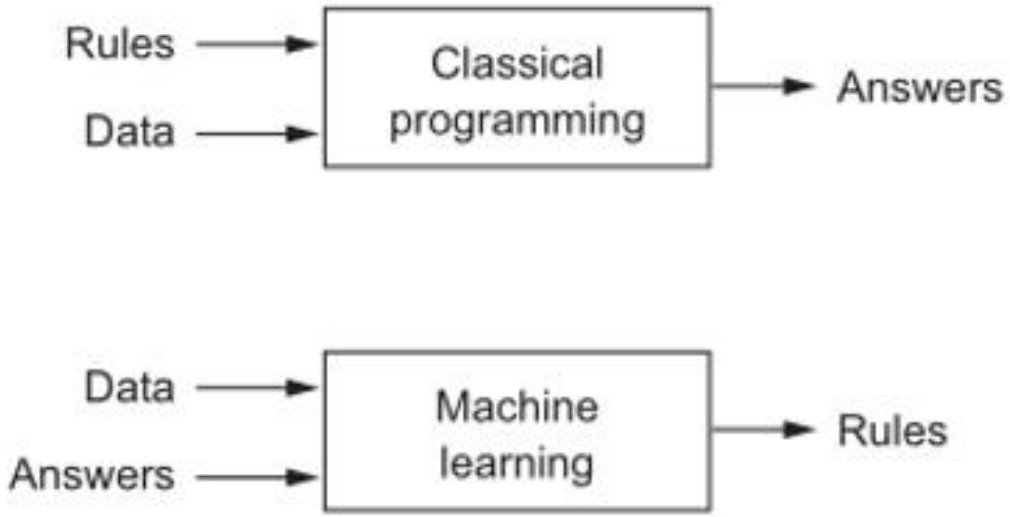
Brief Introduction of Deep Learning

2018.9

Content

Part 1 : Relationship between the Machine Learning and Deep Learning



A machine-learning system is trained rather than explicitly programmed. It's presented with many examples relevant to a task, and it finds statistical structure in these examples that eventually allows the system to come up with rules for automating the task.

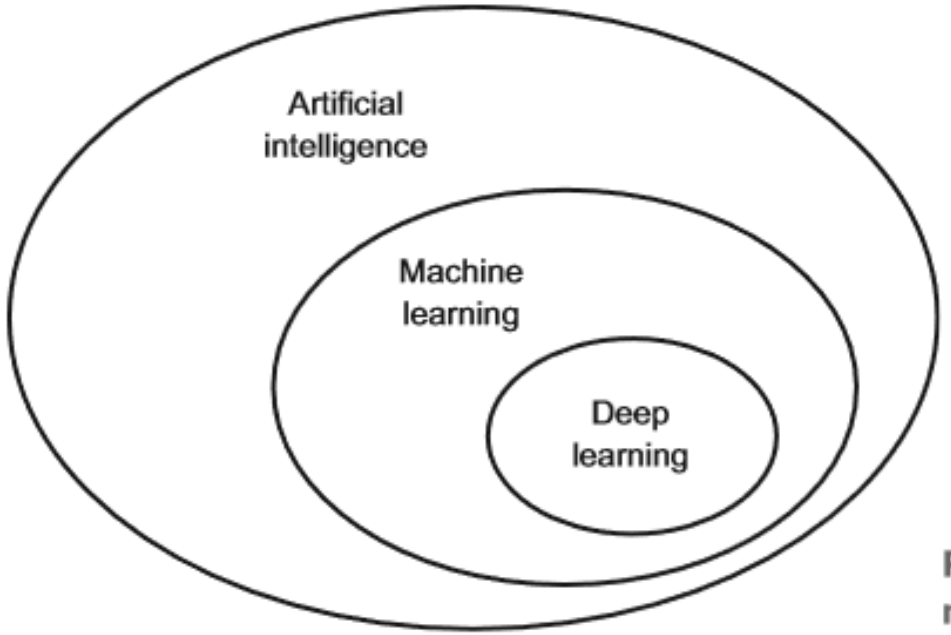
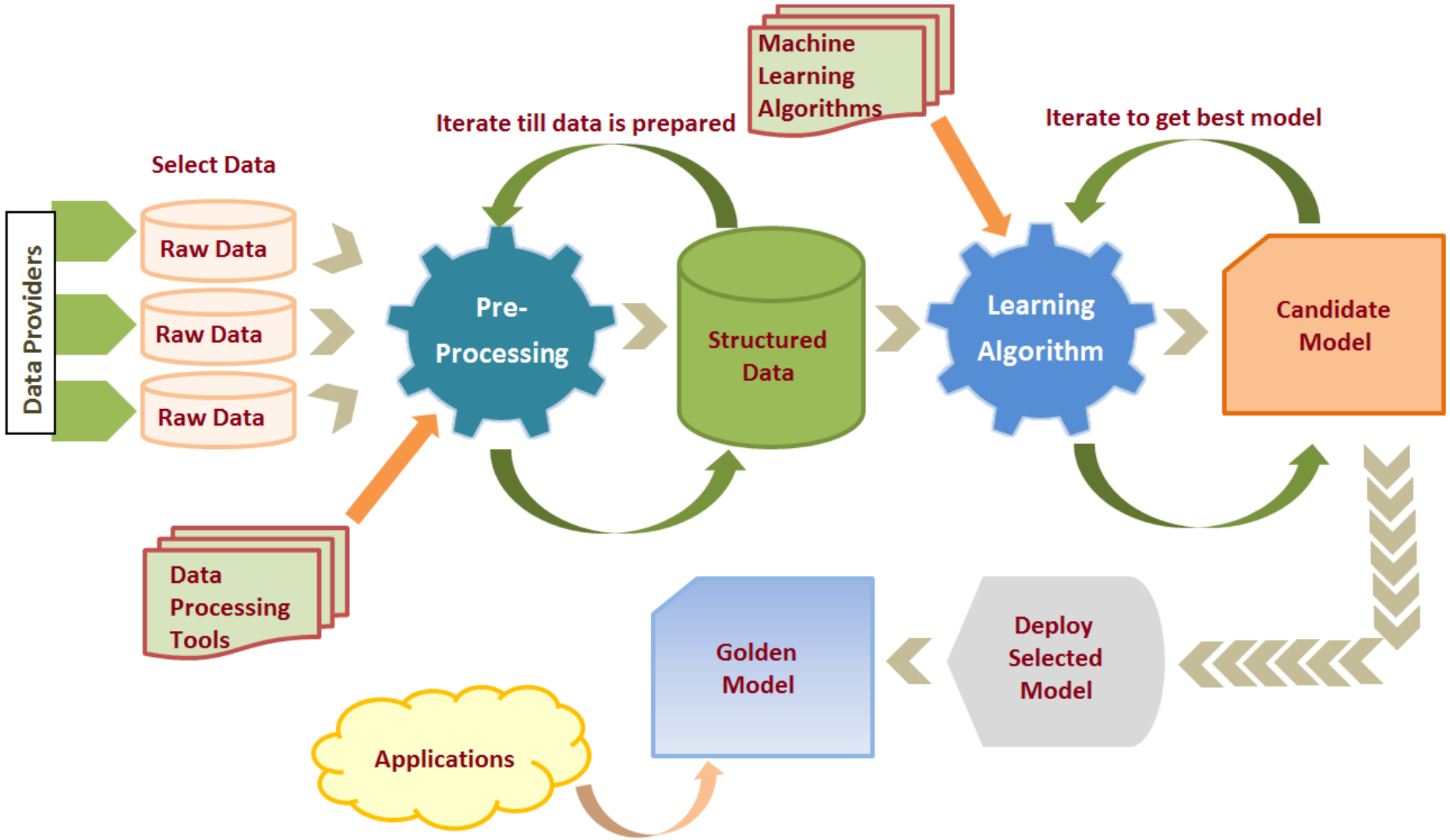


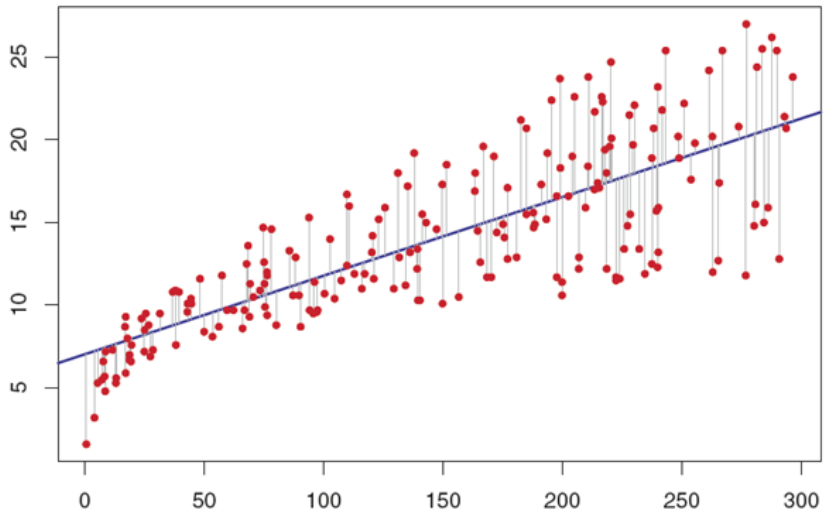
Figure 1.1 Artificial intelligence, machine learning, and deep learning

- Machine Learning specializes in how computers simulate or implement human learning behaviors to acquire new knowledge or skills and reorganize existing knowledge structures to continuously improve their performance.
- Deep learning is a branch of machine learning, an algorithm that attempts to abstract data from multiple layers using multiple processing layers that consist of complex structures or multiple nonlinear transforms.

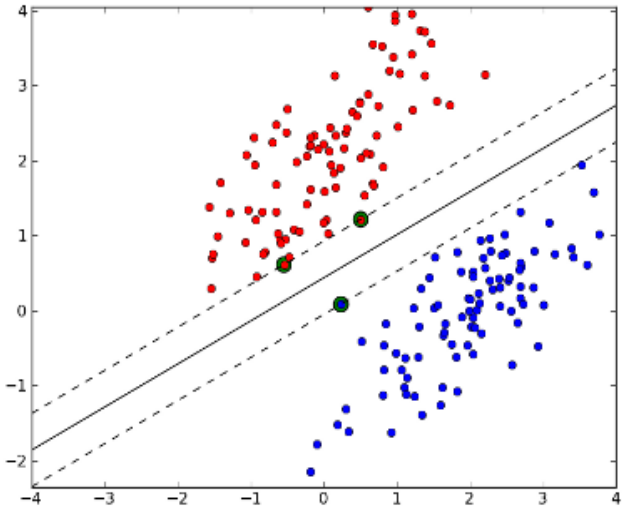
Part 2 : The workflow of Machine Learning


Machine Learning Algorithms
Iterate till data is prepared
Iterate to get best model
Select Data
Data Providers
Raw Data
Raw Data
Raw Data
Pre-Processing
Structured Data
Learning Algorithm
Candidate Model
Data Processing Tools
Golden Model
Deploy Selected Model
Applications

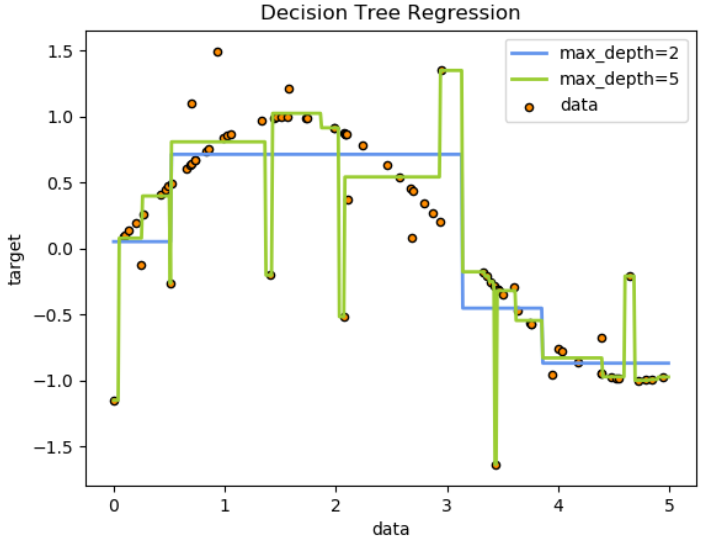
Part 3 : Basic concept of Machine Learning



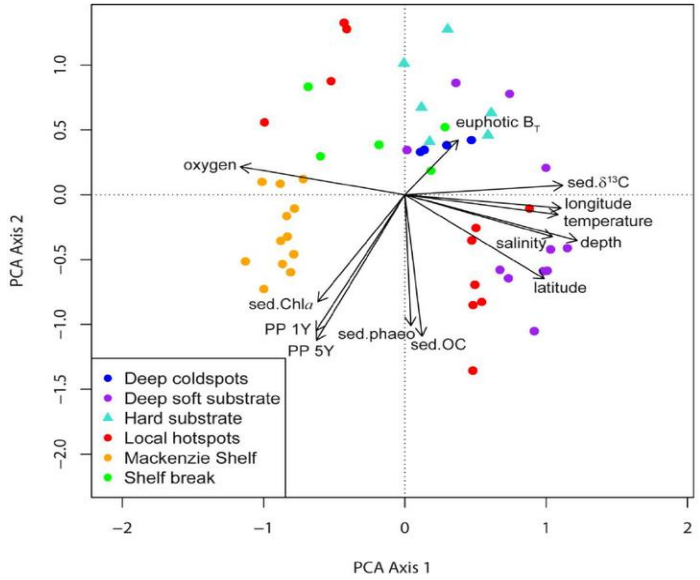
Linear Regression



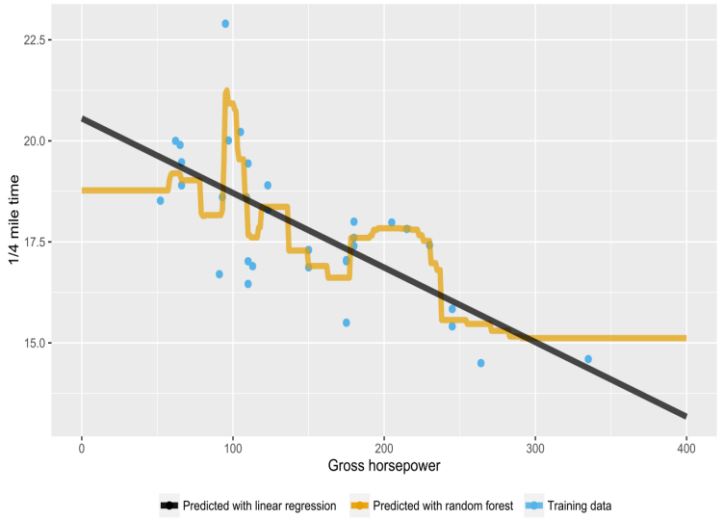
SVM



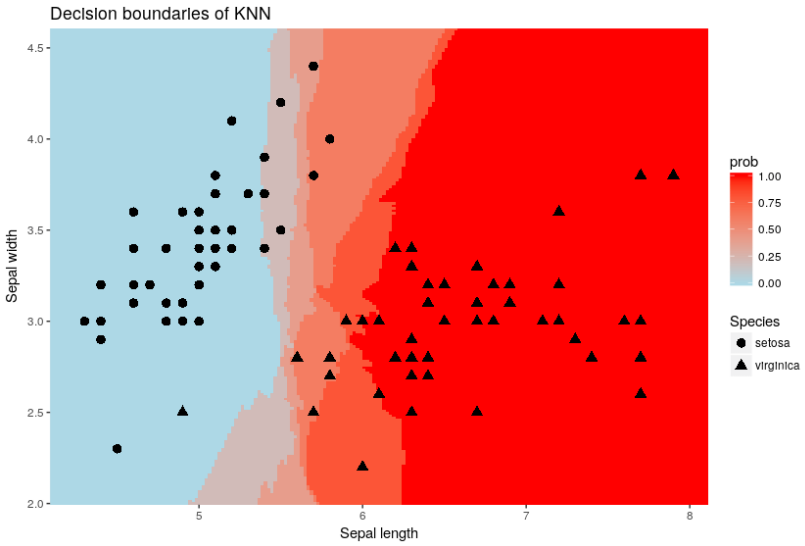
Decision Tree



PCA

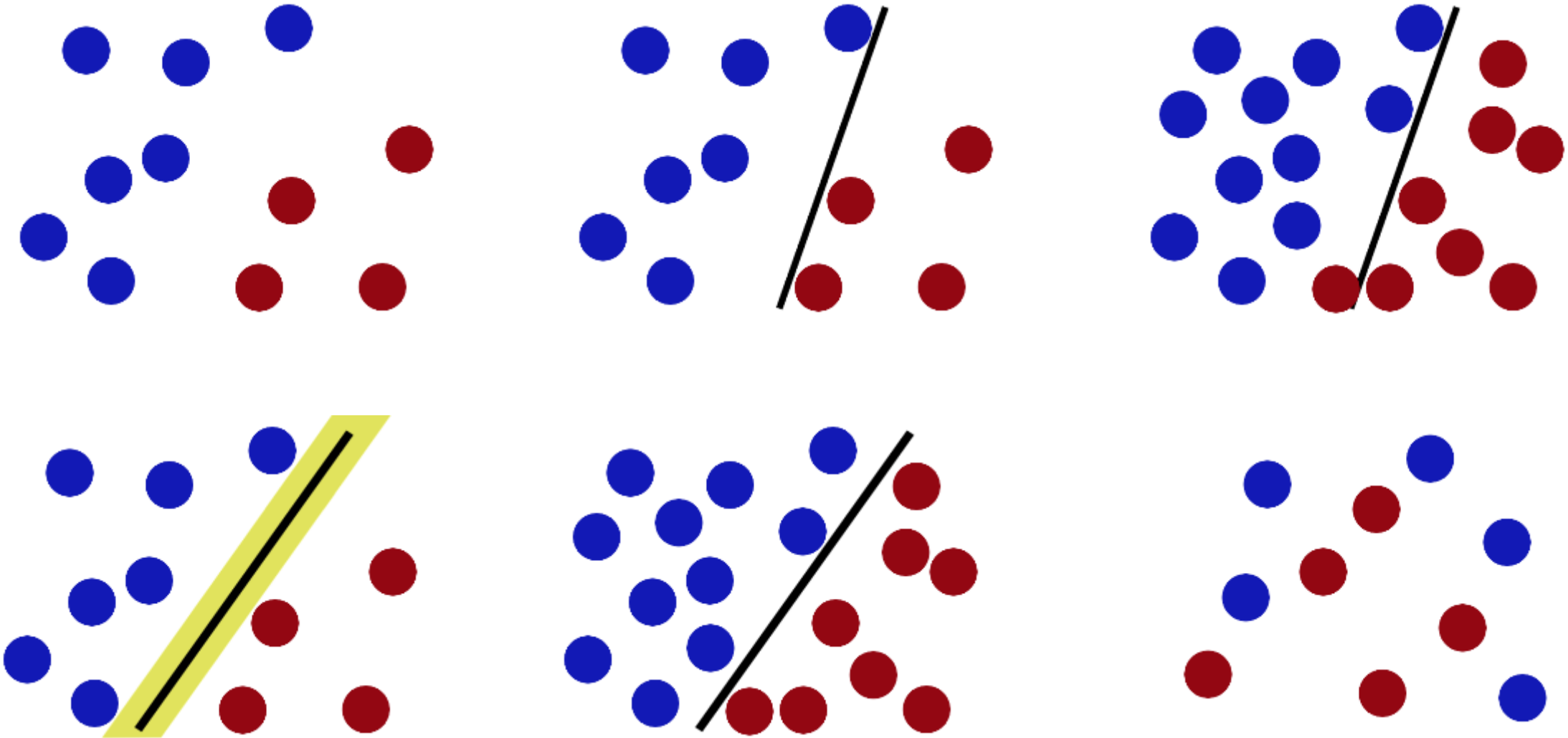


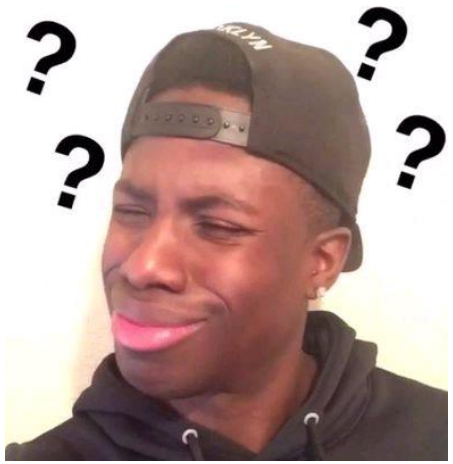
Random Forest

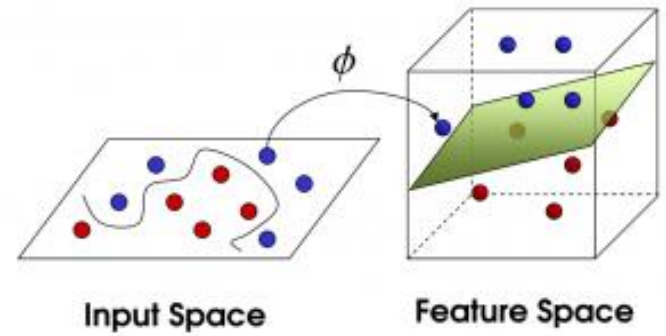


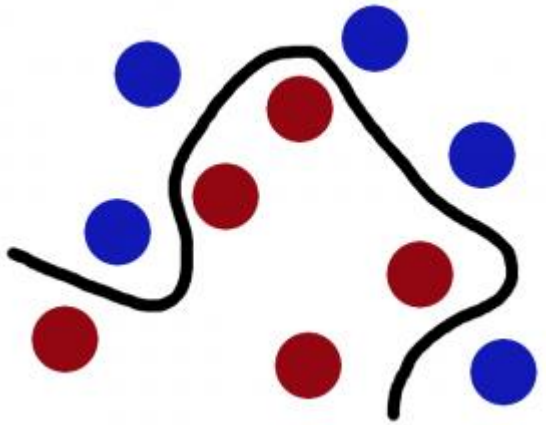
KNN

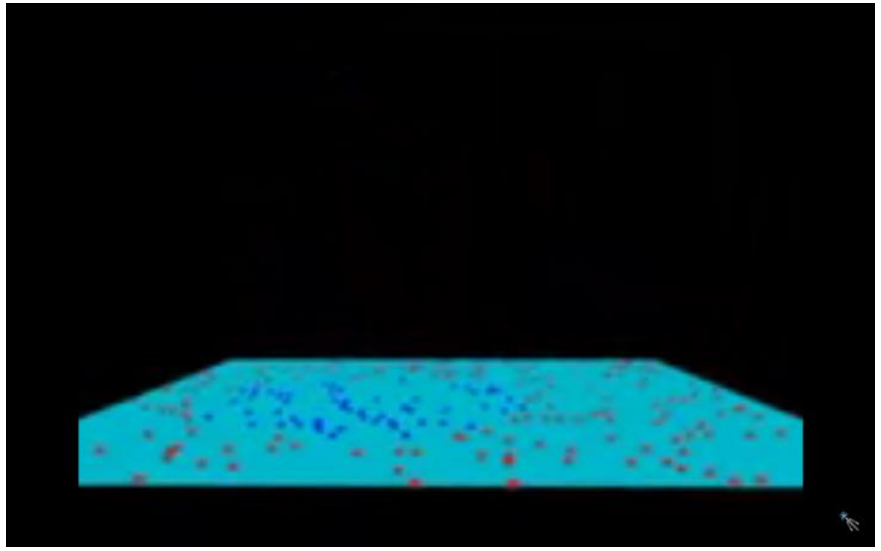
Support Vector Machine

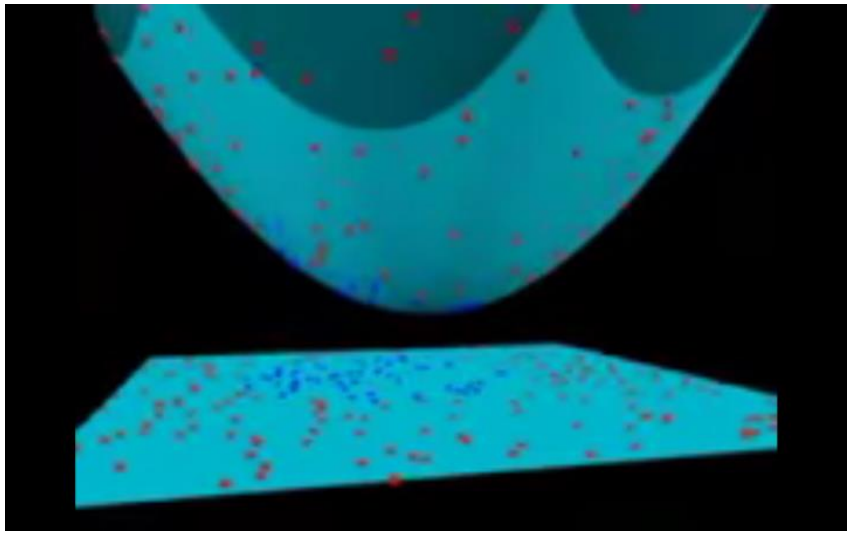


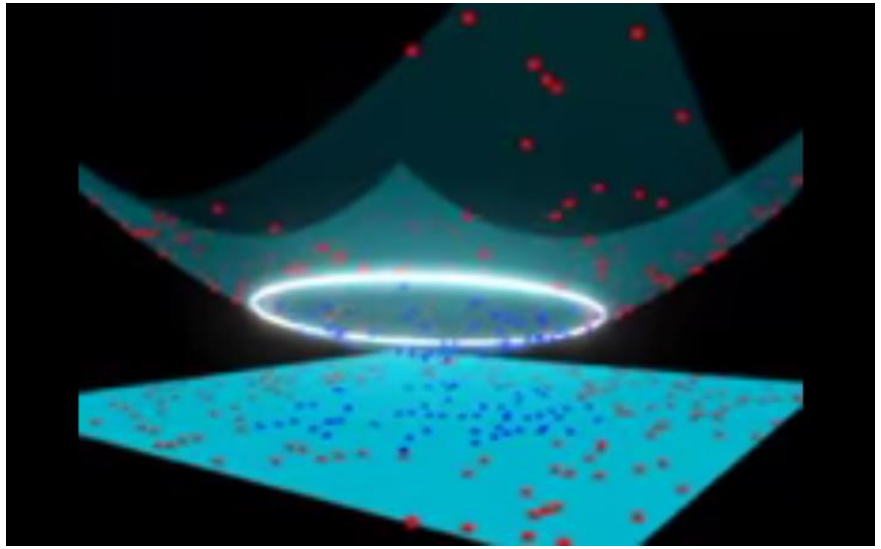


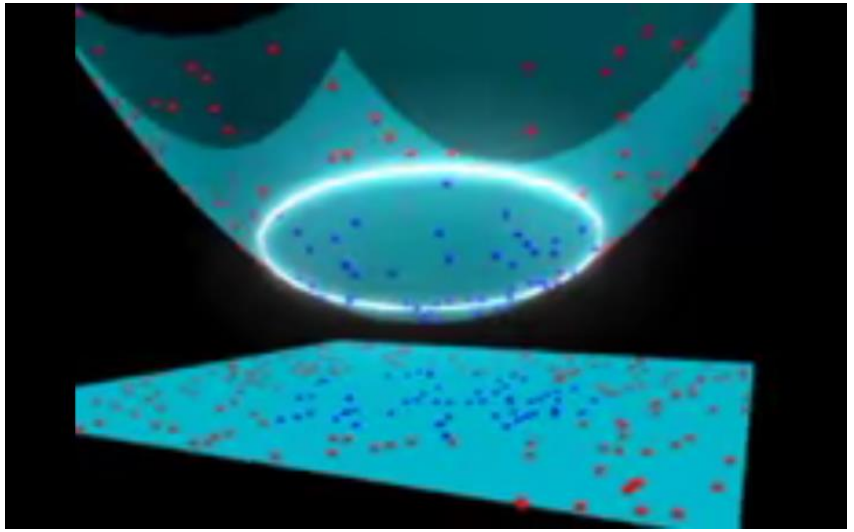

ϕ
Input Space
Feature Space











Ref: https://www.youtube.com/watch?v=3liCbRZPrZA&feature=youtu.be

Iris Data

- Iris Data Set (Iris Data Set) is the oldest data set I have ever come across. It was first introduced to linear discriminant analysis in the 1936 paper The Use of Multiple Measures in Taxonomic Problems by Ronald Fisher, a famous British statistician and biologist.

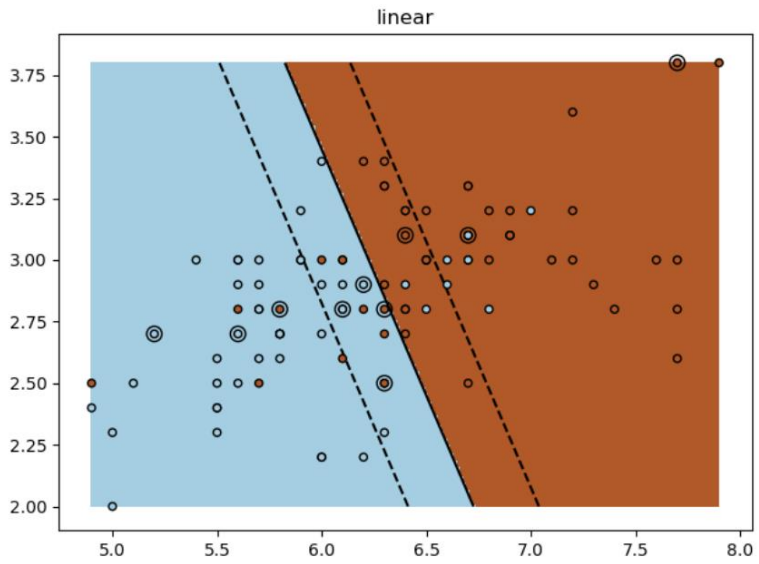
- In this data set, three different Iris species are included: Iris Setosa, Iris Versicolour, and Iris Virginica. Each category collected 50 samples, so the dataset contained 150 samples.

- The dataset measured 4 characteristics of all 150 samples with 3 labels.
  - sepal length
  - sepal width
  - petal length
  - petal width

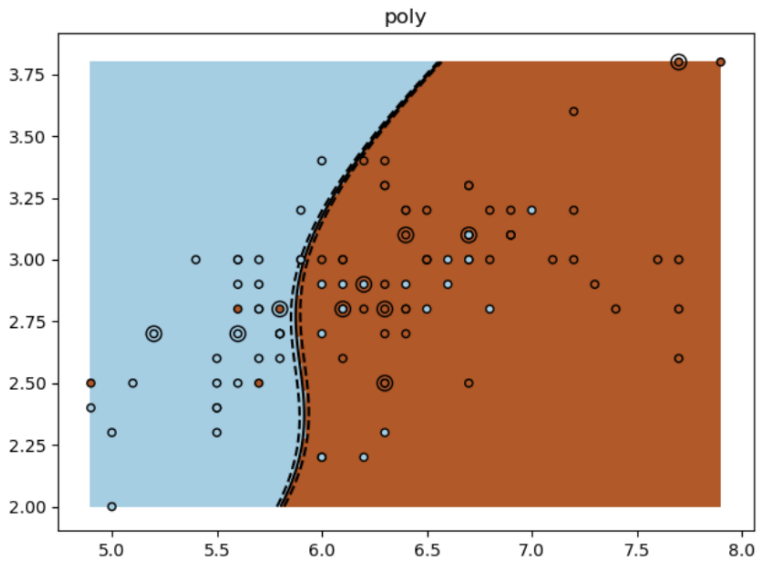
  - Iris - Setosa
  - Iris – Versicolour
  - Iris - Virginica

Ref: https://archive.ics.uci.edu/ml/datasets/Iris/

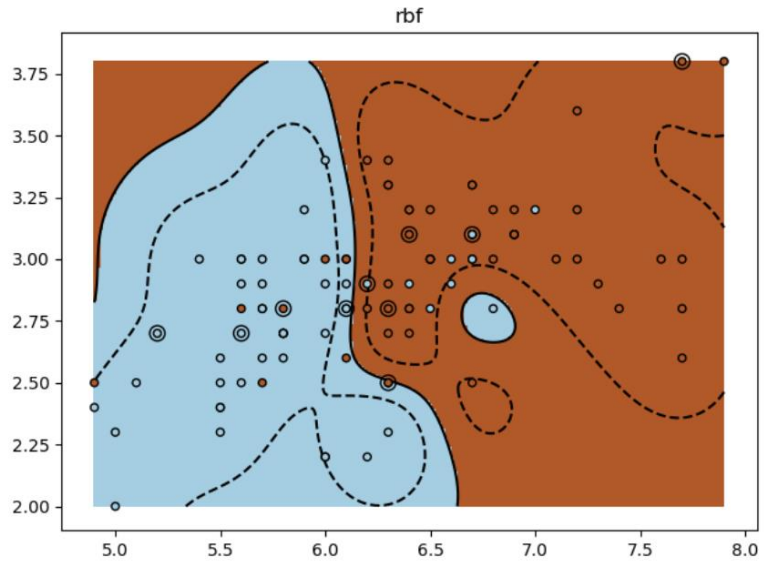
SVM kernels



Linear Kernel

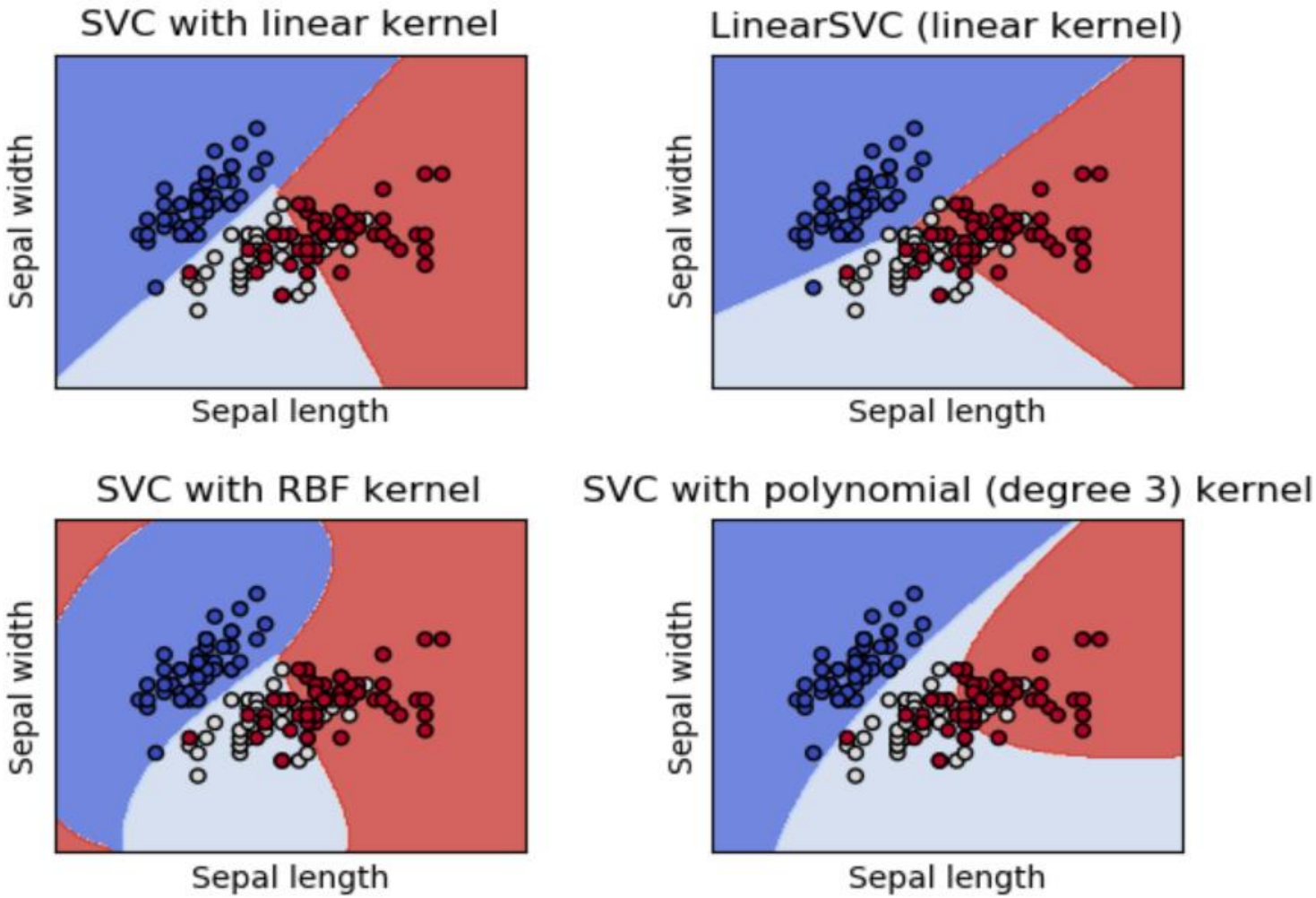


Poly kernel



Rbf Kernel

SVM-Classification



SVM-Regression

- Modeling species' geographic distributions is an important problem in conservation biology. In this example we model the geographic distribution of two south american mammals given past observations and 14 environmental variables.

- The two species are:
  - "Bradypus variegatus" , the Brown-throated Sloth.
  - "Microryzomys minutus" , also known as the Forest Small Rice Rat, a rodent that lives in Peru, Colombia, Ecuador, Peru, and Venezuela.

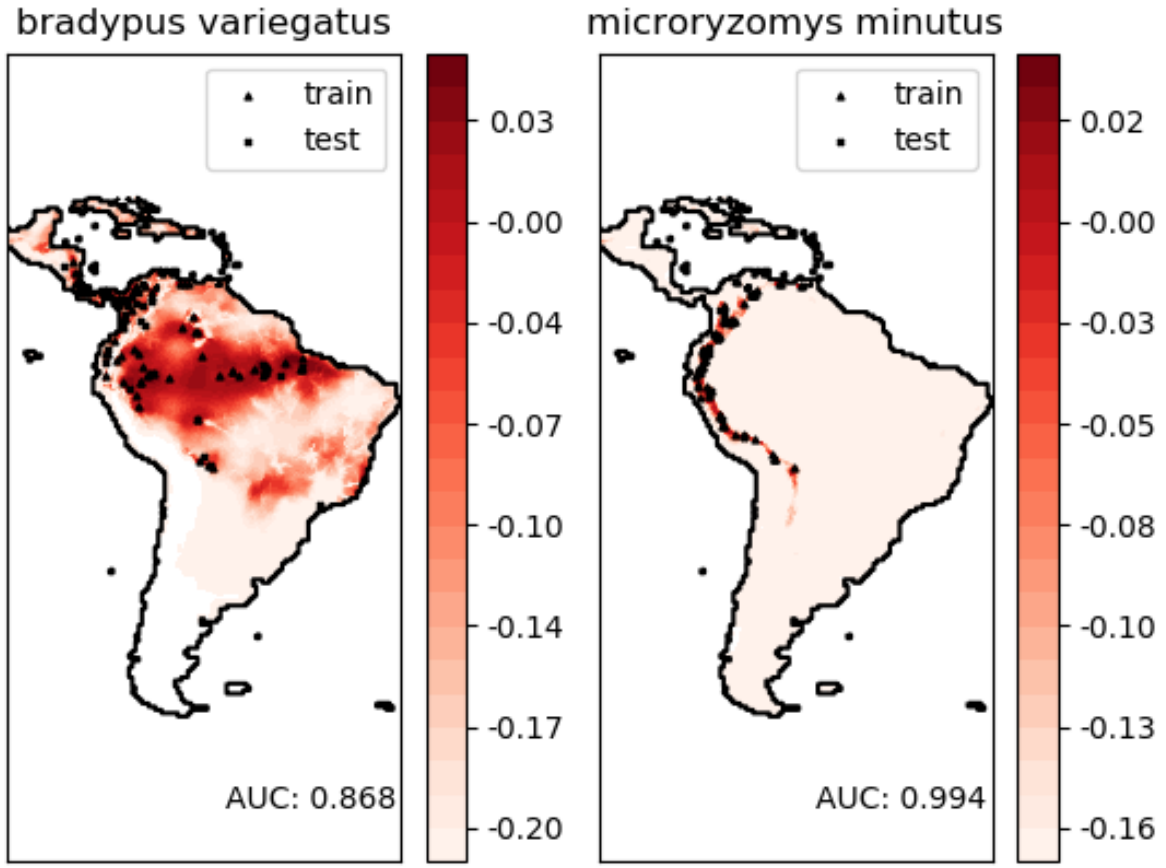
Ref: "Maximum entropy modeling of species geographic distributions" S. J. Phillips, R. P. Anderson, R. E. Schapire - Ecological Modelling, 190:231-259, 2006.

Who are they?

Bradypus variegatus

Microryzomys minutus

SVM-Regression



Ref: "Maximum entropy modeling of species geographic distributions" S. J. Phillips, R. P. Anderson, R. E. Schapire - Ecological Modelling, 190:231-259, 2006.

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection.

The advantages of support vector machines are:

•Effective in high dimensional spaces.

•Still effective in cases where number of dimensions is greater than the number of samples.

•Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.

•Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

The disadvantages of support vector machines include:

•If the number of features is much greater than the number of samples, avoid over-fitting in choosing Kernel functions and regularization term is crucial.

•SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation (see Scores and probabilities, below).

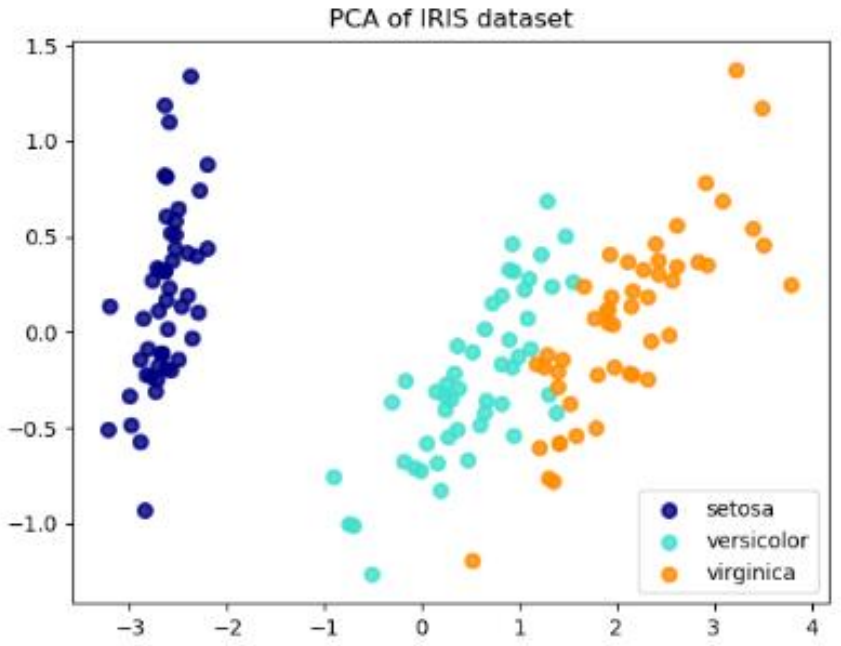
Dimension Reduction

- The Iris dataset represents 3 kind of Iris flowers (Setosa, Versicolour and Virginica) with 4 attributes: sepal length, sepal width, petal length and petal width.
- We human beings could not see the data more than 3 dimensions.
- So, how do we make data visualization with data more than 3 dimensions?
- We reduce the dimension of the data.

