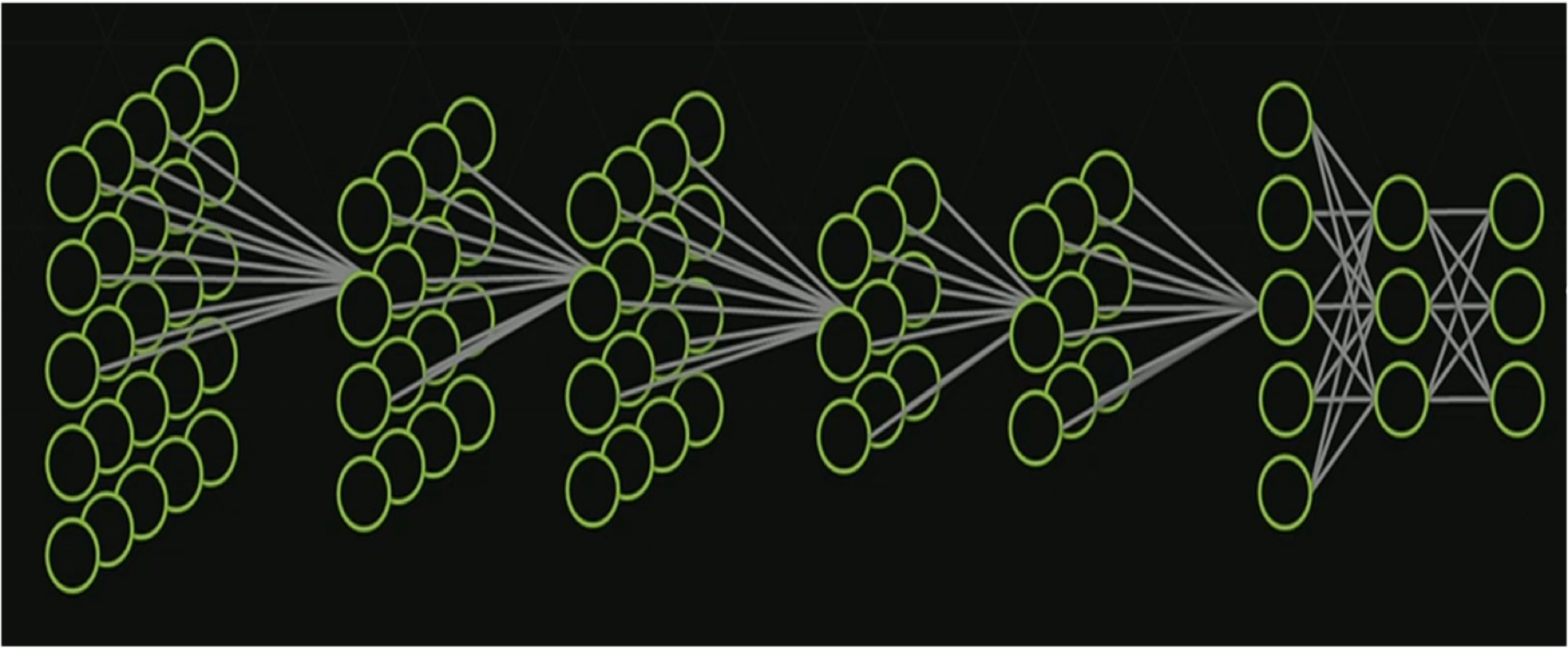


Introduction to Deep Learning

Zhanxing Zhu
zhanxing.zhu@pku.edu.cn
Peking University

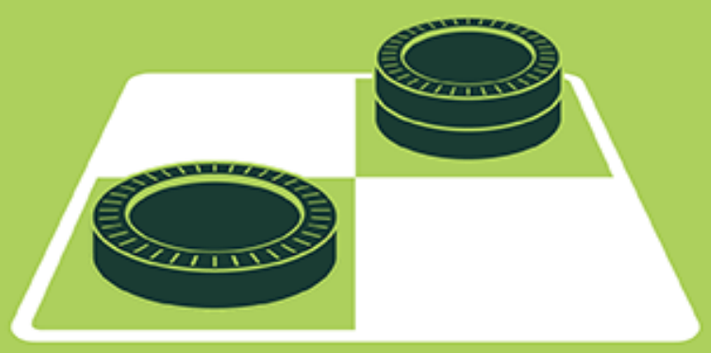
AI HAS EXPLODED SINCE 2015

Contents

- Brief introduction to deep learning
- Multi-Layer Perceptron
- Training neural nets
- Convolutional Neural Nets, Recurrent Neural Nets, and Autoencoder
- Generative modeling with deep learning

ARTIFICIAL INTELLIGENCE

Early artificial intelligence stirs excitement.



MACHINE LEARNING

Machine learning begins to flourish.

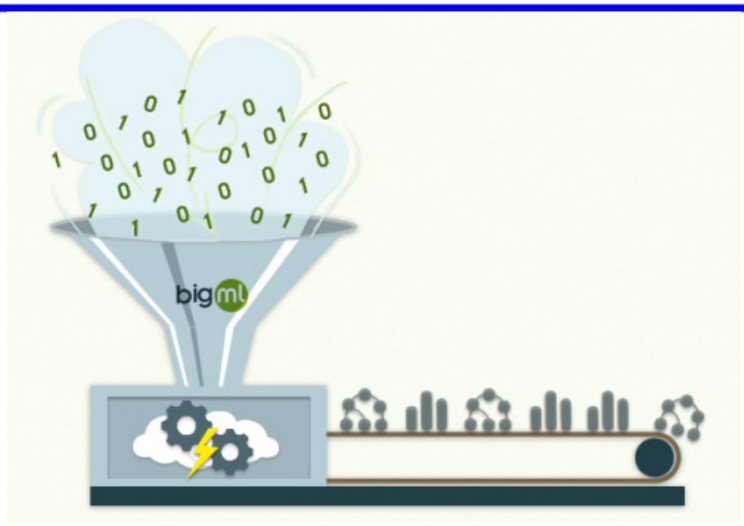
DEEP LEARNING

Deep learning breakthroughs drive AI boom.

1950's 1960's 1970's 1980's 1990's 2000's 2010's

Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

Machine Learning

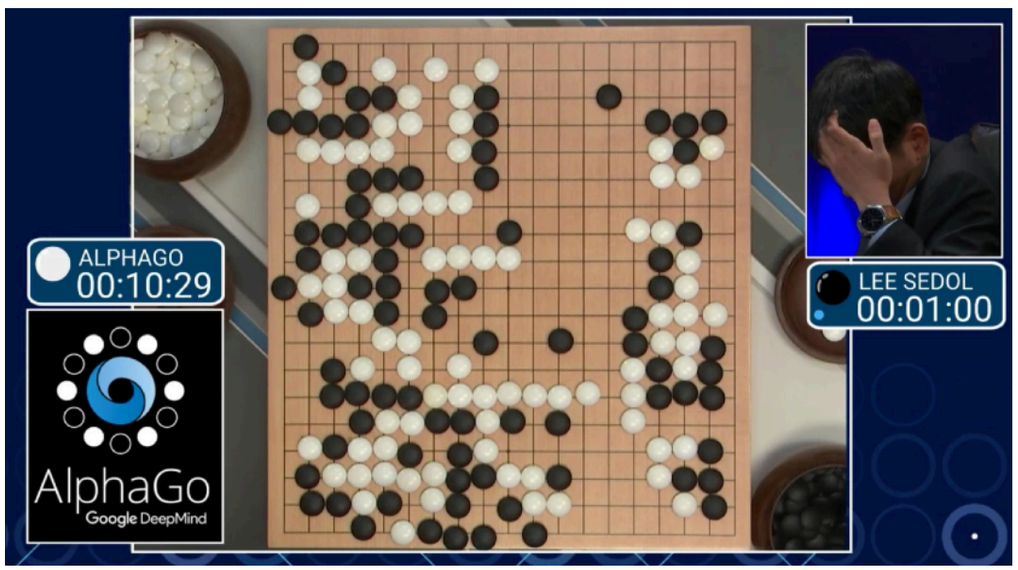


- Models learn from data
- To make predictions or learn patterns
- Models are ``trained'' by data and algorithms to obtain ability to perform the **specific** tasks
- Applications: computer vision, natural language processing, robot, financial markets, healthcare, etc.

