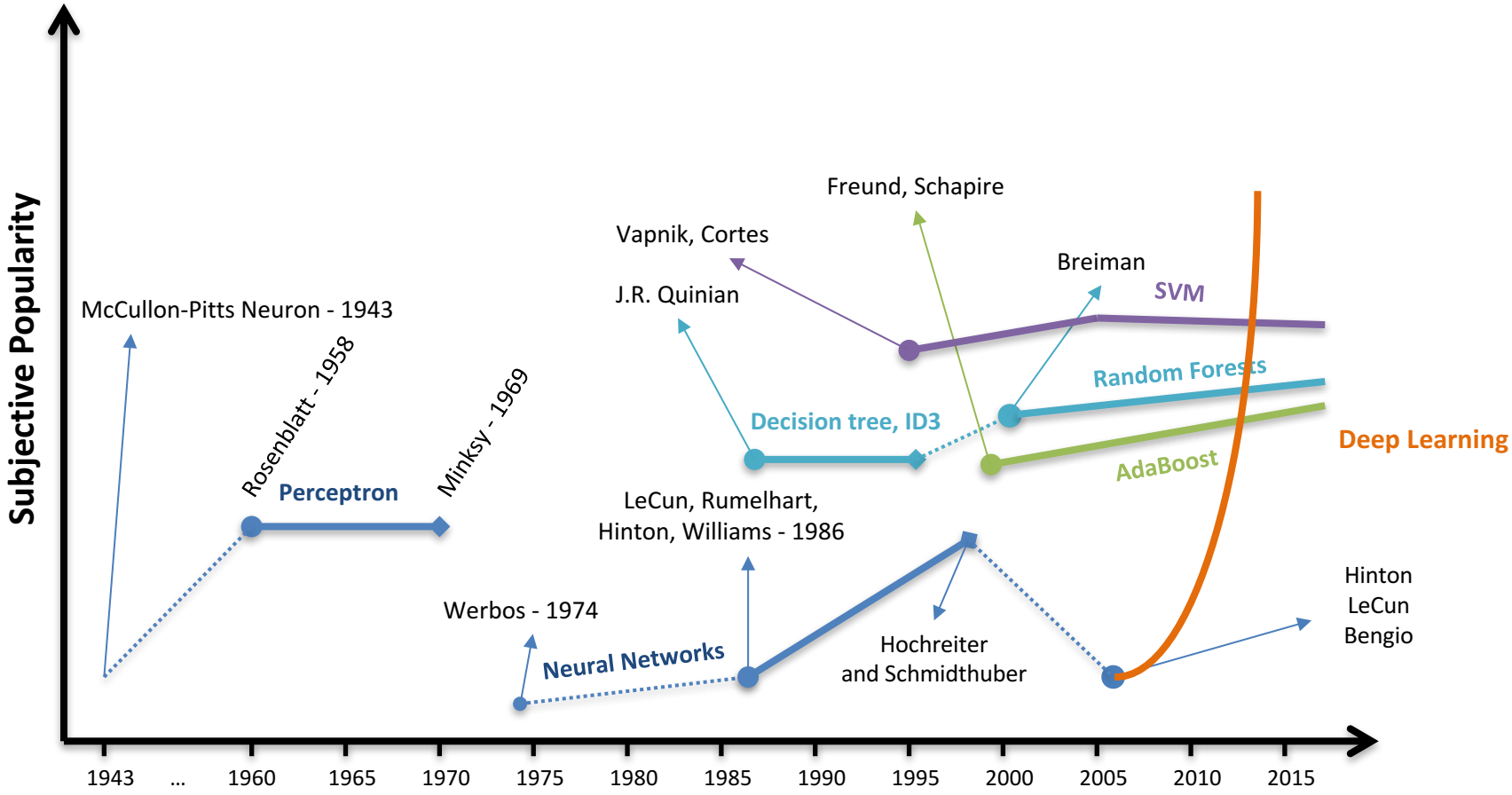
# A Concise History of Neural Networks

Dr. Haojin Yang

# Why Learn History?

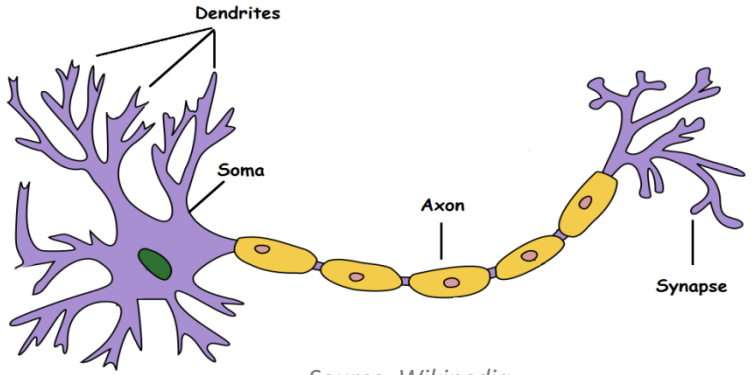
"… the best reason to learn history: not in order to predict the future, but to free yourself of the past and imagine alternative destinies. Of course this is not total freedom - we cannot avoid being shaped by the past. But some freedom is better than none."

- *Homo Deus: A Brief History of Tomorrow* (Yuval Noah Harari, 2015)


Subjective Popularity
McCullon-Pitts Neuron - 1943
Rosenblatt - 1958
Perceptron
Minksy - 1969
Vapnik, Cortes
Freund, Schapire
J.R. Quinian
Breiman
SVM
Random Forests
Decision tree, ID3
AdaBoost
Deep Learning
LeCun, Rumelhart,
Hinton, Williams - 1986
Werbos - 1974
Neural Networks
Hochreiter
and Schmidthuber
Hinton
LeCun
Bengio
1943
...
1960
1965
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015

# A Biological Neuron

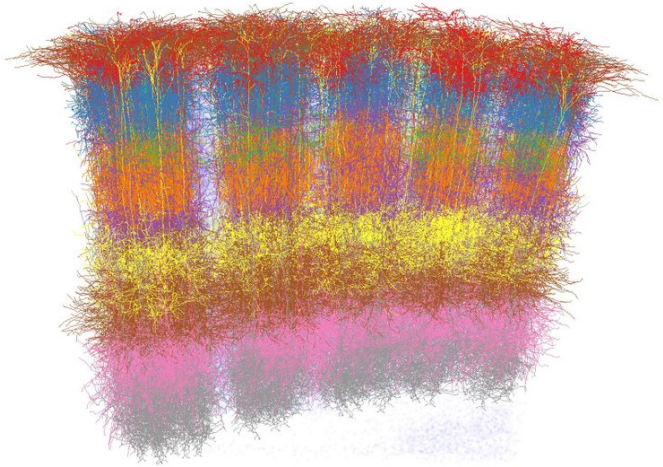
- Dendrite: Receives signals from other neurons
- Soma: Processes the information
- Axon: Transmits the output of this neuron
- Synapse: Point of connection to other neurons




*Source: Wikipedia*

# Cortical Columns

- Neurons in cortex arranged into many stacks, or “columns” that process information in parallel

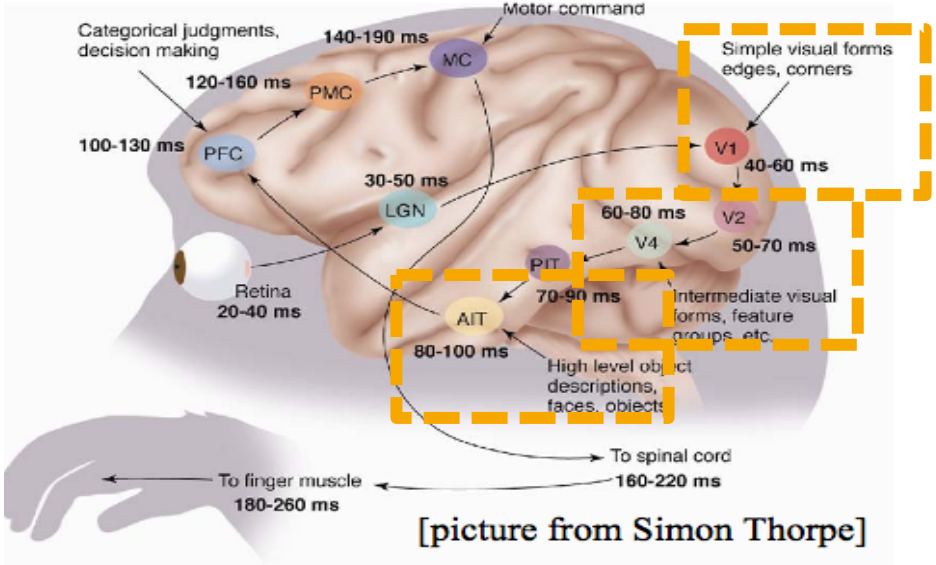


*credit: Marcel Oberlaender et al.*

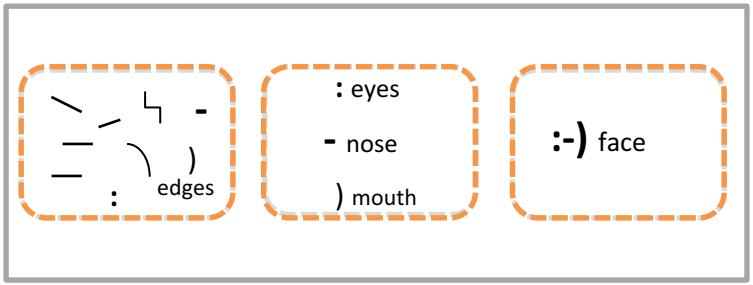
# Enlightenment:
# The Mammalian Visual Cortex is Hierarchical

The ventral (recognition) pathway in the visual cortex has multiple stages:

- Retina - LGN - V1 - V2 - V4 - PIT - AIT …., lots of intermediate representations

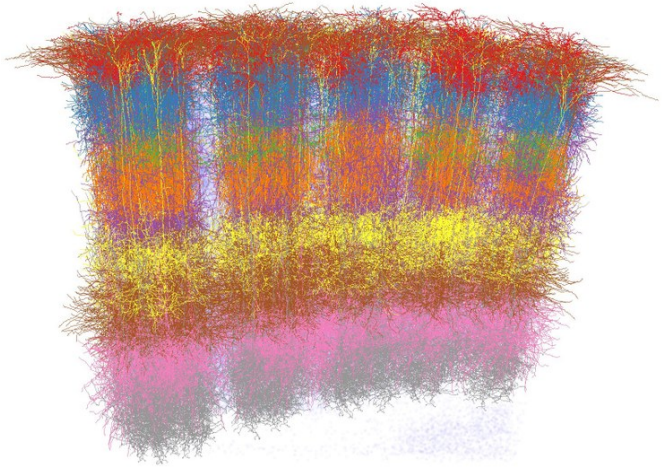



[Gallant & Van Essen]



# Cortical Columns

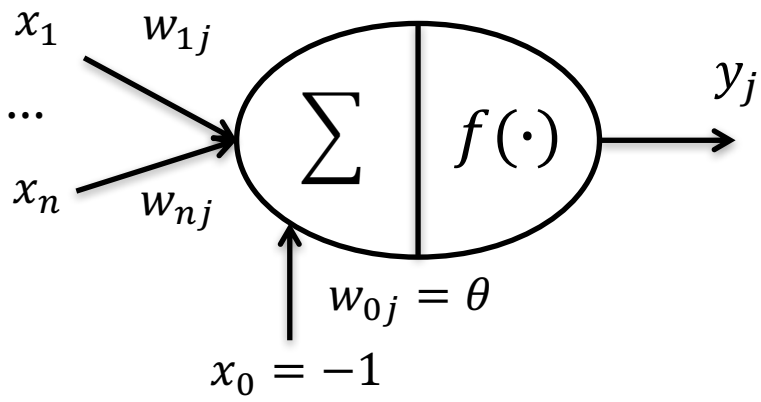
- “mini-columns” of around 100 neurons are organized into larger “hyper-columns”.
- There are 100 million mini-columns in your cortex
- mini-column is statistically meaningful, but what is the functional meaning?



*credit: Marcel Oberlaender et al.*

# M-P Neuron

*A Logical Calculus of the Ideas Immanent in Nervous Activity,* Warren McCulloch and Walter Pitts (1943)



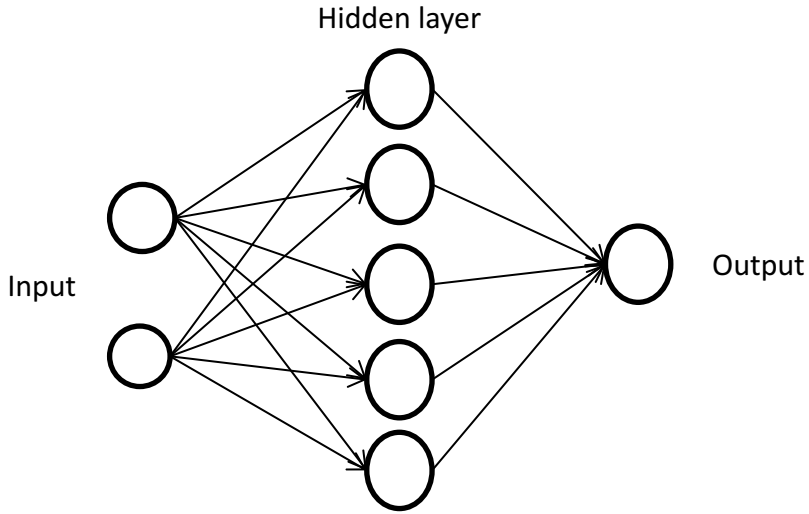

$$y_j = \sum_{i=1}^{n} w_{ij} x_i - \theta_j$$

| Biological | M-P |
|---|---|
| neuron | $j$ |
| inputs | $x_i$ |
| weights | $w_{ij}$ |
| output | $y_j$ |
| sum | $\sum$ |
| membrane potential | $\sum_{i=1}^{n} w_{ij} x_i$ |
| threshold | $\theta$ |

Warren McCulloch & Walter Pitts

# Perceptron

- Single layer neural network




Frank Rosenblatt
(1928-1971)

$$y_j = \begin{cases} 0 \;\; if \sum_{i=1}^{n} w_{ij} x_i \leq \theta_j \\ 1 \;\; if \sum_{i=1}^{n} w_{ij} x_i > \theta_j \end{cases}$$