Dimension Reduction

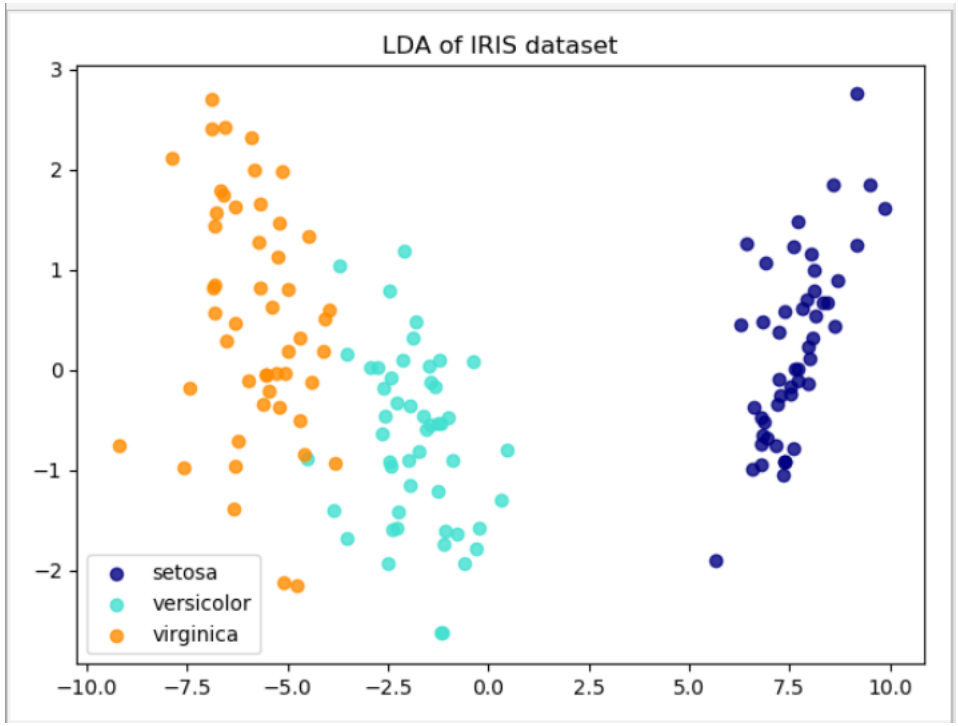
- Principal Component Analysis (PCA) applied to this data identifies the combination of attributes (principal components, or directions in the feature space) that account for the most variance in the data.
- Here we plot the different samples on the 2 first principal components.

```
[-5.67401294  1.66134615]
[-5.19712883 -0.36550576]
[-4.98171163  0.81297282]
[-5.90148603  2.32075134]
[-4.68400868  0.32508073]]
explained variance ratio (first two components): [0.92461621 0.05301557]
```



Dimension Reduction

- Linear Discriminant Analysis (LDA) tries to identify attributes that account for the most variance between classes. In particular, LDA, in contrast to PCA, is a supervised method, using known class labels.
- Here we plot the different samples on the 2 first principal components.



Part 4 : Basic concept of Deep Learning
&
A Crash Course of Neural Network

Three Steps for Deep Learning

Step 1: Network Structure

Step 2: Cost Function

Step 3: Optimization

based on training data

Three Steps for Deep Learning

Step 1. A neural network is a function composed of simple functions (neurons)

- Usually we design the network structure, and let machine find parameters from data

Step 2. Cost function evaluates how good a set of parameters is

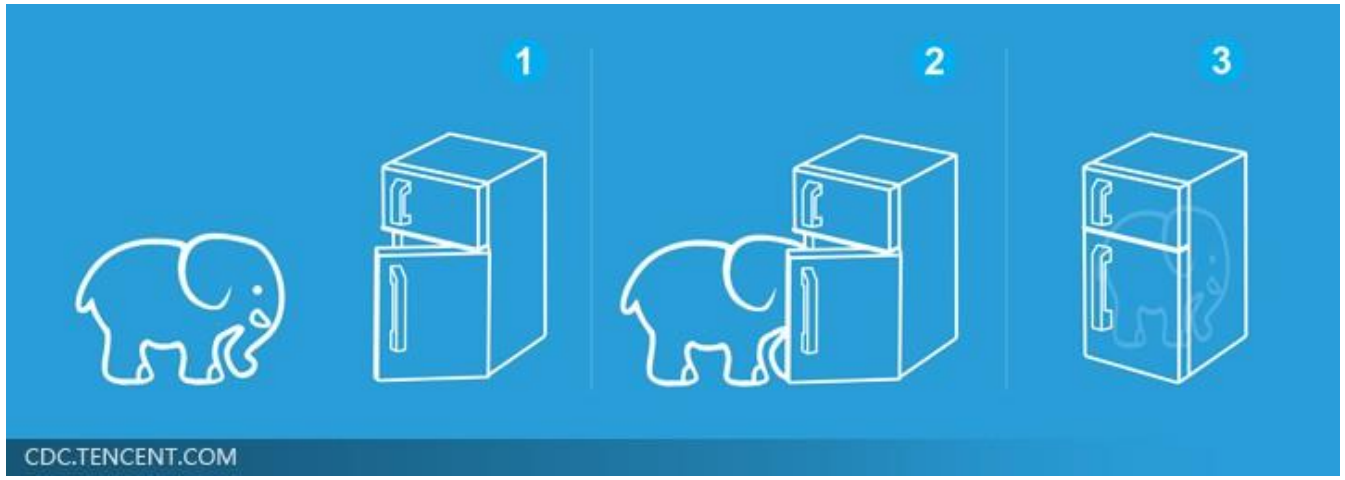
- We design the cost function based on the task

Step 3. Find the best function set (e.g. gradient descent)

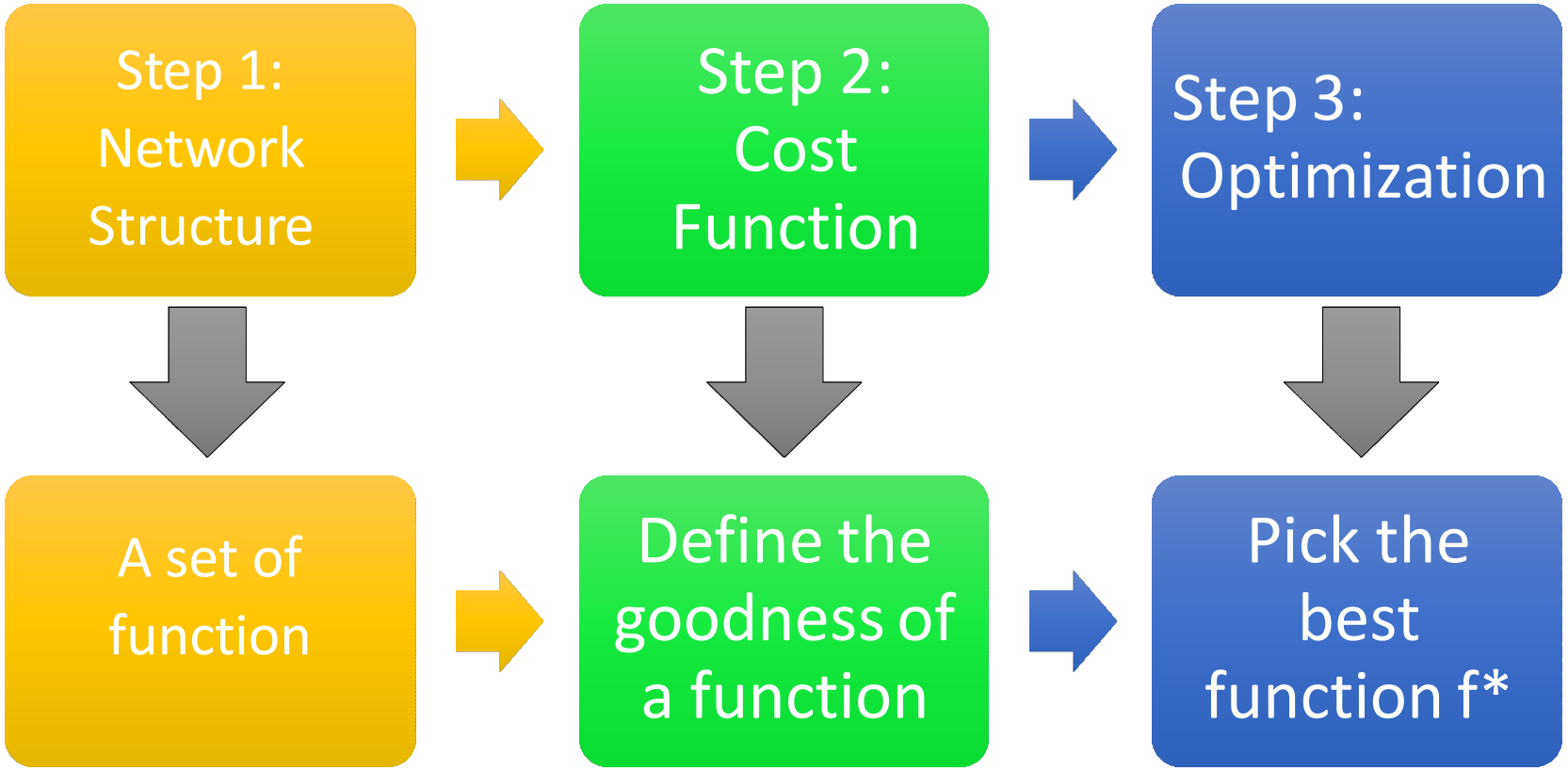
Three Steps for Deep Learning

Step 1: Network Structure → Step 2: Cost Function → Step 3: Optimization

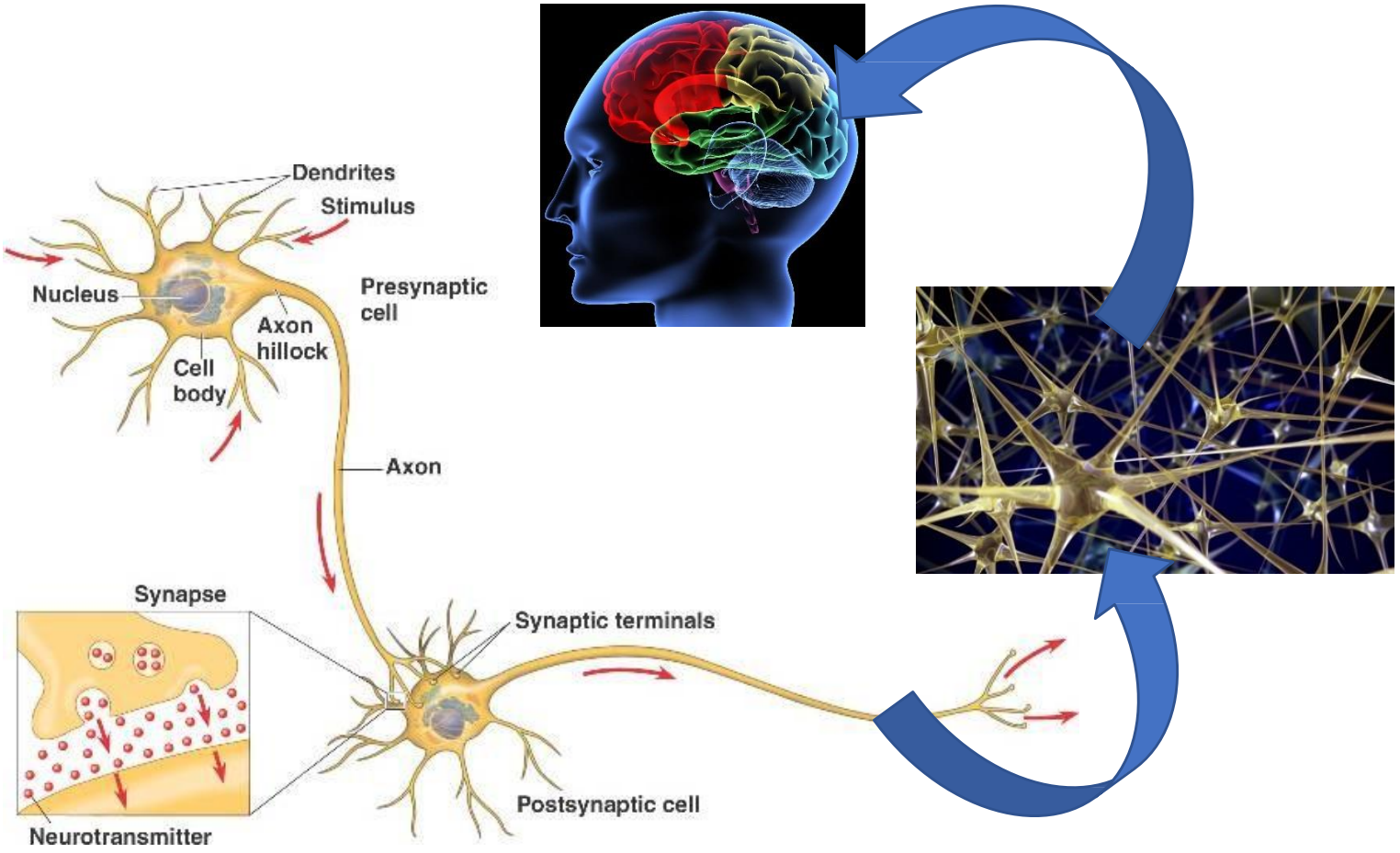
Deep Learning is so simple ……



Three Steps for Deep Learning



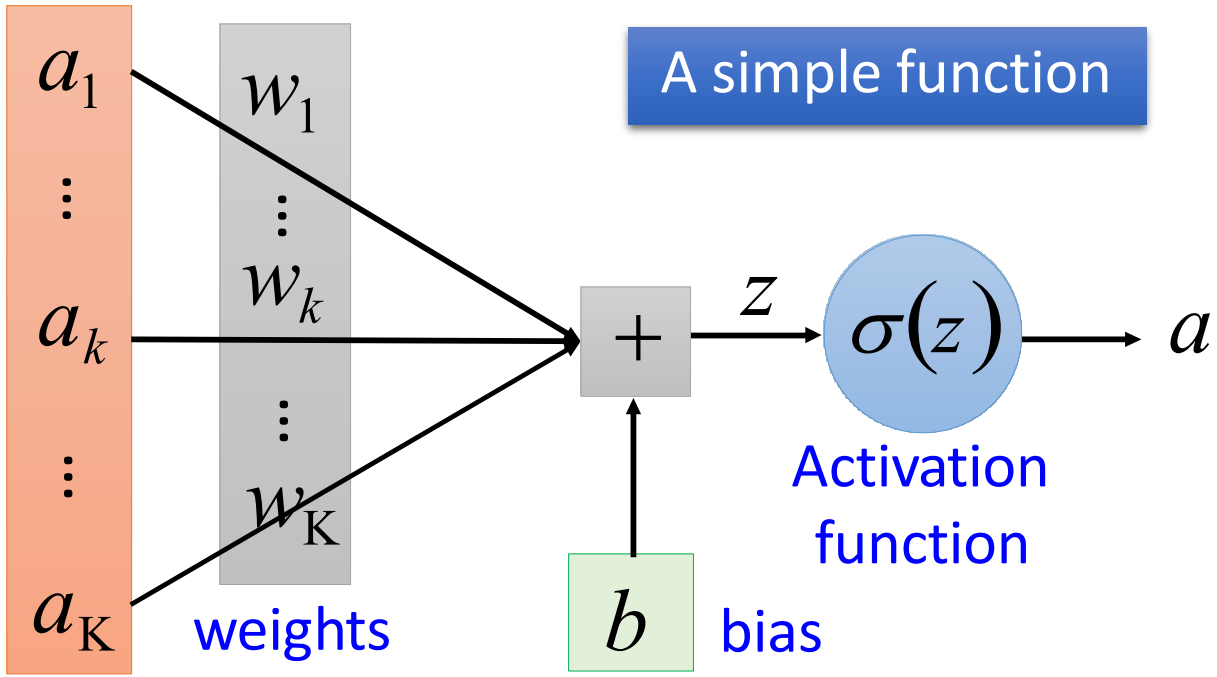
Human Brains



Neural Network

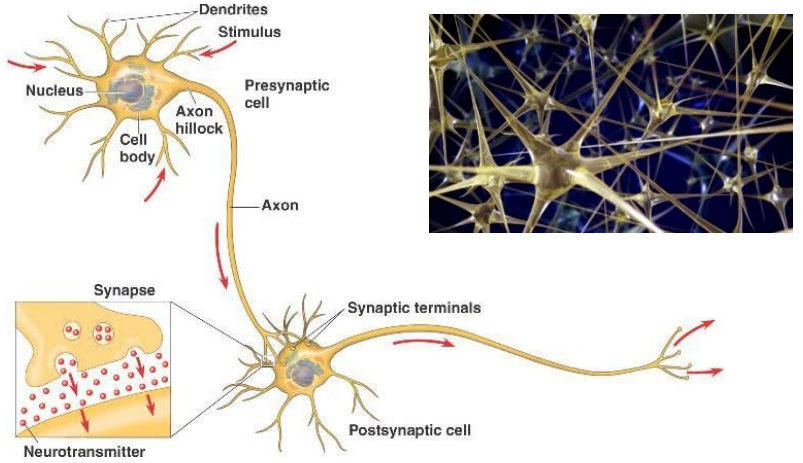
Neuron

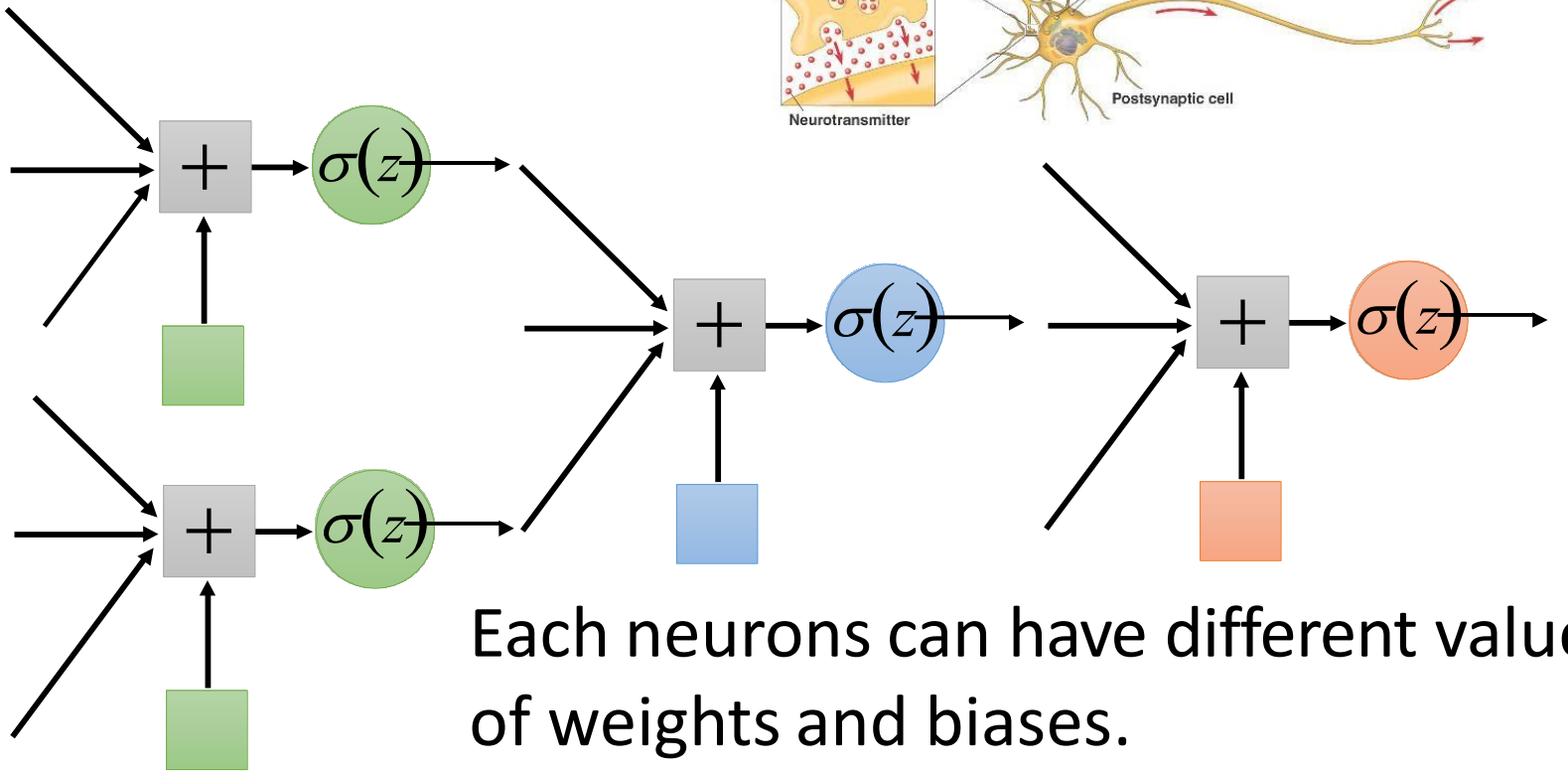
$$z = a_1 w_1 + \cdots + a_k w_k + \cdots + a_K w_K + b$$



Neural Network

Different connections leads to different network structured



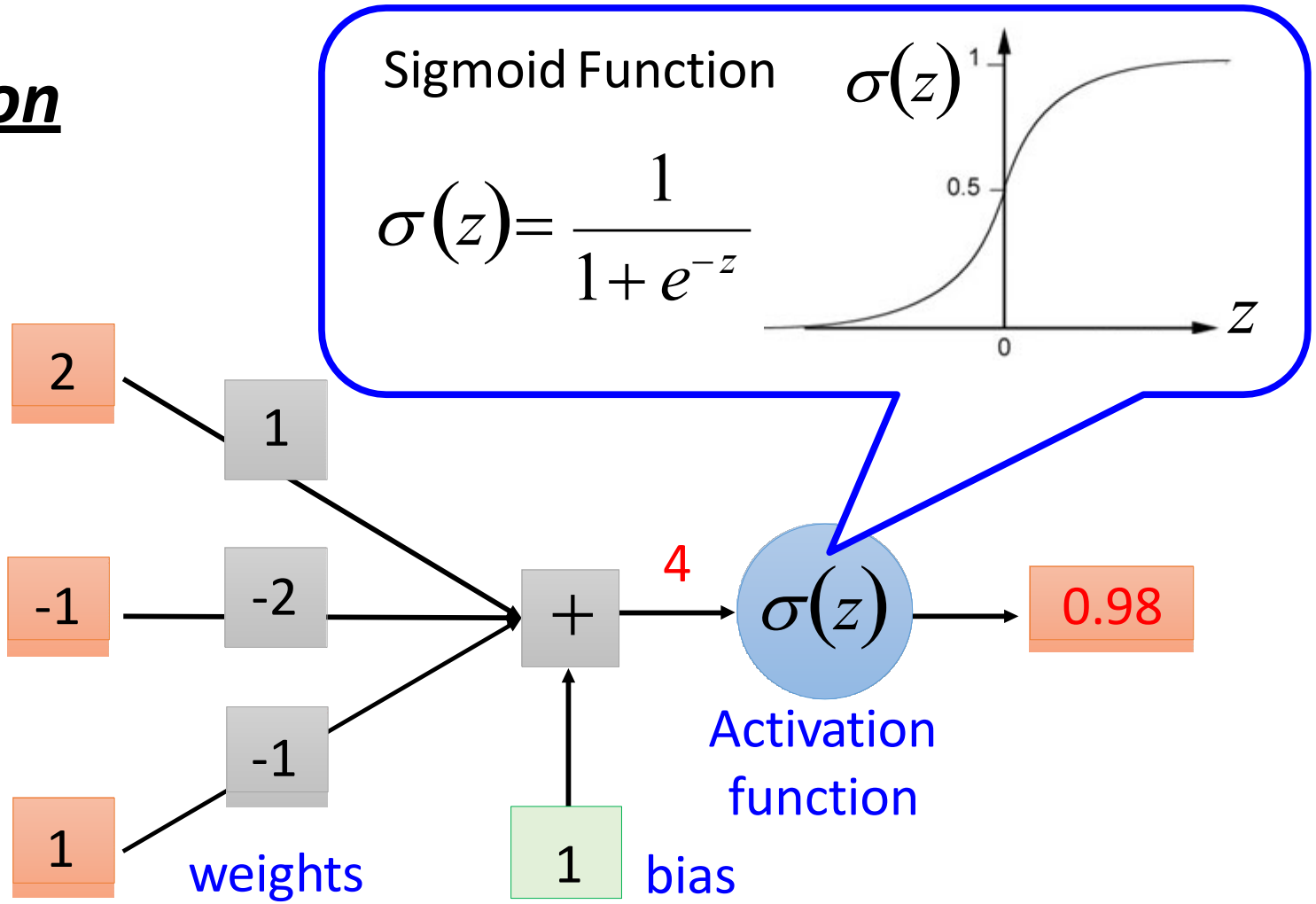


Each neurons can have different values of weights and biases.

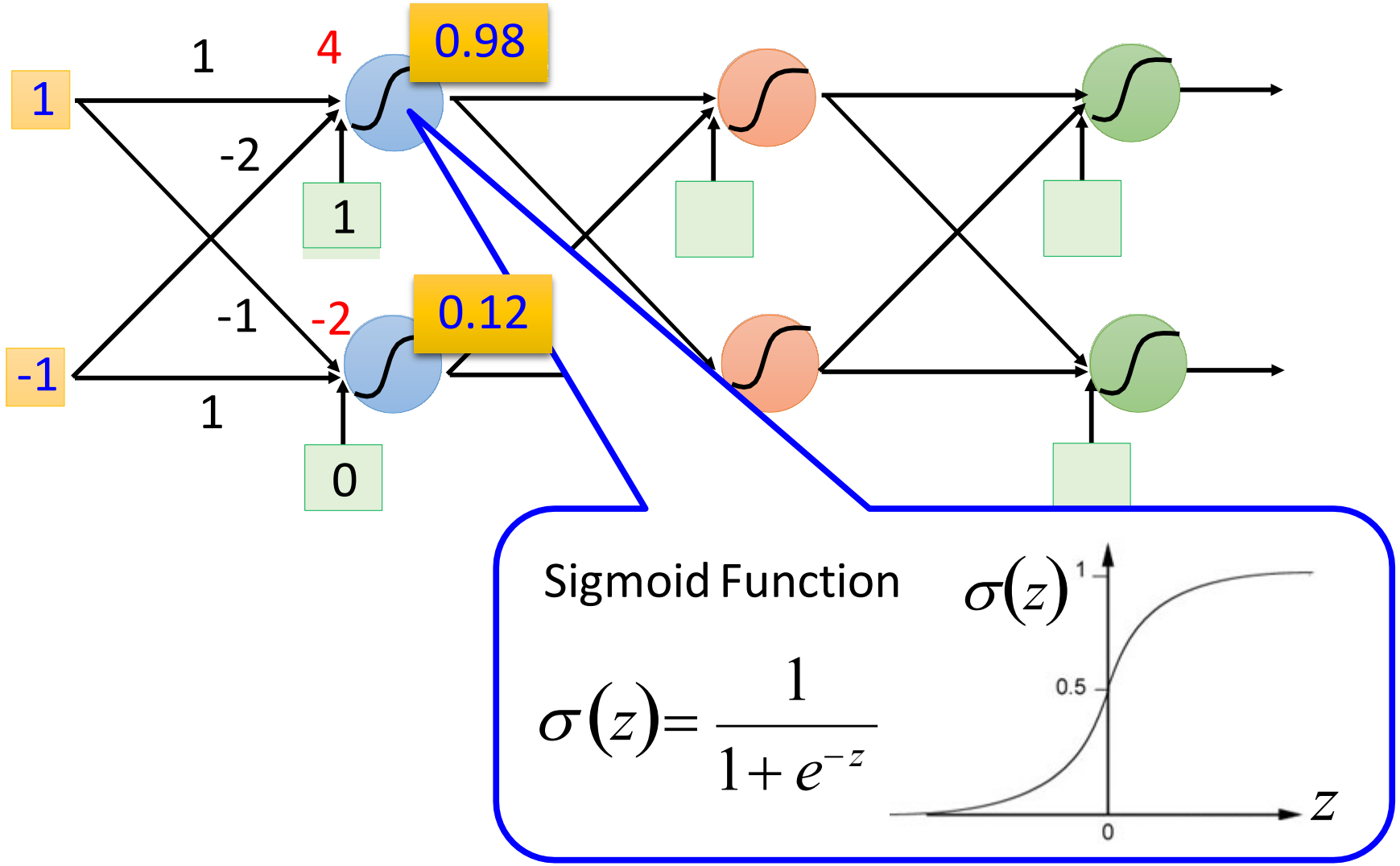
Weights and biases are network parameters

Neural Network

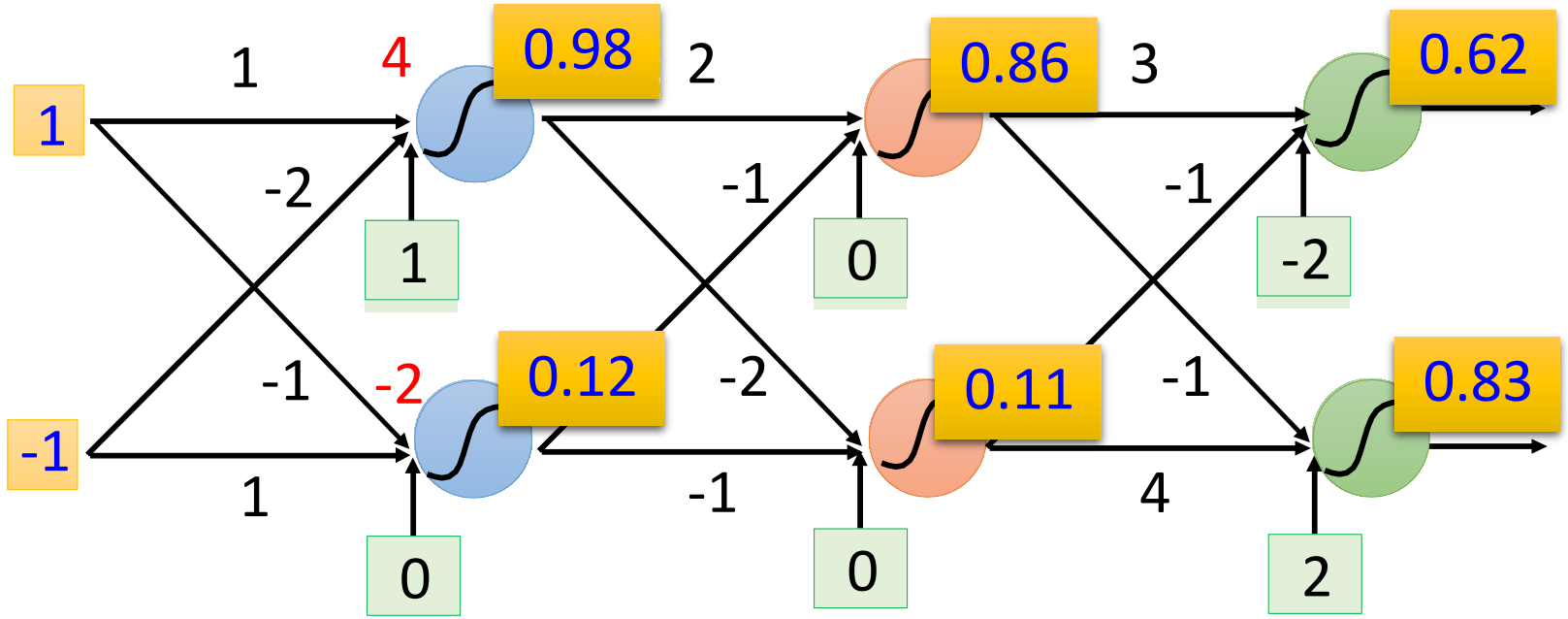
Neuron



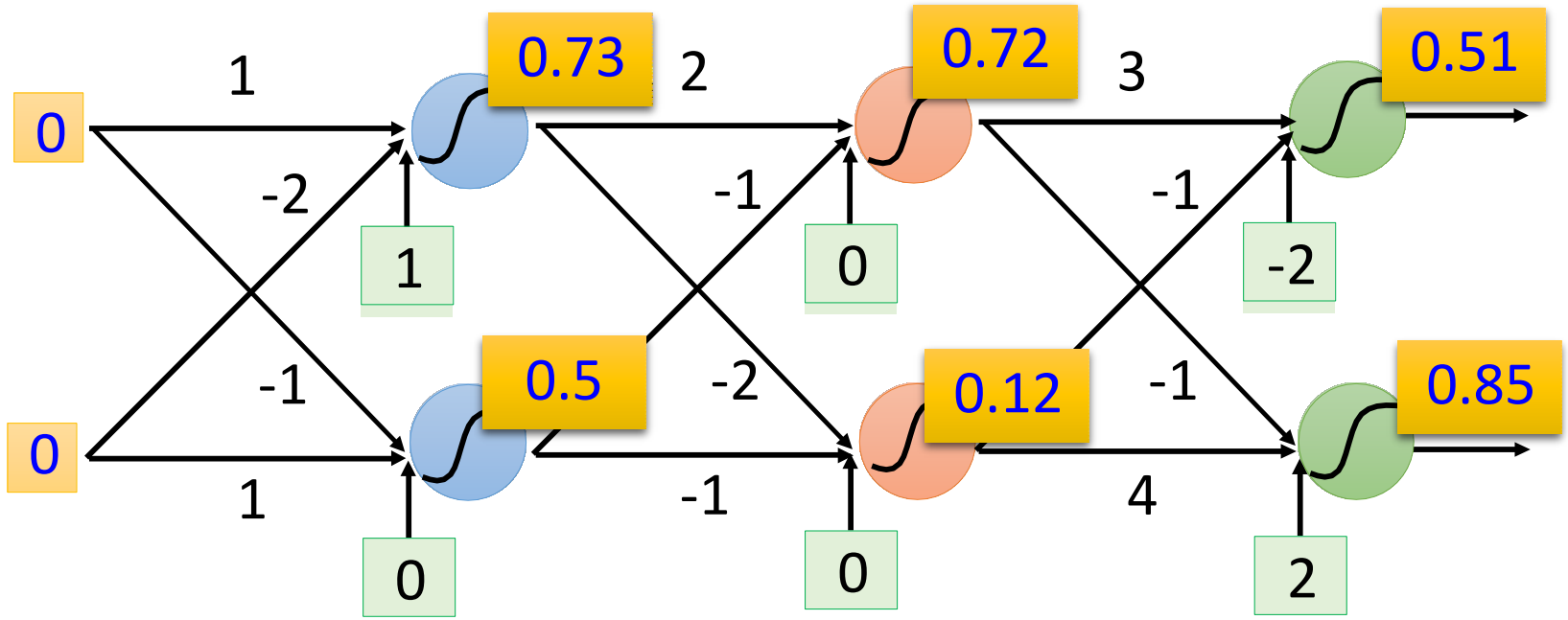
Fully Connect Feedforward



Fully Connect Feedforward Network



Fully Connect Feedforward Network



Network is a function.

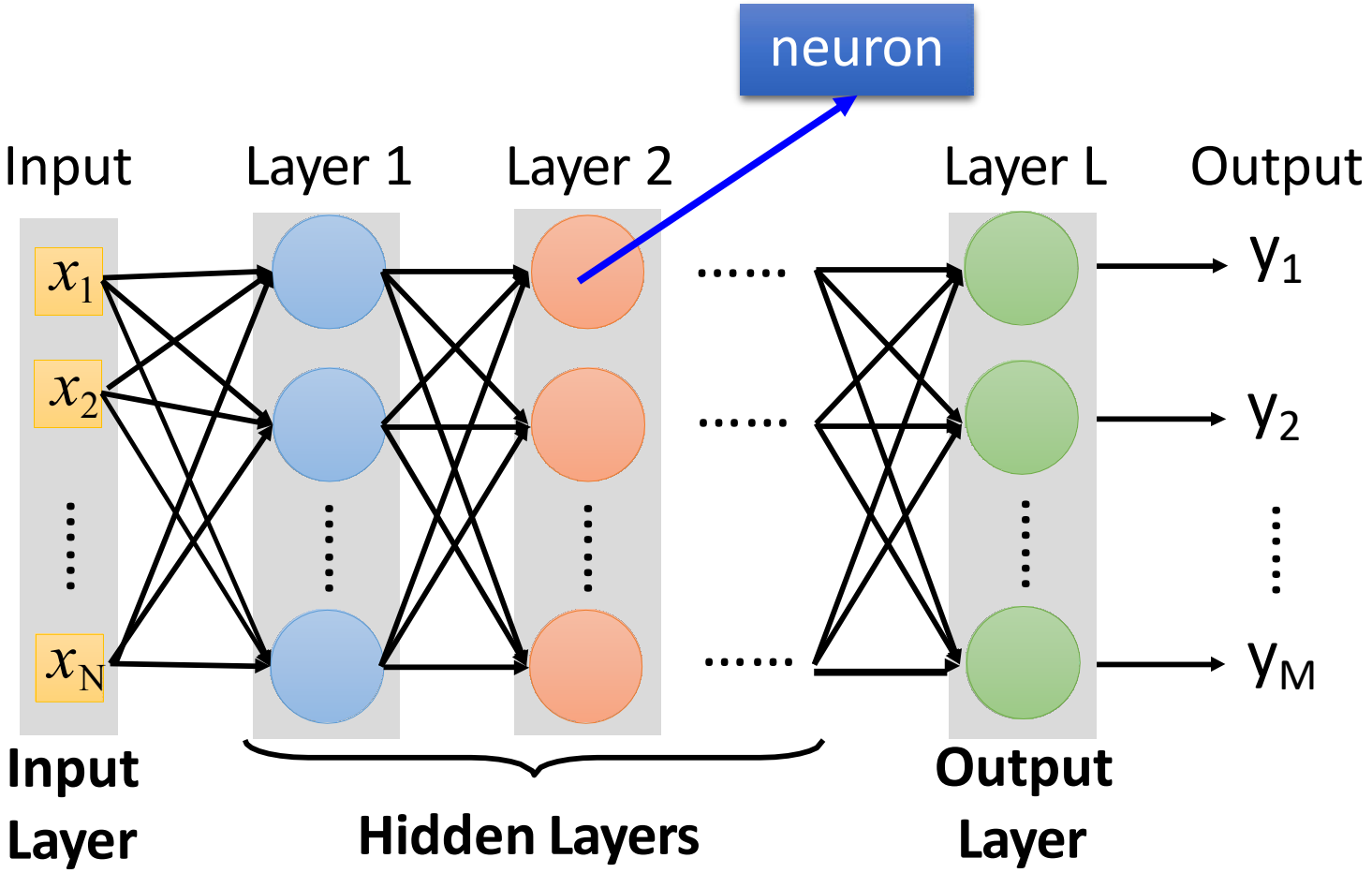
Input vector, output vector

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

Given parameters θ define a function

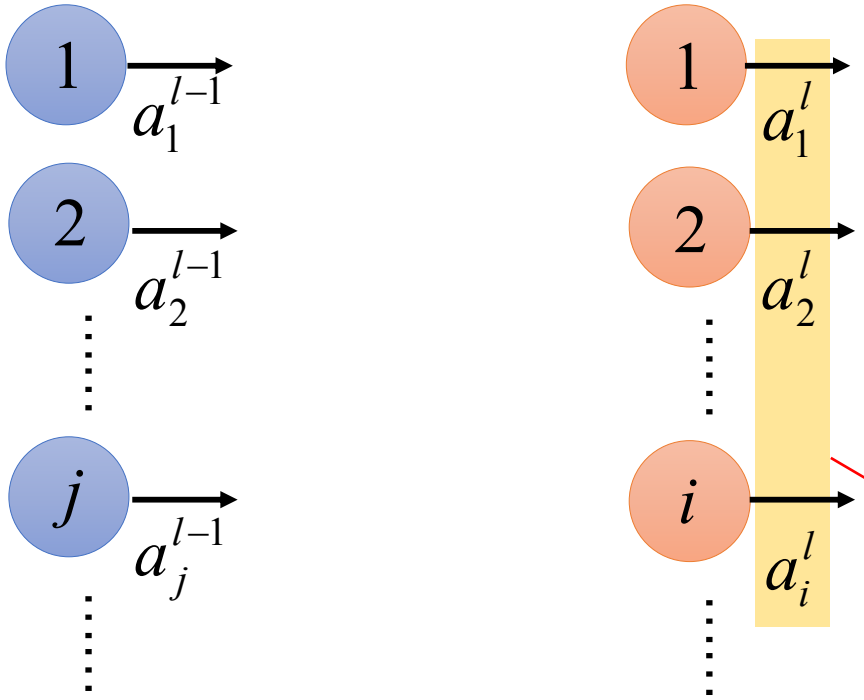
Given network structure, define ***a function set***

