CSCI 561
Foundation for Artificial Intelligence

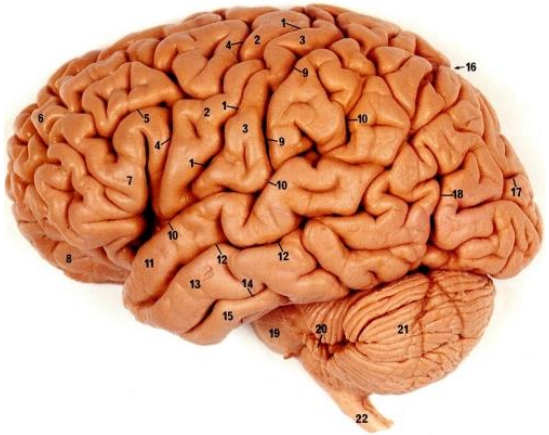
18. Artificial Neural Networks

Professor Wei-Min Shen
University of Southern California

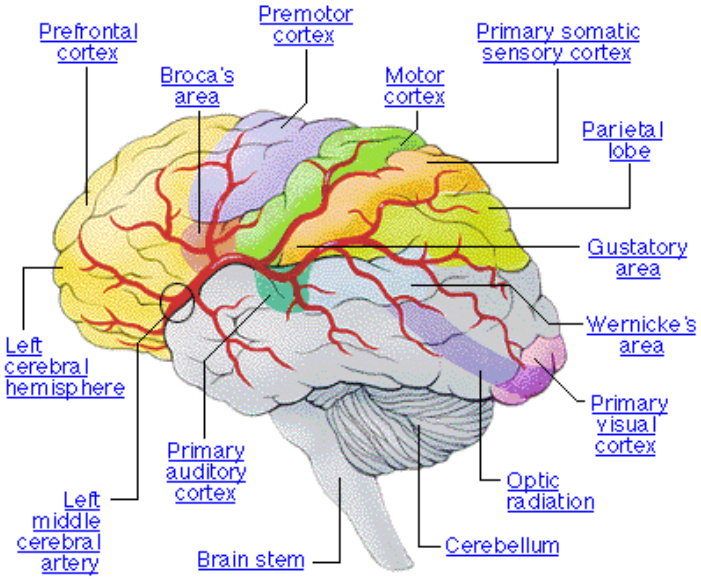
Outline

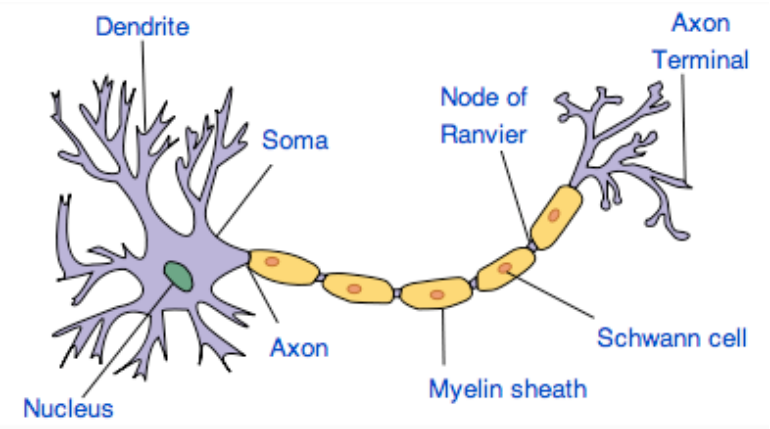
- General Artificial Neural Networks and How They "Learn"
  - Basic Units, Perceptron, Multiple-Layers
  - Back Propagation Algorithm (minimizing the errors)
- Modeling the Brain (very ambitious)
- Deep Learning Neural Networks
  - See more details in
    - This week extra slides
    - Next week's discussion
- Applications

Neural Networks in Brain



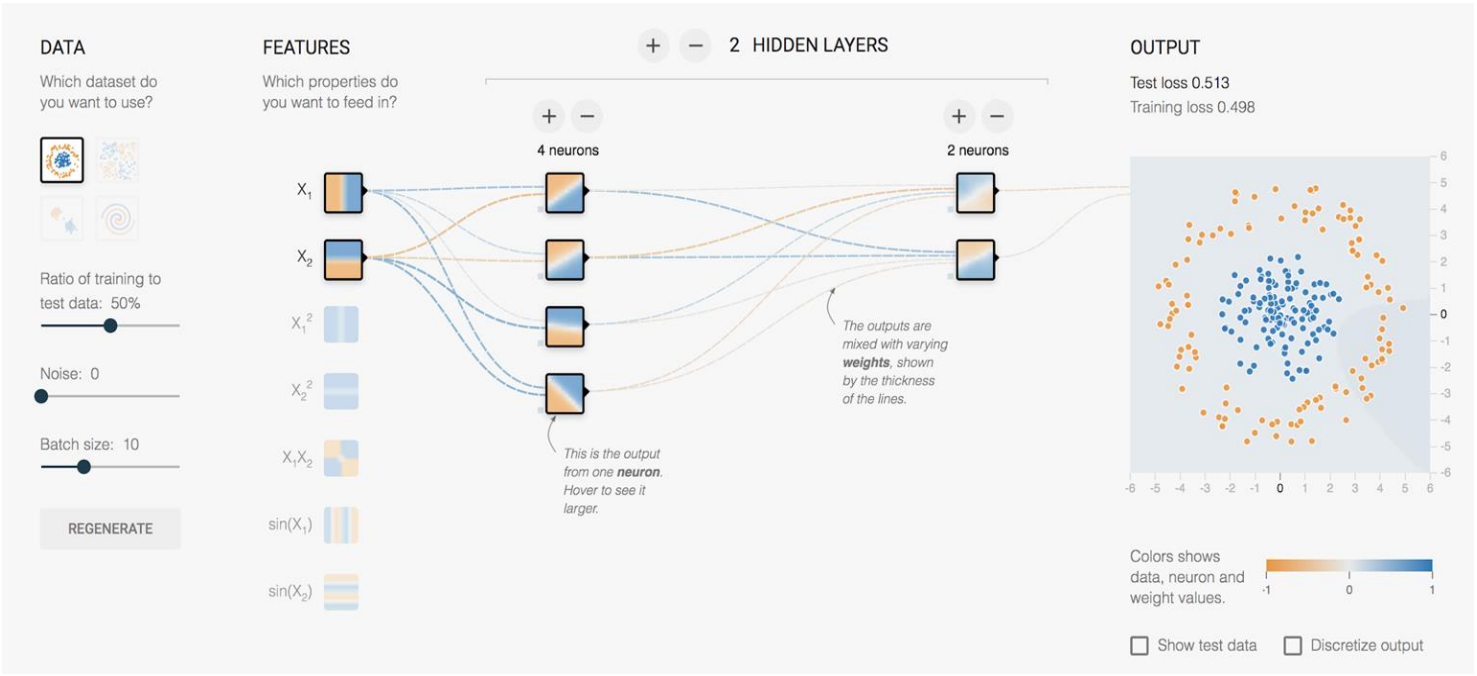
- Human brain is divided into regions with some functional specialization
  - E.g., Wernicke's and Broca's areas for language
- Computation is driven by very large networks of rather slow *neurons* connected via *synapses*
  - 10^{11} neurons of > 20 types
  - 10^{14} synapses
  - 1-10ms cycle time
  - Signals are noisy spike trains of electrical potential
- In AI/ML, models of neurons are highly simplified
- New Discovery: Glial Cells in Brain boost learning

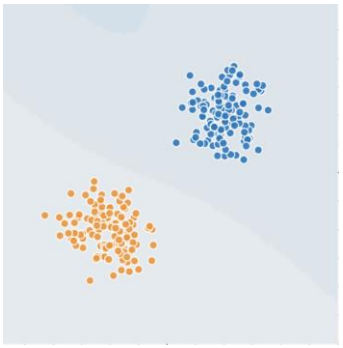


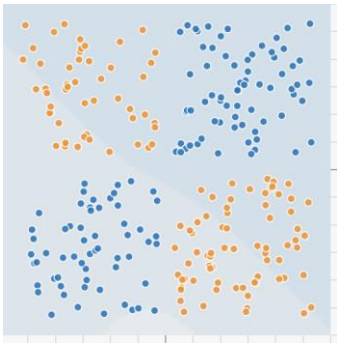


Google's Neural Network Playground

- https://www.playground.tensorflow.org/

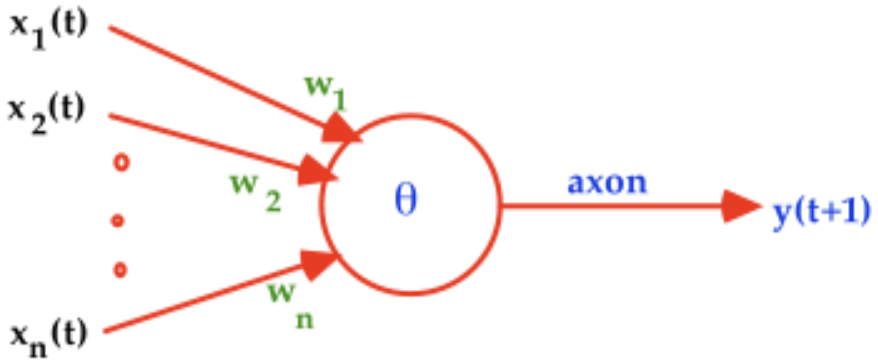






Warren McCulloch and Walter Pitts (1943)

- **A** McCulloch-Pitts neuron operates on a discrete time-scale, t = 0,1,2,3, ... with time tick equal to one refractory period

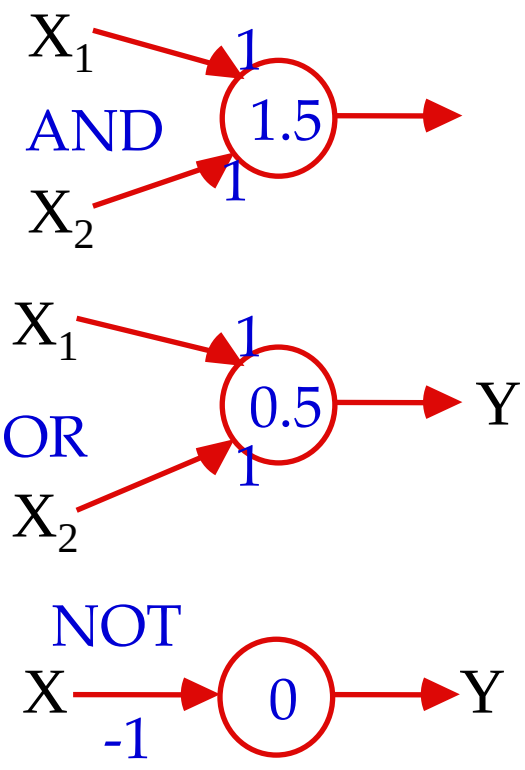


- At each time step, an input or output is

  on or off — 1 or 0, respectively.

