

Introduction to Statistical Learning and Machine Learning

Chap 7 -
Neural Network

Yanwei Fu
SDS, Fudan University



Chap 7 - Neural Network

Main Content:

1. Regularization, Error Back-propagation;
2. Loss function and Optimization;
3. Feed-forward neural networks
(Multi-layer perception,
Convolutional neural networks);



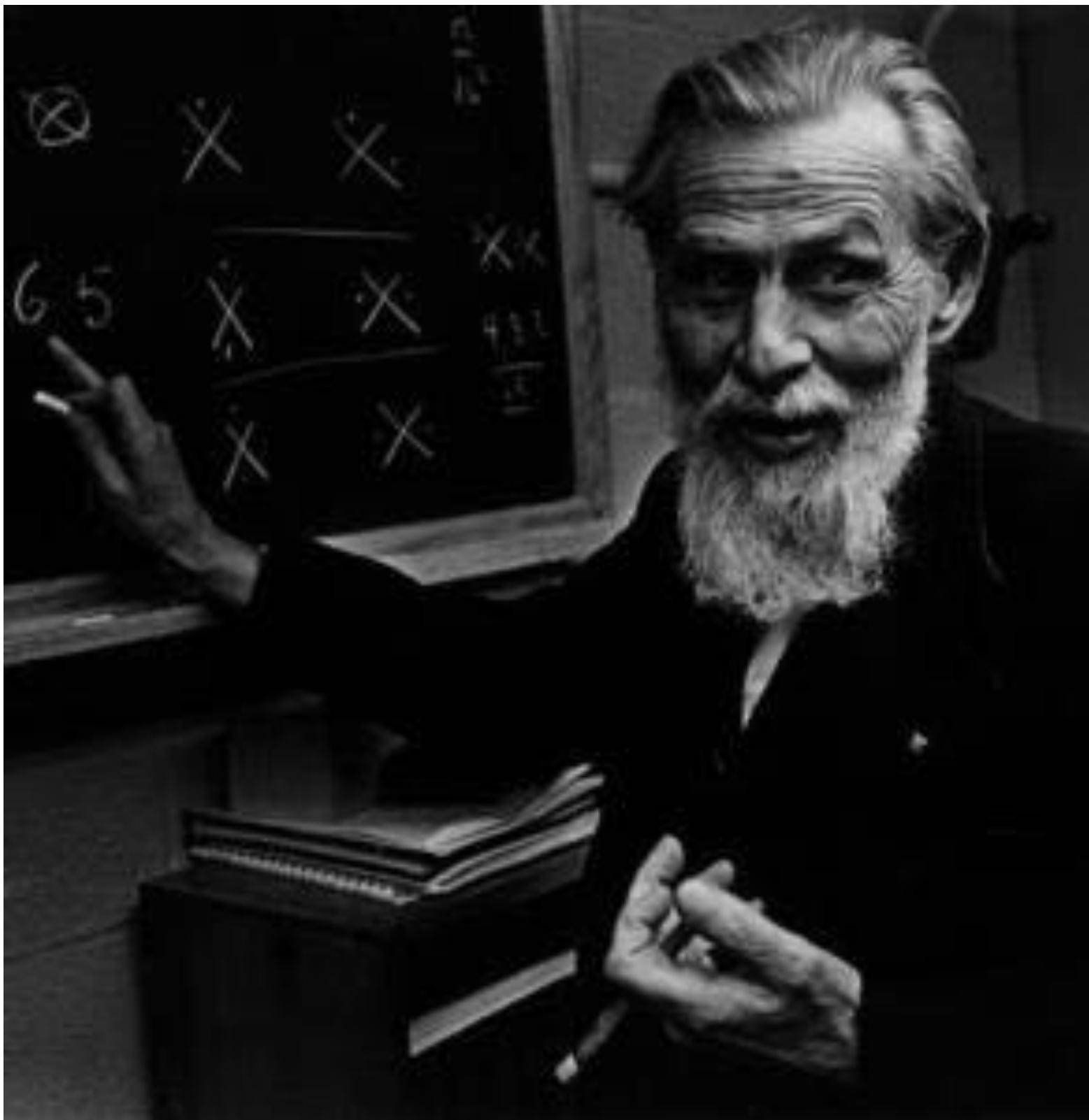
History



神经元进入计算领域的视野



Walter Pitts
(1923-1969)



Warren McCulloch
(1898-1969)

BULLETIN OF
MATHEMATICAL BIOPHYSICS
VOLUME 5, 1943

A LOGICAL CALCULUS OF THE IDEAS IMMANENT IN NERVOUS ACTIVITY

WARREN S. McCULLOCH AND WALTER PITTS

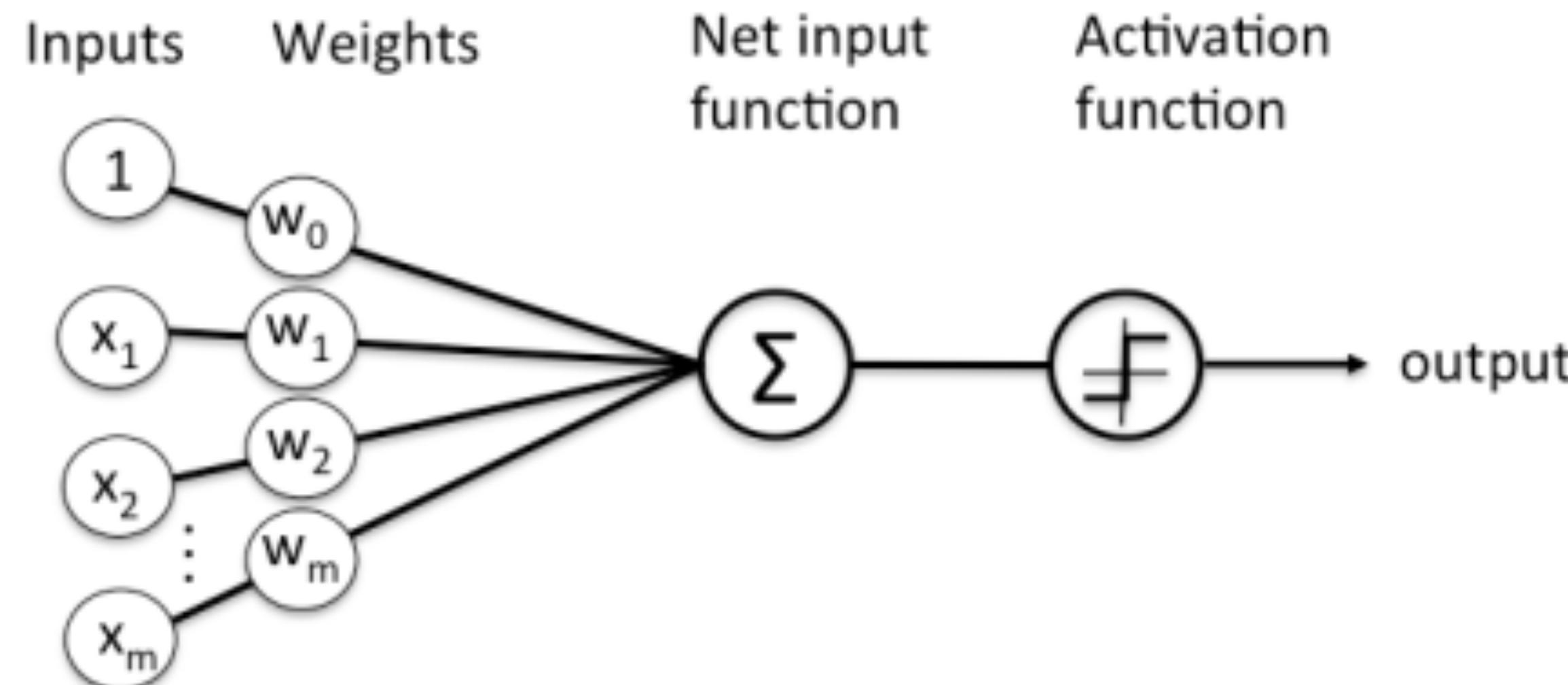
FROM THE UNIVERSITY OF ILLINOIS, COLLEGE OF MEDICINE,
DEPARTMENT OF PSYCHIATRY AT THE ILLINOIS NEUROPSYCHIATRIC INSTITUTE,
AND THE UNIVERSITY OF CHICAGO

Because of the "all-or-none" character of nervous activity, neural events and the relations among them can be treated by means of propositional logic. It is found that the behavior of every net can be described in these terms, with the addition of more complicated logical means for nets containing circles; and that for any logical expression satisfying certain conditions, one can find a net behaving in the fashion it describes. It is shown that many particular choices among possible neurophysiological assumptions are equivalent, in the sense that for every net behaving under one assumption, there exists another net which behaves under the other and gives the same results, although perhaps not in the same time. Various applications of the calculus are discussed.

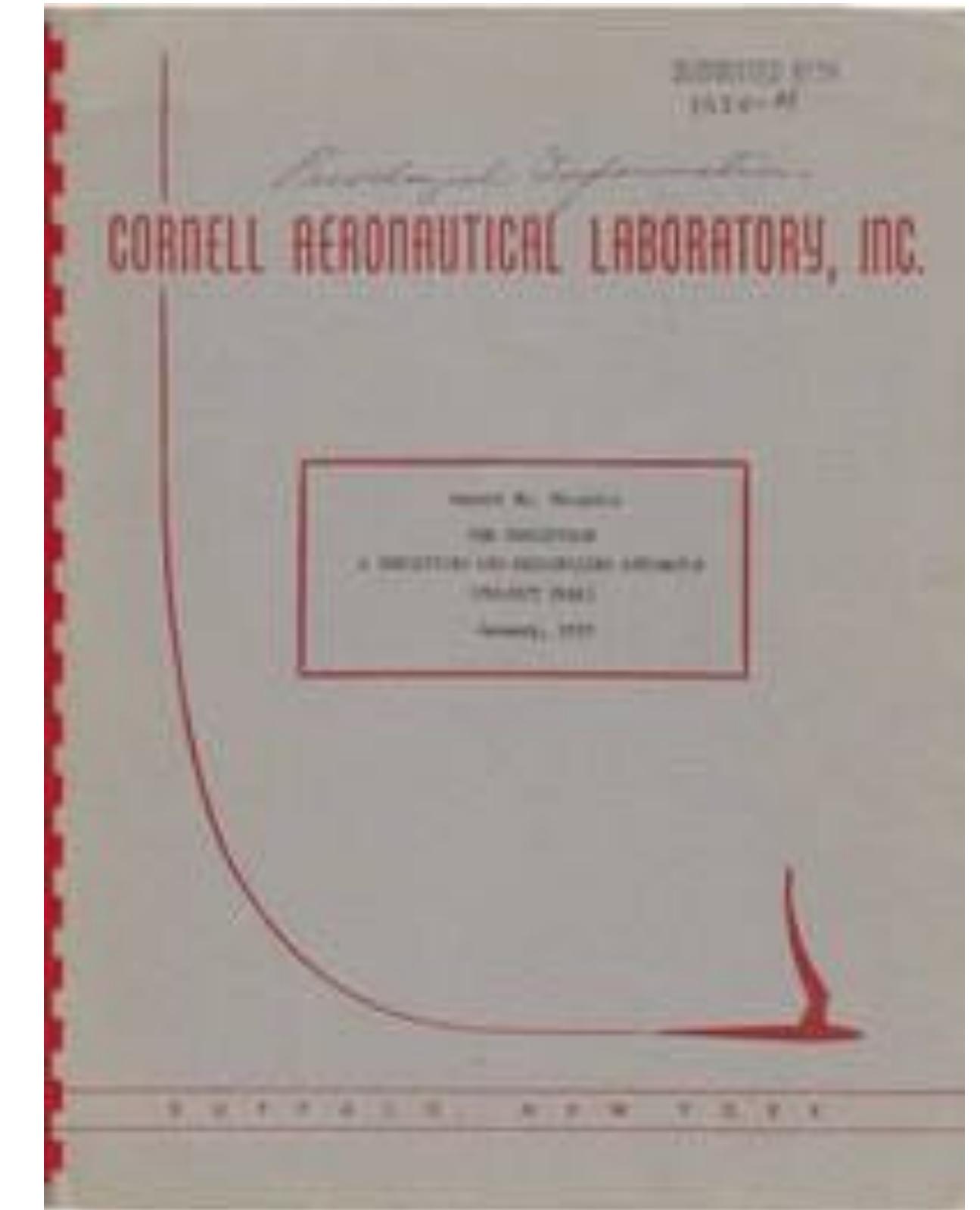
引入感知机 (Perceptron) 概念



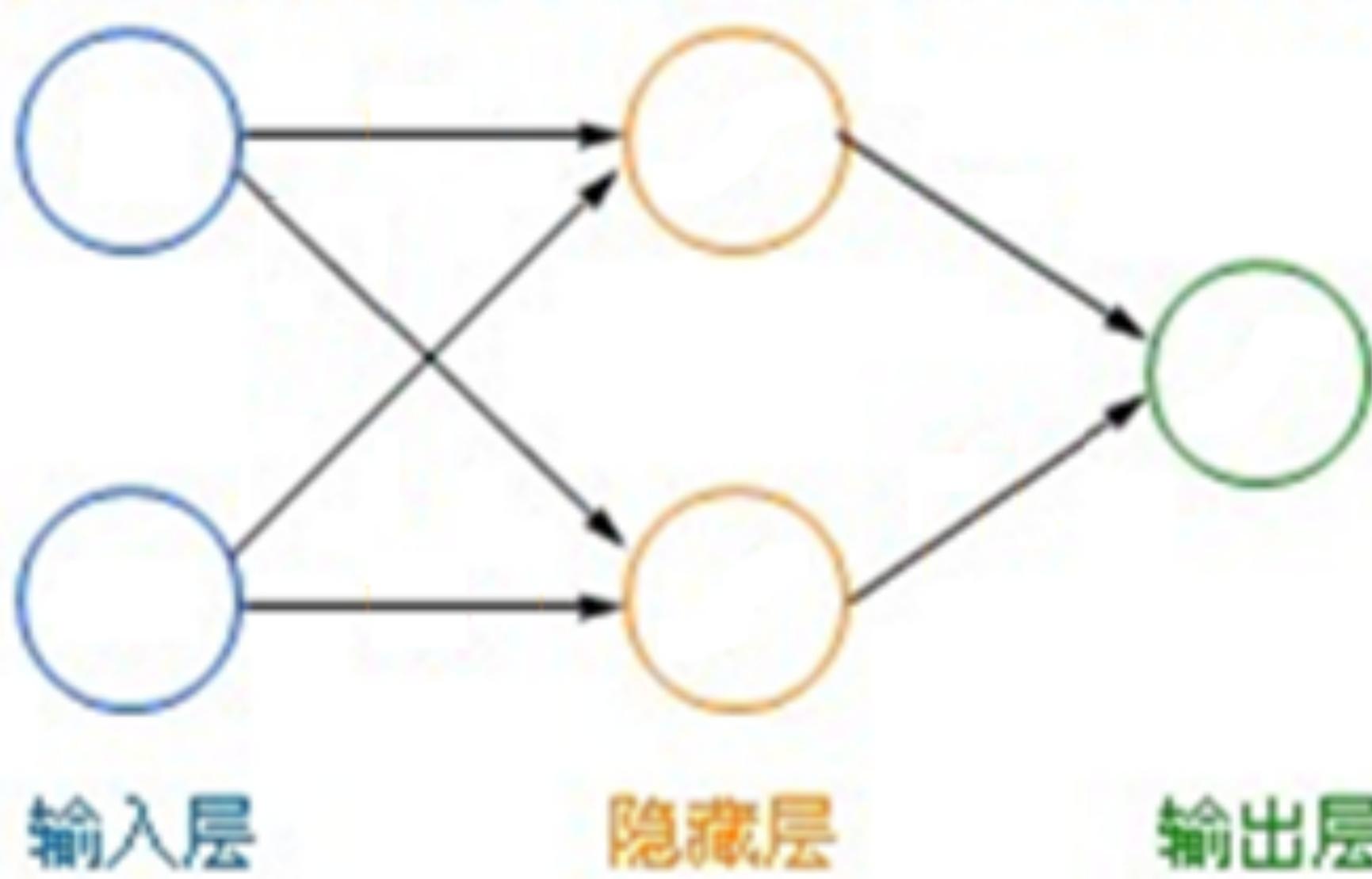
Frank Rosenblatt
(1923-1971)



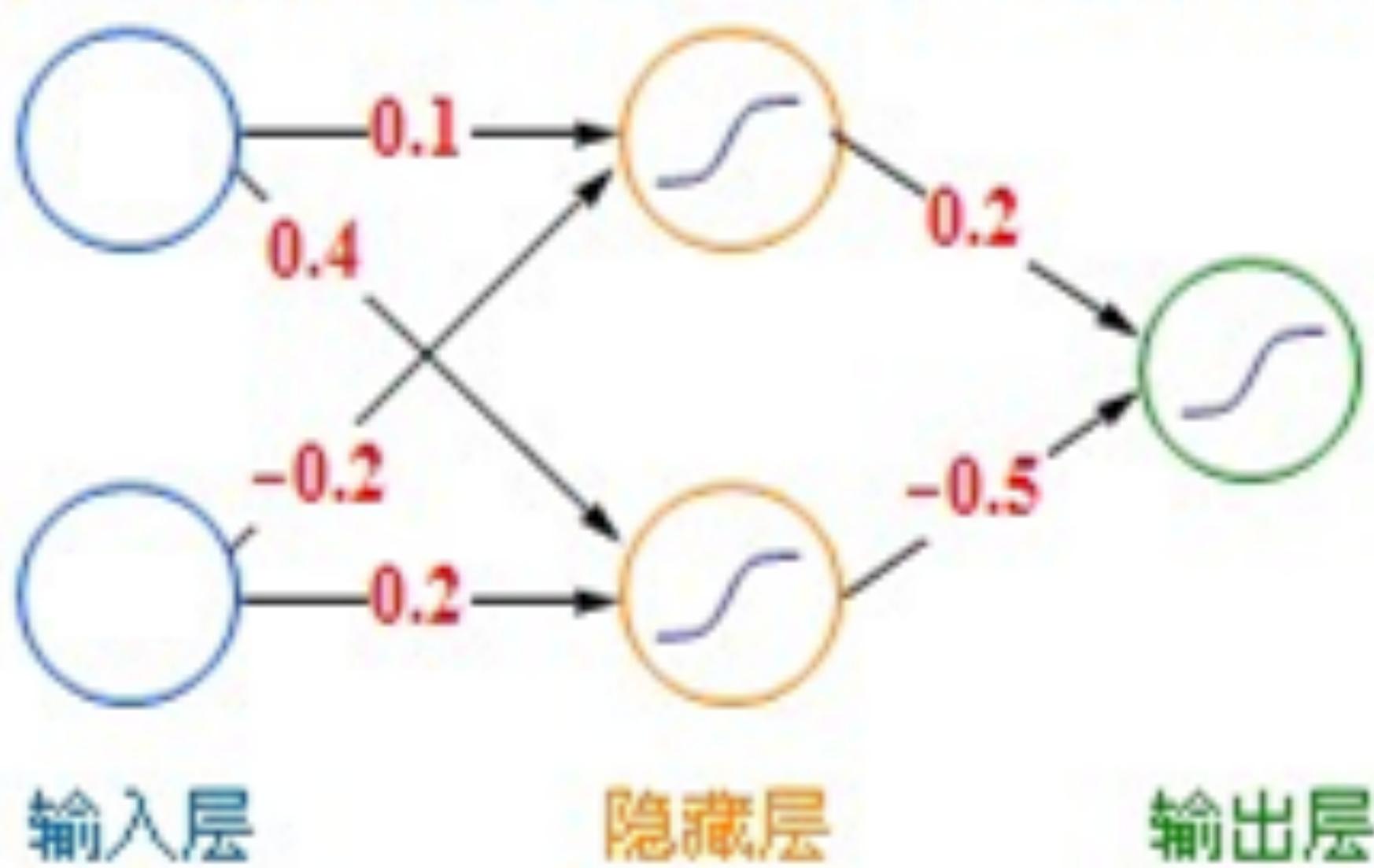
The perceptron, a perceiving and
recognizing automaton Project
Para. Cornell Aeronautical
Laboratory, 1957



神经网络基础模型

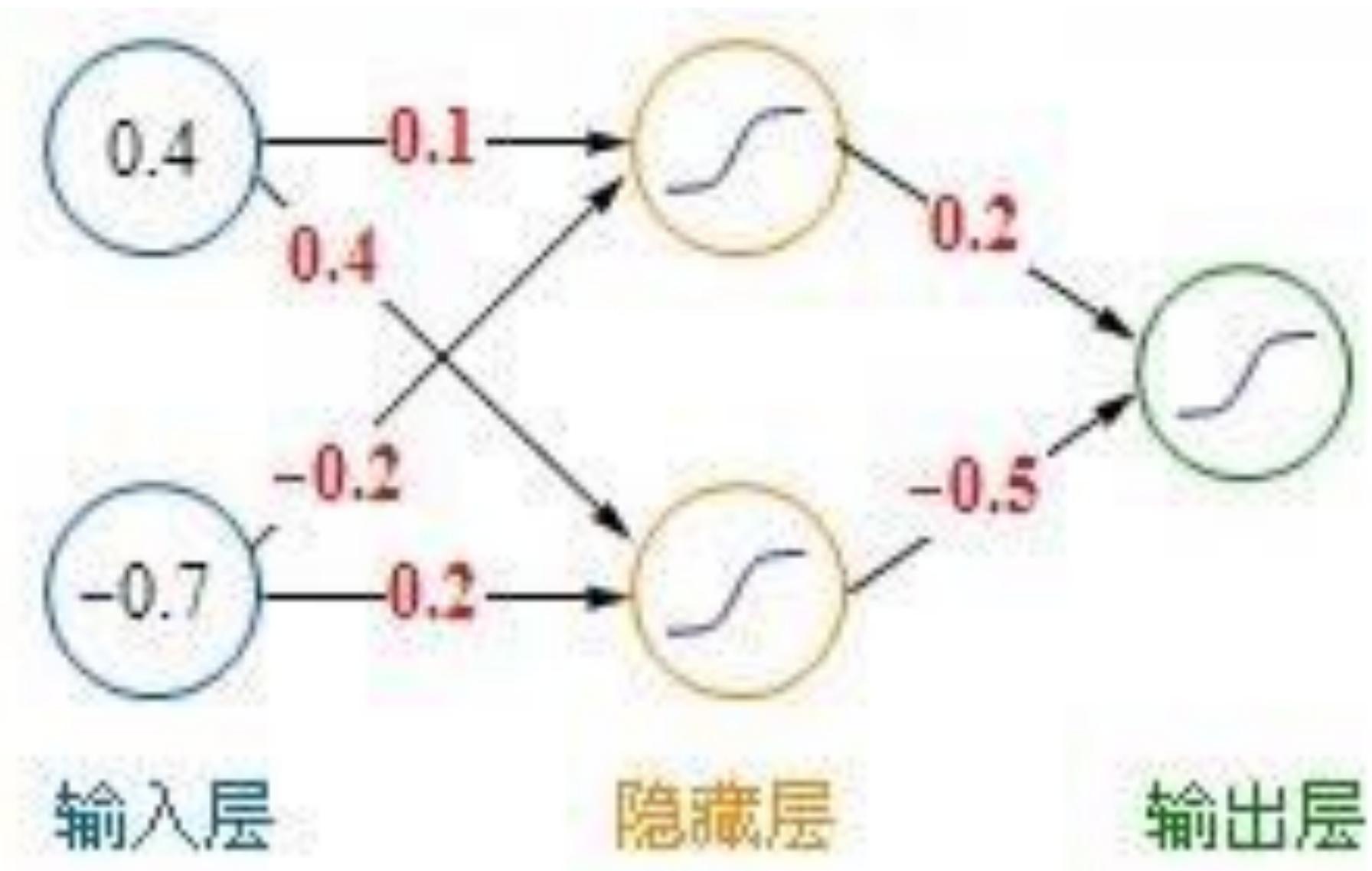


神经网络基础模型

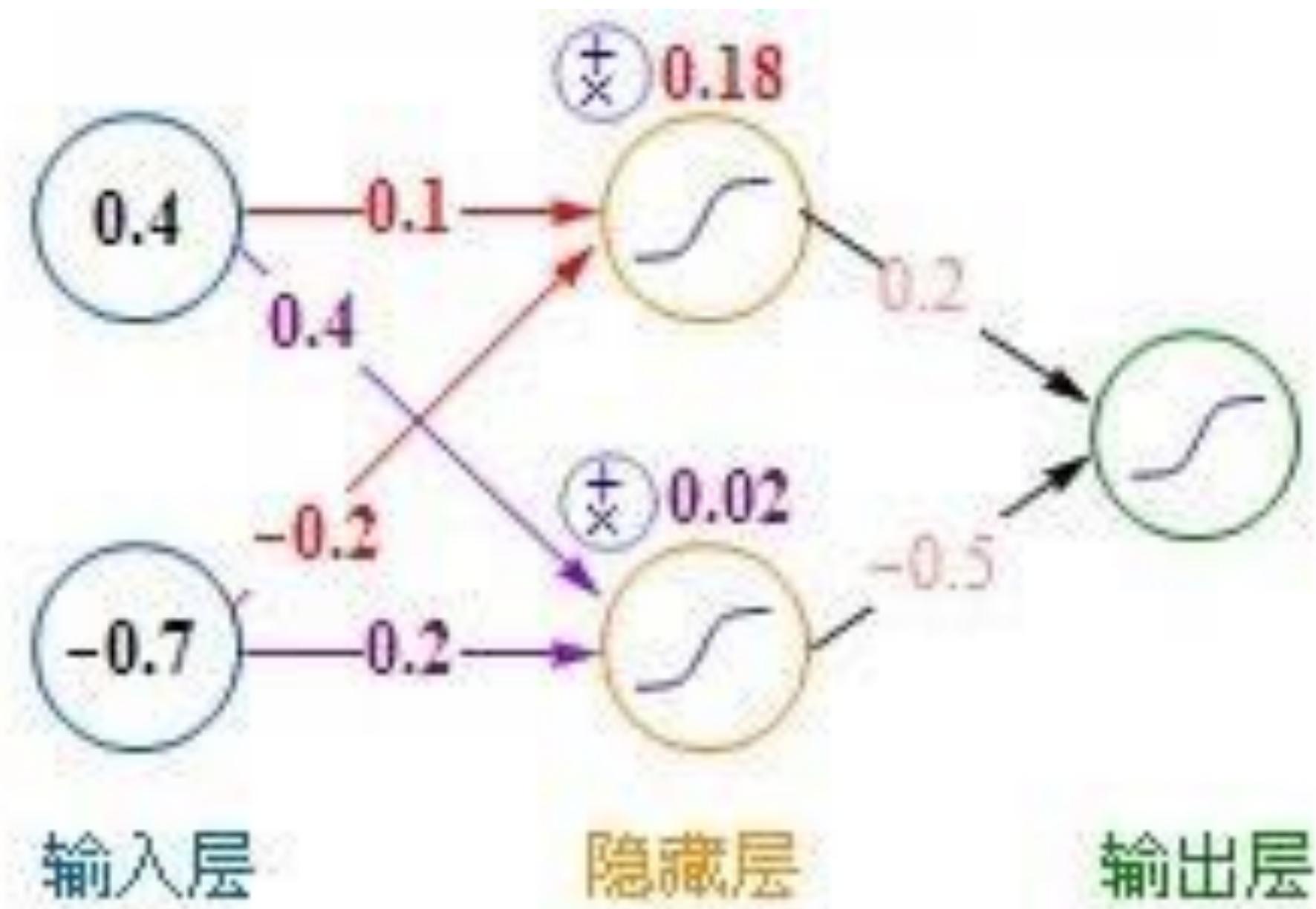


$$y = \text{logsig}(x) = \frac{1}{1 + e^{-x}}$$

神经网络基础模型



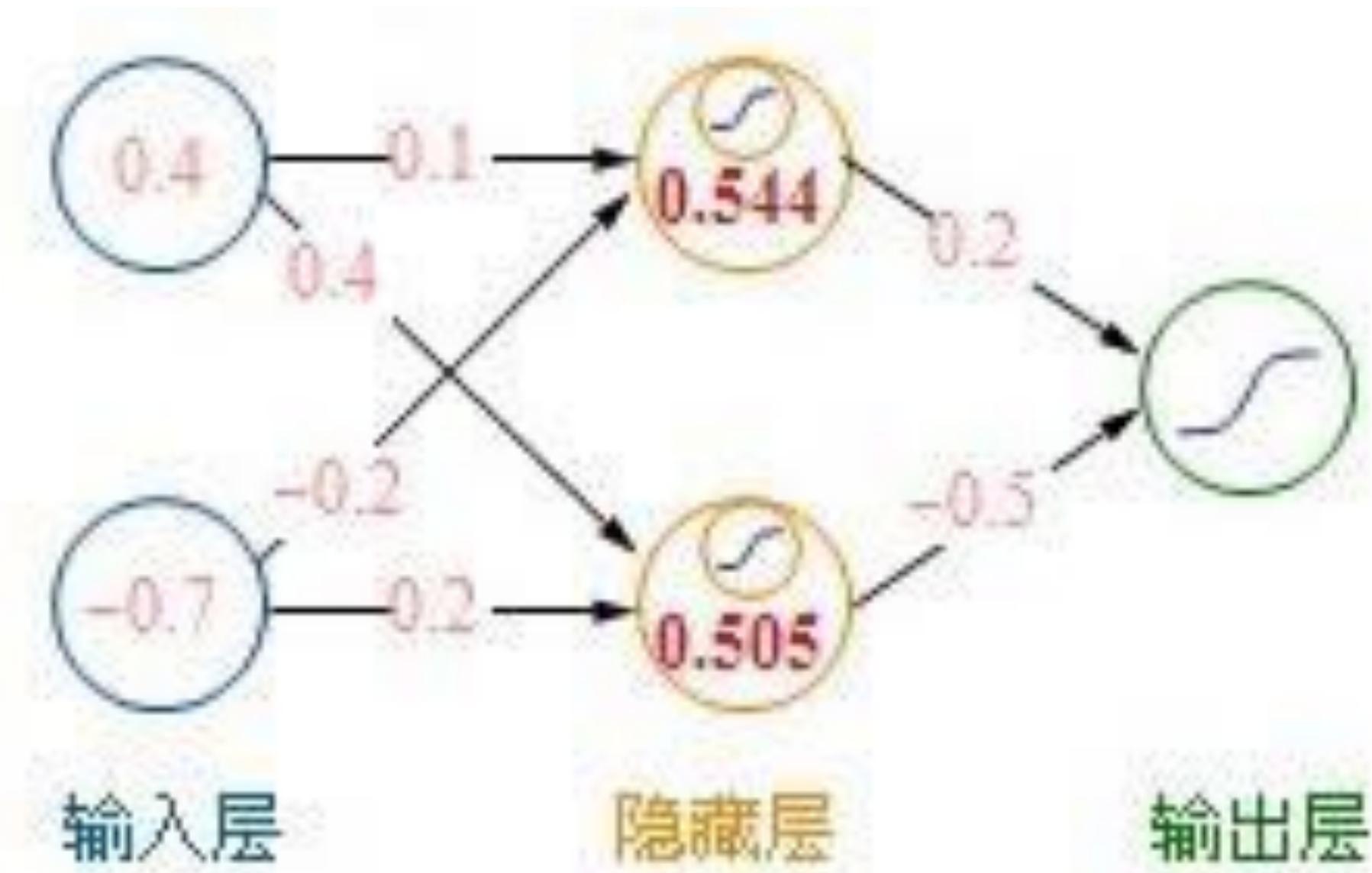
神经网络基础模型



$$0.4 \times 0.1 + (-0.7) \times (-0.2) = 0.18$$

$$0.4 \times 0.4 + (-0.7) \times (0.2) = 0.02$$

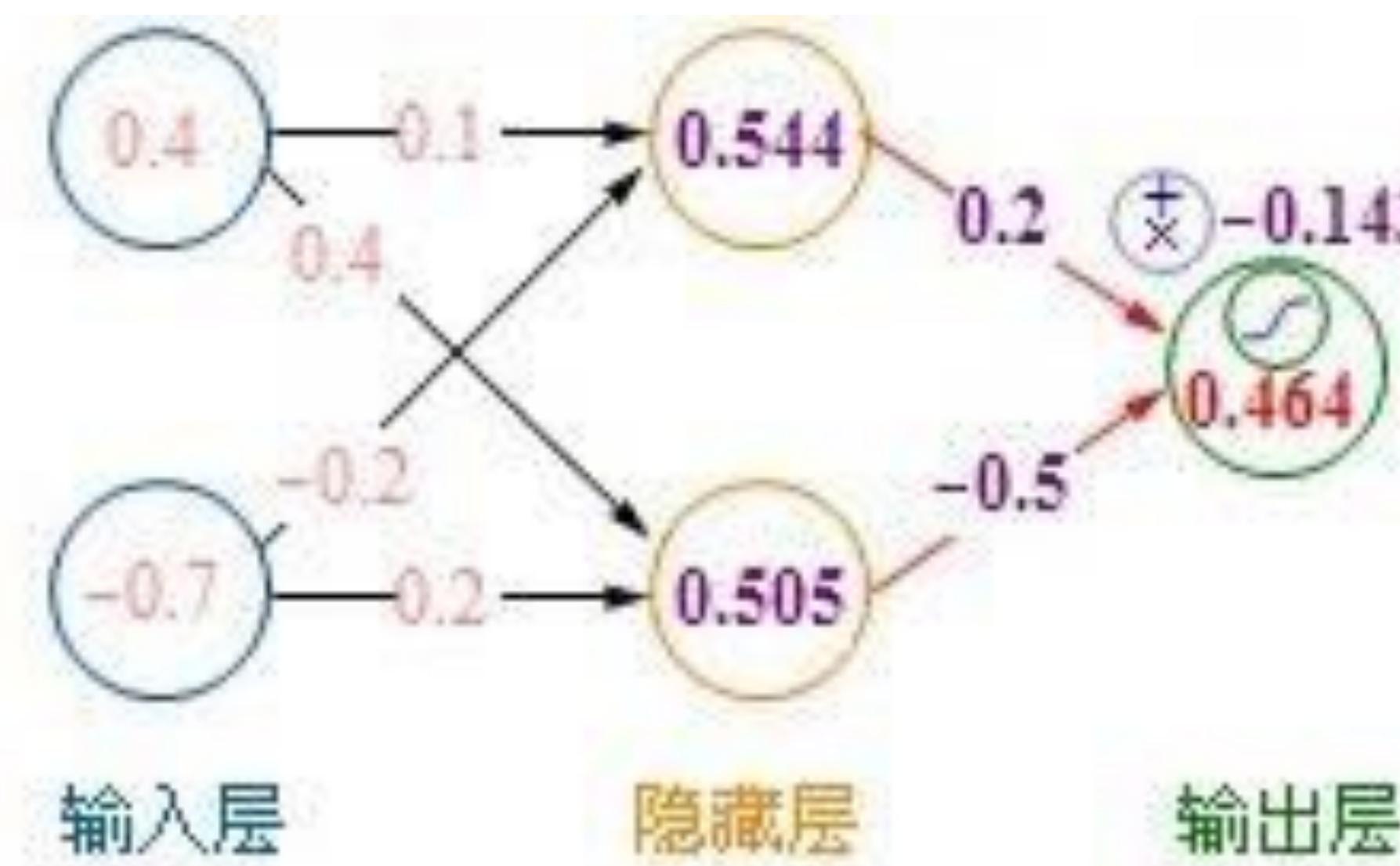
神经网络基础模型



$$\text{logsig}(0.18) = \frac{1}{1 + e^{-0.18}} = 0.544$$

$$\text{logsig}(0.02) = \frac{1}{1 + e^{-0.02}} = 0.505$$

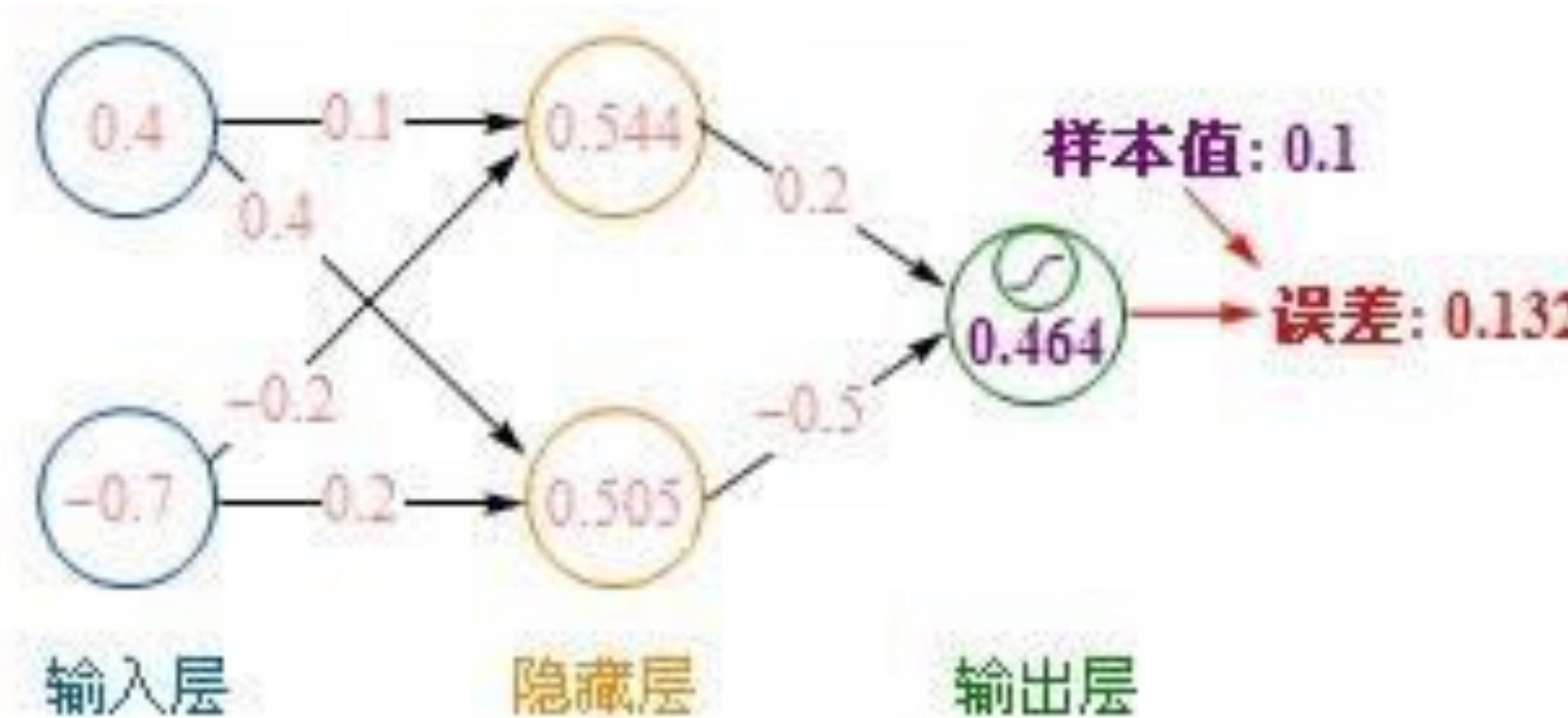
神经网络基础模型



$$0.544 \times 0.2 + 0.505 \times (-0.5) = -0.143$$

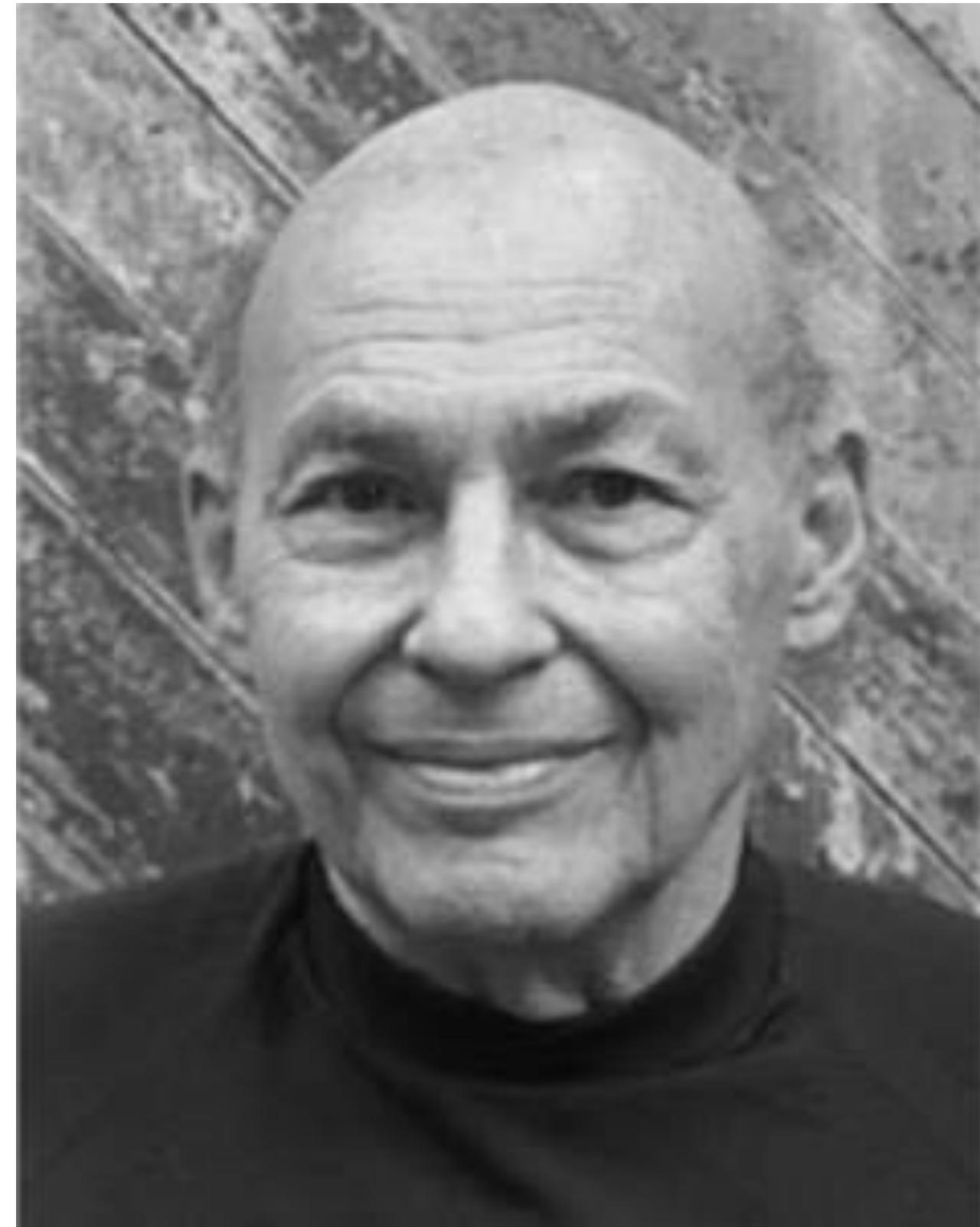
$$\text{logsig}(-0.143) = \frac{1}{1 + e^{-0.143}} = 0.464$$

