DEEP LEARNING

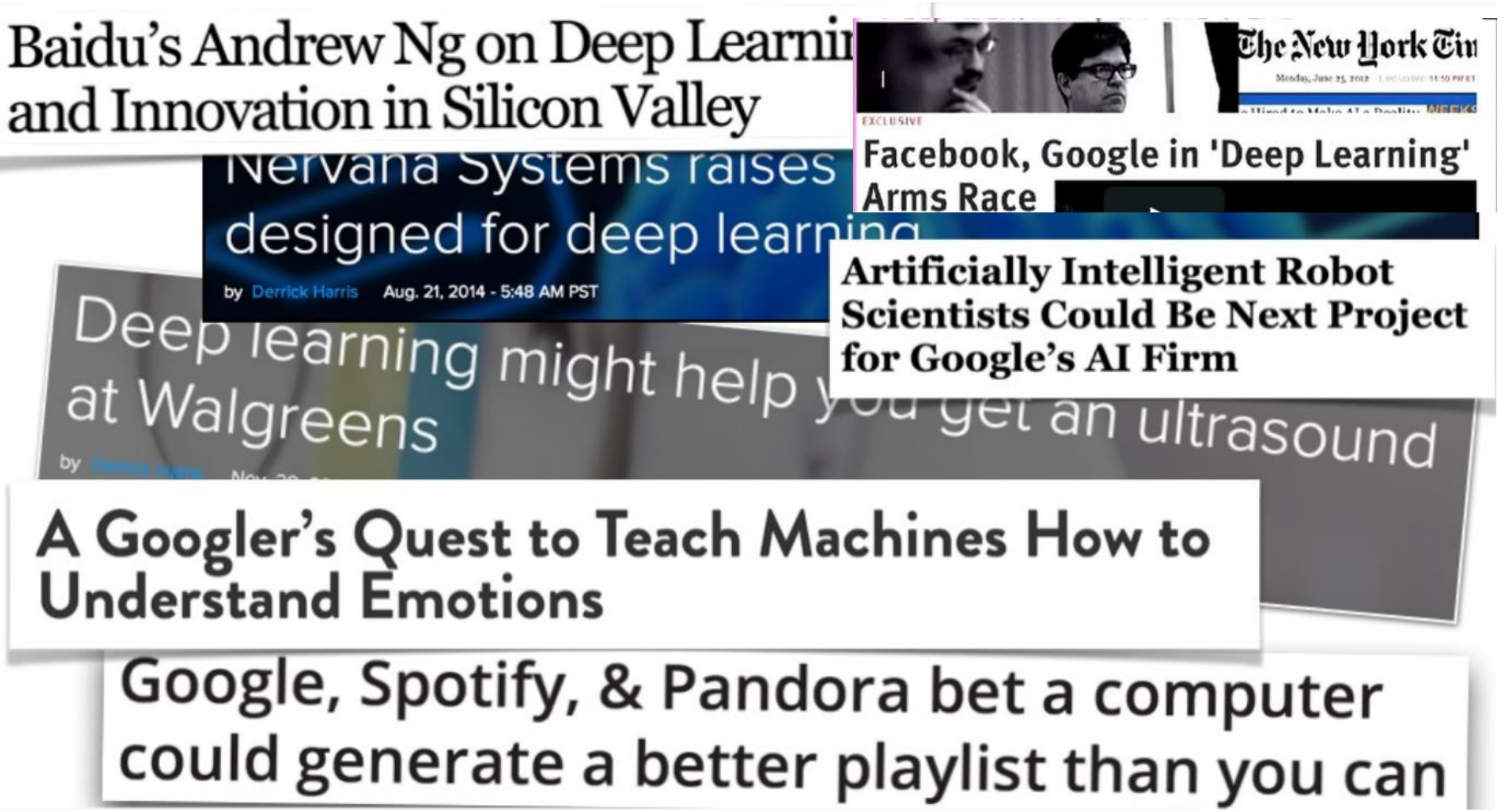
Lecture-5 & 6 , Day-3

STTP on “Deep Learning, Computer Vision and Speech Processing”

By: Suprava, Patnaik, Professor, ExTC, XIE, Mumbai

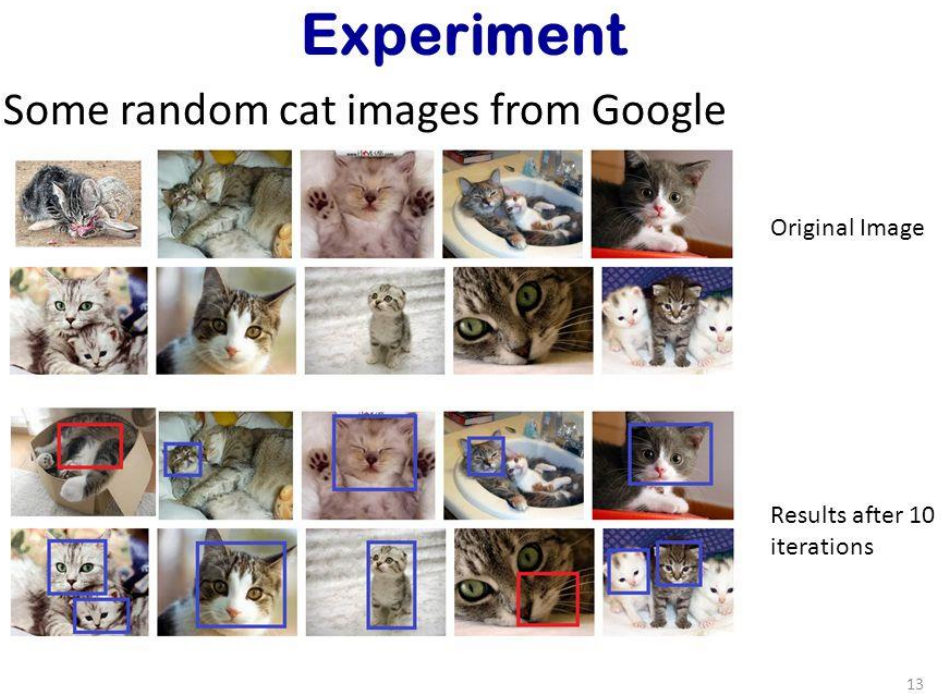
- Introduction
- Deeplearning vs traditional learning
  - Modularity
  - Augmenting
  - Learned Features
  - Weight Sharing (CNN)
  - Way out to Vanishing Gradient
  - Pooling
  - Auto-encoders

INTRODUCTION



DEEP LEARNING IS SUDDENLY
CHANGING OUR LIFE

Key-Moments in De



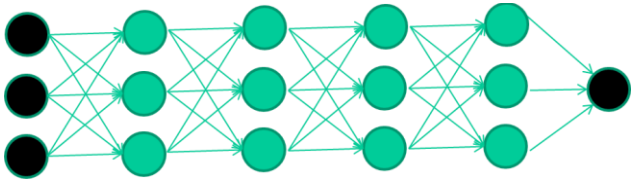
- 1958-1986: Era of Neural
- 1986-1997: research goals language processing, spea
- 1997: IBM's **Deep Blue** be in chess by using AI techn
- 2000-2012: **ImageNet** by
- 2012: Google Brain- "cat experiment"-A NN shown 10 million unlabeled YouTube images, has trained itself to recognize cats.
- 2014: Google acquired DeepMind for $600 milion (reinforcement learning)
- 2015: IBM group outperformed human on ImageNet
- DeepMind's AlphaGo defeated world champion Lee Sedol

Decades-old discoveries are now electrifying..!!

- Speech-recognition functions on our smartphones work much better than they used to. (siri, cortona, Alexa)
- Automatically organize collections of photos with no identifying tags (Apple, Google, Facebook)
- Better image recognition is crucial to unleashing improvements in robotics, autonomous drones, and of course self-driving cars (Ford, Tesla, Uber, Google)
- Machine translation, recognition and processing have become far more convincing.

Neural network vs ”Deep learning”

- Goal: Replicating Human Perception
  1. ‘Deep Learning’ means using a neural network with several layers of neurons between input and output
  2. The series of layers between input & output do feature identification and processing in a series of stages, just as our brains seem to.



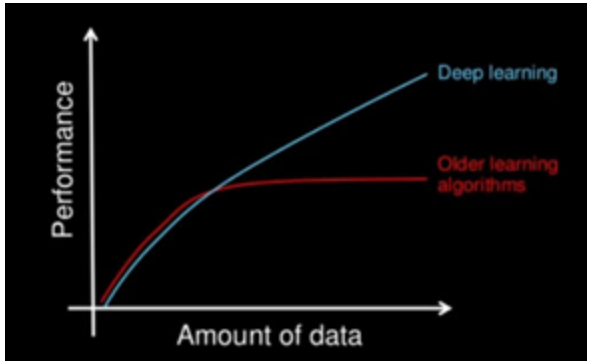
- Challenges of computer vision and audio processing are tackled.
- DL is providing breakthrough results in speech recognition and image classification …How..????

How……??

- By harnessing vast computational power. (CPU…to.. GPU)

- Access to enormous storehouse of data- :images, video, audio, text etc.

- Technology to handle algorithms.



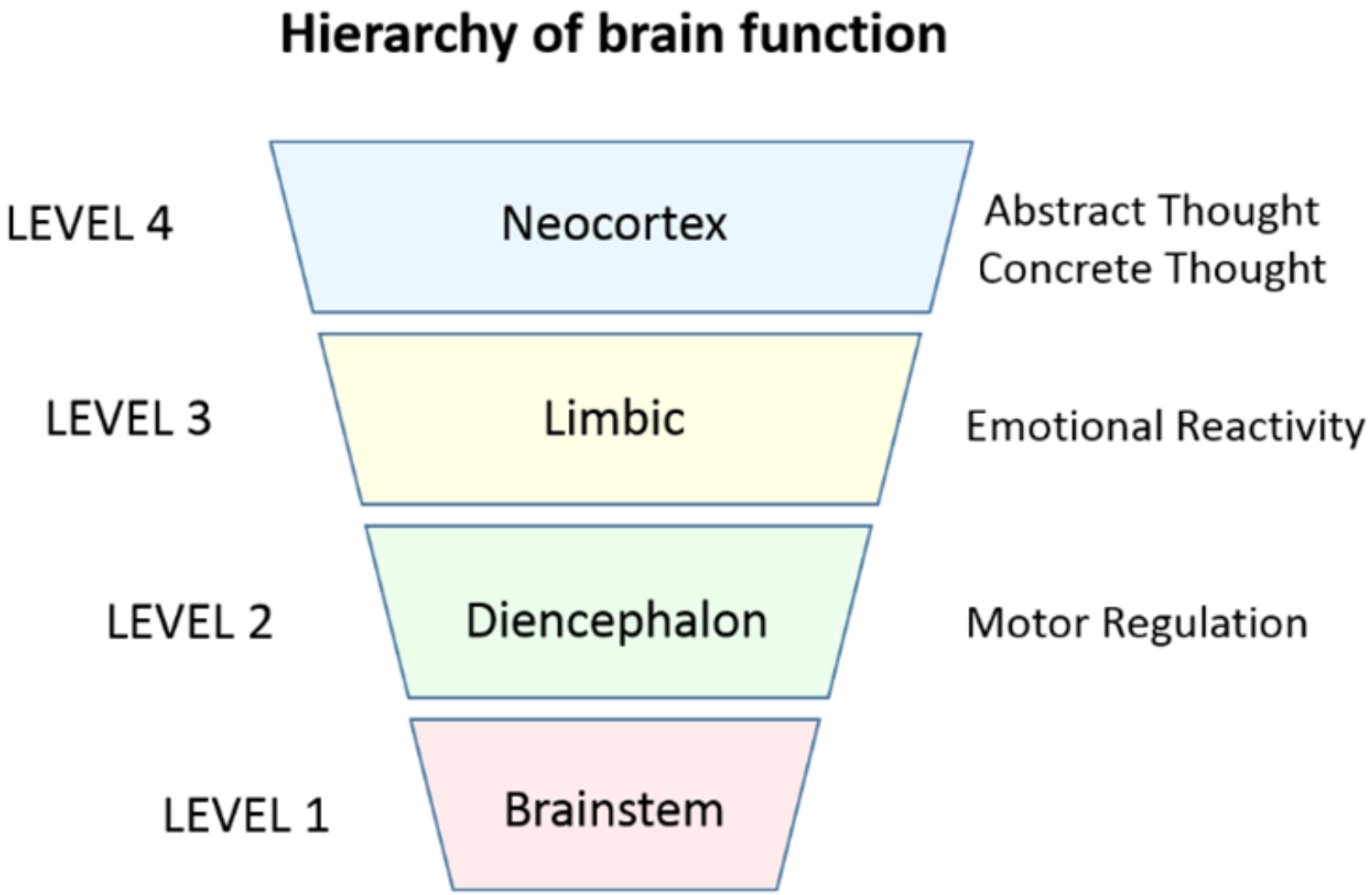
Visual Cortex: **sensing and perceiving?**

- Sensations linger in the memory for less than a second
- Perception is the process by which the brain makes sense of these incoming data, mixing memory, emotion, and cognition into the experience.

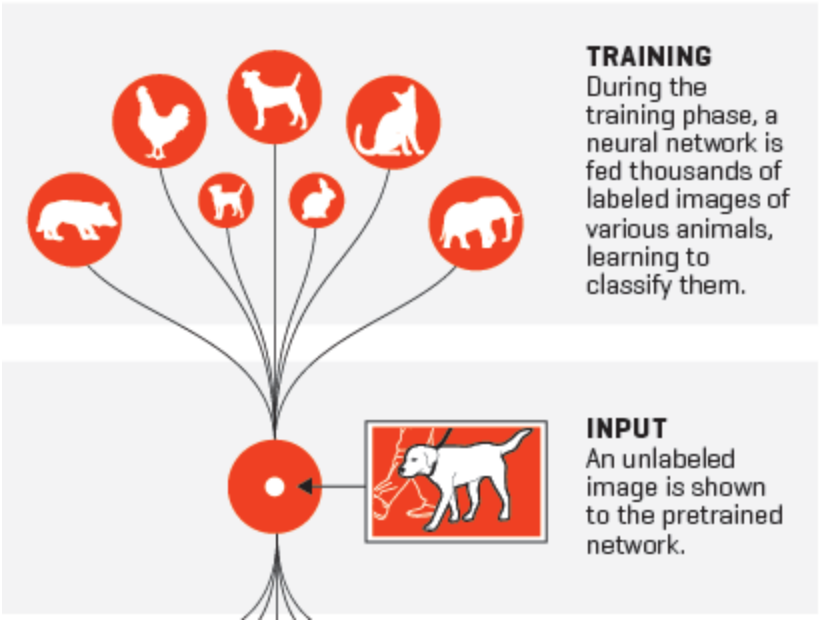
- Higher brain areas don't just respond to sensory information, they actively condition it: inhibiting irrelevant input, and establishing meaningful structures from fragments-e.g. constructing words from partial sounds.

- Small simple receptive fields are combined to form complex receptive fields

Describe these ….!!!!

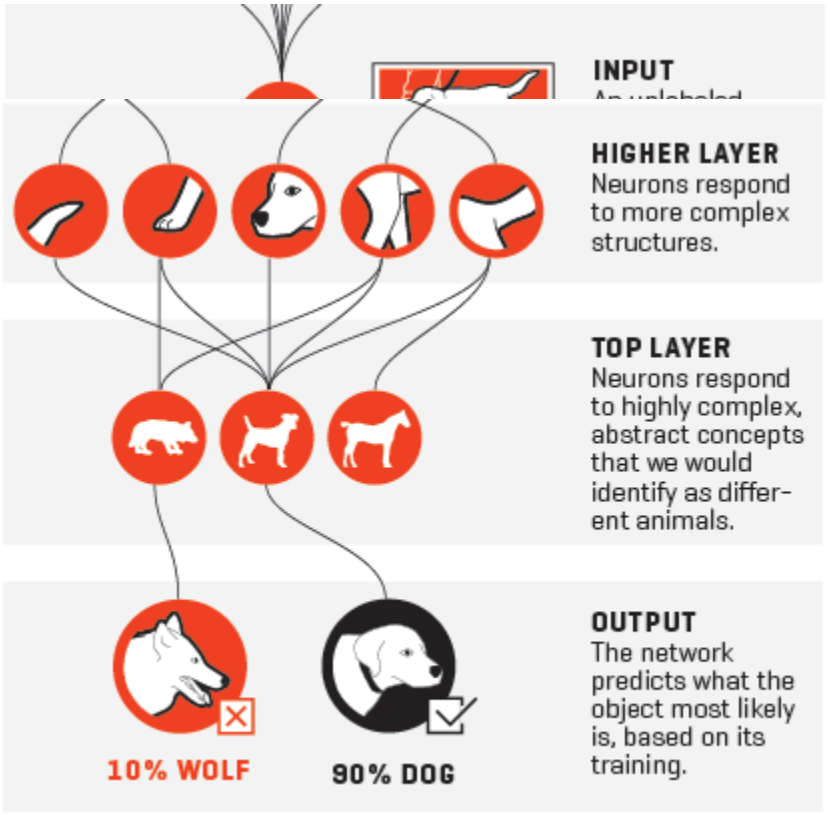


How NN recognizes a dog in a photo?



TRAINING
During the training phase, a neural network is fed thousands of labeled images of various animals, learning to classify them.

INPUT
An unlabeled image is shown to the pretrained network.



INPUT

HIGHER LAYER
Neurons respond to more complex structures.

TOP LAYER
Neurons respond to highly complex, abstract concepts that we would identify as different animals.

OUTPUT
The network predicts what the object most likely is, based on its training.

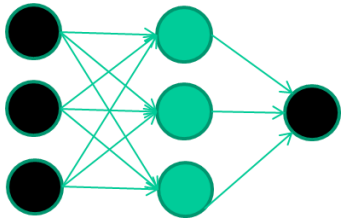
10% WOLF

90% DOG

TRADITIONAL LEARNING VS DEEPLEARNING

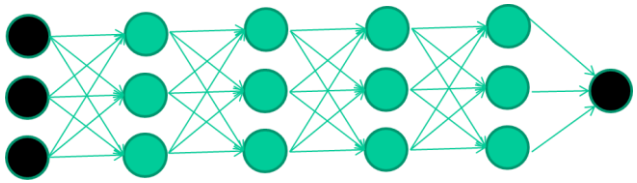
What's new…..?

- **we have always had good algorithms for learning the weights in networks with 1 or 2 hidden layer**



- **but these algorithms are not good at learning the weights for networks with more hidden layers**

- **what's new is: algorithms for training many-layered networks**



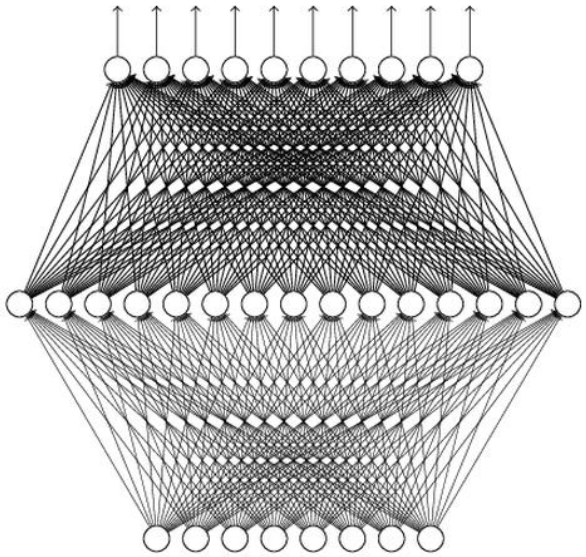
Universality Theorem

Any continuous function f

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

Can be realized by a network with one hidden layer

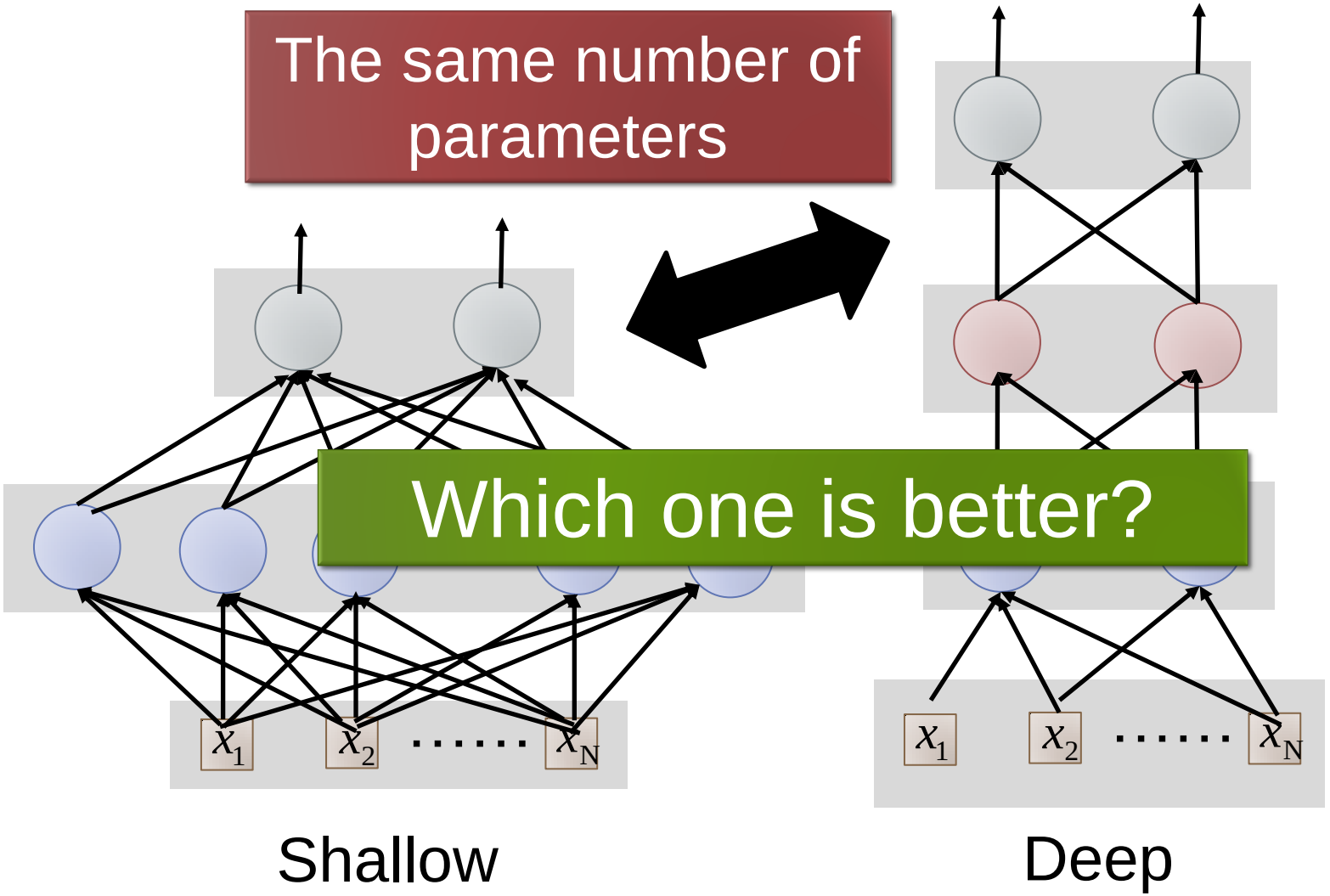
(given **enough** hidden neurons)



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

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

Fat + Short v.s. Thin + Tall



Fat + Short v.s. Thin + Tall

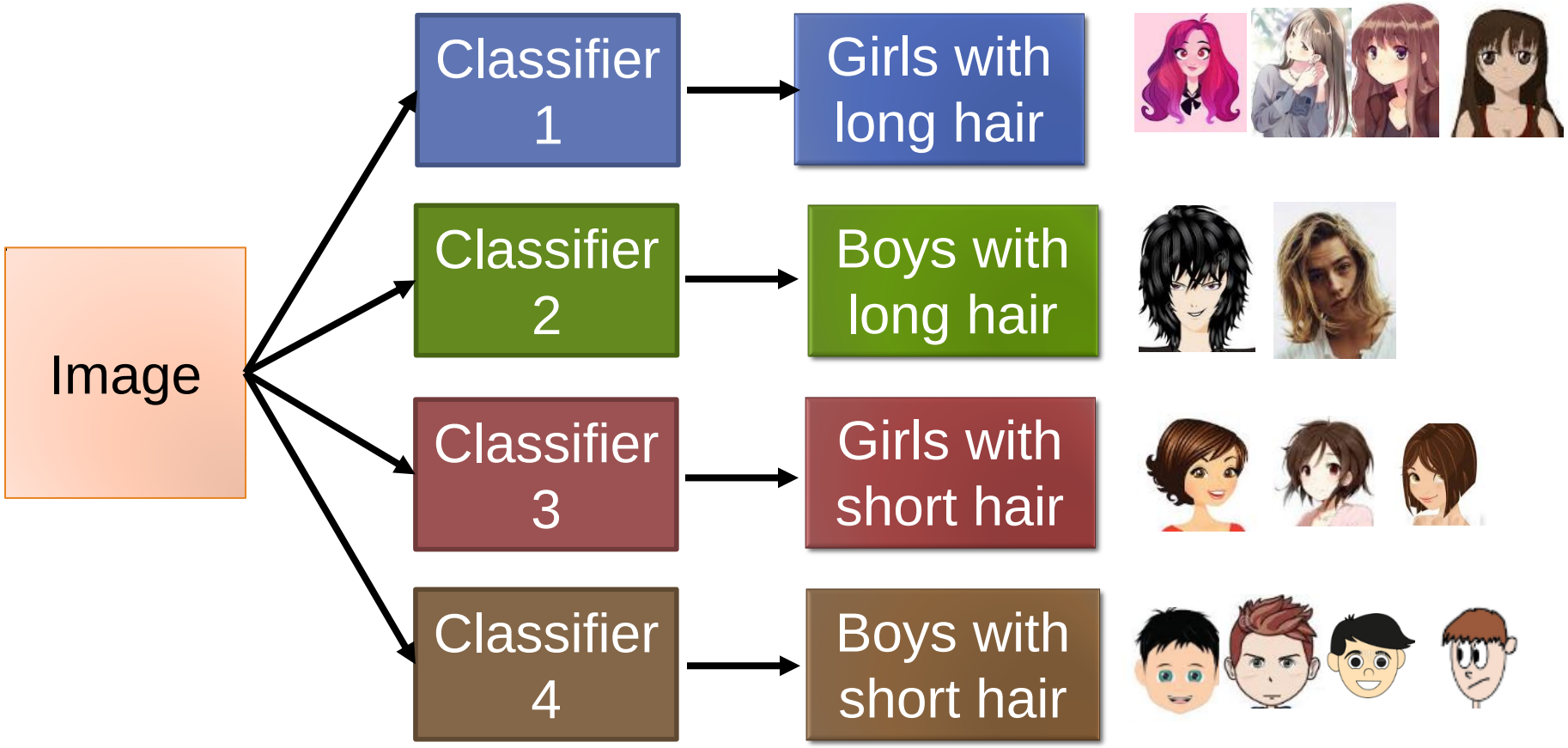
Layer X Size	Word Error Rate (%)	Layer X Size	Word Error Rate (%)
1 X 2k	24.2		
2 X 2k	20.4		
3 X 2k	18.4		
4 X 2k	17.8		
5 X 2k	17.2	1 X 3772	22.5
7 X 2k	17.1	1 X 4634	22.6
		1 X 16k	22.1

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

Modularity

Why Deep?

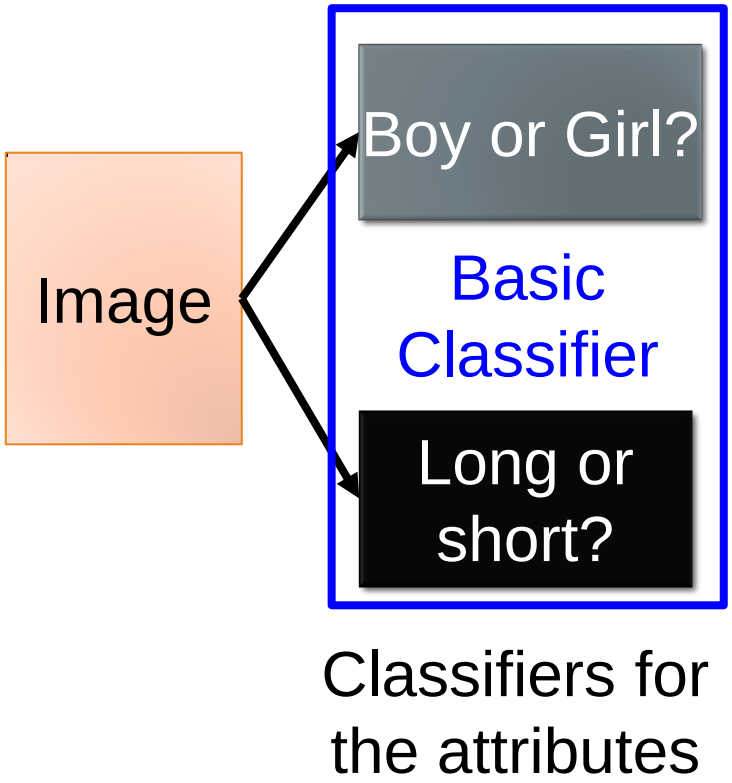
- Deep → Modularization

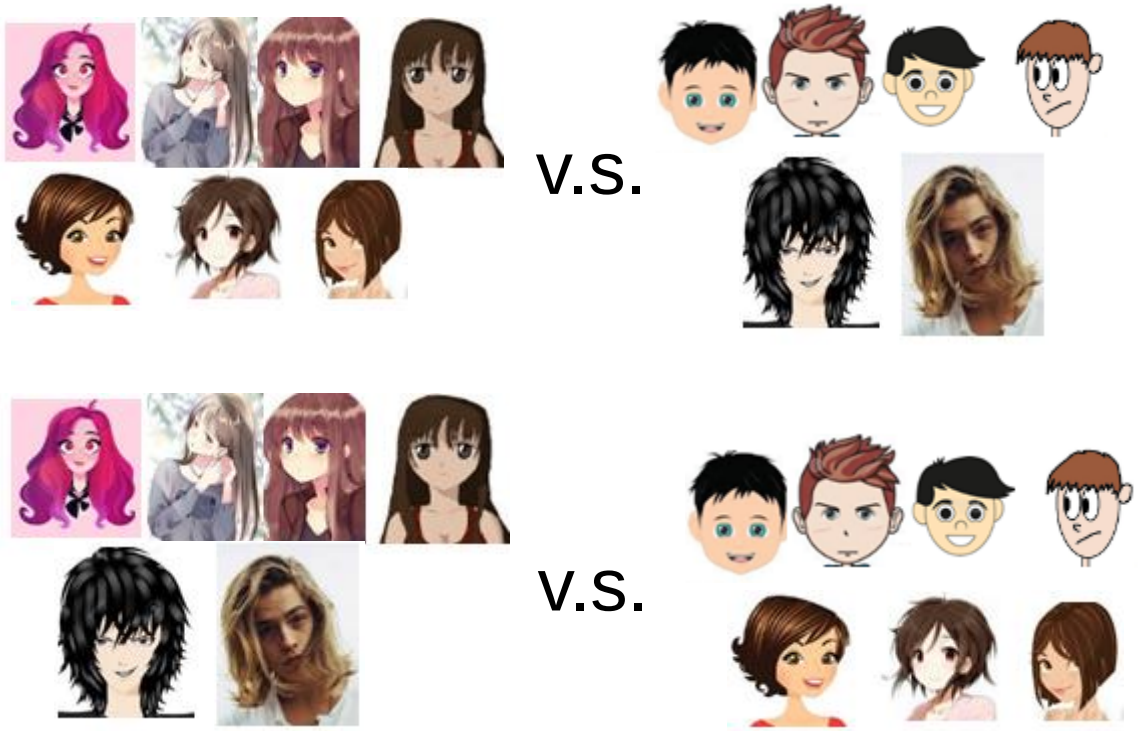


Why Deep?

Each basic classifier can have sufficient training examples.

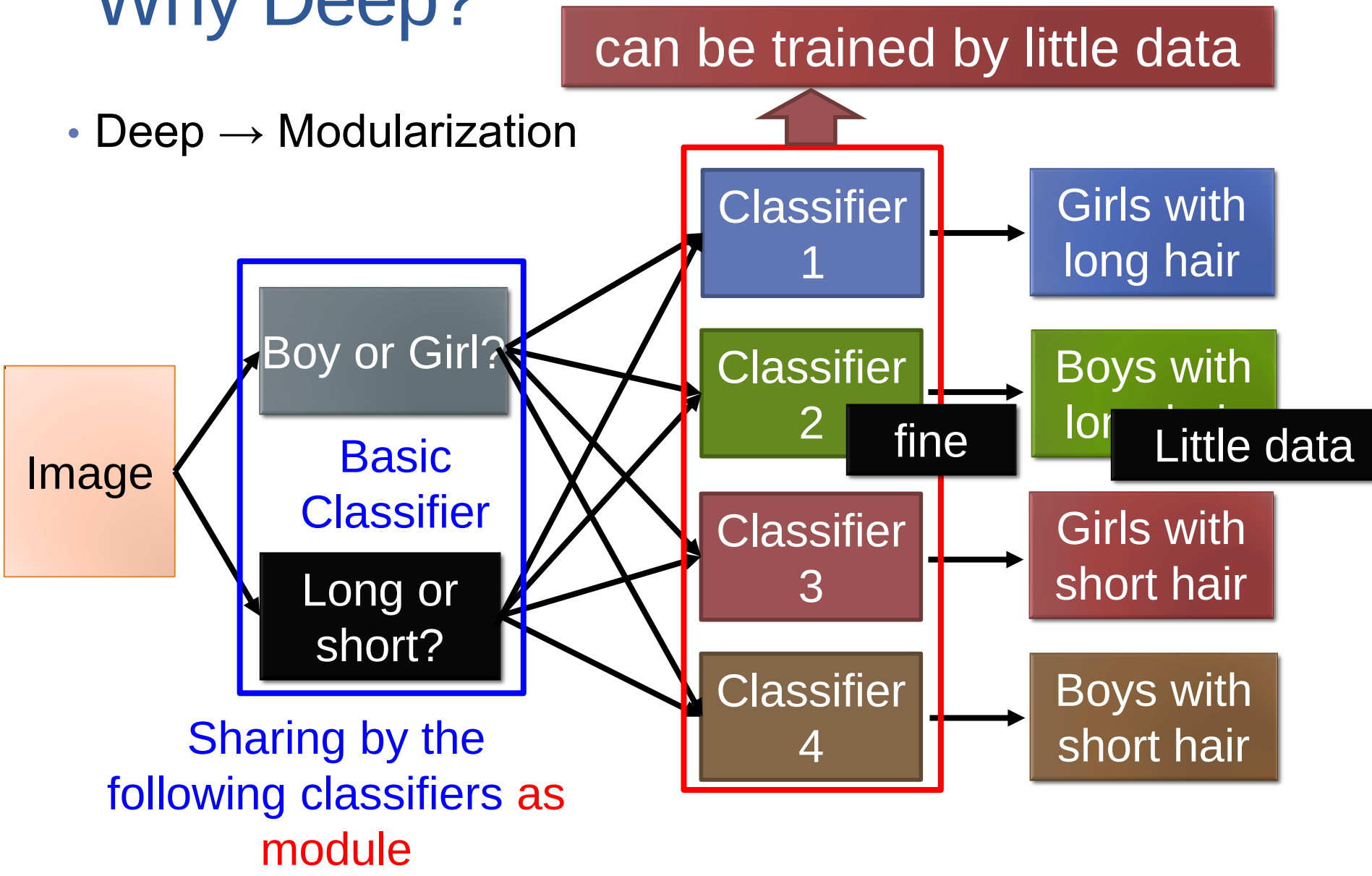
- Deep → Modularization





Why Deep?

- Deep → Modularization

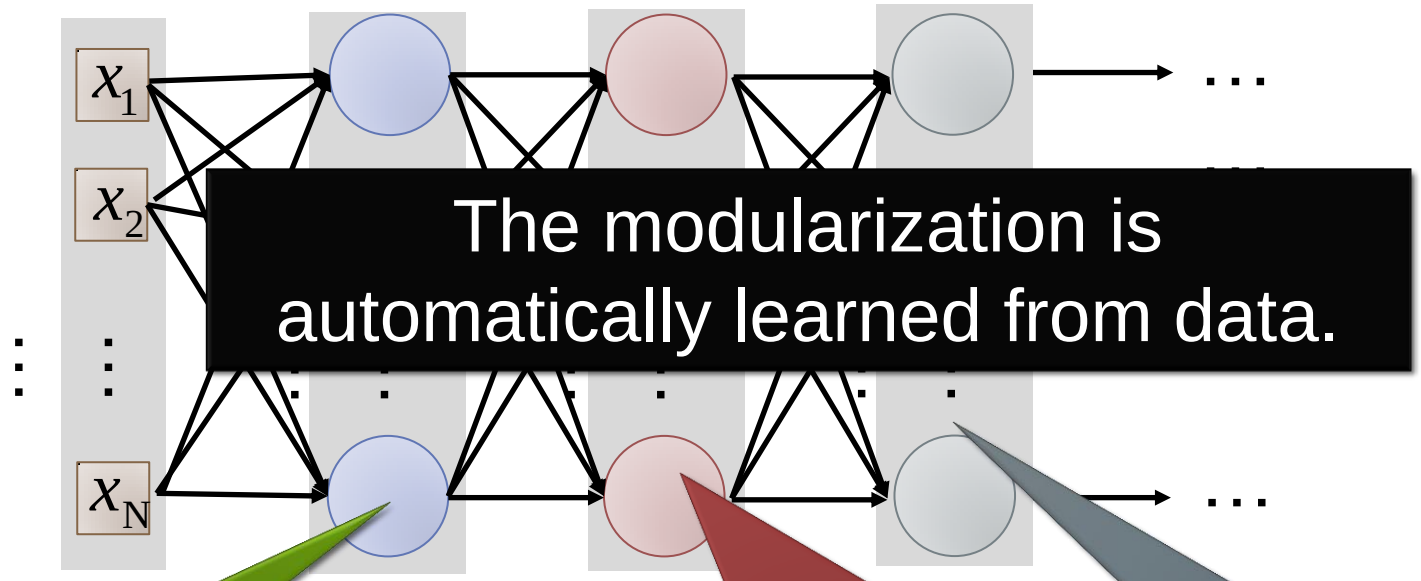


Why Deep?

Deep Learning also works on small data set like TIMIT.

- Deep → Modularization

→ Less training data?



The most basic classifiers

Use 1st layer as module to build classifiers

Use 2nd layer as module combinations…

Modularity in deeplearning

- **Splitting**—Independent Modules (auto-encoders).
- **Substituting**—Modules can be substituted and interchanged.
- **Augmenting**—New Modules can be added to create new solutions.
- **Inverting**—The hierarchical dependencies between Modules can be rearranged.
- **Porting**—Modules can be applied to different contexts. **generative adversarial networks (GANs):**

generator, generates new data instances, while the other, the *discriminator*, evaluates them for authenticity

- **Excluding**—Existing Modules can be removed to build a usable solution.

INFORMATION AUGMENTATION

GAN for data augmentation

- Face Detection:

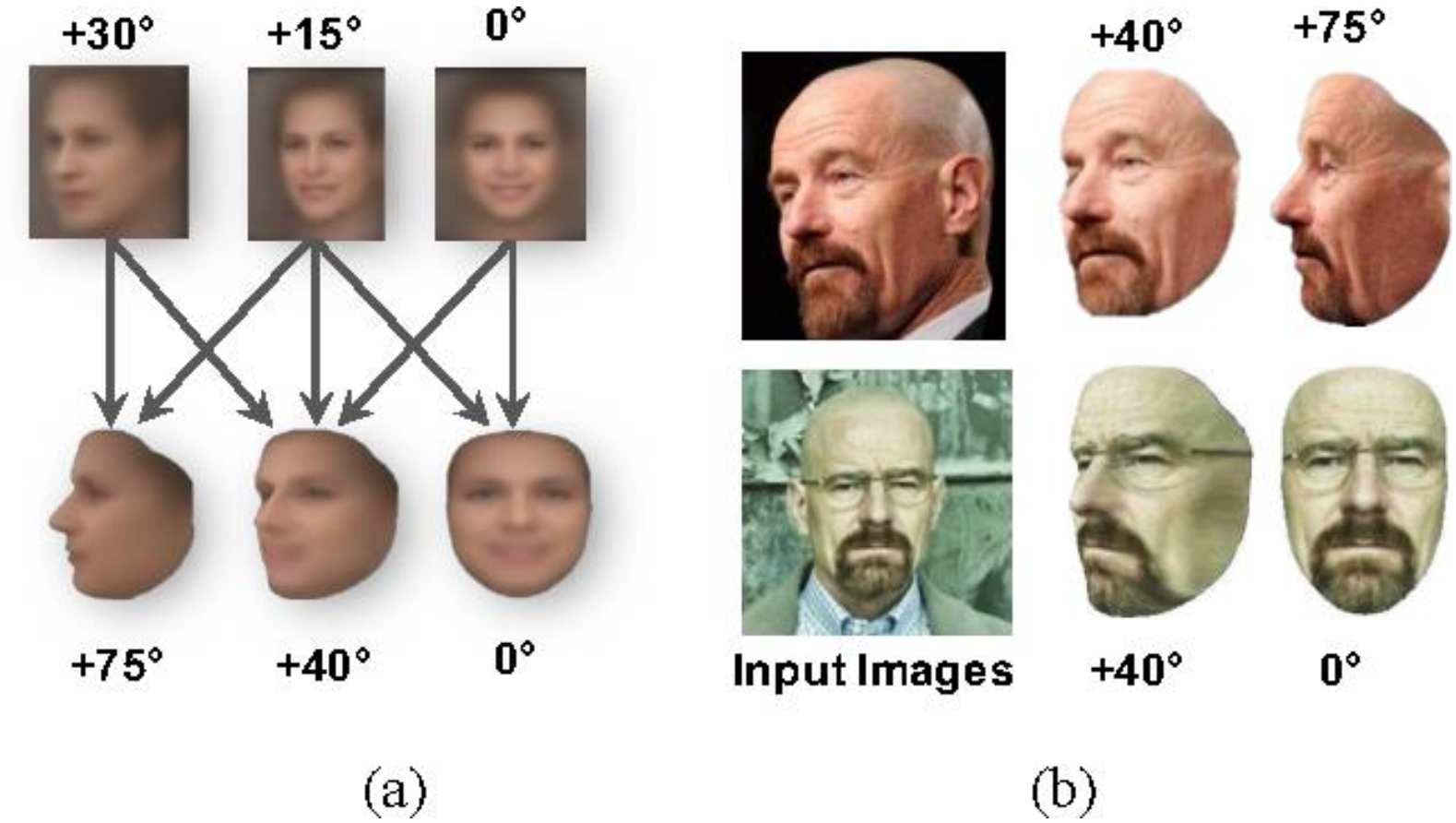


Figure 2: Example of images showing occlusion.

ion

- Face Detection:

Male with Spec - Male + Female

Example of images showing occlusion.