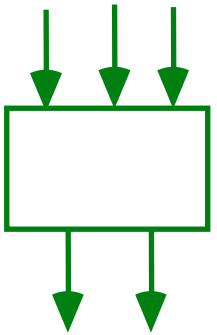
- Each connection or synapse from the output of one neuron to the input of another, has an attached weight.

From Logical Neurons to Finite Automata



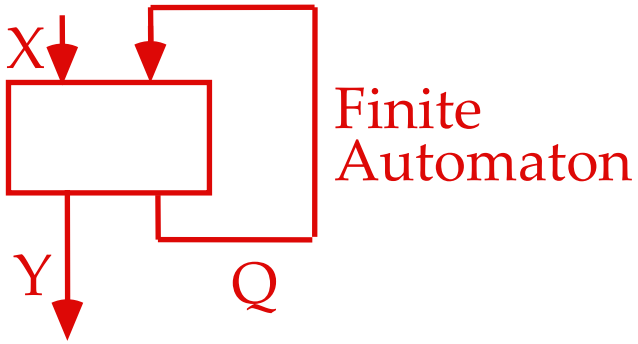
Hopfield Units

Brains, Machines, and Mathematics, 2nd ed, 1987



Boolean Net

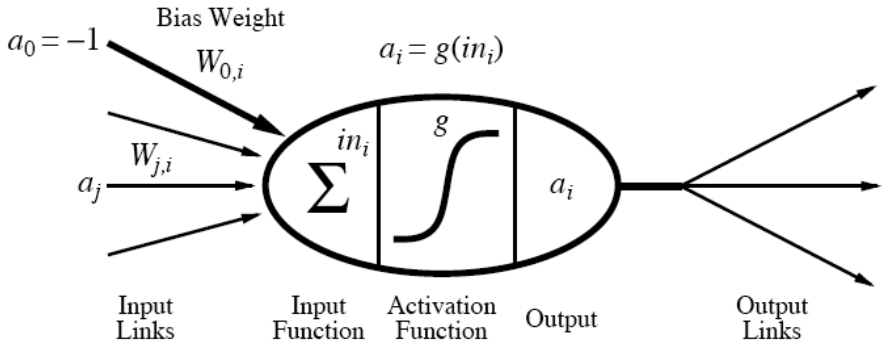
X → Y



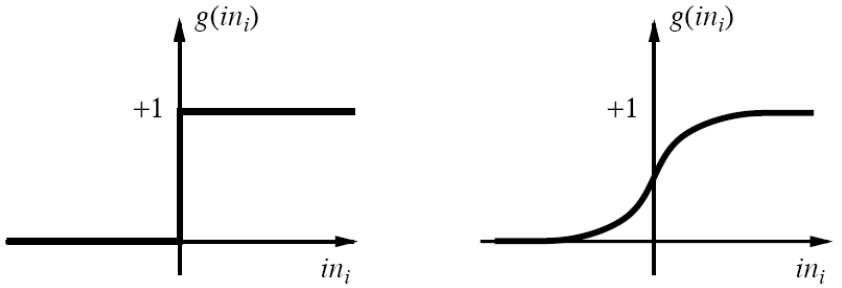
A modern variation of "Units"

- Output is a "squashed" linear function of inputs:

$$a_i \leftarrow g(in_i) = g\left(\sum_j W_{j,i} a_j\right)$$



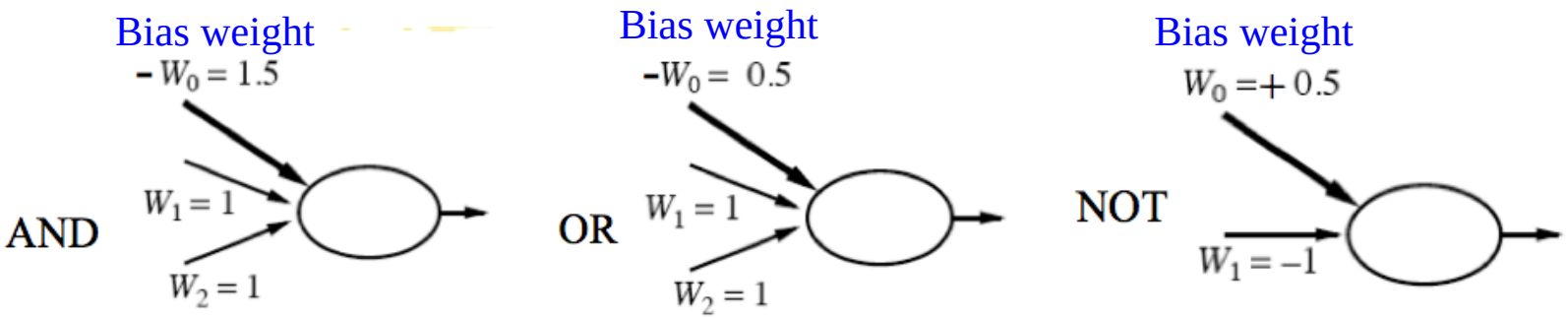
- Activation function can be *step/threshold* or *sigmoid*



Bias weight sets threshold for *linear threshold unit*

Implementing Logical Functions

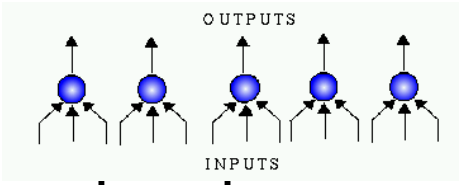
- Any Boolean function can be implemented in a network of linear threshold units (*perceptrons*)

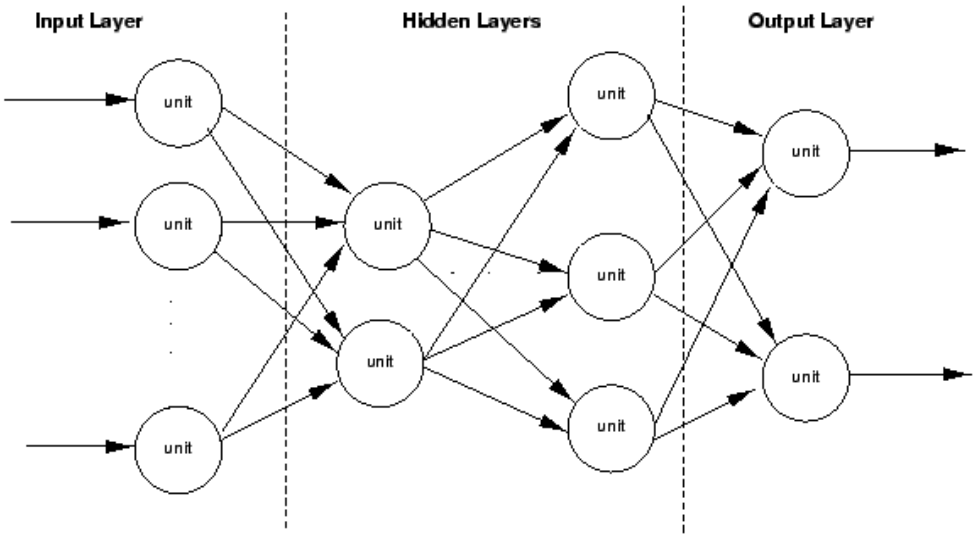


- But *single layer networks* are limited
  - E.g., cannot compute XOR
  - Minsky & Papert essentially killed off neural network field for a decade by showing this in 1969

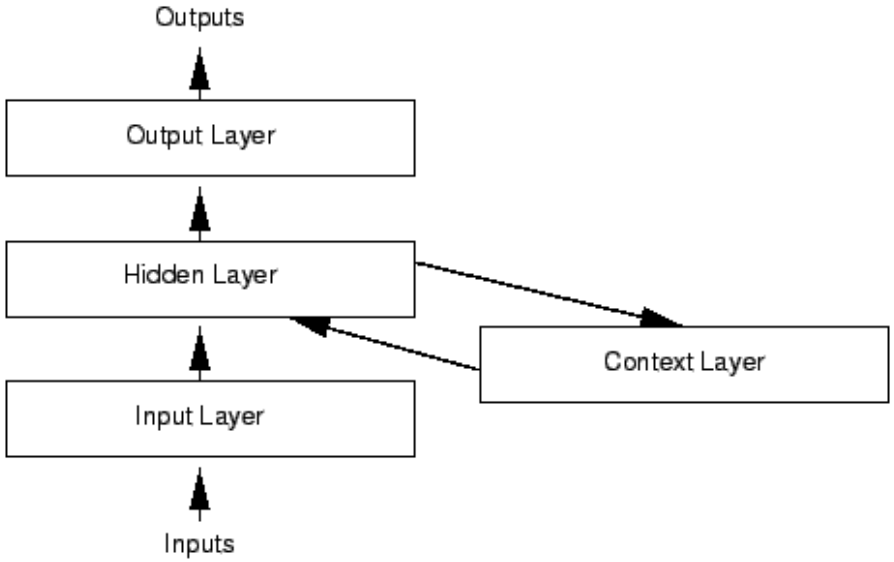
Variety of Network Structures

- Feed-forward networks
  - May be single- or multi-layer
  - Implement functions/reflexes
    - No internal state



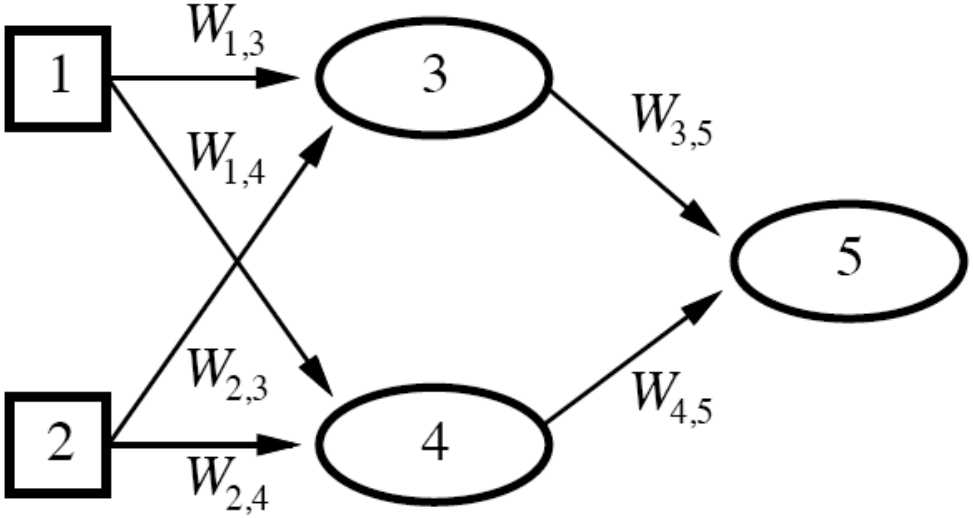


- Recurrent networks
  - Directed cycles with delays
    - Internal state, like flip-flops
    - Dynamical systems
  - Hopfield networks
    - LTUs w/ symmetric weights
    - Minimize overall energy
  - Boltzmann machines
    - Stochastic, simulated annealing