Output = Model (input; parameters)

Deep learning

- A key solution to AI
  - Intuitive tasks for human
    - vision, natural language processing, speech, Go play, driving…
  - Some breakthroughs
    - AlexNet, ResNet (2016) for image recognition (surpass human performance) on ImageNet.
    - AlphaGo(DeepMind)
    - Health/Medical
      - Electrical Heath Records Deep Patient ***Nature*** 2016
      - Skin cancer



Andre Esteva et.al *Nature* 2017

Some interesting stuffs…

- Neural artist style (CNN, App: Prisma)

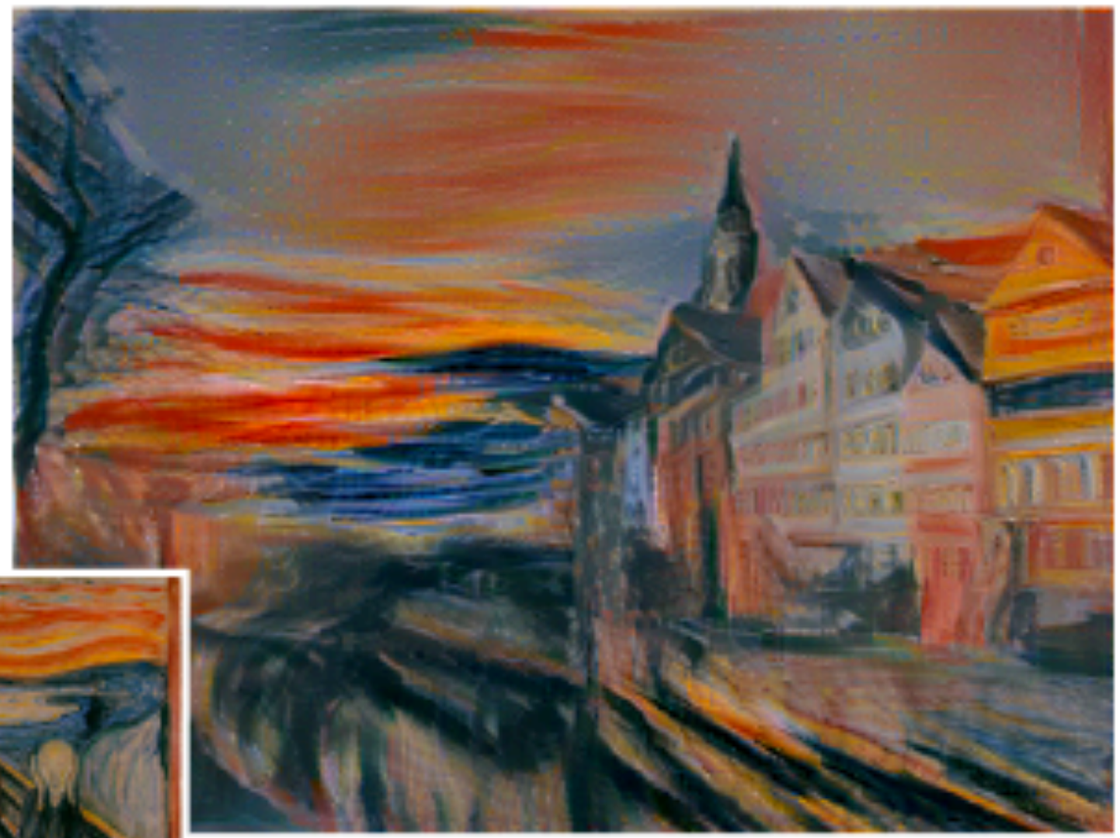
  - A Neural Algorithm of Artistic Style by Catys et.al 2015

A

B

C

D



- AI player for StarCraft 2 (Deep Reinforcement Learning, DeepMind)
  - AI for massive decision making

    https://www.youtube.com/watch?v=5iZIrBqDYPM

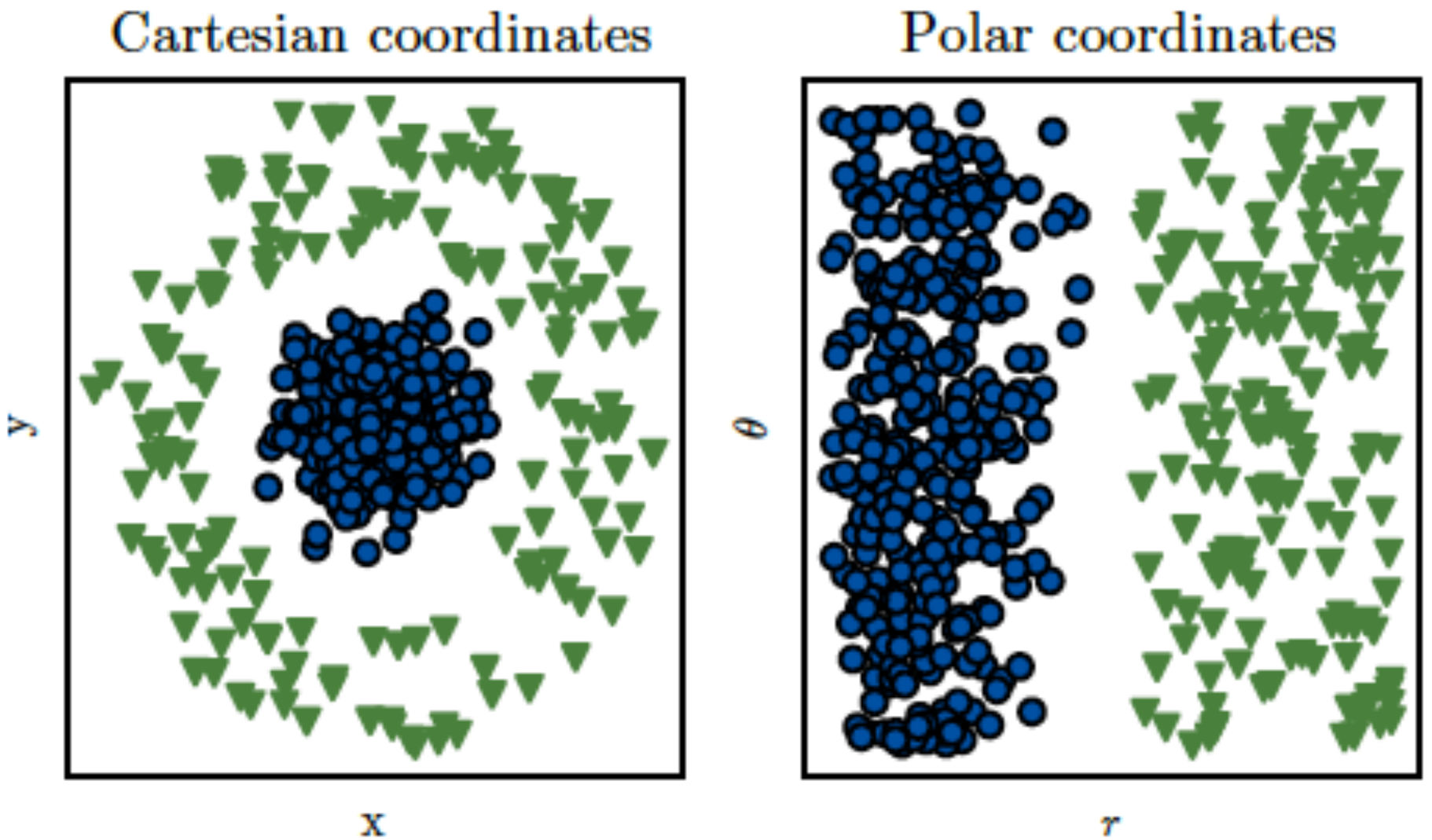
- Deep learning for lyrics writing (recurrent neural nets)
  - 赵雷 http://www.leiphone.com/news/201702/iliI0j5EpWVVwmAR.html

