Introduction to Deep Learning

Stefan Lattner

About me

- B.Sc in Mediatechnology and -Design in Hagenberg, Austria (2010)
- Master in JKU Linz, Pervasive Computing (2014)
- PhD in JKU Linz (2018)
    - Austrian Research Institute for AI, Vienna
    - Computational Perception Institute, Linz
- Sony CSL Paris (since 2018)

Objectives

- Understanding of basic working of Neural Networks
- Training procedure (Backpropagation)
- Intuitive understanding of training dynamics and problems
  - Improvements to counter problems
- Historical evolution of Neural Networks

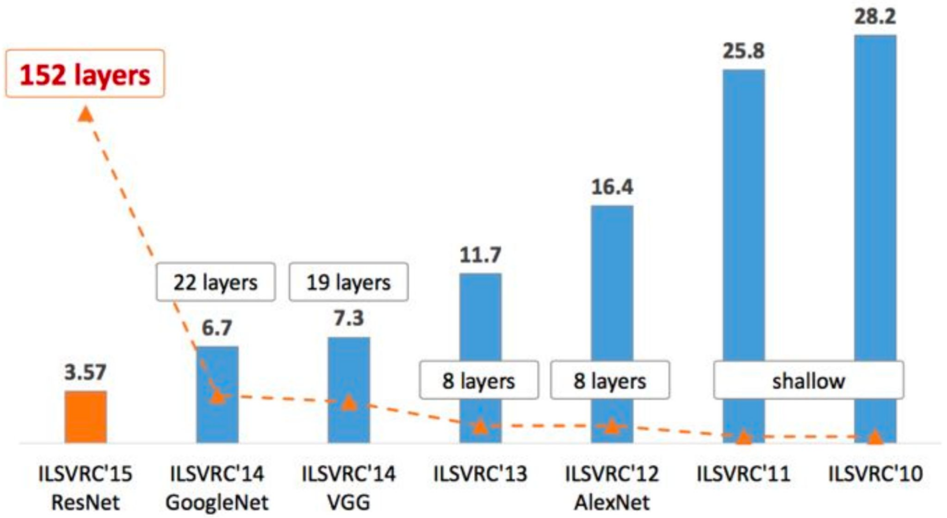
Deep Learning

- Meanwhile umbrella term for whole field of Neural Networks
- Attributed to Geoffrey Hinton, but was already used earlier
- Became popular as Neural Networks actually became deep
  - approx. 2010 - 2015

Geoffrey Hinton

Deep Learning

- Became popular as Neural Networks actually became deep
  - approx. 2010 - 2015



Neural Network Evolution - Milestones

- Perceptron (1957)
- Multi-Layer Perceptron (1958)
- Convolutional Neural Networks (1982/1998)
- Recurrent Neural Networks (1982/1986/1990)
- Backpropagation (1986)
- Long-short Term Memory Networks (1997)
- Generative Adversarial Networks (2014)
- Transformers (2017)
- Diffusion Models (2021)

Neural Network Evolution

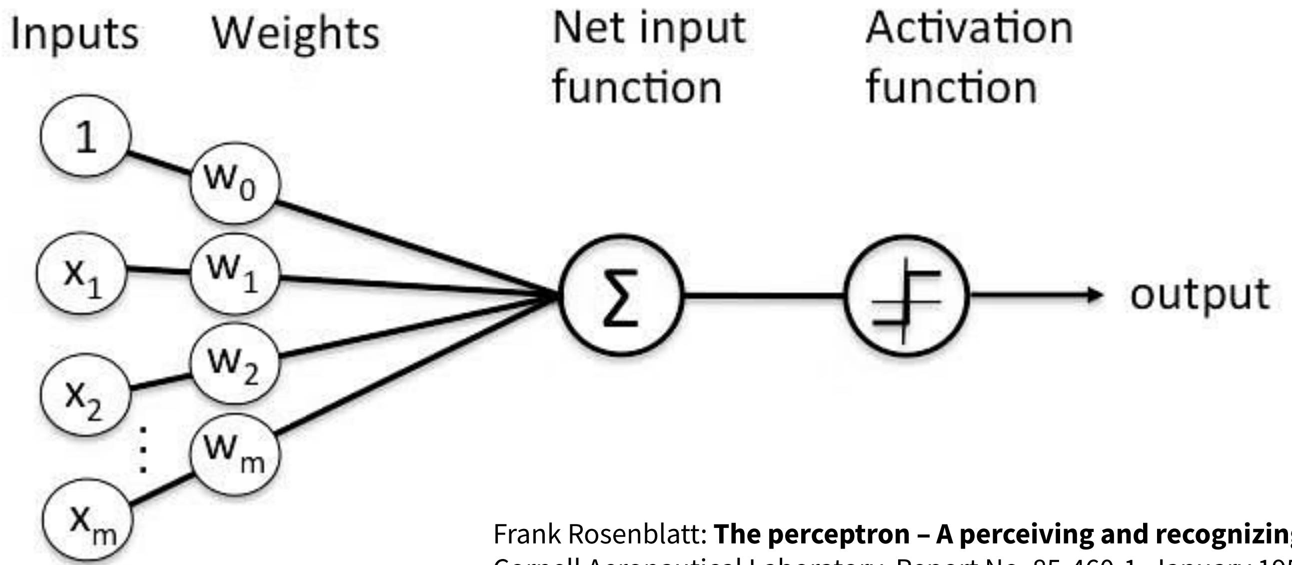
Part 1 (Basics)

Neural Network Evolution - Part 1 (Basics)

- Perceptron (1957)
- Multi-Layer Perceptron (1958)
- Convolutional Neural Networks (1982/1998)
- Backpropagation (1986)
- Going Deeper (2010-2015)

Perceptron

Perceptron (1957)

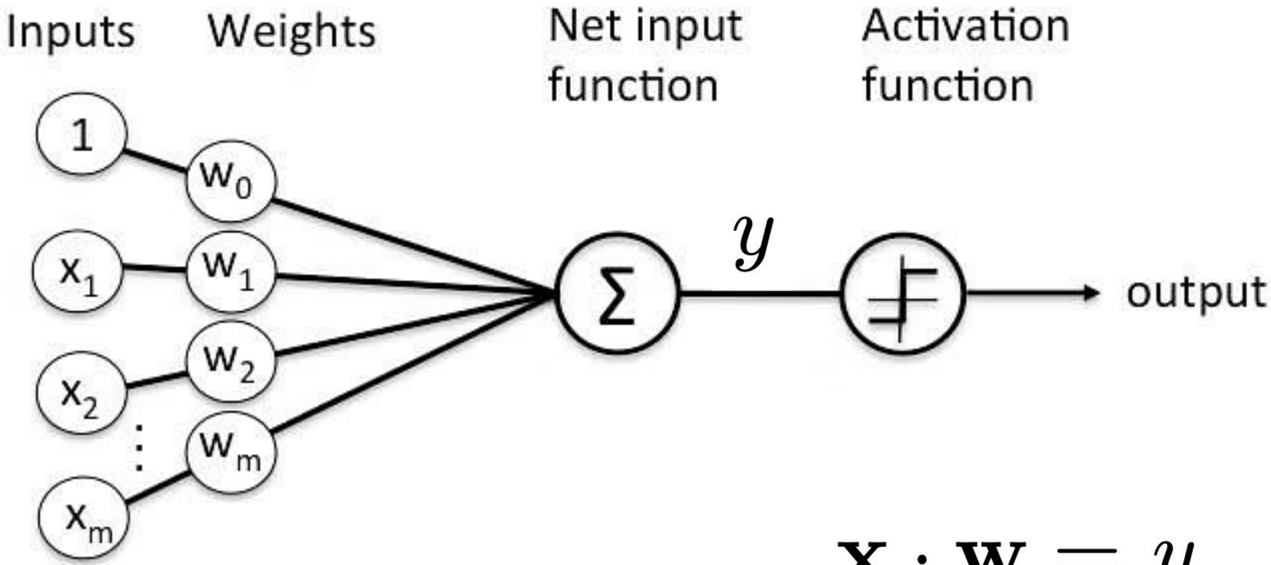


Frank Rosenblatt: **The perceptron – A perceiving and recognizing automaton**. Cornell Aeronautical Laboratory, Report No. 85-460-1, January 1957

$$\mathbf{y} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$$

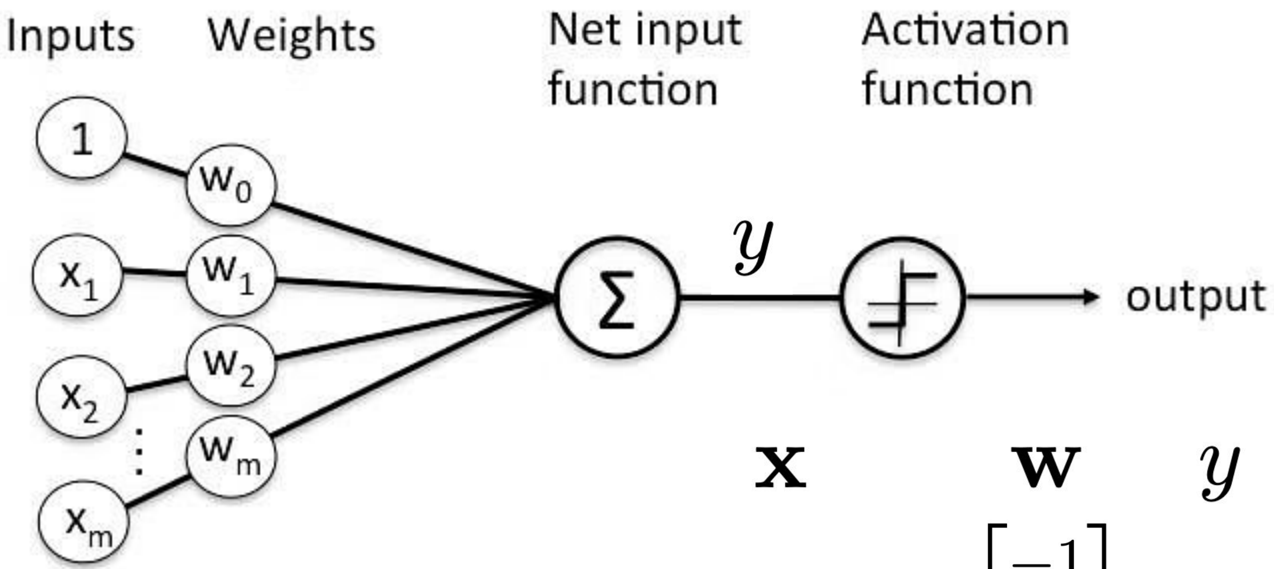
$$\mathbf{y} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$$

Perceptron (1957)



$$\mathbf{x} \cdot \mathbf{w} = y$$

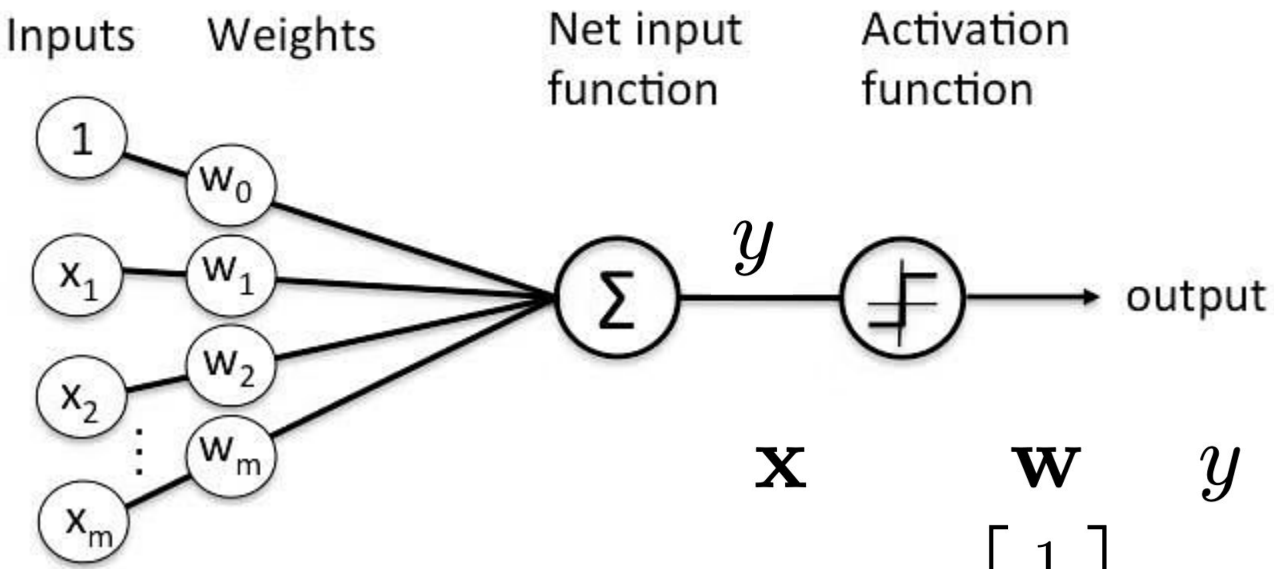
Perceptron (1957)



$$\mathbf{x} \qquad \mathbf{w} \qquad y$$

$$\begin{bmatrix} -1 & 2 & -1 \end{bmatrix} \begin{bmatrix} -1 \\ 2 \\ -1 \end{bmatrix} = 6$$

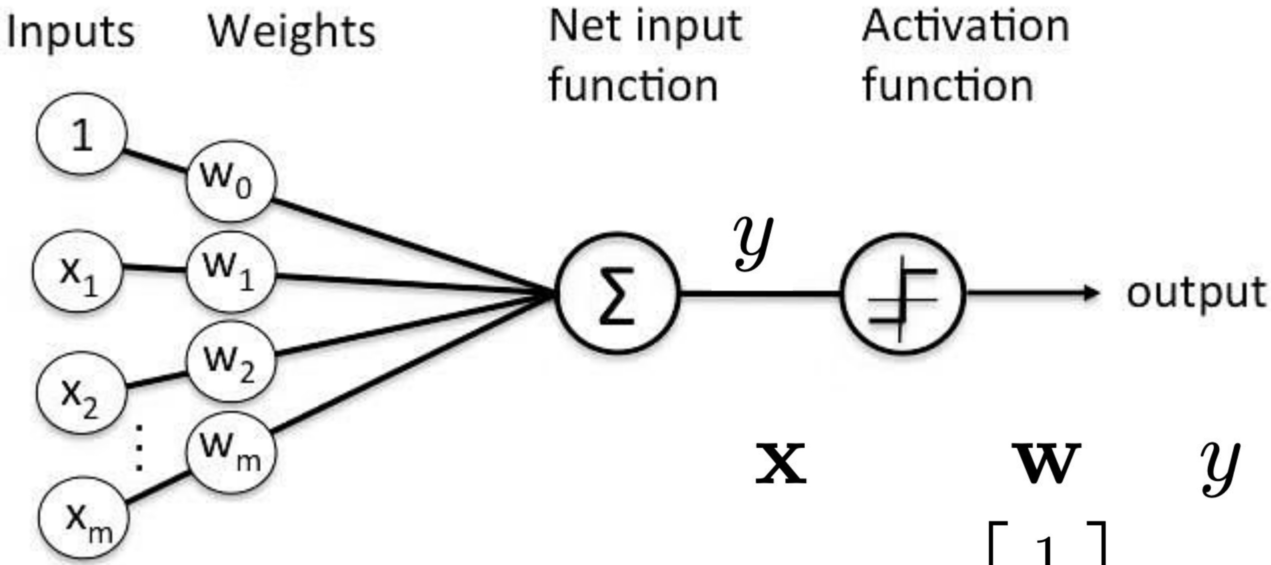
Perceptron (1957)



$$\mathbf{x} \qquad \mathbf{w} \qquad y$$

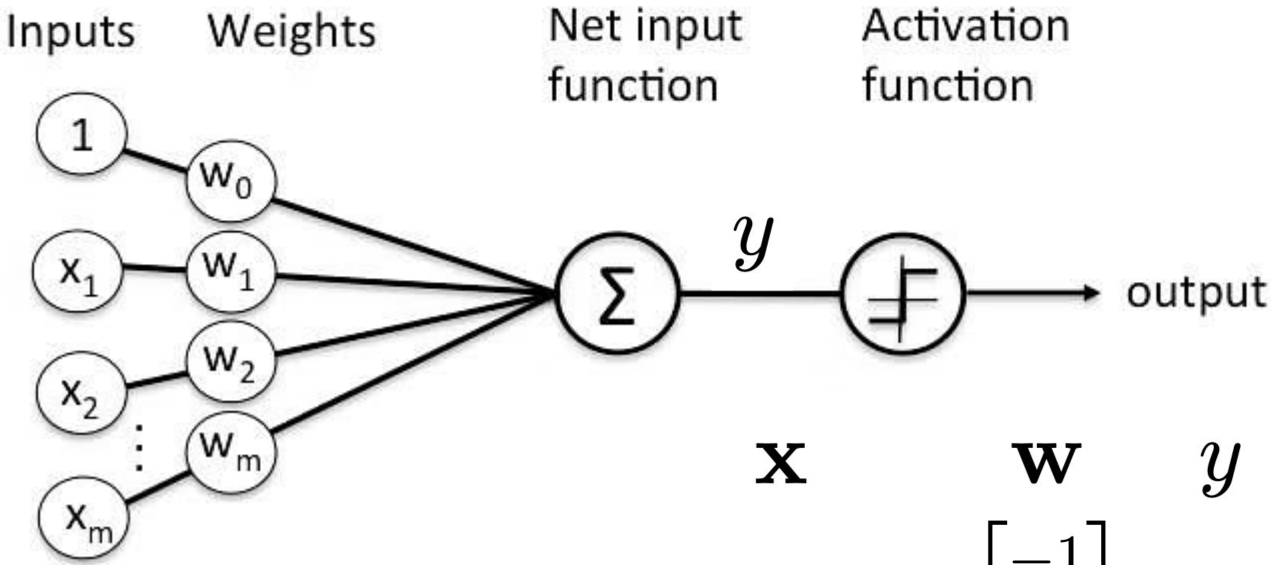
$$\begin{bmatrix} -1 & 2 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix} = -6$$

