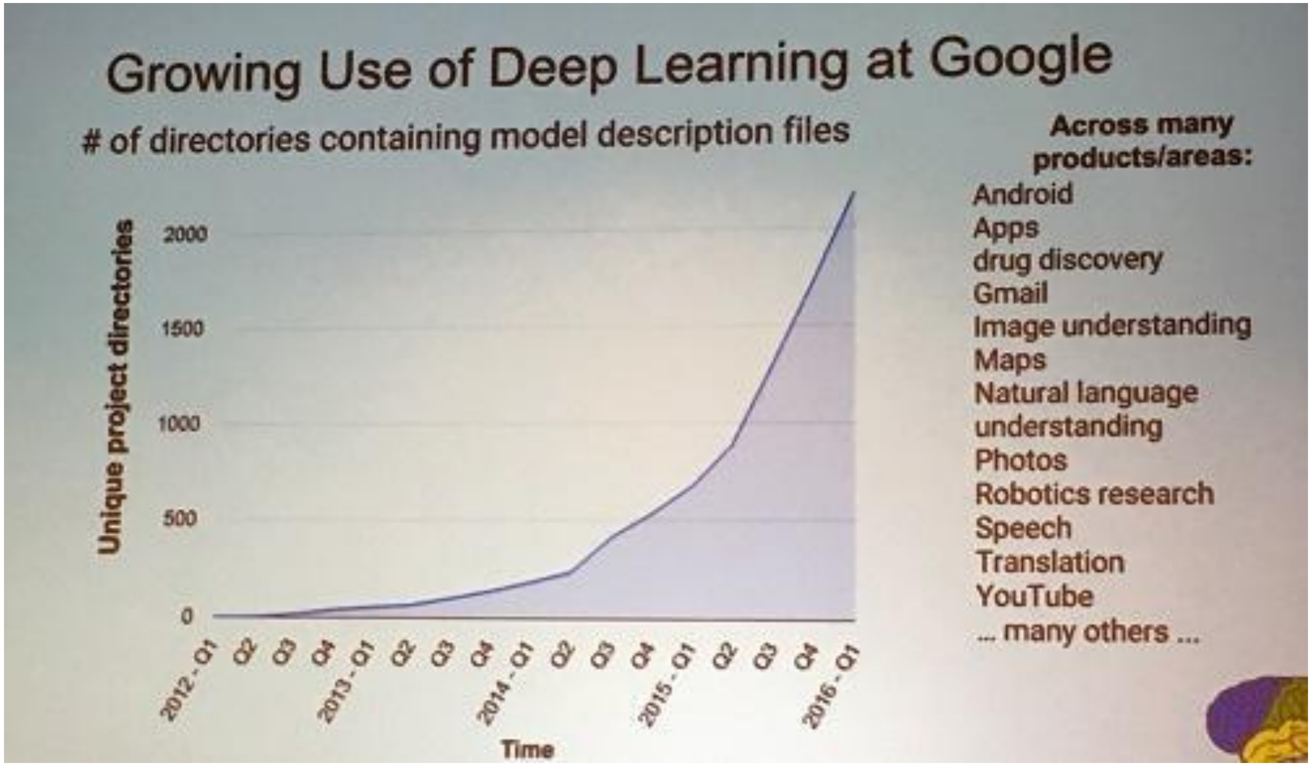
Deep Learning Tutorial

李宏毅

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 talk focuses on the basic techniques.

Outline

Lecture I: Introduction of Deep Learning

Lecture II: Variants of Neural Network

Lecture III: Beyond Supervised Learning

Lecture I:
Introduction of Deep Learning

Outline

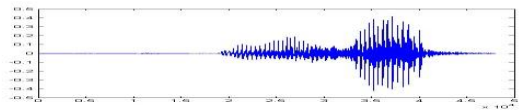
Introduction of Deep Learning

“Hello World” for Deep Learning

Tips for Deep Learning

Machine Learning ≈ Looking for a Function

- Speech Recognition

$f($  $) =$ "How are you"

- Image Recognition

$f($  $) =$ "Cat"

- Playing Go

$f($  $) =$ "5-5" (next move)

- Dialogue System

$f($ "Hi" $) =$ "Hello"

(what the user said) (system response)

Framework

Image Recognition:

$f($  $) =$ "cat"

A set of function

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

Model

$f_1, f_2 \cdots$

↓

Goodness of function f

↑

Training Data

$f_1($  $) =$ "cat"

$f_2($  $) =$ "money"

Better!

$f_1($  $) =$ "dog"

$f_2($  $) =$ "snake"

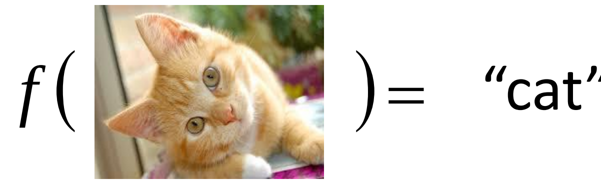
Supervised Learning

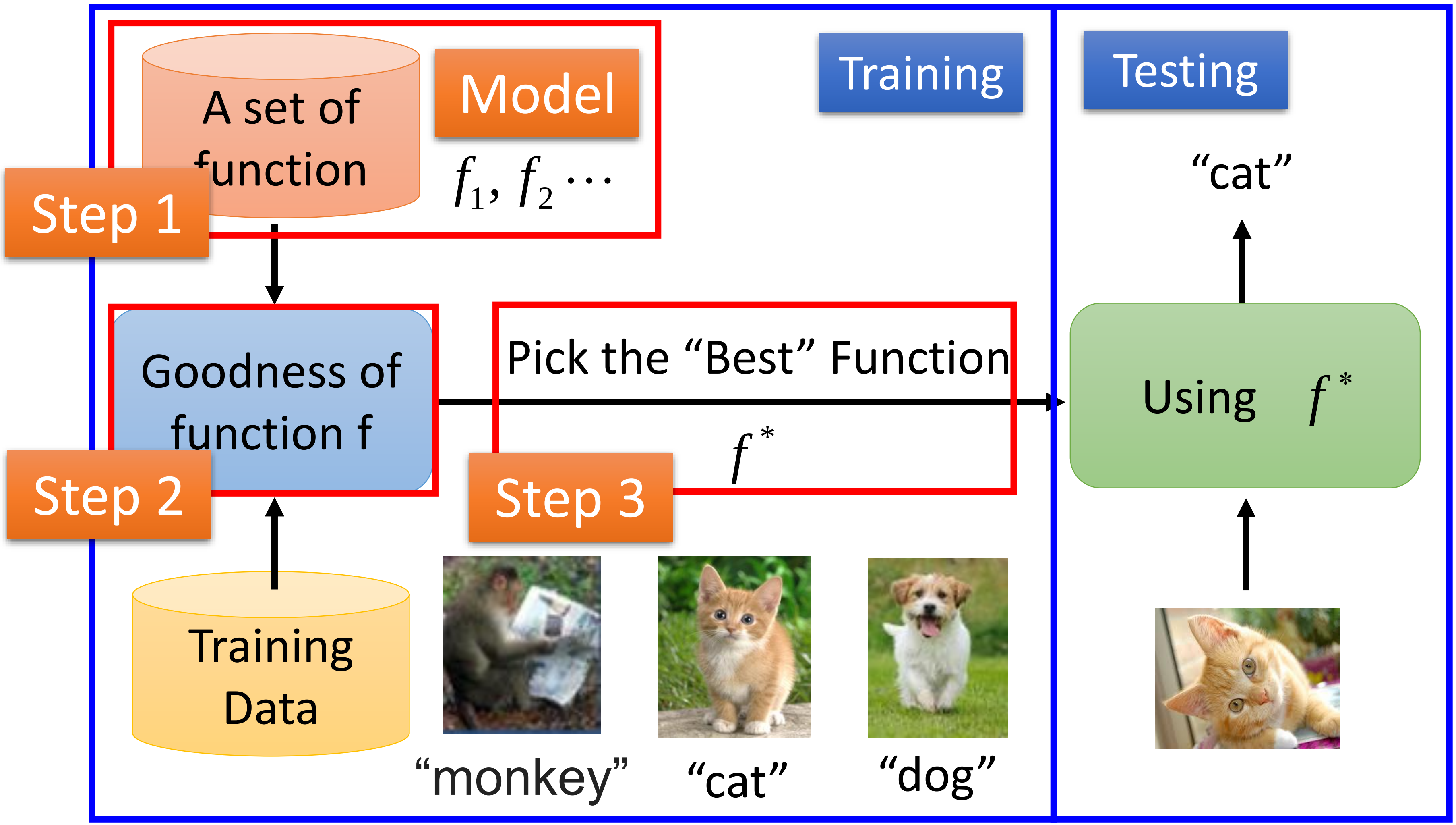
function input:   

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

Framework

Image Recognition:





Three Steps for Deep Learning

Step 1: define a set of function

Neural Network

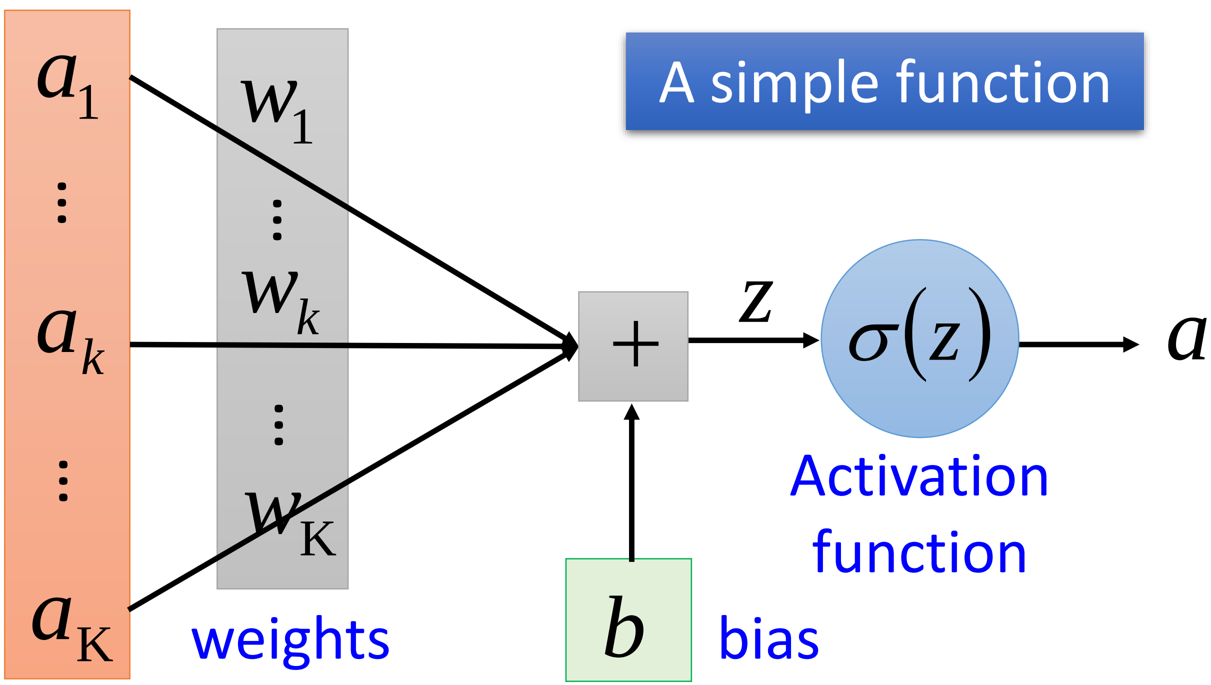
Step 2: goodness of function

Step 3: pick the best function

Neural Network

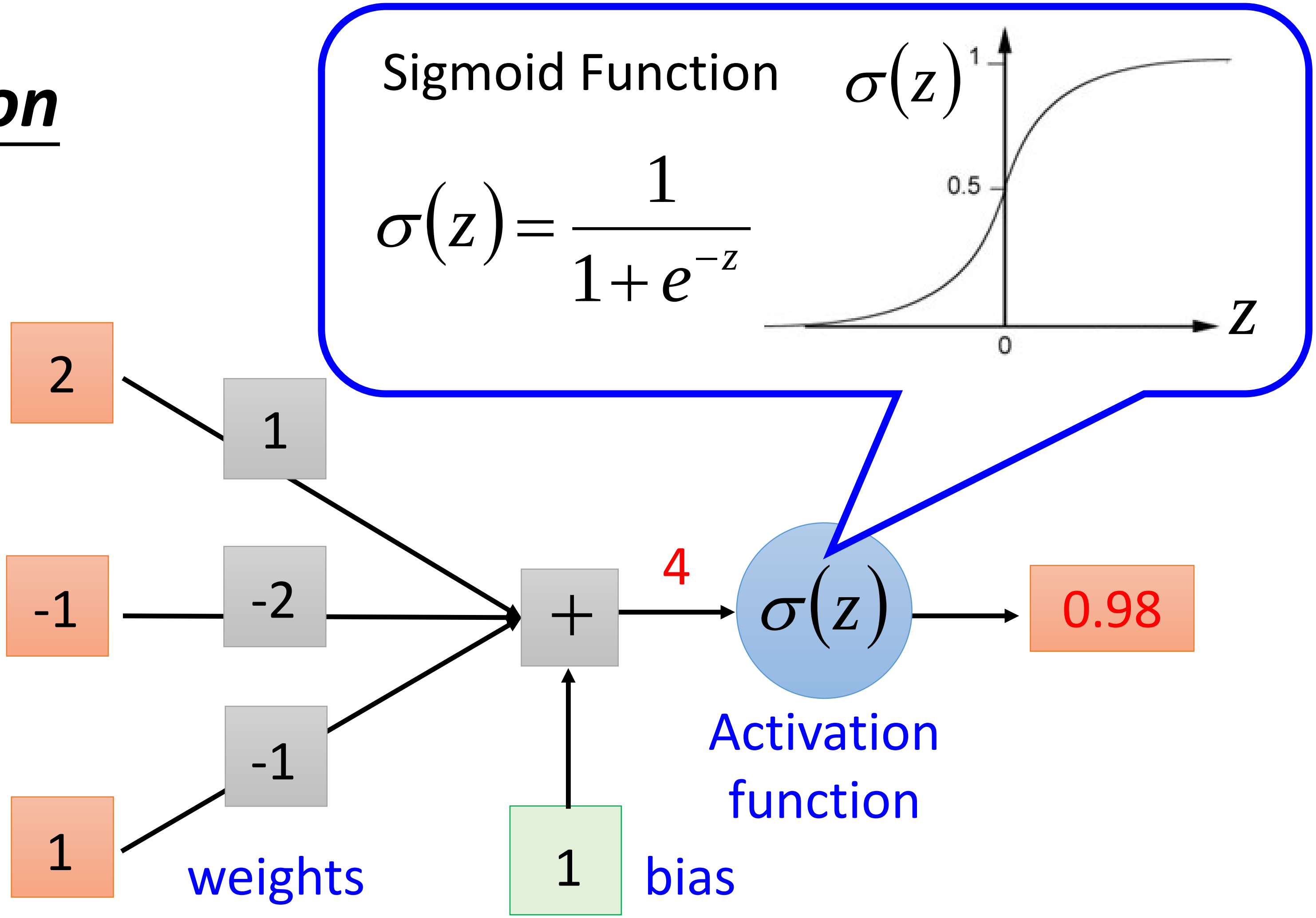
Neuron

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

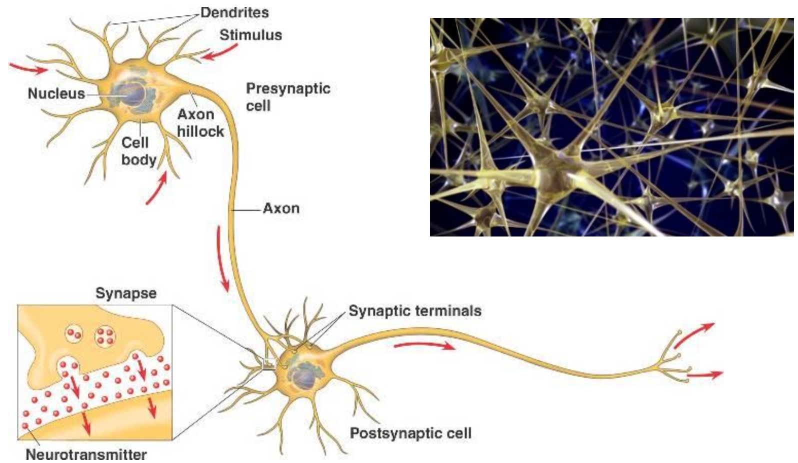


Neural Network

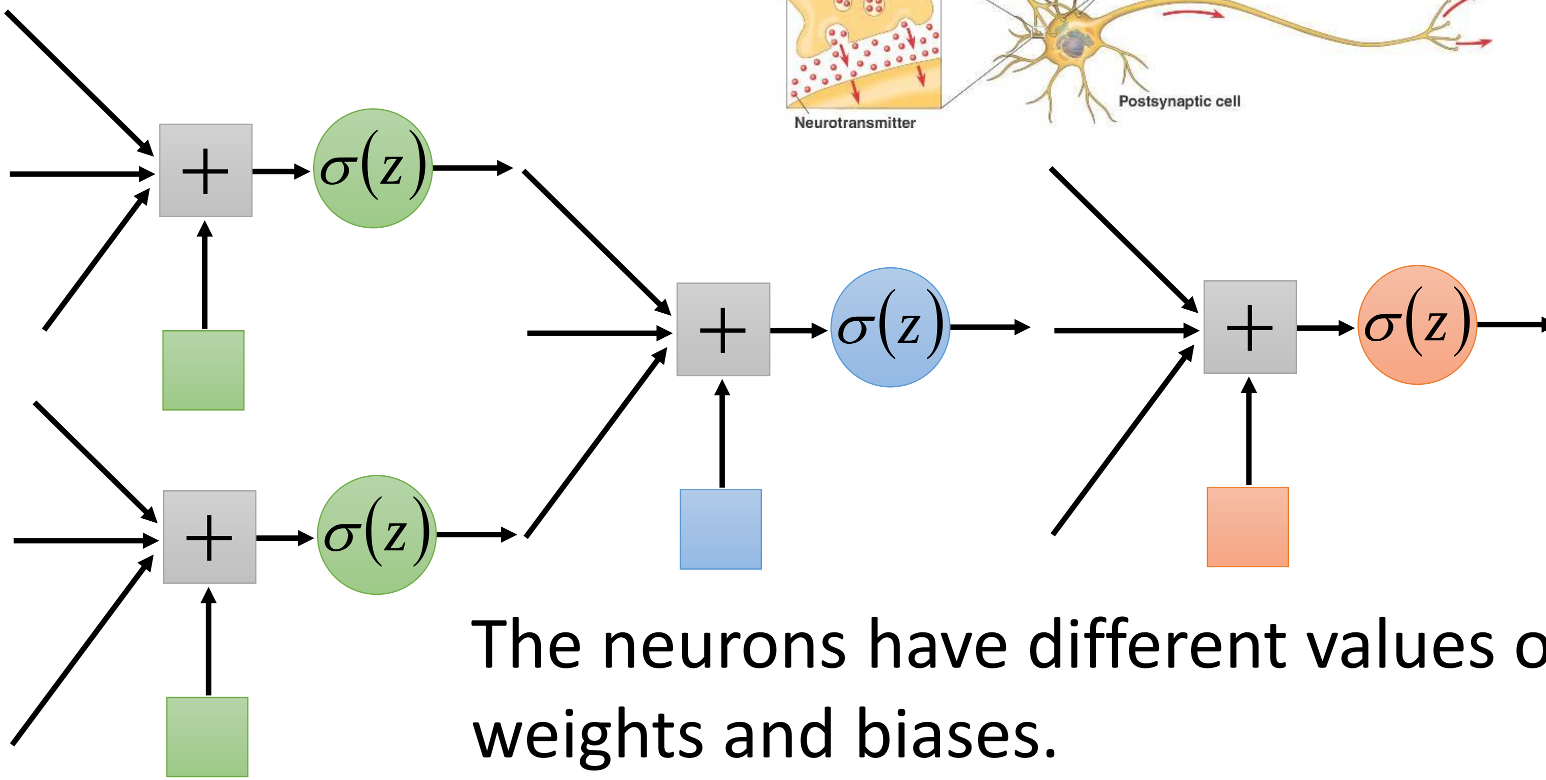
Neuron



Neural Network



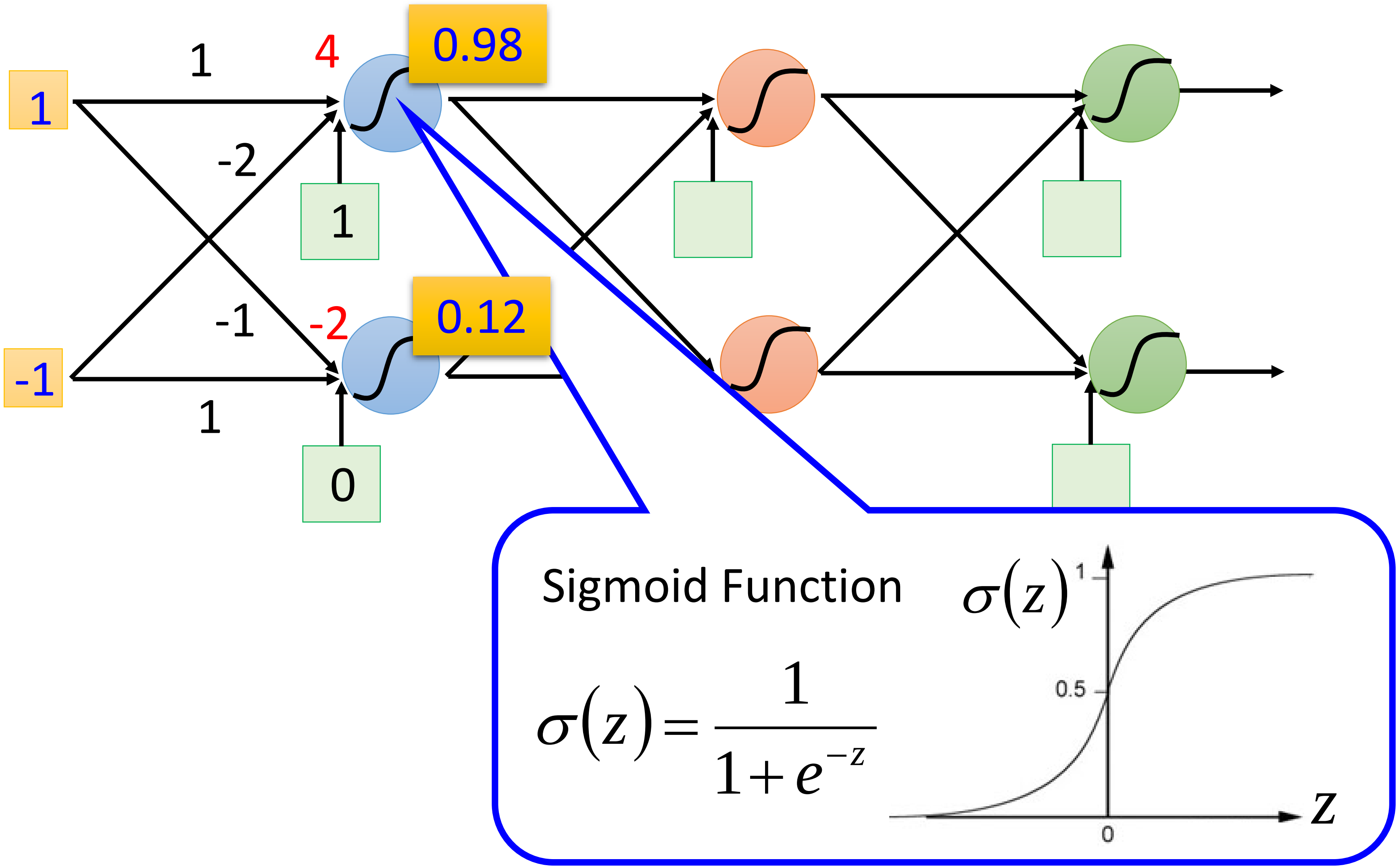
Different connections lead to different network structures



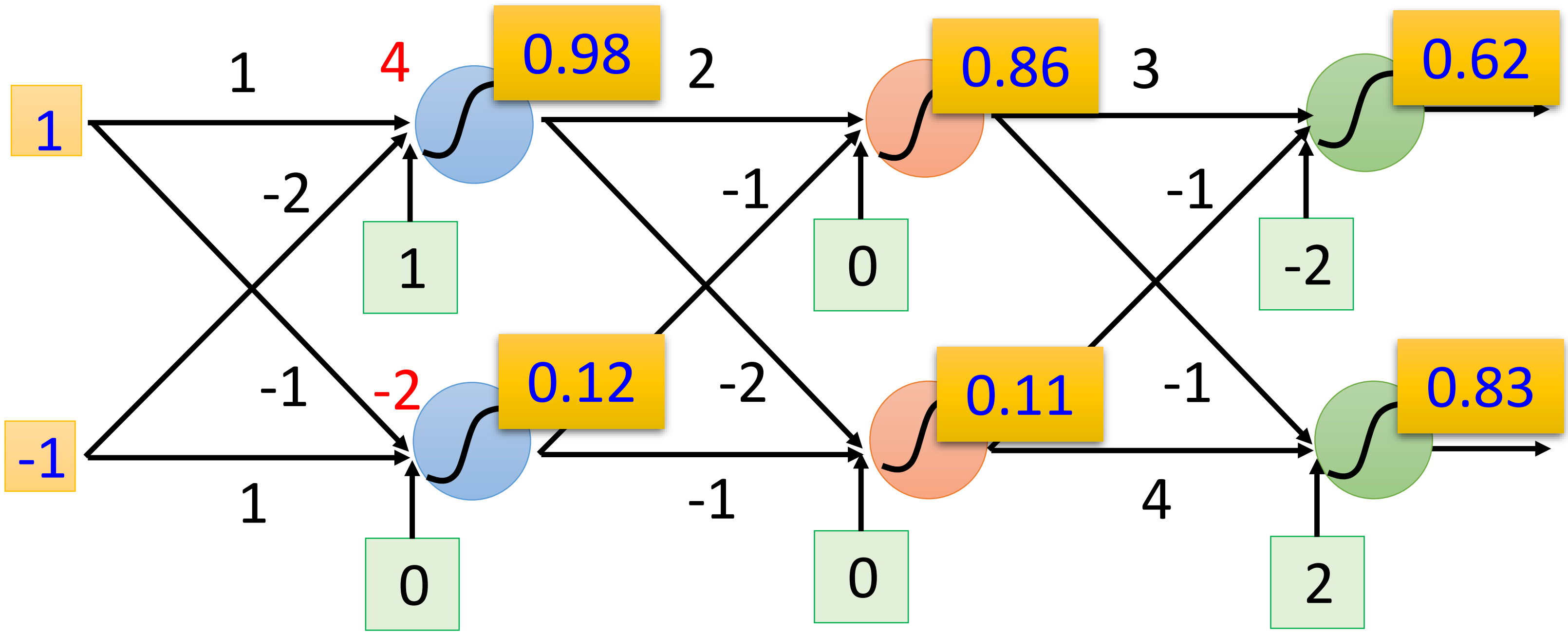
The neurons have different values of weights and biases.

Weights and biases are network parameters θ

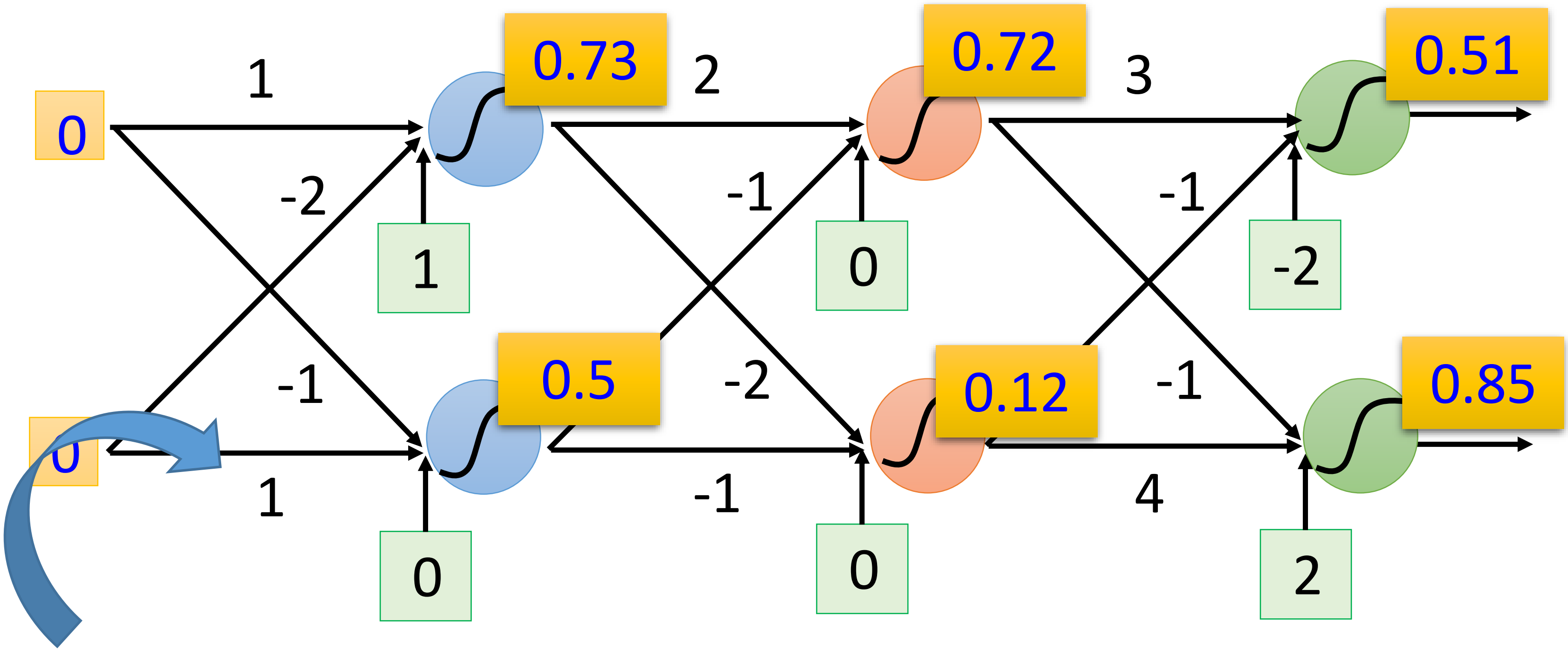
Fully Connect Feedforward Network



Fully Connect Feedforward Network



Fully Connect Feedforward Network



This is a function.

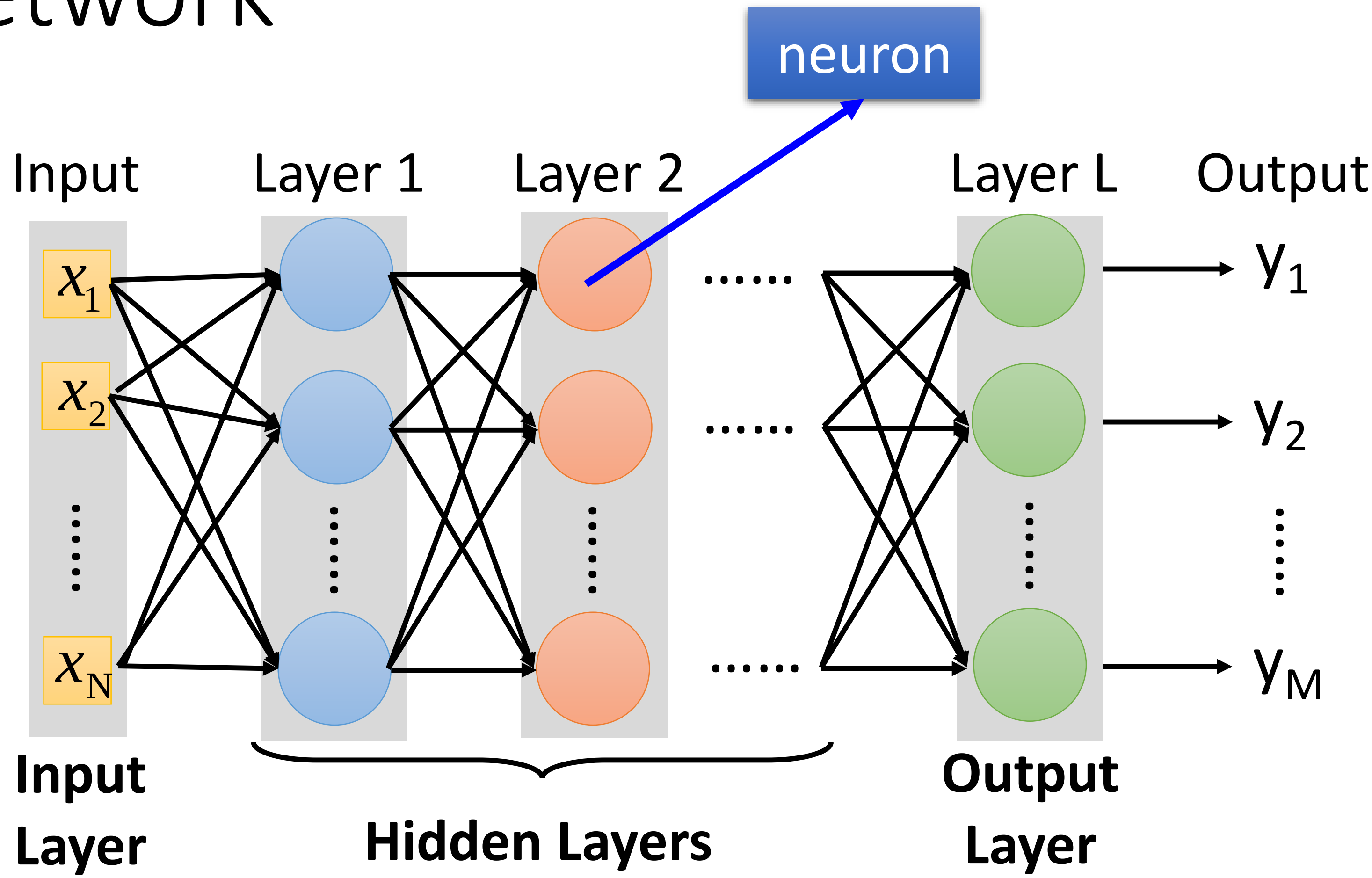
Input vector, output vector

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

Given parameters θ, define a function

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

Fully Connect Feedforward Network



Deep means many hidden layers

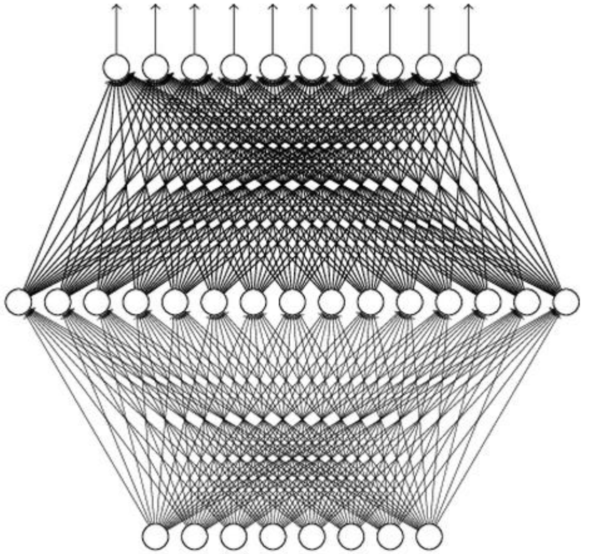
Why Deep? Universality Theorem

Any continuous function f

$$f : R^N \rightarrow R^{\mathrm{M}}$$

Can be realized by a network with one hidden layer

(given **enough** hidden neurons)



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

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

Why Deep? Analogy

Logic circuits

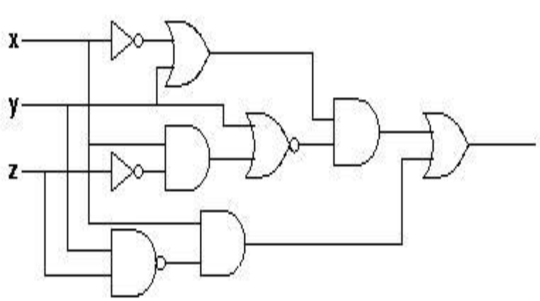
- Logic circuits consists of **gates**
- **A two layers of logic gates** can represent **any Boolean function.**
- Using multiple layers of logic gates to build some functions are much simpler

→ 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?



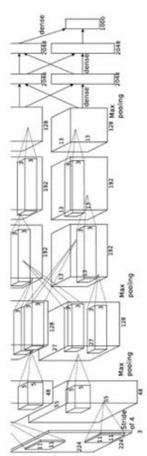
More reason:
https://www.youtube.com/watch?v=XsC9byQkUH8&list=PLJV_el3uVTsPy9oCRY30oBPNLCo89yu49&index=13

Deep = Many hidden layers

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

8 layers

16.4%



AlexNet (2012)

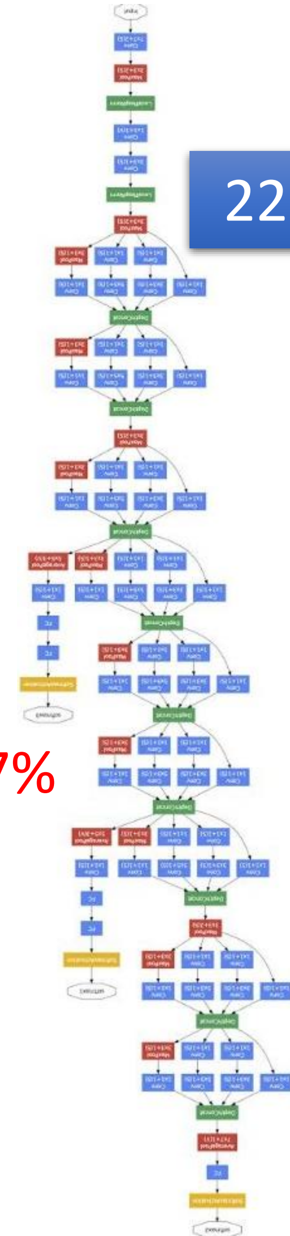
7.3%

19 layers

VGG (2014)

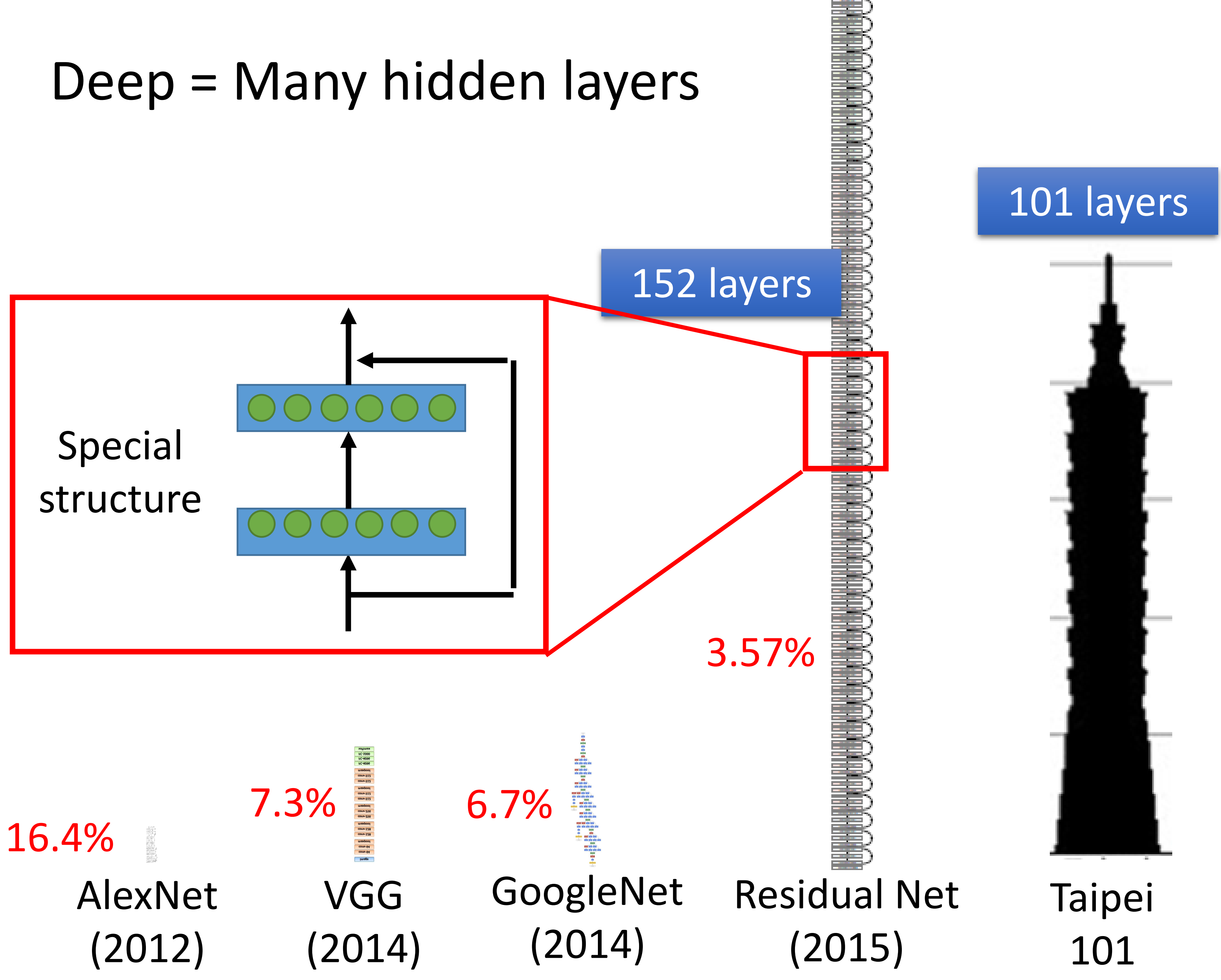
22 layers

6.7%



GoogleNet (2014)

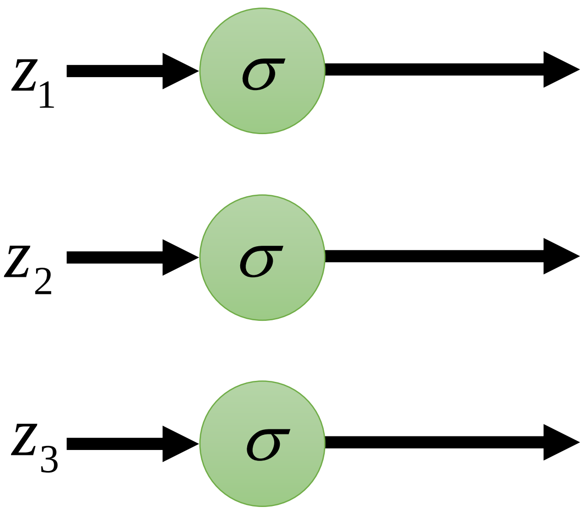
Deep = Many hidden layers



Output Layer

- Softmax layer as the output layer

<u>Ordinary Layer</u>



$$y_1 = \sigma(z_1)$$

$$y_2 = \sigma(z_2)$$

$$y_3 = \sigma(z_3)$$

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

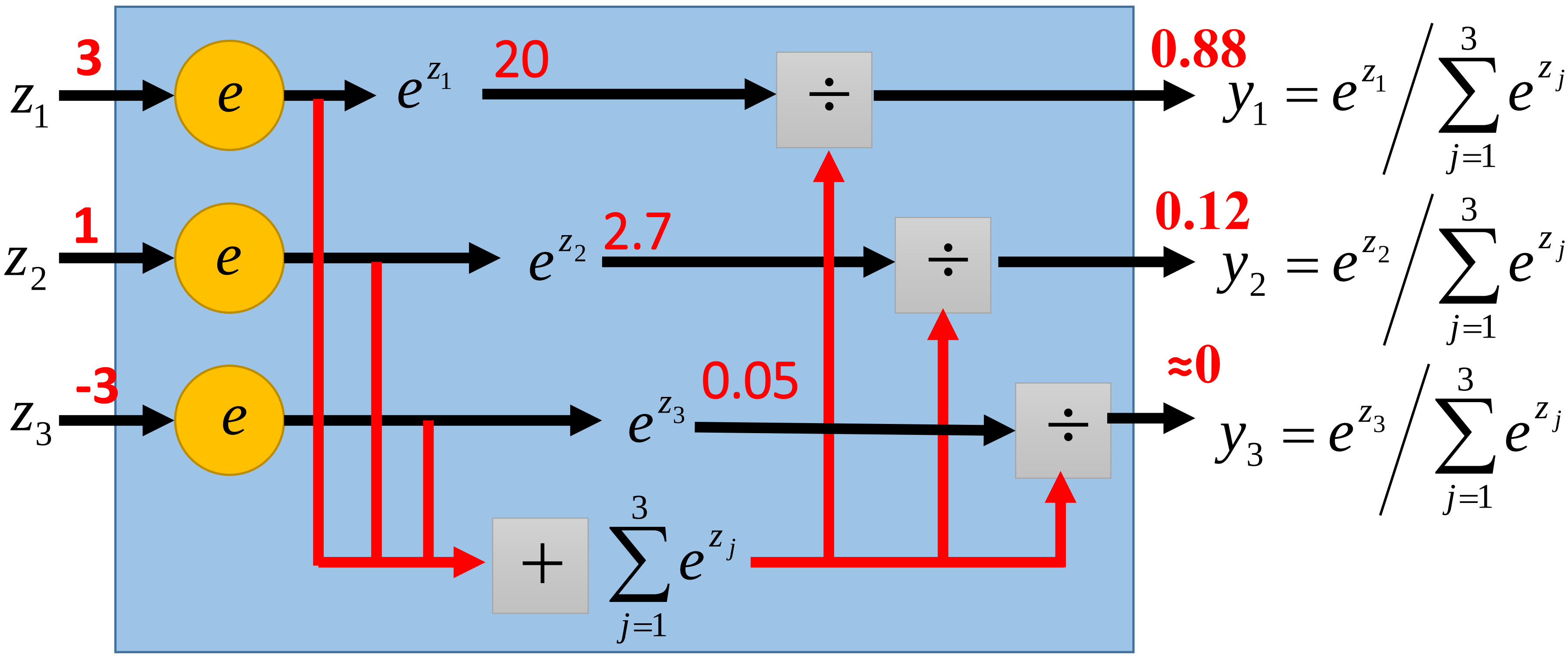
May not be easy to interpret

Output Layer

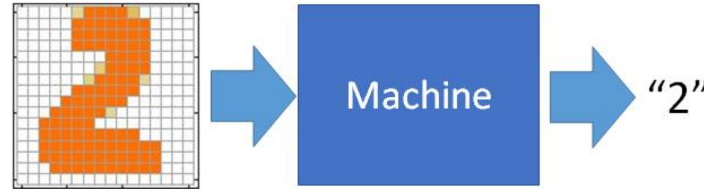
- Softmax layer as the output layer

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

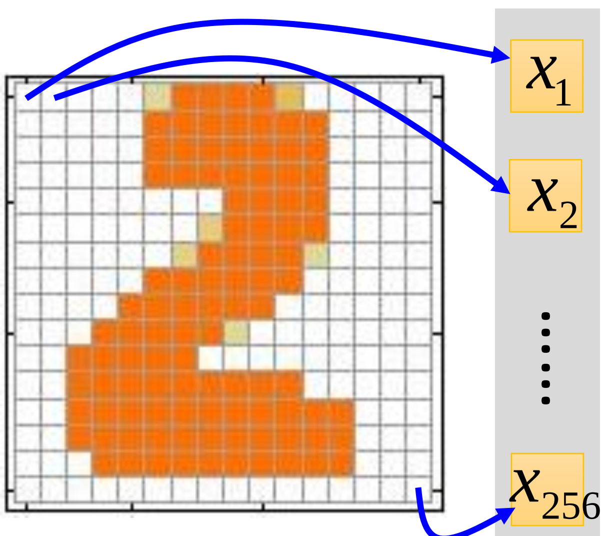
Softmax Layer



Example Application



Input

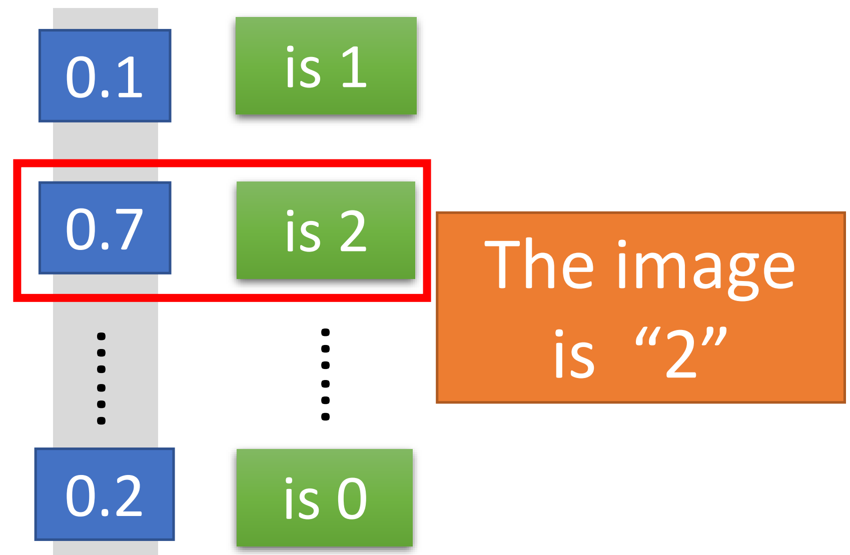


16 x 16 = 256

Ink → 1

No ink → 0

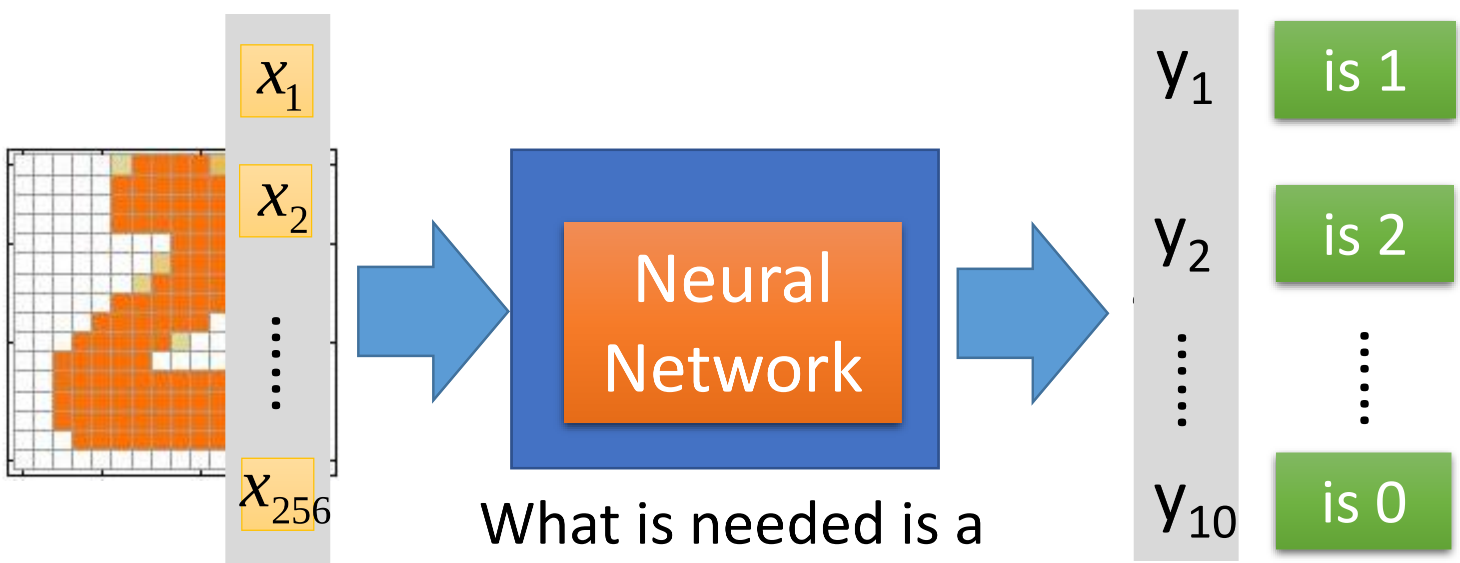
Output



Each dimension represents the confidence of a digit.

Example Application

- Handwriting Digit Recognition

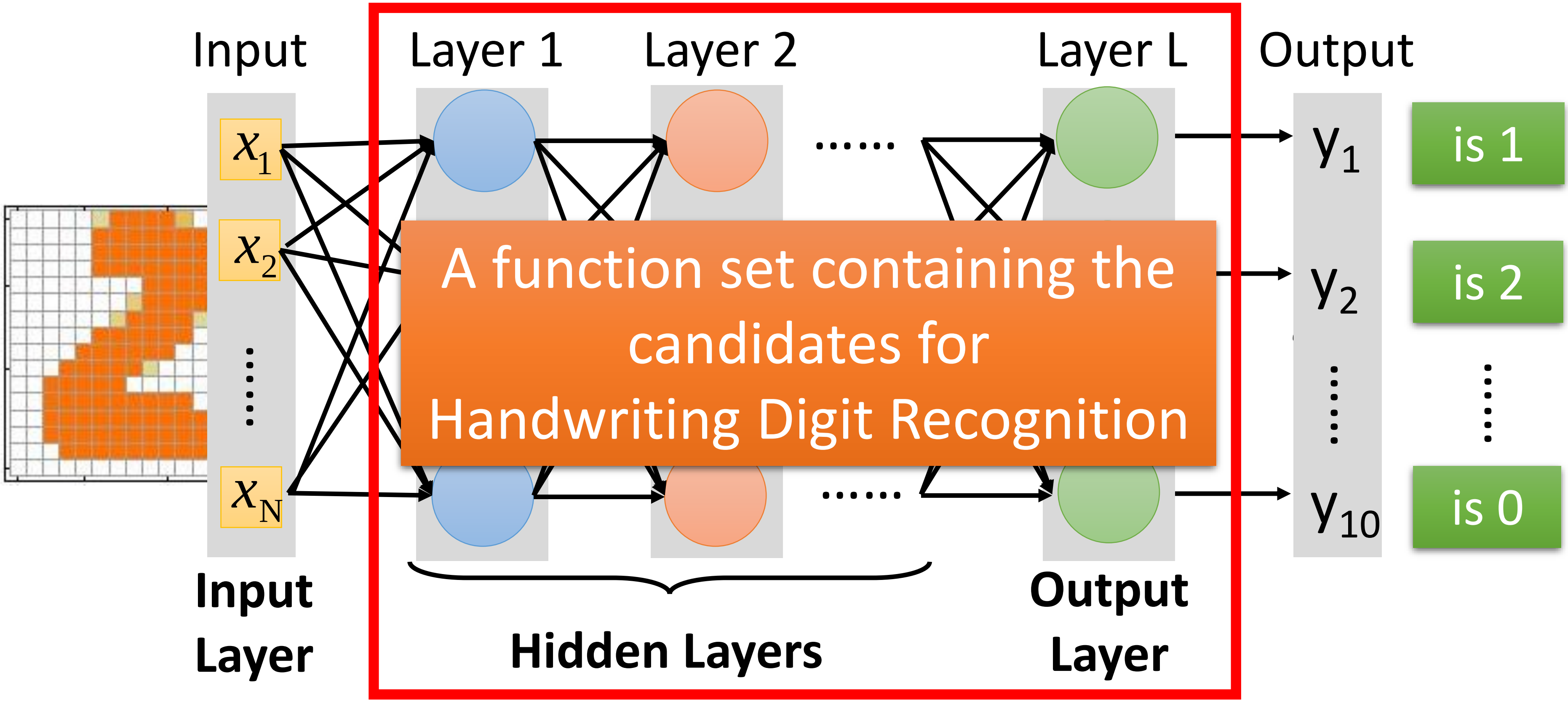


What is needed is a function ......

Input:
256-dim vector

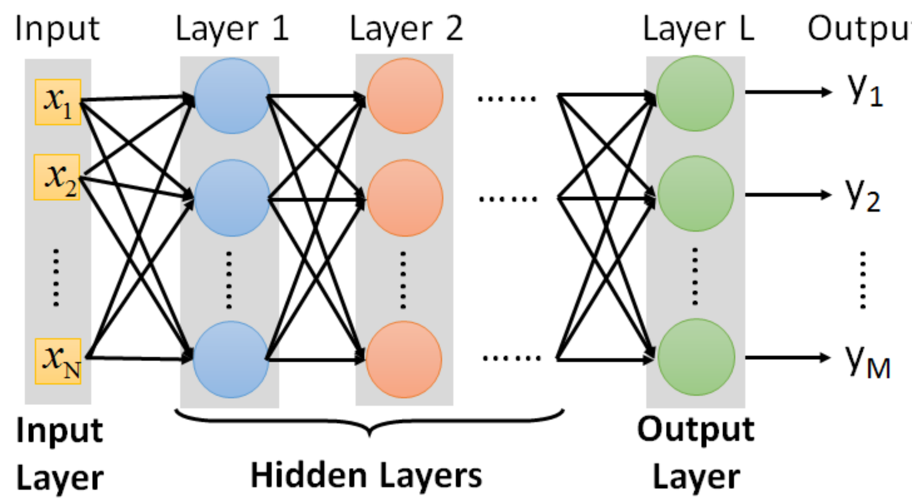
output:
10-dim vector

Example Application



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

FAQ



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

  Trial and Error + **Intuition**

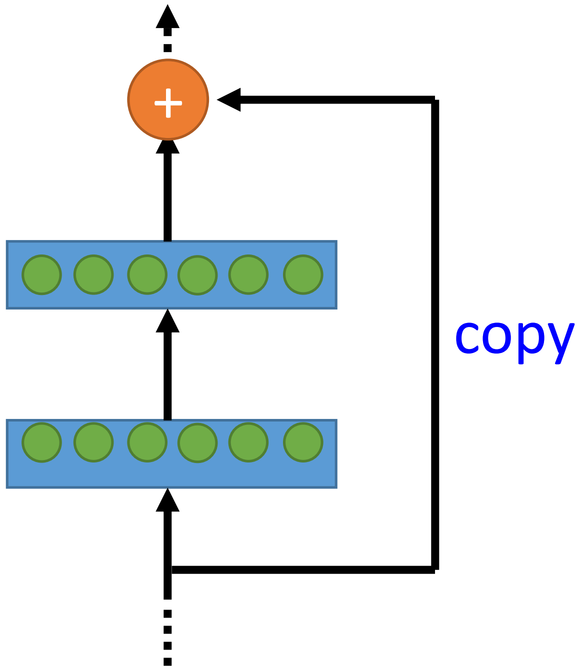
- Q: Can we design the network structure?

  Convolutional Neural Network (CNN) in the next lecture

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

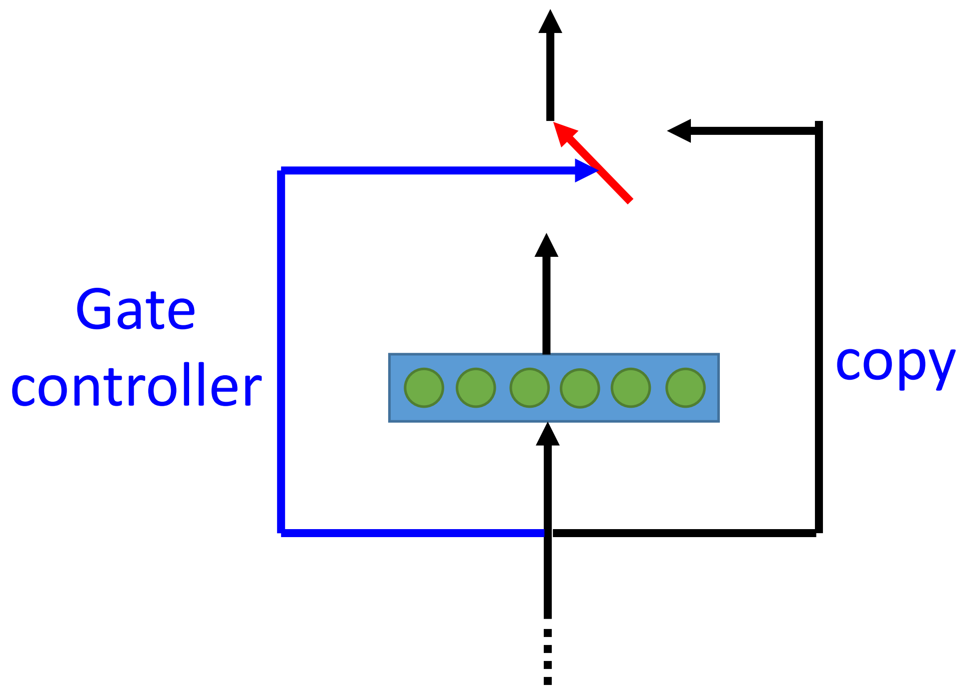
Highway Network

- **Residual Network**

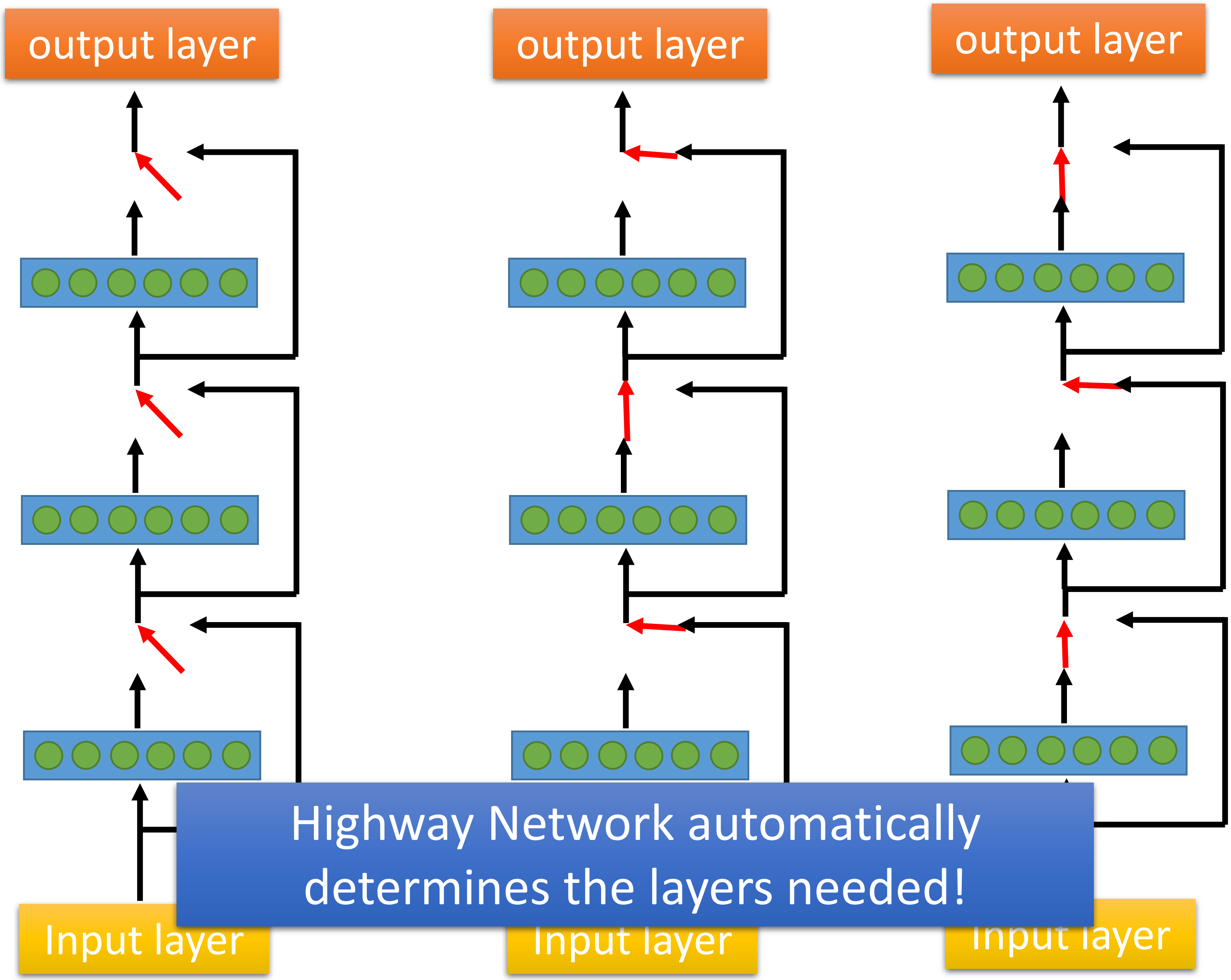


Deep Residual Learning for Image Recognition
http://arxiv.org/abs/1512.03385

- **Highway Network**



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


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

Three Steps for Deep Learning

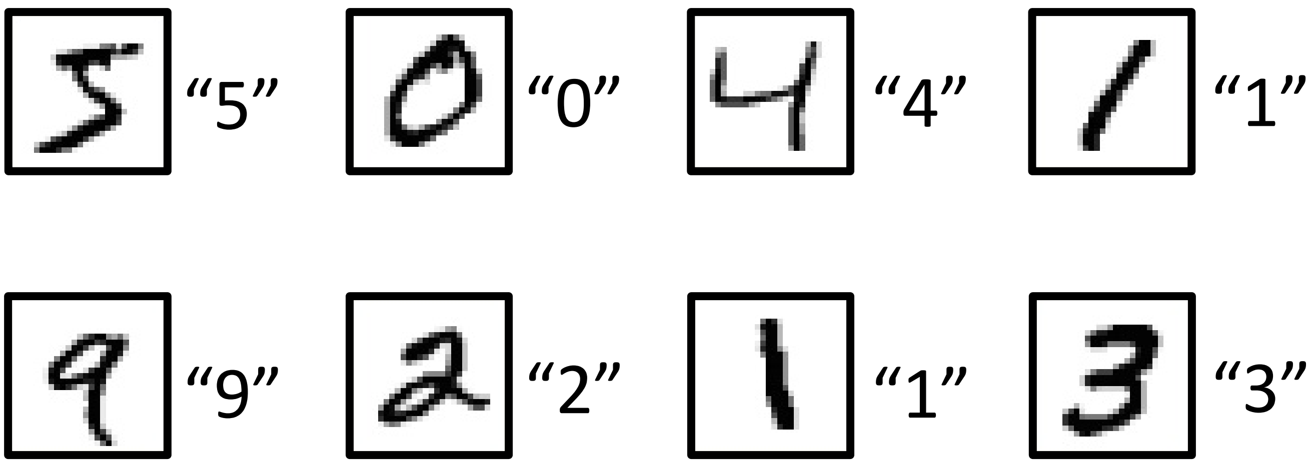
Step 1: define a set of function

Step 2: goodness of function

Step 3: pick the best function

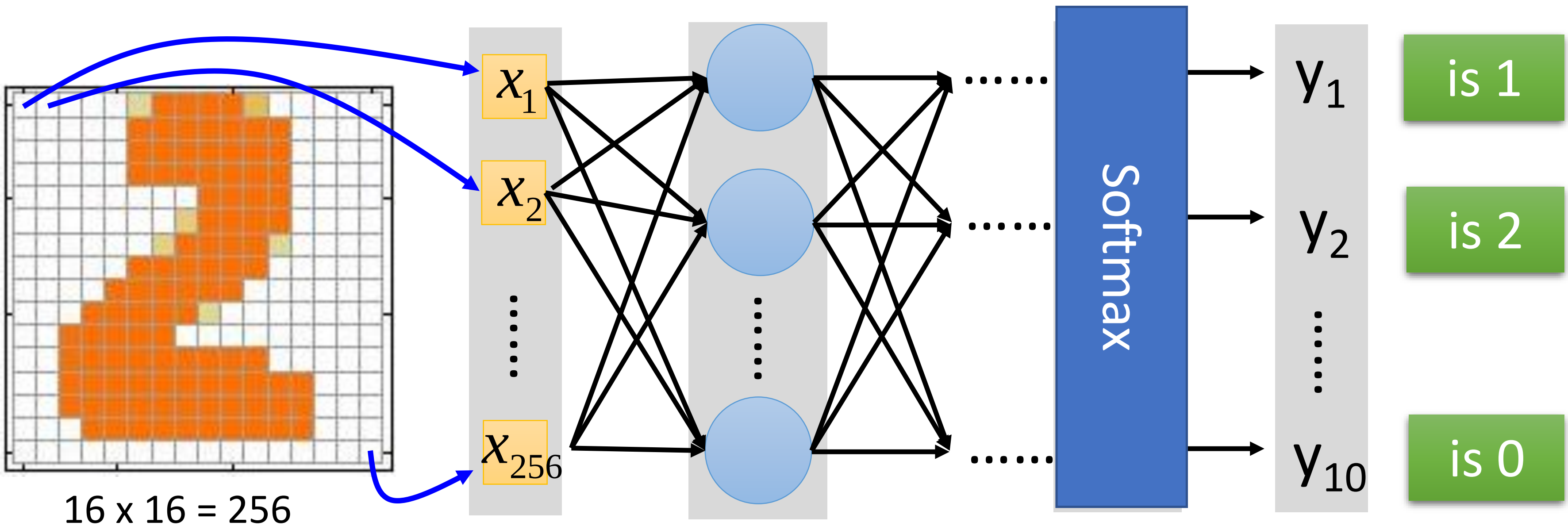
Training Data

- Preparing training data: images and their labels



The learning target is defined on the training data.

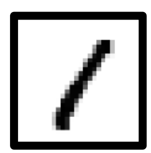
Learning Target



Ink → 1

No ink → 0

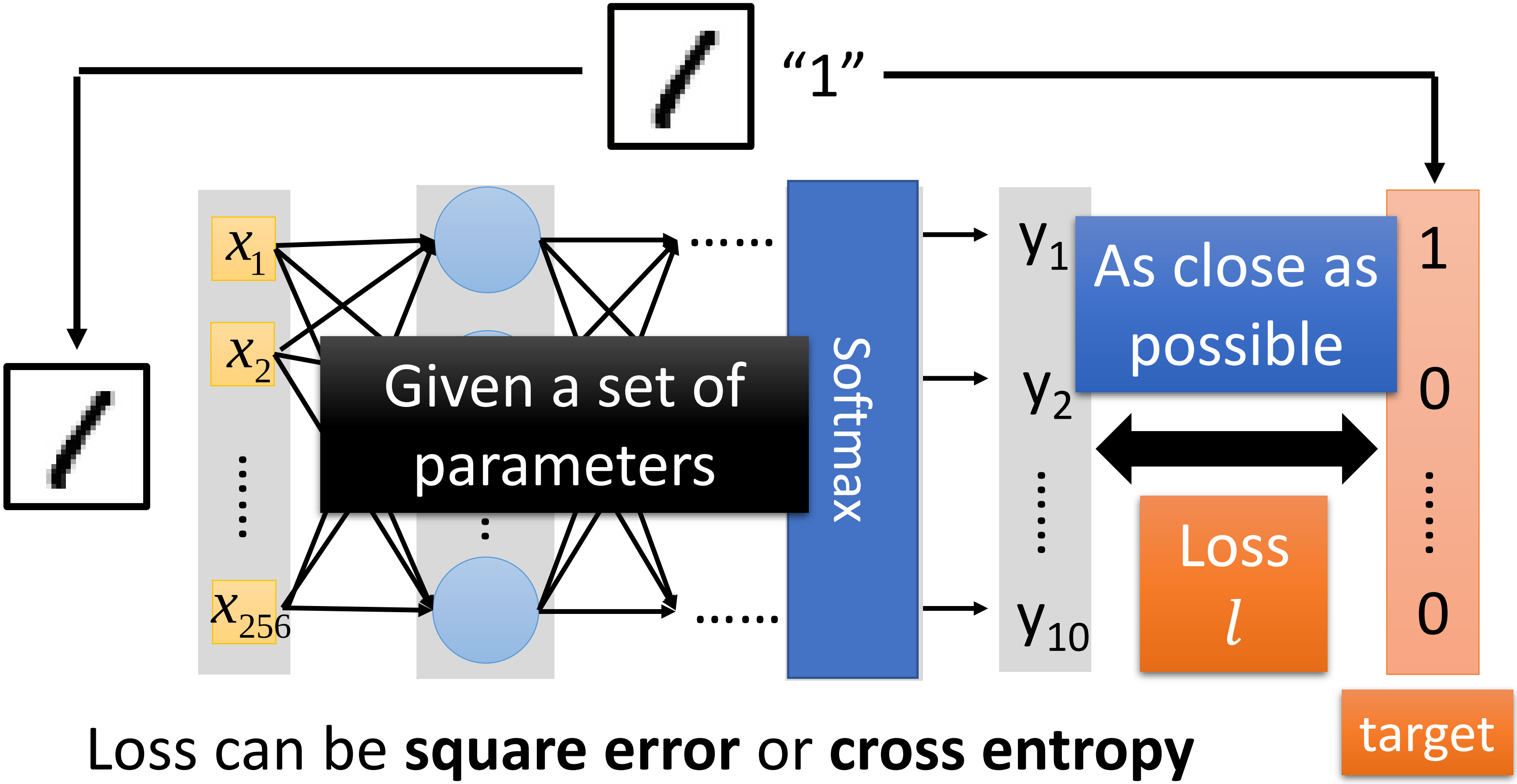
The learning target is ……

Input:  → y_1 has the maximum value

Input:  → y_2 has the maximum value

Loss

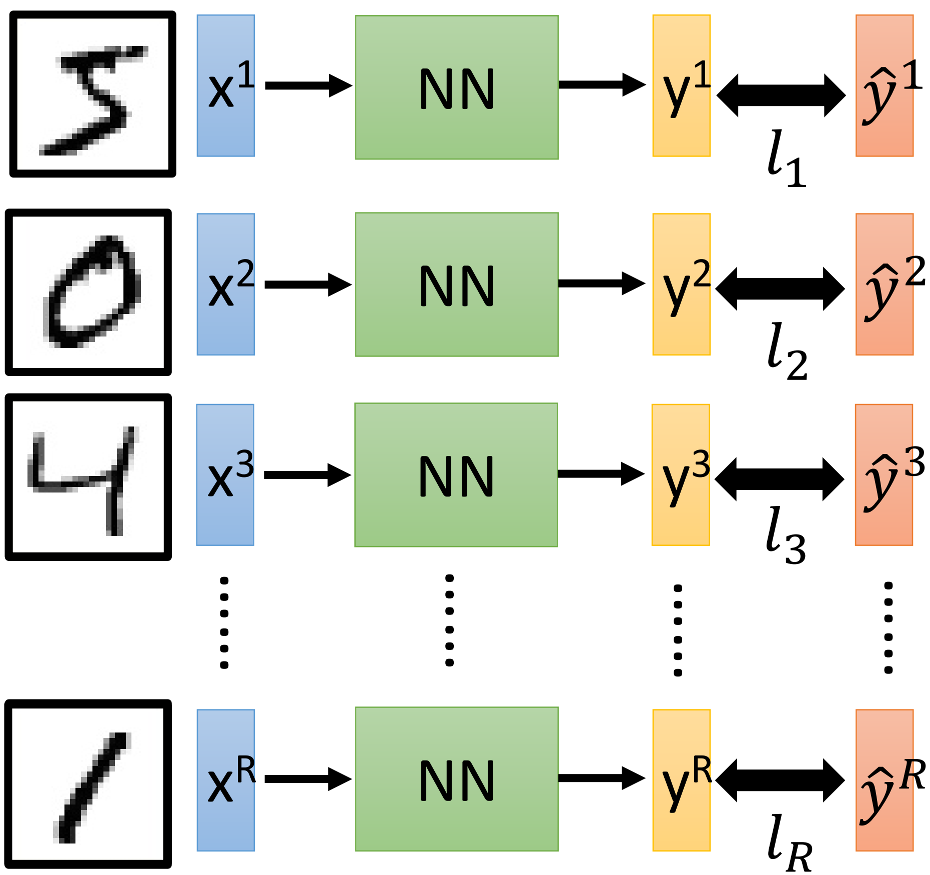
A good function should make the loss of all examples as small as possible.



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

Total Loss

For all training data ...



Total Loss:

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

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

Step 1: define a set of function

Step 2: goodness of function

Step 3: pick the best function

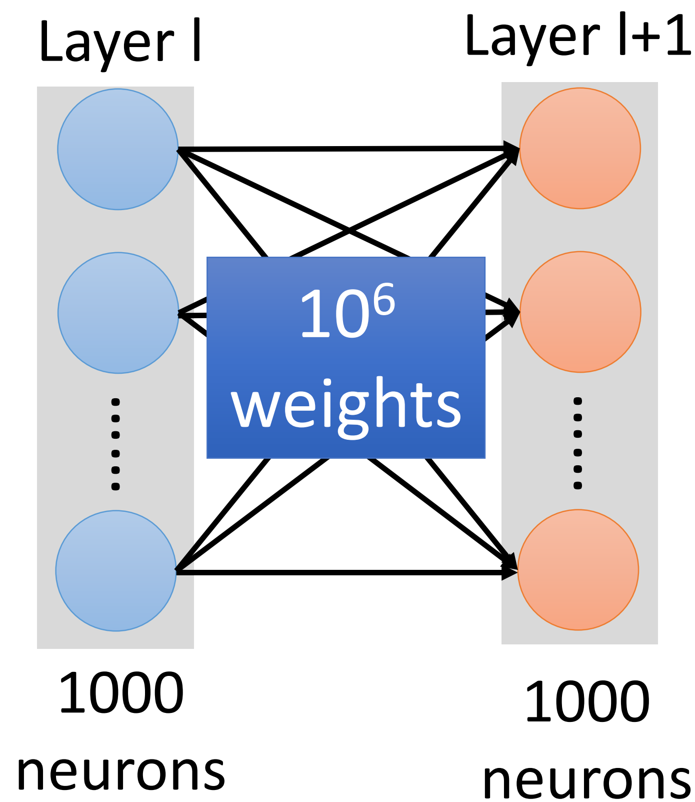
How to pick the best function

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

Enumerate all possible values

$$\underbrace{\text{Network parameters } \theta = \{w_1, w_2, w_3, \cdots, b_1, b_2, b_3, \cdots\}}_{\text{Millions of parameters}}$$

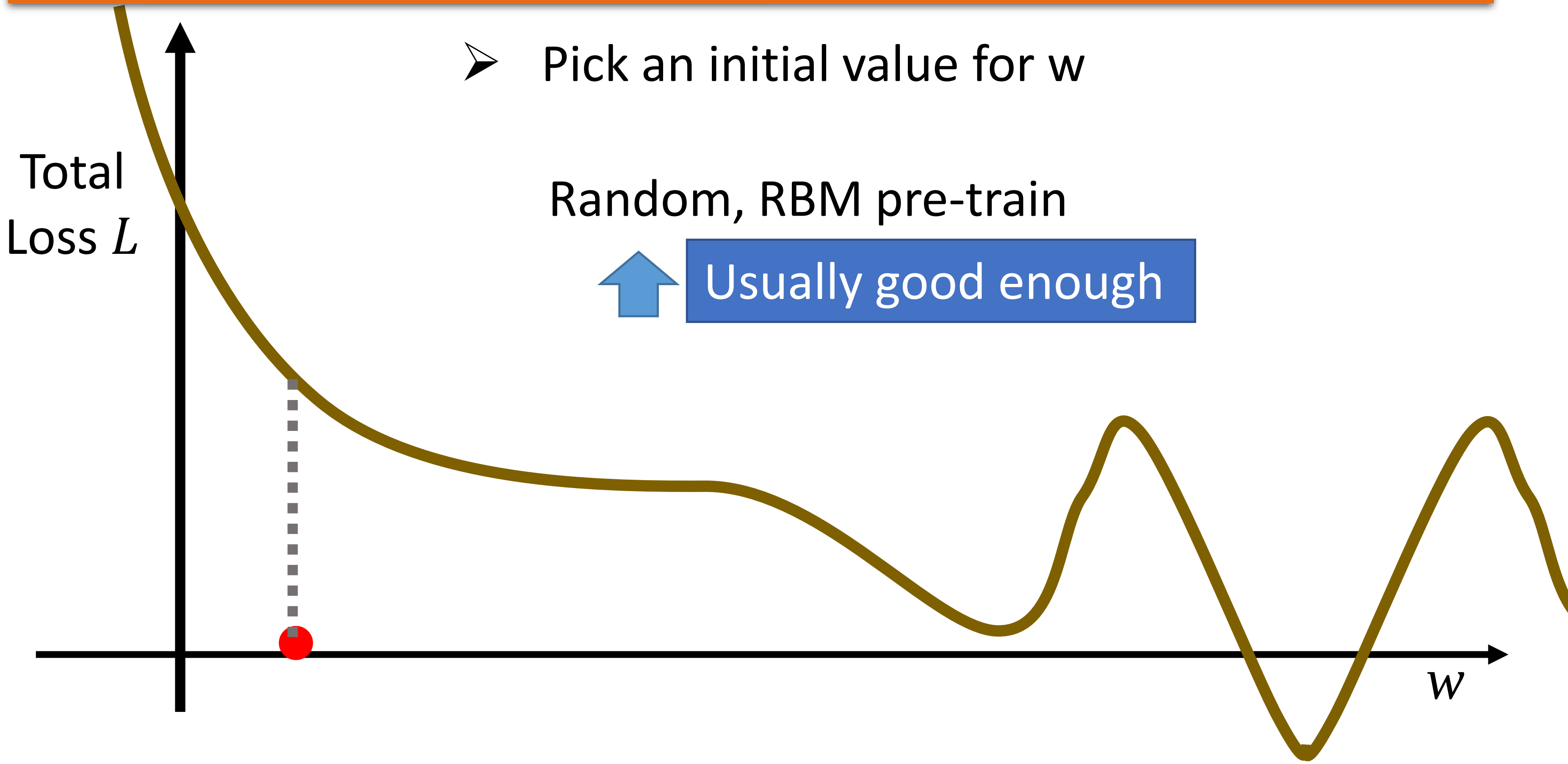
E.g. speech recognition: 8 layers and 1000 neurons each layer



Gradient Descent

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

Find ***network parameters*** $\boldsymbol{\theta^*}$ that minimize total loss L



- Pick an initial value for w

Random, RBM pre-train

Usually good enough

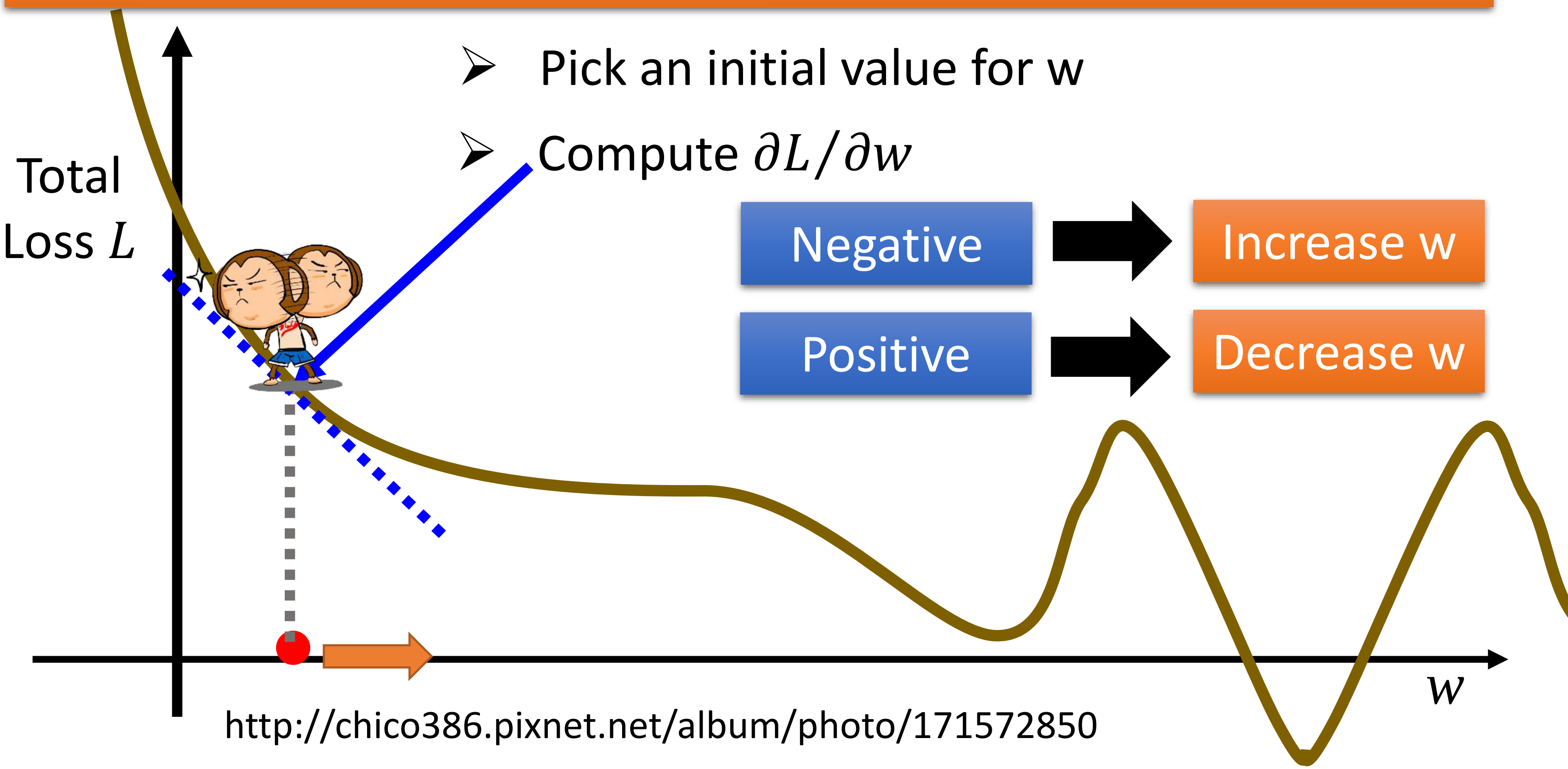
Gradient Descent

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

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

- Pick an initial value for w
- Compute $\partial L / \partial w$

Negative	➡	Increase w
Positive	➡	Decrease w



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

Gradient Descent

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

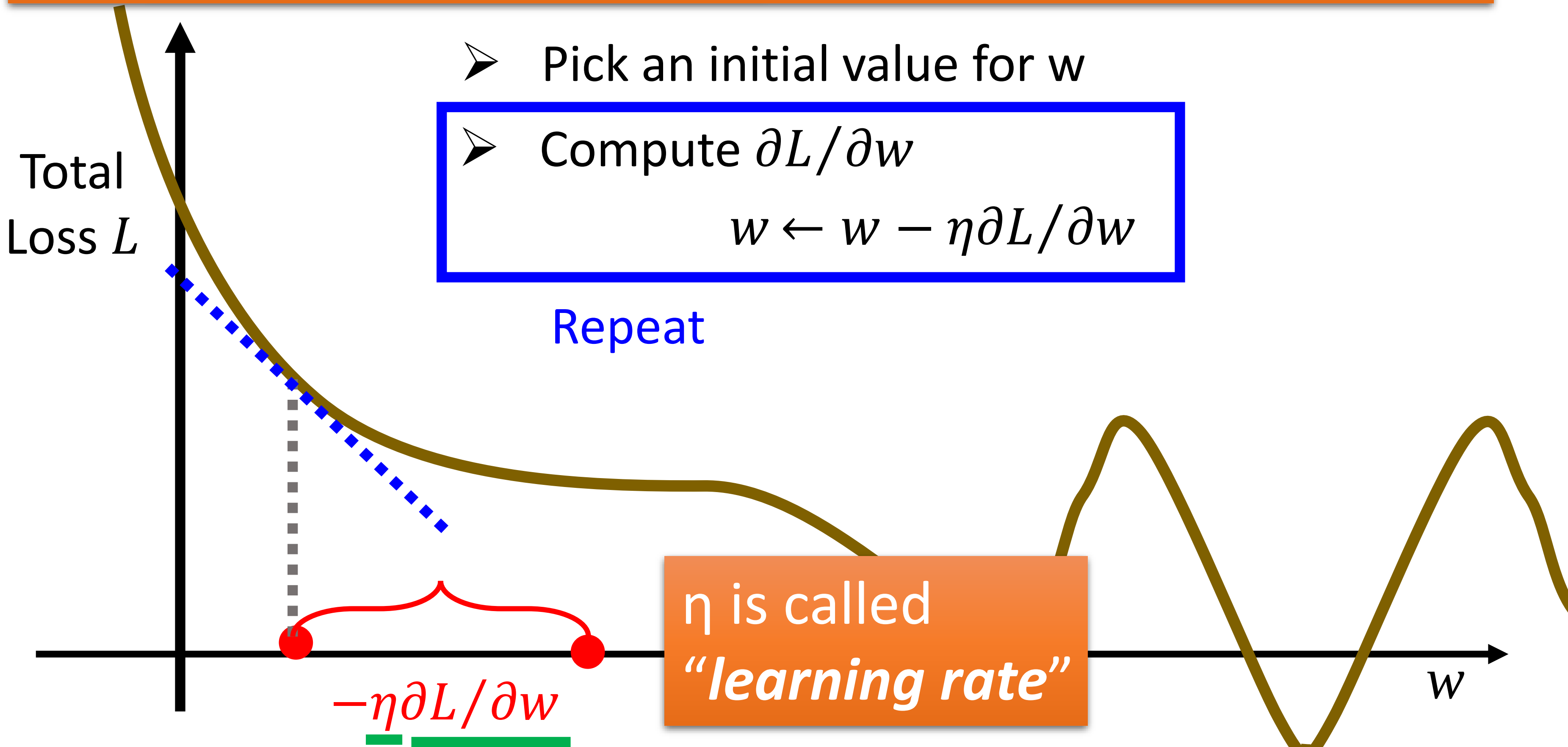
Find ***network parameters*** θ^* that minimize total loss L