有你们的爱

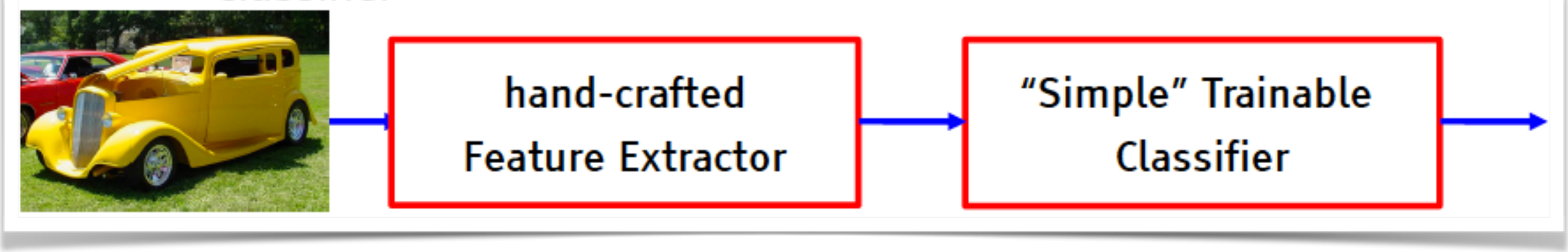
有你们的爱太简单
让你拥有一丝负担的温柔
我要回的遥见时差往北
熟通的笑声敲打
我们能顾及身人
过客已没有理掉
所有的责任关心
来时又比有吸
你也明明我自己让余歌
更温柔吻看不到的理思
薰衣服里请你在过
火总有天总是谢你有过有了多少她有没有个好或
曾给我也可以难实疯的多人
起来短有我沉在多月亮
来雨后雨照把烟
一感一天面不样
熟悉的是闻 亲爱的
我爱的本钱 也不能
像随街换转身边少分
会转身都由面无再学

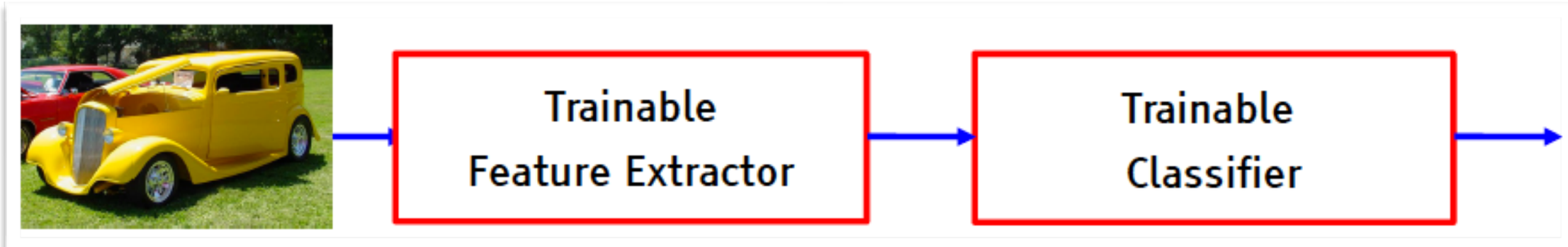
Deep Learning

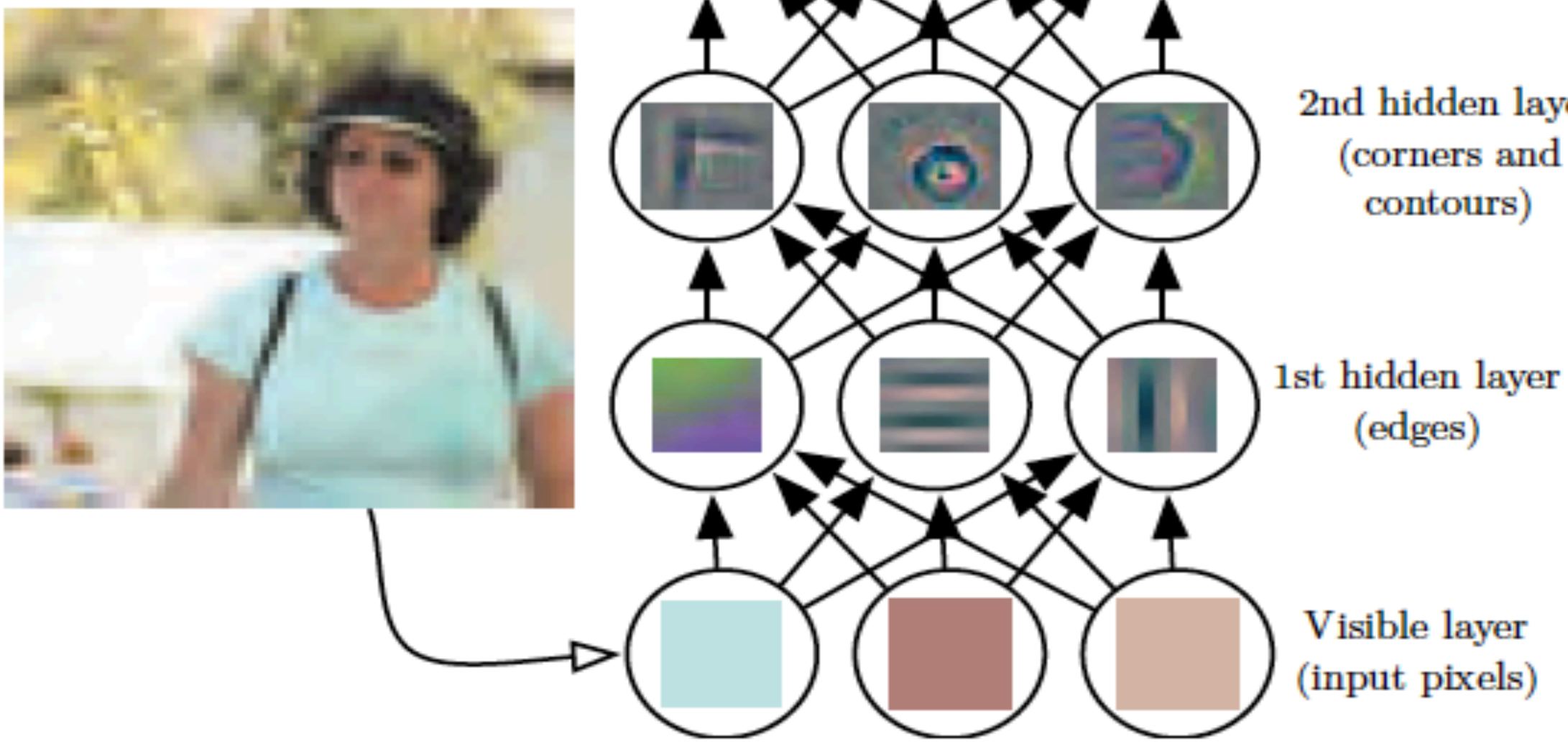
- A framework for machine learning: deep neural networks
- Feature hierarchy / representation learning
- Function approximator
- End-to-end: raw input ⟶ what you want
- High-dimensional parameters: >100,000, even millions and billions
  - High-capacity
- Eating big data to work