Perceptron (1957)



$$\underset{\mathbf{x}}{\begin{bmatrix} -1 & 2 & -1 \end{bmatrix}} \underset{\mathbf{w}}{\begin{bmatrix} 1 \\ 1 \\ -2 \end{bmatrix}} = \underset{y}{3}$$

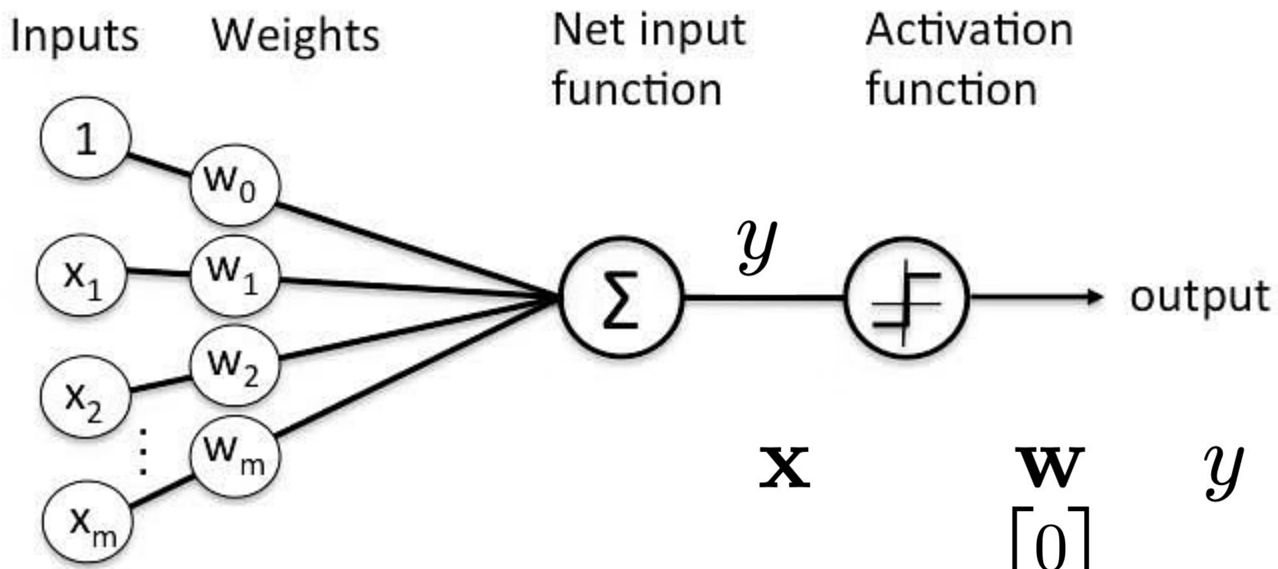
Perceptron (1957)



$$\mathbf{x} \qquad \mathbf{w} \qquad y$$

$$\begin{bmatrix} -1 & 2 & -1 \end{bmatrix} \begin{bmatrix} -1 \\ -1 \\ 2 \end{bmatrix} = -3$$

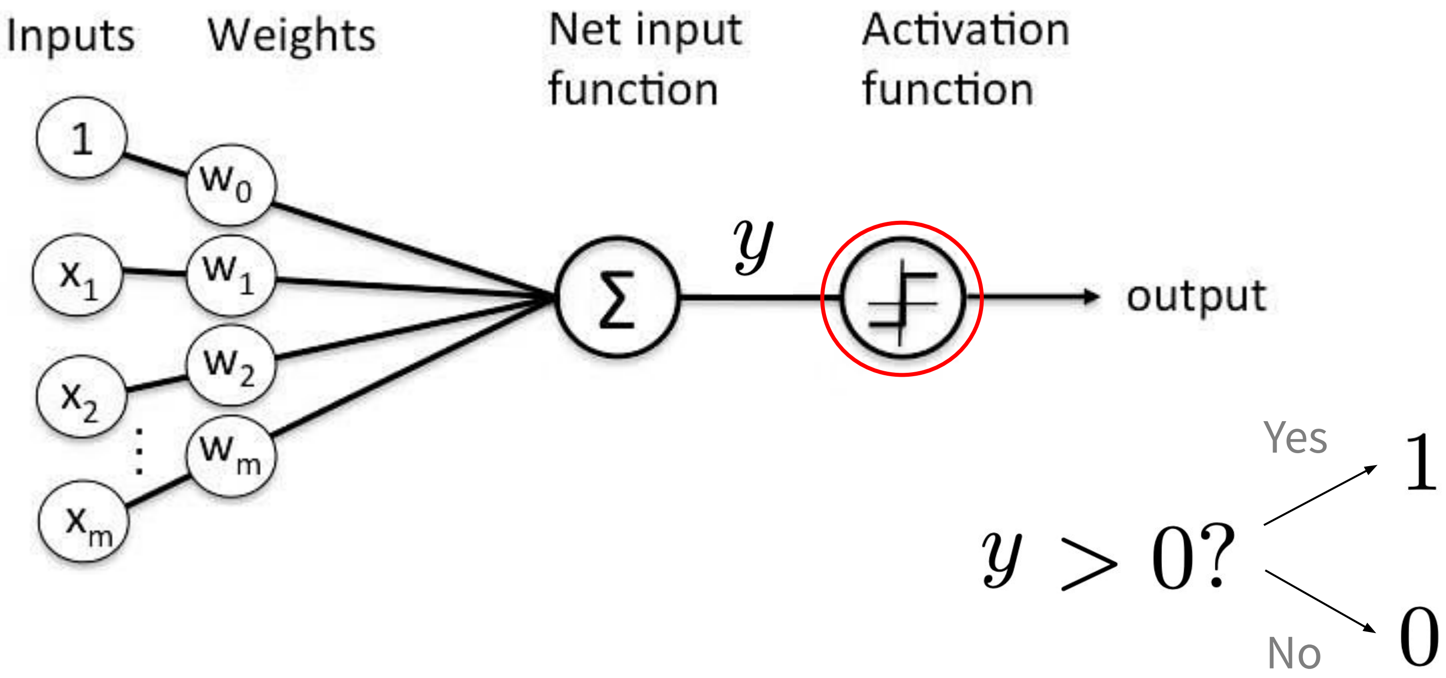
Perceptron (1957)



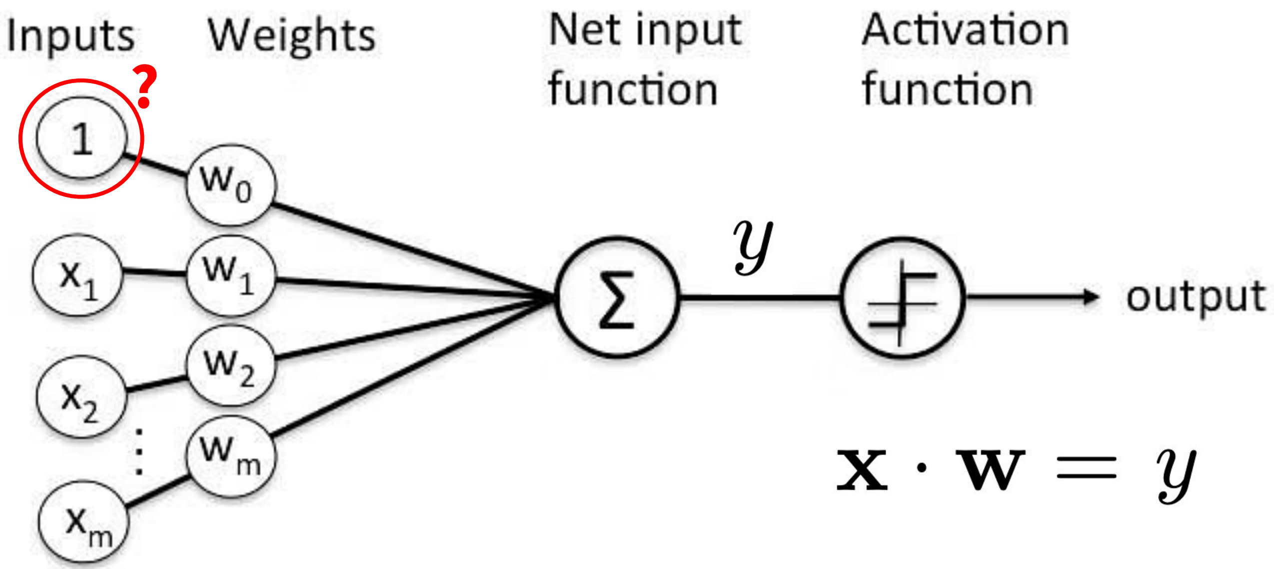
$$\begin{matrix} \mathbf{x} & \mathbf{w} & y \end{matrix}$$

$$\begin{bmatrix} 3 & -1 & 5 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = 5$$

Perceptron (1957)

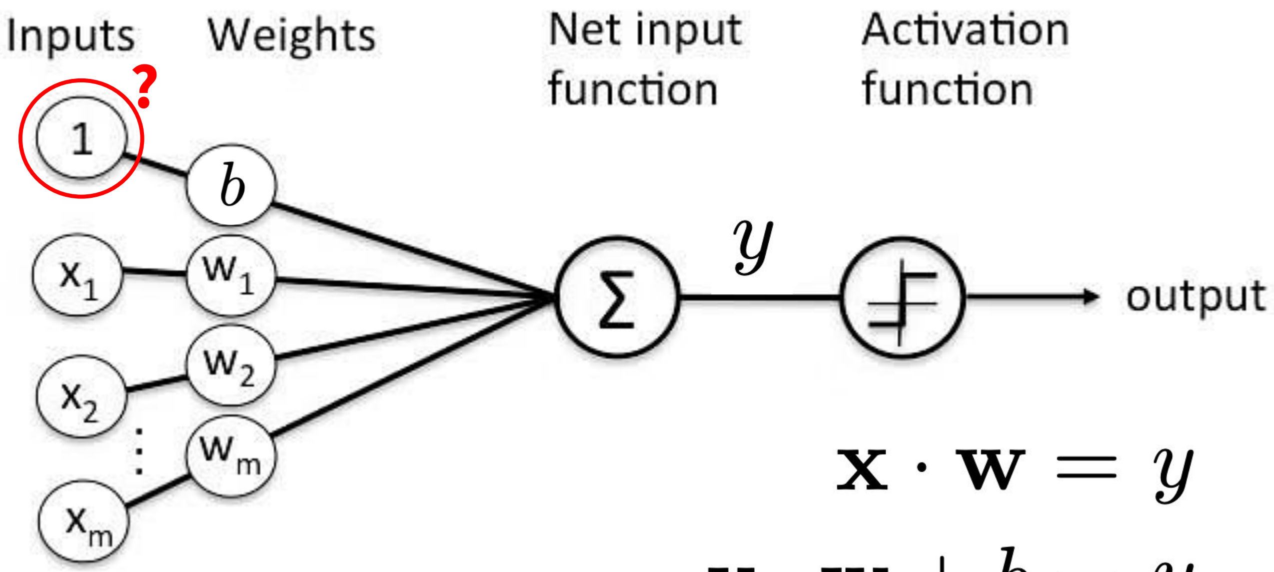


Perceptron (1957)



$$\mathbf{x} \cdot \mathbf{w} = y$$

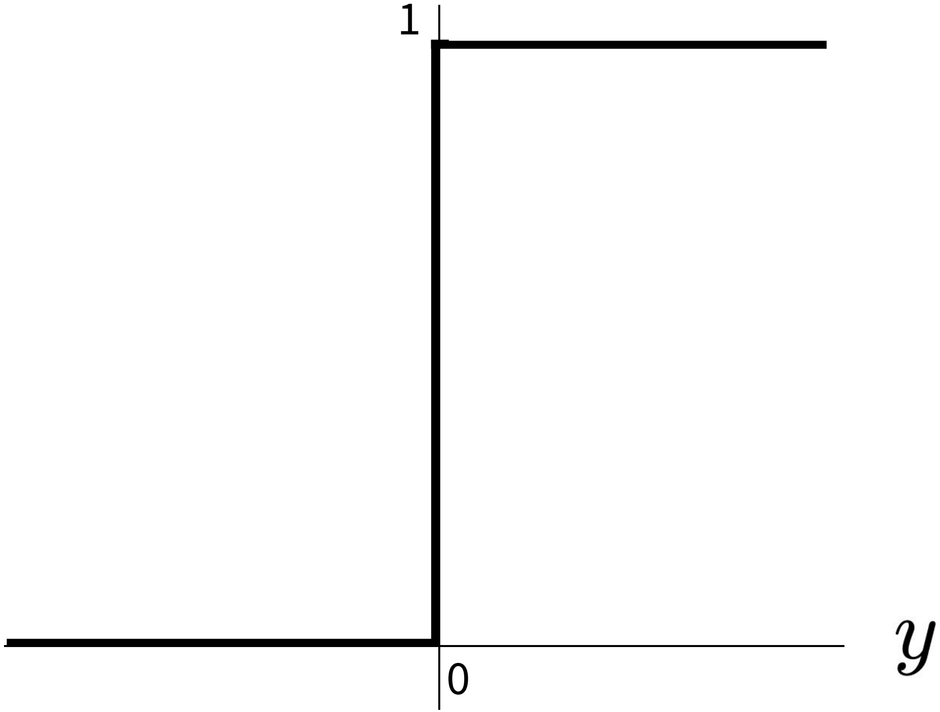
Perceptron (1957)



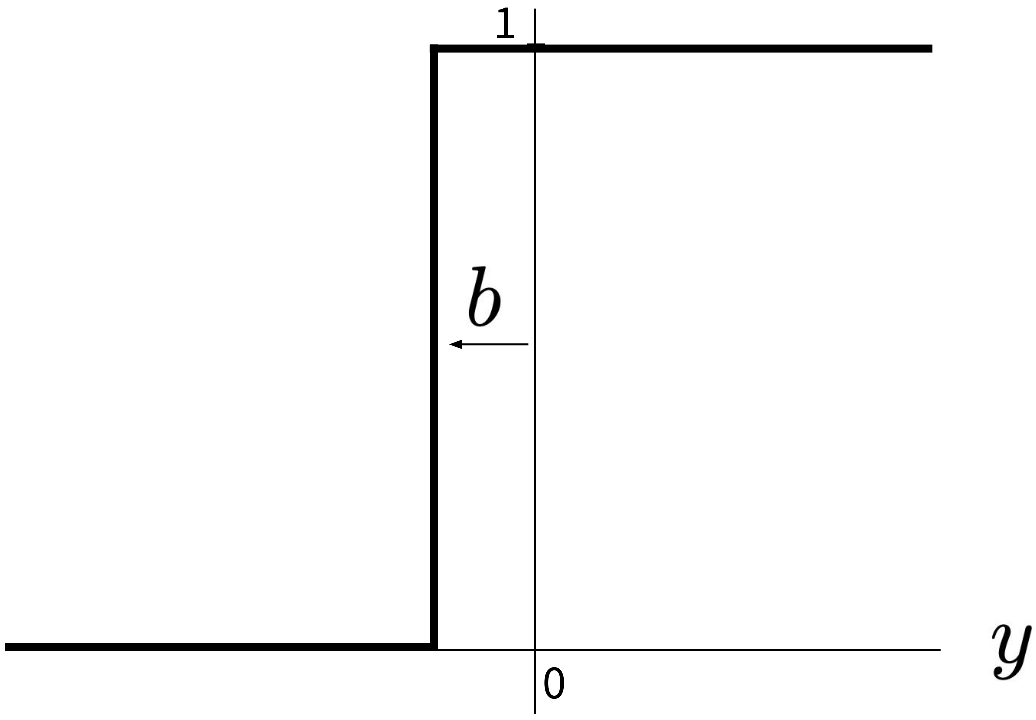
$$\mathbf{x} \cdot \mathbf{w} = y$$

$$\mathbf{x} \cdot \mathbf{w} + b = y$$

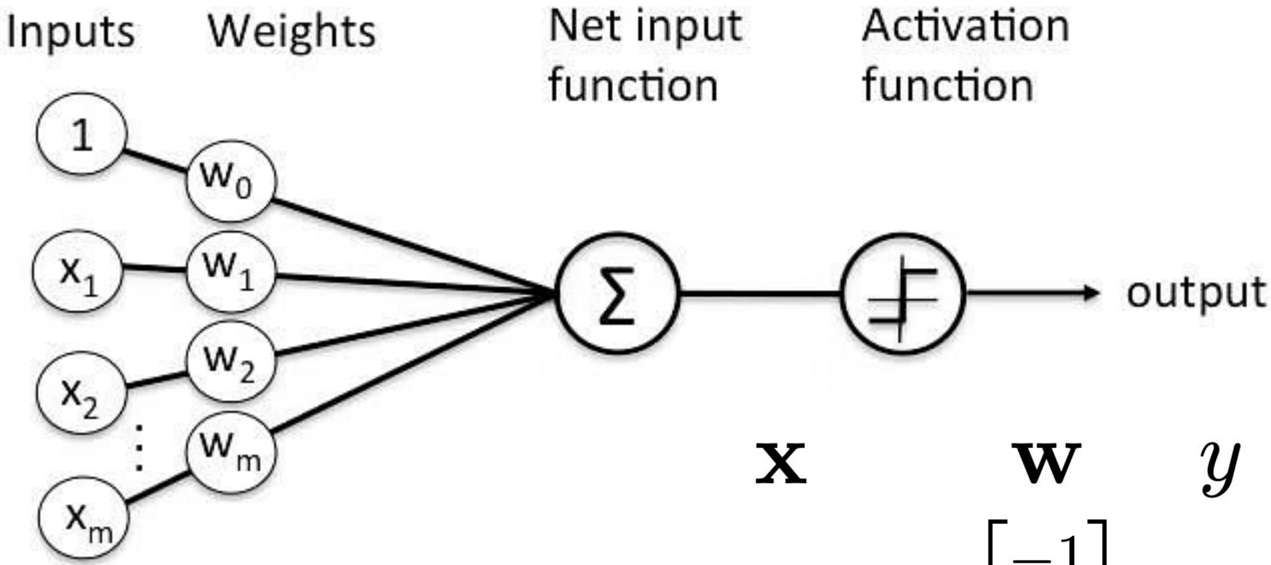
Bias



Bias



Perceptron (1957)



$$\mathbf{x} \qquad \mathbf{w} \qquad y$$

$$\begin{bmatrix} -1 & 2 & -1 \end{bmatrix} \begin{bmatrix} -1 \\ 2 \\ -1 \end{bmatrix} = 6$$