- Pick an initial value for w

- Compute $\partial L / \partial w$

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

Repeat



η is called *"learning rate"*

Gradient Descent

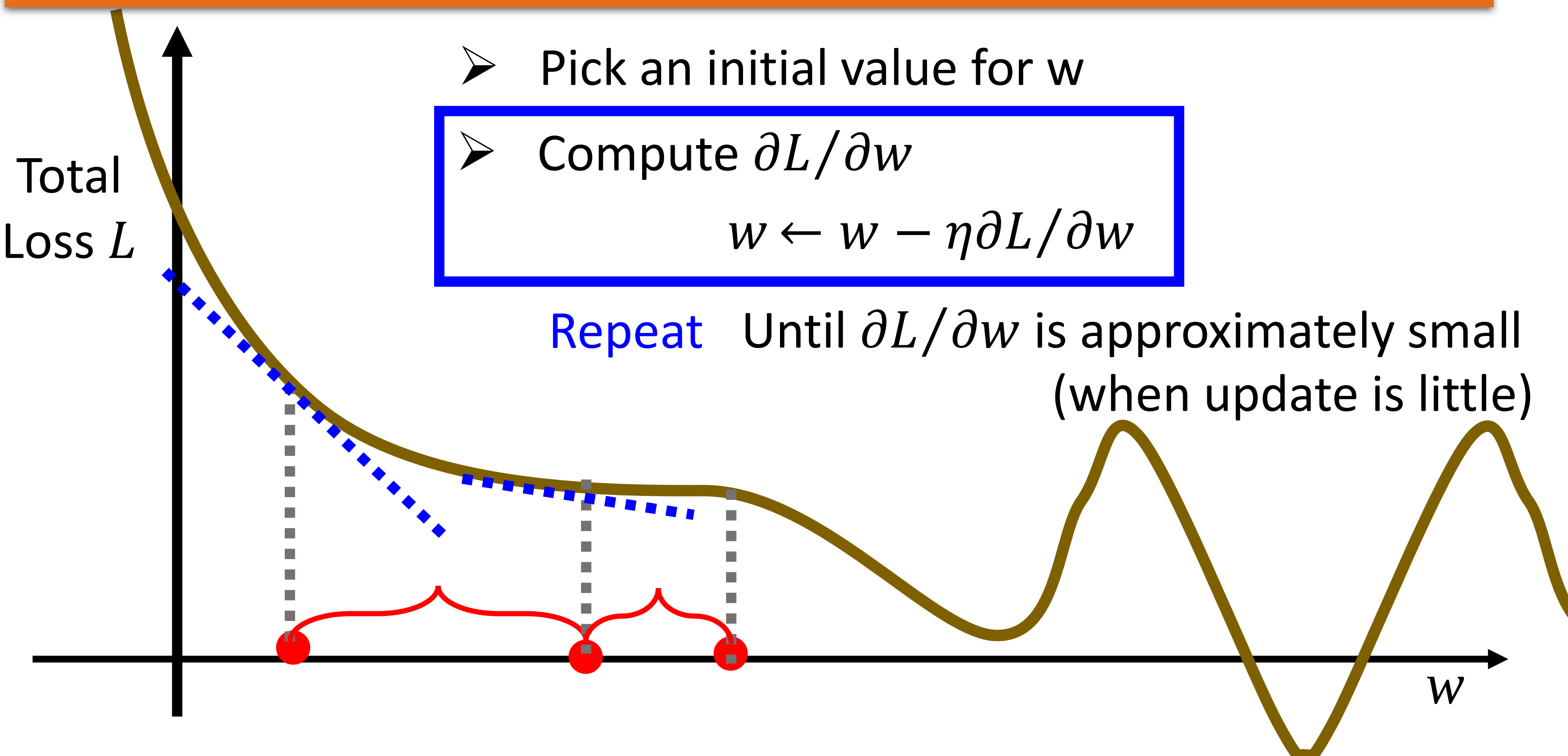
Network parameters $\theta = \{w_1, w_2, \cdots, b_1, b_2, \cdots\}$

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

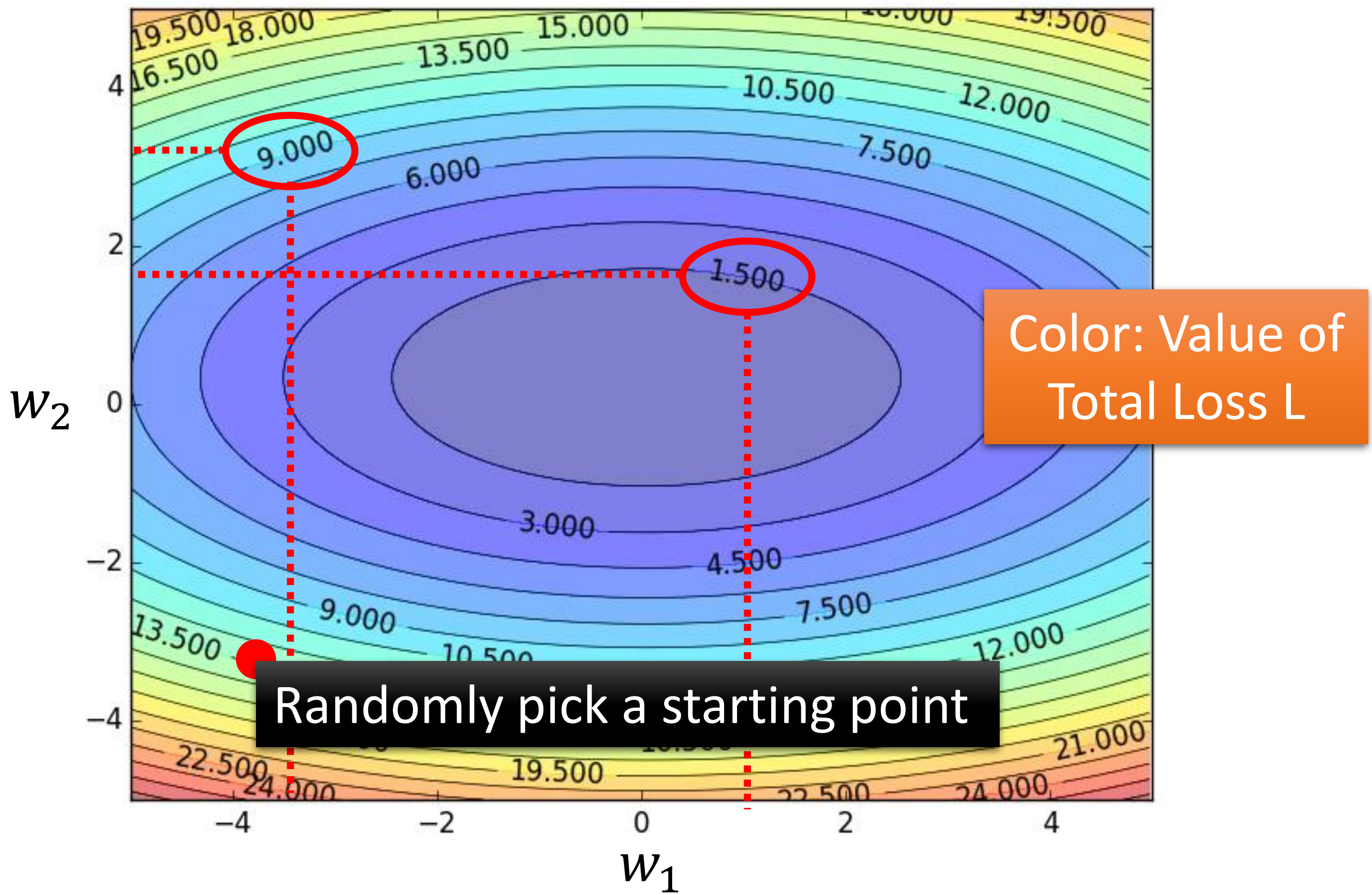
- Pick an initial value for w
- Compute $\partial L / \partial w$

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

Repeat Until $\partial L / \partial w$ is approximately small (when update is little)

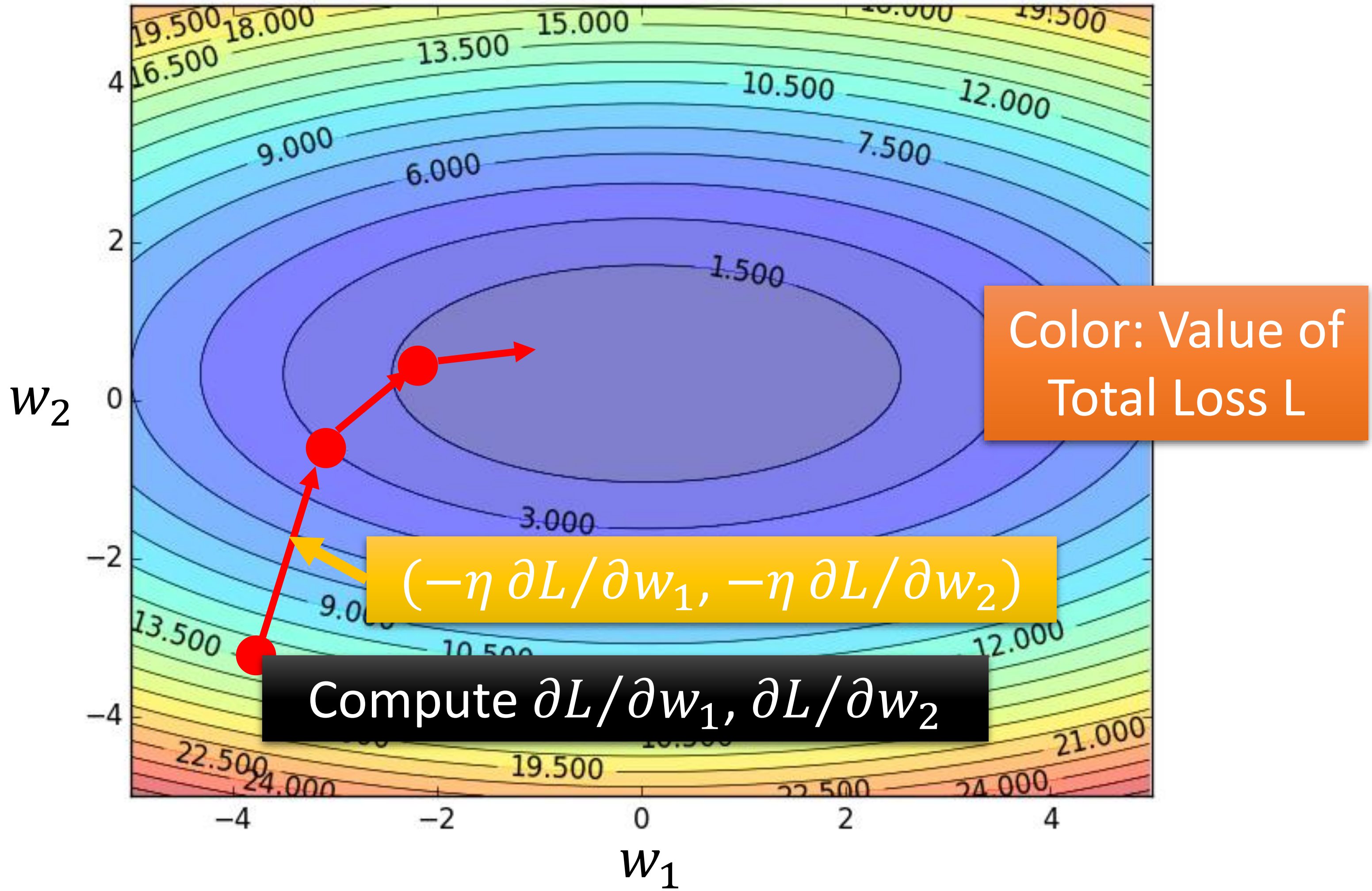


Gradient Descent

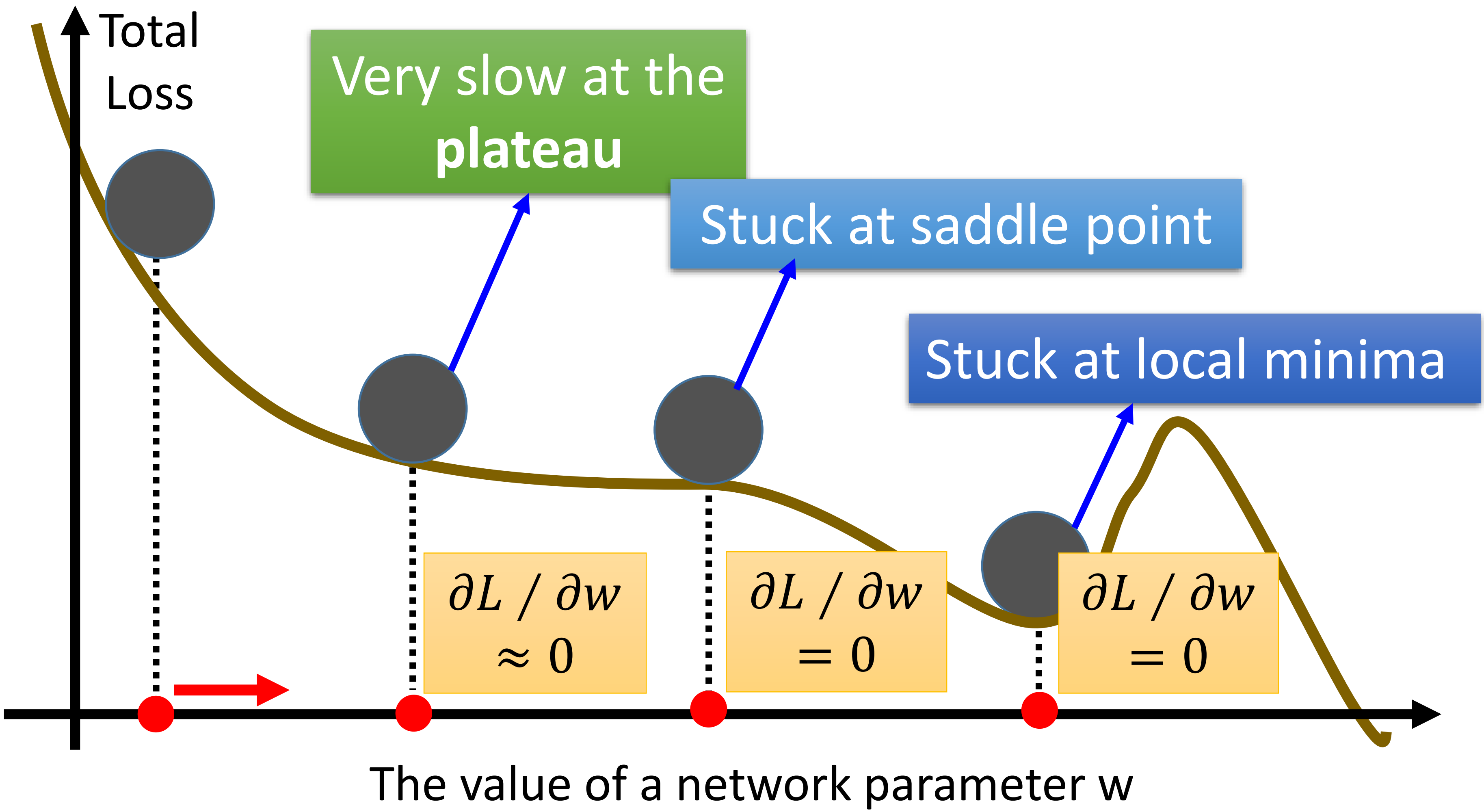


Gradient Descent

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

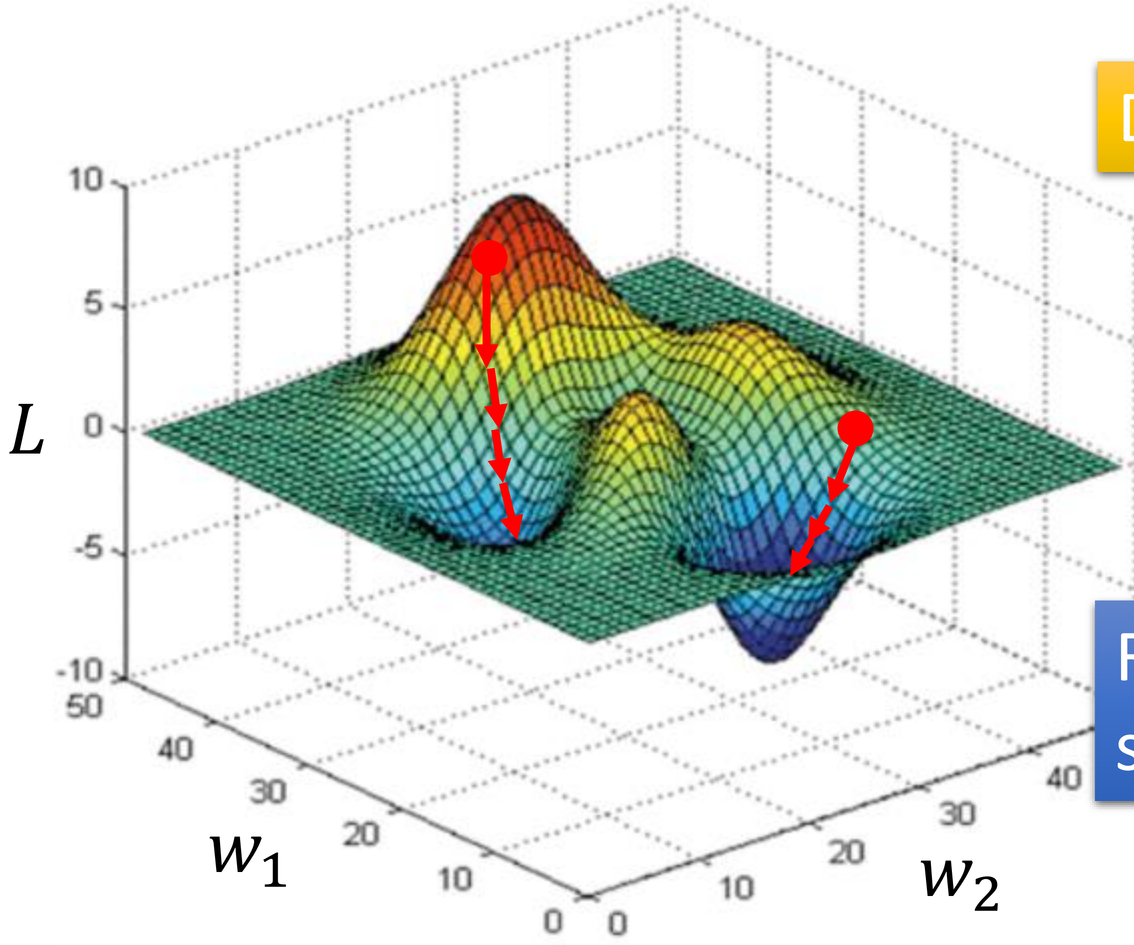


Local Minima



Local Minima

- Gradient descent never guarantee global minima



Different initial point

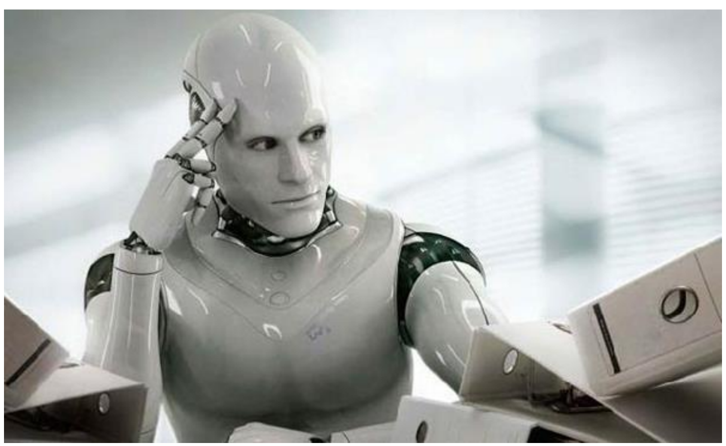
Reach different minima, so different results

Gradient Descent

This is the "learning" of machines in deep learning ……

➡ Even alpha go using this approach.

People image ……



Actually …..

I hope you are not too disappointed :p

Backpropagation

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

TensorFlow

torch

theano

Caffe

Microsoft CNTK

Chainer

Deep Learning library produced by Amazon
DSSTNE

mxnet

libdnn
台大周伯威
同學開發

Ref: https://www.youtube.com/watch?v=ibJpTrp5mcE

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 ……

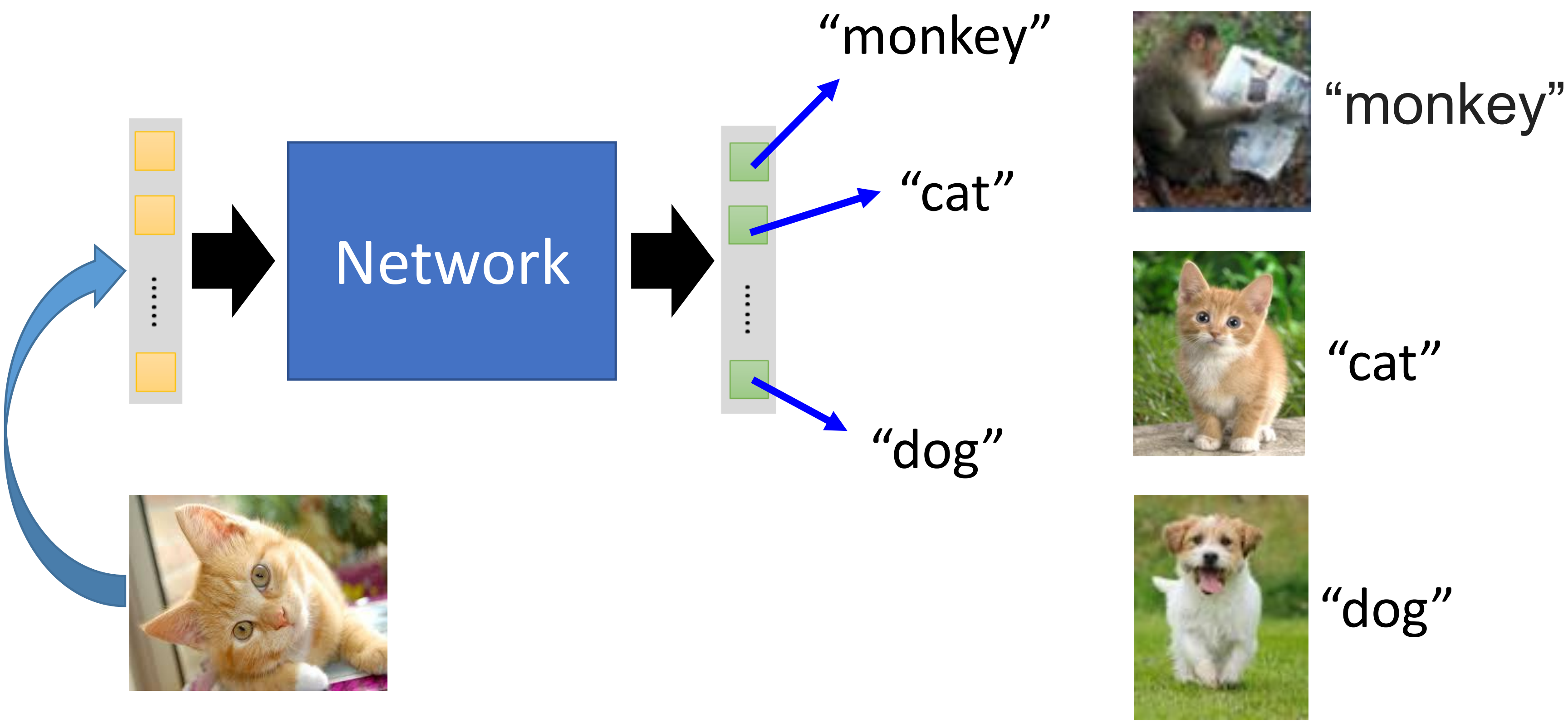
Now If you want to find a function

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

"Talk" in e-mail

"free" in e-mail

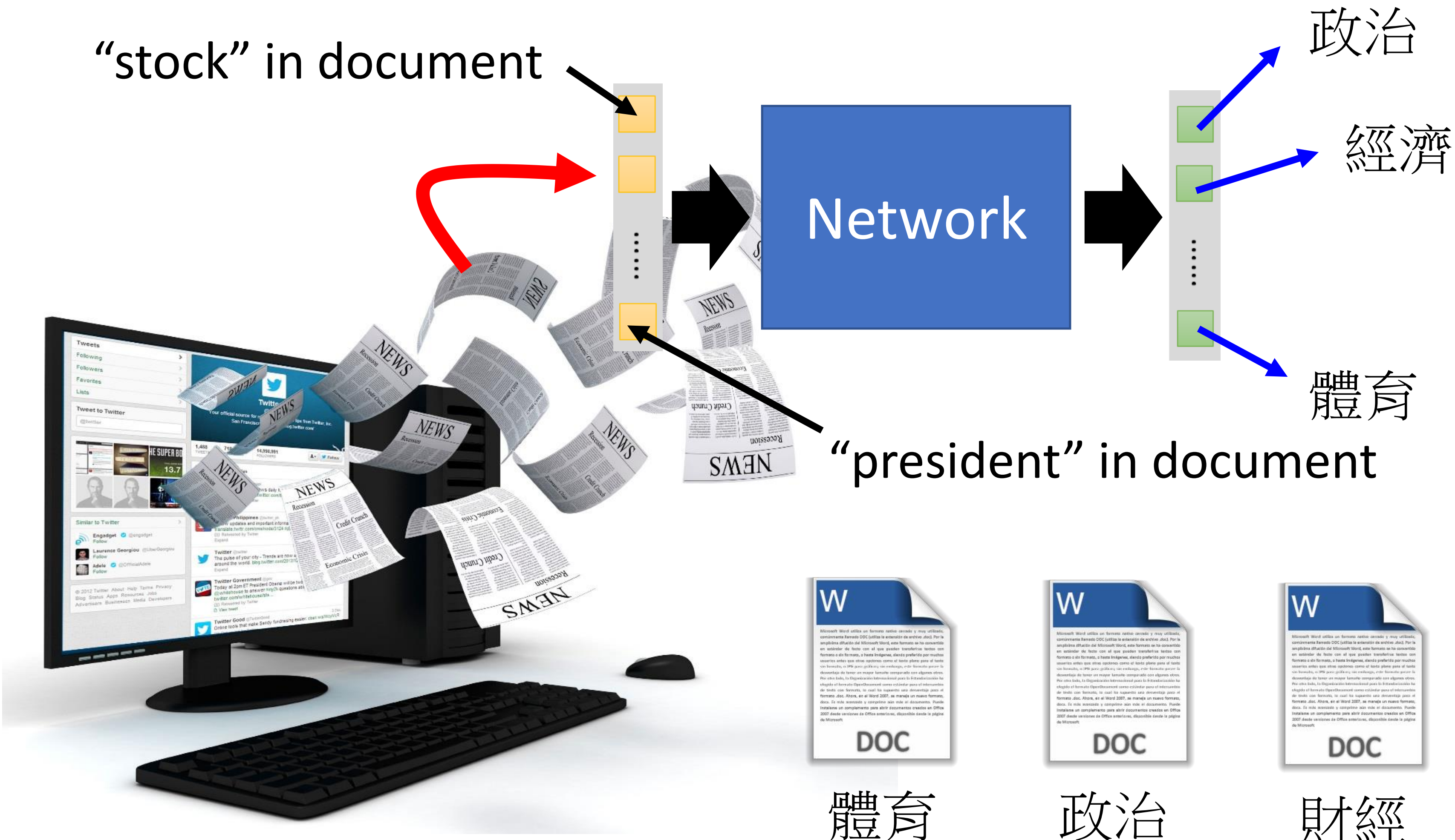
Network

1/0

(Yes/No)

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

For example, you can do …….



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

Outline

Introduction of Deep Learning

"Hello World" for Deep Learning

Tips for Deep Learning

Keras

If you want to learn theano:

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

http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2/Lecture/RNN%20training%20(v6).ecm.mp4/index.html

TensorFlow 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

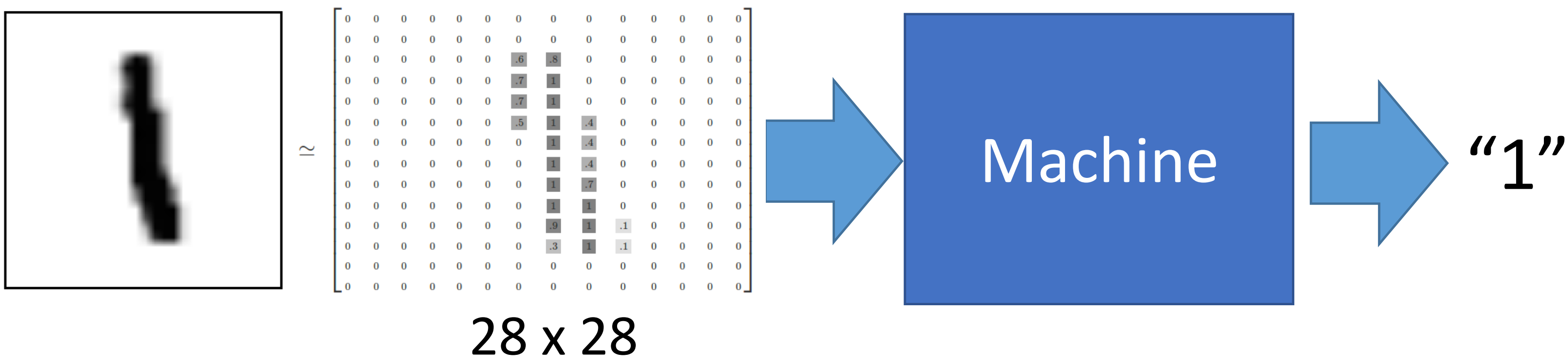
- François Chollet is the author of Keras.
  - He currently works for Google as a deep learning engineer and researcher.
- Keras means *horn* in Greek
- Documentation: http://keras.io/
- Example: https://github.com/fchollet/keras/tree/master/examples

感謝 沈昇勳 同學提供圖檔

使用 Keras 心得

Example Application

- Handwriting Digit Recognition



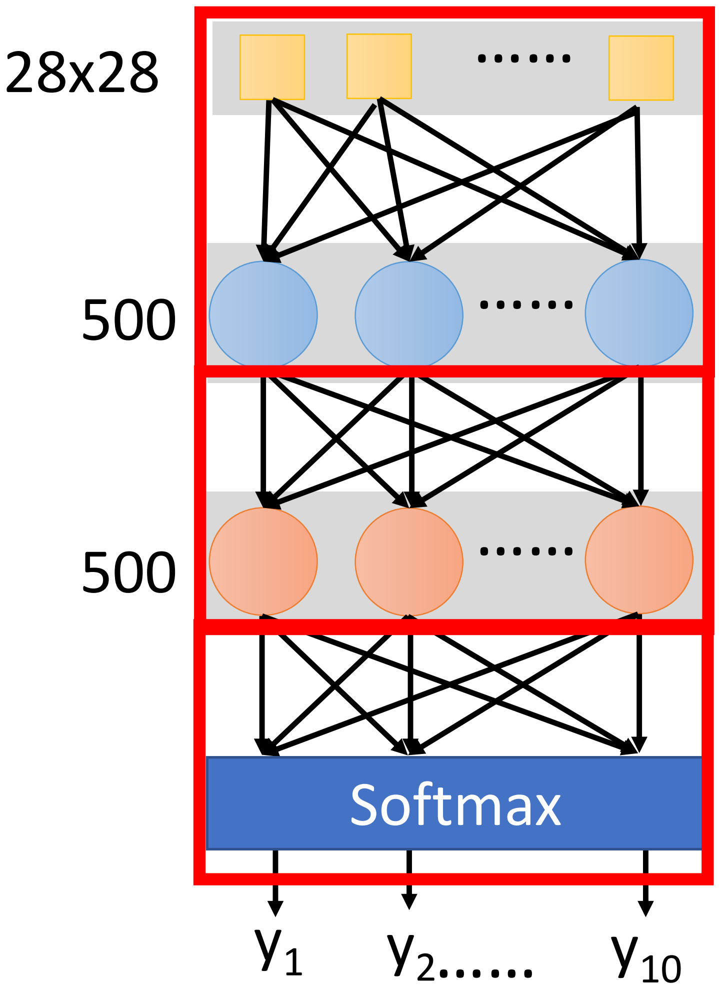
MNIST Data: http://yann.lecun.com/exdb/mnist/

“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



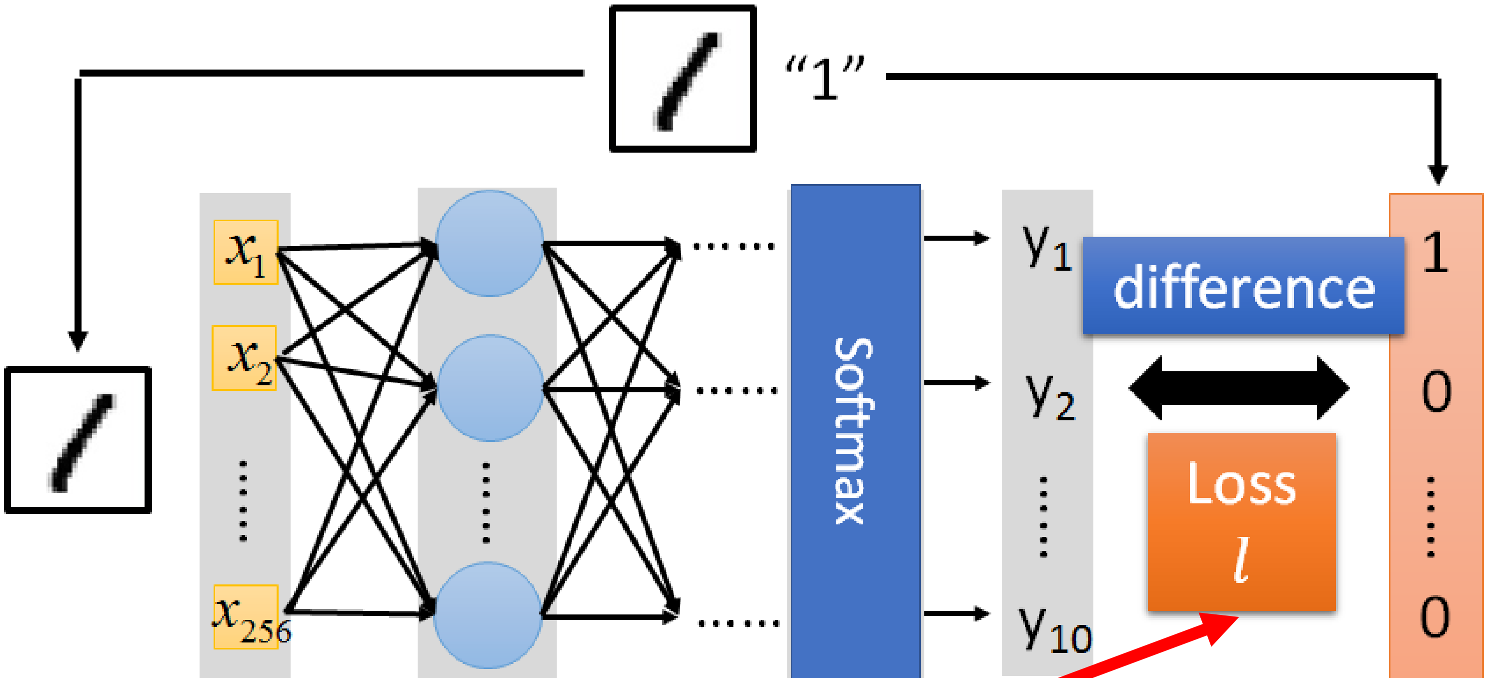

```
model = Sequential()
```

```
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




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

Keras

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

Step 3.1: Configuration

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

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

η: 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) → `x_train`

Labels (digits) → `y_train`

Keras

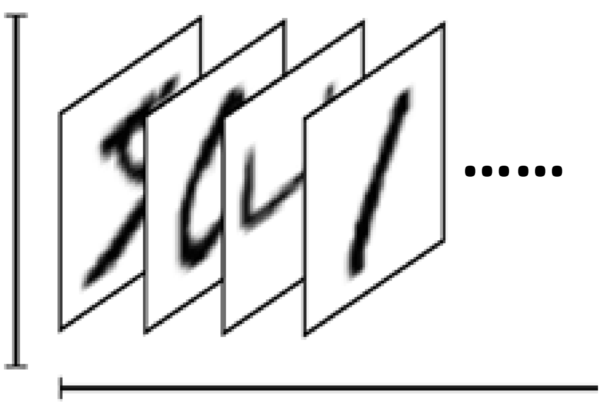
Step 1: define a set of function → **Step 2: goodness of function** → **Step 3: pick the best function**

Step 3.2: Find the optimal network parameters

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

numpy array

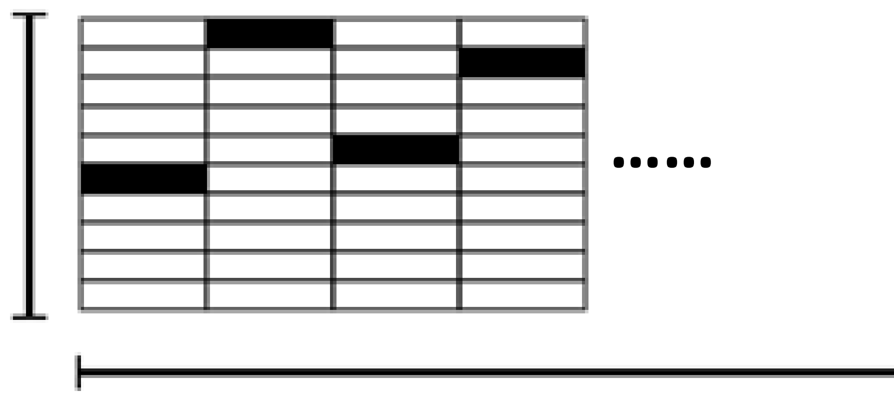
28 x 28
=784



Number of training examples

numpy array

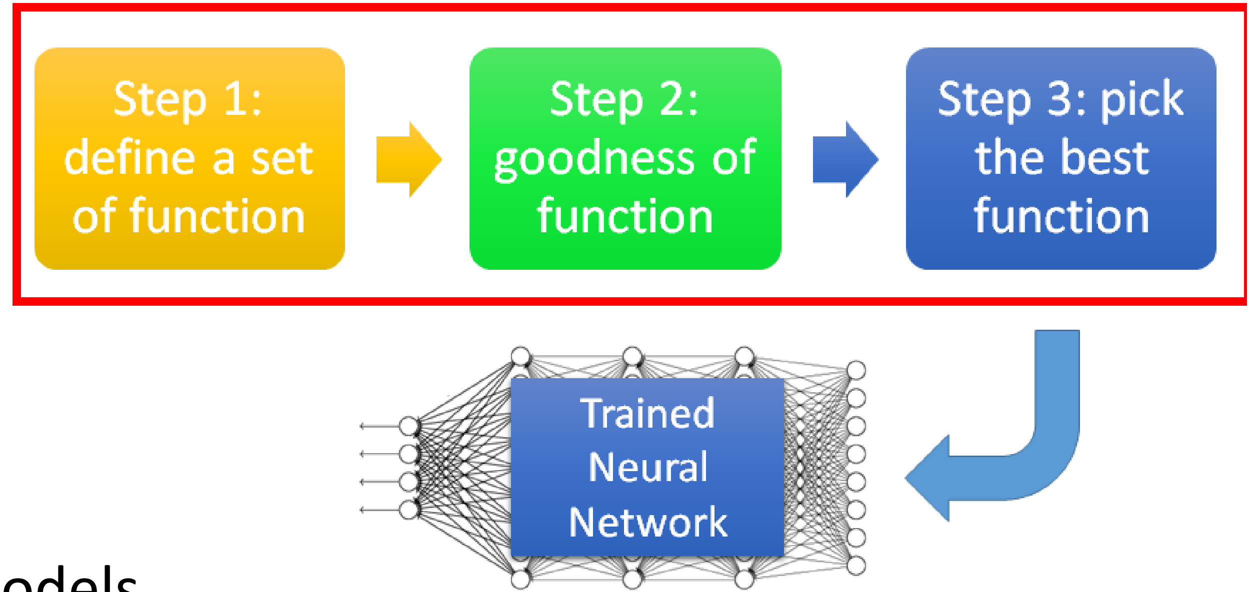
10



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

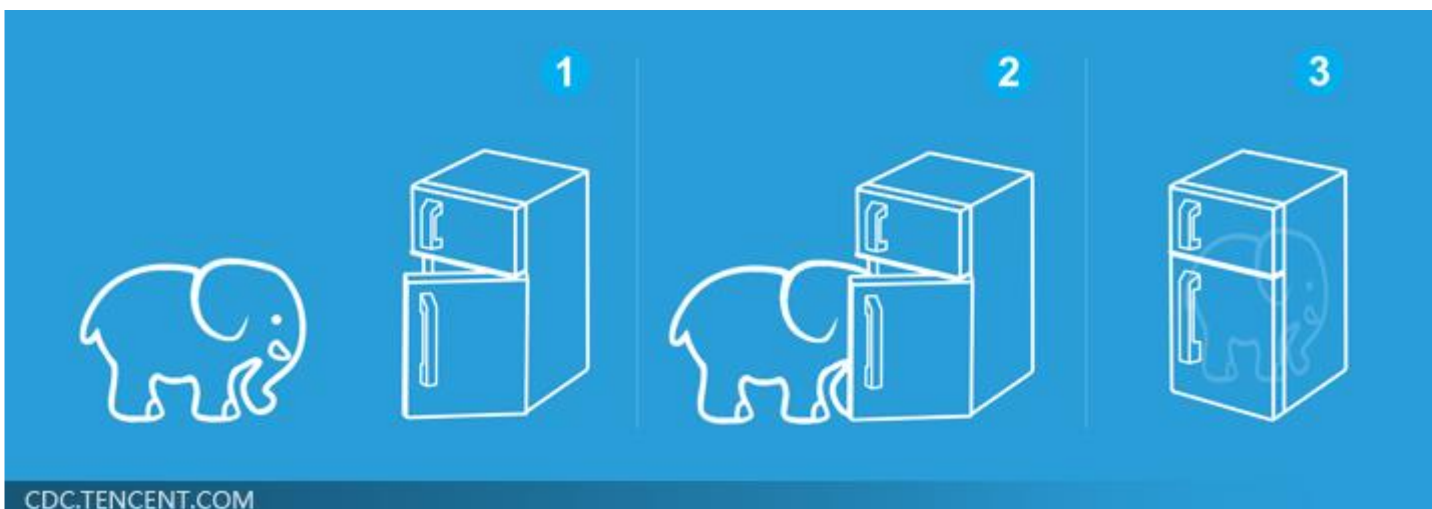
- Using GPU to speed training
  - Way 1
    - THEANO_FLAGS=device=gpu0 python YourCode.py
  - Way 2 (in your code)
    - import os
    - os.environ["THEANO_FLAGS"] = "device=gpu0"

Demo

Three Steps for Deep Learning

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

Deep Learning is so simple ……



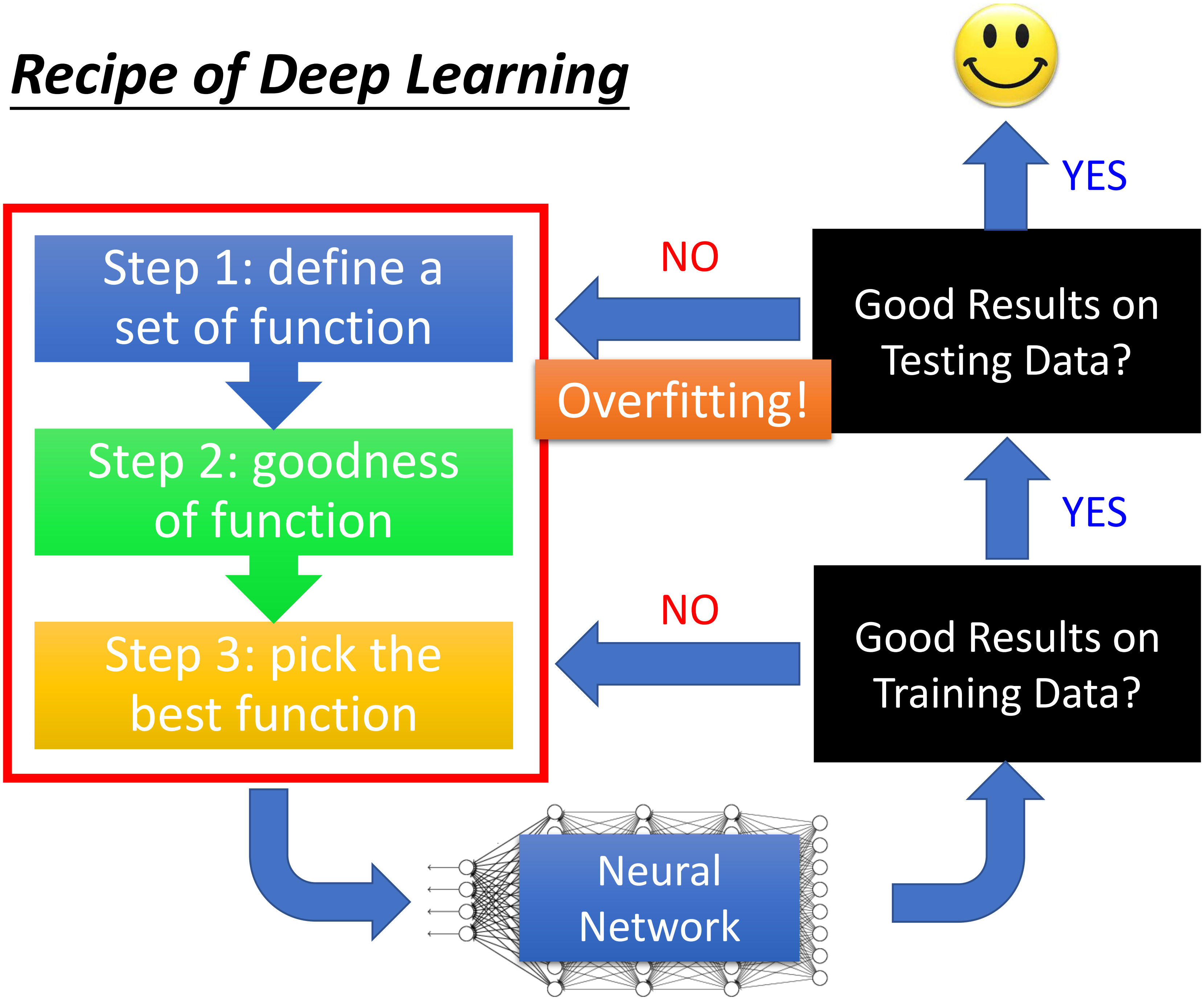
Outline

Introduction of Deep Learning

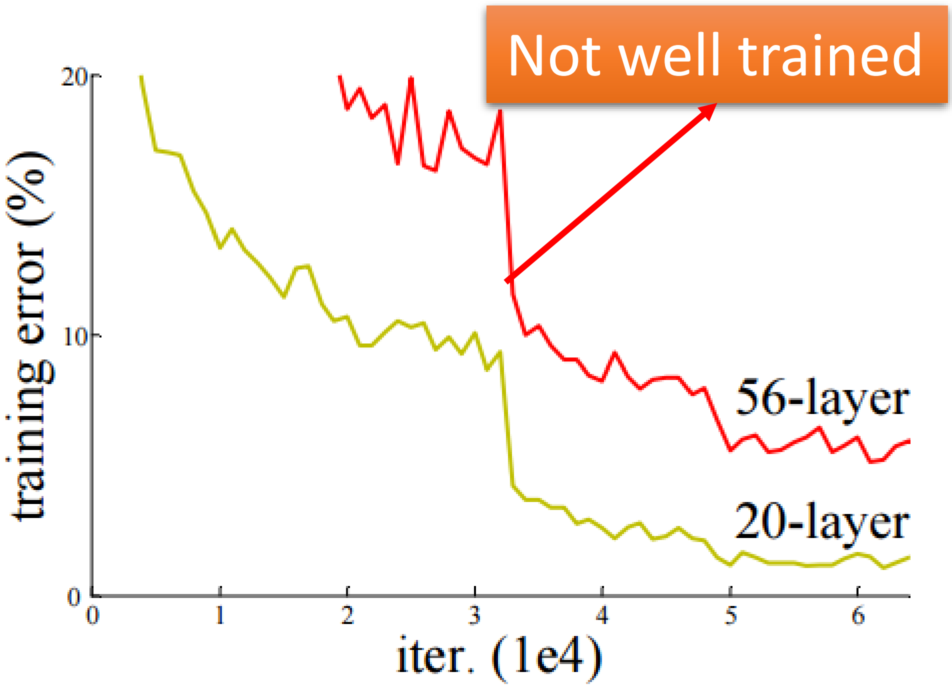
"Hello World" for Deep Learning

Tips for Deep Learning

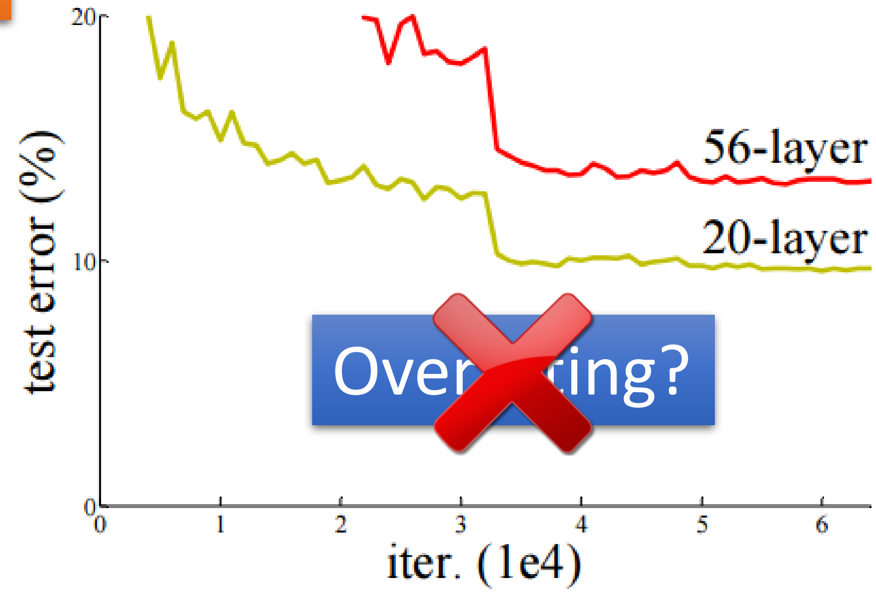
Recipe of Deep Learning



Do not always blame Overfitting



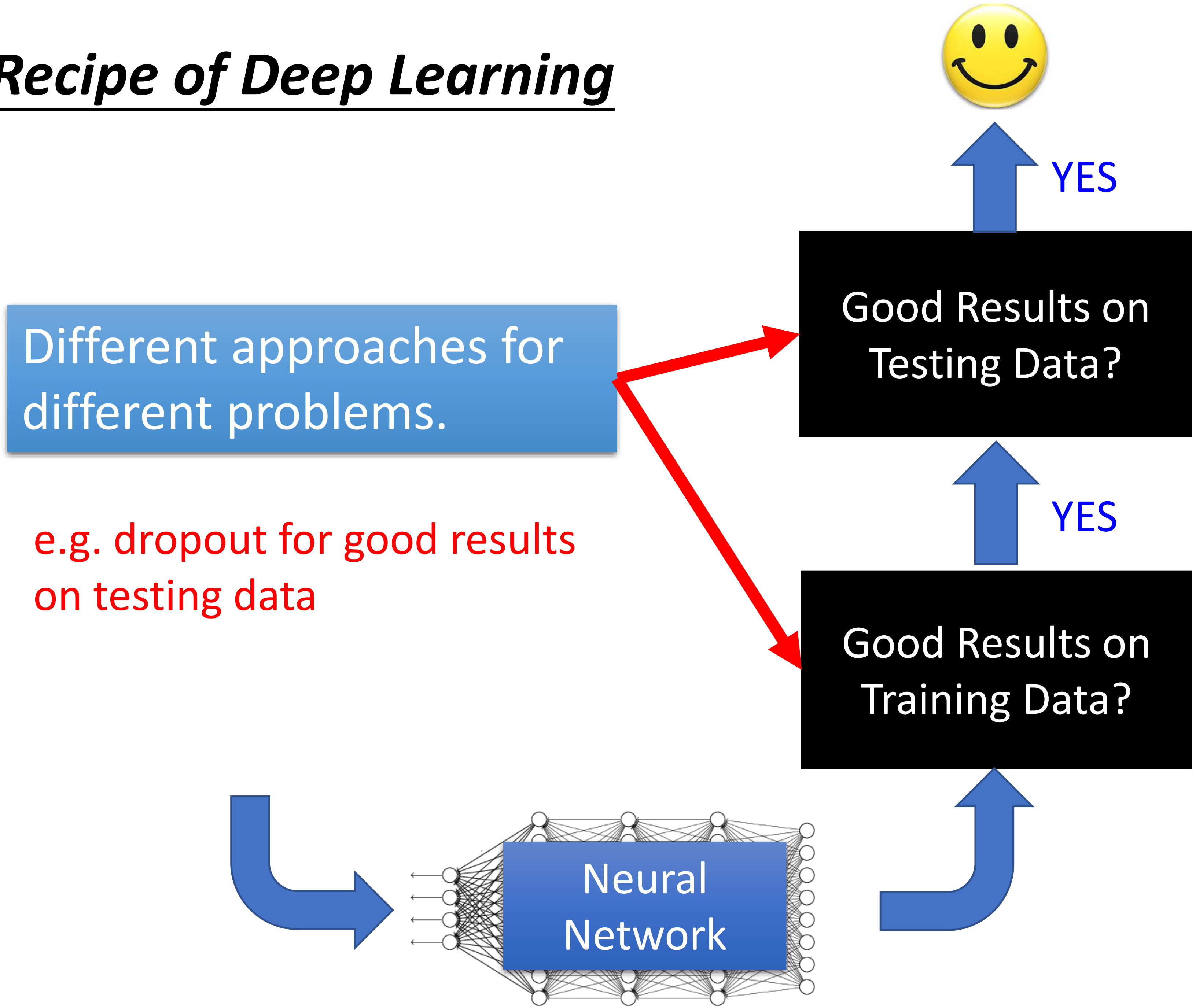
Training Data



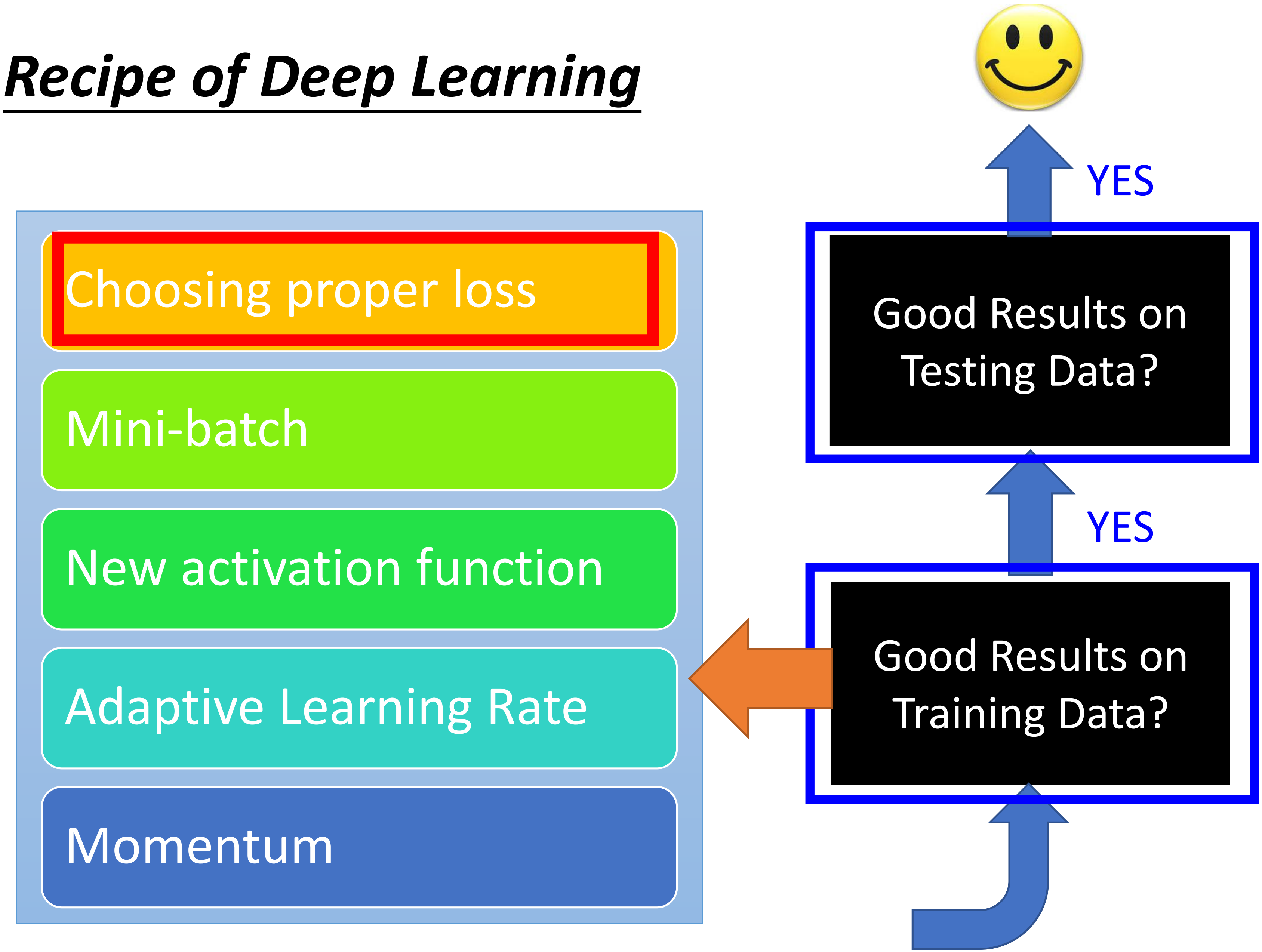
Testing Data

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

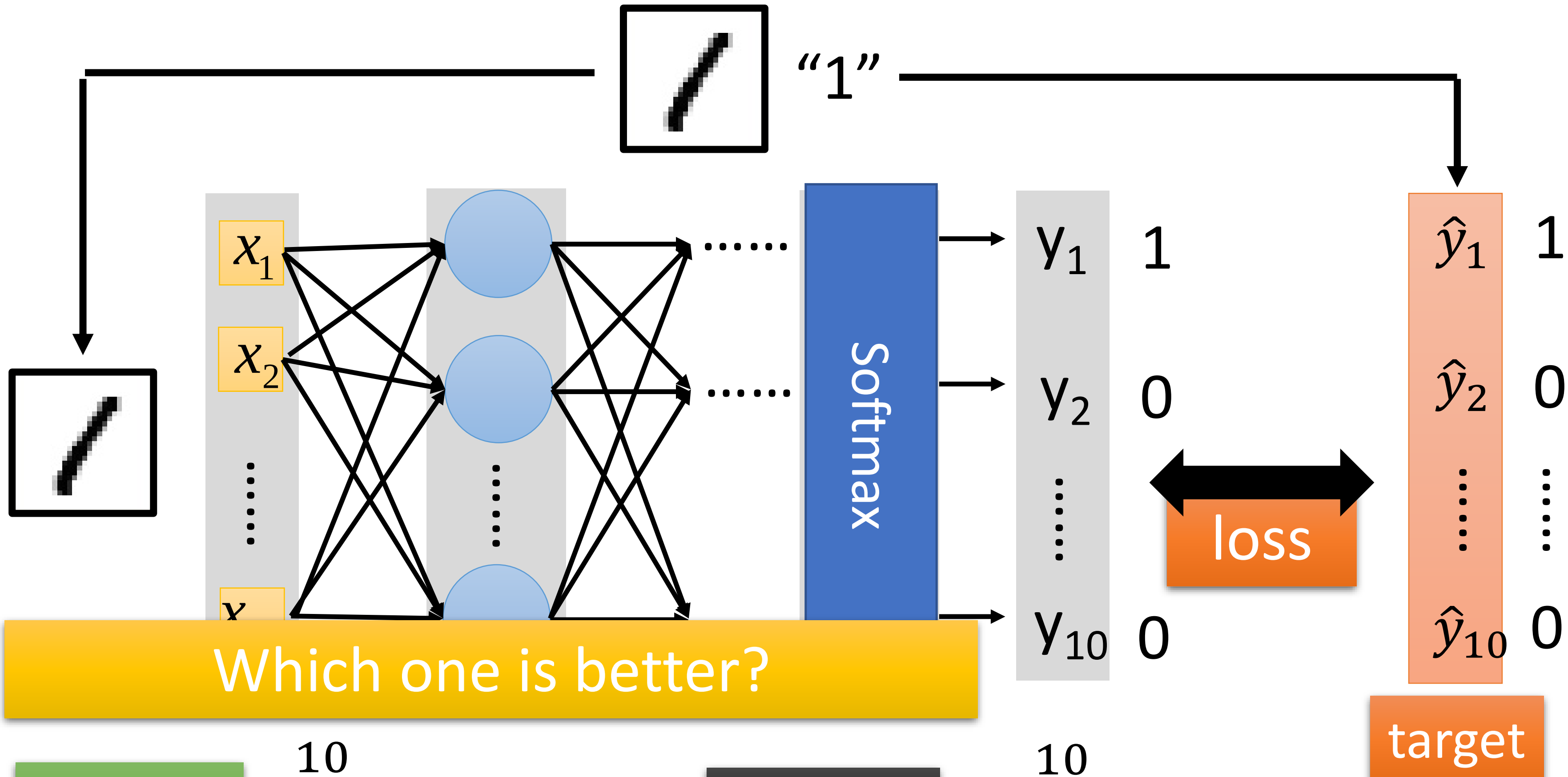
Recipe of Deep Learning



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

Demo

Square Error

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

Cross Entropy

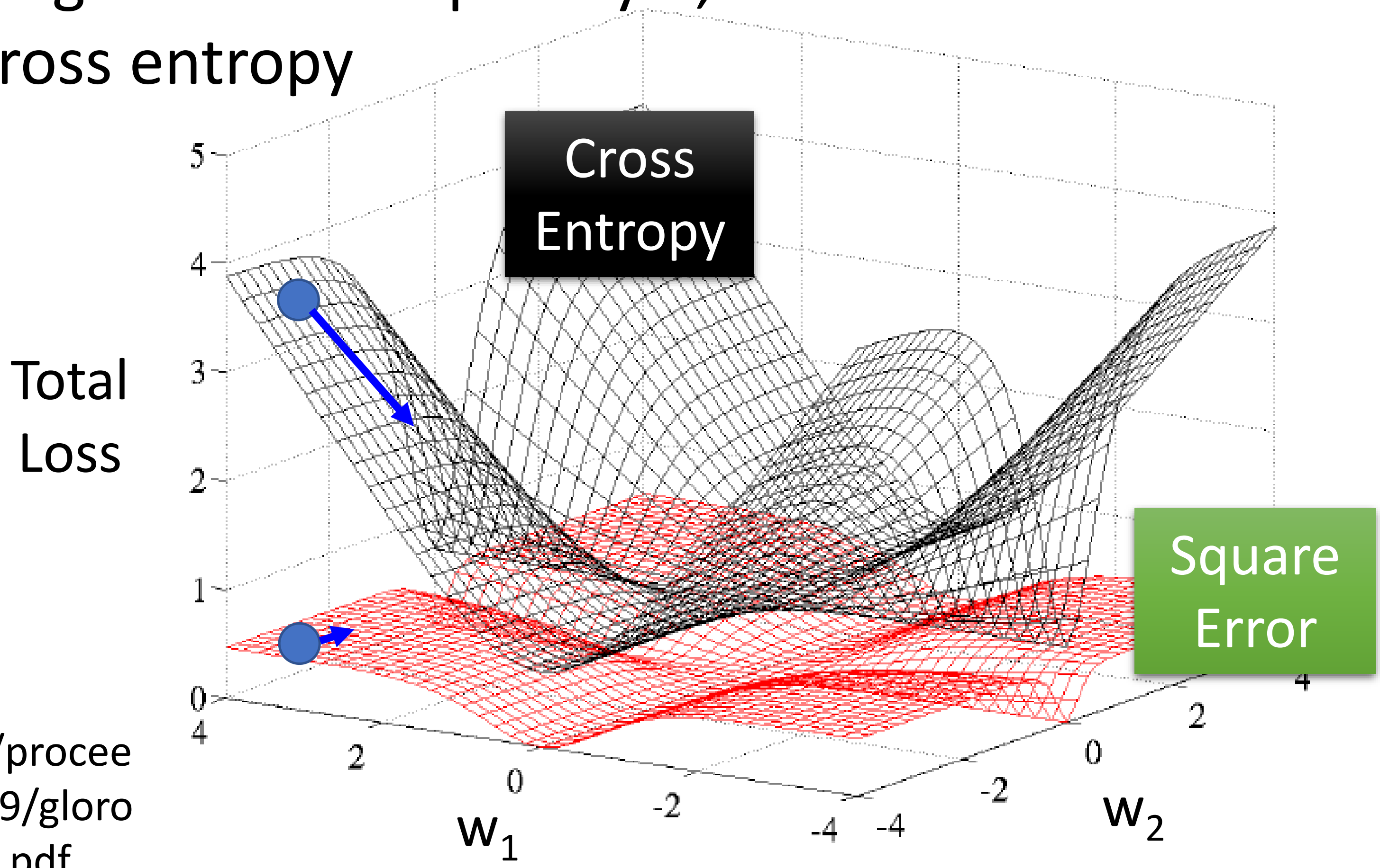
```
model.compile(loss='categorical_crossentropy',
                          optimizer=SGD(lr=0.1),
                          metrics=['accuracy'])
```

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

Demo

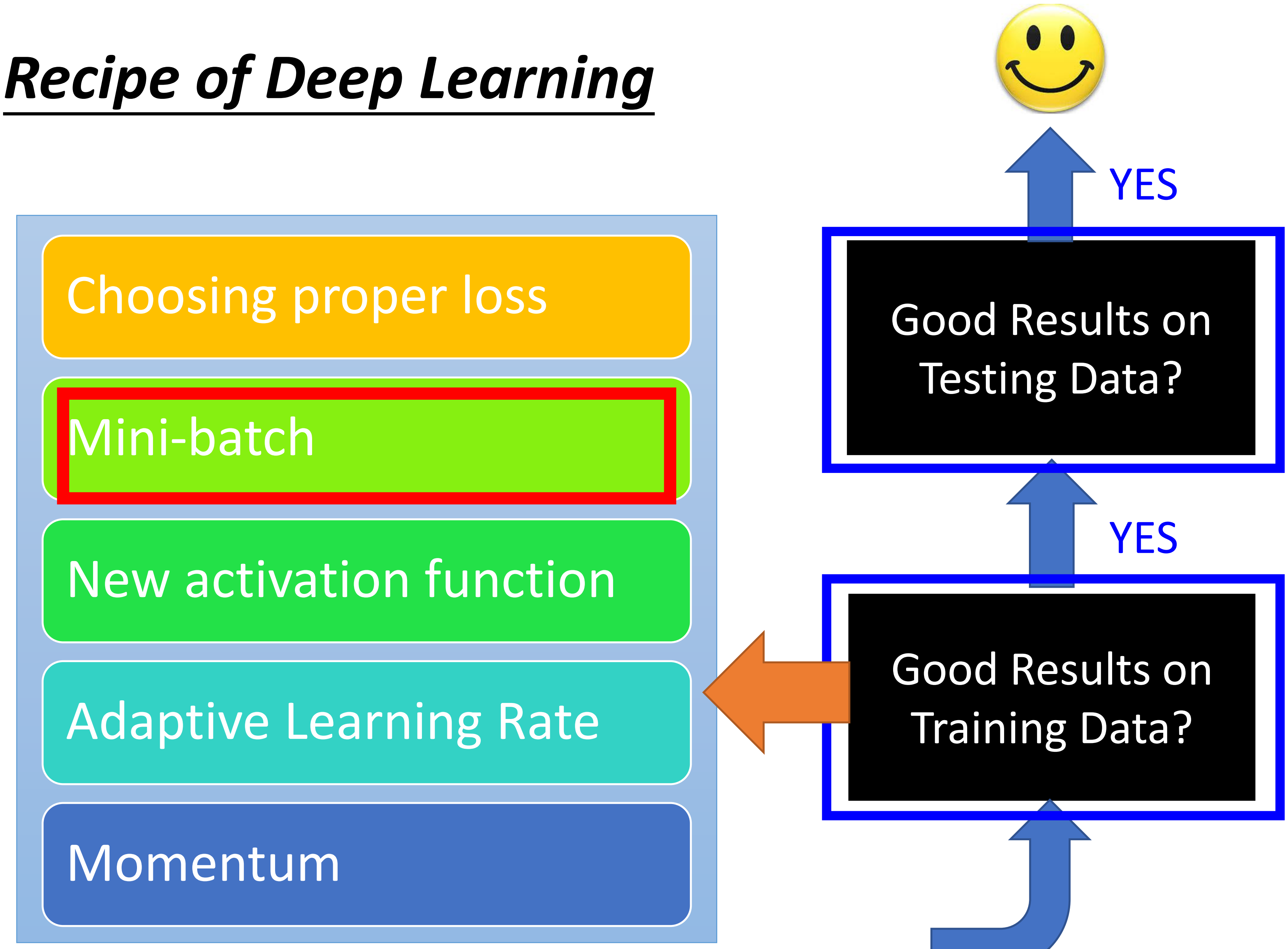
Choosing Proper Loss

When using softmax output layer, choose cross entropy



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

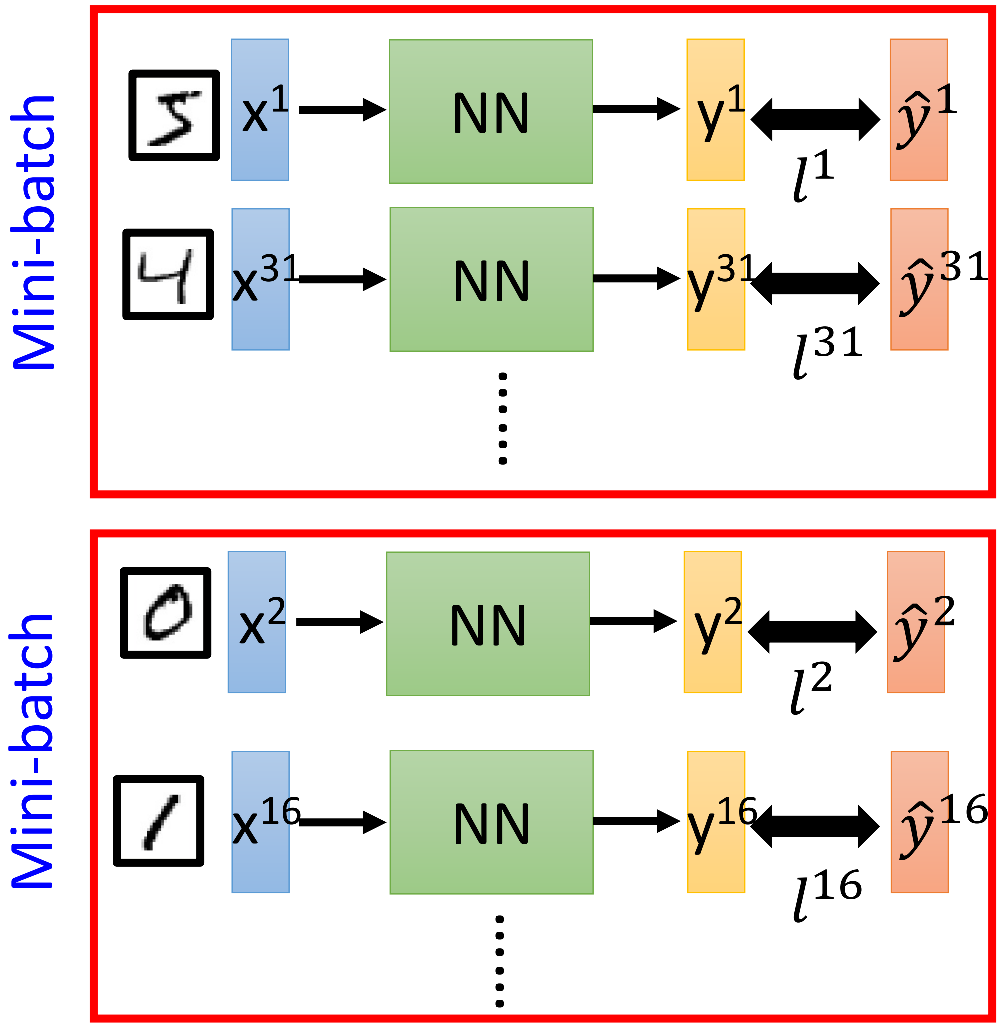
Recipe of Deep Learning




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

We do not really minimize total loss!

Mini-batch



- Randomly initialize network parameters
- Pick the 1st batch
  $L' = l^1 + l^{31} + \cdots$
  Update parameters once
- Pick the 2nd batch
  $L'' = l^2 + l^{16} + \cdots$
  Update parameters once

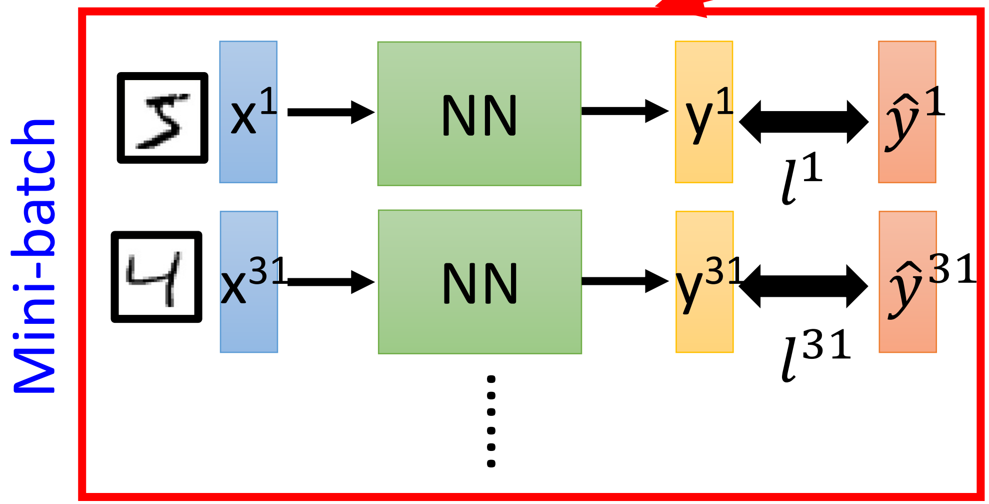
  ⋮
- Until all mini-batches have been picked

one epoch

Repeat the above process

Mini-batch

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



100 examples in a mini-batch

- Pick the 1st batch

  $L' = l^1 + l^{31} + \cdots$

  Update parameters once

- Pick the 2nd batch

  $L'' = l^2 + l^{16} + \cdots$

  Update parameters once

  ⋮

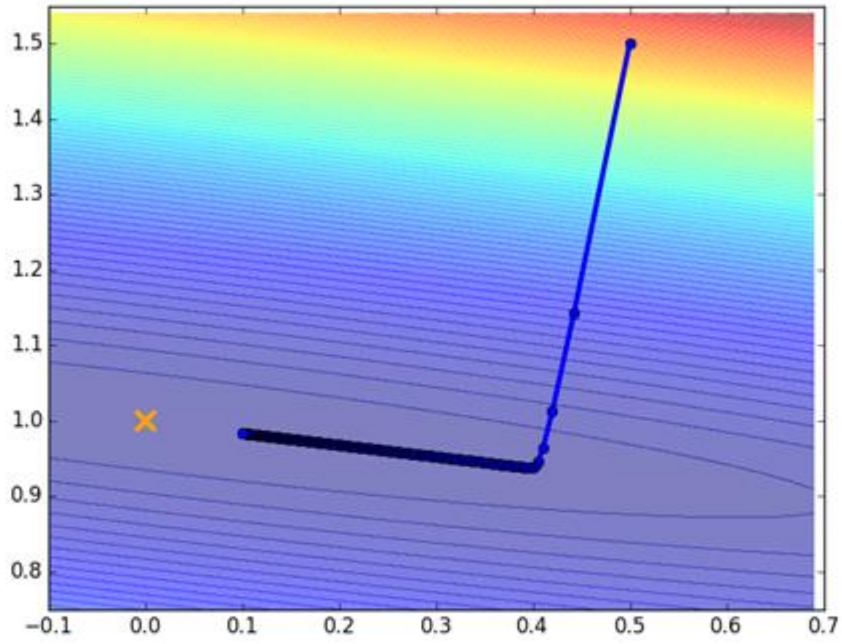
- Until all mini-batches have been picked

one epoch

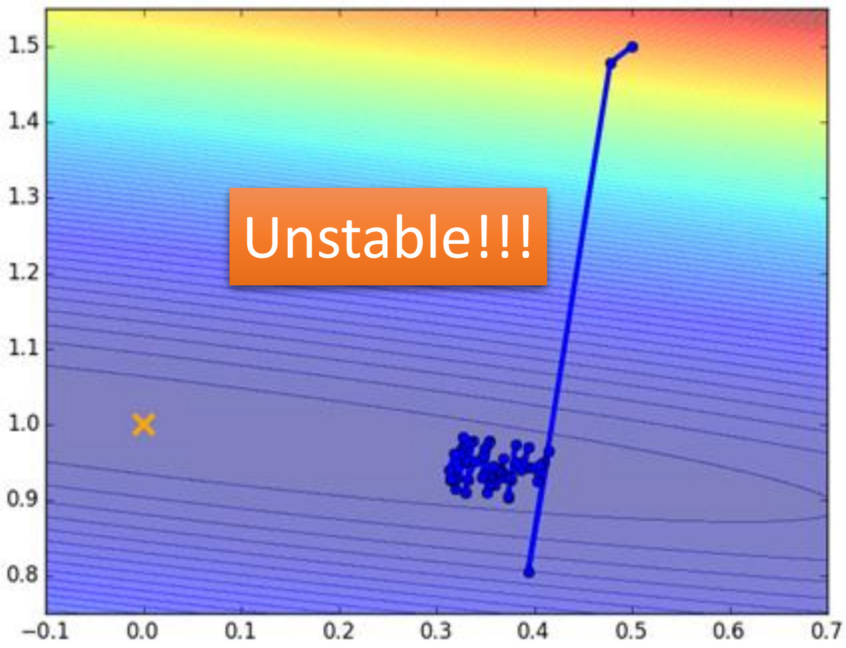
Repeat 20 times

Mini-batch

Original Gradient Descent



With Mini-batch



The colors represent the total loss.

Mini-batch is Faster

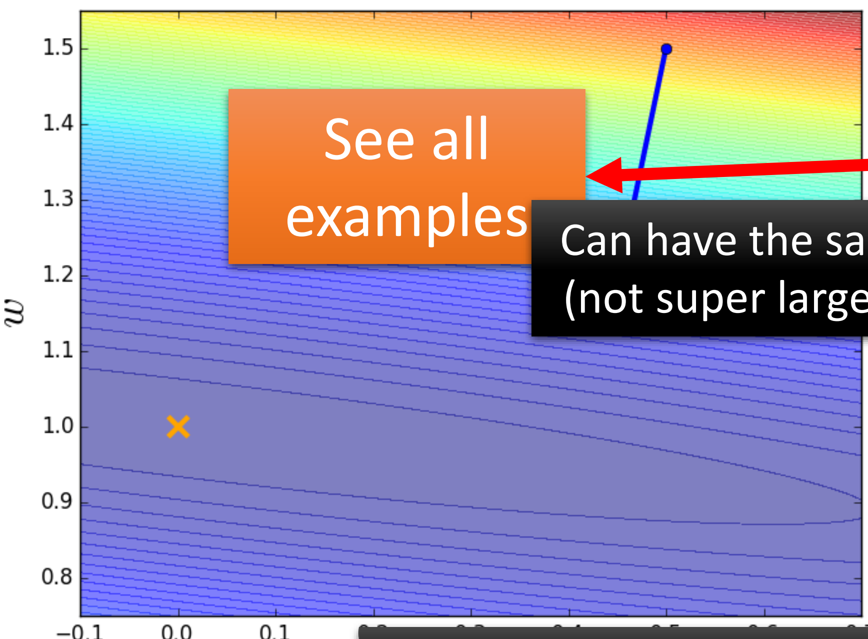
Not always true with parallel computing.

Original Gradient Descent

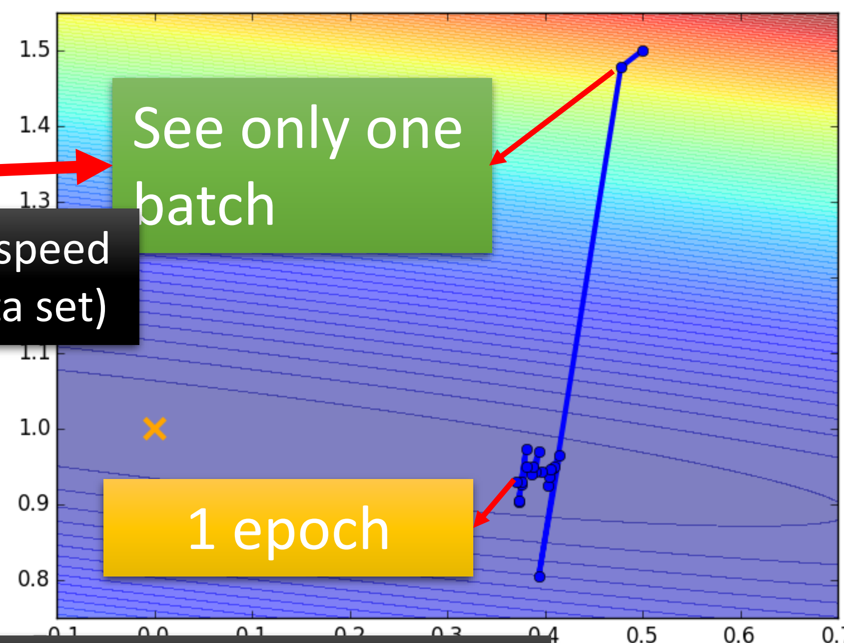
Update after seeing all examples

With Mini-batch

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



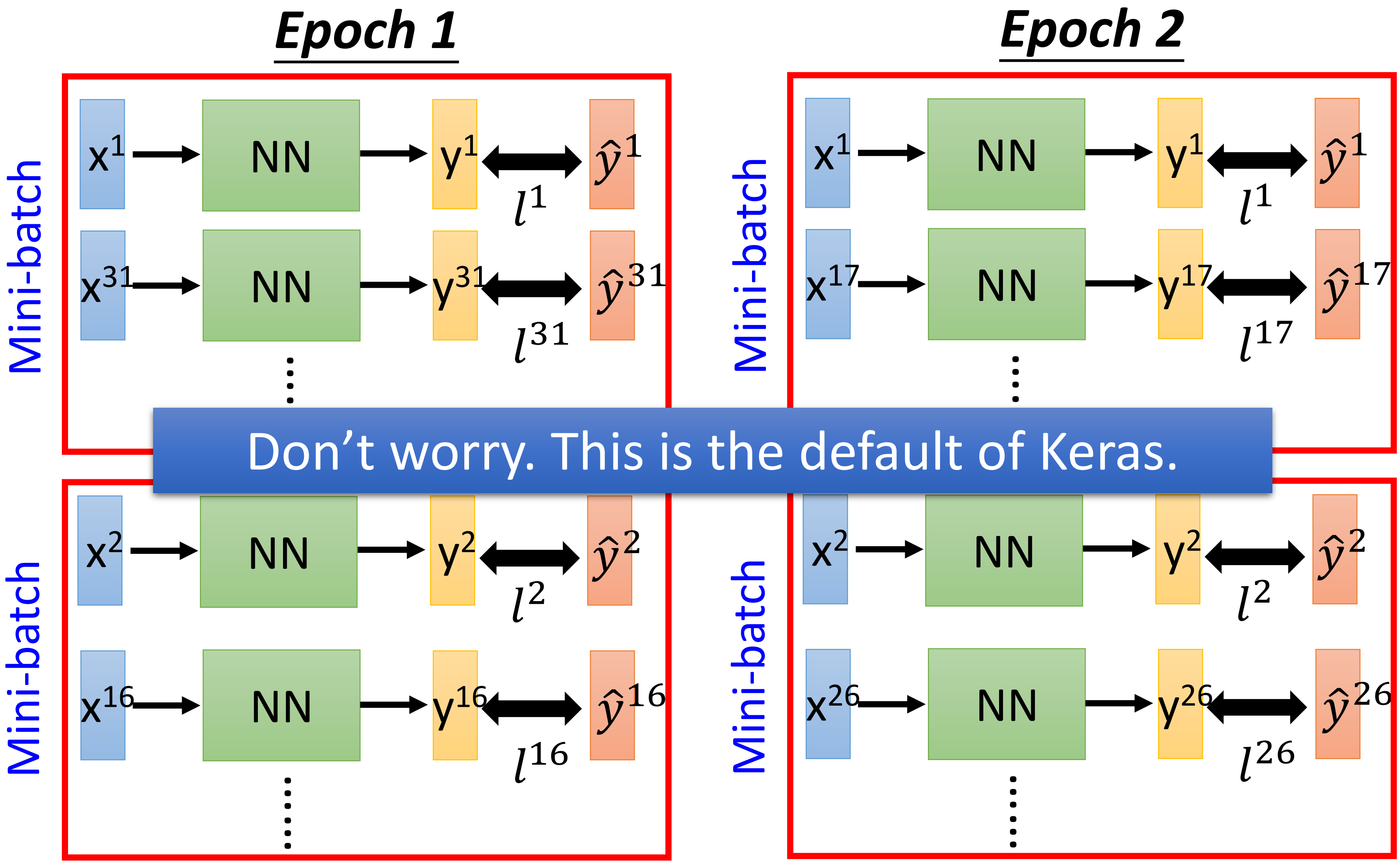
Can have the same speed (not super large data set)



Mini-batch has better performance!

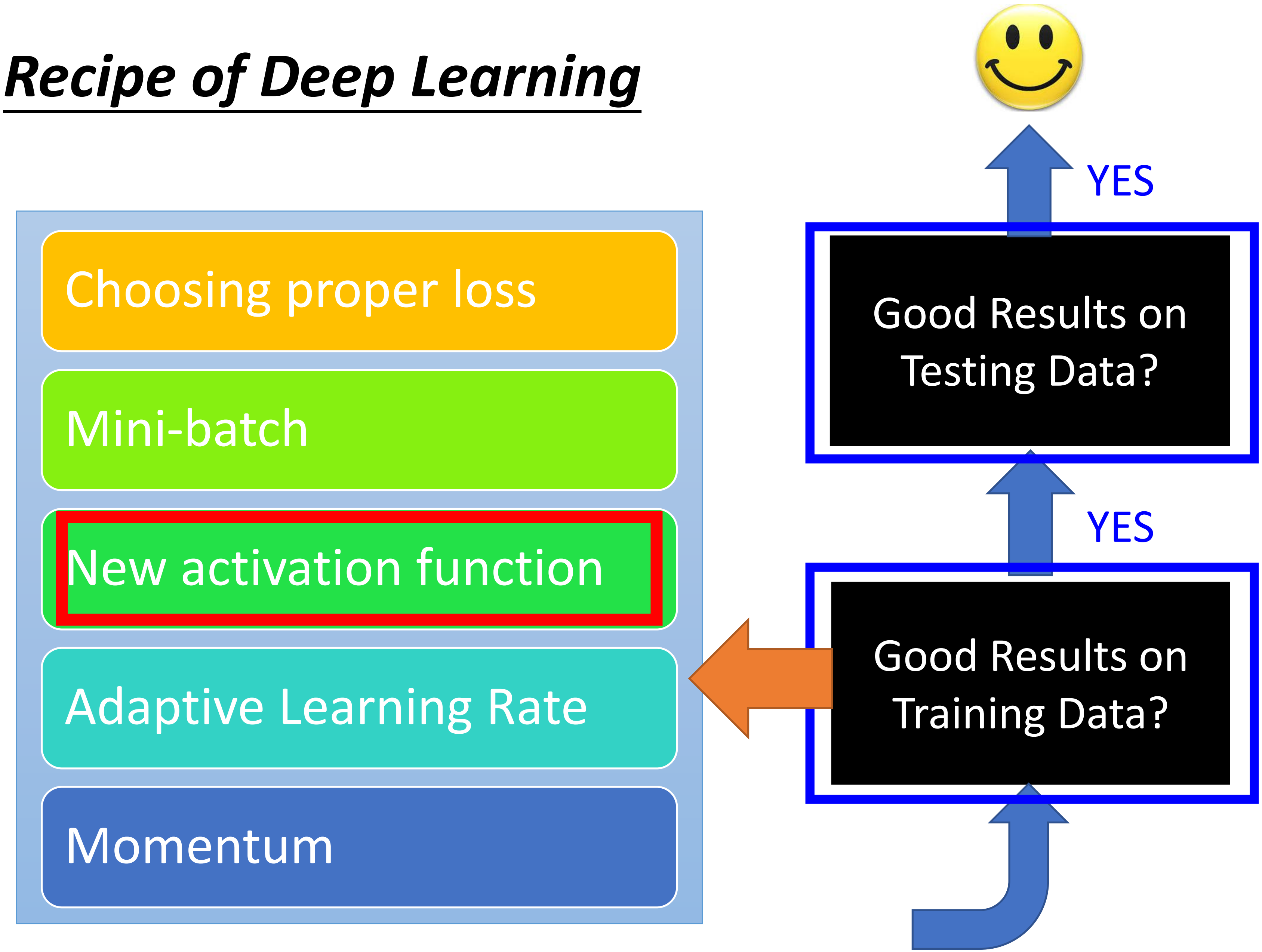
Demo

Shuffle the training examples for each epoch

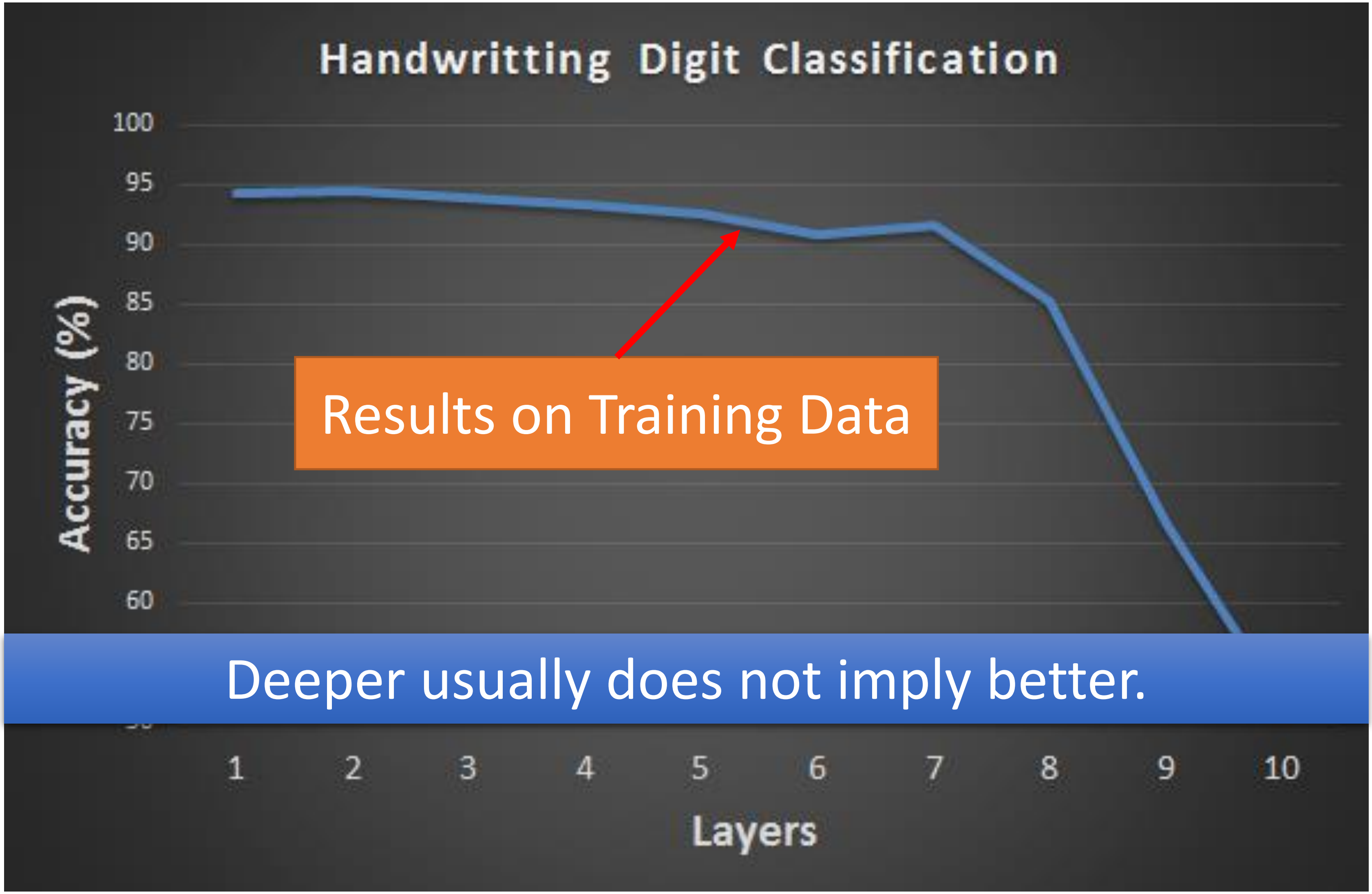


Don’t worry. This is the default of Keras.

Recipe of Deep Learning

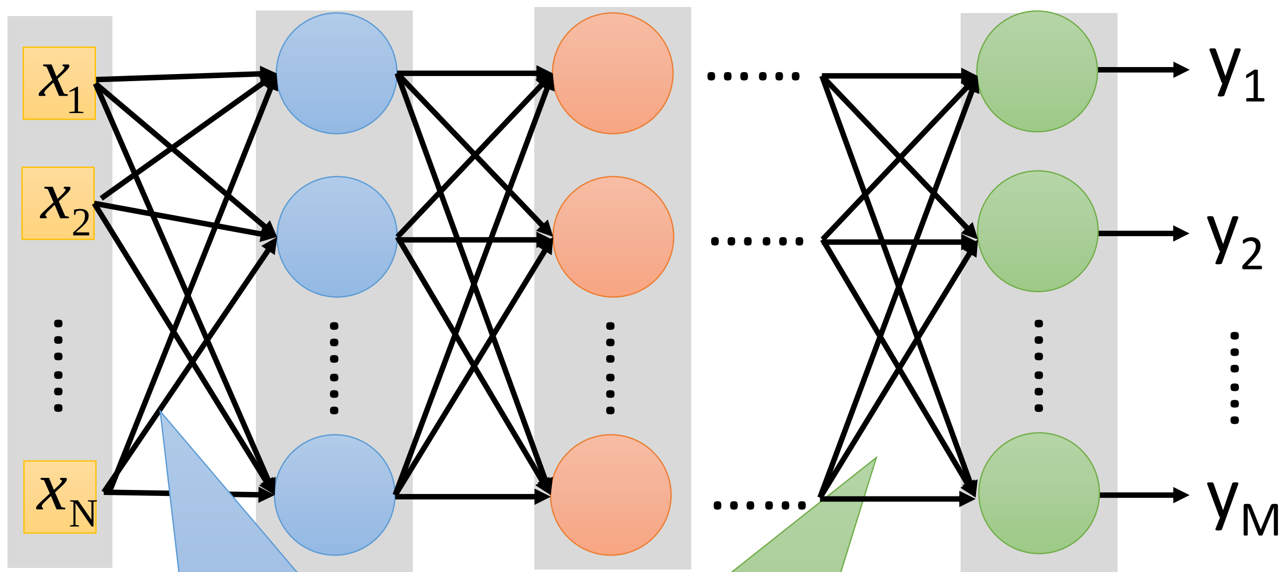


Hard to get the power of Deep …



Demo

Vanishing Gradient Problem



Smaller gradients

Learn very slow

Almost random

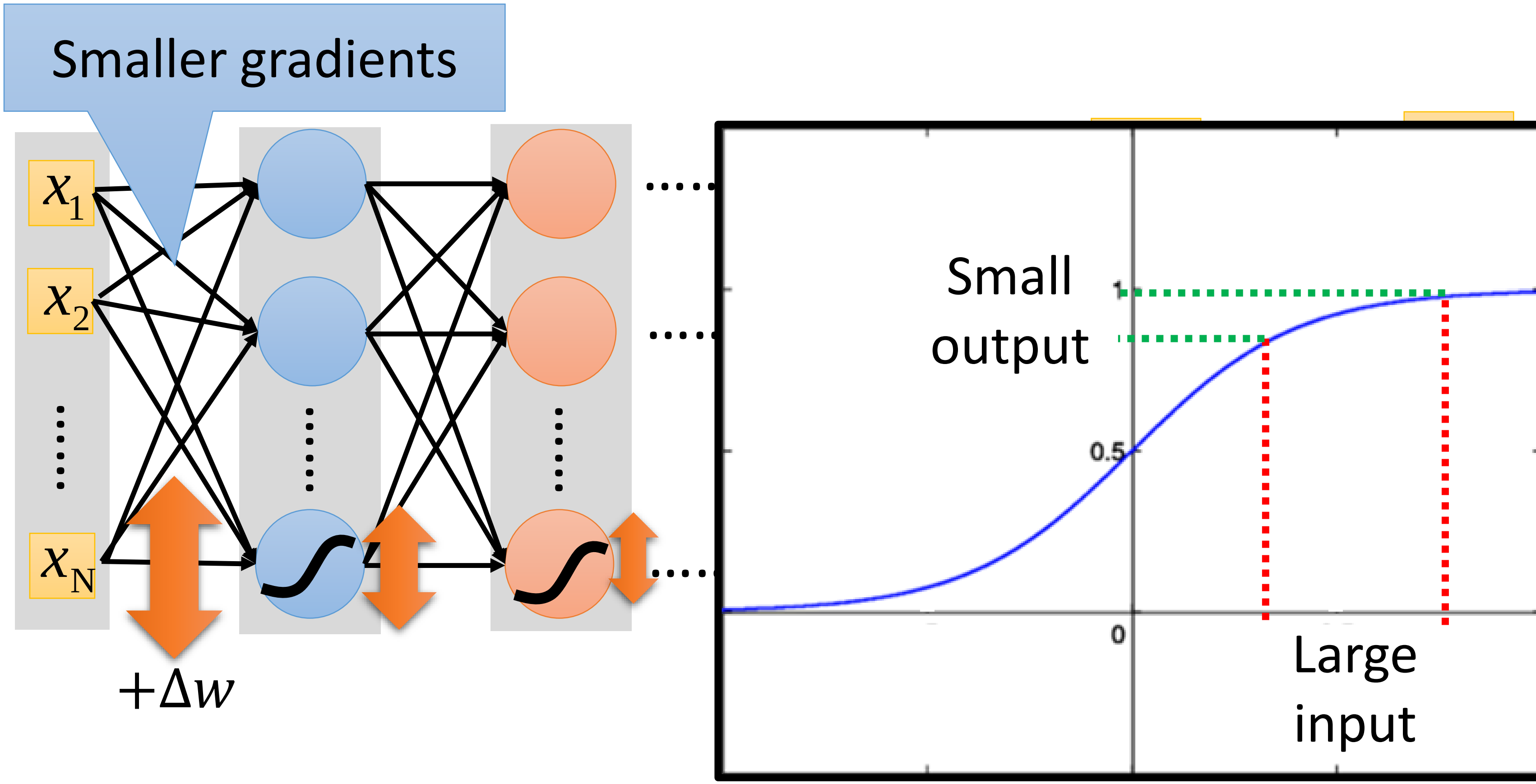
Larger gradients

Learn very fast

Already converge

based on random!?

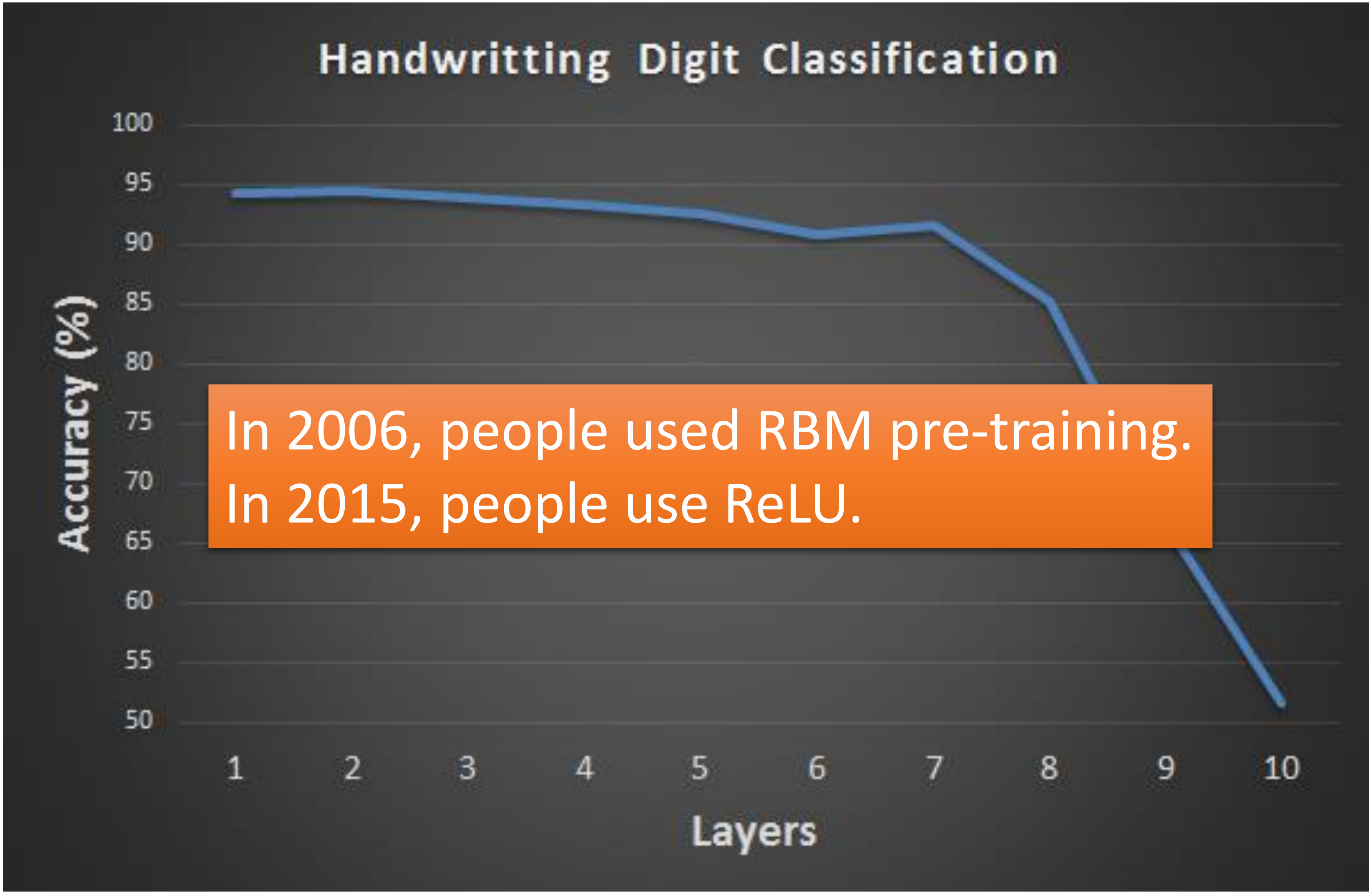
Vanishing Gradient Problem



Intuitive way to compute the derivatives …

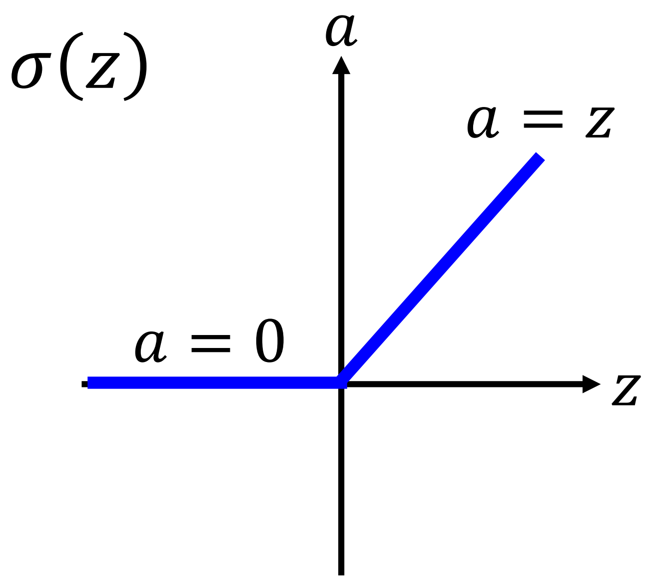
$$\frac{\partial l}{\partial w} =? \frac{\Delta l}{\Delta w}$$

Hard to get the power of Deep …



ReLU

- Rectified Linear Unit (ReLU)

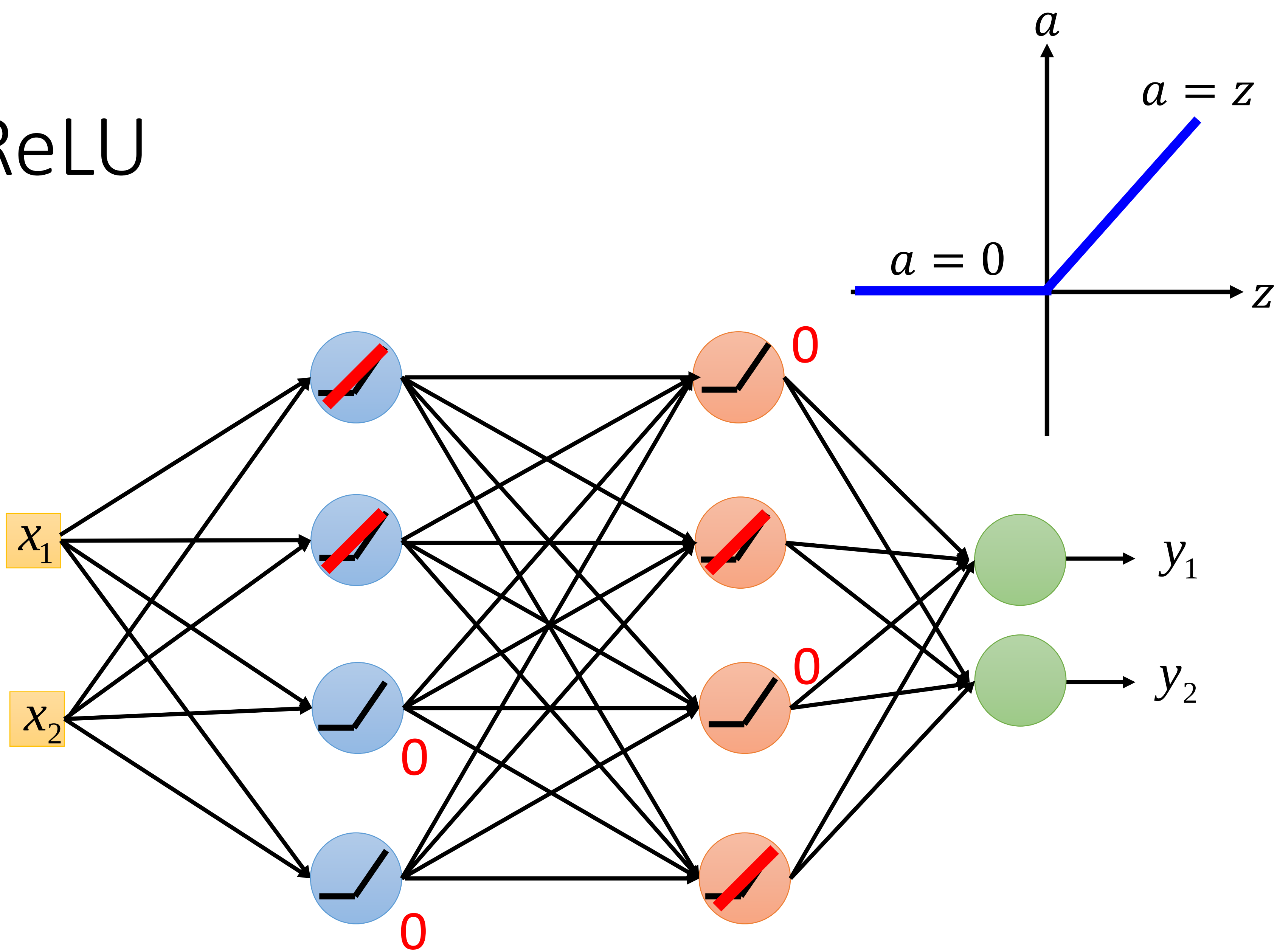


[Xavier Glorot, AISTATS'11]
[Andrew L. Maas, ICML'13]
[Kaiming He, arXiv'15]

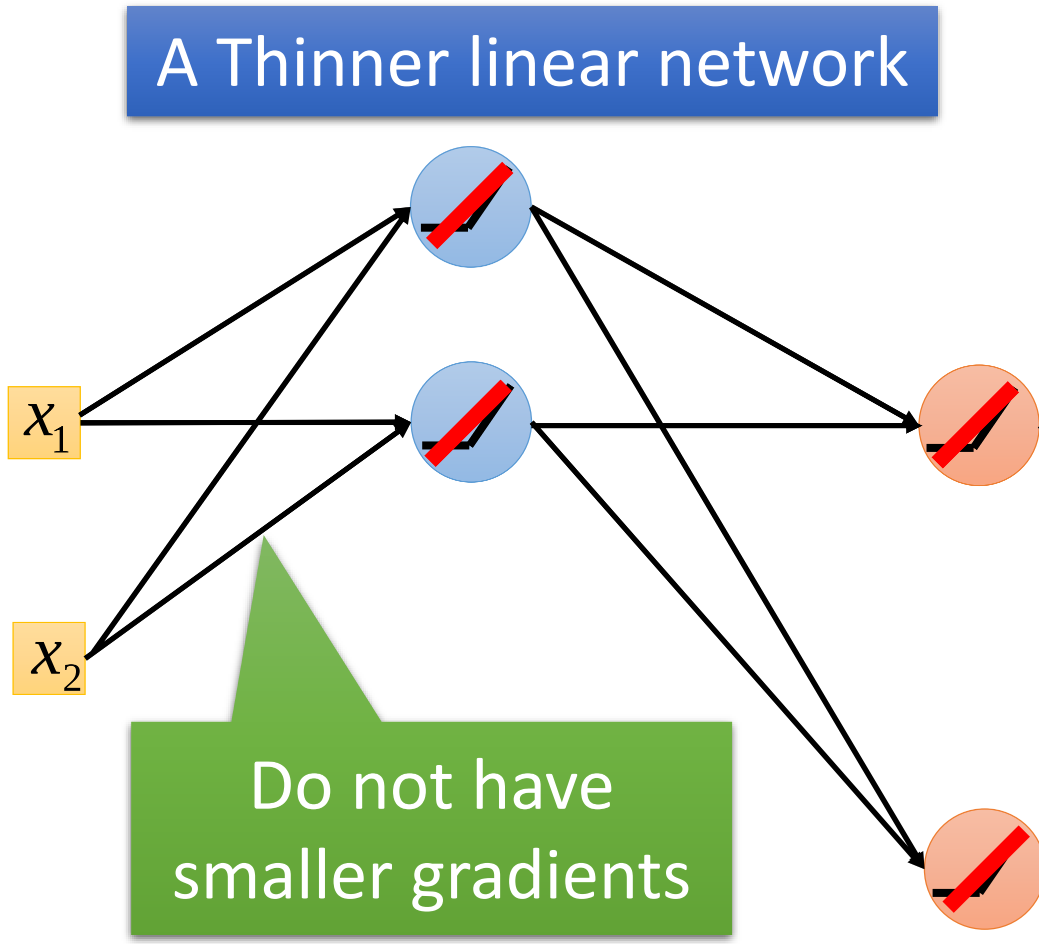
Reason:

1. Fast to compute
2. Biological reason
3. Infinite sigmoid with different biases
4. $\boxed{\text{Vanishing gradient problem}}$

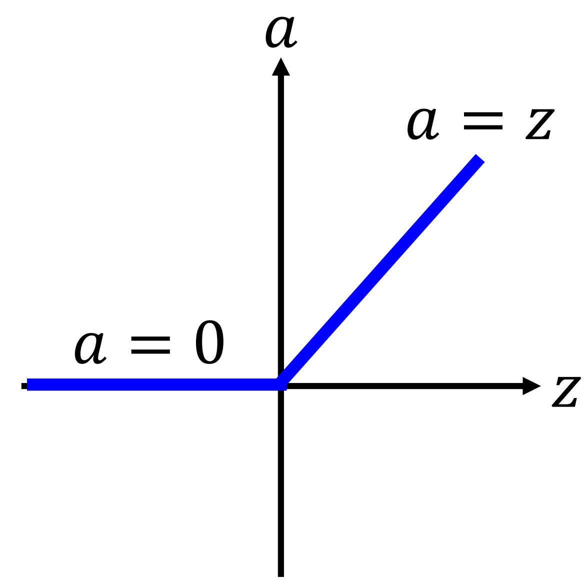
ReLU



ReLU



A Thinner linear network
x_1
x_2
y_1
y_2
Do not have smaller gradients

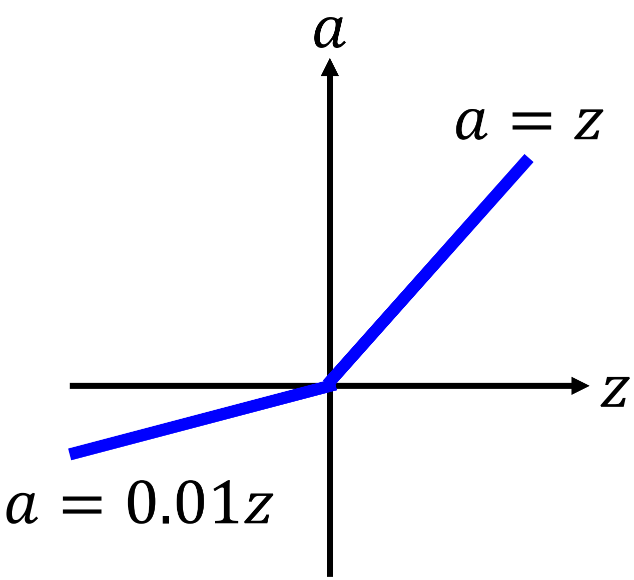


a
$a = z$
$a = 0$
z

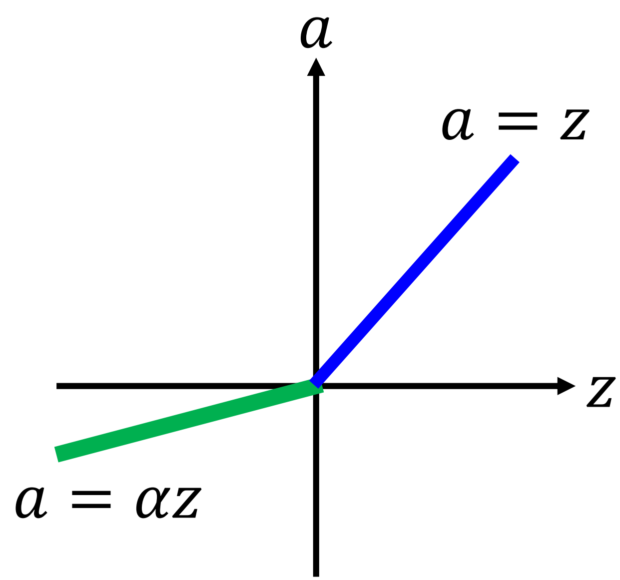
Demo

ReLU - variant

$Leaky\ ReLU$



$Parametric\ ReLU$

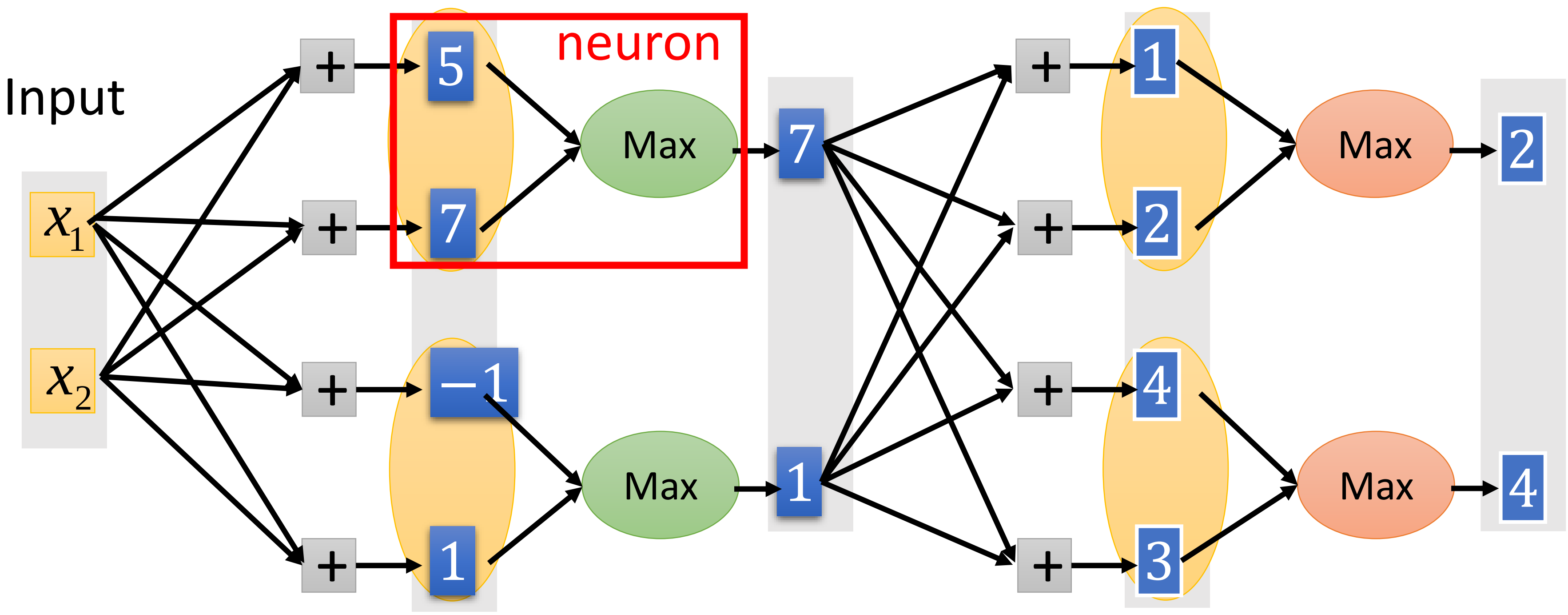


α also learned by gradient descent

Maxout

ReLU is a special cases of Maxout

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



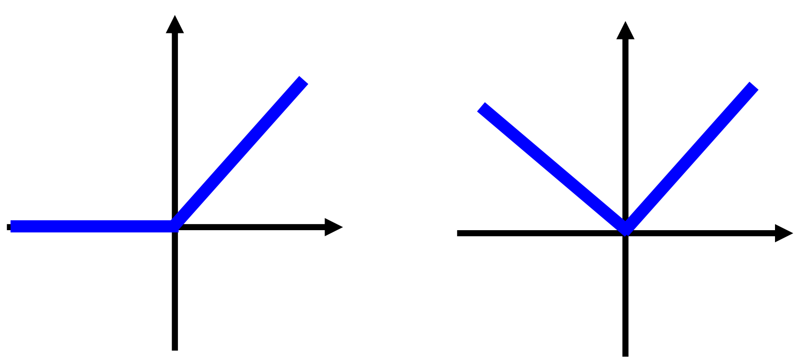
You can have more than 2 elements in a group.

Maxout

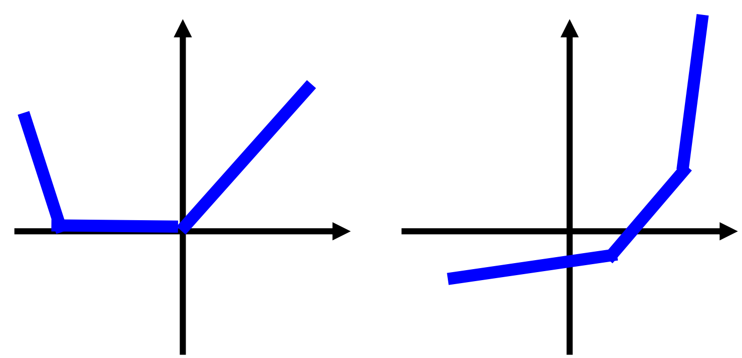
ReLU is a special cases of Maxout

- 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

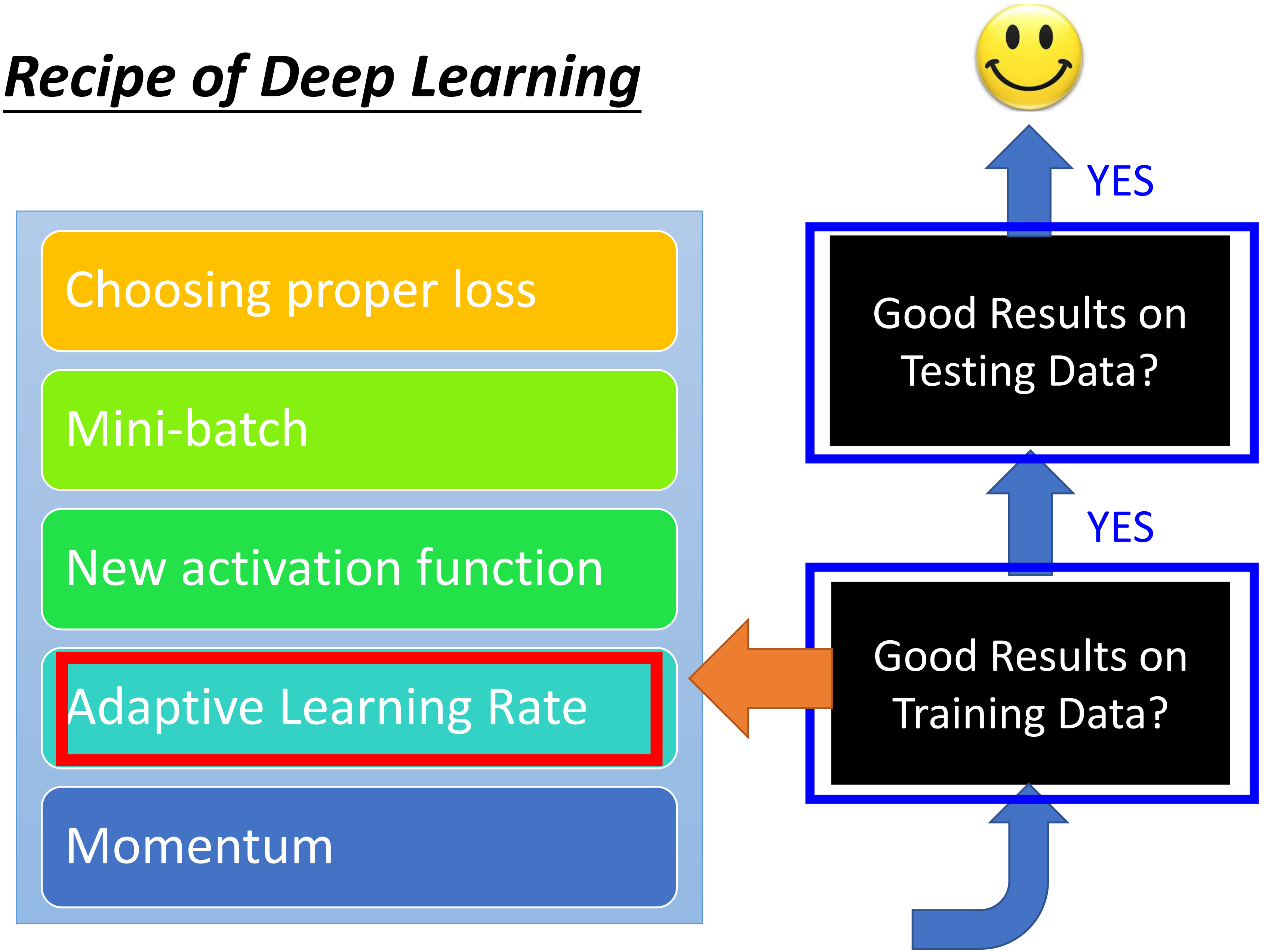
2 elements in a group



3 elements in a group

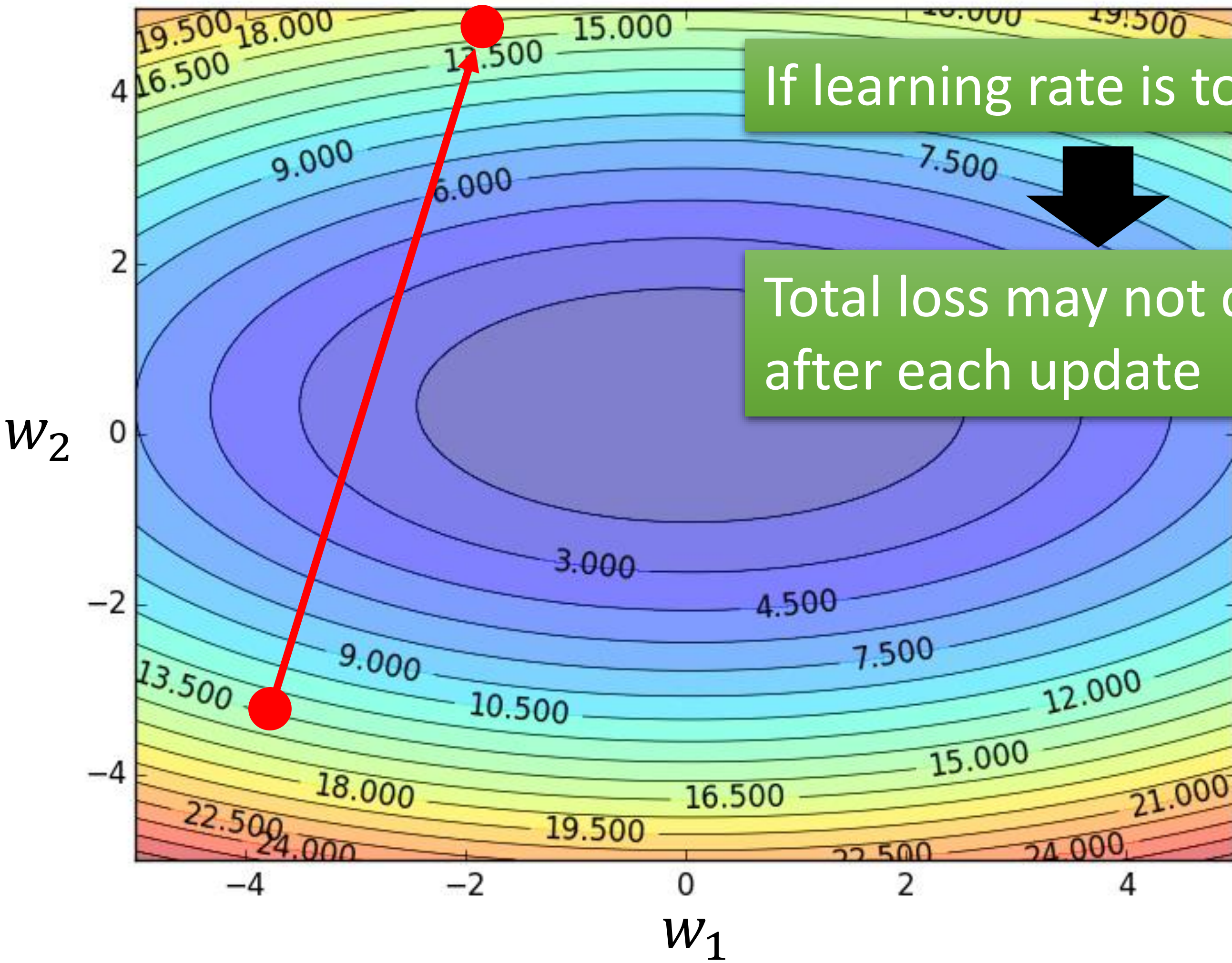


Recipe of Deep Learning



Learning Rates

Set the learning rate η carefully

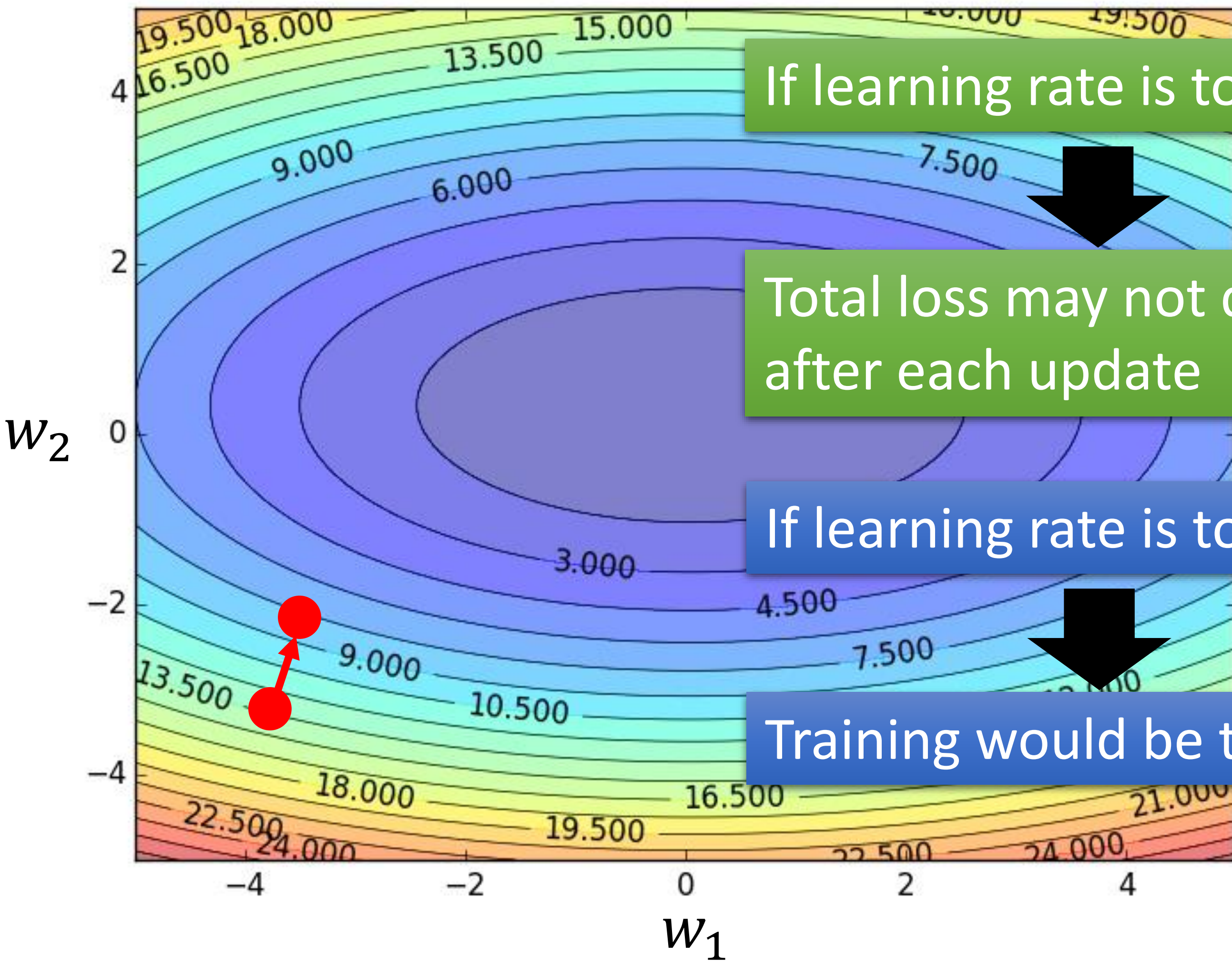


If learning rate is too large

Total loss may not decrease after each update

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: $\mathrm{w} \leftarrow w - \boxed{\eta_w} \partial L \,/\, \partial w$

Parameter dependent learning rate

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

constant

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

Summation of the square of the previous derivatives

Adagrad

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

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:

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

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

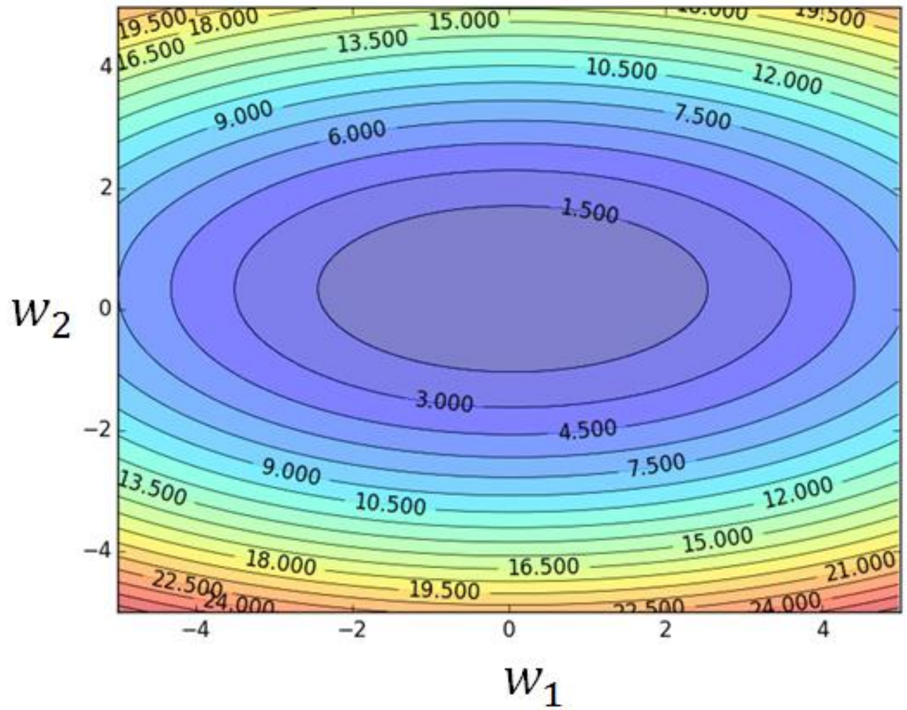
Observation: 1. Learning rate is smaller and smaller for all parameters

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

Why?

Larger derivatives

Smaller Learning Rate



Smaller Derivatives

Larger Learning Rate

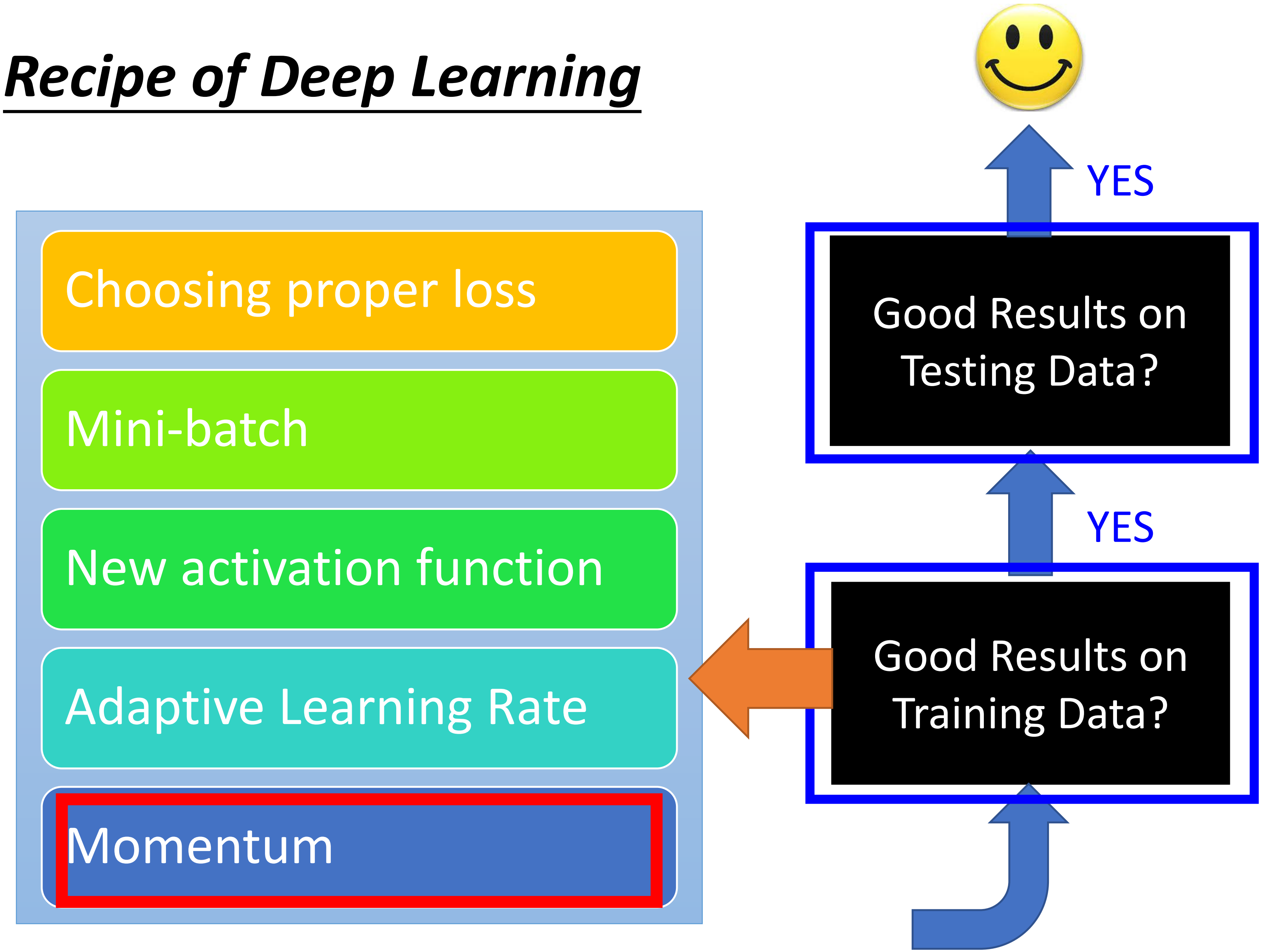
2. Smaller derivatives, larger learning rate, and vice versa

Why?

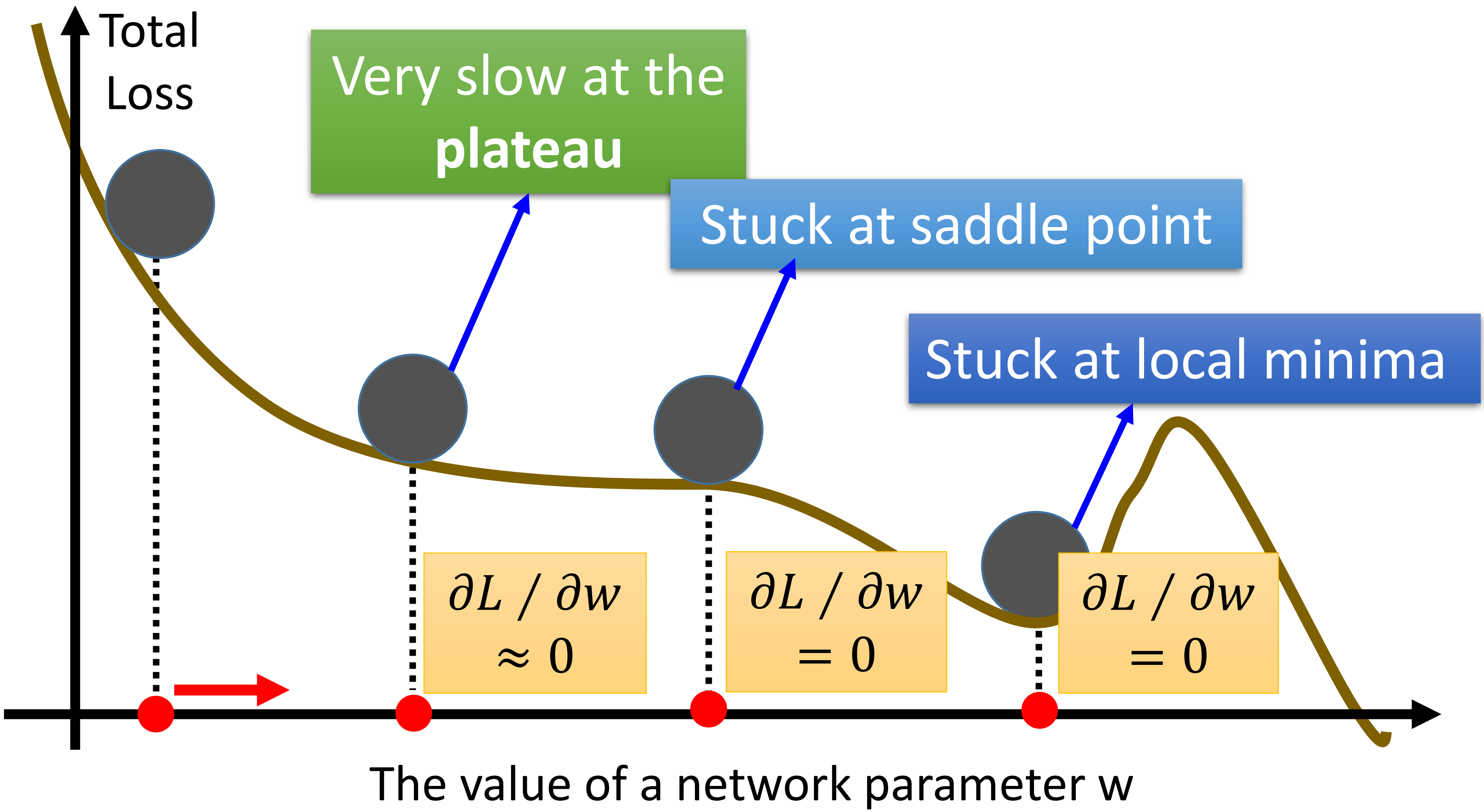
Not the whole story ……

- 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

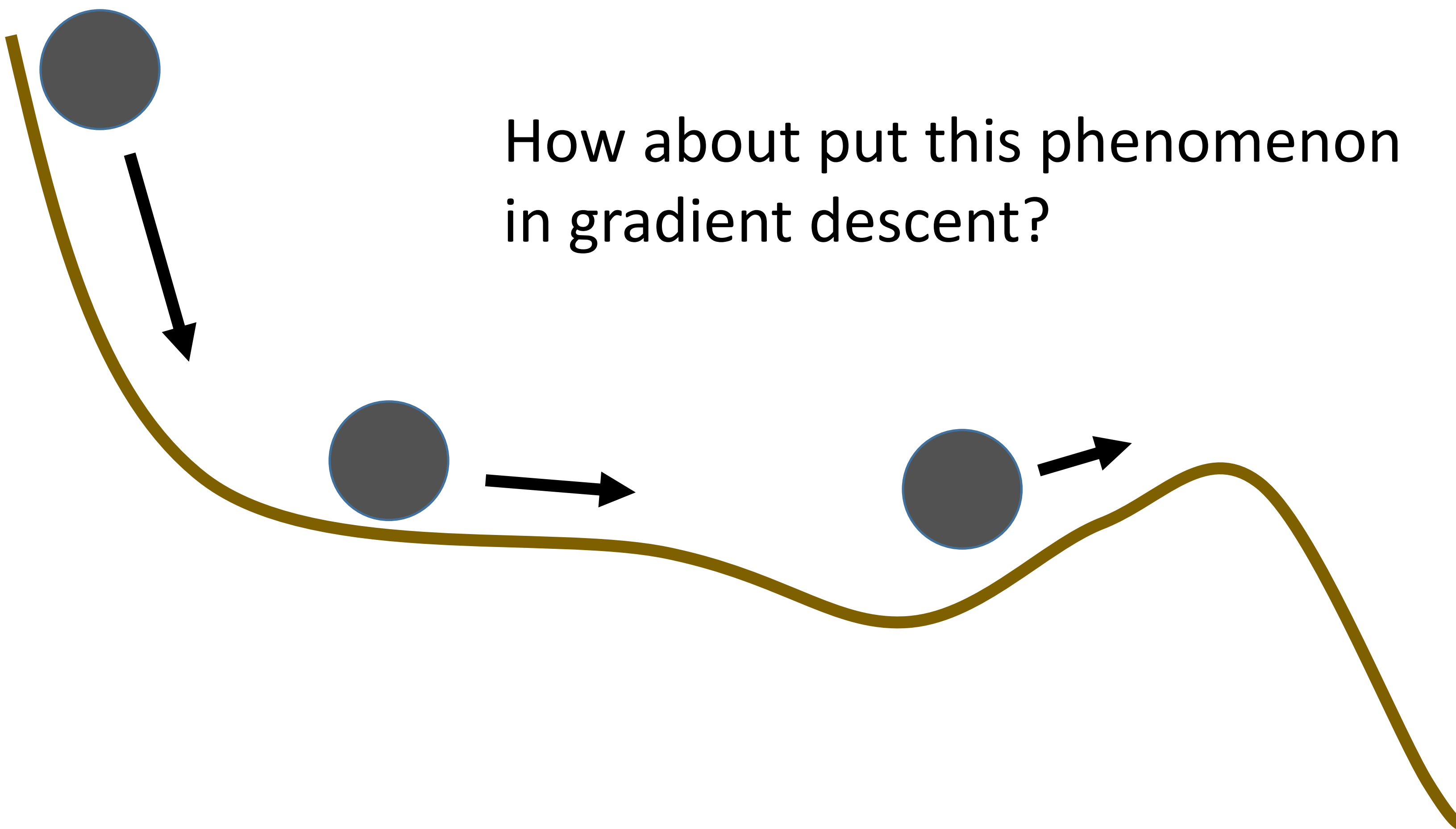


Hard to find optimal network parameters



In physical world ……

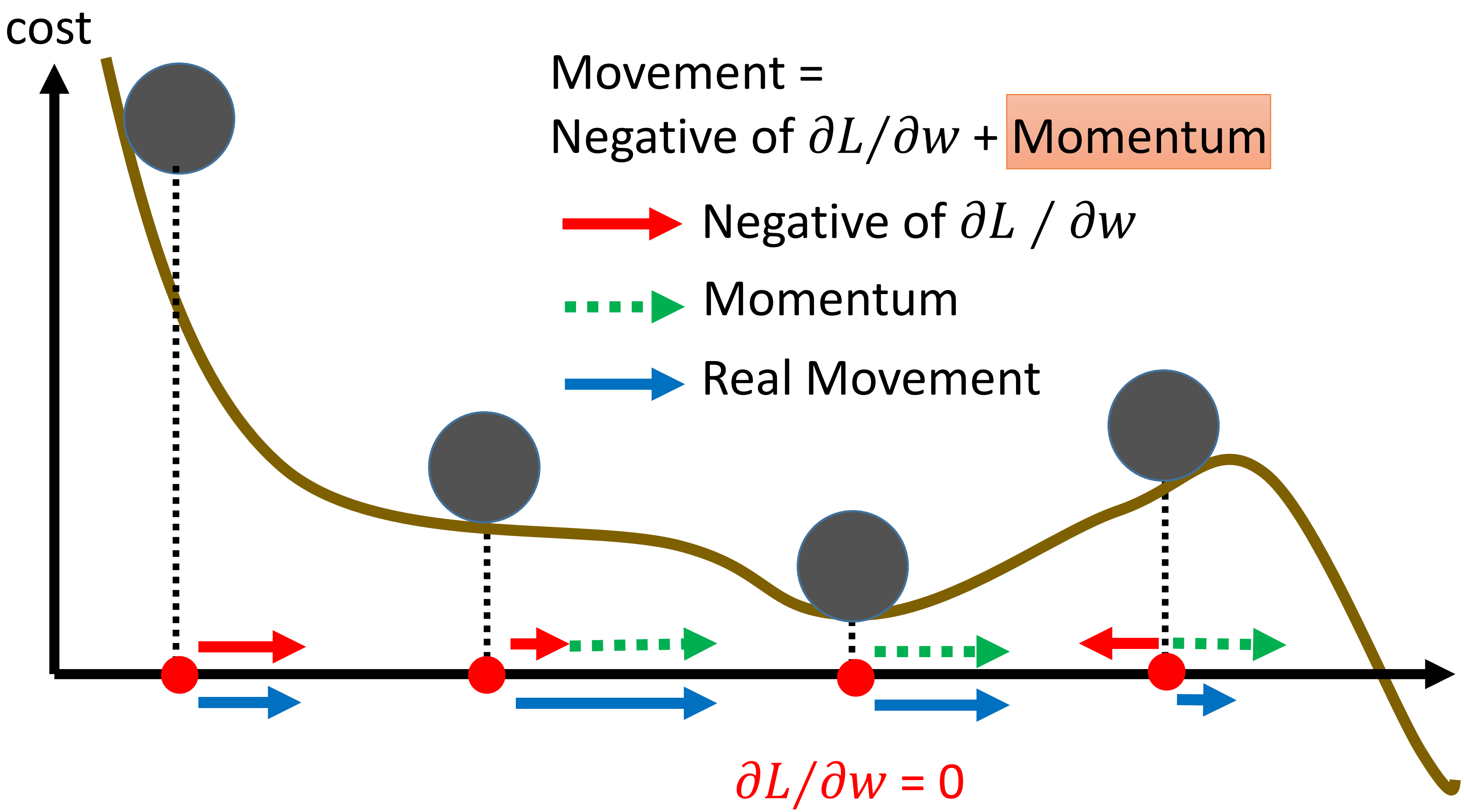
- Momentum



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'])
```

