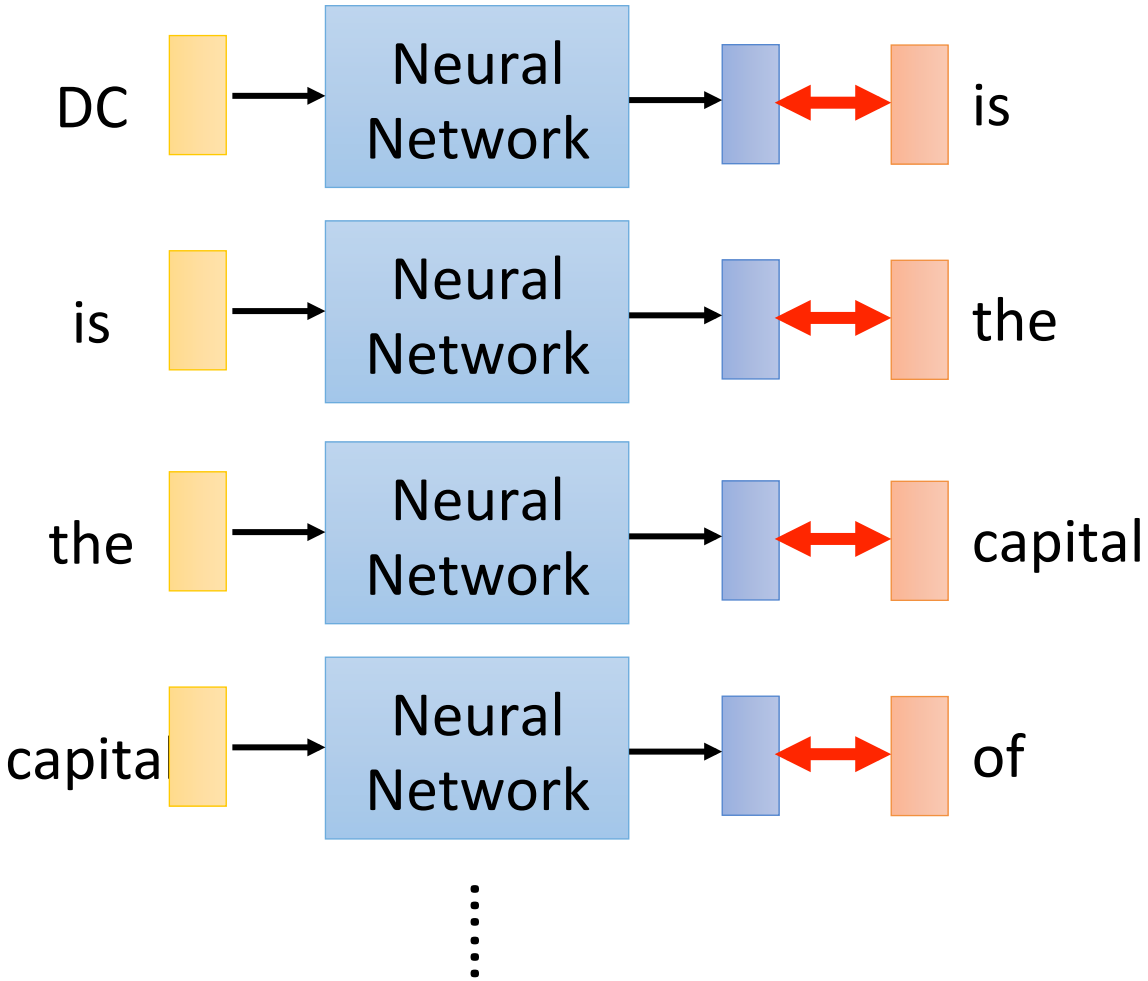


Prediction-based

Minimizing cross entropy

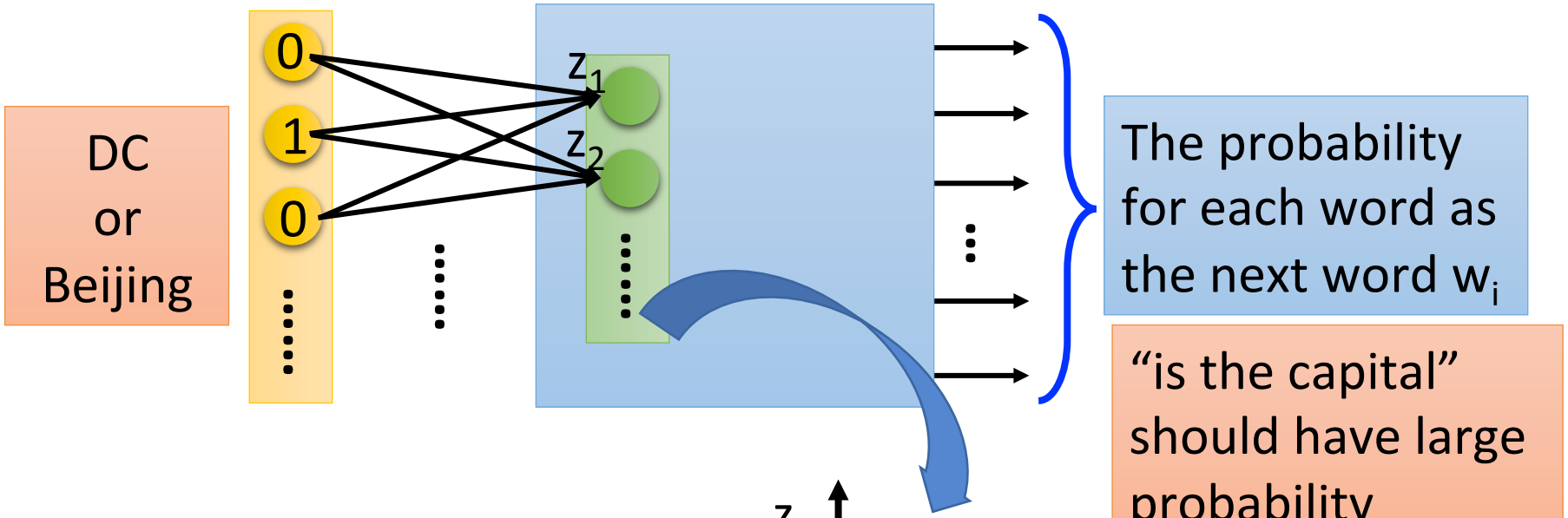
Collect data:

DC is the capital of USA…
UMD is in the DC area…
……



Prediction-based

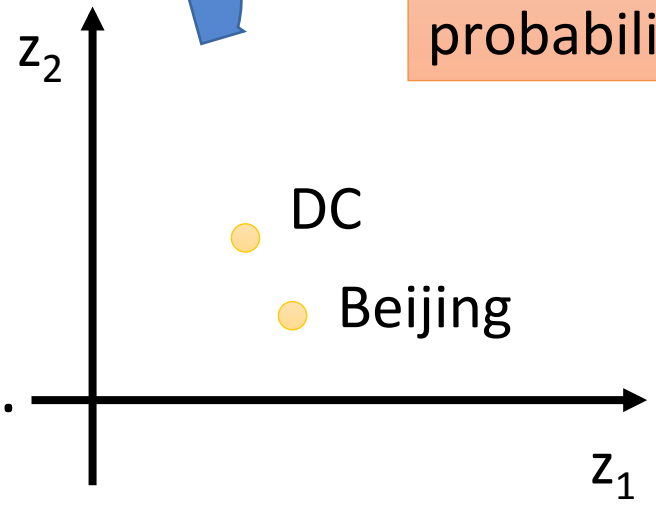
You shall know a word by its context



Training text:

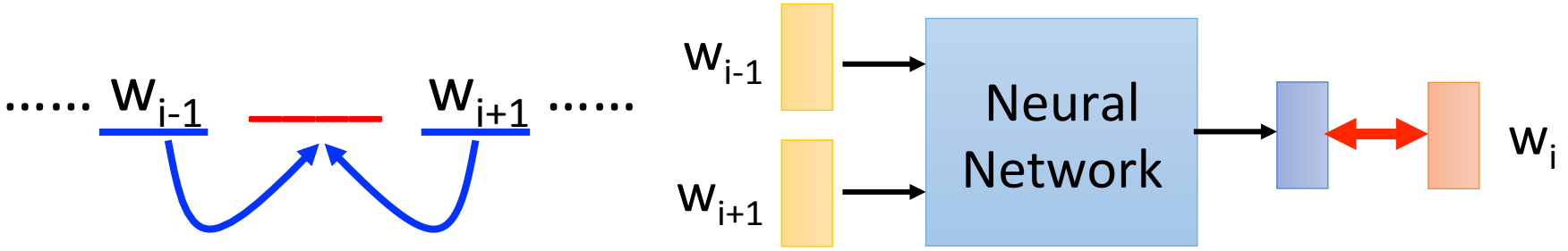
…… DC is the capital……
w_{i-1} w_i

…… Beijing is the capital……
w_{i-1} w_i



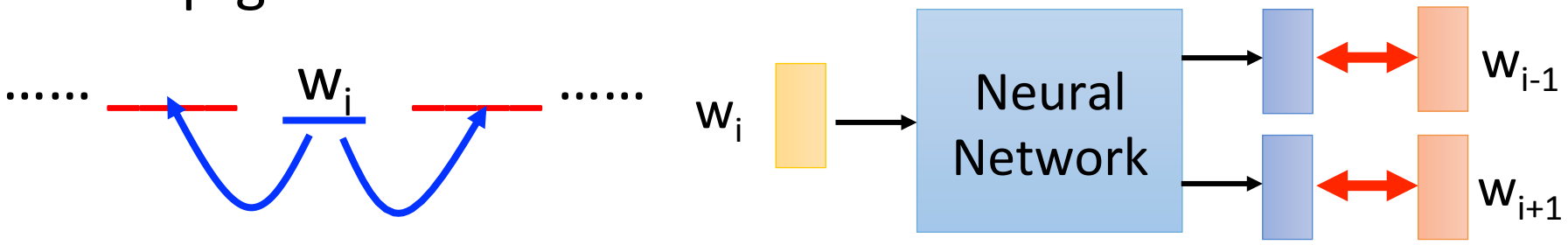
Prediction-based – Various Architectures