神经网络基础模型

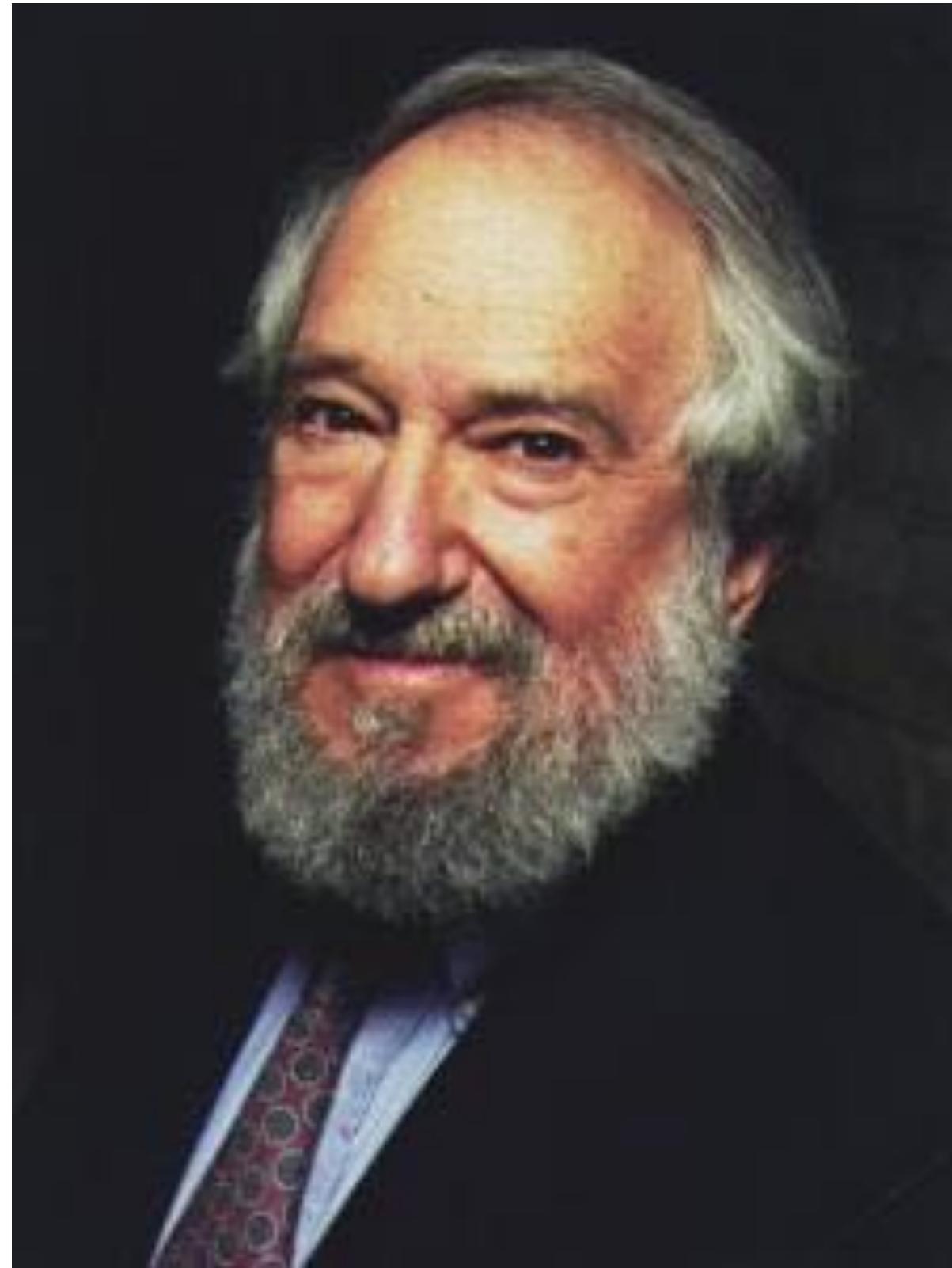


$$\text{误差} = (0.464 - 0.1)^2 = 0.132 \quad (\text{误差} < 0.0001, \text{可以收敛})$$

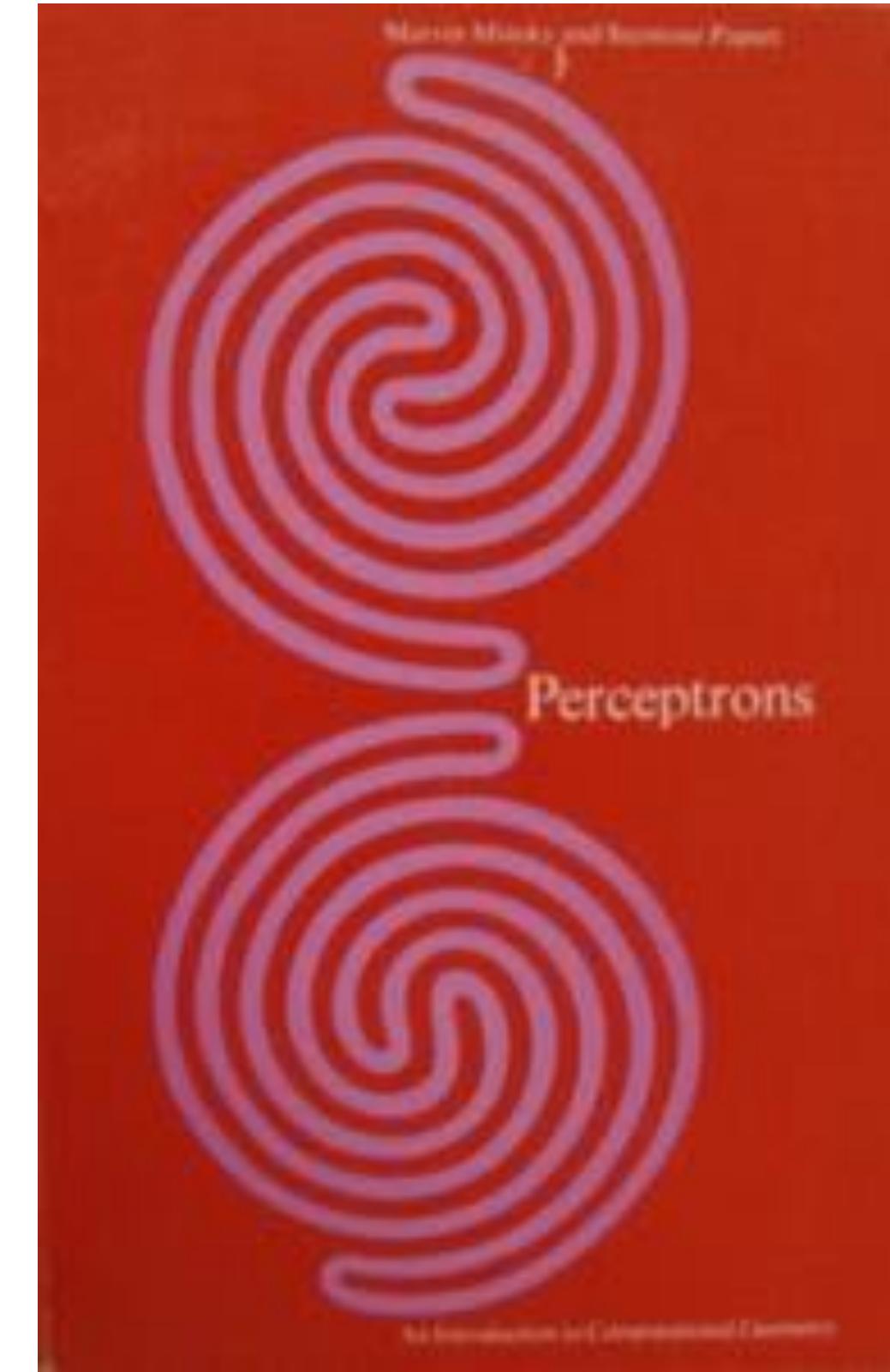
神经网络的第一次寒冬



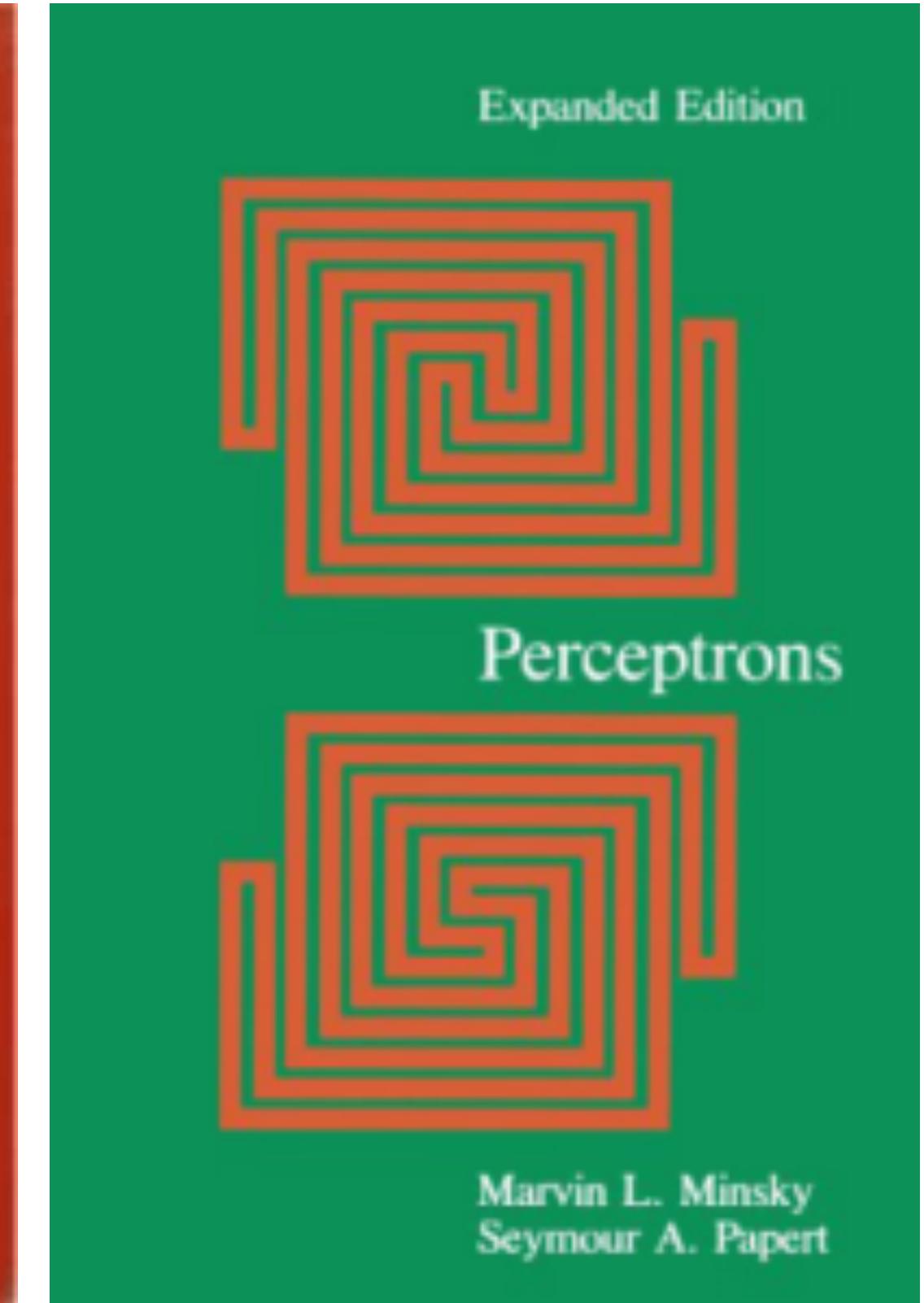
Marvin Minsky
(1927-2016)



Seymour Papert
(1928-)



1969年出版《Perceptrons》一书，认为仅靠局部连接的神经网络无法有效开展训练以及很多被后来的读者们以讹传讹的观点



反向传播算法（BP）的提出



Geoffrey Hinton
(1947-)

David Rumelhart
(1942-2011)

Learning representations by back-propagating errors

**David E. Rumelhart*, Geoffrey E. Hinton†
& Ronald J. Williams***

* Institute for Cognitive Science, C-015, University of California,
San Diego, La Jolla, California 92093, USA

† Department of Computer Science, Carnegie-Mellon University,
Pittsburgh, Philadelphia 15213, USA

1986年发表在Nature上的文章将BP算法用于神经网络模型，极大降低了计算量 ($O(n^2) \rightarrow O(n)$)

当时的计算机普及率、计算能力也远胜60年代

“首个”重要应用：手写数字识别



**Yann LeCun
(1960-)**

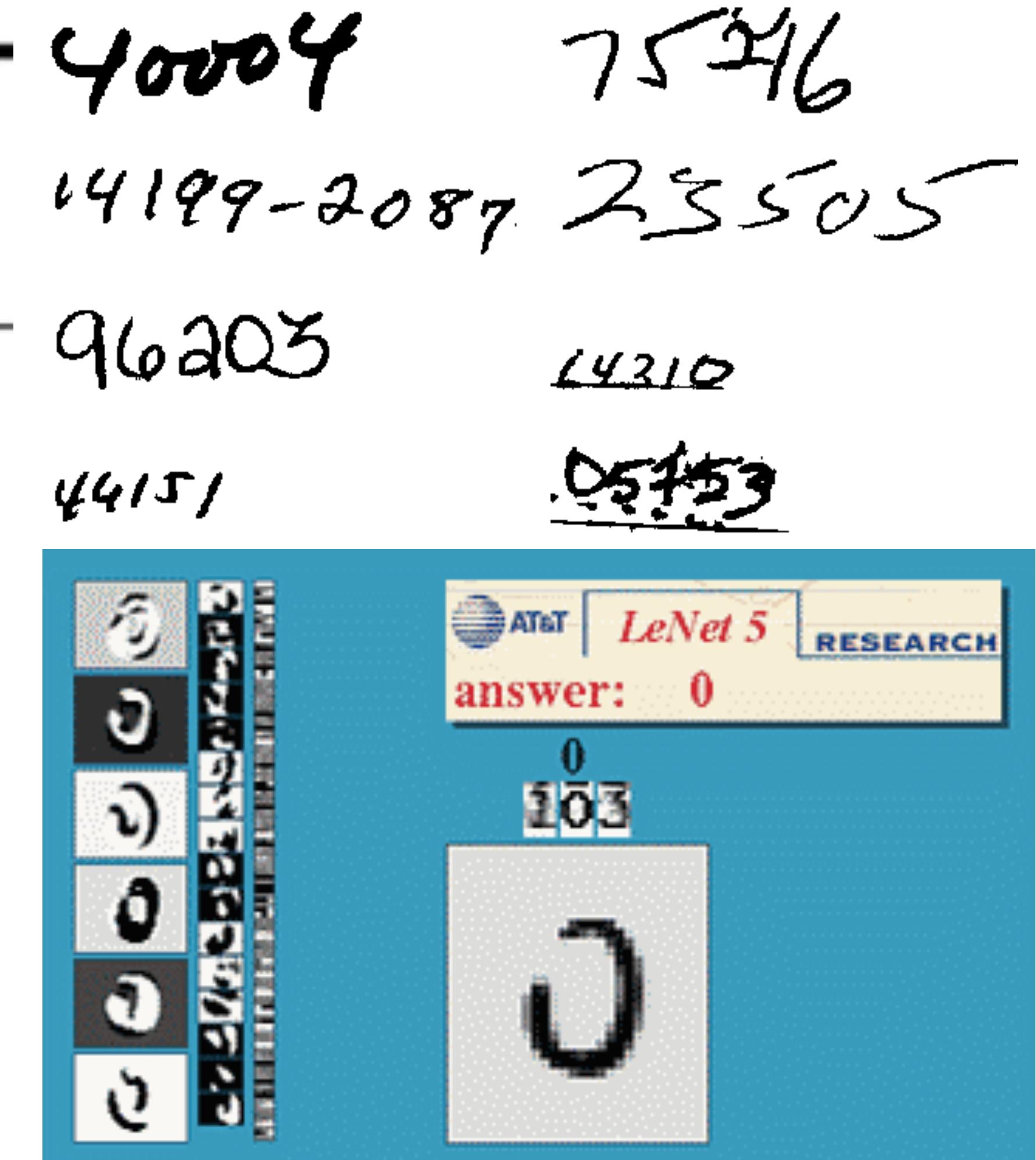
Handwritten Digit Recognition with a Back-Propagation Network

Y. Le Cun, B. Boser, J. S. Denker, D. Henderson,
R. E. Howard, W. Hubbard, and L. D. Jackel
AT&T Bell Laboratories, Holmdel, N. J. 07733

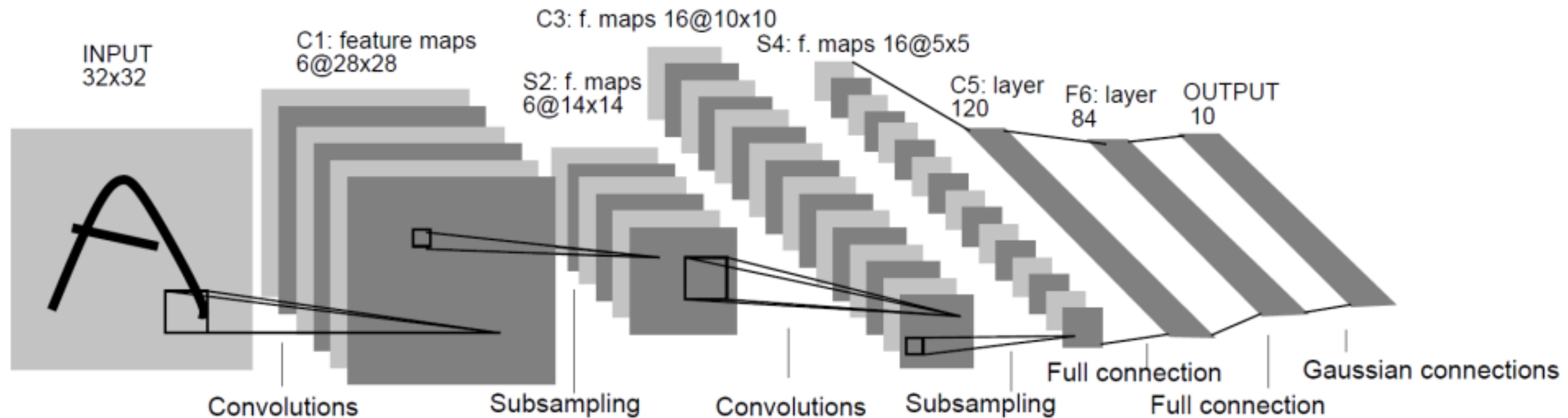
ABSTRACT

We present an application of back-propagation networks to handwritten digit recognition. Minimal preprocessing of the data was required, but architecture of the network was highly constrained and specifically designed for the task. The input of the network consists of normalized images of isolated digits. The method has 1% error rate and about a 9% reject rate on zipcode digits provided by the U.S. Postal Service.

到上世纪九十年代末，超过10%的美国支票识别采用了相关技术

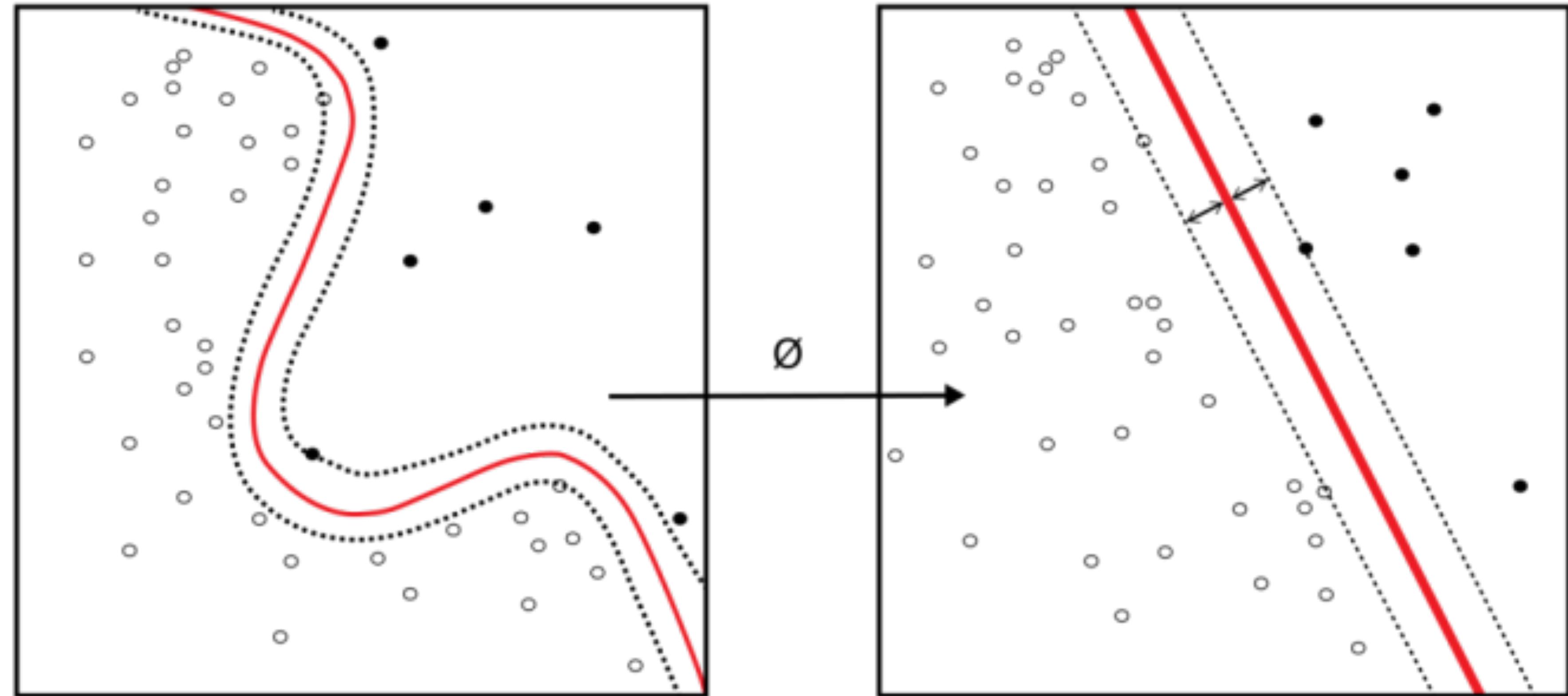


LeNet-5



1998年发表的Gradient-based learning applied to Document Recognition一文中提出了LeNet-5
已是Convolutional Neural Network的基本框架

神经网络的第二次寒冬



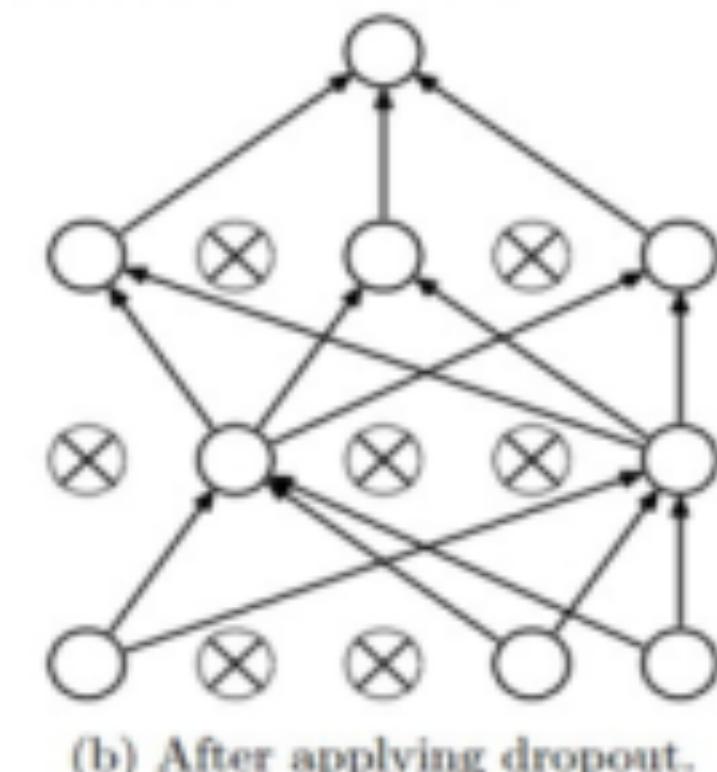
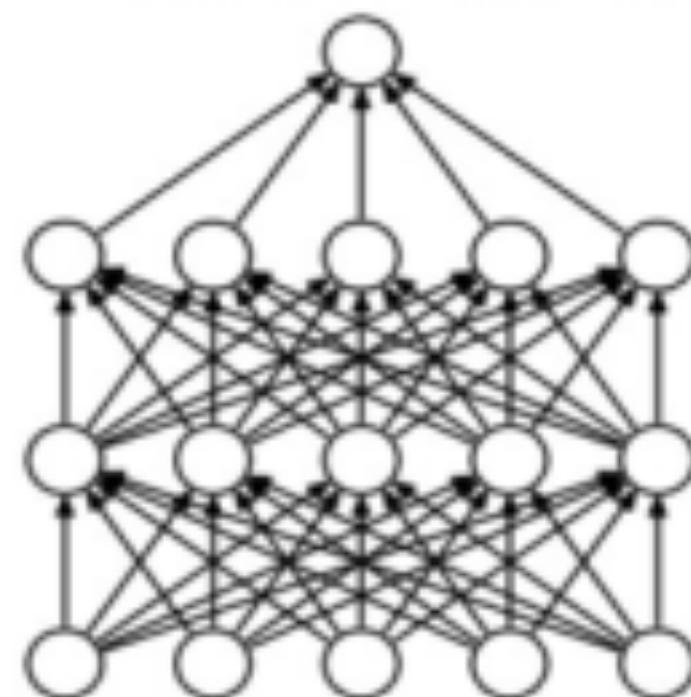
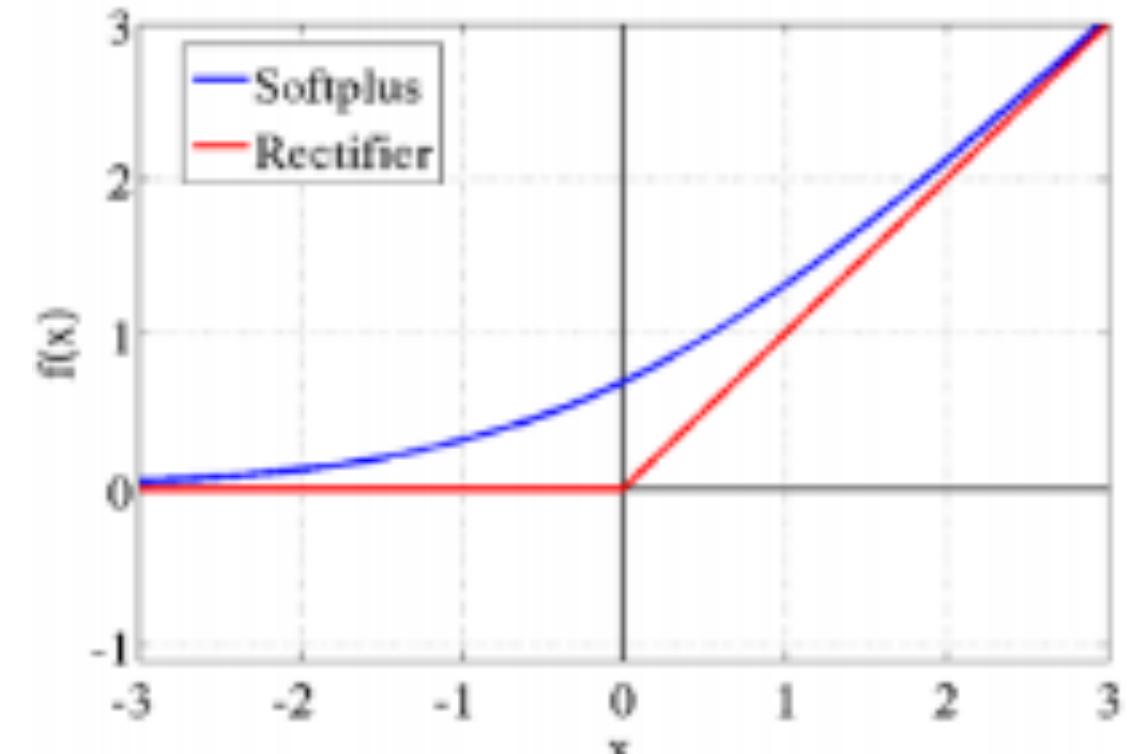
**Vladimir Vapnik
(1936-)**

在1963年发表的论文中即提出Support Vector Machine的概念
在2002年左右，不断改进的SVM方法将手写数字识别的错误率降到了0.56%
不需要大量样本（事实上也难以支撑大量样本），速度相对可以接受

卷土重来的（深度）神经网络

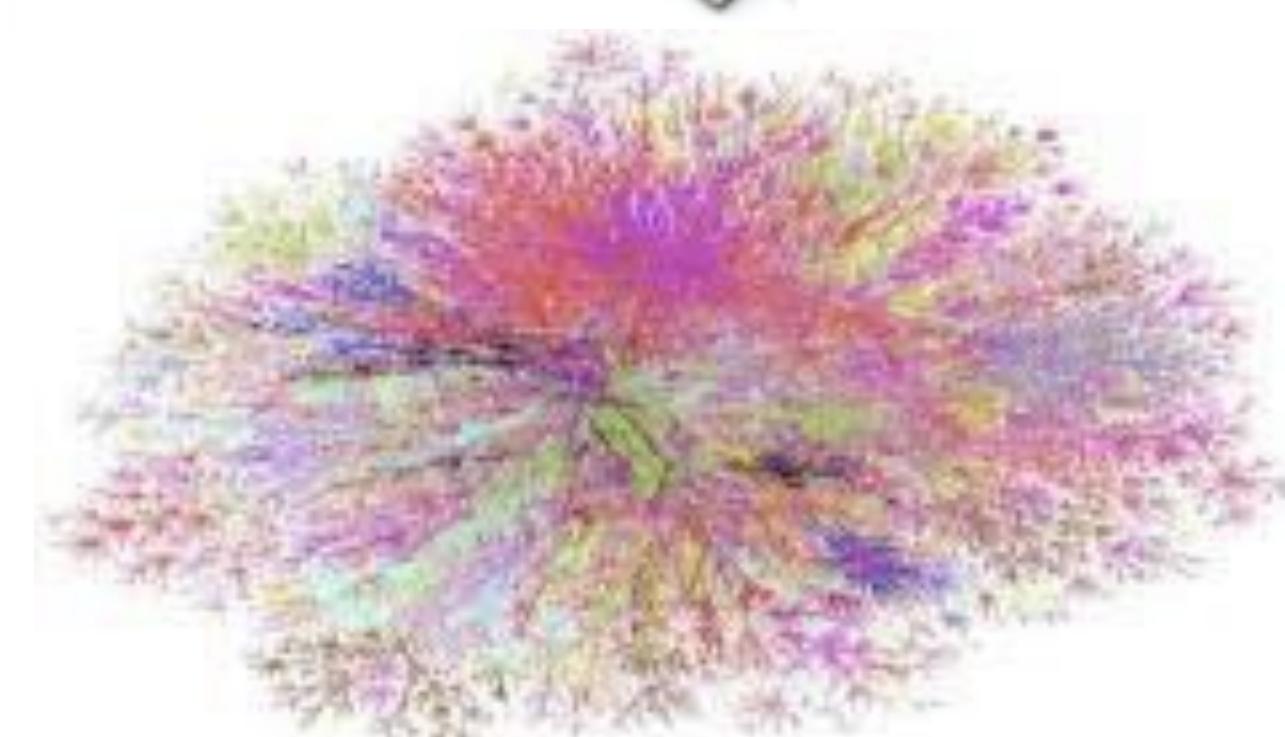
优化策略

梯度弥散问题通过ReLU、Dropout、Deep Residual Learning等方法得到缓解



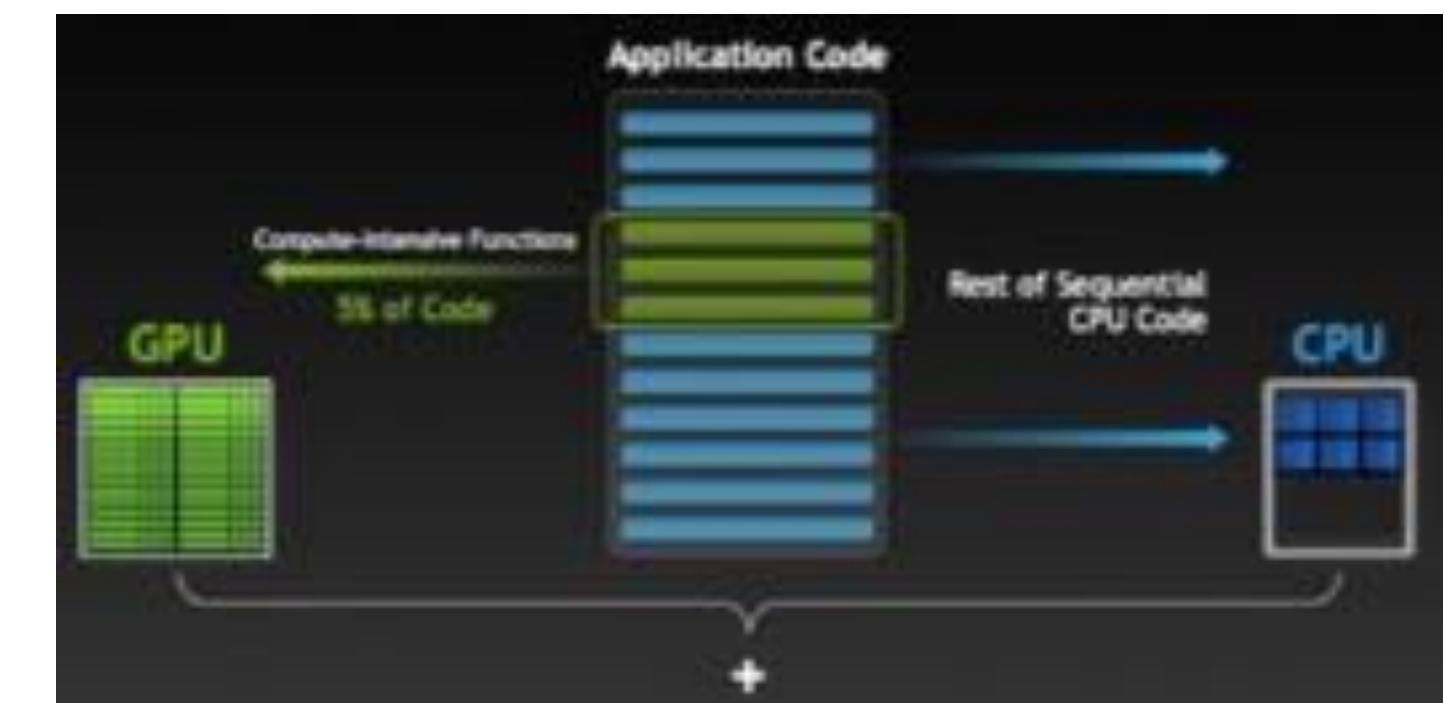
数据规模

真正意义上的大数据出现
社会标注机制



计算能力

CPU、GPU的长足进步
共享权值



AlexNet

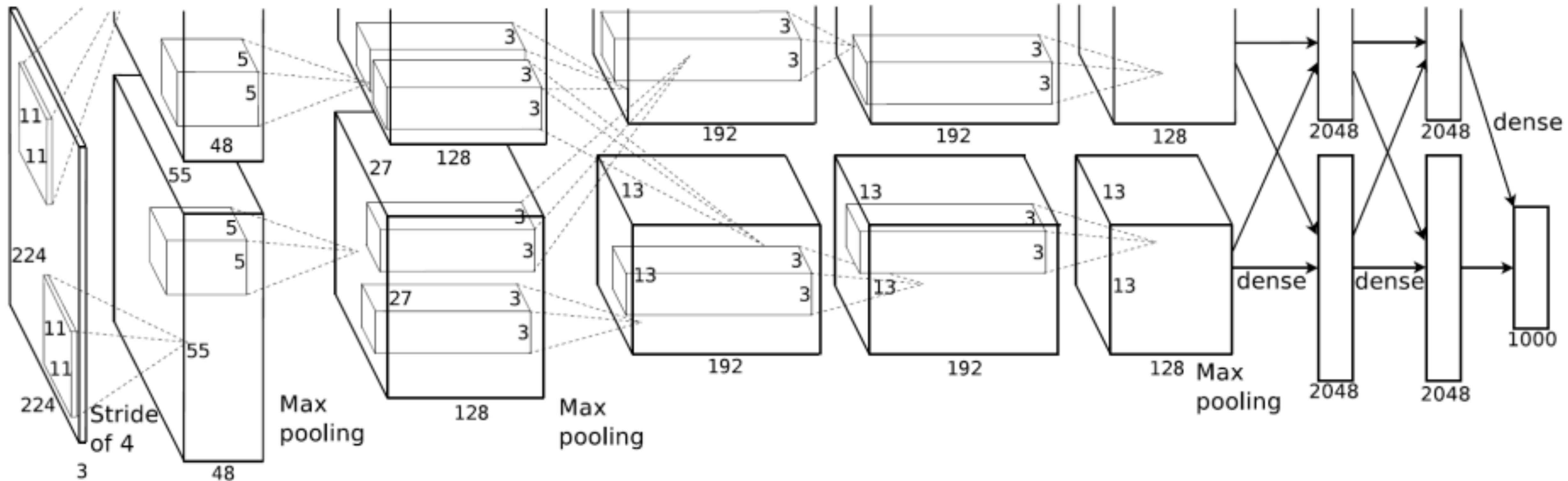
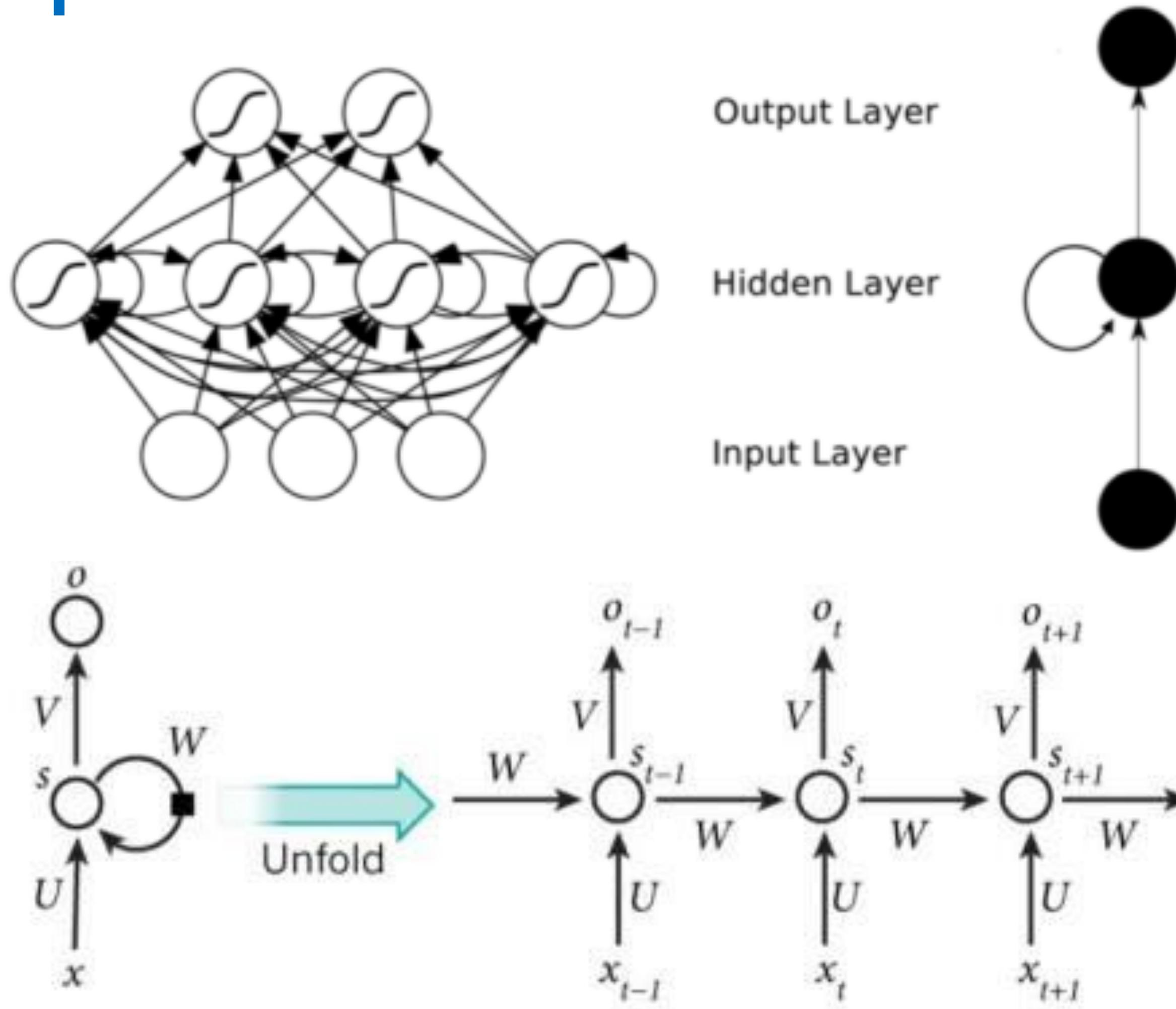