# Perceptron

- Follows the Hebbian theory

$$w_{ij} = x_j \rightarrow x_i$$

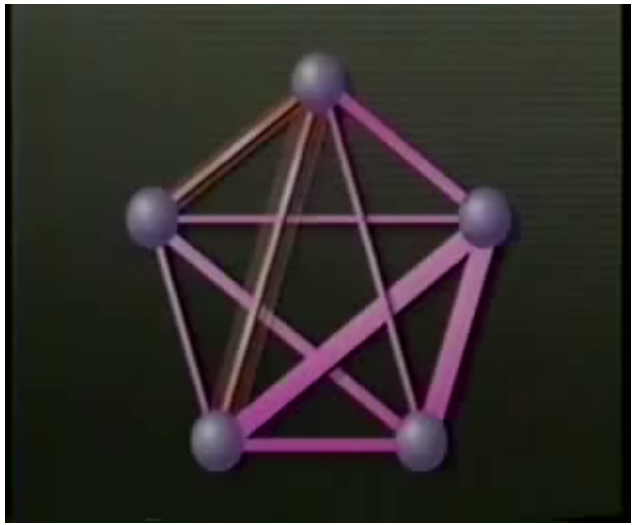
$$\Delta w_{ij} = \eta \cdot a_i \cdot y_j$$

$\Delta w_{ij}$: the changes in the i_th synaptic weight

$\eta$ : learning rate

$a_i$ : activation value of i_th synapse

$y_j$ : output of j_th synapse



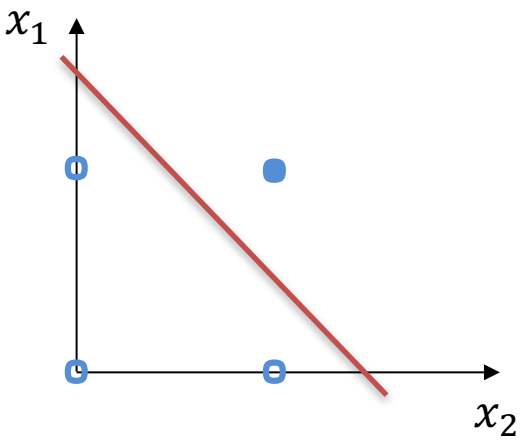
*source: youtube*

# Perceptron

*Perceptrons,* Marvin Minsky and Seymour Papert (1969)

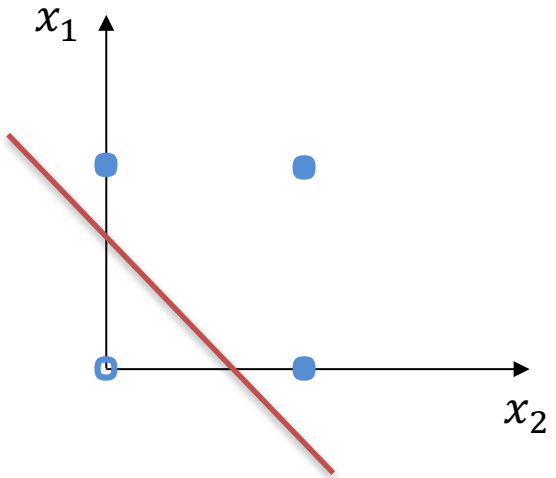
| $x_1$ | $x_2$ | $y$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

***AND***




| $x_1$ | $x_2$ | $y$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

***OR***

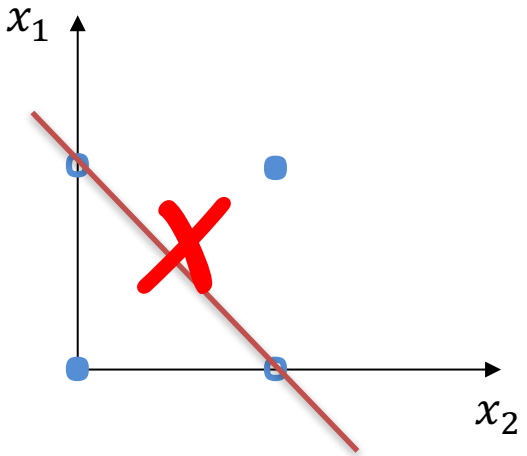


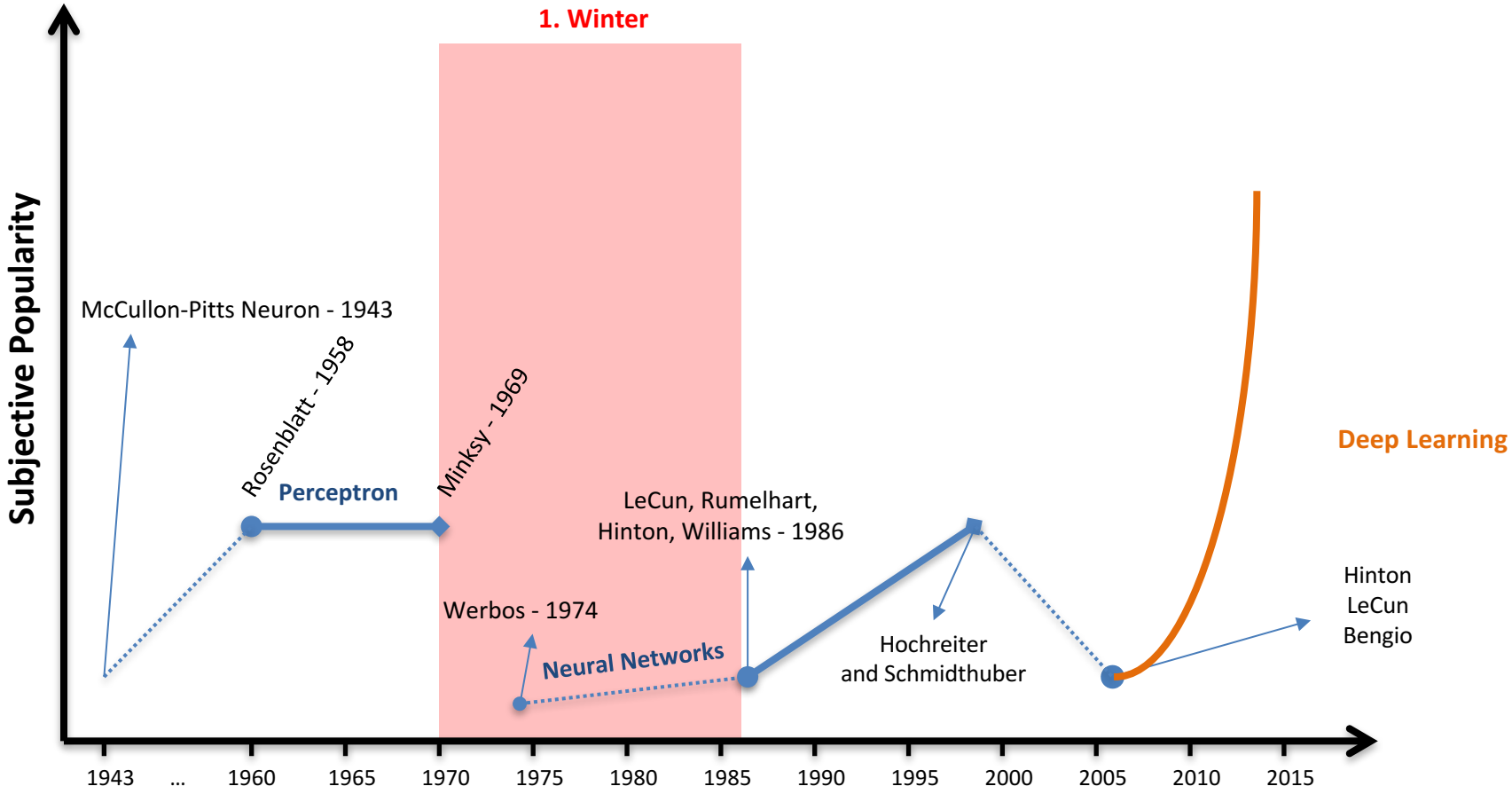
# Perceptron

*Perceptrons,* Marvin Minsky and Seymour Papert (1969)

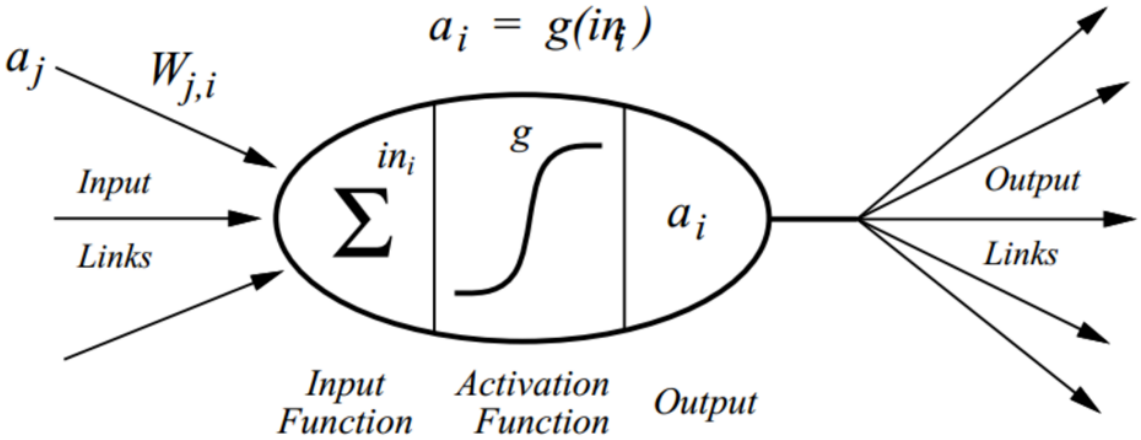
| $x_1$ | $x_2$ | $y$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

***XOR***

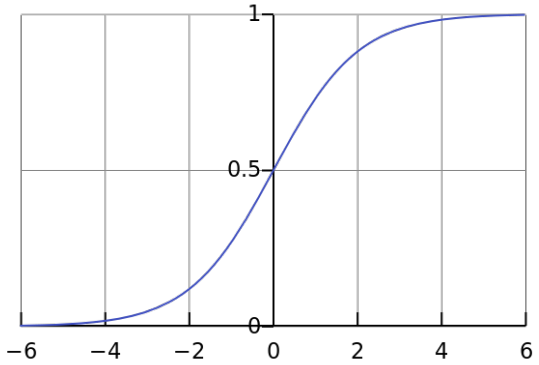



1. Winter
Subjective Popularity
McCullon-Pitts Neuron - 1943
Rosenblatt - 1958
Perceptron
Minksy - 1969
LeCun, Rumelhart,
Hinton, Williams - 1986
Deep Learning
Hinton
LeCun
Bengio
Werbos - 1974
Neural Networks
Hochreiter
and Schmidthuber
1943
...
1960
1965
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015

# Non-linear Activation



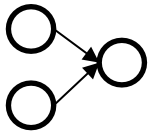
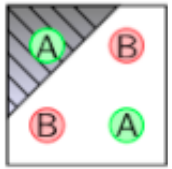
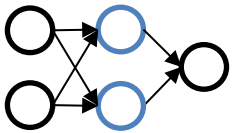
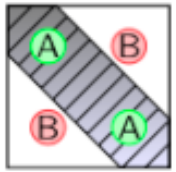
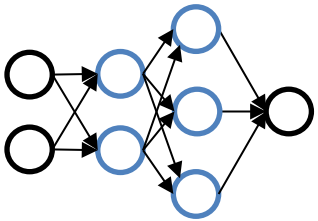
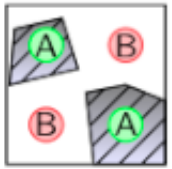

**Sigmoid function**




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

# Multi-Layer Perceptron

*Perceptrons - Expanded Edition,* Marvin Minsky (1987)

| Structure | Decision area | Area shape | XOR problem |
|---|---|---|---|
| linear activation  | divided by one hyperplane |  |   |
| 1 hidden layer, non-linear  | open convex area and closed convex area |  |   |
| double hidden layer  | arbitrary shape (complexity depending on the number of units) |  |   |

*Kurt Hornik et al., Multilayer feedforward networks are universal approximators. In: Neural networks 2 (1989)*

*George Cybenko, Approximation by superpositions of a sigmoidal function. In: Mathematics of control, signals and systems 2 (1989)*

# Backpropagation

- ~1986 **Backpropagation**: “Learning representations by back-propagating errors”
- BP enables training multi-layer artificial neural networks
  - Update weights using Gradient Descent
  - BP is very efficient for calculating $\nabla_\theta \text{Loss}$