- Continuous bag of word (CBOW) model



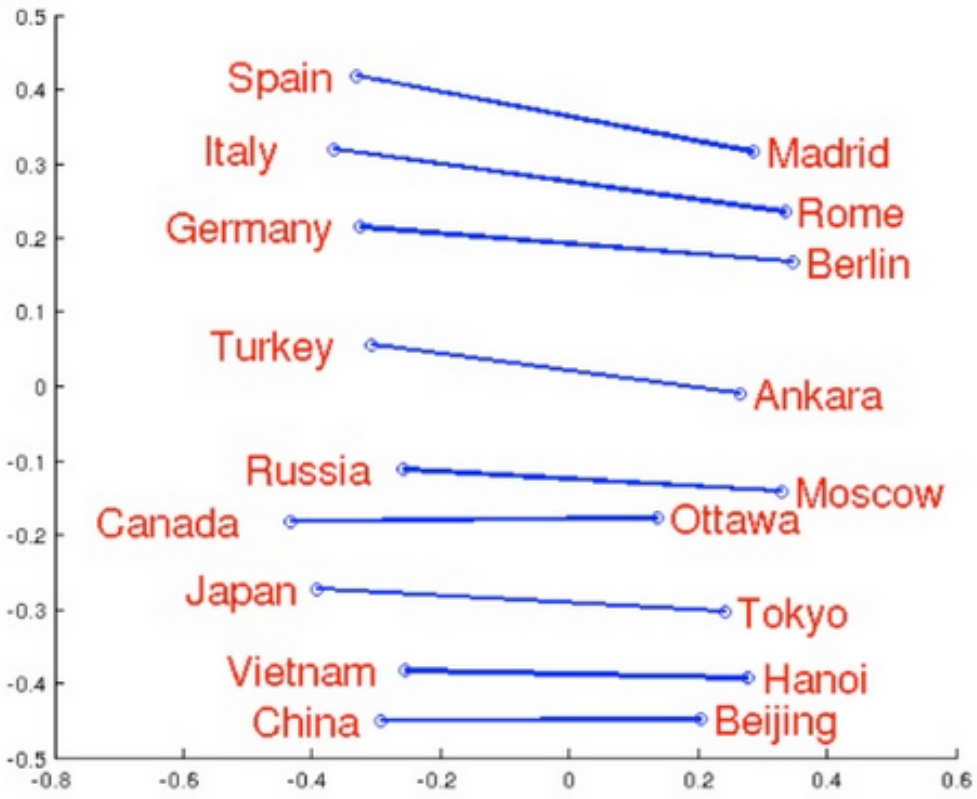
predicting the word given its context

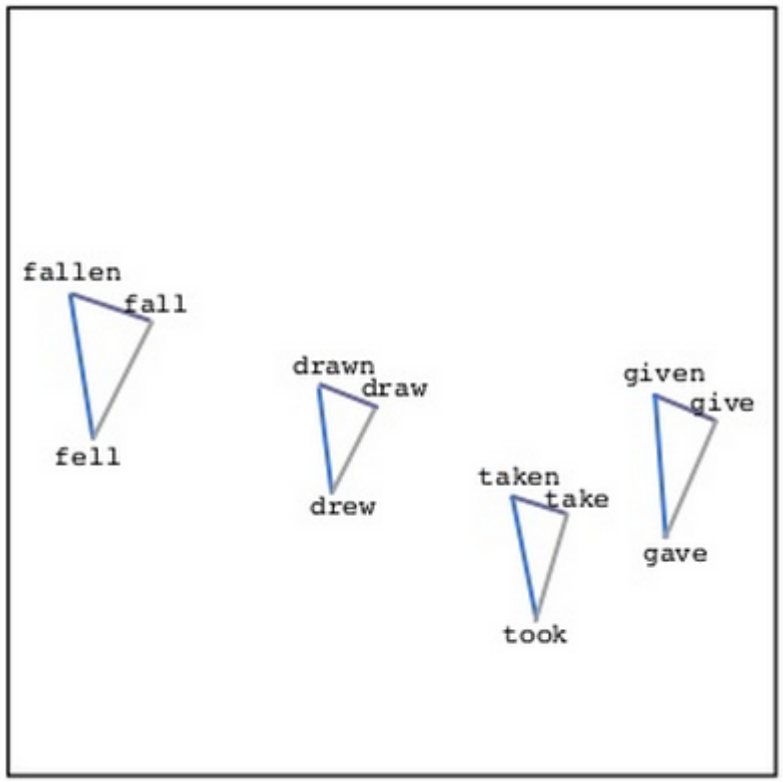
- Skip-gram



predicting the context given a word

Word Embedding





Source: http://www.slideshare.net/hustwj/cikm-keynotenov2014

Word Embedding

$$V(Germany) \approx V(Berlin) - V(Rome) + V(Italy)$$

- Characteristics

$$V(hotter) - V(hot) \approx V(bigger) - V(big)$$

$$V(Rome) - V(Italy) \approx V(Berlin) - V(Germany)$$

$$V(king) - V(queen) \approx V(uncle) - V(aunt)$$

- Solving analogies

Rome : Italy = Berlin : ?

Compute $V(Berlin) - V(Rome) + V(Italy)$

Find the word w with the closest V(w)

Document to Vector

- Paragraph Vector: Le, Quoc, and Tomas Mikolov. "Distributed Representations of Sentences and Documents.“ ICML, 2014
- Seq2seq Auto-encoder: Li, Jiwei, Minh-Thang Luong, and Dan Jurafsky. "A hierarchical neural autoencoder for paragraphs and documents." arXiv preprint, 2015
- Skip Thought: Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, Sanja Fidler, “Skip-Thought Vectors” arXiv preprint, 2015.
- Exploiting other kind of labels:
  - Huang, Po-Sen, et al. "Learning deep structured semantic models for web search using clickthrough data." ACM, 2013.
  - Shen, Yelong, et al. "A latent semantic model with convolutional-pooling structure for information retrieval." ACM, 2014.
  - Socher, Richard, et al. "Recursive deep models for semantic compositionality over a sentiment treebank." EMNLP, 2013.
  - Tai, Kai Sheng, Richard Socher, and Christopher D. Manning. "Improved semantic representations from tree-structured long short-term memory networks." arXiv preprint, 2015.

Audio Word to Vector

Machine does not have any prior knowledge

Machine listens to lots of audio book

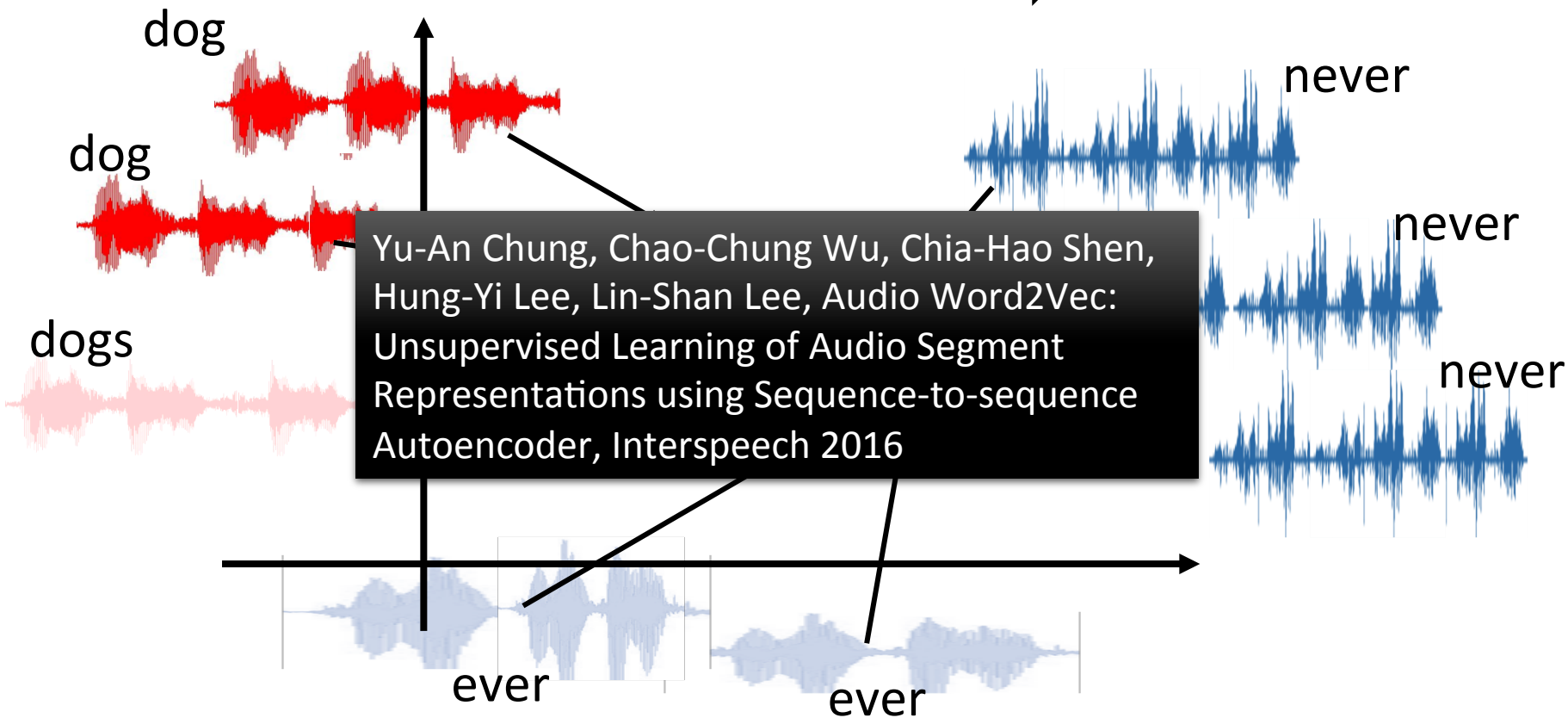
Like an infant

[Chung, Interspeech 16)