Perceptron (1957)

$$\begin{array}{ccc} \mathbf{x} & \mathbf{w} & y \\ \begin{bmatrix} -1 & 2 & -1 \end{bmatrix} & \begin{bmatrix} -1 \\ 2 \\ -1 \end{bmatrix} & = 6 \end{array}$$

Perceptron (1957)

$$\underset{\mathbf{w}}{\begin{bmatrix} -1 & 2 & -1 \end{bmatrix}} \underset{\mathbf{x}}{\begin{bmatrix} -1 \\ 2 \\ -1 \end{bmatrix}} = \underset{y}{6}$$

Perceptron (1957)

$$\begin{matrix} \mathbf{W} & \mathbf{x} & \mathbf{y} \end{matrix}$$

$$\begin{bmatrix} -1 & 2 & -1 \\ 1 & -2 & 1 \end{bmatrix} \begin{bmatrix} -1 \\ 2 \\ -1 \end{bmatrix} = \begin{bmatrix} 6 \\ -6 \end{bmatrix}$$

Perceptron (1957)

$$\overset{\mathbf{W}}{\begin{bmatrix} -1 & 2 & -1 \\ 1 & -2 & 1 \\ 0 & 1 & 0 \end{bmatrix}} \overset{\mathbf{x}}{\begin{bmatrix} -1 \\ 2 \\ -1 \end{bmatrix}} = \overset{\mathbf{y}}{\begin{bmatrix} 6 \\ -6 \\ 2 \end{bmatrix}}$$

Perceptron (1957)

$$\underset{\mathbf{W}}{\begin{bmatrix} -1 & 2 & -1 \\ 1 & -2 & 1 \\ 0 & 1 & 0 \end{bmatrix}} \underset{\mathbf{x}}{\begin{bmatrix} -1 \\ 2 \\ -1 \end{bmatrix}} = \underset{\mathbf{y}}{\begin{bmatrix} 6 \\ -6 \\ 2 \end{bmatrix}}$$

$$\underset{\mathbf{W}}{\begin{bmatrix} -1 & 2 & -1 \\ 1 & -2 & 1 \\ 0 & 1 & 0 \end{bmatrix}} \underset{\mathbf{x}}{\begin{bmatrix} -1 \\ 2 \\ -1 \end{bmatrix}} + \underset{\mathbf{b}}{\begin{bmatrix} 1 \\ 1 \\ -3 \end{bmatrix}} = \underset{\mathbf{y}}{\begin{bmatrix} 7 \\ -5 \\ -1 \end{bmatrix}}$$

Perceptron (1957)

$$\sigma\left(\underbrace{\begin{bmatrix}-1 & 2 & -1\\ 1 & -2 & 1\\ 0 & 1 & 0\end{bmatrix}}_{\mathbf{W}}\underbrace{\begin{bmatrix}-1\\ 2\\ -1\end{bmatrix}}_{\mathbf{x}}+\underbrace{\begin{bmatrix}1\\ 1\\ -3\end{bmatrix}}_{\mathbf{b}}\right)=\underbrace{\begin{bmatrix}1\\ 0\\ 0\end{bmatrix}}_{\mathbf{y}}$$

$$\underbrace{\begin{bmatrix}-1 & 2 & -1\\ 1 & -2 & 1\\ 0 & 1 & 0\end{bmatrix}}_{\mathbf{W}}\underbrace{\begin{bmatrix}-1\\ 2\\ -1\end{bmatrix}}_{\mathbf{x}}+\underbrace{\begin{bmatrix}1\\ 1\\ -3\end{bmatrix}}_{\mathbf{b}}=\underbrace{\begin{bmatrix}7\\ -5\\ -1\end{bmatrix}}_{\mathbf{y}}$$

Perceptron (1957)

$$\overset{\mathbf{W}}{\sigma\left(\begin{bmatrix} -1 & 2 & -1 \\ 1 & -2 & 1 \\ 0 & 1 & 0 \end{bmatrix}\right.}\overset{\mathbf{x}}{\begin{bmatrix} -1 \\ 2 \\ -1 \end{bmatrix}} + \overset{\mathbf{b}}{\left.\begin{bmatrix} 1 \\ 1 \\ -3 \end{bmatrix}\right)} = \overset{\mathbf{y}}{\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}}$$

$$\mathbf{y} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$$

Perceptron (1957)

$$\sigma\left(\underbrace{\begin{bmatrix}-1 & 2 & -1\\ 1 & -2 & 1\\ 0 & 1 & 0\end{bmatrix}}_{\mathbf{W}}\underbrace{\begin{bmatrix}-1\\ 2\\ -1\end{bmatrix}}_{\mathbf{x}}+\underbrace{\begin{bmatrix}1\\ 1\\ -3\end{bmatrix}}_{\mathbf{b}}\right)=\underbrace{\begin{bmatrix}1\\ 0\\ 0\end{bmatrix}}_{\mathbf{y}}$$

$$\mathbf{y}=\sigma(\mathbf{W}\mathbf{x}+\mathbf{b})$$

btw. $\mathbf{y}=\sigma(\mathbf{W}\mathbf{X}+\mathbf{b})$

$$\mathbf{y} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$$

$$\mathbf{y} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$$

Perceptron

- Brain-inspired
- Represented as matrix multiplication + bias + activation function
- Basis of most neural networks computation

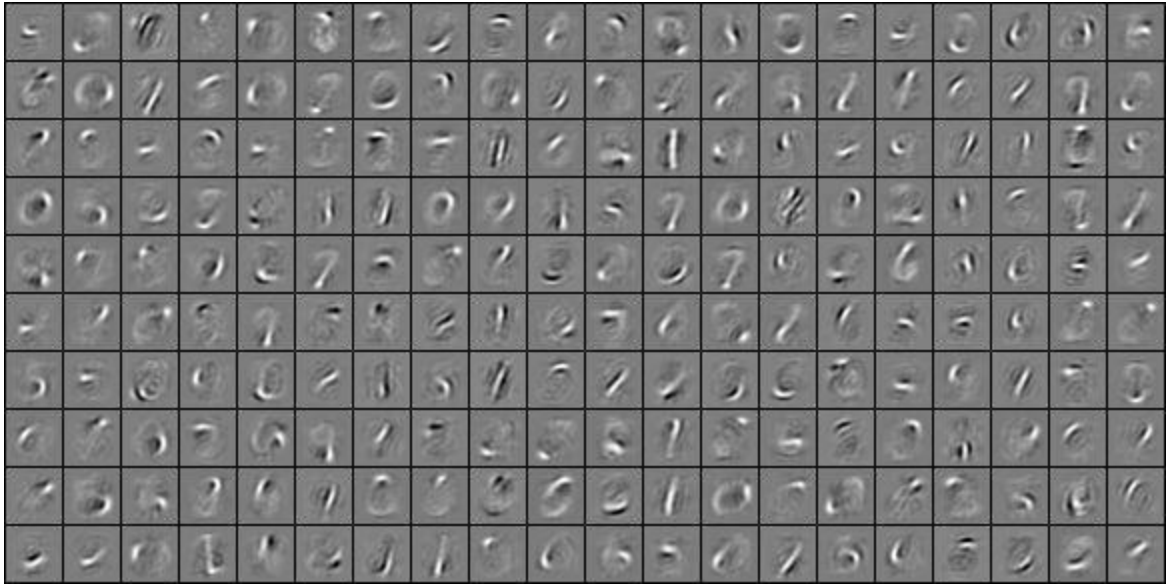
Btw.

- GPU!
- Camera rotation → Matrix multiplication $\mathbf{WX}$
- Mainly: Wide hardware "pathway" between HD and GPU compared to HD and CPU

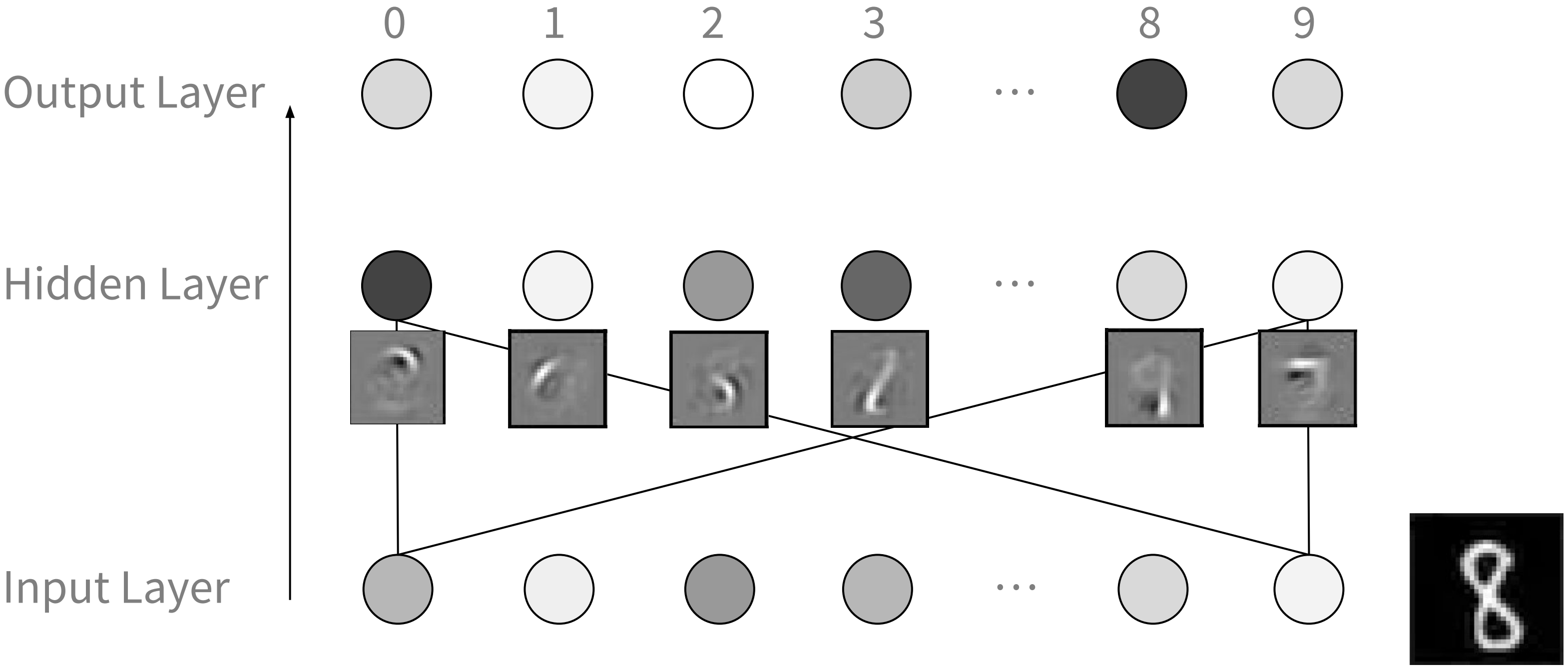
Frank Rosenblatt, The perceptron - A perceiving and recognizing automaton. Cornell Aeronautical Laboratory, Report No. 85-460-1, January 1957