Fully Connect Feedforward Network



Deep means many hidden layers

Fully Connected Layer



Layer $l-1$

N_{l-1} nodes

Layer l

N_l nodes

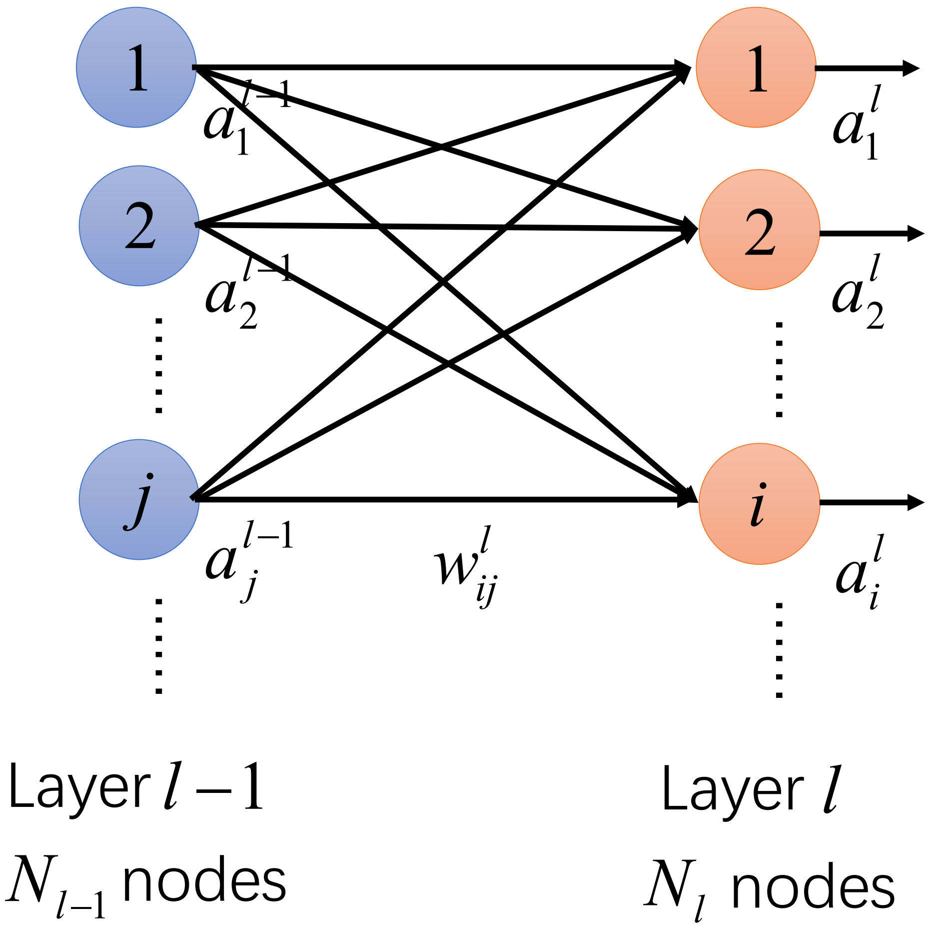
Output of a neuron:

a_i^l — Layer l; Neuron i

Output of one layer:

a^l : a vector

Fully Connected Layer



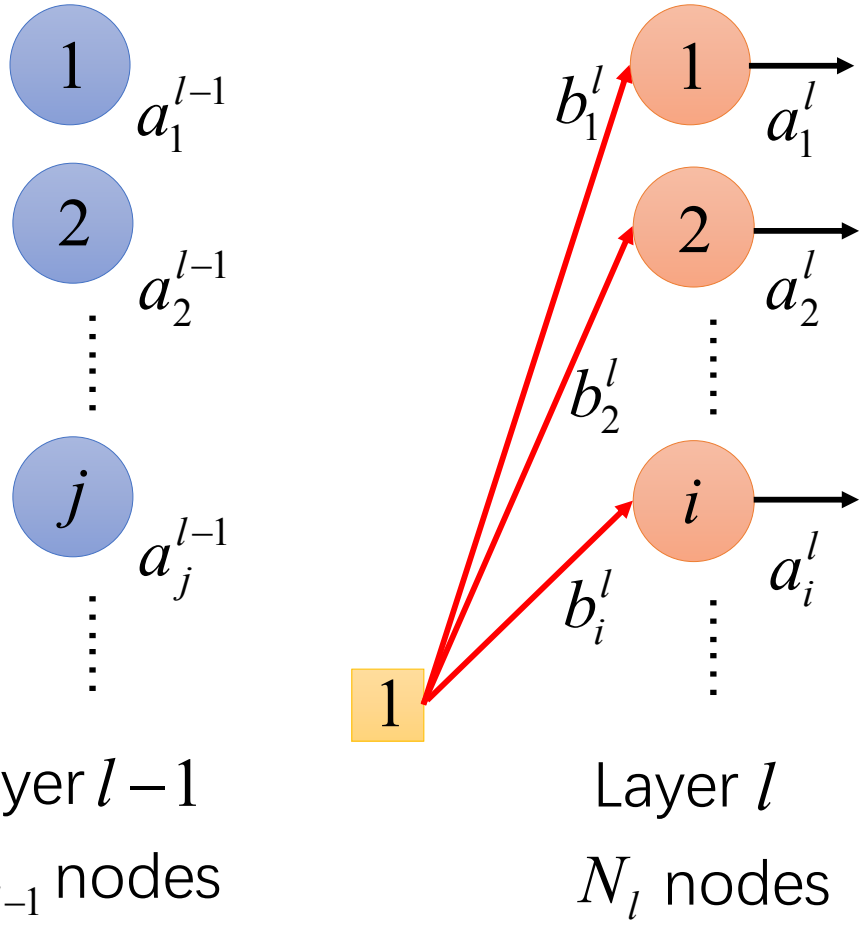
w_{ij}^l → Layer $l-1$ to Layer l

w_{ij}^l → from neuron j(Layer $l-1$) to neuron i (Layer l)

$$W^l = \begin{bmatrix} w_{11}^l & w_{12}^l & \cdots \\ w_{21}^l & w_{22}^l & \\ \vdots & & \ddots \end{bmatrix}$$

Columns: N_{l-1}; Rows: N_l

Fully Connected Layer

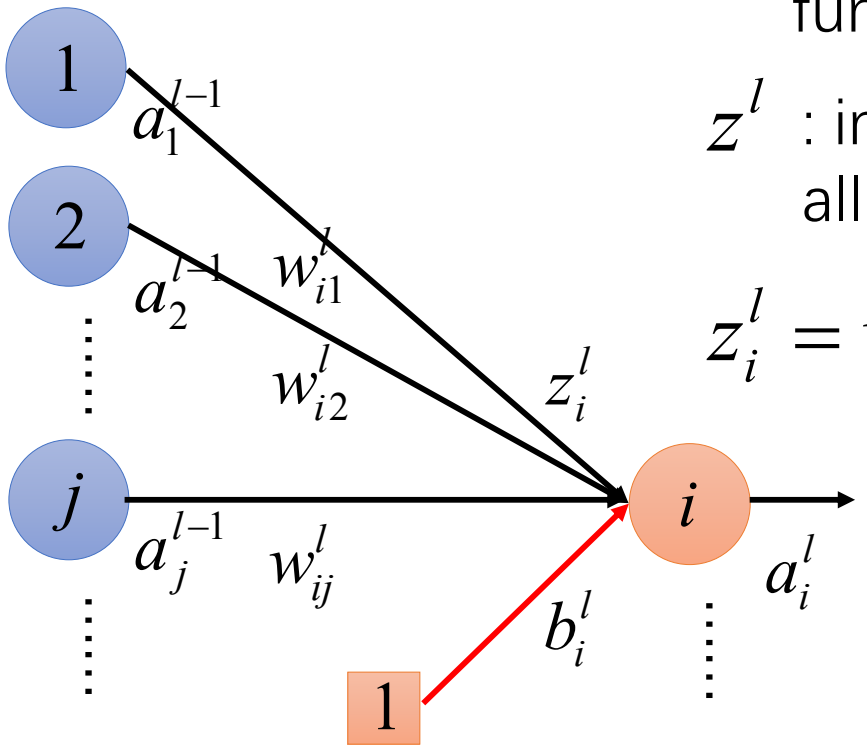


b_i^l : bias for neuron i at layer l

$$b^l = \begin{bmatrix} b_1^l \\ b_2^l \\ \vdots \\ b_i^l \\ \vdots \end{bmatrix}$$

bias for all neurons in layer l

Fully Connected Layer



Layer $l-1$

N_{l-1} nodes

Layer l

N_l nodes

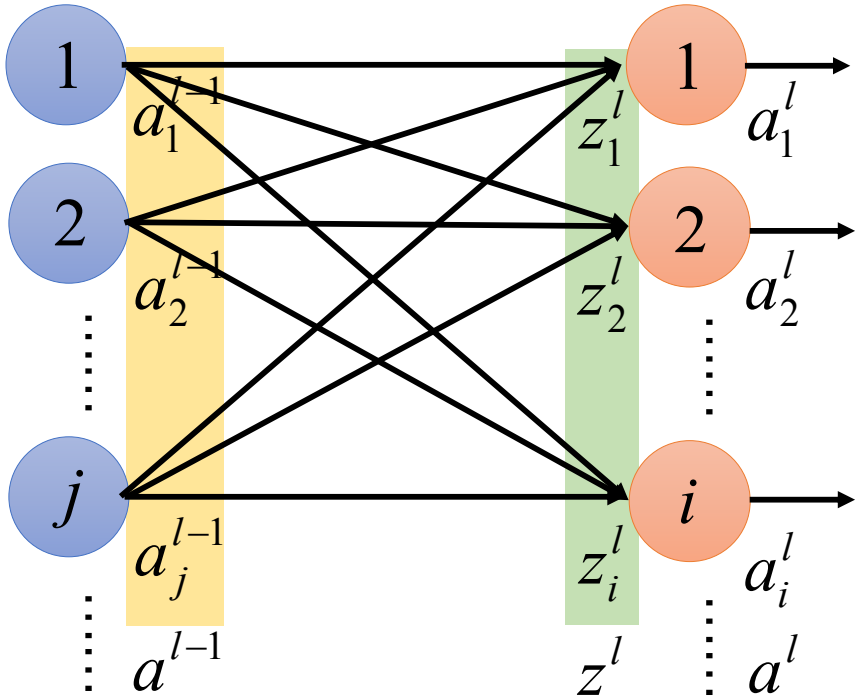
z_i^l : input of the activation function for neuron i at layer l

z^l : input of the activation function all the neurons in layer l

$$z_i^l = w_{i1}^l a_1^{l-1} + w_{i2}^l a_2^{l-1} \ldots + b_i^l$$

$$z_i^l = \sum_{j=1}^{N_{l-1}} w_{ij}^l a_j^{l-1} + b_i^l$$

Relations between Layer Outputs



Layer $l-1$

N_{l-1} nodes

Layer l

N_l nodes

$$z_1^l = w_{11}^l a_1^{l-1} + w_{12}^l a_2^{l-1} + \cdots + b_1^l$$

$$z_2^l = w_{21}^l a_1^{l-1} + w_{22}^l a_2^{l-1} + \cdots + b_2^l$$

$$\vdots$$

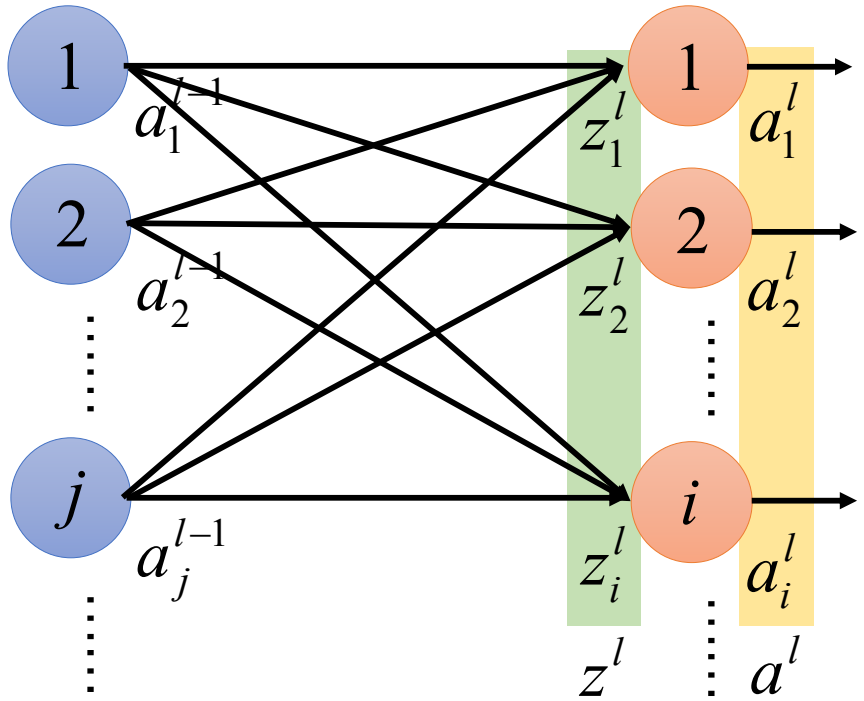
$$z_i^l = w_{i1}^l a_1^{l-1} + w_{i2}^l a_2^{l-1} + \cdots + b_i^l$$

$$\vdots$$

$$\begin{bmatrix} z_1^l \\ z_2^l \\ \vdots \\ z_i^l \\ \vdots \end{bmatrix} = \begin{bmatrix} w_{11}^l & w_{12}^l & \cdots \\ w_{21}^l & w_{22}^l & \\ \vdots & & \ddots \\ & & \end{bmatrix} \begin{bmatrix} a_1^{l-1} \\ a_2^{l-1} \\ \vdots \\ a_i^{l-1} \\ \vdots \end{bmatrix} + \begin{bmatrix} b_1^l \\ b_2^l \\ \vdots \\ b_i^l \\ \vdots \end{bmatrix}$$

$$z^l = W^l a^{l-1} + b^l$$

Relations between Layer Outputs



Layer $l-1$

N_{l-1} nodes

Layer l

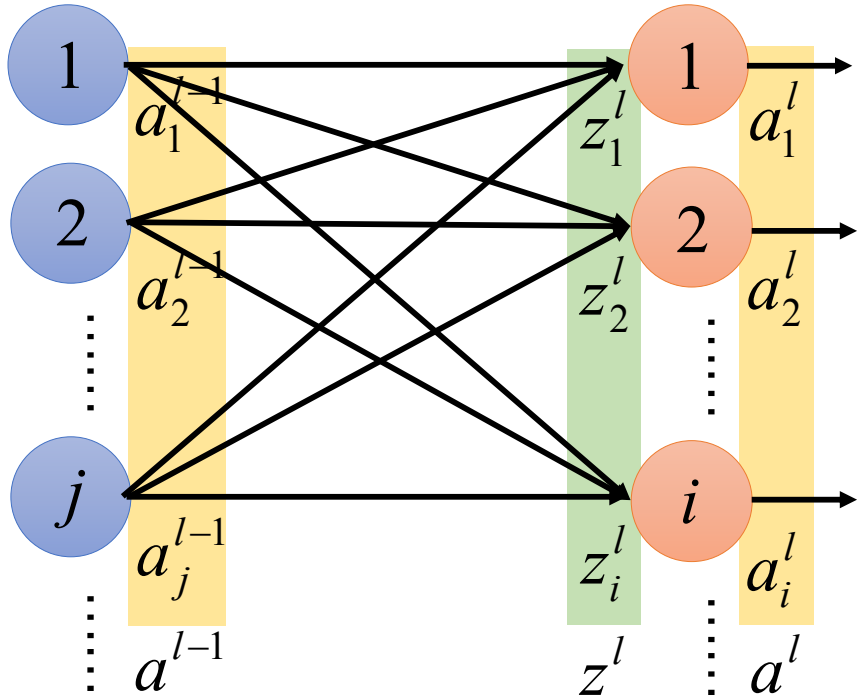
N_l nodes

$$a_i^l = \sigma\left(z_i^l\right)$$

$$\begin{bmatrix} a_1^l \\ a_2^l \\ \vdots \\ a_i^l \\ \vdots \end{bmatrix} = \begin{bmatrix} \sigma\left(z_1^l\right) \\ \sigma\left(z_2^l\right) \\ \vdots \\ \sigma\left(z_i^l\right) \\ \vdots \end{bmatrix}$$

$$a^l = \sigma\left(z^l\right)$$

Relations between Layer Outputs



Layer $l-1$

N_{l-1} nodes

Layer l

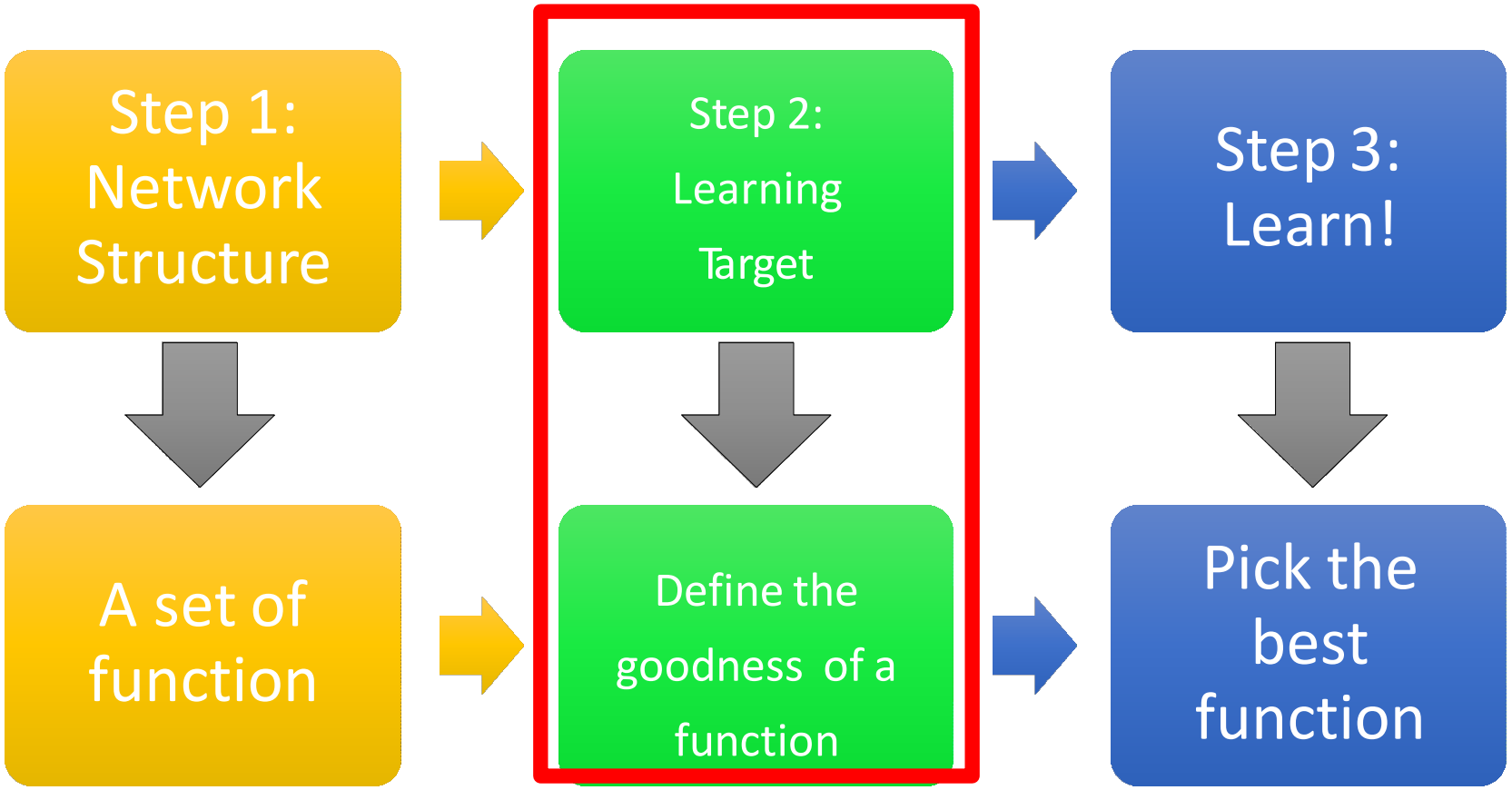
N_l nodes

$$z^l = W^l a^{l-1} + b^l$$

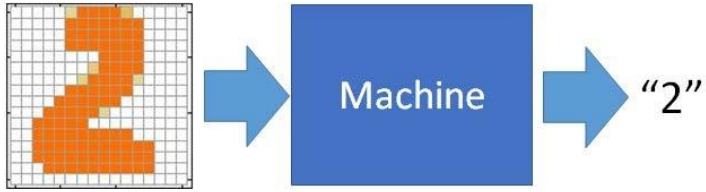
$$a^l = \sigma\left(z^l\right)$$

$$a^l = \sigma\left(W^l a^{l-1} + b^l\right)$$

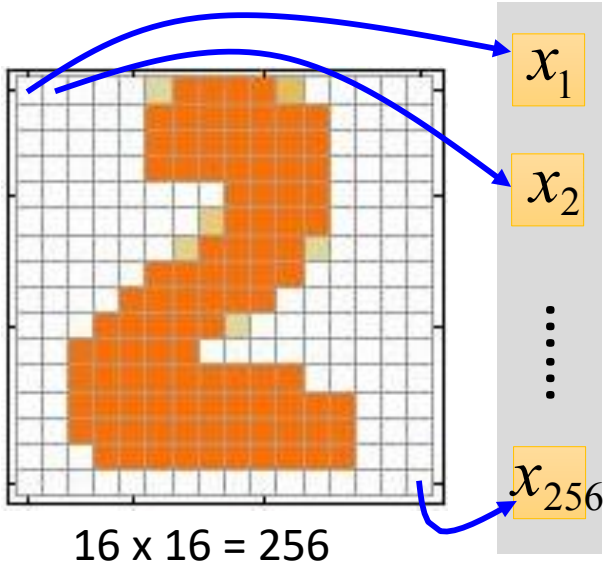
Three Steps for Deep Learning



Example Application



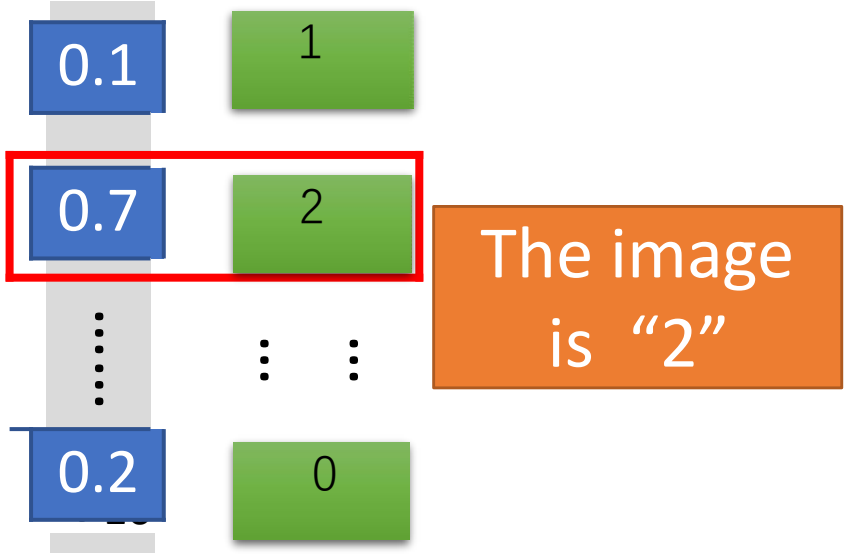
Input



Ink → 1

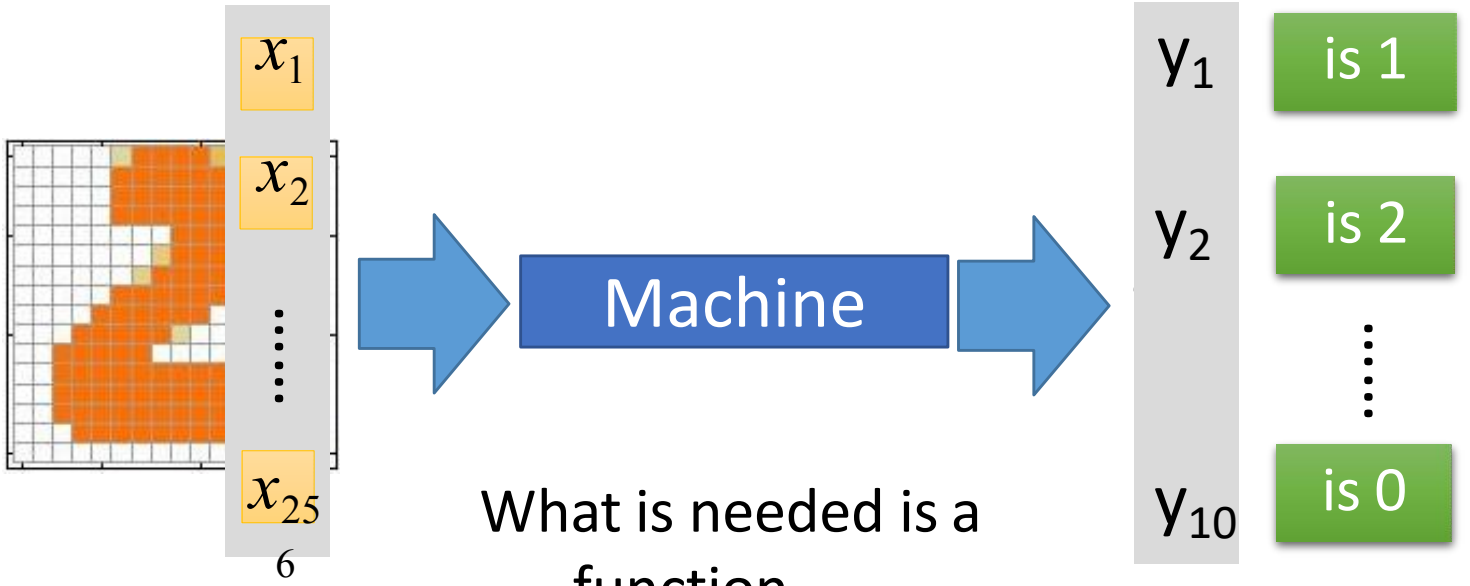
No ink → 0

Output

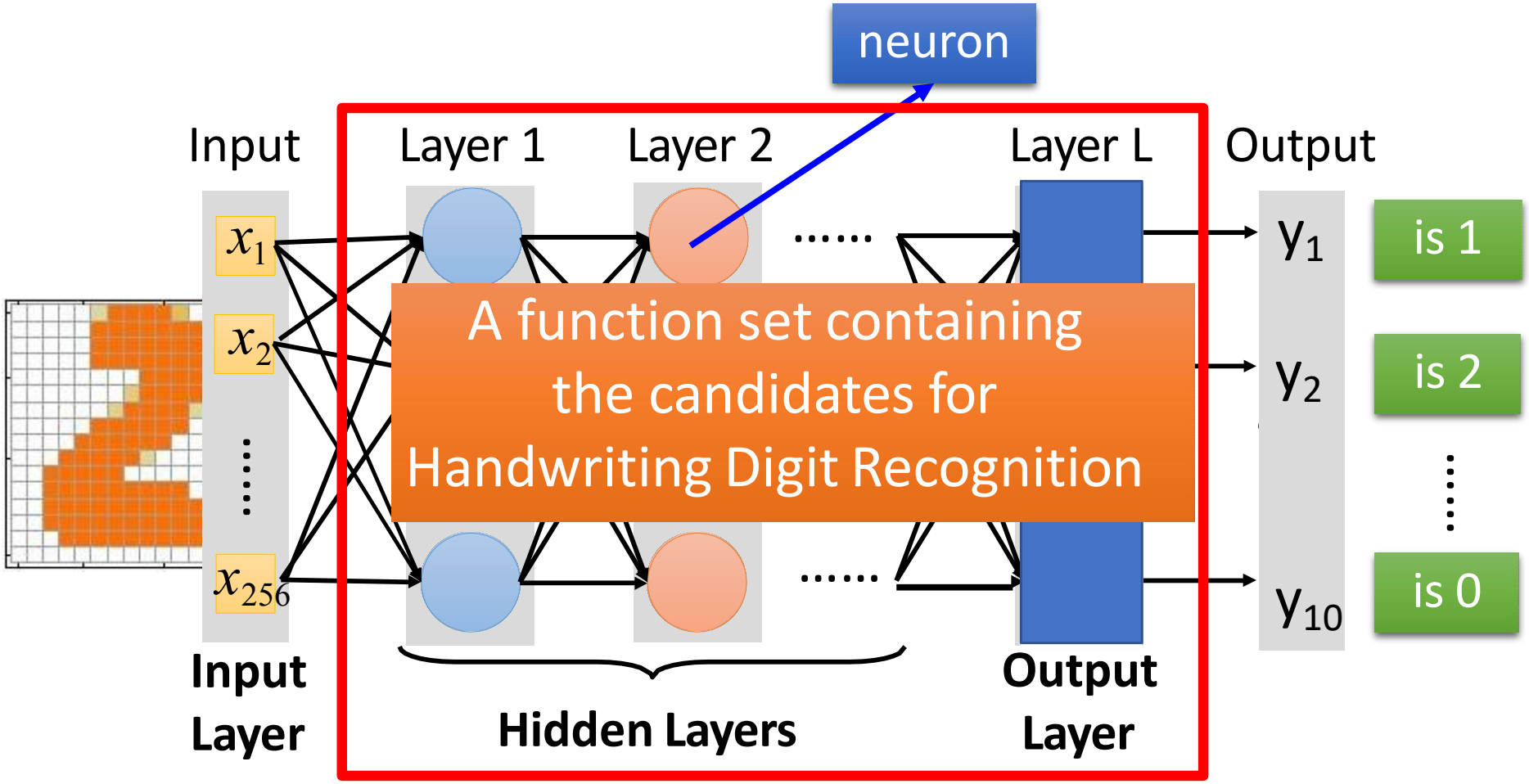


Example Application

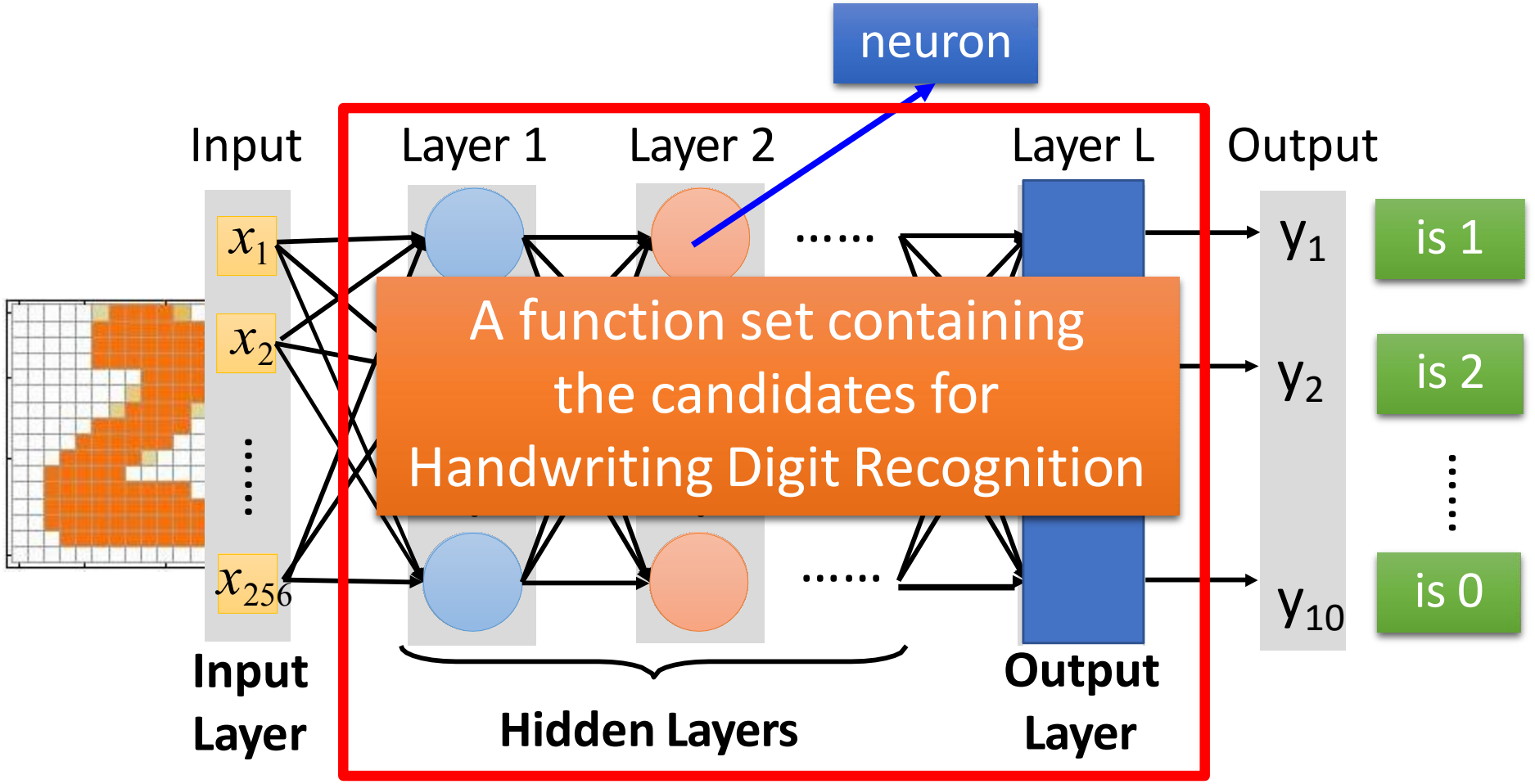
- Handwriting Digit Recognition



Fully Connect Feedforward Network



Fully Connect Feedforward Network

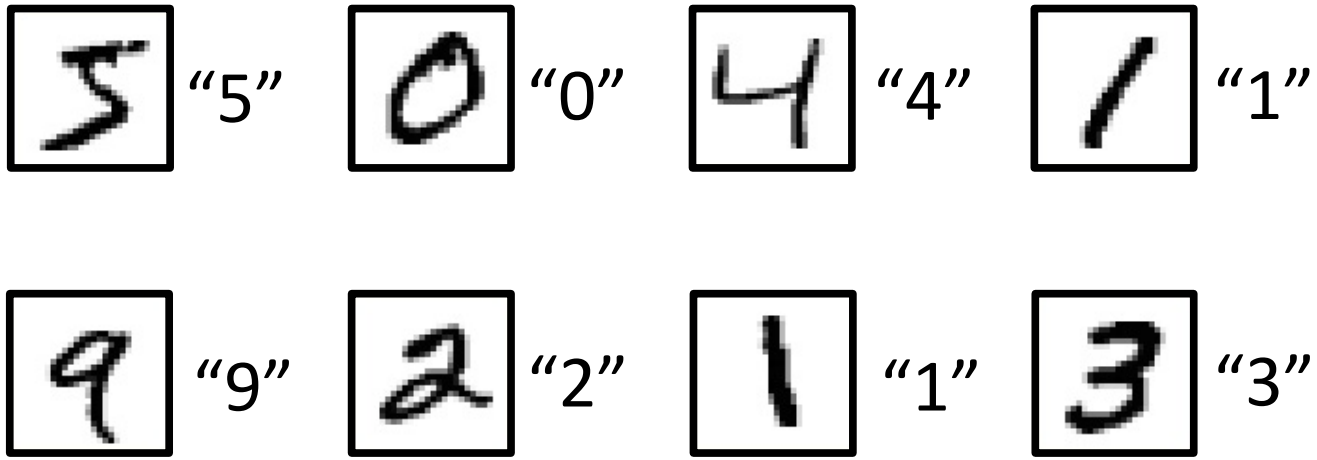


Step 2 Define the goodness of function based on training data （Ensure the learning target）

Step 3 Pick the best function(optimize the model)

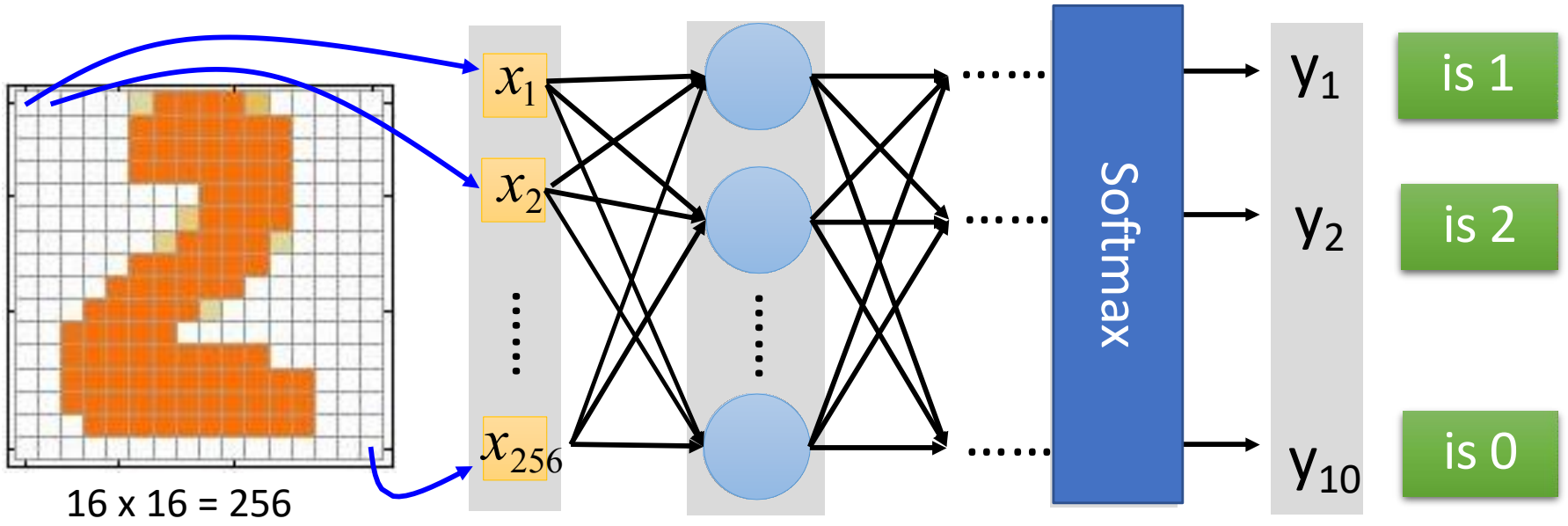
Training Data

- Preparing training data: images and their labels



The learning target is defined on the training data.

