Deep Learning Tutorial

李宏毅

Hung-yi Lee

Outline

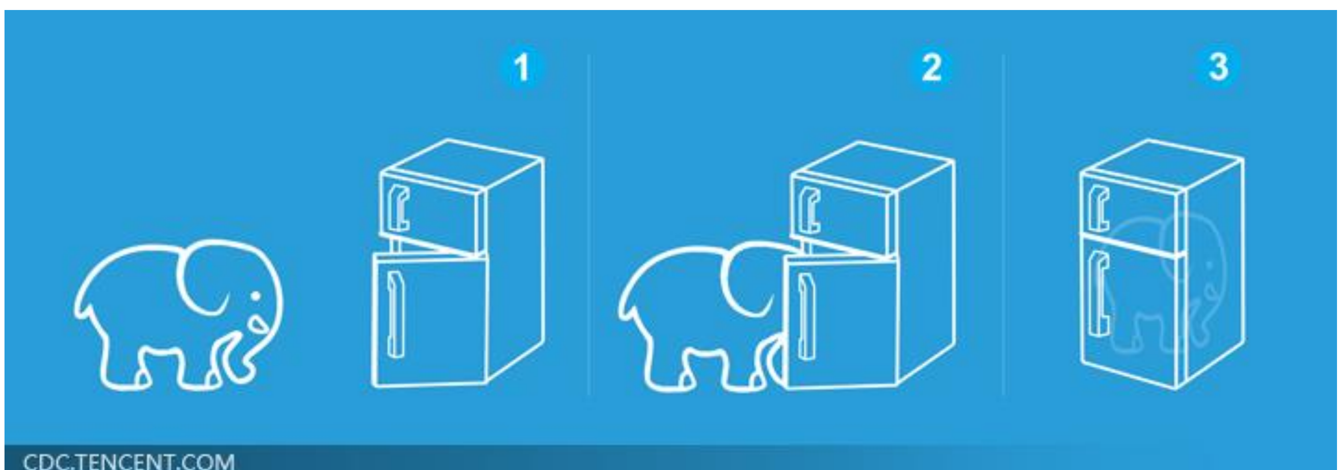
- Part I: Introduction of Deep Learning
- Part II: New Tips for Training Deep Neural Network
- Part III: Why Deep?
- Part IV: Convolutional Neural Network (CNN)
- Part V: Recurrent Neural Network (RNN)
- Part VI: What’s Next?

Part I:
Introduction of Deep Learning

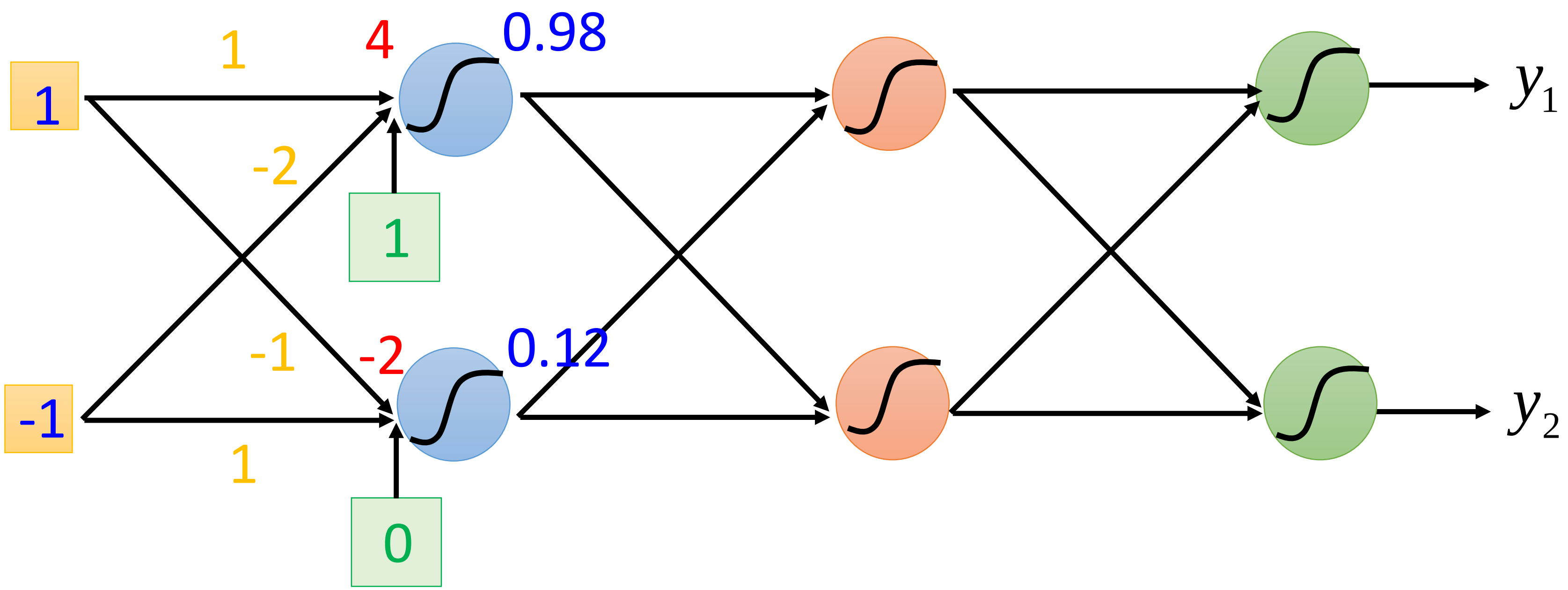
Three Steps for Deep Learning

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

Deep Learning is so simple ……

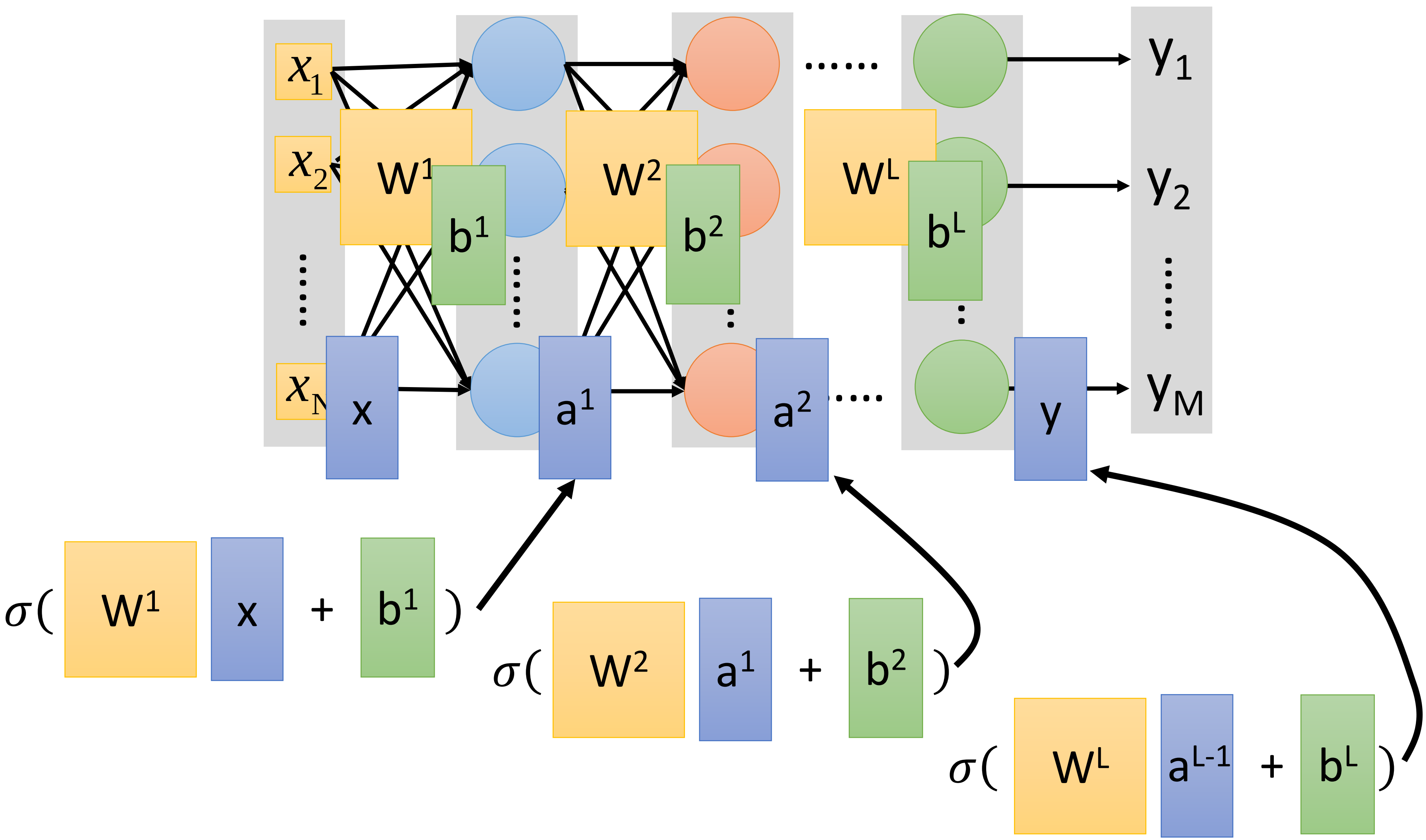


Fully Connected Feedforward Network

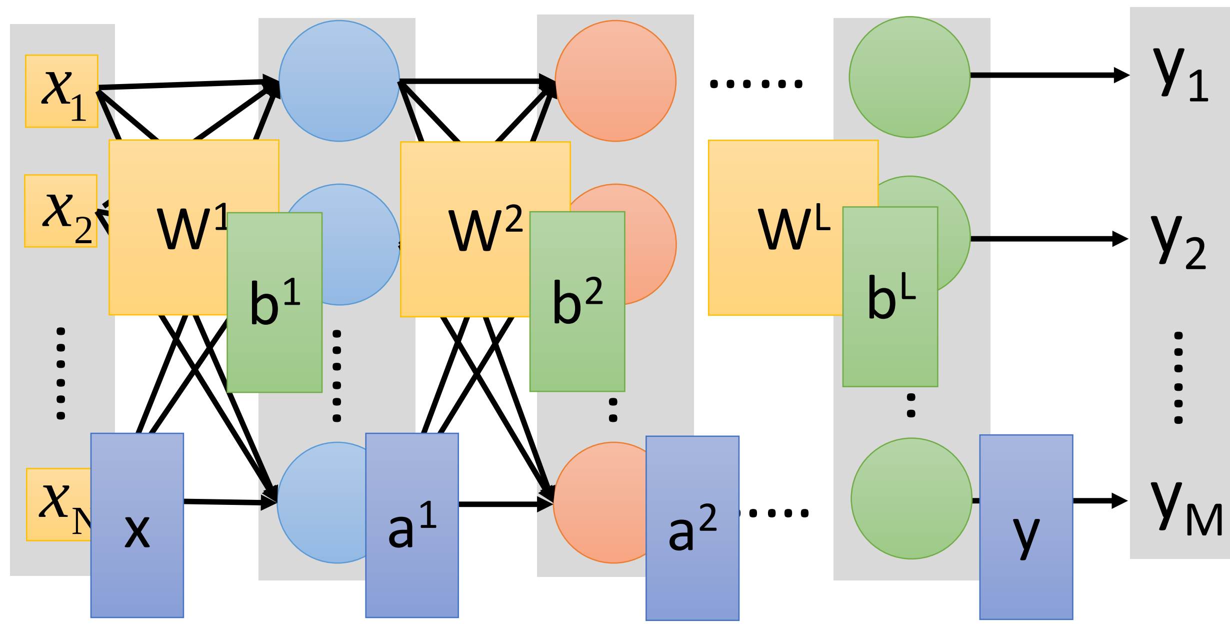


$$\sigma\Bigg(\underbrace{\begin{bmatrix} 1 & -2 \\ -1 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ -1 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix}}_{\begin{bmatrix} 4 \\ -2 \end{bmatrix}}\Bigg) = \begin{bmatrix} 0.98 \\ 0.12 \end{bmatrix}$$

Neural Network



Neural Network



$$y = f(x)$$

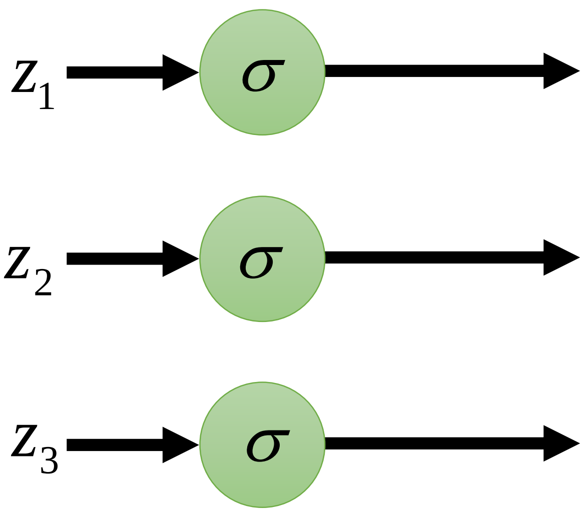
Using parallel computing techniques to speed up matrix operation

$$= \sigma(W^L \cdots \sigma(W^2 \, \sigma(W^1 x + b^1) + b^2) \cdots + b^L)$$

Output Layer

- Softmax layer as the output layer

<u>Ordinary Layer</u>



$$y_1 = \sigma(z_1)$$

$$y_2 = \sigma(z_2)$$

$$y_3 = \sigma(z_3)$$

In this case, the output of network can have any value.

May not be easy to interpret

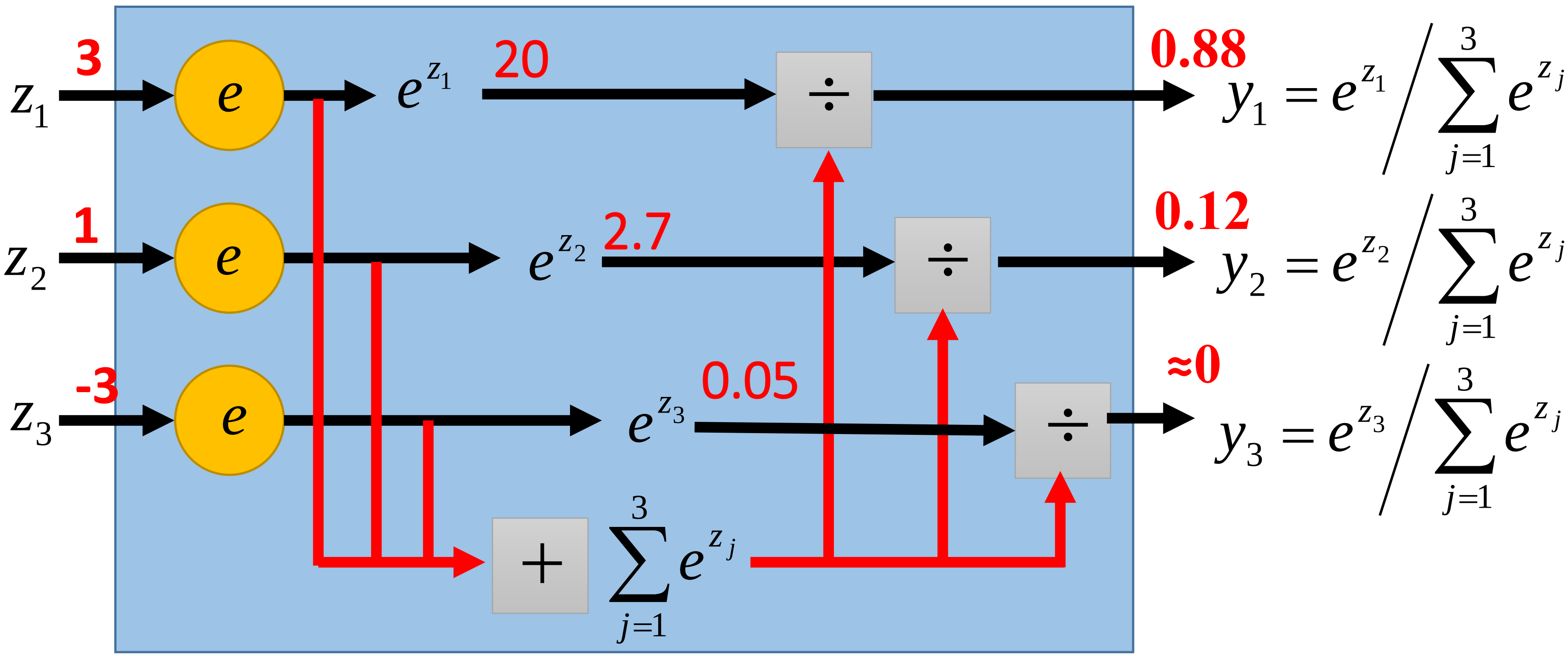
Output Layer

- Softmax layer as the output layer

Probability:

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

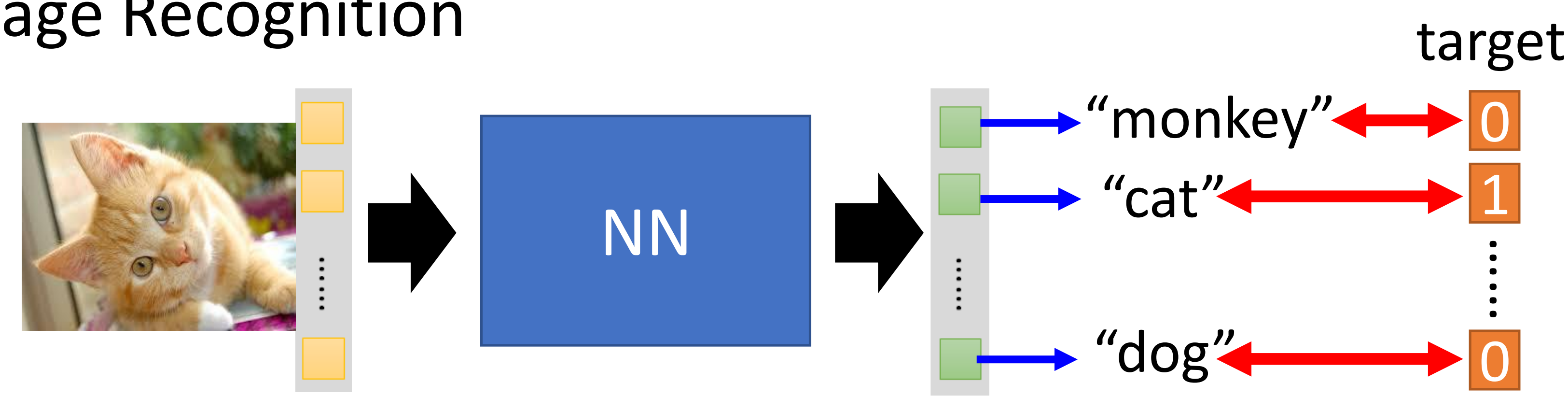
Softmax Layer



Three Steps for Deep Learning

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

Image Recognition

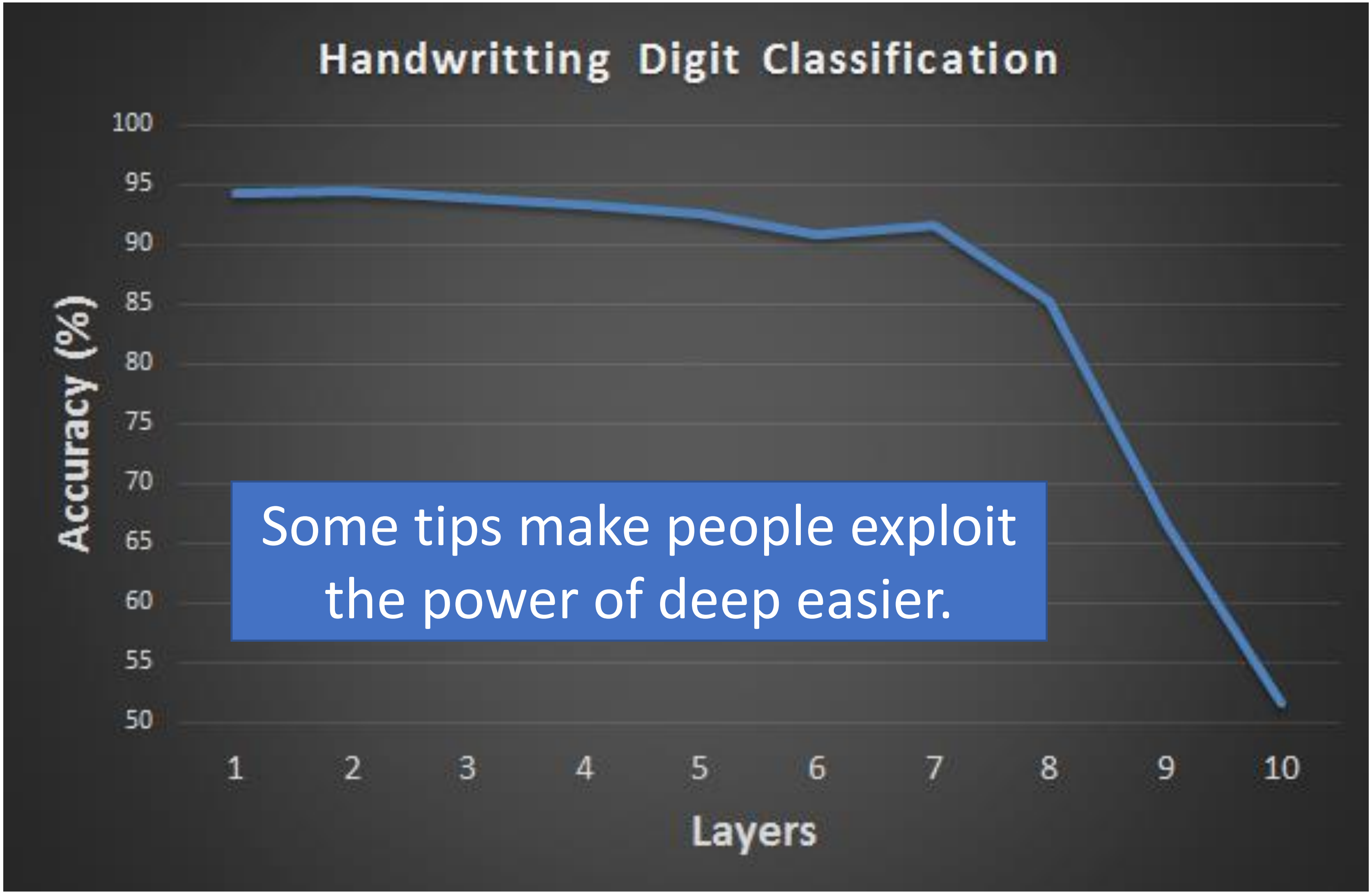


Learning: Nothing special, just gradient descent ……

What is the difference between multi-layer perceptron (MLP) and deep learning?

Basically, old wine in new bottles.

In the past, Deeper ≠ Better



Part II:
Tips for Training Deep Neural Network

What people did not know in 1980s

New Techniques

New Activation Function

- ReLU and Maxout network

New Structure

- Residue network and Highway network

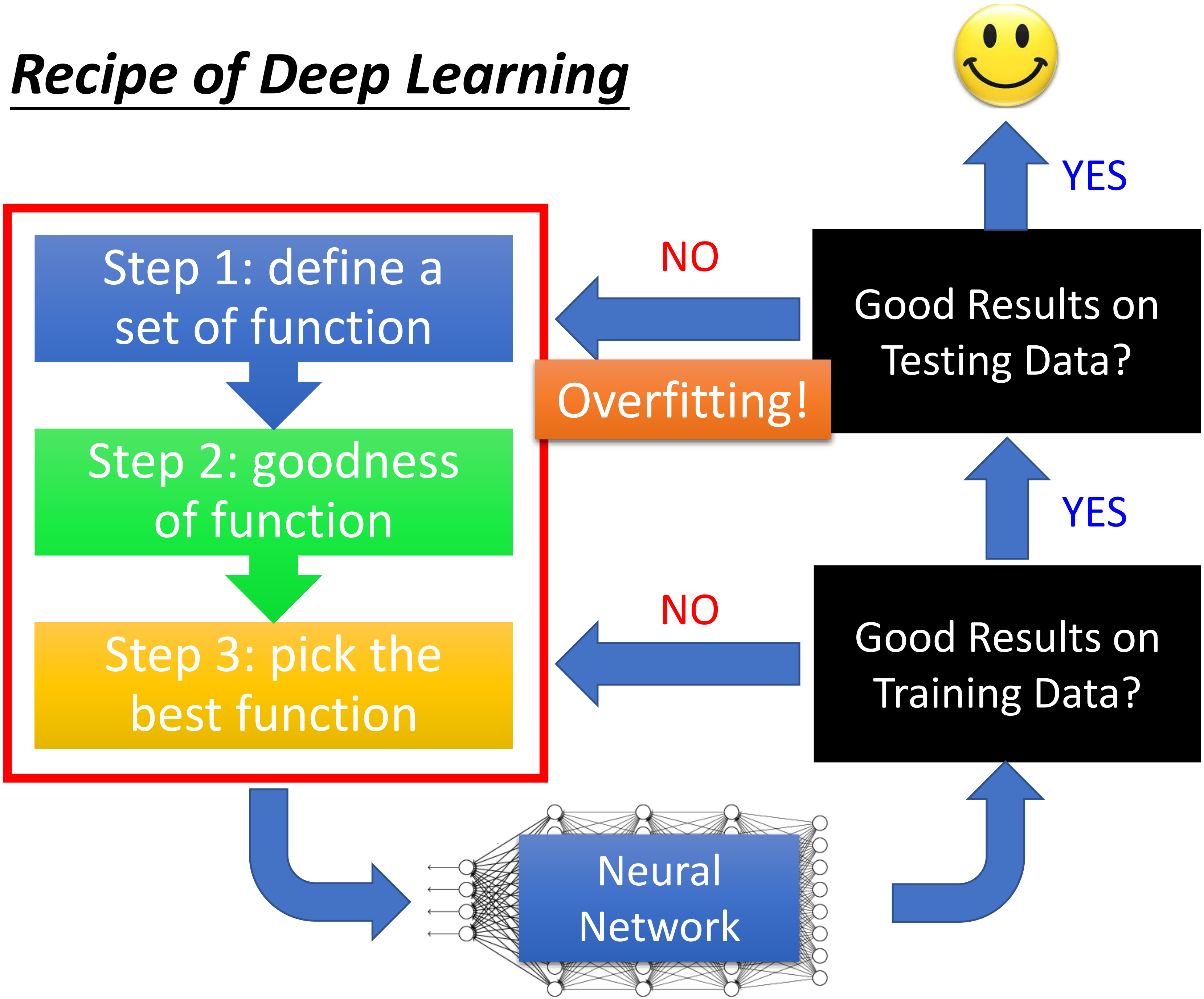
Better optimization Strategy

- E.g. Adam

Dropout

- Prevent Overfitting

Recipe of Deep Learning



New Techniques

New Activation Function

- ReLU and Maxout network

New Structure

- Residue network and Highway network

Better optimization Strategy

- E.g. Adam

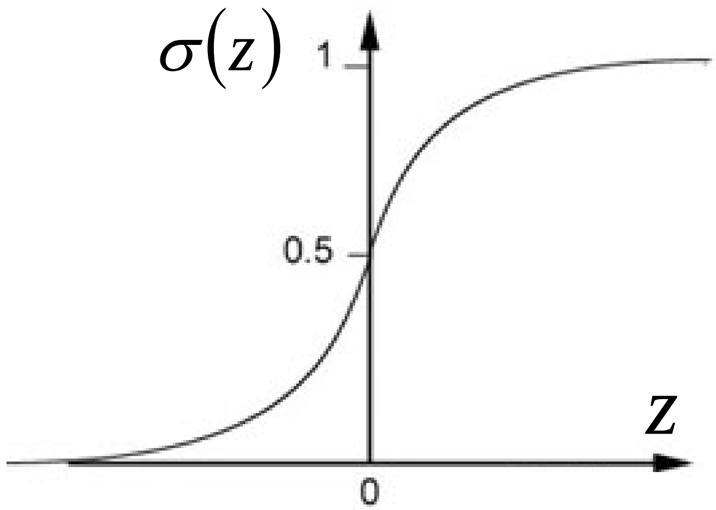
Dropout

- Prevent Overfitting

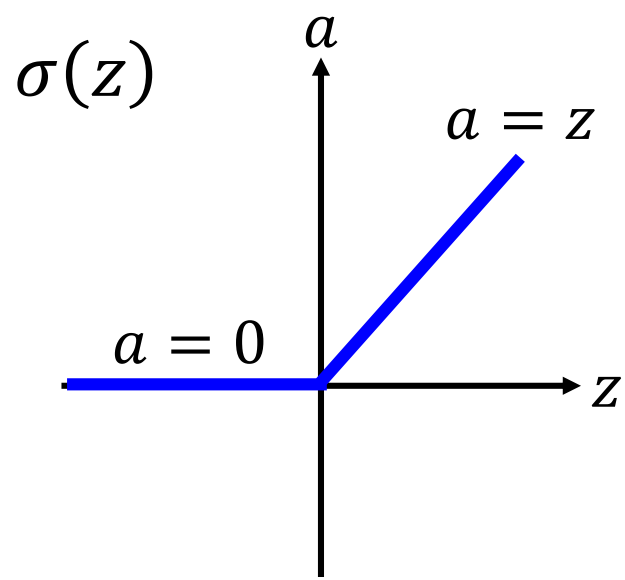
Using this approach when you obtained good results on the training data.