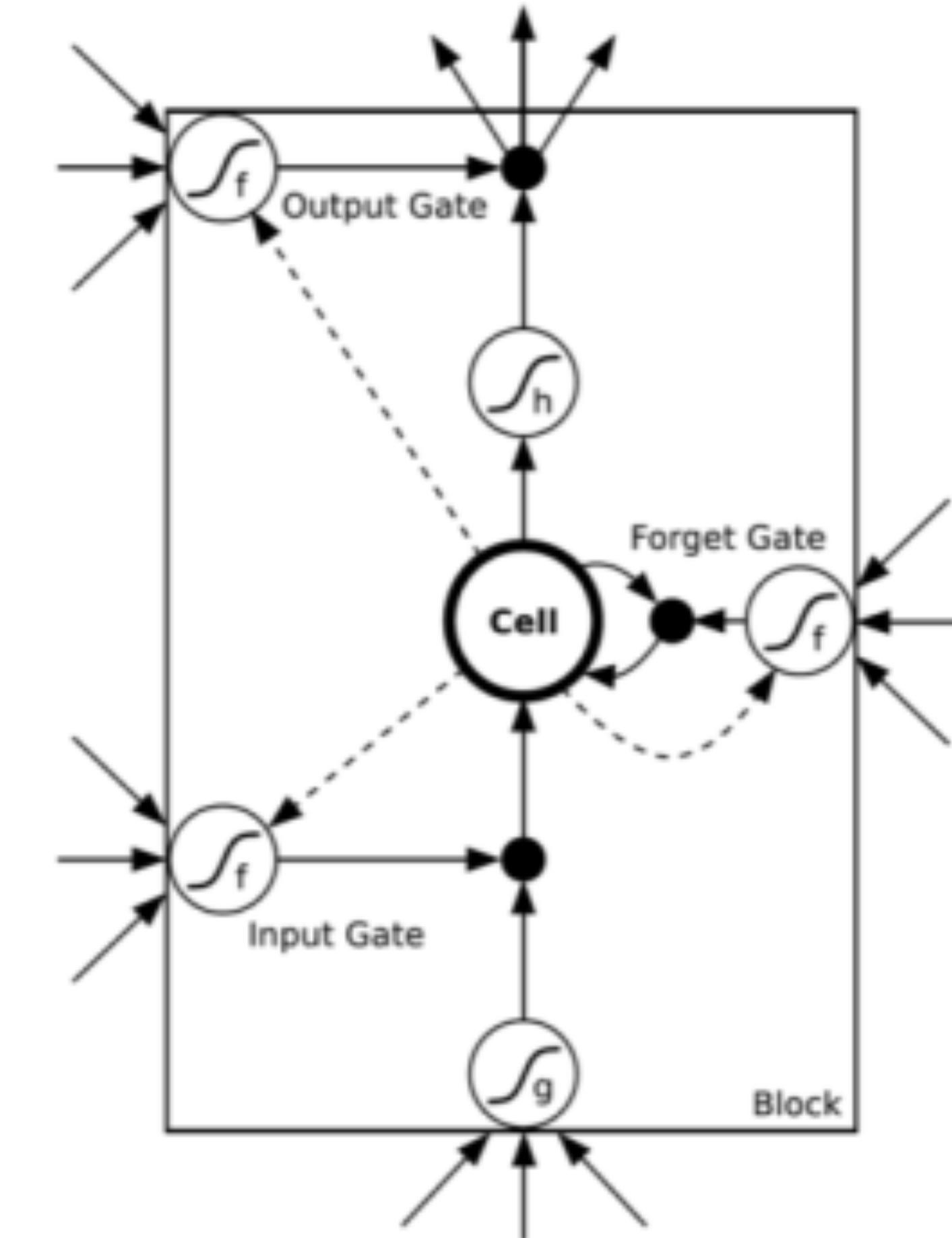
Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

可处理时序数据的深度学习机制

| Recursive Neural Networks



| Long Short Term Memory



大型企业不同于以往的密切关注



开源框架

名称	基本语言	是否支持多GPU	速度	应用领域
TensorFlow	 Python、C++	是	★★★☆	通用
Caffe	C++	是	★★★★☆	图像分类
Torch	 Lua	是	★★☆☆	通用
Theano	 Python	默认为否	★★★★☆	通用
IBM DL Platform	 SystemML等	是	?	通用

Details of the History of neural network (1)

- Pioneering work on the mathematical model of neural networks
 - McCulloch and Pitts 1943;
 - Include recurrent and non-recurrent (with “circles”) networks;
 - Use thresholding function as nonlinear activation; No learning
- Early works on learning neural networks
 - Starting from Rosenblatt 1958;
 - Using thresholding function as nonlinear activation prevented computing derivatives with the chain rule, and so errors could not be propagated back to guide the computation of gradients
- Backpropagation was developed in several steps since 1960
 - The key idea is to use the chain rule to calculate derivatives;
 - It was reflected in multiple works, earliest from the field of control
- Standard backpropagation for neural networks
 - Rumelhart, Hinton, and Williams, Nature 1986: Clearly appreciated the power of backpropagation and demonstrated it on key tasks, and applied it to pattern recognition generally
 - In 1985, Yann LeCun independently developed a learning algorithm for three-layer networks in which target values were propagated, rather than derivatives. In 1986, he proved that it was equivalent to standard backpropagation
- Prove the universal expressive power of three-layer neural networks
 - Hecht-Nielsen 1989



- Convolutional neural network
 - Introduced by Kunihiko Fukushima in 1980
 - Improved by LeCun, Bottou, Bengio, and Haffner in 1998
- Deep belief net (DBN)
 - Hinton, Osindero, and Teh 2006
- Auto encoder
 - Hinton and Salakhutdinov 2006 (Science)
- Deep learning
 - Hinton. Learning multiple layers of representations. Trends in Cognitive Sciences, 2007.
 - Unsupervised multilayer pre-training + supervised fine-tuning (BP)
- Large-scale deep learning in speech recognition
 - Geoff Hinton and Li Deng started this research at Microsoft Research Redmond in late 2009.
 - Generative DBN pre-training was not necessary
 - Success was achieved by large-scale training data + large deep neural network (DNN) with large, context-dependent output layers
- Unsupervised deep learning from large scale images
 - Andrew Ng et al. 2011
 - Unsupervised feature learning
 - 16000 CPUs
- Large-scale supervised deep learning in ImageNet image classification
 - Krizhevsky, Sutskever, and Hinton 2012
 - Supervised learning with convolutional neural network
 - No unsupervised pre-training



Neural Network in a Nut Shell



Representations Matter

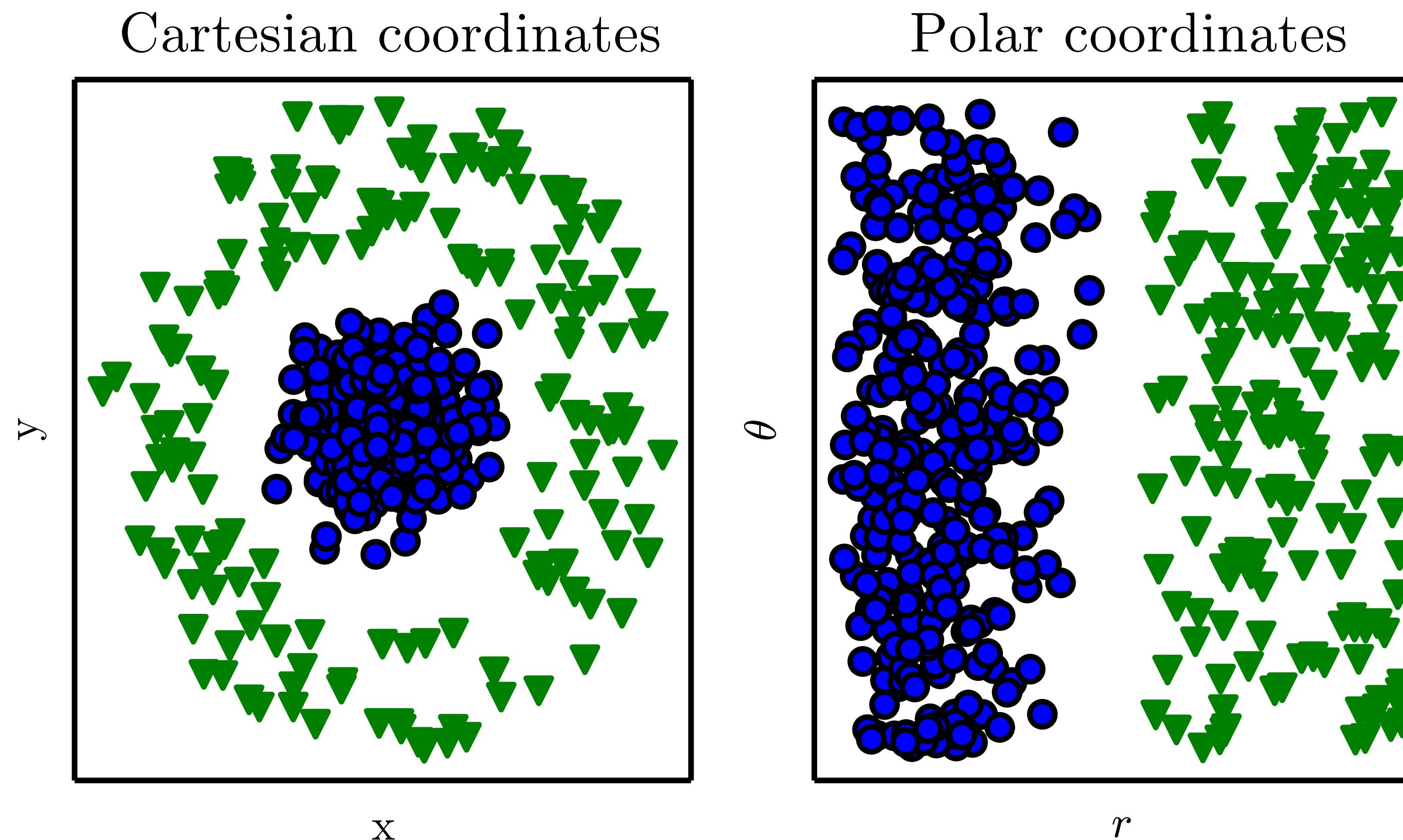


Figure 1.1

(Goodfellow 2016)

Depth: Repeated Composition

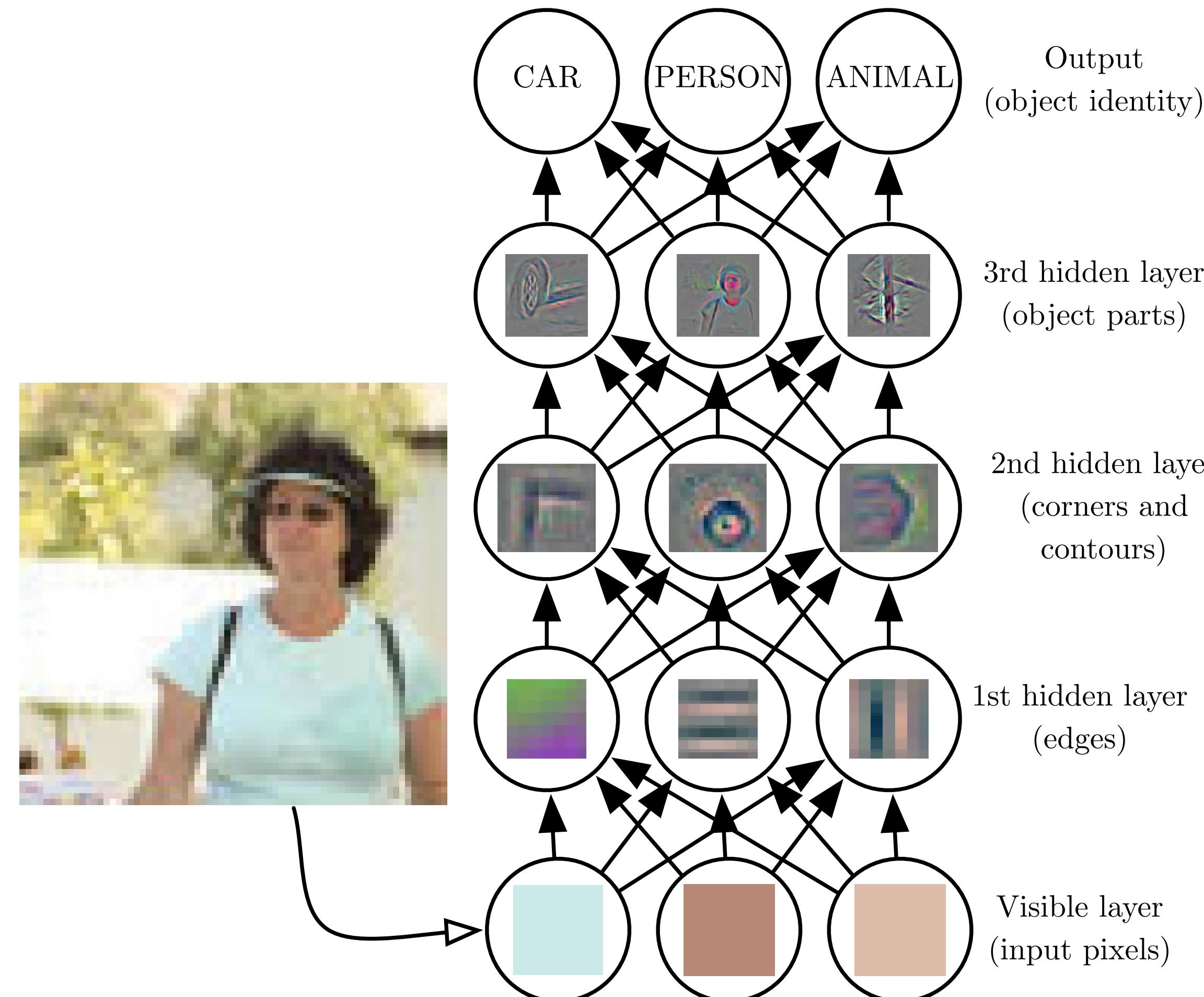


Figure 1.2

(Goodfellow 2016)

Computational Graphs

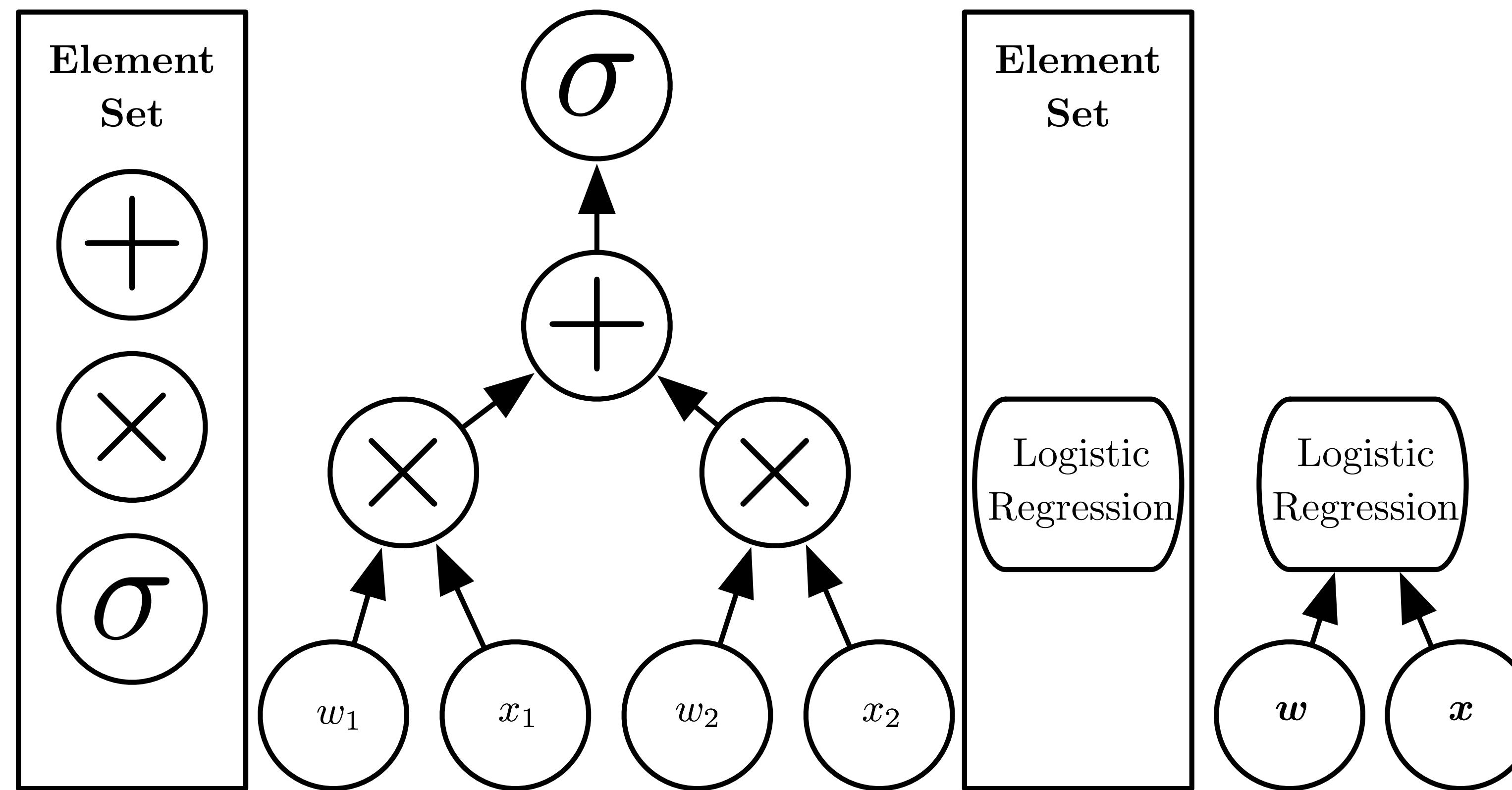


Figure 1.3

(Goodfellow 2016)

Machine Learning and AI

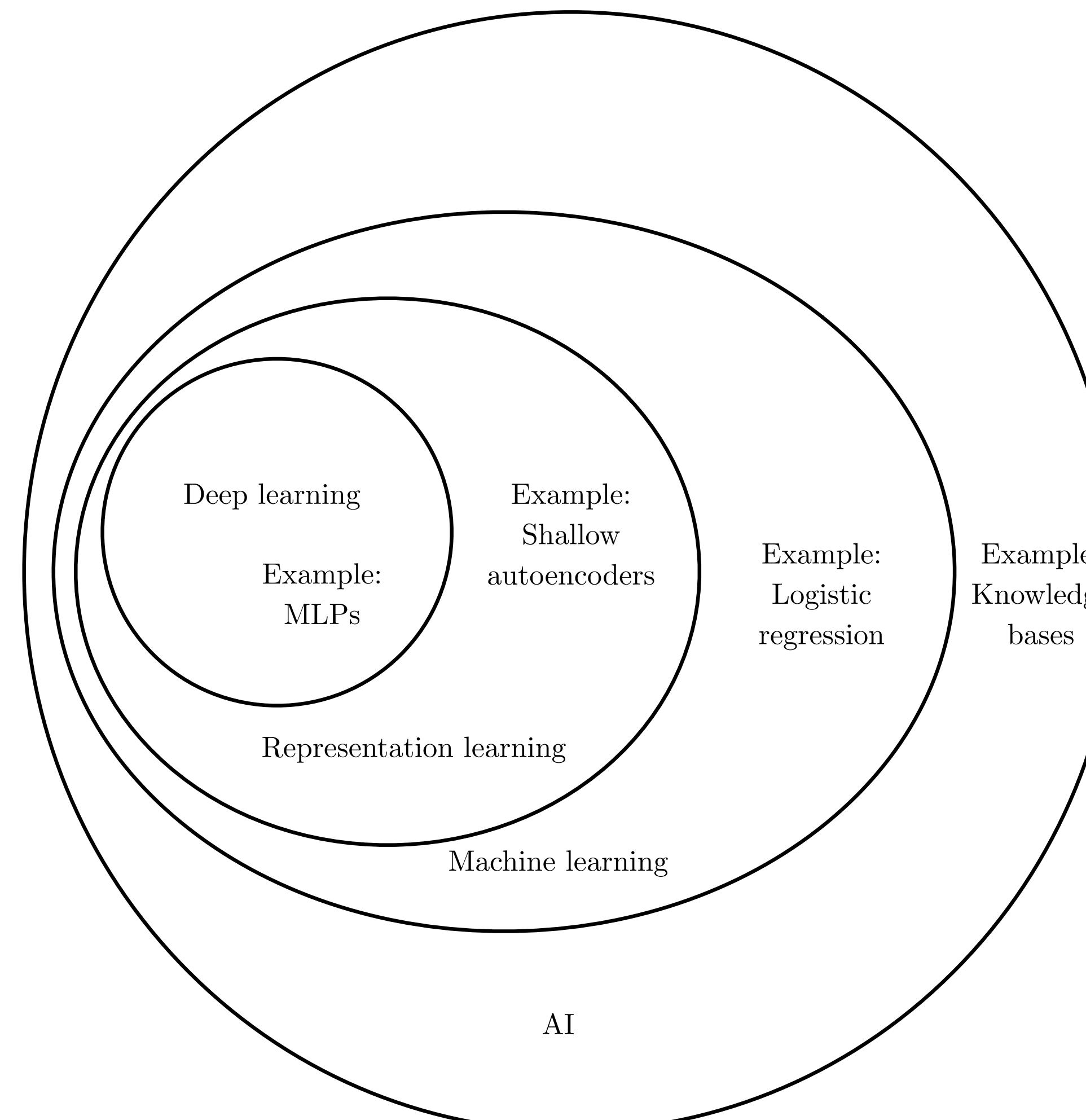
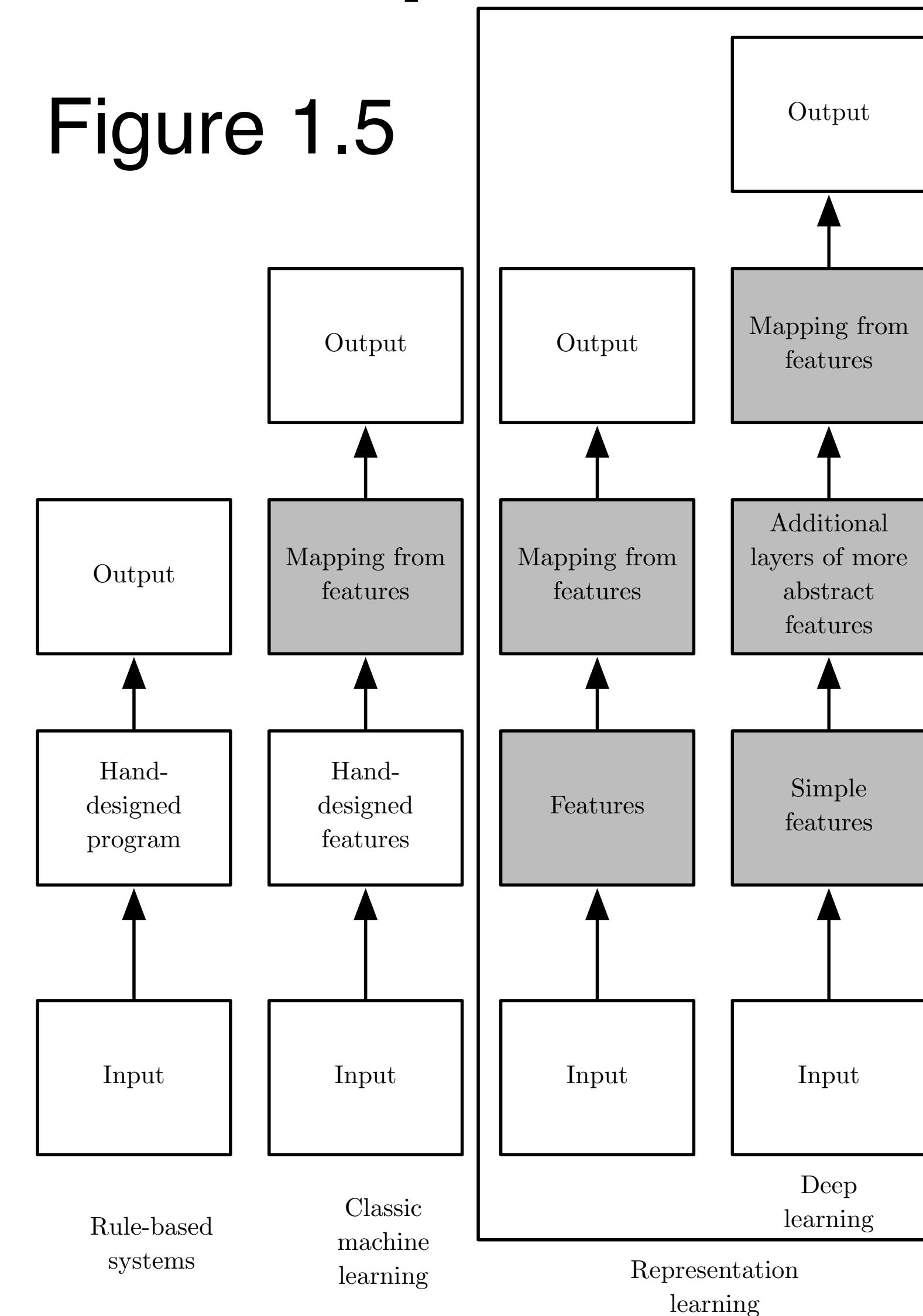


Figure 1.4

(Goodfellow 2016)

Learning Multiple Components

Figure 1.5



Historical Waves

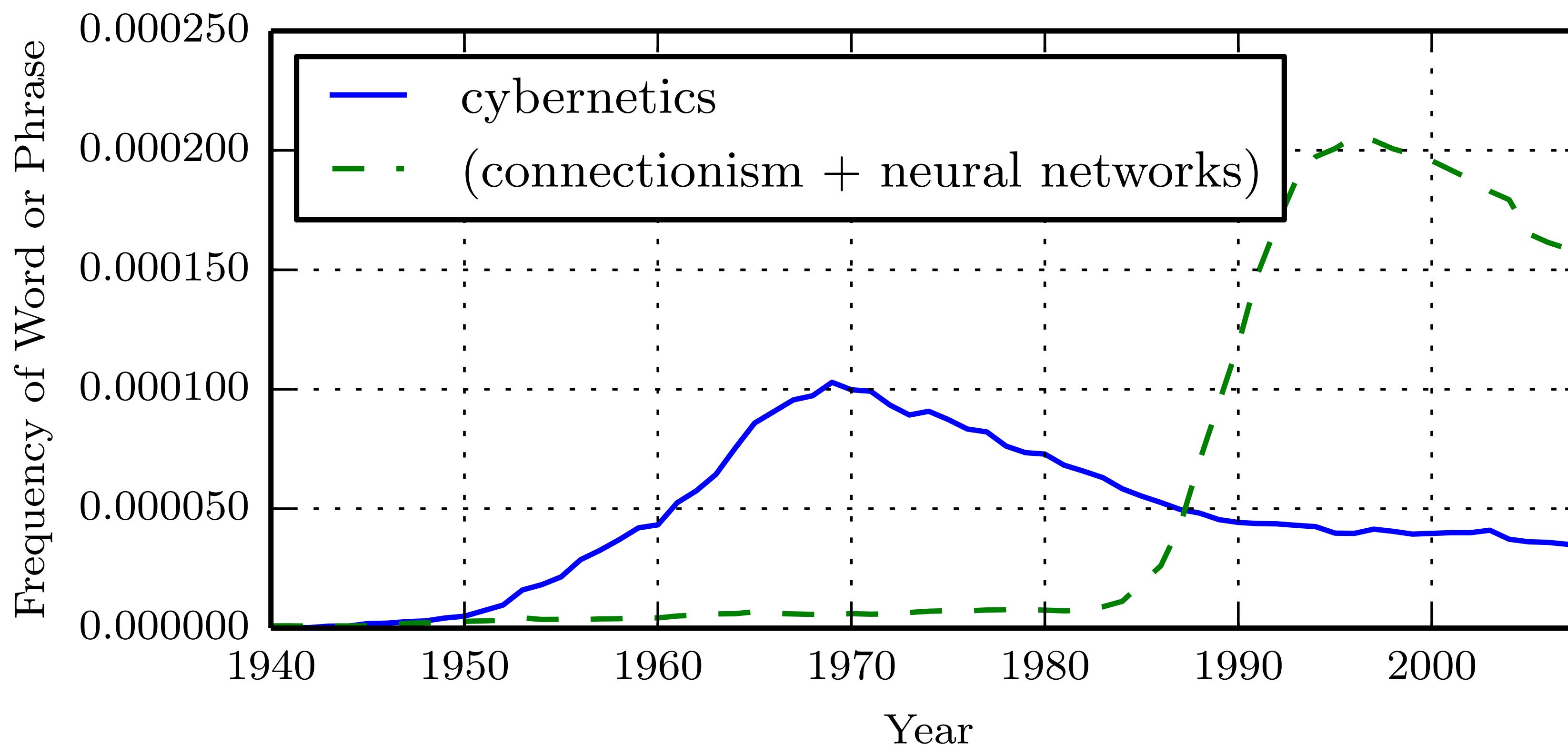


Figure 1.7

(Goodfellow 2016)

Historical Trends: Growing Datasets

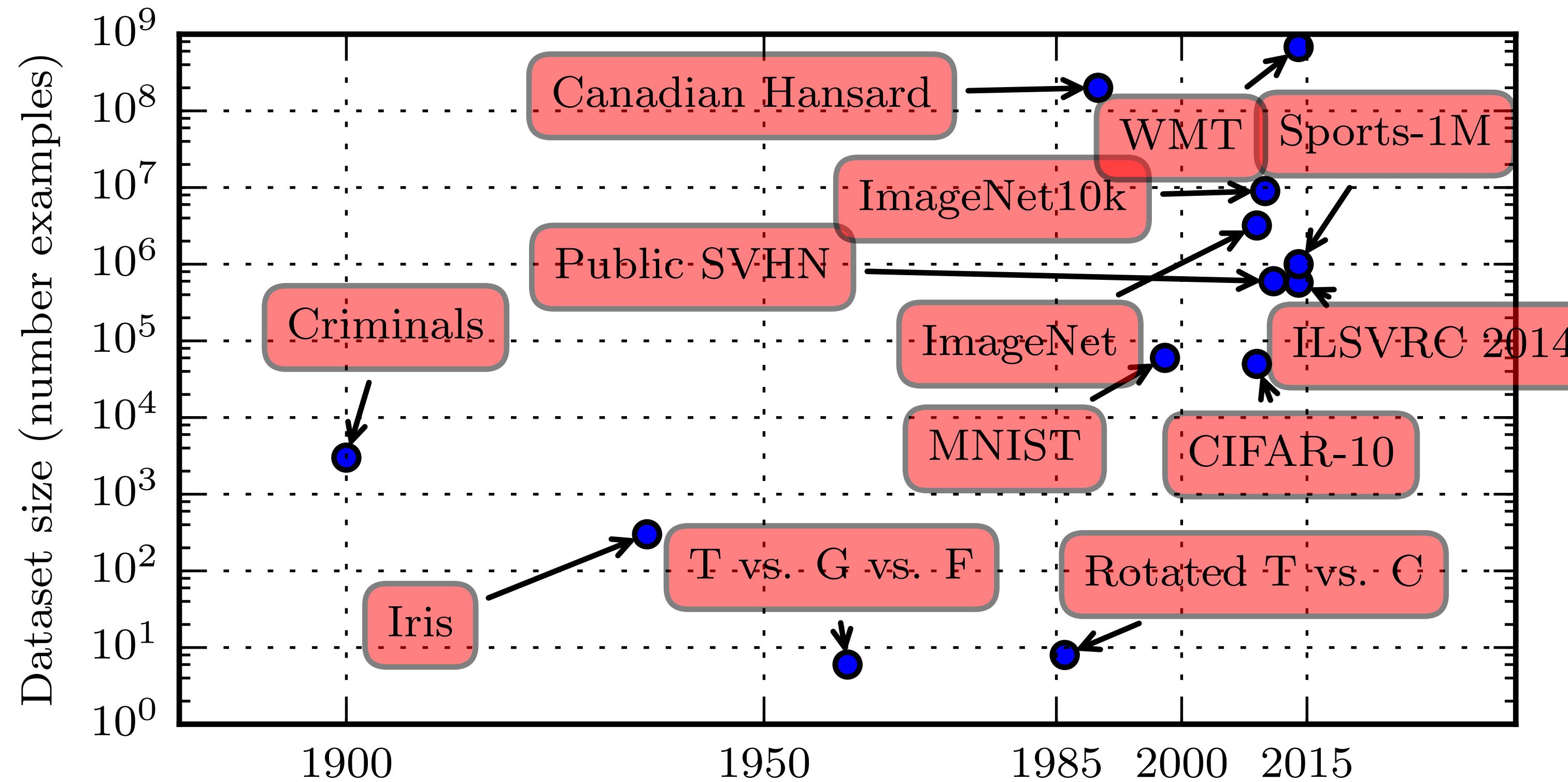


Figure 1.8

(Goodfellow 2016)

The MNIST Dataset

8	9	0	1	2	3	4	7	8	9	0	1	2	3	4	5	6	7	8	6
4	2	6	4	7	5	5	4	7	8	9	2	9	3	9	3	8	2	0	5
0	1	0	4	2	6	5	3	5	3	8	0	0	3	4	1	5	3	0	8
3	0	6	2	7	1	1	8	1	7	1	3	8	9	7	6	7	4	1	6
7	5	1	7	1	9	8	0	6	9	4	9	9	3	7	1	9	2	2	5
3	7	8	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	0
1	2	3	4	5	6	7	8	9	8	1	0	5	5	1	9	0	4	1	9
3	8	4	7	7	8	5	0	6	5	5	3	3	3	9	8	1	4	0	6
1	0	0	6	2	1	1	3	2	8	8	7	8	4	6	0	2	0	3	6
8	7	1	5	9	9	3	2	4	9	4	6	5	3	2	8	5	9	4	1
6	5	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7
8	9	0	1	2	3	4	5	6	7	8	9	6	4	2	6	4	7	5	5
4	7	8	9	2	9	3	9	3	8	2	0	9	8	0	5	6	0	1	0
4	2	6	5	5	5	4	3	4	1	5	3	0	8	3	0	6	2	7	1
1	8	1	7	1	3	8	5	4	2	0	9	7	6	7	4	1	6	8	4
7	5	1	2	6	7	1	9	8	0	6	9	4	9	9	6	2	3	7	1
9	2	2	5	3	7	8	0	1	2	3	4	5	6	7	8	0	1	2	3
4	5	6	7	8	0	1	2	3	4	5	6	7	8	9	2	1	2	1	3
9	9	8	5	3	7	0	7	7	5	7	9	9	4	7	0	3	4	1	4
4	7	5	8	1	4	8	4	1	8	6	6	4	6	3	5	7	2	5	9

Figure 1.9

(Goodfellow 2016)

Connections per Neuron

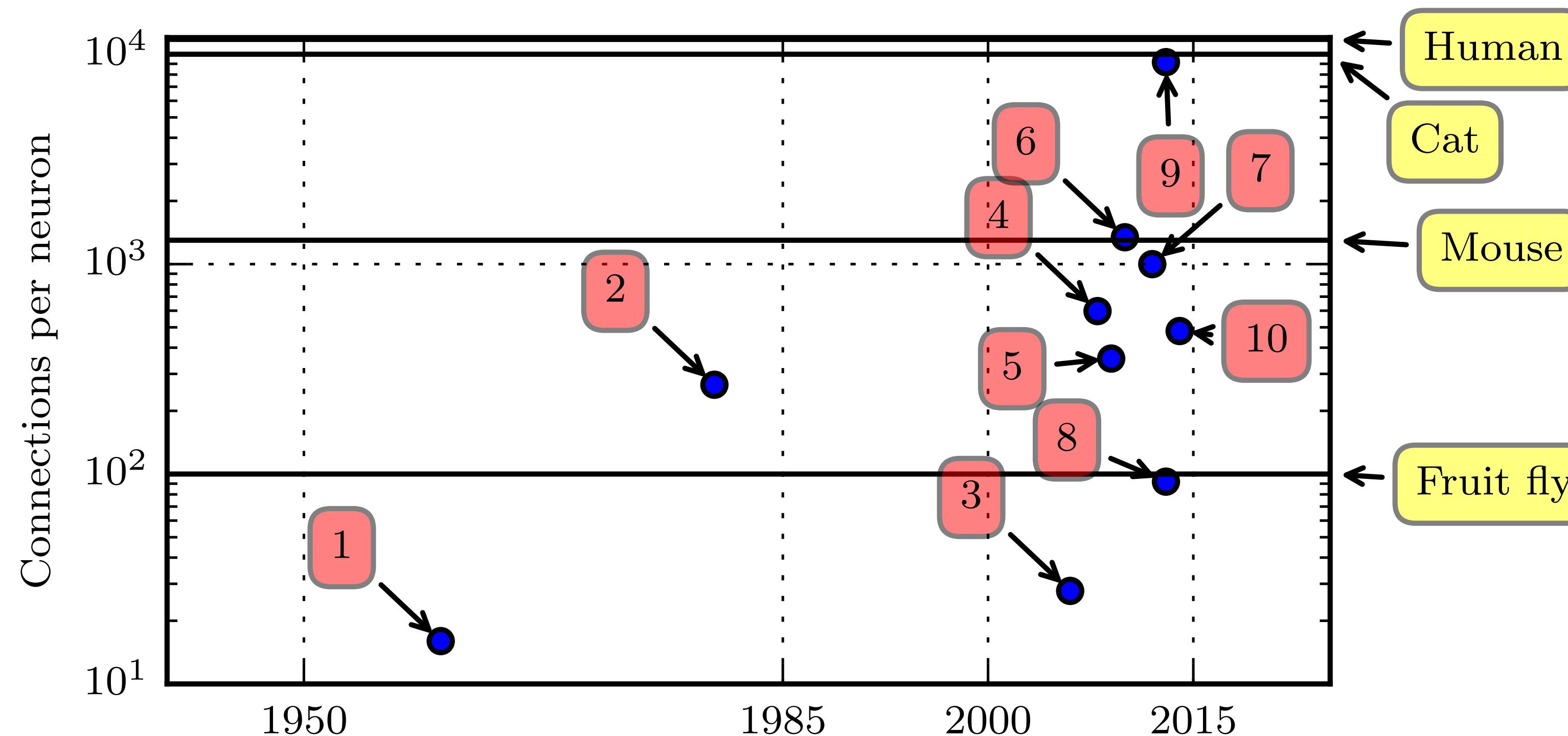


Figure 1.10

(Goodfellow 2016)

