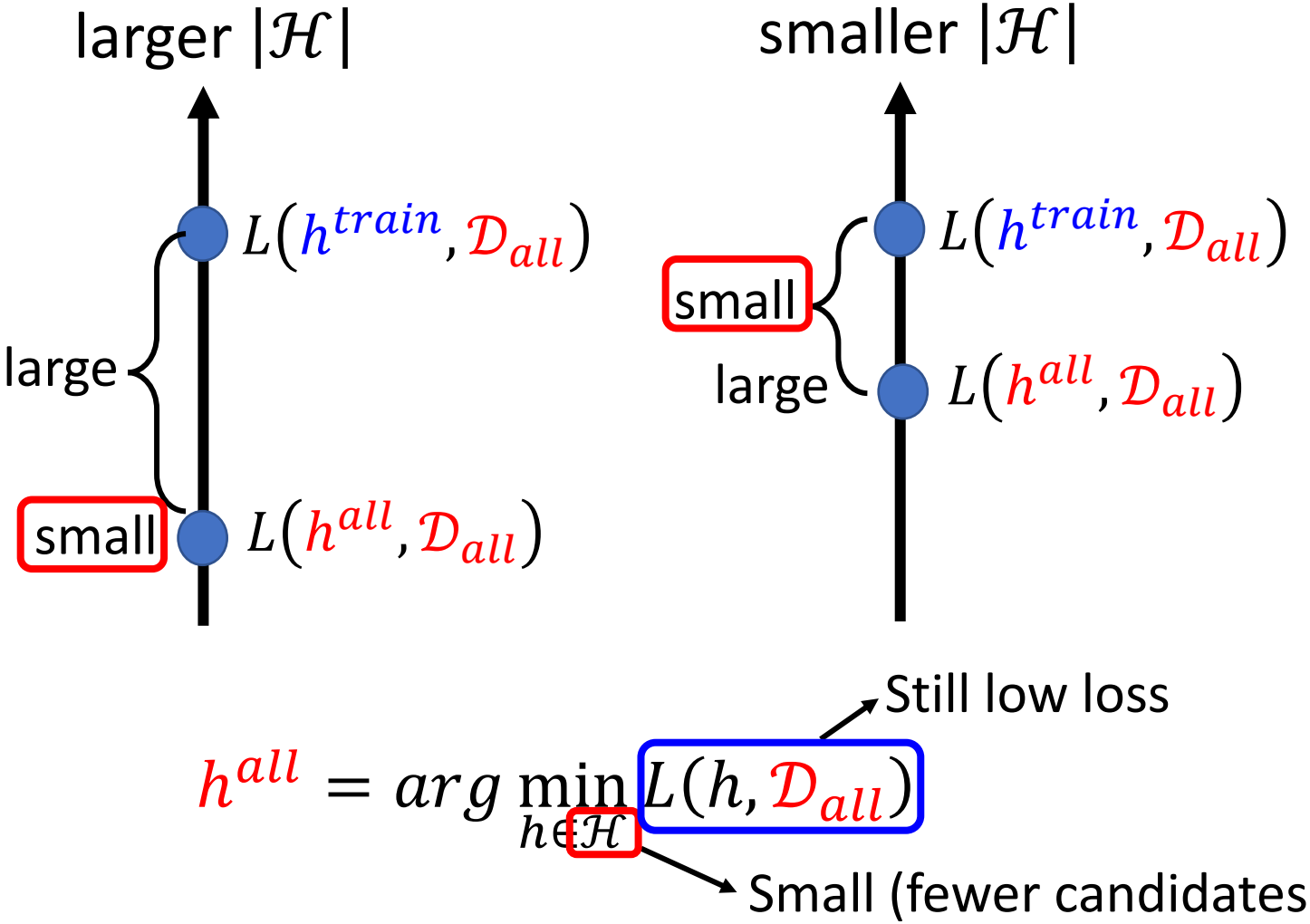
人工智能技术及应用

Artificial Intelligence and Application

Why Deep Learning?

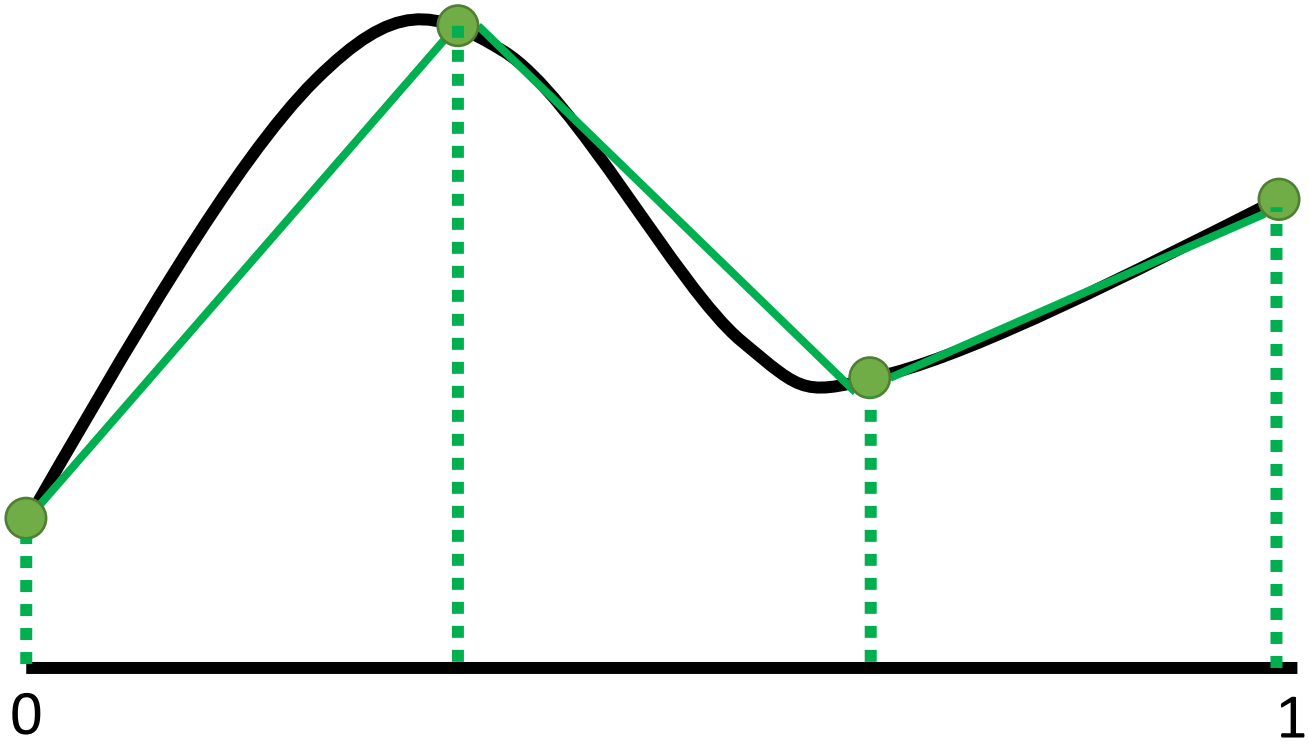
鱼和熊掌能否兼得?

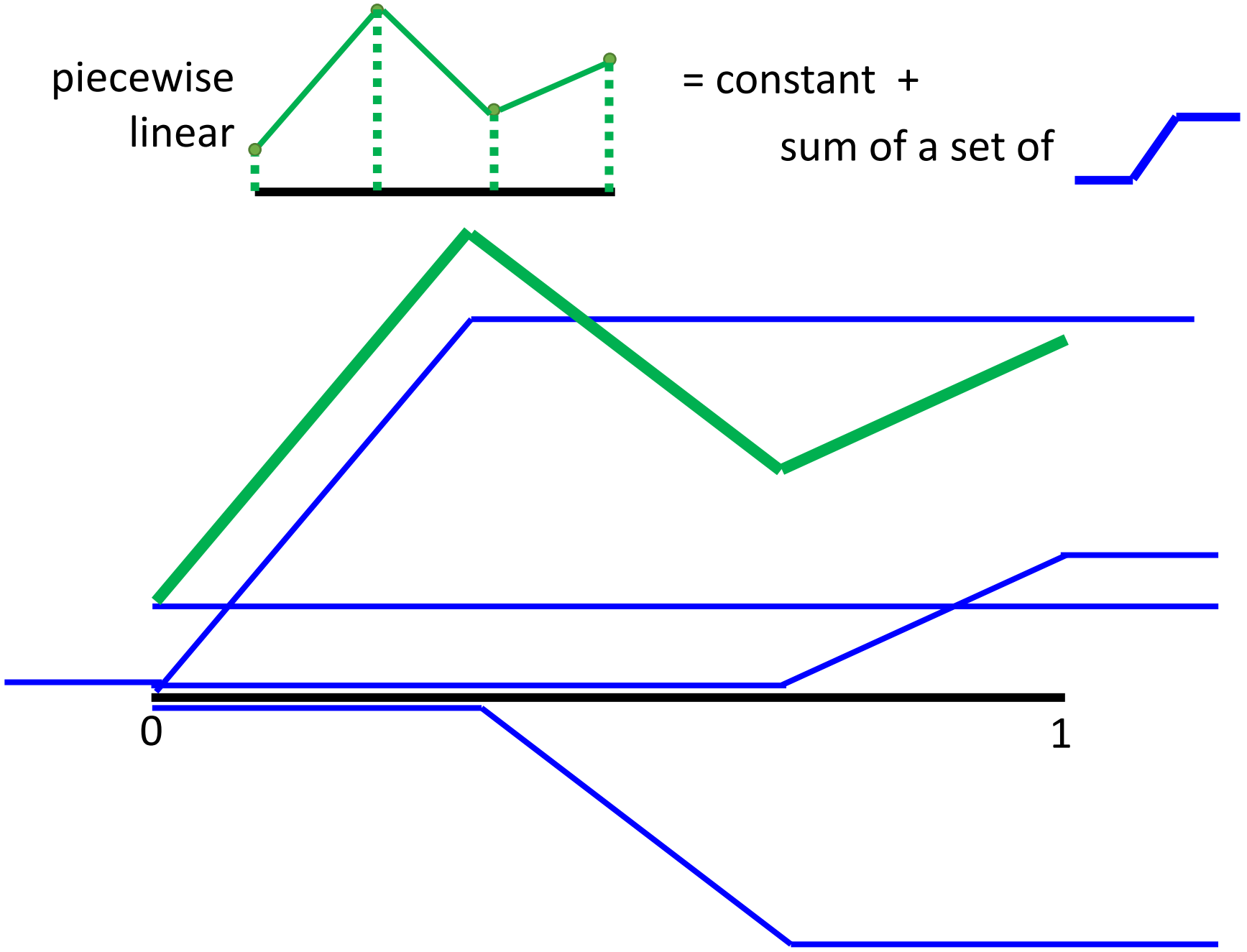

larger $|\mathcal{H}|$
$L(h^{train}, \mathcal{D}_{all})$
large
small
$L(h^{all}, \mathcal{D}_{all})$
smaller $|\mathcal{H}|$
$L(h^{train}, \mathcal{D}_{all})$
small
large
$L(h^{all}, \mathcal{D}_{all})$
Still low loss
$h^{all} = arg \min_{h \in \mathcal{H}} L(h, \mathcal{D}_{all})$
Small (fewer candidates)

Review: Why Hidden Layer?

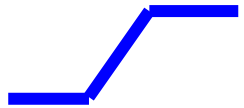
Piecewise Linear

We can have good approximation with sufficient pieces.

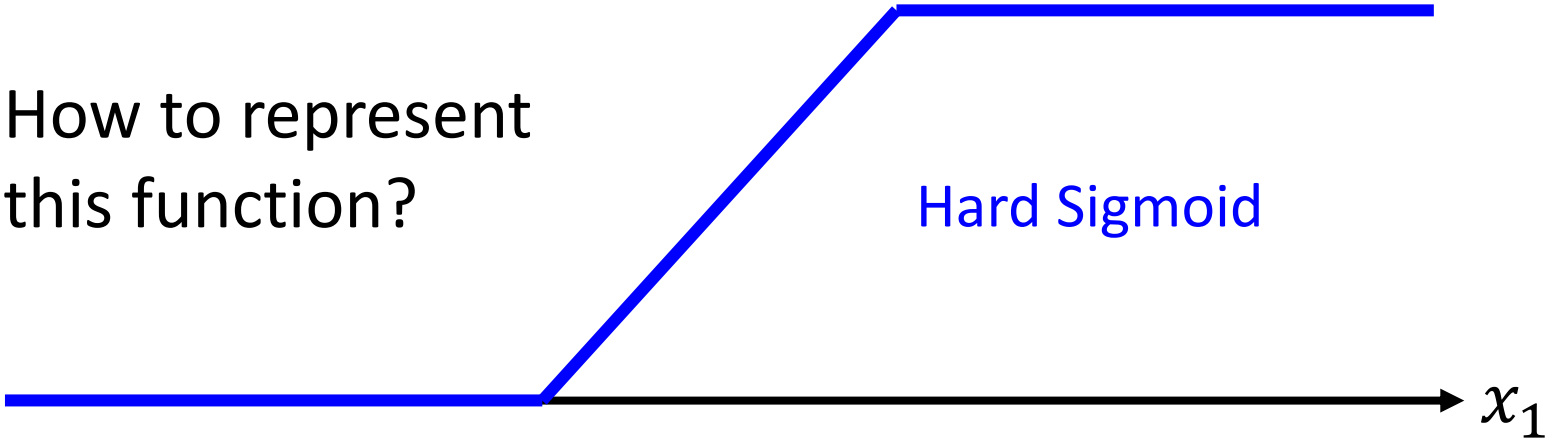



piecewise
linear
= constant +
sum of a set of
0
1

Piecewise linear = constant + sum of a set of



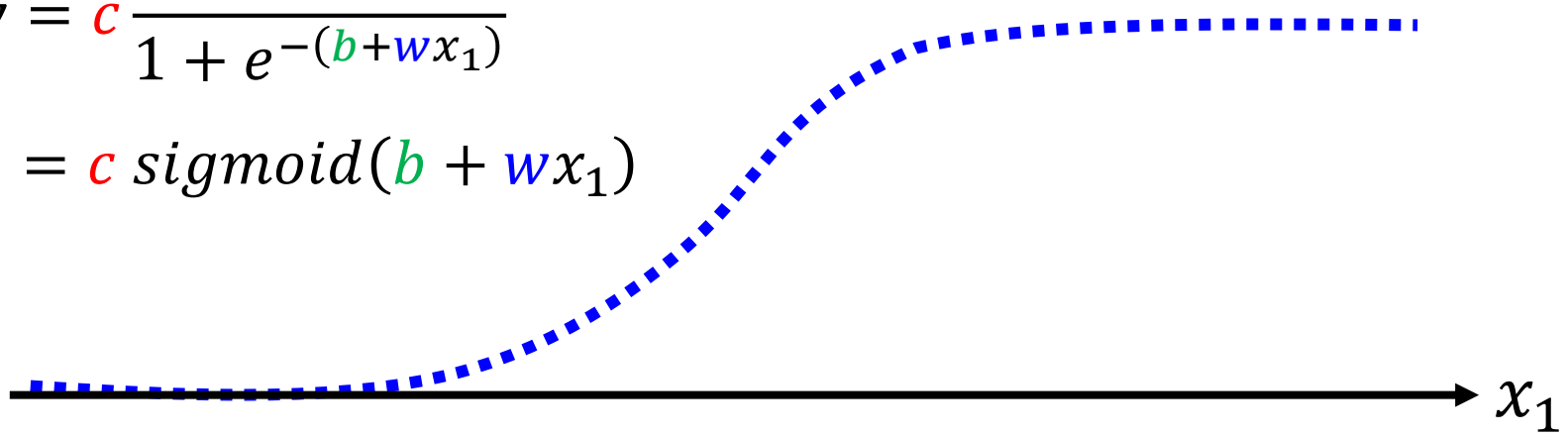
How to represent this function?



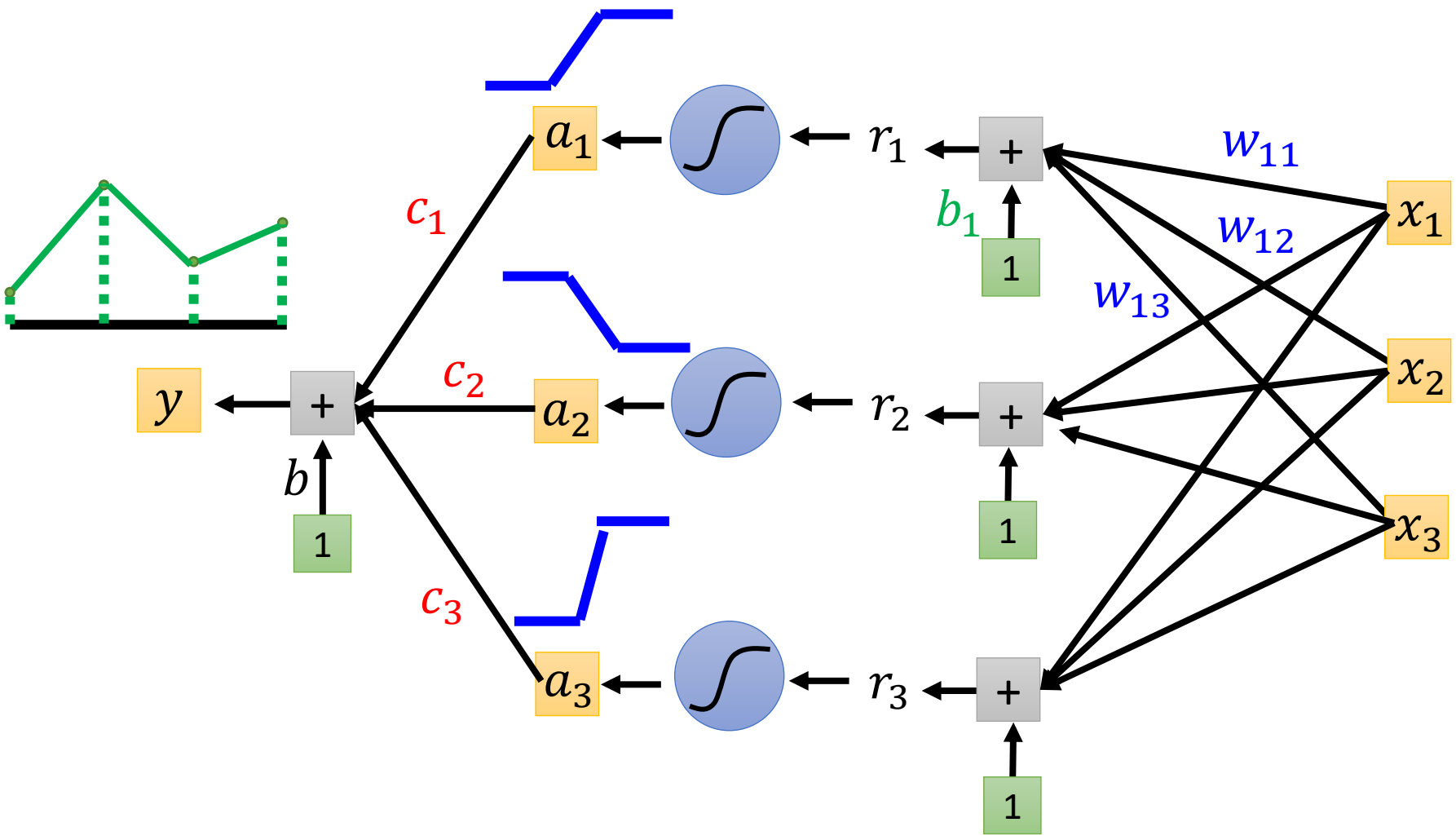
Sigmoid Function

$$y = c\frac{1}{1+e^{-(b+wx_1)}}$$

$$= c\ sigmoid(b+wx_1)$$

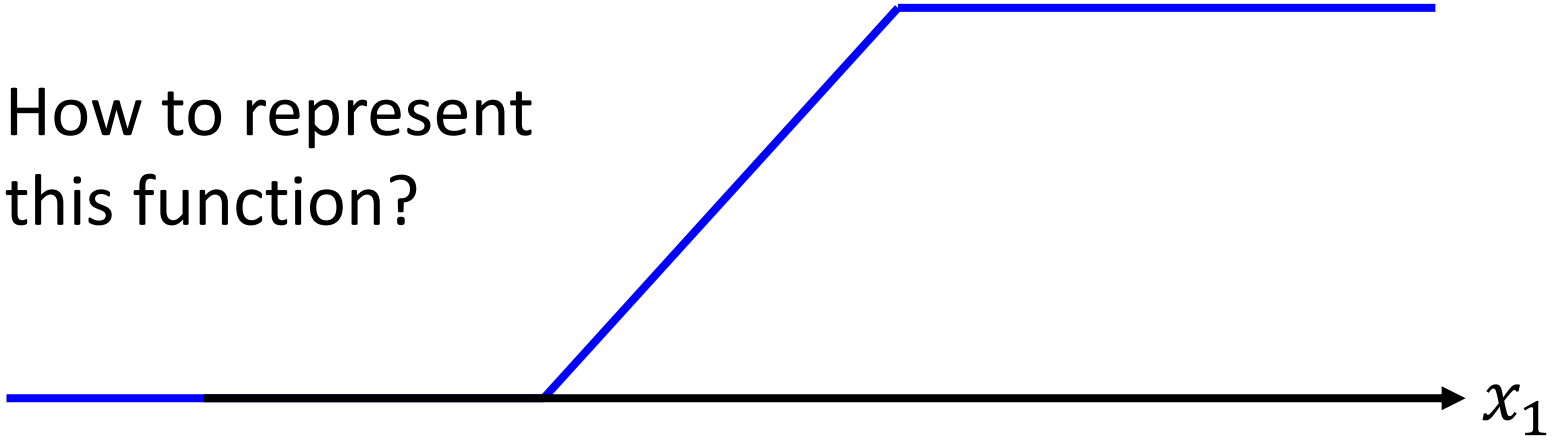


Piecewise linear = constant + sum of a set of

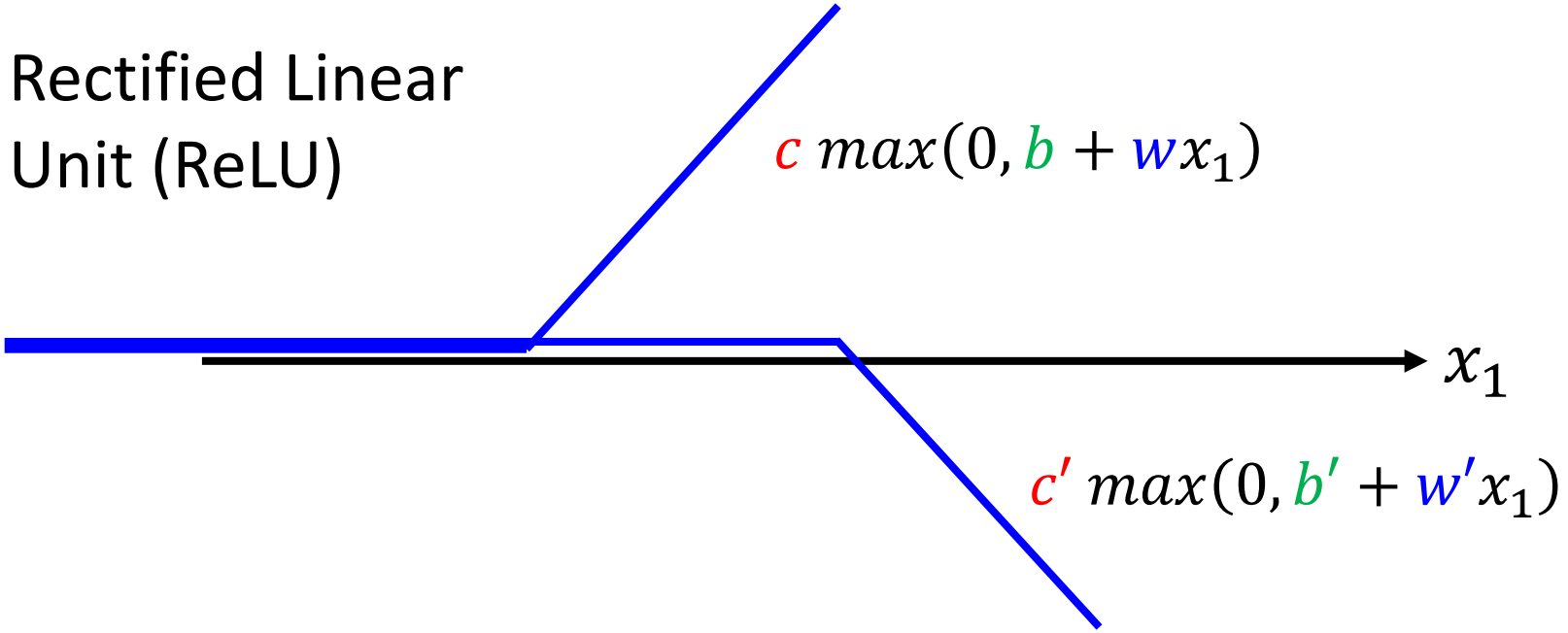


Hard Sigmoid → ReLU

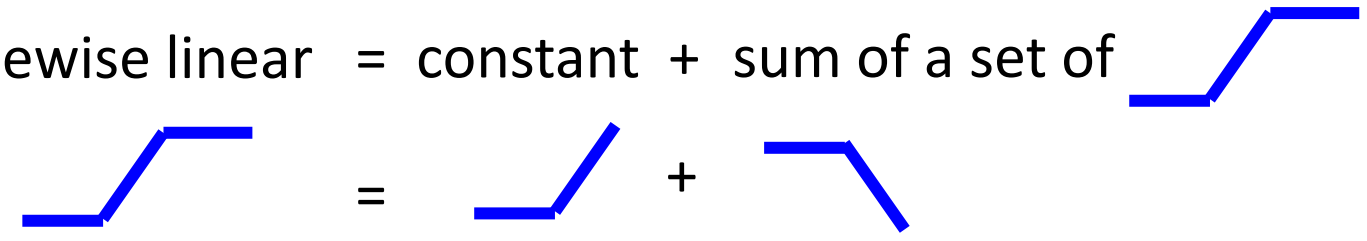
How to represent this function?

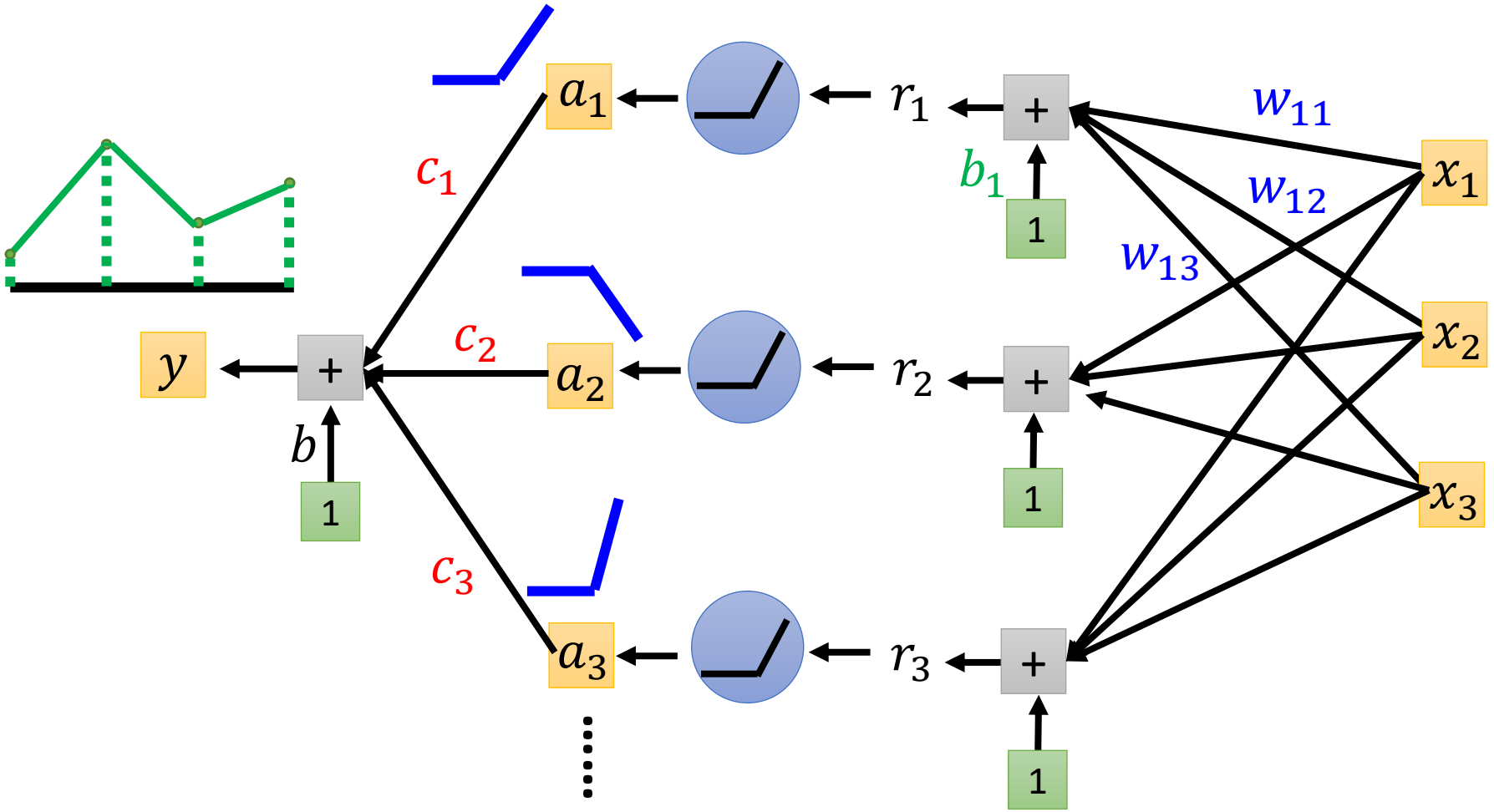


Rectified Linear Unit (ReLU)



Piecewise linear = constant + sum of a set of





Why we want "***Deep***" network, not "***Fat***" network?

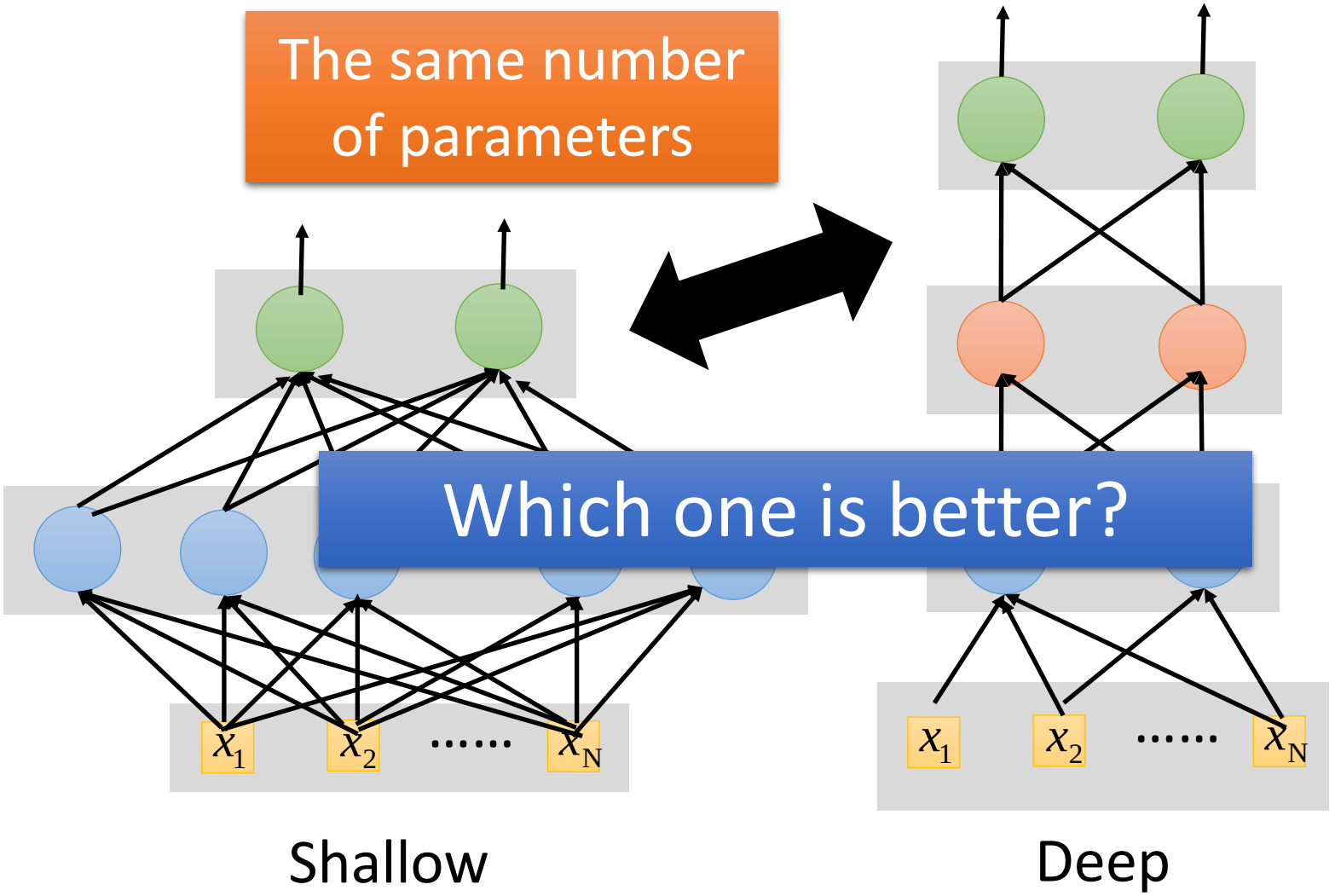
Deeper is Better?

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.

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.

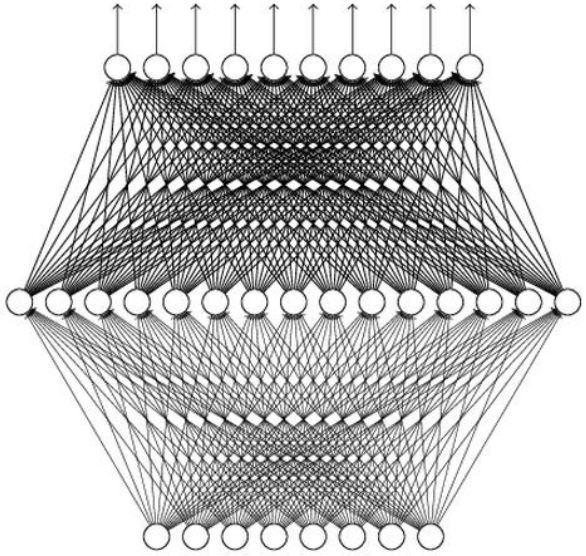
Universality Theorem

Any continuous function f

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

Can be realized by a network with one hidden layer

(given **enough** hidden neurons)

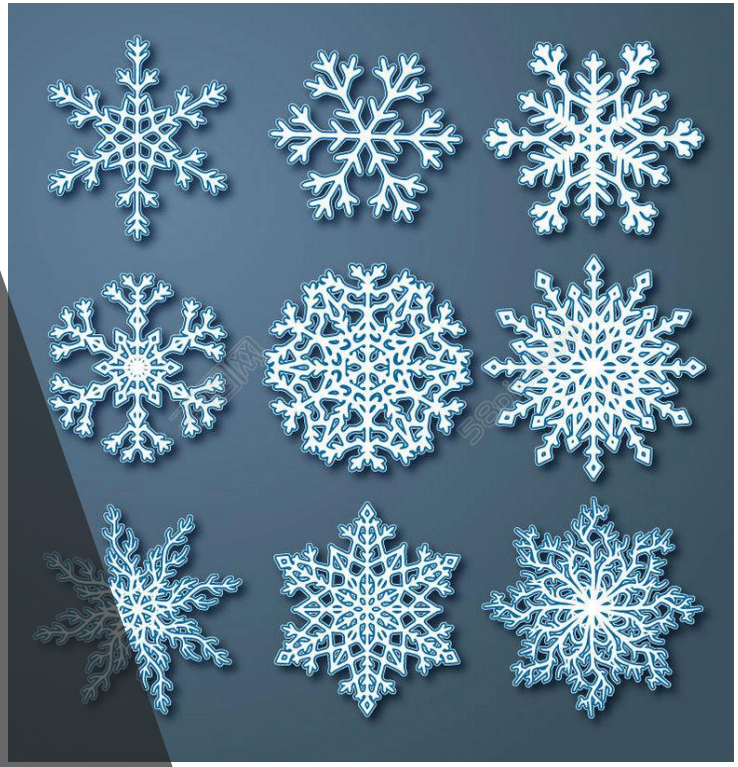


Reference for the reason:
http://neuralnetworksanddeplearning.com/chap4.html

Yes, shallow network can represent any function.