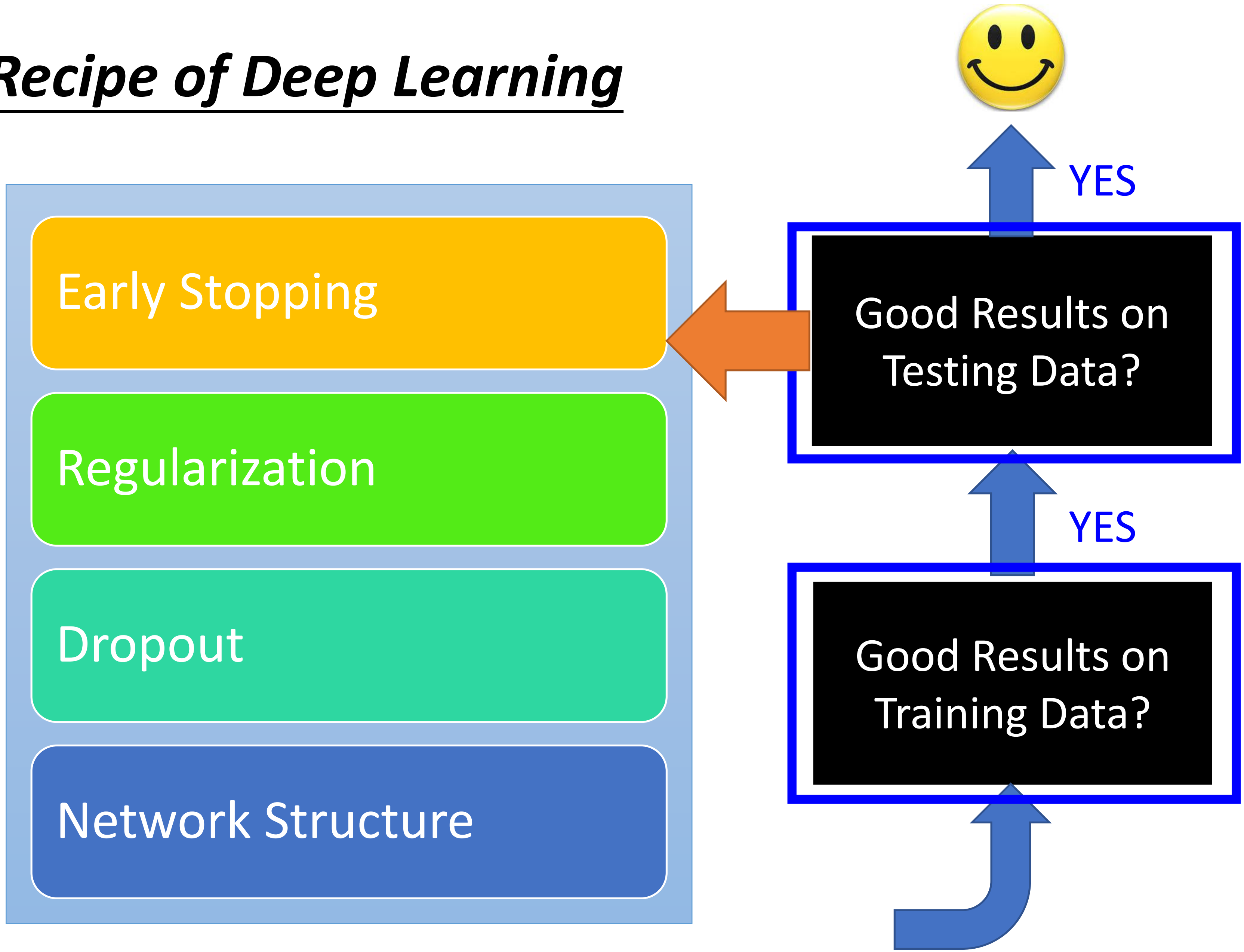
```
model.compile(loss='categorical_crossentropy',
              optimizer=Adam(),
              metrics=['accuracy'])
```

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)

Demo

Recipe of Deep Learning

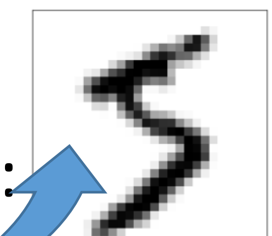


Panacea for Overfitting

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

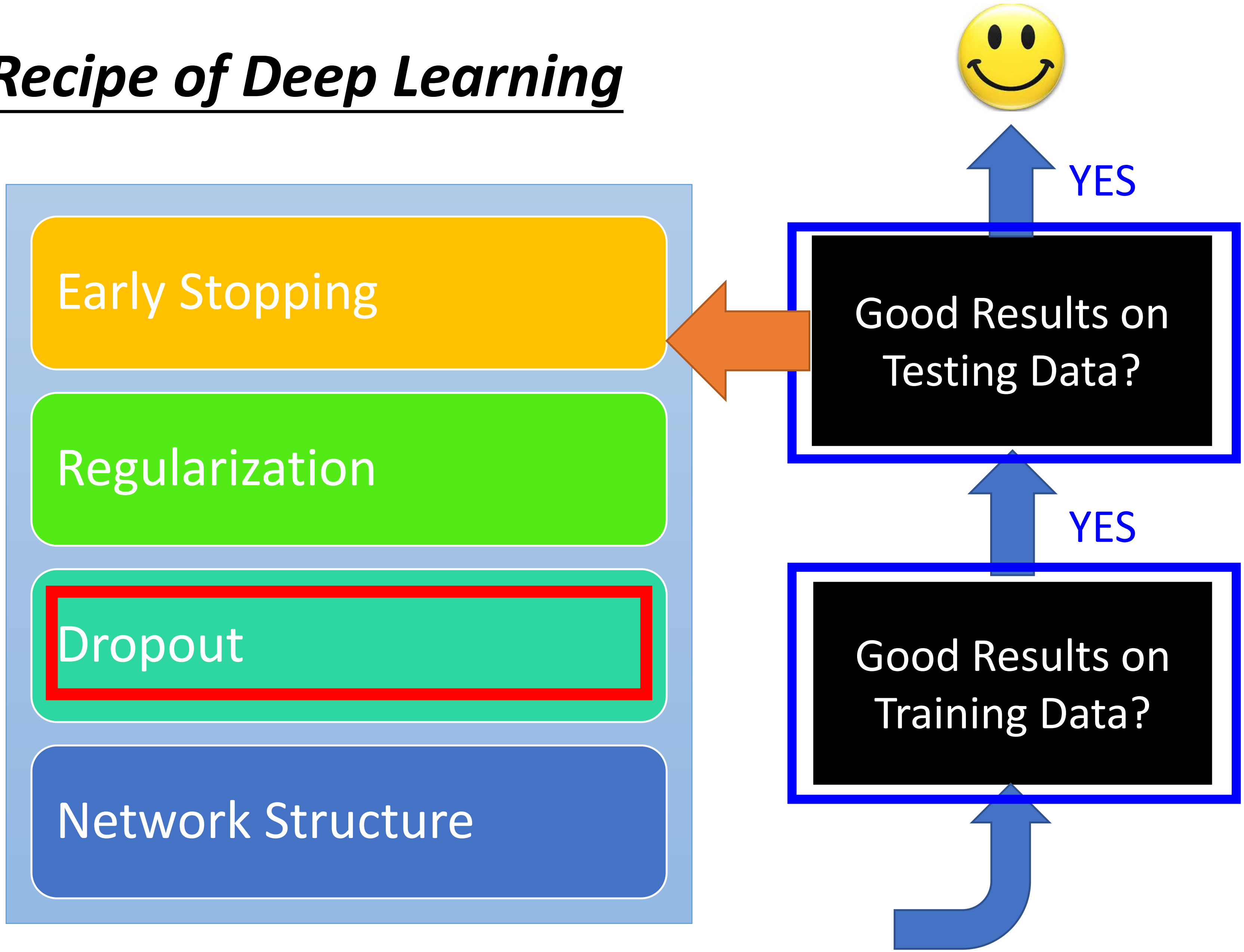
Handwriting recognition:

Original Training Data: 

Created Training Data: 

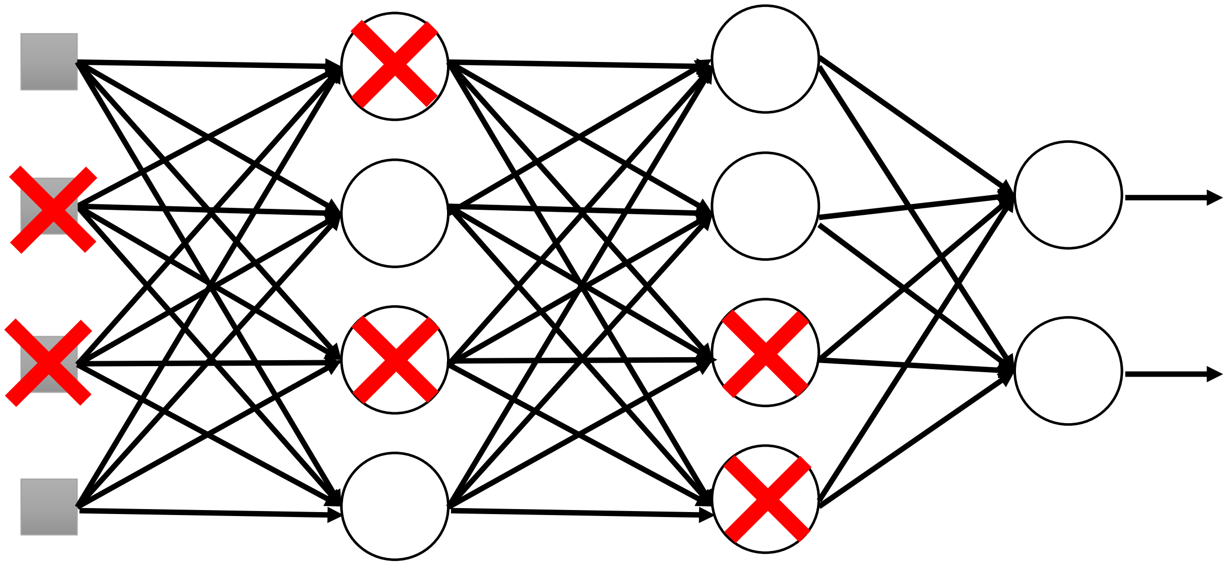
Shift 15 °

Recipe of Deep Learning



Dropout

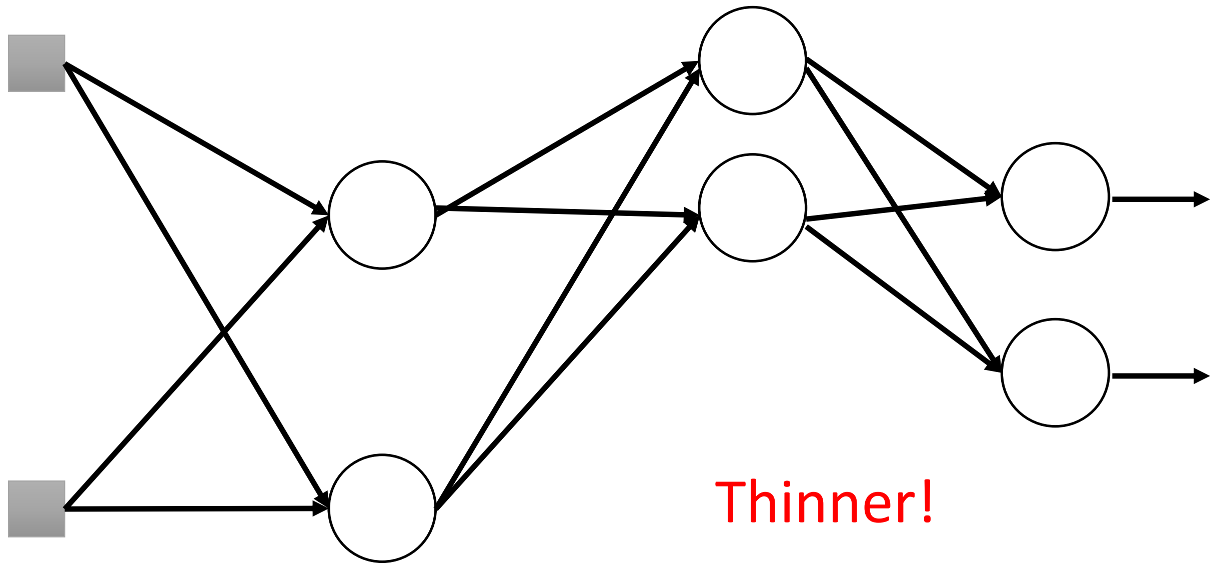
Training:



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

Dropout

Training:

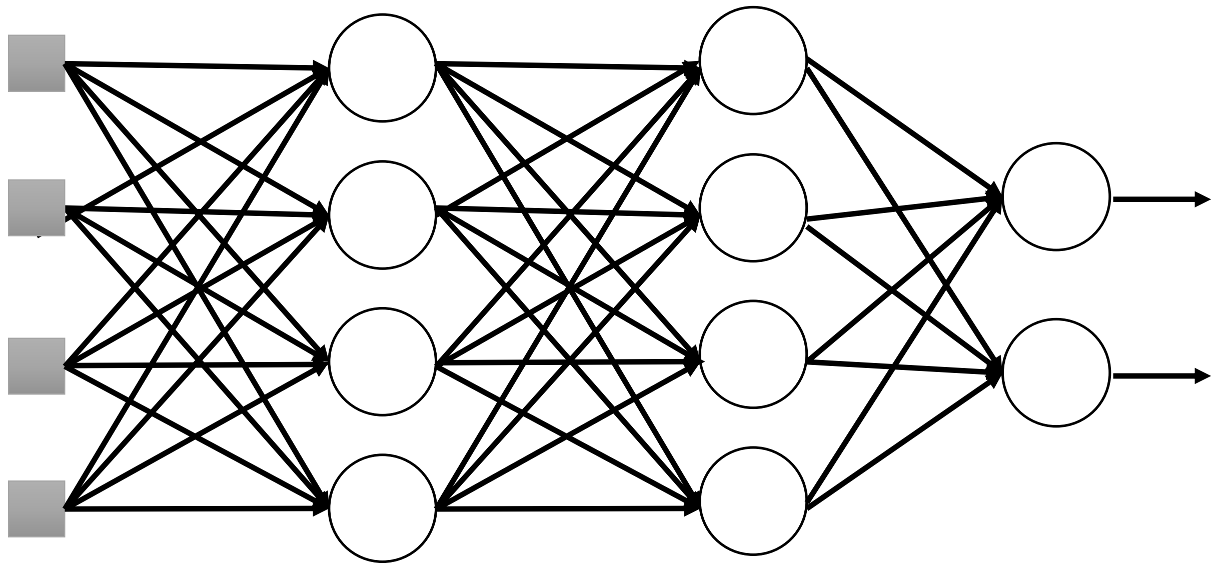


- **Each time before updating the parameters**
  - Each neuron has p% to dropout
    - **The structure of the network is changed.**
  - Using the new network for training

For each mini-batch, we resample the dropout neurons

Dropout

Testing:



➢ **No dropout**

- If the dropout rate at training is p%, all the weights times 1-p%
- Assume that the dropout rate is 50%. If a weight $w = 1$ by training, set $w = 0.5$ for testing.

Dropout - Intuitive Reason

Testing

No dropout

(拿下重物後就變很強)

Training

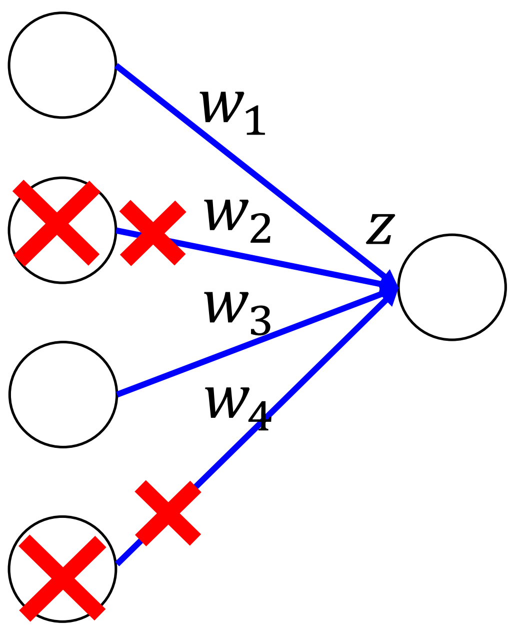
Dropout (腳上綁重物)

Dropout - Intuitive Reason

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

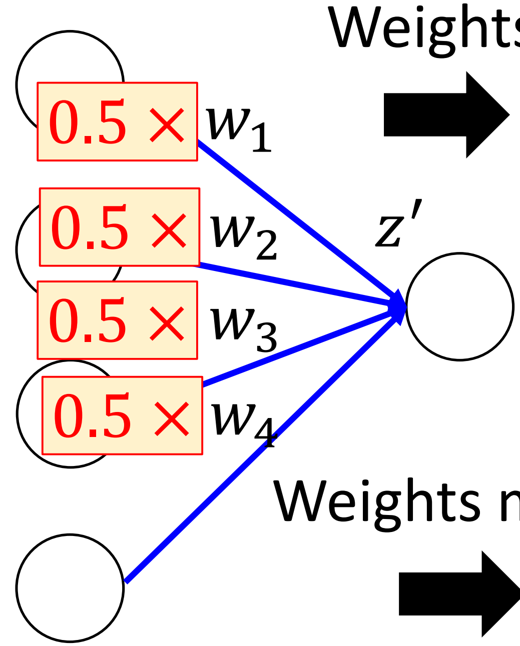
<u>Training of Dropout</u>

Assume dropout rate is 50%



<u>Testing of Dropout</u>

No dropout



Weights from training

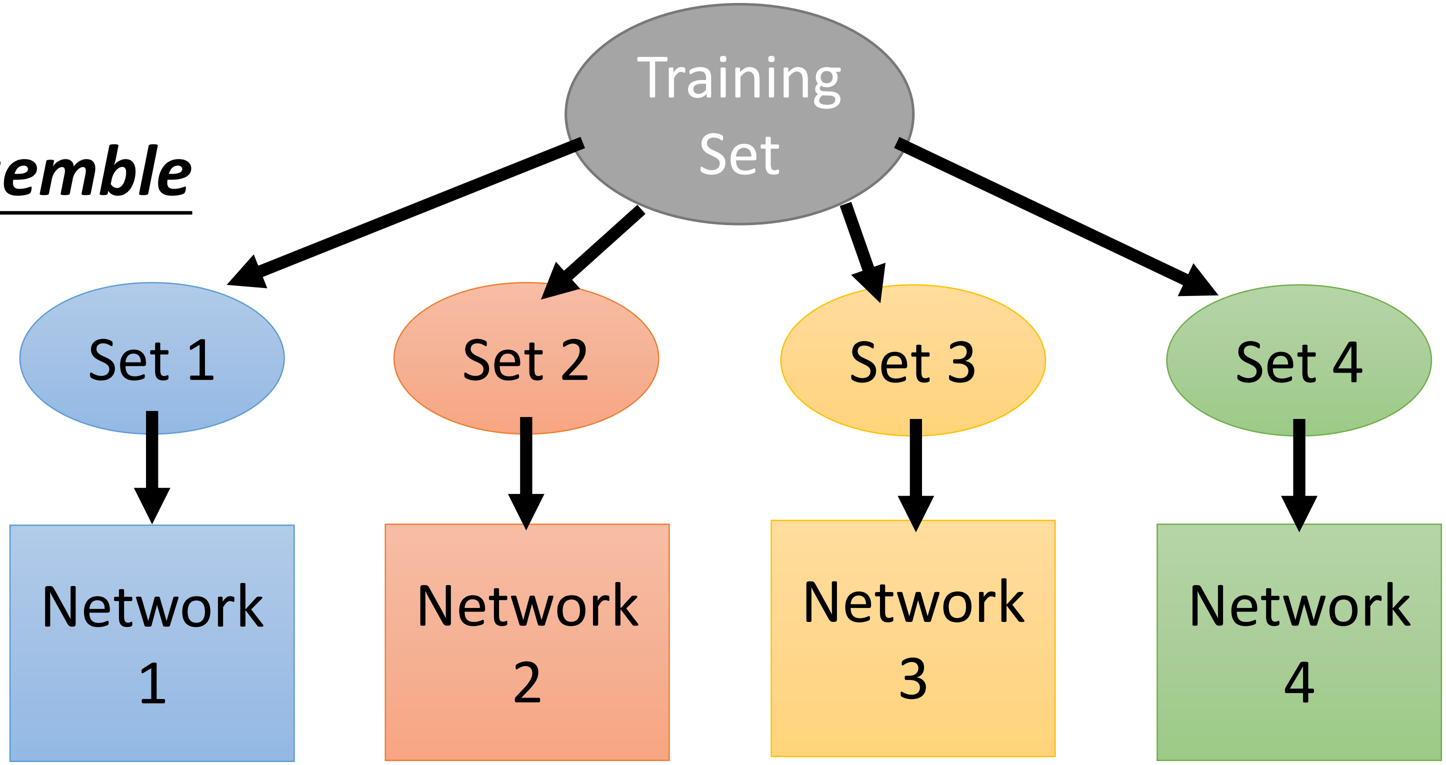
$\Rightarrow z' \approx 2z$

Weights multiply 1-p%

$\Rightarrow z' \approx z$

Dropout is a kind of ensemble.

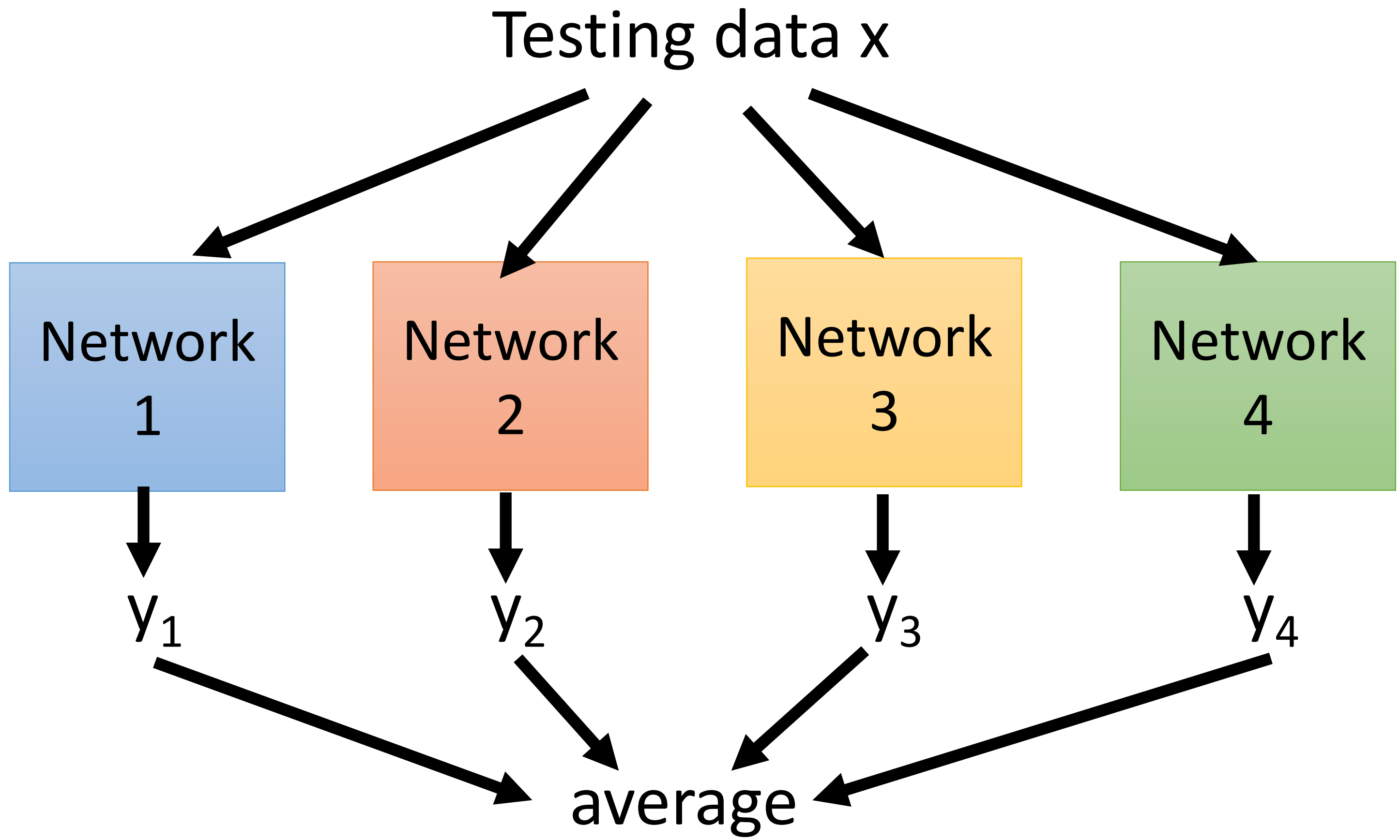
Ensemble



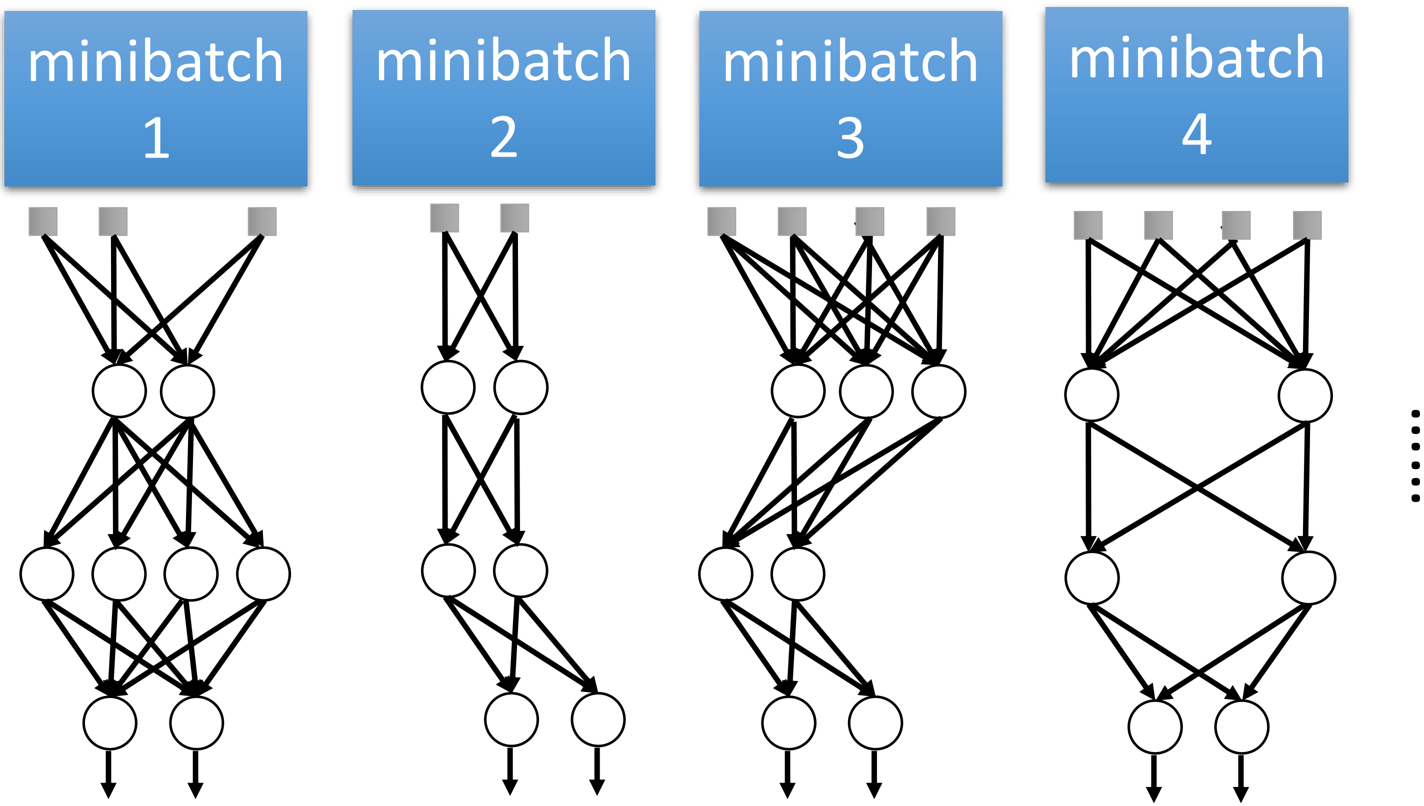
Train a bunch of networks with different structures

Dropout is a kind of ensemble.

Ensemble



Dropout is a kind of ensemble.



Training of Dropout

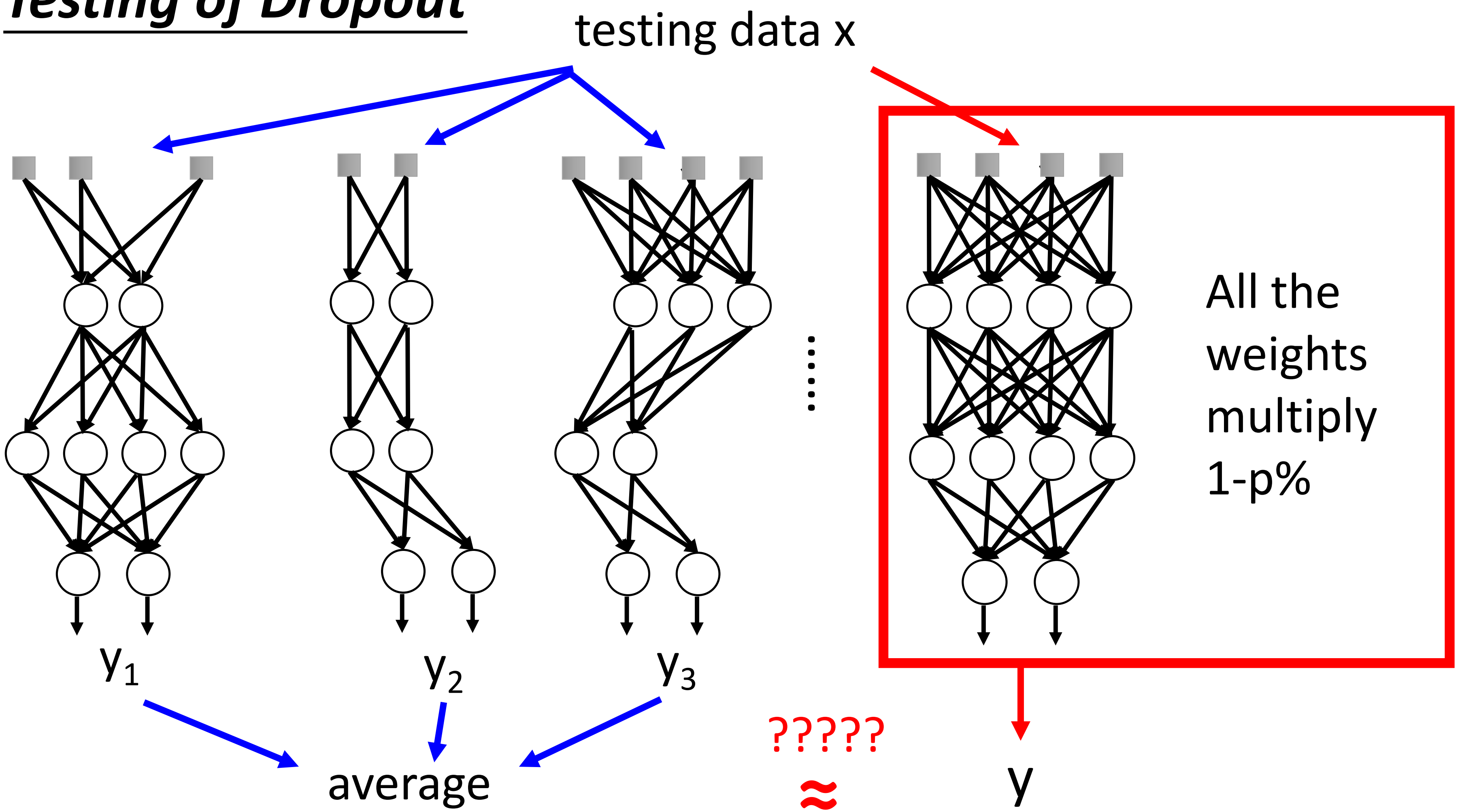
M neurons

2^M possible networks

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

Dropout is a kind of ensemble.