Number of Neurons

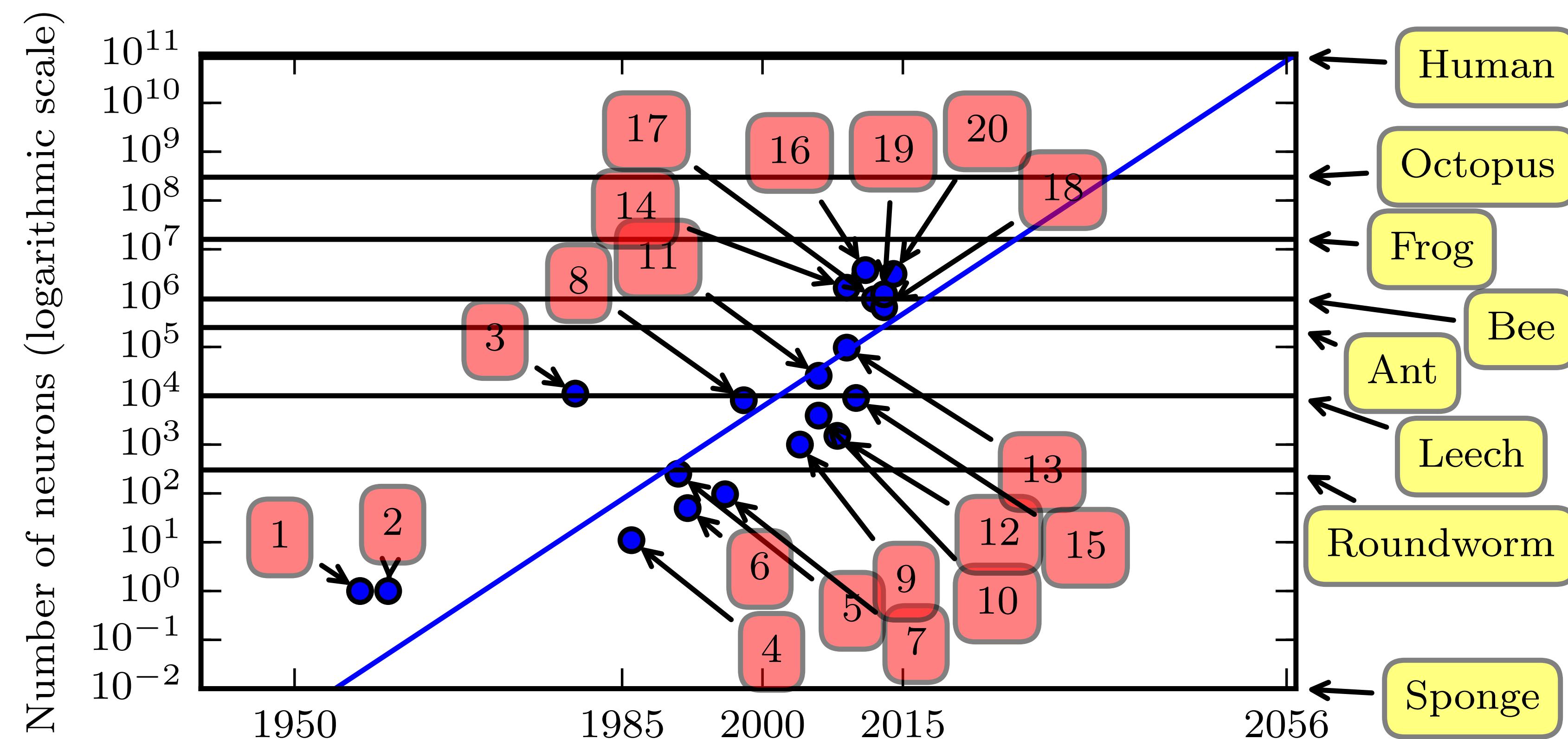


Figure 1.11

(Goodfellow 2016)

Solving Object Recognition

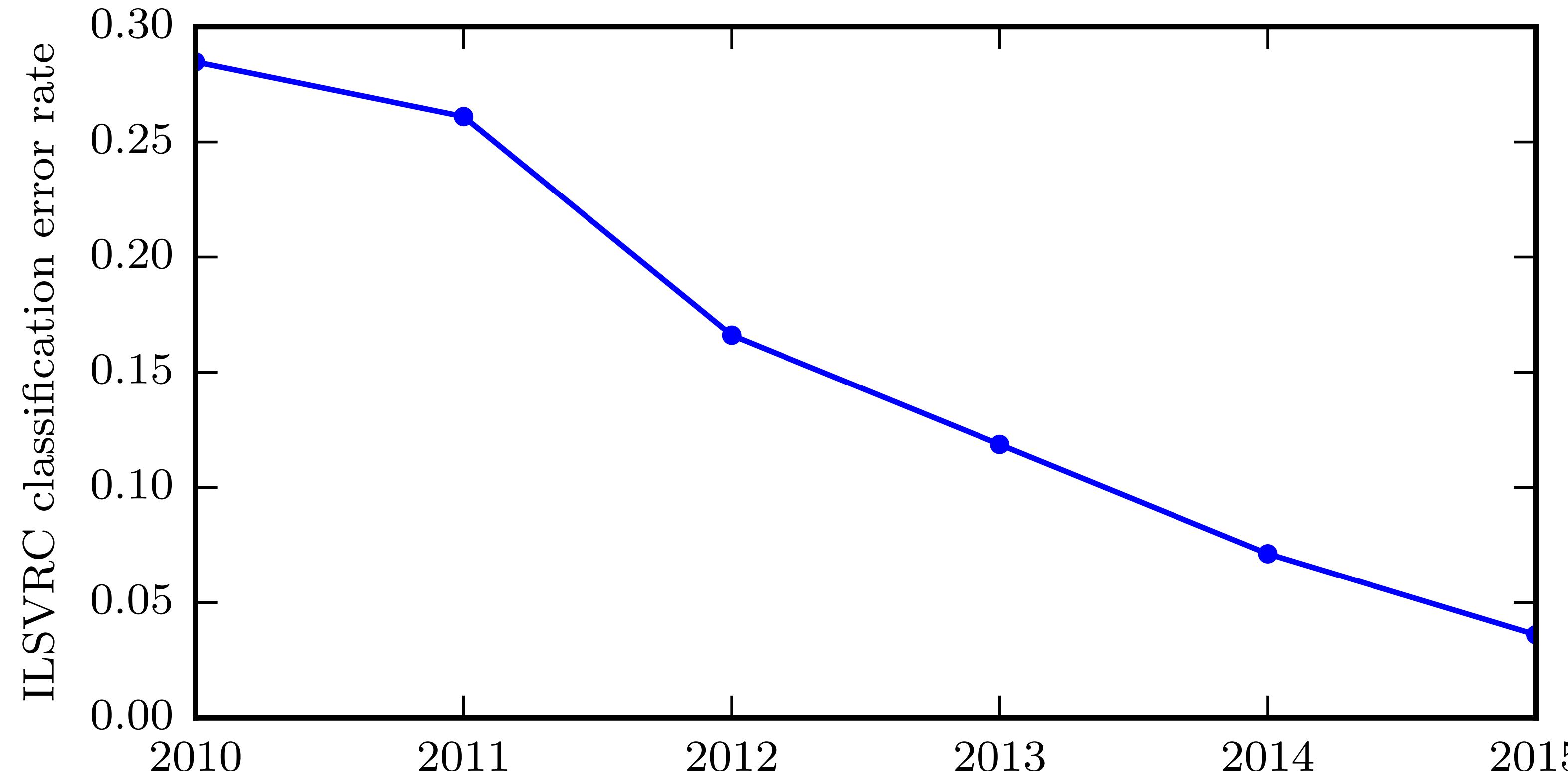


Figure 1.12

(Goodfellow 2016)

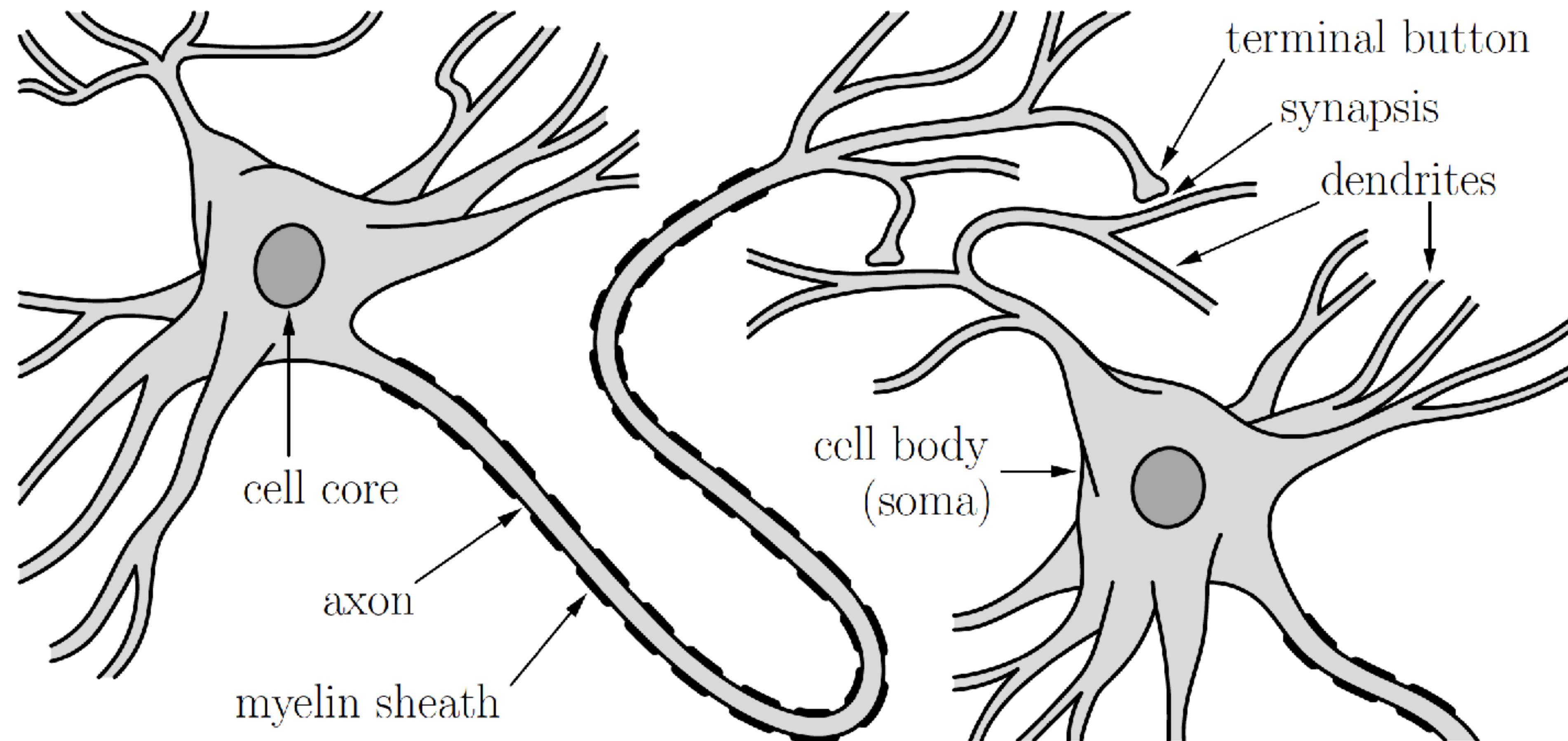
Motivation



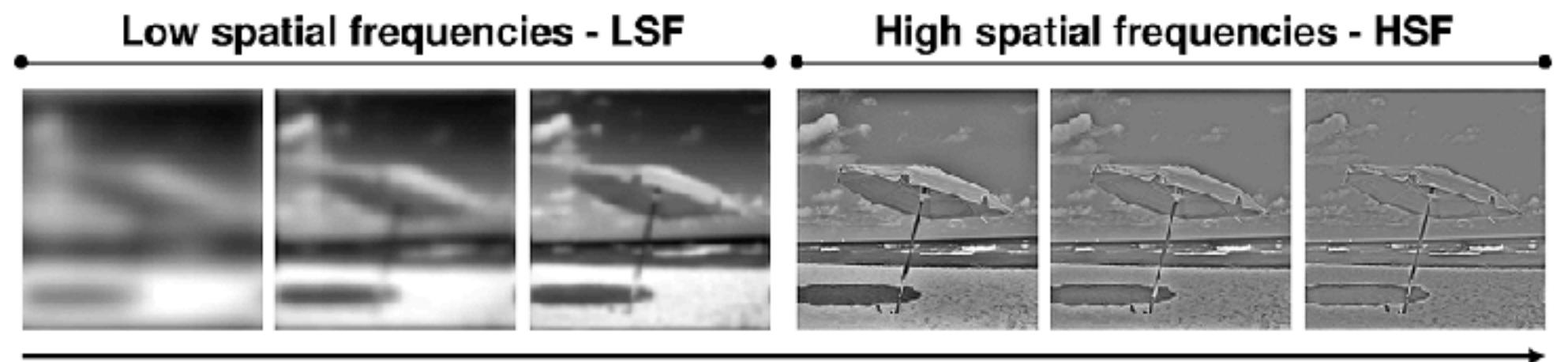
Biological Background

Optional subtitle

Structure of a prototypical biological neuron

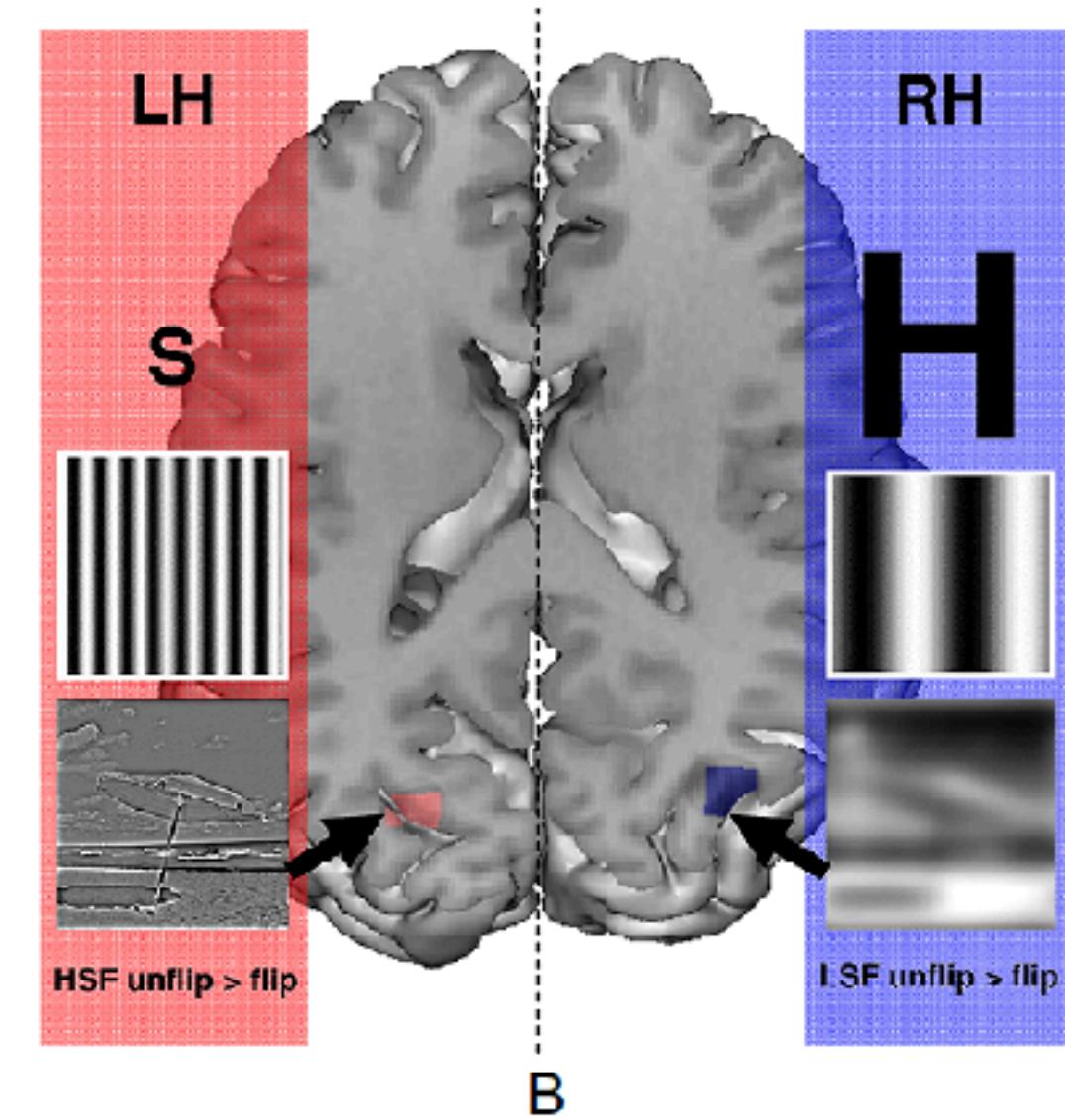
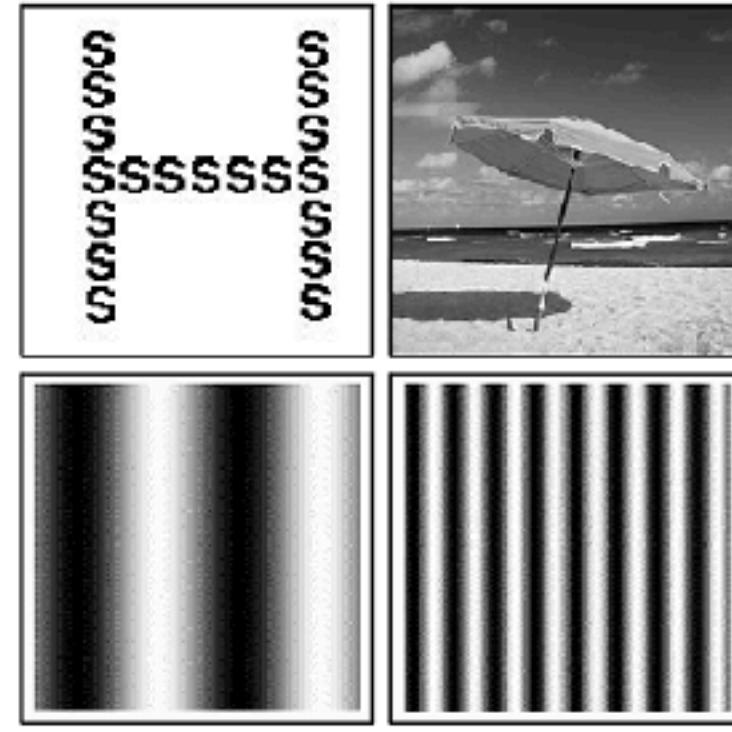


Optic



Coarse-to-Fine

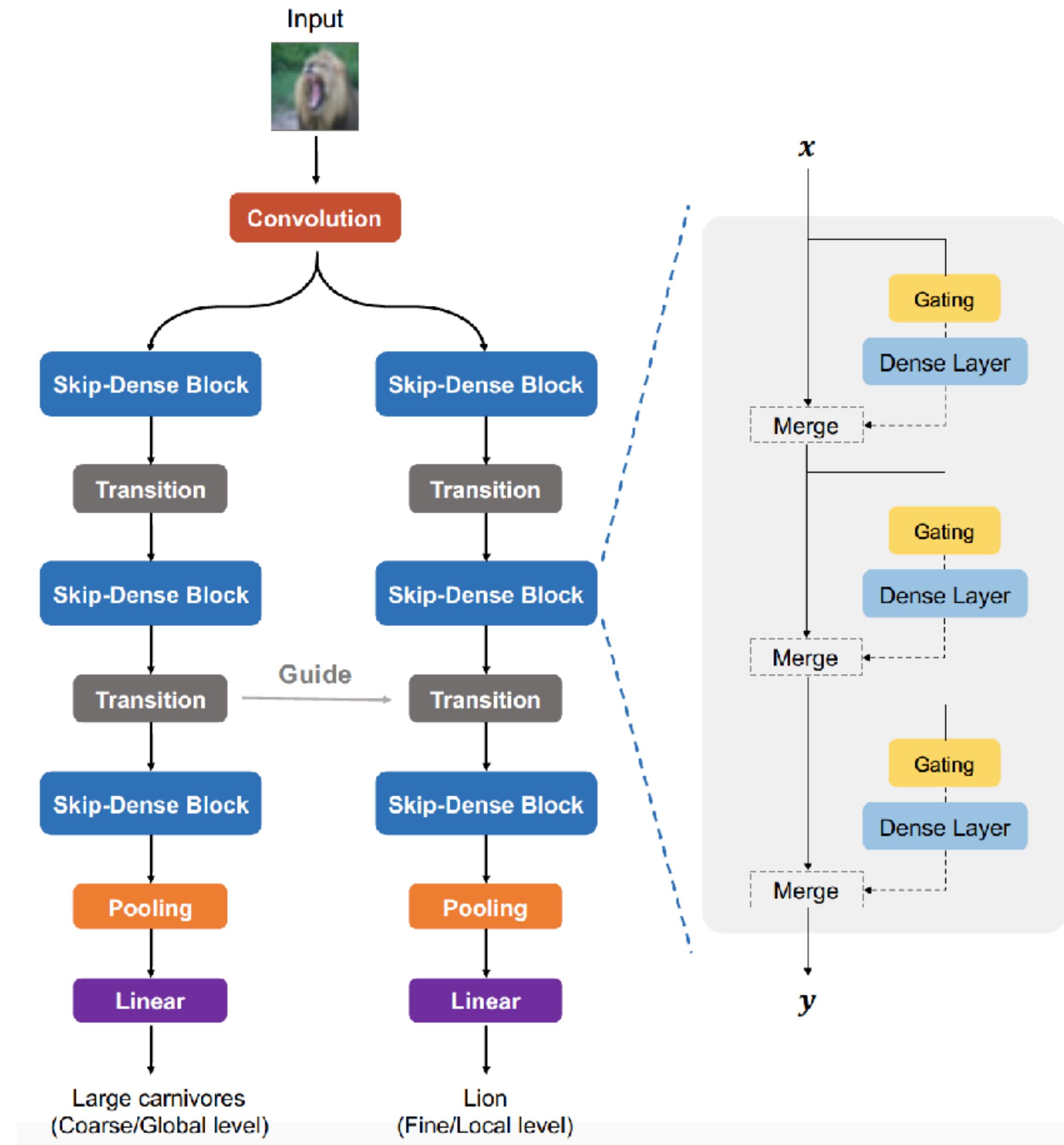
(1)



A

(2)

Figure 1: (2-A) Example images to assess cerebral asymmetries for spatial frequencies. (2-B) Hemispheric specialization: the left hemisphere (LH)/the right hemisphere (RH) are predominantly involved in the local/global letter identification. Figures from (Kauffmann, Ramanoël, and Peyrin 2014).

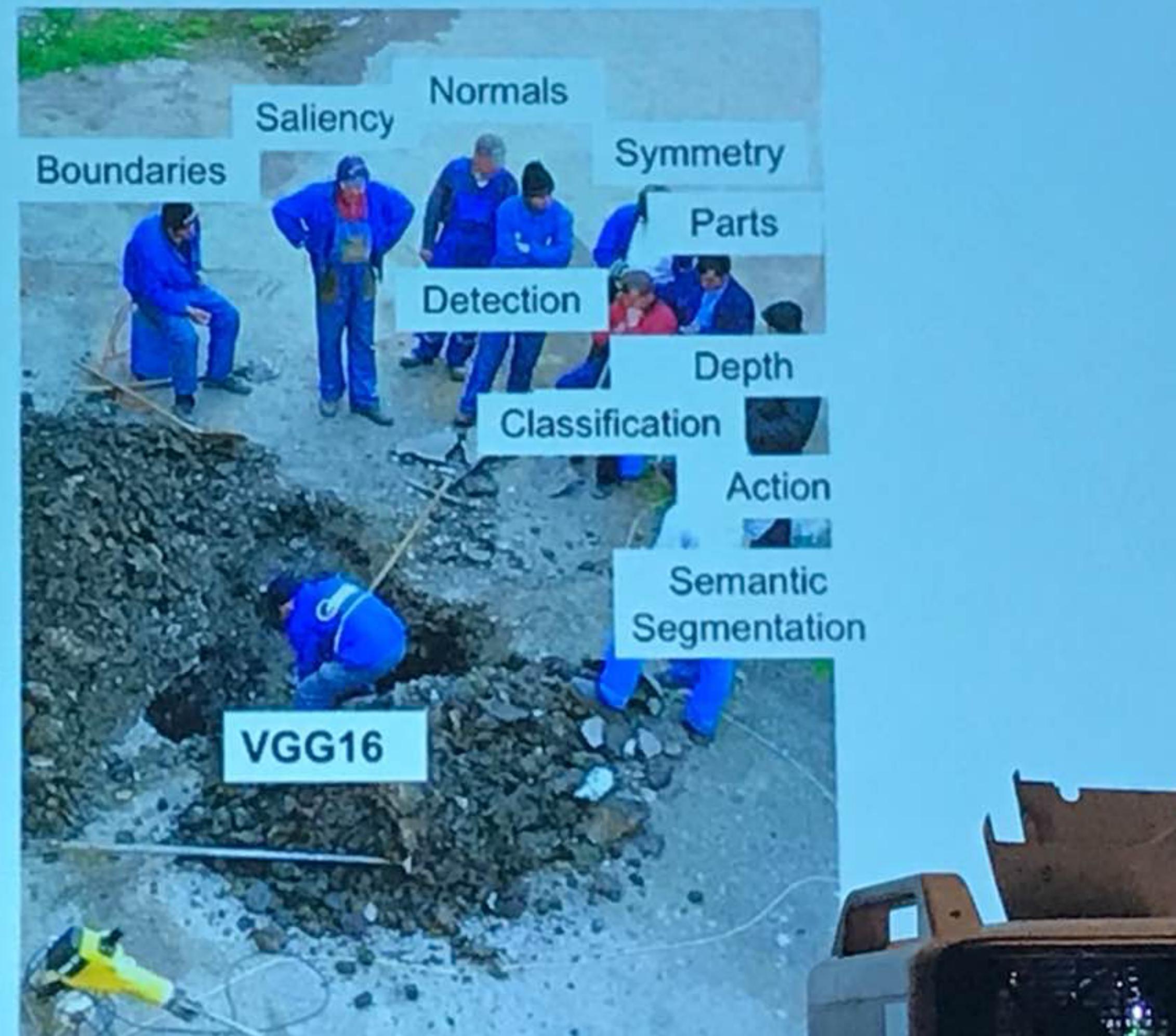


Neural Network In General

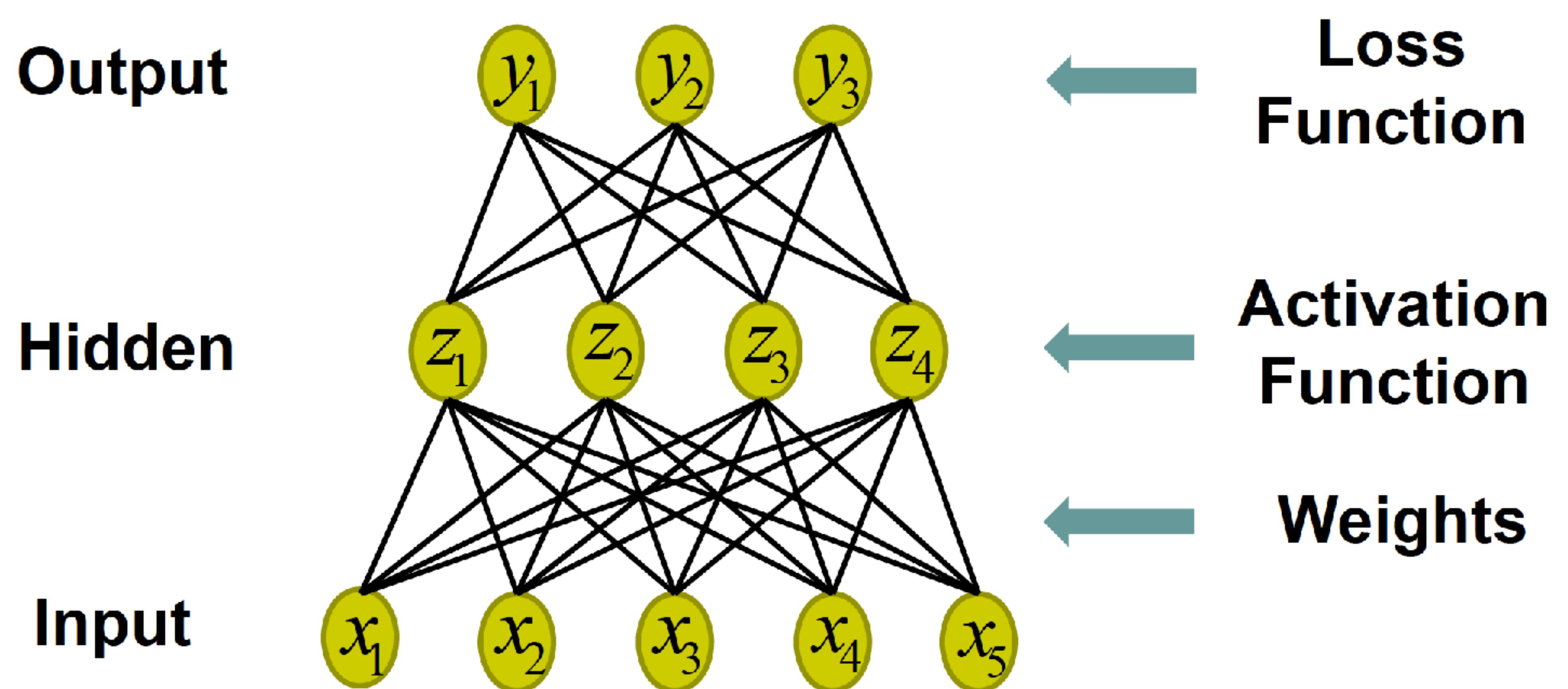


A real-life analogy

Stronger workers
(ResNets)
More workers
Some coordination!



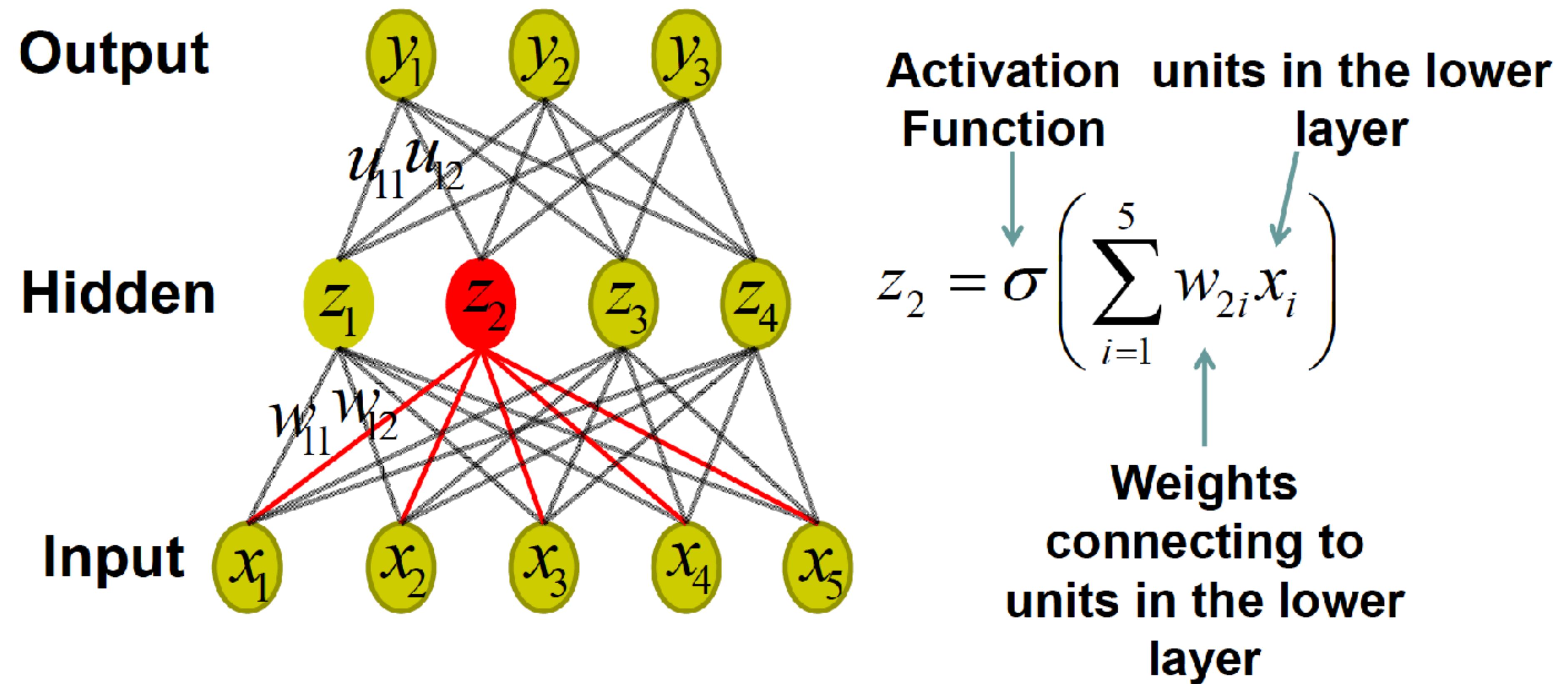
Neural Network



Local Computation At Each Unit

Optional subtitle

Linear Combination + Nonlinear Activation



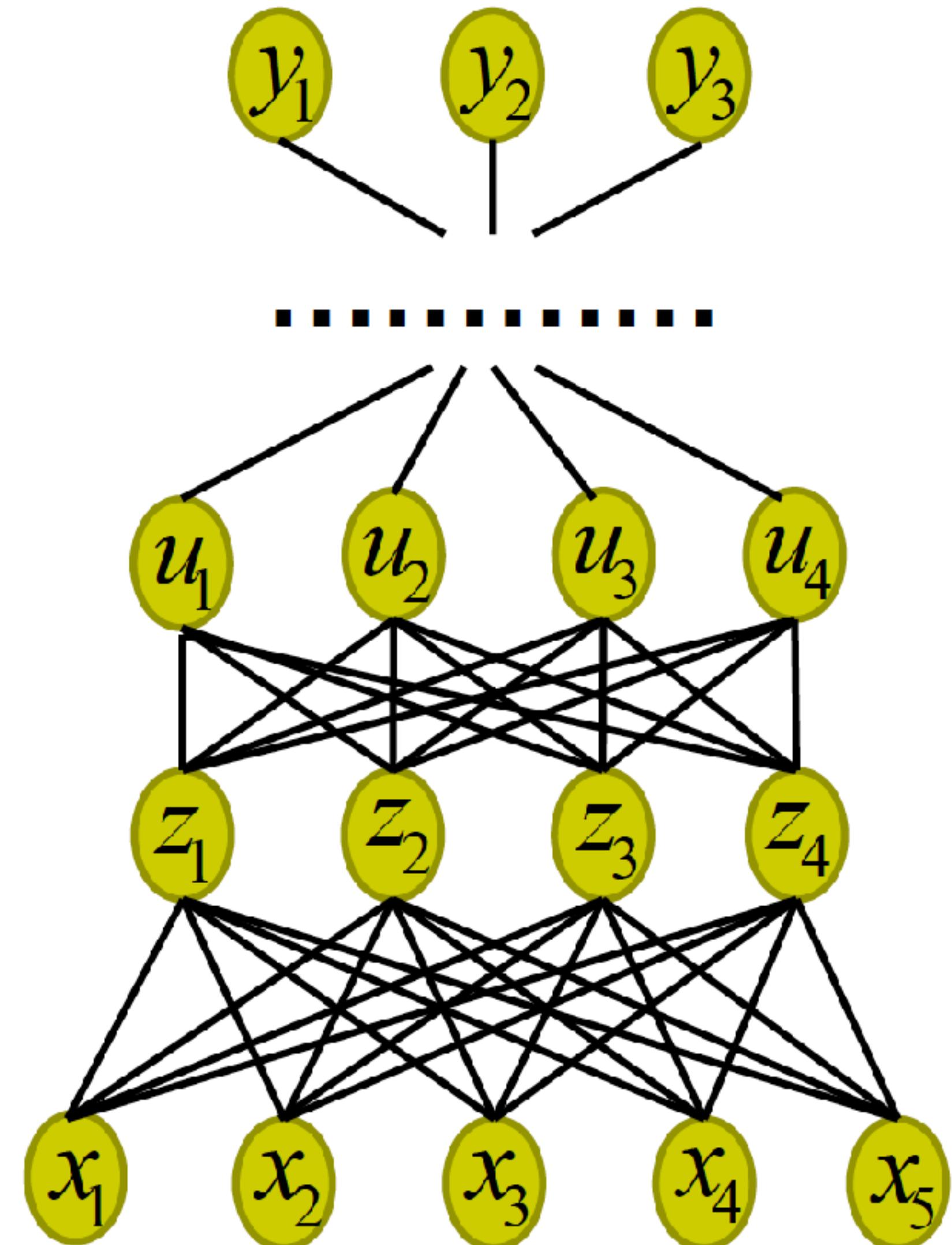
Deep Neural Network

Optional subtitle

Output

**More
Hidden
Layers**

Input

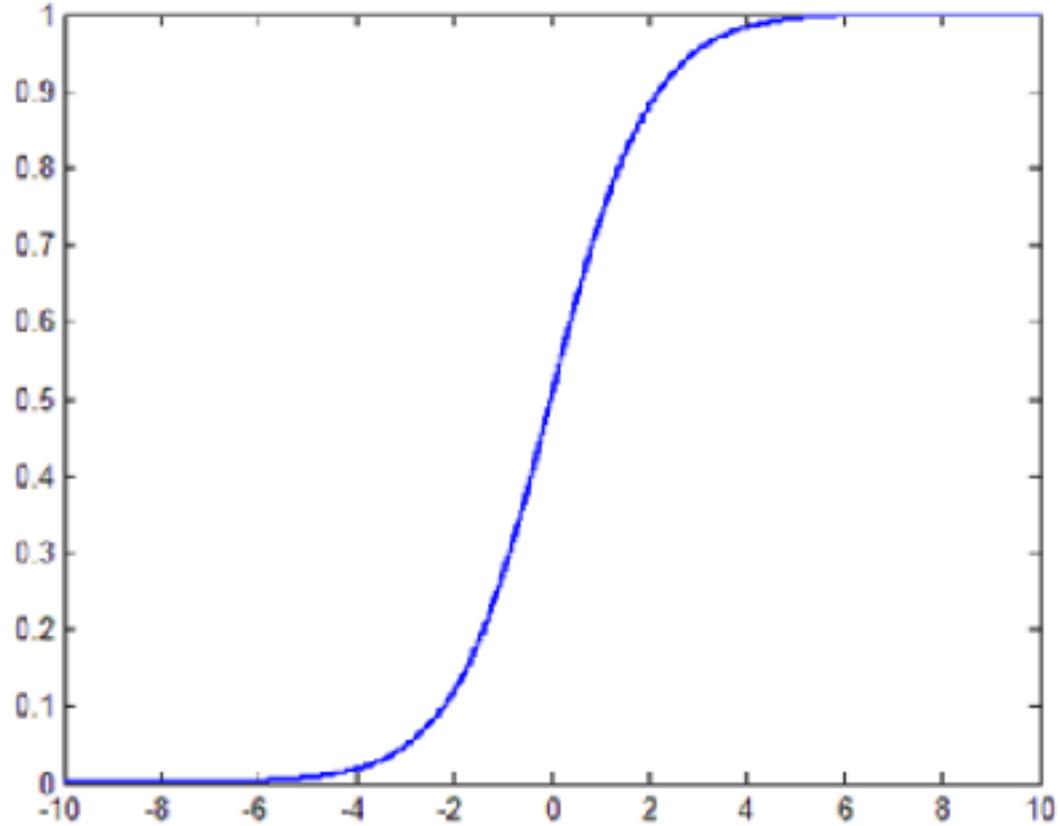


Activation Functions

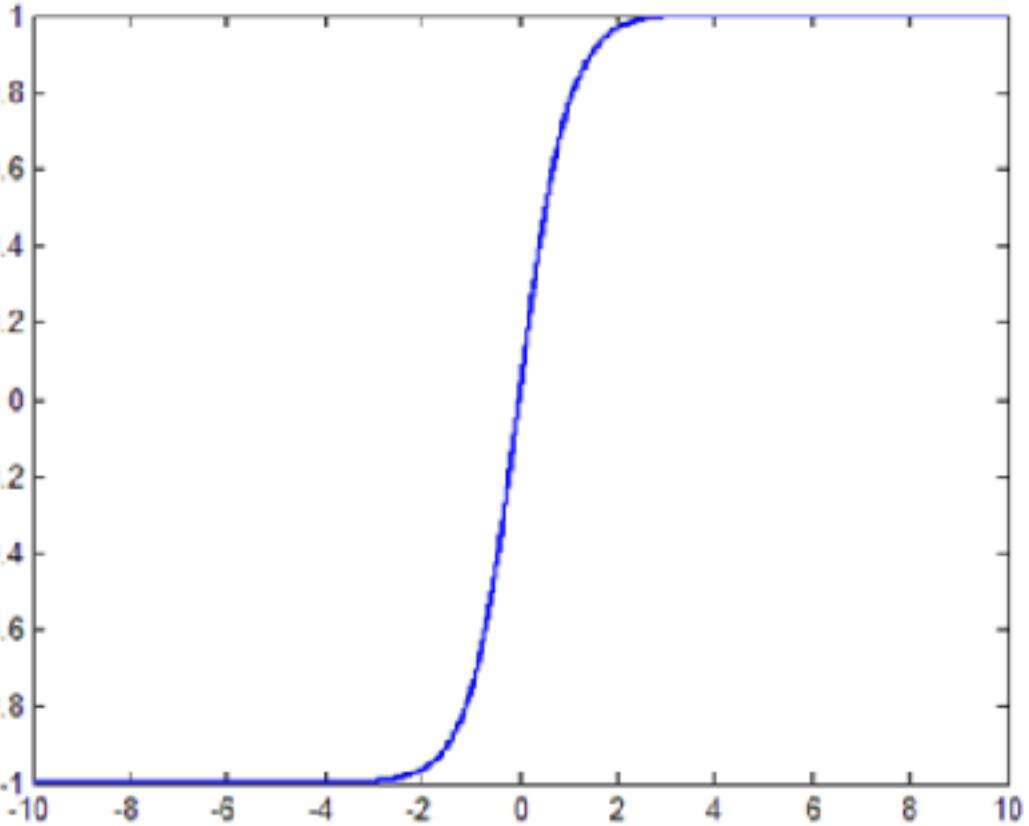
Optional subtitle

- Applied on the hidden units
- Achieve nonlinearity
- Popular activation functions

