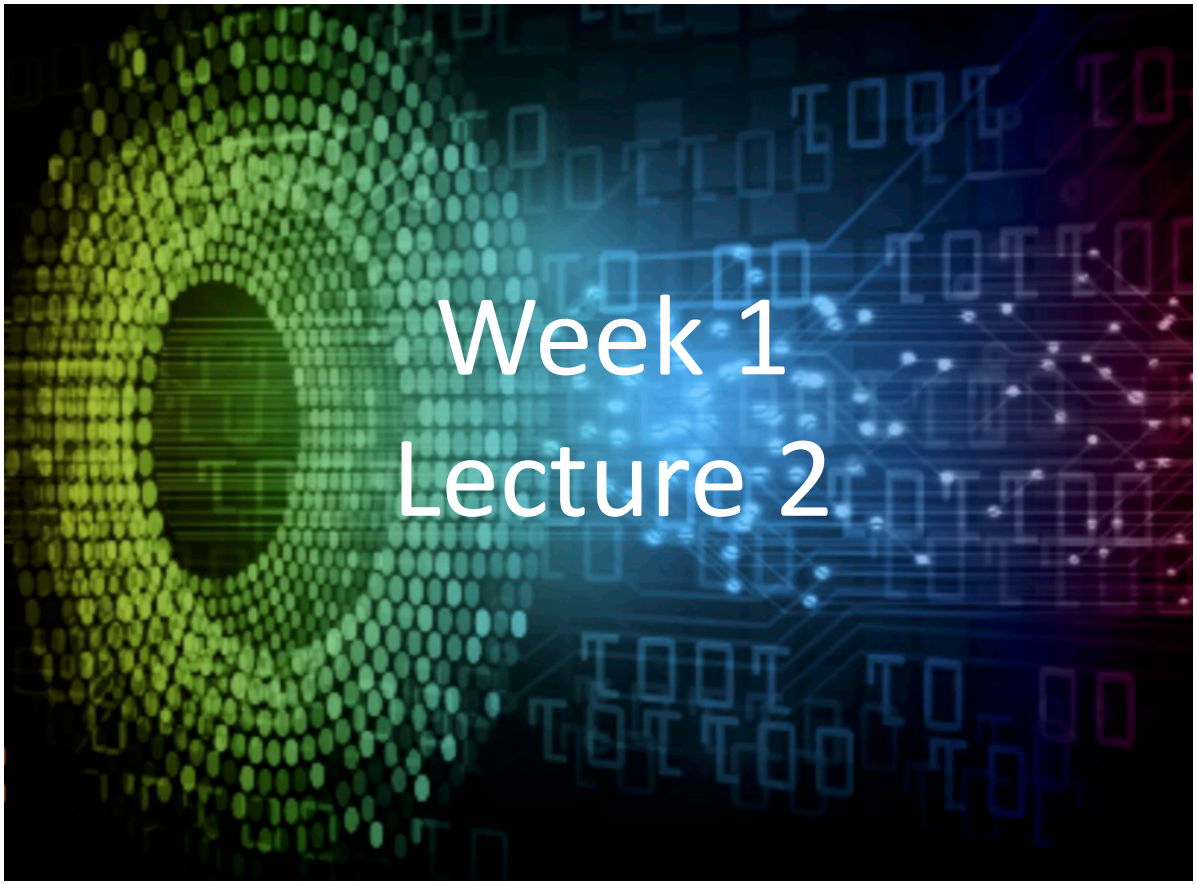
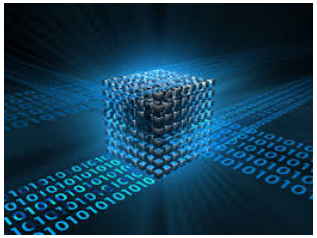
Introduction to Deep Learning
Applications and Theory

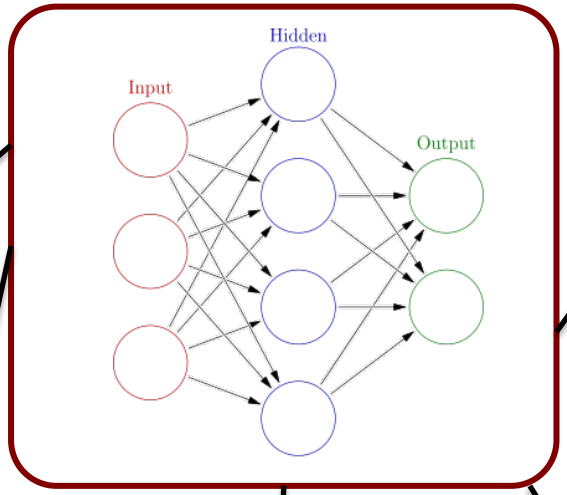


ECE 596 / AMATH 563

Previous Lecture: Artificial Neural Nets (ANN)

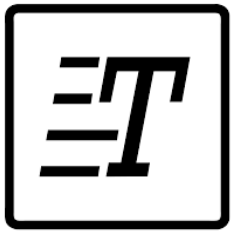


Data Processing

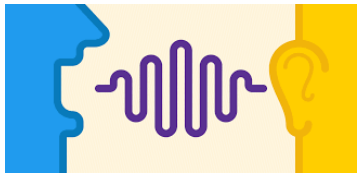


Computer Vision

Navigation



Text Processing



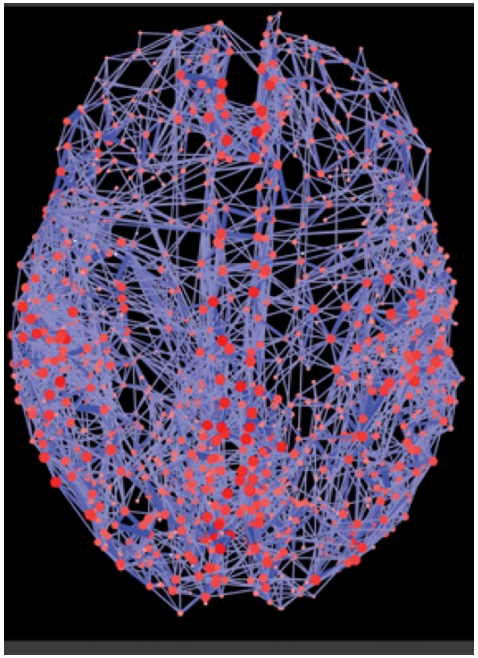
Speech Understanding

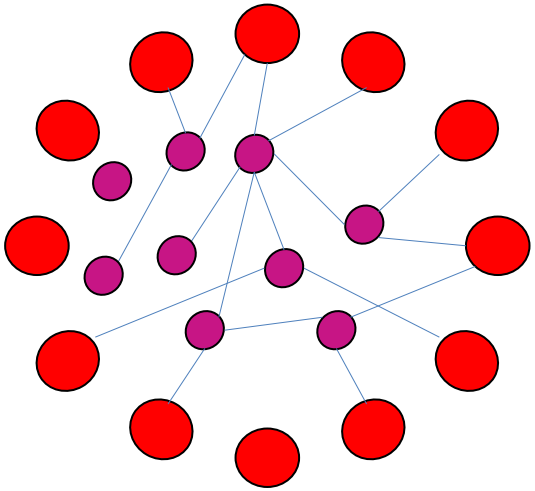
Previous Lecture:
Neural Networks / DL

“Deep Learning is the new electricity"