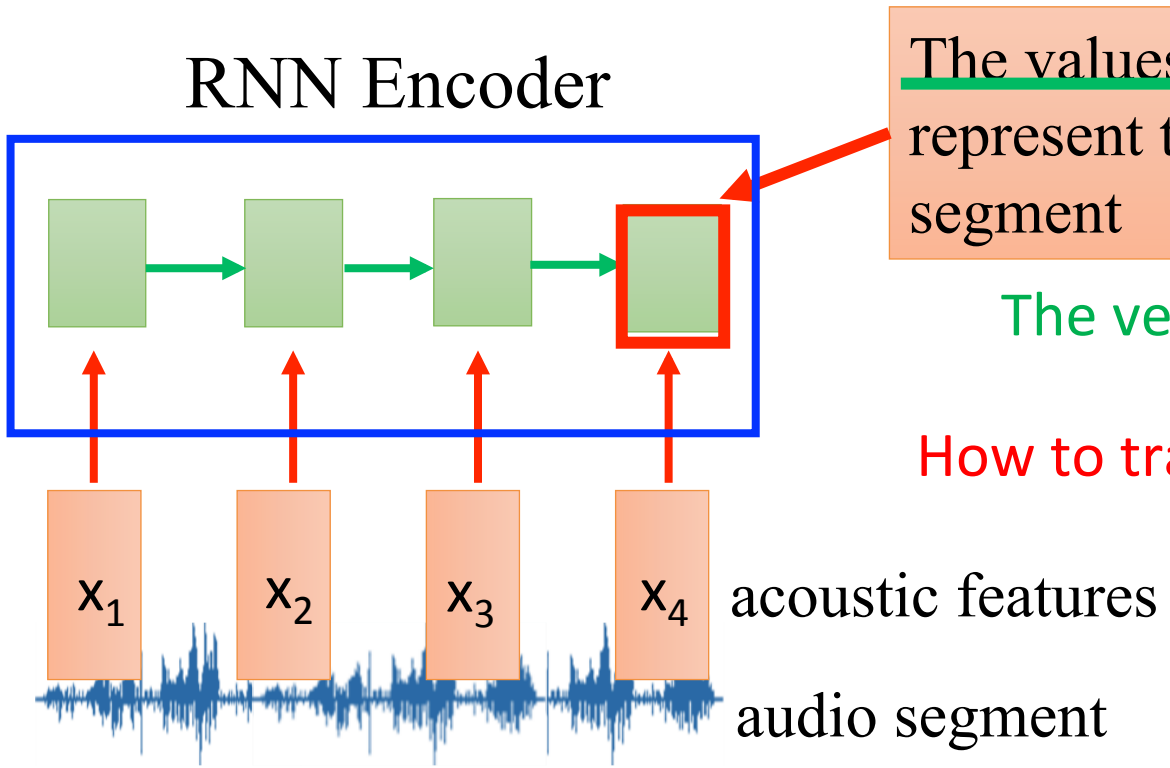
Audio Word to Vector

- Dimension reduction for a sequence with variable length

audio segments (word-level) → Fixed-length vector



Sequence-to-sequence Auto-encoder



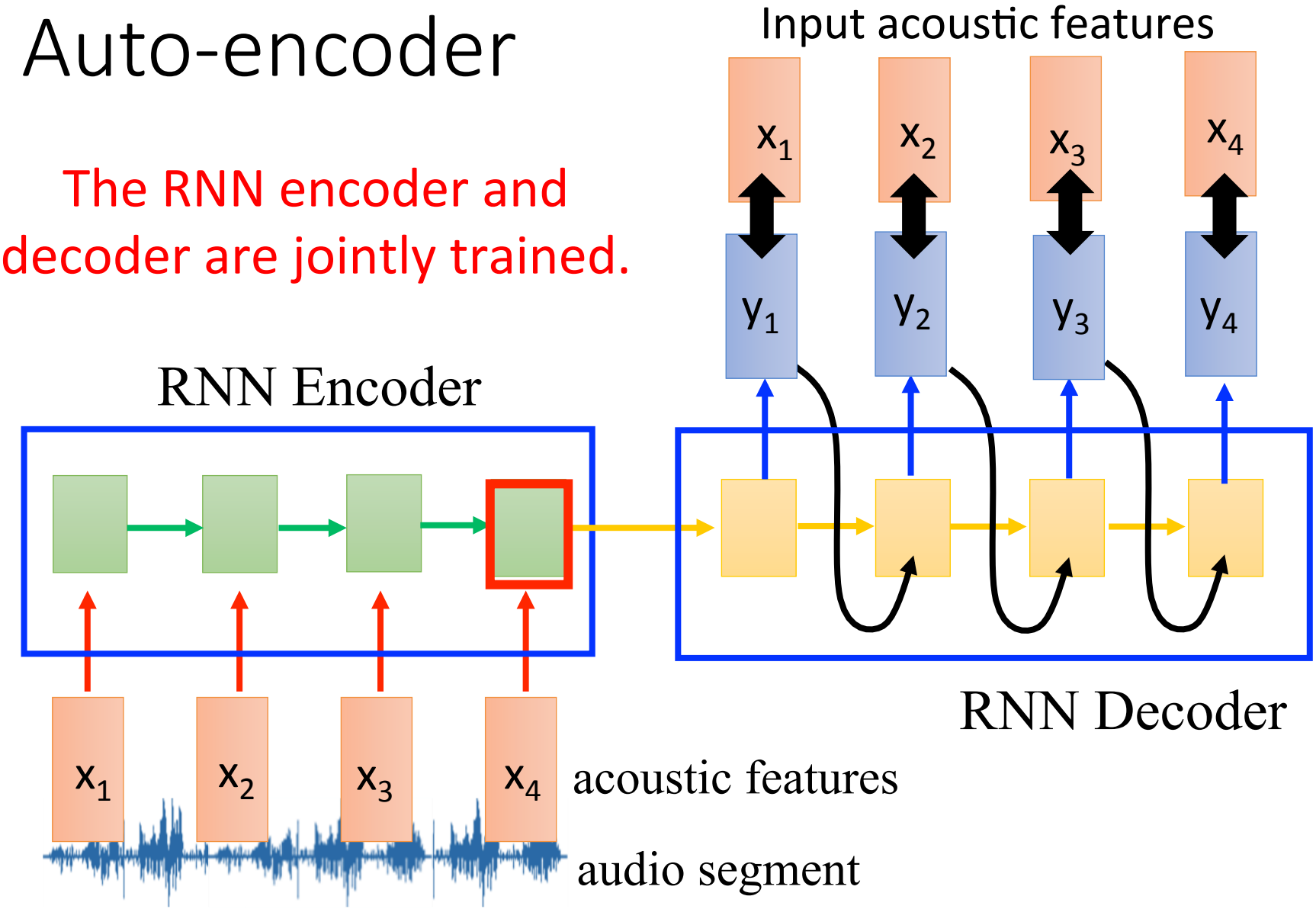
The values in the memory represent the whole audio segment

The vector we want

How to train RNN Encoder?

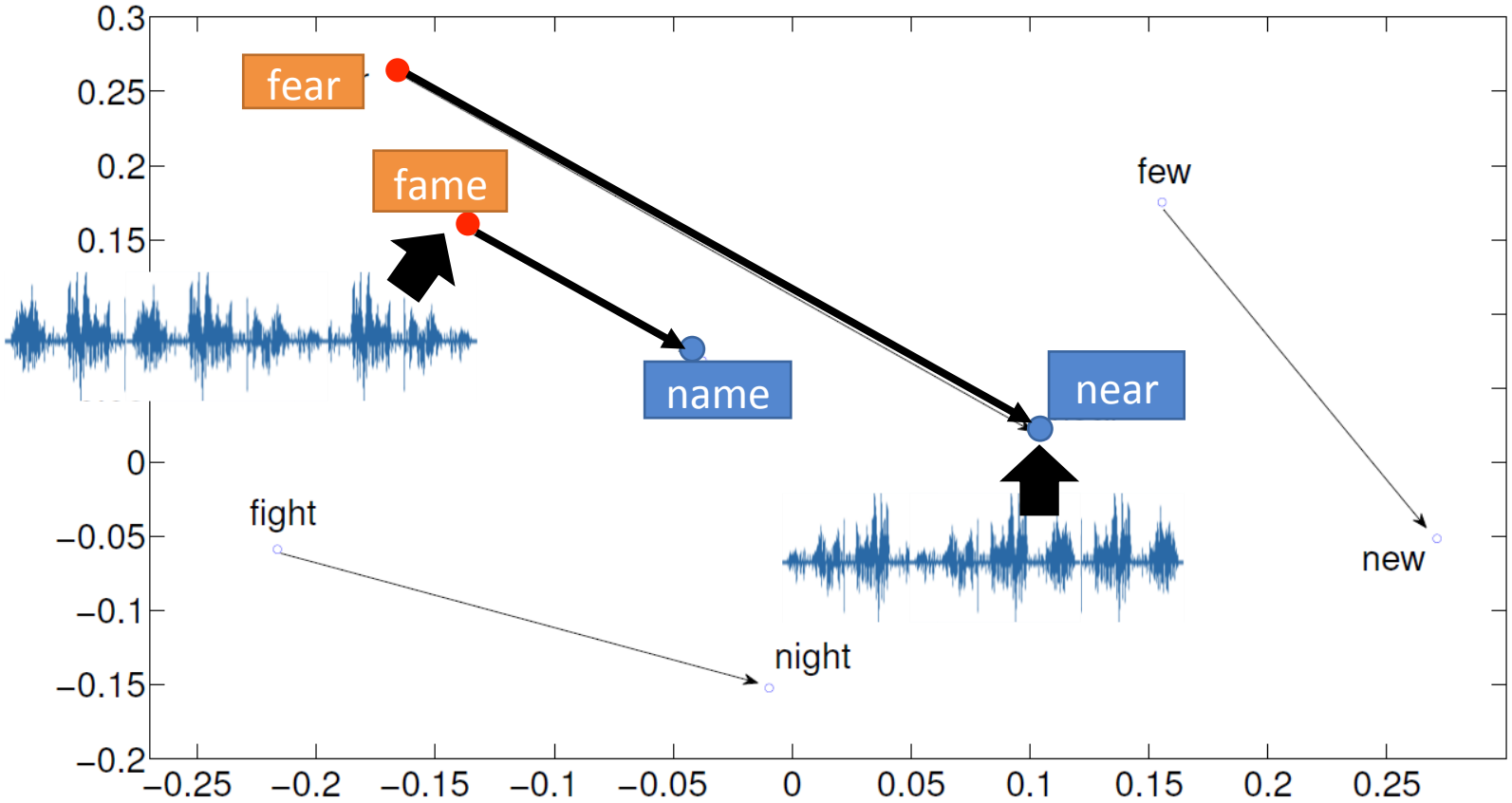
Sequence-to-sequence Auto-encoder

The RNN encoder and decoder are jointly trained.

