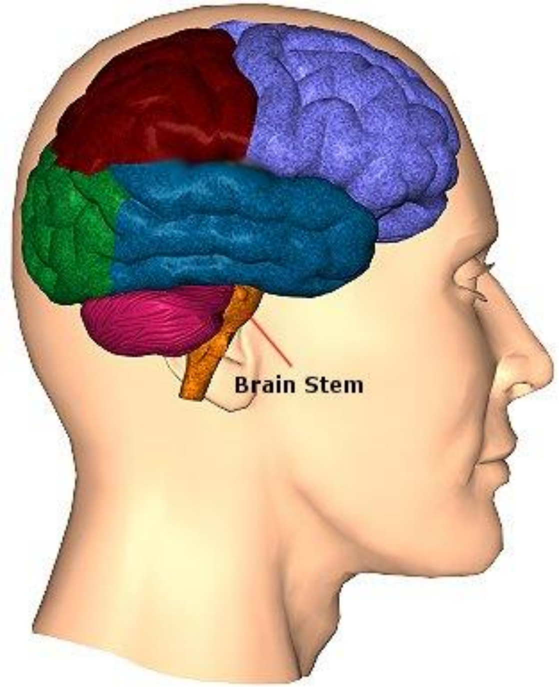
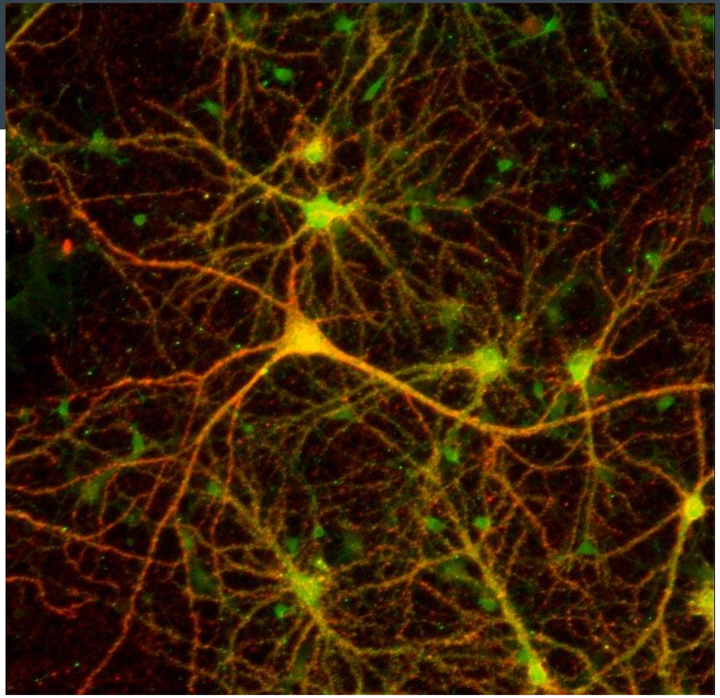
COSI 104a Introduction to machine learning

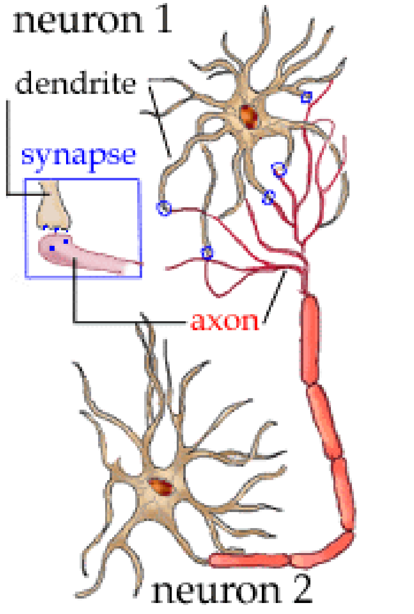
Chapter 10 – NN

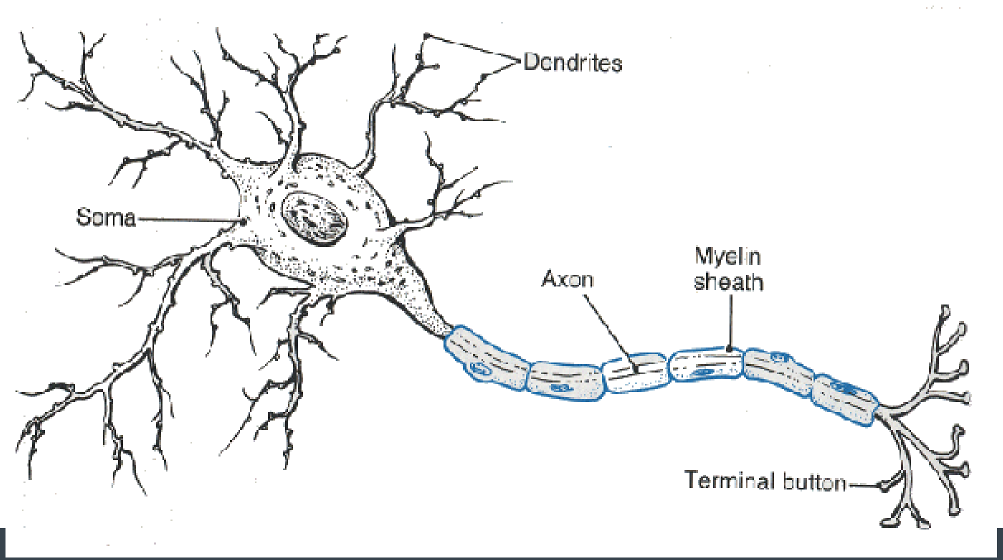
Instructor: Dr. Hongfu Liu
Email: hongfuliu@brandeis.edu