Andrew Ng

This Lecture: Learning from the Brain

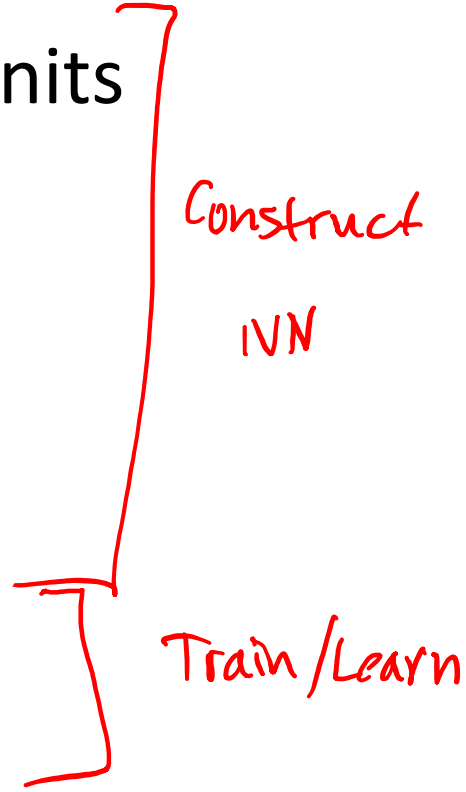




Neurons and Networks

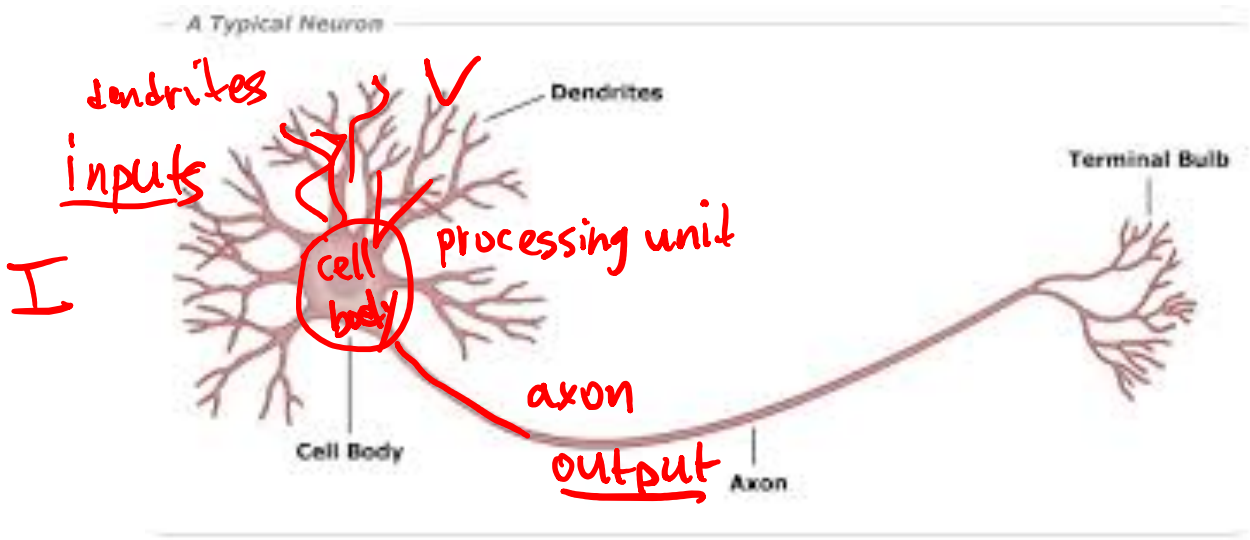
Properties of Brain Networks and Learning

1. Neurons: simple computational units
2. Well connected
3. Parallelism and Recurrence
4. Hierarchically structured
5. Learning
    1. Features
    2. Changing connections



Brain Neuronal Networks

- **Neurons: simple computational units**



Brain Neuronal Networks

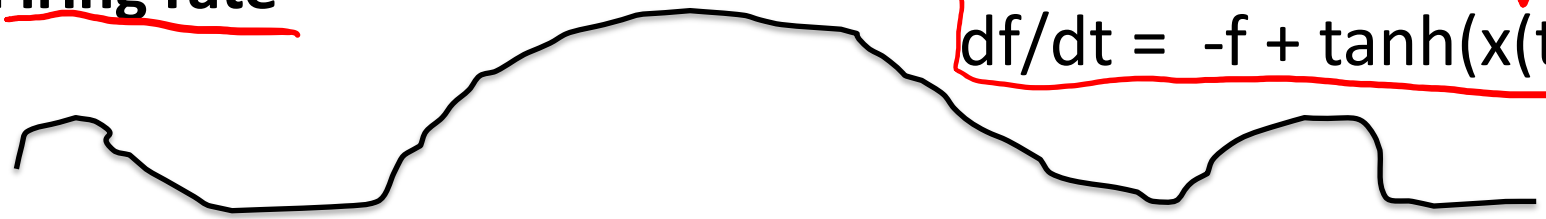
1. Neurons: simple computational units

Spike train

Binary

0	1	0	0	1	1	1	1	1	0						

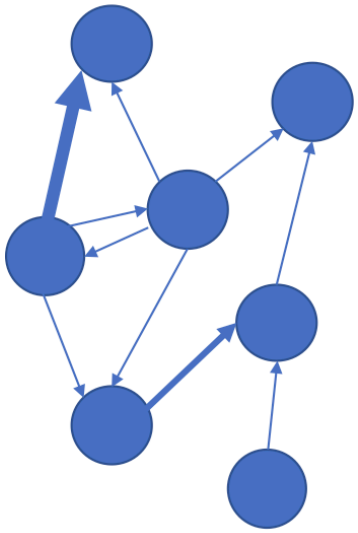
Firing rate



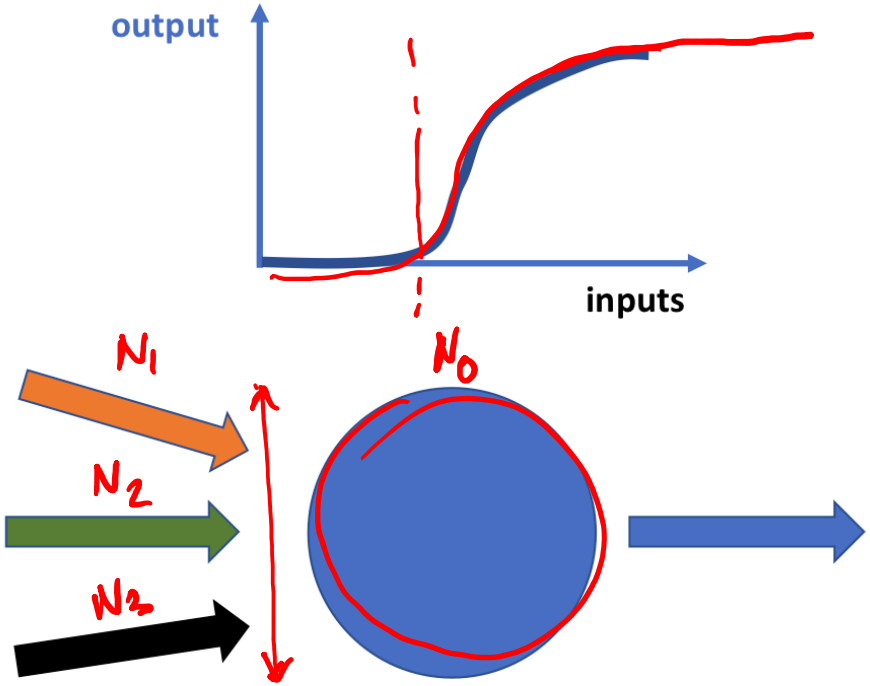
Input

$$df/dt = -f + \tanh(x(t))$$

Firing rate units



Firing-rate model

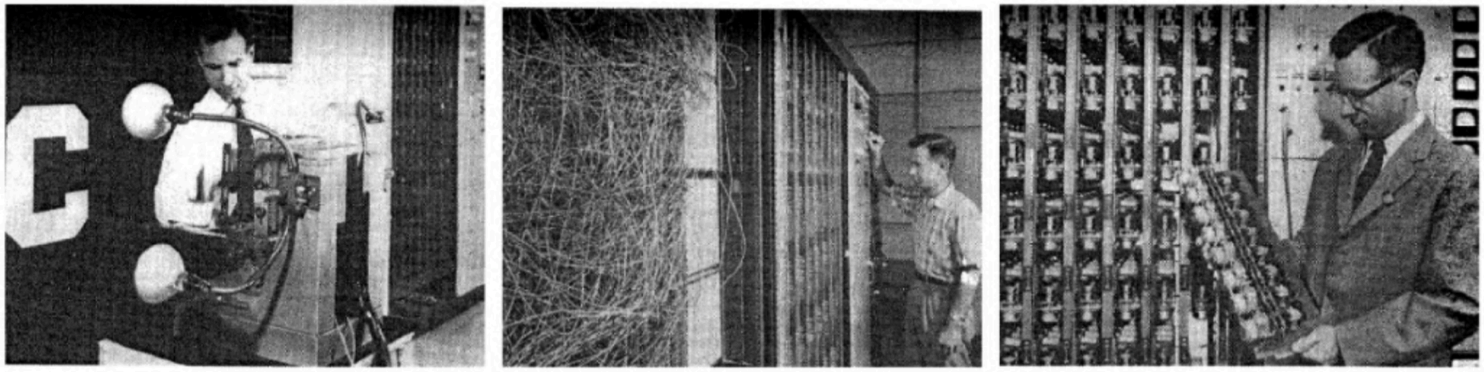


inputs (synapses): from other units

neuron: firing-rate unit

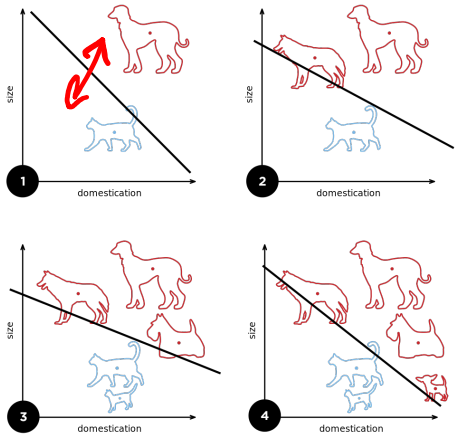
output: to other units

Perceptron



$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

Network trains its weights based on **labeled training** data, i.e. optimizes **cost** function between expected outputs and actual outputs