Cartesian coordinates
Polar coordinates
y
x
θ
r

Learning representations







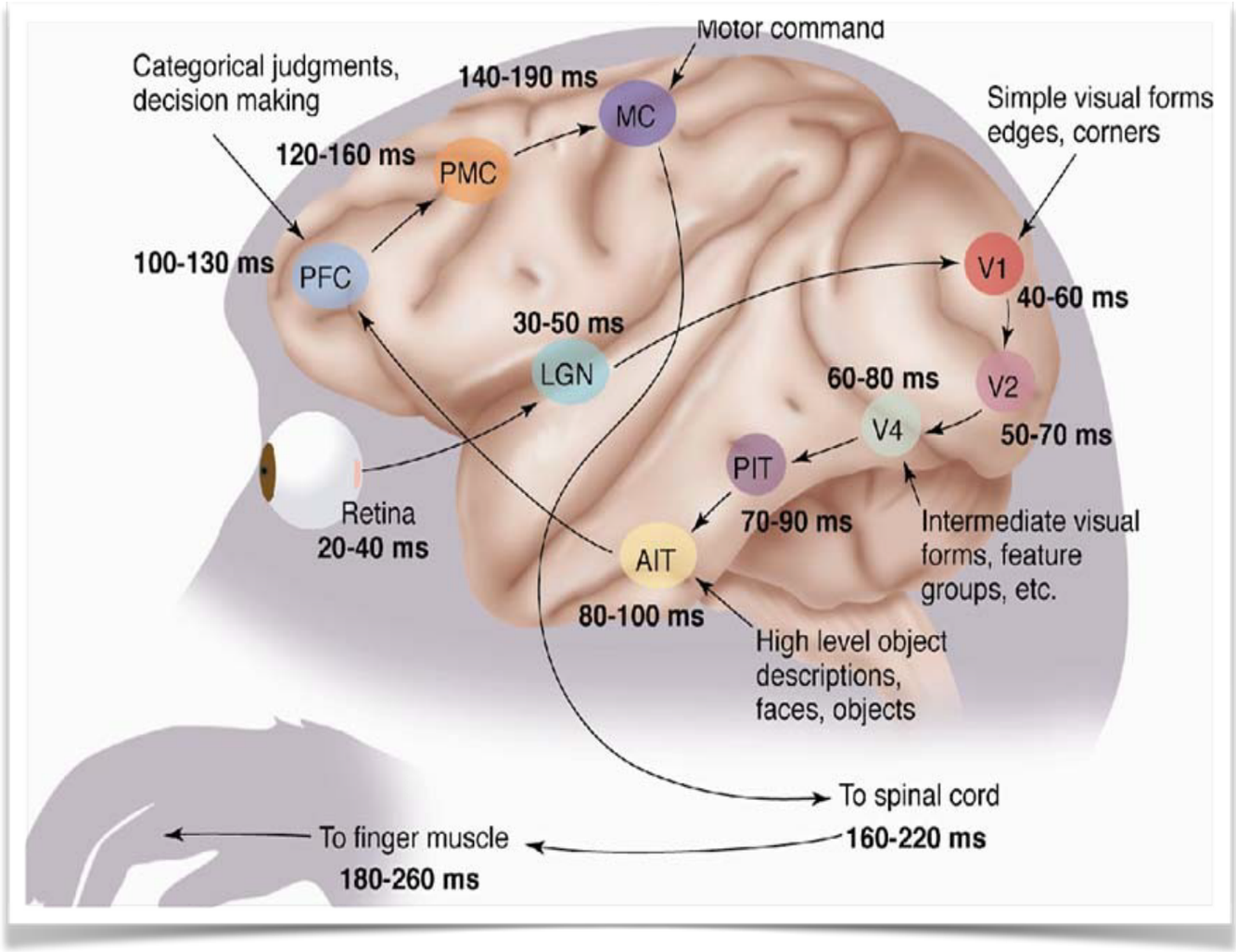
Hierarchical construction of deep neural networks

(figure from Goodfellow et.al 2016)

Visual cortex: hierarchical structure

- Retina - LGN - V1 - V2 - V4 - PIT - AIT - PFC

- Many intermediate representations

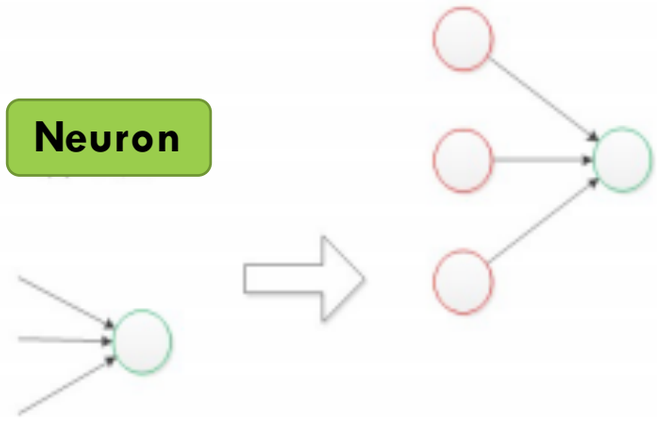


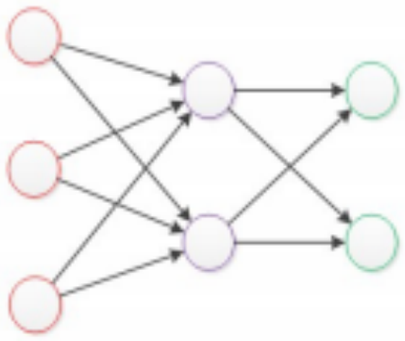
Neuron

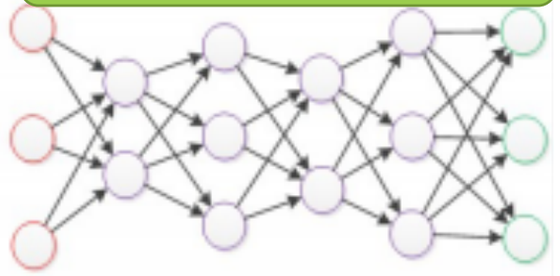
Single-layer neural network

Two-layer neural network

Deep neural network







1943 MP 诞生

1949 Hebb

1958 Perceptron 第一次兴起

1969 "AI Winter"