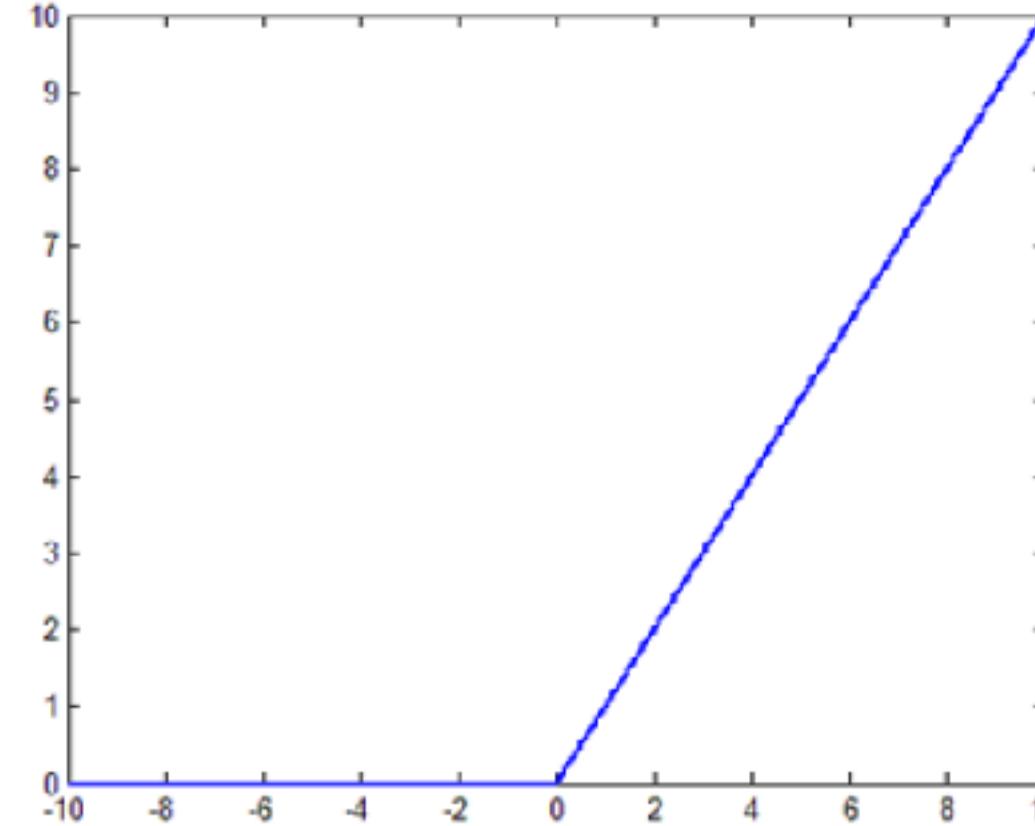
Sigmoid



Tanh



Rectified Linear



More Activation functions

Popular choice of $f(\cdot)$

- Sigmoid function

$$f(s) = \frac{1}{1 + e^{-s}}$$

- Tanh function (shift the center of Sigmoid to the origin)

$$f(s) = \frac{e^s - e^{-s}}{e^s + e^{-s}}$$

- Hard tanh

$$f(s) = \max(-1, \min(1, x))$$

- Rectified linear unit (ReLU)

$$f(s) = \max(0, x)$$

- Softplus: smooth version of ReLU

$$f(s) = \log(1 + e^s)$$

- Softmax: mostly used as output non-linearity for predicting discrete probabilities

$$f(s_k) = \frac{e^{s_k}}{\sum_{k'=1}^C e^{s_{k'}}}$$

- Maxout: it generalizes the rectifier assuming there are multiple netactivations

$$f(s_1, \dots, s_n) = \max_i(s_i)$$



Reading Material – “Recent Advances in Convolutional Neural Networks”

Optional subtitle

<https://arxiv.org/pdf/1512.07108.pdf>

The screenshot shows a red header bar with a search icon and the text '(Help |)'. Below it is a grey navigation bar with the text 'Computer Science > Computer Vision and Pattern Recognition'. The main title 'Recent Advances in Convolutional Neural Networks' is displayed in large bold black font. Below the title, the authors' names are listed in blue: Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Liyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxin Tsuhan Chen. A note at the bottom indicates the paper was submitted on Dec 22, 2015 (v1) and last revised on Oct 19, 2017 (this version, v6).

Recent Advances in Convolutional Neural Networks

Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Liyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxin Tsuhan Chen

(Submitted on 22 Dec 2015 (v1), last revised 19 Oct 2017 (this version, v6))

In the last few years, deep learning has led to very good performance on a variety of problems, such as visual reprocessing. Among different types of deep neural networks, convolutional neural networks have been most extended due to the amount of the annotated data and the great improvements in the strengths of graphics processor units, the emerged swiftly and achieved state-of-the-art results on various tasks. In this paper, we provide a broad survey of convolutional neural networks. We detailize the improvements of CNN on different aspects, including layer design, activation function, computation. Besides, we also introduce various applications of convolutional neural networks in computer vision.

Comments: Pattern Recognition, Elsevier

Subjects: Computer Vision and Pattern Recognition (cs.CV); Learning (cs.LG); Neural and Evolutionary Computing (cs.NE)

Cite as: [arXiv:1512.07108 \[cs.CV\]](https://arxiv.org/abs/1512.07108)

(or [arXiv:1512.07108v6 \[cs.CV\]](https://arxiv.org/abs/1512.07108v6) for this version)

Submission history

From: Jiuxiang Gu Mr [view email]

[v1] Tue, 22 Dec 2015 14:54:34 GMT (1693kb,D)

[v2] Tue, 5 Jul 2016 11:39:16 GMT (578kb,D)

[v3] Mon, 1 Aug 2016 01:54:59 GMT (439kb,D)

[v4] Sat, 6 Aug 2016 12:38:35 GMT (438kb,D)

[v5] Thu, 5 Jan 2017 05:45:53 GMT (3100kb,D)

[v6] Thu, 19 Oct 2017 16:34:35 GMT (4099kb,D)



Loss Functions

Optional subtitle

- Squared loss for regression

$$(y - t)^2$$

A diagram showing the squared loss function $(y - t)^2$. It consists of two teal arrows pointing towards each other from below, forming a V-shape. Below the left arrow is the text "Prediction" and below the right arrow is the text "True value".

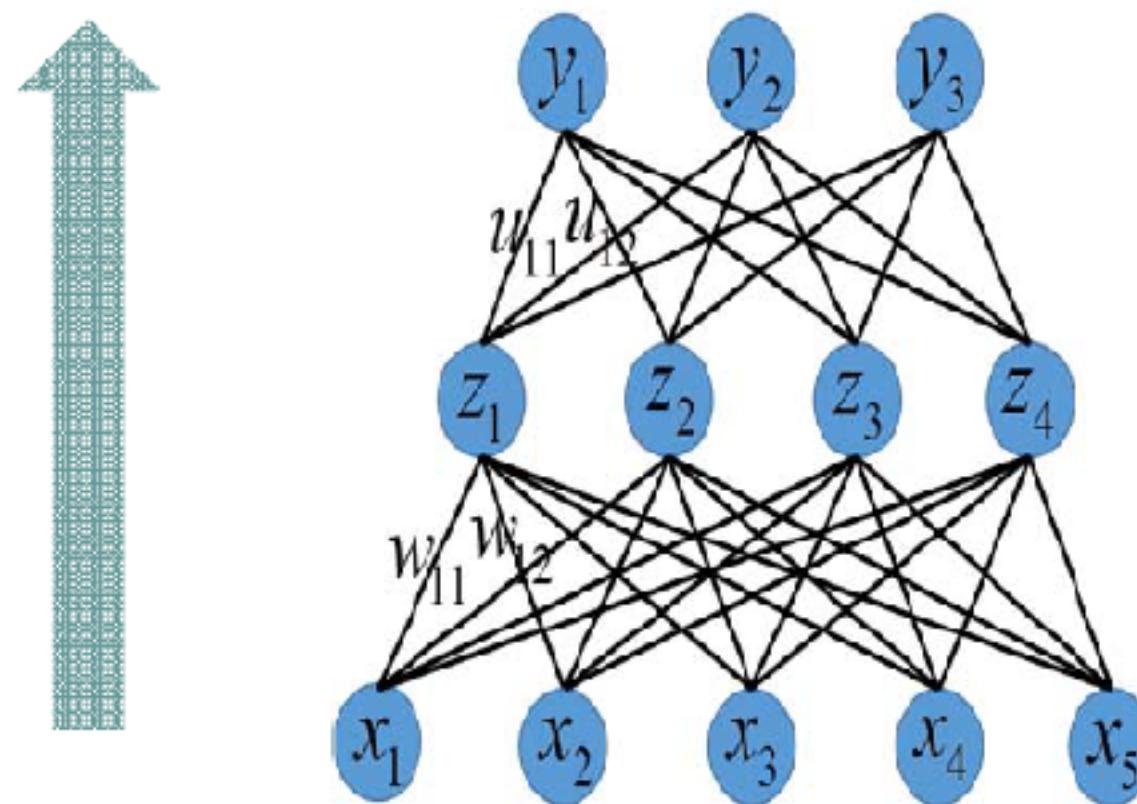
- Cross entropy loss for classification

$$-\sum_{k=1}^K t_k \ln a_k \quad a_k = \frac{\exp(y_k)}{\sum_{j=1}^K \exp(y_j)}$$

A diagram showing the cross entropy loss function. It consists of two teal arrows pointing upwards from below, forming a V-shape. Below the left arrow is the text "Class label" and below the right arrow is the text "Prediction".

Neural Network Prediction

- Compute unit values layer by layer in a forward manner



- Prediction function

Activation Function

Input

Output

Weights

$$y_k = \sum_{j=1}^4 u_{kj} \sigma(\sum_{i=1}^5 w_{ji} x_i)$$

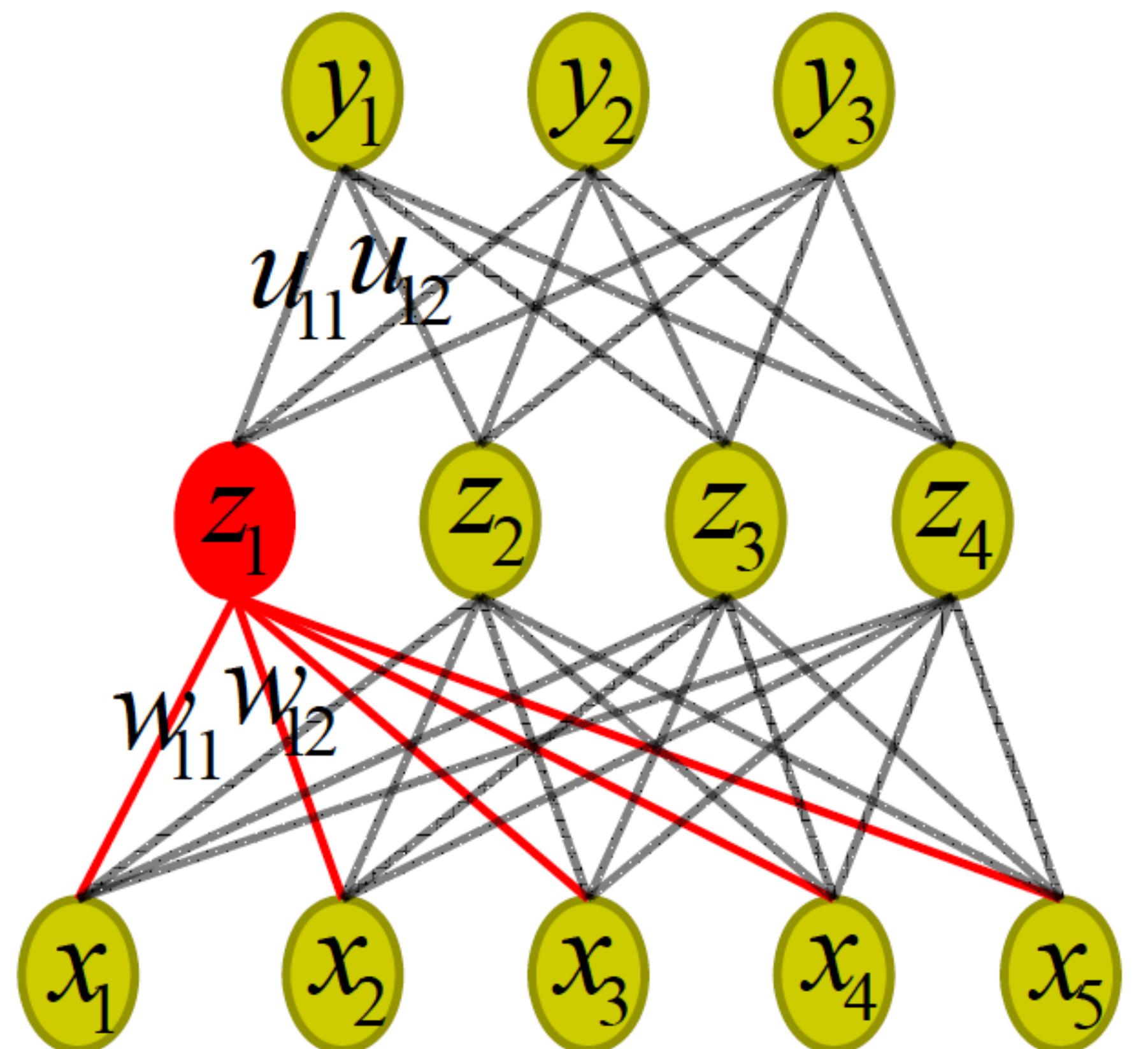
Neural Network Prediction

Optional subtitle

Output

Hidden

Input

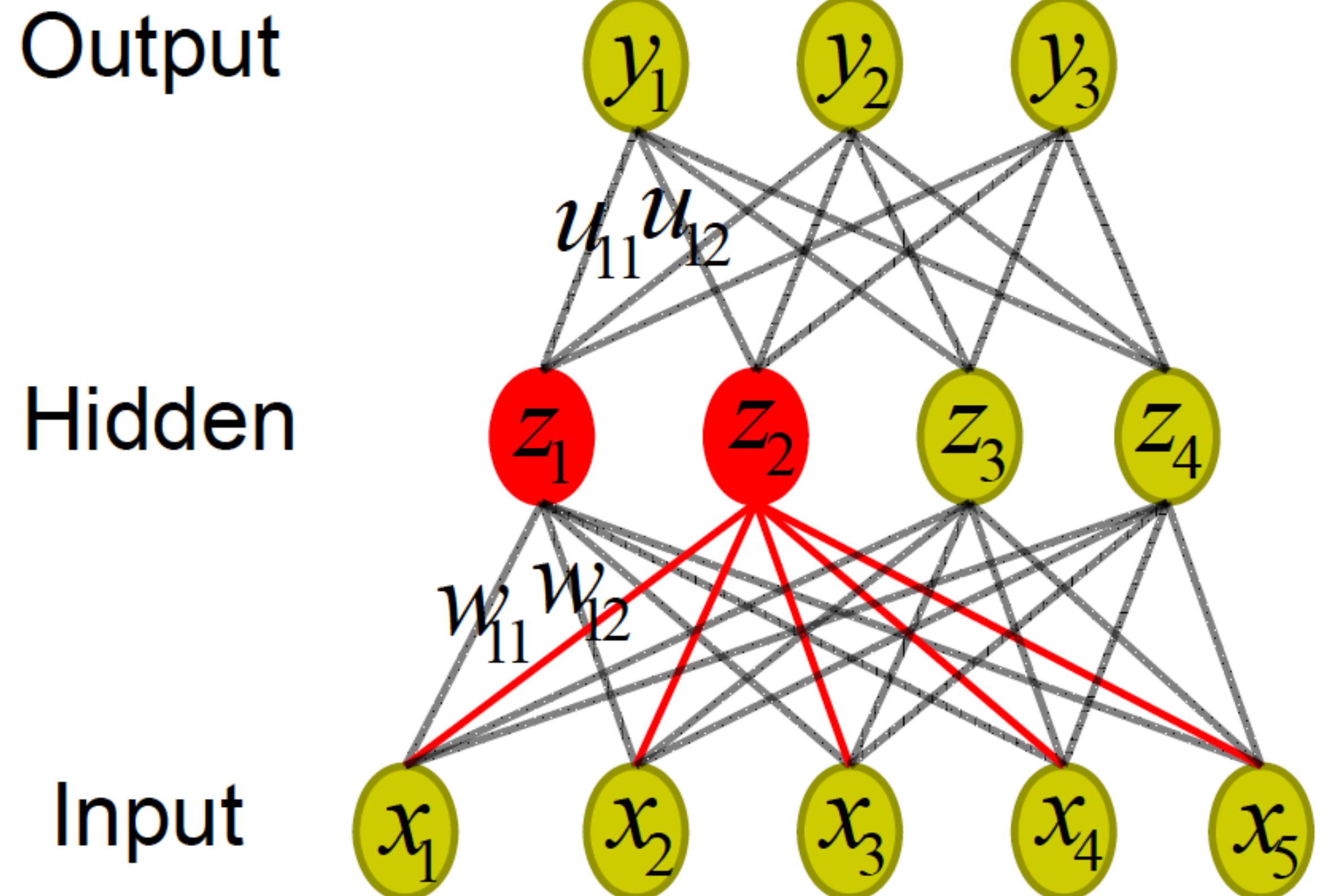


$$z_1 = \sigma \left(\sum_{i=1}^5 w_{1i} x_i \right)$$

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

Neural Network Prediction

Optional subtitle

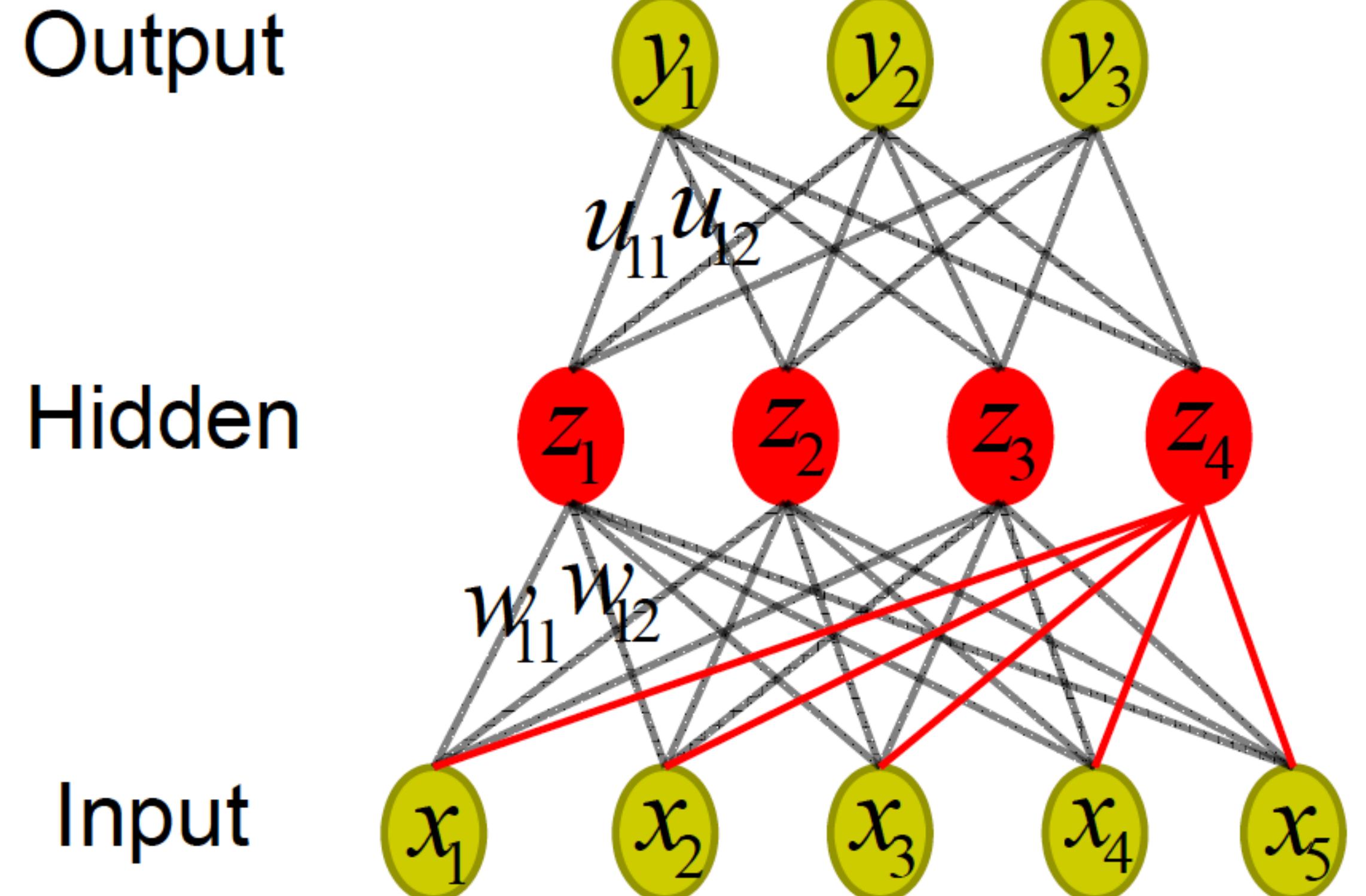


$$z_2 = \sigma \left(\sum_{i=1}^5 w_{2i} x_i \right)$$

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

Neural Network Prediction

Optional subtitle



$$z_4 = \sigma \left(\sum_{i=1}^5 w_{4i} x_i \right)$$

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

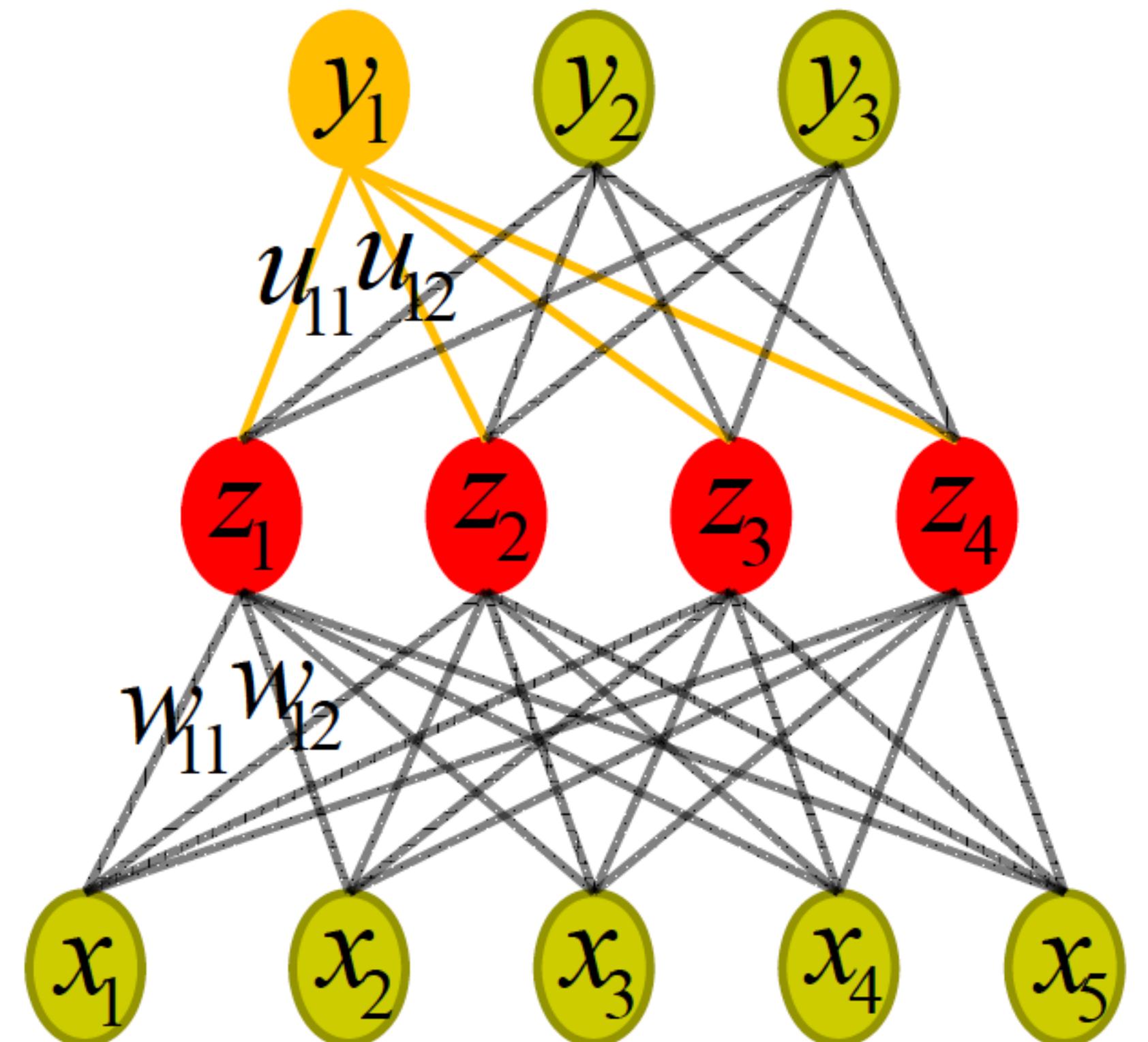
Neural Network Prediction

Optional subtitle

Output

Hidden

Input



$$y_1 = \sigma \left(\sum_{i=1}^4 u_{1i} z_i \right)$$

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

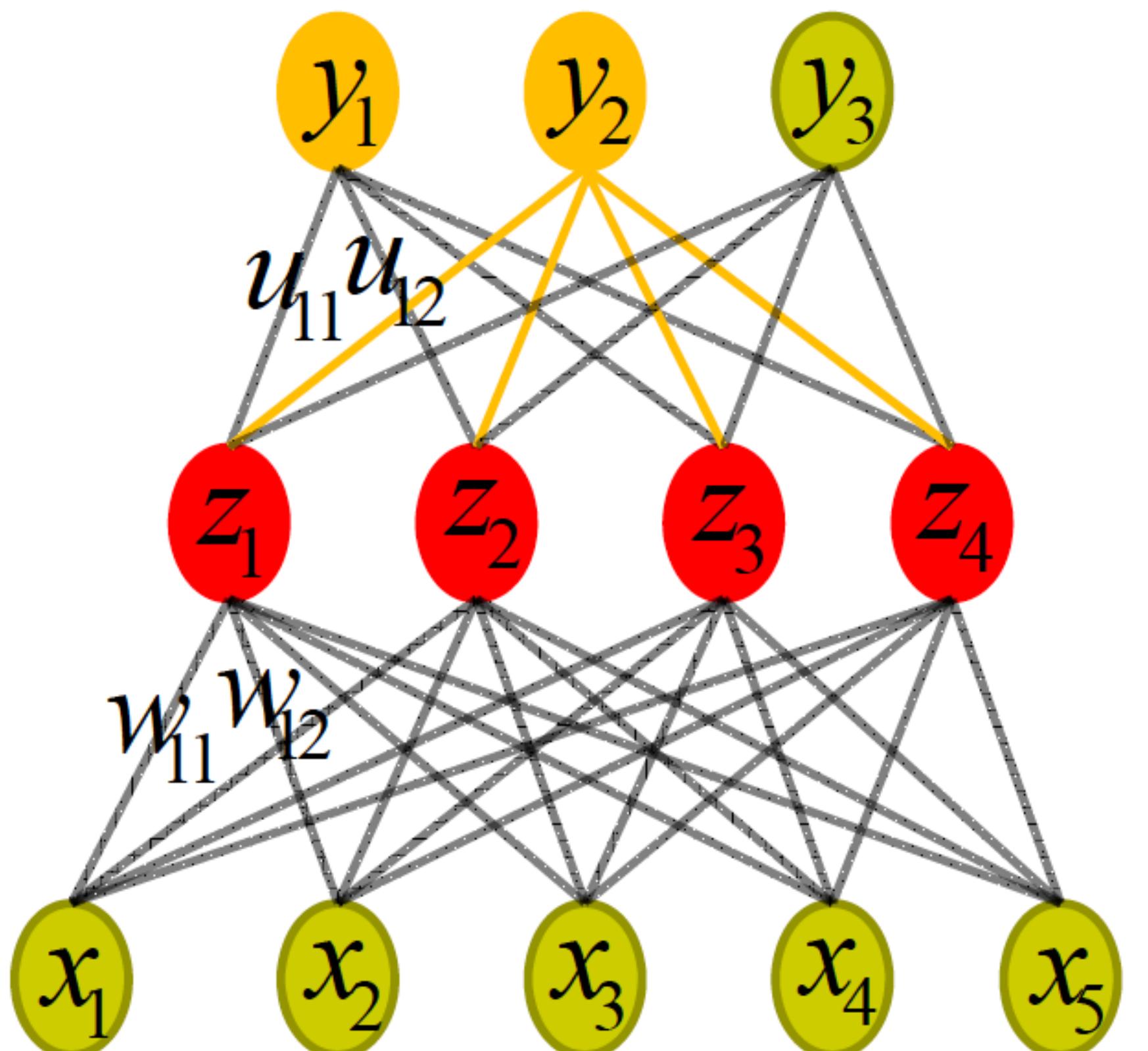
Neural Network Prediction

Optional subtitle

Output

Hidden

Input

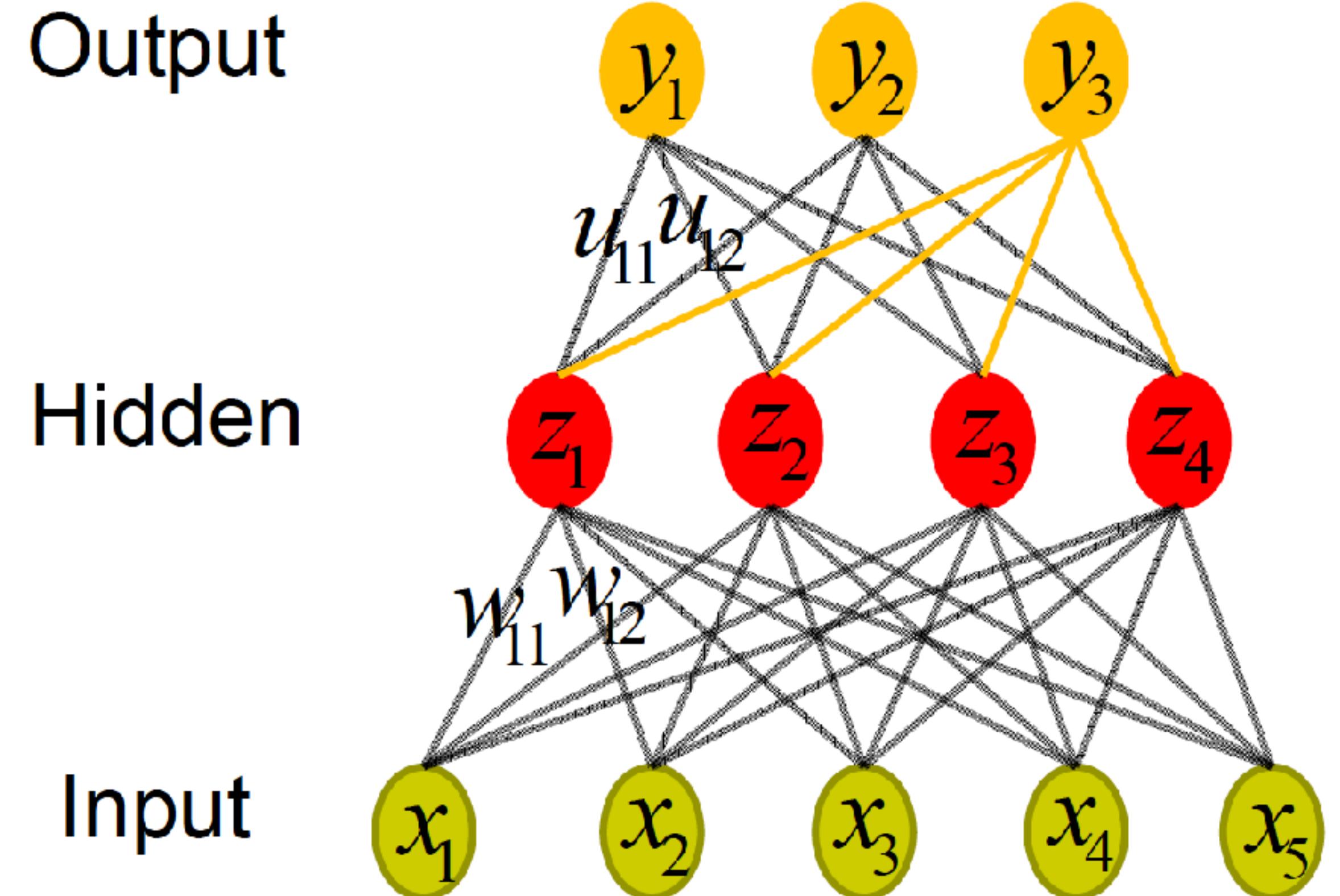


$$y_2 = \sigma \left(\sum_{i=1}^4 u_{2i} z_i \right)$$

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

Neural Network Prediction

Optional subtitle



$$y_3 = \sigma \left(\sum_{i=1}^4 u_{3i} z_i \right)$$

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

Neural Network Training

- Goal: compute gradient

$$\frac{\partial L}{\partial w_{ij}} \leftarrow \begin{array}{l} \text{Training loss} \\ \text{Weight between unit } i \text{ and } j \end{array}$$

- Apply chain rule

$$\frac{\partial L}{\partial w_{ji}} = \frac{\partial L}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}} \leftarrow \begin{array}{l} \text{Linear combination} \\ \text{value } a_j = \sum_i w_{ji} z_i \end{array}$$

$$\boxed{\frac{\partial L}{\partial a_j}} = \sum_k \frac{\partial L}{\partial a_k} \frac{\partial a_k}{\partial a_j}$$

Called error, computed
recursively in a
backward manner



Neural Network Training

Optional subtitle

- Apply chain rule (cont'd)

$$\frac{\partial a_j}{w_{ji}} = z_i \leftarrow \text{Derivative of activation function}$$
$$\frac{\partial a_k}{\partial a_j} = \frac{\partial a_k}{\partial z_j} \frac{\partial z_j}{\partial a_j} = w_{kj} \sigma'(a_j)$$

$\text{gradient} = \frac{\partial L}{\partial w_{ji}} = \frac{\partial L}{\partial a_j} \frac{\partial a_j}{w_{ji}} = \text{backward error} \times \text{forward activation}$

- Pseudo code of BP

While not converge

1. compute forward activations
2. compute backward errors
3. compute gradients of weights
4. perform gradient descent



Pretraining

Optional subtitle

- A better initialization strategy of weight parameters
 - Based on Restricted Boltzmann Machine
 - An auto-encoder model
 - Unsupervised
 - Layer-wise, greedy
- Useful when training data is limited
- Not necessary when training data is rich



Multilayer Perceptrons (MLPs) are universal approximators.

- **MLPs are very flexible models**: they can approximate any smooth input-output transformation. **Really?!!**
- Most widely used pattern recognition models (such as SVM, boosting, and KNN) can be approximated as neural networks with one or two hidden layers. They are called **models with shallow architectures**.
- Shallow models divide the feature space into regions and match templates in local regions. $O(N)$ parameters are needed to represent N regions.
- **Deep architecture**: the number of hidden nodes can be reduced exponentially with more layers for certain problems. (1000-Layer ResNet)



Multilayer Perceptrons (MLPs) are universal approximators

Optional subtitle

An MLP with one hidden layer can approximate any smooth function to any desired accuracy, subject to a sufficient number of hidden nodes. [1]

[1] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximations. Neural Networks, 1989.