Feed-Forward Example

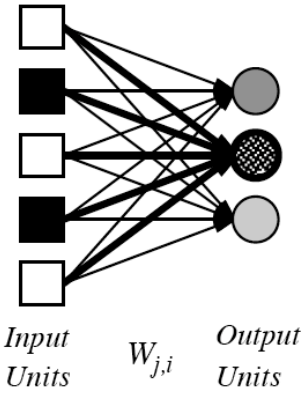
- Feed-forward networks provide a parameterized family of nonlinear functions

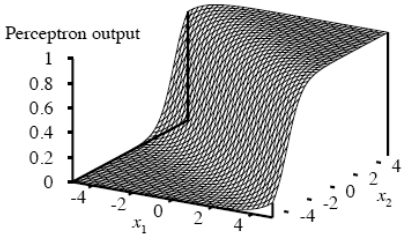


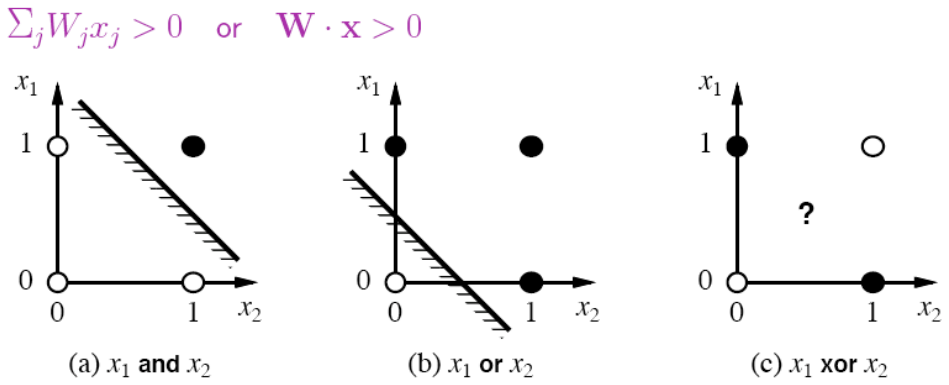
- $a_5 = g(W_{3,5}a_3 + W_{4,5}a_4)$
  $= g(W_{3,5}g(W_{1,3}a_1+W_{2,3}a_2) + W_{4,5}g(W_{1,4}a_1+W_{2,4}a_2))$
- *Learning occurs by adjusting weights*

Single-Layer Perceptrons

- Each output unit operates separately
  - Implements a *linear separator*

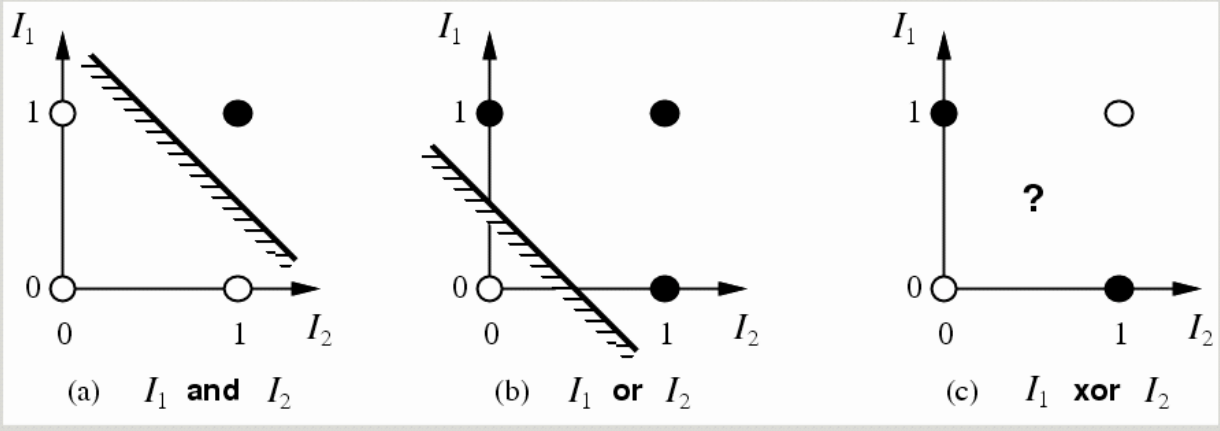






- Adjusting weights alters *location*, *orientation*, and *steepness* of cliff

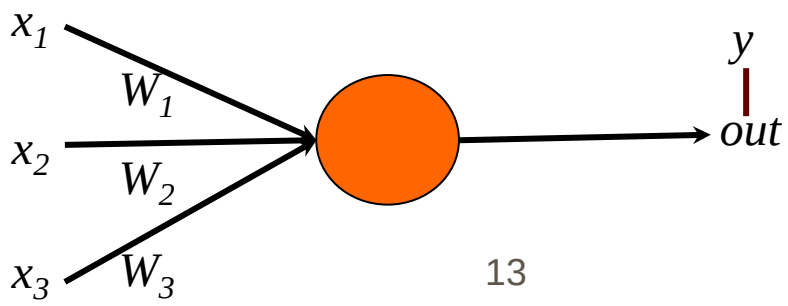
limitations of a single layer perceptron and linear separability problem



- a single layer perceptron can only represent **linearly separable functions** which follows from the inequalities that determine its firing (Minsky & Papert(1969))
- **y(t+1) = 1 if and only if** $\Sigma w_i x_i(t) \geq \theta$ else **y(t+1) = 0**
- i.e. input space is divided in 2

Perceptron Learning

- Learn by adjusting weights on each iteration to reduce error
  - Gradient descent on squared error $(y - out)^2$
  - Update rule for weights of a threshold function:
    $W_j' = W_j + \alpha (y - out) x_j$
  - Comparable rules exist for differentiable activation functions
- This *perceptron learning rule* is guaranteed to converge to a consistent function if the data is linearly separable



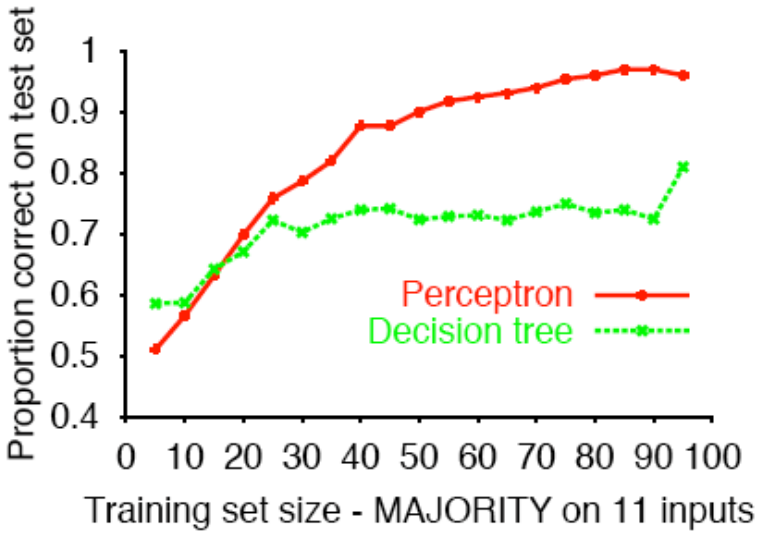
Majority Function

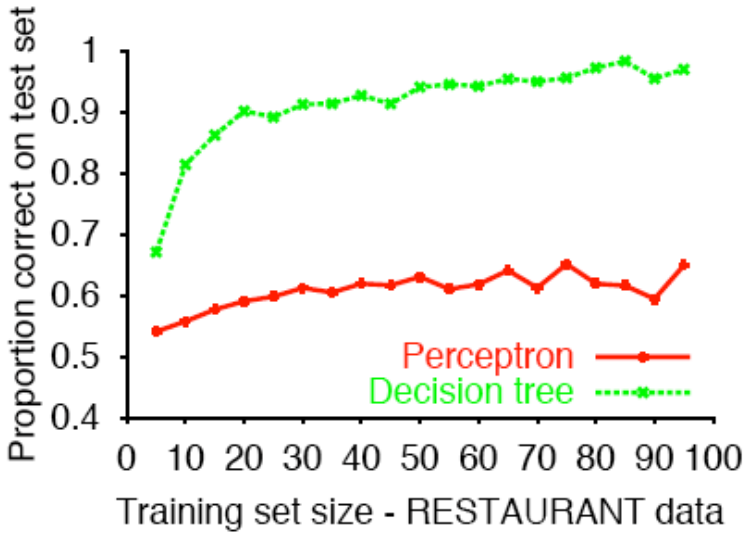
Output 1 if and only if > ½ of n binary variables are 1

- Representable within a perceptron
  - $w_i = 1$
  - $w_0 = -(n/2)$
  - $o = 1$ if and only if $a_1+a_2+...+a_n - (n/2) \geq 0$
- Compare to a decision tree needs an exponential number of branches
  - One branch for each combination with a majority of Trues
  - e.g., **IF** a_1=1 **AND** a_2=1 **AND**… $a_{n/2}$=1 **AND** $a_{n/2+1}$=1 **THEN** 1
  - **ELSE IF** a_1=0 **AND** a_2=1 **AND**… $a_{n/2+1}$=1 **AND** $a_{n/2+2}$=1 **THEN** 1

Sample Results

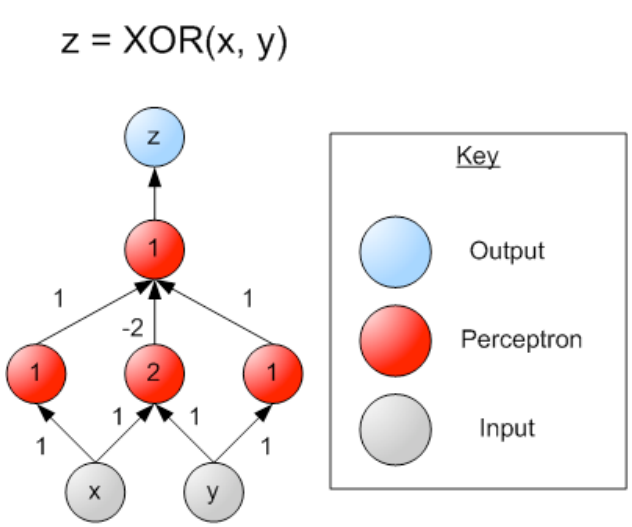
- Perceptron learns majority function easily; DTL hopeless
  - Linearly separable, but difficult to express with yes/no splits on attributes without a path for each combination of attributes that would yield a majority $\binom{11}{6}$
- DTL learns restaurant function easily; single layer perceptron can't represent it
  - Data came from a decision tree
  - Data is not linearly separable

