

Deep Learning Tutorial

李宏毅

Hung-yi Lee

Outline

Lecture I: Introduction of Deep Learning



Lecture II: Variants of Neural Network



Lecture III: Beyond Supervised Learning

Lecture I: Introduction of Deep Learning

Outline

Introduction of Deep Learning

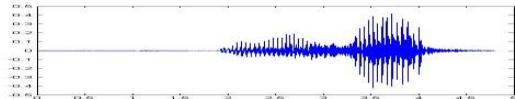
“Hello World” for Deep Learning

Tips for Deep Learning

Machine Learning ≈ Looking for a Function

- Speech Recognition

$$f($$



) = “How are you”

- Image Recognition

$$f($$



) = “Cat”

- Playing Go

$$f($$



) = “5-5”
(next move)

- Dialogue System

$$f($$

“Hi”

(what the user said)

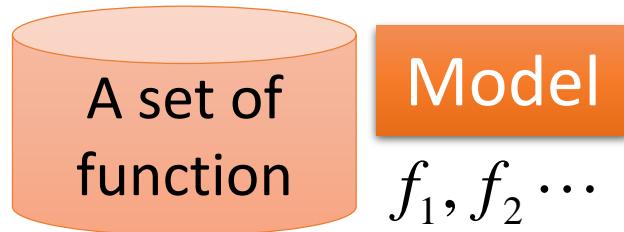
) = “Hello”

(system response)

Framework

Image Recognition:

$$f(\text{}) = \text{"cat"}$$



$$f_1(\text{}) = \text{"cat"} \quad f_2(\text{}) = \text{"money"}$$

$$f_1(\text{}) = \text{"dog"} \quad f_2(\text{}) = \text{"snake"}$$

Framework

A set of function

Model
 $f_1, f_2 \dots$

Goodness of function f

Training Data

Image Recognition:

$f($  $) = \text{"cat"}$

$f_1($  $) = \text{"cat"}$ $f_2($  $) = \text{"money"}$

$f_1($  $) = \text{"dog"}$ $f_2($  $) = \text{"snake"}$

Better!

Supervised Learning

function input:

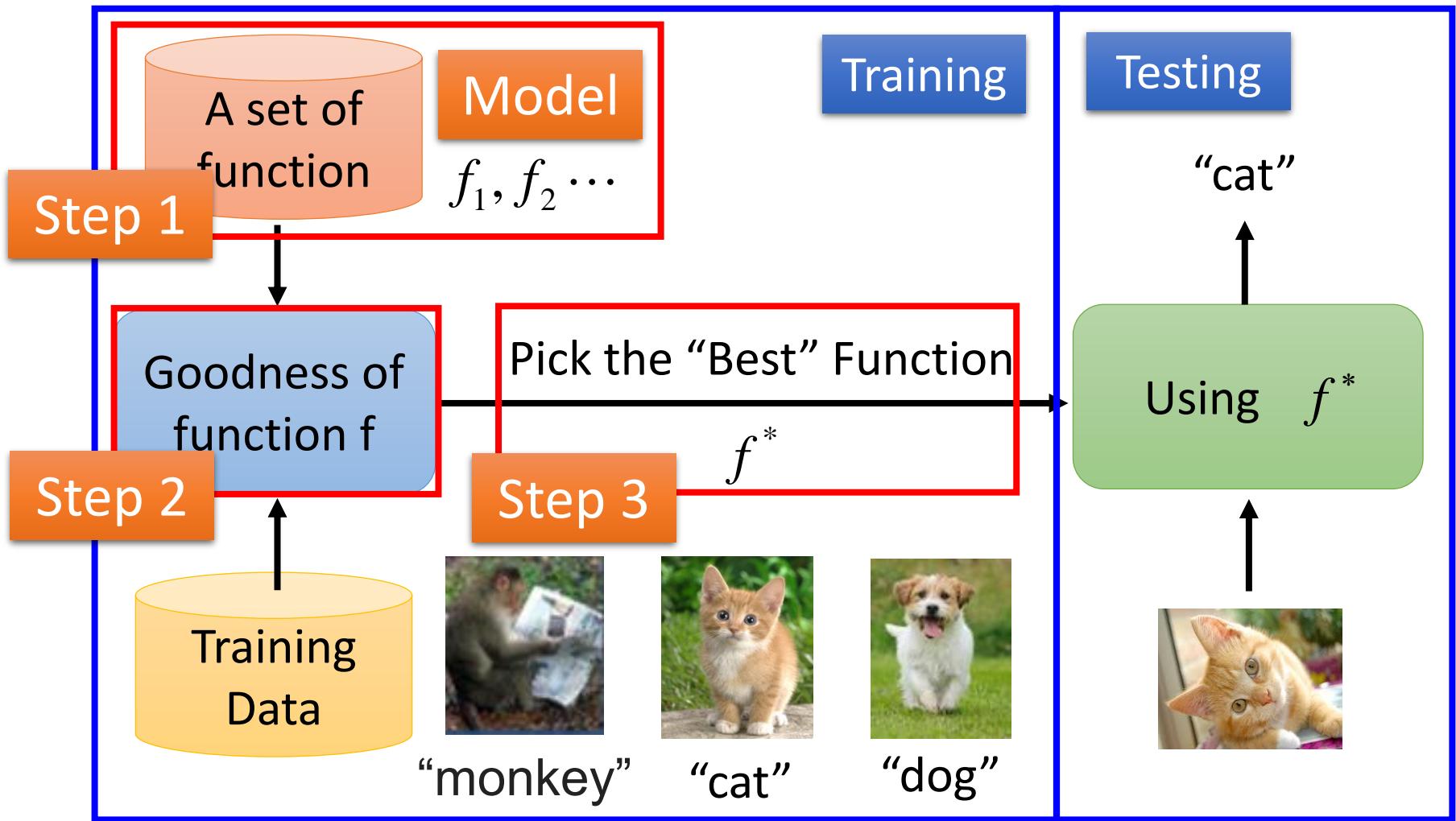


function output: "monkey" "cat" "dog"

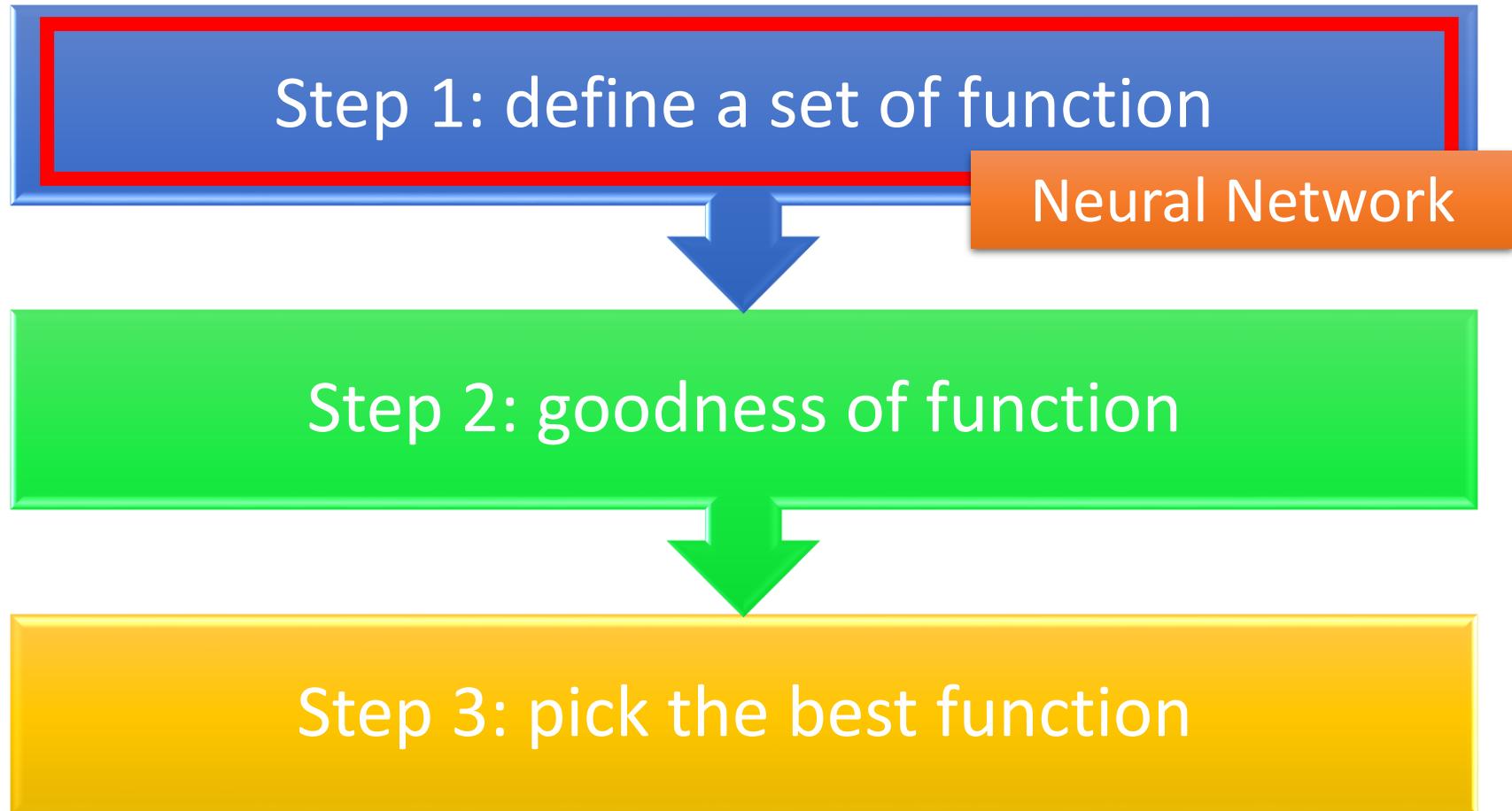
Framework

Image Recognition:

$$f(\text{cat image}) = \text{"cat"}$$



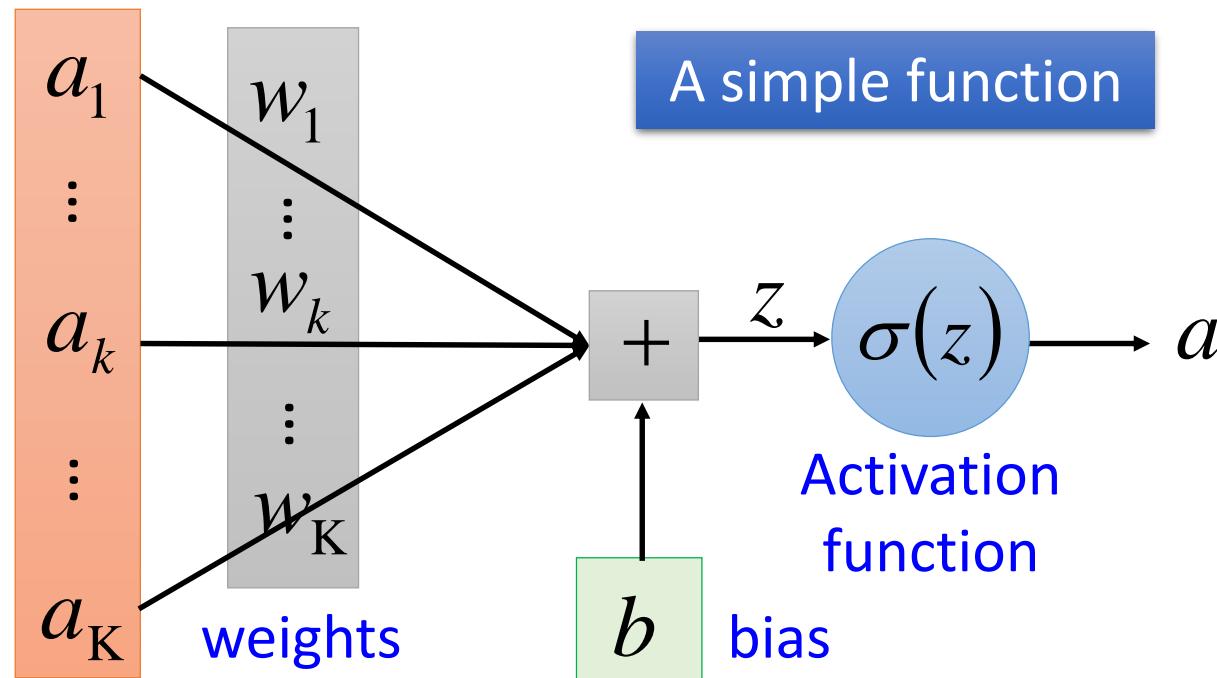
Three Steps for Deep Learning



Neural Network

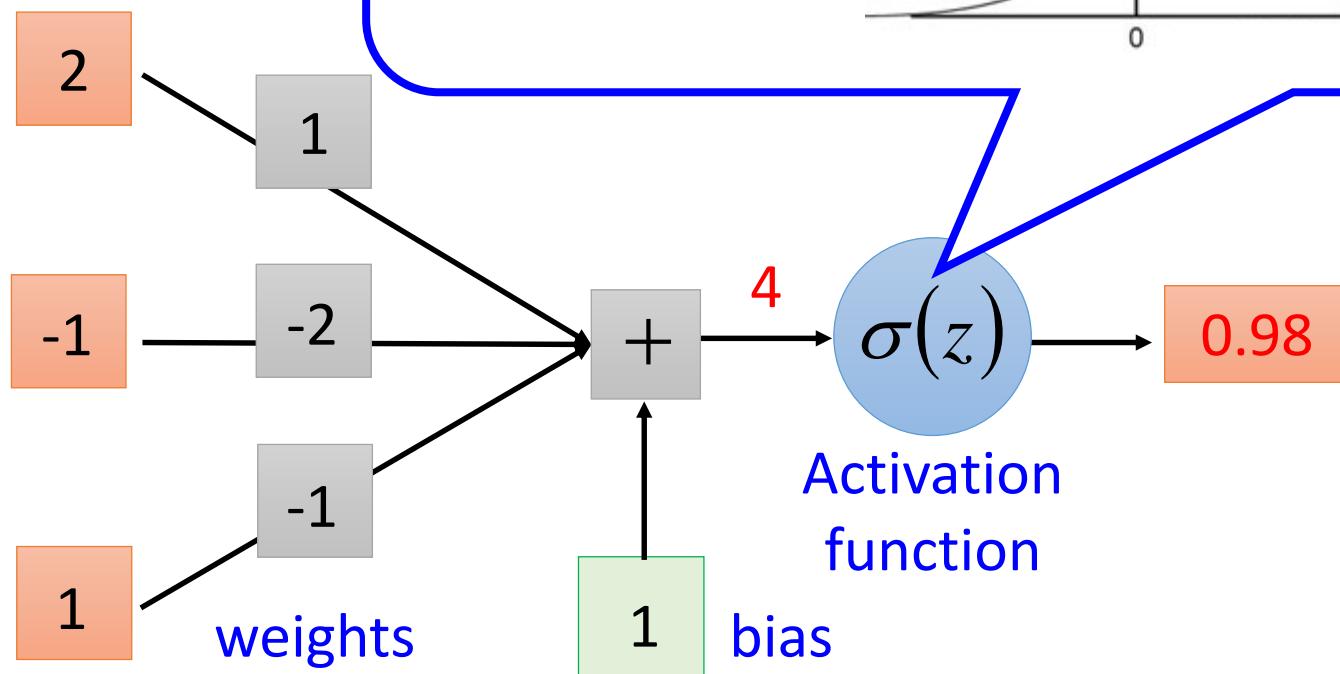
Neuron

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



Neural Network

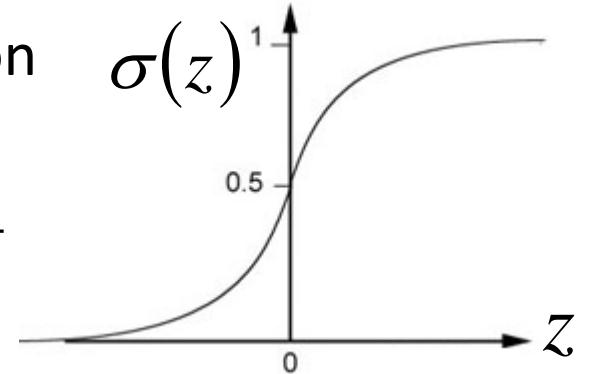
Neuron



Sigmoid Function

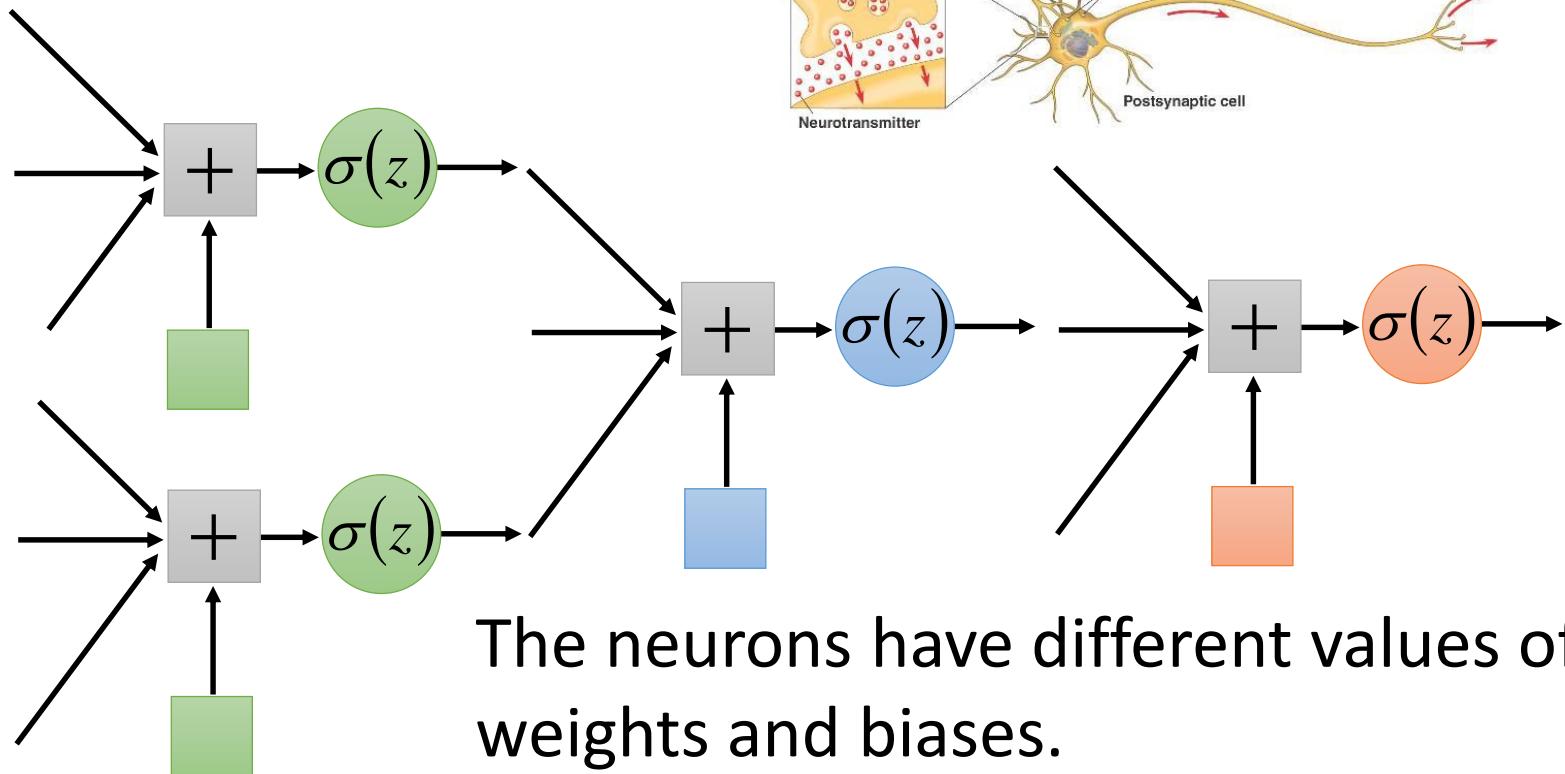
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

$$\sigma(z)$$



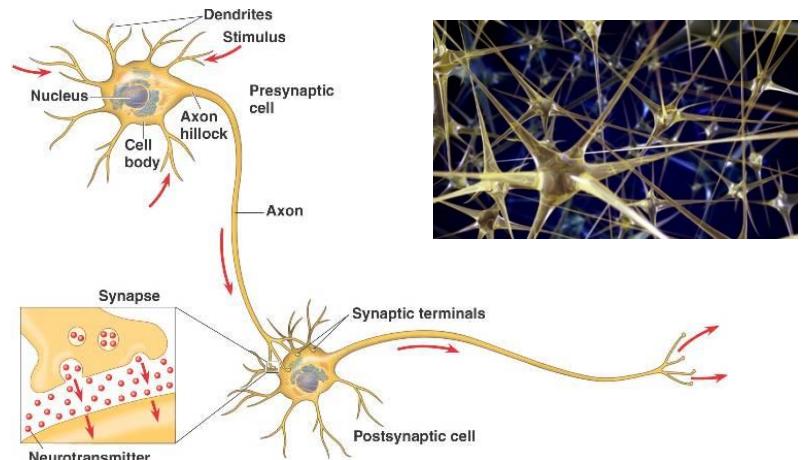
Neural Network

Different connections lead to different network structures

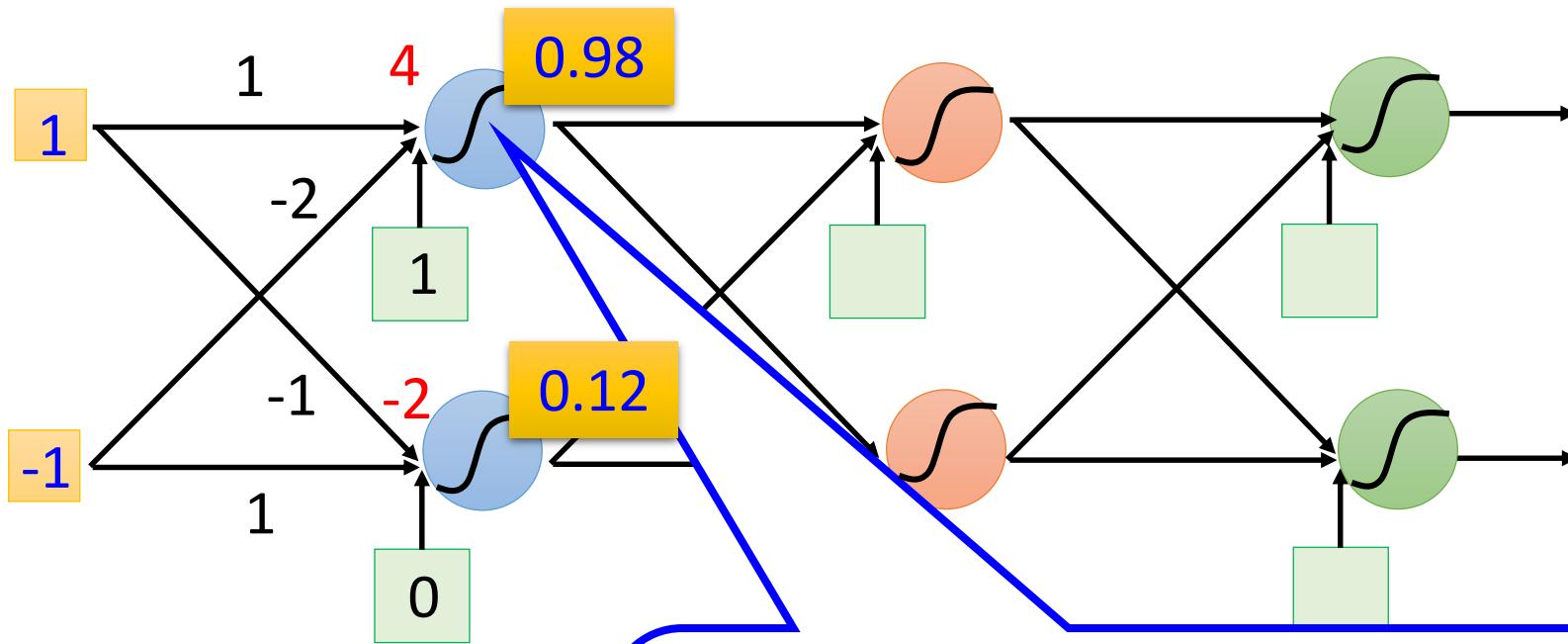


The neurons have different values of weights and biases.

Weights and biases are network parameters θ

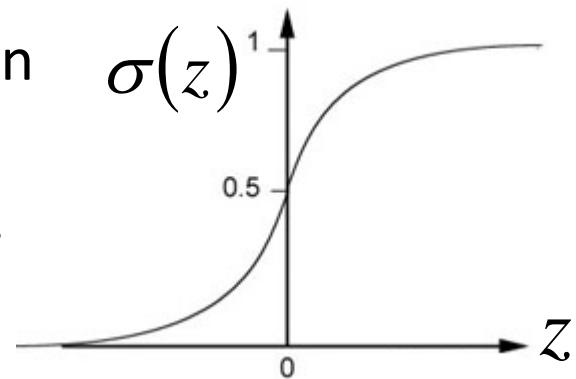


Fully Connect Feedforward Network

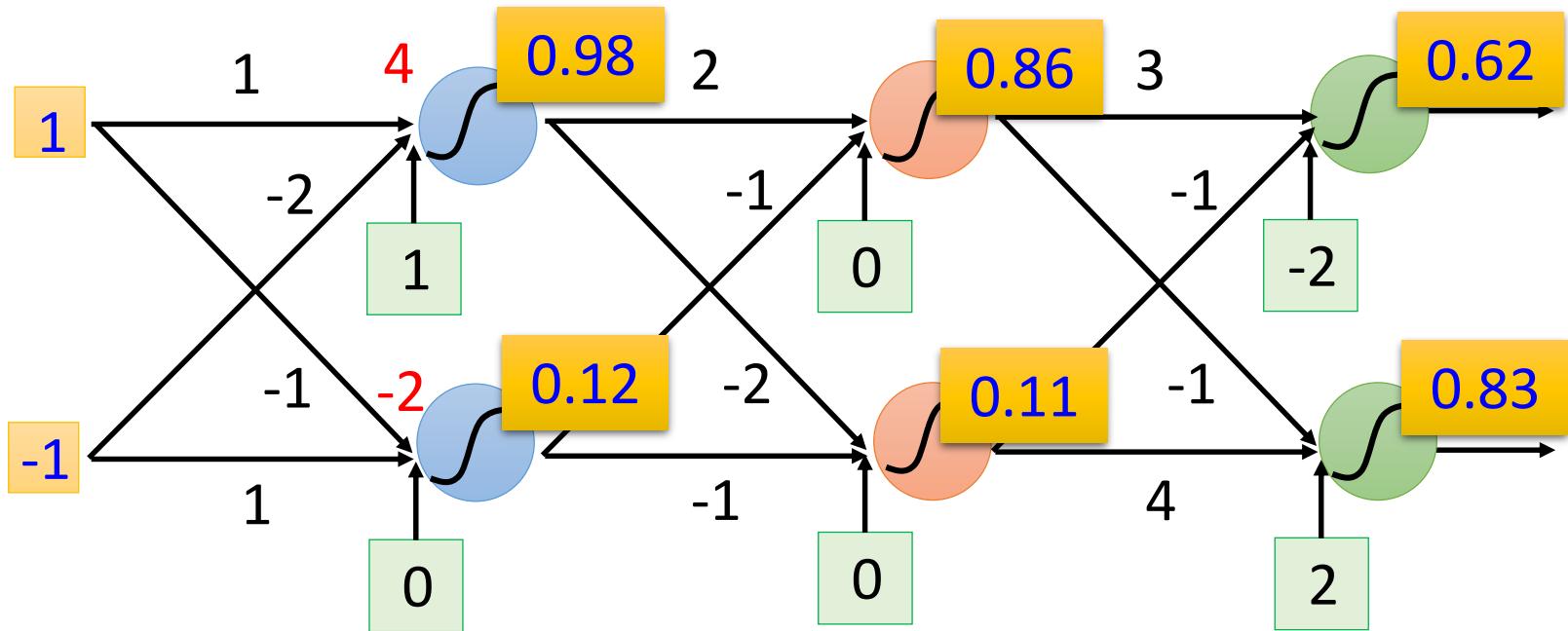


Sigmoid Function

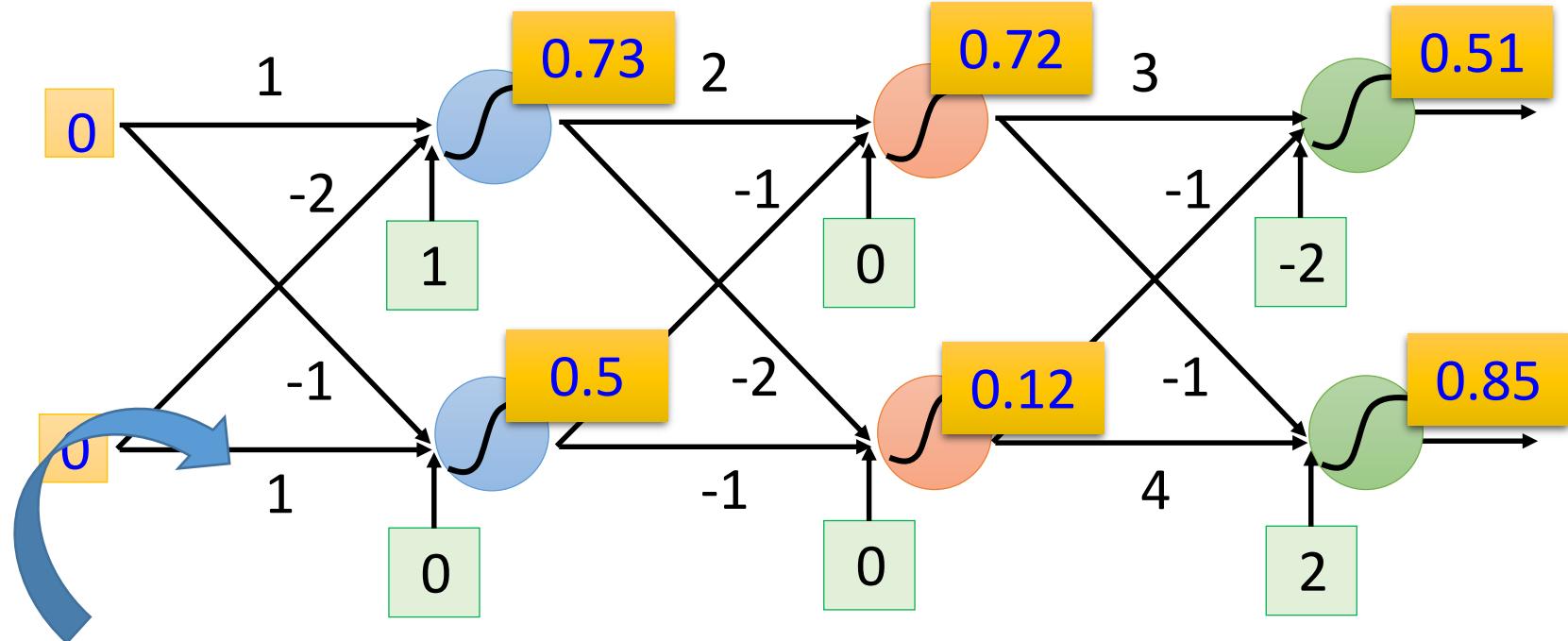
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$



Fully Connect Feedforward Network



Fully Connect Feedforward Network



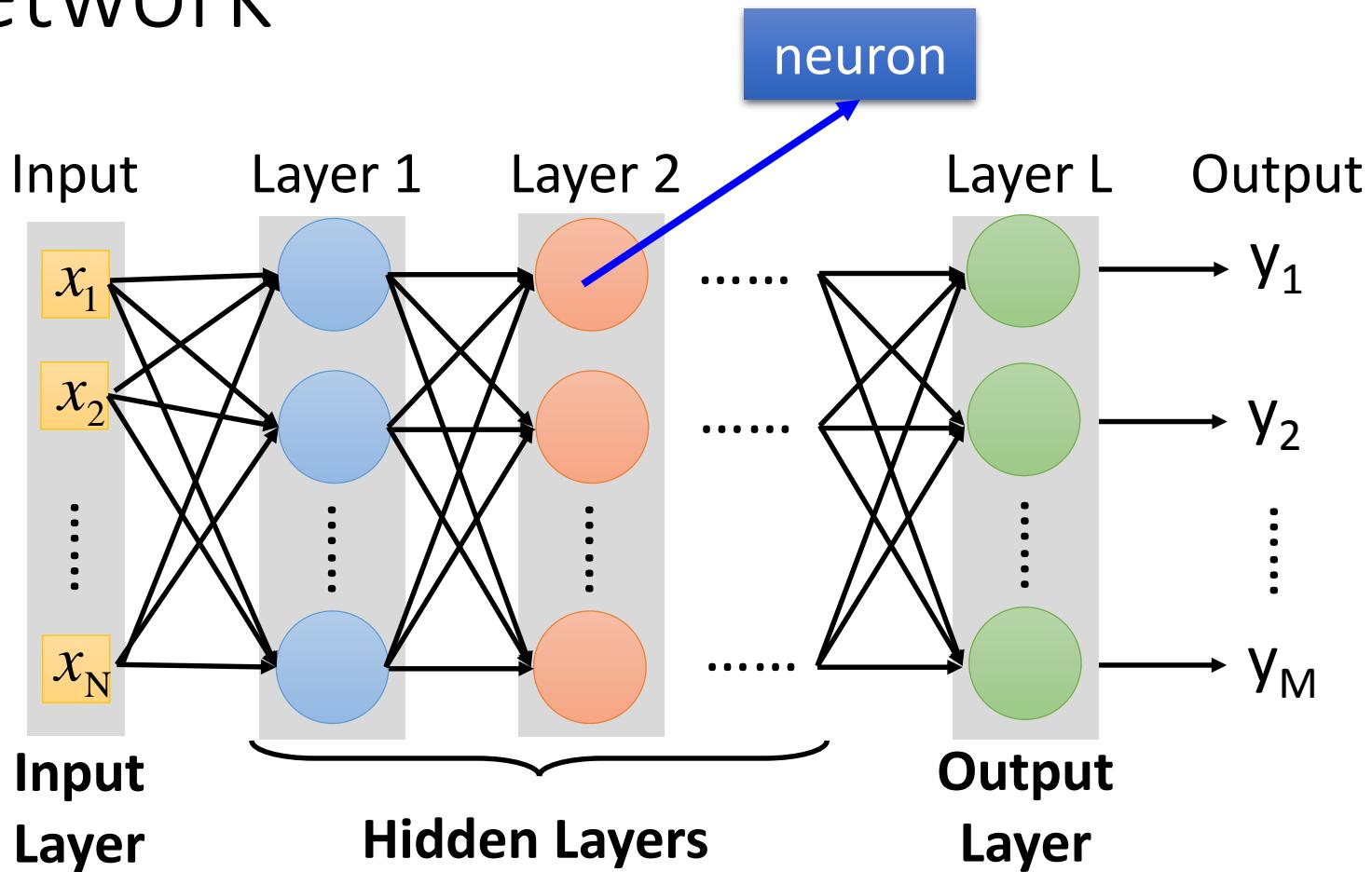
This is a function.
Input vector, output vector

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

Given parameters θ , define a function

Given network structure, define a function set

Fully Connect Feedforward Network



Deep means many hidden layers

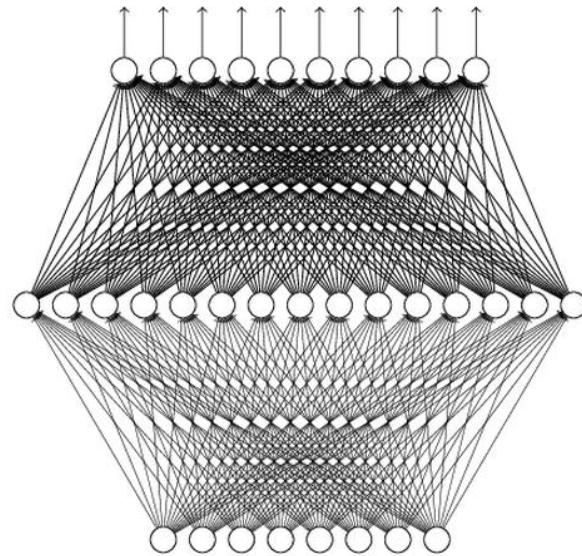
Why Deep? Universality Theorem

Any continuous function f

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

Can be realized by a network
with one hidden layer

(given **enough** hidden
neurons)



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

Why “Deep” neural network not “Fat” neural network?

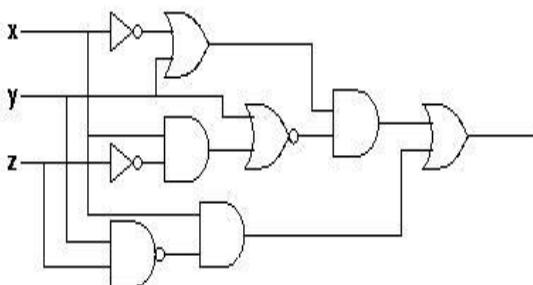
Why Deep? Analogy

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

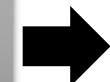


Neural network

- 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?

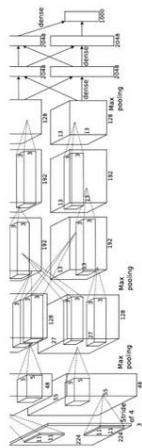
More reason:

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

Deep = Many hidden layers

http://cs231n.stanford.edu/slides/winter1516_lecuture8.pdf

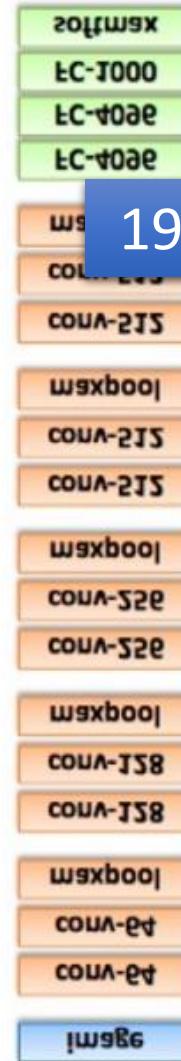
16.4%



AlexNet (2012)

8 layers

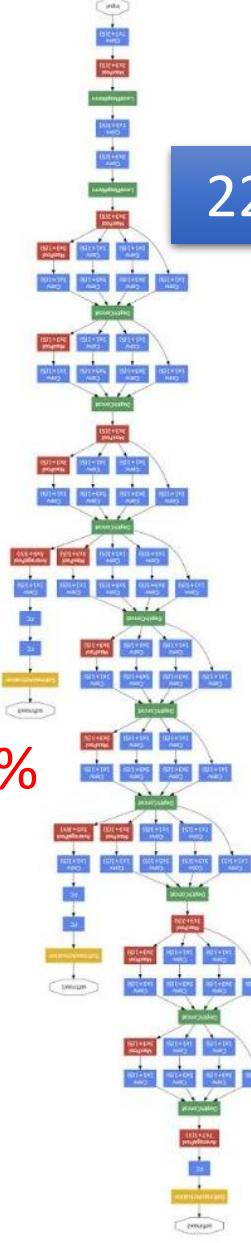
7.3%



VGG (2014)

19 layers

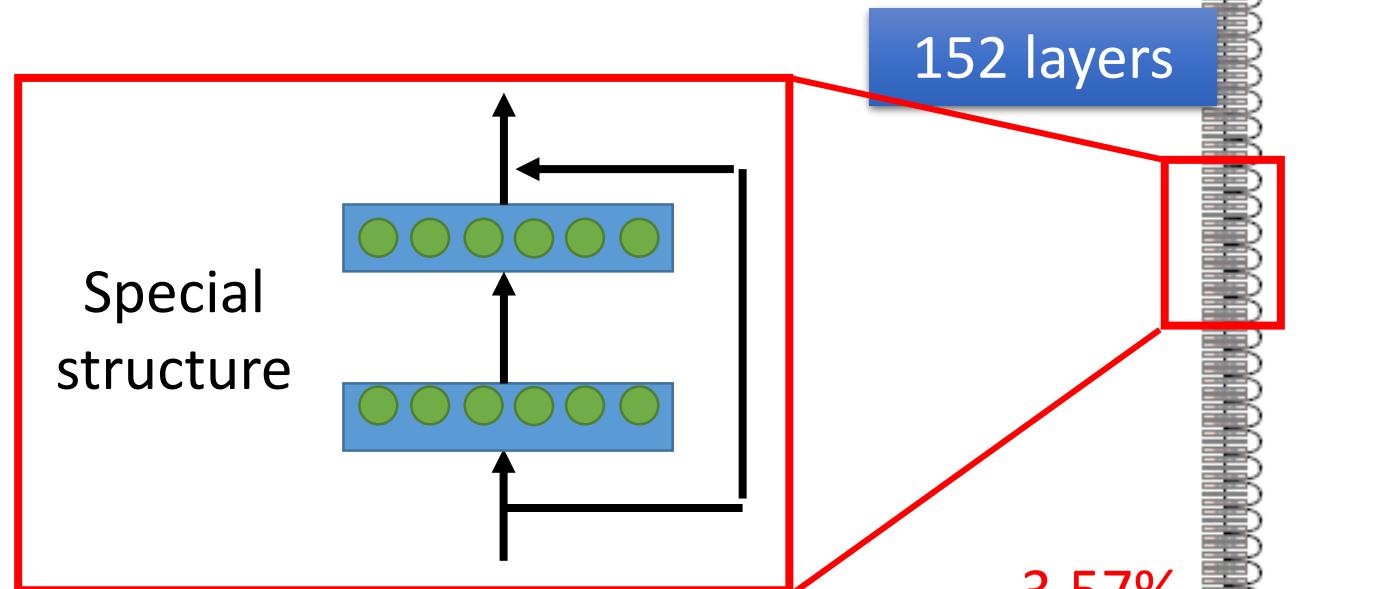
6.7%



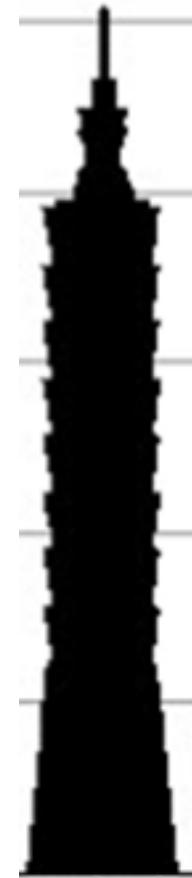
GoogleNet (2014)

22 layers

Deep = Many hidden layers



101 layers



Output Layer

- Softmax layer as the output layer

Ordinary Layer

$$z_1 \rightarrow \sigma \rightarrow y_1 = \sigma(z_1)$$

$$z_2 \rightarrow \sigma \rightarrow y_2 = \sigma(z_2)$$

$$z_3 \rightarrow \sigma \rightarrow y_3 = \sigma(z_3)$$

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

May not be easy to interpret

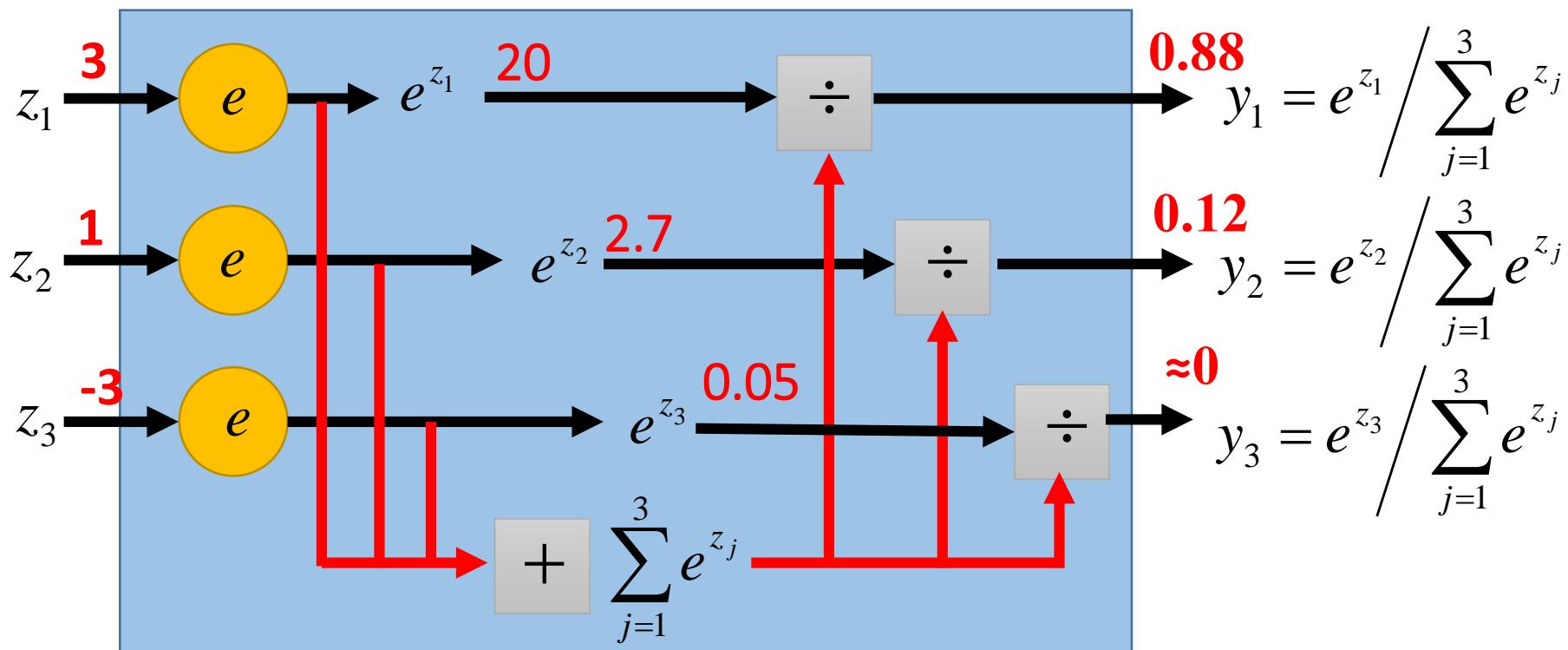
Output Layer

- Softmax layer as the output layer

Probability:

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

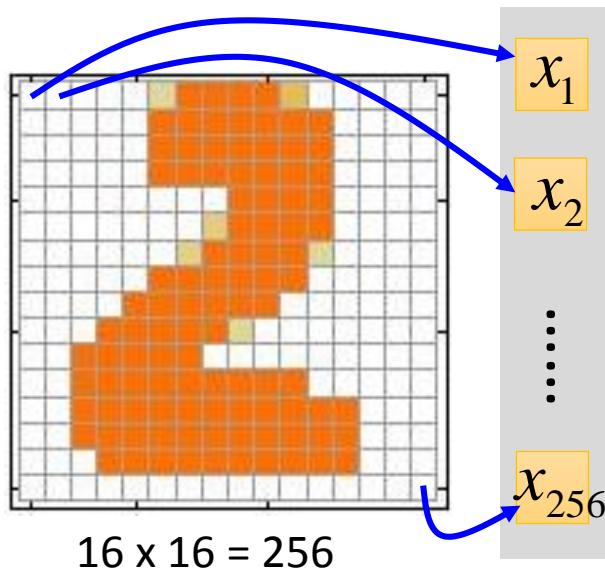
Softmax Layer



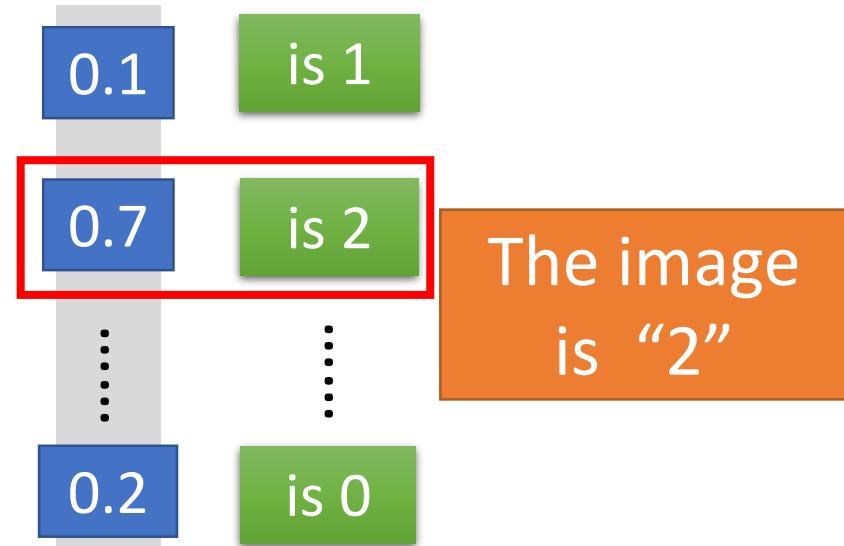
Example Application



Input



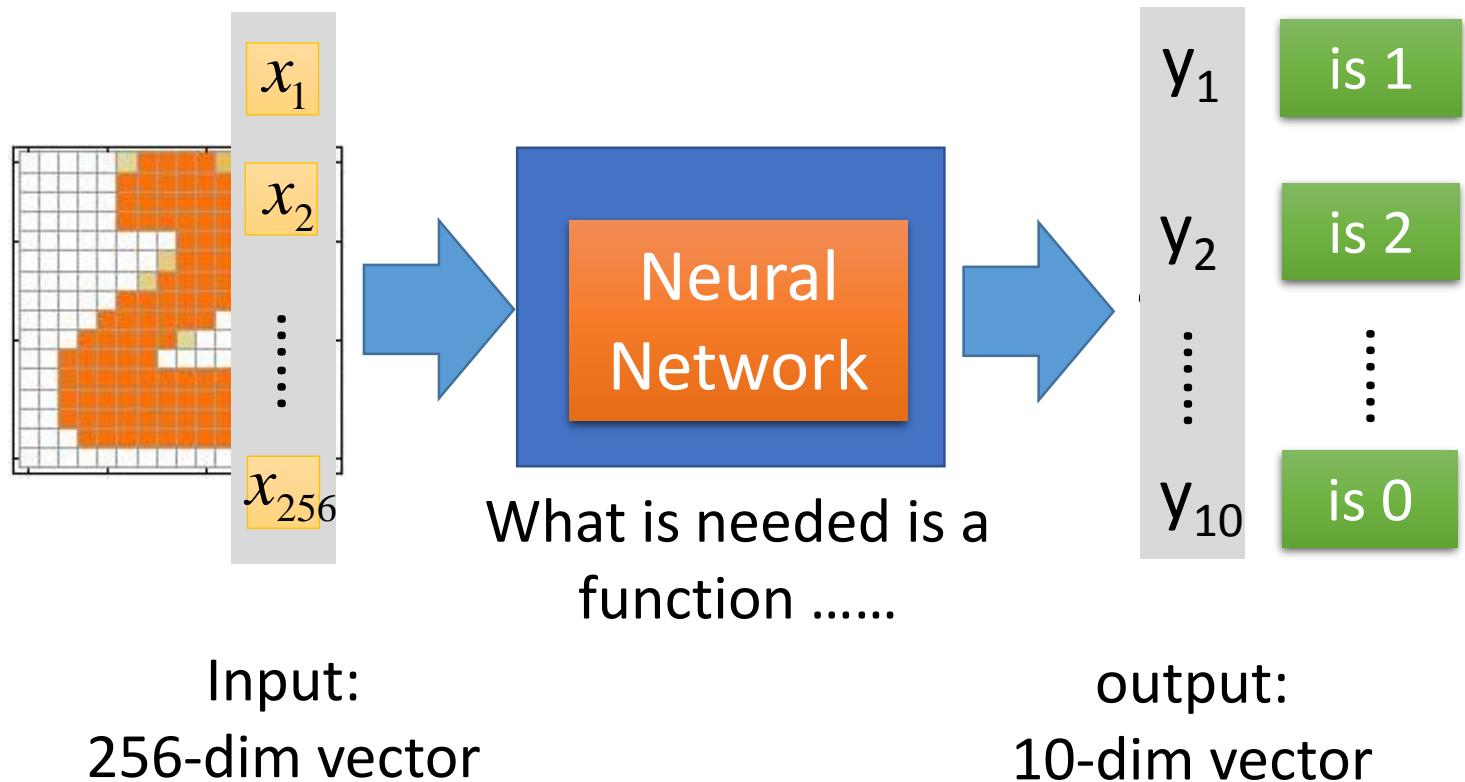
Output



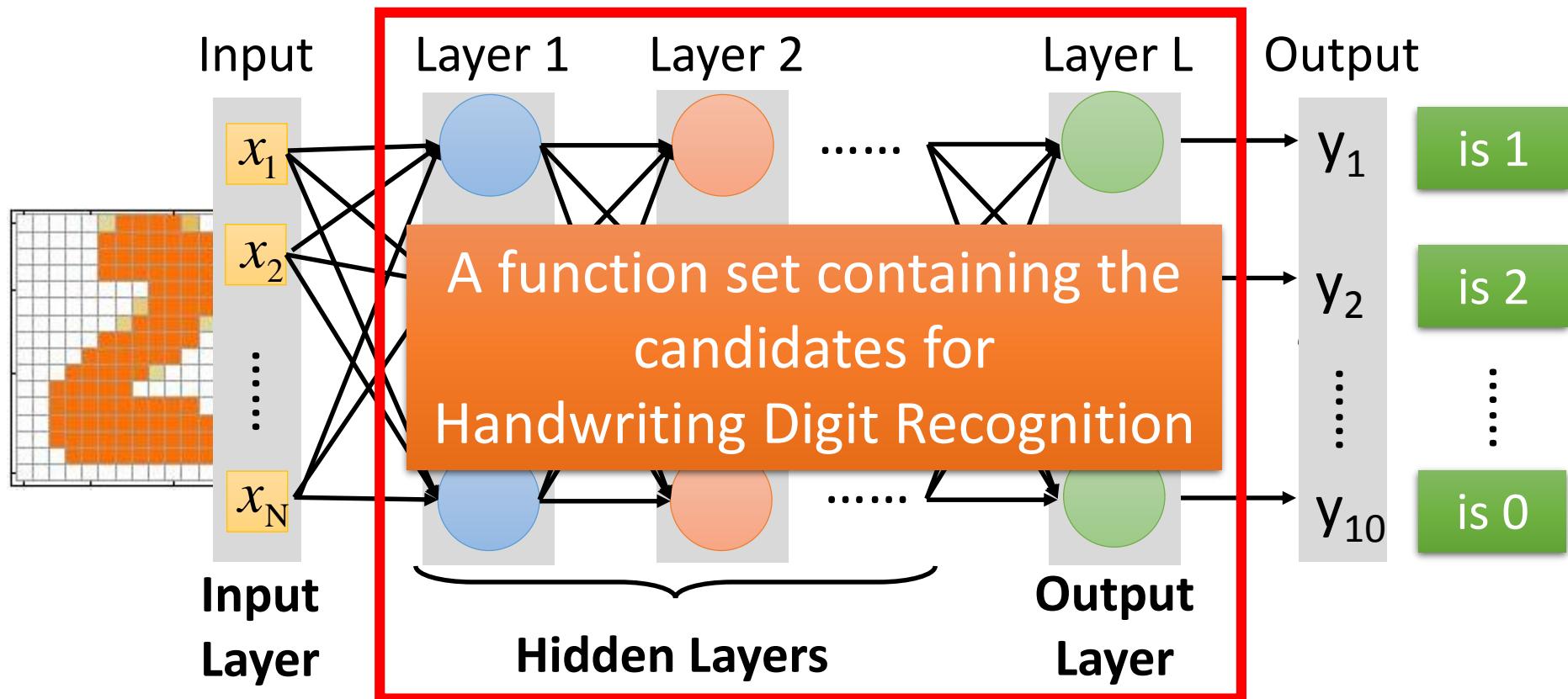
Each dimension represents the confidence of a digit.

Example Application

- Handwriting Digit Recognition

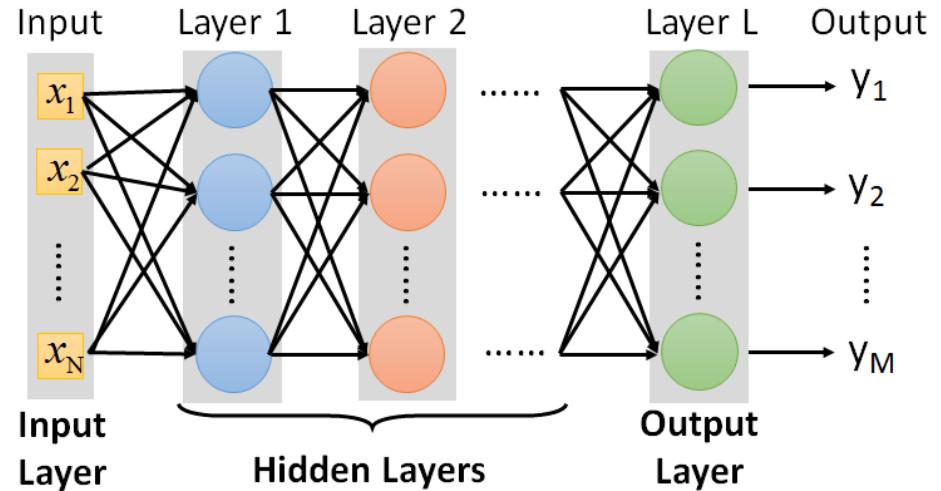


Example Application



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

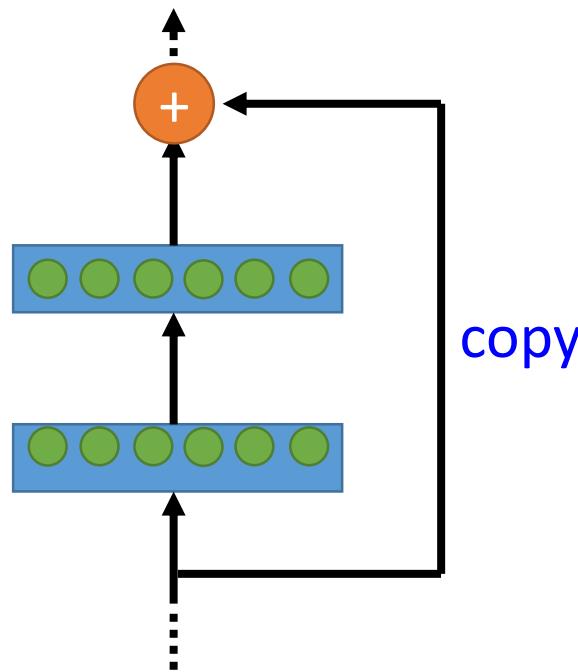
- Q: Can we design the network structure?

Convolutional Neural Network (CNN)
in the next lecture

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

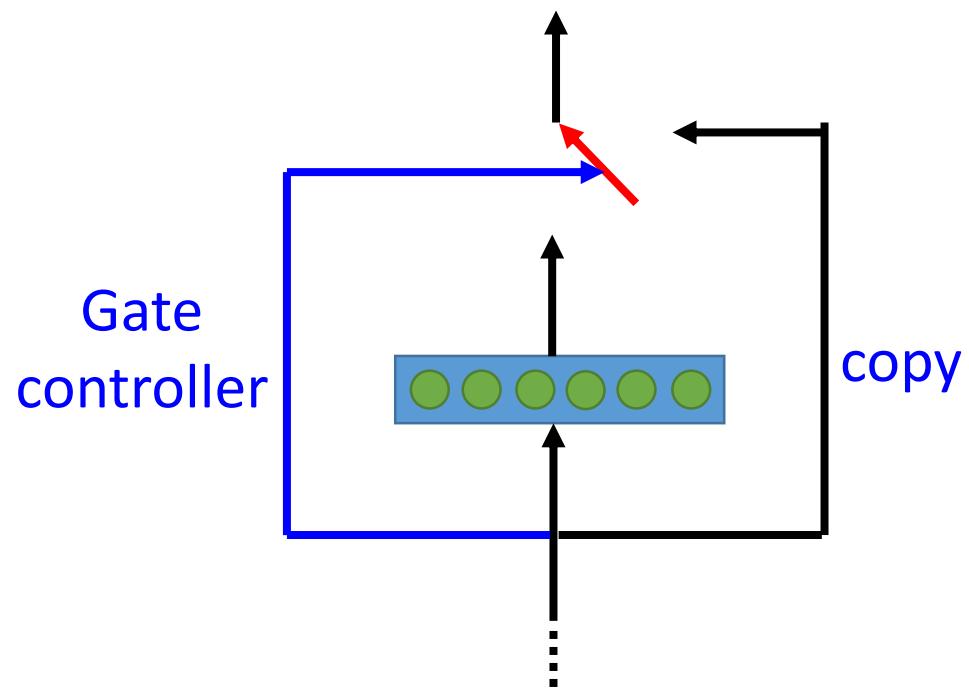
Highway Network

- **Residual Network**



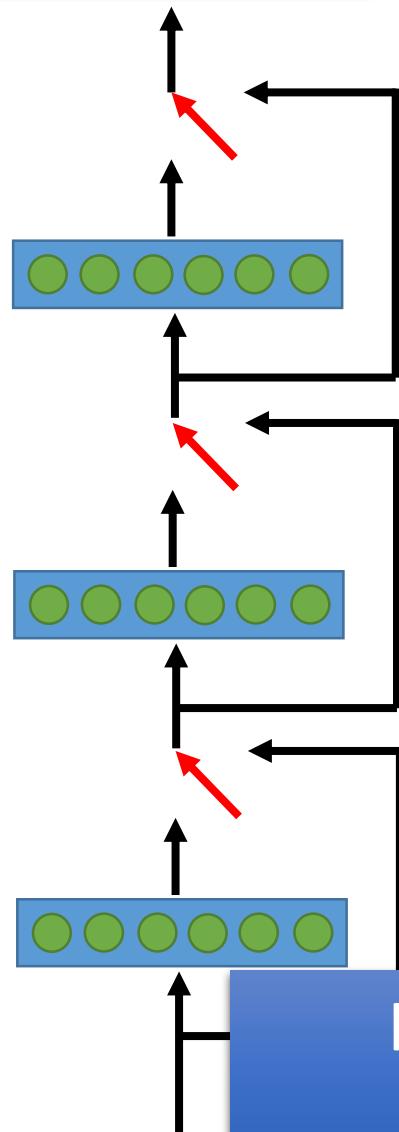
Deep Residual Learning for Image
Recognition
<http://arxiv.org/abs/1512.03385>

- **Highway Network**



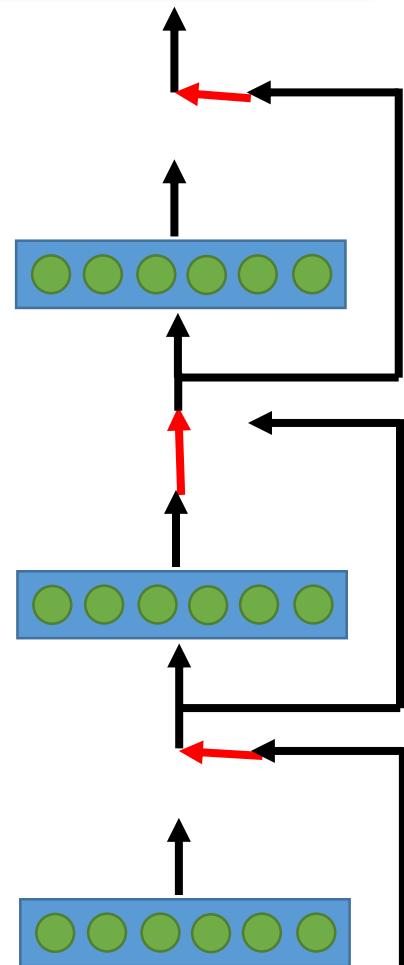
Training Very Deep Networks
<https://arxiv.org/pdf/1507.06228v2.pdf>

output layer



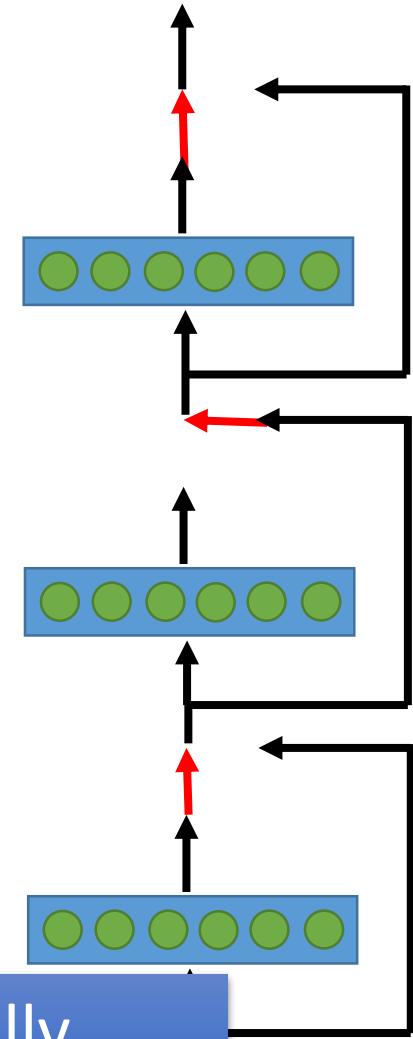
Input layer

output layer



Input layer

output 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



“5”



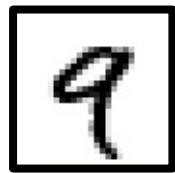
“0”



“4”



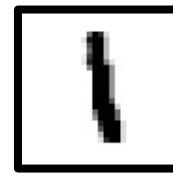
“1”



“9”



“2”



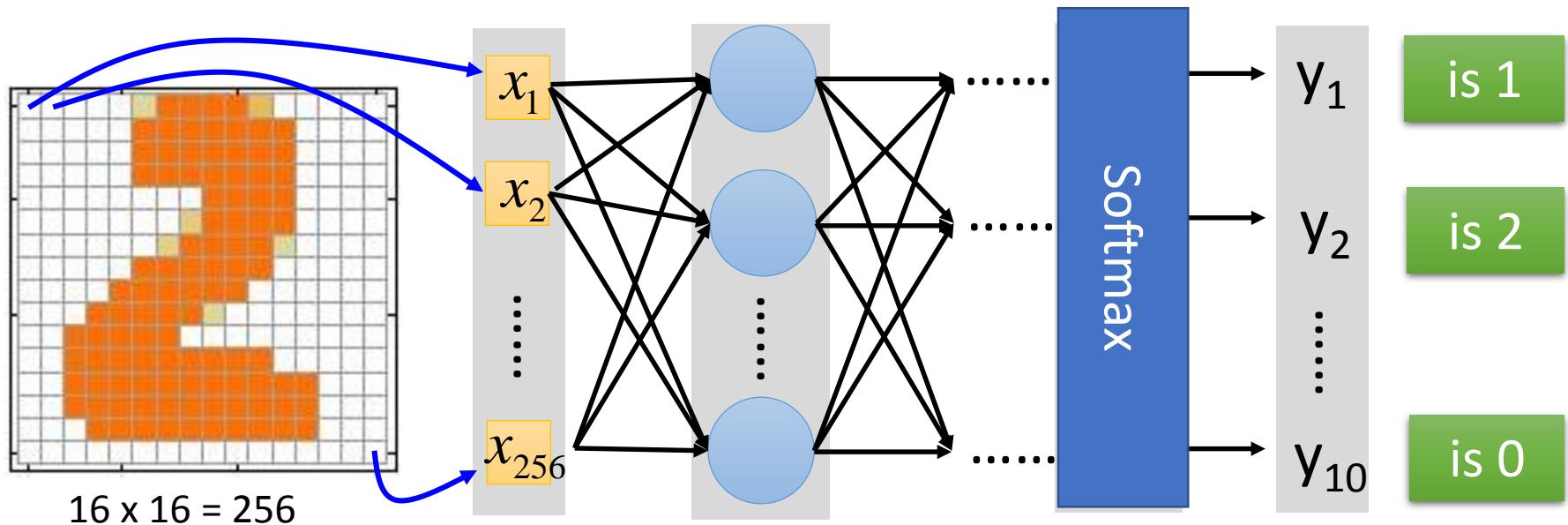
“1”



“3”

The learning target is defined on
the training data.

Learning Target



Ink $\rightarrow 1$

No ink $\rightarrow 0$

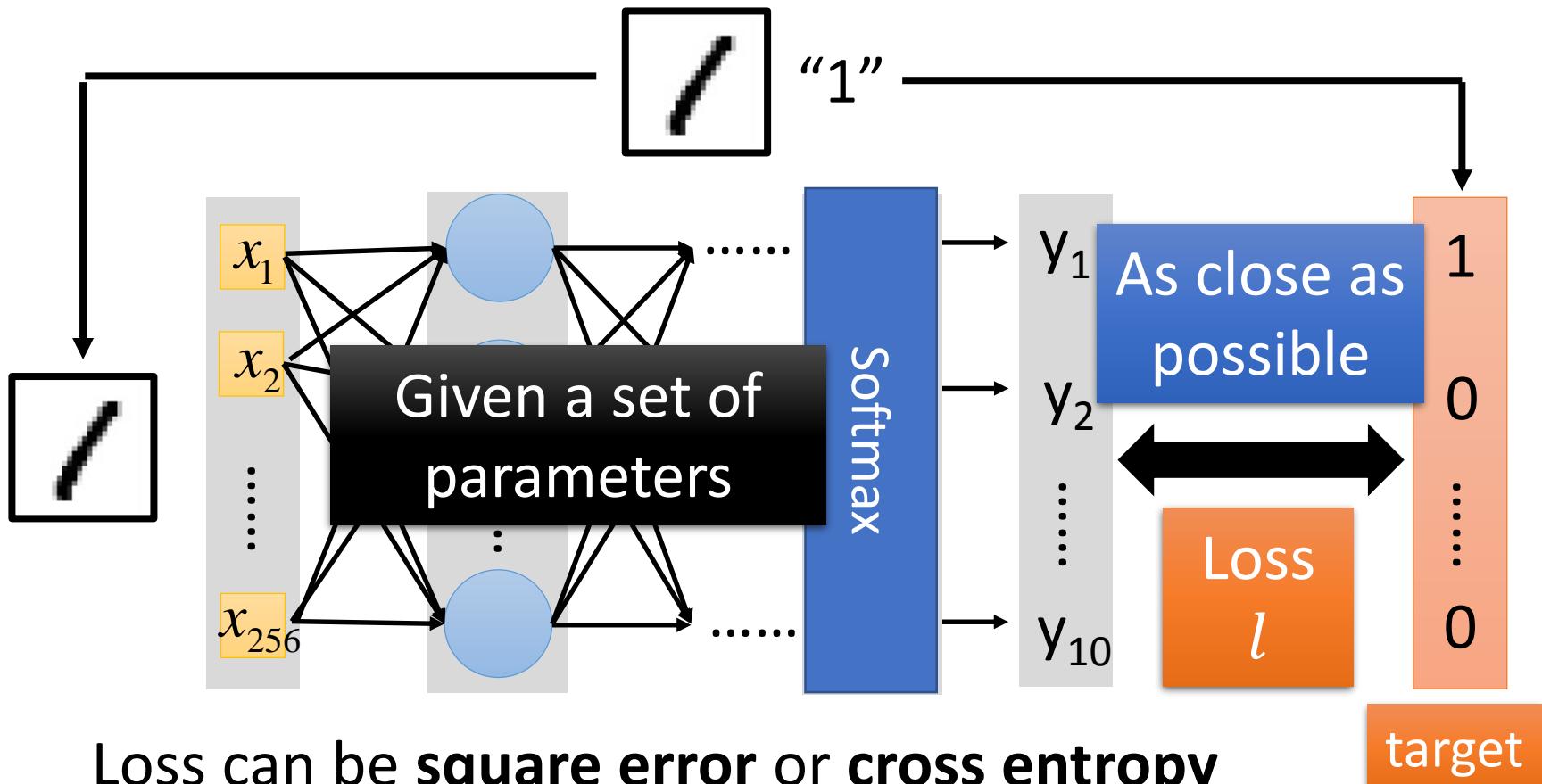
The learning target is

Input:  $\rightarrow y_1$ has the maximum value

Input:  $\rightarrow y_2$ has the maximum value

LOSS

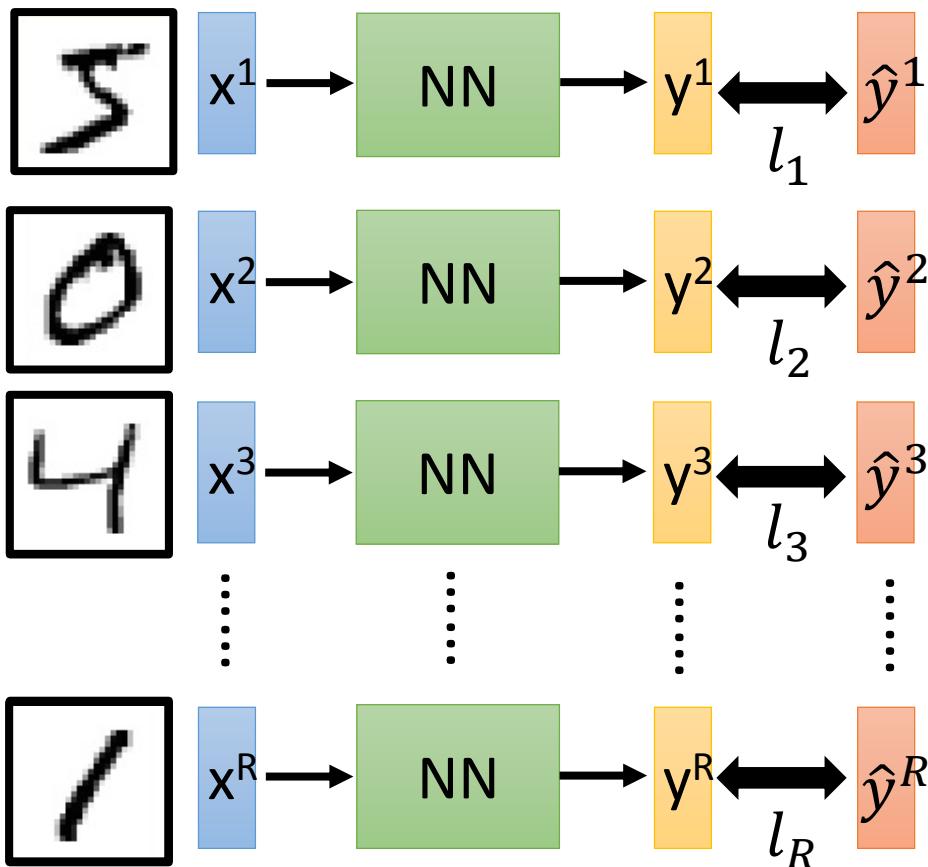
A good function should make the loss of all examples as small as possible.



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

Total Loss

For all training data ...



Total Loss:

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

As small as possible

Find a function in function set that minimizes total loss L

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

Three Steps for Deep Learning

Step 1: define a set of function



Step 2: goodness of function



Step 3: pick the best function

How to pick the best function

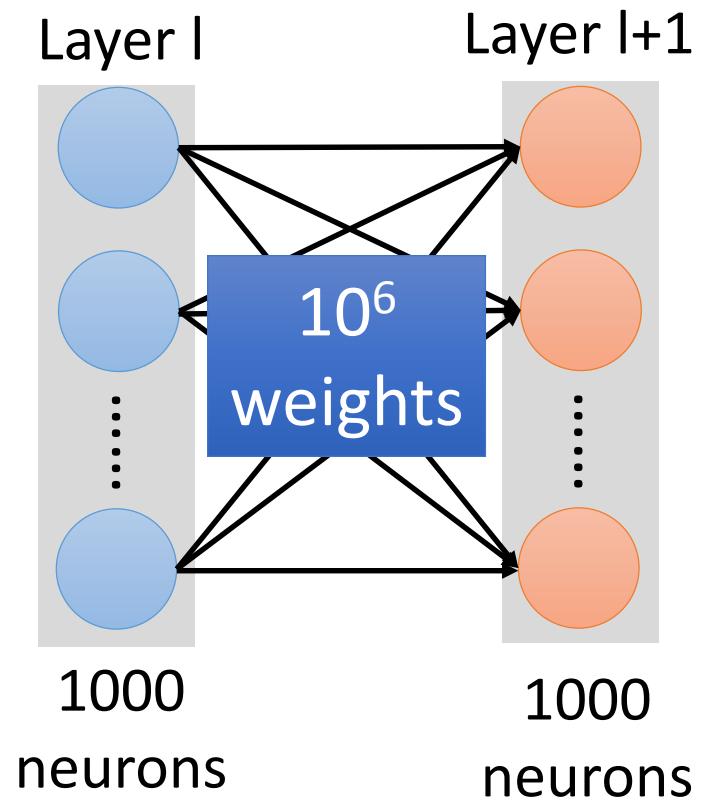
Find network parameters θ^* that minimize total loss L

Enumerate all possible values

Network parameters θ
 $= \{w_1, w_2, \dots, b_1, b_2, b_3, \dots\}$

Millions of parameters

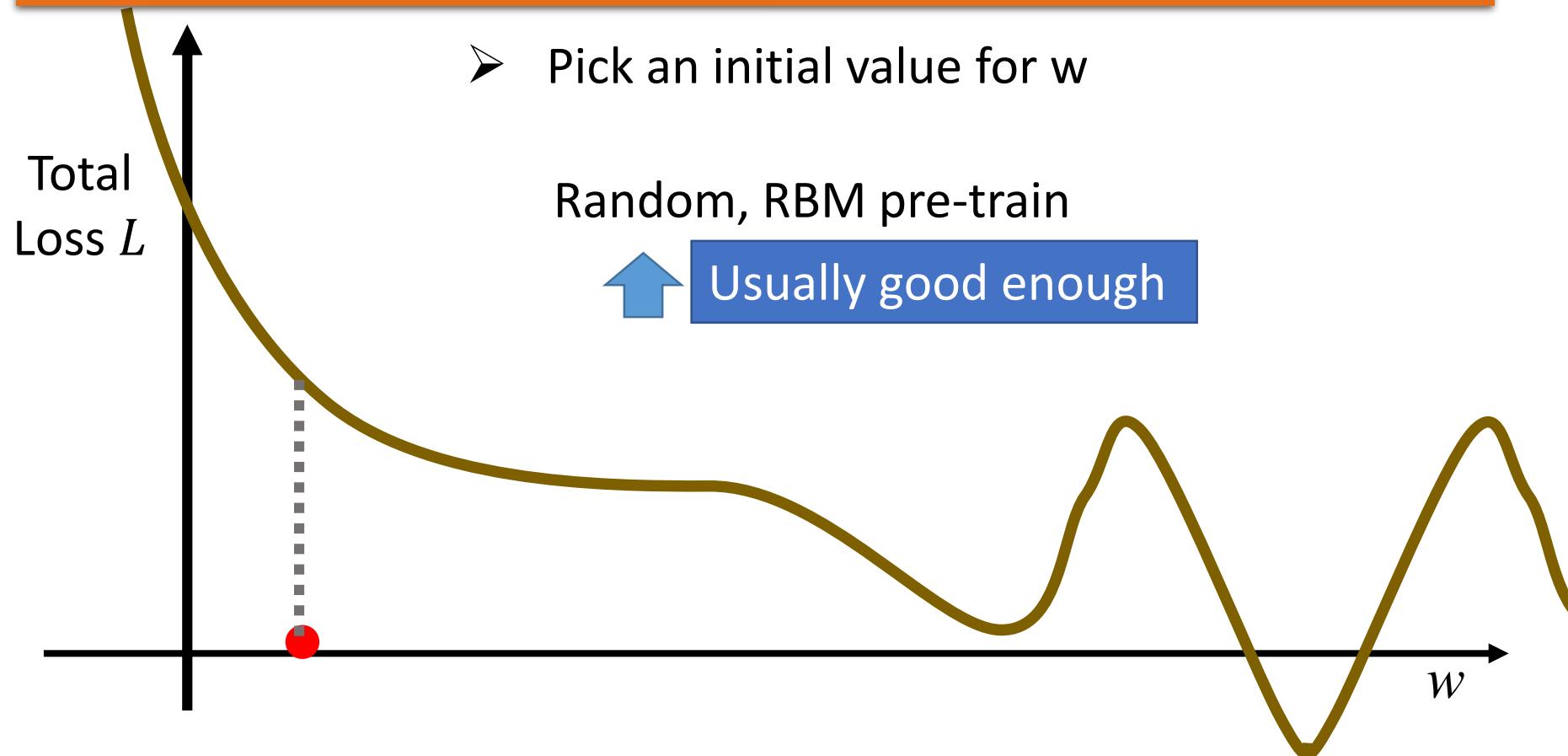
E.g. speech recognition: 8 layers and
1000 neurons each layer



Gradient Descent

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

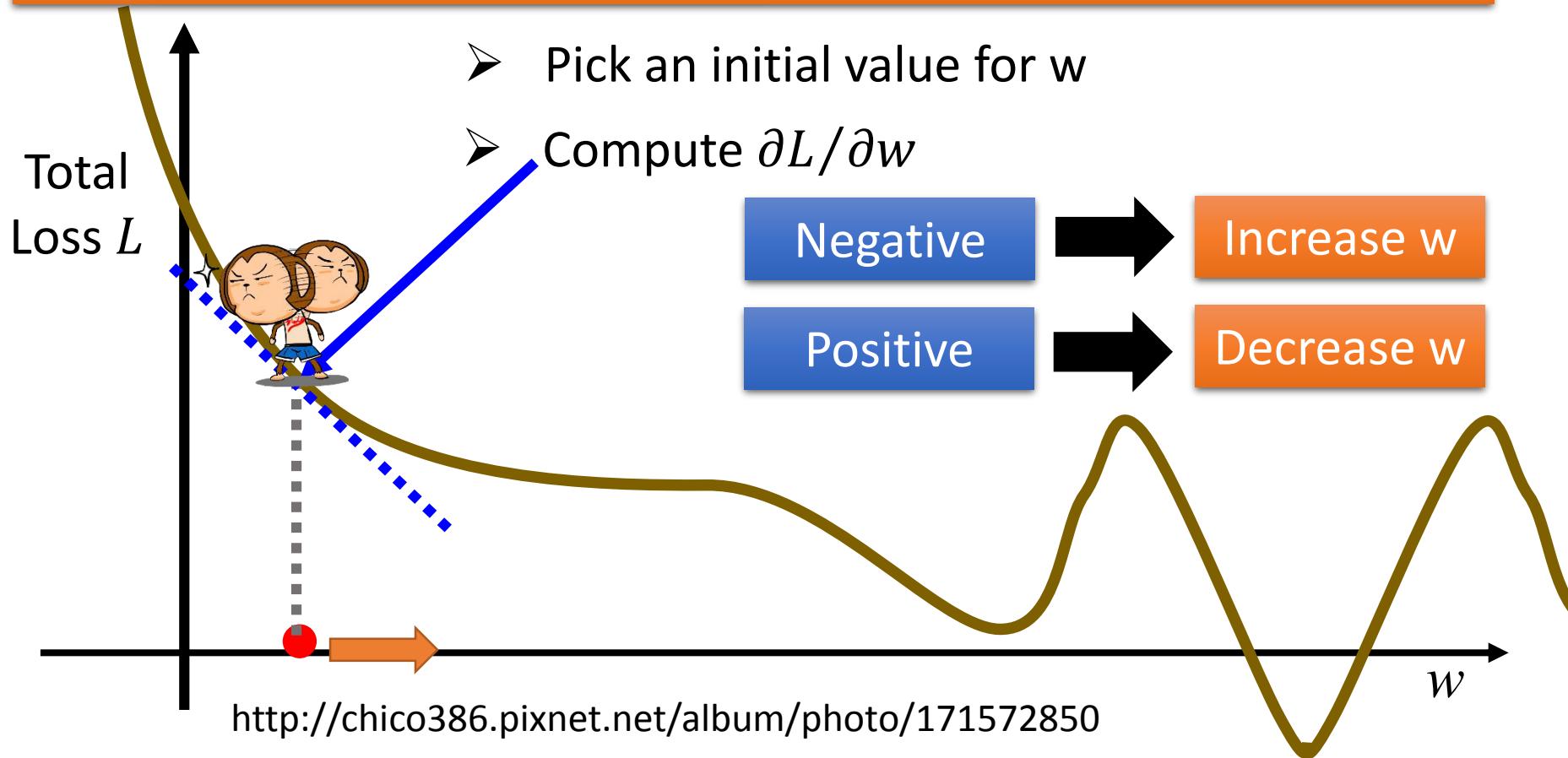
Find network parameters θ^* that minimize total loss L



Gradient Descent

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

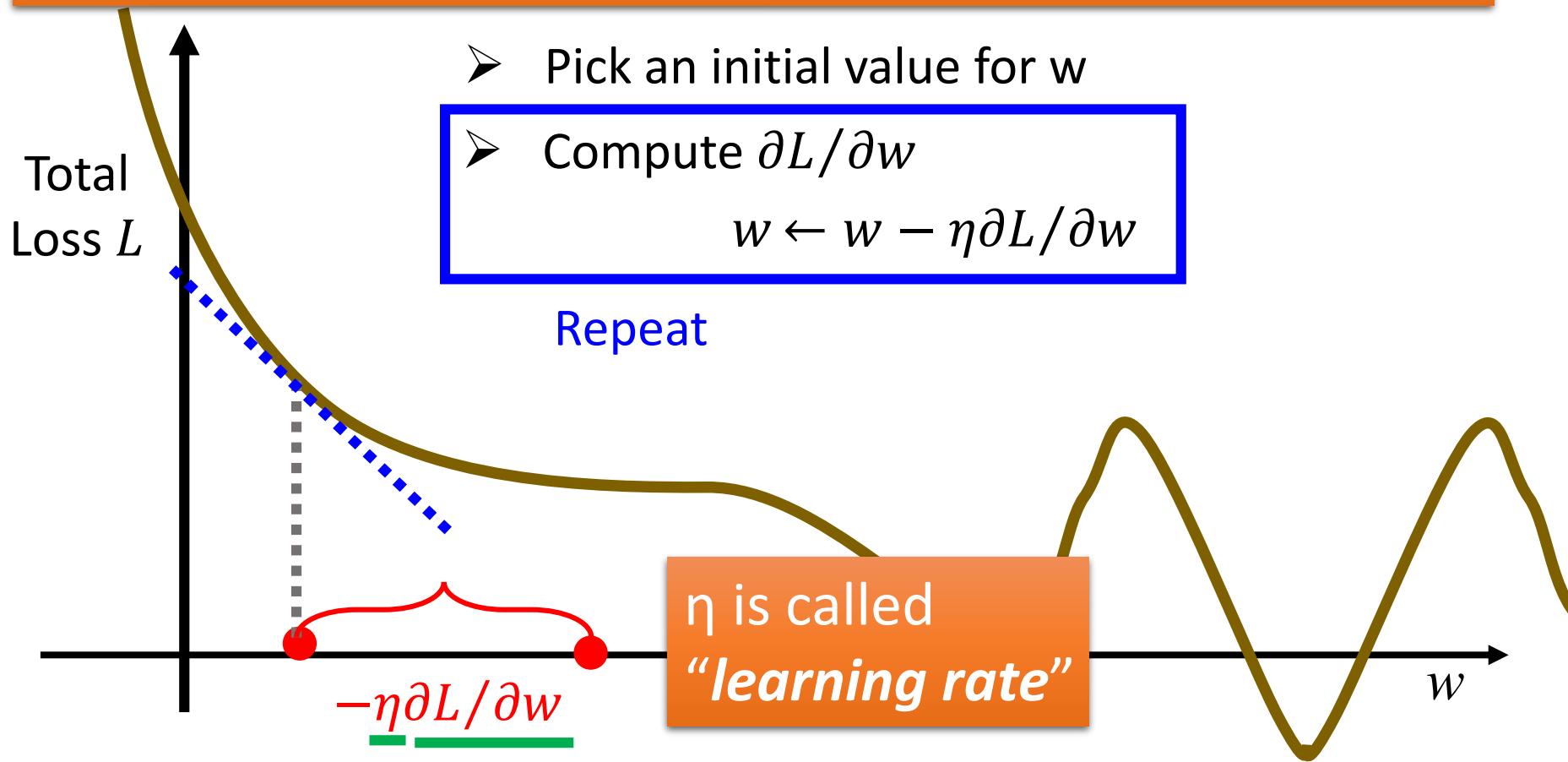
Find network parameters θ^* that minimize total loss L