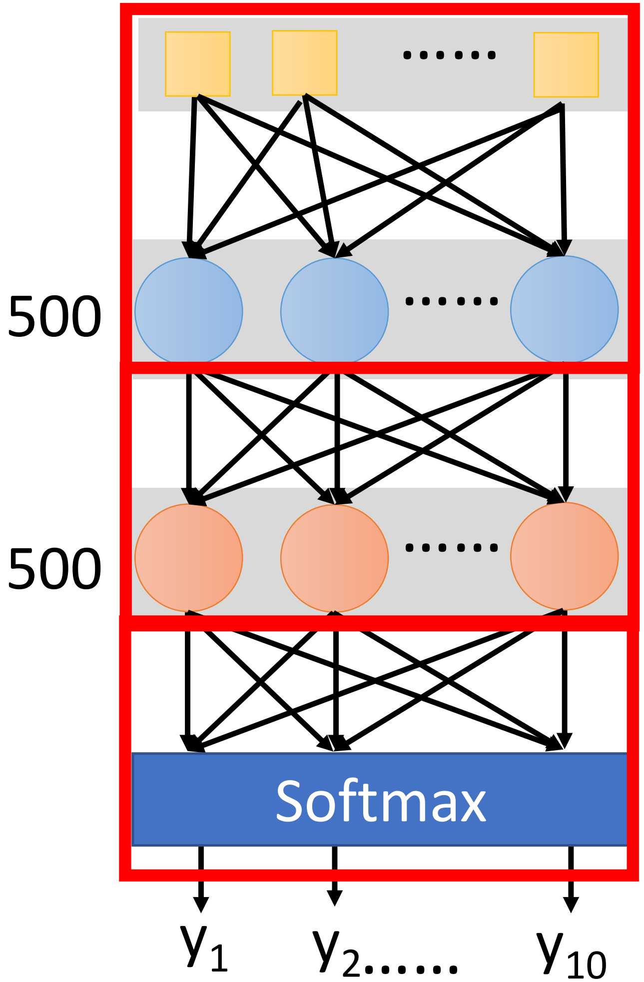
Testing of Dropout



More about dropout

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

Demo




```
model = Sequential()
```

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

model.add(dropout(0.8))

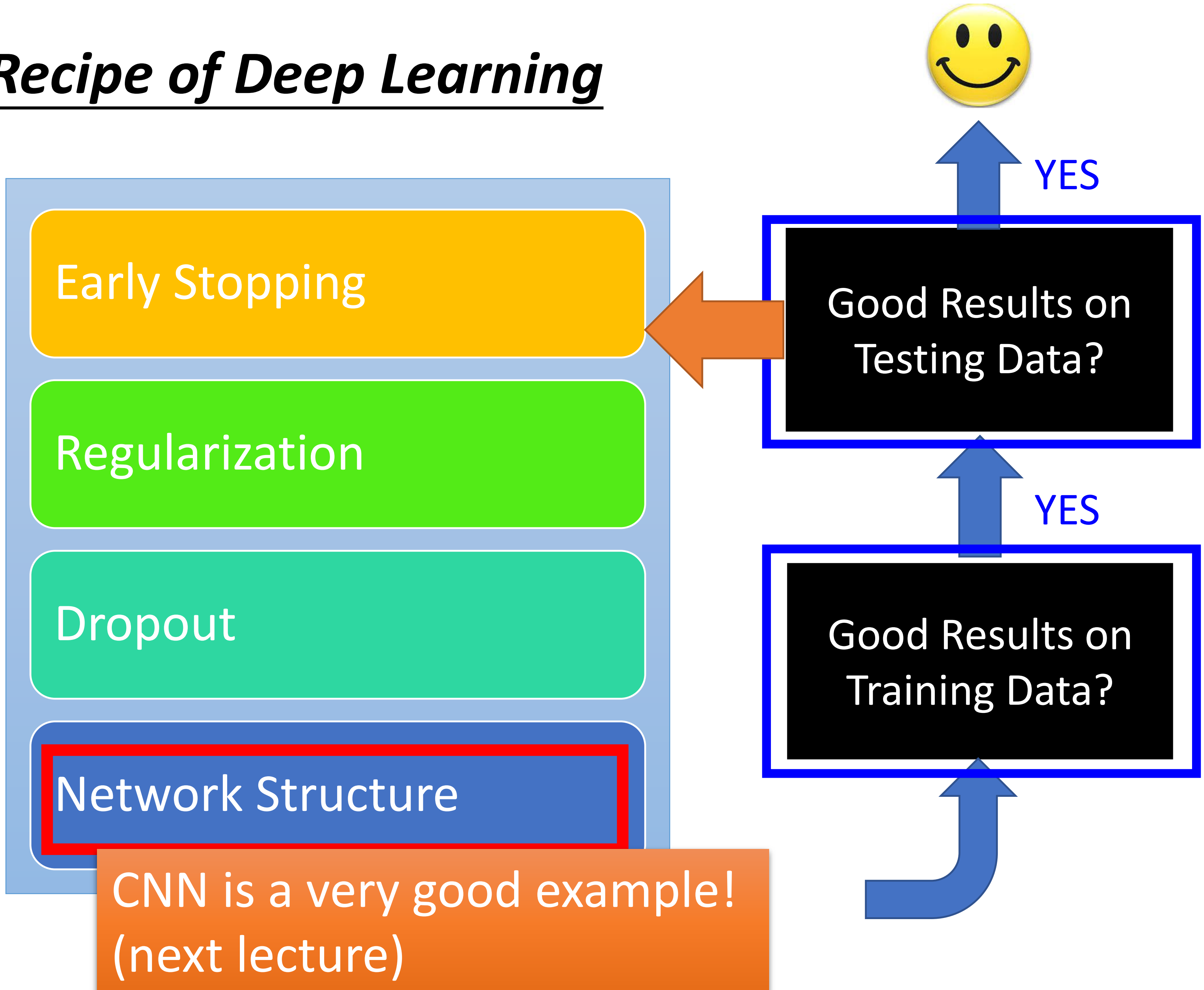
```
model.add( Dense( output_dim=500 ) )
model.add( Activation('sigmoid') )
```

model.add(dropout(0.8))

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

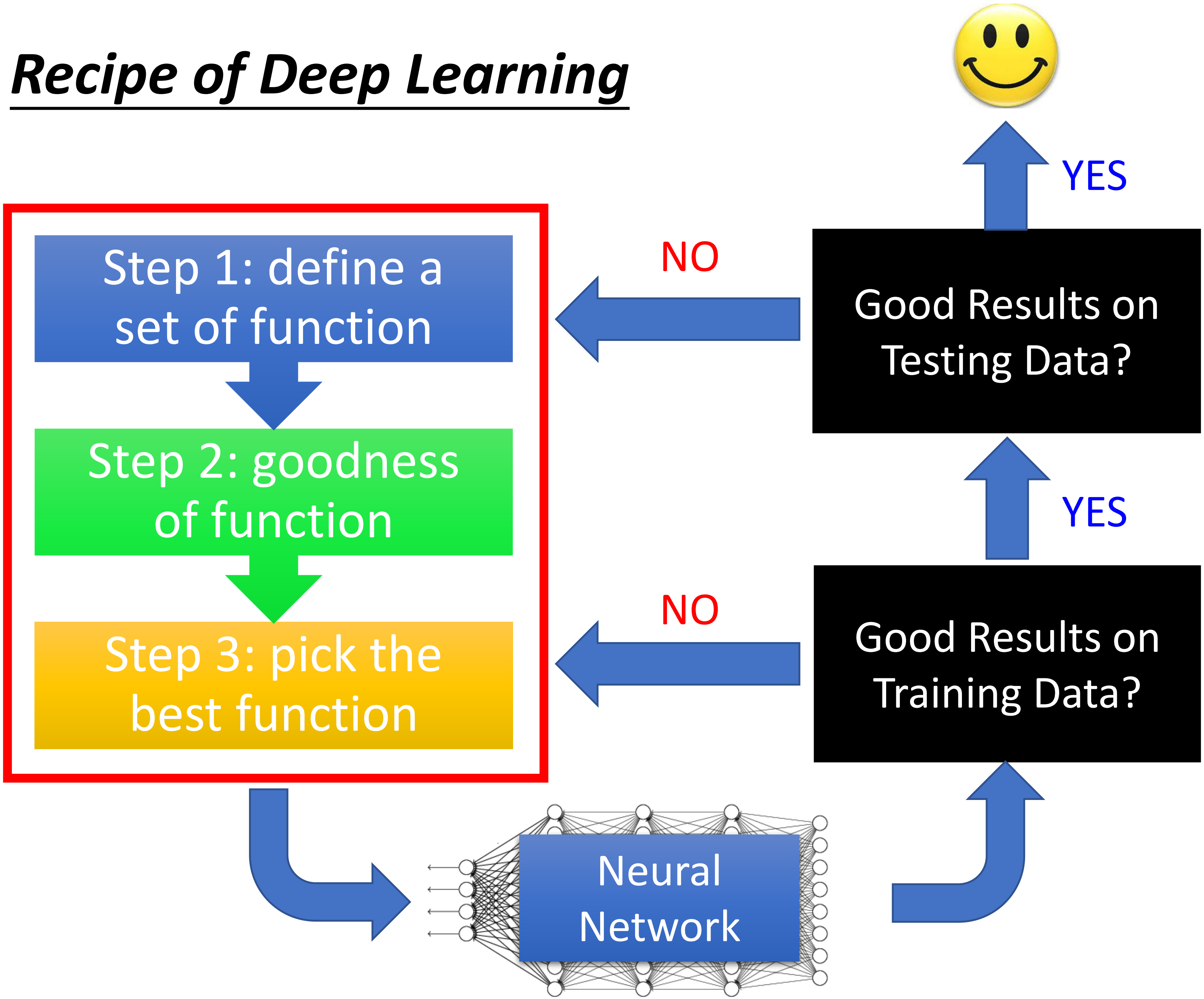
Demo

Recipe of Deep Learning



Concluding Remarks

Recipe of Deep Learning



Lecture II:
Variants of Neural Networks

Variants of Neural Networks

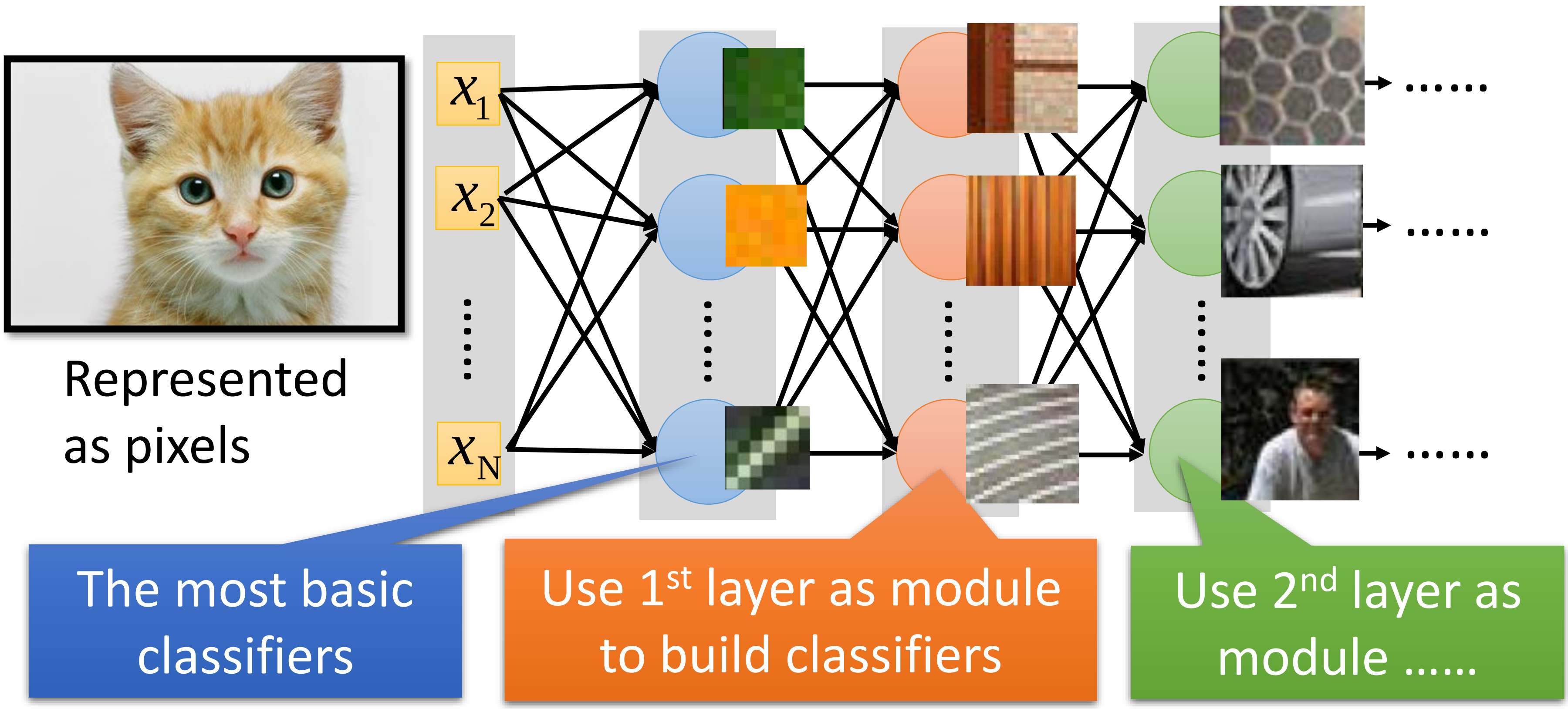
Convolutional Neural Network (CNN)

Widely used in image processing

Recurrent Neural Network (RNN)

Why CNN for Image?

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



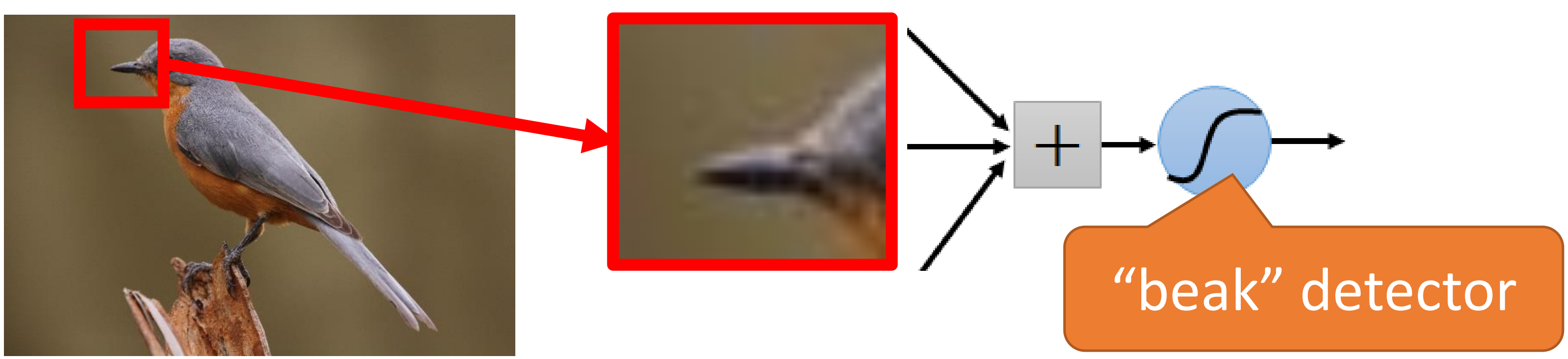
Can the network be simplified by considering the properties of images?

Why CNN for Image

- Some patterns are much smaller than the whole image

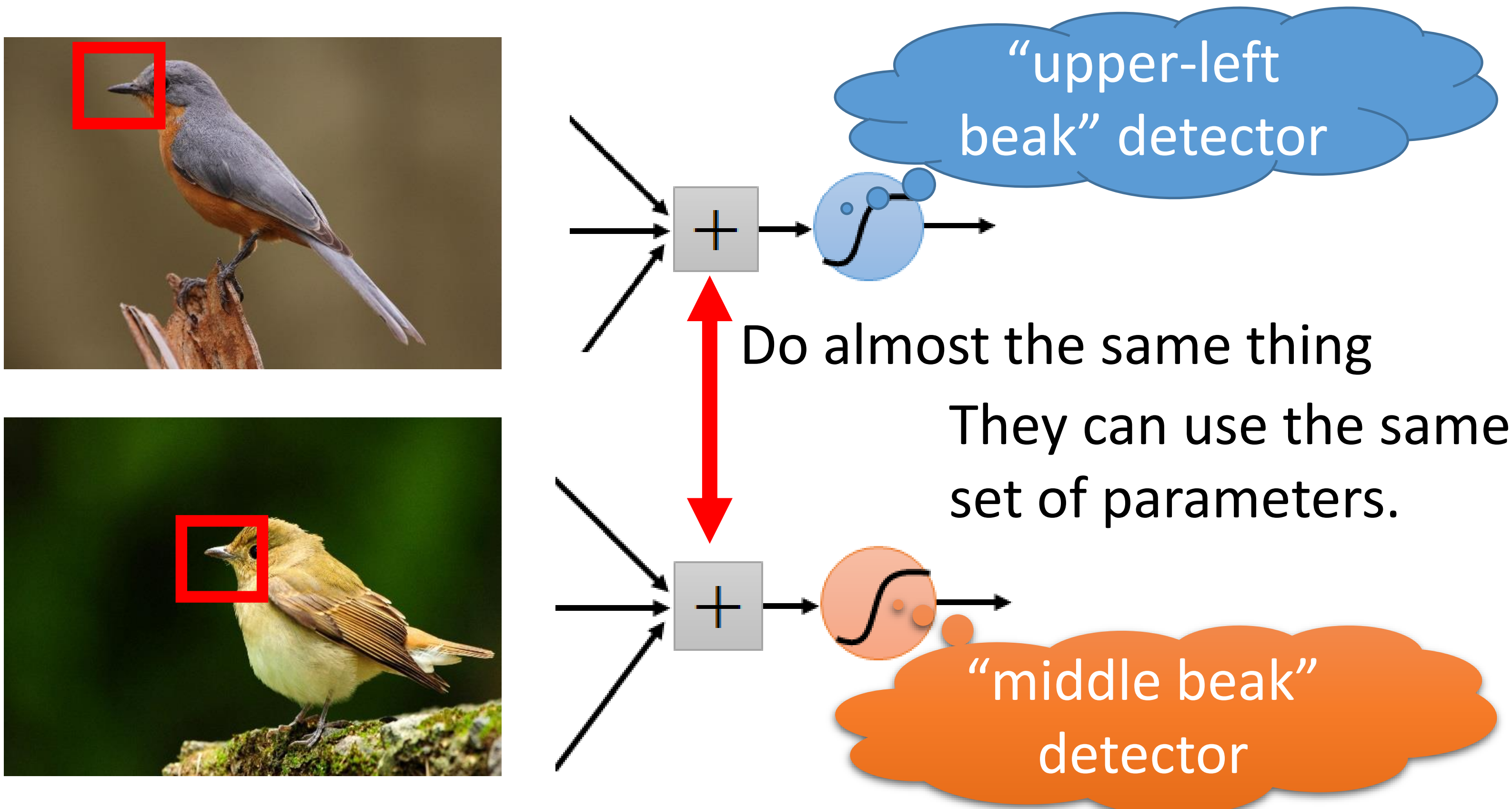
A neuron does not have to see the whole image to discover the pattern.

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

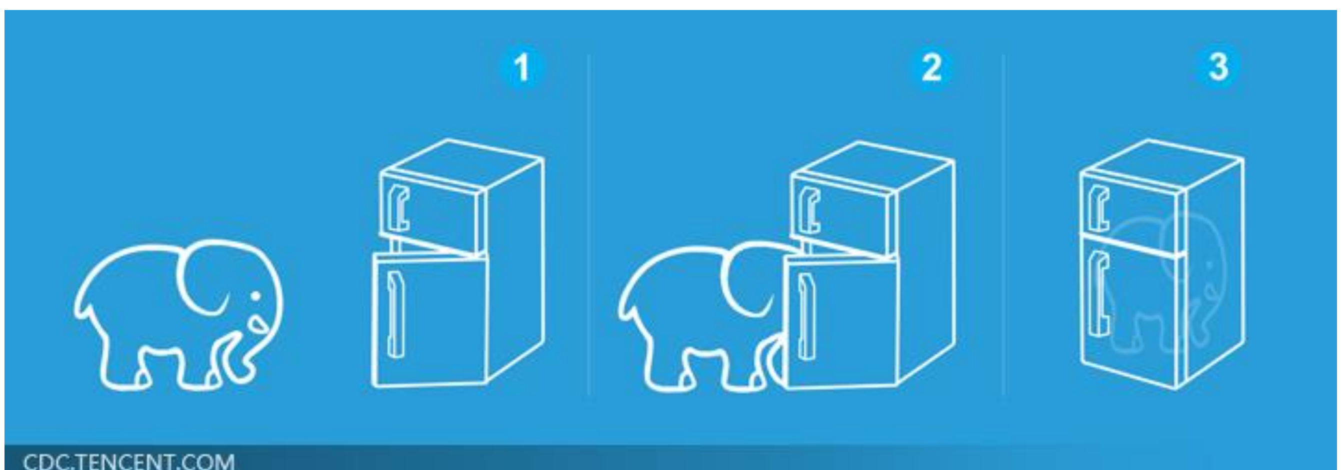
We can subsample the pixels to make image smaller

Less parameters for the network to process the image

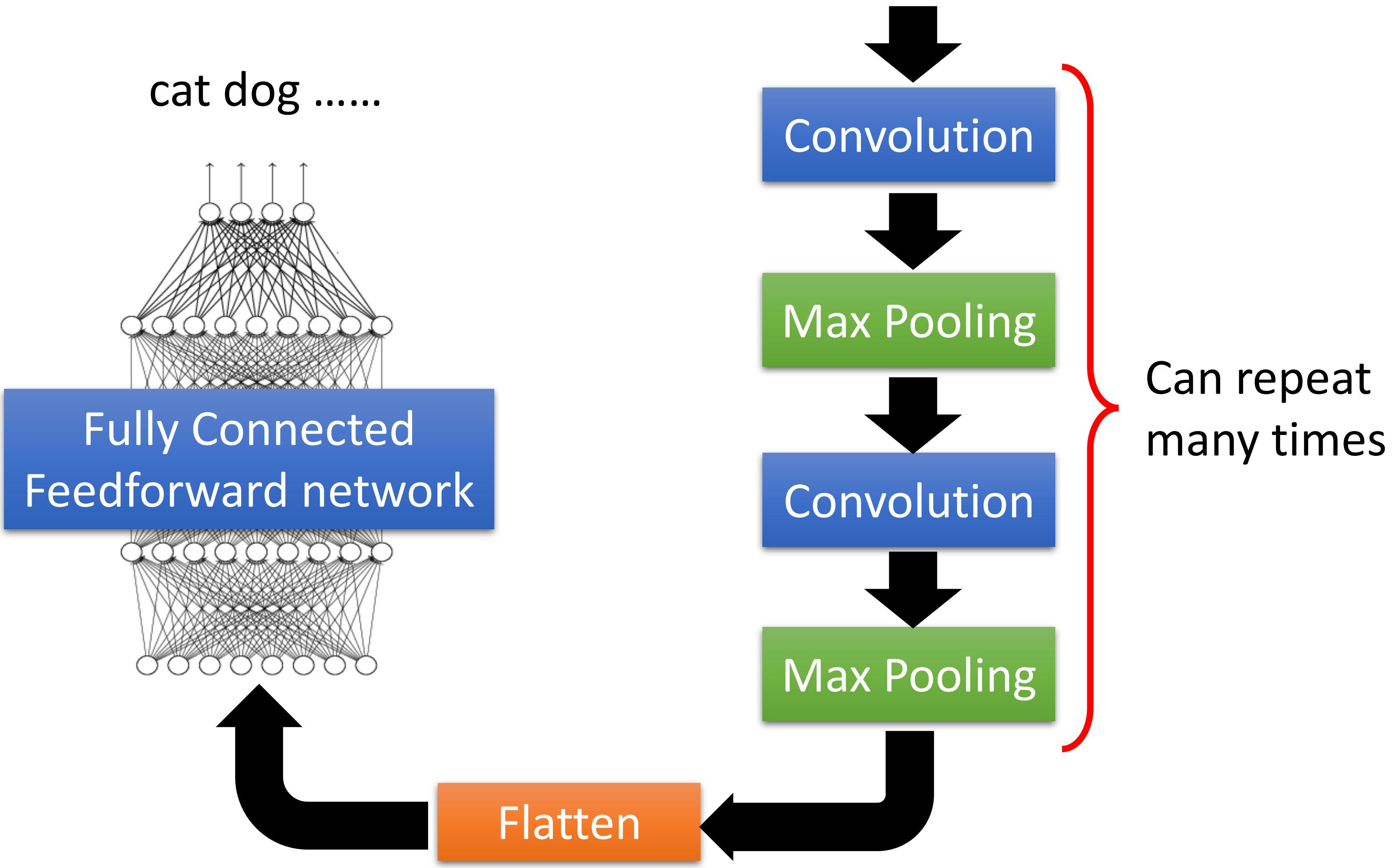
Three Steps for Deep Learning

Step 1: Convolutional Neural Network → Step 2: goodness of function → Step 3: pick the best function

Deep Learning is so simple ……



The whole CNN



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

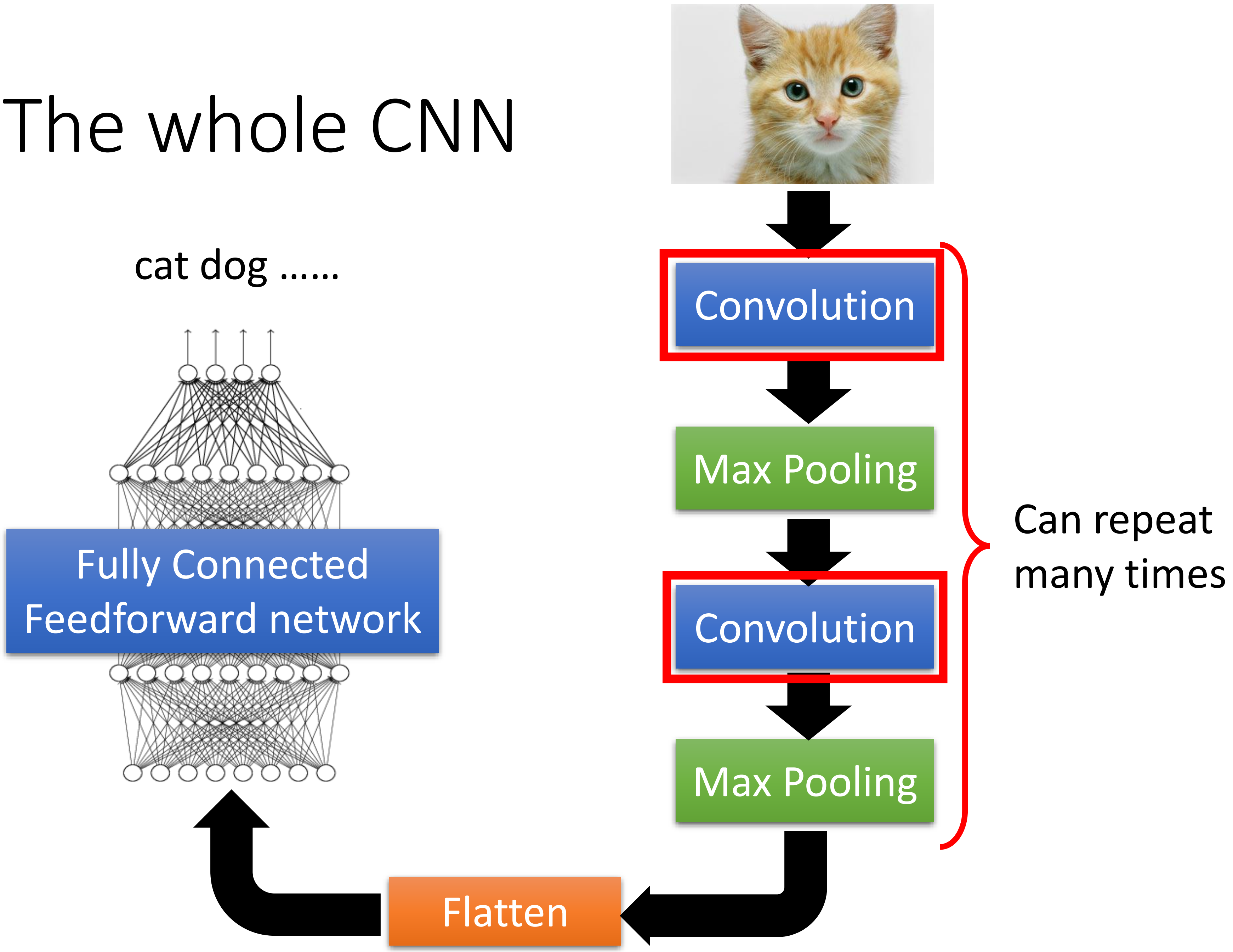
Convolution

Max Pooling

Can repeat many times

Flatten

The whole CNN



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

3 -1

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

3   -3

We set stride=1 below

CNN – Convolution

Filter 1

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

3	-1	-3	-1
-3	1	0	-3
-3	-3	0	1
3	-2	-2	-1

Property 2

CNN – Convolution

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

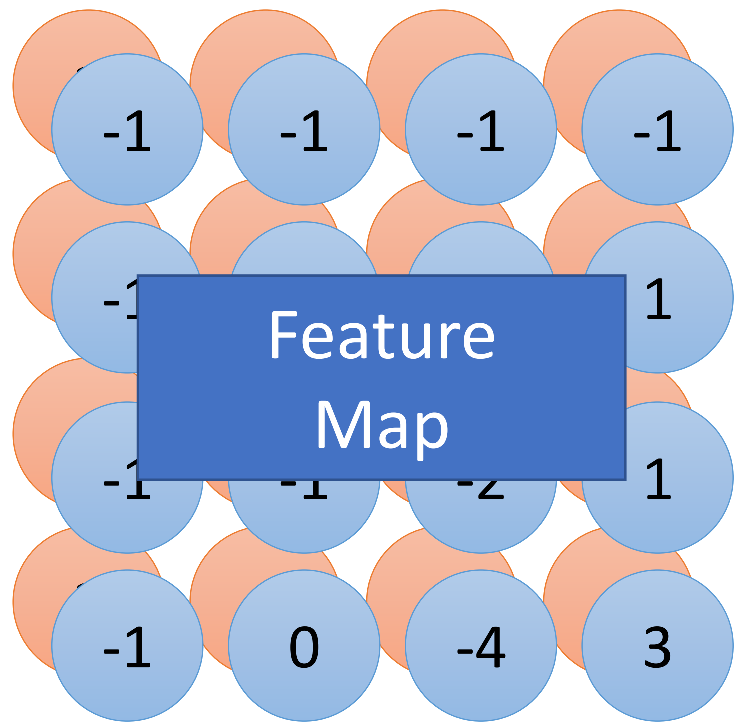
Filter 2

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

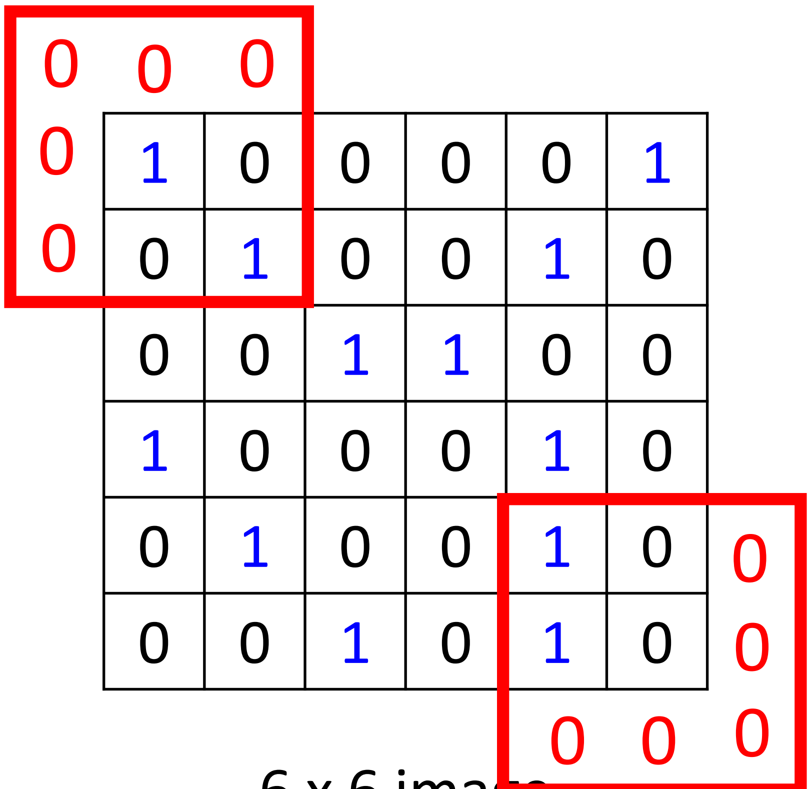


4 x 4 image

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

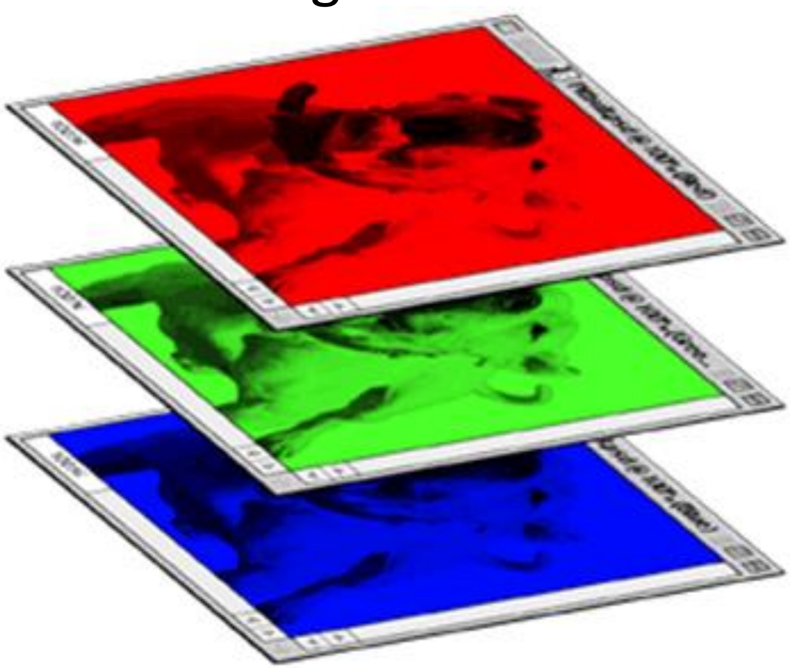
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

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

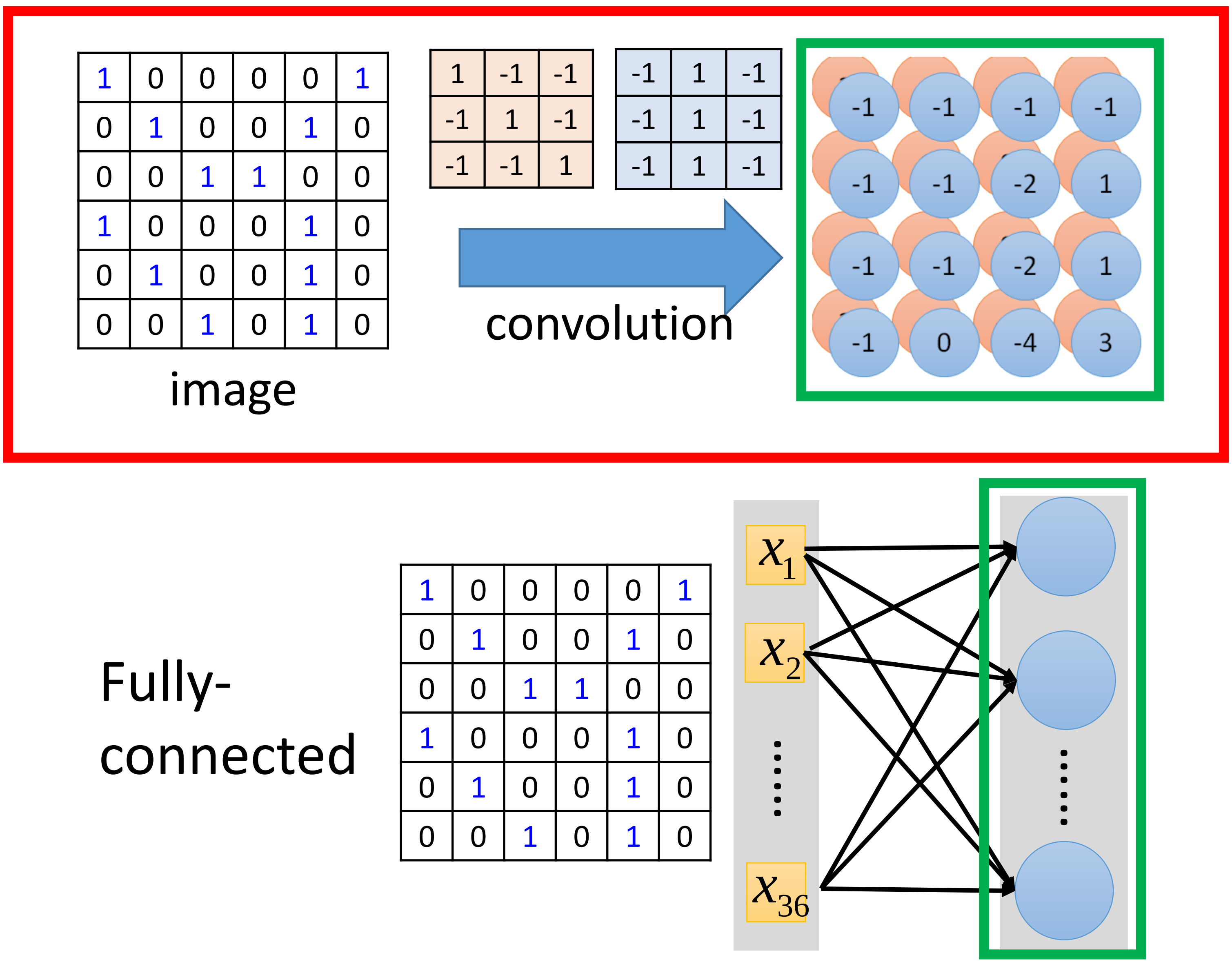
Filter 2

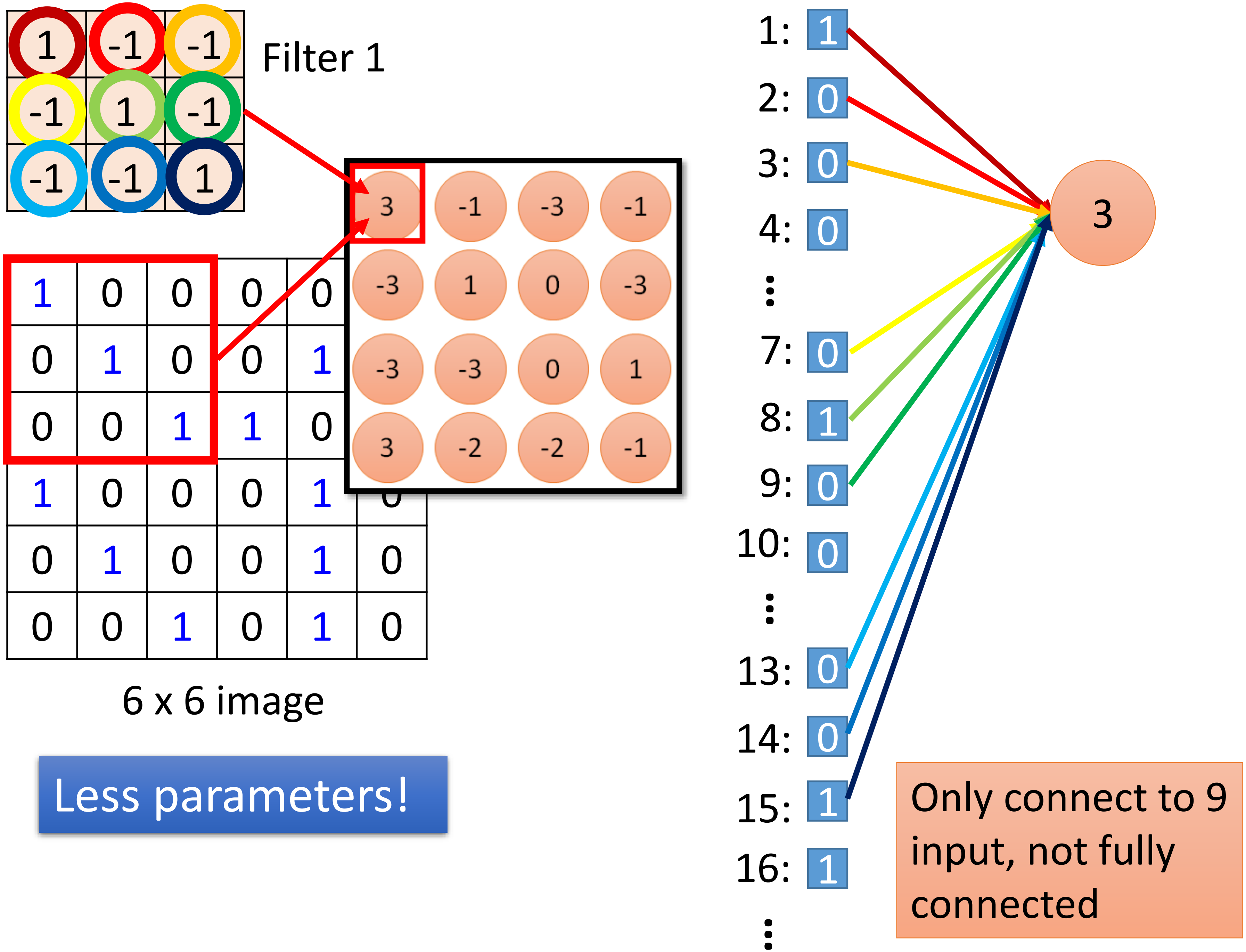
Colorful image

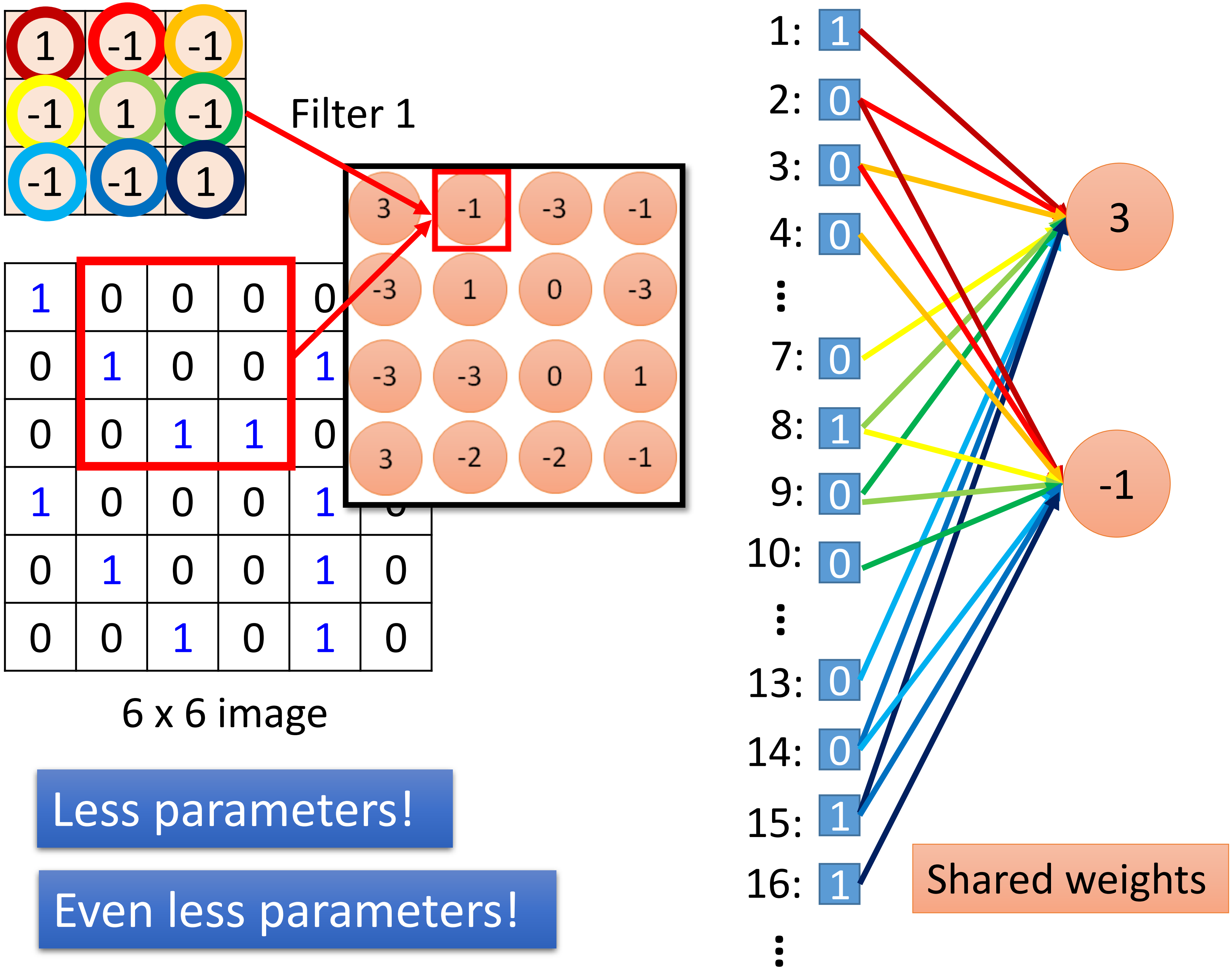


1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

Convolution v.s. Fully Connected

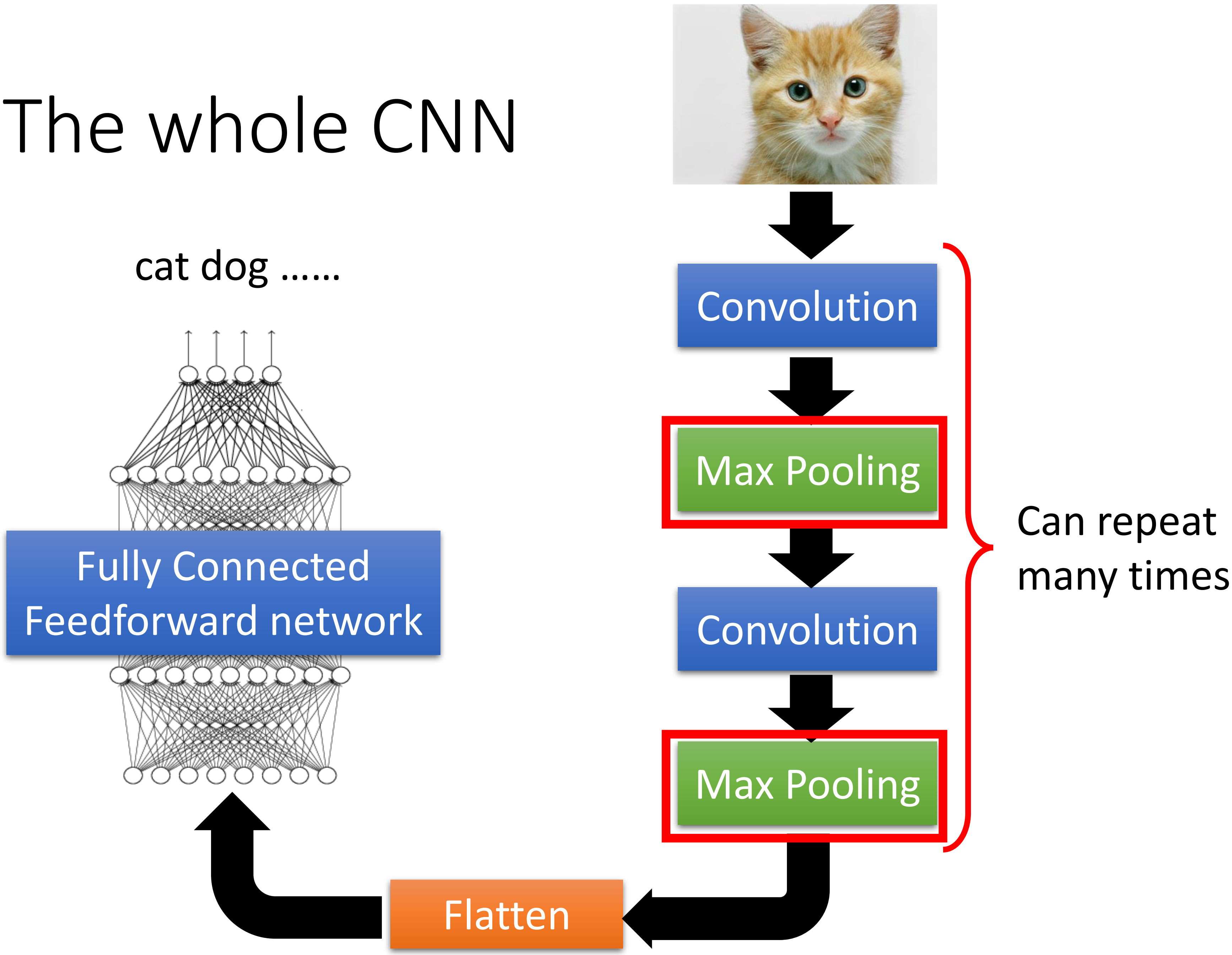



Filter 1
1 -1 -1
-1 1 -1
-1 -1 1
6 x 6 image
1 0 0 0 0 1
0 1 0 0 1 0
0 0 1 1 0 0
1 0 0 0 1 0
0 1 0 0 1 0
0 0 1 0 1 0
3 -1 -3 -1
-3 1 0 -3
-3 -3 0 1
3 -2 -2 -1
1: 1
2: 0
3: 0
4: 0
7: 0
8: 1
9: 0
10: 0
13: 0
14: 0
15: 1
16: 1
3
Less parameters!
Only connect to 9 input, not fully connected



Filter 1
6 x 6 image
Less parameters!
Even less parameters!
Shared weights
1:
2:
3:
4:
7:
8:
9:
10:
13:
14:
15:
16:

The whole CNN



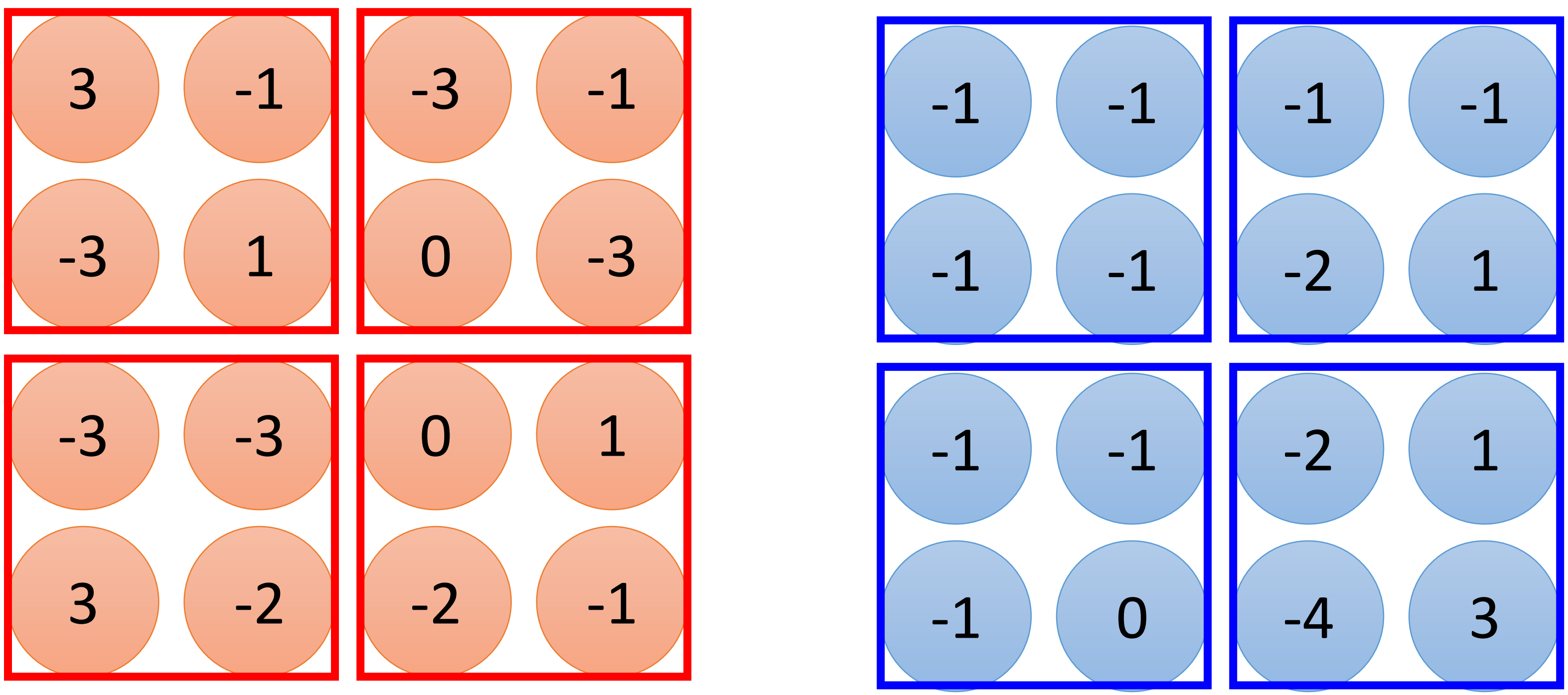
CNN – Max Pooling

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

Filter 1

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

Filter 2

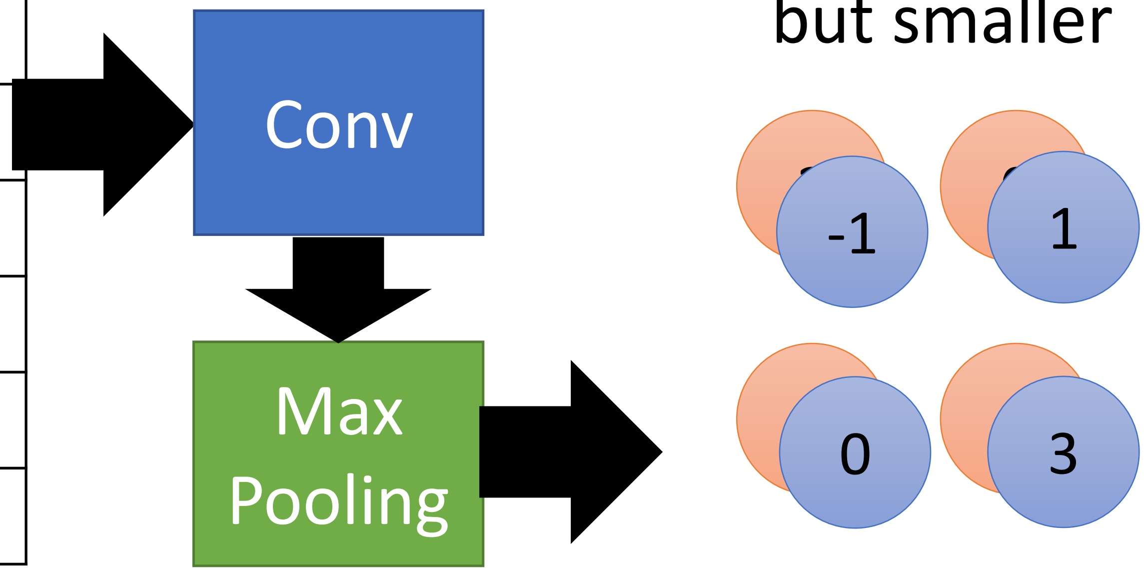


CNN – Max Pooling

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

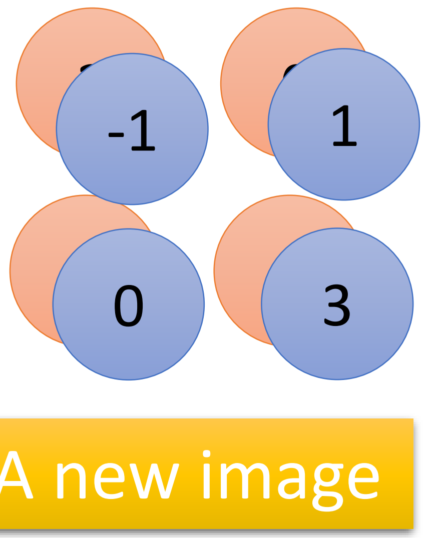
New image
but smaller

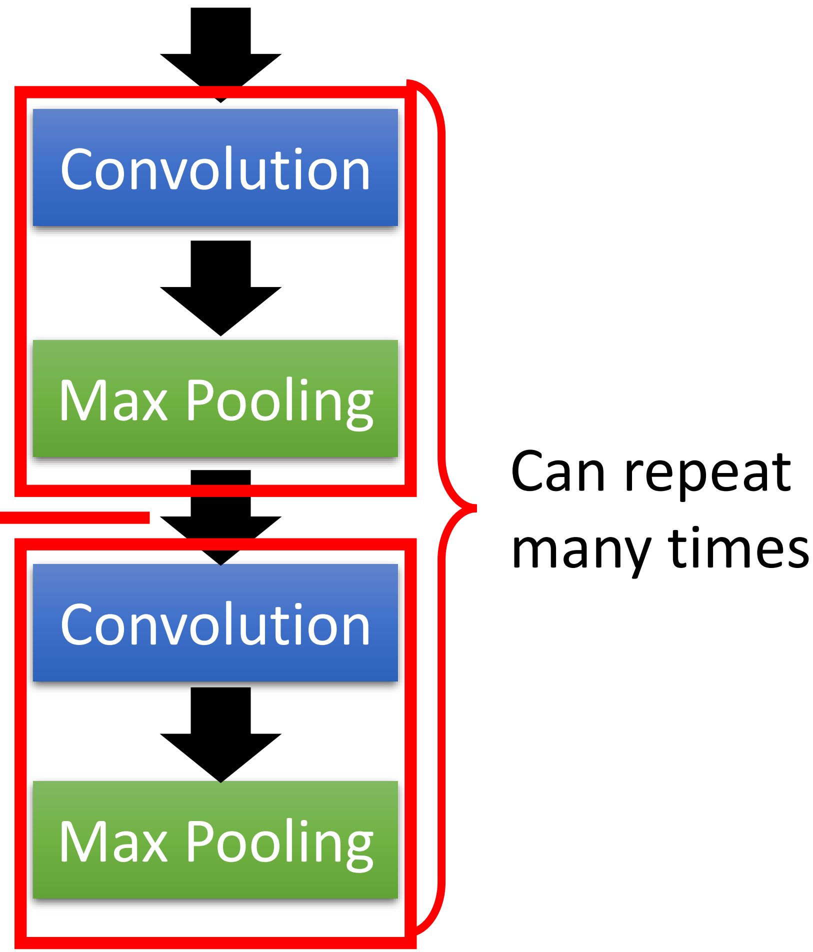


2 x 2 image

Each filter
is a channel

The whole CNN

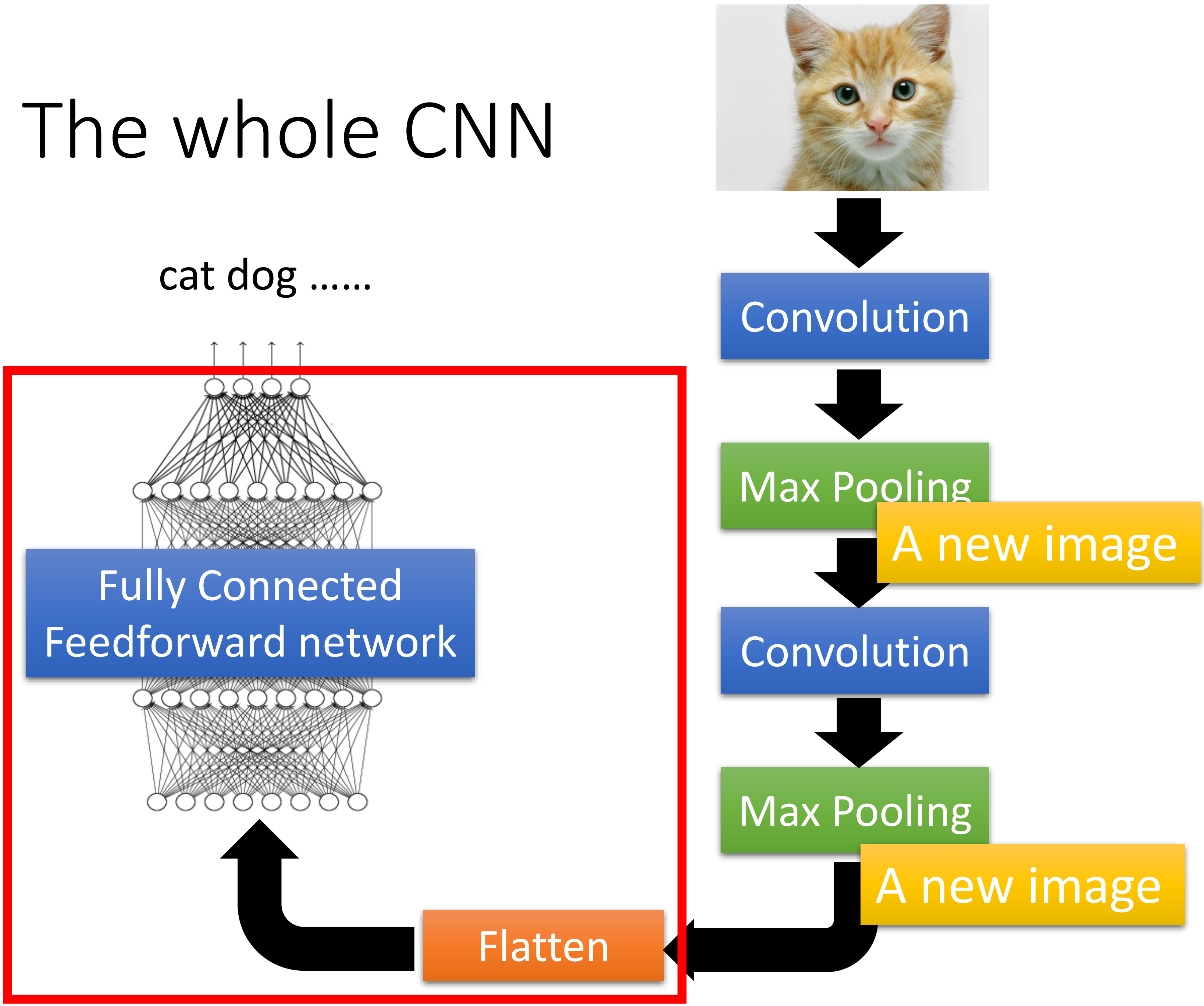




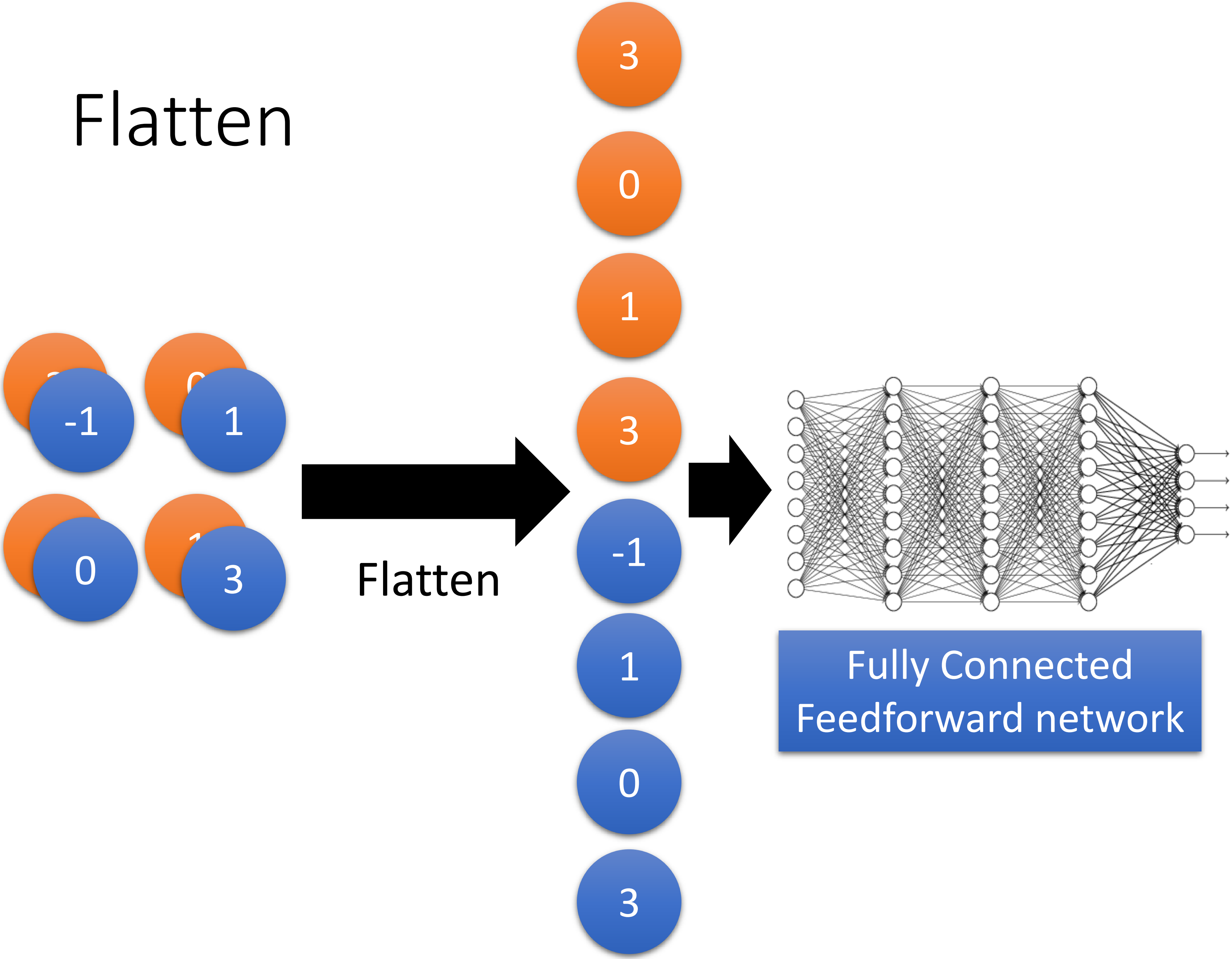
Smaller than the original image

The number of the channel is the number of filters

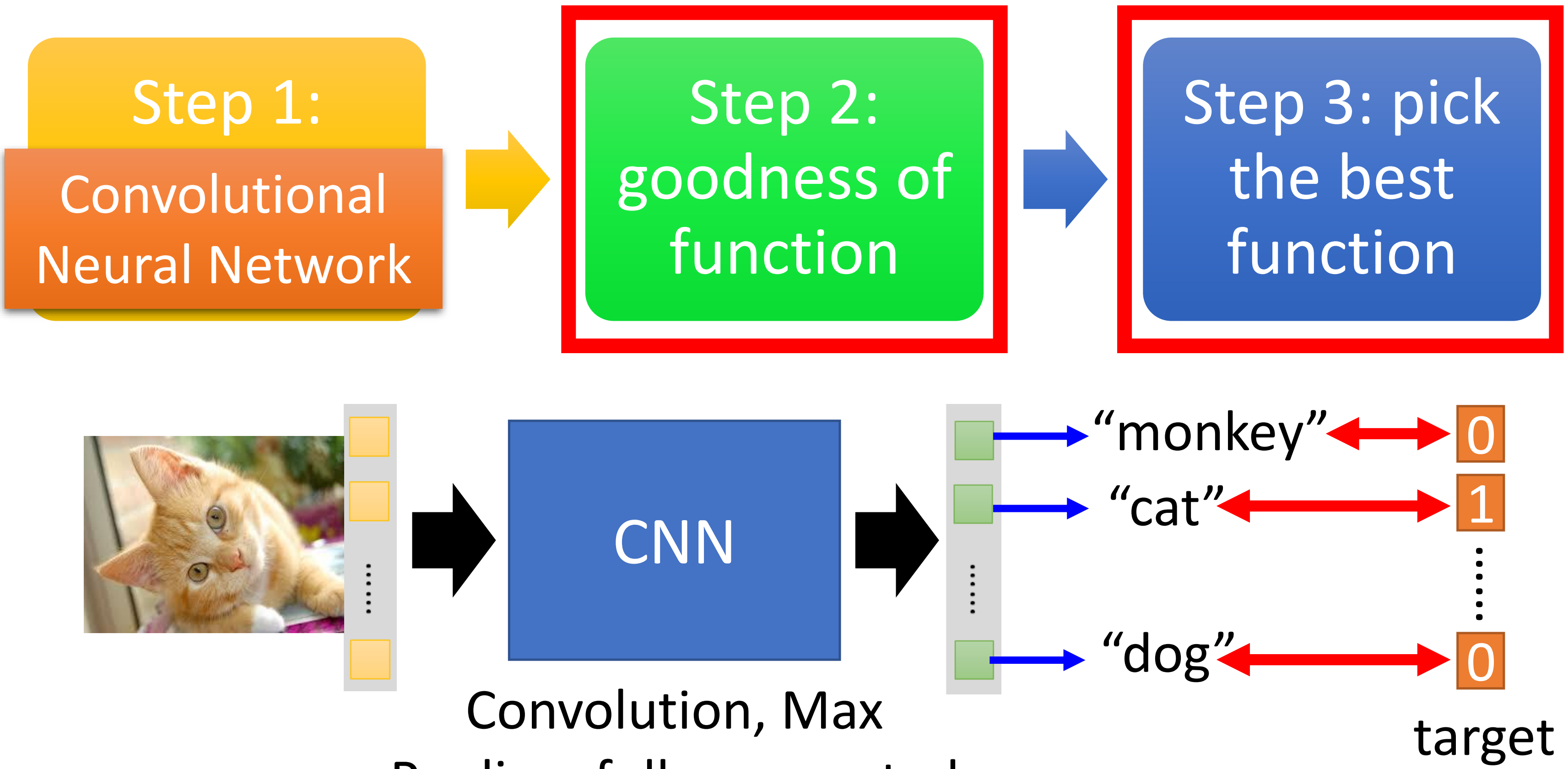
The whole CNN



Flatten


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

Convolutional Neural Network

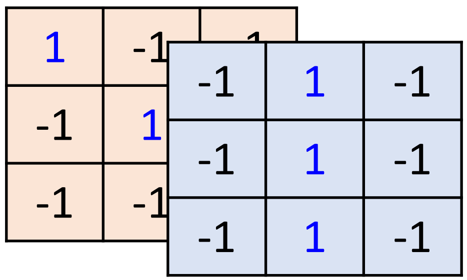


Learning: Nothing special, just gradient descent ……

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) ) )
```



……

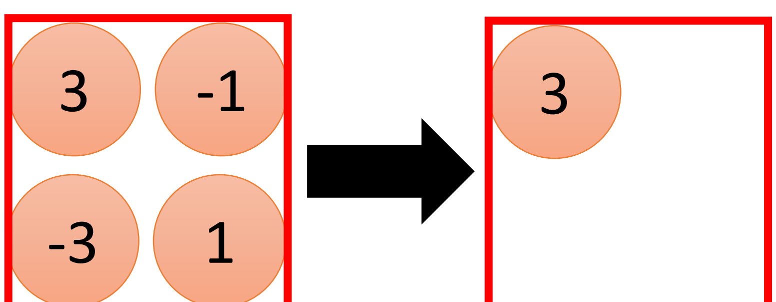
There are 25 3x3 filters.

Input_shape = (1 , 28 , 28)

1: black/weight, 3: RGB

28 x 28 pixels

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



input

Convolution

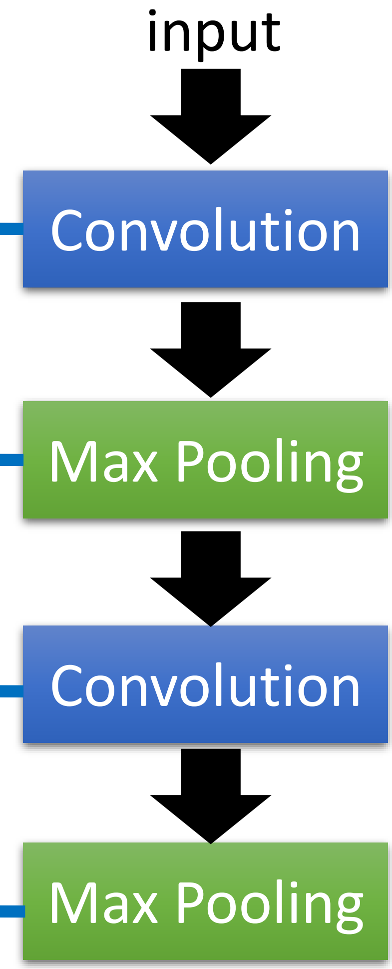
Max Pooling

Convolution

Max Pooling

CNN in Keras

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



1 x 28 x 28

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

How many parameters for each filter? 9

25 x 26 x 26

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

25 x 13 x 13