Learning Target



Ink → 1

No ink → 0

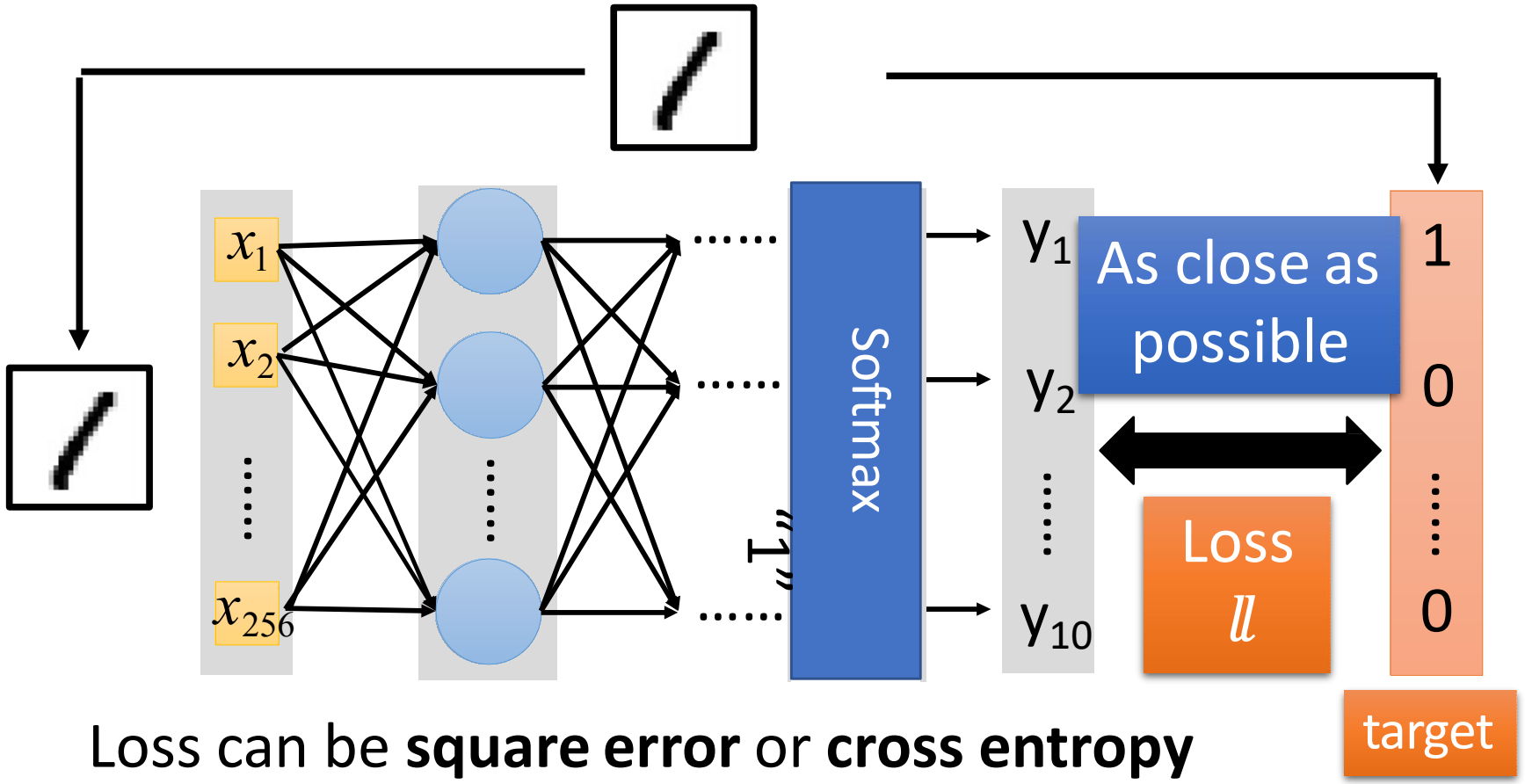
The learning target is ……

Input:  → y_1 has the maximum value

Input:  → y_2 has the maximum value

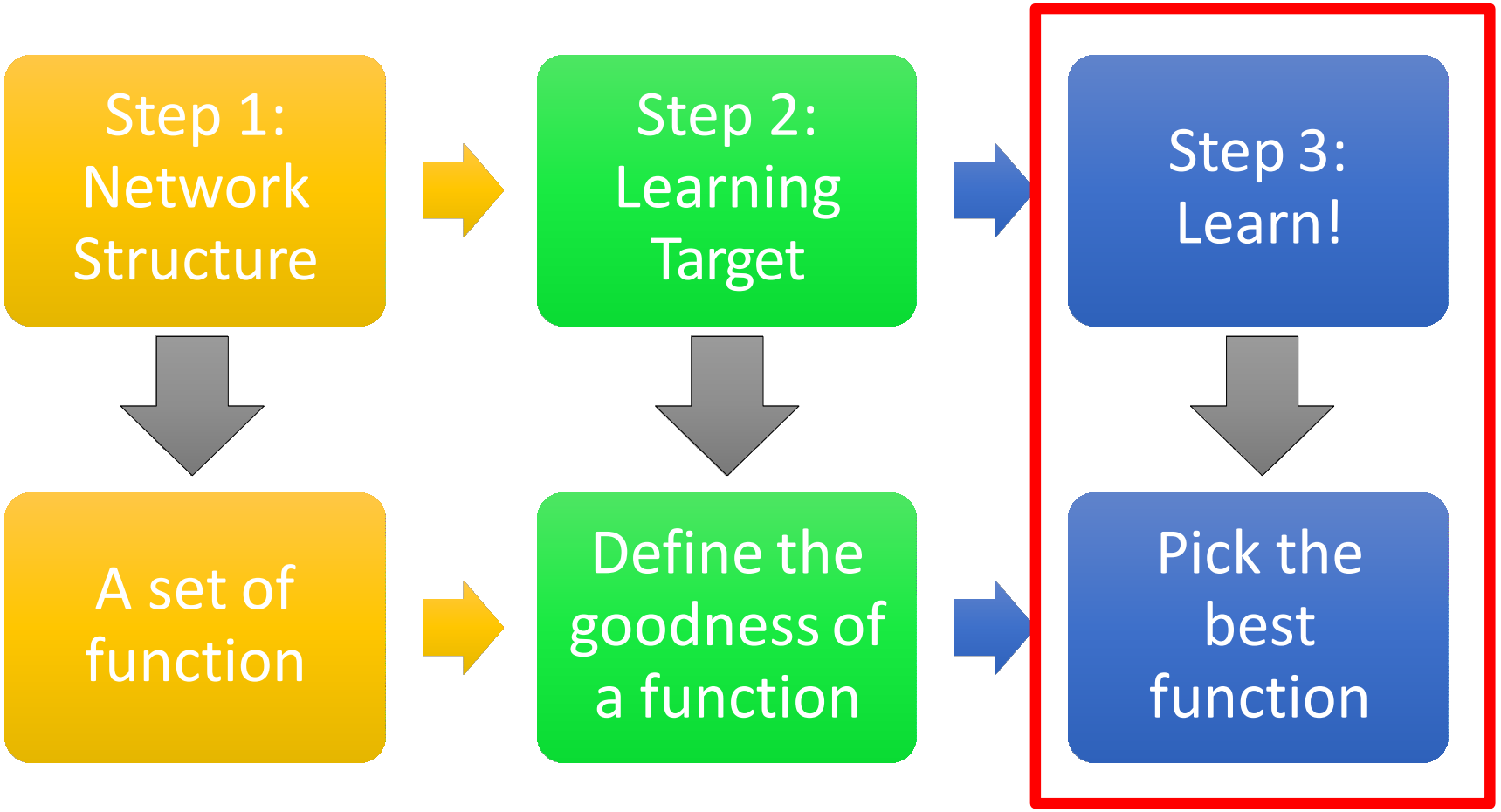
Loss

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



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

You can do lots of different things



How to pick the best function

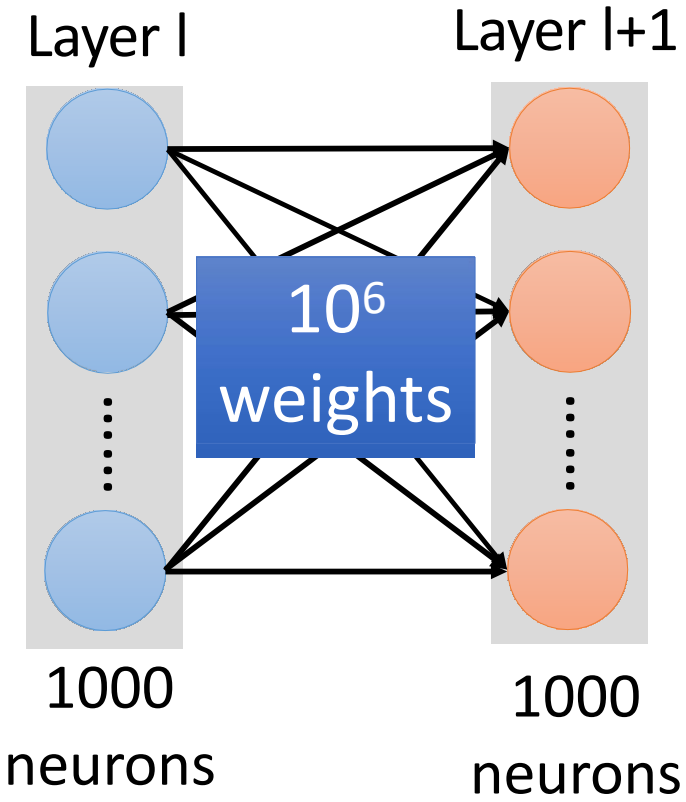
Find ***network parameters θ*** that minimize total loss L

Enumerate all possible values

Network parameters $\theta = \{ w_1, w_2, \ldots, b_1, b_2, b_3, \ldots \}$

Millions of parameters

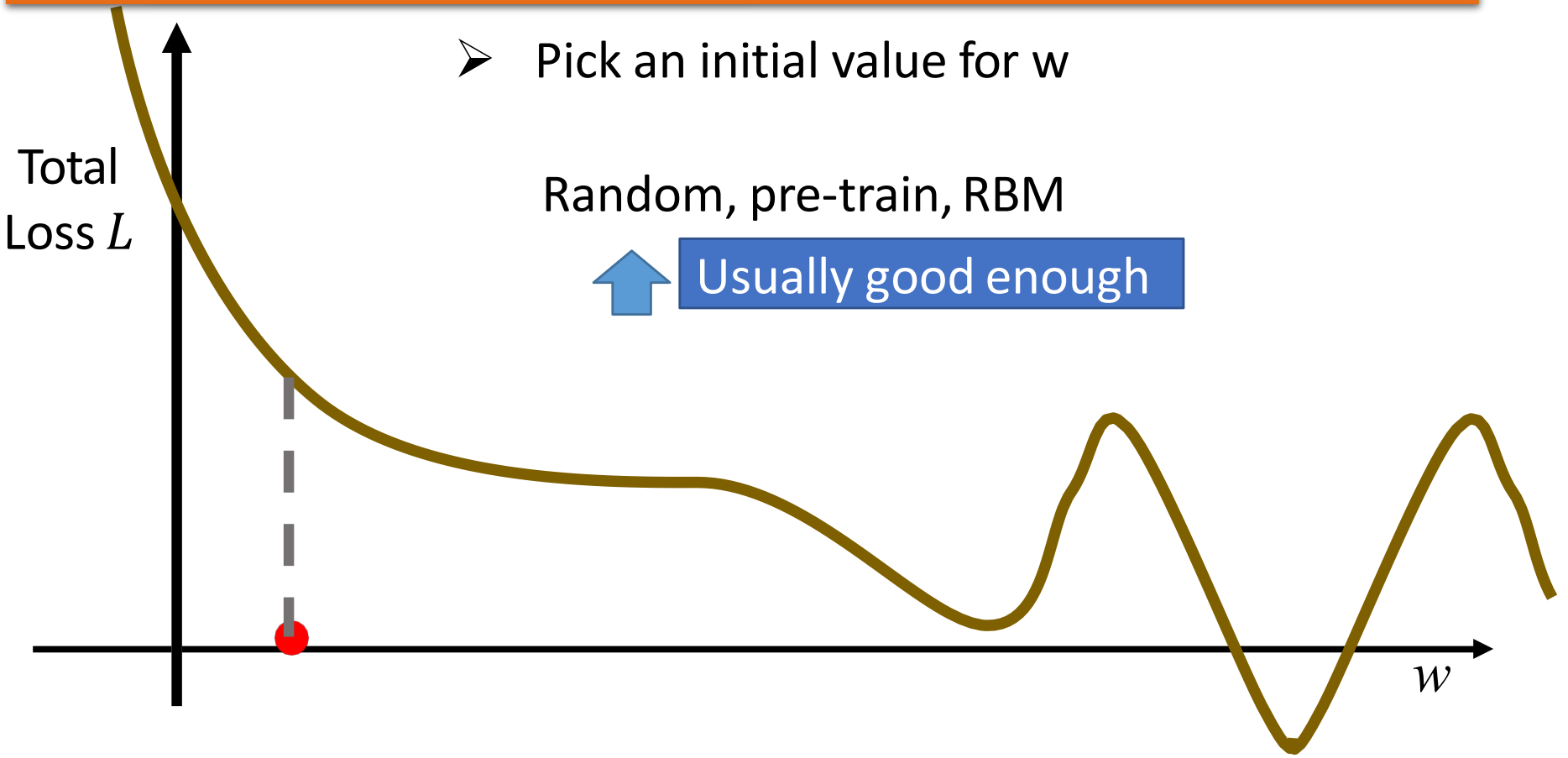
E.g. speech recognition: 8 layers and 1000 neurons each layer



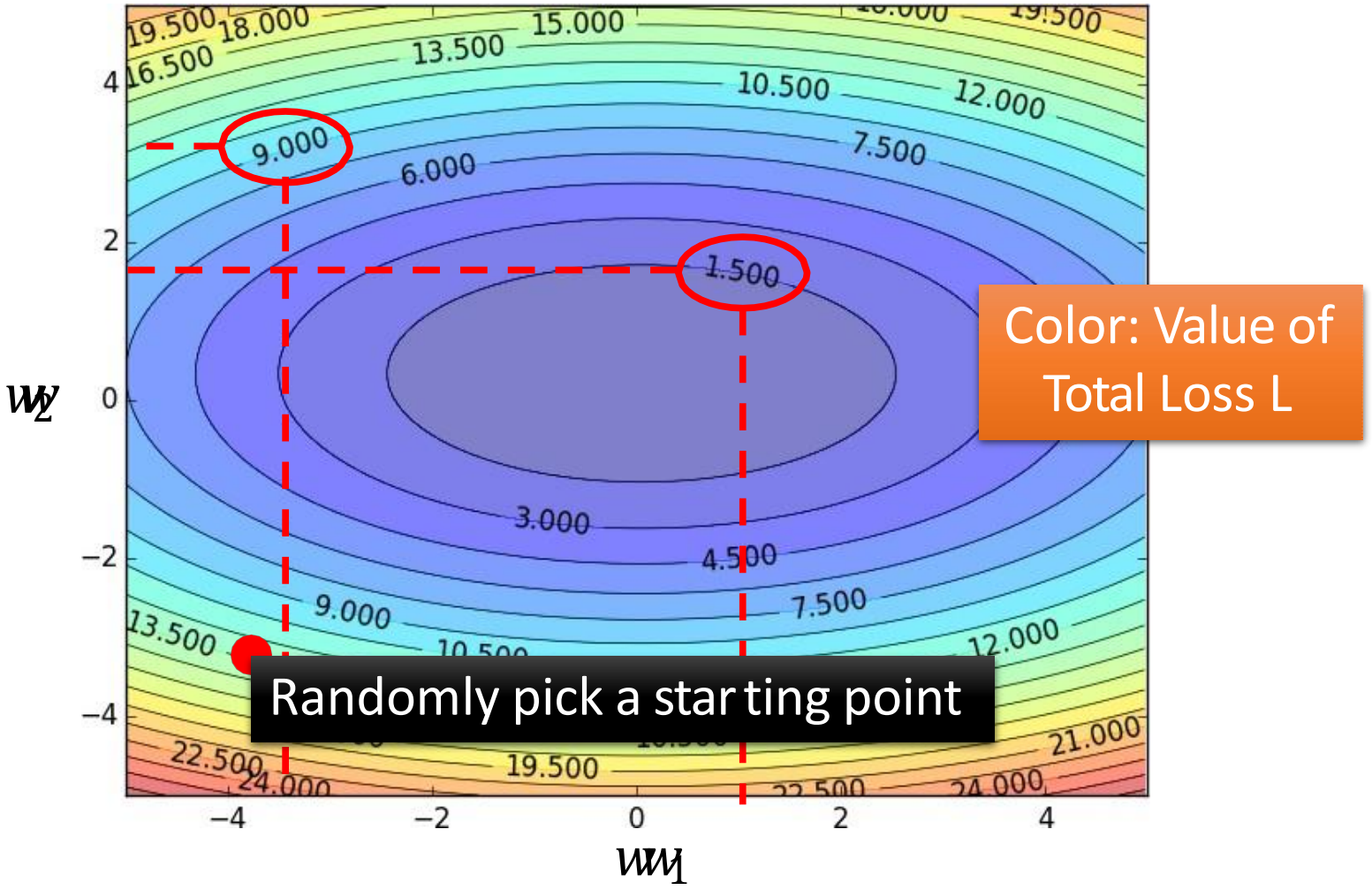
Gradient Descent

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

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

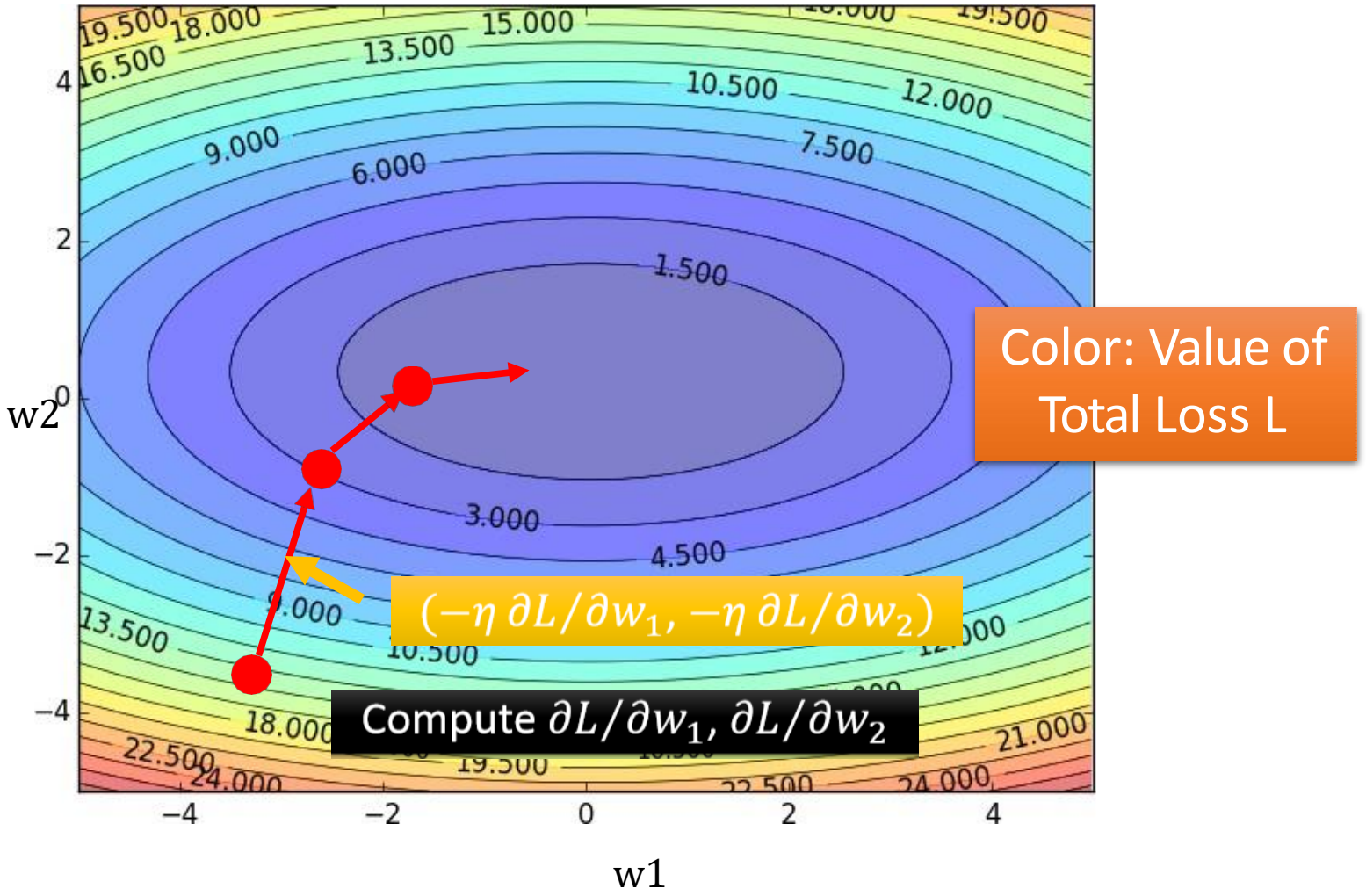


Gradient Descent



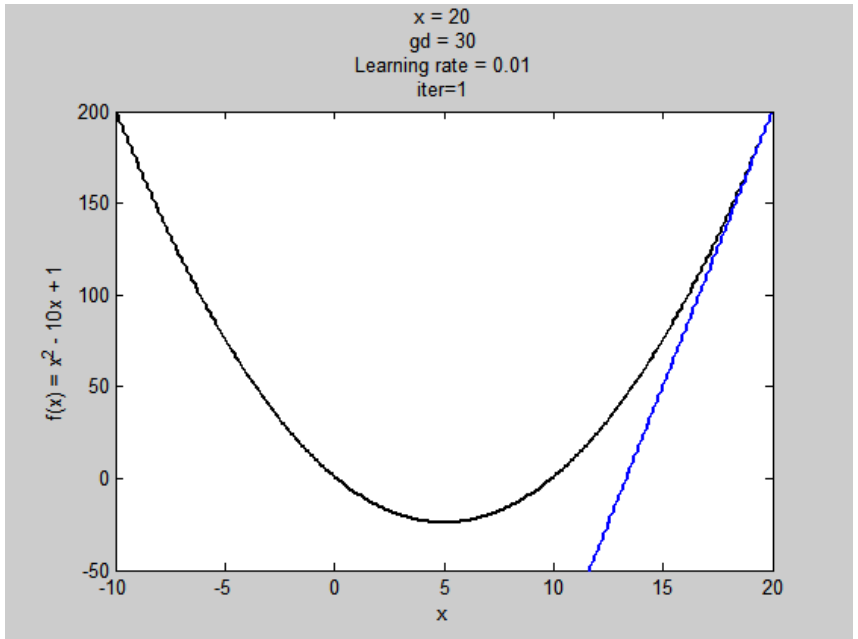
Gradient Descent

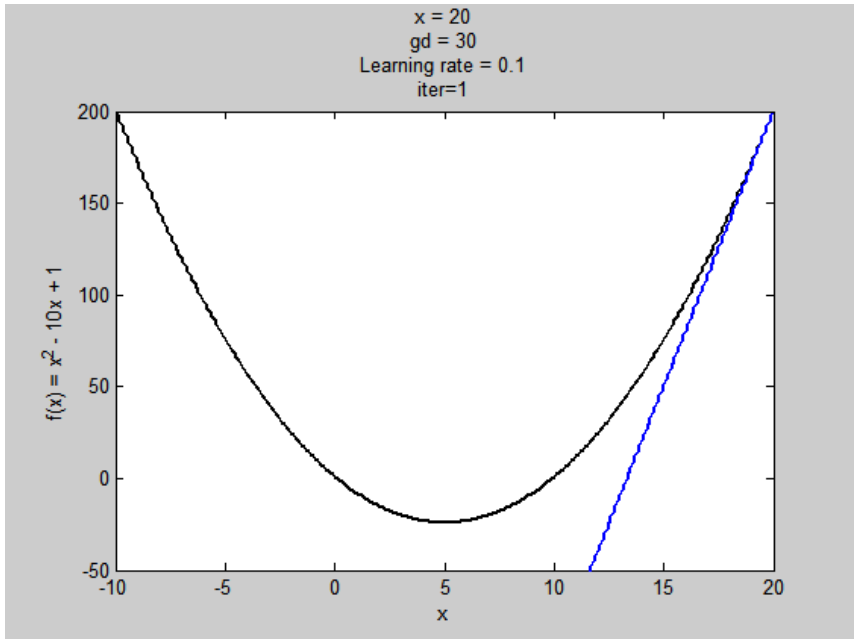
Hopfully, we would reach a minima .....

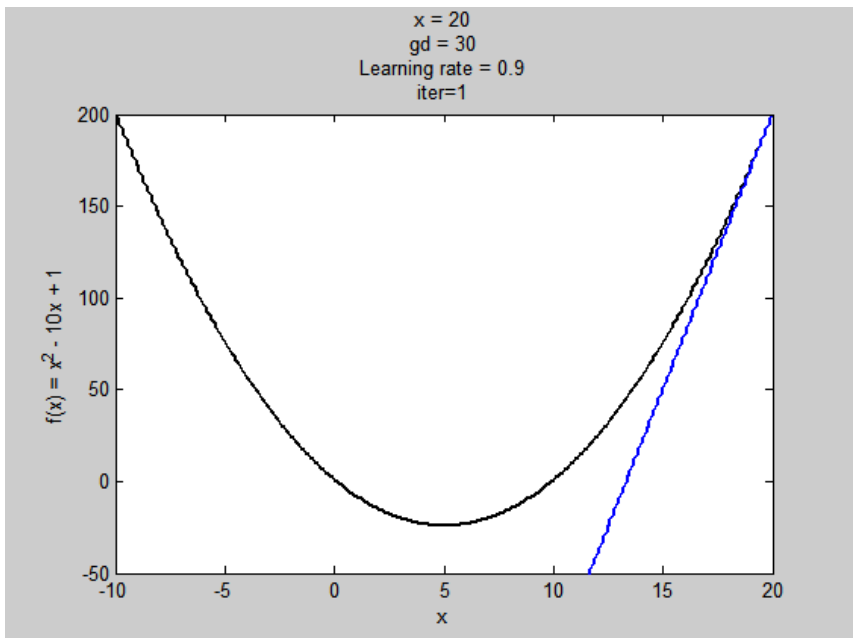


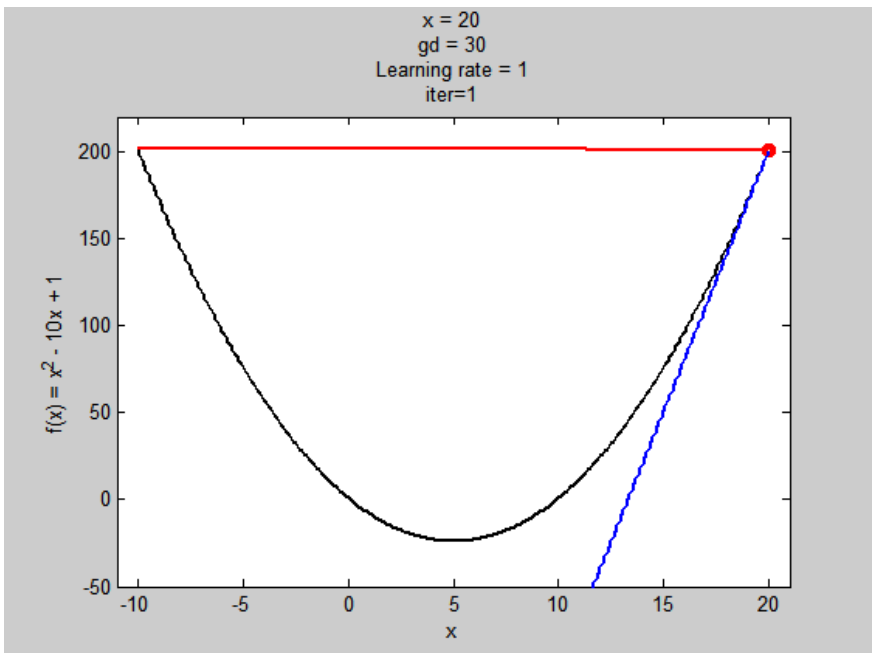
Color: Value of Total Loss L

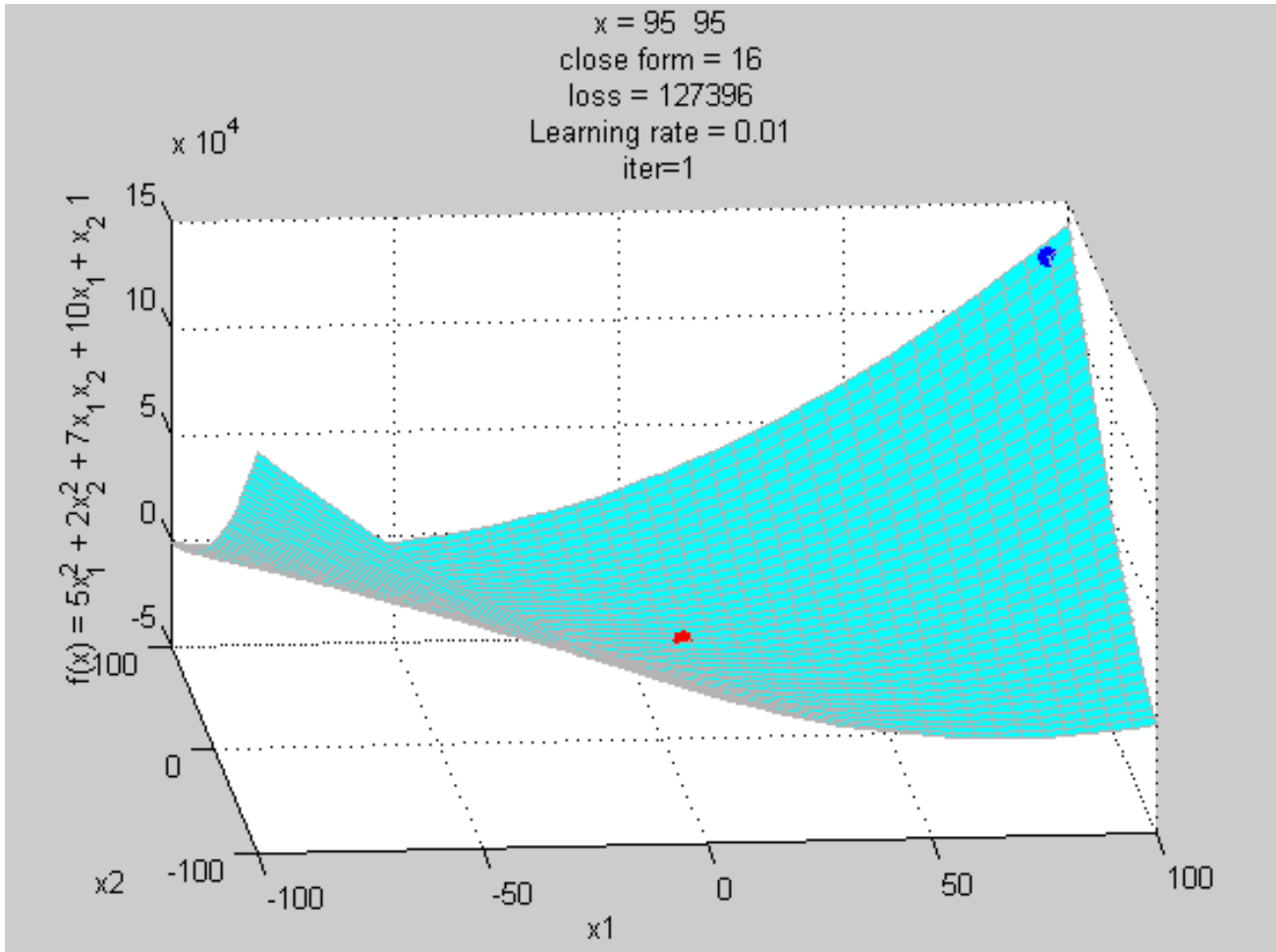
Ref: https://en.wikipedia.org/wiki/Gradient_descent


x = 20
gd = 30
Learning rate = 0.01
iter=1
f(x) = x^2 - 10x + 1
x


x = 20
gd = 30
Learning rate = 0.1
iter=1
f(x) = x^2 - 10x + 1
x

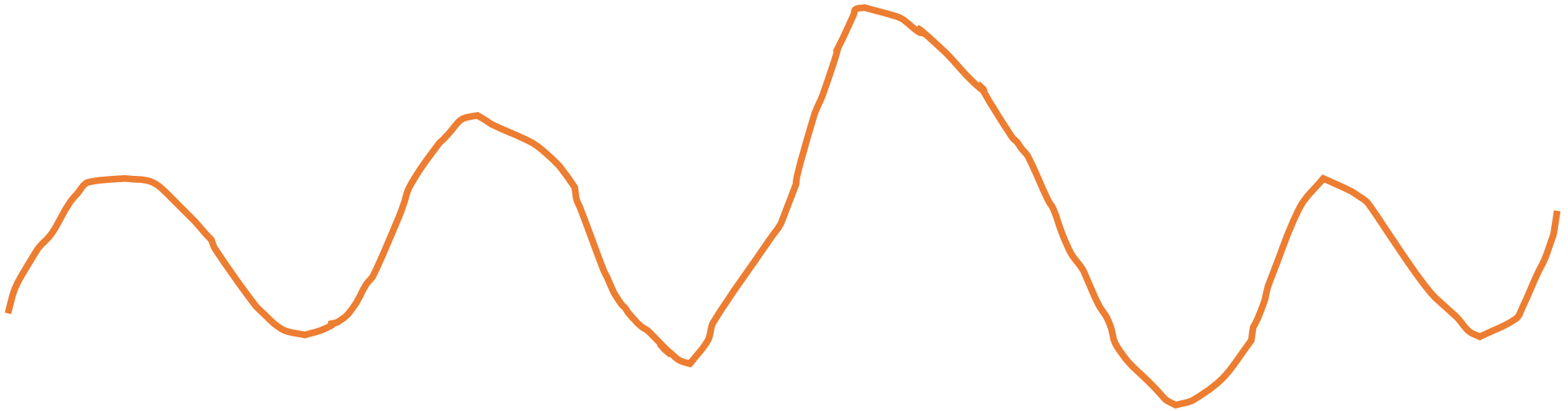

x = 20
gd = 30
Learning rate = 0.9
iter=1
f(x) = x^2 - 10x + 1
x


x = 20
gd = 30
Learning rate = 1
iter=1
f(x) = x^2 - 10x + 1
x


x = 95 95
close form = 16
loss = 127396
Learning rate = 0.01
iter=1
x 10^4
15
10
5
0
-5
f(x) = 5x1^2 + 2x2^2 + 7x1 x2 + 10x1 + x2 1
100
0
-100
x2
-100
-50
0
50
100
x1

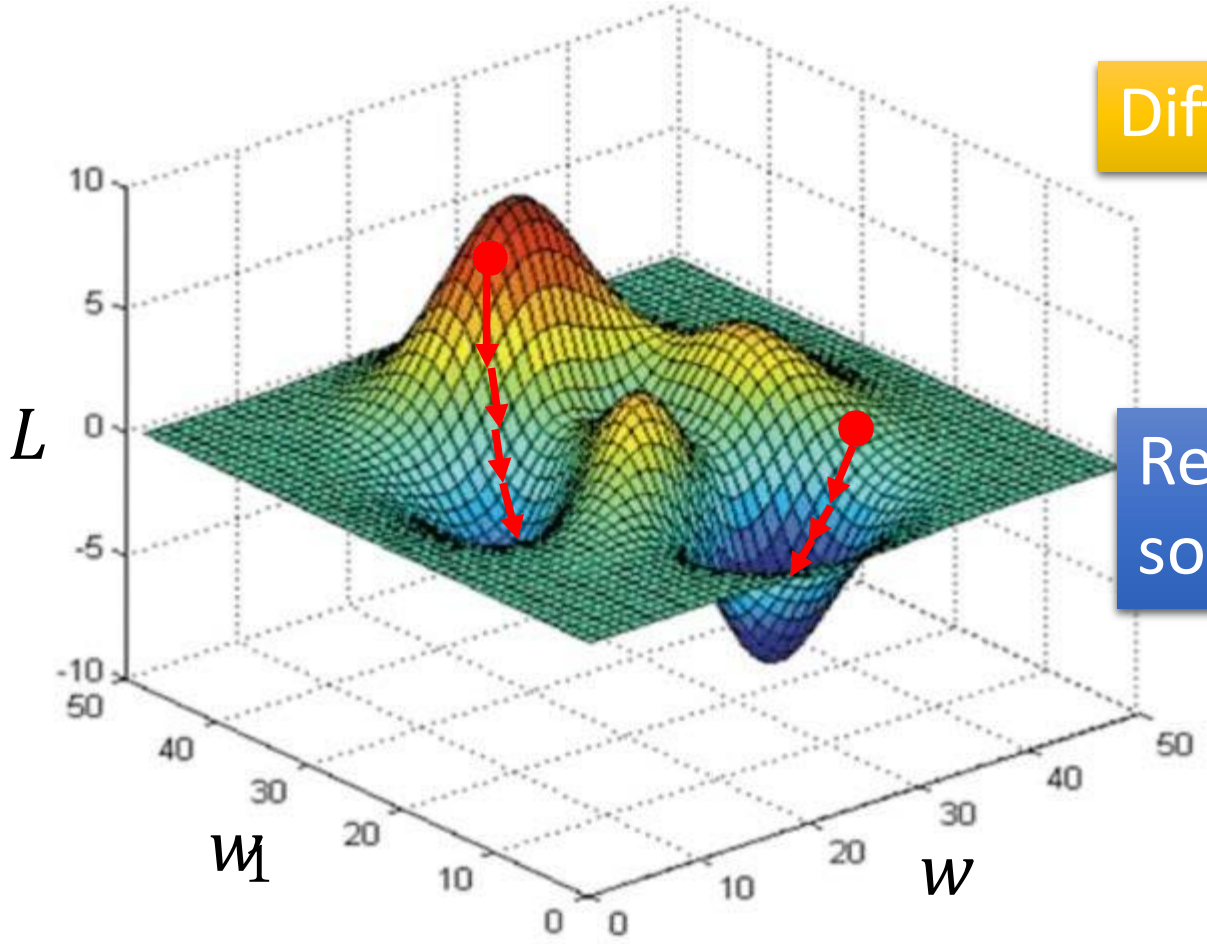
Gradient Descent

- When considering multiple parameters together, do you see any problem?



Local Minima

- Gradient descent never guarantee global minima



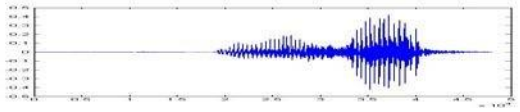
Different initial point

Reach different minima, so different results

Improve Optimization Strategy !!

Three Steps for Deep Learning

- Speech Recognition

f^* () = “你好”

- Handwritten Recognition

f^* () = “2”

- Playing Go

f^* () = “5-5” (step)

- Dialogue System

f^* (“Hi”) = “Hello”

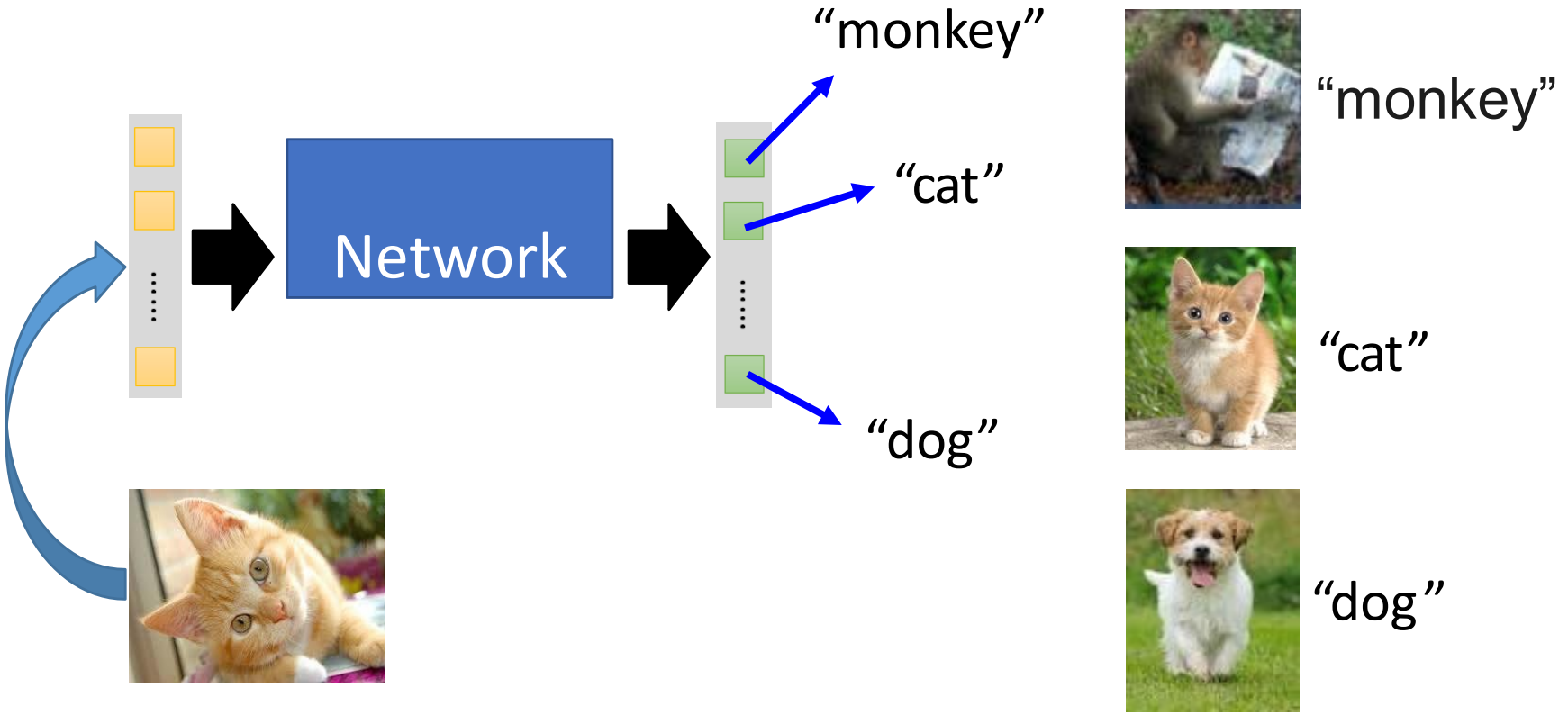
(what the user said) (system response)

Step 3: Learn!