Gradient Descent

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

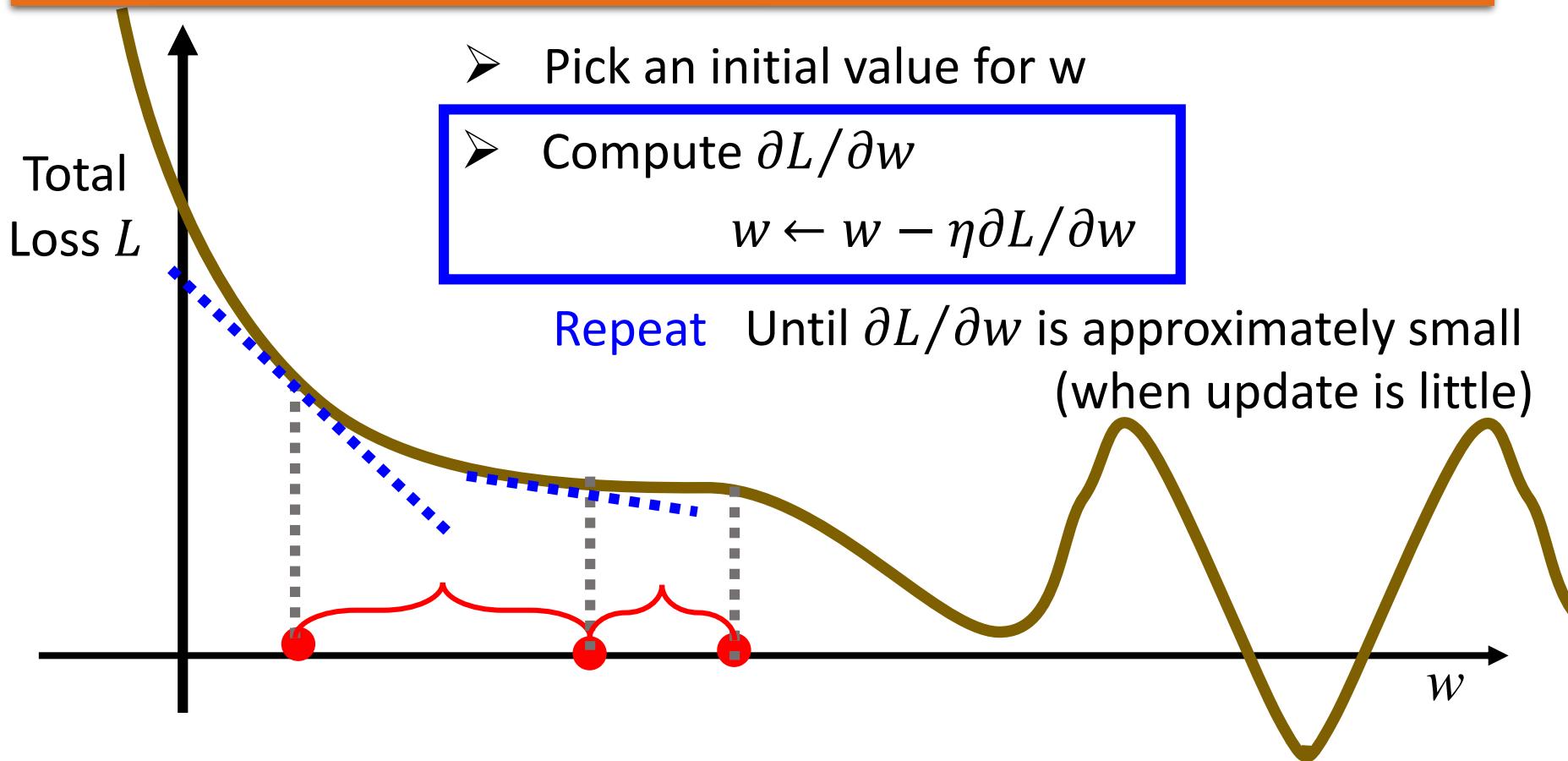
Find network parameters θ^* that minimize total loss L



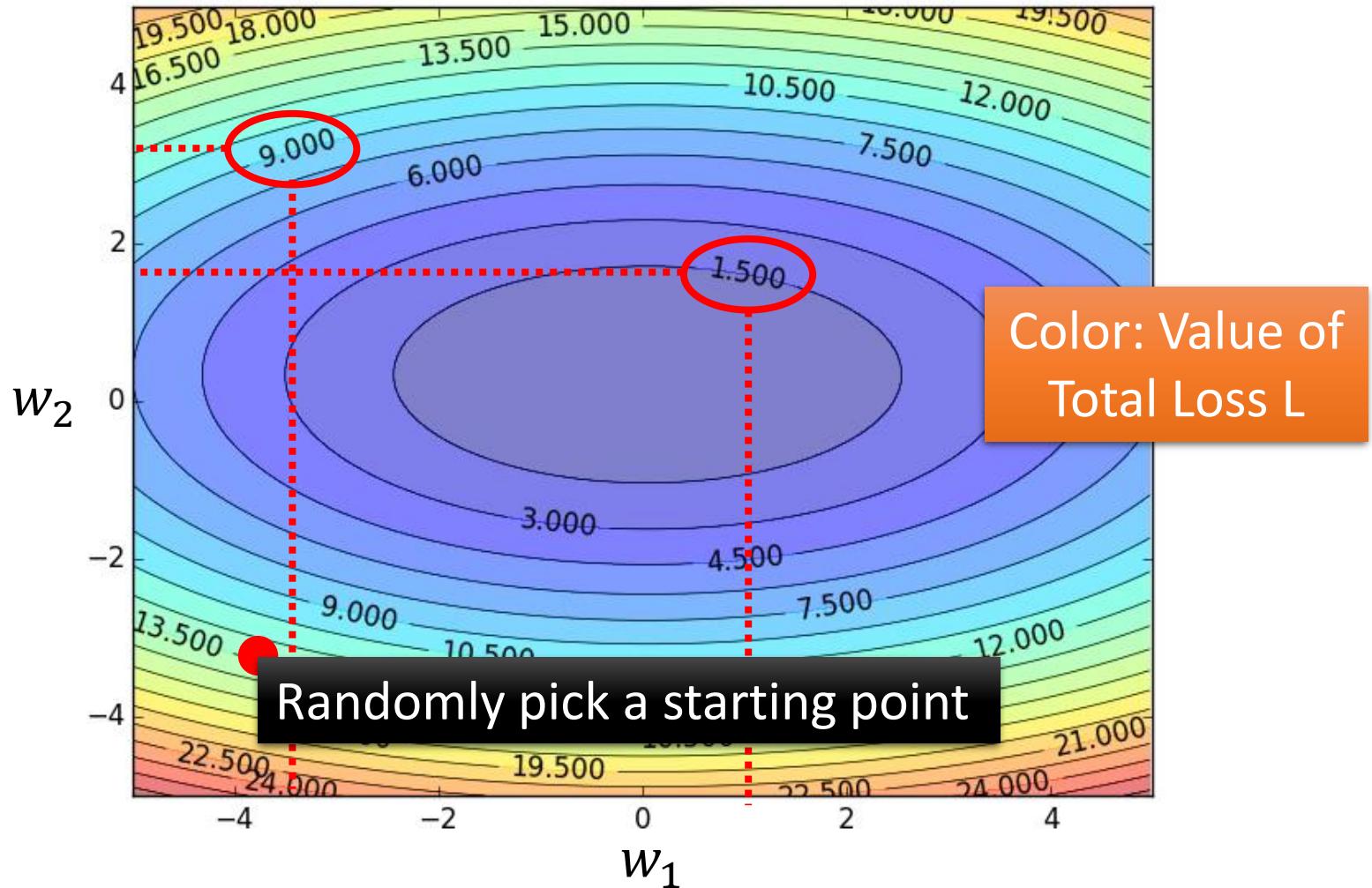
Gradient Descent

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

Find network parameters θ^* that minimize total loss L

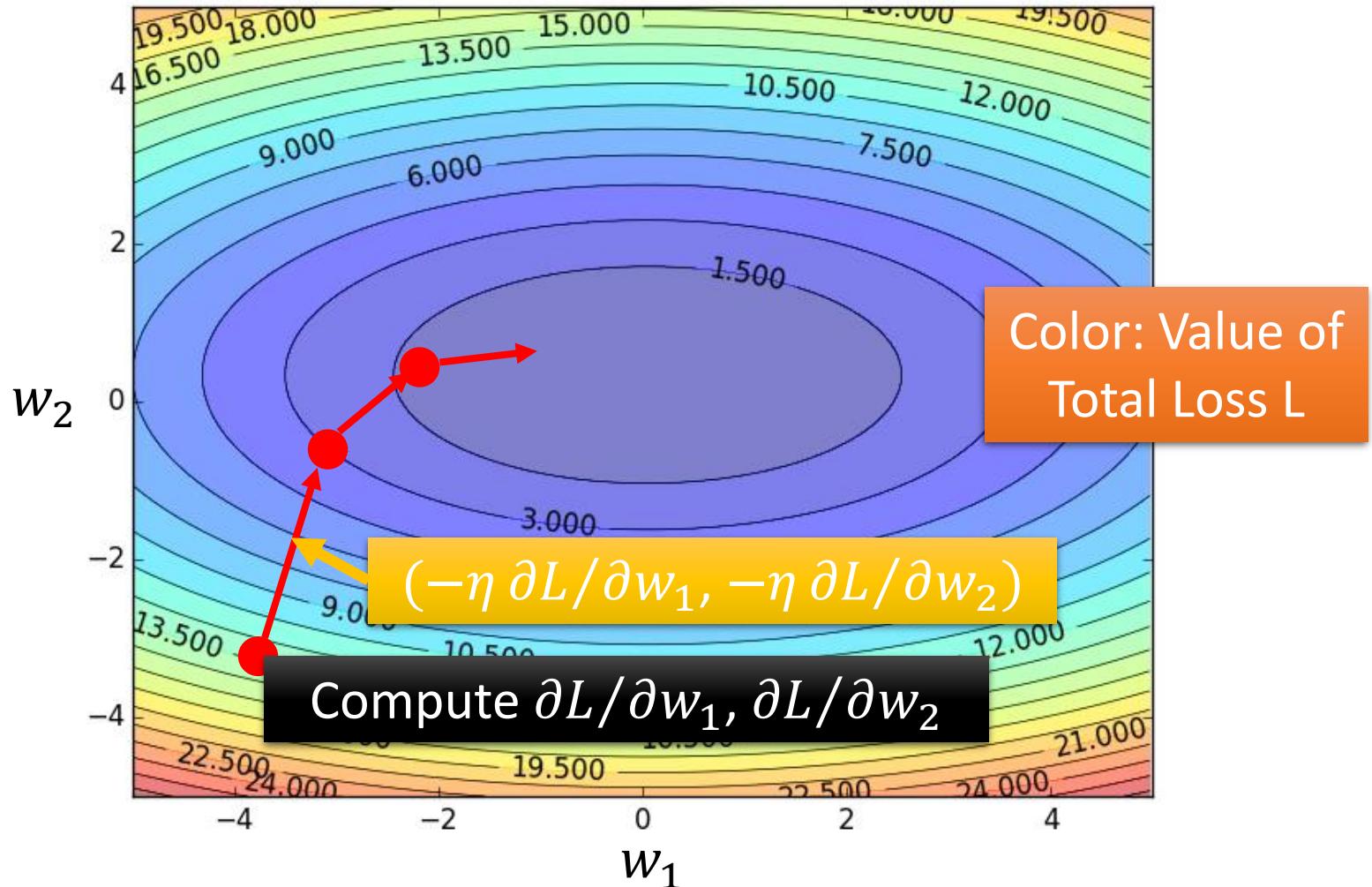


Gradient Descent

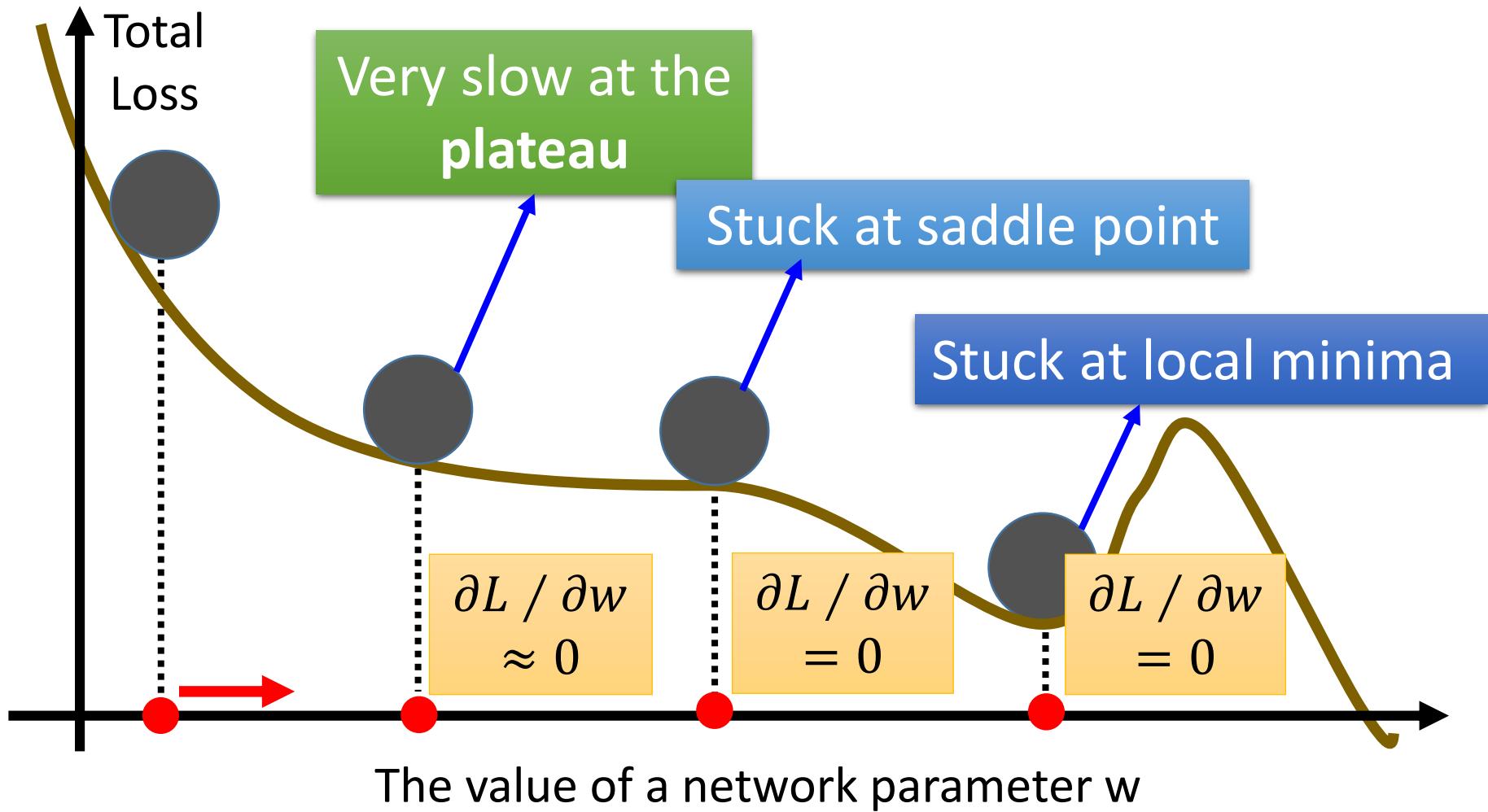


Gradient Descent

Hopfully, we would reach a minima

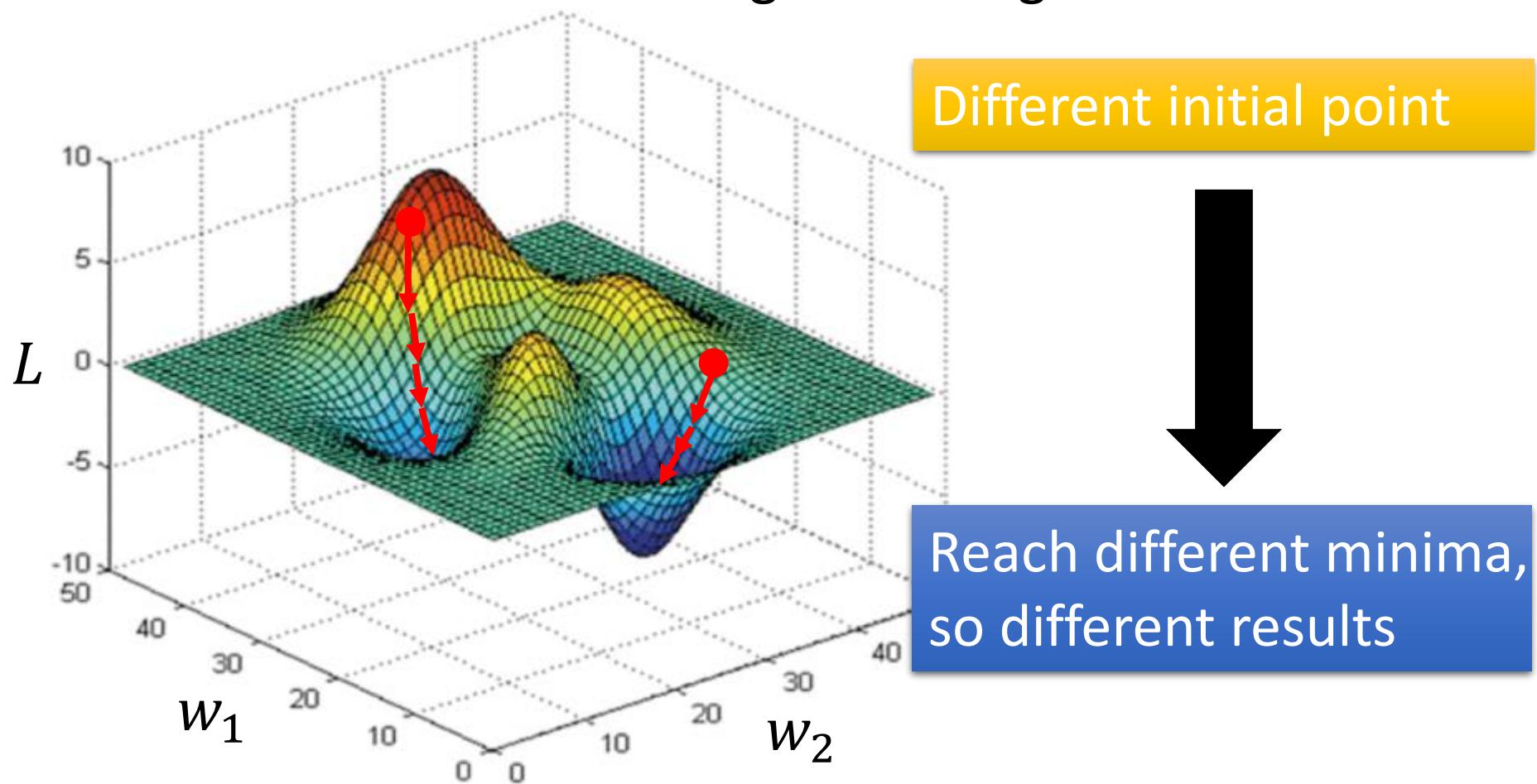


Local Minima



Local Minima

- Gradient descent never guarantee global minima

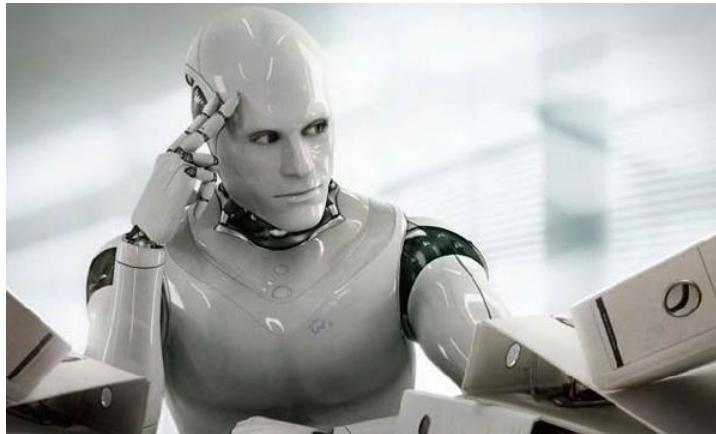


Gradient Descent

This is the “learning” of machines in deep learning

→ Even alpha go using this approach.

People image



Actually



I hope you are not too disappointed :p

Backpropagation

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



theano

Caffe

The Microsoft CNTK logo features the Microsoft logo (four colored squares) followed by the word "CNTK" in a large, dark grey, sans-serif font.The MXNet logo features the letters "mxnet" in a white, lowercase, sans-serif font inside a blue rounded rectangle.

libdnn
台大周伯威
同學開發

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

Three Steps for Deep Learning



Deep Learning is so simple

Now If you want to find a function

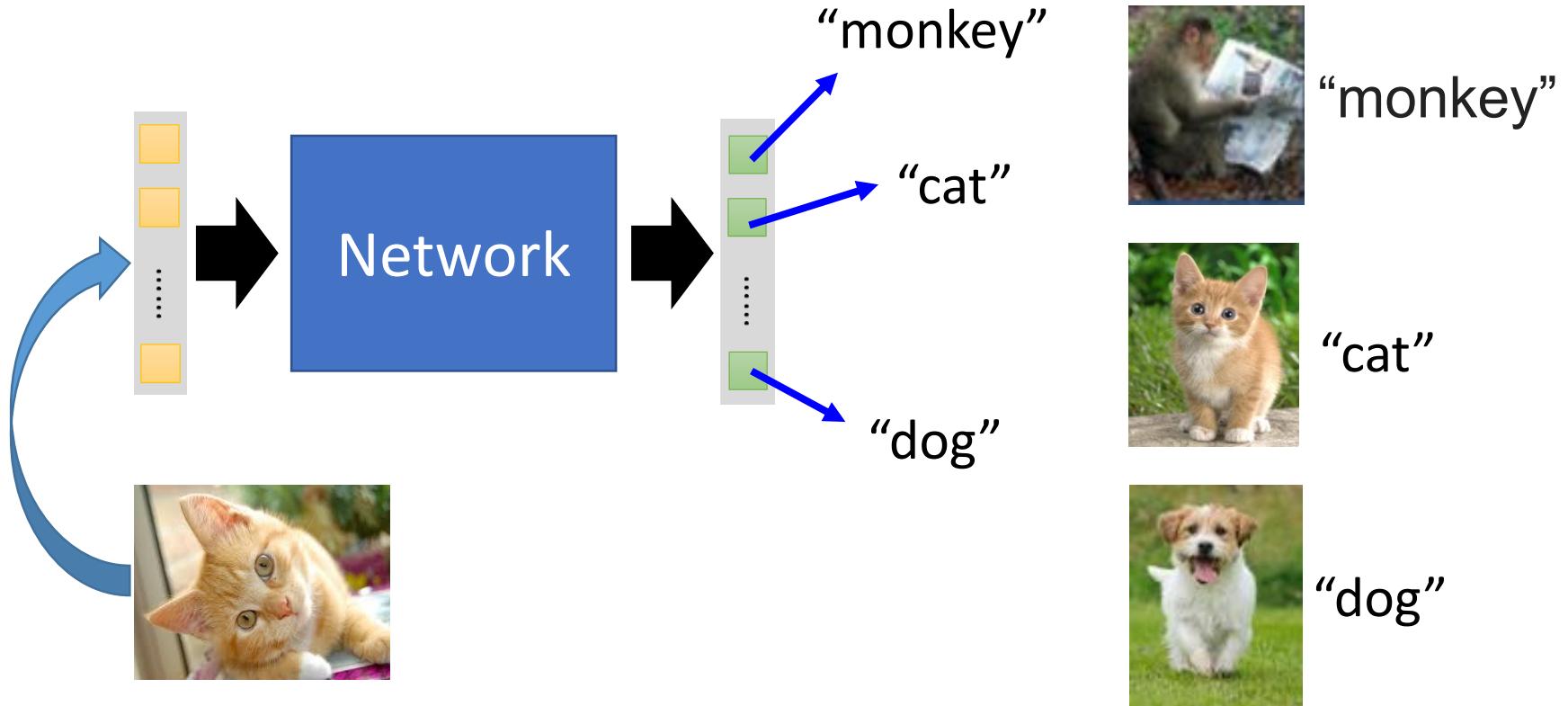
If you have lots of function input/output (?) as training data



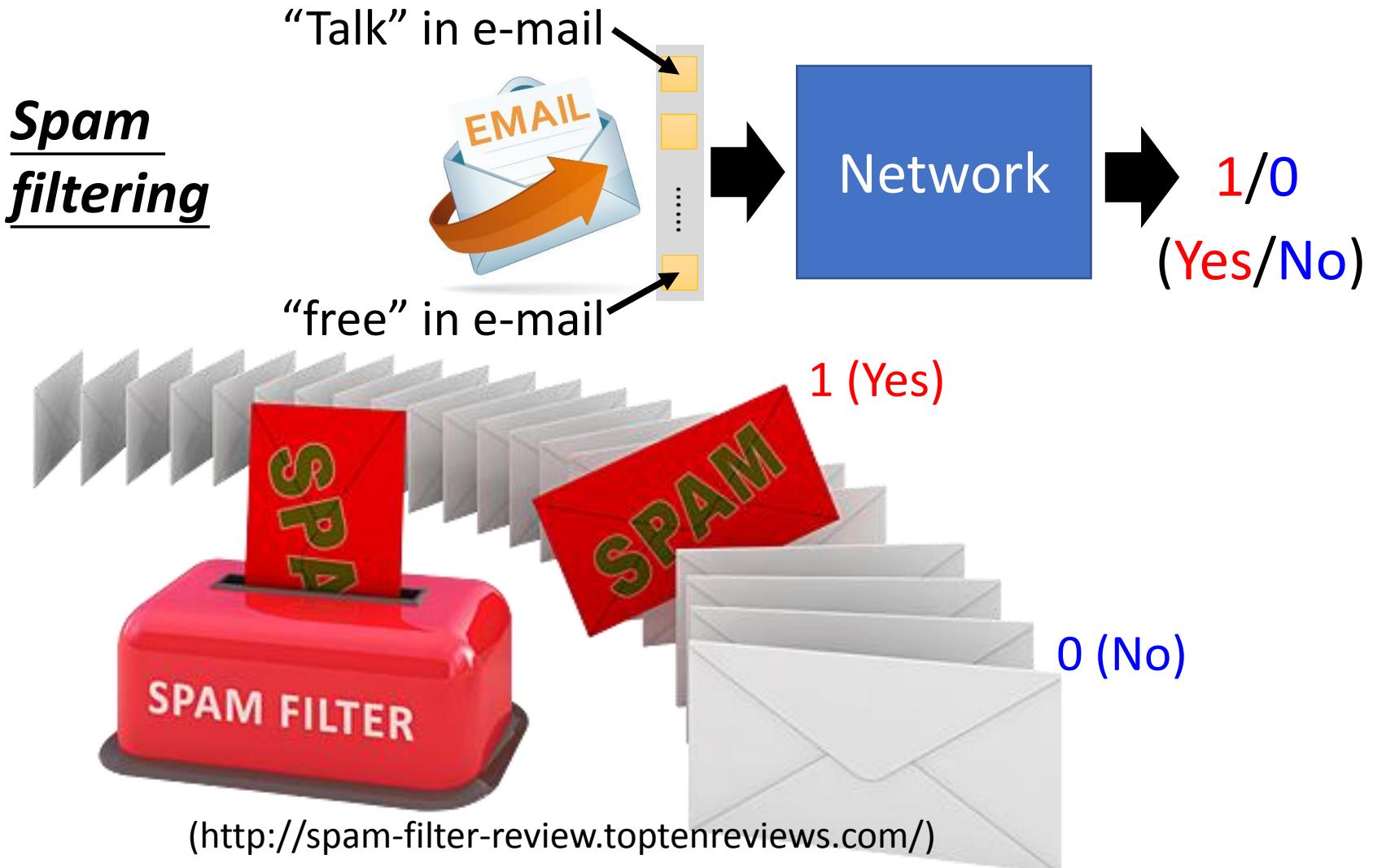
You can use deep learning

For example, you can do

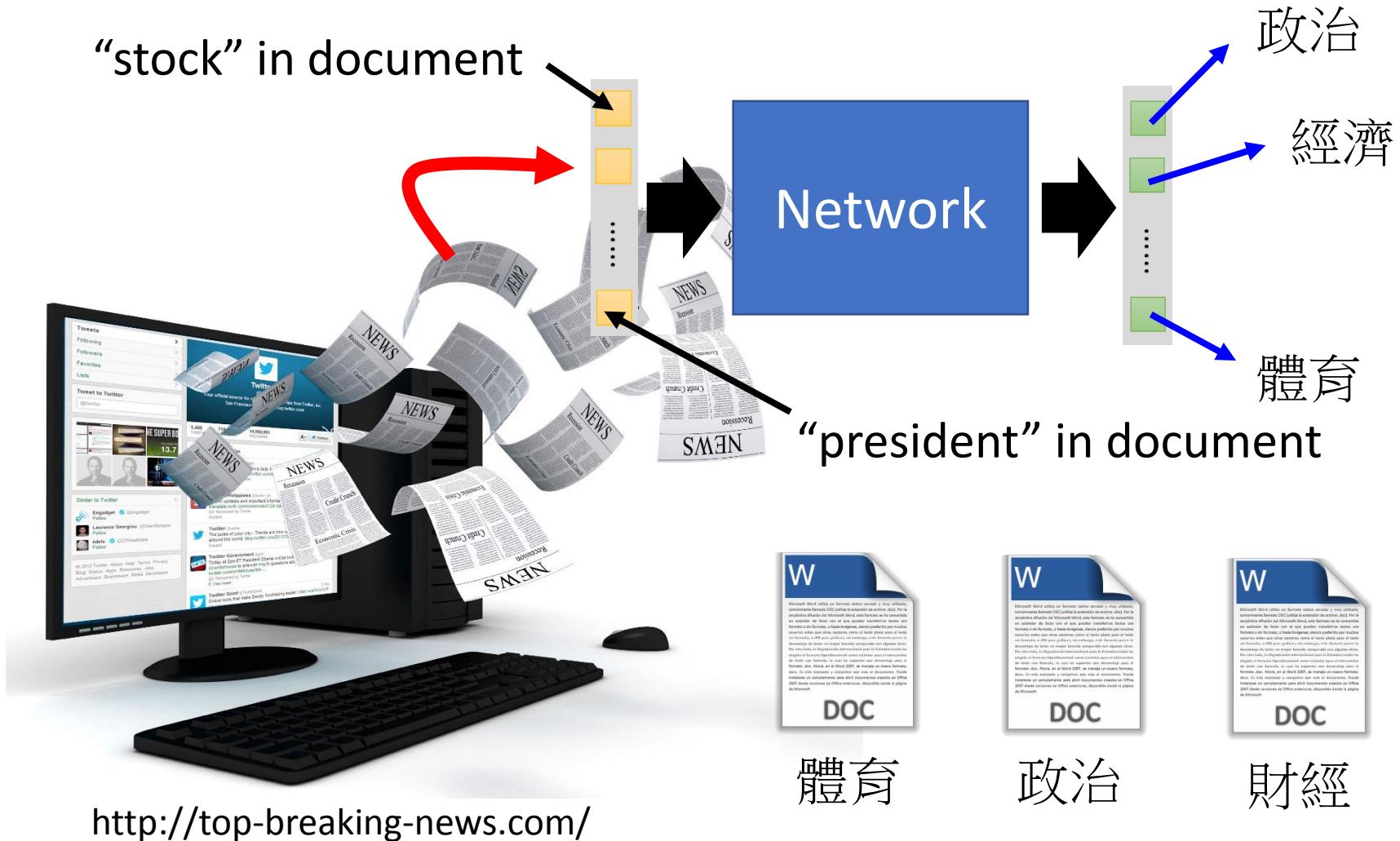
- Image Recognition



For example, you can do



For example, you can do



Outline

Introduction of Deep Learning

“Hello World” for Deep Learning

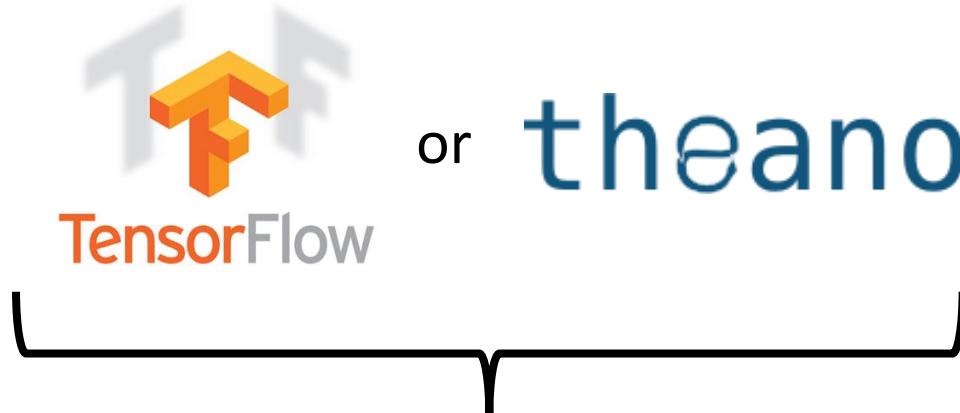
Tips for Deep Learning

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



Interface of
TensorFlow or
Theano



Very flexible
Need some
effort to learn

Easy to learn and use
(still have some flexibility)

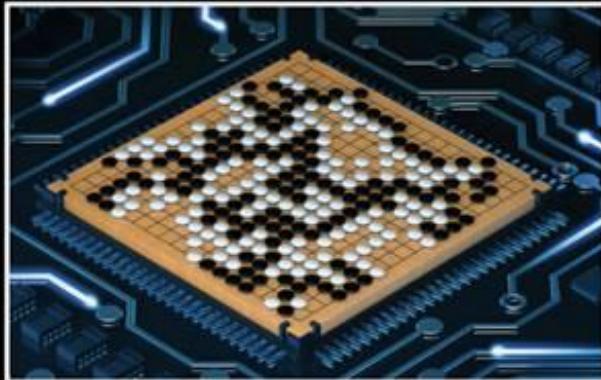
You can modify it if you can write
TensorFlow or Theano

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>

使用 Keras 心得

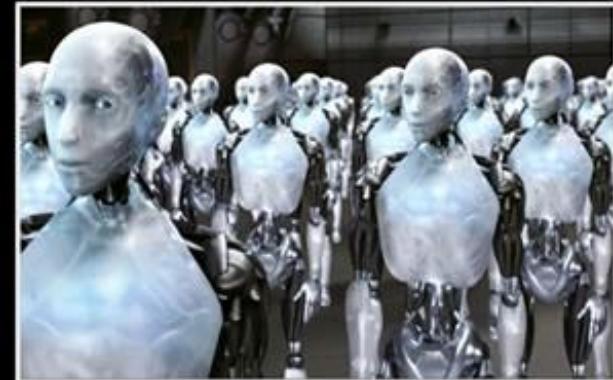
Deep Learning研究生



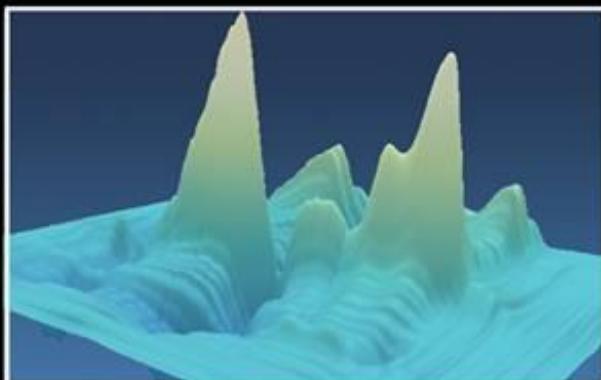
朋友覺得我在



我媽覺得我在



大眾覺得我在



指導教授覺得我在



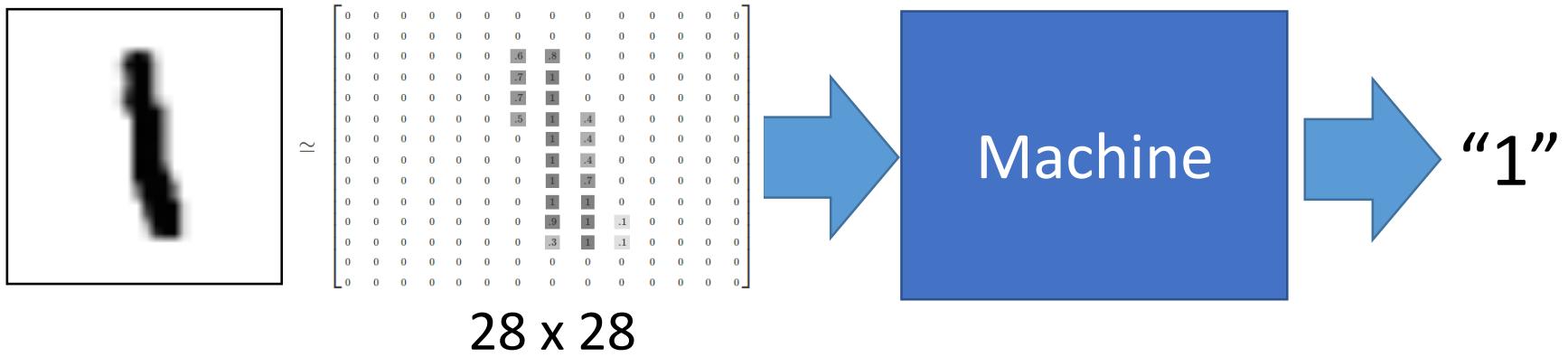
我以為我在



事實上我在

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

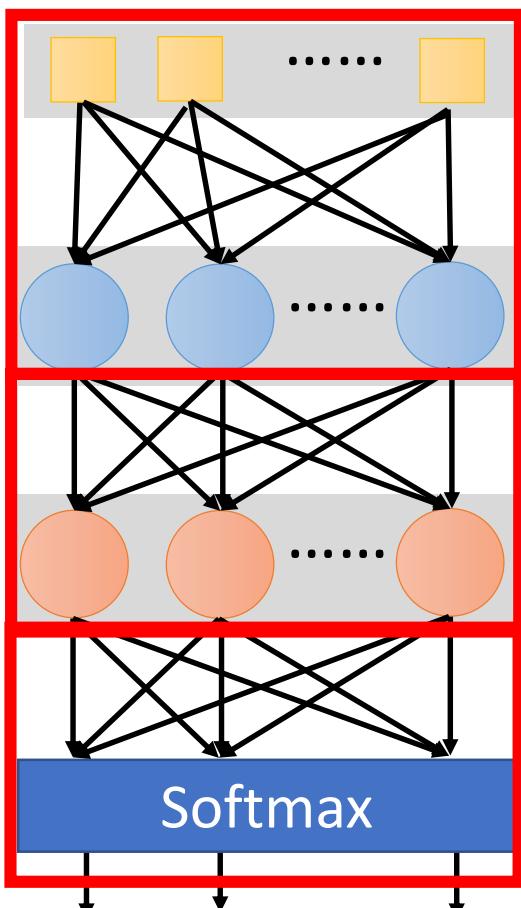
28x28

500

500

Softmax

y_1 y_2 y_{10}



```
model = Sequential()
```

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

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

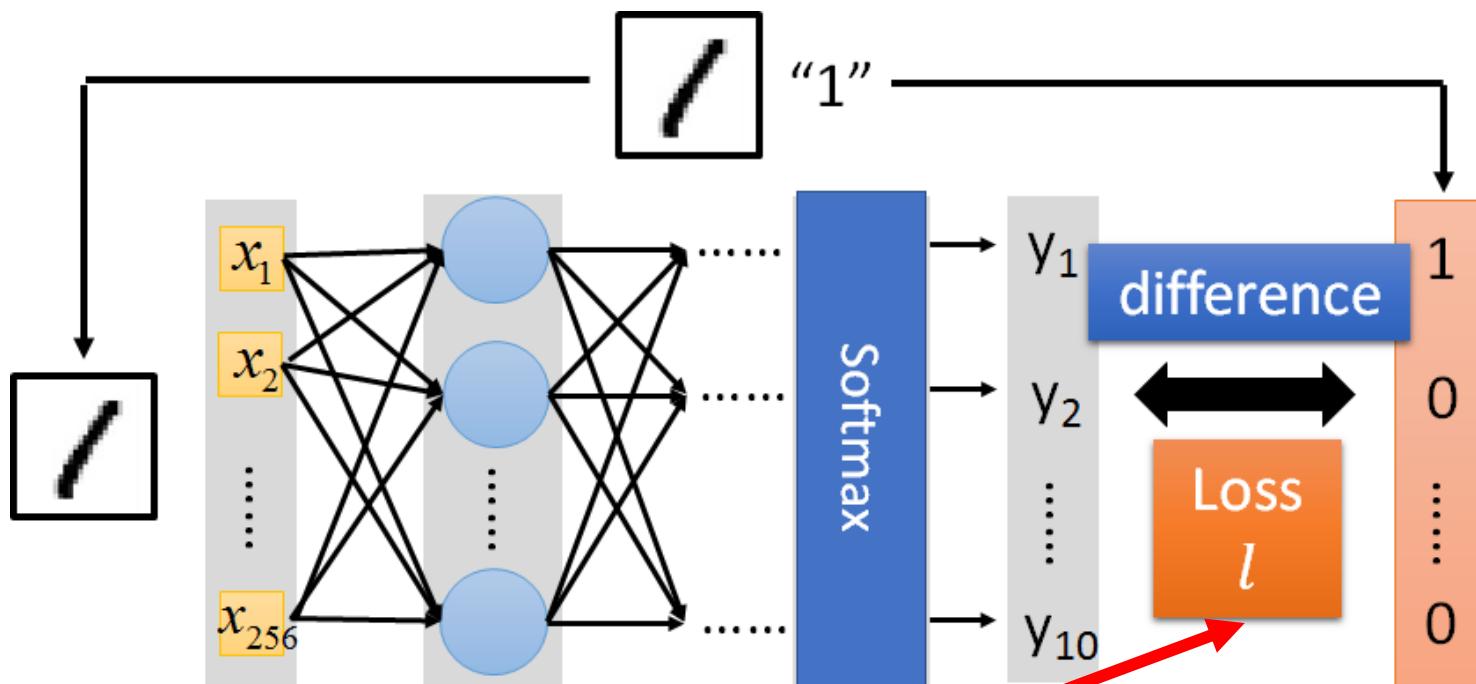
```
model.add( Dense( output_dim=10 ) )  
model.add( Activation('softmax') )
```

Keras

Step 1:
define a set
of function

Step 2:
goodness of
function

Step 3: pick
the best
function



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

Keras

Step 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'])
```

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

0.1

Step 3.2: Find the optimal network parameters

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

Training data
(Images)

Labels
(digits)

Keras

Step 1:
define a set
of function

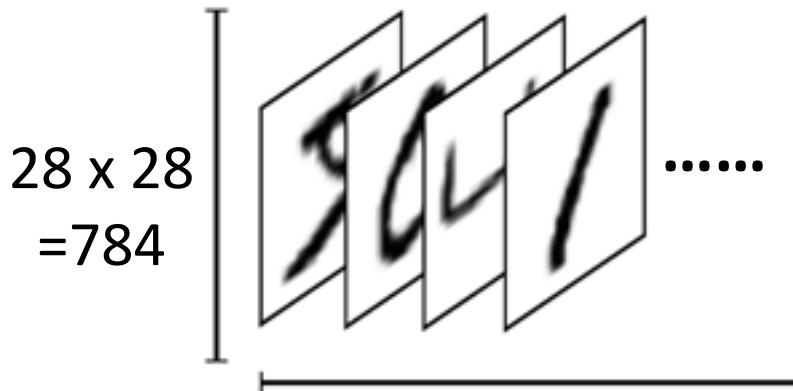
Step 2:
goodness of
function

Step 3: pick
the best
function

Step 3.2: Find the optimal network parameters

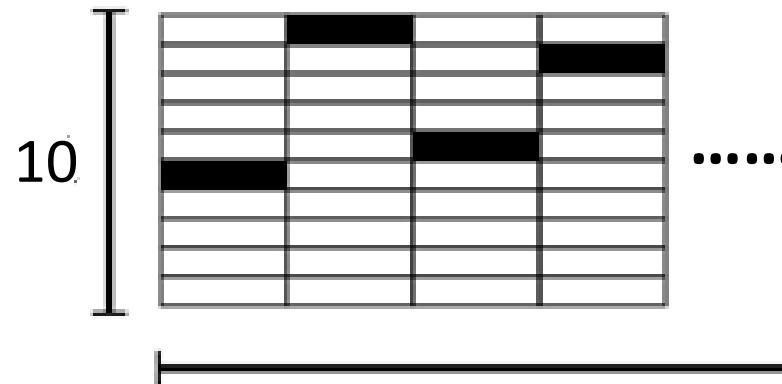
```
model.fit(x_train, y_train, batch_size=100, nb_epoch=20)
```

numpy array



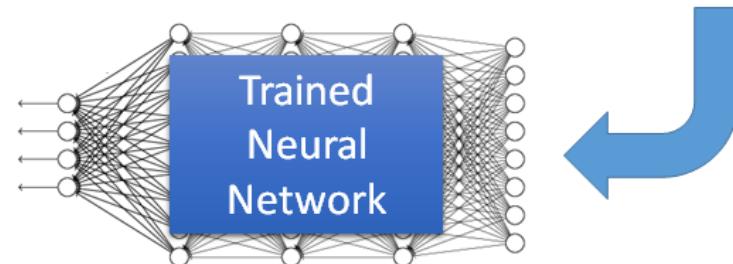
Number of training examples

numpy array



Number of training examples

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):

```
score = model.evaluate(x_test, y_test)
case 1: 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

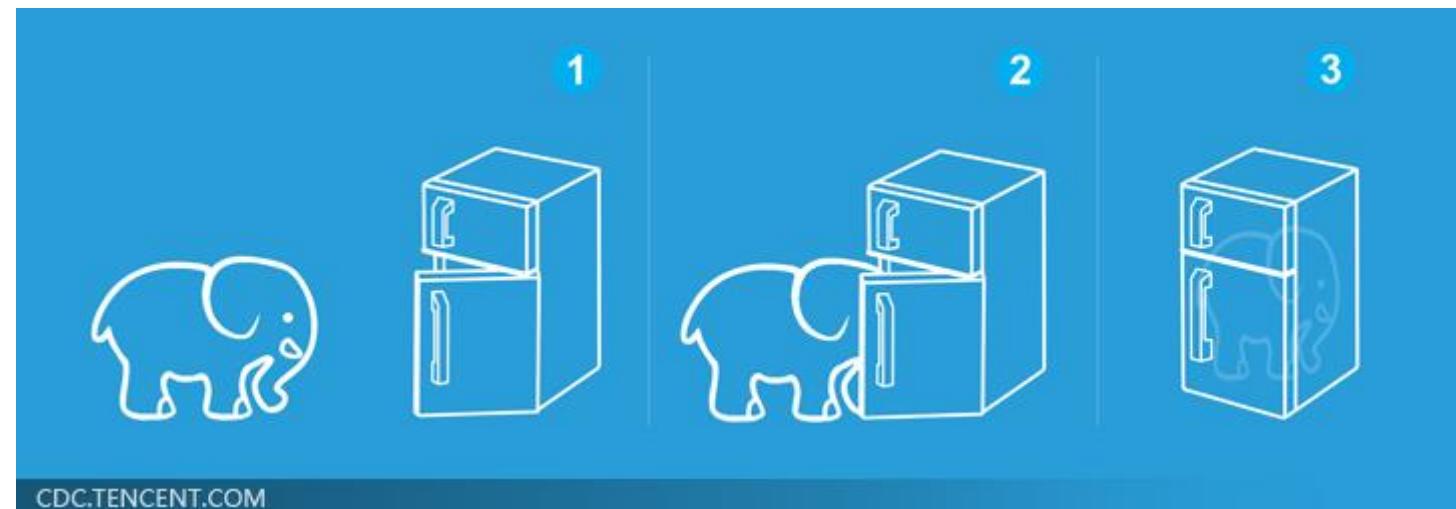
Three Steps for Deep Learning

Step 1:
define a set
of function

Step 2:
goodness of
function

Step 3: pick
the best
function

Deep Learning is so simple



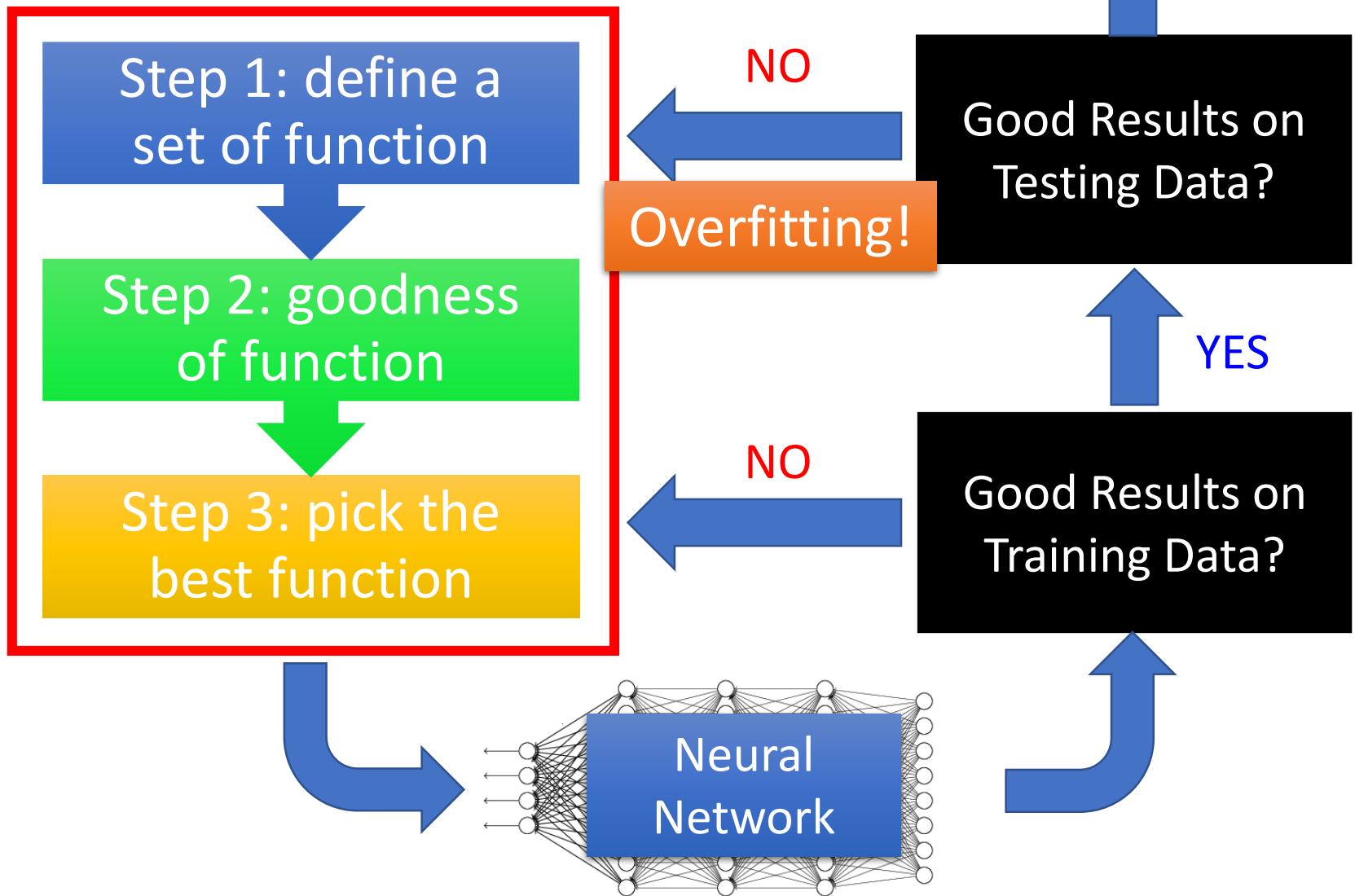
Outline

Introduction of Deep Learning

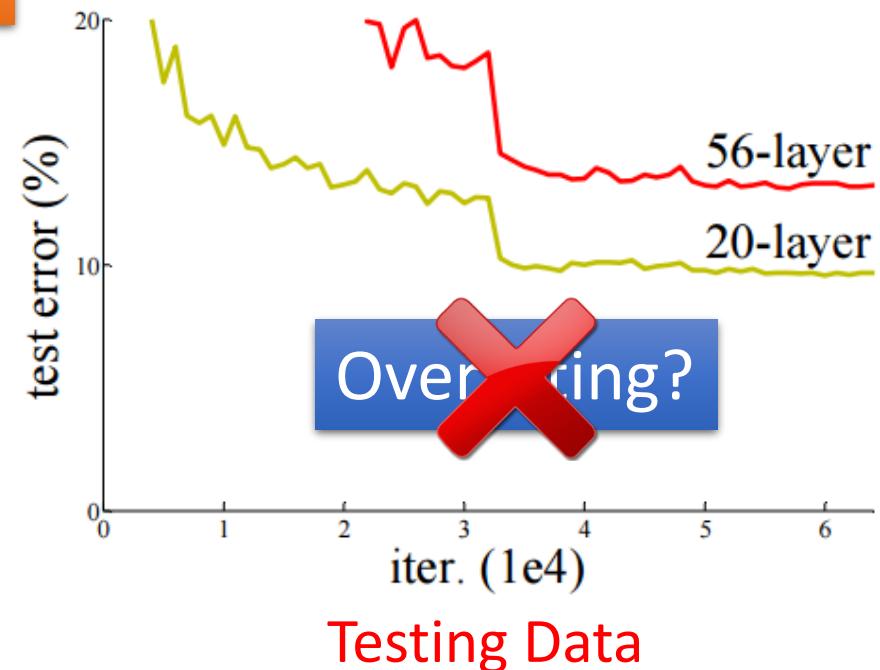
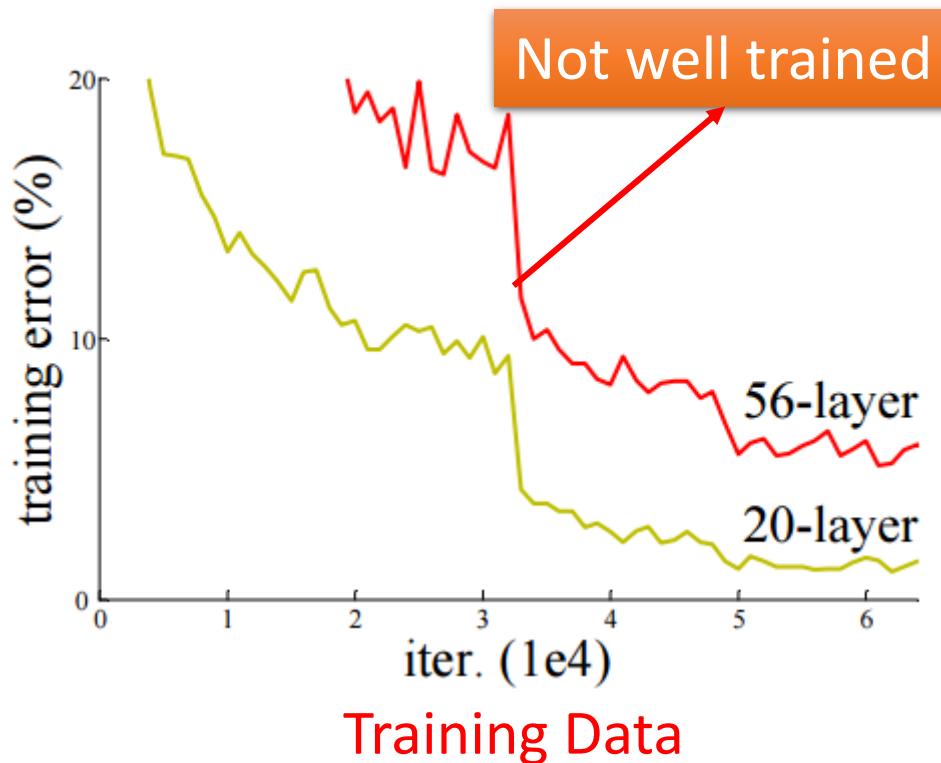
“Hello World” for Deep Learning

Tips for Deep Learning

Recipe of Deep Learning



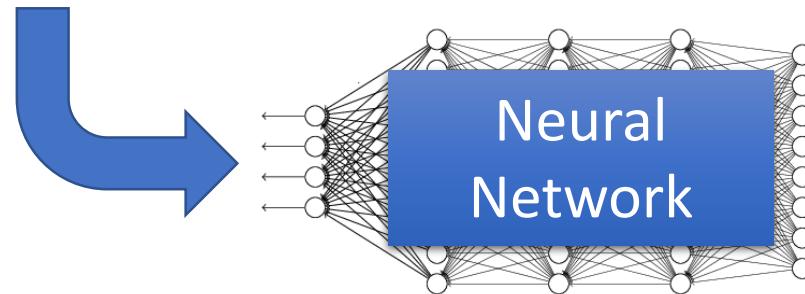
Do not always blame Overfitting



Recipe of Deep Learning

Different approaches for different problems.

e.g. dropout for good results on testing data



Good Results on Testing Data?

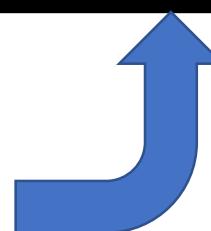
Good Results on Training Data?



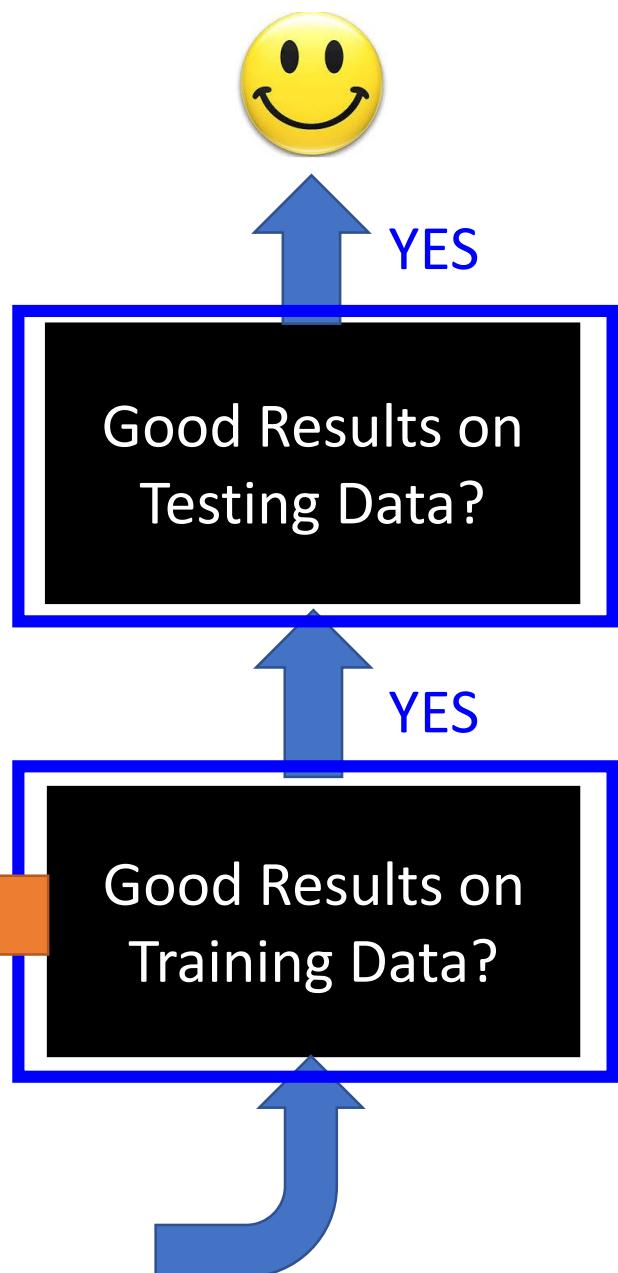
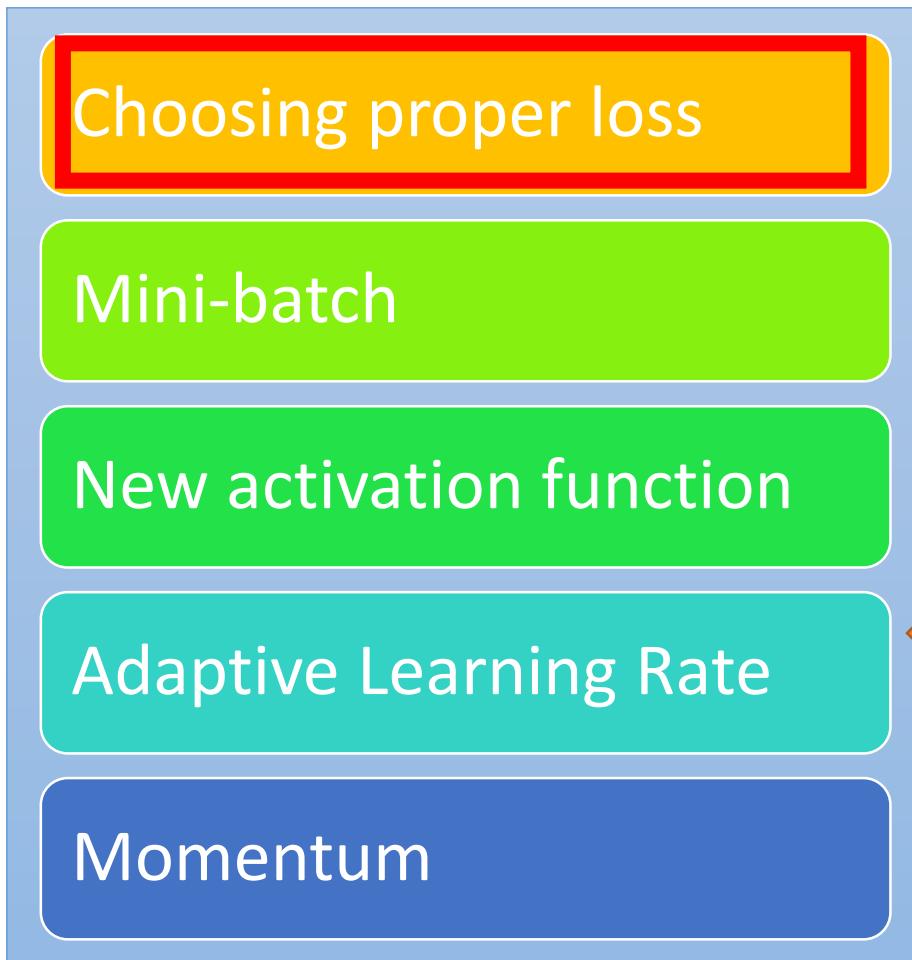
YES



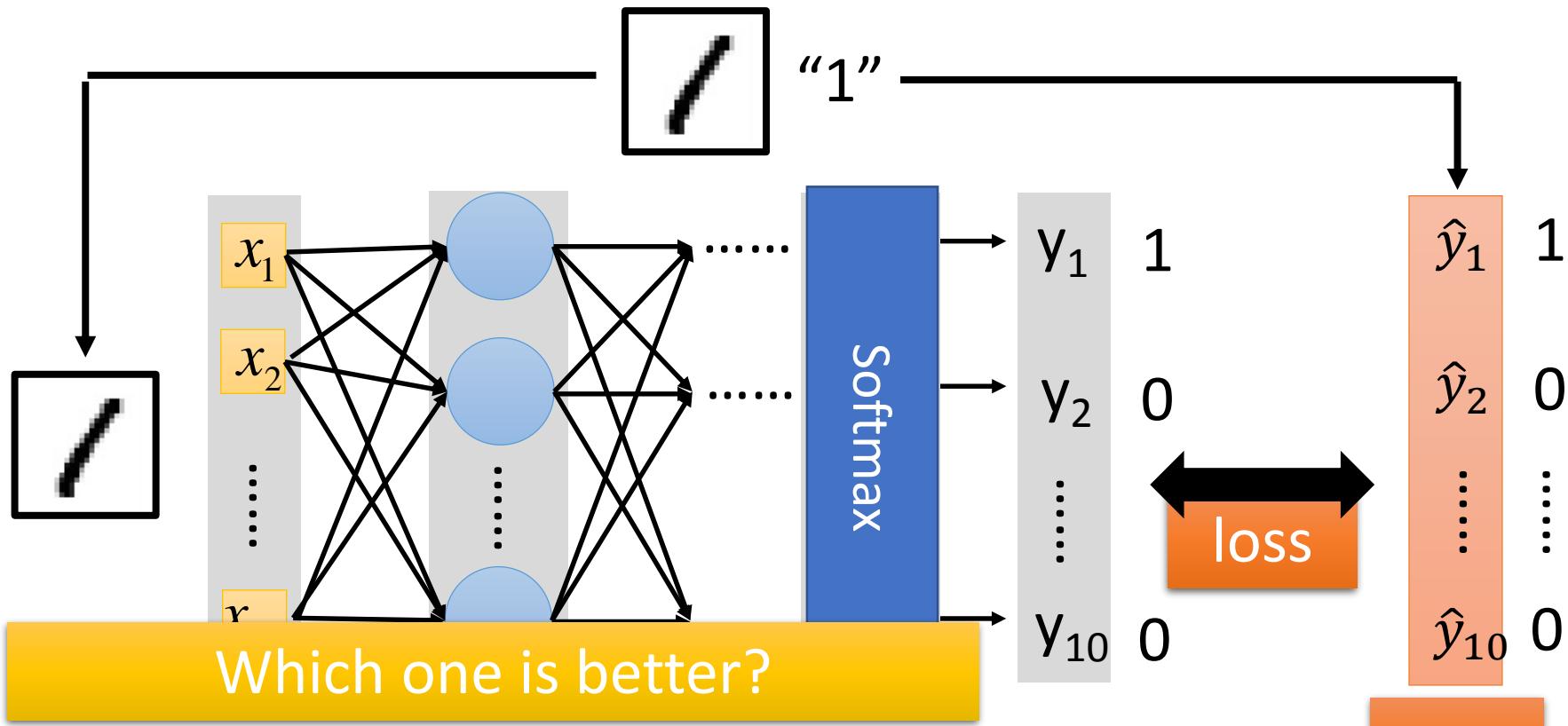
YES



Recipe of Deep Learning



Choosing Proper Loss



Square
Error

$$\sum_{i=1}^{10} (y_i - \hat{y}_i)^2 = 0$$

Cross
Entropy

$$-\sum_{i=1}^{10} \hat{y}_i \ln y_i = 0$$

target

Demo

Square Error

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

Cross Entropy

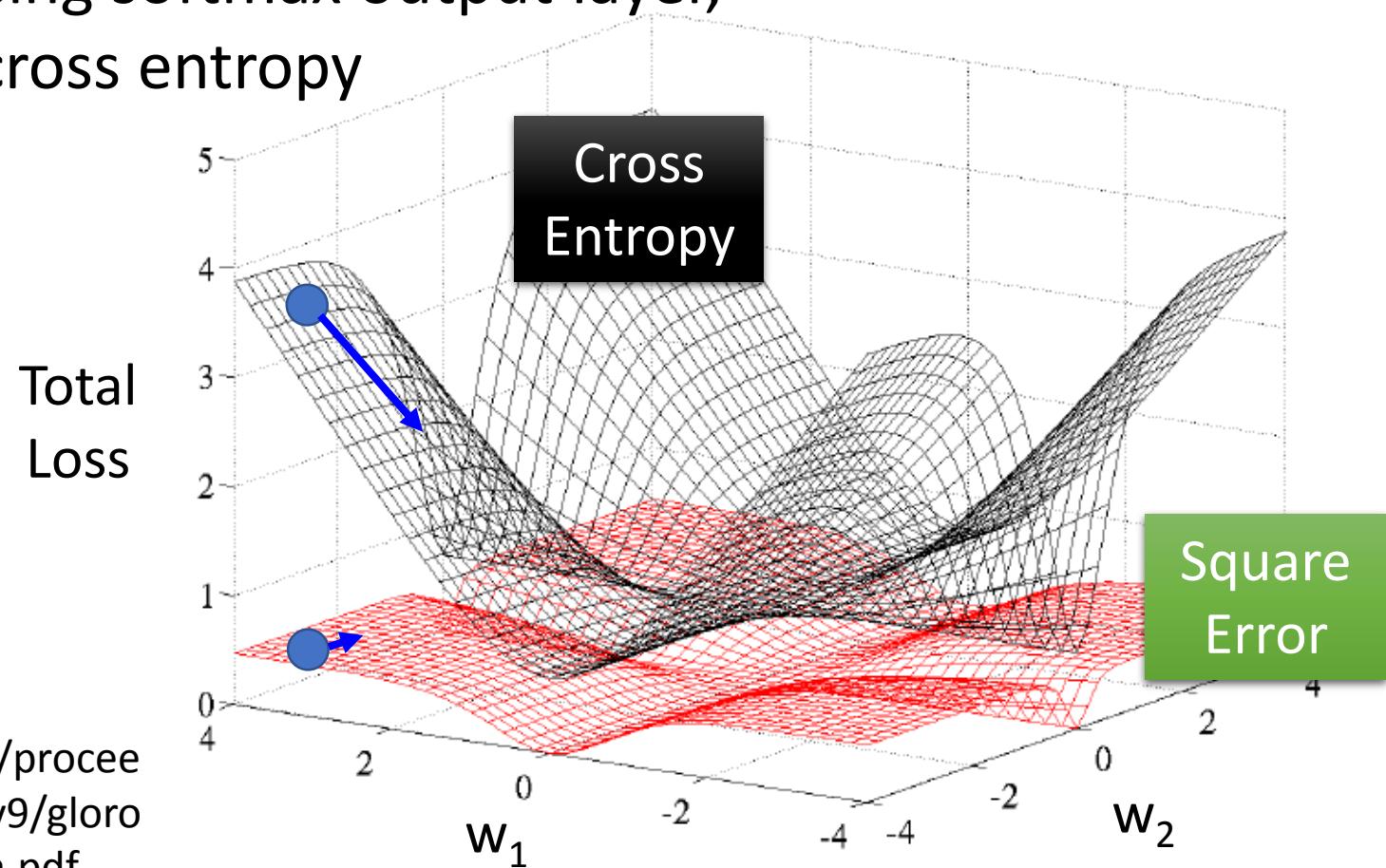
```
model.compile(loss='categorical_crossentropy',  
              optimizer=SGD(lr=0.1),  
              metrics=['accuracy'])
```

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

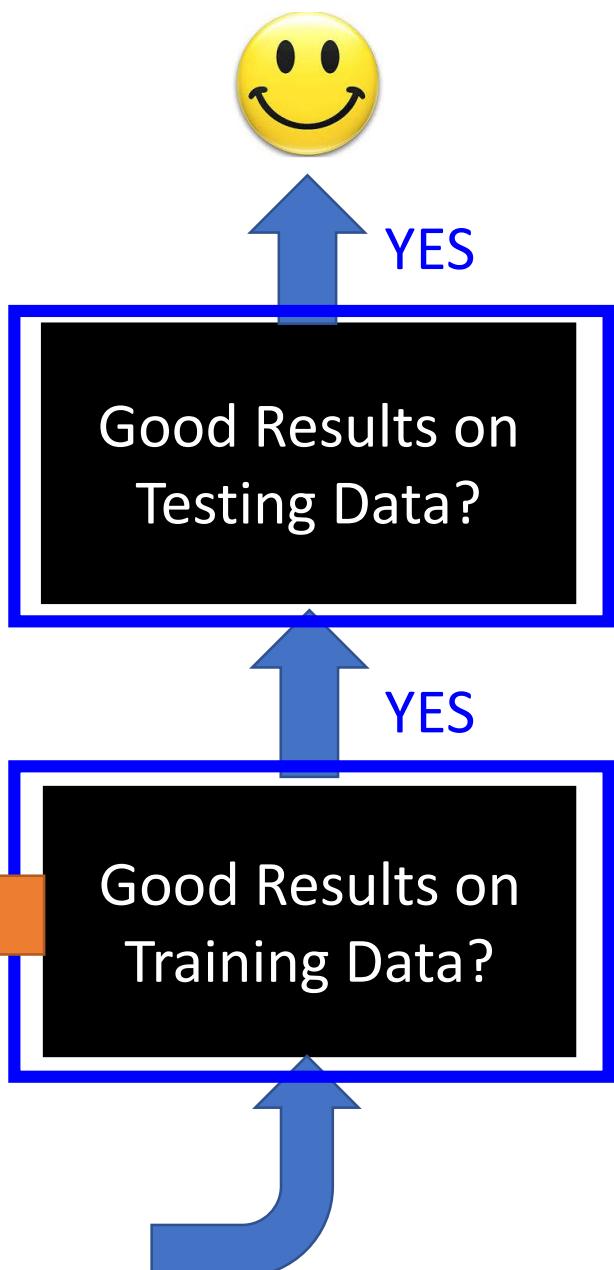
Demo

Choosing Proper Loss

When using softmax output layer,
choose cross entropy



Recipe of Deep Learning

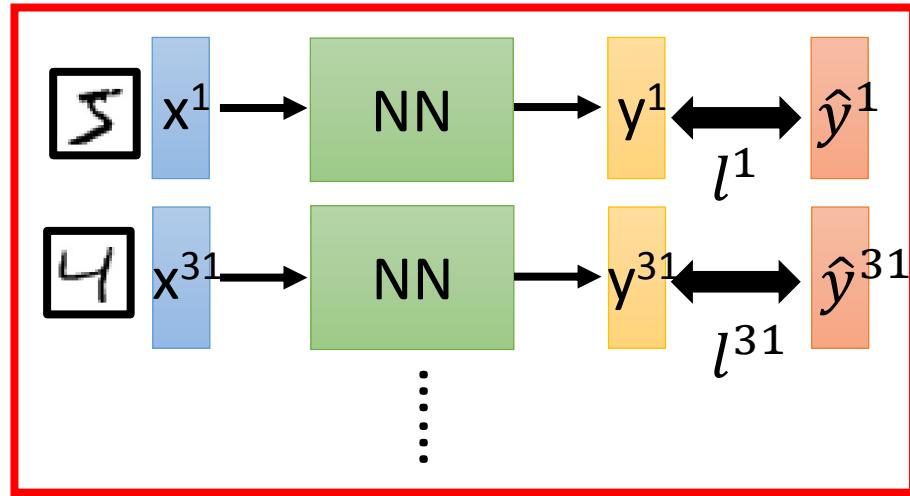


```
model.fit(x_train, y_train, batch_size=100, nb_epoch=20)
```

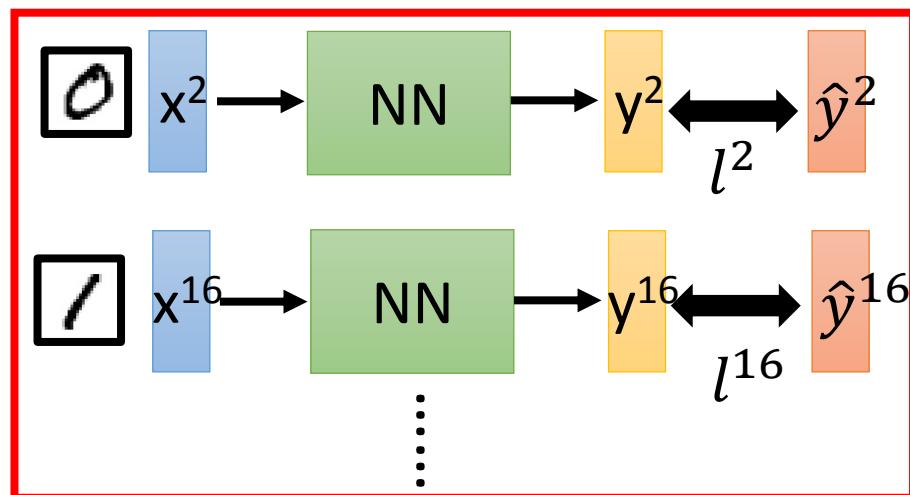
We do not really minimize total loss!

Mini-batch

Mini-batch



Mini-batch



➤ Randomly initialize network parameters

➤ Pick the 1st batch
 $L' = l^1 + l^{31} + \dots$
Update parameters once

➤ Pick the 2nd batch
 $L'' = l^2 + l^{16} + \dots$
Update parameters once

⋮

➤ Until all mini-batches have been picked

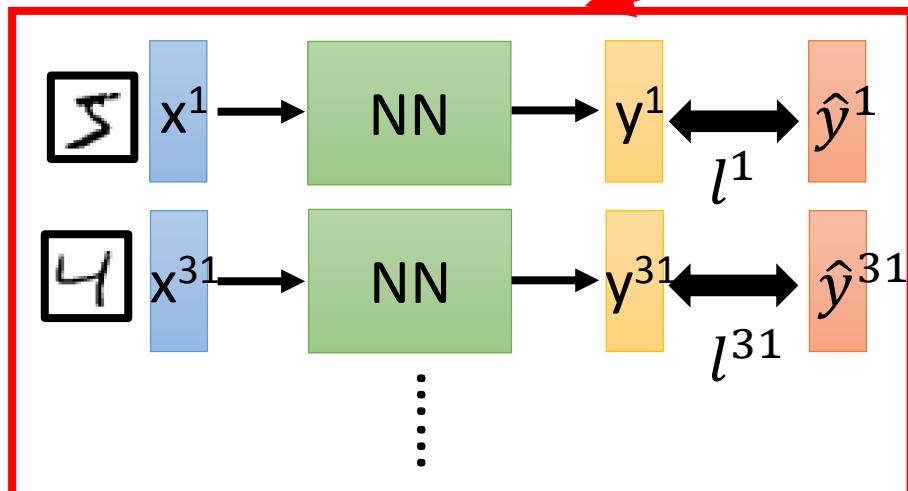
one epoch

Repeat the above process

Mini-batch

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

Mini-batch



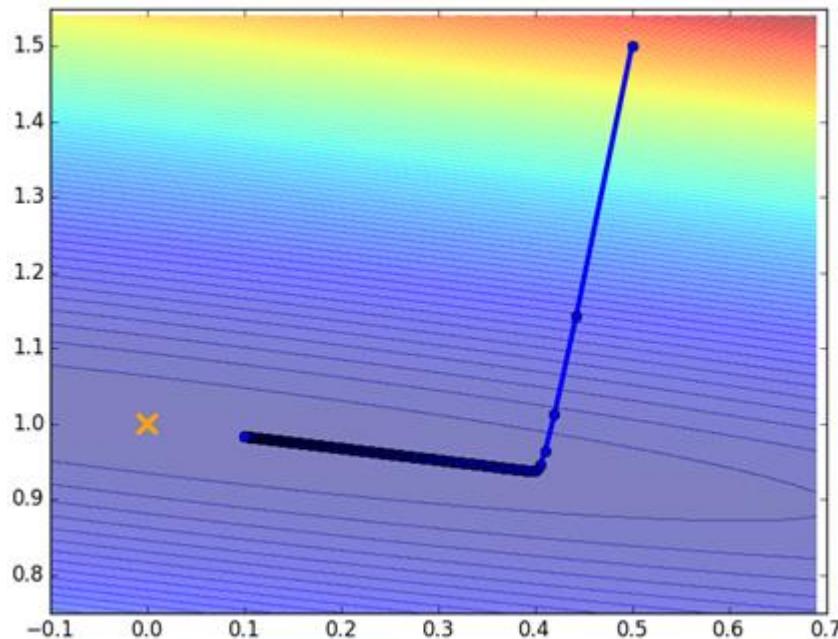
100 examples in a mini-batch

Repeat 20 times

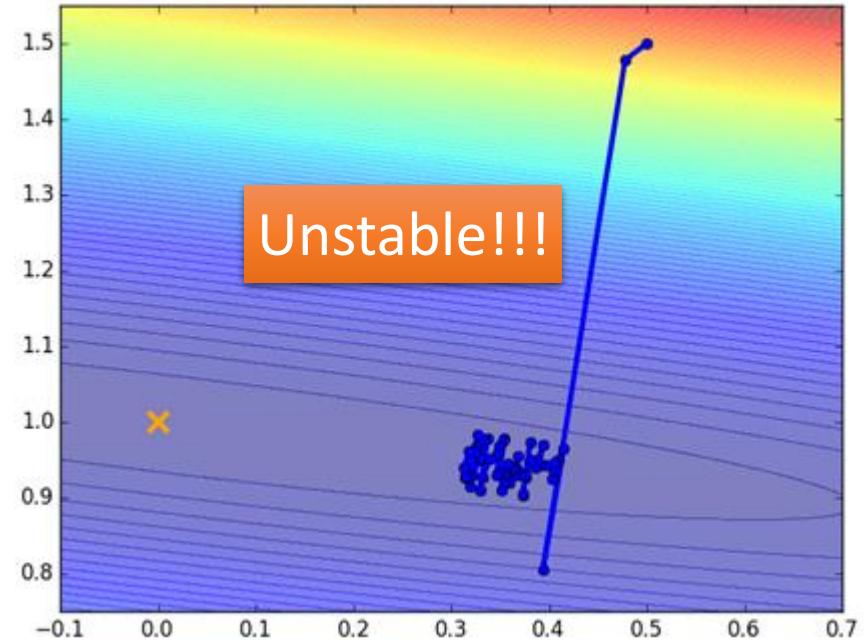
- Pick the 1st batch
 $L' = l^1 + l^{31} + \dots$
Update parameters once
 - Pick the 2nd batch
 $L'' = l^2 + l^{16} + \dots$
Update parameters once
⋮
 - Until all mini-batches
have been picked
- one epoch

Mini-batch

Original Gradient Descent



With Mini-batch



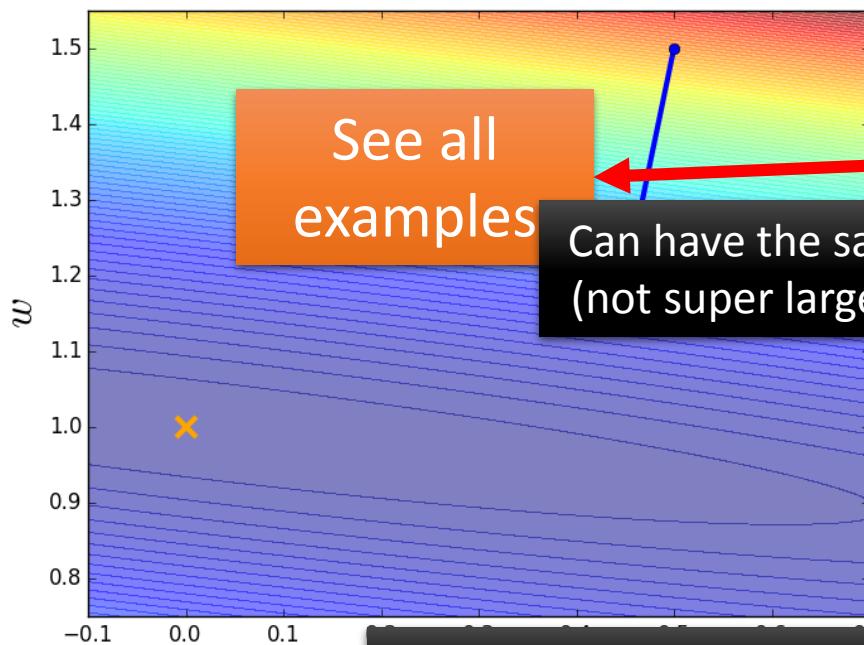
The colors represent the total loss.

Mini-batch is Faster

Not always true with parallel computing.

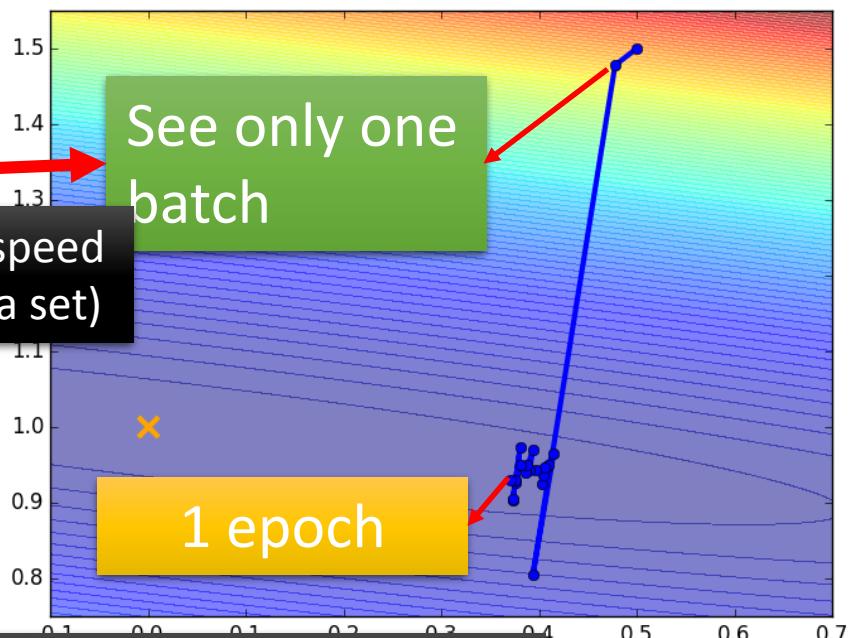
Original Gradient Descent

Update after seeing all examples



With Mini-batch

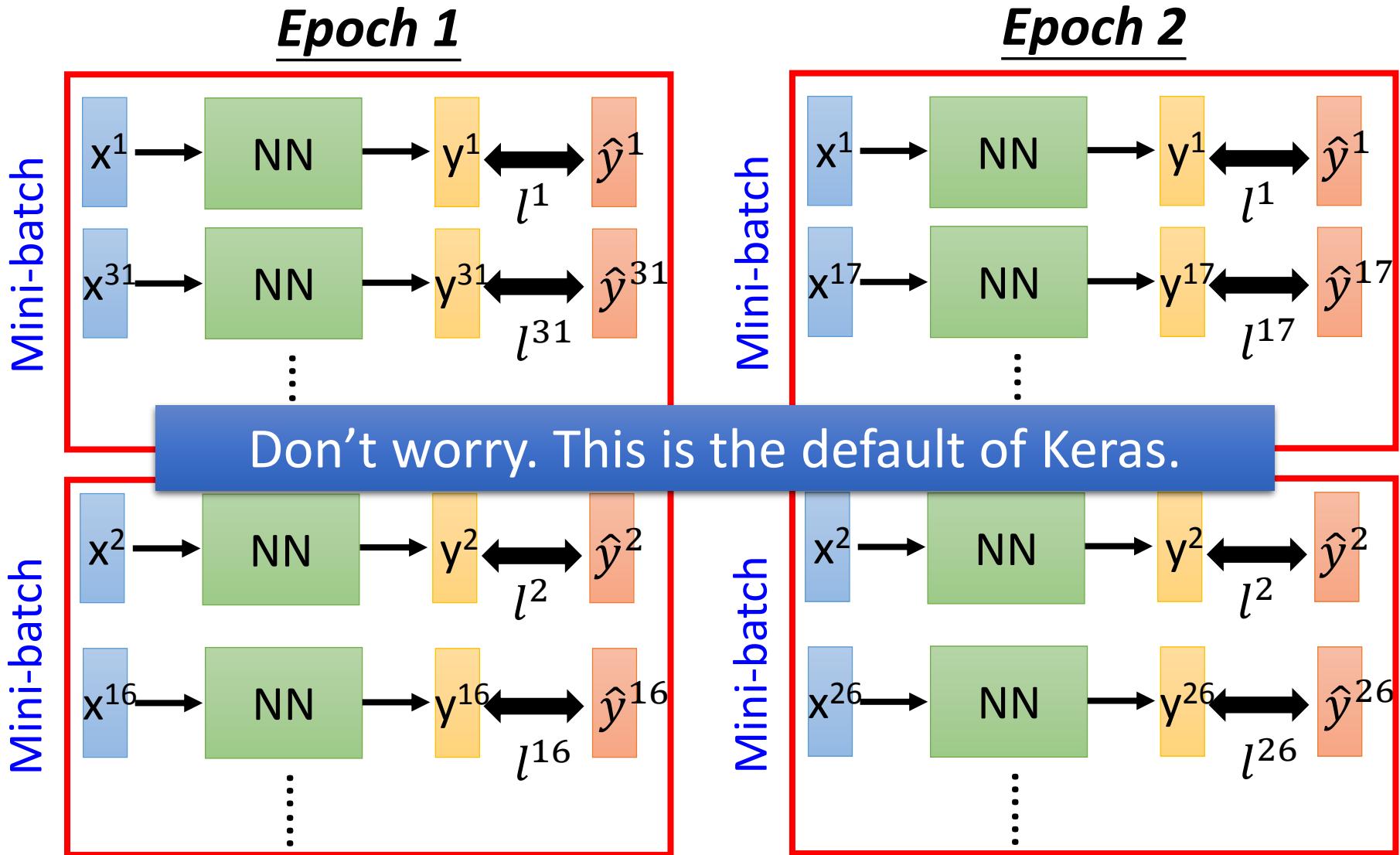
If there are 20 batches, update 20 times in one epoch.