The perceptron, 1957

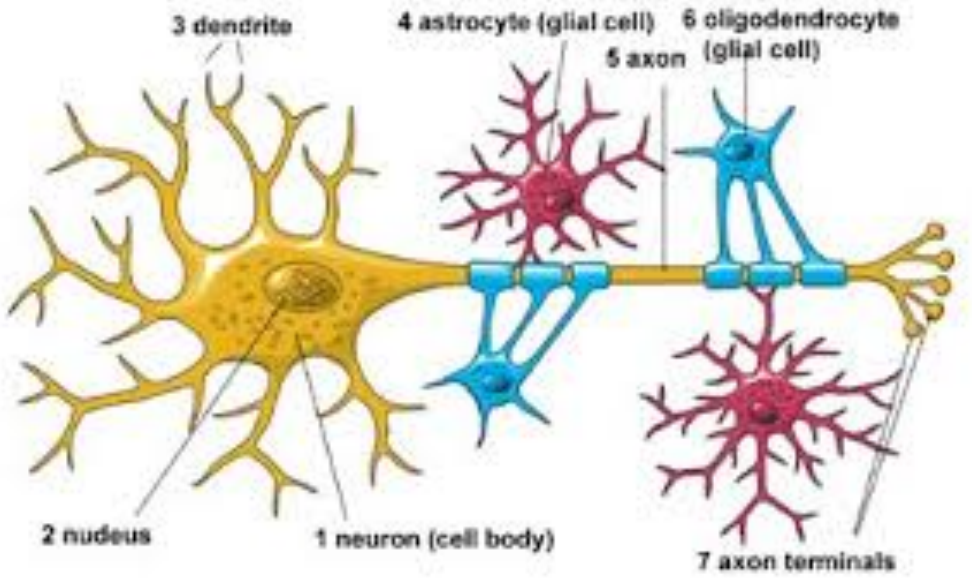
Neuronal Networks

2. Well Connected

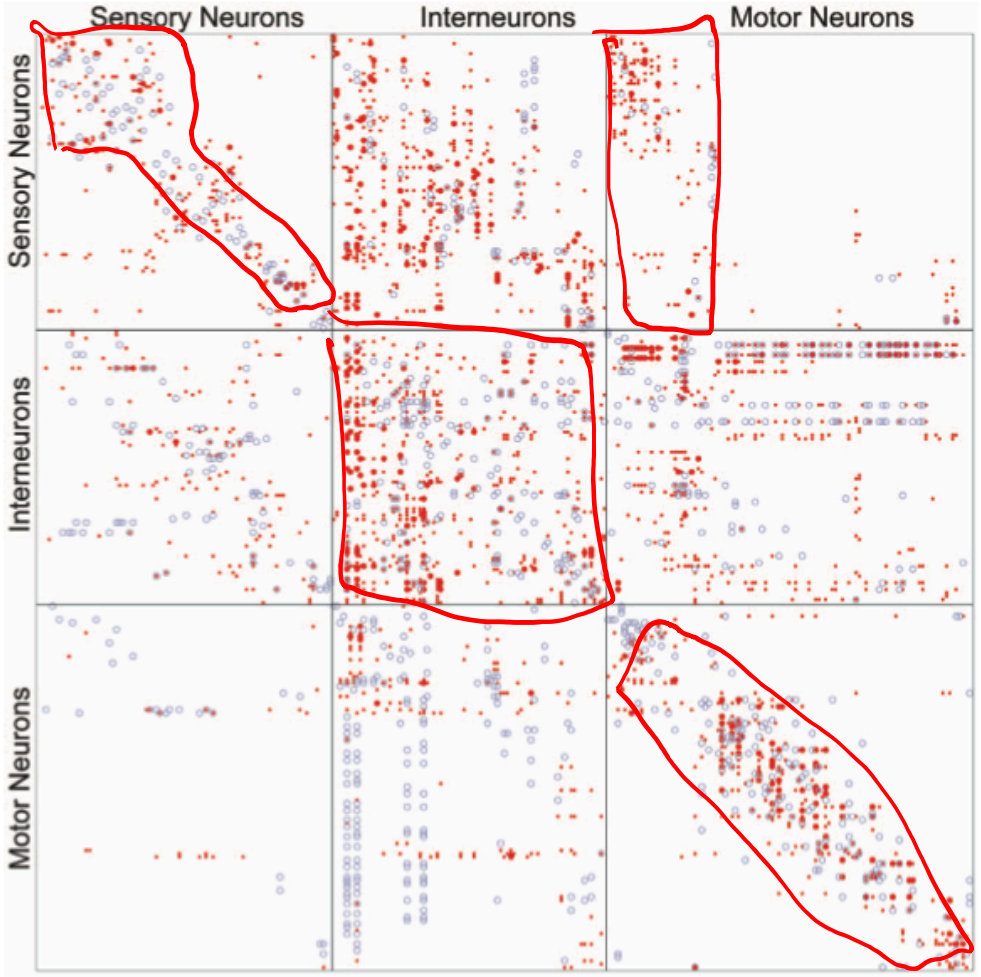
Human Brain

10^{11} neurons

10^{14} connections (**synapses**)



Connectome



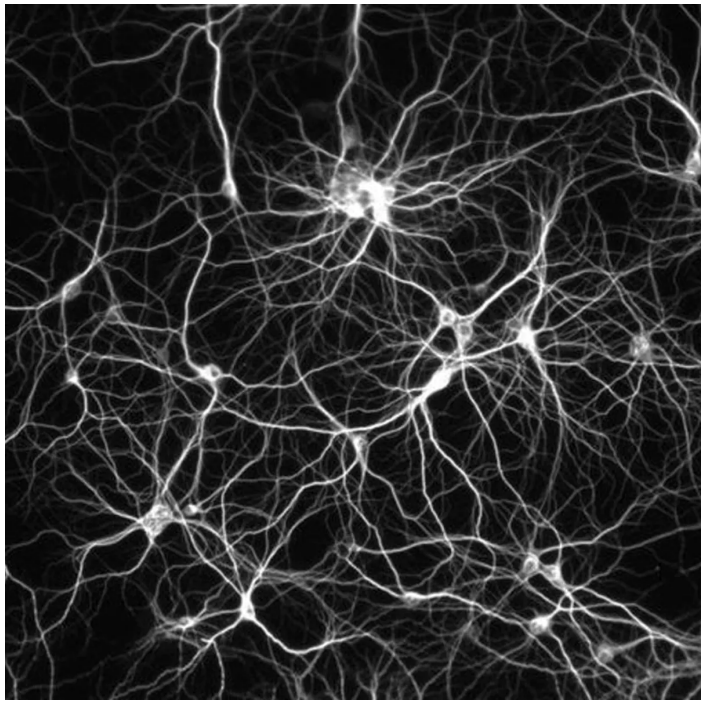
C. elegans

300 neuron

Neuronal Networks

3. Parallelism

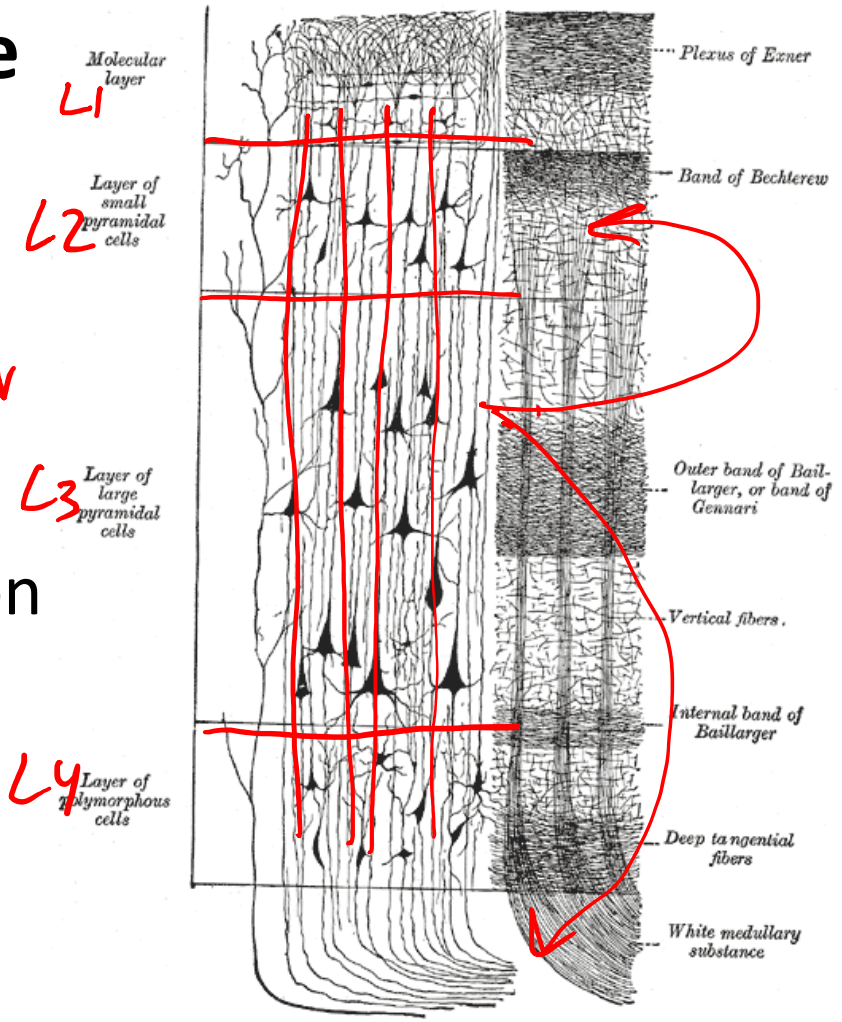
- Computations done on various scales
- Many recurrences