Multi-Layer Perceptron

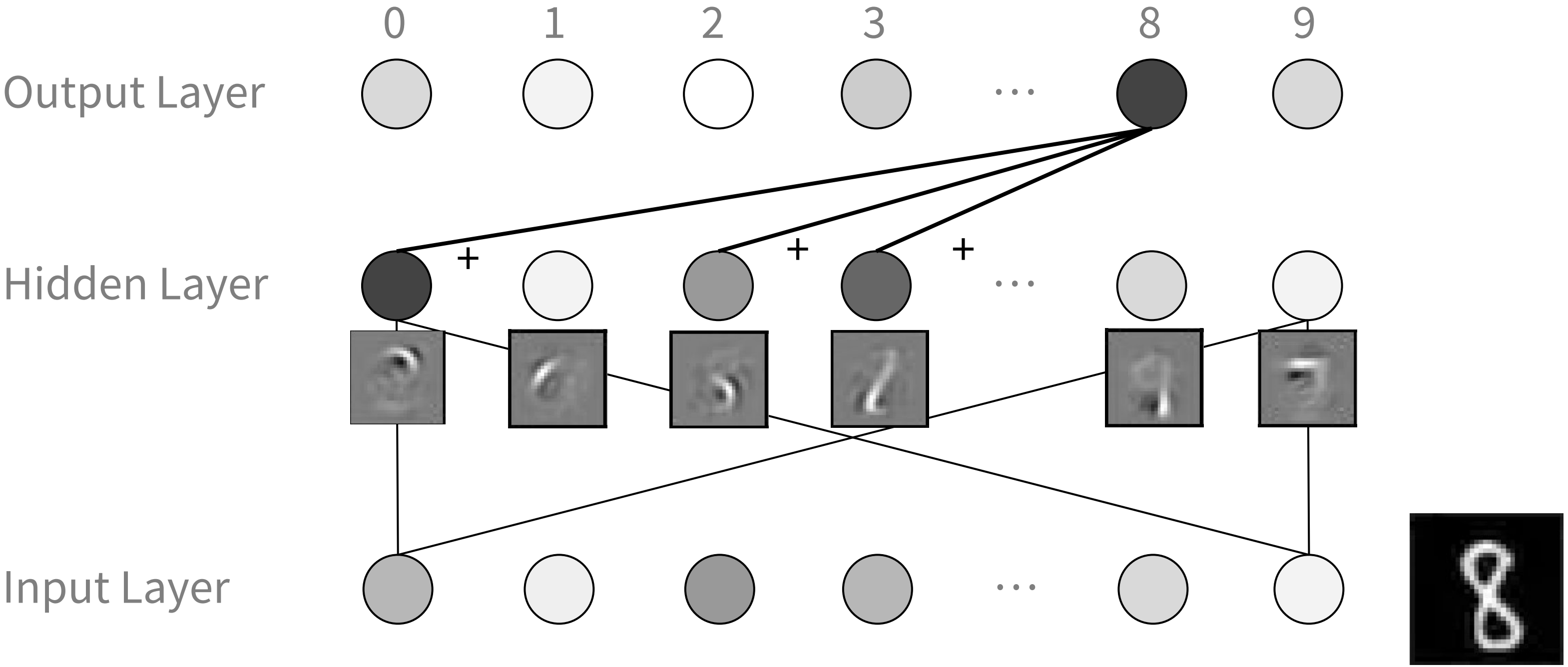
MNIST Filters



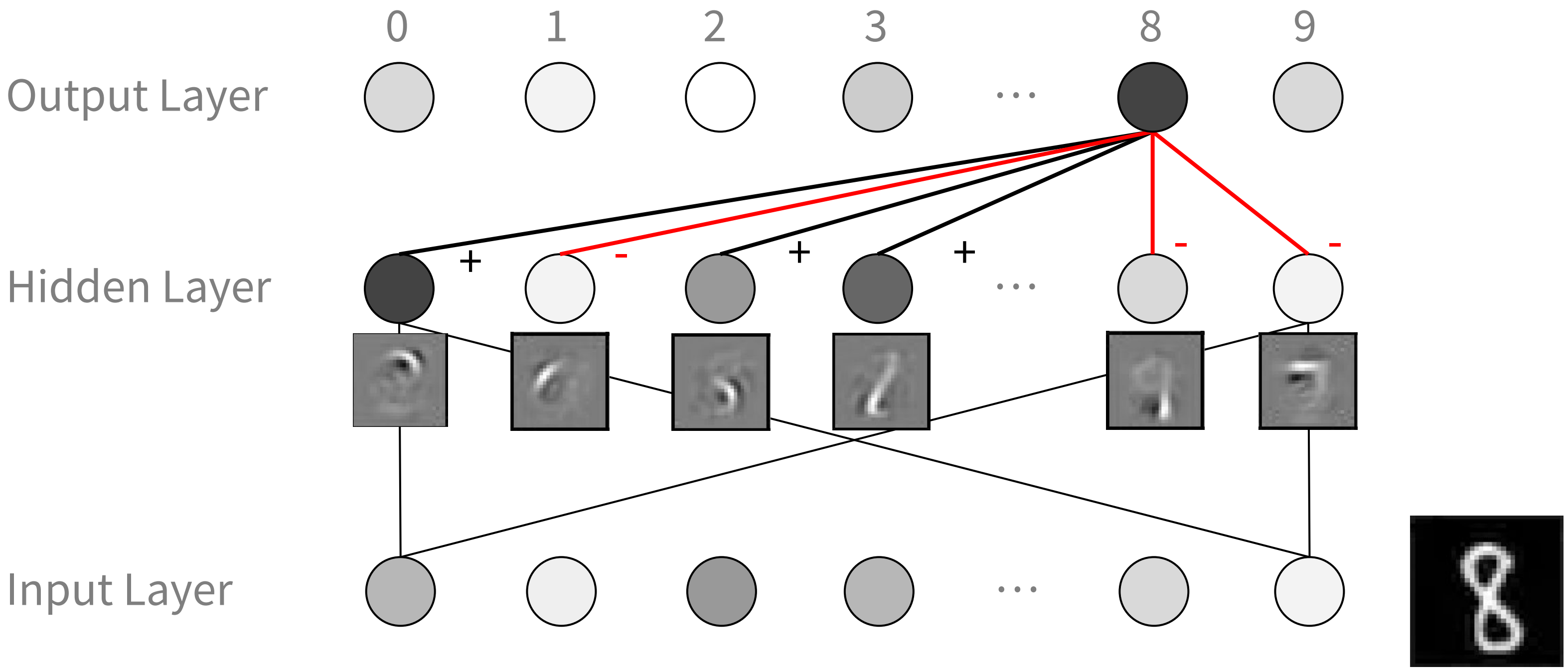
MNIST Classification



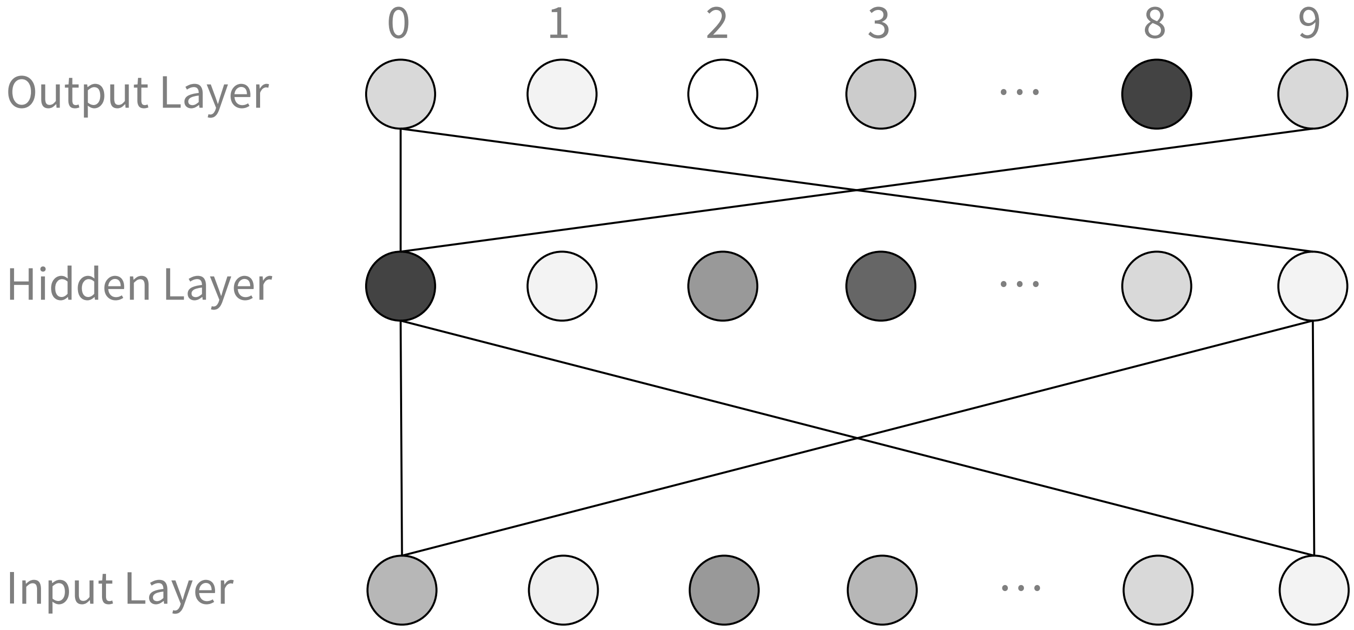
MNIST Classification



MNIST Classification



Multi-Layer Perceptron



Frank Rosenblatt, **The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain,** (1958)

Multi-Layer Perceptron

Demo:

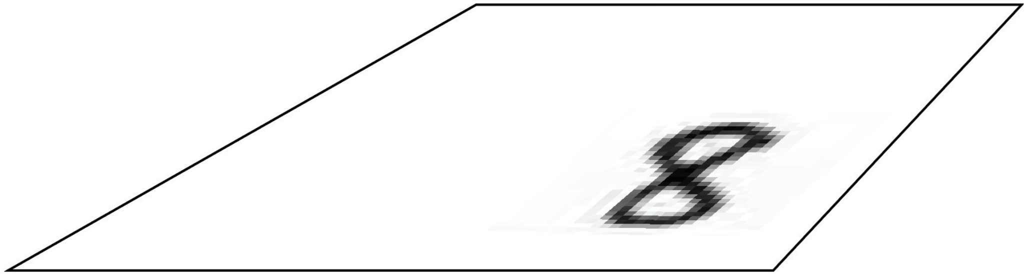
https://www.3blue1brown.com/lessons/neural-network-analysis

Multi-Layer Perceptron

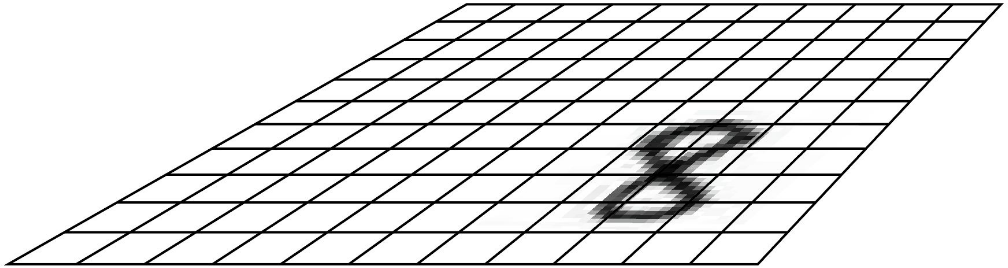
- Going deeper
- Hierarchical feature representation
- “Dense feed-forward NN”