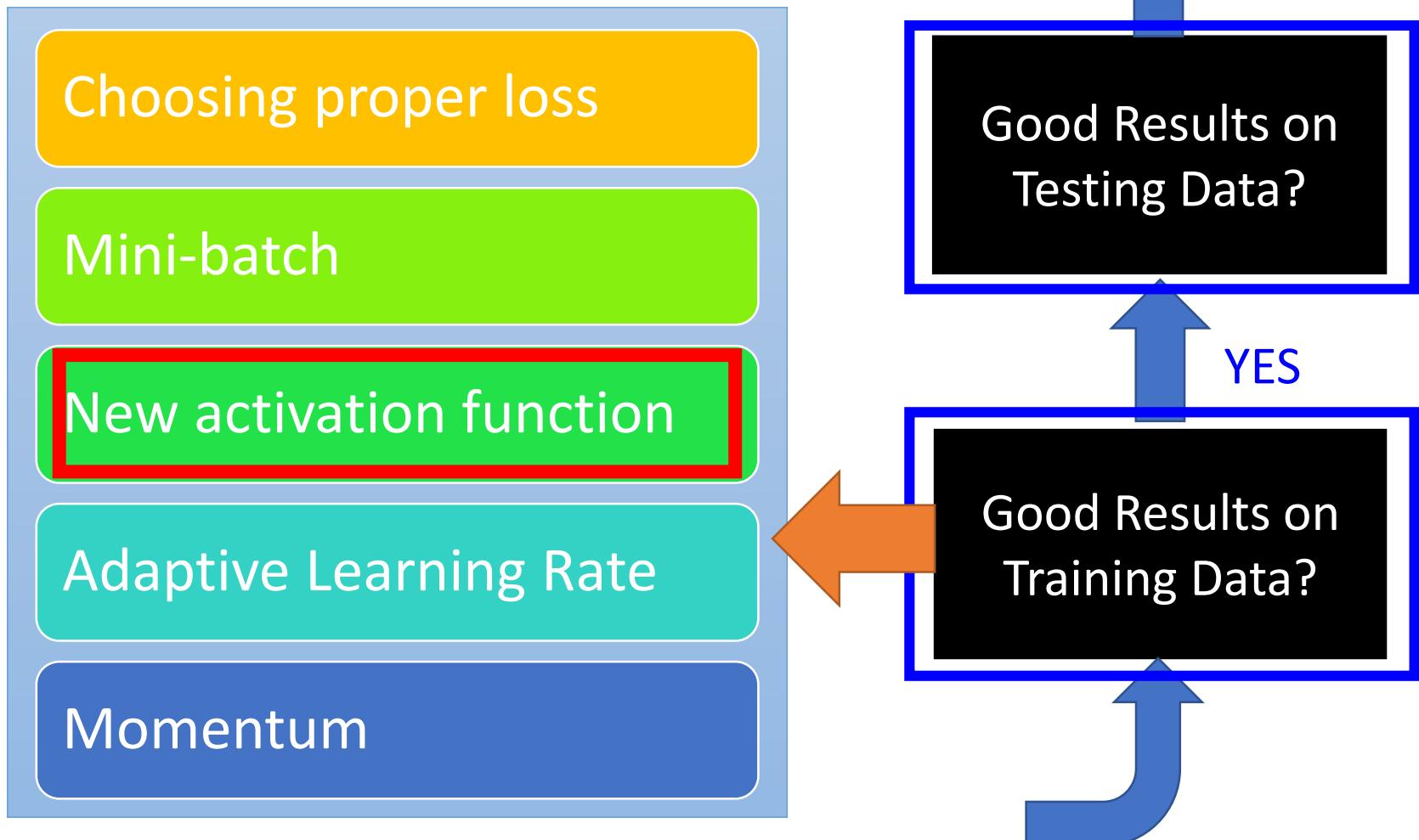
Mini-batch has better performance!

Demo

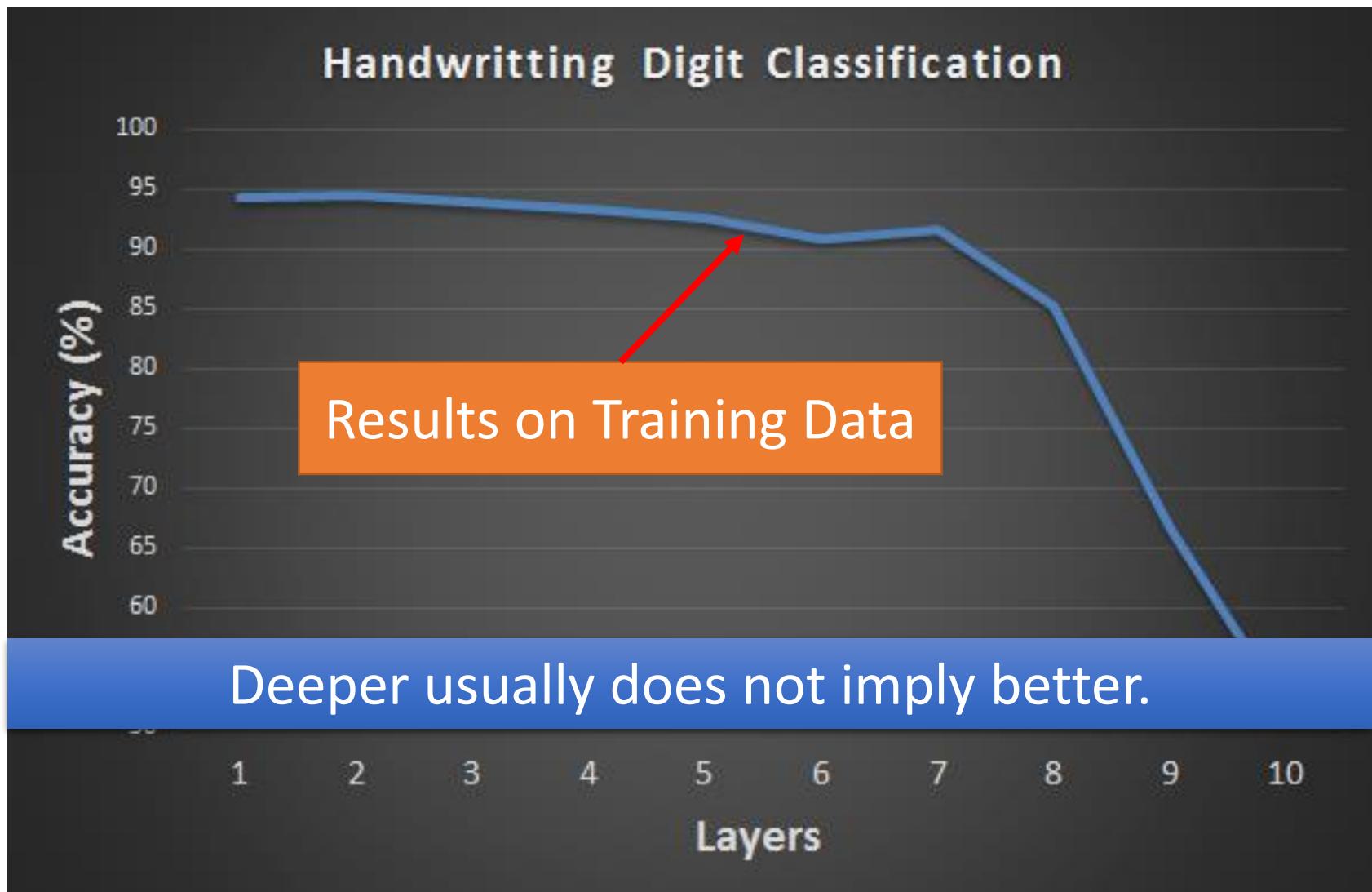
Shuffle the training examples for each epoch



Recipe of Deep Learning

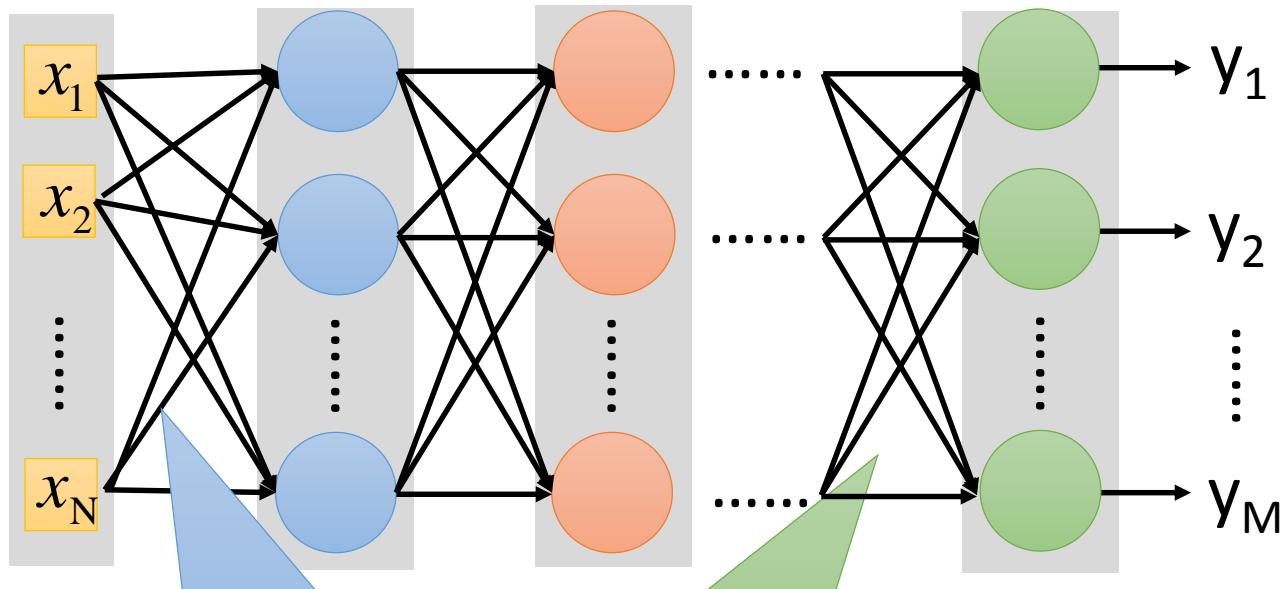


Hard to get the power of Deep ...



Demo

Vanishing Gradient Problem



Smaller gradients

Learn very slow

Almost random

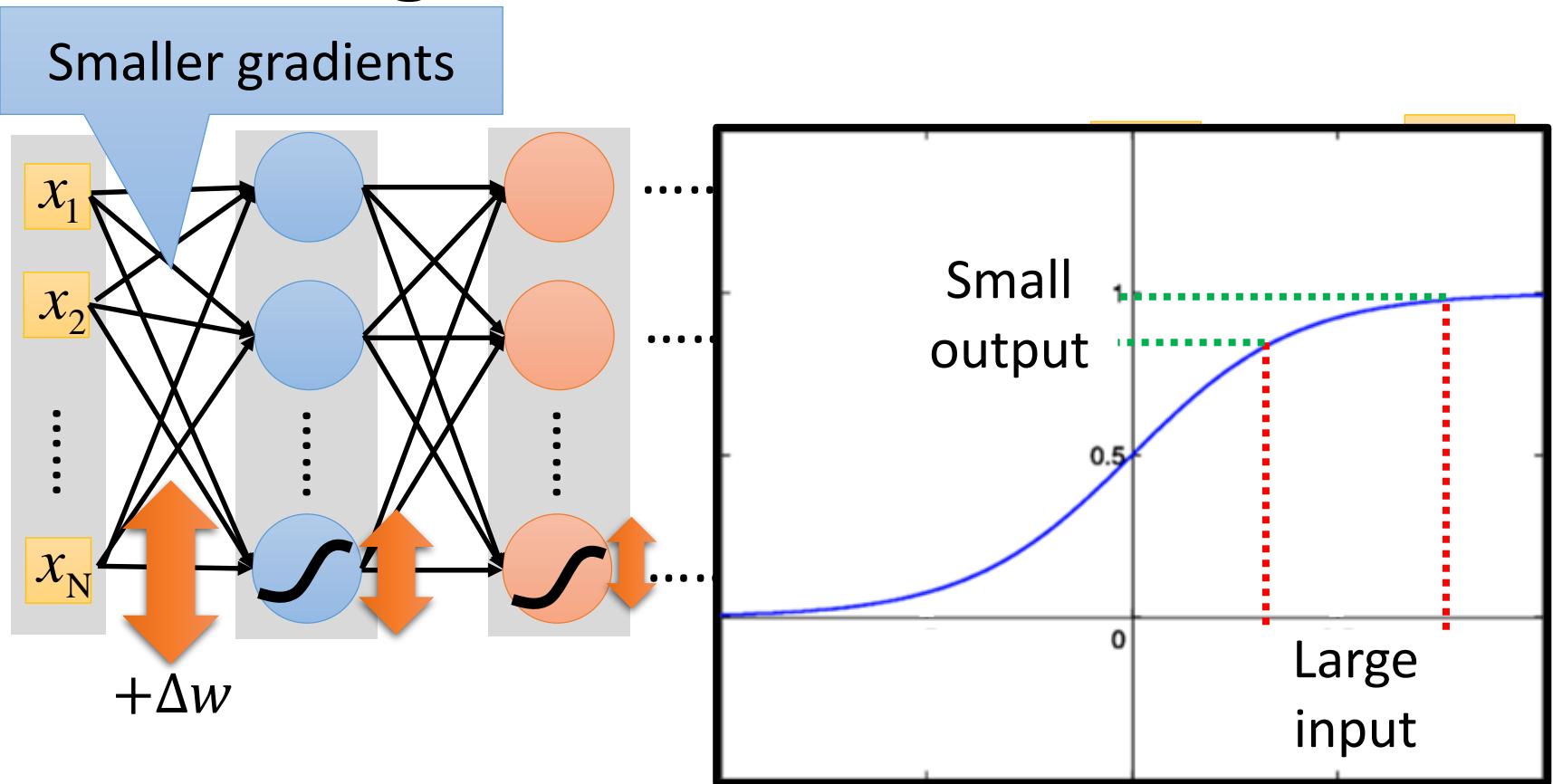
Larger gradients

Learn very fast

Already converge

based on random!?

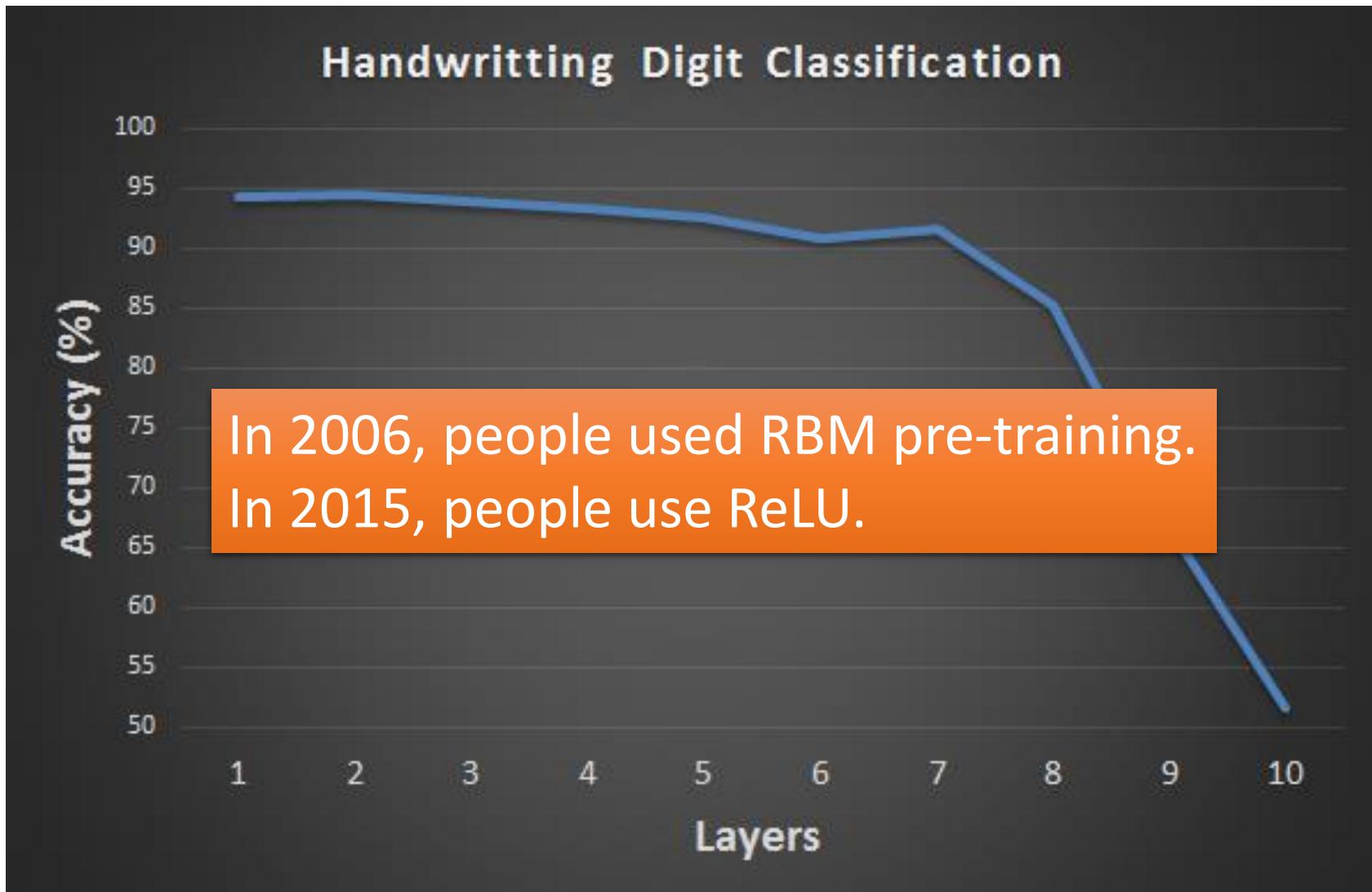
Vanishing Gradient Problem



Intuitive way to compute the derivatives ...

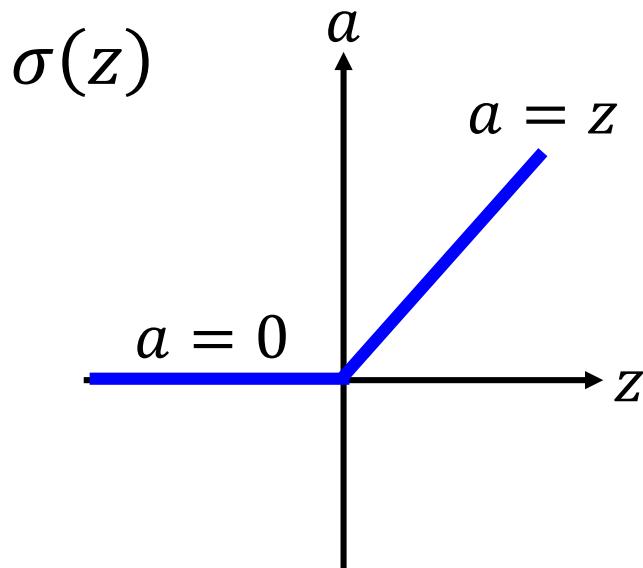
$$\frac{\partial l}{\partial w} = ? \frac{\Delta l}{\Delta w}$$

Hard to get the power of Deep ...



ReLU

- Rectified Linear Unit (ReLU)

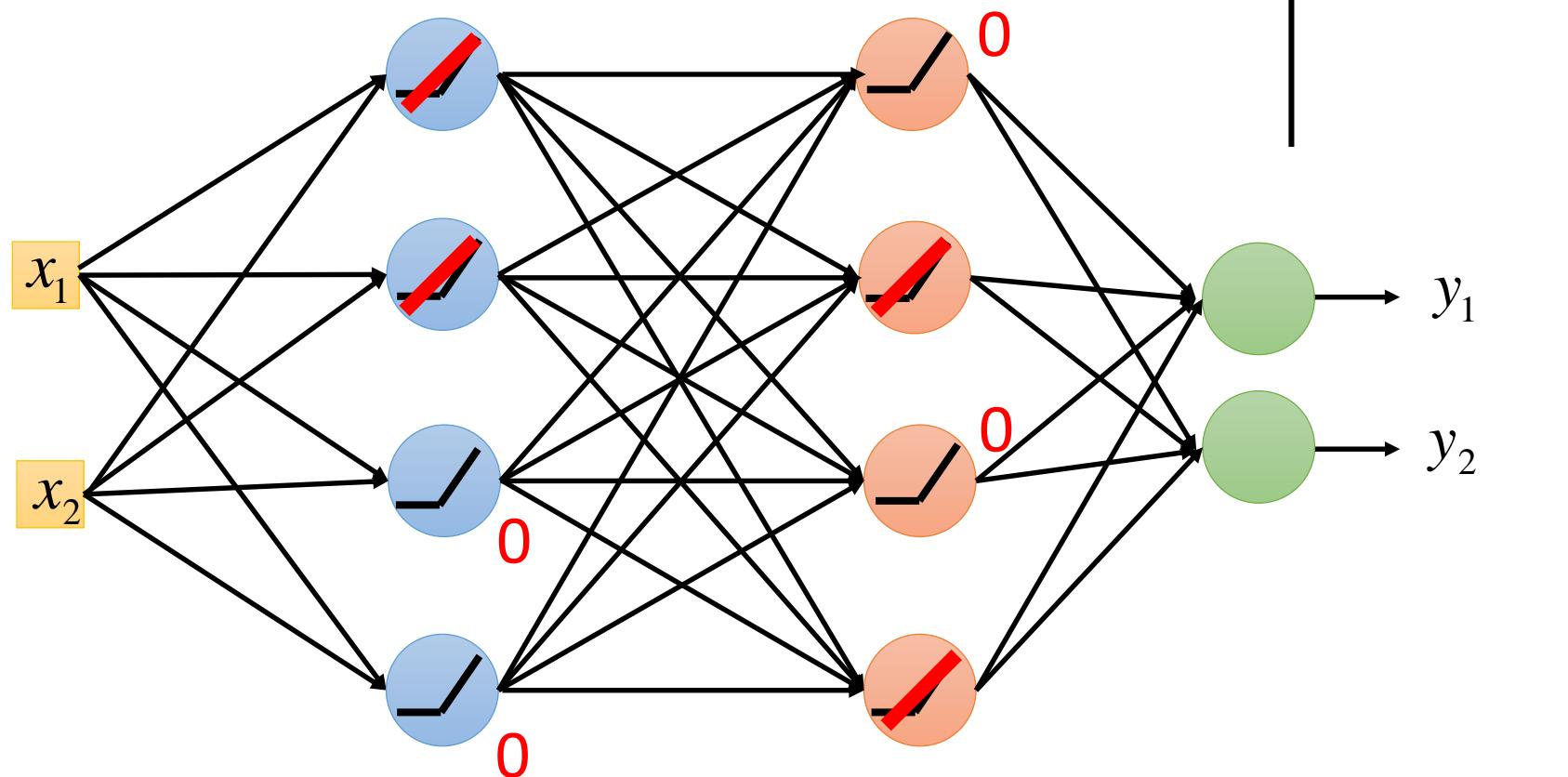


[Xavier Glorot, AISTATS'11]
[Andrew L. Maas, ICML'13]
[Kaiming He, arXiv'15]

Reason:

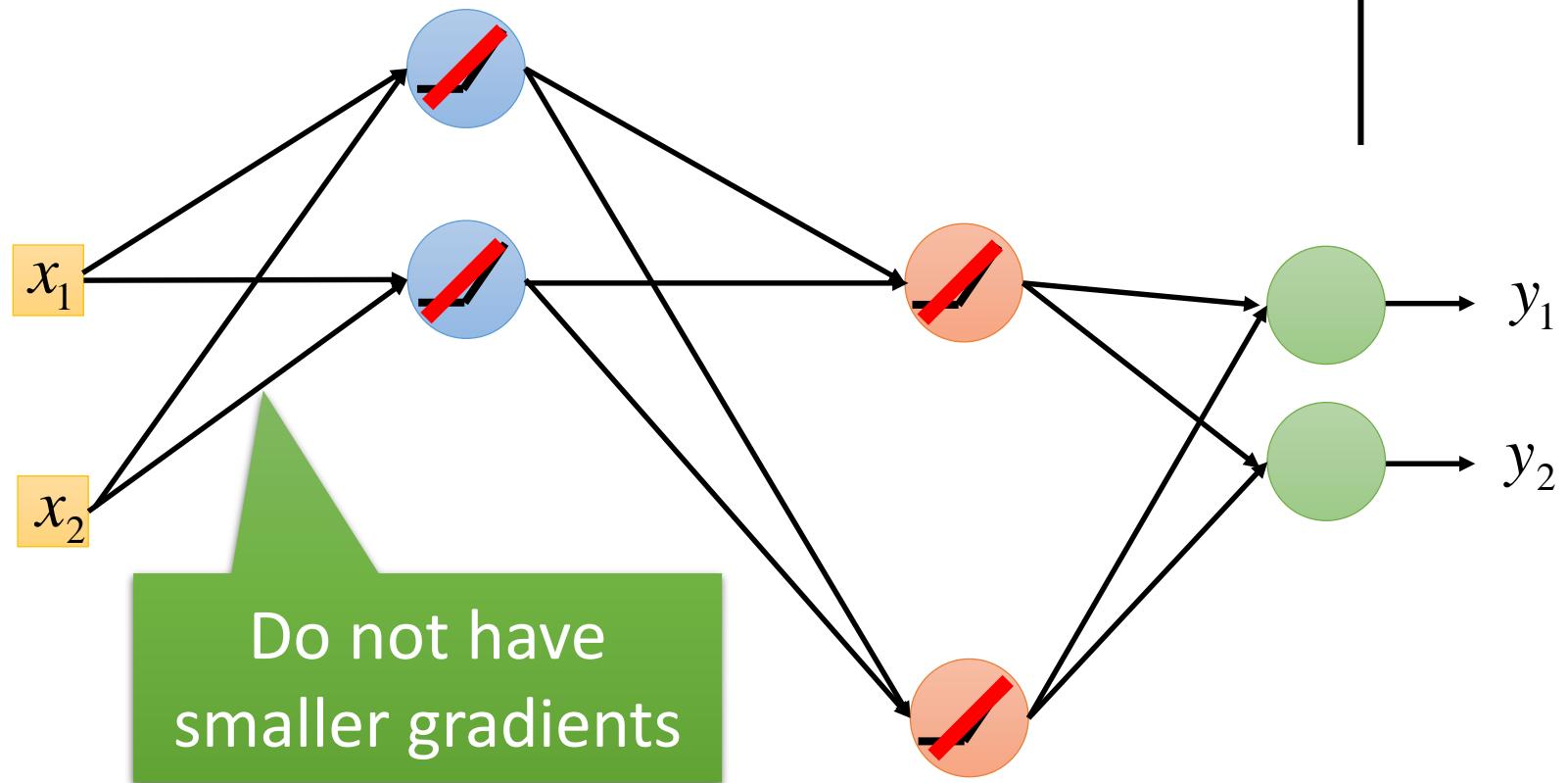
1. Fast to compute
2. Biological reason
3. Infinite sigmoid with different biases
4. Vanishing gradient problem

ReLU



ReLU

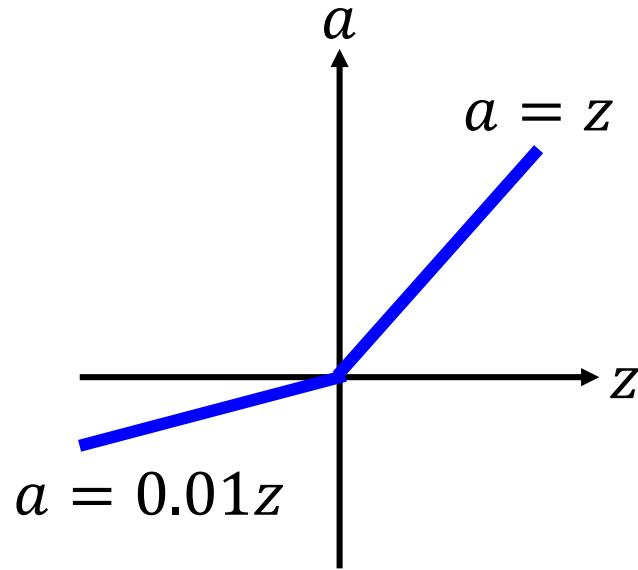
A Thinner linear network



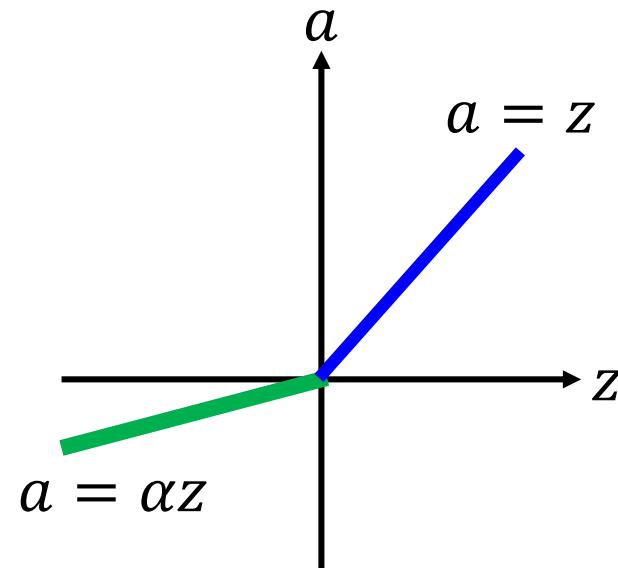
Demo

ReLU - variant

Leaky ReLU



Parametric ReLU

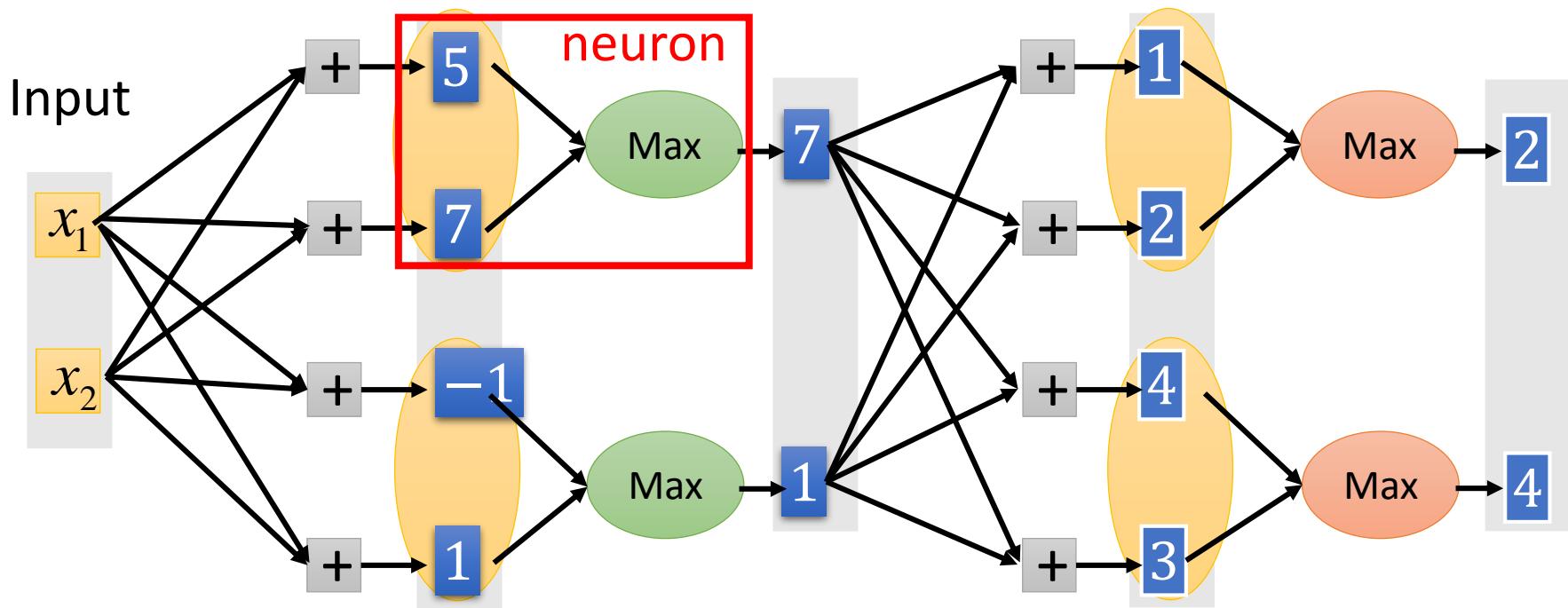


α also learned by
gradient descent

Maxout

ReLU is a special cases of Maxout

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



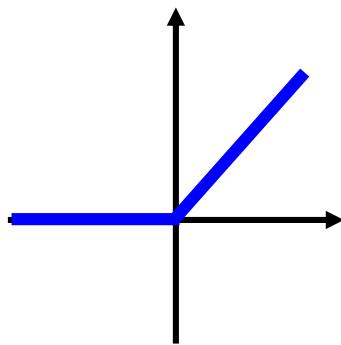
You can have more than 2 elements in a group.

Maxout

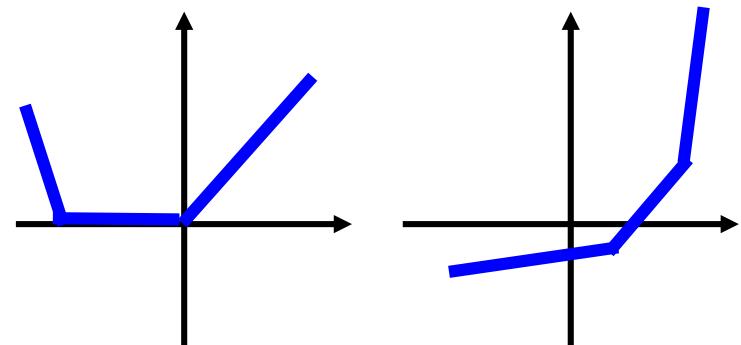
ReLU is a special cases of Maxout

- 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

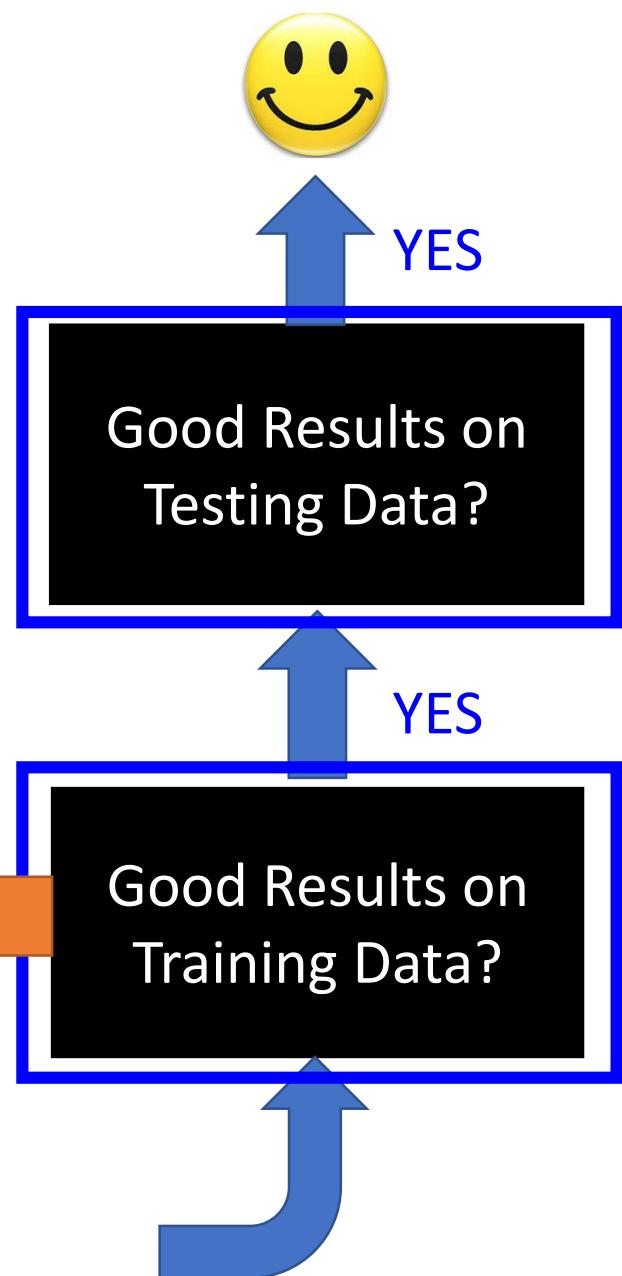
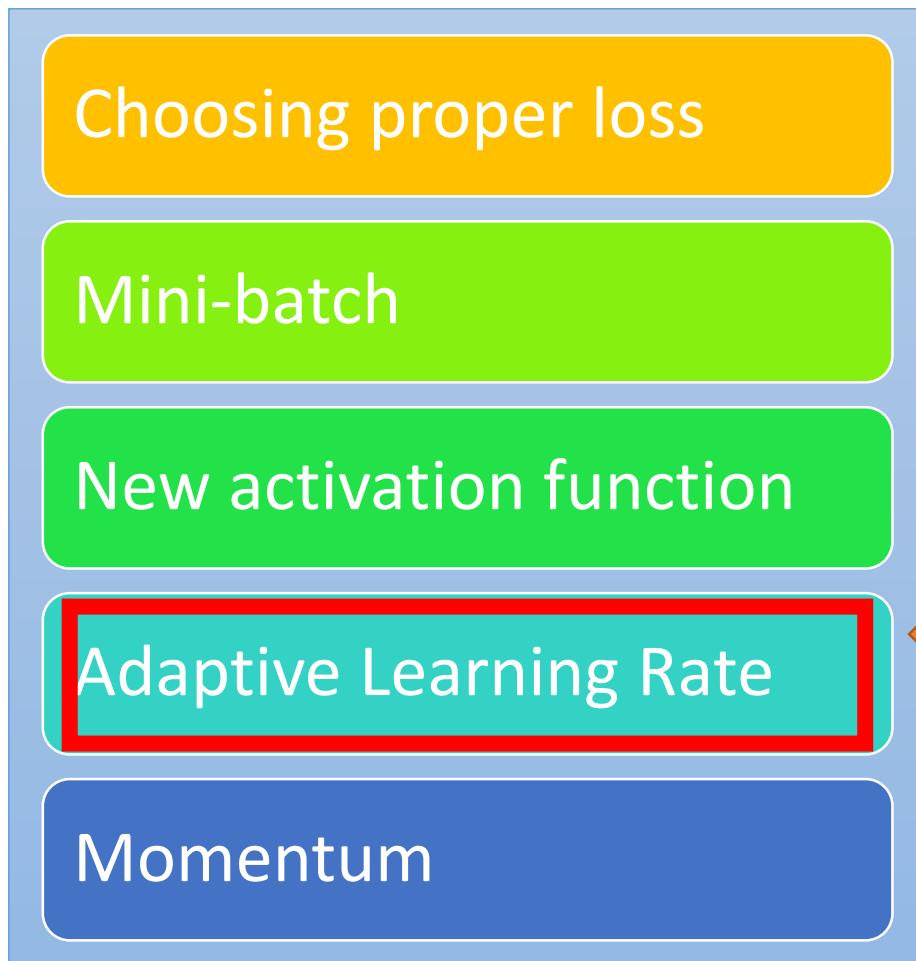
2 elements in a group



3 elements in a group

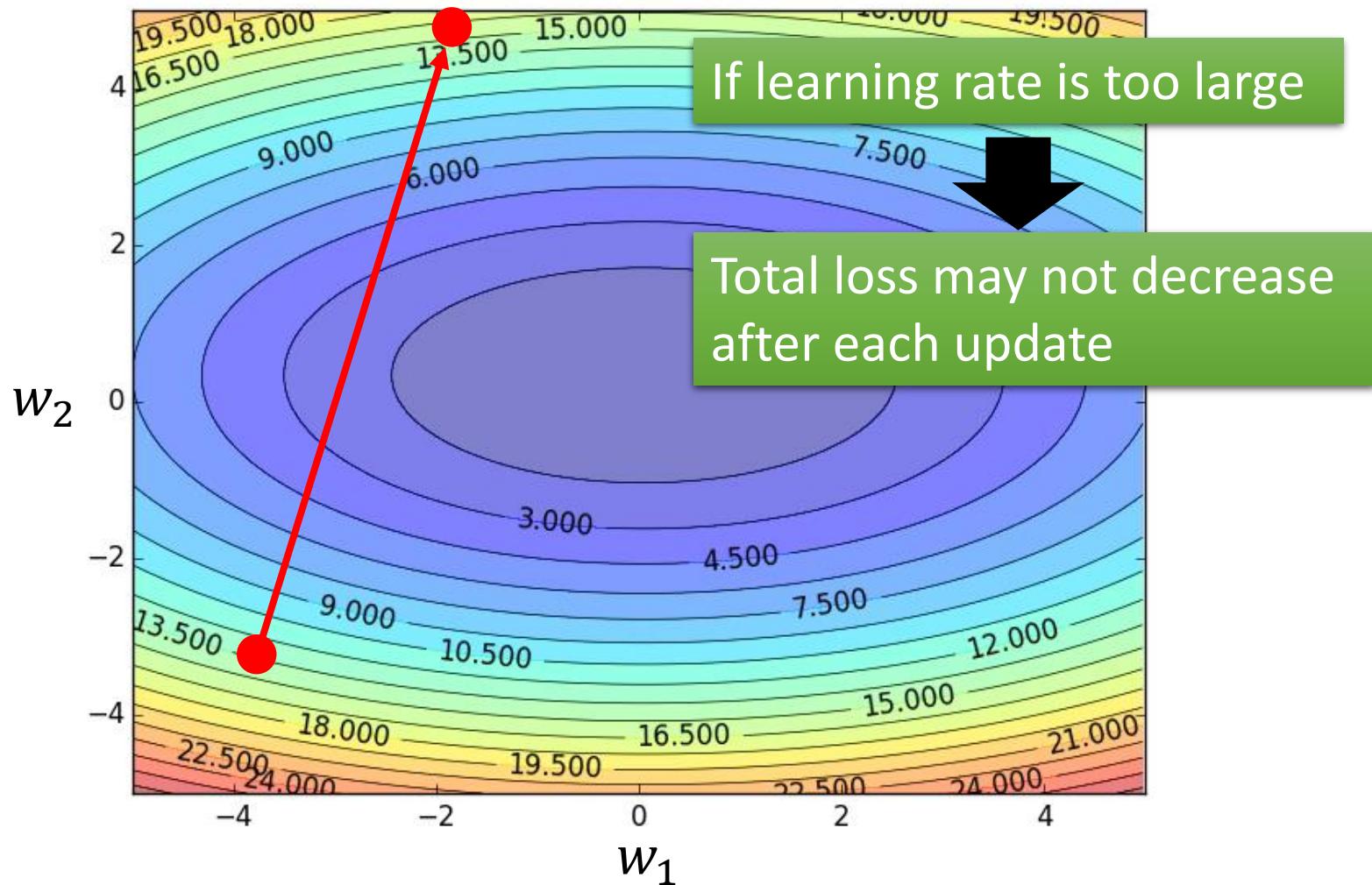


Recipe of Deep Learning



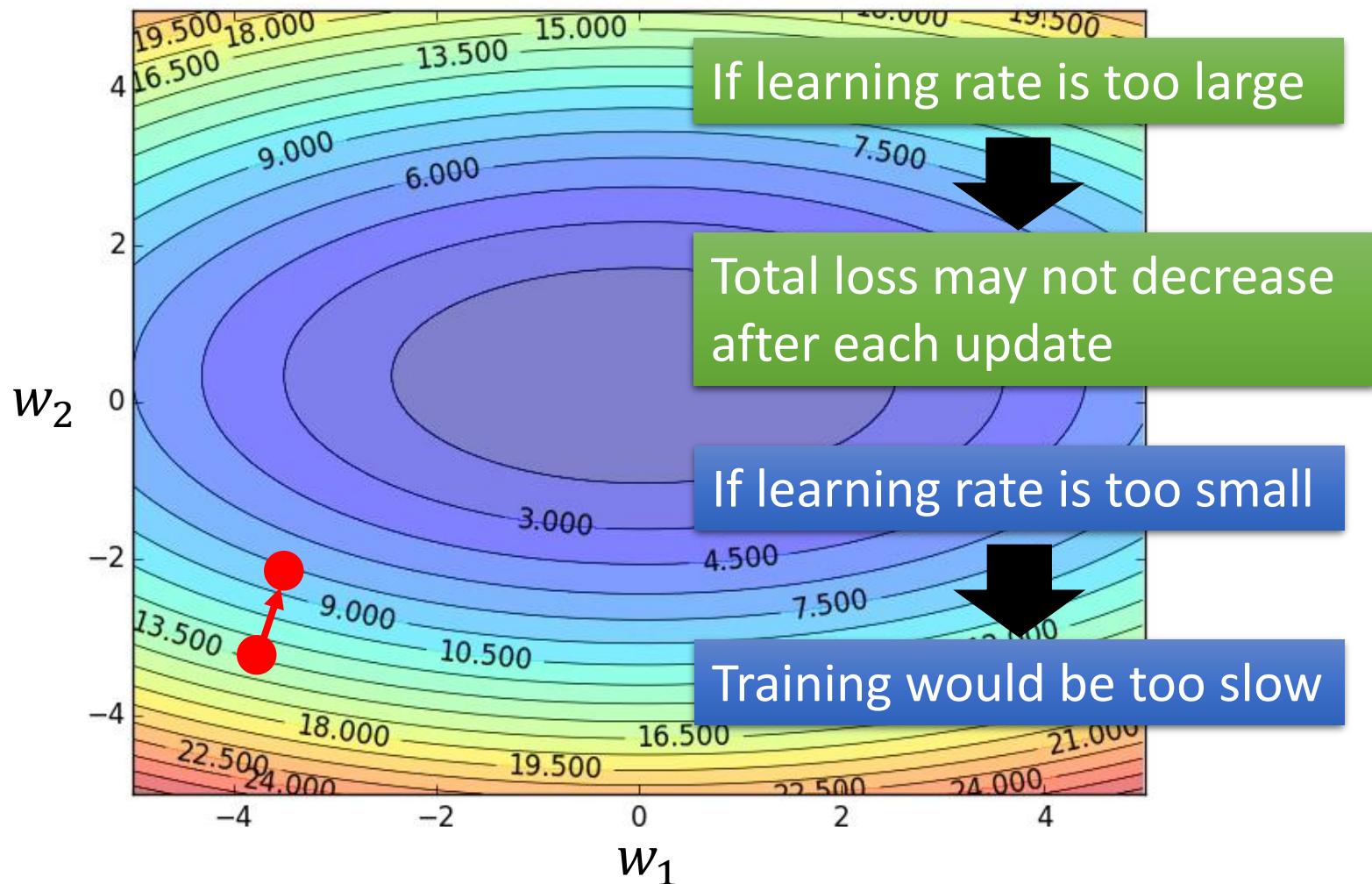
Learning Rates

Set the learning rate η carefully



Learning Rates

Set the learning rate η carefully



Learning Rates

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

Adagrad

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

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

Parameter dependent learning rate

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

η  constant

$\sqrt{\sum_{i=0}^t (g^i)^2}$  g^i is $\partial L / \partial w$ obtained at the i-th update

Summation of the square of the previous derivatives

Adagrad

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

w_1	g^0
	0.1

Learning rate:

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

$$= \frac{\eta}{0.1}$$

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

$$= \frac{\eta}{0.22}$$



w_2	g^0
	20.0

Learning rate:

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

$$= \frac{\eta}{20}$$

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

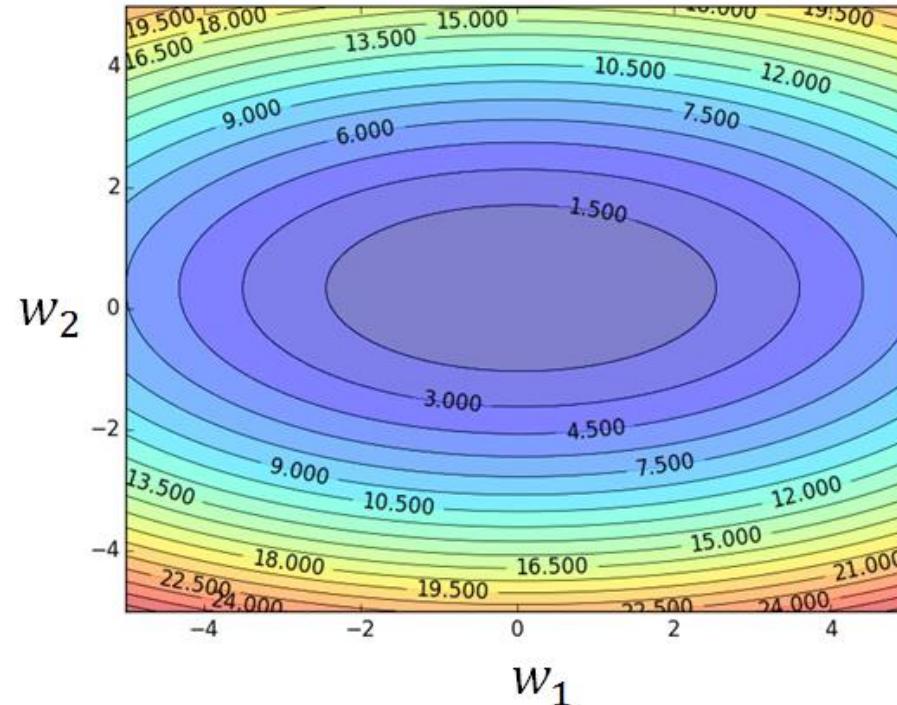
$$= \frac{\eta}{22}$$

- Observation:**
1. Learning rate is smaller and smaller for all parameters
 2. Smaller derivatives, larger learning rate, and vice versa

Why?

Larger derivatives

Smaller Learning Rate



Smaller Derivatives

Larger Learning Rate

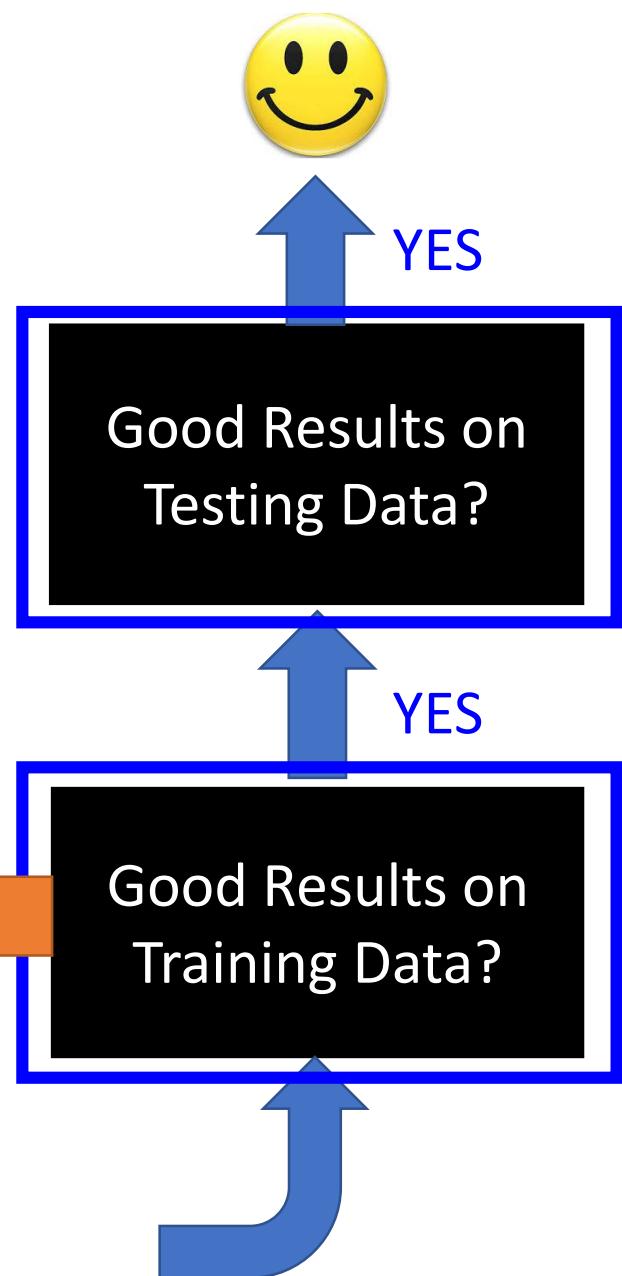
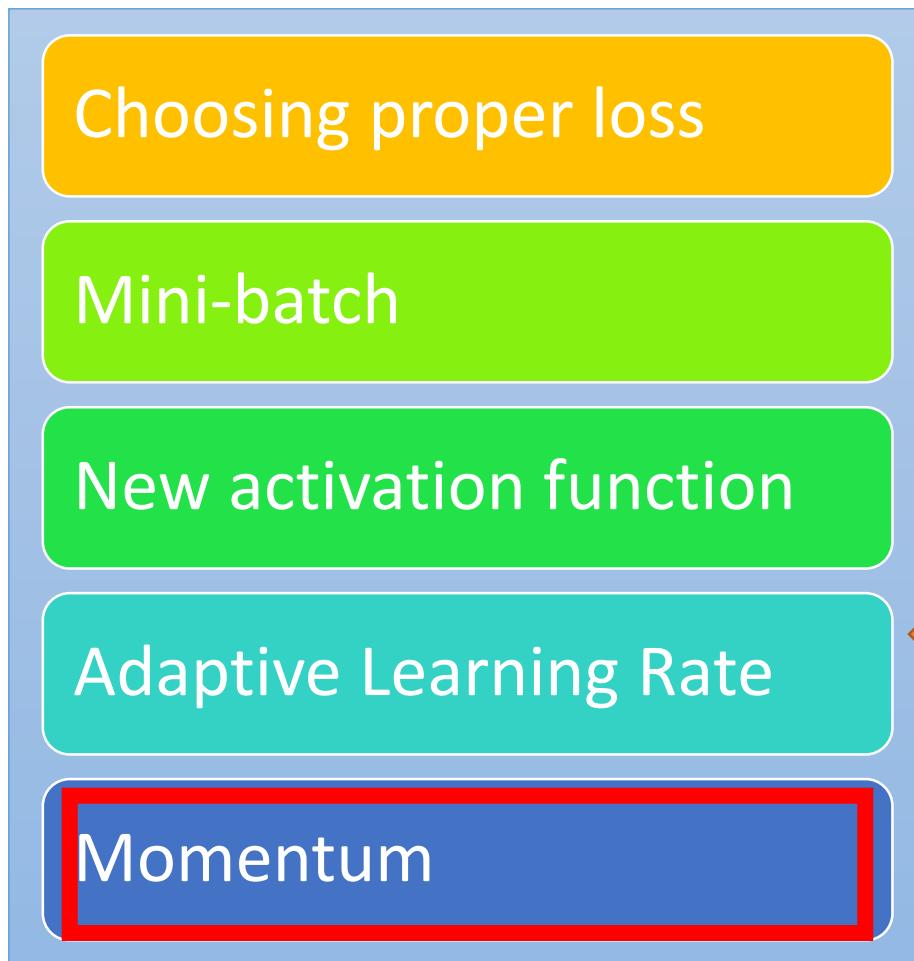
2. Smaller derivatives, larger learning rate, and vice versa

Why?