Main source: https://www-users.cs.umn.edu/~kumar001/dmbook/index.php

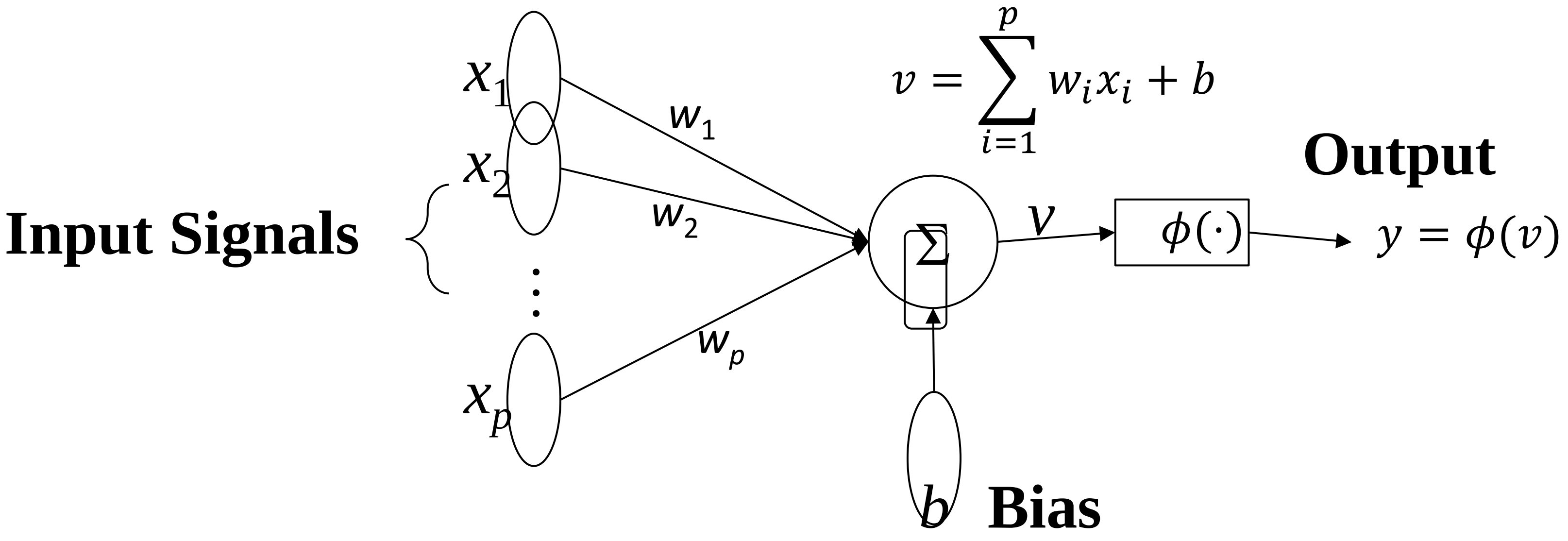

Brain Stem




neuron 1
dendrite
synapse
axon
neuron 2

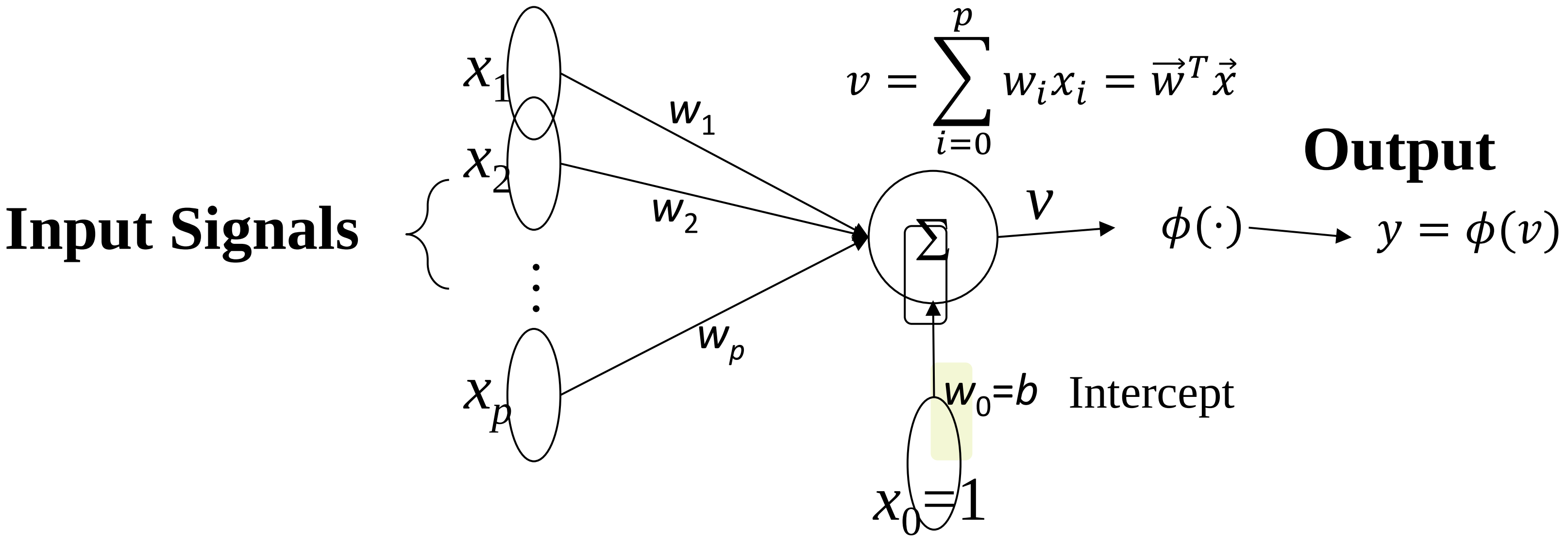

Dendrites
Soma
Axon
Myelin sheath
Terminal button

Mathematical Model of a Neuron



$$y = f(\vec{x}) = \phi\left(\sum_{i=1}^{p} w_i x_i + b\right)$$

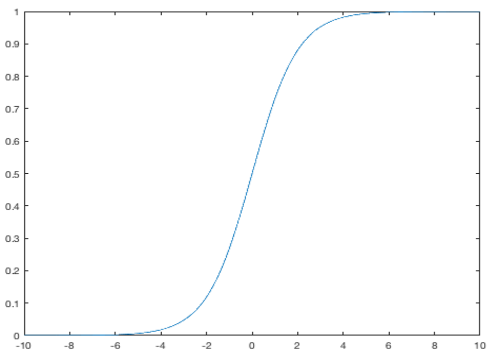
Mathematical Model of a Neuron



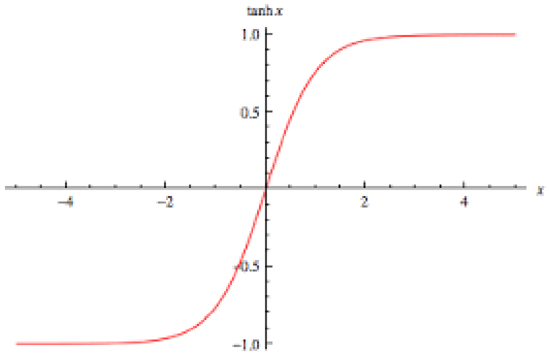
$$y = f(\vec{x}) = \phi(\vec{w}^T \vec{x})$$

Activation Functions

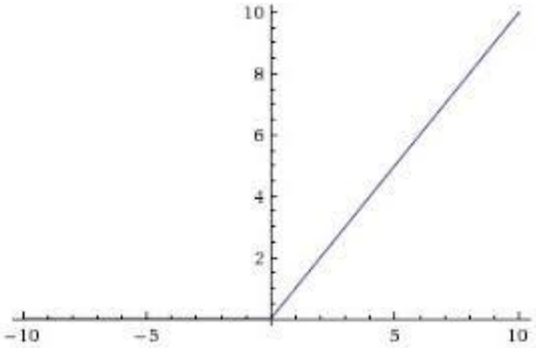
$$f(\vec{x}) = \boldsymbol{\phi}(\vec{w}^T \vec{x})$$