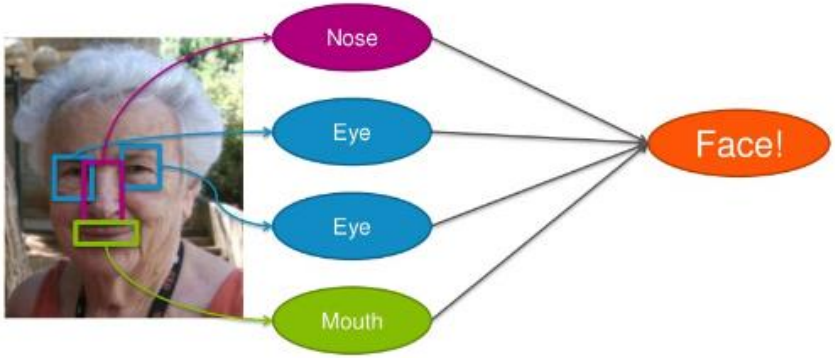
Stack-GAN

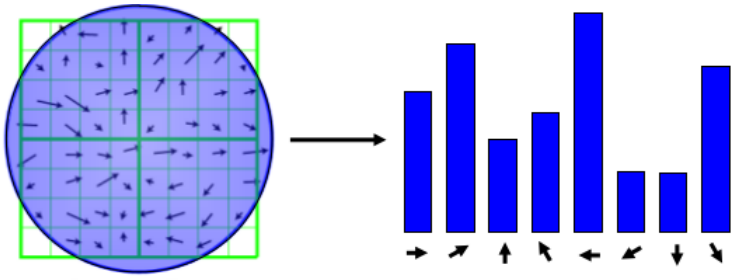
- The task here is to take as input a text description and generate an image corresponding to the description.
- The second GAN is able to refine the fuzzy image into one of a higher resolution.
- The motivation here is to achieve a **better decoupling** between the pairs and **periodically swap around the discriminator and generator pairs**

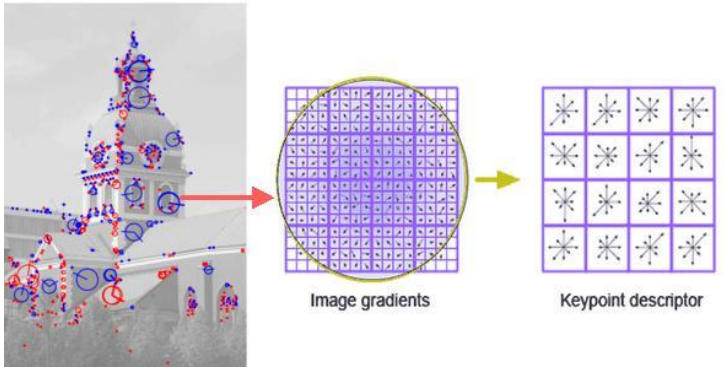
LEARNED FEATURES

Why features..?

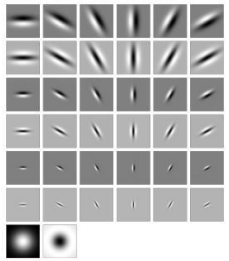
- Features = local descriptors
- Combined to make objects
- Low-level representation
- Eyes + Mouth + Nose+..=Face







Maximum Response Filters (ECCV 2002)

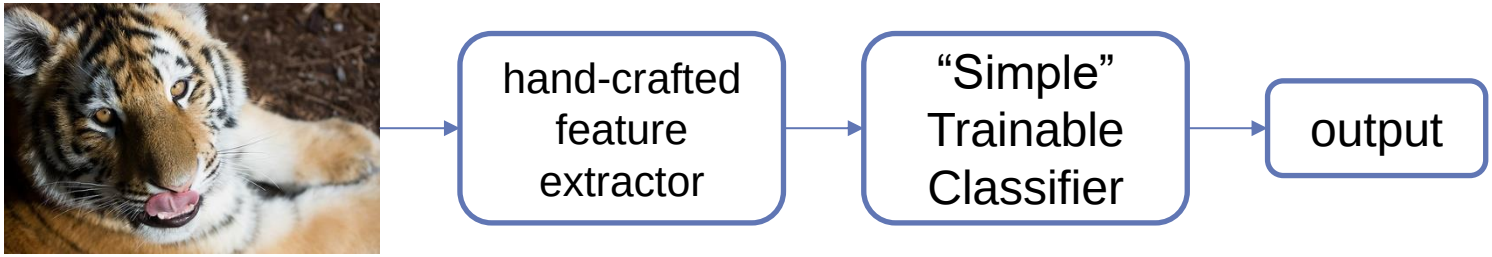


Maximum Response (MR8) filters – 38 filters, 8 filter responses

HOG
SIFT
TEXTONS
…..

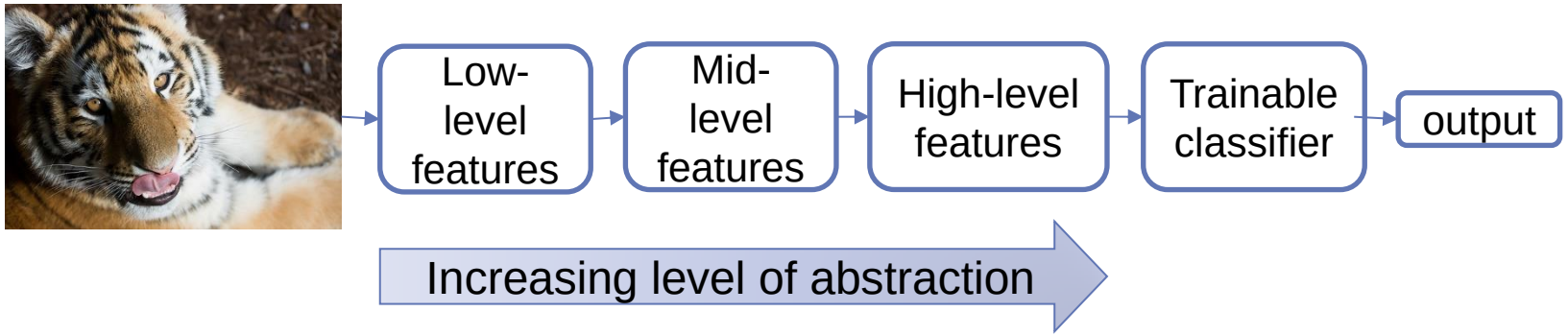
Feature extraction

- Traditional pattern recognition models use hand-crafted features and relatively simple trainable classifier.



- This approach has the following limitations:
  - It is very **tedious** and costly to develop hand-crafted features
  - The hand-crafted features are usually **highly dependent on applications**, and cannot be transferred easily to other applications

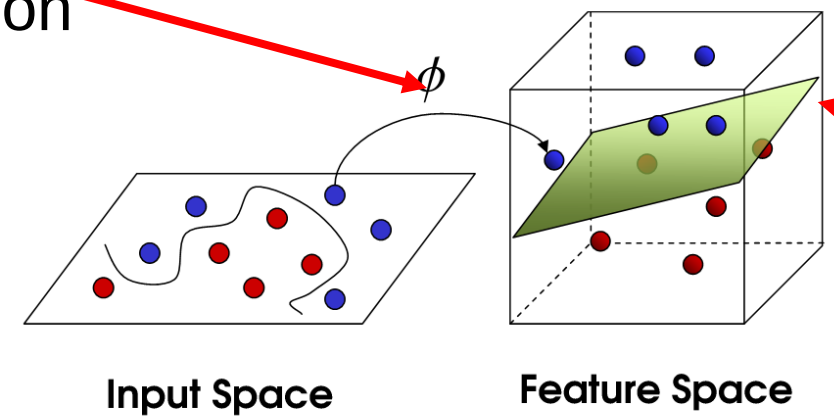
Learning Hierarchical Representations



- Hierarchy of representations with increasing level of abstraction. Each stage is a kind of trainable nonlinear feature transform
- Image recognition
  - Pixel → edge → texton → motif → part → object
- Text
  - Character → word → word group → clause → sentence → story

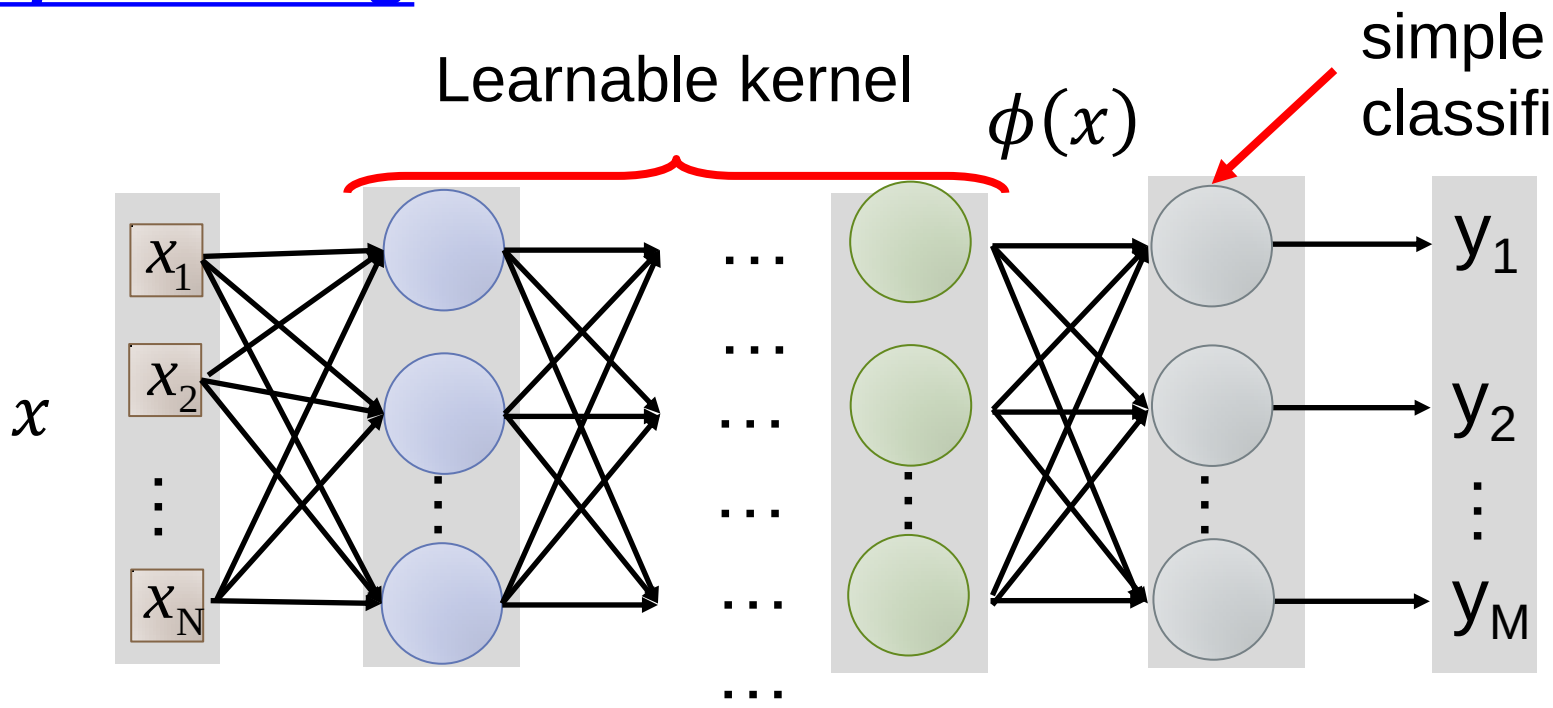
Hand-crafted kernel function

SVM

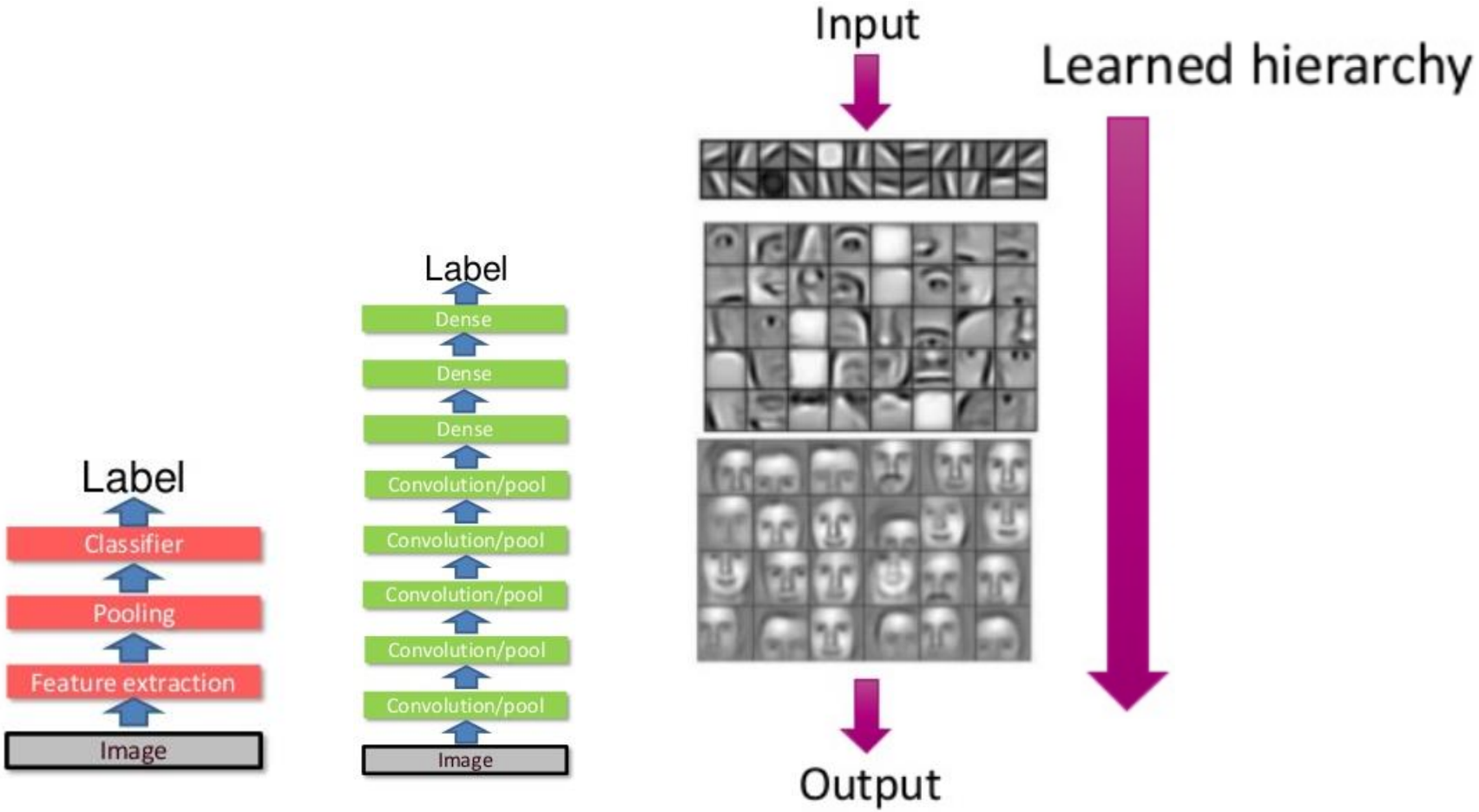


Apply simple linear classifier

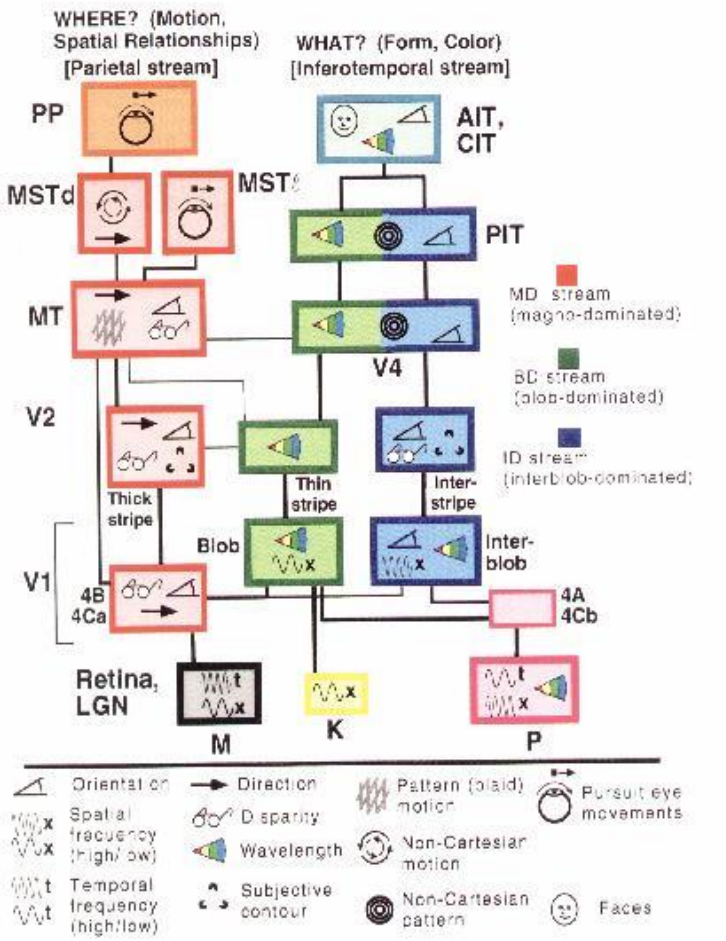
Deep Learning



Engineered vs Learned Features



The Mammalian Visual Cortex is Hierarchical



(van Essen and Gallant, 1994)

- It is good to be inspired by nature, but not too much.
- We need to understand which details are important, and which details are merely the result of evolution.
- Each module in Deep Learning transforms its input representation into a higher-level one, in a way similar to human cortex.

WEIGHT SHARING

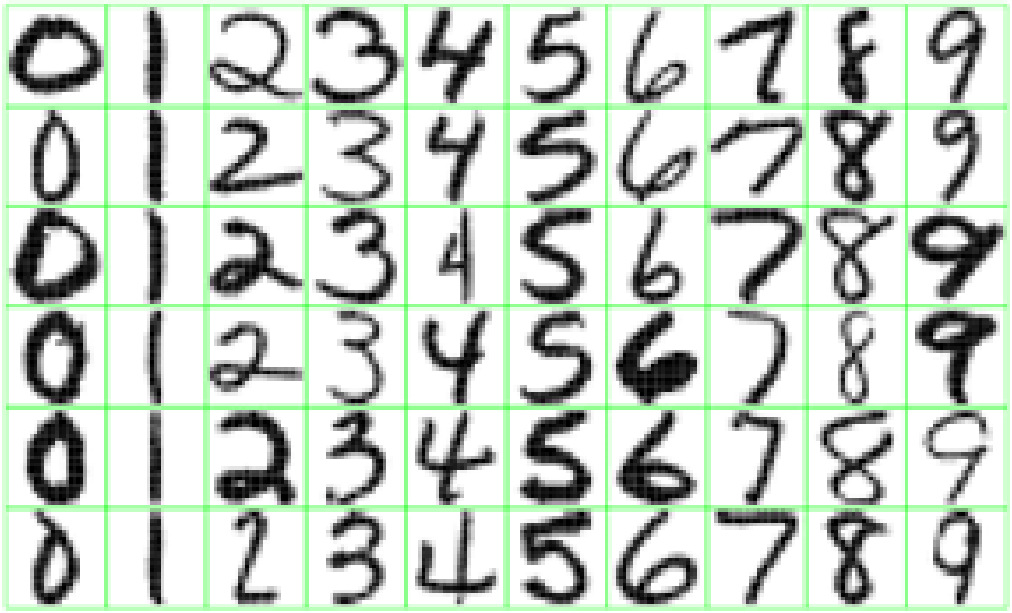


Figure 1.2: *Examples of handwritten digits from U.S. postal envelopes.*

What features might you expect a good NN to learn, when trained with data like this?

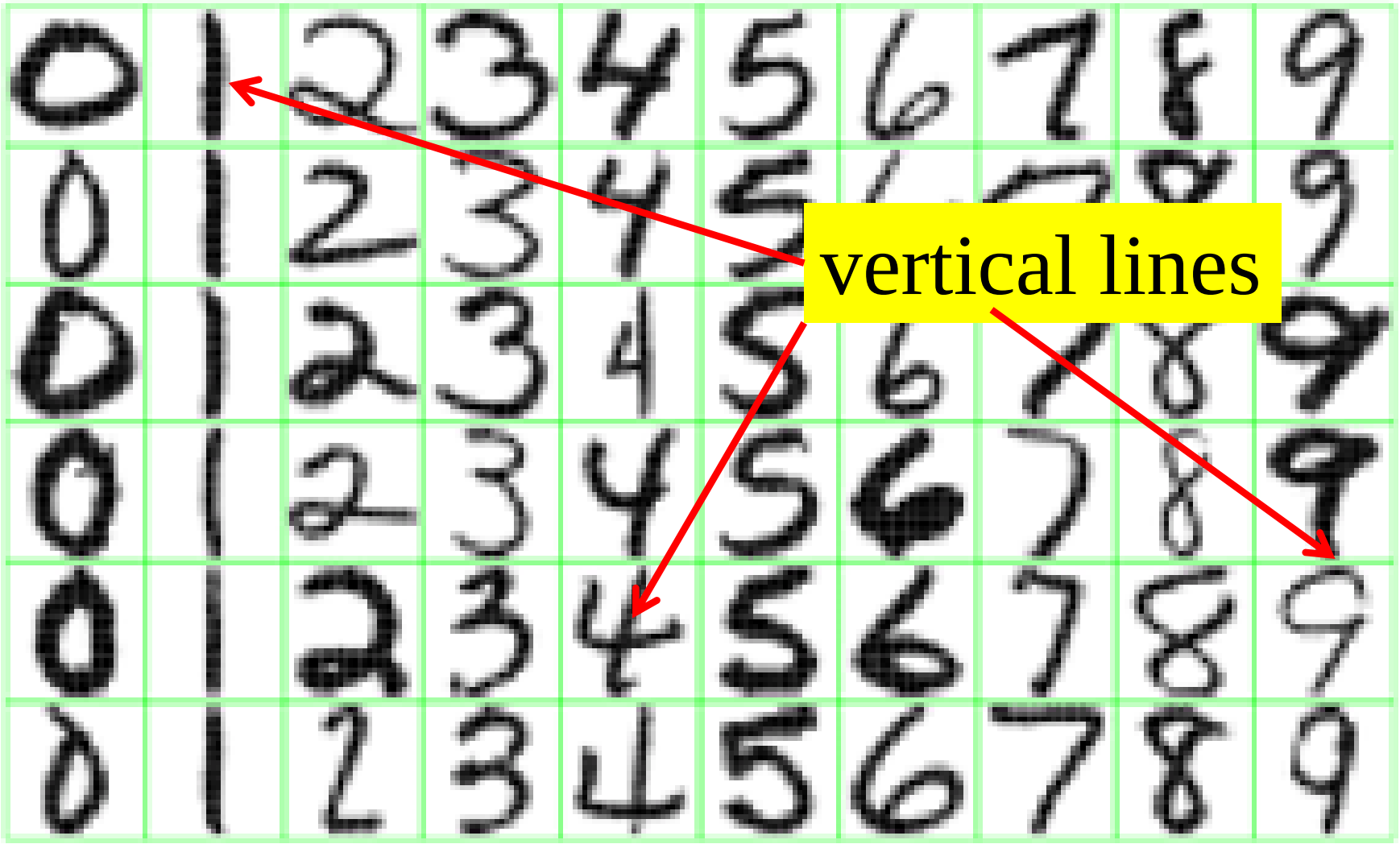


Figure 1.2: *Examples of handwritten digits from U.S. postal envelopes.*

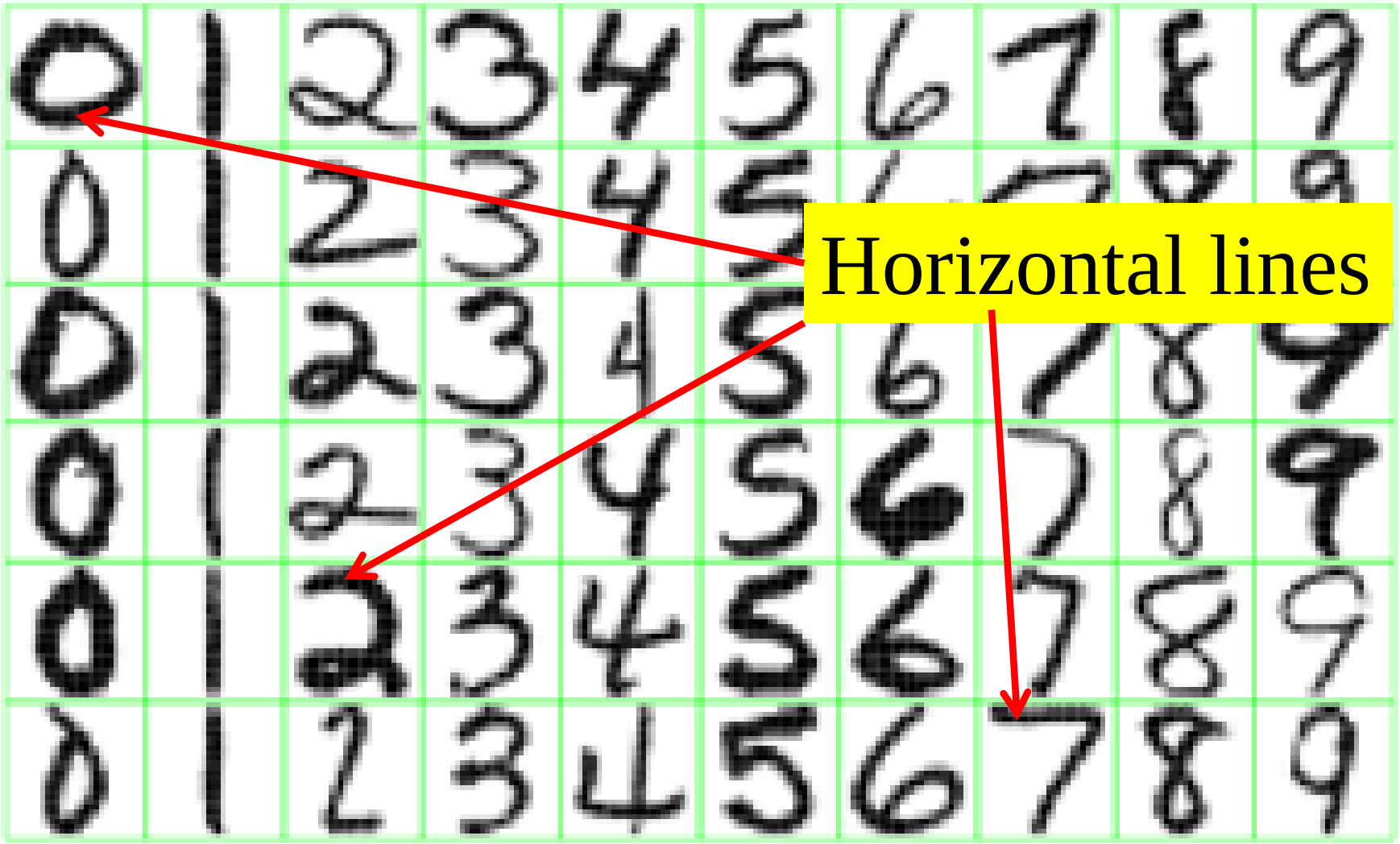


Figure 1.2: *Examples of handwritten digits from U.S. postal envelopes.*

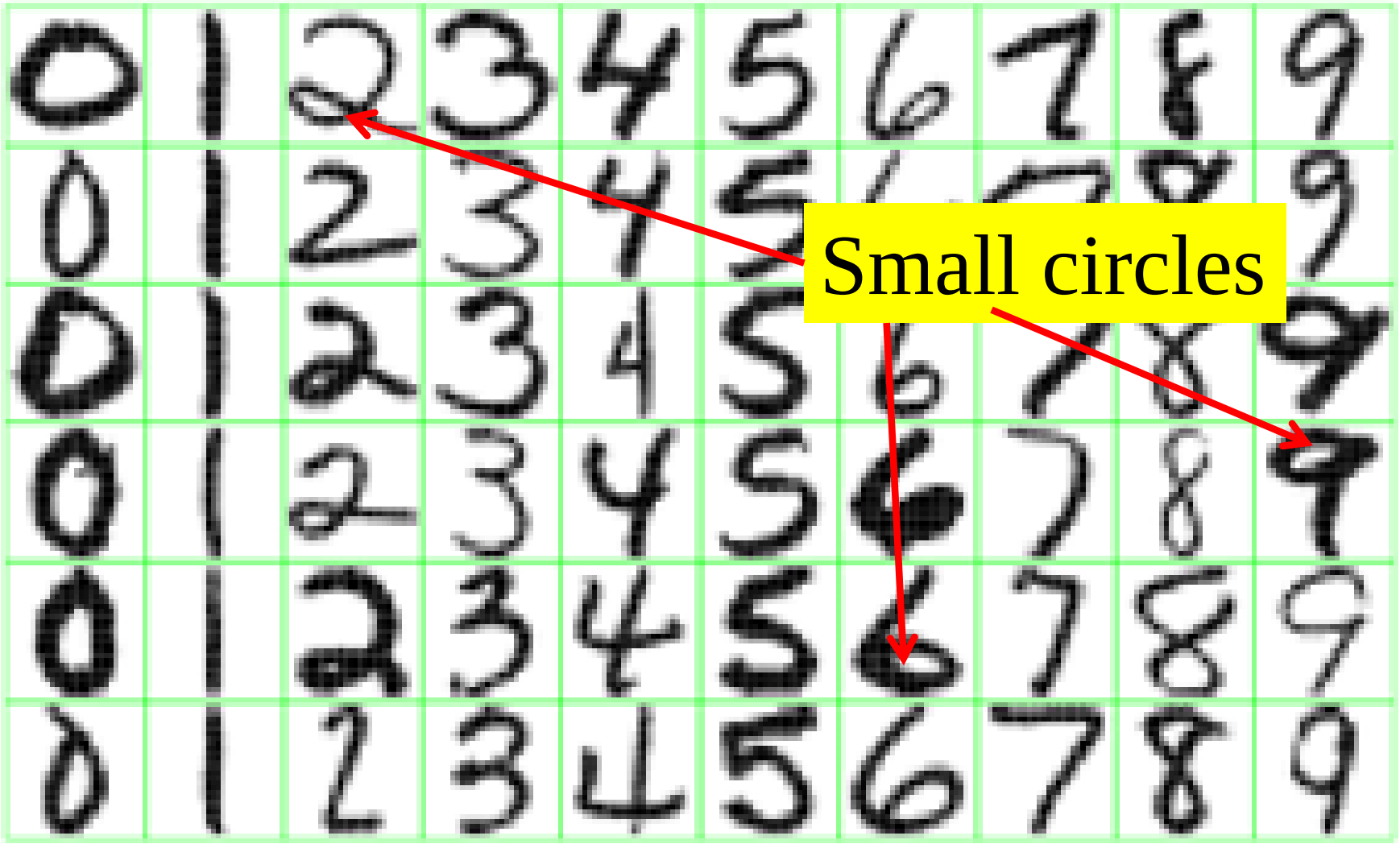
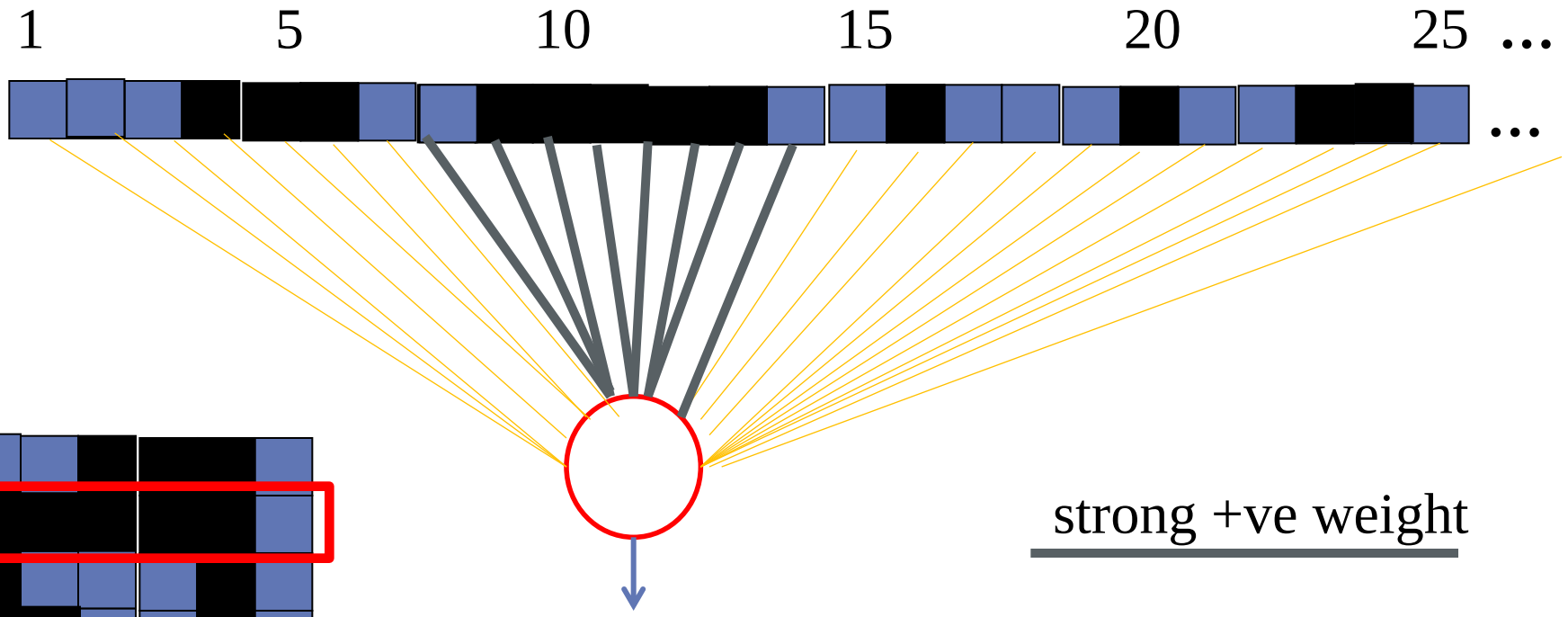
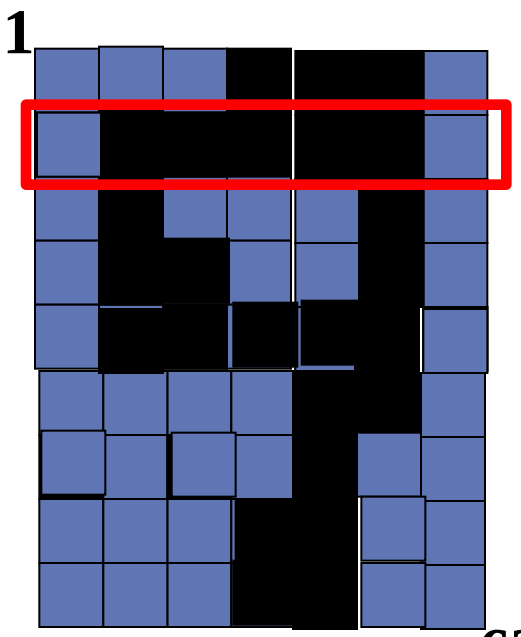


Figure 1.2: *Examples of handwritten digits from U.S. postal envelopes.*

What does this unit detect?



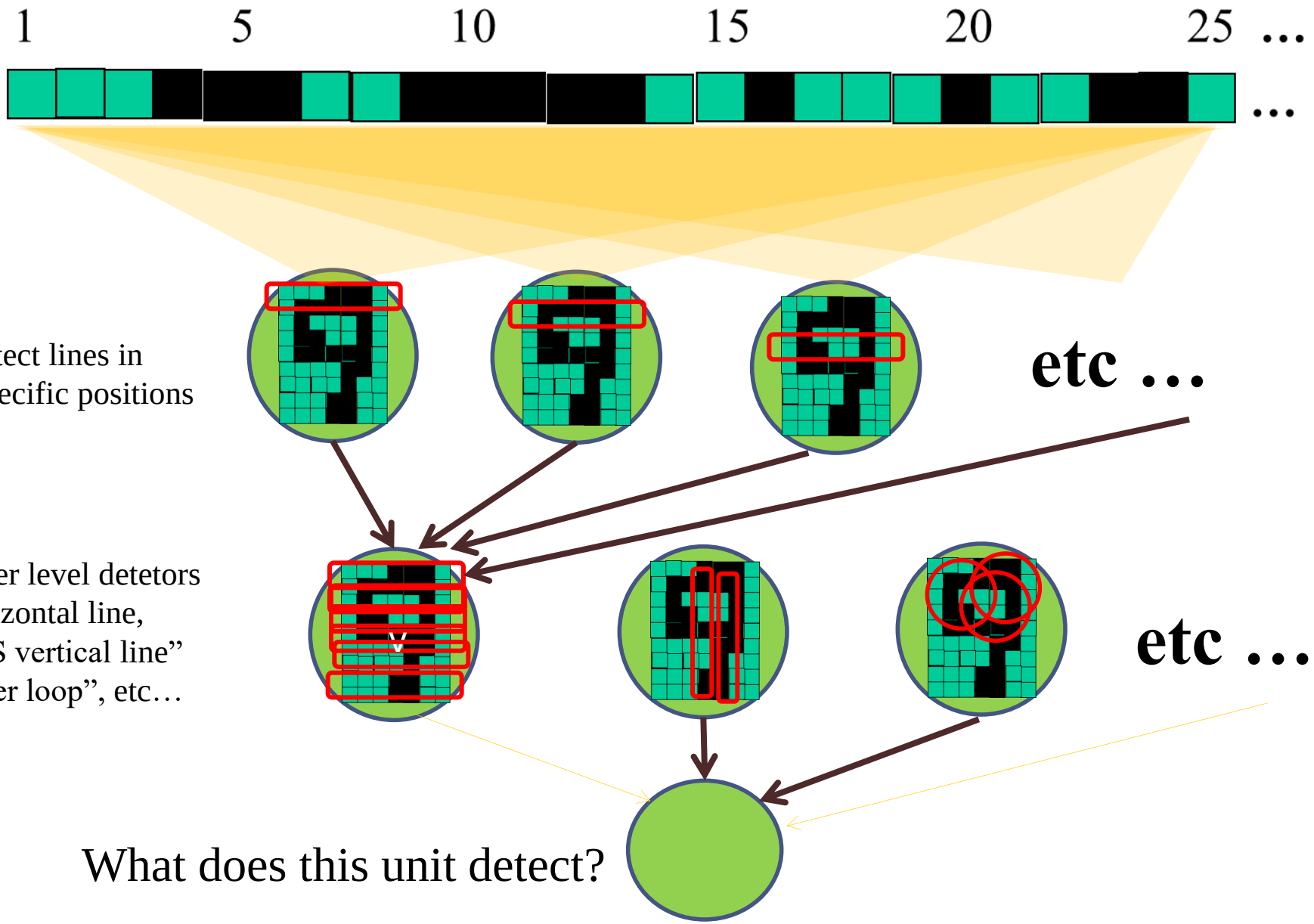


strong +ve weight

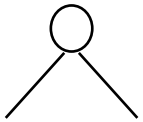
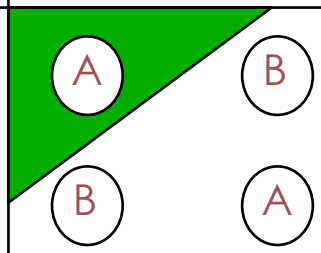
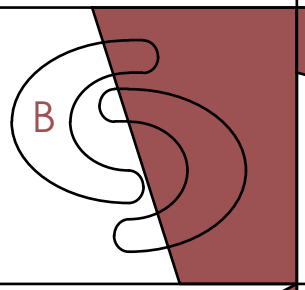
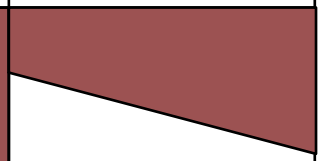
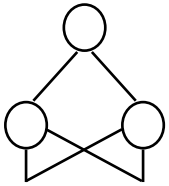
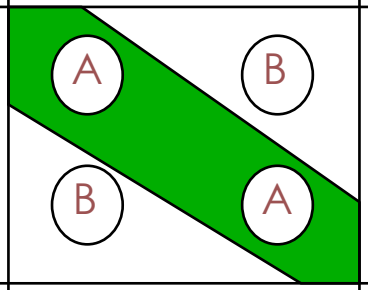
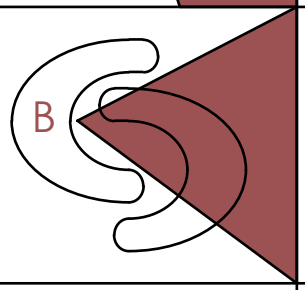
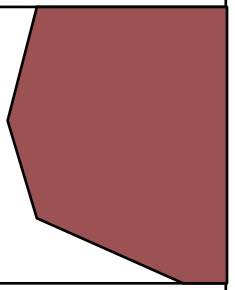
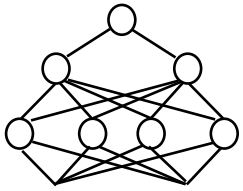
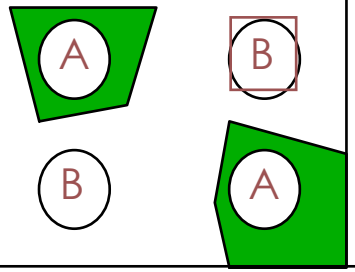
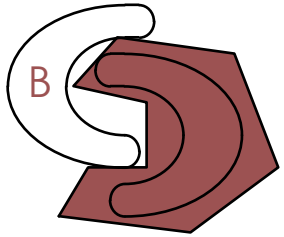
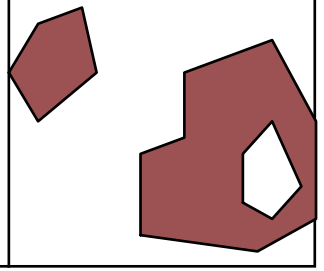
low/zero weight

it will send strong signal for a horizontal line in the SECOND row, ignoring everywhere else

successive layers can learn higher-level features …

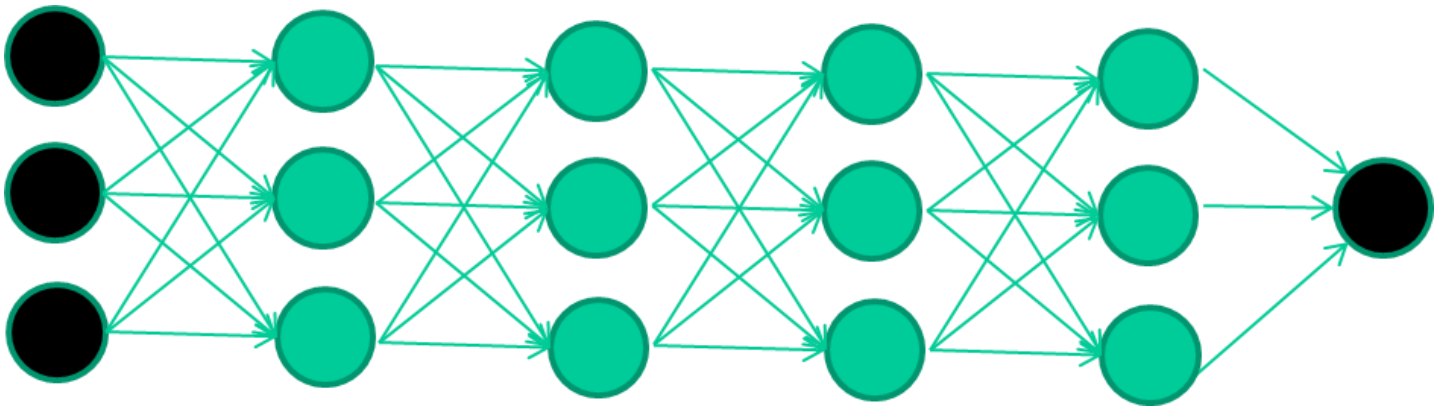


Behavior of multilayer neural networks

Structure	*Types of Decision Regions*	*Exclusive-OR Problem*	*Classes with Meshed regions*	*Most General Region Shapes*
Single-Layer 	*Half Plane Bounded By Hyper plane*			
Two-Layer 	*Convex Open Or Closed Regions*			
Three-Layer 	Arbitrary (Complexity Limited by No. of Nodes)			