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

New Activation Function



Sigmoid Function



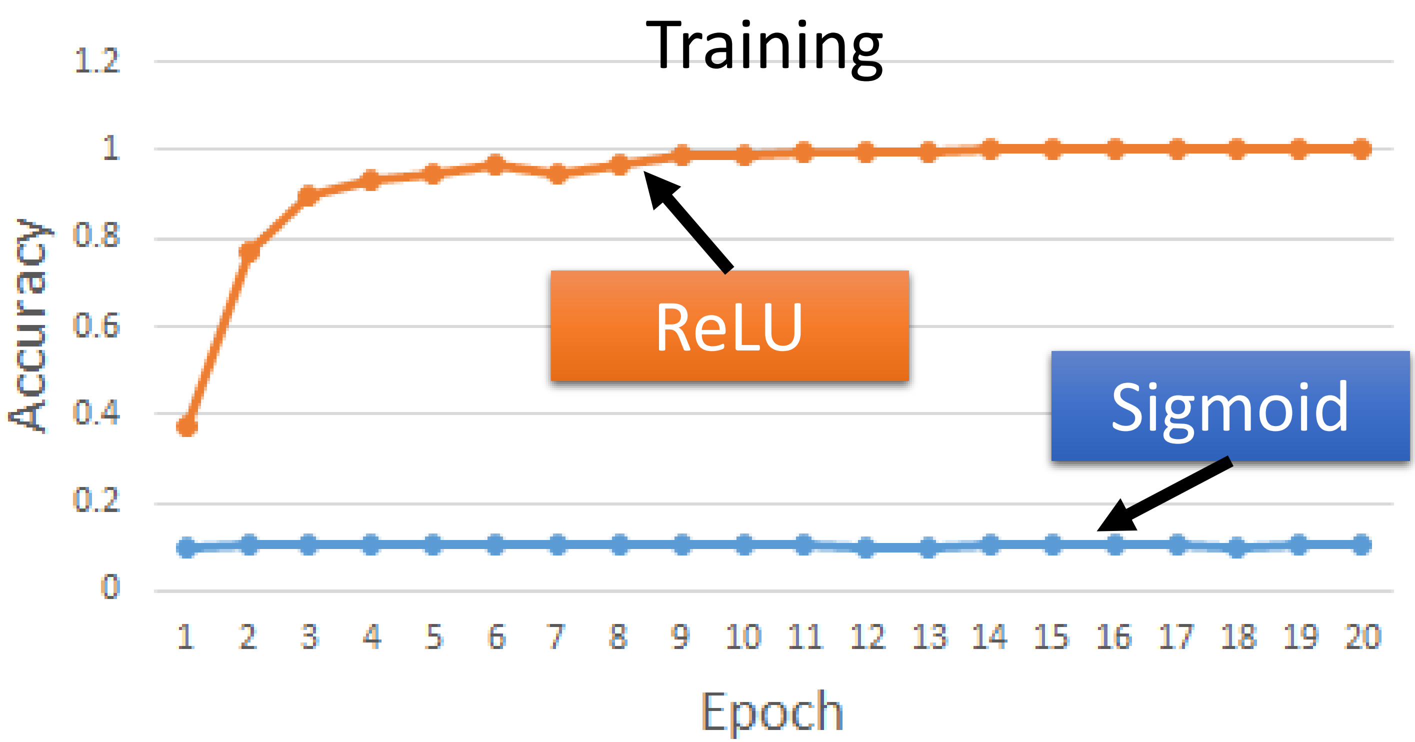
Rectified Linear Unit (ReLU)

Experiments

Testing:

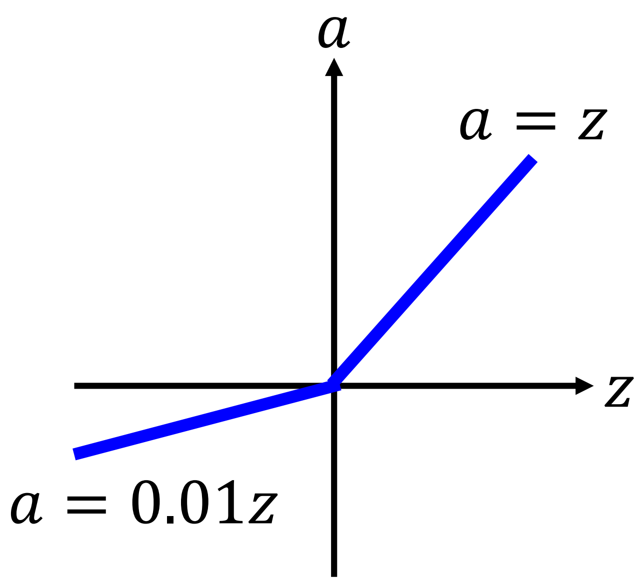
9 layers	Accuracy
Sigmoid	0.11
ReLU	0.96

- Hand-writing Digit Classification
  - 9 layers

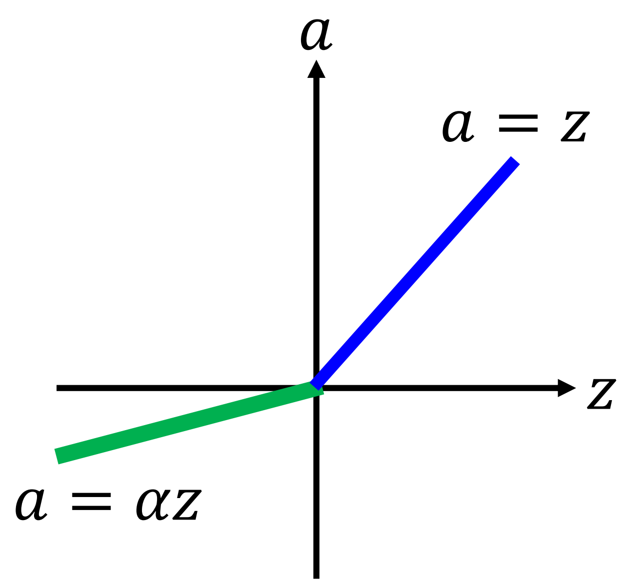


ReLU - variant

$Leaky\ ReLU$



$Parametric\ ReLU$

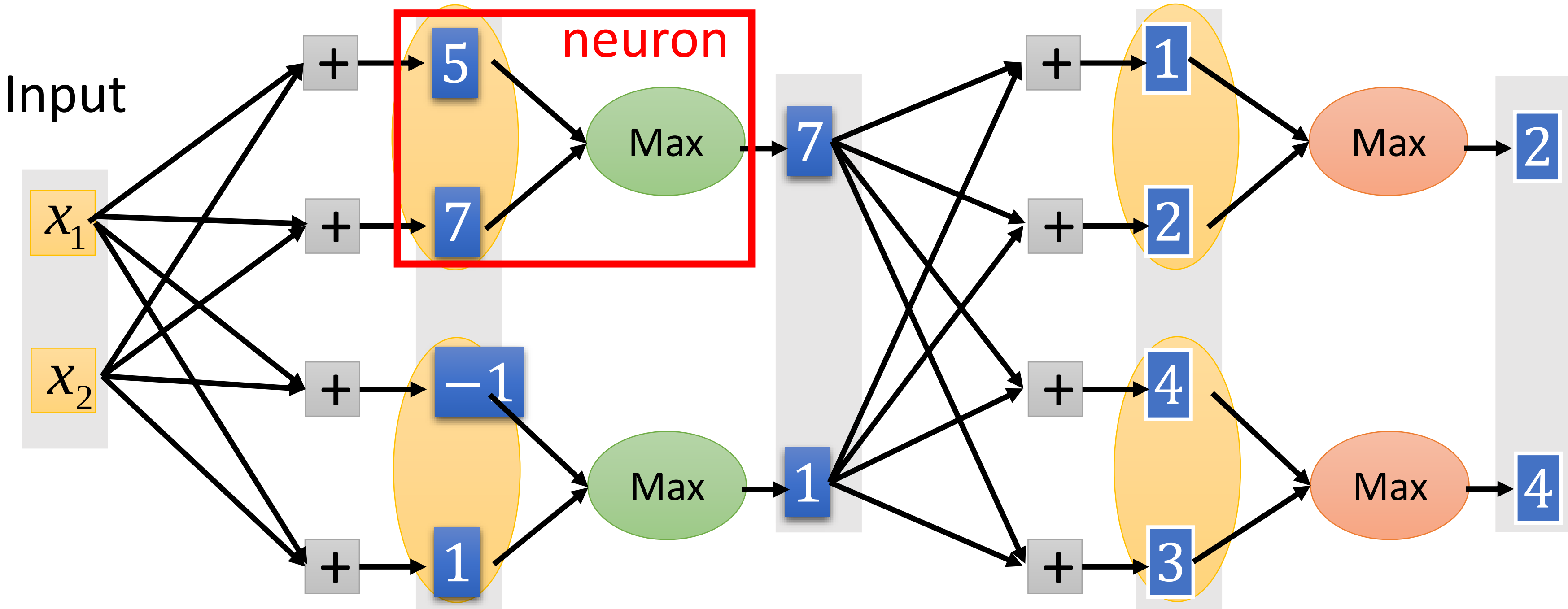


α also learned by gradient descent

Maxout

ReLU is a special case of Maxout

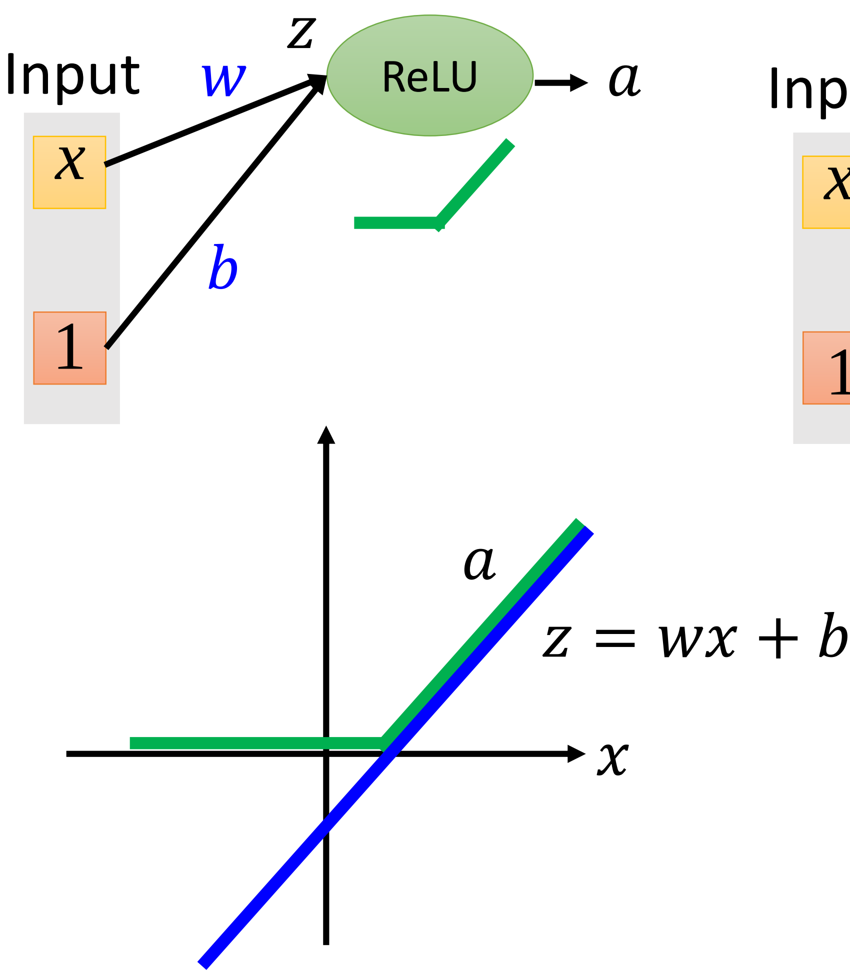
- Learnable activation function [Ian J. Goodfellow, ICML'13]

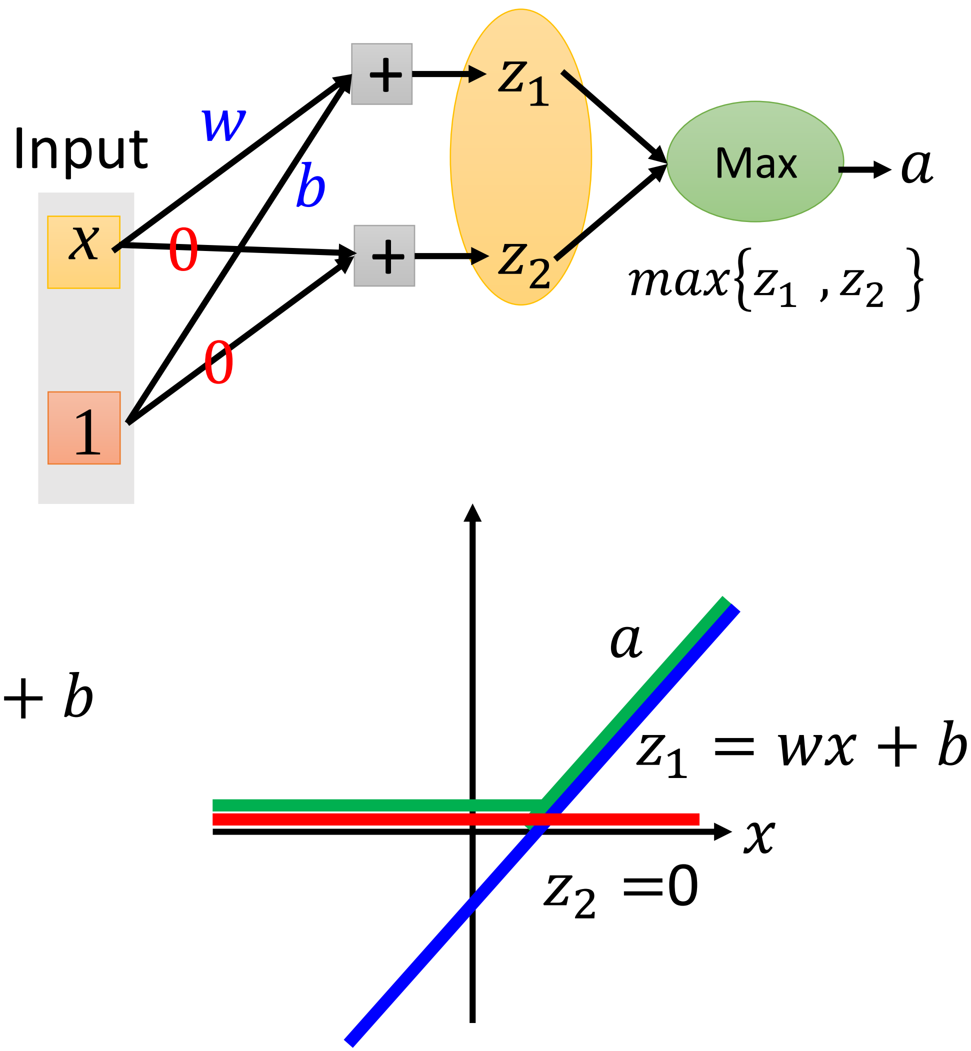


You can have more than 2 elements in a group.