However, using deep structure is more effective.

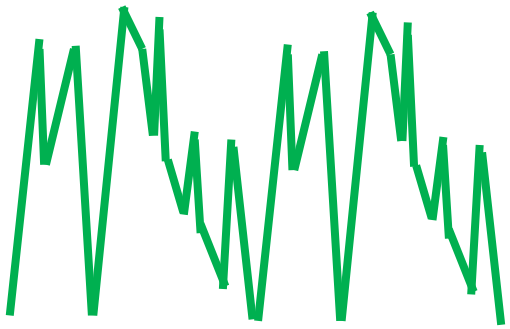
Why we need deep?

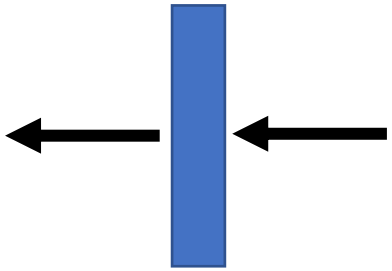


Yes, one hidden layer can represent any function.

However, using deep structure is more effective.

Why we need deep?

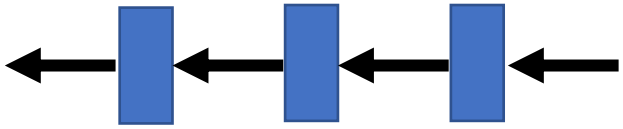




Shallow

v

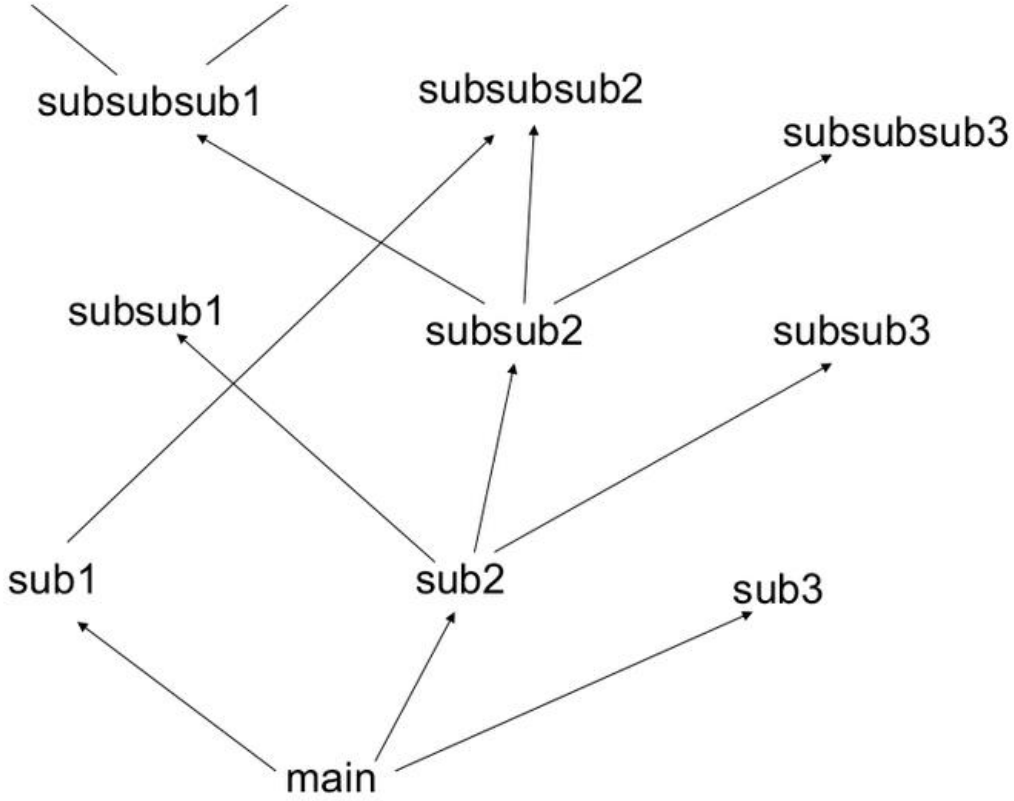
More parameters



Deep

Analogy – Programming

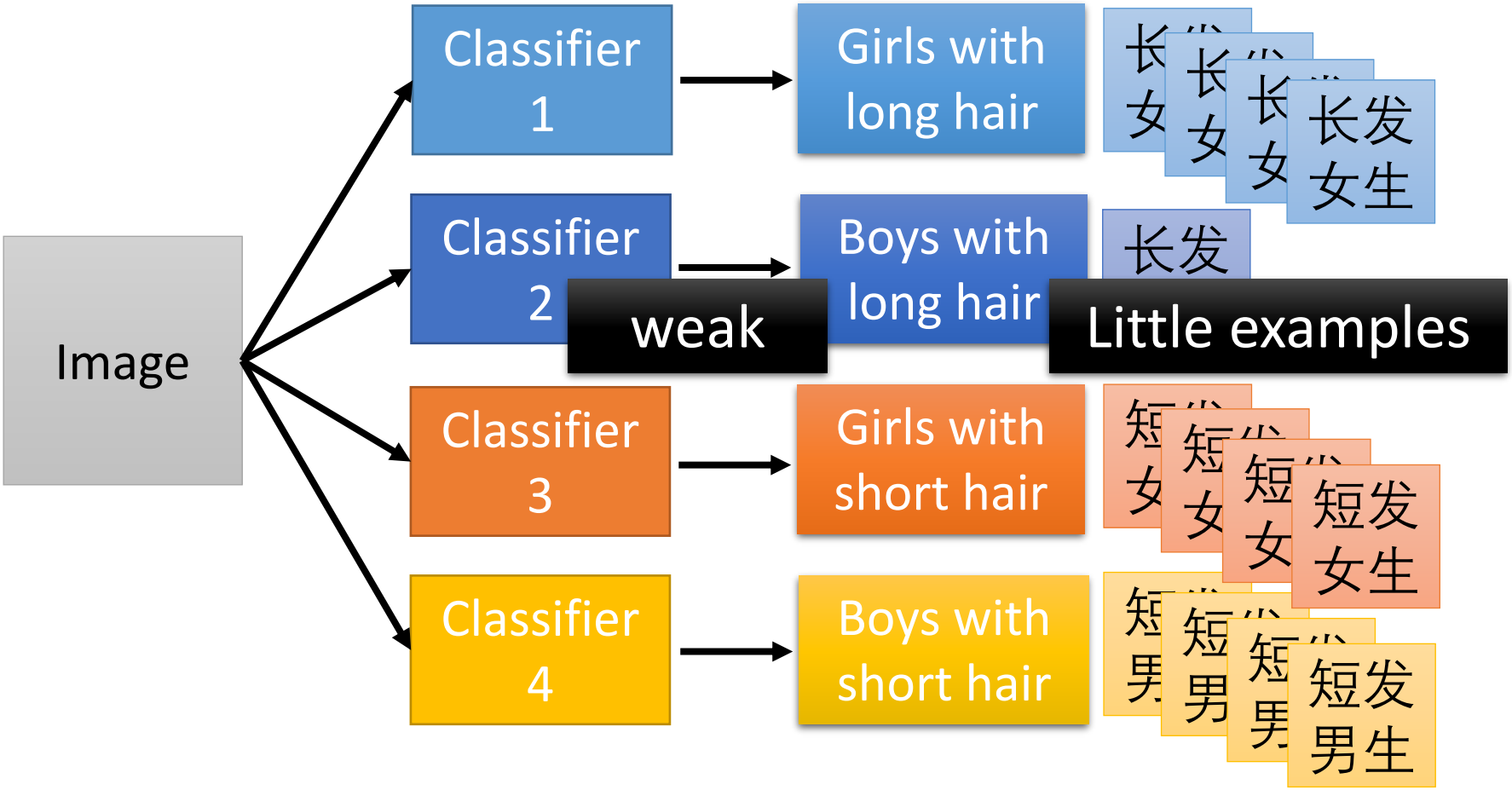
Don’t put everything in your main function.



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

Modularization

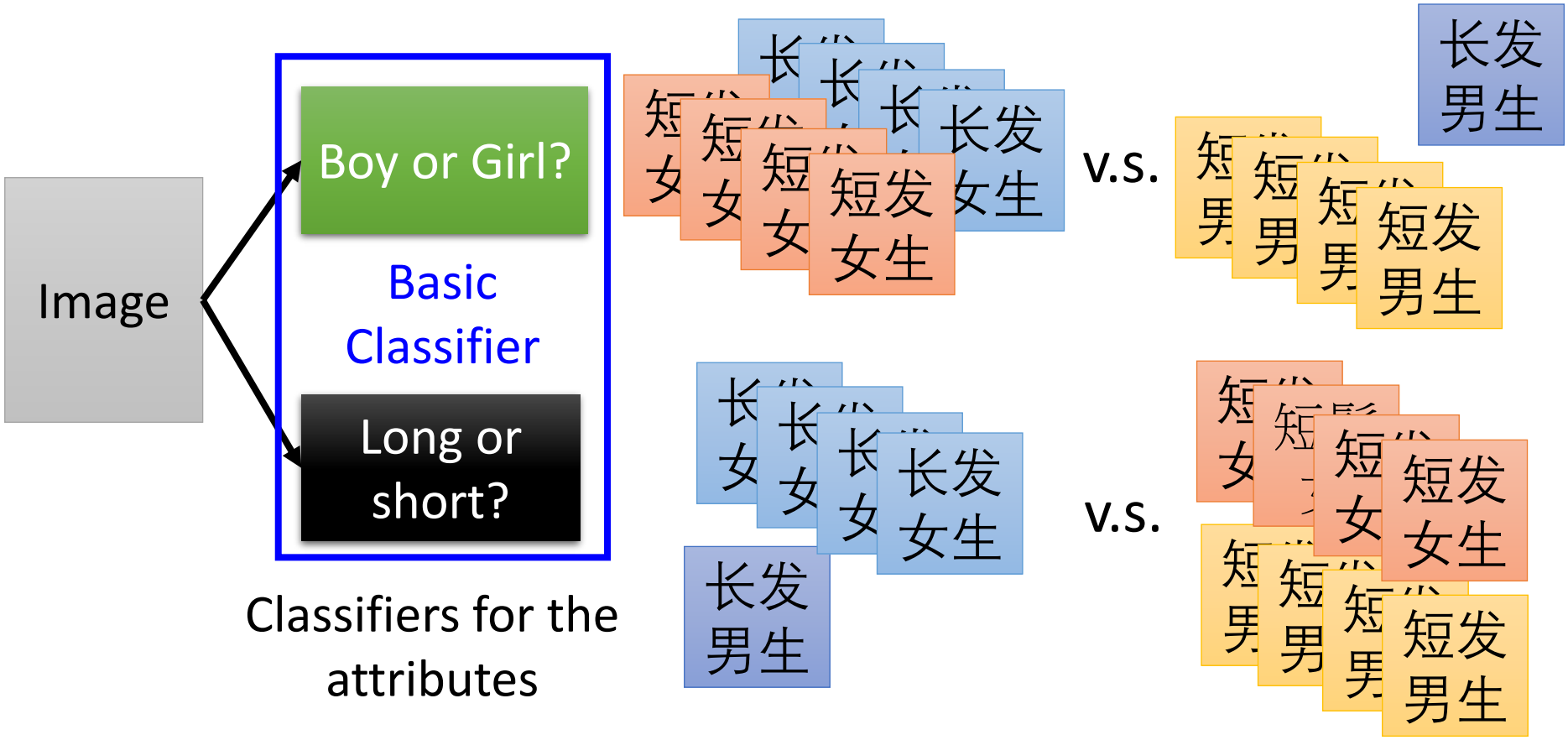
- Deep → Modularization



Modularization

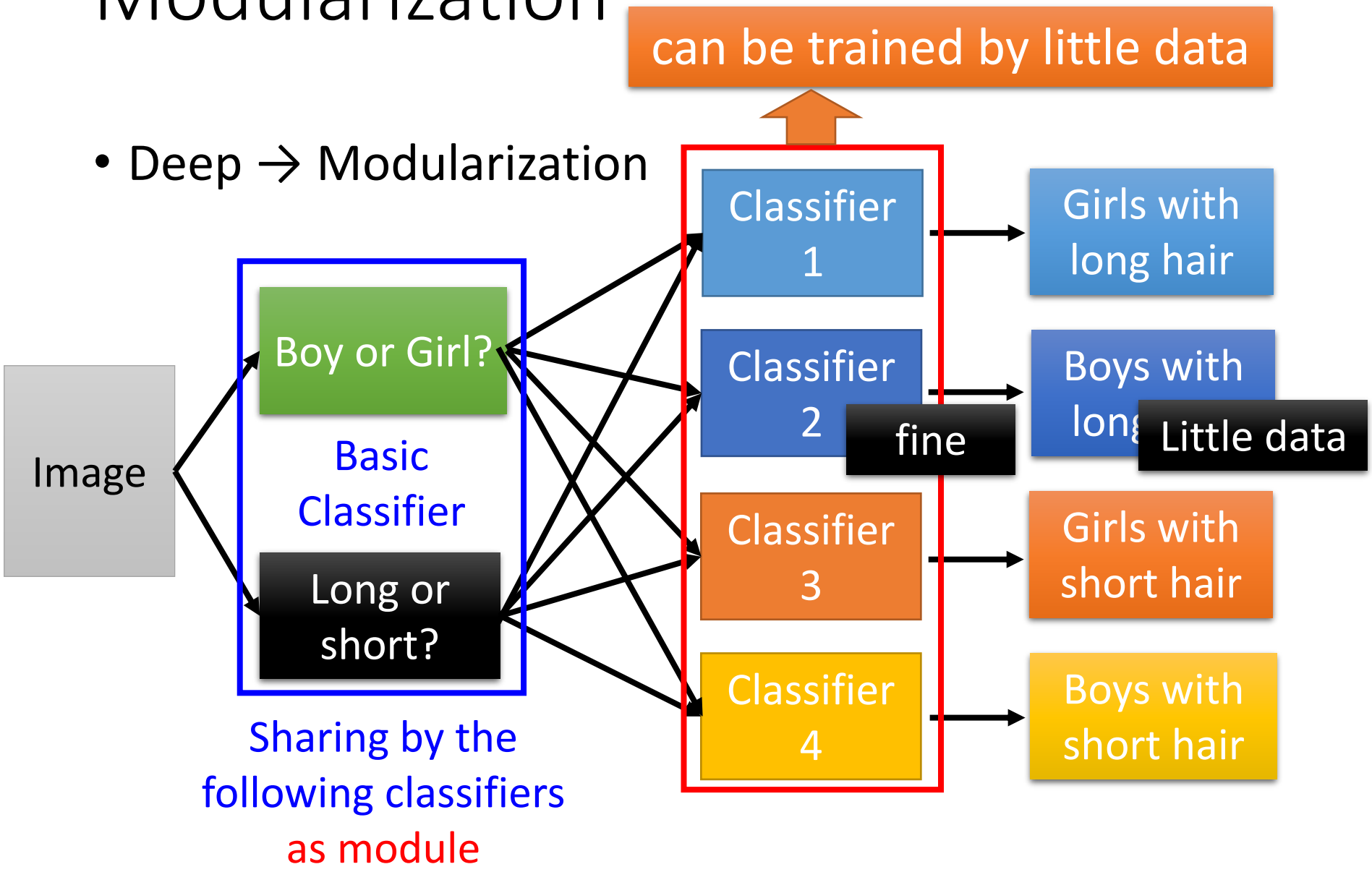
Each basic classifier can have sufficient training examples.

- Deep → Modularization



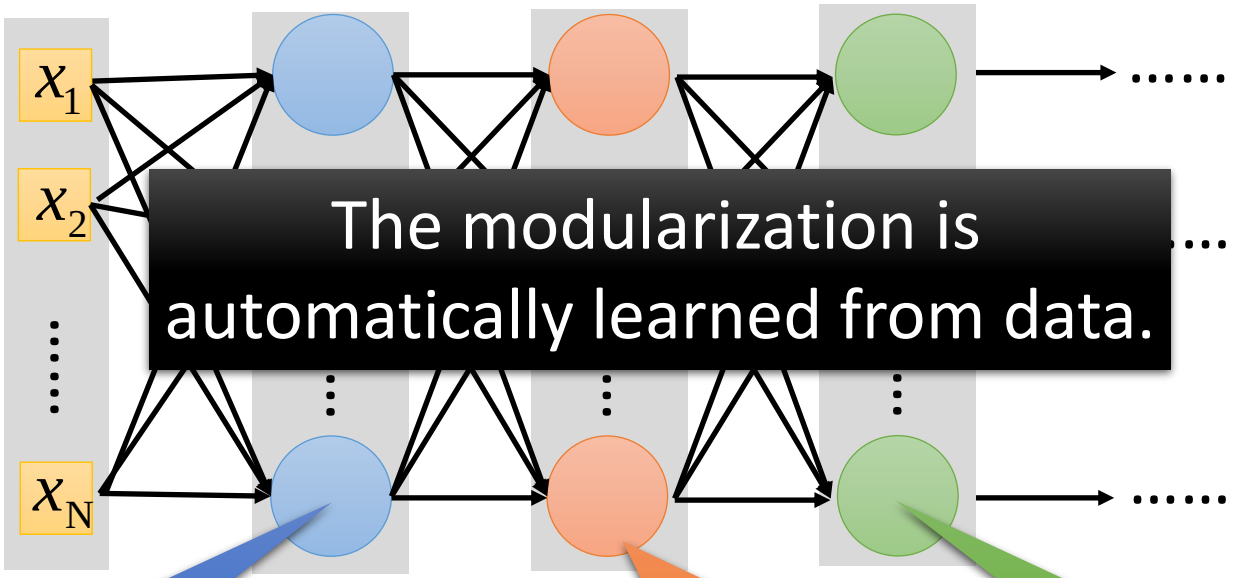
Modularization

- Deep → Modularization



Modularization

- Deep → Modularization → Less training data?



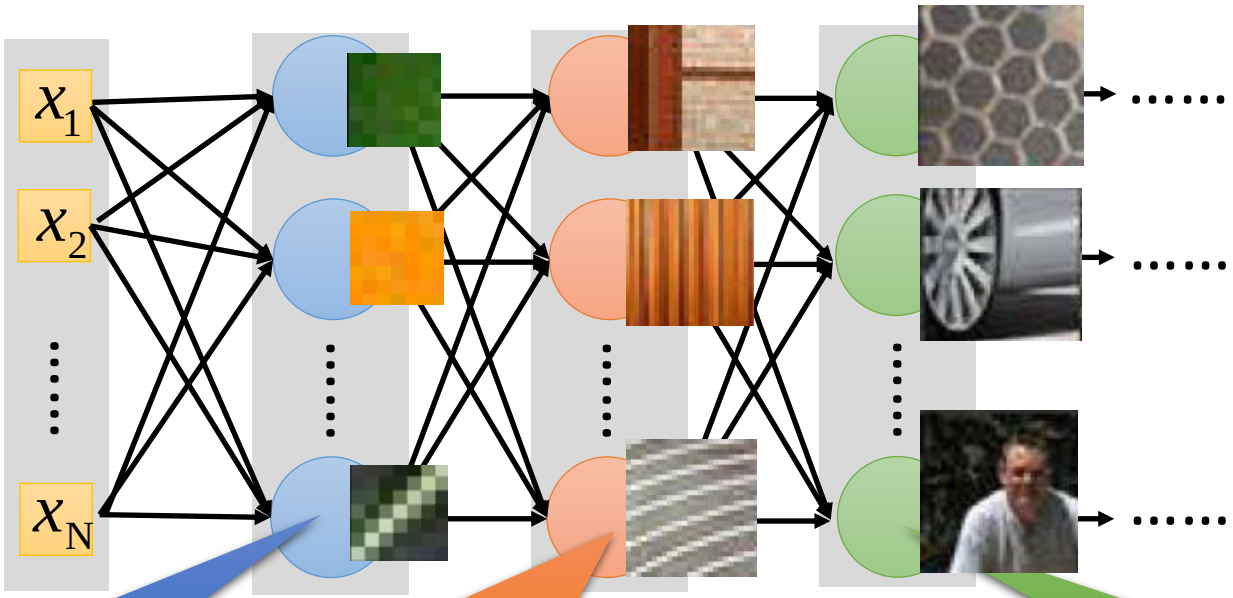
The most basic classifiers

Use 1st layer as module to build classifiers

Use 2nd layer as module .....

Modularization - Image

- Deep → Modularization



The most basic classifiers

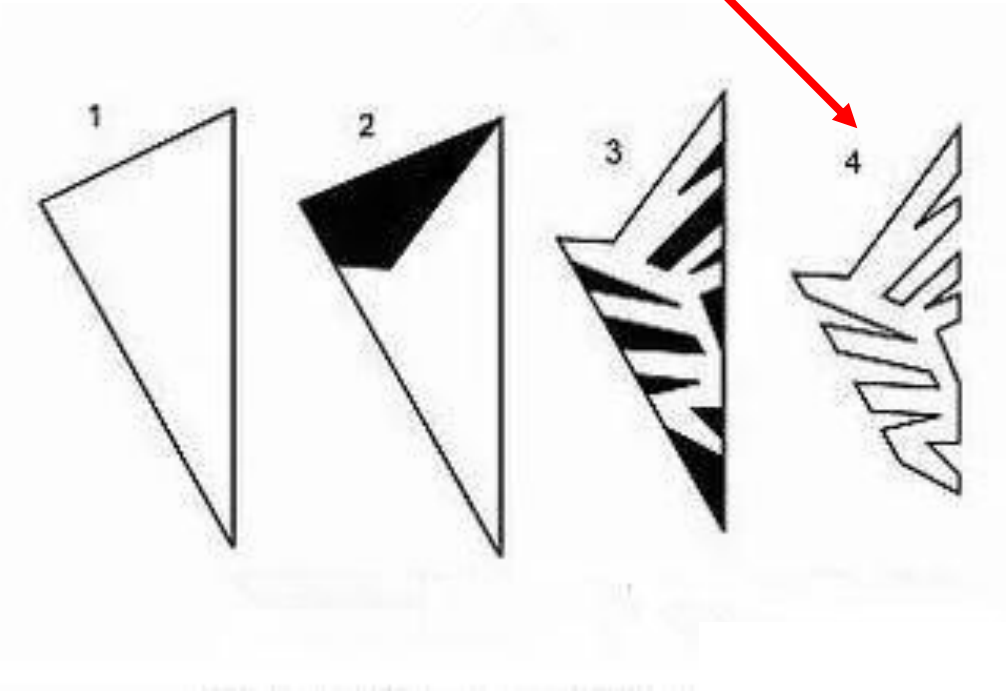
Use 1st layer as module to build classifiers