Geoffrey Hinton

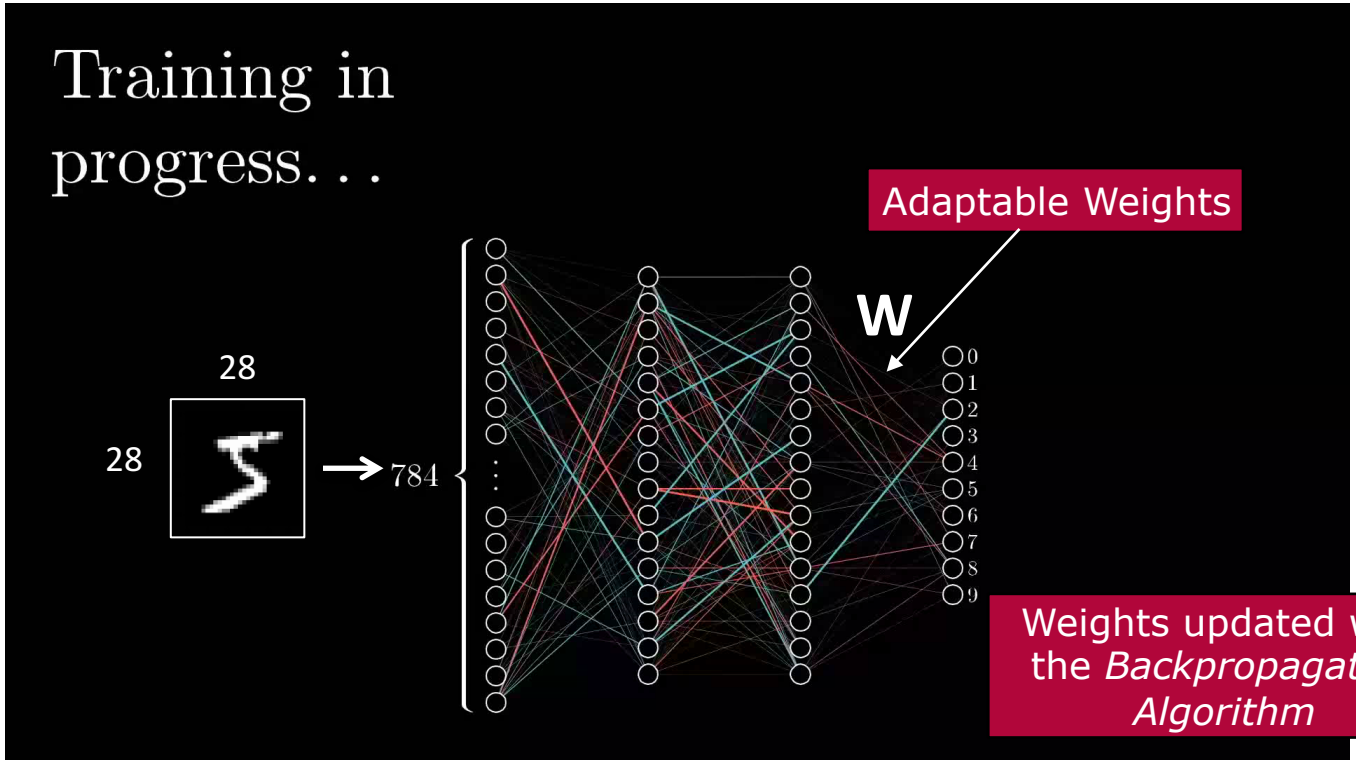
NATURE VOL. 323 9 OCTOBER 1986

**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

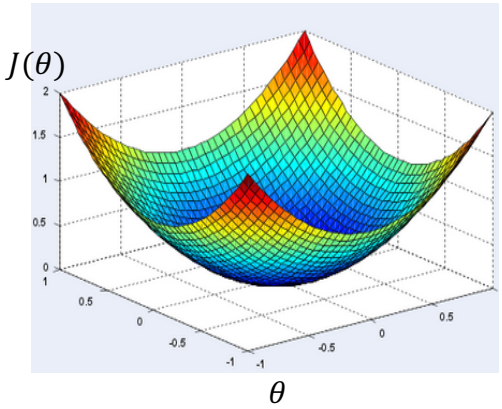
# Artificial Neural Networks

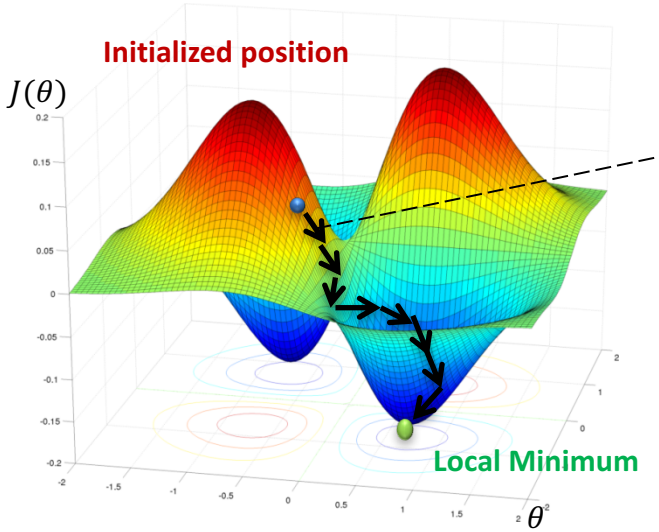



*Source: gfycat.com*

# Gradient Descent

$$\theta := \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i;i+n)}; y^{(i;i+n)})$$





Learning rate $\eta$ controls the step size

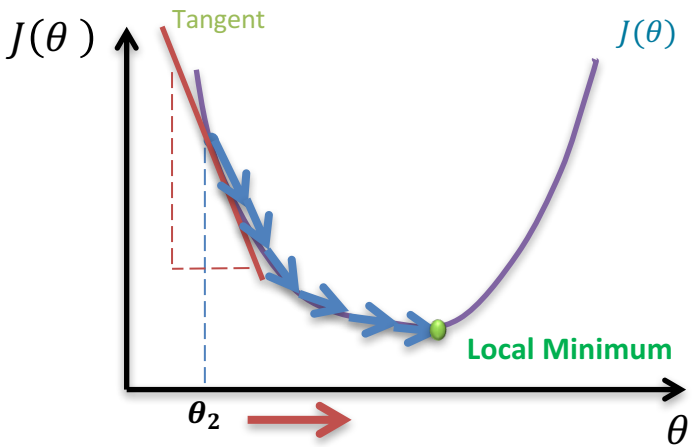
# Gradient Descent

- Optimization objective: $min\, J\left(\theta; x^{(i;i+n)}; y^{(i;i+n)}\right), \theta \epsilon R$

$$\theta := \theta - \eta \cdot \frac{\partial J(\theta; x^{(i;i+n)}; y^{(i;i+n)})}{\partial \theta}$$

Learning rate

Derivative

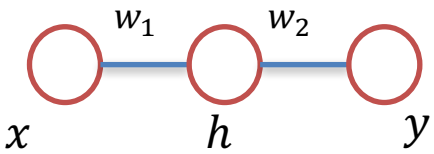



$$\theta_2 := \theta_2 - \eta \cdot \frac{\partial J(\theta_2)}{\partial \theta_2}$$

$< 0$

$$\theta_2 := \theta_2 - \eta \cdot (negative\ num)$$

# Chaining




$$h = x * w_1$$

$$\frac{\partial h}{\partial w_1} = x$$

$$y = h * w_2$$

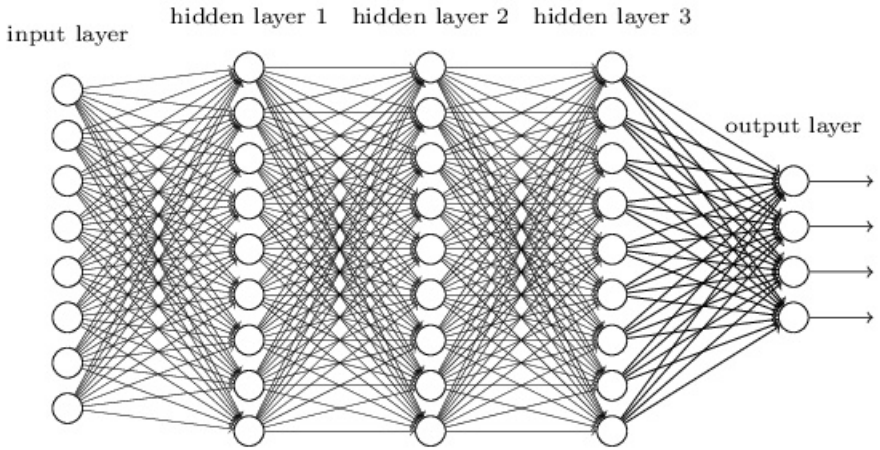
$$\frac{\partial y}{\partial h} = w_2$$

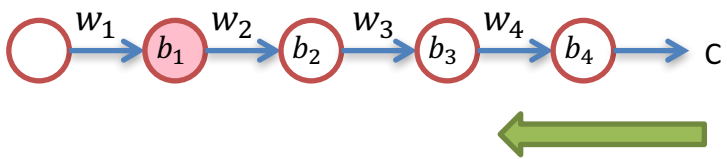
$$y = x * w_1 * w_2$$

$$\frac{\partial y}{\partial w_1} = w_2 * x$$

$$\boldsymbol{\frac{\partial y}{\partial w_1} = \frac{\partial y}{\partial h}\frac{\partial h}{\partial w_1}}$$

# Chaining



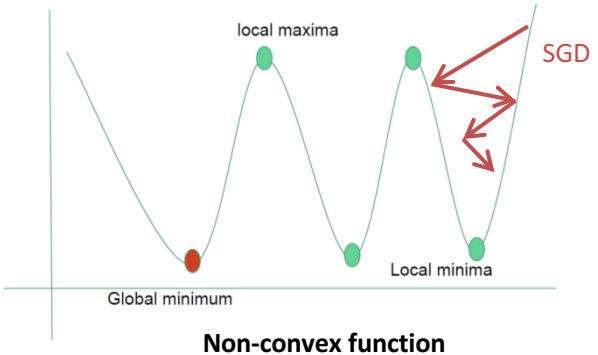




$$y = x_{i+1} = \sigma(z_i) = \sigma(w_i x_i + b_i),\ where\ z_i = w_i x_i + b_i$$

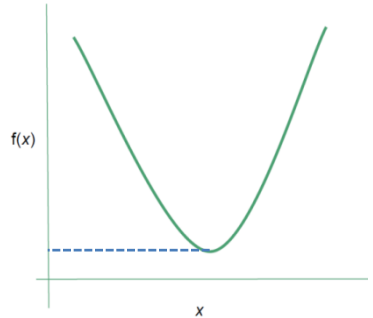
$$\frac{\partial C}{\partial b_1} = \frac{\boldsymbol{\partial C}}{\partial y_4}\frac{\partial y_4}{\partial z_4}\frac{\partial z_4}{\partial x_4}\frac{\partial x_4}{\partial z_3}\frac{\partial z_3}{\partial x_3}\frac{\partial x_3}{\partial z_2}\frac{\partial z_2}{\partial x_2}\frac{\partial x_2}{\partial z_1}\frac{\partial z_1}{\boldsymbol{\partial b_1}}$$

# Backpropagation

- BP is biologically difficult to explain
- BP based on Gradient Descent algorithms → **non-convex** optimization problem
- Other methods were better e.g., SVM
- **Unable to successfully train deeper networks**