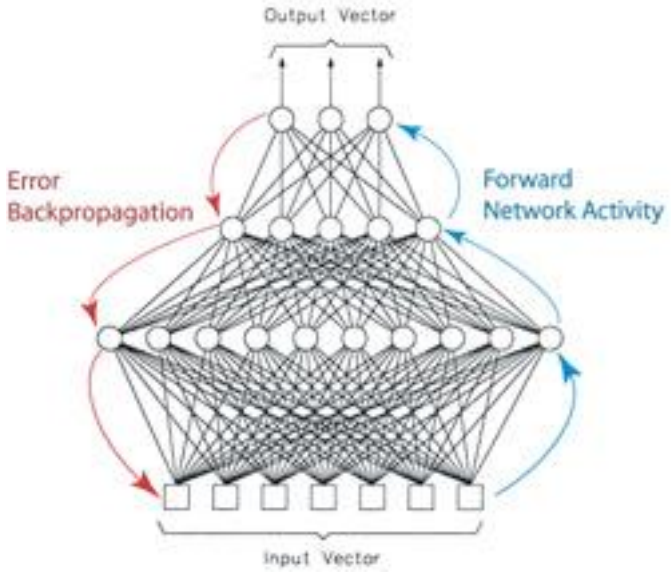




Multi-layer Perceptrons

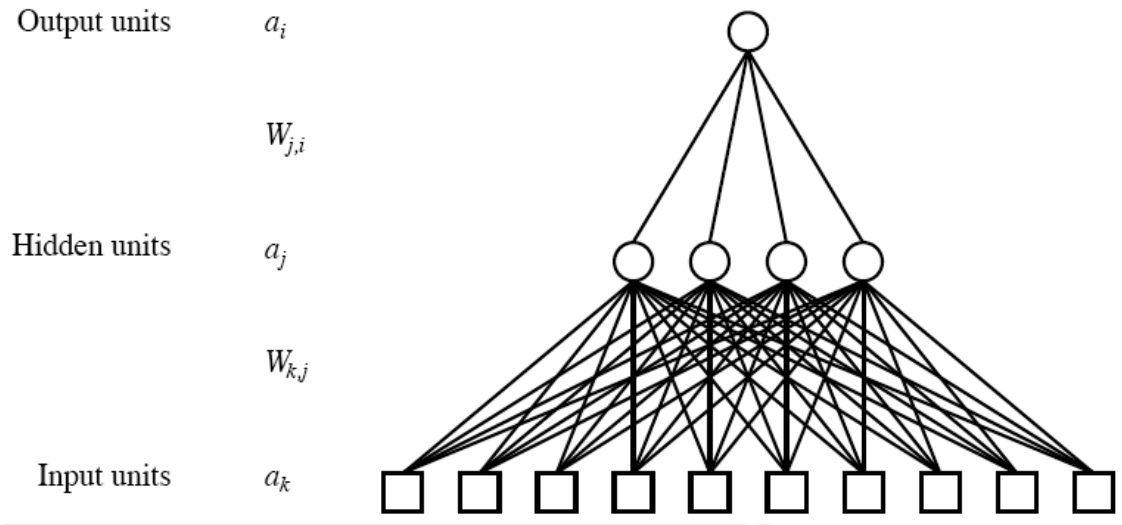
- single layer perceptrons can only represent linear decision surfaces
- multi-layer perceptrons can represent non-linear decision surfaces.





Multilayer Neural Networks

- Units typically fully connected across layers unless have domain-specific knowledge to guide
- Number of hidden units either chosen by hand or found via search
  - If too many, network will memorize input (overfitting)
  - If too few, function will be difficult to represent
  - *As with Goldilocks, need it just right (to generalize)*



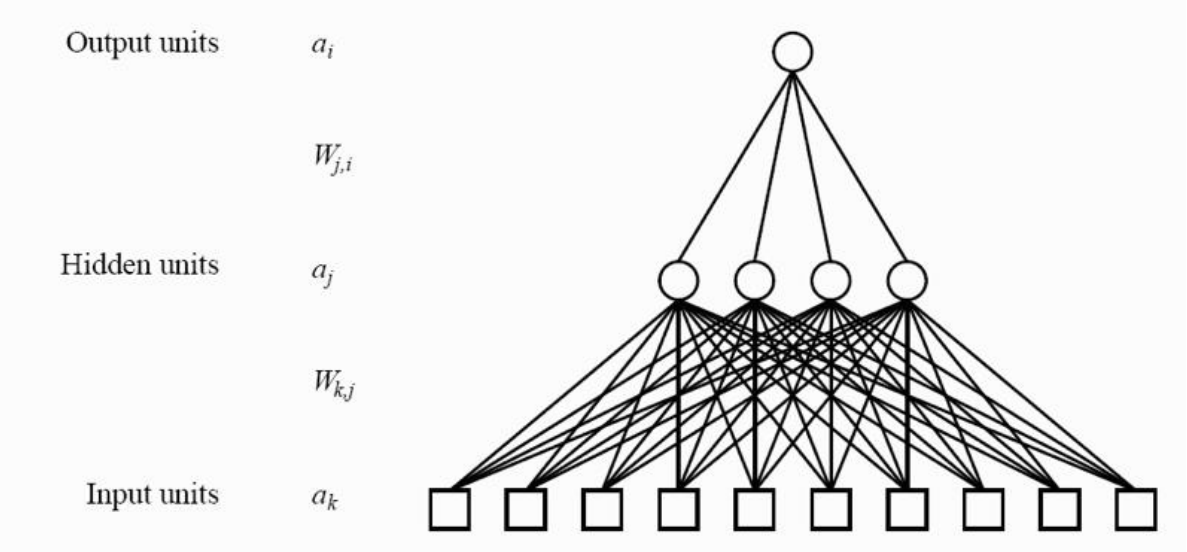
Back-Propagation (learning)

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. ***Nature***, 323(9) 533-536 1986.
www.cs.toronto.edu/hinton/absps/naturebp.pdf.

Learn by Back-Propagation

- Challenges for learning in networks with hidden units
  - Can't directly compute error on output of hidden units because don't have the true values for them
  - Also a clustering problem/opportunity
    - May yield new useful features
- General approach is to (re)apportion error on output units to predecessors based on the weight of their connection, and continue this recursively backwards through the network

Training a multilayer network using Backpropagation



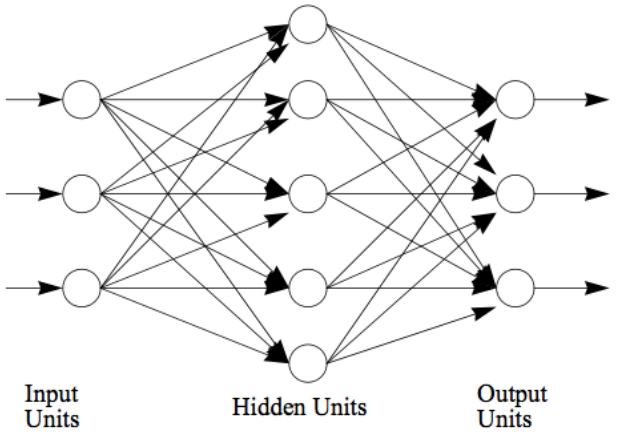
$$W_{ji} = W_{ji} + \alpha * a_j * \delta_i$$

$$\delta_i = (T_i - O_i) * g'(in_i)$$

$$W_{kj} = W_{kj} + \alpha * a_k * \delta_j$$

$$\delta_j = g'(in_j) \sum W_{ji}\delta_i$$

Neural Networks
(The back-propagation math)
(WS Chapter 4)



The output o_j of a unit j in a network depends on the unit's mapping function f_j, its inputs $o_{i_1}, o_{i_2}, \ldots, o_{i_m}$ (these are outputs of units that feed into unit j), and the weights associated with each input $w_{i_1 j}, w_{i_2 j}, \ldots, w_{i_m j}$. If we define the net input to the unit j to be

$$I_j = \sum_i o_i w_{ij}$$

then the output of unit j is given as

$$o_j = f_j(I_j)$$

The objective of a learning algorithm, such as back-propagation, is to change the weights in a network to reduce the difference between the the actual output of the network and the desired output provided by the training examples.

Back-Propagation (1/3)

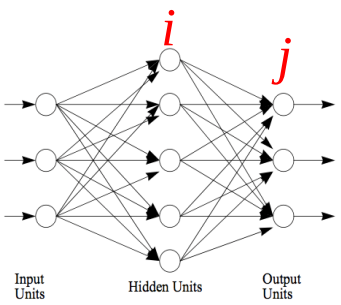
Let t_j be the desired output of unit j and o_j the actual output of unit j, we define the difference to be

$$E = \frac{1}{2}(t_j - o_j)^2$$

Using the gradient descent method, how much a weight w_{ij} needs to be changed can be computed as

$$w_{ij} = w_{ij} - \eta \nabla E \tag{4.2}$$

where η is a step size. The factor ∇E can be computed as follows:



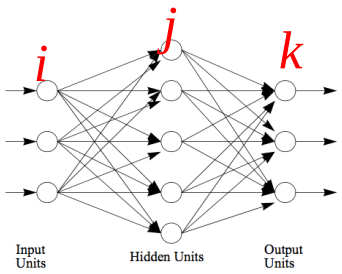
$$\begin{aligned}
\nabla E &= \frac{\partial E}{\partial w_{ij}} = \left(\frac{\partial E}{\partial o_j}\right)\left(\frac{\partial o_j}{\partial w_{ij}}\right) \\
&= \left(\frac{\partial E}{\partial o_j}\right)\left(\frac{\partial o_j}{\partial I_j}\right)\left(\frac{\partial I_j}{\partial w_{ij}}\right) \\
&= \frac{\partial E}{\partial o_j} f_j'(I_j) o_i
\end{aligned}$$

When the unit j is an output unit, then according to the definition of E, we have $\frac{\partial E}{\partial o_j} = -(t_j - o_j)$, thus

$$\nabla E = -f'(I_j) o_i (t_j - o_j) \tag{4.3}$$

Back-Propagation (2/3)