Hierarchical Layered Structure

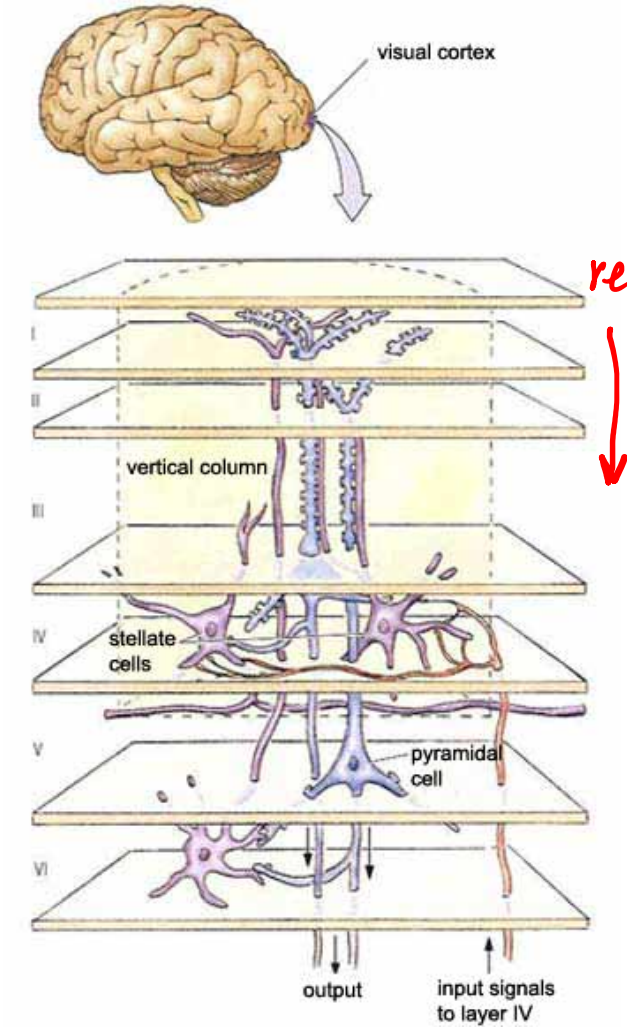
4. Hierarchical Structure on System Level