Convolutional Neural Networks

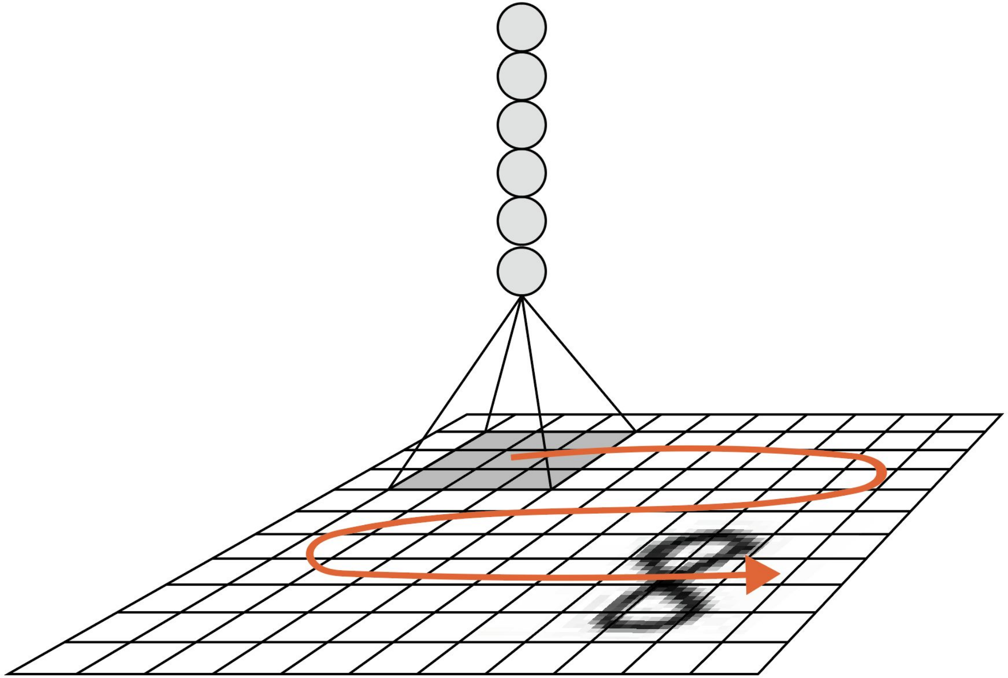
MNIST Classification - Convolutional



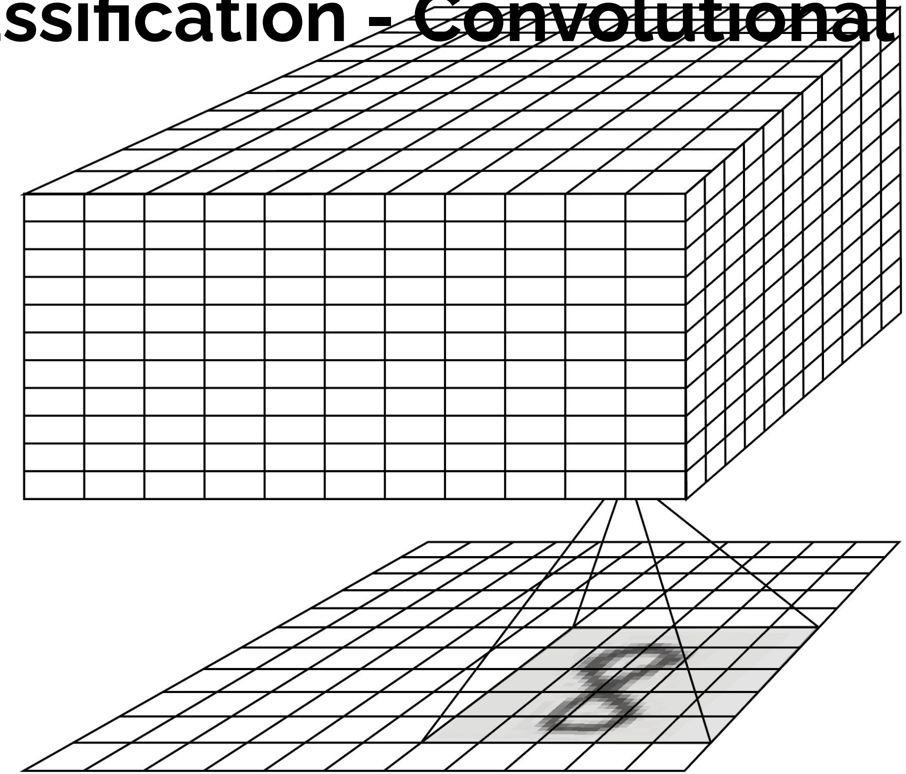
MNIST Classification - Convolutional



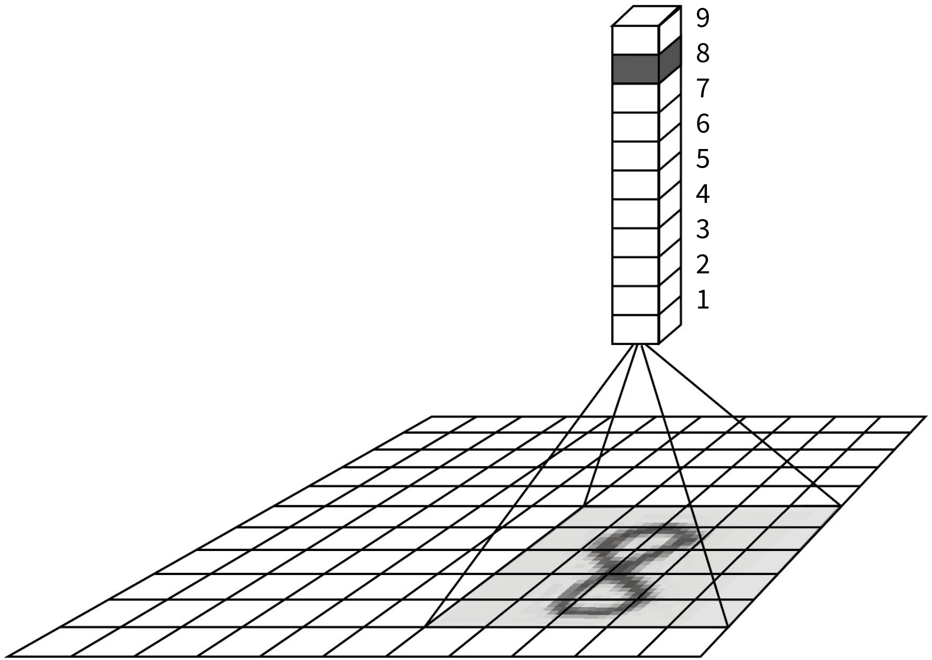
MNIST Classification - Convolutional



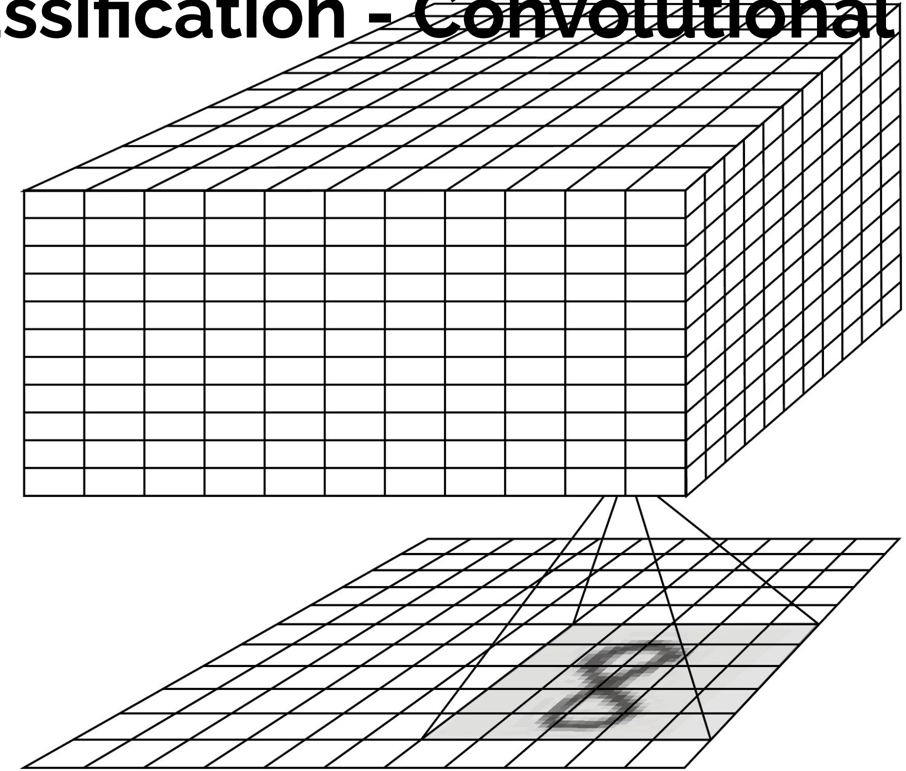
MNIST Classification - Convolutional



MNIST Classification - Convolutional

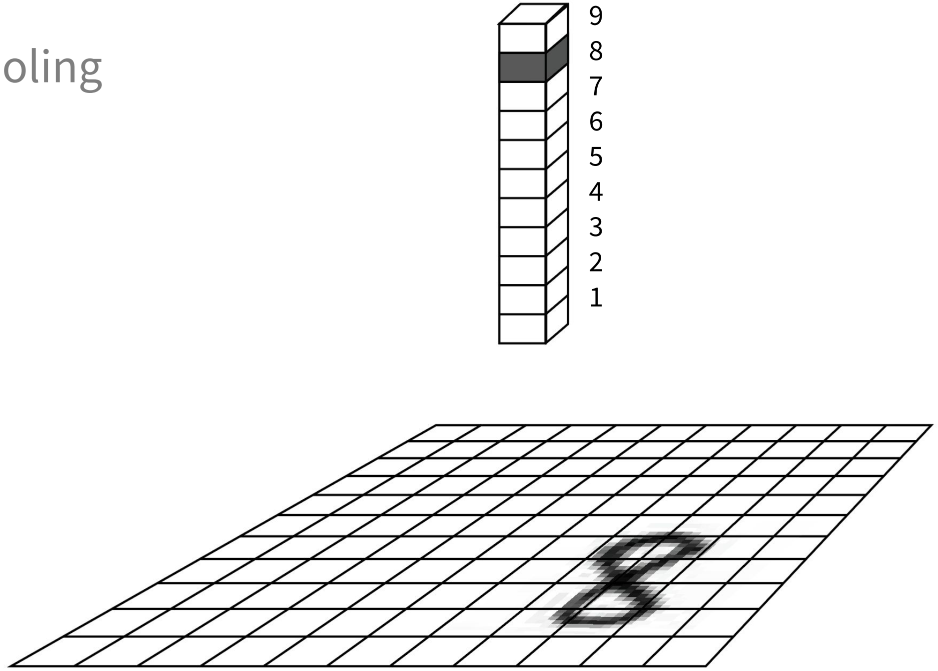


MNIST Classification - Convolutional



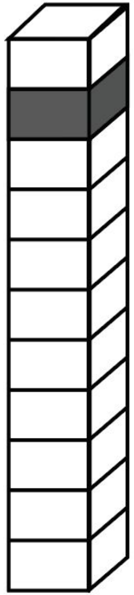
MNIST Classification - Convolutional

Average Pooling



MNIST Classification - Convolutional

Average Pooling



Result: **8**

Convolutional Neural Network

Yann LeCun

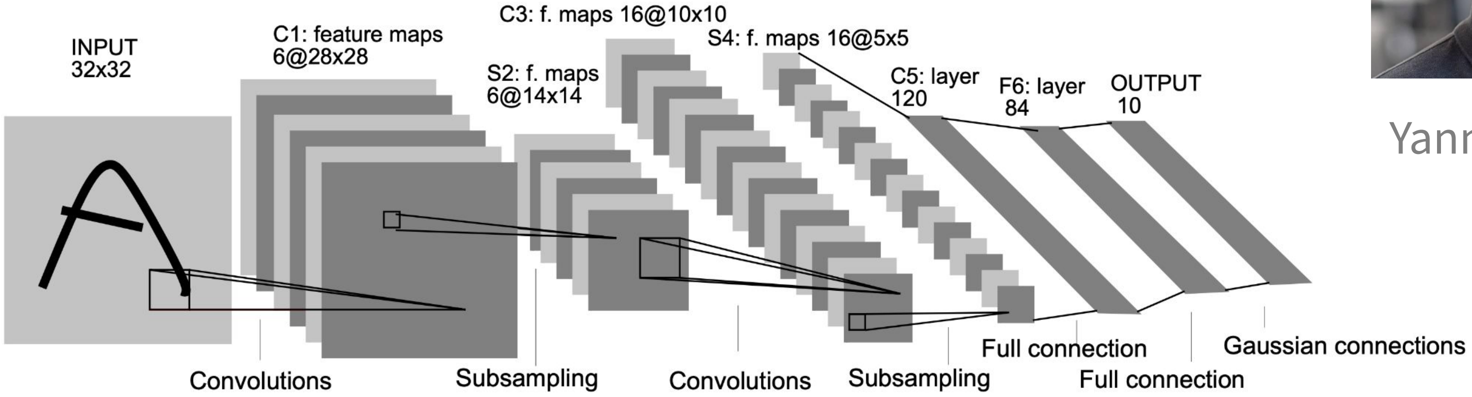


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

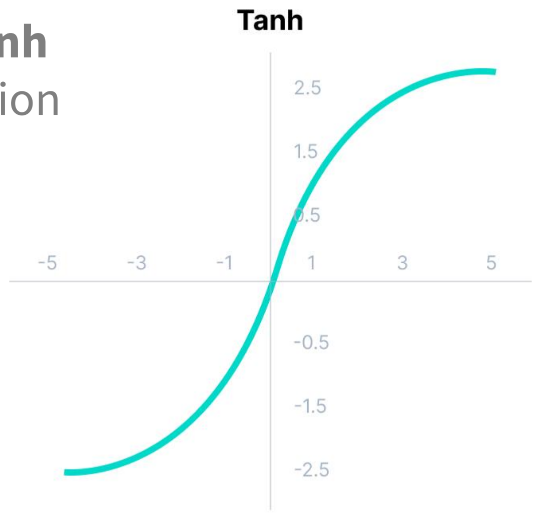
Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner:
Gradient-based learning applied to document recognition. Proc. IEEE 86(11): 2278-2324 (1998)

Convolutional Neural Networks

- Hierarchical Feature Learning
- Translation Invariance
- Reduced Parameter Count
- Robust to Variations and Distortions

Convolutional Neural Network

- LeNet 5 uses **Tanh** activation function



Yann LeCun

Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner:
Gradient-based learning applied to document recognition. Proc. IEEE 86(11): 2278-2324 (1998)

Neocognitron

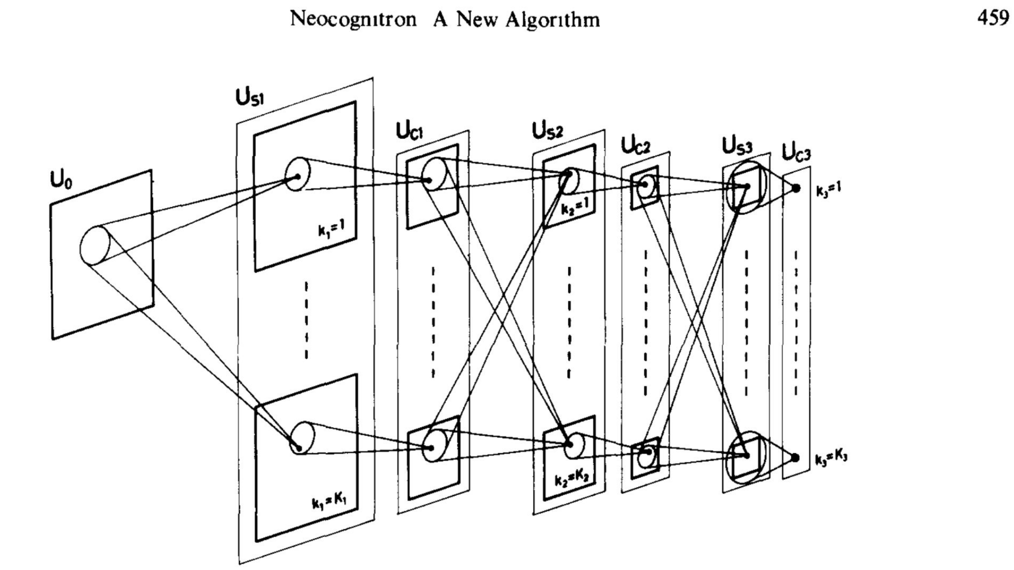


Fig 4 Schematic diagram illustrating the interconnections between layers in the neocognitron

Kunihiko Fukushima

Kunihiko Fukushima, Sei Miyake:
Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position.
Pattern Recognit. 15(6): 455-469 (1982)

Neocognitron

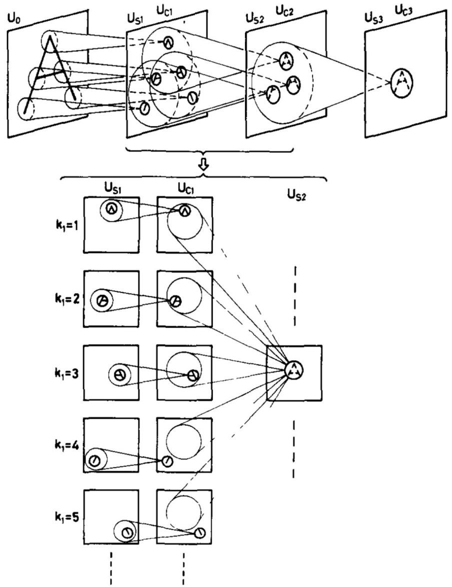


Fig 8 An example of the interconnections between cells and the response of the cells after completion of the self-organization

Kunihiko Fukushima

Kunihiko Fukushima, Sei Miyake:
Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position.
Pattern Recognit. 15(6): 455-469 (1982)

Backpropagation

Gradient Descent

For any differentiable function, we can ask how to change any parameter in order to decrease the function.

E.g., for

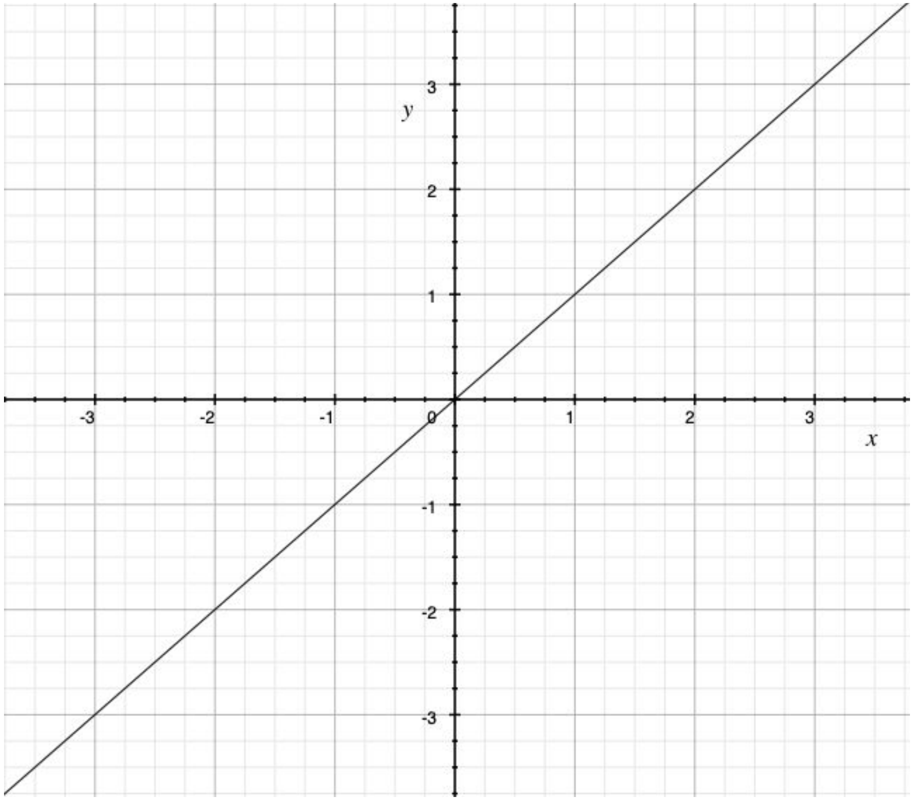
$$\mathbf{y} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$$
$$\epsilon = \sum \mathrm{abs}(\mathbf{y} - \hat{\mathbf{y}})$$

where **W** and **b** are the parameters.