Use 2nd layer as module ……

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

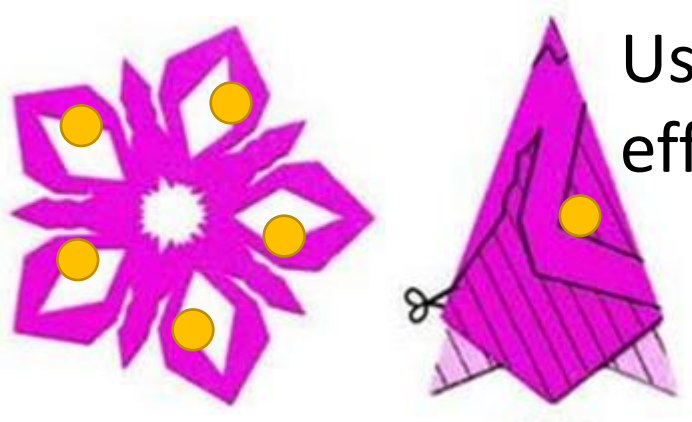
More Analogy

比較有效率

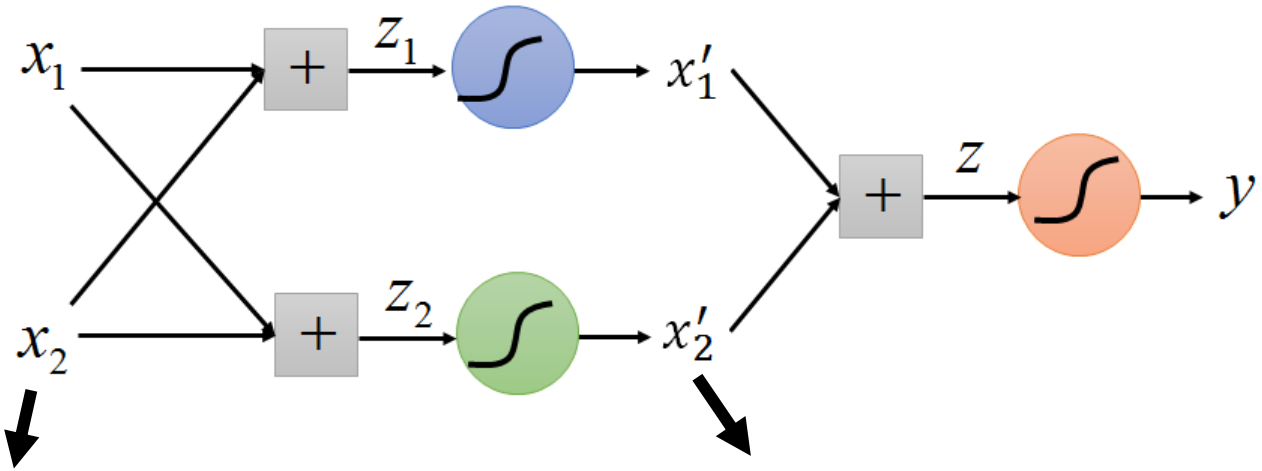


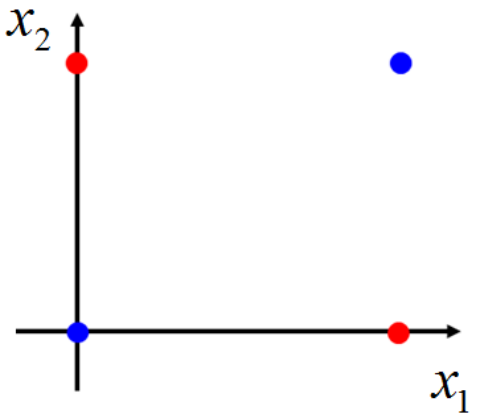
剪很多刀

More Analogy

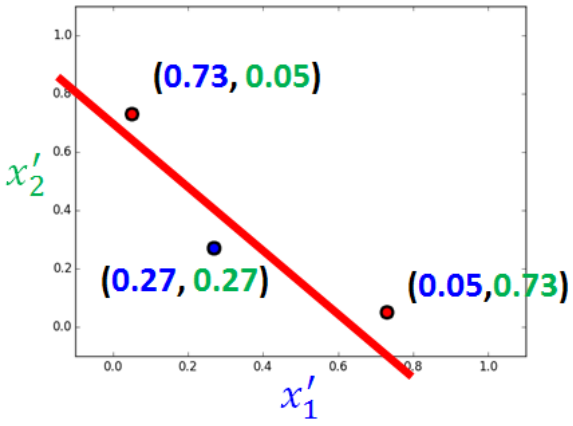


Use data effectively



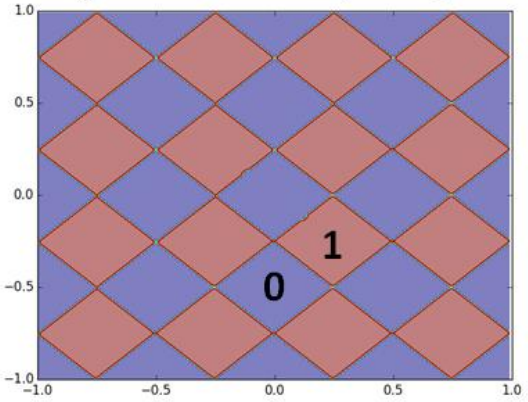


Folding the space



More Analogy - Experiment

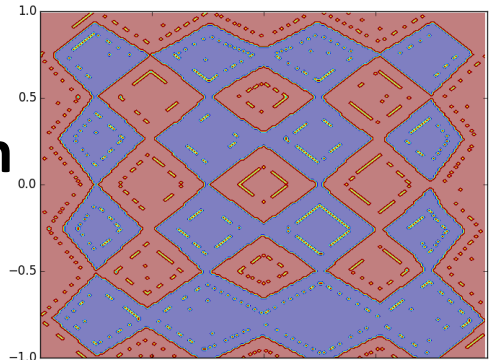
$f : R^2 \rightarrow \{0,1\}$

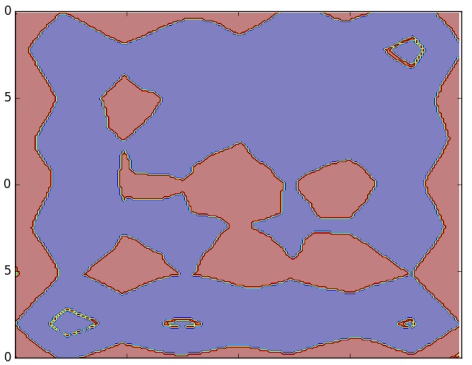


Different numbers of training examples

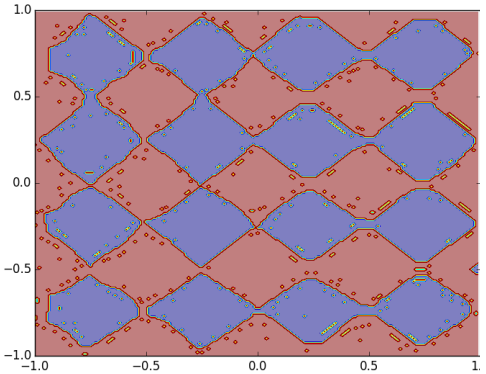
10,0000 2,0000

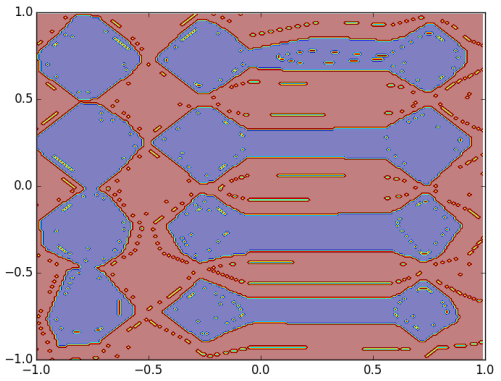
1 hidden layer

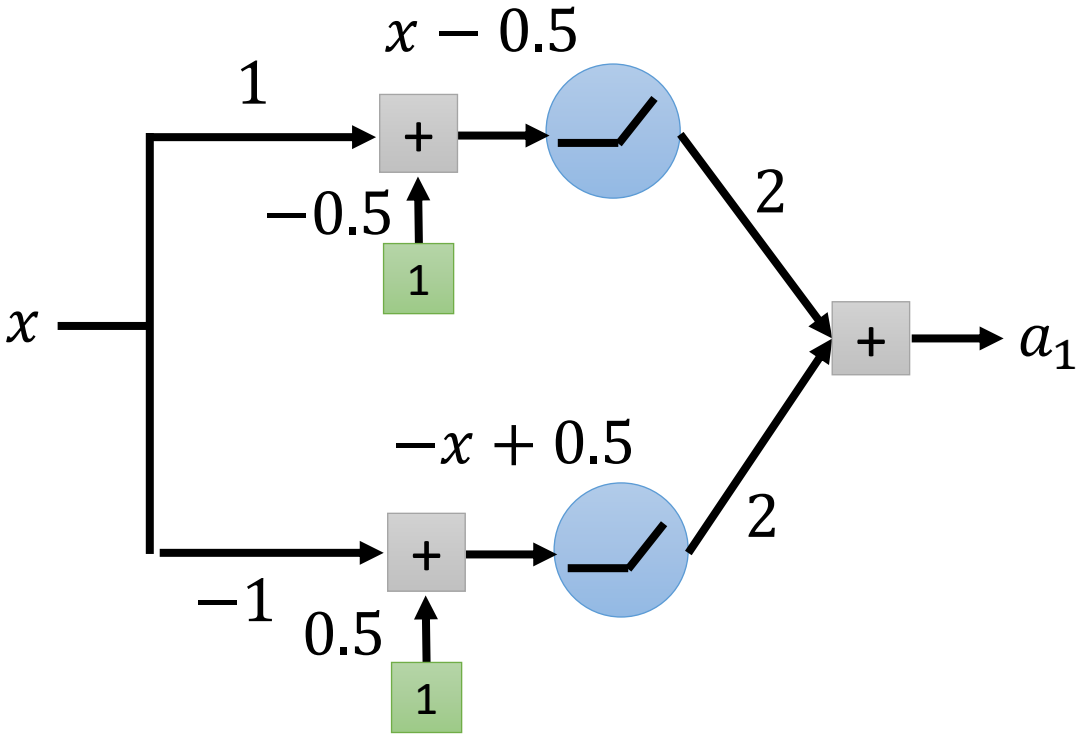


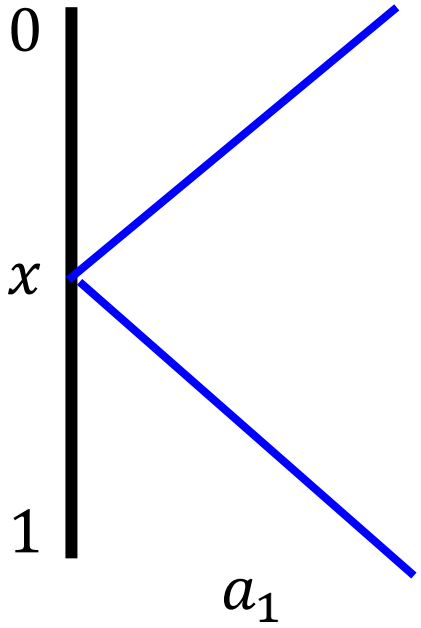


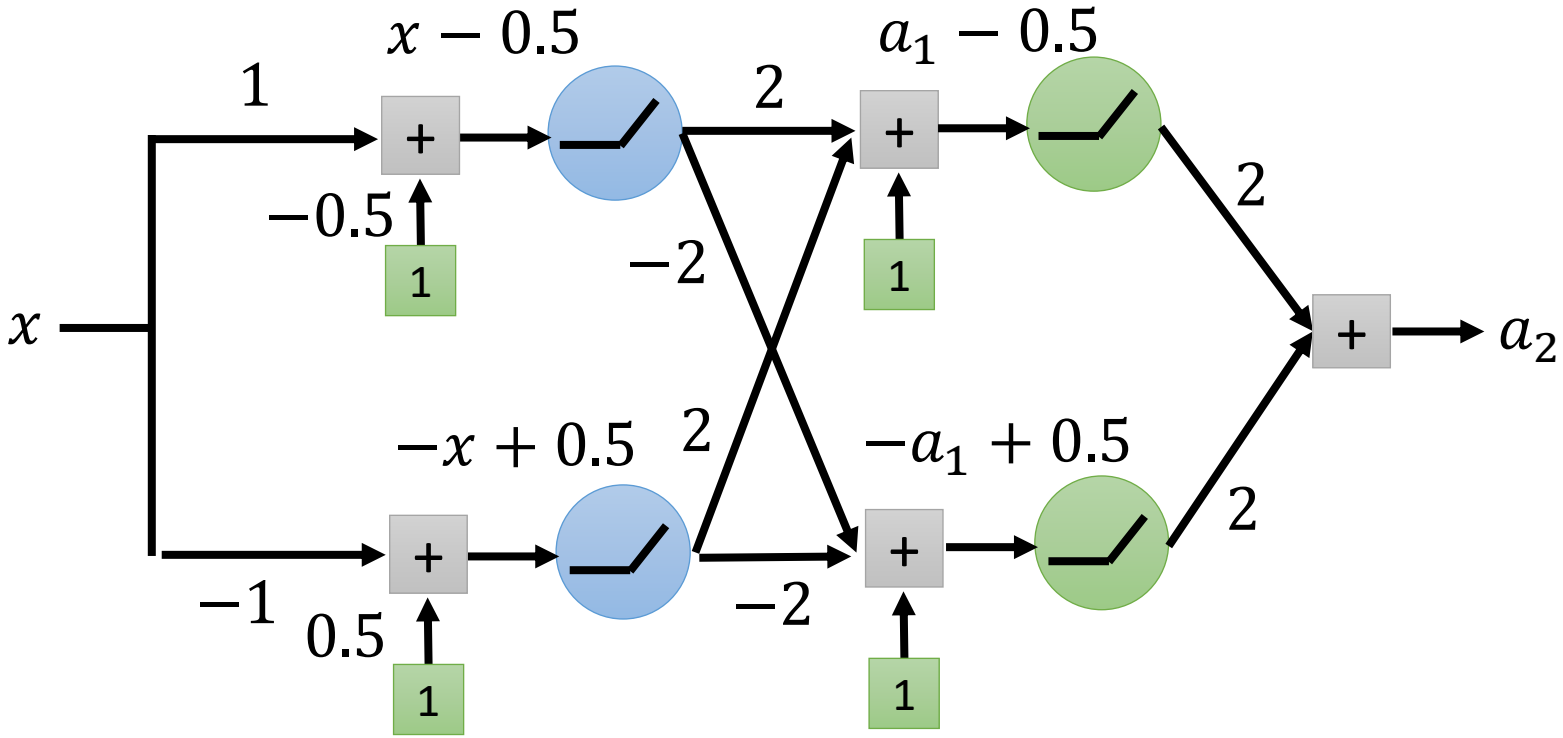
3 hidden layers

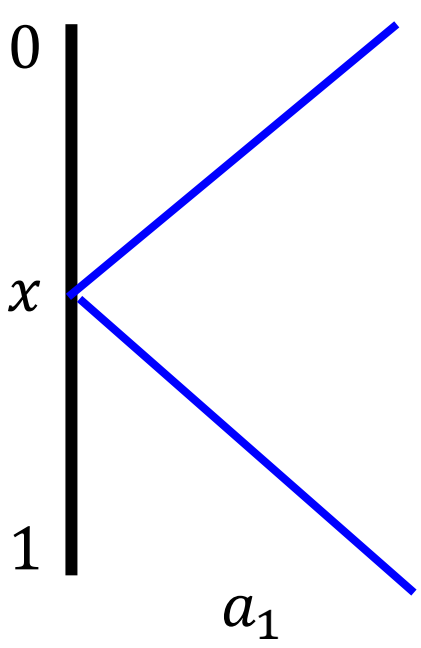


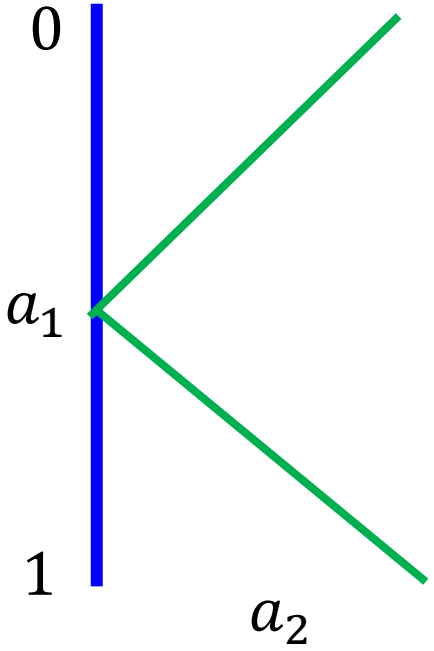


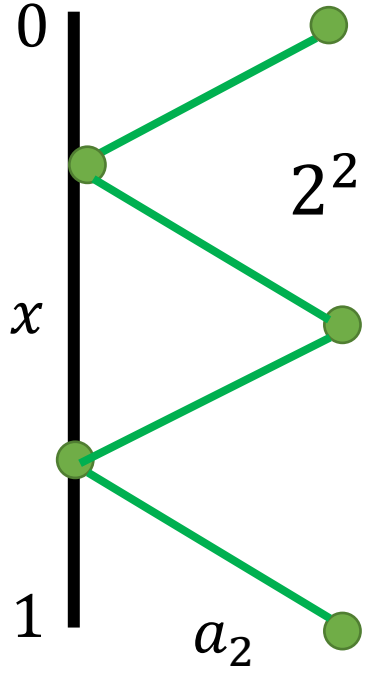

$x - 0.5$
1
+
-0.5
1
x
2
+
a_1
$-x + 0.5$
2
+
-1
0.5
1


0
x
1
a_1

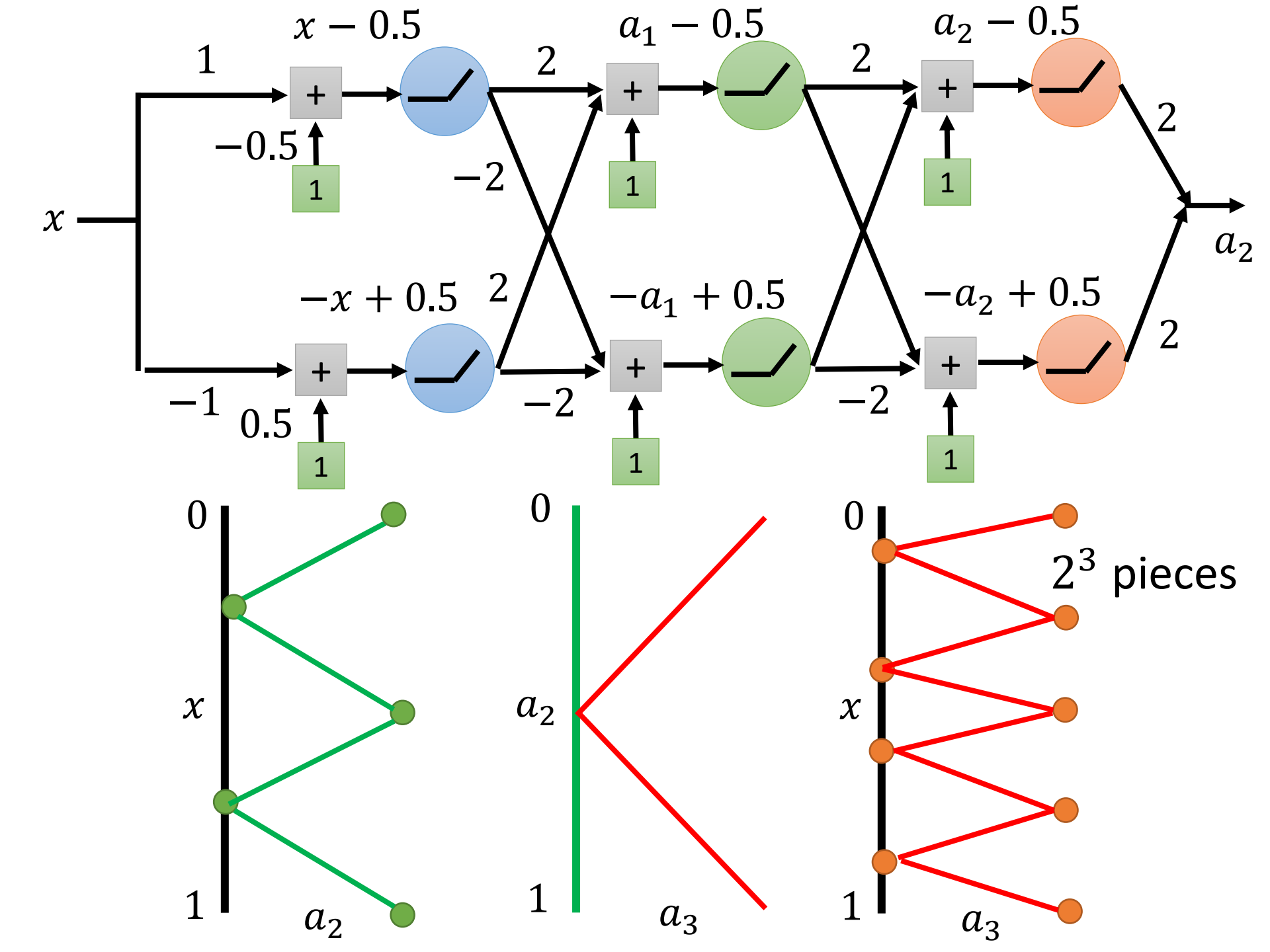




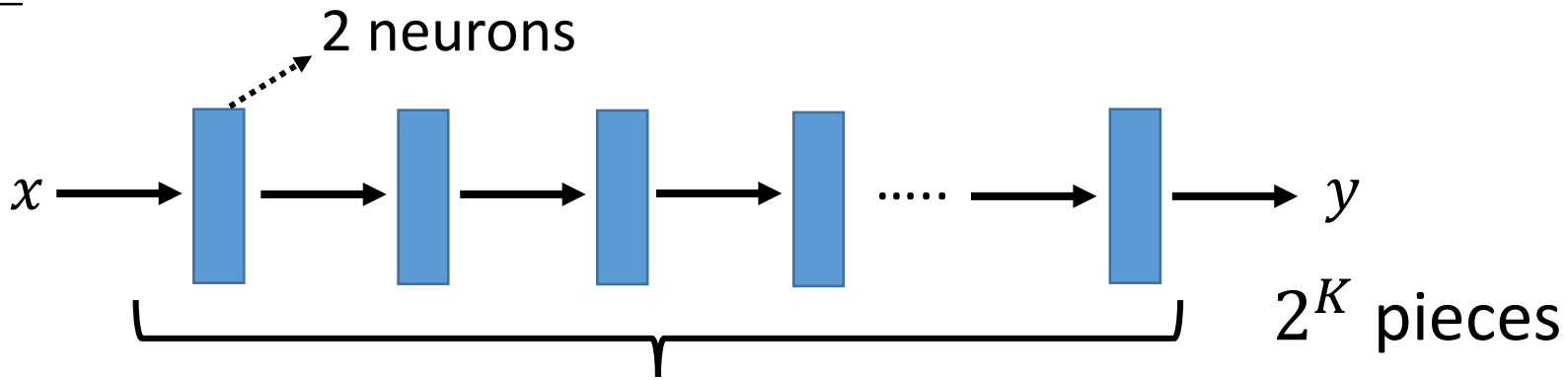




2^2 pieces

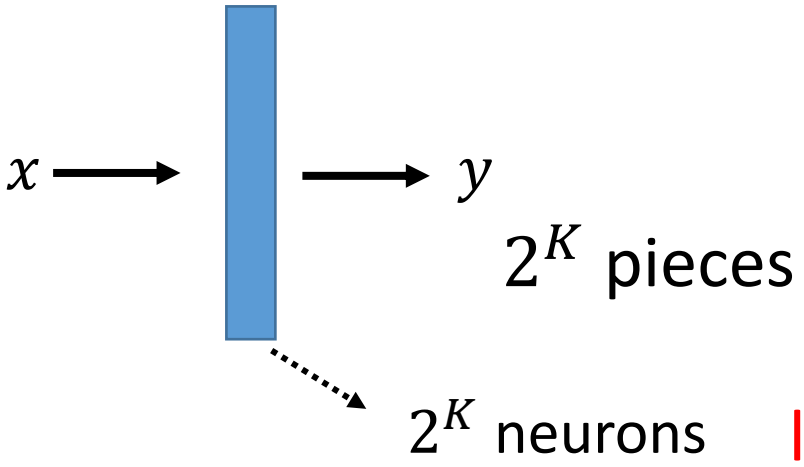

x
$x - 0.5$
$-x + 0.5$
$a_1 - 0.5$
$-a_1 + 0.5$
$a_2 - 0.5$
$-a_2 + 0.5$
a_2
1
-0.5
-1
0.5
2
-2
0
1
a_2
a_3
2^3 pieces

Deep



smaller $|\mathcal{H}|$

Shallow



larger $|\mathcal{H}|$

Thinks more ……

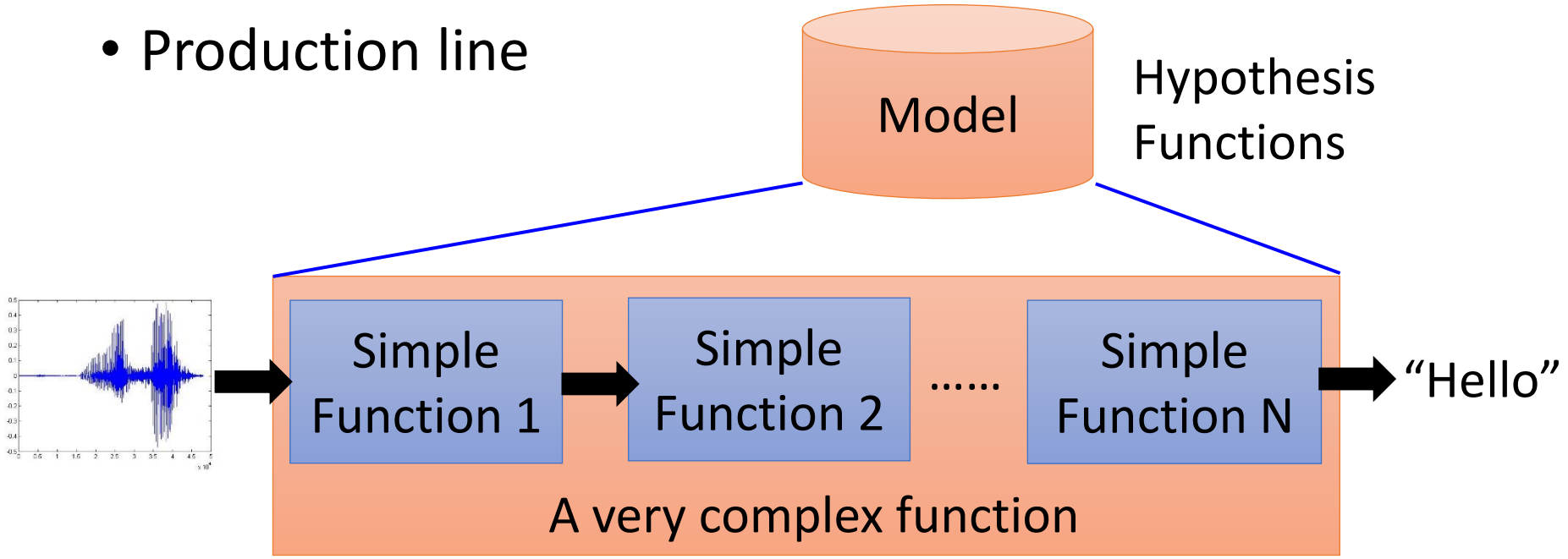
- Deep networks outperforms shallow ones when the required functions are complex and regular.

  Image, speech, etc. have this characteristics.

- Deep is exponentially better than shallow even when $y = x^2$.

End-to-end Learning

- Production line

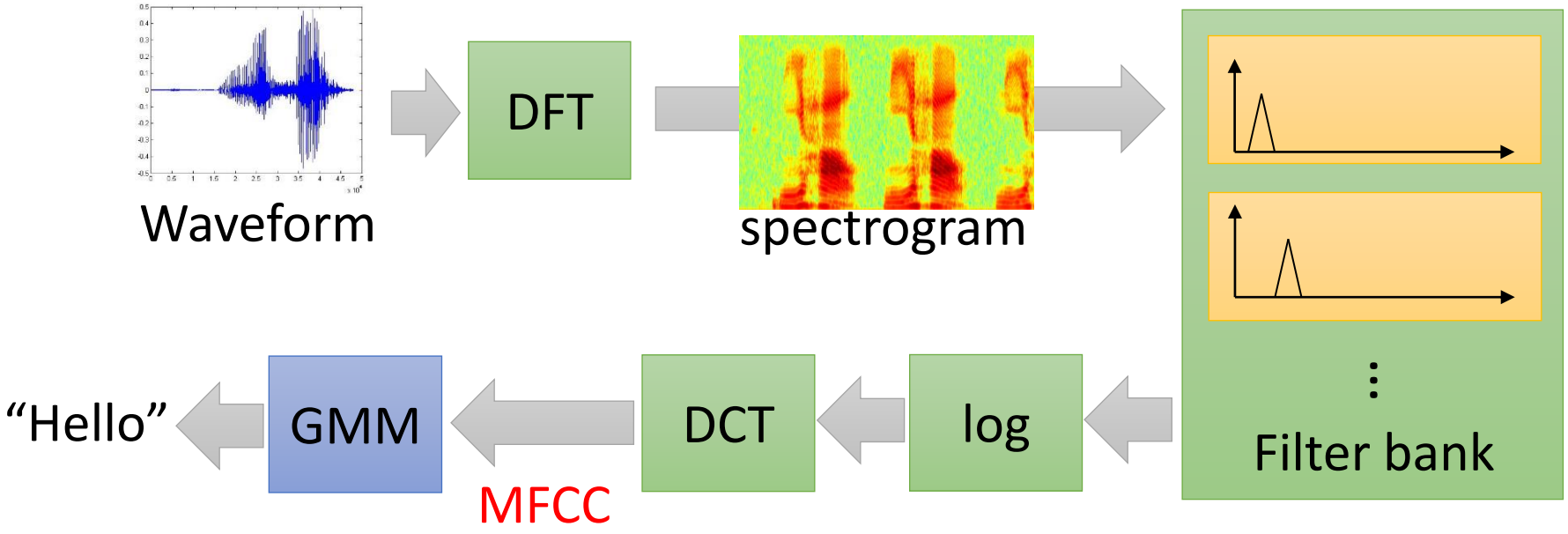


End-to-end training:

What each function should do is learned automatically

End-to-end Learning - Speech Recognition

- Shallow Approach

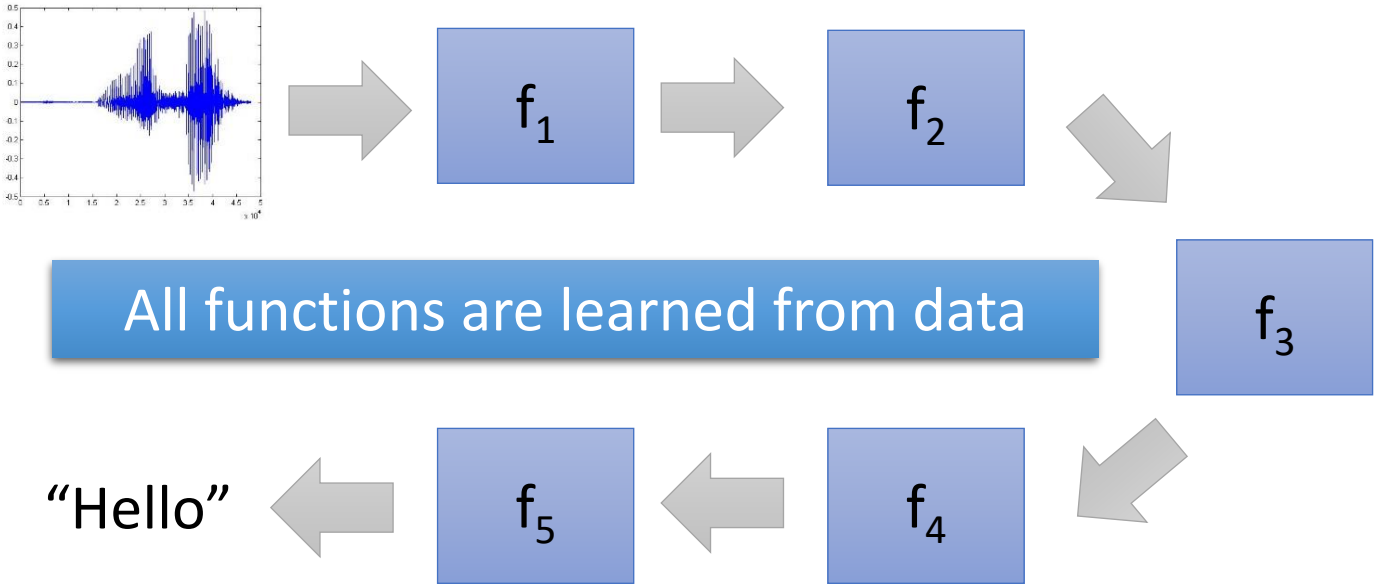


Each box is a simple function in the production line:

:hand-crafted :learned from data

End-to-end Learning - Speech Recognition

- Deep Learning

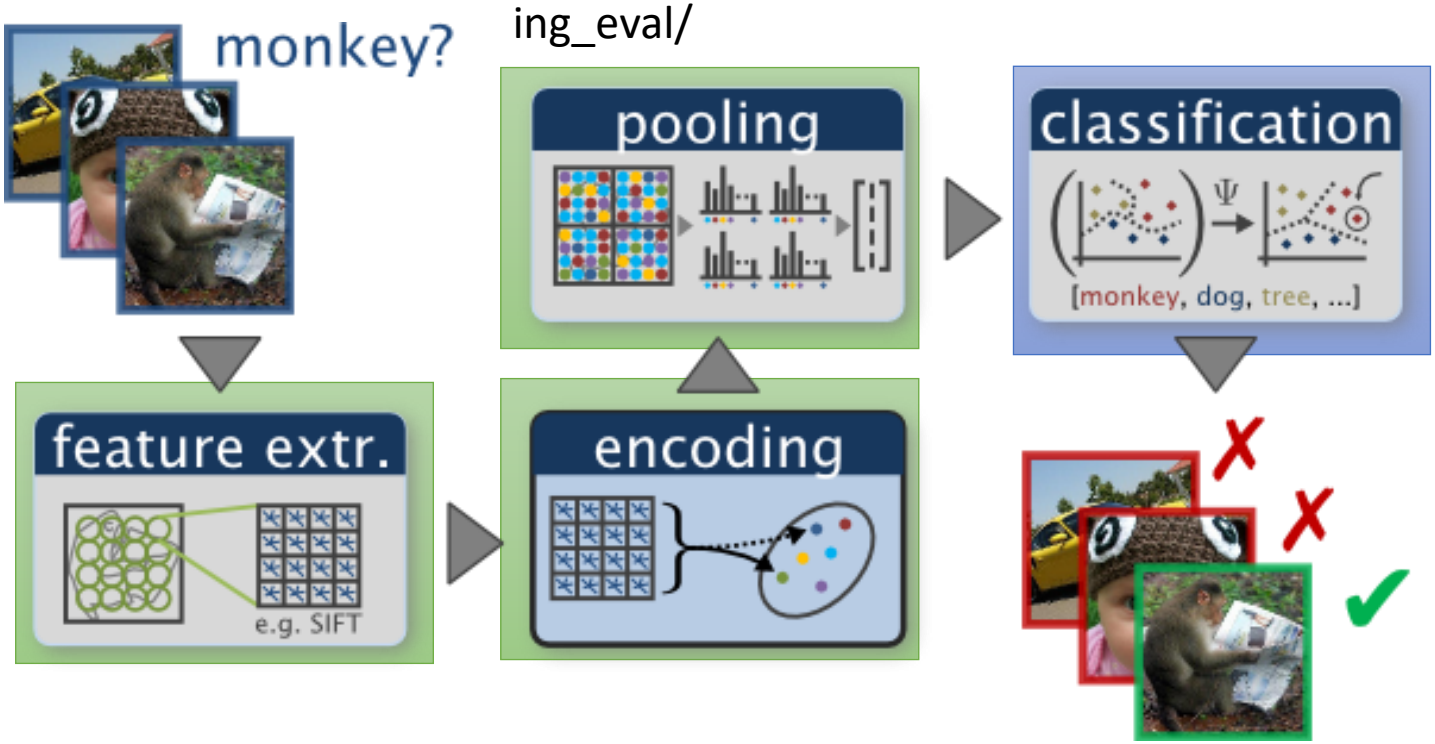


Less engineering labor, but machine learns more

End-to-end Learning - Image Recognition

- Shallow Approach

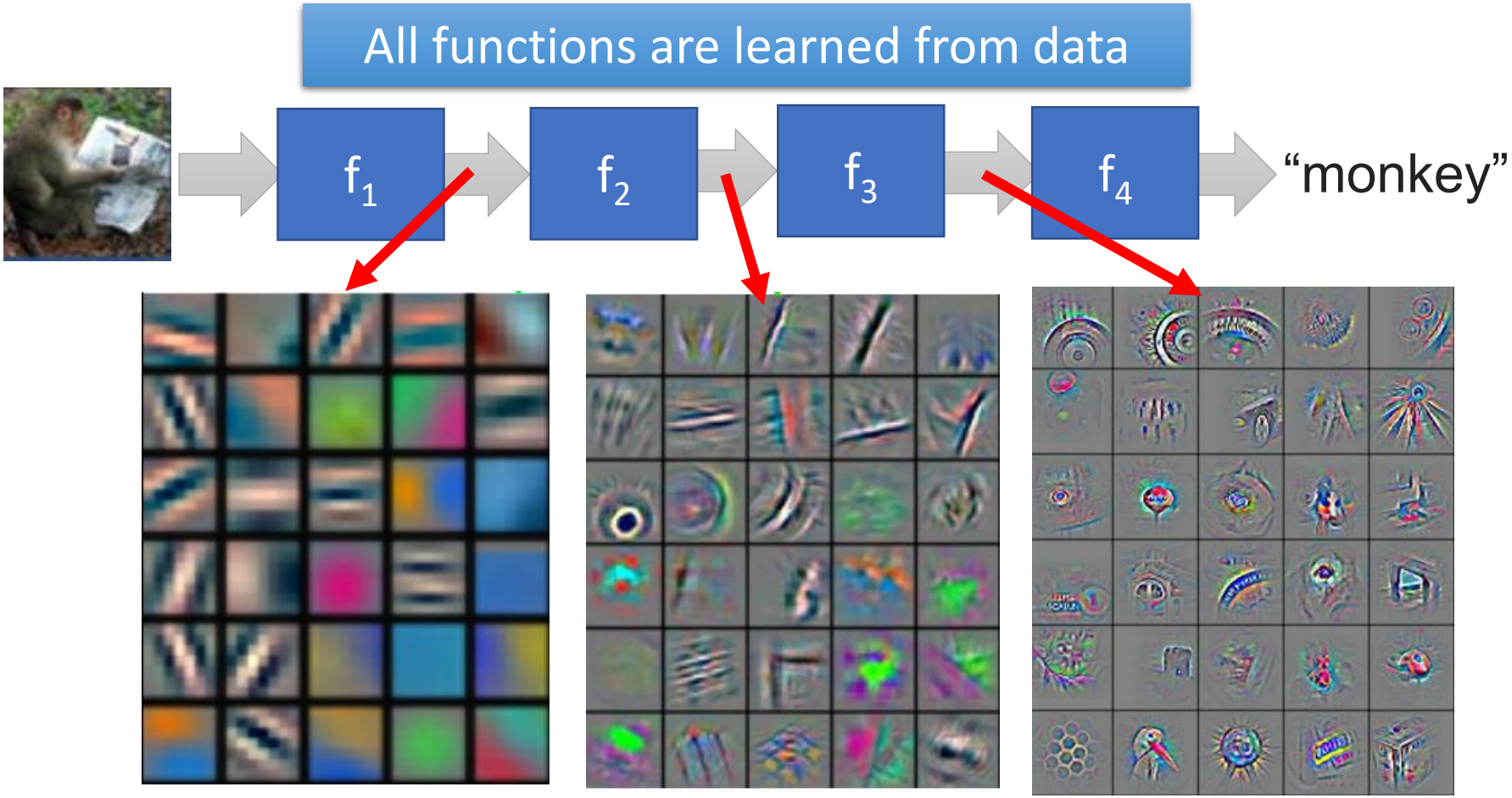
http://www.robots.ox.ac.uk/~vgg/research/encoding_eval/



End-to-end Learning - Image Recognition

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

- Deep Learning



Complex Task …

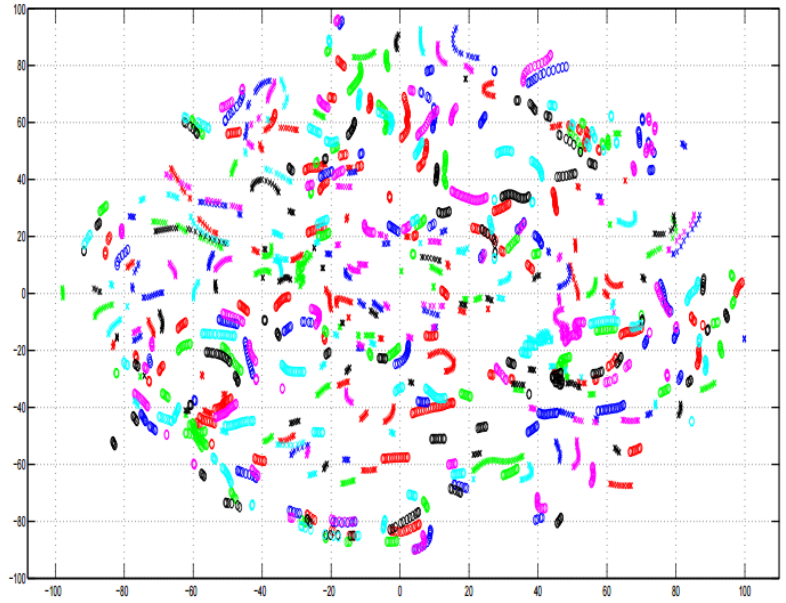
- Very similar input, different output

- Very different input, similar output

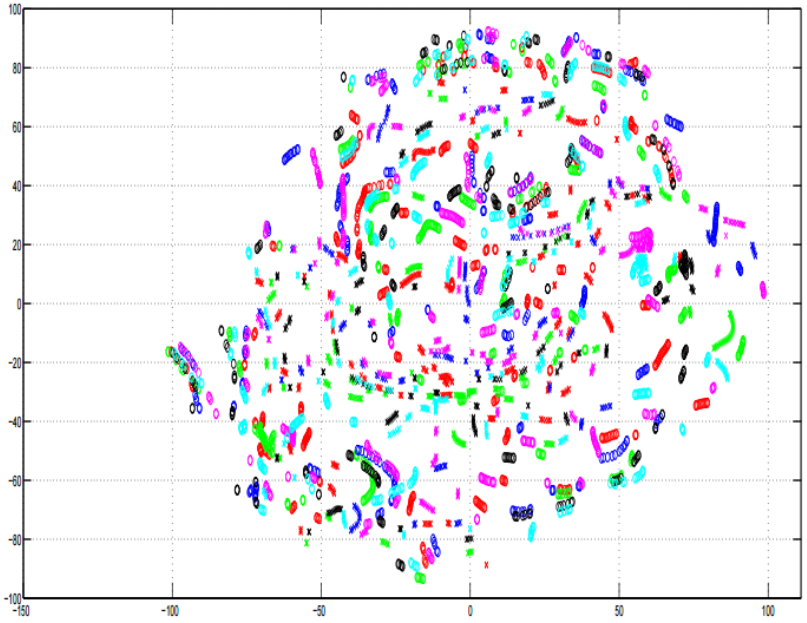
Complex Task …

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)