```
model2.add(Convolution2D(50,3,3))
```

How many parameters for each filter? 225

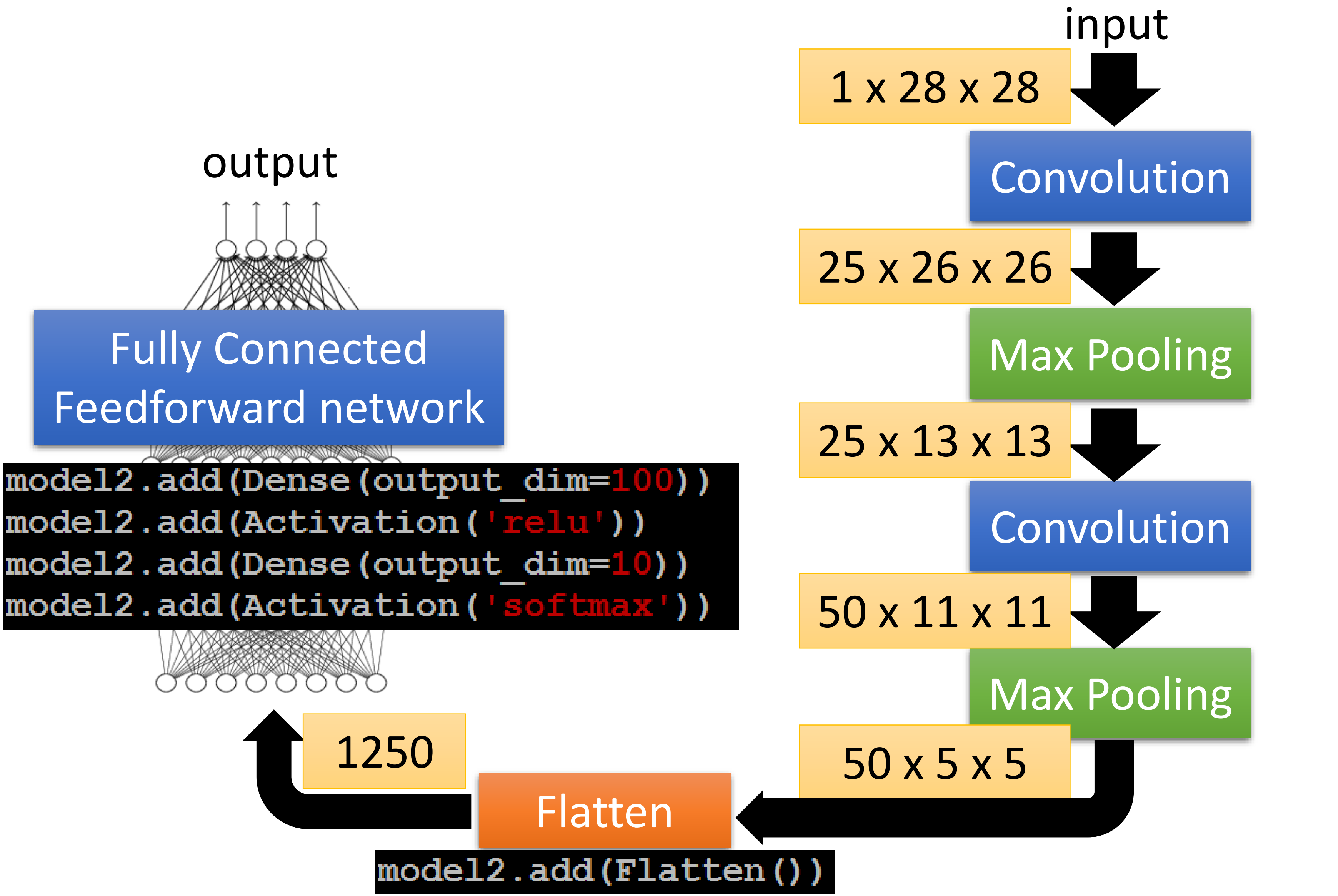
50 x 11 x 11

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

50 x 5 x 5

CNN in Keras

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



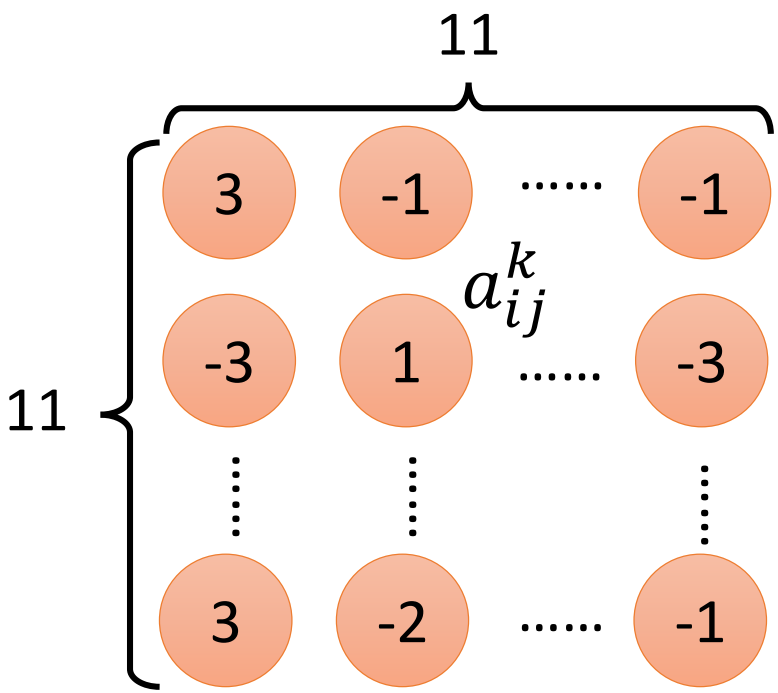
Live Demo

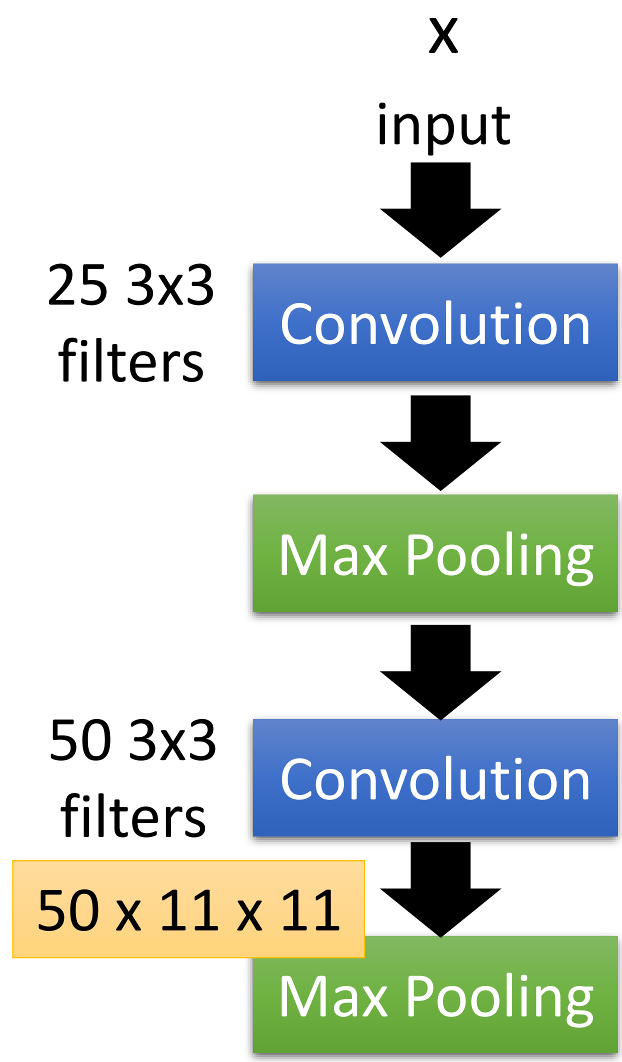
What does CNN learn?

The output of the k-th filter is a 11 x 11 matrix.

Degree of the activation of the k-th filter: $a^k = \sum_{i=1}^{11} \sum_{j=1}^{11} a_{ij}^k$

$x^* = arg \max_x a^k$ (gradient ascent)



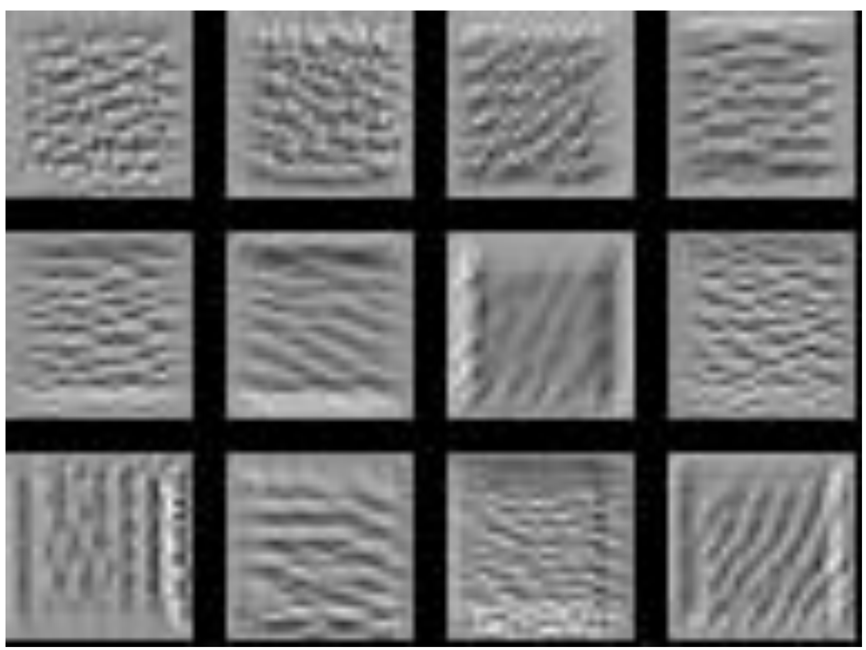


What does CNN learn?

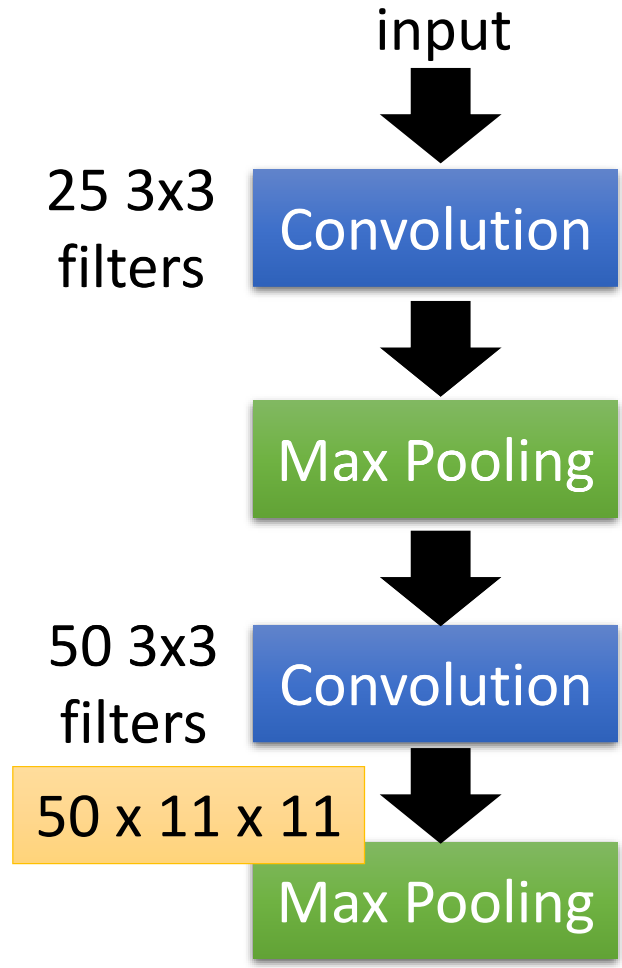
The output of the k-th filter is a 11 x 11 matrix.

Degree of the activation of the k-th filter: $$a^k = \sum_{i=1}^{11}\sum_{j=1}^{11} a_{ij}^k$$

$x^* = arg\max_{x} a^k$ (gradient ascent)



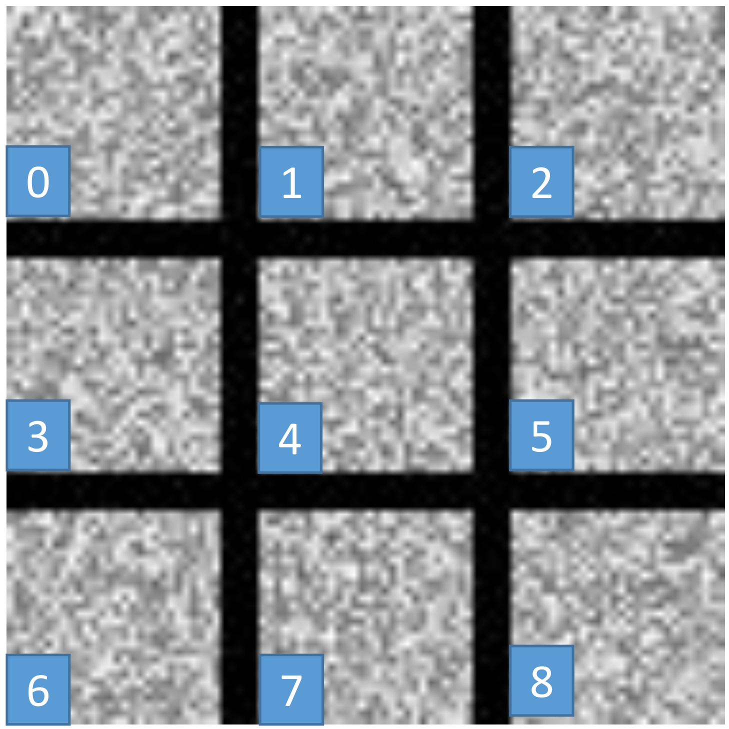
For each filter

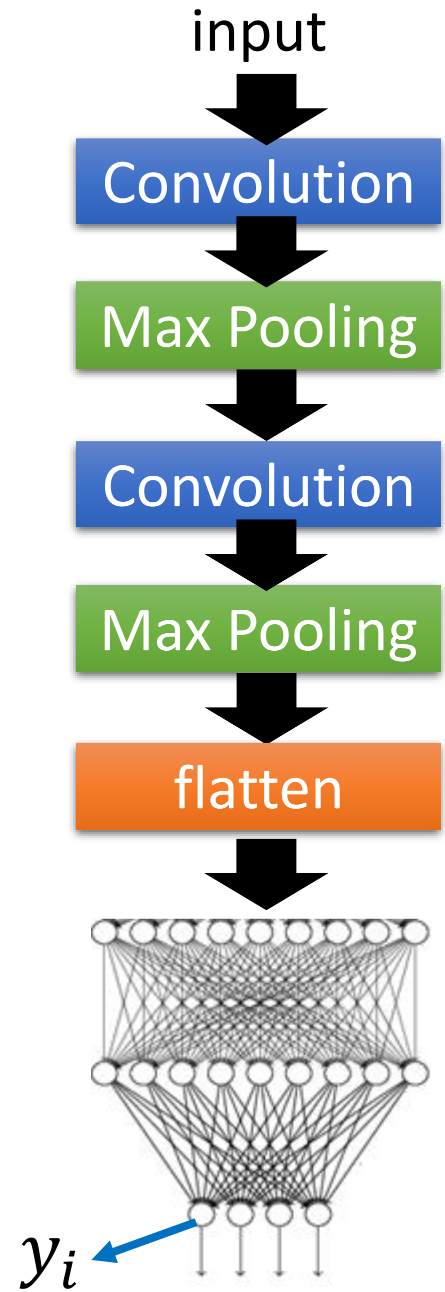


What does CNN learn?

$$x^* = arg\max_x y^i$$

Can we see digits?





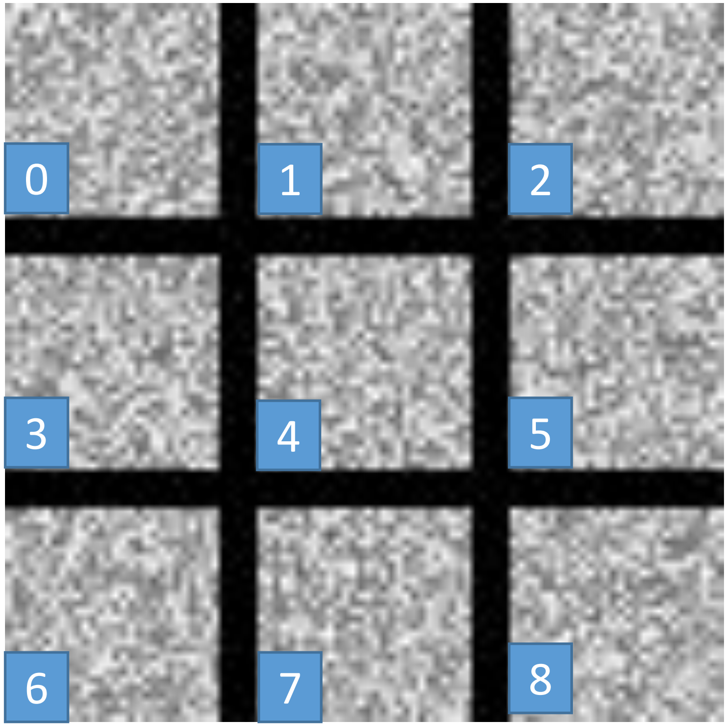
Deep Neural Networks are Easily Fooled
https://www.youtube.com/watch?v=M2IebCN9Ht4

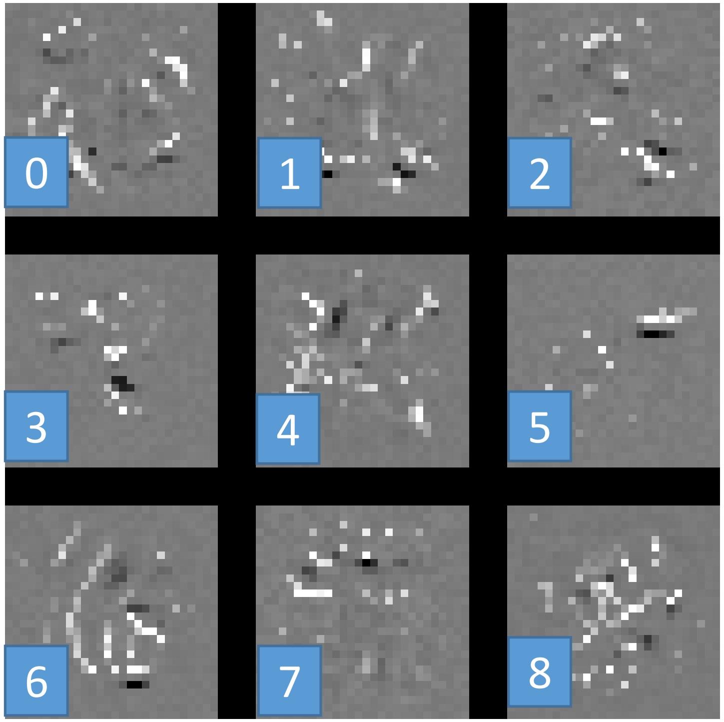
What does CNN learn?

Over all pixel values

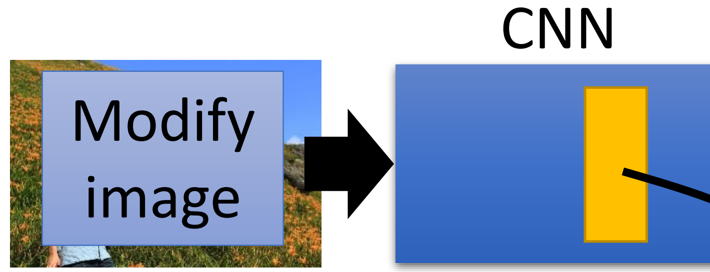
$$x^* = arg \max_x y^i$$

$$x^* = arg \max_x \left(y^i + \boxed{\sum_{i,j} |x_{ij}|} \right)$$





Deep Dream



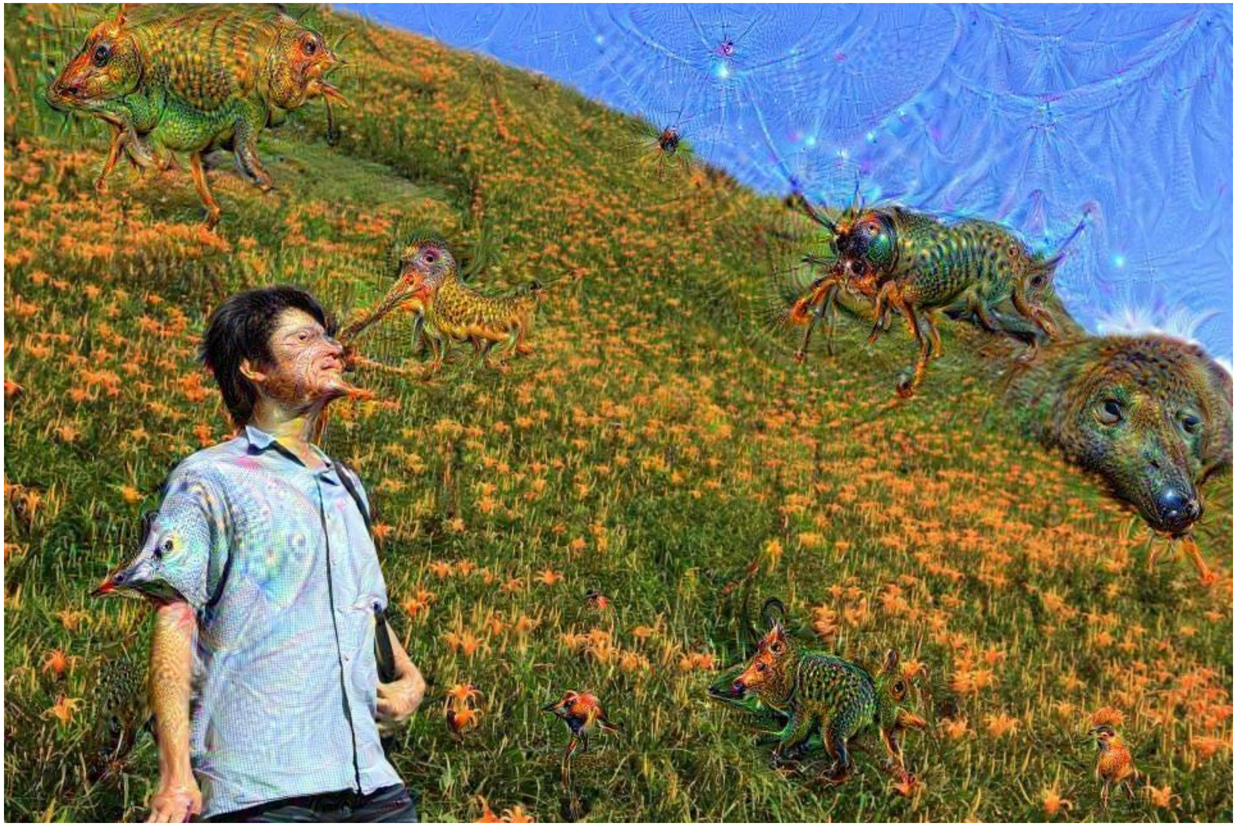
$$\begin{bmatrix} 3.9 \\ -1.5 \\ 2.3 \\ \vdots \end{bmatrix}$$

- Given a photo, machine adds what it sees ……

http://deepdreamgenerator.com/

Deep Dream

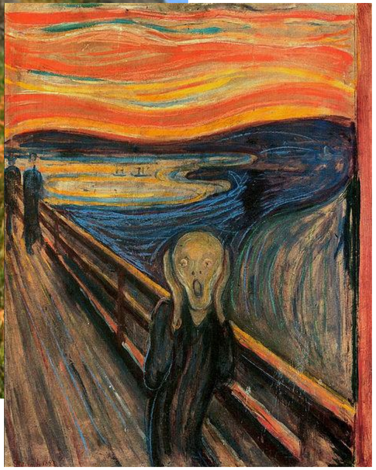
- Given a photo, machine adds what it sees ……



http://deepdreamgenerator.com/

Deep Style

- Given a photo, make its style like famous paintings



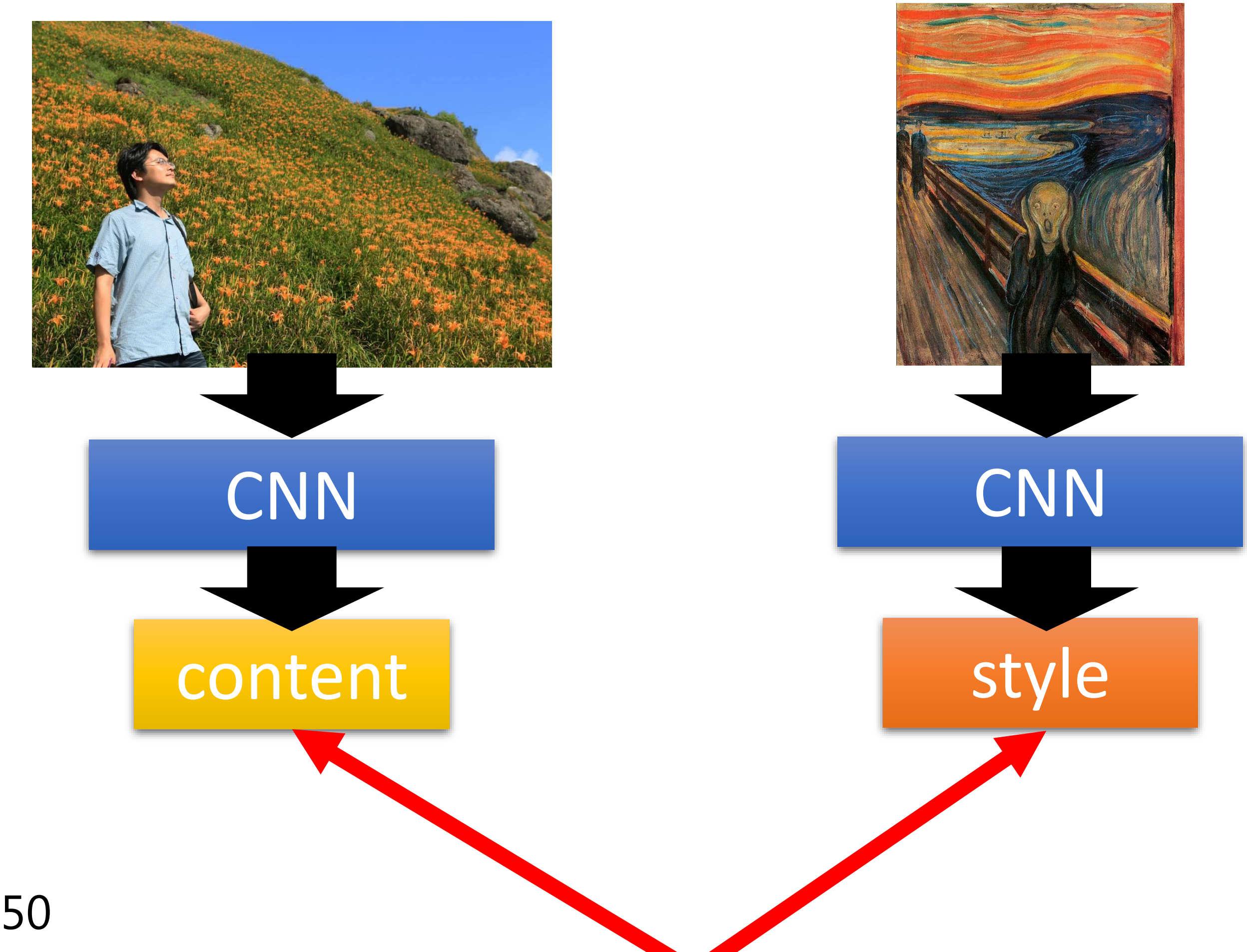
https://dreamscopeapp.com/

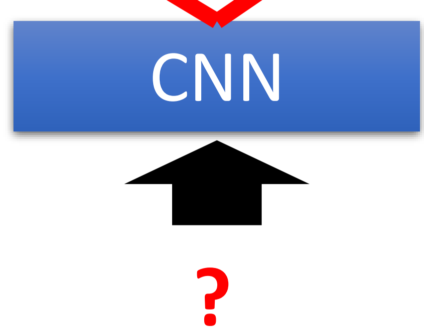
Deep Style

- Given a photo, make its style like famous paintings

https://dreamscopeapp.com/

Deep Style





A Neural Algorithm of Artistic Style
https://arxiv.org/abs/1508.06576

More Application: Playing Go

19 x 19 matrix (image)

Network

Next move (19 x 19 positions)

19 x 19 vector

Black: 1

white: -1

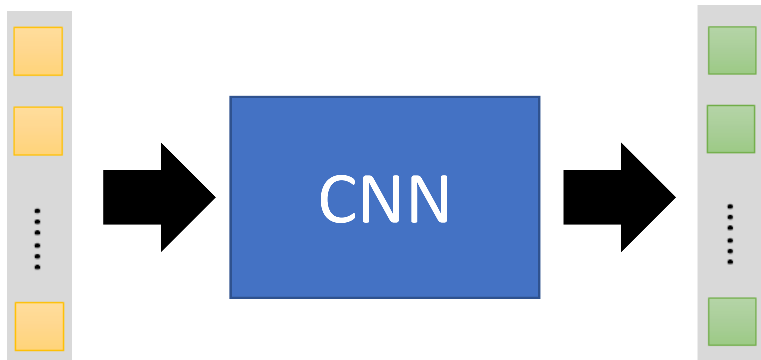
none: 0

Fully-connected feedforward network can be used

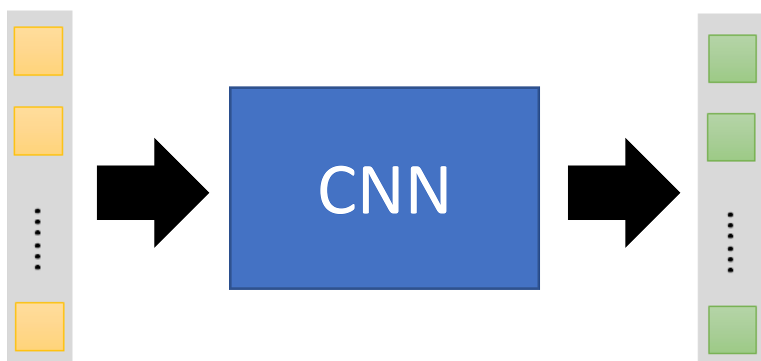
But CNN performs much better.

More Application: Playing Go

Training: record of previous plays 黑: 5之五 → 白: 天元 → 黑: 五之5 …



Target:
"天元" = 1
else = 0

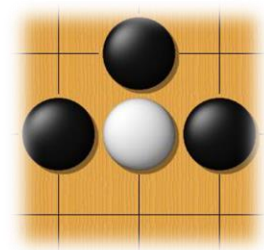


Target:
"五之 5" = 1
else = 0

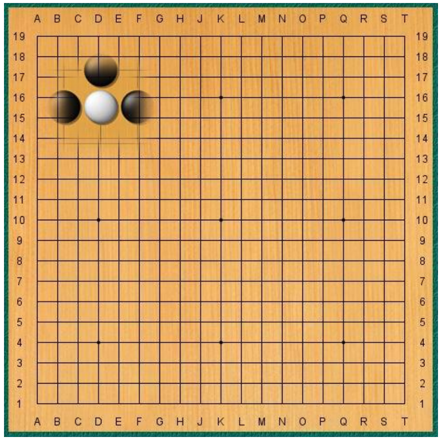
Why CNN for playing Go?

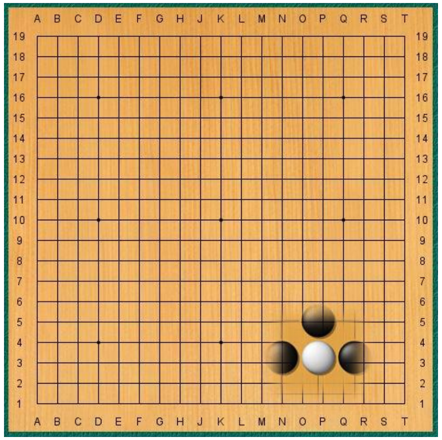
- Some patterns are much smaller than the whole image

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, with a different bias for each position, and applies a softmax function. The … Extended Data Table 3 additionally show the results of training with $k = 128$, 256 and 384 filters.

Alpha Go does not use Max Pooling ……

Variants of Neural Networks

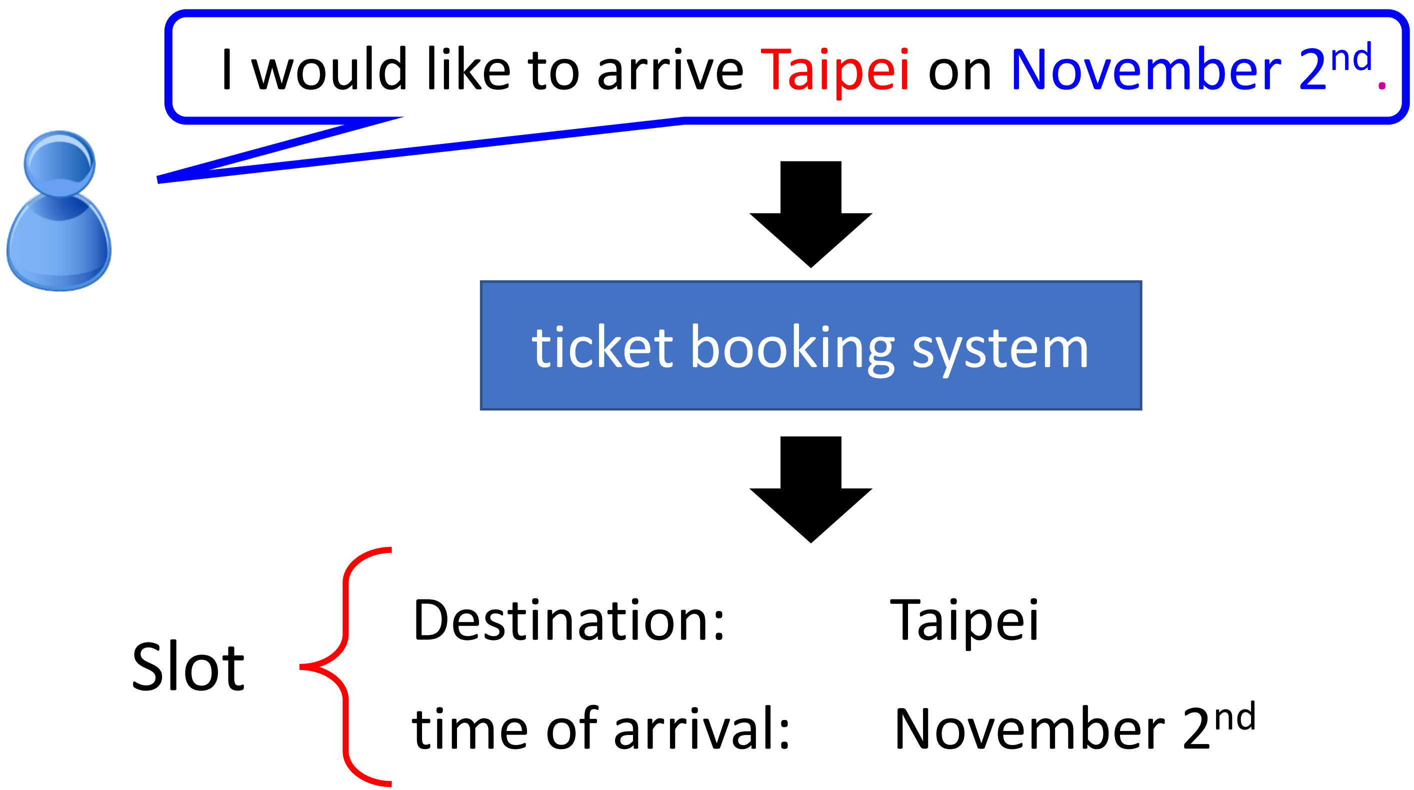
Convolutional Neural Network (CNN)

Recurrent Neural Network (RNN)

Neural Network with Memory

Example Application

- Slot Filling

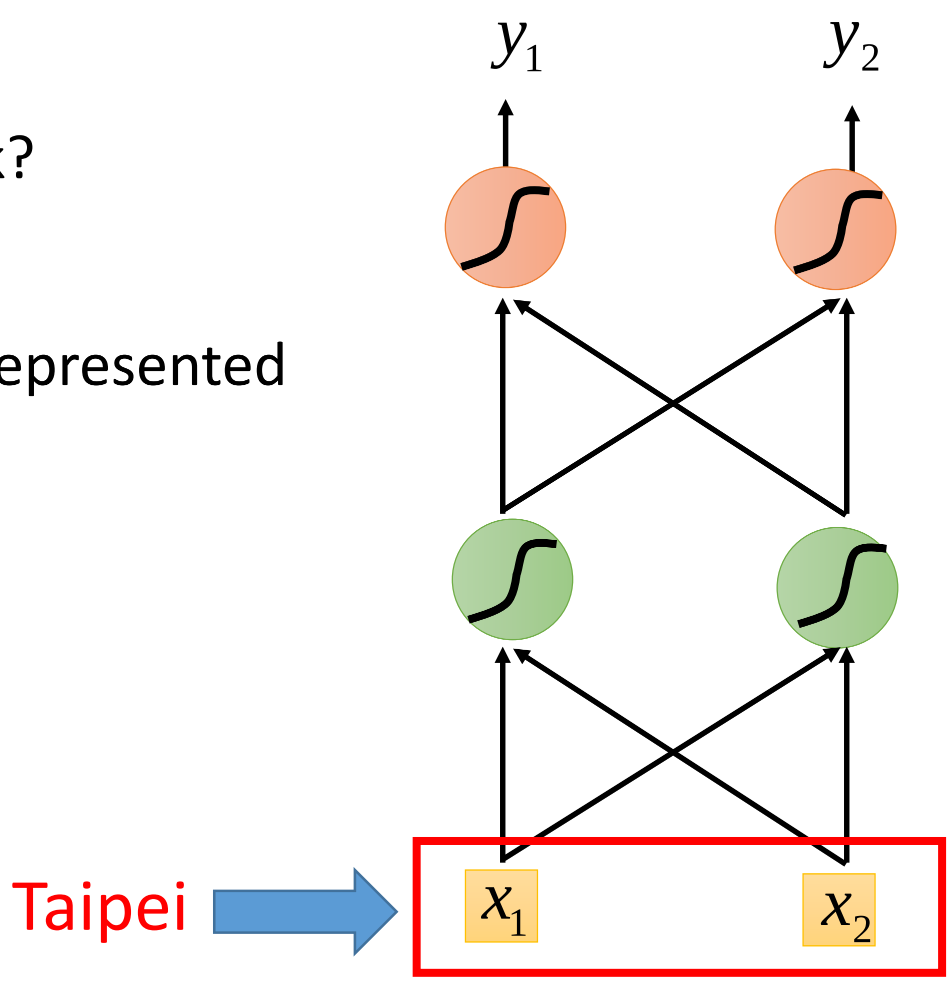


Example Application

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]

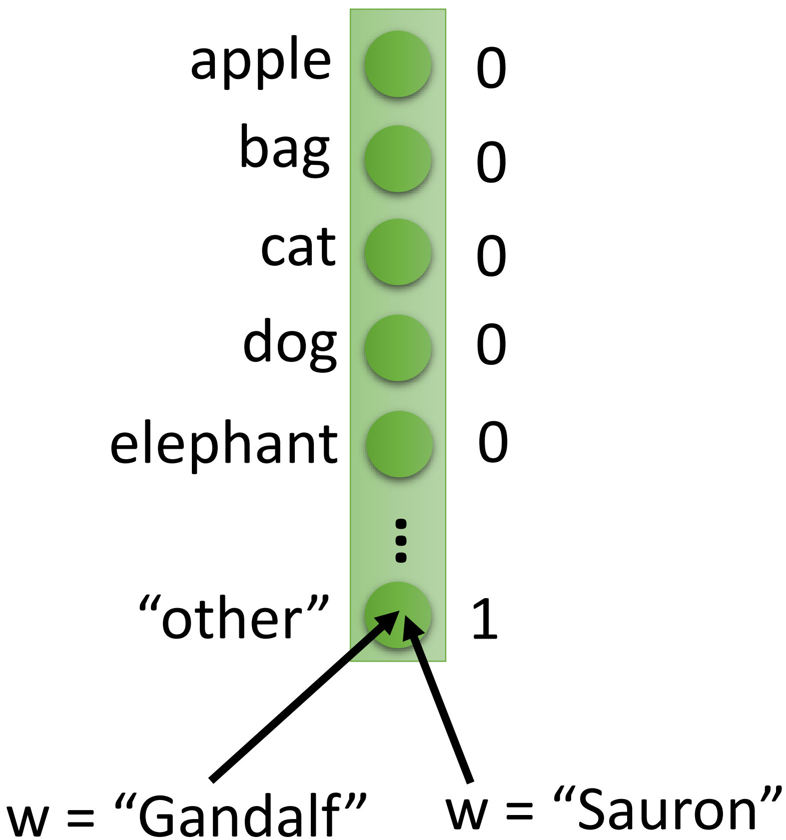
cat = [0 0 1 0 0]

dog = [0 0 0 1 0]

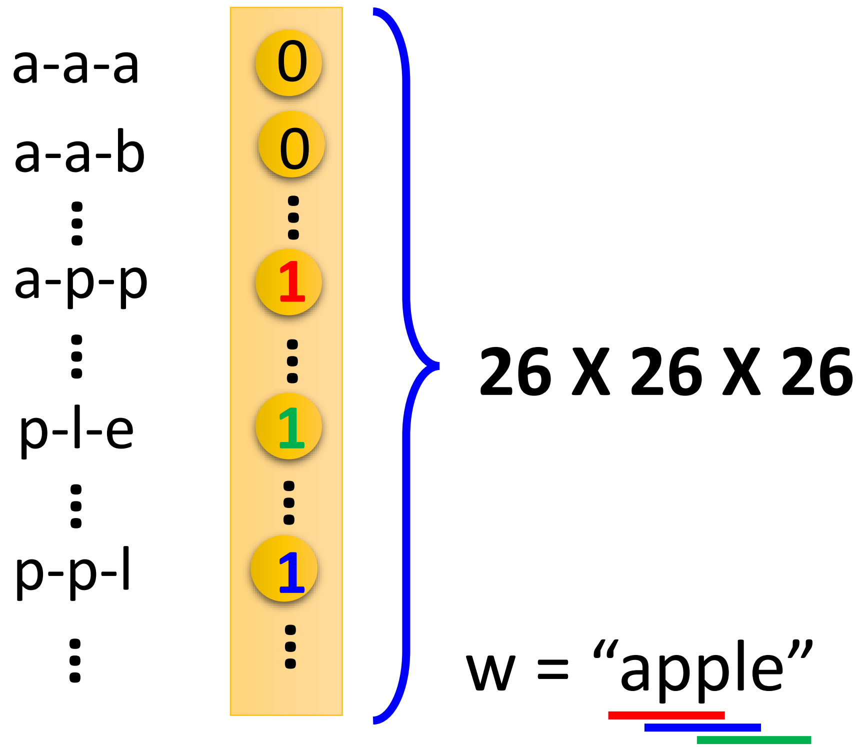
elephant = [0 0 0 0 1]

Beyond 1-of-N encoding

Dimension for "Other"



Word hashing



Example Application

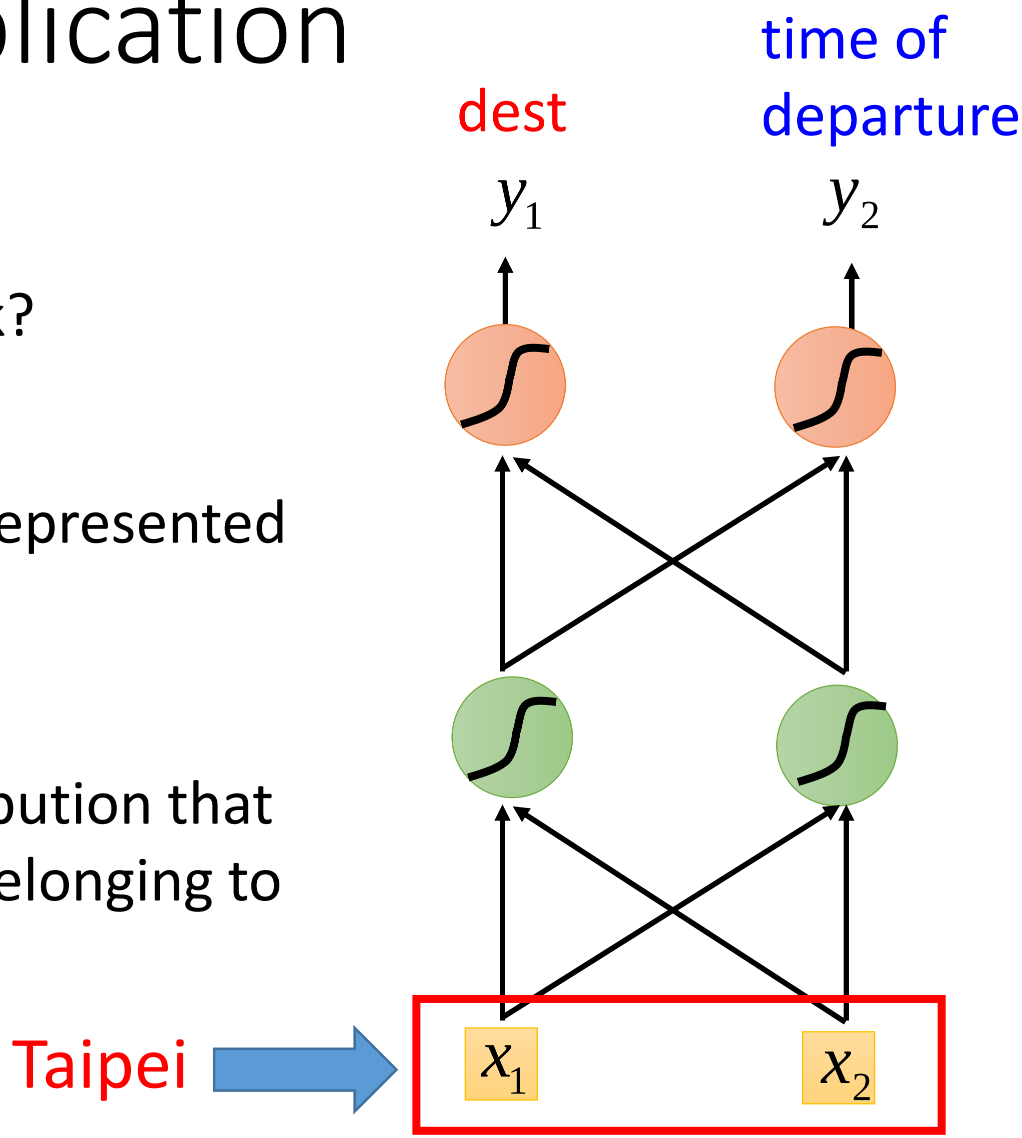
Solving slot filling by Feedforward network?

Input: a word

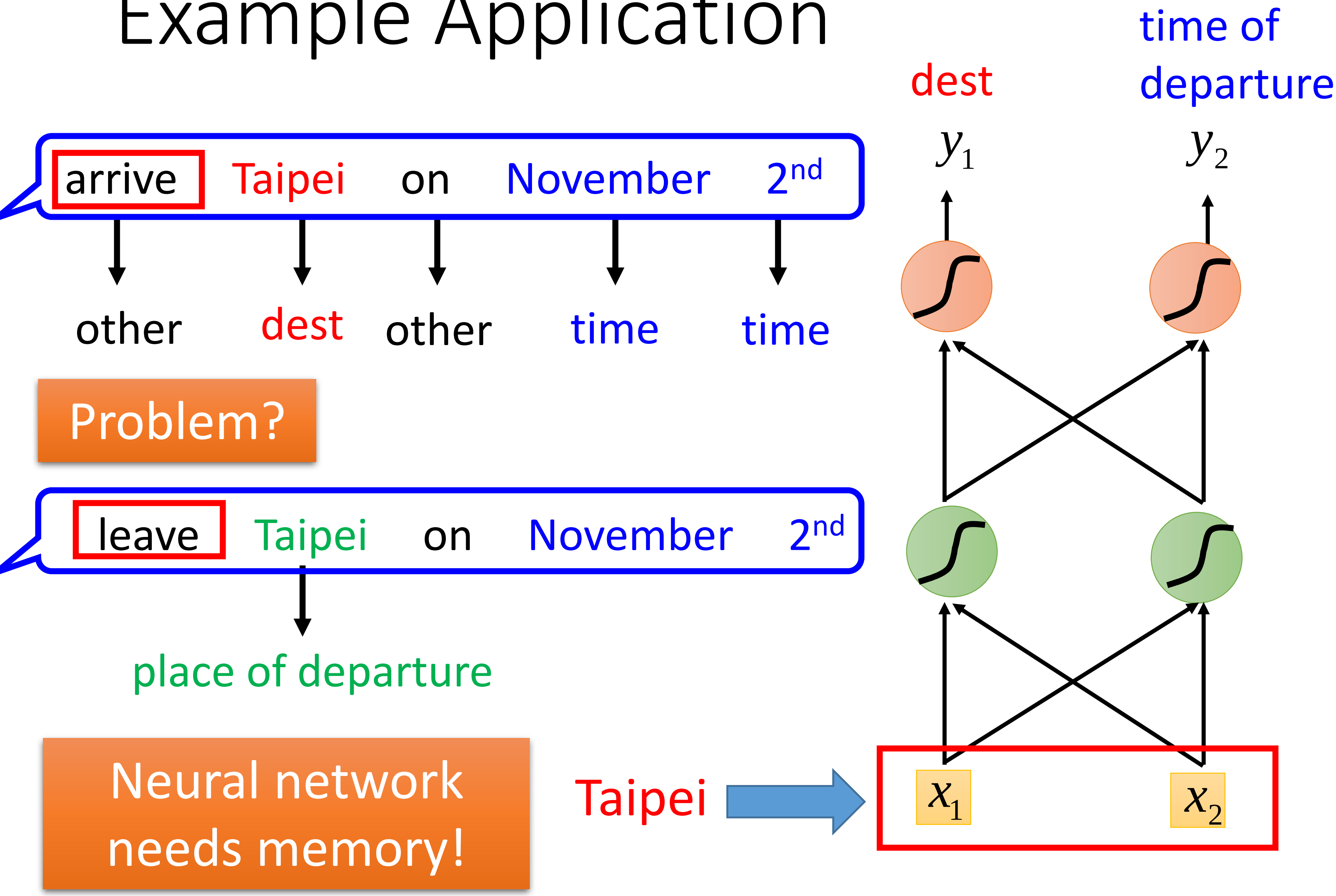
(Each word is represented as a vector)

Output:

Probability distribution that the input word belonging to the slots



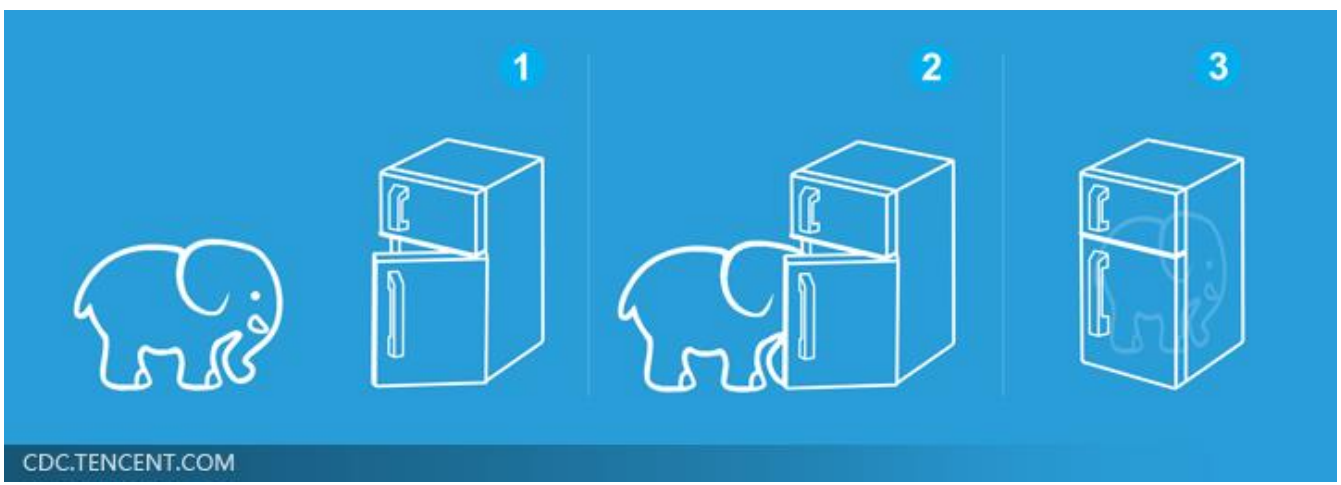
Example Application



Three Steps for Deep Learning

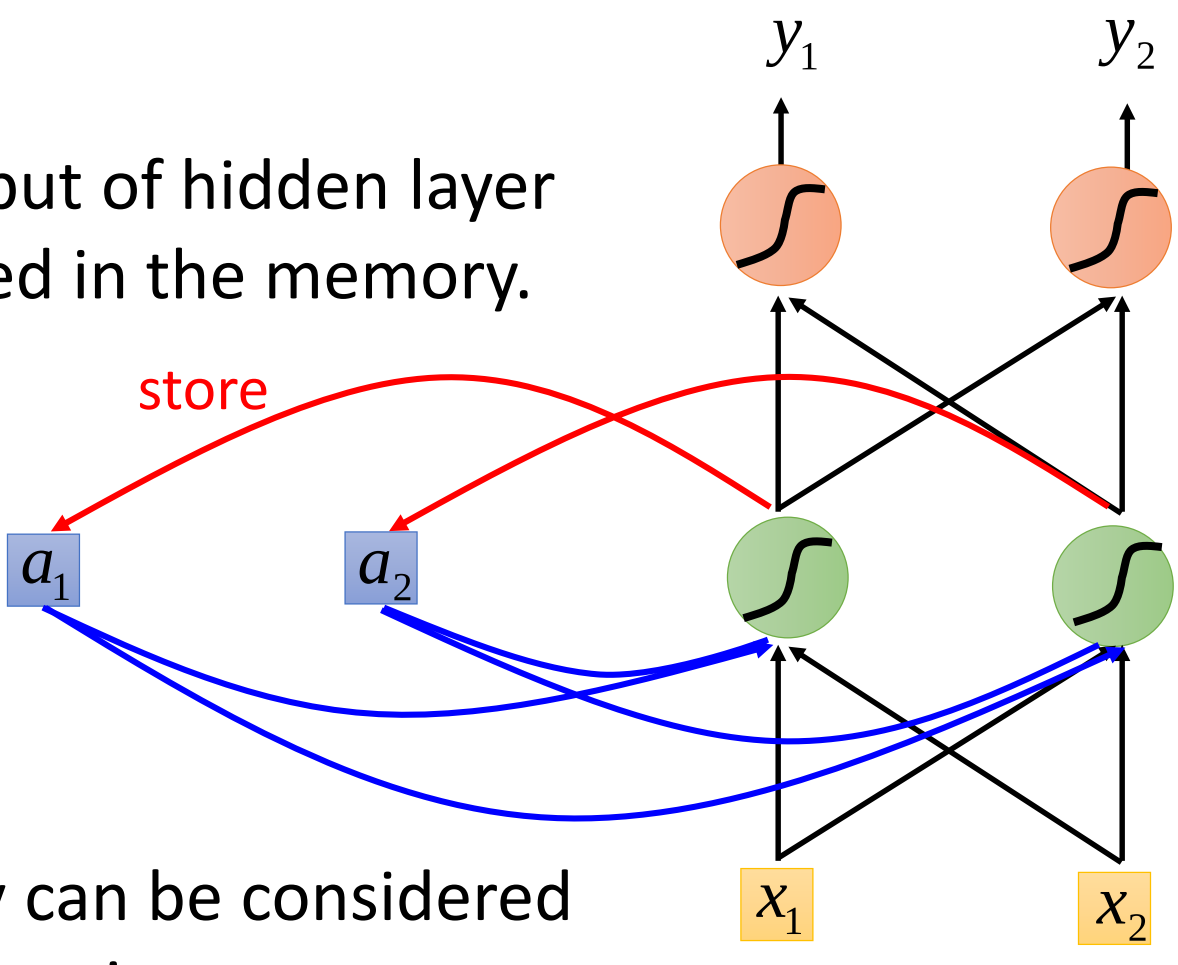
Step 1: Recurrent Neural Network → Step 2: goodness of function → Step 3: pick the best function

Deep Learning is so simple ……



Recurrent Neural Network (RNN)

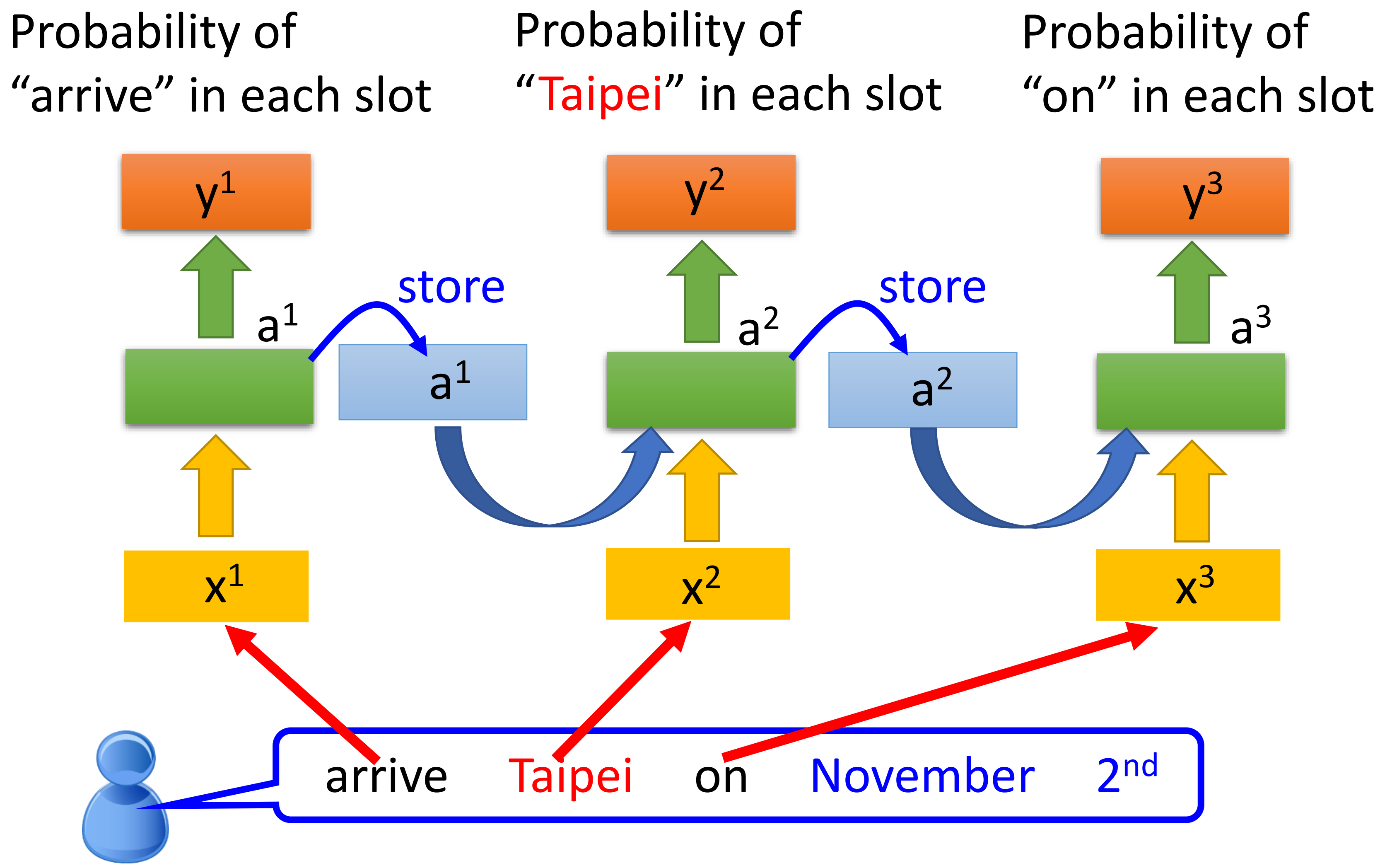
The output of hidden layer are stored in the memory.



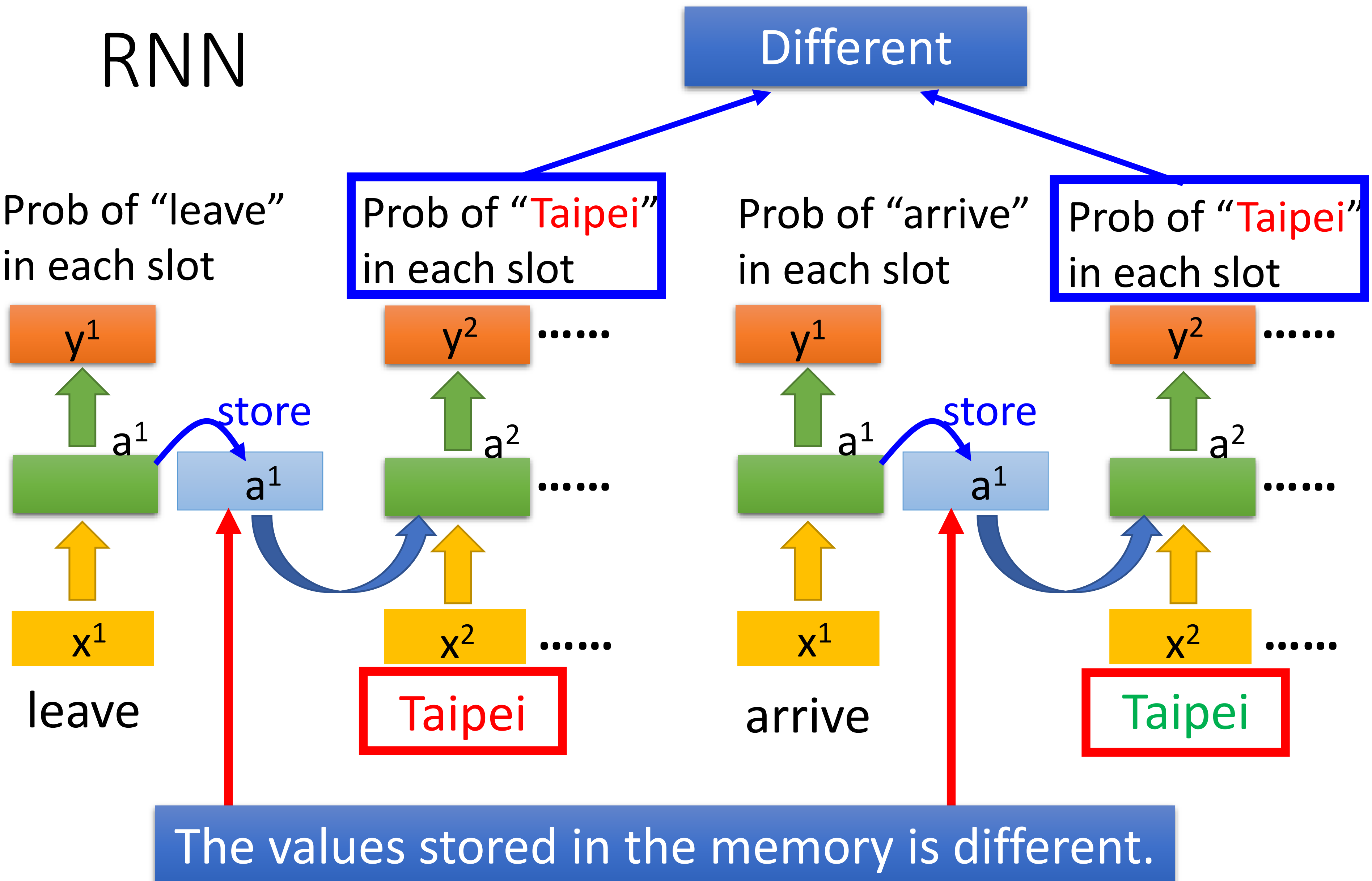
Memory can be considered as another input.

RNN

The same network is used again and again.

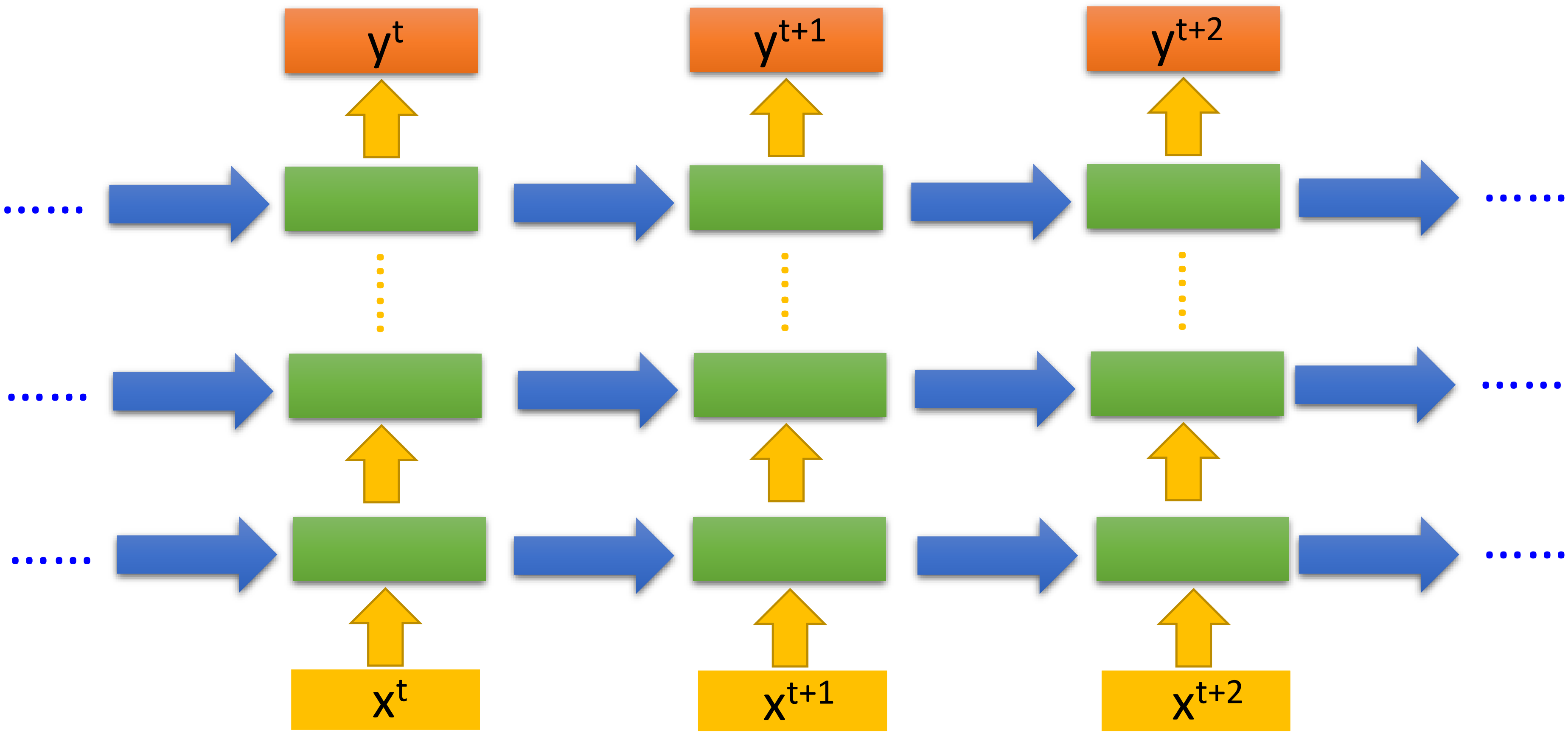


RNN

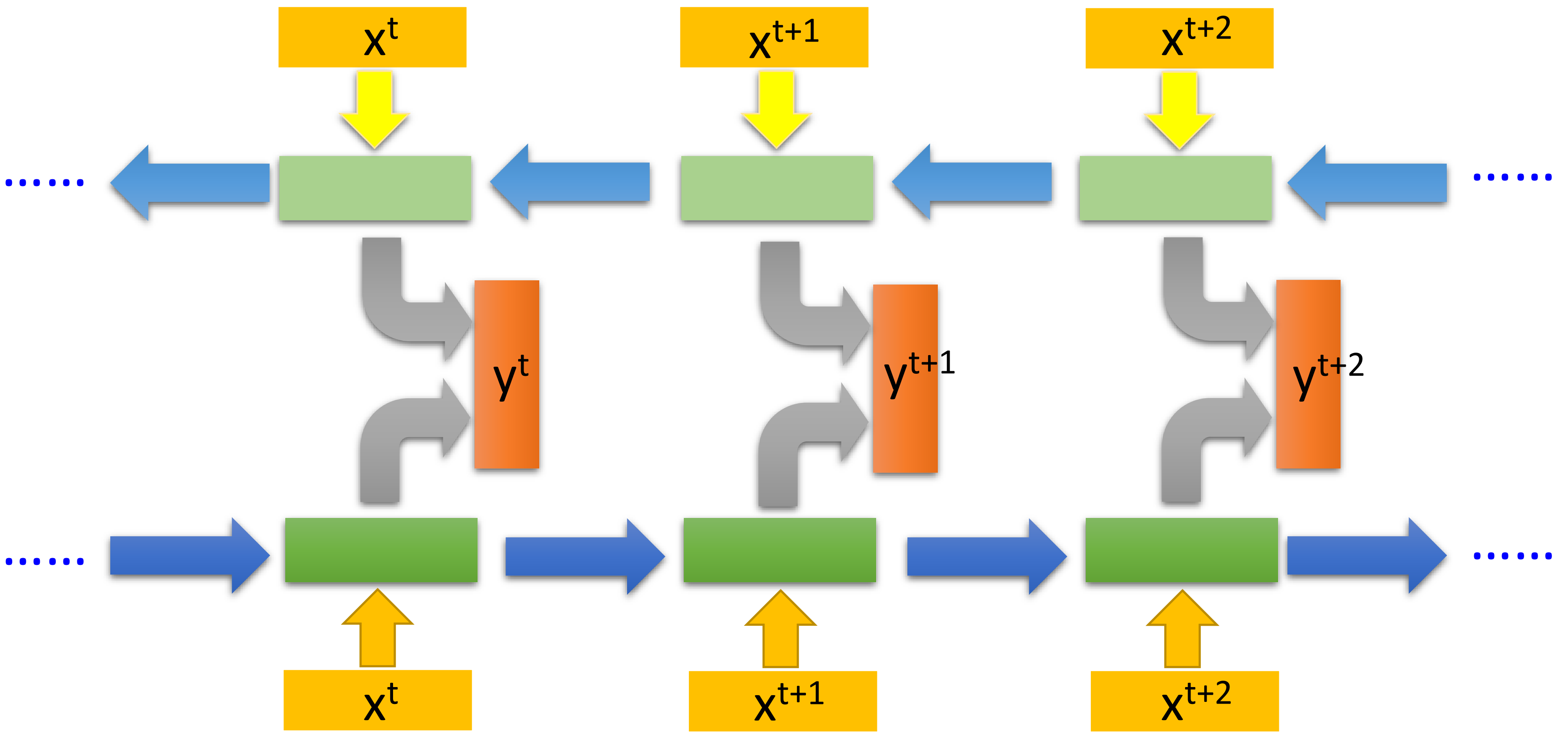


Different
Prob of "leave" in each slot
Prob of "Taipei" in each slot
Prob of "arrive" in each slot
Prob of "Taipei" in each slot
y1
y2
......
a1
store
a2
x1
x2
leave
Taipei
arrive
Taipei
The values stored in the memory is different.

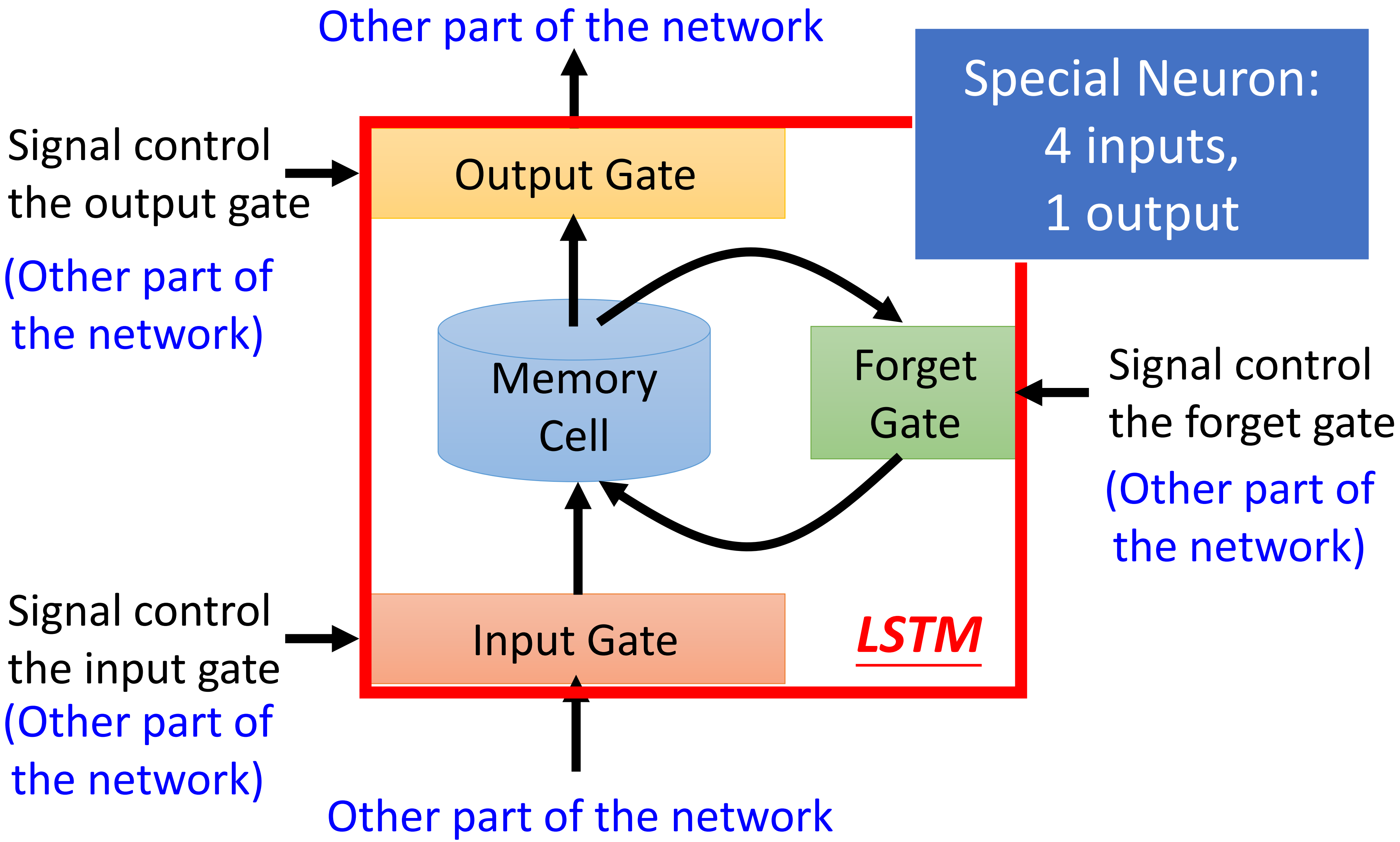
Of course it can be deep …

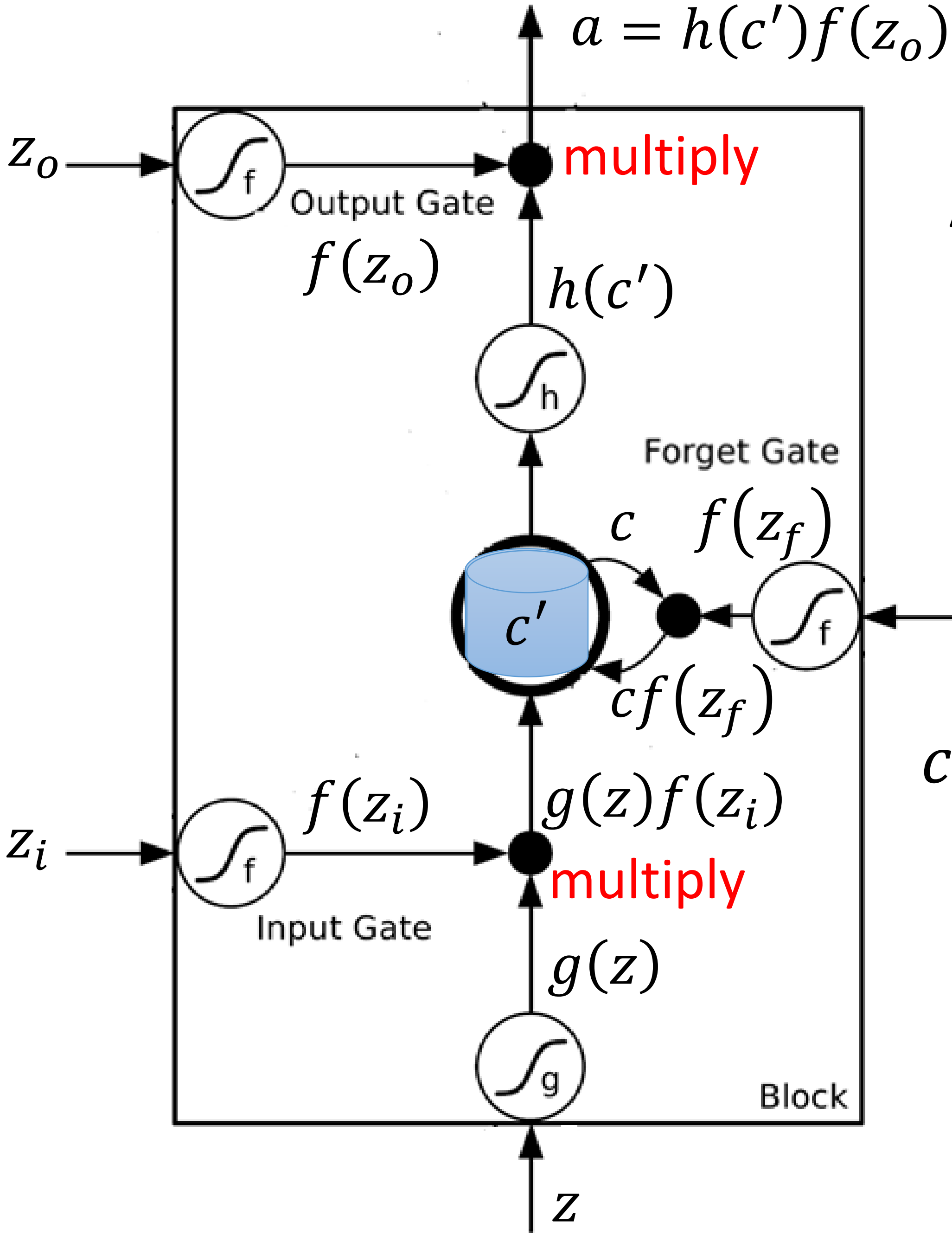


Bidirectional RNN



Long Short-term Memory (LSTM)



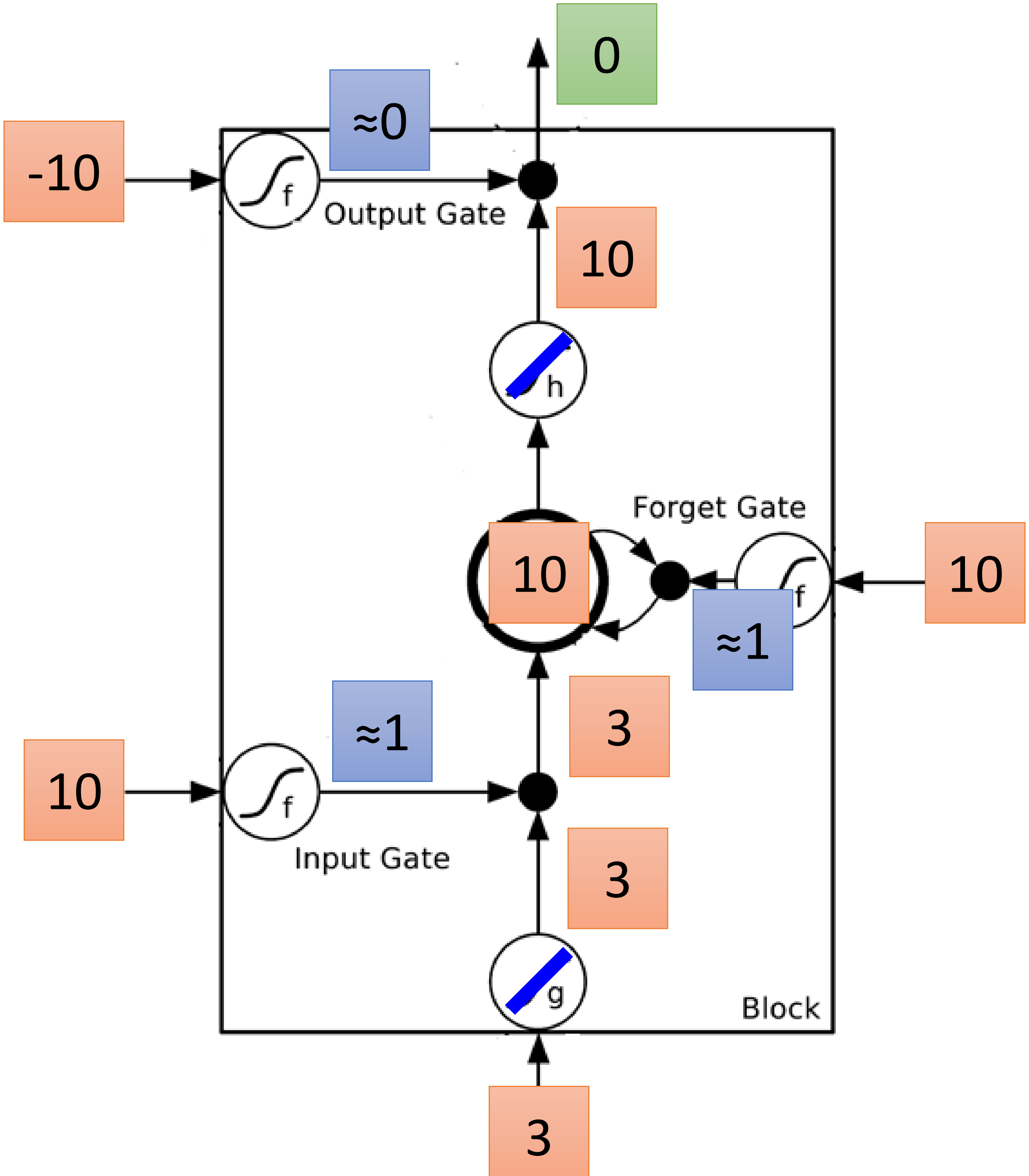


Activation function f is usually a sigmoid function

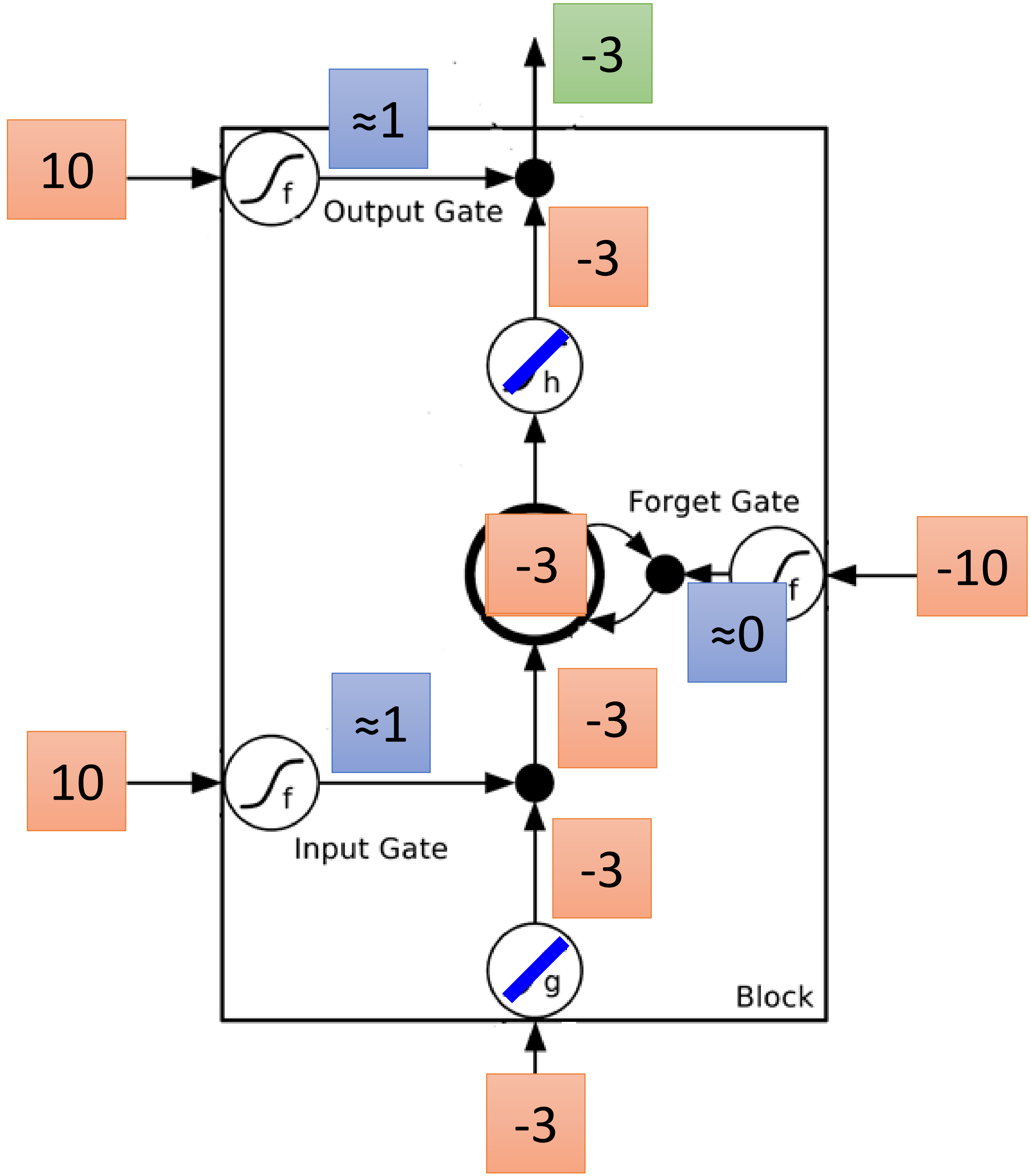
Between 0 and 1

Mimic open and close gate

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

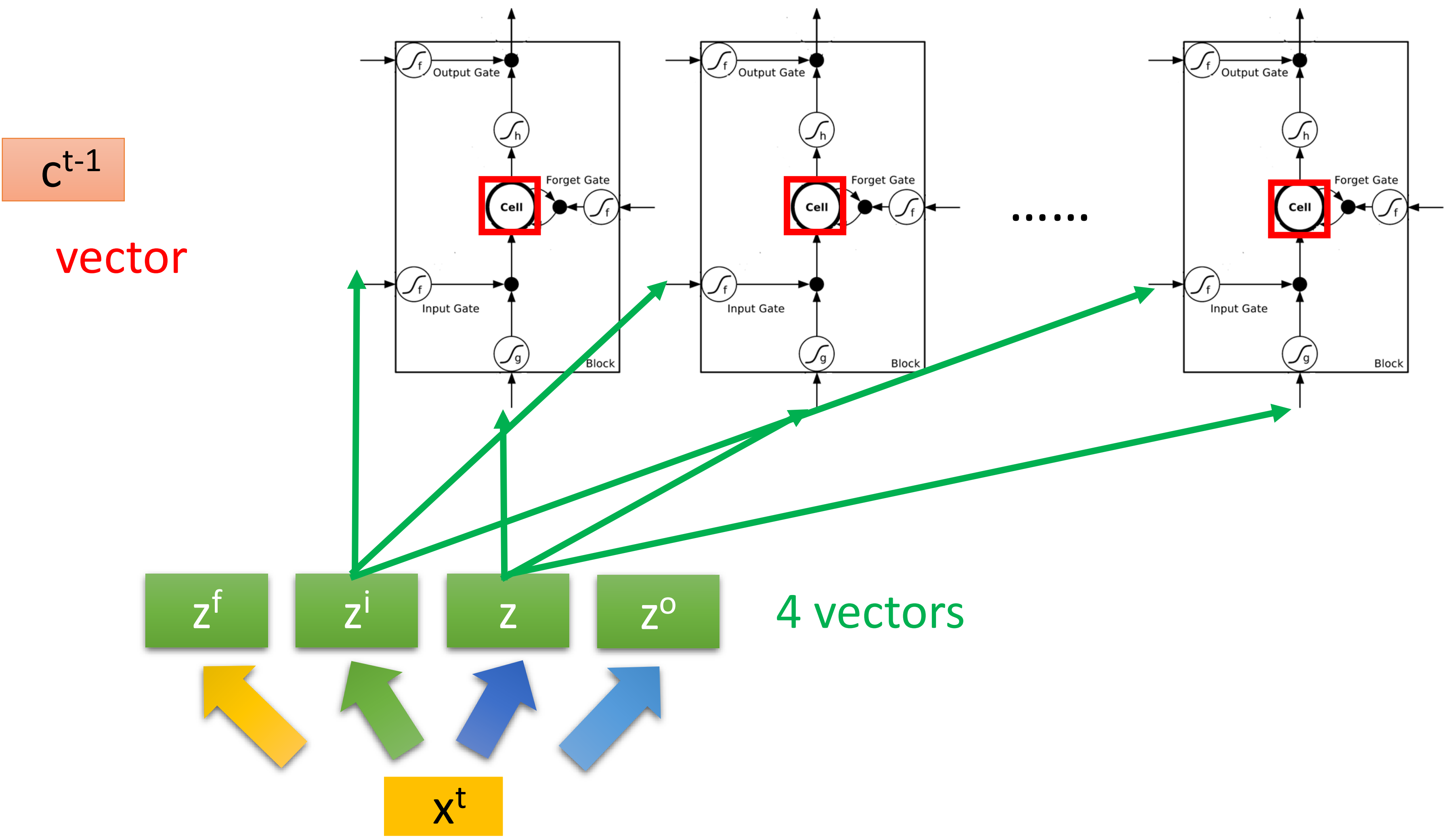


0
≈0
-10
Output Gate
10
h
Forget Gate
10
10
≈1
≈1
10
3
Input Gate
3
g
Block
3



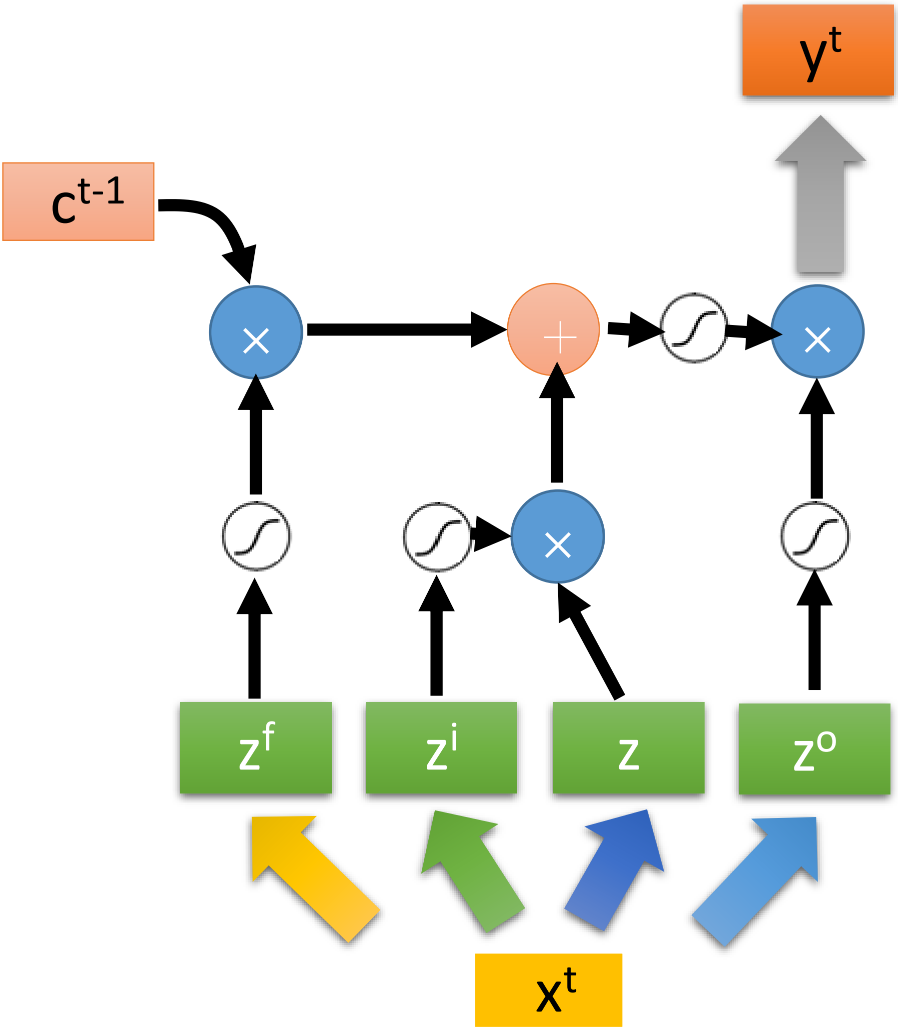
-3
≈1
10
Output Gate
-3
h
Forget Gate
-3
-10
≈0
≈1
-3
10
Input Gate
-3
g
Block
-3

LSTM

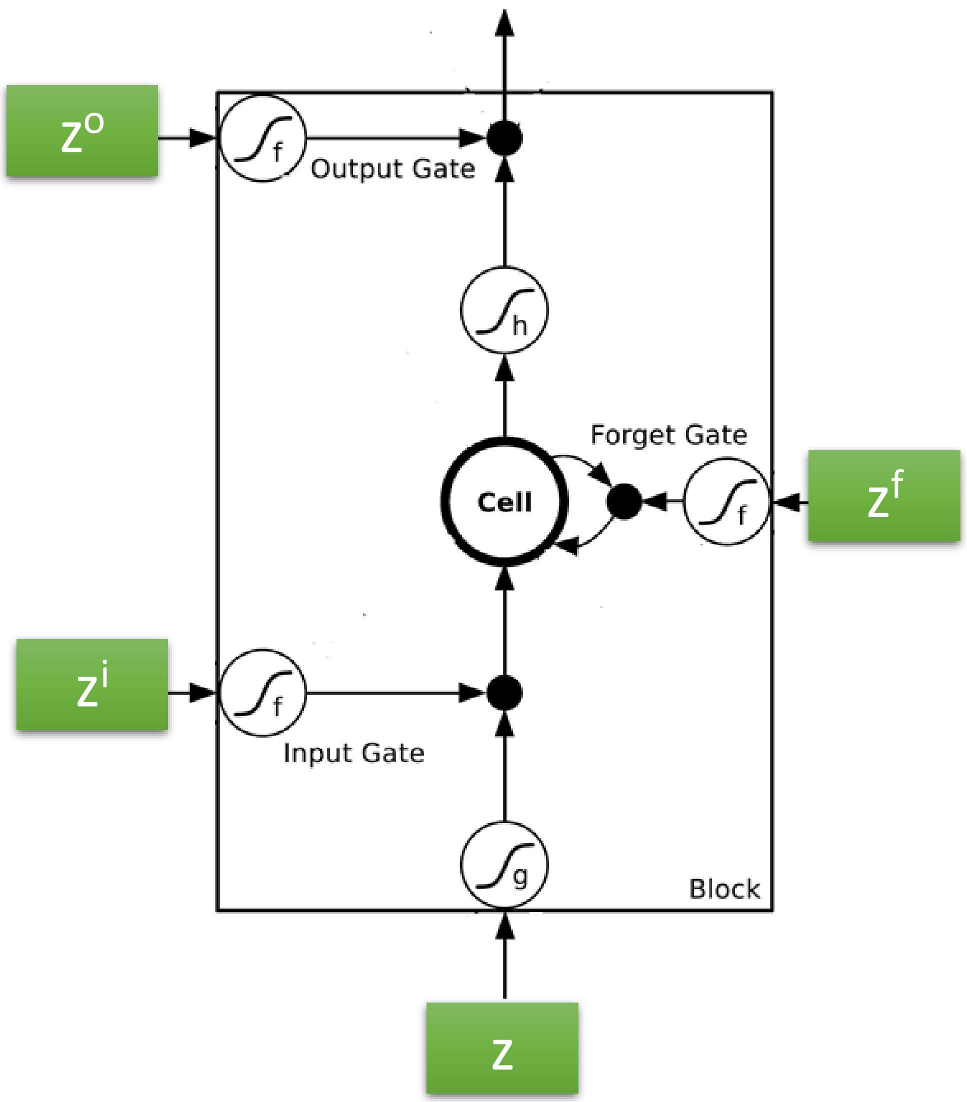


c^{t-1}
vector
Output Gate
Forget Gate
Input Gate
Cell
Block
......
z^f
z^i
z
z^o
4 vectors
x^t

LSTM



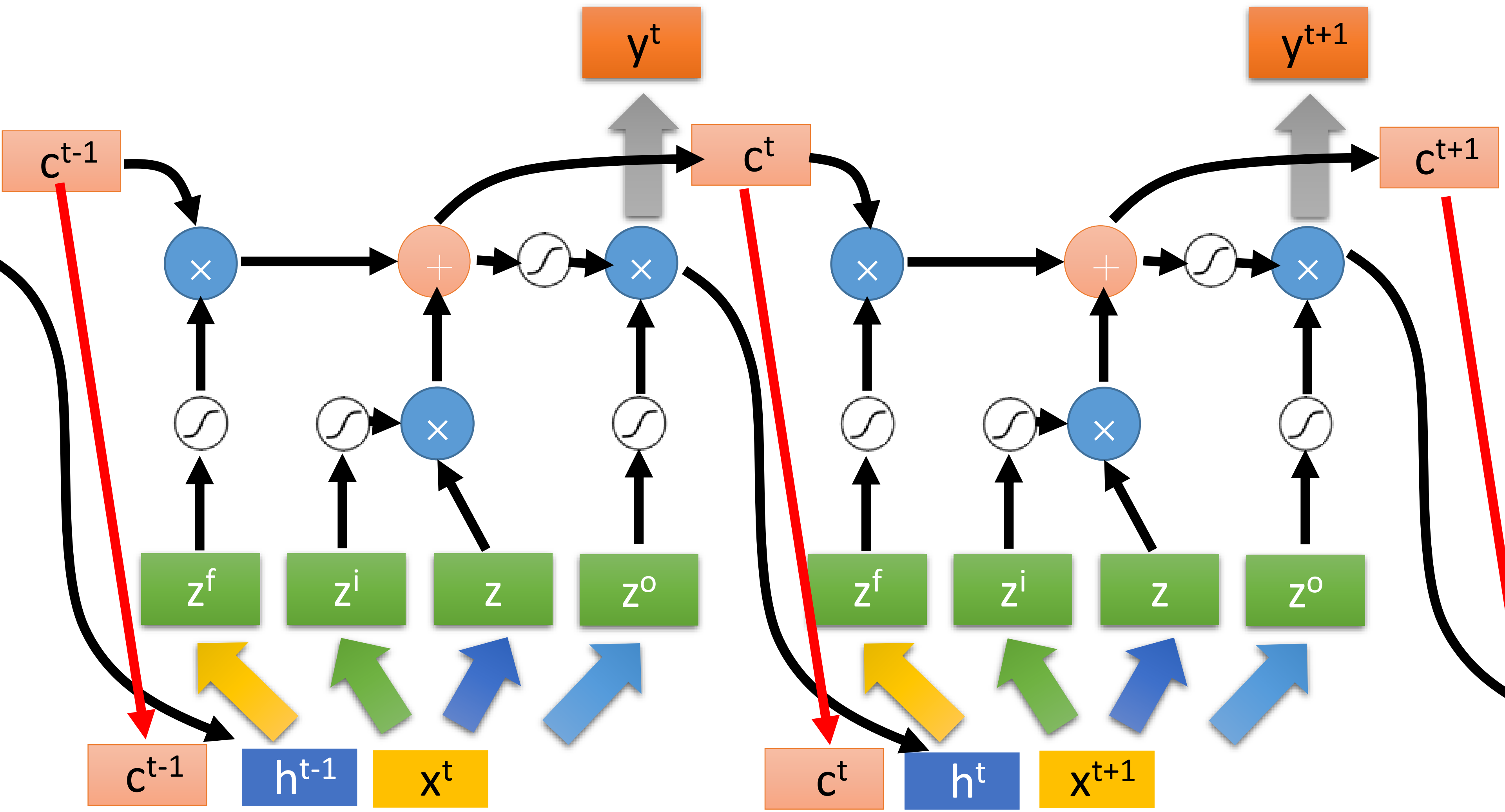
y^t
c^{t-1}
z^f
z^i
z
z^o
x^t



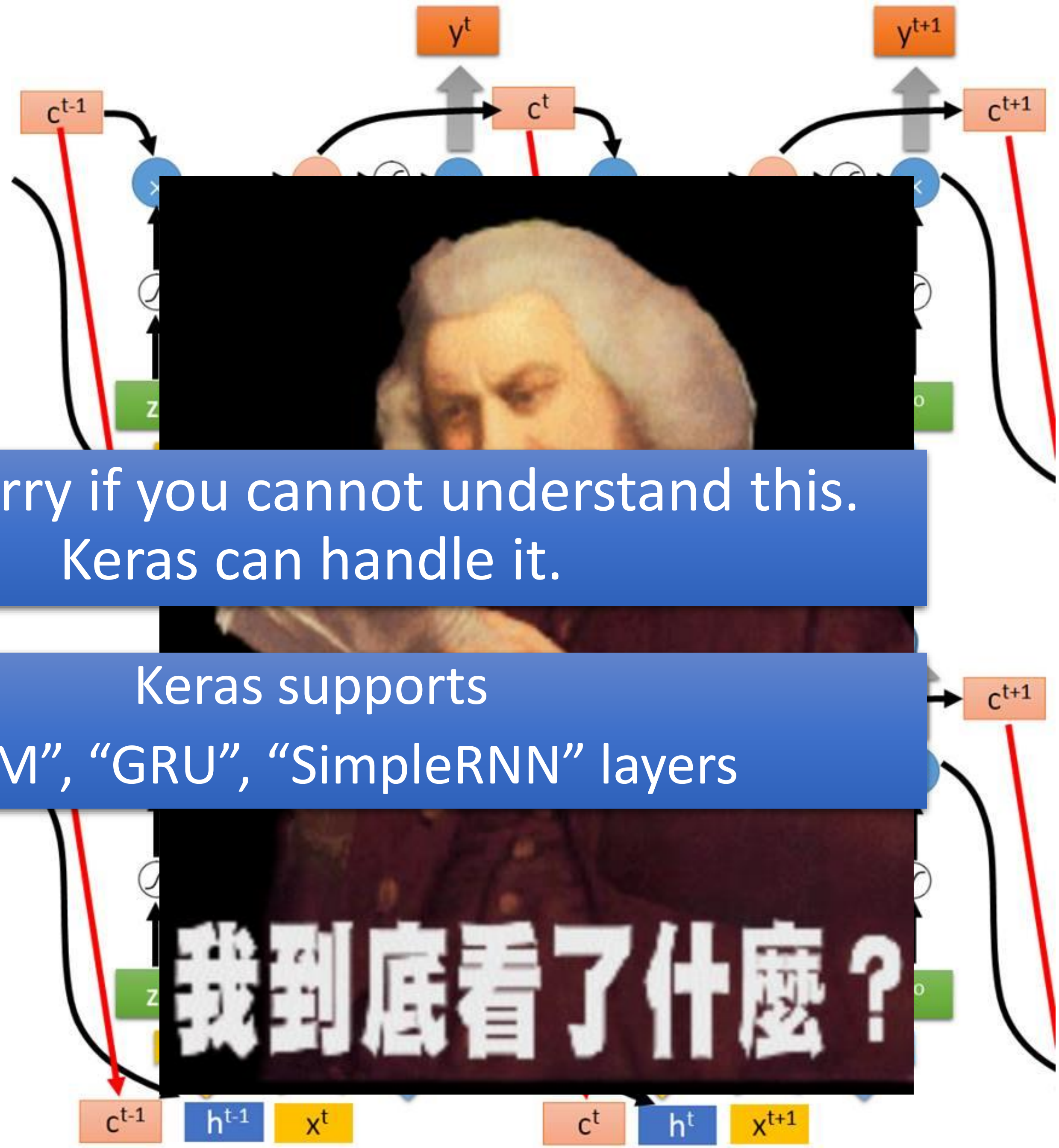
z^o
Output Gate
Forget Gate
Cell
z^f
z^i
Input Gate
Block
z

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/1442696762713l.gif

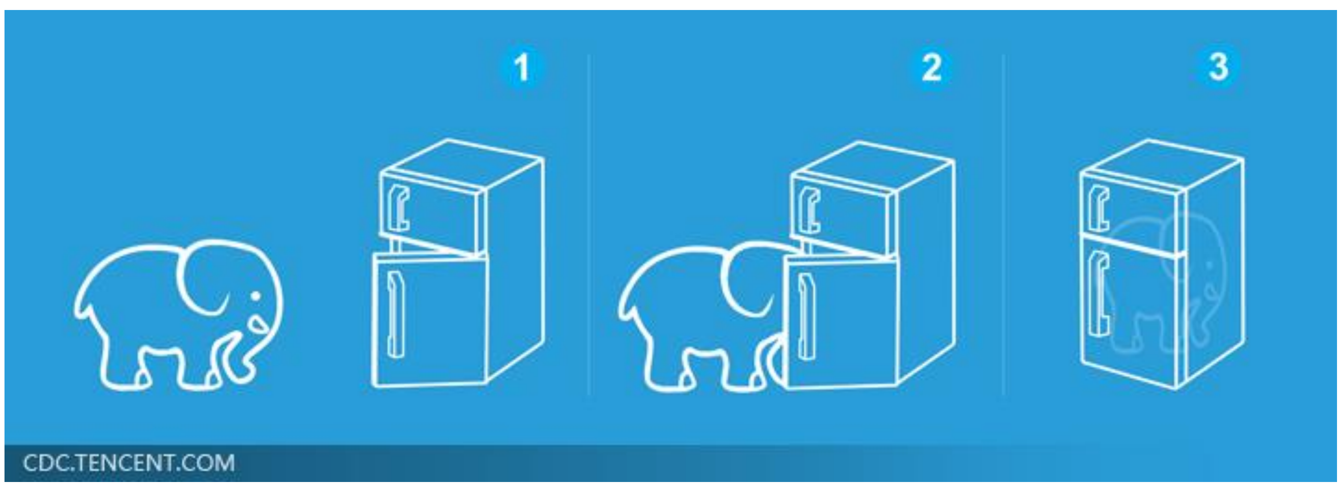
Three Steps for Deep Learning

Step 1: define a set of function

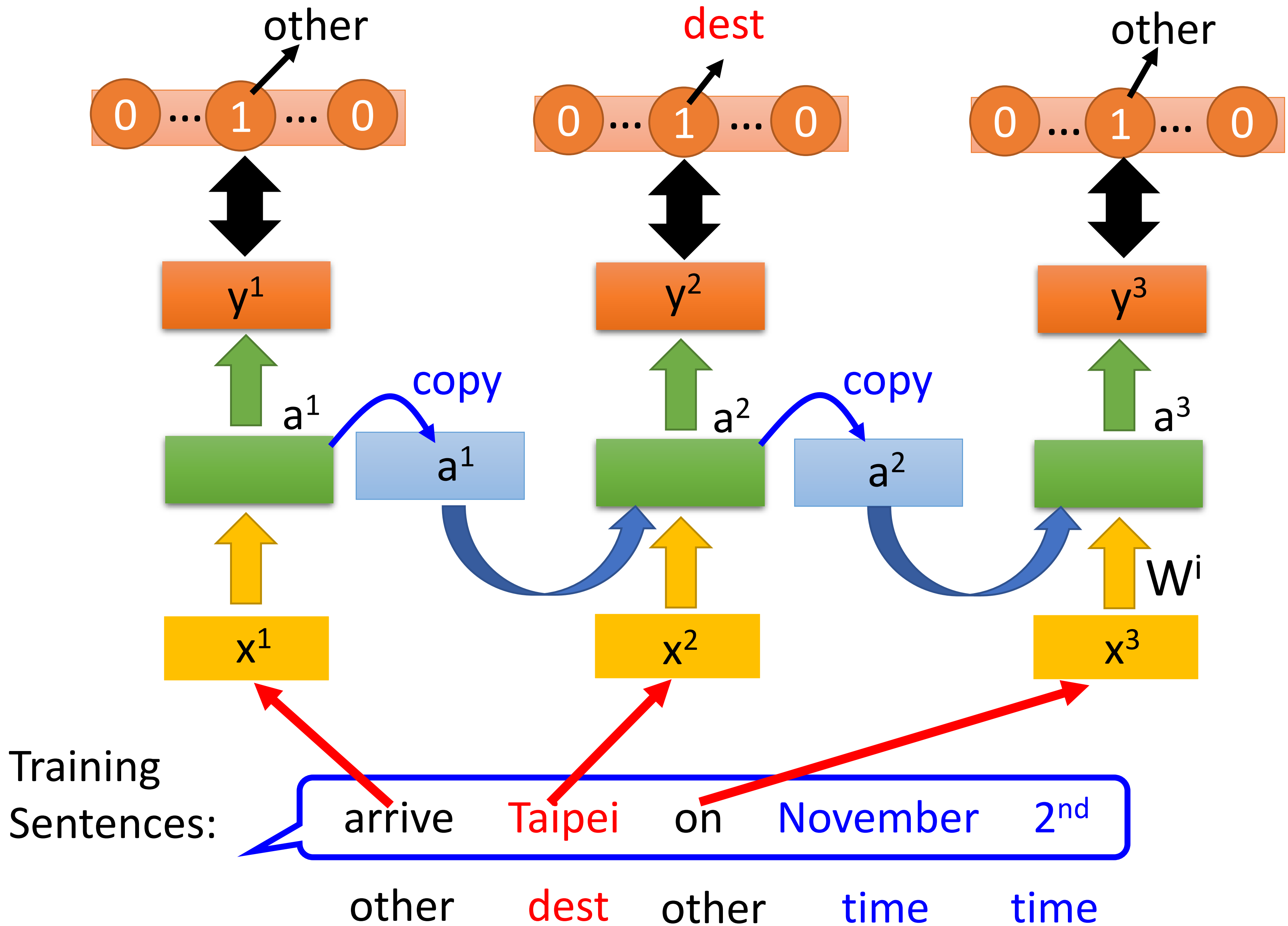
Step 2: goodness of function

Step 3: pick the best function

Deep Learning is so simple ……



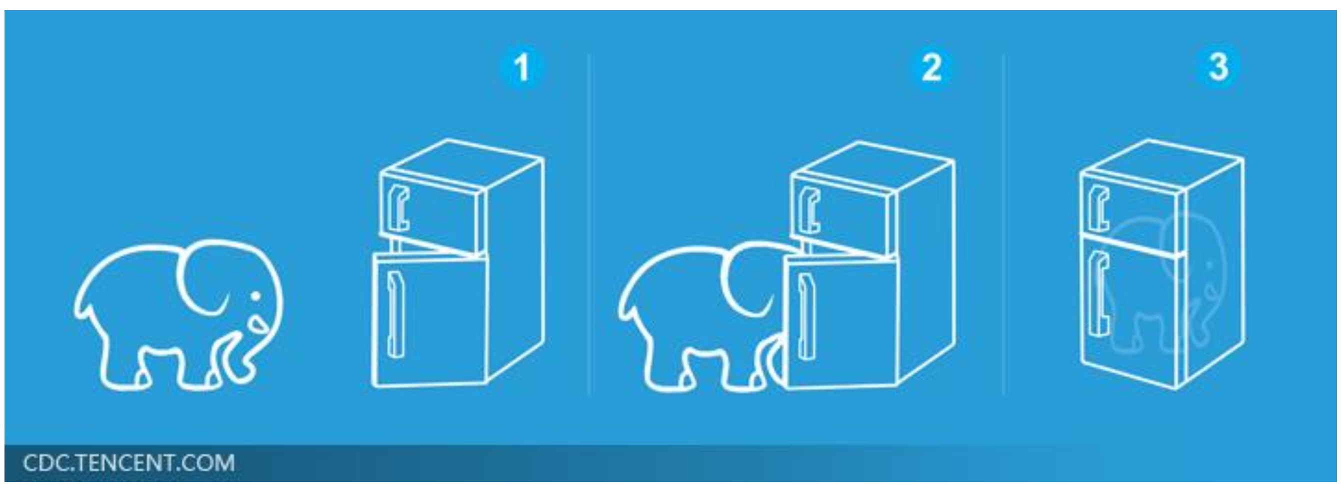
Learning Target



Three Steps for Deep Learning

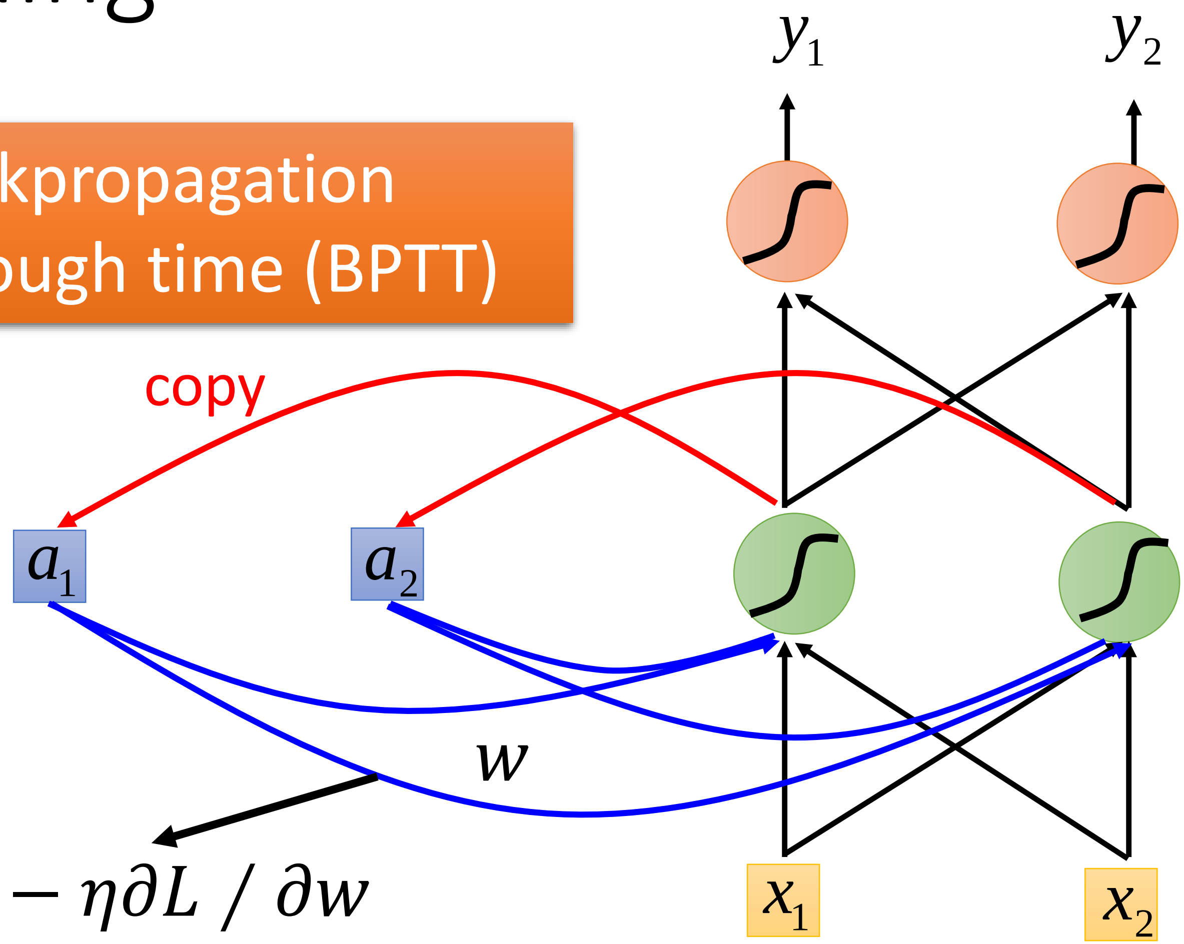
Step 1: define a set of function → Step 2: goodness of function → Step 3: pick the best function

Deep Learning is so simple ……



Learning

Backpropagation through time (BPTT)



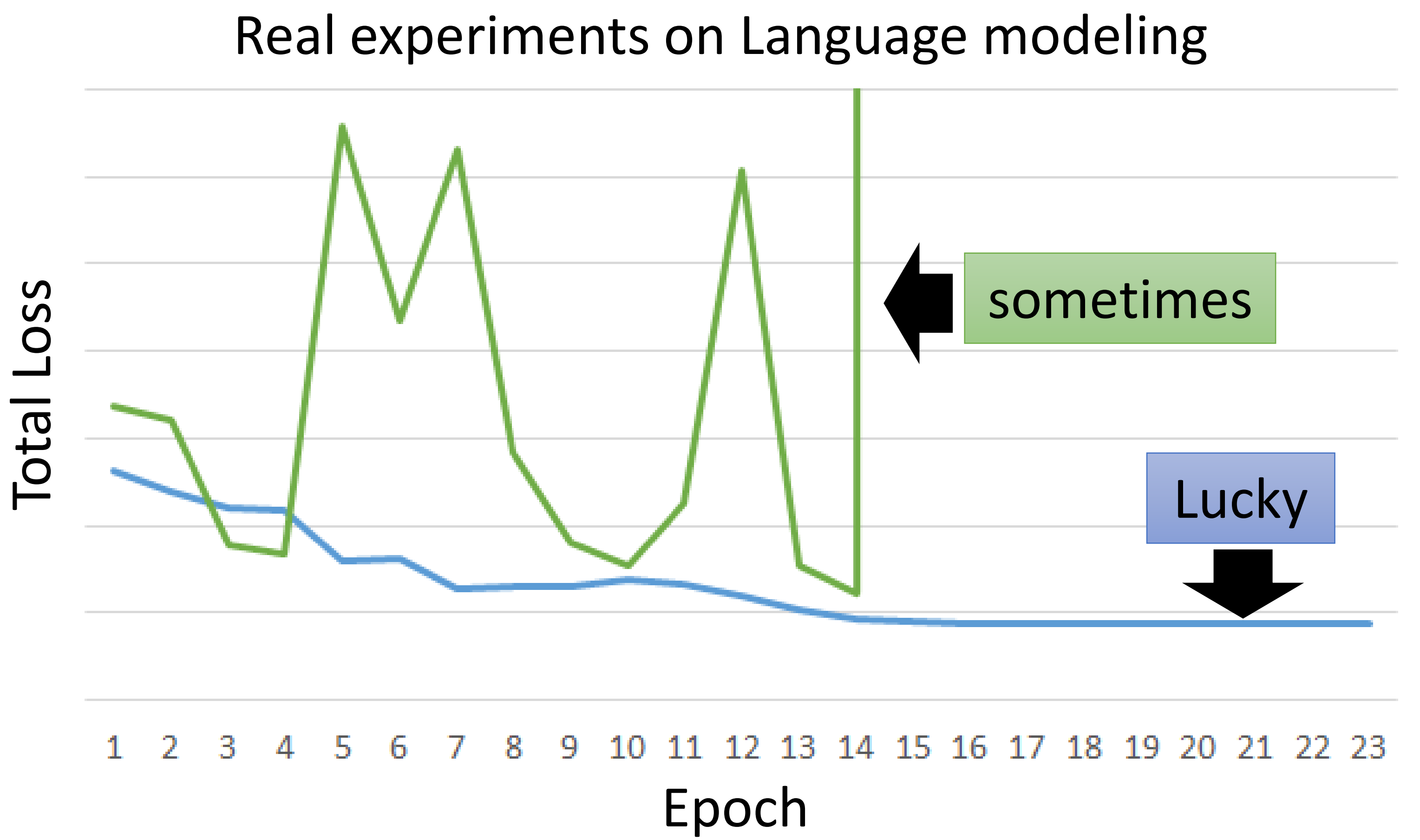
$$w \leftarrow w - \eta \partial L / \partial w$$

RNN Learning is very difficult in practice.

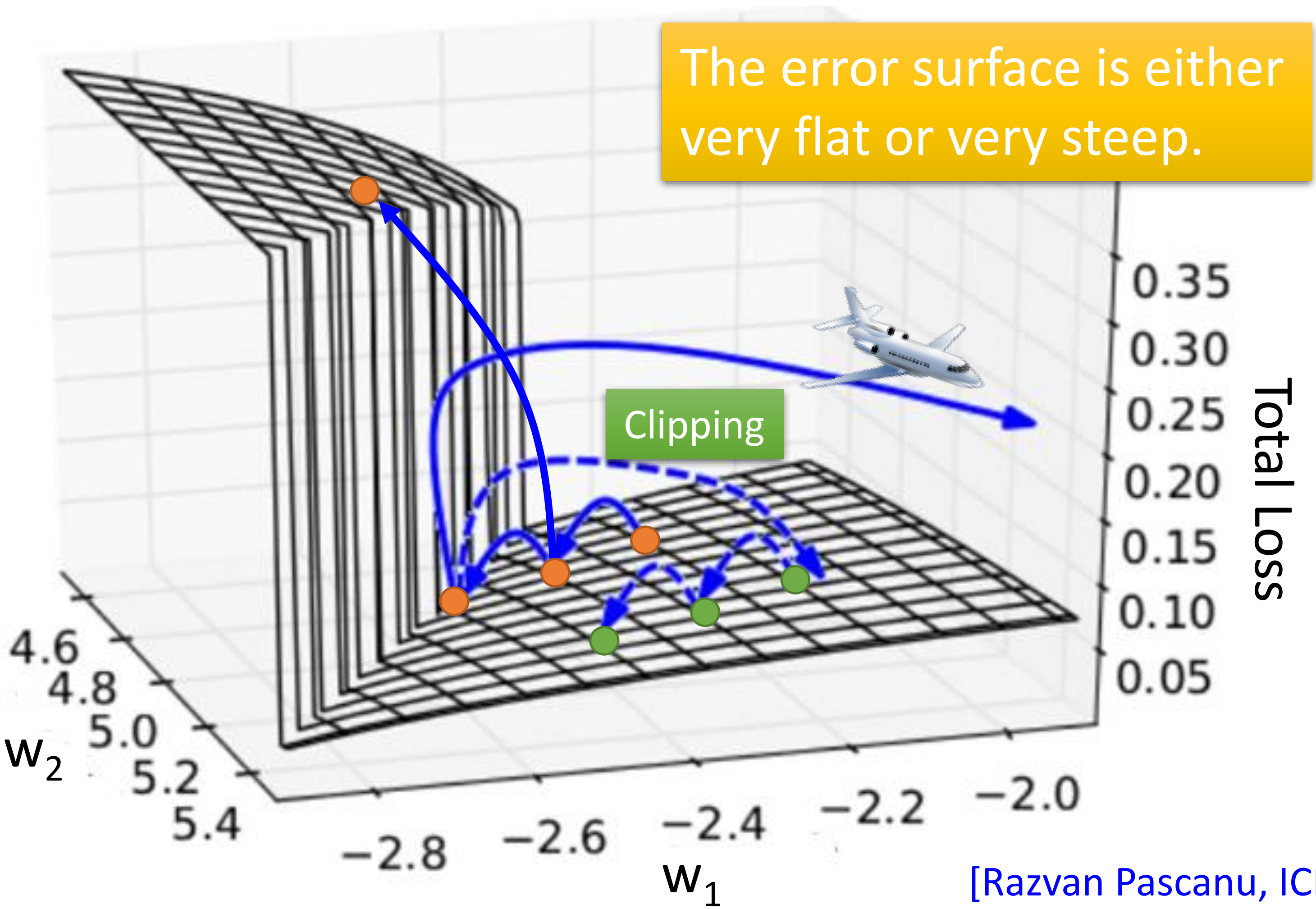
感謝 曾柏翔 同學
提供實驗結果

Unfortunately ……

- RNN-based network is not always easy to learn



The error surface is rough.



[Razvan Pascanu, ICML'13]

Why?

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

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

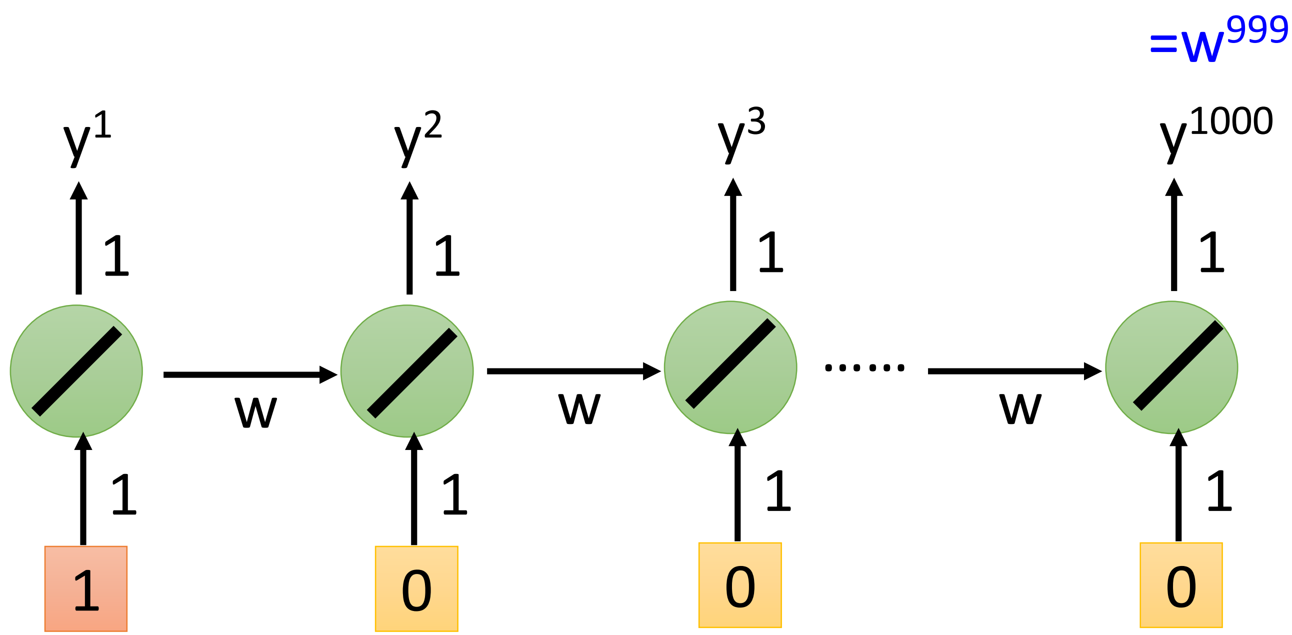
Large $\partial L / \partial w$ → Small Learning rate?

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

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

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

Toy Example



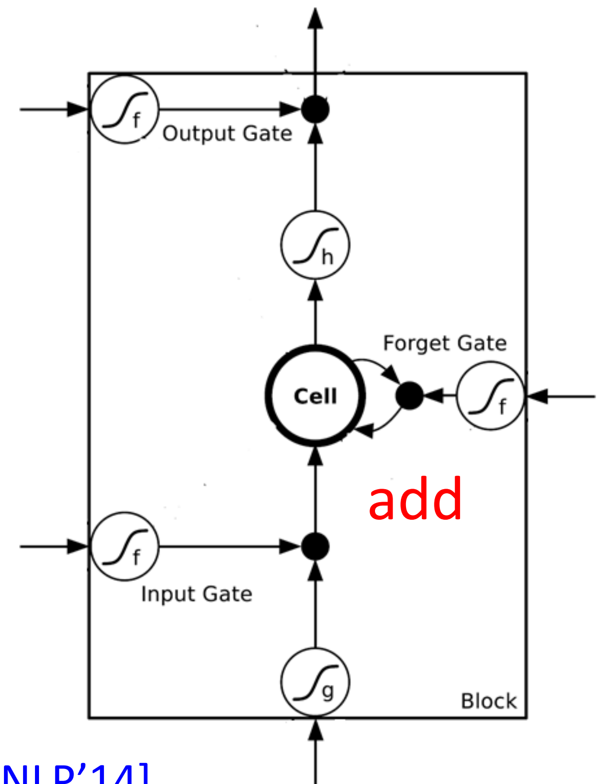
Helpful Techniques

- 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.)

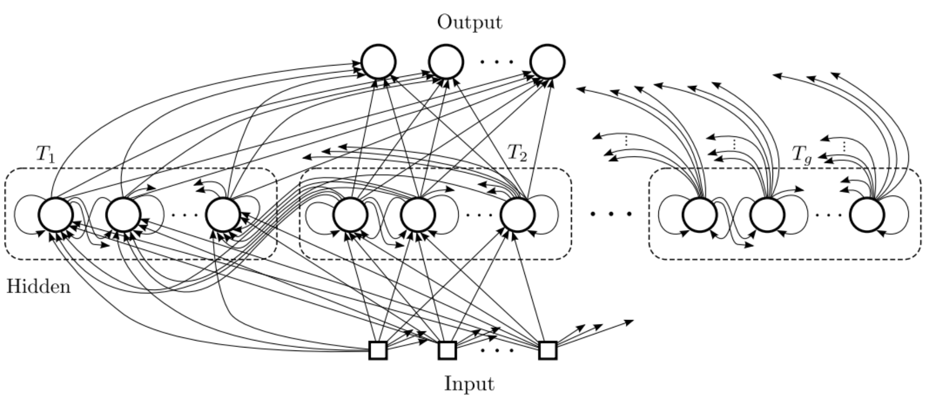


Gated Recurrent Unit (GRU): simpler than LSTM

[Cho, EMNLP'14]

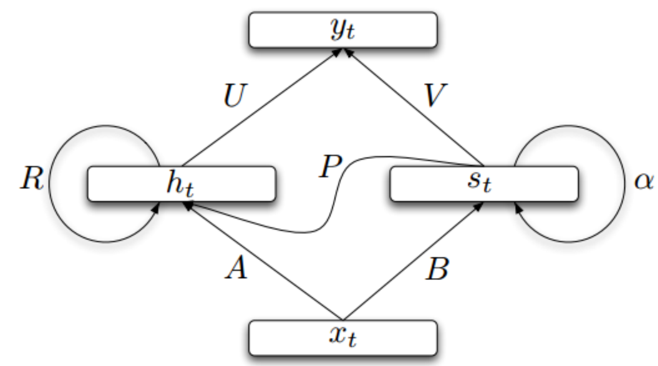