- Layers
- Columns
- Defined Flow
- Functional Organization
- Feedback

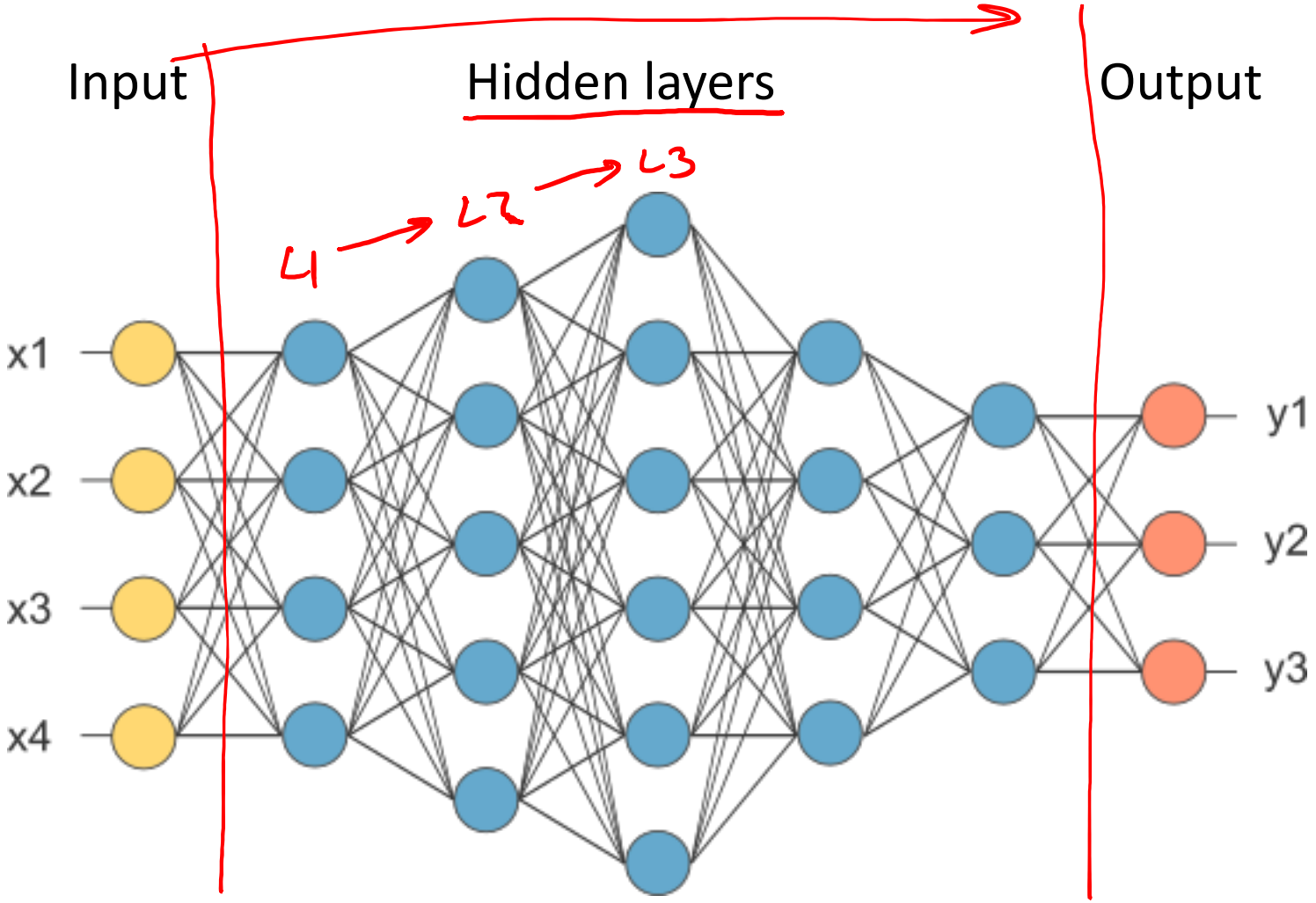


Cerebral cortex

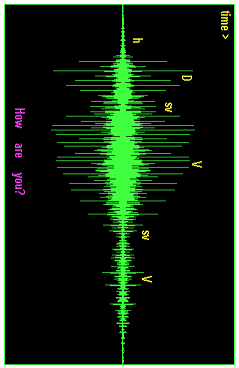
The Visual Cortex

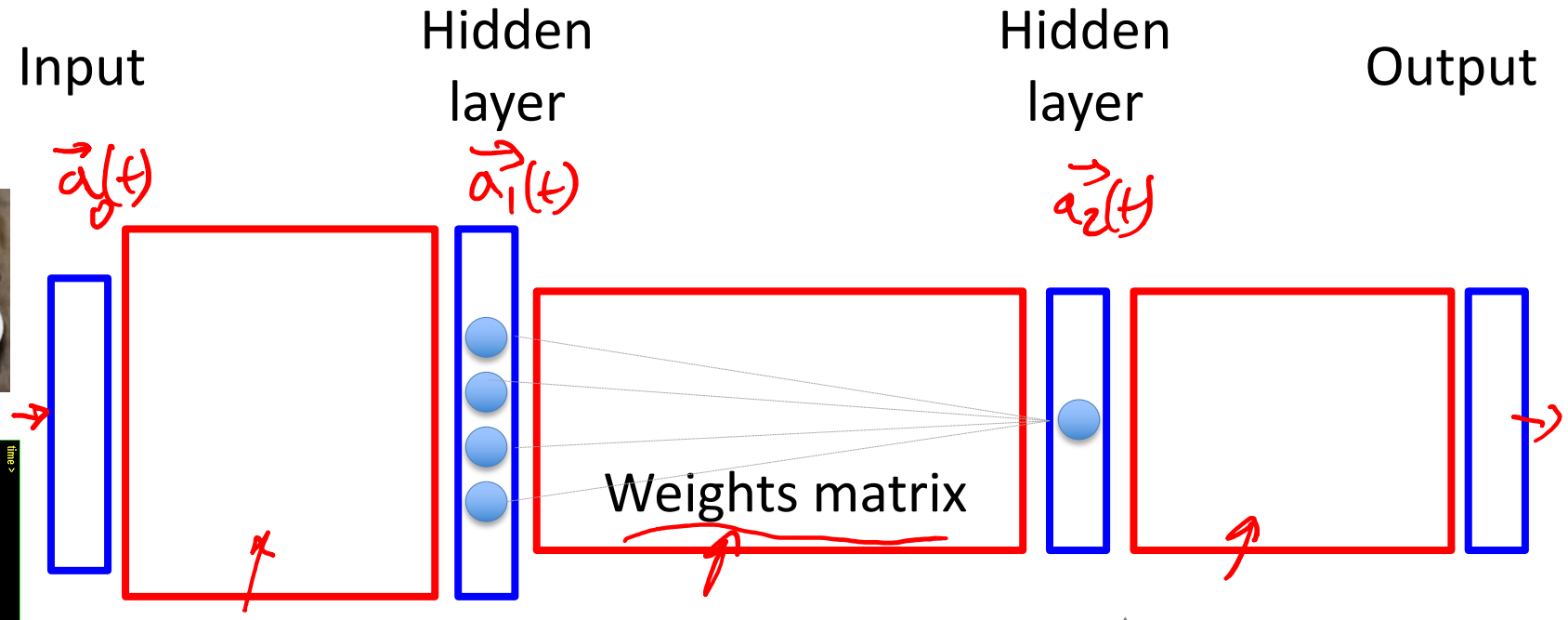


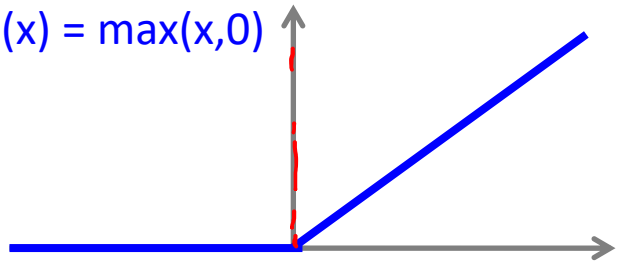
Deep Neural Networks



Deep Neural Networks - Forward Propagation



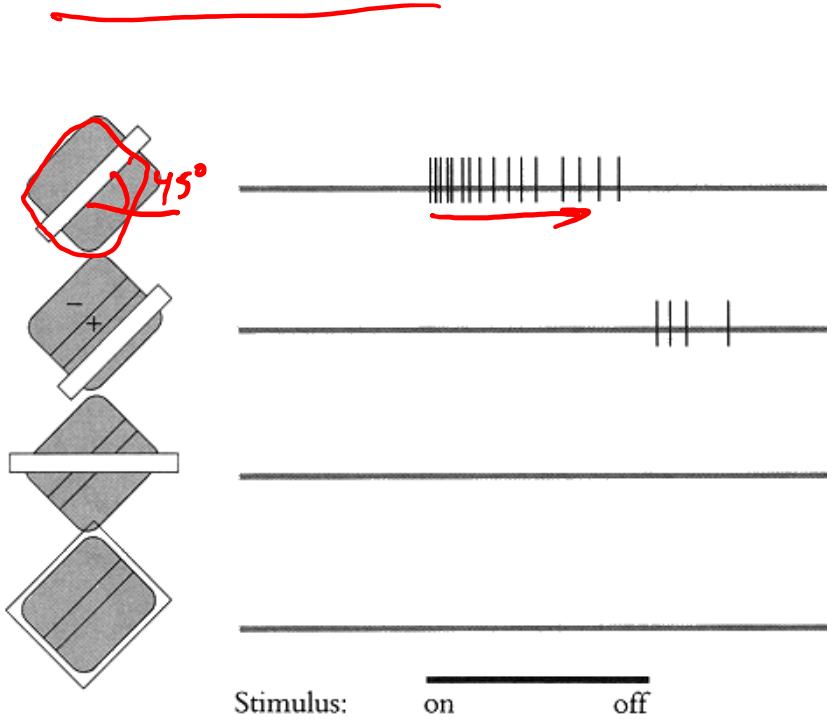




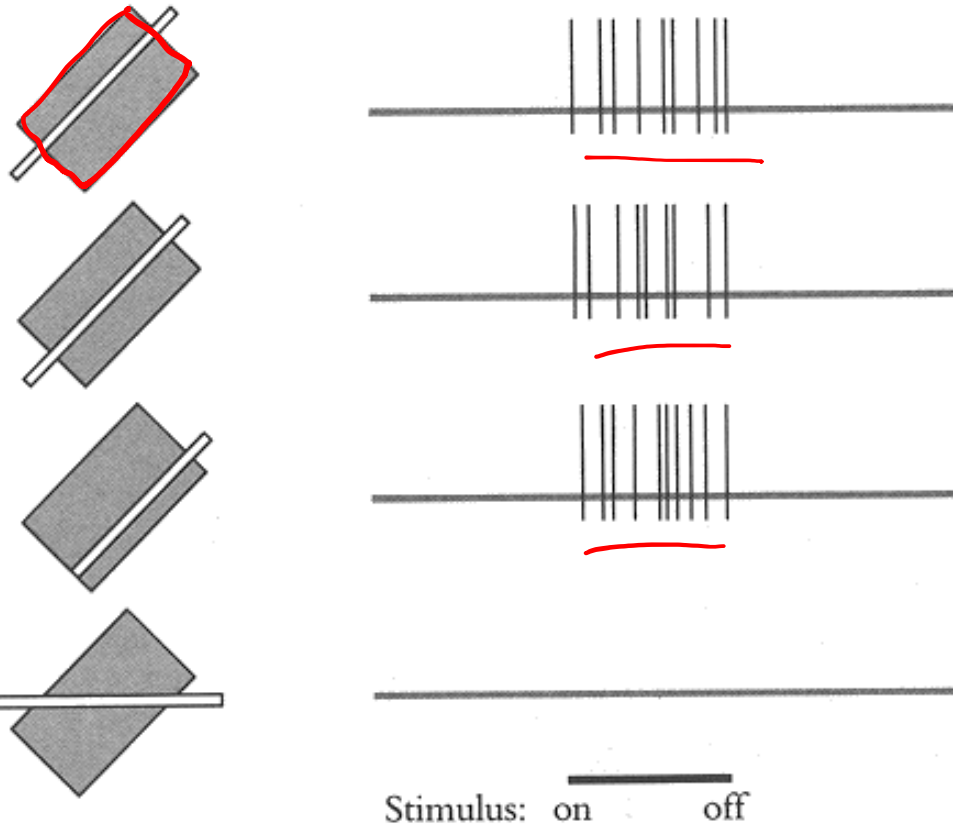
- Multiple layers of **simple units**
- Each unit computes **a weighted sum** of its inputs
- Weighted sum is passed through **a nonlinear** function
- The learning algorithm changes the **weights**