Maxout

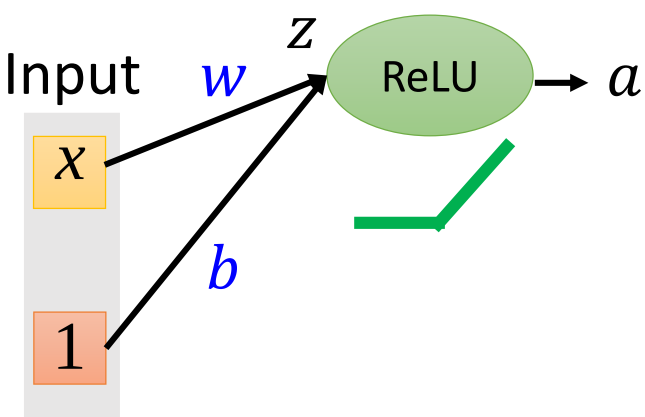
ReLU is a special case of Maxout

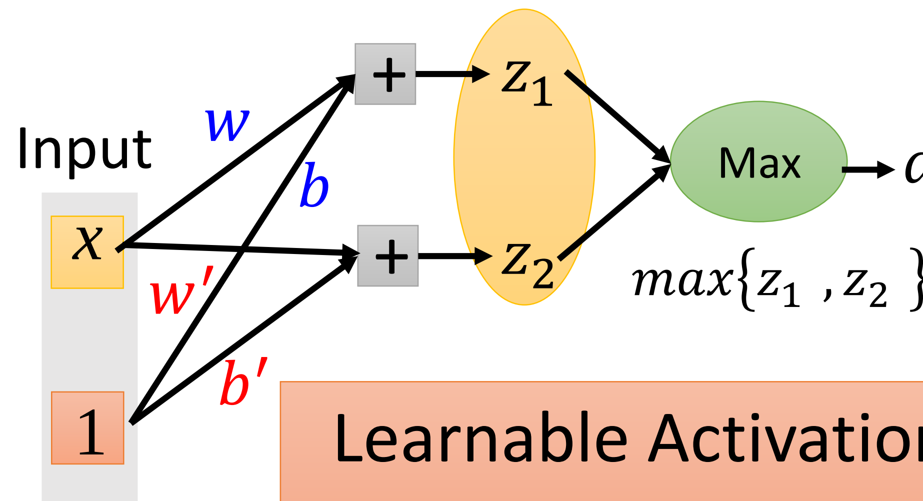




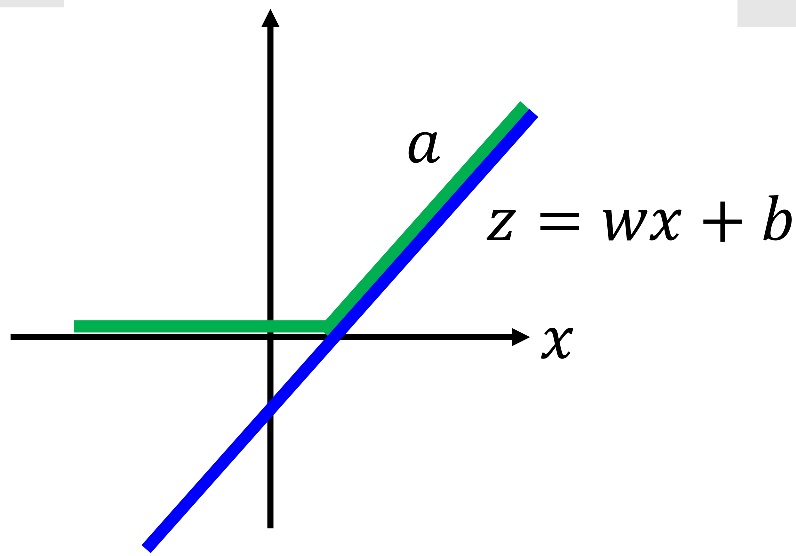
Maxout

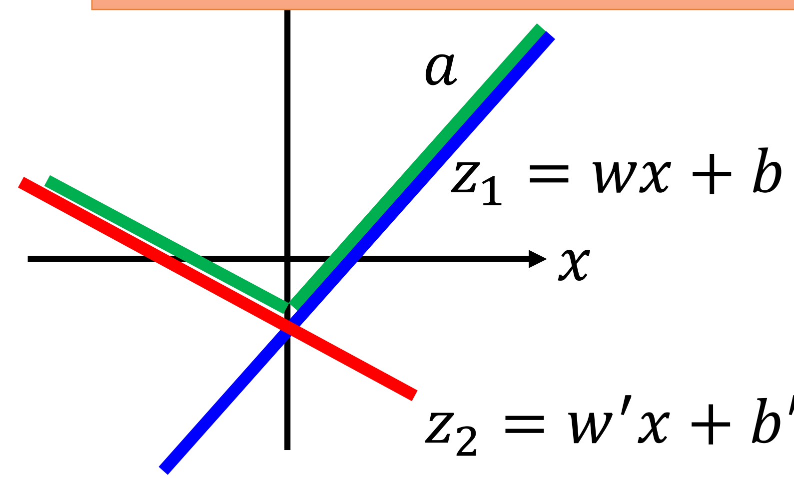
More than ReLU





Learnable Activation Function

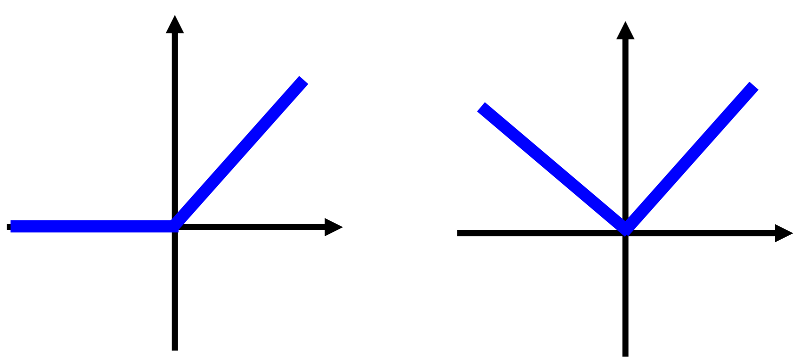




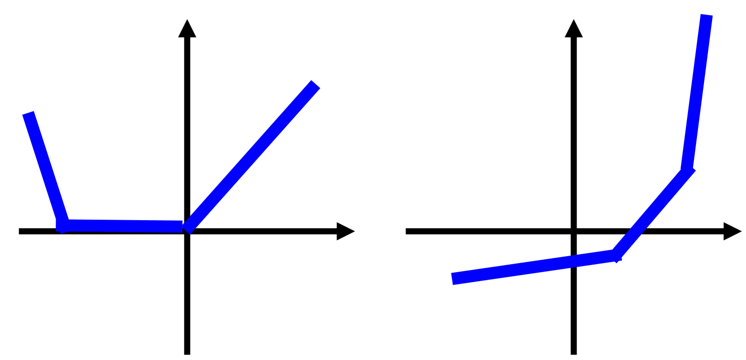
Maxout

- 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

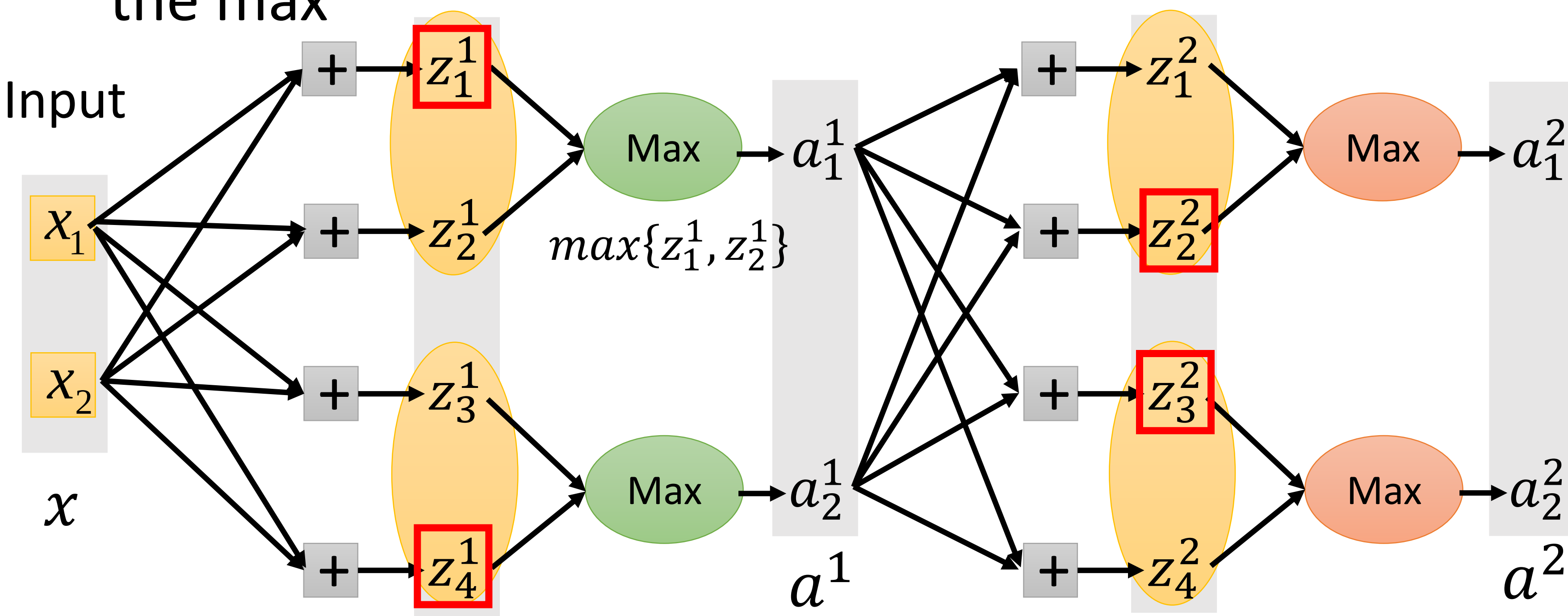


3 elements in a group



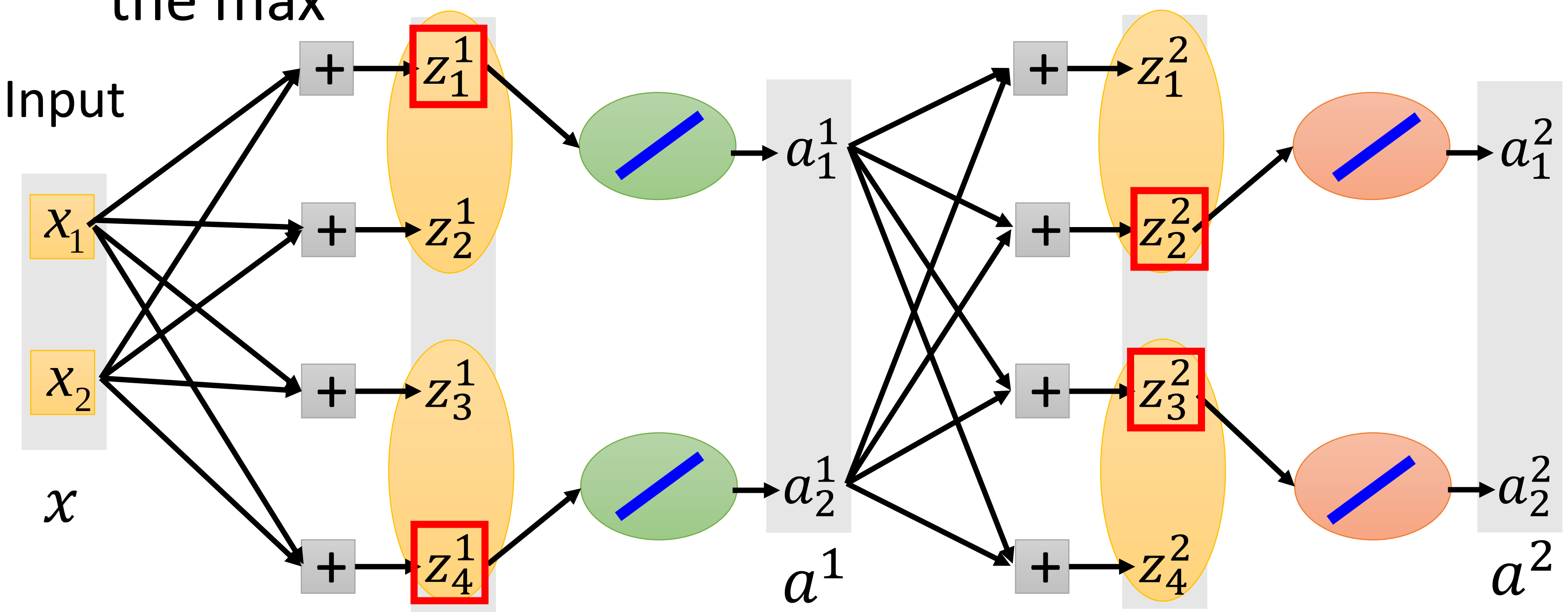
Maxout - Training

- Given a training data x, we know which z would be the max



Maxout - Training

- Given a training data x, we know which z would be the max



- Train this thin and linear network

Different thin and linear network for different examples

New Techniques

New Activation Function

- ReLU and Maxout network

New Structure

- Residue network and Highway network

For ultra deep network

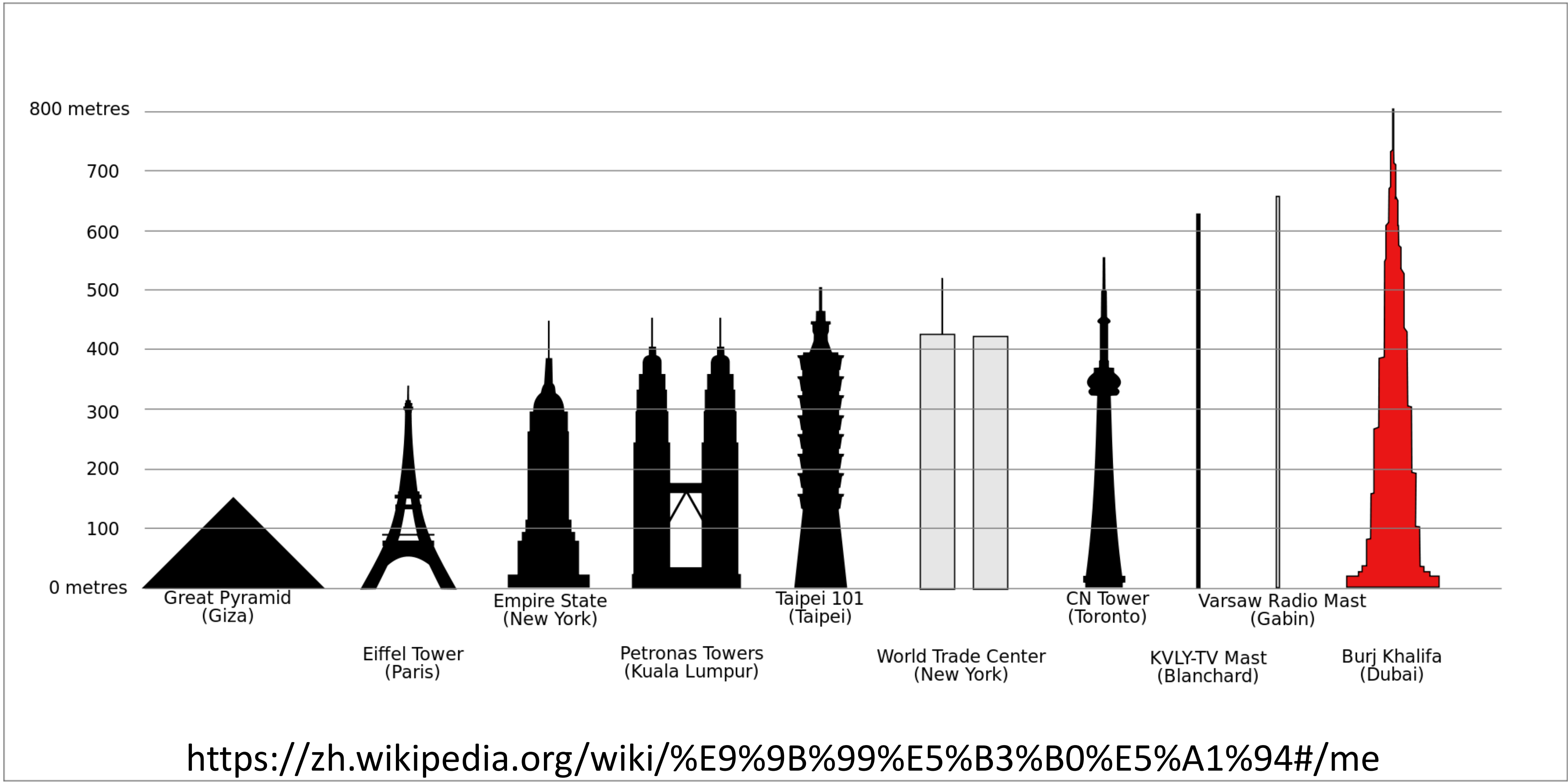
Better optimization Strategy

- E.g. Adam

Dropout

- Prevent Overfitting

Skyscraper



https://zh.wikipedia.org/wiki/%E9%9B%99%E5%B3%B0%E5%A1%94#/media/File:BurjDubaiHeight.svg

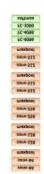
Ultra Deep Network

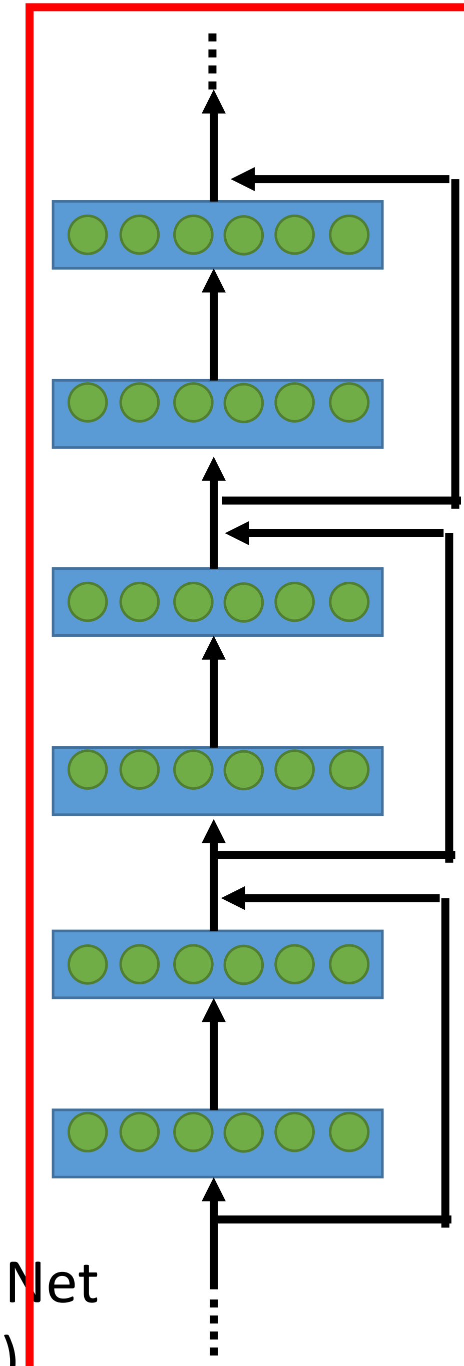
Worry about overfitting?

Worry about training first!

This ultra deep network have special structure.

152 layers





16.4% AlexNet (2012)

7.3% VGG (2014)

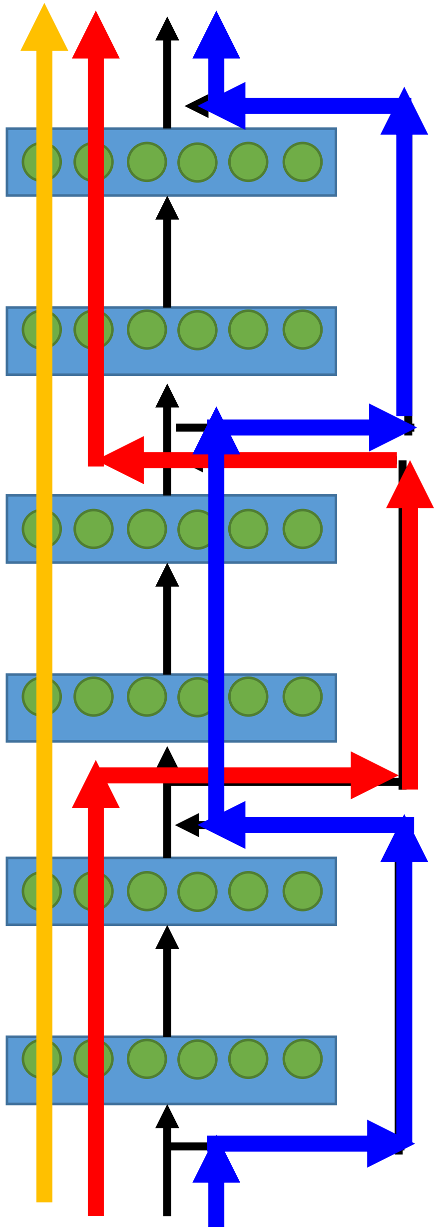
6.7% GoogleNet (2014)

3.57% Residual Net (2015)

Ultra Deep Network

- Ultra deep network is the ensemble of many networks with different depth.

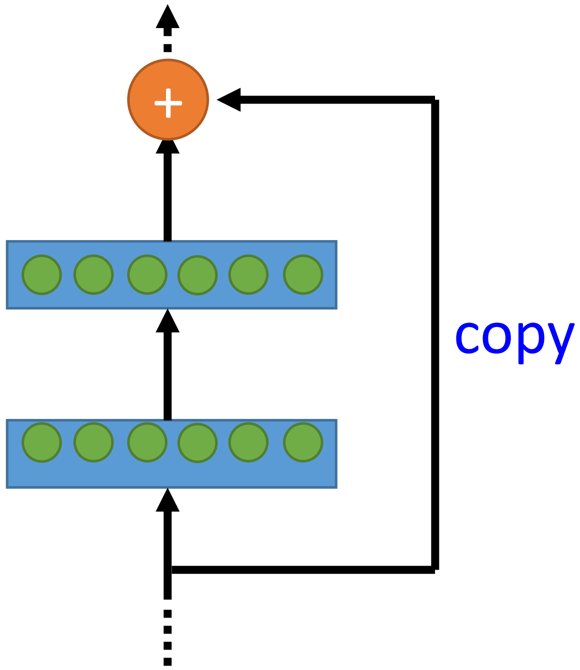
Ensemble { **6 layers**, **4 layers**, **2 layers** }

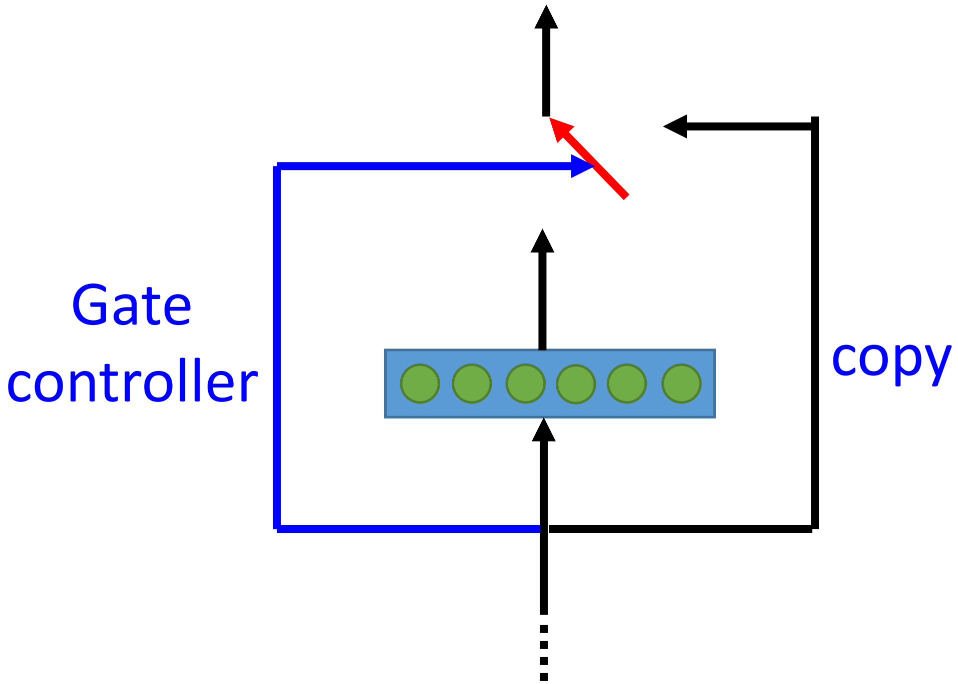


Residual Networks are Exponential Ensembles of Relatively Shallow Networks https://arxiv.org/abs/1605.06431

Ultra Deep Network

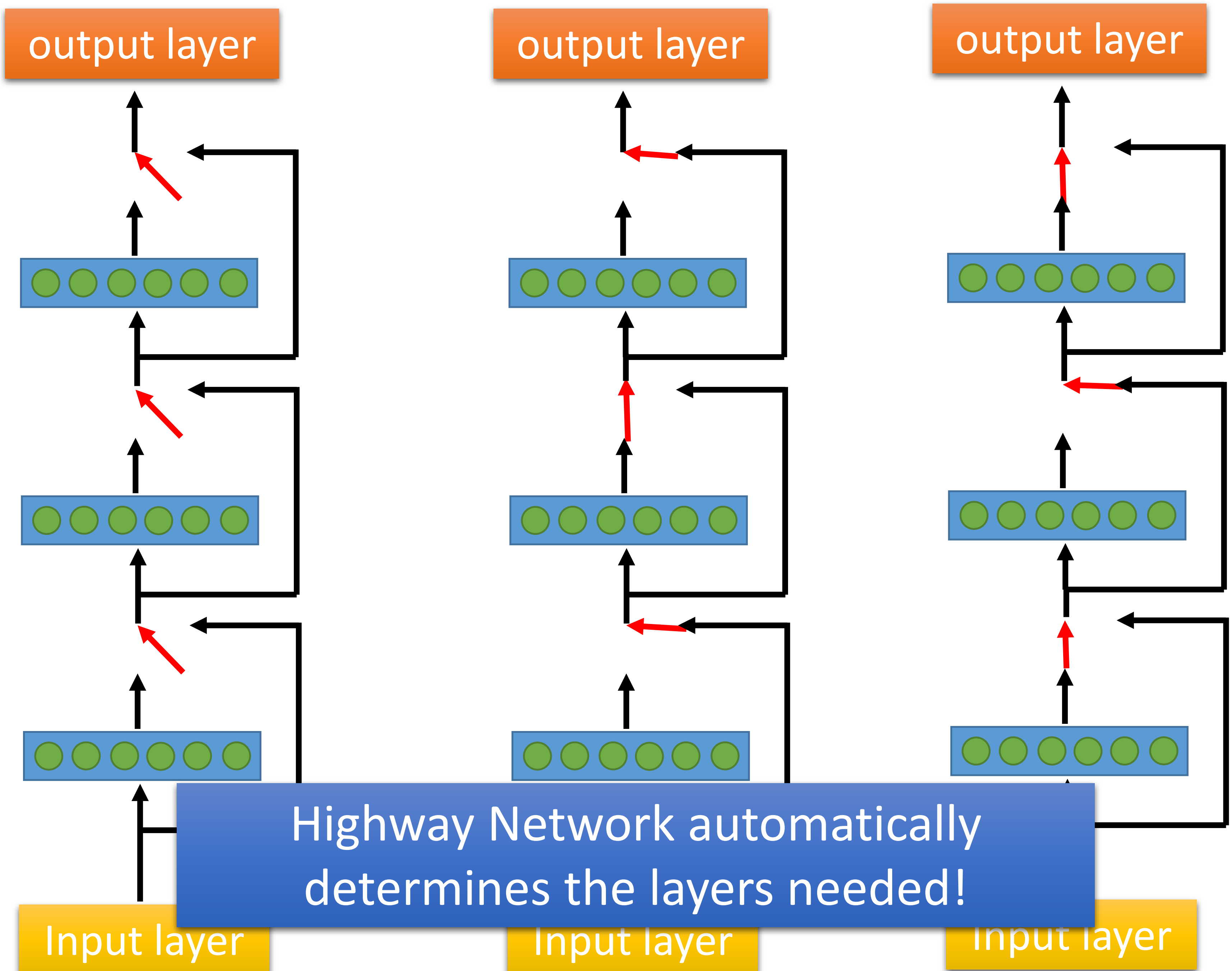
- **Residual Network**
- **Highway Network**





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

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



Highway Network automatically determines the layers needed!

New Techniques

New Activation Function

- ReLU and Maxout network

New Structure

- Residue network and Highway network

Better optimization Strategy

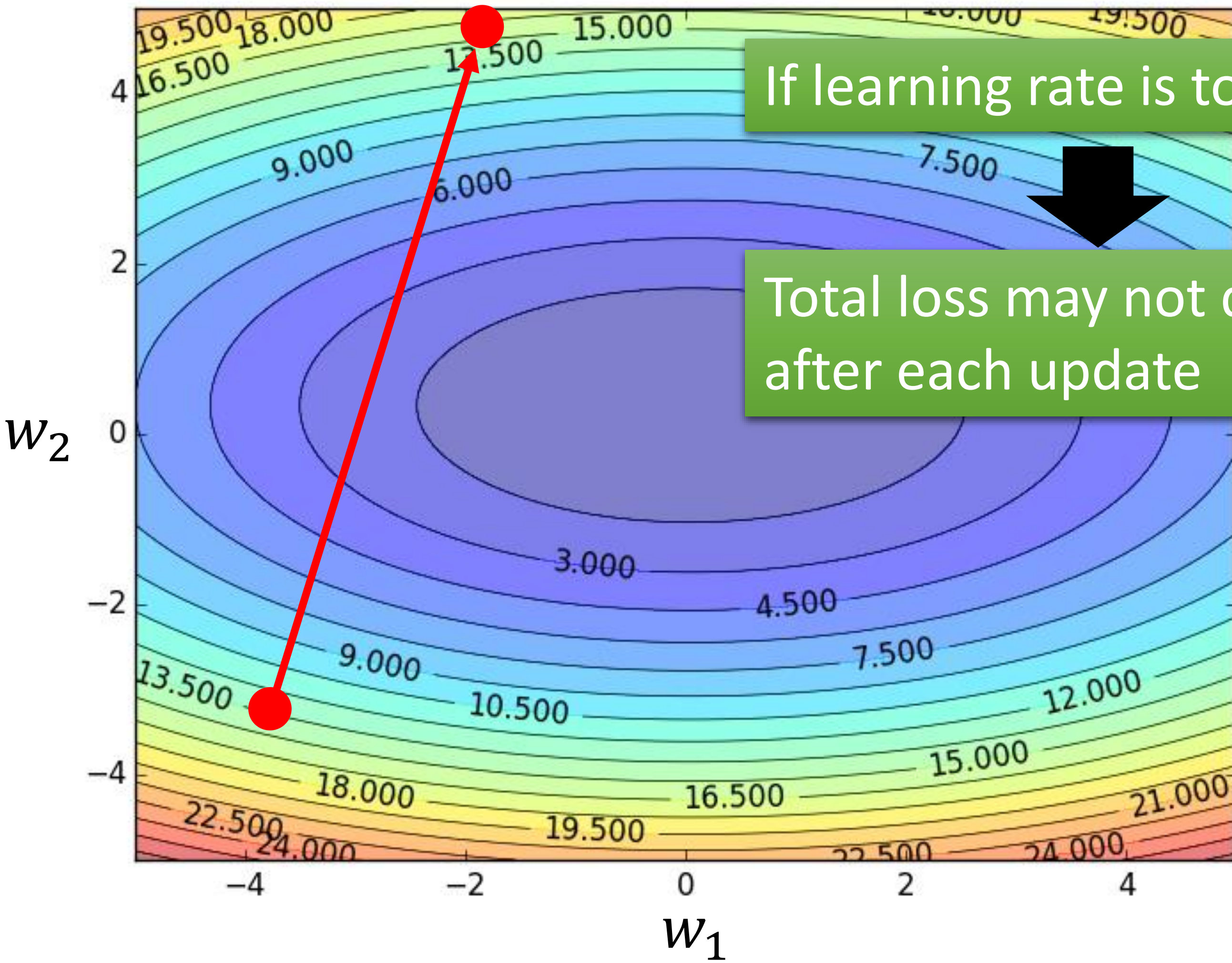
- E.g. Adam

Dropout

- Prevent Overfitting

Learning Rates

Set the learning rate η carefully

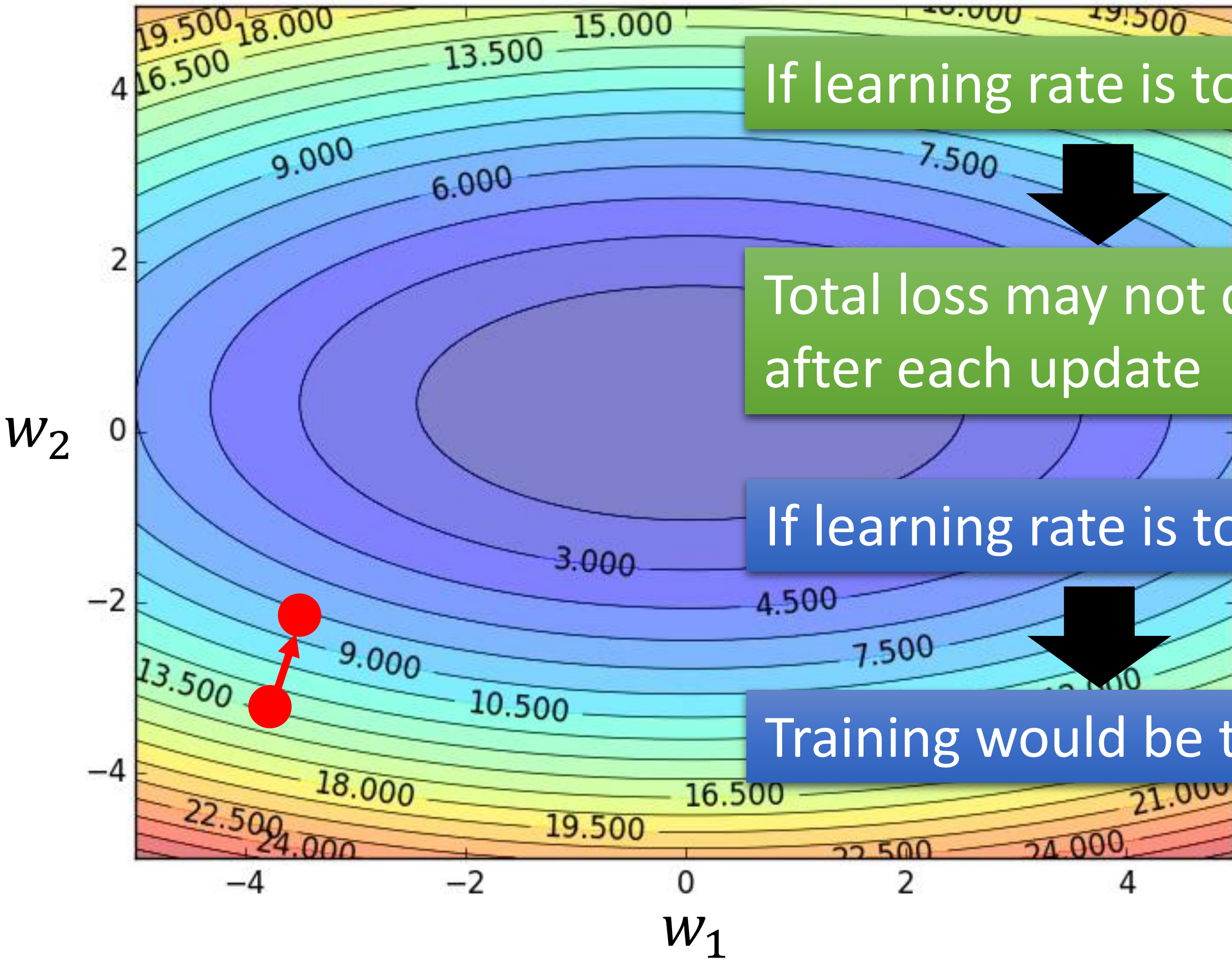


If learning rate is too large

Total loss may not decrease after each update

Learning Rates

Set the learning rate η carefully



If learning rate is too large

Total loss may not decrease after each update

If learning rate is too small

Training would be too slow

Learning Rates

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

Adagrad

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

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

Parameter dependent learning rate

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

constant

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

Summation of the square of the previous derivatives

Adagrad

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

w_1	g^0
	0.1

w_2	g^0
	20.0

Learning rate:

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

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

Learning rate:

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

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

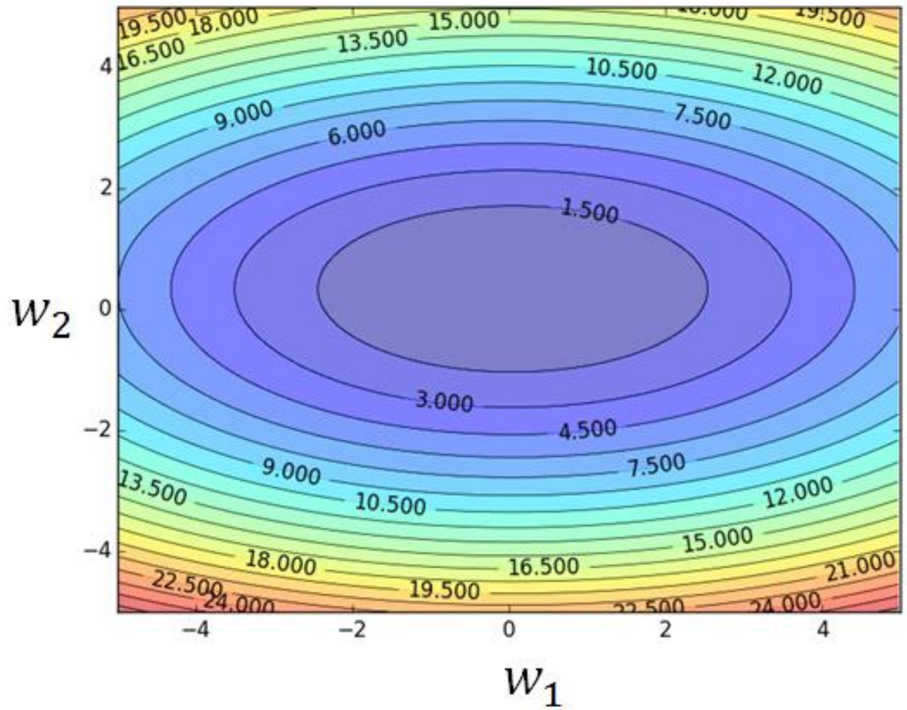
Observation: 1. Learning rate is smaller and smaller for all parameters

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

Why?