When the unit j is not an output unit, then the error of o_j depends on the units k that j outputs to. Thus, we can rewrite $\frac{\partial E}{\partial o_j}$ as



$$\frac{\partial E}{\partial o_j} = \sum_k \left(\frac{\partial E}{\partial I_k}\right)\left(\frac{\partial I_k}{\partial o_j}\right) = \sum_k \frac{\partial E}{\partial I_k} w_{jk}$$

where unit k is the unit from j through w_{jk} and I_k is net input of the unit k. Thus, we have

$$\nabla E = f'(I_j) o_i \sum_k \frac{\partial E}{\partial I_k} w_{jk} \qquad (4.4)$$

Using Equations 4.2,4.3,and 4.4 we can update every weight in the network. These three equations represent the method of back-propagation, where

$$\frac{\partial E}{\partial I_k} w_{jk} = \frac{\partial E}{\partial o_k}\frac{\partial o_k}{\partial I_k} w_{jk} = \frac{\partial E}{\partial o_k} f'(I_k) w_{jk}$$

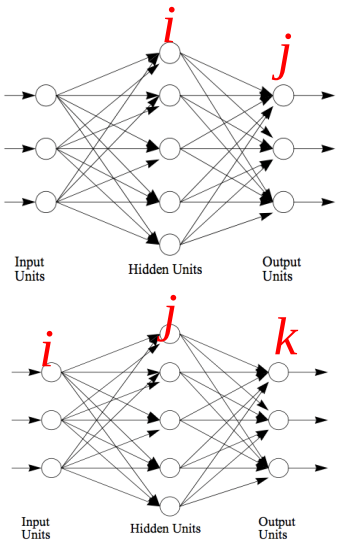
Back-Propagation (3/3)

Summery: change a link weight from unit *i* to *j* using these formulas:

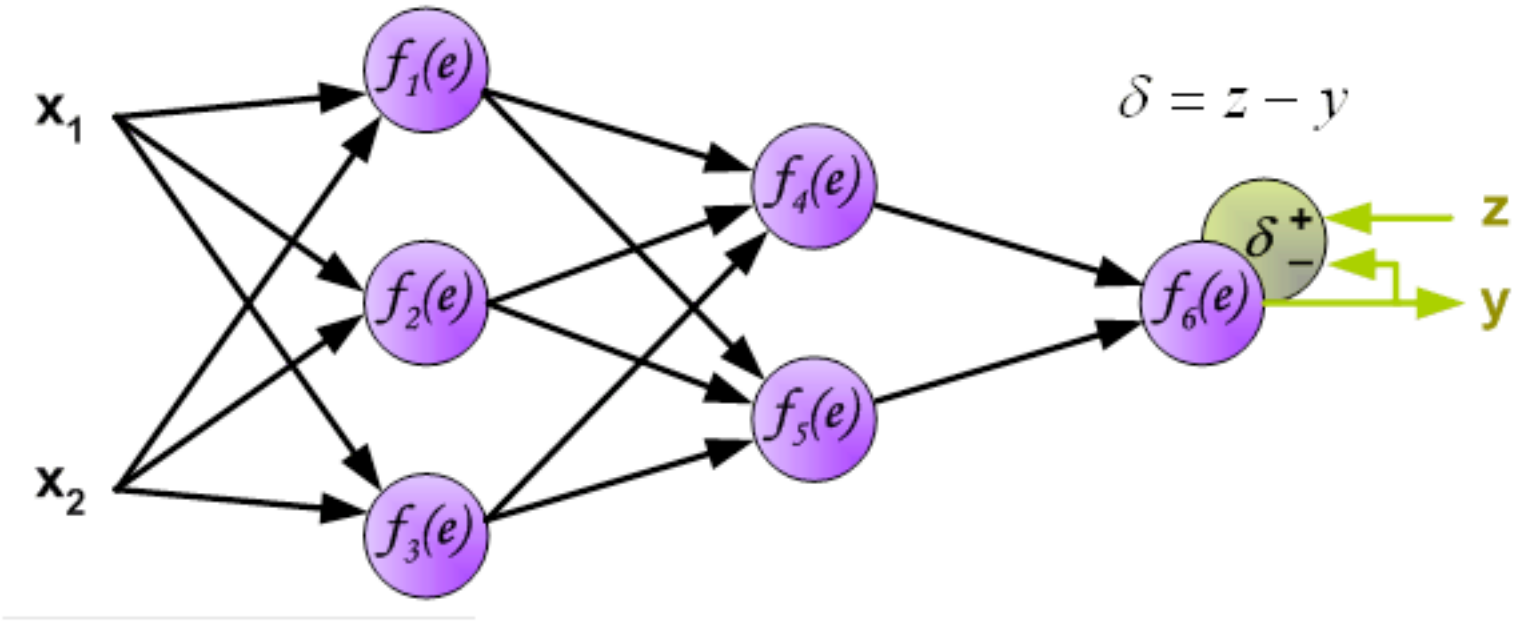
$$w_{ij} = w_{ij} - \eta \nabla E$$

$$\nabla E = o_i f'(I_j) \frac{\partial E}{\partial o_j}$$

$$\frac{\partial E}{\partial o_j} = \begin{cases} -(t_j - o_j) & \textit{When } j \textit{ is an output unit} \\ \sum_k w_{jk} f'(I_k) \frac{\partial E}{\partial o_k} & \textit{When } j \textit{ is not an output unit} \end{cases}$$

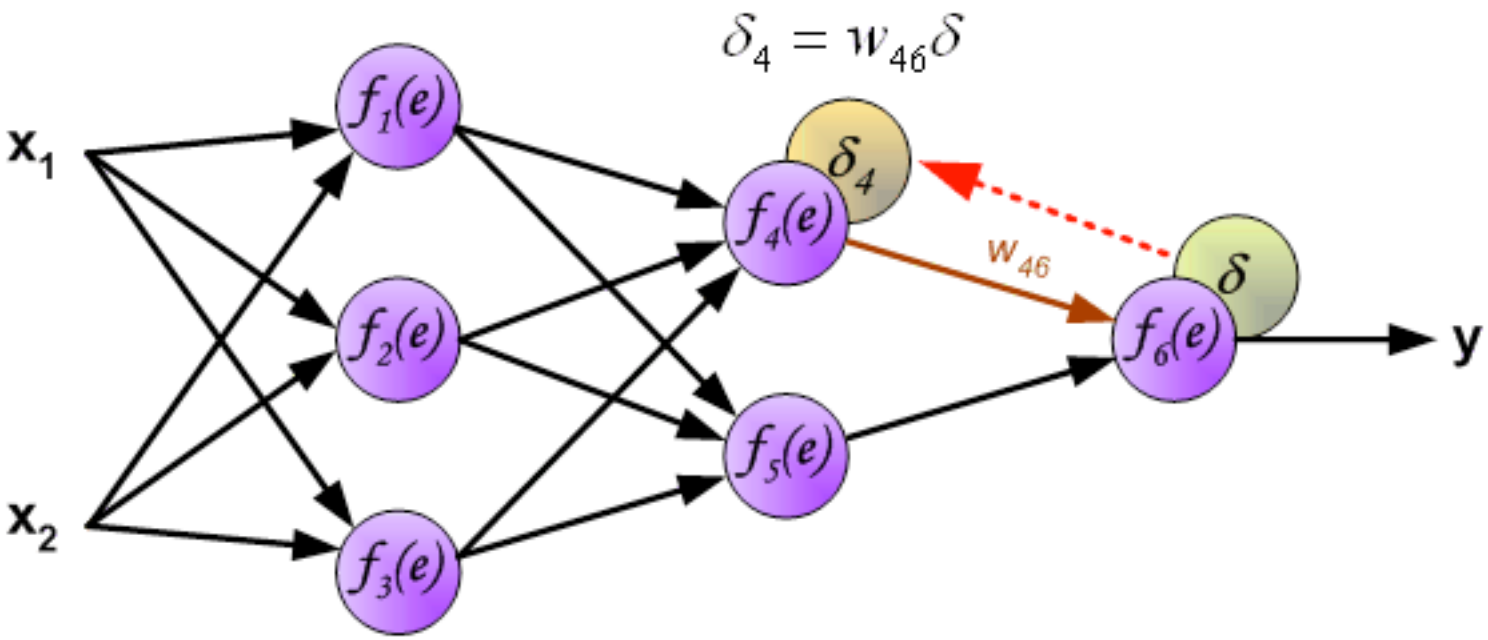


Example of Back-propagation

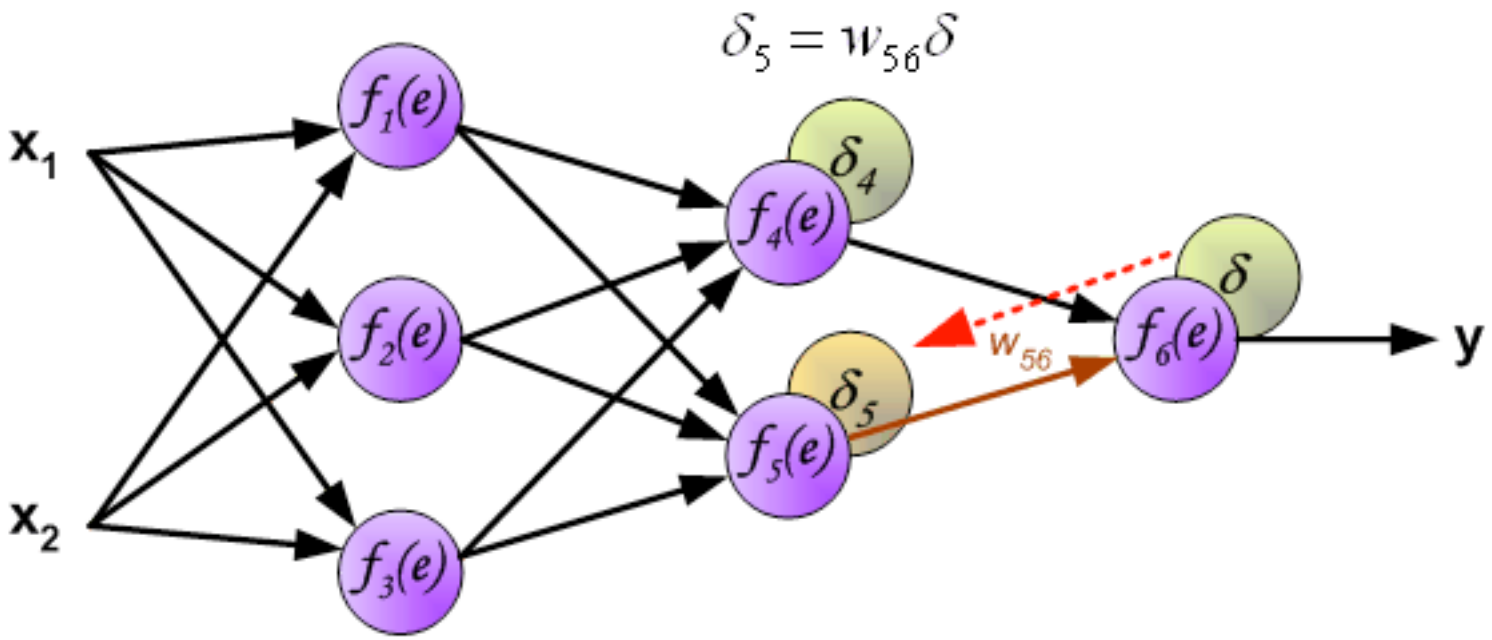


Note: δ here is equivalent to $\frac{\partial E}{\partial o_j}$ in the previous slides, and assuming $f'(e) = 1.0$.