Pick the best function f*

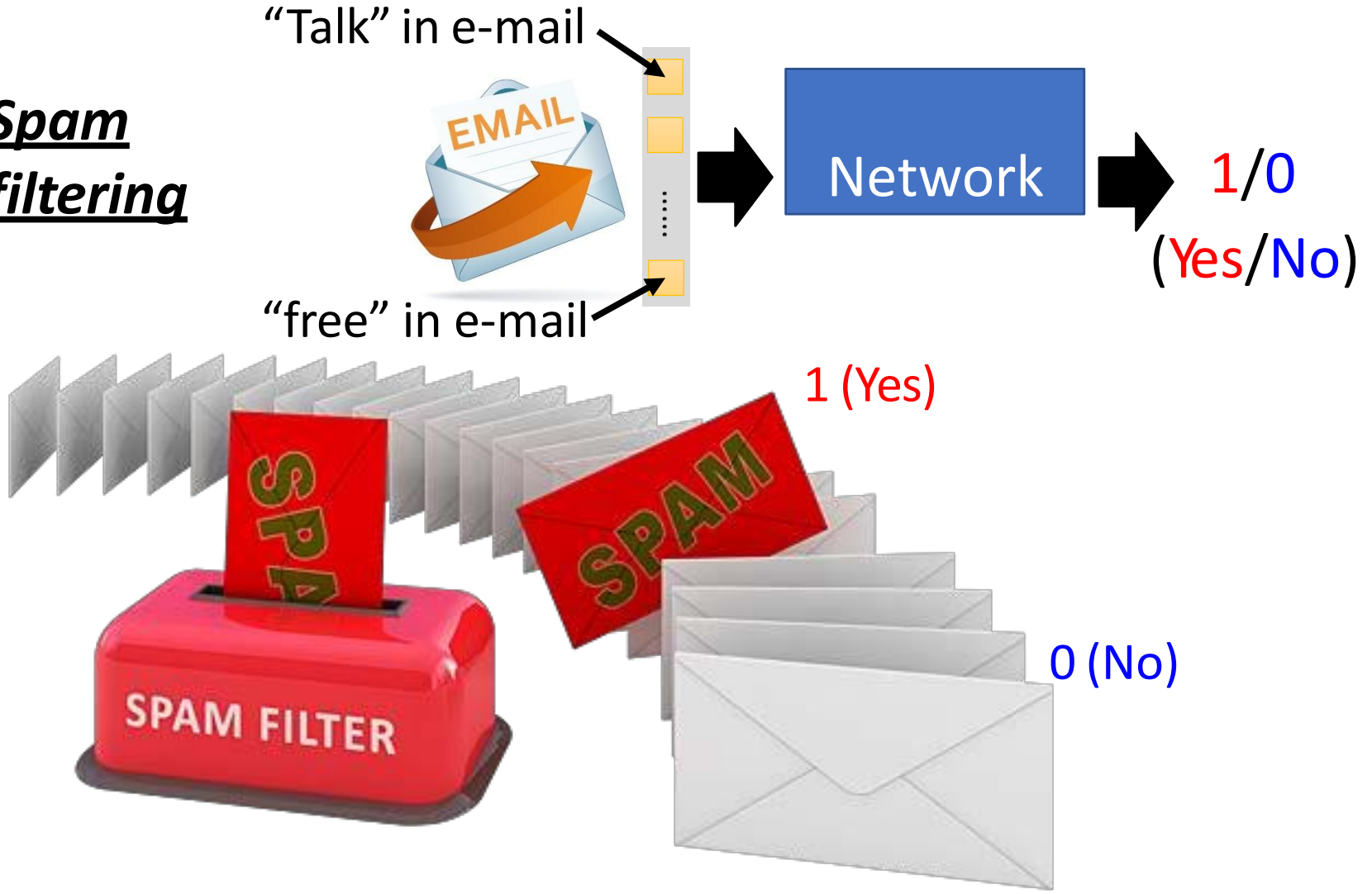
For example, you can do ……

- Image Recognition

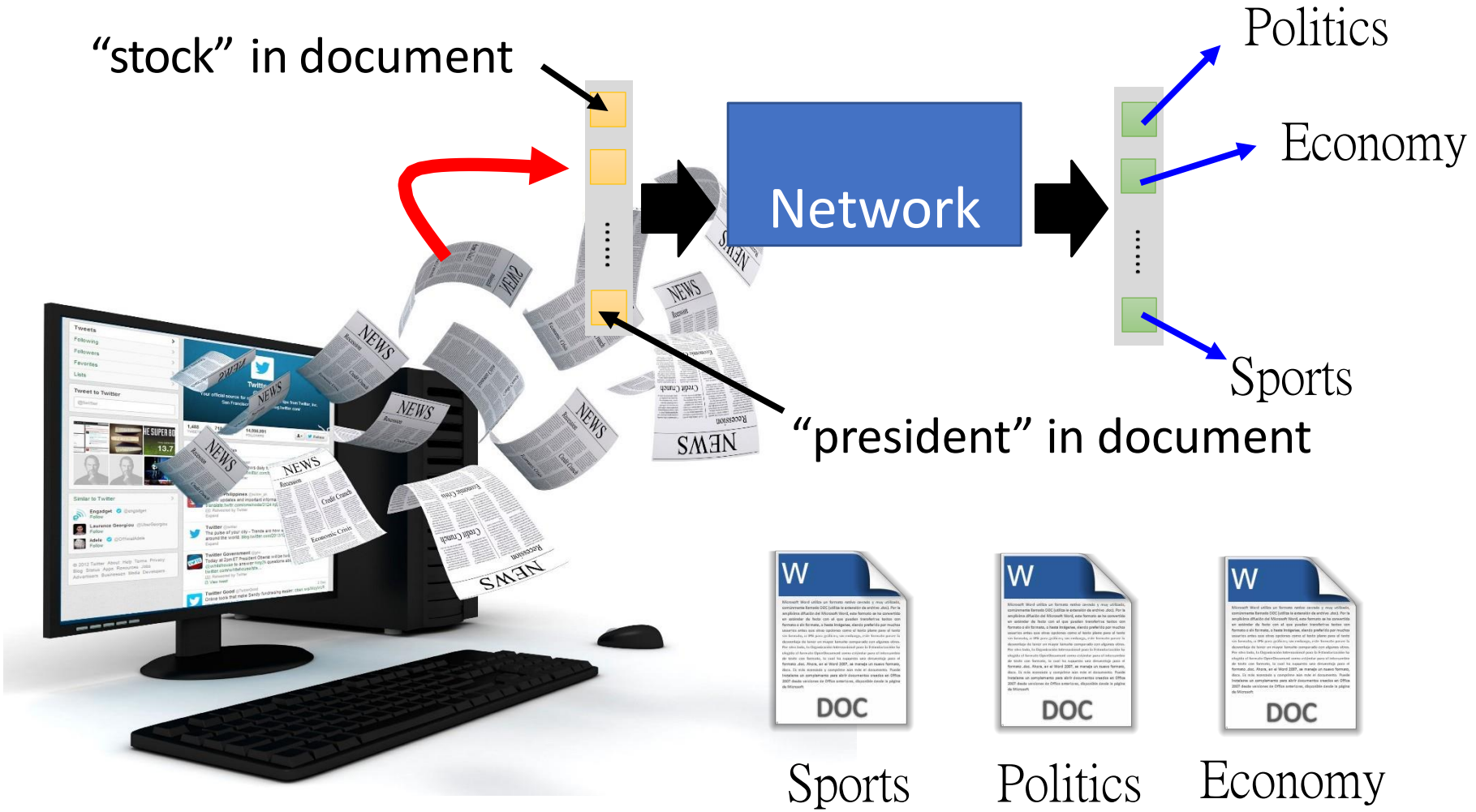


For example, you can do ……

Spam filtering



Document Classification



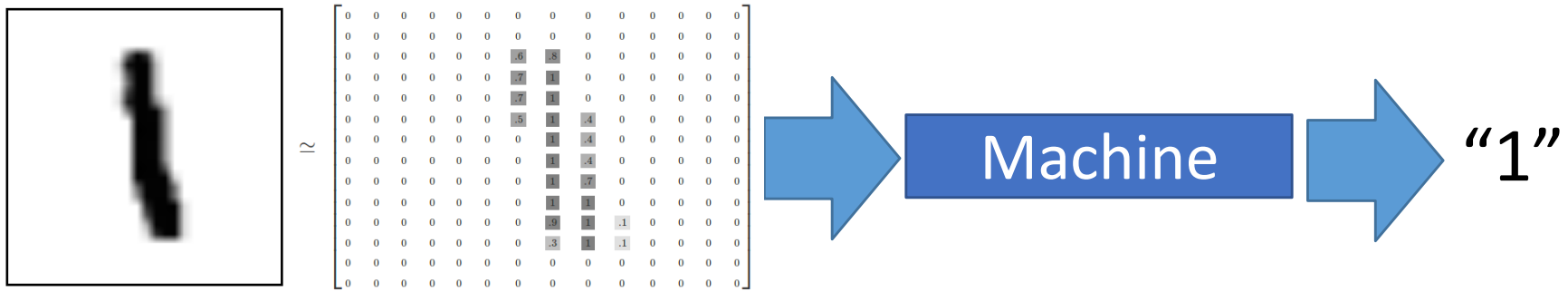
Part 5 : ‘Hello World’for Deep Learning
&
What’s more!

Handwriting Digit Recognition

Recipe of Deep Learning

Example Application

- Handwriting Digit Recognition

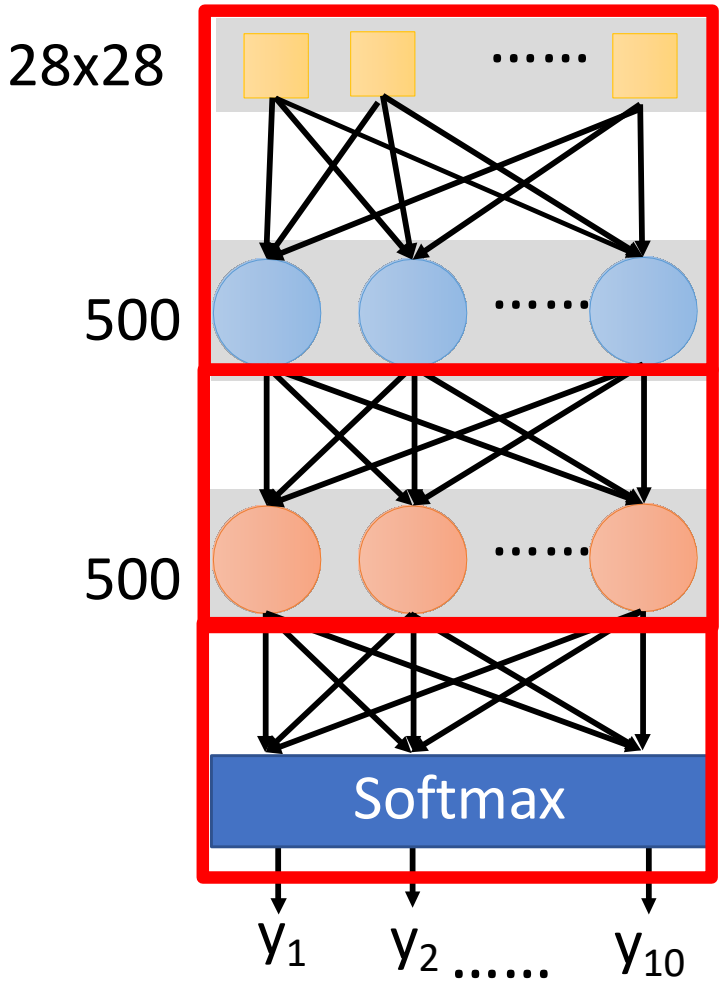


28 x 28

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

“Hello world” for deep learning

Step 1: Network Structure → Step 2: Learning Target → Step 3: Learn!



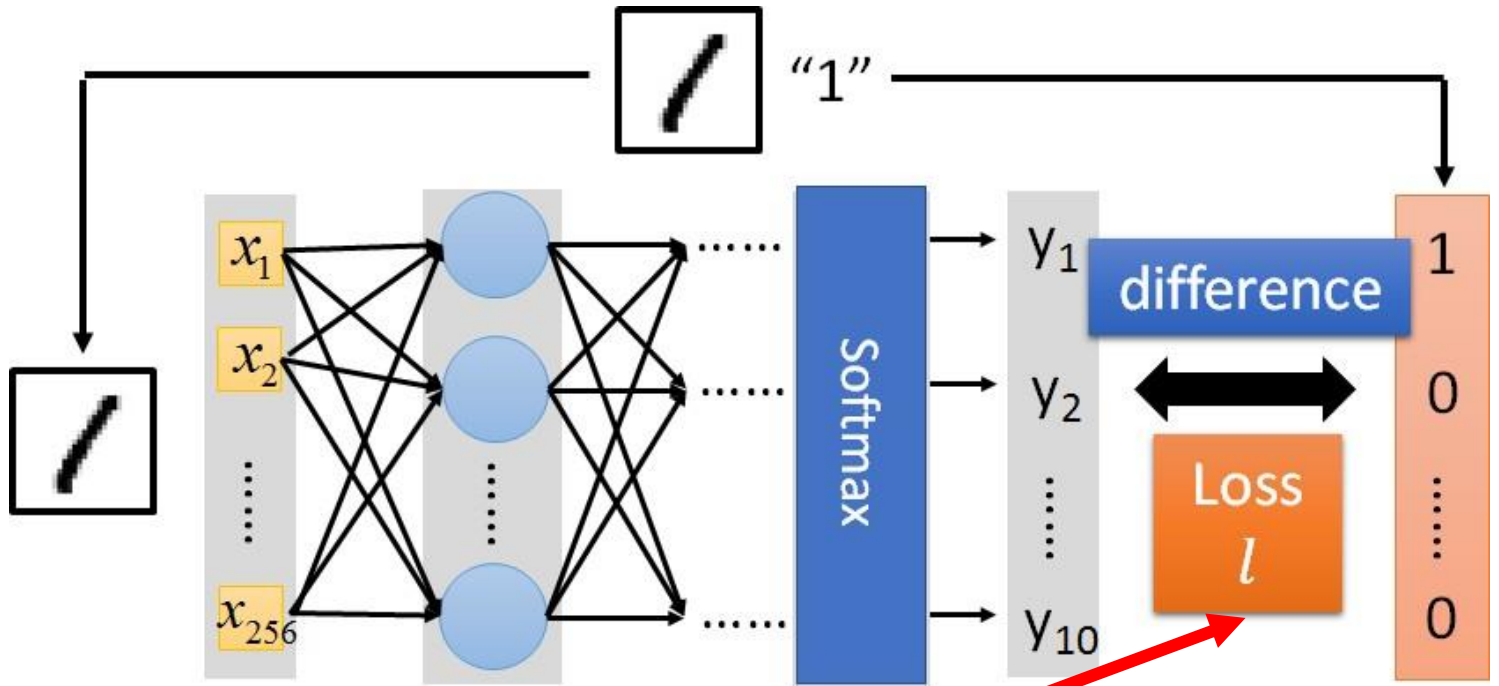

```
model = Sequential()
```

```
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') )
```

Step 1: Network Structure → Step 2: Learning Target → Step 3: Learn!




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

Step 3.1: Configuration

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

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)

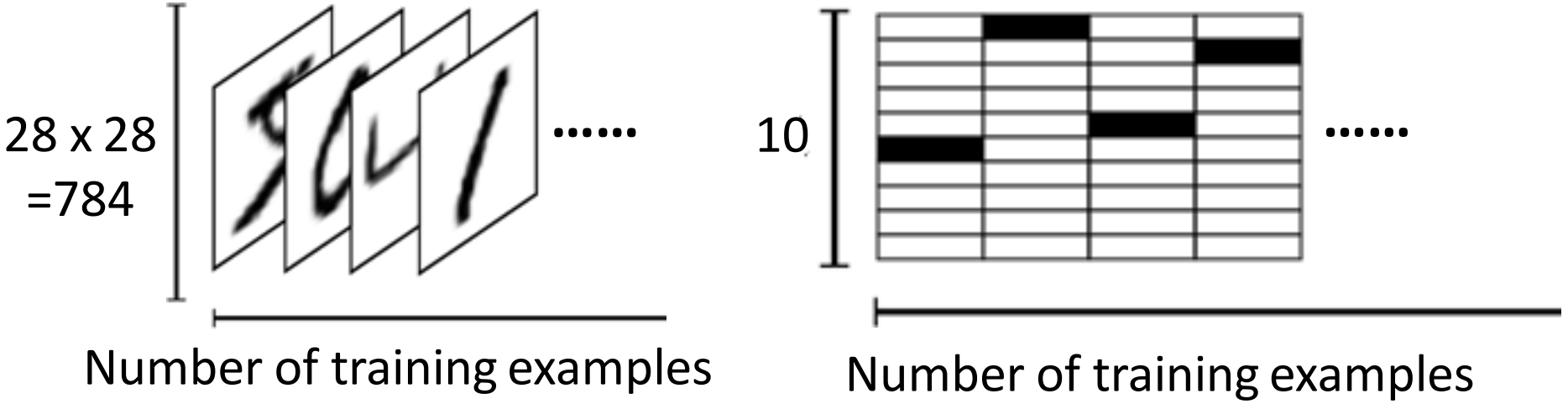
Batch_size/nb_epoch

Step 3.2: Find the optimal network parameters

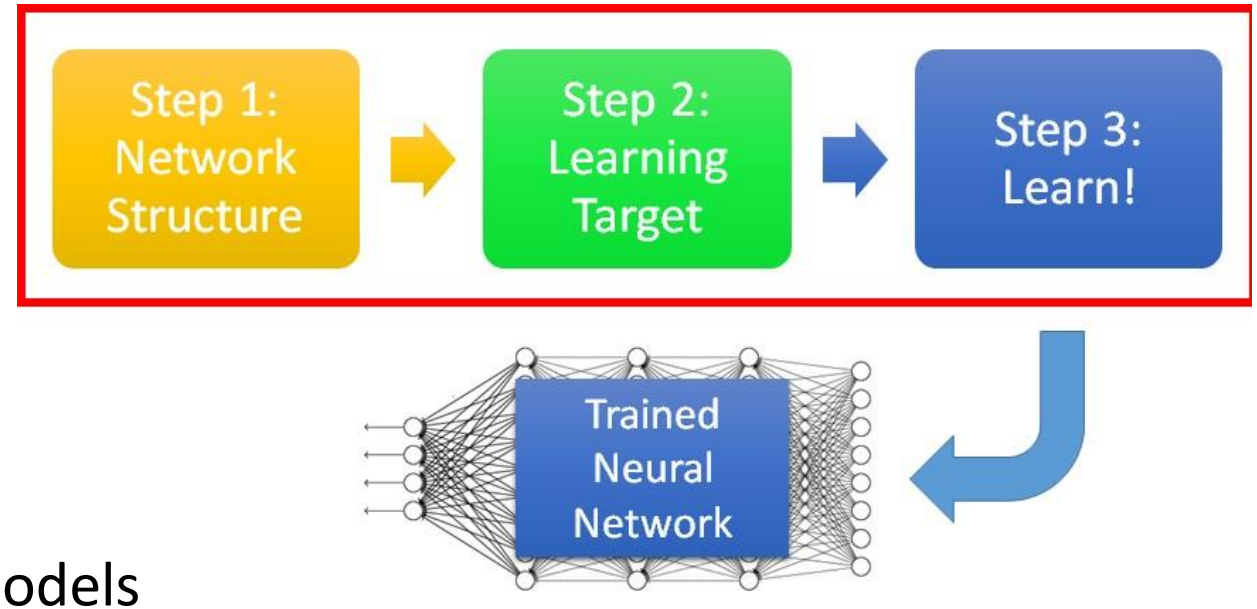
```
model.fit(x_train, y_train, batch_size=100, nb_epoch=20)
```

numpy array

numpy array



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



Save and load models

How to use the neural network (testing):

case 1:

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

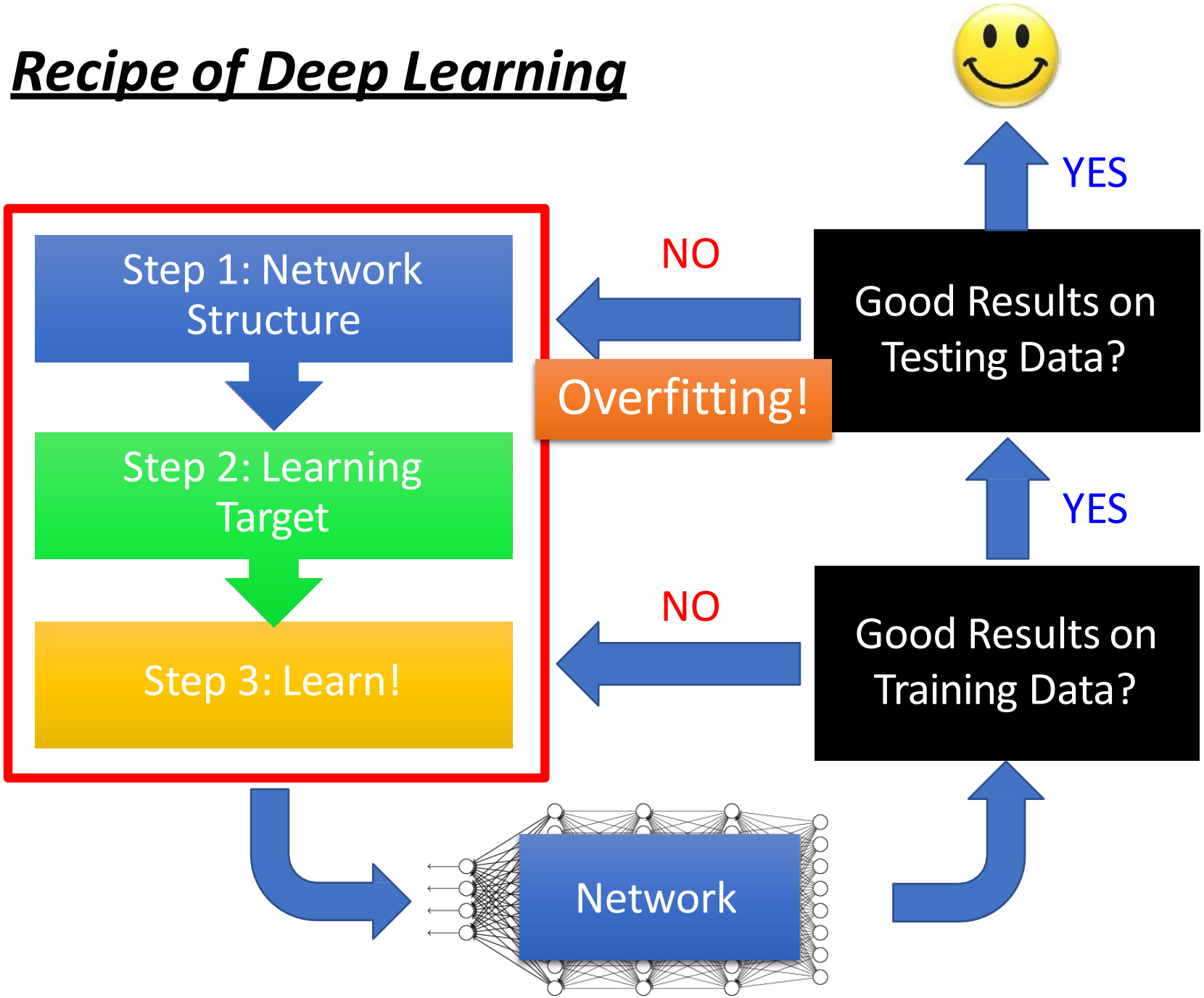
case 2:

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

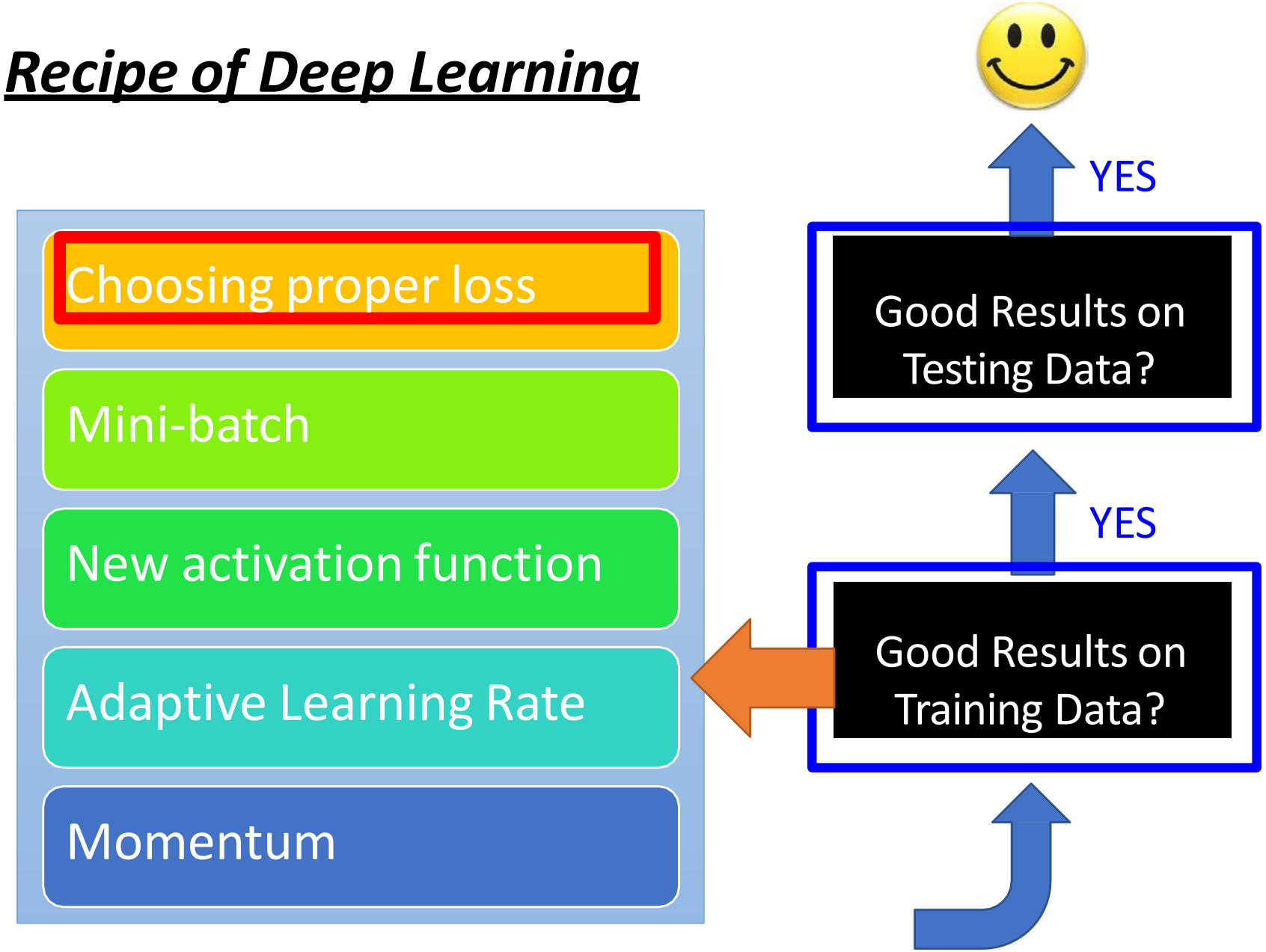
“Hello World” for Deep Learning

Recipe of Deep Learning

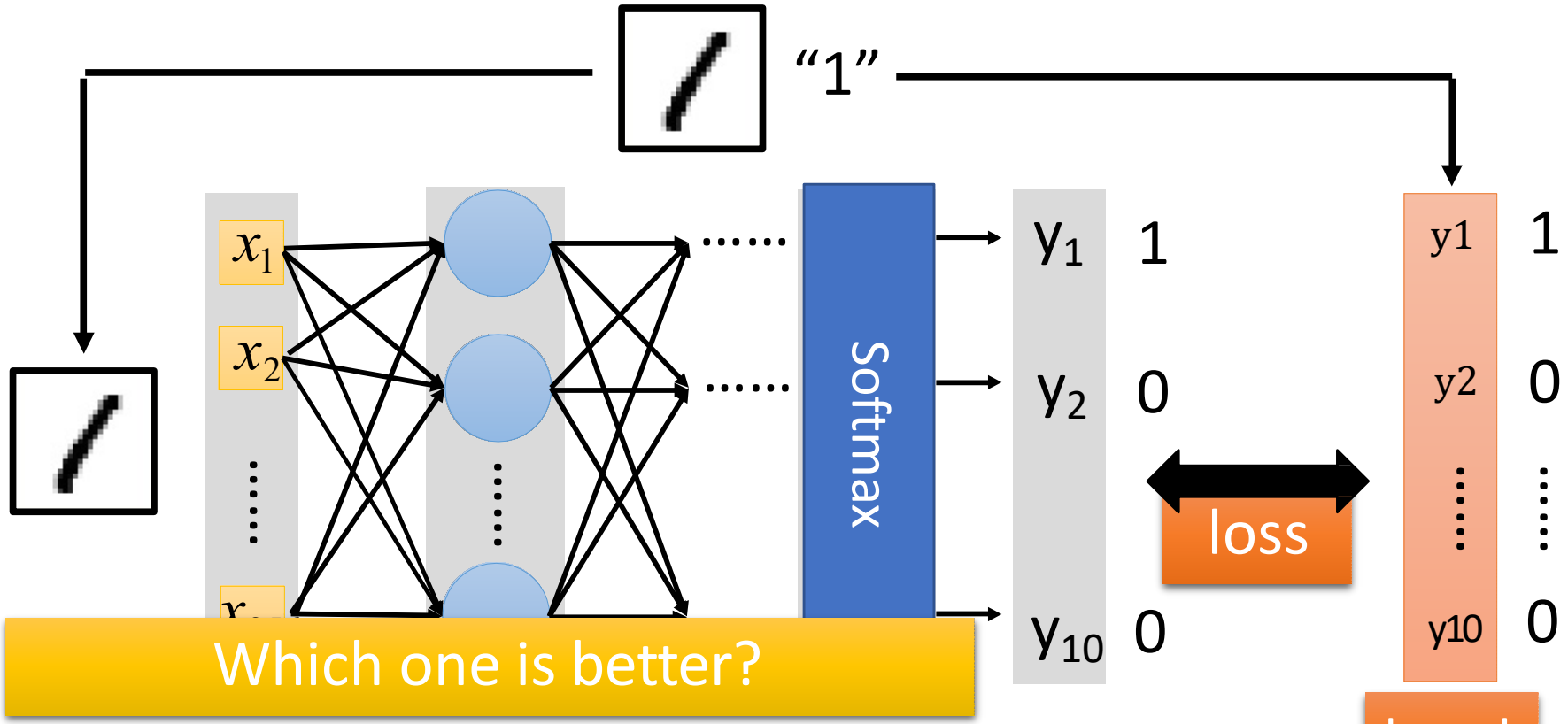
Recipe of Deep Learning



Recipe of Deep Learning



Choosing Proper Loss



Square Error

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

=0

Cross Entropy

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

=0

Let's try it

Square Error

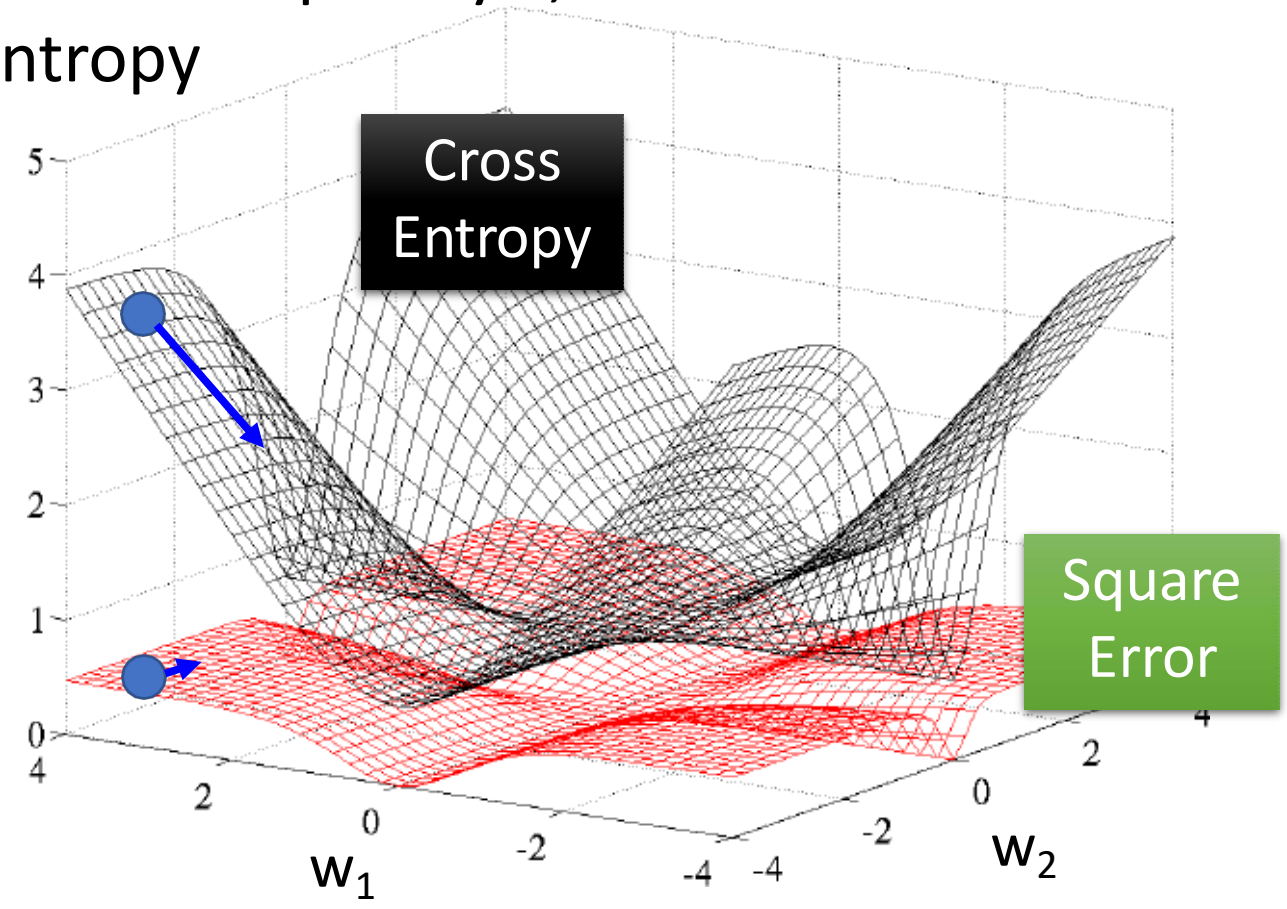
```
model.compile(loss='mse',
              optimizer=SGD(lr=0.1),
              metrics=['accuracy'])
```

Cross Entropy

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

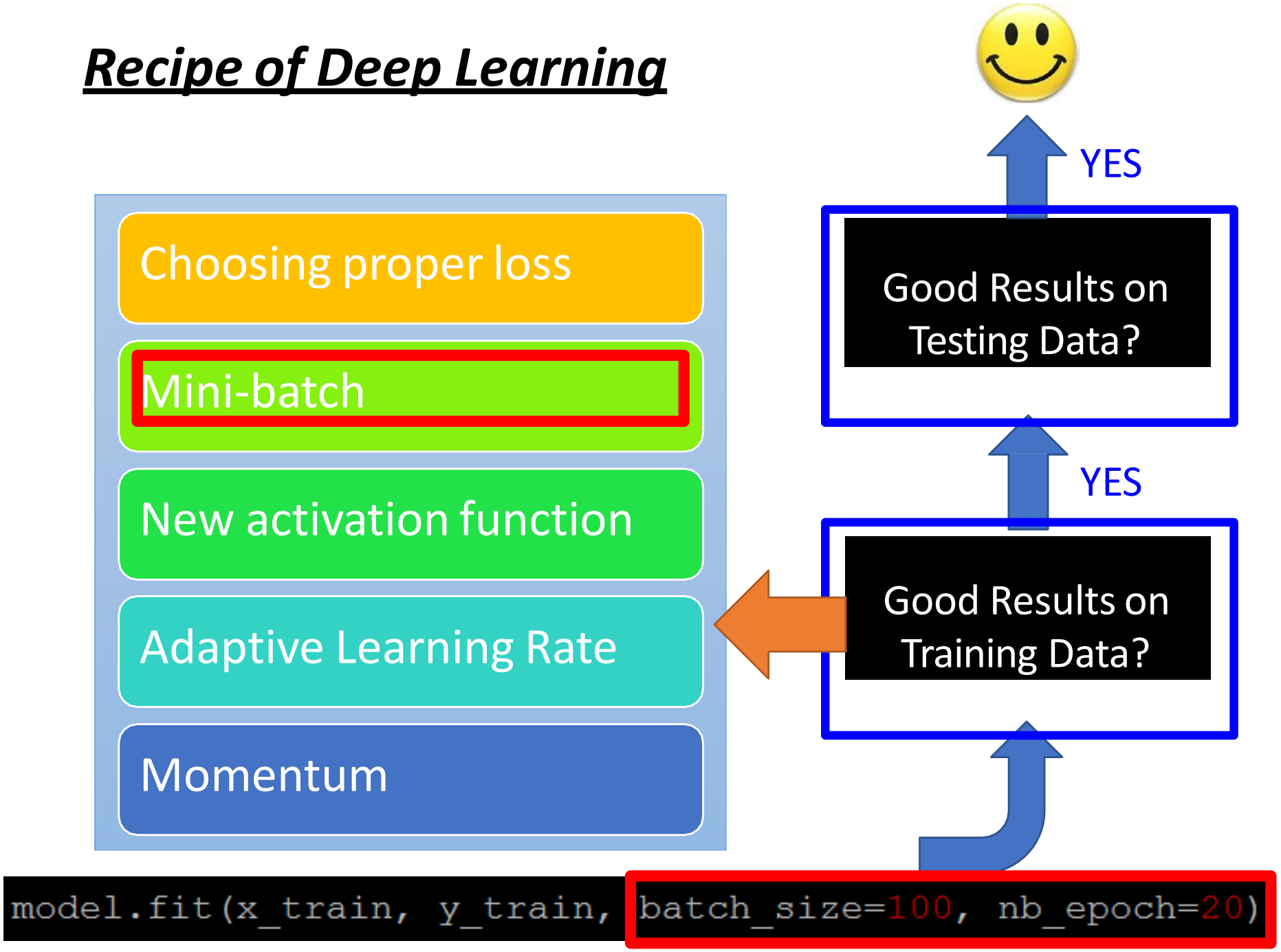
Choosing Proper Loss

When using softmax output layer, choose cross entropy



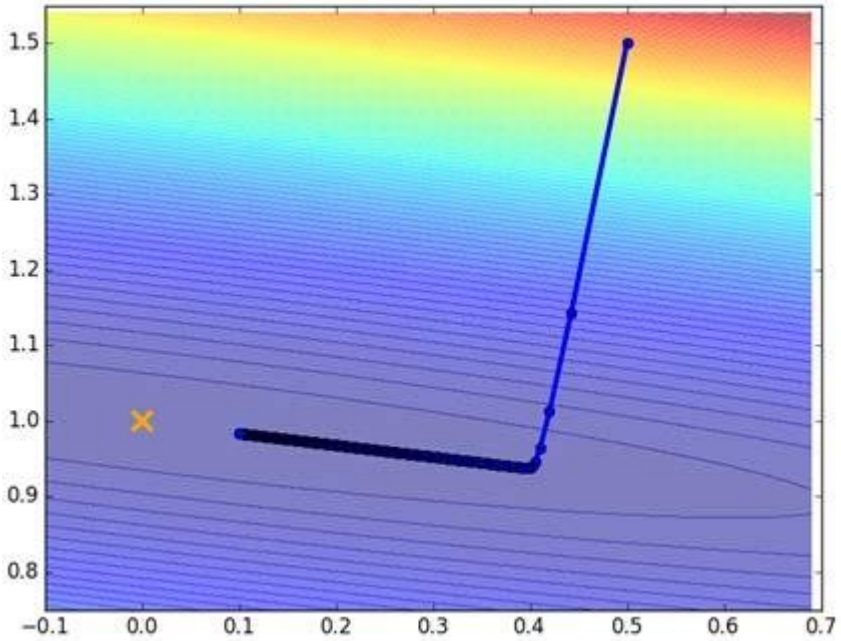
http://jmlr.org/proceedings/papers/v9/glorot10a/glorot10a.pdf

Recipe of Deep Learning

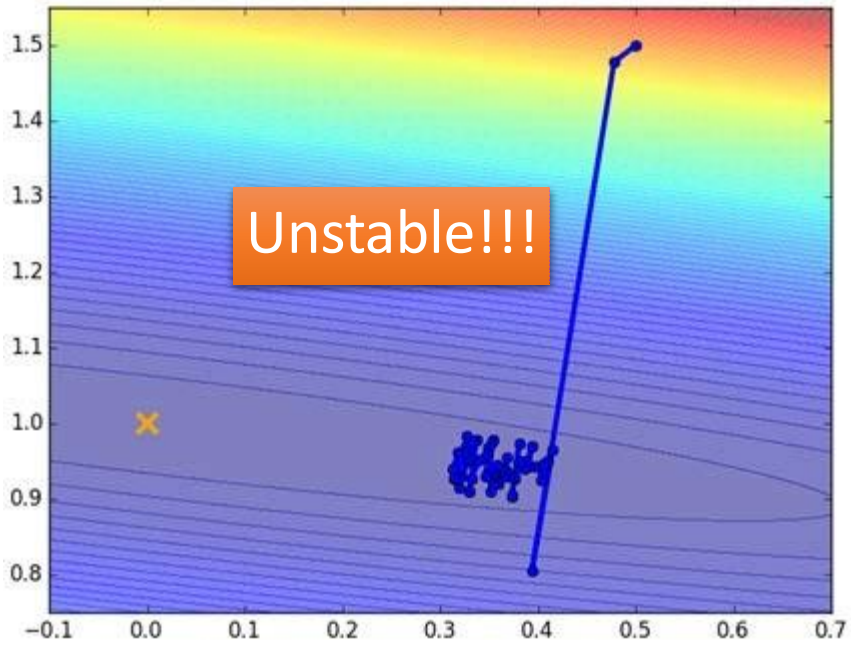


Mini-batch

Original Gradient Descent



With Mini-batch



The colors represent the total loss.

Mini-batch is Faster

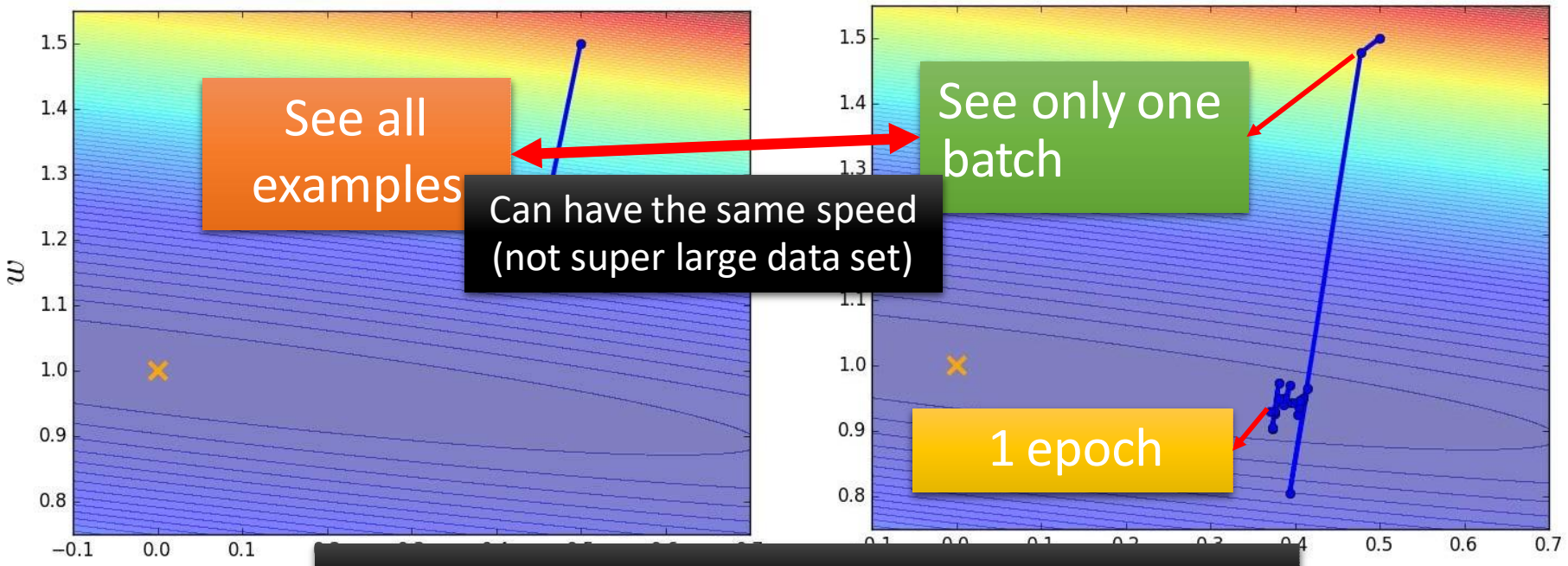
Not always true with parallel computing.