Visual System

Simple Cells

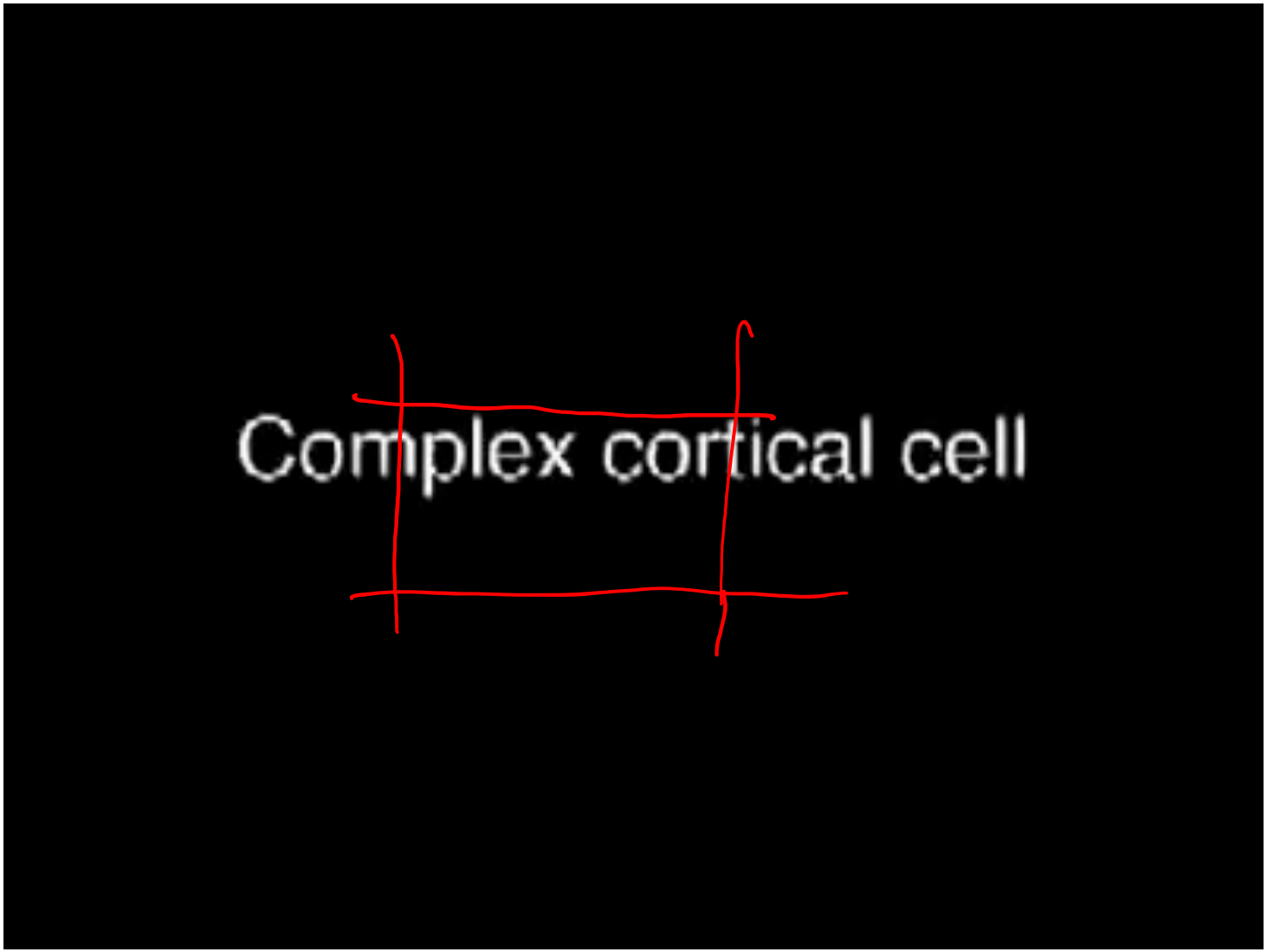


Complex Cells

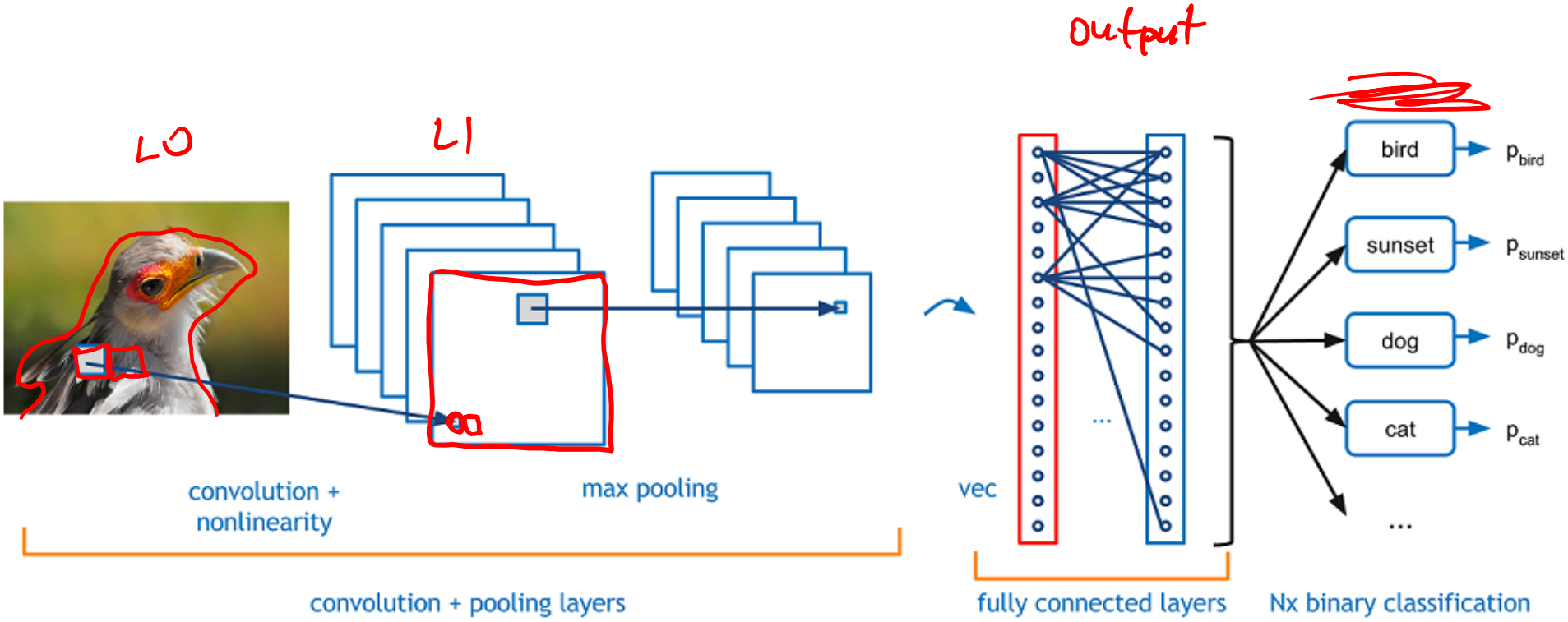


Receptive Fields, Hubel & Weisel , 1959-62 (Nobel Prize)

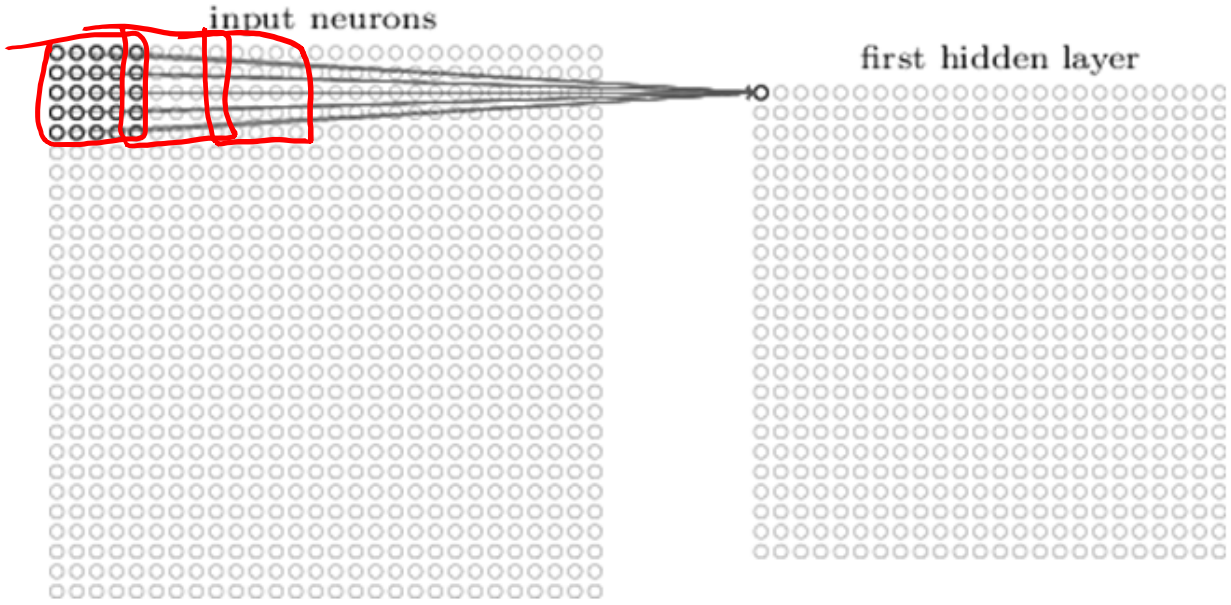
Receptive Fields, Hubel & Weisel (Nobel Prize)


Complex cortical cell

Convolutional Nets (CNNs)



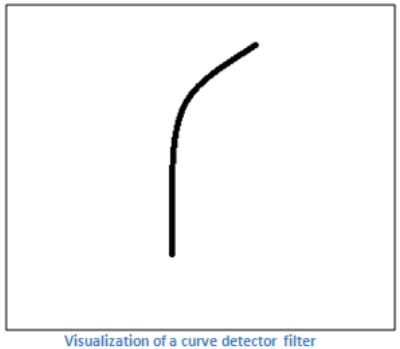
Convolutional Nets



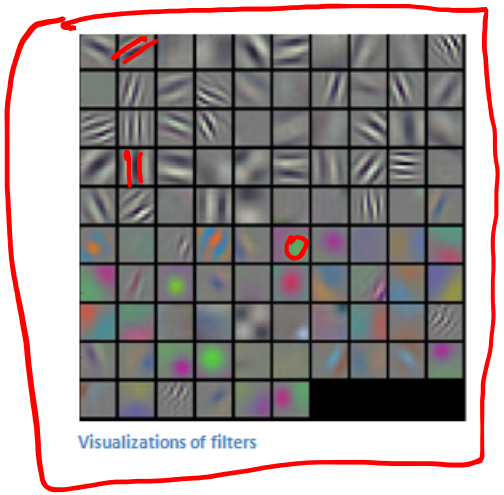
Visualization of 5 x 5 filter convolving around an input volume and producing an activation map