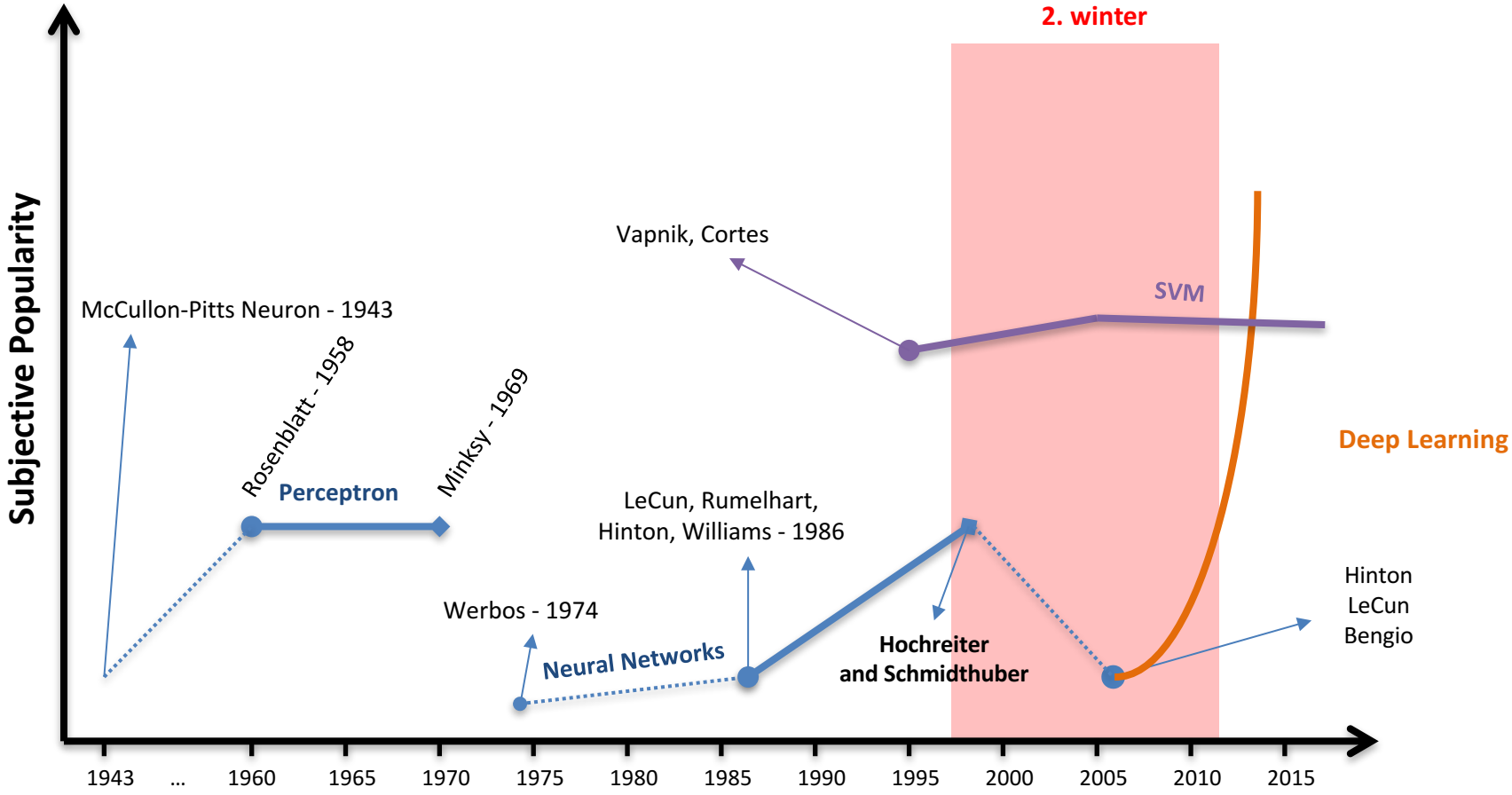



**Non-convex function**




Strongly convex

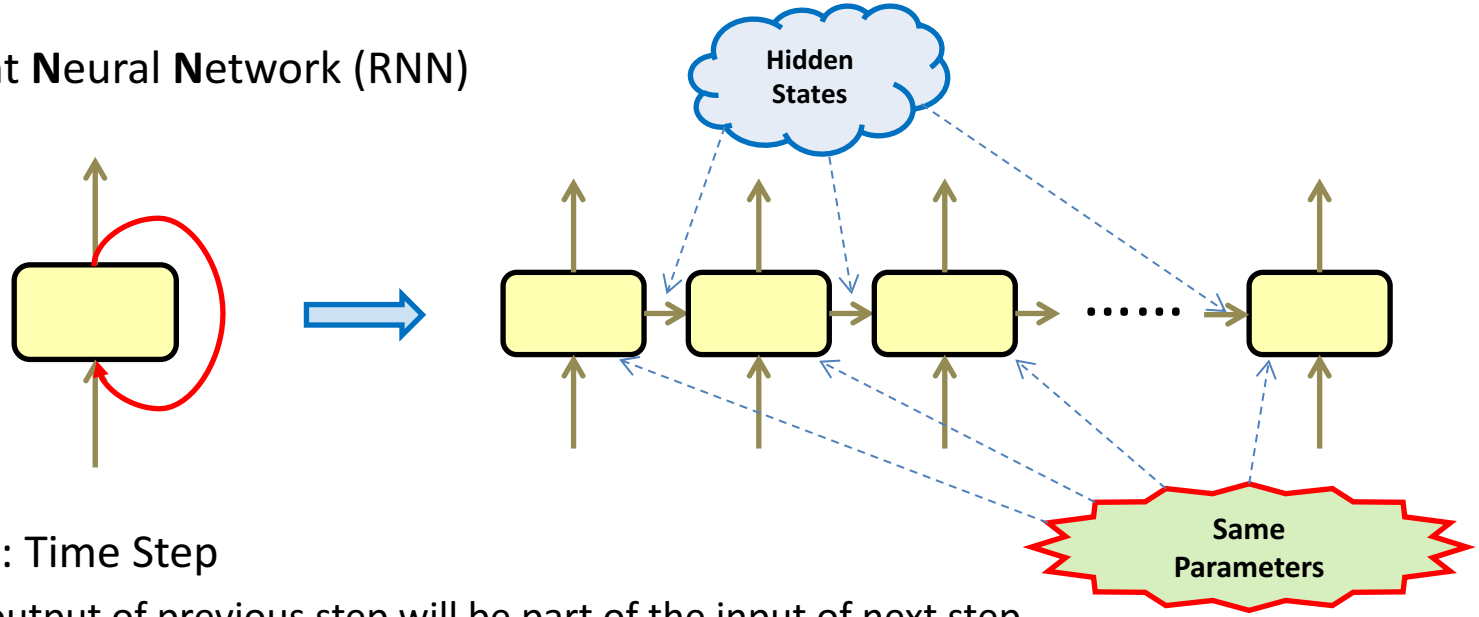
*(Image source: Issam Laradji)*


2. winter
Subjective Popularity
Vapnik, Cortes
SVM
McCullon-Pitts Neuron - 1943
Rosenblatt - 1958
Minksy - 1969
Perceptron
Deep Learning
LeCun, Rumelhart,
Hinton, Williams - 1986
Werbos - 1974
Neural Networks
Hochreiter
and Schmidthuber
Hinton
LeCun
Bengio
1943
...
1960
1965
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015

# Why can't it be deeper?

*Untersuchungen zu dynamischen neuronalen Netzen*. Sepp Hochreiter, Diploma thesis, TU Munich (1991)

- **R**ecurrent **N**eural **N**etwork (RNN)




- **Keyword**: Time Step
  - The output of previous step will be part of the input of next step.
    - "Hidden State"
  - Parameters in the layer are kept static through continuous steps.

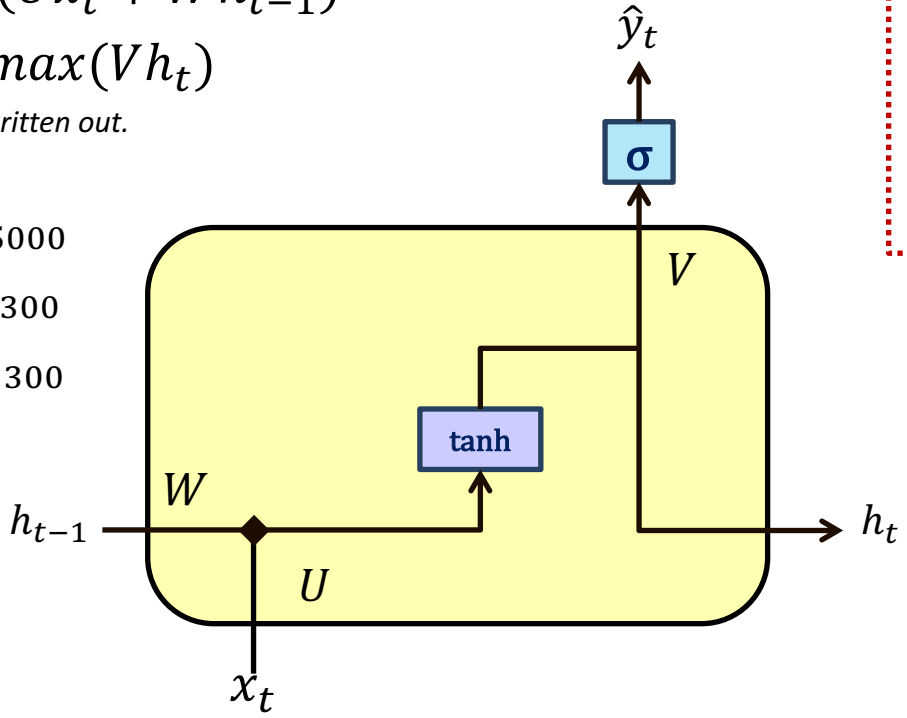
# Gradient Vanishing/Exploding

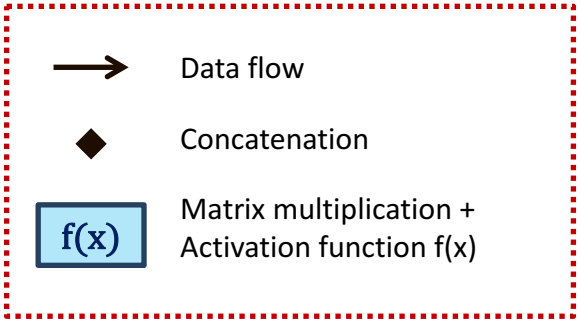
- When the chain of back propagation is very long, so many times of **multiplication** would be applied, somehow repeated (e.g. in $W^T$).
  - **Gradient Vanishing**
    - … when the values are smaller than 1.
    - … the model only learns how to keep “short memory”.
    - … no good solution found with vanilla RNN.
  - **Gradient Exploding**
    - … when the values are greater than 1.
    - … could be partially solved by setting a threshold to “clip” the gradient.

# Vanilla RNN

- $h_t = tanh(Ux_t + Wh_{t-1})$
- $\hat{y}_t = softmax(Vh_t)$
- ** Bias 'b' is NOT written out.*

- $U \in \mathbb{R}^{300\times5000}$
- $W \in \mathbb{R}^{300\times300}$
- $V \in \mathbb{R}^{5000\times300}$

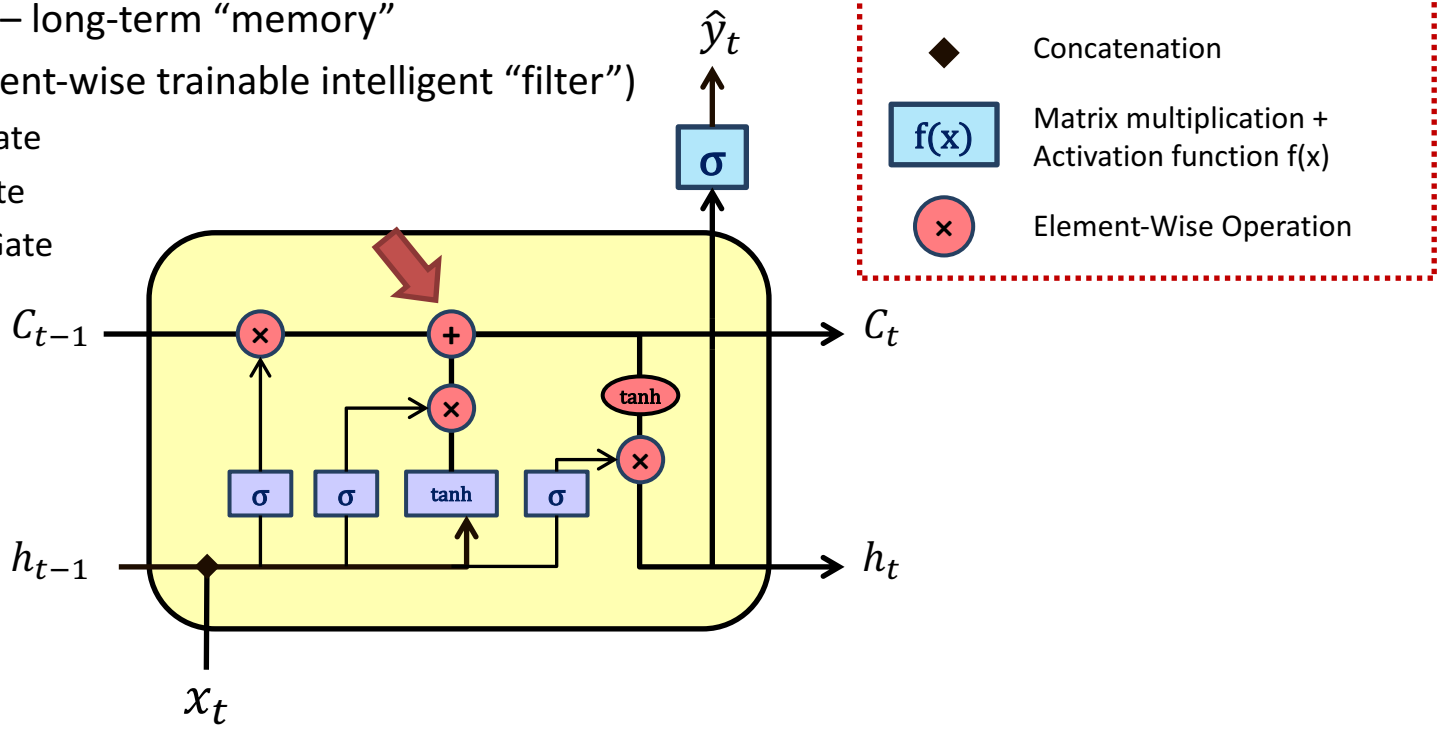


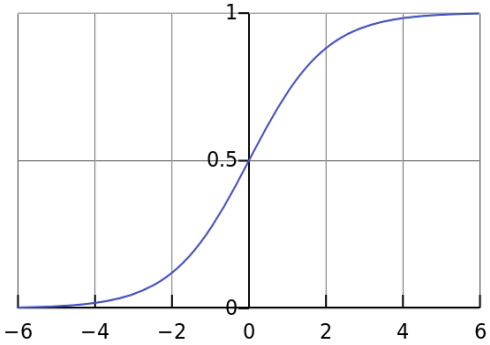
# LSTM

Hochreiter & Schmidhuber (1997)

- **L**ong **S**hort **T**erm **M**emory
  - Cell State $C_t$ – long-term "memory"
  - Gates: (element-wise trainable intelligent "filter")
    - Forget Gate
    - Input Gate
    - Output Gate

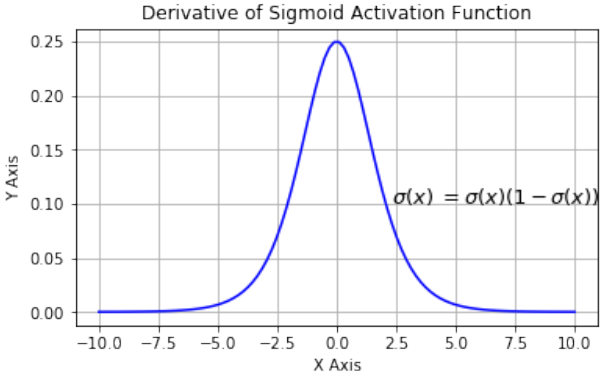


# Sigmoid Function



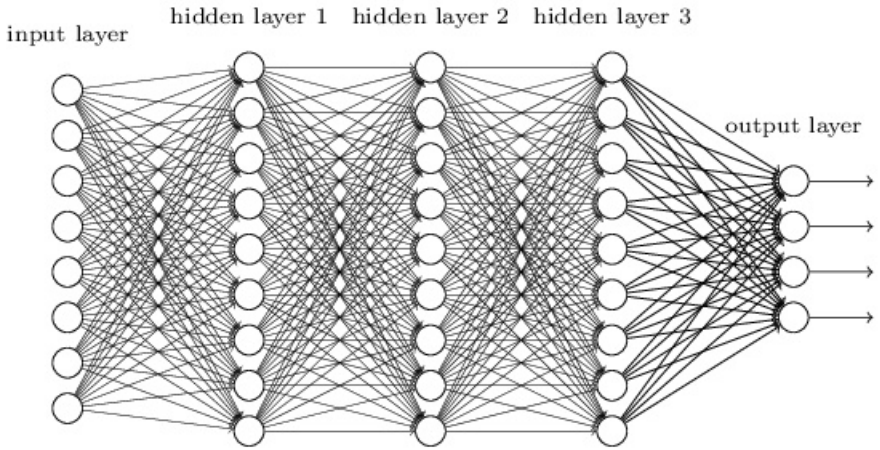

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

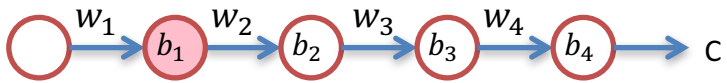
```
def sigmoid(x):
    return 1.0 / (1.0 + math.exp(-x))
```




$$\sigma'(x) = \sigma\,(x)\big(1 - \sigma\,(x)\big)$$

# Chaining

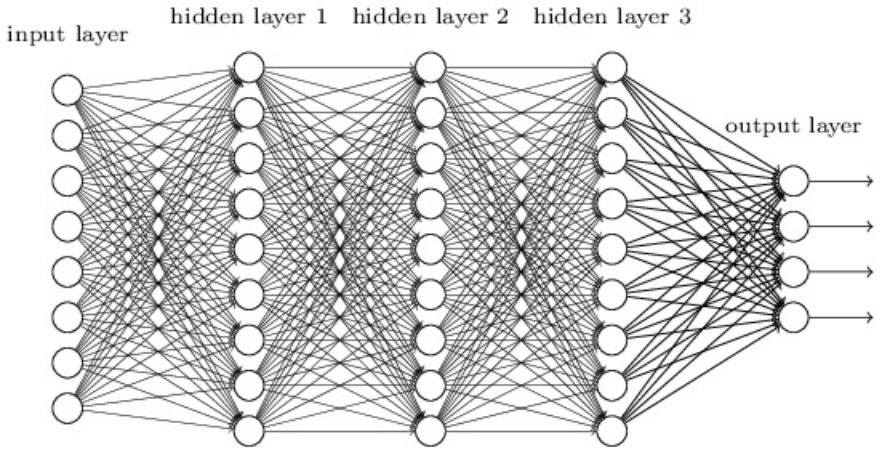


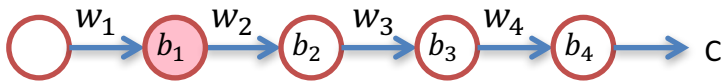




$$y = x_{i+1} = \sigma(z_i) = \sigma(w_i x_i + b_i),\ where\ z_i = w_i x_i + b_i$$

$$\frac{\partial C}{\partial b_1} = \frac{\partial C}{\partial y_4} = \frac{\partial C}{\partial y_4}$$