Larger derivatives

Smaller Learning Rate



Smaller Derivatives

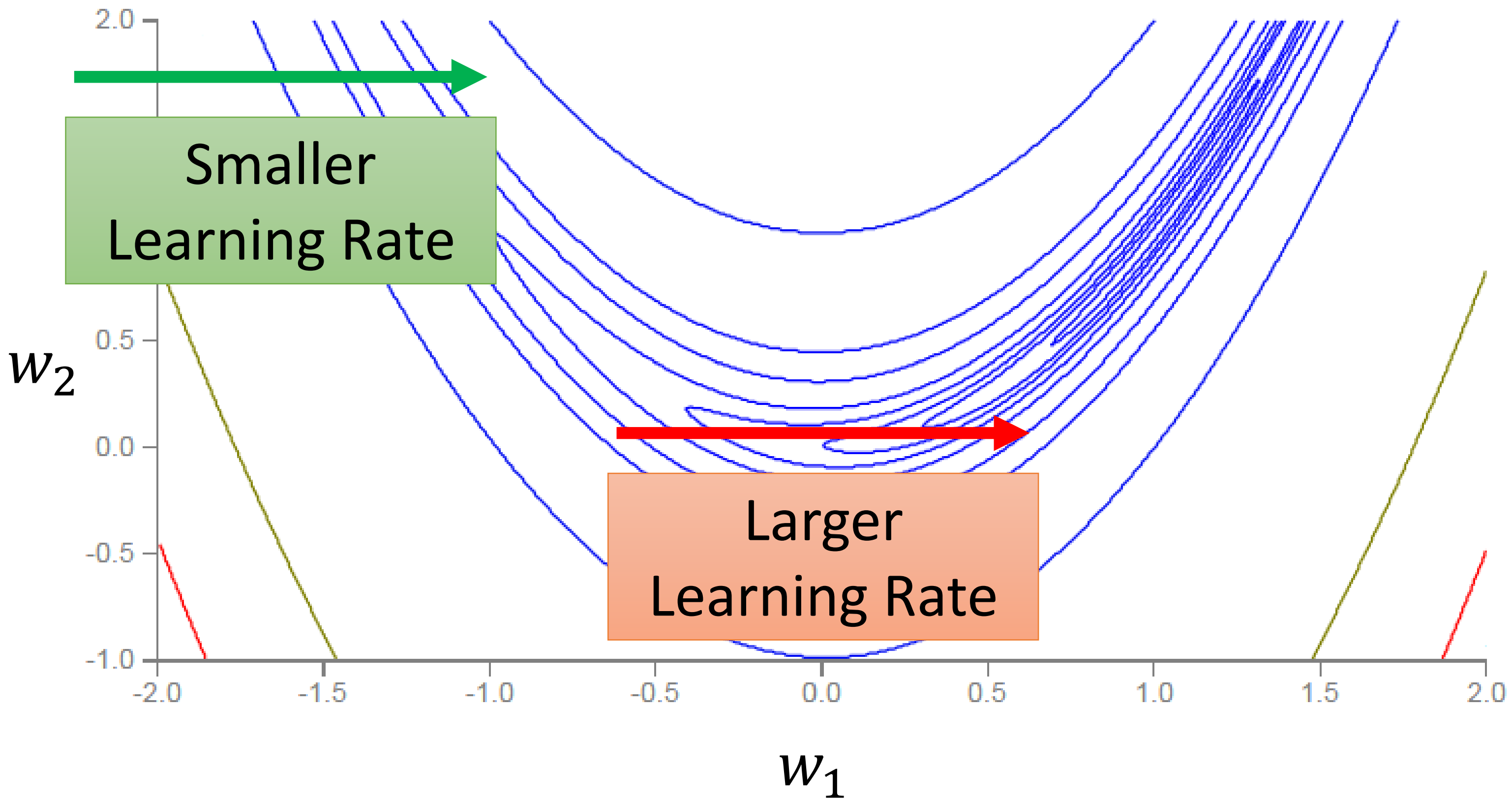
Larger Learning Rate

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

Why?

RMSProp

Error Surface can be very complex when training NN.



RMSProp

$$w^1 \leftarrow w^0 - \frac{\eta}{\sigma^0} g^0 \qquad \sigma^0 = g^0$$

$$w^2 \leftarrow w^1 - \frac{\eta}{\sigma^1} g^1 \qquad \sigma^1 = \sqrt{\alpha(\sigma^0)^2 + (1-\alpha)(g^1)^2}$$

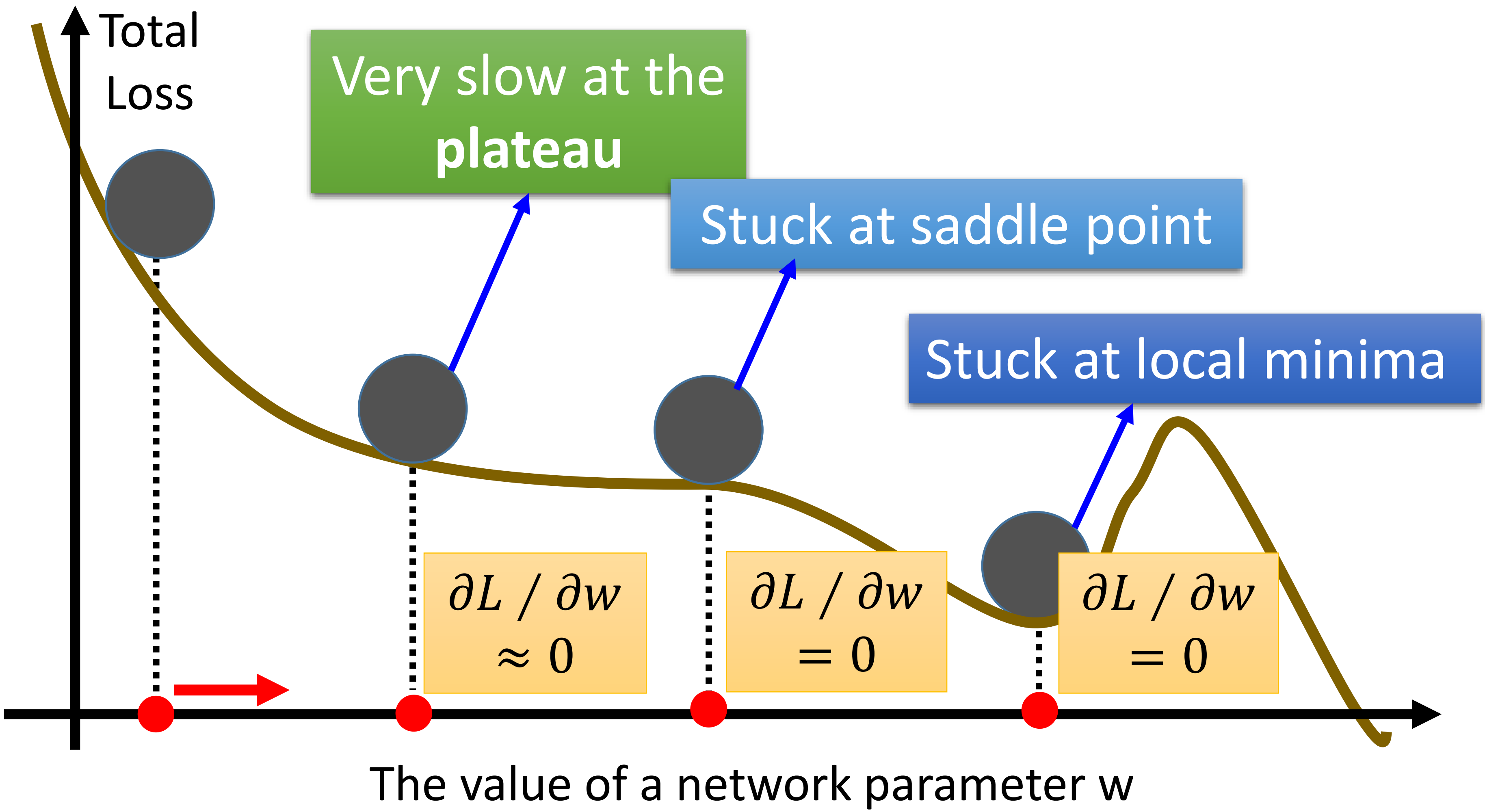
$$w^3 \leftarrow w^2 - \frac{\eta}{\sigma^2} g^2 \qquad \sigma^2 = \sqrt{\alpha(\sigma^1)^2 + (1-\alpha)(g^2)^2}$$

$$\vdots$$

$$w^{t+1} \leftarrow w^t - \frac{\eta}{\sigma^t} g^t \qquad \sigma^t = \sqrt{\alpha(\sigma^{t-1})^2 + (1-\alpha)(g^t)^2}$$

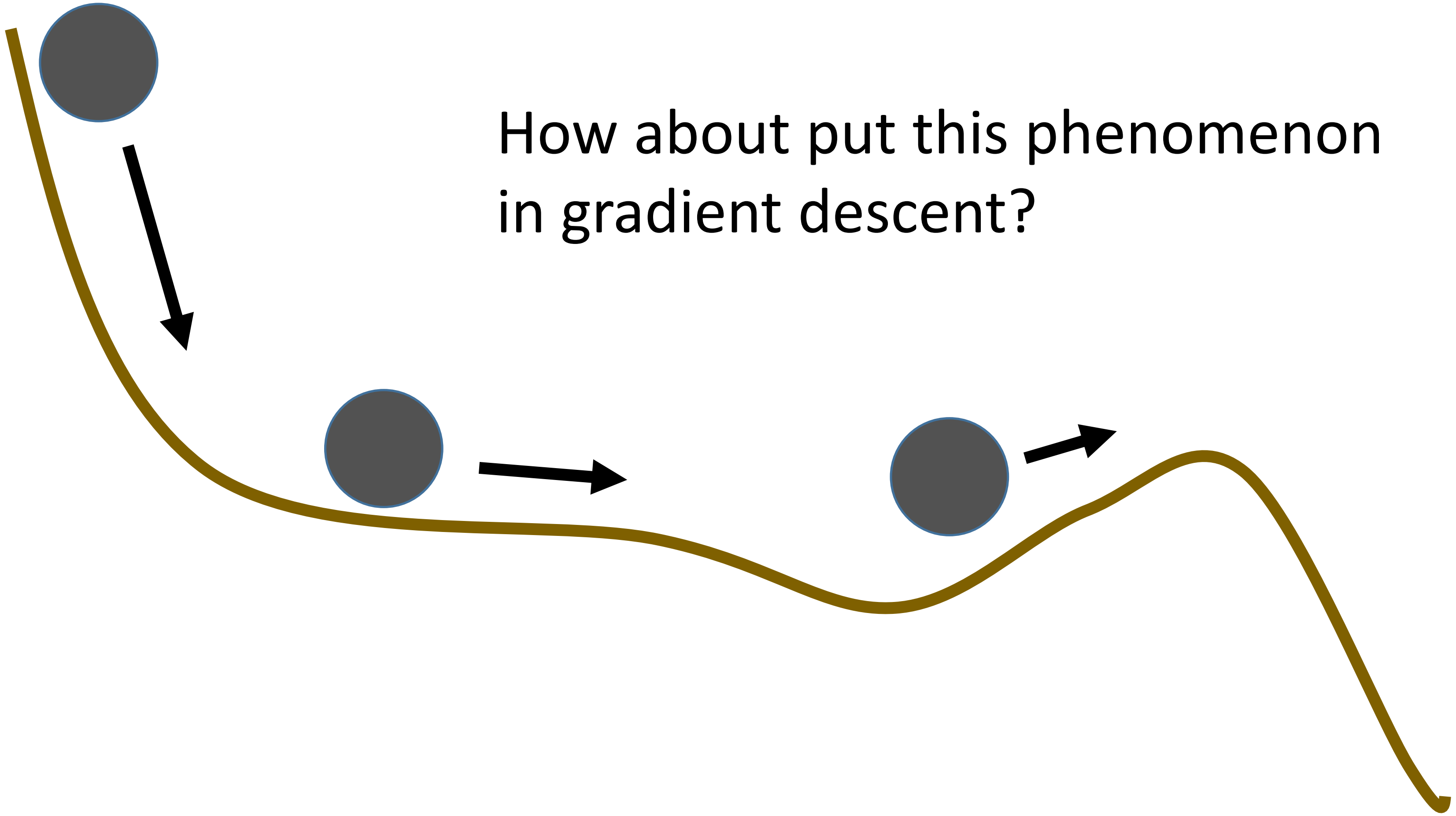
Root Mean Square of the gradients
with previous gradients being decayed

Hard to find optimal network parameters



In physical world ……

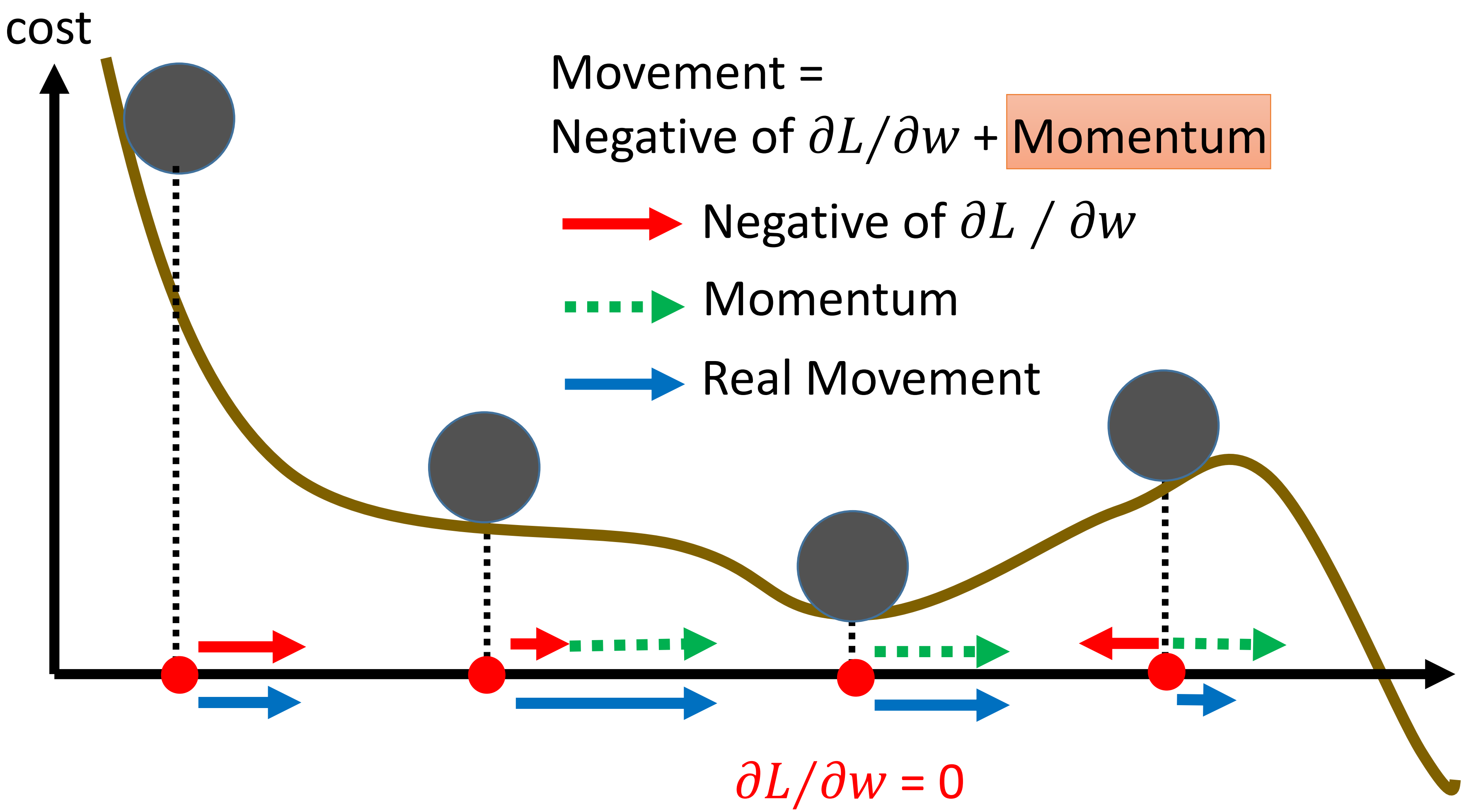
- Momentum



How about put this phenomenon in gradient descent?

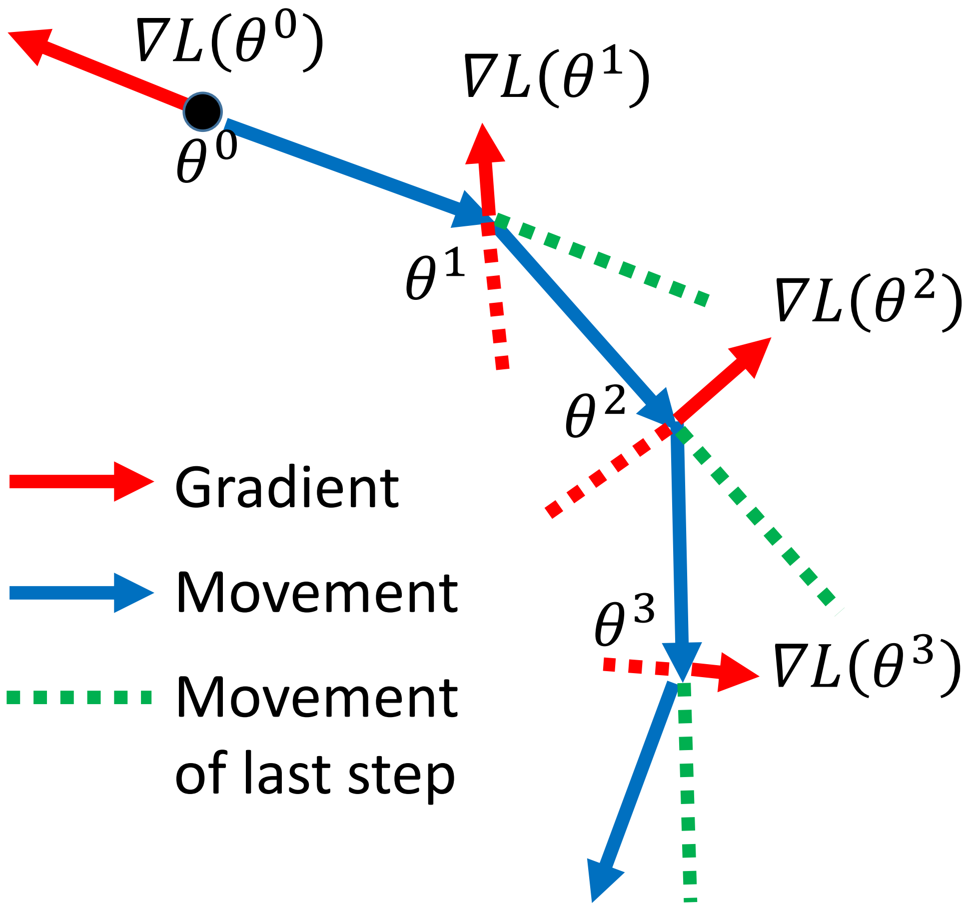
Momentum

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



Momentum

Movement: movement of last step minus gradient at present



Start at point θ^0

Movement $v^0=0$

Compute gradient at θ^0

Movement $v^1 = \lambda v^0 - \eta \nabla L(\theta^0)$

Move to $\theta^1 = \theta^0 + v^1$

Compute gradient at θ^1

Movement $v^2 = \lambda v^1 - \eta \nabla L(\theta^1)$

Move to $\theta^2 = \theta^1 + v^2$

Movement not just based on gradient, but previous movement.

Adam

RMSProp + Momentum

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.

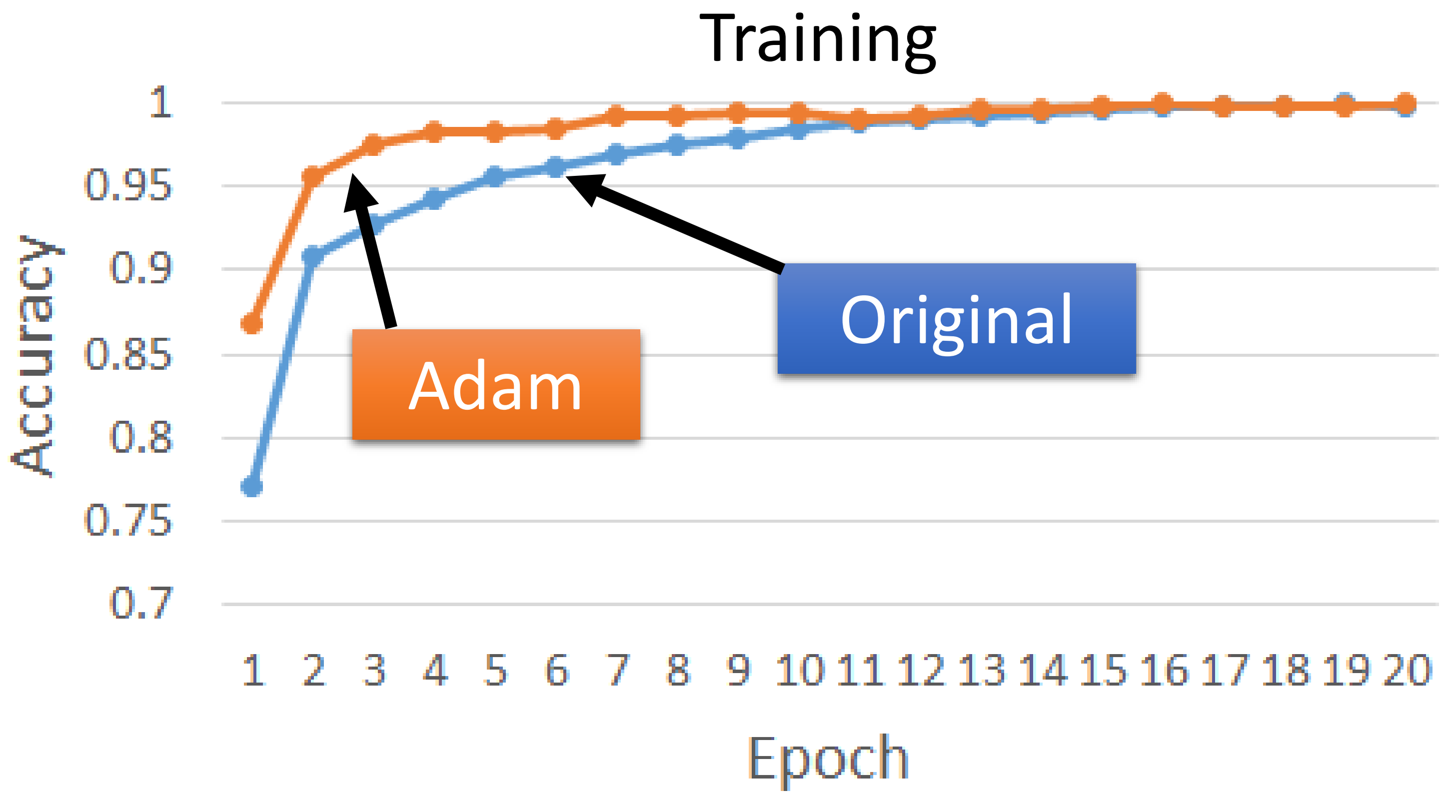
Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1st moment vector) → for momentum
  $v_0 \leftarrow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

Experiments

Testing:

	Accuracy
Original	0.96
Adam	0.97

- Hand-writing Digit Classification
  - ReLU, 3 layer



New Techniques

New Activation Function

- ReLU and Maxout network

New Structure

- Residue network and Highway network

Better optimization Strategy

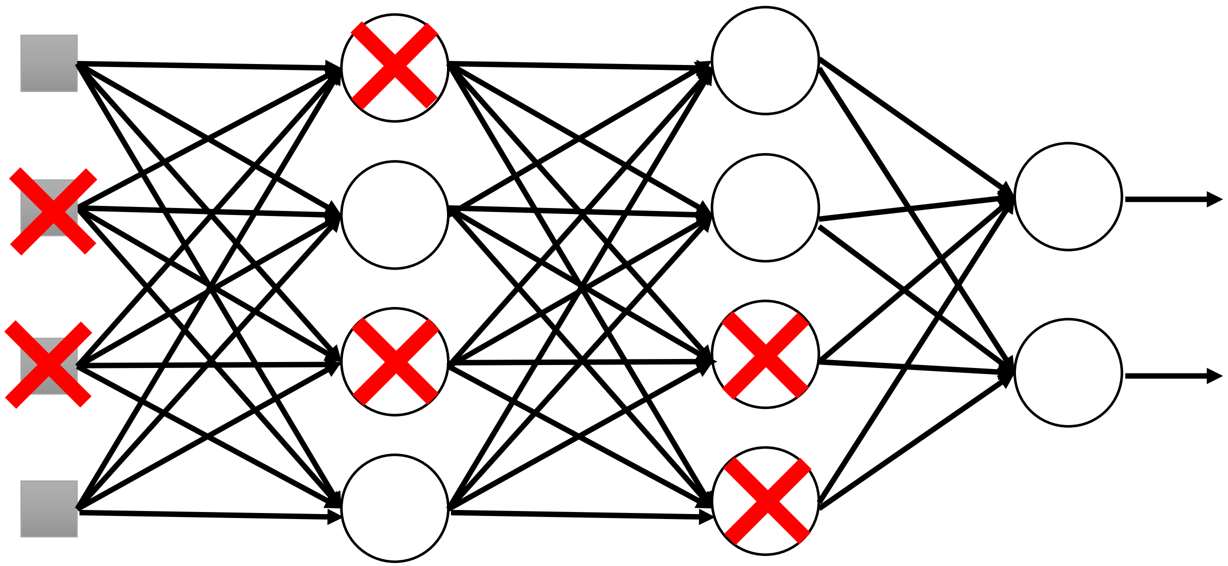
- E.g. Adam

Dropout

- Prevent Overfitting

Dropout

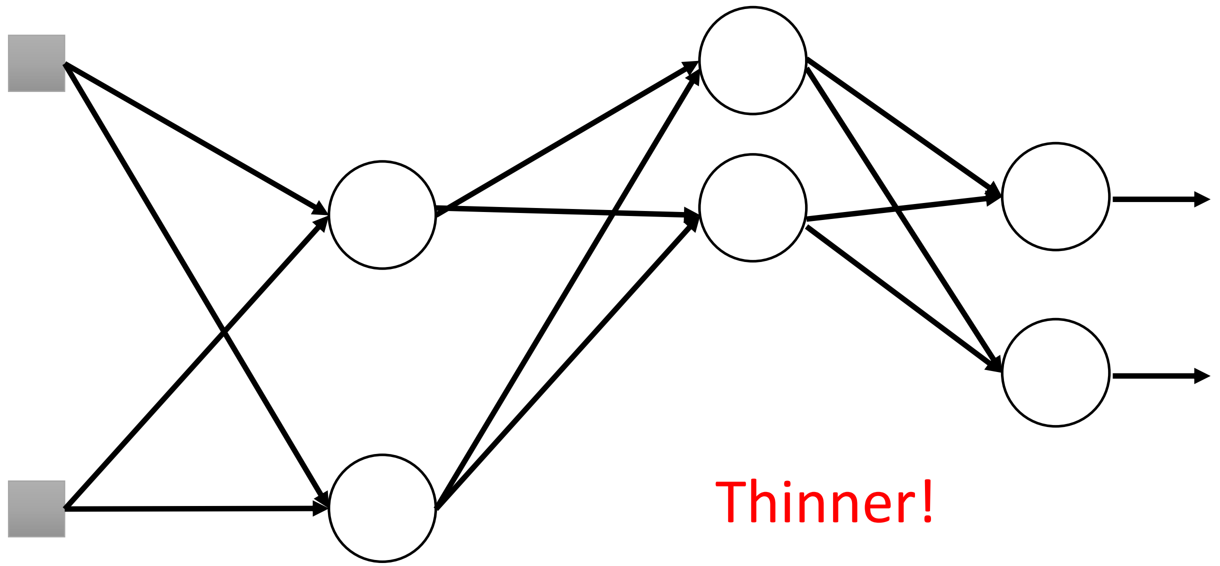
Training:



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

Dropout

Training:

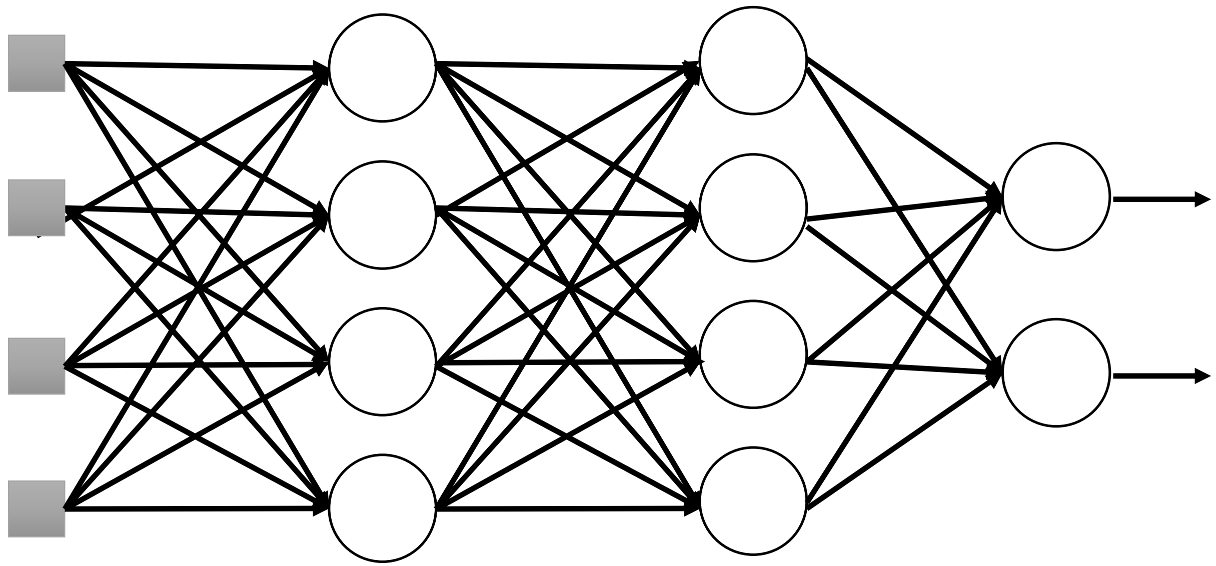


- **Each time before updating the parameters**
  - Each neuron has p% to dropout
    - ⇒ **The structure of the network is changed.**
  - Using the new network for training

For each mini-batch, we resample the dropout neurons

Dropout

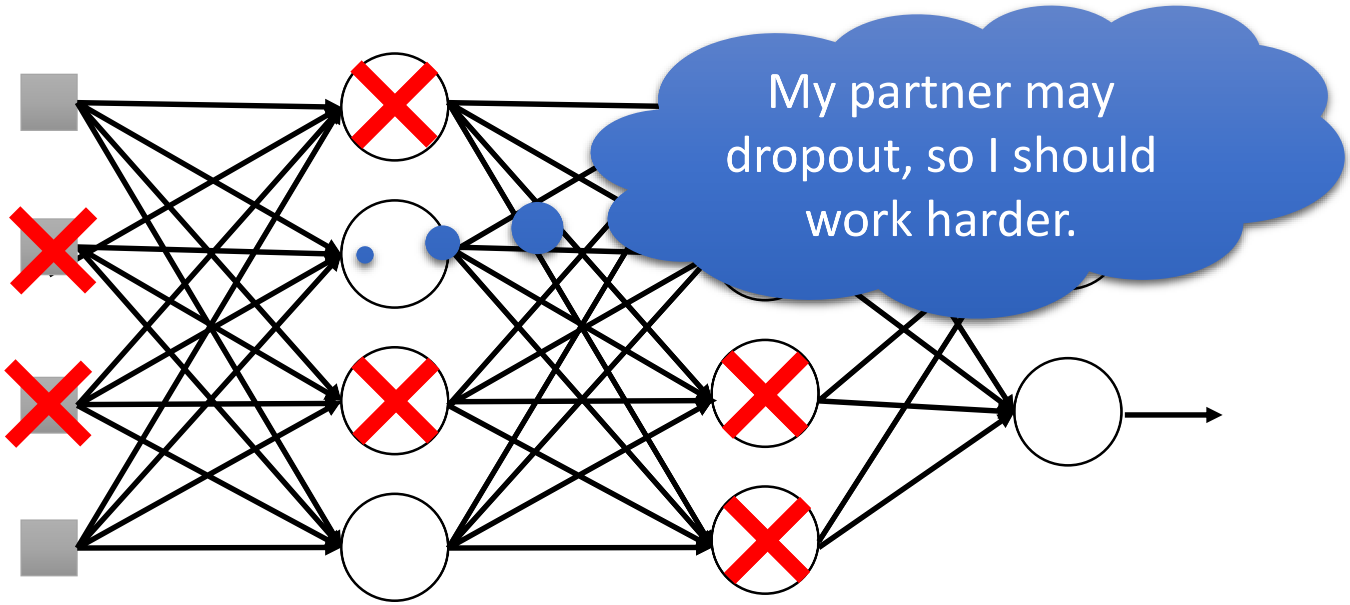
Testing:



➢ **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



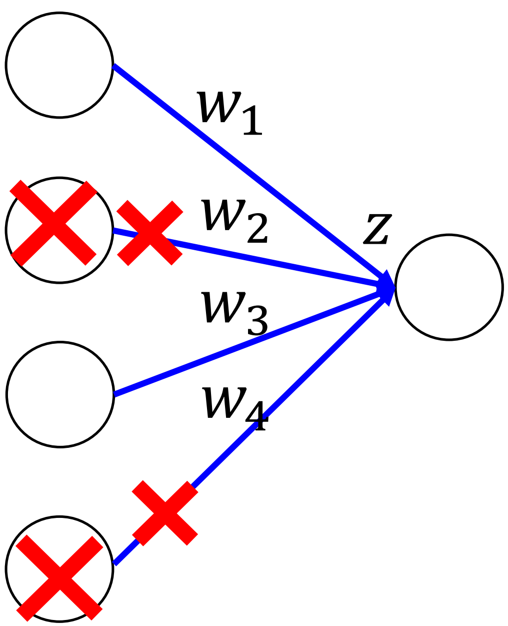
- When teams up, if everyone expect the partner will do the work, nothing will be done finally.
- However, if you know your partner will slack off (dropout), you will do better.
- When testing, no one dropout actually, so obtaining good results eventually.

Dropout - Intuitive Reason

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

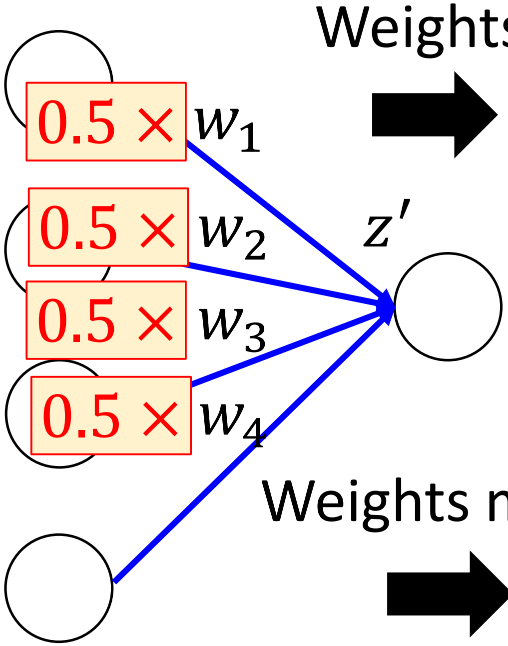
Training of Dropout

Assume dropout rate is 50%



Testing of Dropout

No dropout



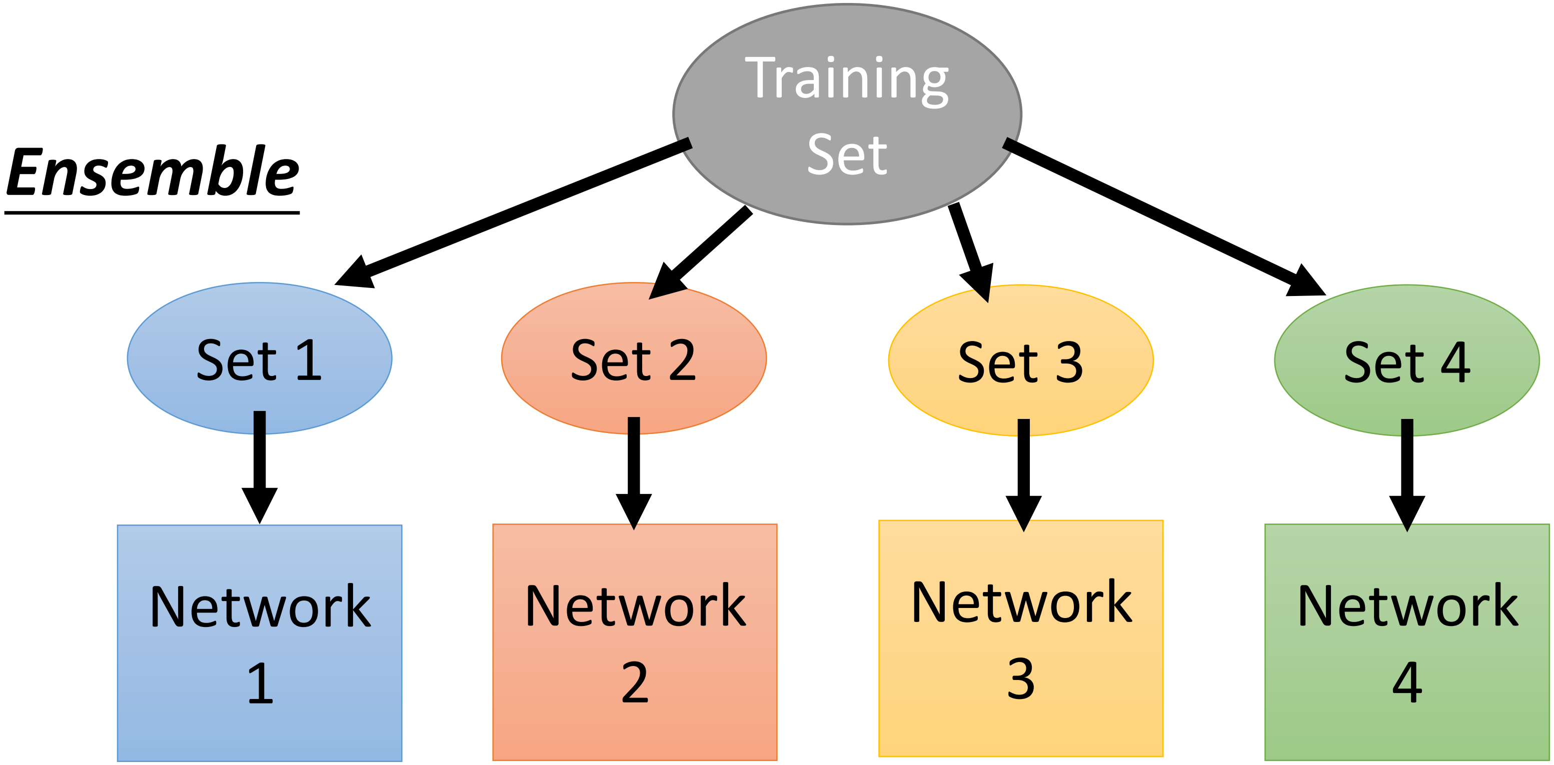
Weights from training

$z' \approx 2z$

Weights multiply 1-p%

$z' \approx z$

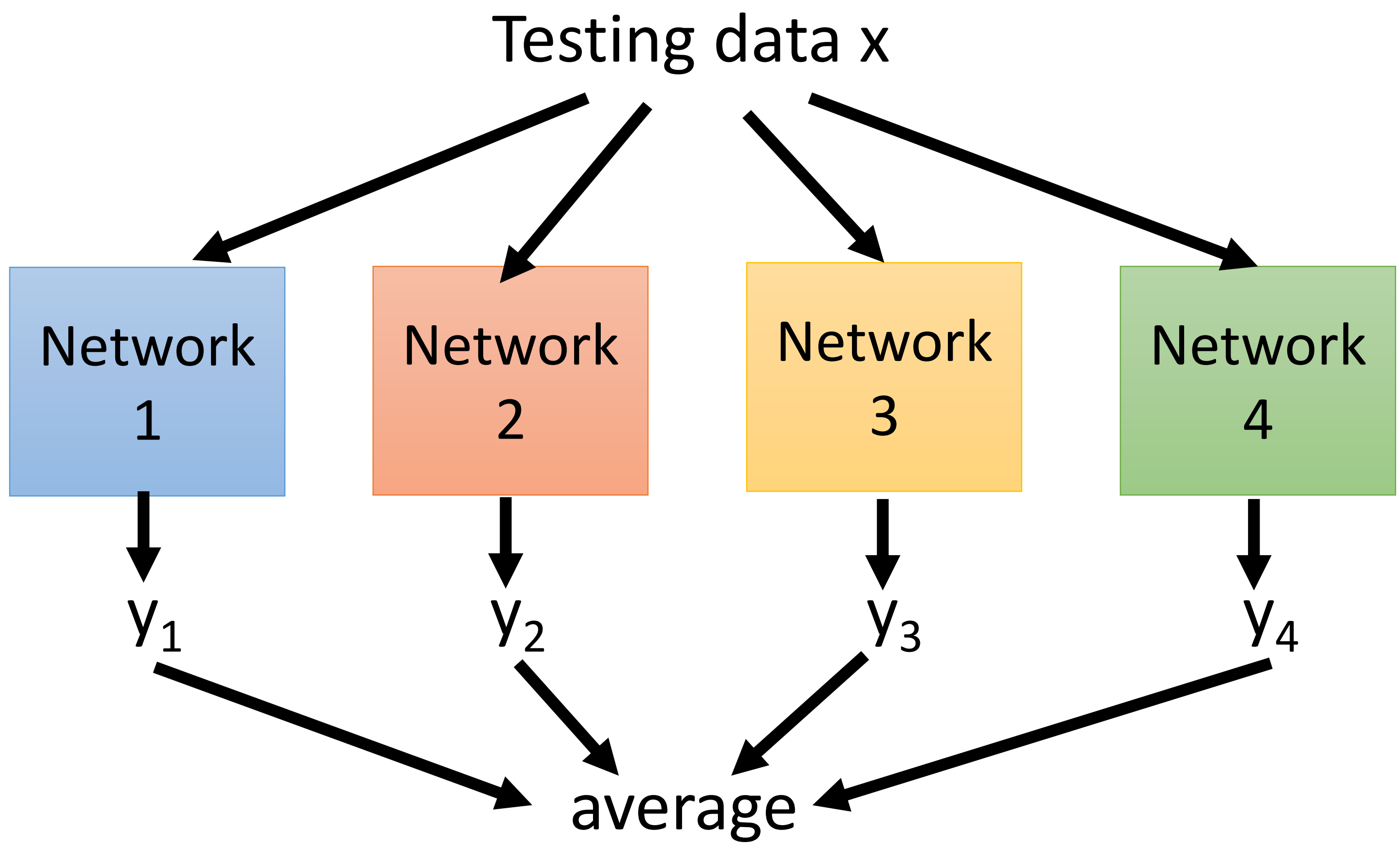
Dropout is a kind of ensemble.



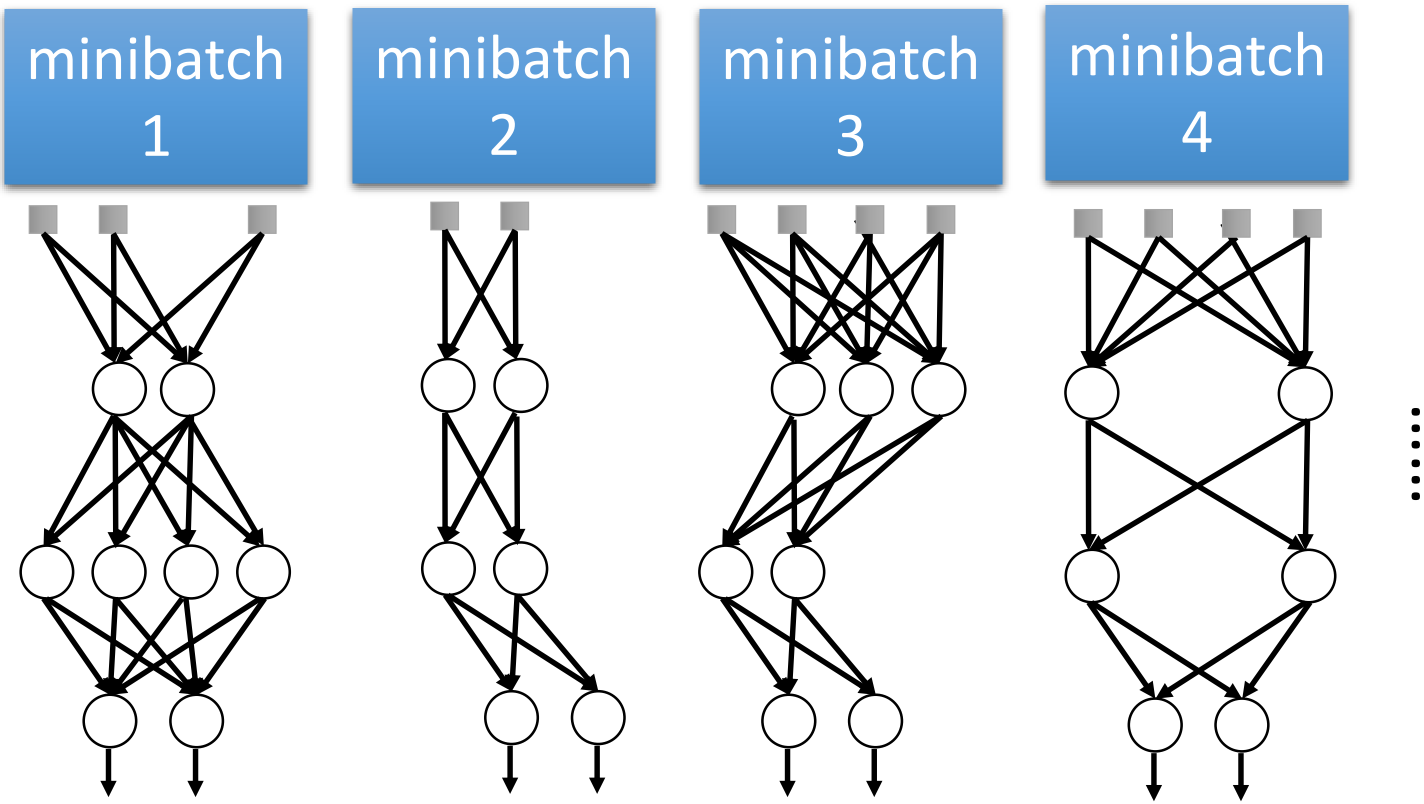
Train a bunch of networks with different structures

Dropout is a kind of ensemble.

Ensemble

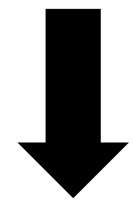


Dropout is a kind of ensemble.



Training of Dropout

M neurons

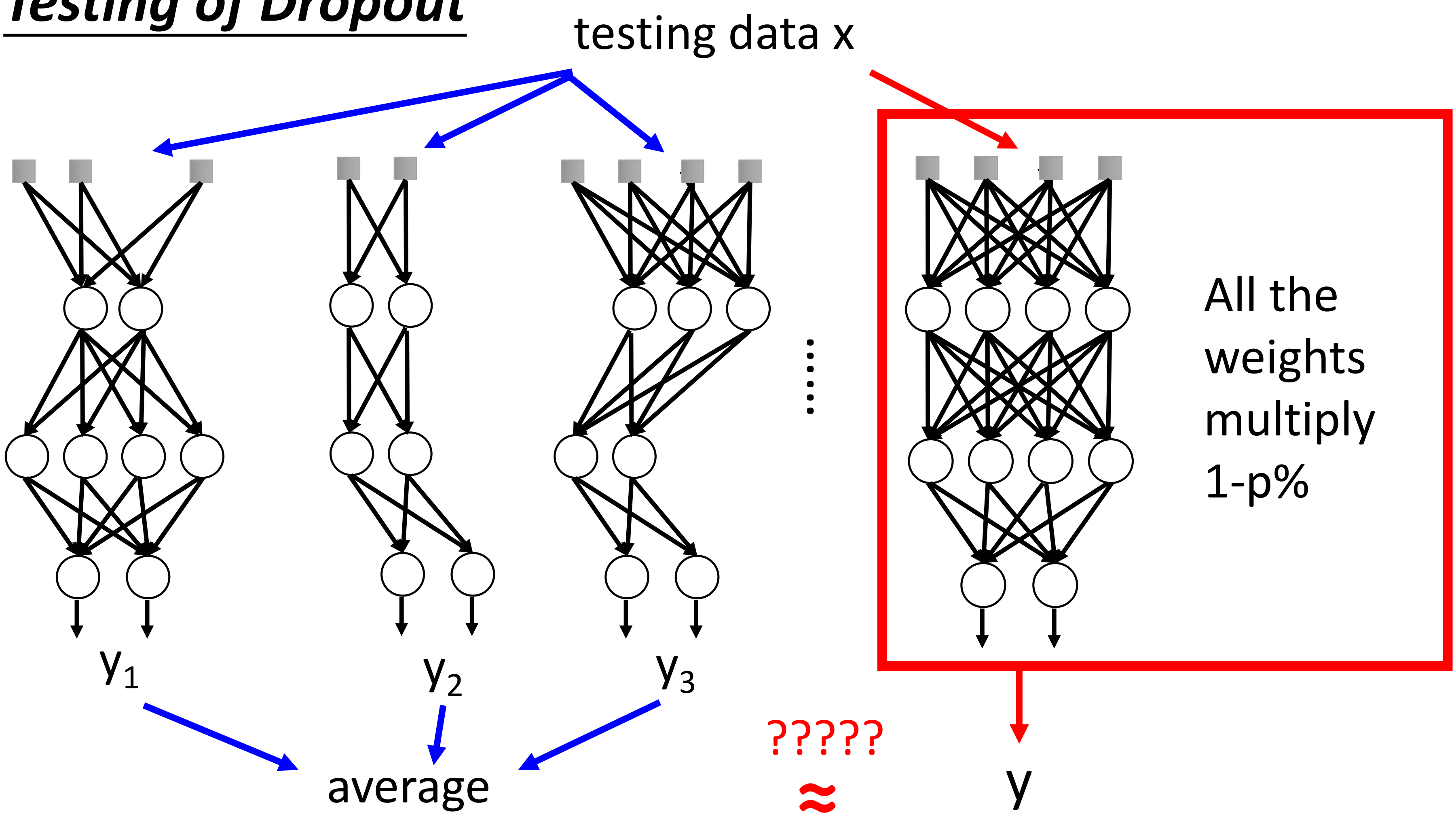


2^M possible networks

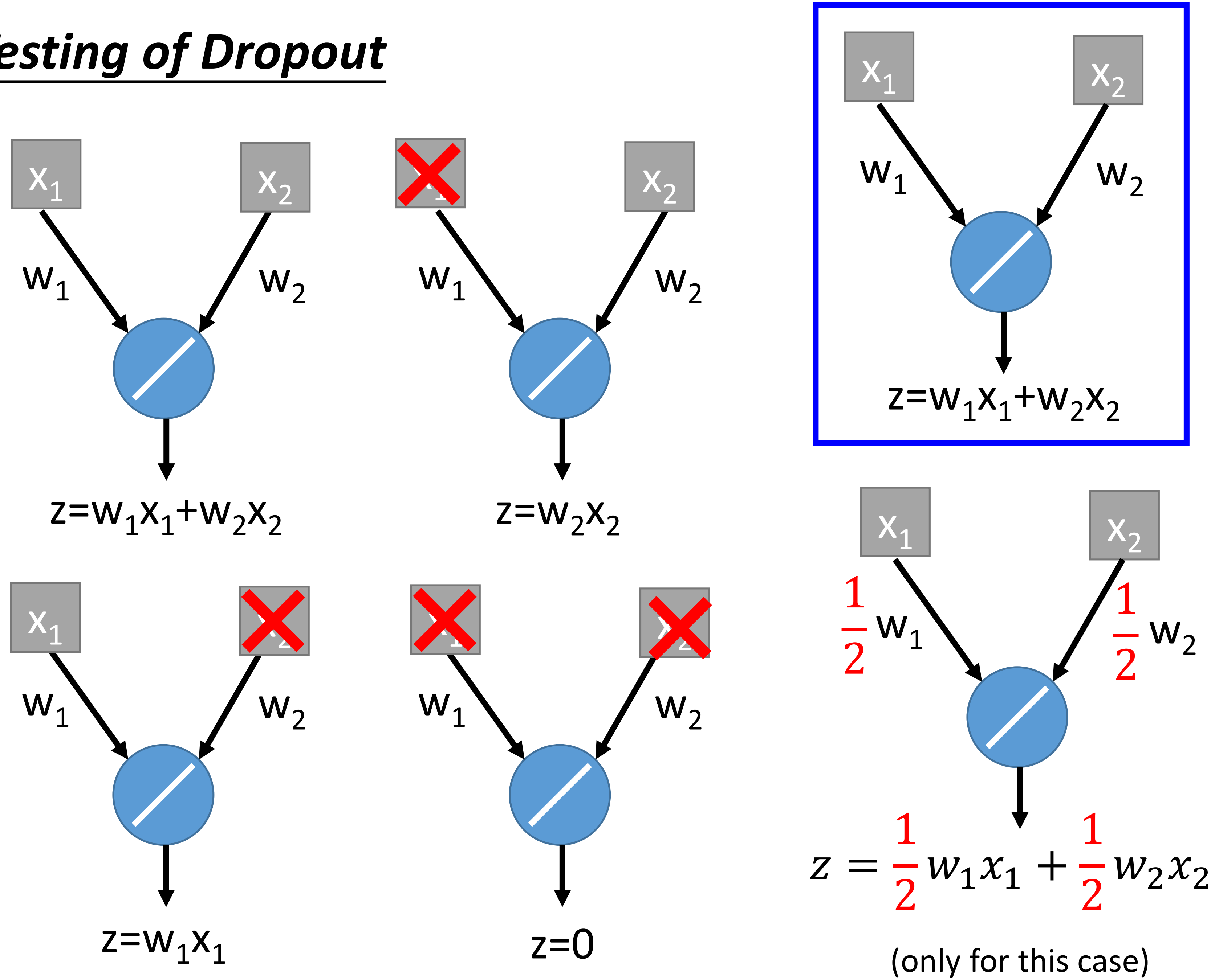
- ➢Using one mini-batch to train one network
- ➢Some parameters in the network are shared

Dropout is a kind of ensemble.