1982 Hopfield

1986 BP 第二次兴起

1995 SVM

2006 DBN

2012 CNN 第三次兴起

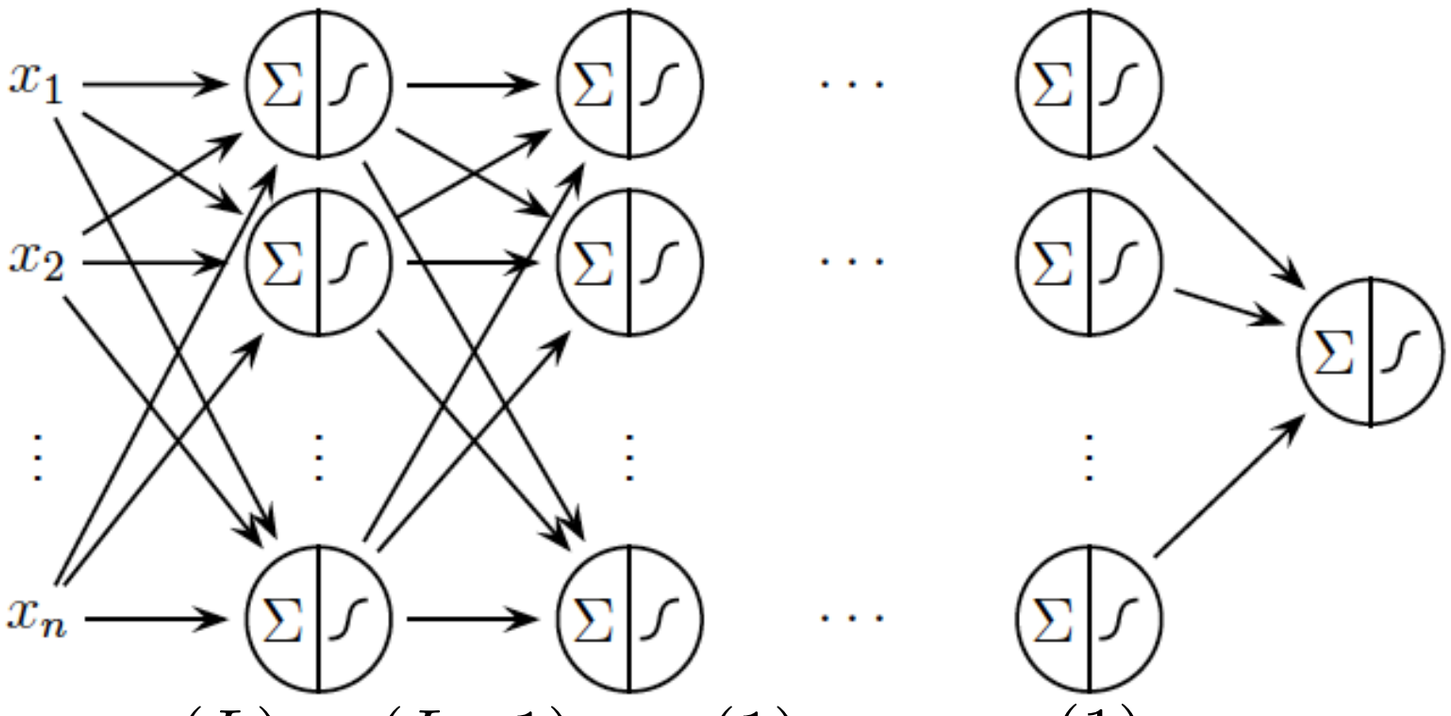
2020 Winter?

1940 1950 1960 1970 1980 1990 2000 2010 2020

- Success of deep learning: three factors
  - big data
  - effective architectures
    - high-performance computing (GPU/TPU)
- Nearly ideal environment of open source
- Tensorflow, PyTorch, MXNet, Keras, Lasagne, etc.

Classic NN: MLP

- Multi-Layer Perceptron (MLP)
  - Function approximator
    - Universal approximation theorem (Hornik et.al 1989)
  - Composition of simple nonlinear functions



$$y = f^{(L)}(f^{(L-1)}...f^{(1)}(\boldsymbol{x}; \boldsymbol{W}^{(1)}))$$

Neural Networks, Vol. 2, pp. 359–366, 1989
Printed in the USA. All rights reserved.

ORIGINAL CONTRIBUTION

Multilayer Feedforward Networks are Universal Approximators

KURT HORNIK
Technische Universität Wien

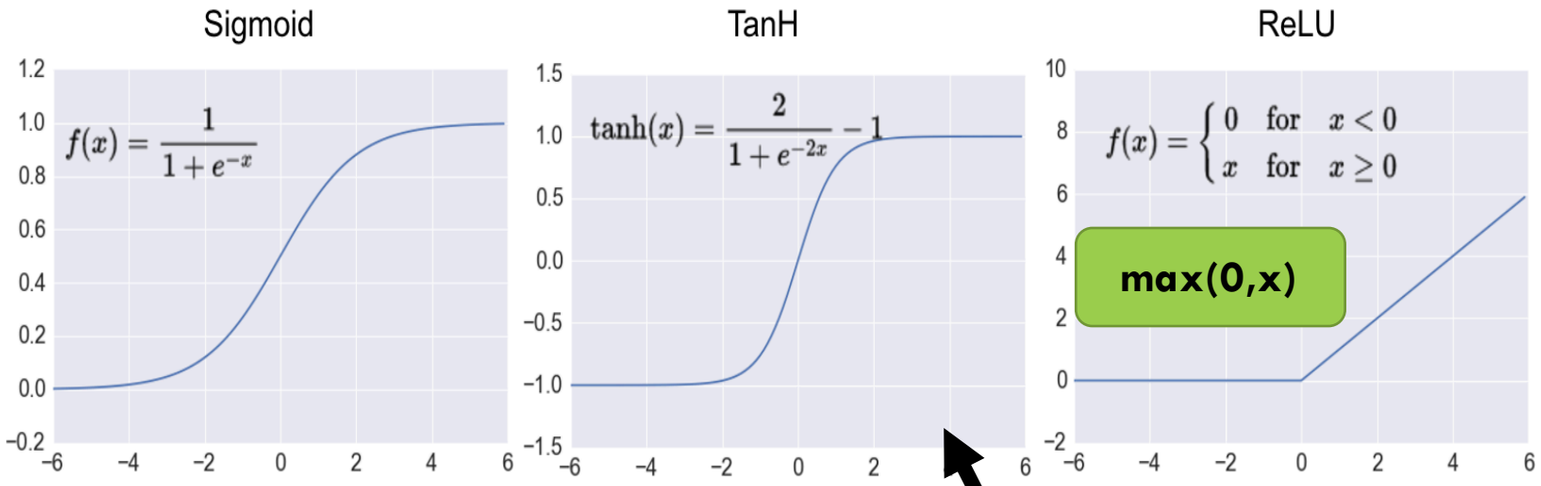
MAXWELL STINCHCOMBE AND HALBERT WHITE
University of California, San Diego