# Chaining



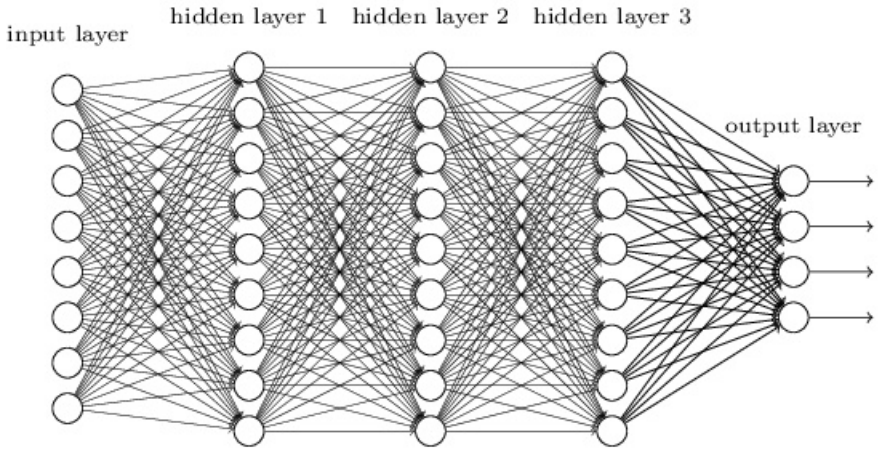




$$y = x_{i+1} = \sigma(z_i) = \sigma(w_i x_i + b_i),\ where\ z_i = w_i x_i + b_i$$

$$\frac{\partial C}{\partial b_1} = \frac{\partial C}{\partial y_4}\frac{\partial y_4}{\partial z_4} = \frac{\partial C}{\partial y_4}\sigma'(z_4)$$

$$y_4 = \sigma(z_4)$$

# Chaining



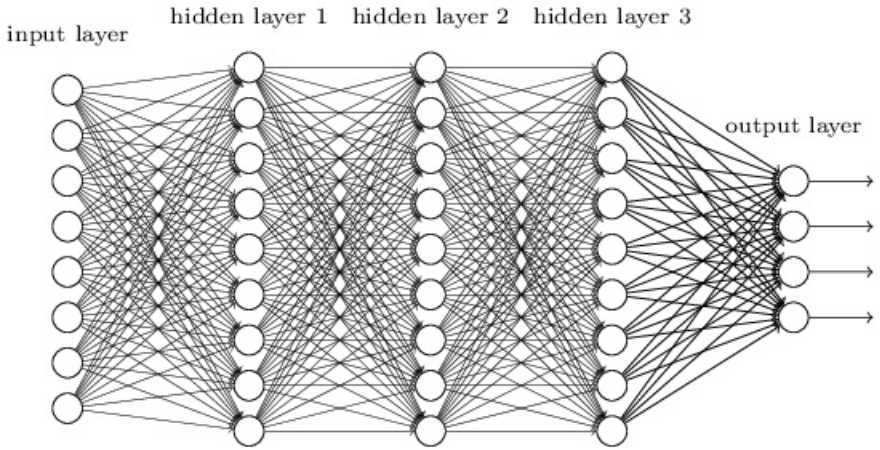


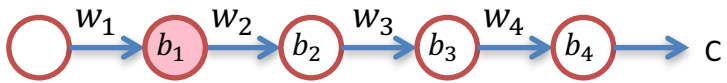
$$y = x_{i+1} = \sigma(z_i) = \sigma(w_i x_i + b_i),\ where\ z_i = w_i x_i + b_i$$

$$\frac{\partial C}{\partial b_1} = \frac{\partial C}{\partial y_4}\frac{\partial y_4}{\partial z_4}\frac{\partial z_4}{\partial x_4} = \frac{\partial C}{\partial y_4}\sigma'(z_4)w_4$$

$$\frac{\partial z_4}{\partial x_4} = \frac{\partial(w_4 x_4 + b_4)}{\partial x_4}$$

# Chaining



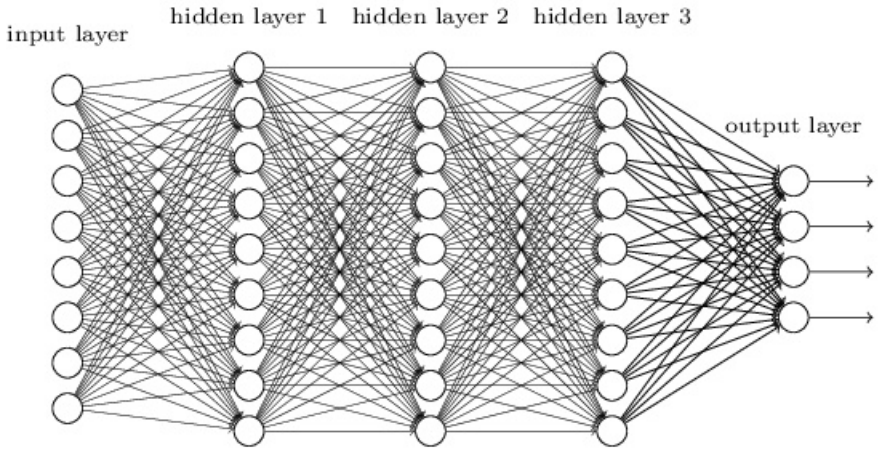

$$y = x_{i+1} = \sigma(z_i) = \sigma(w_i x_i + b_i),\ where\ z_i = w_i x_i + b_i$$

$$\frac{\partial C}{\partial b_1} = \frac{\partial C}{\partial y_4}\frac{\partial y_4}{\partial z_4}\frac{\partial z_4}{\partial x_4}\frac{\partial x_4}{\partial z_3}$$

$$= \frac{\partial C}{\partial y_4}\sigma'(z_4)w_4\sigma'(z_3)$$

$$x_4 = \sigma(z_3)$$

# Chaining





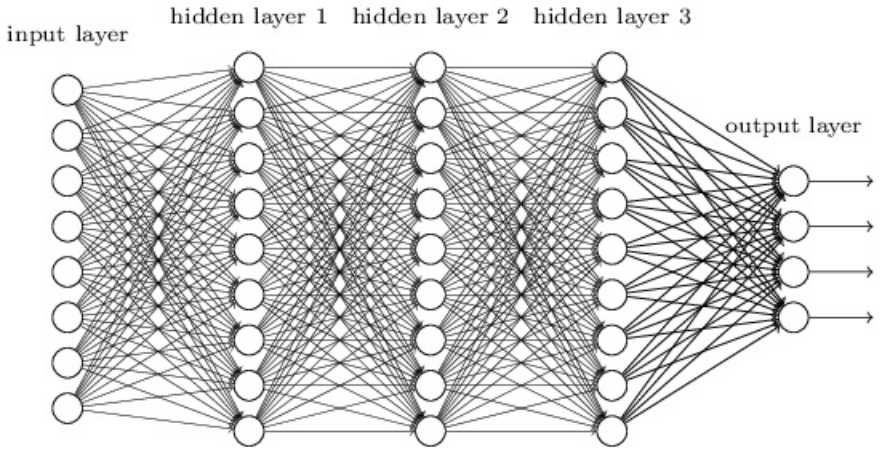
$$y = x_{i+1} = \sigma(z_i) = \sigma(w_i x_i + b_i),\ where\ z_i = w_i x_i + b_i$$

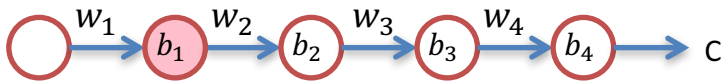
$$\frac{\partial C}{\partial b_1} = \frac{\partial C}{\partial y_4}\frac{\partial y_4}{\partial z_4}\frac{\partial z_4}{\partial x_4}\frac{\partial x_4}{\partial z_3}\frac{\partial z_3}{\partial x_3}$$

$$= \frac{\partial C}{\partial y_4}\sigma'(z_4)w_4\sigma'(z_3)w_3$$

$$\frac{\partial z_3}{\partial x_3} = \frac{\partial(w_3 x_3 + b_3)}{\partial x_3}$$

# Chaining



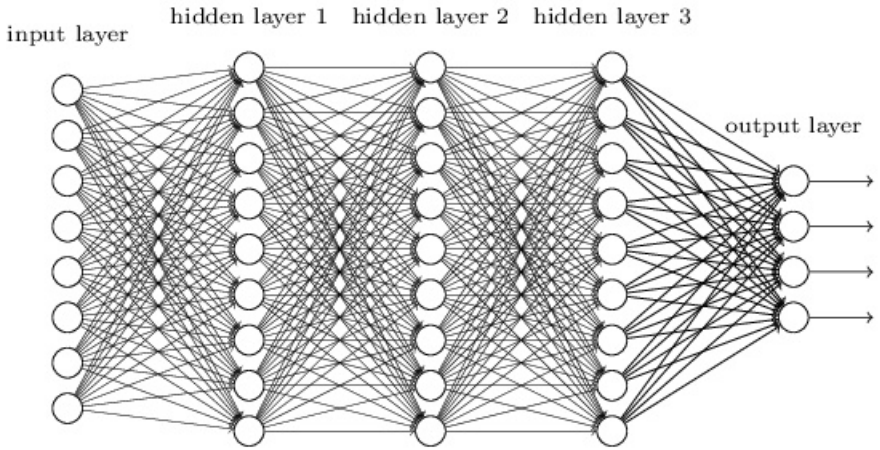

$$y = x_{i+1} = \sigma(z_i) = \sigma(w_i x_i + b_i),\ where\ z_i = w_i x_i + b_i$$

$$\frac{\partial C}{\partial b_1} = \frac{\partial C}{\partial y_4}\frac{\partial y_4}{\partial z_4}\frac{\partial z_4}{\partial x_4}\frac{\partial x_4}{\partial z_3}\frac{\partial z_3}{\partial x_3}\frac{\partial x_3}{\partial z_2}$$

$$= \frac{\partial C}{\partial y_4}\sigma'(z_4)w_4\sigma'(z_3)w_3\sigma'(z_2)$$

$$x_3 = \sigma(z_2)$$

# Chaining





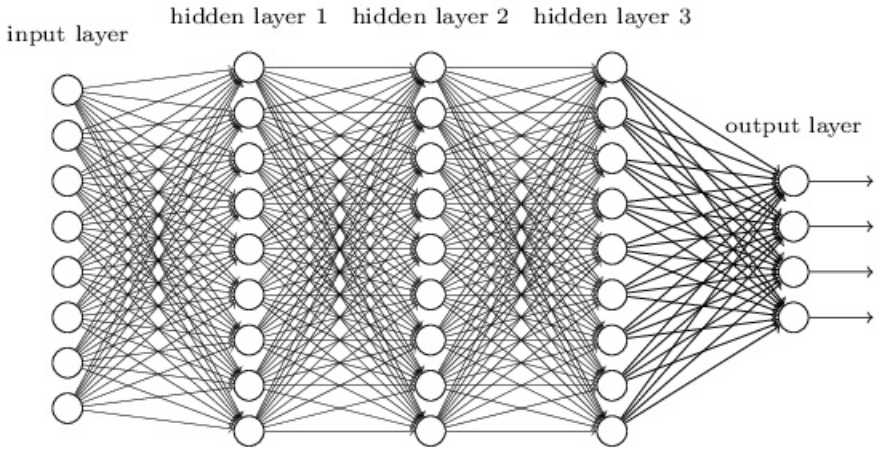
$$y = x_{i+1} = \sigma(z_i) = \sigma(w_i x_i + b_i),\ where\ z_i = w_i x_i + b_i$$

$$\frac{\partial C}{\partial b_1} = \frac{\partial C}{\partial y_4}\frac{\partial y_4}{\partial z_4}\frac{\partial z_4}{\partial x_4}\frac{\partial x_4}{\partial z_3}\frac{\partial z_3}{\partial x_3}\frac{\partial x_3}{\partial z_2}\frac{\partial z_2}{\partial x_2}$$

$$= \frac{\partial C}{\partial y_4}\sigma'(z_4)w_4\sigma'(z_3)w_3\sigma'(z_2)w_2$$

$$\frac{\partial z_2}{\partial x_2} = \frac{\partial(w_2 x_2 + b_2)}{\partial x_2}$$