Original Gradient Descent

Update after seeing all examples

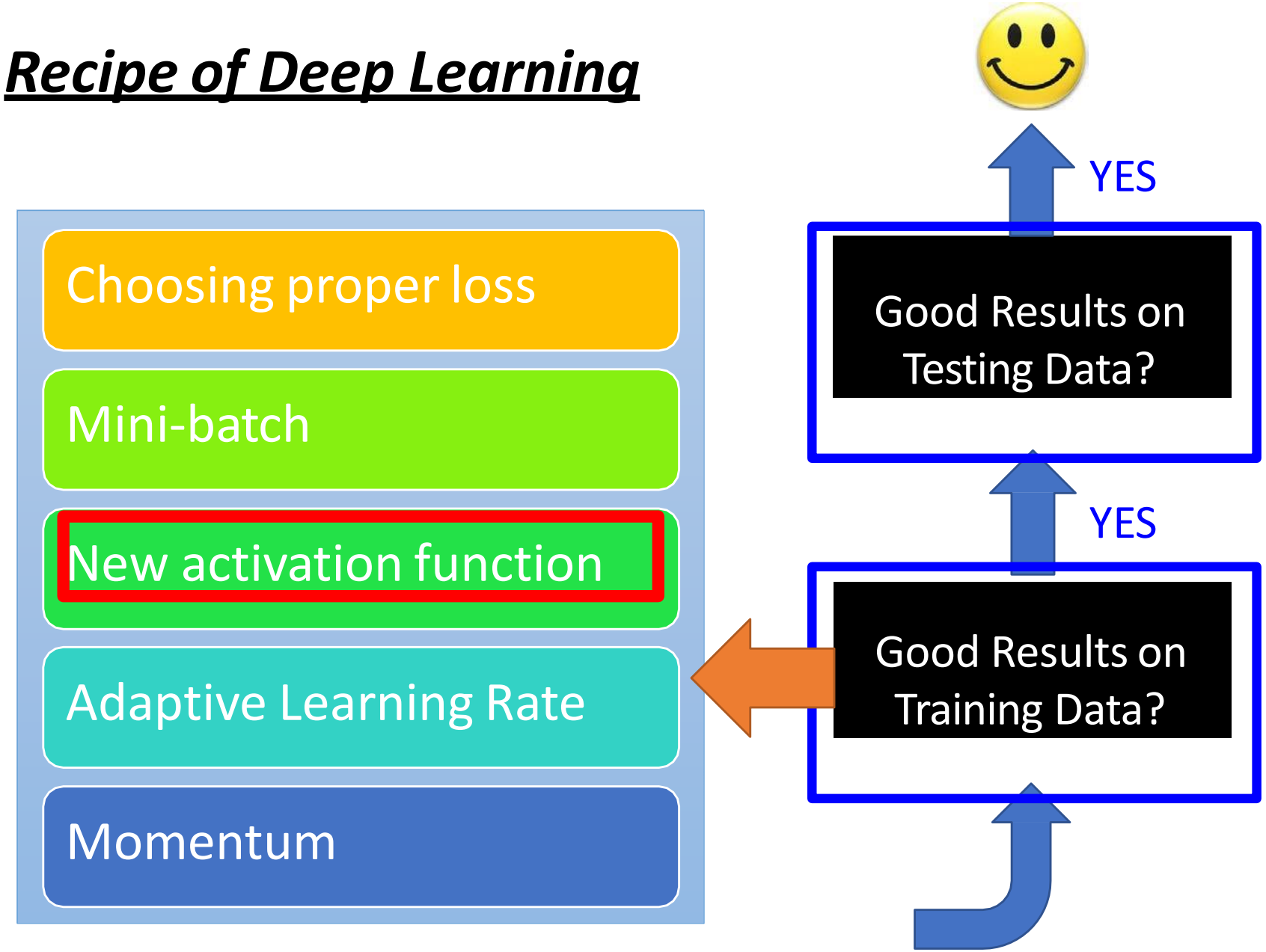
With Mini-batch

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

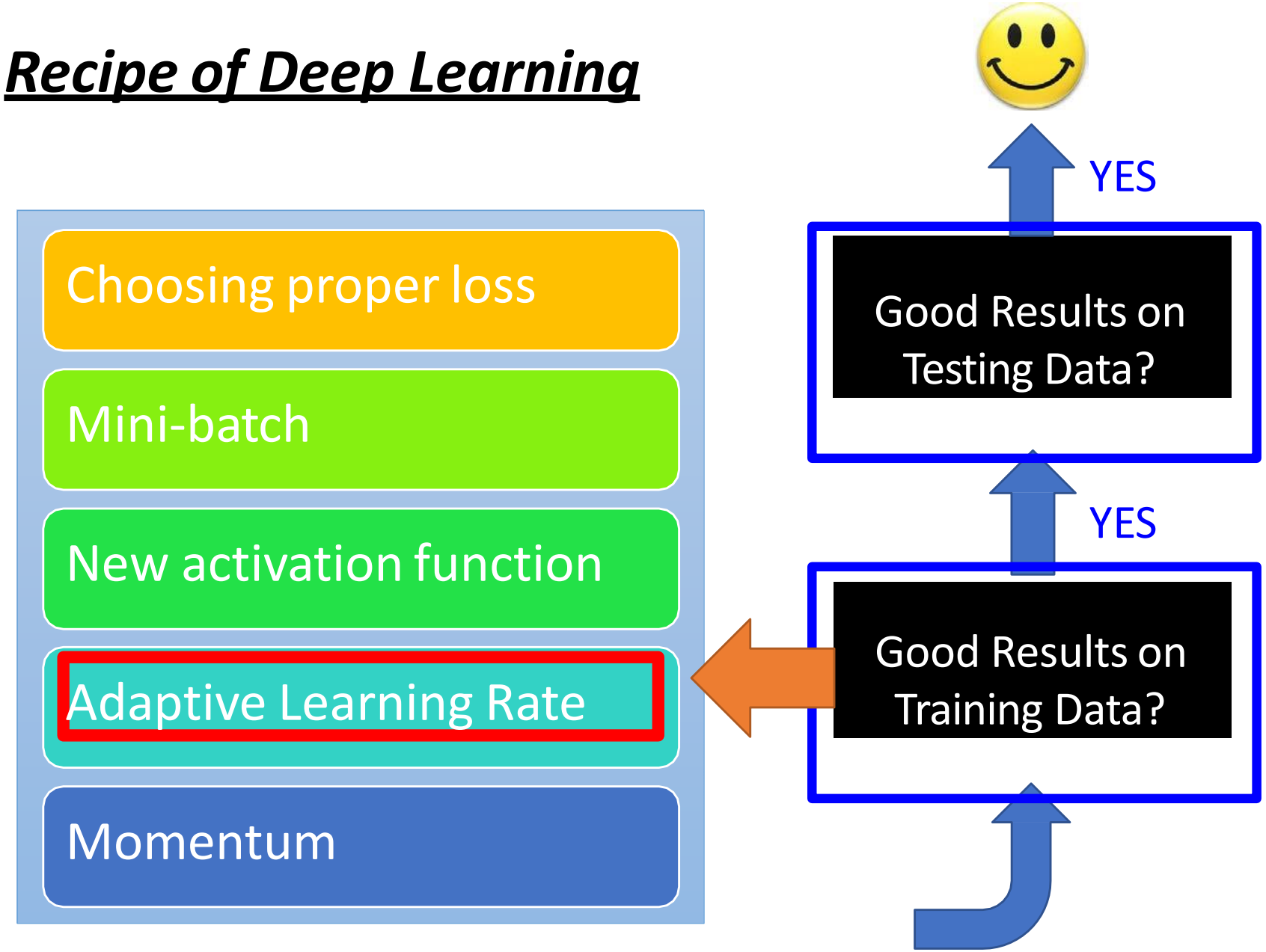


Mini-batch has better performance!

Recipe of Deep Learning

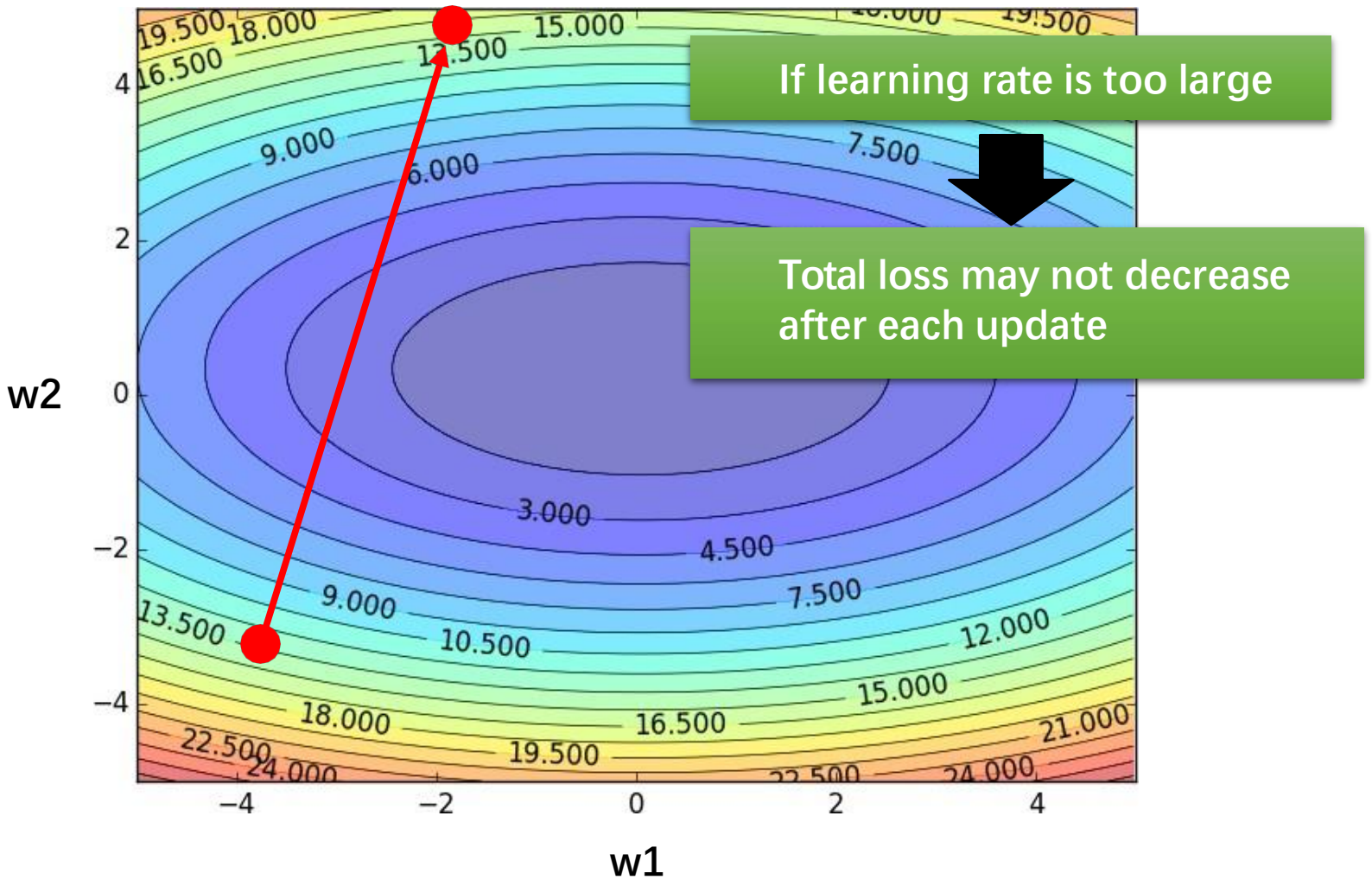


Recipe of Deep Learning



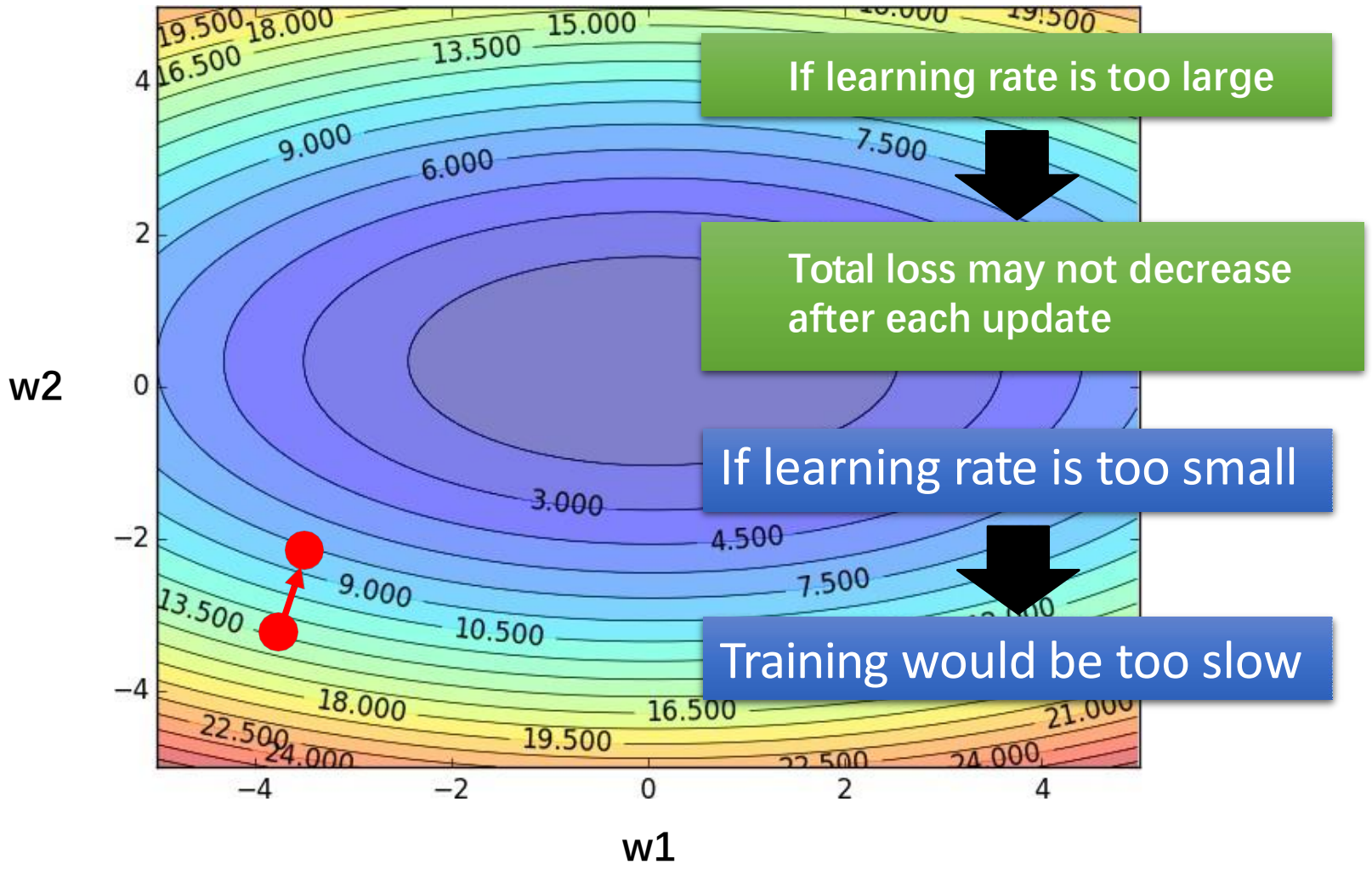
Learning Rates

Set the learning rate η carefully



Learning Rates

Set the learning rate η carefully



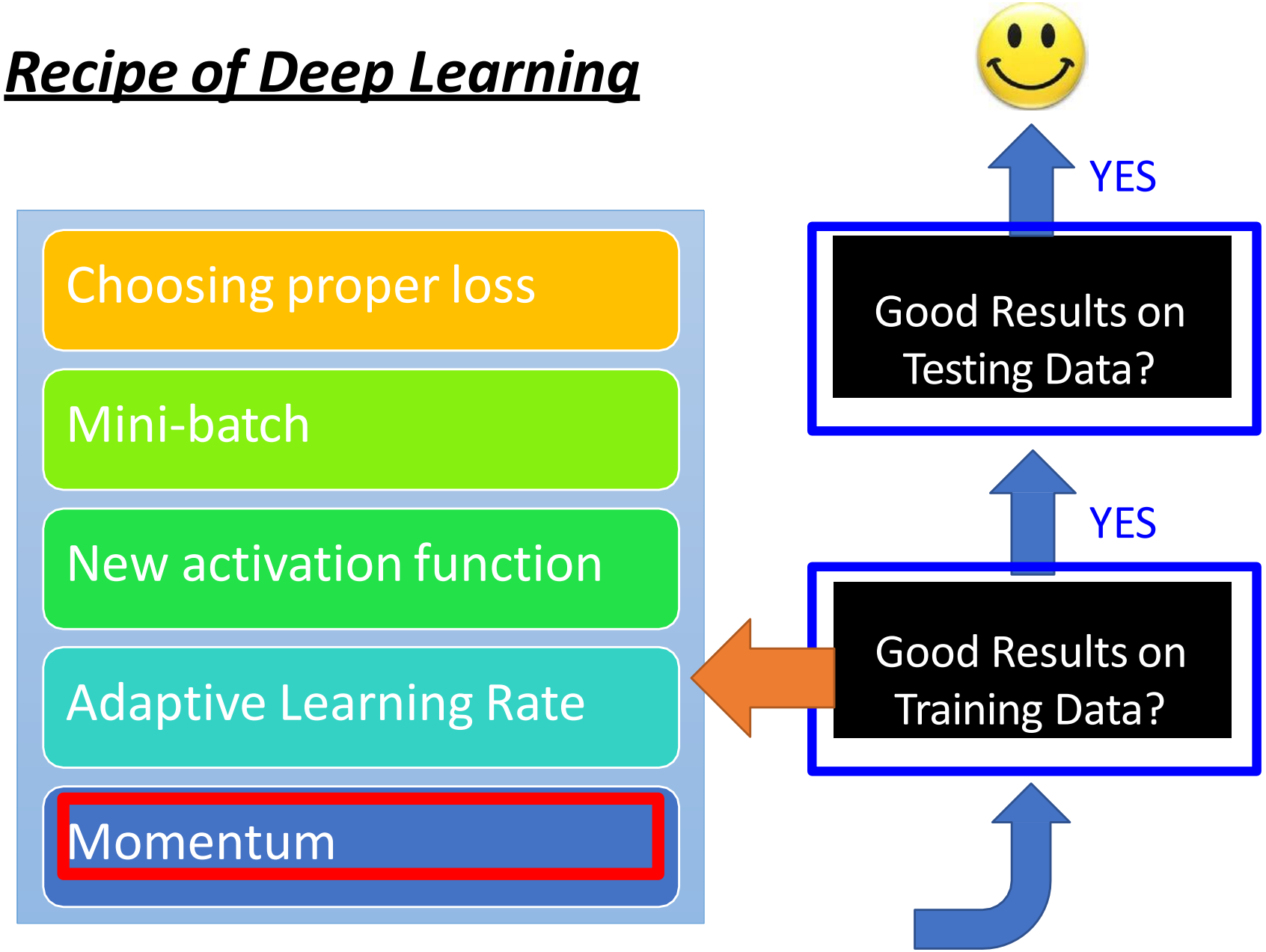
If learning rate is too large

Total loss may not decrease after each update

If learning rate is too small

Training would be too slow

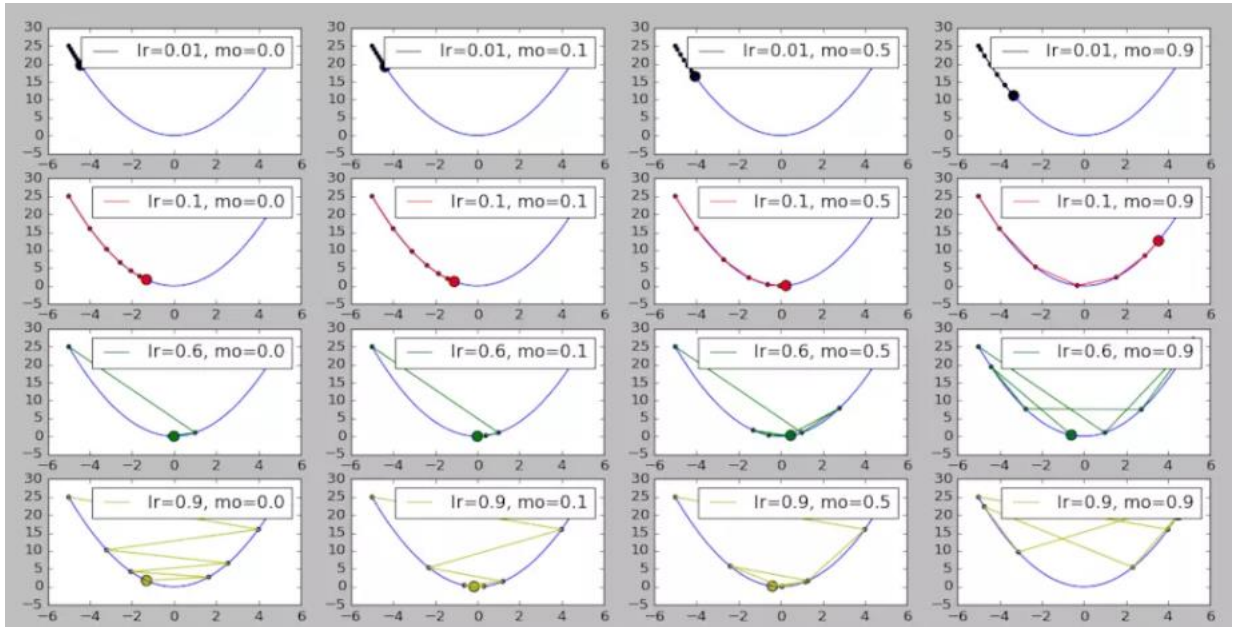
Recipe of Deep Learning



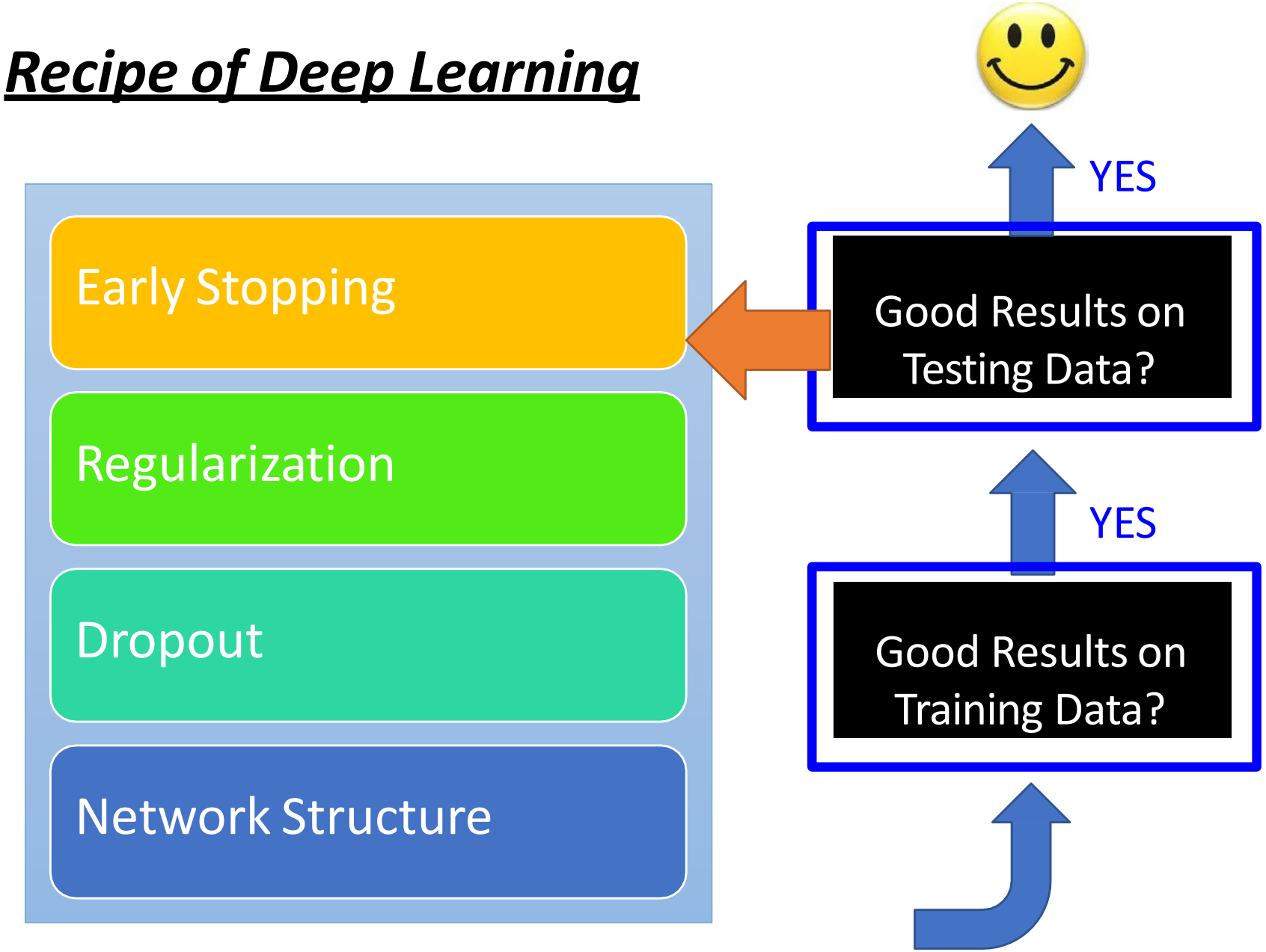
Momentum

- Momentum is a method which helps accelerate gradients vectors in the right directions, thus leading to faster converging. It is one of the most popular optimization algorithms and many state-of-the-art models are trained using it.

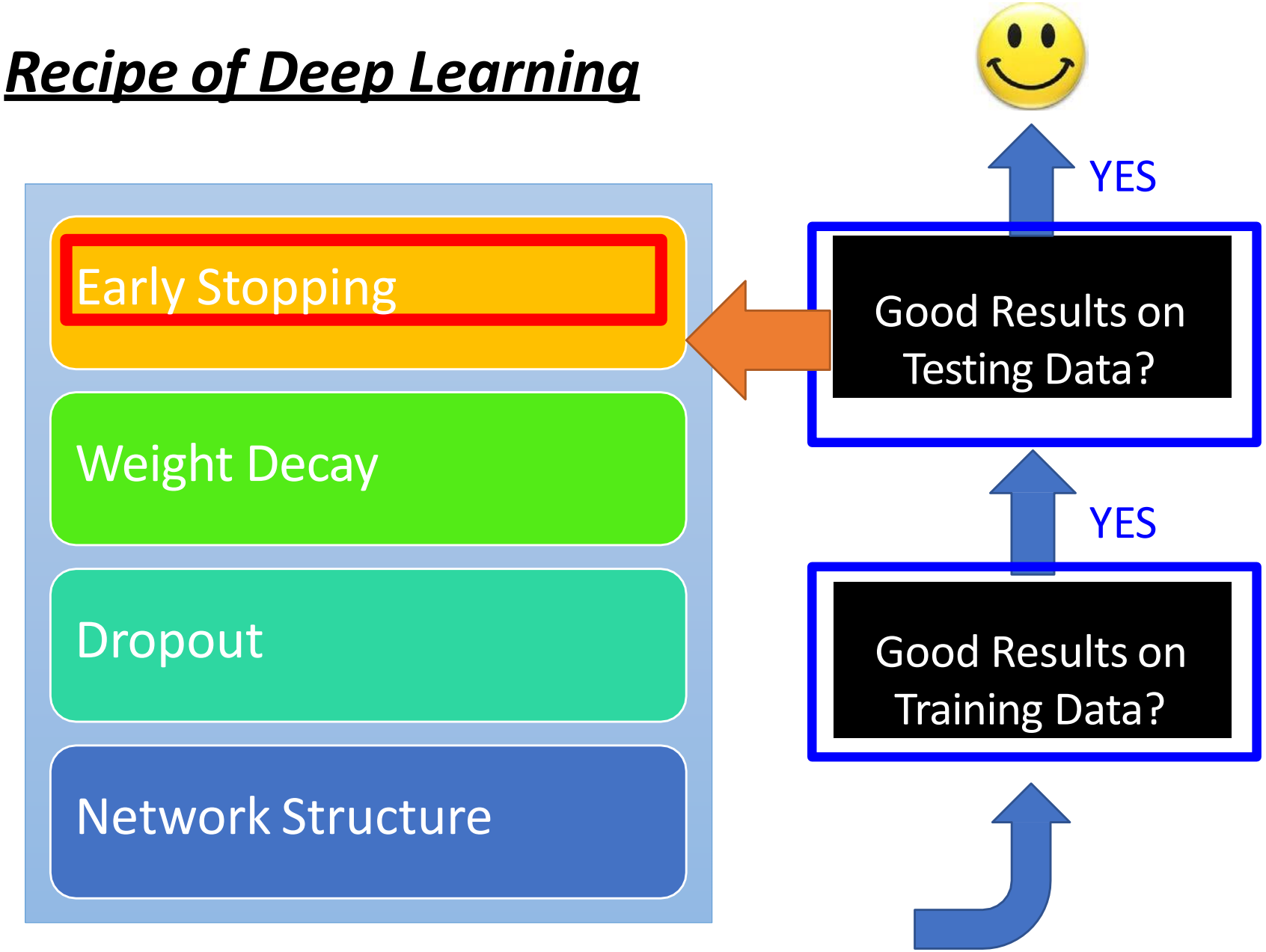
```python
def GD_momentum(x_start, df, epochs, lr, momentum):
    """
    带有冲量的梯度下降法。
    :param x_start: x的起始点
    :param df: 目标函数的一阶导函数
    :param epochs: 迭代周期
    :param lr: 学习率
    :param momentum: 冲量
    :return: x在每次迭代后的位置（包括起始点），长度为epochs+1
    """
    xs = np.zeros(epochs+1)
    x = x_start
    xs[0] = x
    v = 0
    for i in range(epochs):
        dx = df(x)
        # v表示x要改变的幅度
        v = - dx * lr + momentum * v
        x += v
        xs[i+1] = x
    return xs
```



Recipe of Deep Learning



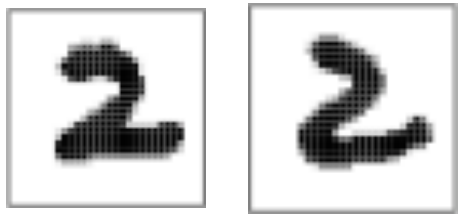
Recipe of Deep Learning



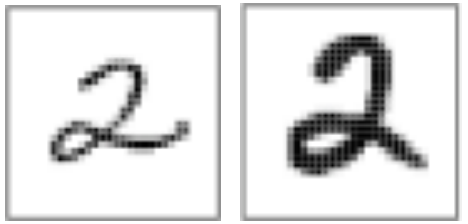
Why Overfitting?

- Training data and testing data can be different.

Training Data:



Testing Data:

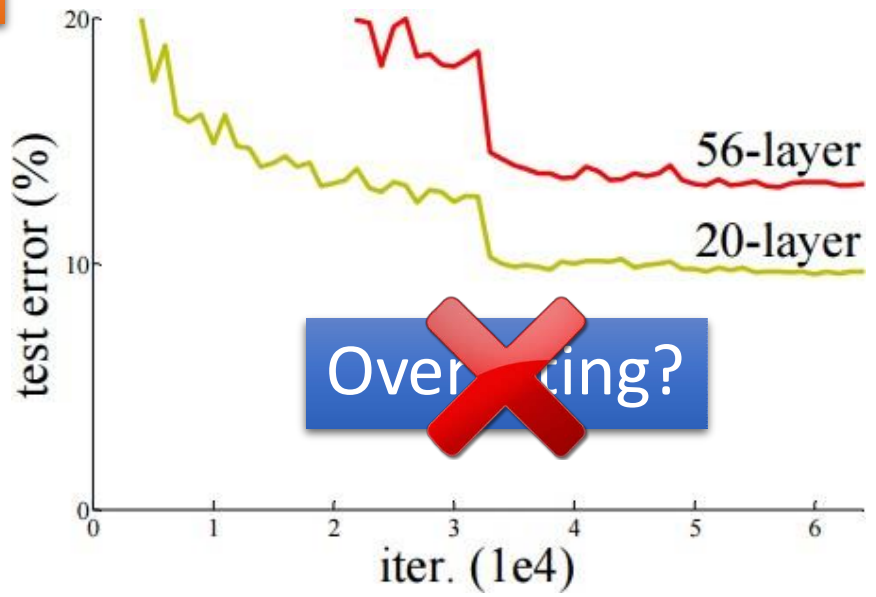


Learning target is defined by the training data.

The parameters achieving the learning target do not necessary have good results on the testing data.

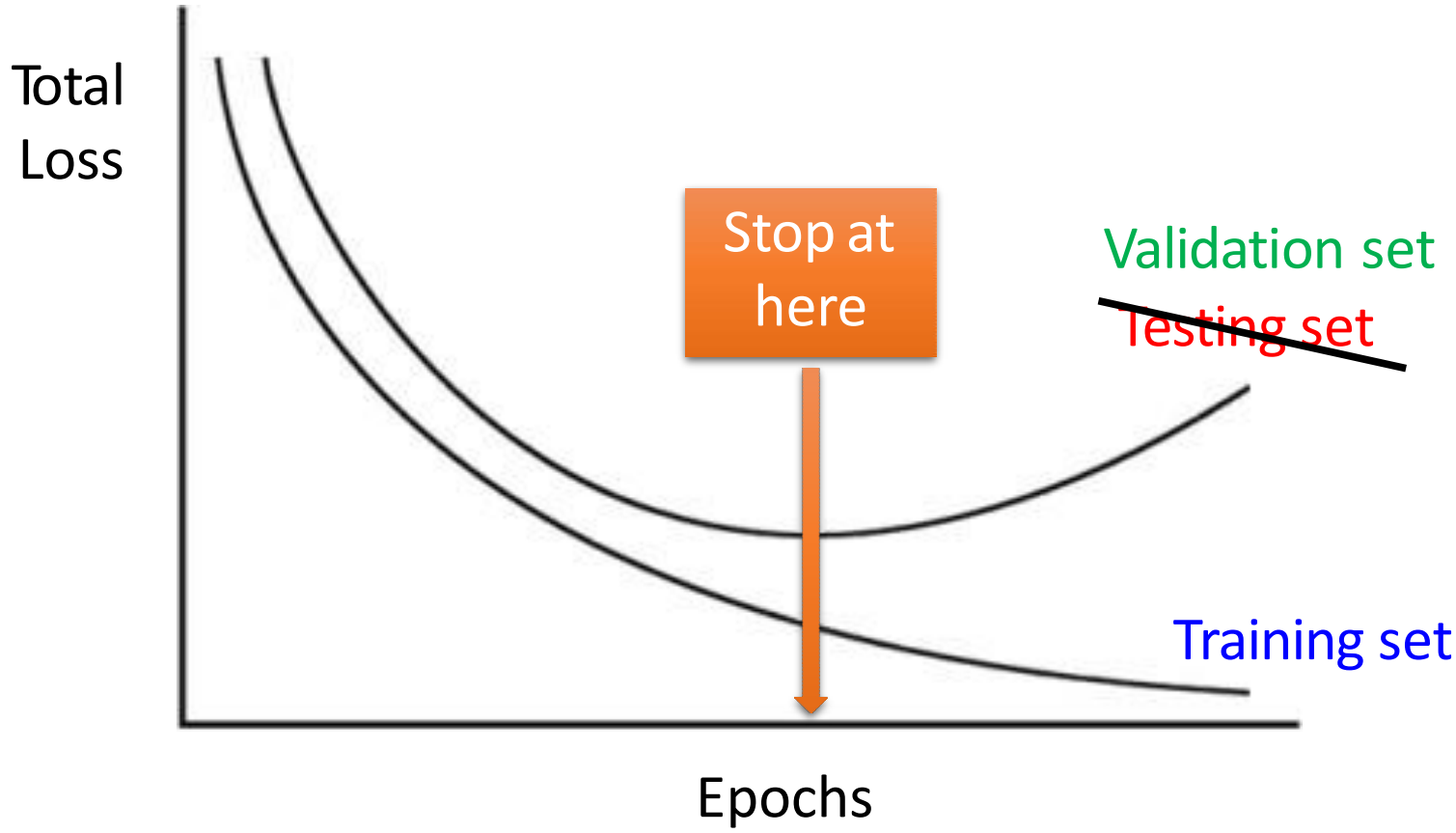
Do not always blame Overfitting

Training Data

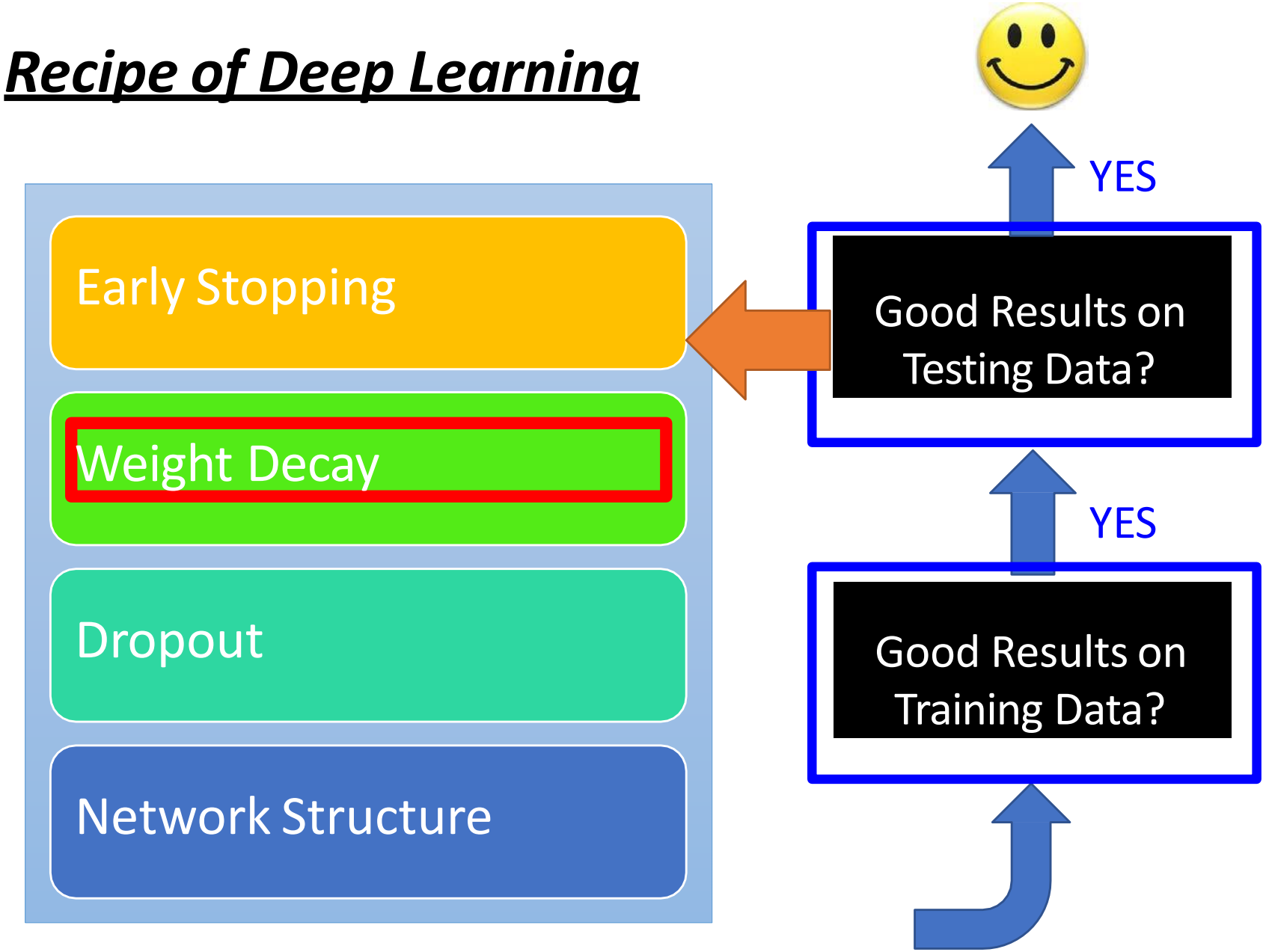


Testing Data

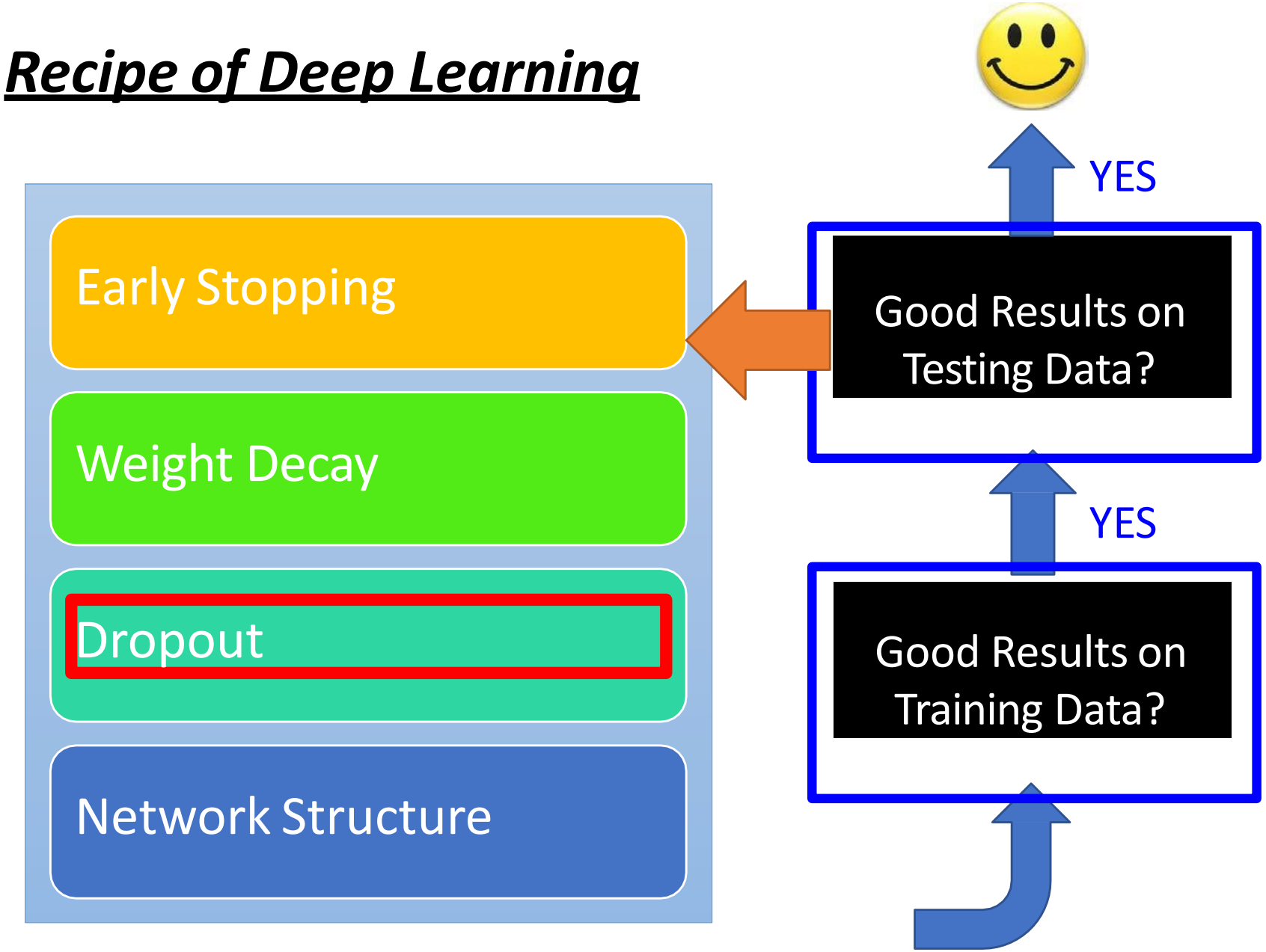
Early Stopping



Recipe of Deep Learning

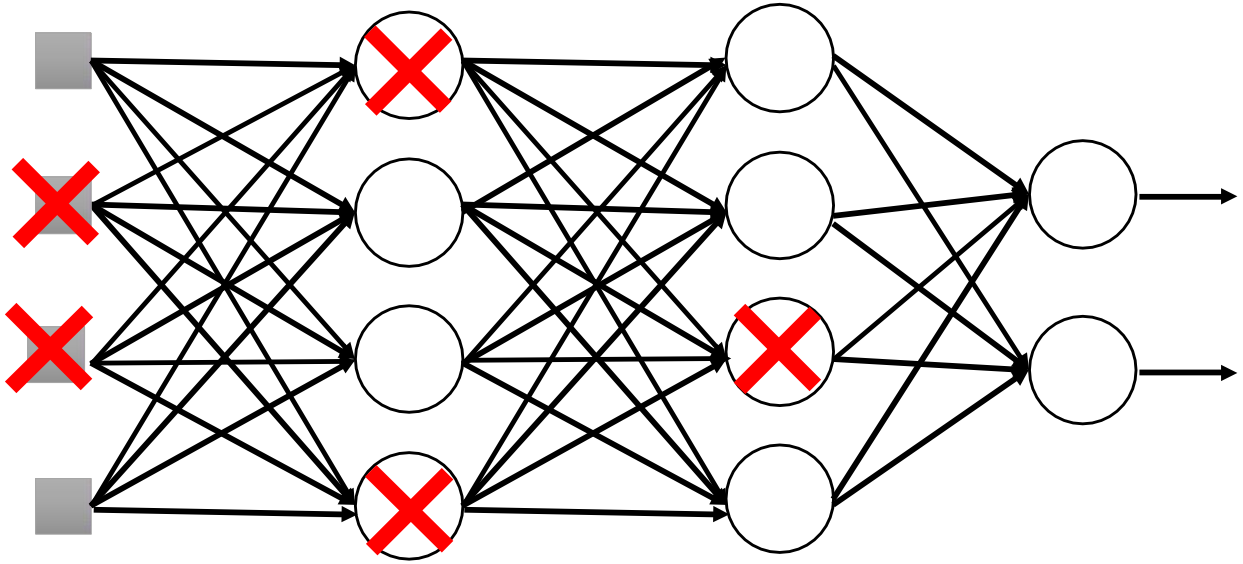


Recipe of Deep Learning



Dropout

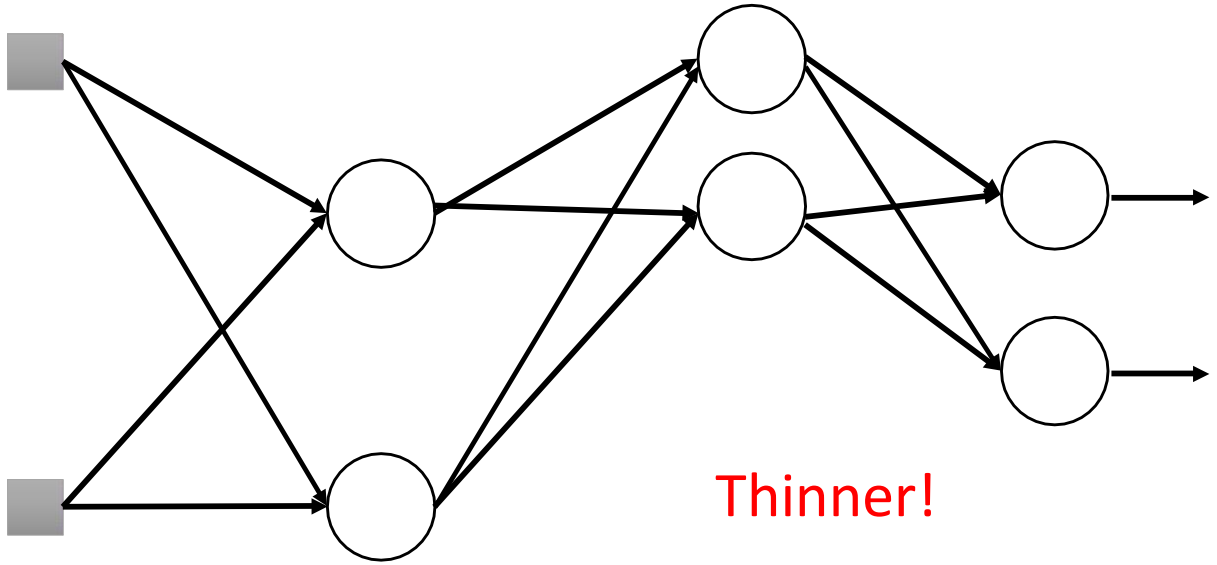
Training:



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

Dropout

Training:

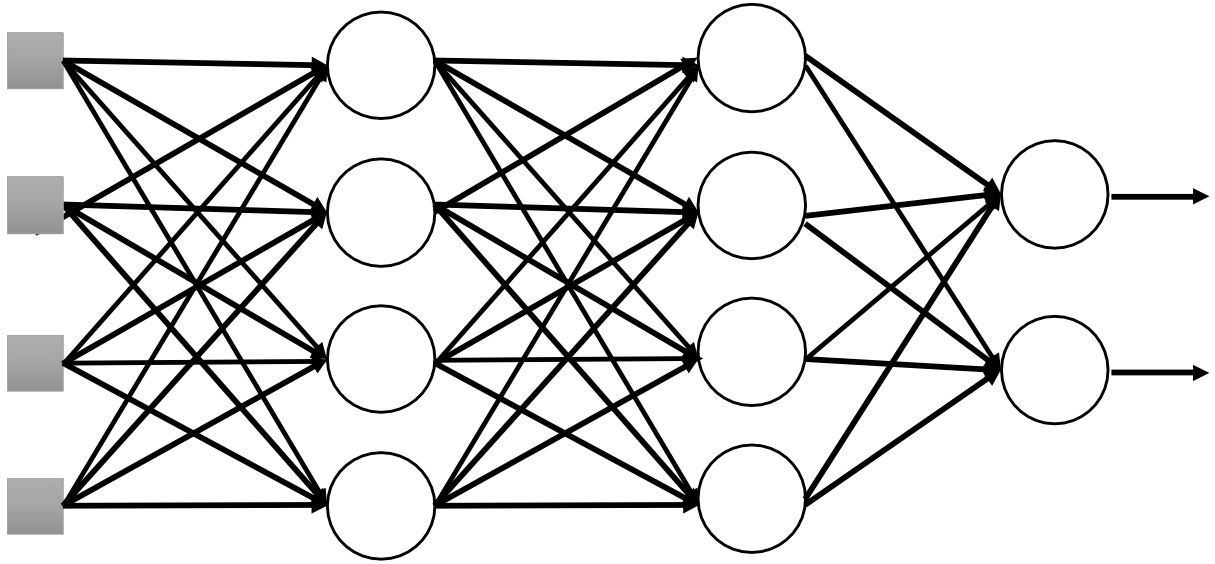


- **Each time before updating the parameters**
  - Each neuron has p% to dropout

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

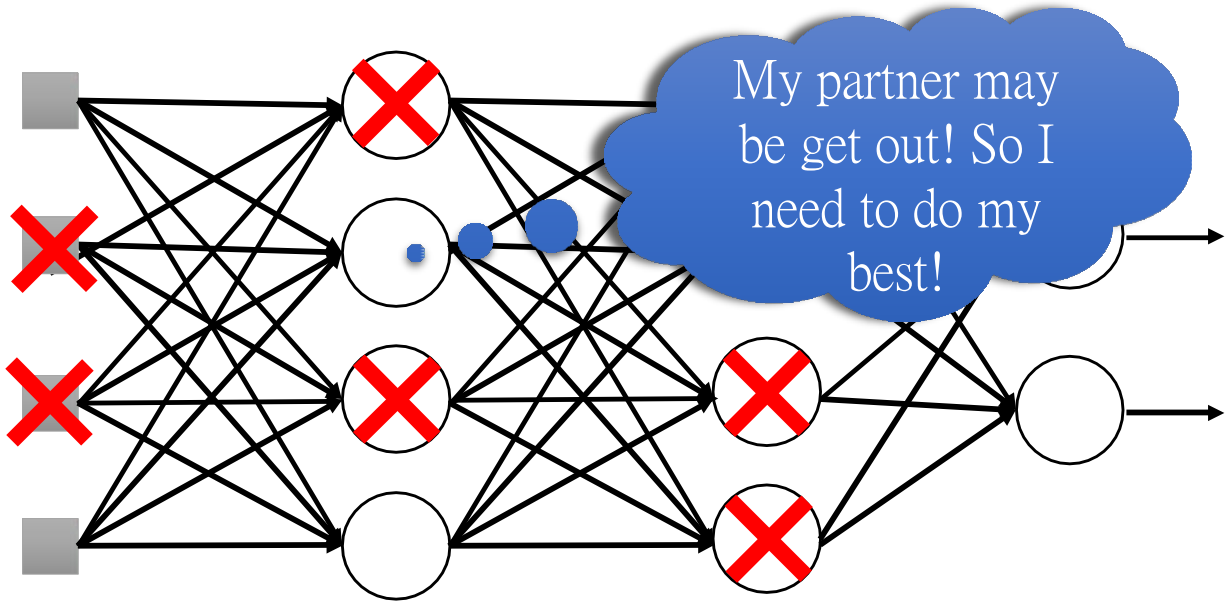
Dropout

Testing:



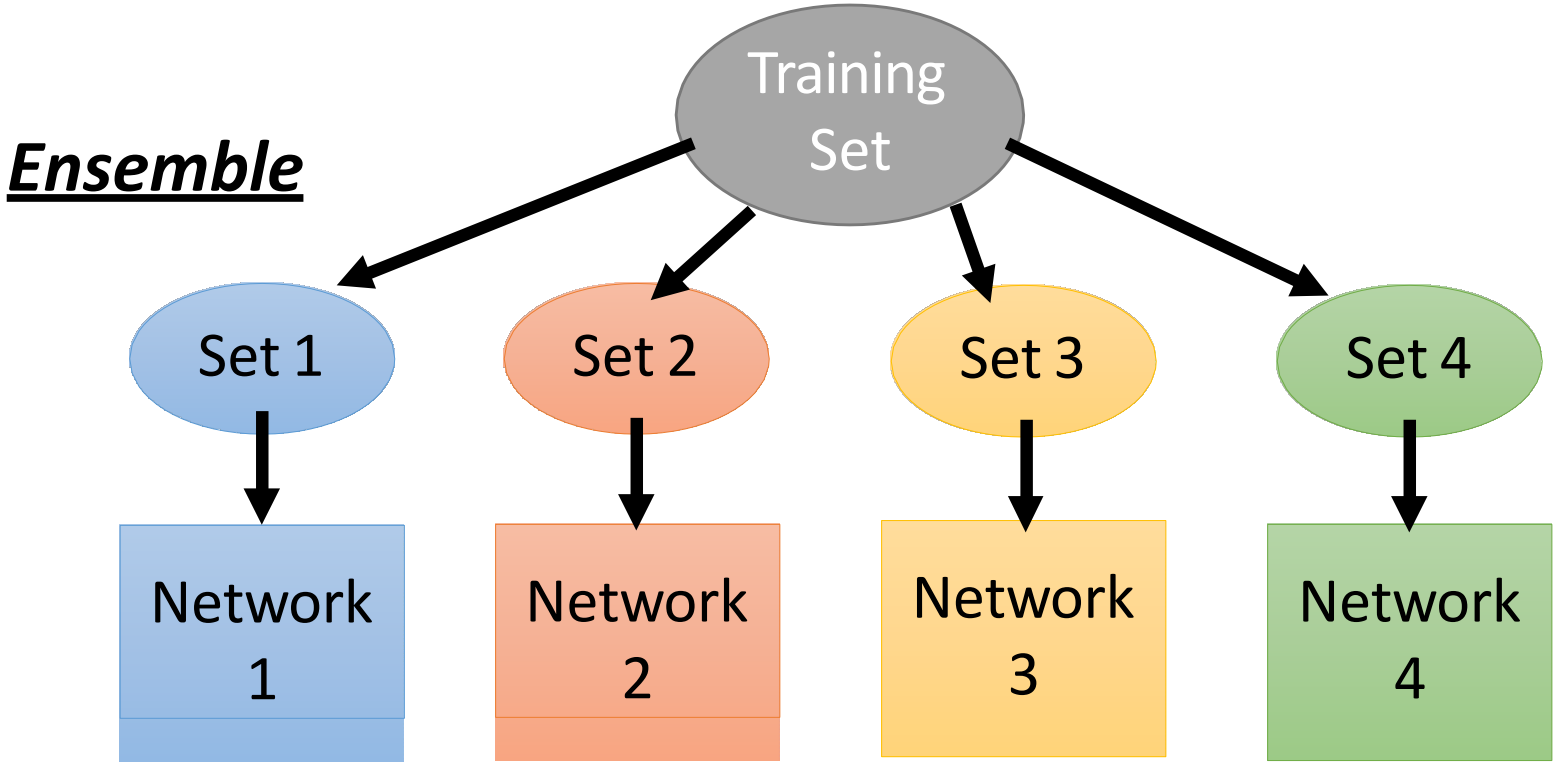
- **No dropout !!!**

Dropout - Intuitive Reason



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

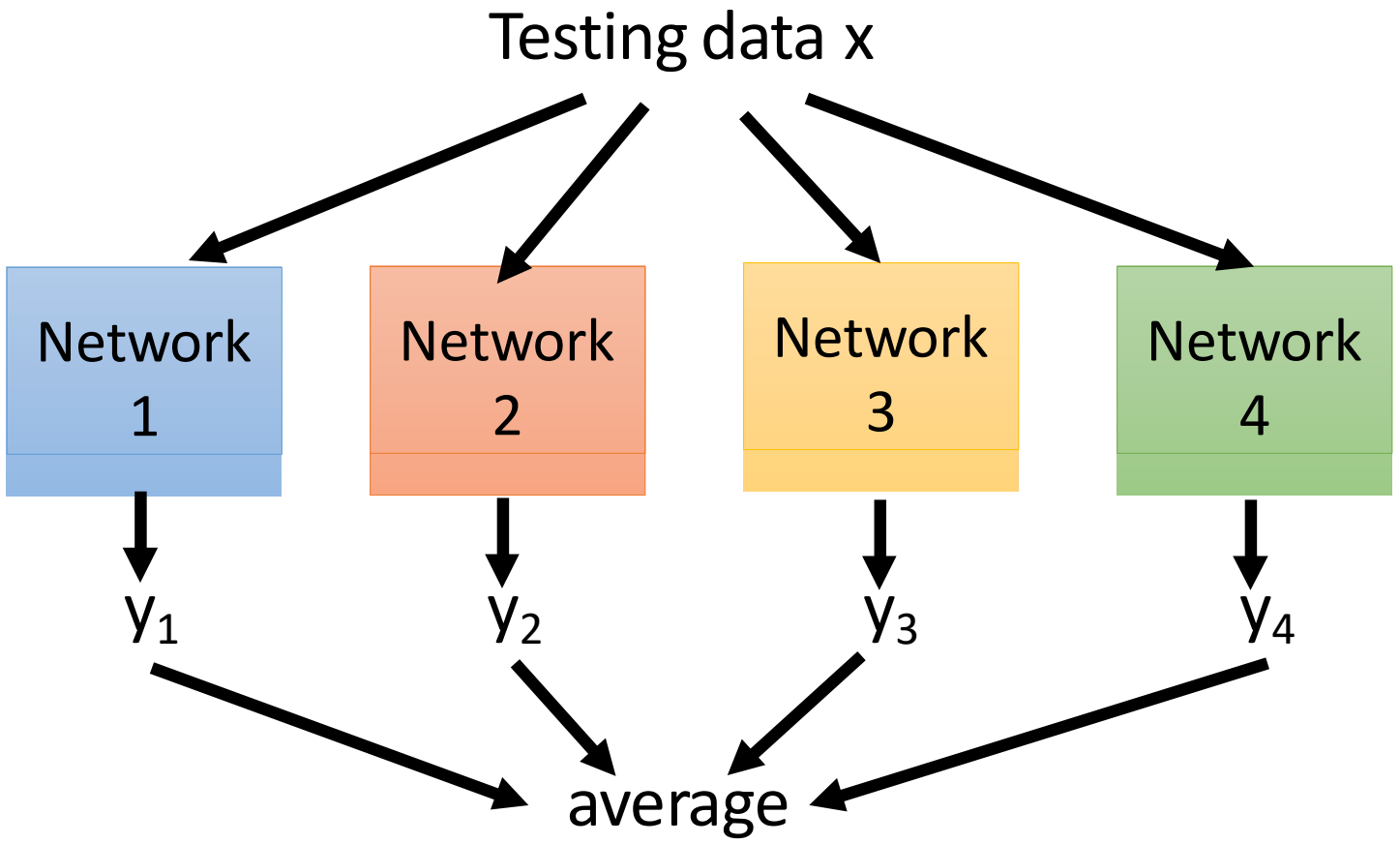
Dropout is a kind of ensemble



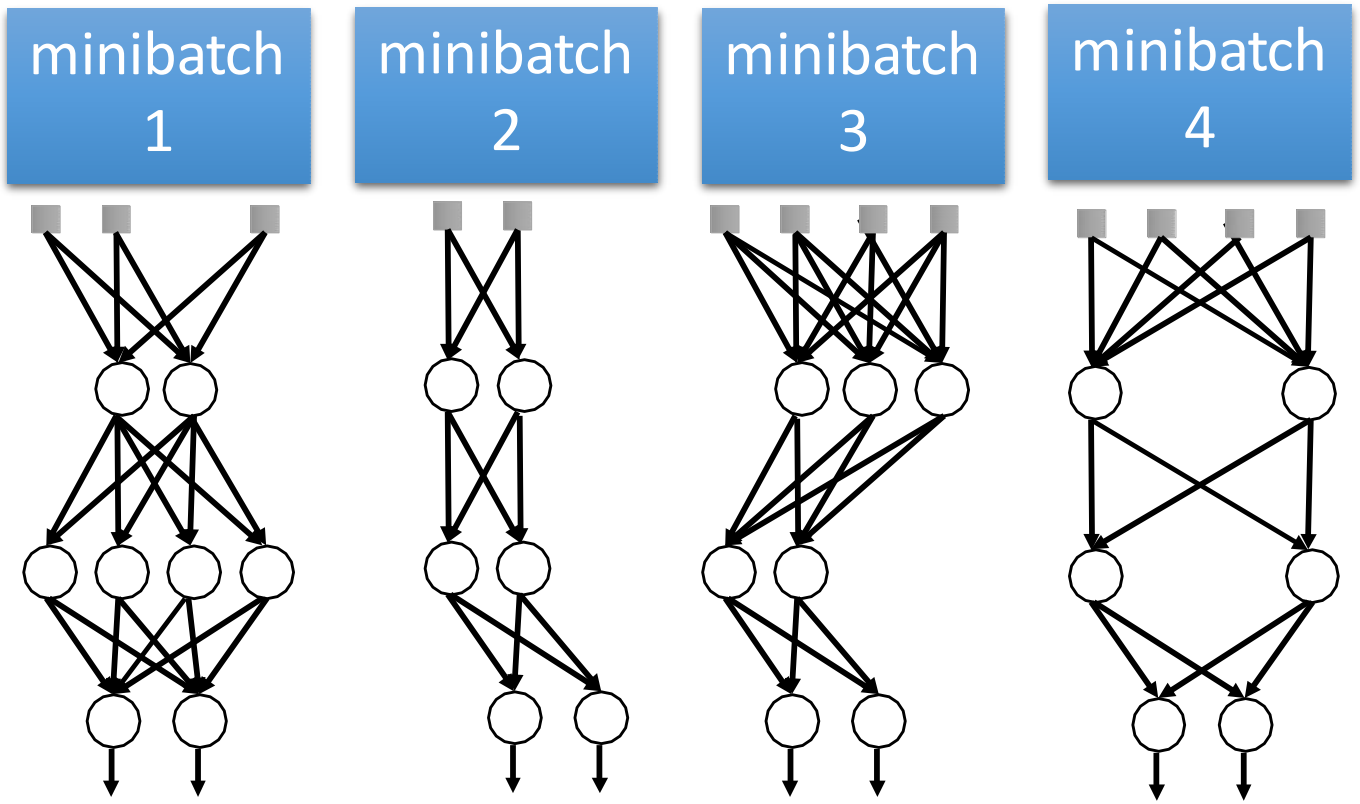
Train a bunch of networks with different structures

Dropout is a kind of ensemble

Ensemble



Dropout is a kind of ensemble



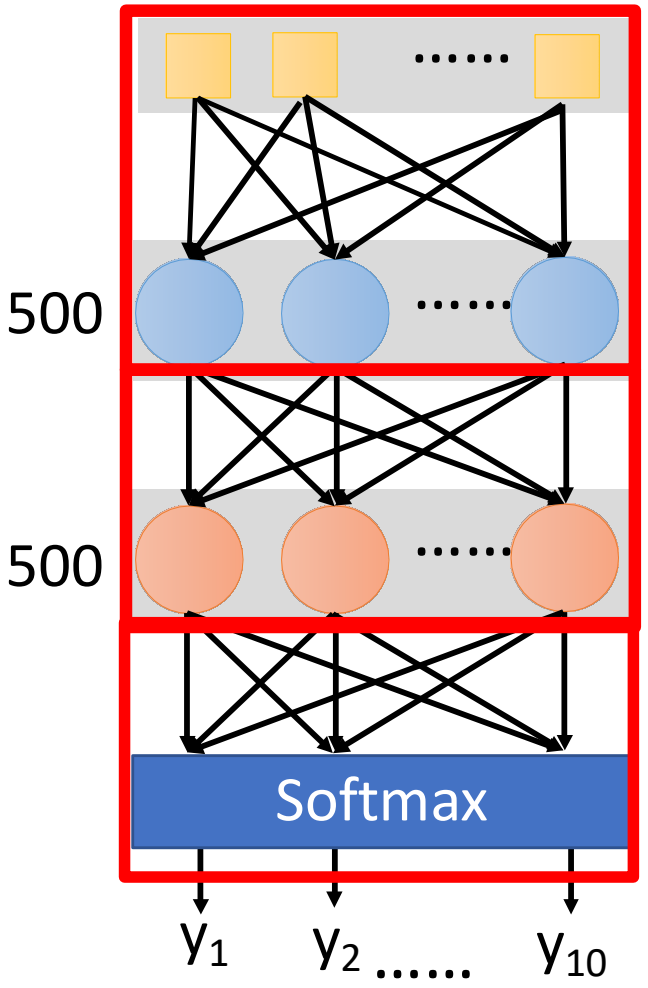
Training of Dropout

M neurons

2^M possible networks

- Using one mini-batch to train one network

Let's try it




```
model = Sequential()
```

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

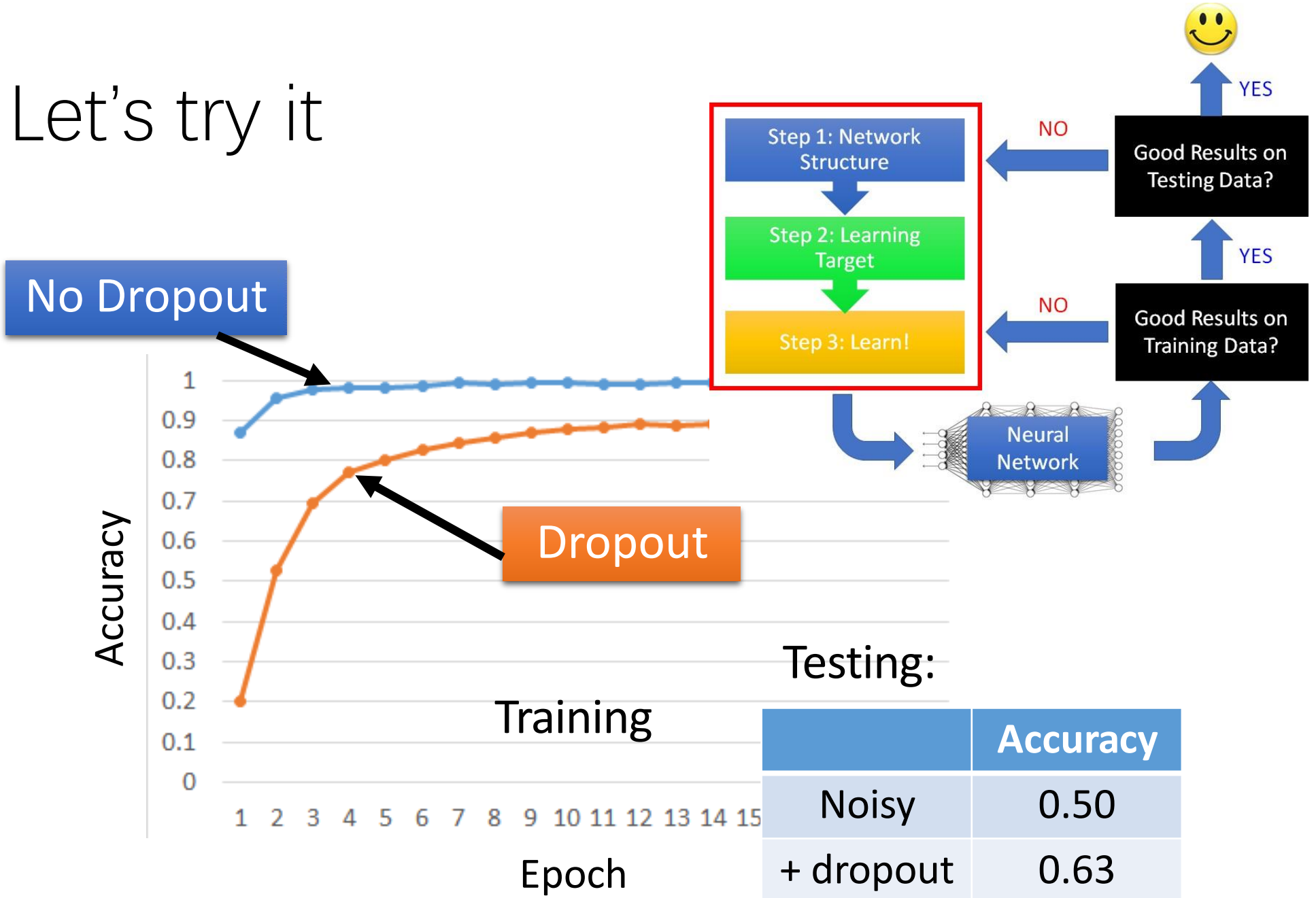
model.add(dropout(0.8))

```
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') )
```

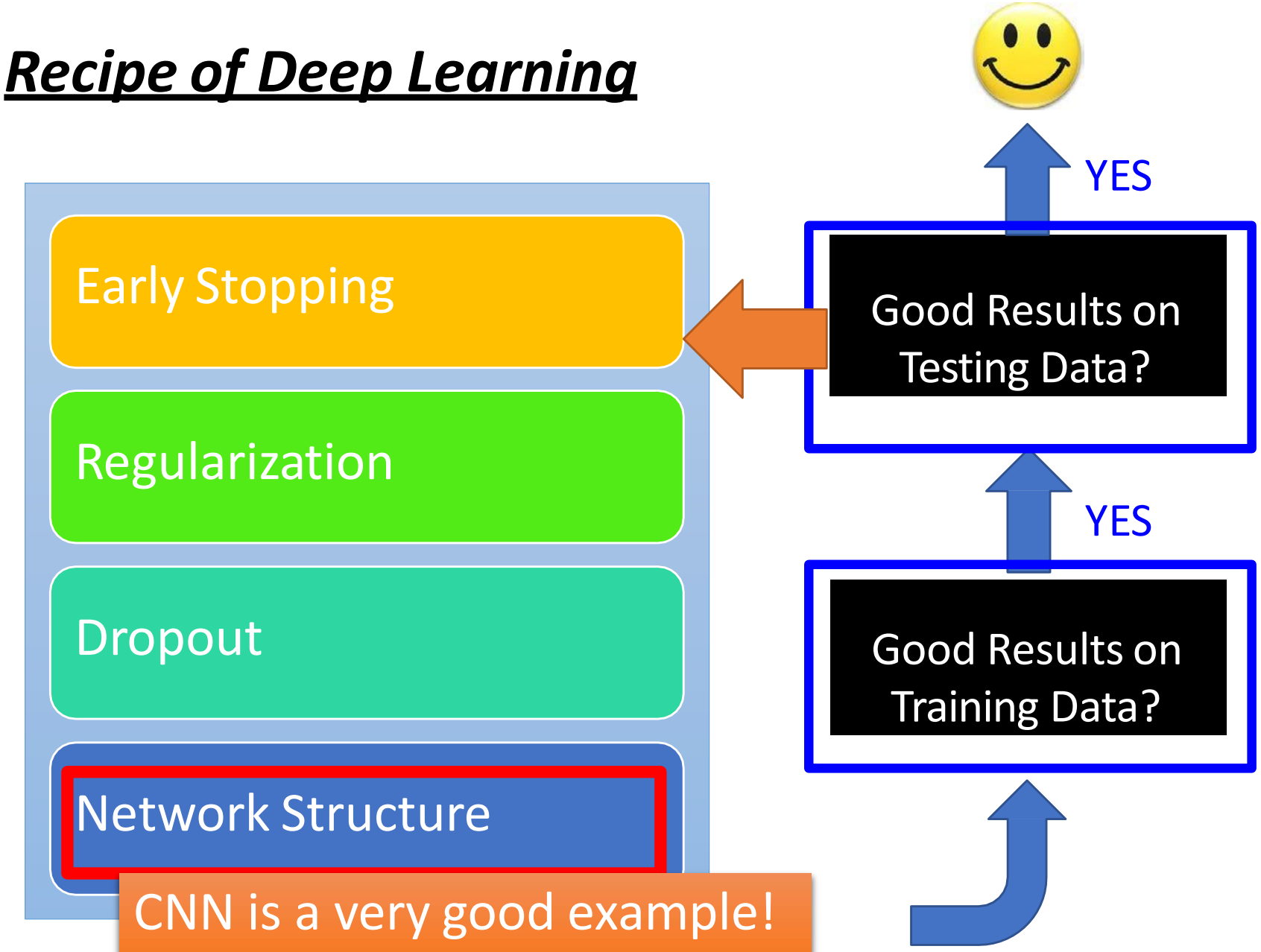
Let's try it



Testing:

	Accuracy
Noisy	0.50
+ dropout	0.63

Recipe of Deep Learning



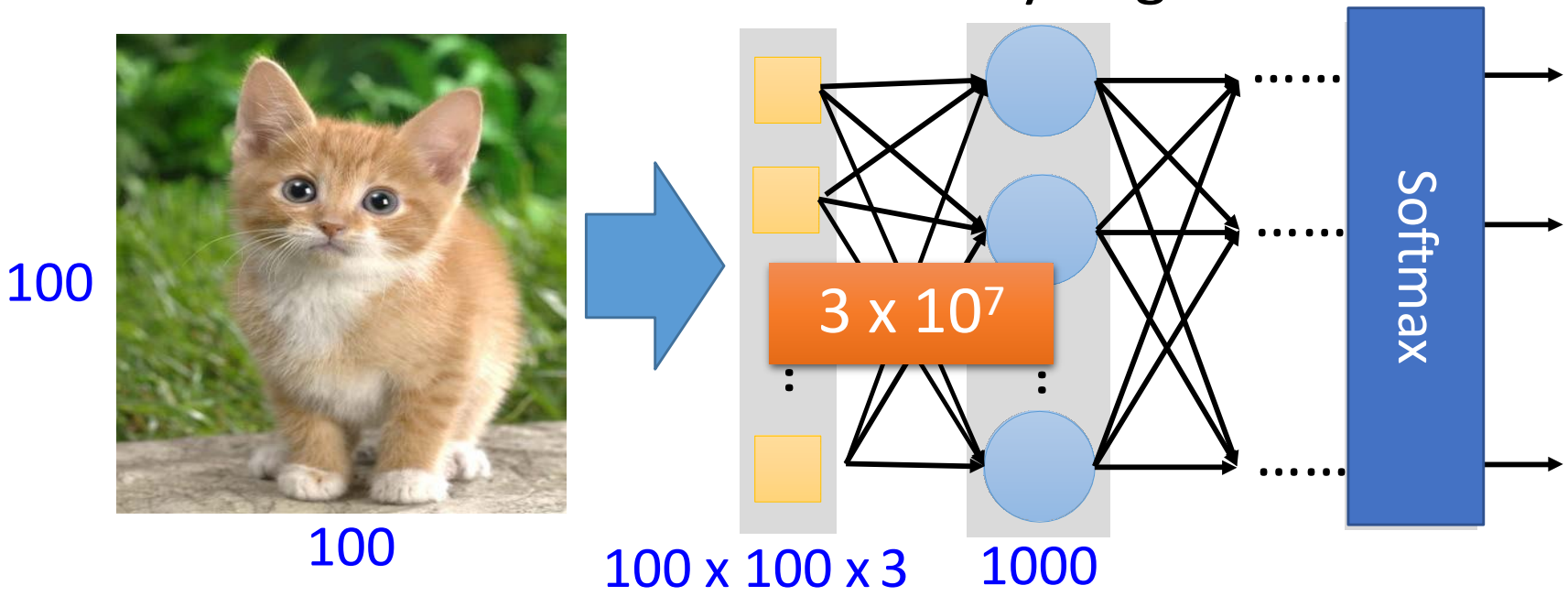
Part5 :Convolution Neural Network

Variants of Neural Networks

Convolutional Neural Network (CNN)	Deep Learning for Computer Vision
Recurrent Neural Network (RNN)	Deep Learning for text and time-series data
Long Short-Term Memory (LSTM)	Deep Learning for text and time-series data
Gated Recurrent Unit (GRU)	Deep Learning for text and time-series data

Why CNN for Image?

- When processing image, the first layer of fully connected network would be very large



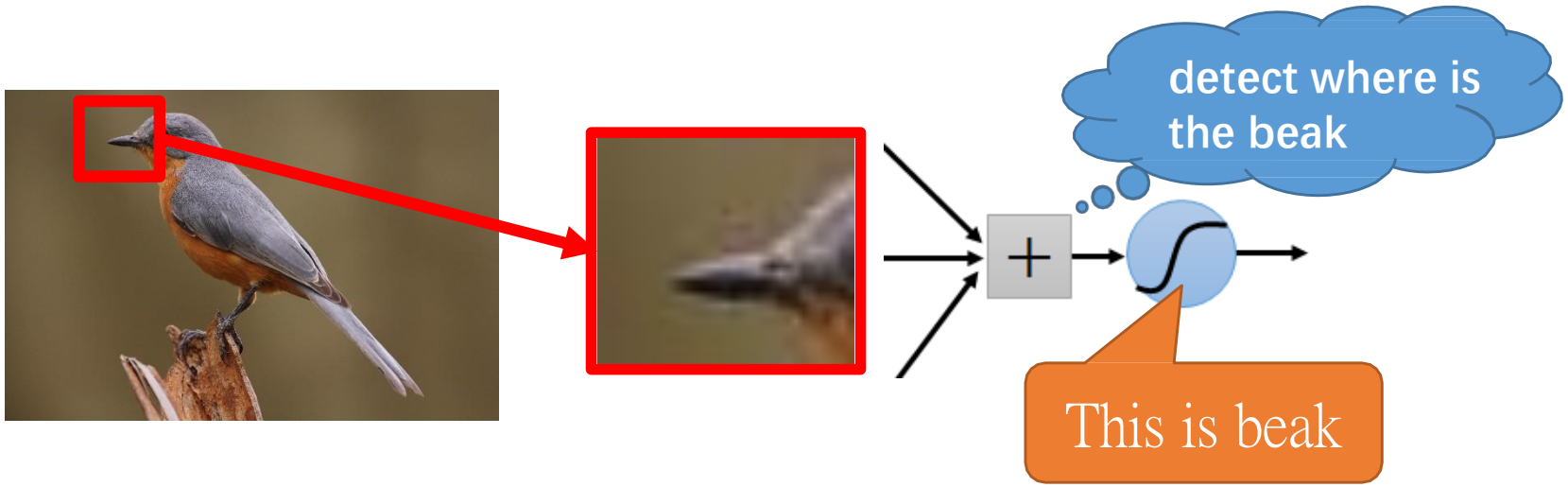
Can the fully connected network be simplified by considering the properties of image recognition?

Why CNN for Image

- Some patterns are much smaller than the whole image

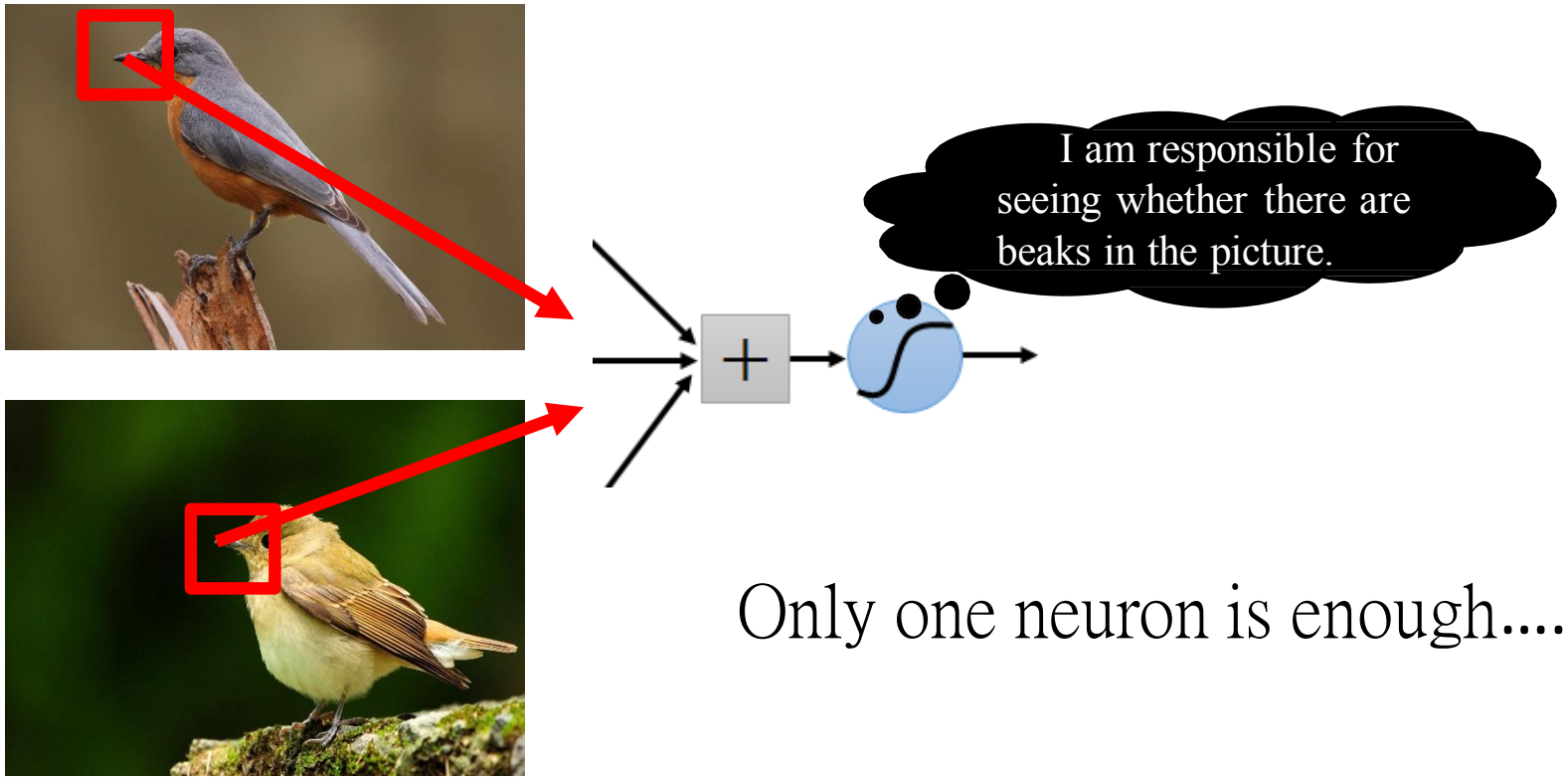
A neuron does not have to see the whole image to discover the pattern.

Connecting to small region with less parameters



Why CNN for Image

- The same patterns appear in different regions.



Only one neuron is enough……

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

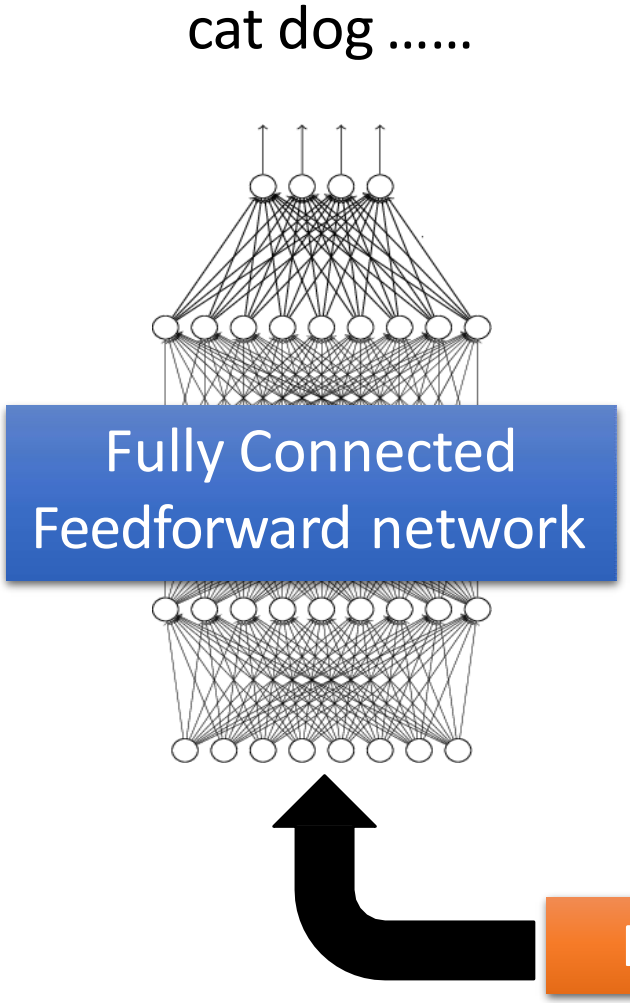
Convolutional Neural Network

Step 1: Neural Network

Step 2: Learning Target

Step 3: Learn!