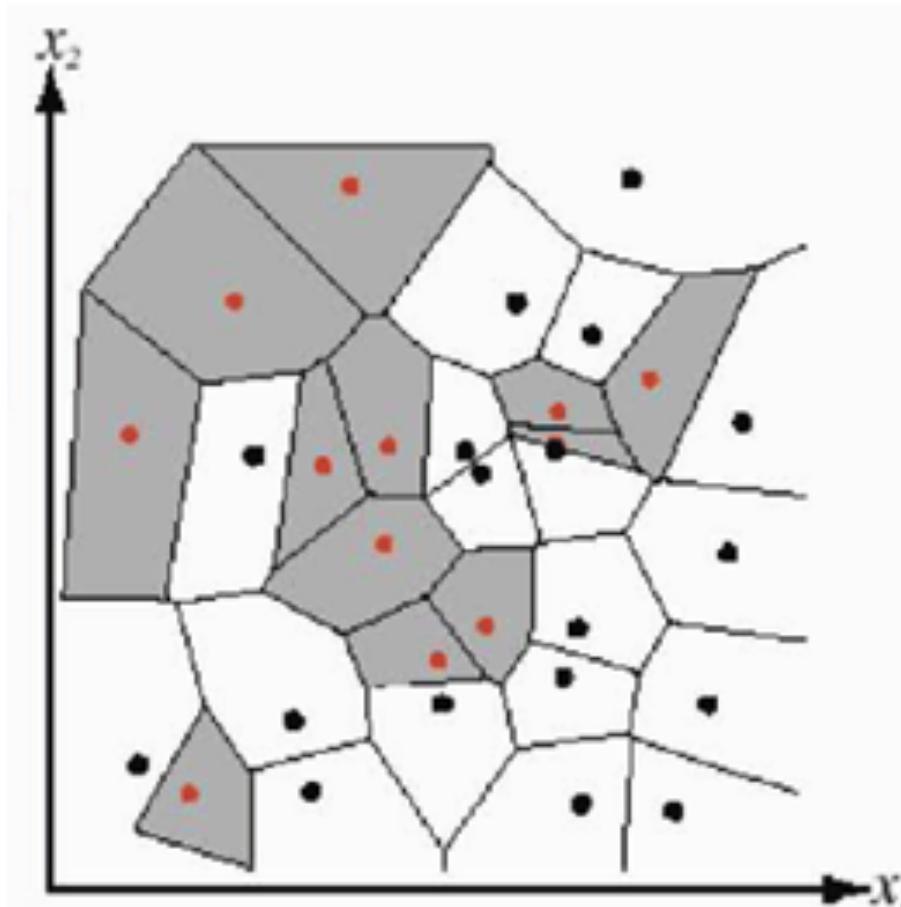
That's a very interesting result: neural-like architectures composed of simple units (linear summation and squashing sigmoidal transfer functions), organised in layers with at least a hidden layer are what we need to model arbitrary smooth input-output transformations.



Expressive power of a three-layer neural network

Optional subtitle

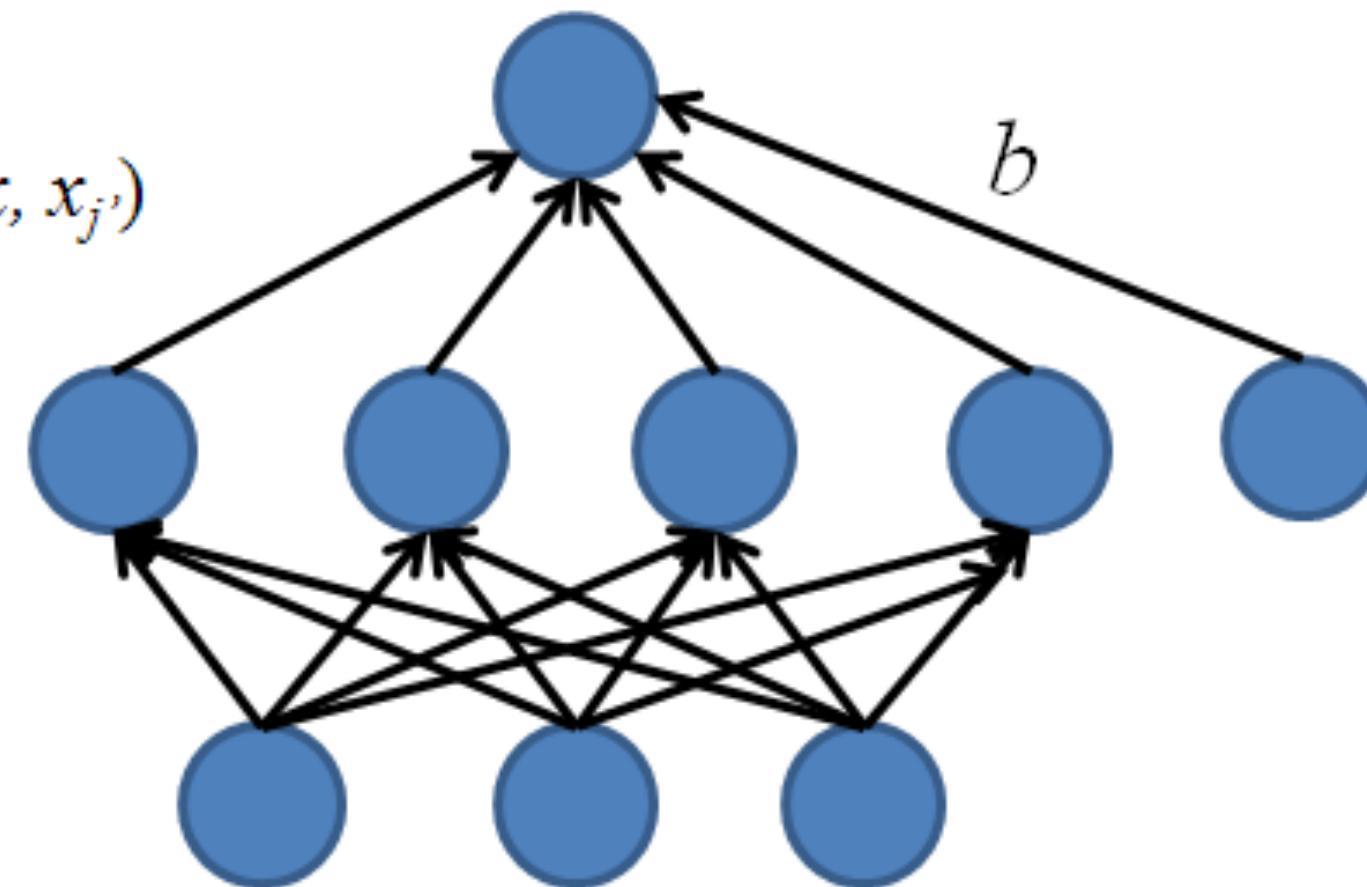
KNN



output $g(x) = \text{label}_j$
 $j = \operatorname{argmin}_{j'} d(x, x_{j'})$

hidden $d(x, x_j)$

input x

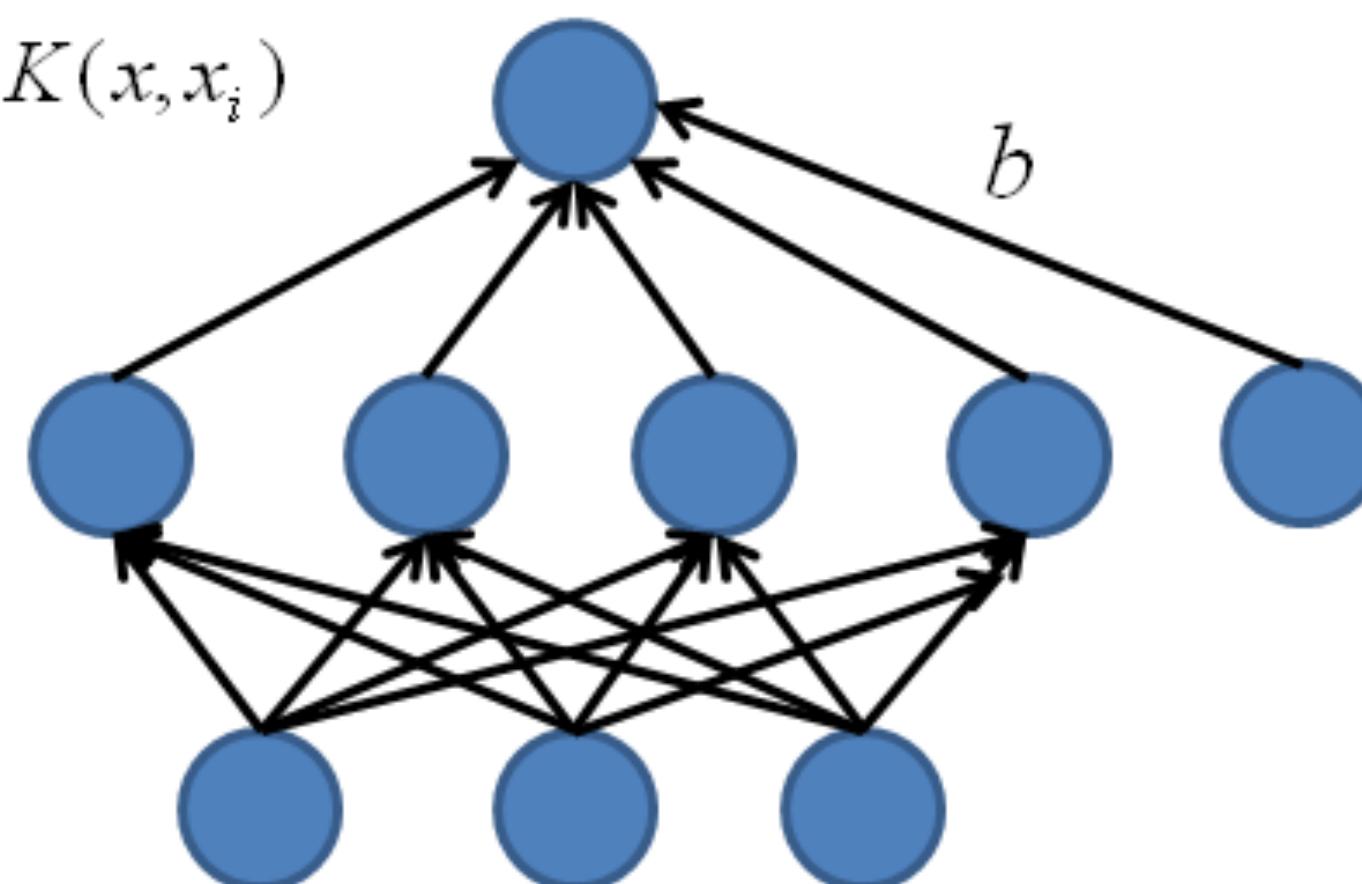


SVM

output $g(x) = b + \sum_i \alpha_i K(x, x_i)$

hidden $K(x, x_j)$

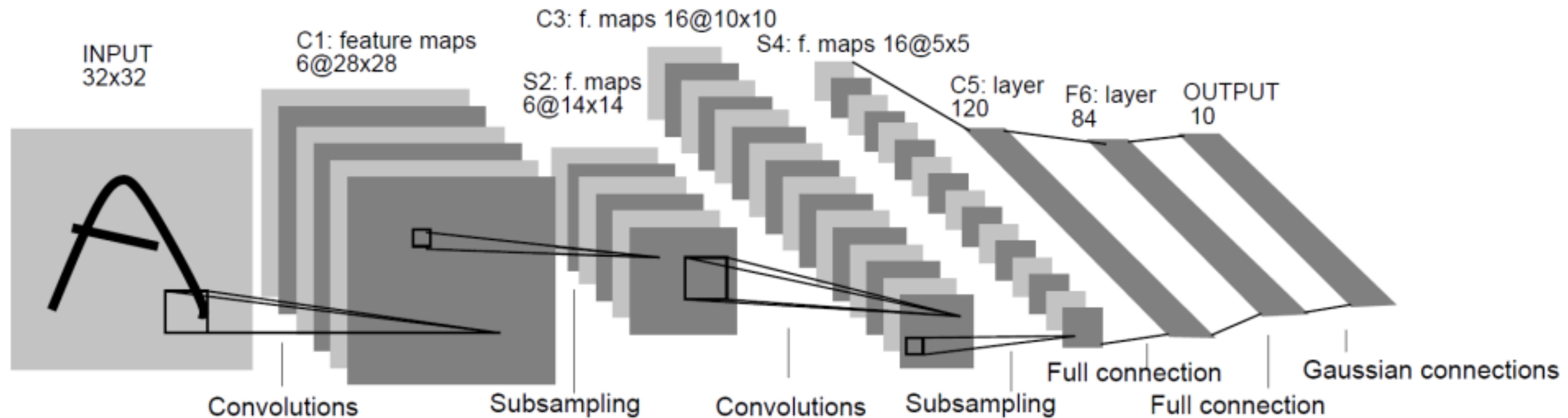
input x



Convolutional Neural Network(CNN)

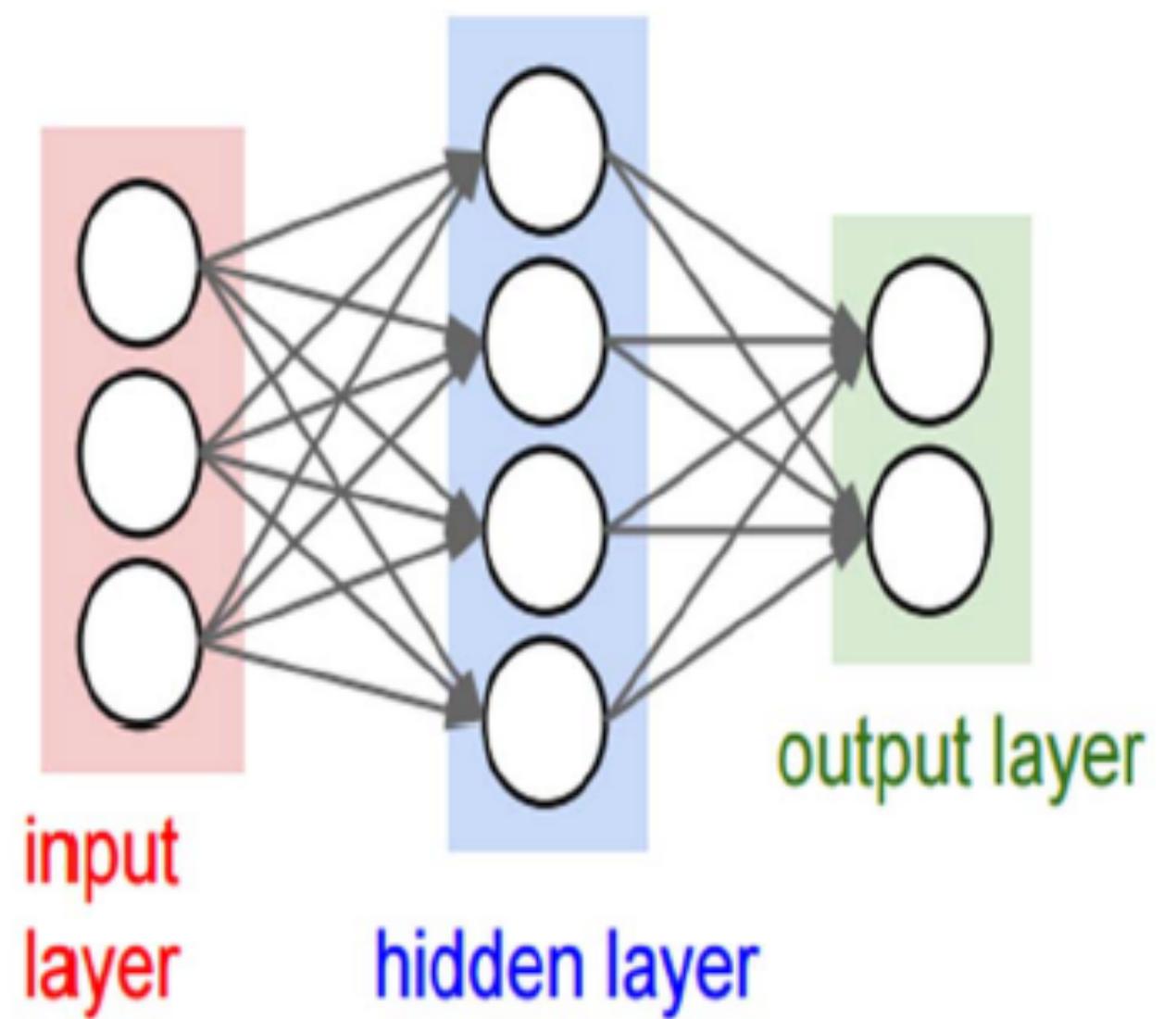


LeNet-5

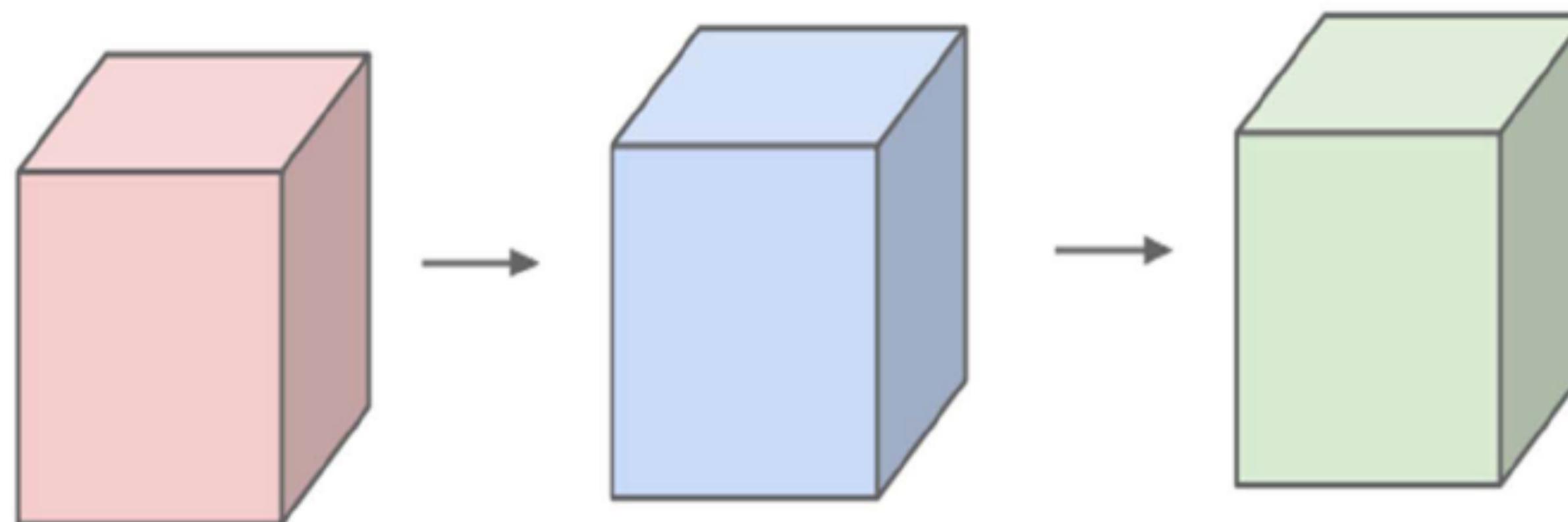


1998年发表的Gradient-based learning applied to Document Recognition一文中提出了LeNet-5
已是Convolutional Neural Network的基本框架

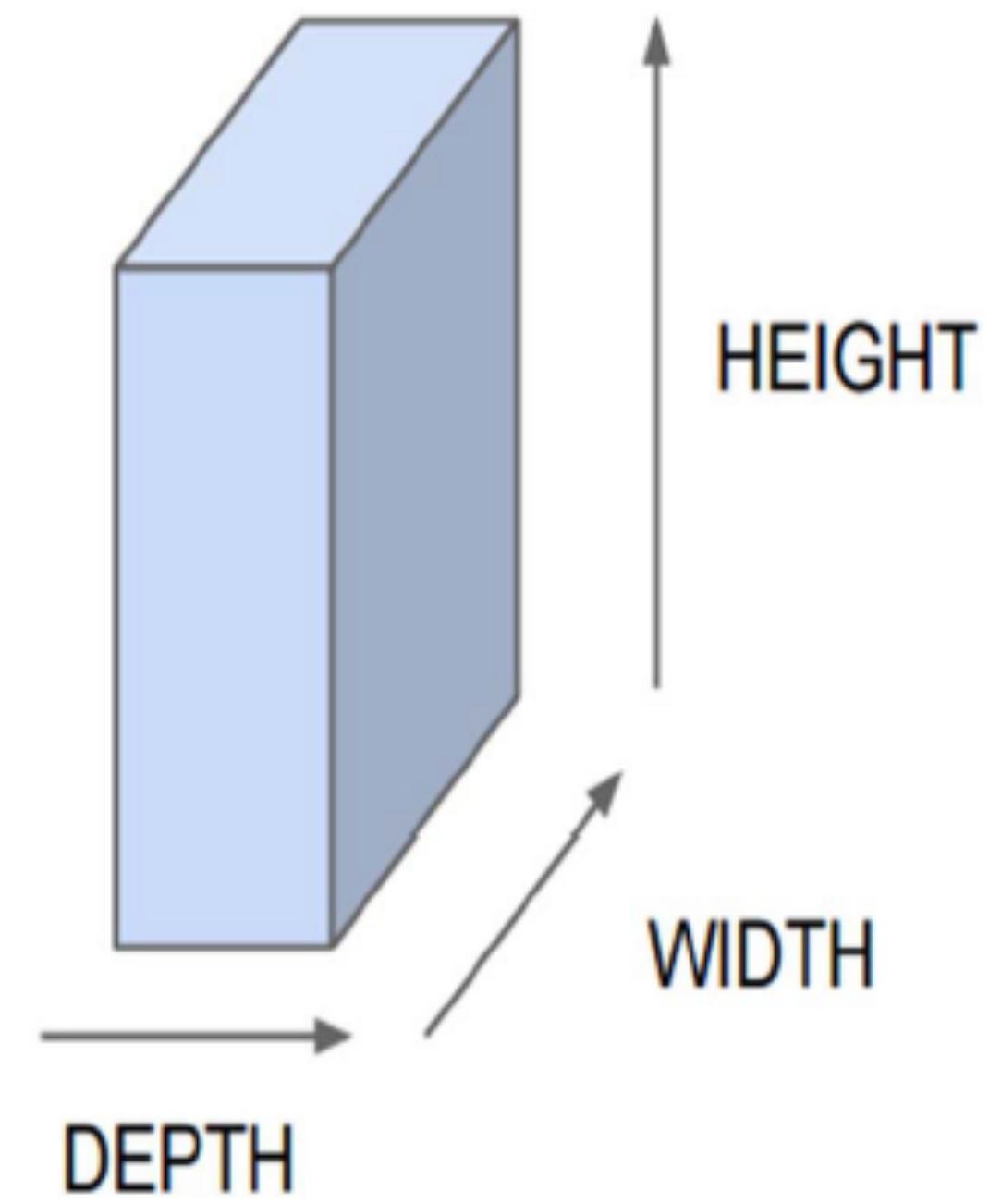
Ordinary Neural Network



Now



**All Neural Net
activations
arranged in 3
dimensions|**



For example, a CIFAR-10 image is a $32 \times 32 \times 3$ volume: 32 width, 32 height, 3 depth (RGB)

Local connectivity

Optional subtitle

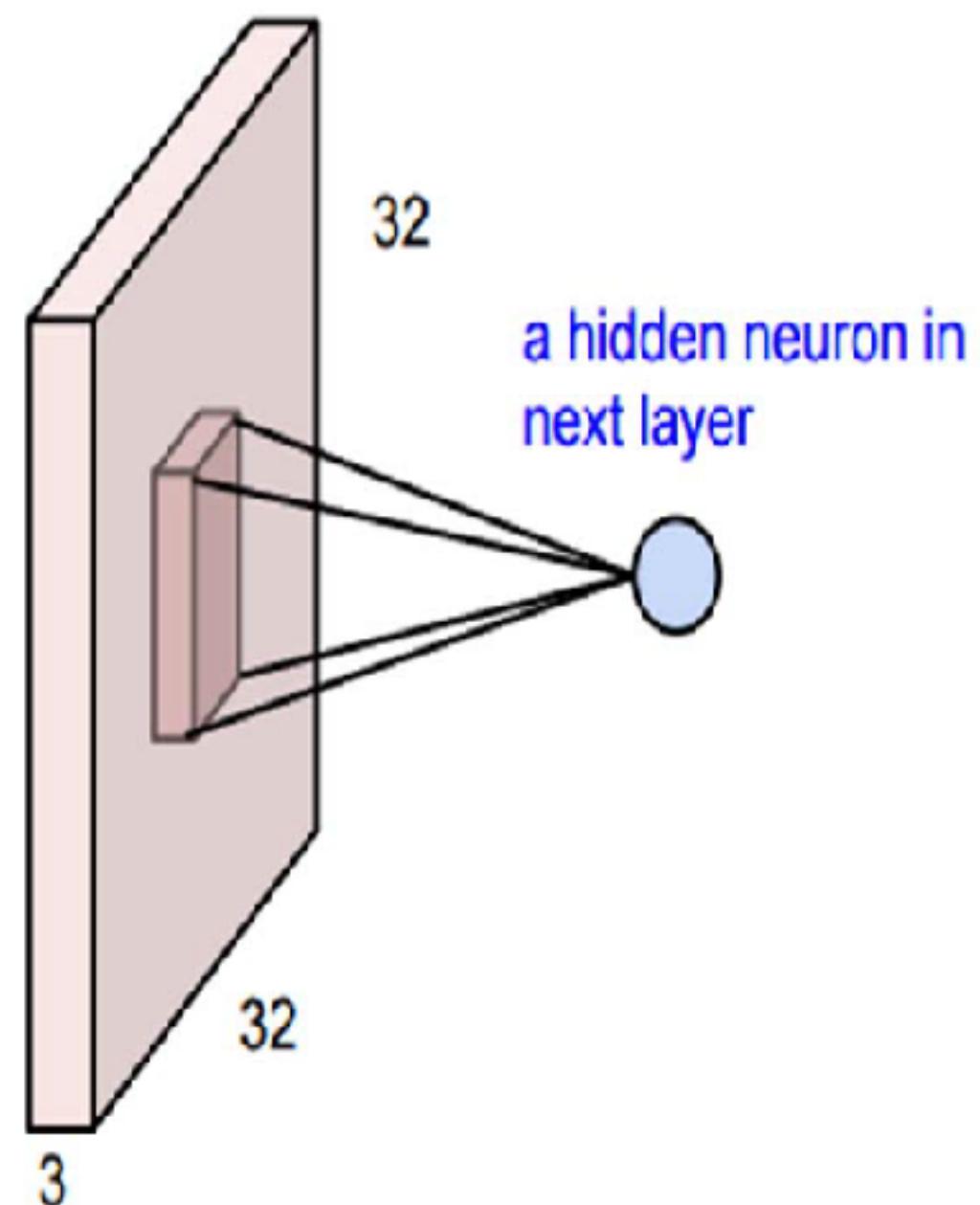
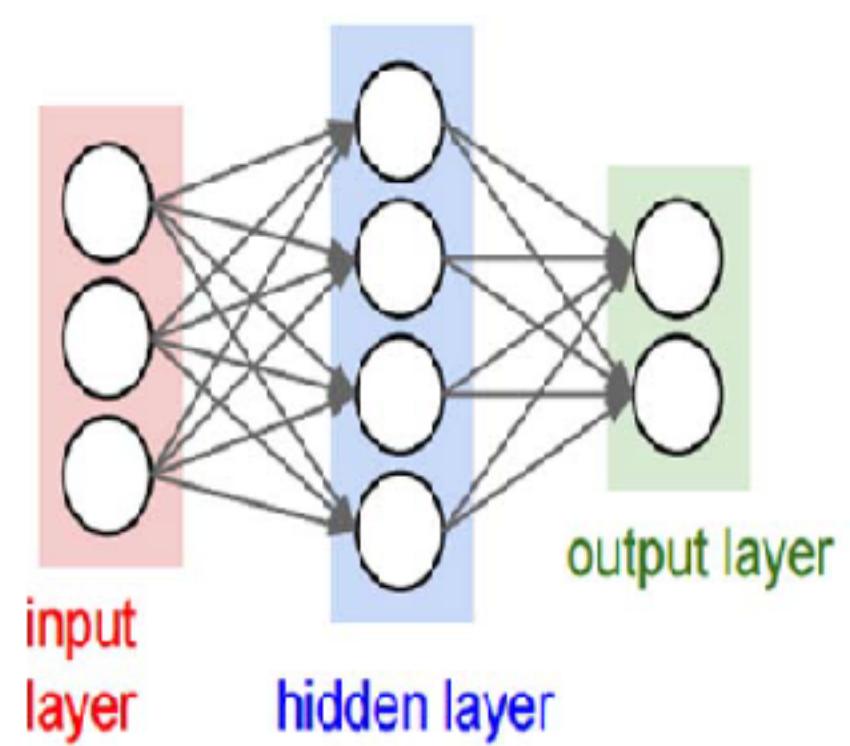


image: $32 * 32 * 3$ volume

**before: full connectivity:
 $32 * 32 * 3$ weights for each
neuron**

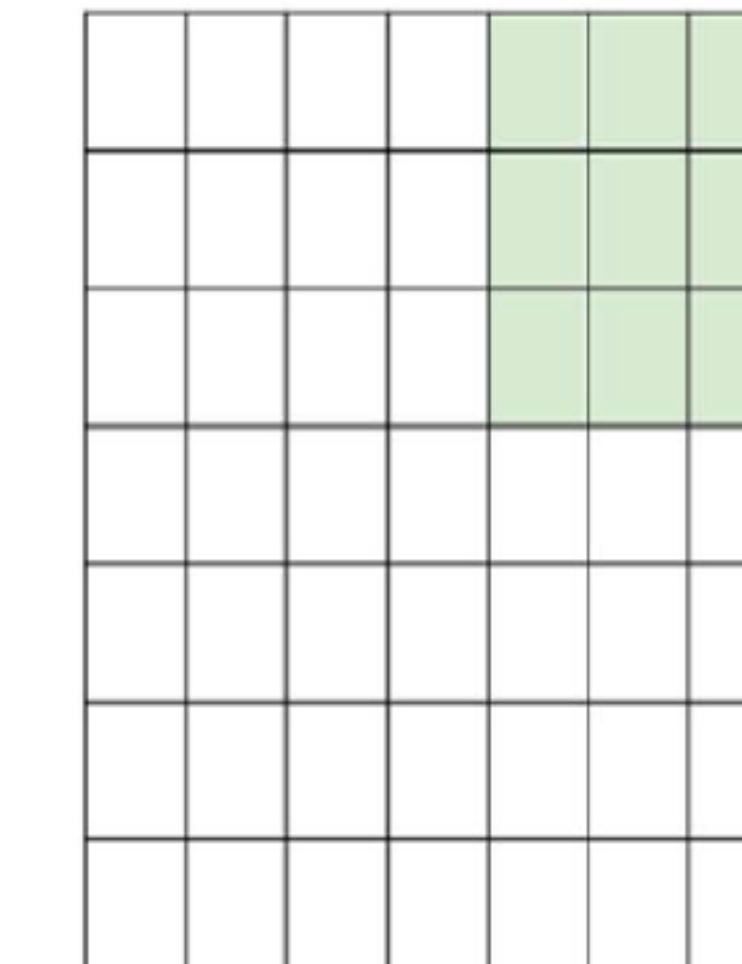
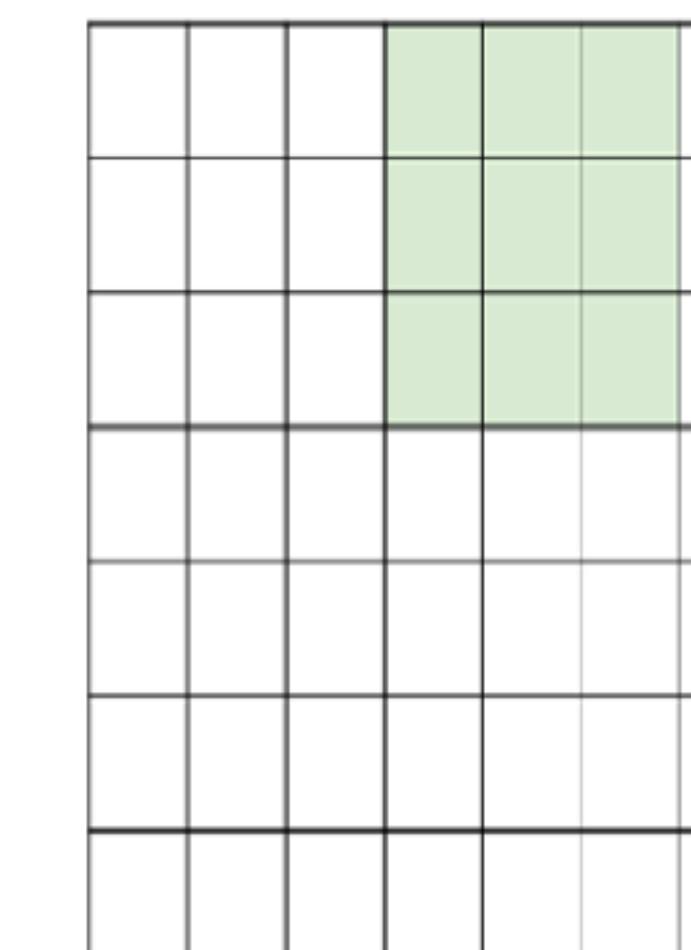
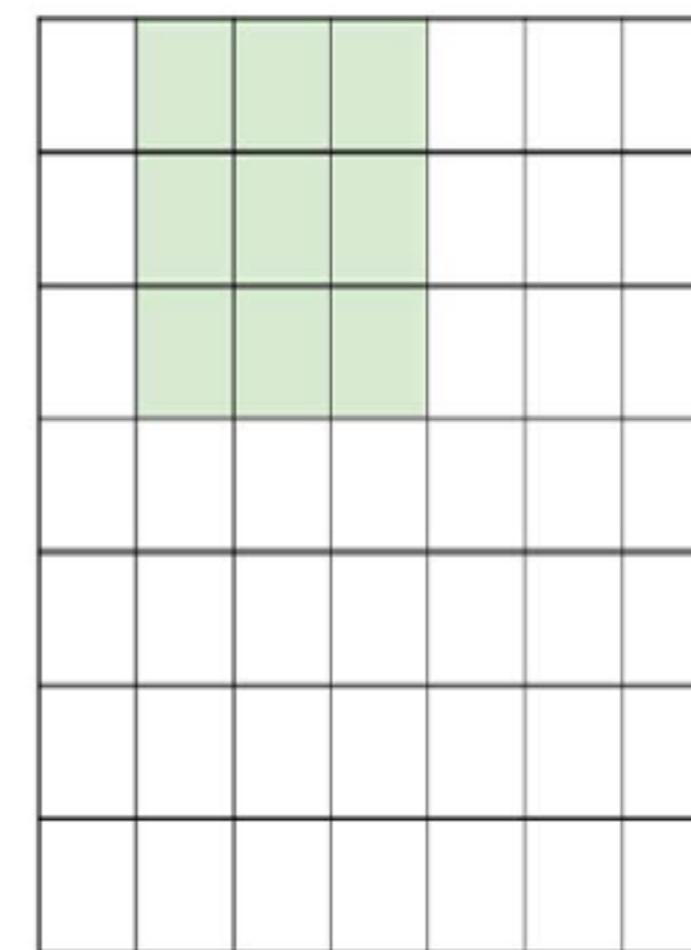
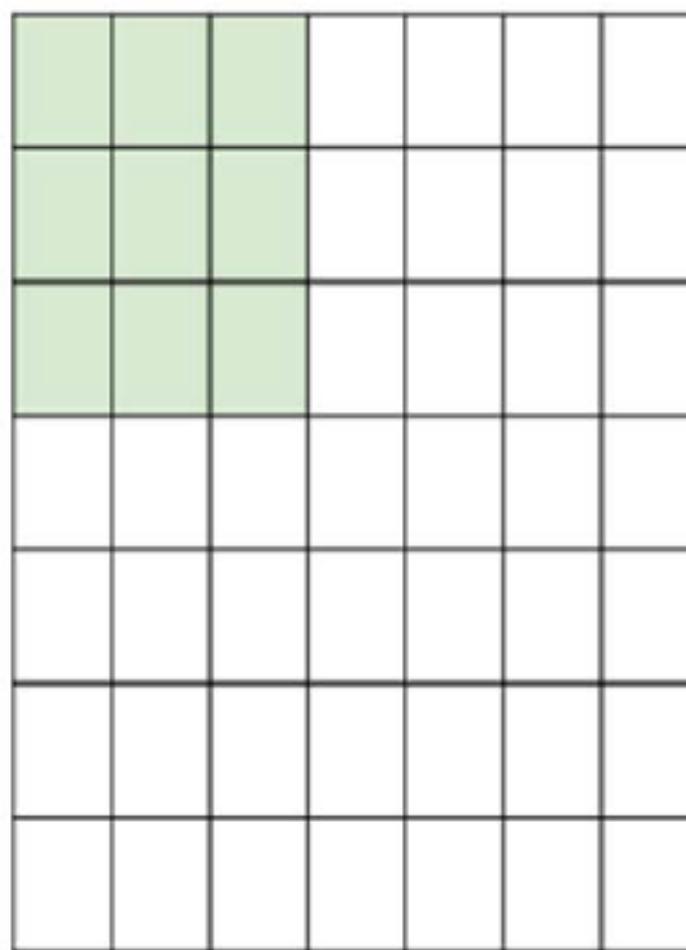
**now: one unit will connect
to, e.g. $5*5*3$ chunk and
only have $5*5*3$ weights**

Note the connectivity is:

- **local in space**
- **full in depth**

Convolution

- One local region only gives one output
- Convolution: Replicate the column of hidden units across space, with some stride