Sigmoid



tanh

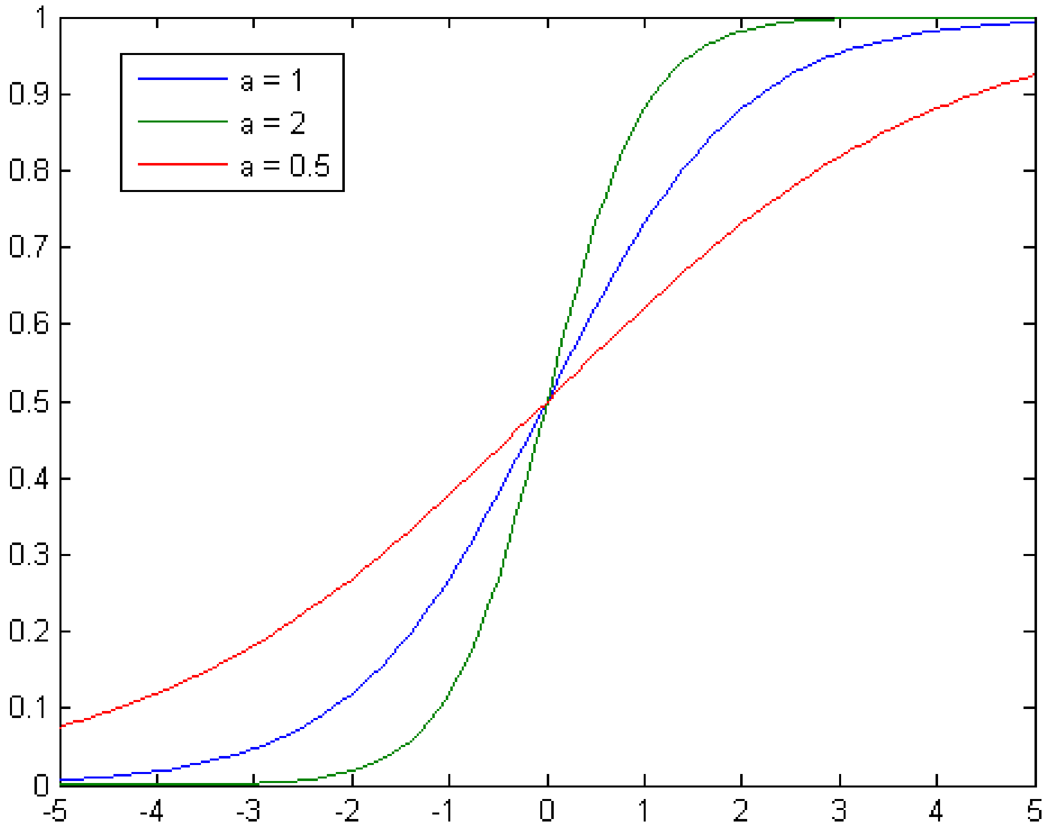


relu

Activation Functions

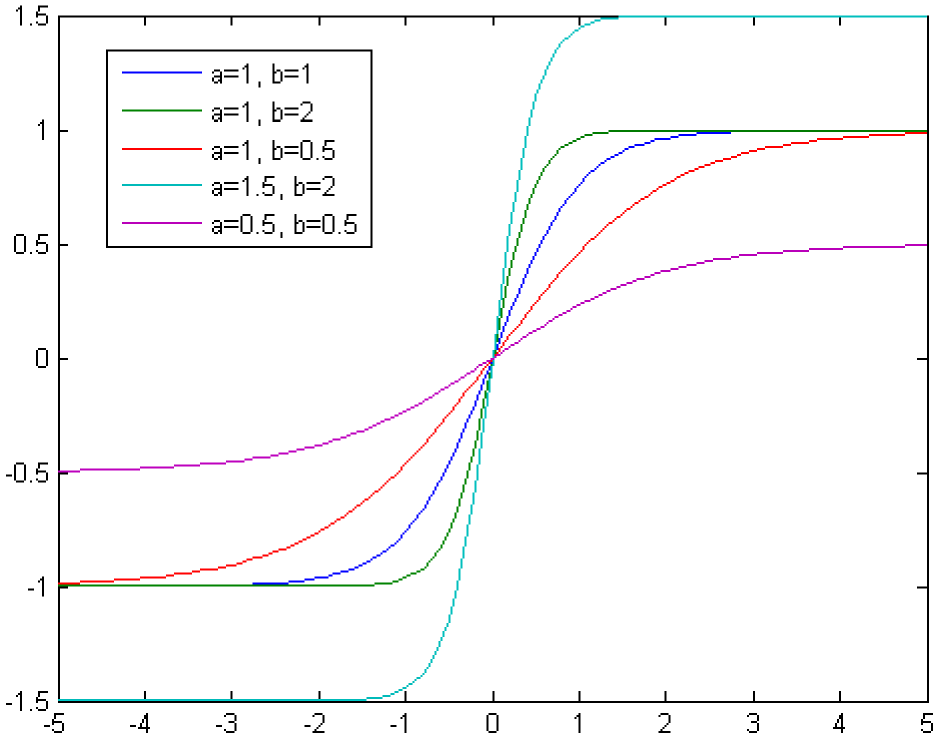
Sigmoid Function

$$\phi(v) = \frac{1}{1+\exp(-av)} \qquad a > 0,\ -\infty < v < \infty$$



Hyperbolic Tangent Function

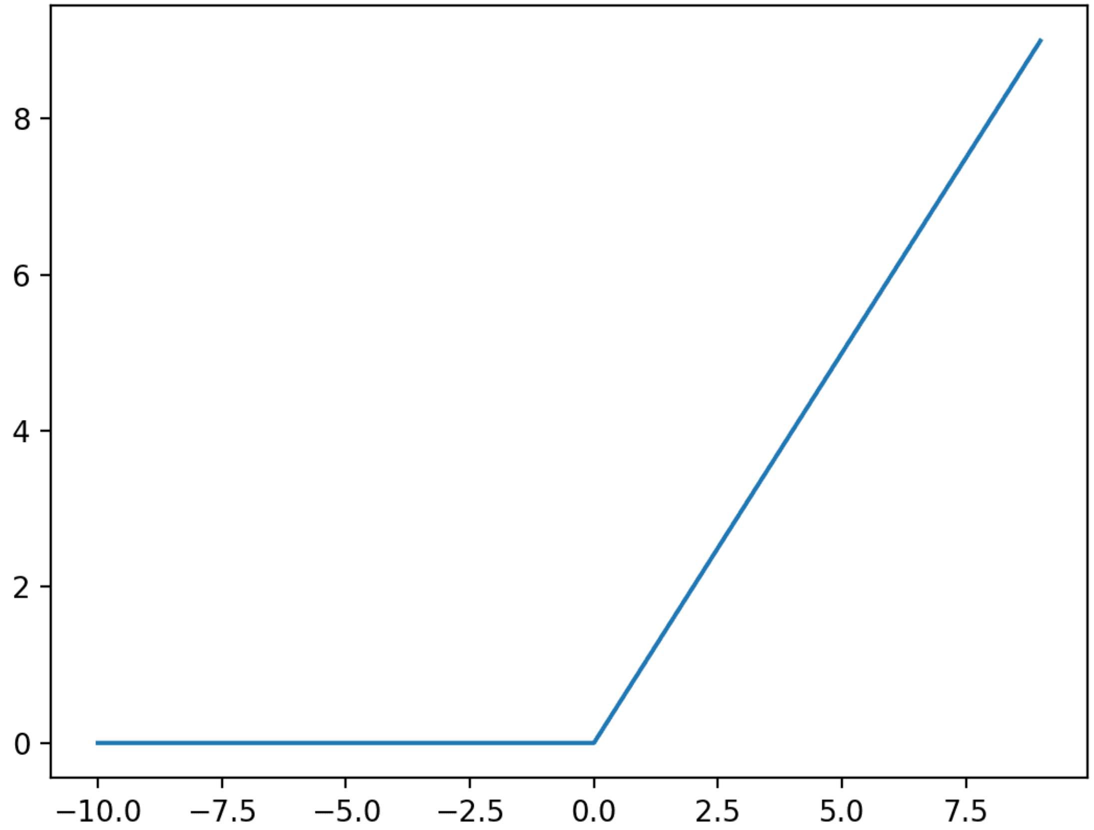
$$\phi(v) = a\tanh(bv) = a\frac{e^{bv} - e^{-bv}}{e^{bv} + e^{-bv}} \qquad a > 0,\ b > 0$$



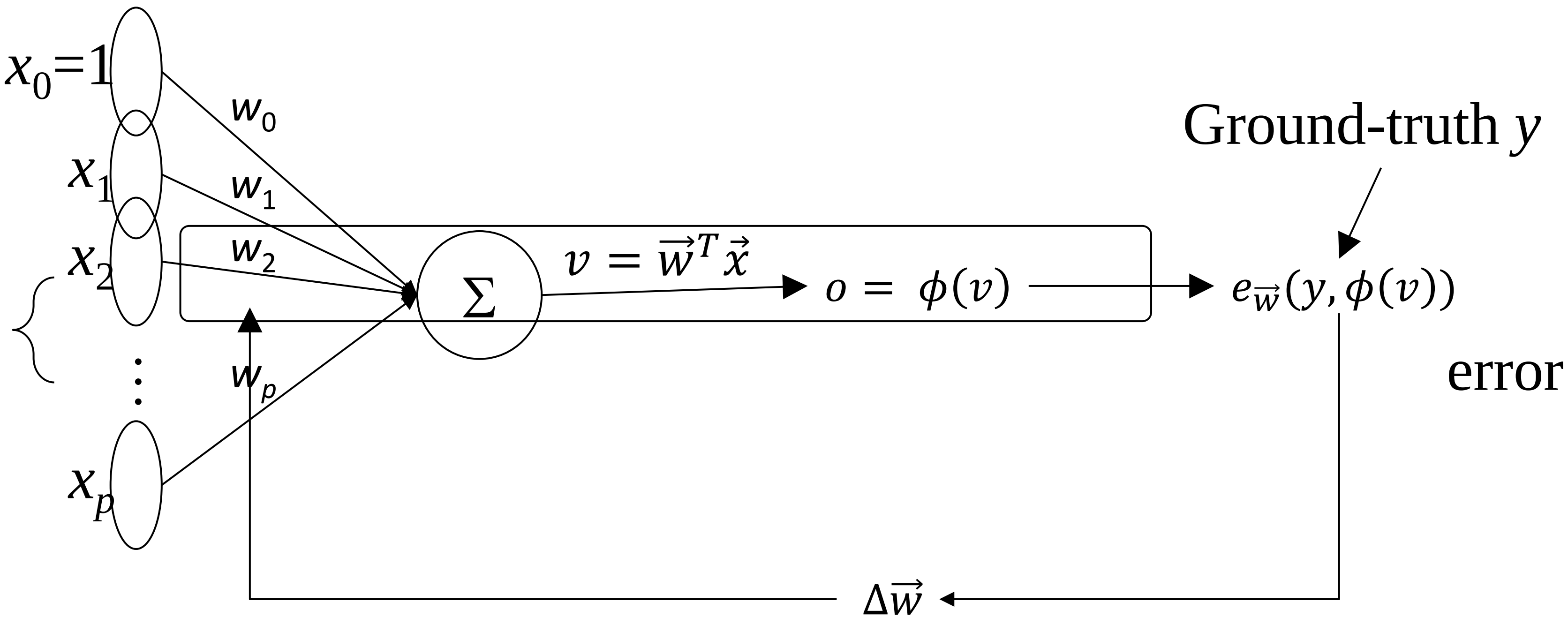
Activation Functions

ReLU (rectified linear unit)

$$\phi(x) = x^+ = \max(0, x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$



Error-Correction Learning



If the error function is continuously differentiable, we can use gradient descent

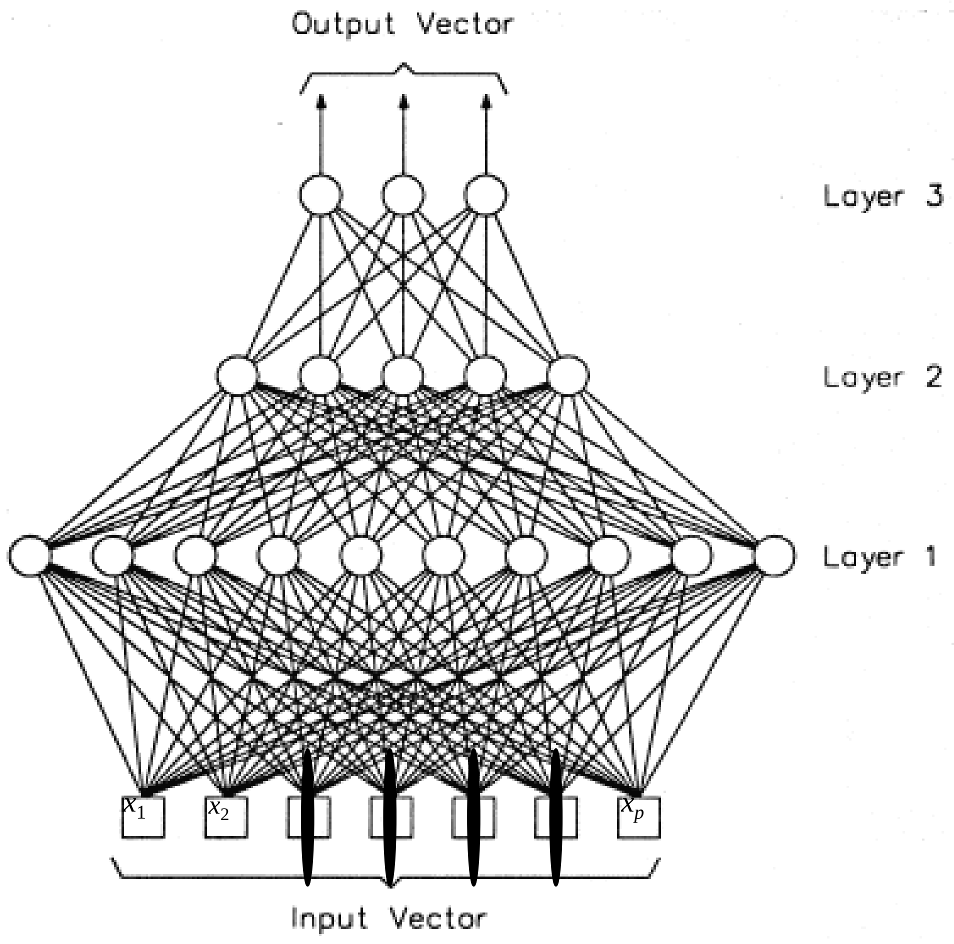
$$\Delta\vec{w} = \eta \frac{\partial e_{\vec{w}}(y, \phi(v))}{\partial \vec{w}}$$