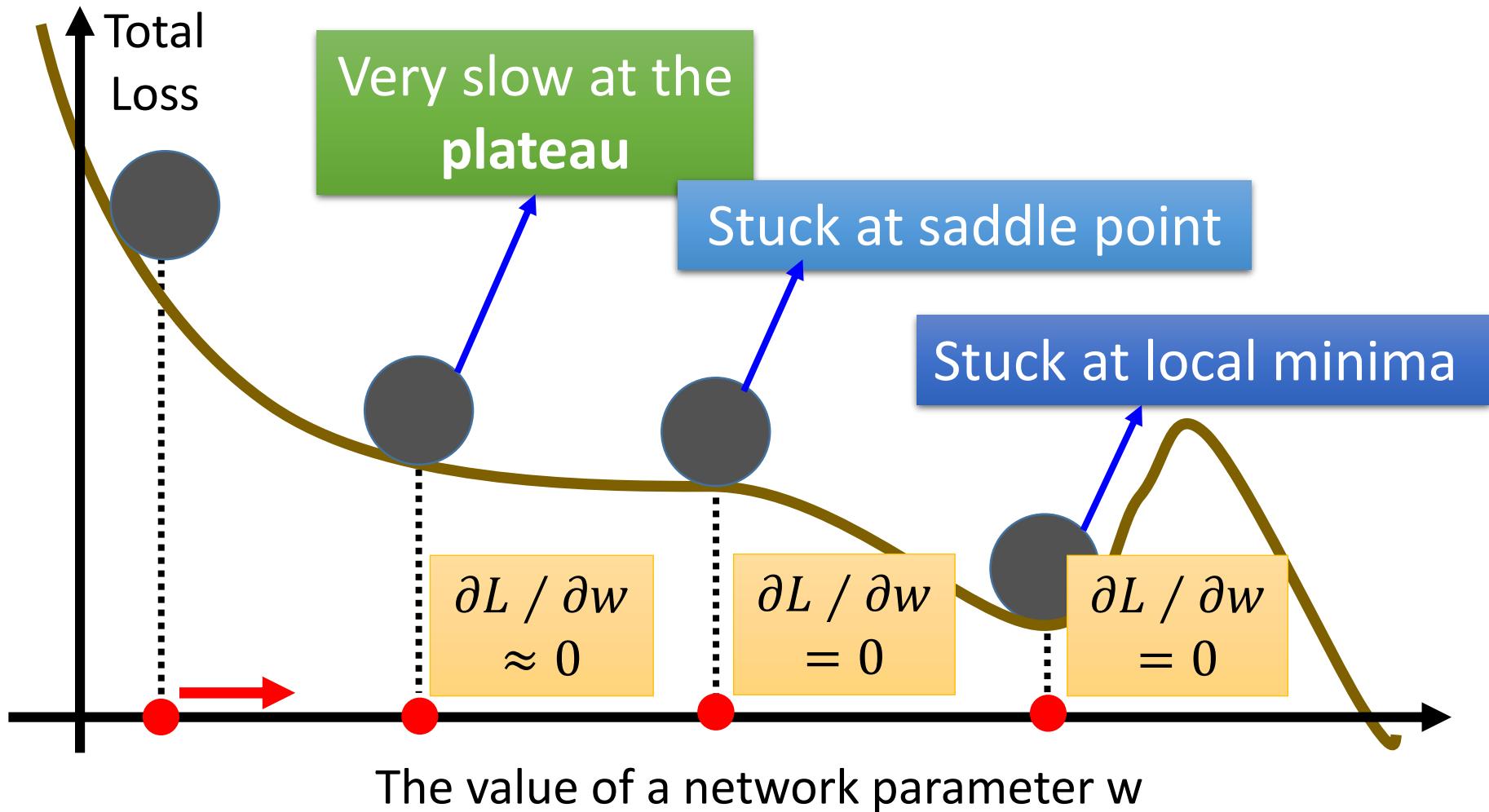
Not the whole story

- 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

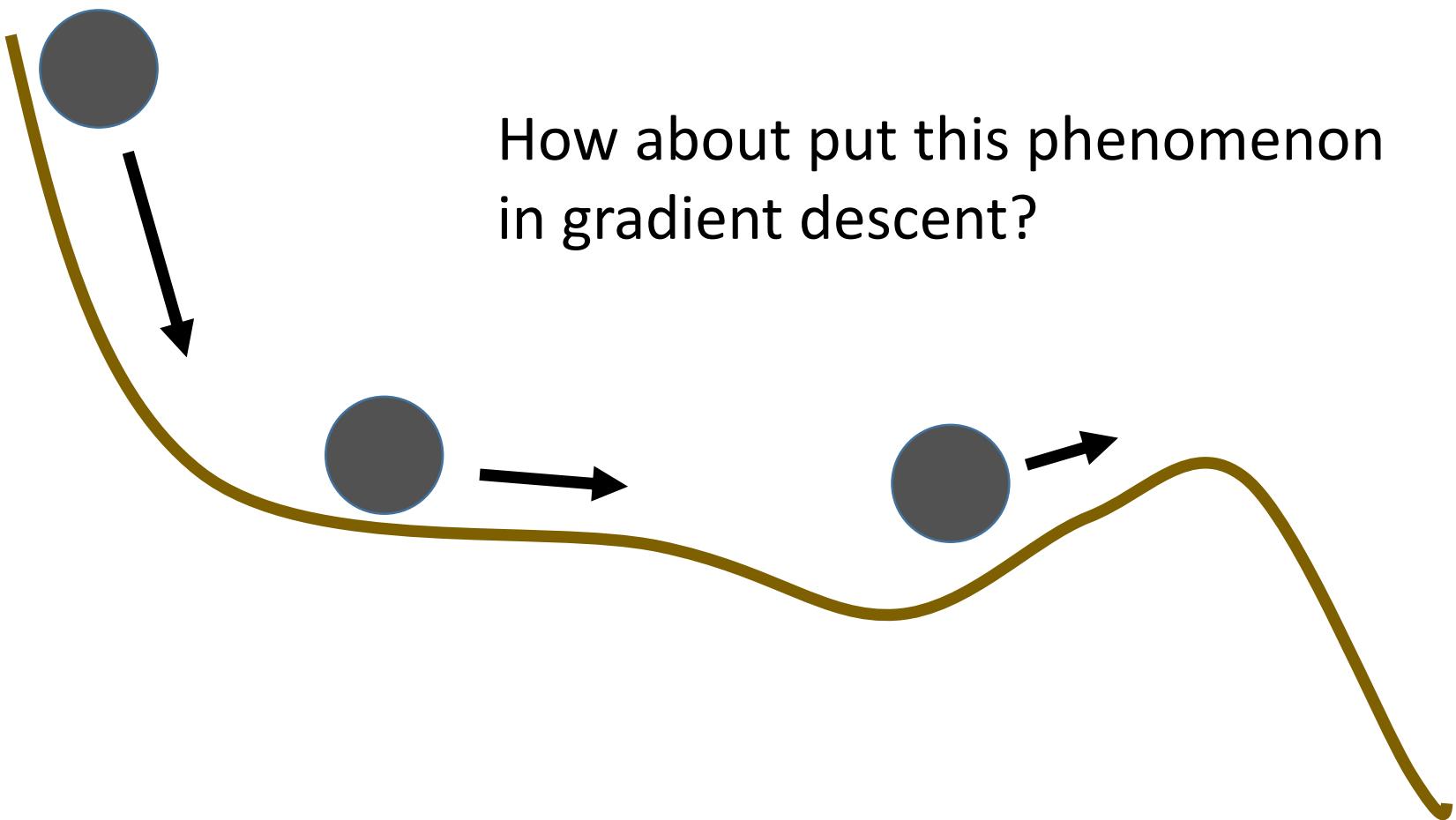


Hard to find optimal network parameters



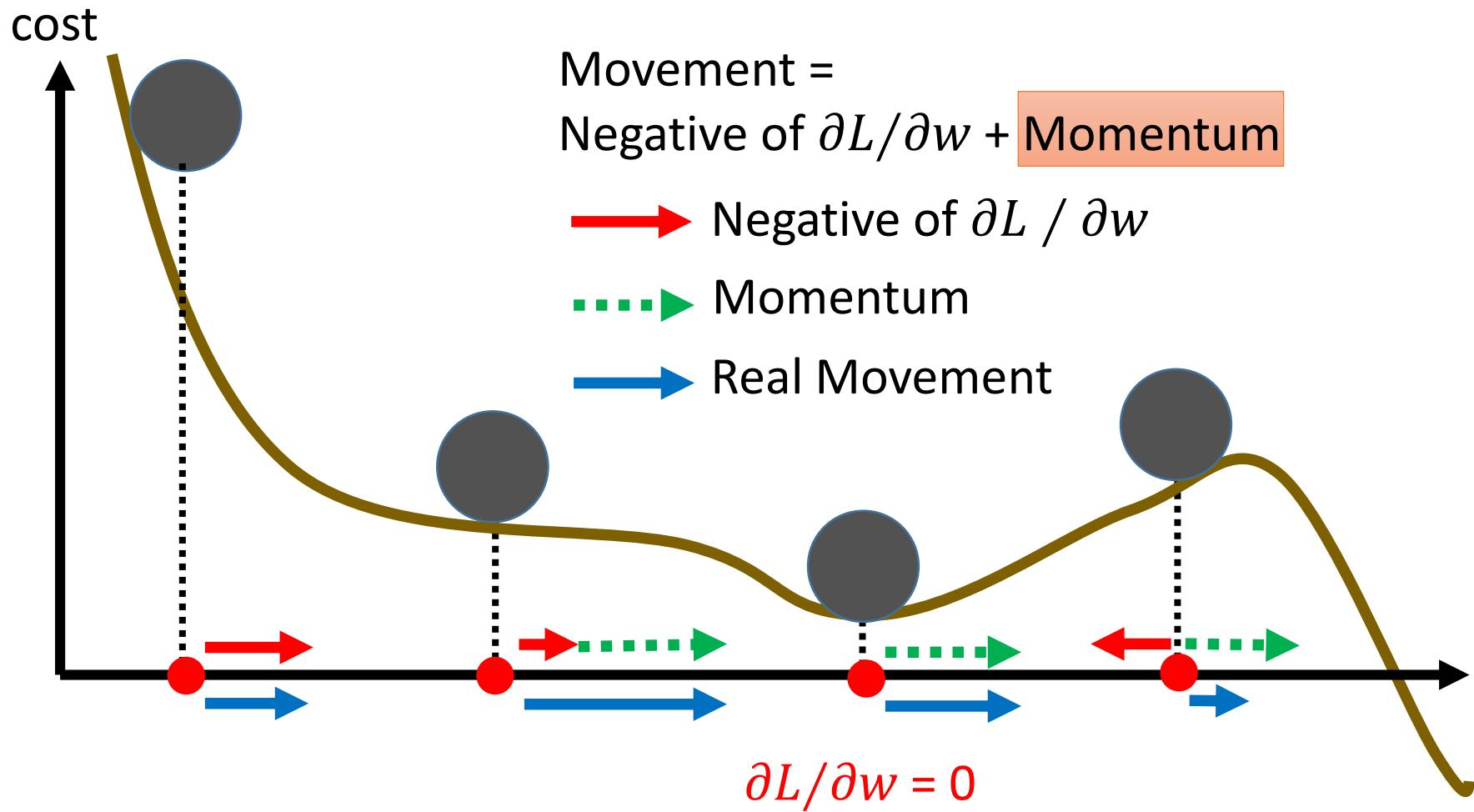
In physical world

- Momentum



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'])
```

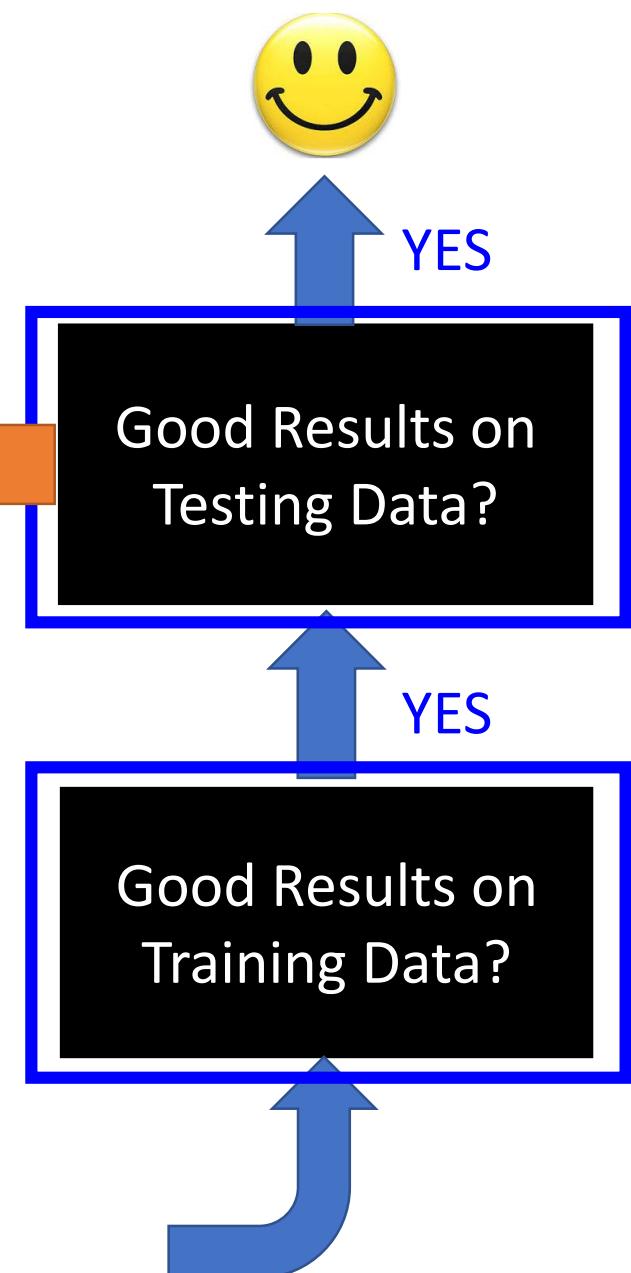
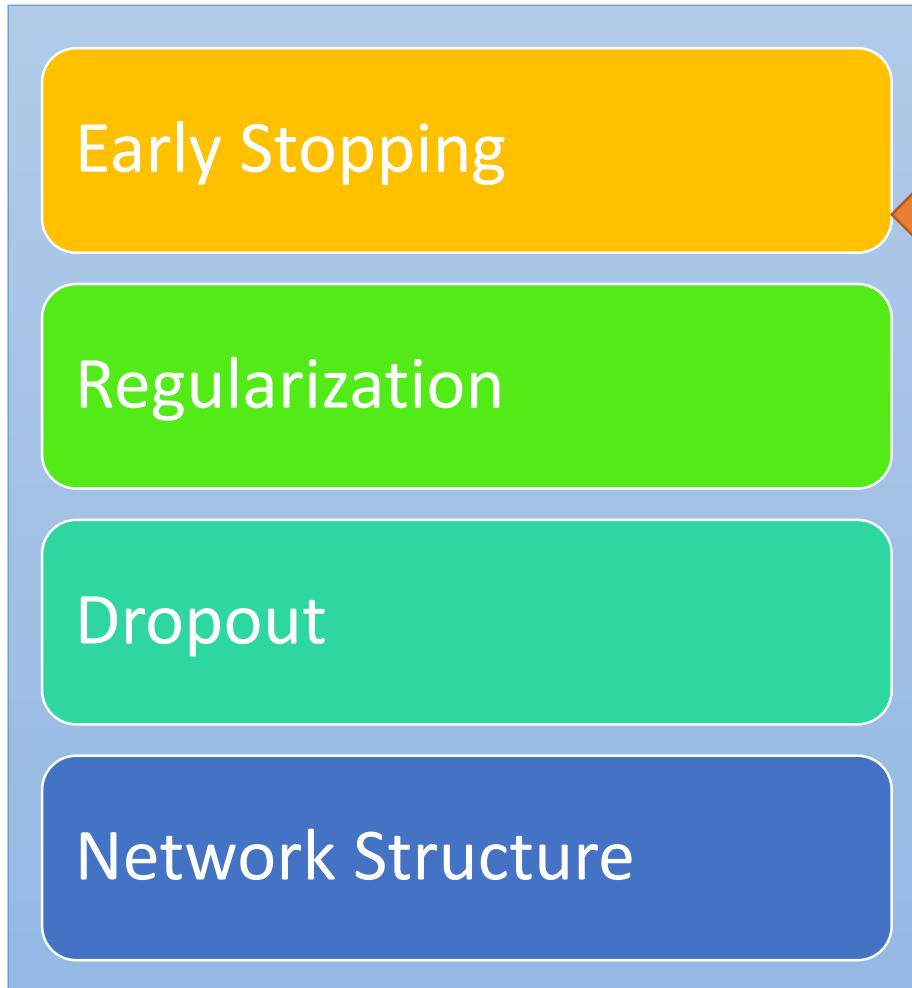
```
model.compile(loss='categorical_crossentropy',  
              optimizer=Adam(),  
              metrics=['accuracy'])
```

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)
 $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
 $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
 $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$ (Update parameters)
end while
return θ_t (Resulting parameters)

Demo

Recipe of Deep Learning

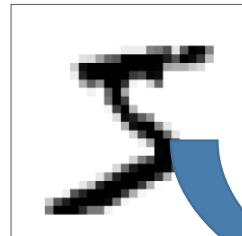


Panacea for Overfitting

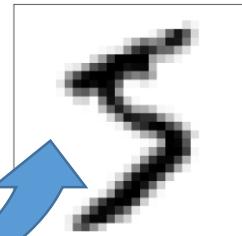
- Have more training data
- *Create* more training data (?)

Handwriting recognition:

Original
Training Data:

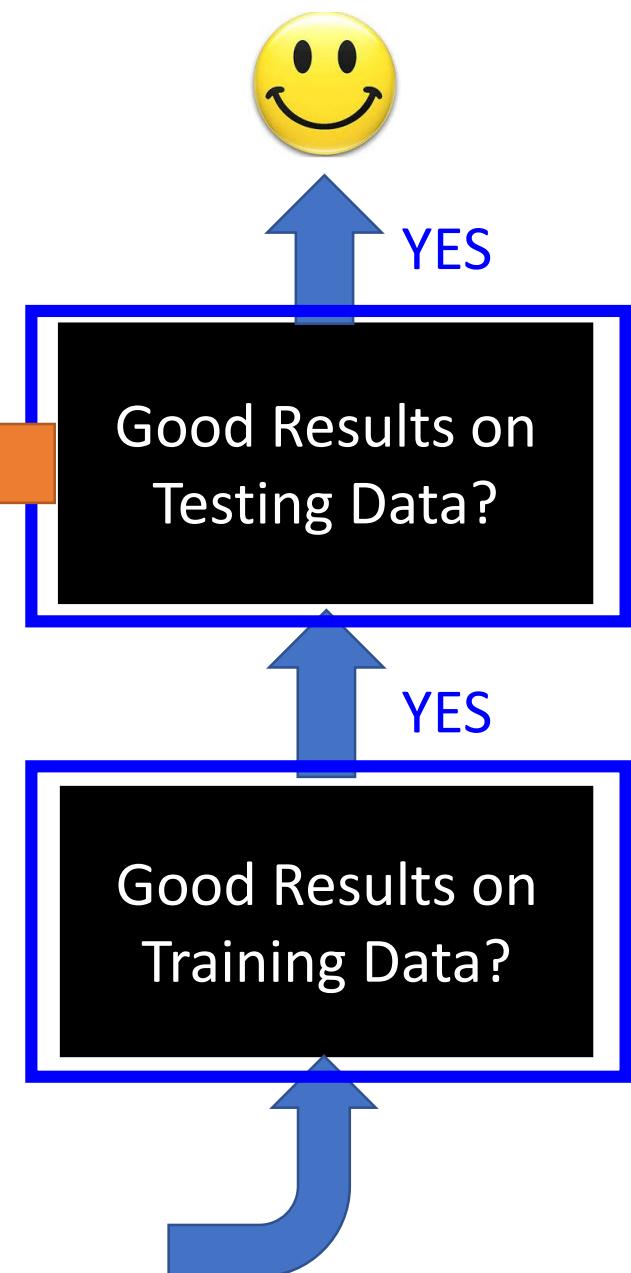
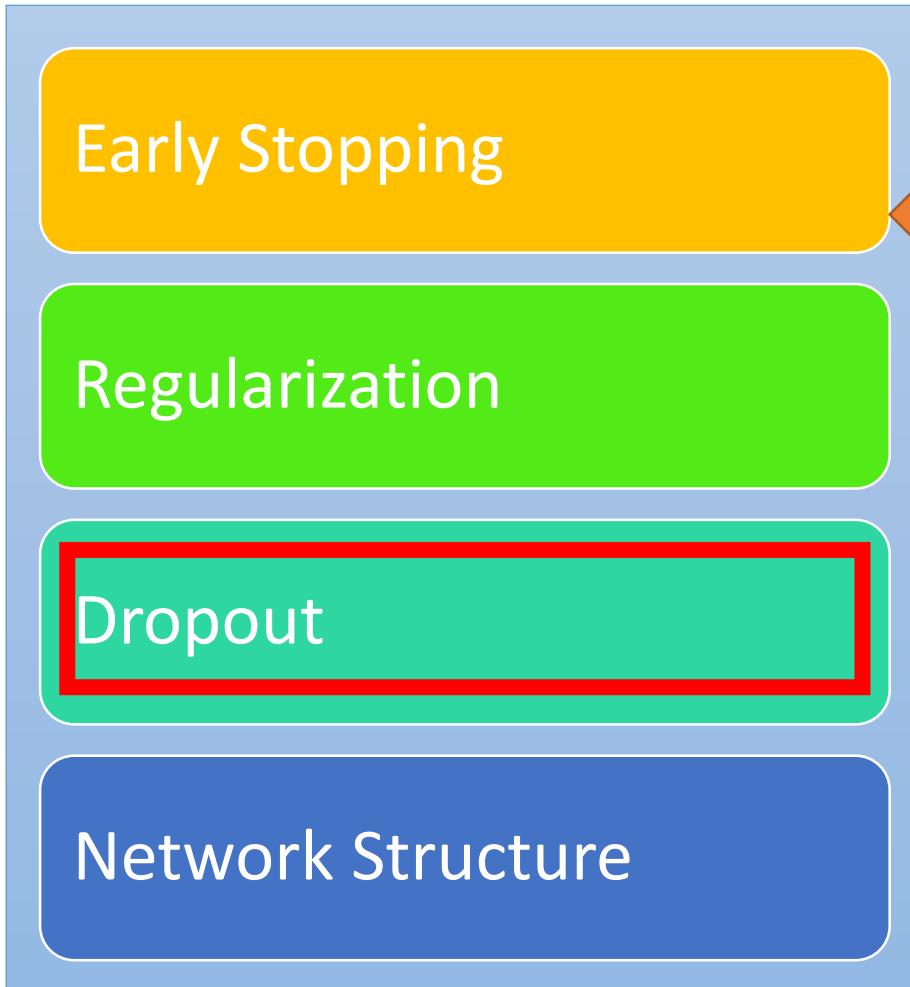


Created
Training Data:



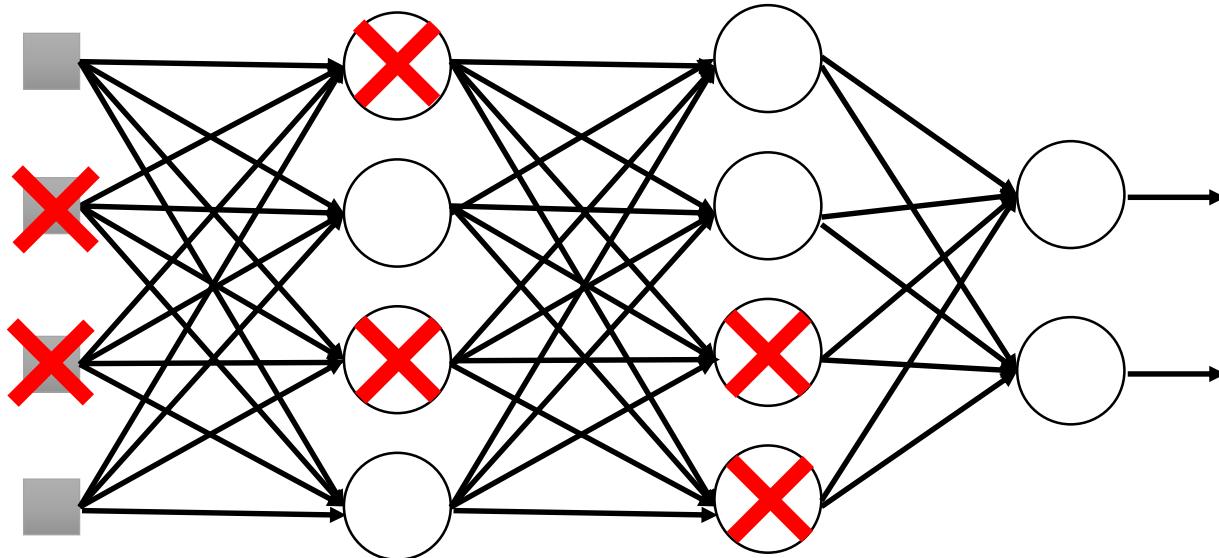
Shift 15 °

Recipe of Deep Learning



Dropout

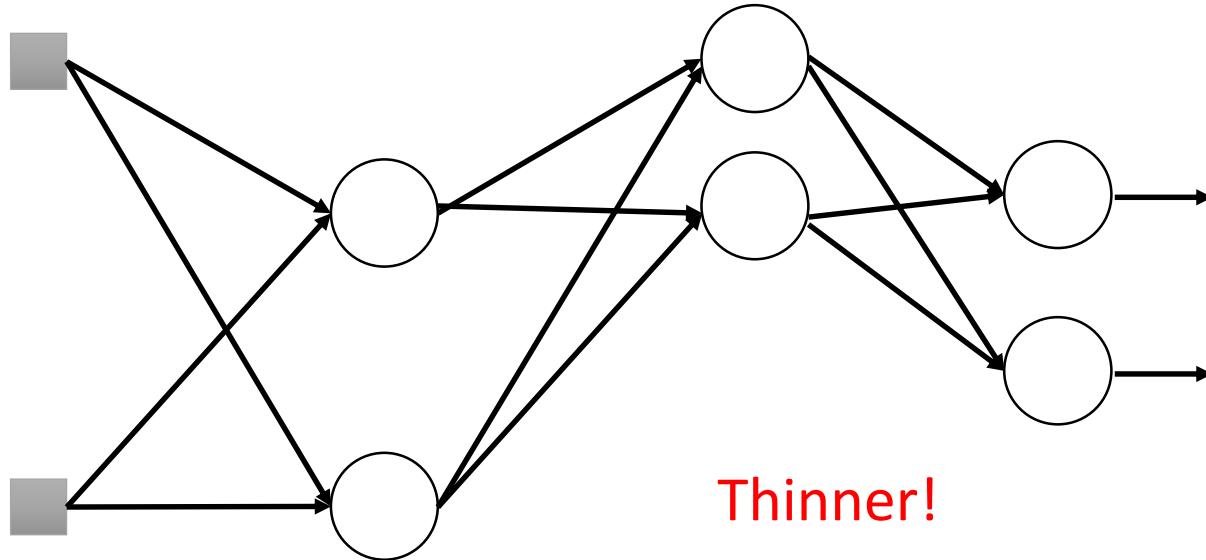
Training:



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

Dropout

Training:

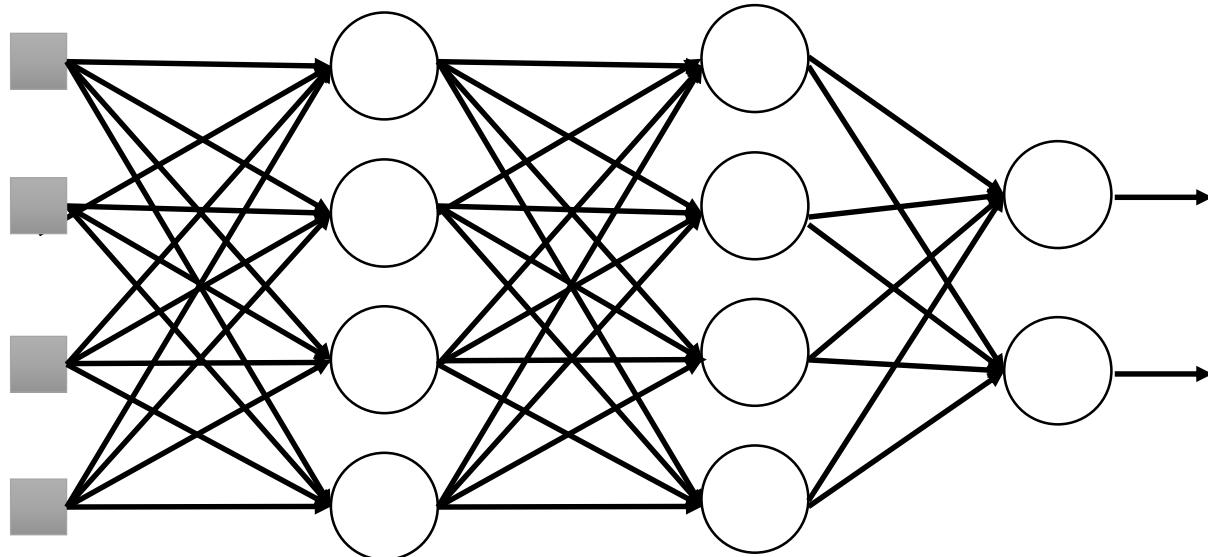


- **Each time before updating the parameters**
 - Each neuron has $p\%$ to dropout
 - ➡ **The structure of the network is changed.**
 - Using the new network for training

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

Training

Dropout (腳上綁重物)



Testing
No dropout
(拿下重物後就變很強)

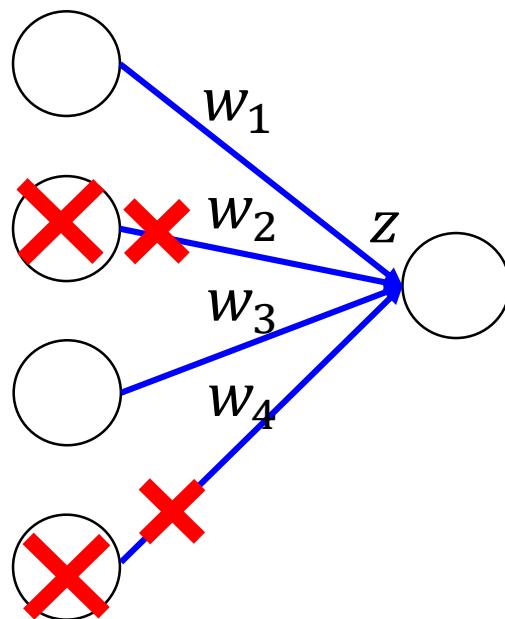


Dropout - Intuitive Reason

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

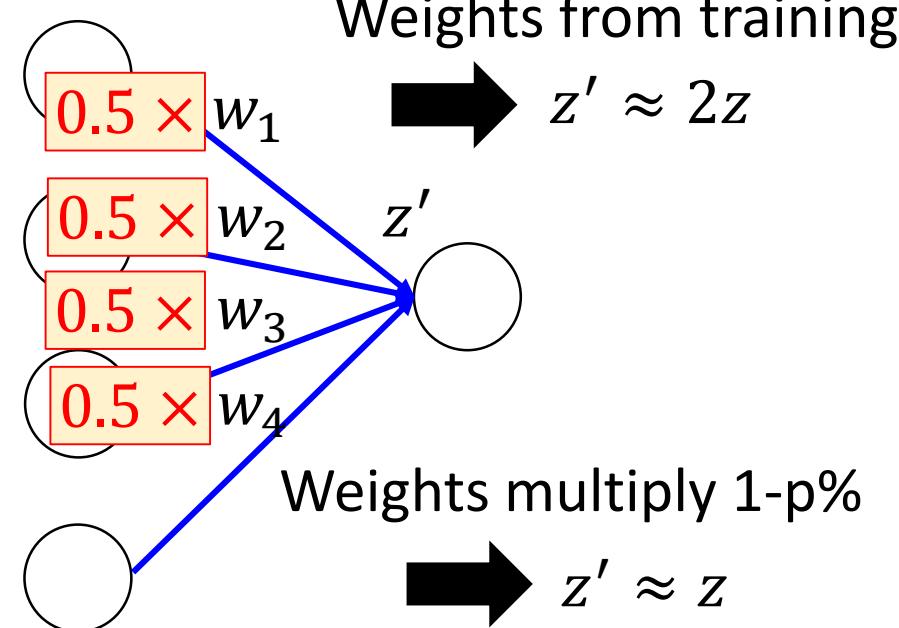
Training of Dropout

Assume dropout rate is 50%



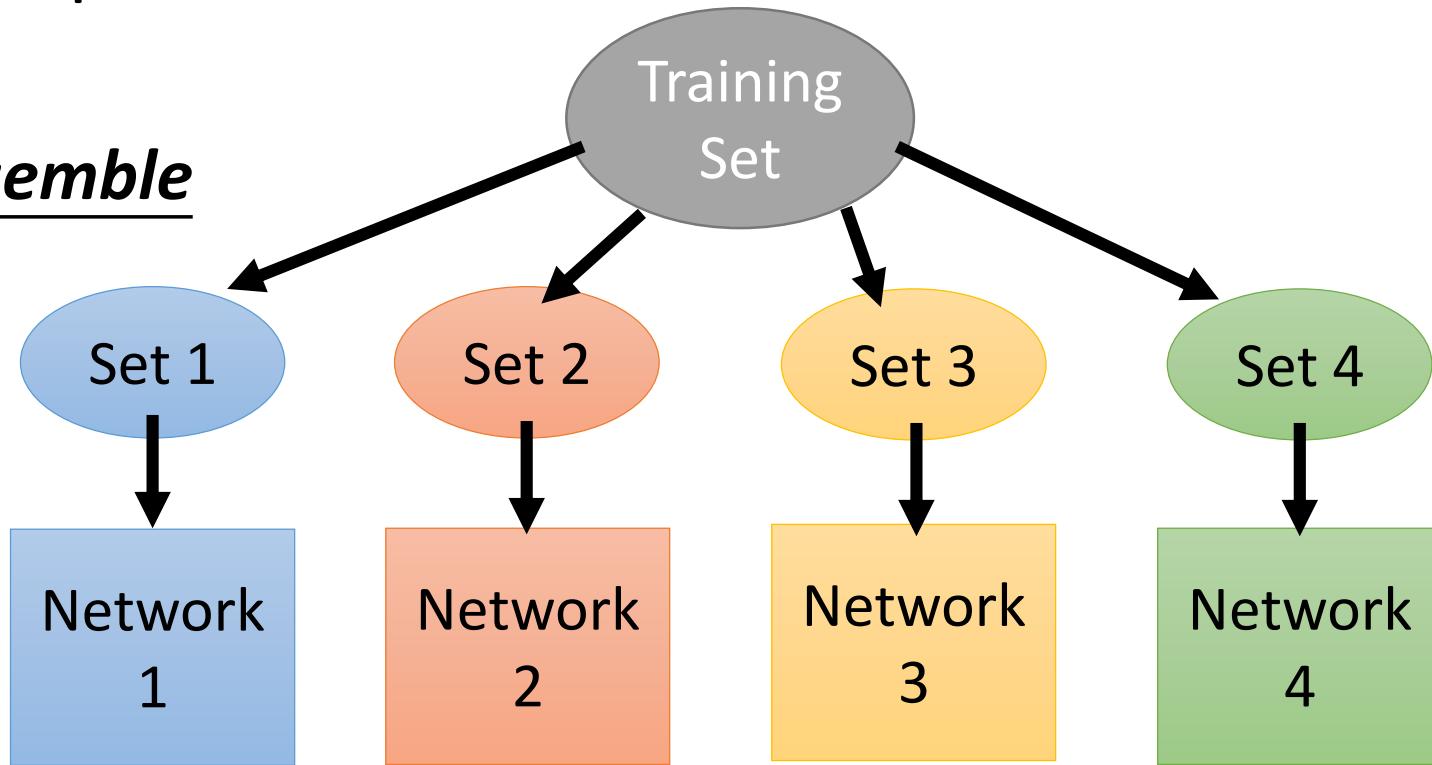
Testing of Dropout

No dropout



Dropout is a kind of ensemble.

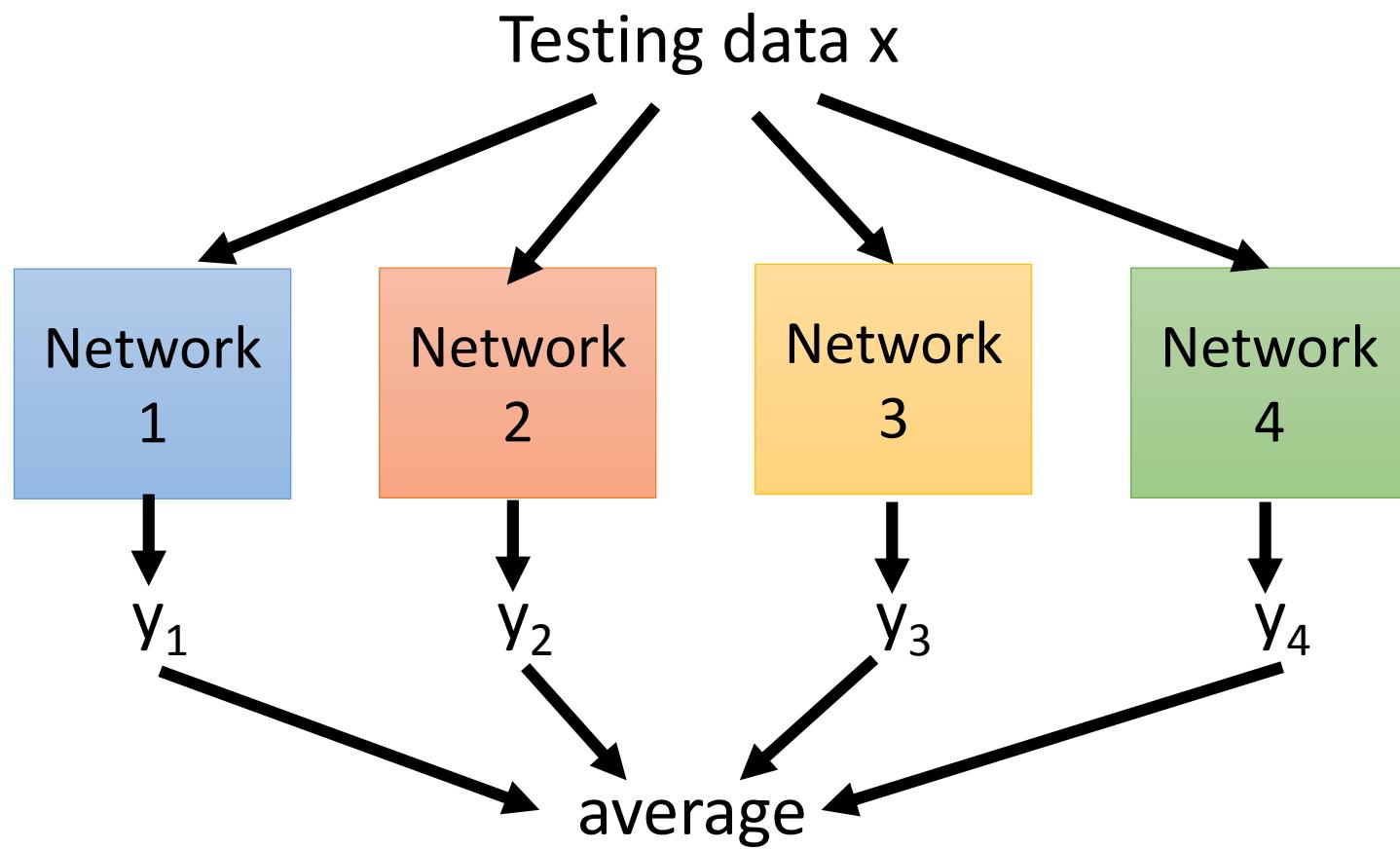
Ensemble



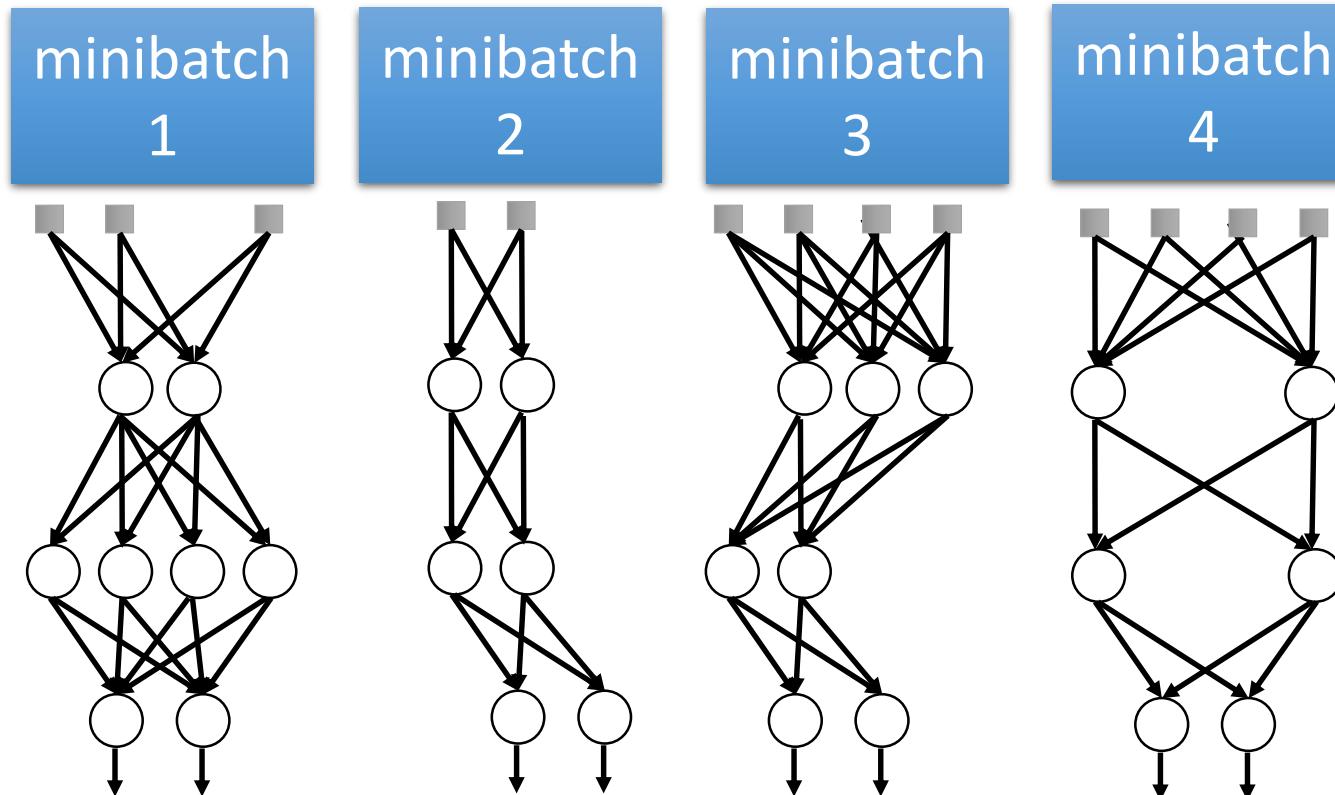
Train a bunch of networks with different structures

Dropout is a kind of ensemble.

Ensemble



Dropout is a kind of ensemble.



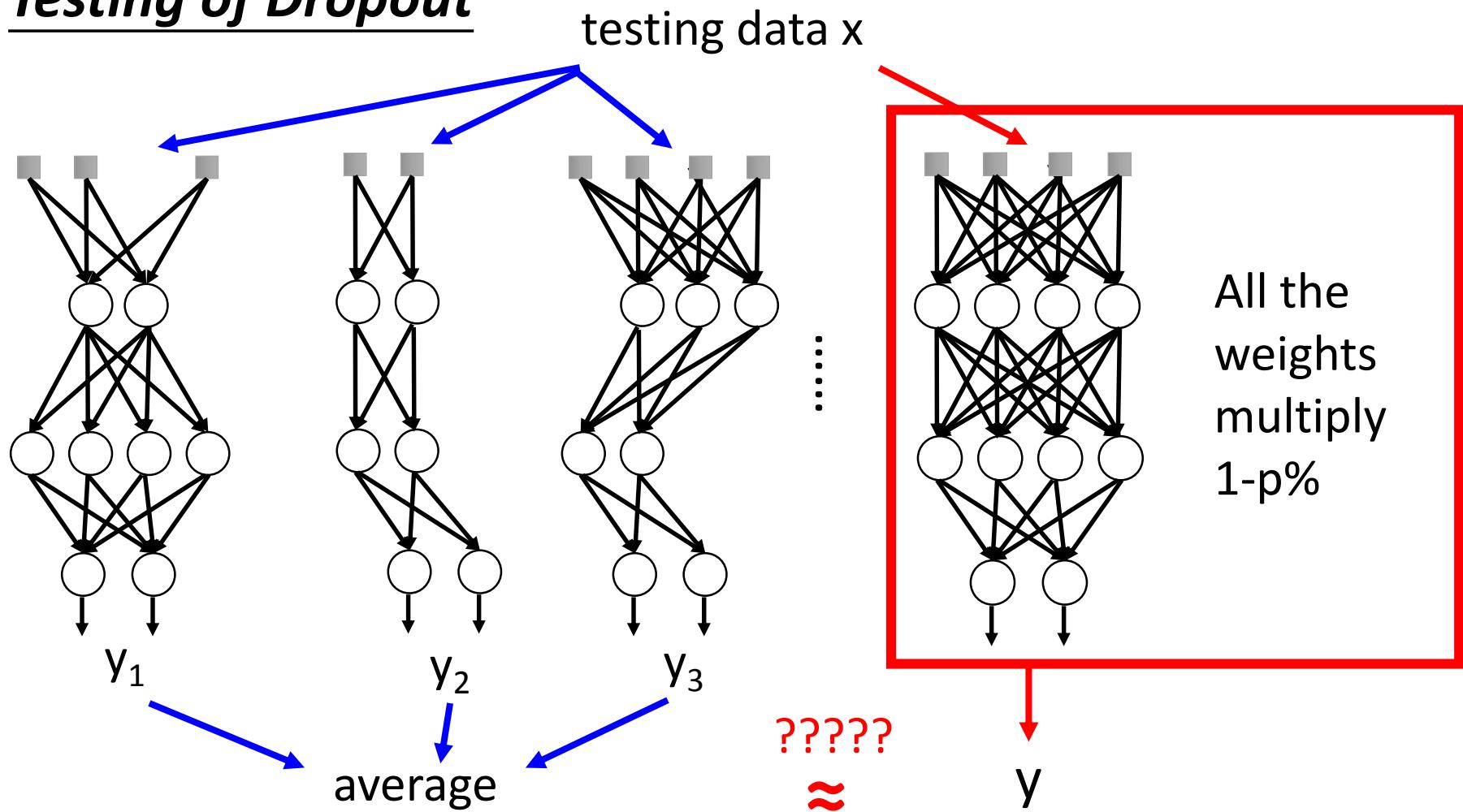
Training of Dropout

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.

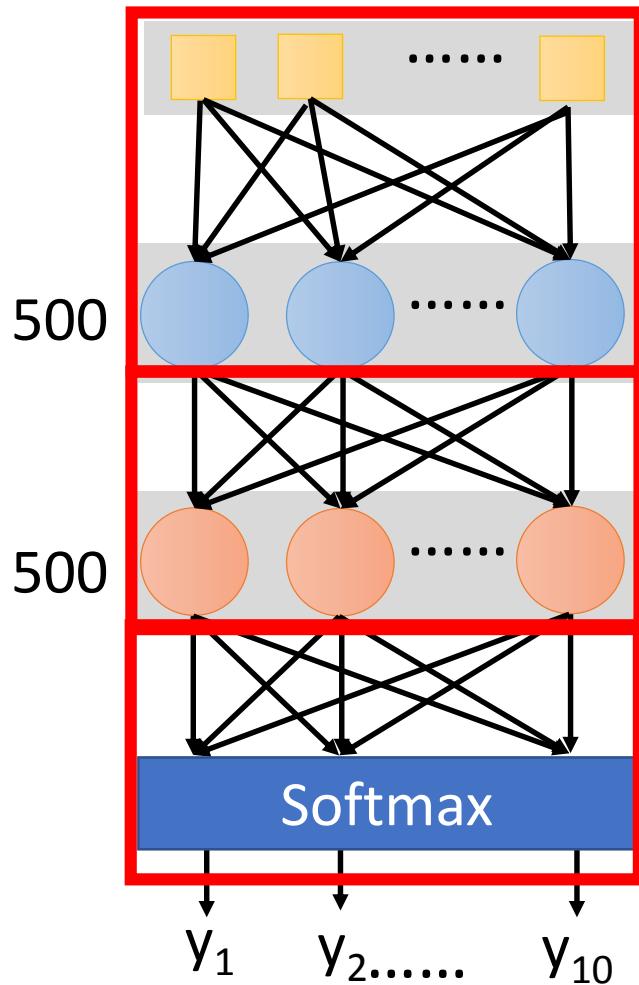
Testing of Dropout



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))

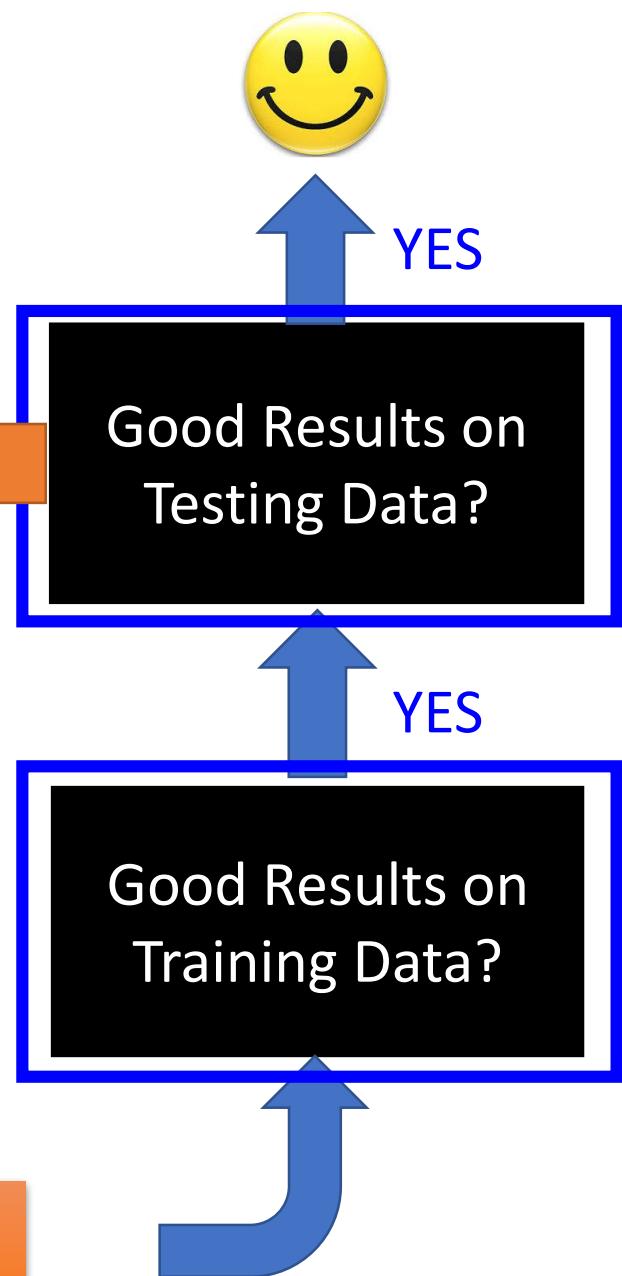
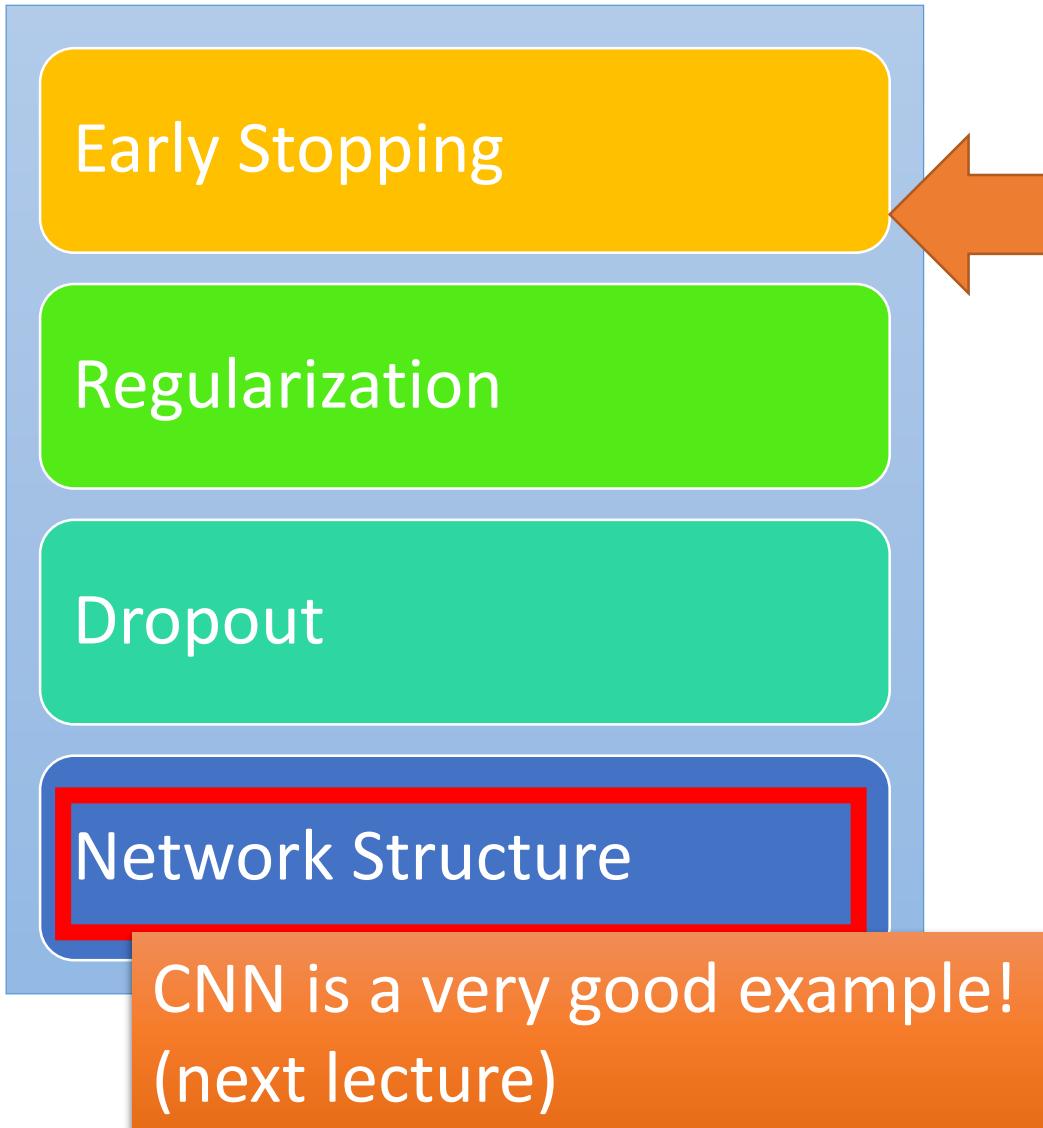
```
model.add( Dense( output_dim=500 ) )  
model.add( Activation('sigmoid') )
```

model.add(dropout(0.8))

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

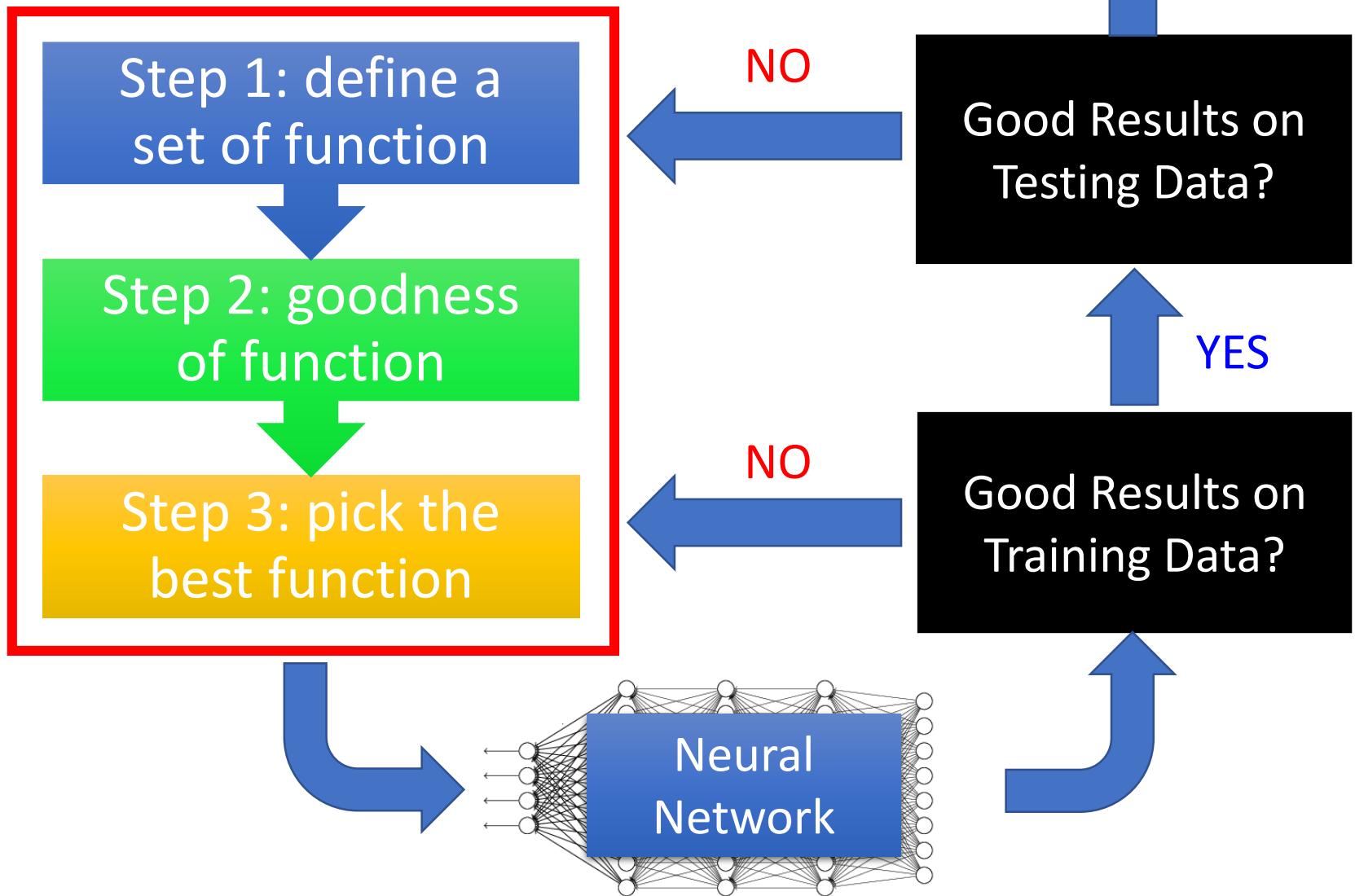
Demo

Recipe of Deep Learning



Concluding Remarks

Recipe of Deep Learning



Lecture II:

Variants of Neural Networks

Variants of Neural Networks

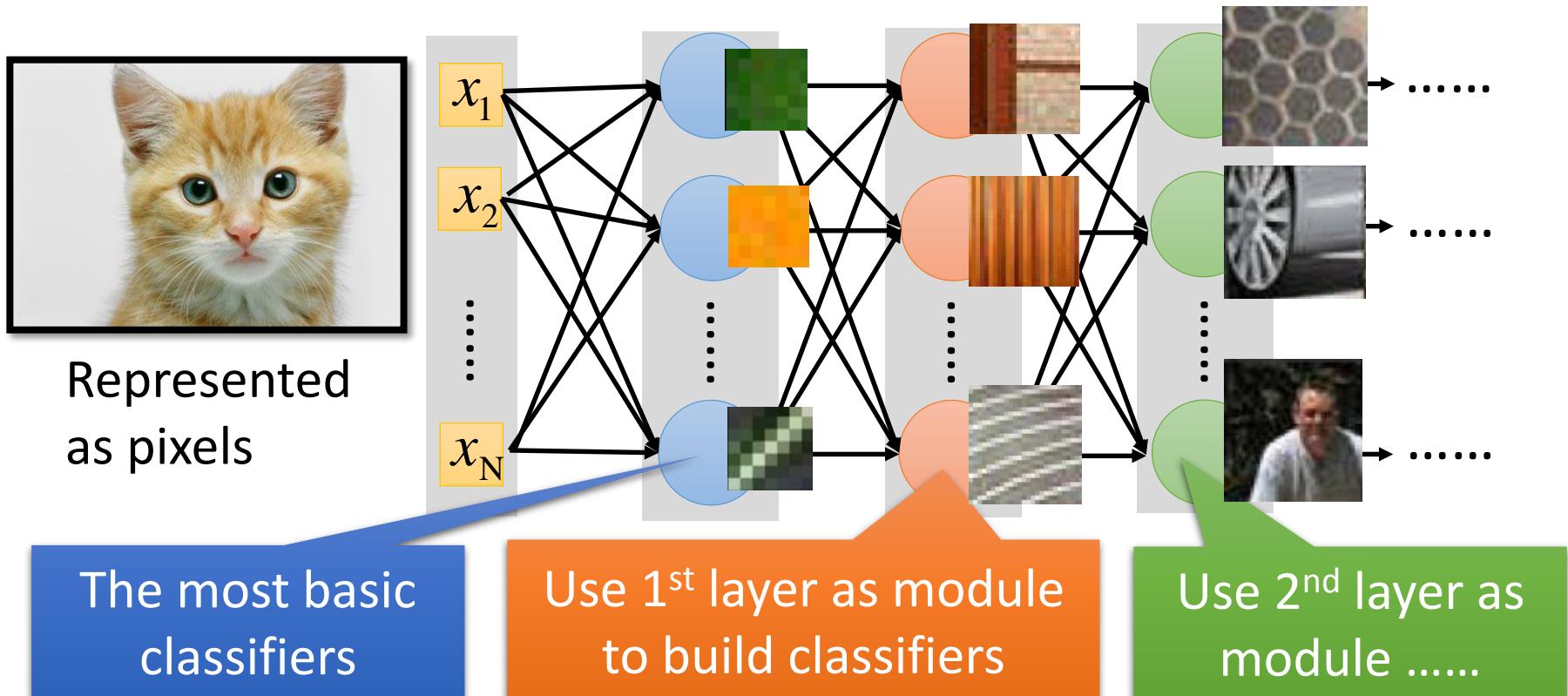
Convolutional Neural
Network (CNN)

Widely used in
image processing

Recurrent Neural Network
(RNN)

Why CNN for Image?

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



The most basic classifiers

Use 1st layer as module
to build classifiers

Use 2nd layer as
module

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

Why CNN for Image

- Some patterns are much smaller than the whole image

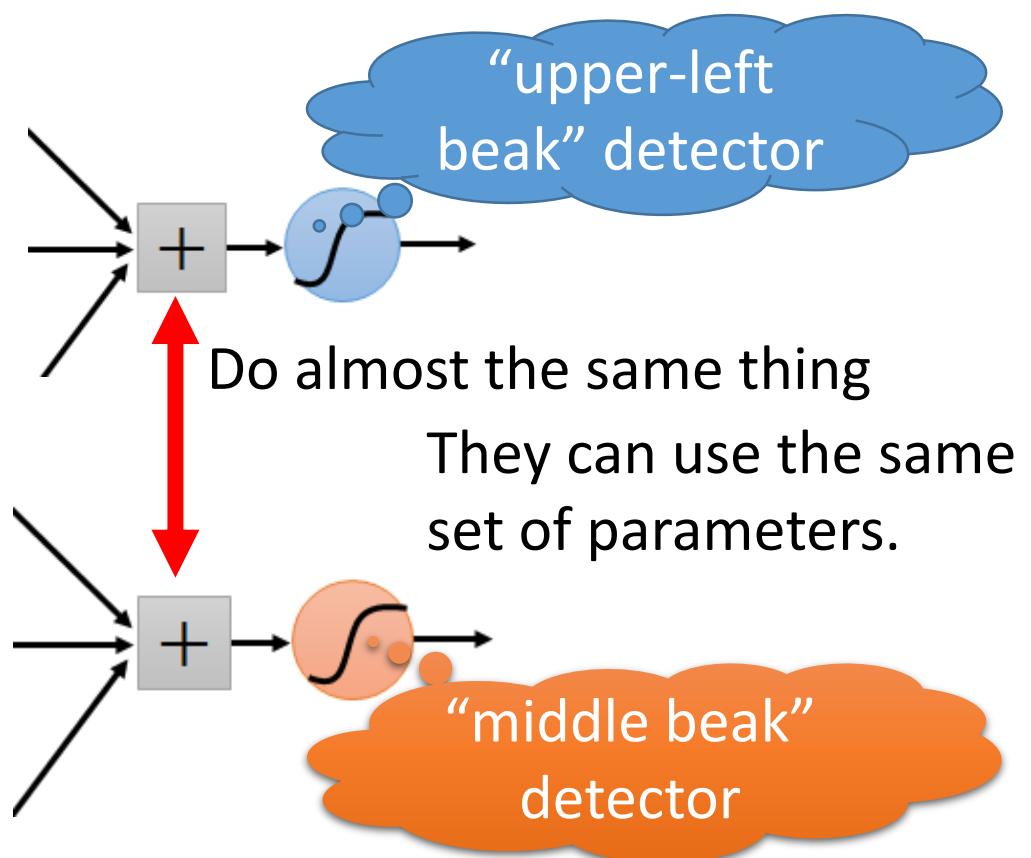
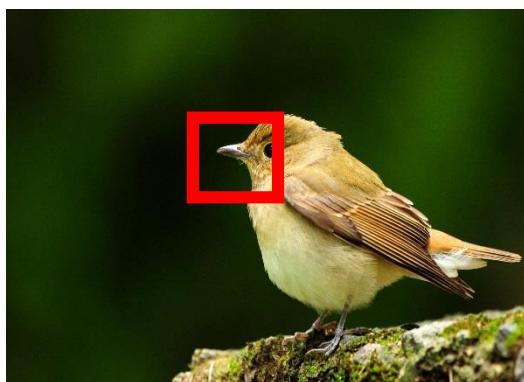
A neuron does not have to see the whole image to discover the pattern.

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



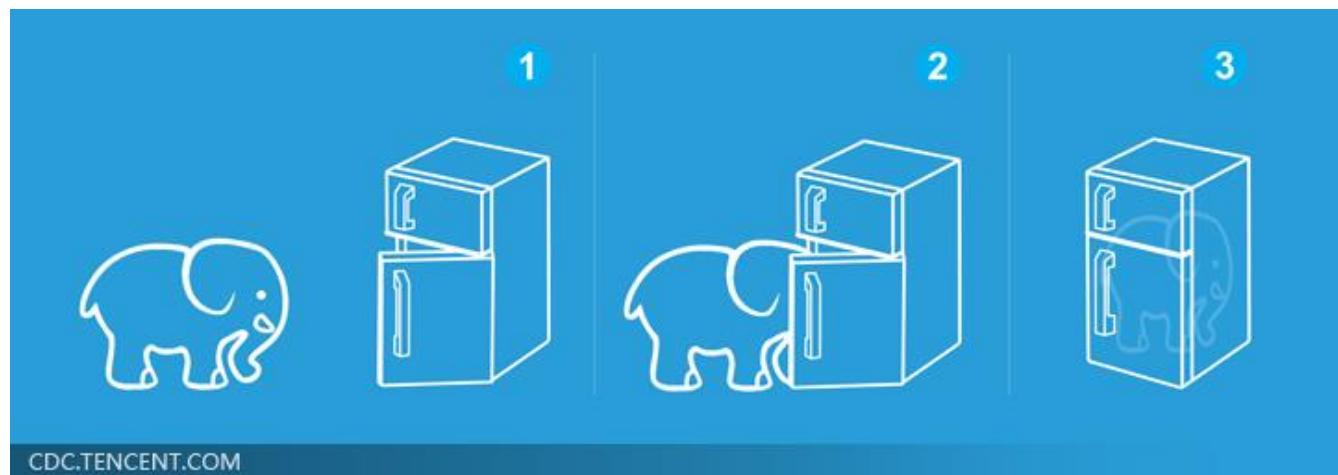
We can subsample the pixels to make image smaller

→ Less parameters for the network to process the image

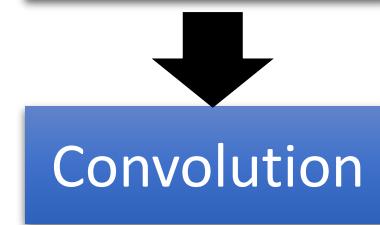
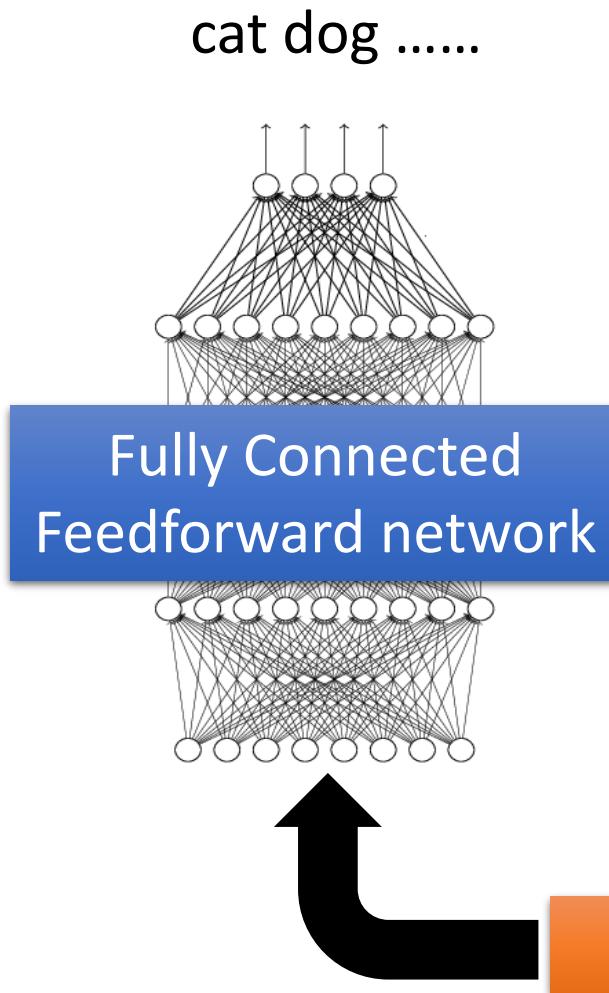
Three Steps for Deep Learning



Deep Learning is so simple



The whole CNN



Can repeat
many times

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

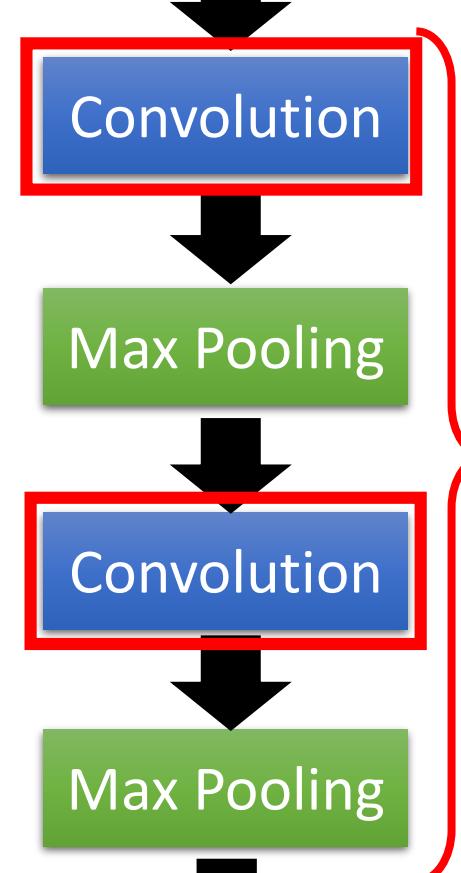
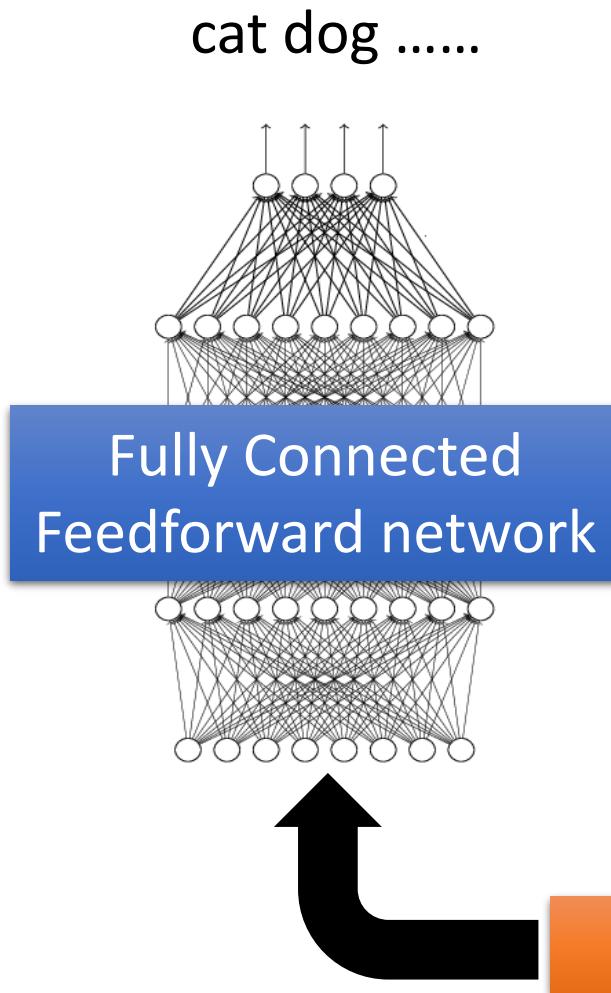
Convolution

Max Pooling

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

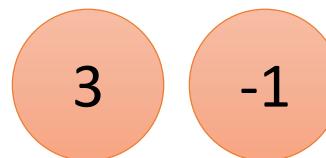
stride=1

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

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

Filter 1



CNN – Convolution

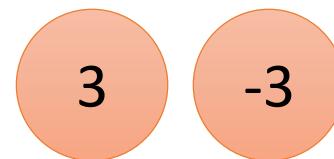
If stride=2

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

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

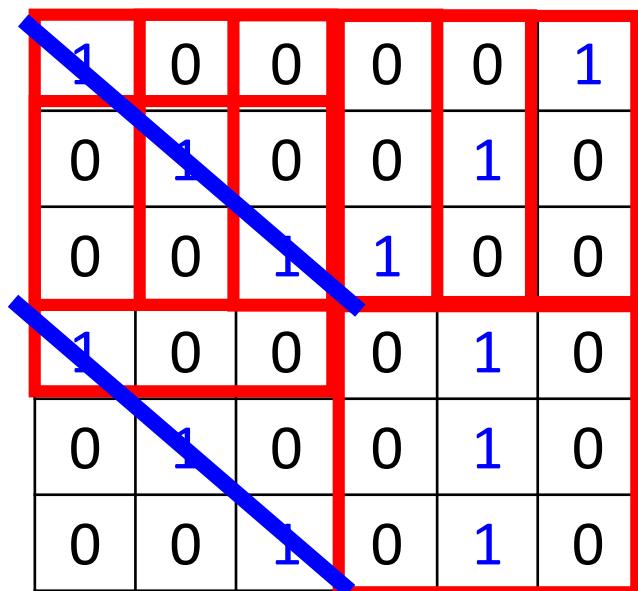
Filter 1



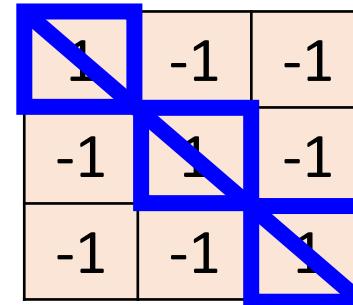
We set stride=1 below

CNN – Convolution

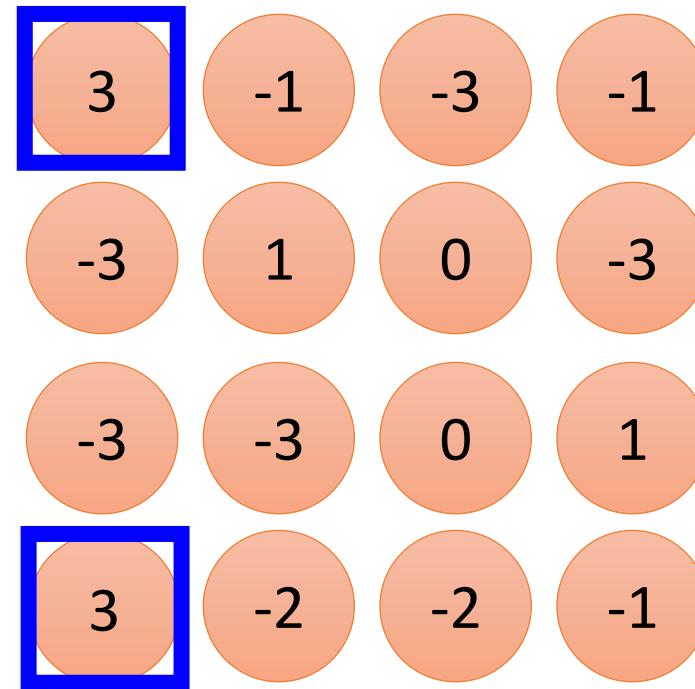
stride=1



6 x 6 image



Filter 1



Property 2

CNN – Convolution

stride=1

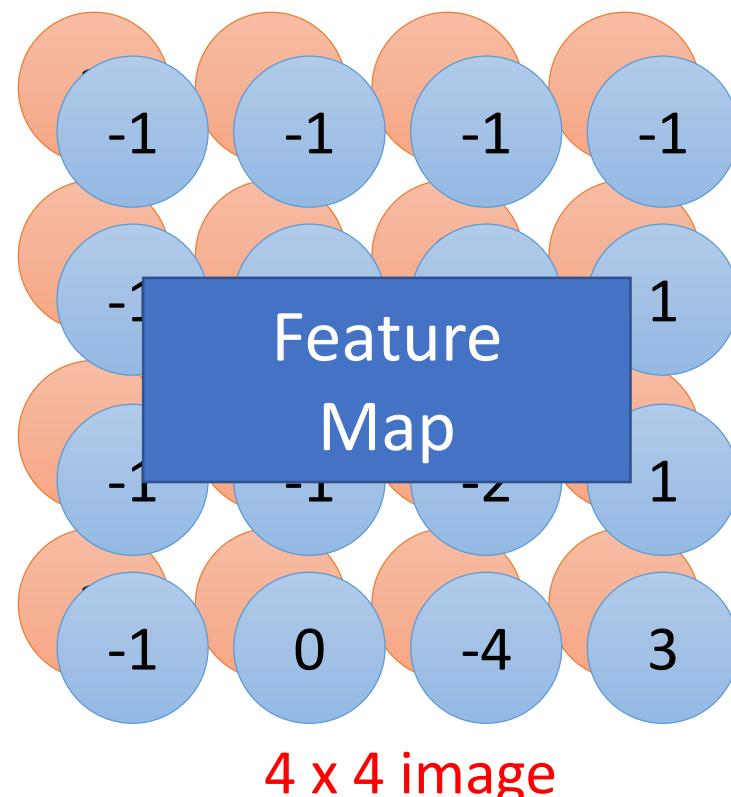
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

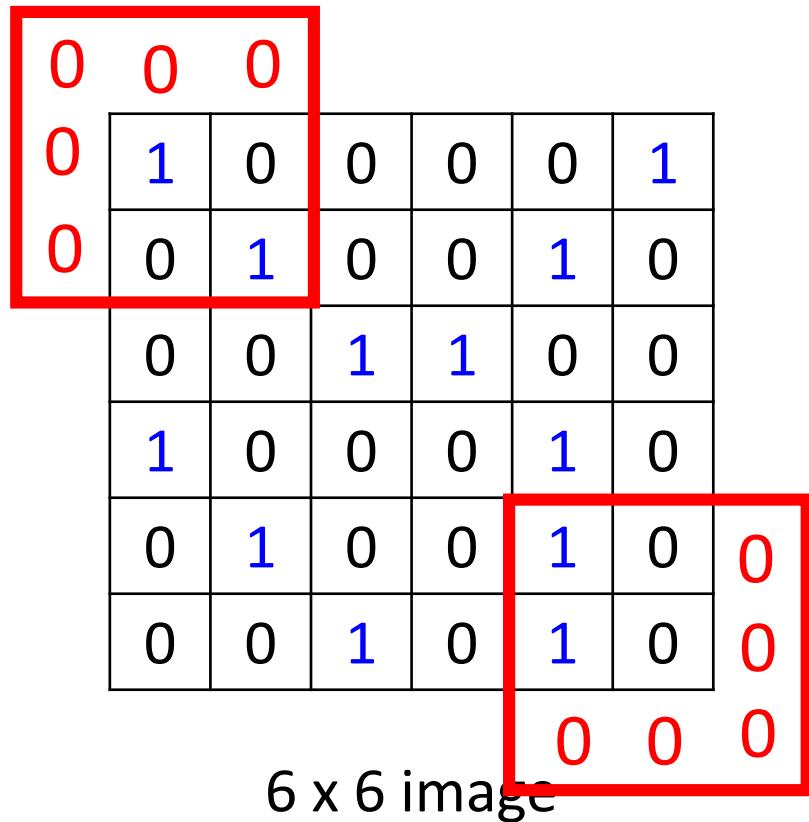
-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

Do the same process for every filter



CNN – Zero Padding



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

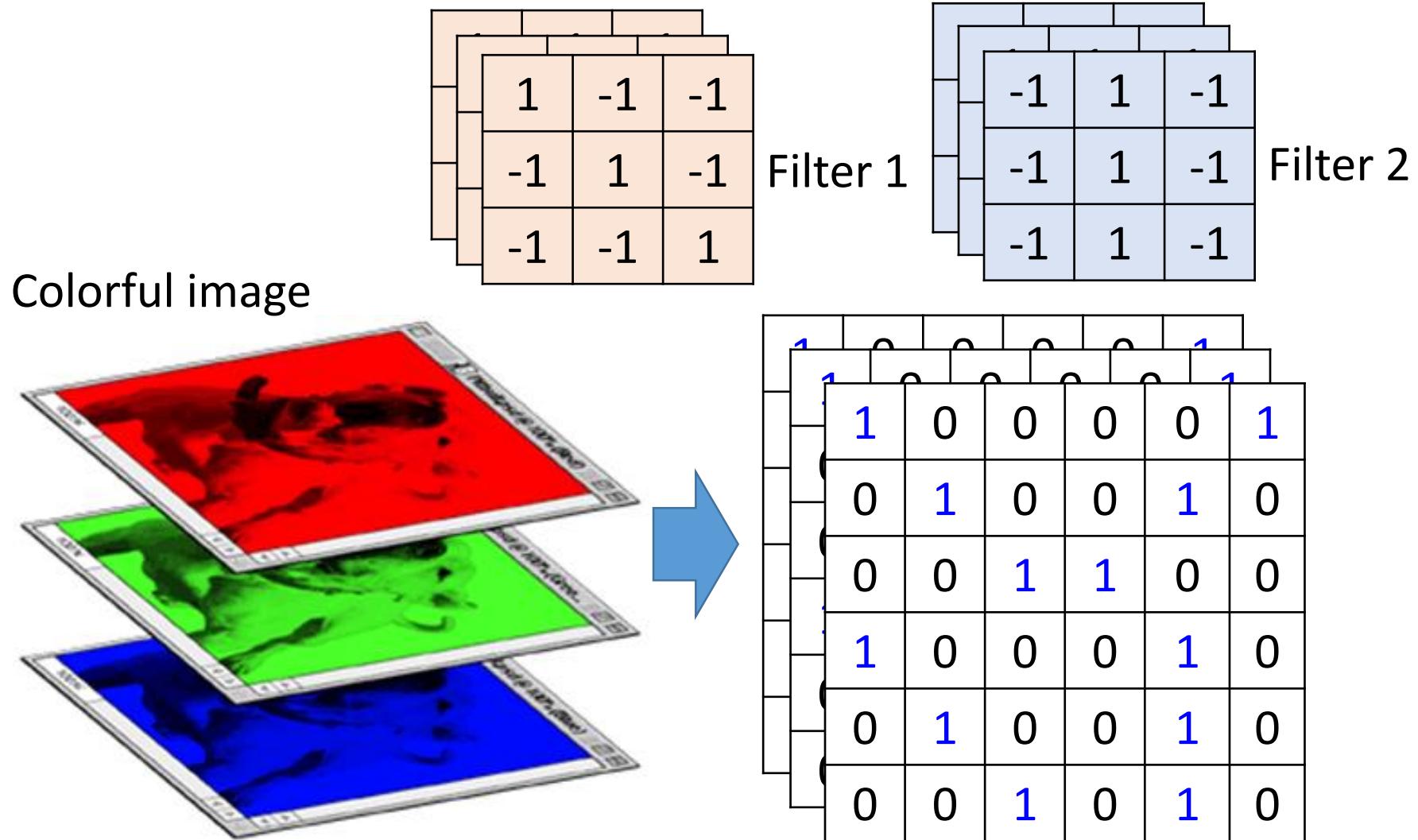
Filter 1

You will get another 6×6 images in this way

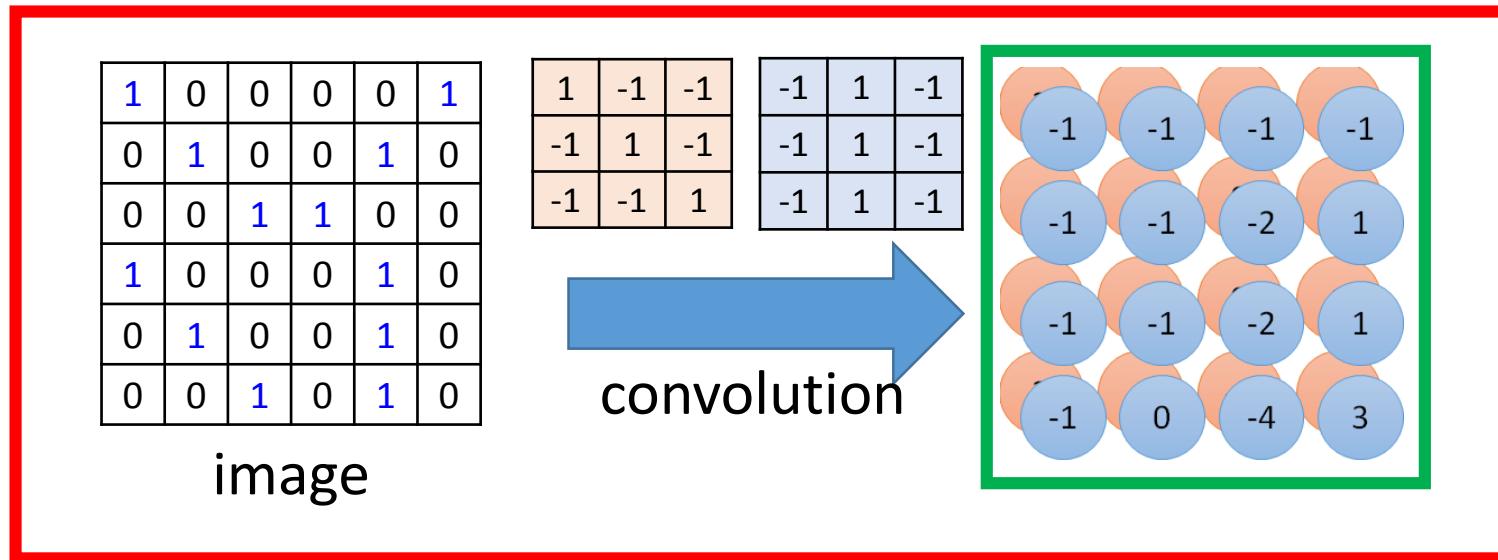


Zero padding

CNN – Colorful image

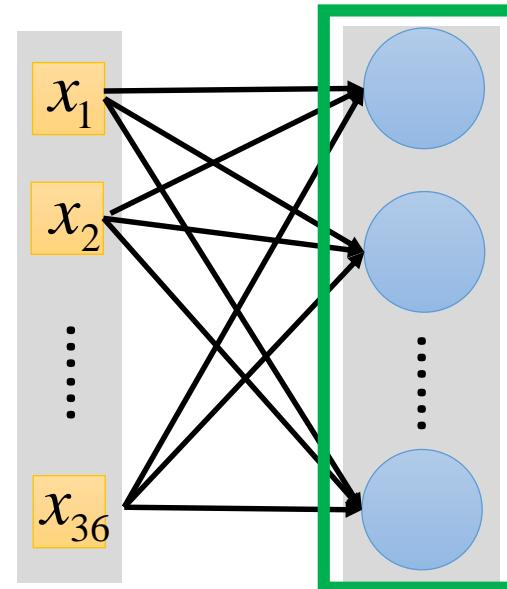


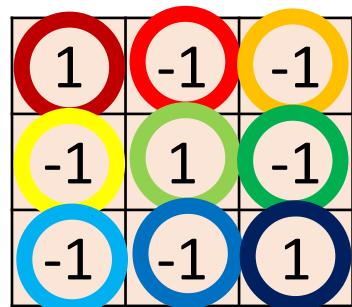
Convolution v.s. Fully Connected



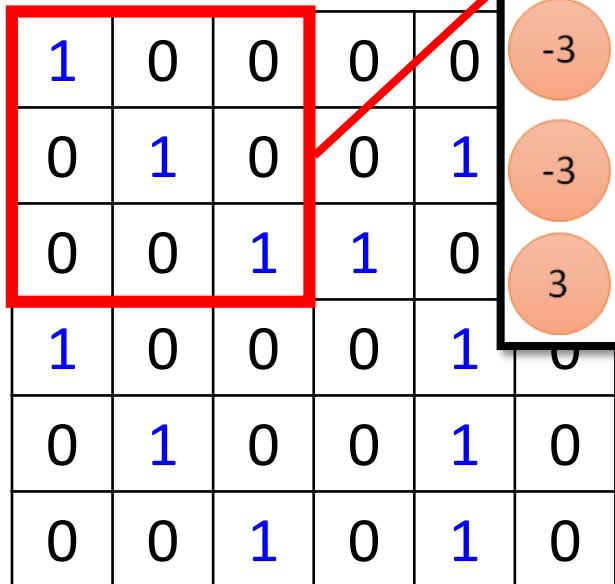
Fully-
connected

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

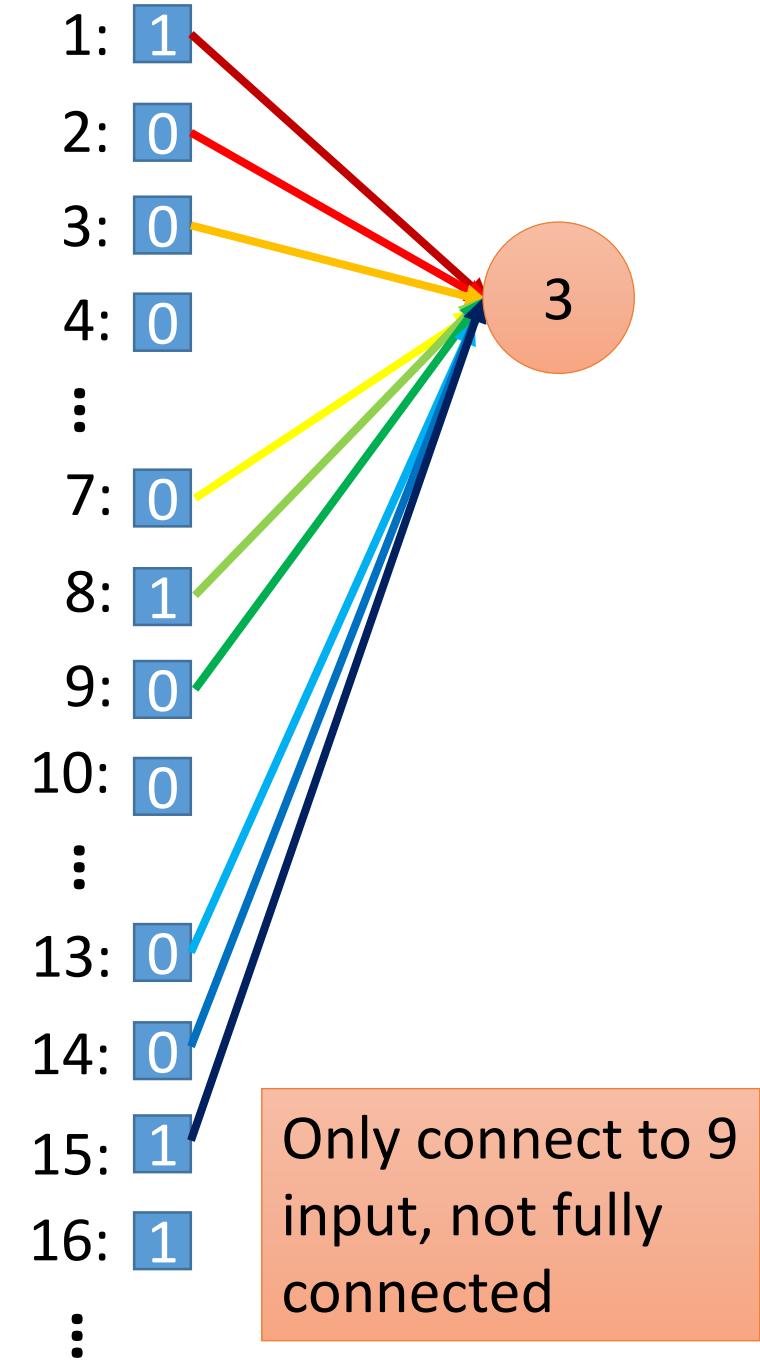
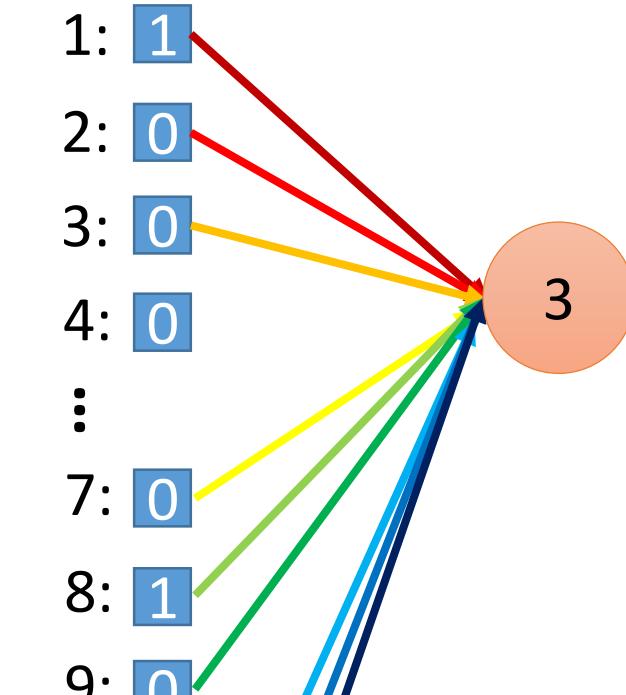
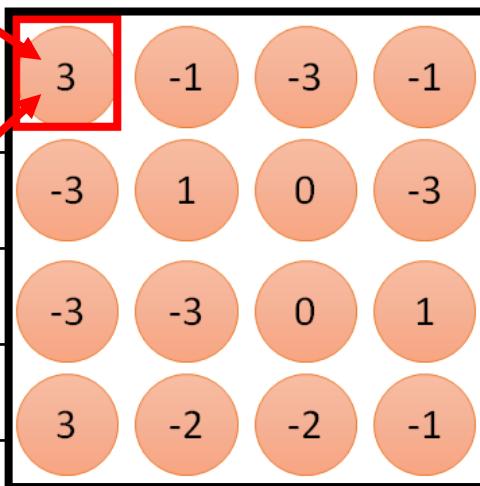


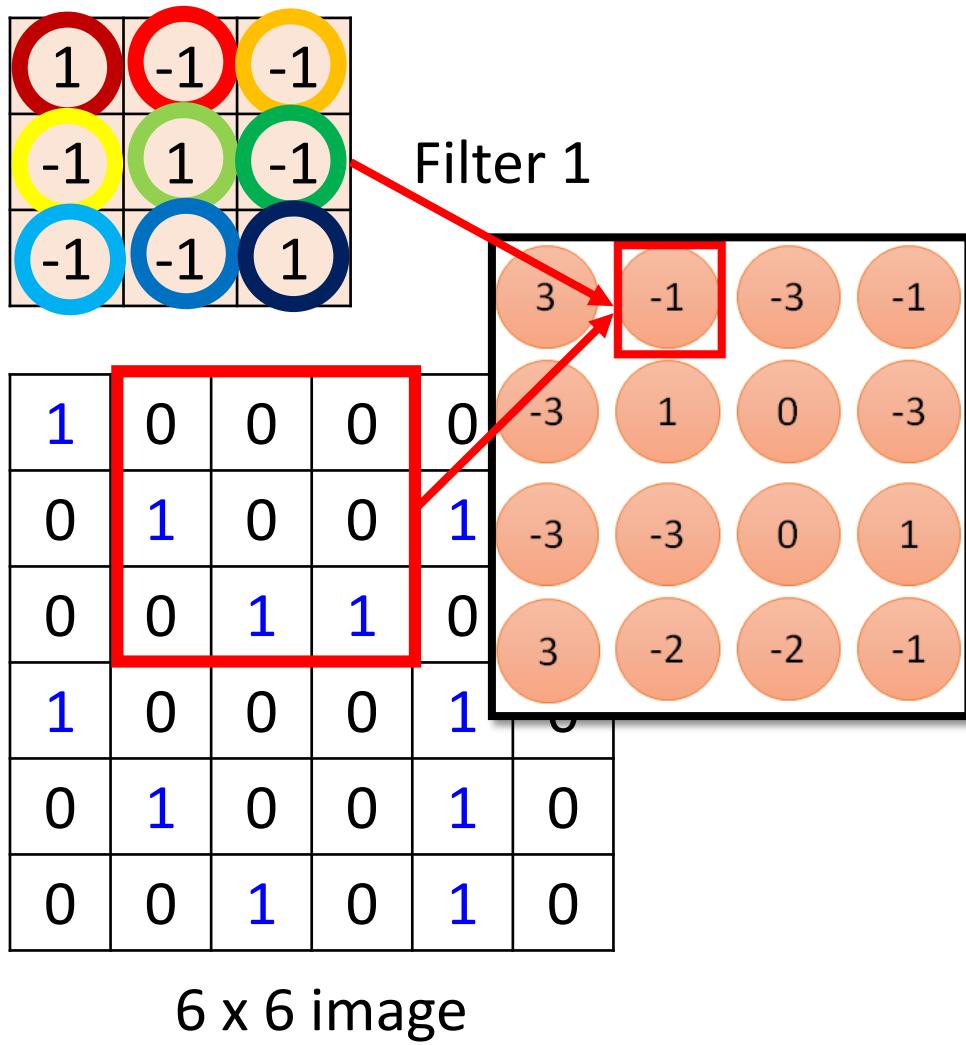


Filter 1



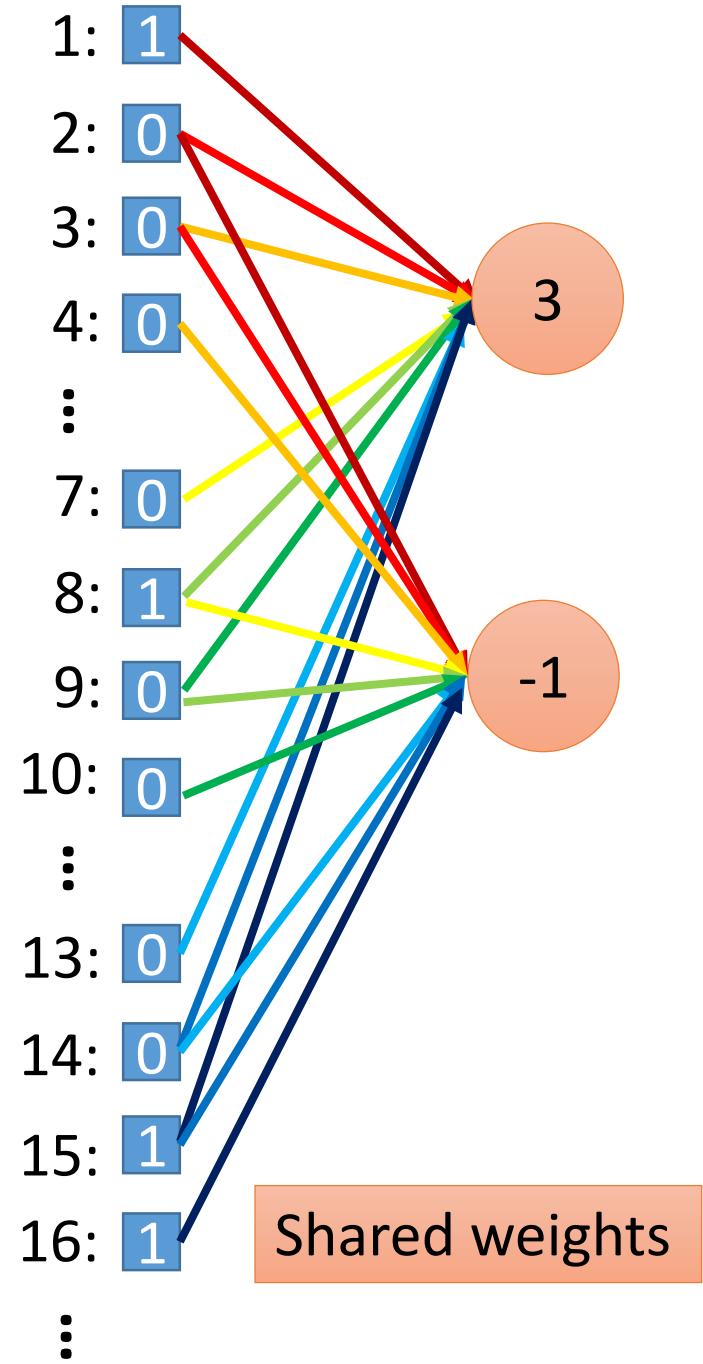
Less parameters!



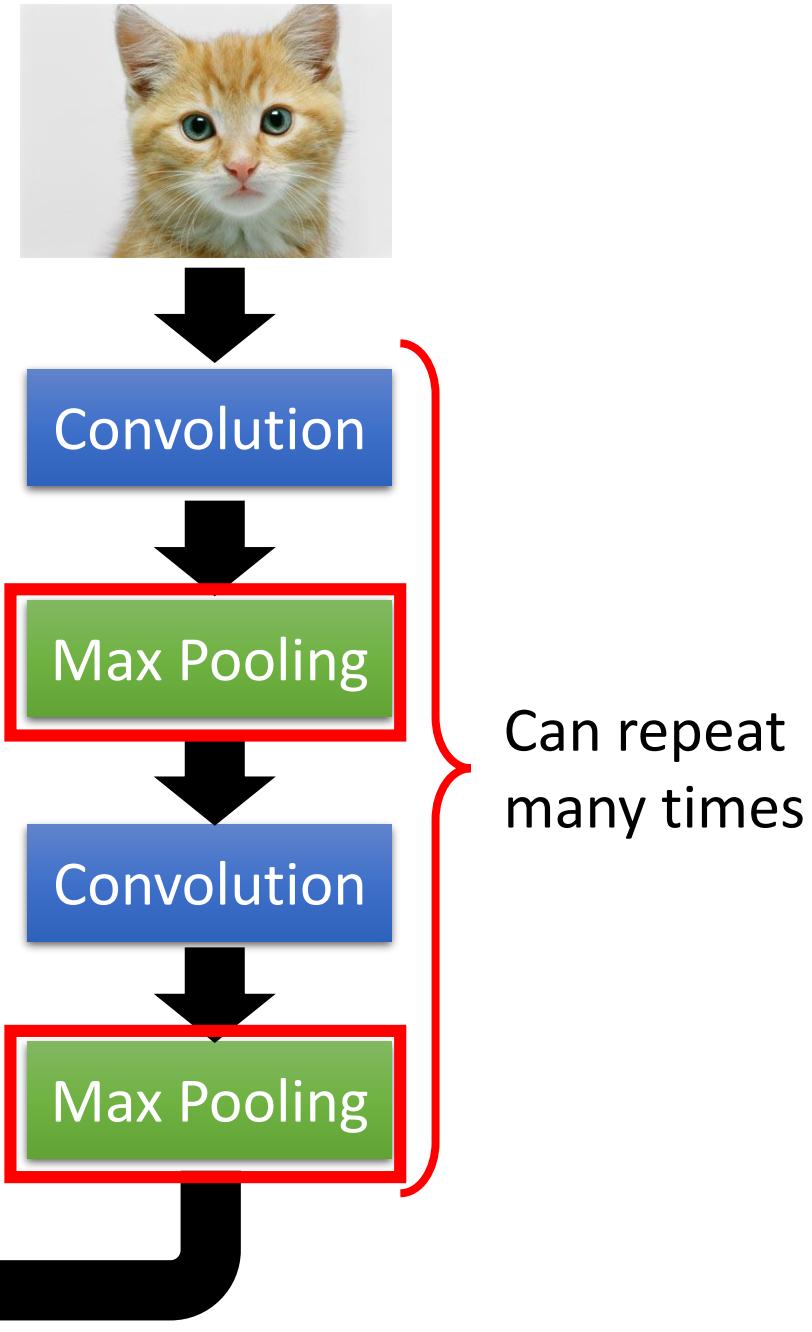
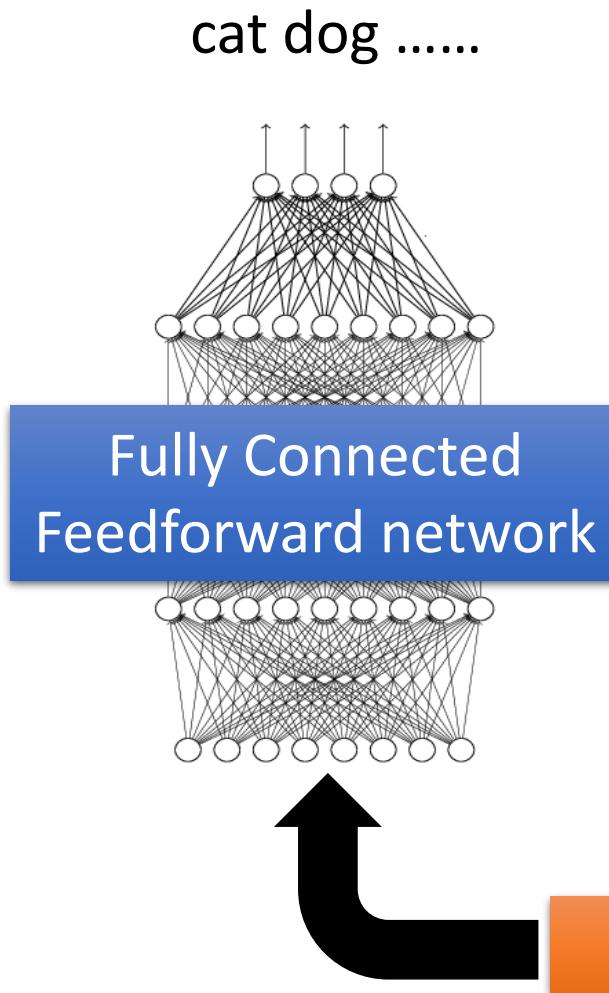


Less parameters!

Even less parameters!



The whole CNN



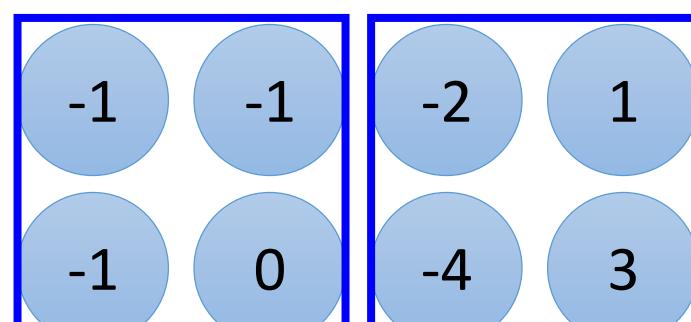
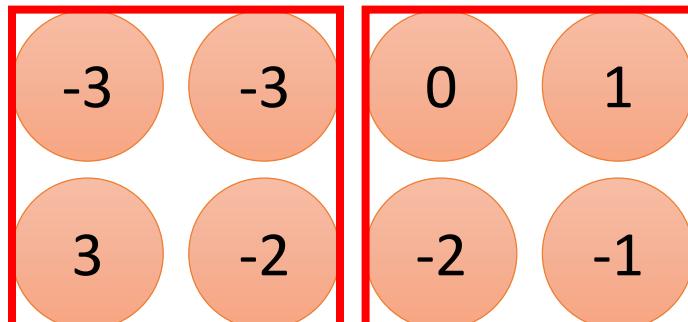
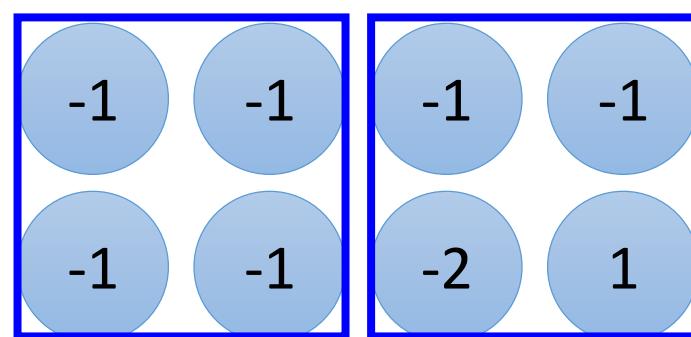
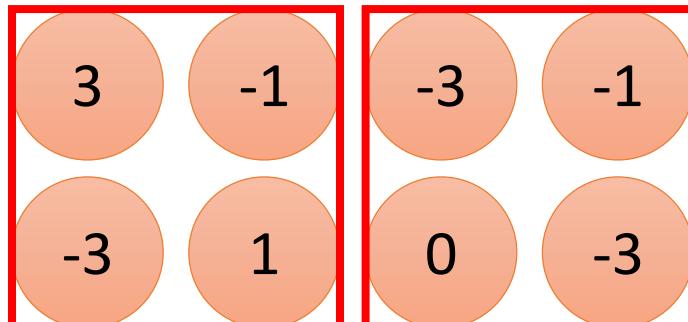
CNN – Max Pooling

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

Filter 1

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

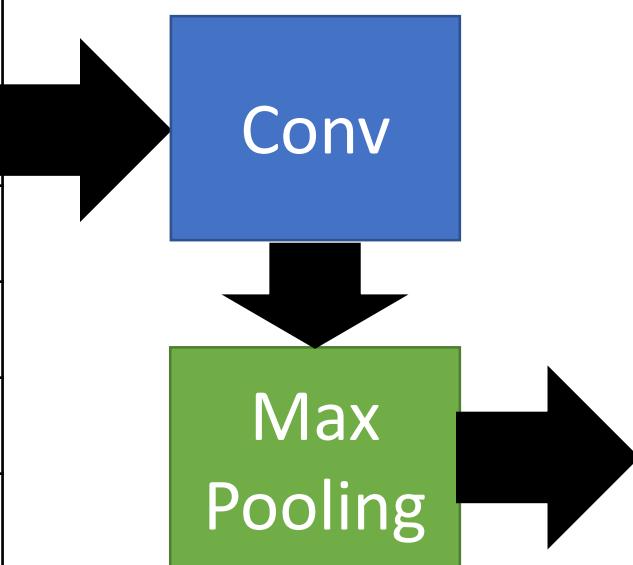
Filter 2



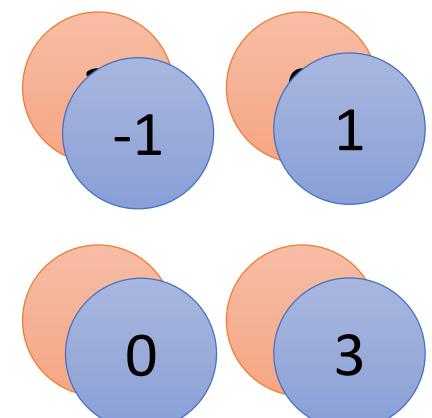
CNN – Max Pooling

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



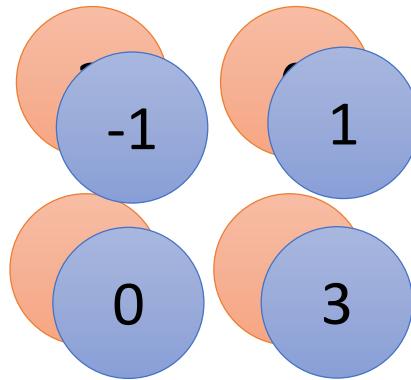
New image
but smaller



2 x 2 image

Each filter
is a channel

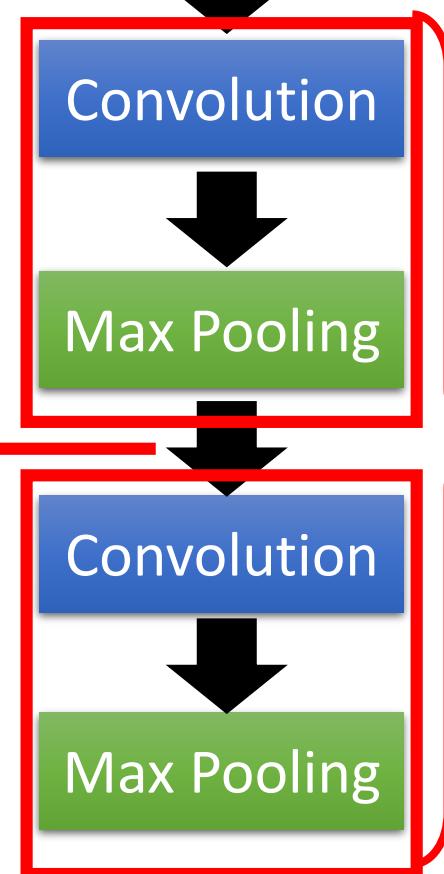
The whole CNN



A new image

Smaller than the original image

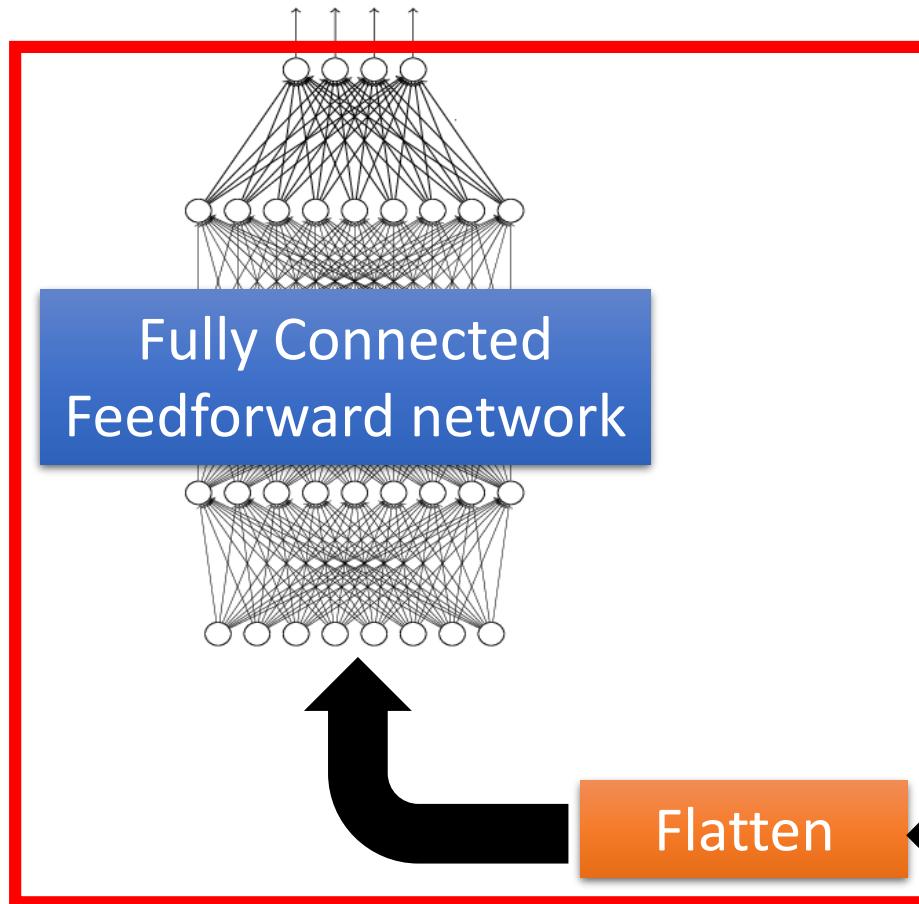
The number of the channel is the number of filters



Can repeat many times

The whole CNN

cat dog



Convolution

Max Pooling

A new image

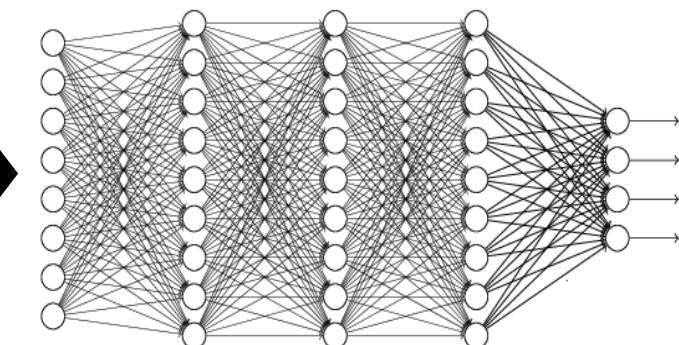
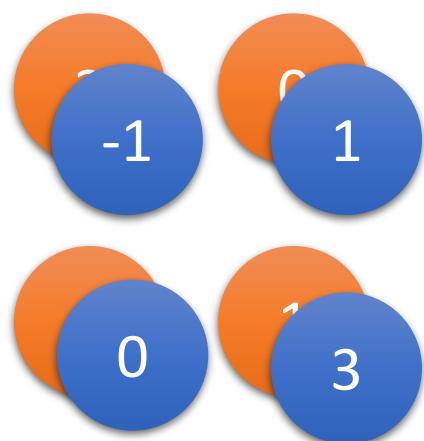
Convolution

Max Pooling

A new image

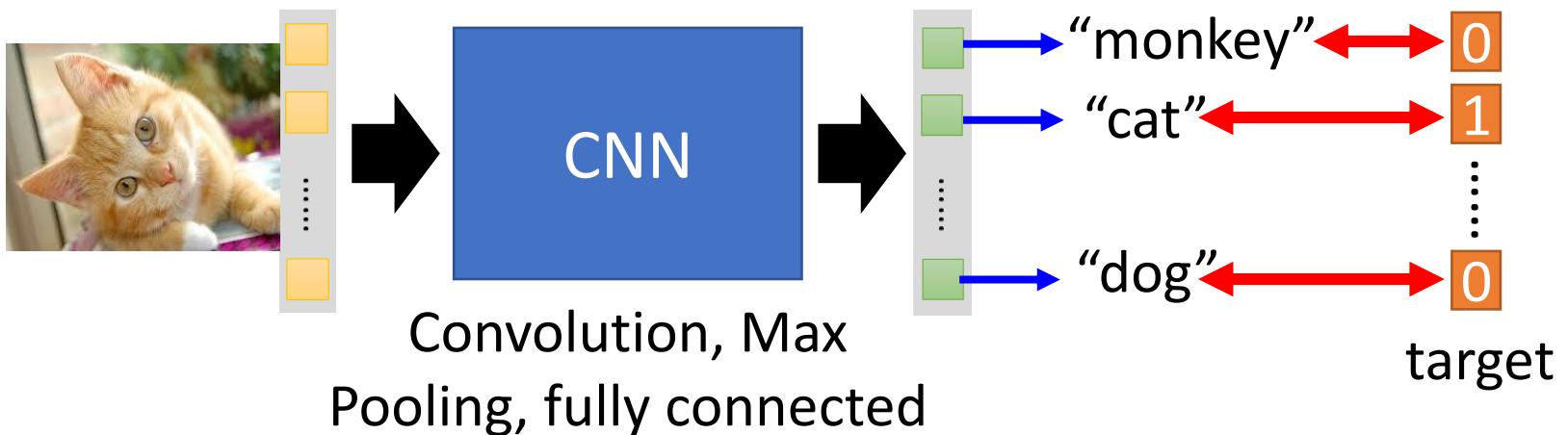
Flatten

Flatten



Fully Connected
Feedforward network

Convolutional Neural Network

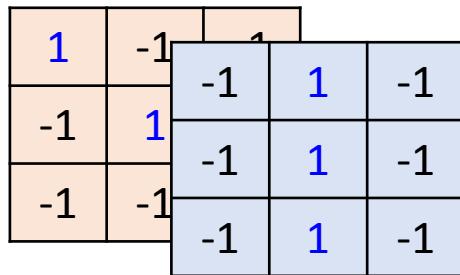


Learning: Nothing special, just gradient descent

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) ) )
```

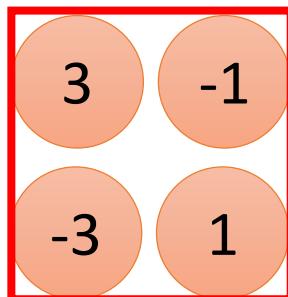


There are 25
3x3 filters.