- 7 * 7 Input
- Assume 3*3 connectivity, stride = 1

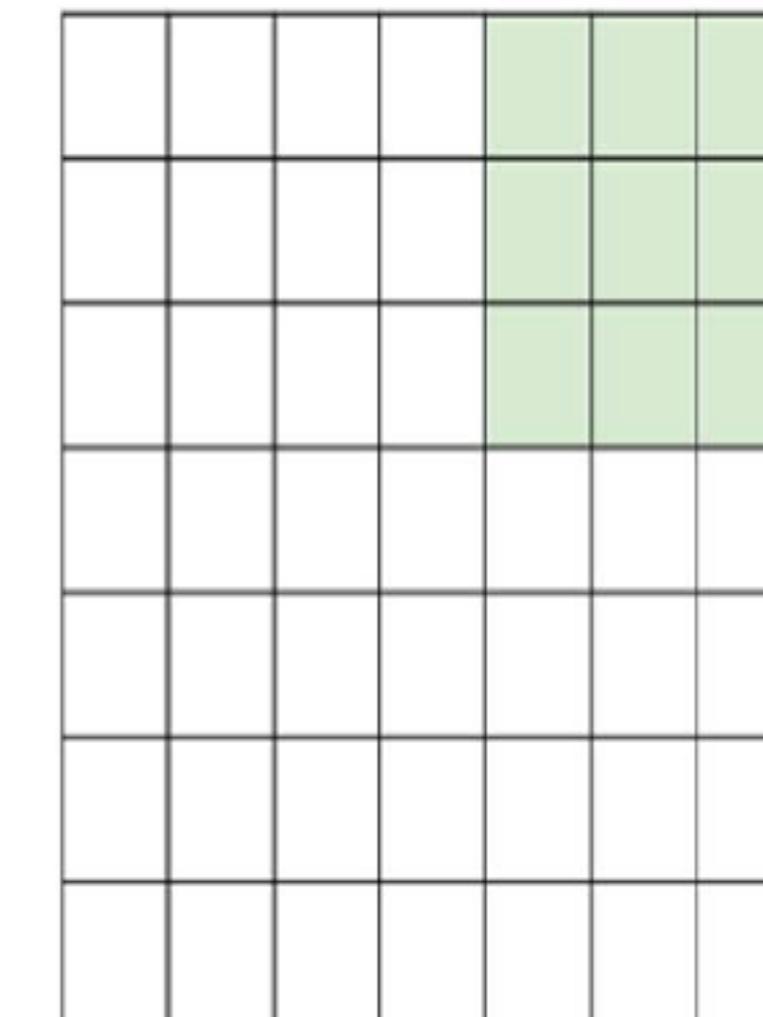
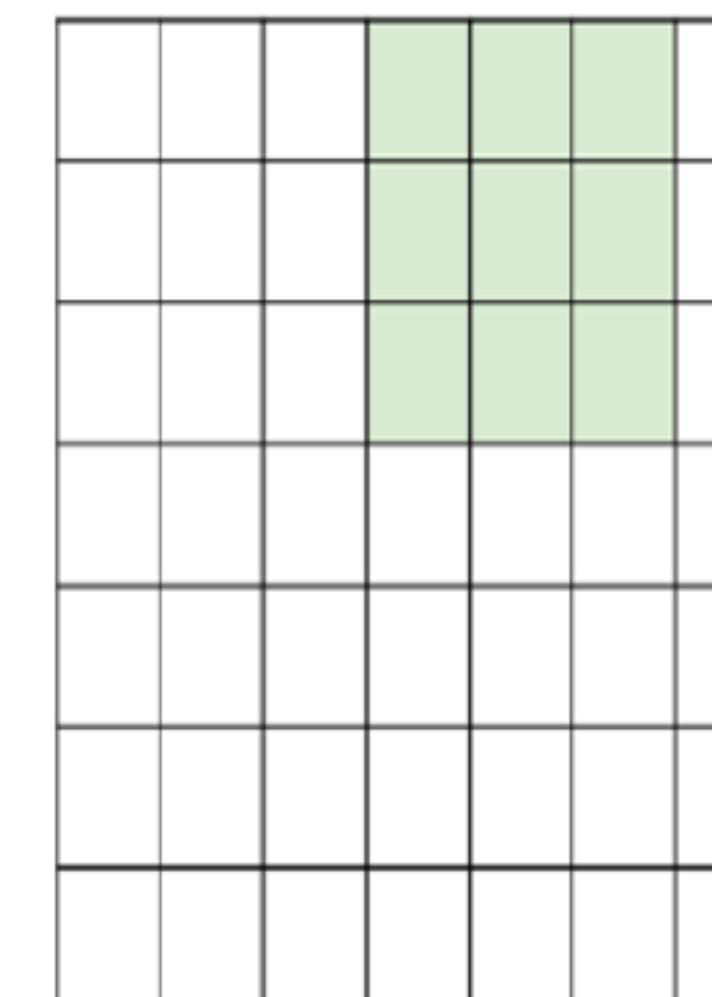
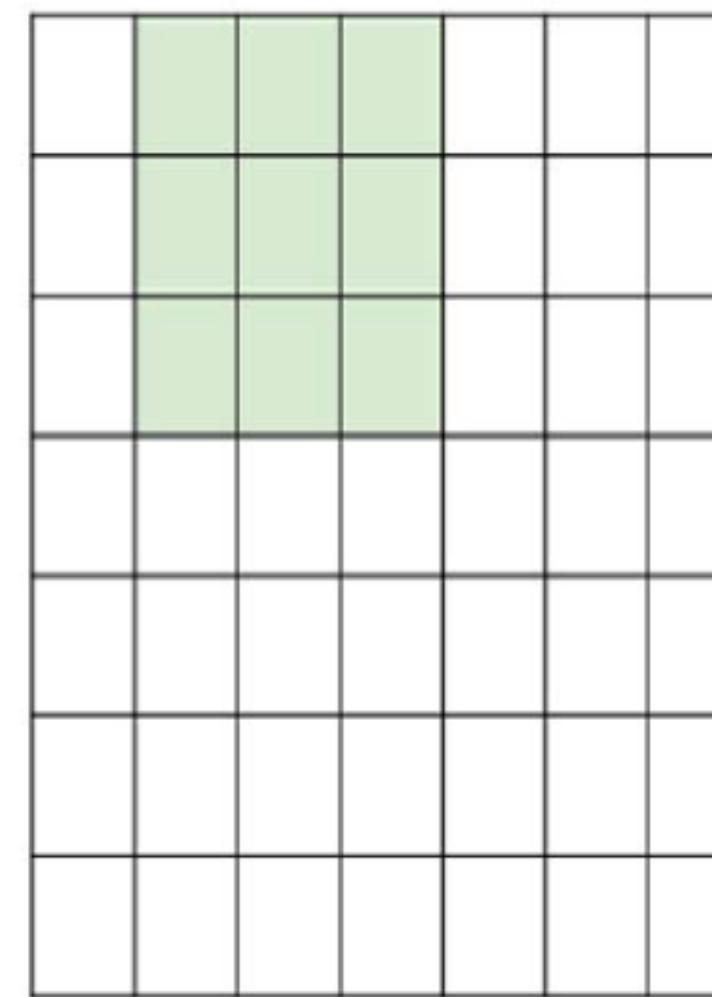
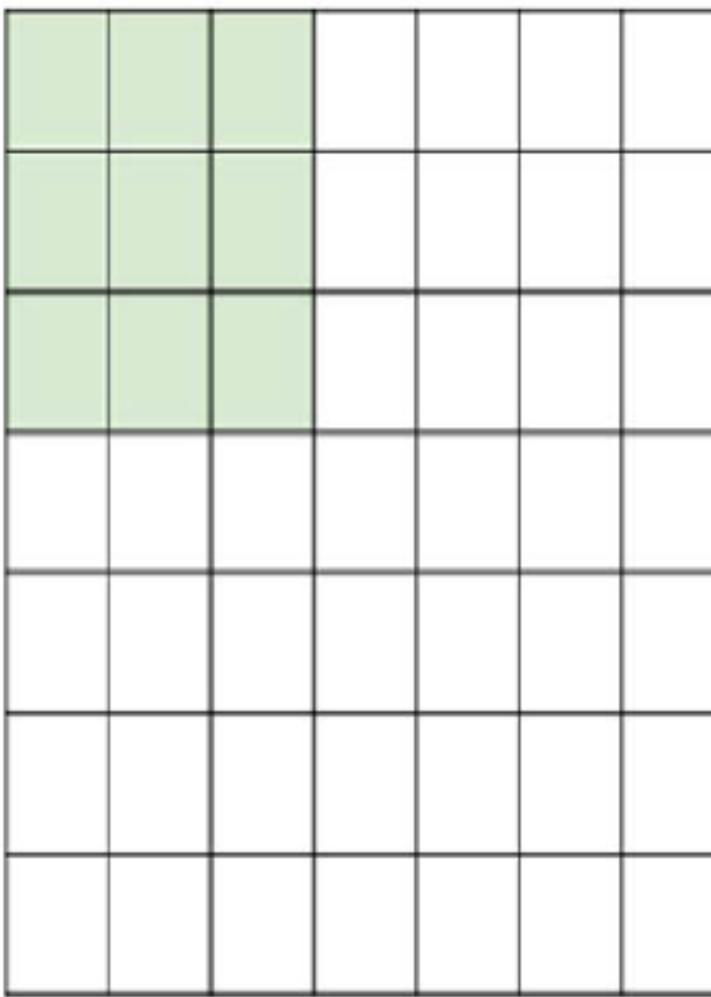


- Produce a map
- What's the size of the map?
 $5 * 5$

Convolution

Optional subtitle

- One local region only gives one output
- Convolution: Replicate the column of hidden units across space, with some stride



- 7 * 7 Input
- Assume 3*3 connectivity,
stride = 1

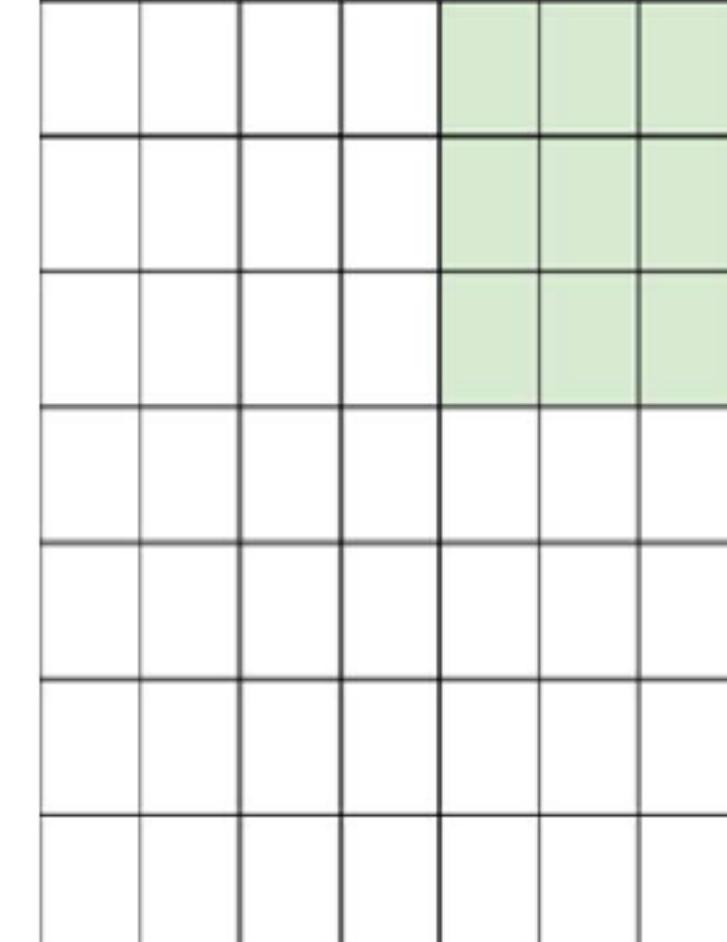
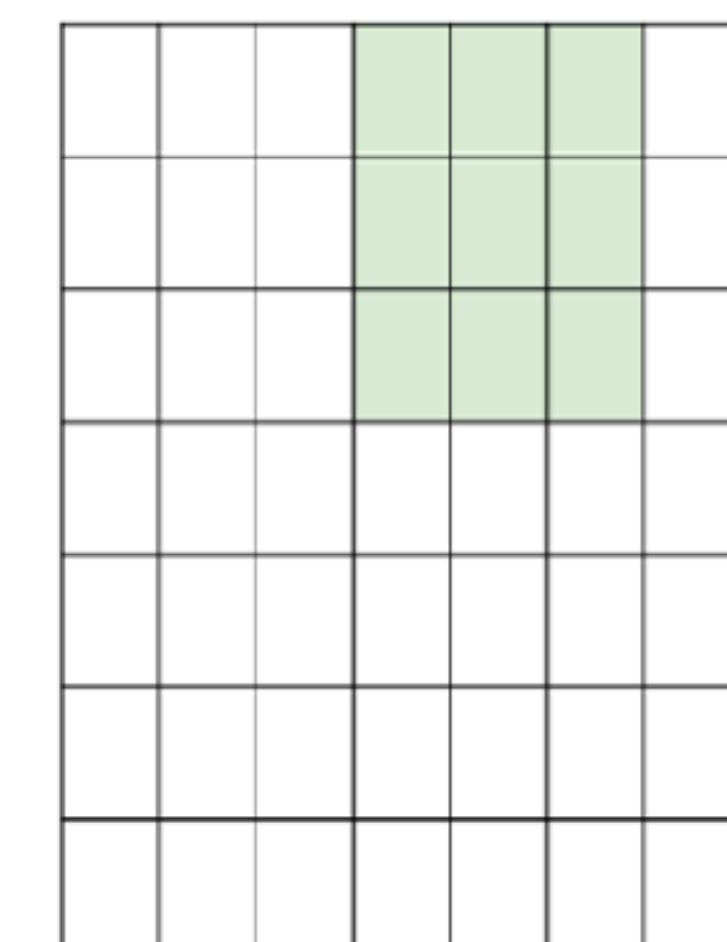
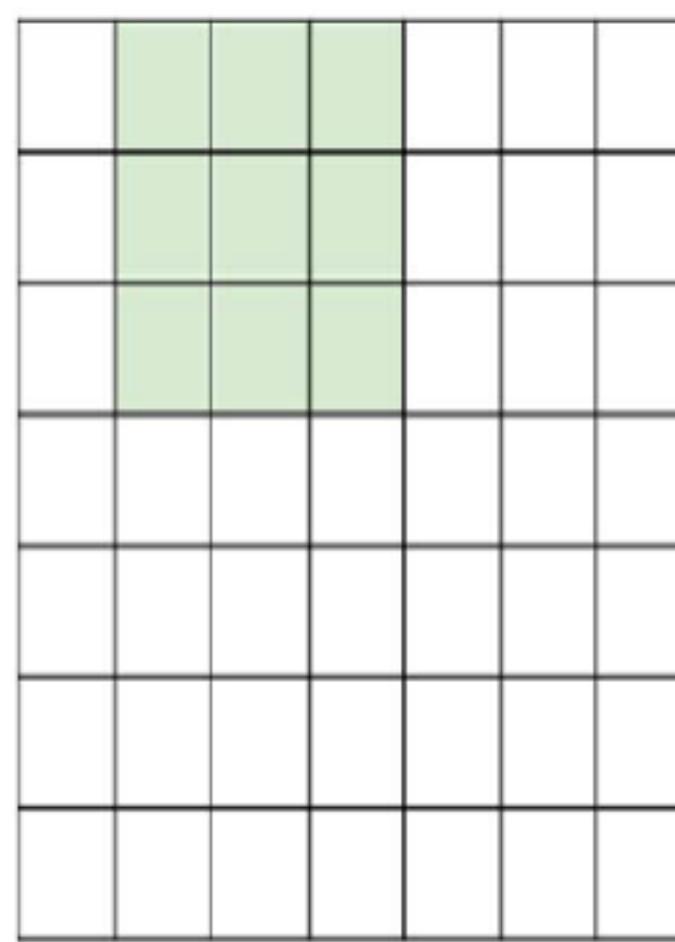
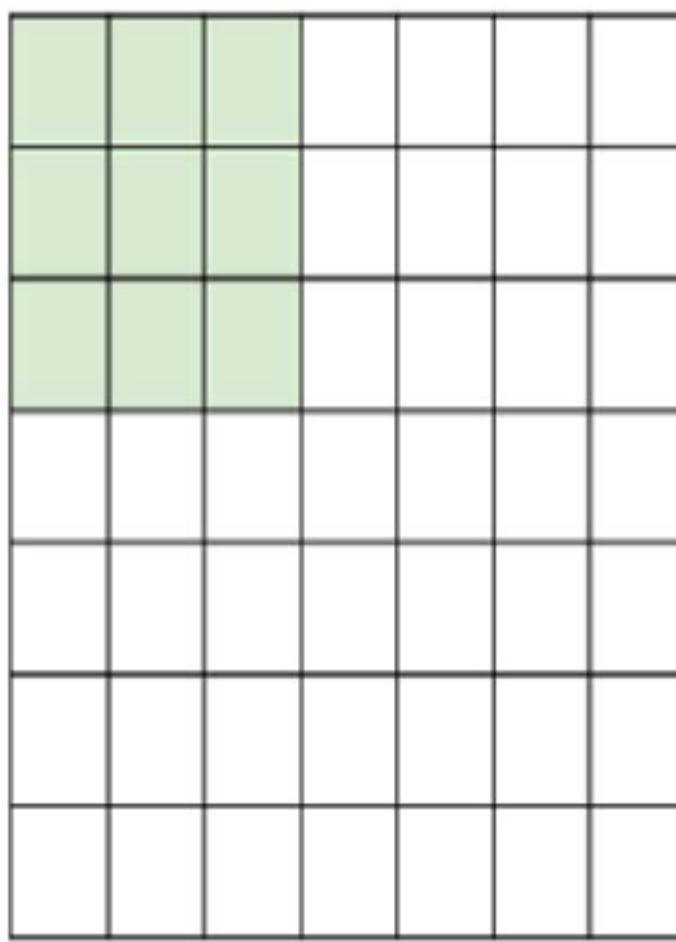


- What if stride = 2?

Convolution

Optional subtitle

- One local region only gives one output
- Convolution: Replicate the column of hidden units across space, with some stride



- $7 * 7$ Input
- Assume $3*3$ connectivity,
stride = 1



- What if stride = 3?

Convolution: In Practice

Optional subtitle

- Zero Padding
 - Input size: $7 * 7$
 - Filter Size: $3*3$, stride 1
 - Pad with 1 pixel border
- Output size?
 - $7 * 7 \Rightarrow$ preserved size!

0	0	0	0	0	0				
0									
0									
0									
0									

Convolution: Summary

- Zero Padding

- Input volume of size $[W1 * H1 * D1]$
- Using K units with receptive fields $F \times F$ and applying them at strides of S gives

Output volume: $[W2, H2, D2]$

- $W2 = (W1 - F)/S + 1$
- $H2 = (H1 - F) / S + 1$
- $D2 = k$



Convolution: Problem

Optional subtitle

- Assume input $[32 * 32 * 3]$
- 30 units with receptive field $5 * 5$, applied at stride 1/pad 1
=> Output volume: $[30 * 30 * 30]$

At each position of the output volume, we need $5 * 5 * 3$ weights

=> Number of weights in such layer: $27000 * 75 = 2$ million 😦

Idea:
Weight sharing!

Learn one unit, let the unit convolve across all local receptive fields!



Convolution: Problem

Optional subtitle

- Assume input $[32 * 32 * 3]$
- 30 units with receptive field $5 * 5$, applied at stride 1/pad 1
=> Output volume: $[30 * 30 * 30] = 27000$ units

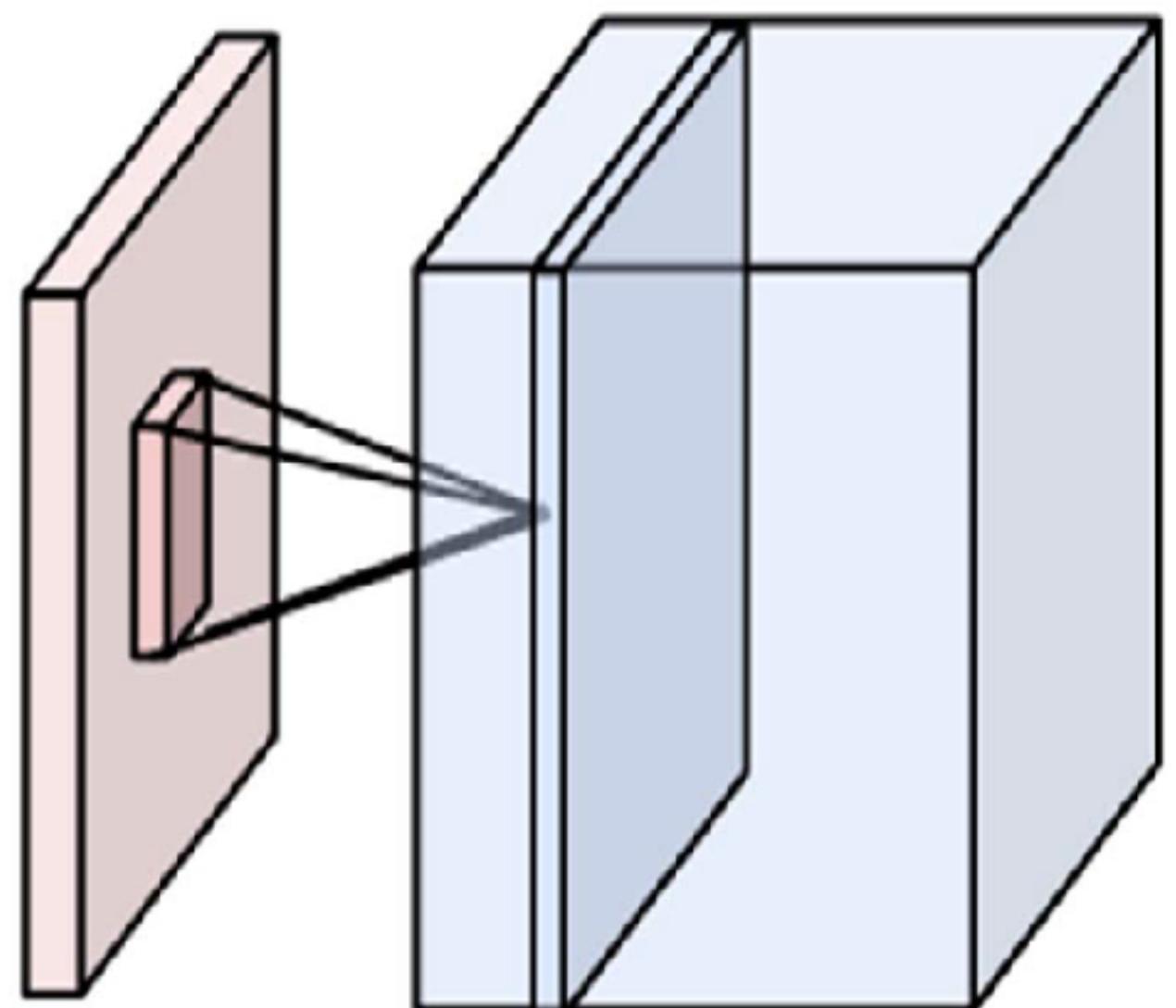
Weight sharing

- => Before: Number of weights in such layer: $27000 * 75 = 2$ million ☹
- => After: weight sharing: $30 * 75 = 2250$ ☺



Convolutional Layers

- Connect units only to local receptive fields
- Use the same unit weight parameters for units in each “depth slice” (i.e. across spatial positions)



one activation map (a depth slice),
computed with one set of weights

Can call the units “**filters**”

We call the layer convolutional because
it is related to convolution of two signals

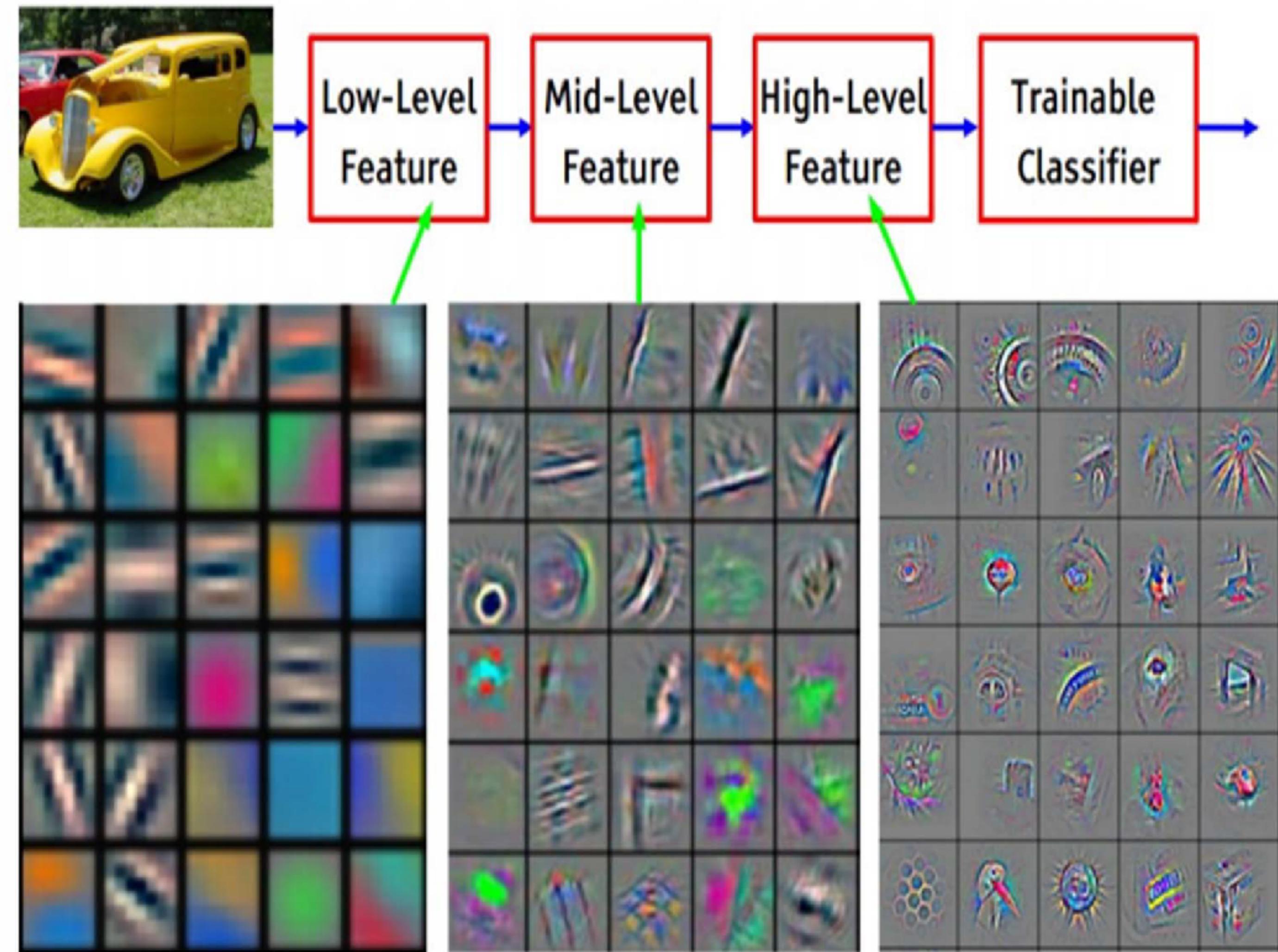
$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1, n_2] \cdot g[x-n_1, y-n_2]$$

Sometimes we also add a bias term b , $y = Wx + b$,
like what we have done for ordinary NN

**Short question: Will convolution layers
introduce nonlinearity?**

Stacking Convolutional Layers

Optional subtitle



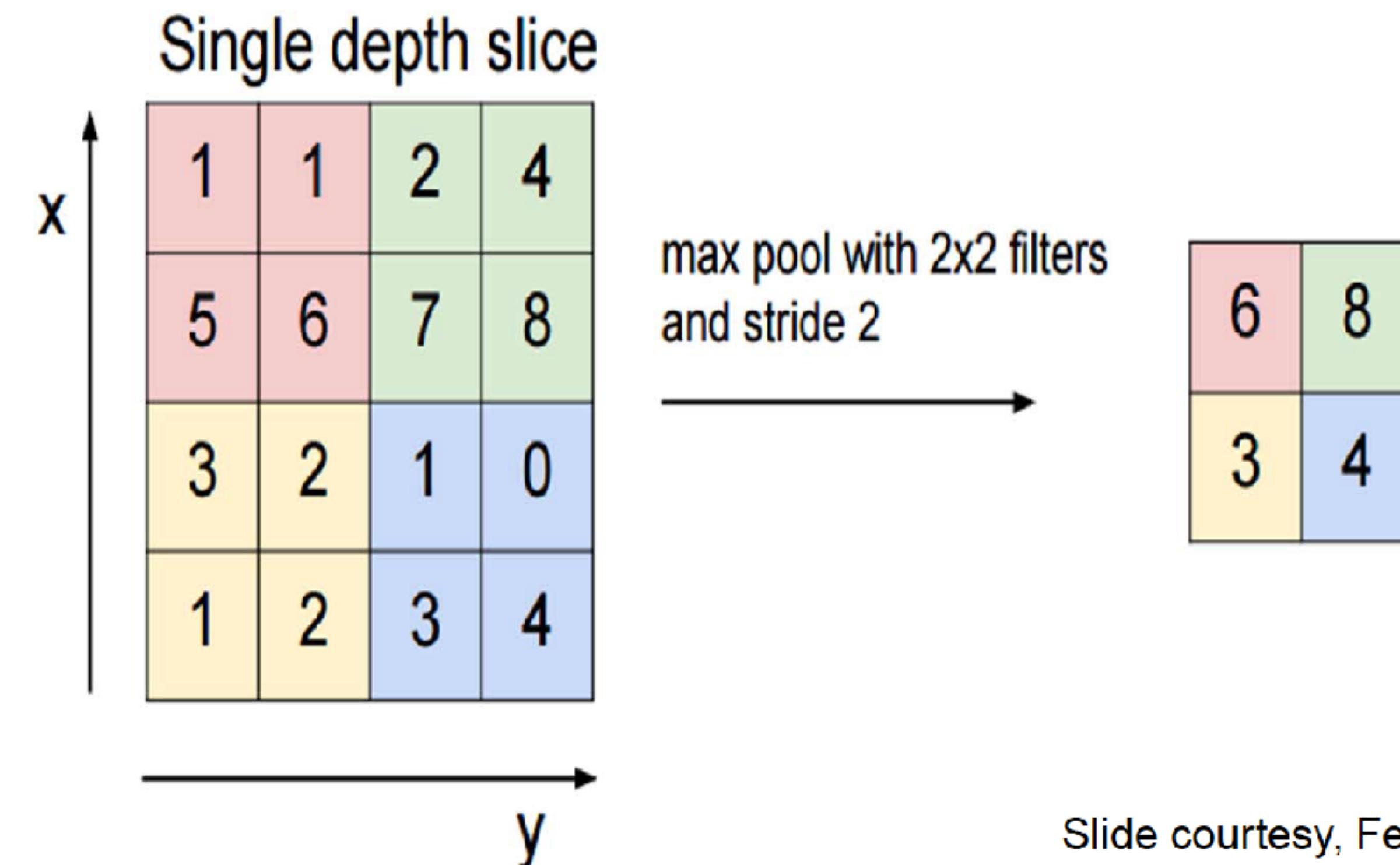
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]



Pooling Layers

Optional subtitle

- In ConvNet architectures, Conv layers are often followed by Pool layers
 - makes the representations smaller and more manageable without losing too much information. Computes MAX operation (most common)



Slide courtesy, Fei-Fei, Andrej Karpathy

Pooling Layers

Optional subtitle

- In ConvNet architectures, Conv layers are often followed by Pool layers
 - makes the representations smaller and more manageable without losing too much information. Computes MAX operation (most common)
- Input volume of size $[W_1 \times H_1 \times D_1]$
- Pooling unit receptive fields $F \times F$ and applying them at strides of S gives
- Output volume: $[W_2, H_2, D_1]$: depth unchanged!

$$W_2 = (W_1 - F)/S + 1,$$

$$H_2 = (H_1 - F)/S + 1$$

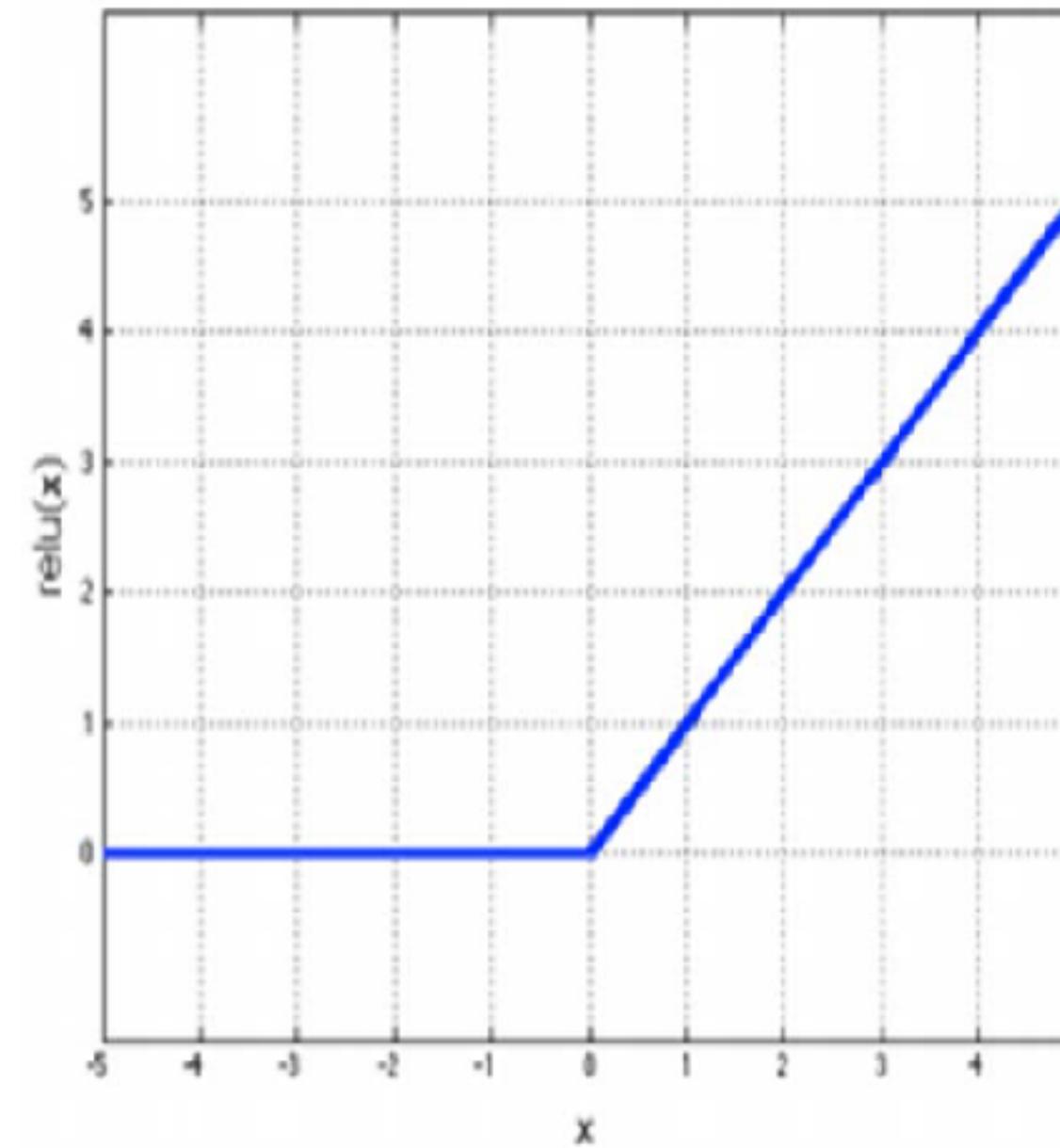
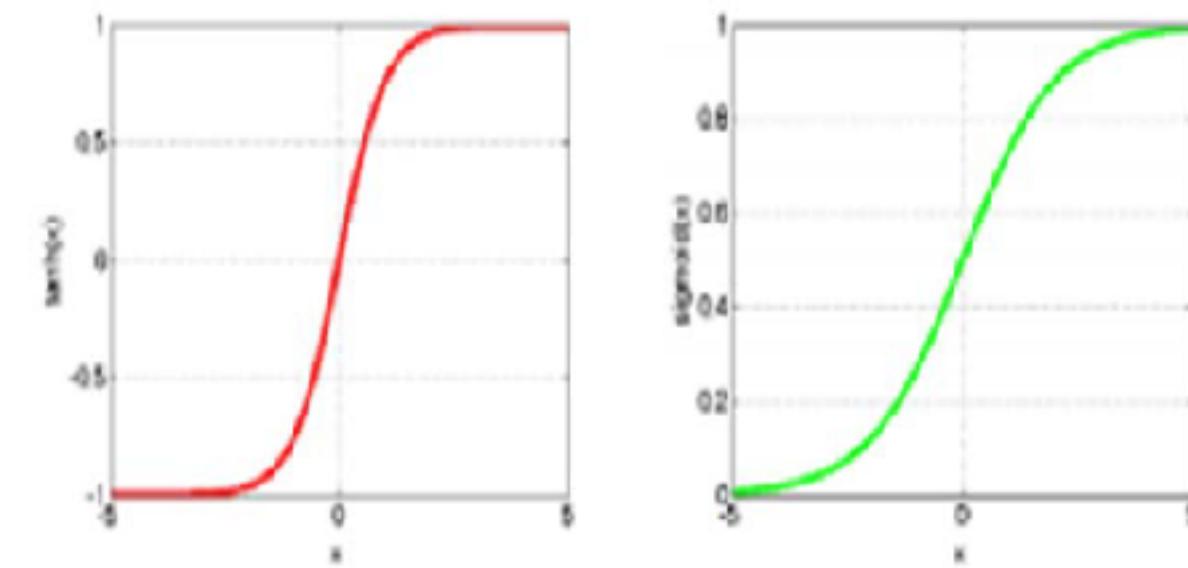
Short question: Will pooling layer introduce nonlinearity?



Nonlinearity

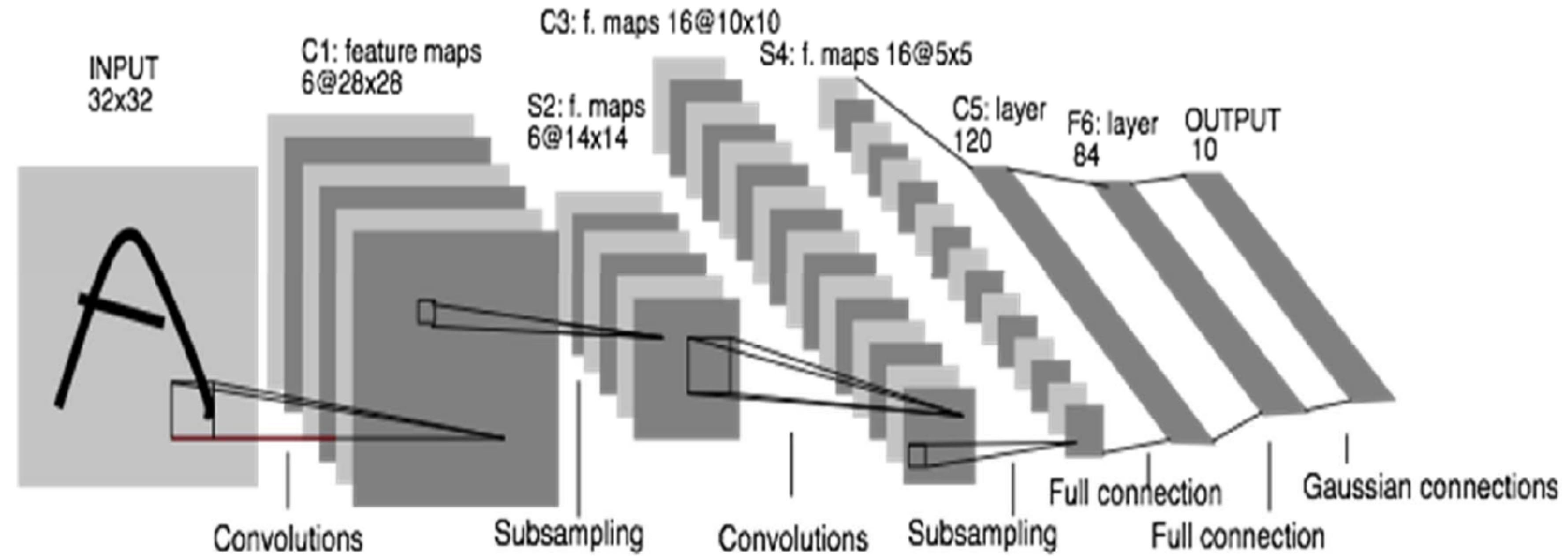
Optional subtitle

- Similar to NN, we need to introduce nonlinearity in CNN
 - Sigmoid
 - Tanh
 - RELU: Rectified Linear Units
 - Simplifies backpropagation
 - Makes learning faster
 - Avoids saturation issues



Convolutional Networks: 1989

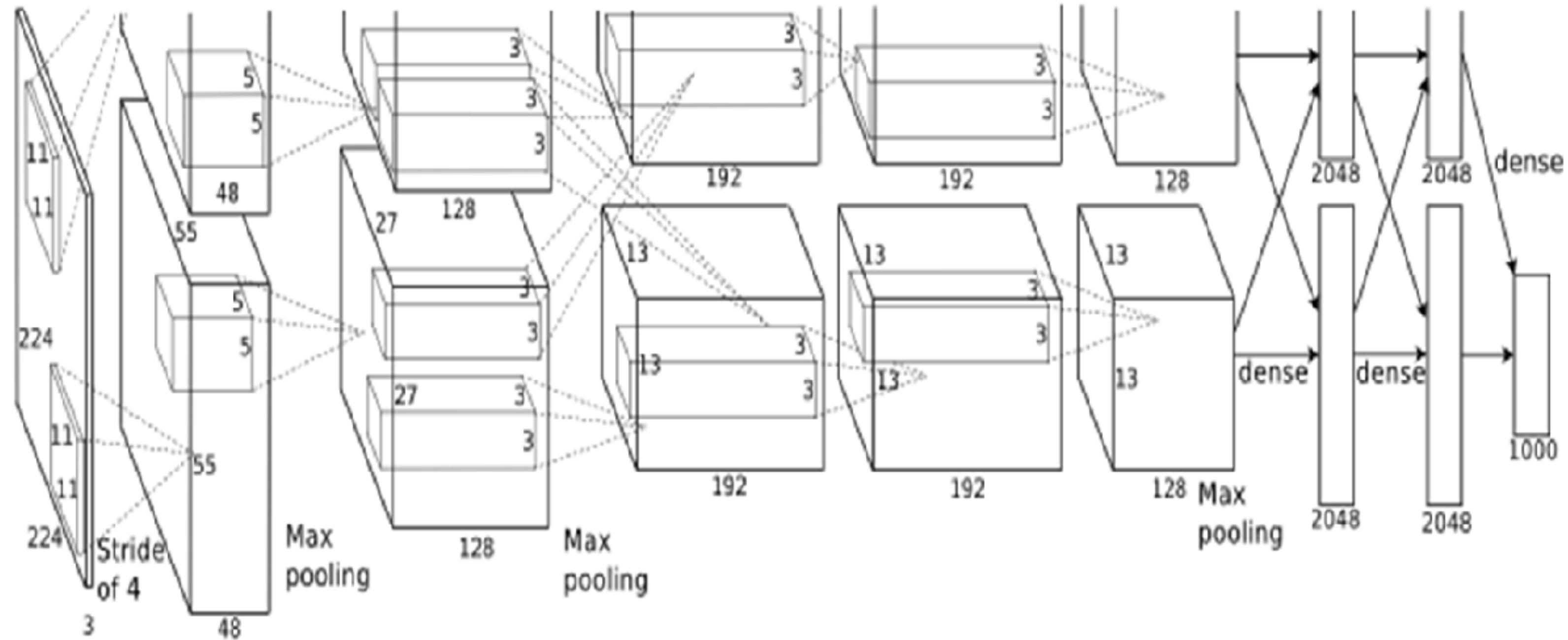
Optional subtitle



LeNet: a layered model composed of convolution and subsampling operations followed by a holistic representation and ultimately a classifier for handwritten digits.
[LeNet]

Convolutional Nets: 2012

Optional subtitle



AlexNet: a layered model composed of convolution, subsampling, and further operations followed by a holistic representation and all-in-all a landmark classifier on ILSVRC12. [AlexNet]