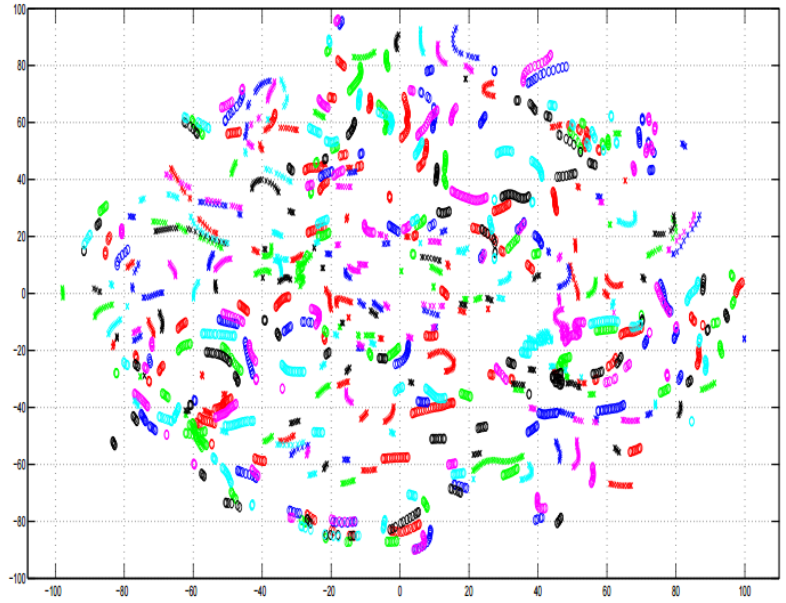


1-st Hidden Layer

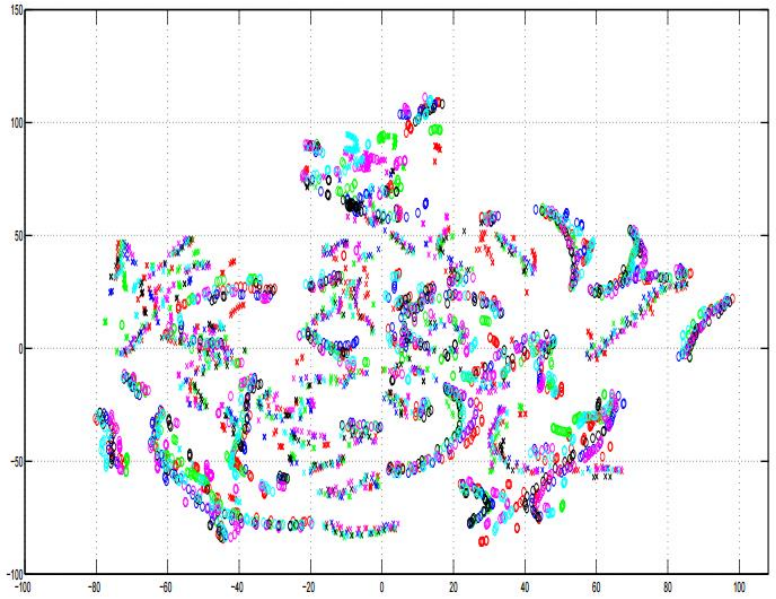
Complex Task …

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

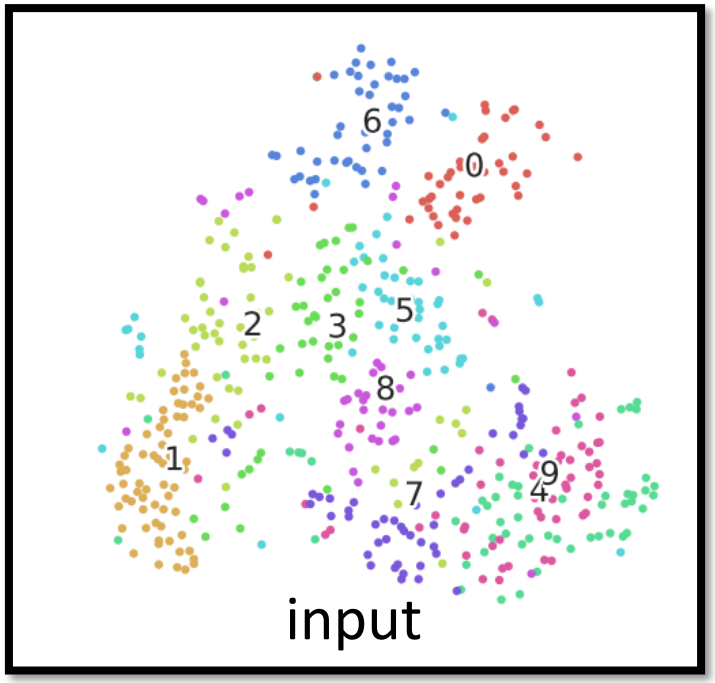


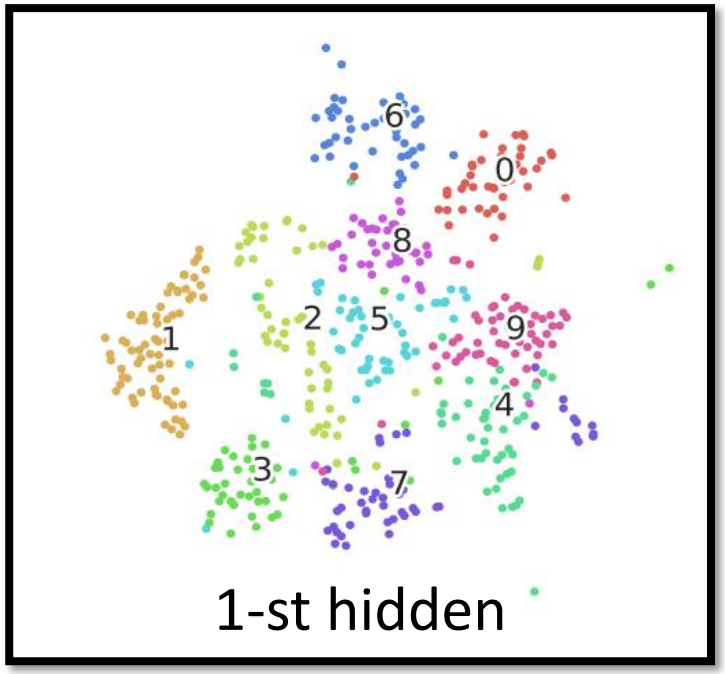
Input Acoustic Feature (MFCC)

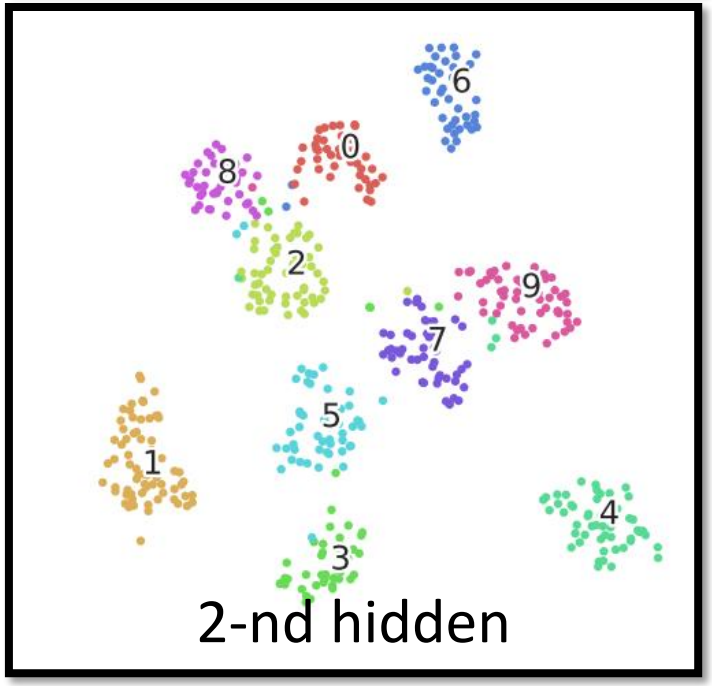


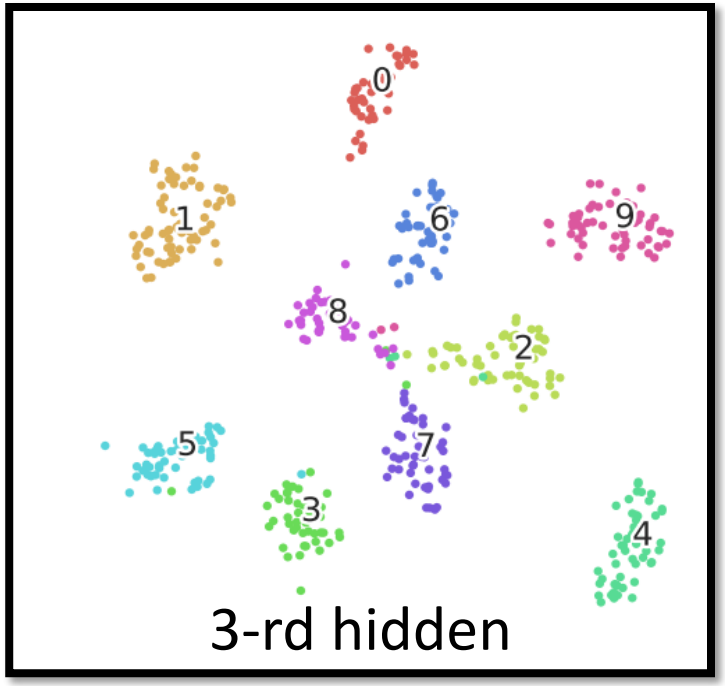
8-th Hidden Layer

MNIST


6
0
2
3
5
8
1
9
4
7
input

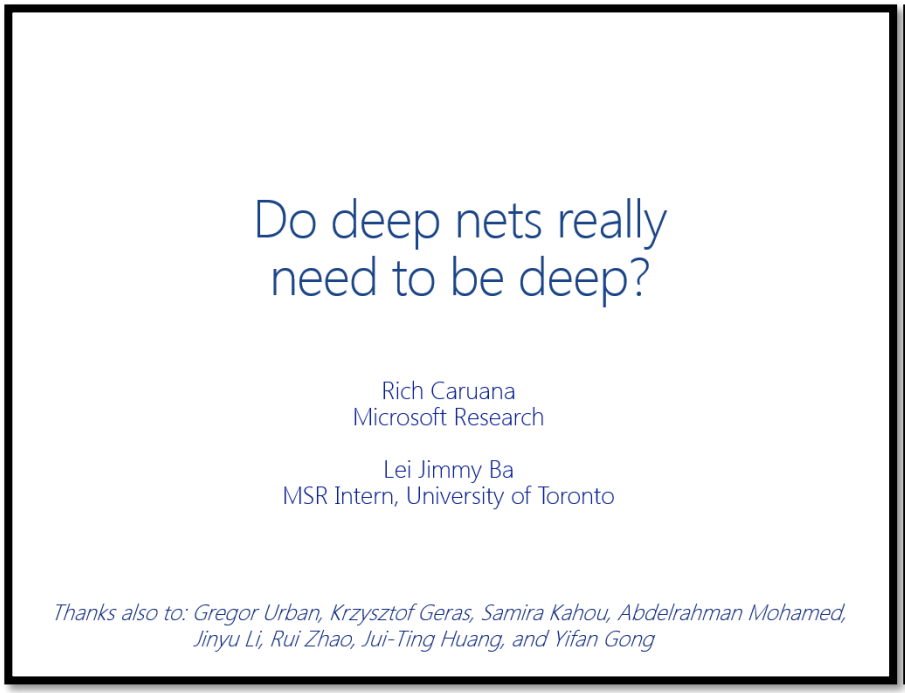

6
0
8
1
2
5
9
4
3
7
1-st hidden

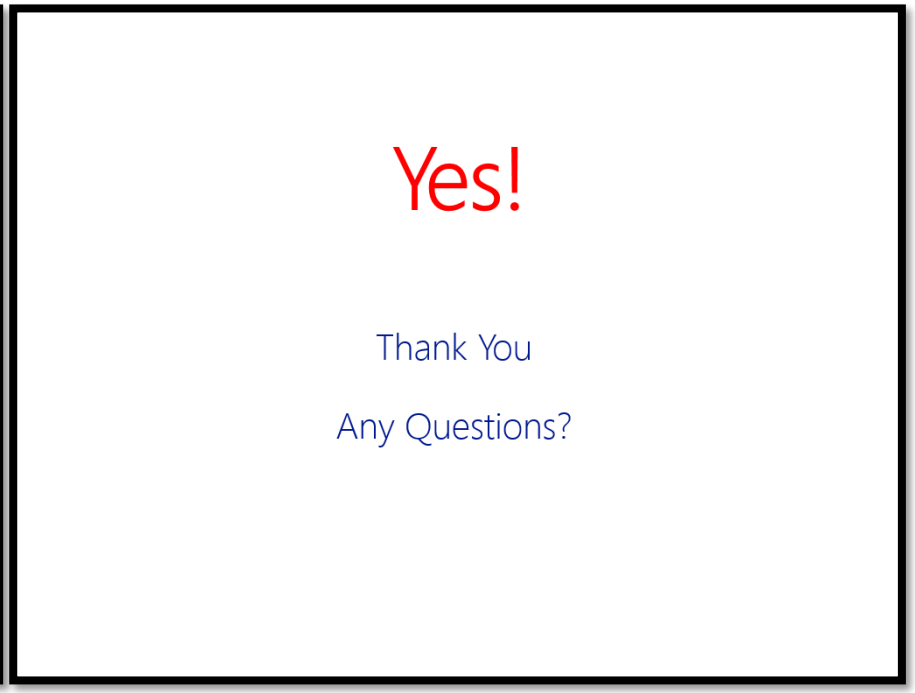

6
0
8
2
9
7
5
1
4
3
2-nd hidden


0
1
6
9
8
2
5
7
3
4
3-rd hidden

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





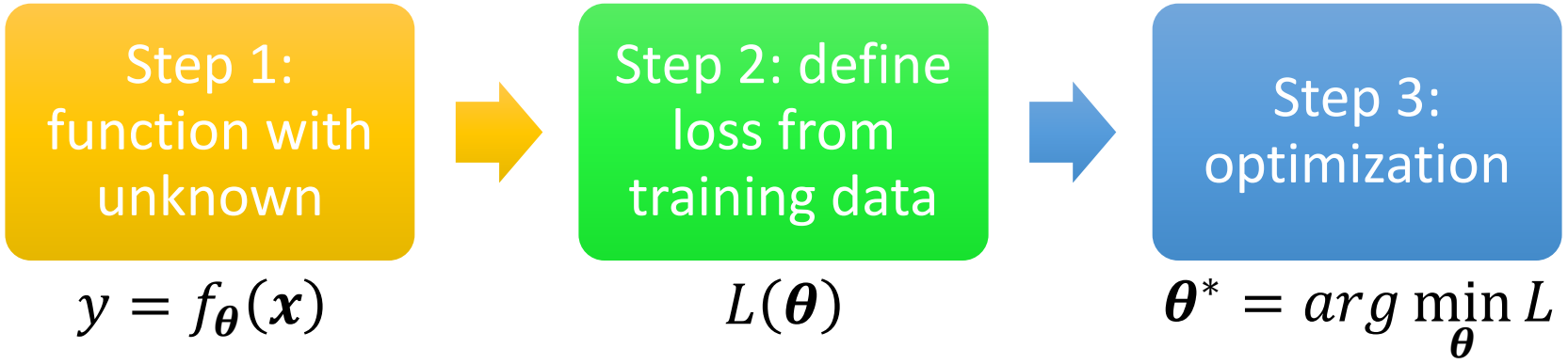
keynote of Rich Caruana at ASRU 2015

General Guidance for Deep Learning

Framework of ML

Training data: $\{(\boldsymbol{x^1}, \hat{y}^1), (\boldsymbol{x^2}, \hat{y}^2), \ldots, (\boldsymbol{x^N}, \hat{y}^N)\}$

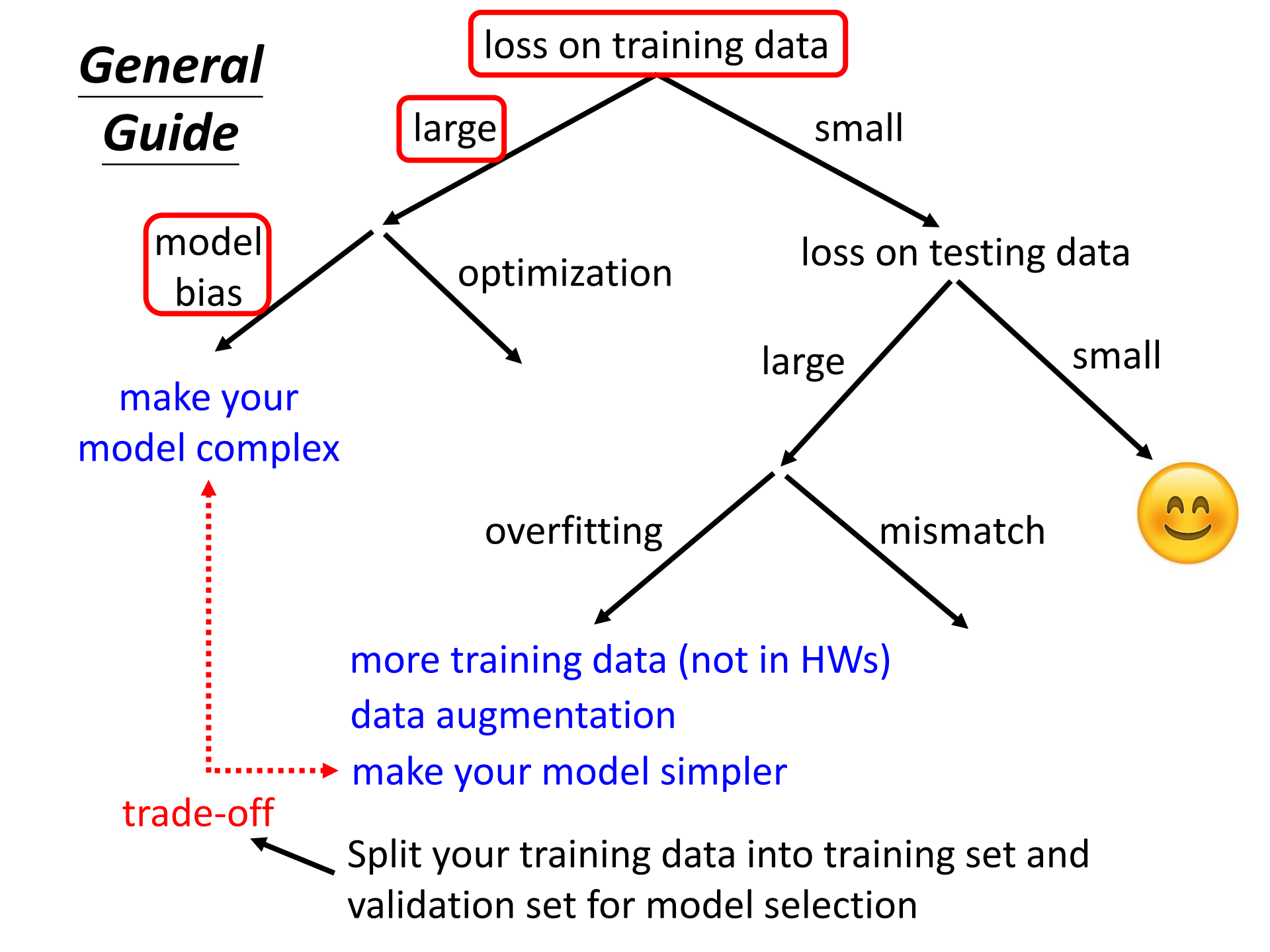
Training:



Testing data: $\{\boldsymbol{x^{N+1}}, \boldsymbol{x^{N+2}}, \ldots, \boldsymbol{x^{N+M}}\}$

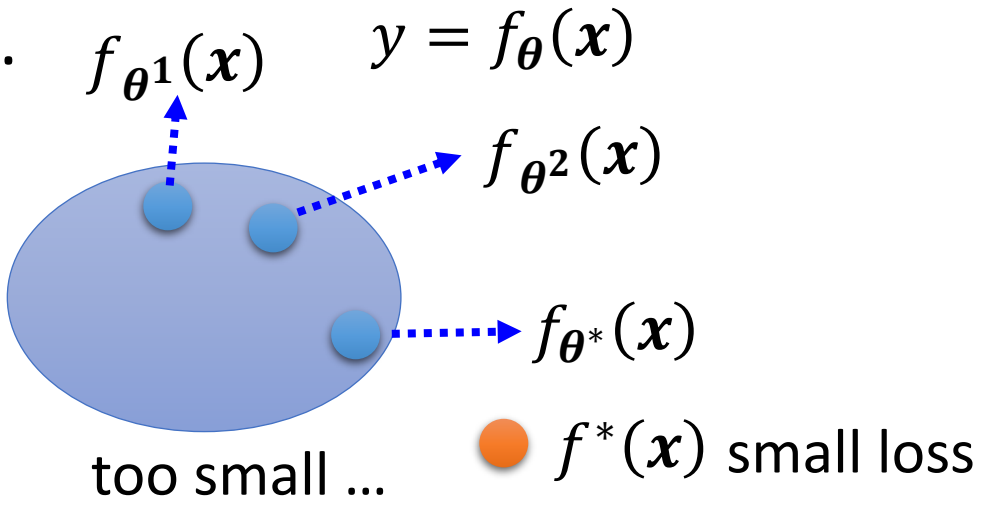
Use $y = f_{\boldsymbol{\theta}^*}(\boldsymbol{x})$ to label the testing data

$\{y^{N+1}, y^{N+2}, \ldots, y^{N+M}\}$ → Upload to Kaggle


General Guide
loss on training data
large
small
model bias
optimization
loss on testing data
large
small
make your model complex
overfitting
mismatch
more training data (not in HWs)
data augmentation
make your model simpler
trade-off
Split your training data into training set and validation set for model selection

Model Bias

- The model is too simple.



find a needle in a haystack …

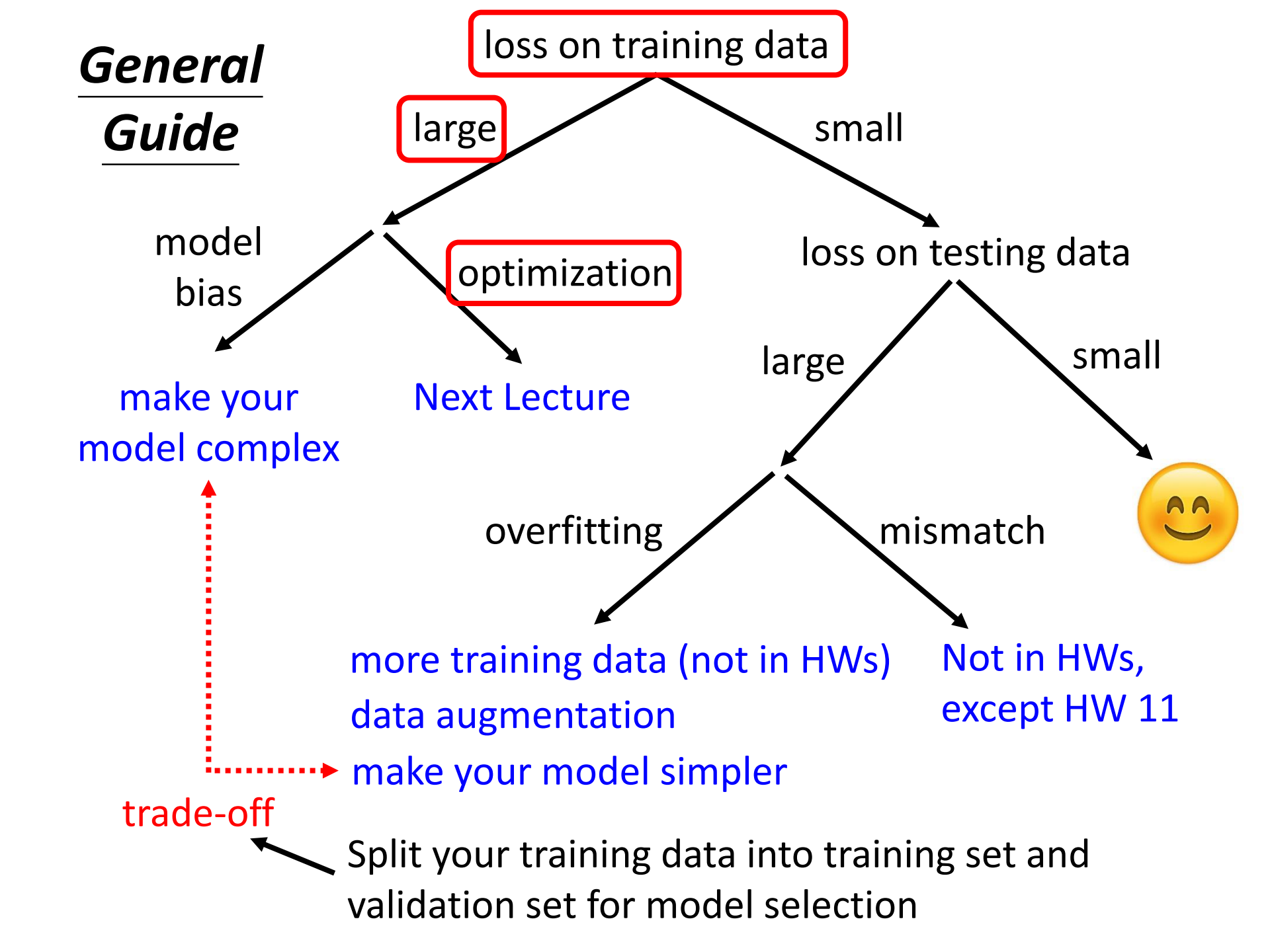
… but there is no needle

- Solution: redesign your model to make it more flexible

$$y = b + wx_1 \xrightarrow{\text{More features}} y = b + \sum_{j=1}^{56} w_j x_j$$

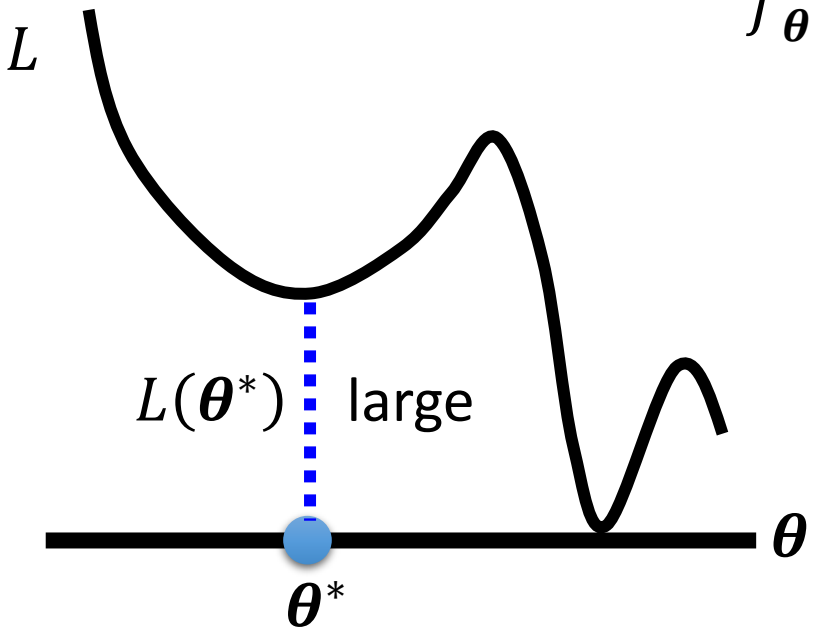
Deep Learning
(more neurons, layers)

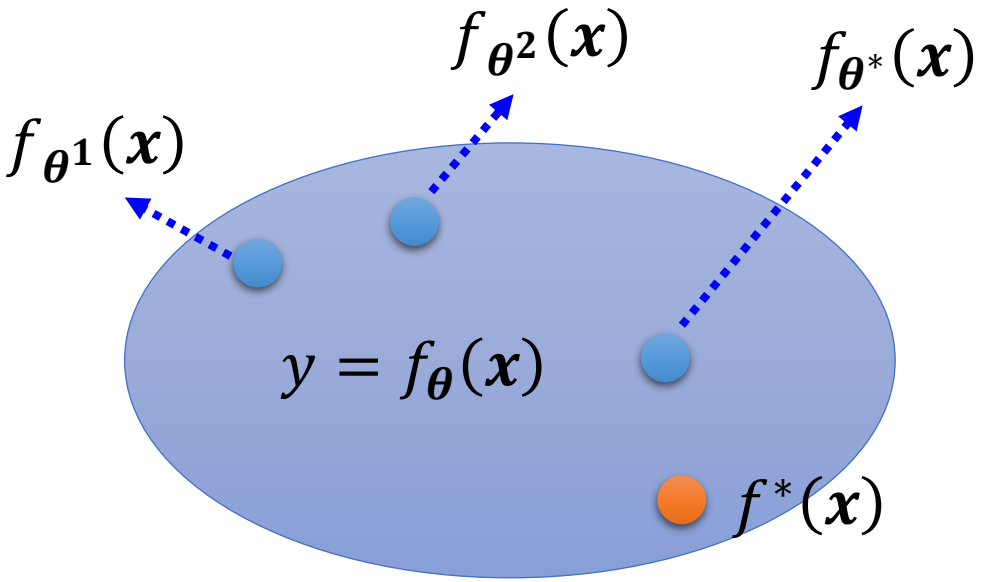
$$y = b + \sum_i c_i \, sigmoid\left(b_i + \sum_j w_{ij} x_j\right)$$


General Guide
loss on training data
large
small
model bias
optimization
loss on testing data
make your model complex
Next Lecture
large
small
overfitting
mismatch
more training data (not in HWs)
data augmentation
Not in HWs, except HW 11
make your model simpler
trade-off
Split your training data into training set and validation set for model selection

Optimization Issue

- Large loss not always imply model bias. There is another possibility …





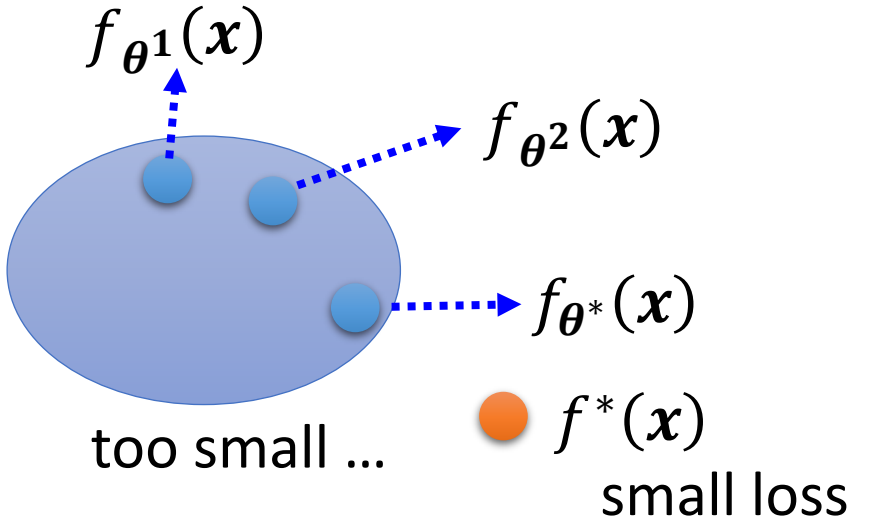
A needle is in a haystack …

… Just cannot find it.

Model Bias

find a needle in a haystack …

… but there is no needle

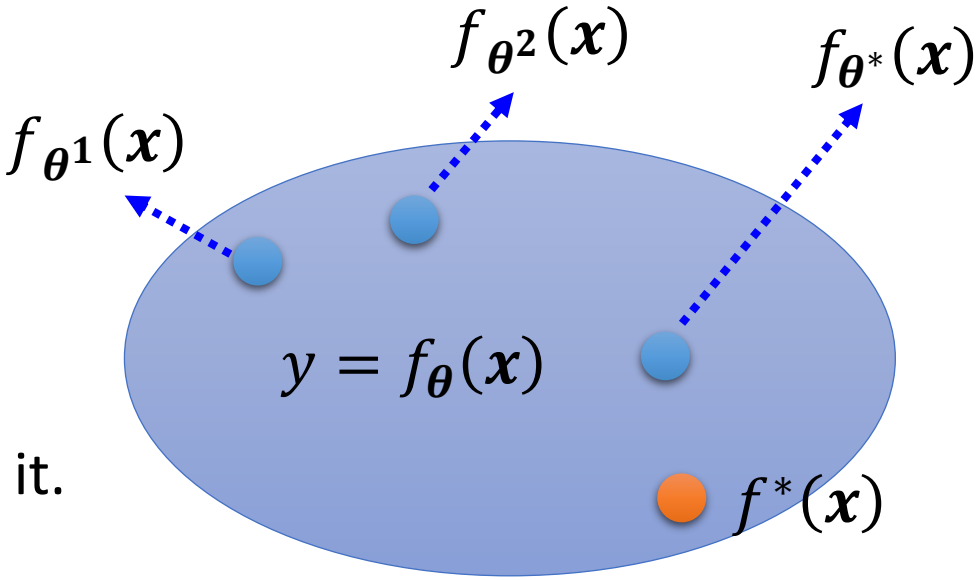


Which one???

Optimization Issue

A needle is in a haystack …

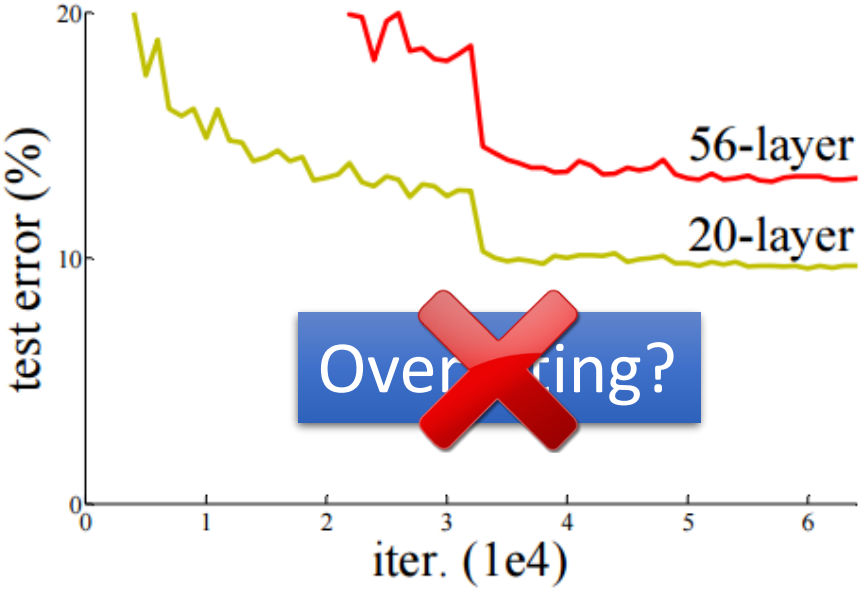
… Just cannot find it.