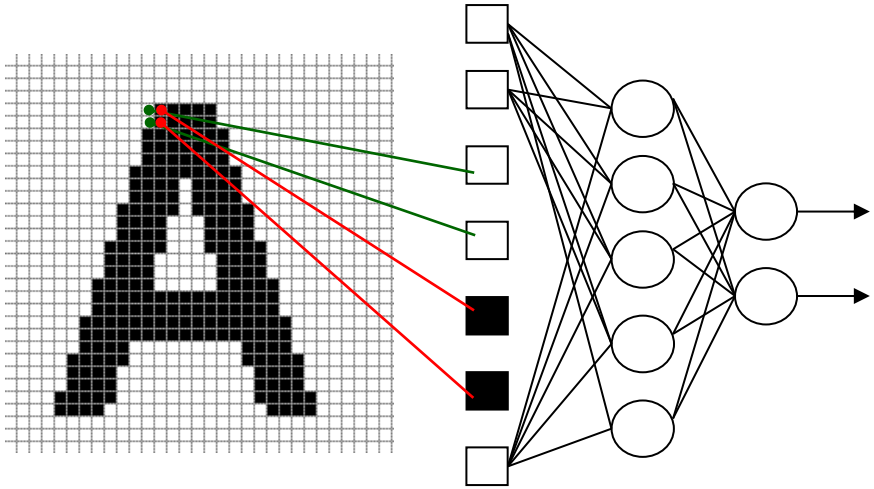
So: *multiple layers make sense*

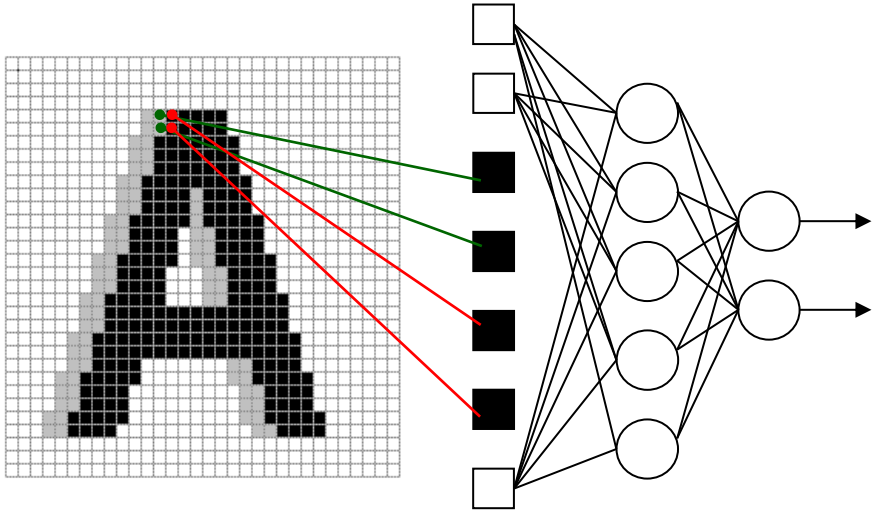
Many-layer neural network architectures should be capable of learning the true underlying features and 'feature logic', and therefore generalise very well …

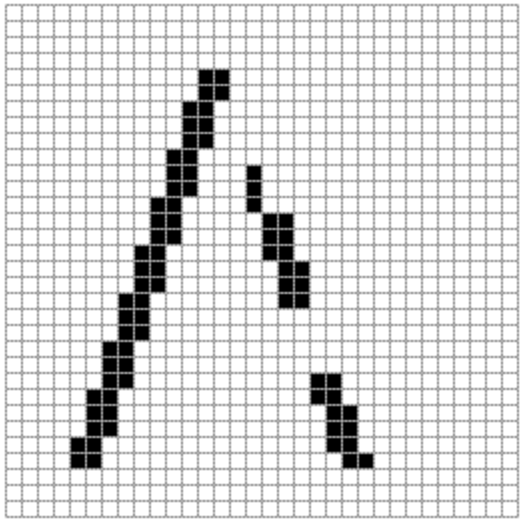


Drawbacks of previous neural networks

- Little or no invariance to shifting, scaling, and other forms of distortion







154 input change from 2 shift left
77 : black to white
77 : white to black

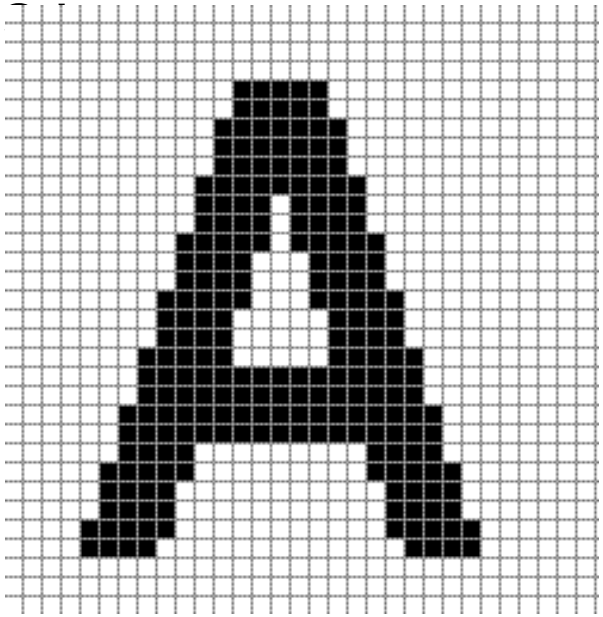
Drawbacks of previous neural networks

$$2^{32*32} = 2^{1024}$$

Black and white patterns:

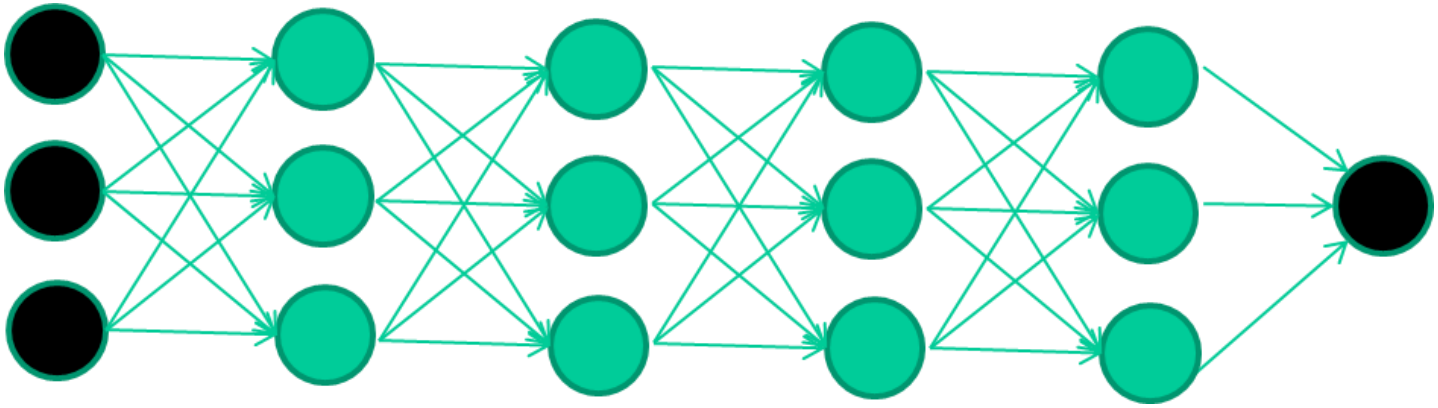
$$256^{32*32} = 256^{1024}$$

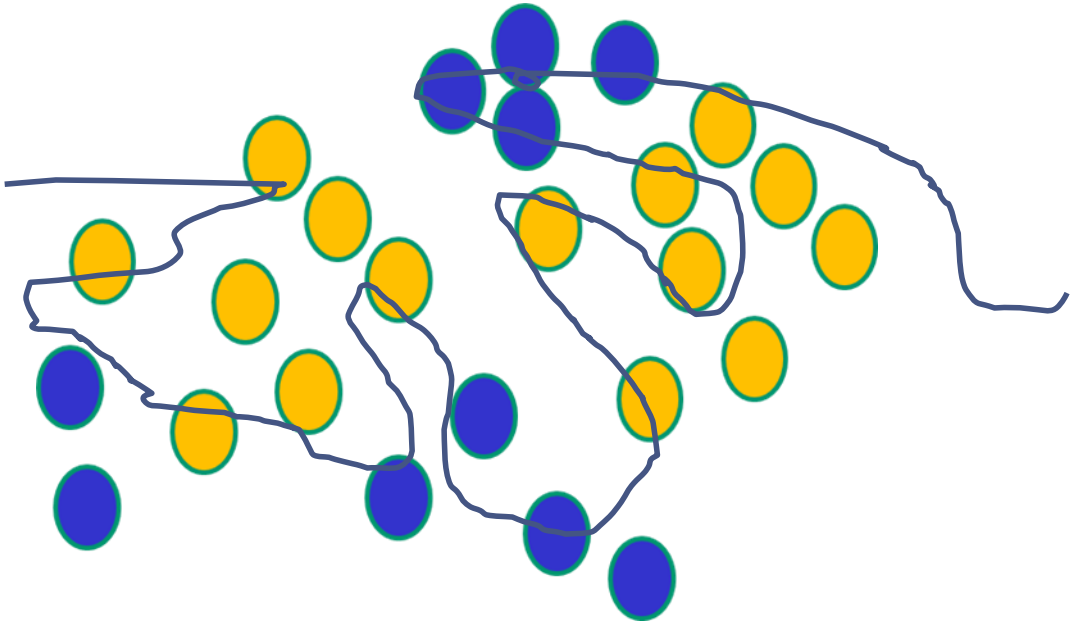
Gray scale pattern



32 * 32 input image

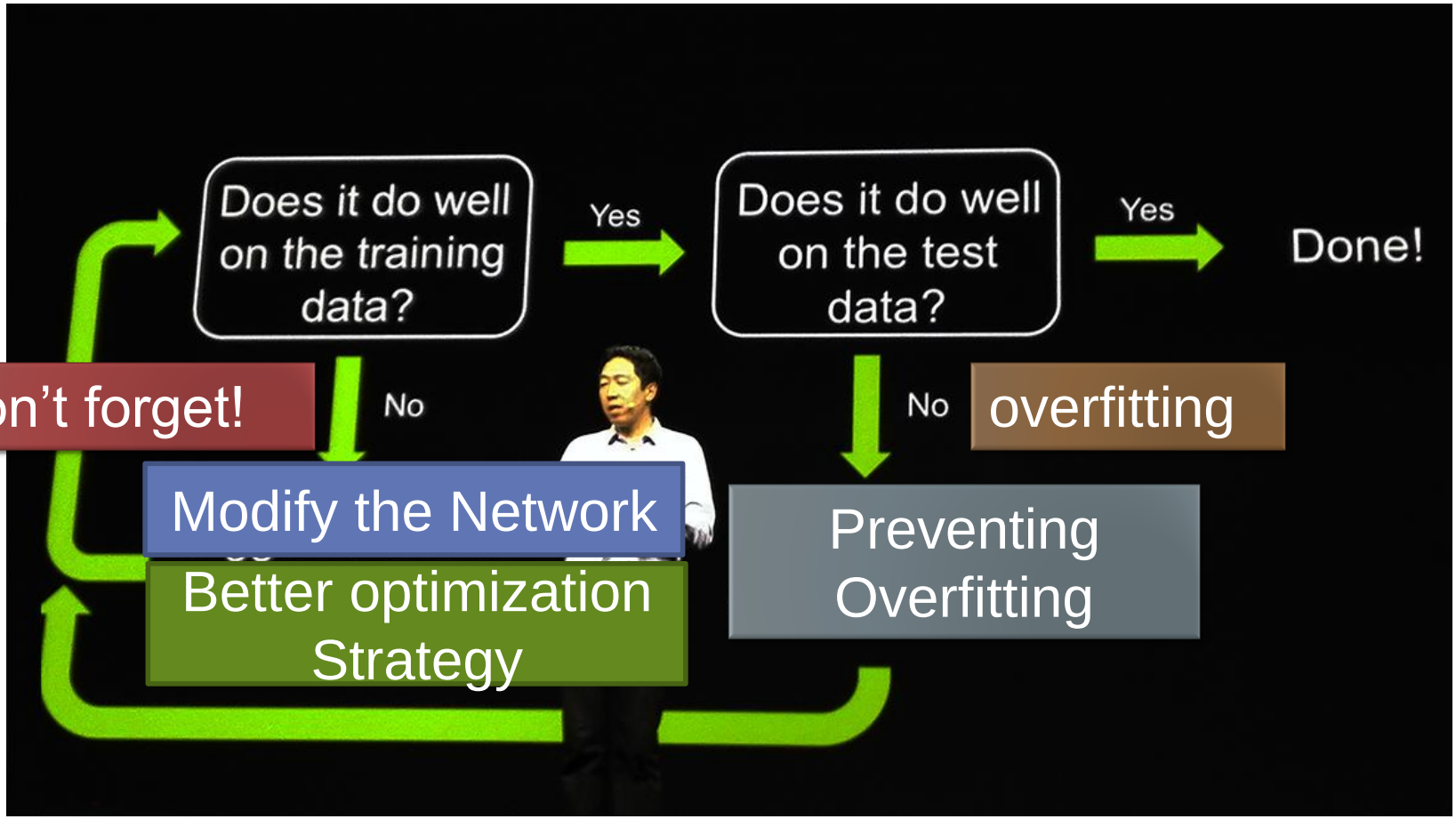
But, until very recently, our weight-learning algorithms simply did not work on multi-layer architectures





NEW WAYS TO TRAIN MULTI-LAYERED NN

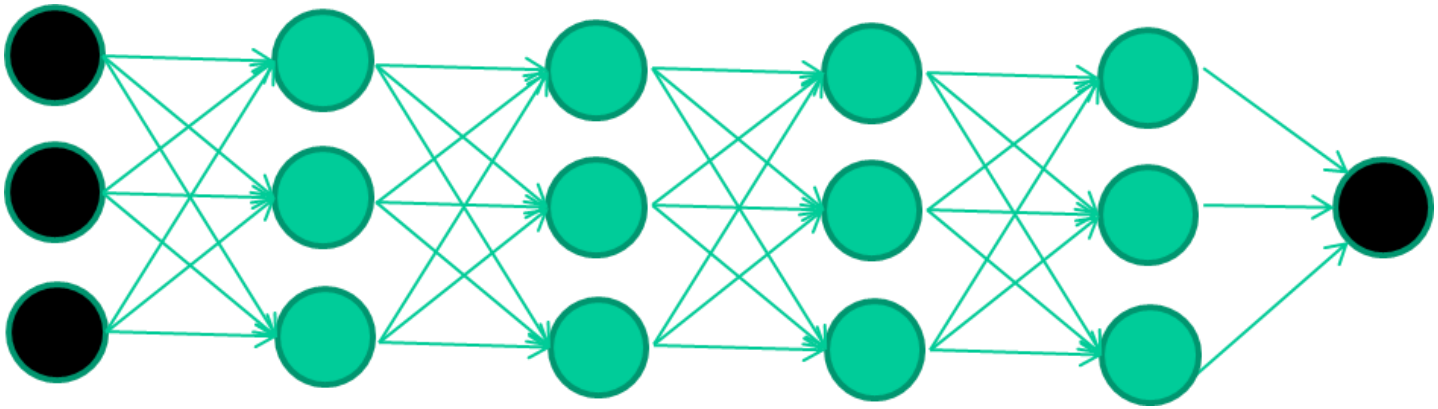
Recipe for Learning



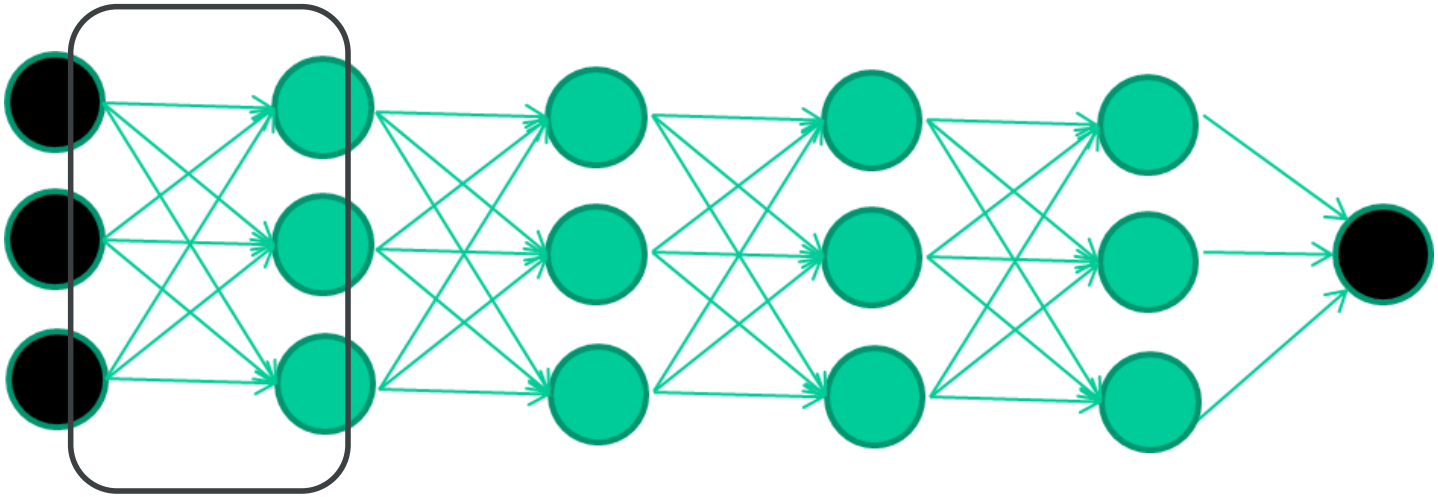
http://www.gizmodo.com.au/2015/04/the-basic-recipe-for-machine-learning-explained-in-a-single-powerpoint-slide/

Auto-Encoder

The new way to train multi-layer NNs…

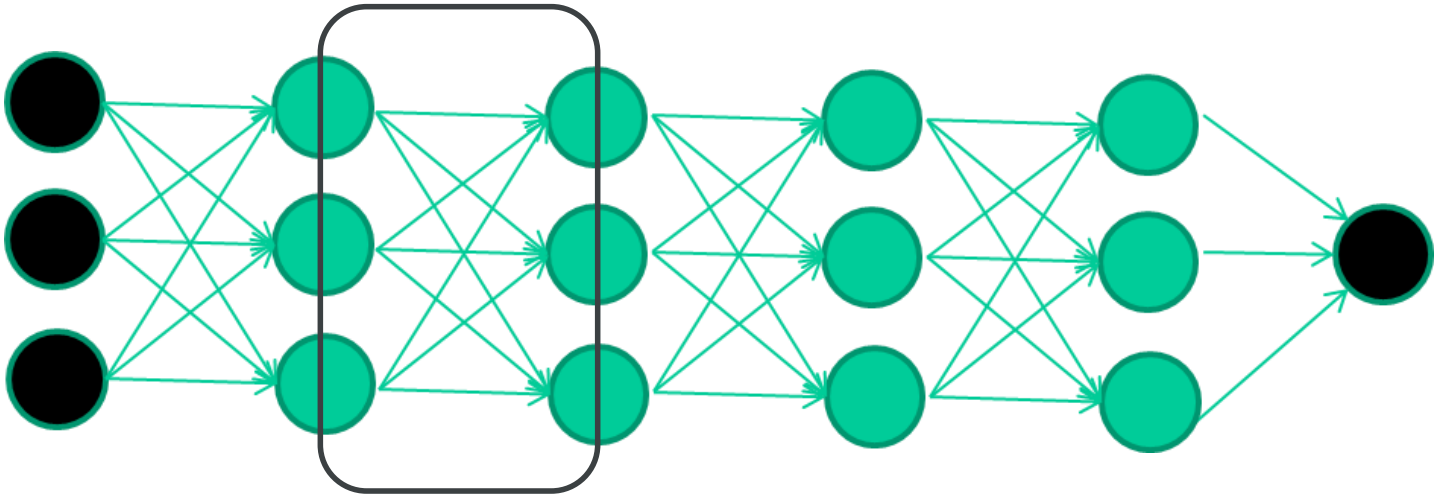


The new way to train multi-layer NNs…



Train **this** layer first

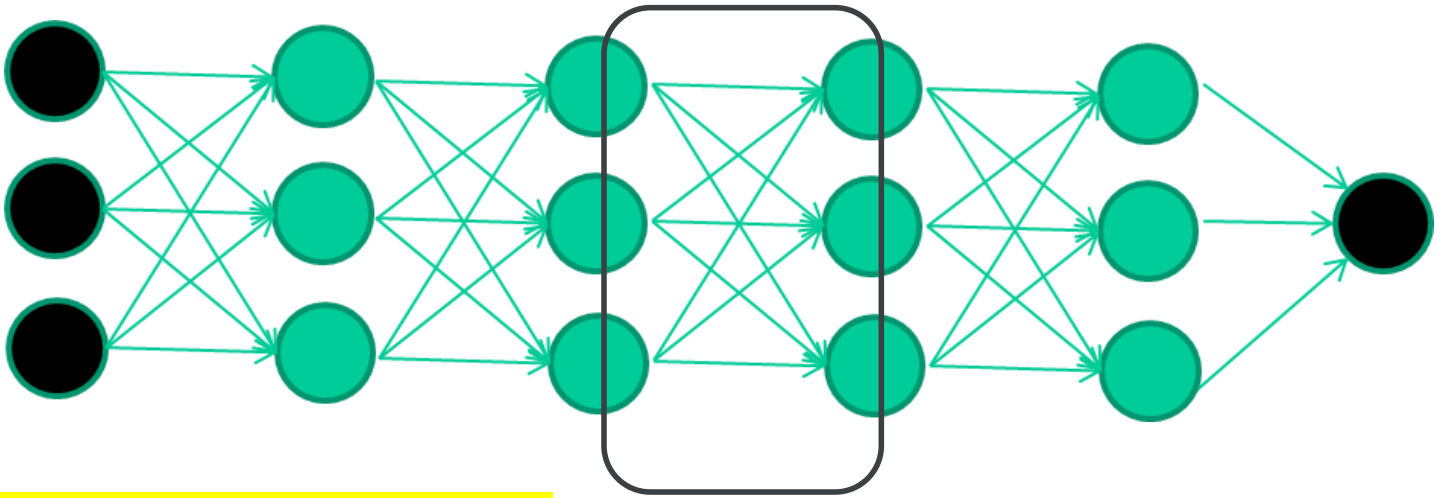
The new way to train multi-layer NNs…



Train **this** layer first

then **this** layer

The new way to train multi-layer NNs…

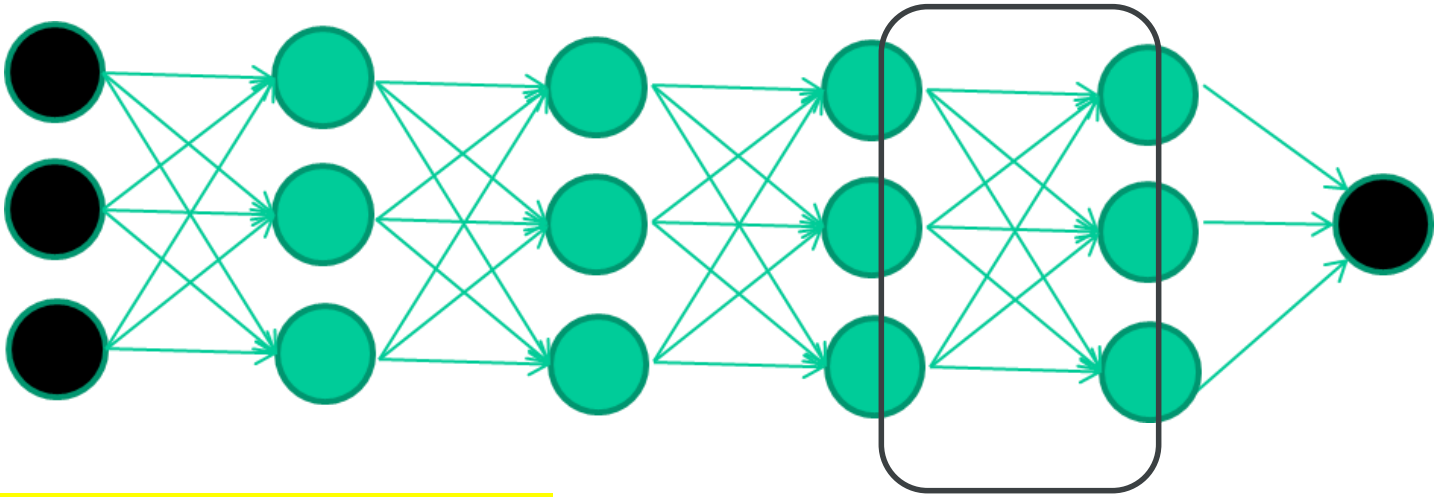


Train **this** layer first

then **this** layer

then **this** layer

The new way to train multi-layer NNs…



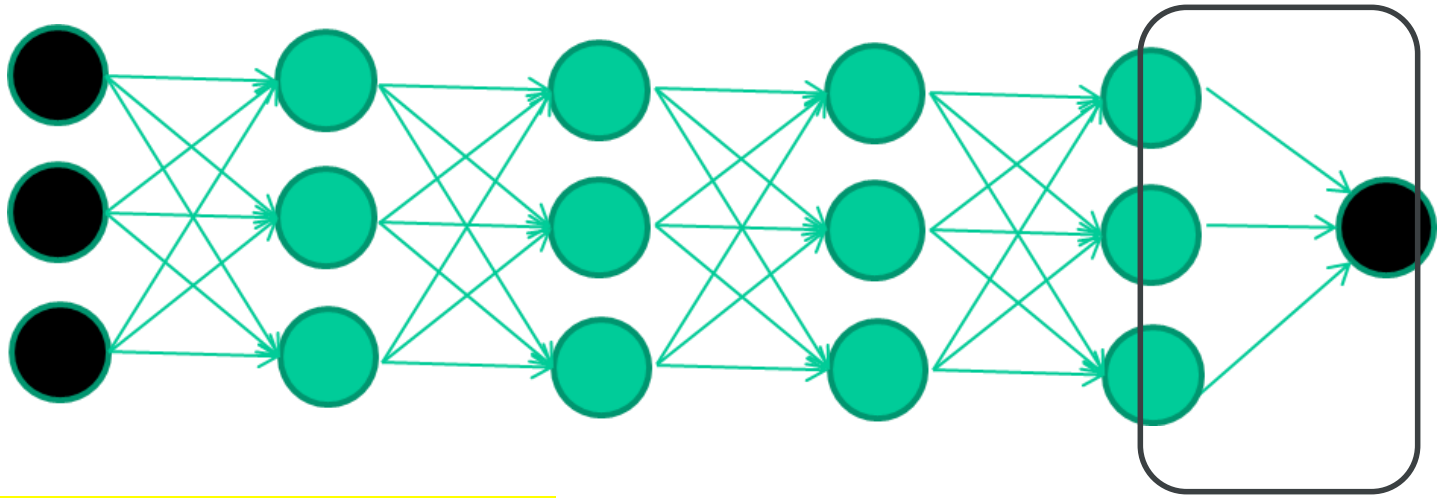
Train **this** layer first

then **this** layer

then **this** laver

then **this** layer

The new way to train multi-layer NNs…



Train **this** layer first

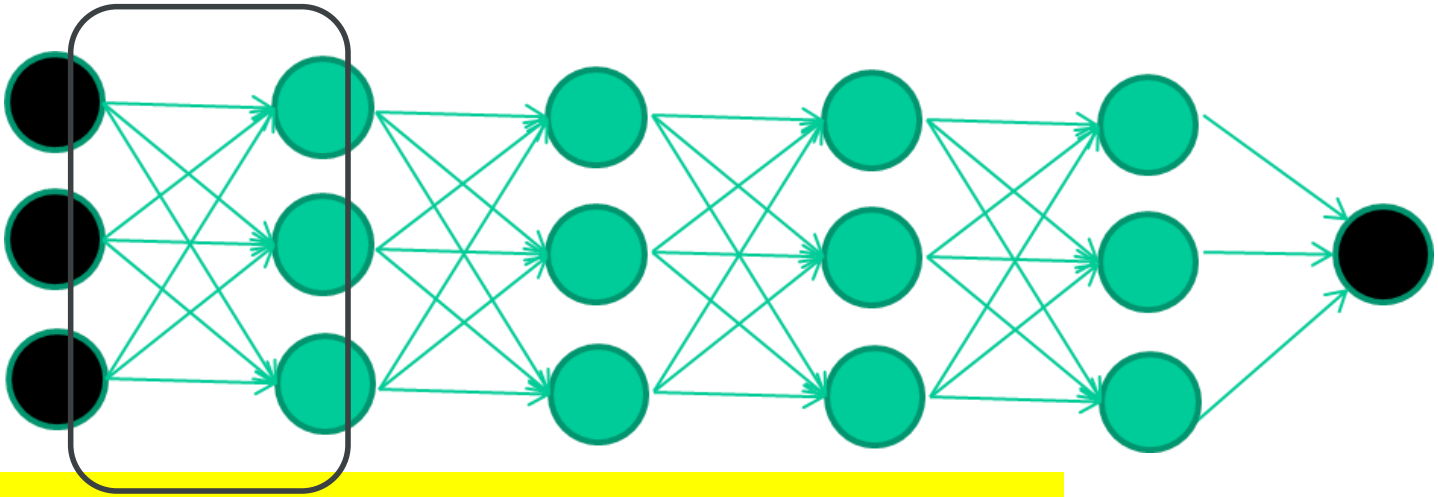
then **this** layer

then **this** laver

then **this** laver

finally **this** layer

The new way to train multi-layer NNs…



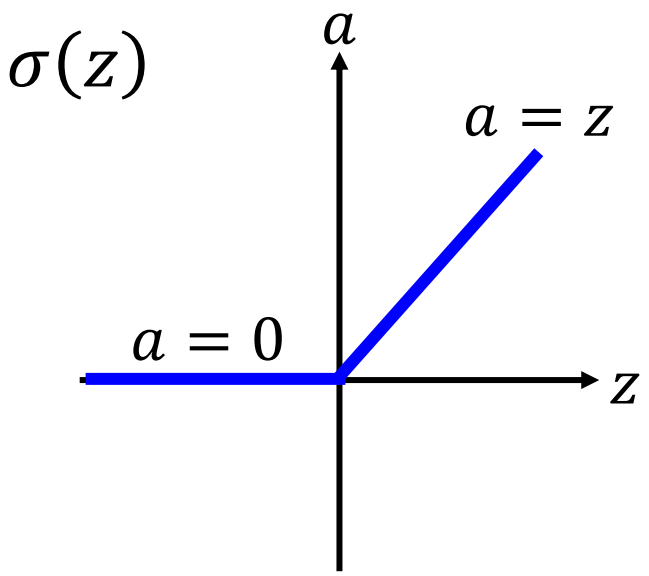
EACH of the (non-output) layers is trained to be an ***auto-encoder***

Basically, it is forced to learn good features that describe what comes from the previous layer

New Activation Functions

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

Vanishing Gradient Problem

In 2006, people used RBM pre-training.

In 2015, people use ReLU.

Smaller gradients

Learn very slow

Almost random

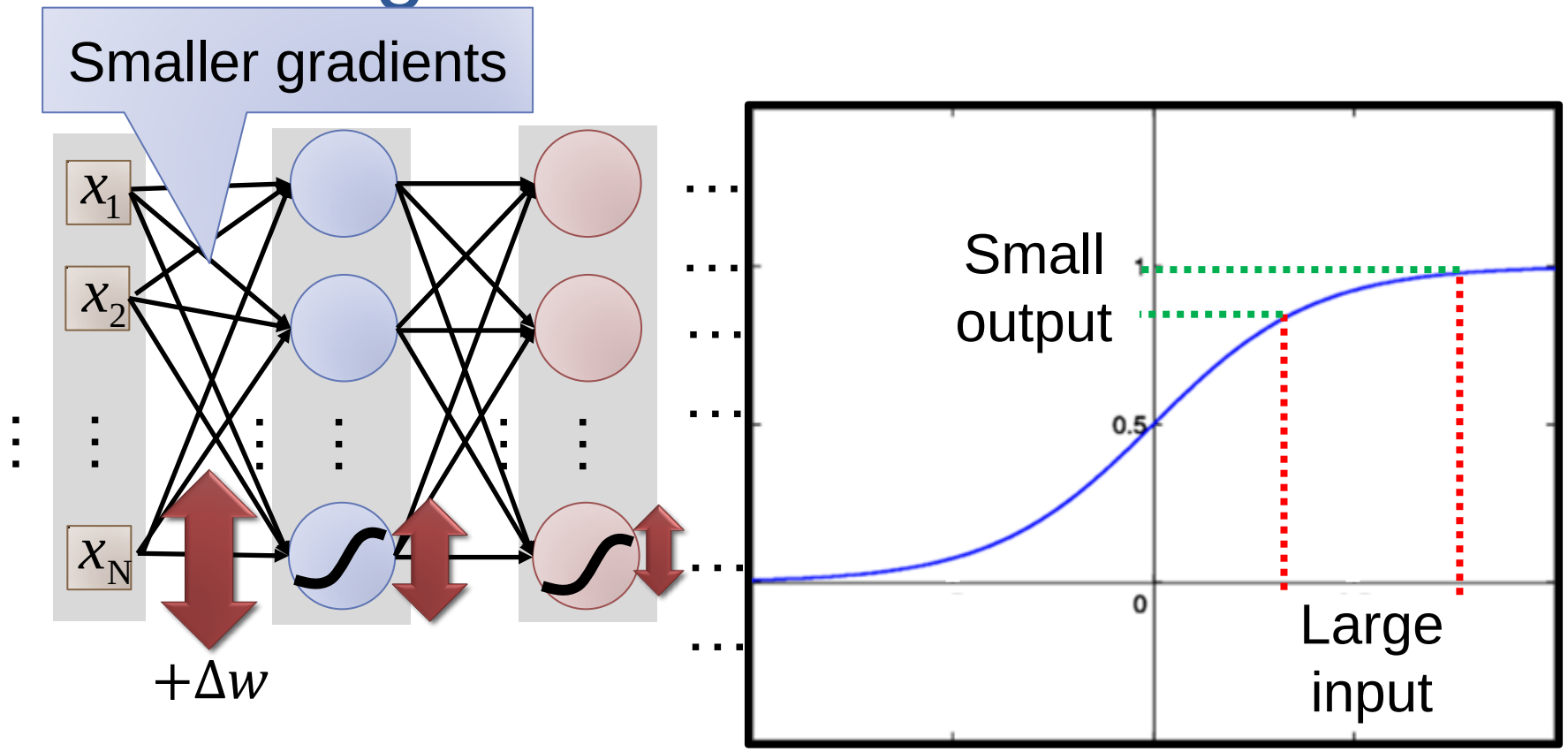
Larger gradients

Learn very fast

Already converge

based on randomness?