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)*

How many parameters
for each filter?

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

9

25 x 26 x 26

input



Convolution



Max Pooling



Convolution



Max Pooling

How many parameters
for each filter?

225

50 x 11 x 11

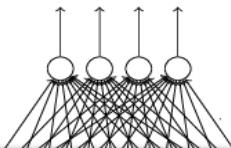
```
model2.add( MaxPooling2D( (2, 2) ) )
```

50 x 5 x 5

CNN in Keras

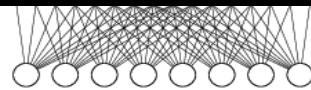
Only modified the ***network structure*** and ***input format (vector -> 3-D tensor)***

output



Fully Connected
Feedforward network

```
model2.add(Dense(output_dim=100))  
model2.add(Activation('relu'))  
model2.add(Dense(output_dim=10))  
model2.add(Activation('softmax'))
```



1250

Flatten

```
model2.add(Flatten())
```

input

$1 \times 28 \times 28$

Convolution

$25 \times 26 \times 26$

Max Pooling

$25 \times 13 \times 13$

Convolution

$50 \times 11 \times 11$

Max Pooling

$50 \times 5 \times 5$

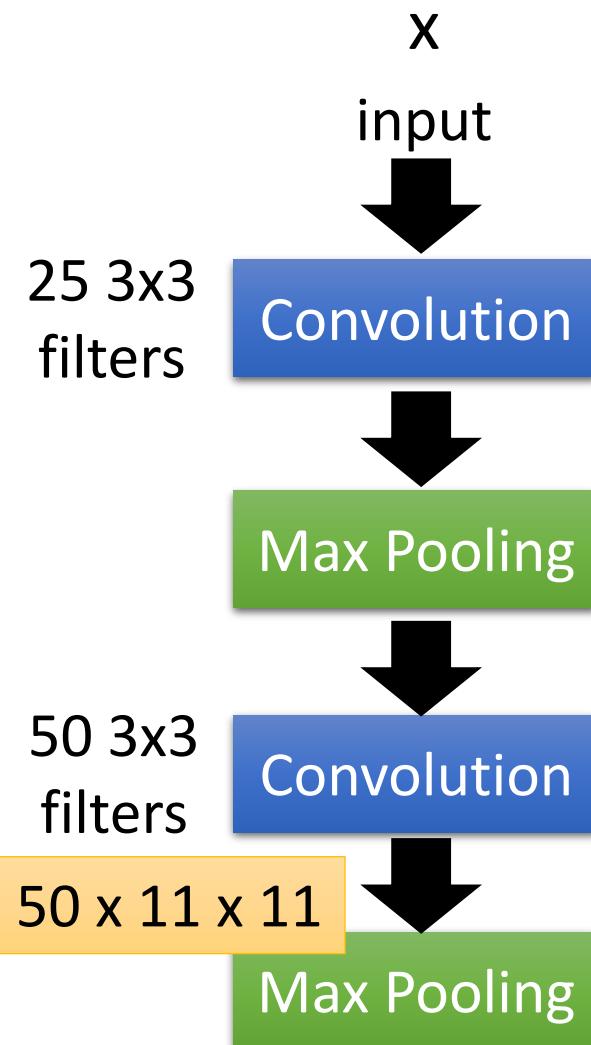
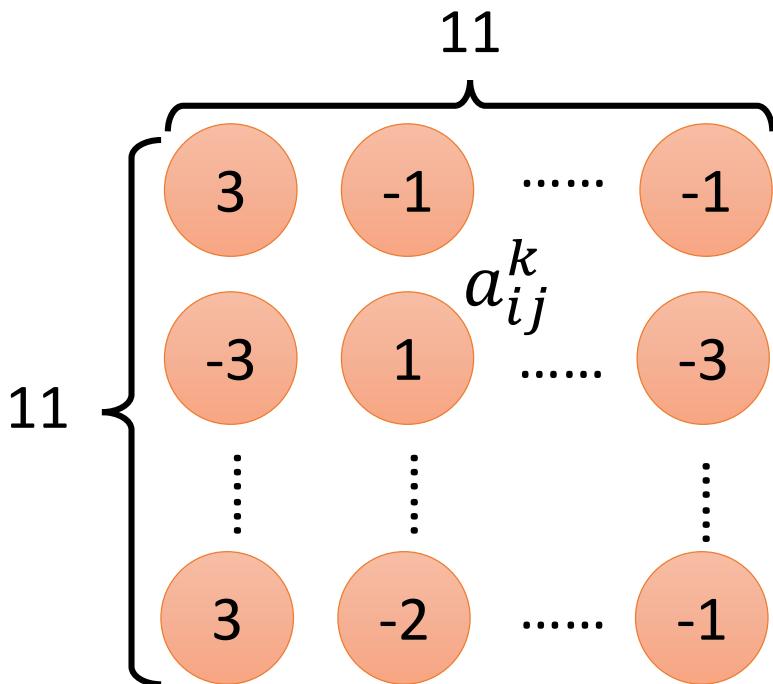
Live Demo

What does CNN learn?

The output of the k-th filter is a 11×11 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)

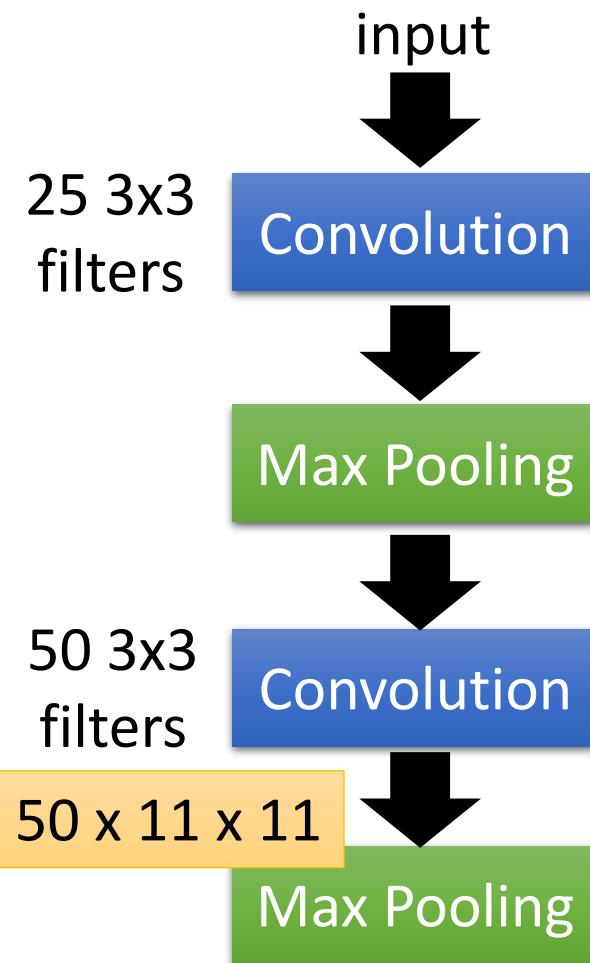
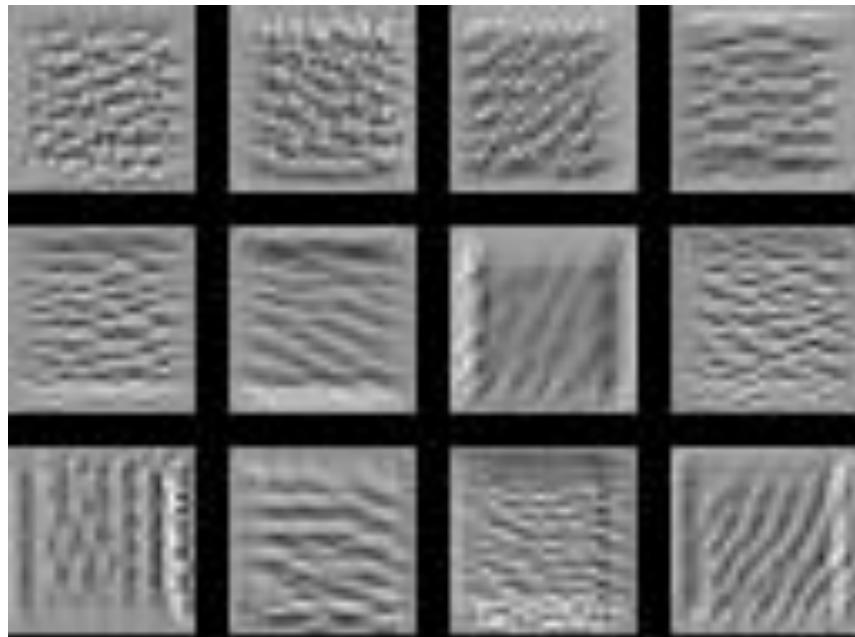


What does CNN learn?

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

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

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

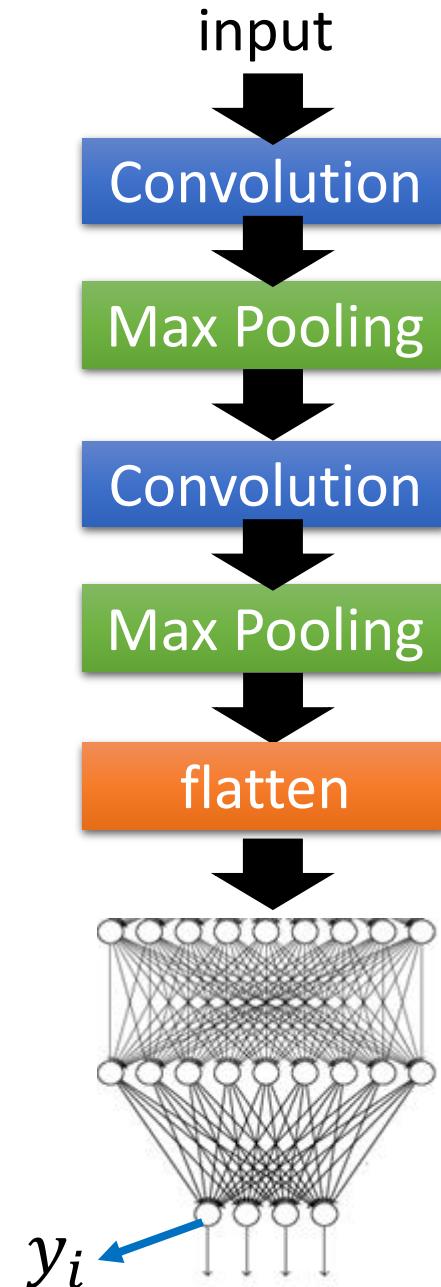
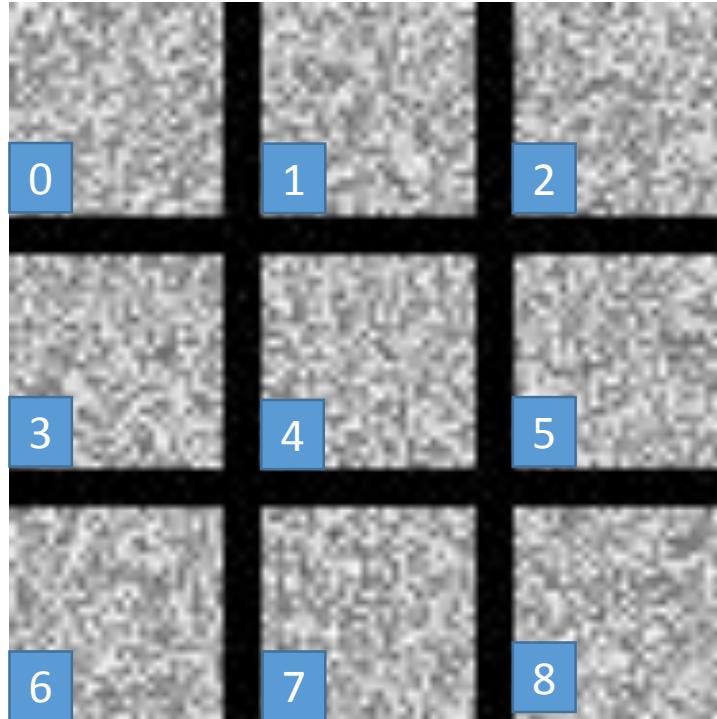


For each filter

What does CNN learn?

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

Can we see digits?

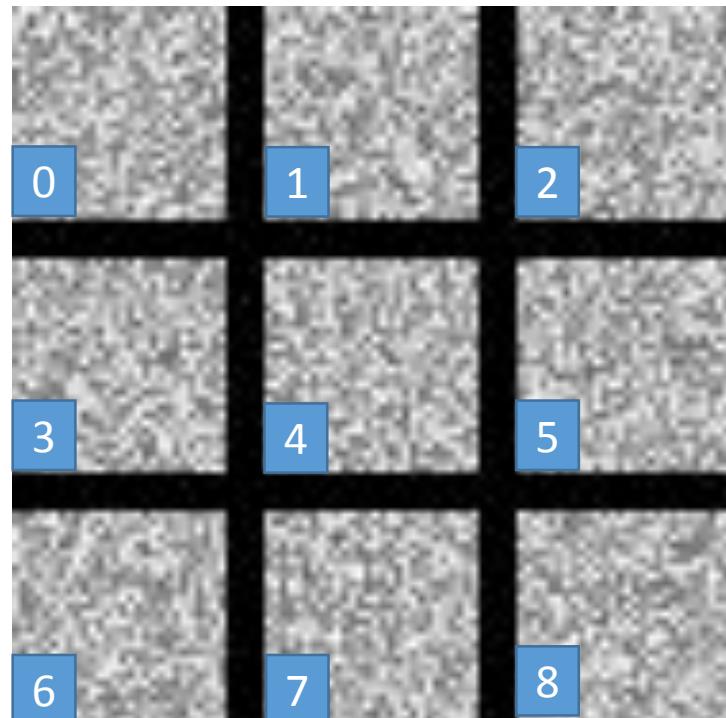


Deep Neural Networks are Easily Fooled

<https://www.youtube.com/watch?v=M2IebCN9Ht4>

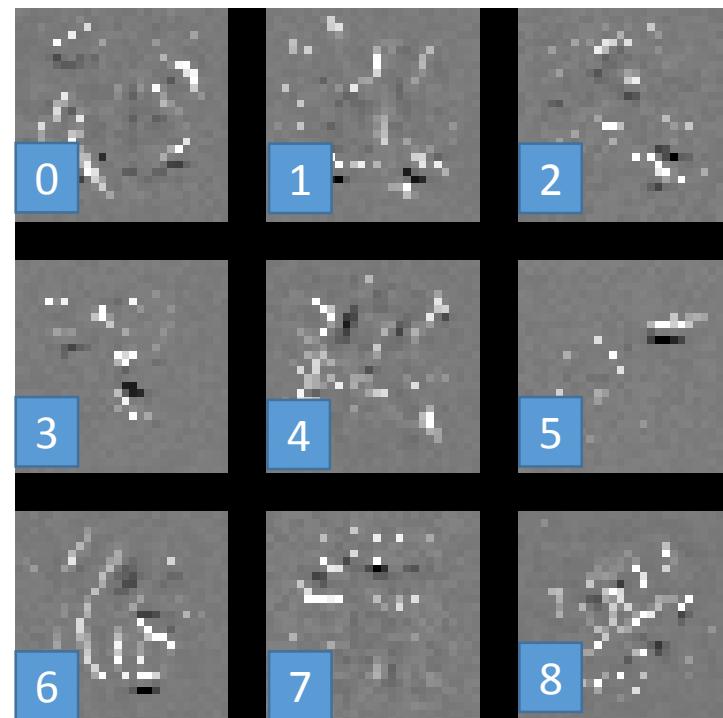
What does CNN learn?

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

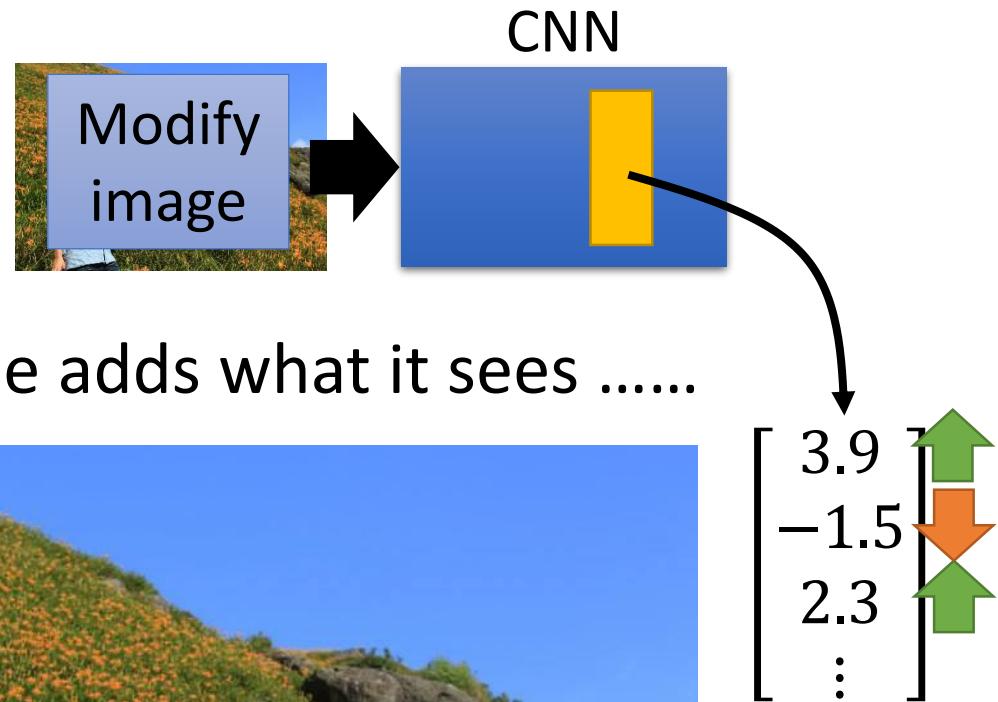


Over all
pixel values

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



Deep Dream

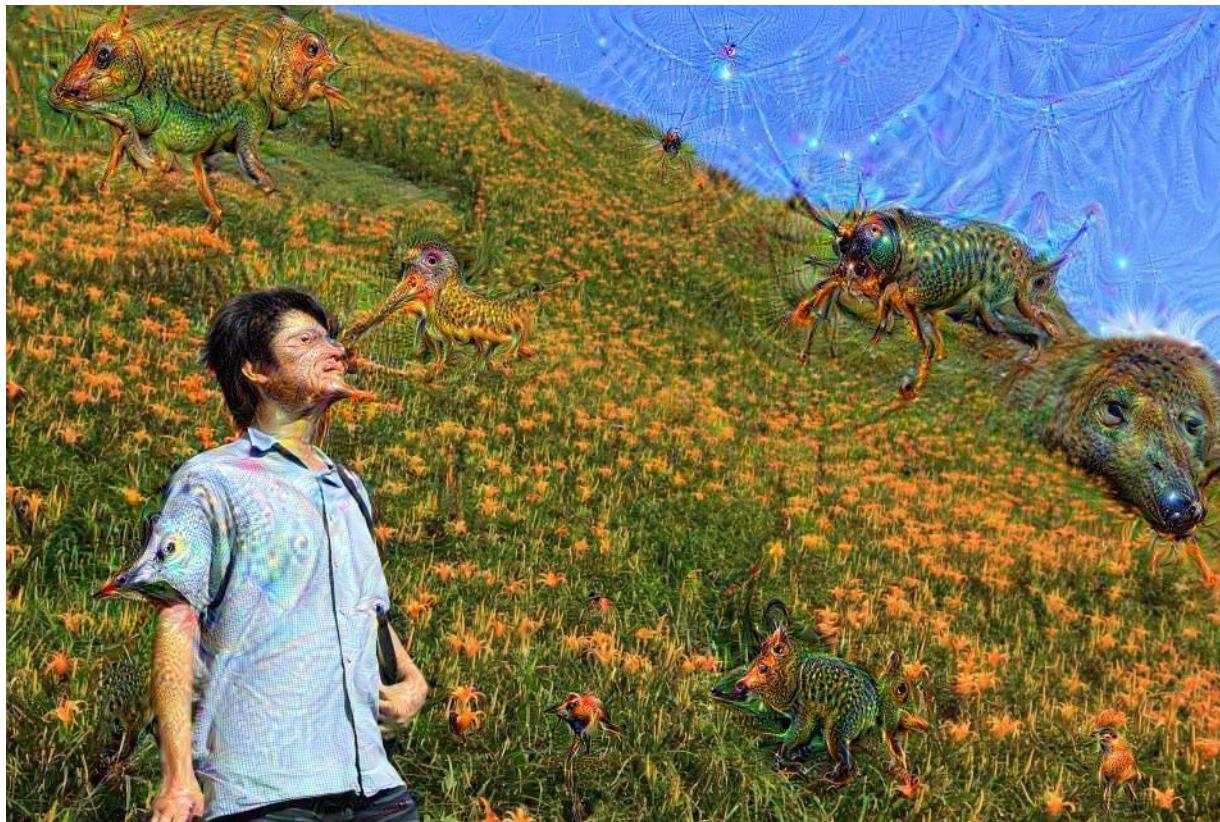


- Given a photo, machine adds what it sees



Deep Dream

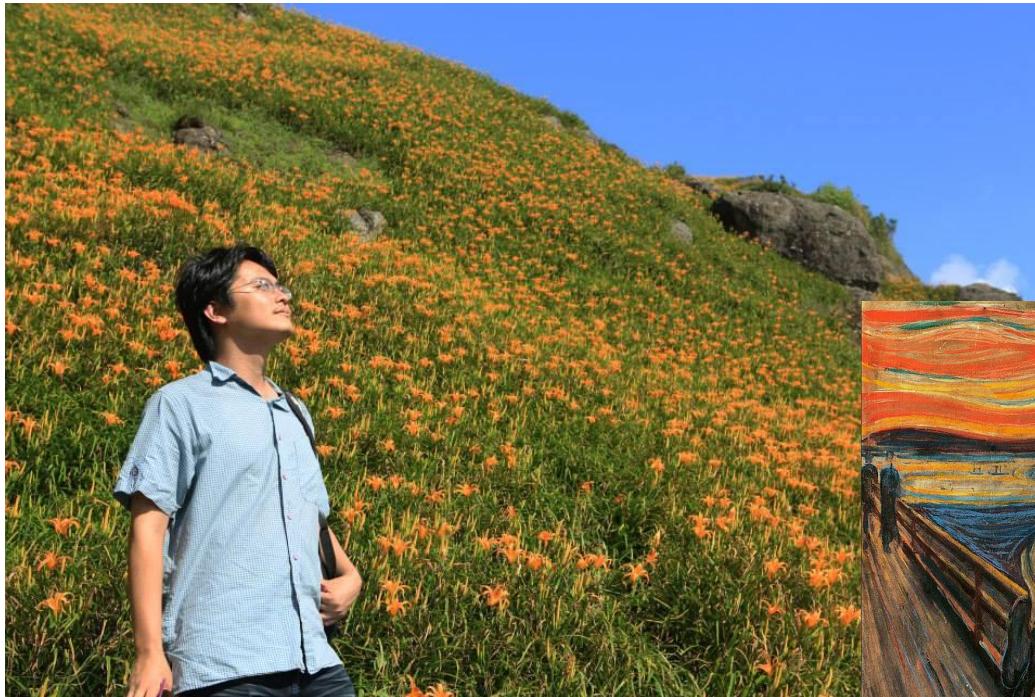
- Given a photo, machine adds what it sees



<http://deepdreamgenerator.com/>

Deep Style

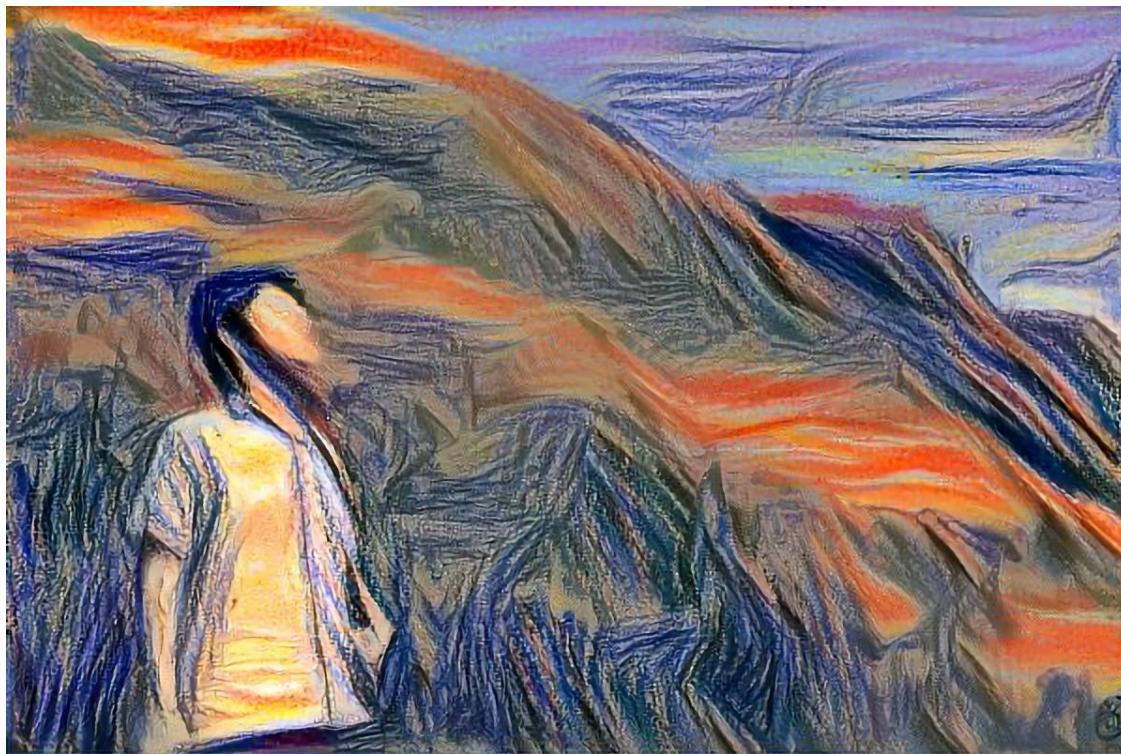
- Given a photo, make its style like famous paintings



<https://dreamscopeapp.com/>

Deep Style

- Given a photo, make its style like famous paintings

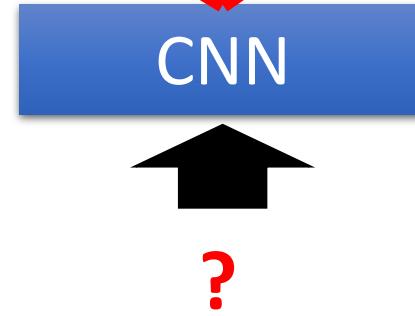
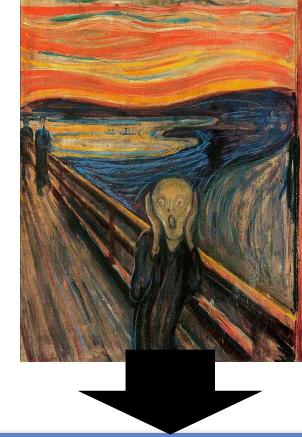


<https://dreamscopeapp.com/>

Deep Style

A Neural
Algorithm of
Artistic Style

<https://arxiv.org/abs/1508.06576>



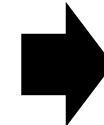
More Application: Playing Go



19 x 19 matrix
(image)



Network



Next move
(19 x 19
positions)

Black: 1
white: -1
none: 0

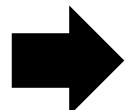
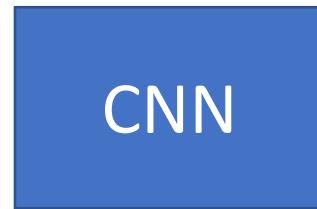
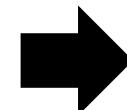
Fully-connected feedforward
network can be used

But CNN performs much better.

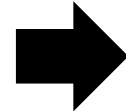
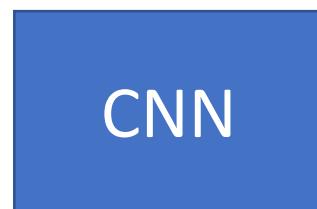
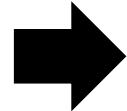
More Application: Playing Go

Training: record of previous plays

黑: 5之五 → 白: 天元 → 黑: 五之5 ...



Target:
“天元” = 1
else = 0

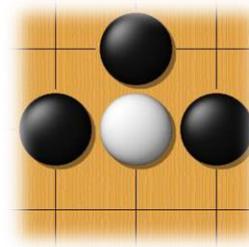


Target:
“五之5” = 1
else = 0

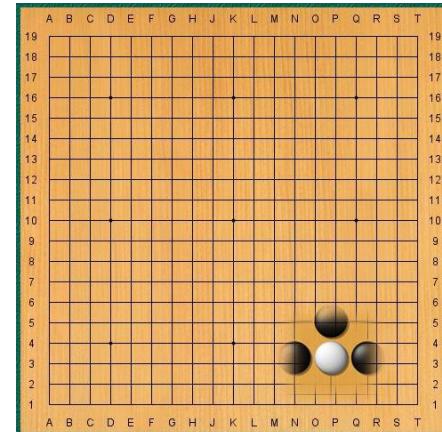
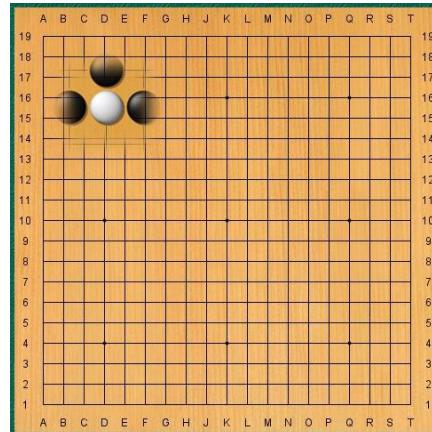
Why CNN for playing Go?

- Some patterns are much smaller than the whole image

Alpha Go uses 5×5 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 **Alpha Go does not use Max Pooling** Extended Data Table 3 additionally show the results of training with $k = 128, 256$ and 384 filters.

Variants of Neural Networks

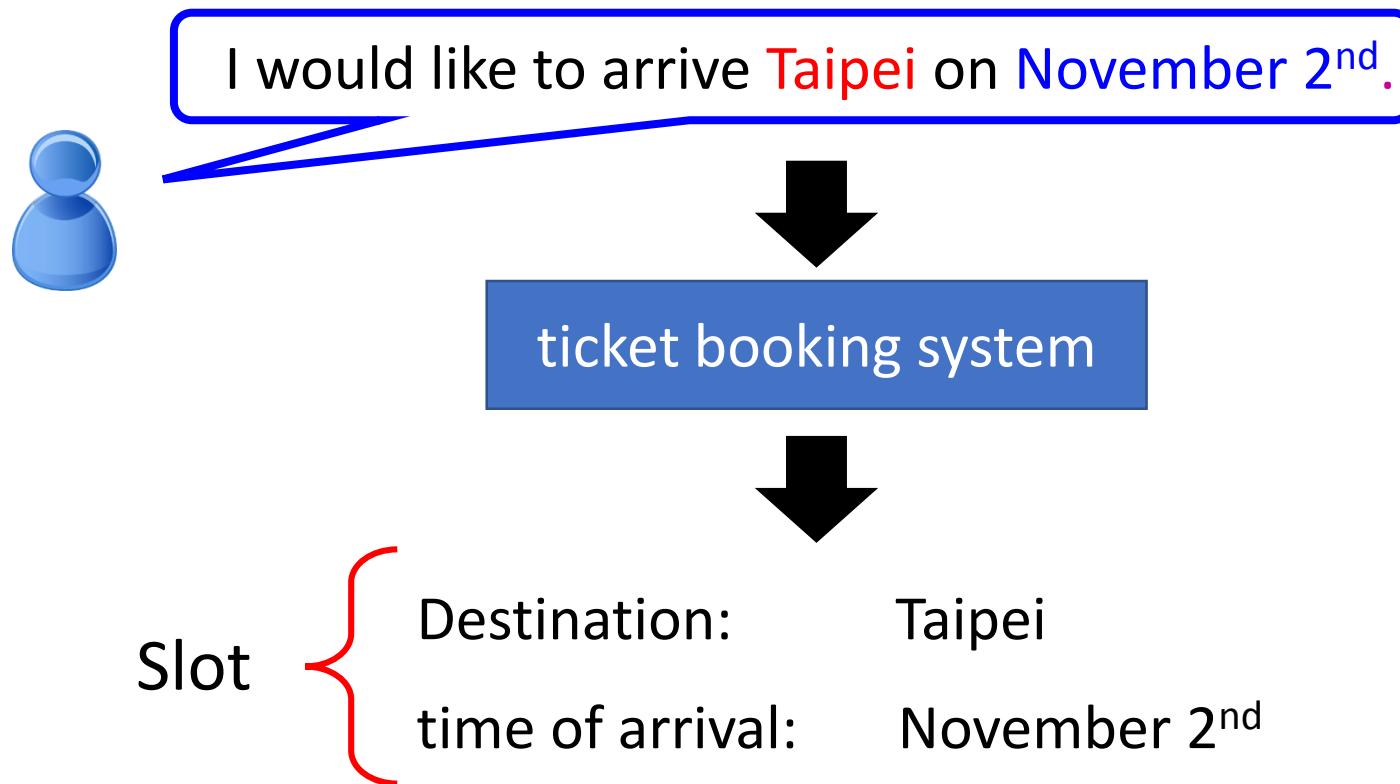
Convolutional Neural
Network (CNN)

Recurrent Neural Network
(RNN)

Neural Network with Memory

Example Application

- Slot Filling

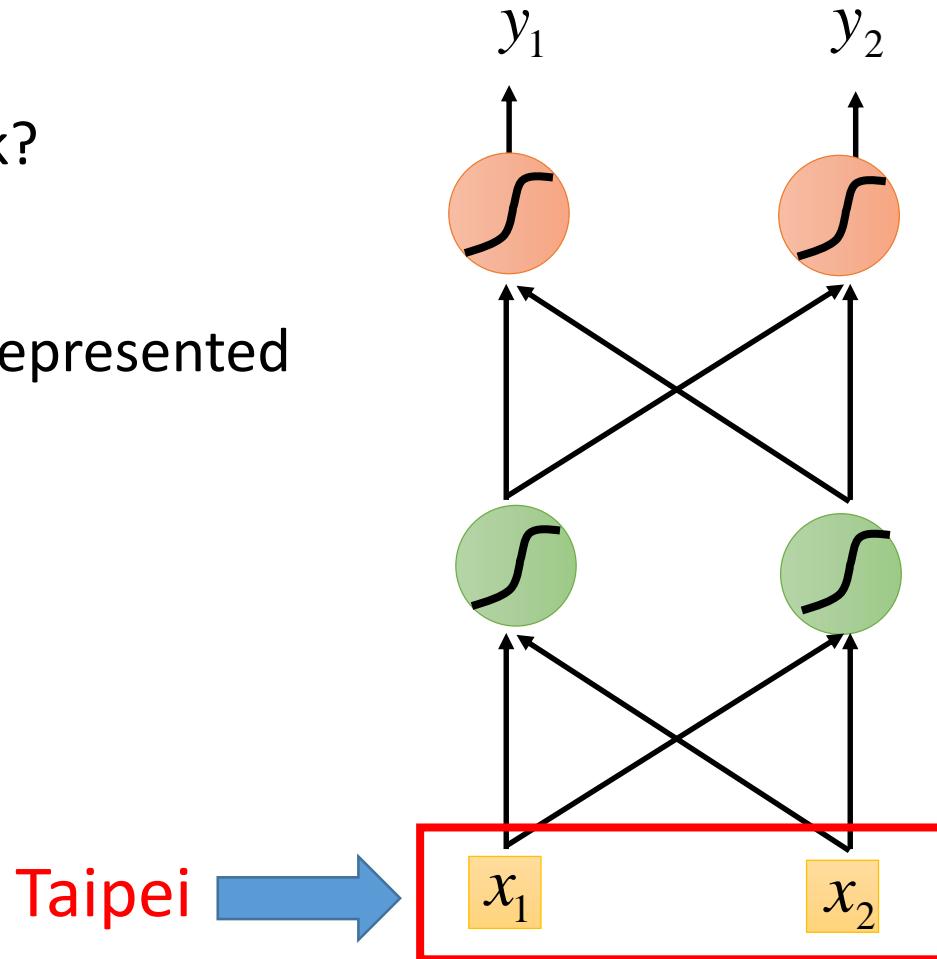


Example Application

Solving slot filling by
Feedforward network?

Input: a word

(Each word is represented
as a vector)



1-of-N encoding

How to represent each word as a vector?

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

The vector is lexicon size.

Each dimension corresponds
to a word in the lexicon

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

$$\text{apple} = [1 \ 0 \ 0 \ 0 \ 0]$$

$$\text{bag} = [0 \ 1 \ 0 \ 0 \ 0]$$

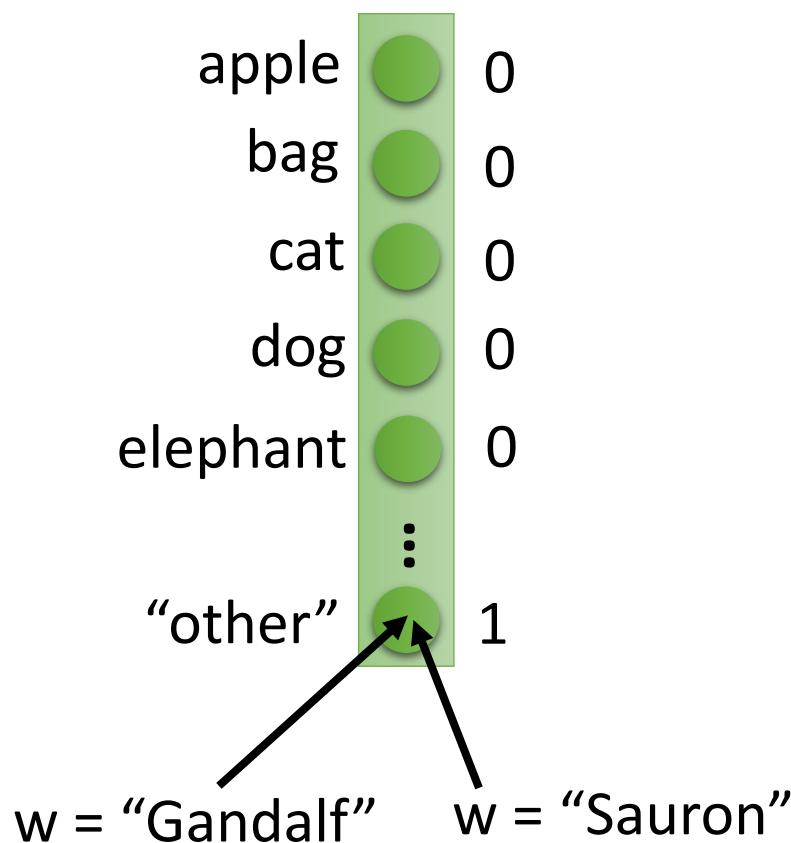
$$\text{cat} = [0 \ 0 \ 1 \ 0 \ 0]$$

$$\text{dog} = [0 \ 0 \ 0 \ 1 \ 0]$$

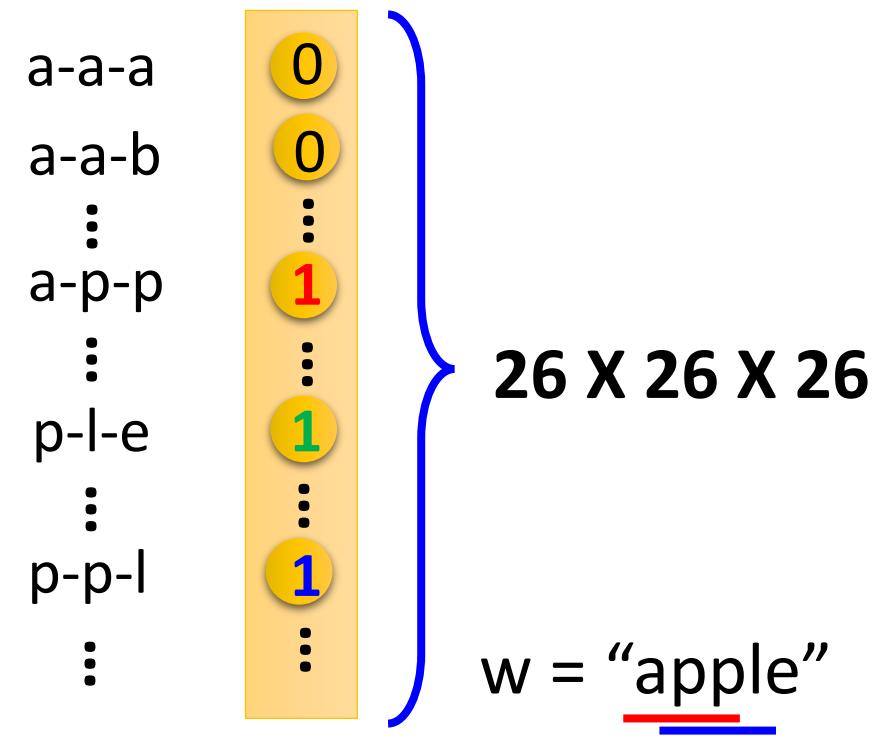
$$\text{elephant} = [0 \ 0 \ 0 \ 0 \ 1]$$

Beyond 1-of-N encoding

Dimension for “Other”



Word hashing



Example Application

Solving slot filling by
Feedforward network?

Input: a word

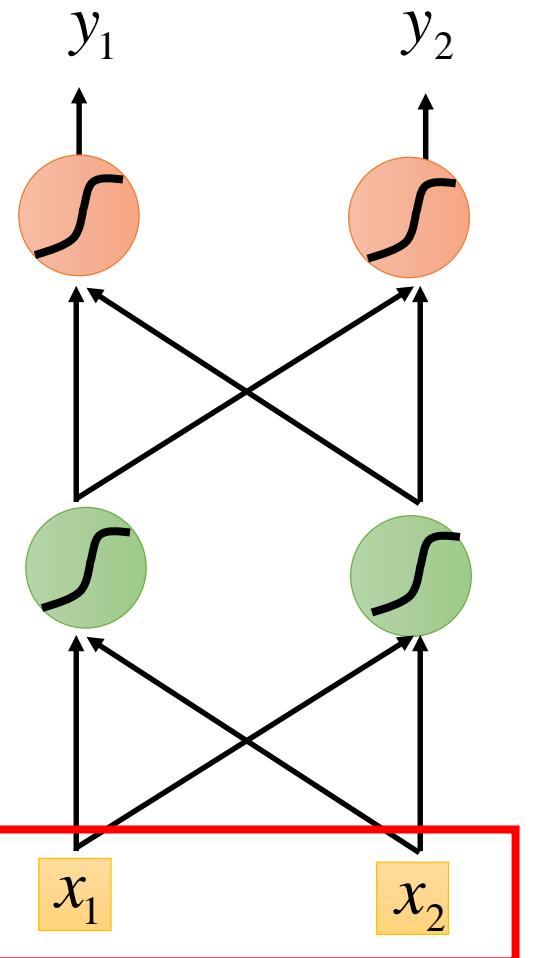
(Each word is represented
as a vector)

Output:

Probability distribution that
the input word belonging to
the slots

Taipei

dest
time of
departure



Example Application

arrive Taipei on November 2nd

other dest other time time

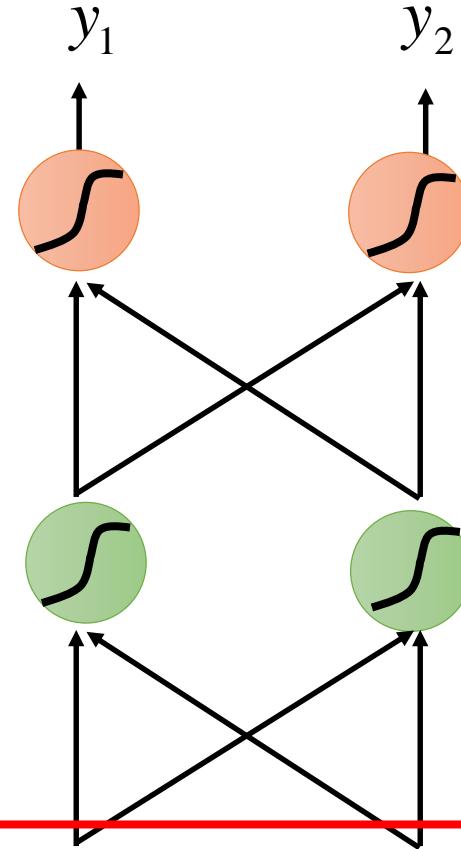
Problem?

leave Taipei on November 2nd

place of departure

Neural network
needs memory!

dest time of
departure



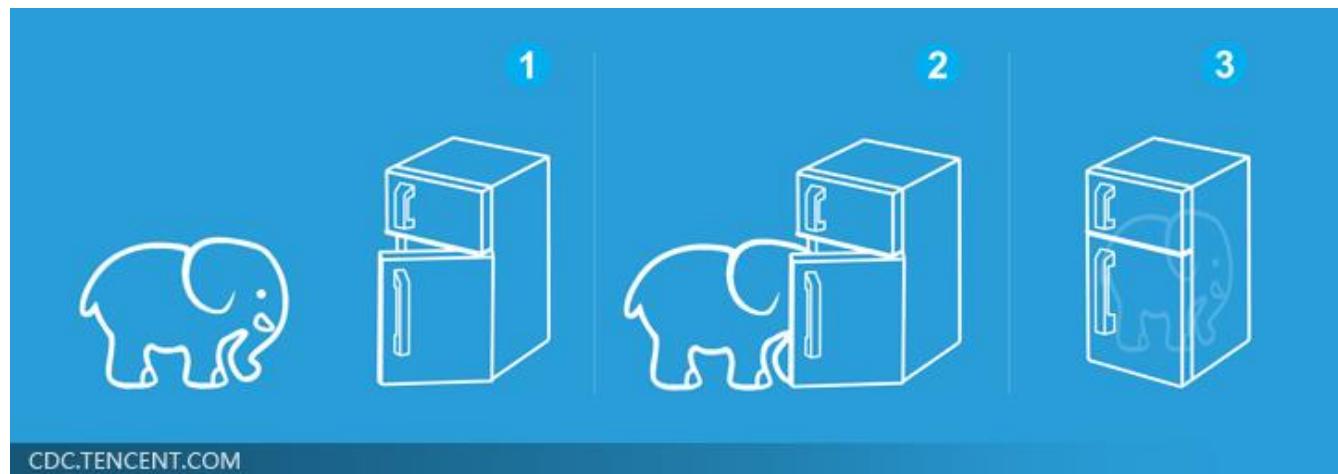
Taipei 

 x_1 x_2

Three Steps for Deep Learning

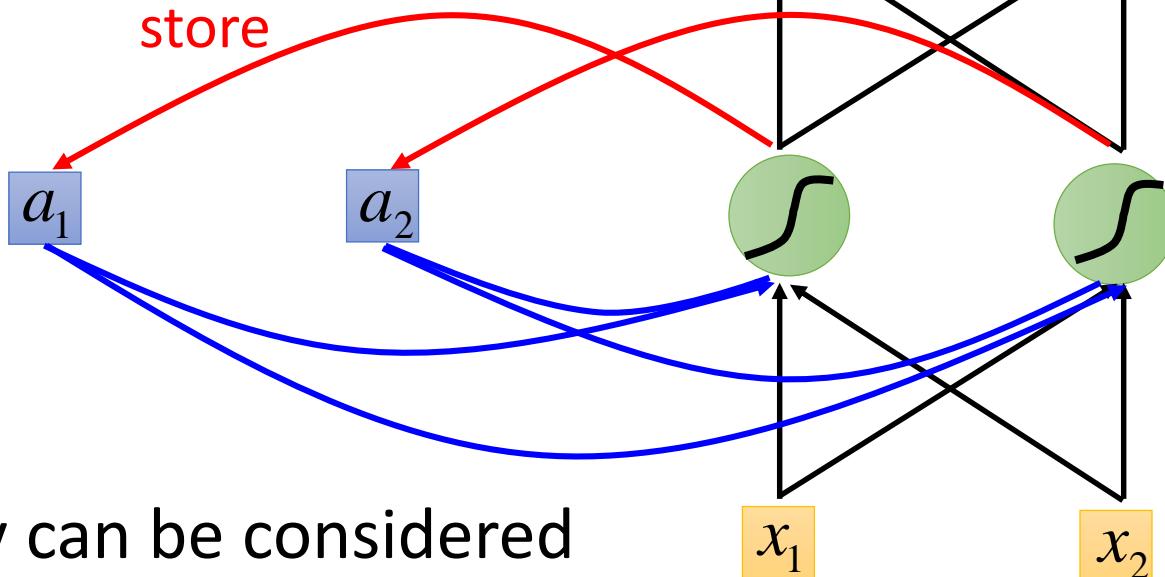


Deep Learning is so simple



Recurrent Neural Network (RNN)

The output of hidden layer are stored in the memory.



Memory can be considered as another input.

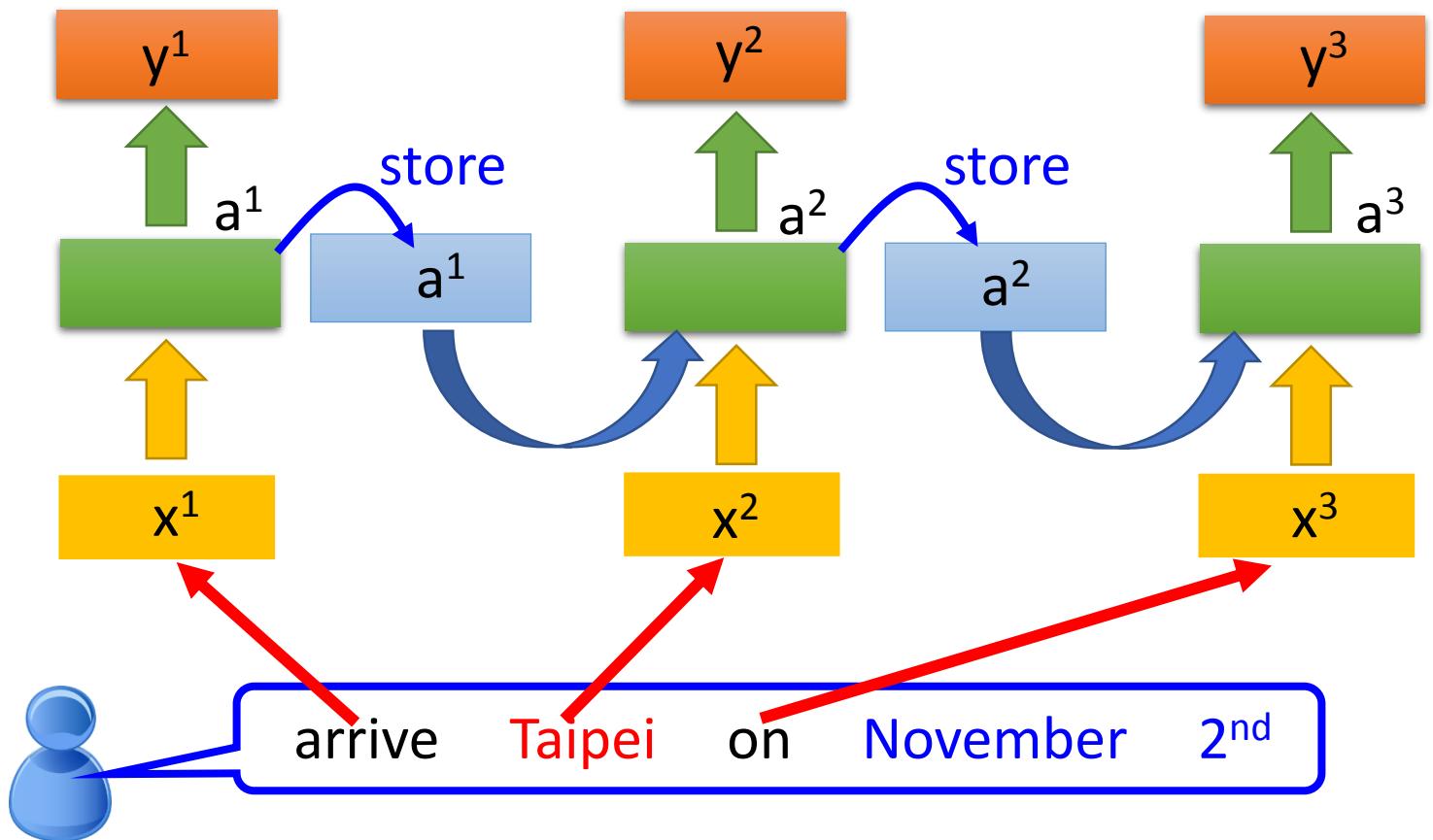
RNN

The same network is used again and again.

Probability of
“arrive” in each slot

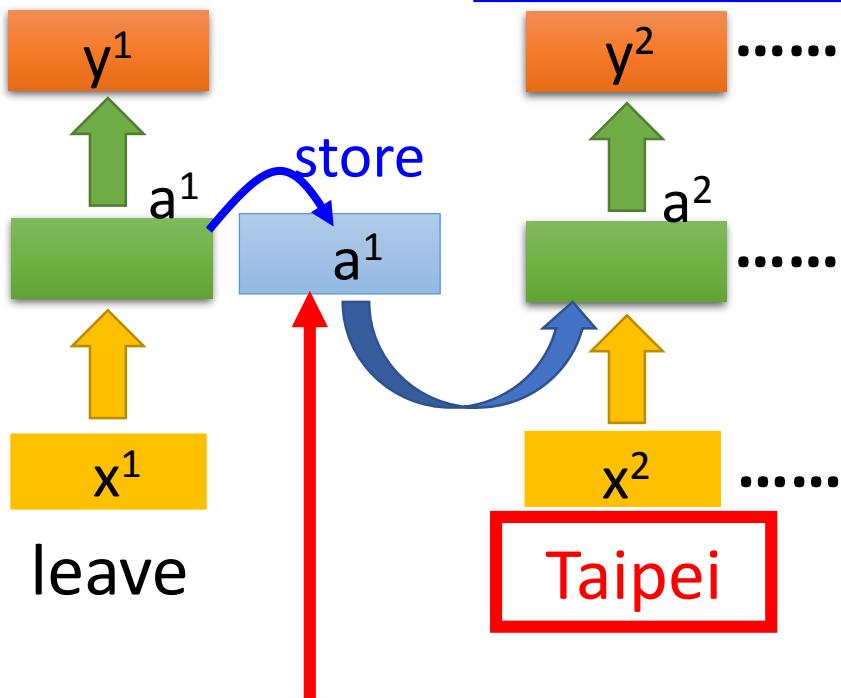
Probability of
“Taipei” in each slot

Probability of
“on” in each slot



RNN

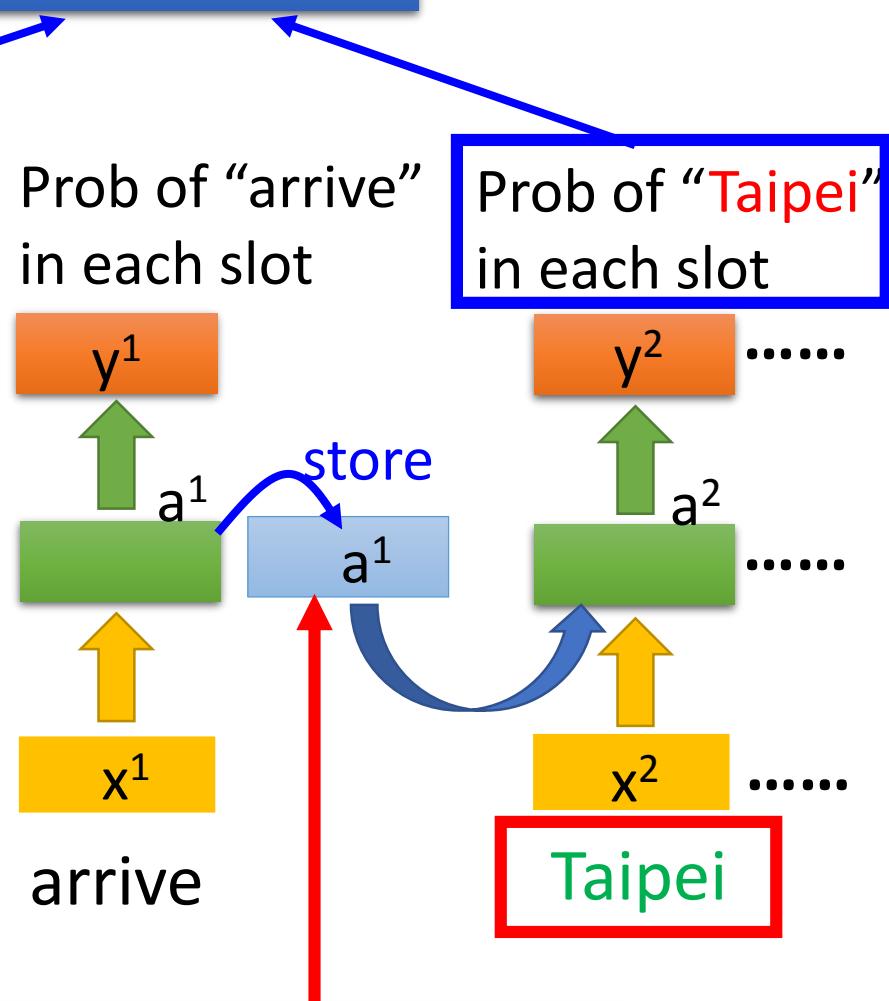
Prob of “leave”
in each slot



Prob of “Taipei”
in each slot

Different

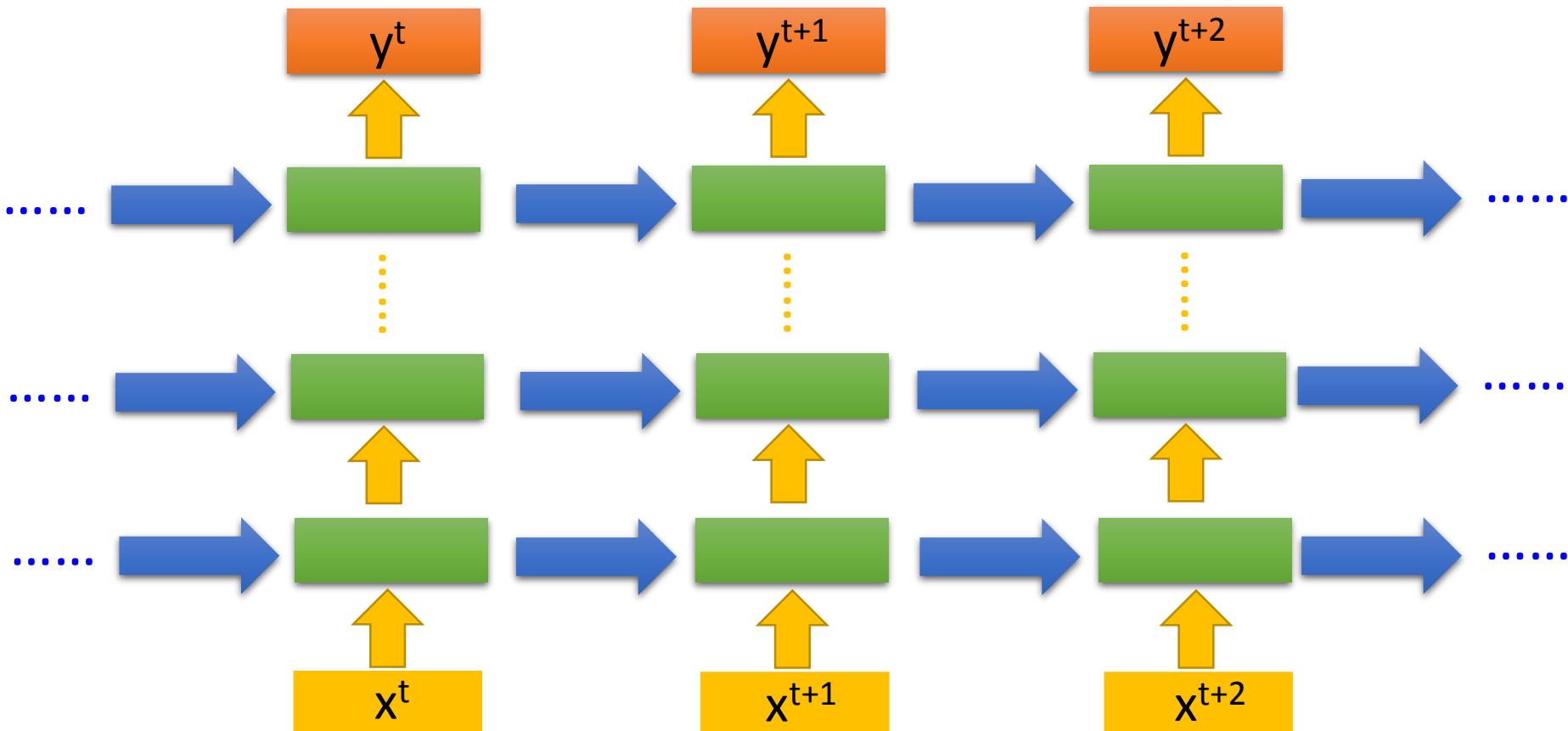
Prob of “arrive”
in each slot



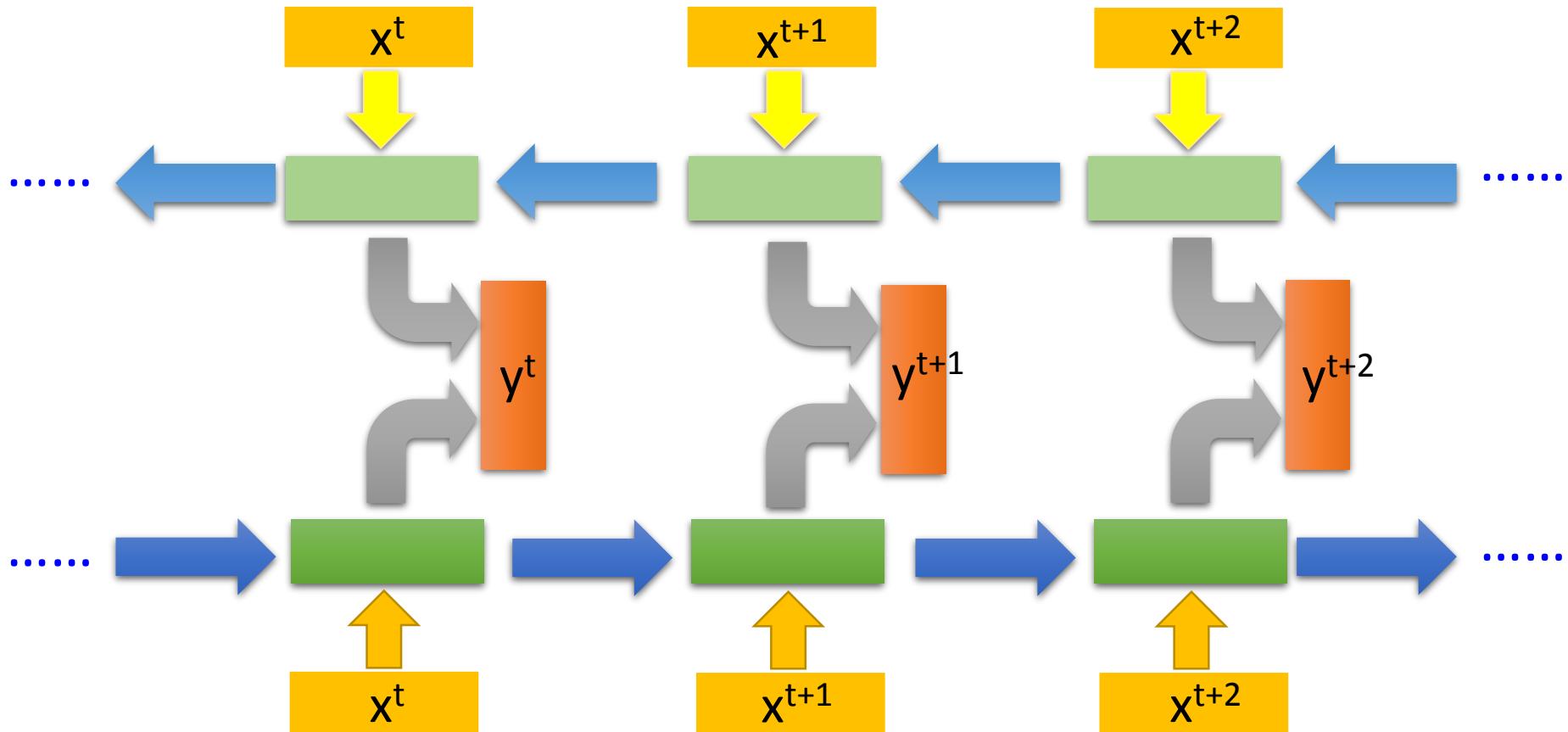
Prob of “Taipei”
in each slot

The values stored in the memory is different.

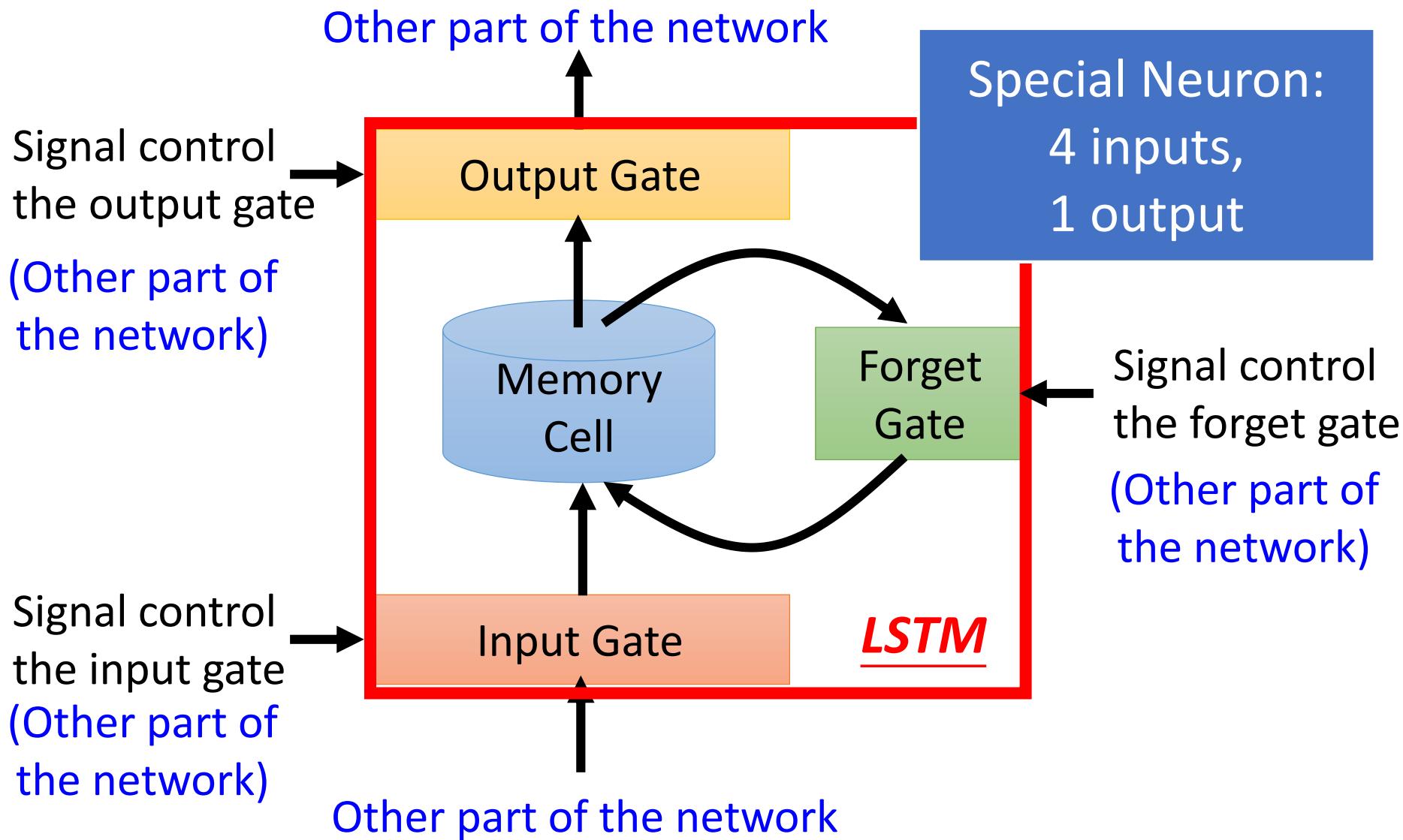
Of course it can be deep ...

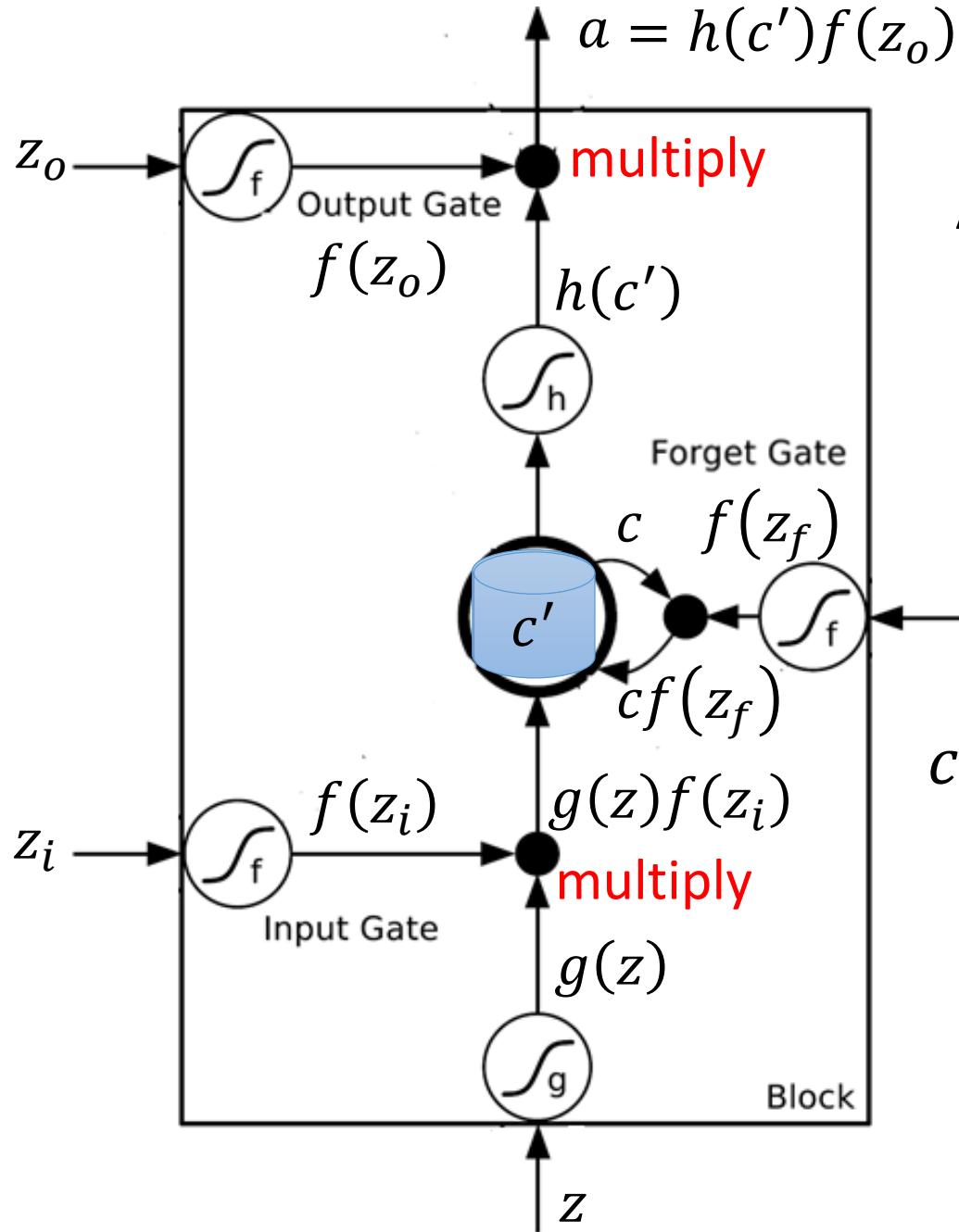


Bidirectional RNN



Long Short-term Memory (LSTM)



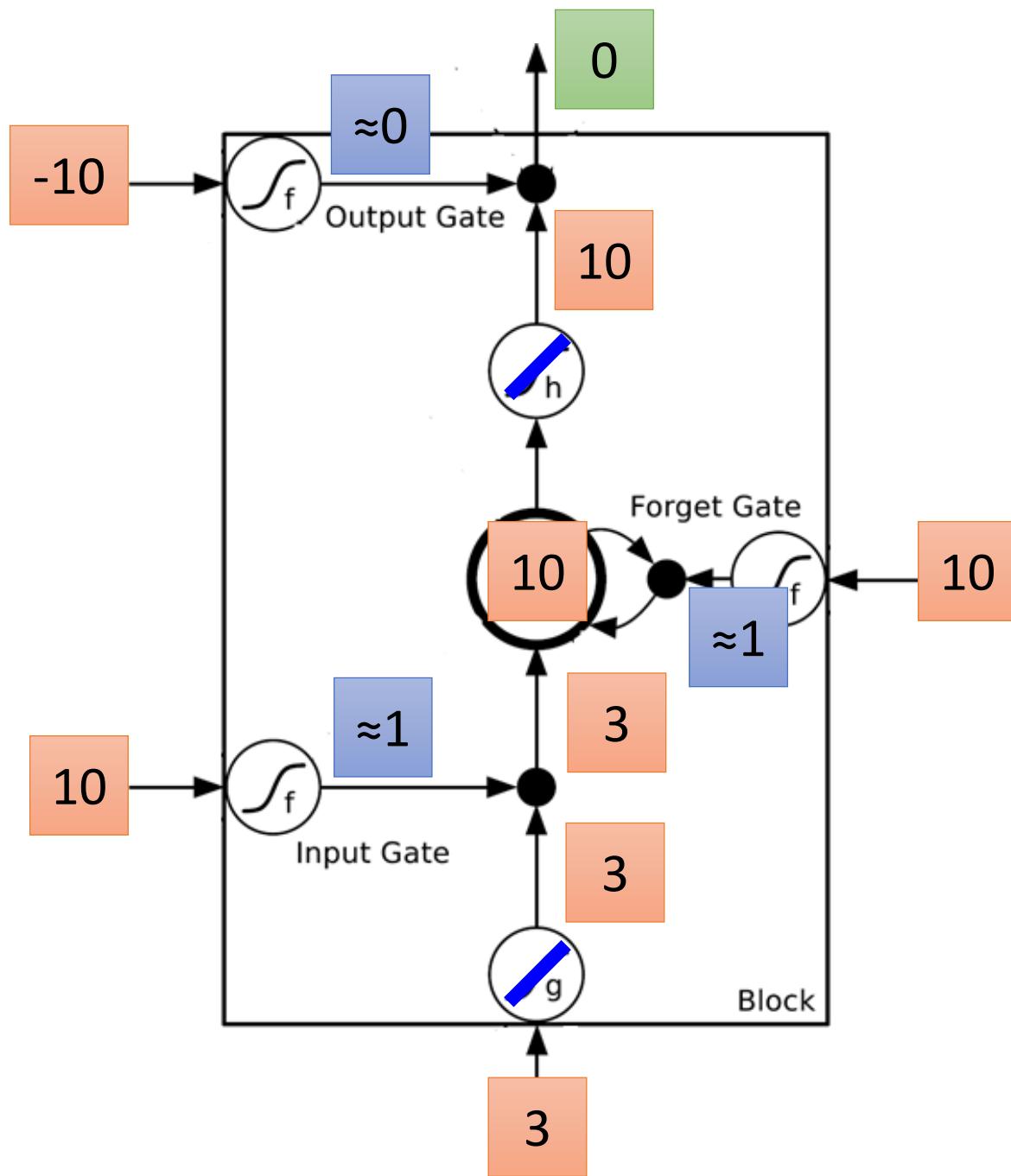


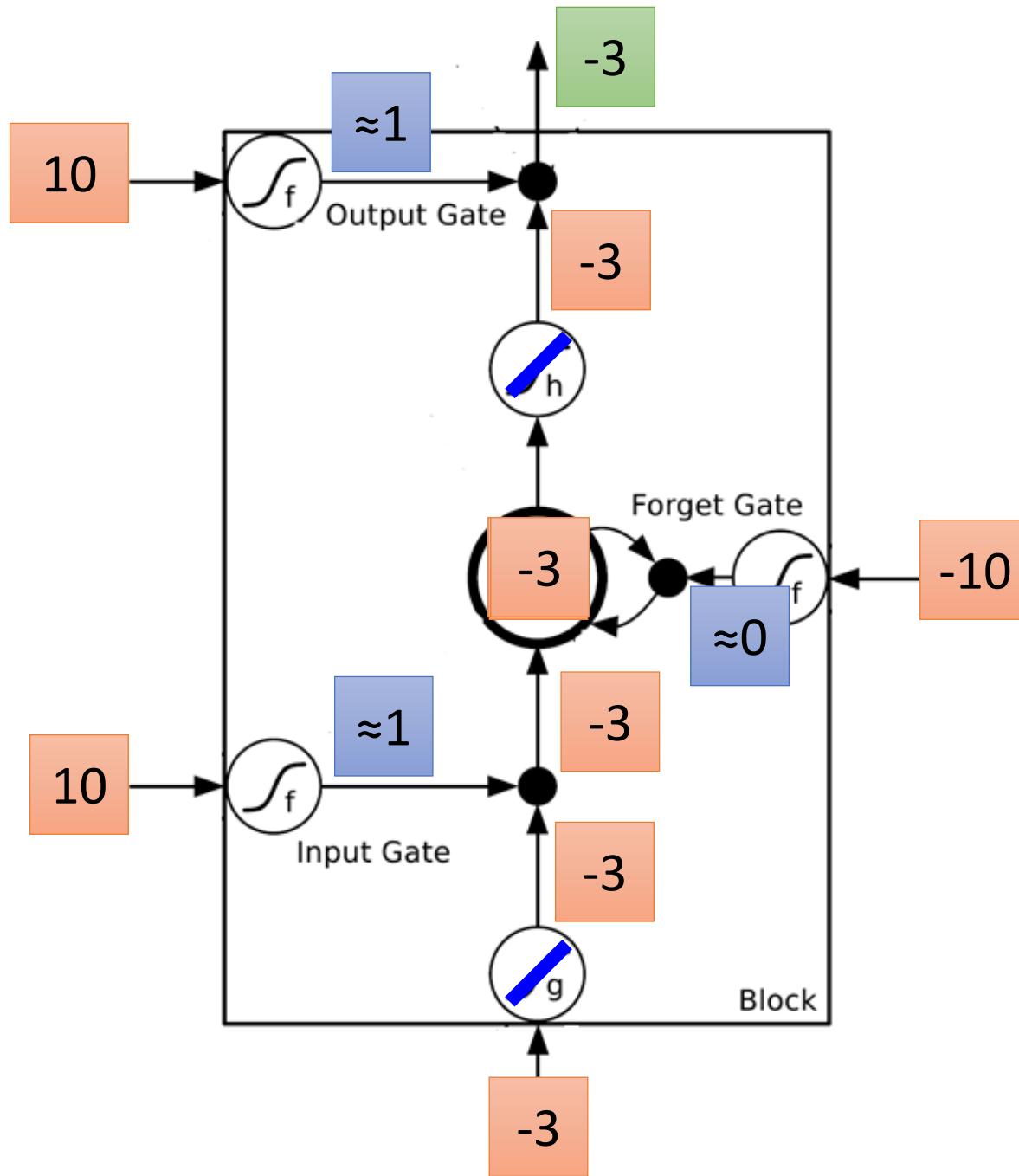
Activation function f is usually a sigmoid function

Between 0 and 1

Mimic open and close gate

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

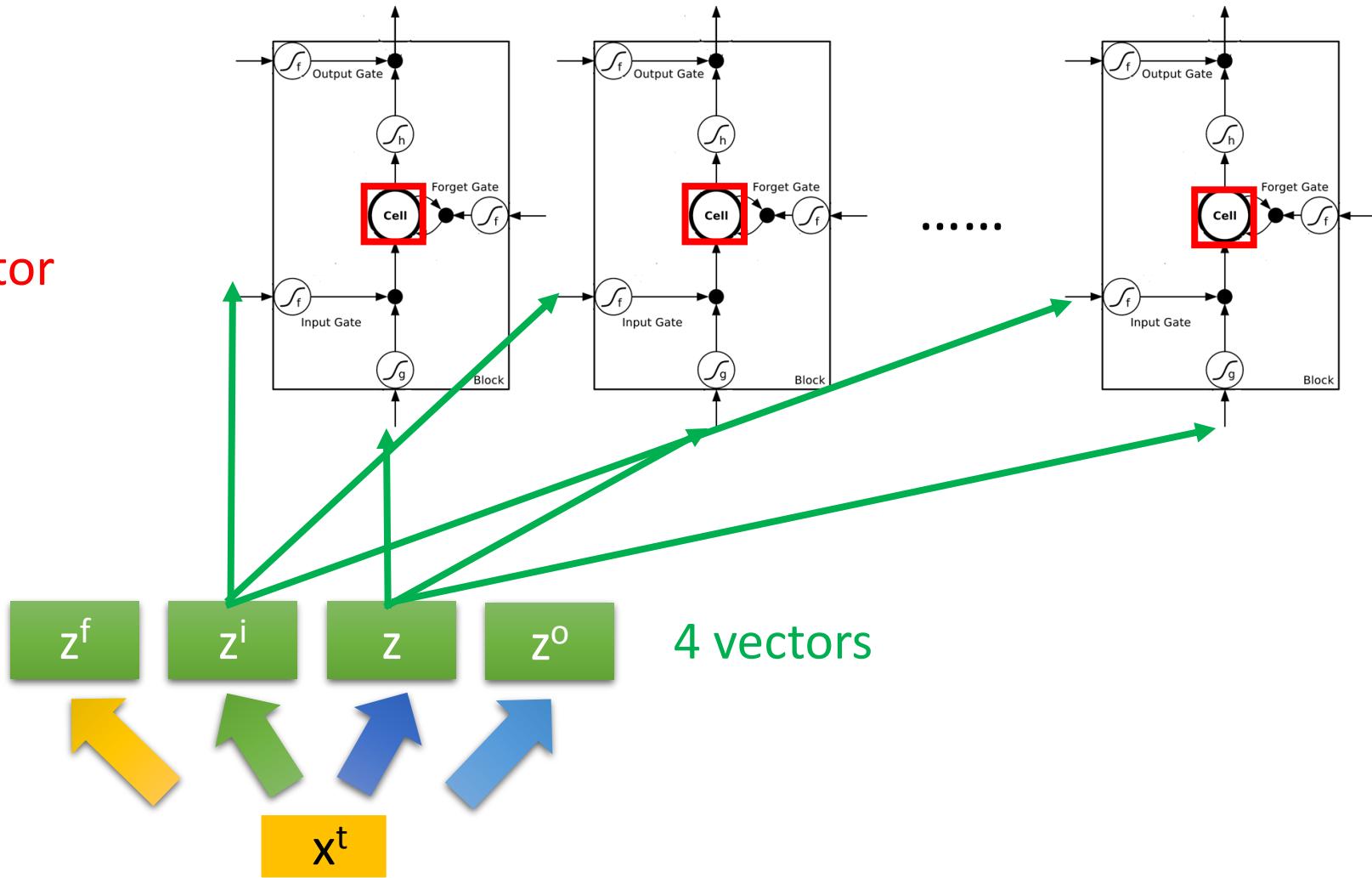




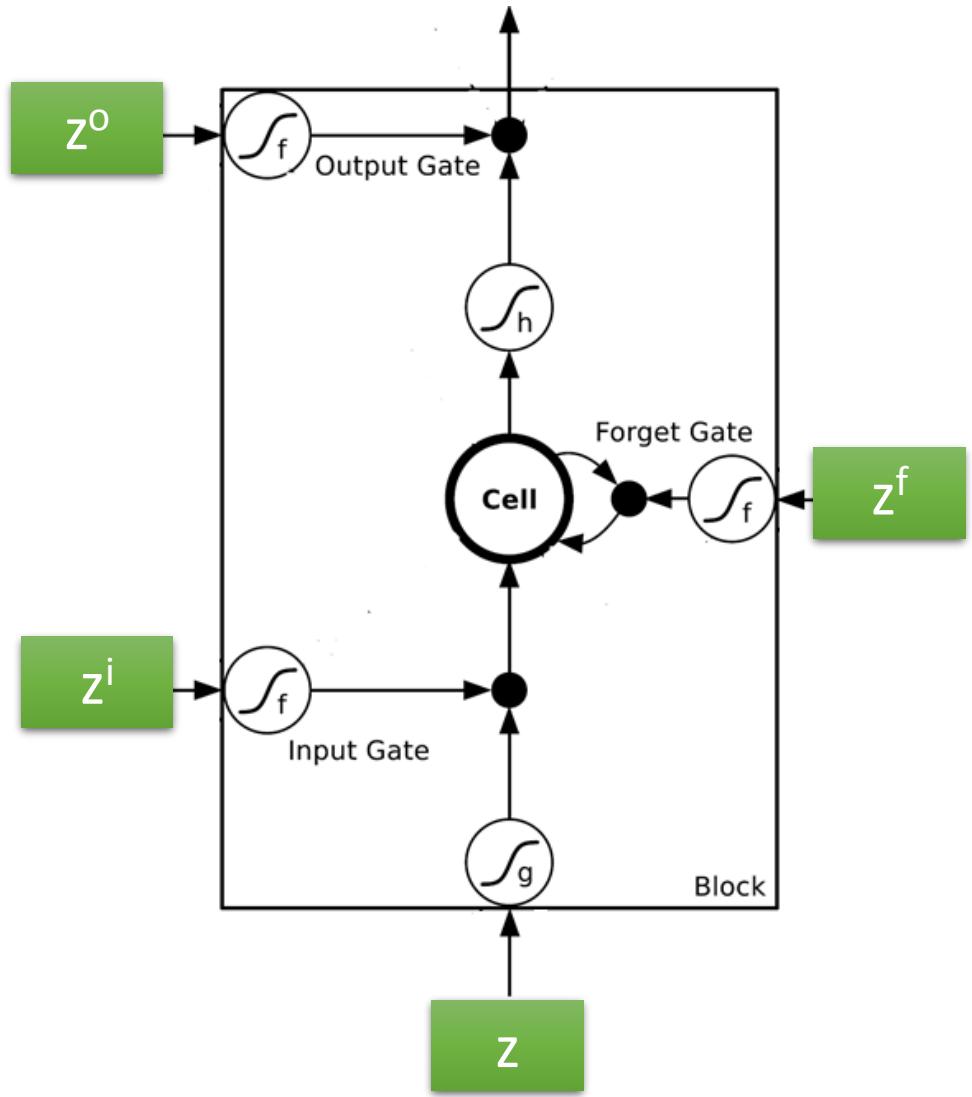
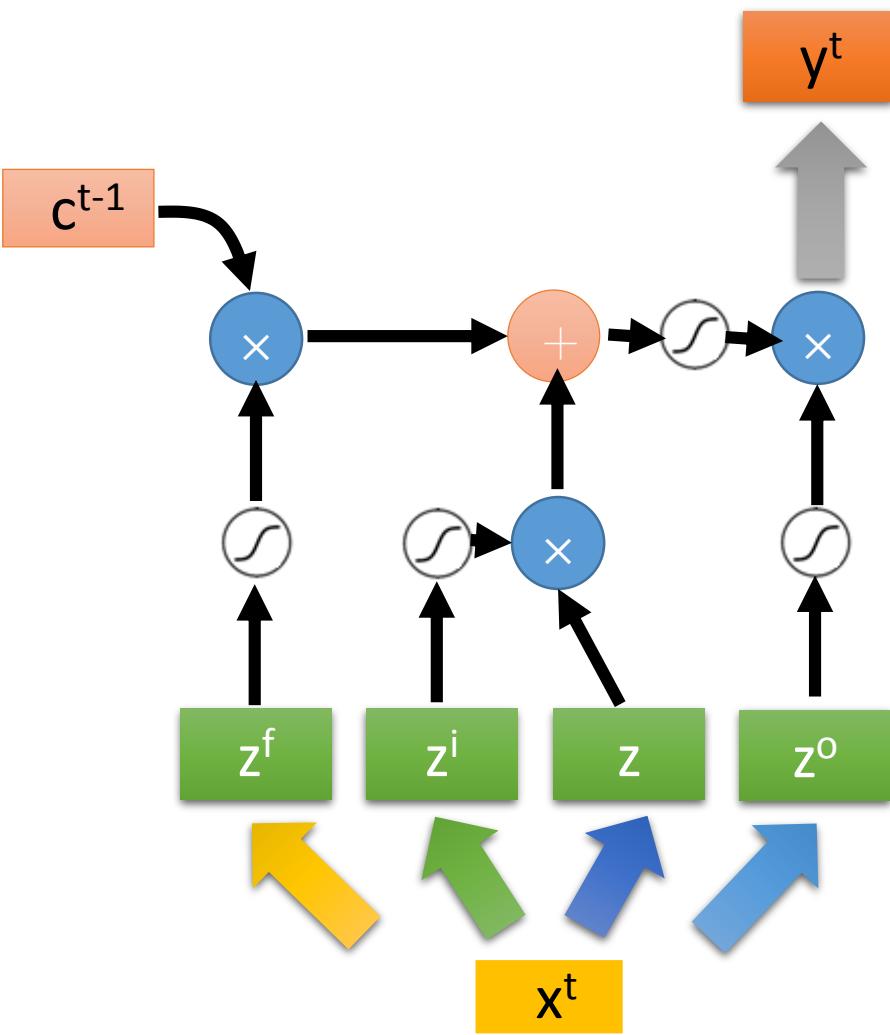
LSTM

C^{t-1}

vector

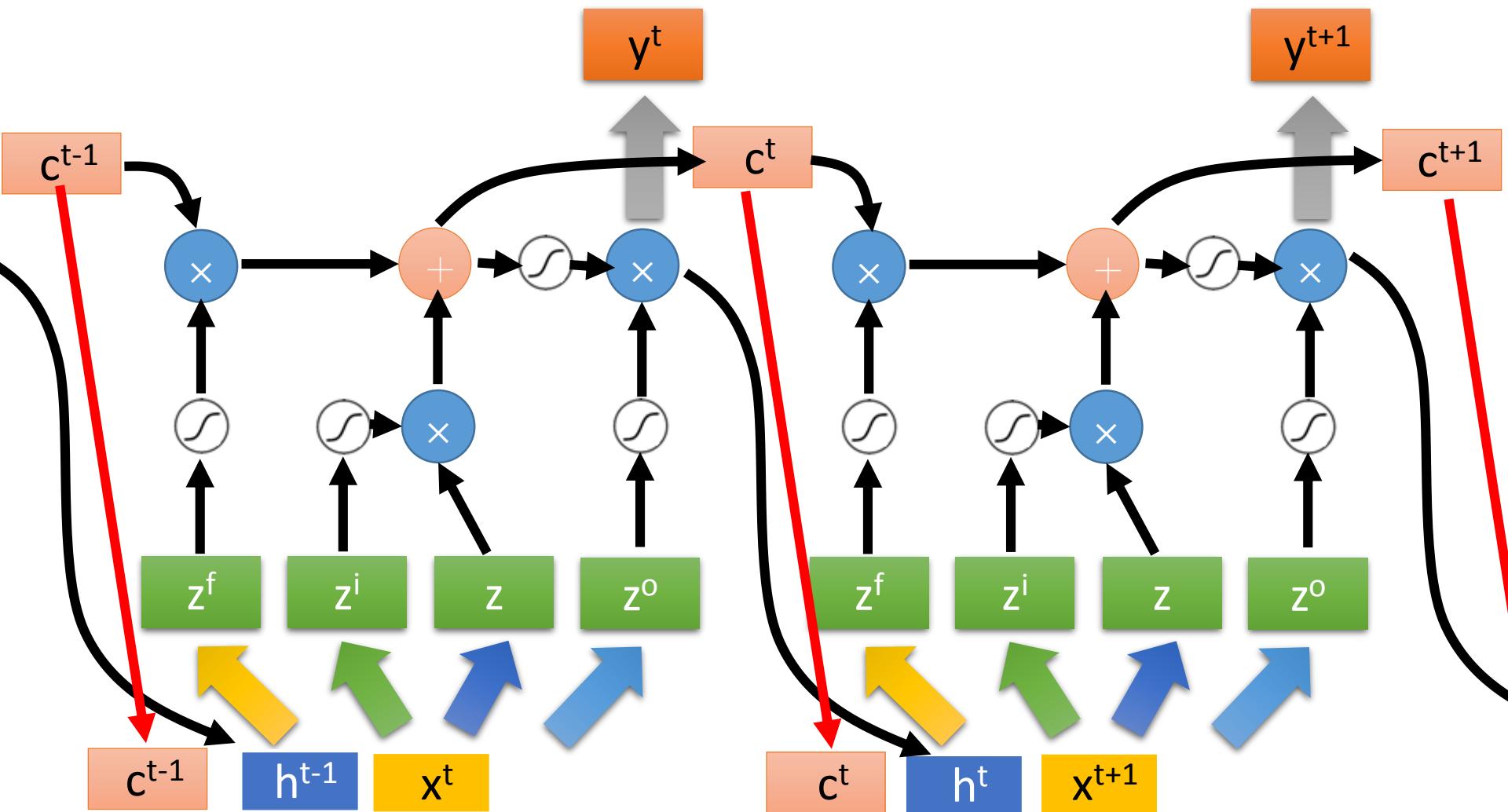


LSTM

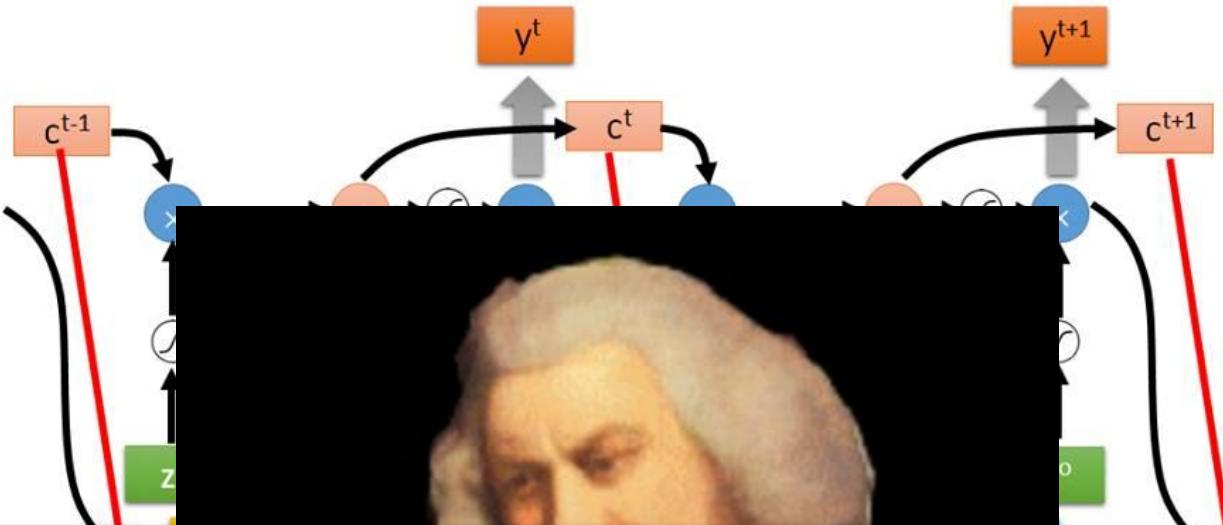


LSTM

Extension: “peephole”



Multiple-layer LSTM



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

Keras supports
“LSTM”, “GRU”, “SimpleRNN” layers

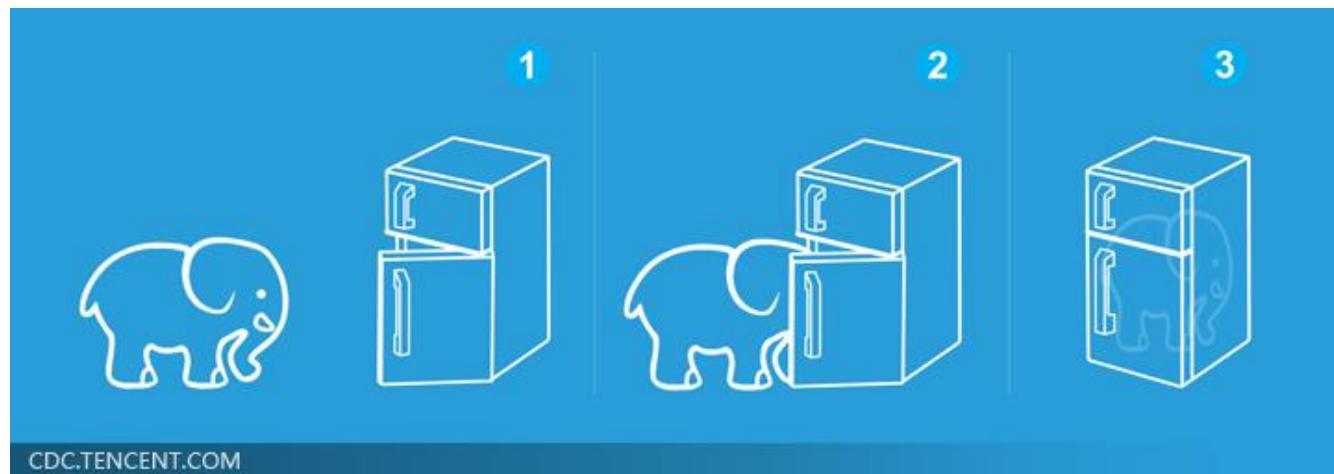
This is quite
standard now.