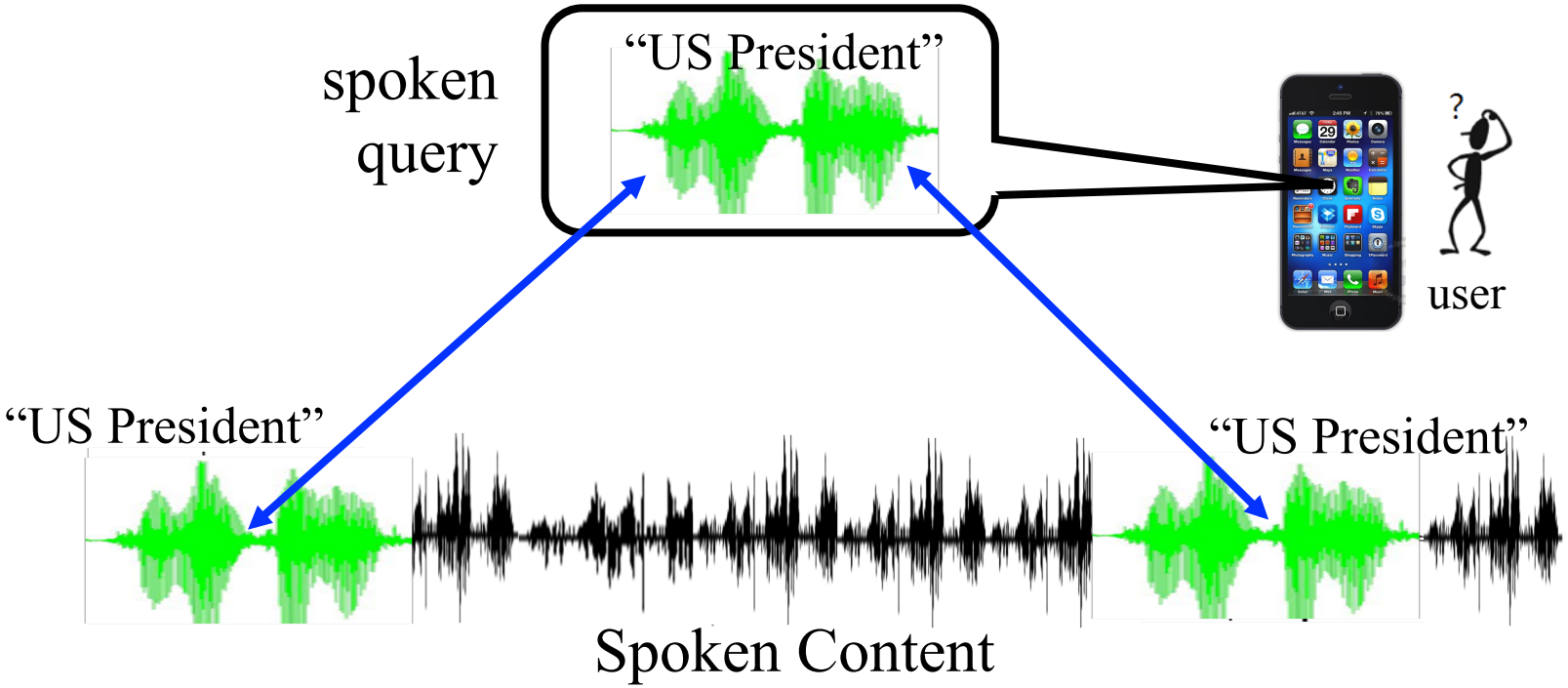


Sequence-to-sequence Auto-encoder

- Visualizing embedding vectors of the words

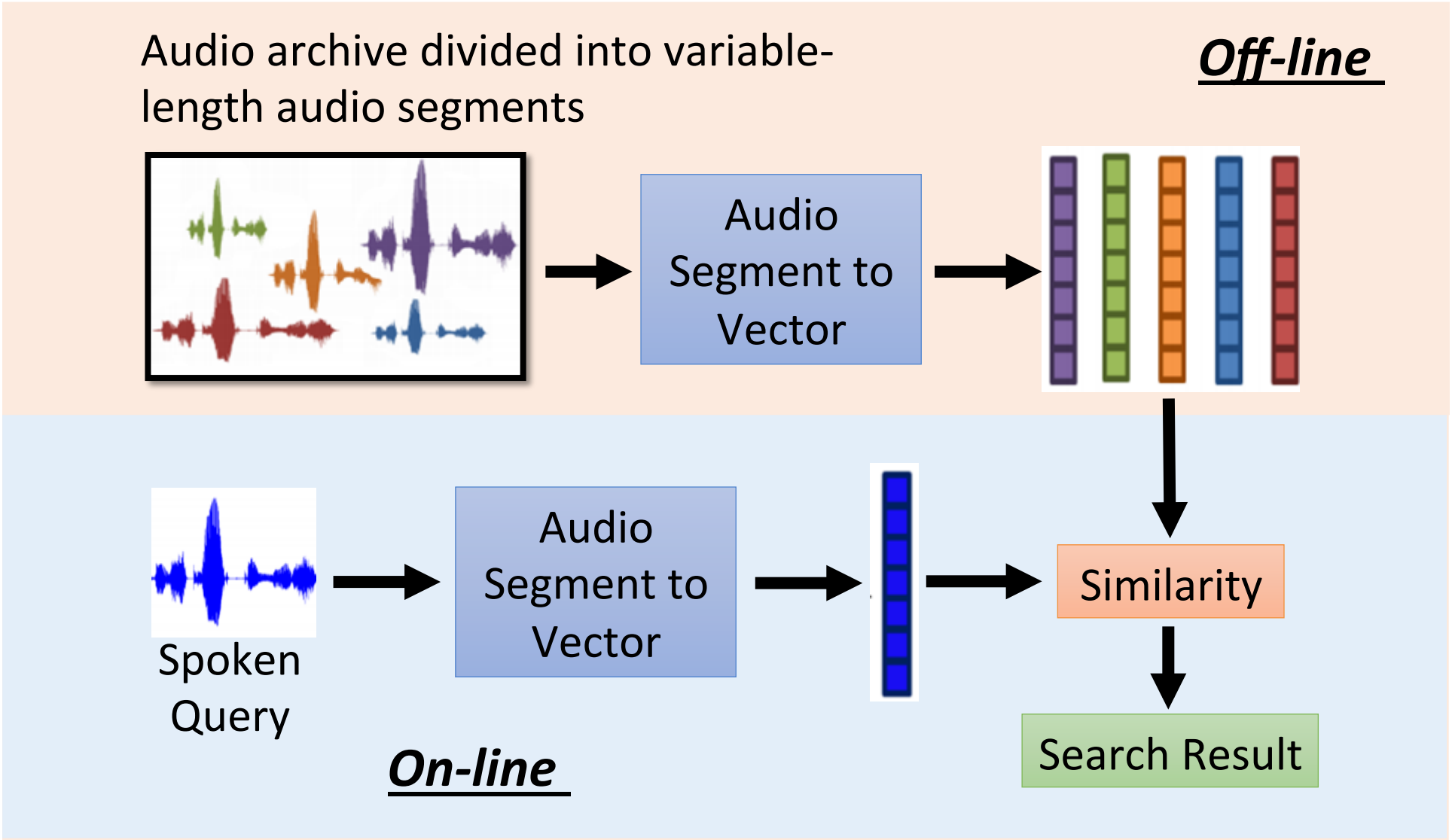


Audio Word to Vector – Application



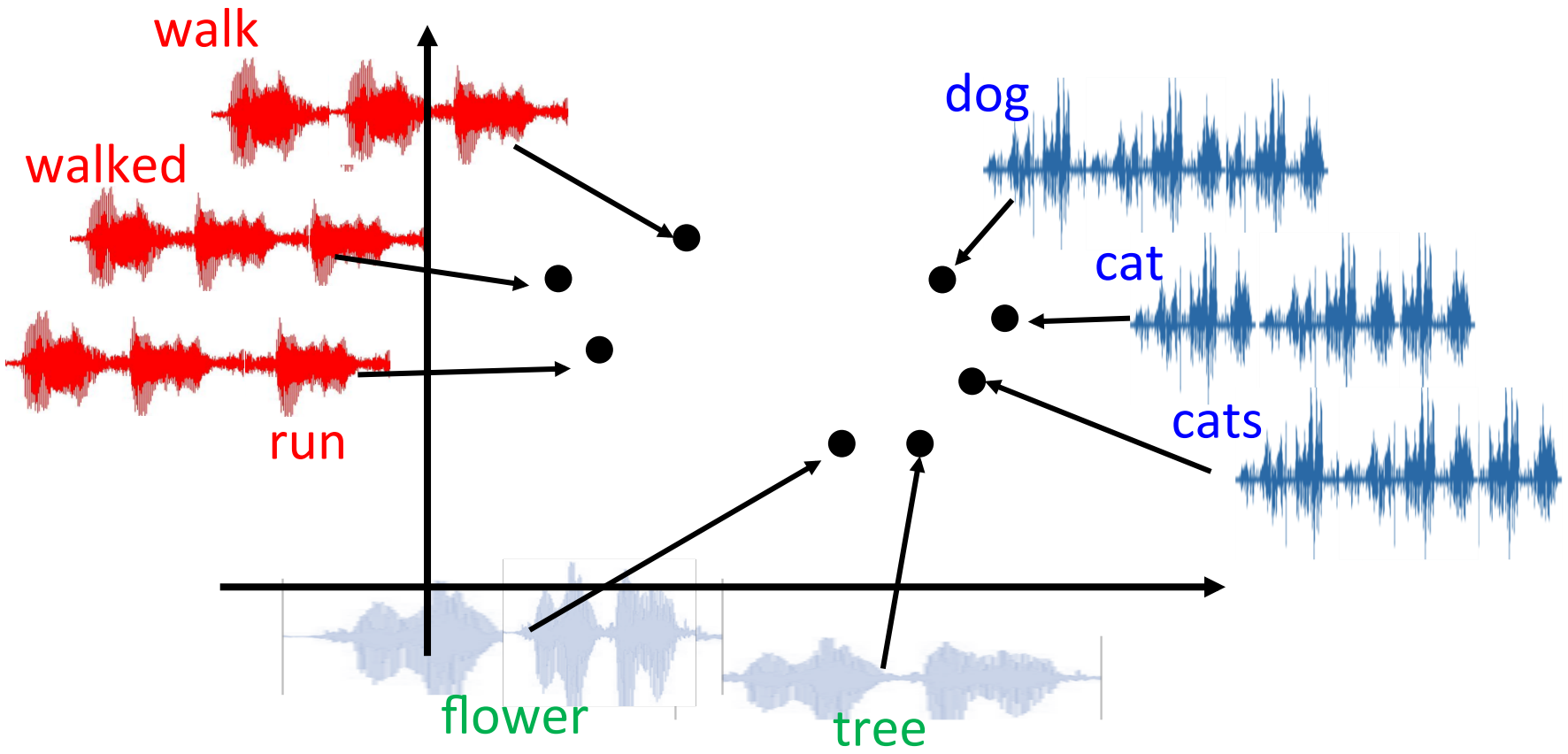
Compute similarity between spoken queries and audio files on acoustic level, and find the query term

Audio Word to Vector – Application



Next Step ……

- Can we include semantics?



Outline

Unsupervised Learning

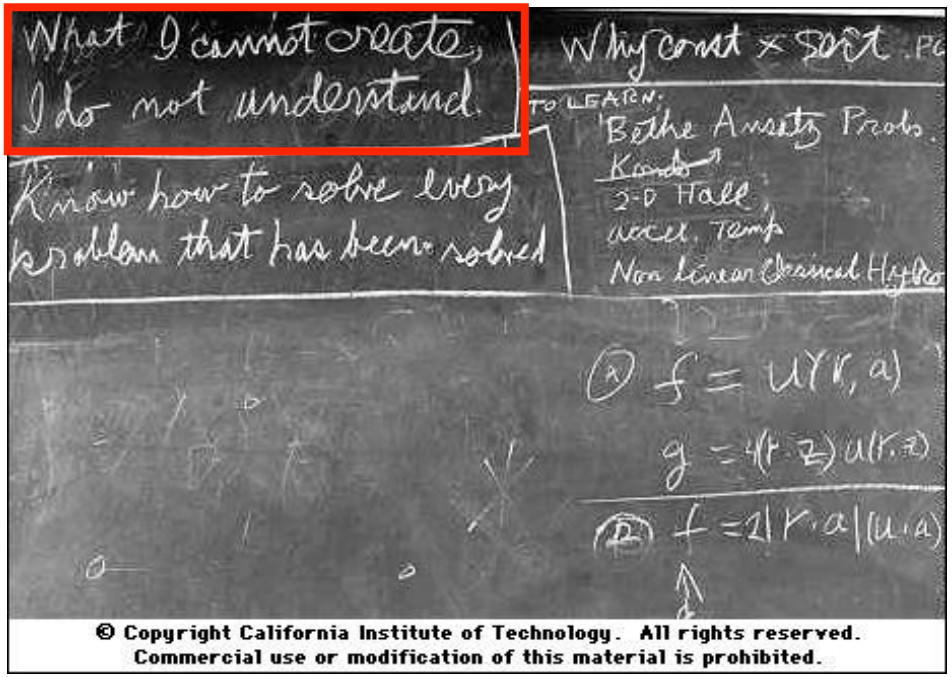
- Abstraction
  - Auto-encoder
  - Word Vector and Audio Word Vector
- Out of nothing

Reinforcement Learning

Creation

Creation

- Generative Models: https://openai.com/blog/generative-models/



What I cannot create, I do not understand.