If the activation function ϕ is differentiable, the necessary condition for minimize the error is

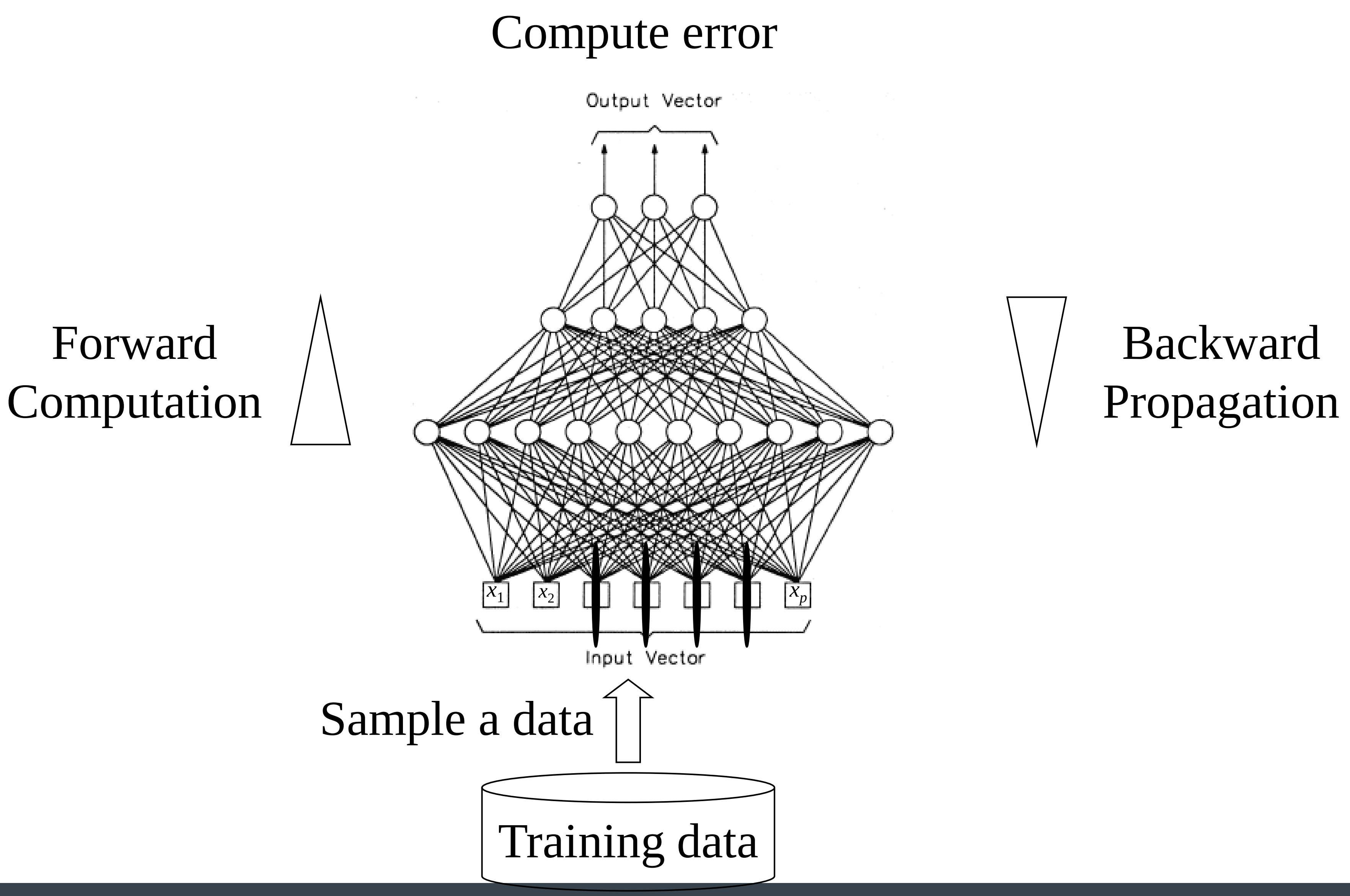
$$\frac{\partial e_{\vec{w}}(y,\phi(v))}{\partial \vec{w}} = \begin{bmatrix} \frac{\partial e_{\vec{w}}(y,\phi(v))}{\partial w_0} \\ \frac{\partial e_{\vec{w}}(y,\phi(v))}{\partial w_1} \\ \vdots \\ \frac{\partial e_{\vec{w}}(y,\phi(v))}{\partial w_p} \end{bmatrix} = 0$$

Hence, we can use gradient-based search algorithm to train a neuron.

Multilayer Neural Networks



The Back-Propagation Algorithm



Sequential *vs* Batch

- Sequential mode
- Batch mode

$$Err = \frac{1}{2N} \sum_{n \in samples} \sum_{\substack{j \in output \\ neurons}} e_{nj}^2$$

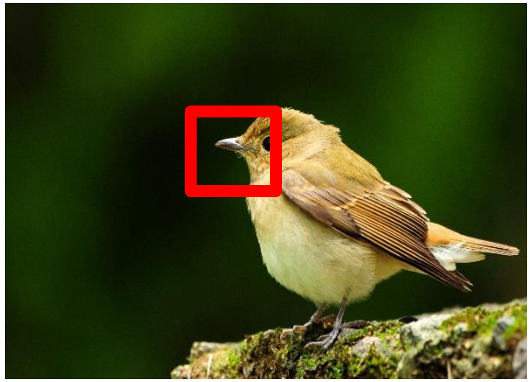
$$\Delta w_{ji} = -\eta \frac{\partial Err(t)}{\partial w_{ji}(t)} = \eta \sum_{n} e_{nj} \frac{\partial e_{nj}}{\partial w_{ji}}$$

Online, memory, implementation, stochastic

Discussion

- Number of hidden neurons
- Sequential training mode
- Batch training mode

Convolution (Conv)