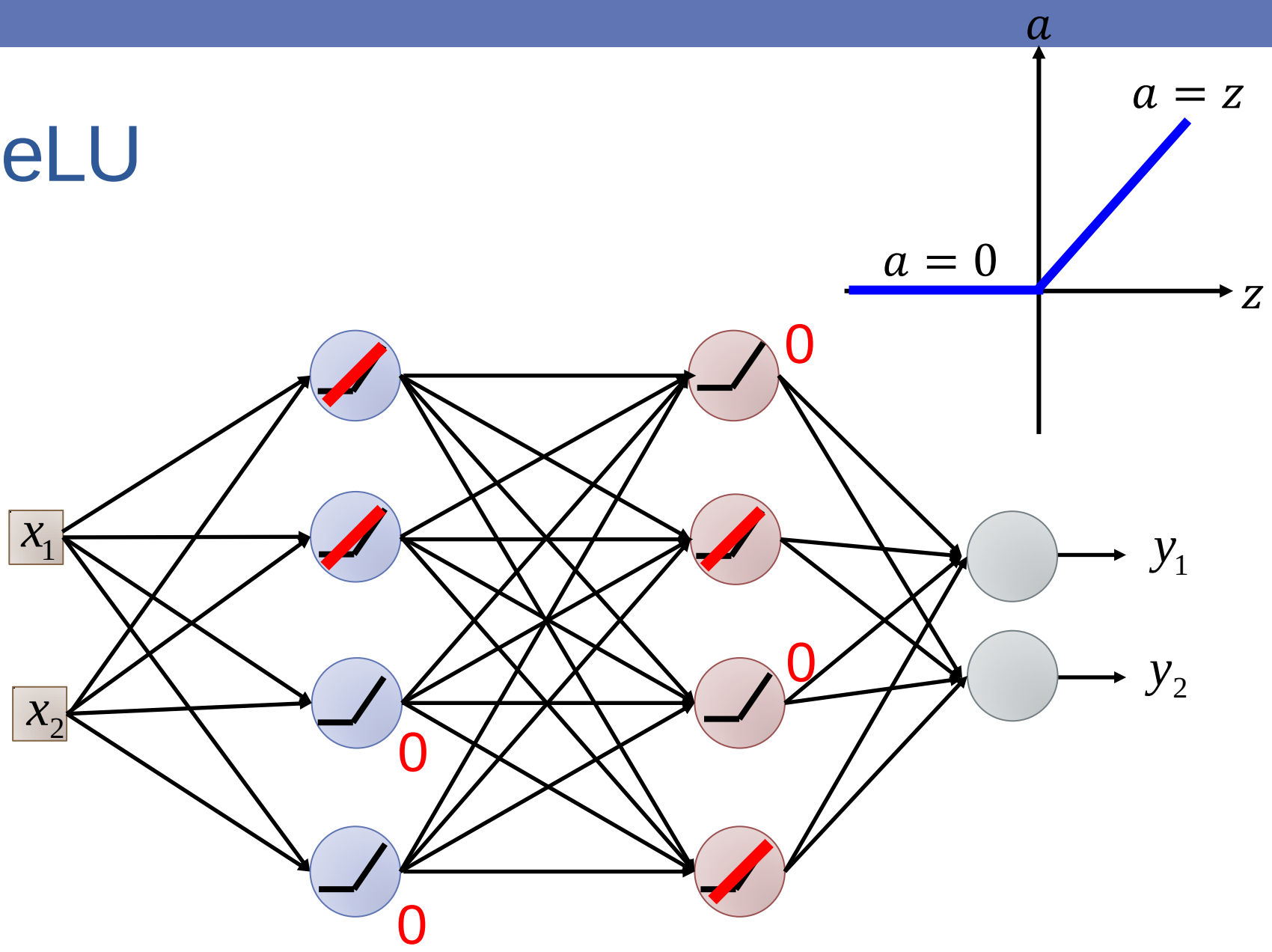
Vanishing Gradient Problem



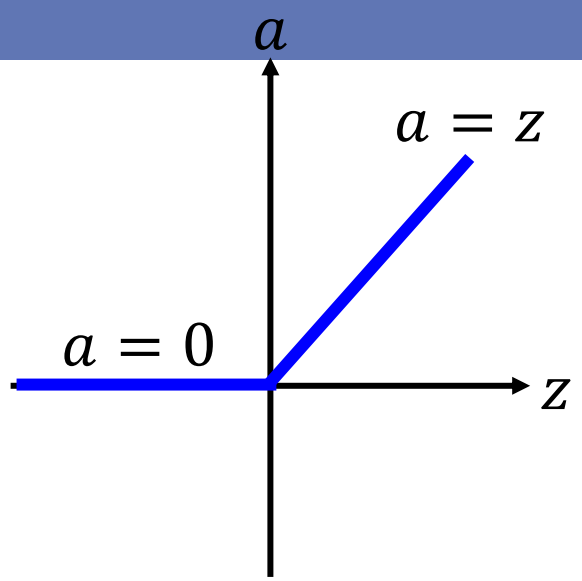
Intuitive way to compute the gradient …

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

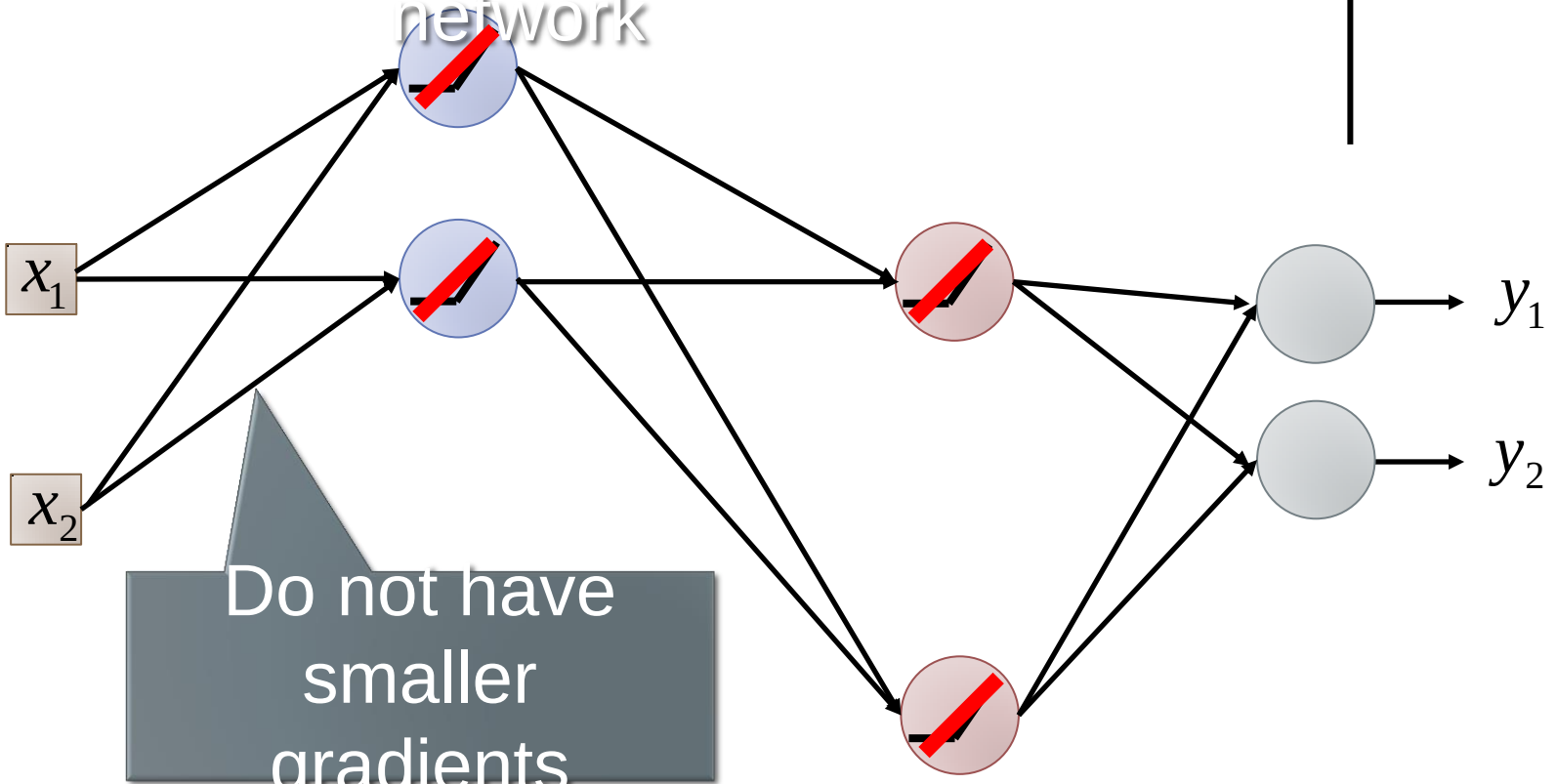
ReLU



ReLU



A Thinner linear network

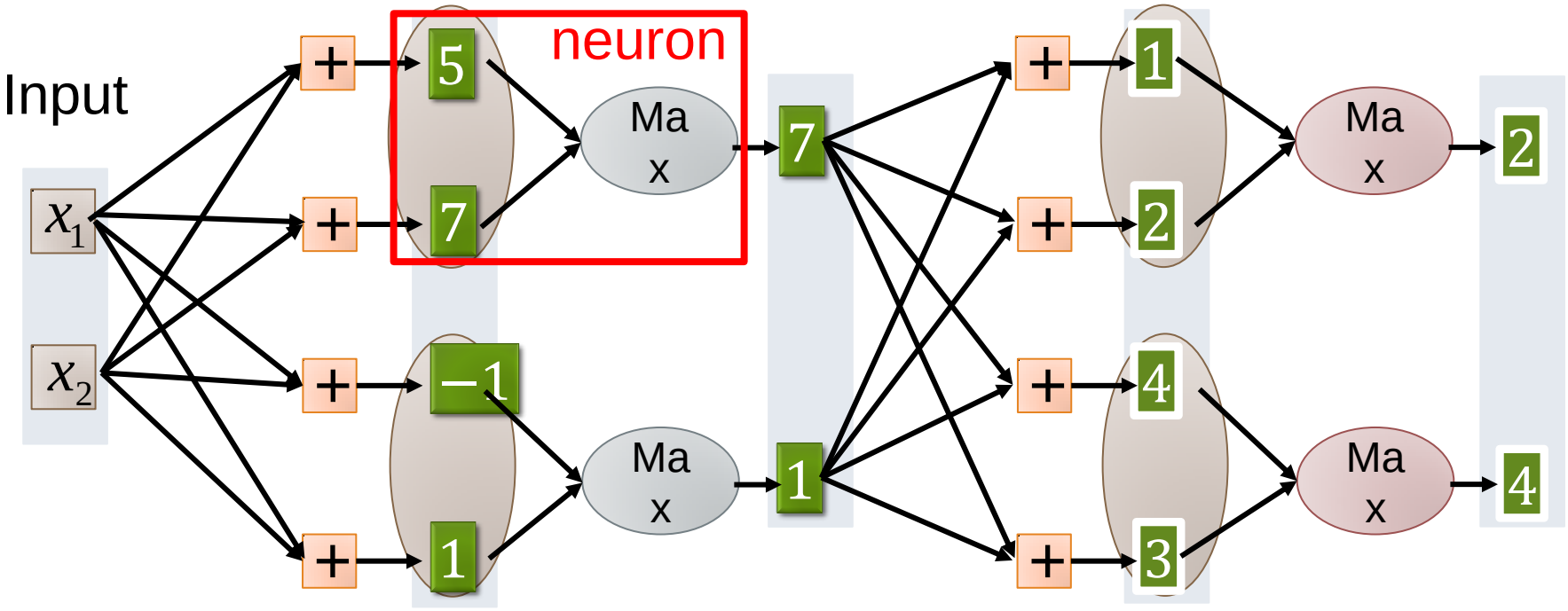


Do not have smaller gradients

Maxout

ReLU is a special cases of Maxout

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



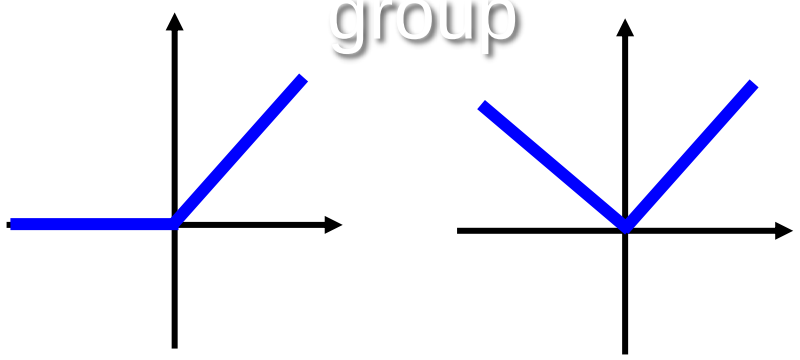
You can have more than 2 elements in a group.

Maxout

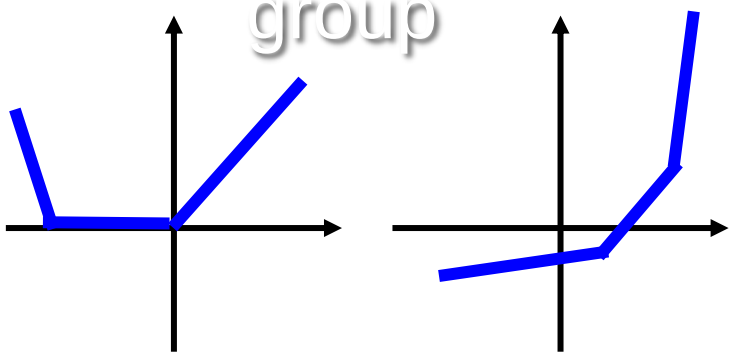
ReLU is a special cases of Maxout

- Learnable activation function [Ian J. Goodfellow, ICML'13]
  - Activation function in maxout network can be any piecewise linear convex function
  - How many pieces depending on how many elements in a group

2 elements in a group

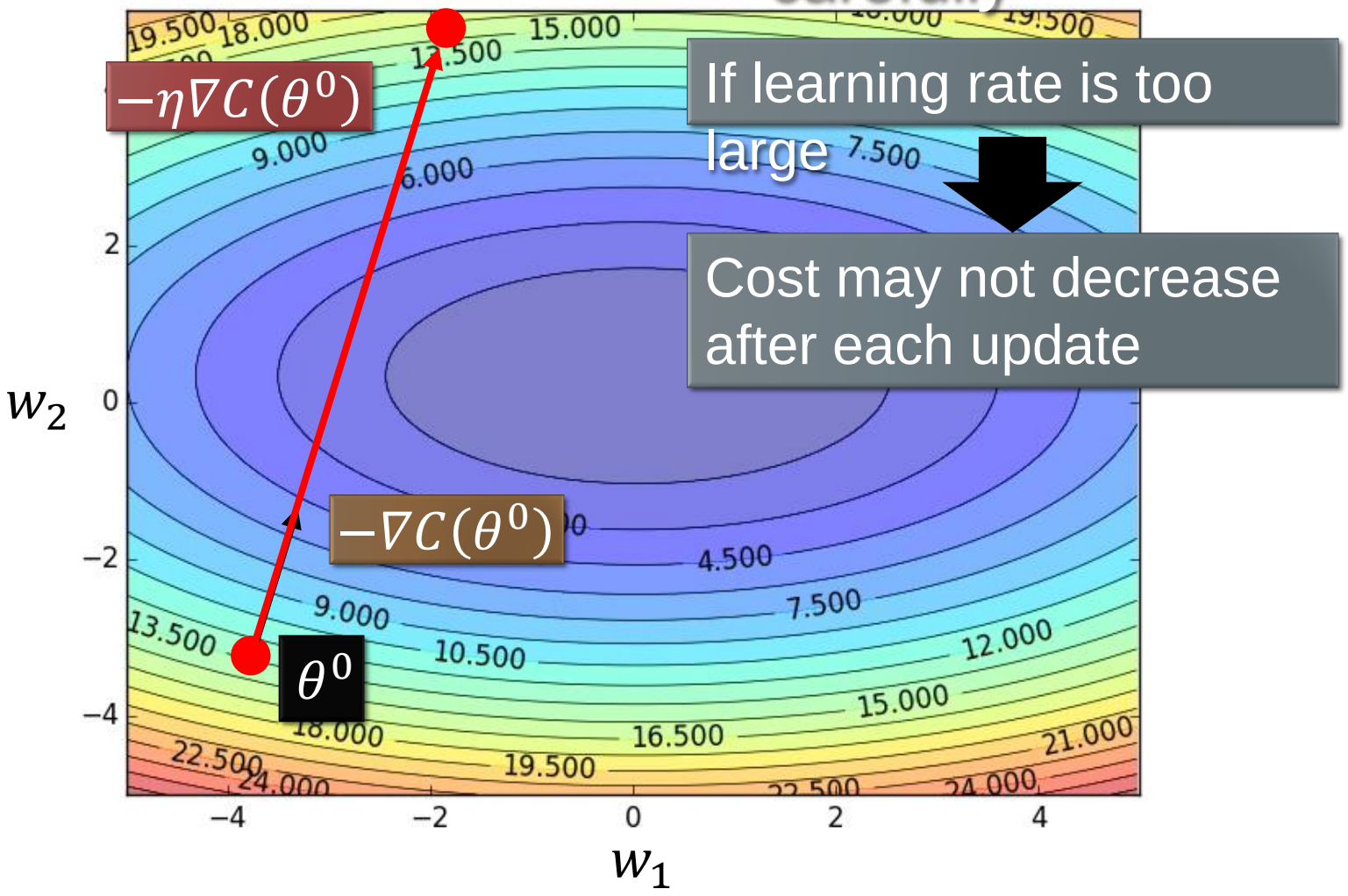


3 elements in a group



Learning Rate

Set the learning rate η carefully



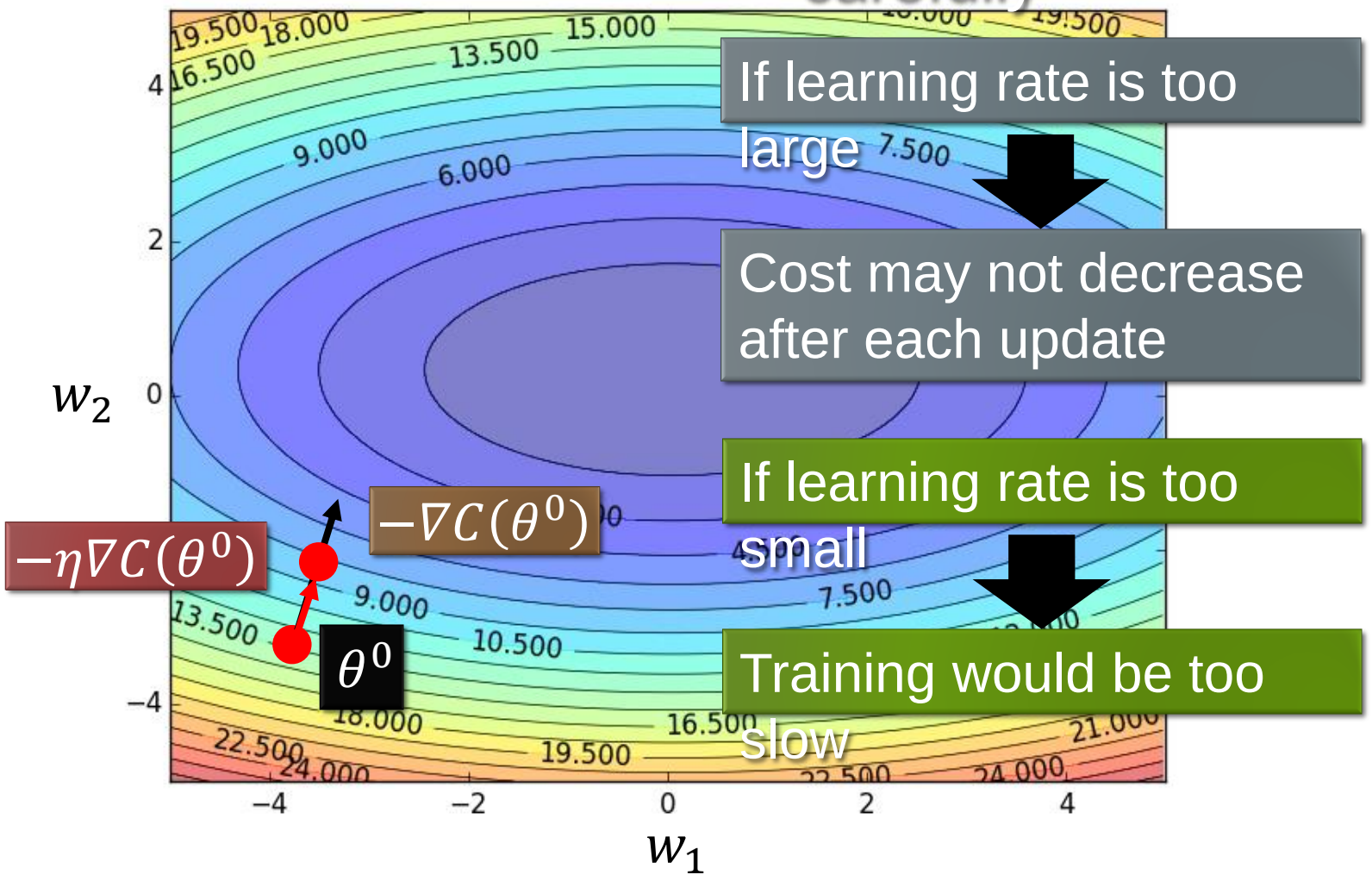
If learning rate is too large

Cost may not decrease after each update

Learning Rate

Can we give different parameters different learning rates?

Carefully



If learning rate is too large

Cost may not decrease after each update

If learning rate is too small

Training would be too slow

Adagrad

Original Gradient Descent

$$\theta^t \leftarrow \theta^{t-1} - \eta \nabla C(\theta^{t-1})$$

Each parameter w are considered separately

$$w^{t+1} \leftarrow w^t - \eta_w g^t \qquad g^t = \frac{\partial C(\theta^t)}{\partial w}$$

Parameter dependent learning rate

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

constant

Summation of the square of the previous derivatives

Adagrad

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

	g^0
w_1	0.1

	g^0
w_2	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:

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

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

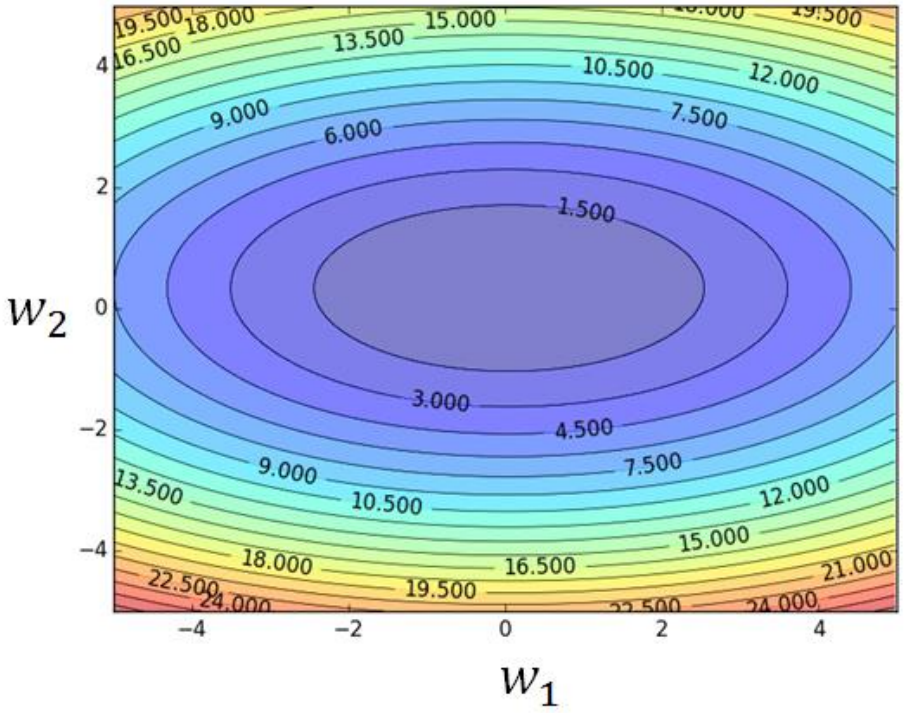
Observation:

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, large learning rate, and vice versa

Why?

Recipe for Learning

Modify the Network

- New activation functions, for example, ReLU or Maxout

Better optimization Strategy

- Adaptive learning rates

Prevent Overfitting

- Dropout

Only use this approach when you already obtained good results on the training data.

Deep learning methods

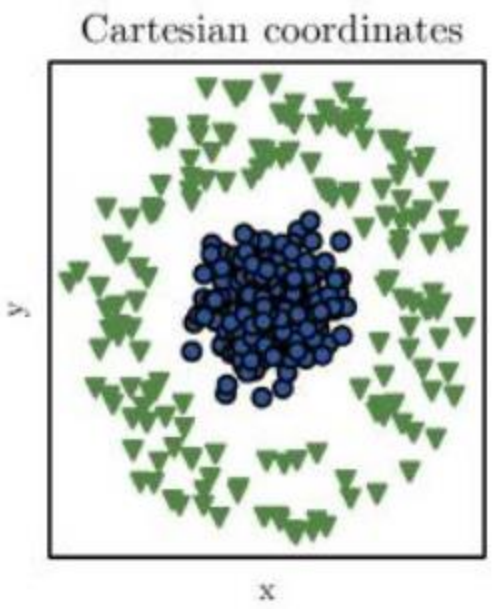
- Unsupervised Methods •
  - Restricted Boltzmann Machines •
  - Deep Belief Networks •
  - Auto encoders: unsupervised feature extraction/learning

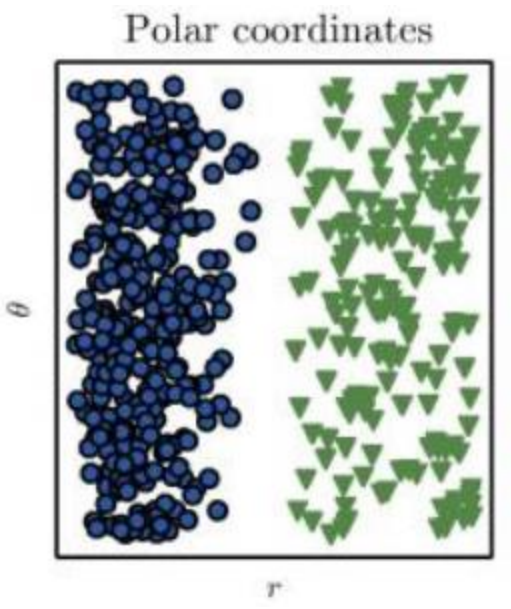
- Supervised Methods •
  - Deep Neural Networks •
  - Recurrent Neural Networks •
  - Convolutional Neural Networks

  - Flownet: From image to caption
  - Google's cat detection neural network http://www.resnap.com/imageselection-technology/deep-learning-image-classification/ • Example auto-encoder : http://nghiaho.com/?p=1765 • SGD : http://blog.datumbox.com/tuning-the-learning-rate-in-gradientdescent/

Representation Learning

- Auto-encoders: Encoder+ Decoder
  - Separate out the factors of variation/ identify the constructors
  - Example: Identify a car-position, color, wheels, disentangle free silhoutte





Not the whole story ……

- Adagrad [John Duchi, JMLR'11]
- RMSprop
  - https://www.youtube.com/watch?v=O3sxAc4hxZU
- Adadelta [Matthew D. Zeiler, arXiv'12]
- Adam [Diederik P. Kingma, ICLR'15]
- AdaSecant [Caglar Gulcehre, arXiv'14]
- "No more pesky learning rates" [Tom Schaul, arXiv'12]

PART III:

TIPS FOR TRAINING DNN

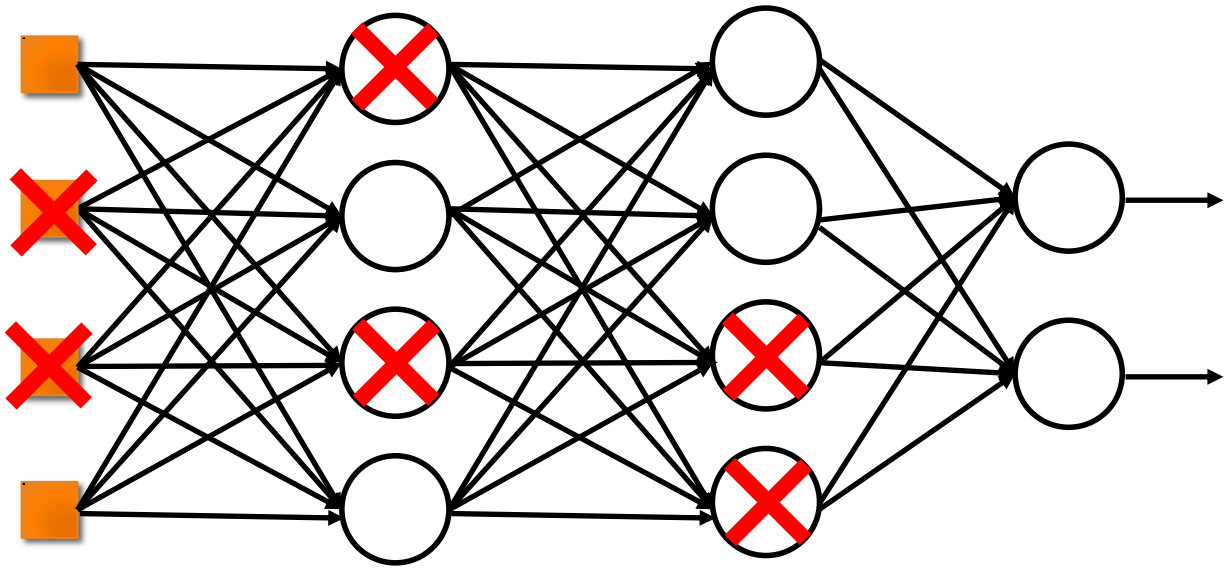
Dropout

Dropout

Pick a mini-batch

$$\theta^t \leftarrow \theta^{t-1} - \eta \nabla C(\theta^{t-1})$$

Training:



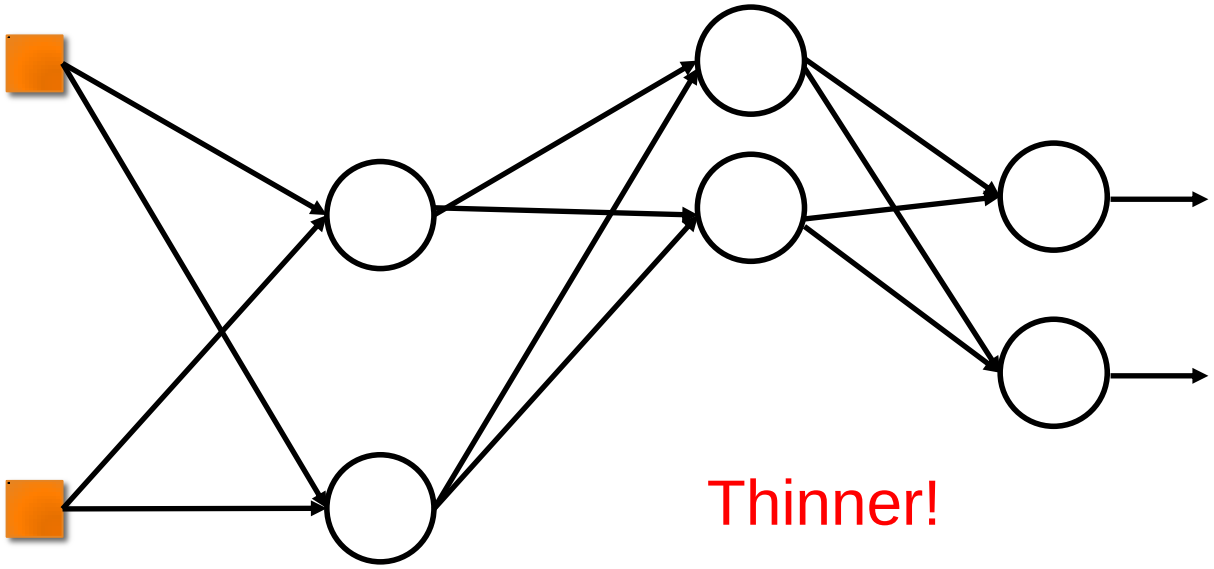
- **Each time before computing the gradients**
  - Each neuron has p% to dropout

Dropout

Pick a mini-batch

$$\theta^t \leftarrow \theta^{t-1} - \eta \nabla C(\theta^{t-1})$$

Training:



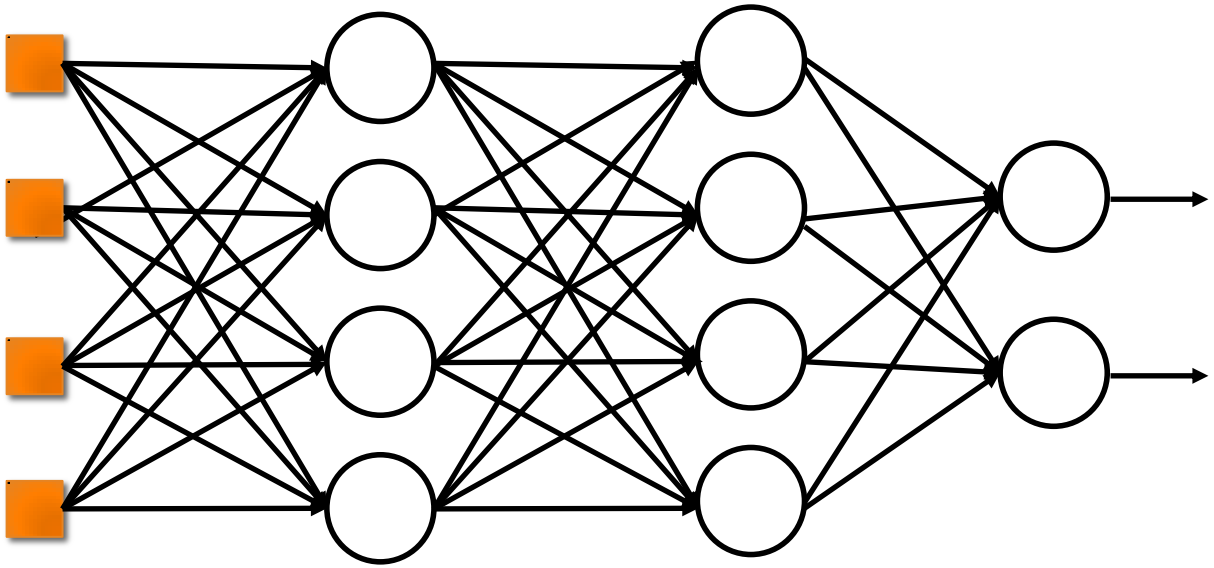
- **Each time before computing the gradients**
  - Each neuron has p% to dropout

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

For each mini-batch, we resample the dropout neurons

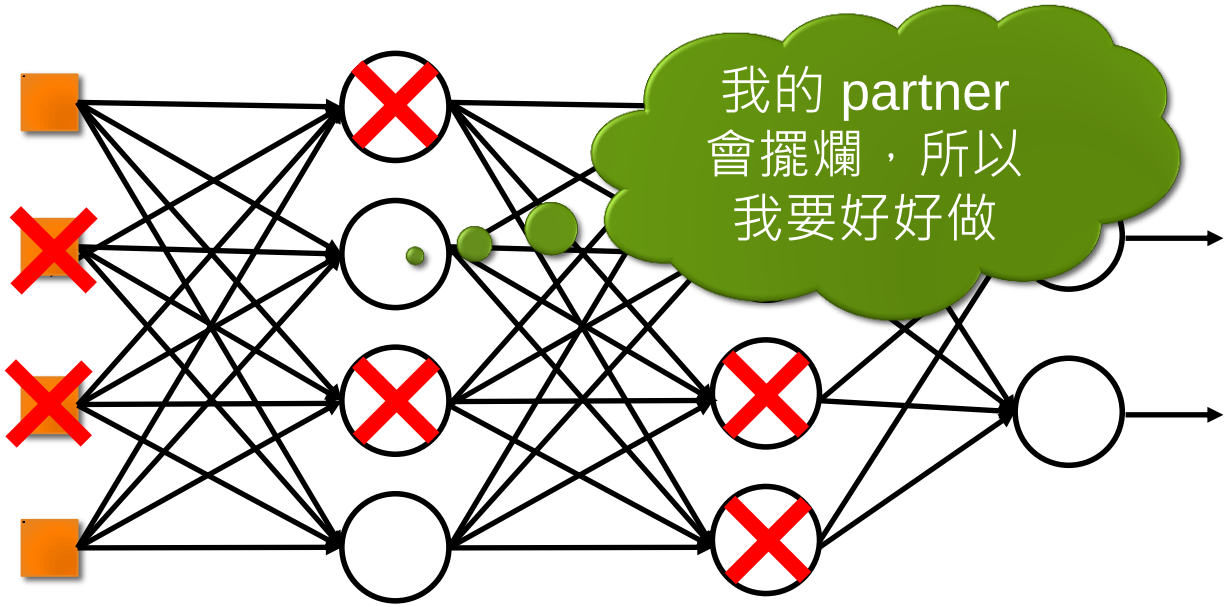
Dropout

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



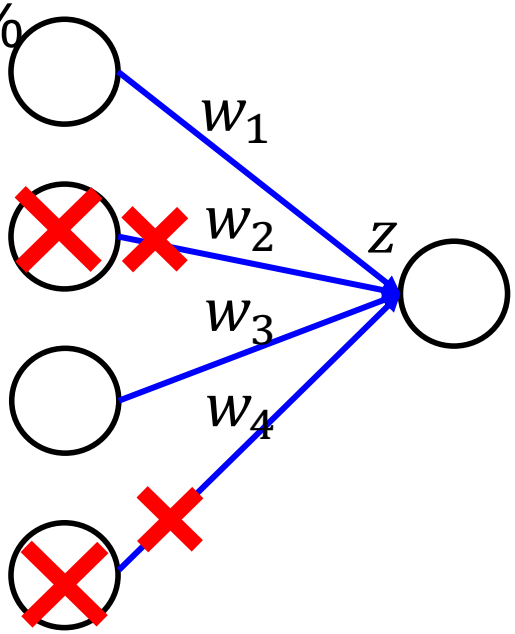
- When teams up, if everyone expect the partner will do the work, nothing will be done finally.
- However, if you know your partner will dropout, you will do better.
- When testing, no one dropout actually, so obtaining good results eventually.

Dropout - Intuitive Reason

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

Training of Dropout

Assume dropout rate is 50%

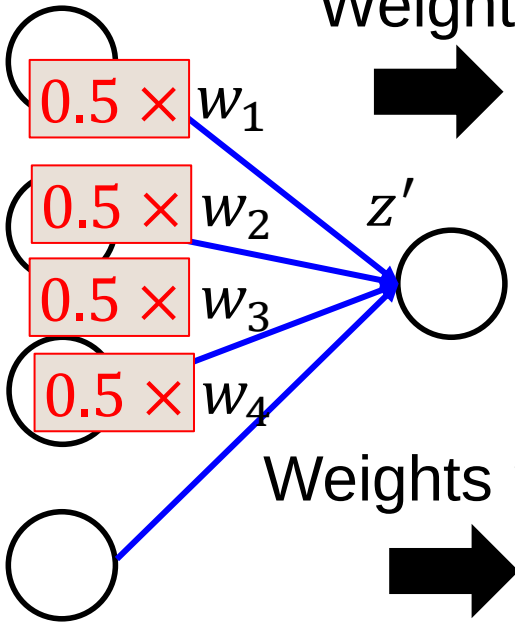


Testing of Dropout

No dropout

Weights from training

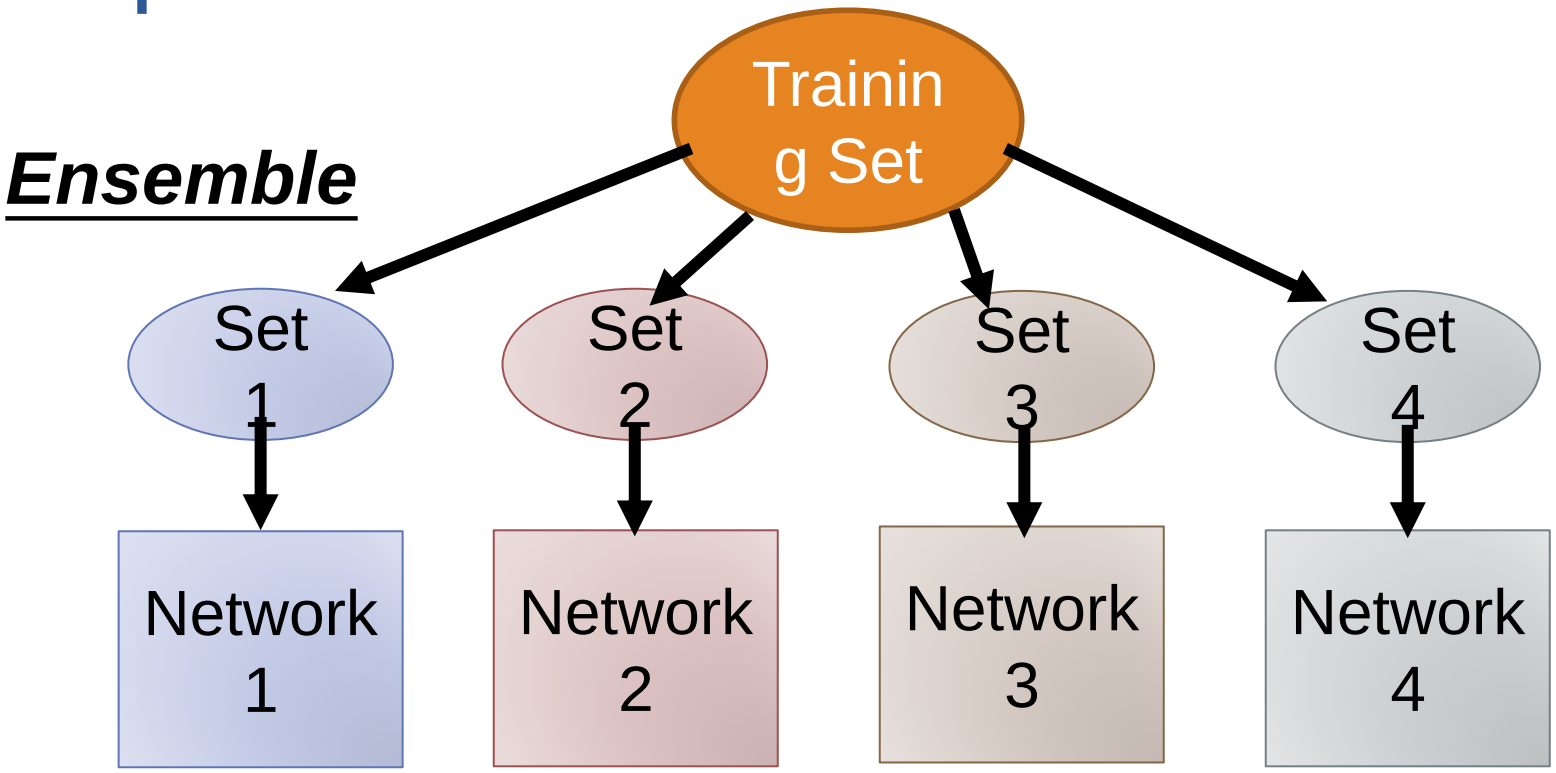
$z' \approx 2z$



Weights multiply (1-p)%

$z' \approx z$

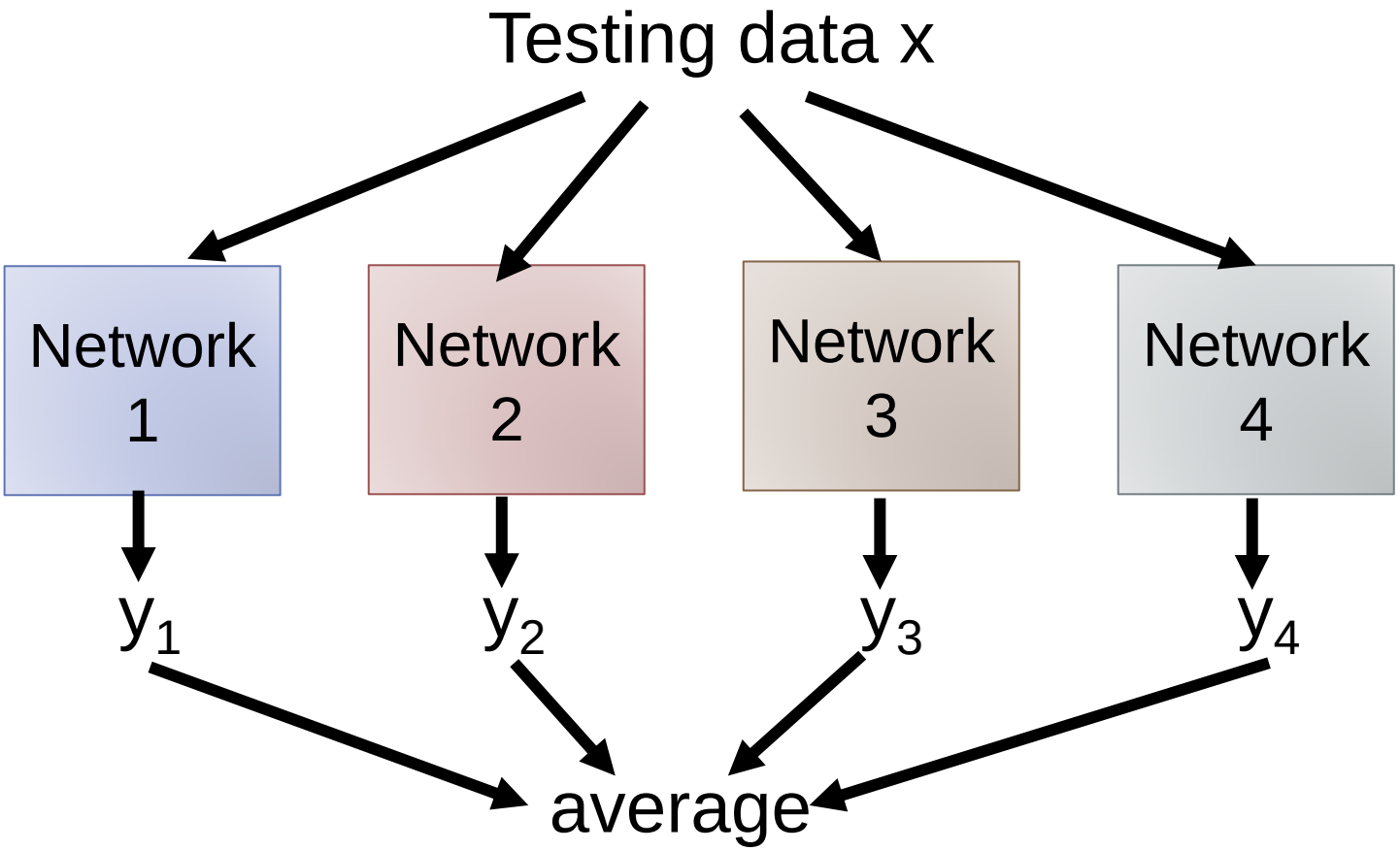
Dropout is a kind of ensemble.