Richard Feynman

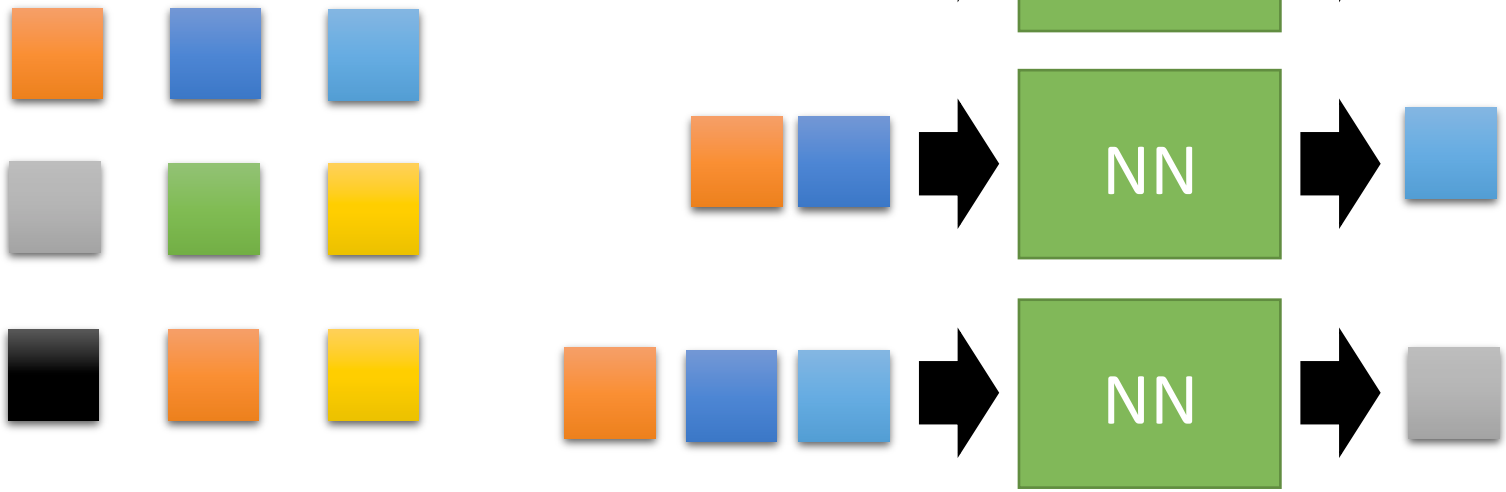
https://www.quora.com/What-did-Richard-Feynman-mean-when-he-said-What-I-cannot-create-I-do-not-understand

PixelRNN

Ref: Aaron van den Oord, Nal Kalchbrenner, Koray Kavukcuoglu, Pixel Recurrent Neural Networks, arXiv preprint, 2016

- To create an image, generating a pixel each time

E.g. 3 x 3 images



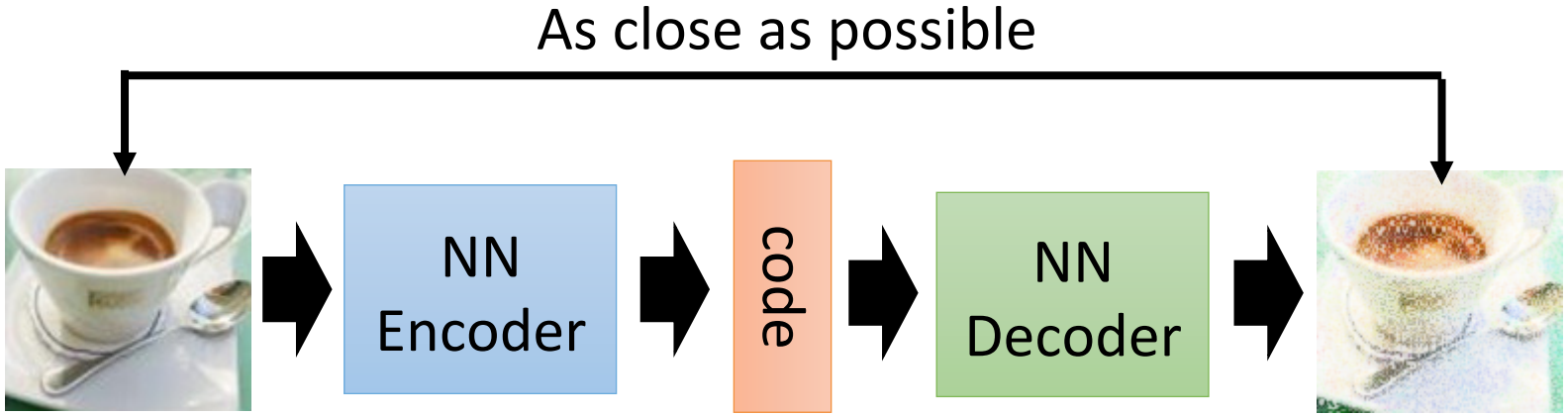
Can be trained just with a large collection of images without any annotation

PixelRNN

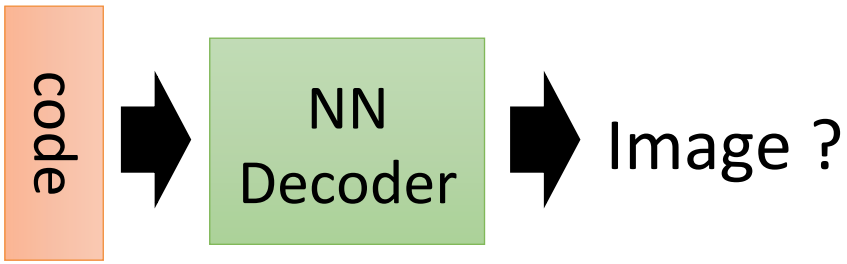
Ref: Aaron van den Oord, Nal Kalchbrenner, Koray Kavukcuoglu, Pixel Recurrent Neural Networks, arXiv preprint, 2016

Real World

Auto-encoder



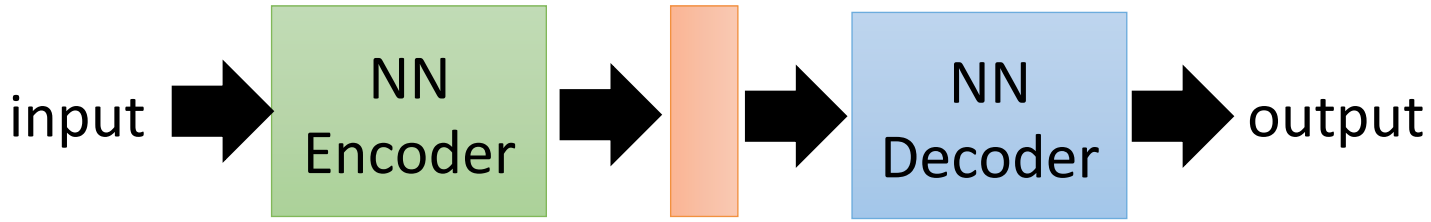
Randomly generate a vector as code



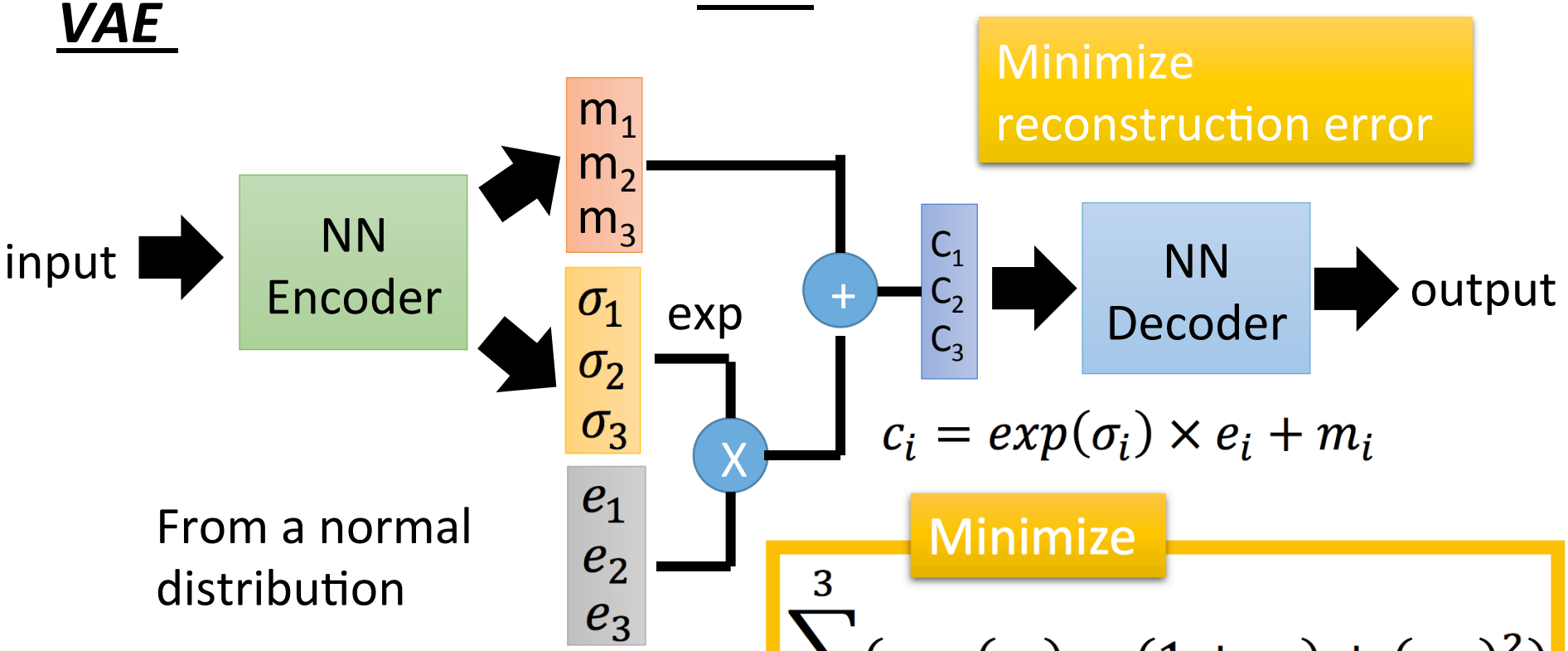
Variation Auto-encoder (VAE)