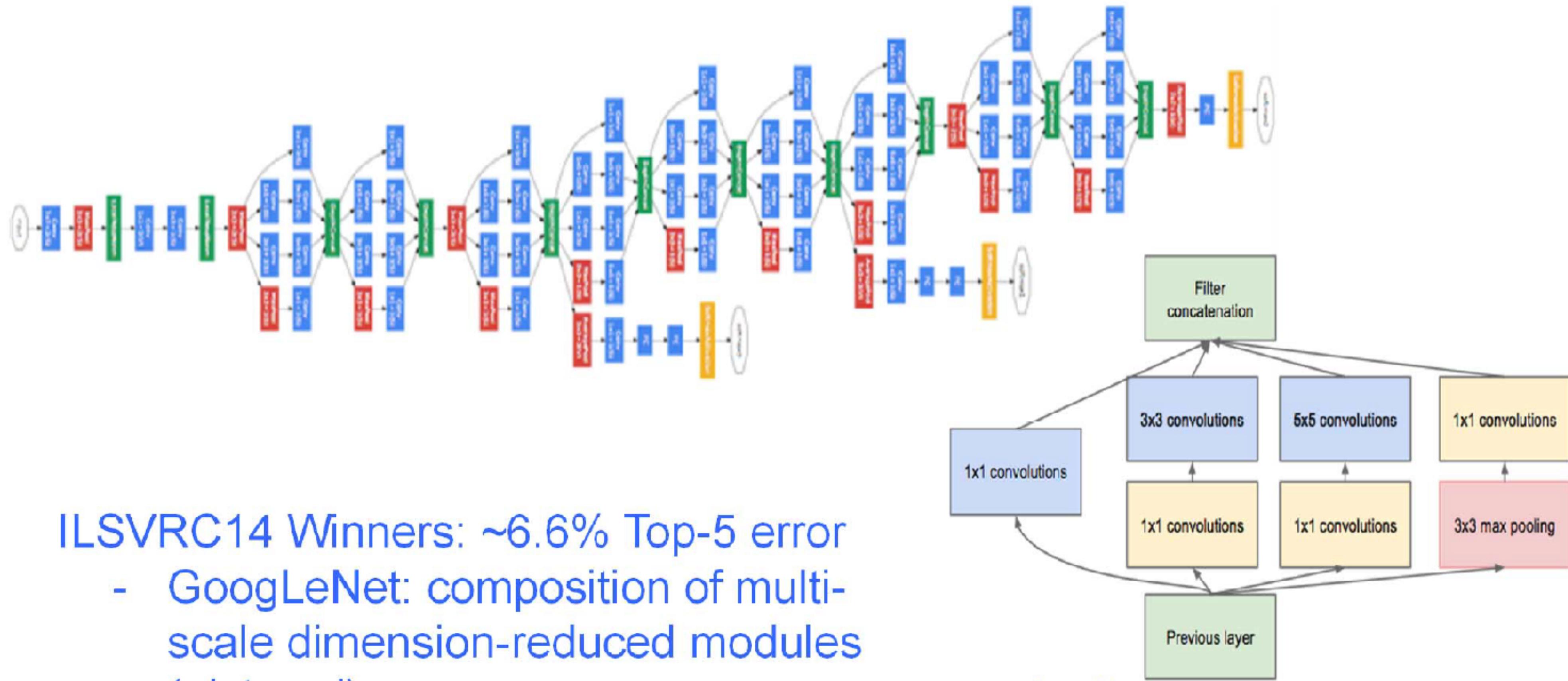
- + data
- + gpu
- + non-saturating nonlinearity
- + regularization

Slide courtesy Venagina lie



Convolutional Nets: 2014

Optional subtitle



ILSVRC14 Winners: ~6.6% Top-5 error

- GoogLeNet: composition of multi-scale dimension-reduced modules (pictured)
- VGG: 16 layers of 3x3 convolution interleaved with max pooling + 3 fully-connected layers

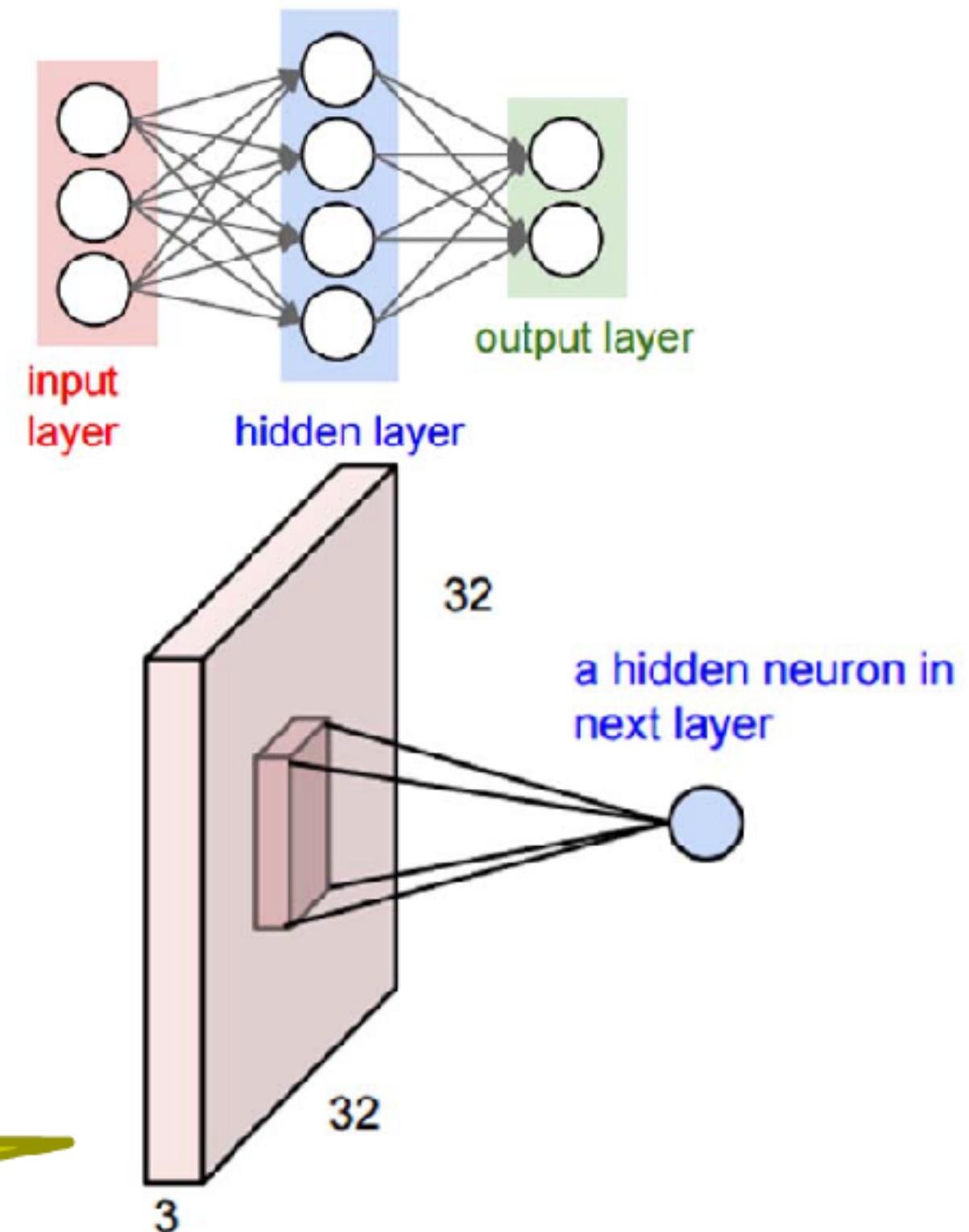
+ depth
+ data
+ dimensionality reduction

Slide courtesy: Yandan Li

Training CNN: Use GPU

- Convolutional layers
 - Reduce parameters BUT Increase computations
- FC layers
 - each neuron has more weights
 - but less computations
- Conv layers
 - each neuron has less weights
 - but more computations. Why?
because it will be used by many neurons

GPU is good at convolution!



Training CNN: depth cares!

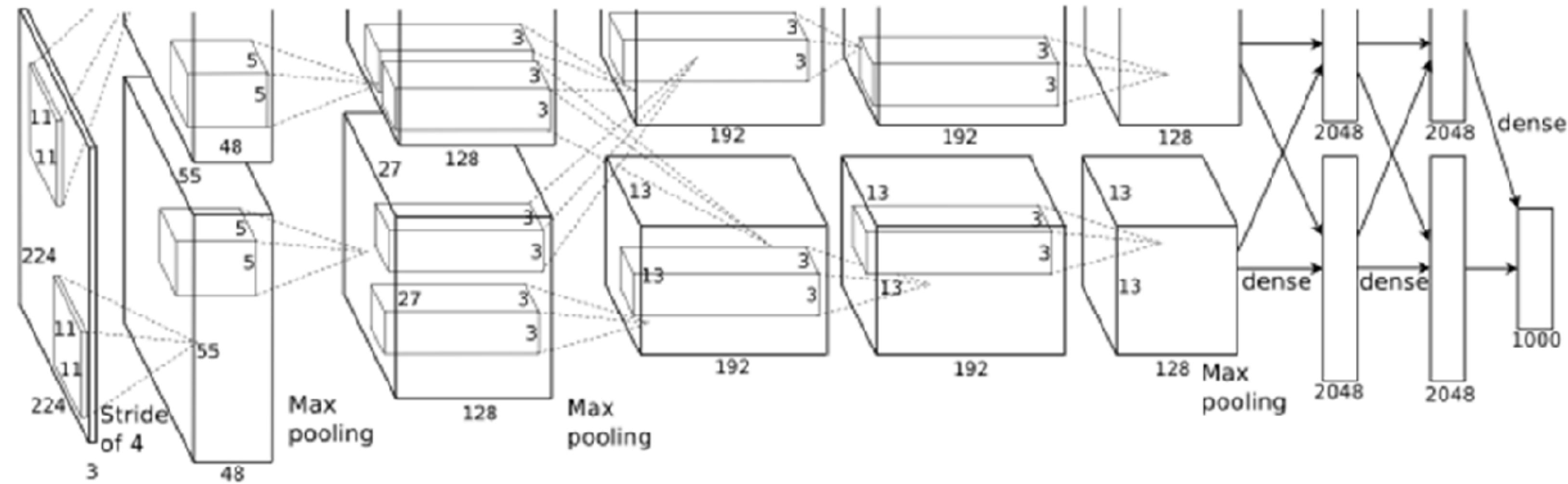
Optional subtitle



- 21 Layers!
- Gradient vanishes when the network is too deep: Lazy to learn!
- Add intermediate loss layers to produce error signals!
- Do contrast normalization after each conv layer!
- Use ReLU to avoid saturation!

Training CNN: Huge model needs more data!

Optional subtitle

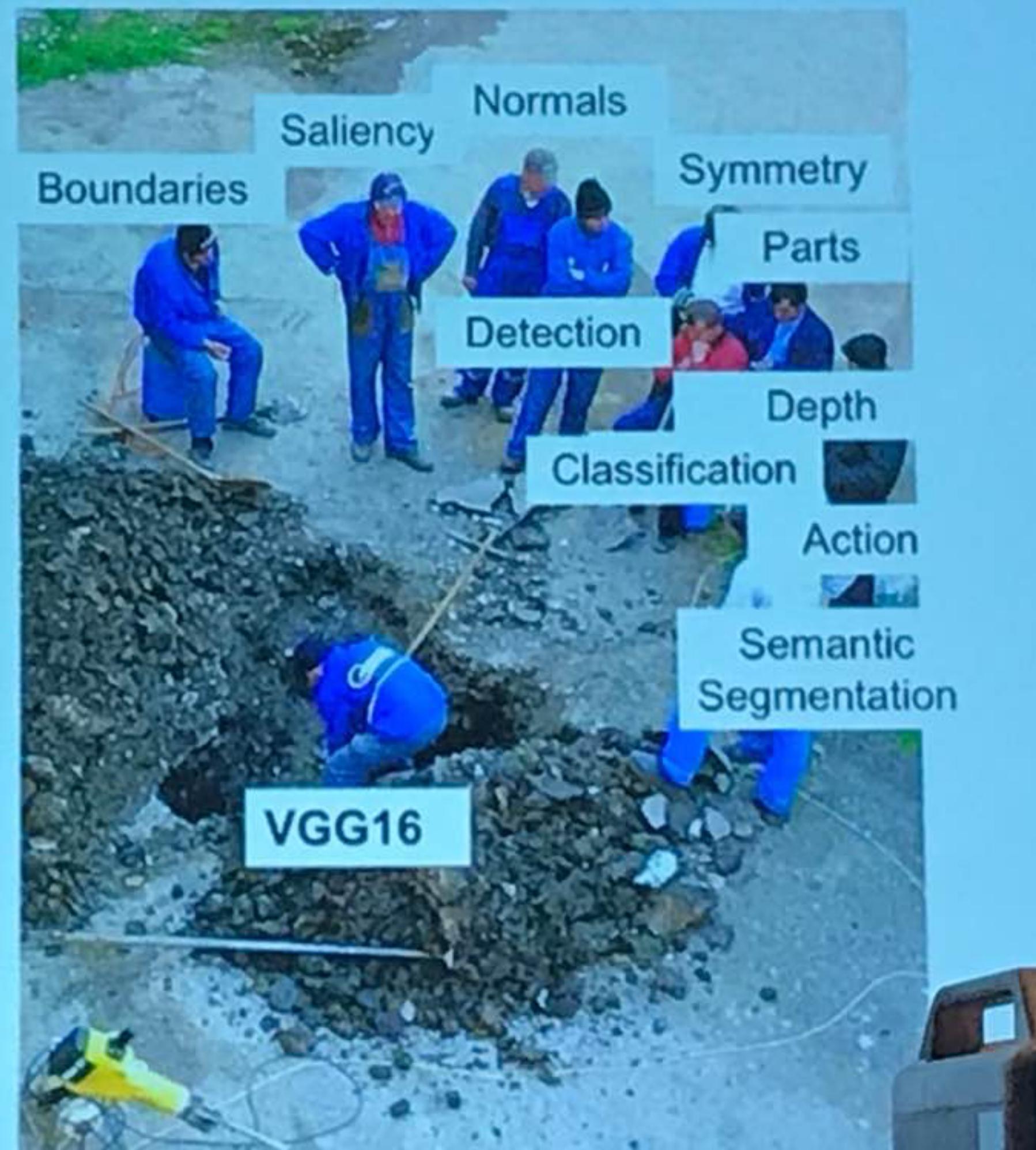


IMAGENET

- Only 7 layers, 60M parameters!
- Need more labeled data to train!
- Data augmentation: crop, translate, rotate, add noise!

A real-life analogy

Stronger workers
(ResNets)
More workers
Some coordination!



Training CNN: highly non-convex objective

Optional subtitle

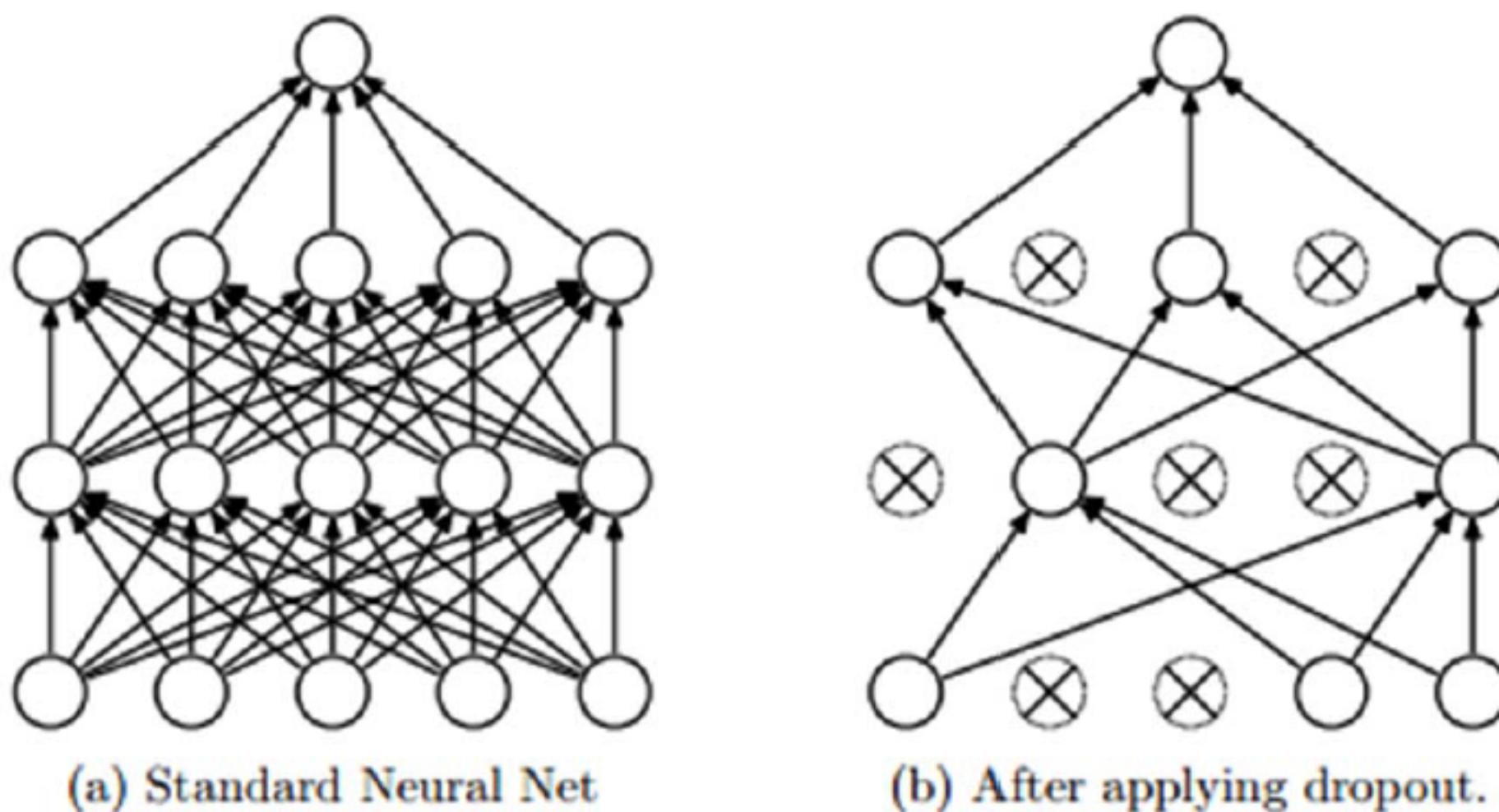
- Demand more advanced optimization techniques
 - Add momentum as we have done for NN
 - Learning rate policy
 - decrease learning rate regularly!
 - different layers use different learning rate!
 - observe the trend of objective curve more often!
 - Initialization really cares!
 - Supervised pretraining
 - Unsupervised pretraining



Training CNN: avoid overfitting

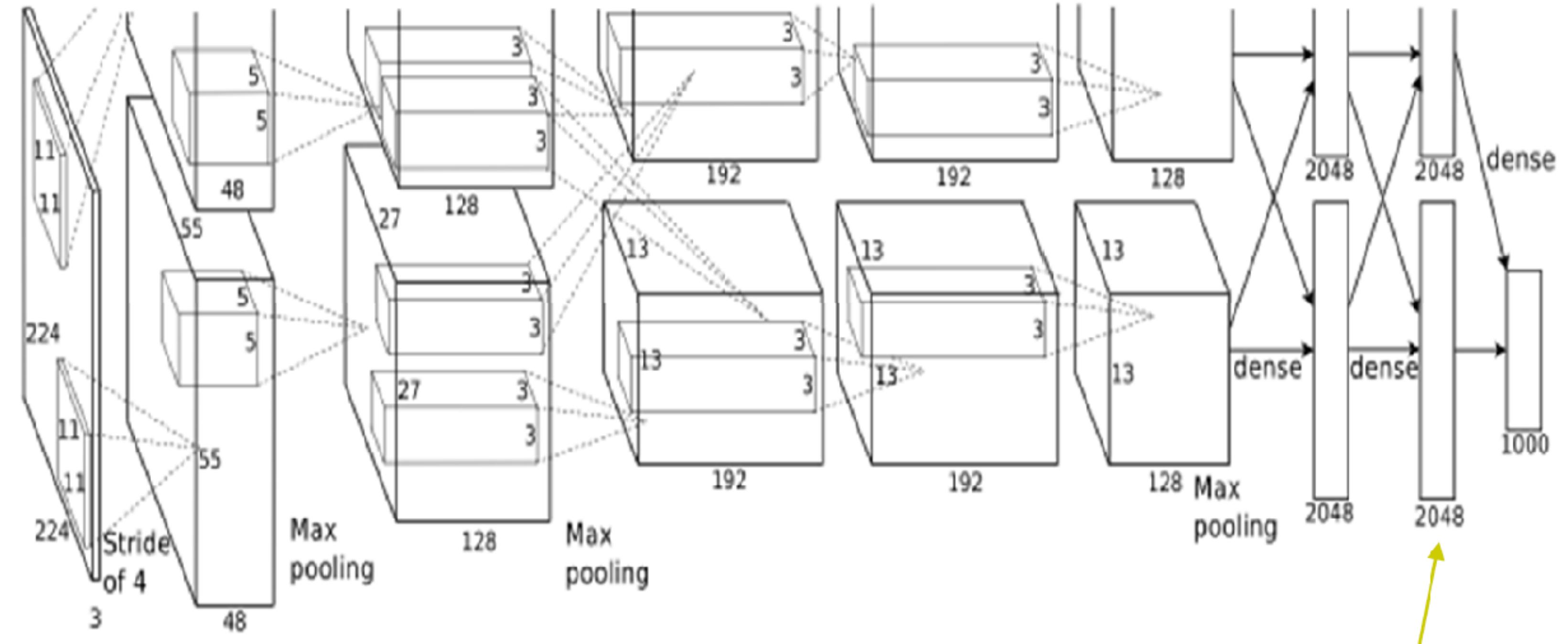
Optional subtitle

- More data are always the best way to avoid overfitting
 - data augmentation
- Add regularizations: recall what we have done for linear regression
- Dropout



Visualize and Understand CNN

Optional subtitle

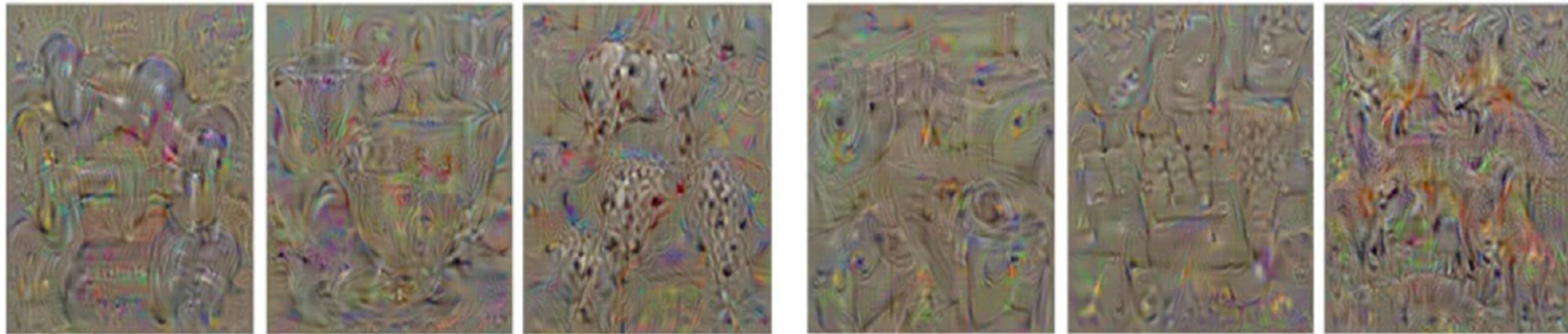


A CNN transforms the image to 4096 numbers that are then linearly classified.

Visualize and Understand CNN

Optional subtitle

- Find images that maximize some class score:



dumbbell

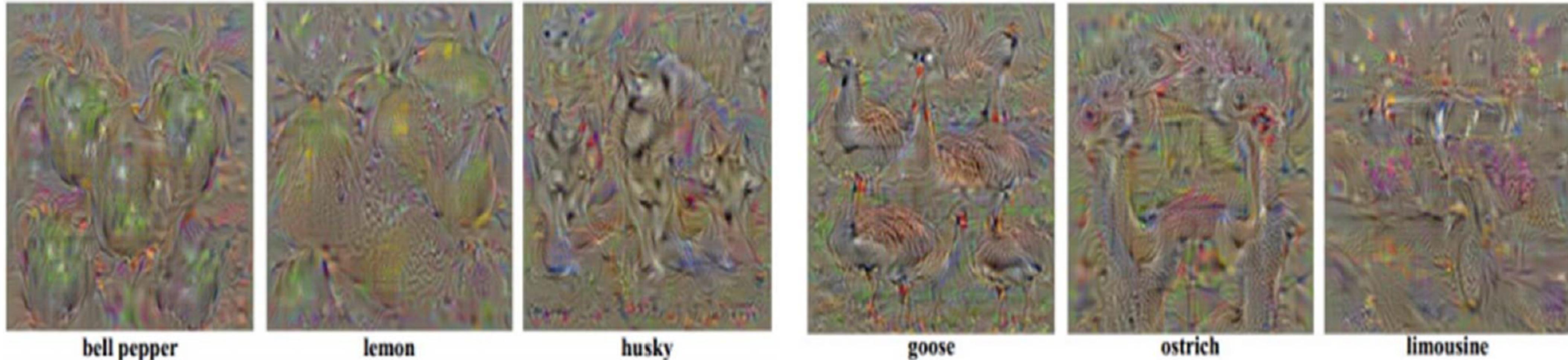
cup

dalmatian

washing machine

computer keyboard

kit fox



bell pepper

lemon

husky

goose

ostrich

limousine

Limitations

Optional subtitle

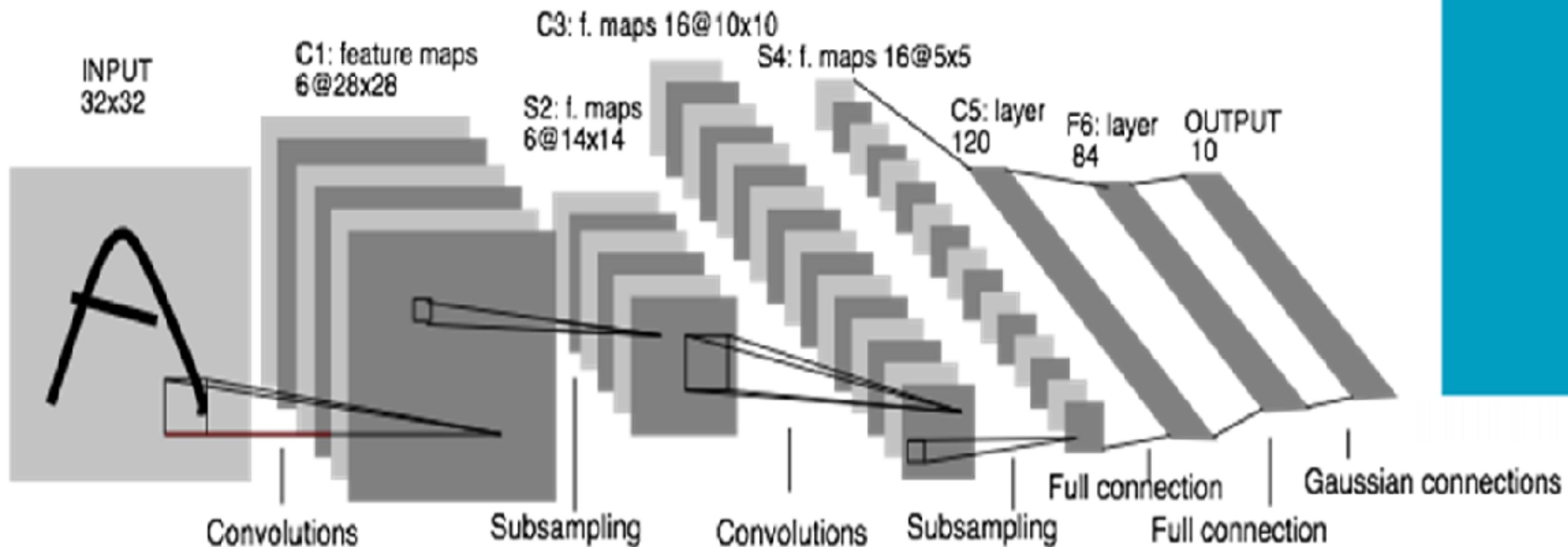
- Supervised Training
 - Need huge amount of labeled data, but label is scarce!
- Slow Training
 - Train an AlexNet on a single machine need one week!
- Optimization
 - Highly nonconvex objective
- Parameter tuning is hard
 - The parameter space is so large...



Summary

Optional subtitle

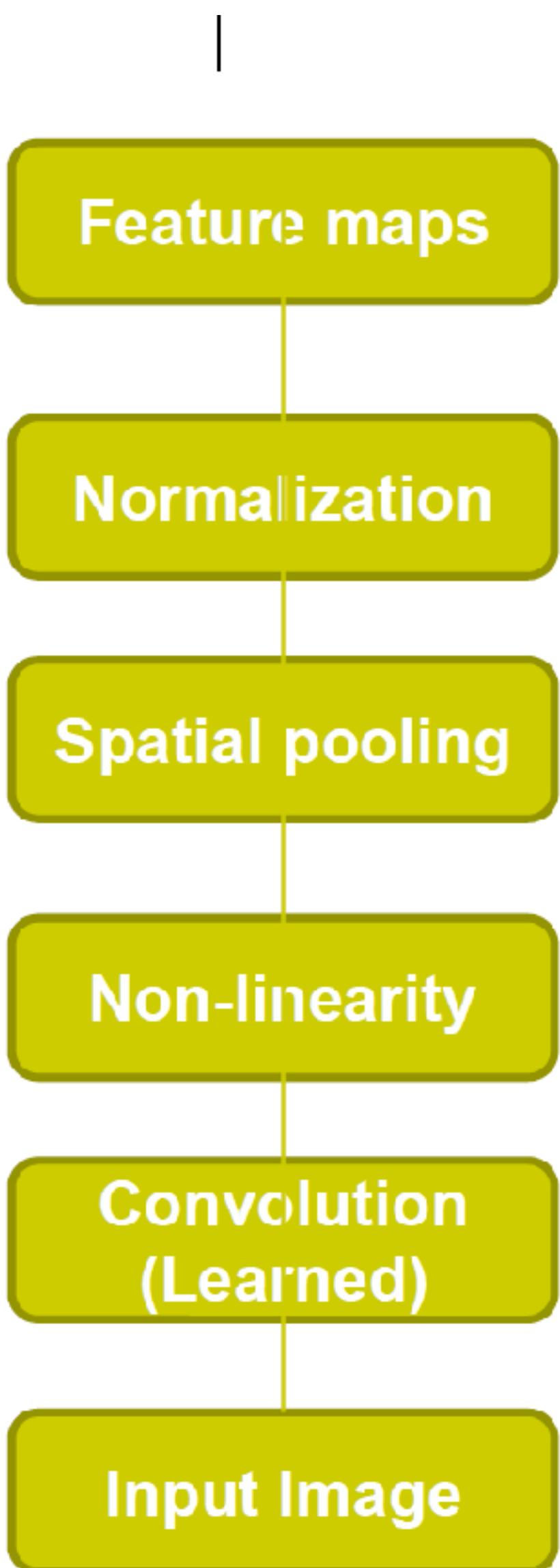
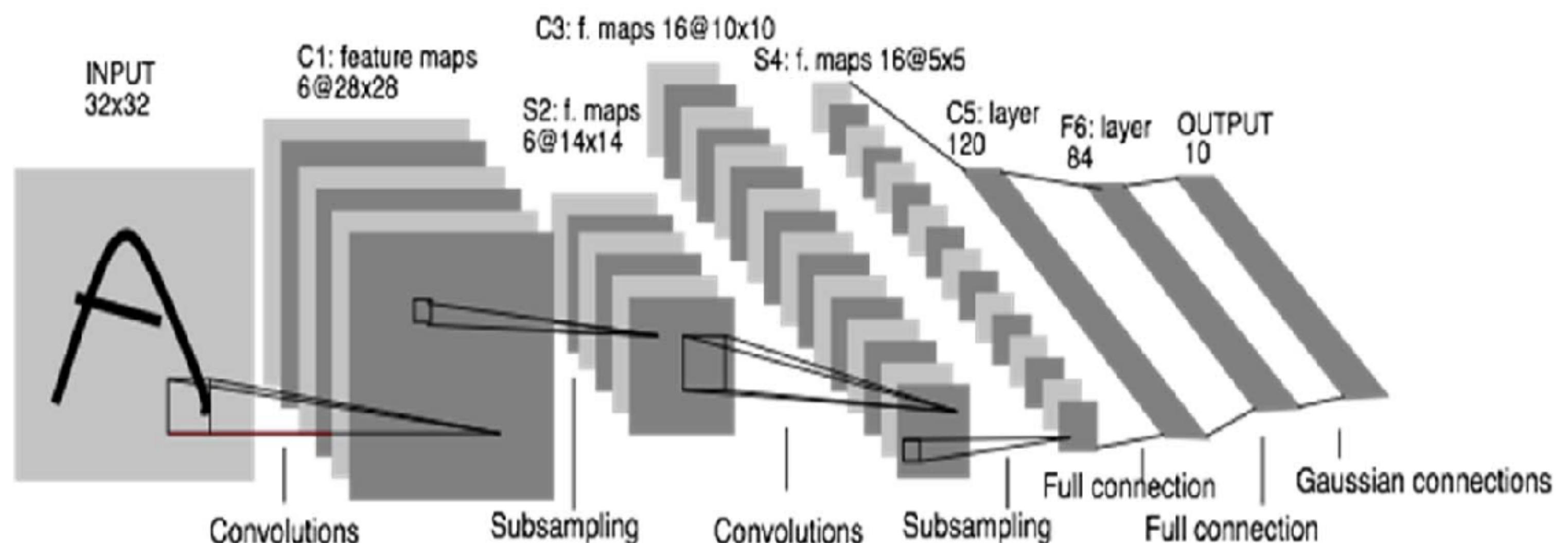
- Neural network with specialized connectivity structure
- Stack multiple stages of feature extractors
- Higher stages compute more global, more invariant features
- Classification layer at the end



Summary

Optional subtitle

- Feed-forward
 - Convolve input
 - Non-linearity (rectified linear)
 - Pooling (local max, mean)
- Supervised
- Train convolutional filters by back-propagation classification error at the end



Further reading

Optional subtitle

<http://www.robots.ox.ac.uk/~vgg/practicals/cnn/>
<http://cs231n.github.io/convolutional-networks/>

- Andrej Karpathy: The Unreasonable Effectiveness of Recurrent Neural Networks
 - (<http://karpathy.github.io/2015/05/21/rnn-effectiveness>)
- Recurrent Neural Networks Tutorial
 - (<http://www.wildml.com/2015/09/recurrent-neural-networkstutorial-part-1-introduction-to-rnns>)

[http://ufldl.stanford.edu/wiki/index.php/UFLDL 教程](http://ufldl.stanford.edu/wiki/index.php/UFLDL_教程)



Appendix



AI的产生与发展

1. 孕育期（1943年—1955年）

1943年Warren McCulloch 和Walter Pitts:人工神经元模型(MP model)

1946年美国人毛奇莱(Mauchly)和艾克特(Eckert):世界上第一台电子计算机ENIAC

1945年冯诺依曼(John Von Neumann): 冯诺依曼结构。

1948年维纳(N.Wiener) :创立了控制论。控制论向人工智能的渗透，形成了行为主义学派。

1951年Marvin Minsky 和Dean Edmonds:神经元网络计算机

1950年Alan Turing: Computer Machinery and Intelligence



AI的产生与发展

2. AI的诞生（1956年）

AI诞生于一次历史性的聚会

时间：1956年夏季

地点：达特莫斯 (Dartmouth) 大学

目的：为使计算机变得更“聪明”，或者说使计算机具有智能

发起人：

麦卡锡(J.McCarthy) ， Dartmouth的年轻数学家、计算机专家，后为MIT教授

明斯基(M.L.Minsky) ， 哈佛大学数学家、神经学家，后为MIT教授

洛切斯特(N.Lochester), IBM公司信息中心负责人

香农(C.E.Shannon)，贝尔实验室信息部数学研究员

参加人：

莫尔(T.more)、塞缪尔(A.L.Samuel), IBM公司

塞尔夫里奇(O.Selfridge)、索罗蒙夫(R.Solomonff) , MIT

纽厄尔(A.Newell), 兰德(RAND)公司

西蒙(H.A.Simon), 卡内基(Carnegie)工科大学

会议结果：

由麦卡锡提议正式采用了“Artificial Intelligence”这一术语



人工智能三大学派

1956年，美国的达特茅斯历史意义的会议，标志着人工智能的正式诞生。
McCarthy——“人工智能之父”

符号主义（以数理逻辑和知识为核心）
联结主义（如人工神经网络）
行为主义（控制论等）



Hi Serge,

We decided to withdraw our paper # [ID no.] from CVPR "[Paper Title]" by [Author Name] et al.
We posted it on ArXiv: <http://arxiv.org/> [Paper ID] .

We are withdrawing it for three reasons: 1) the scores are so low, and the reviews so ridiculous, that I don't know how to begin writing a rebuttal without insulting the reviewers;
2) we prefer to submit the paper to ICML where it might be better received;
3) with all the fuss I made, leaving the paper in would have looked like I might have tried to bully the program committee into giving it special treatment.

Getting papers about feature learning accepted at vision conference has always been a struggle, and I've had more than my share of bad reviews over the years. Thankfully, quite a few of my papers were rescued by area chairs.

This time though, the reviewers were particularly clueless, or negatively biased, or both. I was very sure that this paper was going to get good reviews because: 1) it has two simple and generally applicable ideas for segmentation ("purity tree" and "optimal cover"); 2) it uses no hand-crafted features (it's all learned all the way through. Incredibly, this was seen as a negative point by the reviewers!); 3) it beats all published results on 3 standard datasets for scene parsing; 4) it's an order of magnitude faster than the competing methods.

If that is not enough to get good reviews, I just don't know what is.

So, I'm giving up on submitting to computer vision conferences altogether. CV reviewers are just too likely to be clueless or hostile towards our brand of methods. Submitting our papers is just a waste of everyone's time (and incredibly demoralizing to my lab members)

I might come back in a few years, if at least two things change:

- Enough people in CV become interested in feature learning that the probability of getting a non-clueless and non-hostile reviewer is more than 50% (hopefully [Computer Vision Researcher]'s tutorial on the topic at CVPR will have some positive effect).
- CV conference proceedings become open access.

We intent to resubmit the paper to ICML, where we hope that it will fall in the hands of more informed and less negatively biased reviewers (not that ML reviewers are generally more informed or less biased, but they are just more informed about our kind of stuff). Regardless, I actually have a keynote talk at [Machine Learning Conference], where I'll be talking about the results in this paper.

Be assured that I am not blaming any of this on you as the CVPR program chair. I know you are doing your best within the traditional framework of CVPR.

I may also submit again to CV conferences if the reviewing process is fundamentally reformed so that papers are published before they get reviewed.

You are welcome to forward this message to whoever you want.

I hope to see you at NIPS or ICML.

Cheers,

- [Author]

Editor's note: the following is an anonymized letter from a Machine Learning researcher who decided to withdraw his submission from CVPR 2012. The submission received ratings of "Definitely Reject," "Borderline" and "Weakly Reject." The letter and the paper reviews are posted here with his

Question	
Please briefly describe the paper's contributions, and list its positive and negative points.	This paper presents a method for scene parsing
Overall Rating	Definitely Reject
Please explain your rating. If the paper is so unclear that it should be rejected, please explain that. If the paper is not novel please explain, citing the work that makes it so. You should take into account whether this paper is of wide	The key contribution of the paper is the method for purity
Additional comments to author(s)	The authors may want to consider trying Kolmogorov's

Question	
Please briefly describe the paper's contributions, and list its positive and negative points.	The paper addresses the problem of semantic segmentation, where every pixel in an image is assigned
Overall Rating	Borderline
Please explain your rating. If the paper is so unclear that it should be rejected, please explain that. If the paper is not novel please explain, citing the work that makes it so. You should take into account whether this paper is of wide	One of the strongest points of the paper is that it presents an interesting framework for the scene



Acknowledgement

Some slides come from the course of “deep learning”, Prof. Xiaoyang Wang, and Prof. Eric Xing.