The whole CNN



The whole CNN

Property 1

- Some patterns are much smaller than the whole image

Property 2

- The same patterns appear in different regions.

Property 3

- Subsampling the pixels will not change the object

Convolution

Max Pooling

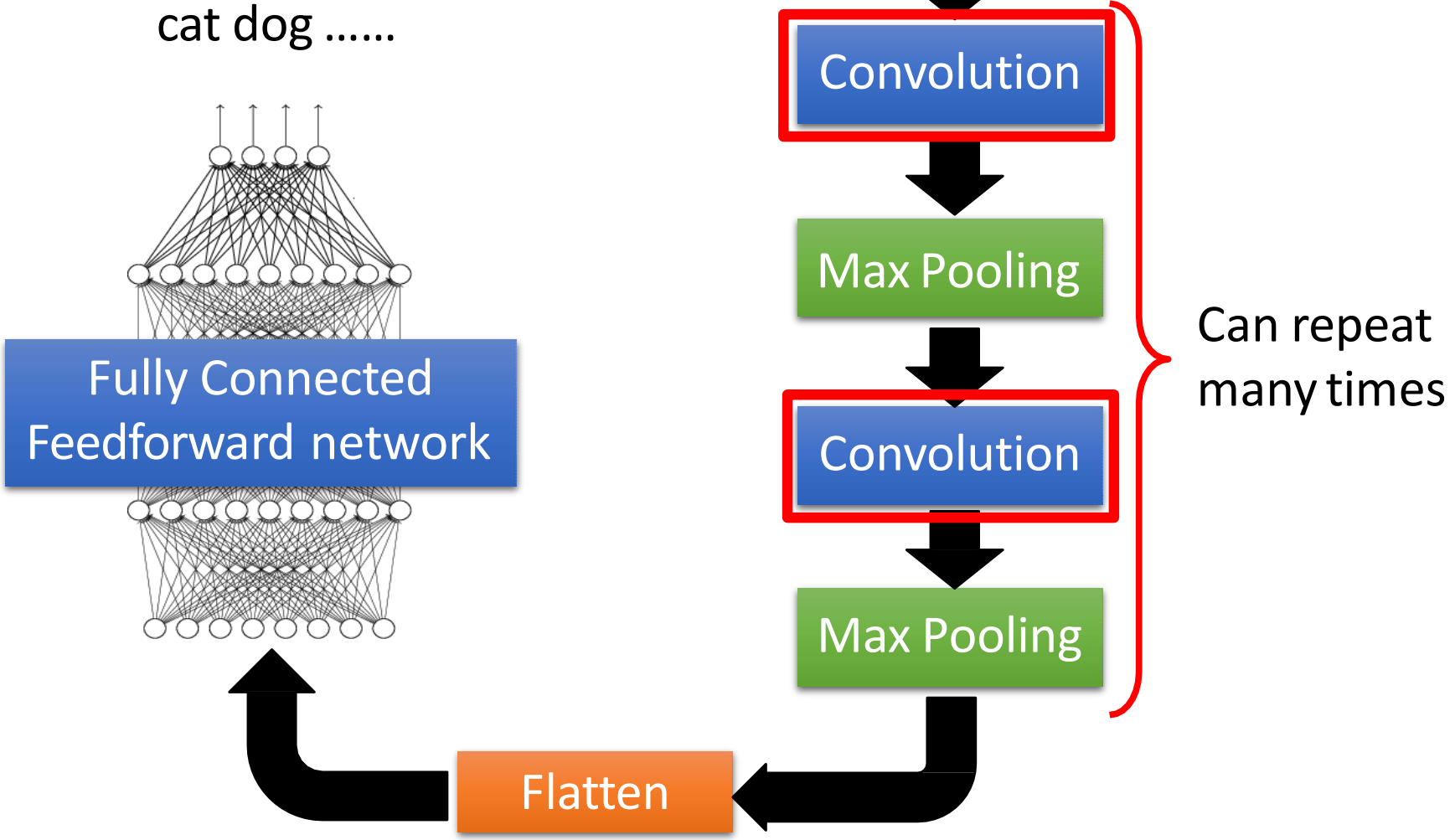
Convolution

Max Pooling

Can repeat many times

Flatten

The whole CNN



CNN – Convolution

Those are the network parameters to be learned.

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

6 x 6 image

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

Filter 1

Matrix

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

Filter 2

Matrix

⋮

- Some patterns are much smaller than the whole image

Property 1

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

CNN – Convolution

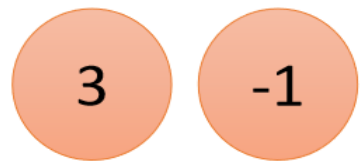
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

stride=1

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

6 x 6 image



CNN – Convolution

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

Filter 1

If stride=2

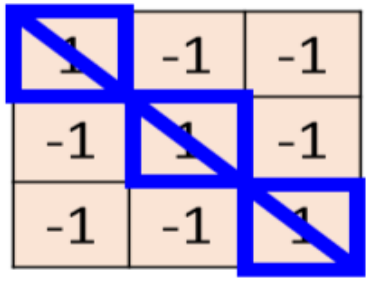
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

3 -3

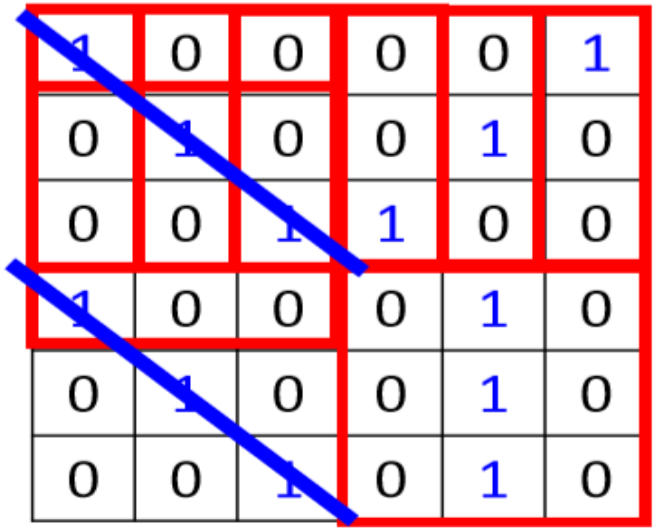
We set stride=1 below

CNN – Convolution

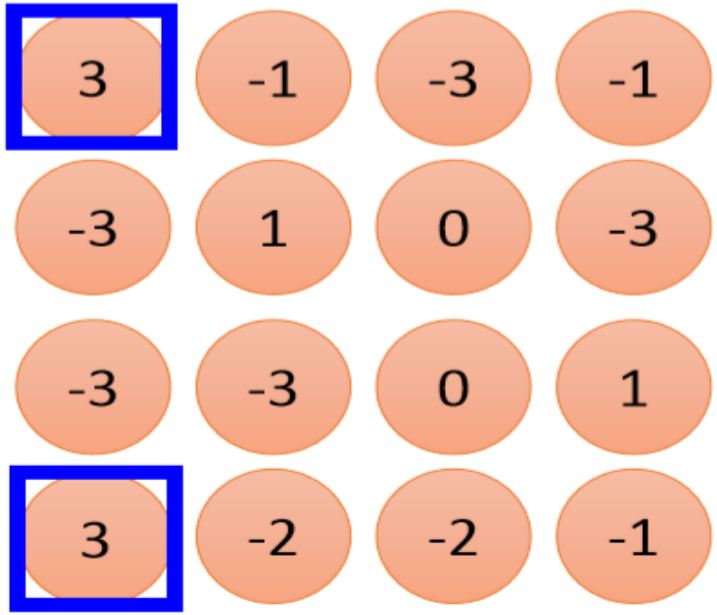


Filter 1

stride=1



6 x 6 image



Property 2

- The same patterns appear in different regions.

CNN – Convolution

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

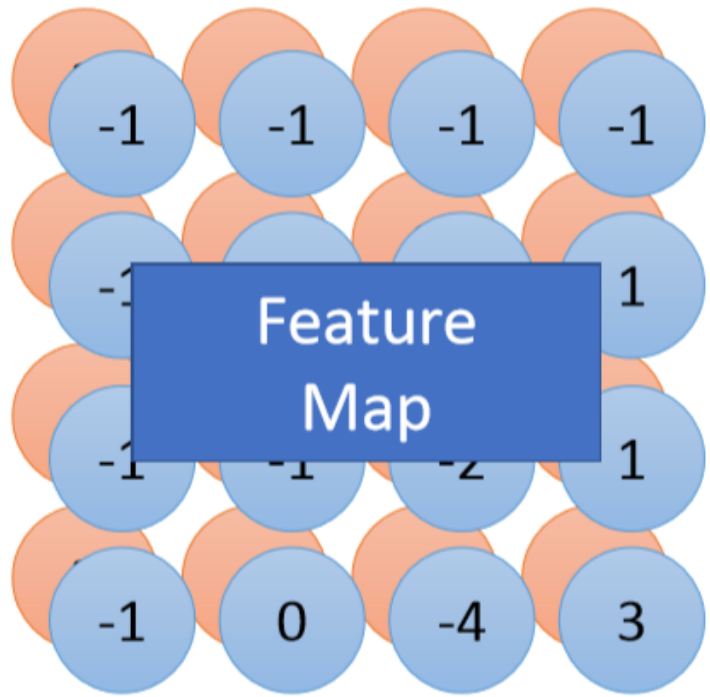
Filter 2

stride=1

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

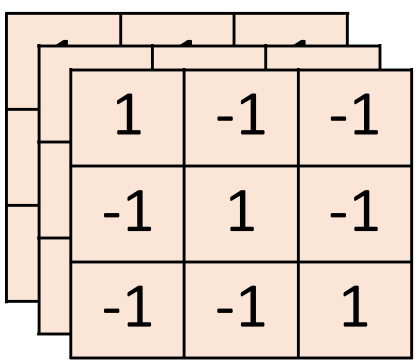
6 x 6 image

Do the same process for every filter

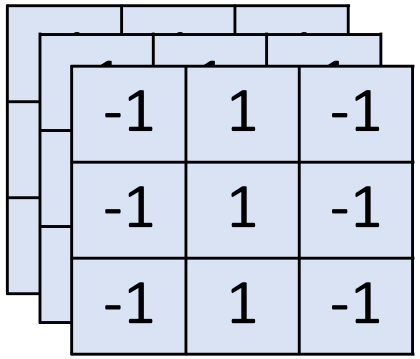


4 x 4 image

CNN – Colorful image

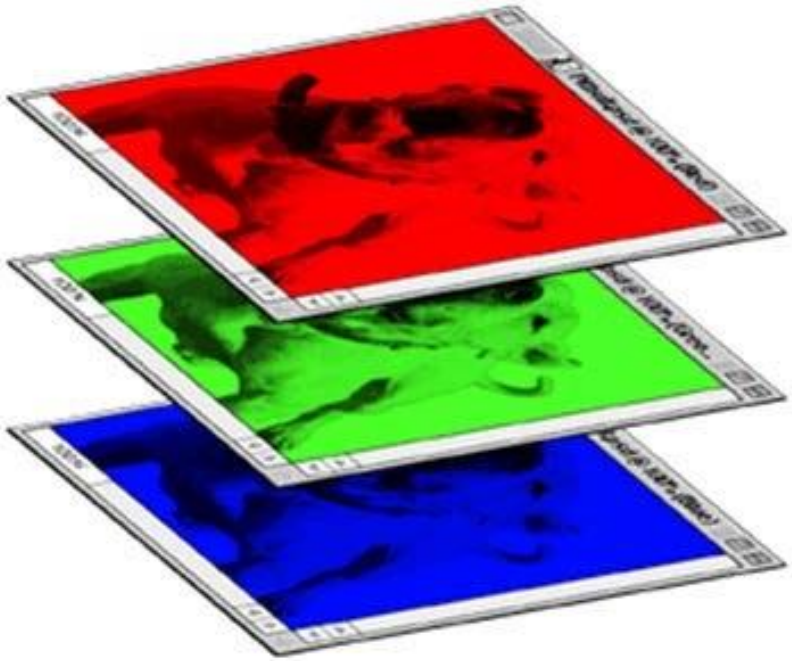


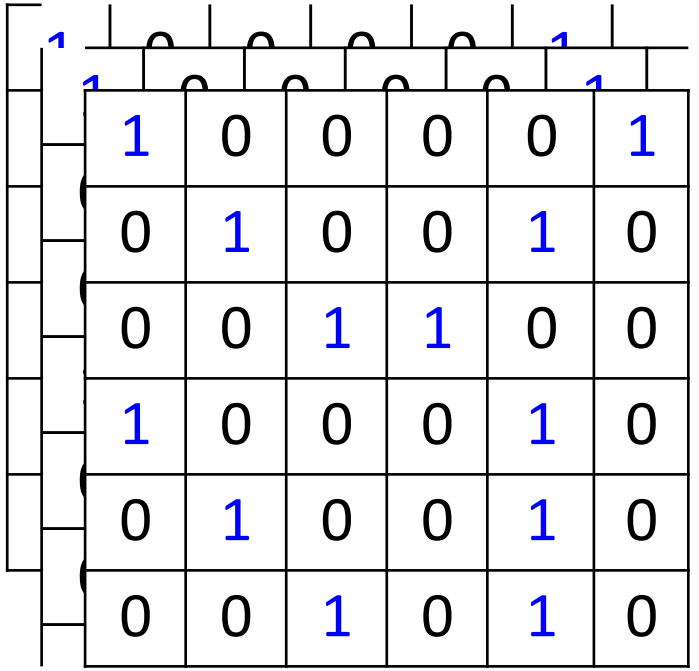
Filter 1



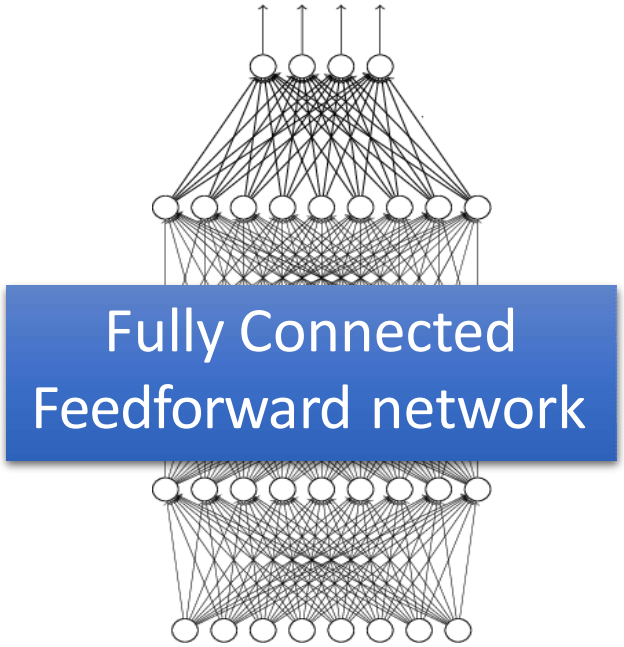
Filter 2

Colorful image





The whole CNN



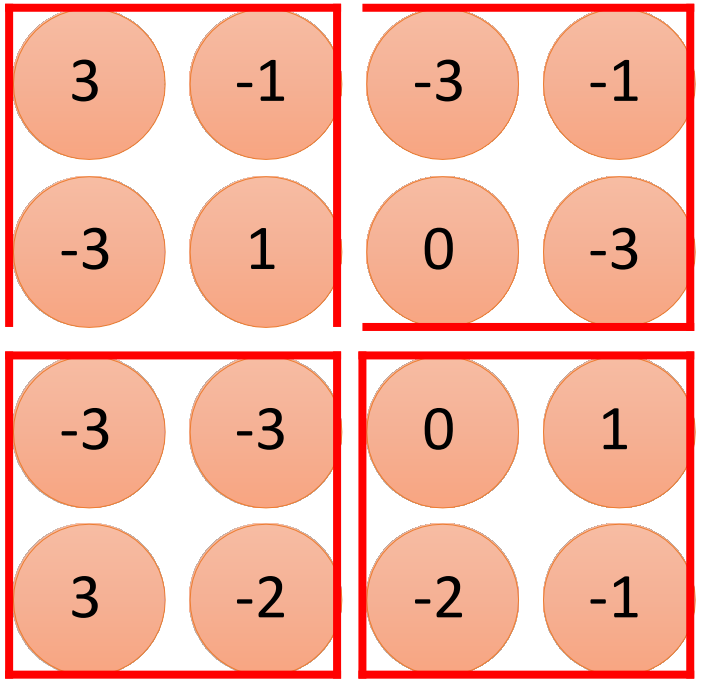
CNN – Max Pooling

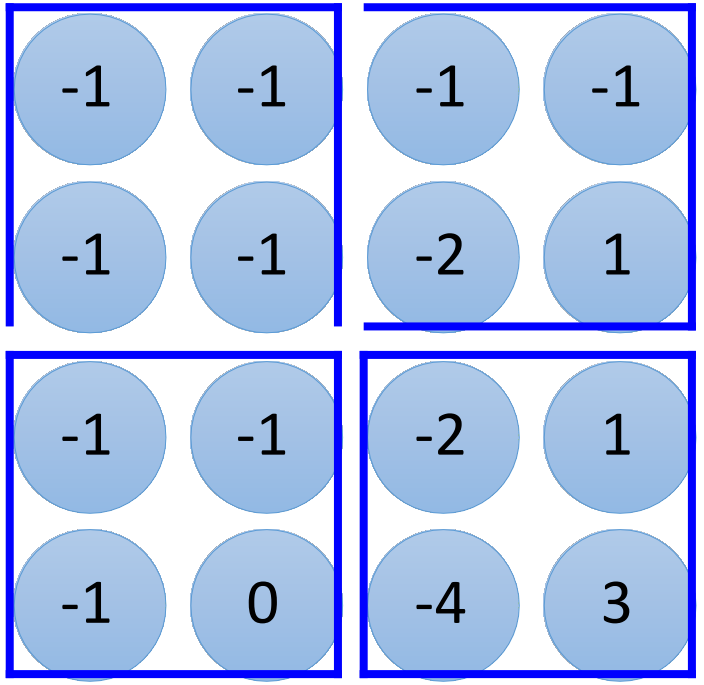
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

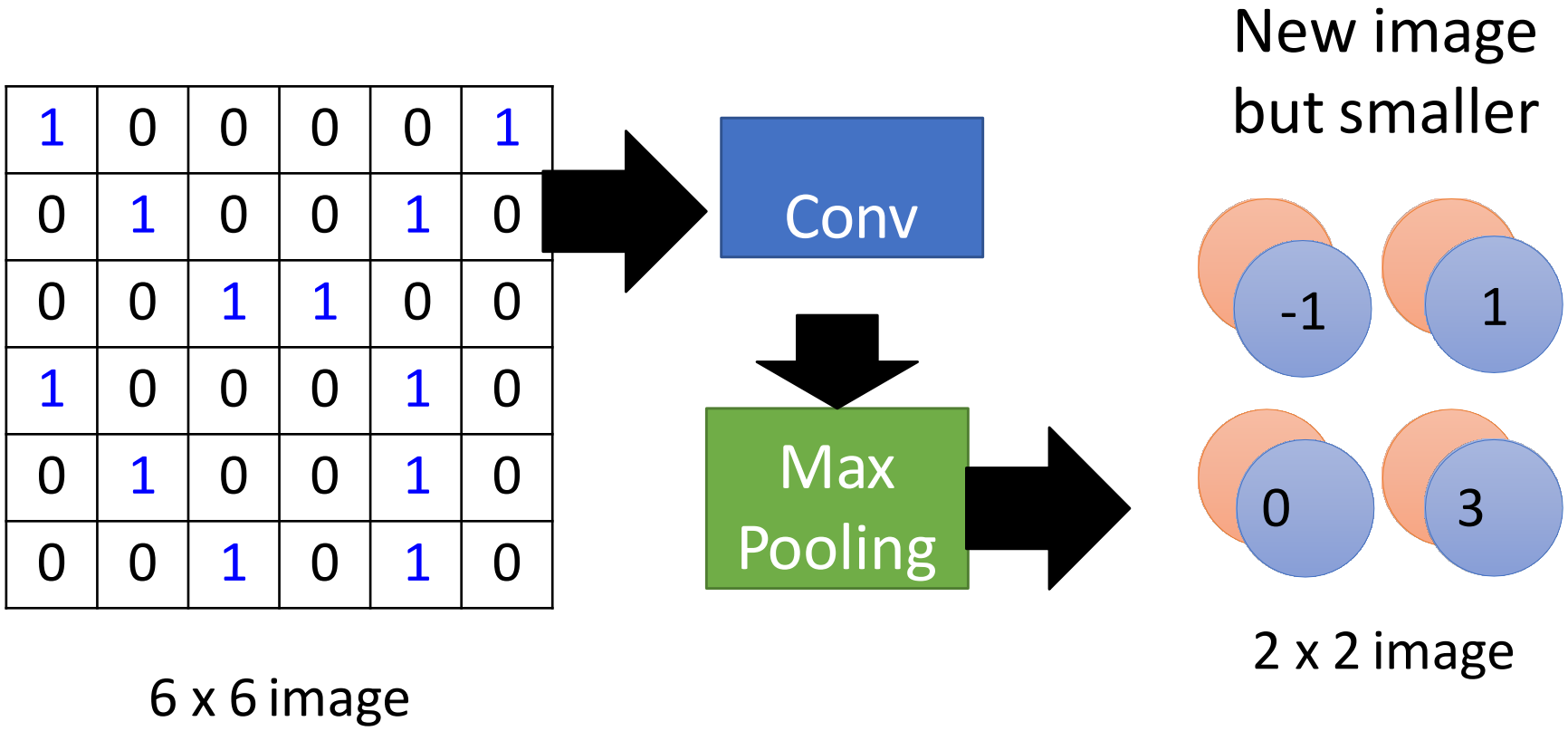
-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

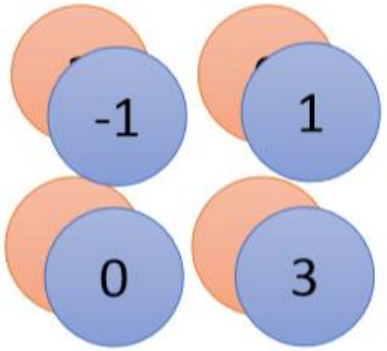




CNN – Max Pooling



The whole CNN



Convolution

Max Pooling

A new image

Convolution

Max Pooling

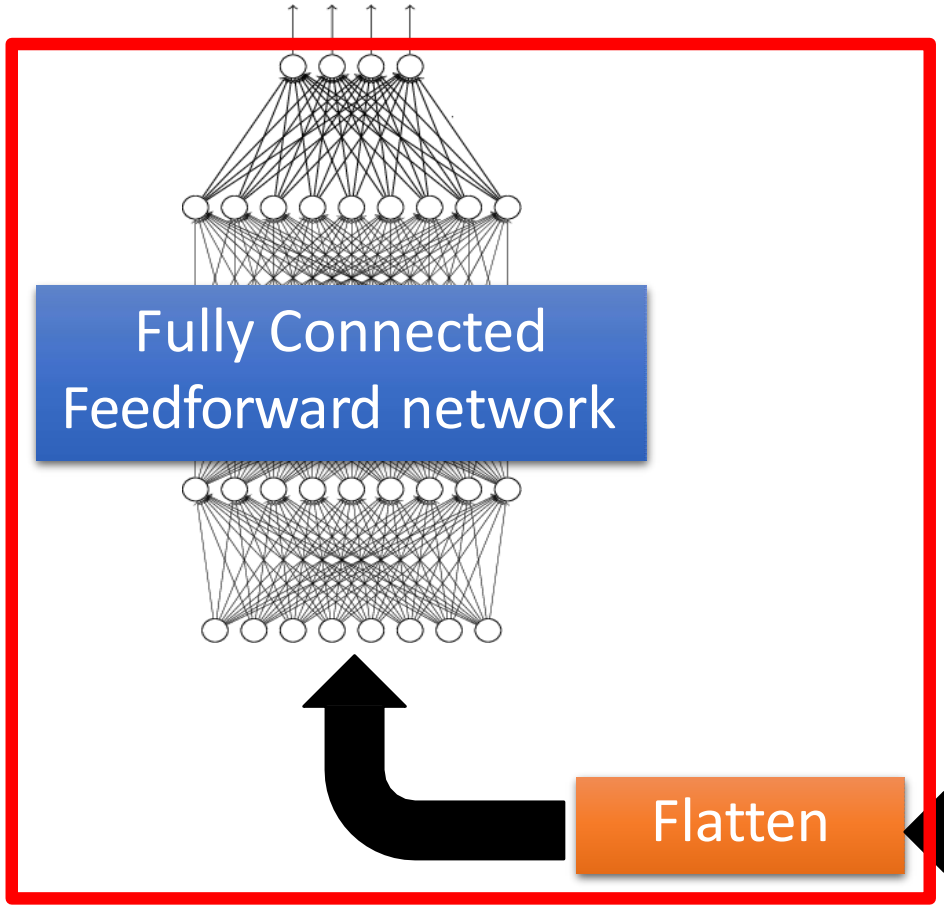
Can repeat many times

Smaller than the original image

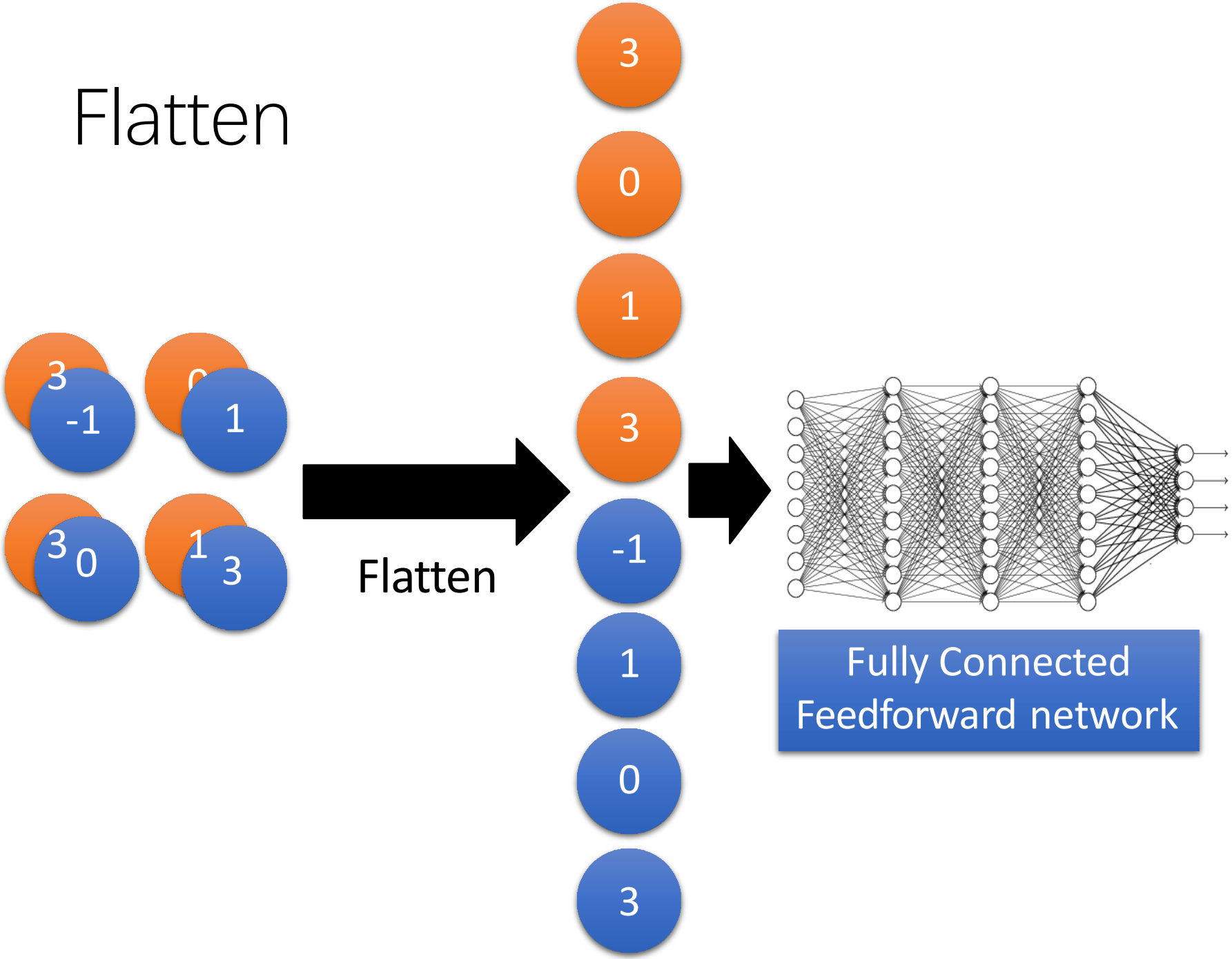
The number of the channel is the number of filters

The whole CNN

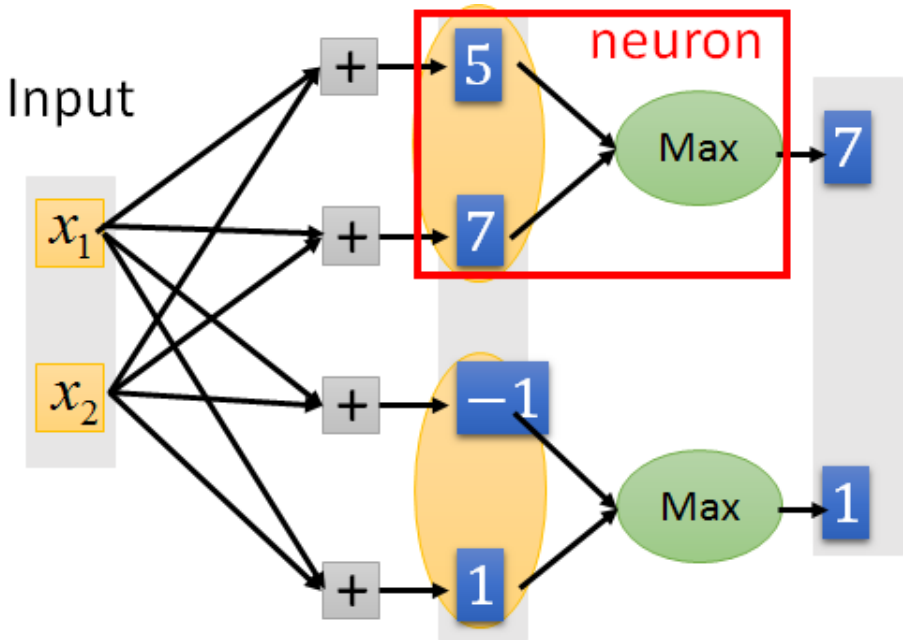
cat dog ……



Flatten



The whole CNN



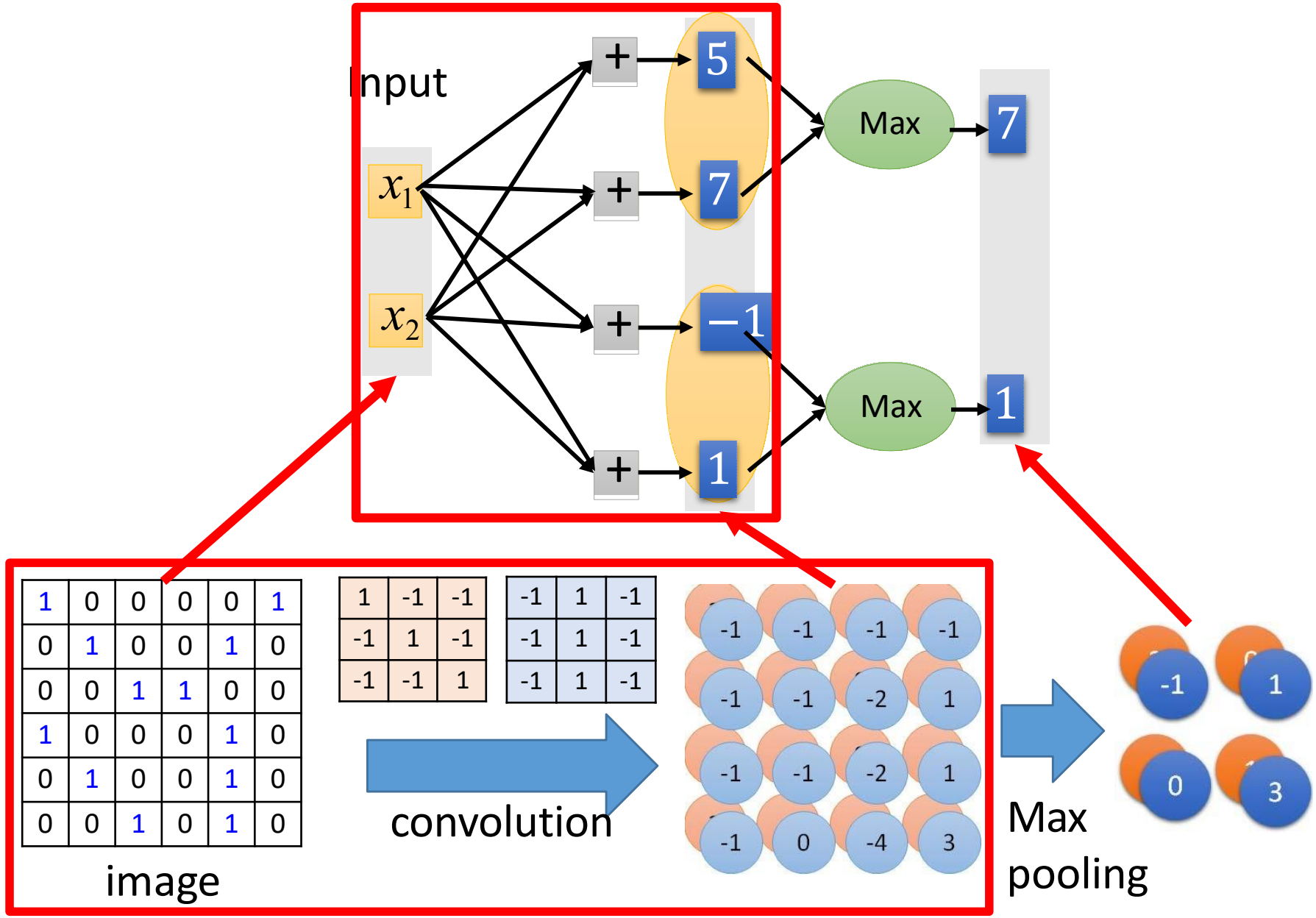
Convolution

Max Pooling

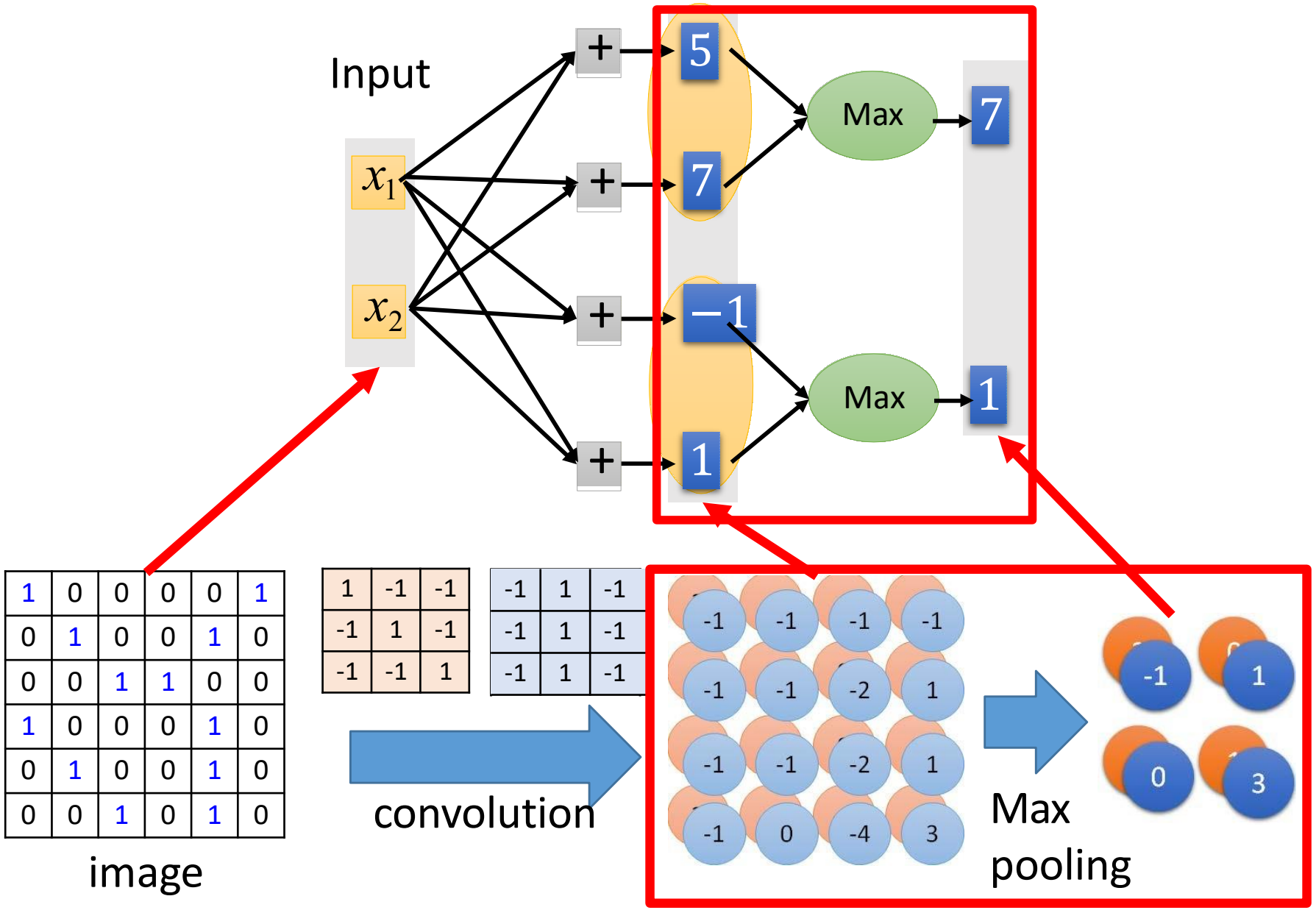
Convolution

Max Pooling

Can repeat many times

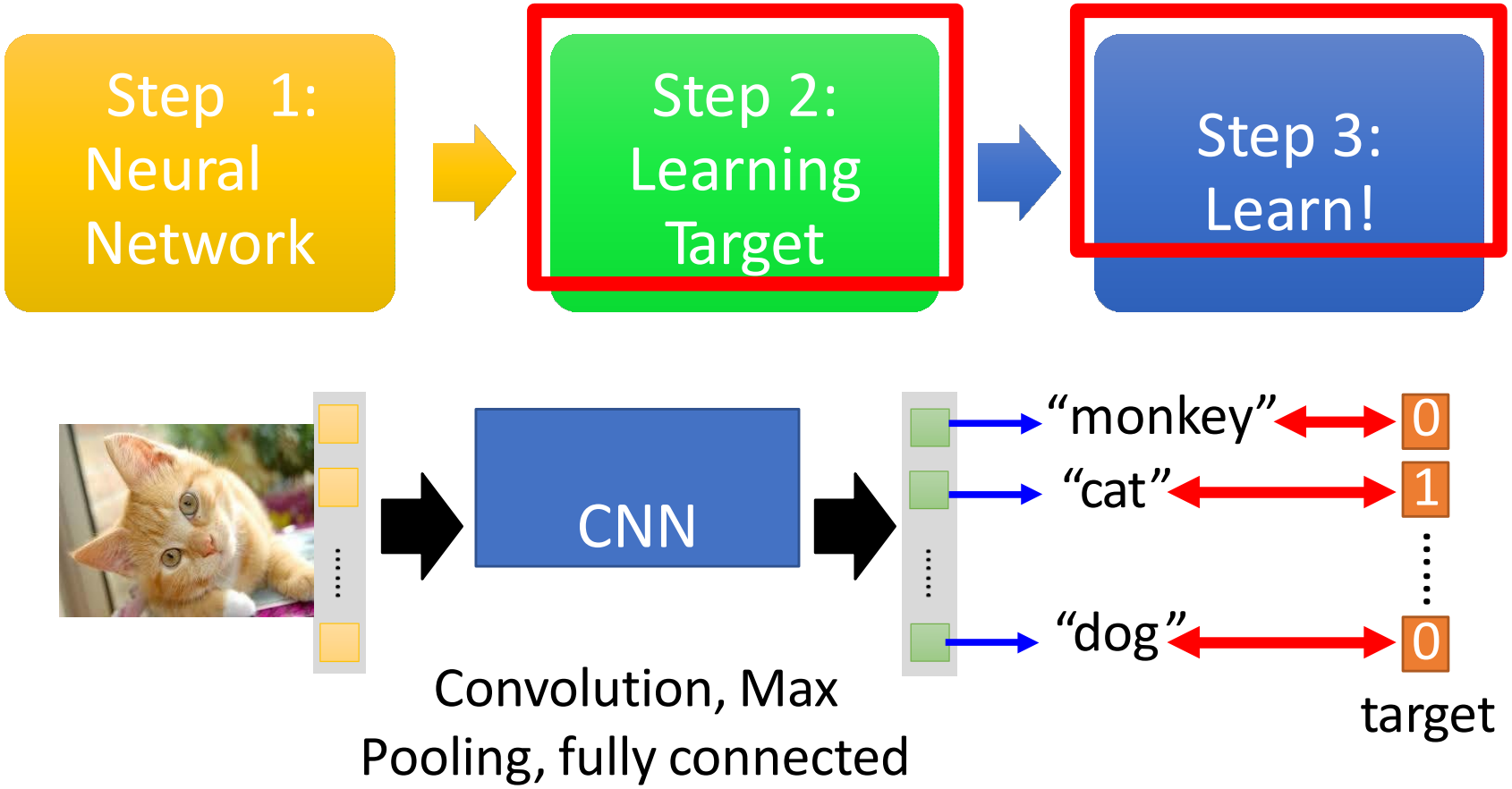


(Ignoring the non-linear activation function after the convolution.)



(Ignoring the non-linear activation function after the convolution.)

Convolutional Neural Network



Learning: Nothing special, just gradient descent ……