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter



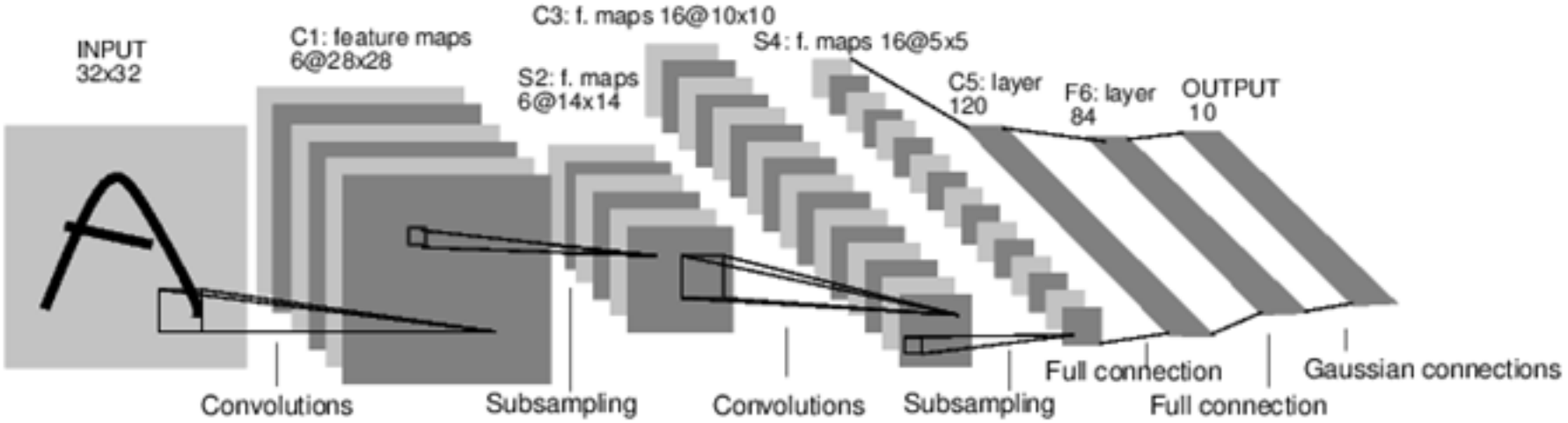
Visualization of a curve detector filter



Visualizations of filters

Convolutional Nets

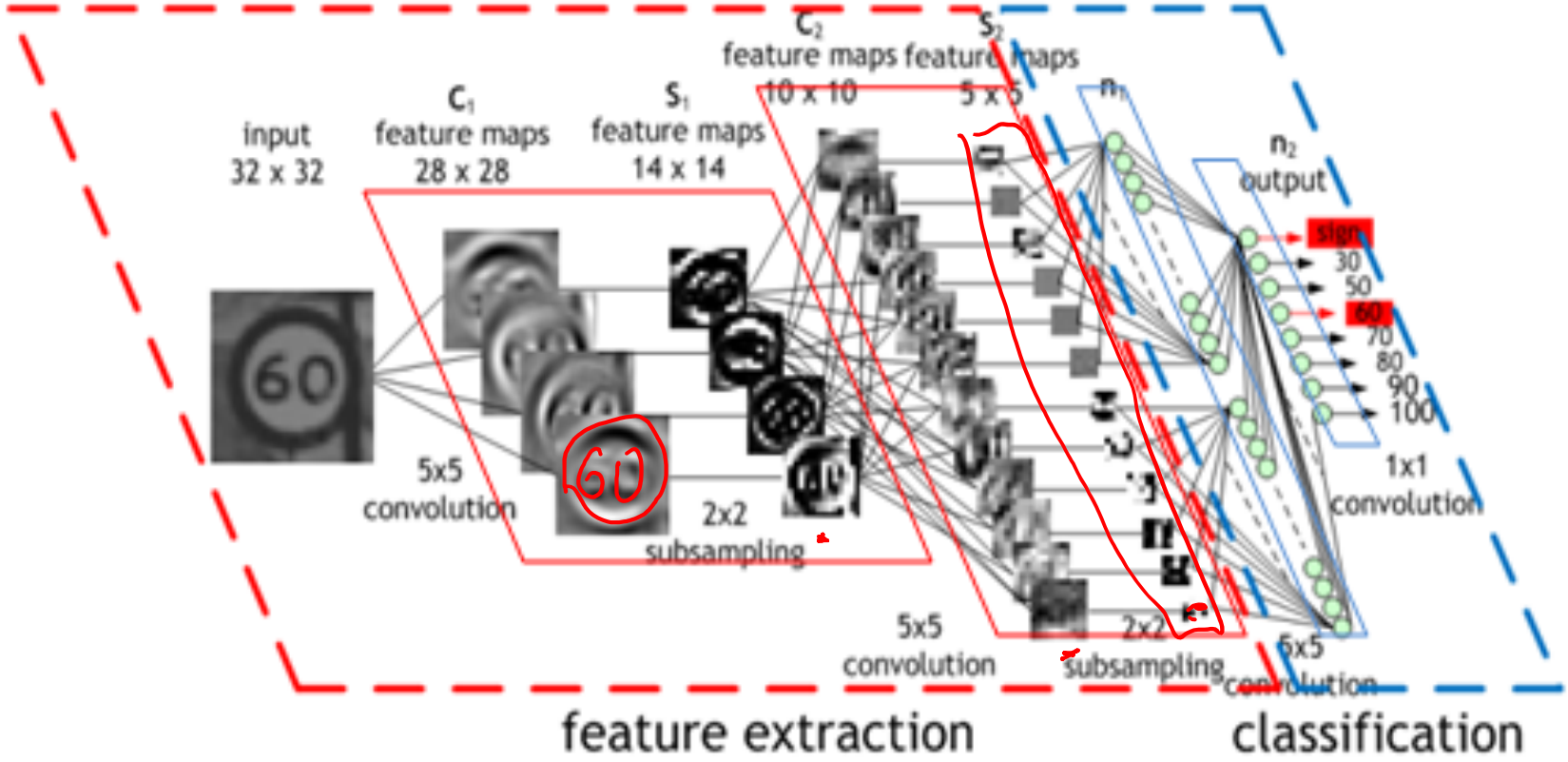
- LeNet 5 (Handwriting recognition)



A Full Convolutional Neural Network (LeNet)