# Chaining





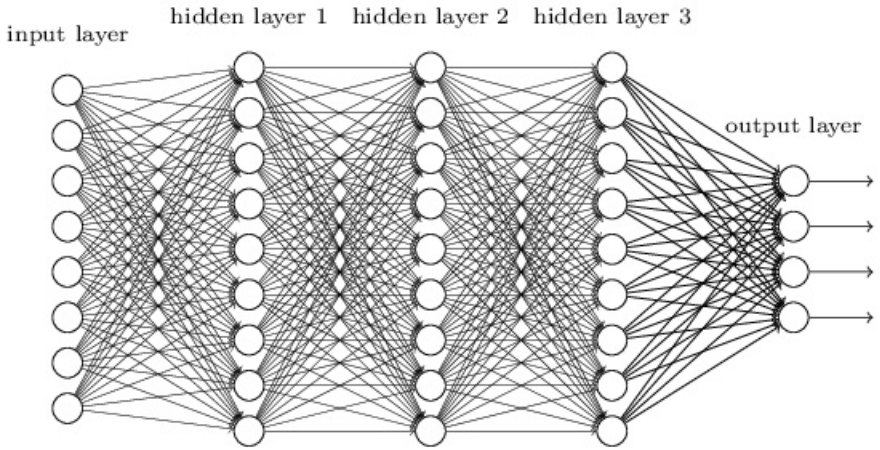
$$y = x_{i+1} = \sigma(z_i) = \sigma(w_i x_i + b_i),\ where\ z_i = w_i x_i + b_i$$

$$\frac{\partial C}{\partial b_1} = \frac{\partial C}{\partial y_4}\frac{\partial y_4}{\partial z_4}\frac{\partial z_4}{\partial x_4}\frac{\partial x_4}{\partial z_3}\frac{\partial z_3}{\partial x_3}\frac{\partial x_3}{\partial z_2}\frac{\partial z_2}{\partial x_2}\frac{\partial x_2}{\partial z_1}$$

$$= \frac{\partial C}{\partial y_4}\sigma'(z_4)w_4\sigma'(z_3)w_3\sigma'(z_2)w_2\sigma'(z_1)$$

$$x_2 = \sigma(z_1)$$

# Chaining





$$y = x_{i+1} = \sigma(z_i) = \sigma(w_i x_i + b_i),\ where\ z_i = w_i x_i + b_i$$

$$\frac{\partial C}{\partial b_1} = \frac{\partial C}{\partial y_4}\frac{\partial y_4}{\partial z_4}\frac{\partial z_4}{\partial x_4}\frac{\partial x_4}{\partial z_3}\frac{\partial z_3}{\partial x_3}\frac{\partial x_3}{\partial z_2}\frac{\partial z_2}{\partial x_2}\frac{\partial x_2}{\partial z_1}\frac{\partial z_1}{\partial b_1}$$

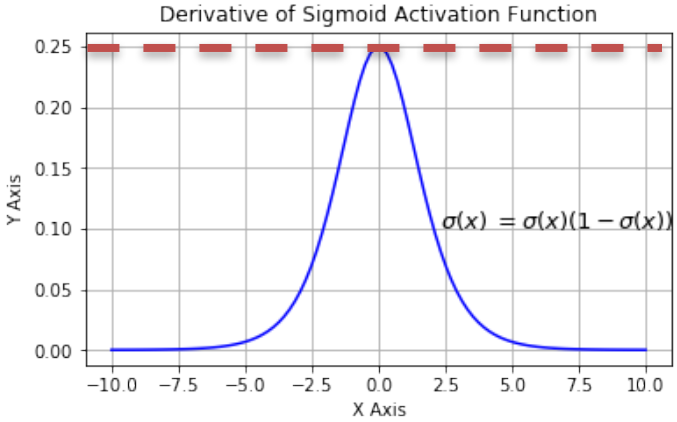
$$= \frac{\partial C}{\partial y_4}\sigma'(z_4)w_4\sigma'(z_3)w_3\sigma'(z_2)w_2\sigma'(z_1) * 1$$

$$\frac{\partial z_1}{\partial b_1} = \frac{\partial(w_1 x_1 + b_1)}{\partial b_1}$$

# Gradient Vanishing Problem

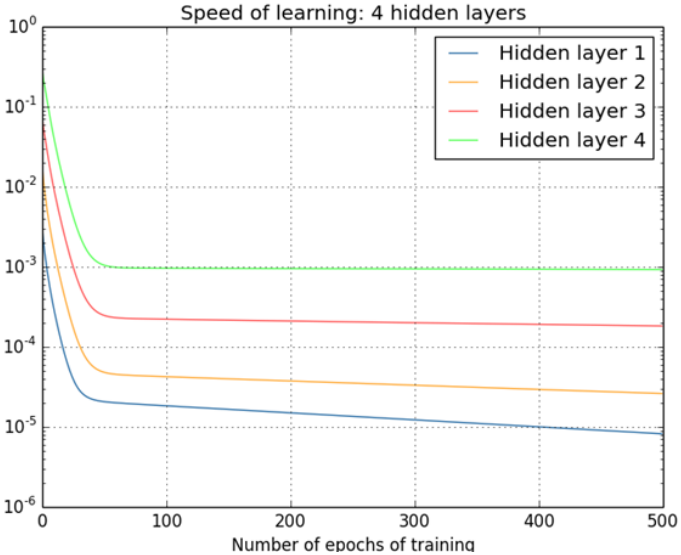

- Maximum of $\sigma'$ is 0.25 and $|w_{init}| < 1$,
- Thus $|w\,\sigma'(z_i)| < 0.25$

$$\frac{\partial C}{\partial b_1} = \frac{\partial C}{\partial y_4}\sigma'(z_4)\underbrace{w_4\sigma'(z_3)}_{<0.25}\underbrace{w_3\sigma'(z_2)}_{<0.25}\underbrace{w_2\sigma'(z_1)}_{<0.25}$$



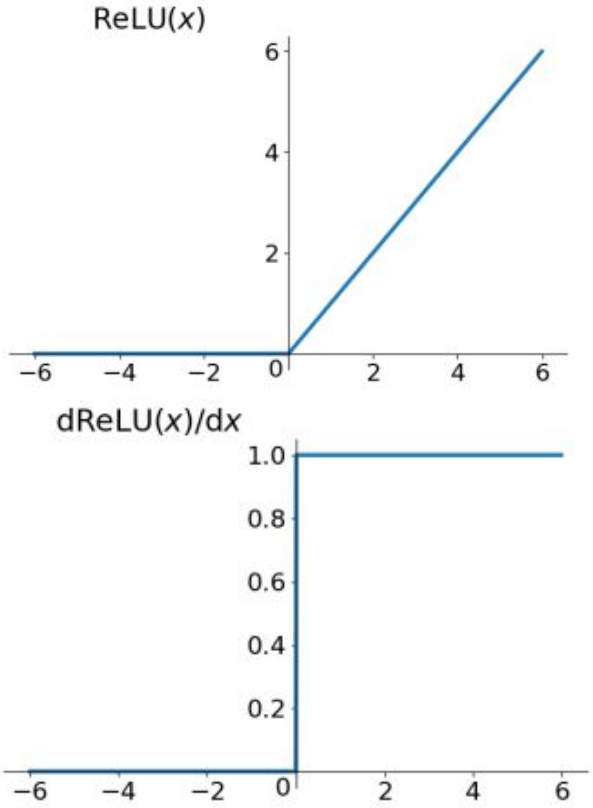
# Gradient Vanishing Problem


- The more front of the layer, the slower the gradient changes
- We were not able to train a very deep network
- How to solve it?



# ReLU (Rectified Linear Unit)

*Rectified linear units improve restricted boltzmann machines*, Nair & Hinton (2010)

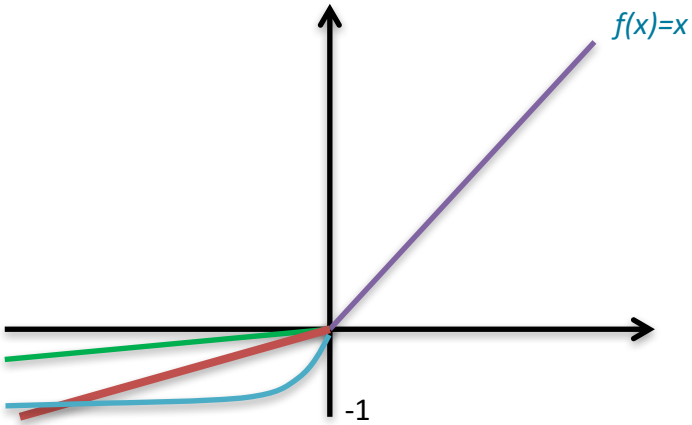



$$f(x) = \max(x, 0)$$

```
# ReLU
def relu(X):
    return np.maximum(X, 0)
```

$$f'(x) = \begin{cases} 0, if\ x < 0 \\ \mathbf{1}, if\ x \geq 0 \end{cases}$$

# Leaky ReLU (LReLU), Parametric ReLU (PReLU), Exponential Linear Unit (ELU)




Leaky ReLU:

$$f(x) = \begin{cases} 0.01x, & if\ x < 0 \\ x, & if\ x \geq 0 \end{cases}$$

$$f'(x) = \begin{cases} 0.01, & if\ x < 0 \\ 1, & if\ x \geq 0 \end{cases}$$

PReLU:

$$f(x) = \begin{cases} ax, & if\ x < 0 \\ x, & if\ x \geq 0 \end{cases}$$

$$f'(x) = \begin{cases} a, & if\ x < 0 \\ 1, & if\ x \geq 0 \end{cases}$$

ELU:

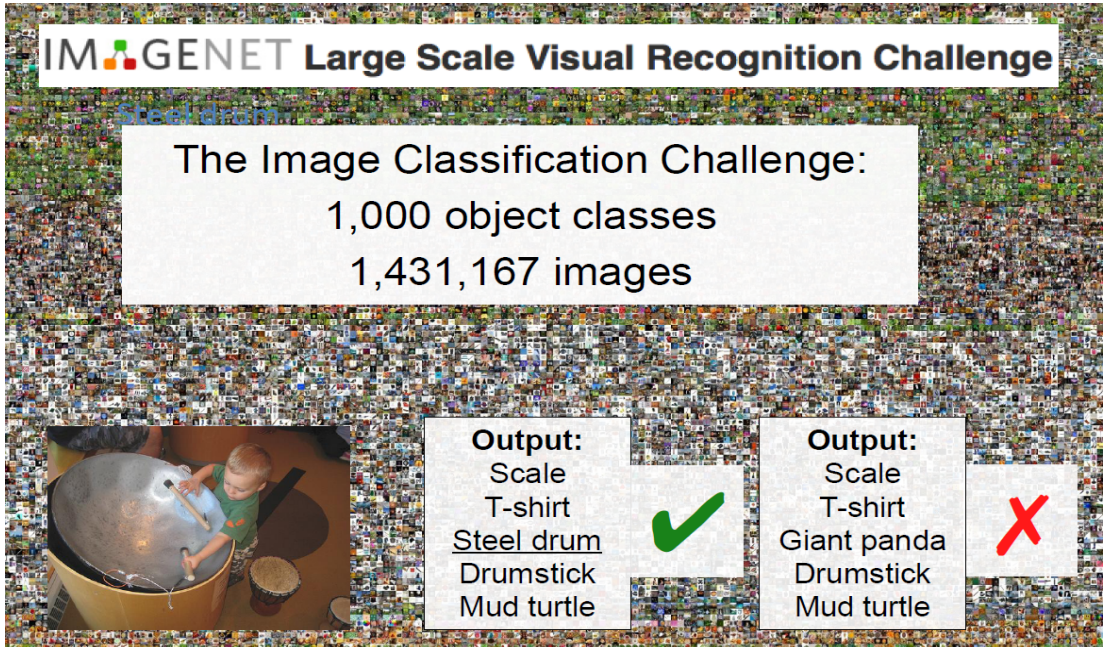
$$f(x) = \begin{cases} x, & if\ x \geq 0 \\ \alpha(e^x - 1), & if\ x < 0 \end{cases}$$

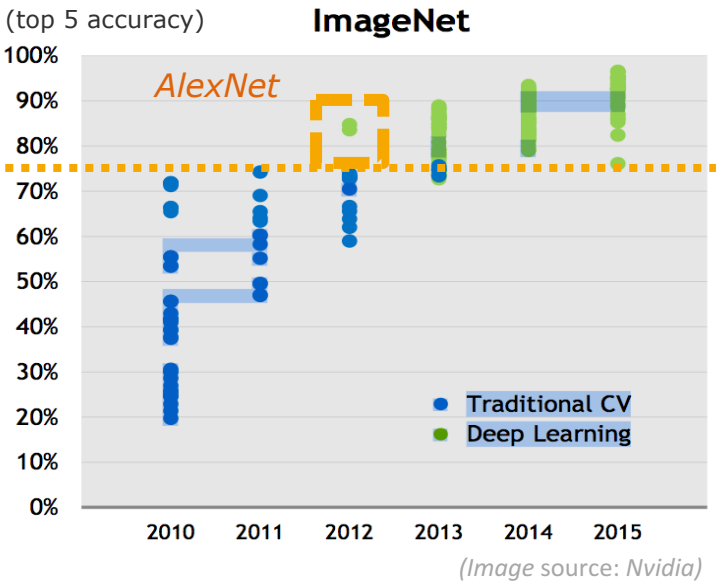
$$f'(x) = \begin{cases} 1, & if\ x \geq 0 \\ f(x) + \alpha, & if\ x < 0 \end{cases}$$

*Maas et al. ICML'13, He et al. ICCV'15, Clevert et al. CoRR 2015*

# ImageNet Challenge 2012

Given an image, classify what is depicted



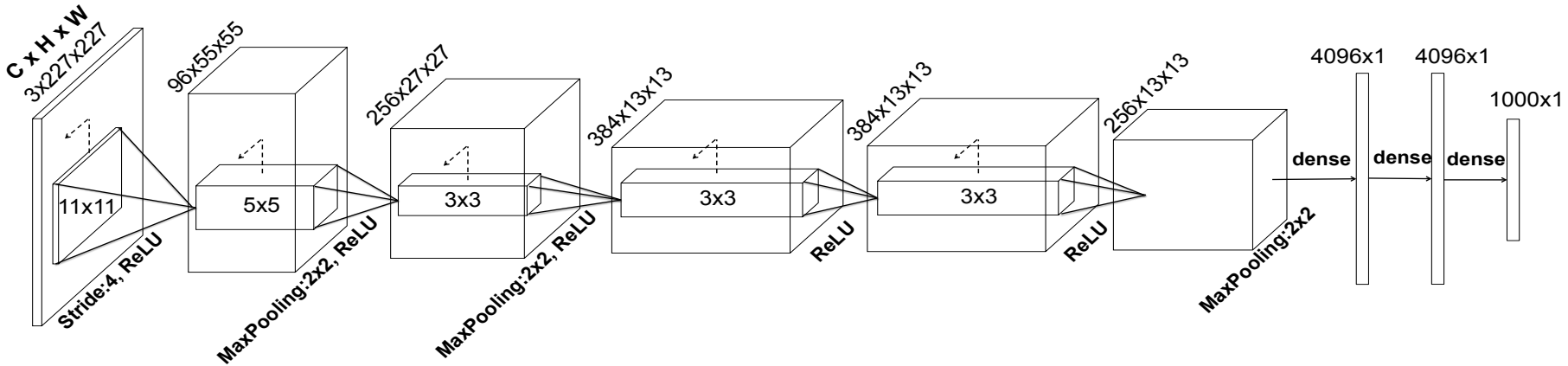

*(Image source: Nvidia)*

(Deng, J. et al. 2014)