- Local receptive field

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

1	0	1
0	1	0
1	0	1

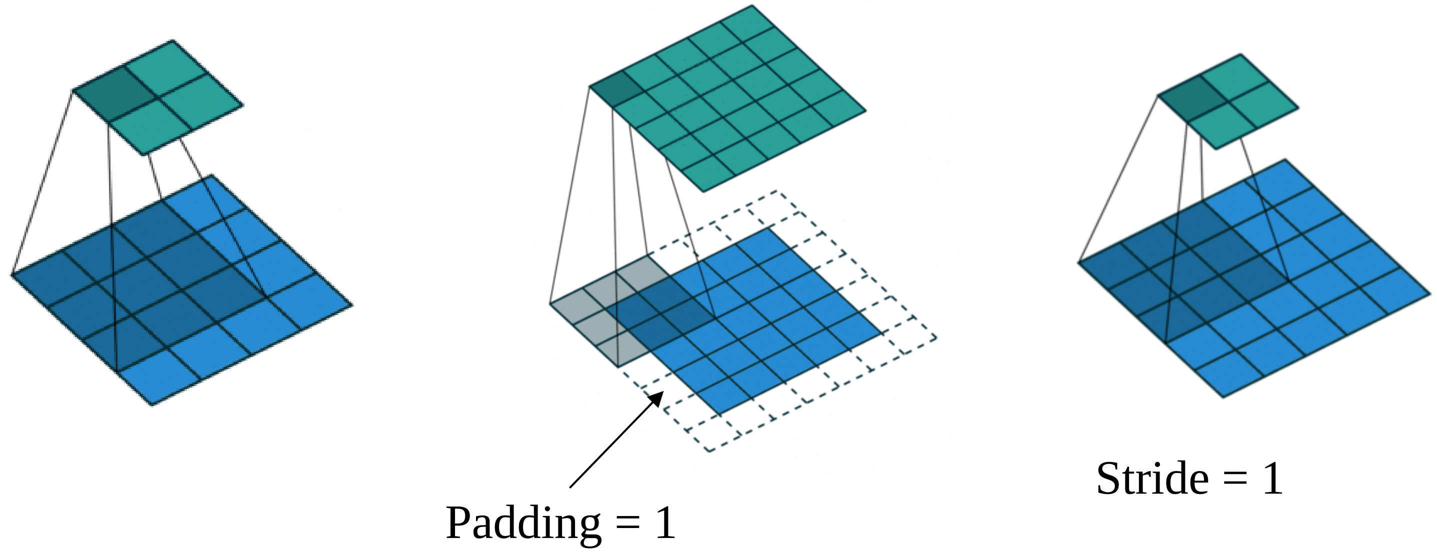
- Spatial invariant

1x1	1x0	1x1	0	0
0x0	1x1	1x0	1	0
0x1	0x0	1x1	1	1
0	0	1	1	0
0	1	1	0	0

4		

- Convolution can be efficiently computed

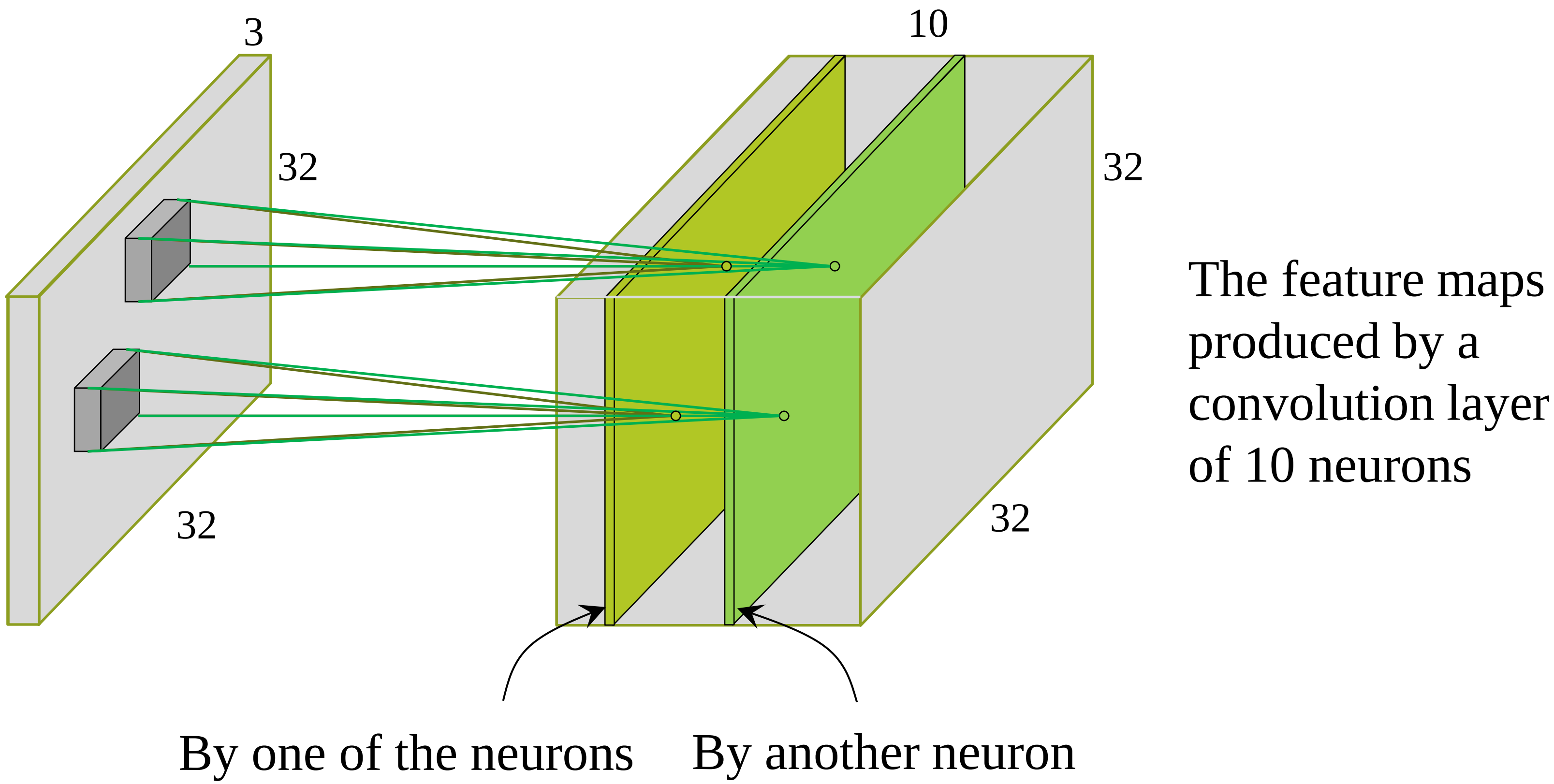
Padding and Stride



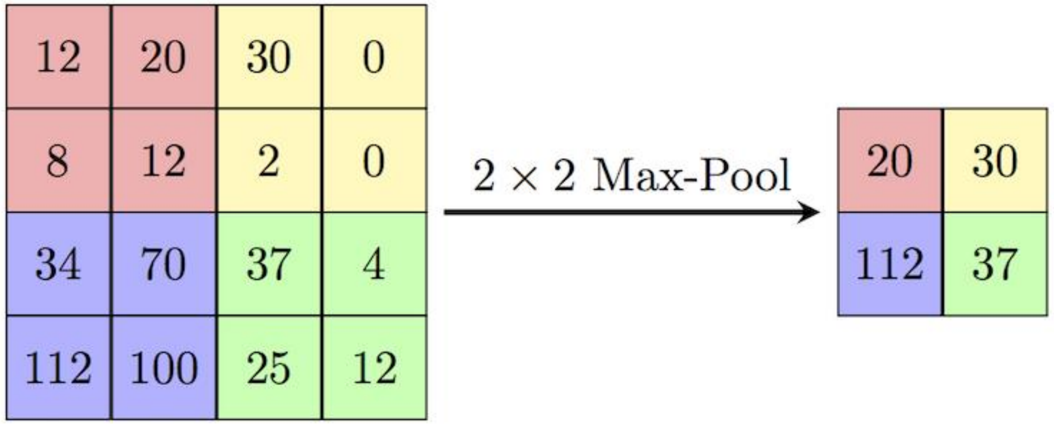
Input size is 4×4, conv filter size is 3×3, padding = 2
What is the output size?

https://github.com/vdumoulin/conv_arithmetic/blob/master/README.md

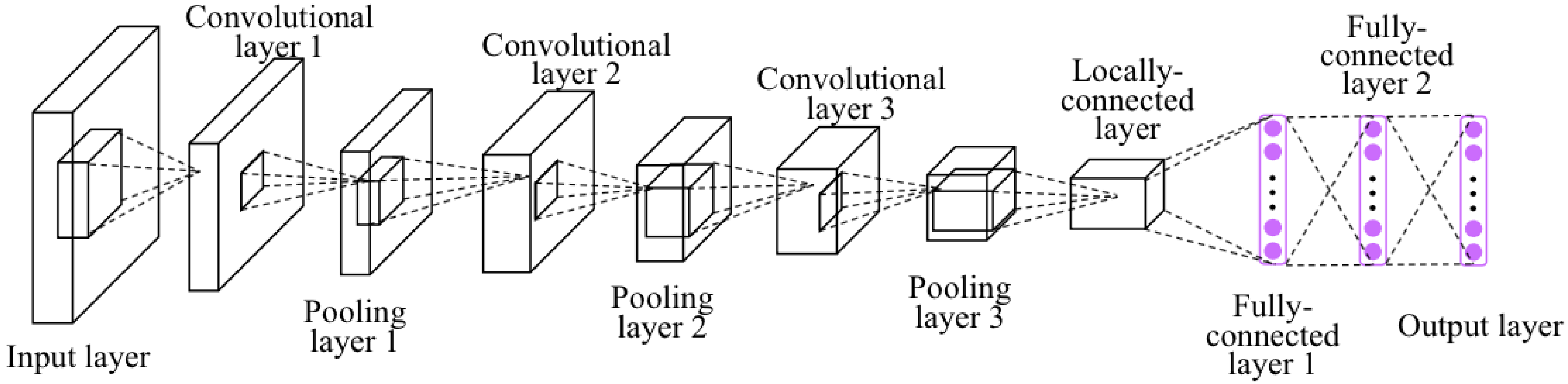
A Conv layer contains multiple convolutional neurons (or filters, feature extractors, …)