Ref: Auto-Encoding Variational Bayes, https://arxiv.org/abs/1312.6114

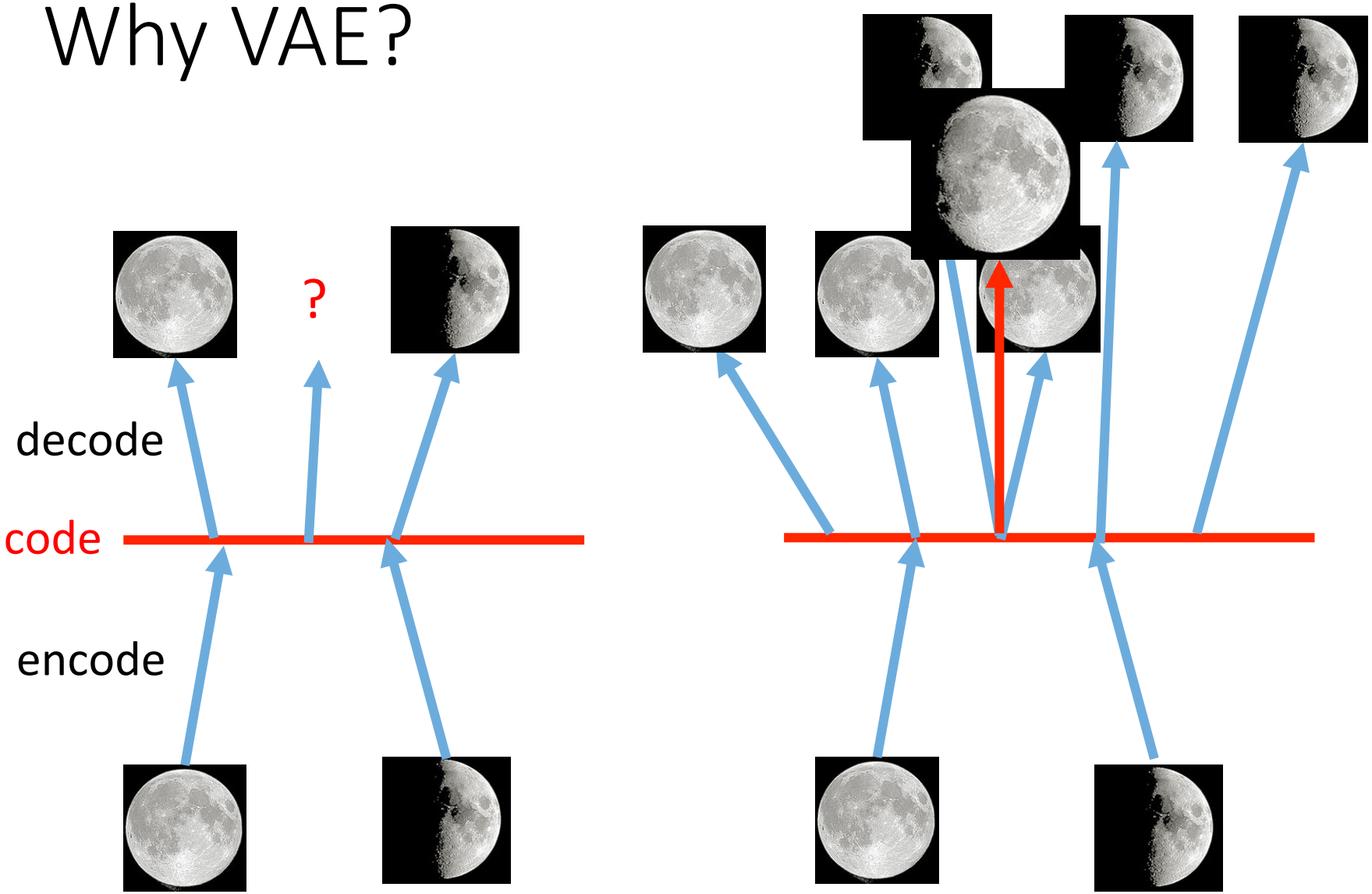
Auto-encoder



VAE



Why VAE?



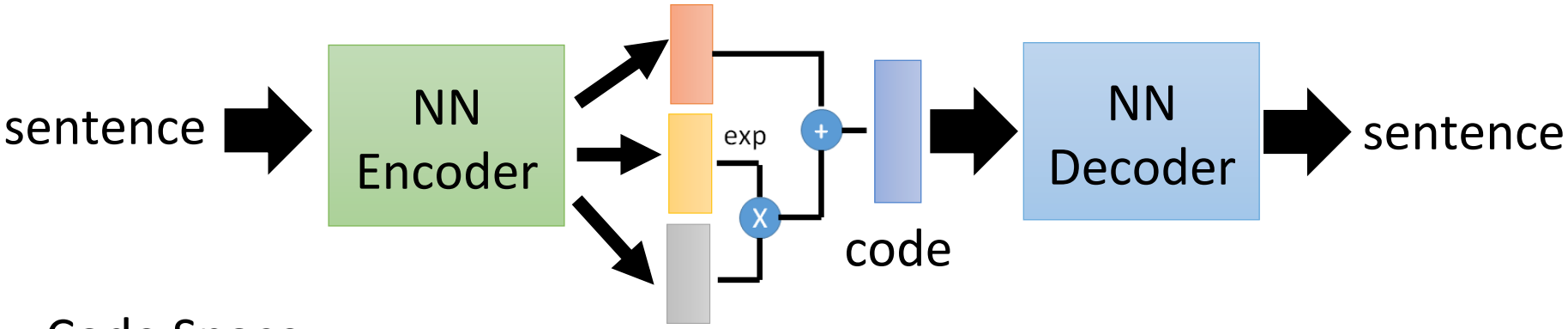
VAE

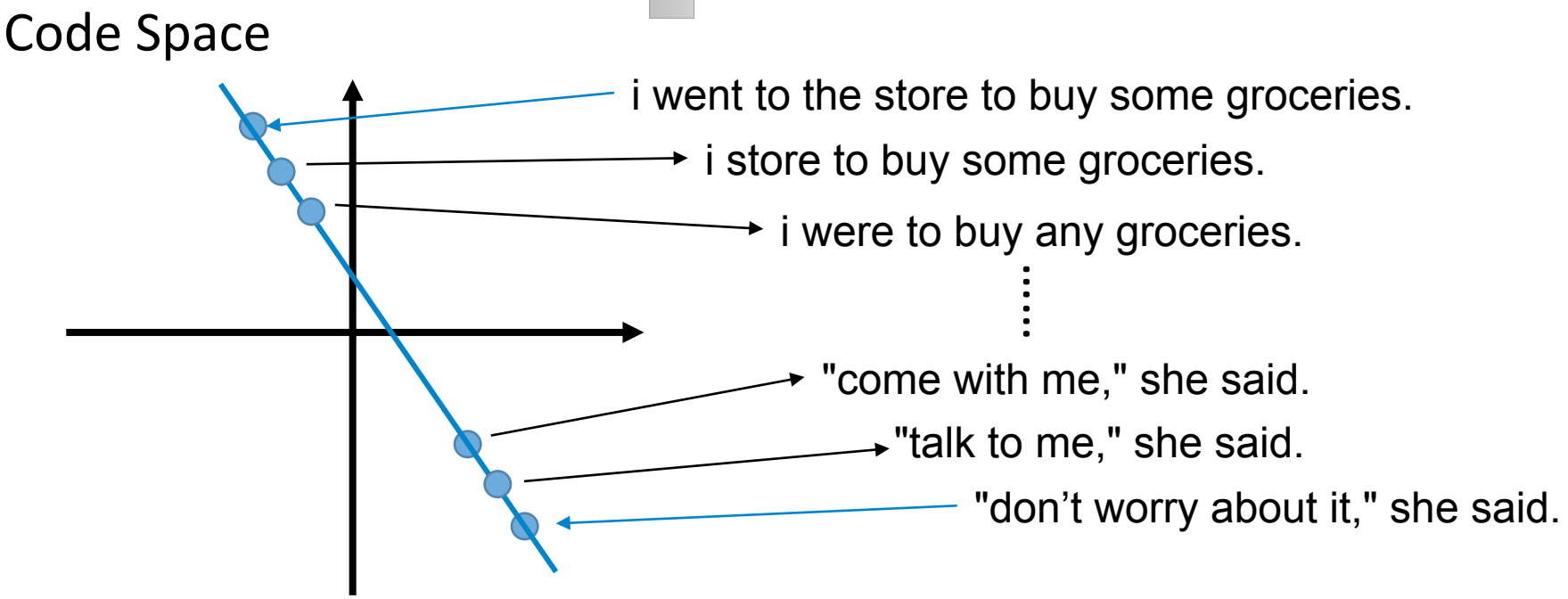
Cifar-10

https://github.com/openai/iaf

Source of image: https://arxiv.org/pdf/1606.04934v1.pdf

VAE - Writing Poetry



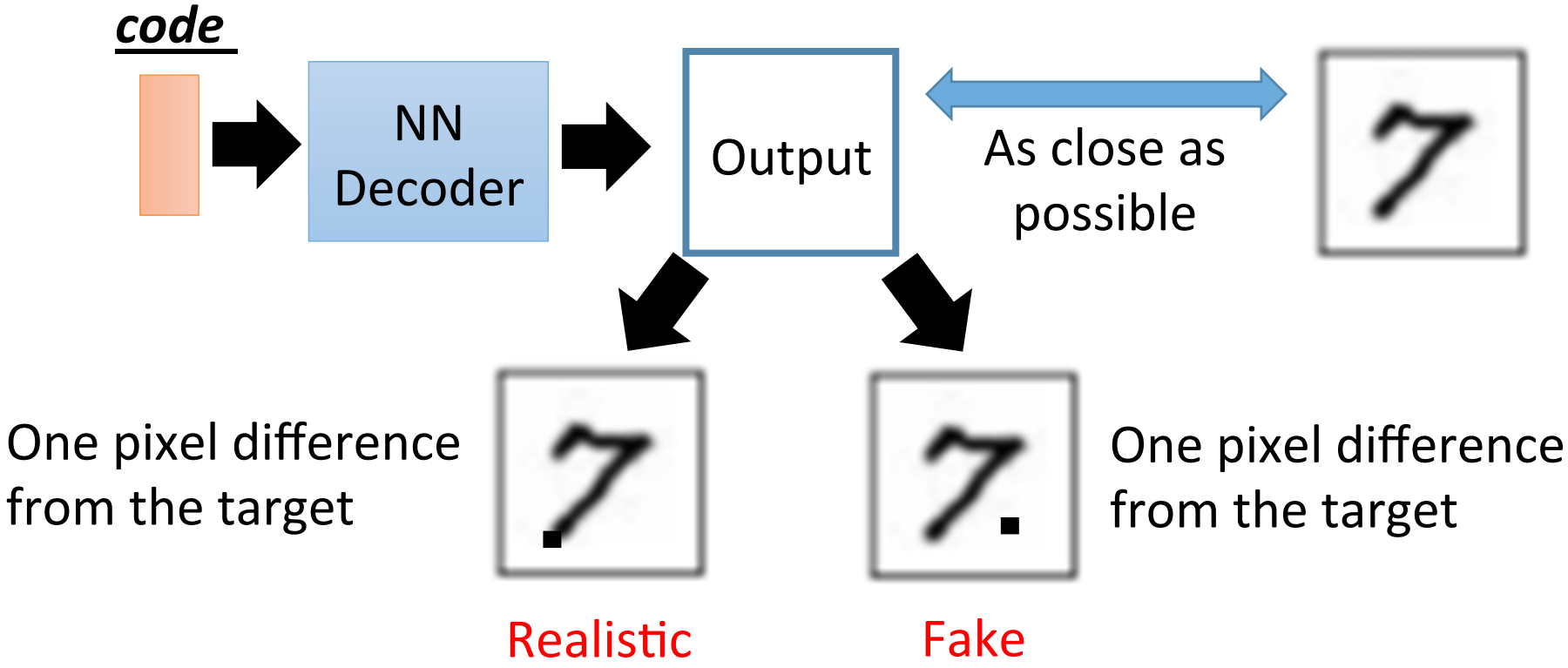


Ref: http://www.wired.co.uk/article/google-artificial-intelligence-poetry

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Jozefowicz, Samy Bengio, Generating Sentences from a Continuous Space, arXiv prepring, 2015

Problems of VAE

- It does not really try to simulate real images



Generative Adversarial Network (GAN)

What are some recent and potentially upcoming breakthroughs in unsupervised learning?