Helpful Techniques

Clockwise RNN



[Jan Koutnik, JMLR'14]

Structurally Constrained Recurrent Network (SCRN)

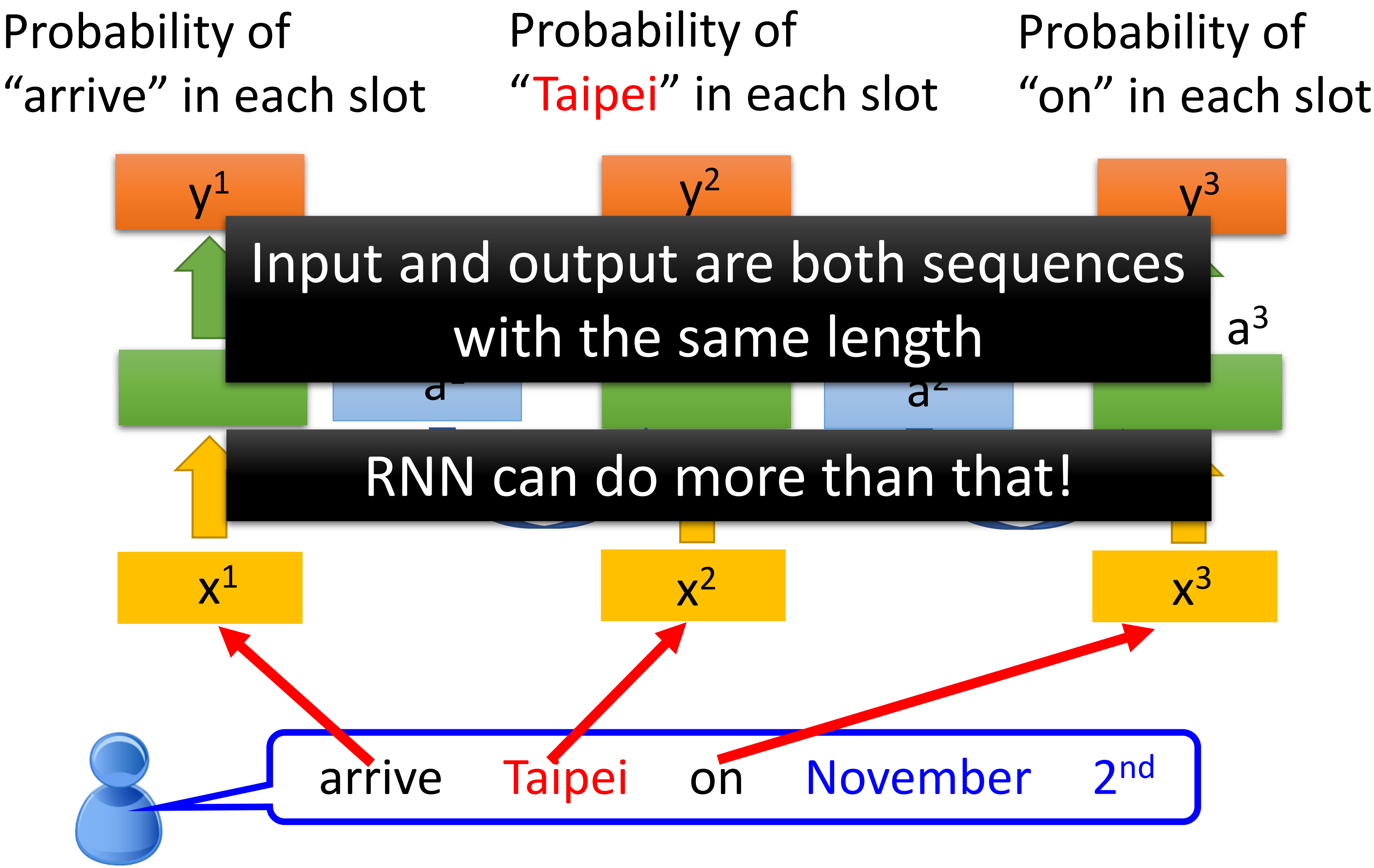


[Tomas Mikolov, ICLR'15]

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 ……



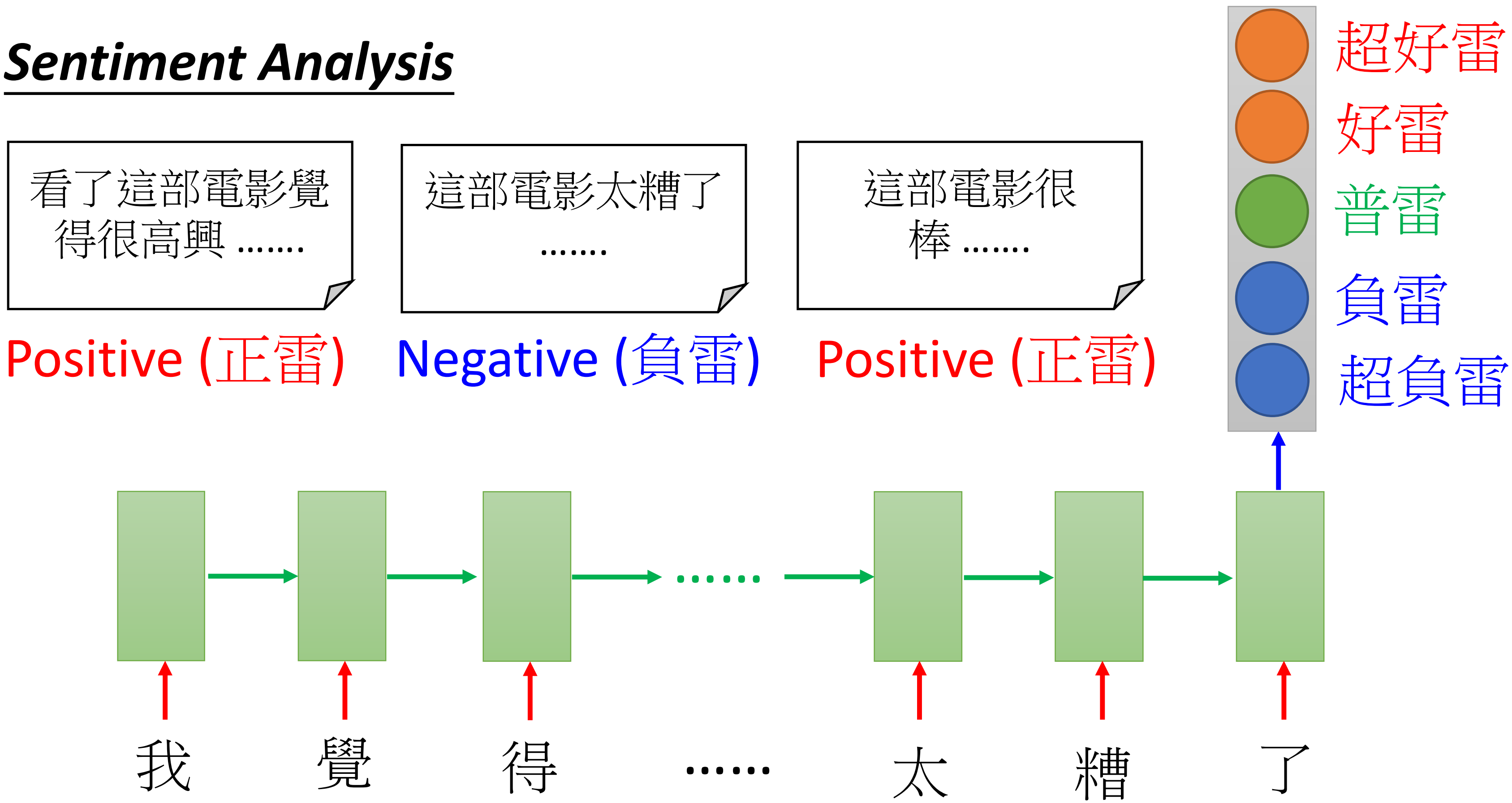
Input and output are both sequences with the same length

RNN can do more than that!

Many to one

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

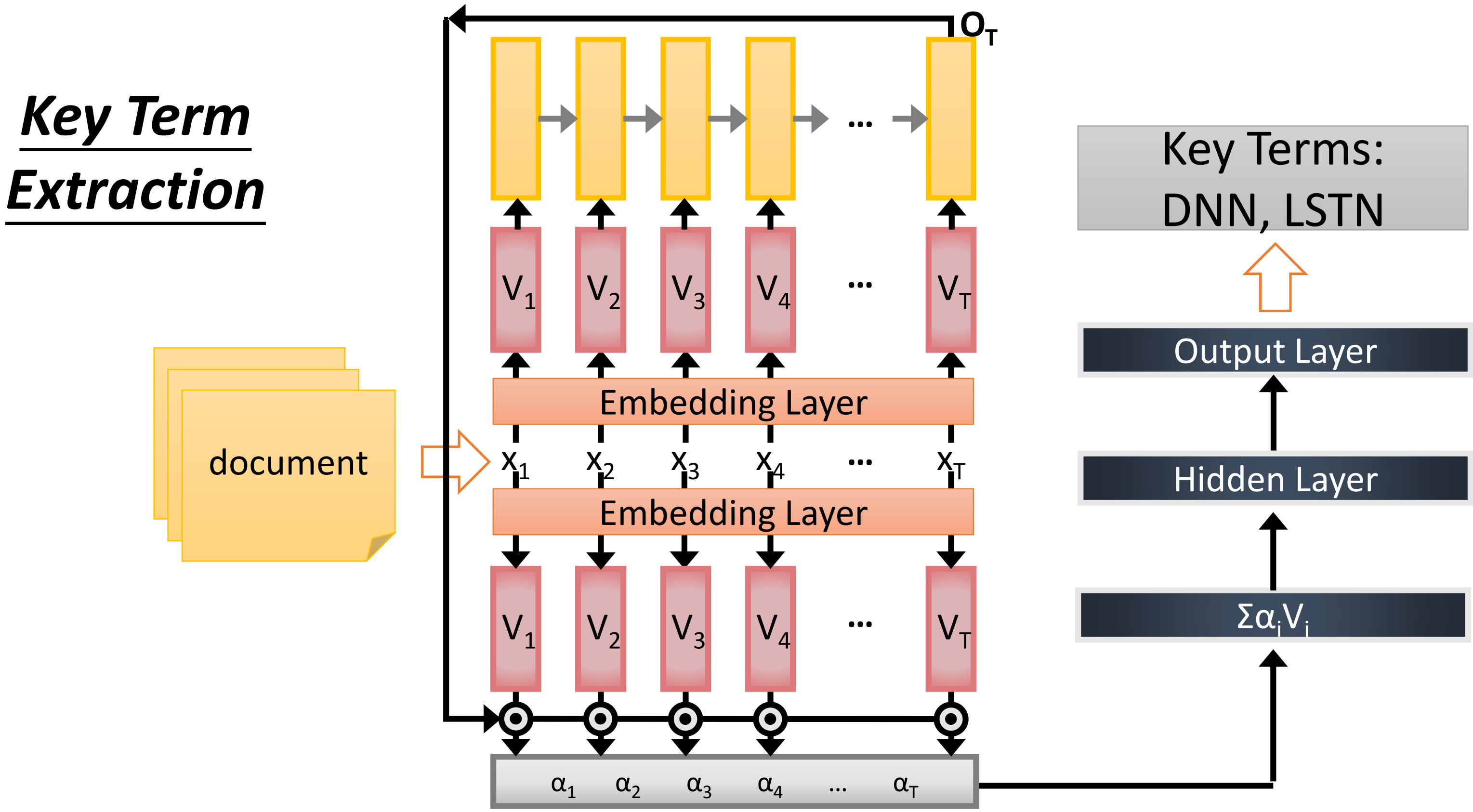
Sentiment Analysis



Many to one

[Shen & Lee, Interspeech 16]

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

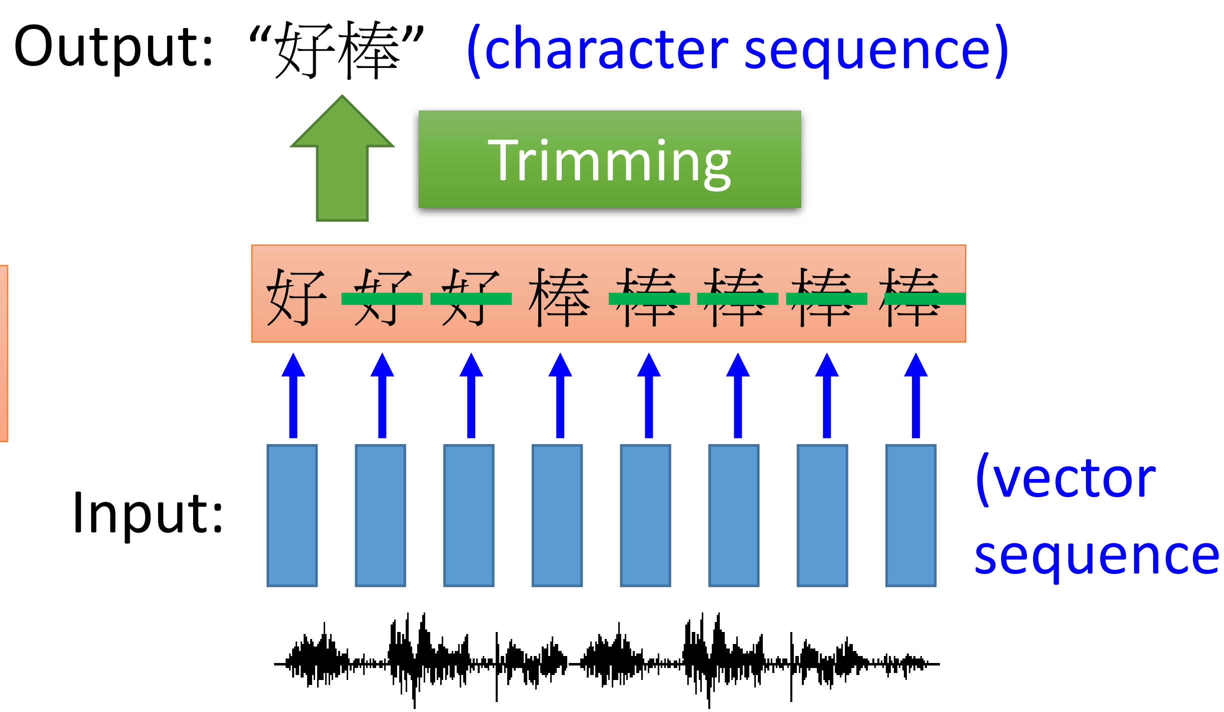


Many to Many (Output is shorter)

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

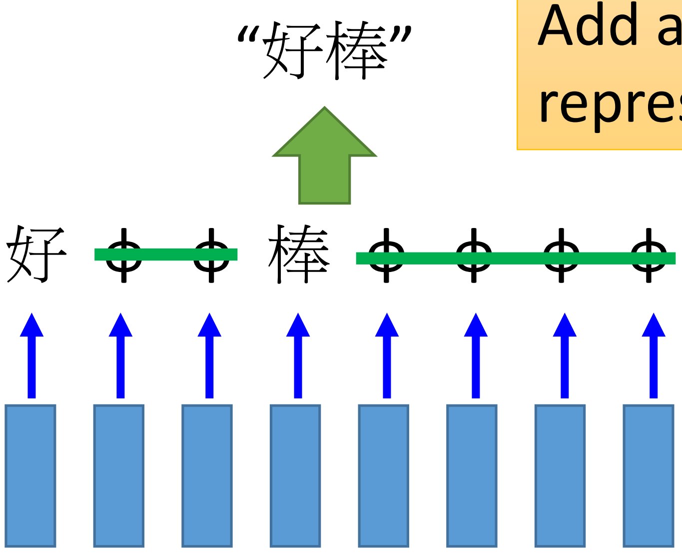
Problem?

Why can't it be "好棒棒"

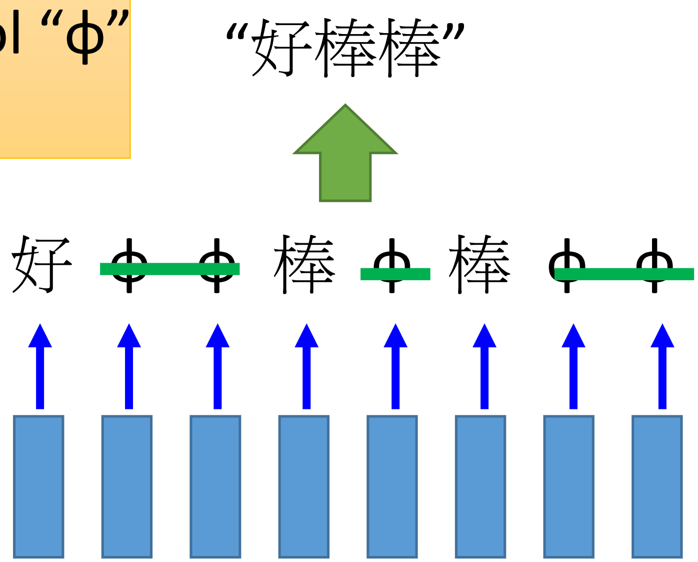


Many to Many (Output is shorter)

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

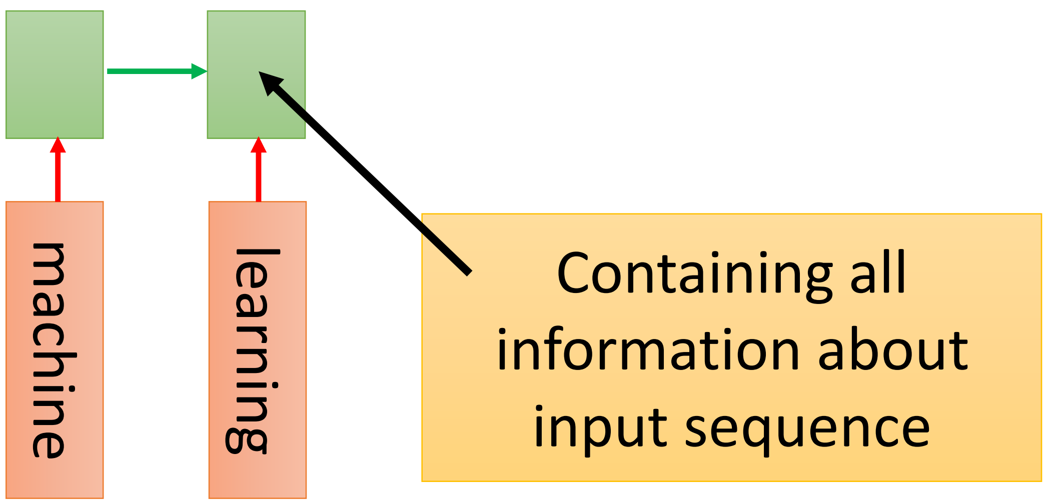


Add an extra symbol “φ” representing “null”



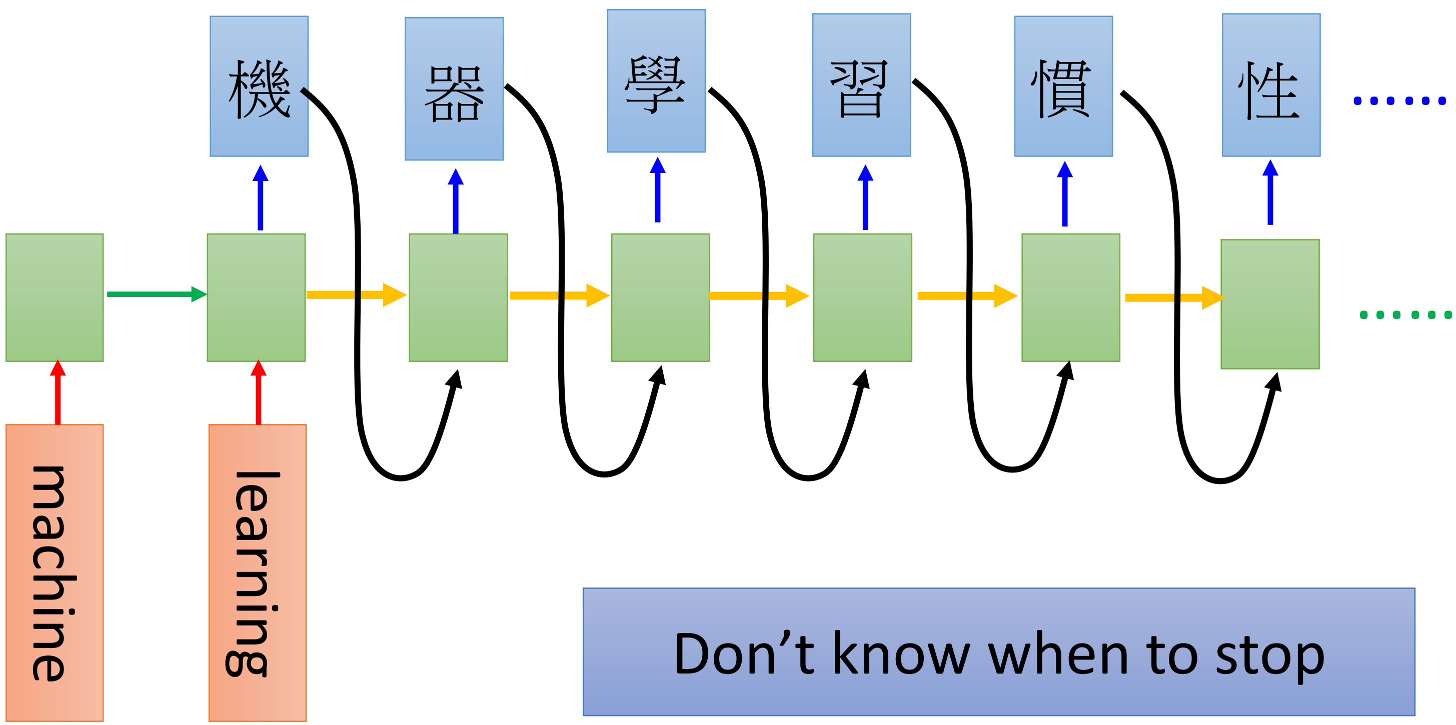
Many to Many (No Limitation)

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

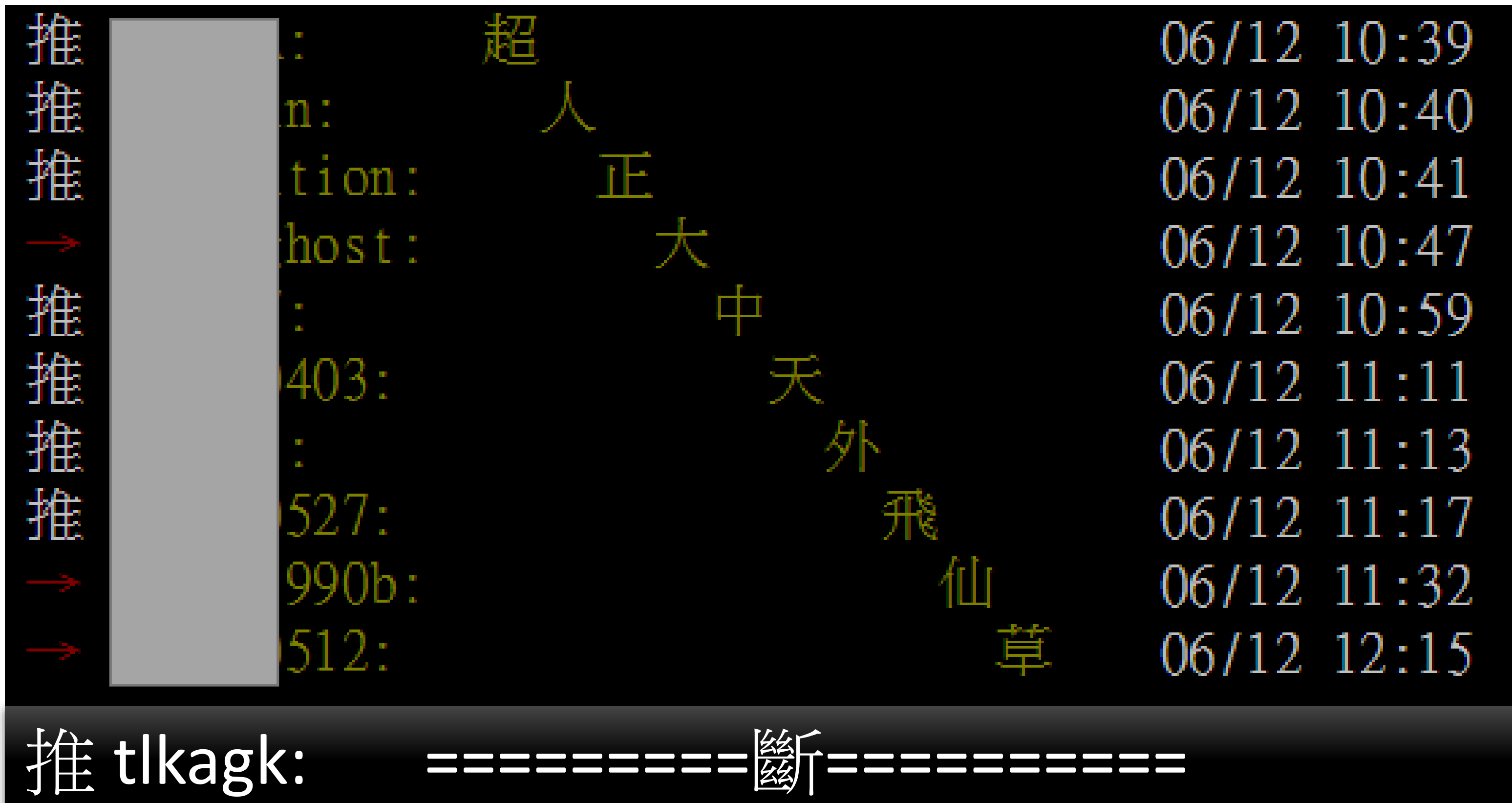


Many to Many (No Limitation)

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



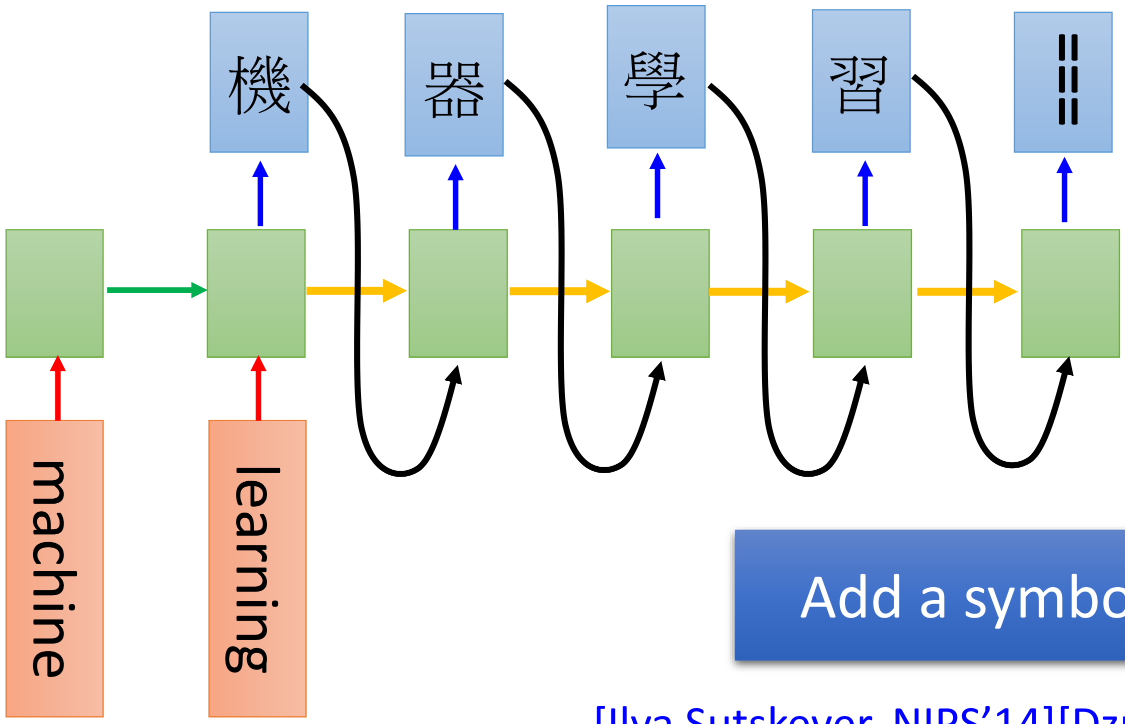
Many to Many (No Limitation)



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

Many to Many (No Limitation)

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



Add a symbol "===" (斷)

[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]

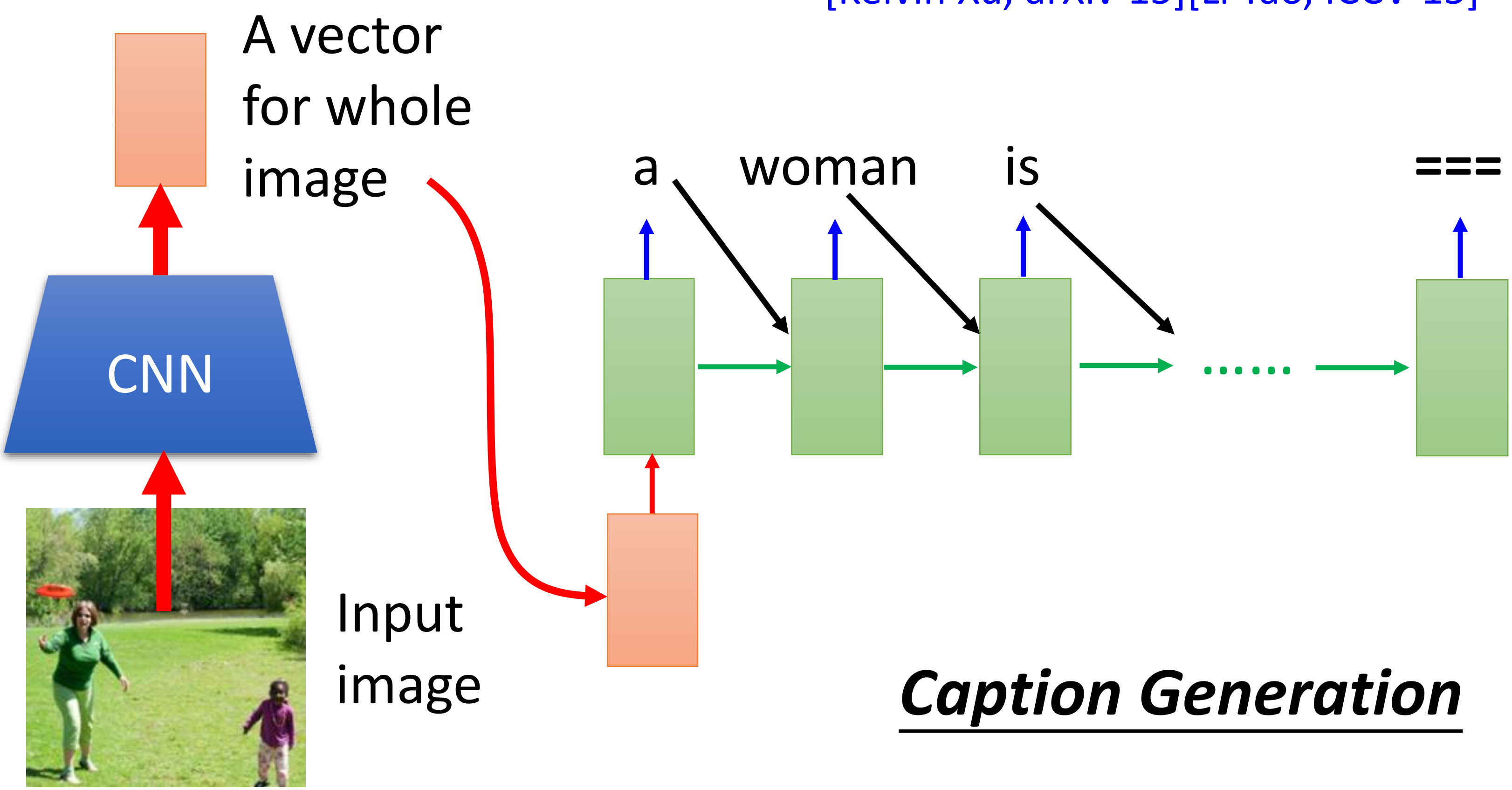


Image Caption Generation

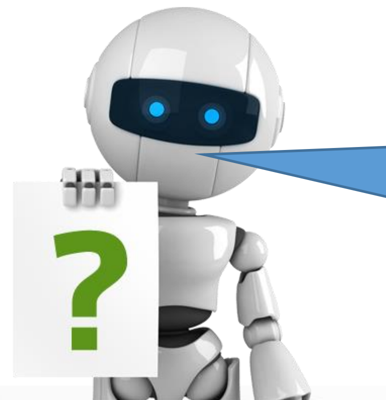
- Can machine describe what it see from image?
- Demo:台大電機系 大四 蘇子睿、林奕辰、徐翊祥、陳奕安

MTK 產學大聯盟

http://news.ltn.com.tw/photo/politics/breakingnews/975542_1

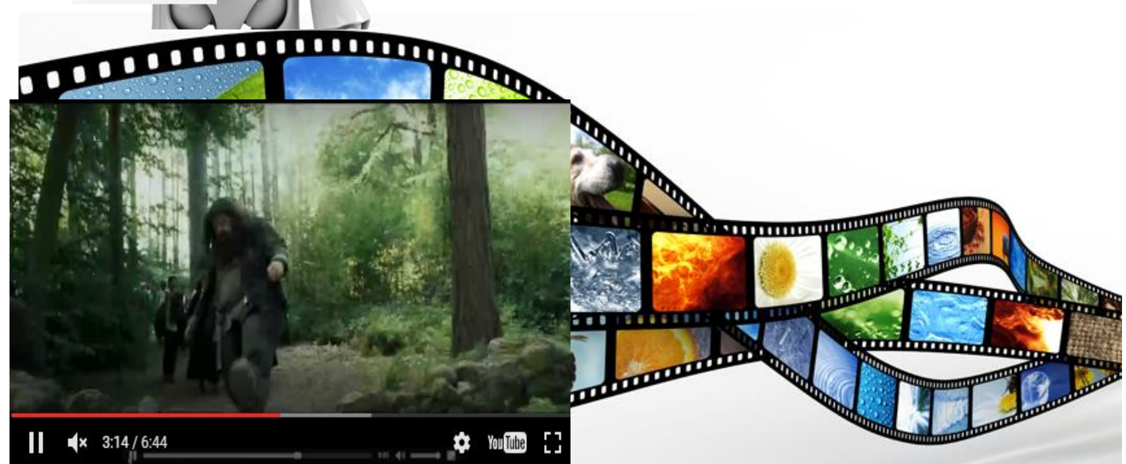
Video Caption Generation

Video



A girl is running.

A group of people is knocked by a tree.

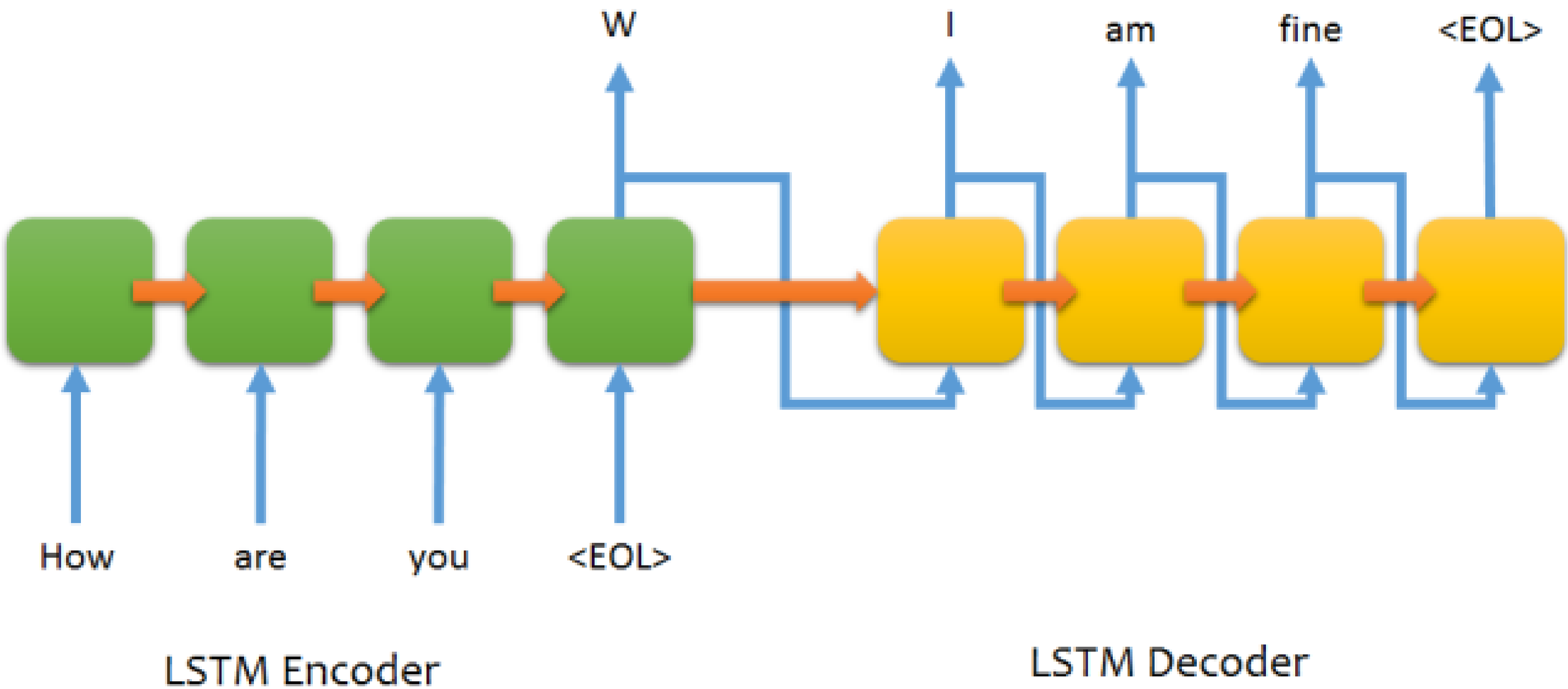


A group of people is walking in the forest.

Video Caption Generation

- Can machine describe what it see from video?
- Demo: 台大語音處理實驗室 曾柏翔、吳柏瑜、盧宏宗

Chat-bot



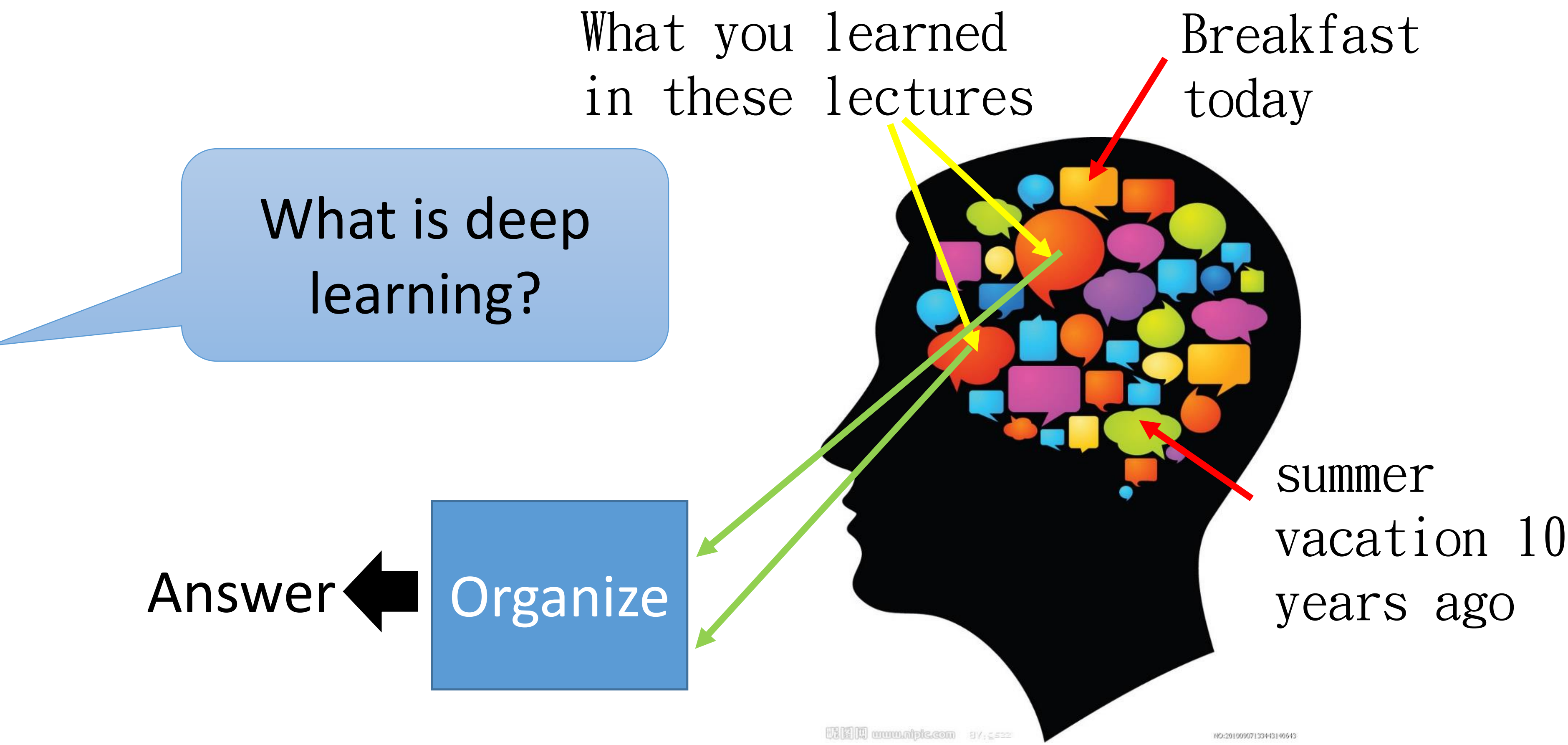
電視影集 (~40,000 sentences)、美國總統大選辯論

Demo

- Develop Team
  - Interface design: Prof. Lin-Lin Chen & Arron Lu
  - Web programming: Shi-Yun Huang
  - Data collection: Chao-Chuang Shih
  - System implementation: Kevin Wu, Derek Chuang, & Zhi-Wei Lee
  - System design: Richard Tsai & Hung-Yi Lee

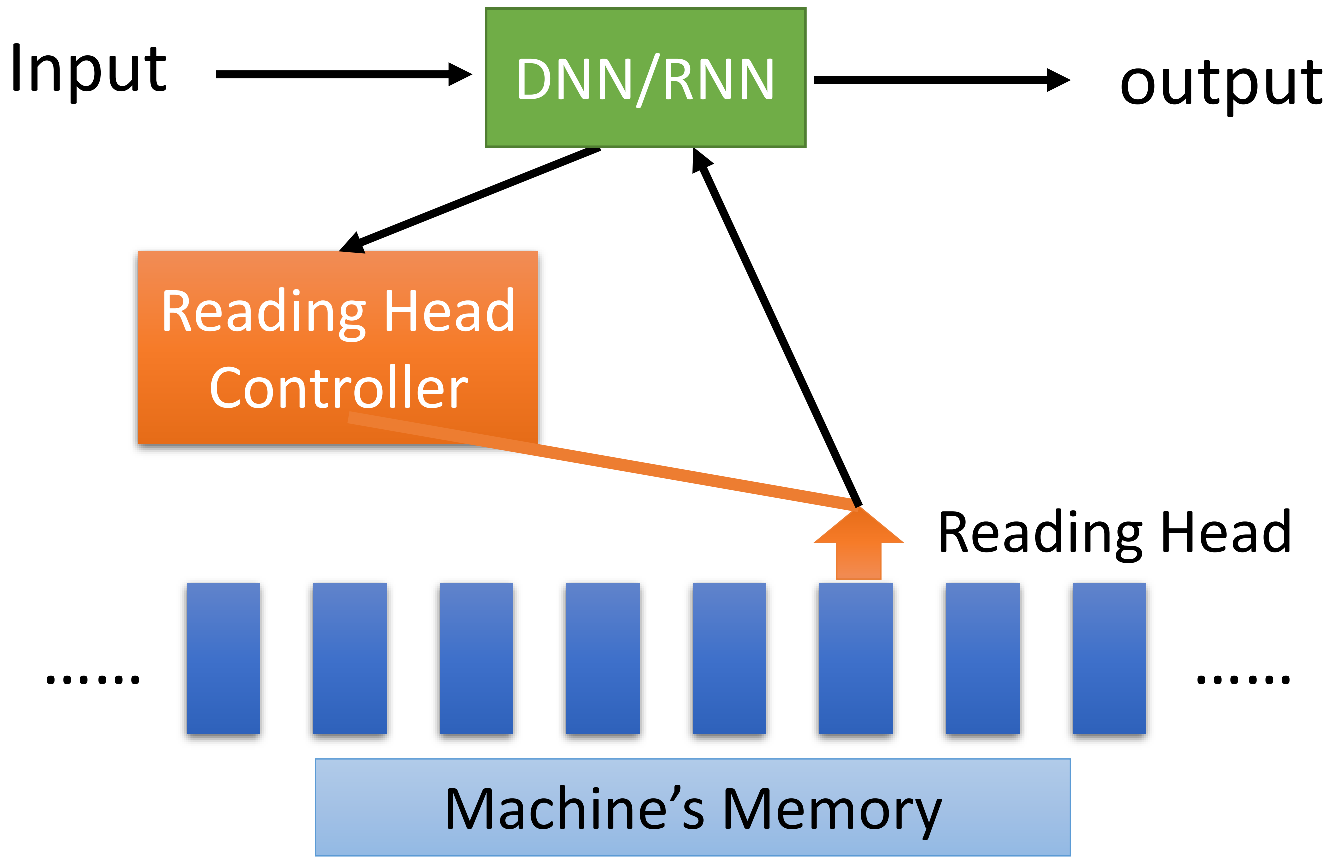
NTU IOX center

Attention-based Model



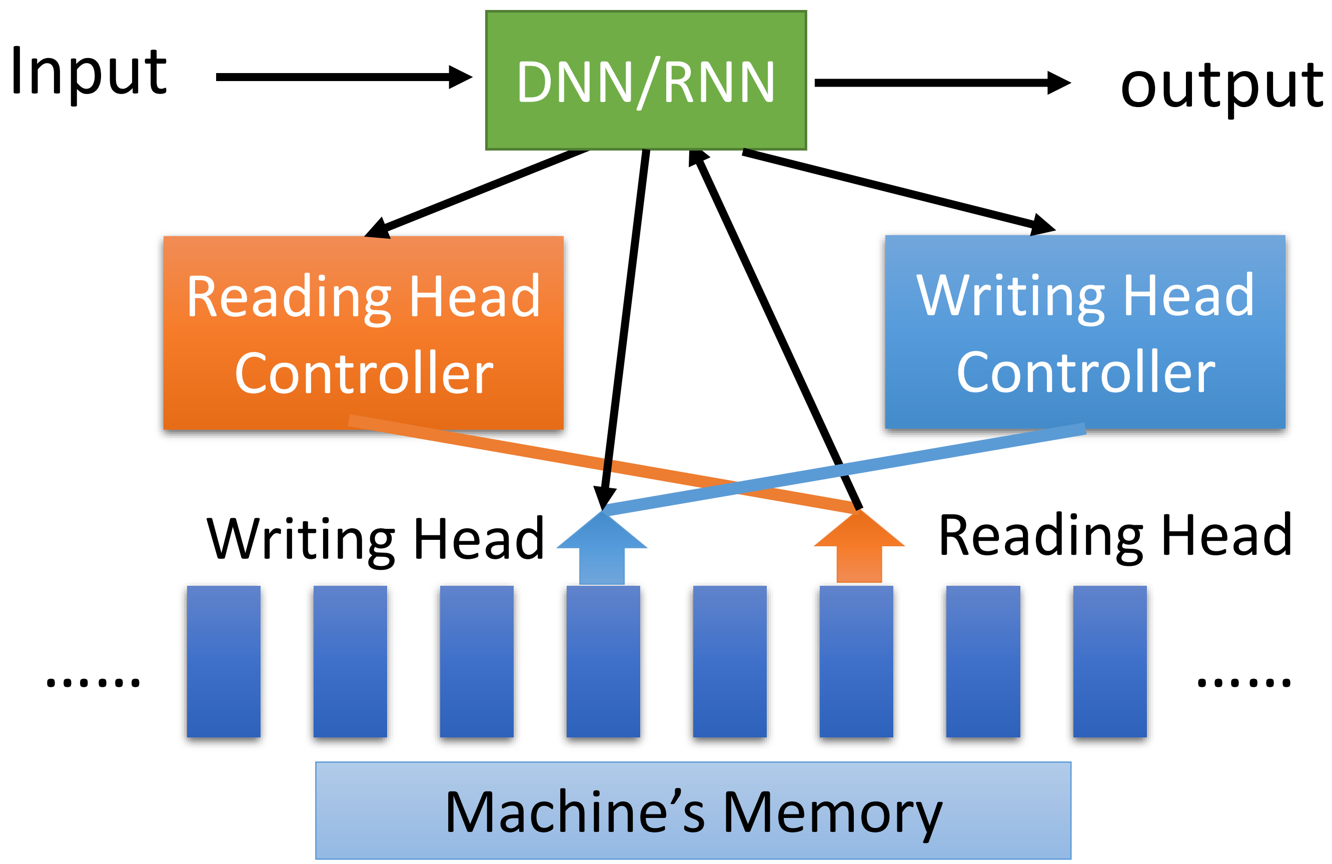
http://henrylo1605.blogspot.tw/2015/05/blog-post_56.html

Attention-based Model



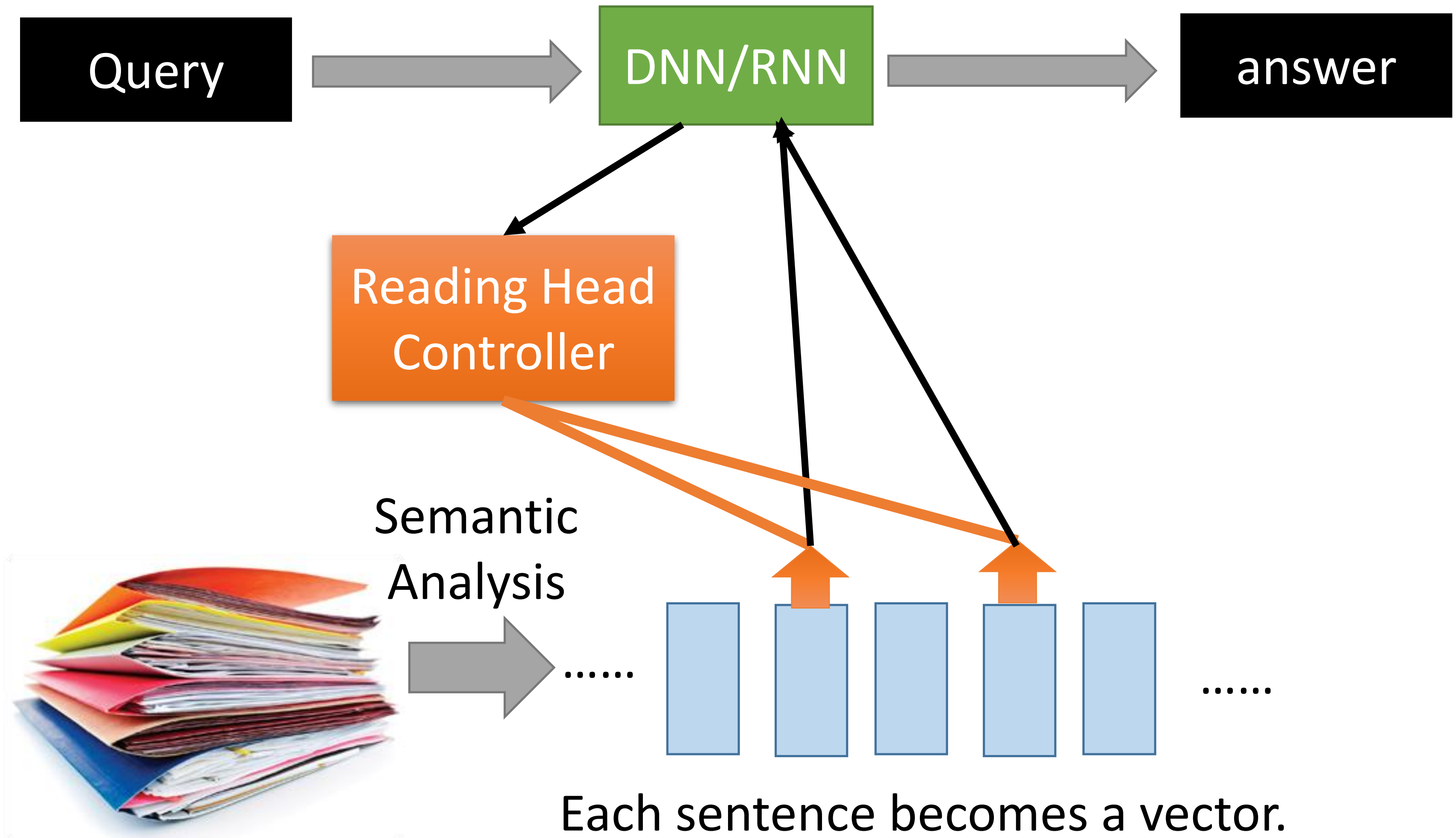
Ref:
http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2/Lecture/Attain%20(v3).ecm.mp4/index.html

Attention-based Model v2



Neural Turing Machine

Reading Comprehension



Reading Comprehension

- End-To-End Memory Networks. S. Sukhbaatar, A. Szlam, J. Weston, R. Fergus. NIPS, 2015.

The position of reading head:

Story (16: basic induction)	Support	Hop 1	Hop 2	Hop 3
Brian is a frog.	yes	0.00	0.98	0.00
Lily is gray.		0.07	0.00	0.00
Brian is yellow.	yes	0.07	0.00	1.00
Julius is green.		0.06	0.00	0.00
Greg is a frog.	yes	0.76	0.02	0.00
What color is Greg? Answer: yellow	**Prediction: yellow**			

Keras has example:
https://github.com/fchollet/keras/blob/master/examples/babi_memnn.py

Visual Question Answering

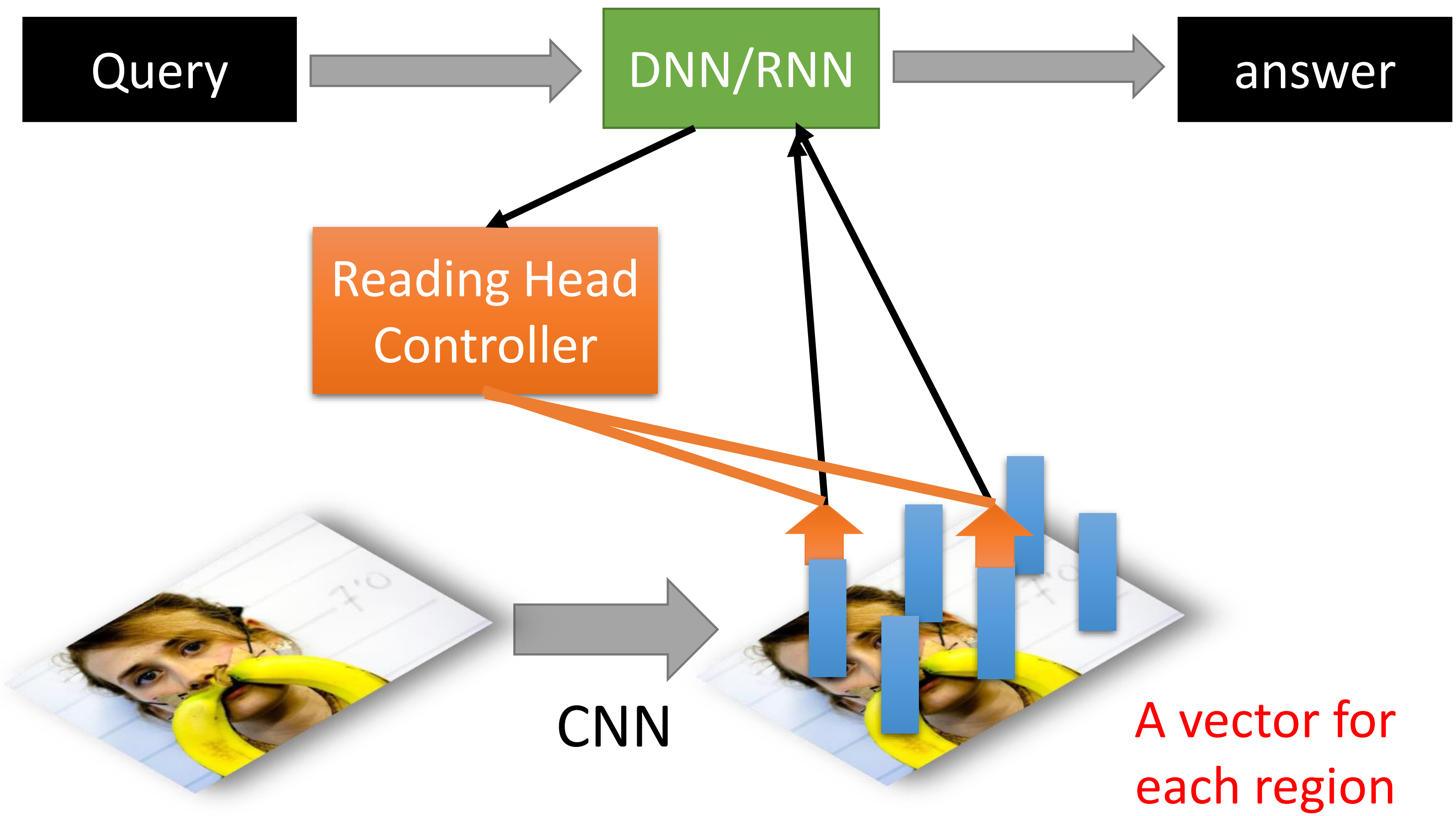
What is the mustache made of?

AI System

bananas

source: http://visualqa.org/

Visual Question Answering



Speech Question Answering

- **TOEFL Listening Comprehension Test by Machine**
- Example:

Audio Story:  (The original story is 5 min long.)

Question: “ What is a possible origin of Venus’ clouds? ”

Choices:

(A) gases released as a result of volcanic activity

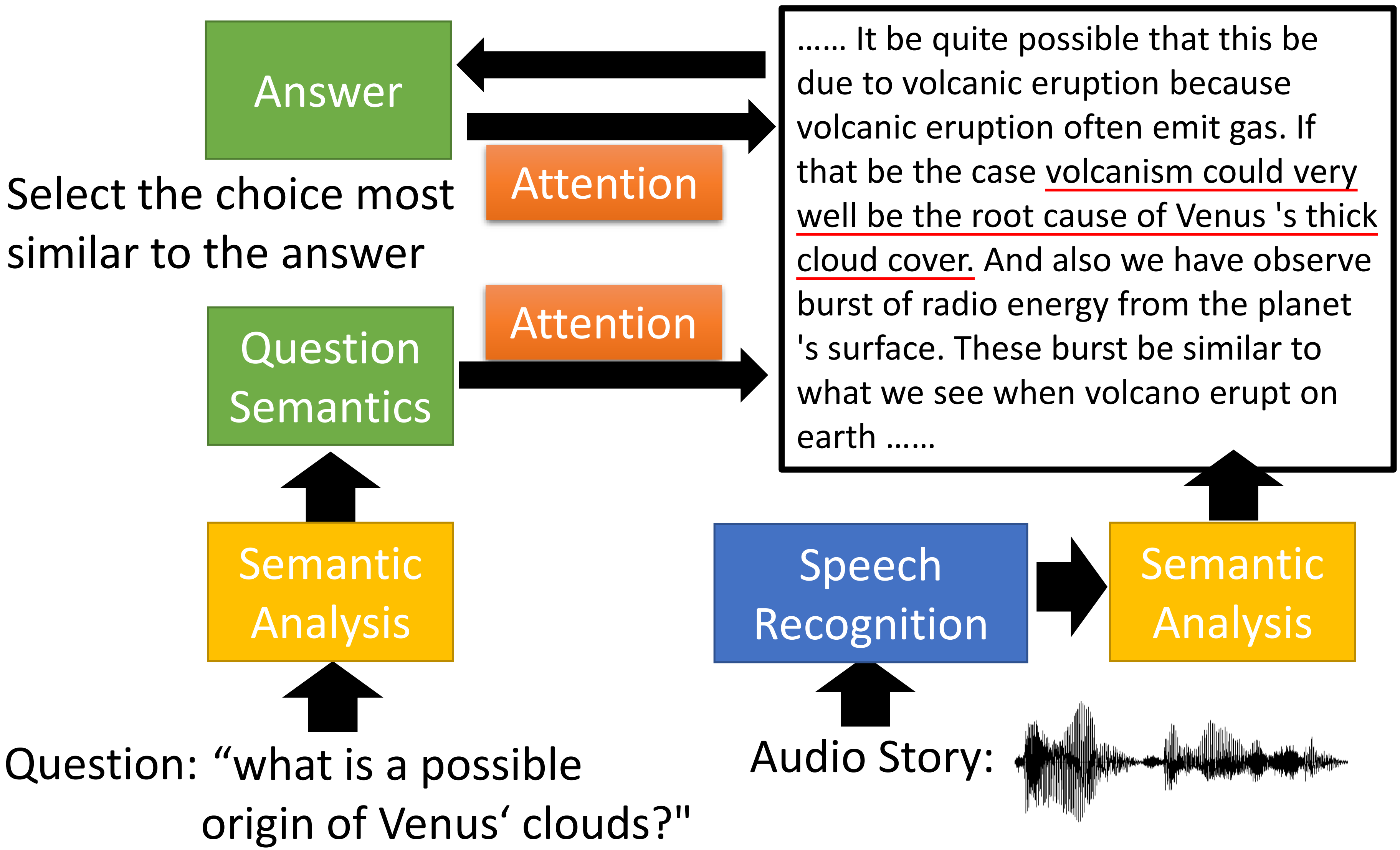
(B) chemical reactions caused by high surface temperatures

(C) bursts of radio energy from the plane's surface

(D) strong winds that blow dust into the atmosphere

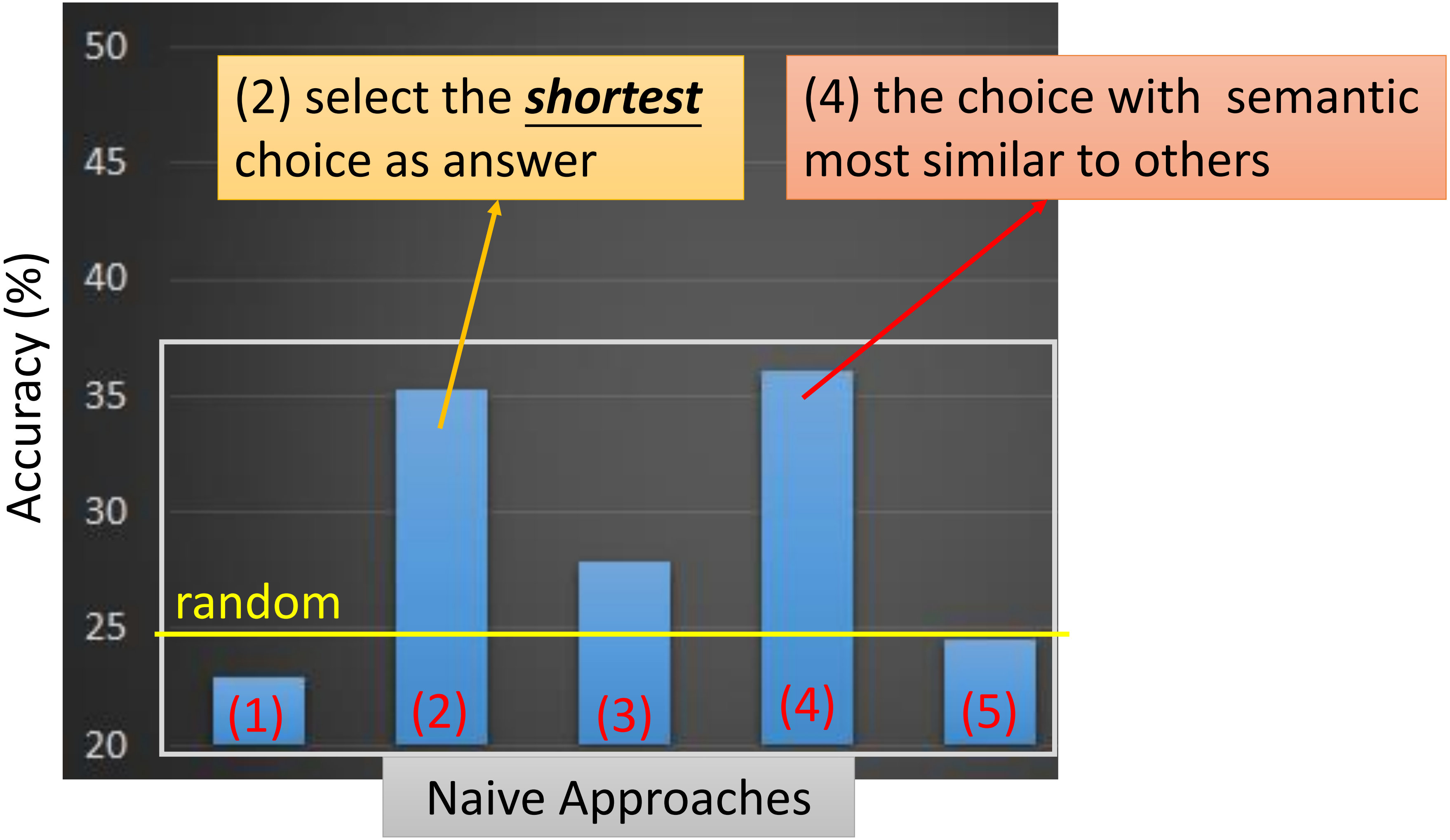
Model Architecture

Everything is learned from training examples

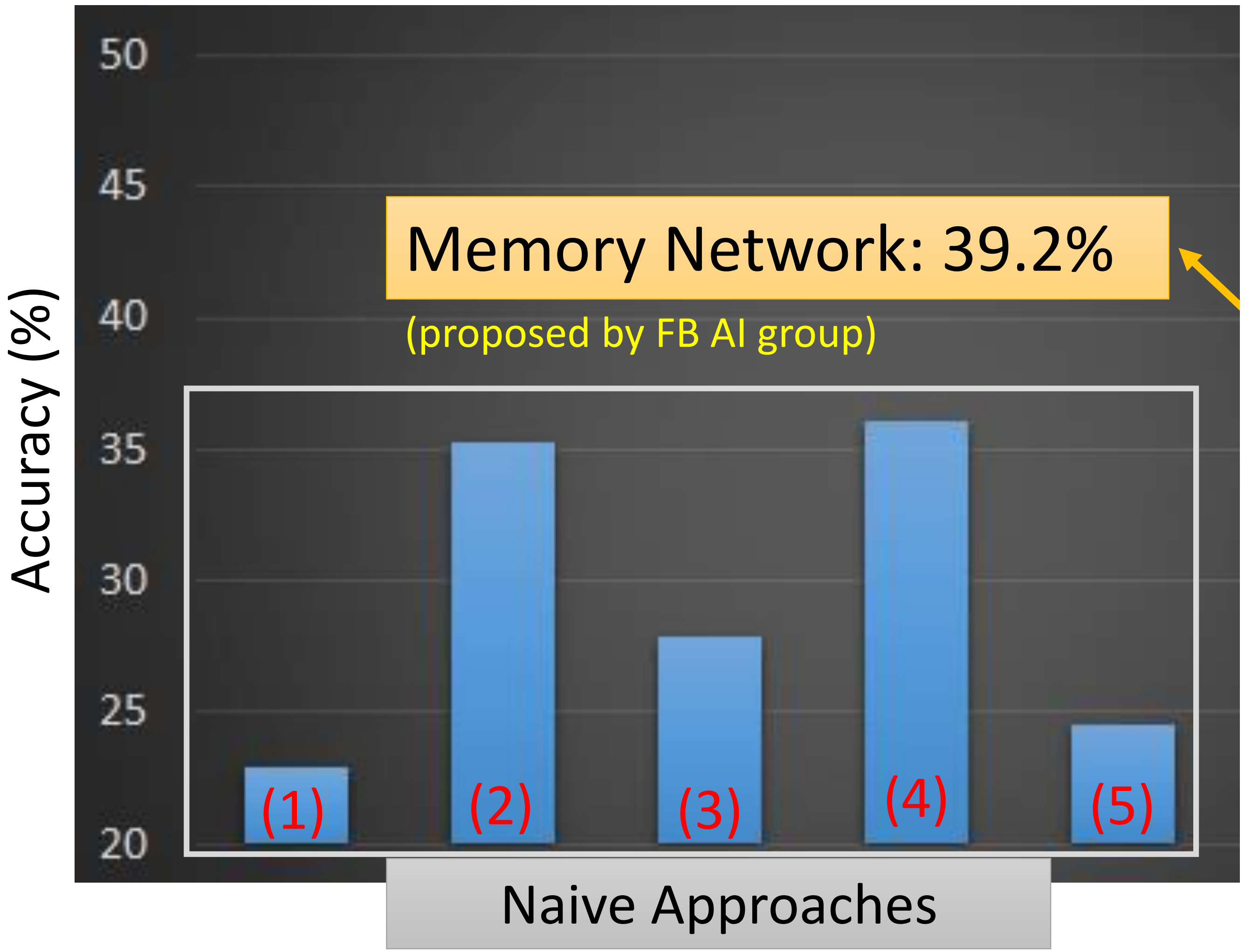


Simple Baselines

Experimental setup:
717 for training,
124 for validation, 122 for testing

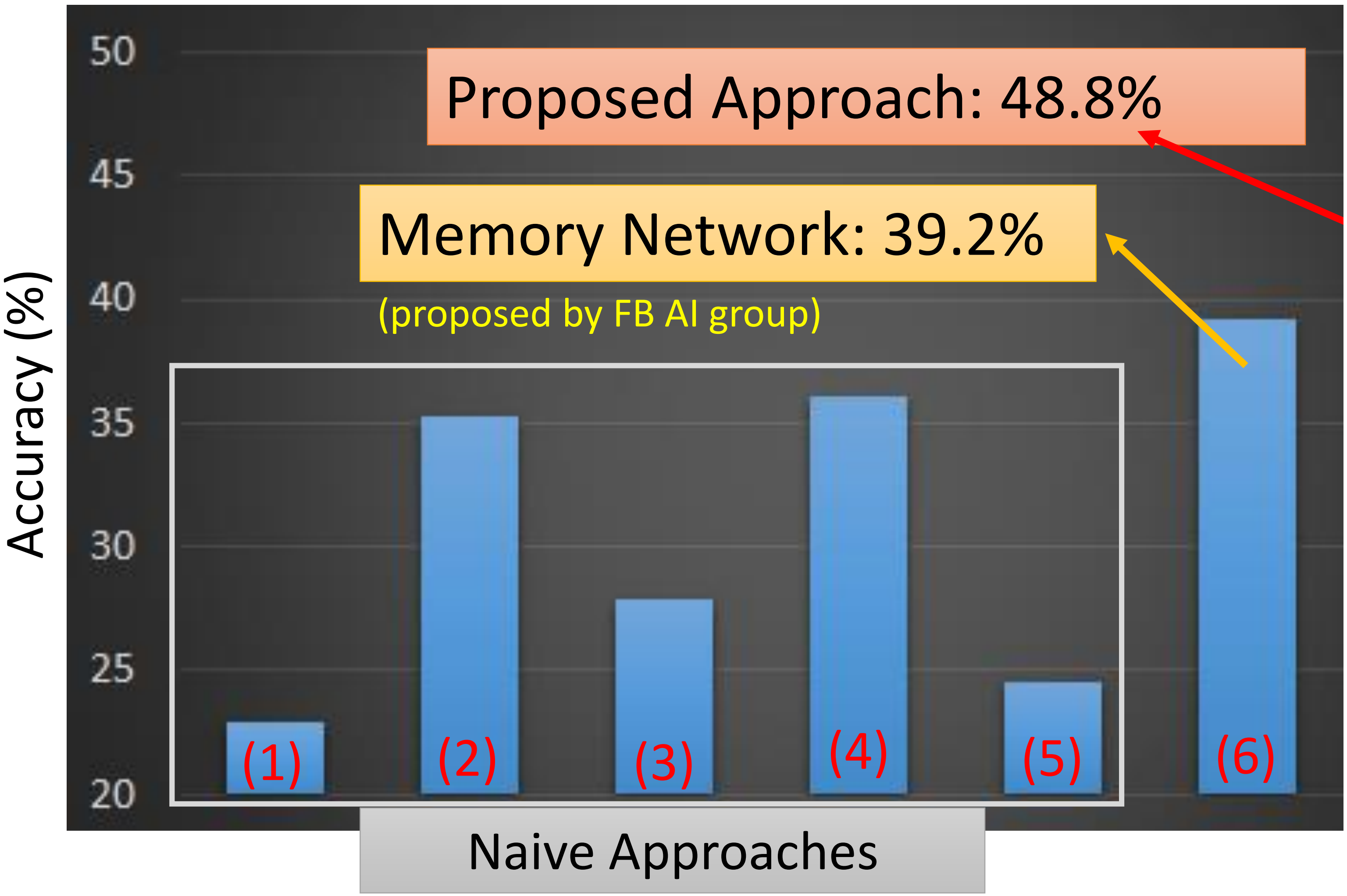


Memory Network



Proposed Approach

[Tseng & Lee, Interspeech 16]
[Fang & Hsu & Lee, SLT 16]



Concluding Remarks

Convolutional Neural Network (CNN)

Recurrent Neural Network (RNN)

Lecture III: Beyond Supervised Learning

Outline

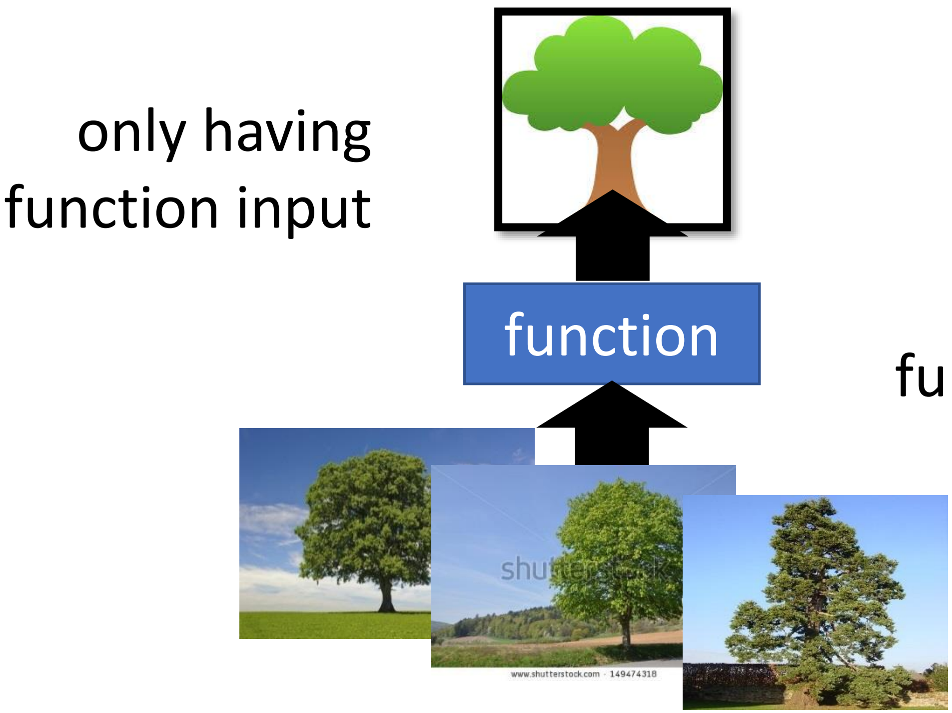
Unsupervised Learning

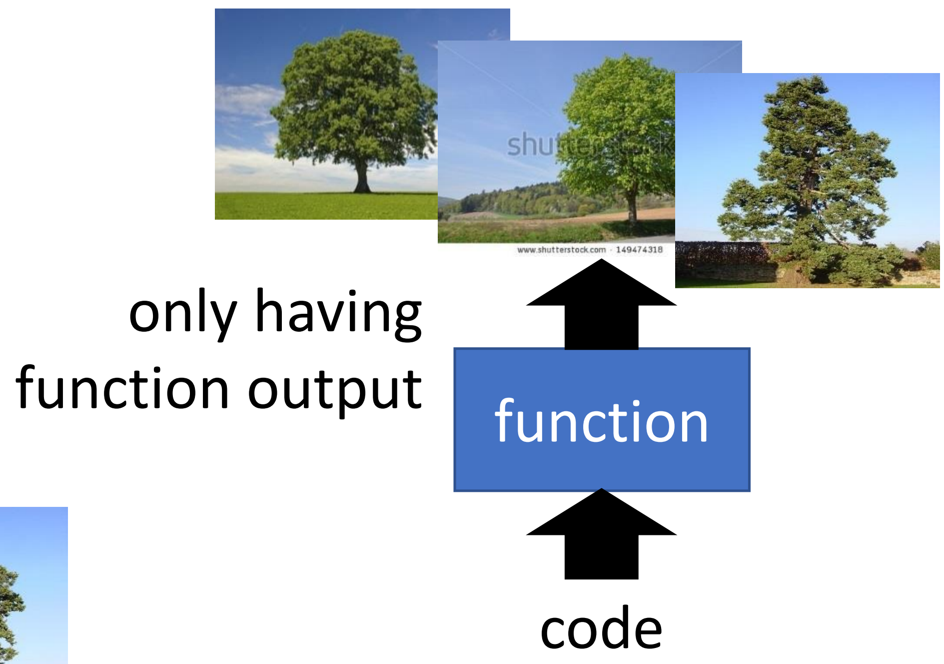
- 化繁為簡
  - Auto-encoder
  - Word Vector and Audio Word Vector
- 無中生有

Reinforcement Learning

Unsupervised Learning

- 化繁為簡
- 無中生有





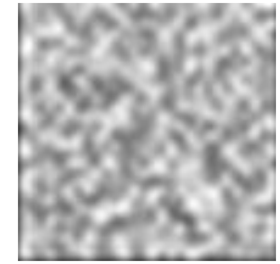
Outline

Unsupervised Learning

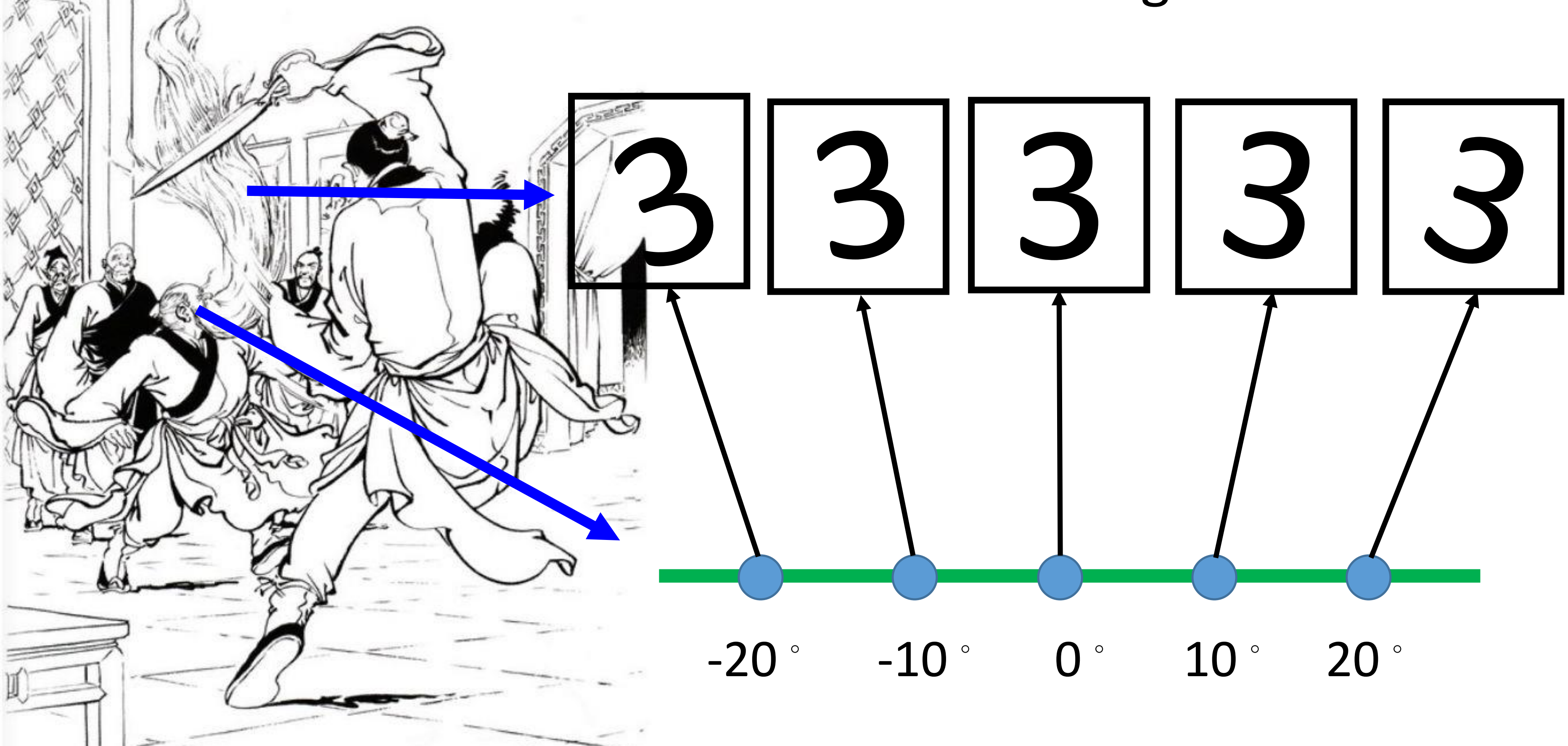
- 化繁為簡
  - Auto-encoder
  - Word Vector and Audio Word Vector
- 無中生有

Reinforcement Learning

Motivation



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



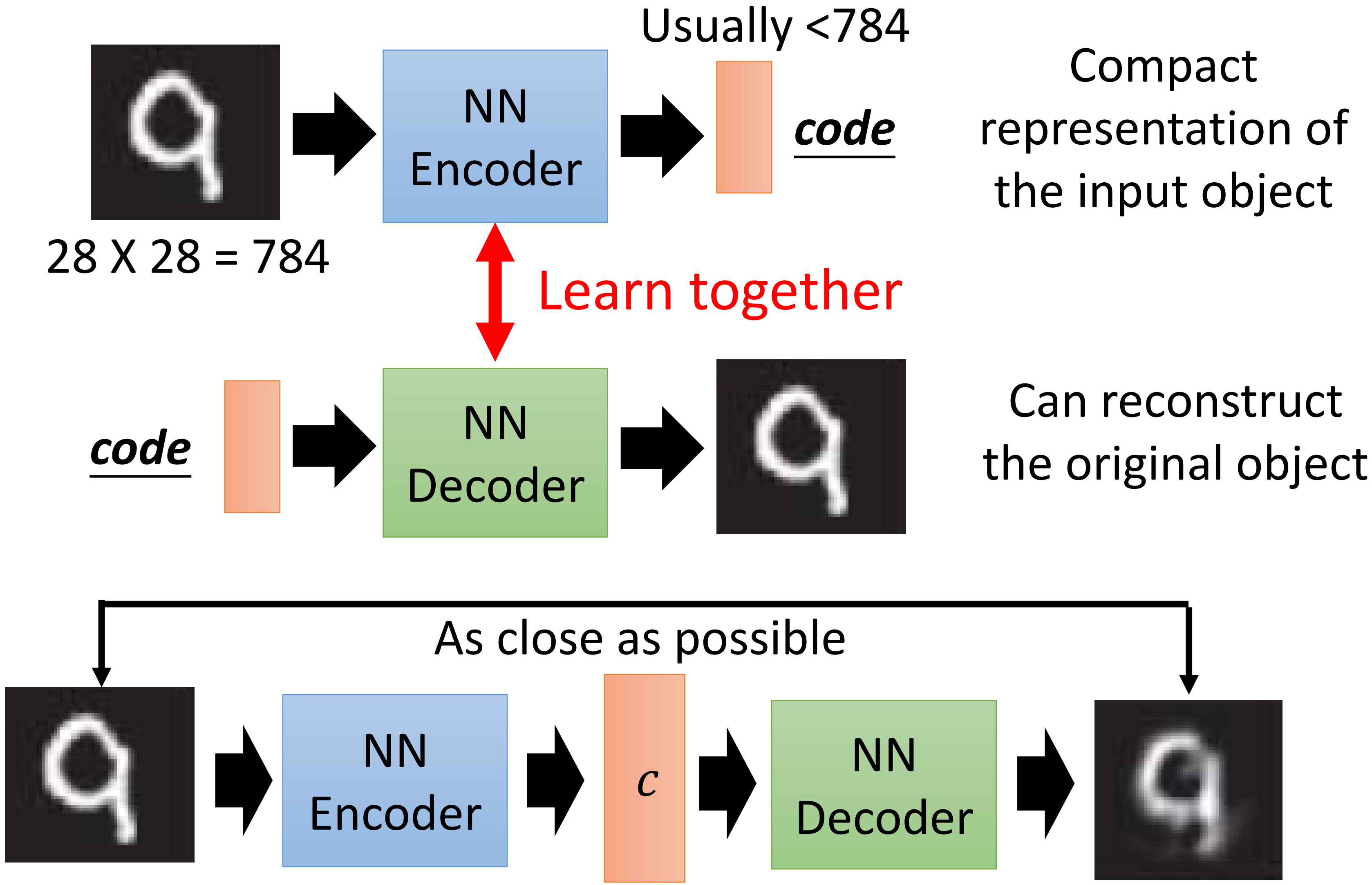
Outline

Unsupervised Learning

- 化繁為簡
  - $\boxed{\text{Auto-encoder}}$
  - Word Vector and Audio Word Vector
- 無中生有

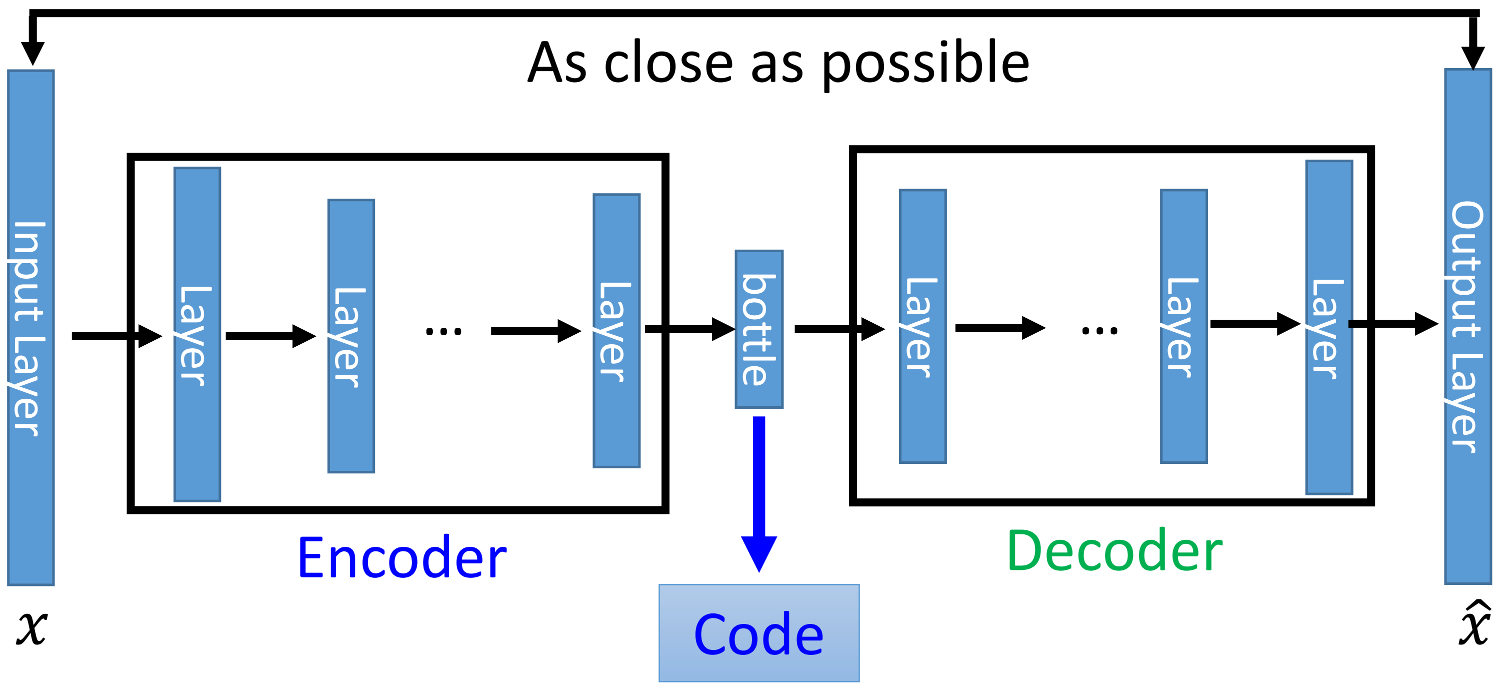
Reinforcement Learning

Auto-encoder



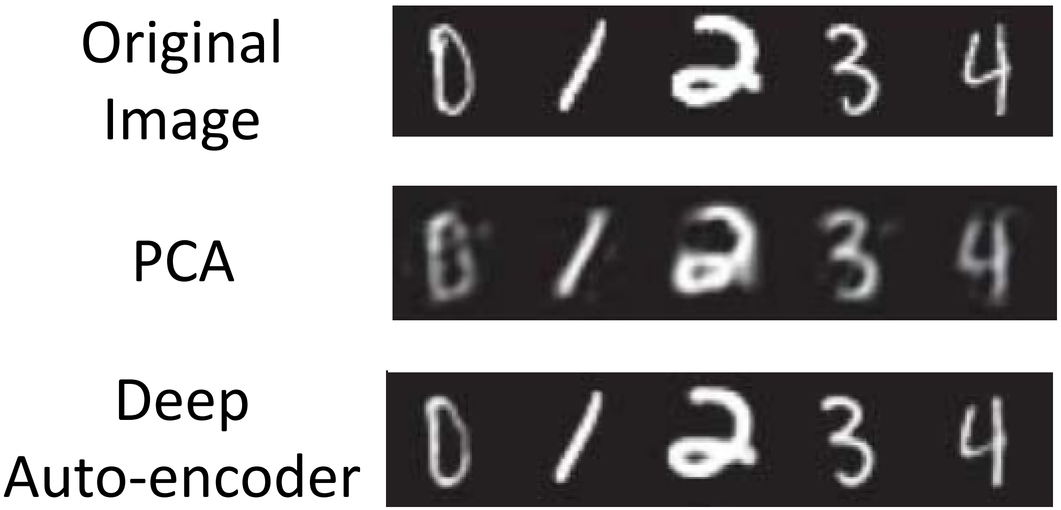
Deep Auto-encoder

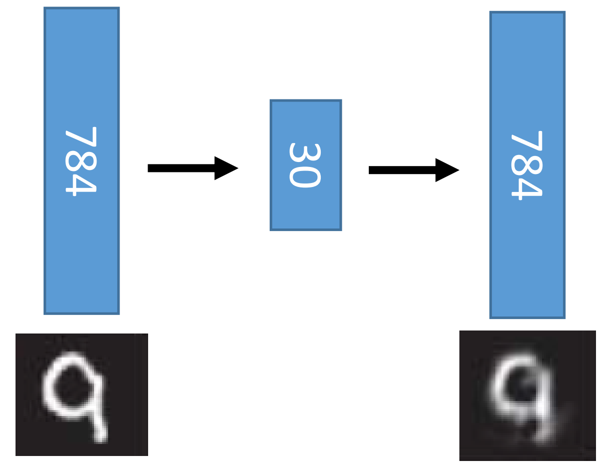
- NN encoder + NN decoder = a deep network

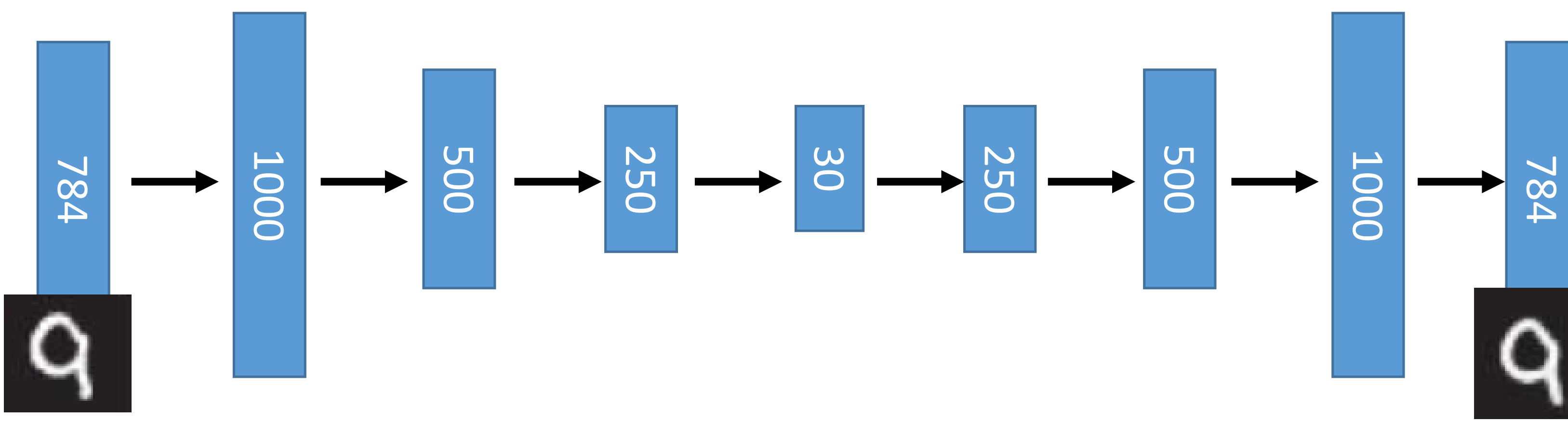


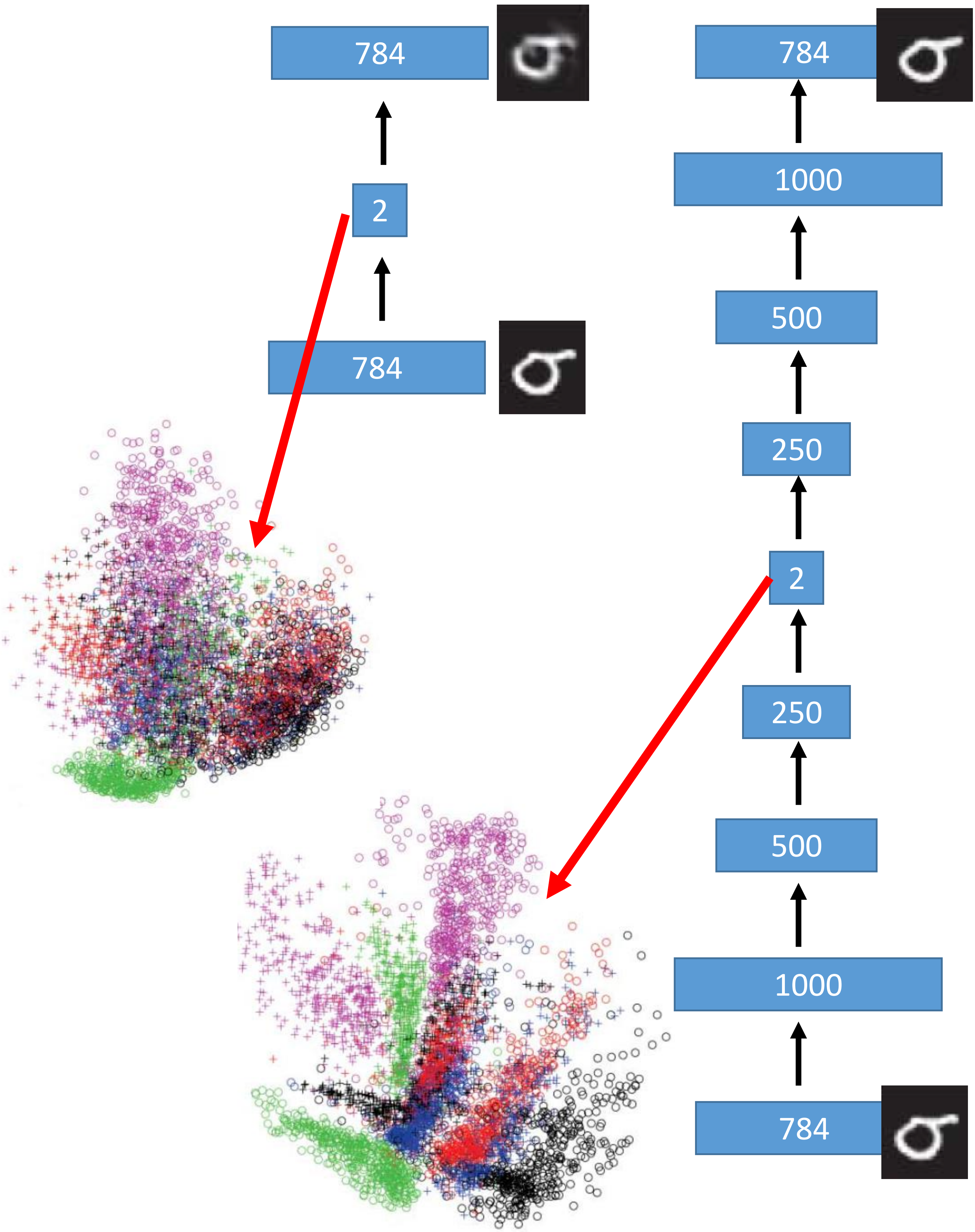
Reference: Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." *Science* 313.5786 (2006): 504-507

Deep Auto-encoder







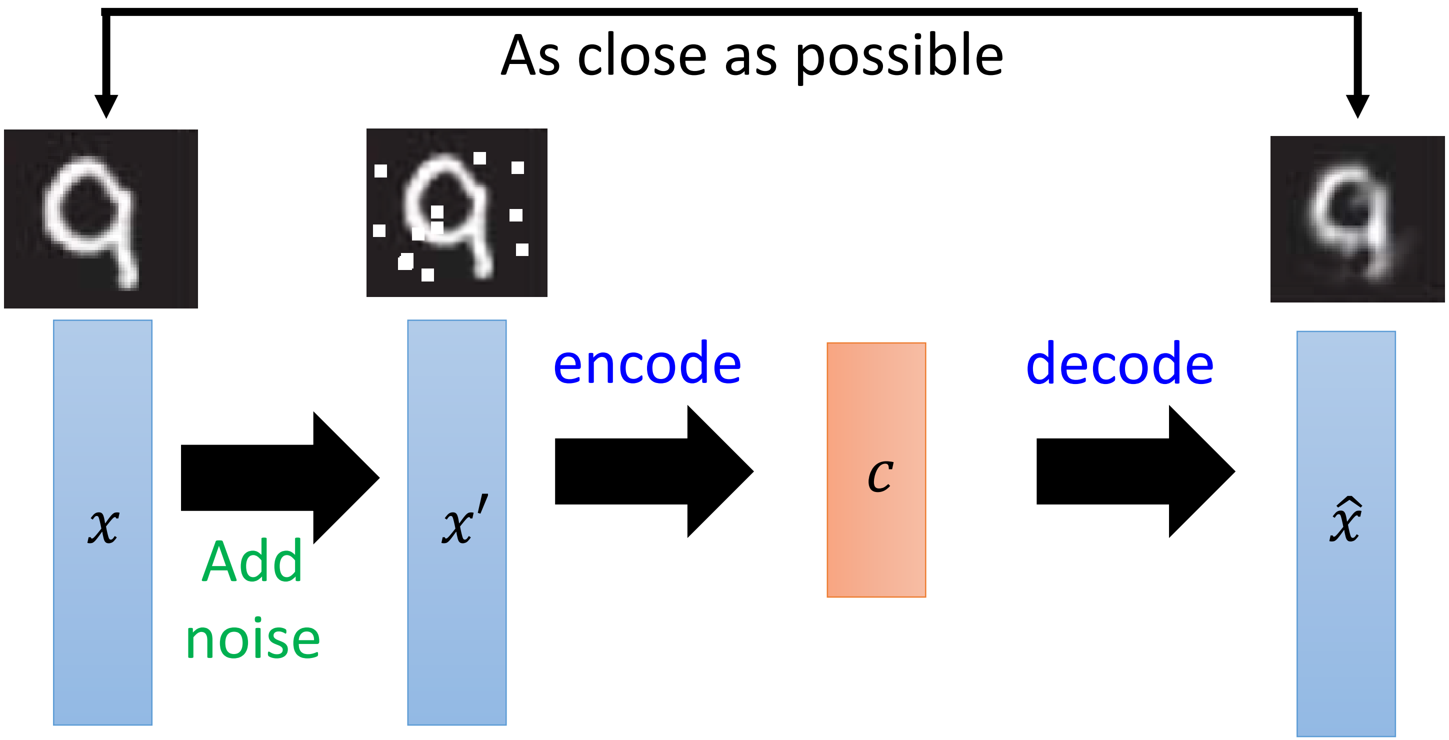

784
2
784
784
1000
500
250
2
250
500
1000
784

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.

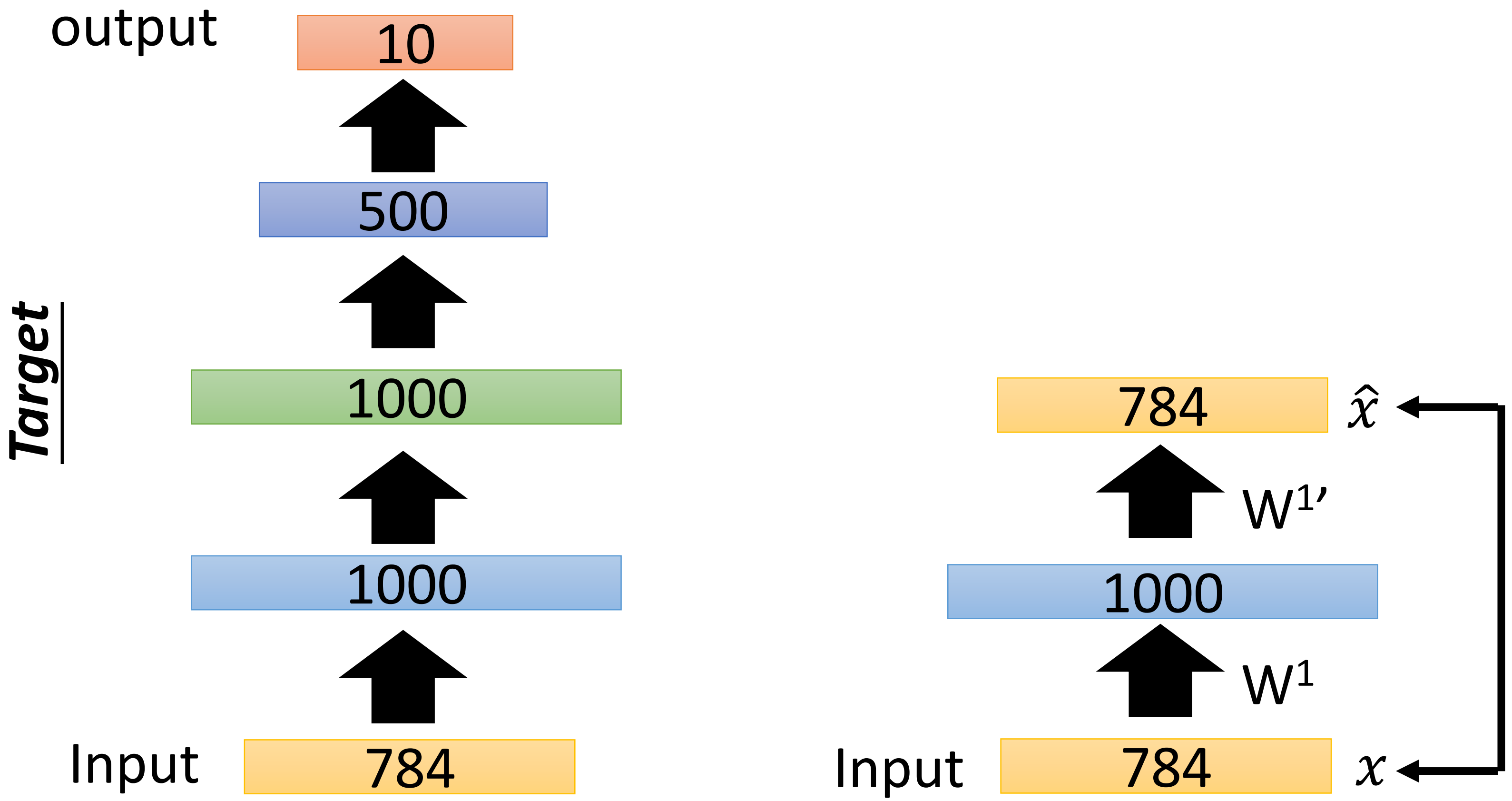
- De-noising auto-encoder



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

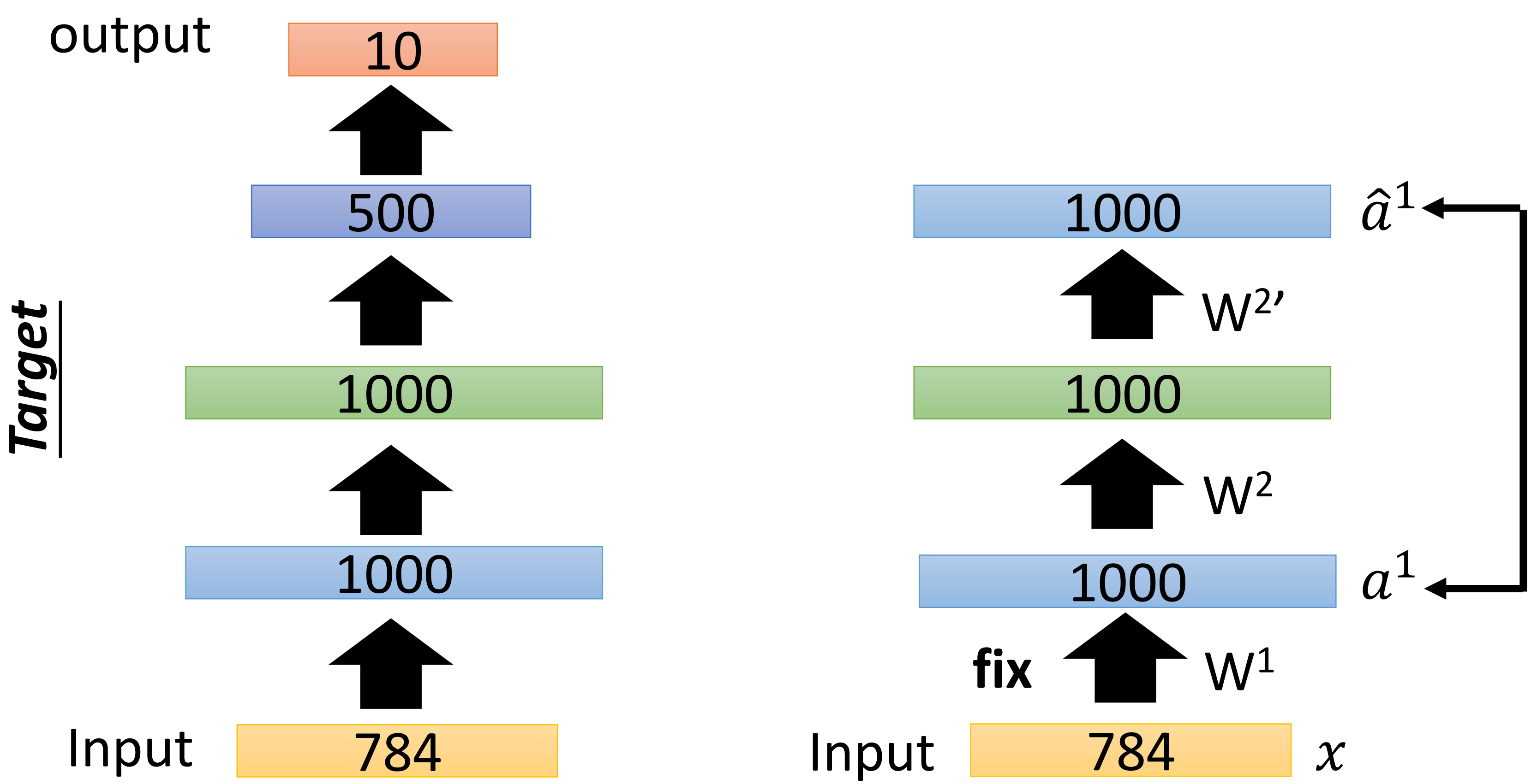
Auto-encoder – Pre-training DNN

- Greedy Layer-wise Pre-training *again*



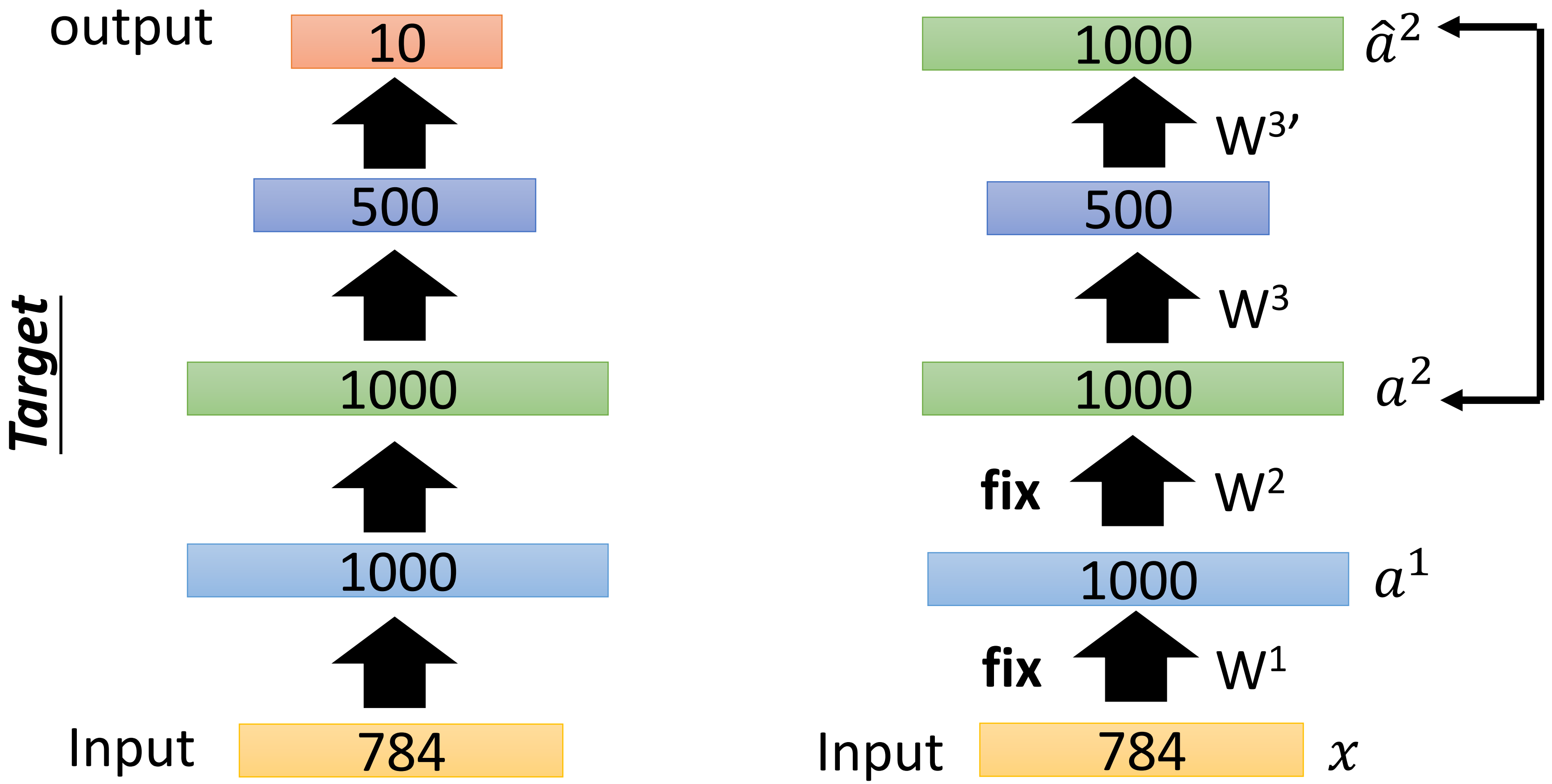
Auto-encoder – Pre-training DNN

- Greedy Layer-wise Pre-training *again*



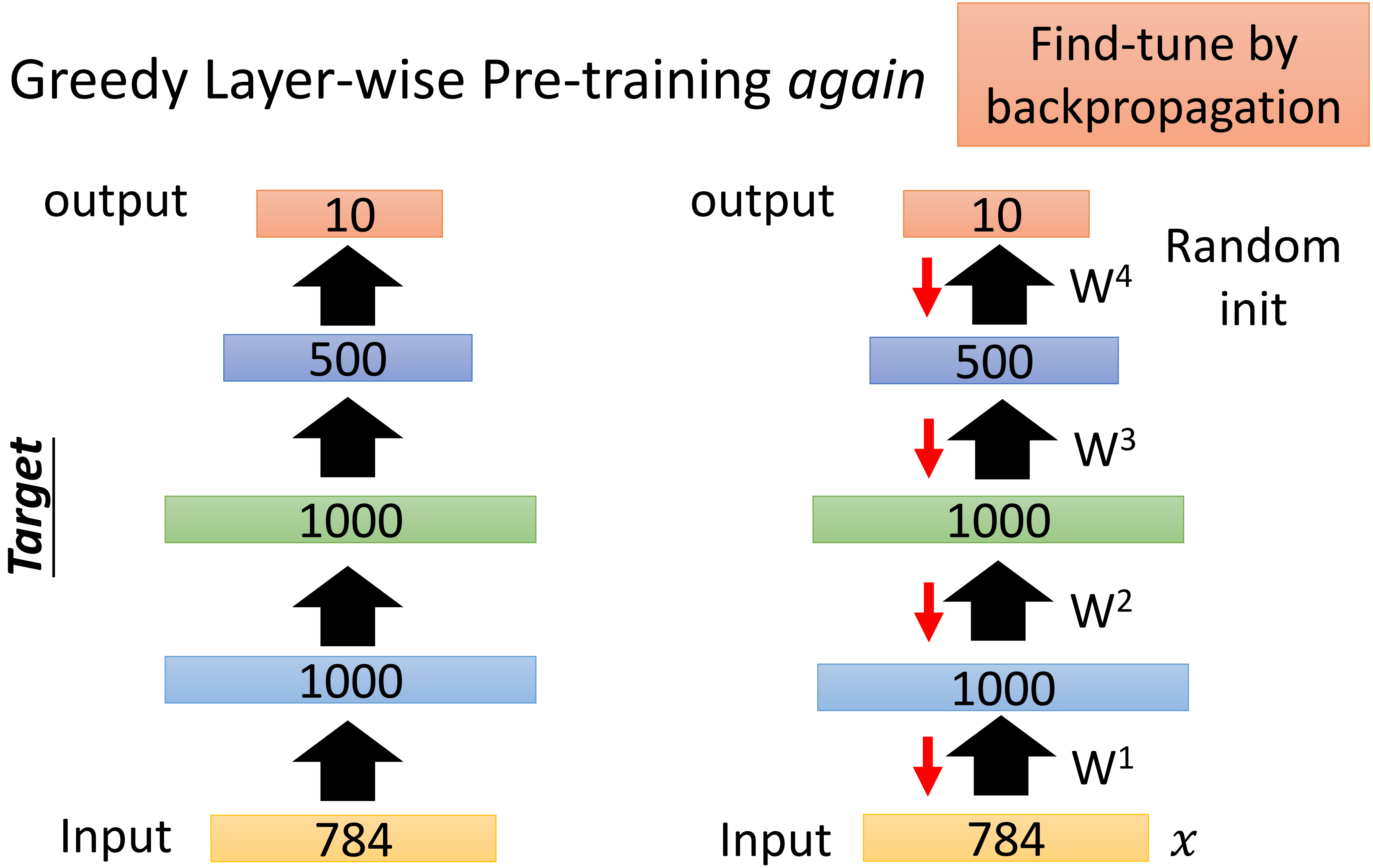
Auto-encoder – Pre-training DNN

- Greedy Layer-wise Pre-training *again*



Auto-encoder – Pre-training DNN

- Greedy Layer-wise Pre-training *again*



Outline

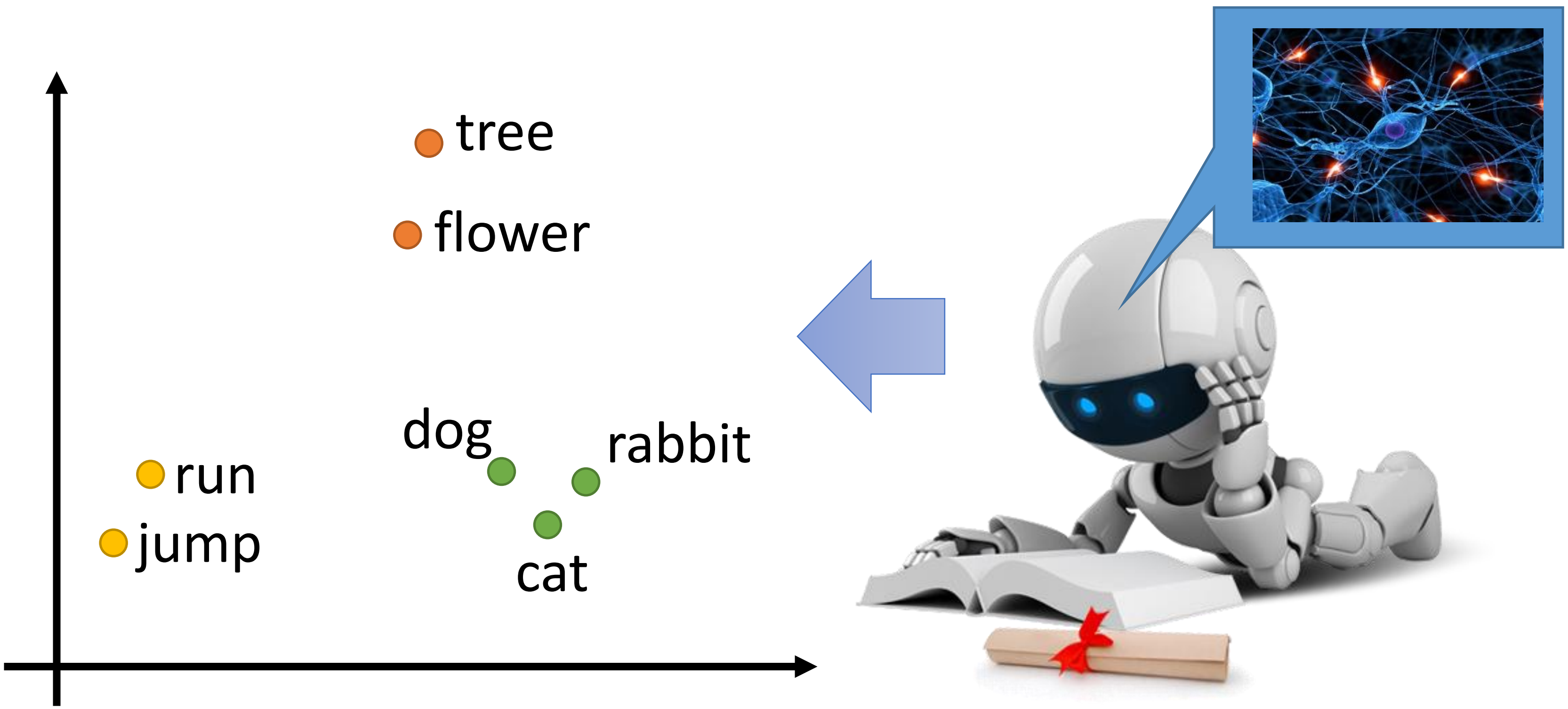
Unsupervised Learning

- 化繁為簡
  - Auto-encoder
  - Word Vector and Audio Word Vector
- 無中生有

Reinforcement Learning

Word Vector/Embedding

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



Word Embedding

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

蔡英文、馬英九 are something very similar

You shall know a word by the company it keeps

馬英九 520宣誓就職

蔡英文 520宣誓就職

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/

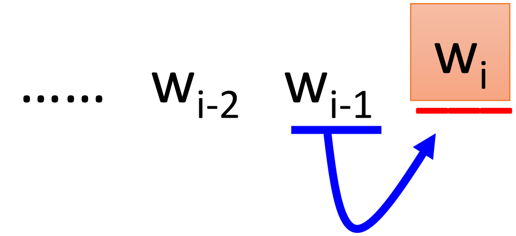
$$V(w_i) \,.\, V(w_j) \Longleftrightarrow N_{i,j}$$

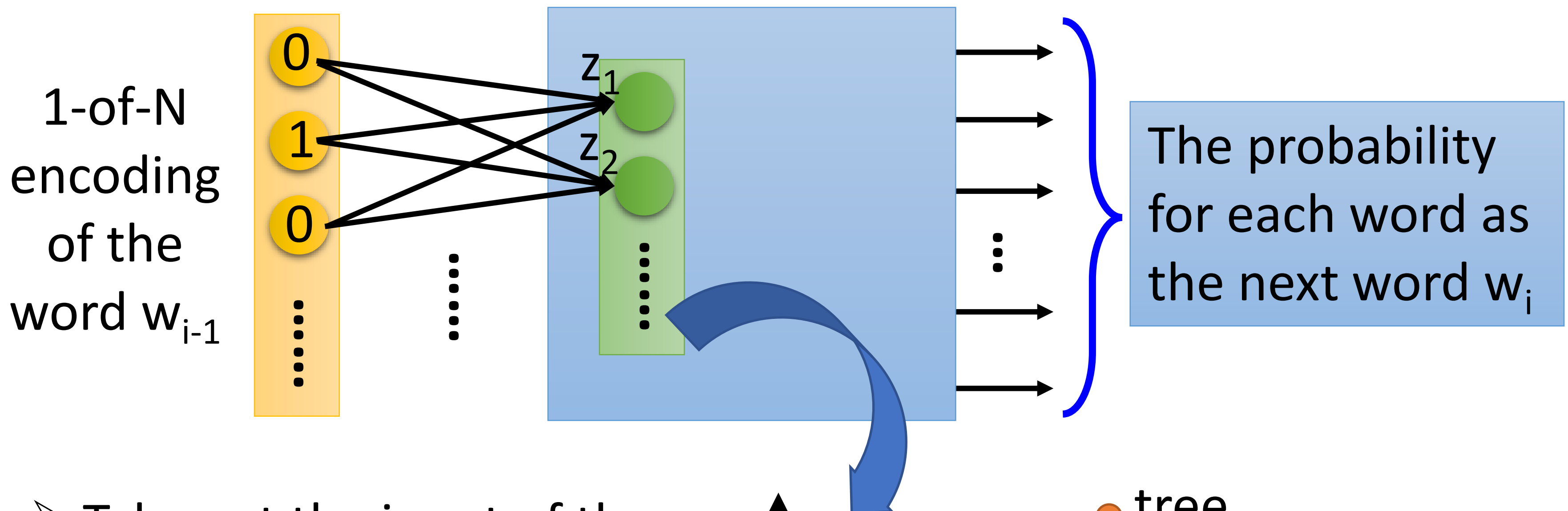
Inner product

Number of times w_i and w_j in the same document

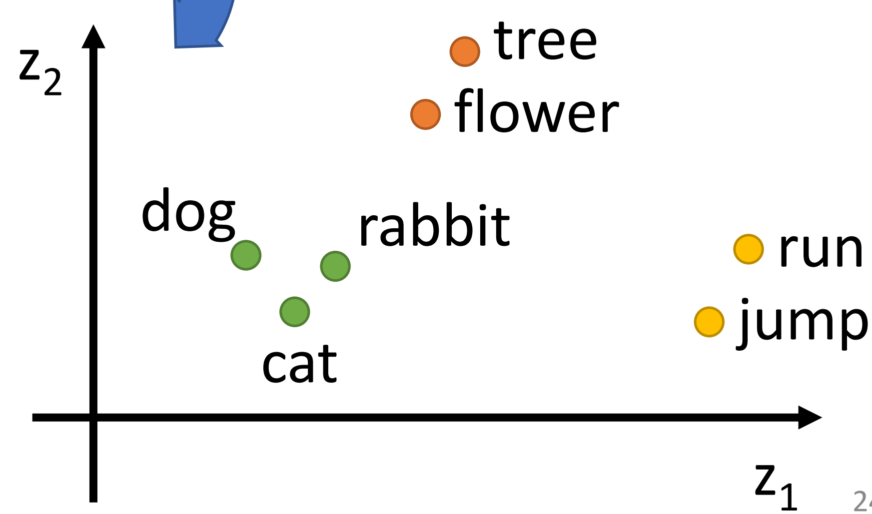
- **Prediction based**

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)

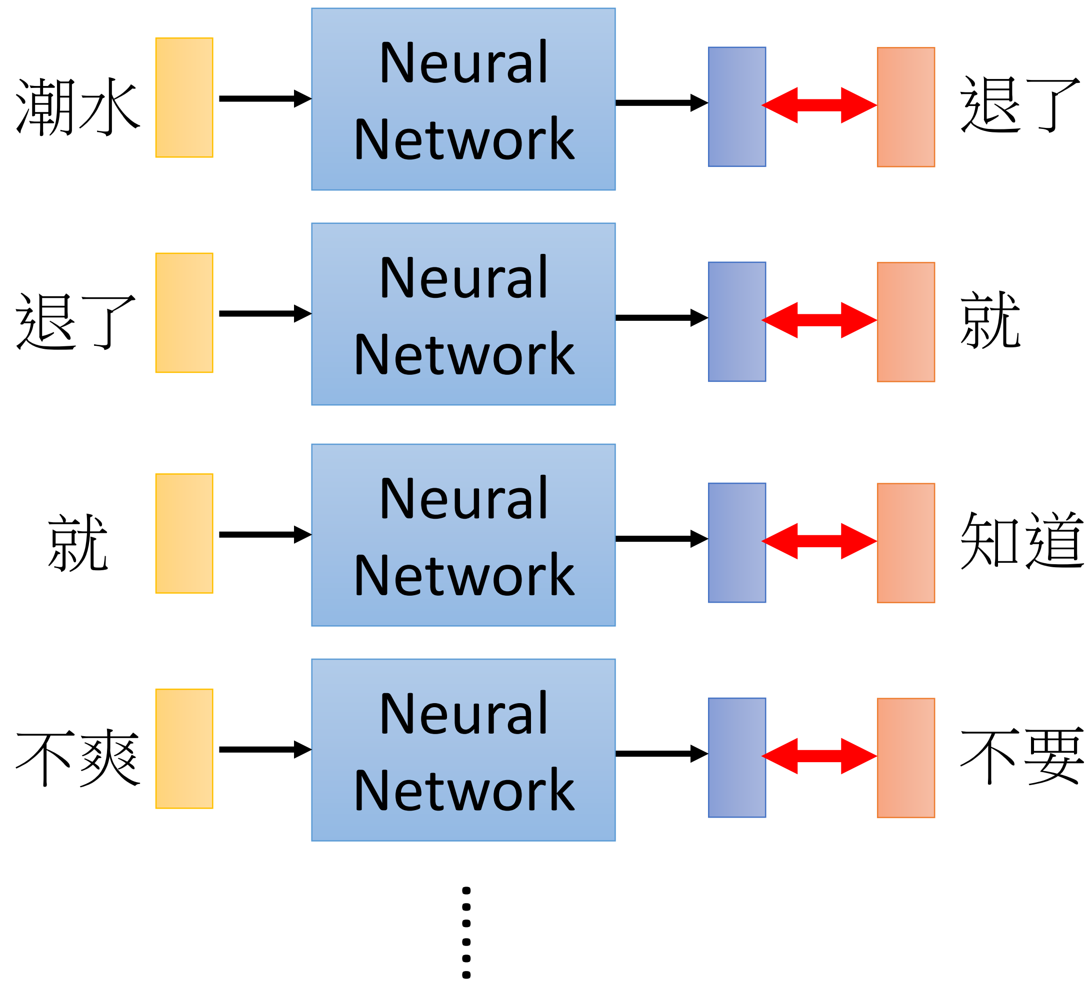