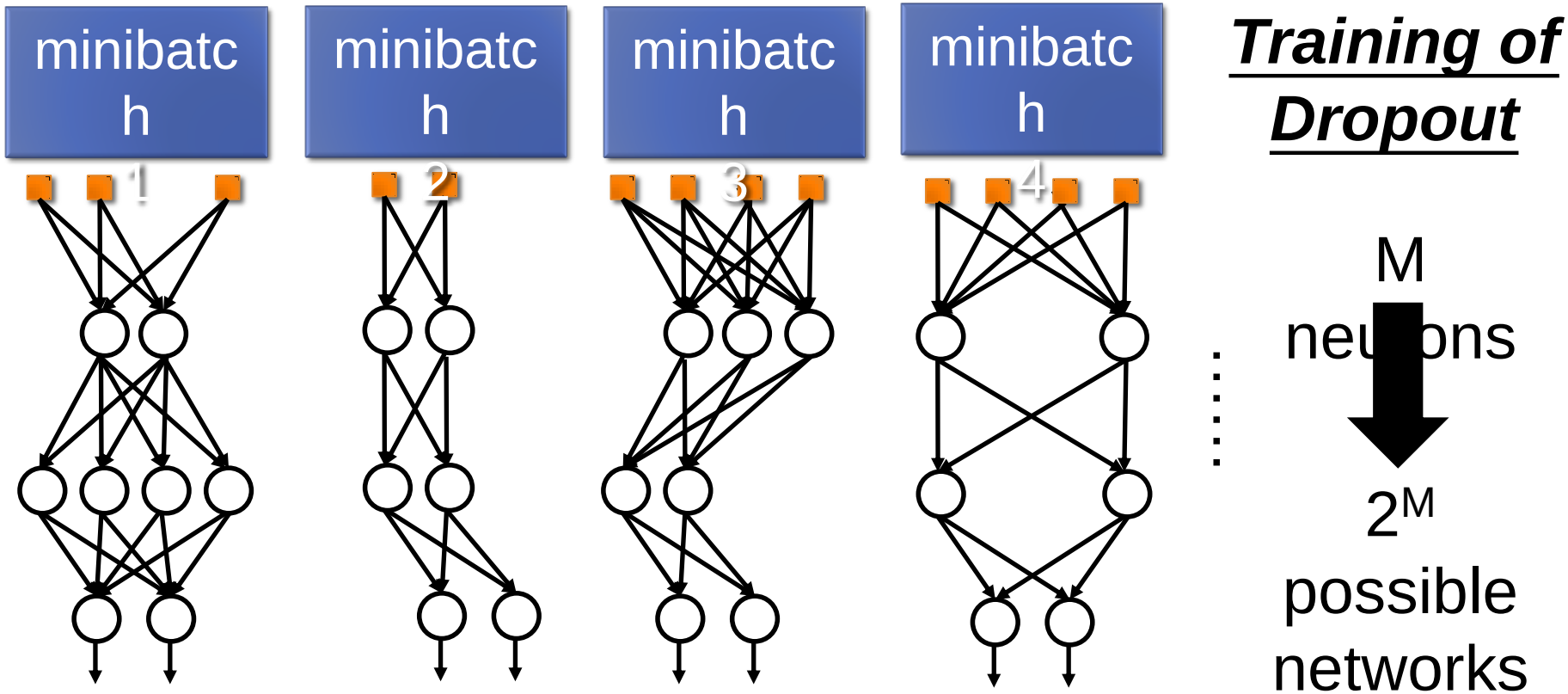
Train a bunch of networks with different structures

Dropout is a kind of ensemble.

Ensemble

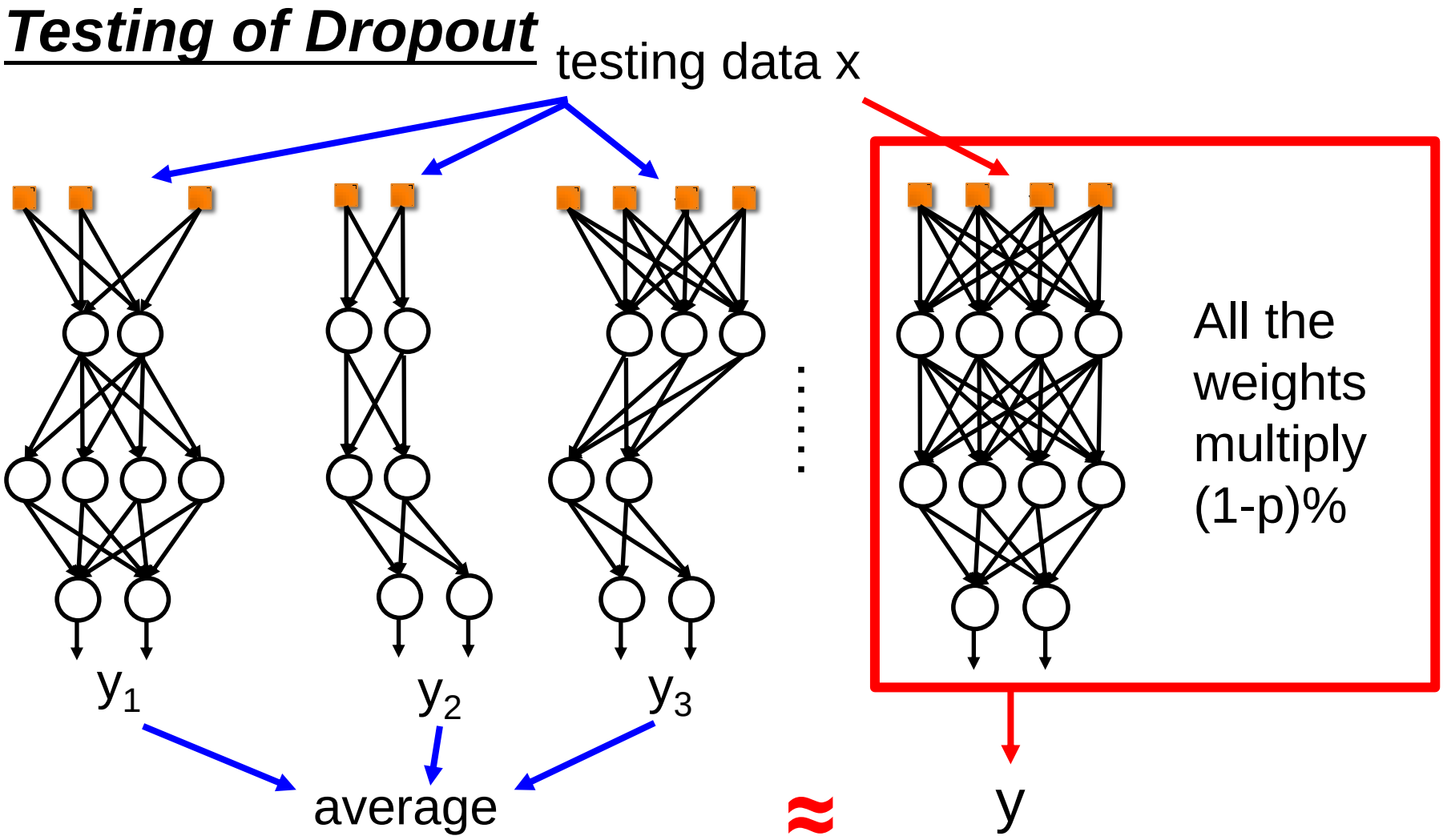


Dropout is a kind of ensemble.



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

Dropout is a kind of ensemble.



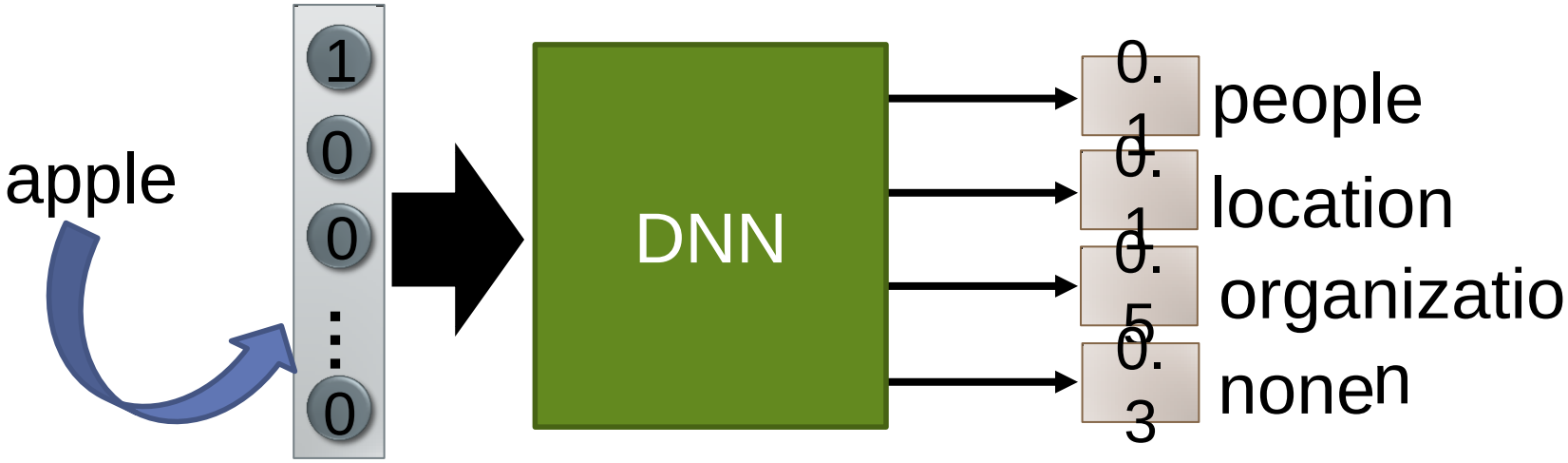
More about dropout

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

PART IV:
NEURAL NETWORK WITH MEMORY

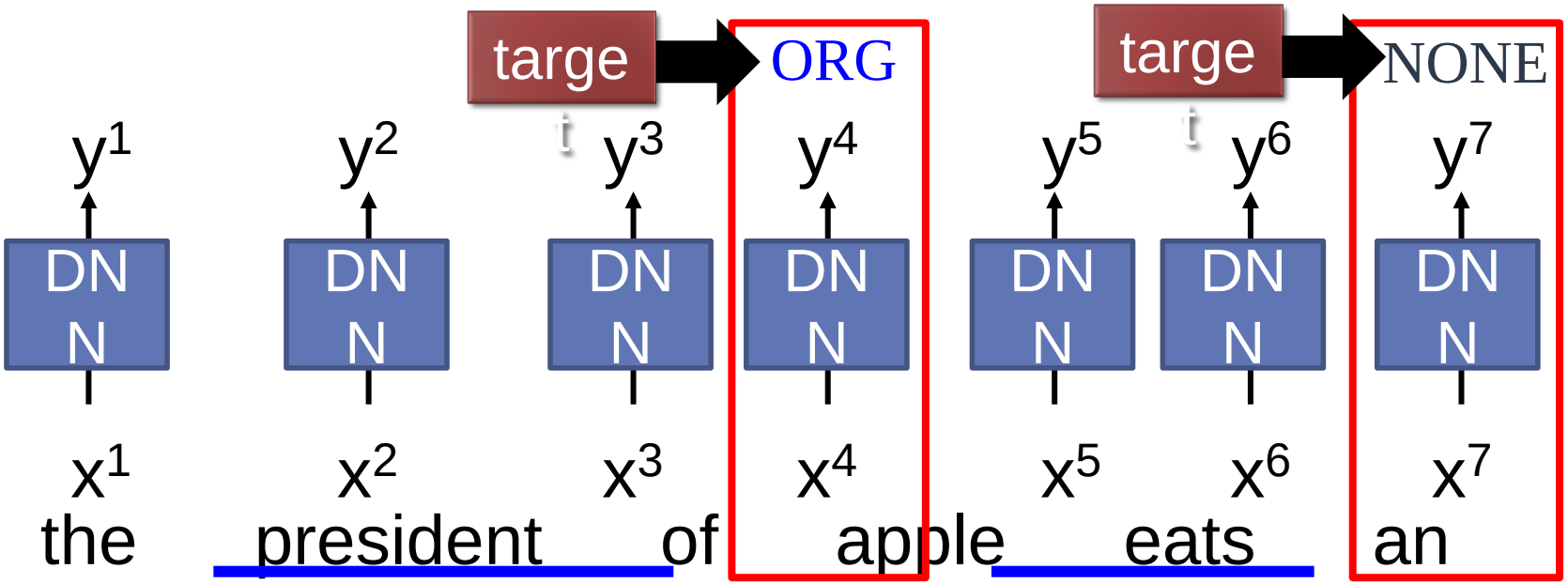
Neural Network needs Memory

- Name Entity Recognition
  - Detecting named entities like name of people, locations, organization, etc. in a sentence.



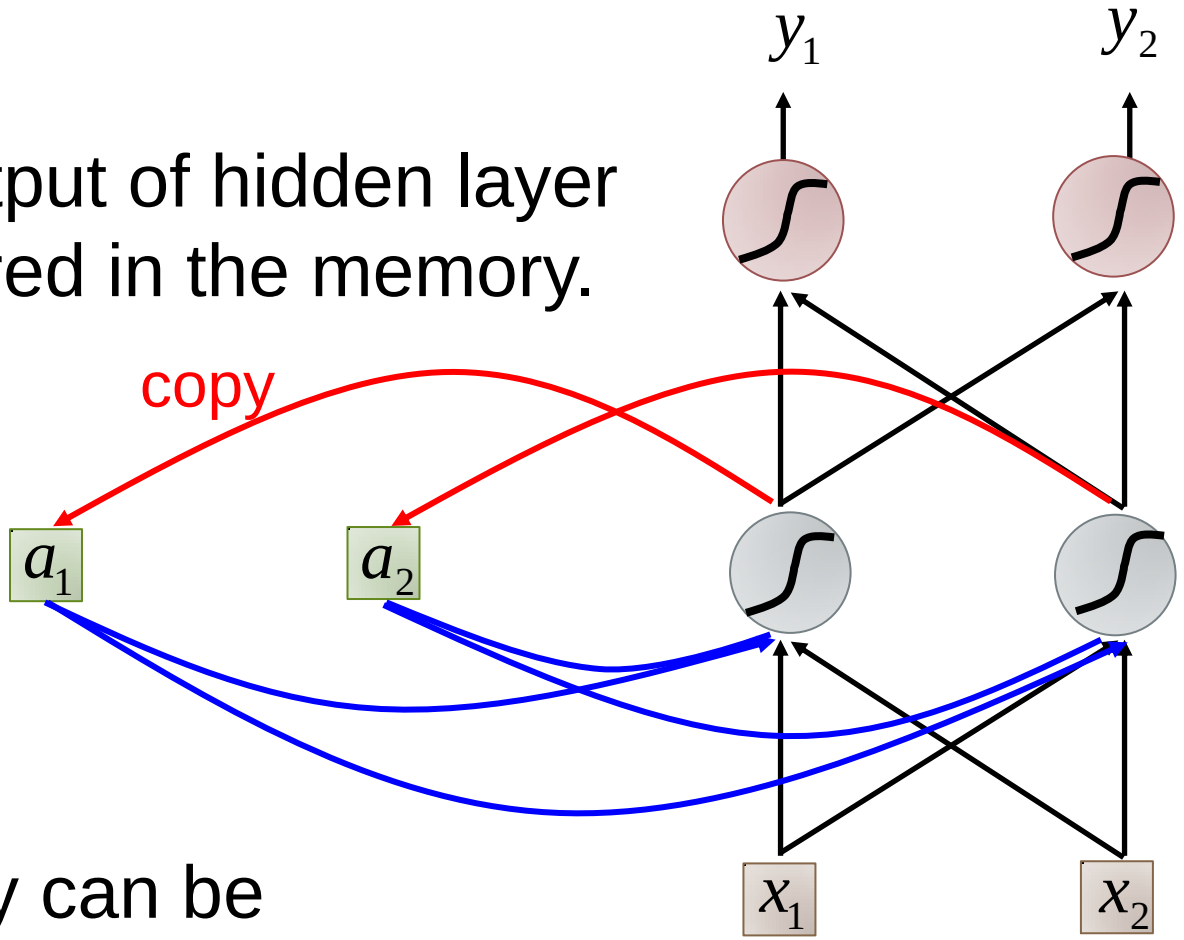
Neural Network needs Memory

- Name Entity Recognition
  - Detecting named entities like name of people, locations, organization, etc. in a sentence.



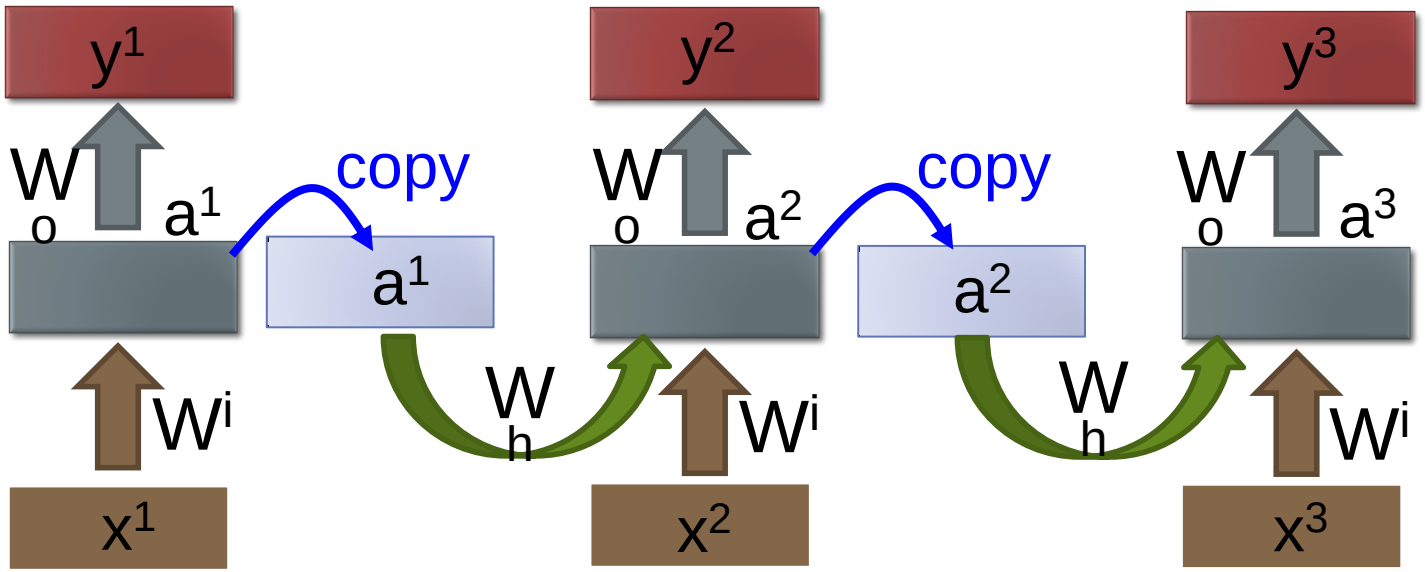
Recurrent Neural Network (RNN)

The output of hidden layer are stored in the memory.



Memory can be considered as another input.

RNN

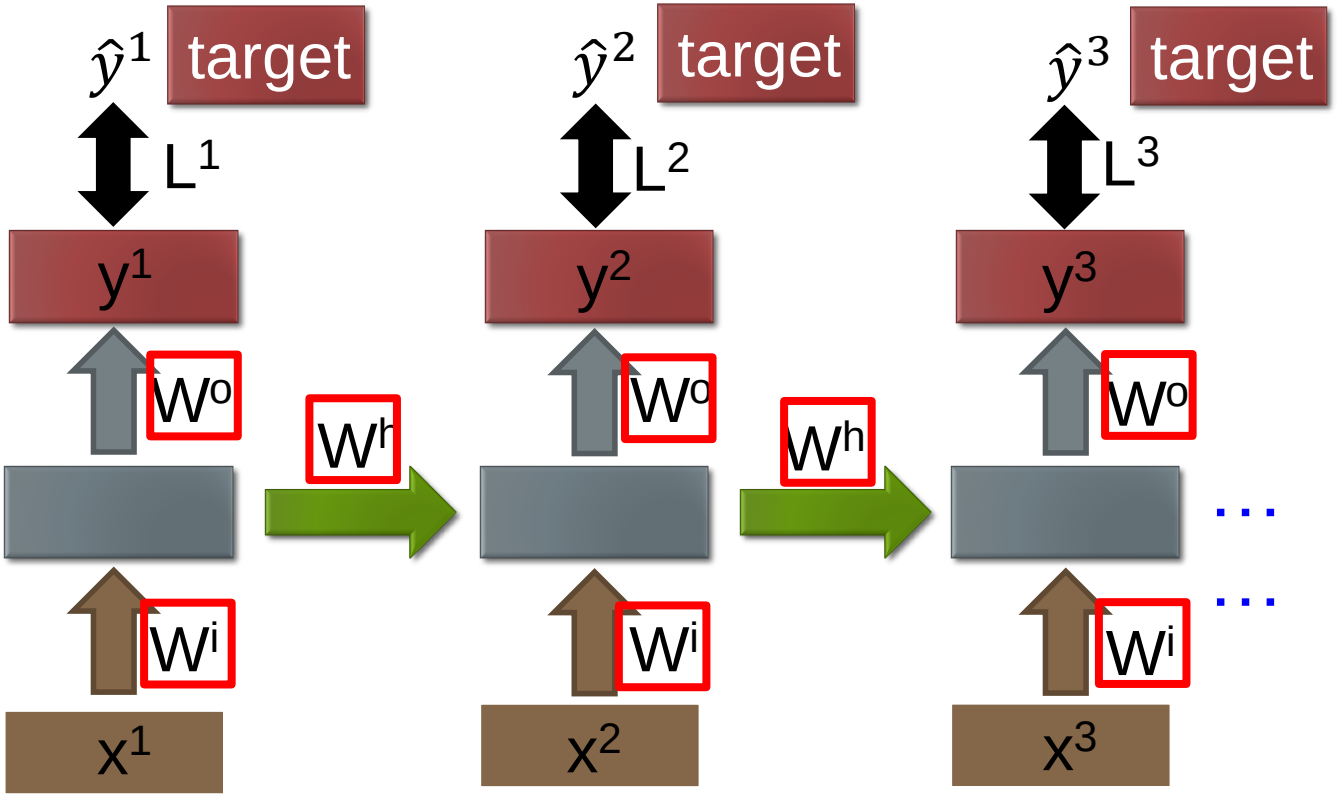


The same network is used again and again.

Output y^i depends on x^1, x^2, …… x^i

RNN

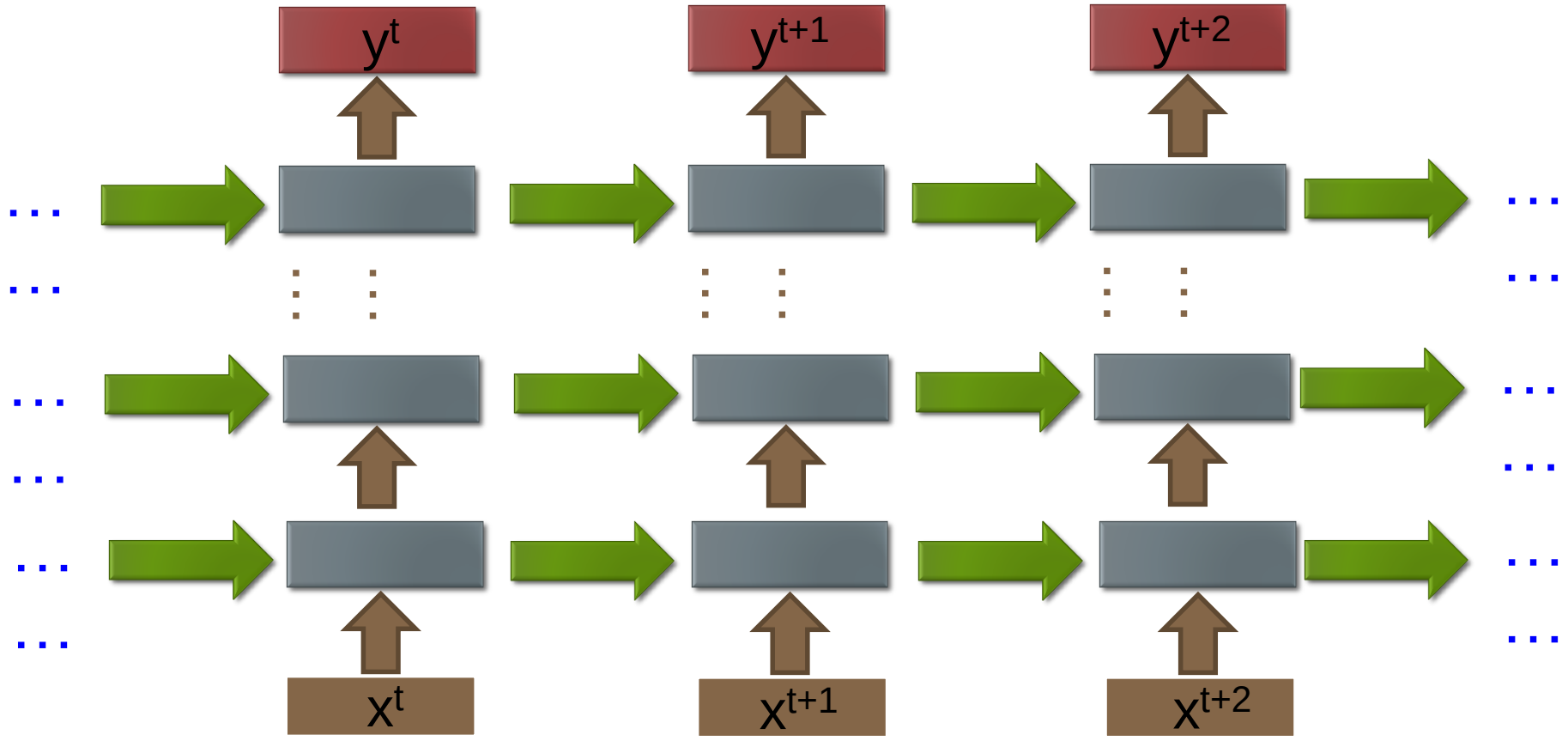
How to train?



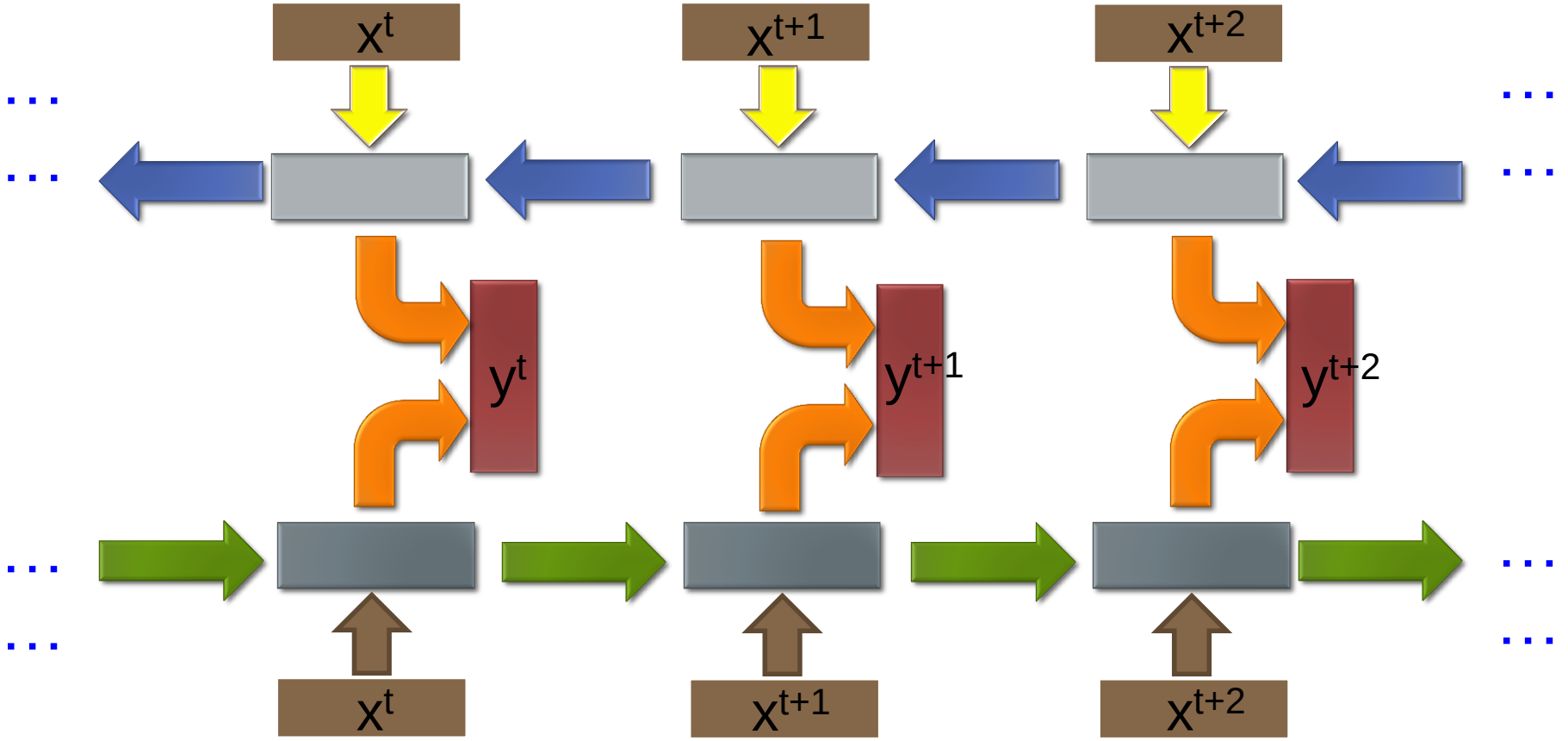
Find the network parameters to minimize the total cost:

Backpropagation through time (BPTT)

Of course it can be deep …

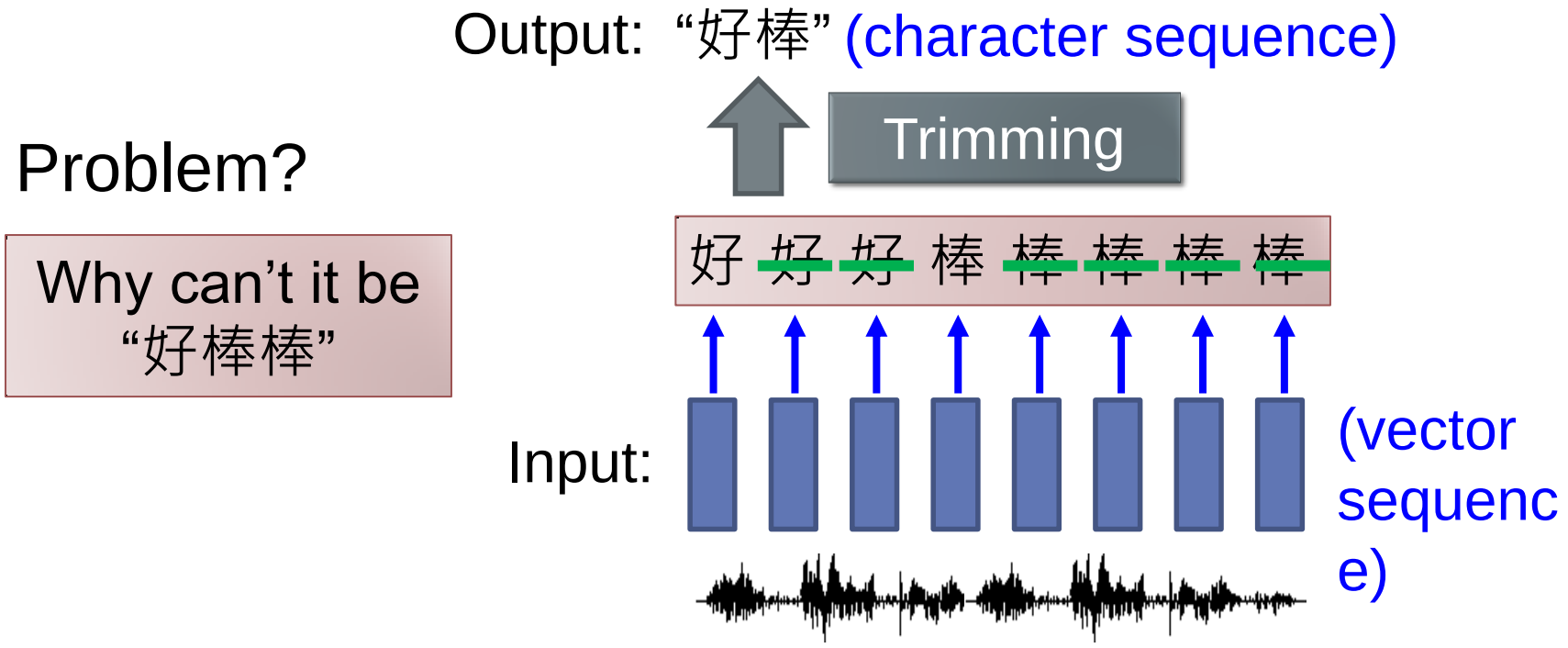


Bidirectional RNN



Many to Many (Output is shorter)

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

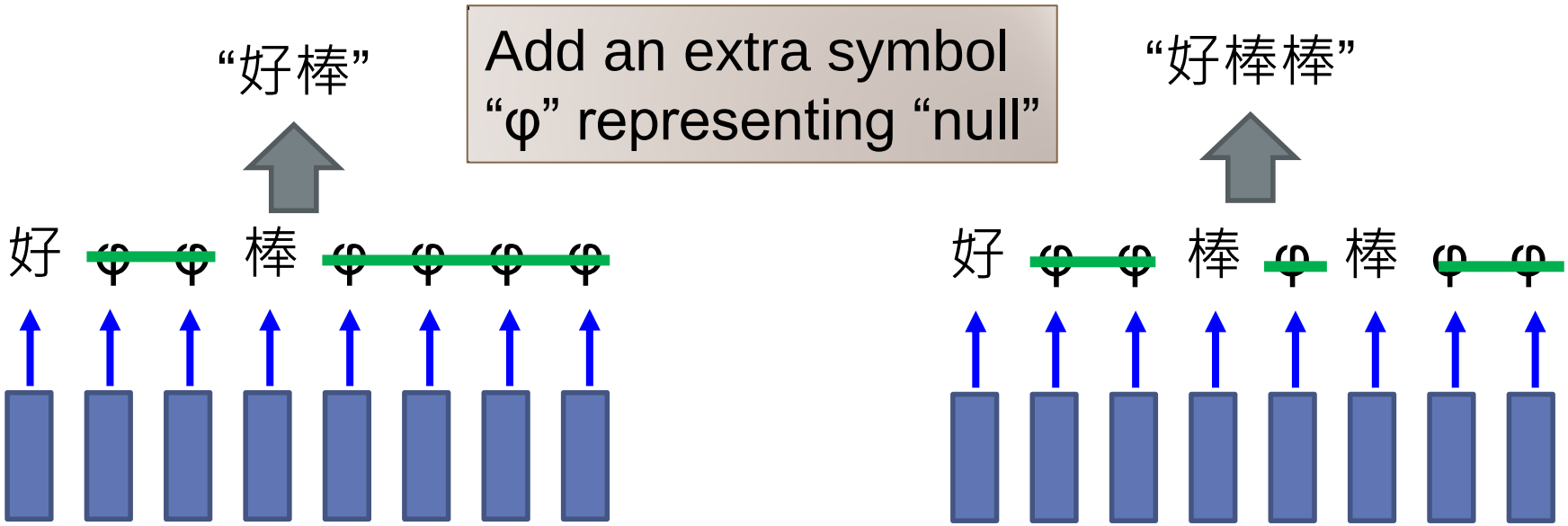


Problem?

Why can’t it be “好棒棒”

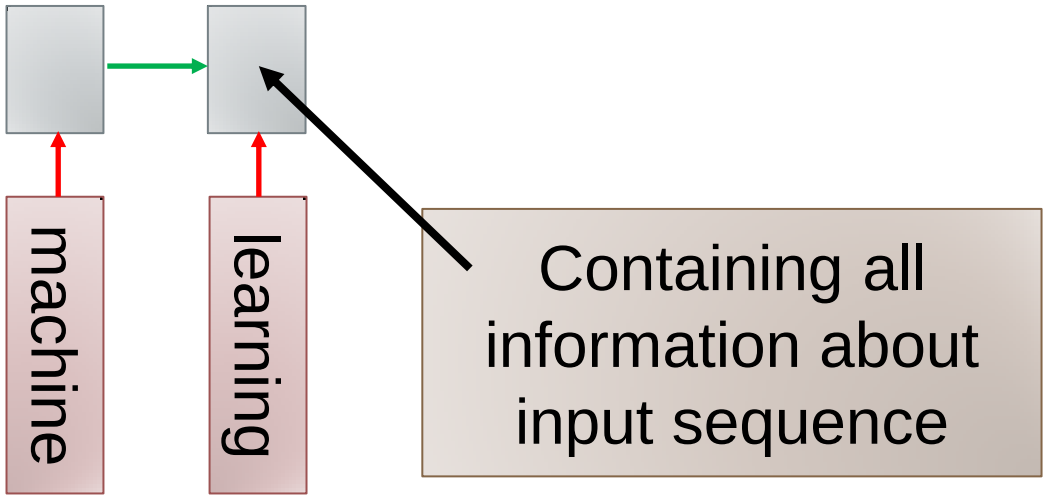
Many to Many (Output is shorter)

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



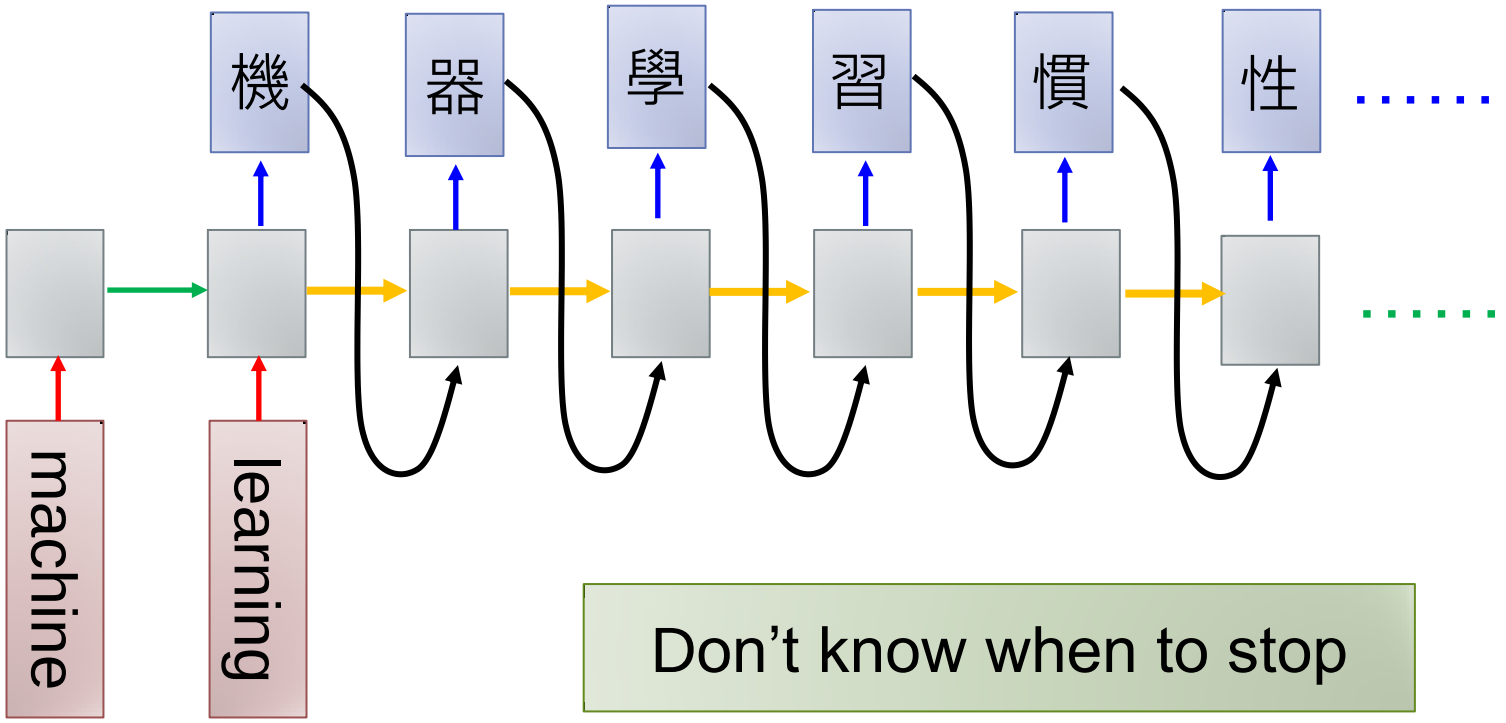
Many to Many (No Limitation)

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

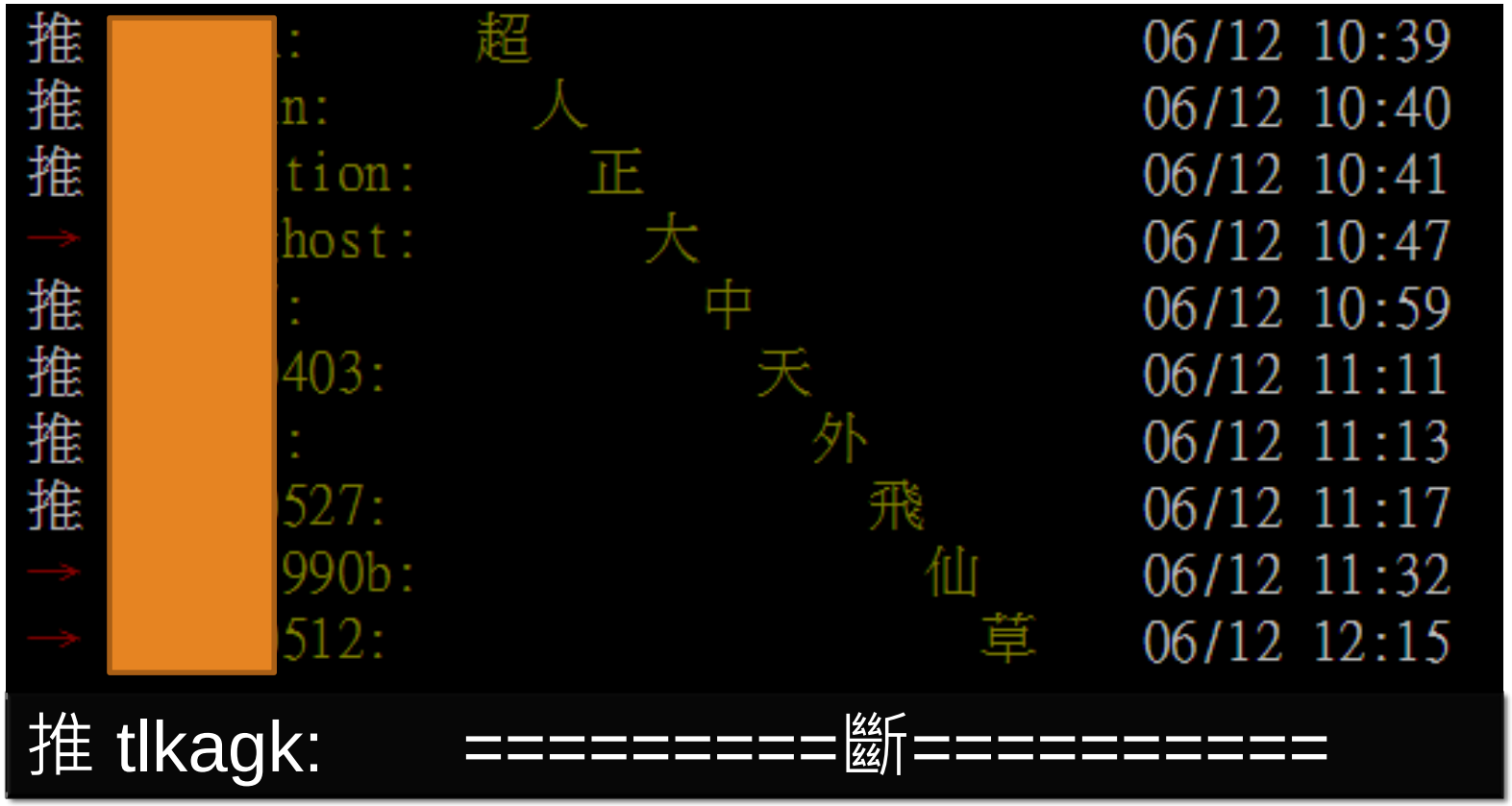


Many to Many (No Limitation)