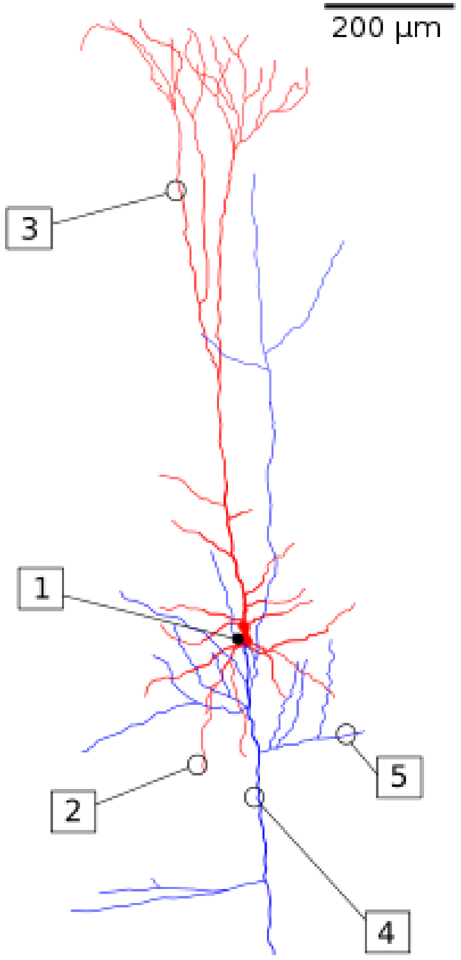
Pooling Layer



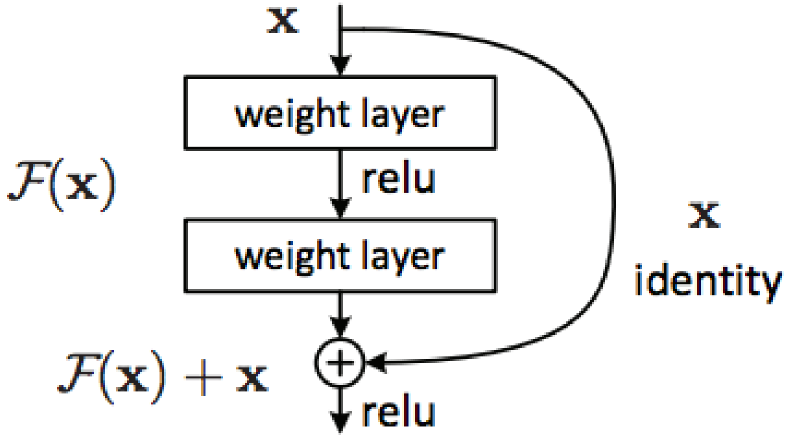
A Deep CNN Example



Residual Neural Network (ResNet)

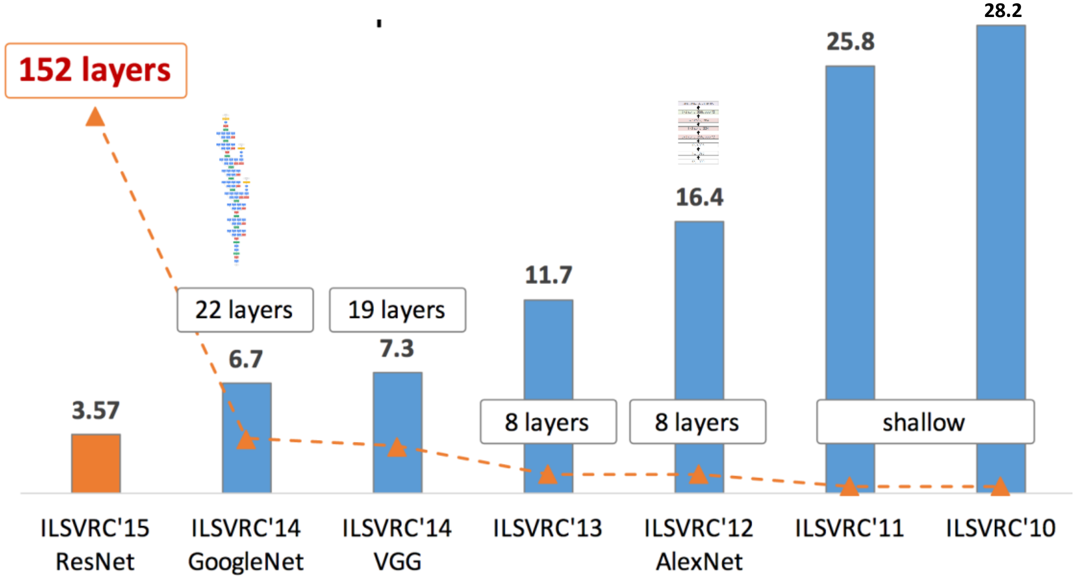


Source: wikipedia



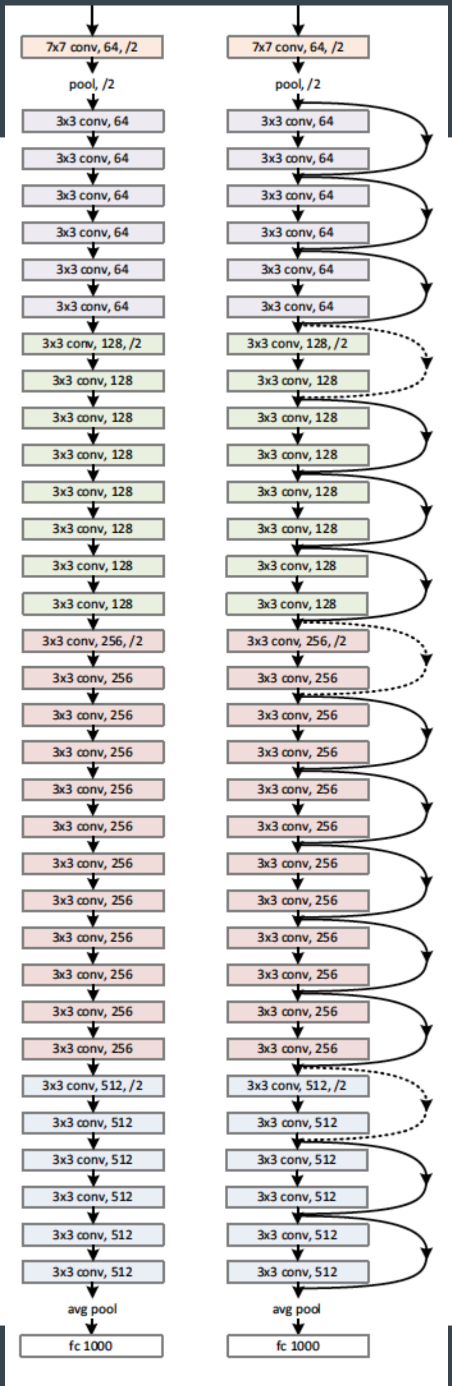
Residual Block

ResNets – Go Deeper



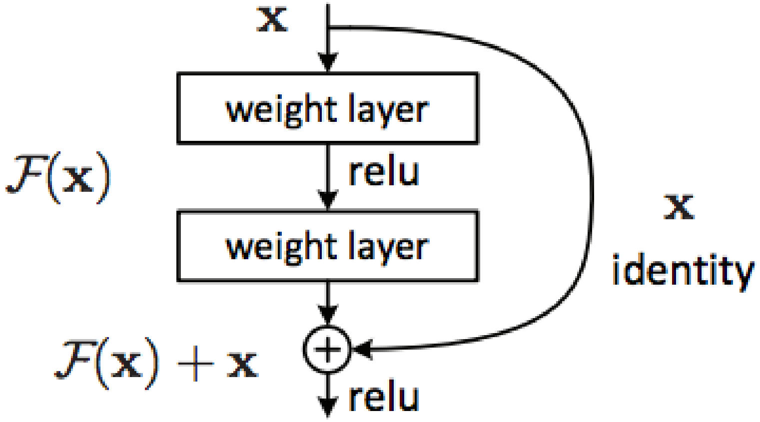
ImageNet top-5 classification error %

Plain

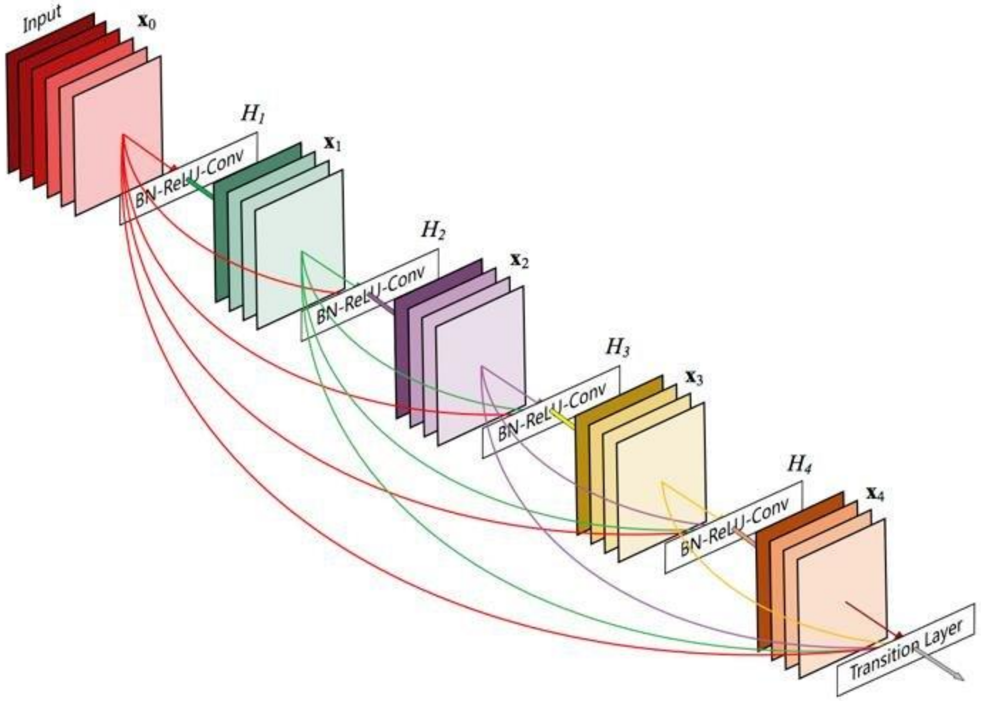


Residual

Densely Connected Convolutional Networks (DenseNet)