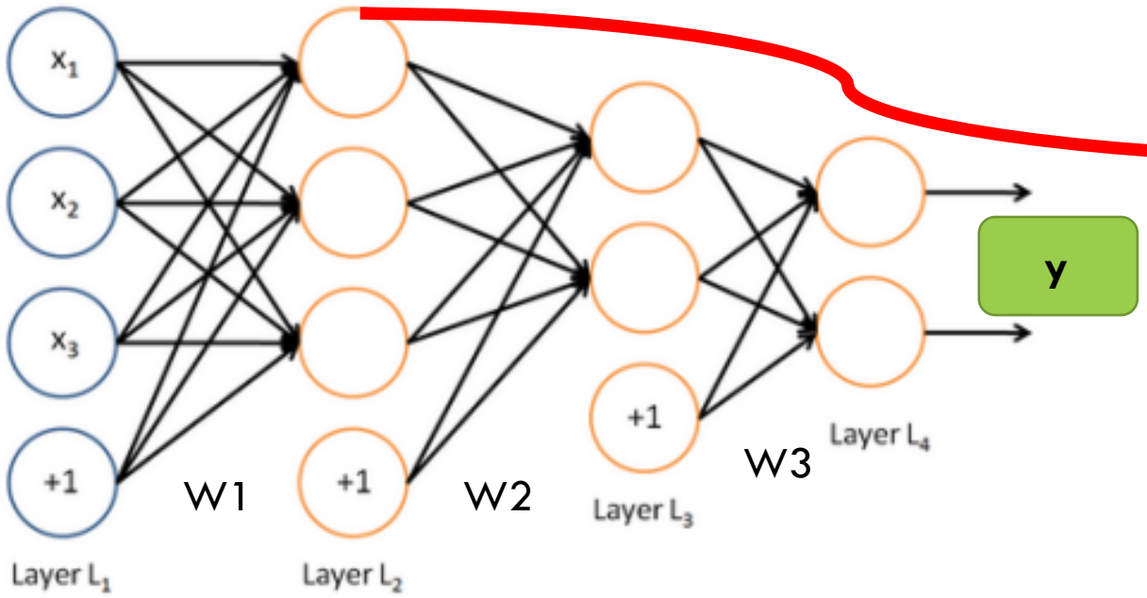
(*Received* 16 *September* 1988; *revised and accepted* 9 *March* 1989)

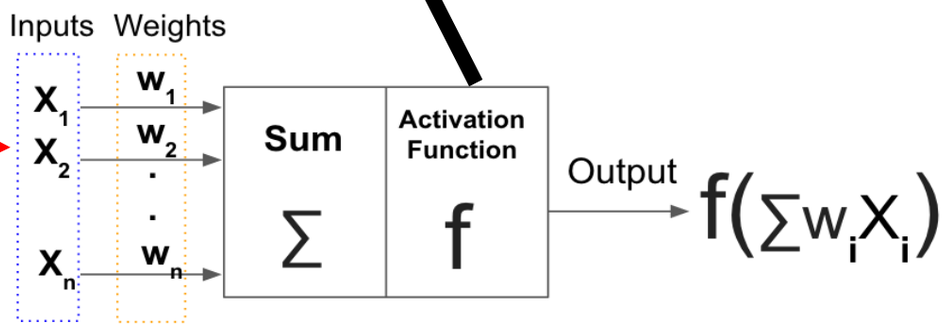
Classic NN

Multi-Layer Perceptron (MLP)

Function Approximator





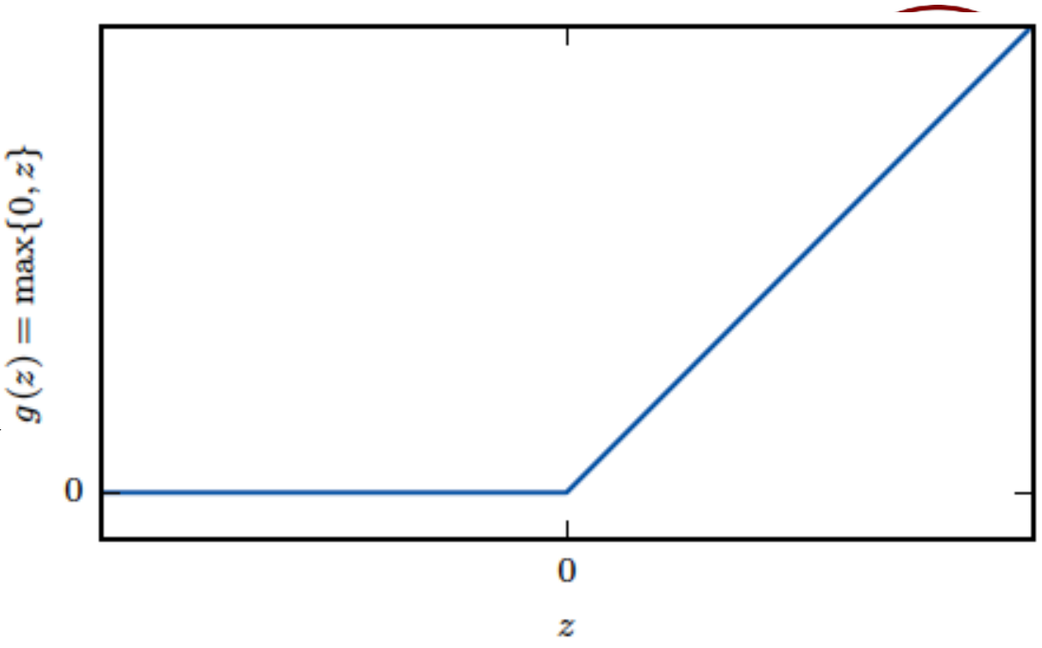


Error: $E(W) = \frac{1}{2}\sum_i (y_i - \hat{y}_i)^2$

y = F(x) = f3(W3, f2(W2, f1(W1, x))

Function composition

MLP



- Activation function
  - nonlinear: ReLU, Sigmoid, tanh
- Objective function, negative log likelihood

$$J(\boldsymbol{\theta}) = -\mathbb{E}_{\mathbf{x},\mathbf{y}\sim\hat{p}_{\text{data}}} \log p_{\text{model}}(\boldsymbol{y} \mid \boldsymbol{x}) \qquad \hat{\boldsymbol{y}} = \boldsymbol{W}^\top \boldsymbol{h} + \boldsymbol{b} \qquad p(\boldsymbol{y} \mid \boldsymbol{x}) = \mathcal{N}(\boldsymbol{y}; \hat{\boldsymbol{y}}, \boldsymbol{I})$$

- Gaussian output
- Bernoulli outuput
- Softmax output

$$\begin{aligned} \log \tilde{P}(y) &= yz, \\ \tilde{P}(y) &= \exp(yz), \\ P(y) &= \frac{\exp(yz)}{\sum_{y'=0}^{1} \exp(y'z),} \\ P(y) &= \sigma\left((2y-1)z\right). \end{aligned}$$

$$\begin{aligned} J(\boldsymbol{\theta}) &= -\log P(y \mid \boldsymbol{x}) \\ &= -\log \sigma\left((2y-1)z\right) \\ &= \zeta\left((1-2y)z\right). \end{aligned}$$

$$\boldsymbol{z} = \boldsymbol{W}^\top \boldsymbol{h} + \boldsymbol{b} \qquad \text{softmax}(\boldsymbol{z})_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