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



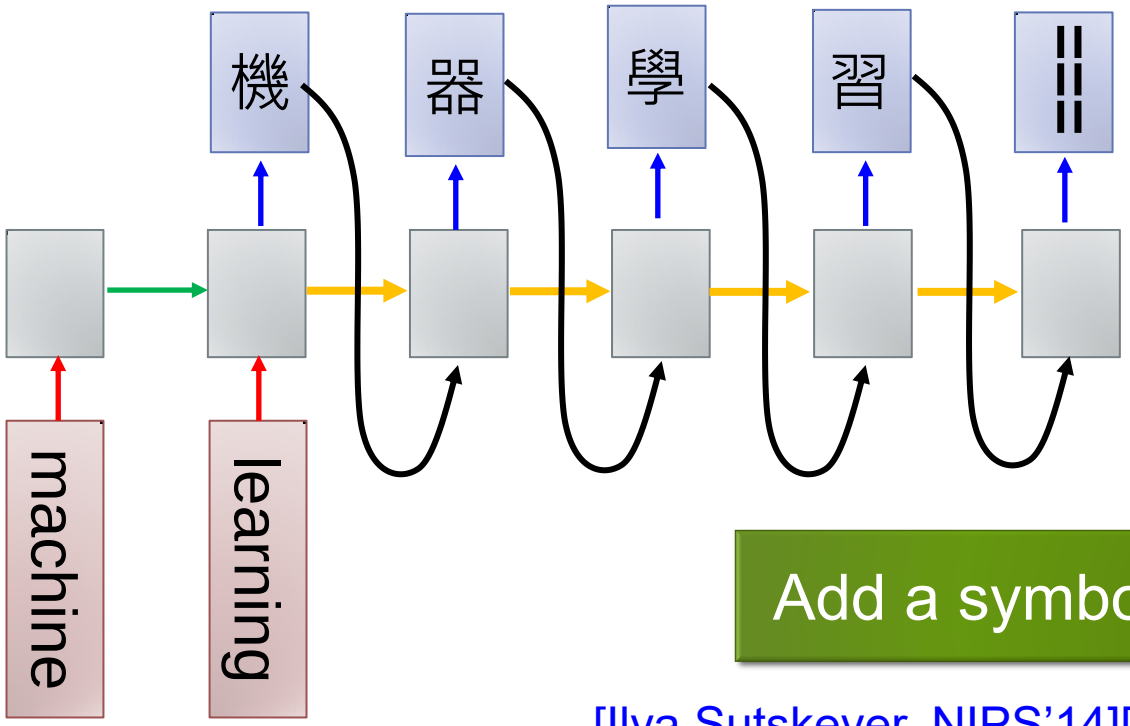
Many to Many (No Limitation)



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

Many to Many (No Limitation)

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



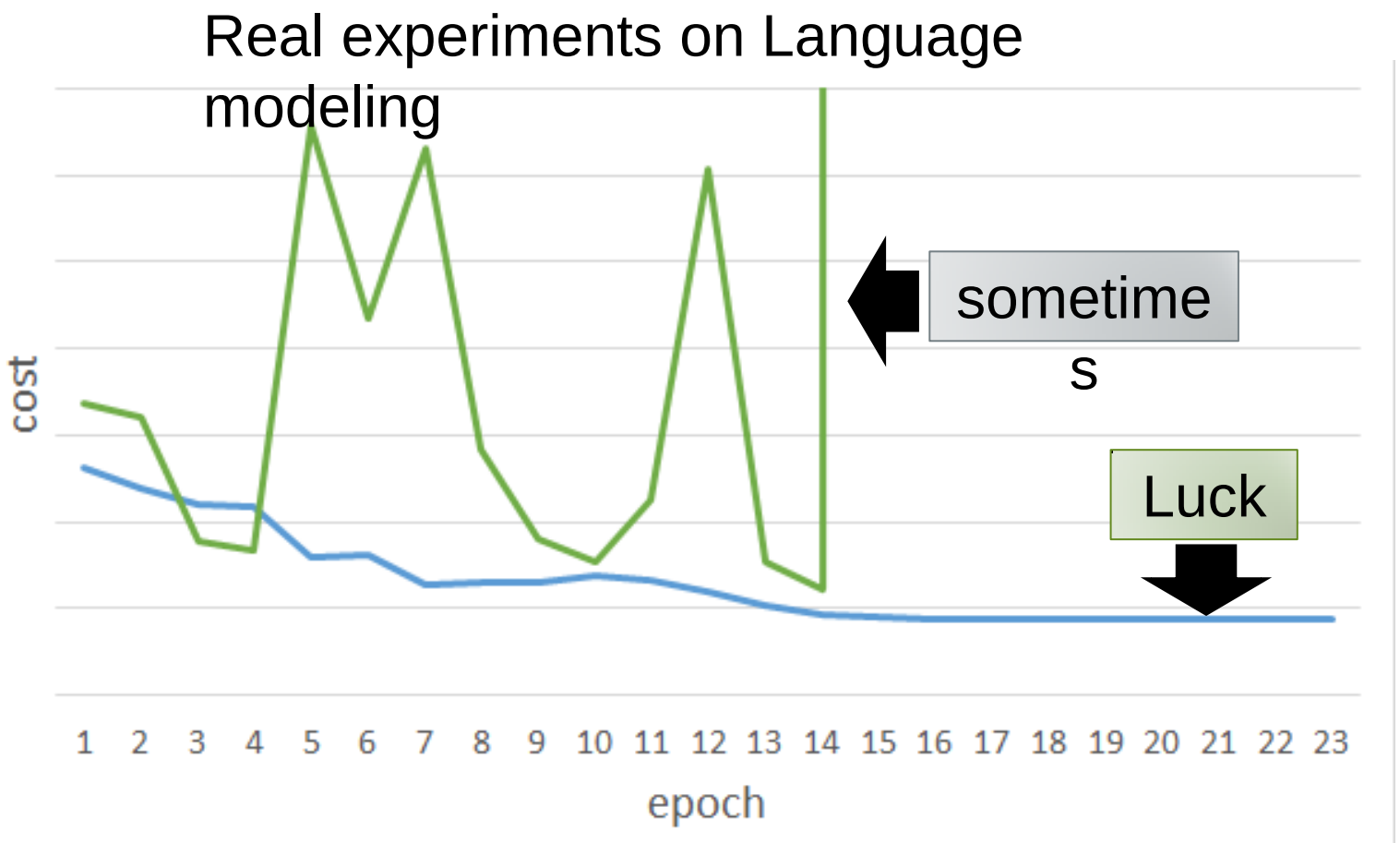
Add a symbol "===" (斷)

[Ilya Sutskever, NIPS'14][Dzmitry Bahdanau, arXiv'1

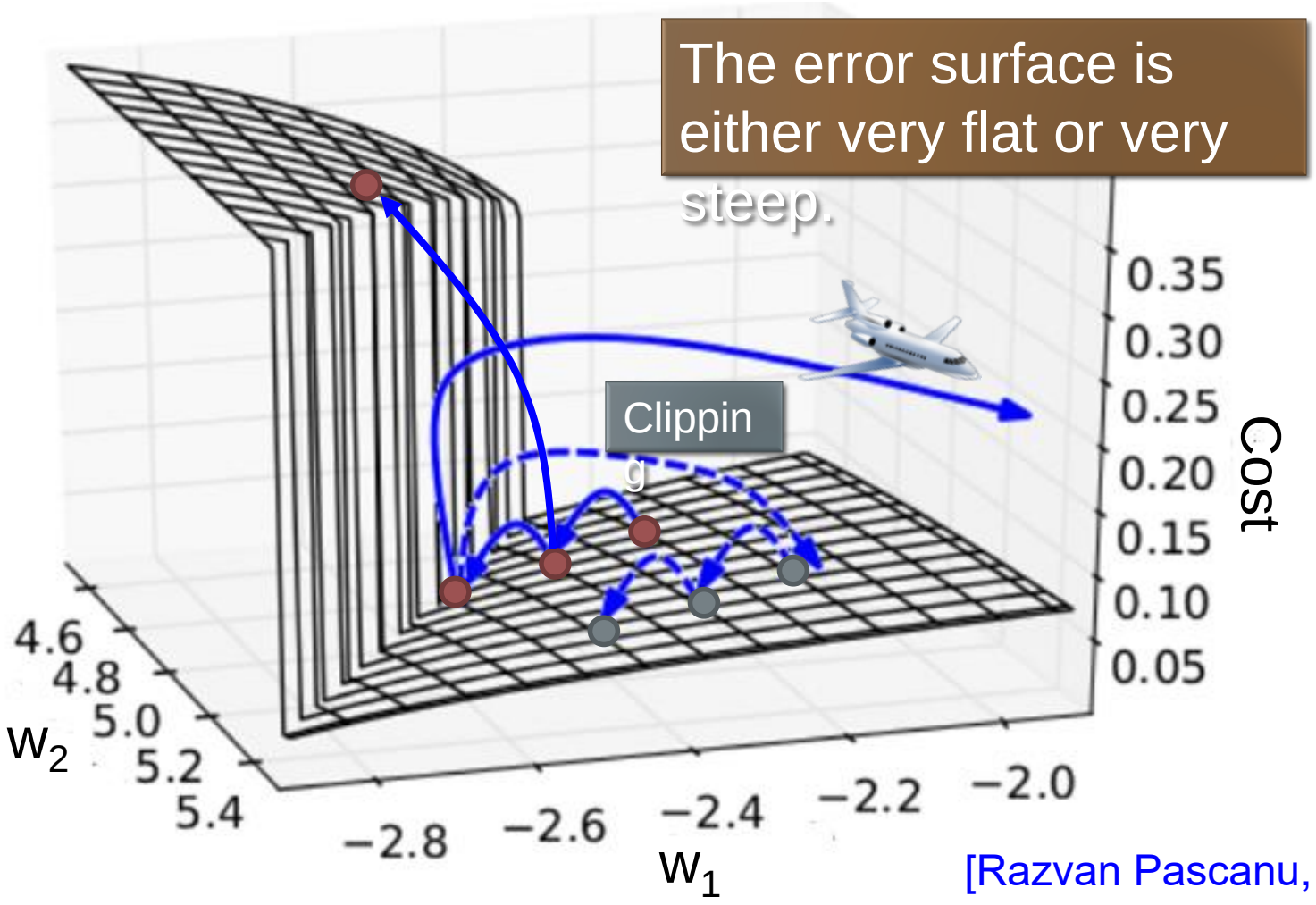
感謝 曾柏翔 同學提供實驗結果

Unfortunately ……

- RNN-based network is not always easy to learn



The error surface is rough.



[Razvan Pascanu, ICML'13]

Why?

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

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

Large gradient → Small Learning rate?

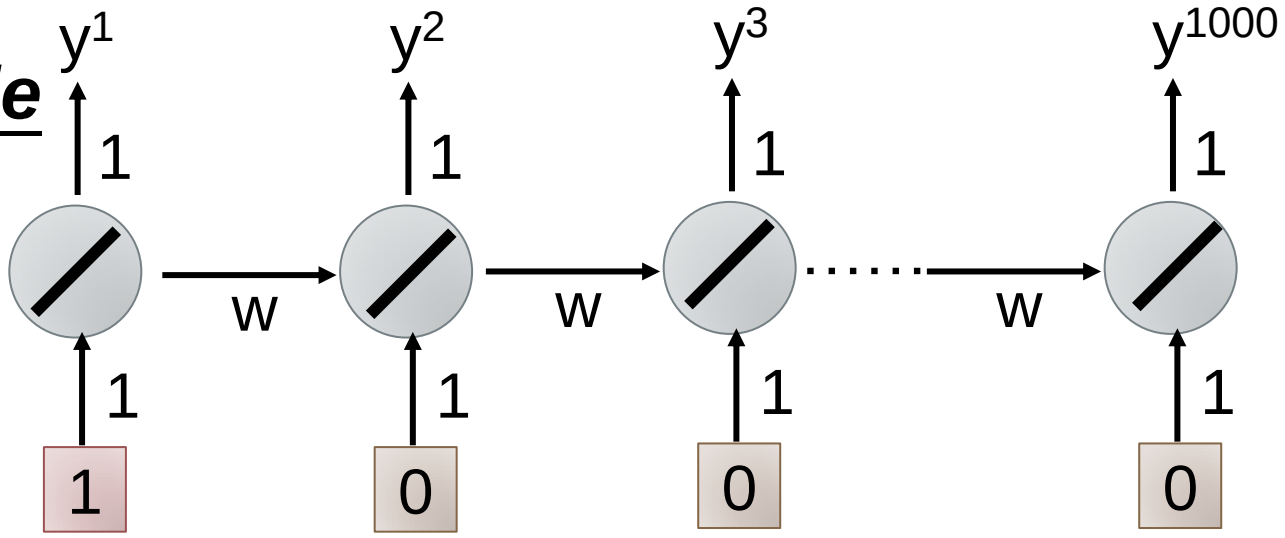
$w = 0.99 \Rightarrow y^{1000} \approx 0$

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

small gradient → Large Learning rate?

$=w^{999}$

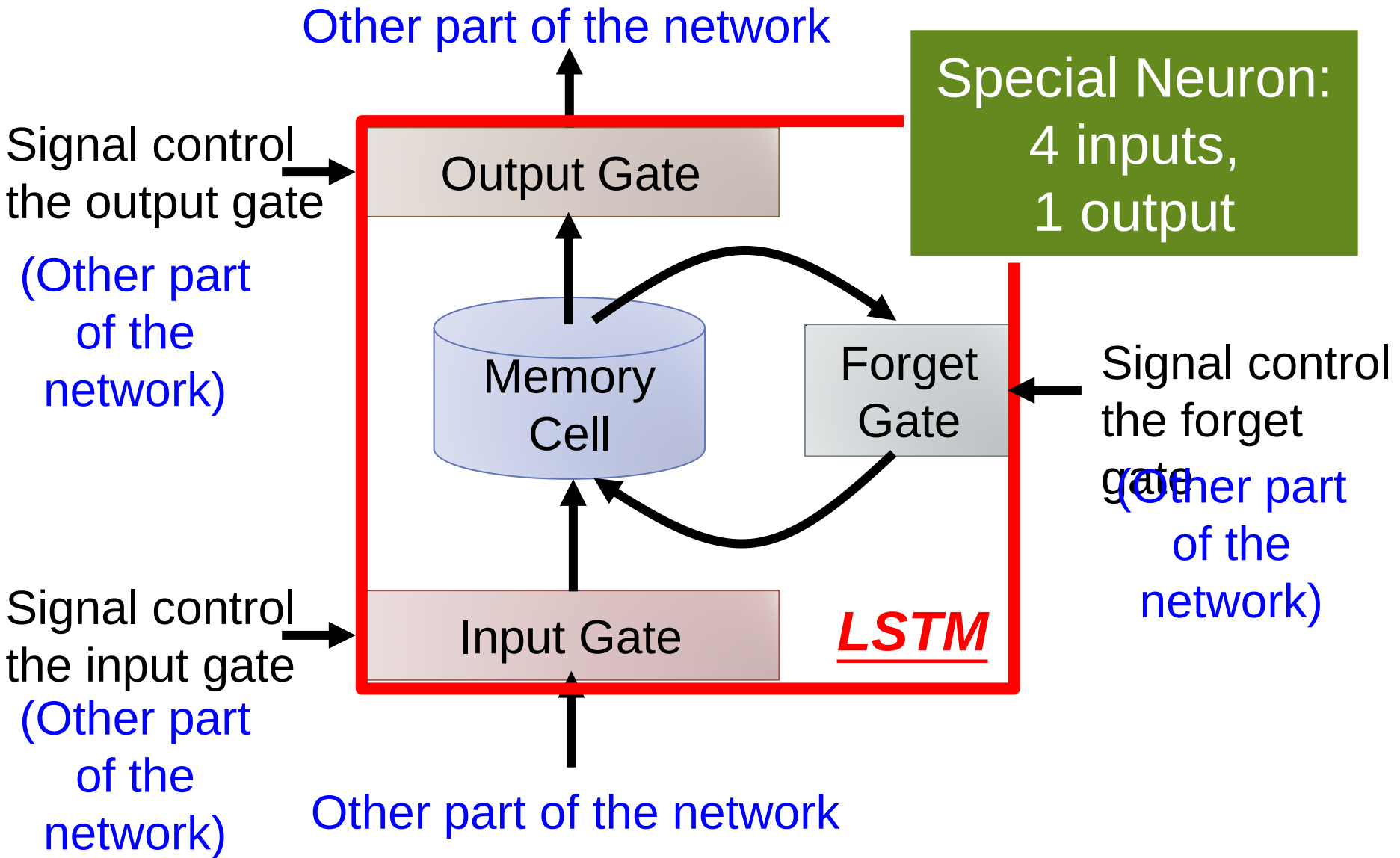
Toy Example

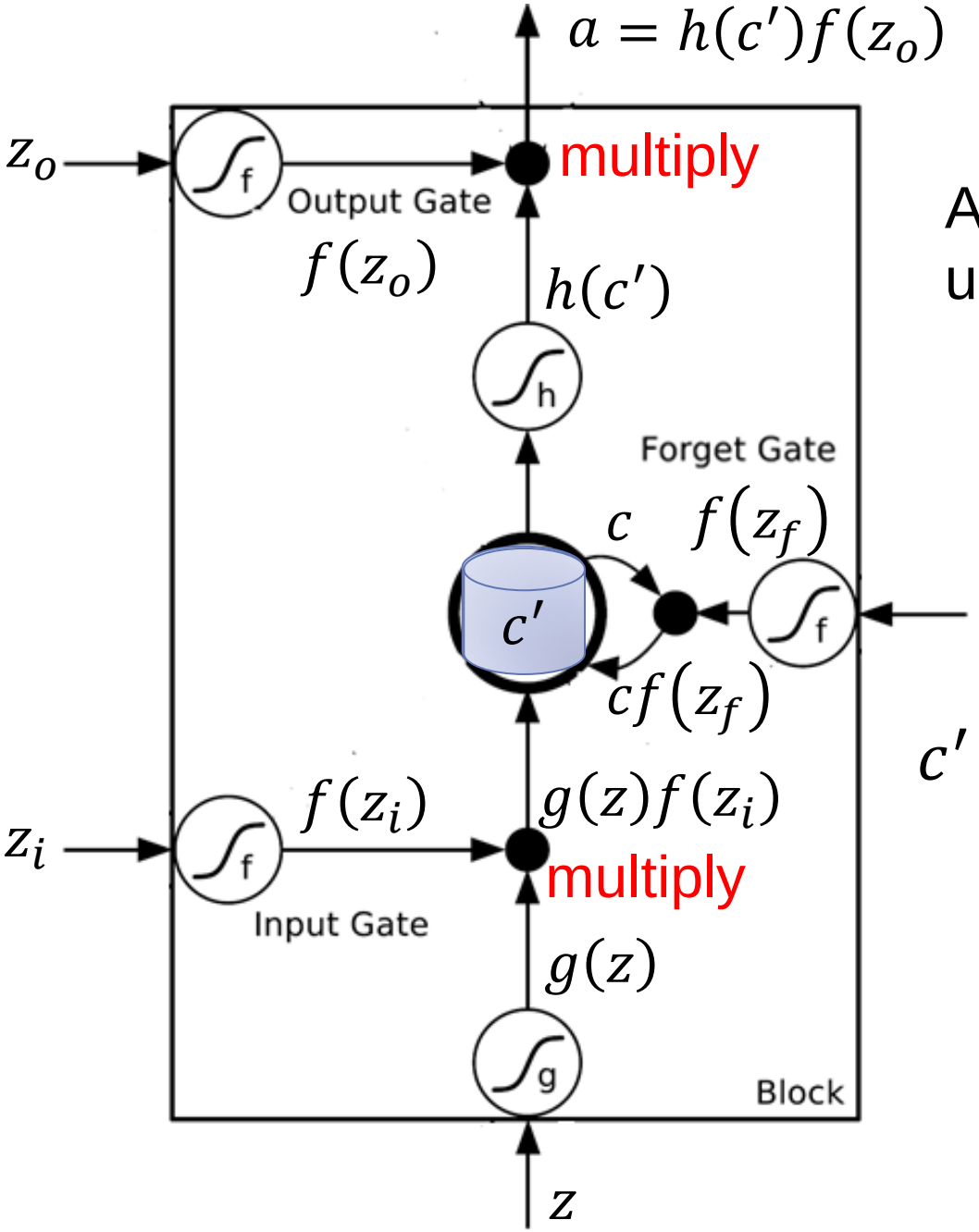


Helpful Techniques

- Nesterov's Accelerated Gradient (NAG):
  - Advance momentum method
- RMS Prop
  - Advanced approach to give each parameter different learning rates
  - Considering the change of Second derivatives
- Long Short-term Memory (LSTM)
  - Can deal with gradient vanishing (not gradient explode)

Long Short-term Memory (LSTM)





Activation function f is usually a sigmoid function

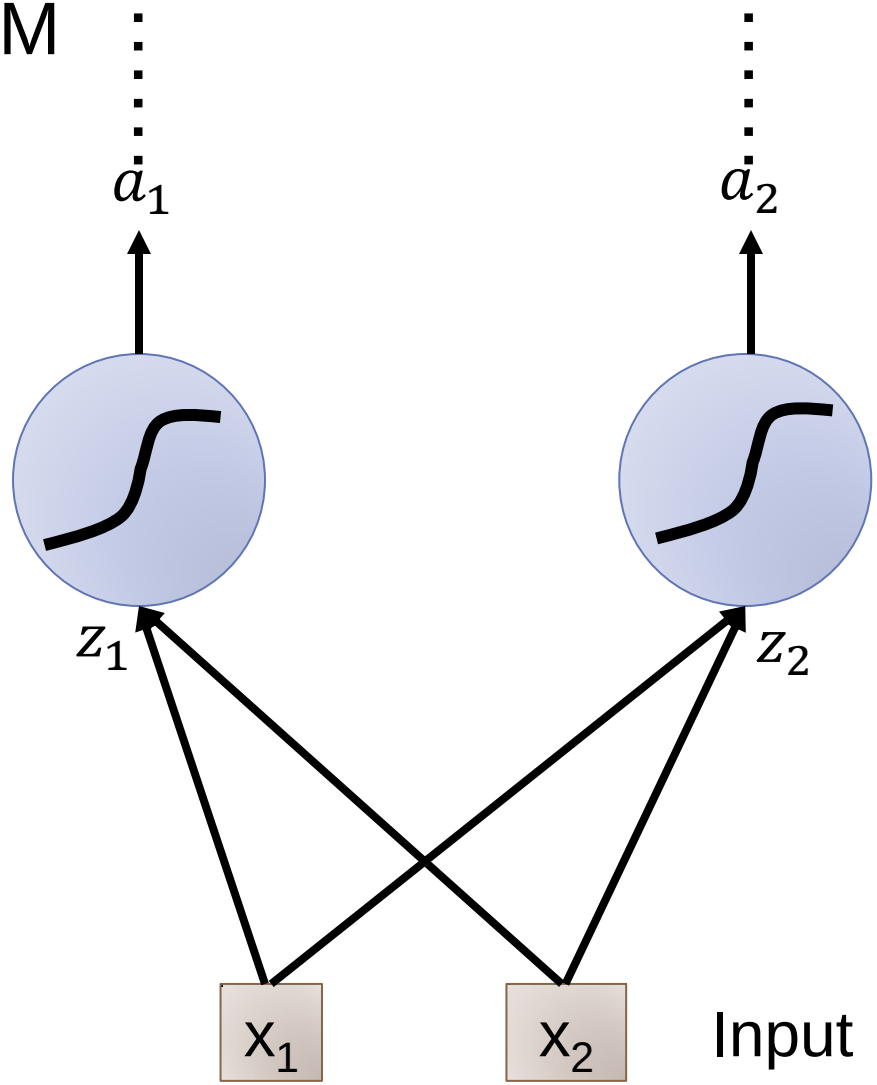
Between 0 and 1

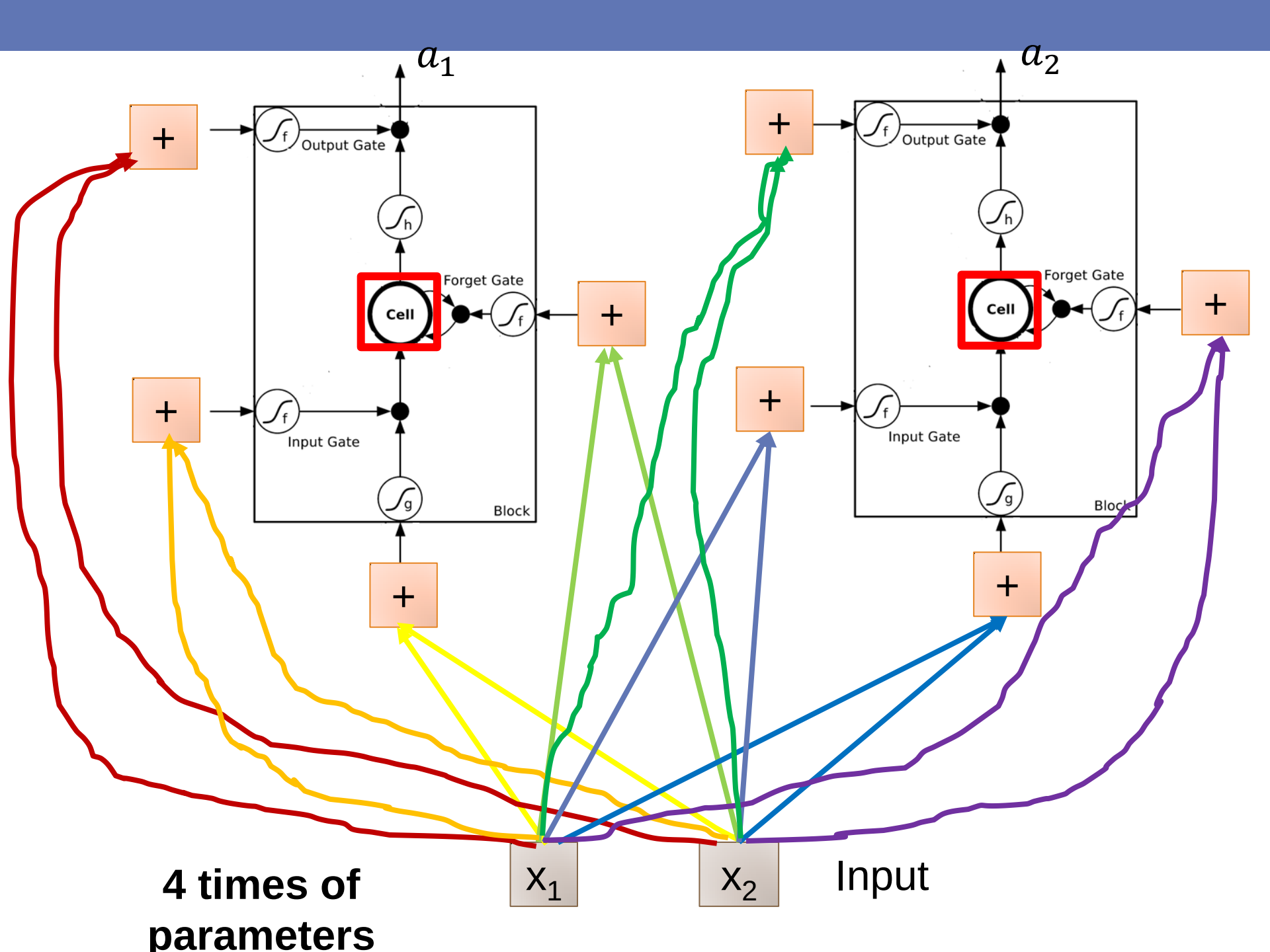
Mimic open and close gate

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

Original Network:

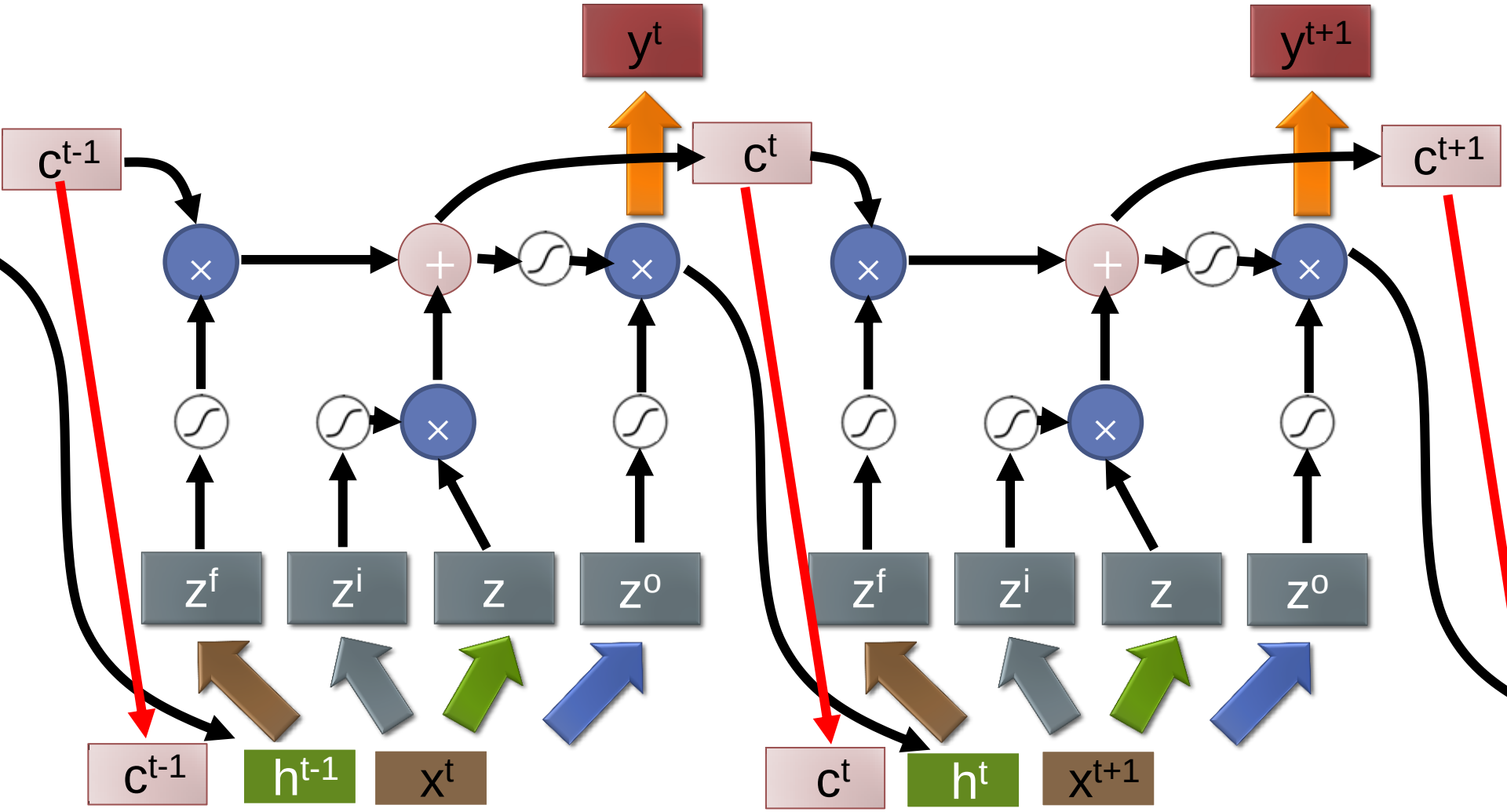
- Simply replace the neurons with LSTM




a_1
a_2
Output Gate
Forget Gate
Input Gate
Cell
Block
x_1
x_2
Input
4 times of parameters

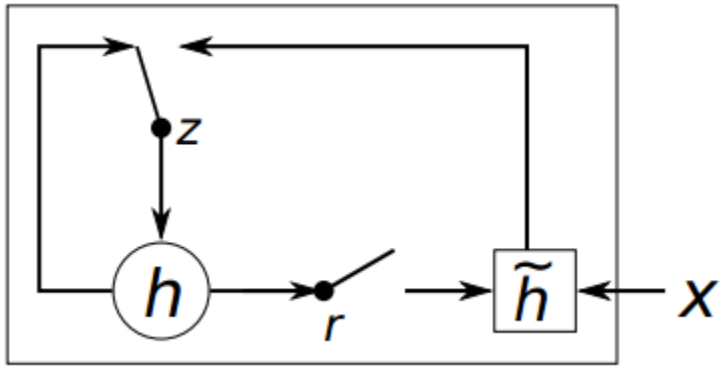
LSTM

Extension:
"peephole"



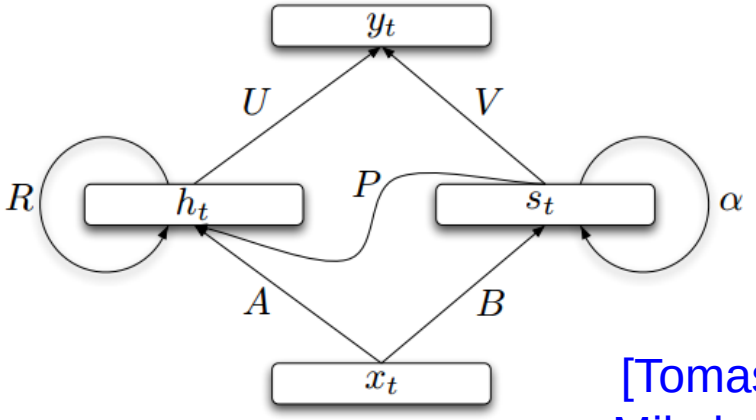
Other Simpler Alternatives

Gated Recurrent Unit (GRU)



[Cho, EMNLP'14]

Structurally Constrained Recurrent Network (SCRN)



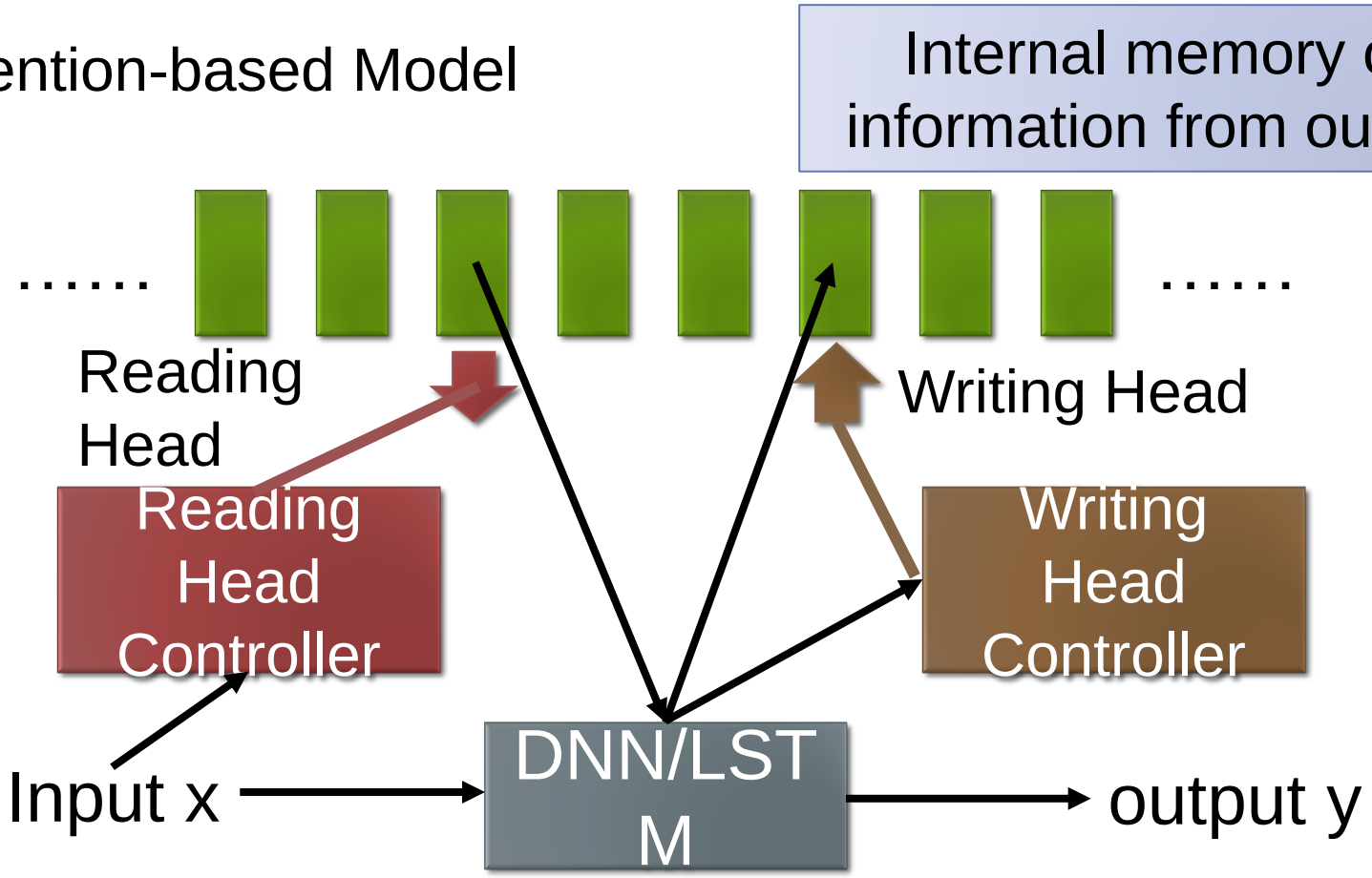
[Tomas Mikolov, ICLR'15]

Vanilla RNN Initialized with Identity matrix + ReLU activation function [Quoc V. Le, arXiv'15]

- Outperform or be comparable with LSTM in 4 different tasks

What is the next wave?

- Attention-based Model



Already applied on speech recognition, caption generation, QA, visual QA

What is the next wave?

- Attention-based Model
- End-To-End Memory Networks. S. Sukhbaatar, A. Szlam, J. Weston, R. Fergus. arXiv Pre-Print, 2015.
- Neural Turing Machines. Alex Graves, Greg Wayne, Ivo Danihelka. arXiv Pre-Print, 2014
- Ask Me Anything: Dynamic Memory Networks for Natural Language Processing. Kumar et al. arXiv Pre-Print, 2015
- Neural Machine Translation by Jointly Learning to Align and Translate. D. Bahdanau, K. Cho, Y. Bengio; International Conference on Representation Learning 2015.
- Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. Kelvin Xu et. al.. arXiv Pre-Print, 2015.
- Attention-Based Models for Speech Recognition. Jan Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, Yoshua Bengio. arXiv Pre-Print, 2015.
- Recurrent models of visual attention. V. Mnih, N. Hees, A. Graves and K. Kavukcuoglu. In NIPS, 2014.
- A Neural Attention Model for Abstractive Sentence Summarization. A. M. Rush, S. Chopra and J. Weston. EMNLP 2015.

CONCLUDING REMARKS

Concluding Remarks

- Introduction of deep learning
- Discussing some reasons using deep learning
- New techniques for deep learning
  - ReLU, Maxout
  - Giving all the parameters different learning rates
  - Dropout
- Network with memory
  - Recurrent neural network
  - Long short-term memory (LSTM)

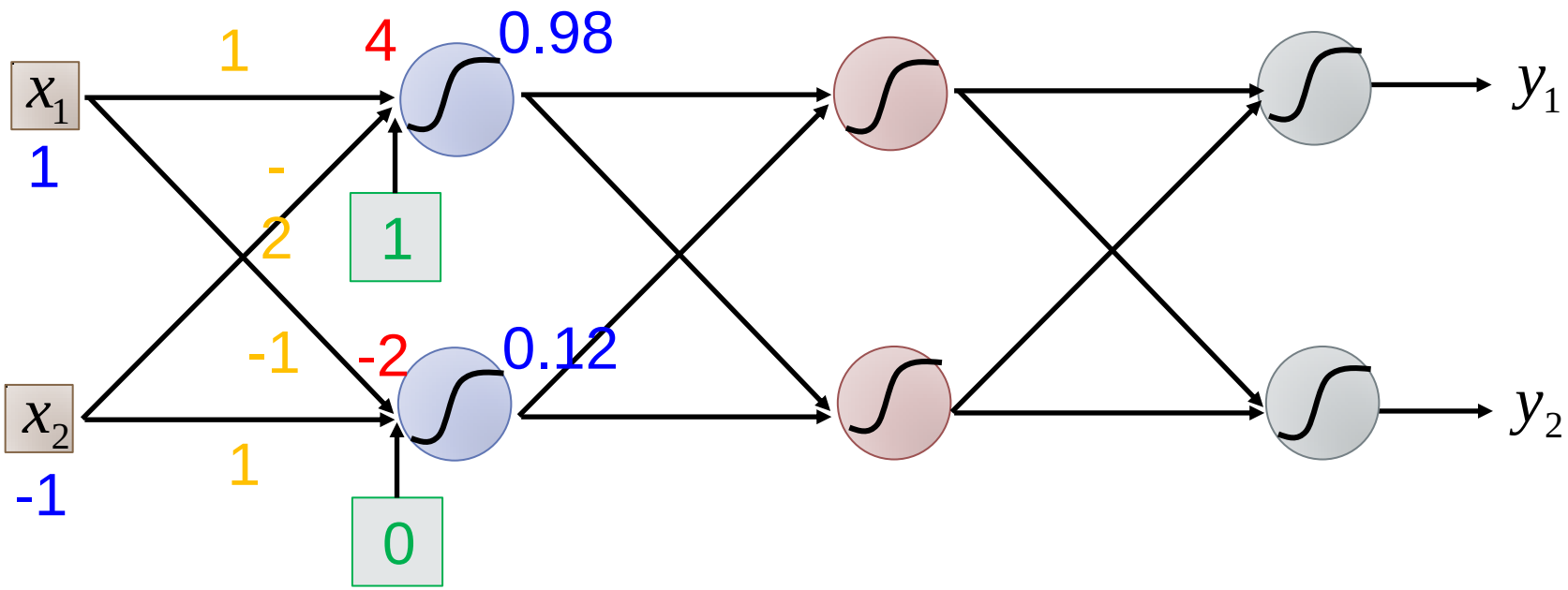
Reading Materials

- “Neural Networks and Deep Learning”
  - written by Michael Nielsen
  - http://neuralnetworksanddeeplearning.com/
- “Deep Learning” (not finished yet)
  - Written by Yoshua Bengio, Ian J. Goodfellow and Aaron Courville
  - http://www.iro.umontreal.ca/~bengioy/dlbook/

THANK YOU
FOR YOUR ATTENTION!