# AlexNet

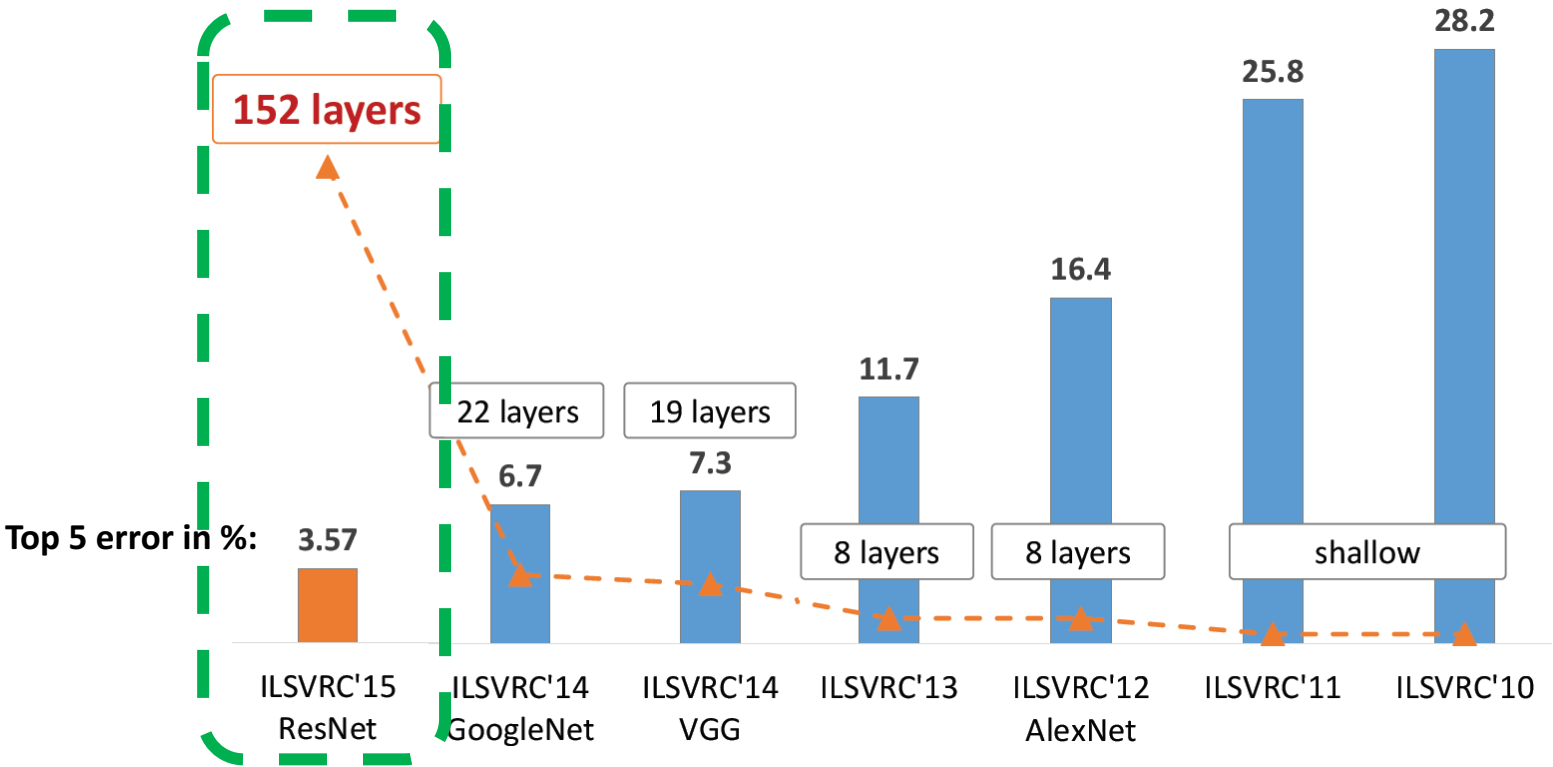
*Imagenet classification with deep convolutional neural network*, Krizhevsky, Sutskever, Hinton. (NIPS'12)

- 62.3 million parameters
  - Conv: 3.7 million (6%) , FC: 58.6 million (94% )
- Computation: Conv (95%) , FC (5%)
- The network takes 90 epochs in **6** days to train on two GTX 580 GPUs




WE NEED TO GO DEEPER

# ImageNet Challenge Winners



# ResNet

*Deep residual learning for image recognition,* He et al. (CVPR'16, best paper award)

- What happens when we continue stacking layers on a "plain" convolutional neural network like AlexNet?

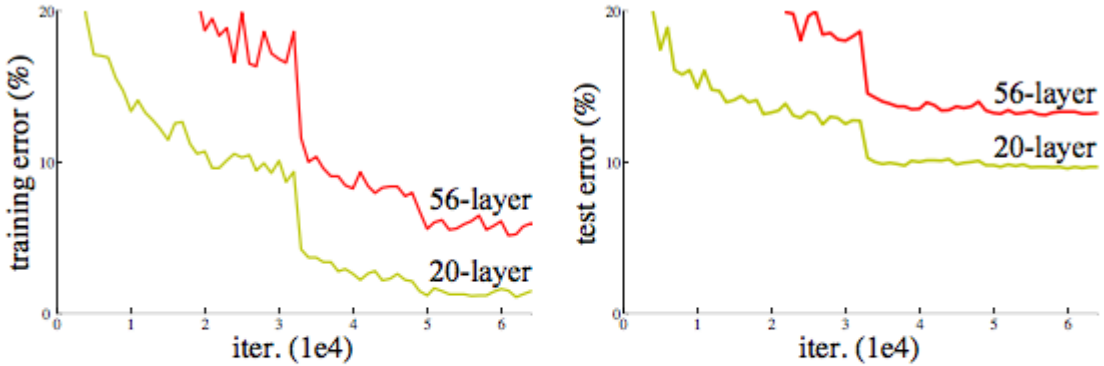


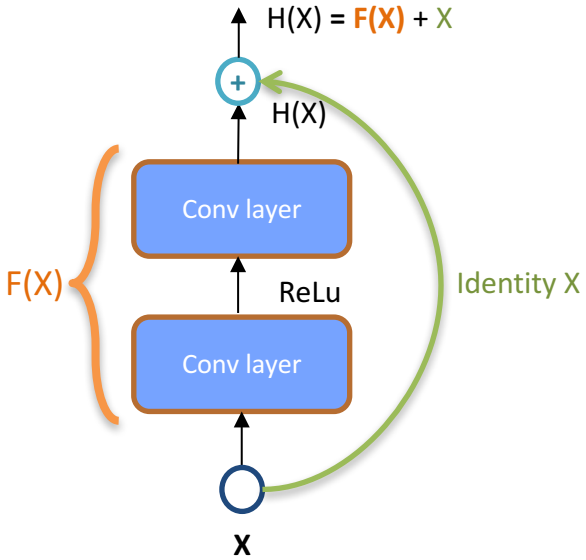

Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

- The deeper model performs worse, overfitting?
- **Network degradation**

# ResNet

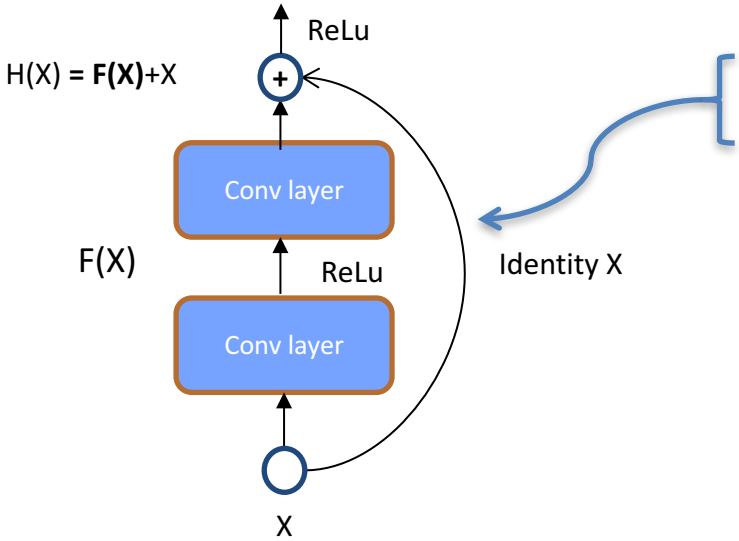
*Deep residual learning for image recognition,* He et al. (CVPR'16)

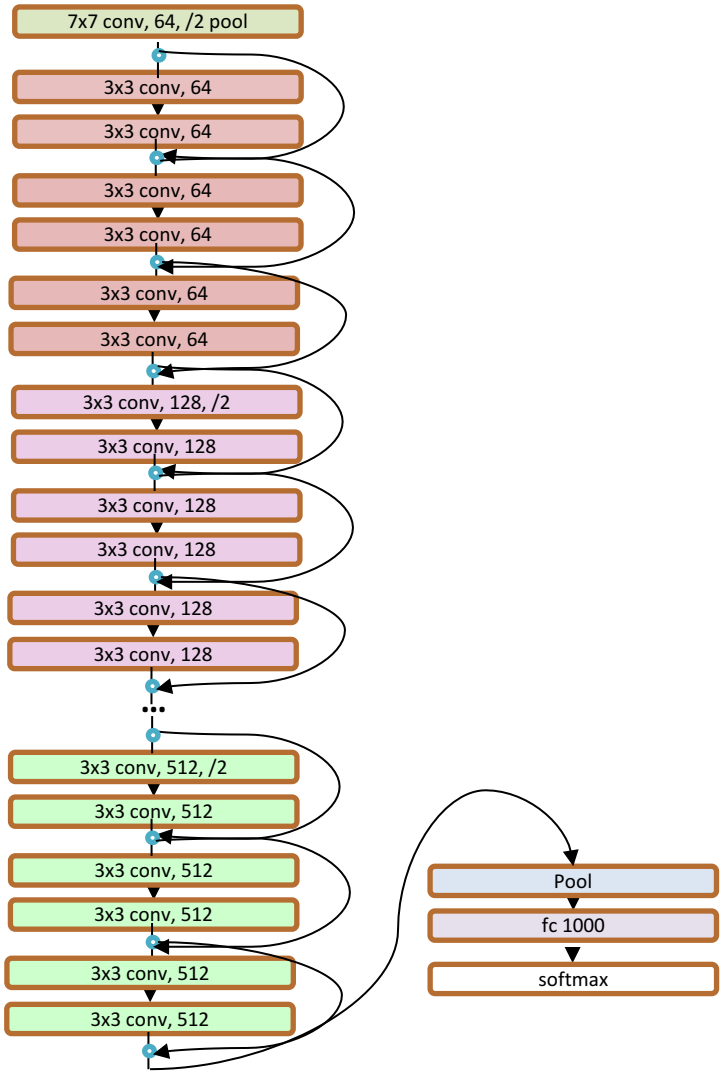
- Intuition: learn F(X) not H(X) in the **residual block**
  - If F(X) = 0, then H(X) is an identity mapping
  - Use layers in residual block to learn the residual function **F(X)** = H(X) - X instead of learning H(X)
  - Learning F(X) should be easier
  - **Better gradient flow**



# ResNet

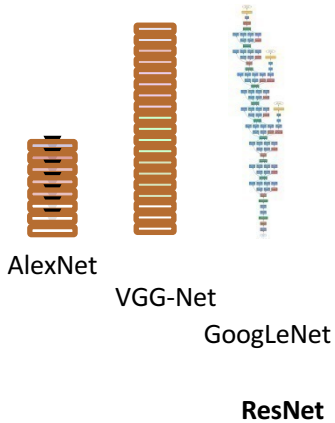
- Stacked residual blocks
- At least two conv layers in each residual block
- Periodically, double filters and downsample spatially using stride 2