Gradient Descent

$$y = x \cdot k$$

$$k = 1$$

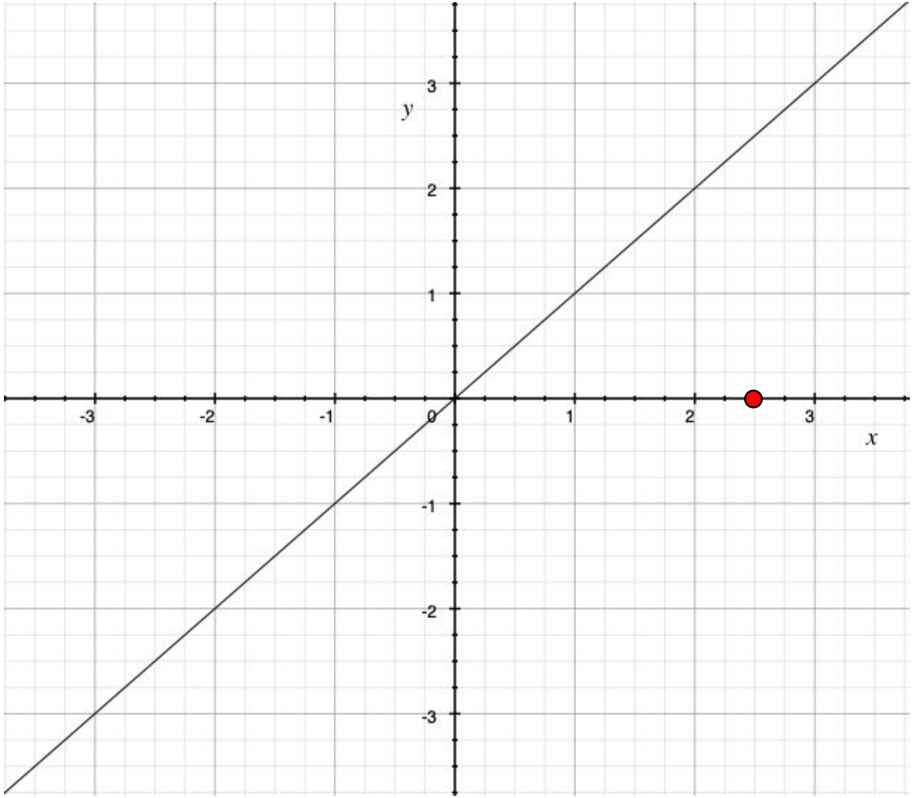


Gradient Descent

$$y = x \cdot k$$

$$k = 1$$

$$x = 2.5$$



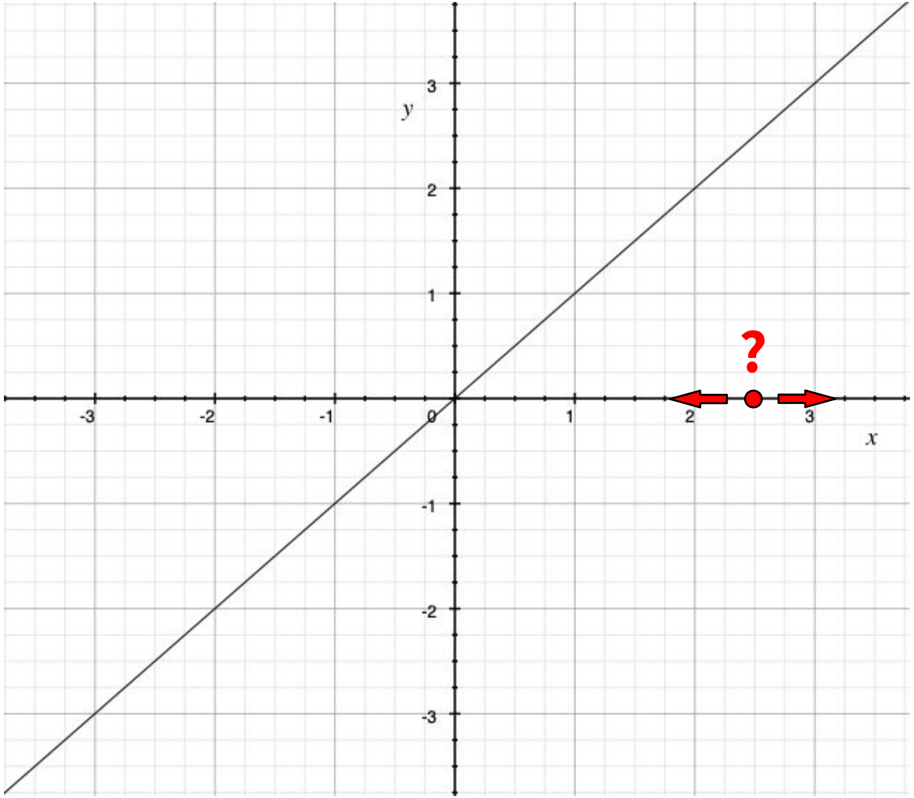
Gradient Descent

$$y = x \cdot k$$

$$k = 1$$

$$x = 2.5$$

$$\delta x\,?$$



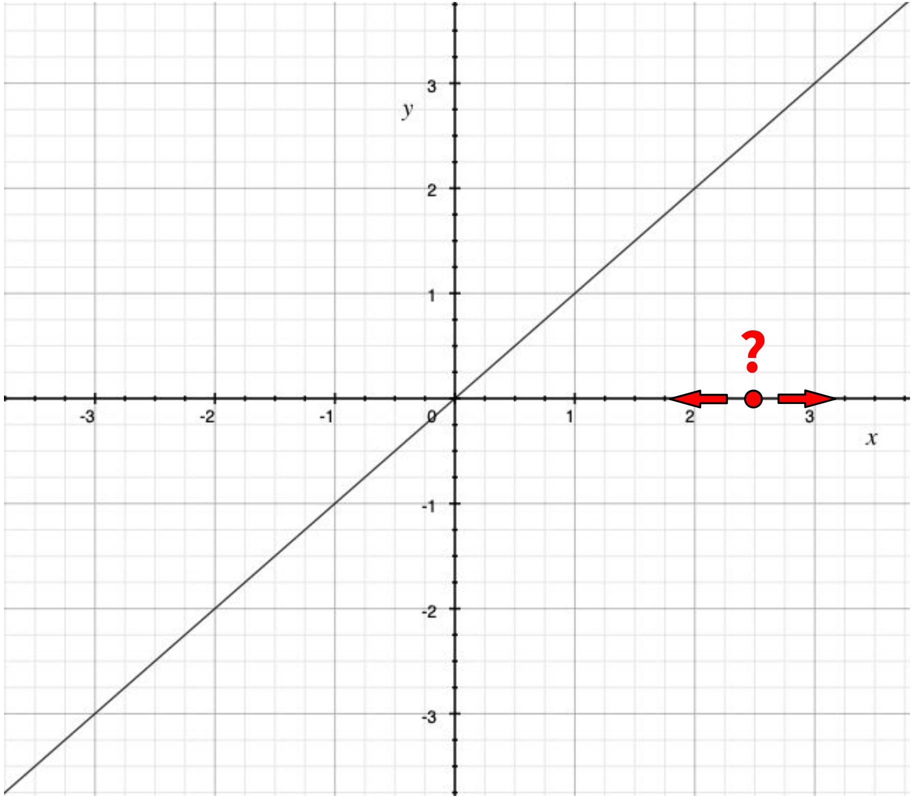
Gradient Descent

$$y = x \cdot k$$

$$k = 1$$

$$x = 2.5$$

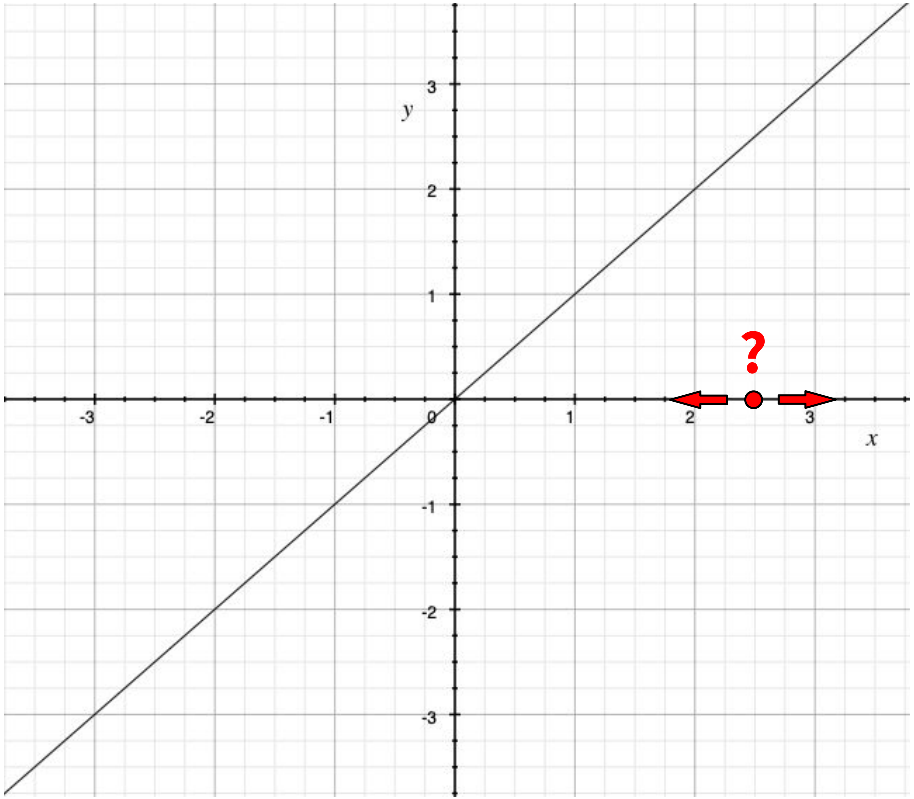
$$\frac{\delta y}{\delta x} ?$$



Gradient Descent

$$y = x \cdot k$$

$$\frac{\delta y}{\delta x} = k$$

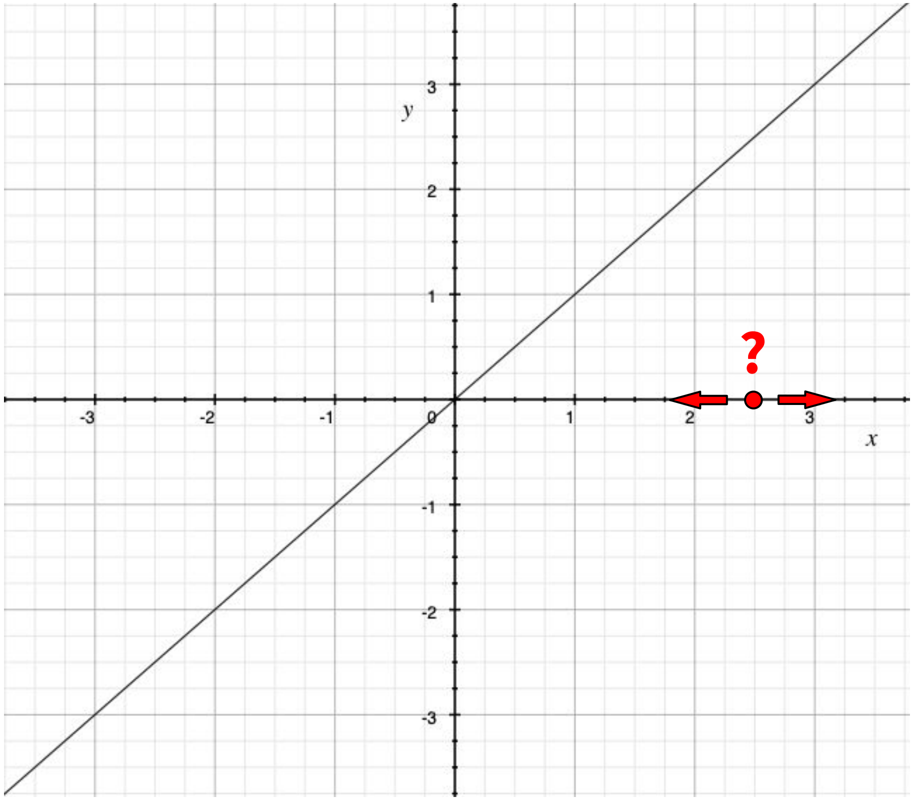


Gradient Descent

$$y = x \cdot k$$

$$\frac{\delta y}{\delta x} = k$$

$$k = 1$$



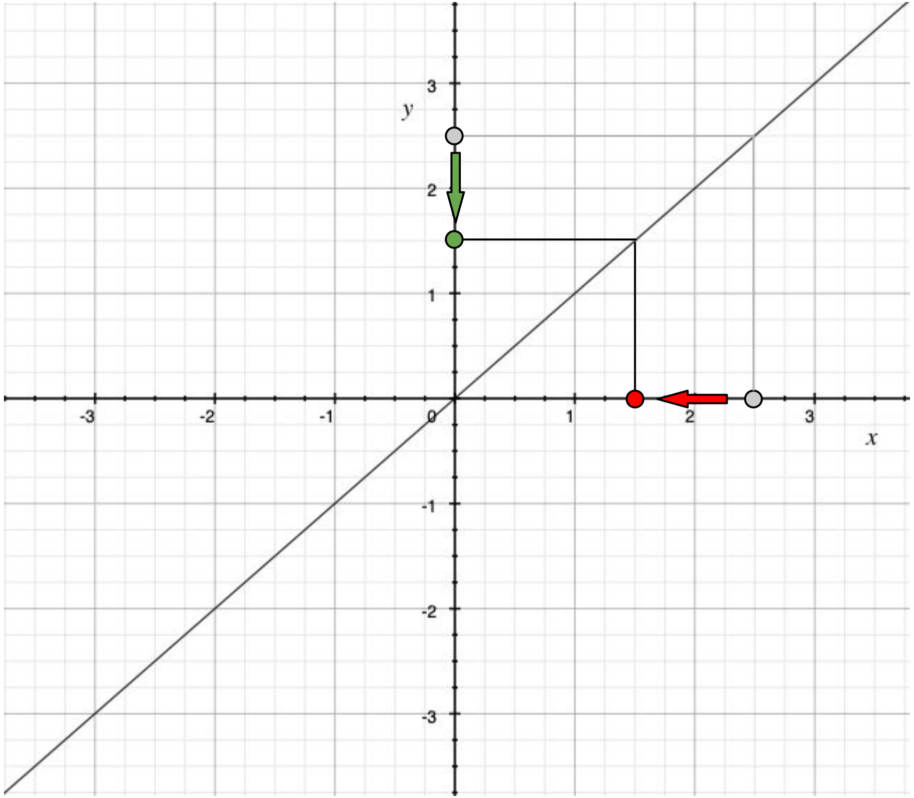
Gradient Descent

$$y = x \cdot k$$

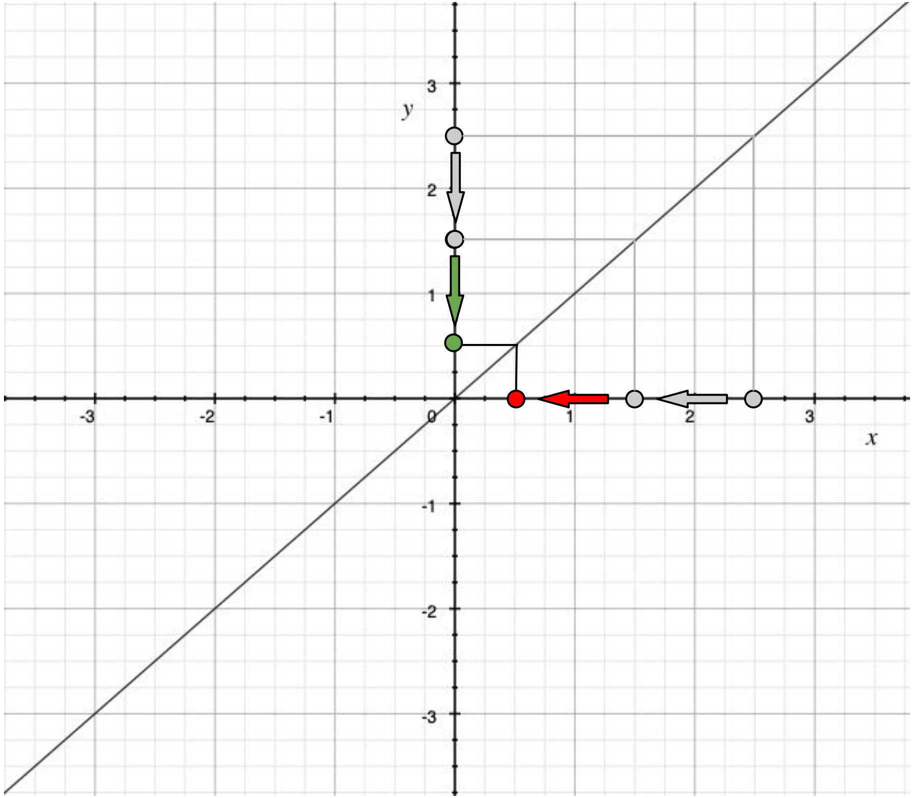
$$\frac{\delta y}{\delta x} = k$$

$$k = 1$$

$$x_{t+1} = x_t - 1$$

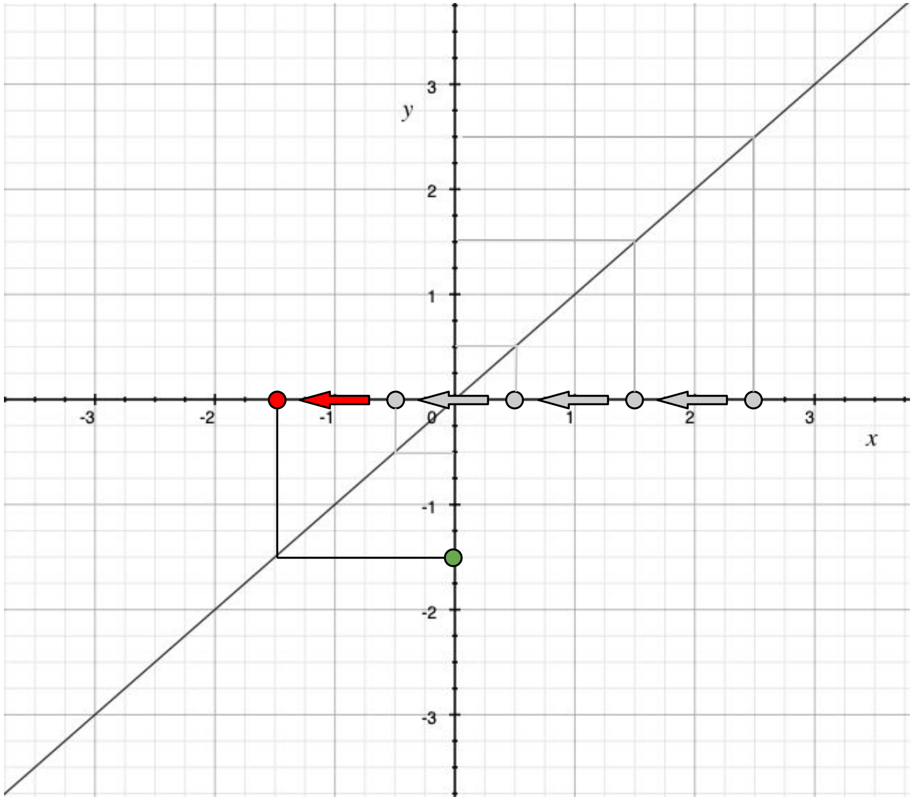


Gradient Descent



$$x_{t+1} = x_t - 1$$

Gradient Descent

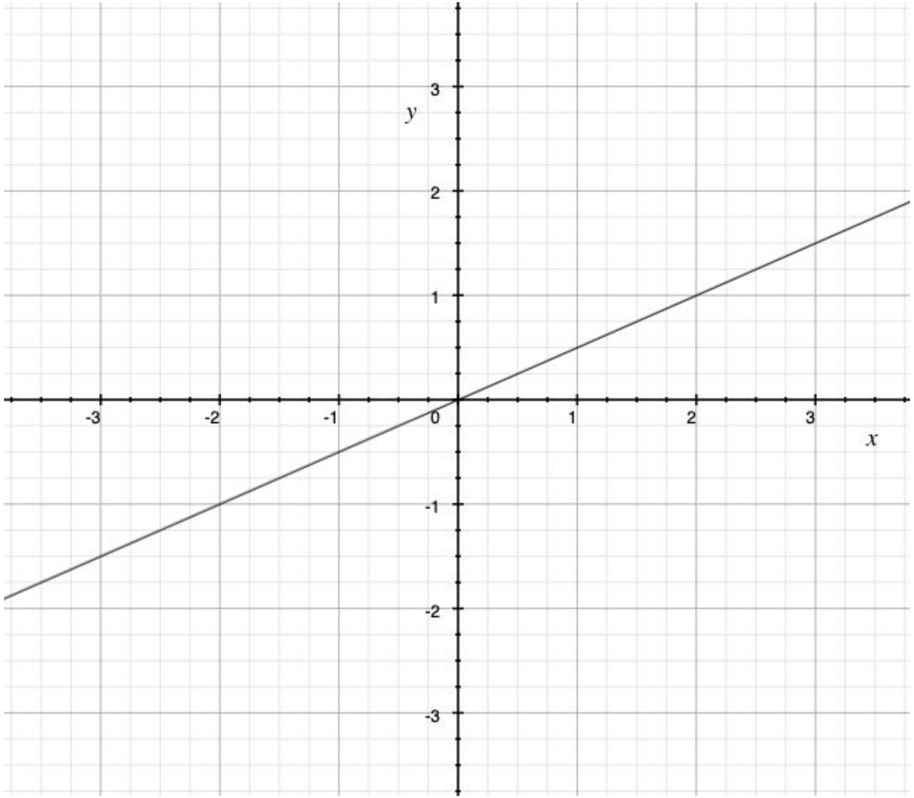


$$x_{t+1} = x_t - 1$$

Gradient Descent

$$y = x \cdot k$$

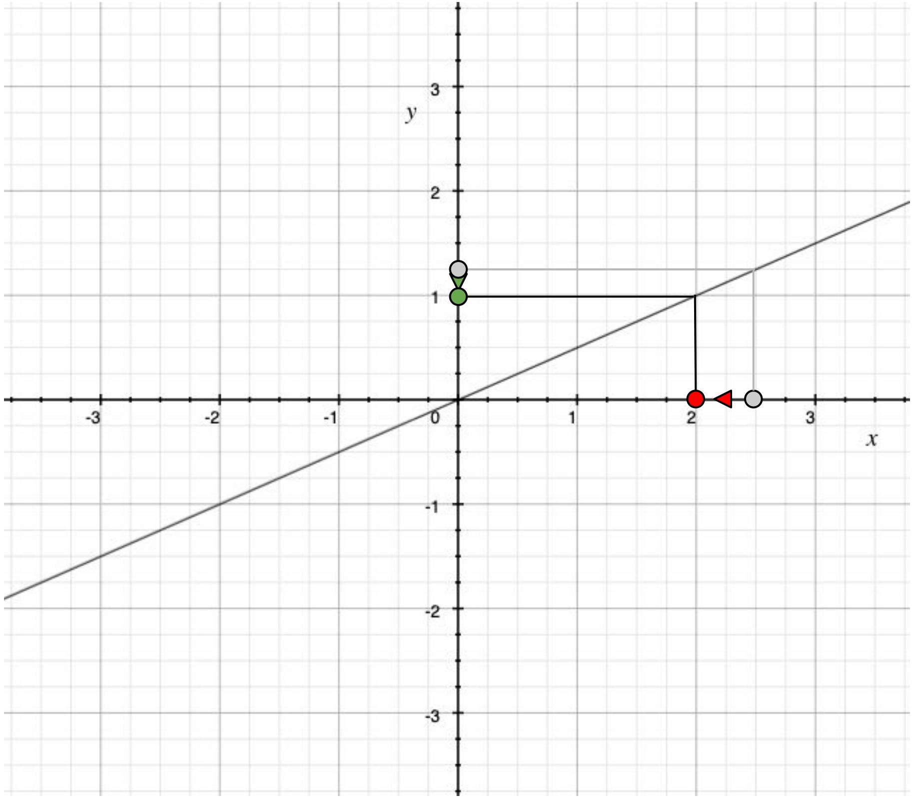
$$k = 0.5$$



Gradient Descent

$$y = x \cdot k$$

$$k = 0.5$$



$$x_{t+1} = x_t - 0.5$$

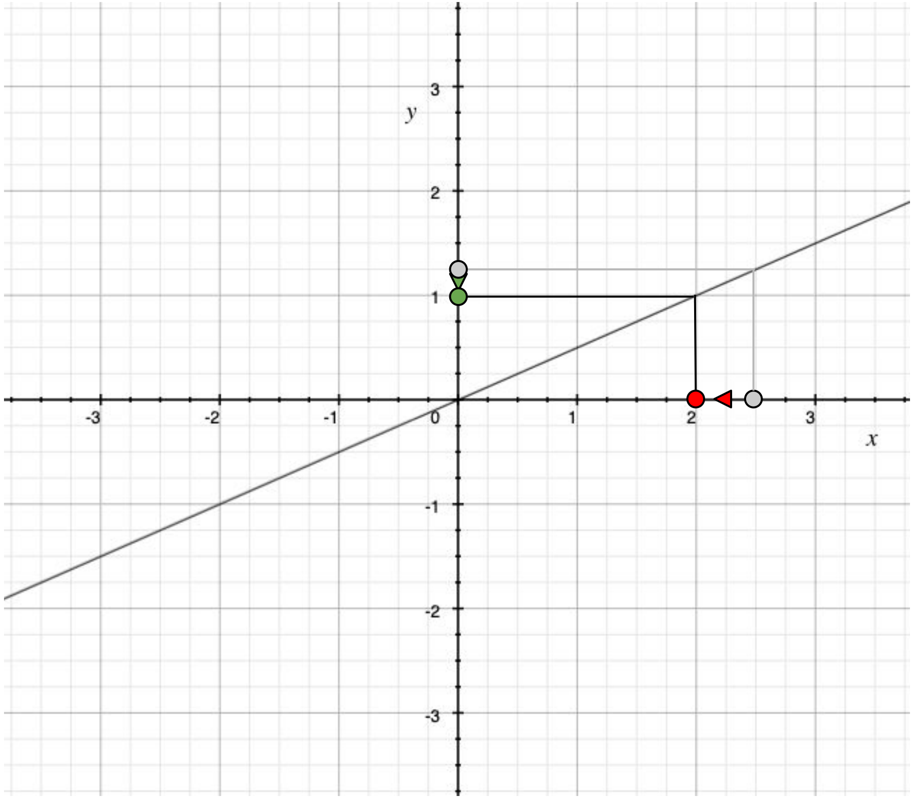
Gradient Descent

$$y = x \cdot k$$

$$k = 0.5$$

$$\delta y = -0.25$$

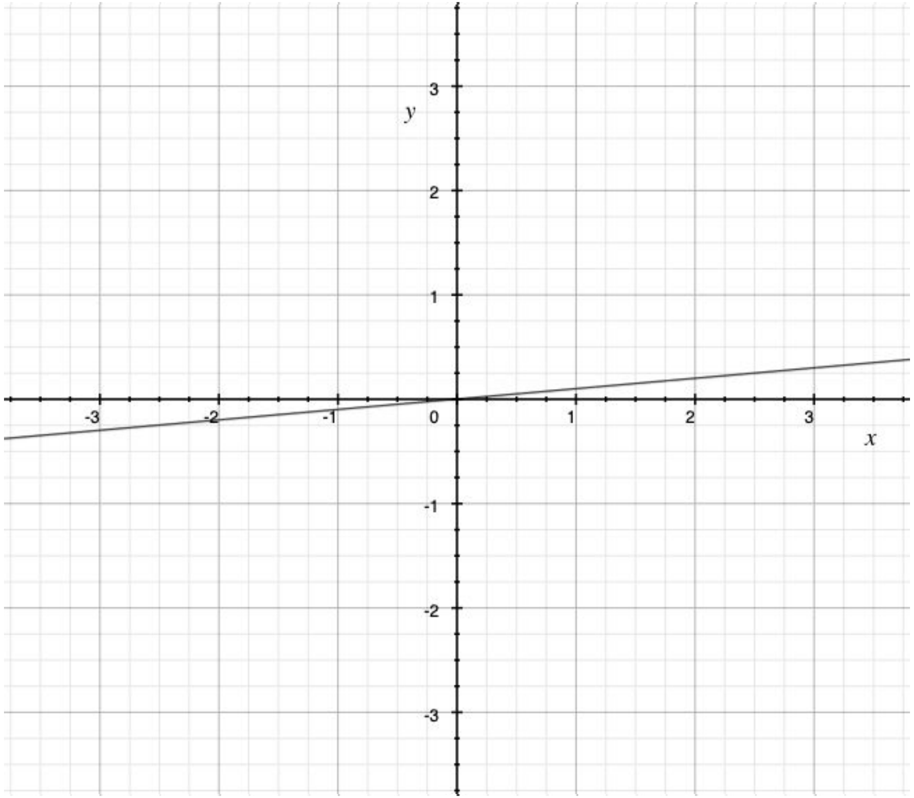
$$x_{t+1} = x_t - 0.5$$



Gradient Descent

$$y = x \cdot k$$