Prediction-based

Minimizing cross entropy

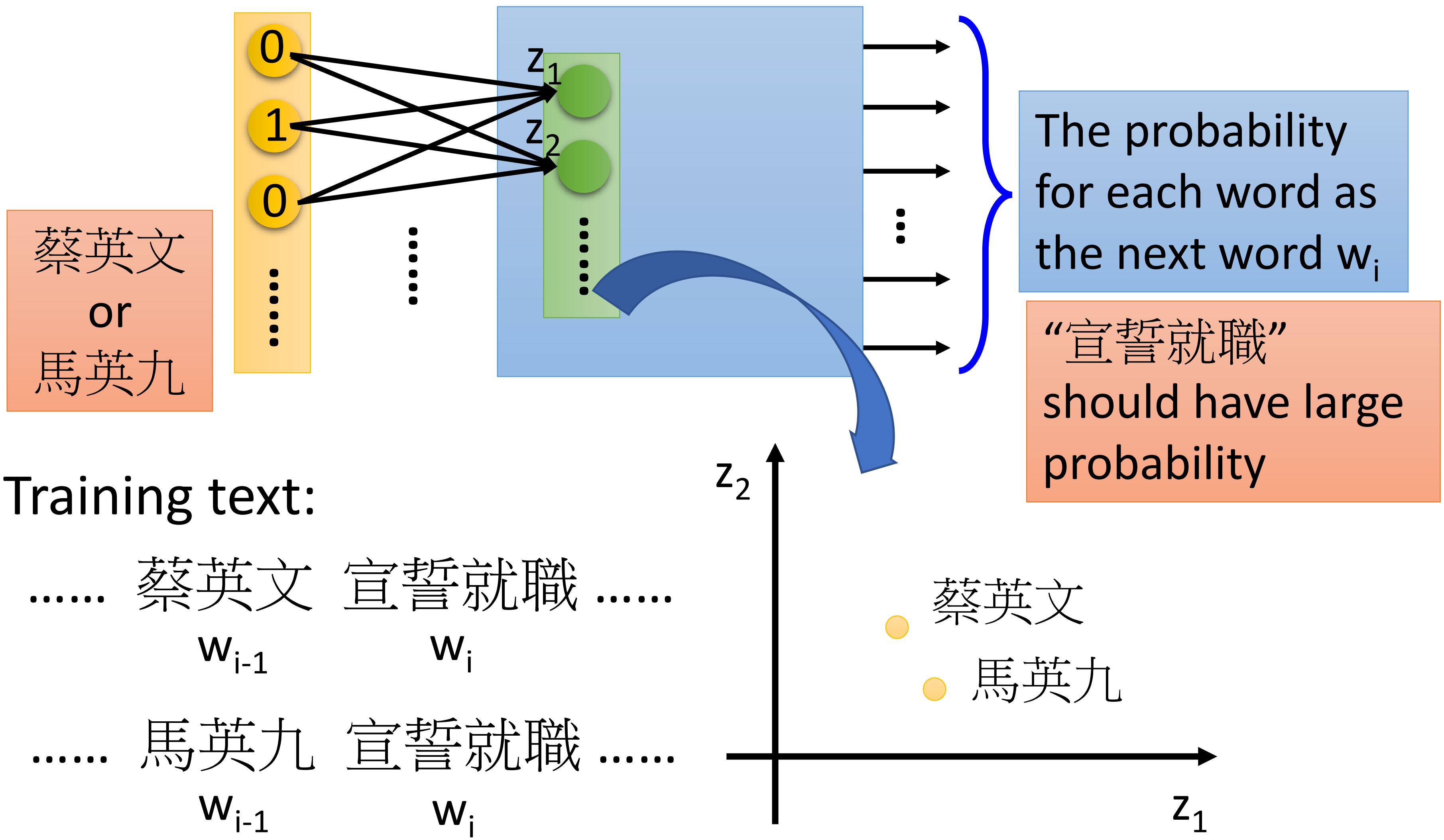
Collect data:

潮水 退了 就 知道 …
不爽 不要 買 …
公道價 八萬 一 …
………



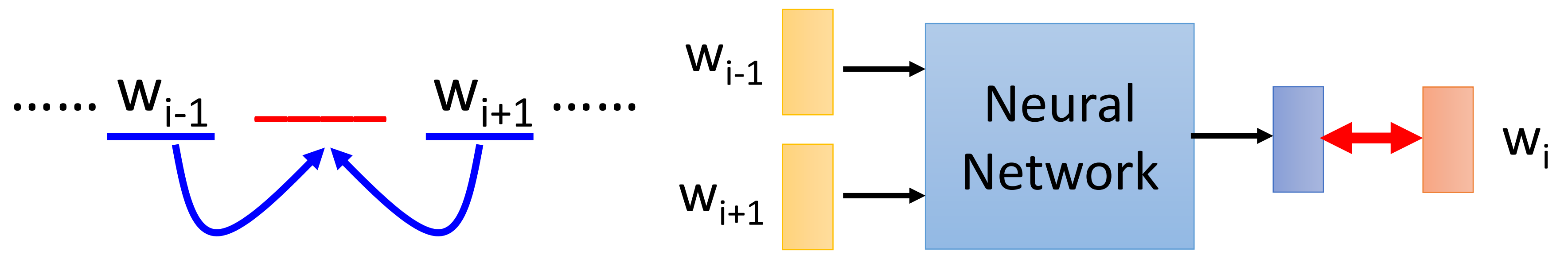
Prediction-based

You shall know a word by the company it keeps



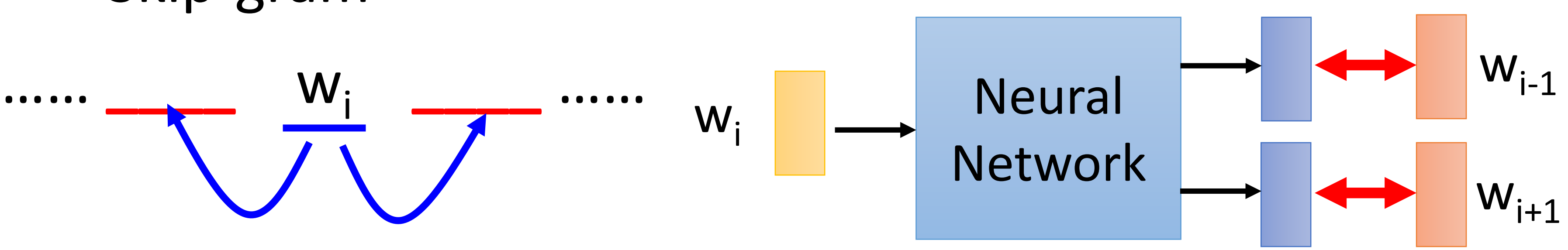
Prediction-based
– Various Architectures

- Continuous bag of word (CBOW) model



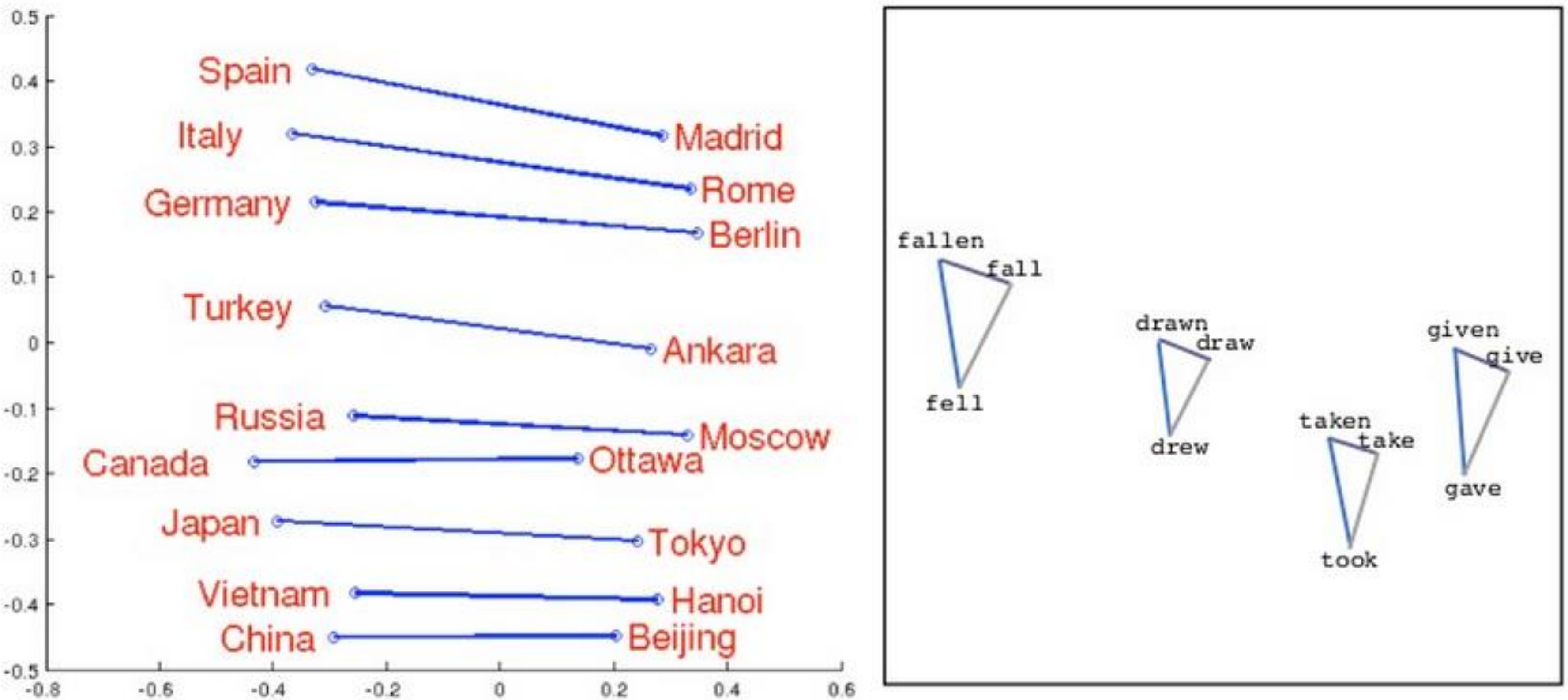
predicting the word given its context

- Skip-gram



predicting the context given a word

Word Embedding



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

Word Embedding

- Characteristics

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

$$V(hotter) - V(hot) \approx V(bigger) - V(big)$$
$$V(Rome) - V(Italy) \approx V(Berlin) - V(Germany)$$
$$V(king) - V(queen) \approx V(uncle) - V(aunt)$$

- Solving analogies

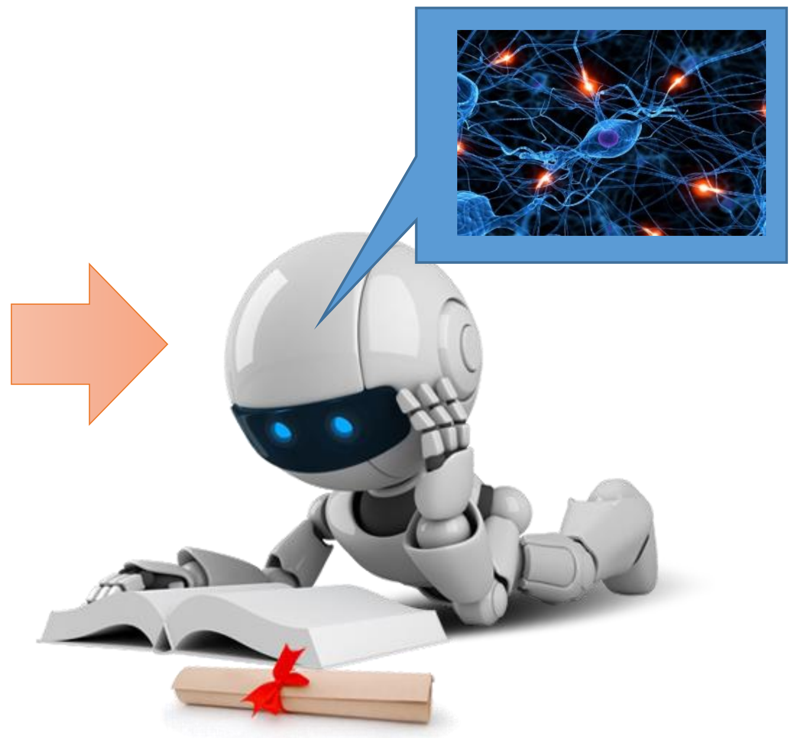
Rome : Italy = Berlin : ?

Compute $\underline{V(Berlin) - V(Rome) + V(Italy)}$

Find the word w with the closest V(w)

Demo

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



Demo

- Model used in demo is provided by 陳仰德
  - Part of the project done by 陳仰德、林資偉
  - TA: 劉元銘
  - Training data is from PTT (collected by 葉青峰)

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

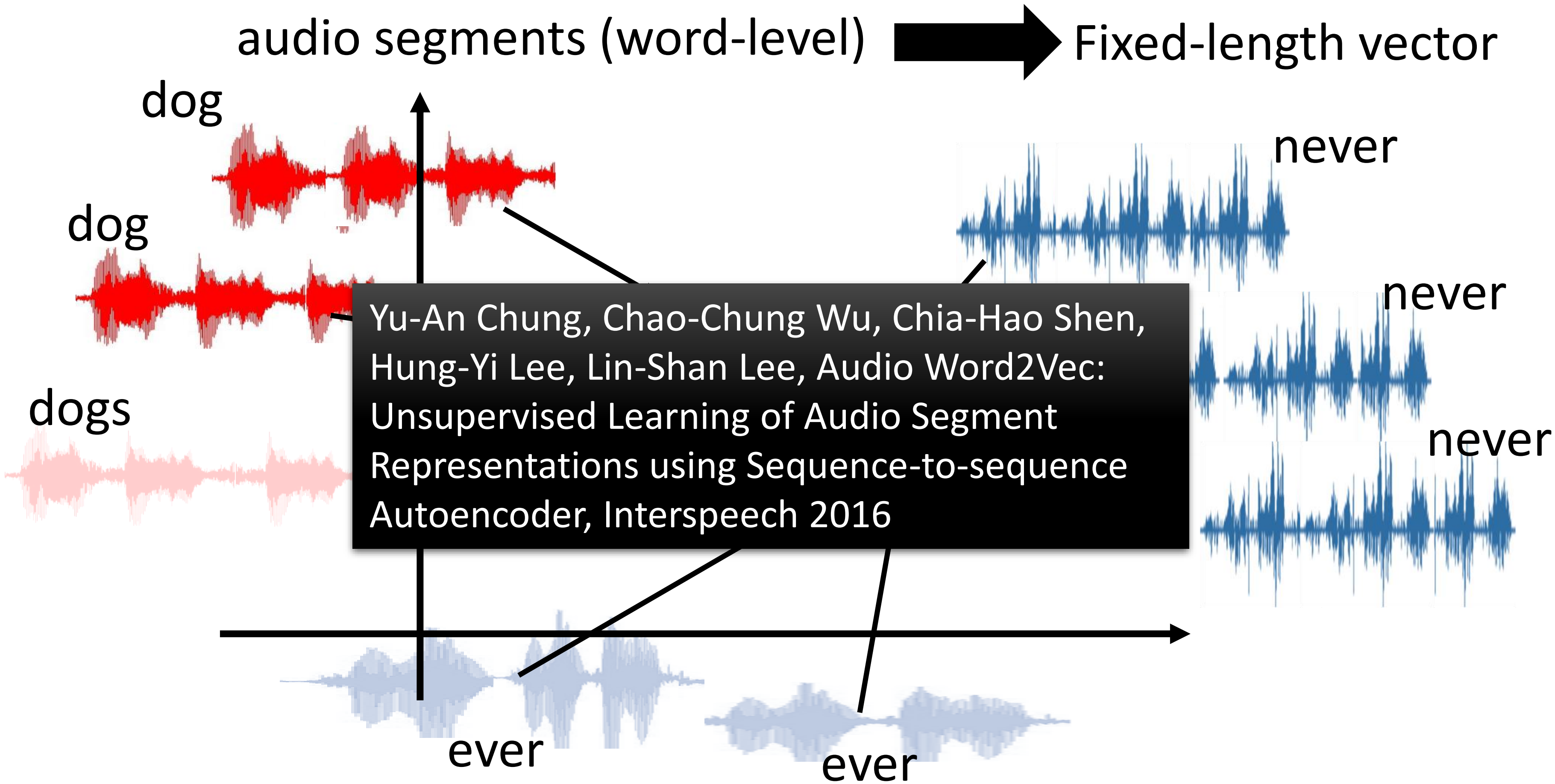
Machine listens to lots of audio book

Like an infant

[Chung, Interspeech 16)

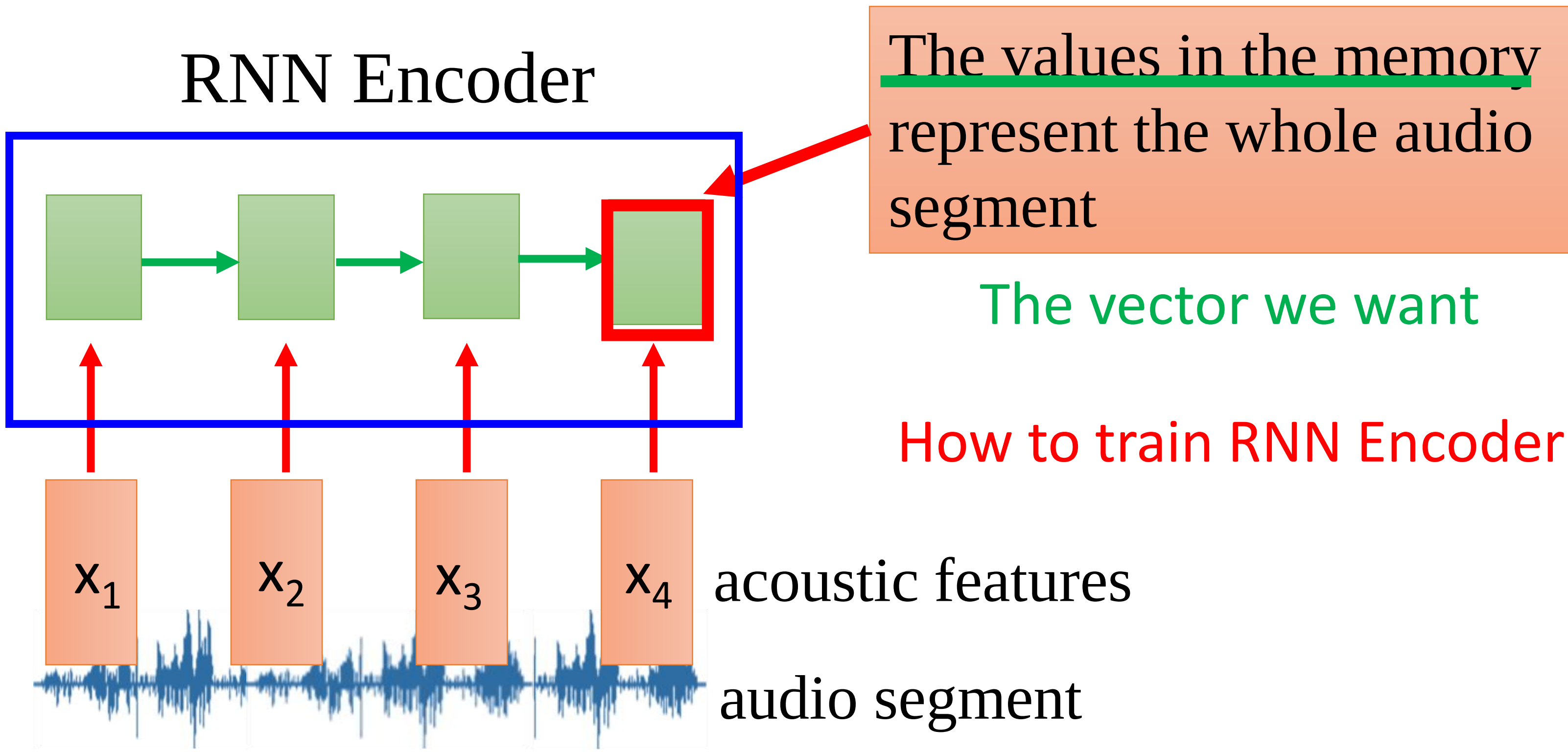
Audio Word to Vector

- Dimension reduction for a sequence with variable length



Yu-An Chung, Chao-Chung Wu, Chia-Hao Shen, Hung-Yi Lee, Lin-Shan Lee, Audio Word2Vec: Unsupervised Learning of Audio Segment Representations using Sequence-to-sequence Autoencoder, Interspeech 2016

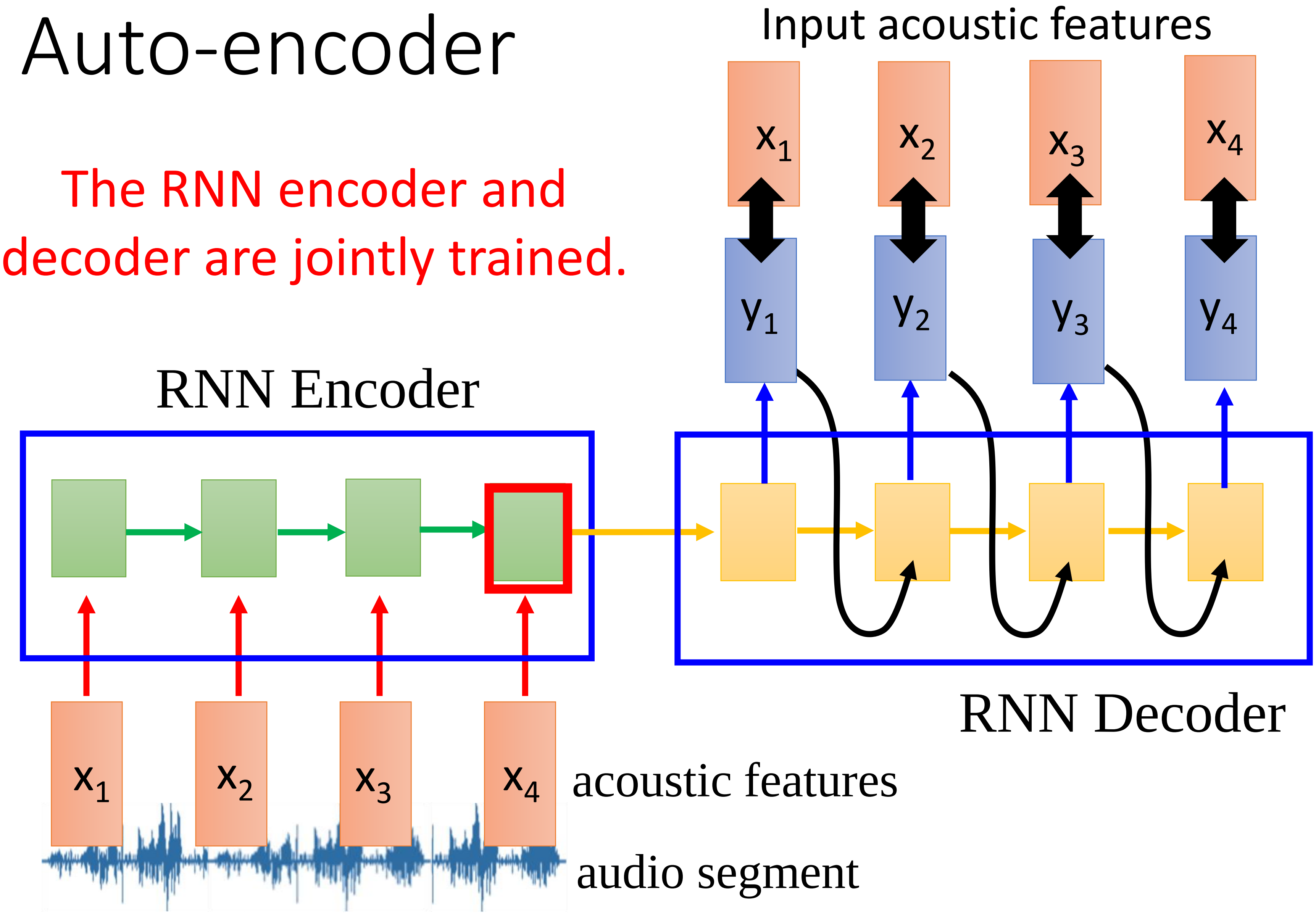
Sequence-to-sequence Auto-encoder



How to train RNN Encoder?

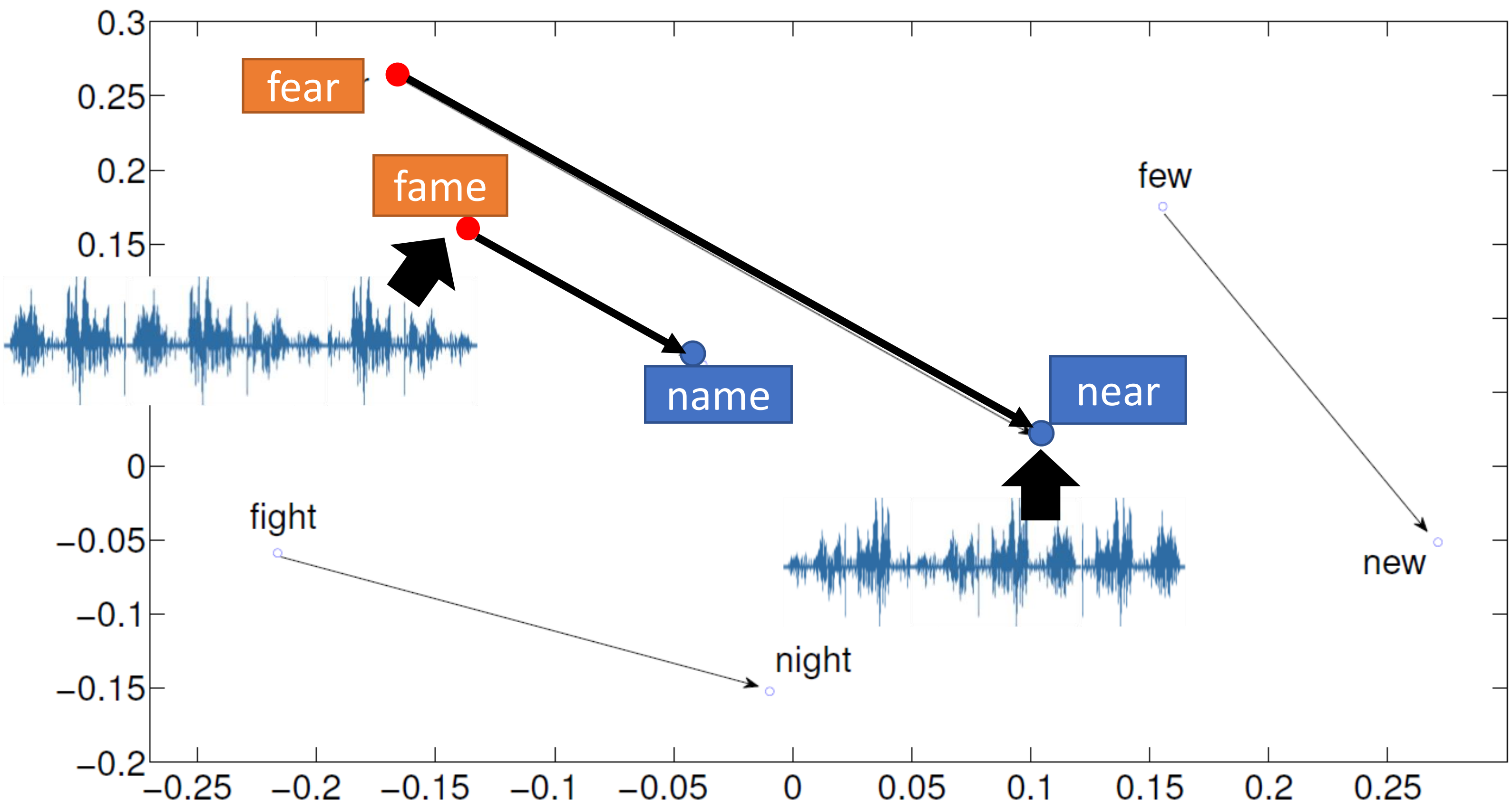
Sequence-to-sequence Auto-encoder

The RNN encoder and decoder are jointly trained.

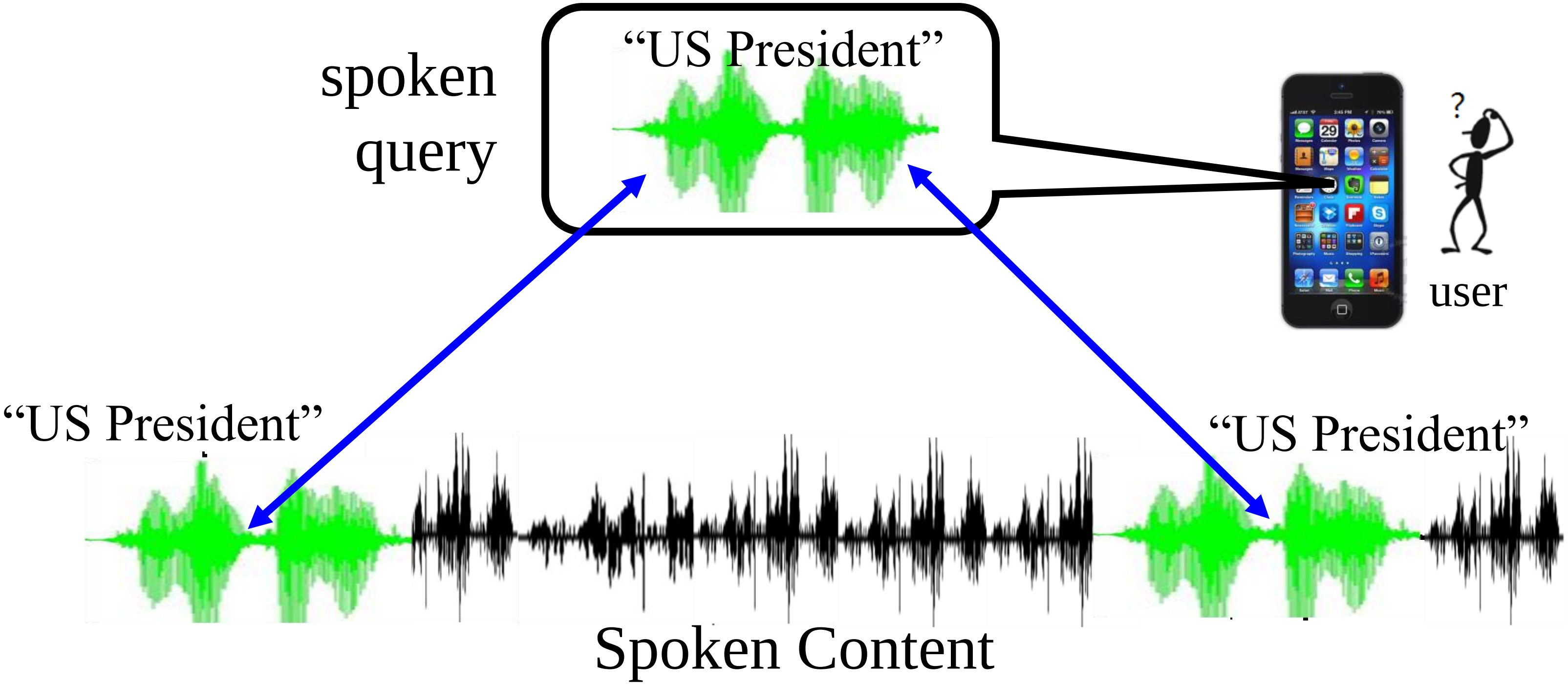


Sequence-to-sequence Auto-encoder

- Visualizing embedding vectors of the words



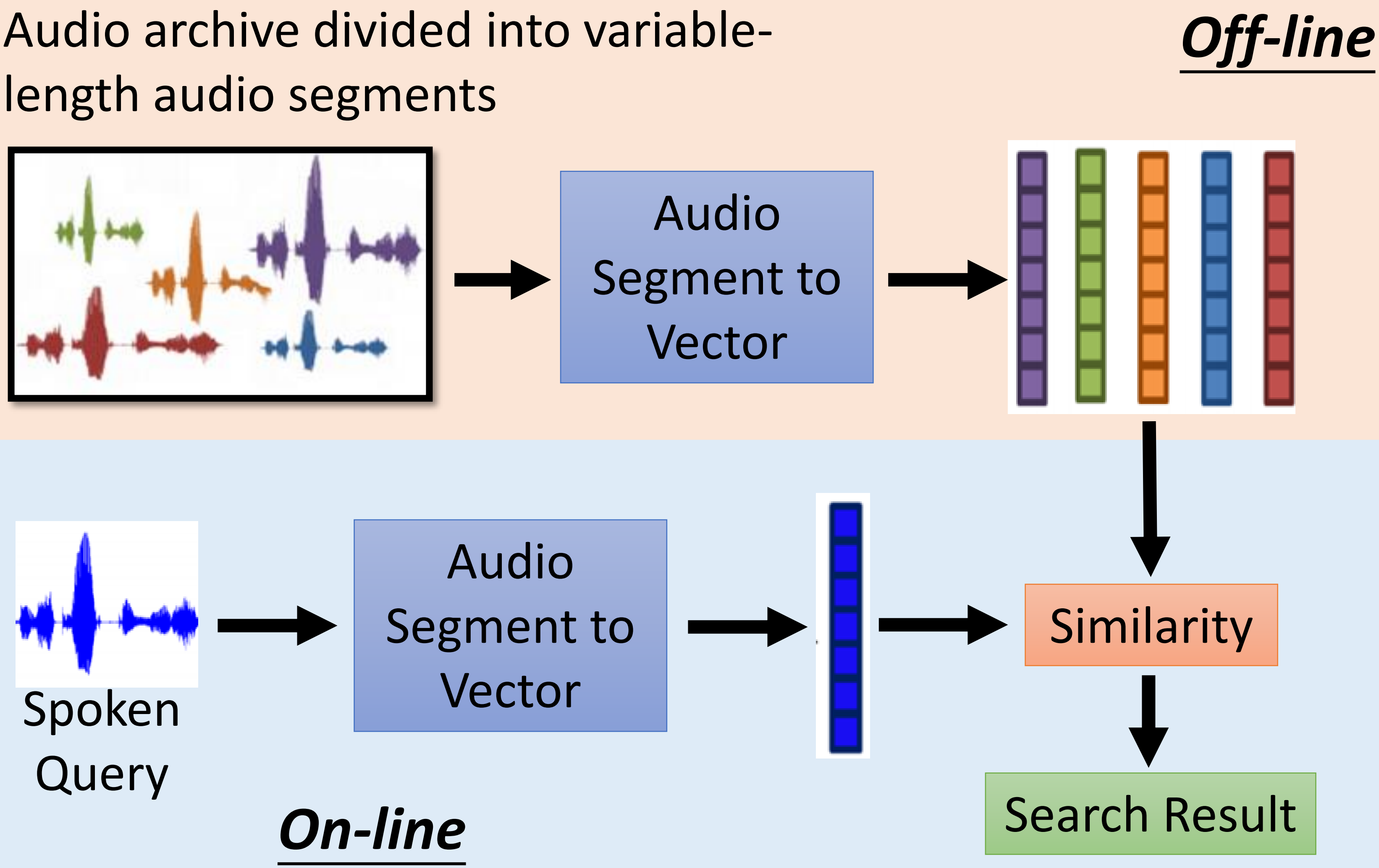
Audio Word to Vector –Application



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

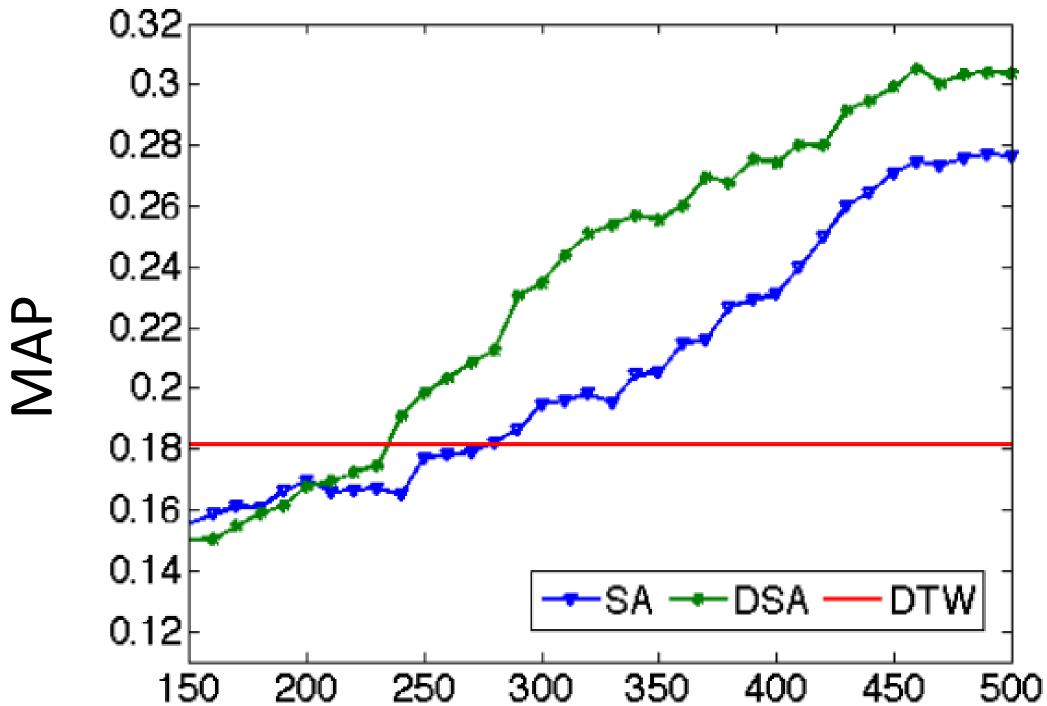
Audio Word to Vector –Application

Audio archive divided into variable-length audio segments



Experimental Results

- Query-by-Example Spoken Term Detection



training epochs for sequence auto-encoder

SA: sequence auto-encoder

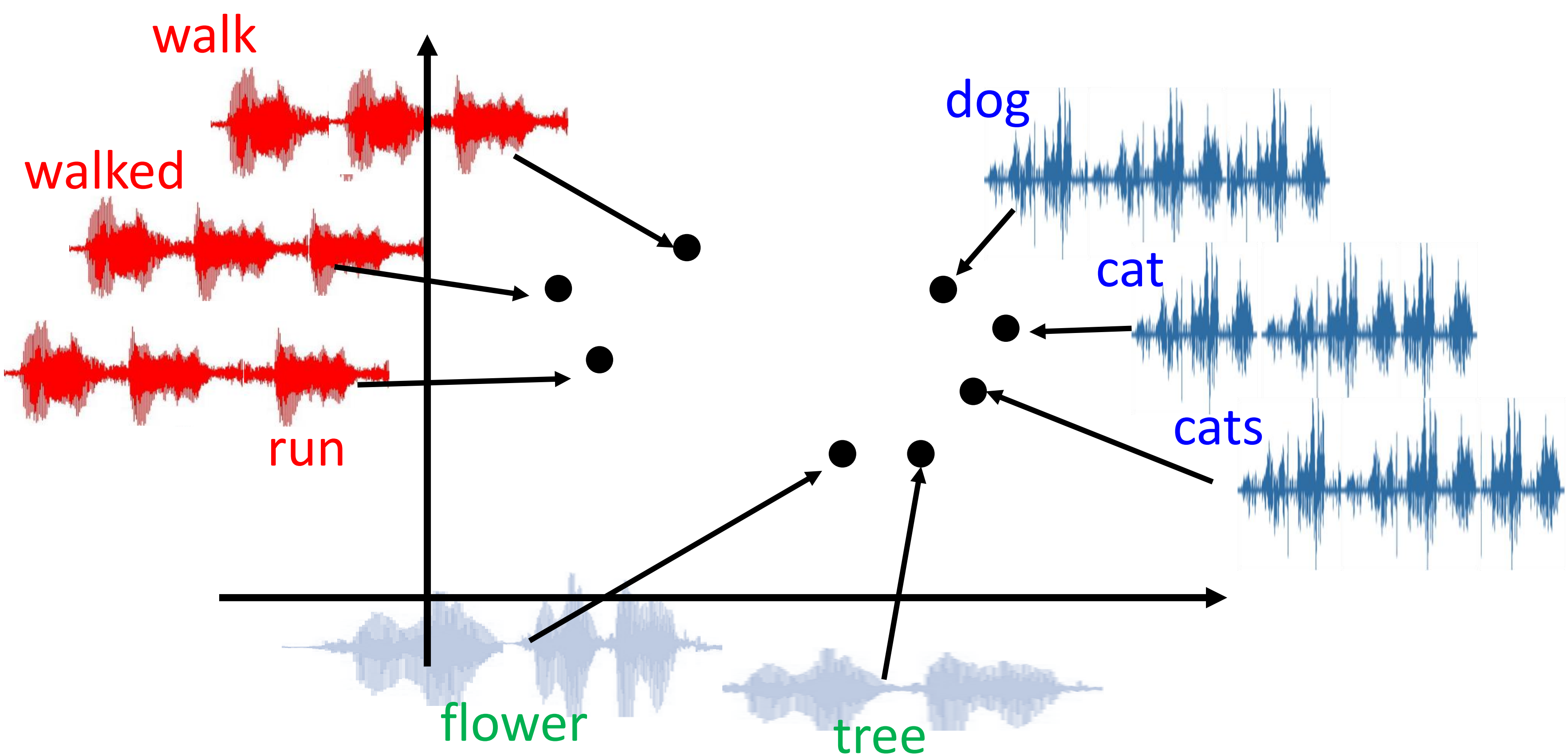
DSA: de-noising sequence auto-encoder

Input: clean speech + noise

output: clean speech

Next Step ……

- Can we include semantics?



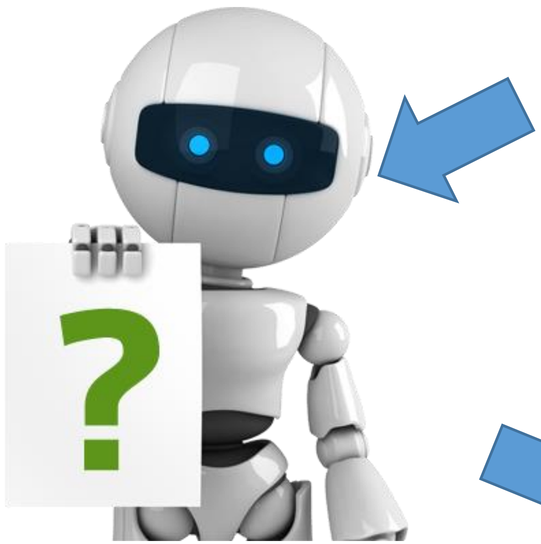
Outline

Unsupervised Learning

- 化繁為簡
  - Auto-encoder
  - Word Vector and Audio Word Vector
- 無中生有

Reinforcement Learning

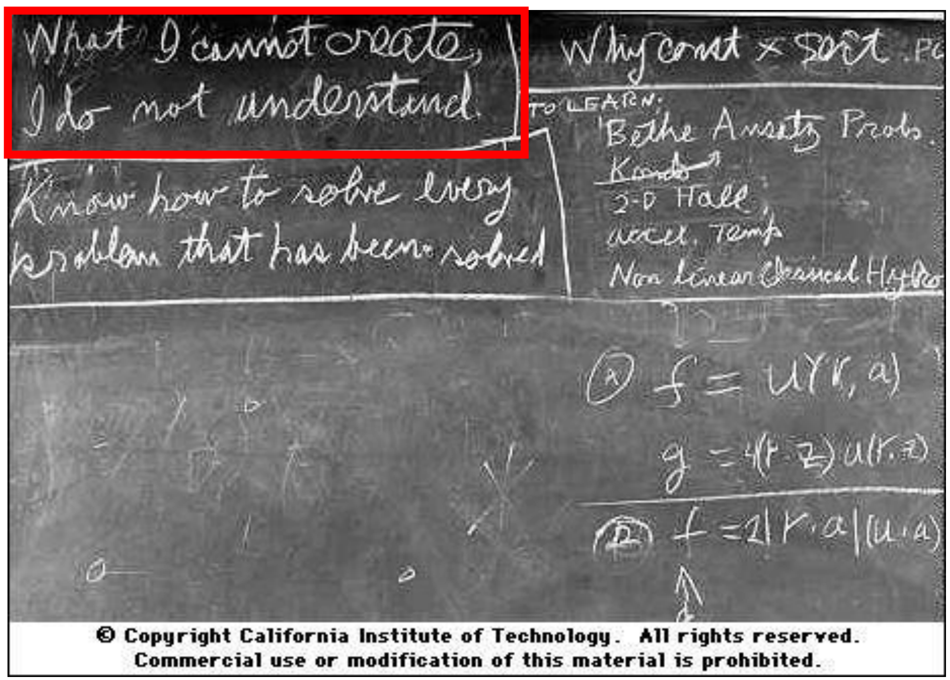
Creation



Draw something!

Creation

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



What I cannot create,
I do not understand.

Richard Feynman

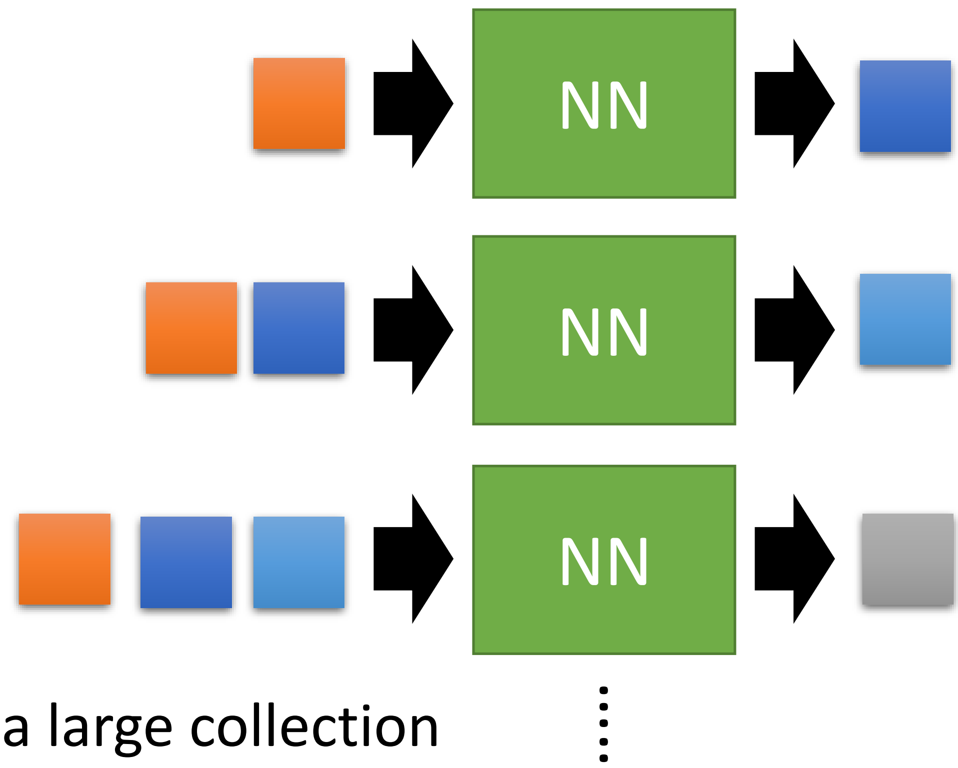
https://www.quora.com/What-did-Richard-Feynman-mean-when-he-said-What-I-cannot-create-I-do-not-understand

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

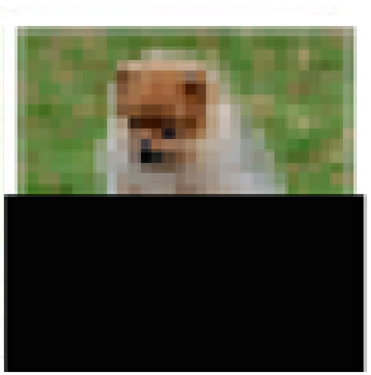
E.g. 3 x 3 images



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

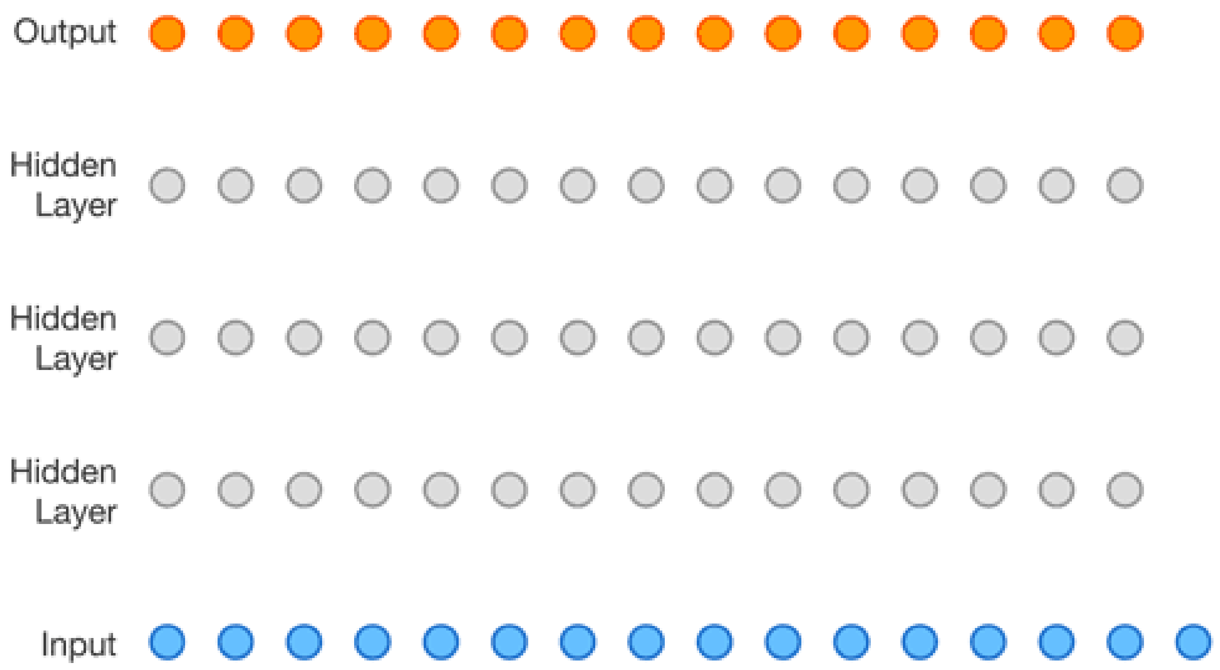
PixelRNN

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



Real World

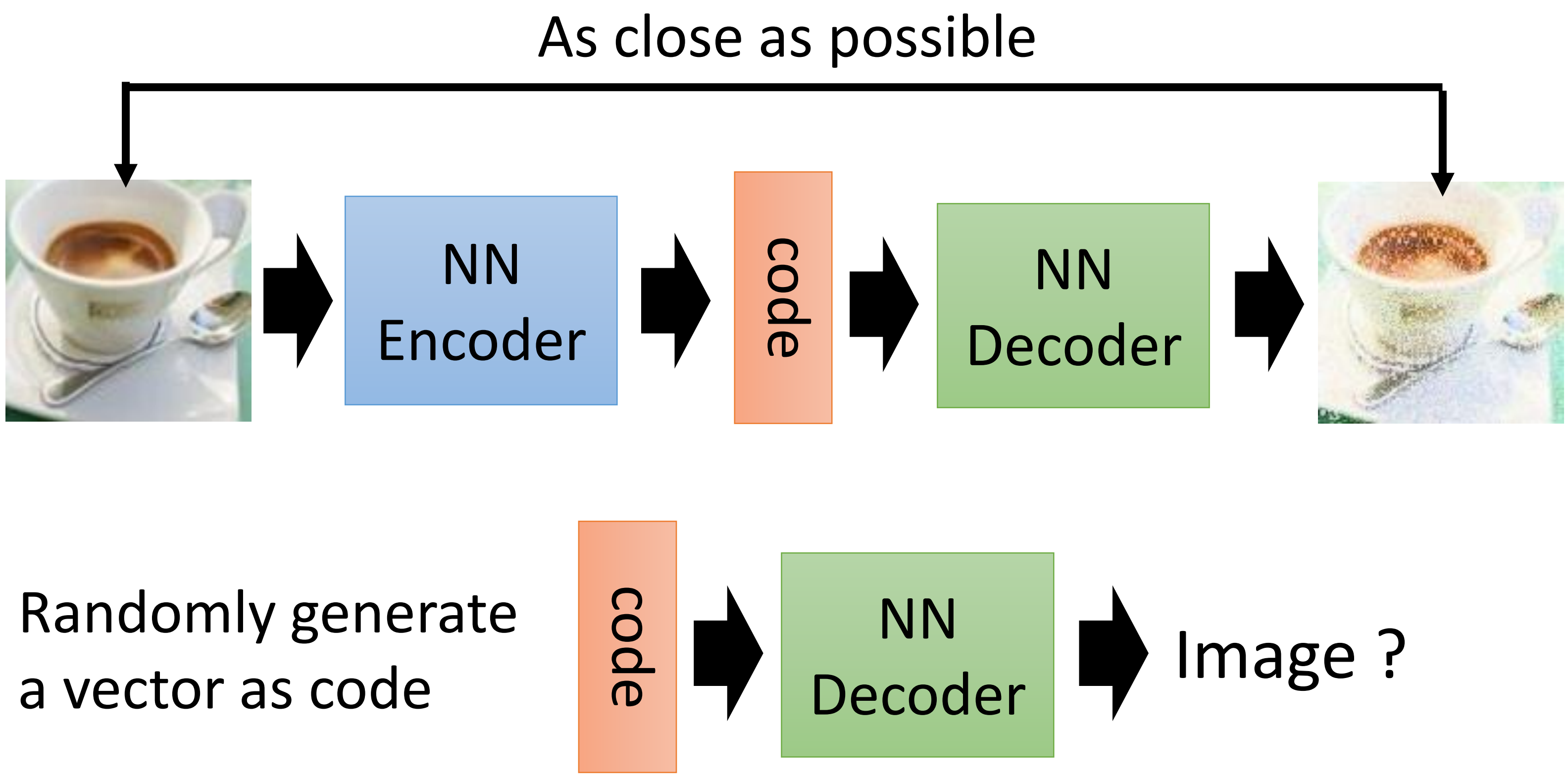
PixelRNN – beyond Image



Audio: Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, Koray Kavukcuoglu, WaveNet: A Generative Model for Raw Audio, arXiv preprint, 2016

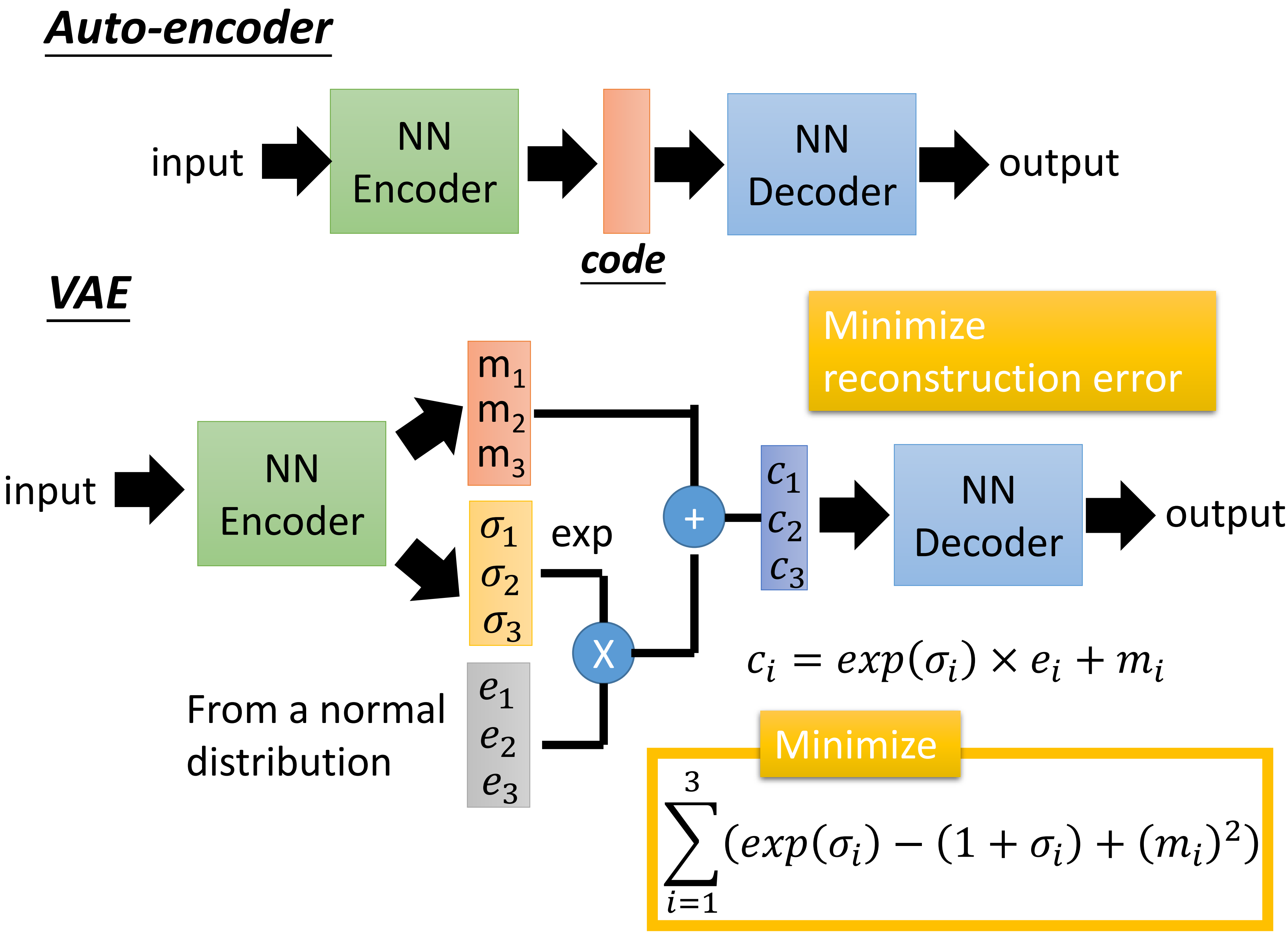
Video: Nal Kalchbrenner, Aaron van den Oord, Karen Simonyan, Ivo Danihelka, Oriol Vinyals, Alex Graves, Koray Kavukcuoglu, Video Pixel Networks , arXiv preprint, 2016

Auto-encoder

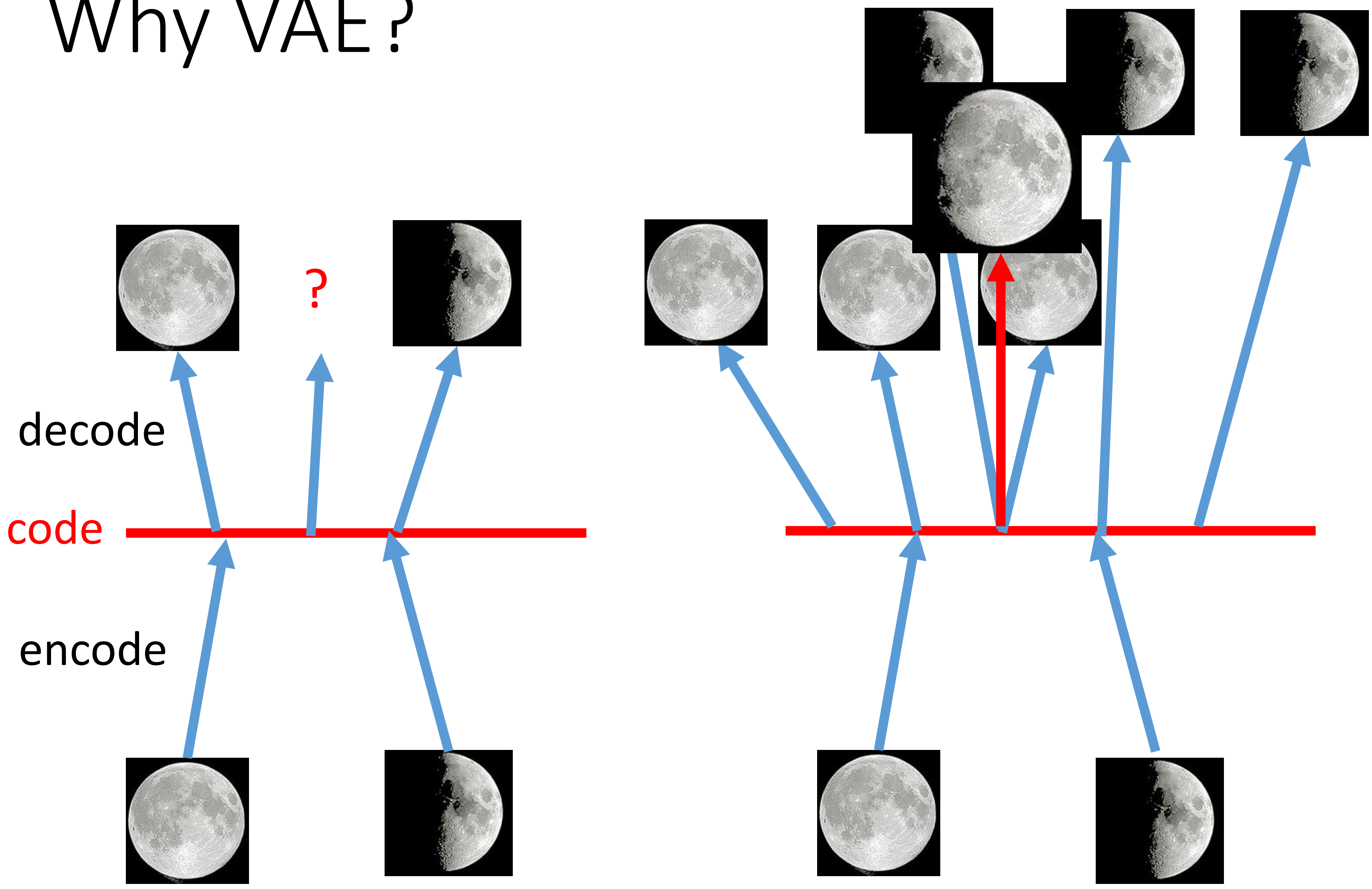


<u>Variation Auto-encoder (VAE)</u>

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


Auto-encoder
input
NN Encoder
code
NN Decoder
output
VAE
input
NN Encoder
m_1 m_2 m_3
σ_1 σ_2 σ_3
exp
From a normal distribution
e_1 e_2 e_3
X
+
c_1 c_2 c_3
NN Decoder
output
Minimize reconstruction error
$c_i = exp(\sigma_i) \times e_i + m_i$
Minimize
$\sum_{i=1}^{3} (exp(\sigma_i) - (1 + \sigma_i) + (m_i)^2)$

Why VAE?



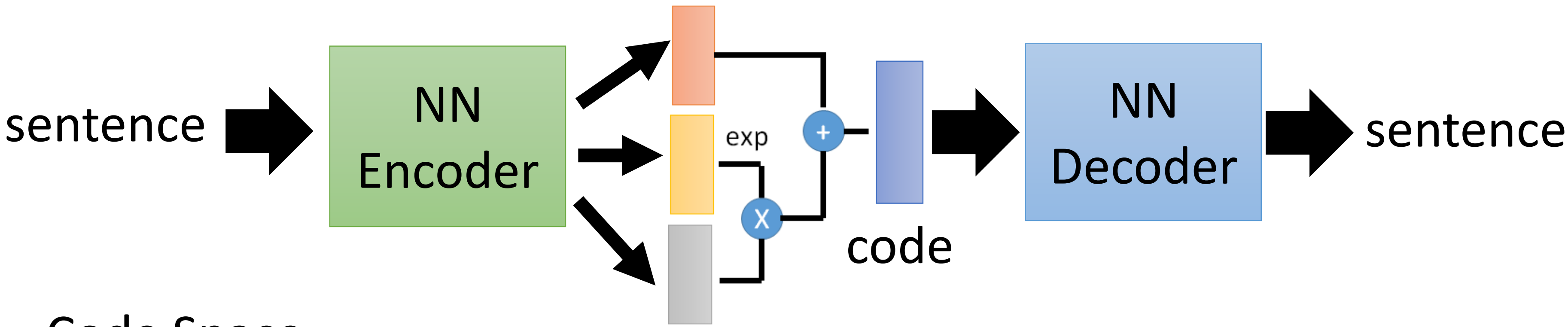
VAE

Cifar-10

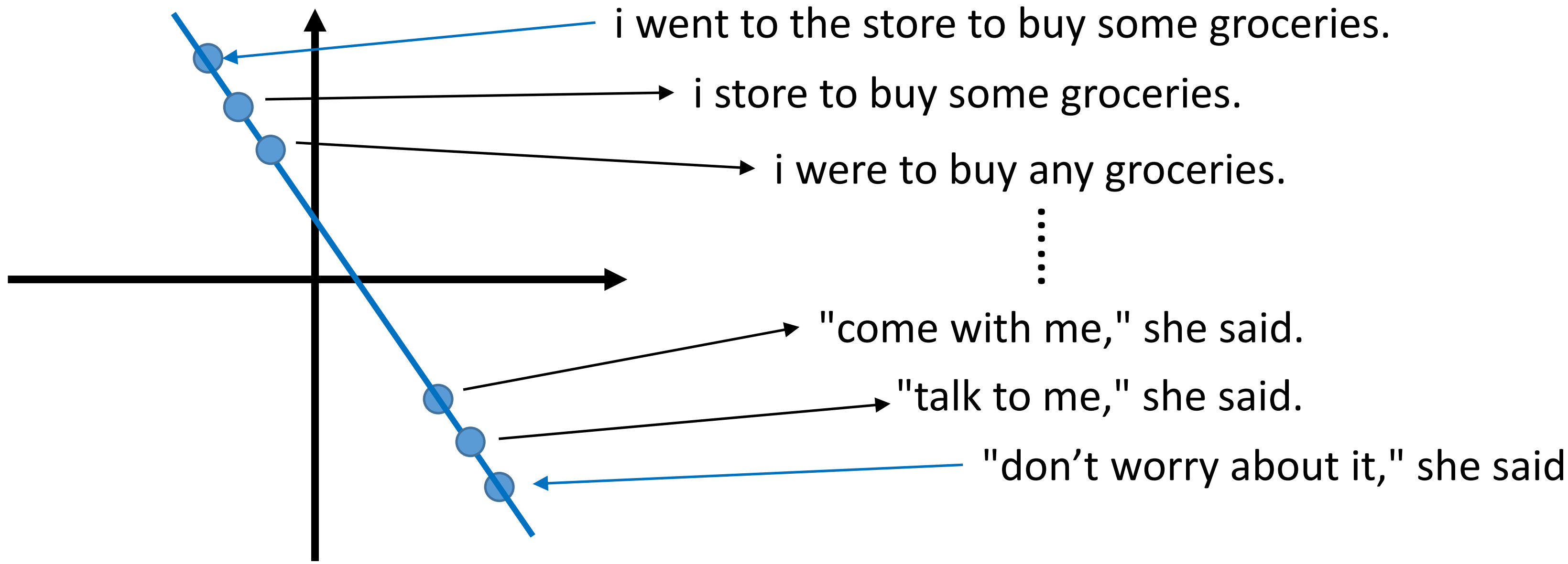
https://github.com/openai/iaf

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

VAE - Writing Poetry



Code Space

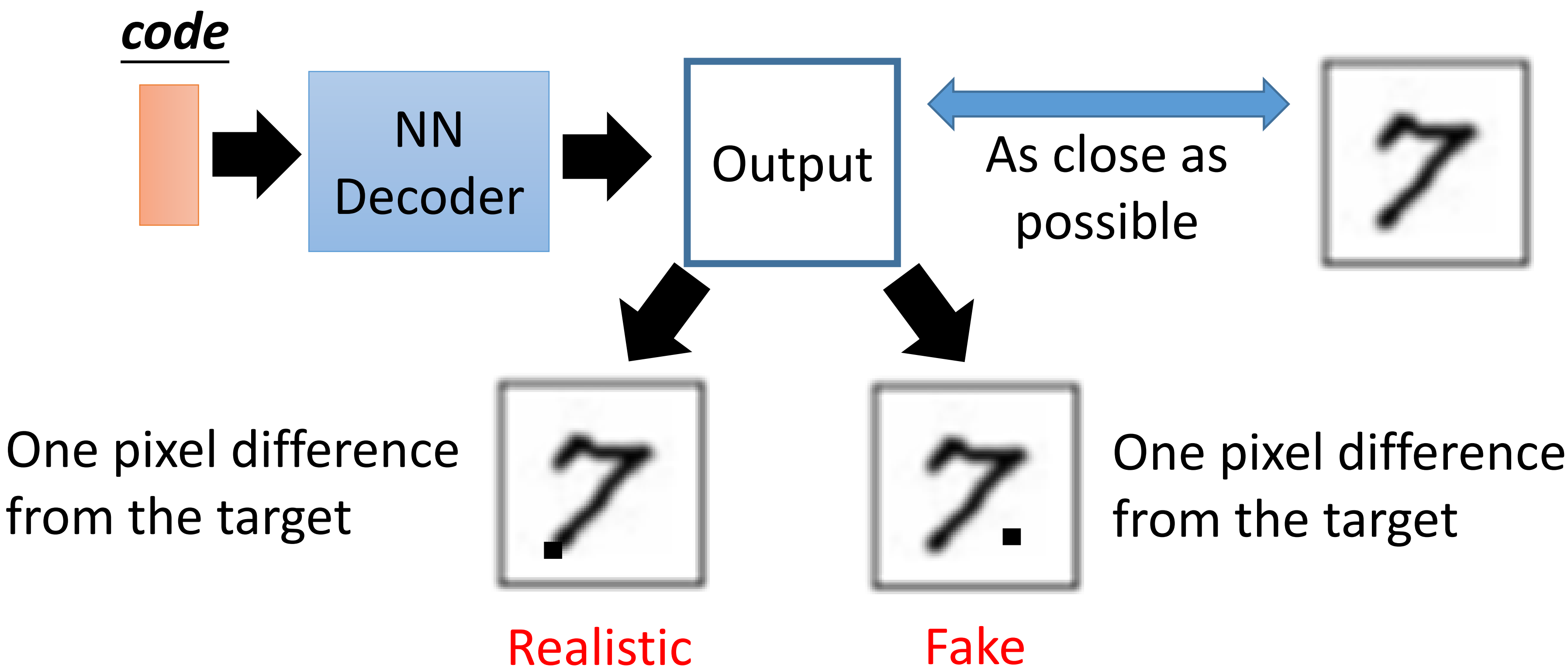


Ref: http://www.wired.co.uk/article/google-artificial-intelligence-poetry

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

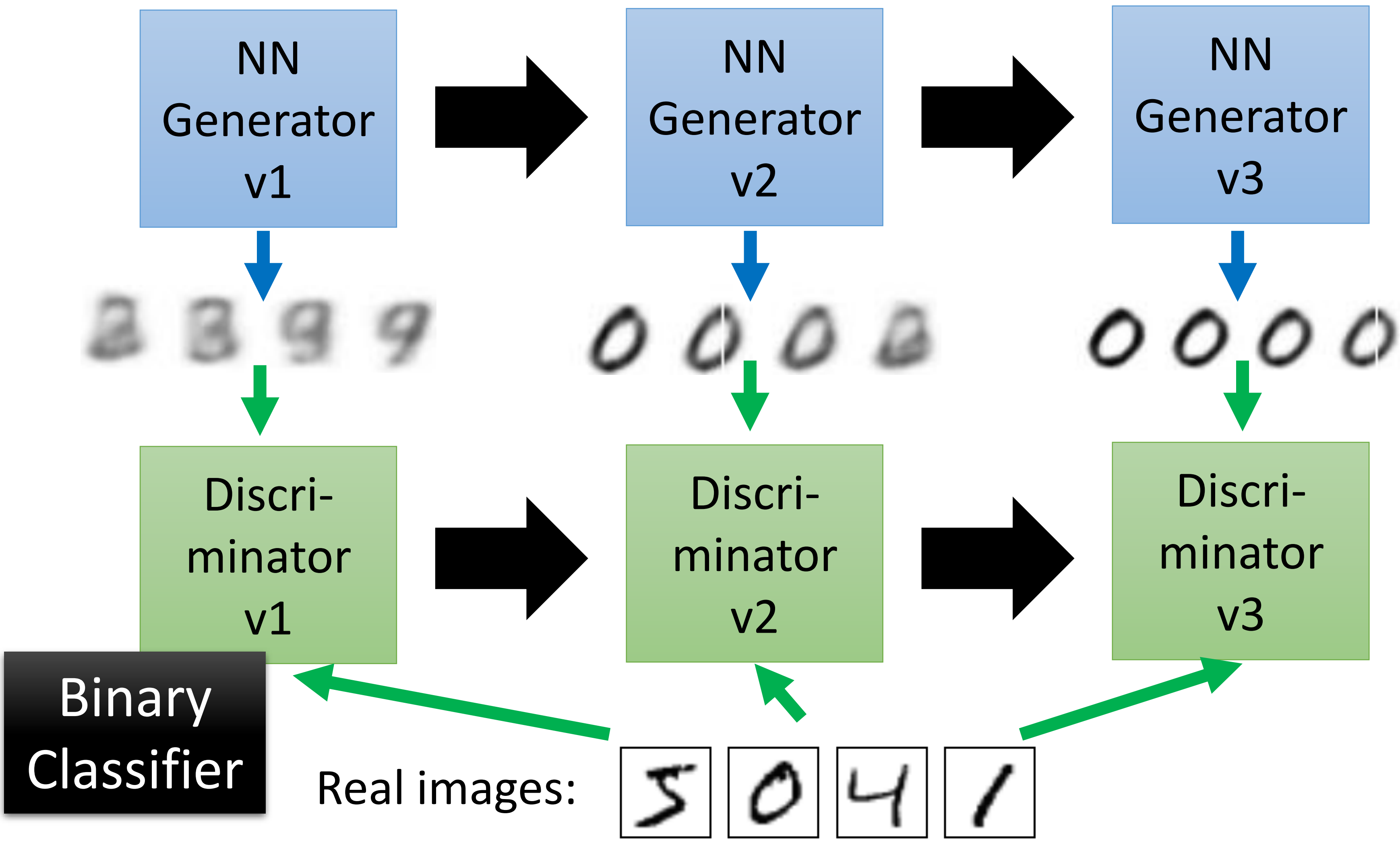
葉脈

蝴蝶不是棕色

蝴蝶沒有葉脈

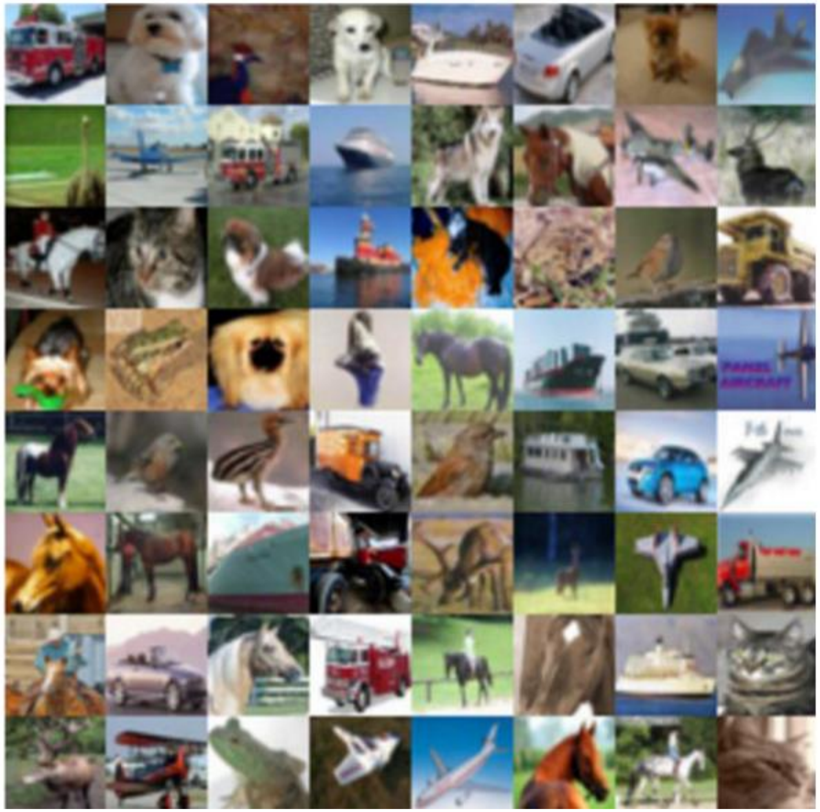
....

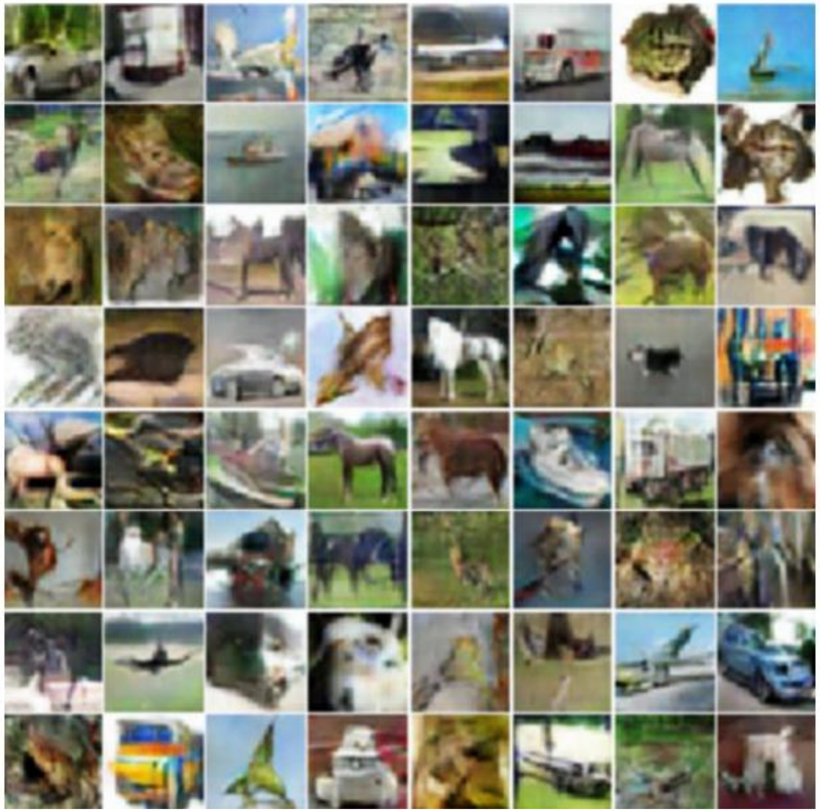
The evolution of generation



Cifar-10

- Which one is machine-generated?

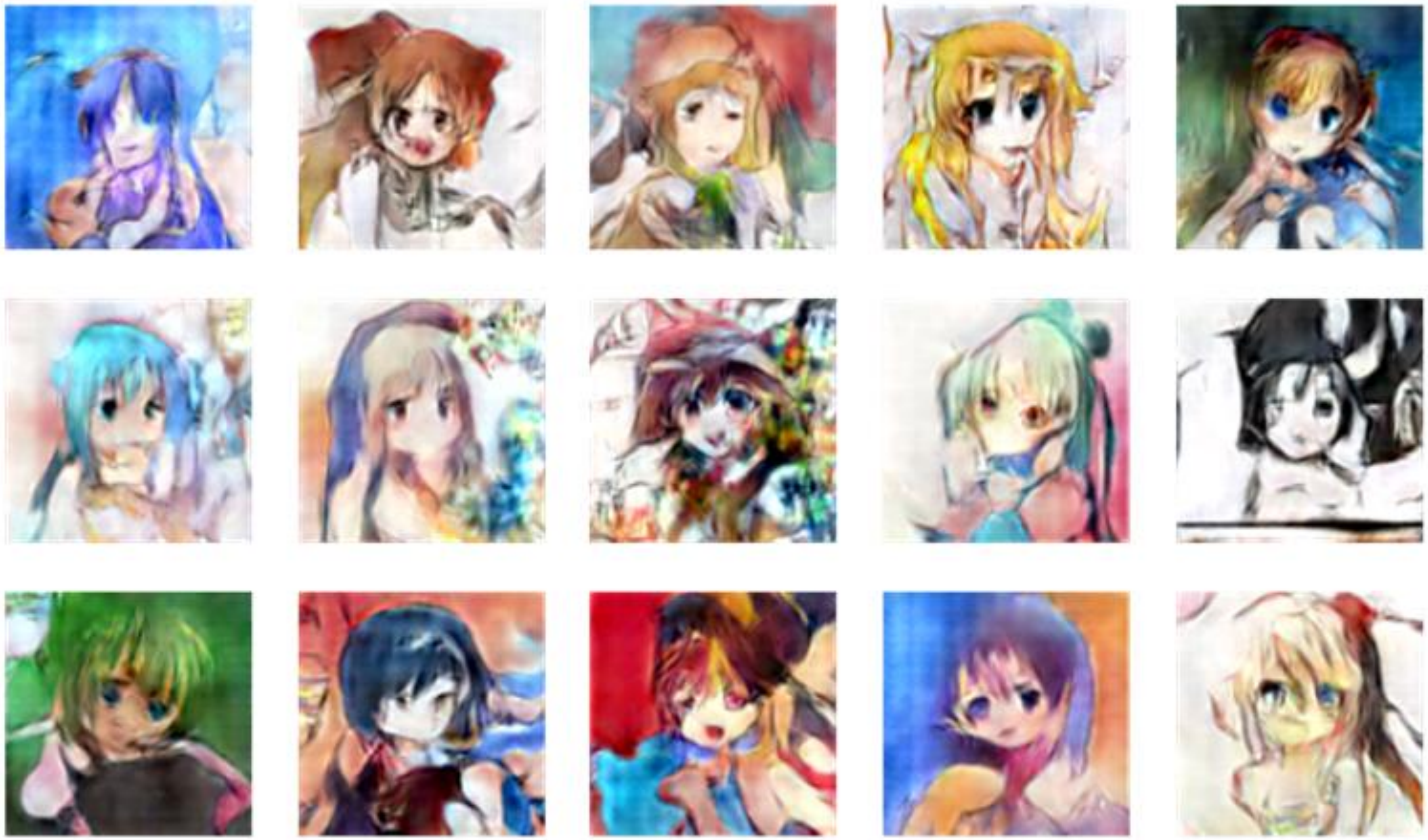




Ref: https://openai.com/blog/generative-models/

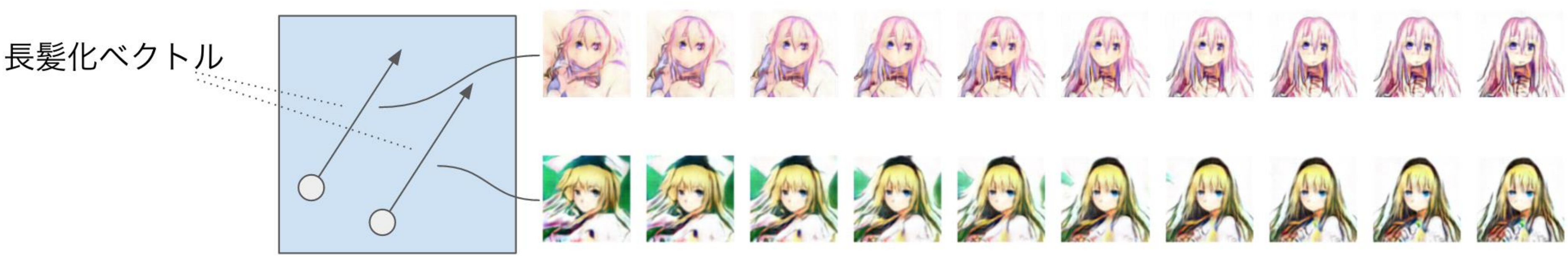
畫漫畫

- 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

Want to practice Generation Models?

Pokémon Creation

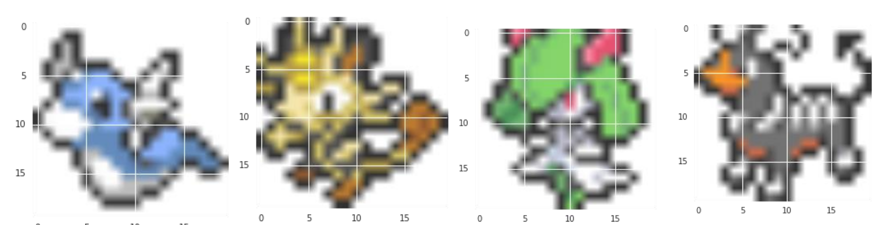
- Small images of 792 Pokémon's
  - Can machine learn to create new Pokémons?

<u>Don't catch them! Create them!</u>

- Source of image: http://bulbapedia.bulbagarden.net/wiki/List_of_Pok%C3%A9mon_by_base_stats_(Generation_VI)

Original image is 40 x 40

Making them into 20 x 20

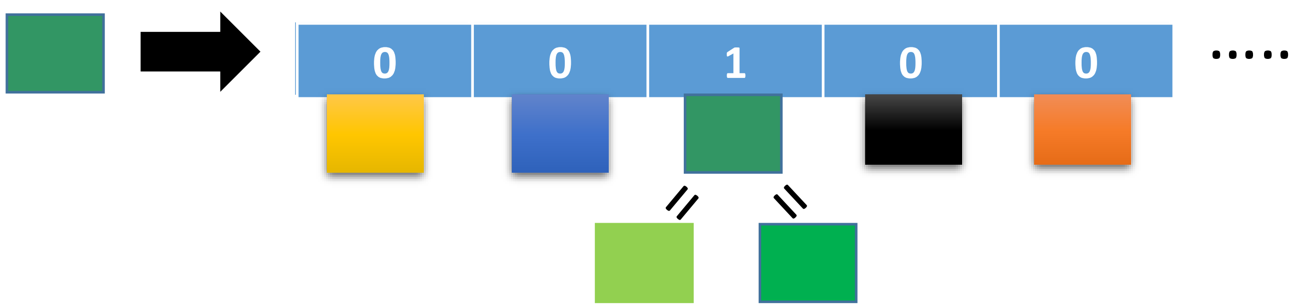


Pokémon Creation

- Each pixel is represented by 3 numbers (corresponding to RGB)

R=50, G=150, B=100

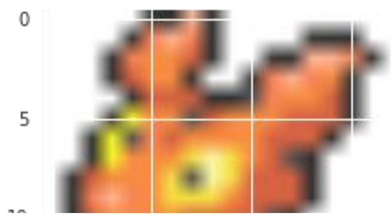
- Each pixel is represented by a 1-of-N encoding feature

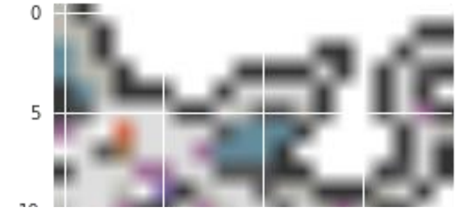


Clustering the similar color → 167 colors in total

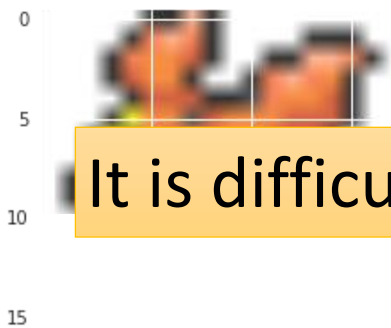
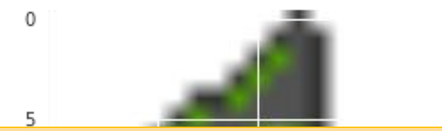
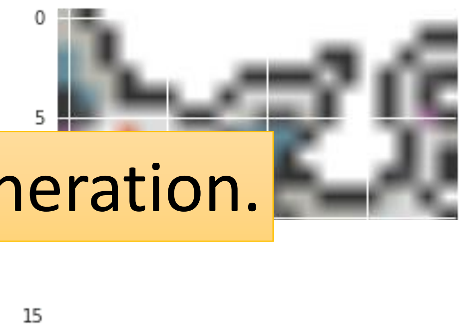
Real Pokémon

Never seen by machine!



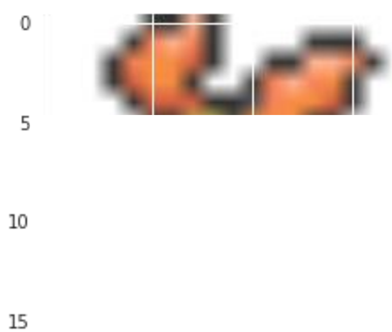


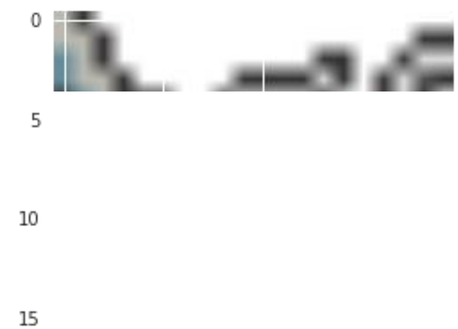
Cover 50%

It is difficult to evaluate generation.

Cover 75%

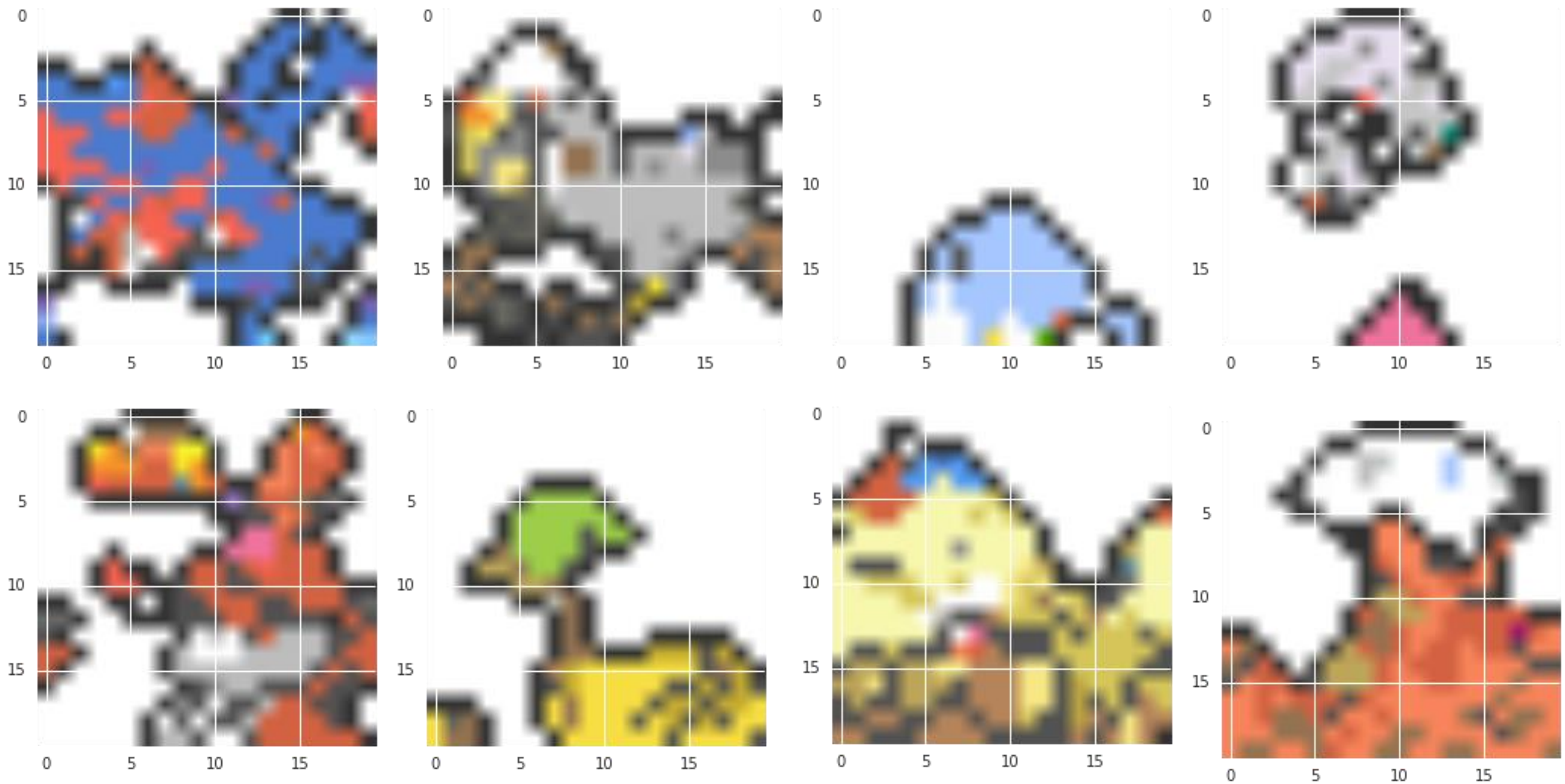




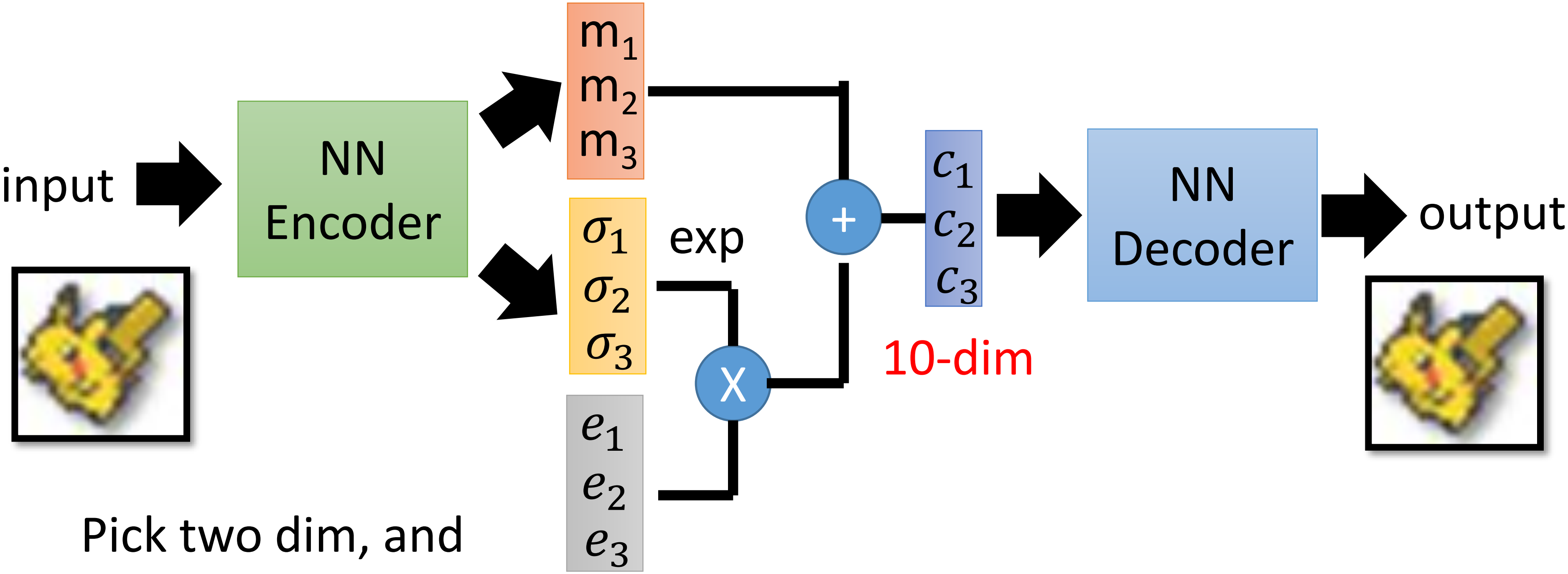
Pokémon Creation

Drawing from scratch

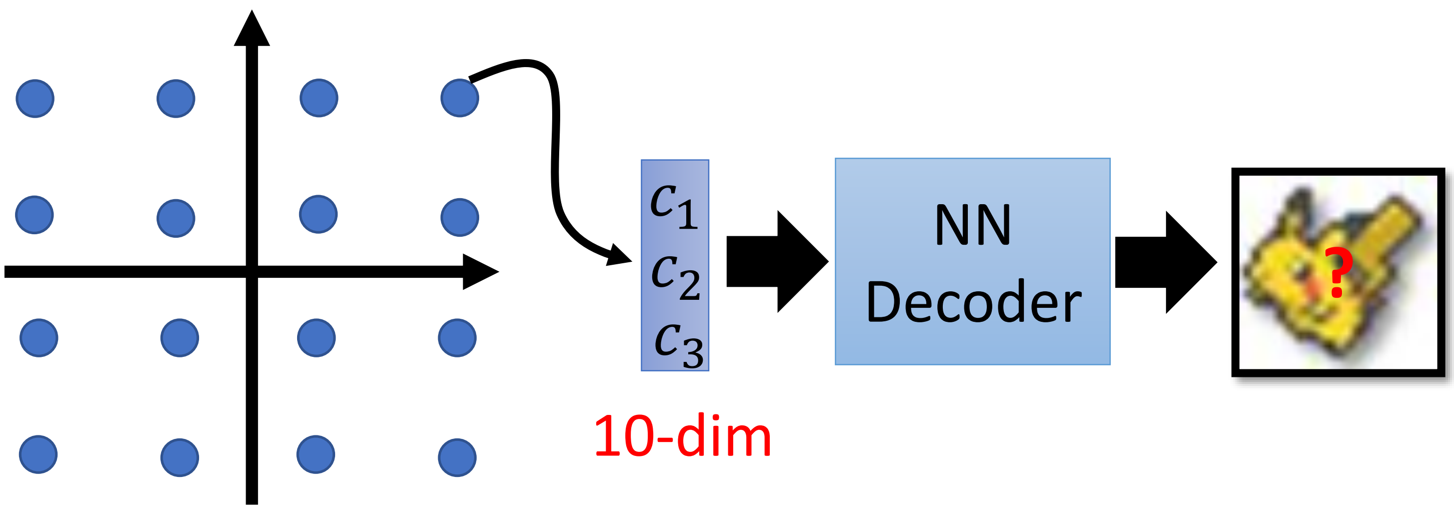
Need some randomness

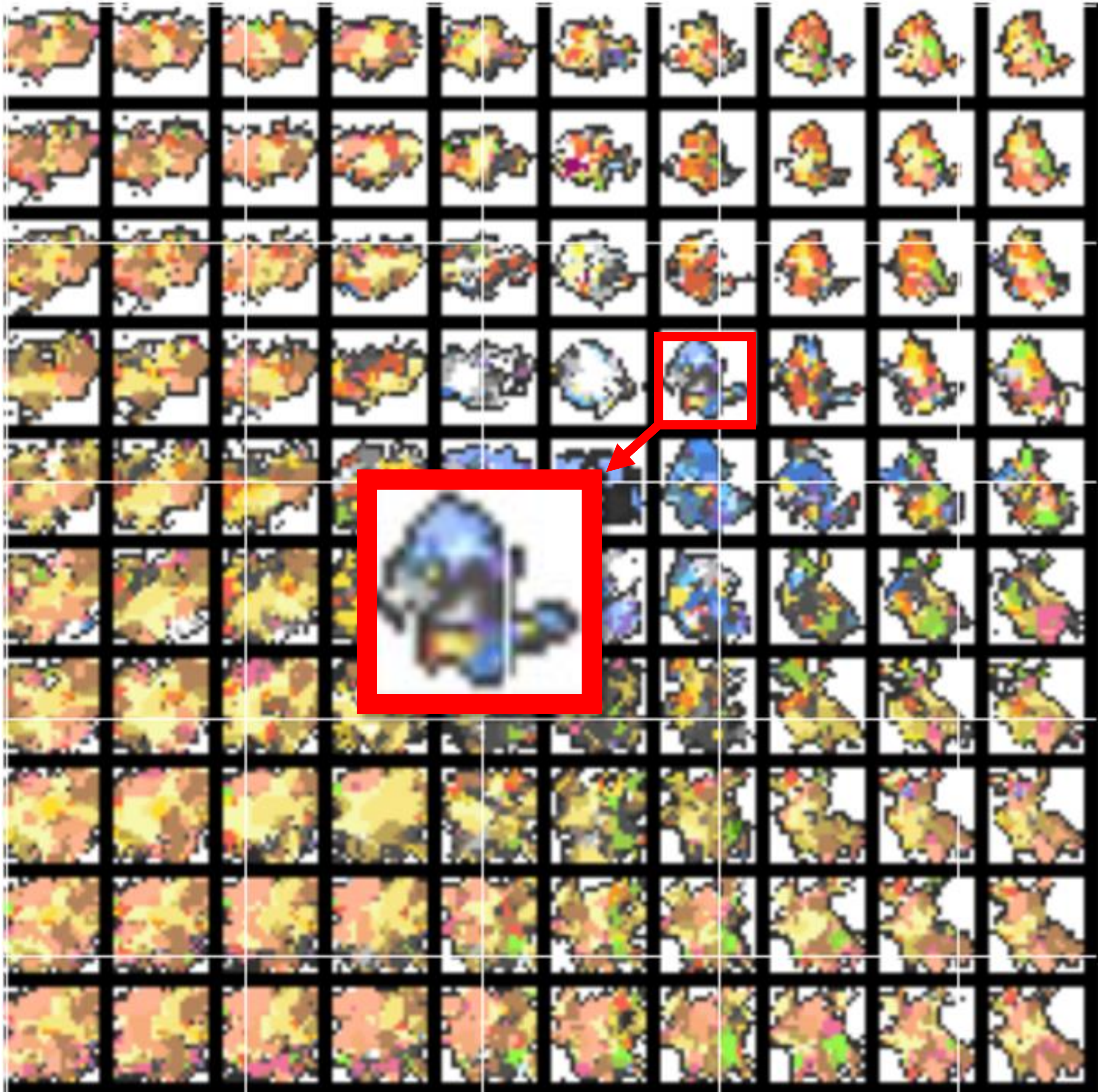


Pokémon Creation



Pick two dim, and
fix the rest eight





Pokémon Creation - Data

- Original image (40 x 40): http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML_2016/Pokemon_creation/image.rar
- Pixels (20 x 20): http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML_2016/Pokemon_creation/pixel_color.txt
  - Each line corresponds to an image, and each number corresponds to a pixel
    - http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML_2016/Pokemon_creation/colormap.txt

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 19 41 34 0 0 19 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 44 74 44 51 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 21 80 80 81 0 0 0 0 0 0 0 0
0 0 0 0 0 1 2 3 18 35 22 0 5 2 0 0 0 0 0 0
93 94 93 93 85 95 38 96 97 98 99 99 67 99 9
0 0 0 0 0 0 1 106 106 106 106 106 61 107 0
```