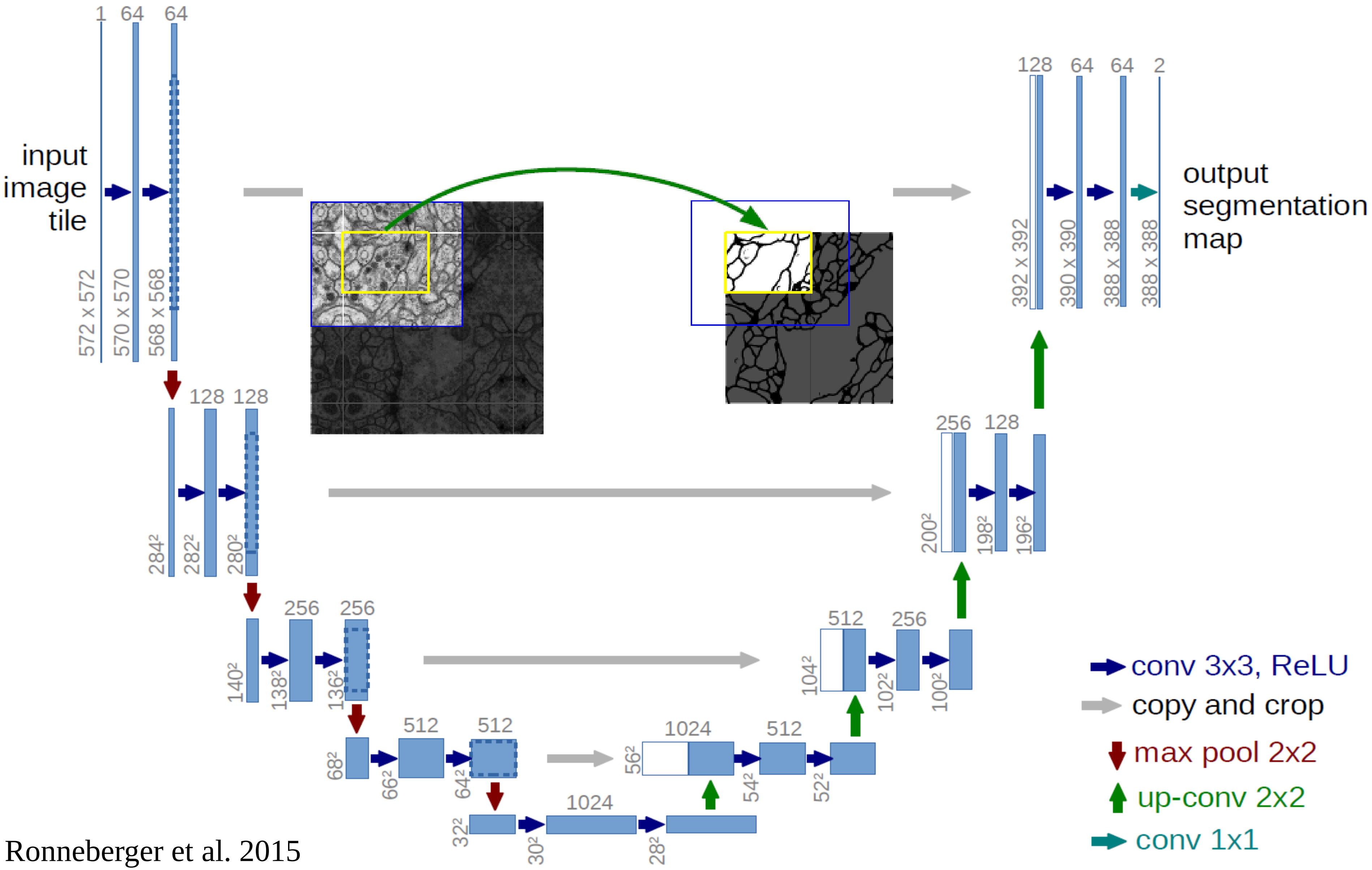
Residual block



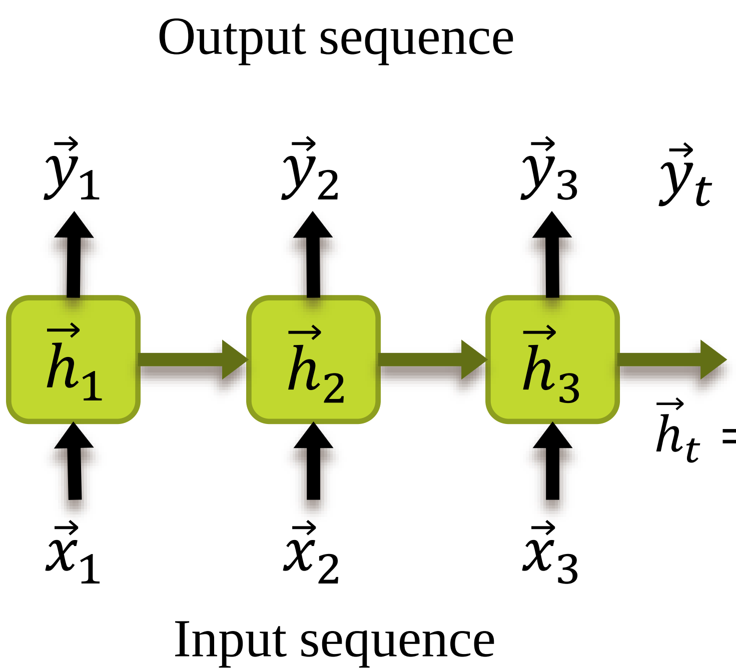
Dense block

Huang et al. 2016

U-Net (for image segmentation)