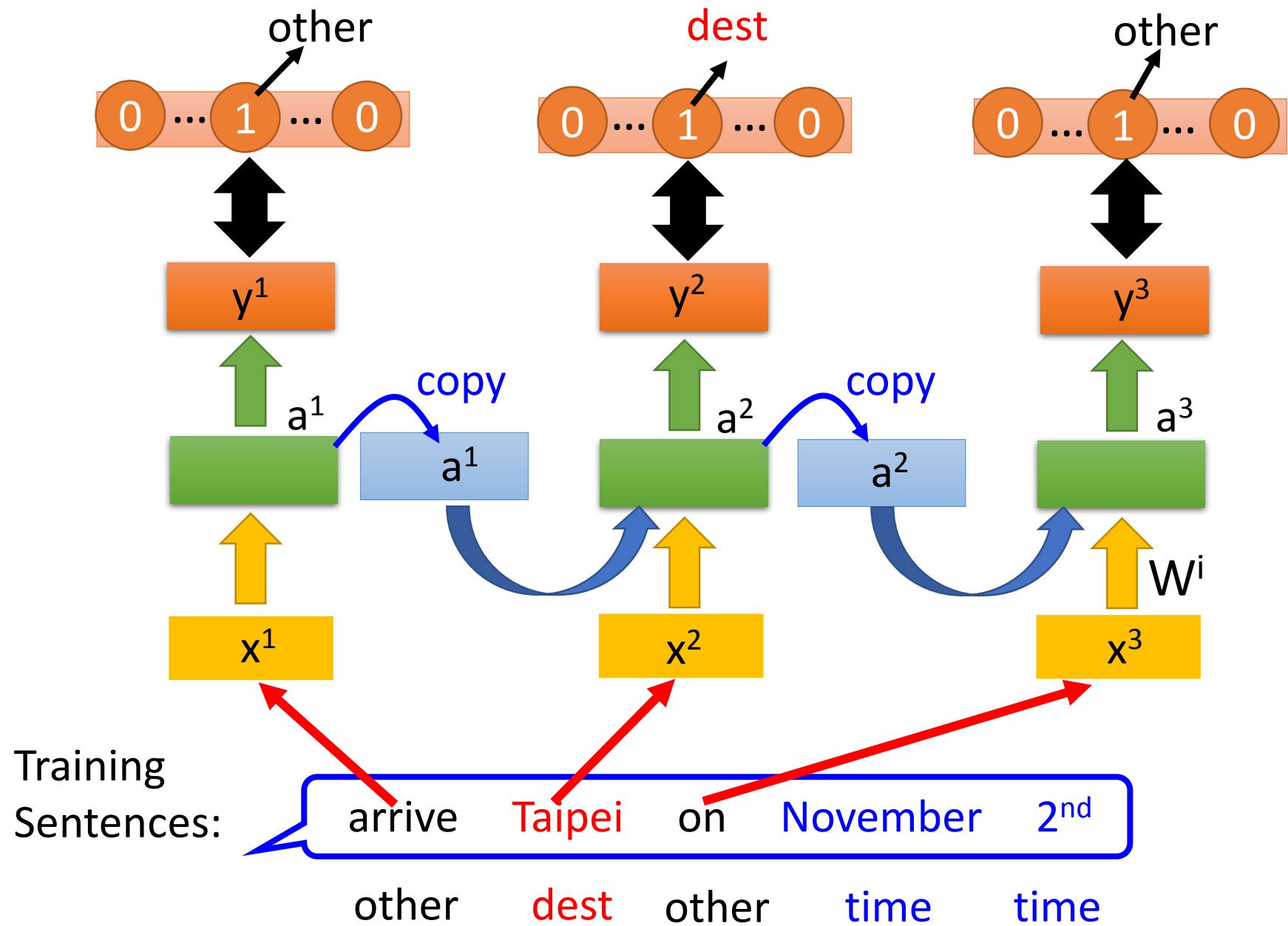
Three Steps for Deep Learning



Deep Learning is so simple



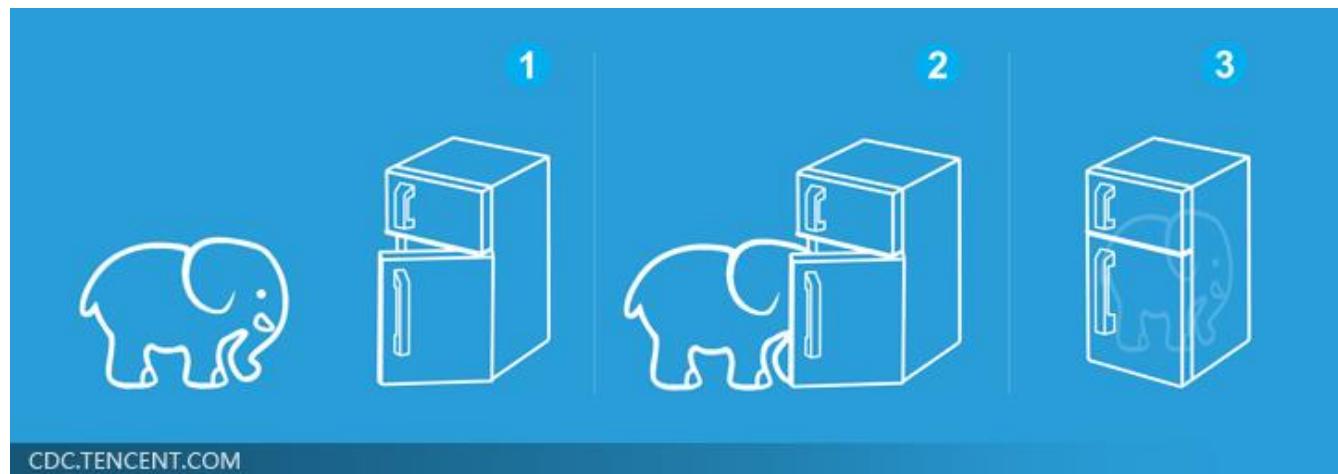
Learning Target



Three Steps for Deep Learning

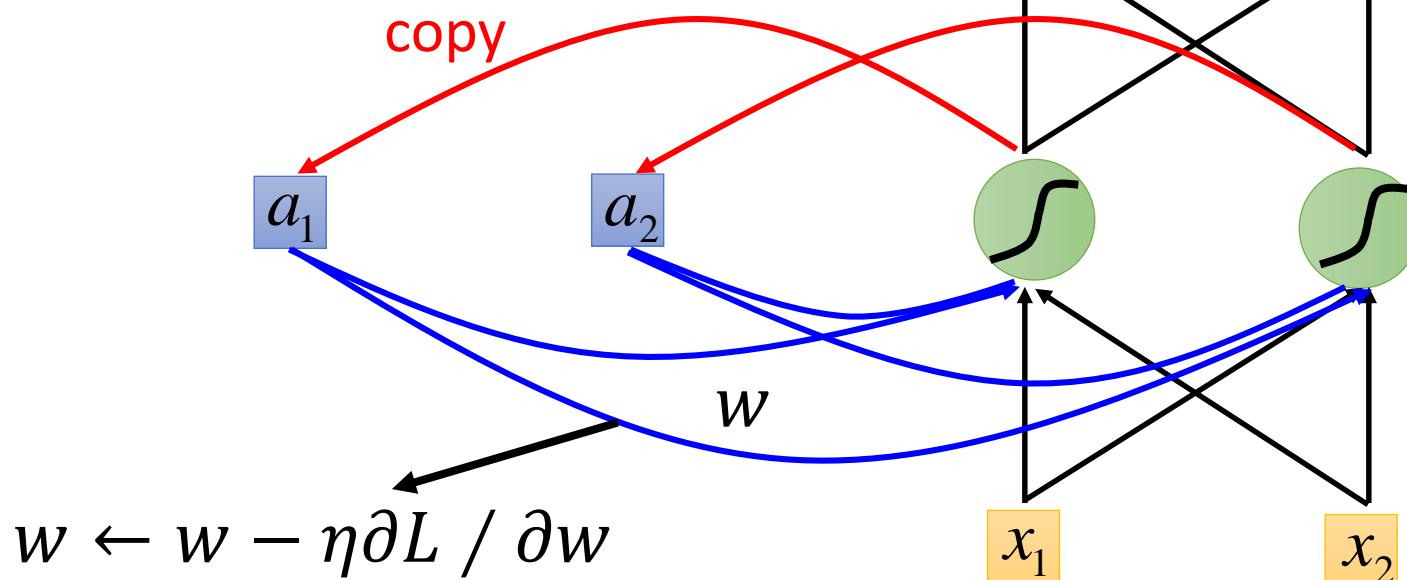


Deep Learning is so simple



Learning

Backpropagation
through time (BPTT)

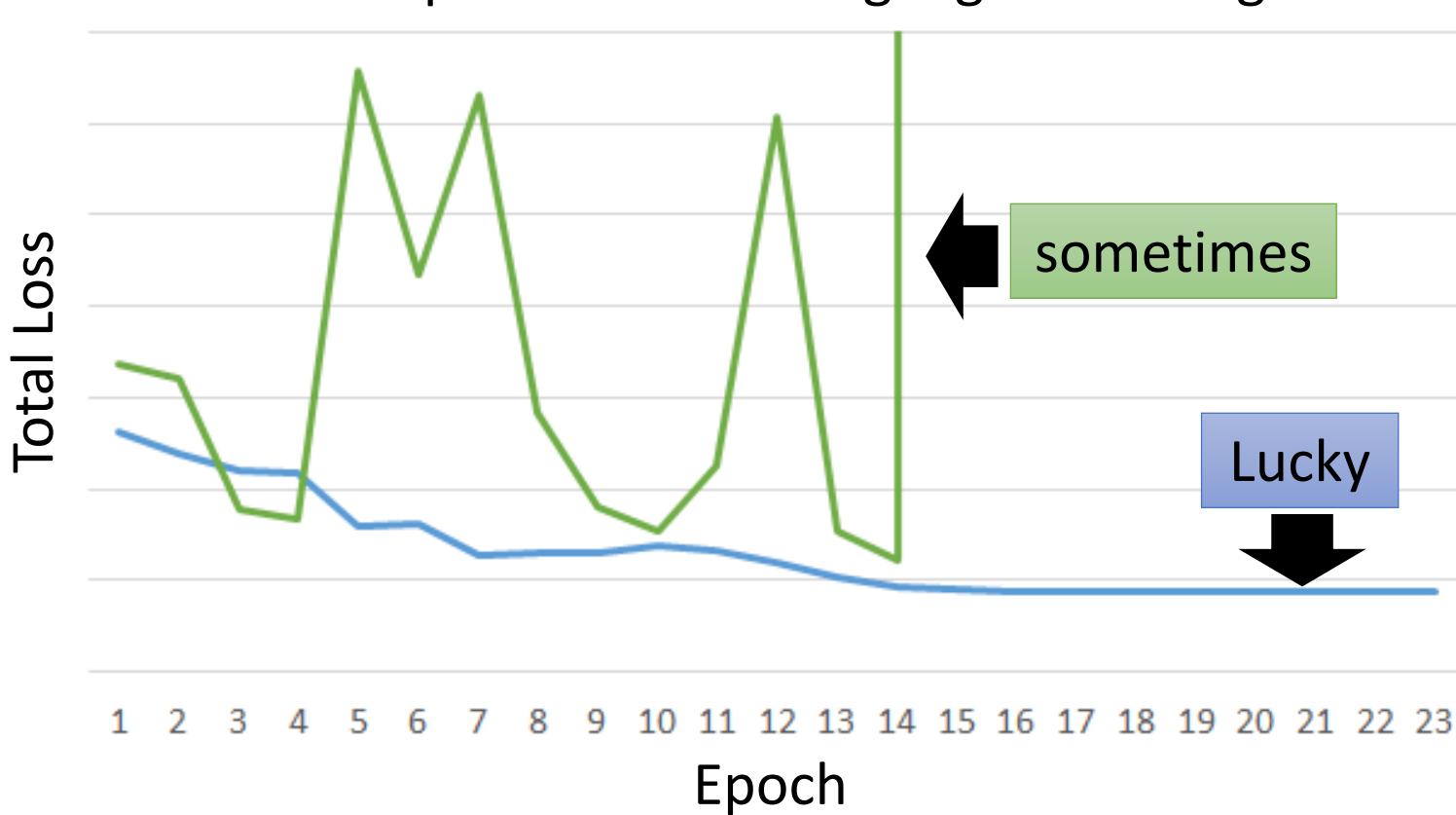


RNN Learning is very difficult in practice.

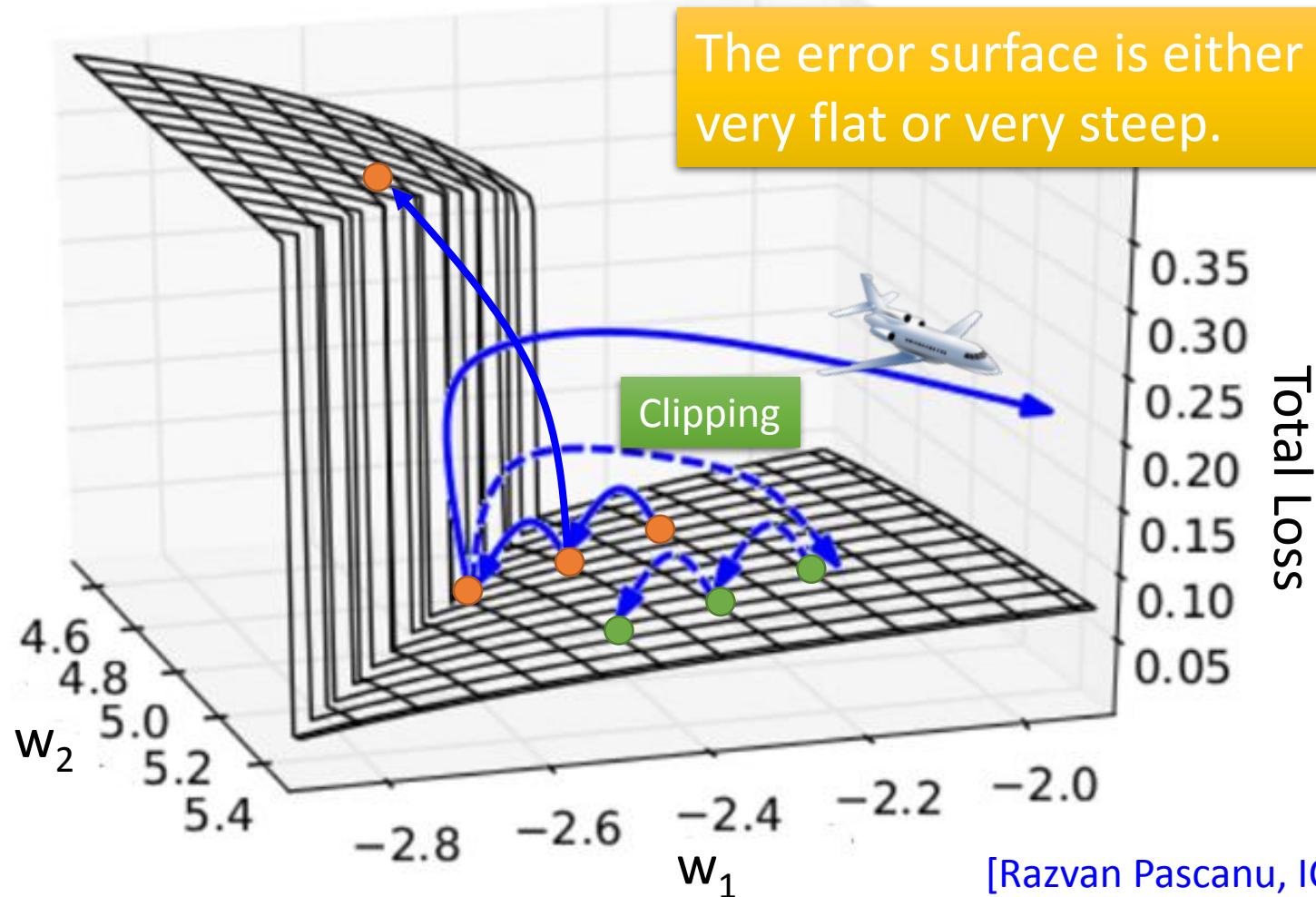
Unfortunately

- RNN-based network is not always easy to learn

Real experiments on Language modeling



The error surface is rough.



Why?

$$w = 1 \quad \rightarrow \quad y^{1000} = 1$$

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

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

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

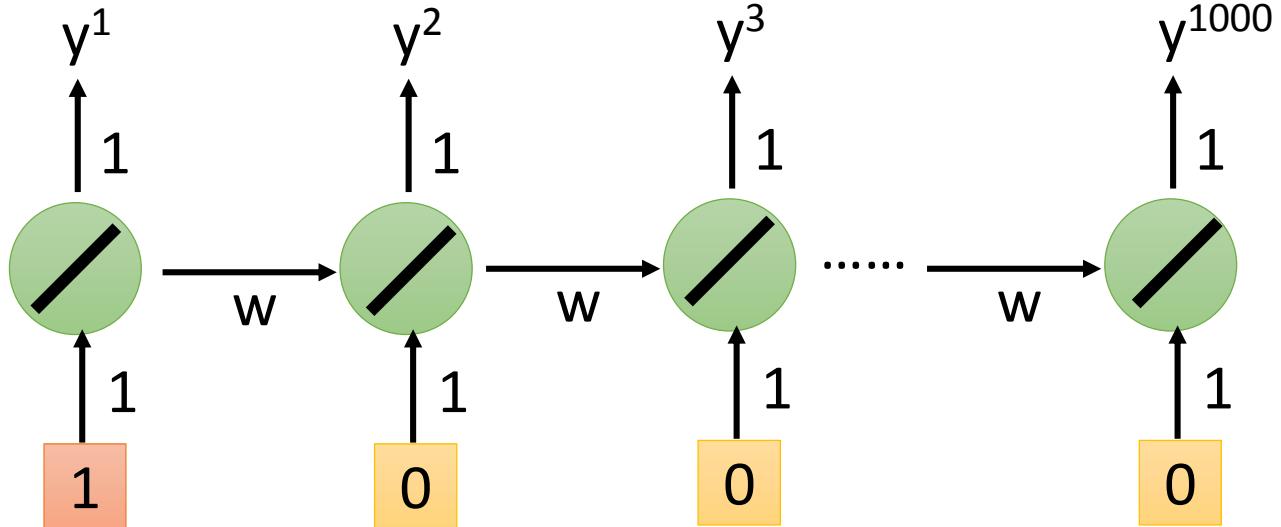
Large
 $\partial L / \partial w$

Small
Learning rate?

small
 $\partial L / \partial w$

Large
Learning rate?

Toy Example



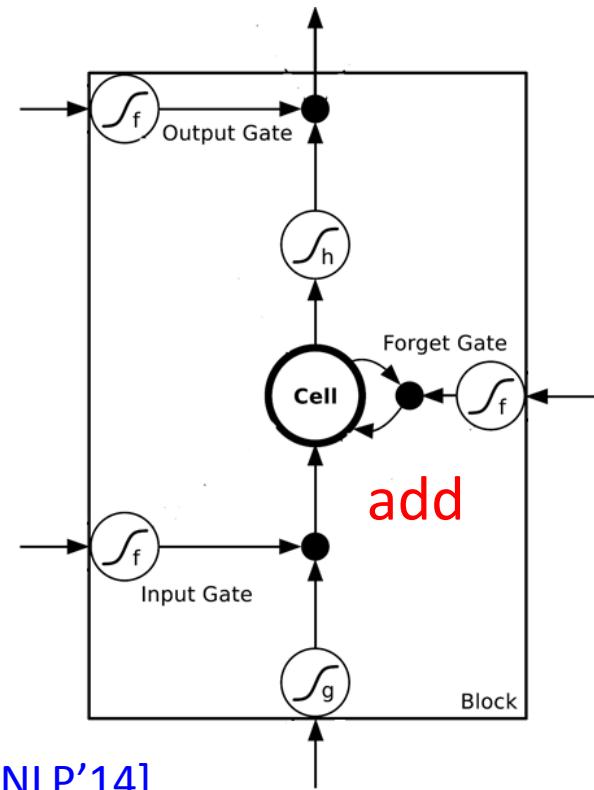
Helpful Techniques

• 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.)

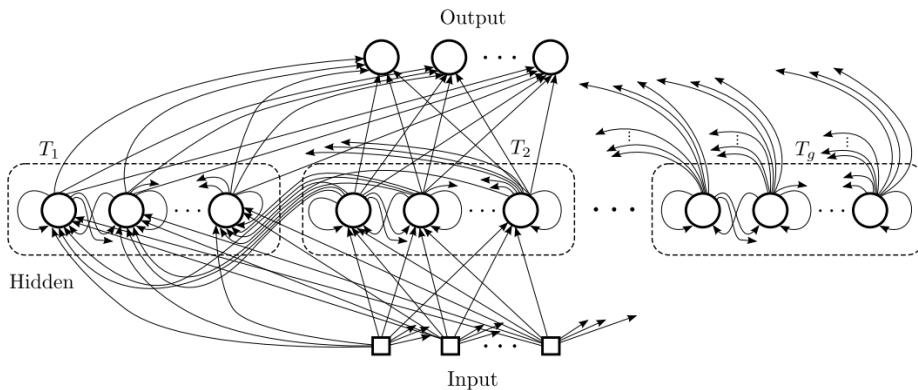
Gated Recurrent Unit (GRU):
simpler than LSTM

[Cho, EMNLP'14]



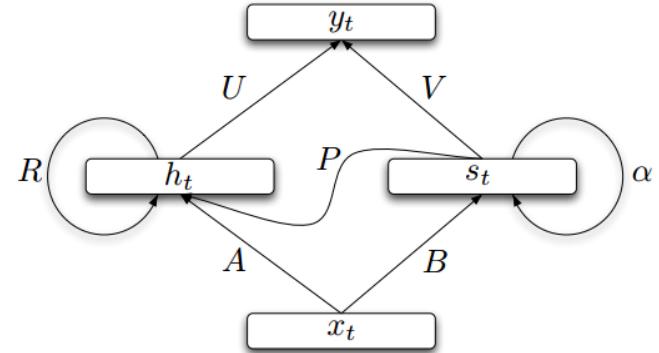
Helpful Techniques

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]

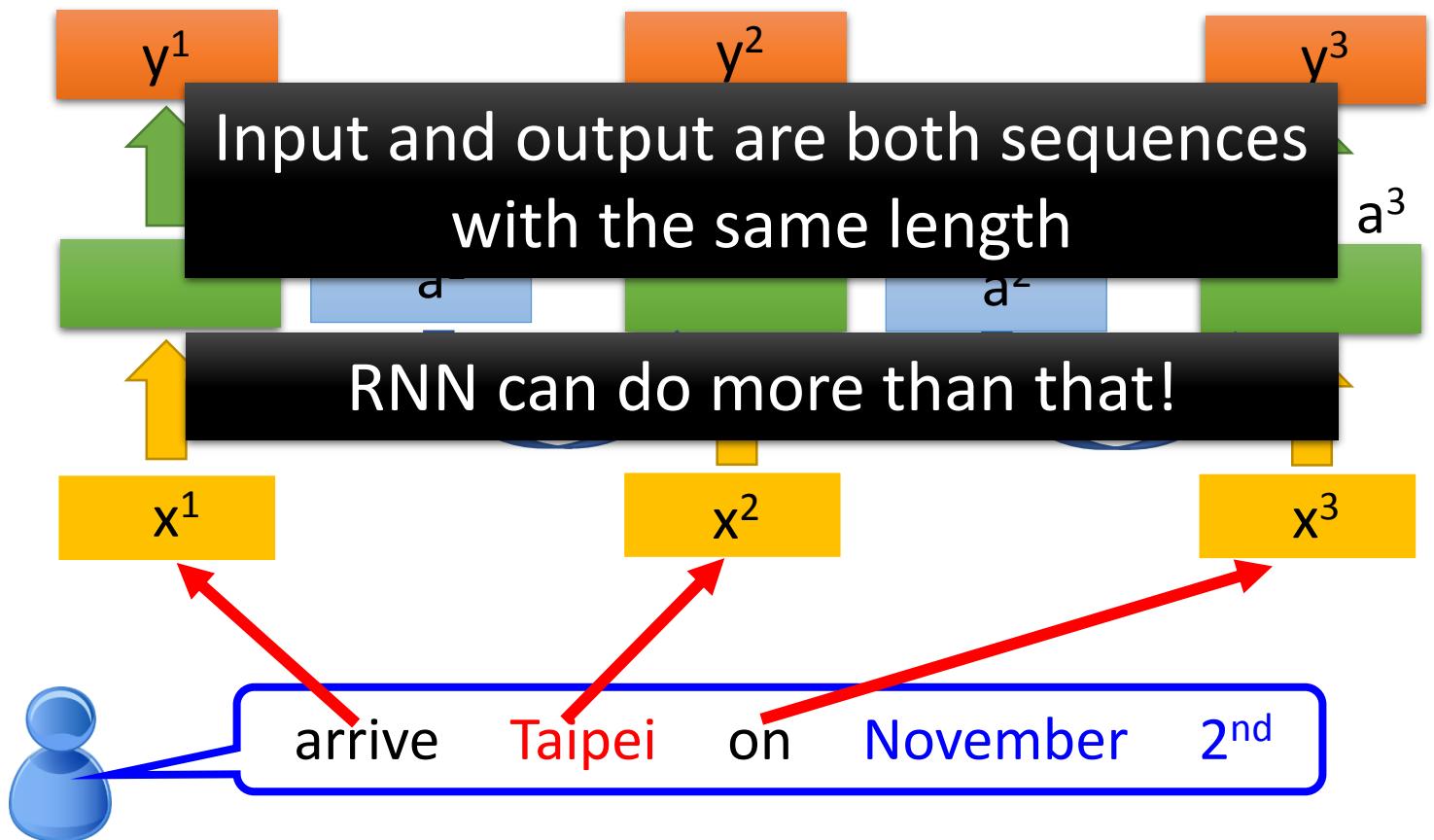
➤ Outperform or be comparable with LSTM in 4 different tasks

More Applications

Probability of
“arrive” in each slot

Probability of
“Taipei” in each slot

Probability of
“on” in each slot



Many to one

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

Sentiment Analysis

看了這部電影覺
得很高興

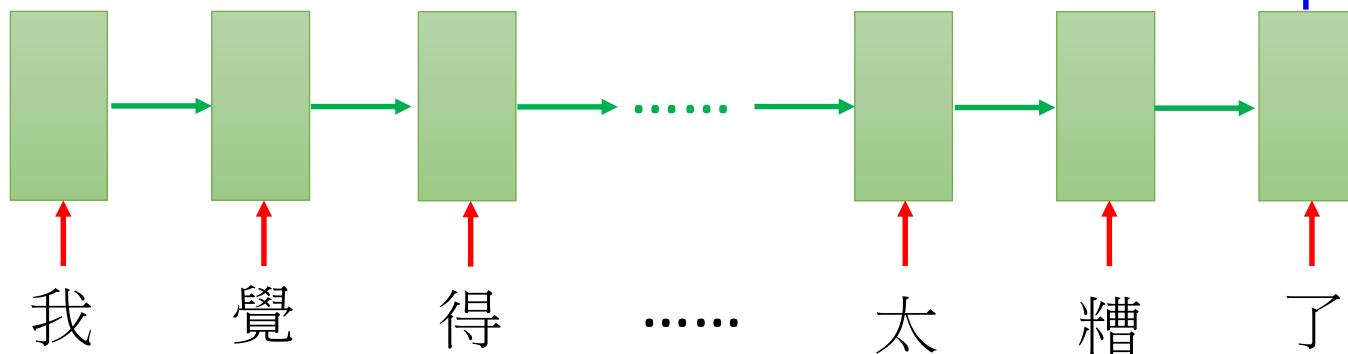
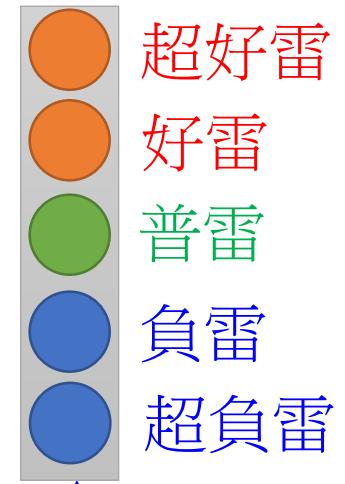
Positive (正雷)

這部電影太糟了
.....

Negative (負雷)

這部電影很
棒

Positive (正雷)

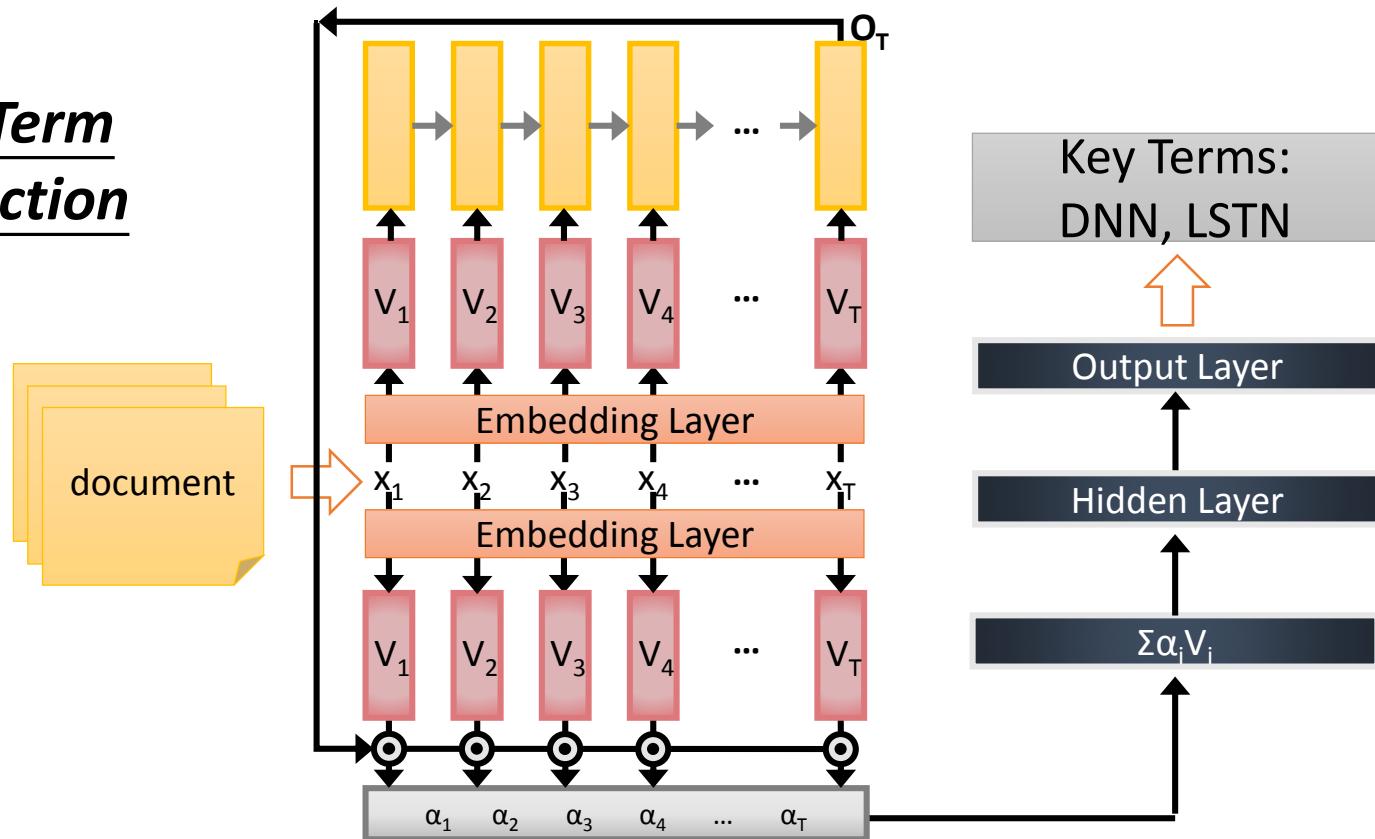


Many to one

[Shen & Lee, Interspeech 16]

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

Key Term Extraction



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”

Output: “好棒” (character sequence)



好 好 好 棒 棒 棒 棒

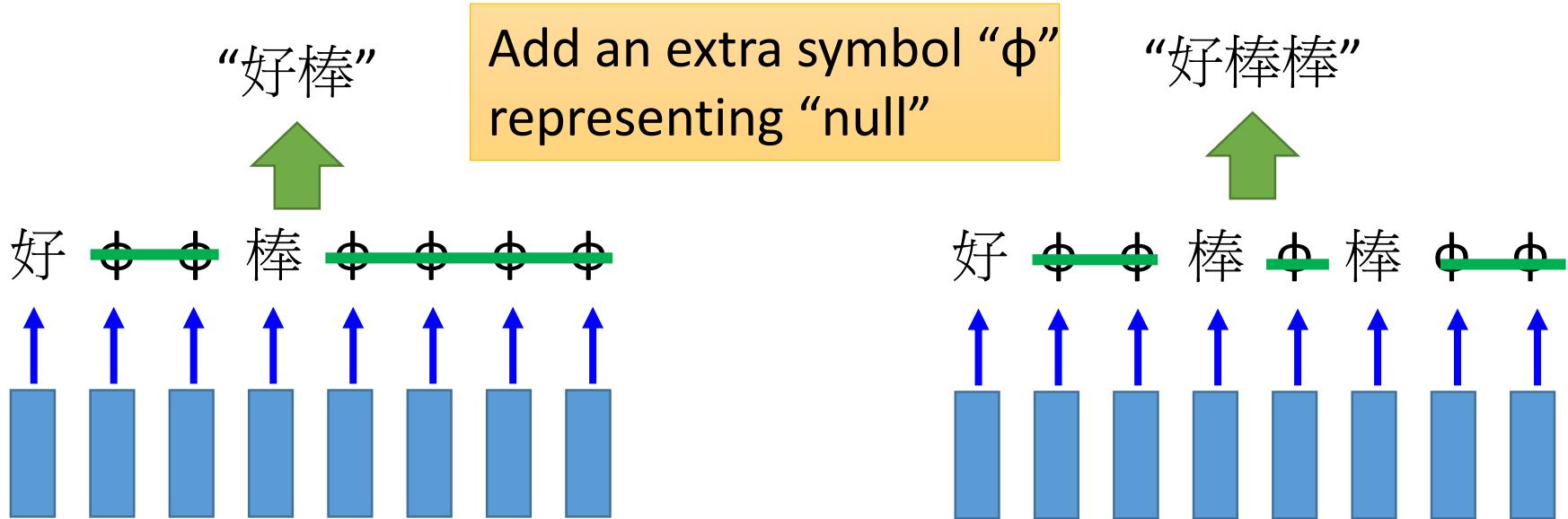
Input:

(vector sequence)



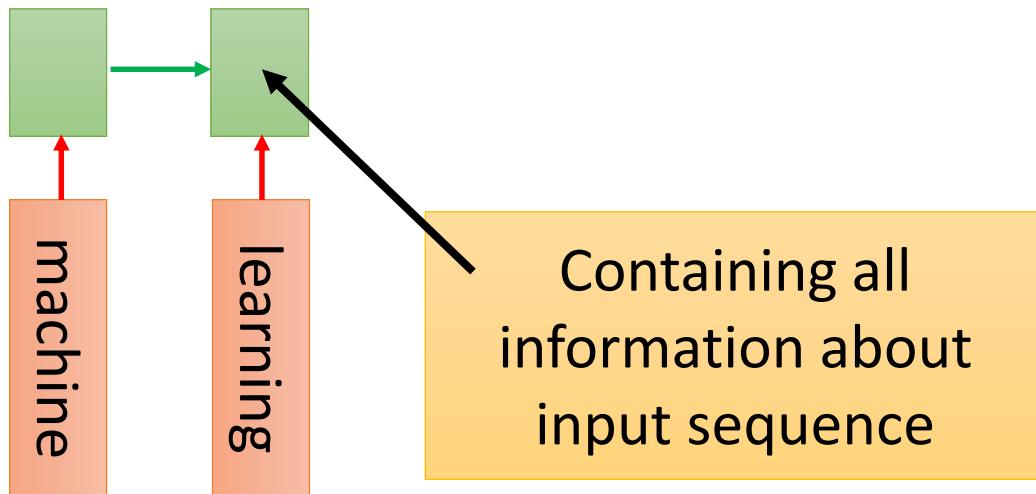
Many to Many (Output is shorter)

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



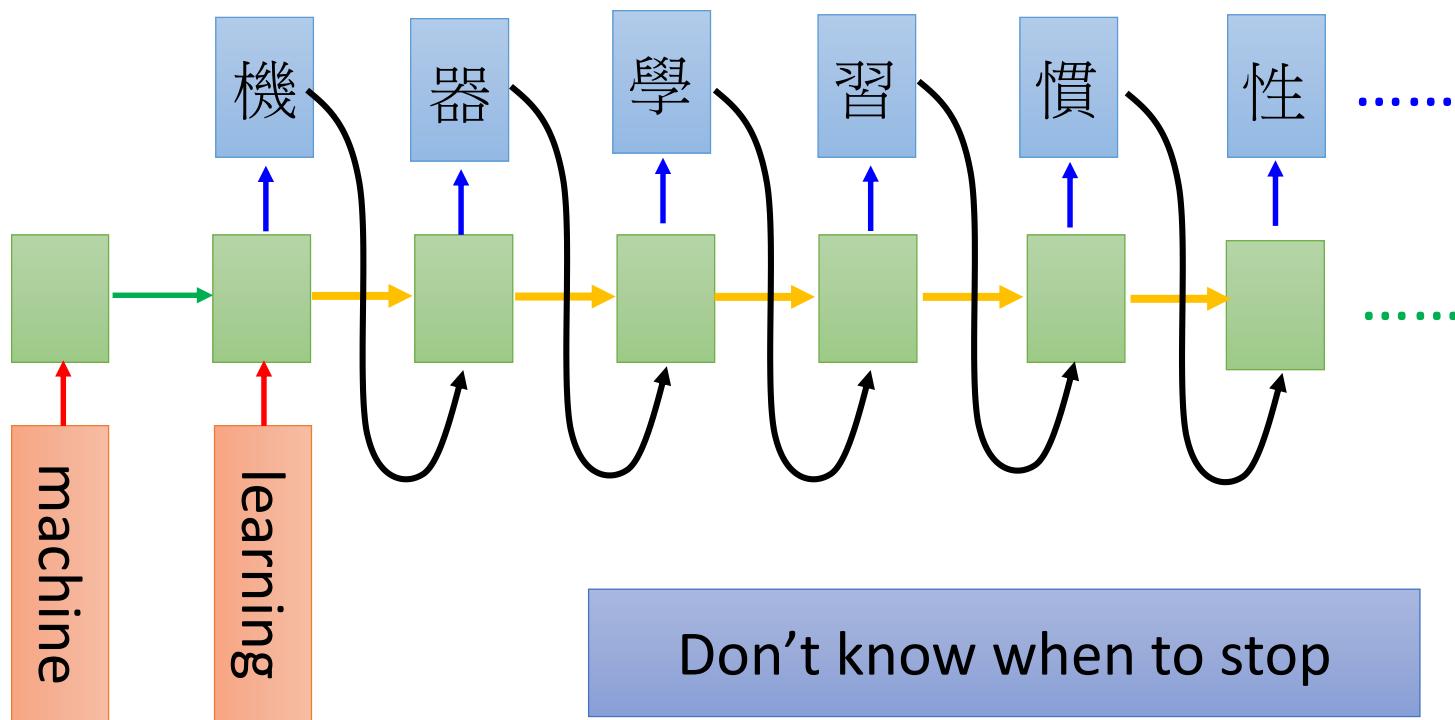
Many to Many (No Limitation)

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



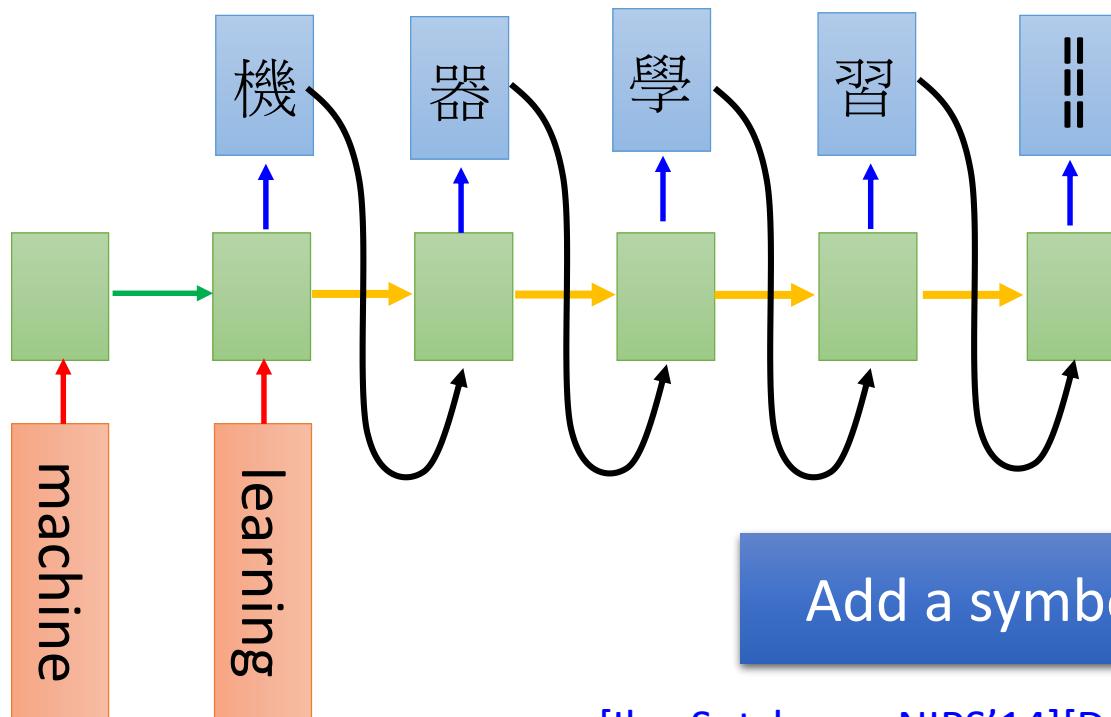
Many to Many (No Limitation)

推	1:	超	06/12 10:39
推	n:	人	06/12 10:40
推	tion:	正	06/12 10:41
→	host:	大	06/12 10:47
推	:	中	06/12 10:59
推	403:	天	06/12 11:11
推	:	外	06/12 11:13
推	527:	飛	06/12 11:17
→	990b:	仙	06/12 11:32
→	512:	草	06/12 12:15

推 tlkagk: =====斷=====

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)



[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]

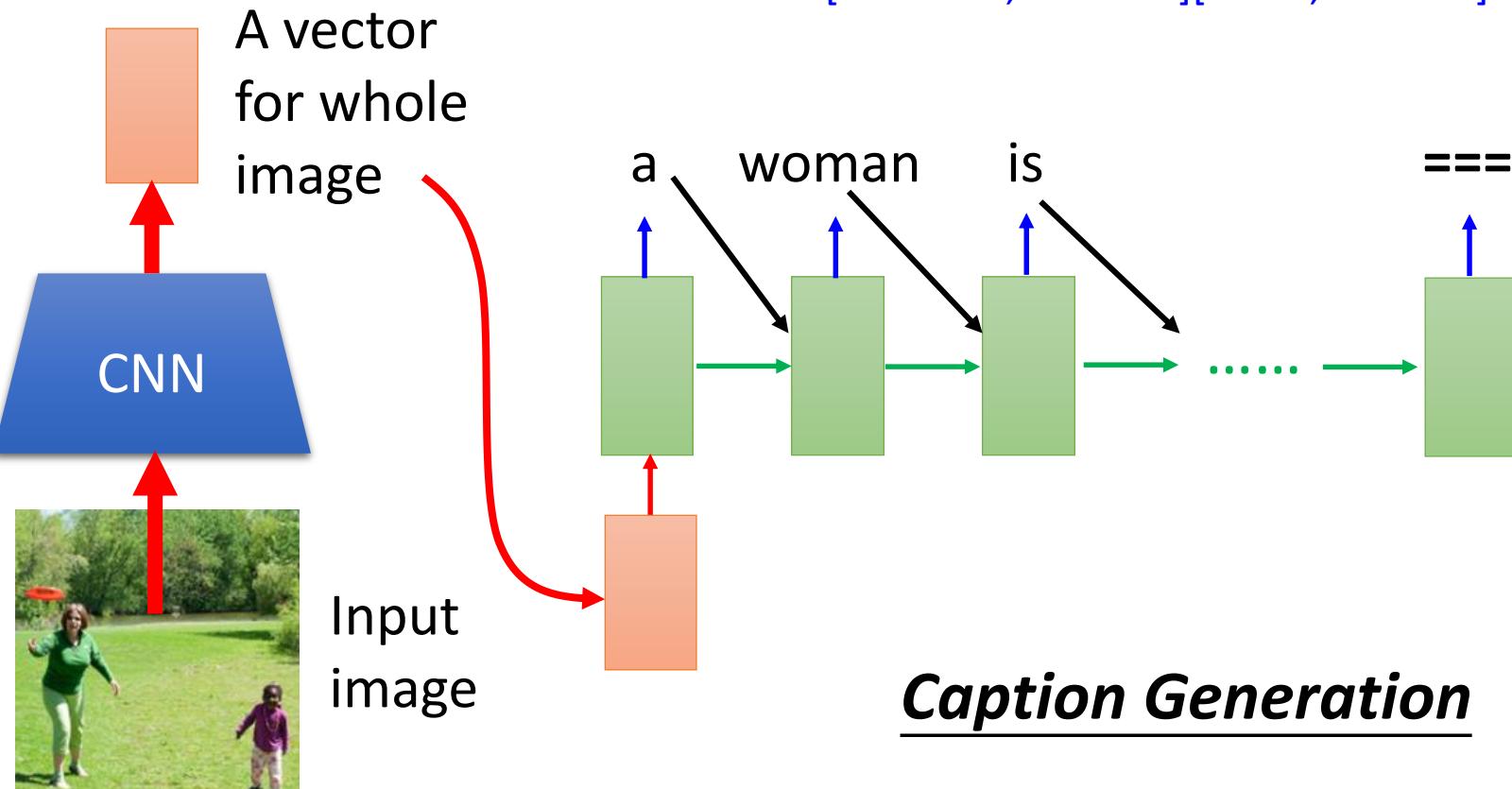


Image Caption Generation

- 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

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



Video Caption Generation



Video



A group of people is knocked by a tree.

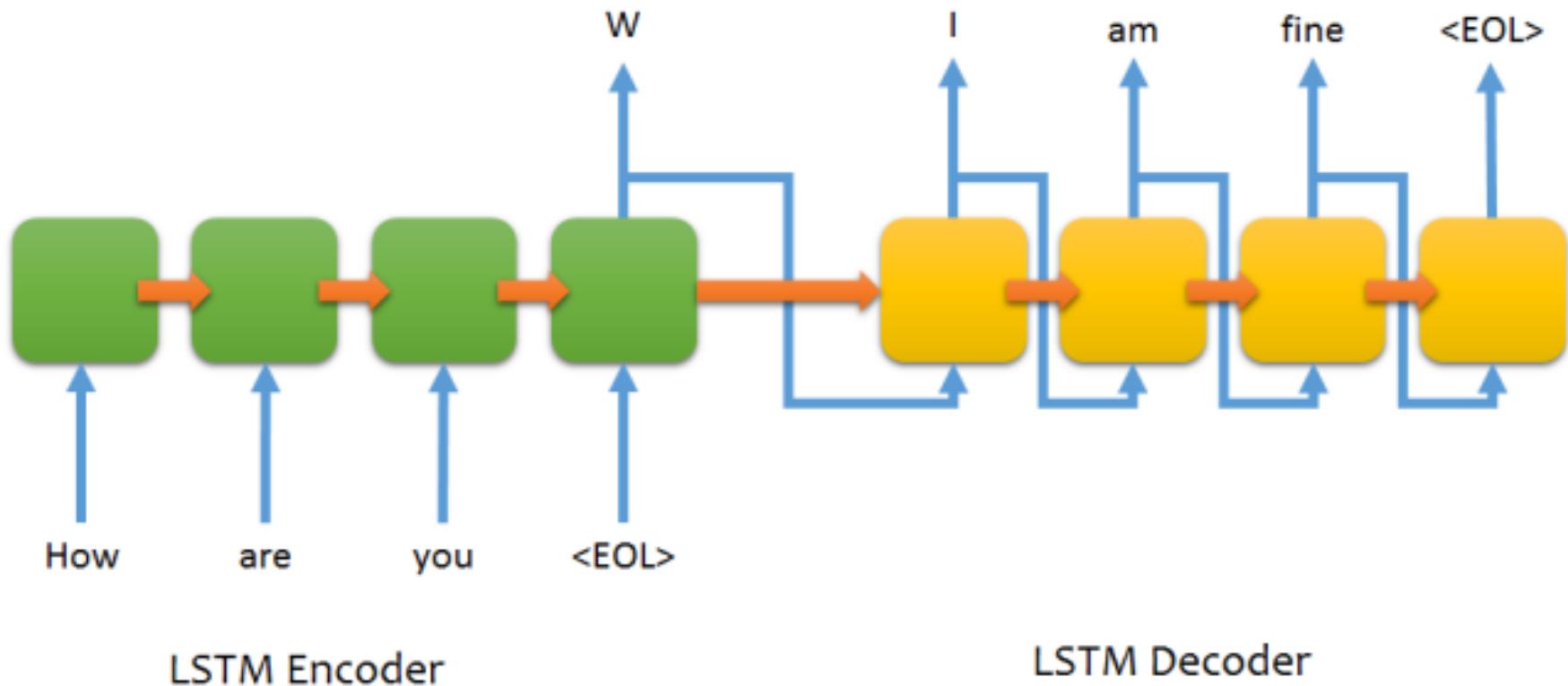


A group of people is walking in the forest.

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

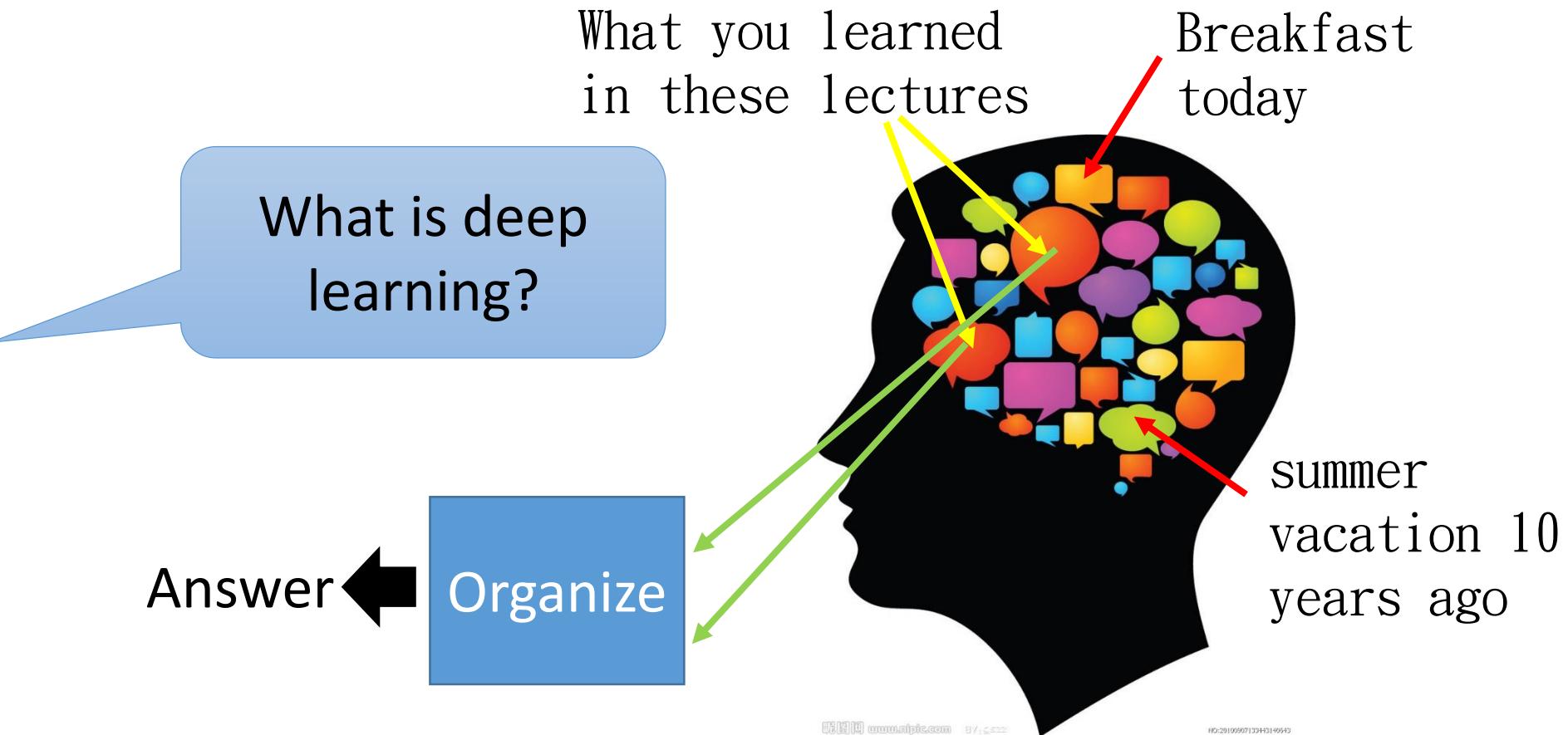


電視影集 (~40,000 sentences)、美國總統大選辯論

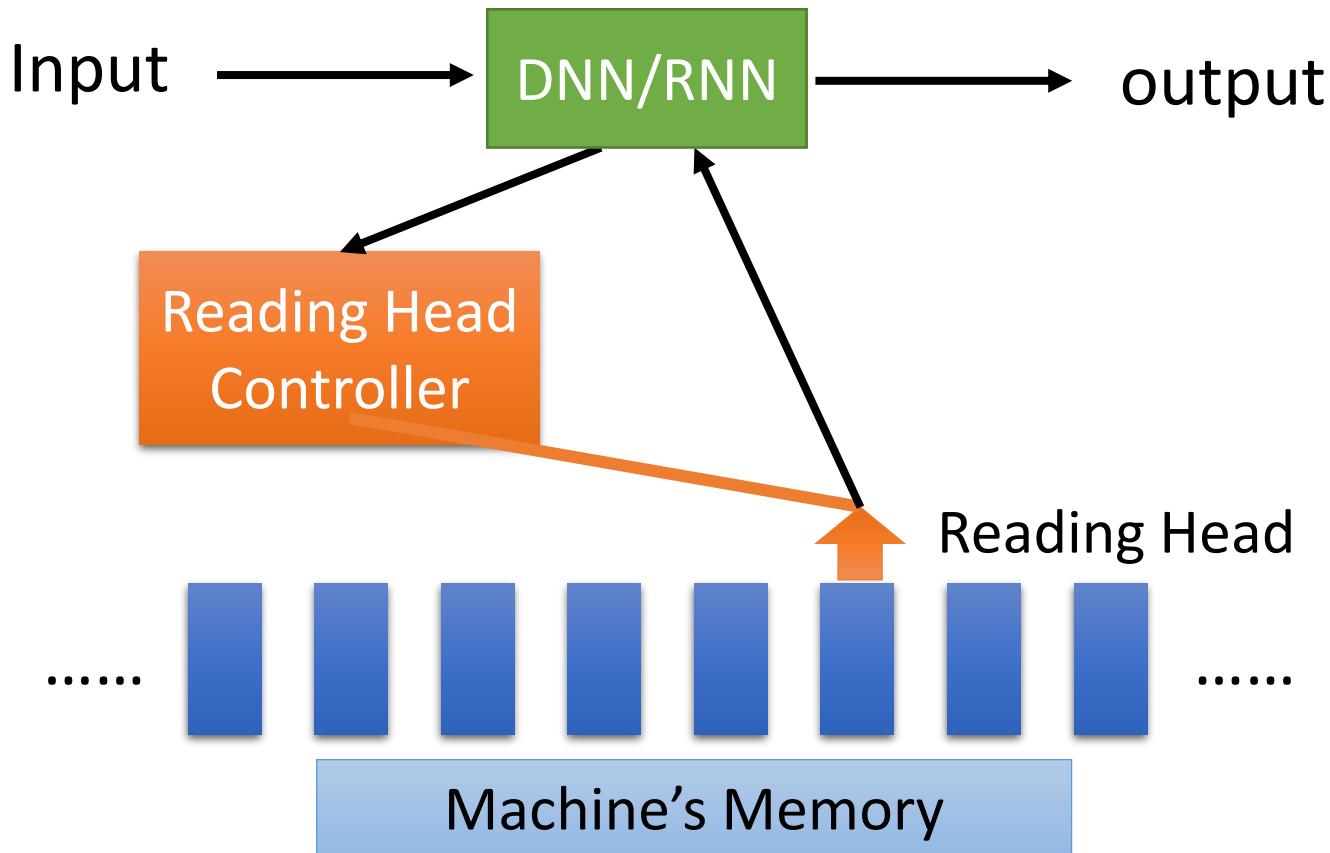
Demo

- 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

Attention-based Model



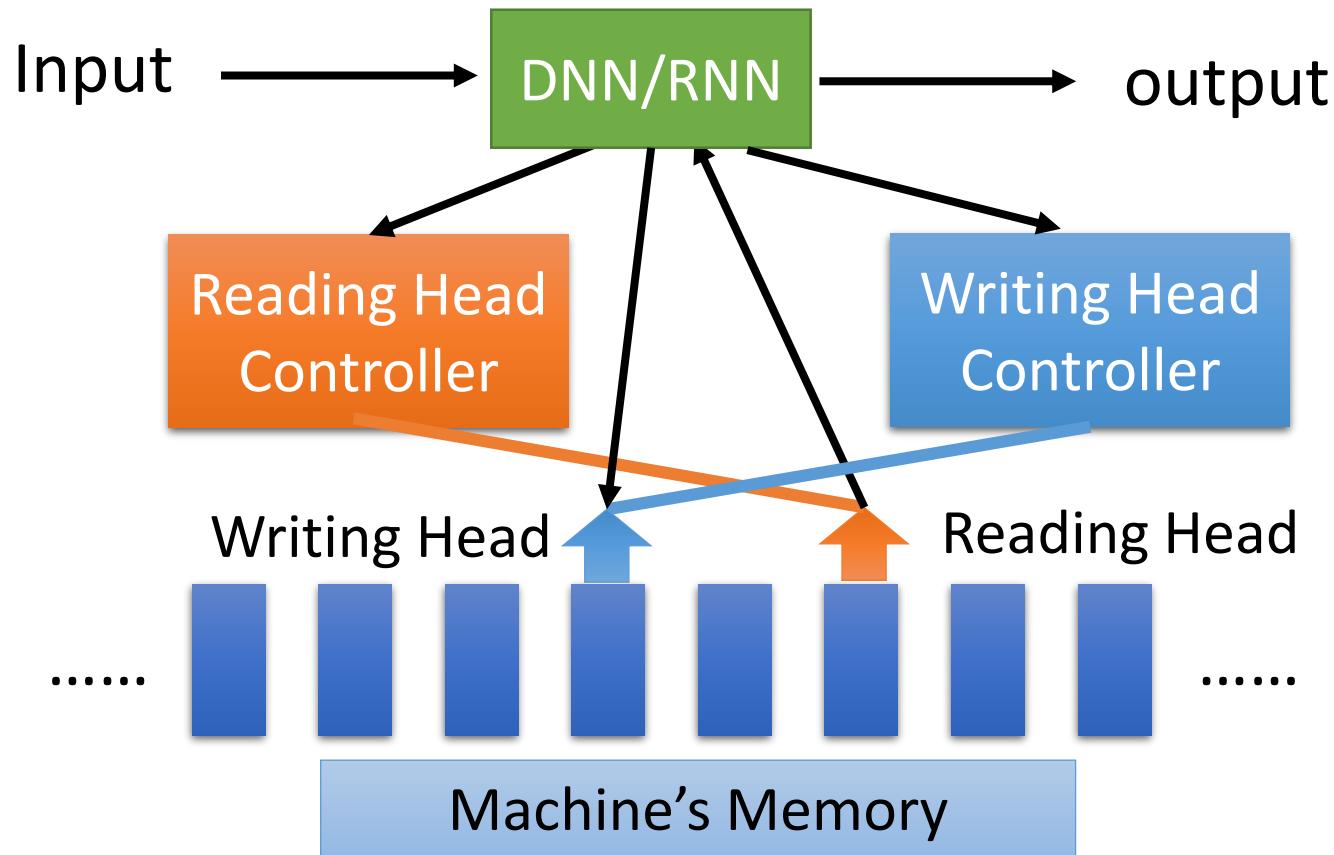
Attention-based Model



Ref:

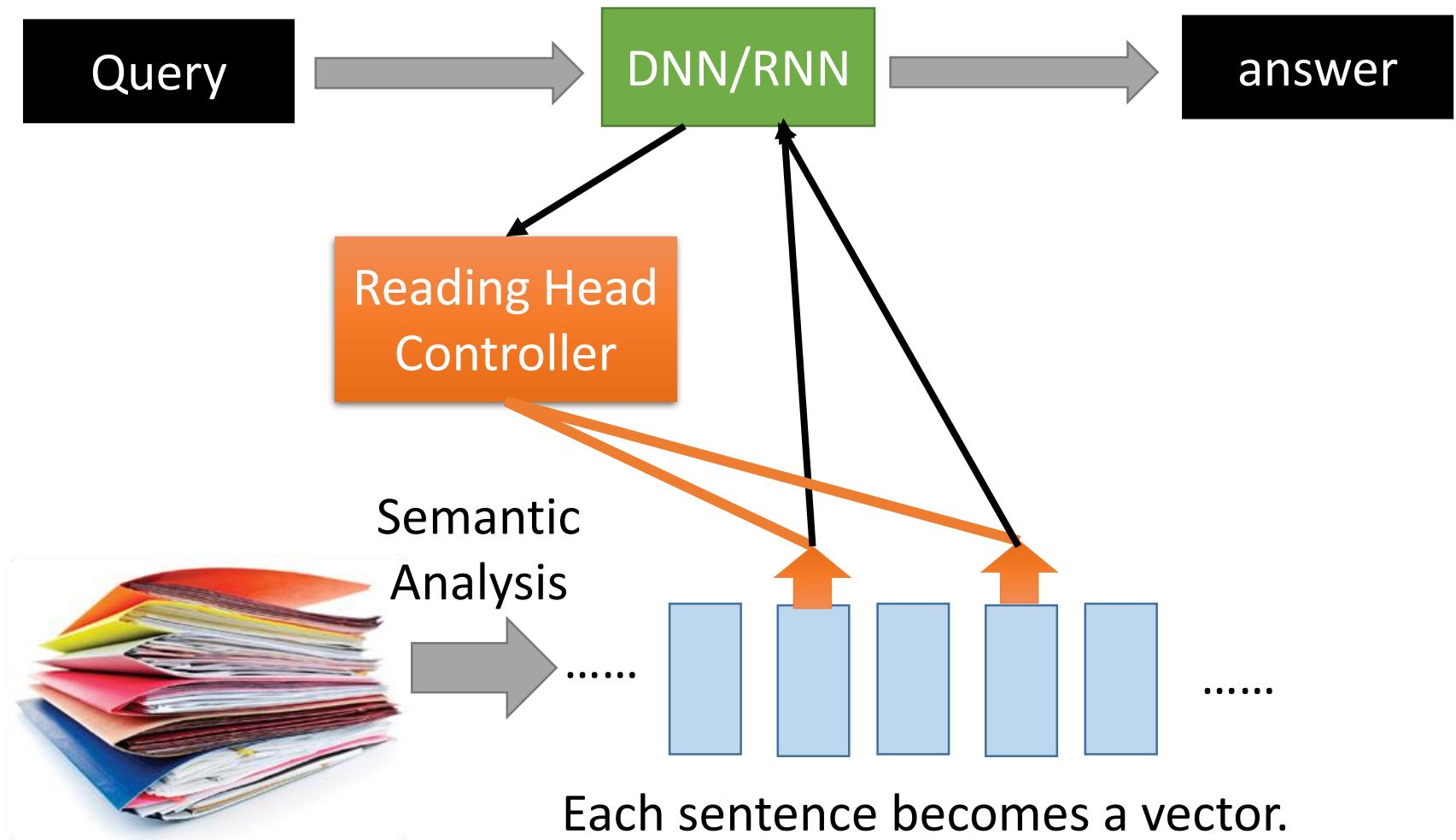
[http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2/Lecture/Attain%20\(v3\).ecm.mp4/index.html](http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2/Lecture/Attain%20(v3).ecm.mp4/index.html)

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.

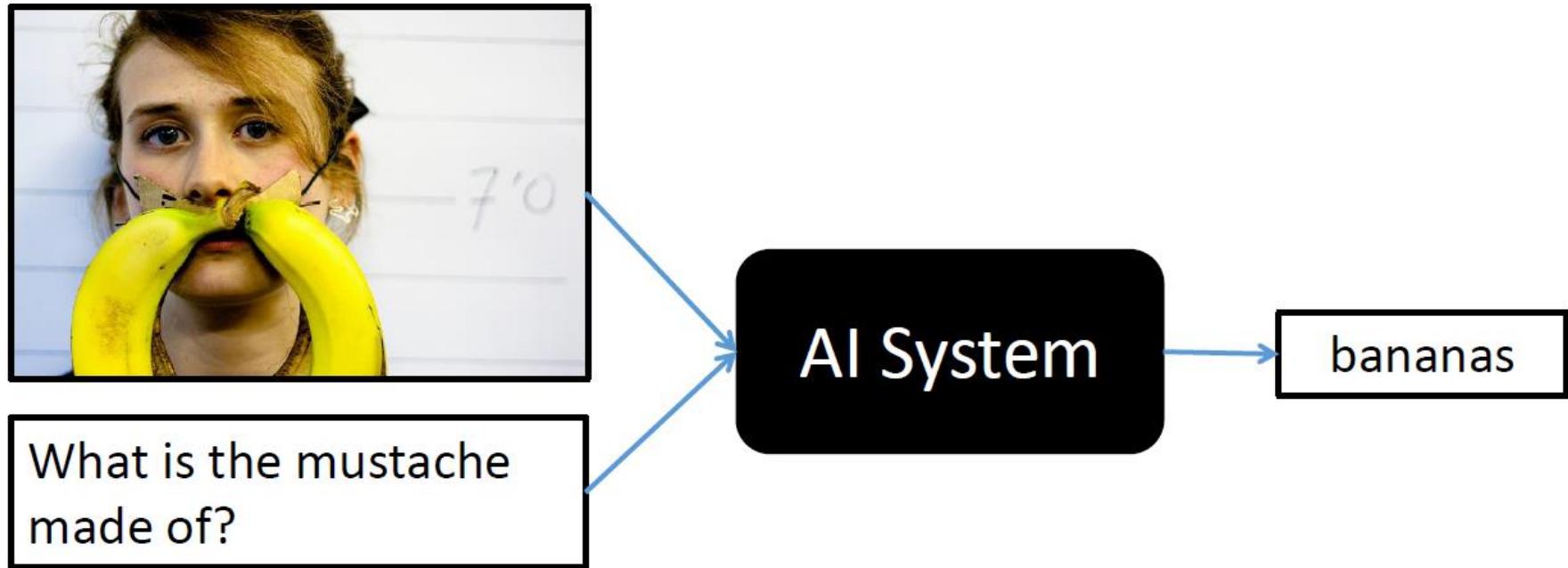
The position of reading head:

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:

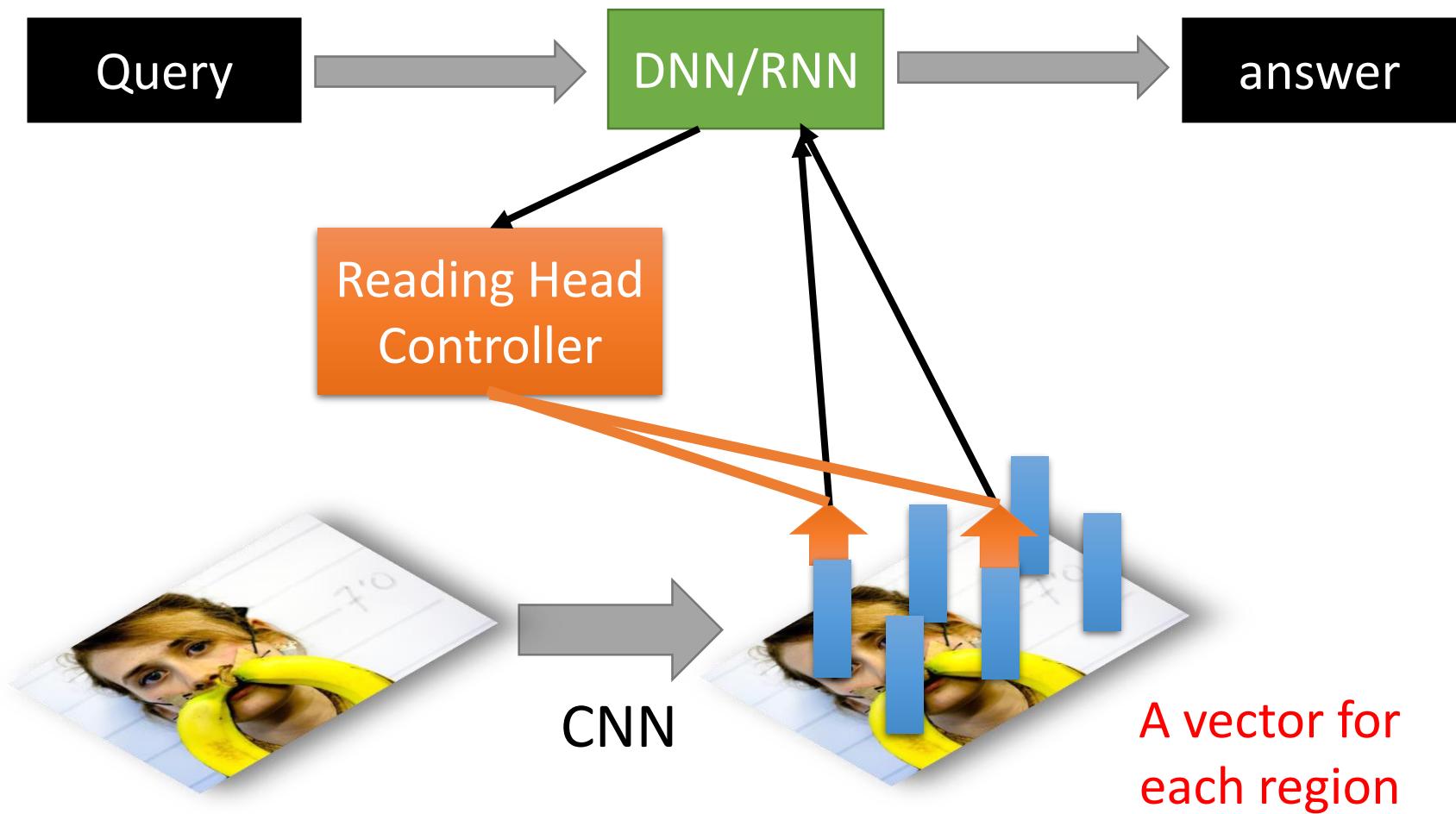
https://github.com/fchollet/keras/blob/master/examples/babi_memnn.py

Visual Question Answering



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.)

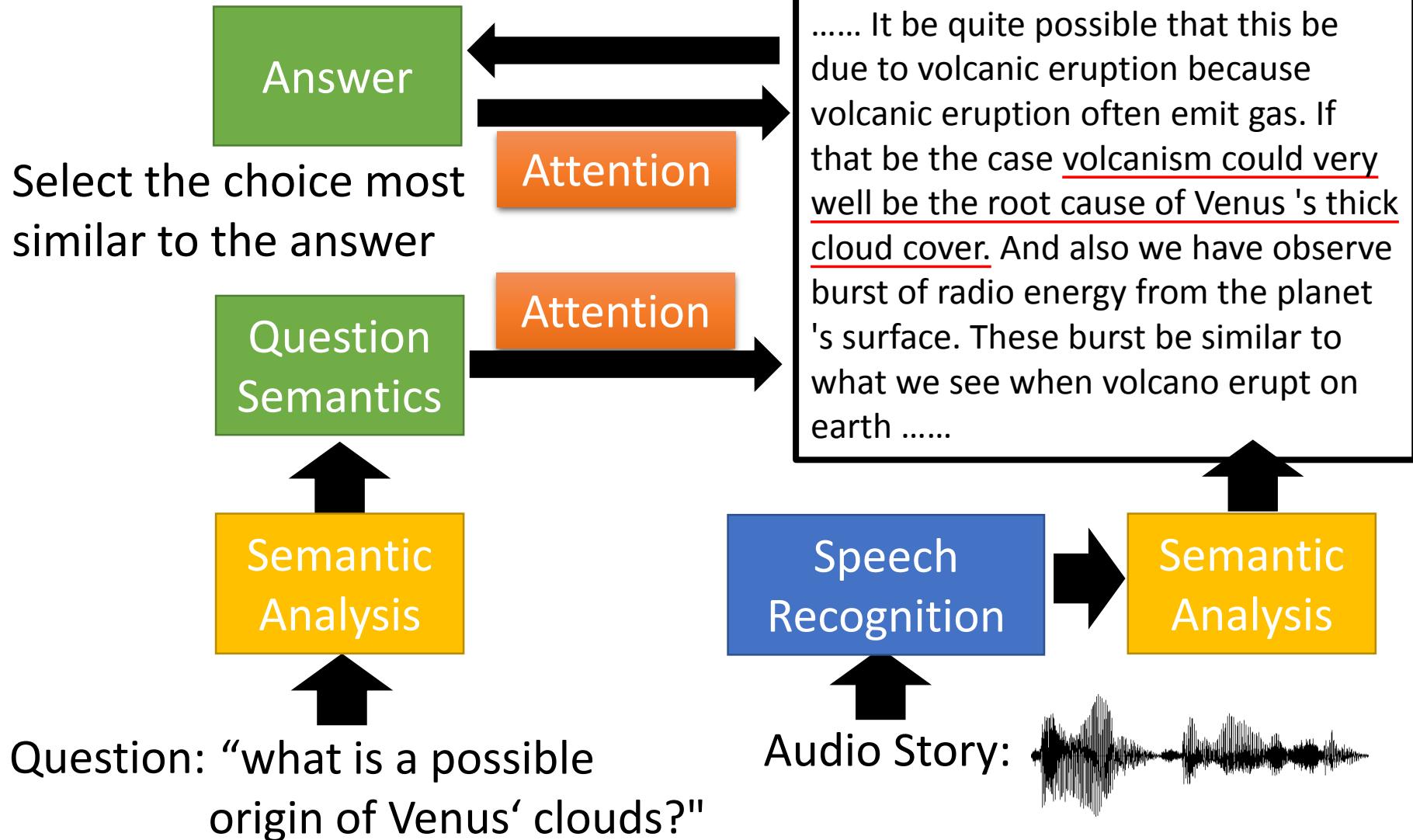
Question: “What is a possible origin of Venus’ clouds?”