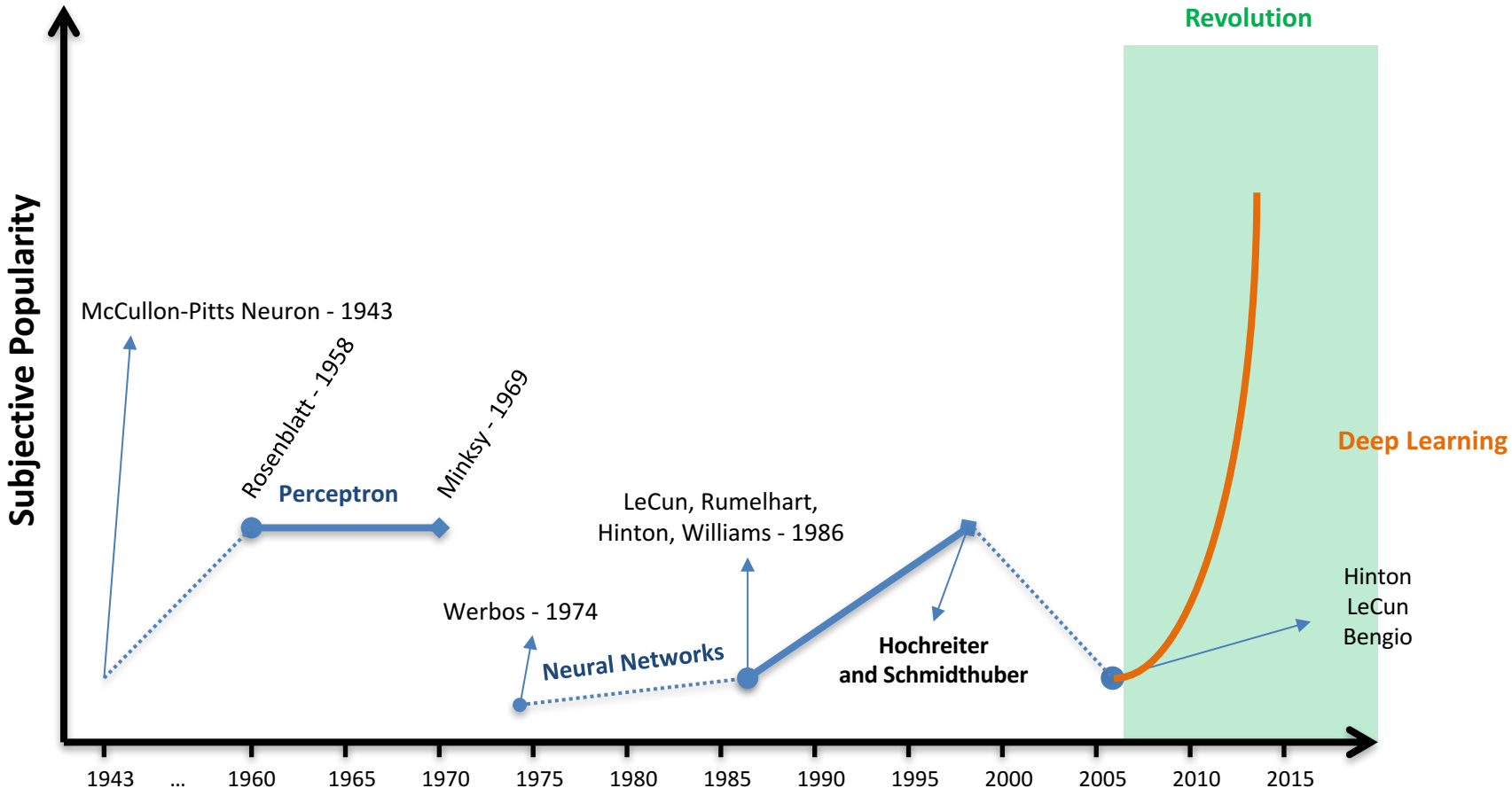
# ResNet

- 152-layer on ImageNet, 1202-layer on CIFAR data set
- Swept 1st place in all ILSVRC and COCO 2015 competitions
  - Including *image classification, object detection* and *object segmentation*
- ILSVRC'15 classification winner (**3.57%** top 5 error) → better than "human performance!" (**5.1%** reported in *Russakovsky 2014*)
- Contribution
  - Preventing network degradation problem with residual connections
  - We are able to train extreme deeper networks

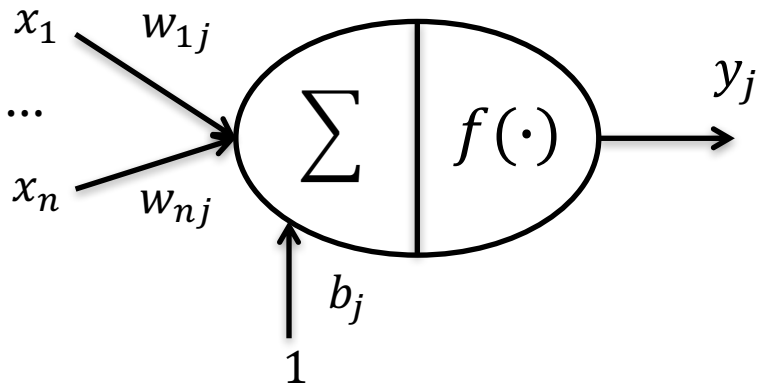


# Recent Efforts

- **Recent efforts more based on Information/Gradient Flow**
  - Sigmoid saturation, gradient vanishing problem : **ReLU** (6003 citations)
  - If x<0, ReLU=0 : **LeakyReLU, PReLU, ELU** etc. (5334 citations in total)
  - Network too “deep”, gradient vanishing : **highway** (parameterized shortcut connection between layers) (779 citations)
  - Simplified highway : **ResNet** (shortcut connection without params) (**23397** citations)
  - Forced stability of the mean and variance of activations: **BatchNorm** (10717)
  - Adding noises in gradient flow : **Dropout** (12387)
  - RNN with gradient exploding and vanishing : **LSTM** (adding more gated controls) (18907)
  - Simplified LSTM : **GRU** (gated recurrent unit) (2664)
  - etc.

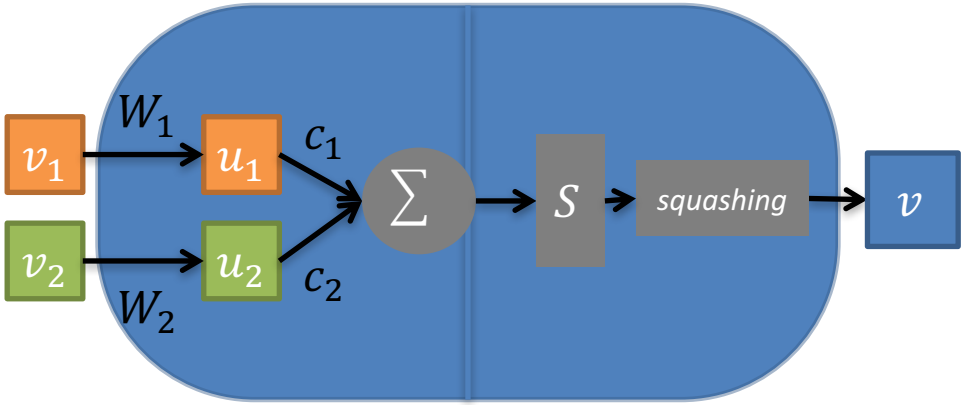

Revolution
Subjective Popularity
McCullon-Pitts Neuron - 1943
Rosenblatt - 1958
Perceptron
Minksy - 1969
LeCun, Rumelhart,
Hinton, Williams - 1986
Deep Learning
Werbos - 1974
Neural Networks
Hochreiter
and Schmidthuber
Hinton
LeCun
Bengio
1943
...
1960
1965
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015

# Next week: Capsule *vs.* Classical NN




Classical neuron: **scalar to scalar**

$$y_j = \sum_{i=1}^{n} w_{ij} x_i + b_j$$



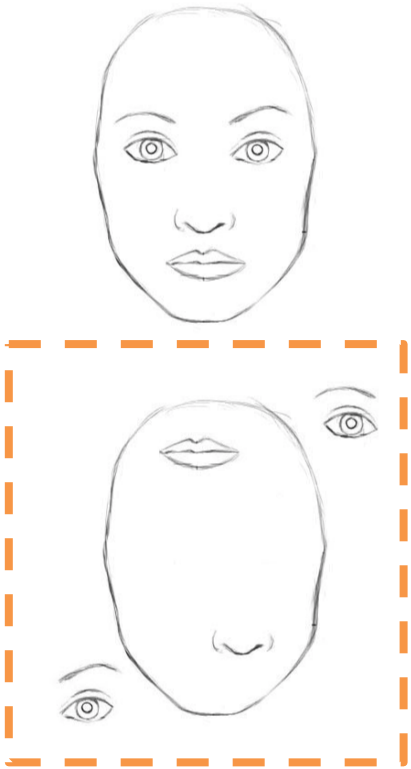

Capsule neuron: **vector to vector**

$$u_i = W_i v_i \qquad s = \sum_{i=1}^{n} c_i u_i$$

$$v = Squashing(s) = \frac{\|s\|^2}{1 + \|s\|^2} \frac{s}{\|s\|}$$

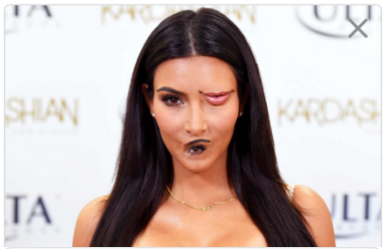
*Dynamic routing between capsules*, Sabour, Frosst & Hinton (NIPS'17)

# Limitations of ConvNets



| person | 0.88 |
|---|---|

| | |
|---|---|
| reddish orange color | 0.78 |
| light brown color | 0.78 |
| starlet | 0.66 |
| entertainer | 0.66 |
| female | 0.60 |
| woman | 0.59 |
| young lady (heroine) | 0.59 |



| person | 0.90 |
|---|---|

| | |
|---|---|
| light brown color | 0.84 |
| starlet | 0.77 |
| entertainer | 0.77 |
| female | 0.65 |
| woman | 0.64 |
| young lady (heroine) | 0.64 |
| reddish orange color | 0.64 |
| newsreader | 0.50 |

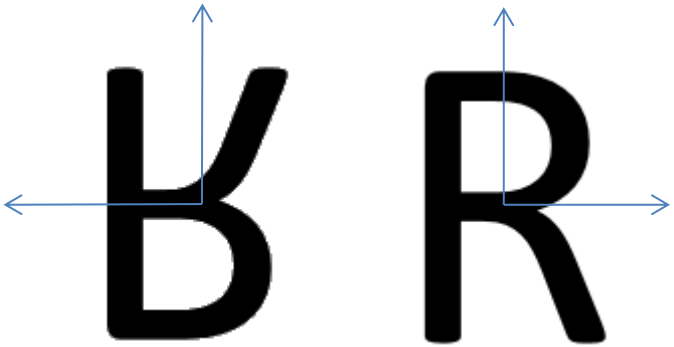
| coal black color | 0.79 |
|---|---|

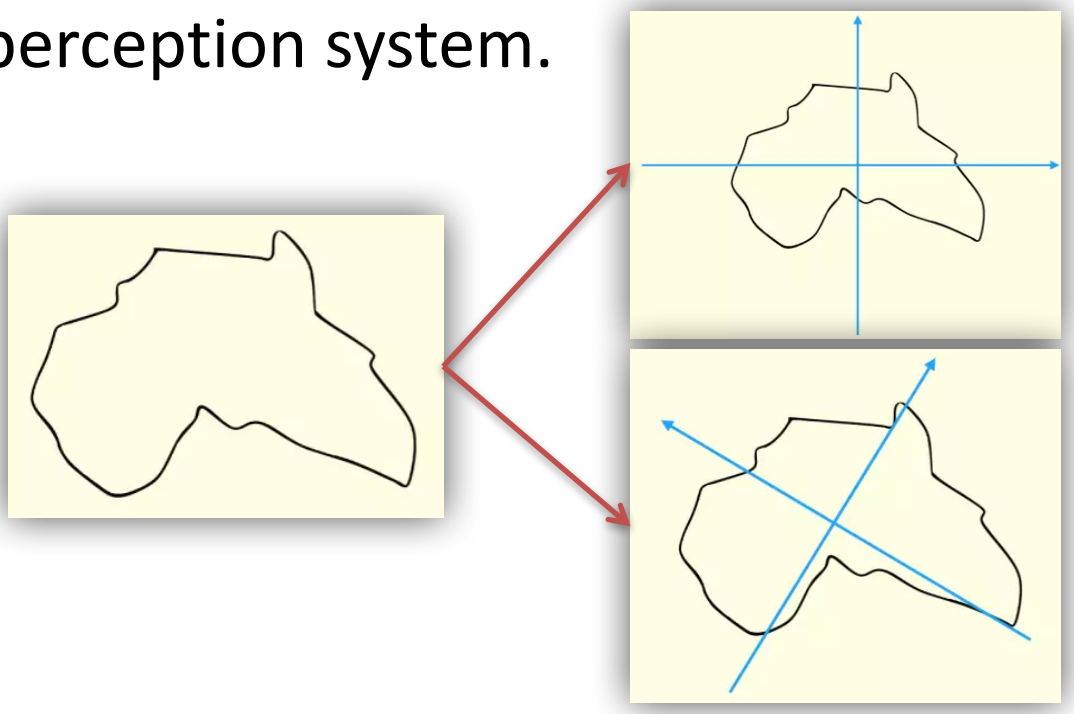
| | |
|---|---|
| hairpiece (hair) | 0.71 |
| dress | 0.71 |
| maroon color | 0.71 |
| person | 0.58 |
| toupee (hairpiece) | 0.58 |
| woman | 0.56 |
| Earrings | 0.55 |
| female | 0.50 |

*Source: freecodecamp*

# Example

- Human vision has **coordinate frame**, which influences the perception system.





# Capsule

Sabour, Frosst & Hinton (NIPS'17, ICLR'18)

## Keypoints

- Using vector (matrix) instead of scalar
- Encapsulate (visual) entity with its pattern
- Equivariance instead of Invariance:
  - Transform(Represent($x$)) = Represent(Transform($x$))
  - Invariance e.g., CNN: Represent($x$) = Represent(Transform($x$))
- Routing by agreement instead of pooling
- Using Matrix to represent the relationships of visual entities

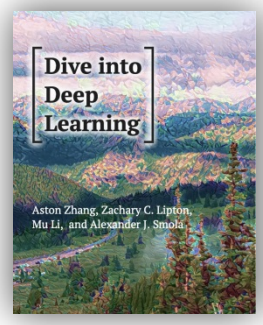
# Literature

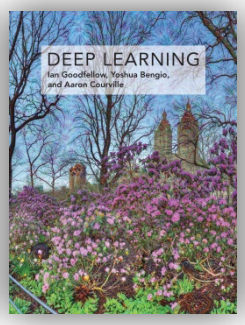
**Books**

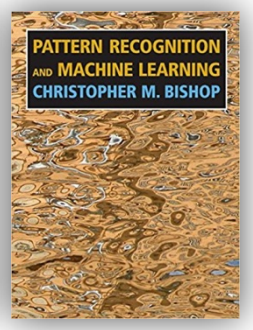
- Alex Smola et al., “Dive into deep learning”, 2019
- Ian J. Goodfellow and Yoshua Bengio and Aaron Courville, “Deep Learning”, 2016
- Christopher M. Bishop “Pattern Recognition and Machine Learning” (google it)

**Online courses:**

- cs231n: Convolutional Neural Networks for Visual Recognition (Feifei Li et al.)
- Deep Learning courses at Coursera (Andrew Ng et al.)




Thank you for your Attention!
Dr. Haojin Yang
Office: H-1.22
Email: haojin.yang@hpi.de