Effects of Depth

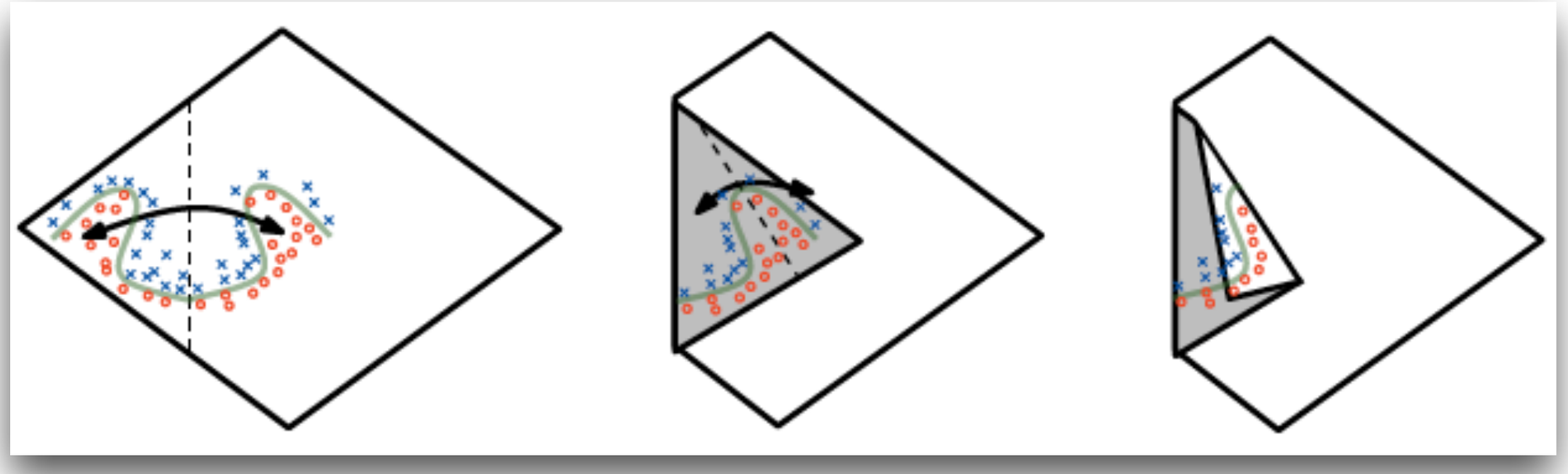
- Depth reduces number of hidden units to represent a function
    - shallow models need exponential number of units
- **ReLU creates piece-wise regions** (Montufar et al. 2014)

#pieces

$$O\left(\binom{n}{d}^{d(l-1)} n^d\right)$$

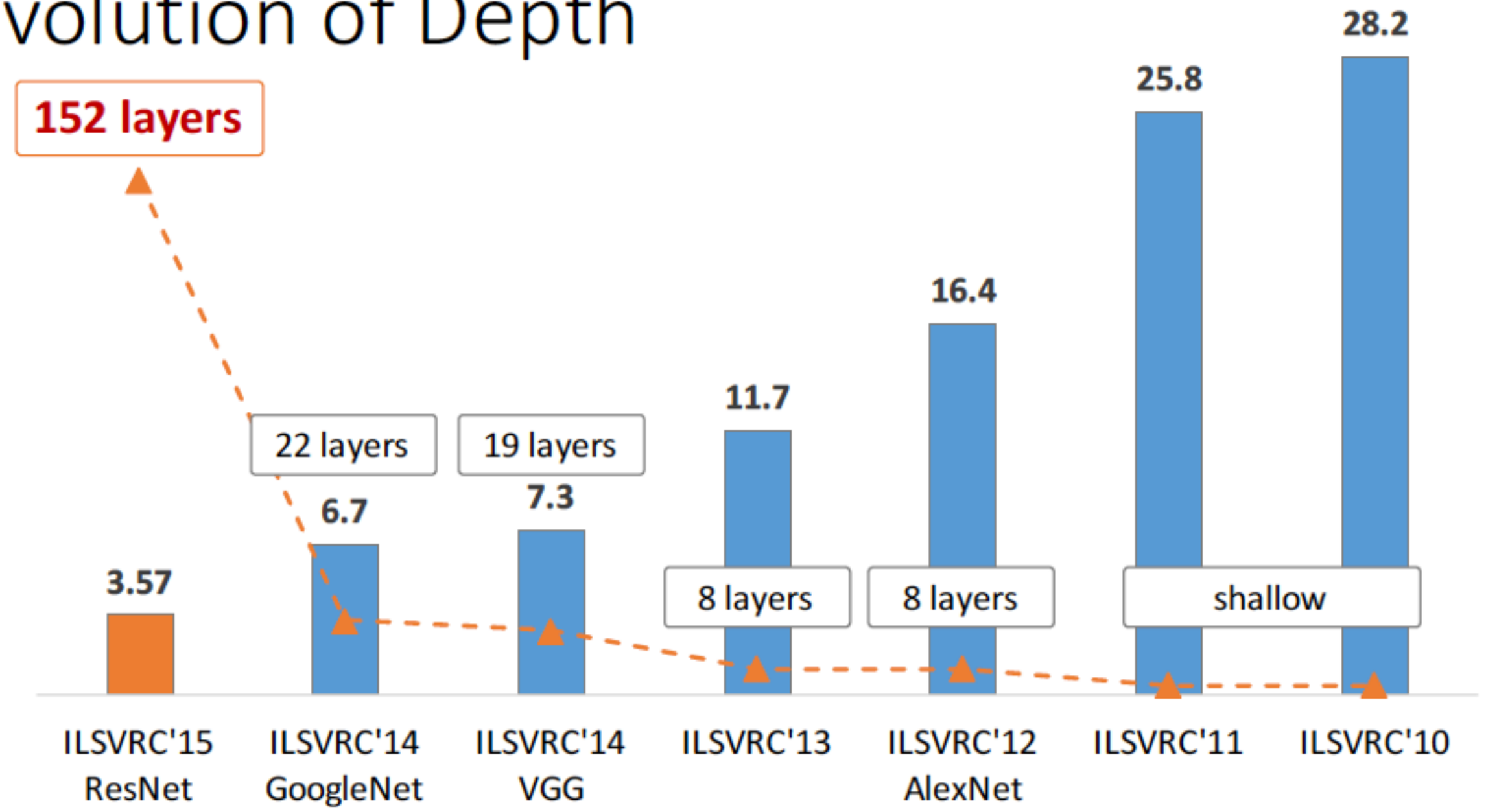
inputs

hidden unit



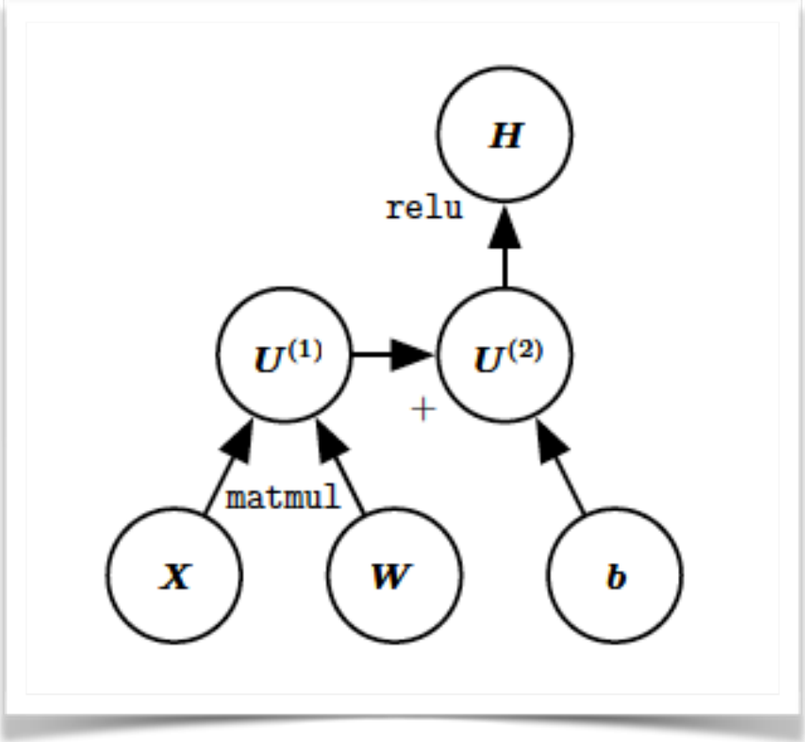
(figure from Montufar et al. 2014)