Yann LeCun, Director of AI Research at Facebook and Professor at NYU
Written Jul 29 · Upvoted by Joaquin Quiñonero Candela, Director Applied Machine Learning at Facebook and Huang Xiao

Adversarial training is the coolest thing since sliced bread.

I've listed a bunch of relevant papers in a previous answer.

Expect more impressive results with this technique in the coming years.

What's missing at the moment is a good understanding of it so we can make it work reliably. It's very finicky. Sort of like ConvNet were in the 1990s, when I had the reputation of being the only person who could make them work (which wasn't true).

Ref: Generative Adversarial Networks, http://arxiv.org/abs/1406.2661

擬態的演化

http://peellden.pixnet.net/blog/post/40406899-2013-%E7%AC%AC%E5%9B%9B%E5%AD%A3%EF%BC%8C%E5%86%AC%E8%9D%B6%E5%AF%82%E5%AF%A5

棕色

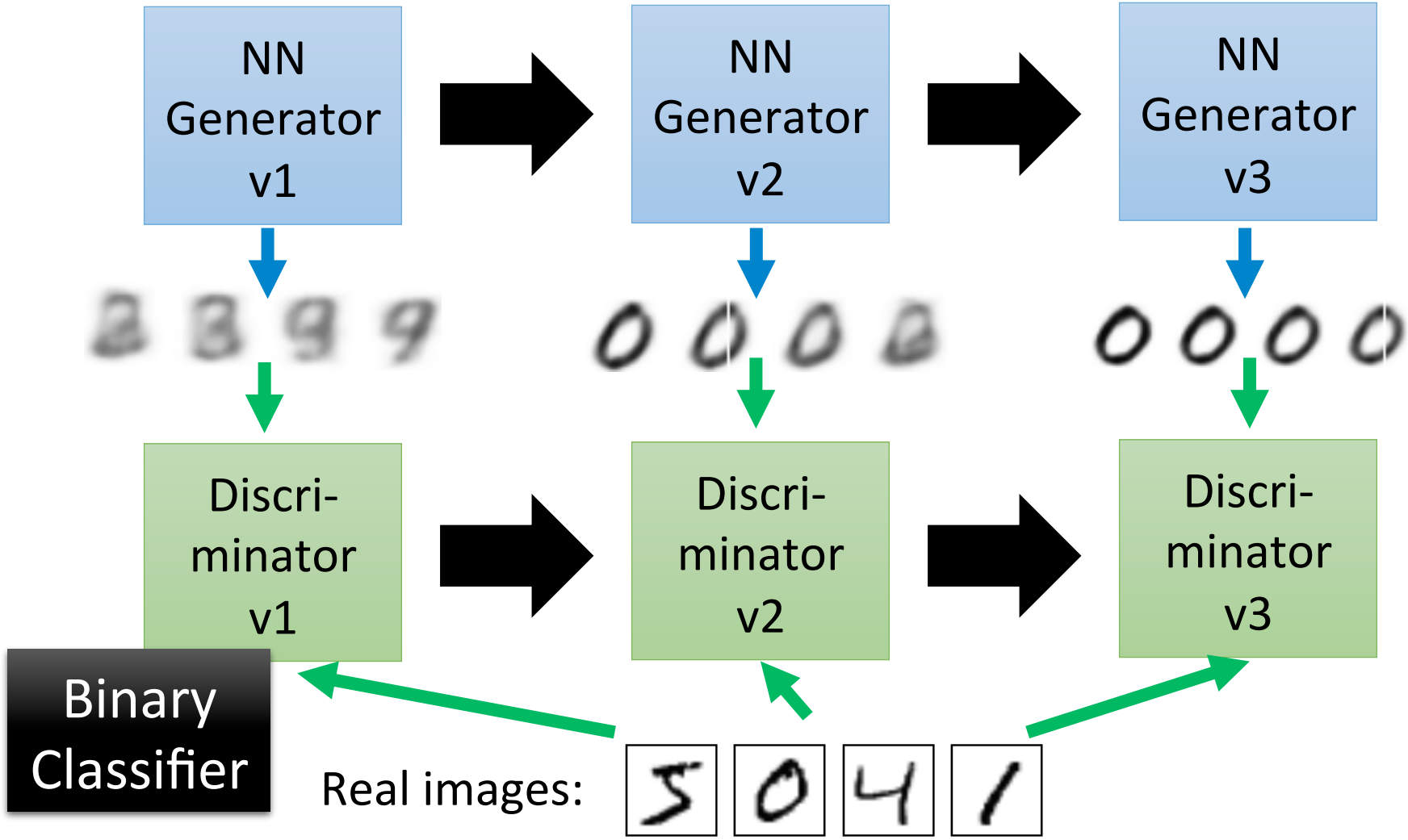
葉脈

蝴蝶不是棕色

蝴蝶沒有葉脈

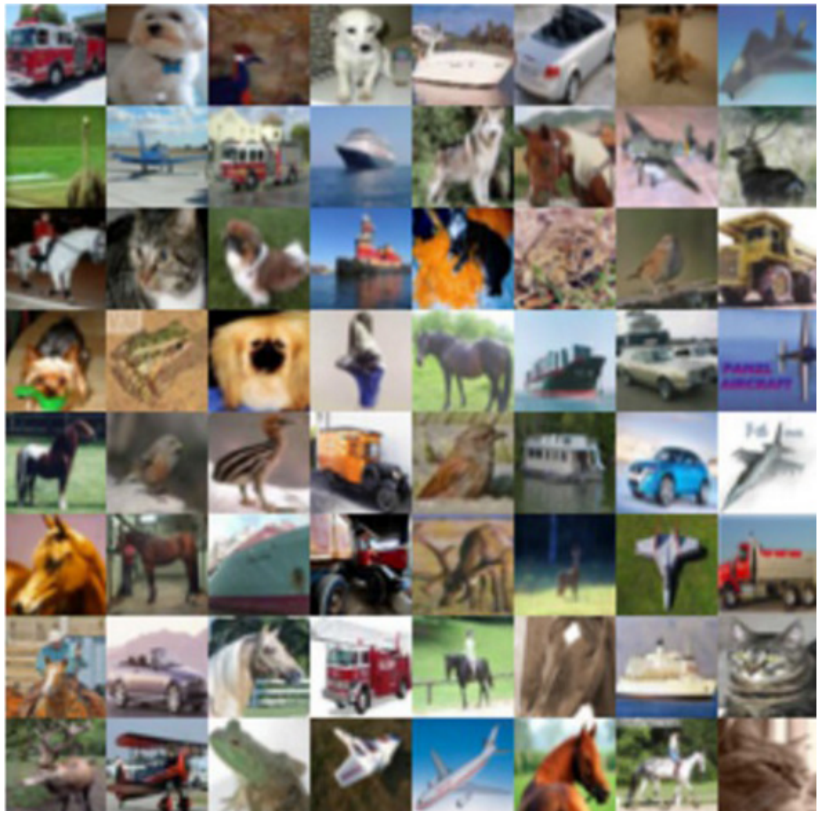
.....

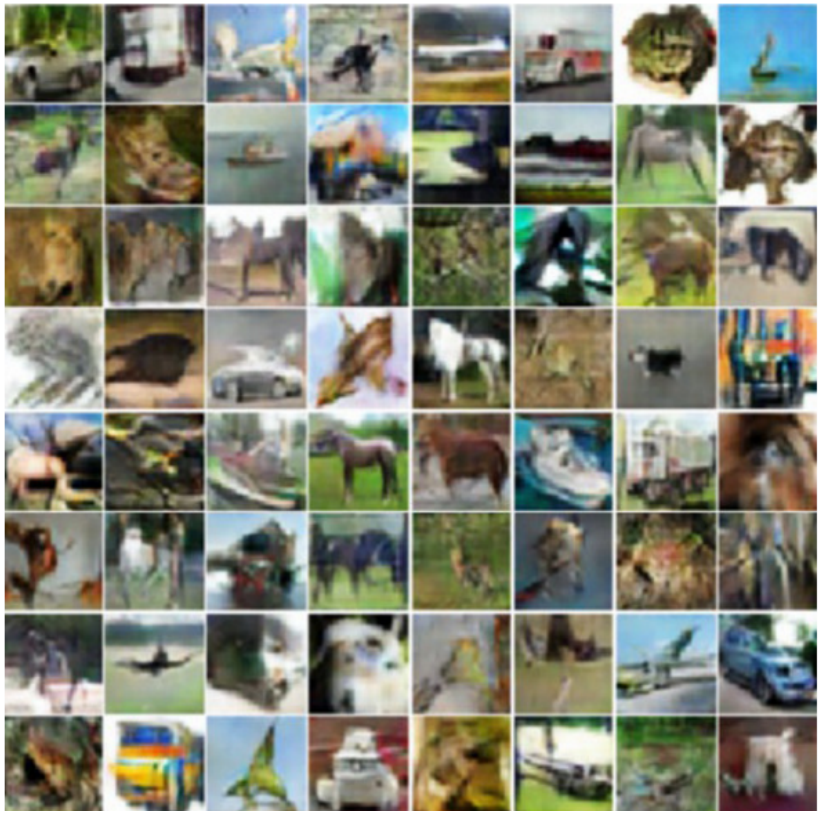
The evolution of generation



Cifar-10

- Which one is machine-generated?