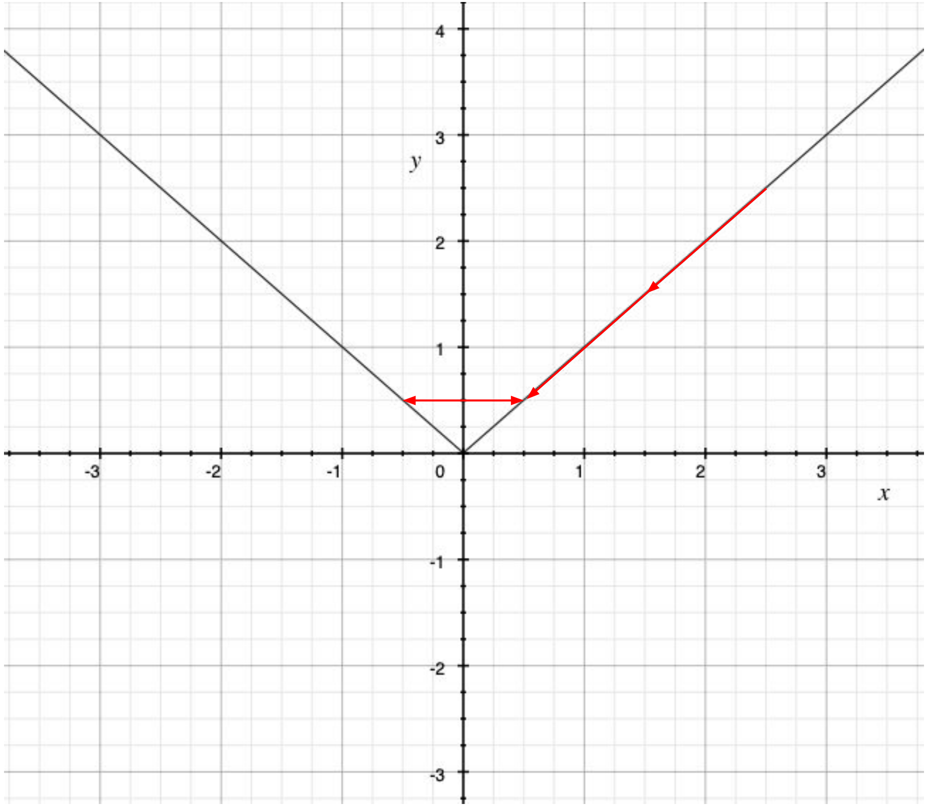
$$k = 0.1$$



$$\delta y = 0.01$$

$$x_{t+1} = x_t - 0.1$$

Gradient Descent



Gradient Descent

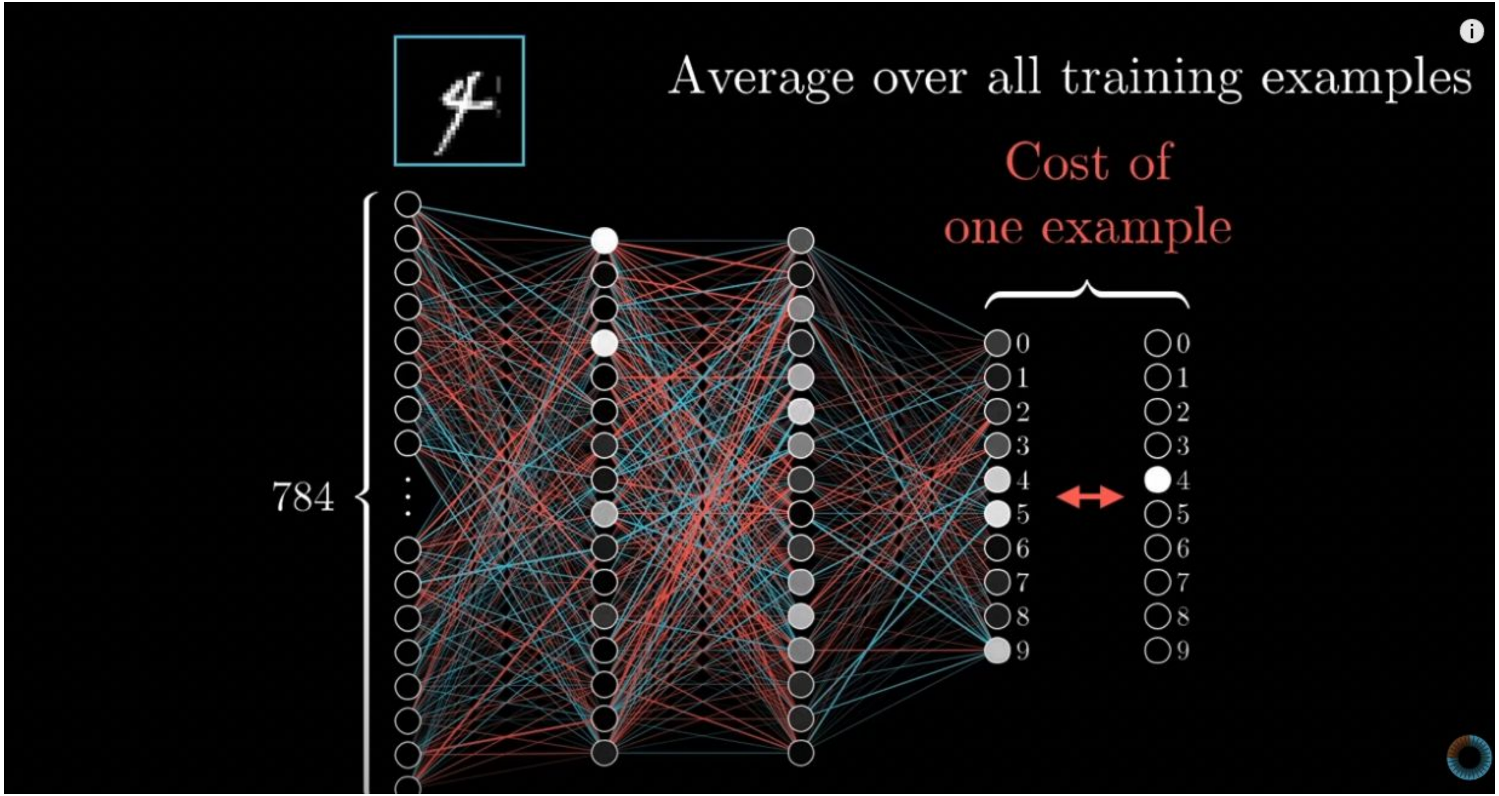
For any differentiable function, we can ask how to change any parameter in order to decrease the function.

E.g., for

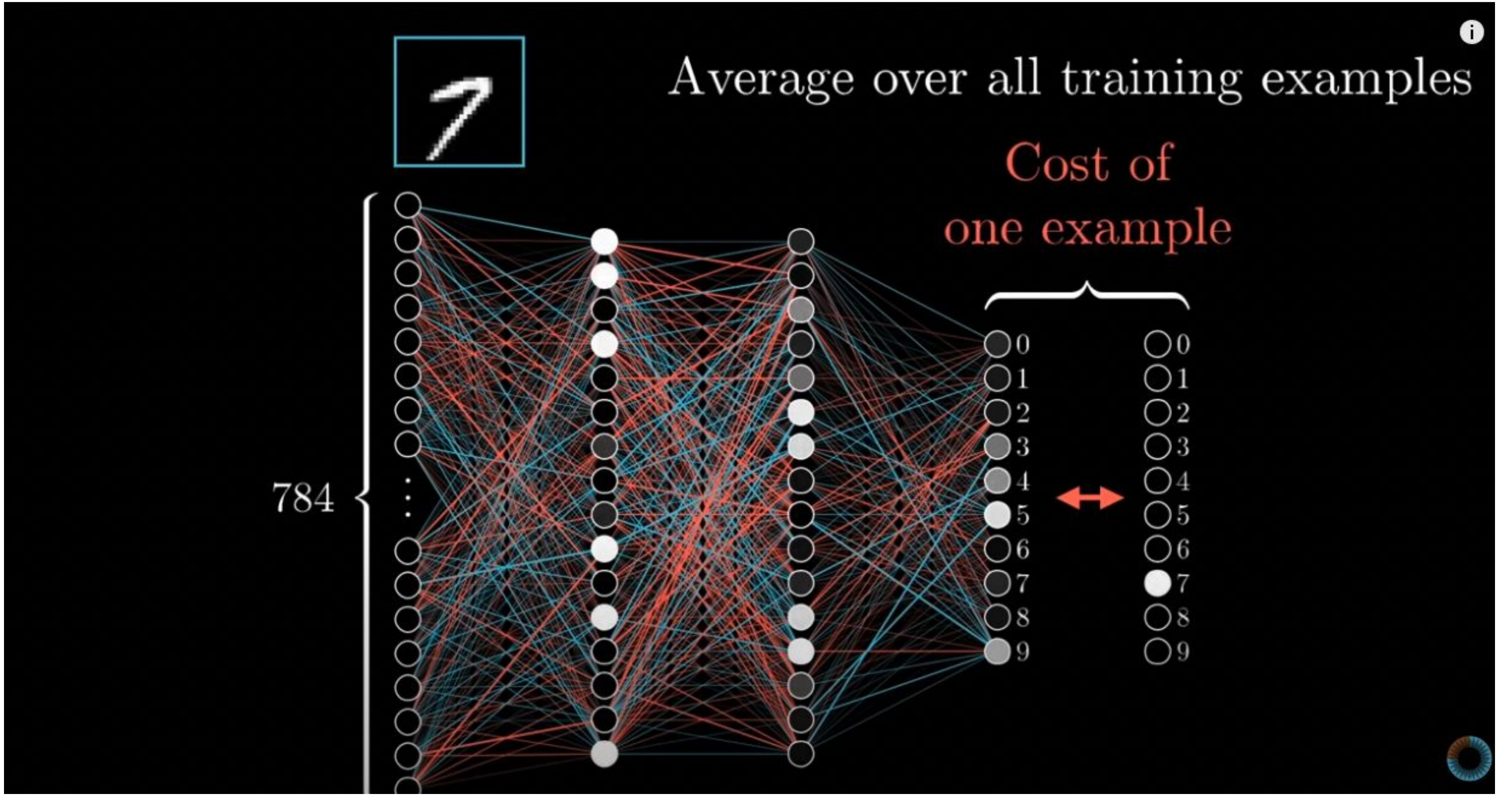
$$\mathbf{y} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$$
$$\epsilon = \sum \text{abs}(\mathbf{y} - \hat{\mathbf{y}})$$

where **W** and **b** are the parameters.

Backpropagation



Backpropagation



Backpropagation (Stochastic Gradient Descent)

- Dataset
- Sample N random examples: **Minibatch**
- For each example:
  - Compute output of Network
  - Compute error
  - Compute how to change **Weights** and **Biases** to decrease error: Diffs/”Gradients”

Backpropagation (Stochastic Gradient Descent)

- Dataset
- Sample N random examples: **Minibatch**
- For each example:
  - Compute output of Network
  - Compute error
  - Compute how to change **Weights** and **Biases** to decrease error: Diffs/”Gradients”
- After having done that for the Minibatch
  - Calculate mean of all gradients and apply the changes to the parameters
  - Usually gradients are multiplied by a learning rate, e.g., lr = 0.0001

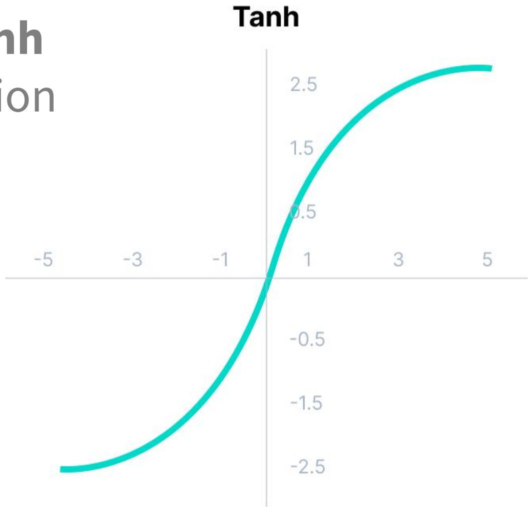
Backpropagation (Stochastic Gradient Descent)

- Main Problems:

  Too big or too small gradients!