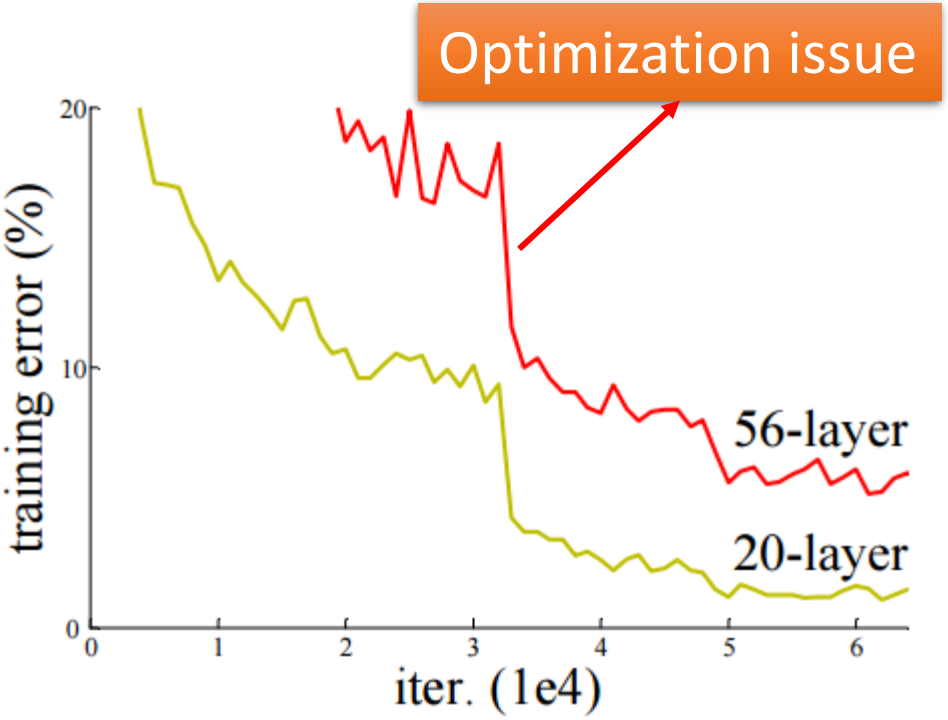
Ref: http://arxiv.org/abs/1512.03385

Model Bias v.s. Optimization Issue

- Gaining the insights from comparison



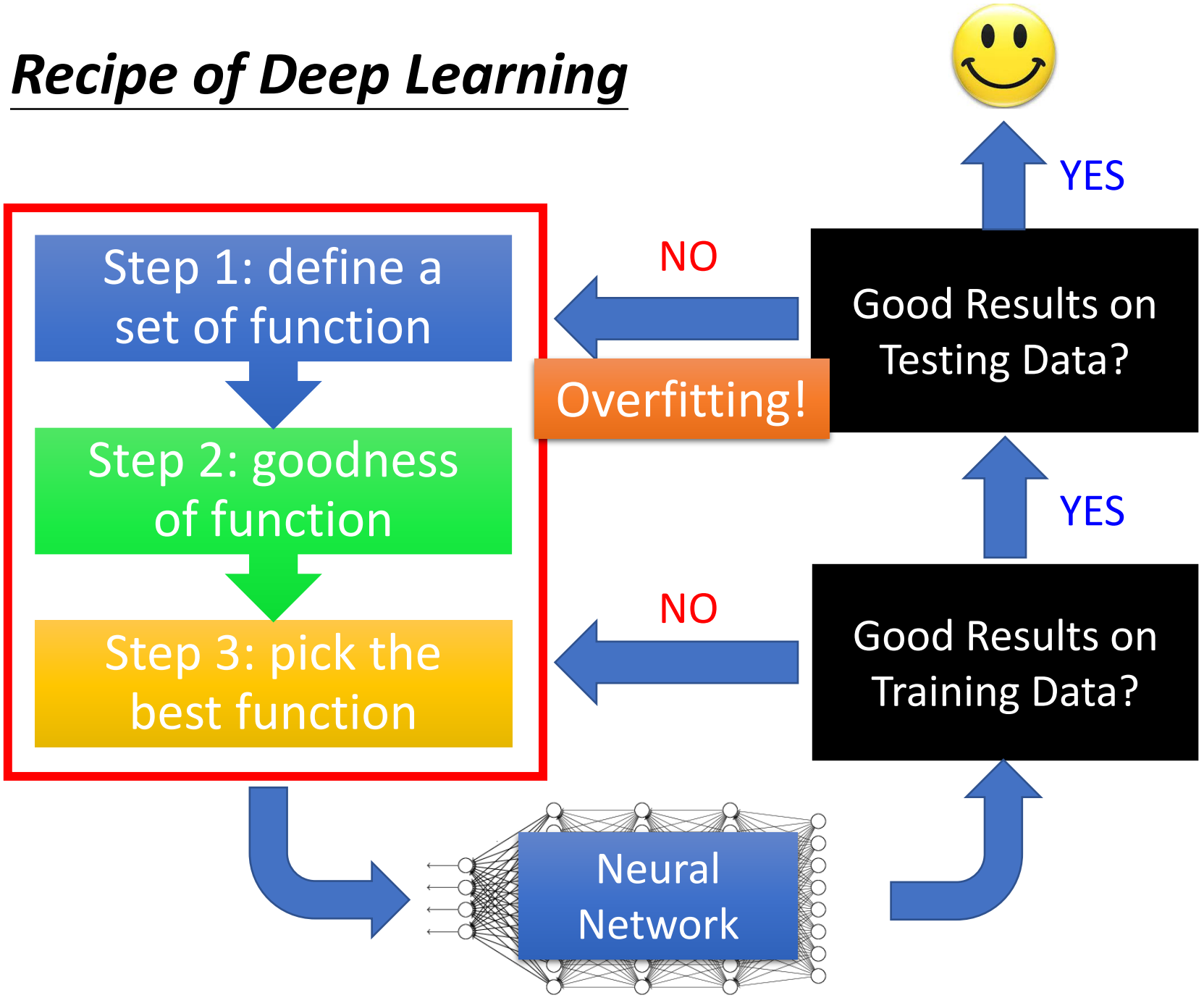
Testing Data



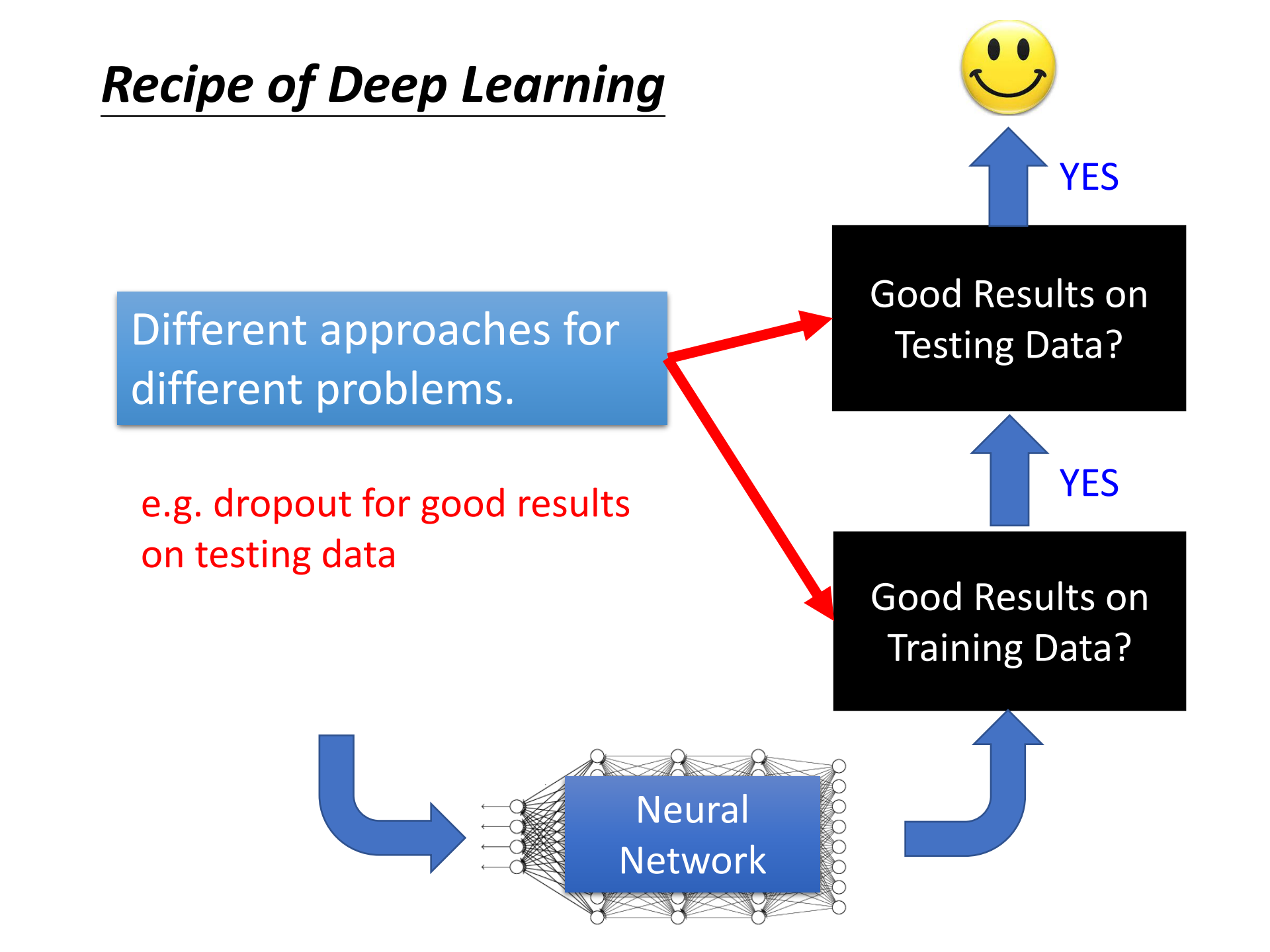
Training Data

Do not always blame Overfitting ！

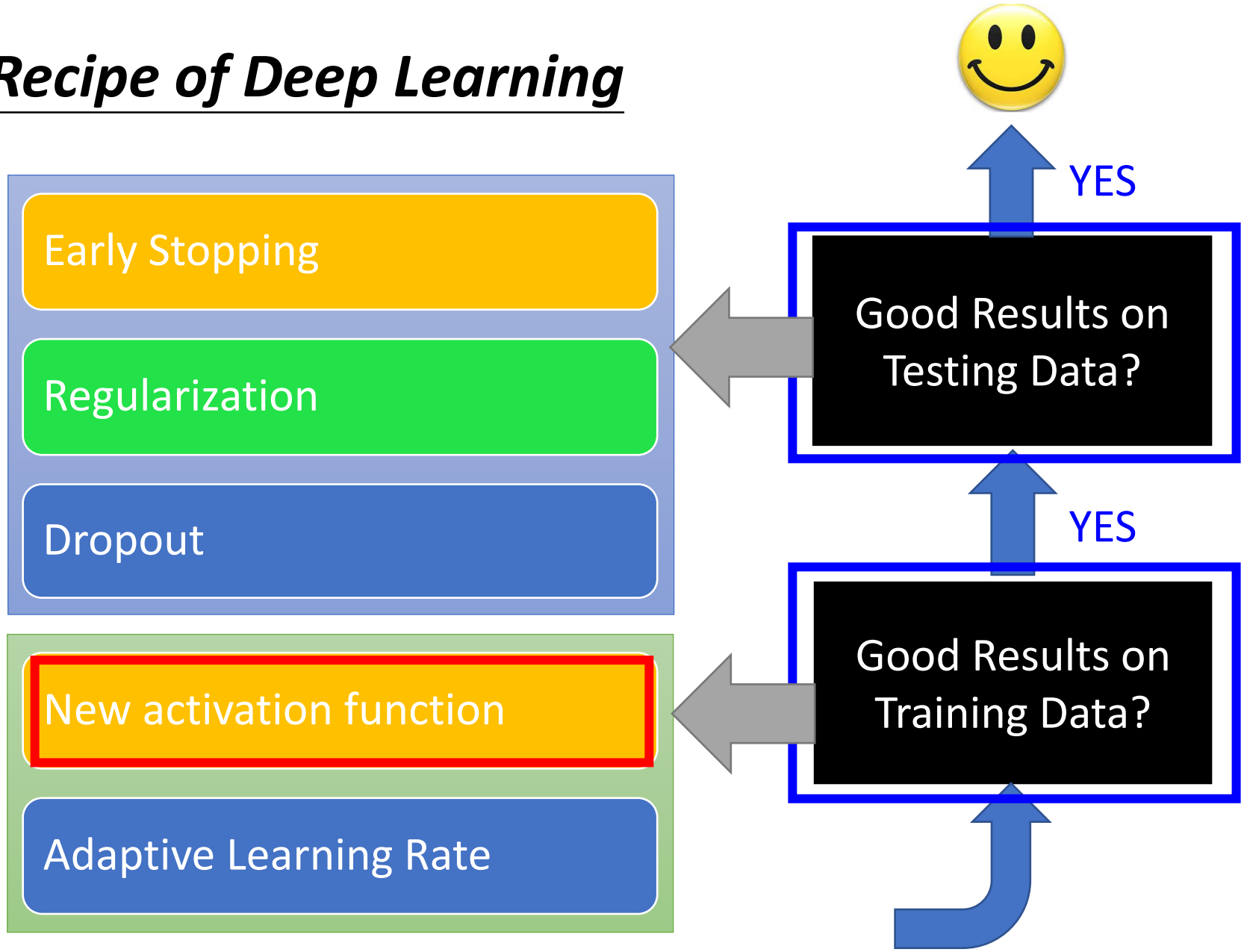
Recipe of Deep Learning



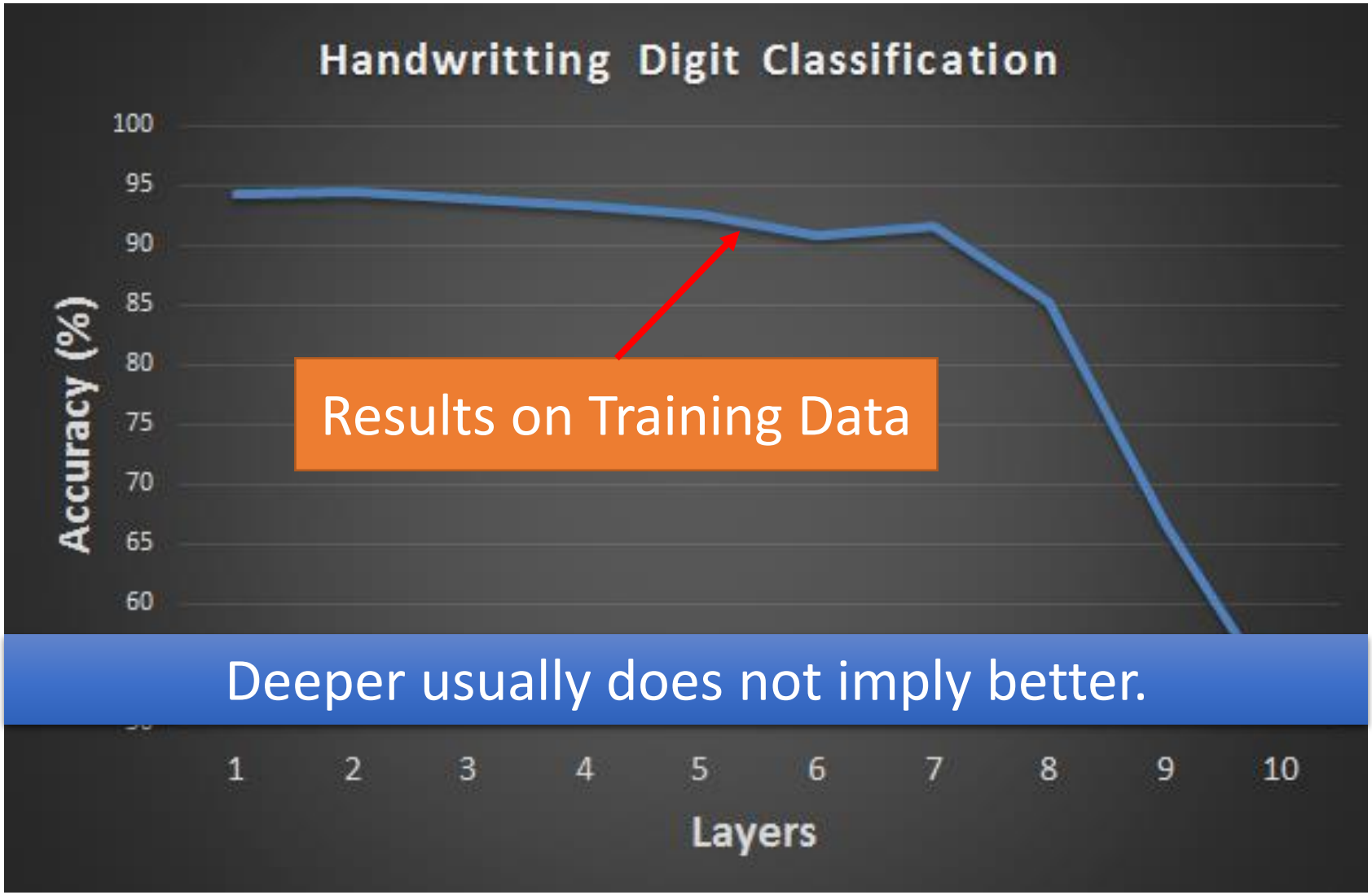
Recipe of Deep Learning



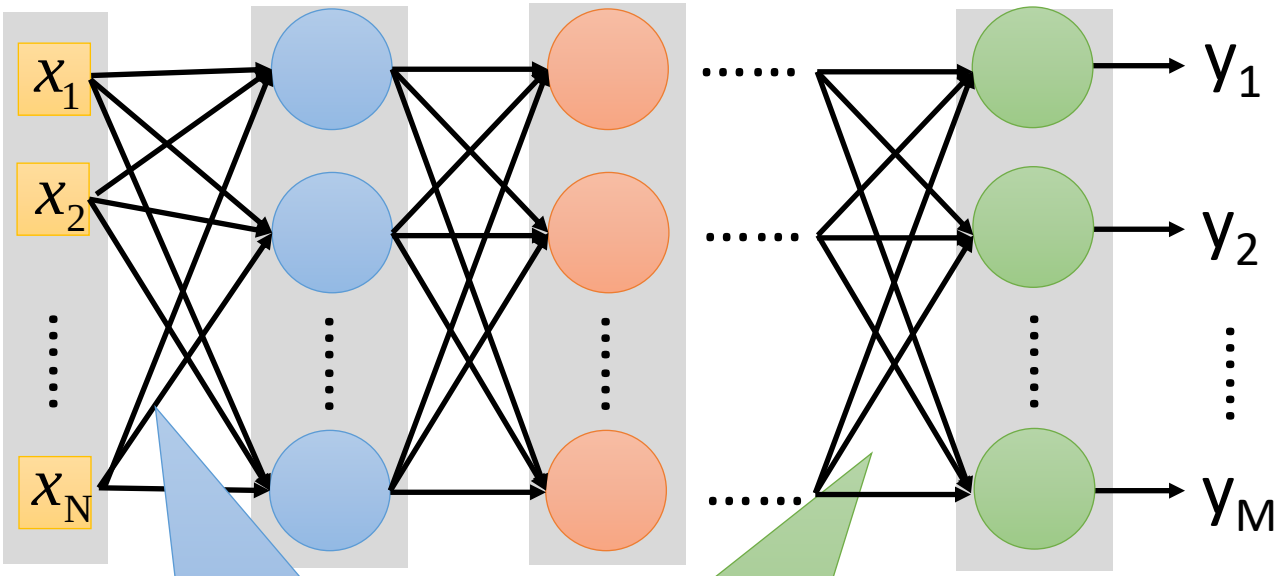
Recipe of Deep Learning



Hard to get the power of Deep …



Vanishing Gradient Problem



Smaller gradients

Learn very slow

Almost random

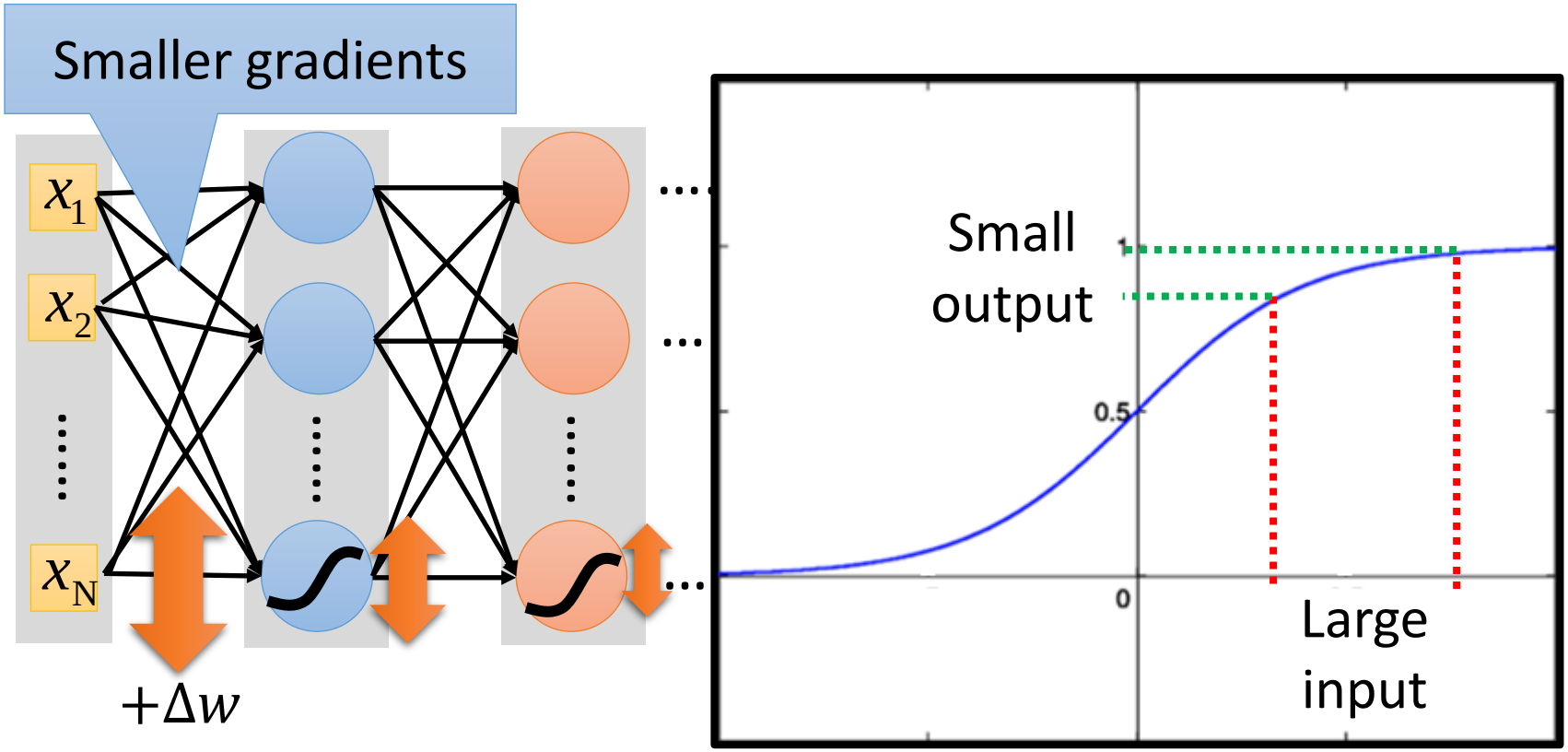
Larger gradients

Learn very fast

Already converge

based on random!?

Vanishing Gradient Problem

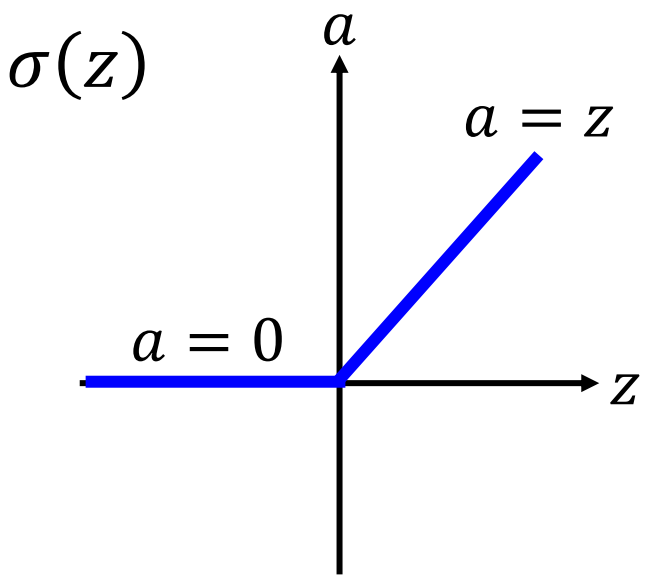


Intuitive way to compute the derivatives …

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

ReLU

- Rectified Linear Unit (ReLU)

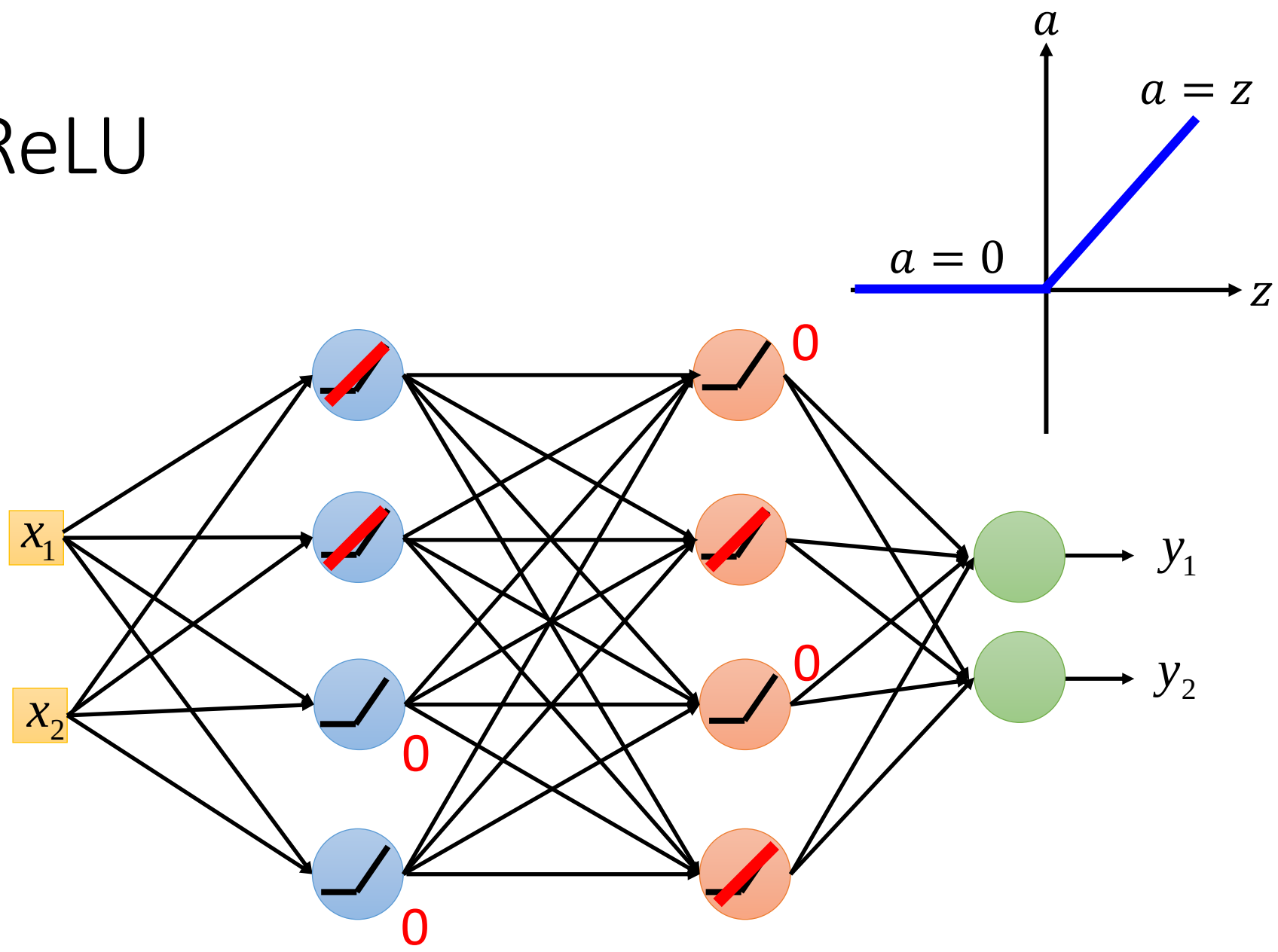


[Xavier Glorot, AISTATS'11]
[Andrew L. Maas, ICML'13]
[Kaiming He, arXiv'15]

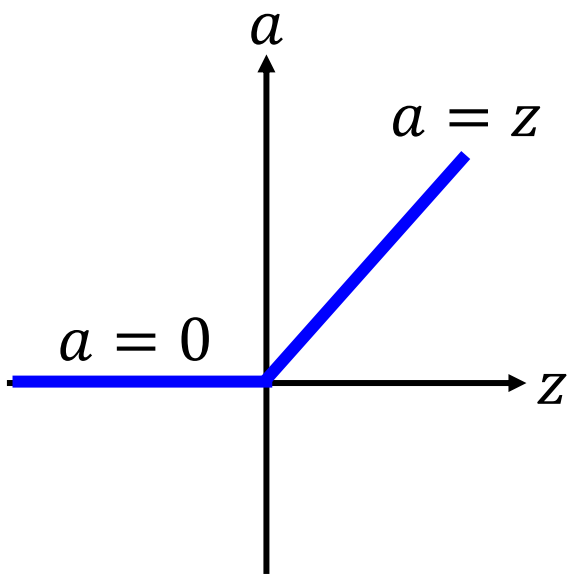
Reason:

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

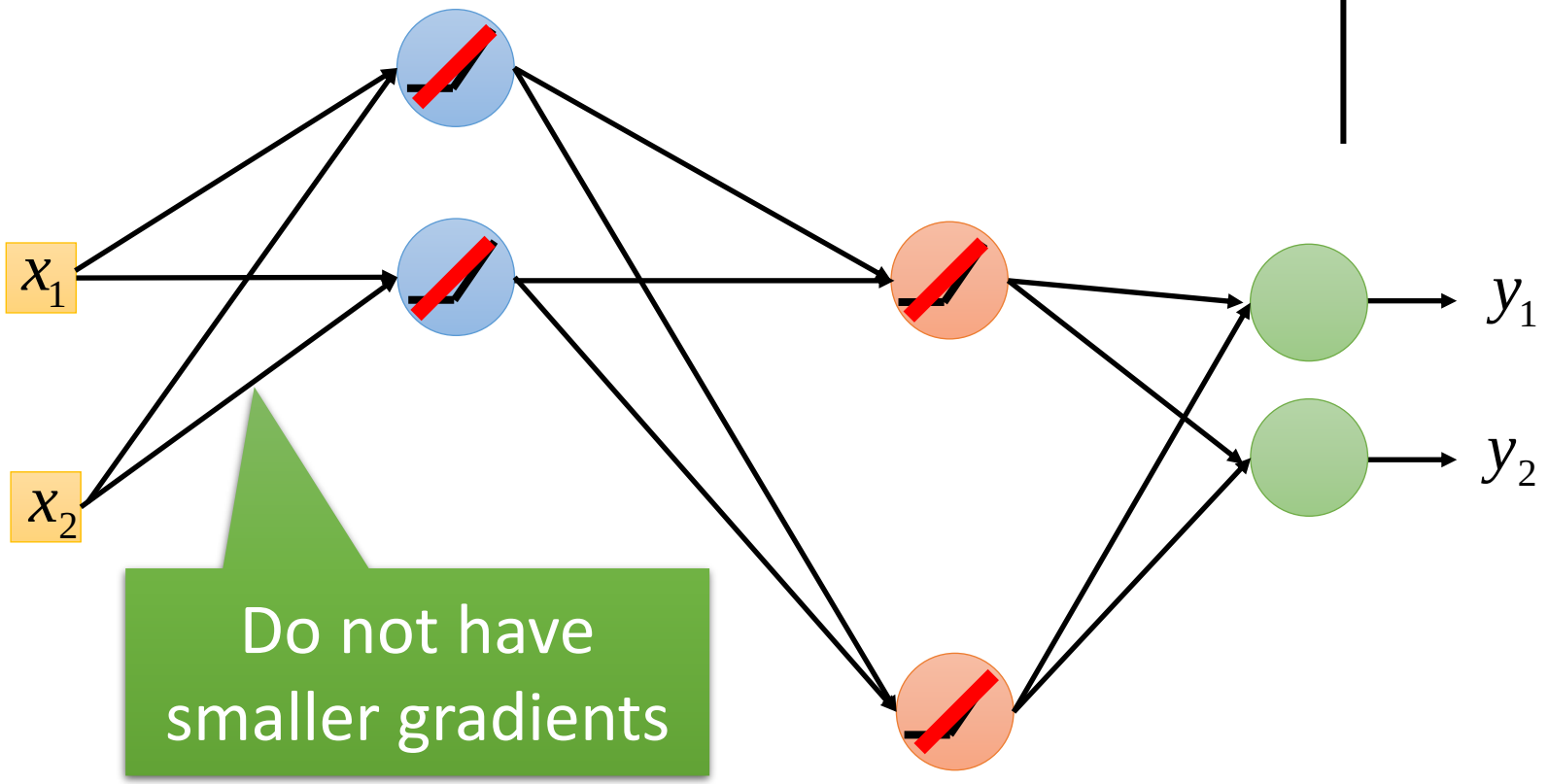
ReLU



ReLU



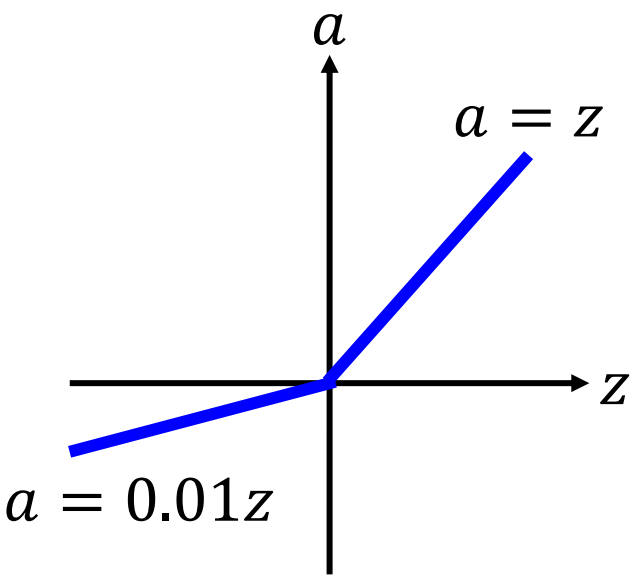
A Thinner linear network



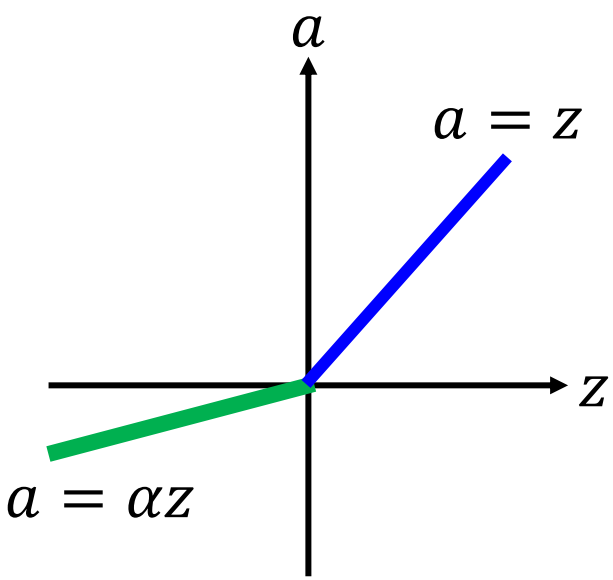
Do not have smaller gradients

ReLU - variant

Leaky ReLU



Parametric ReLU

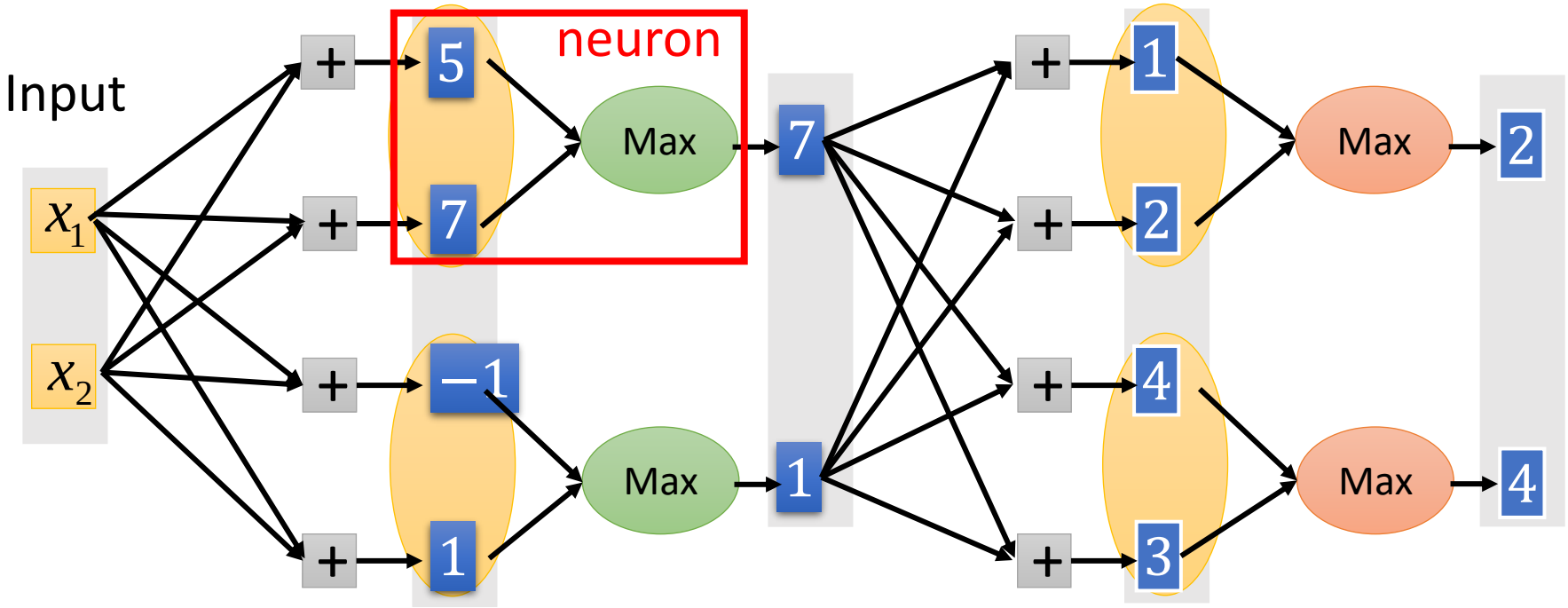


α also learned by gradient descent

Maxout

ReLU is a special cases of Maxout

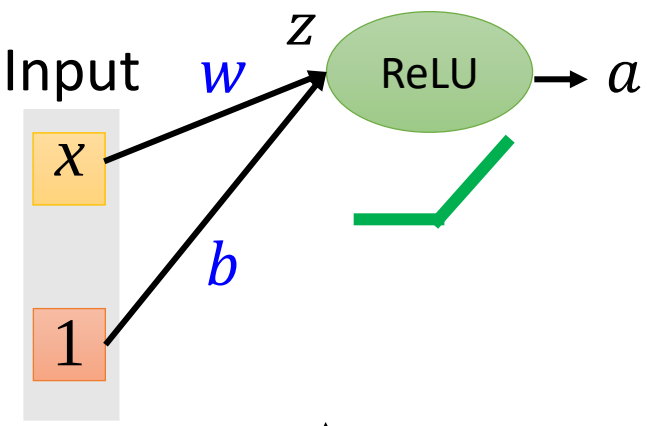
- Learnable activation function [Ian J. Goodfellow, ICML'13]

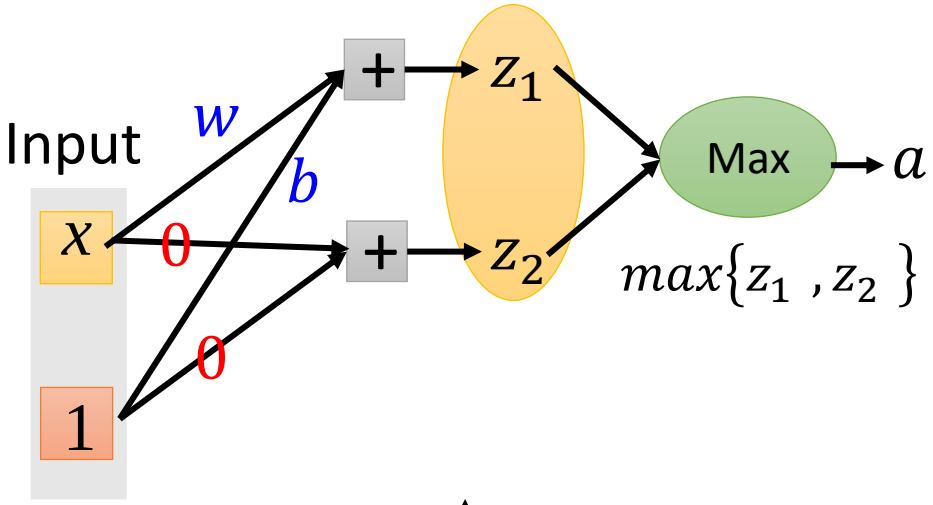


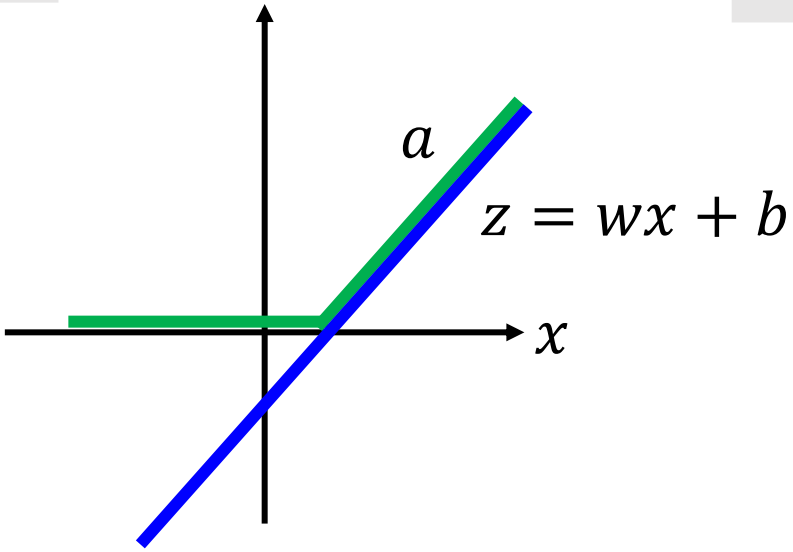
You can have more than 2 elements in a group.