Lecun et al. 1989

Convolutional Nets

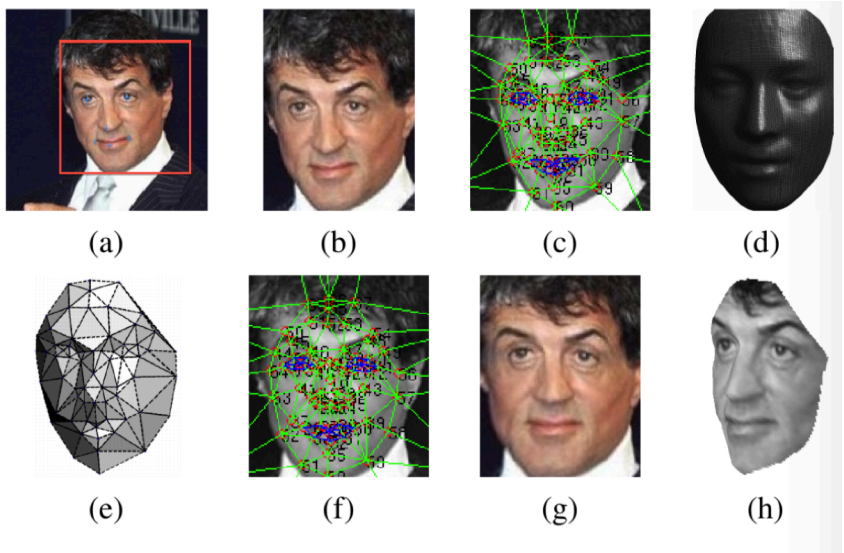


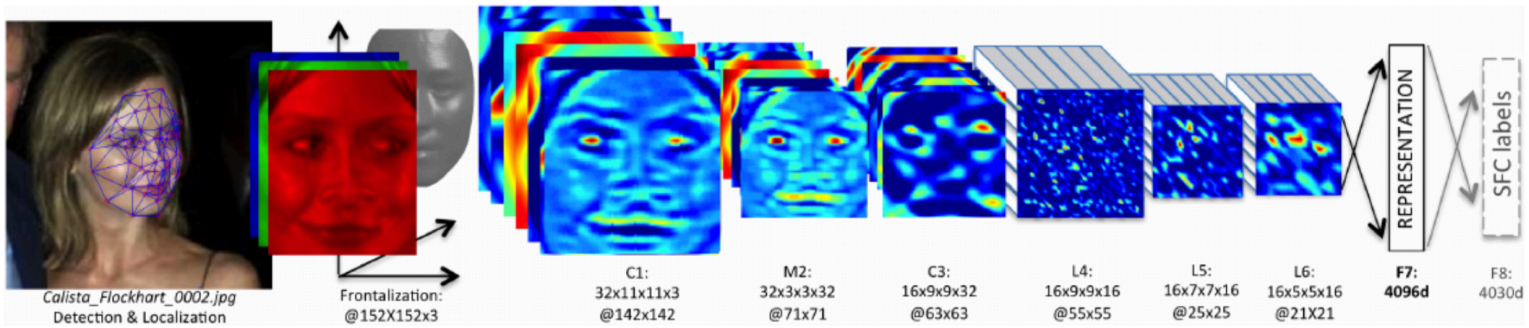
Deep Faces - Face Recognition

- Deployed at Facebook

for each uploaded photo

> 1 Billion photos per day

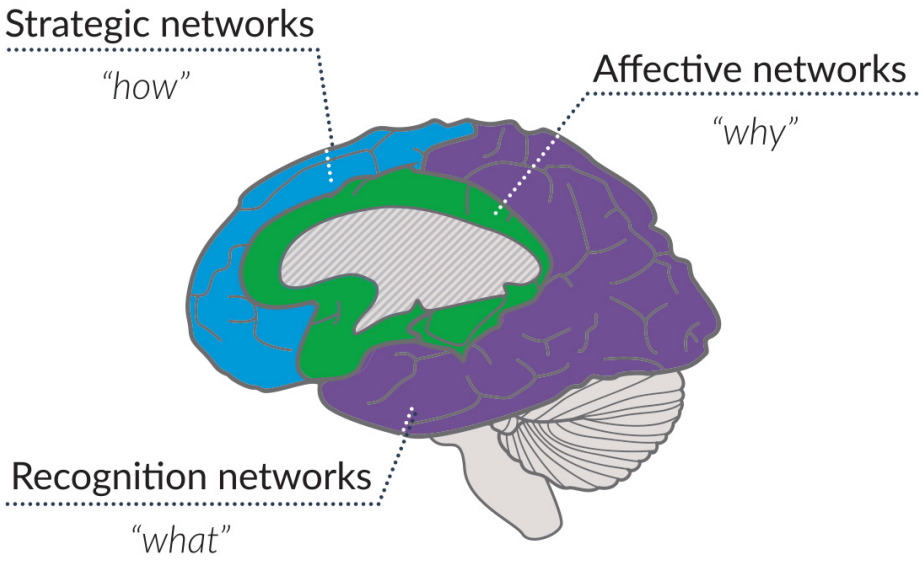




Taigman et al. 2014

Learning

Learning

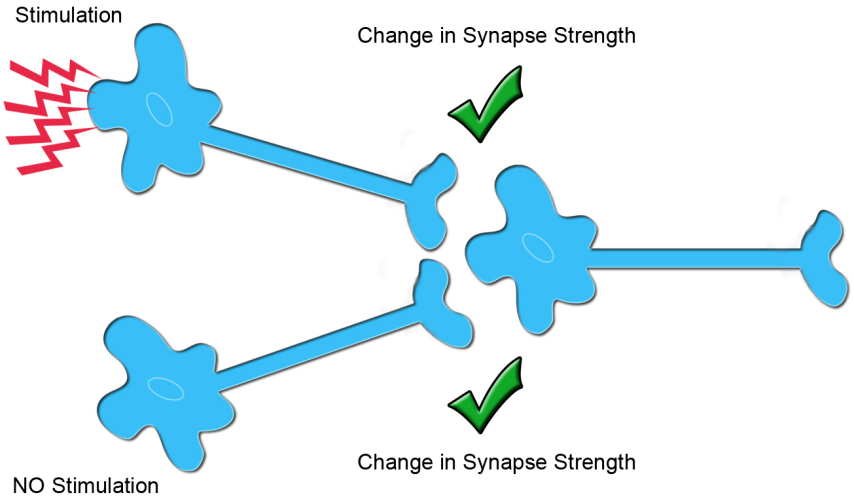


cost function

5.1 Feature Extraction

- Classification/Recognition
- Regression Direct
- Regression Indirect

Learning



5.2 Synaptic Plasticity

- Adaptive connections
- The “strength” of synapses changes

Training (Learning)

- **Supervised Learning**

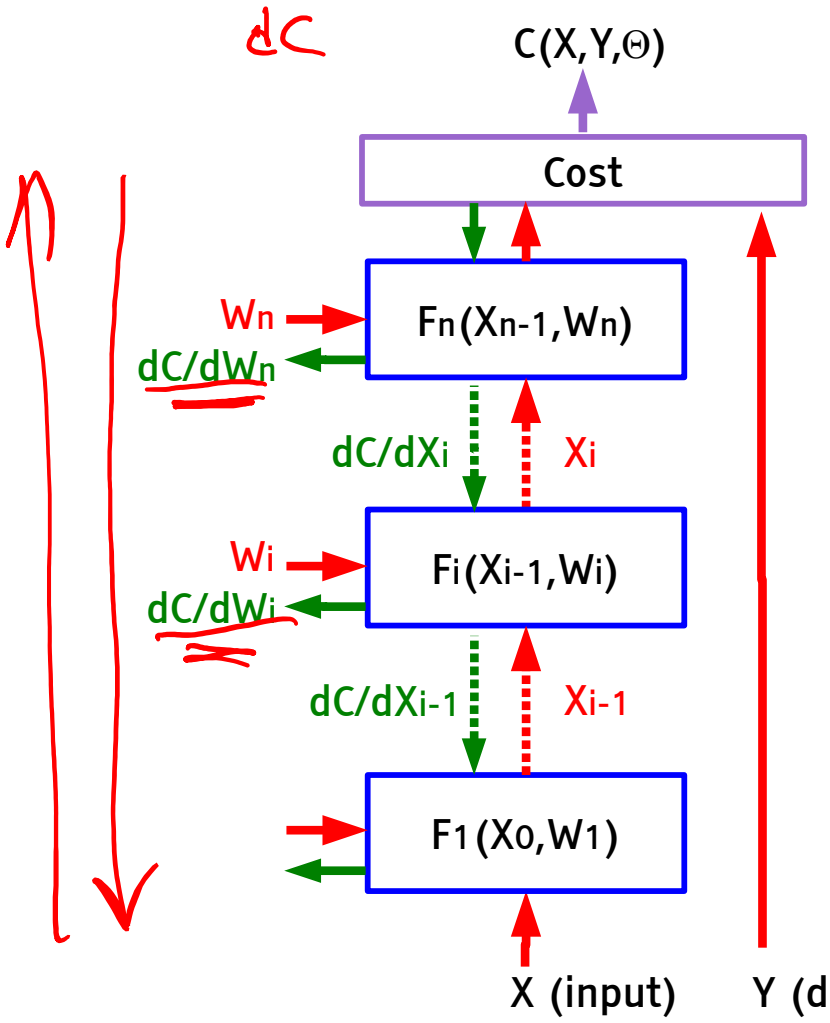
  Network trains its weights based on **labeled training** data, i.e. **optimizes cost** function between expected outputs and actual outputs

$$D = \{\{x_1, d_1\}, \{x_2, d_2\}, \ldots, \{x_m, d_m\}\}$$

input label

$$J = \sum_k (y_k - d_k)^2$$

Back Propagation



$$W_{t+1} = W_t - a\, dC/dW_t$$

- A practical Application of Chain Rule

- Backprop for the state gradients:
- $dC/dX_{i-1} = dC/dX_i \cdot dX_i/dX_{i-1}$
- $dC/dX_{i-1} = dC/dX_i \cdot dF_i(X_{i-1},W_i)/dX_{i-1}$

- Backprop for the weight gradients:
- $dC/dW_i = dC/dX_i \cdot dX_i/dW_i$
- $dC/dW_i = dC/dX_i \cdot dF_i(X_{i-1},W_i)/dW_i$

Training (Learning)

- **Supervised Learning**

  Network trains its weights based on **labeled training** data, i.e. optimizes **cost** function between expected outputs and actual outputs

$$D = \{\{x_1,d_1\}, \{x_2,d_2\}, \ldots , \{x_m,d_m\}\}$$

Training (Learning)

- **Unsupervised Learning**

  Network trains its weights based on **unlabeled training** data, i.e. no clear **cost** function. Clustering often used to create labels (Hebbian Learning). Requires more data.

- **Reinforcement Learning**

  Network trains its weights based on **trials where correct decisions are rewarded**. Requires much more data - many trials. Cannot be done in real time.

Summary

Properties of Brain Networks and Learning

1. Neurons: simple computational units
2. Well connected
3. Parallelism and Recurrence
4. Hierarchically structured
5. Learning
    1. Features
    2. Changing connections

architectures of NN

Lab Preview

DL Platform Setup

- Python Programming Setup
- Python Platforms for DL
- Introduction to Numpy
- Plotting with Matplotlib
- Preparing Data for Machine Learning

Implement your own network (perceptron)

Next

- Machine Learning Basics
- Neural Networks Building Blocks
  - Neurons
  - Integration
  - Activation
  - Layers
  - Training