Testing of Dropout



Testing of Dropout



Concluding Remarks

New Activation Function

- ReLU and Maxout network

New Structure

- Residue network and Highway network

Better optimization Strategy

- E.g. Adam

Dropout

- Prevent Overfitting

Part III:
Why Deep?

Deeper is Better?

Layer X Size	Word Error Rate (%)
1 X 2k	24.2
2 X 2k	20.4
3 X 2k	18.4
4 X 2k	17.8
5 X 2k	17.2
7 X 2k	17.1

Not surprised, more parameters, better performance

Seide, Frank, Gang Li, and Dong Yu. "Conversational Speech Transcription Using Context-Dependent Deep Neural Networks." *Interspeech*. 2011.

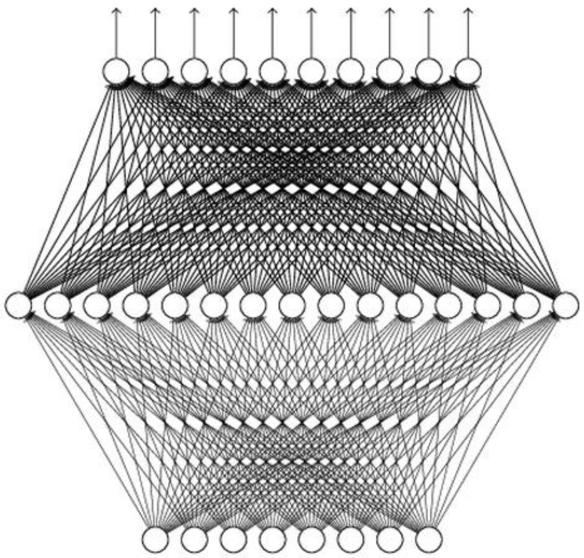
Universality Theorem

Any continuous function f

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

Can be realized by a network with one hidden layer

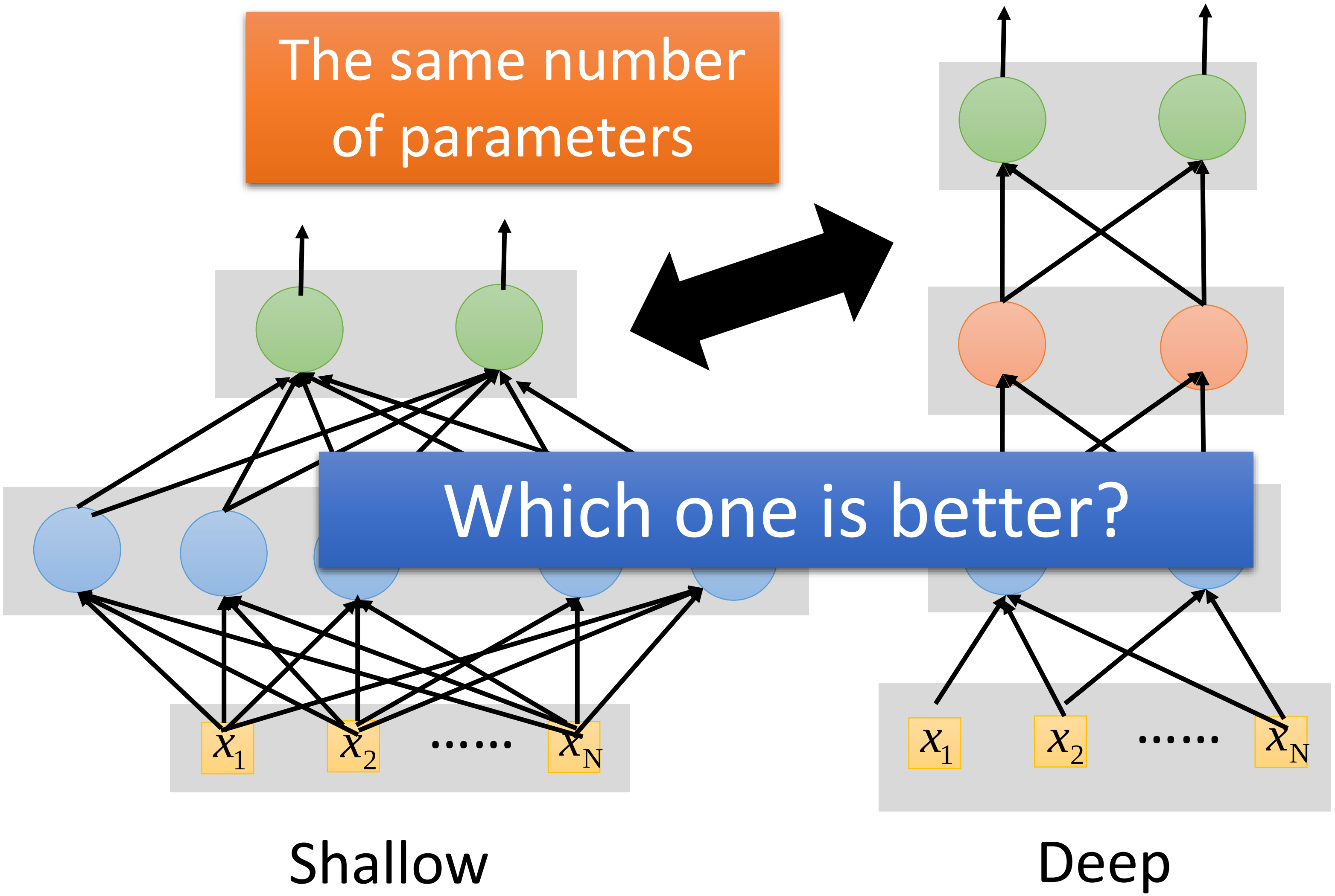
(given **enough** hidden neurons)



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

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

Fat + Short v.s. Thin + Tall



Fat + Short v.s. Thin + Tall

Layer X Size	Word Error Rate (%)	Layer X Size	Word Error Rate (%)
1 X 2k	24.2		
2 X 2k	20.4		
3 X 2k	18.4		
4 X 2k	17.8		
5 X 2k	17.2	1 X 3772	22.5
7 X 2k	17.1	1 X 4634	22.6
		1 X 16k	22.1

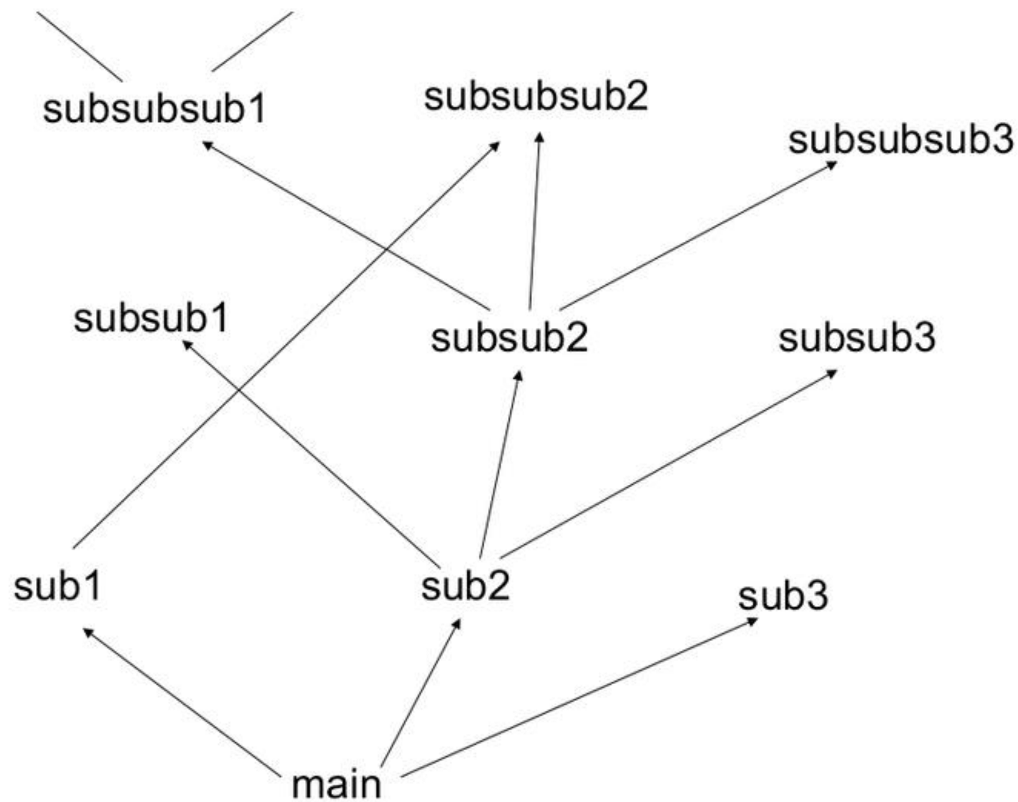
Why?

Seide, Frank, Gang Li, and Dong Yu. "Conversational Speech Transcription Using Context-Dependent Deep Neural Networks." *Interspeech*. 2011.

Modularization

- Deep → Modularization

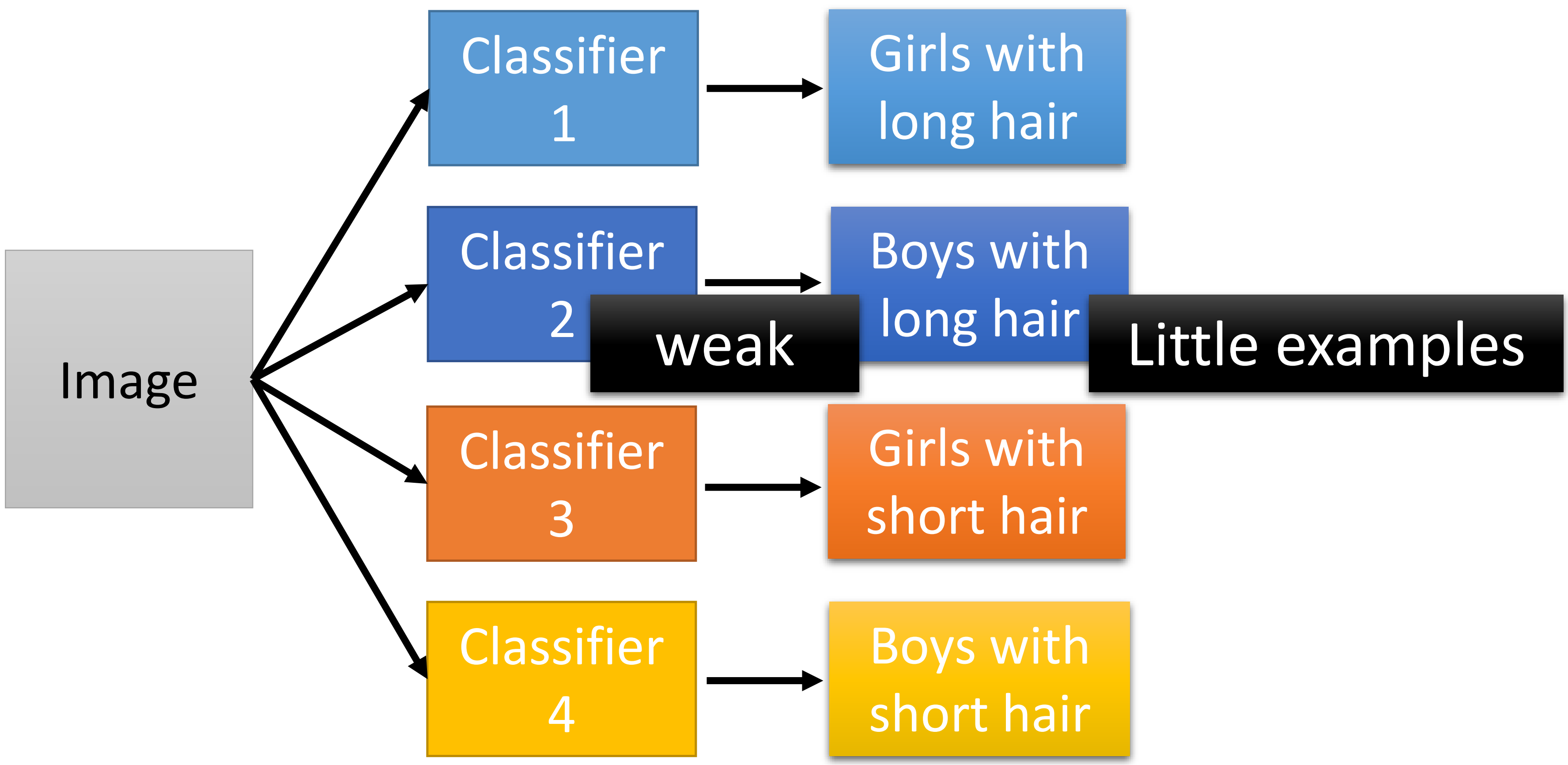
Don't put everything in your main function.



http://rinuboney.github.io/2015/10/18/theoretical-motivations-deep-learning.html

Modularization

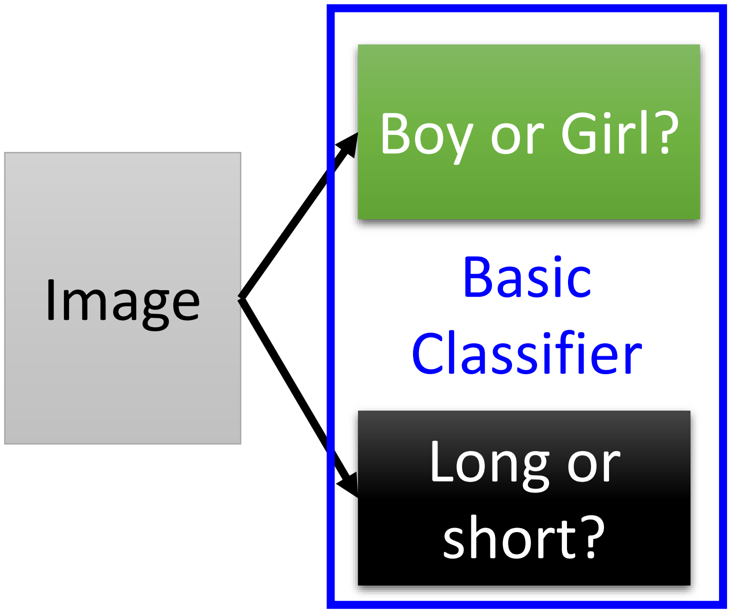
- Deep → Modularization



Modularization

Each basic classifier can have sufficient training examples.

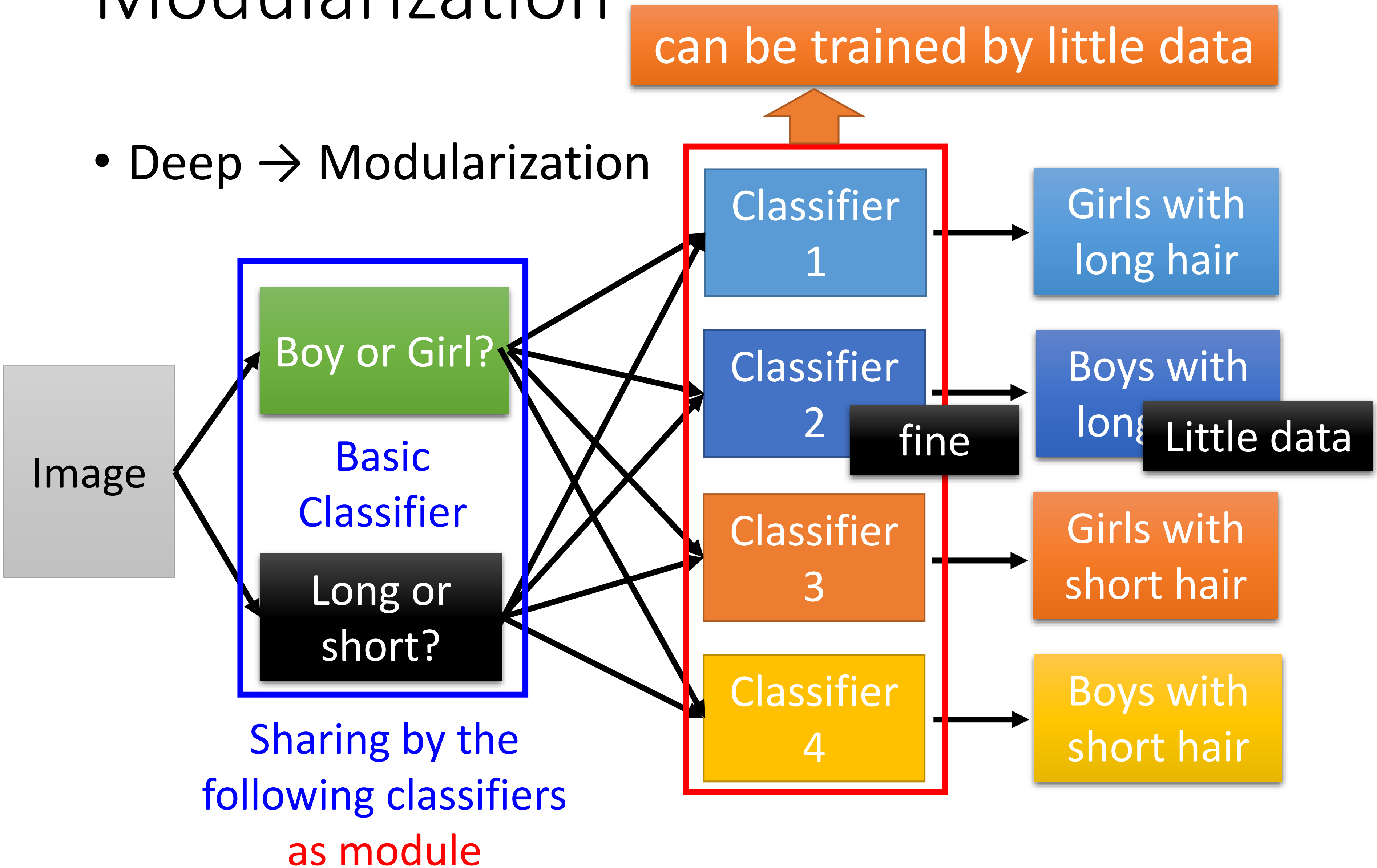
- Deep → Modularization



Classifiers for the attributes

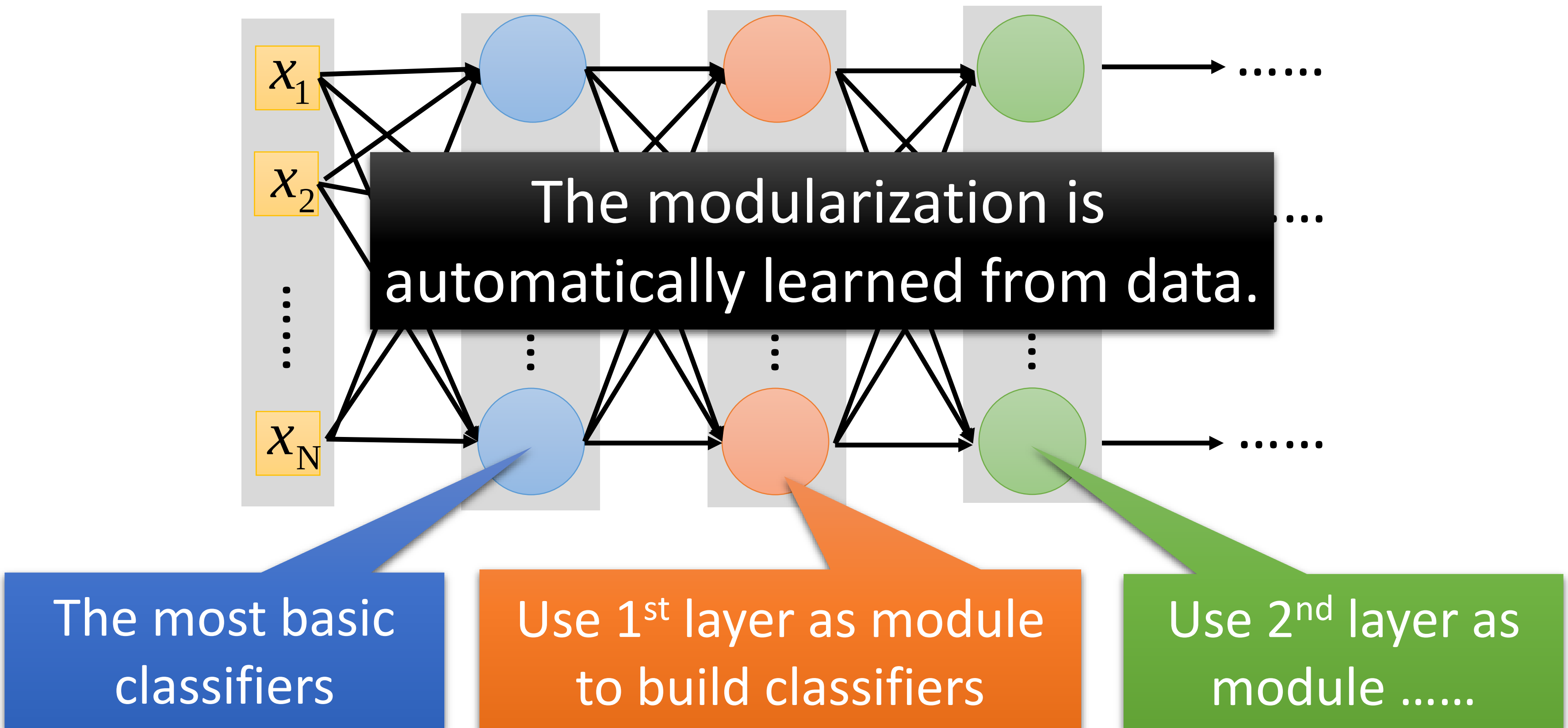
Modularization

- Deep → Modularization



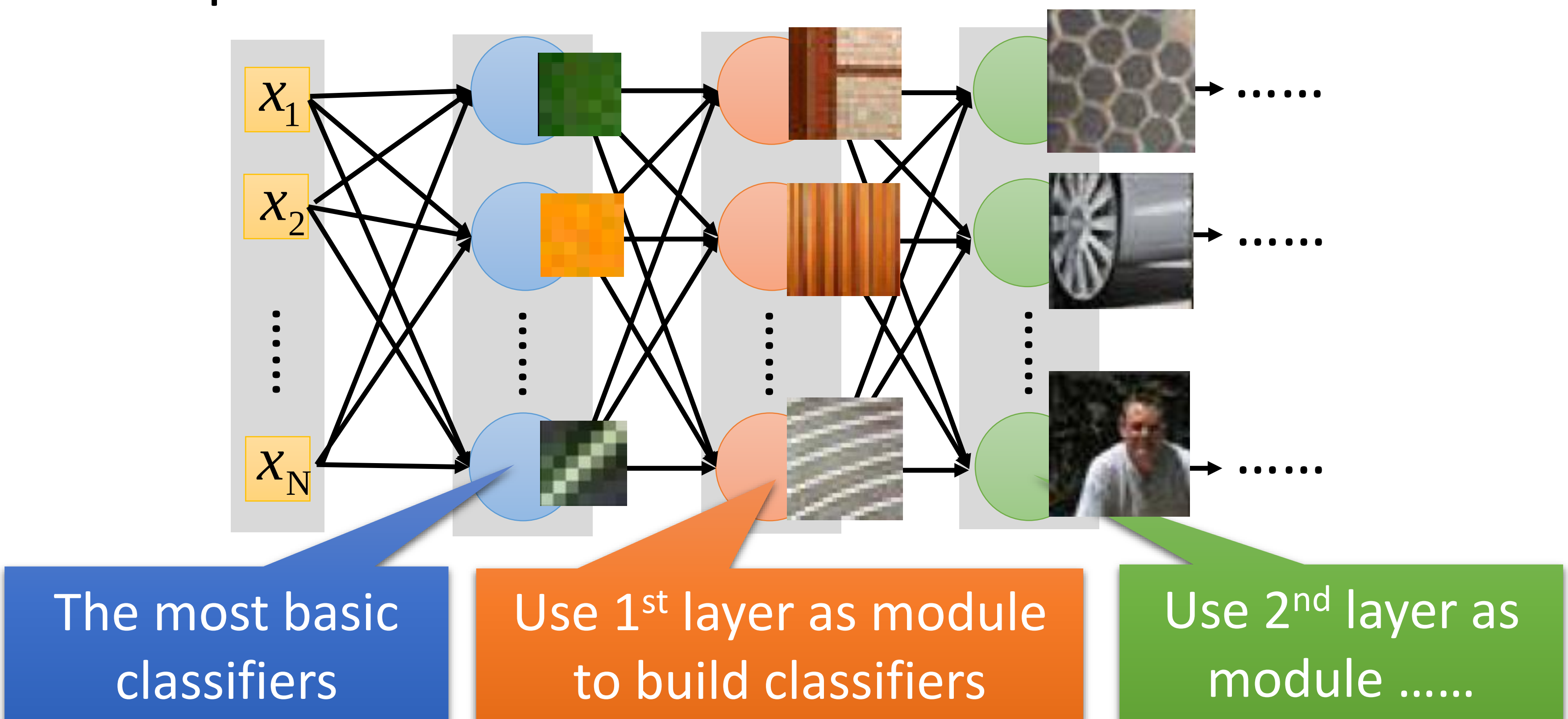
Modularization

- Deep → Modularization → Less training data?