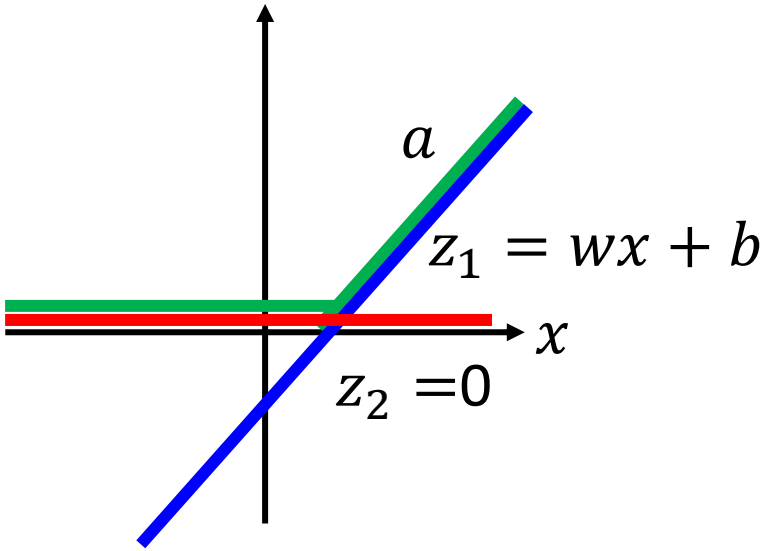
Maxout

ReLU is a special cases of Maxout



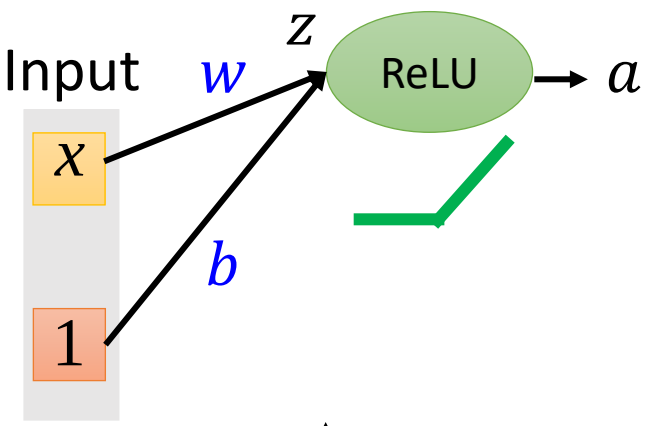


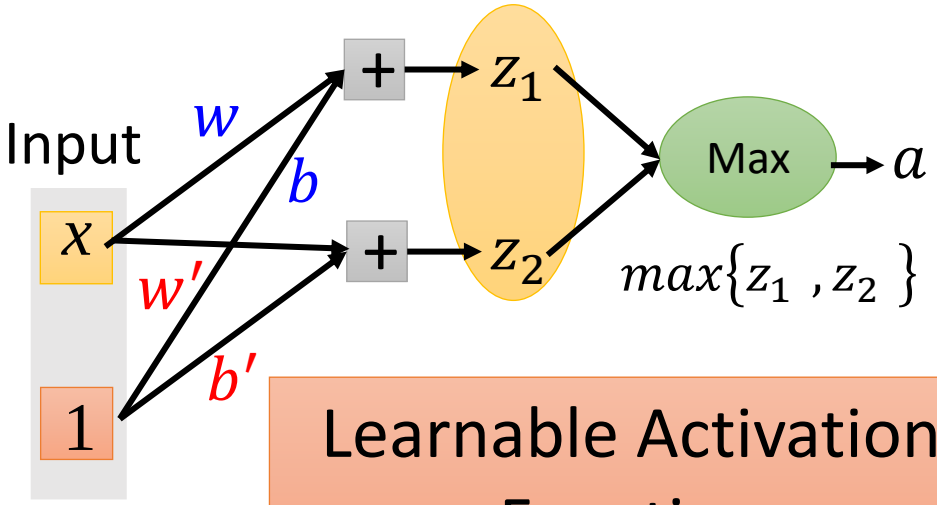




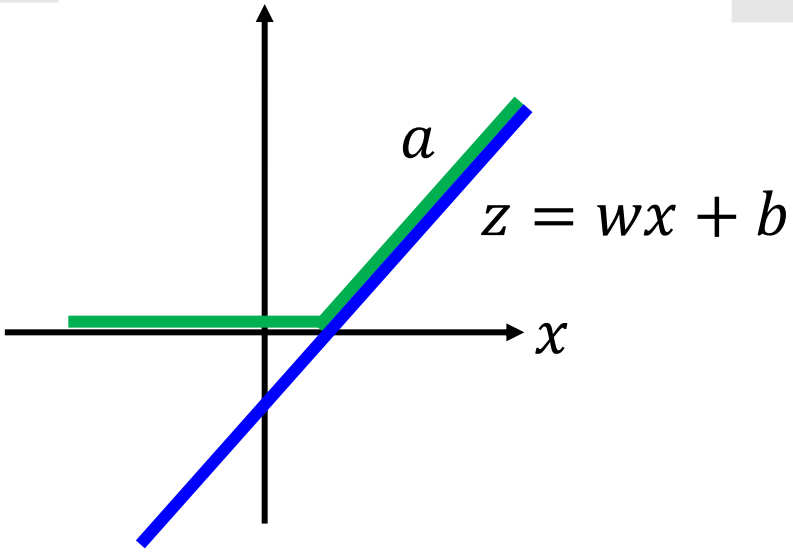
Maxout

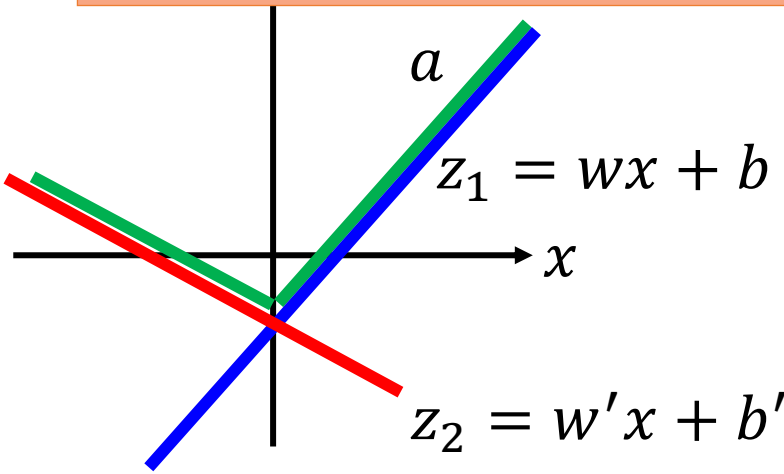
More than ReLU





Learnable Activation Function

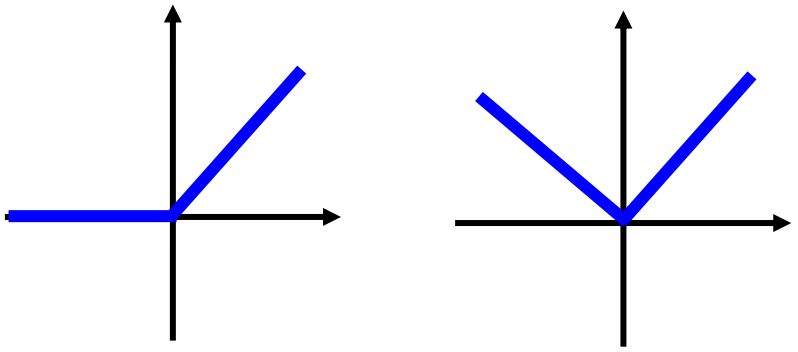




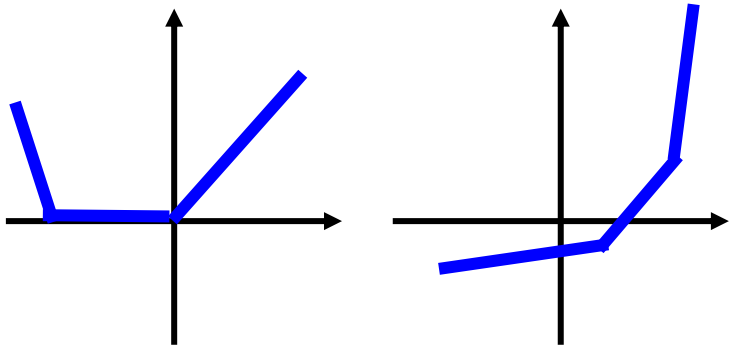
Maxout

- 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

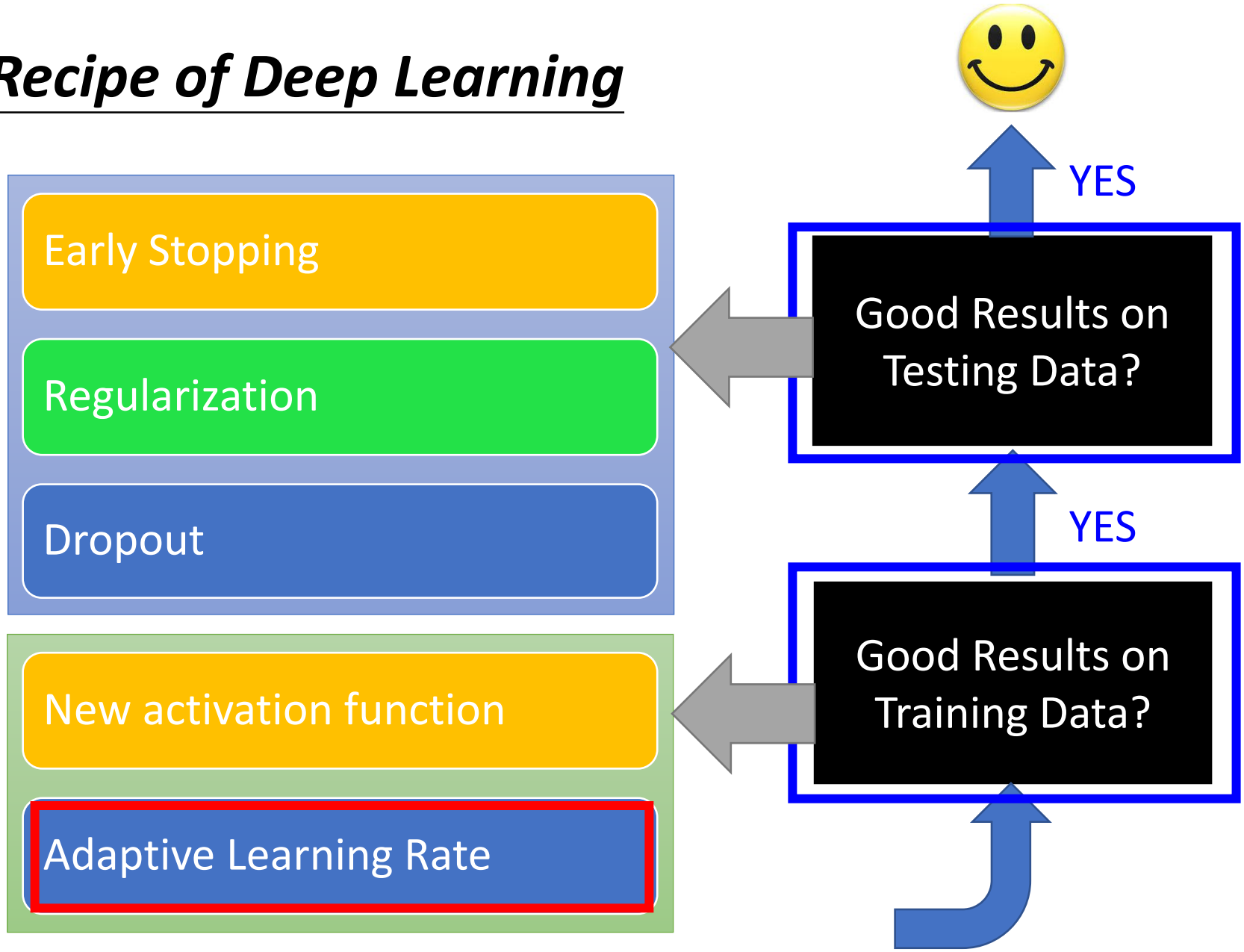
2 elements in a group



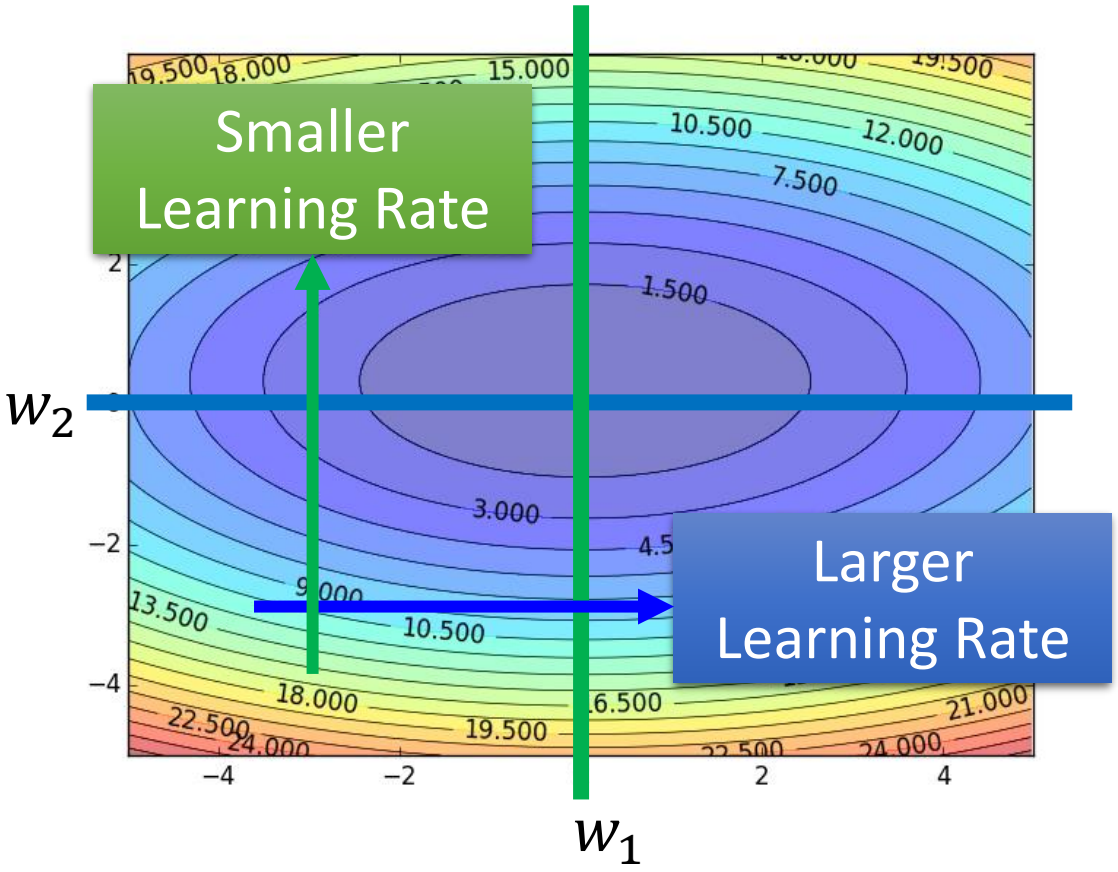
3 elements in a group



Recipe of Deep Learning



Review



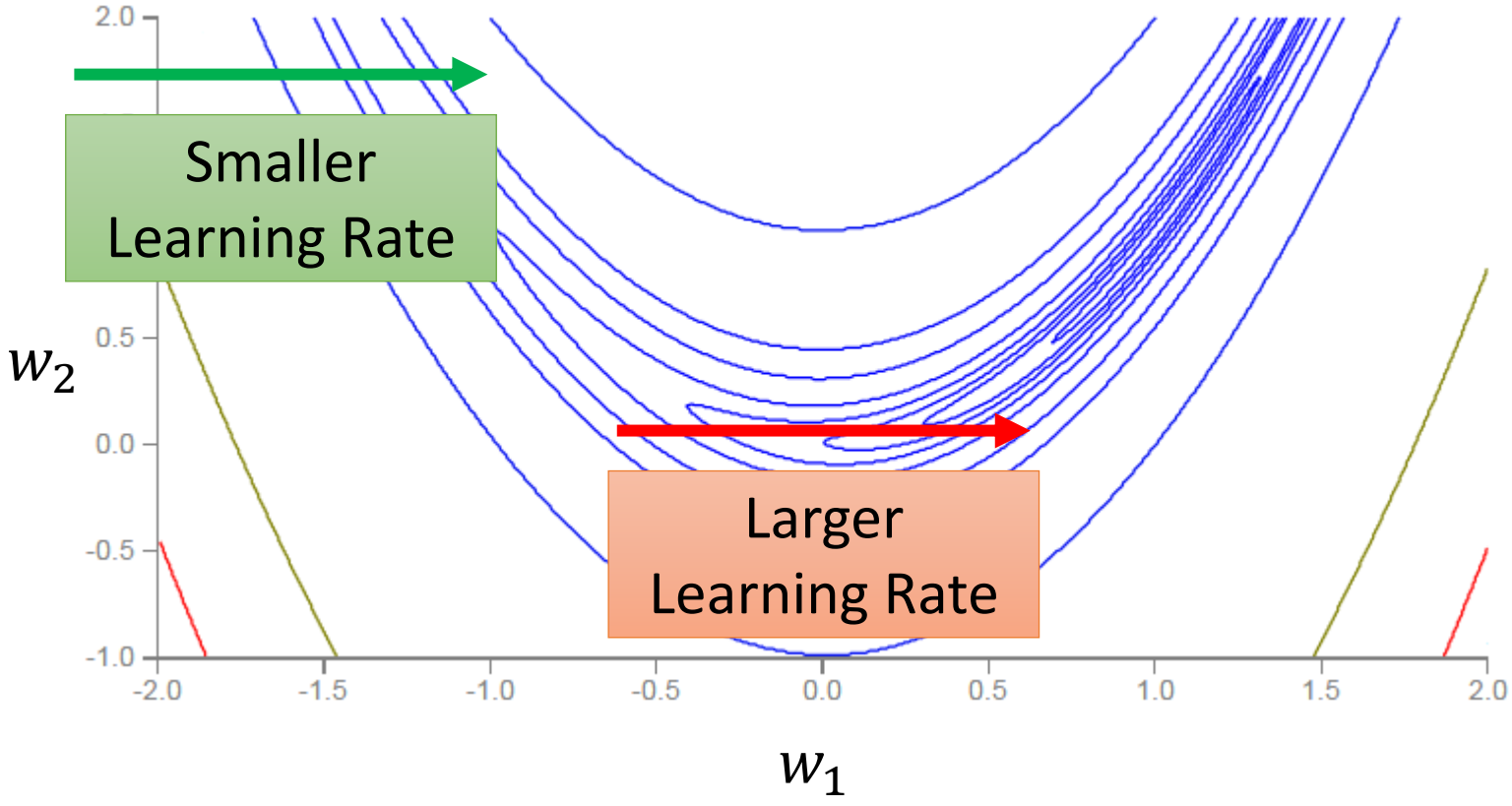
Adagrad

$$w^{t+1} \leftarrow w^t - \frac{\eta}{\sqrt{\sum_{i=0}^{t}(g^i)^2}} g^t$$

Use first derivative to estimate second derivative

RMSProp

Error Surface can be very complex when training NN.



RMSProp

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

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

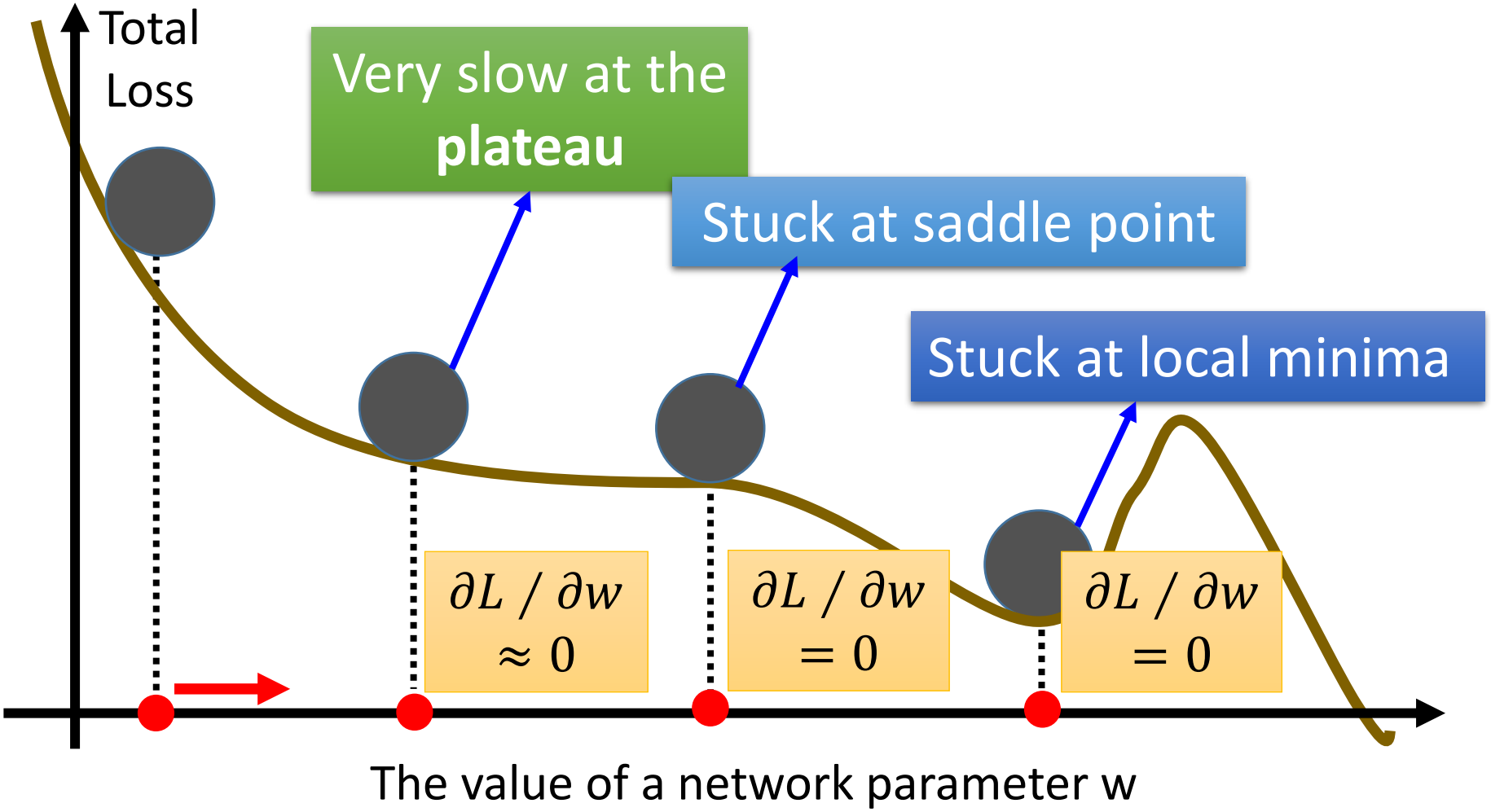
$$w^3 \leftarrow w^2 - \frac{\eta}{\sigma^2} g^2 \qquad \sigma^2 = \sqrt{\alpha(\sigma^1)^2 + (1-\alpha)(g^2)^2}$$

$$\vdots \qquad\qquad \vdots$$

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

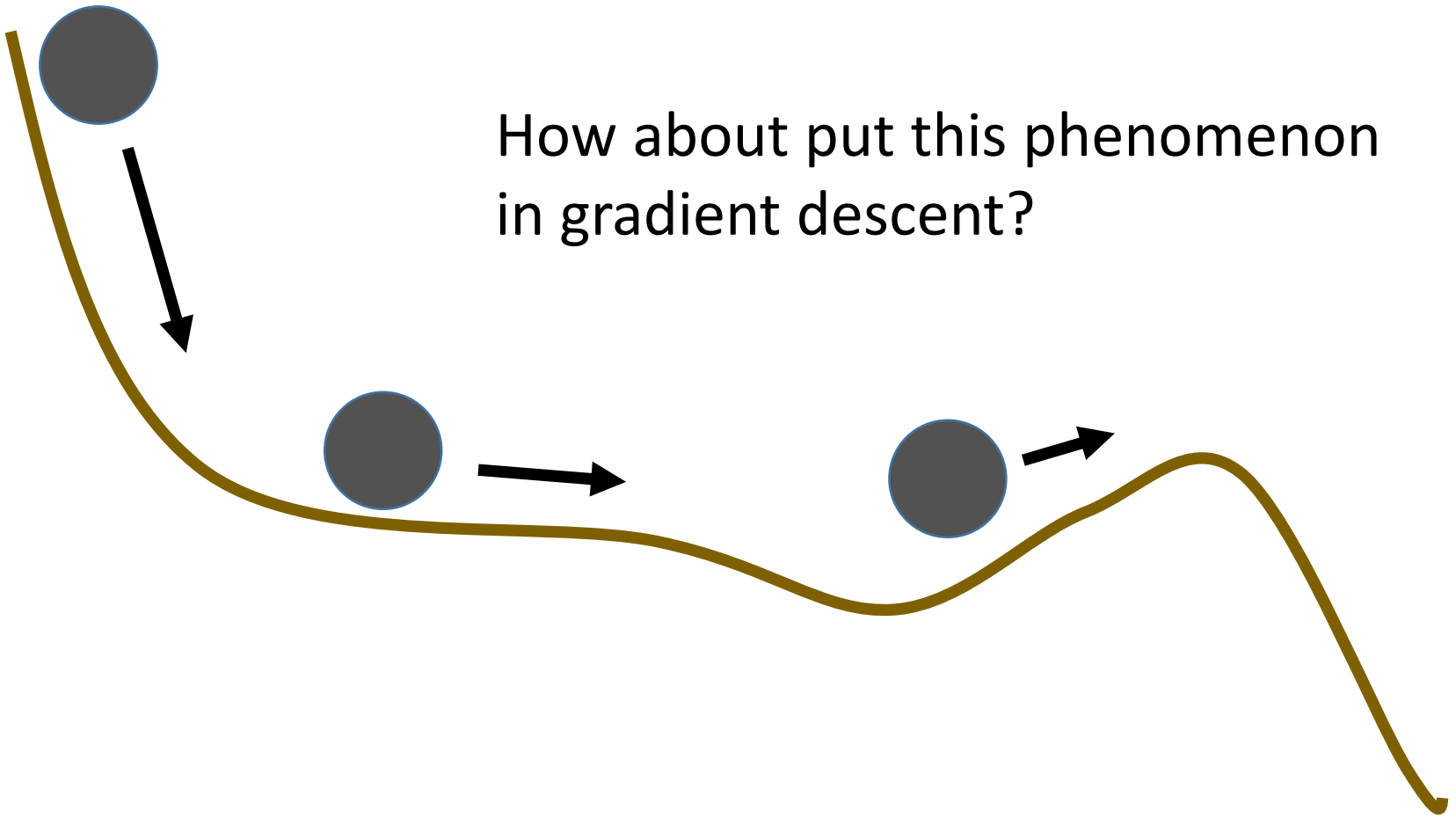
Root Mean Square of the gradients with previous gradients being decayed

Hard to find optimal network parameters



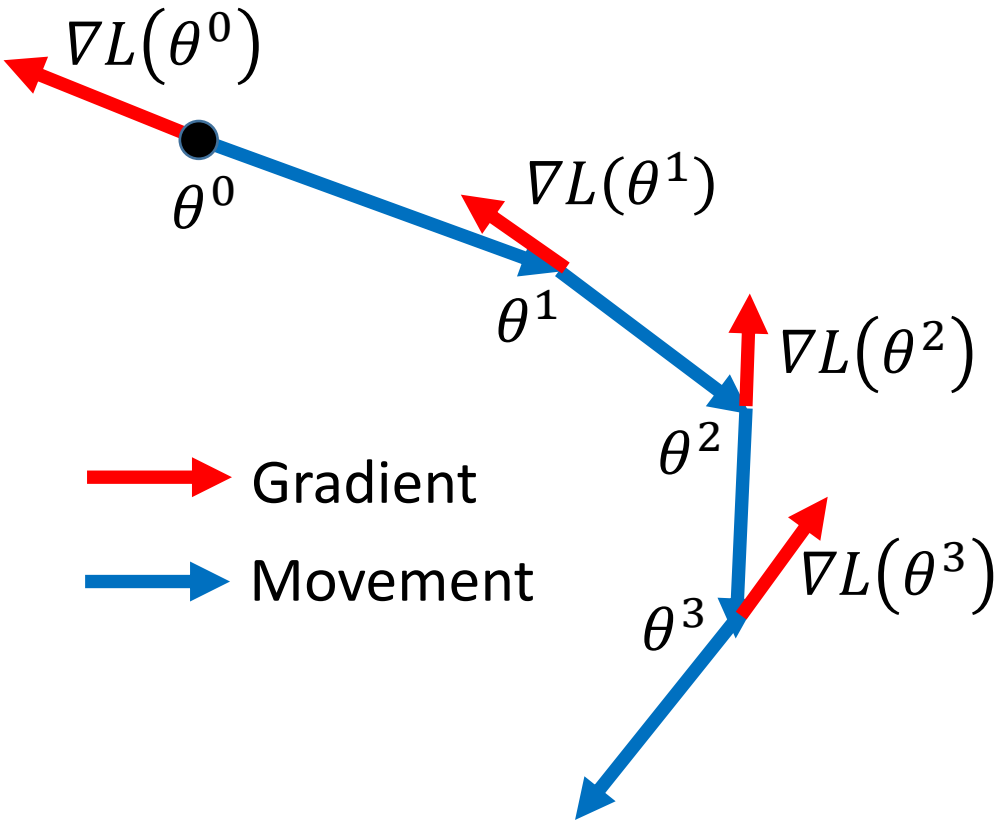
In physical world ……

- Momentum



How about put this phenomenon in gradient descent?