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 planet's surface
- (D) strong winds that blow dust into the atmosphere

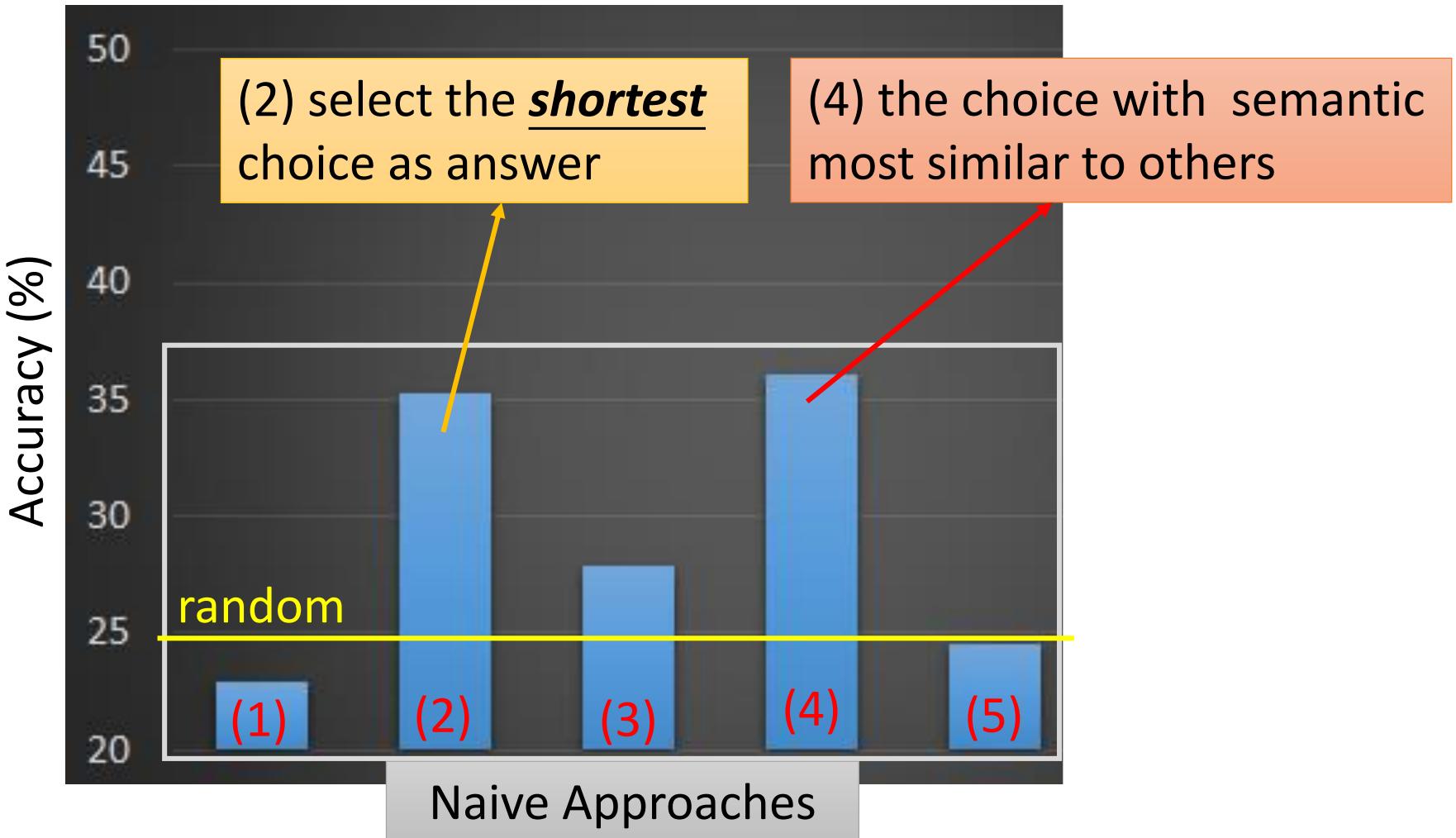
Model Architecture

Everything is learned from training examples

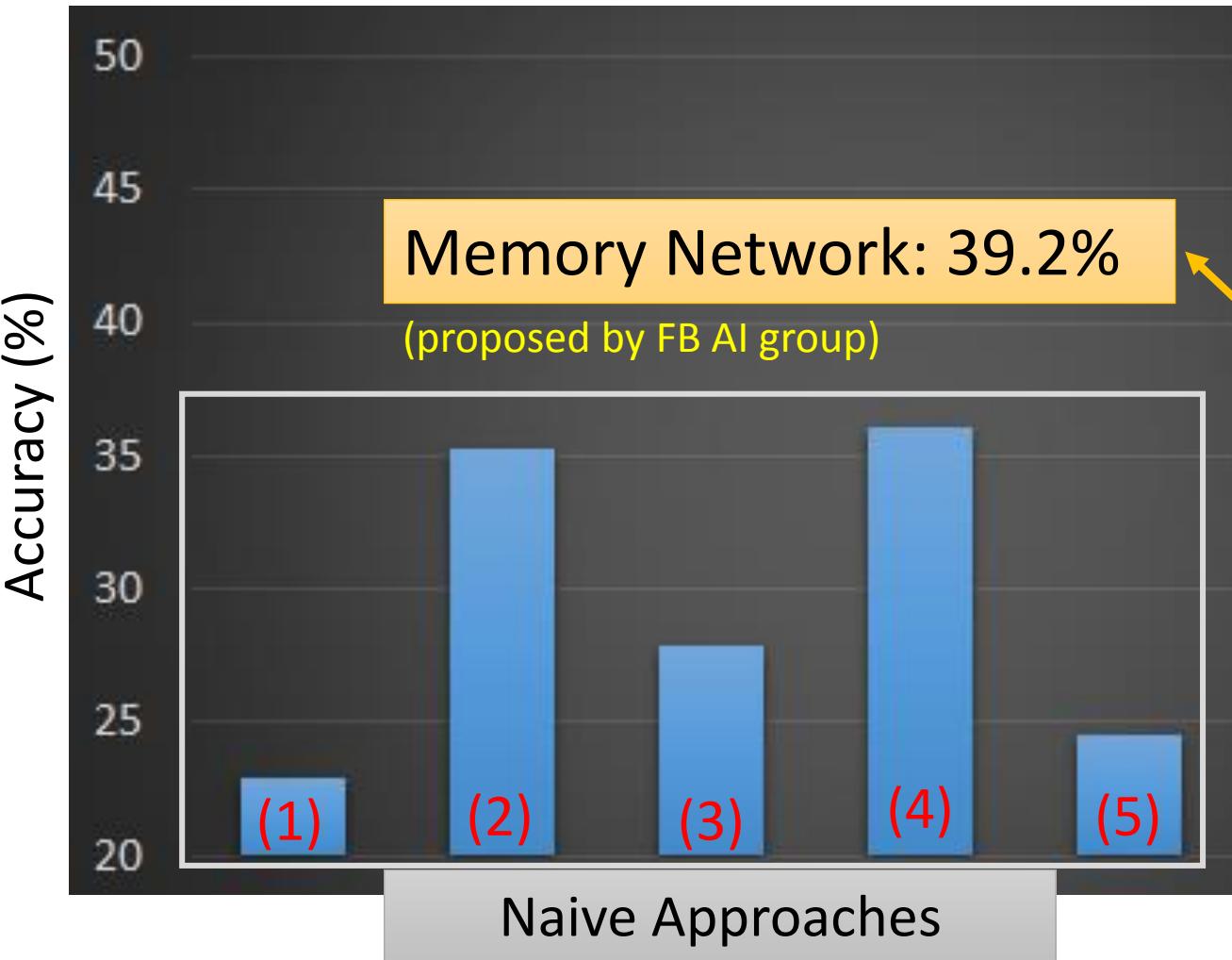


Simple Baselines

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

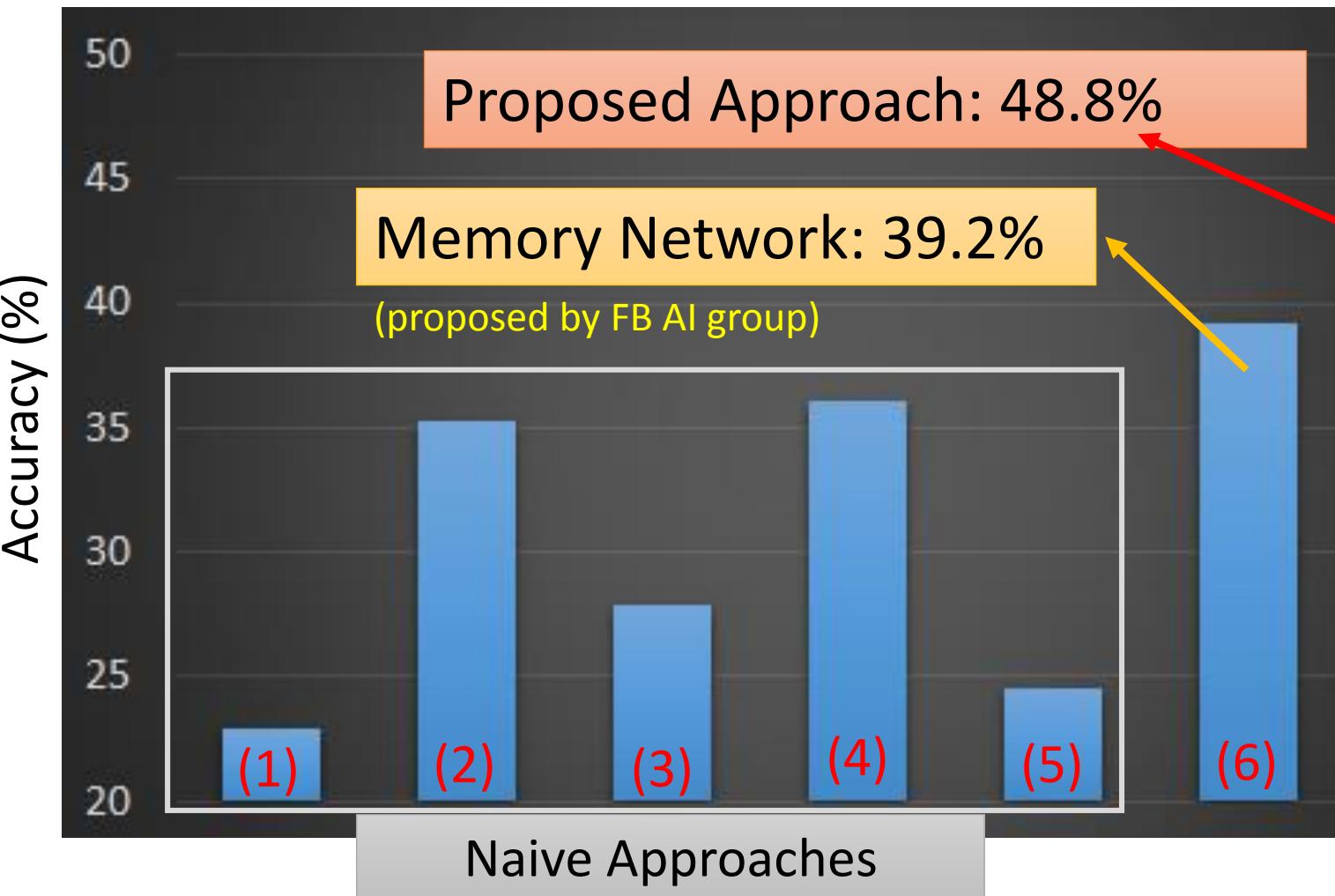


Memory Network



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

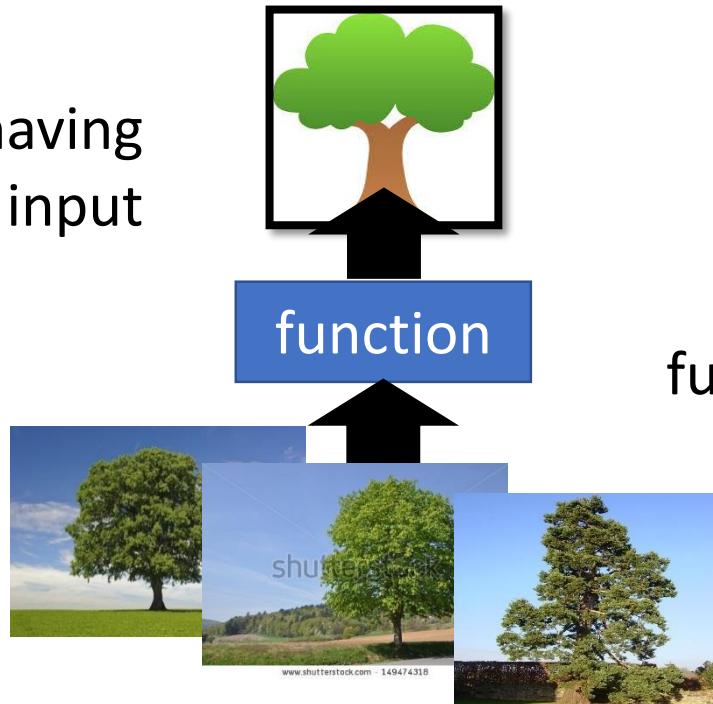
- 化繁為簡
 - Auto-encoder
 - Word Vector and Audio Word Vector
- 無中生有

Reinforcement Learning

Unsupervised Learning

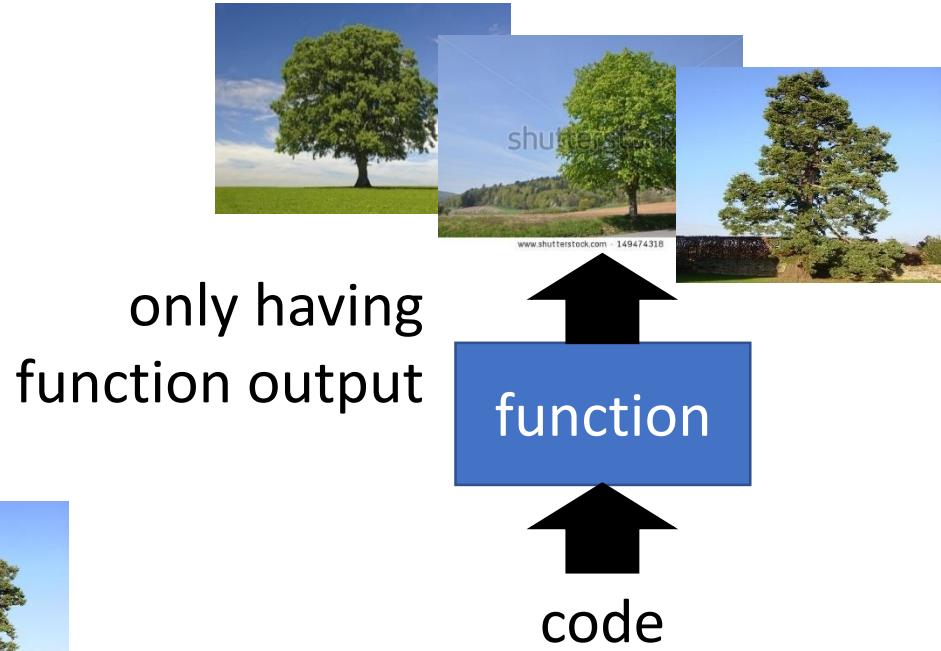
- 化繁為簡

only having
function input



- 無中生有

only having
function output



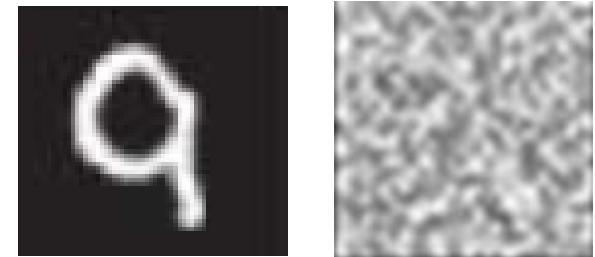
Outline

Unsupervised Learning

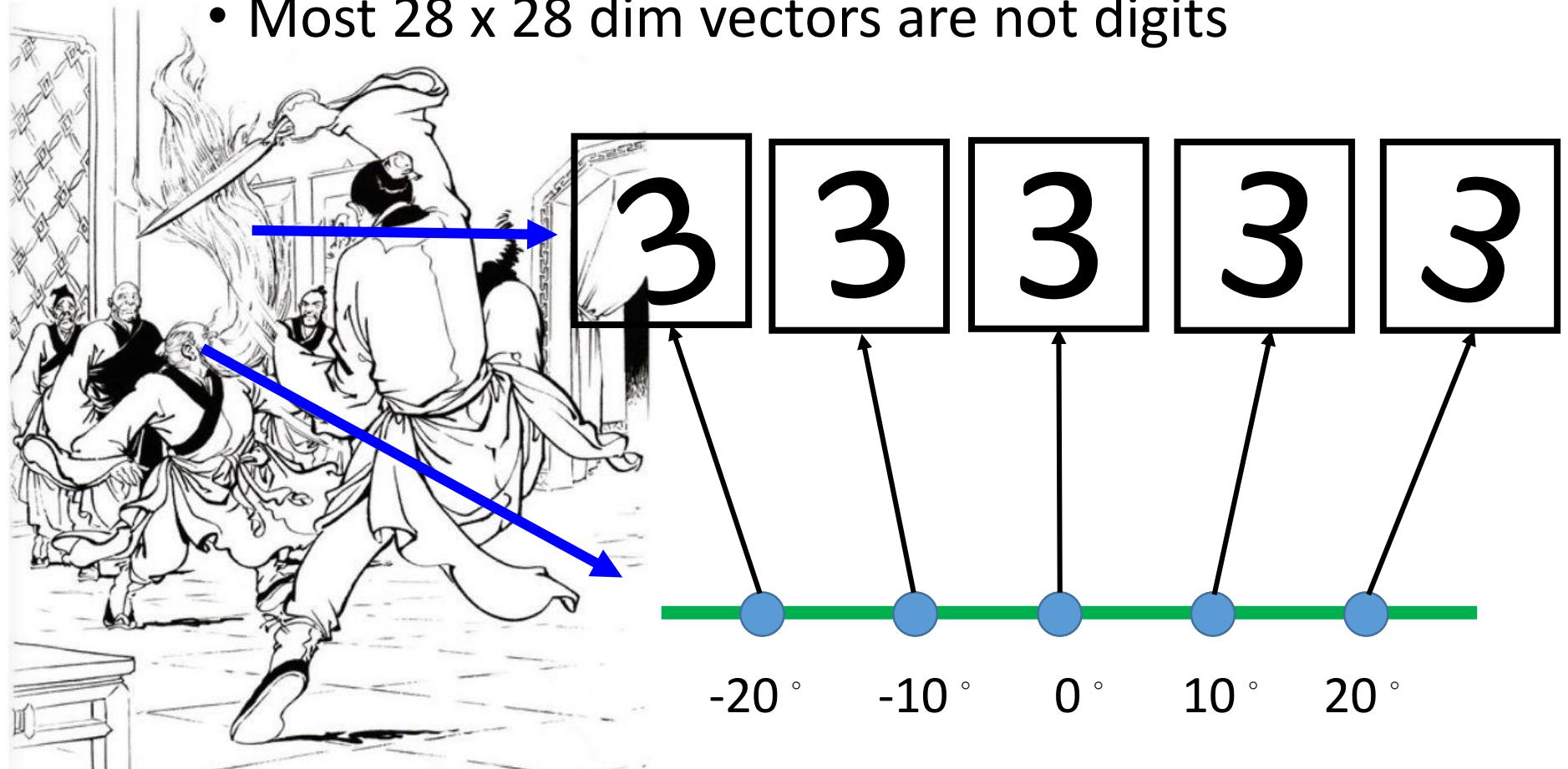
- **化繁為簡**
- Auto-encoder
- Word Vector and Audio Word Vector
- 無中生有

Reinforcement Learning

Motivation



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



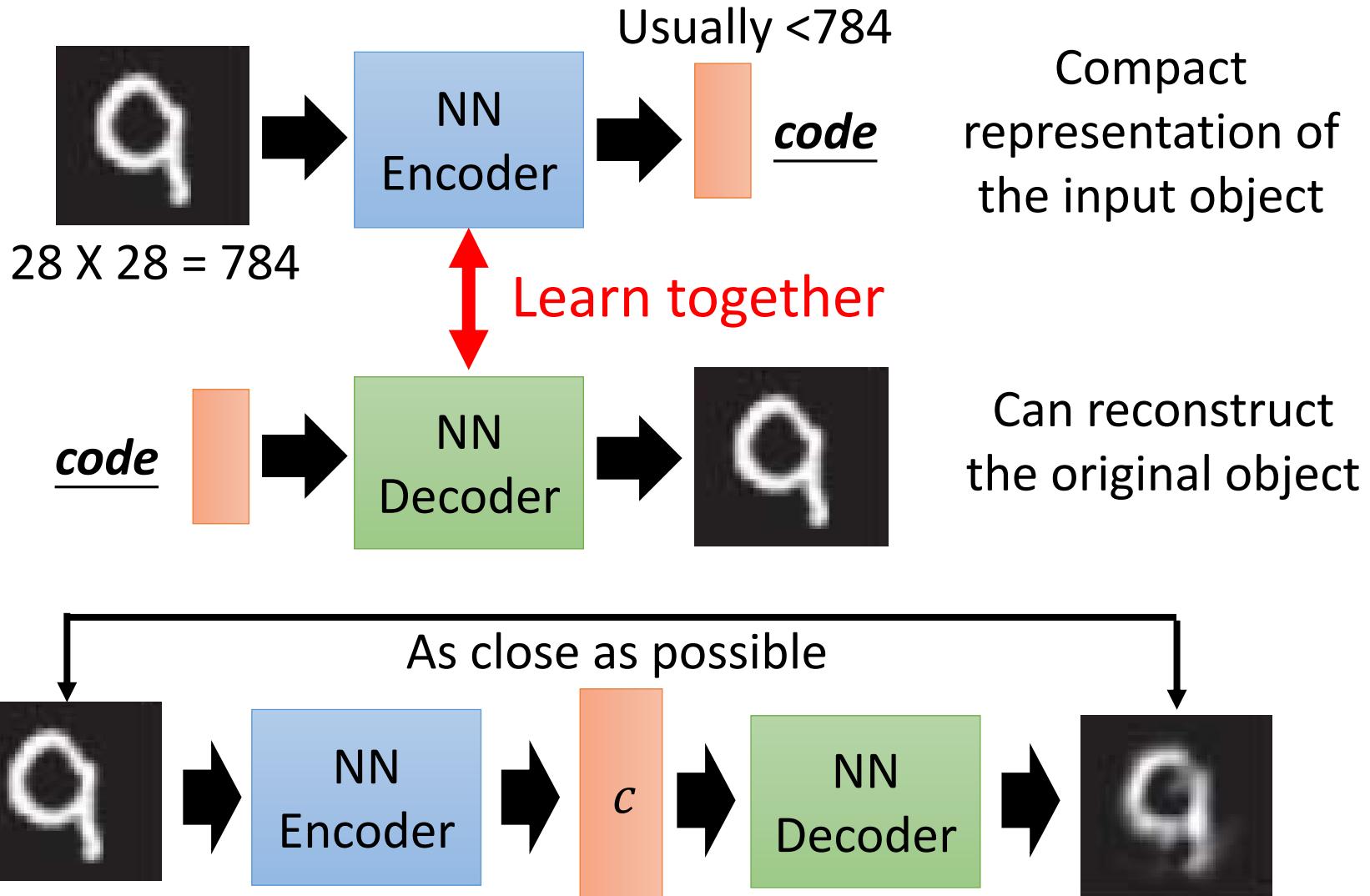
Outline

Unsupervised Learning

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

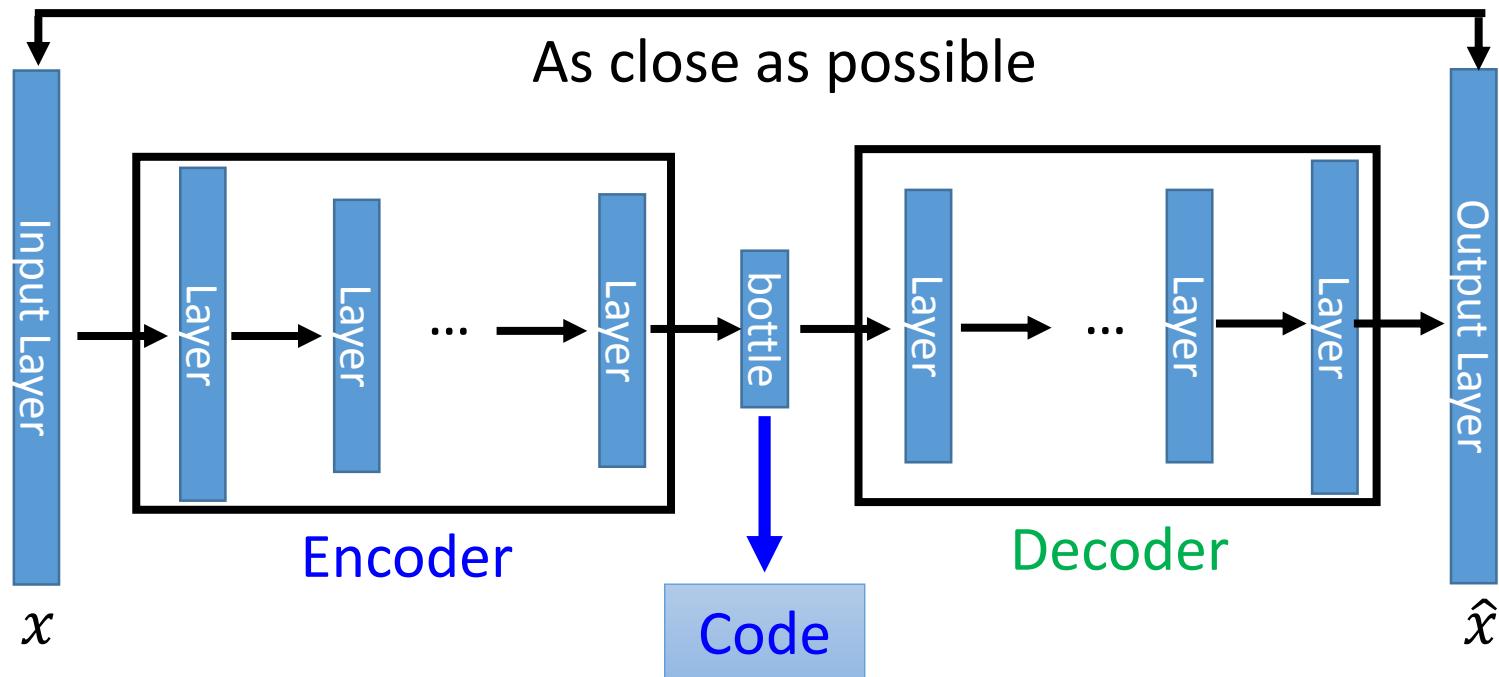
Reinforcement Learning

Auto-encoder



Deep Auto-encoder

- NN encoder + NN decoder = a deep network



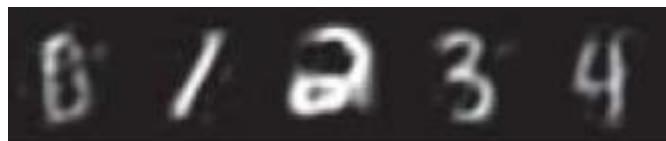
Reference: Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." *Science* 313.5786 (2006): 504-507

Deep Auto-encoder

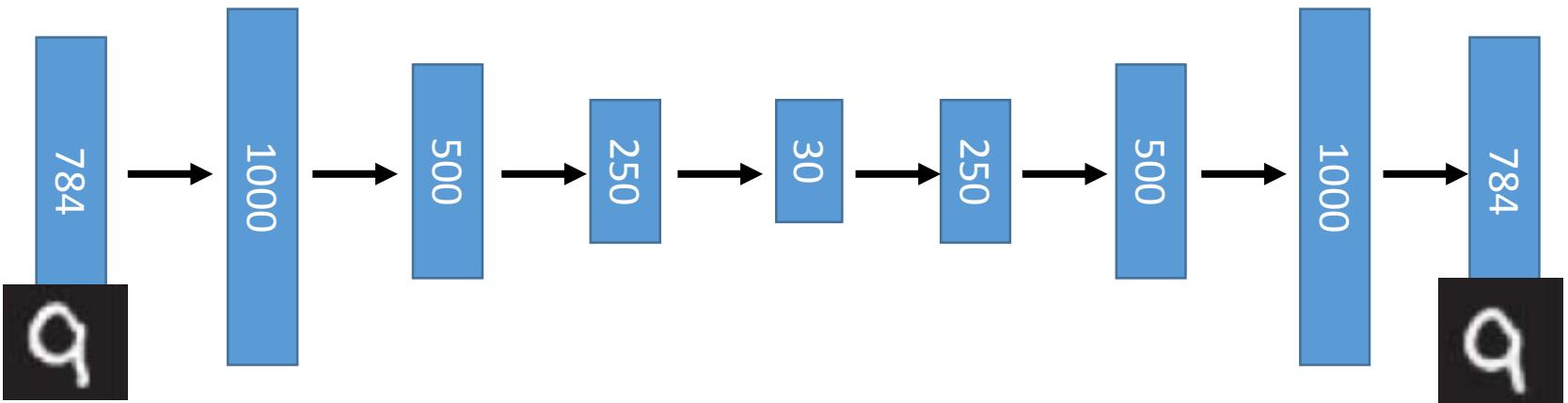
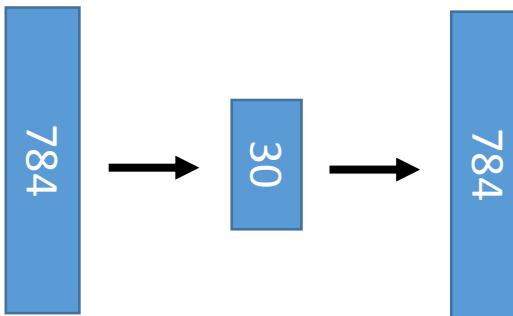
Original Image

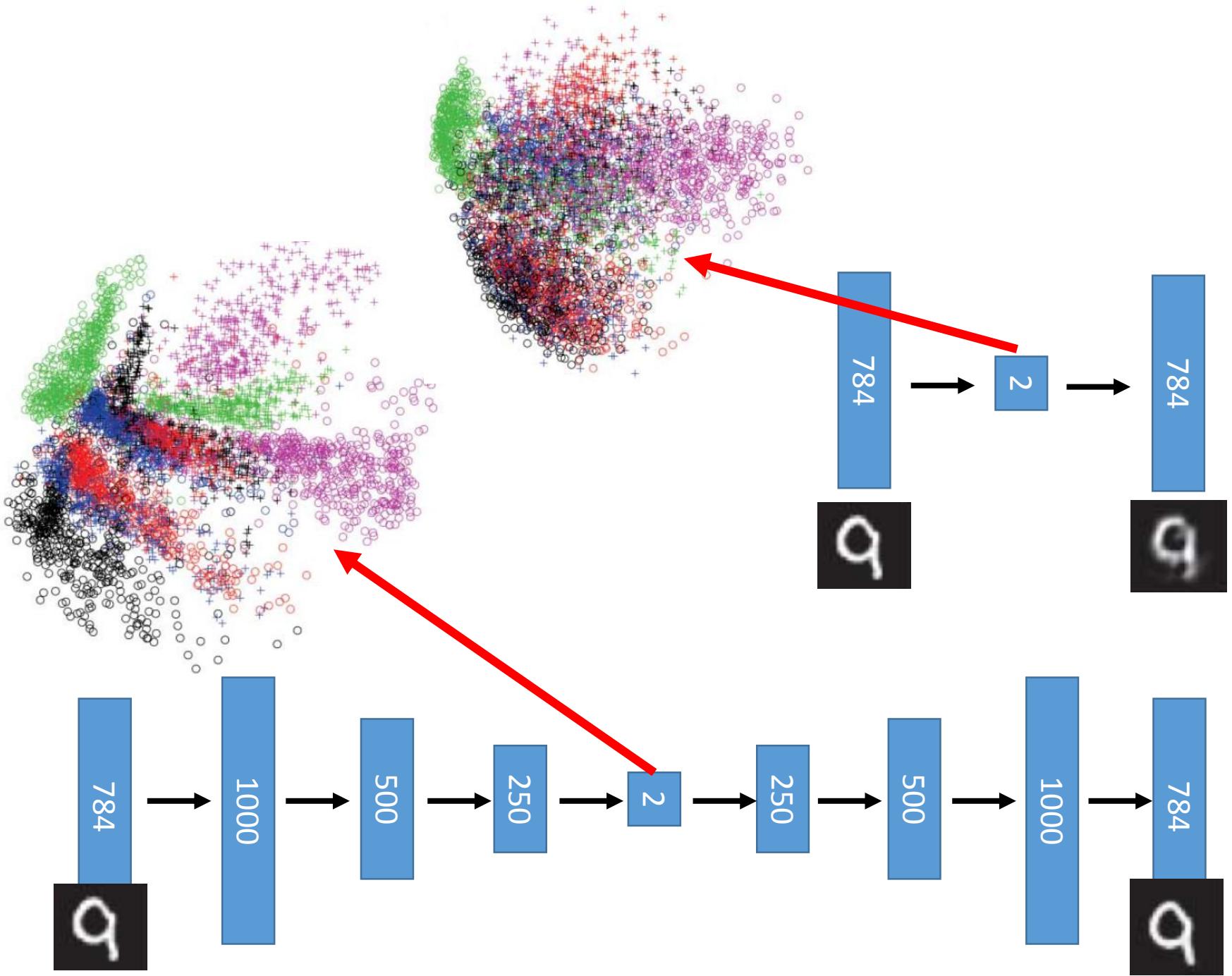


PCA



Deep Auto-encoder



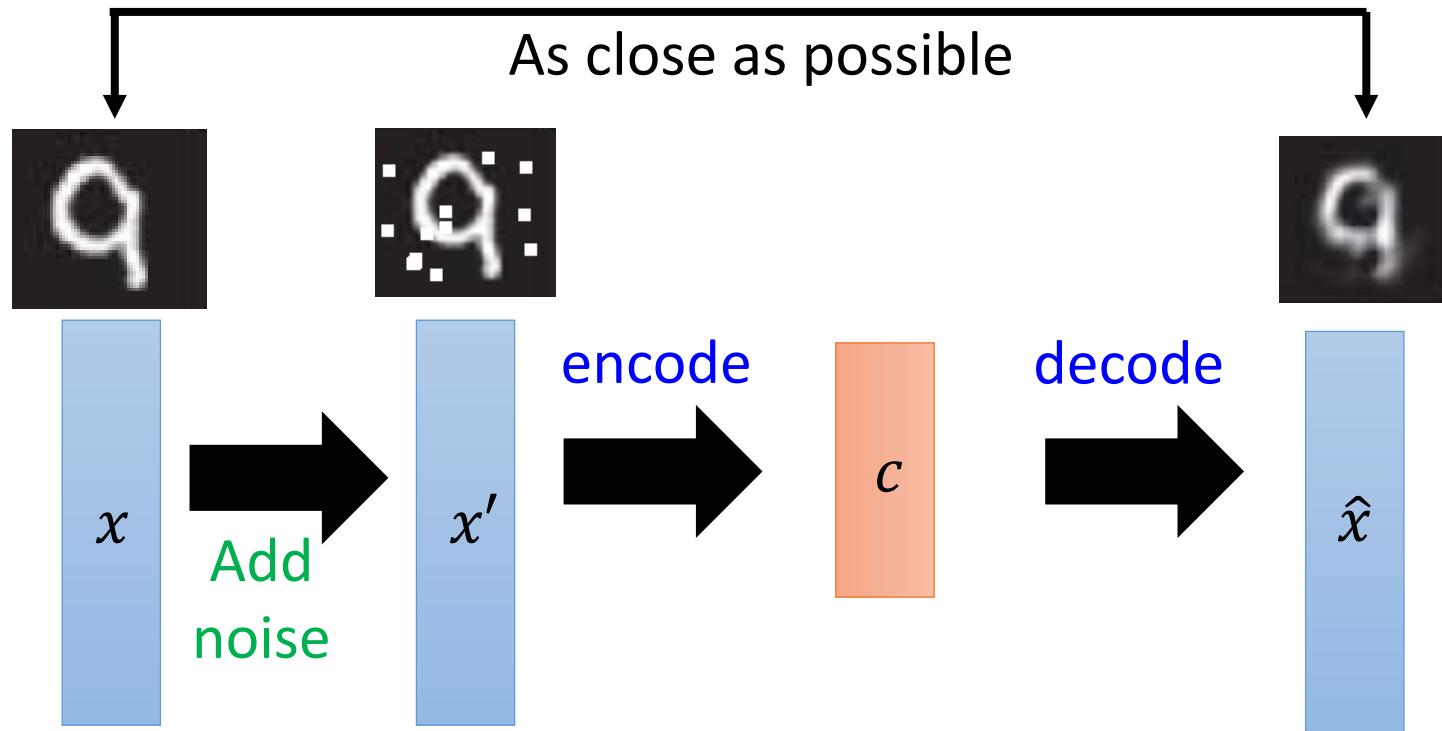


Auto-encoder

- De-noising auto-encoder

More: Contractive auto-encoder

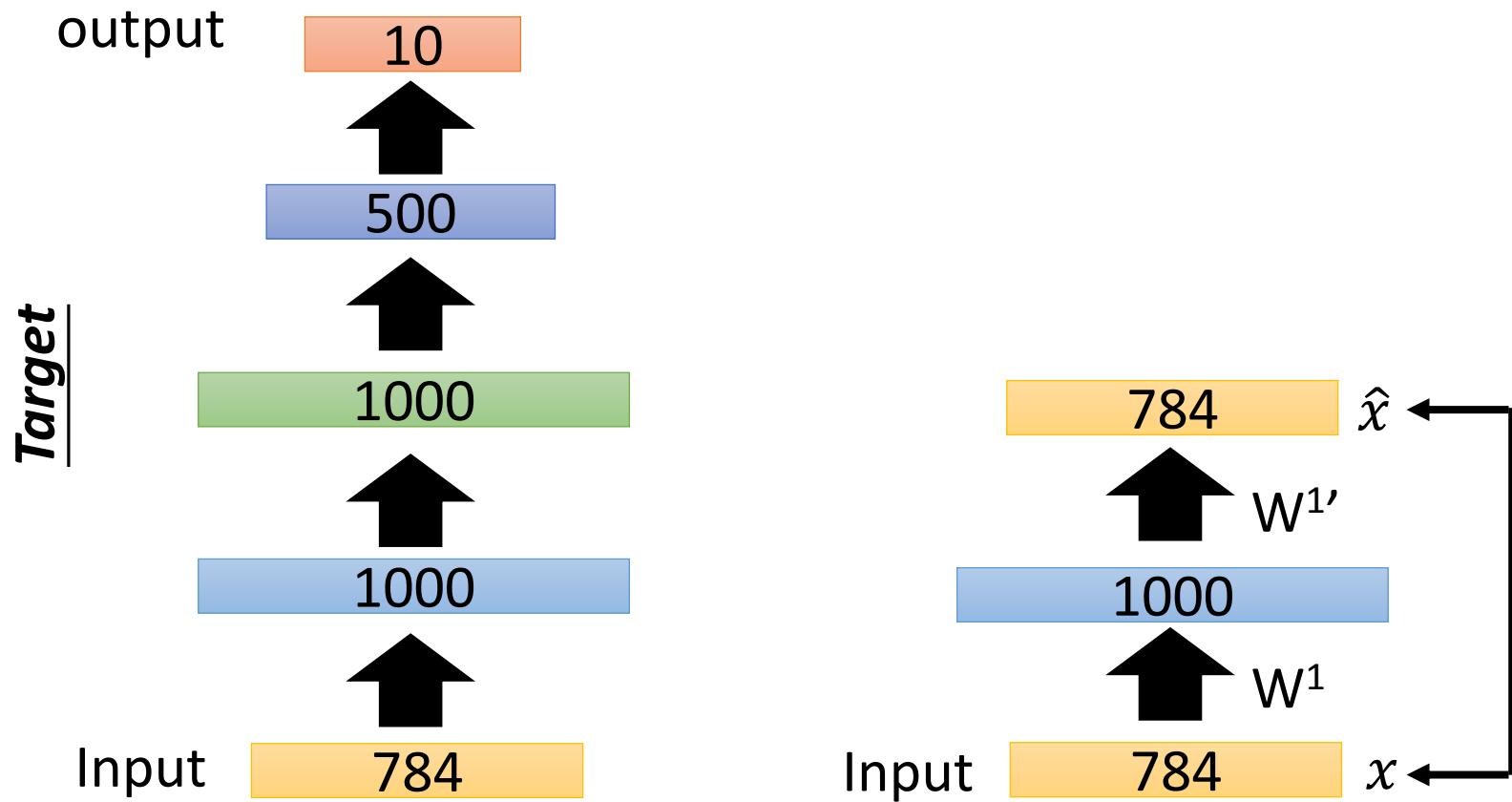
Ref: Rifai, Salah, et al. "Contractive auto-encoders: Explicit invariance during feature extraction." *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*. 2011.



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

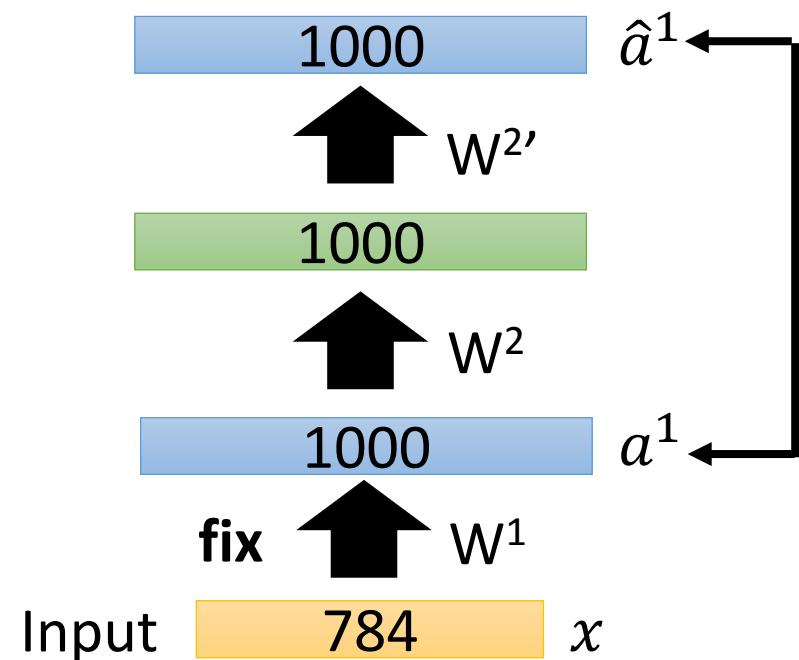
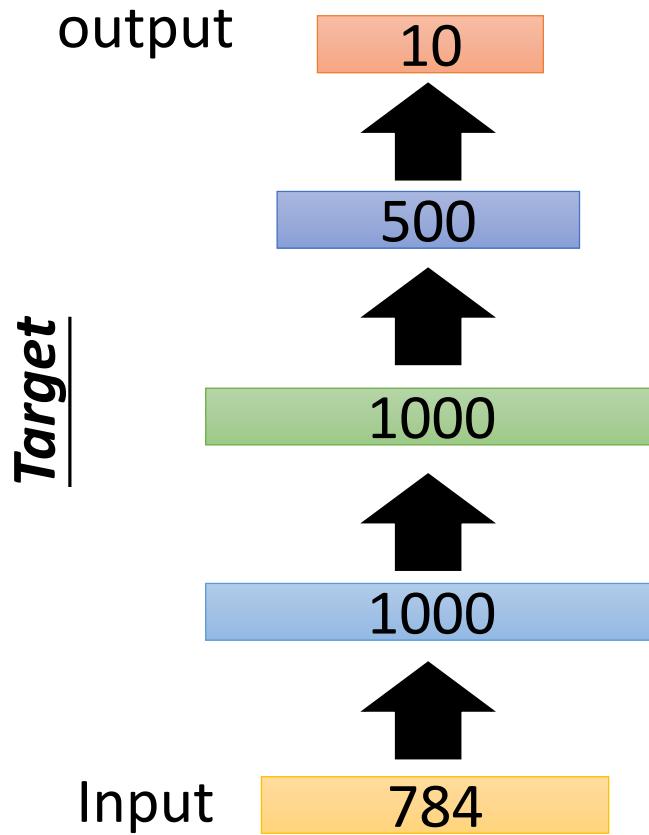
Auto-encoder – Pre-training DNN

- Greedy Layer-wise Pre-training *again*



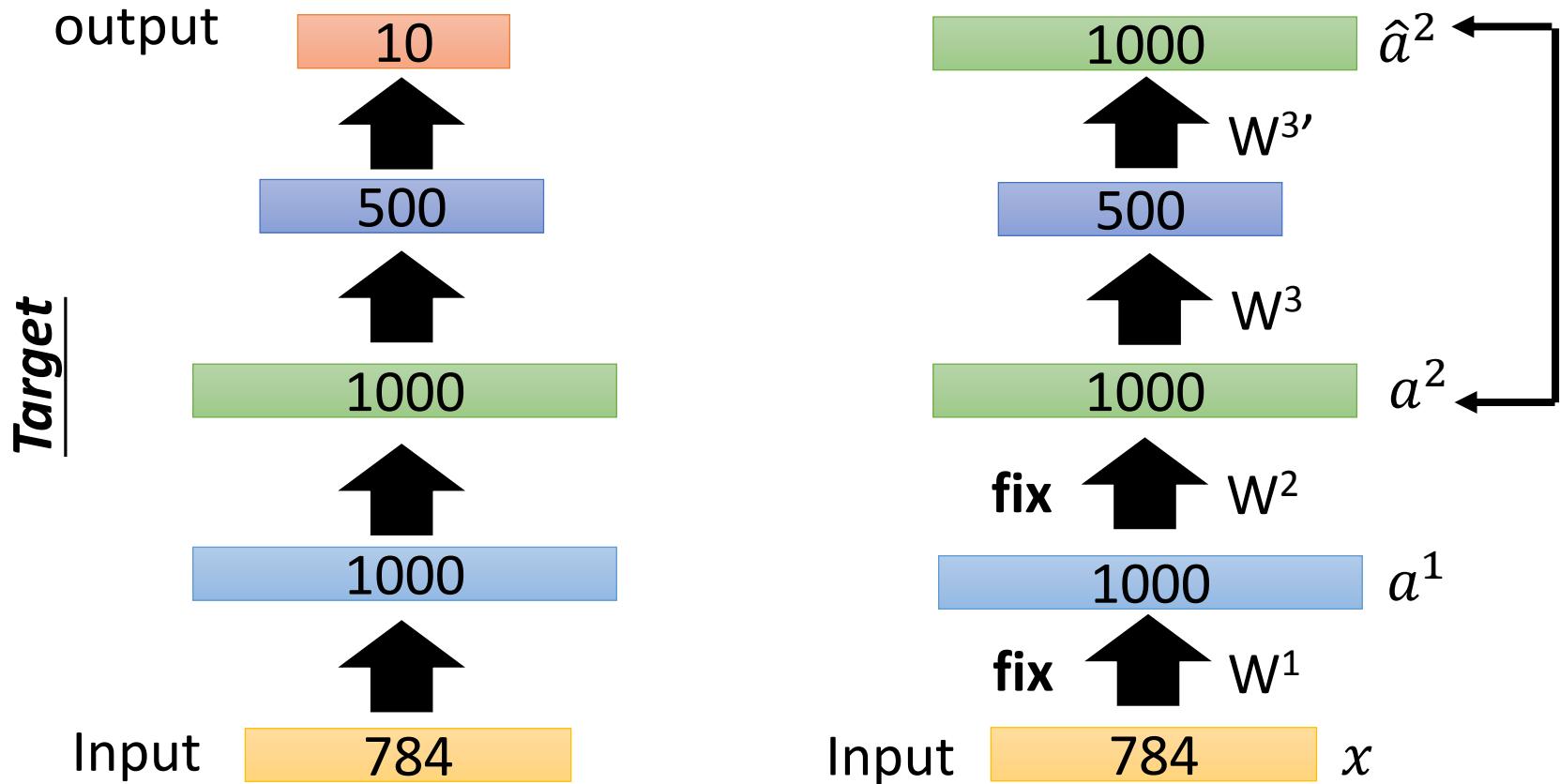
Auto-encoder – Pre-training DNN

- Greedy Layer-wise Pre-training *again*



Auto-encoder – Pre-training DNN

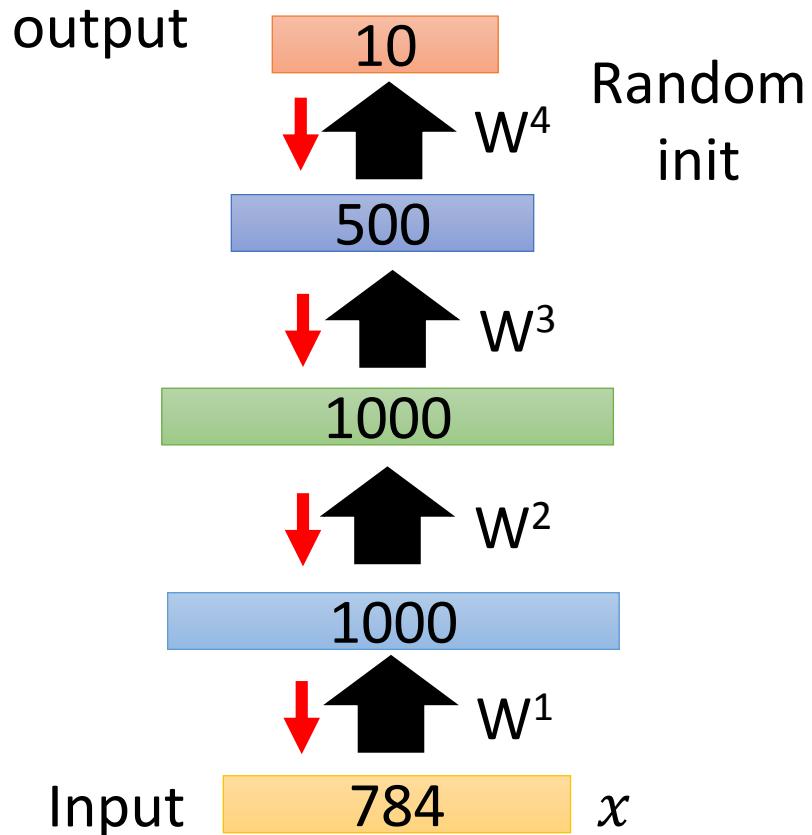
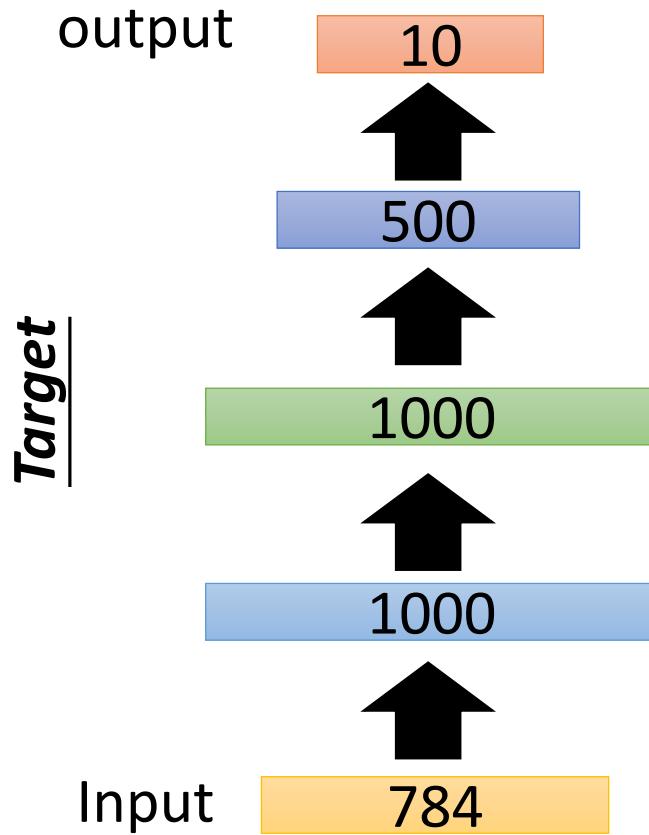
- Greedy Layer-wise Pre-training *again*



Auto-encoder – Pre-training DNN

- Greedy Layer-wise Pre-training *again*

Find-tune by
backpropagation



Outline

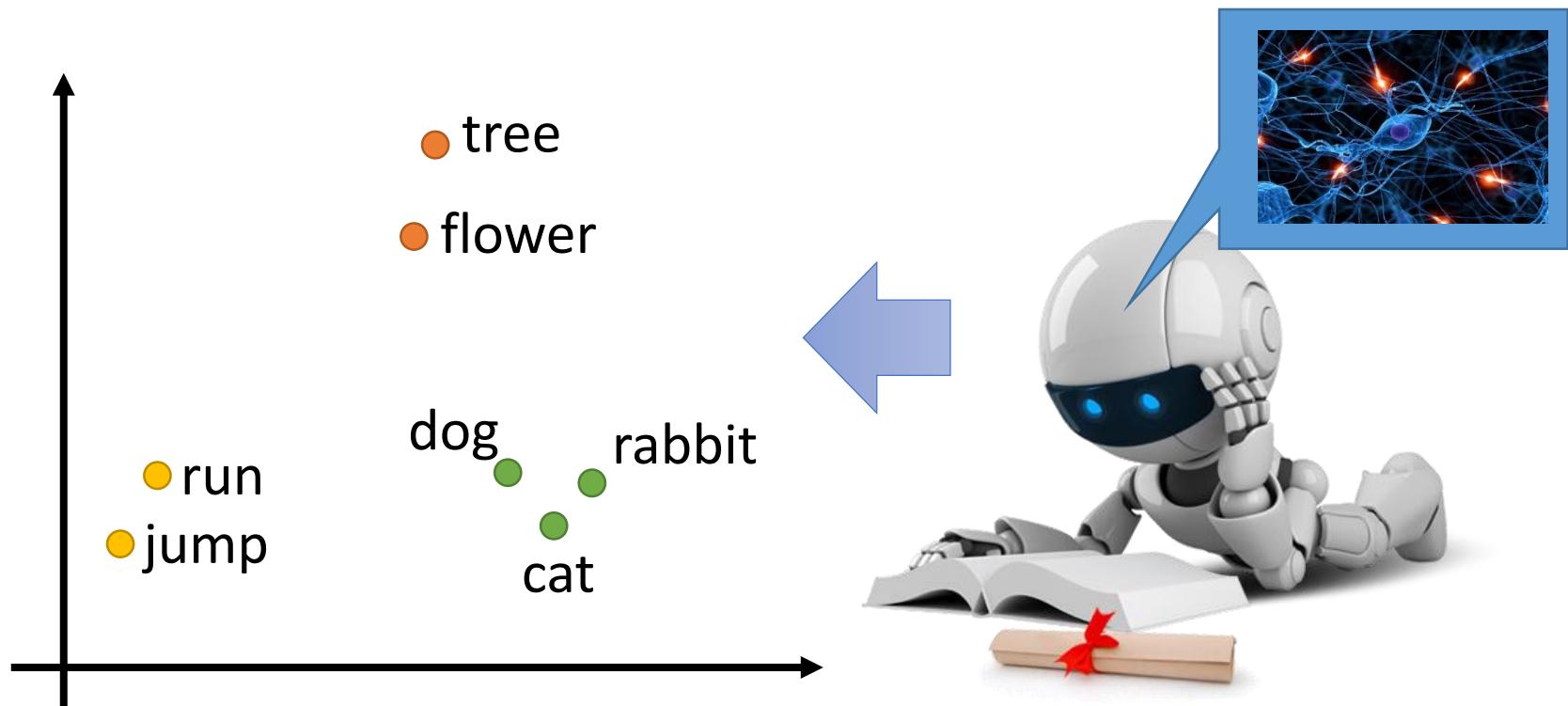
Unsupervised Learning

- 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/>

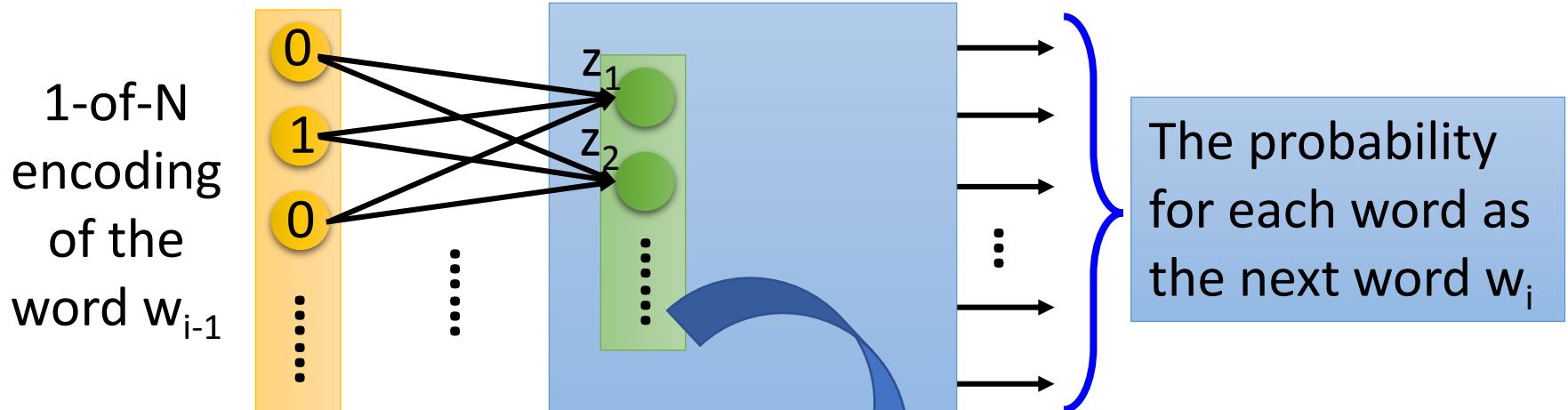
 $V(w_i) \cdot V(w_j)$  $N_{i,j}$

Inner product

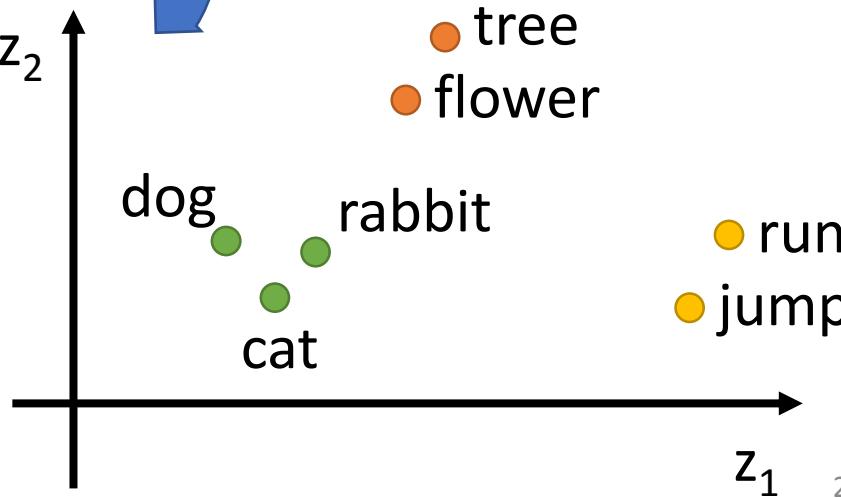
Number of times w_i and w_j
in the same document

- **Prediction based**

Prediction-based



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

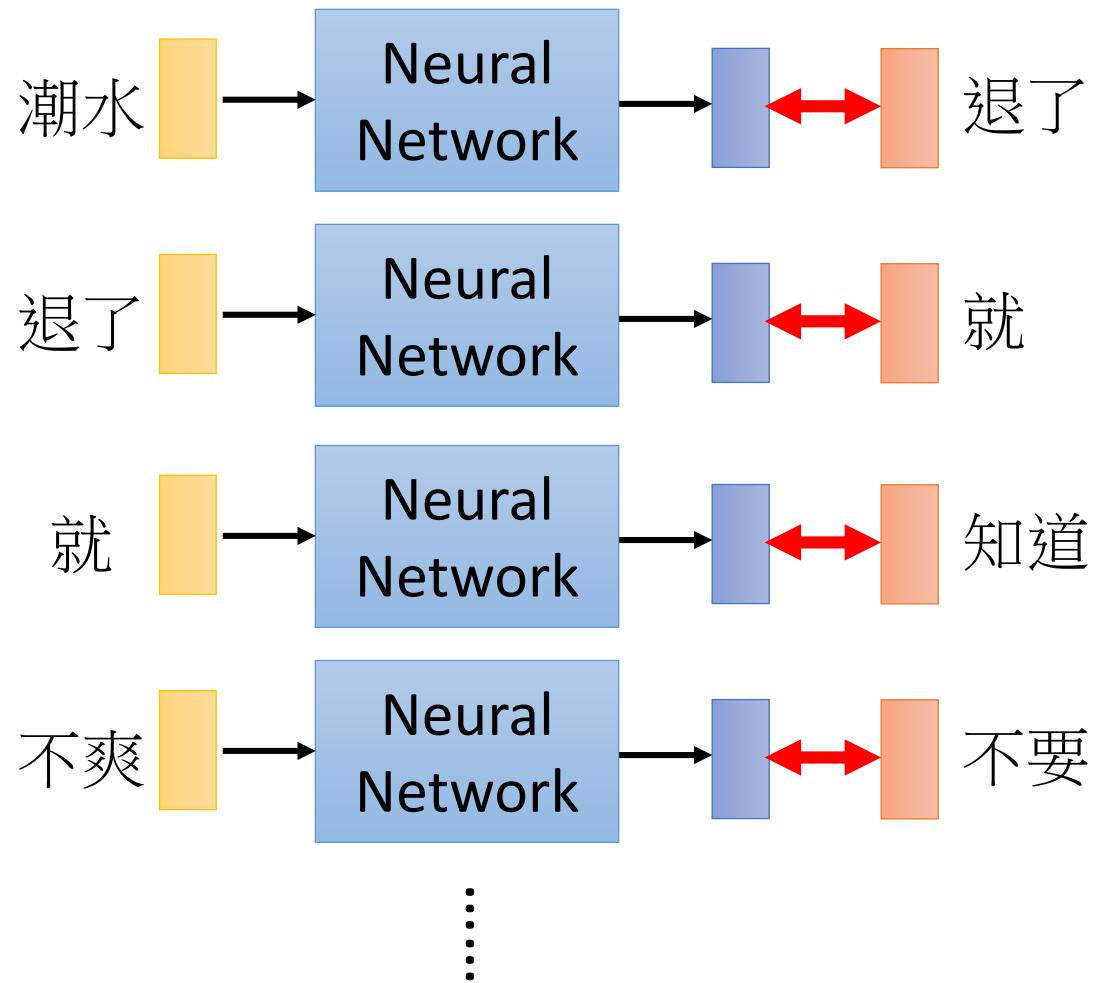


Prediction-based

Minimizing
cross entropy

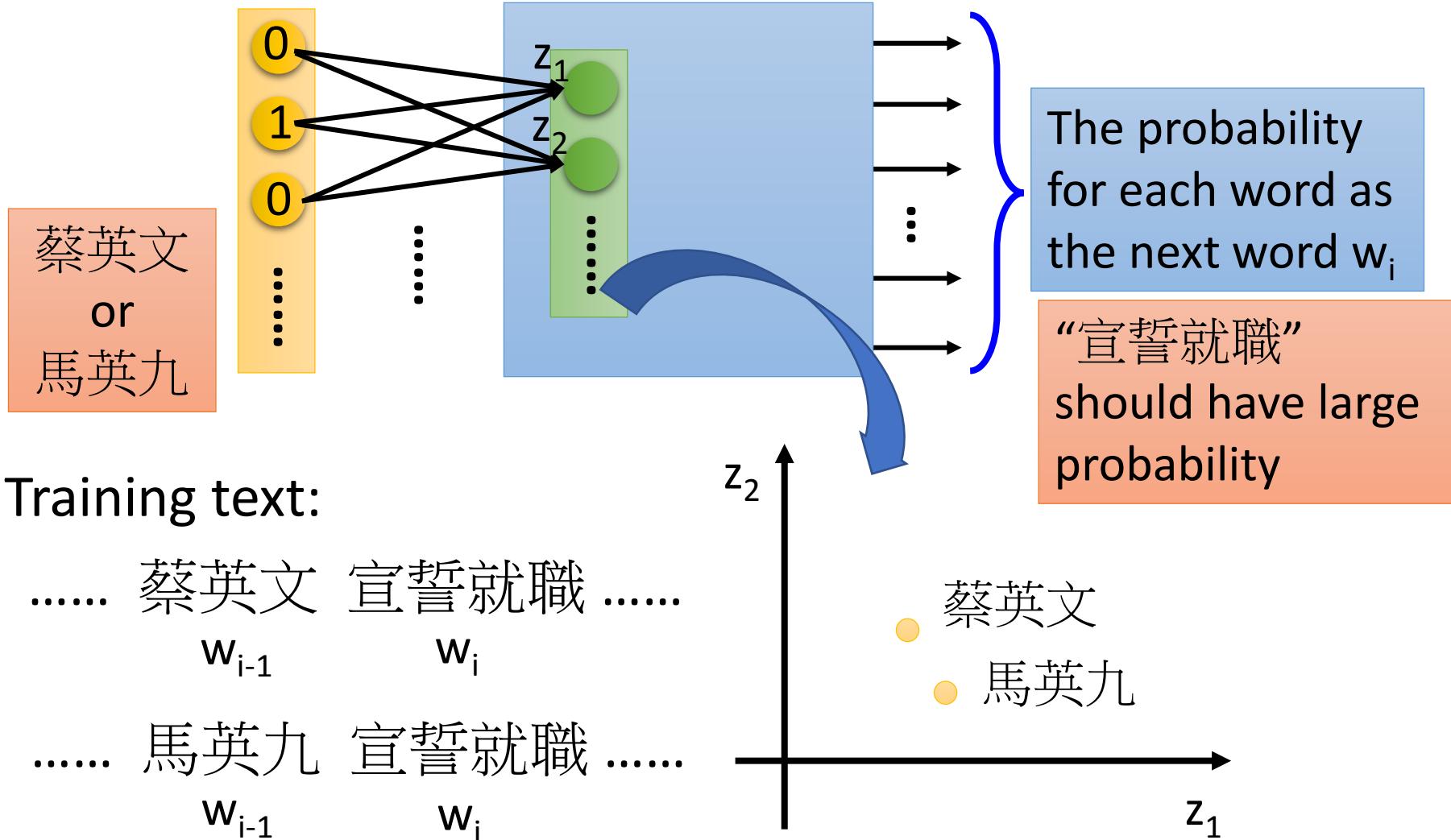
Collect data:

潮水 退了 就 知道 ...
不爽 不要 買 ...
公道價 八萬 一 ...
.....



Prediction-based

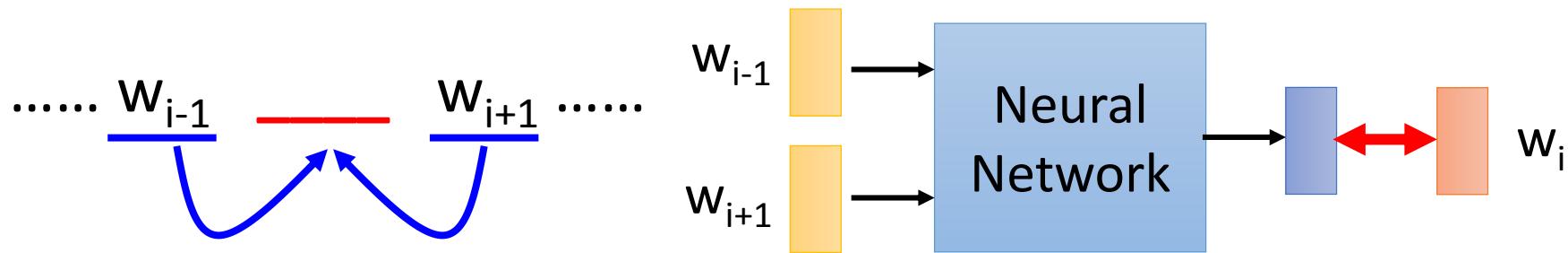
You shall know a word
by the company it keeps



Prediction-based

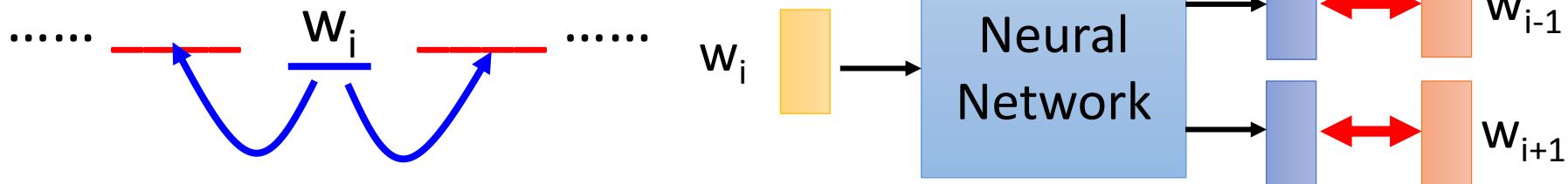
– Various Architectures

- Continuous bag of word (CBOW) model



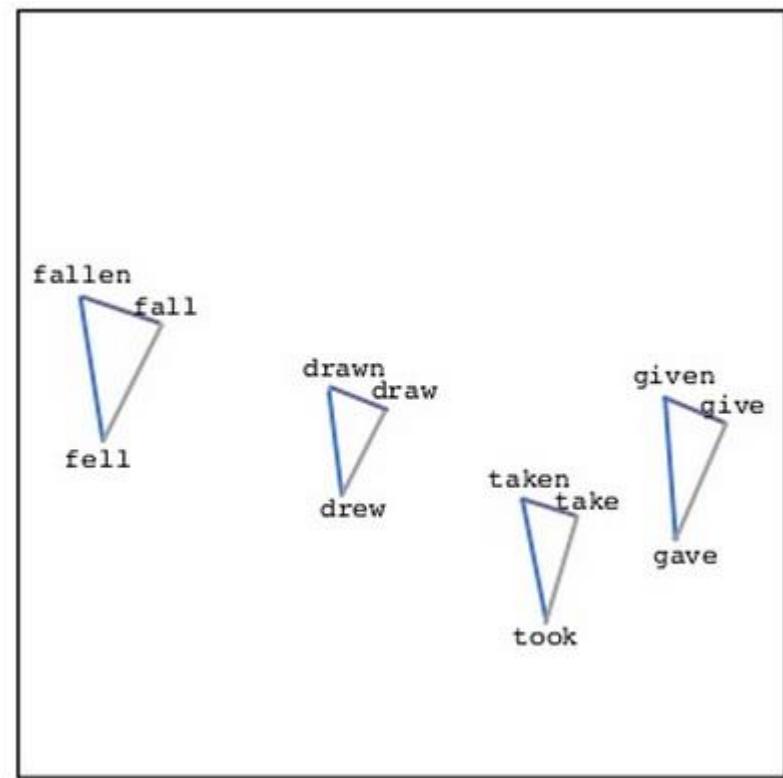
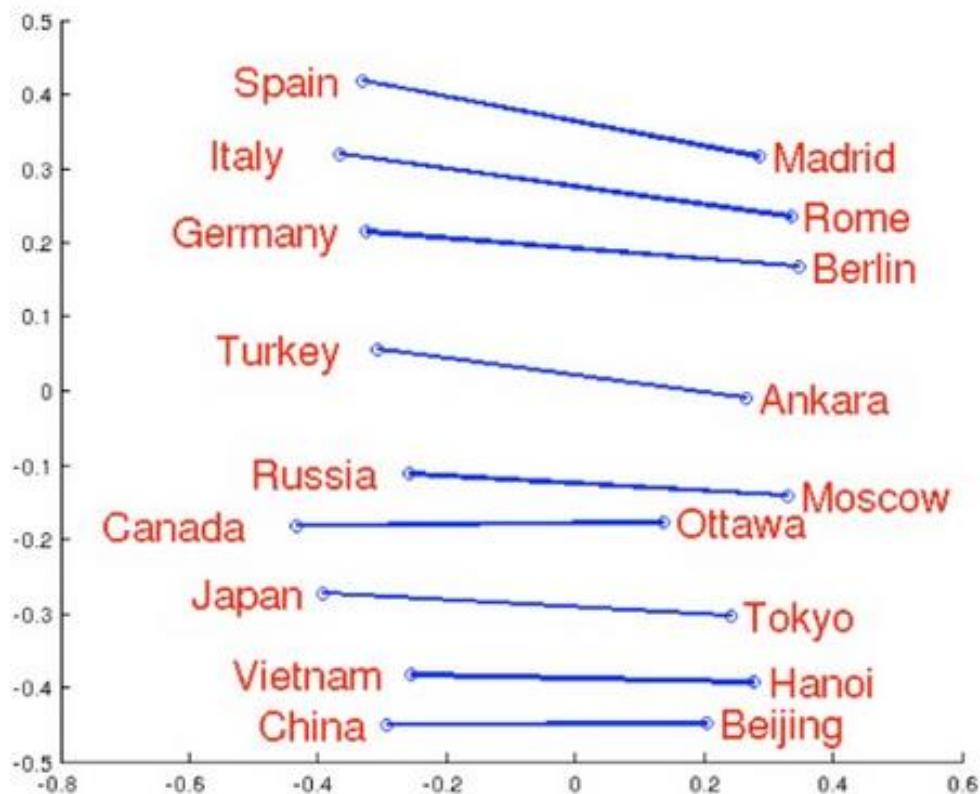
predicting the word given its context

- 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

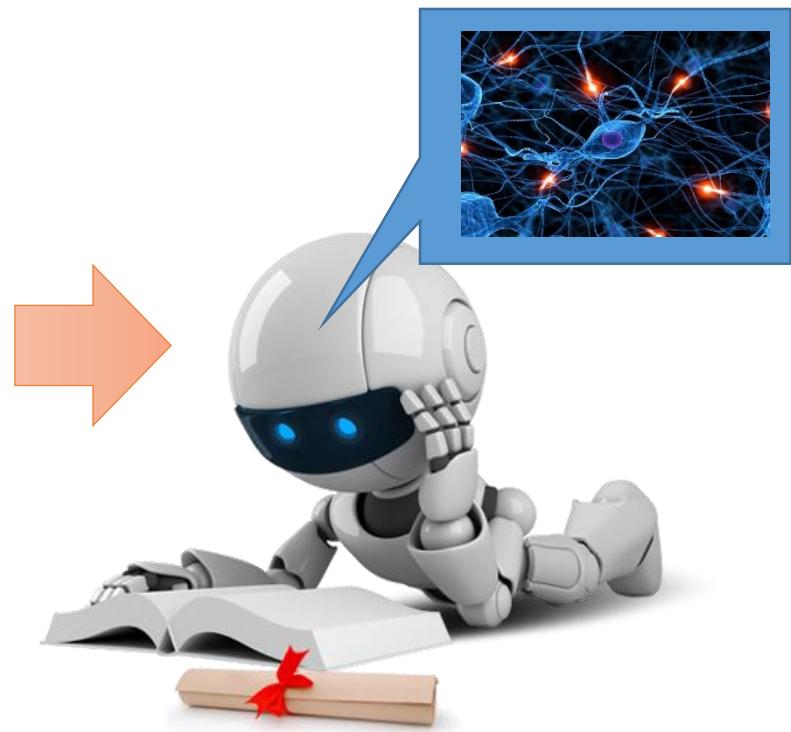
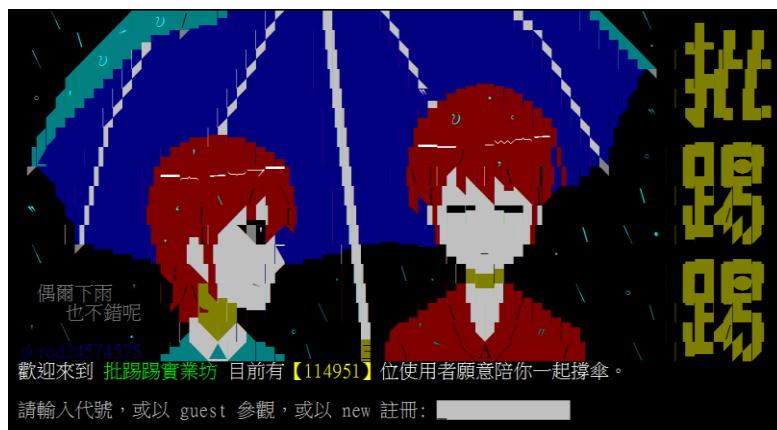
Rome : Italy = Berlin : ?

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

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



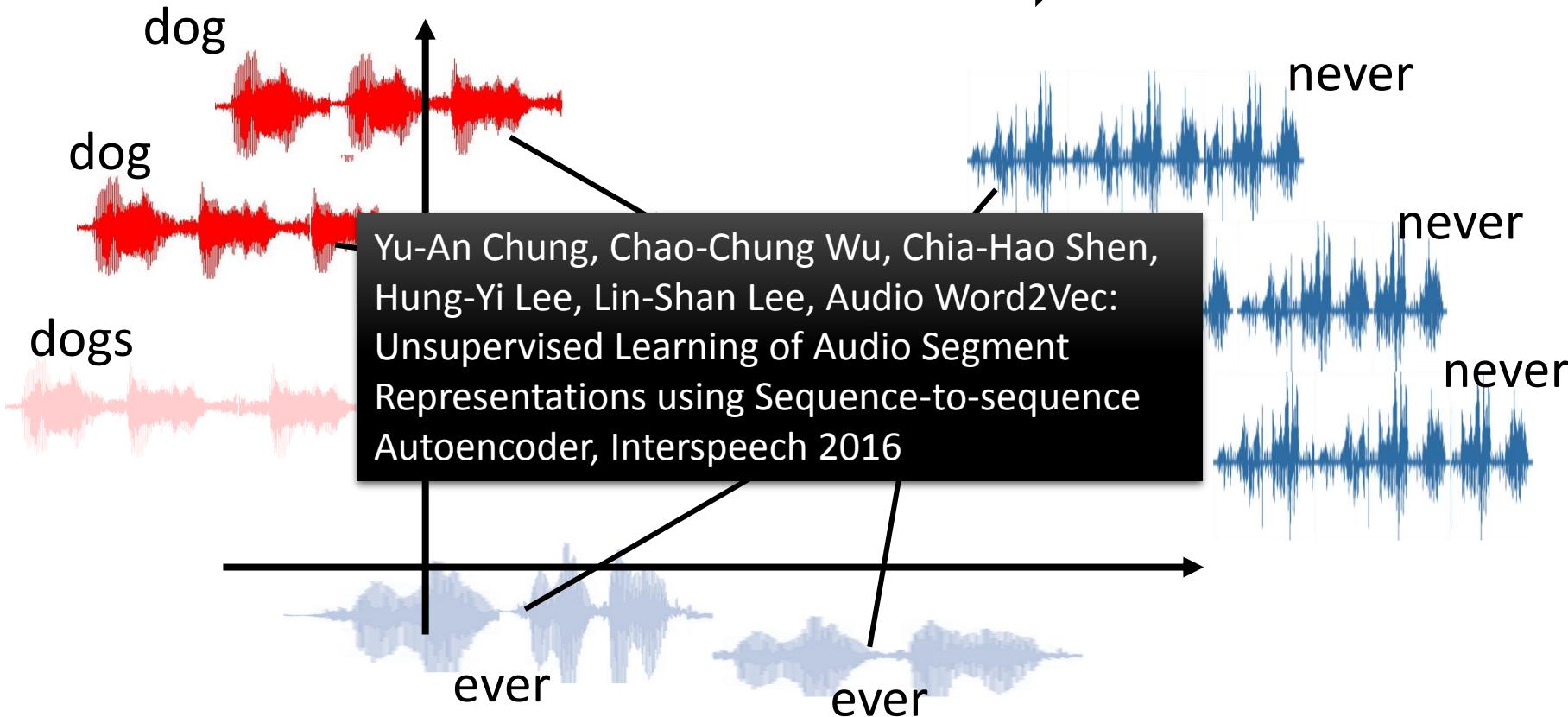
Machine does not have any prior knowledge

Machine listens to lots of audio book

Like an infant

Audio Word to Vector

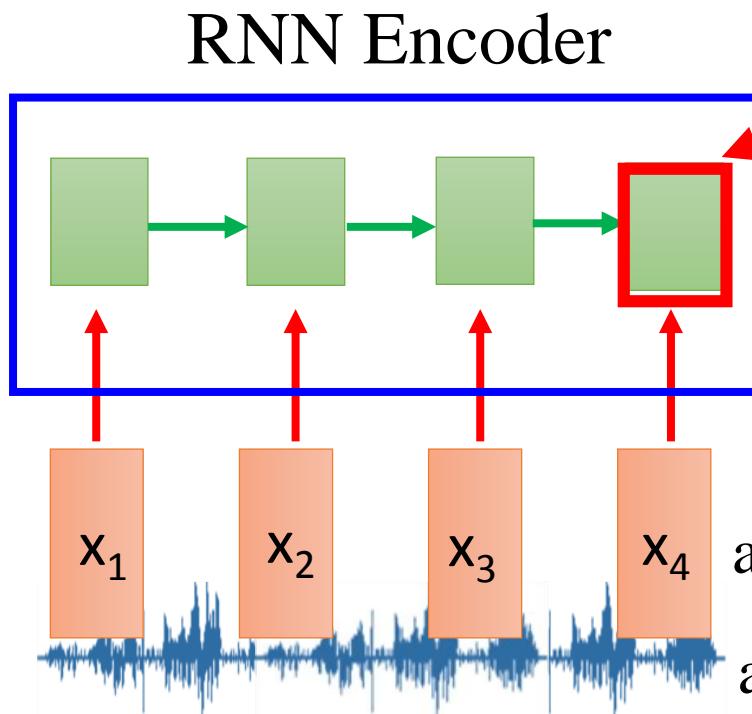
- Dimension reduction for a sequence with variable length
audio segments (word-level)  Fixed-length vector



Sequence-to-sequence Auto-encoder



vector



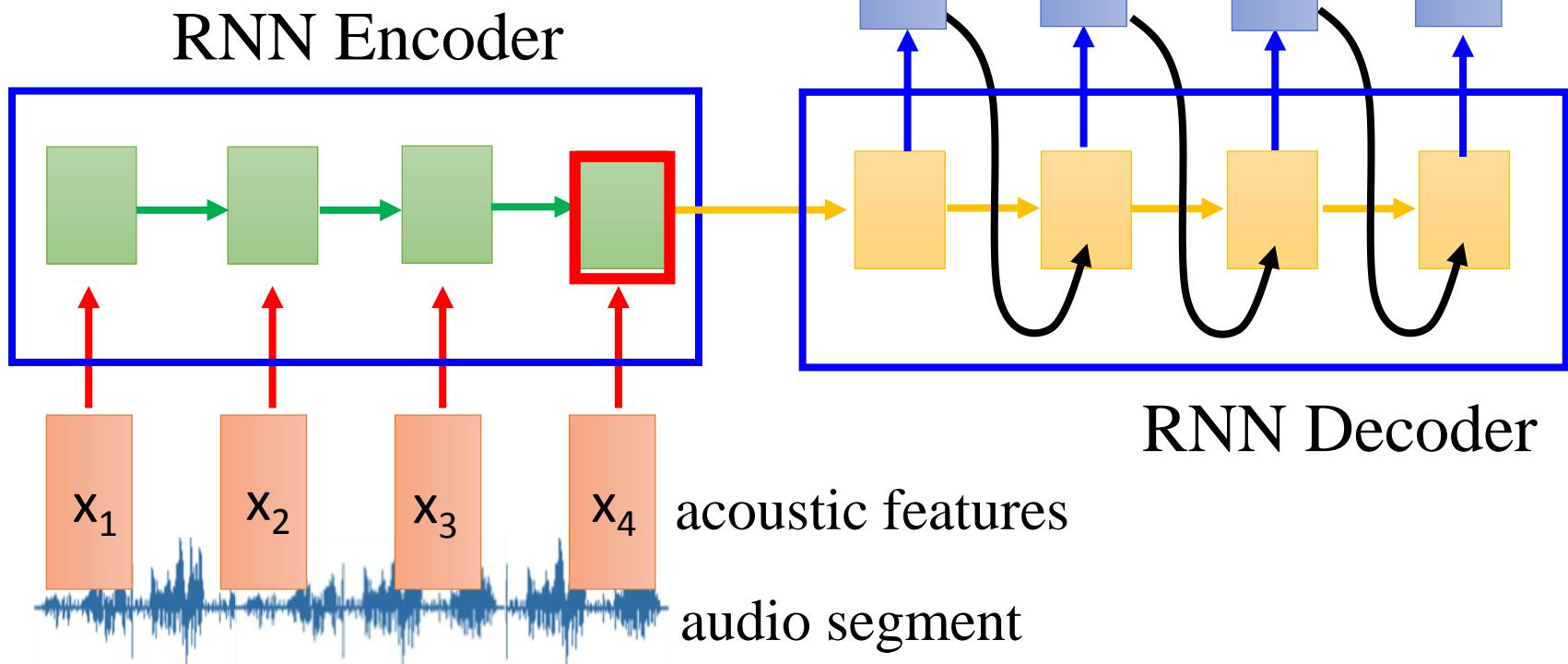
The values in the memory
represent the whole audio
segment

The vector we want

How to train RNN Encoder?

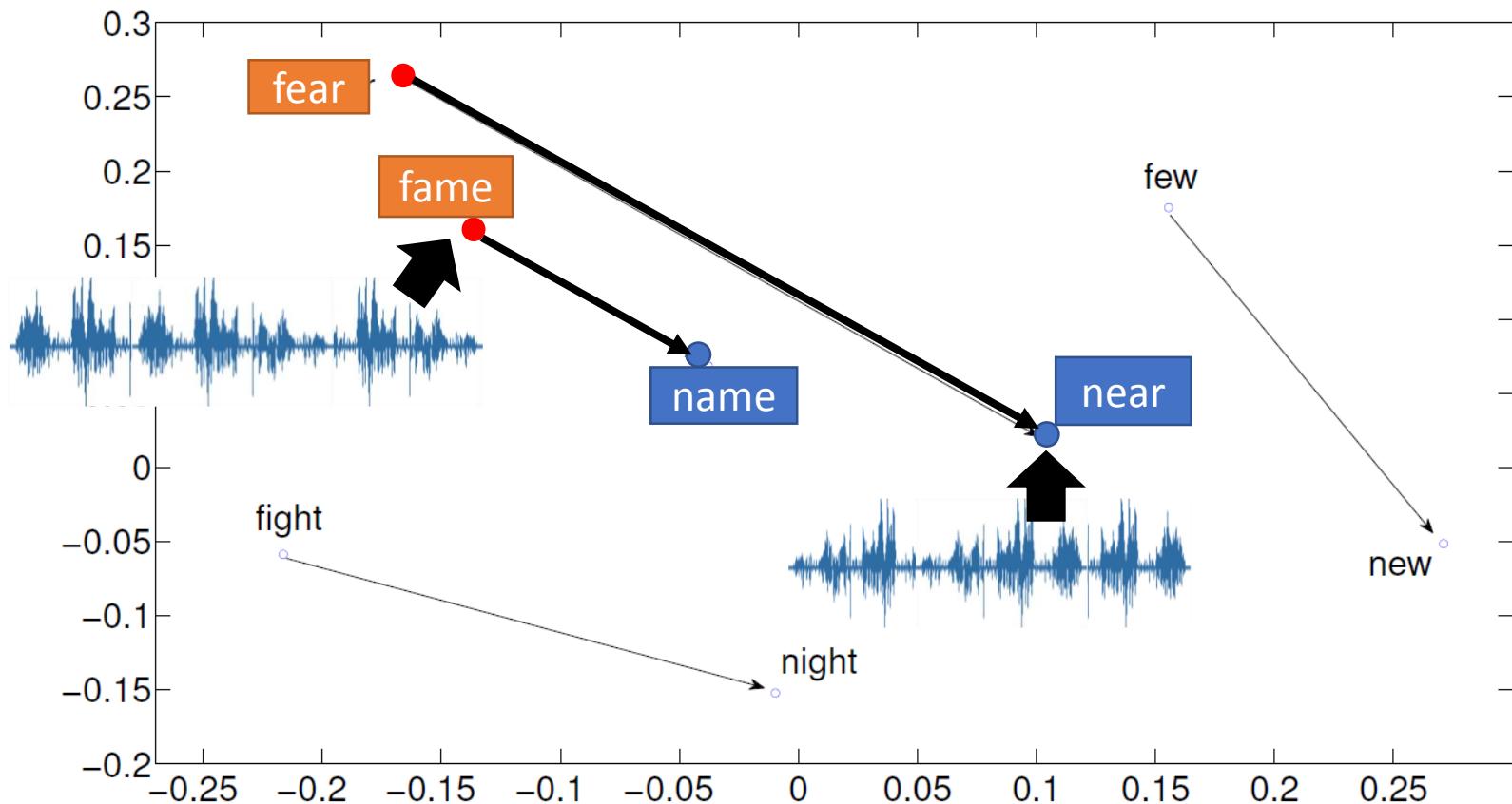
Sequence-to-sequence Auto-encoder

The RNN encoder and
decoder are jointly trained.