APPENDIX

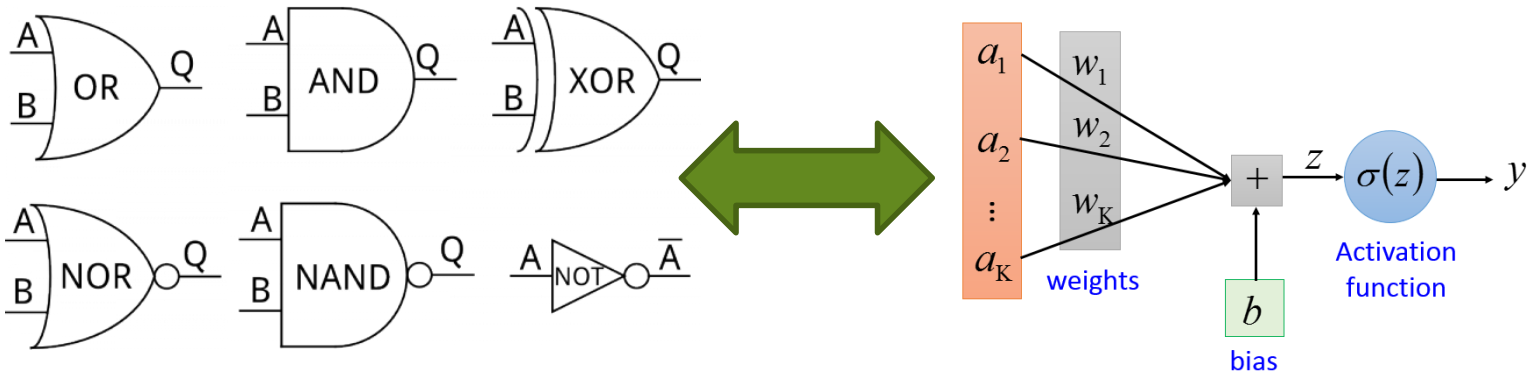
Matrix Operation



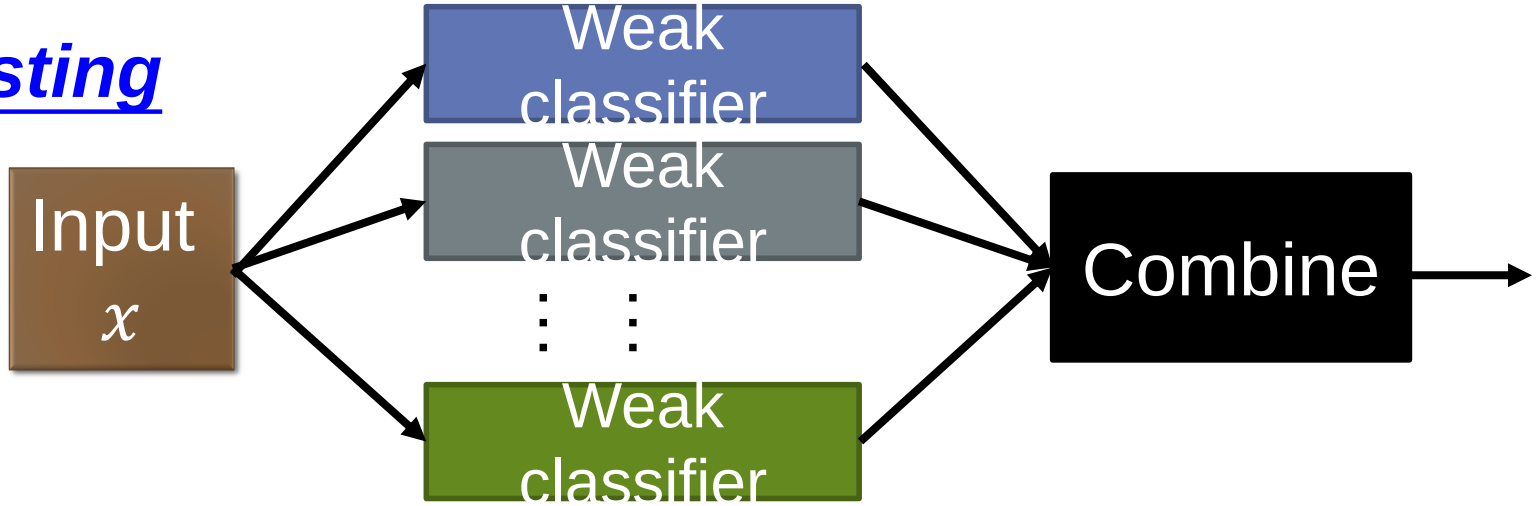
$$\sigma\left(\begin{bmatrix} & W & \end{bmatrix} \begin{bmatrix} x \end{bmatrix} + b \right) = \begin{bmatrix} a \end{bmatrix}$$

Why Deep? – Logic Circuits

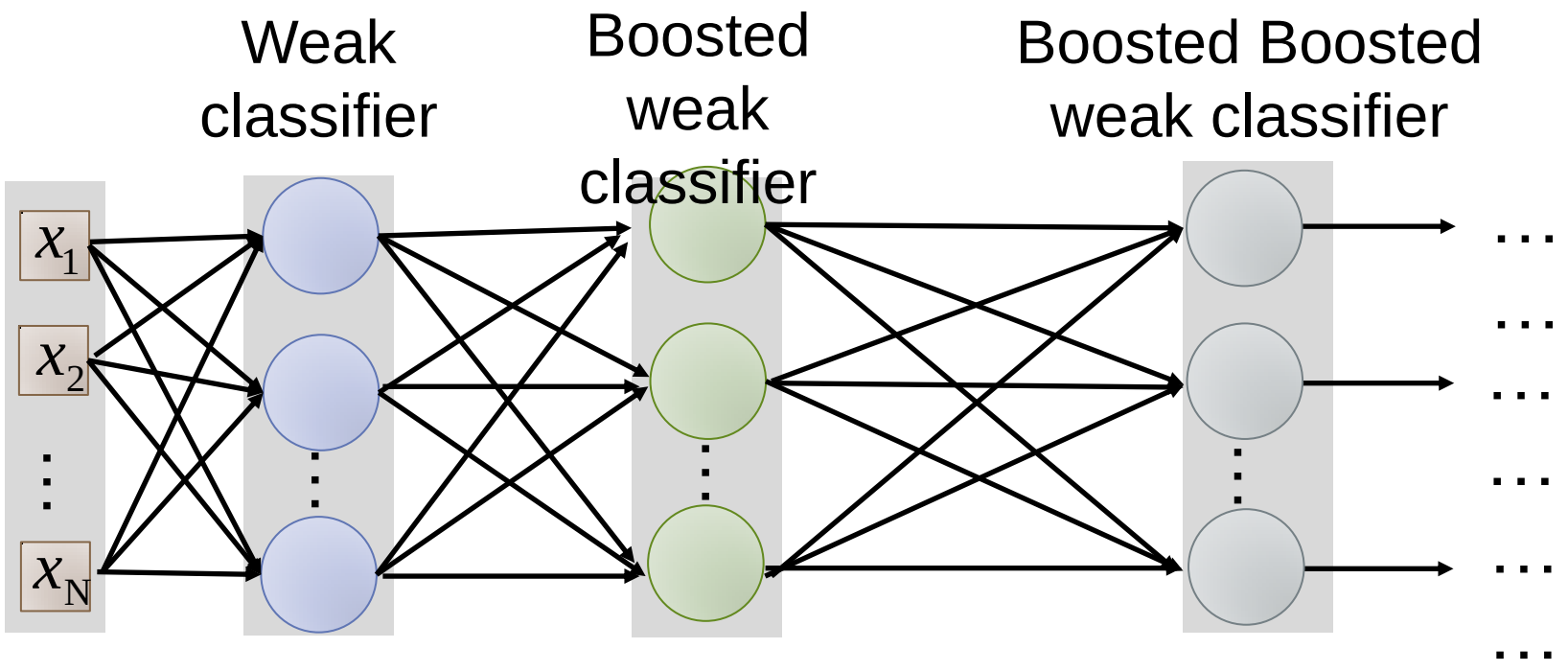
- A two levels of basic logic gates can represent any Boolean function.
- However, no one uses two levels of logic gates to build computers
- Using multiple layers of logic gates to build some functions are much simpler (less gates needed).



Boosting

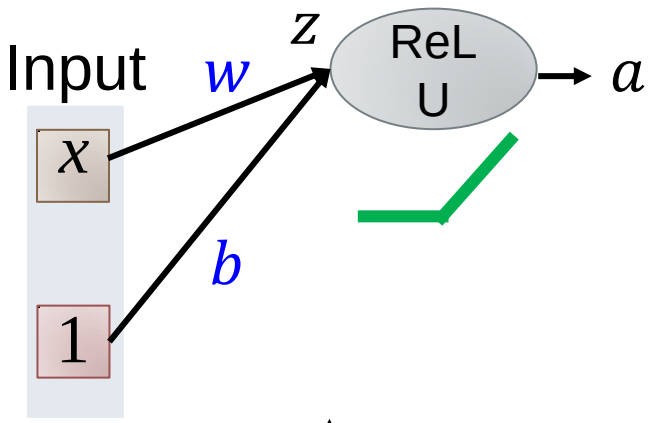


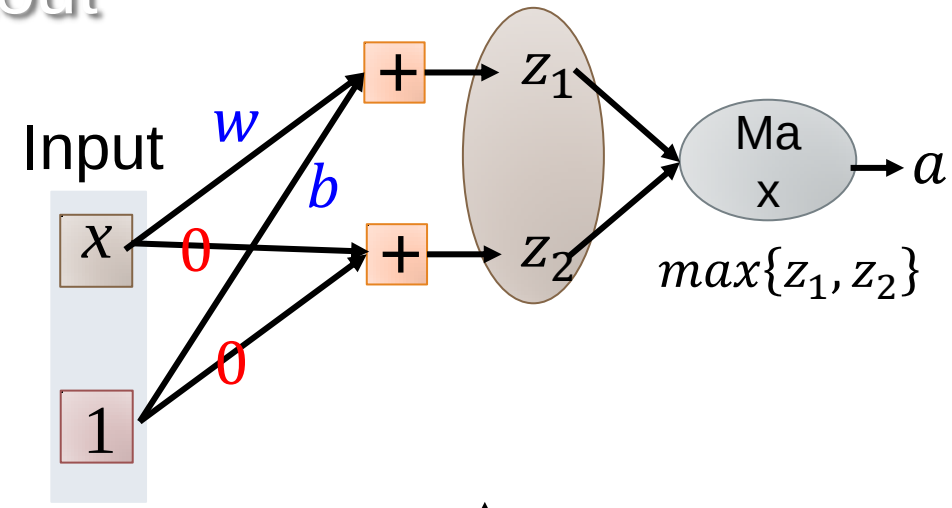
Deep Learning

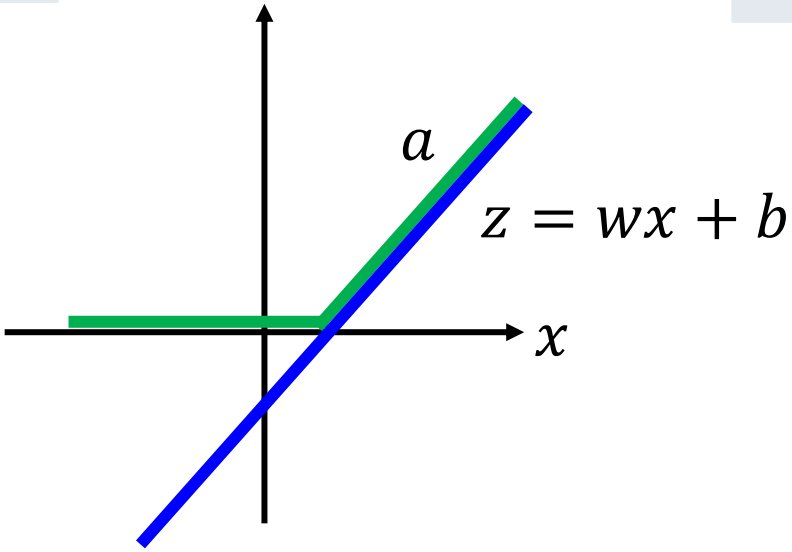


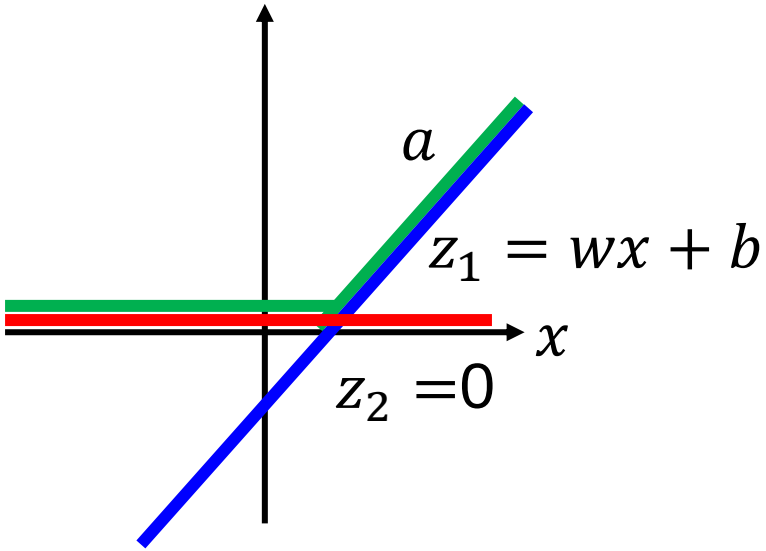
Maxout

ReLU is a special cases of Maxout



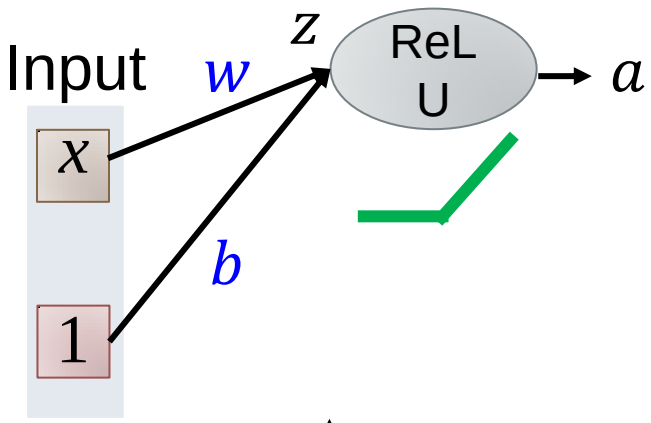


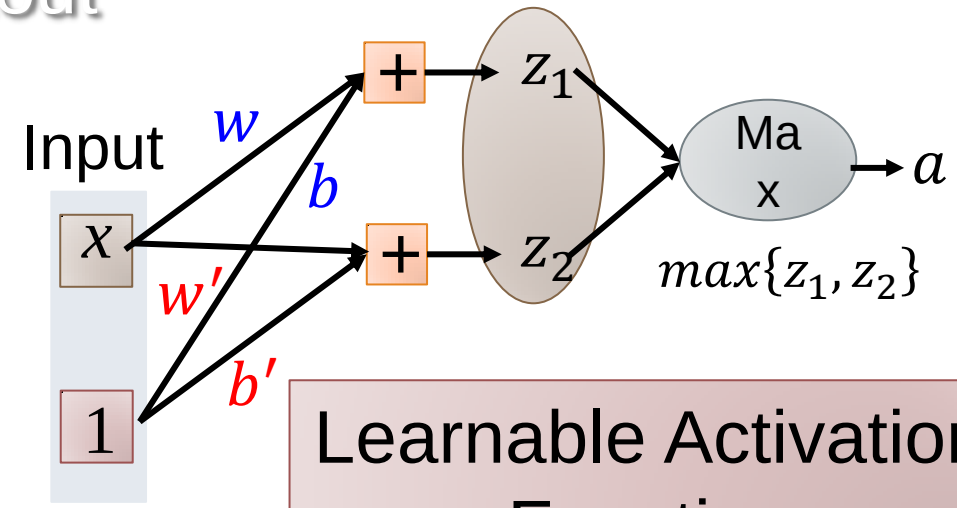




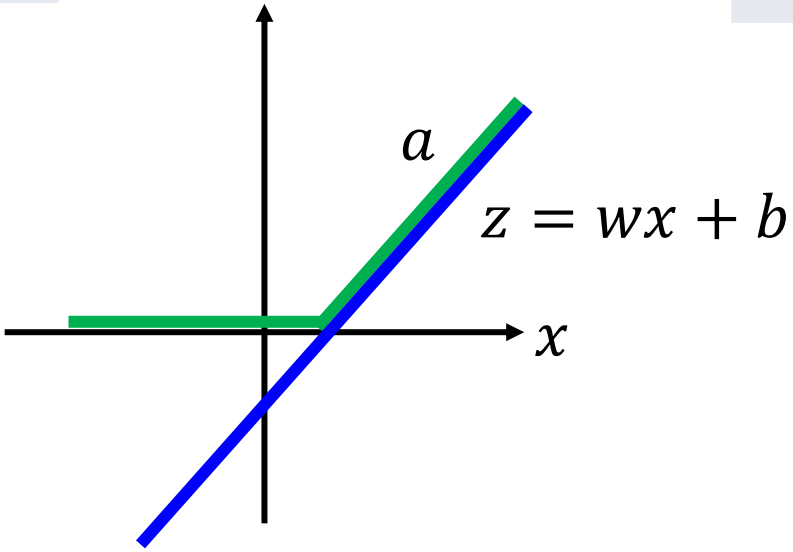
Maxout

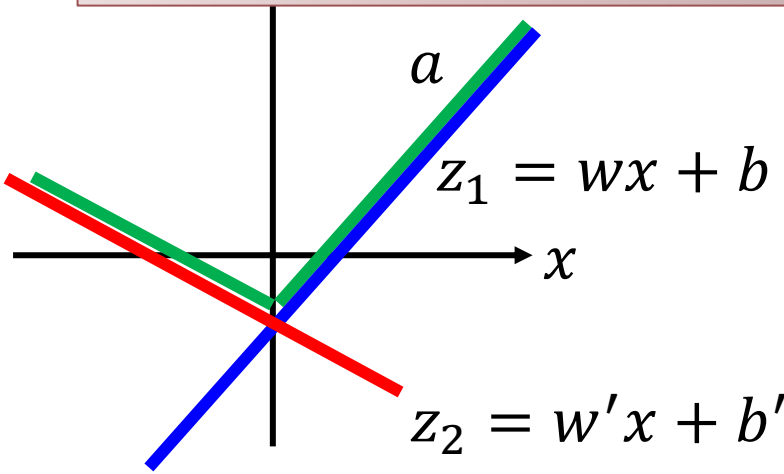
ReLU is a special cases of Maxout





Learnable Activation Function





Why Convolution?
Load Sharing

Convolution

```
n1, n2 = X.
m1, m2 = W.
for i in xr
  for j in
    for ii
      for j
        y[i
```

```
jj]
```

$$x = [x_0, x_1, ..., x_9]^T$$

$$w = [w_0, w_1, w_2]^T$$

$$W = \begin{bmatrix} w_0 & 0 & 0 & ... \\ w_1 & w_0 & 0 & ... \\ w_2 & w_1 & w_0 & ... \\ 0 & w_2 & w_1 & ... \\ 0 & 0 & w_2 & ... \\ ... & ... & ... & ... \end{bmatrix}_{10 \times 10}$$

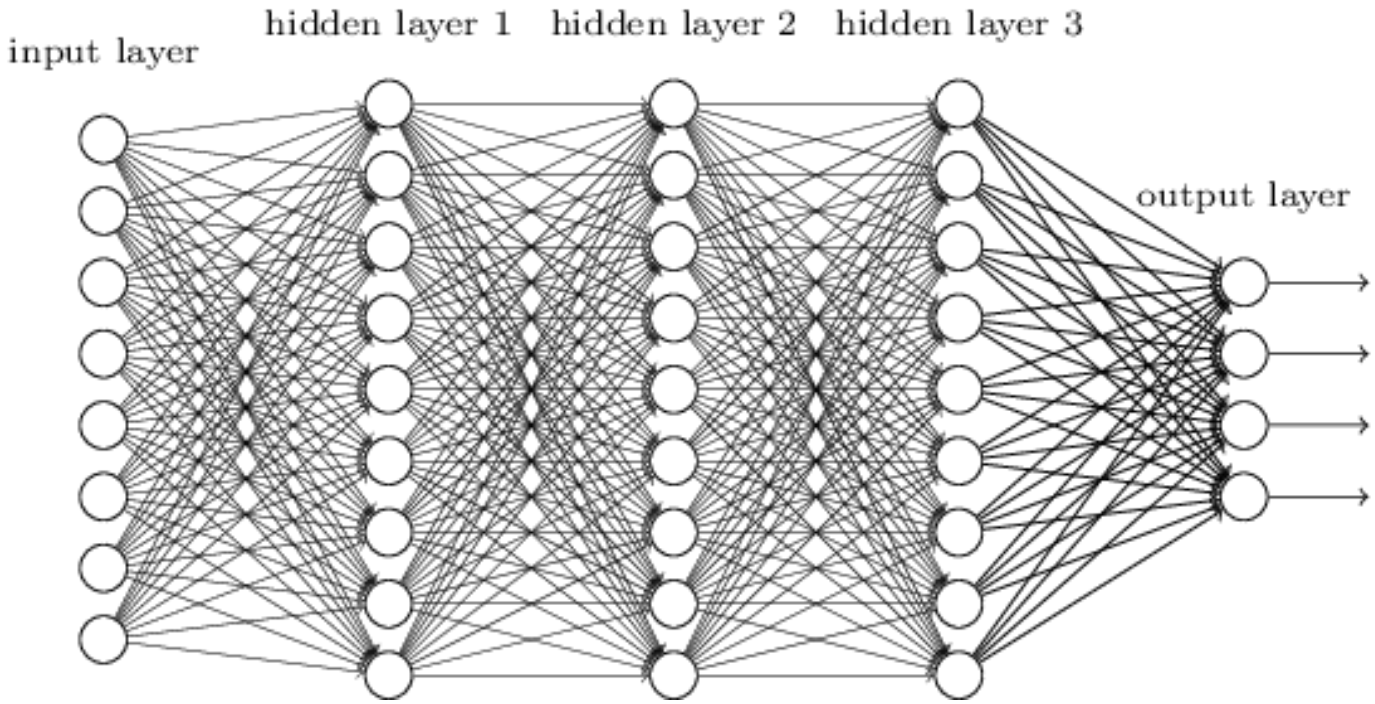
Why Convolution?

- Translation invariance

- Object recognition:
  - Feed-forward NN has different weights everywhere and for every input location.
  - If we train it on left image, it wouldn't recognize the right image

Lecture 5 Smaller Network: CNN

- We know it is good to learn a small model.
- From this fully connected model, do we really need all the edges?
- Can some of these be shared?

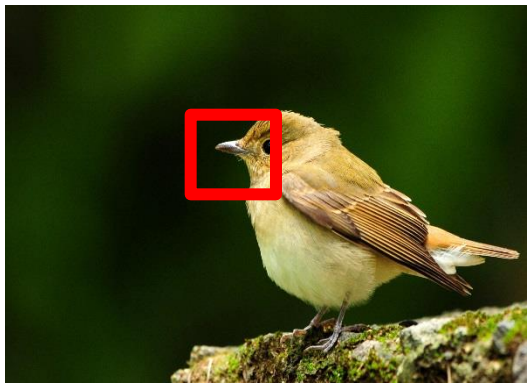


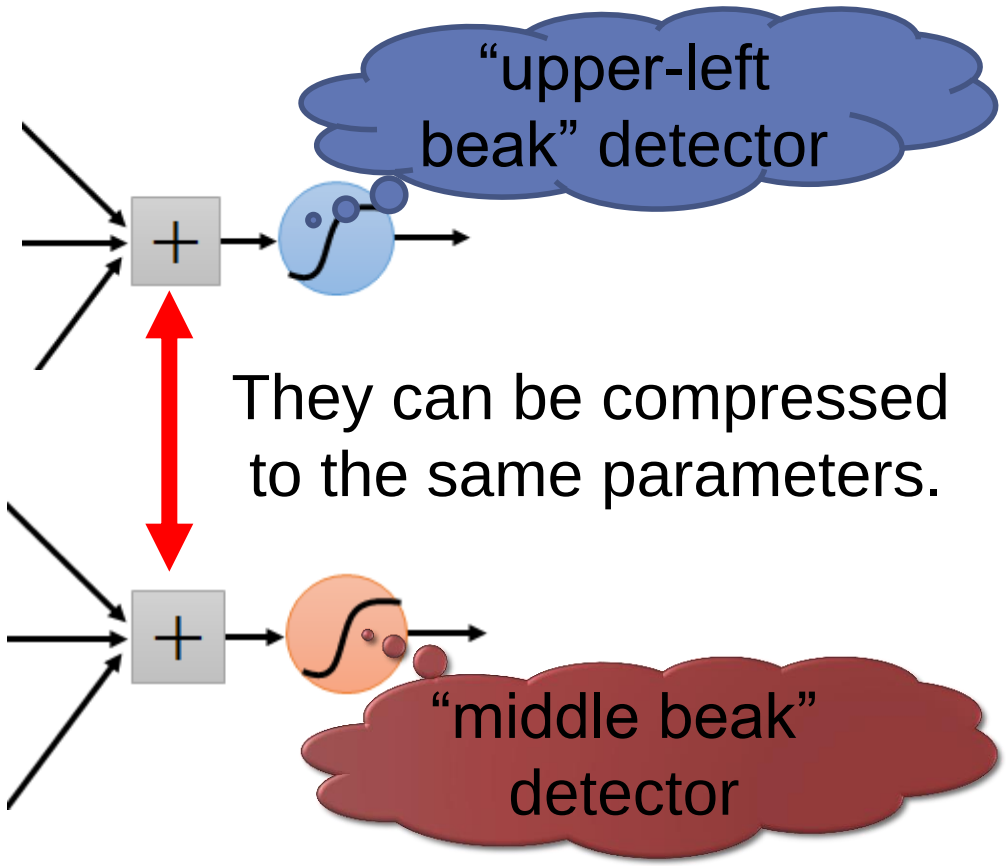
Consider learning an image:

- Some patterns are much smaller than the whole image

Can represent a small region with fewer parameters

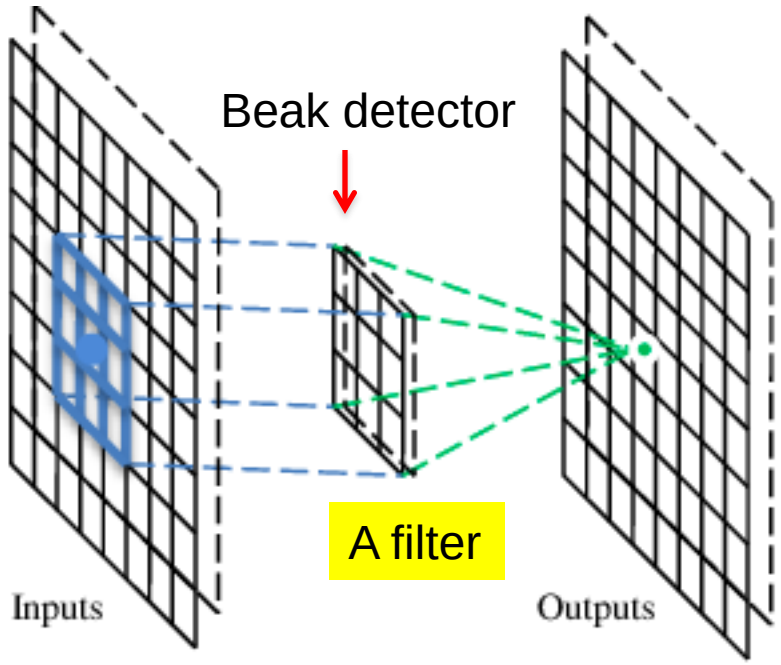
Same pattern appears in different places:
They can be compressed!
What about training a lot of such “small” detectors and each detector must “move around”.





A convolutional layer

A CNN is a neural network with some convolutional layers (and some other layers). A convolutional layer has a number of filters that does convolutional operation.



Convolution

These 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

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

Filter 2

⋮ ⋮

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

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

Dot product → 3 -1

6 x 6 image

Convolution

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

Filter 1

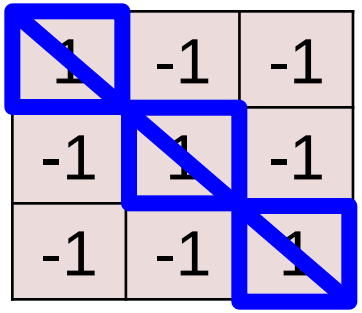
If stride=2

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

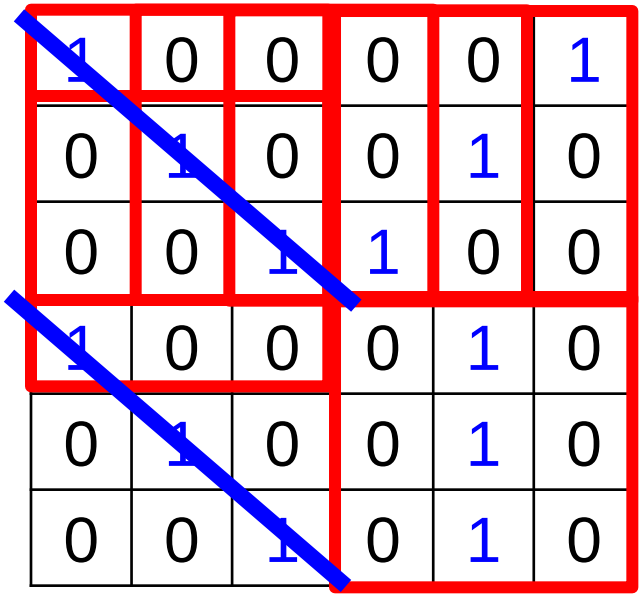
3 -3

Convolution

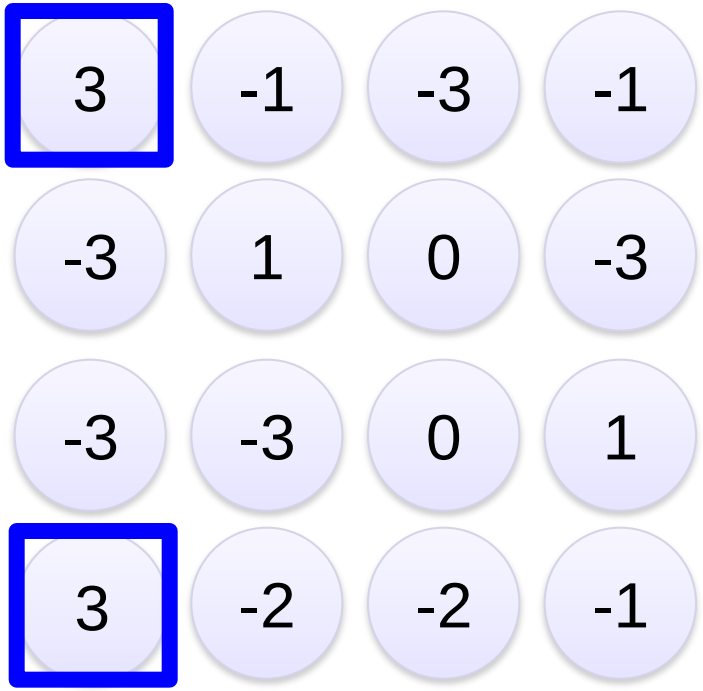


Filter 1

stride=1



6 x 6 image



Convolution

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

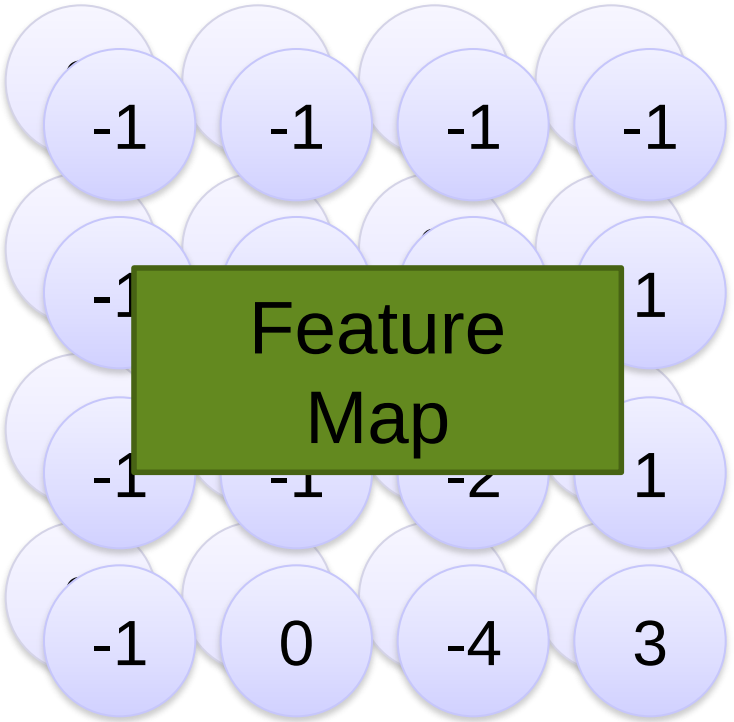
Filter 2

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

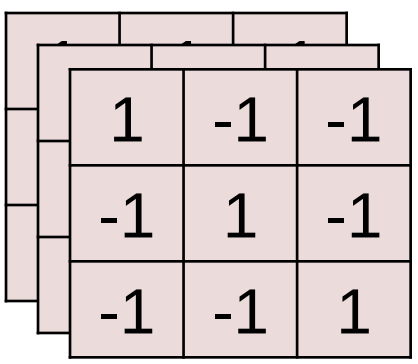
6 x 6 image

Repeat this for each filter

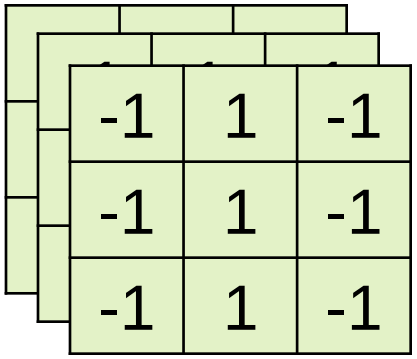


Two 4 x 4 images
Forming 2 x 4 x 4 matrix

Color image: RGB 3 channels

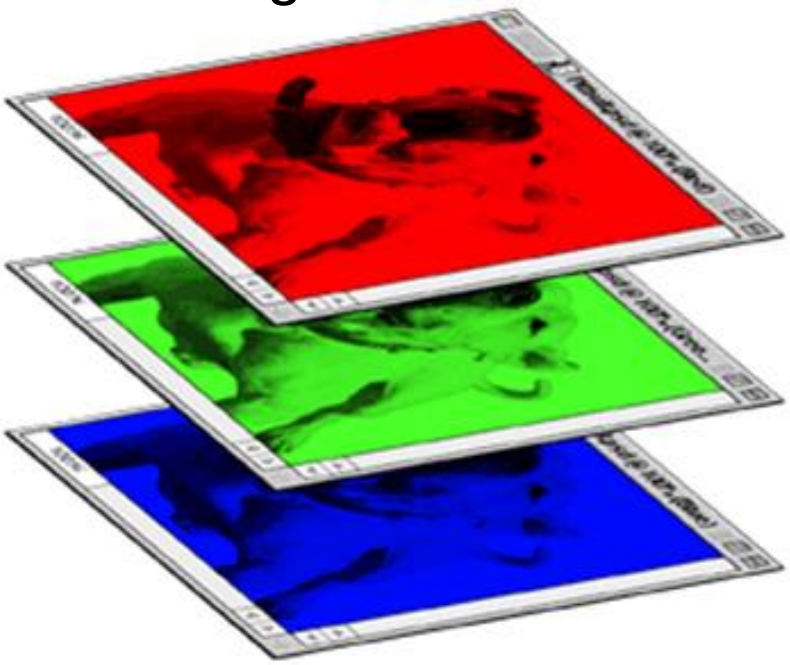


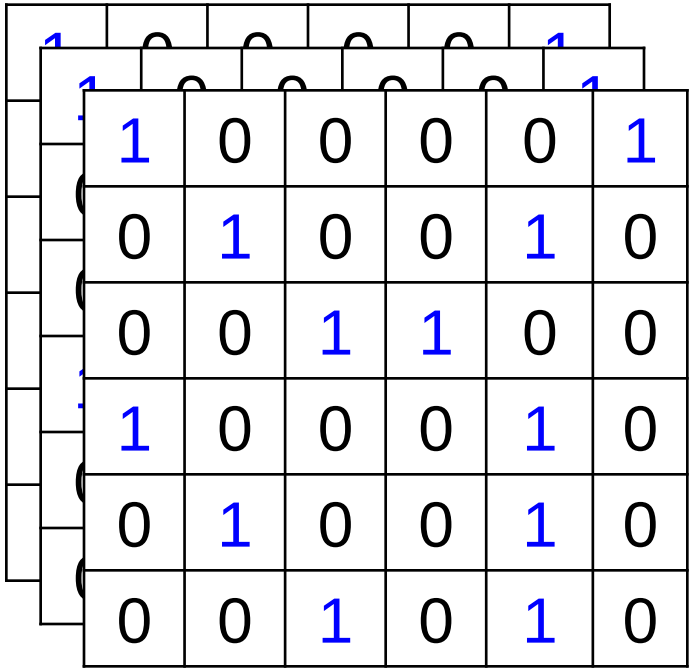
Filter 1



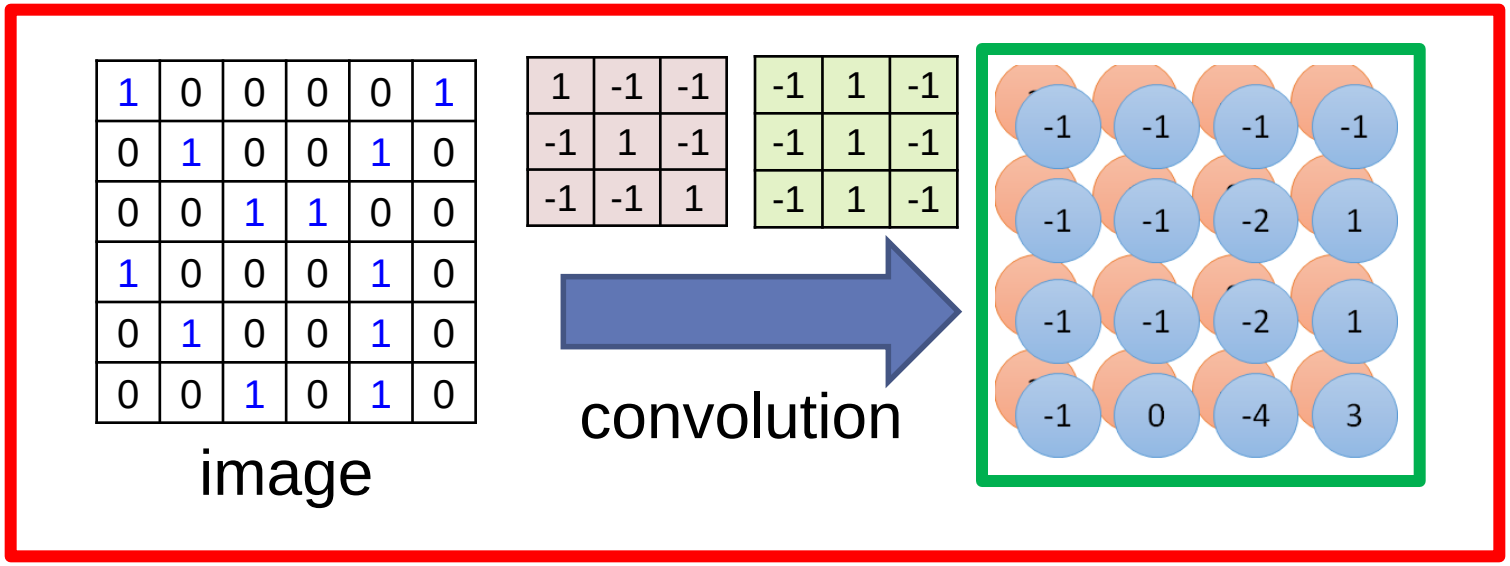
Filter 2

Color image

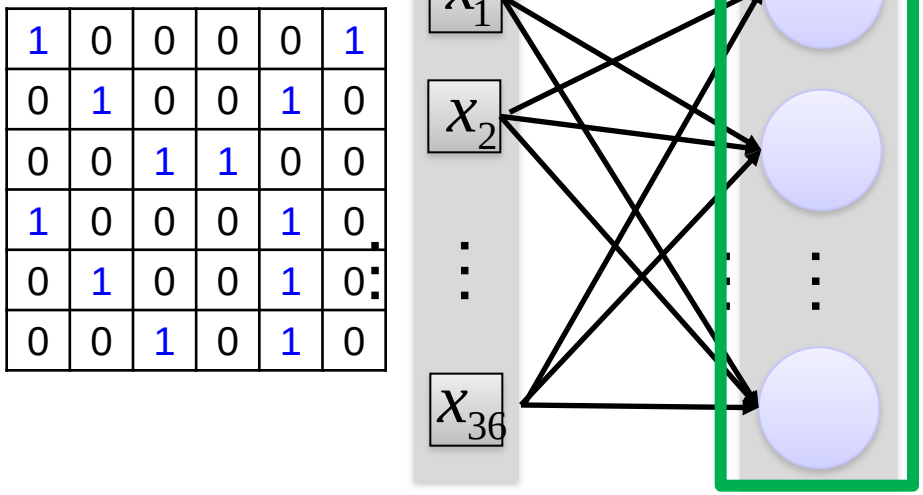


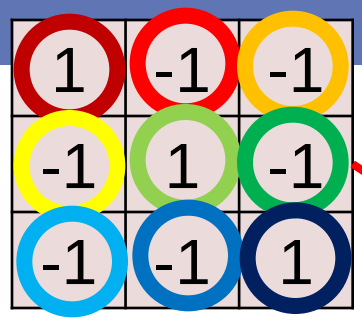


Convolution v.s. Fully Connected



Fully-connected





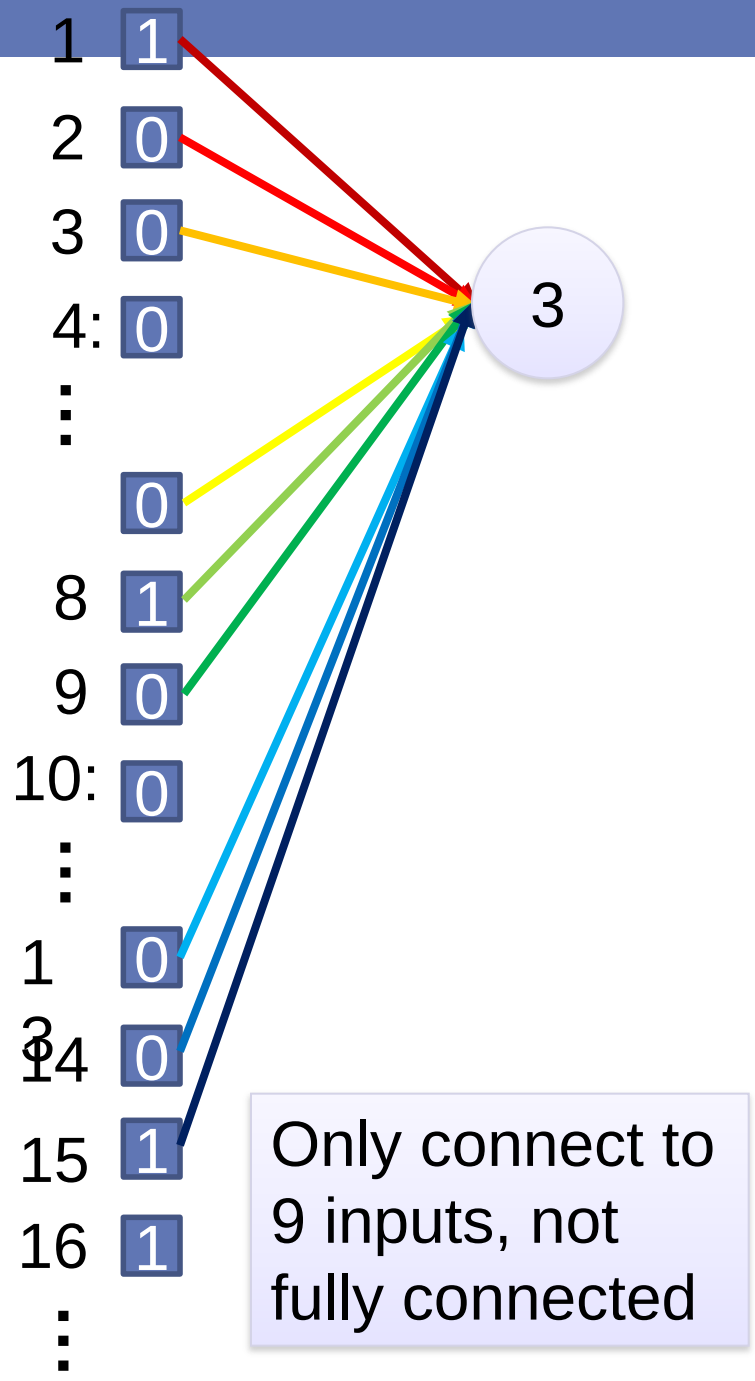
Filter 1

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

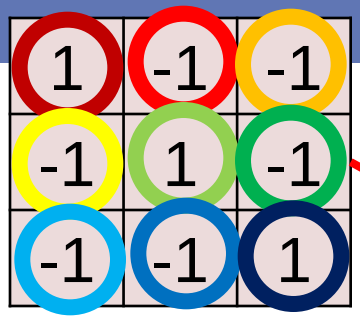
6 x 6 image

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

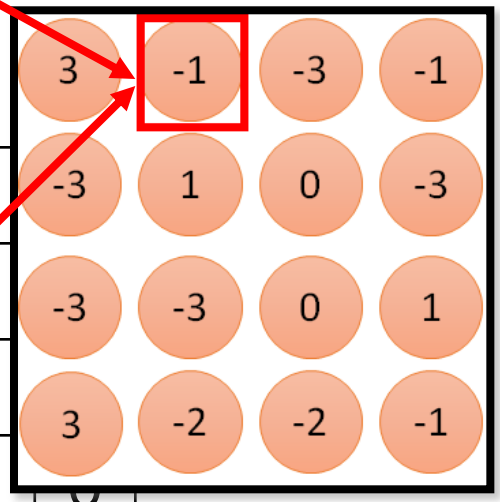
fewer parameters!