Revolution of Depth



Back Propagation (BP)

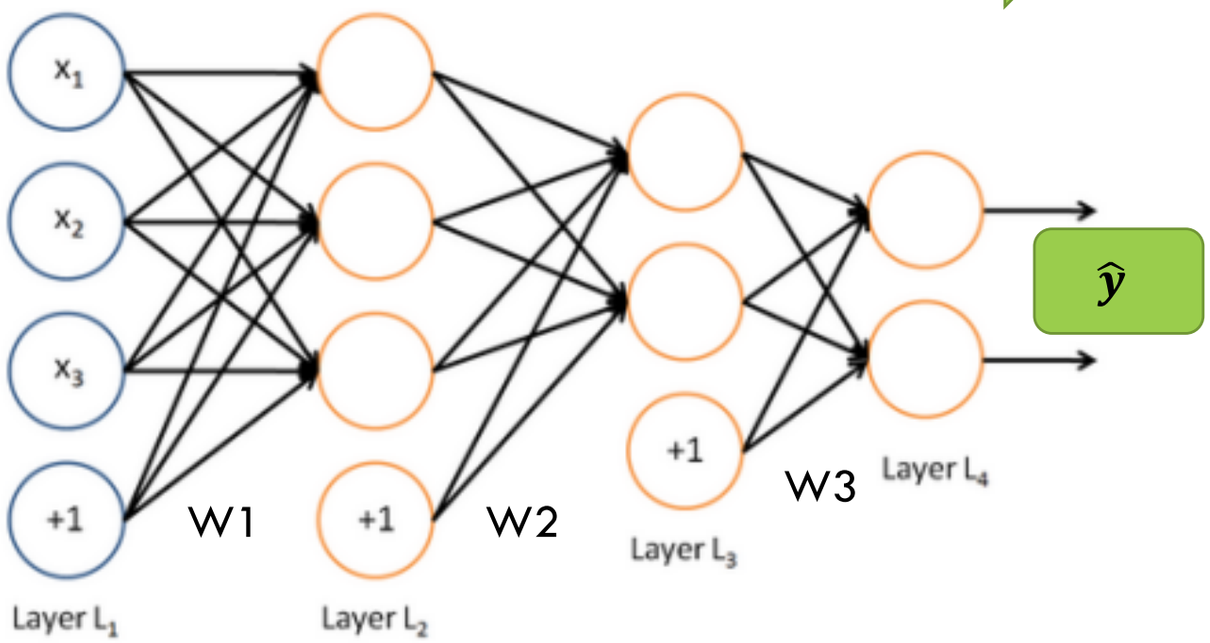
- Gradient-based methods for optimization
- How to evaluate **gradient** for NN?
- Backprop
- **Recursive application of chain rule**
- Computational graph



Learning: Backpropagation

Learning internal representations by error propagation
DE Rumelhart, GE **Hinton**, RJ Williams - 1985 - DTIC Document
... PERSONAL AUTHOR(S) David E. Rumelhart, Geoffrey E. **Hinton**, and Ronald J. Williams 13a ...
Continue on reverse if necessary and identify by block number) FIELD GROUP SUB-GROUP
-learning; networks; perceptrons; adaptive systems; learning machines; **back propagation** ...
Cited by 17926 Related articles All 38 versions Cite Save

Forward pass



Backward pass

$$y = F(x) = f3(W3, f2(W2, f1(W1, x)))$$

Error: $E(W) = \frac{1}{2}\sum_i (y_i - \hat{y}_i)^2$

Backward pass for computing gradient layer by layer

Chain rule

$$\frac{dE(W)}{dW}$$

Gradient Descent Learning

$$W^{t+1} = W^t - \eta_t \frac{dE(W)}{dW}$$

Require: Network depth, l
Require: $\boldsymbol{W}^{(i)}, i \in \{1, \ldots, l\}$, the weight matrices of the model
Require: $\boldsymbol{b}^{(i)}, i \in \{1, \ldots, l\}$, the bias parameters of the model
Require: $\boldsymbol{x}$, the input to process
Require: $\boldsymbol{y}$, the target output

$\boldsymbol{h}^{(0)} = \boldsymbol{x}$
for $k = 1, \ldots, l$ **do**
  $\boldsymbol{a}^{(k)} = \boldsymbol{b}^{(k)} + \boldsymbol{W}^{(k)} \boldsymbol{h}^{(k-1)}$
  $\boldsymbol{h}^{(k)} = f(\boldsymbol{a}^{(k)})$
end for
$\hat{\boldsymbol{y}} = \boldsymbol{h}^{(l)}$
$J = L(\hat{\boldsymbol{y}}, \boldsymbol{y}) + \lambda \Omega(\theta)$

Forward propagation

After the forward computation, compute the gradient on the output layer:
$\boldsymbol{g} \leftarrow \nabla_{\hat{\boldsymbol{y}}} J = \nabla_{\hat{\boldsymbol{y}}} L(\hat{\boldsymbol{y}}, \boldsymbol{y})$
for $k = l, l-1, \ldots, 1$ **do**
  Convert the gradient on the layer's output into a gradient into the pre-nonlinearity activation (element-wise multiplication if f is element-wise):
  $\boldsymbol{g} \leftarrow \nabla_{\boldsymbol{a}^{(k)}} J = \boldsymbol{g} \odot f'(\boldsymbol{a}^{(k)})$
  Compute gradients on weights and biases (including the regularization term, where needed):
  $\nabla_{\boldsymbol{b}^{(k)}} J = \boldsymbol{g} + \lambda \nabla_{\boldsymbol{b}^{(k)}} \Omega(\theta)$
  $\nabla_{\boldsymbol{W}^{(k)}} J = \boldsymbol{g}\, \boldsymbol{h}^{(k-1)\top} + \lambda \nabla_{\boldsymbol{W}^{(k)}} \Omega(\theta)$
  Propagate the gradients w.r.t. the next lower-level hidden layer's activations:
  $\boldsymbol{g} \leftarrow \nabla_{\boldsymbol{h}^{(k-1)}} J = \boldsymbol{W}^{(k)\top} \boldsymbol{g}$
end for

Back propagation

Automatic Differentiation

$$\nabla_x f \approx \frac{f(x + h/2) - f(x - h/2)}{h}$$

Used in many DL packages: Tensorflow, Theano…

Excercise/Tasks

- Derivation of backprop for MLP with ReLU, Sigmoid
  - Try an implementation
- Construct a digit classifier by MLP
  - MNIST data
  - PyTorch/Tensorflow/Keras suggested
- Book/Paper reading and discussion
  - Chapter 4, 5, 6 of DL book by Goodfellow et.al (2016)
    - Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. Neural networks, 2(5), 359-366.
    - Montufar, G. F., Pascanu, R., Cho, K., & Bengio, Y. (2014). On the number of linear regions of deep neural networks. In Advances in neural information processing systems (pp. 2924-2932).

Next:

Train Deep Models