Modularization - Image

- Deep → Modularization



Reference: Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014* (pp. 818-833)

Modularization - Speech

- The hierarchical structure of human languages

what do you think

Phoneme:

hh w aa t d uw y uw th ih ng k

Tri-phone:

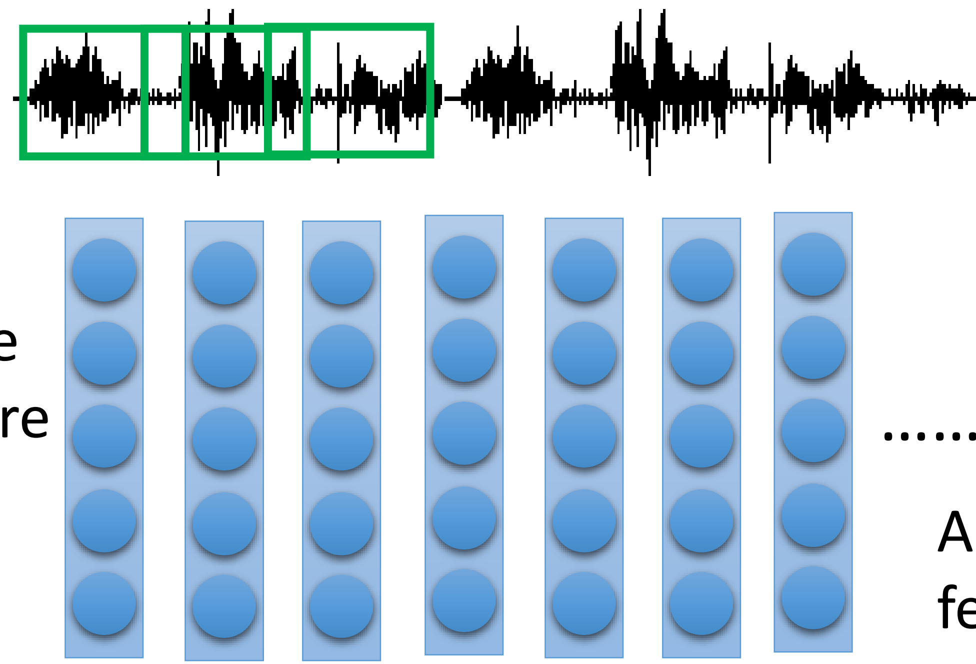
…… t-d+uw d-uw+y uw-y+uw y-uw+th ……

t-d+uw1 t-d+uw2 t-d+uw3 d-uw+y1 d-uw+y2 d-uw+y3

State:

Modularization - Speech

- The first stage of speech recognition
  - Classification: input → acoustic feature, output → state



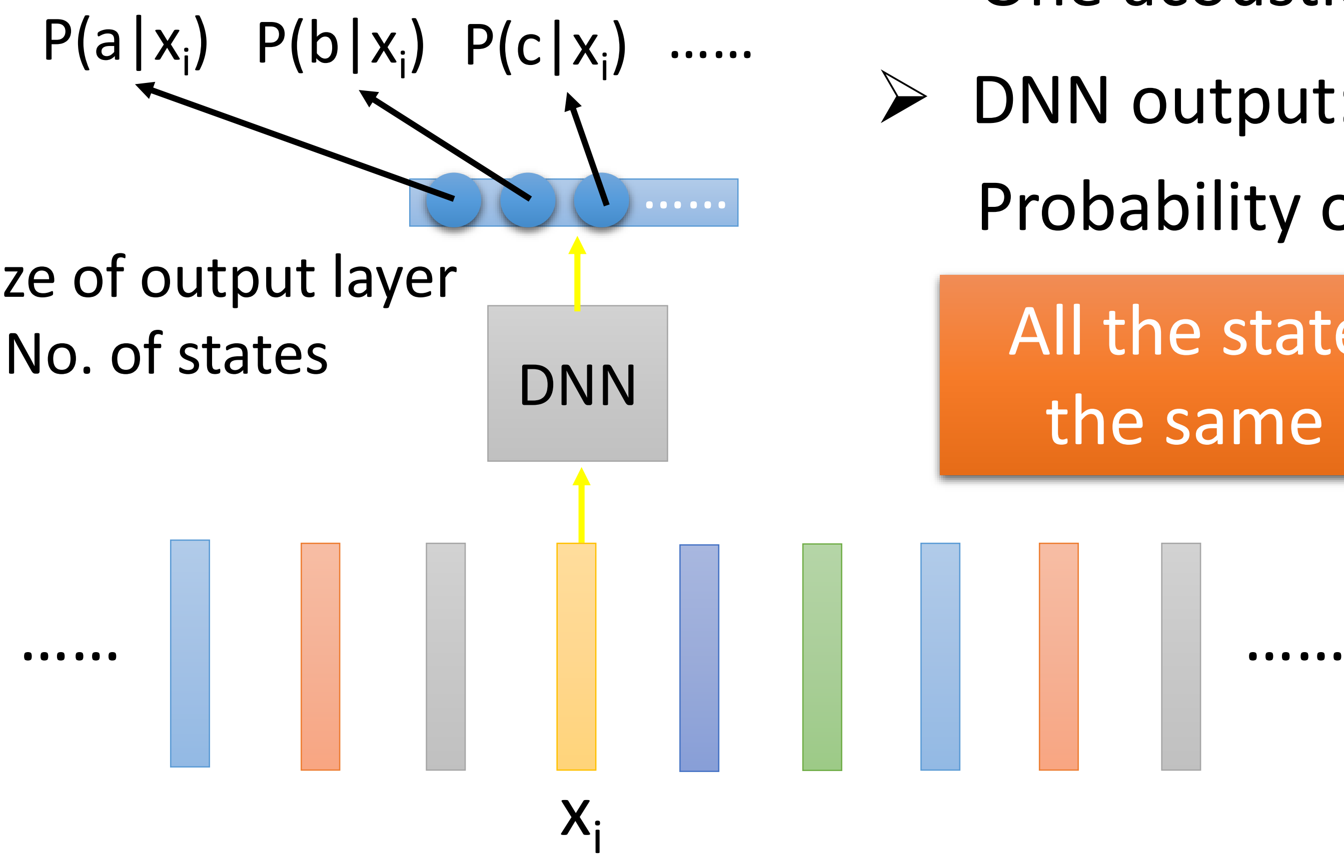
Determine the state each acoustic feature belongs to

......

Acoustic feature

States: a a a b b c c

Modularization - Speech



Size of output layer
= No. of states

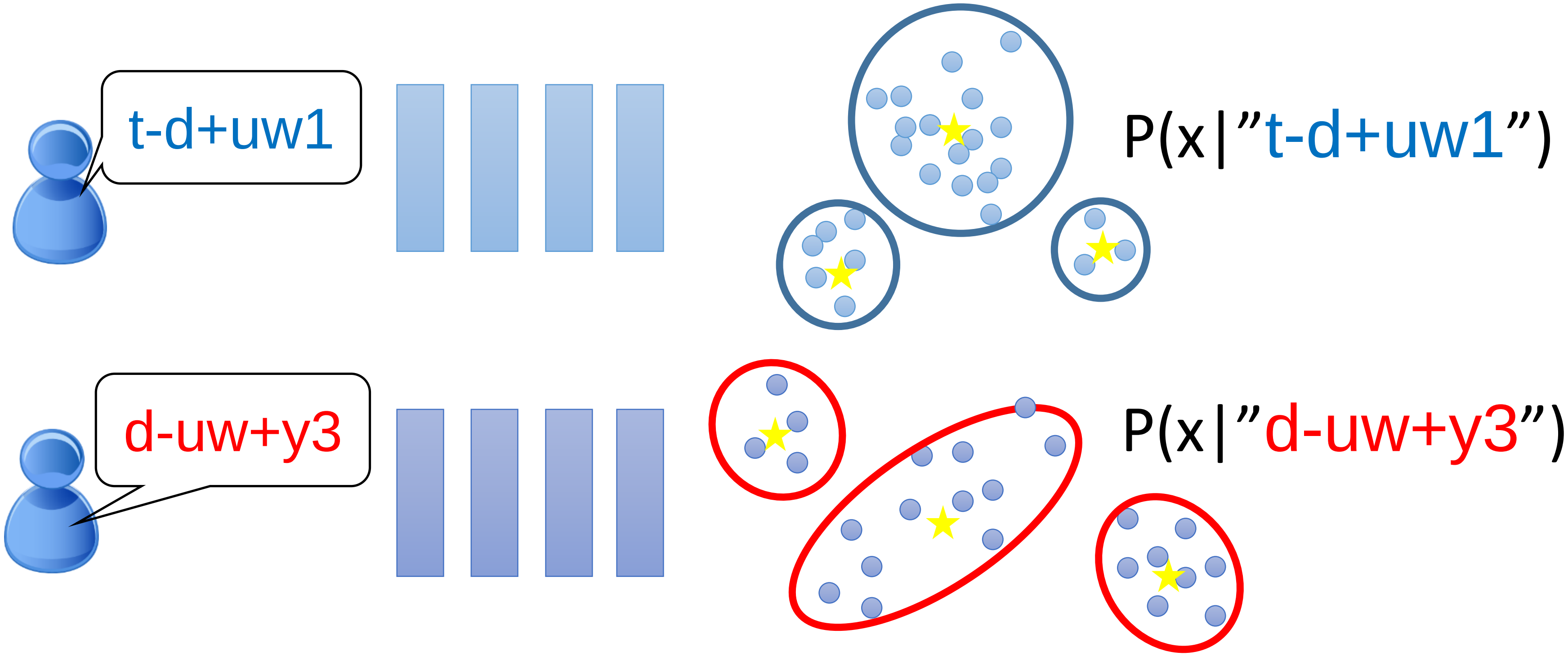
- DNN input:
  One acoustic feature
- DNN output:
  Probability of each state

All the states use the same DNN

Modularization - Speech

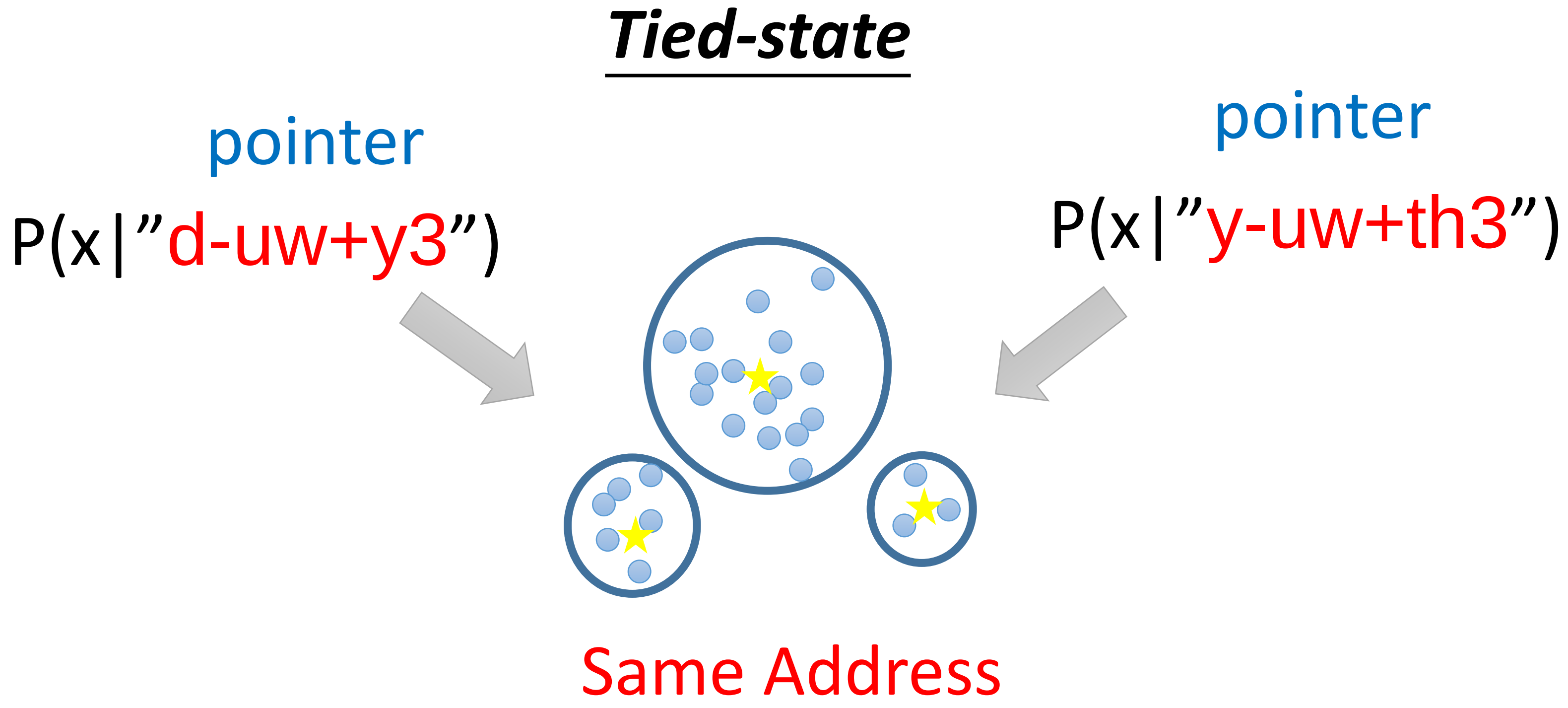
- Each state has a stationary distribution for acoustic features

Gaussian Mixture Model (GMM)



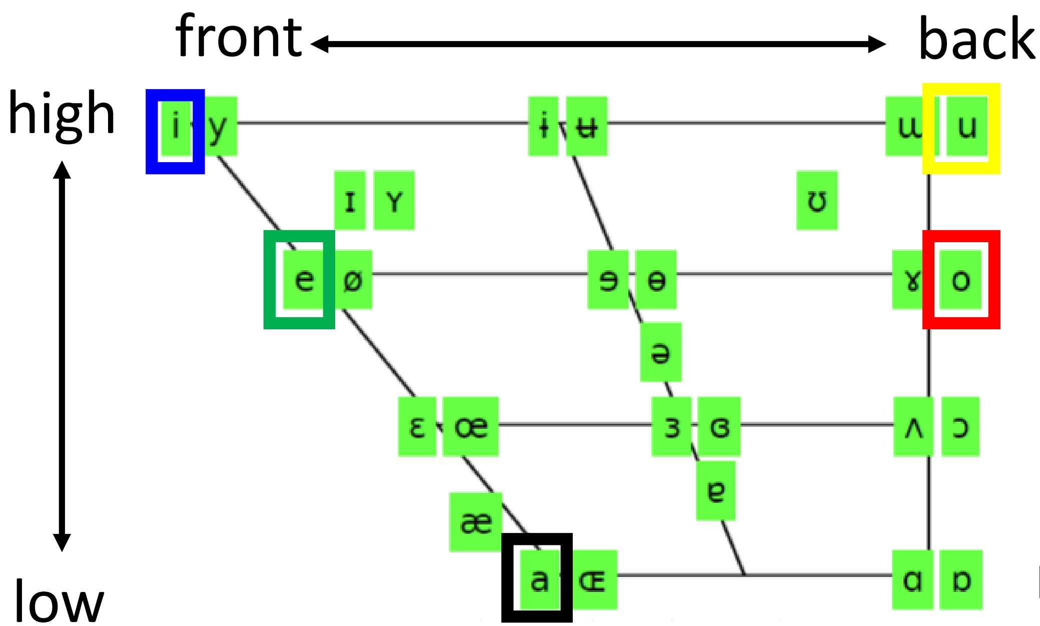
Modularization - Speech

- Each state has a stationary distribution for acoustic features



Modularization - Speech

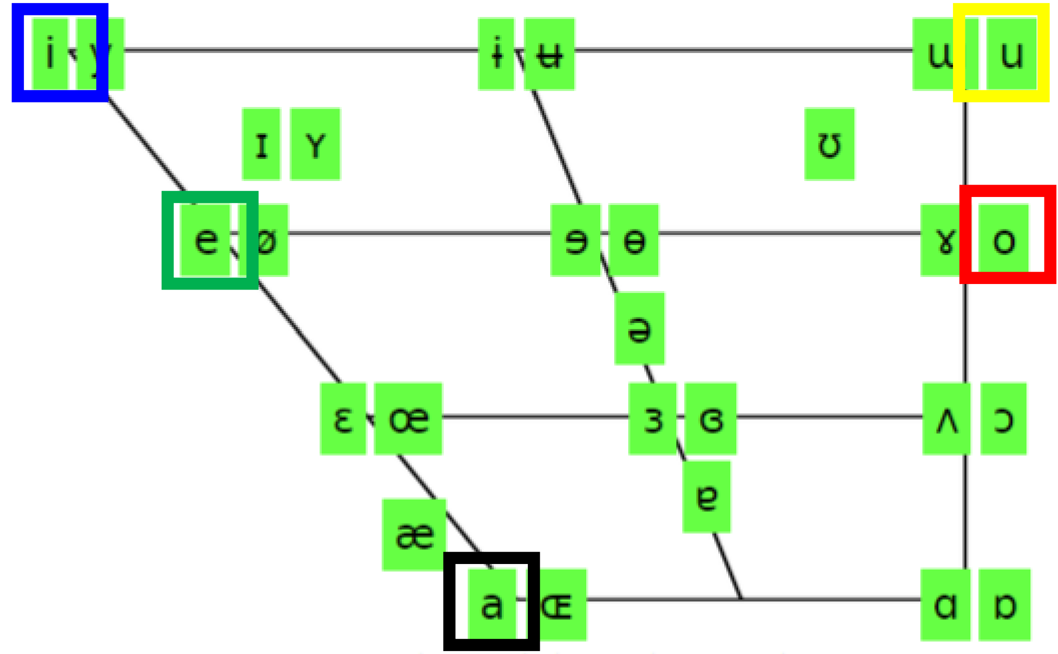
- By GNN, all the phonemes are modeled independently
  - Not an effective way to model human voice



The sound of vowel is only controlled by a few factors.

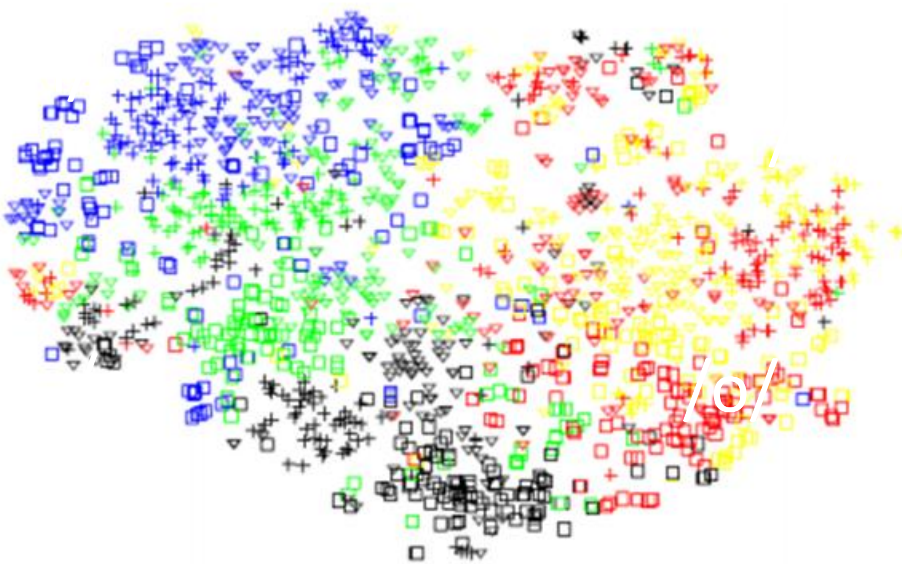
http://www.ipachart.com/

Modularization



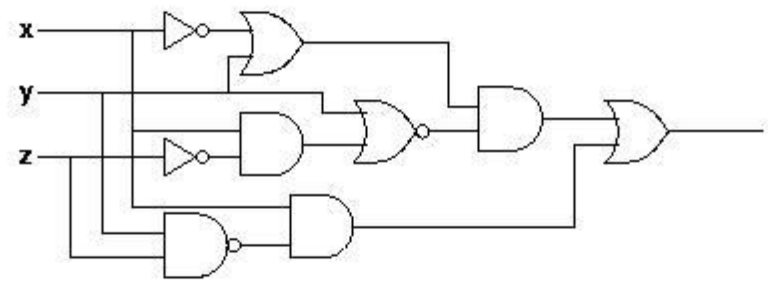
Vu, Ngoc Thang, Jochen Weiner, and Tanja Schultz. "Investigating the Learning Effect of Multilingual Bottle-Neck Features for ASR." *Interspeech*. 2014.

Output of hidden layer reduce to two dimensions



- The lower layers detect the manner of articulation
- All the phonemes share the results from the same set of detectors.
- Use parameters effectively

Analogy



Logic circuits	Neural network
Logic circuits consists of **gates**	Neural network consists of **neurons**
A two layers of logic gates can represent **any Boolean function.**	**A hidden layer network** can represent **any continuous function.**
Using multiple layers of logic gates to build some functions are much simpler	Using multiple layers of neurons to represent some functions are much simpler
→ less gates needed	→ less parameters → less data?

This page is for EE background.

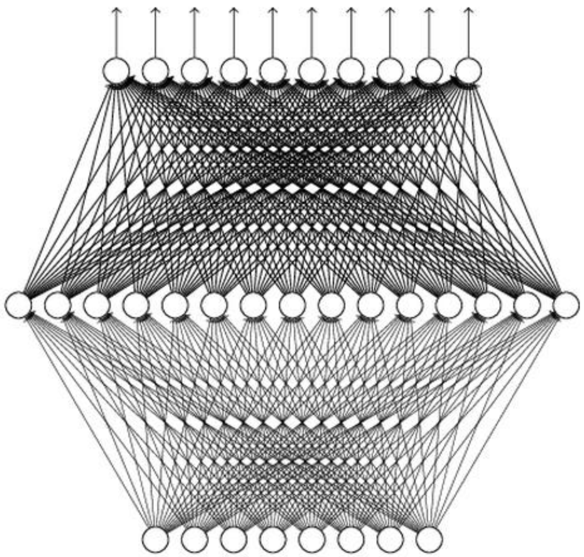
Universality Theorem

Any continuous function f

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

Can be realized by a network with one hidden layer

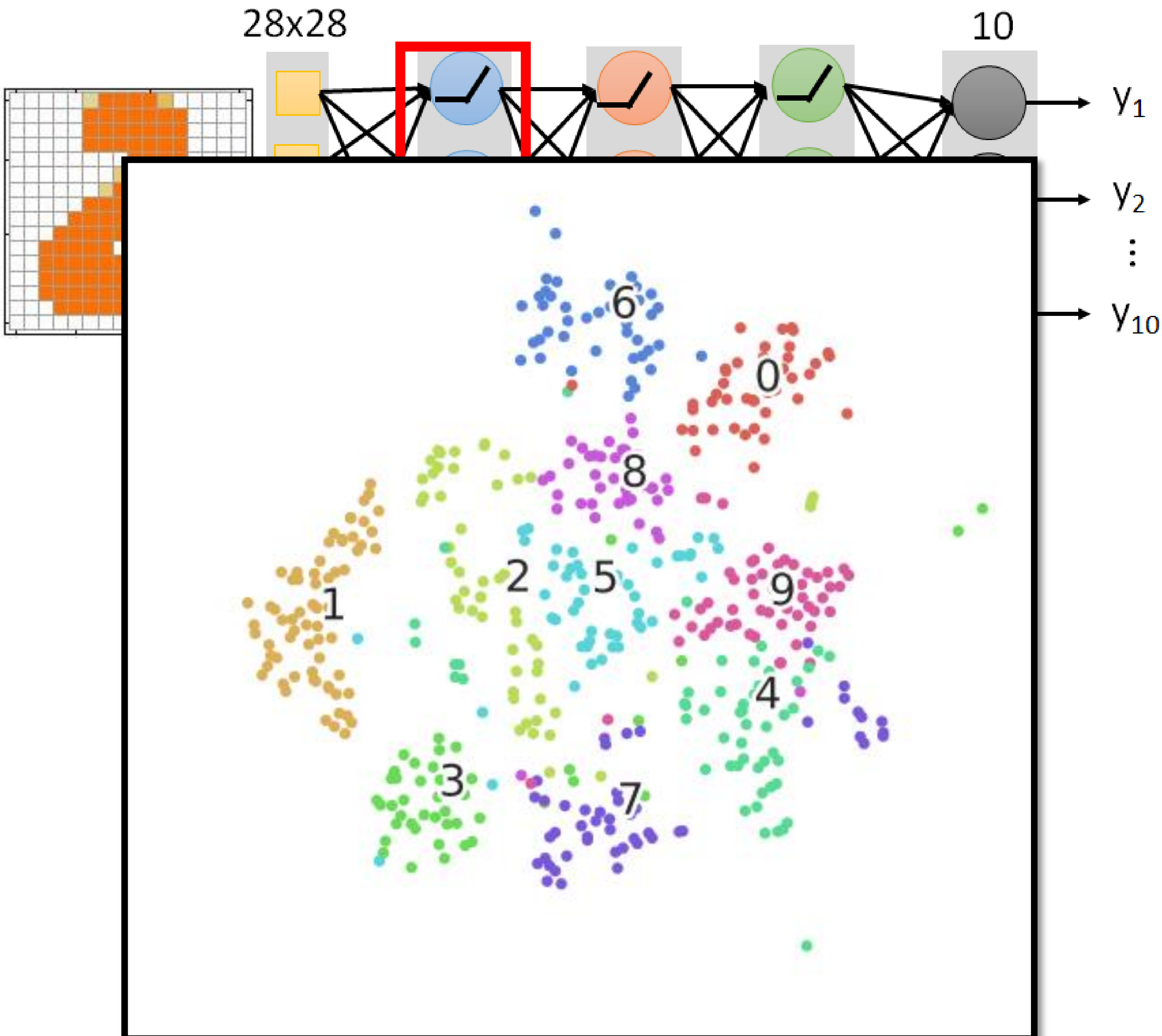
(given **enough** hidden neurons)



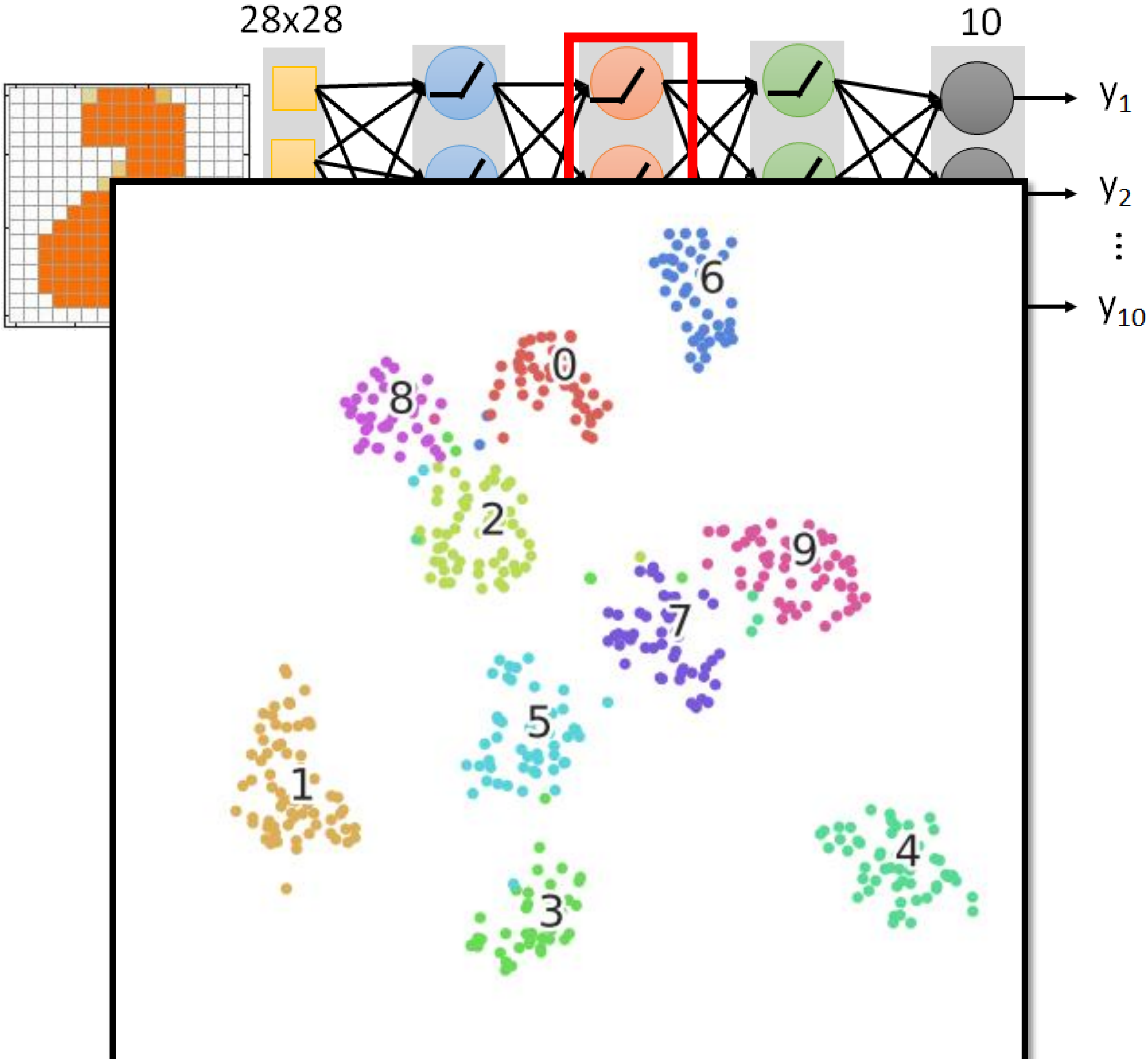
Reference for the reason: http://neuralnetworksanddeeplearning.com/chap4.html