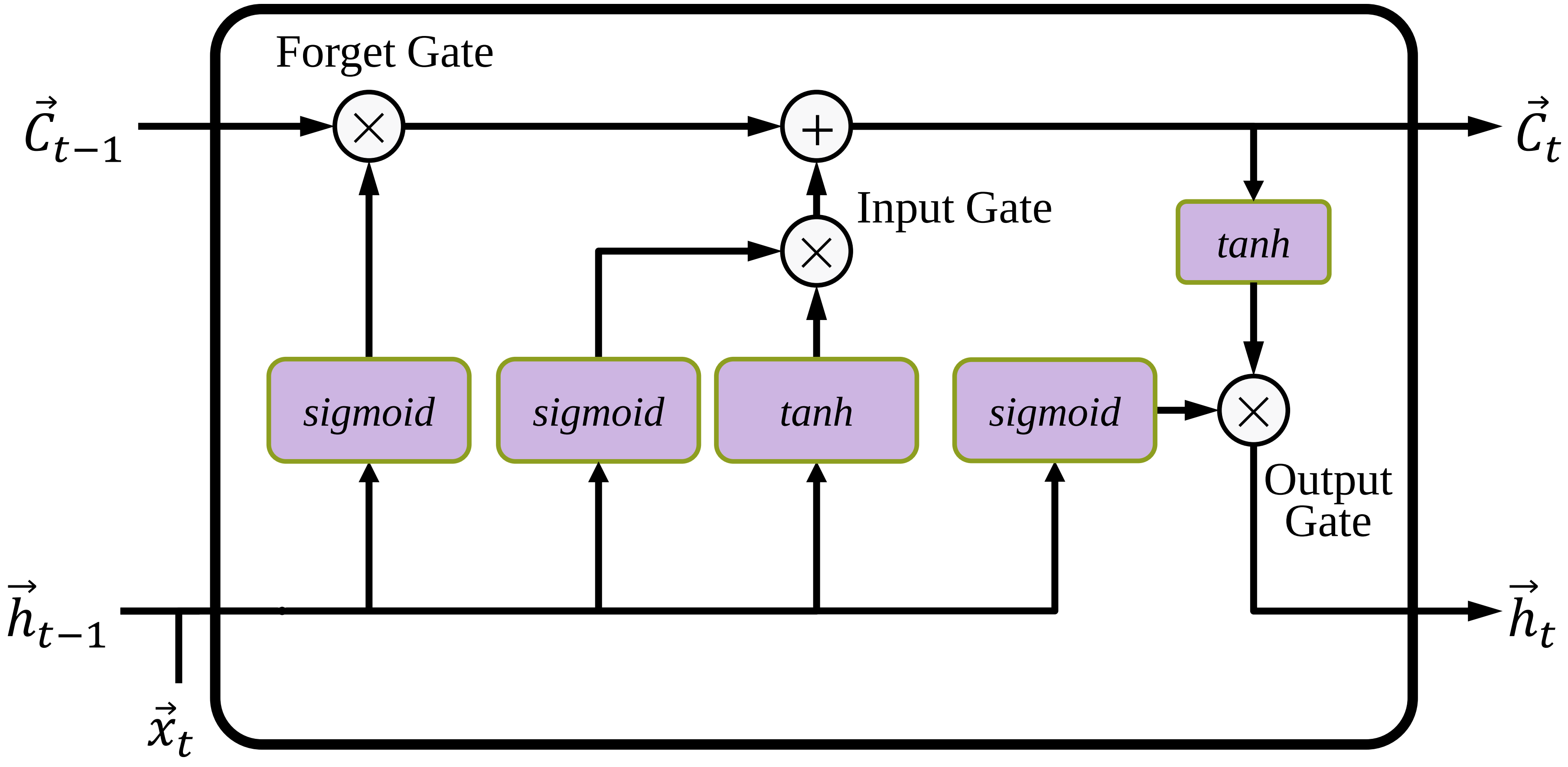
Ronneberger et al. 2015



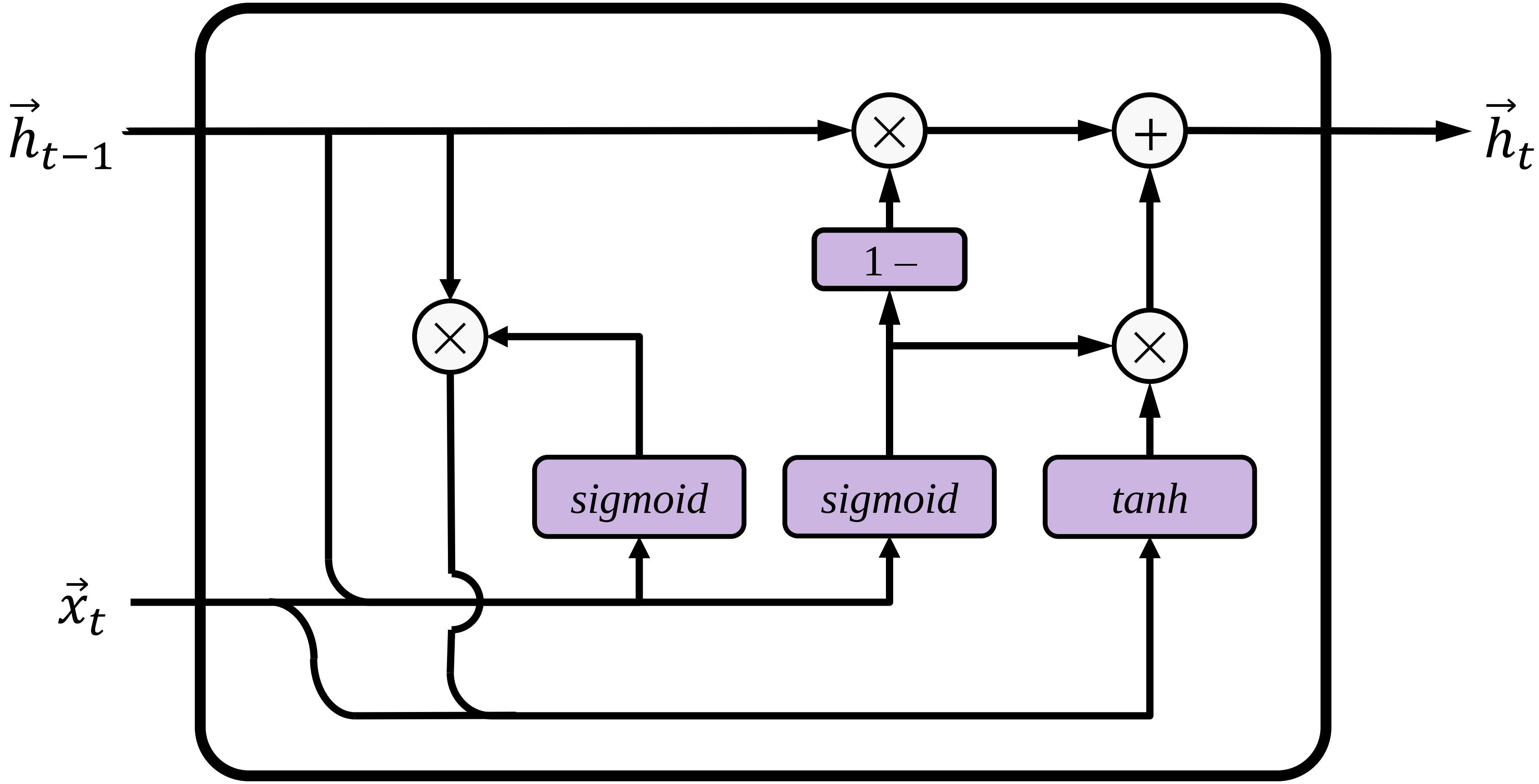
$$\vec{y}_t = \psi(\boldsymbol{W}_2 \vec{h}_t)$$

$$\vec{h}_t = \varphi(\boldsymbol{W}_\mathbf{1}[\vec{x}_t, \vec{h}_{t-1}])$$

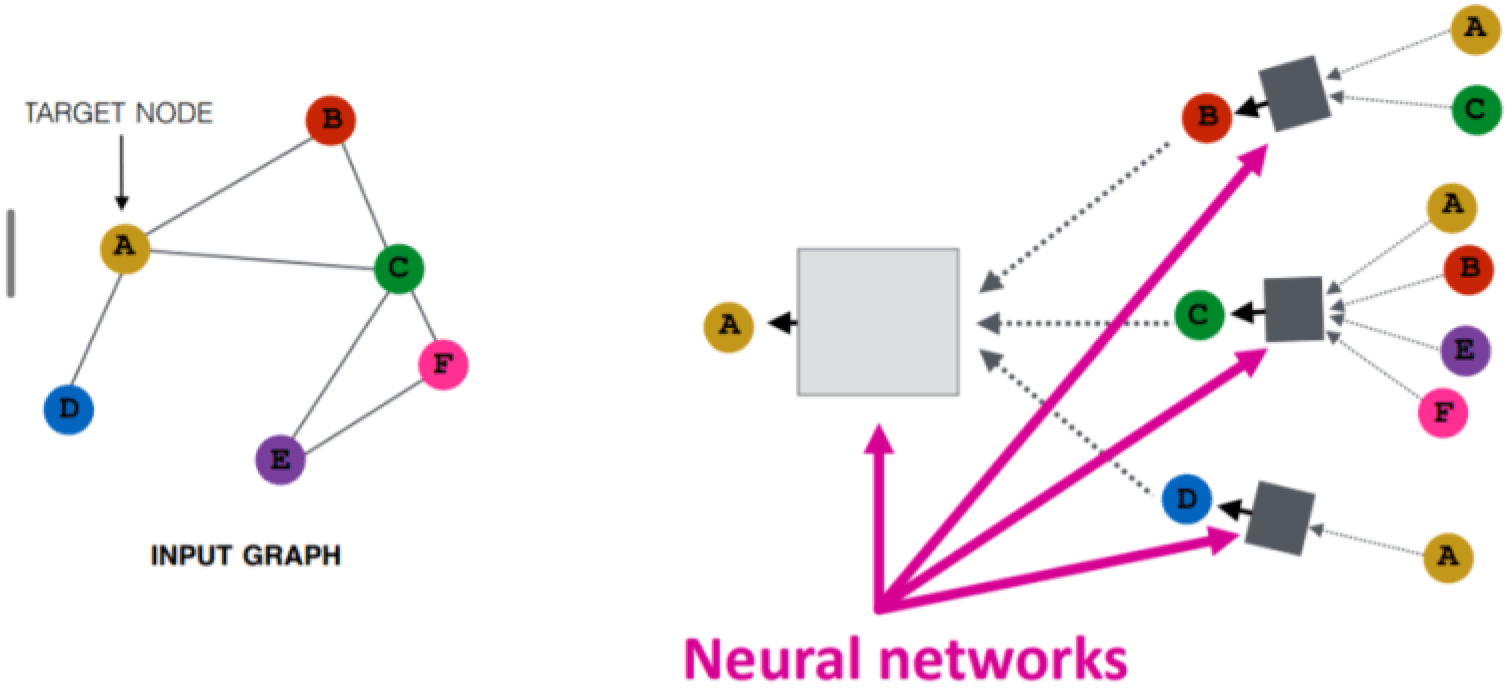
Long Short-Term Memory (LSTM) Cell



Gated Recurrent Unit



GNN



TARGET NODE
A
B
C
D
E
F
INPUT GRAPH
A
B
C
D
A
C
A
B
E
F
A
Neural networks


?
Q
&
A
!