Review: Vanilla Gradient Descent



Start at position θ^0

Compute gradient at θ^0

Move to $\theta^1 = \theta^0 - \eta\nabla L(\theta^0)$

Compute gradient at θ^1

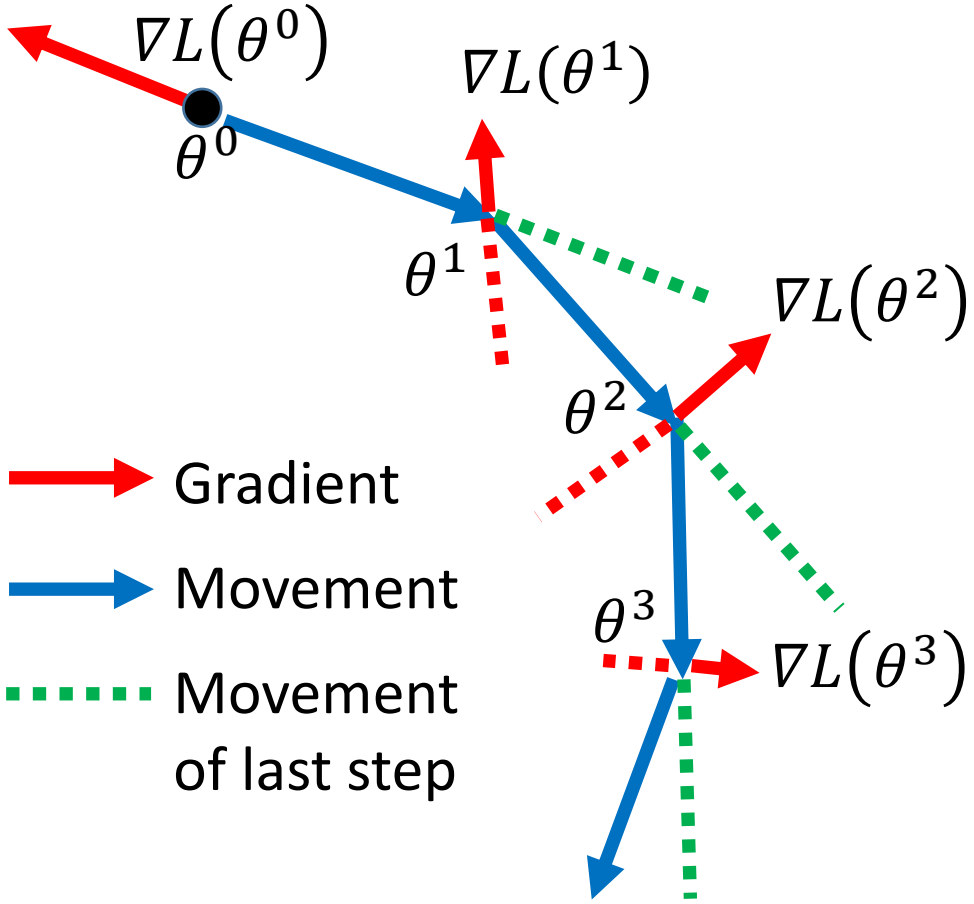
Move to $\theta^2 = \theta^1 - \eta\nabla L(\theta^1)$

⋮

Stop until $\nabla L(\theta^t) \approx 0$

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.

Momentum

Movement: movement of last step minus gradient at present

v^i is actually the weighted sum of all the previous gradient: $\nabla L(\theta^0), \nabla L(\theta^1), \ldots \nabla L(\theta^{i-1})$

$v^0 = 0$

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

$v^2 = -\lambda \eta \nabla L(\theta^0) - \eta \nabla L(\theta^1)$

⋮

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

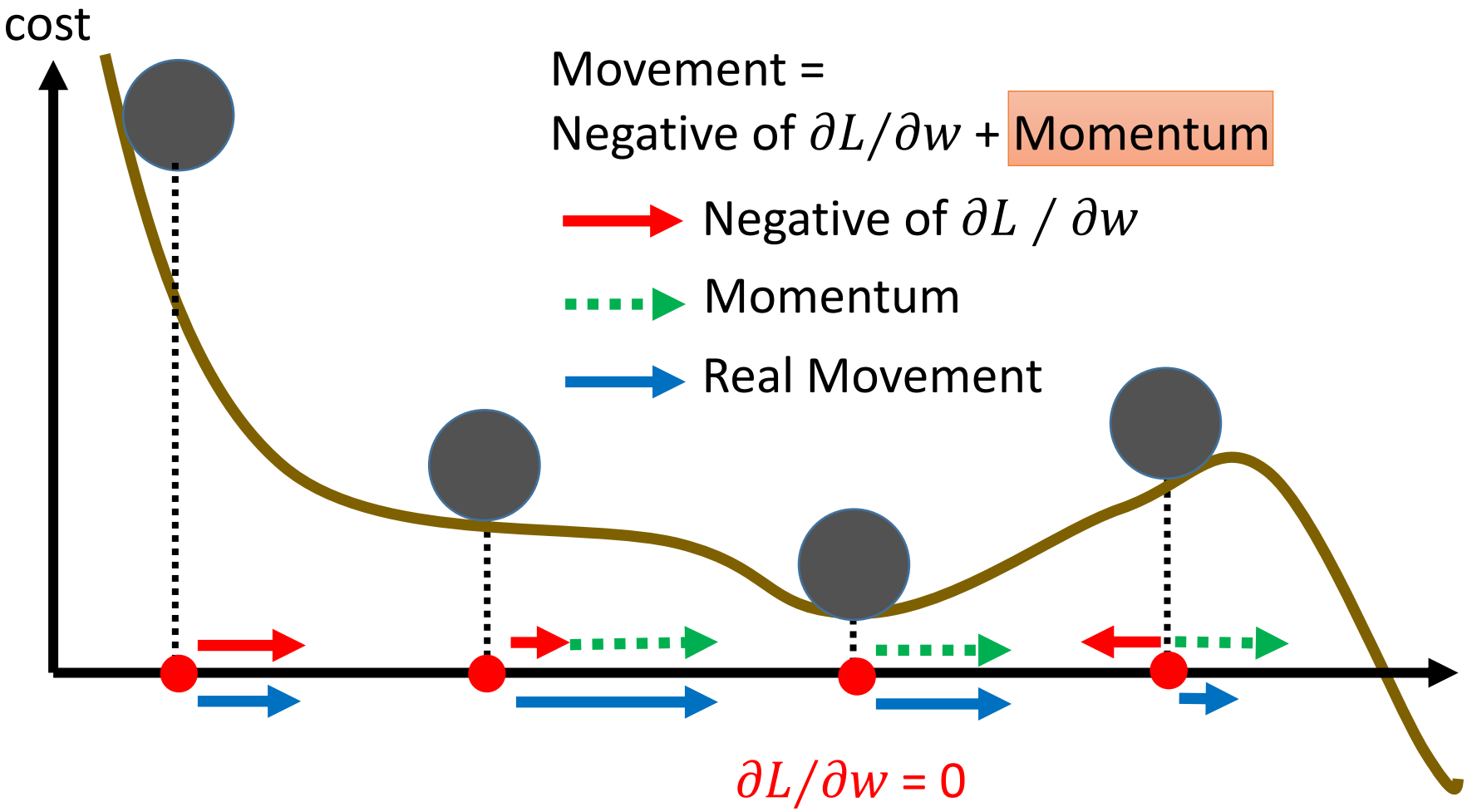
Movement $v^2 = \lambda v^1 - \eta \nabla L(\theta^1)$

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

Movement not just based on gradient, but previous movement

Momentum

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



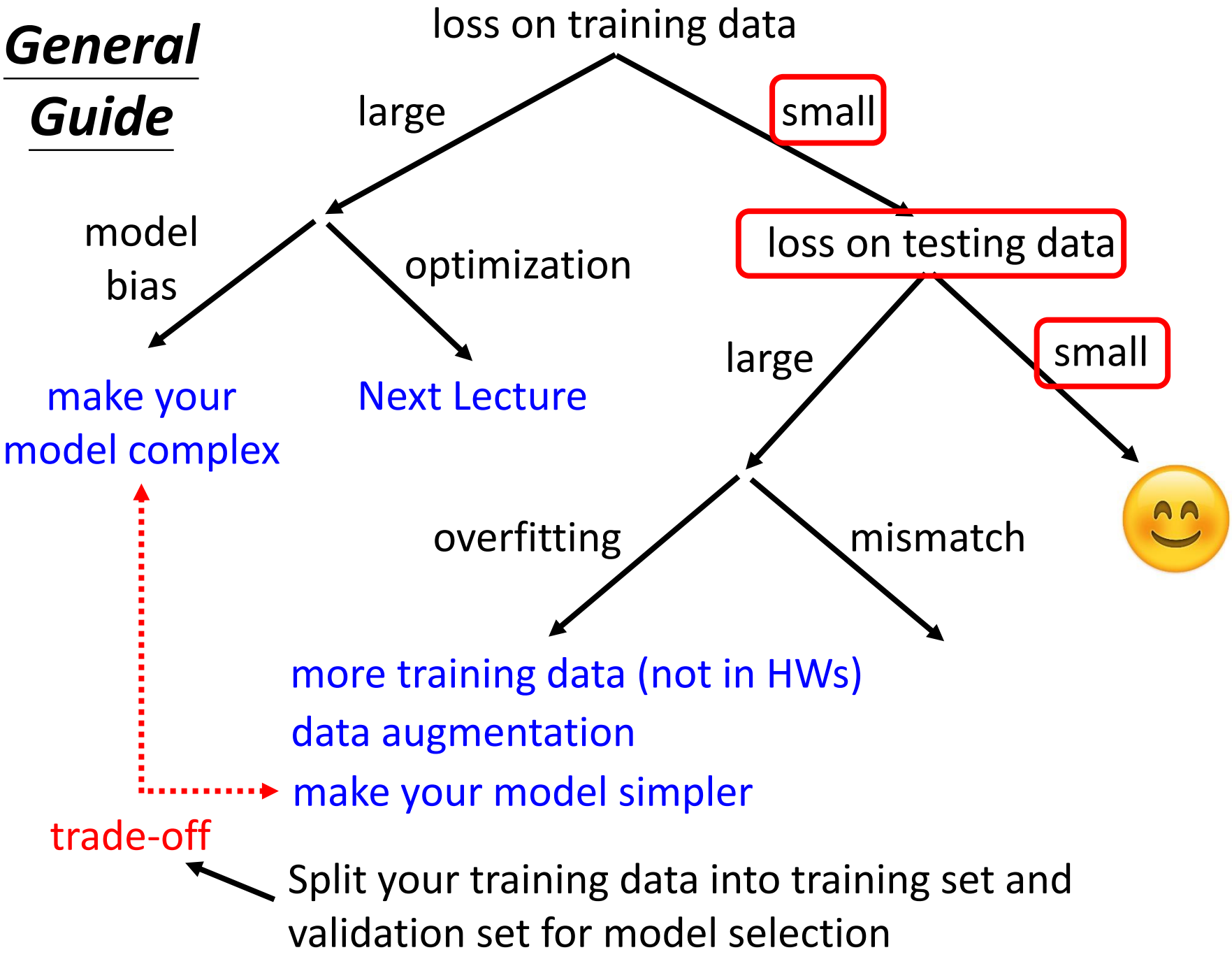
Adam

RMSProp + Momentum

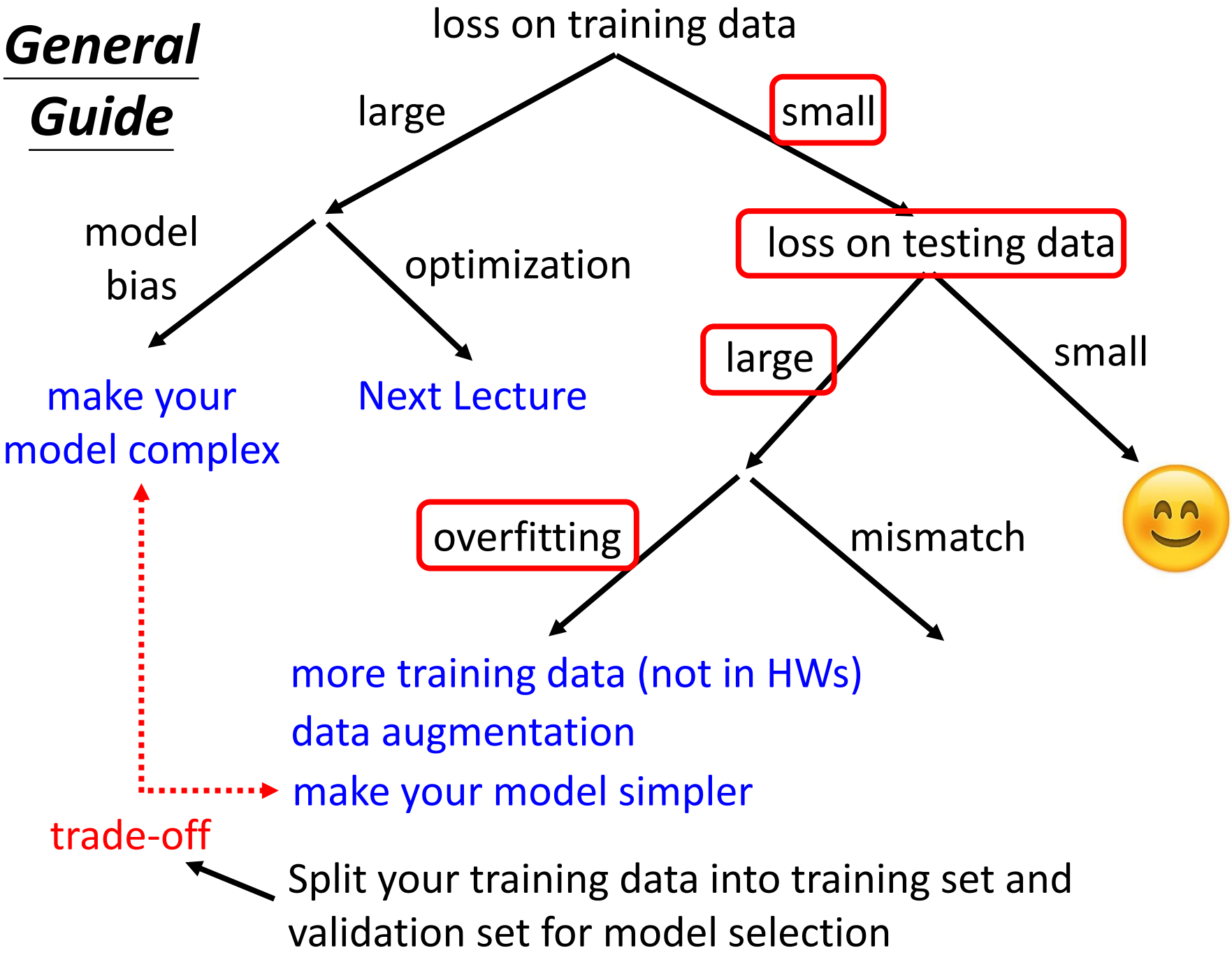
Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha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

General Guide



General Guide



Overfitting

- Small loss on training data, large loss on testing data. Why?

An extreme example

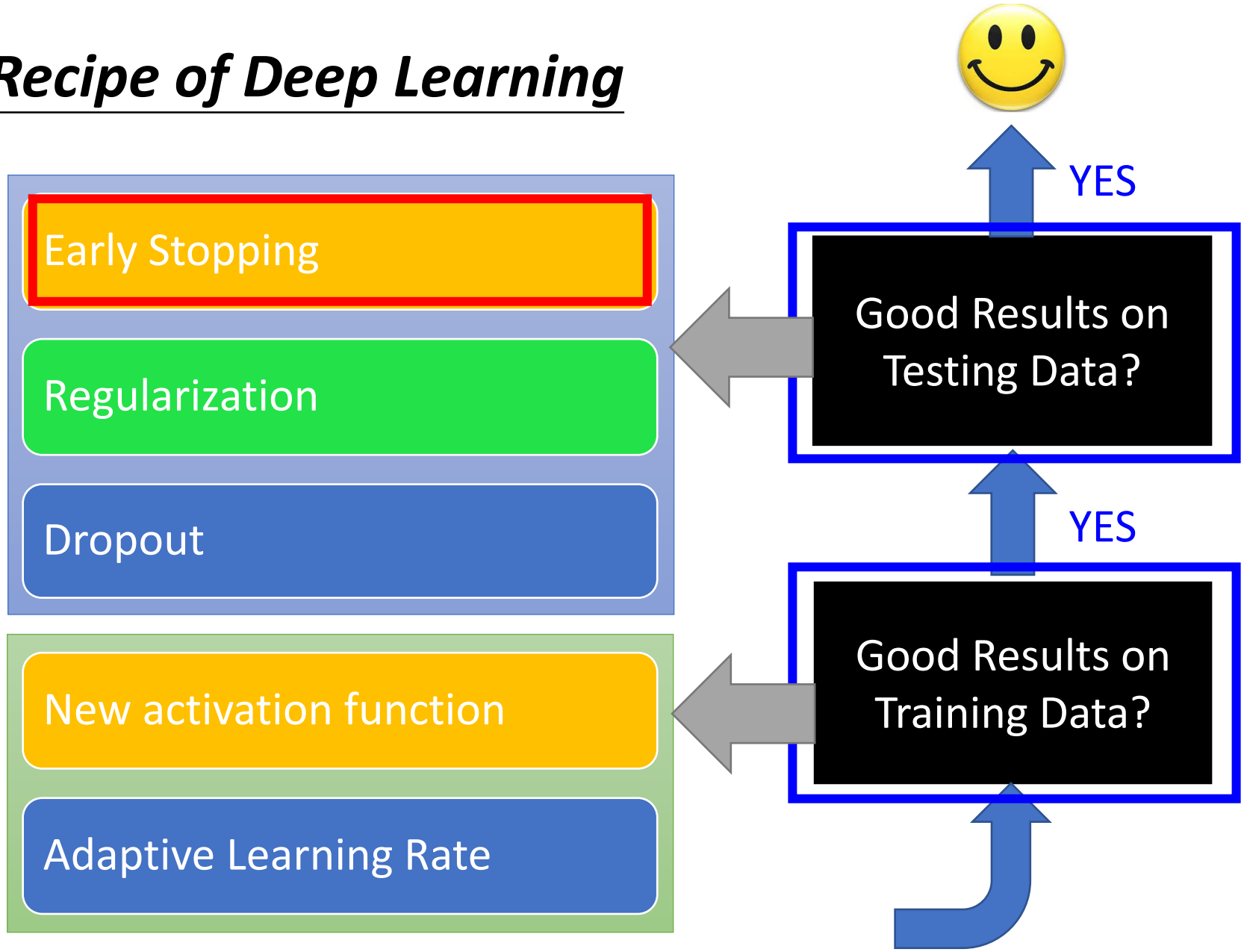
Training data: $\{(\boldsymbol{x^1}, \hat{y}^1), (\boldsymbol{x^2}, \hat{y}^2), \ldots, (\boldsymbol{x^N}, \hat{y}^N)\}$

$$f(\boldsymbol{x}) = \begin{cases} \hat{y}^i & \exists \boldsymbol{x^i} = \boldsymbol{x} \\ random & otherwise \end{cases}$$

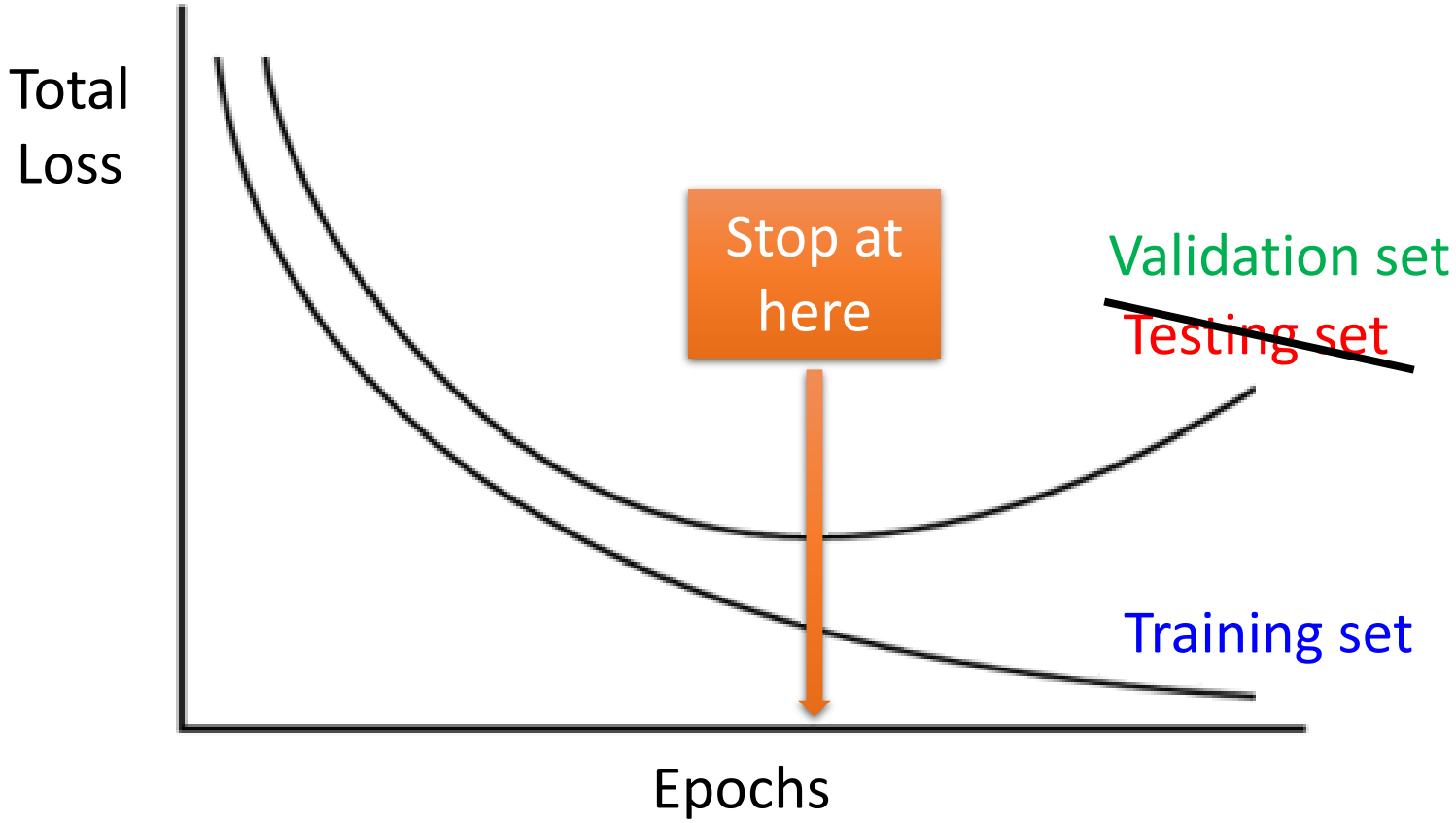
Less than useless …

This function obtains **zero training loss**, but **large testing loss**.

Recipe of Deep Learning

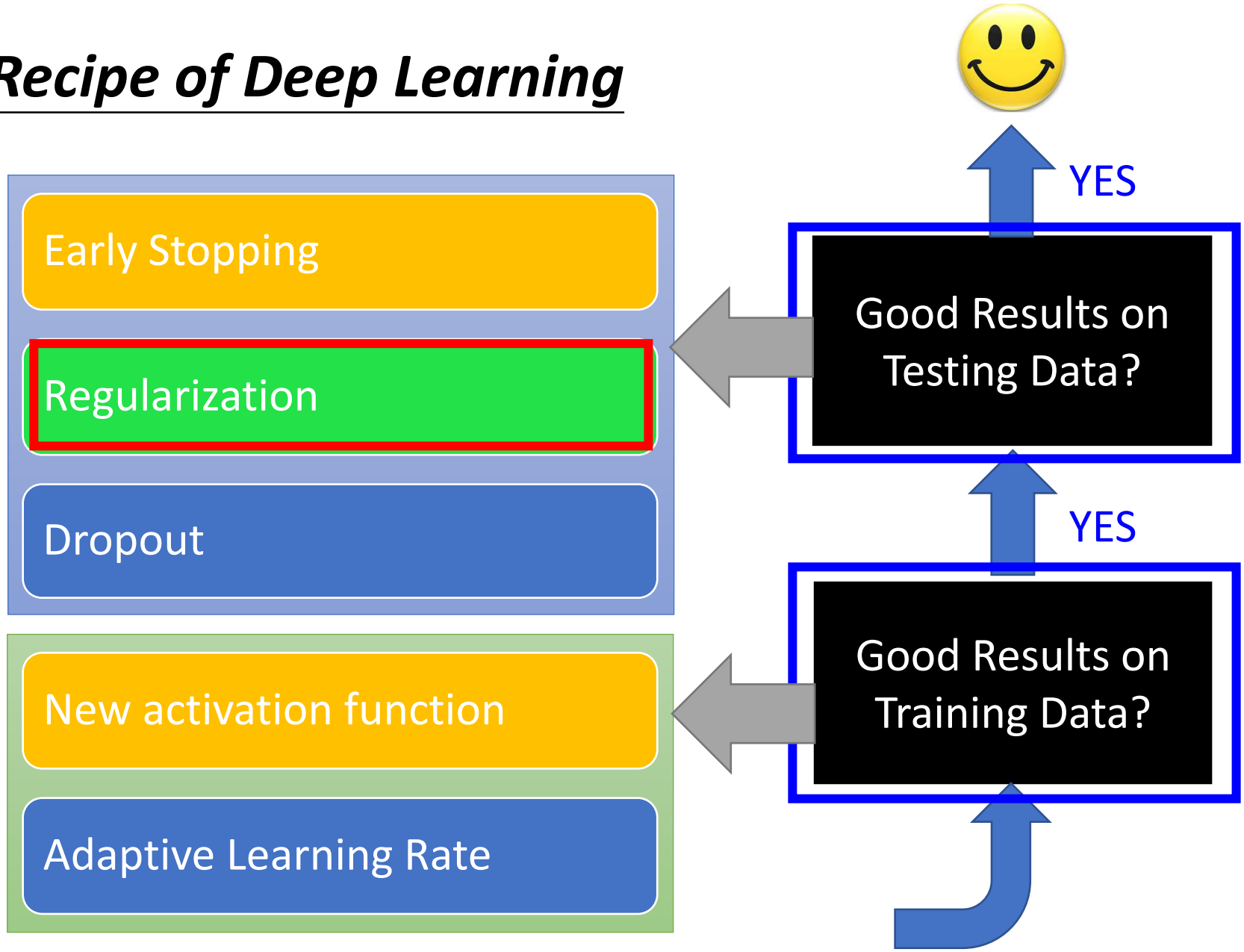


Early Stopping



Keras: http://keras.io/getting-started/faq/#how-can-i-interrupt-training-when-the-validation-loss-isnt-decreasing-anymore

Recipe of Deep Learning



Regularization

- New loss function to be minimized
  - Find a set of weight not only minimizing original cost but also close to zero

$$L'(\theta) = L(\theta) + \lambda \frac{1}{2} \|\theta\|_2$$

→ Regularization term

Original loss
(e.g. minimize square error, cross entropy …)

$$\theta = \{w_1, w_2, \ldots\}$$

L2 regularization:

$$\|\theta\|_2 = (w_1)^2 + (w_2)^2 + \ldots$$

(usually not consider biases)

Regularization

L2 regularization:

$$\|\theta\|_2 = (w_1)^2 + (w_2)^2 + \dots$$

- New loss function to be minimized

$$L'(\theta) = L(\theta) + \lambda \frac{1}{2}\|\theta\|_2$$

Gradient: $\frac{\partial L'}{\partial w} = \frac{\partial L}{\partial w} + \lambda w$

Update:

$$w^{t+1} \rightarrow w^t - \eta \frac{\partial L'}{\partial w} = w^t - \eta\left(\frac{\partial L}{\partial w} + \lambda w^t\right)$$

$$= (1-\eta\lambda) w^t - \eta \frac{\partial L}{\partial w}$$

Weight Decay

Closer to zero

Regularization

L1 regularization:

$$\|\theta\|_1 = |w_1| + |w_2| + \dots$$

- New loss function to be minimized

$$L'(\theta) = L(\theta) + \lambda \frac{1}{2}\|\theta\|_1 \qquad \frac{\partial L'}{\partial w} = \frac{\partial L}{\partial w} + \lambda \,\mathrm{sgn}(w)$$

Update:

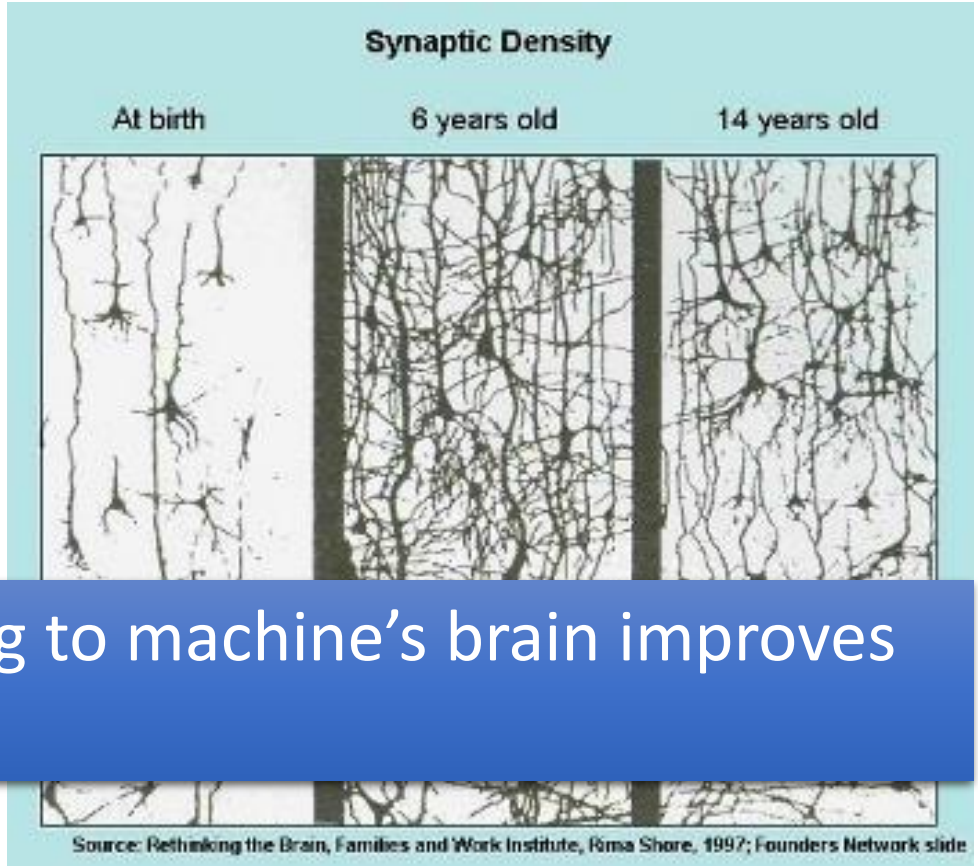
$$w^{t+1} \rightarrow w^t - \eta \frac{\partial L'}{\partial w} = w^t - \eta\left(\frac{\partial L}{\partial w} + \lambda \,\mathrm{sgn}\left(w^t\right)\right)$$

$$= w^t - \eta \frac{\partial L}{\partial w} - \underline{\eta\lambda \,\mathrm{sgn}\left(w^t\right)}$$ Always delete

$$= (1 - \eta\lambda) w^t - \eta \frac{\partial L}{\partial w}$$ …… L2

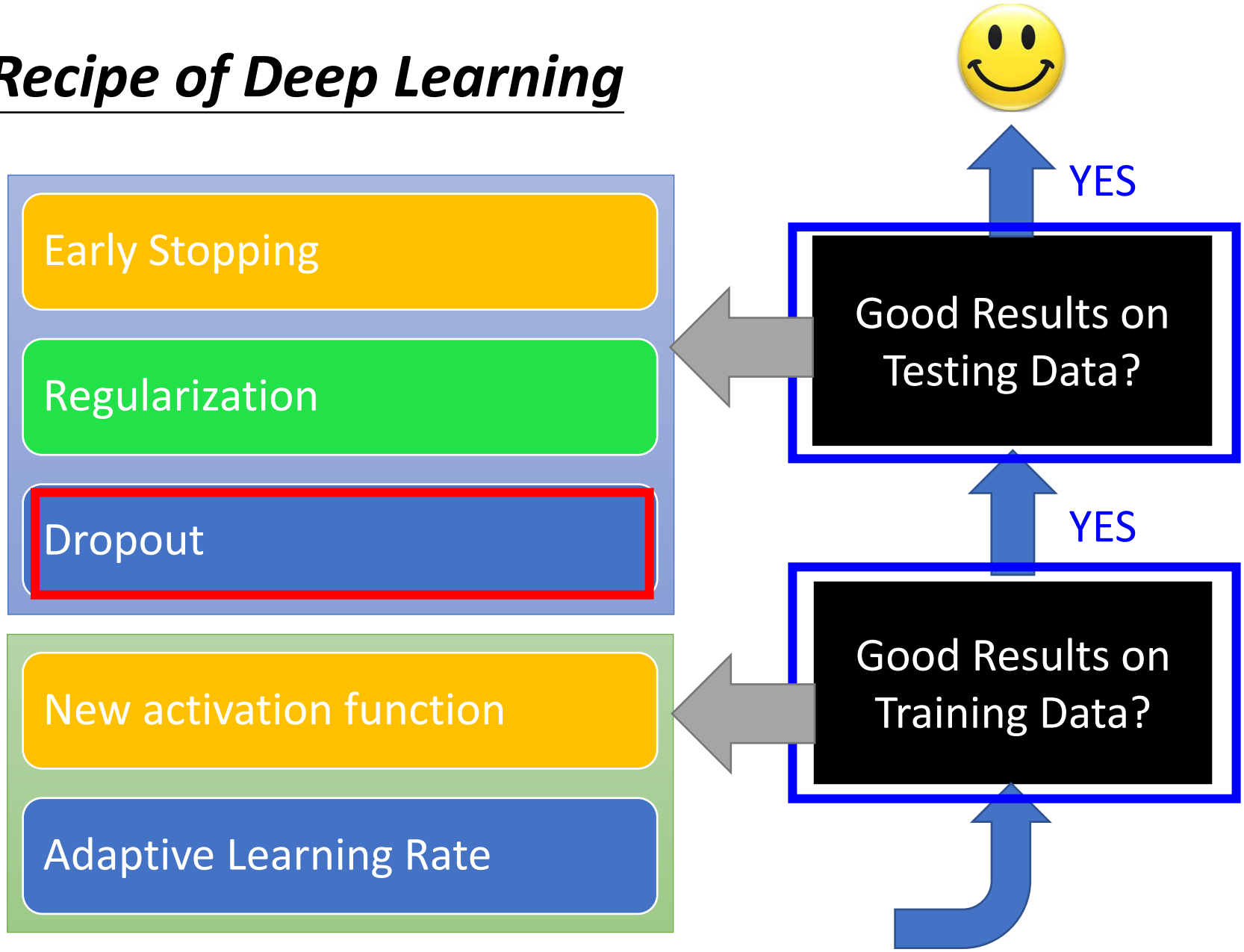
Regularization - Weight Decay

- Our brain prunes out the useless link between neurons.