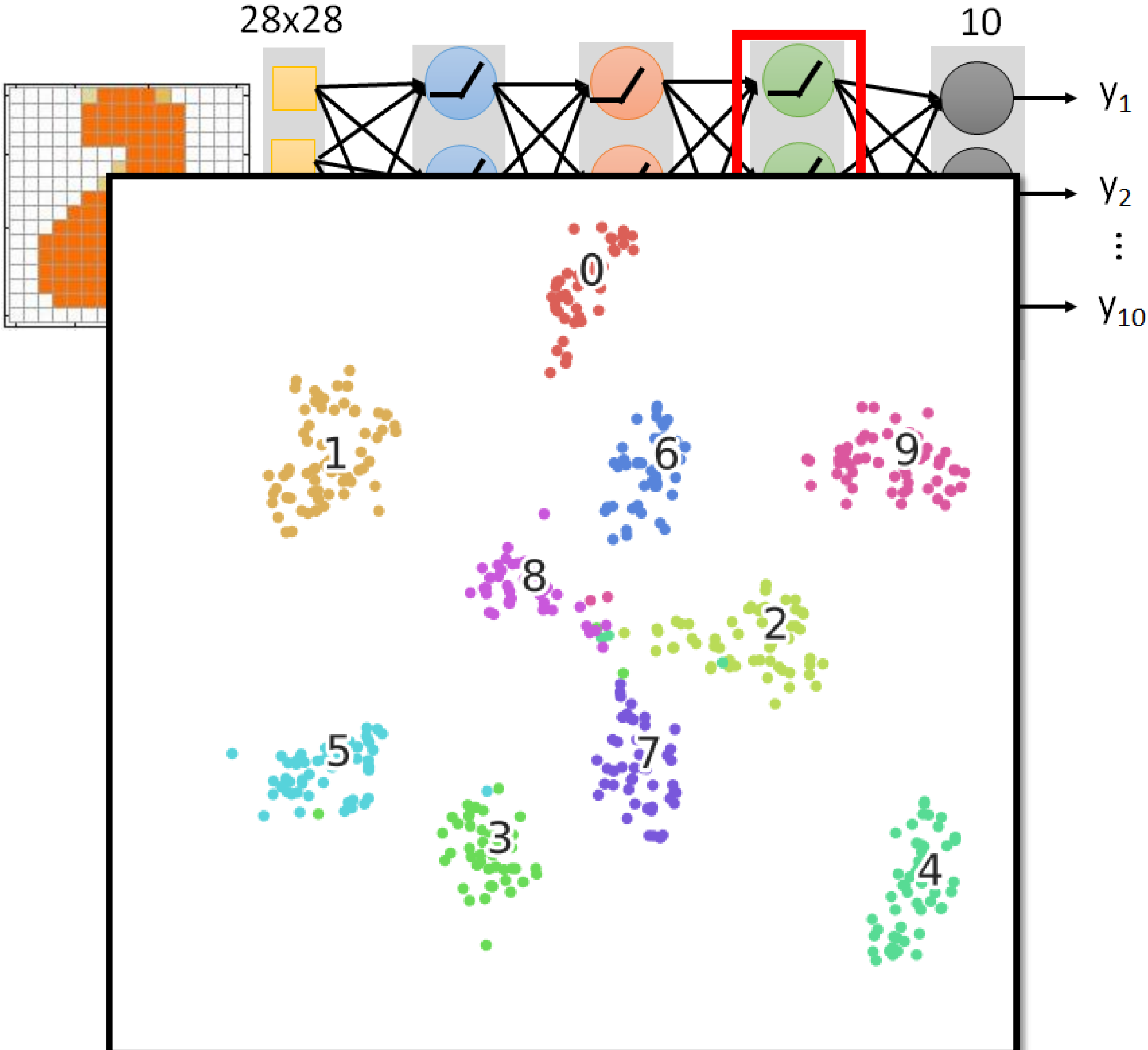
Yes, shallow network can represent any function.

However, using deep structure is more effective.



28x28
10
y_1
y_2
y_{10}
6
0
8
1
2
5
9
4
3
7


28x28
10
y_1
y_2
y_{10}
6
0
8
2
9
7
5
1
4
3

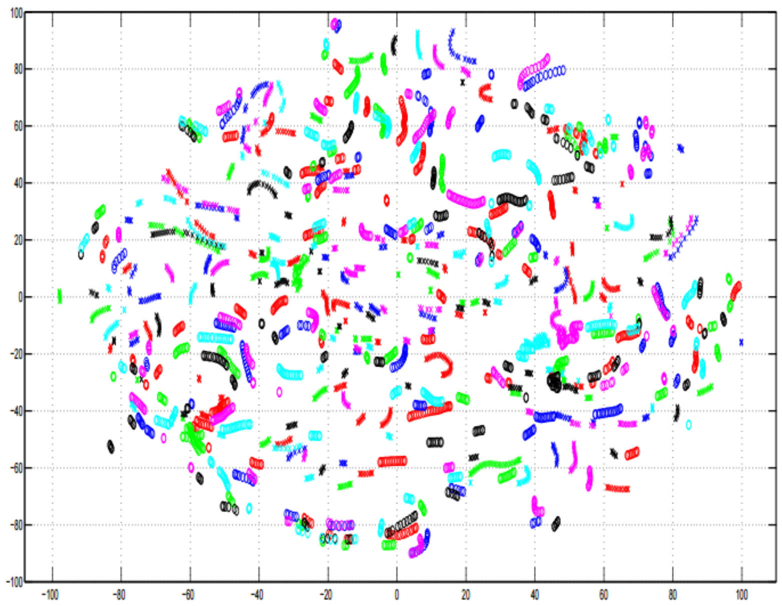


28x28
10
y1
y2
⋮
y10
0
1
6
9
8
2
5
7
3
4

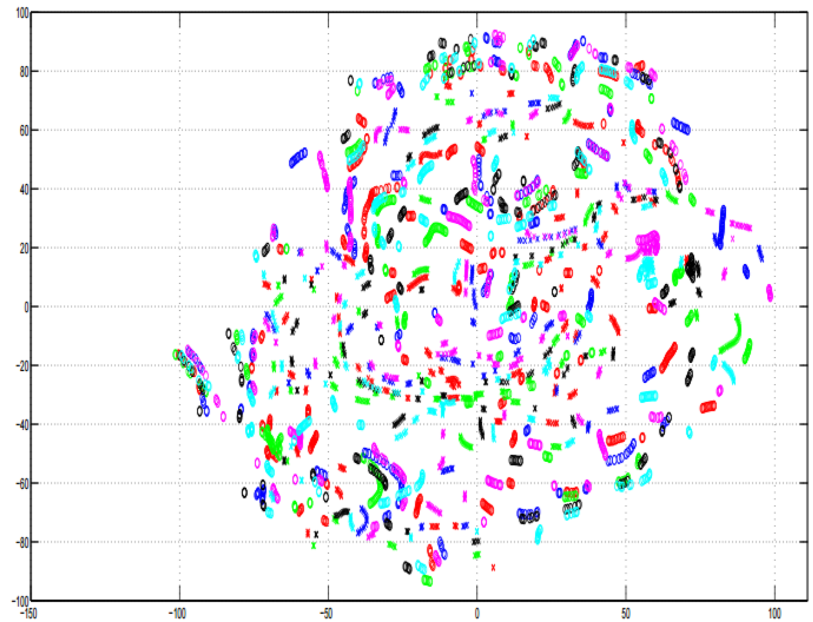
Complex Task …

A. Mohamed, G. Hinton, and G. Penn, “Understanding how Deep Belief Networks Perform Acoustic Modelling,” in ICASSP, 2012.

- Speech recognition: Speaker normalization is automatically done in DNN



Input Acoustic Feature (MFCC)

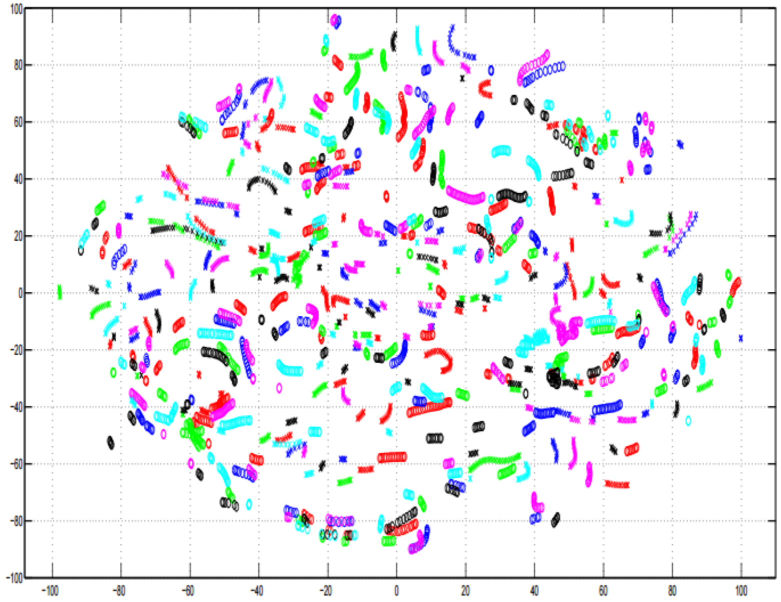


1-st Hidden Layer

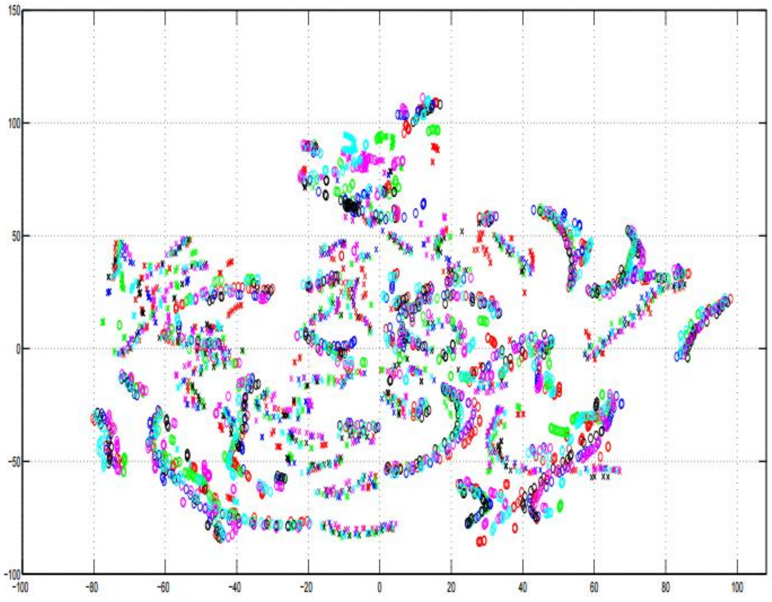
Complex Task …

A. Mohamed, G. Hinton, and G. Penn, “Understanding how Deep Belief Networks Perform Acoustic Modelling,” in ICASSP, 2012.

- Speech recognition: Speaker normalization is automatically done in DNN

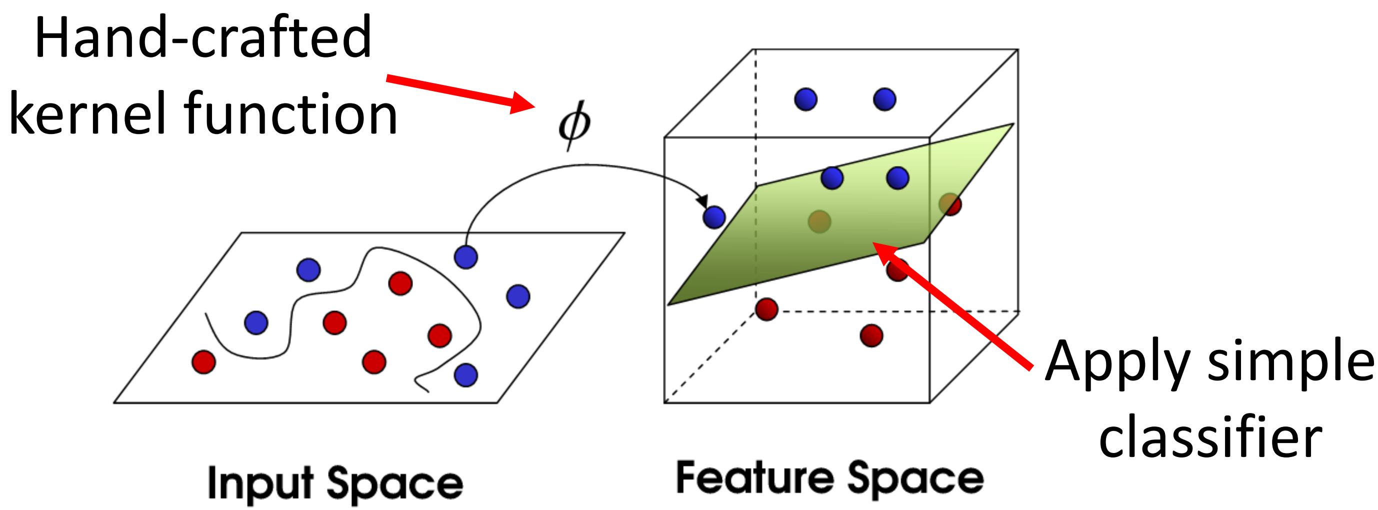


Input Acoustic Feature (MFCC)



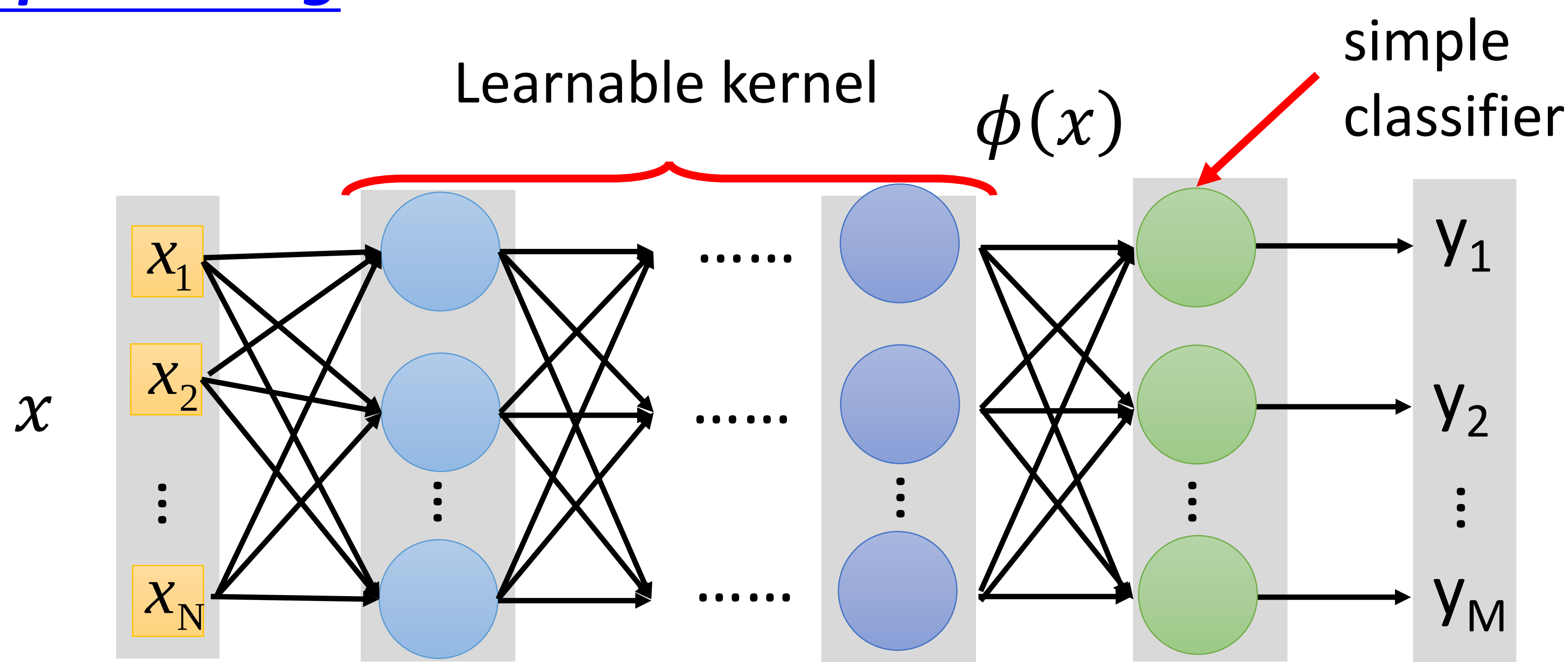
8-th Hidden Layer

SVM

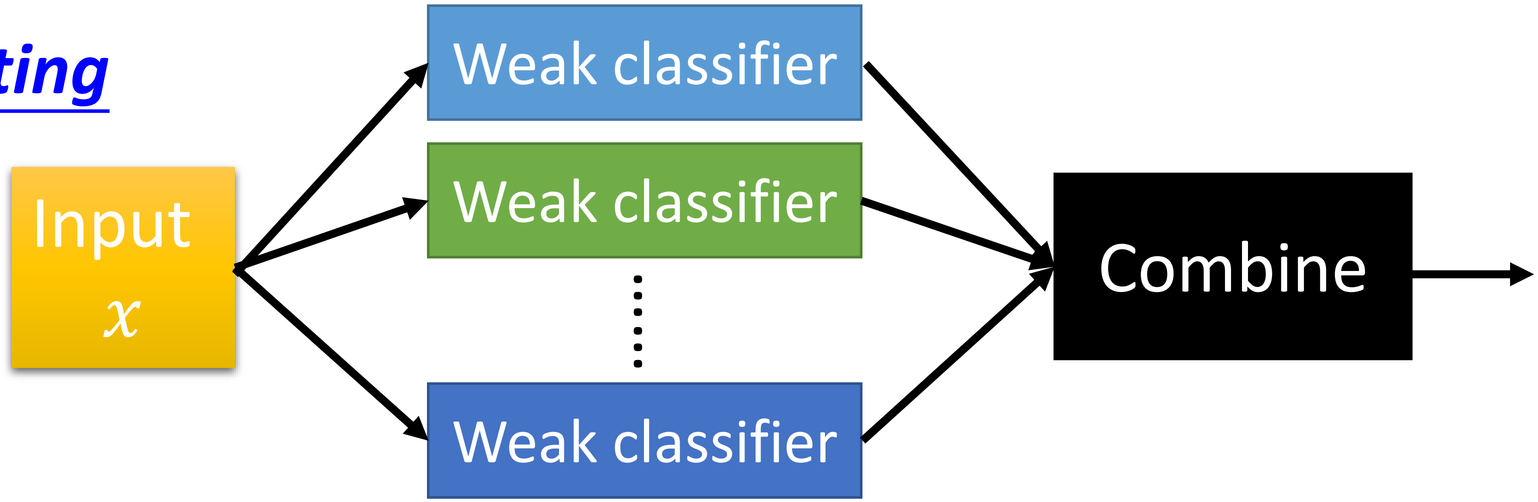


Source of image: http://www.gipsa-lab.grenoble-inp.fr/transfert/seminaire/455_Kadri2013Gipsa-lab.pdf

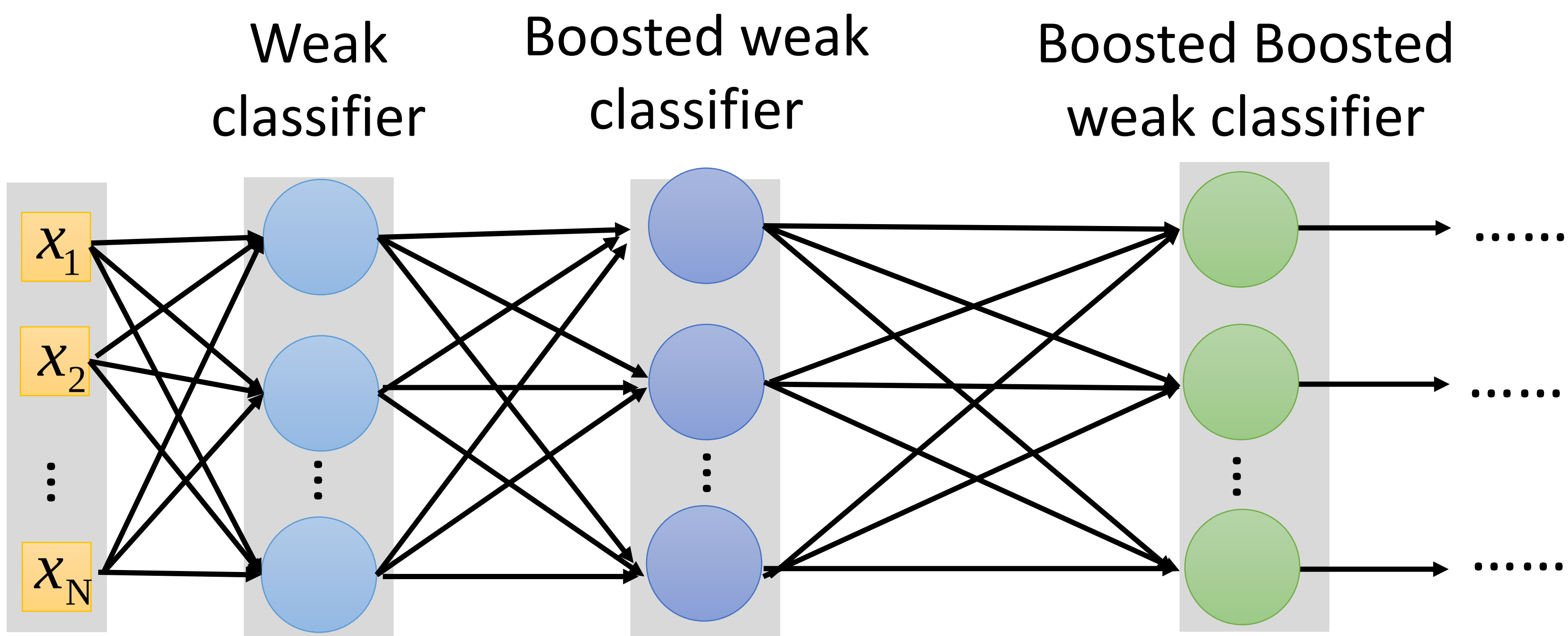
Deep Learning



Boosting

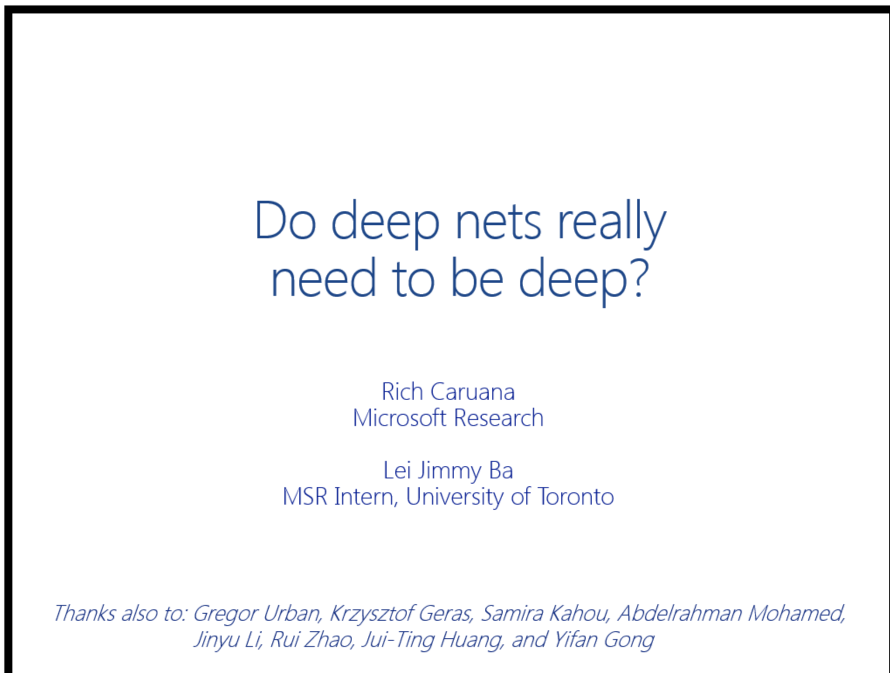


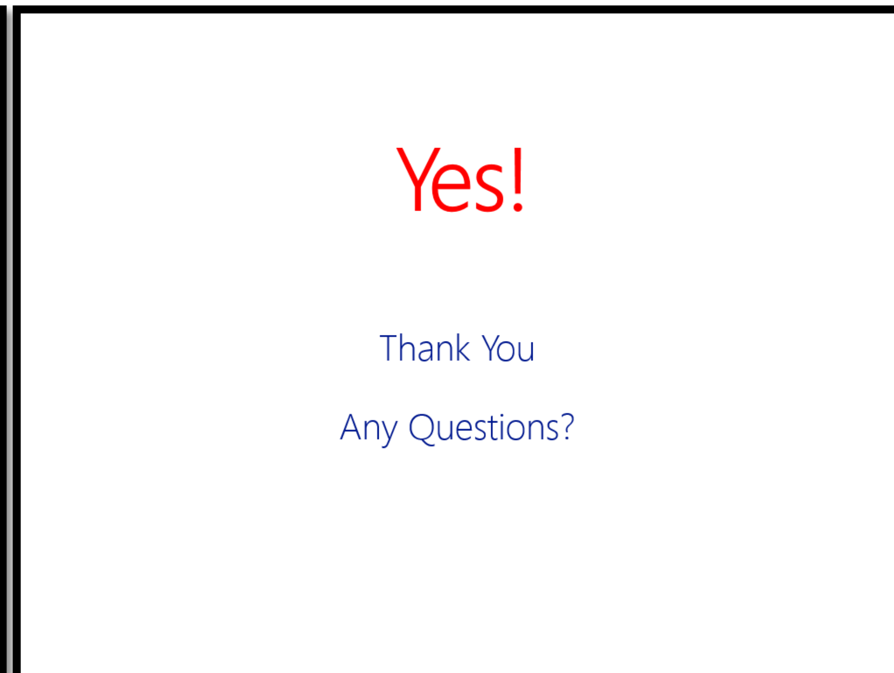
Deep Learning



To learn more …

- Do Deep Nets Really Need To Be Deep? (by Rich Caruana)
- http://research.microsoft.com/apps/video/default.aspx?id=232373&r=1





keynote of Rich Caruana at ASRU 2015