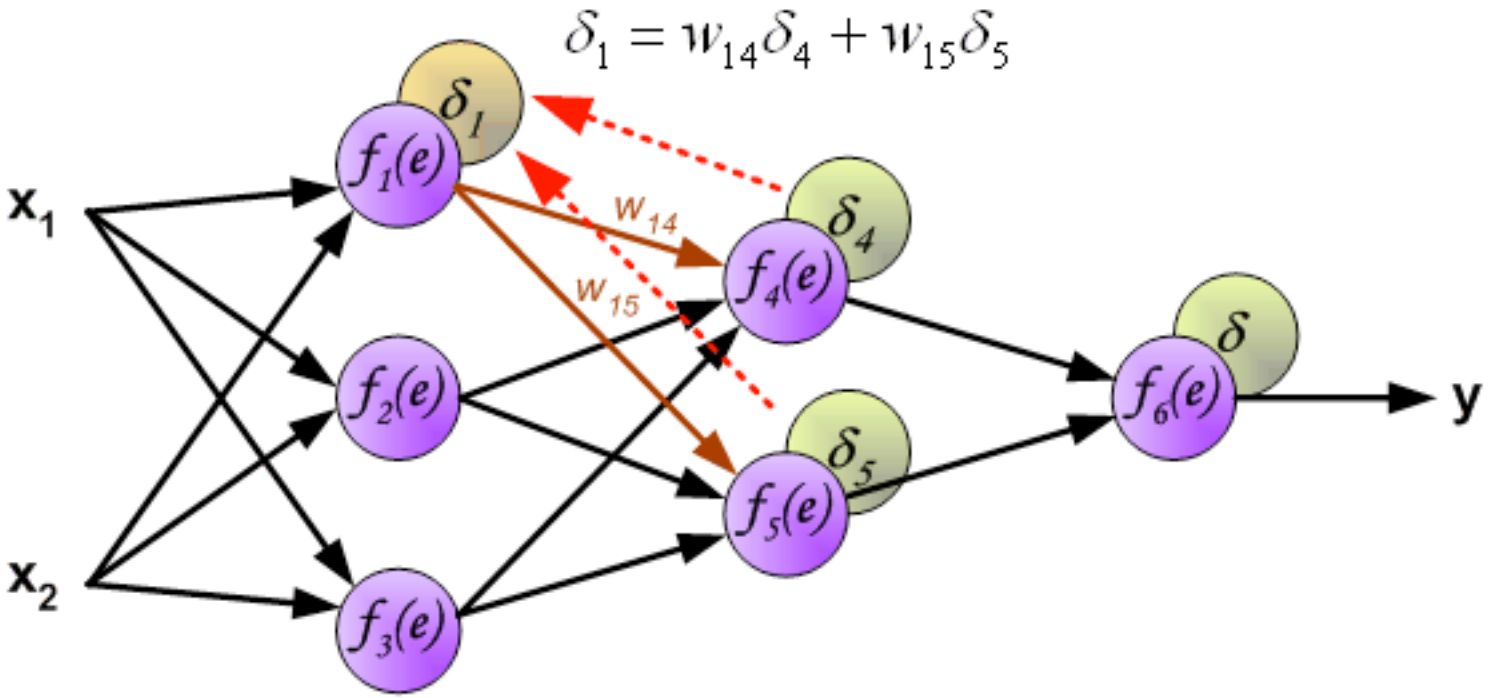
Example of Back-propagation



Example of Back-propagation



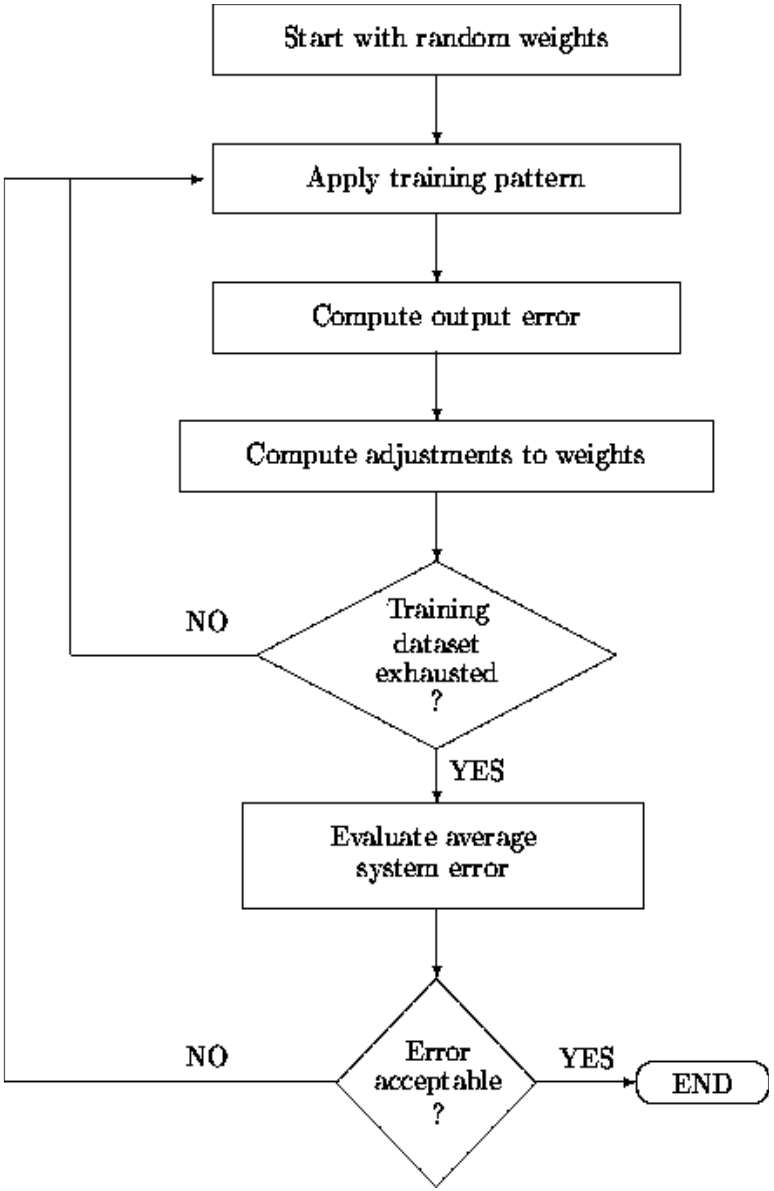
Example of Back-propagation



Some Comments on NNs So Far

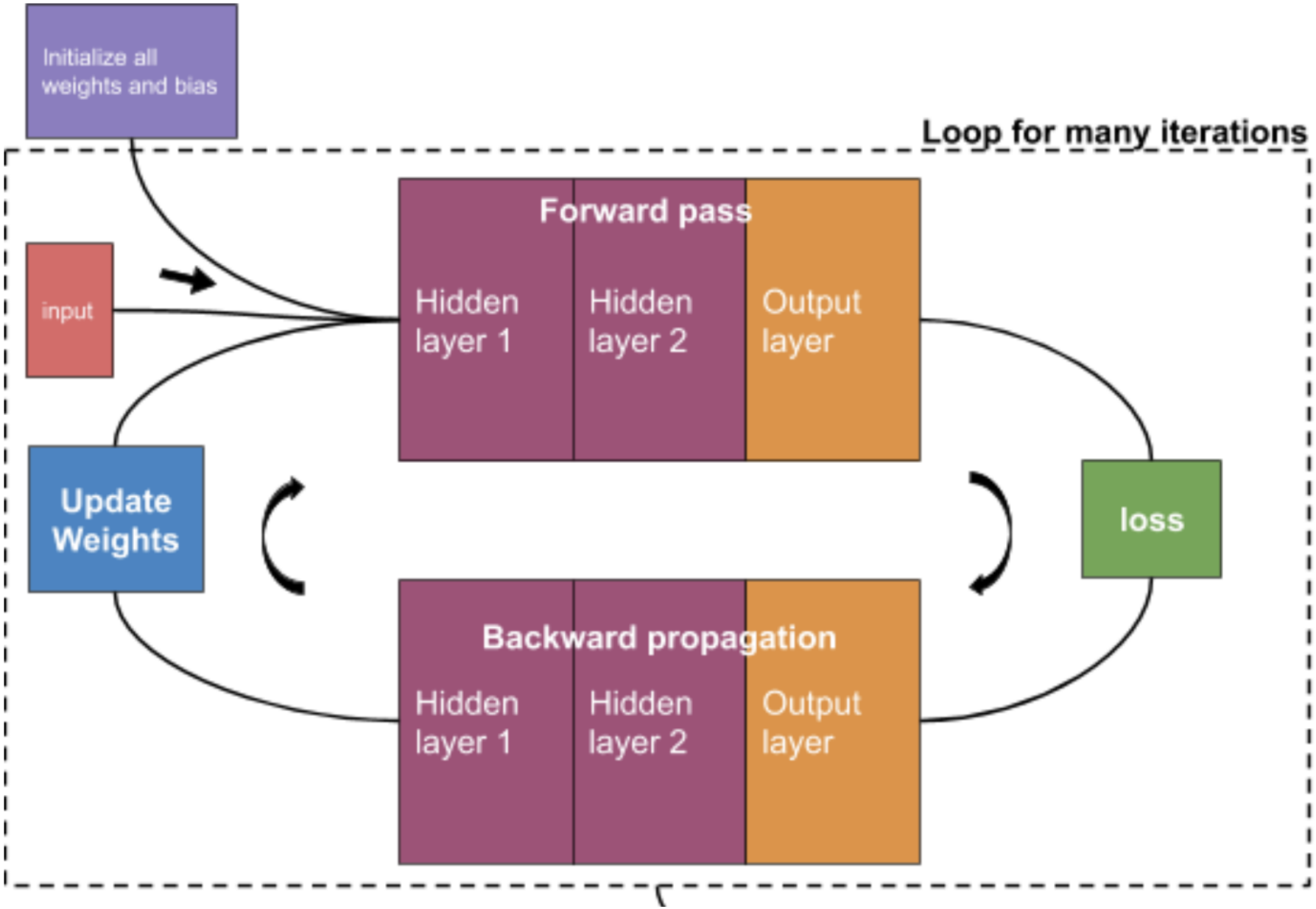
- Multi-layer networks can overcome the representational limitations of single-layer
  - Although may have issues with local minima
- Recurrent networks are even more powerful, but more complex to understand and work with
- Most common applications are in continuous *perceptual* or *perceptuomotor* domains
  - E.g., speech, driving, handwriting, fraud detection
- Recently have been efforts to understand relationship between NN and Bayesian learning