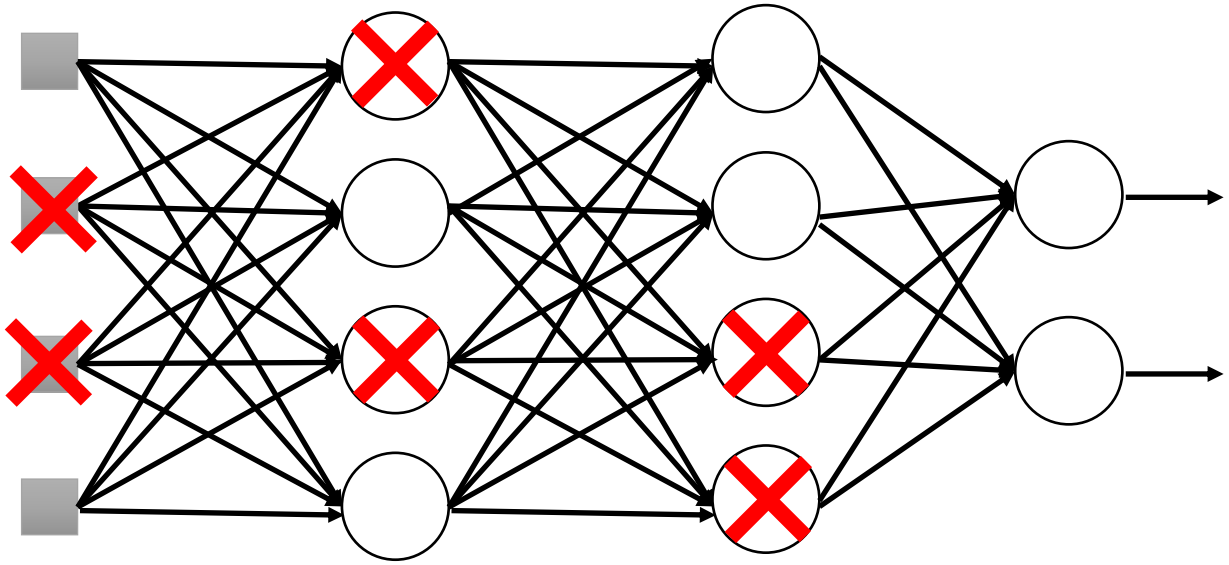
Doing the same thing to machine's brain improves the performance.

Recipe of Deep Learning



Dropout

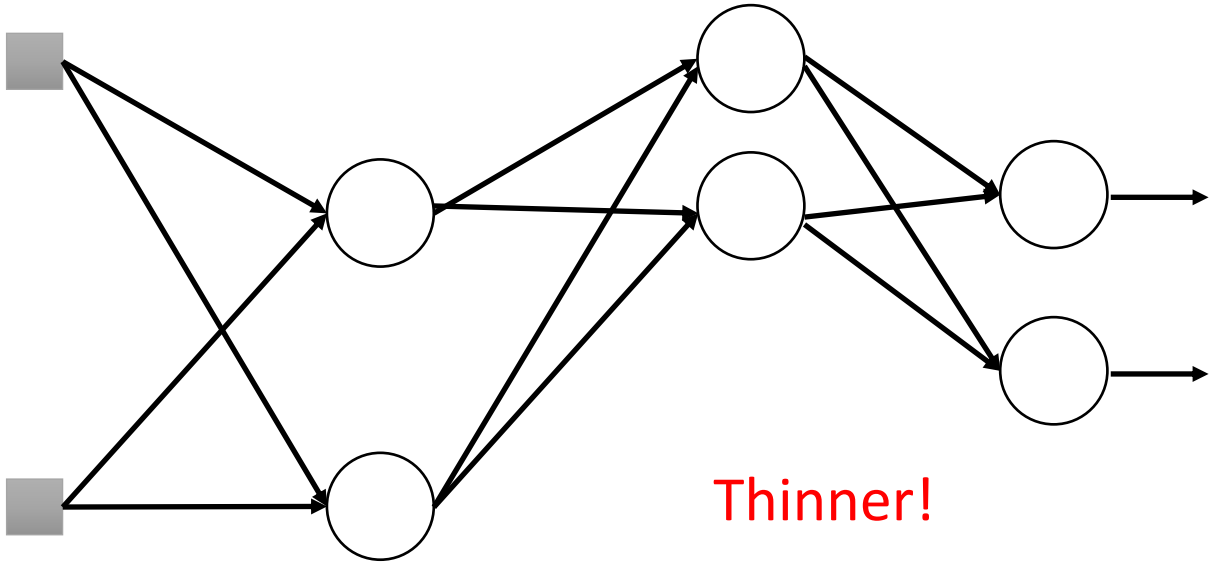
Training:



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

Dropout

Training:



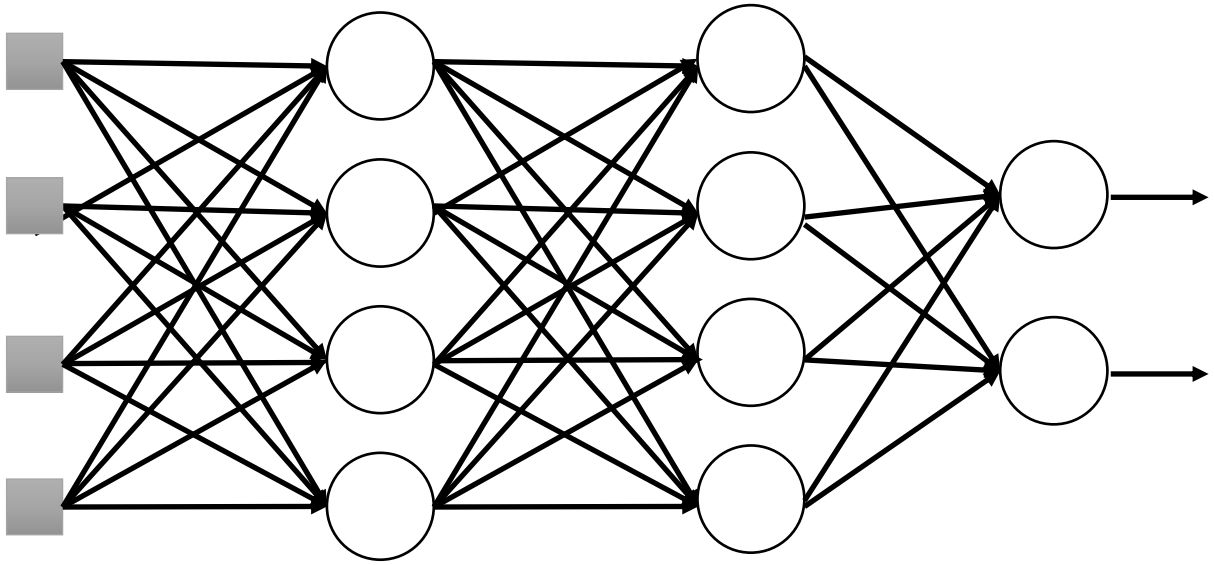
- **Each time before updating the parameters**
  - Each neuron has p% to dropout

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

For each mini-batch, we resample the dropout neurons

Dropout

Testing:



- **No dropout**
  - If the dropout rate at training is p%, all the weights times 1-p%
  - Assume that the dropout rate is 50%. If a weight $w = 1$ by training, set $w = 0.5$ for testing.

Dropout
- Intuitive Reason

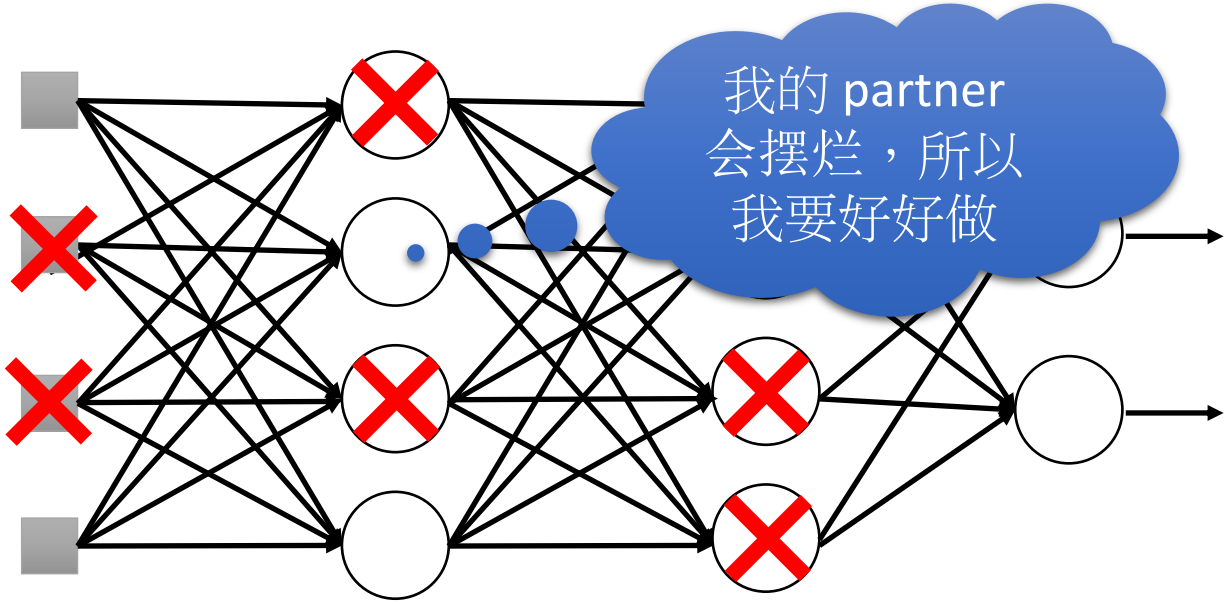
Testing

No dropout
(拿掉重物后就变很强)

Training

Dropout (脚上绑重物)

Dropout - Intuitive Reason



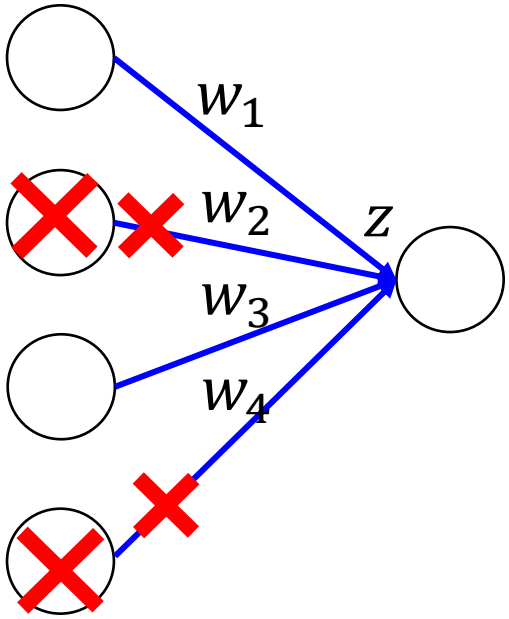
- When teams up, if everyone expect the partner will do the work, nothing will be done finally.
- However, if you know your partner will dropout, you will do better.
- When testing, no one dropout actually, so obtaining good results eventually.

Dropout - Intuitive Reason

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

Training of Dropout

Assume dropout rate is 50%

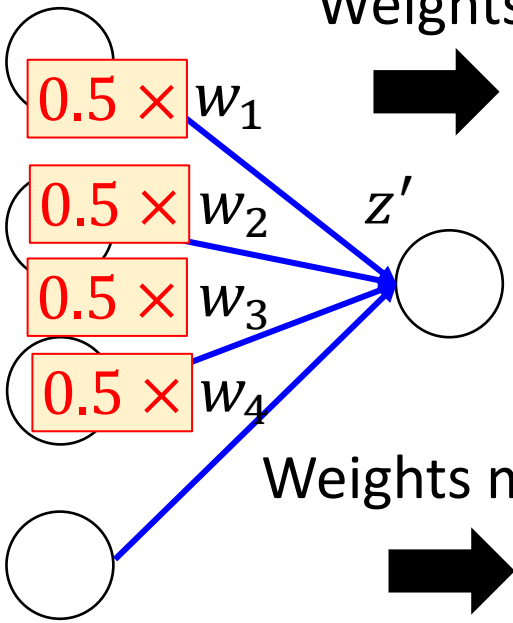


Testing of Dropout

No dropout

Weights from training

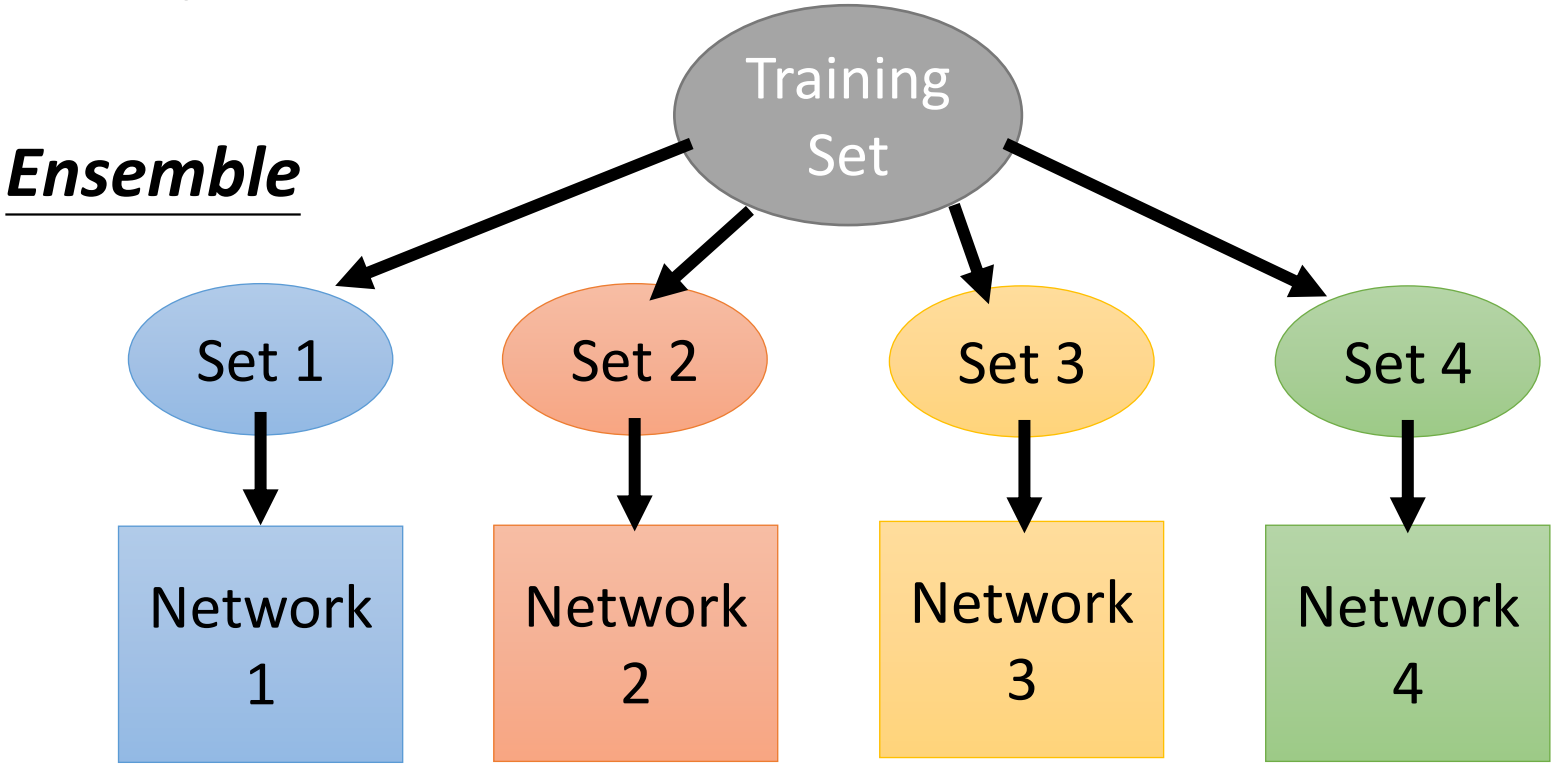
$z' \approx 2z$



Weights multiply 1-p%

$z' \approx z$

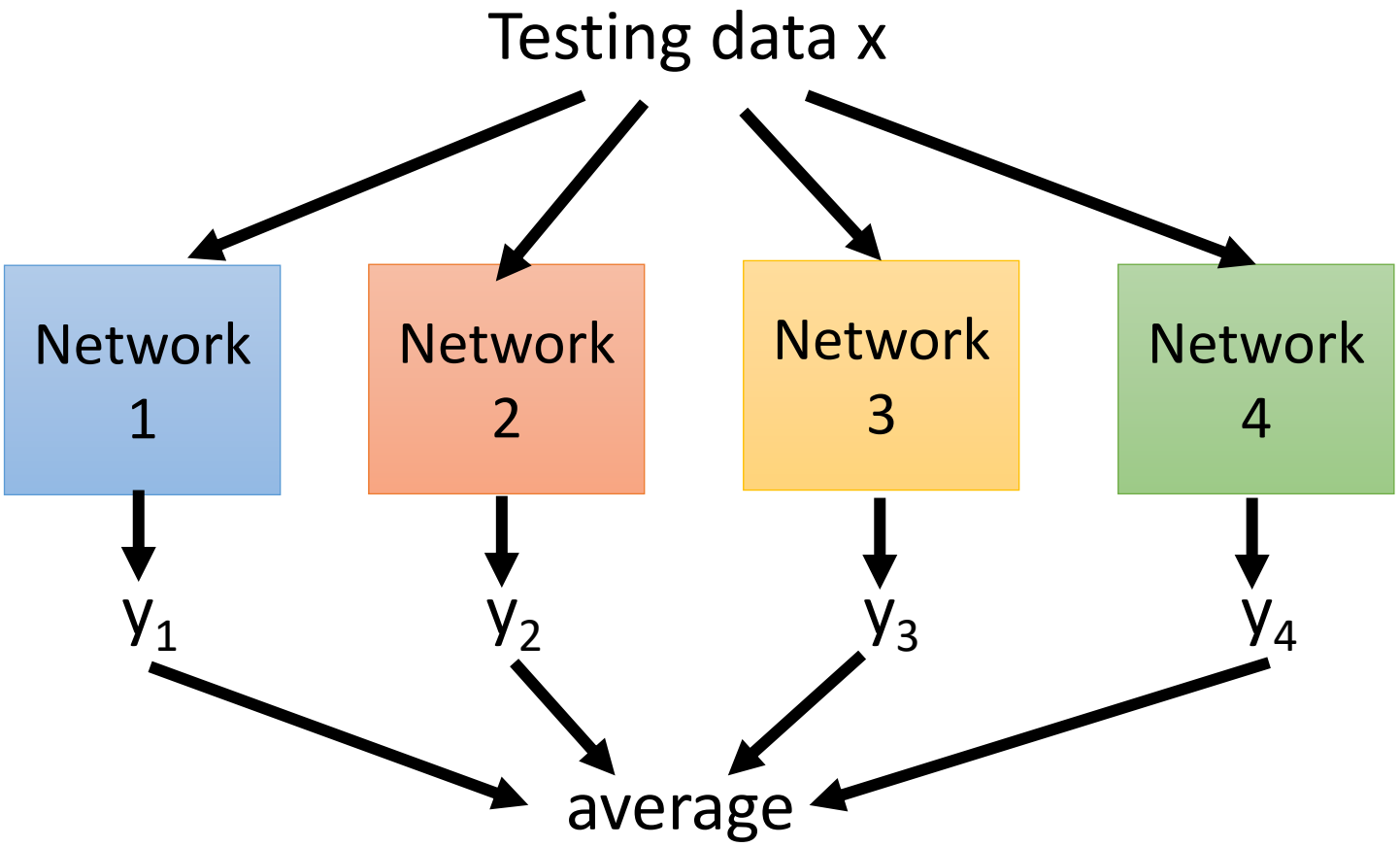
Dropout is a kind of ensemble.



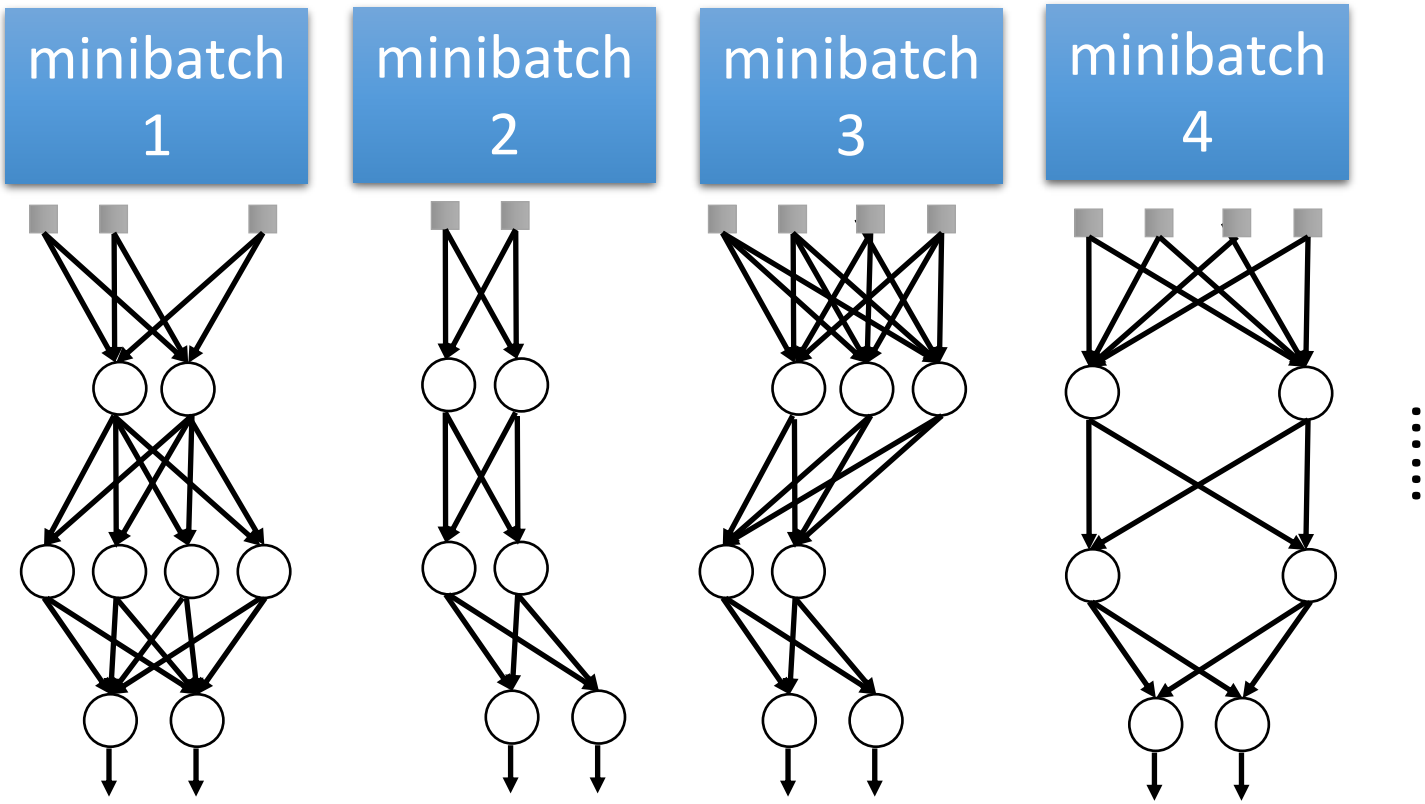
Train a bunch of networks with different structures

Dropout is a kind of ensemble.

Ensemble



Dropout is a kind of ensemble.



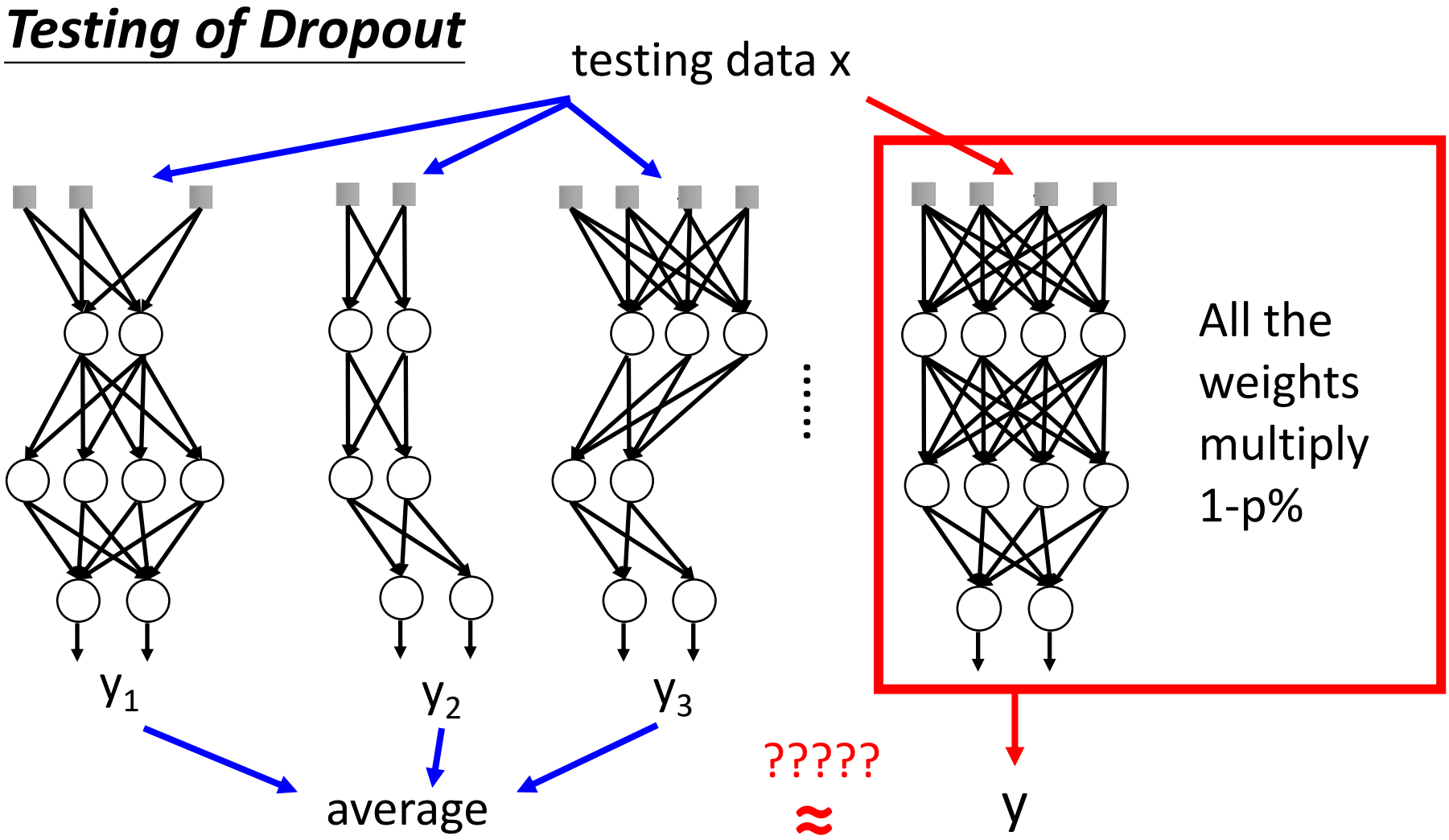
Training of Dropout

M neurons

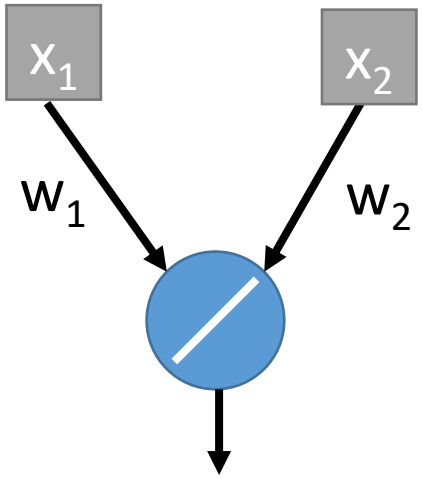
2^M possible networks

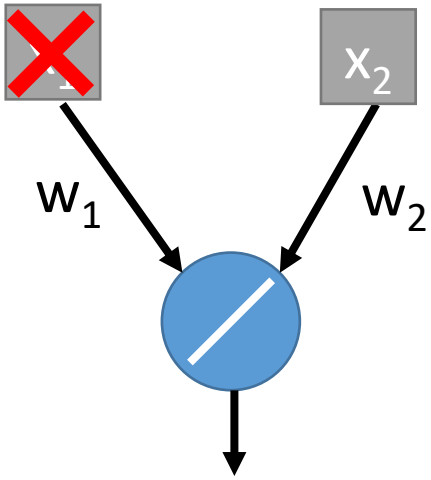
- Using one mini-batch to train one network
- Some parameters in the network are shared

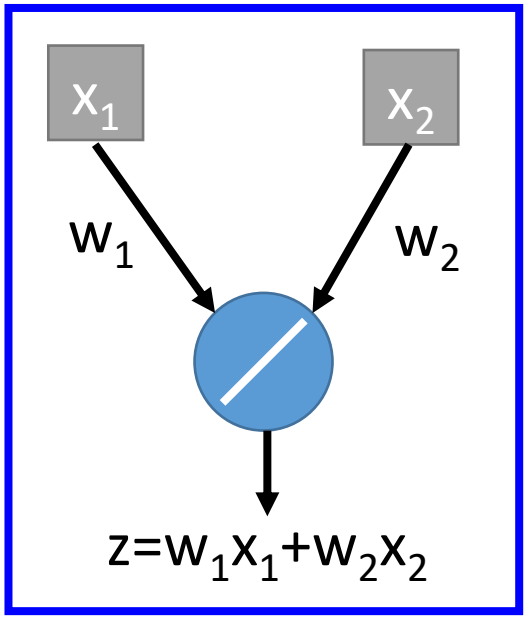
Dropout is a kind of ensemble.

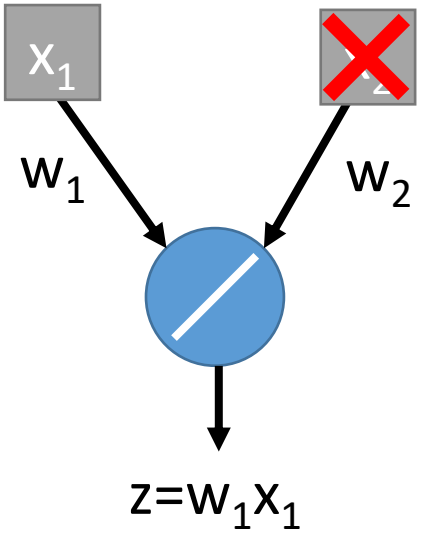


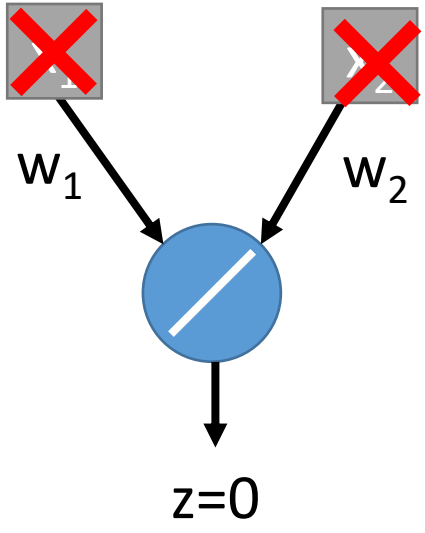
Testing of Dropout

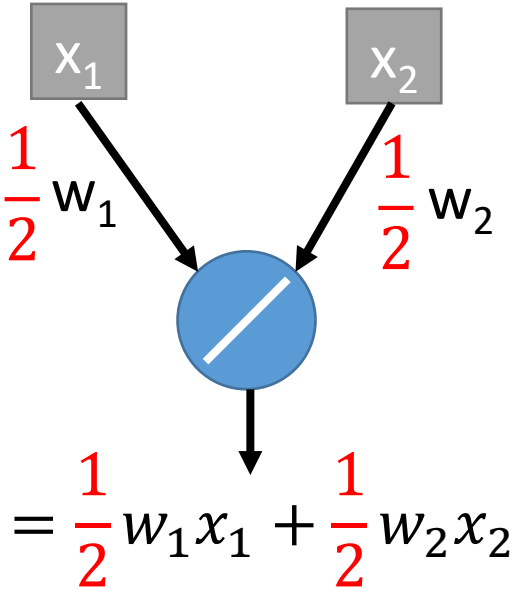




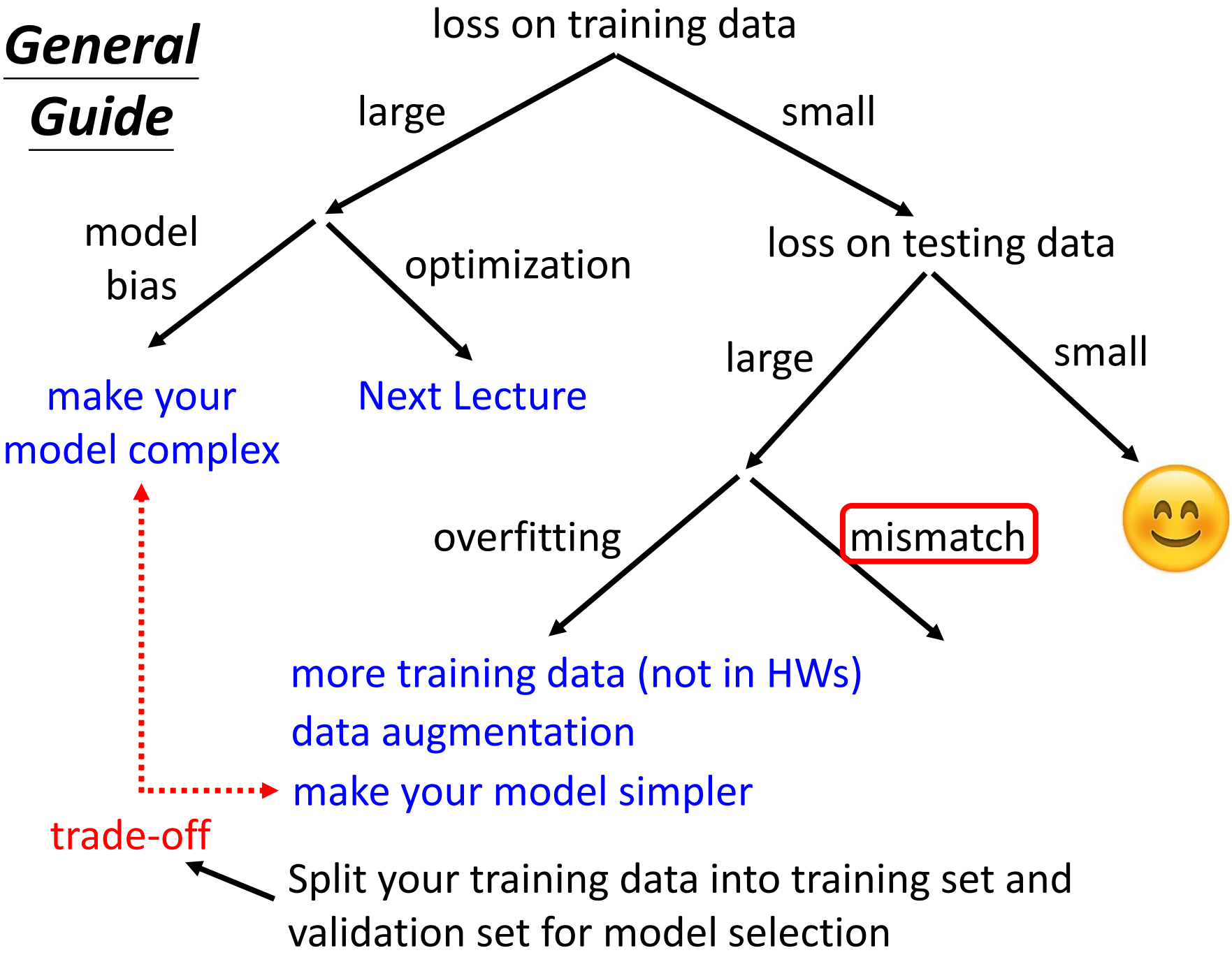








General Guide



Mismatch

- Your training and testing data have different distributions. Be aware of how data is generated.

Training Data

Simply increasing the training data will not help.

Testing Data