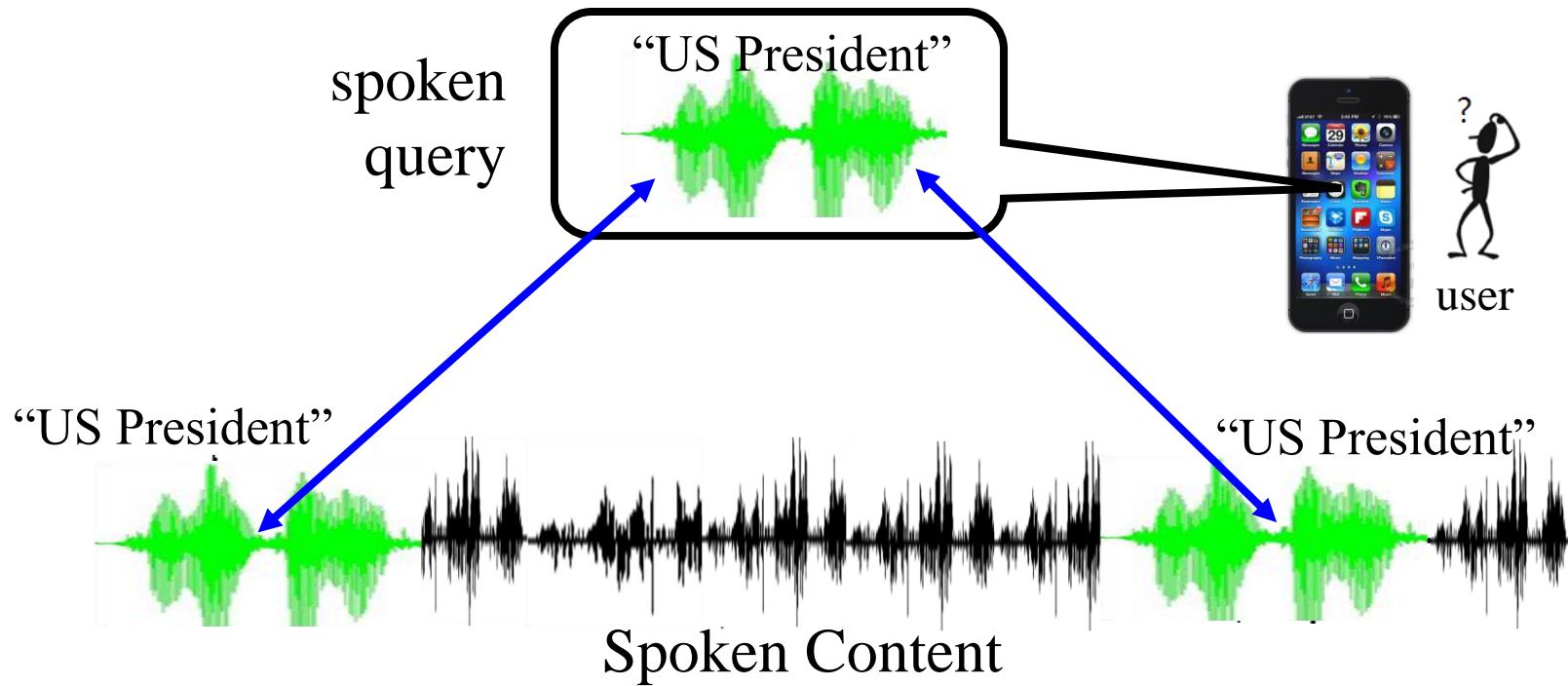


Sequence-to-sequence Auto-encoder

- Visualizing embedding vectors of the words



Audio Word to Vector —Application

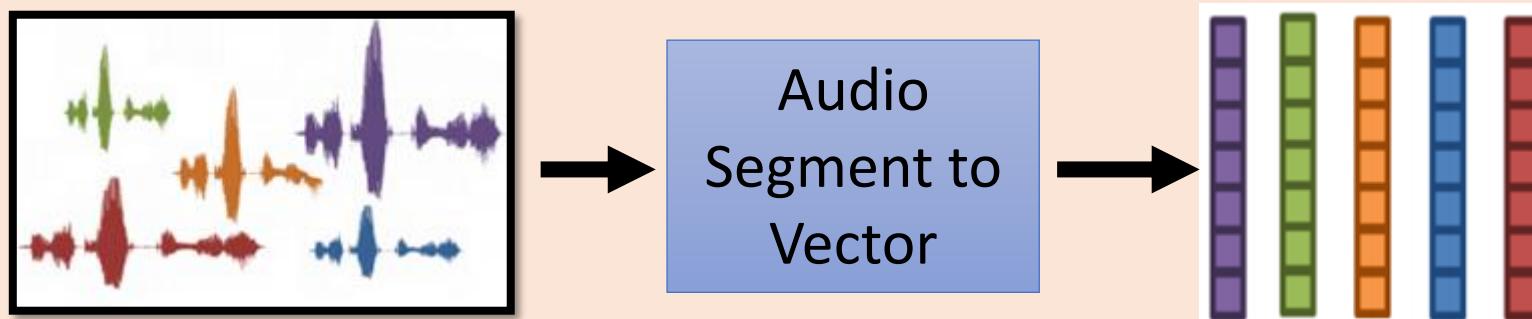


Compute similarity between spoken queries and audio files on acoustic level, and find the query term

Audio Word to Vector –Application

Audio archive divided into variable-length audio segments

Off-line



Spoken
Query

Audio
Segment to
Vector



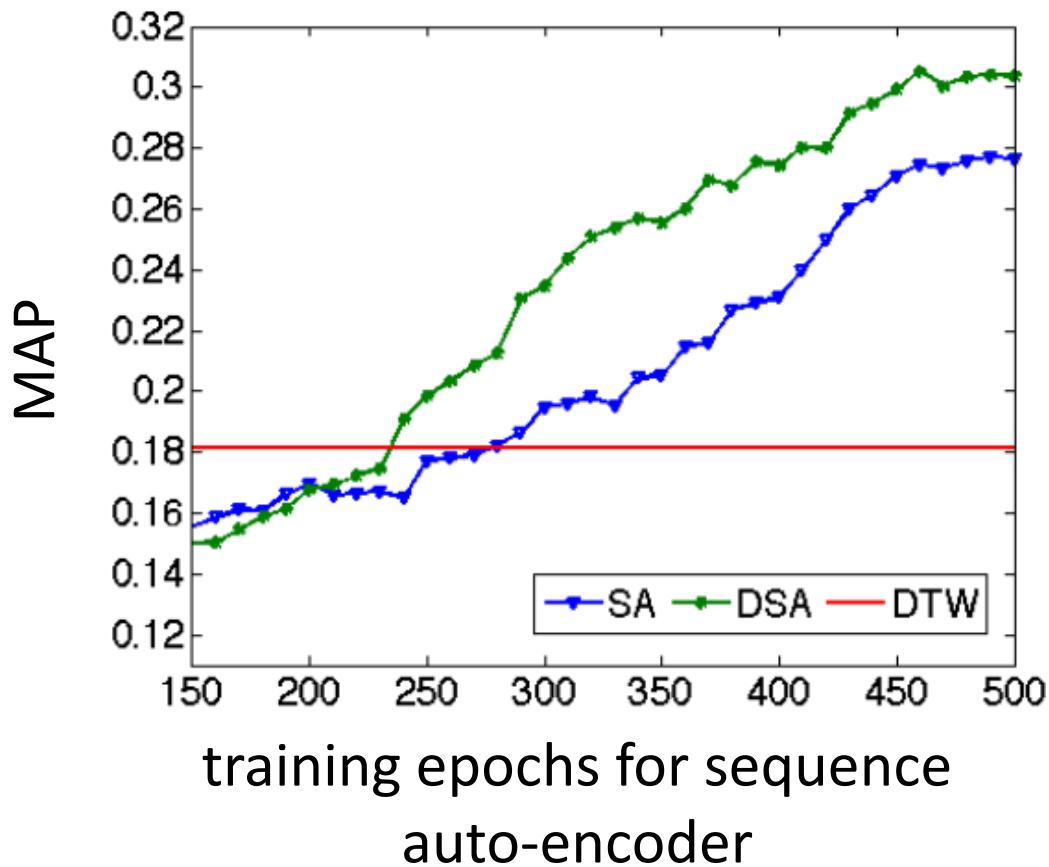
Similarity

Search Result

On-line

Experimental Results

- Query-by-Example Spoken Term Detection



SA: sequence auto-encoder

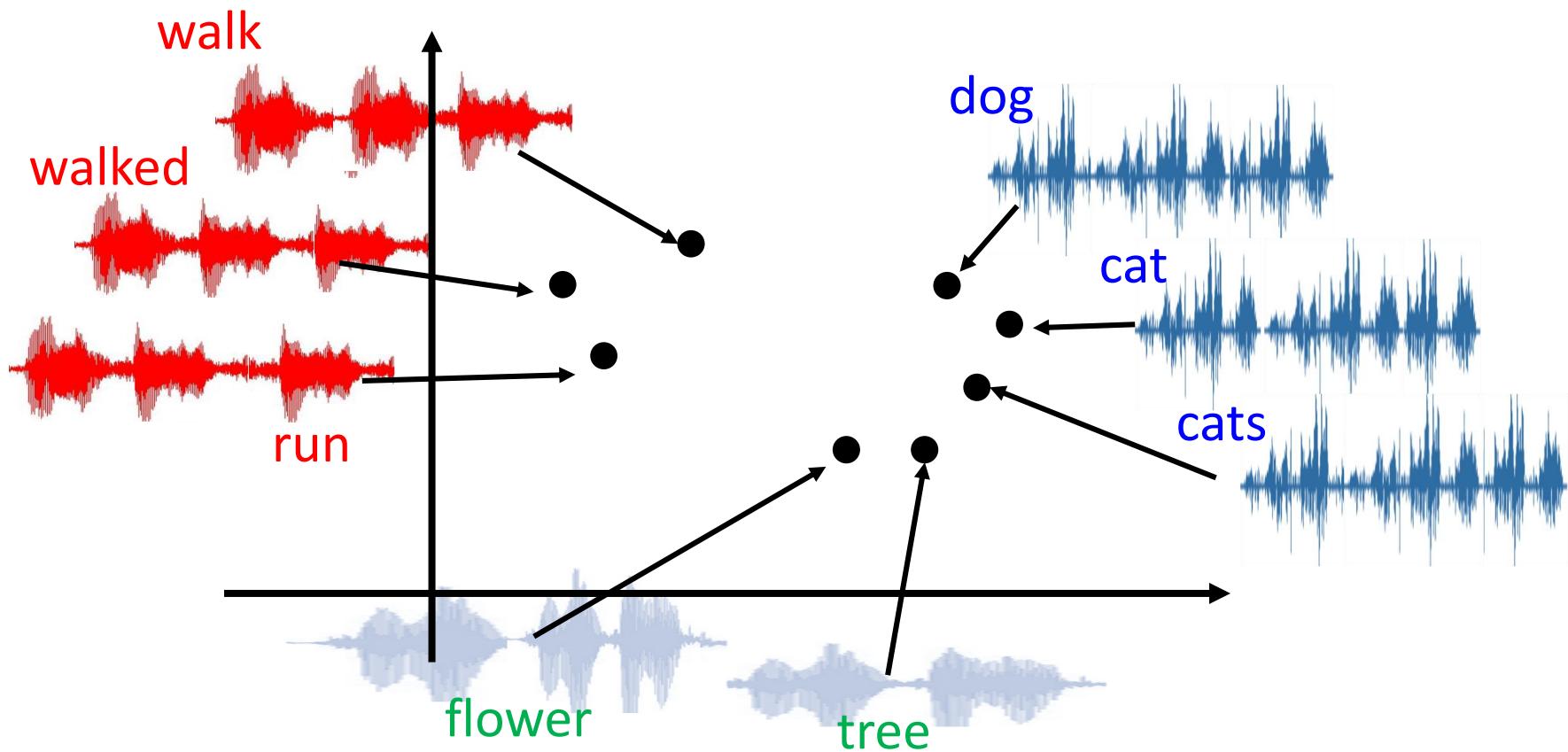
DSA: de-noising sequence auto-encoder

Input: clean speech + noise

output: clean speech

Next Step

- Can we include semantics?



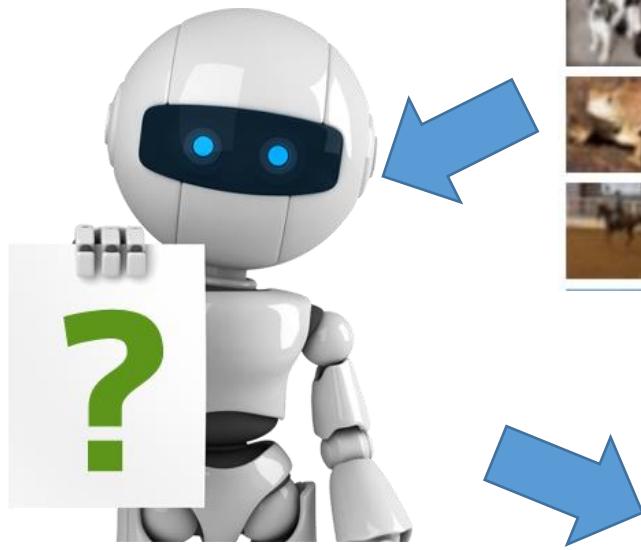
Outline

Unsupervised Learning

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

Reinforcement Learning

Creation

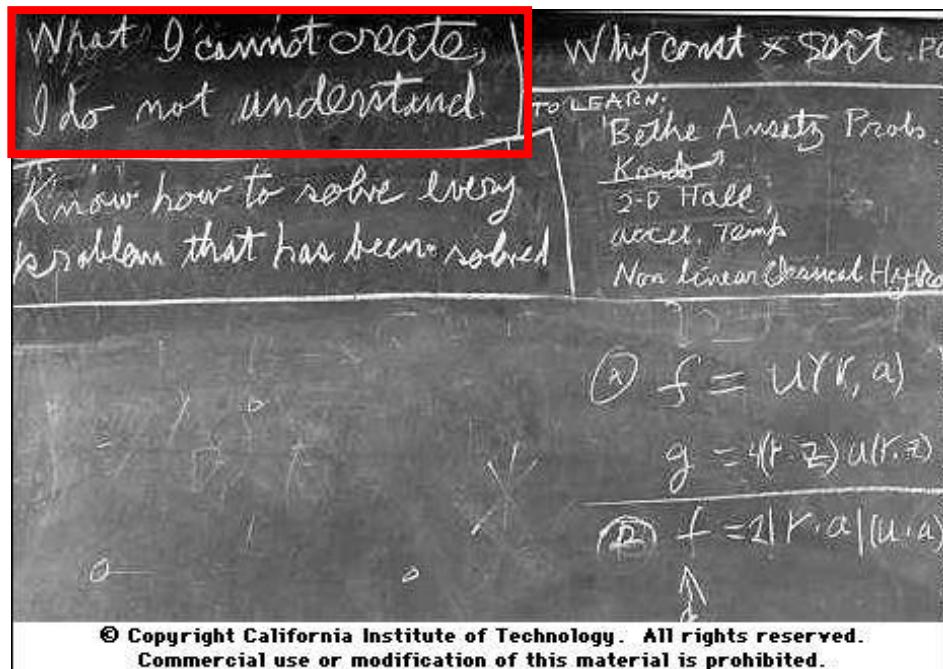


Draw something!



Creation

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



What I cannot create,
I do not understand.

Richard Feynman

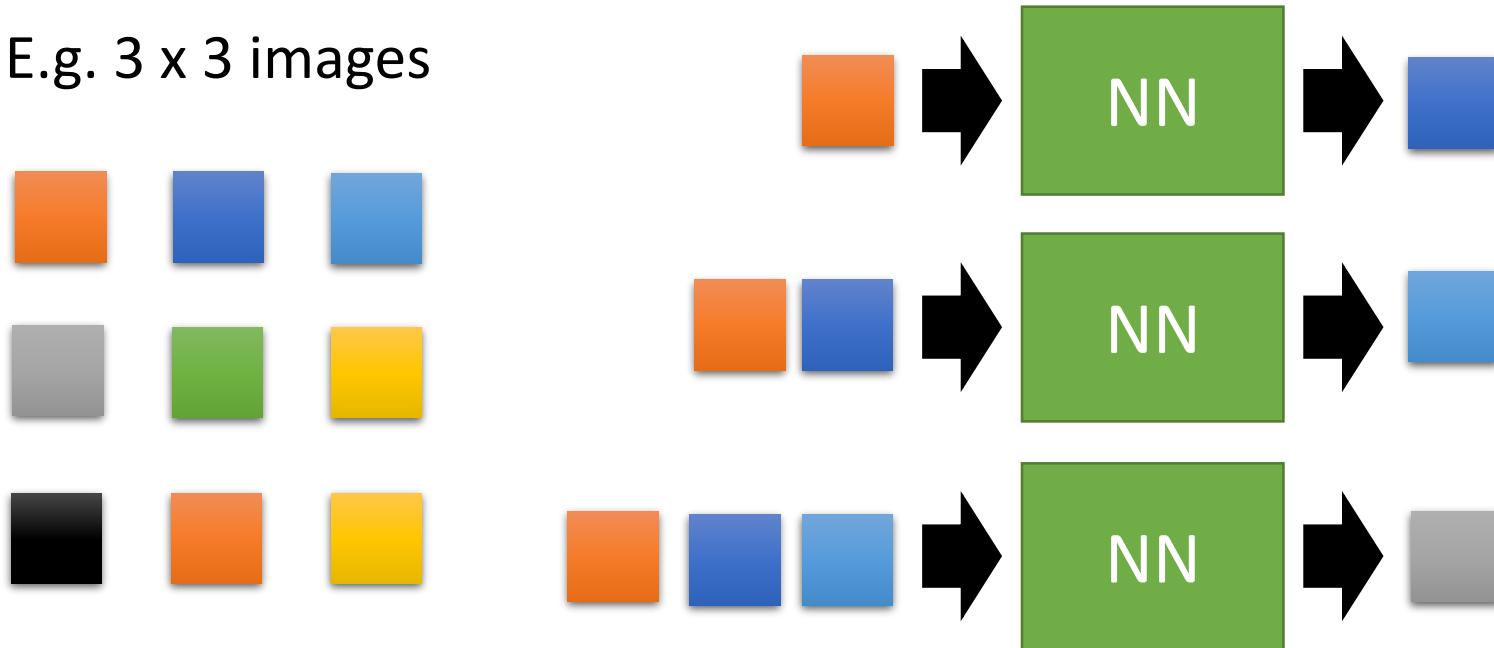
<https://www.quora.com/What-did-Richard-Feynman-mean-when-he-said-What-I-cannot-create-I-do-not-understand>

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



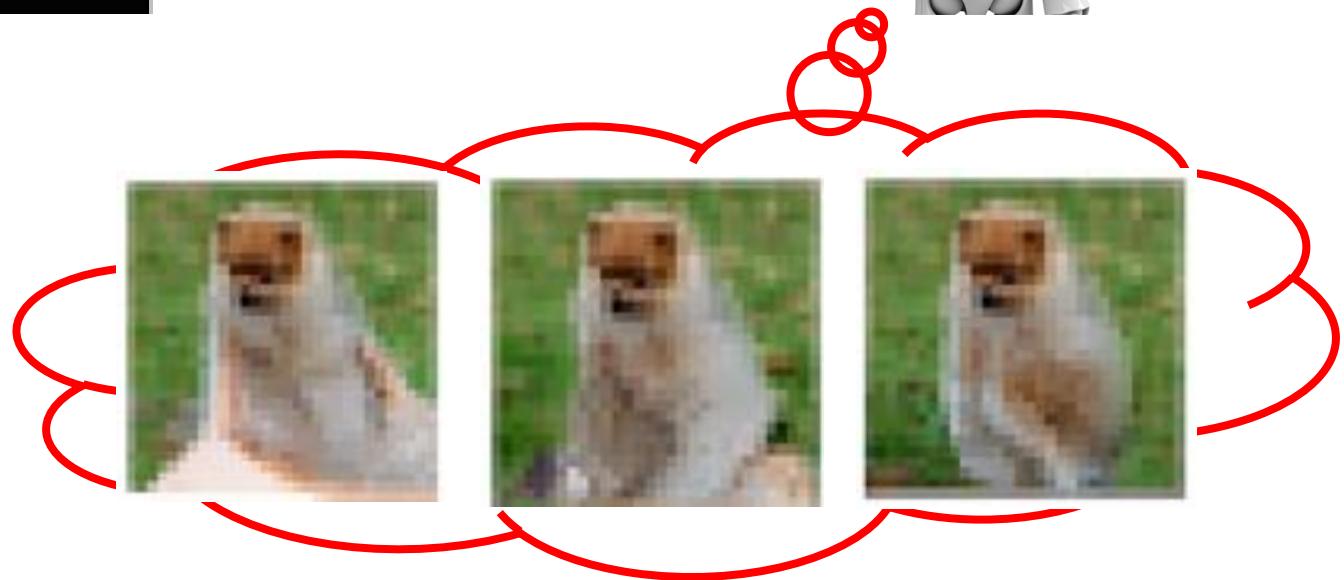
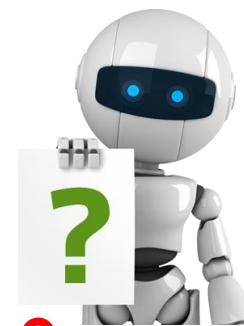
Can be trained just with a large collection
of images without any annotation

PixelRNN

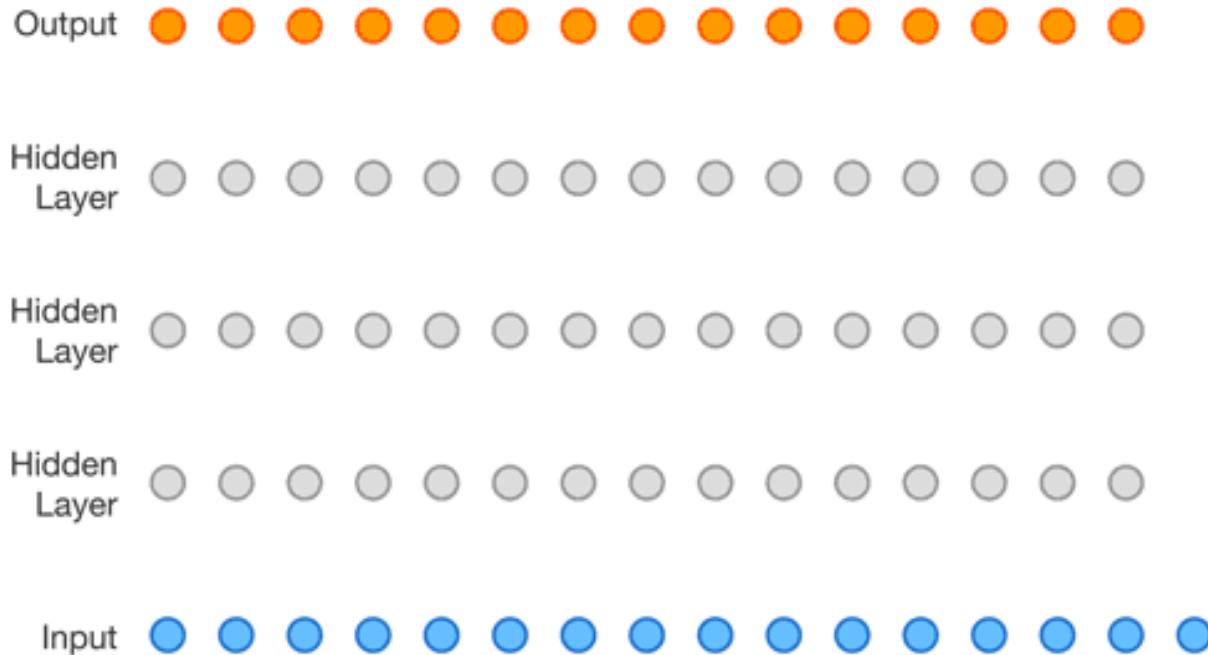
Ref: Aaron van den Oord, Nal Kalchbrenner, Koray Kavukcuoglu, Pixel Recurrent Neural Networks, arXiv preprint, 2016



Real
World



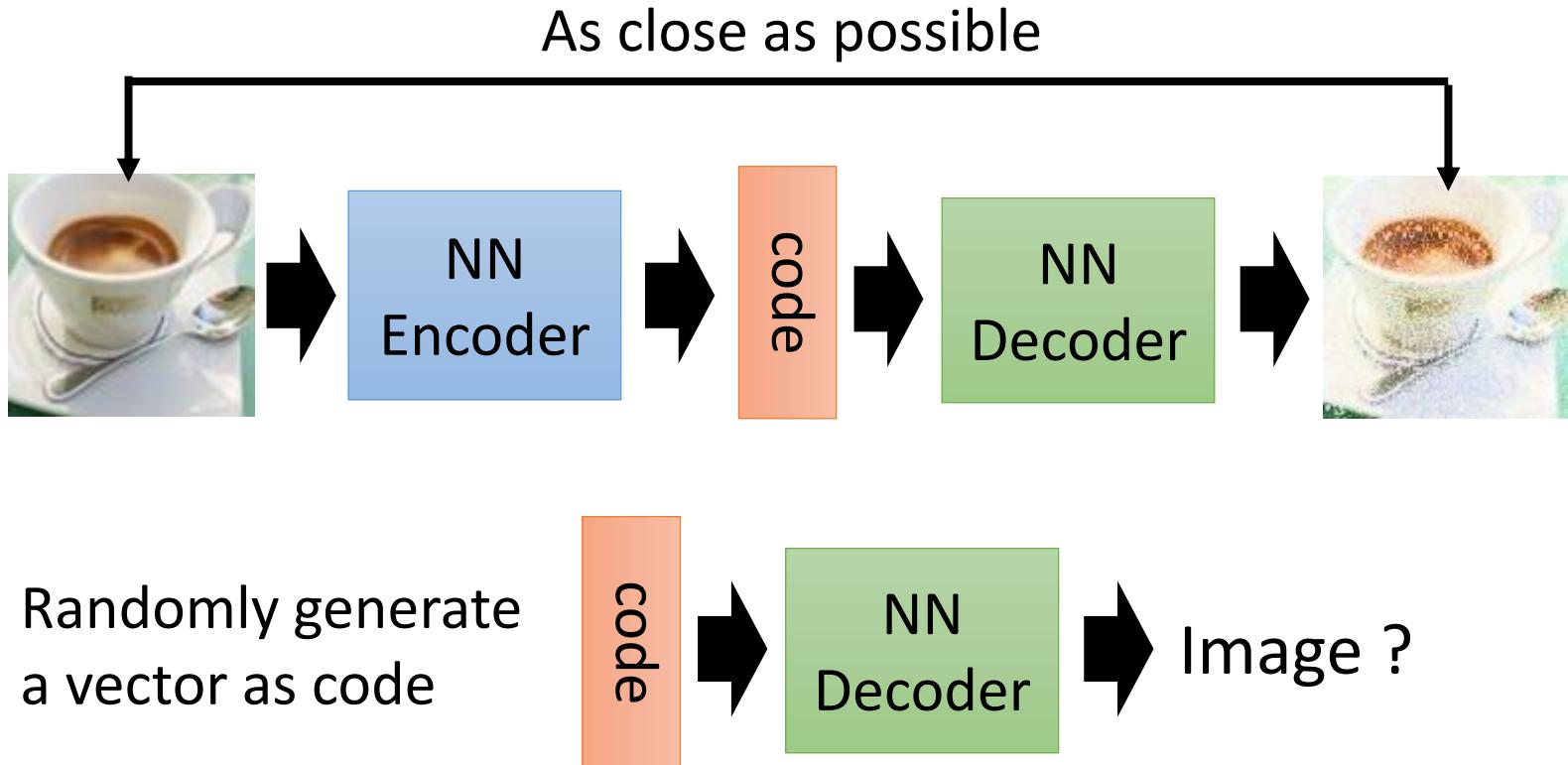
PixelRNN – beyond Image



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

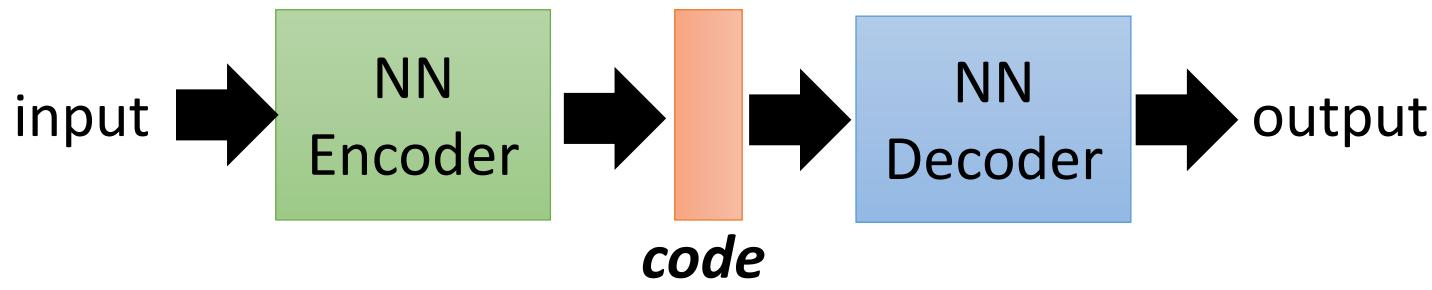
Auto-encoder



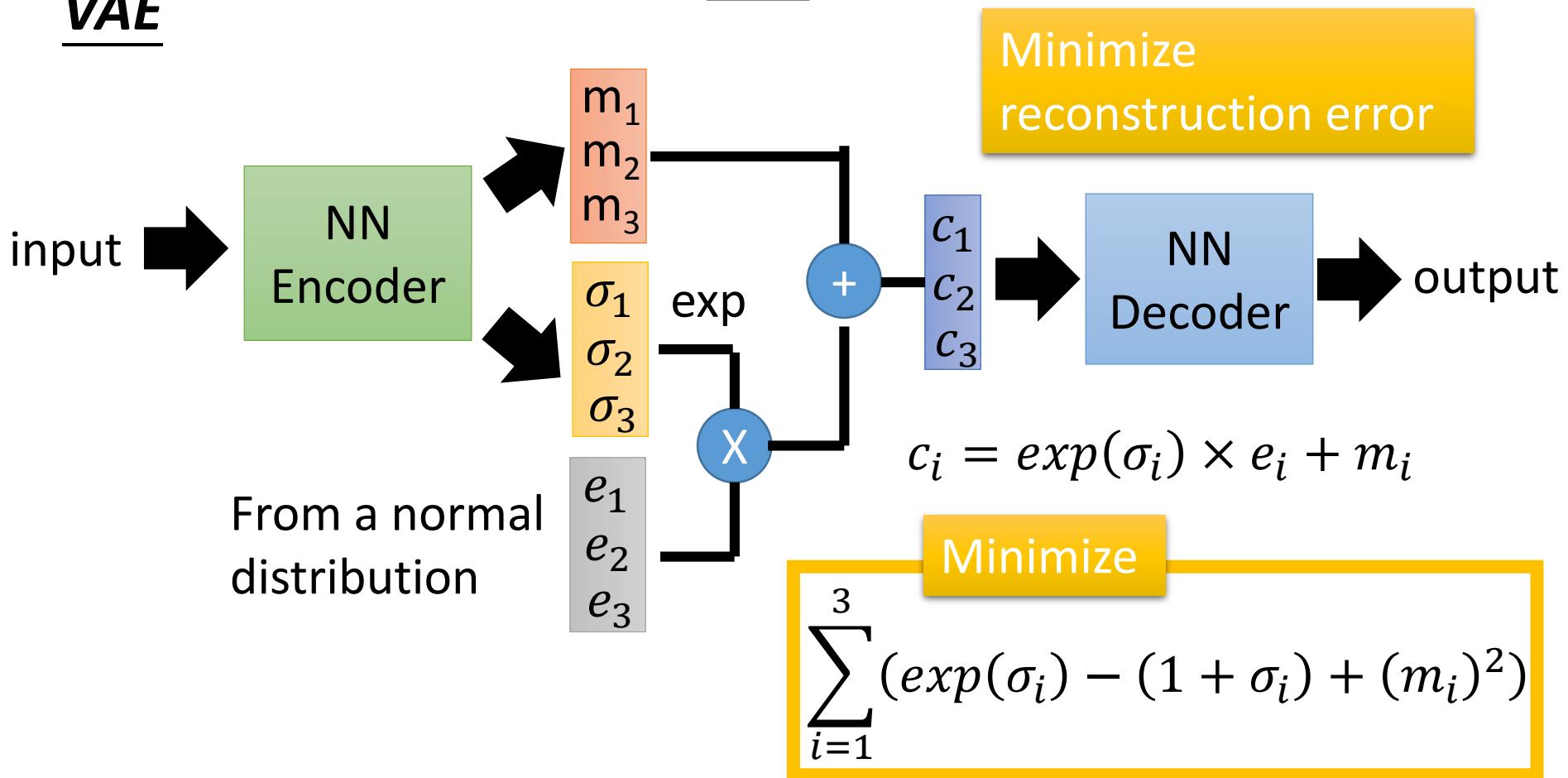
Variation Auto-encoder (VAE)

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

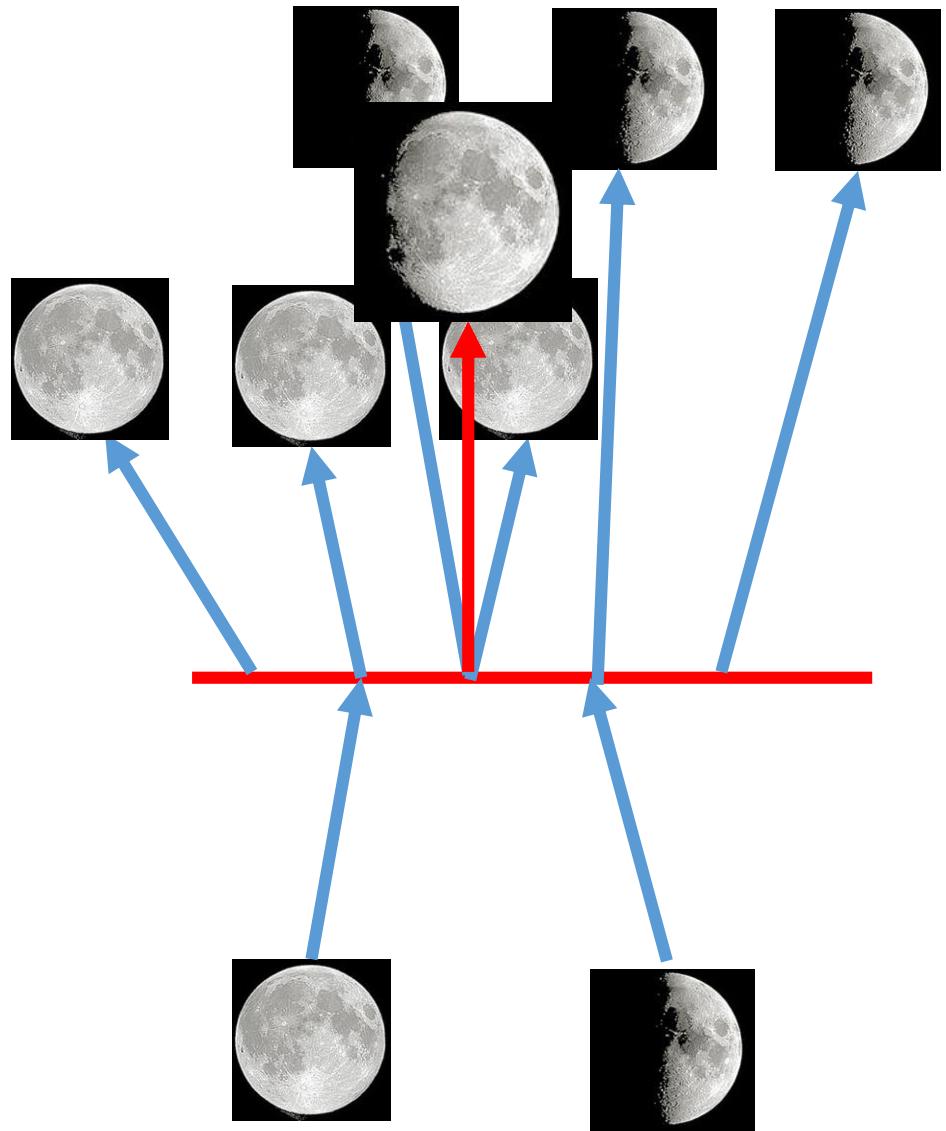
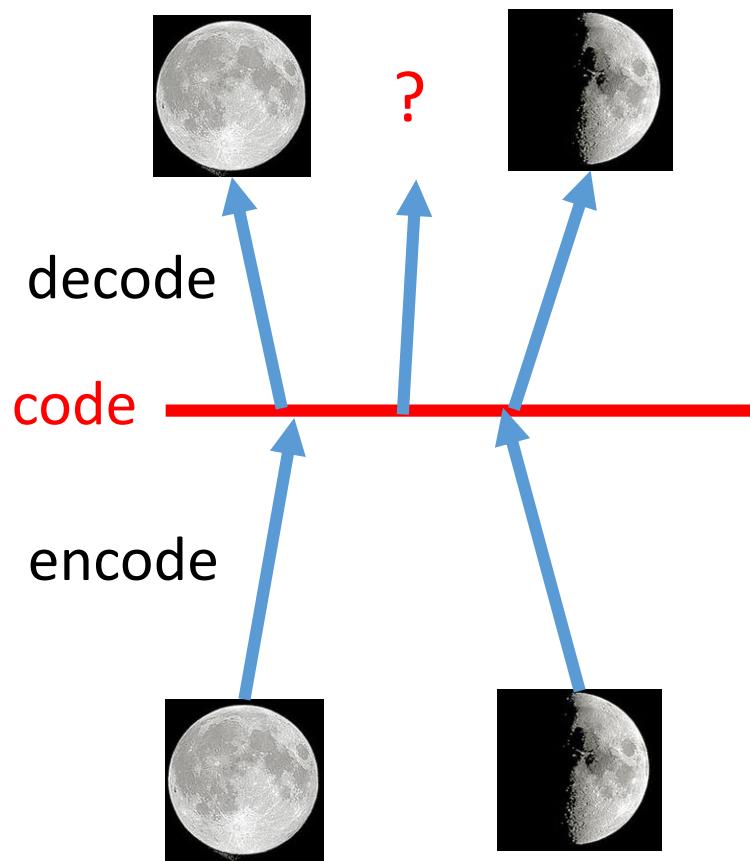
Auto-encoder



VAE



Why VAE?



VAE

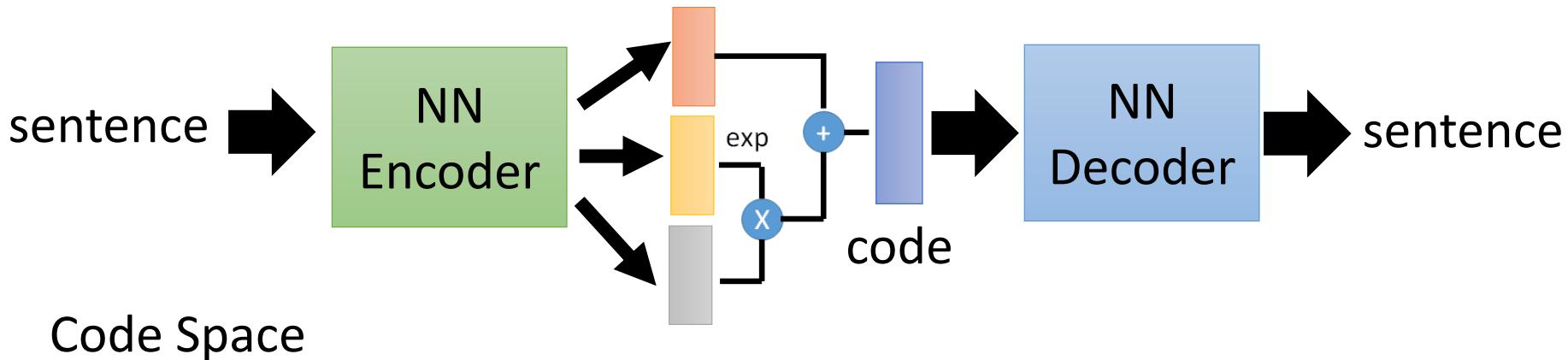
Cifar-10

<https://github.com/openai/iaf>

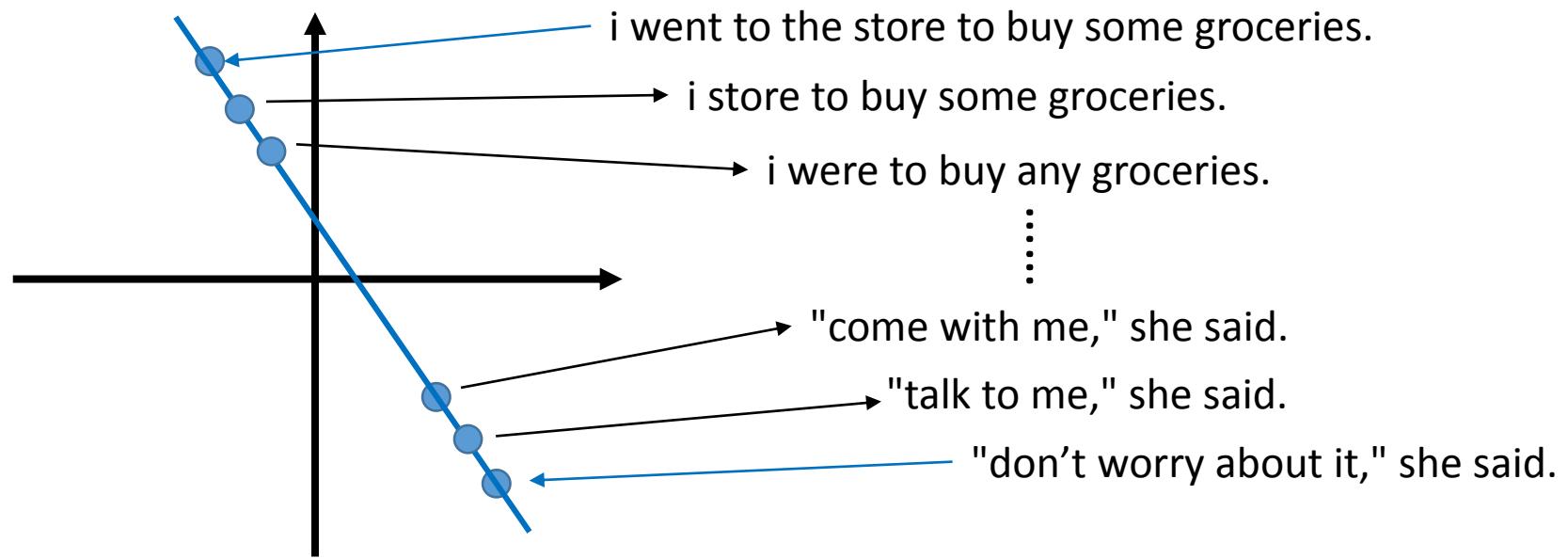


Source of image: <https://arxiv.org/pdf/1606.04934v1.pdf>

VAE - Writing Poetry



Code Space

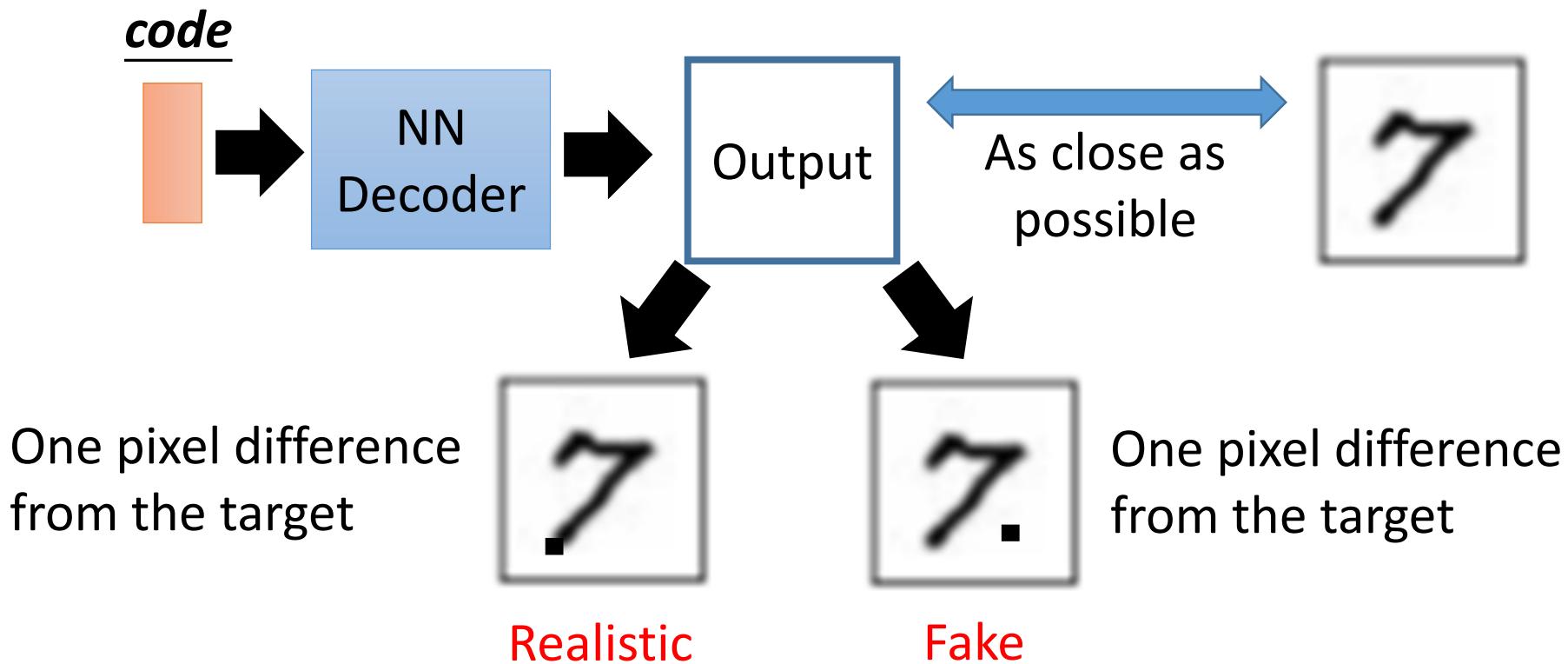


Ref: <http://www.wired.co.uk/article/google-artificial-intelligence-poetry>

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

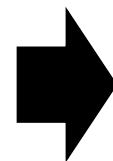
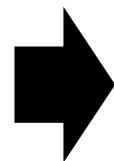
Expect more impressive results with this technique in the coming years.

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>



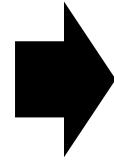
棕色

葉脈

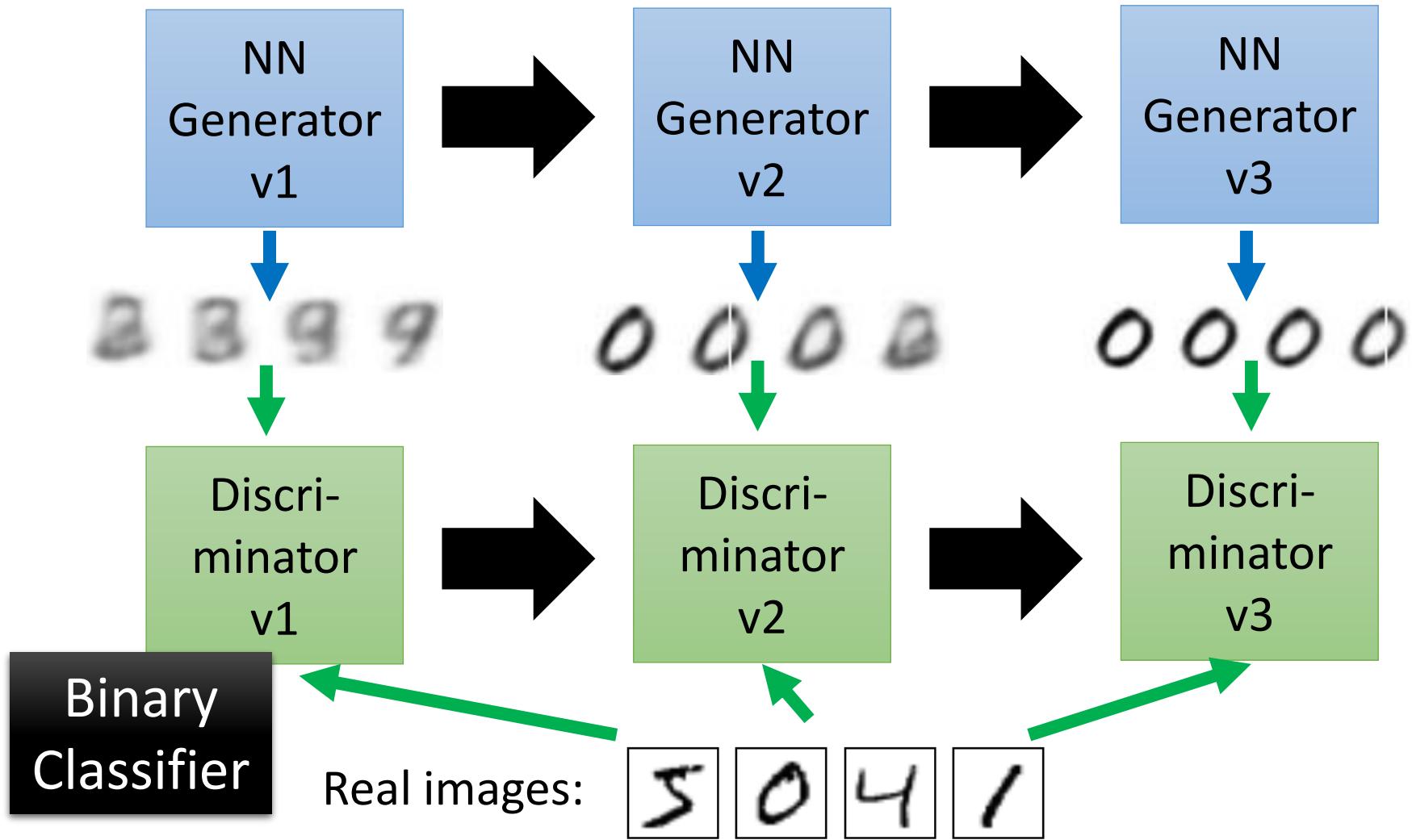
蝴蝶不是棕色

蝴蝶沒有葉脈

.....

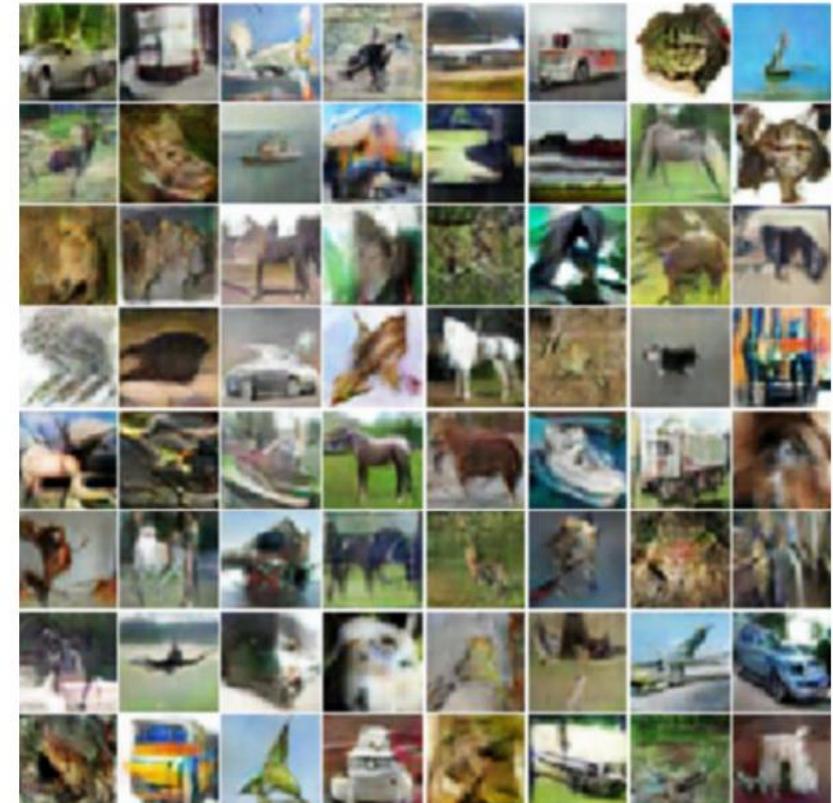
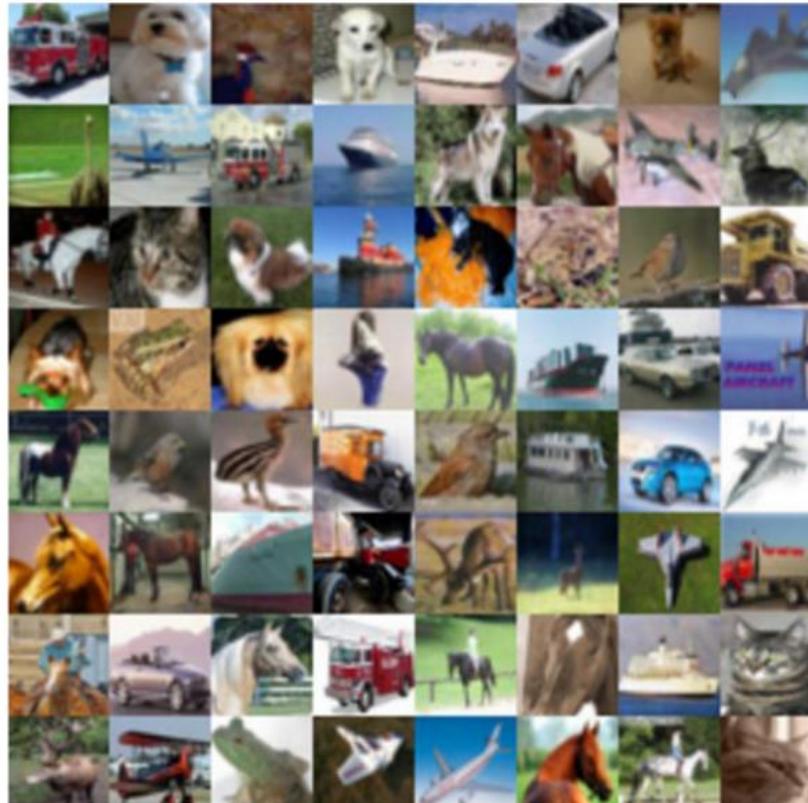


The evolution of generation



Cifar-10

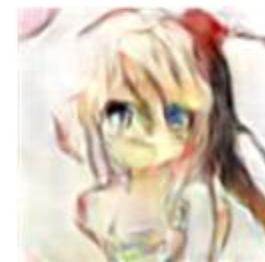
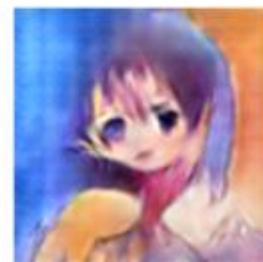
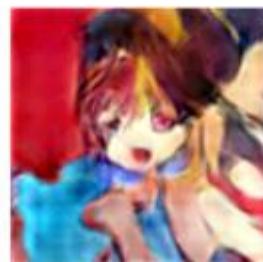
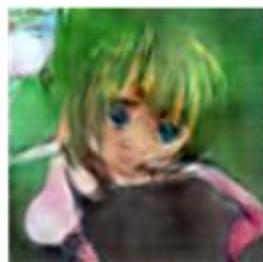
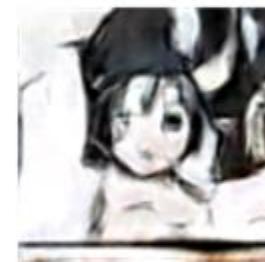
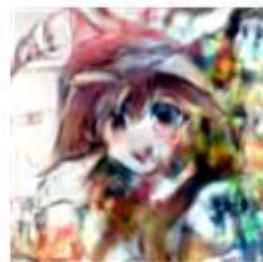
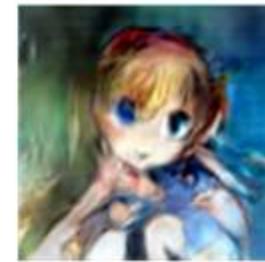
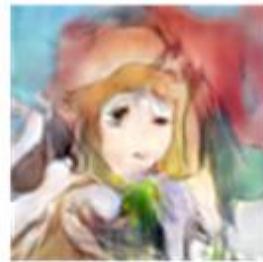
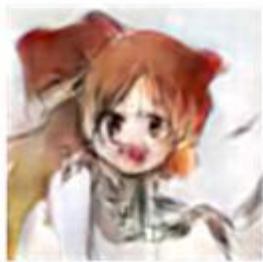
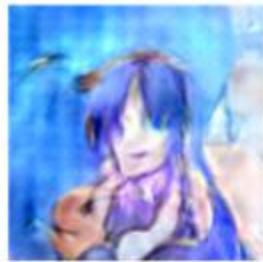
- Which one is machine-generated?



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

畫漫畫

- Ref: <https://github.com/mattyachainer-DCGAN>



漫畫

- Ref: <http://qiita.com/matty/items/e5bfe5e04b9d2f0bbd47>



元画像

-赤髪+金髪

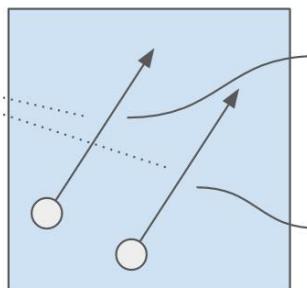
-赤目+青目

+制服+セーラー

+笑顔+口開き

+青背景

長髪化ベクトル



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

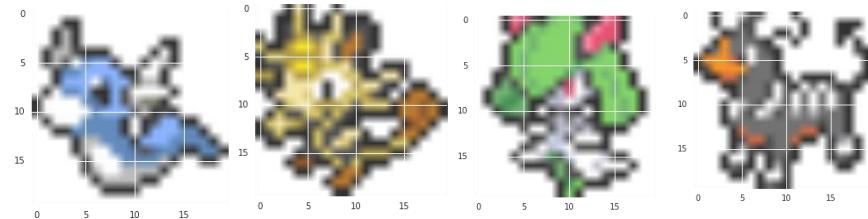
- Small images of 792 Pokémon's
 - Can machine learn to create new Pokémons?

Don't catch them! Create them!

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



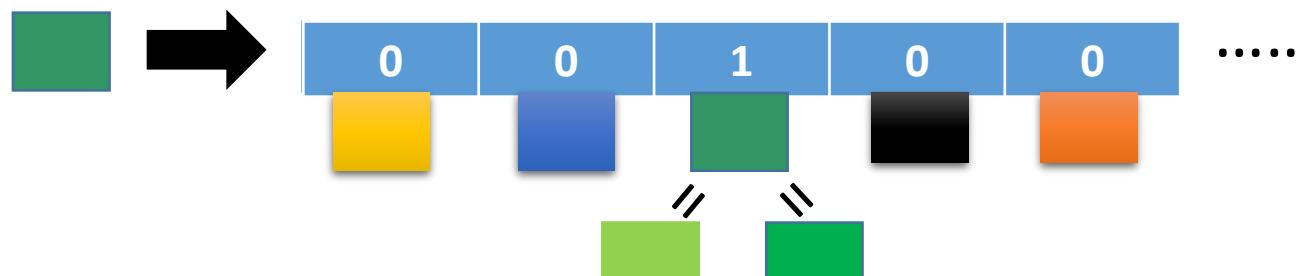
Pokémon Creation

- 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!

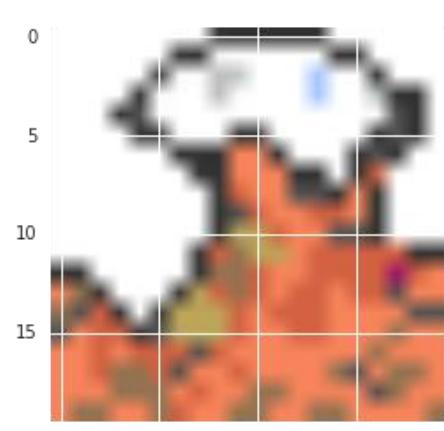
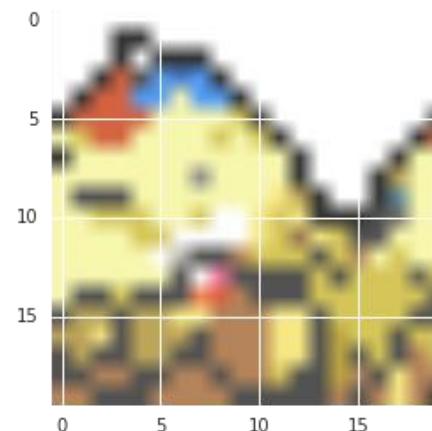
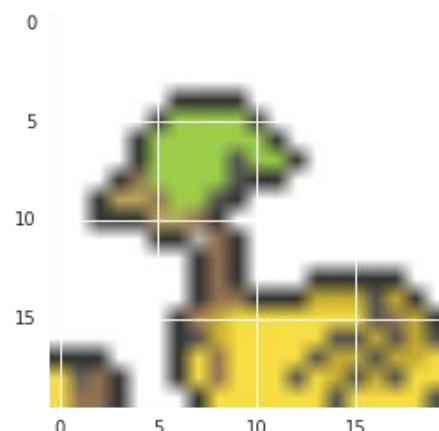
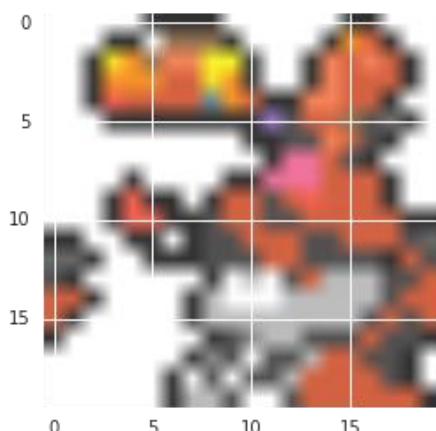
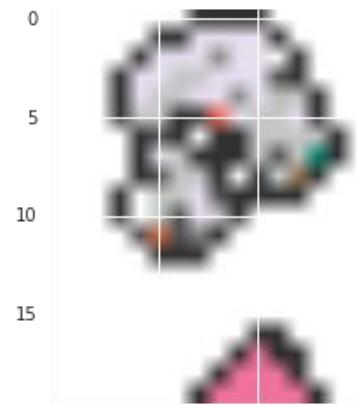
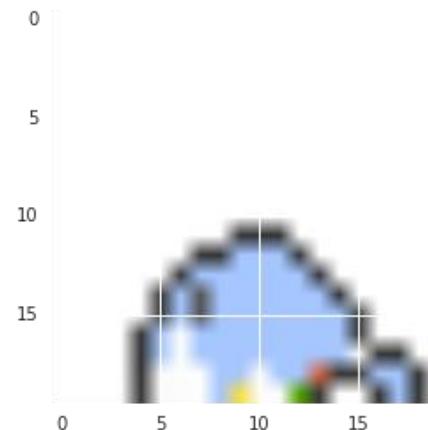
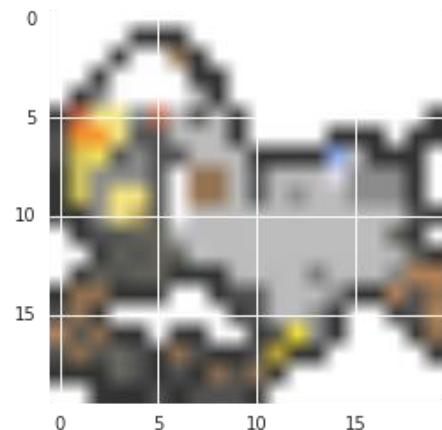
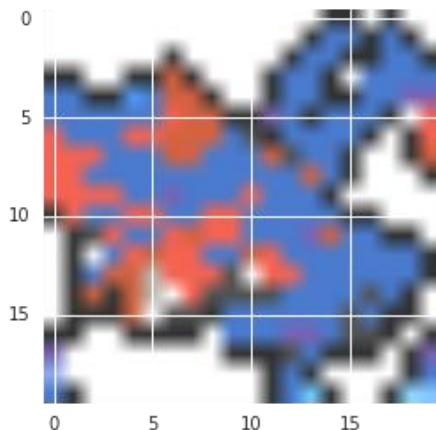
Cover 50%

It is difficult to evaluate generation.

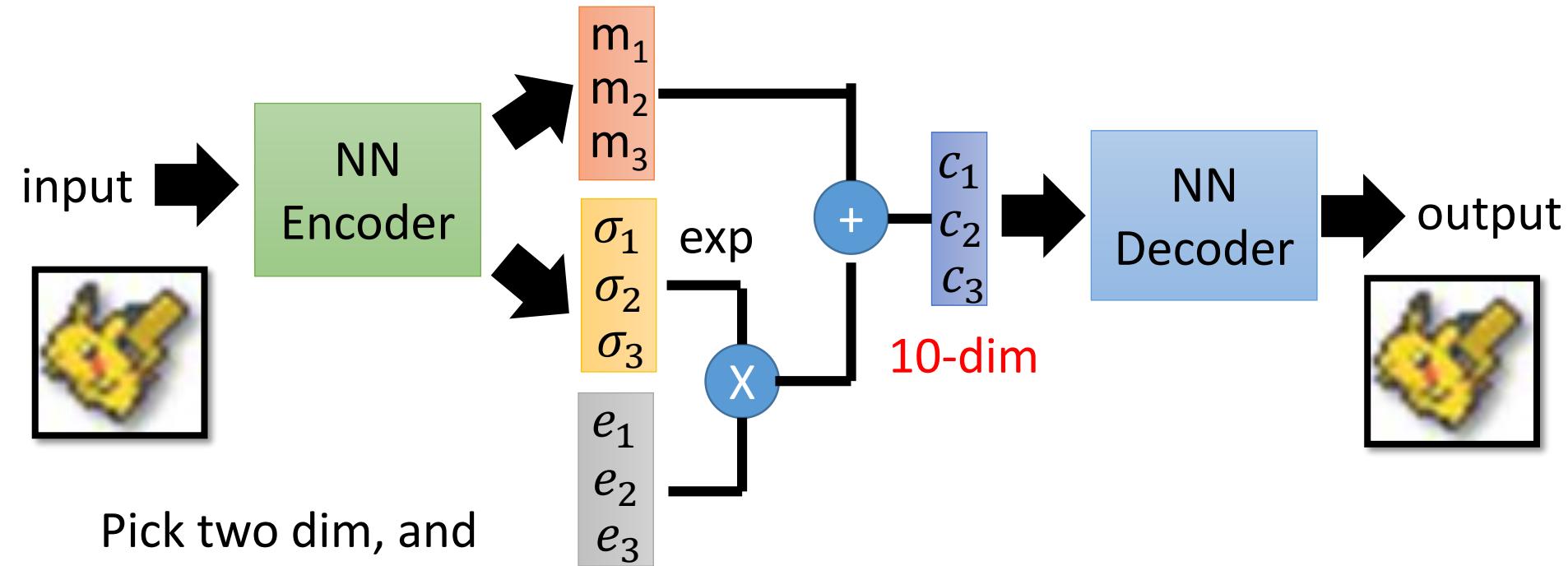
Cover 75%

Pokémon Creation

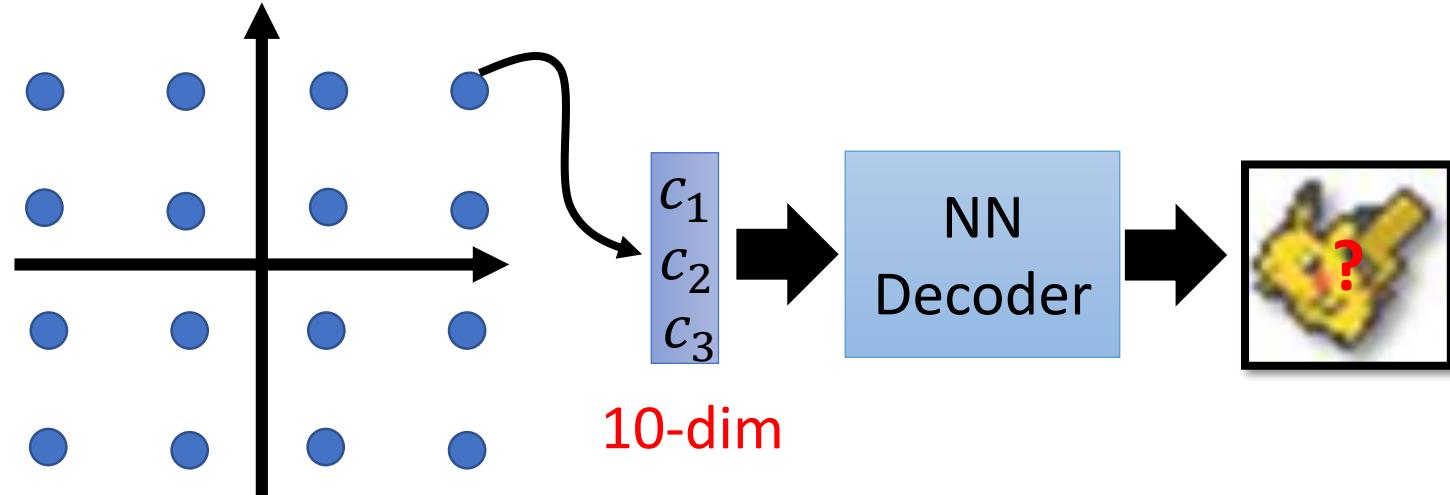
Drawing from scratch
Need some randomness



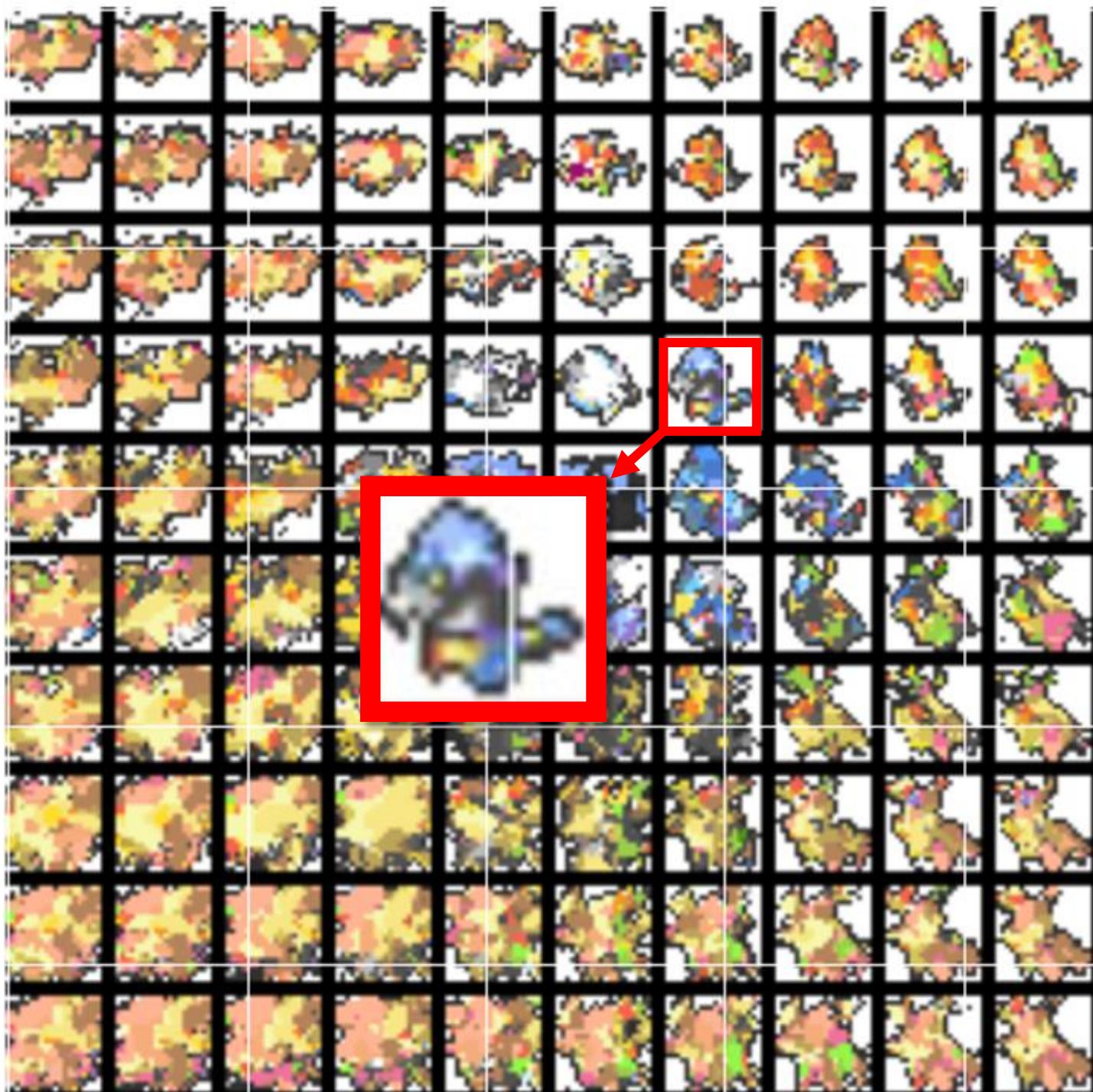
Pokémon Creation



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 19 41 34 0 0 19 0 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 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 → 255 255 255
1 → 53 53 53
2 → 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

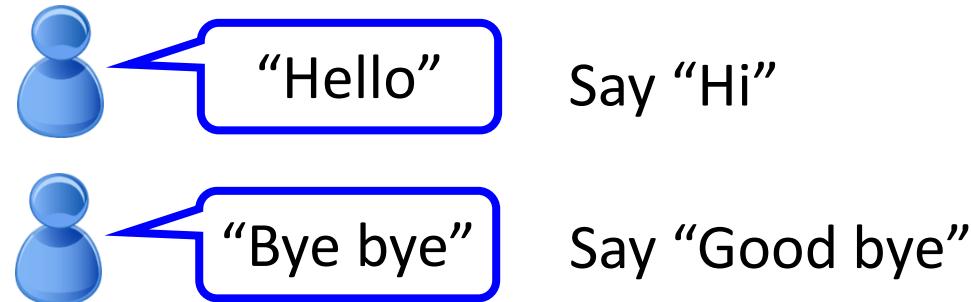
Agent learns to take actions to maximize expected reward.



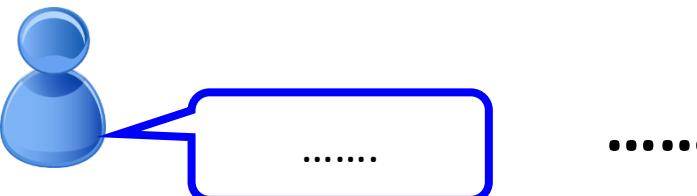
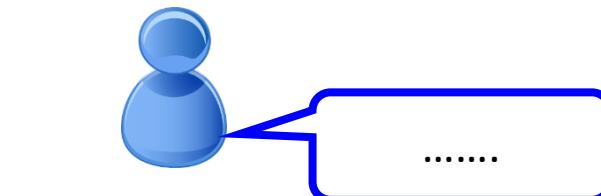
Supervised v.s. Reinforcement

- Supervised

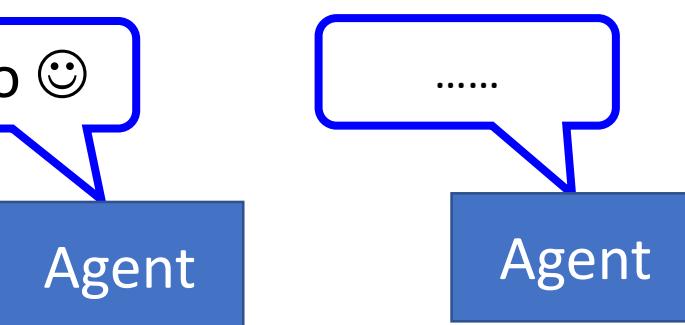
Learning from
teacher



- Reinforcement



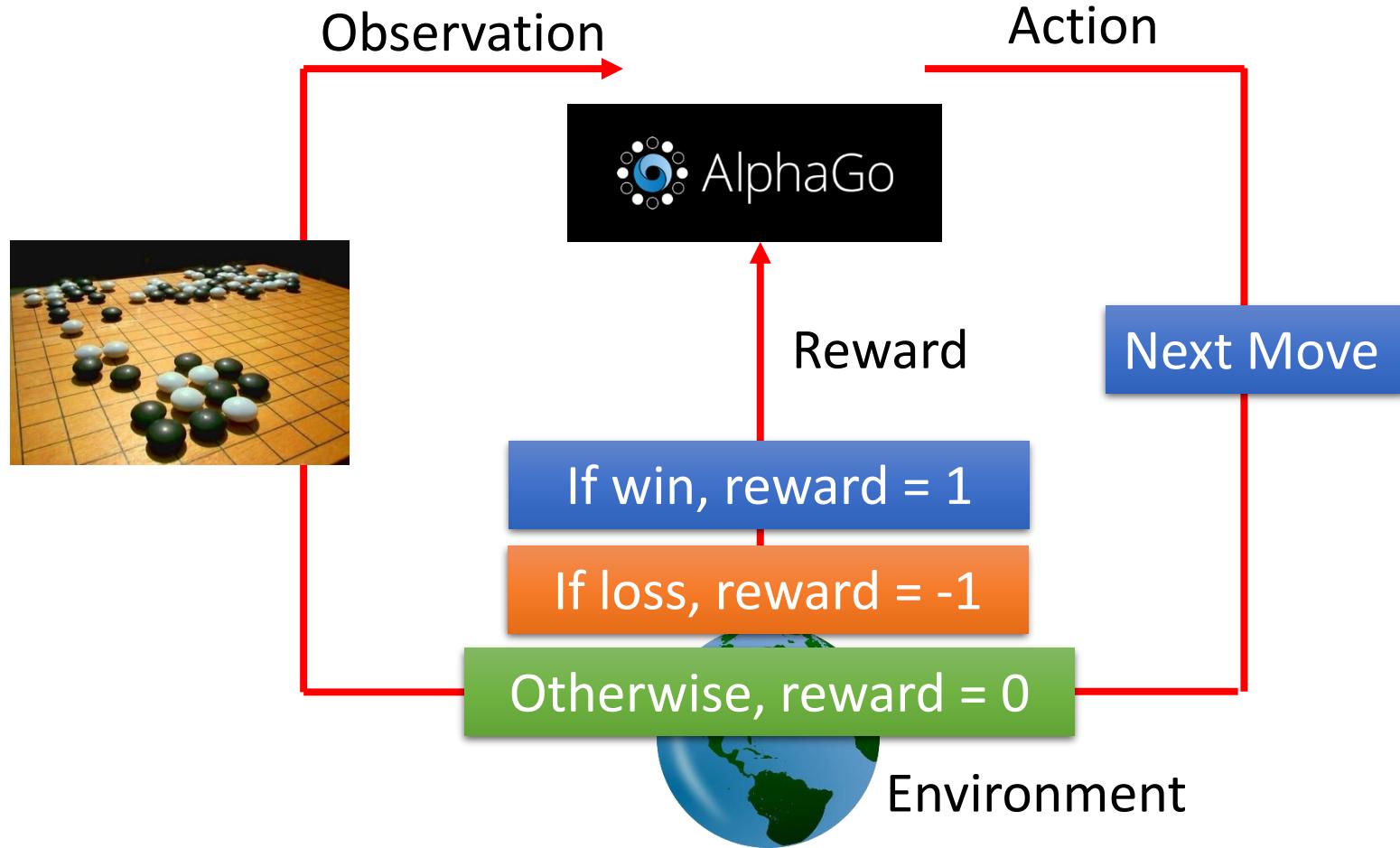
Learning from
critics



Bad

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!

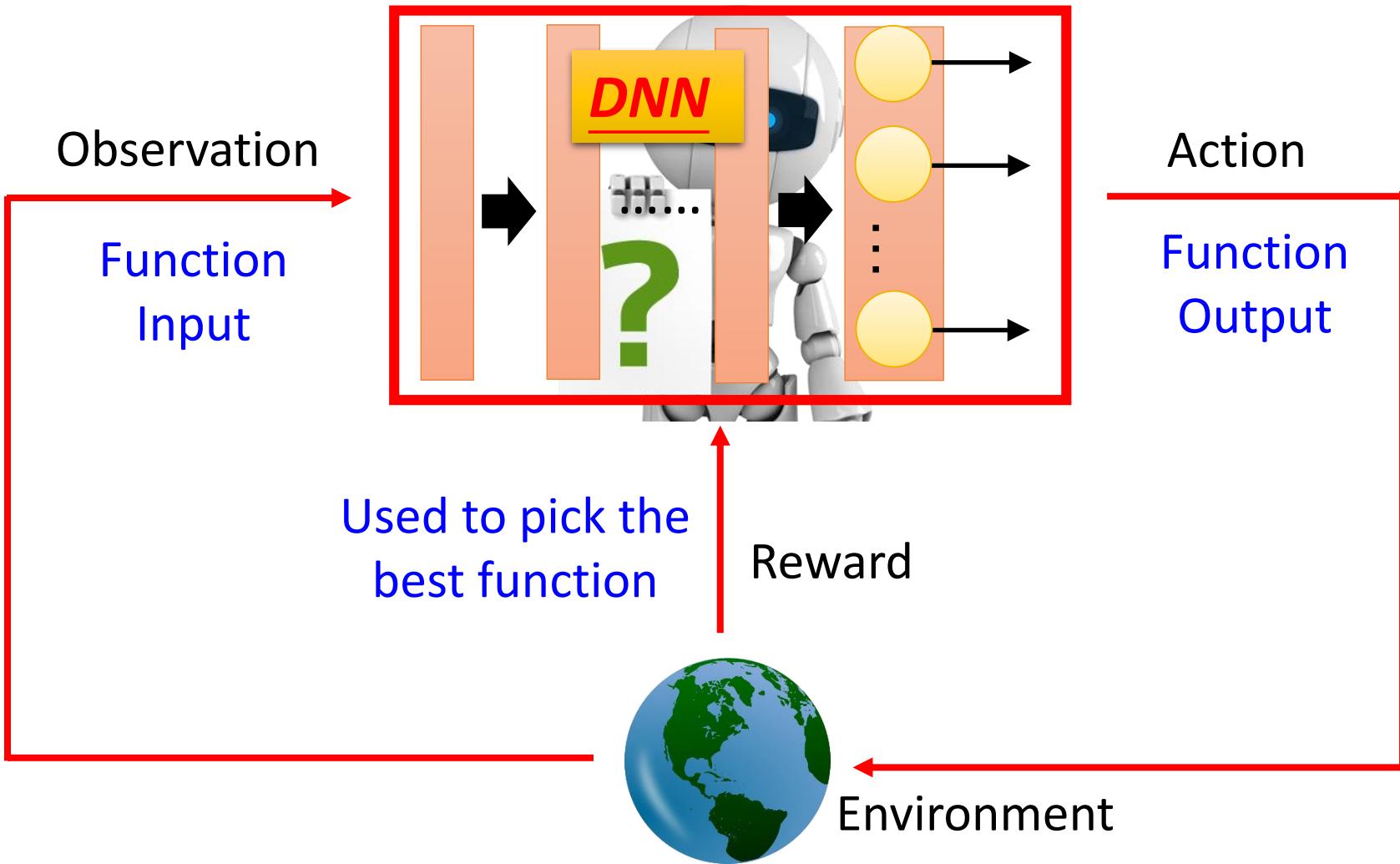
Alpha Go is supervised learning + reinforcement learning.

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



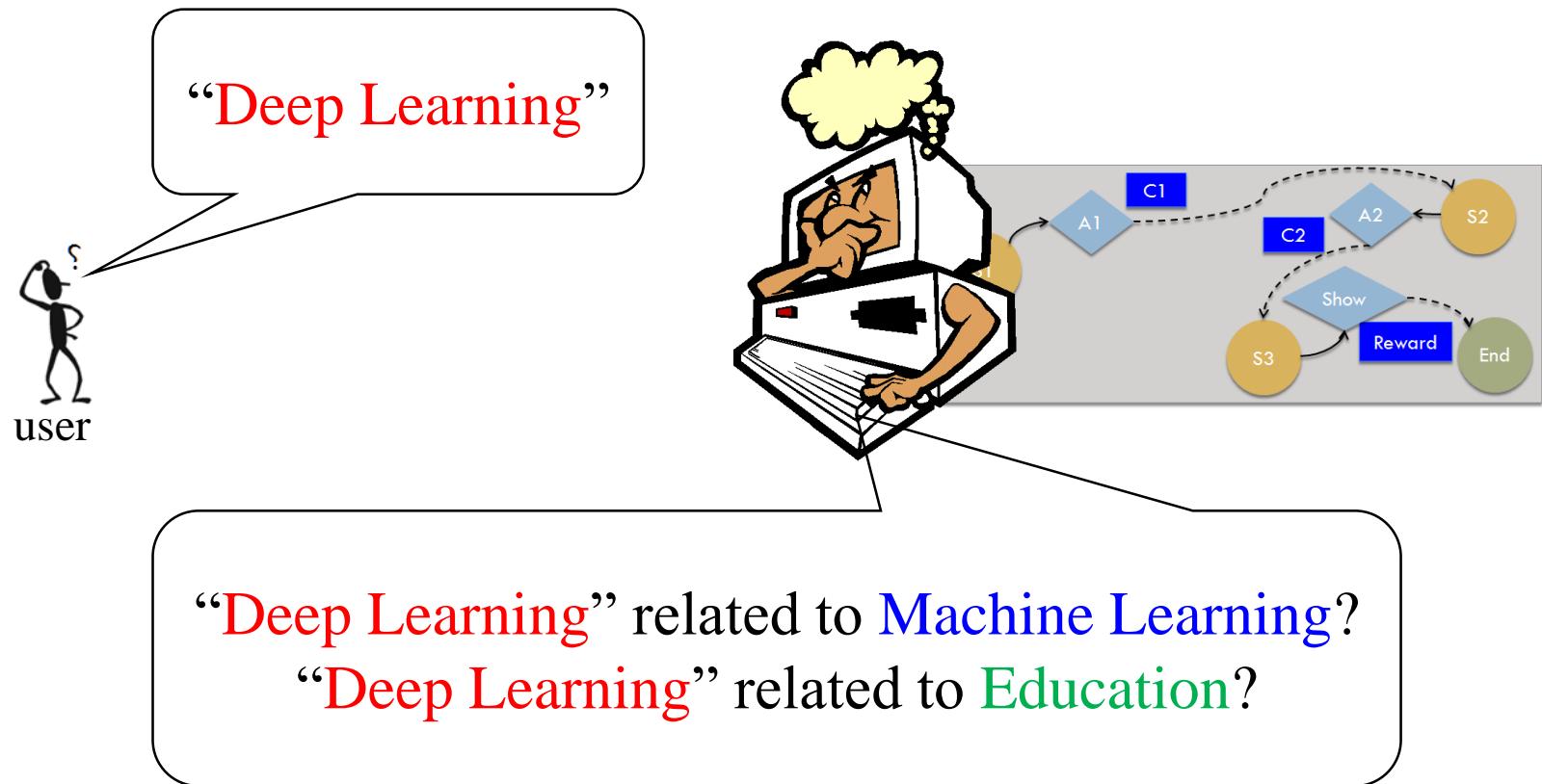
Deep Reinforcement Learning



Application: Interactive Retrieval

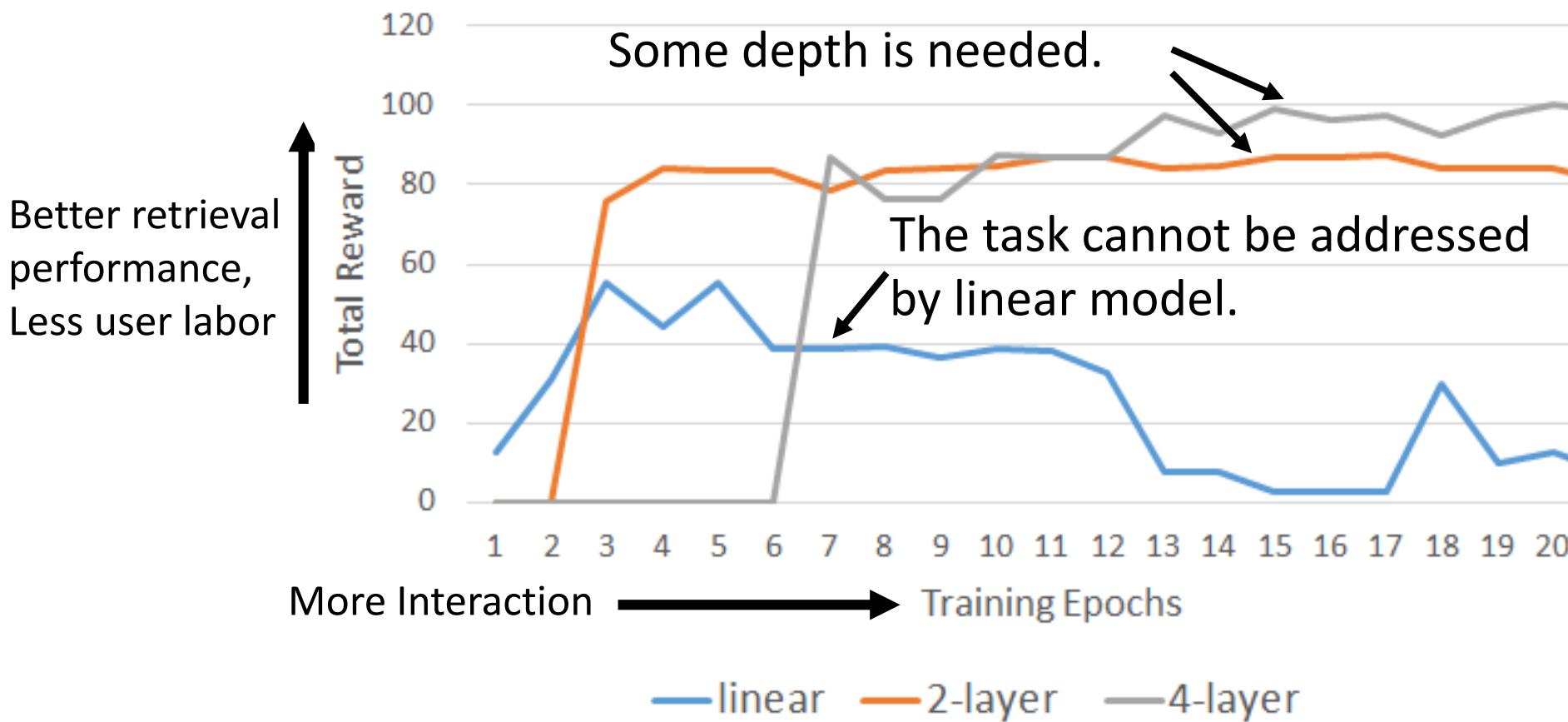
- Interactive retrieval is helpful.

[Wu & Lee, INTERSPEECH 16]



Deep 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/Taching.html>
 - 10 lectures (1:30 each)
- Deep Reinforcement Learning
 - http://videolectures.net/rldm2015_silver_reinforcement_learning/