Training a Multilayer Network



prediction

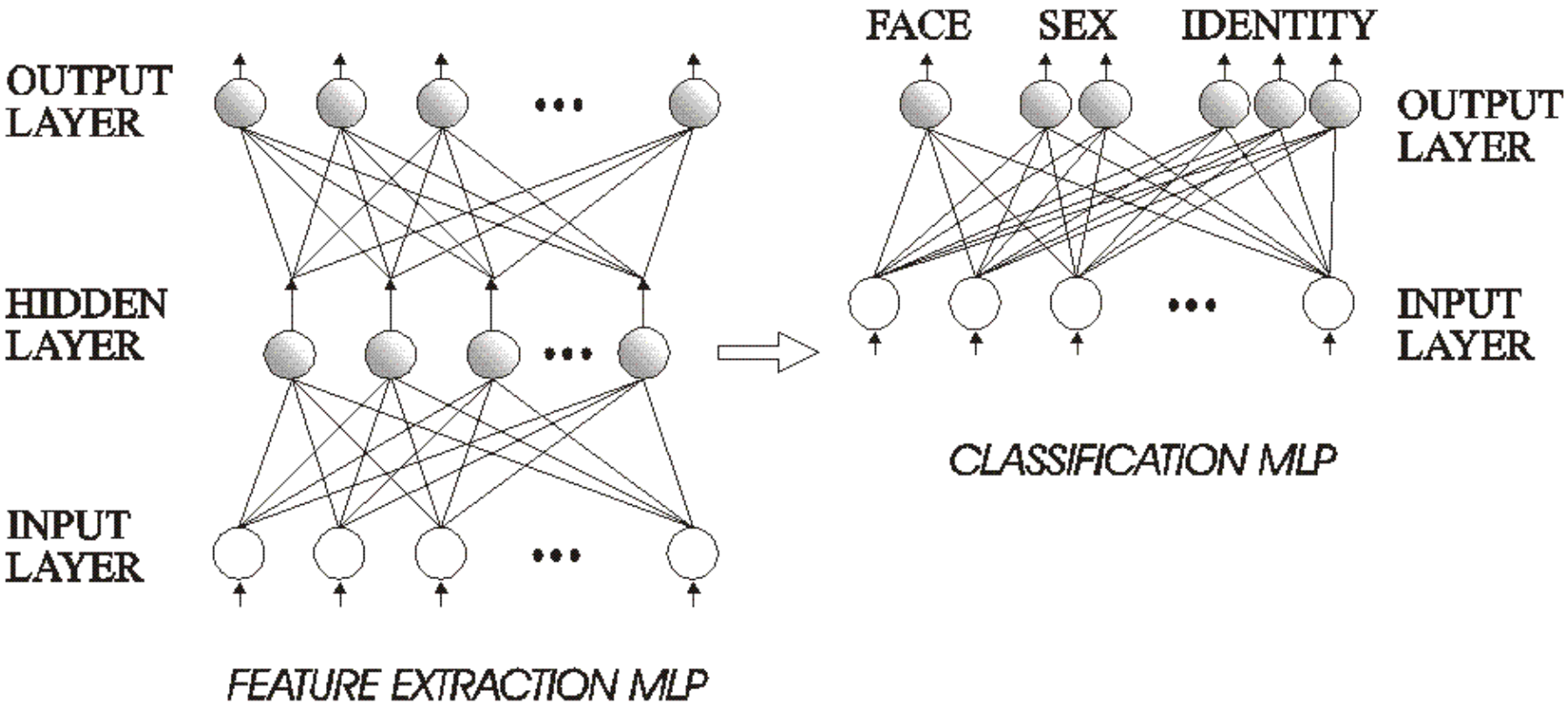
Back propagation


Initialize all weights and bias
Loop for many iterations
Forward pass
input
Hidden layer 1
Hidden layer 2
Output layer
Update Weights
loss
Backward propagation
Hidden layer 1
Hidden layer 2
Output layer

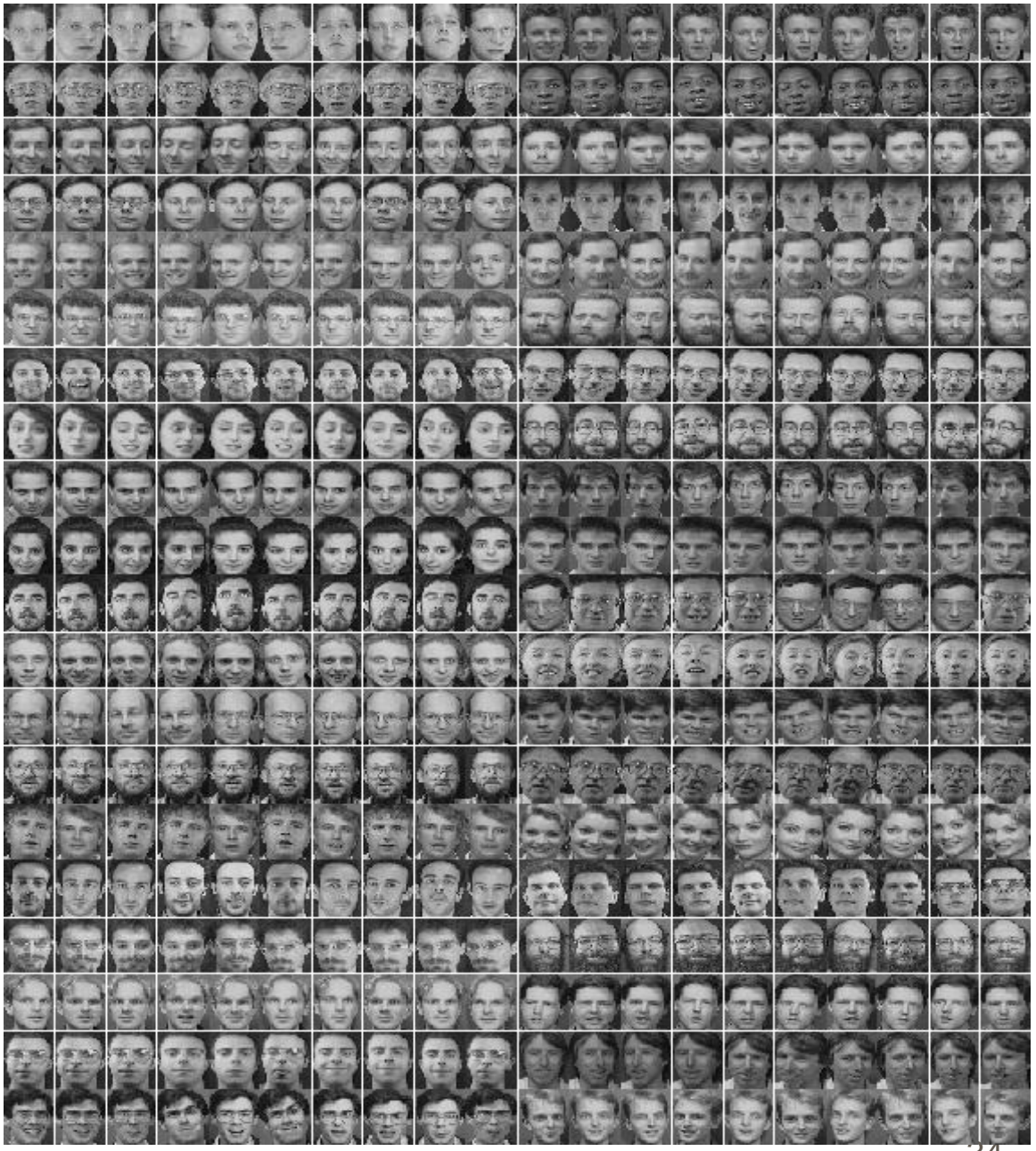
Face Recognition (Example)

Example: Face Recognition

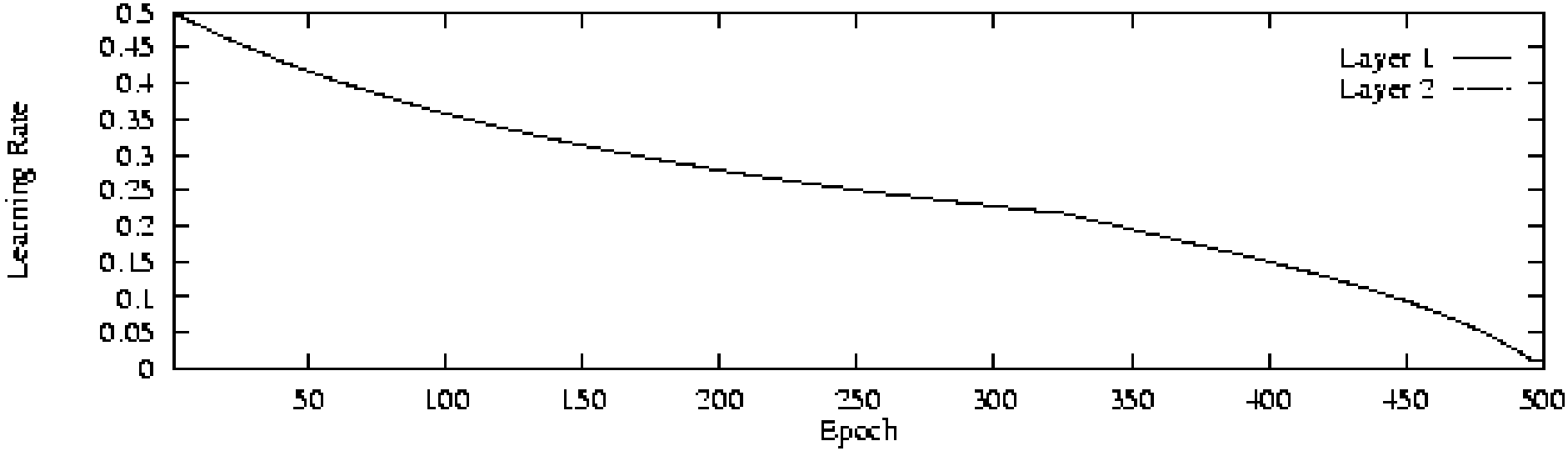
- Using the 2-stage approach (feature extraction + Classification)