Only connect to 9 inputs, not fully connected



Filter 1

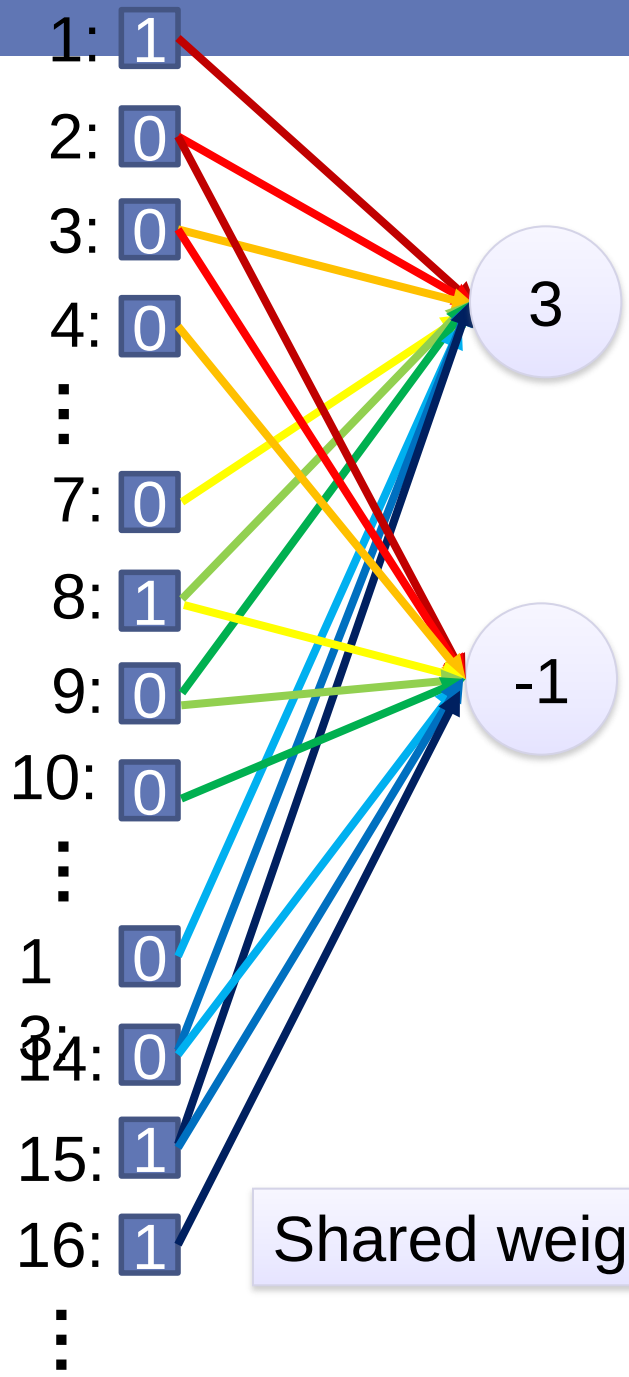


1	0	0	0	0	
0	1	0	0	1	
0	0	1	1	0	
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

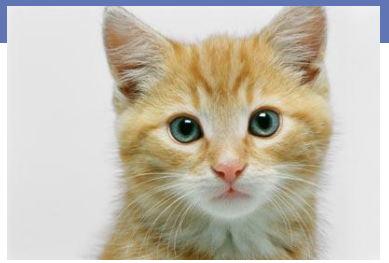
Fewer parameters

Even fewer parameters



Shared weights

The whole CNN



cat dog ……

Convolution

Max Pooling

Convolution

Max Pooling

Can repeat many times

Flattened

Fully Connected Feedforward network

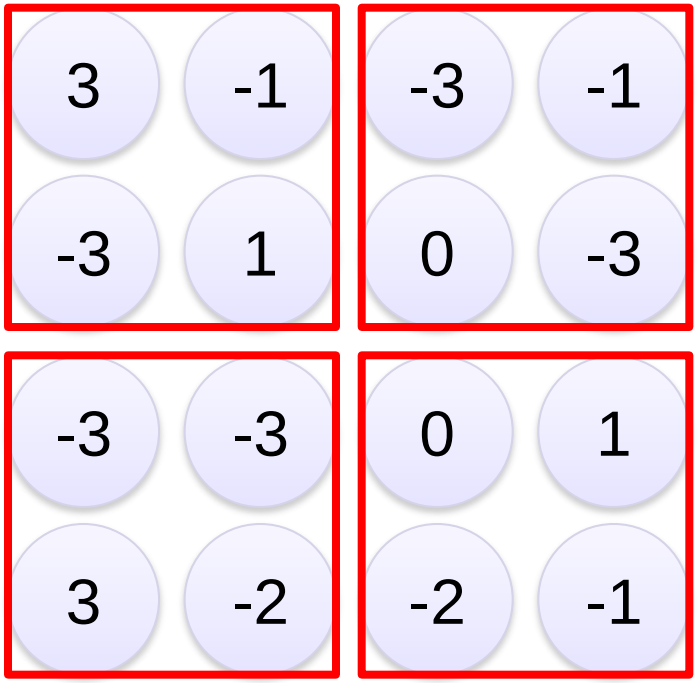
Max Pooling

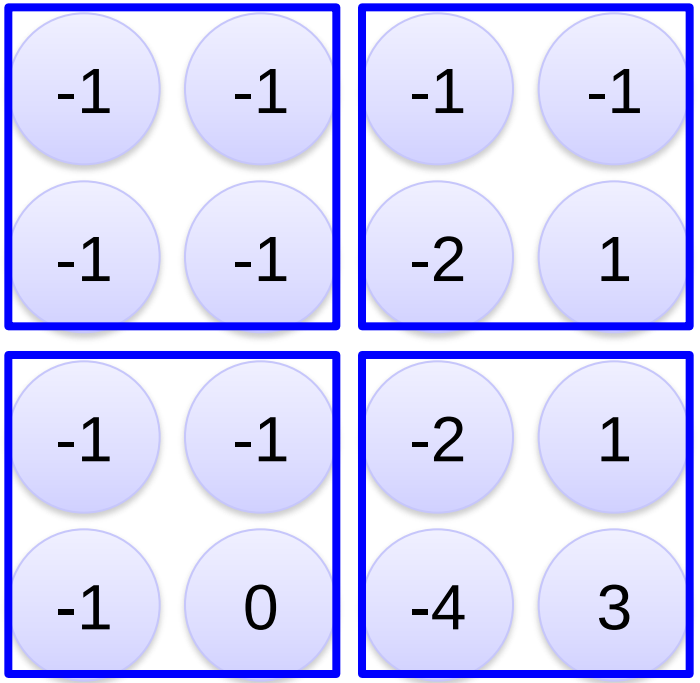
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

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

Filter 2





Why Pooling

- Subsampling pixels will not change the object

bird

Subsampling

bird

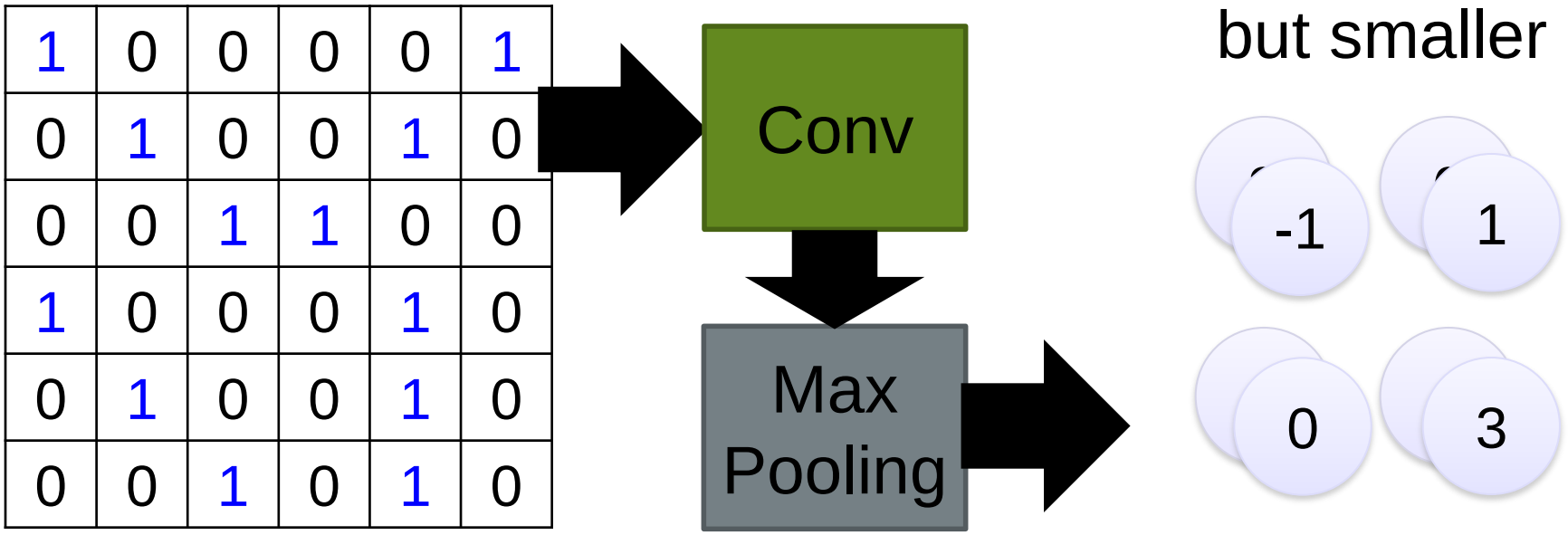
We can subsample the pixels to make image smaller

fewer parameters to characterize the image

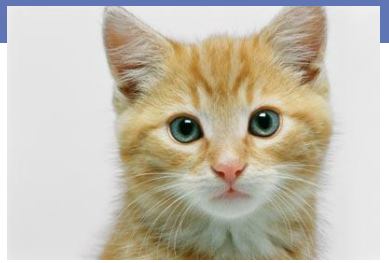
A CNN compresses a fully connected network in two ways:

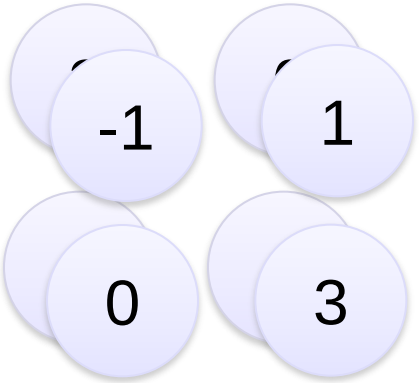
- Reducing number of connections
- Shared weights on the edges
- Max pooling further reduces the complexity

Max Pooling


1 0 0 0 0 1
0 1 0 0 1 0
0 0 1 1 0 0
1 0 0 0 1 0
0 1 0 0 1 0
0 0 1 0 1 0
Conv
Max Pooling
New image but smaller
-1
1
0
3
6 x 6 image
2 x 2 image
Each filter is a channel

The whole CNN





A new image

Convolution

Max Pooling

Convolution

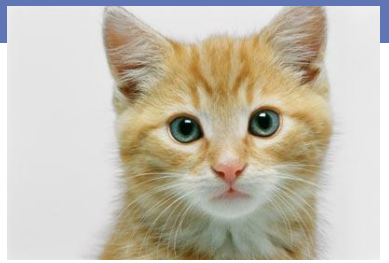
Max Pooling

Can repeat many times

Smaller than the original image

The number of channels is the number of filters

The whole CNN



cat dog ……

Convolution

Max Pooling

A new image

Convolution

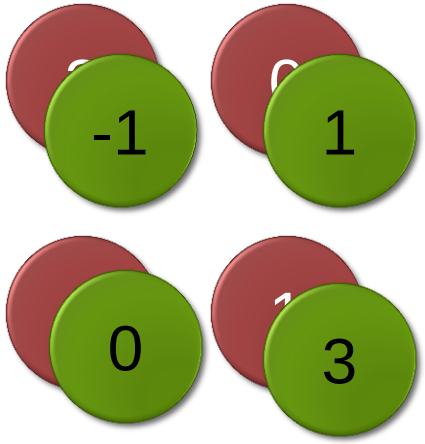
Max Pooling

A new image

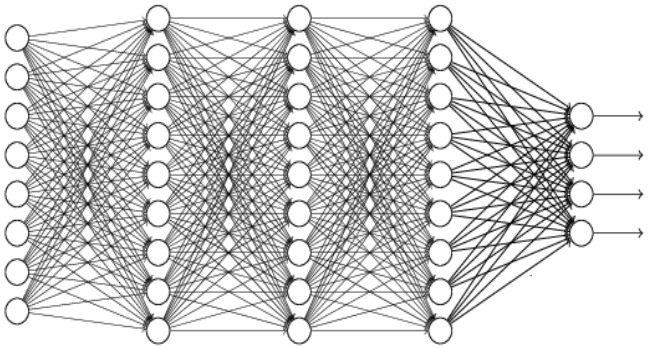
Flattened

Fully Connected Feedforward network

Flattening



Flattened

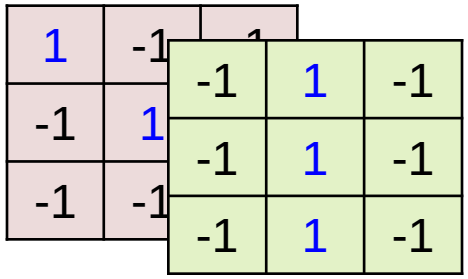


Fully Connected
Feedforward network

CNN in Keras

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

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



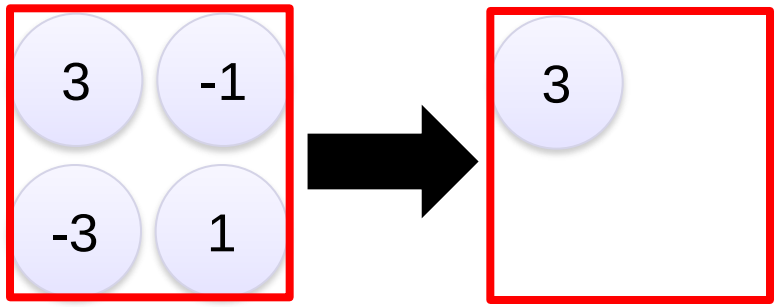
There are 25 3x3 filters.

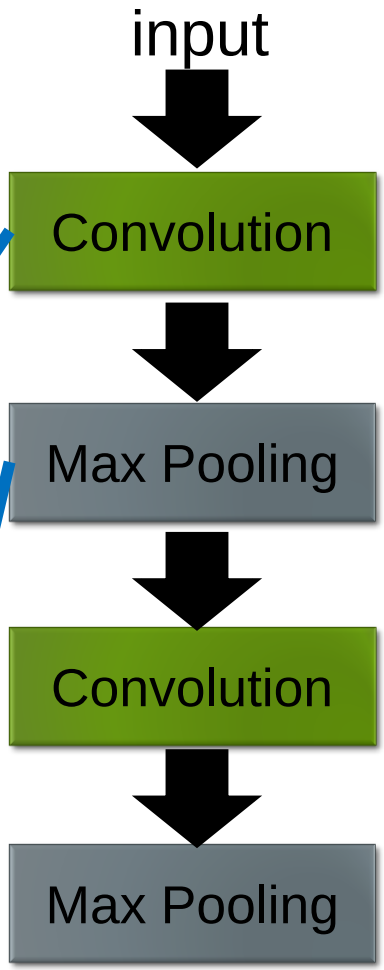
Input_shape = (28 , 28 , 1)

28 x 28 pixels

1: black/white, 3: RGB

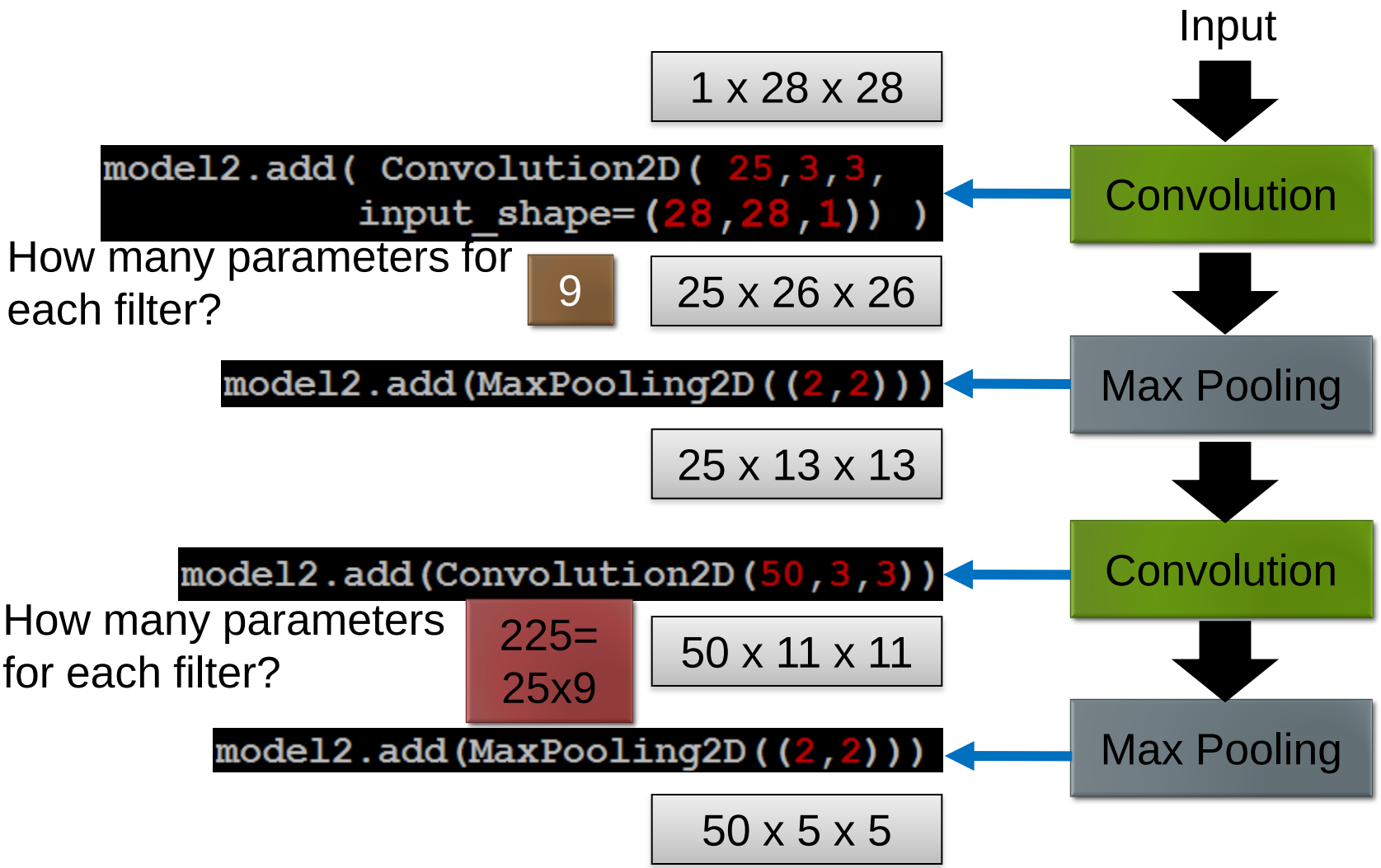
```
model2.add(MaxPooling2D((2,2)))
```





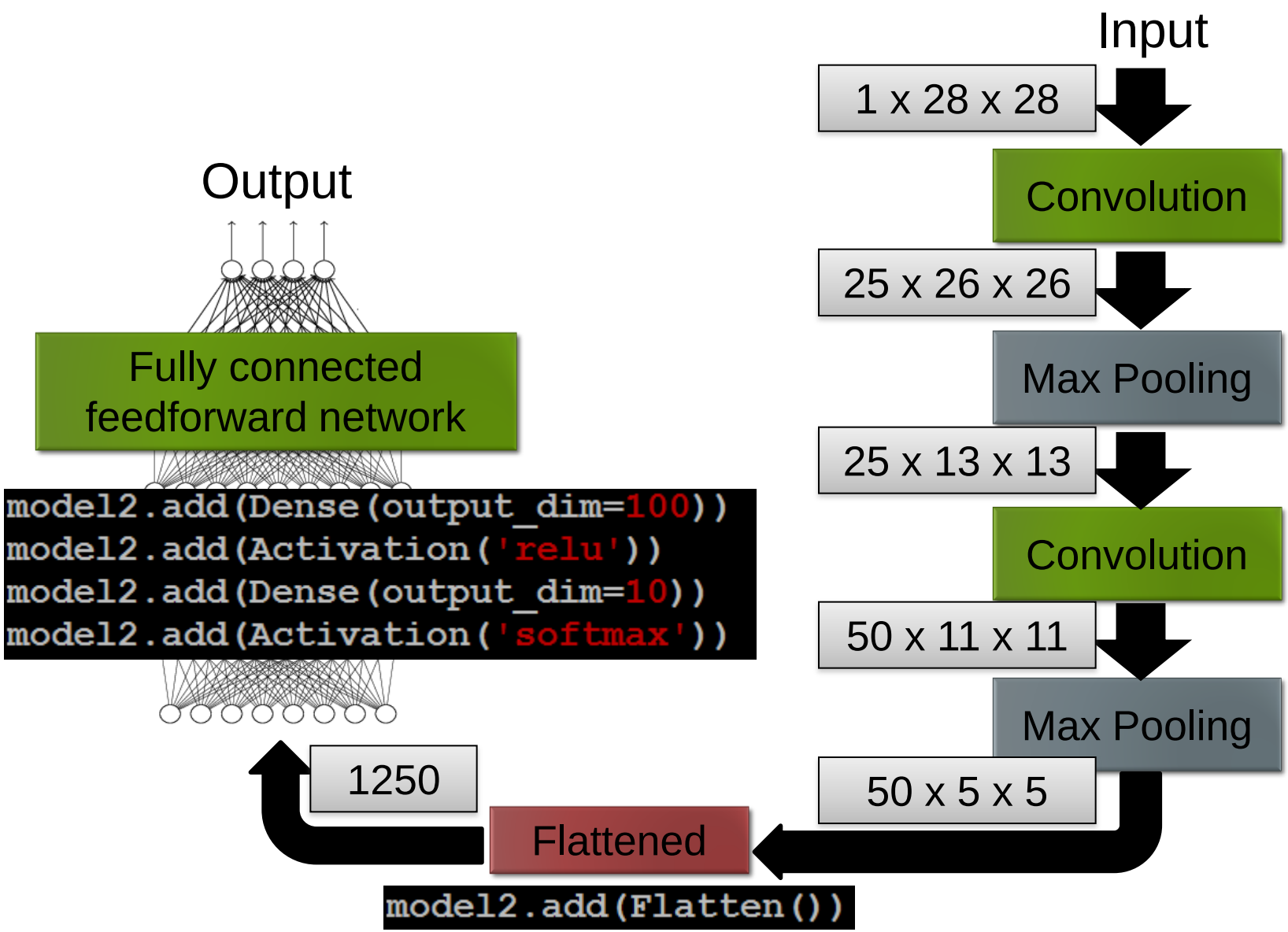
CNN in Keras

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



CNN in Keras

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



AlphaGo

19 x 19 matrix

Black: 1

white: -1

none: 0

Neural Network

Next move

(19 x 19 positions)

Fully-connected feedforward network can be used

But CNN performs much better

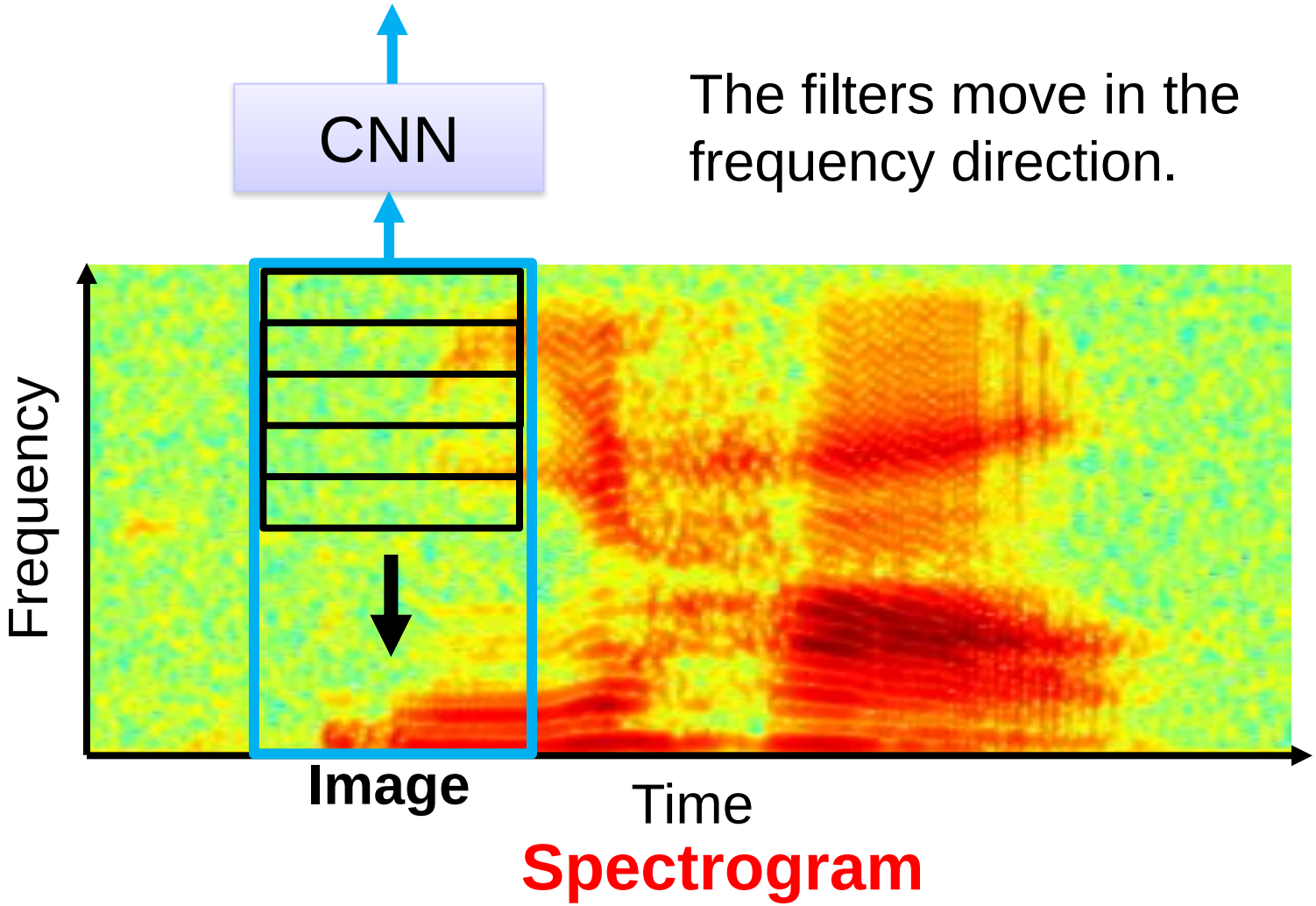
AlphaGo's policy network

The following is quotation from their Nature article:

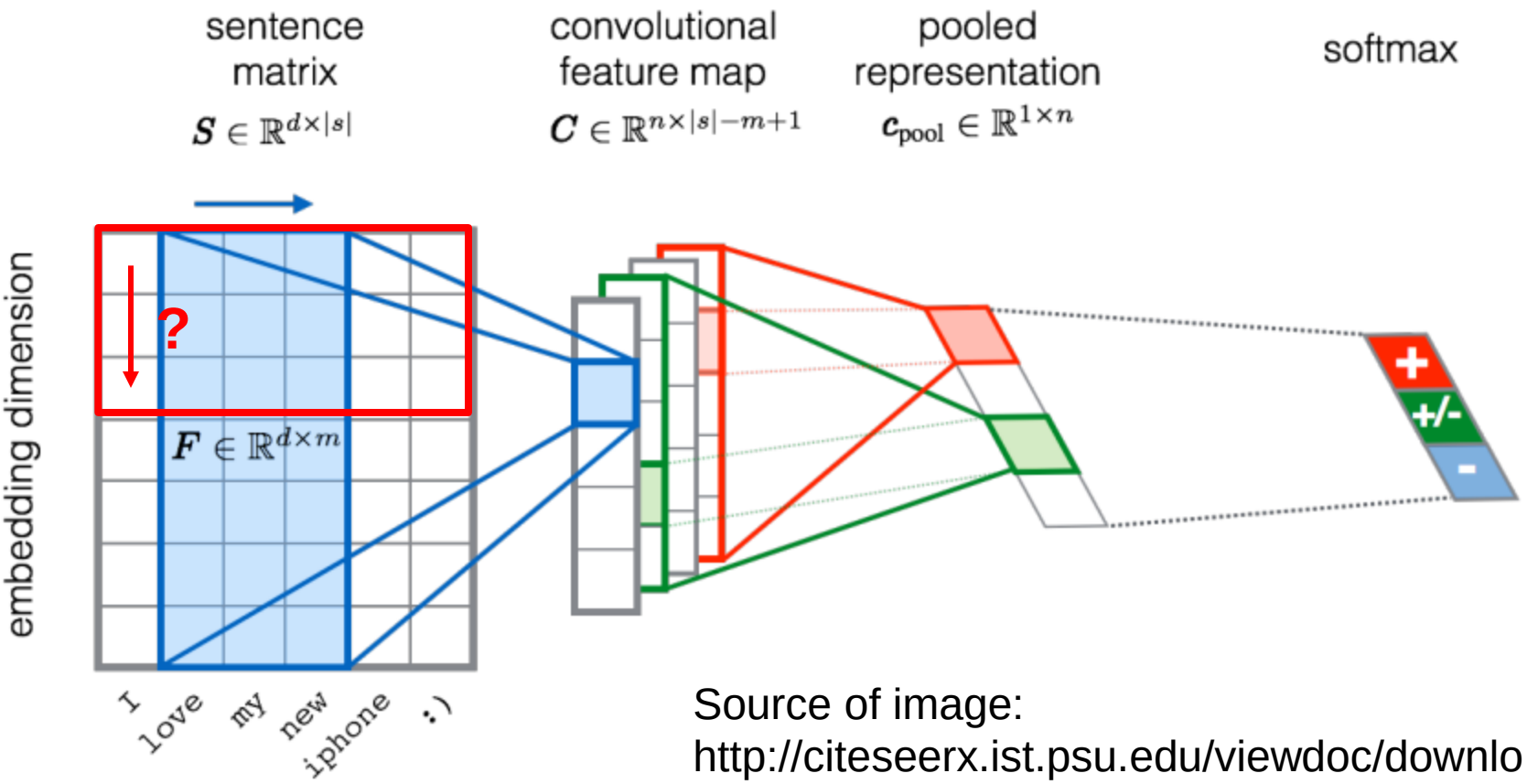
Note: AlphaGo does not use Max Pooling.

Neural network architecture. The input to the policy network is a $19 \times 19 \times 48$ image stack consisting of 48 feature planes. The first hidden layer zero pads the input into a 23×23 image, then convolves k filters of kernel size 5×5 with stride 1 with the input image and applies a rectifier nonlinearity. Each of the subsequent hidden layers 2 to 12 zero pads the respective previous hidden layer into a 21×21 image, then convolves k filters of kernel size 3×3 with stride 1, again followed by a rectifier nonlinearity. The final layer convolves 1 filter of kernel size 1×1 with stride 1, with a different bias for each position, and applies a softmax function. The match version of AlphaGo used $k = 192$ filters; Fig. 2b and Extended Data Table 3 additionally show the results of training with $k = 128$, 256 and 384 filters.

CNN in speech recognition



CNN in text classification



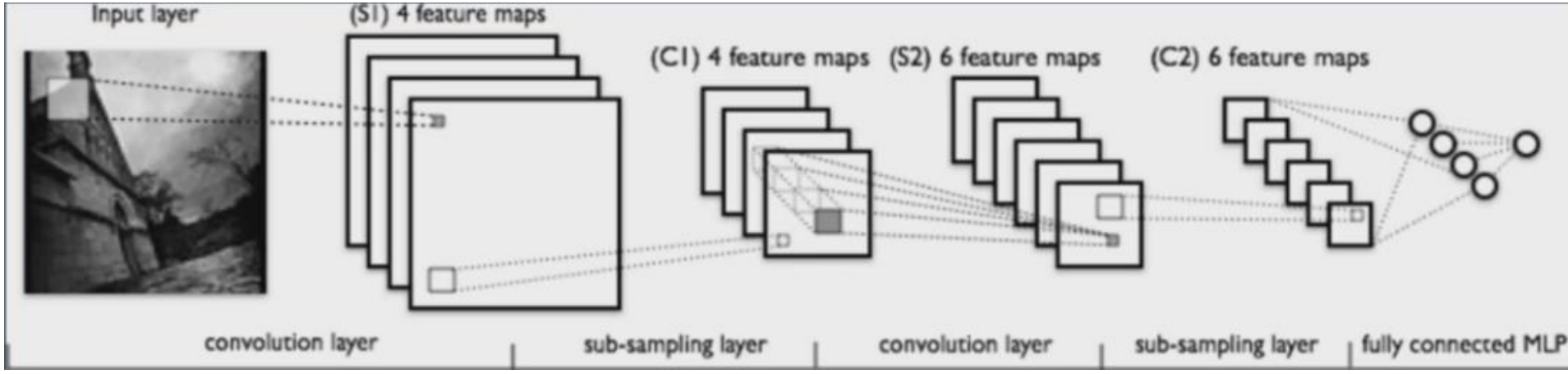
Source of image: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.703.6858&rep=rep1&type=pdf

Advantage of convolution

- Multiplies the same weight everywhere on the image
- Suppose the filter is a feature finder that identifies the car, ie. outputs a comparatively high +ve # , when input is a car.
- It will find a car no matter where it is on the image
- Feed-forward NN contains multiple layers which allow each layer to find successively complex features
- Layers of convolution behave the same. First layer will find small, simple features anywhere on the image(eg. A line or edge). Next layer will find more complex features, and so on
- Weight sharing Better generalization

Architecture of CNN

- After convolution, we just want to know if a feature is found, so take a neighborhood (square) of data and get the max (called maxpooling), (also average pooling)
- Similar to down-sampling or sub-sampling
  - Theano.tensor.signal.downsample.max_pool_2d
  - Tf.nn.max_pool
-



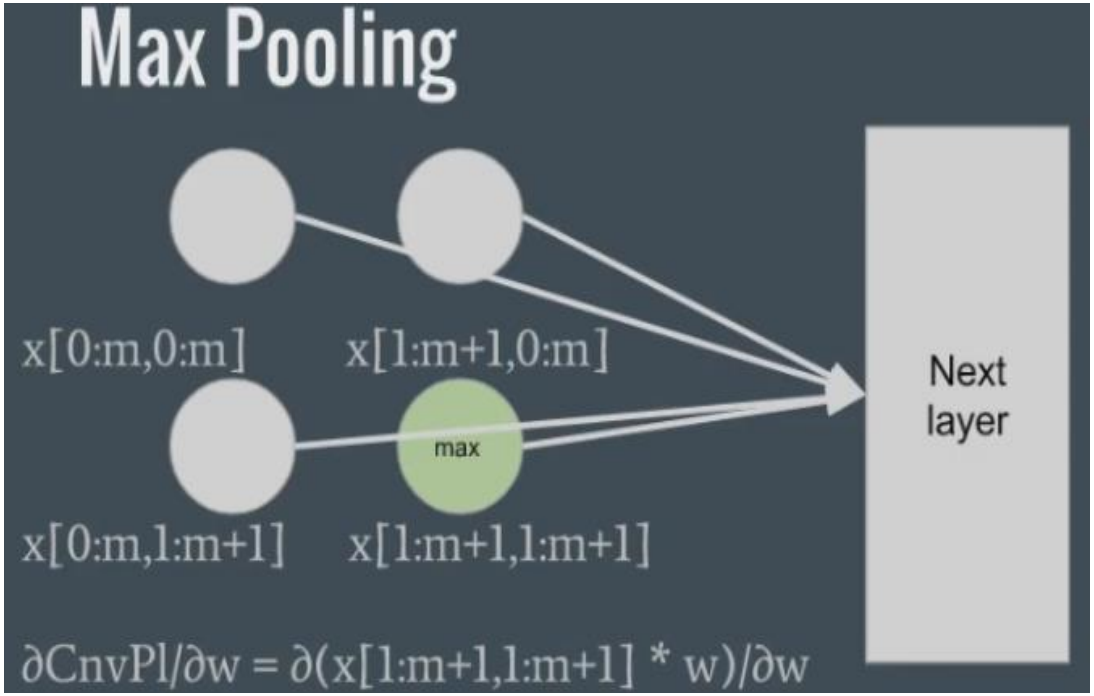
Convolution

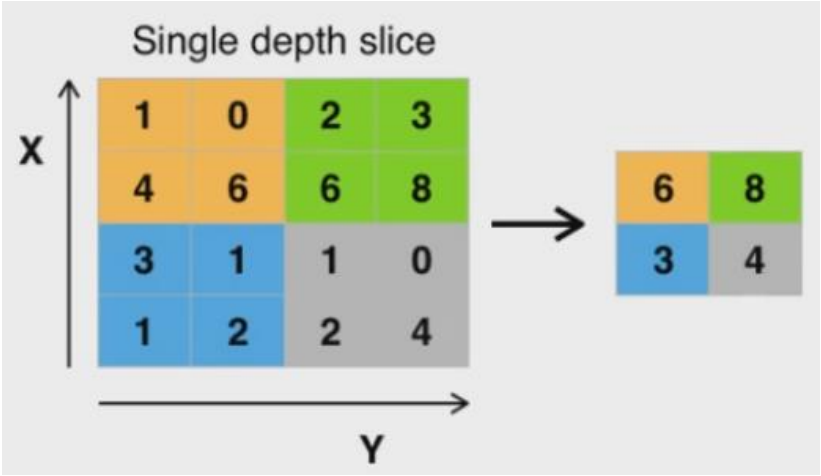
$y(m,n) = \sum_i \sum_j w(i,j)x(m-i,n-j)$

$\partial y(m,n)/\partial w(i,j) = x(m-i,n-j)$

Wow! That was easy!

Max Pooling





Quiz

How many?

- Common question:
- How many ConvPool layers should I include?
- How many feedforward layers should I include?
- What filter sizes should I use?
- What number of feature maps should I use?
- These are all hyperparameters - what is best is specific to your dataset
- Deep Learning part 2 discussed random search, grid search, cross-validation to help you do this