Backpropagation

- LeNet 5 uses **Tanh** activation function

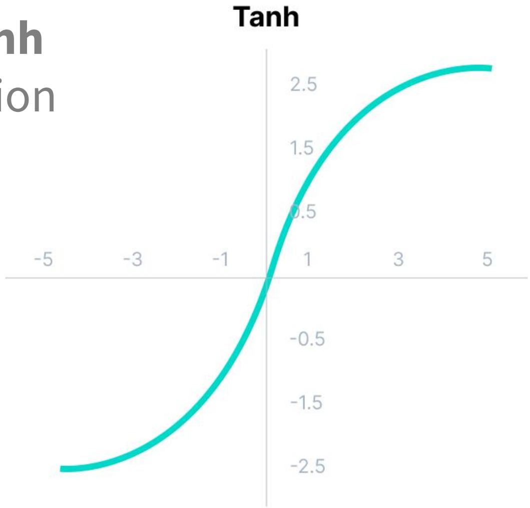


Yann LeCun

Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner:
Gradient-based learning applied to document recognition. Proc. IEEE 86(11): 2278-2324 (1998)

Backpropagation

- LeNet 5 uses **Tanh** activation function



x = 10

Gradient ≈ 0

Yann LeCun

Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner:
Gradient-based learning applied to document recognition. Proc. IEEE 86(11): 2278-2324 (1998)

Backpropagation

Main problems training Deep Learning models

- Vanishing Gradients
  - Through saturated non-linearities
- Exploding Gradients
  - Non-careful initialization of weights
  - Through training dynamics (too high learning rate / exploding weights/biases)
- Technical Limitations
  - Hardware restrictions
  - Compute
  - Memory

Backpropagation

- David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. **Learning Representations by back-propagating errors**, Nature, 1986
- Nowadays all NNs are trained that way
- Enabled the rise of Neural Networks!
- Now the only problems left were
  - **exploding/vanishing gradients**
  - **compute limitations** and
  - **insufficient data ;)**

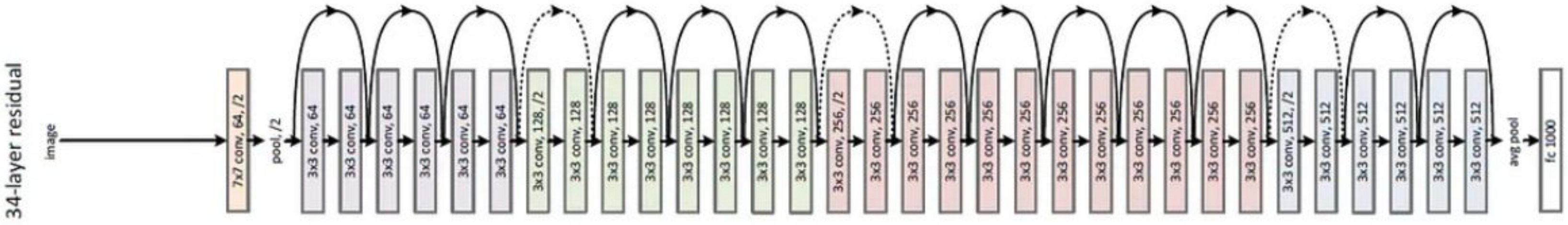
Geoffrey Hinton

Going Deeper

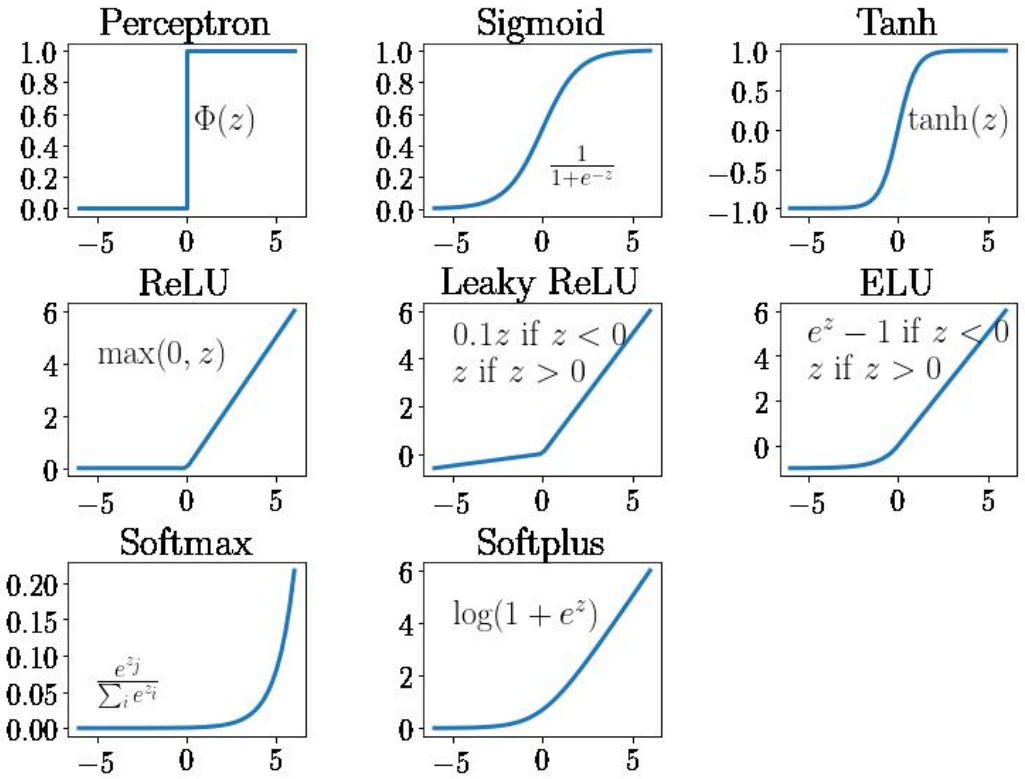
How to overcome training limitations

Deep Architectures

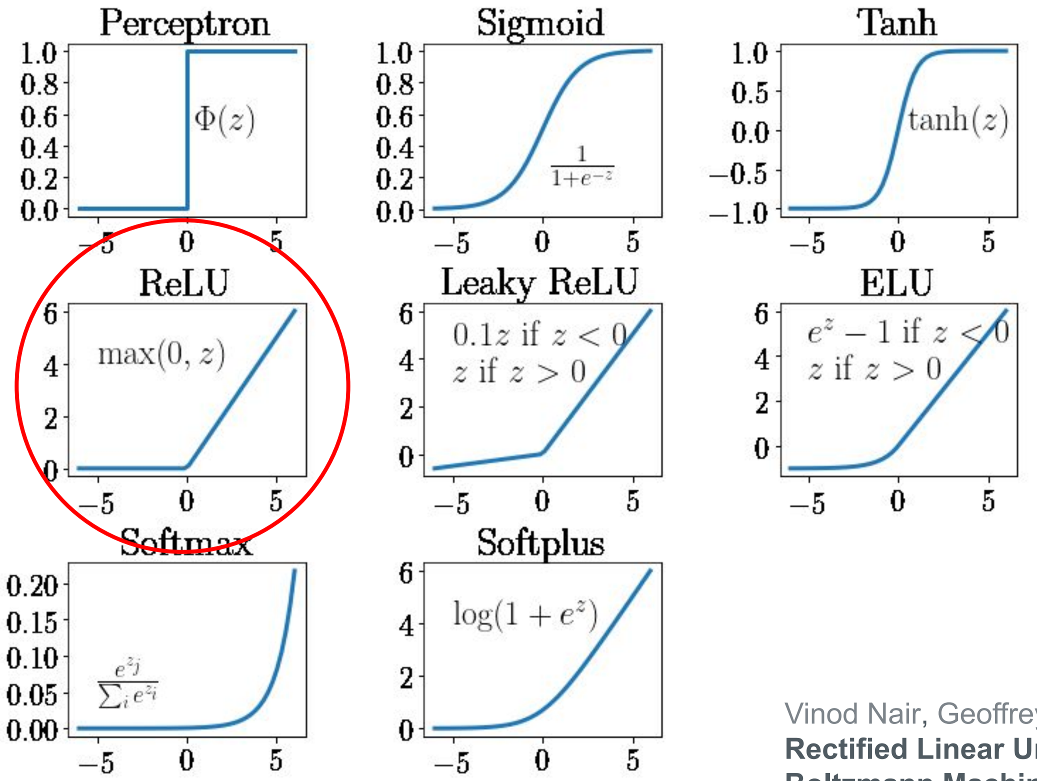
We want gradients of similar magnitudes for parameters of all Layers!



Activation Functions



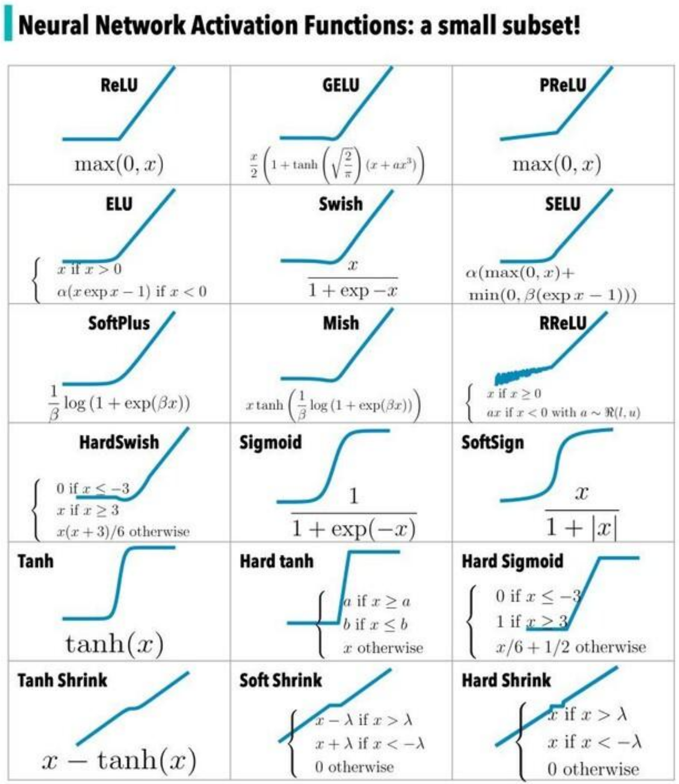
Activation Functions



Geoffrey Hinton

Vinod Nair, Geoffrey E. Hinton:
Rectified Linear Units Improve Restricted Boltzmann Machines. ICML 2010: 807-814

Activation Functions



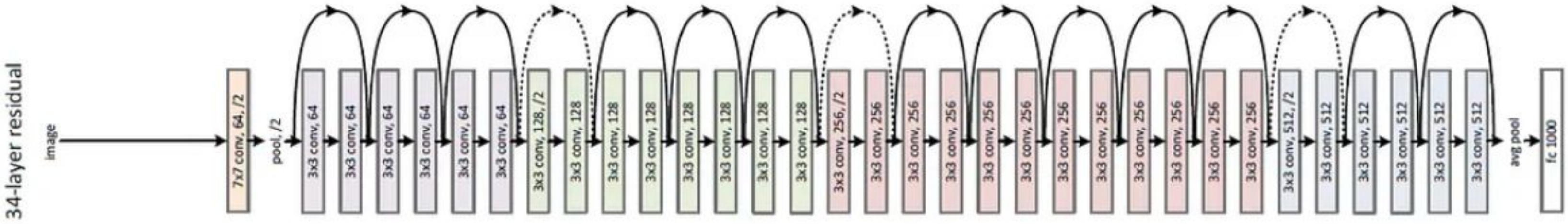
Activation Functions

Why do we need them?

- Theoretical: Universal function approximators
  - Introduce non-linearities
- Practical:
  - E.g., Restrict output between [0, 1] or [-1, 1]
  - Training dynamics
    - (e.g., SELU: Scaled ELUs) converge to standardized distributions
    - Vanishing gradients

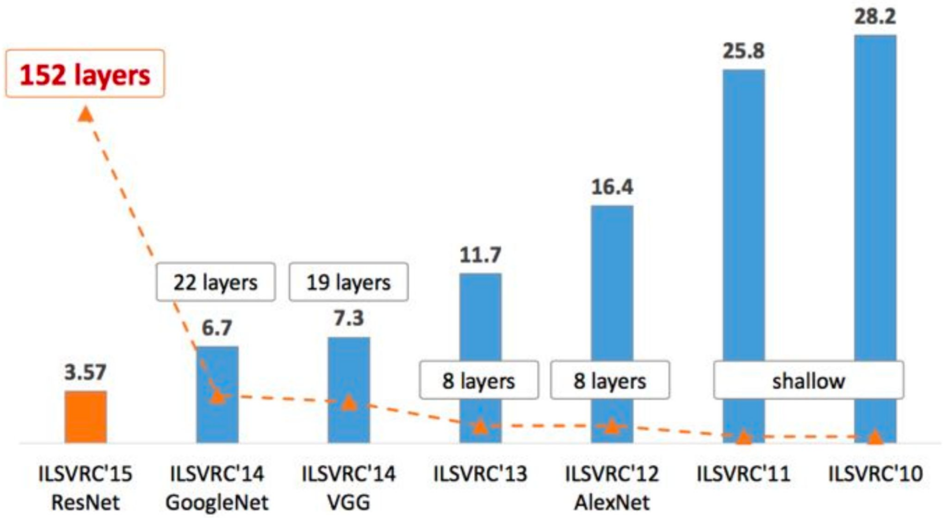
Other improvements

We want gradients of similar magnitudes for parameters of all Layers!



Deep Learning

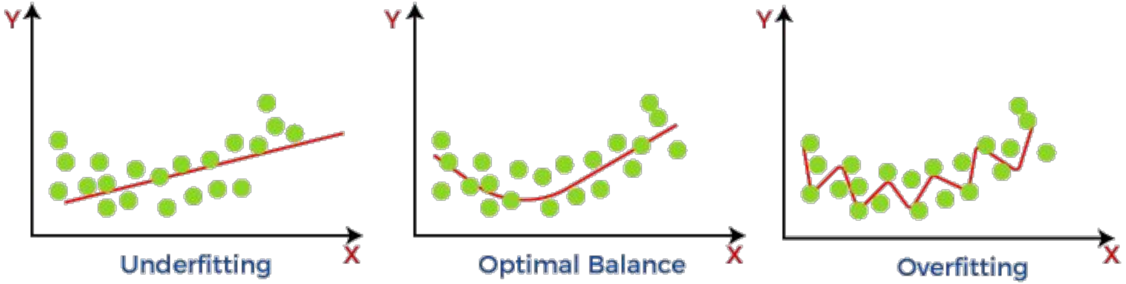
- Became popular as Neural Networks actually became deep
  - approx. 2010 - 2015



Other improvements

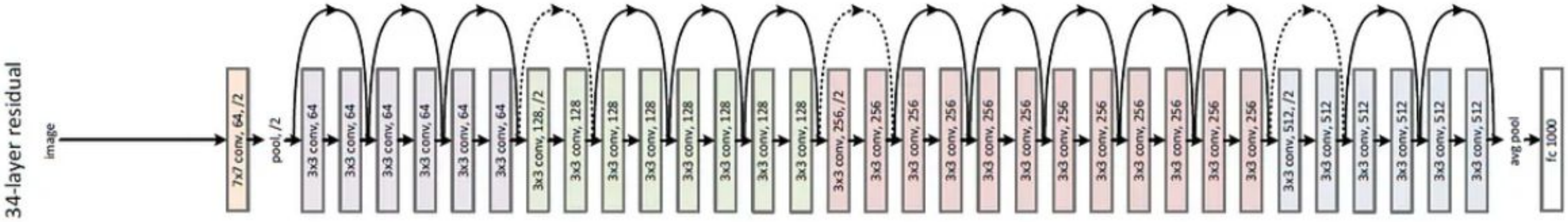
Geoffrey Hinton

- Weight regularization (1992)
    - Additional regularization term
    - Keeping weights small
    - Improve Generalization
- Dropout (2012)
    - Preventing overfitting and allows for over-complete networks
    - Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov: Improving neural networks by preventing co-adaptation of feature detectors. CoRR abs/1207.0580 (2012)
    - Generalization!



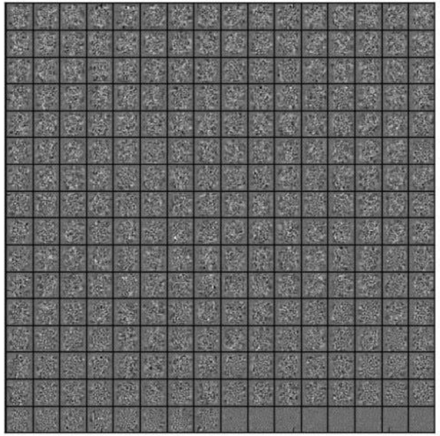
Other improvements (2015!)

- Skip Connections (ResNet, 2015):
    - Bypass one or more layers
    - Combating vanishing gradient problem
    - Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun:
      Deep Residual Learning for Image Recognition. CVPR 2016: 770-778

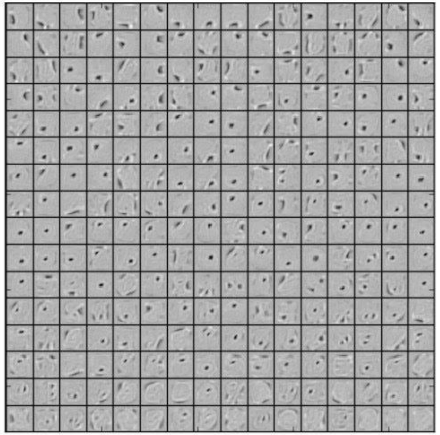


Other improvements (2015!)

- Batch Normalization (2015):
  - Standarized input to each layer (zero mean, unit variance)
  - Sergey Ioffe, Christian Szegedy:
    Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. ICML 2015: 448-456



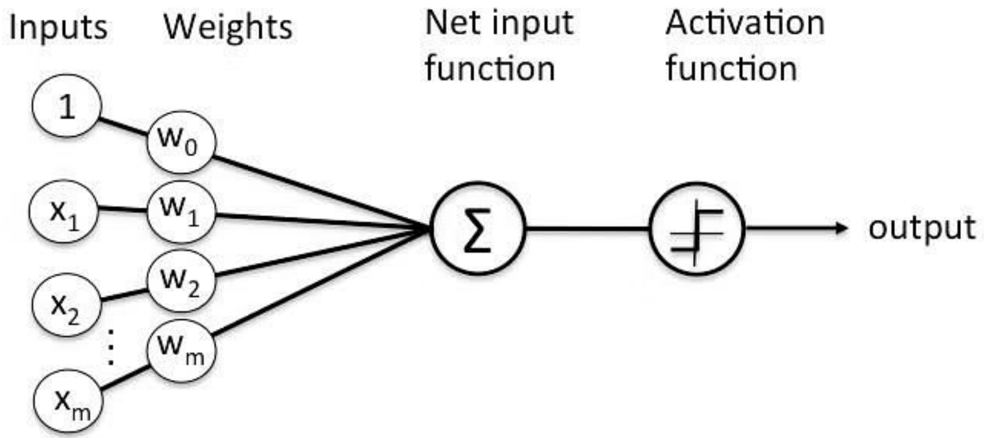
(a) Without dropout



(b) Dropout with $p = 0.5$.

Other improvements (2015!)

- He initialization (2015):
  - A weight initialization method that considers the size of the previous layer
  - Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun:
    Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. ICCV 2015: 1026-1034



Other improvements

- Attention Mechanisms (~2017)
  - Improve capacity to handle long-range dependencies
  - Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin: **Attention is All you Need**. NIPS 2017: 5998-6008
- Gradient Checkpointing (2023)
  - Trades compute time for memory requirement
  - Overcome technical limitation (limited memory)

NN Evolution - Part 1 (Basics) Summary

- Perceptron (1957)
- Multi-Layer Perceptron (1958)
- Convolutional Neural Networks (1982/1998)
- Backpropagation (1986)
- Going Deeper (2010-2015)
  - Non-linearities
  - Dropout
  - Skip Connections
  - Batch Norm
  - Initialization