......

0

1

2

```
255 255 255
53 53 53
49 49 49
186 186 186
51 51 51
54 54 54
187 187 187
83 83 83
50 51 52
251 251 251
52 52 52
```

⋮

You can use the data without permission

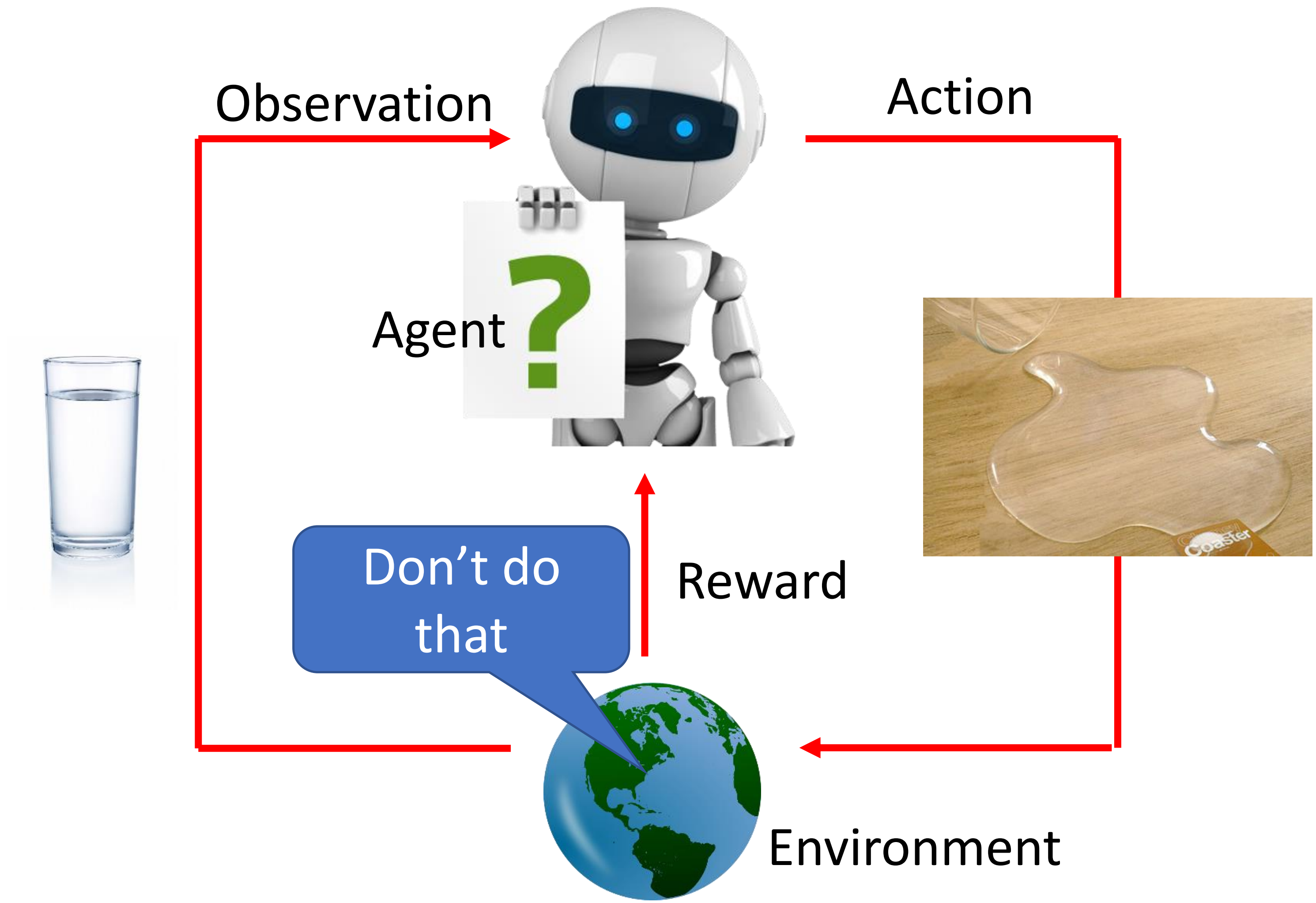
Outline

Unsupervised Learning

- 化繁為簡
  - Example: Word Vector and Audio Word Vector
- 無中生有

Reinforcement Learning

Scenario of Reinforcement Learning



Scenario of Reinforcement Learning

Agent learns to take actions to maximize expected reward.

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

Supervised v.s. Reinforcement

- Supervised

Learning from teacher

"Hello" — Say "Hi"

"Bye bye" — Say "Good bye"

- Reinforcement

……. ……. ……

Bad

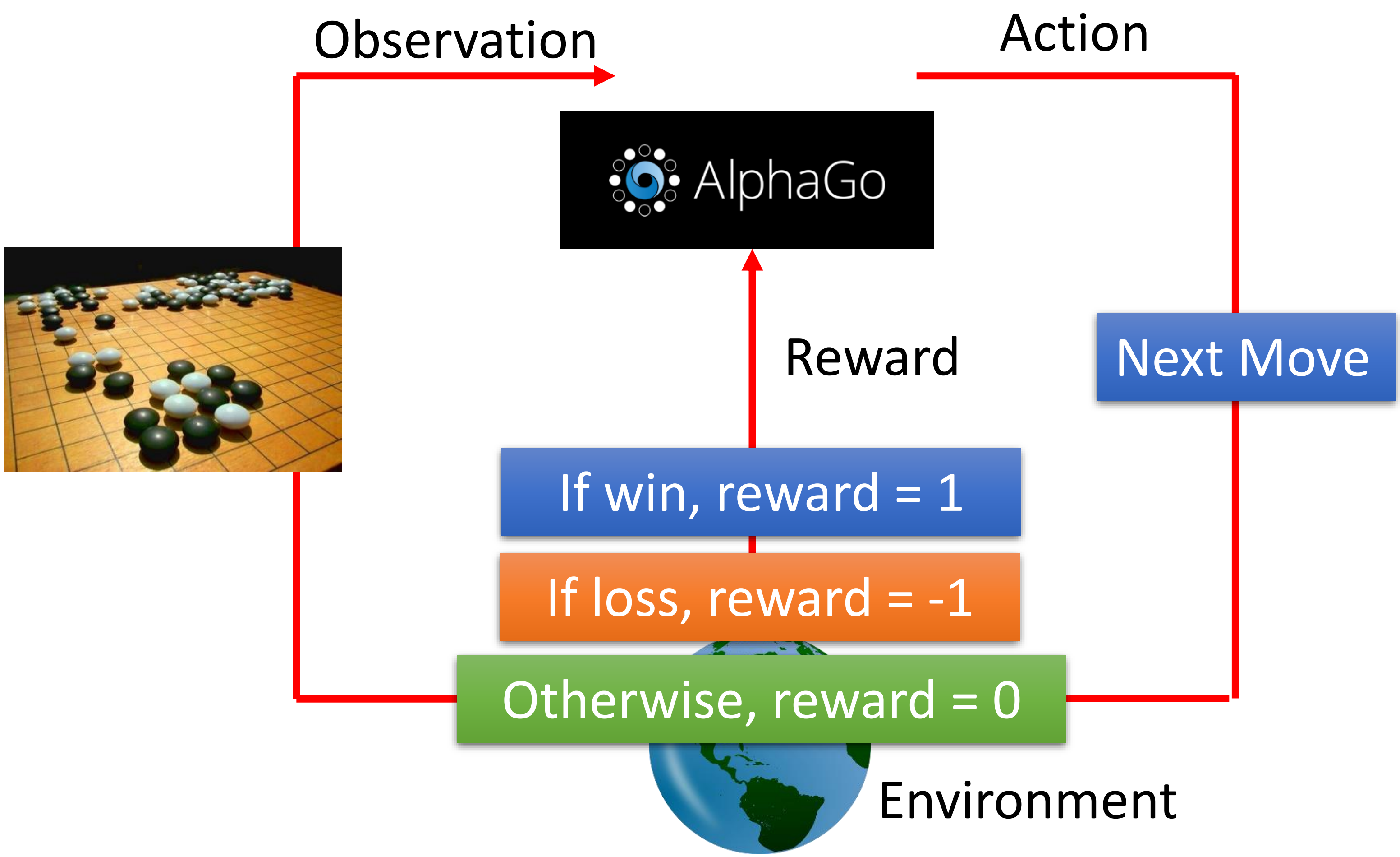
Hello ☺ ……

Agent Agent

Learning from critics

Scenario of Reinforcement Learning

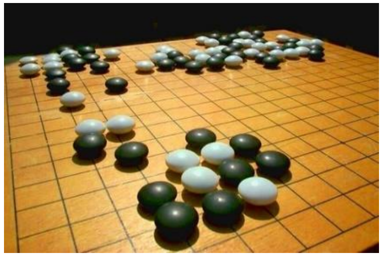
Agent learns to take actions to maximize expected reward.



Supervised v.s. Reinforcement

- Supervised:

Next move: "5-5"



Next move: "3-3"

- Reinforcement Learning

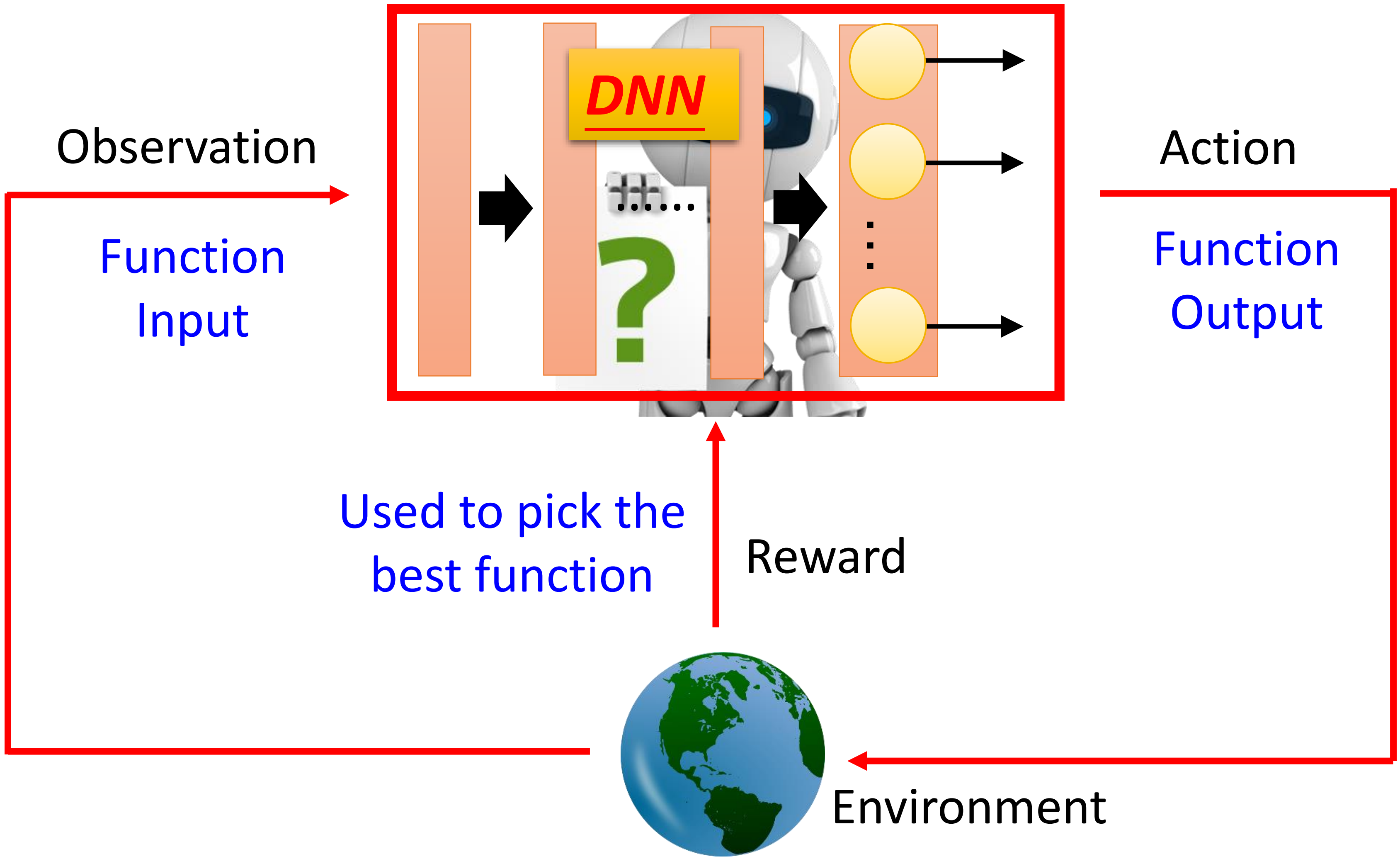
First move → ...... many moves ...... → Win!

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



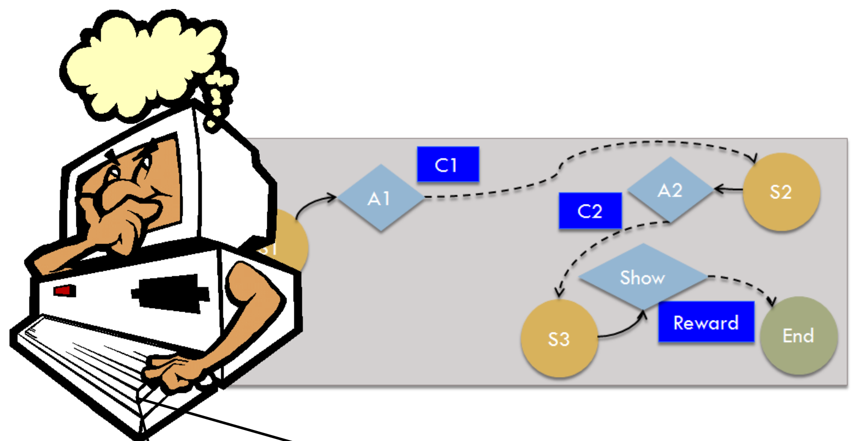
Application: Interactive Retrieval

- Interactive retrieval is helpful.

[Wu & Lee, INTERSPEECH 16]

"Deep Learning"

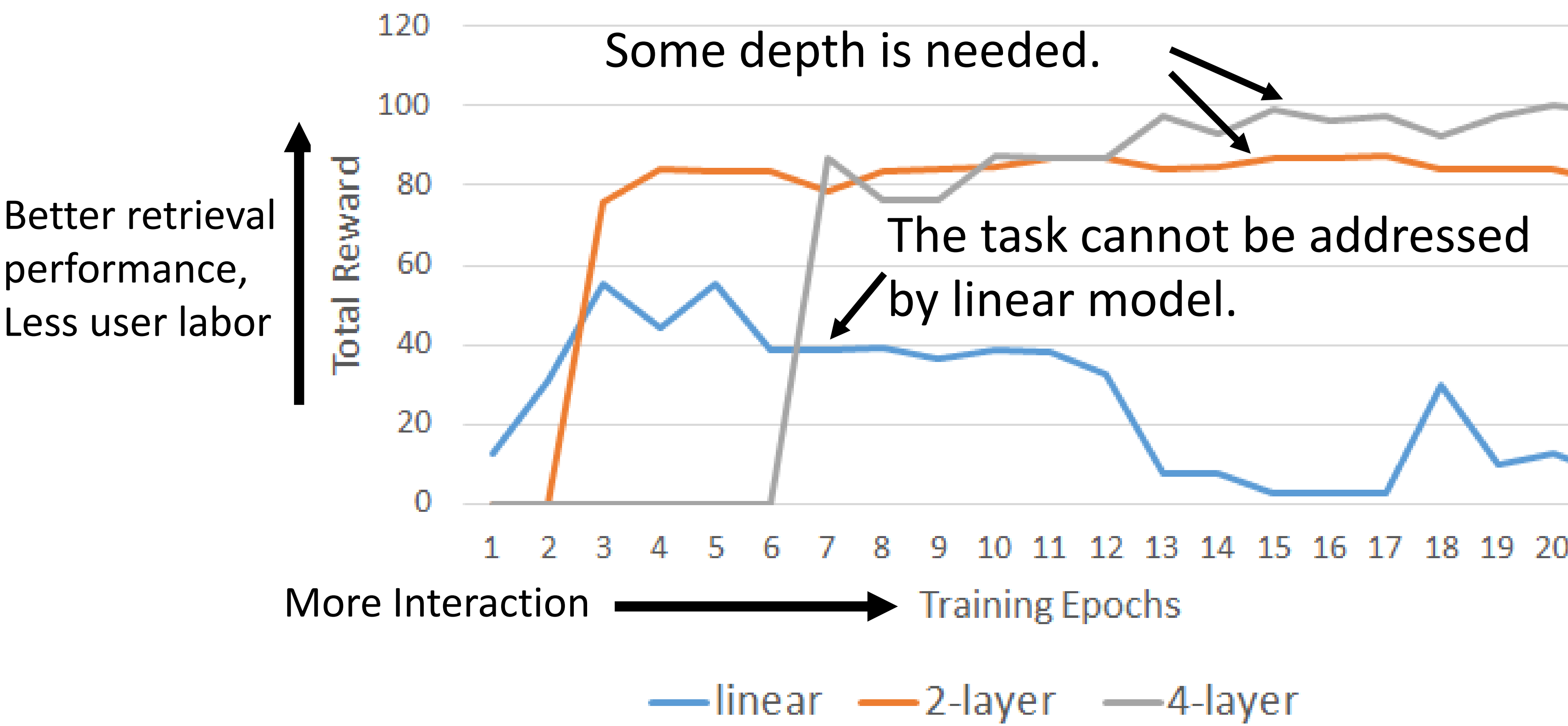
user



"Deep Learning" related to Machine Learning?
"Deep Learning" related to Education?

Deep Reinforcement Learning

- Different network depth



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/

Conclusion

如何成為武林高手

- 內外兼修
  - 內功充沛，恃強克弱
  - 招數精妙，以快打慢
- Deep Learning 也需要內外兼修
  - 內力：運算資源
  - 招數：各種技巧
- 內力充沛，平常的招式也有可能發會巨大的威力
- 只有內力、沒有招數
  - WavNet 並不是只憑蠻力

希望大家都可以成為內外兼修的高手