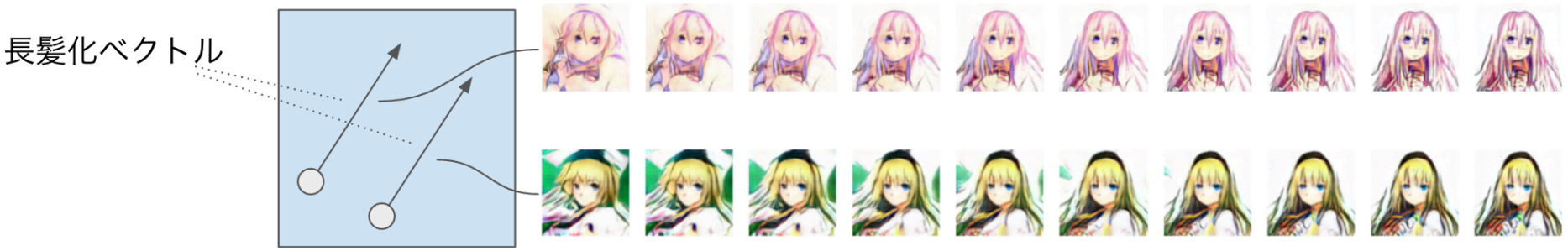
Ref: https://openai.com/blog/generative-models/

畫漫畫

- **Ref:** https://github.com/mattya/chainer-DCGAN

畫漫畫

- Ref: http://qiita.com/mattya/items/e5bfe5e04b9d2f0bbd47



一番左のキャラクターが元画像で、
右に行くほど長髪化ベクトルを強く足している

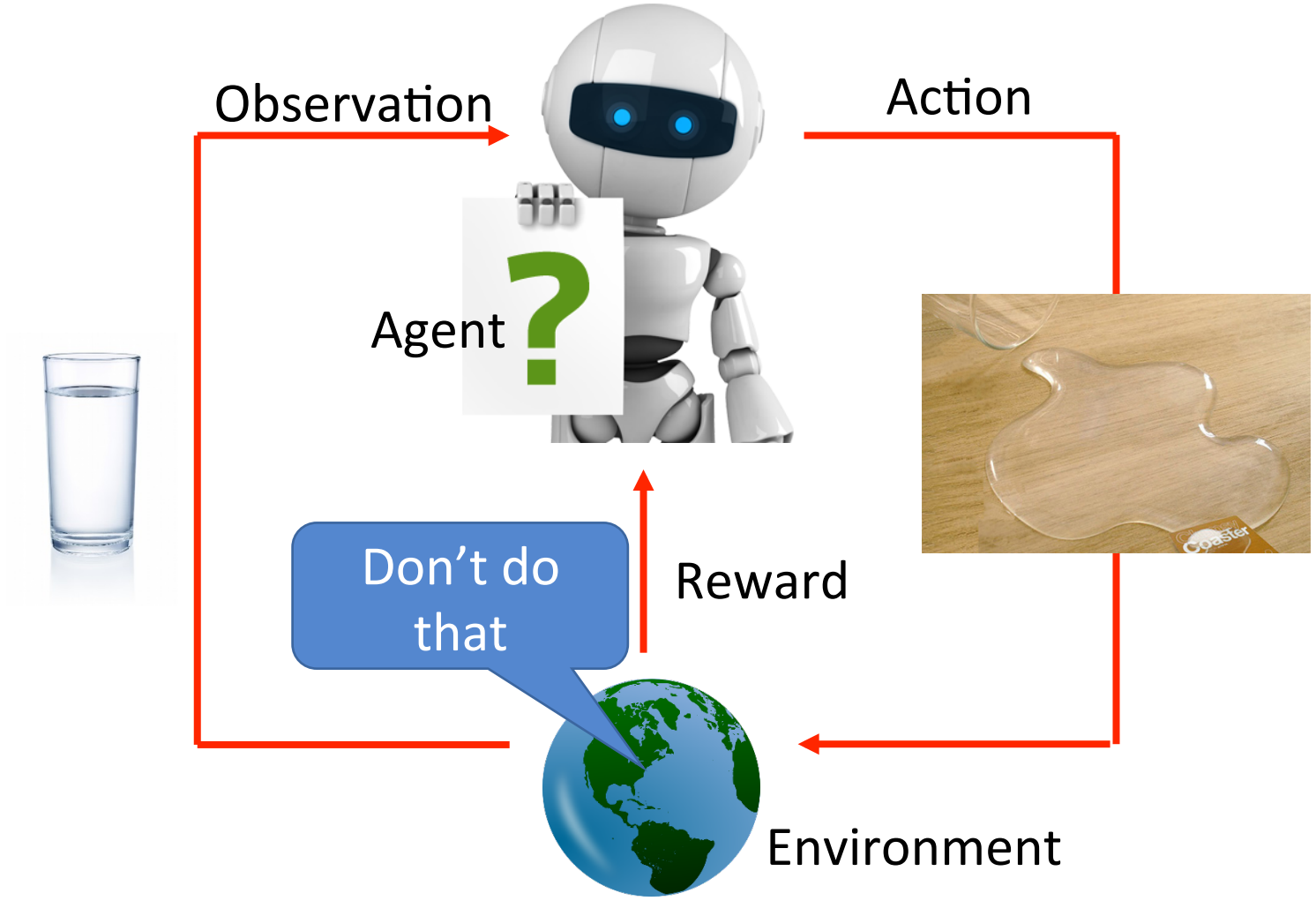
Outline

Unsupervised Learning

- Abstraction
  - Example: Word Vector and Audio Word Vector
- Out of nothing

Reinforcement Learning

Scenario of Reinforcement Learning



Scenario of Reinforcement Learning

http://www.sznews.com/news/content/2013-11/26/content_8800180.htm

Supervised v.s. Reinforcement

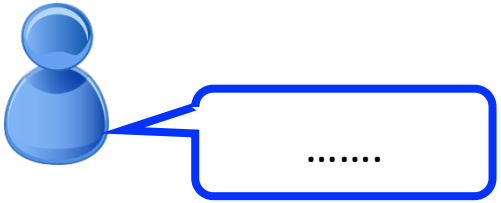
- Supervised

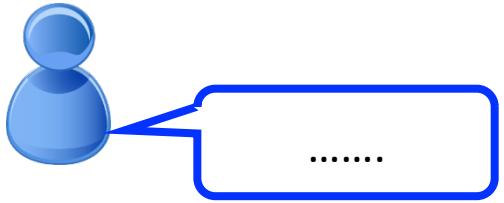
Learning from teacher

Say "Hi"

Say "Good bye"

- Reinforcement

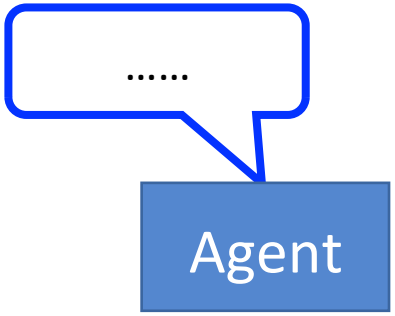




.....

Bad

Learning from critics



Scenario of Reinforcement Learning

Supervised v.s. Reinforcement

- Supervised:

Next move: "5-5"

Next move: "3-3"

- Reinforcement Learning

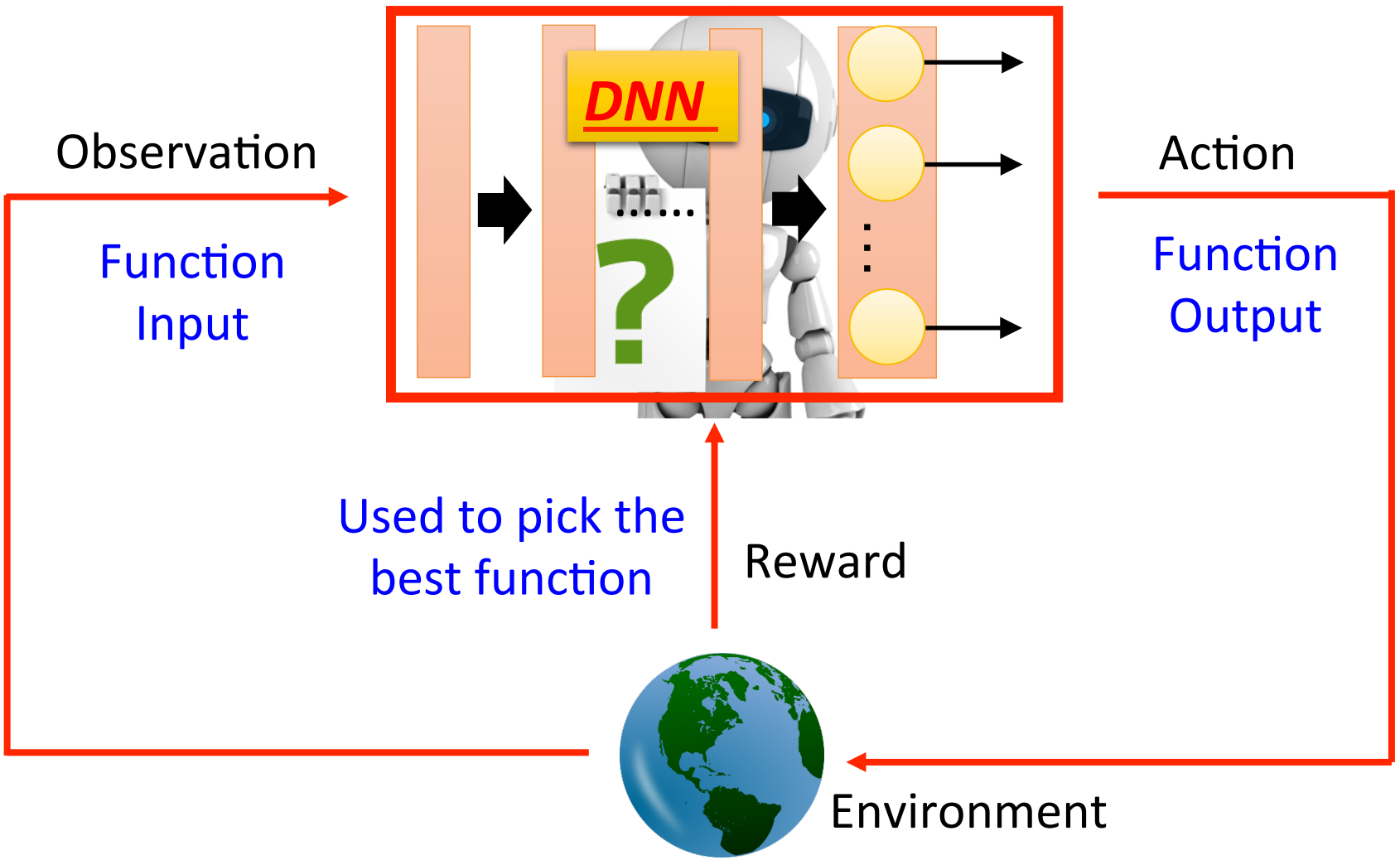
First move → …… many moves …… → Win!

Alpha Go is supervised learning + reinforcement learning.

Difficulties of Reinforcement Learning

- It may be better to sacrifice immediate reward to gain more long-term reward
  - E.g. Playing Go
- Agent's actions affect the subsequent data it receives
  - E.g. Exploration

Deep Reinforcement Learning



More applications

- Alpha Go, Playing Video Games, Dialogue
- Flying Helicopter
  - https://www.youtube.com/watch?v=0JL04JJjocc
- Driving
  - https://www.youtube.com/watch?v=0xo1Ldx3L5Q
- Google Cuts Its Giant Electricity Bill With DeepMind-Powered AI
  - http://www.bloomberg.com/news/articles/2016-07-19/google-cuts-its-giant-electricity-bill-with-deepmind-powered-ai

To learn deep reinforcement learning

- Lectures of David Silver
  - http://www0.cs.ucl.ac.uk/staff/D.Silver/web/Teaching.html
  - 10 lectures (1:30 each)
- Deep Reinforcement Learning
  - http://videolectures.net/rldm2015_silver_reinforcement_learning/

The End