Training Data

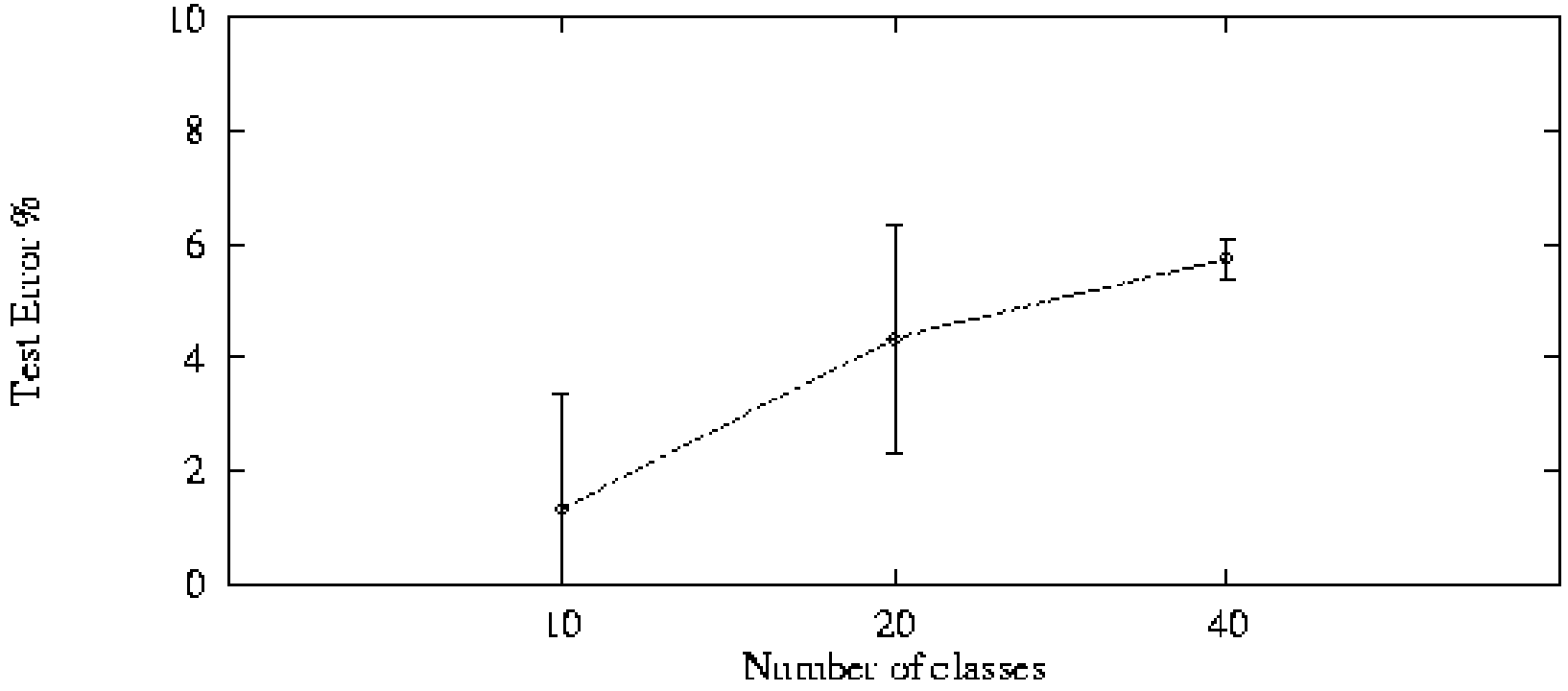


Learning Rate



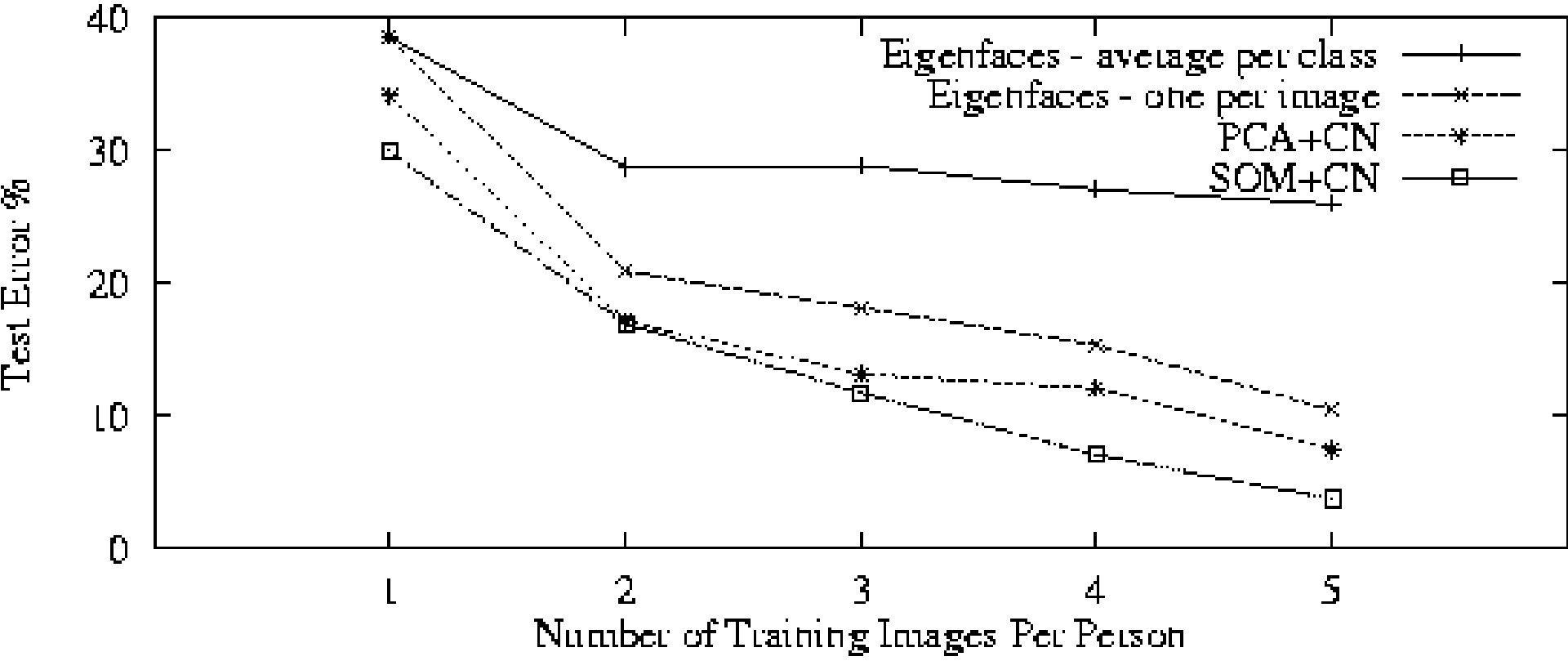
Testing / Evaluation

- Look at performance as a function of network complexity



Testing / Evaluation

- Comparison with other known techniques

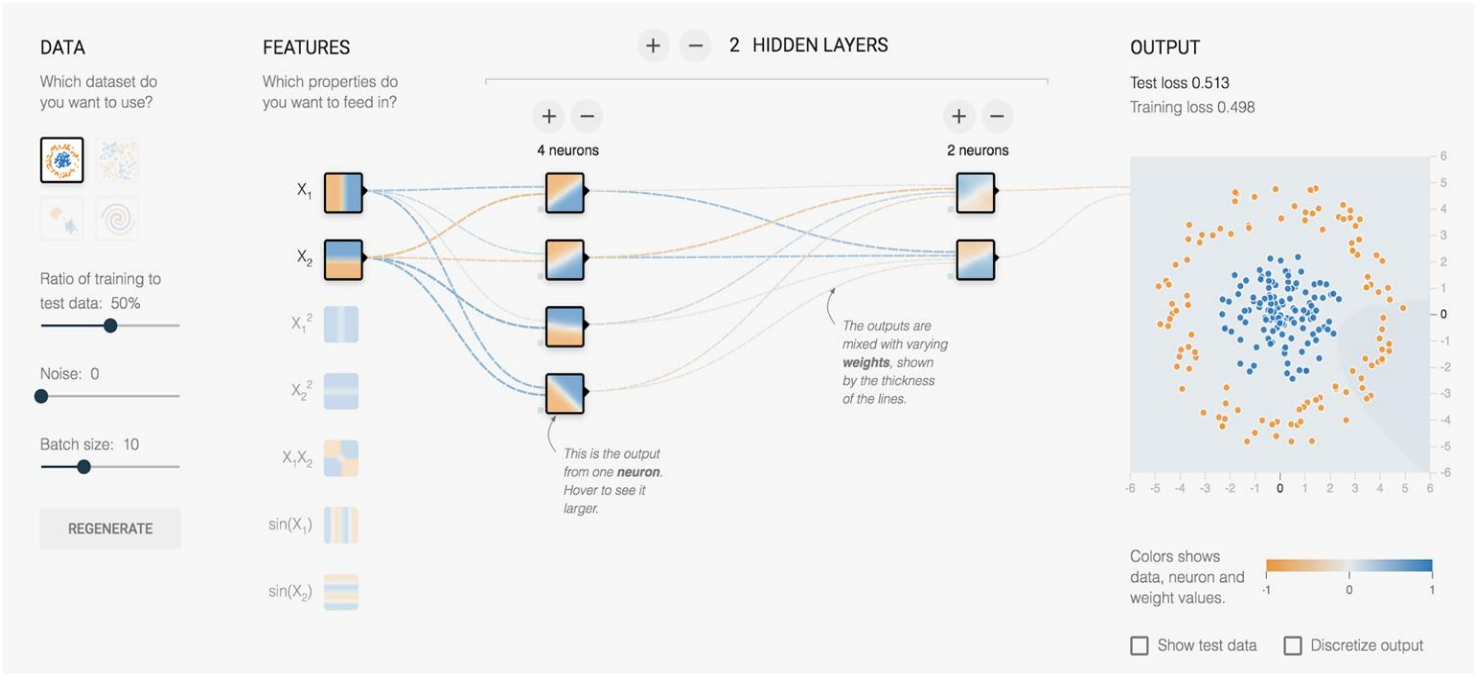


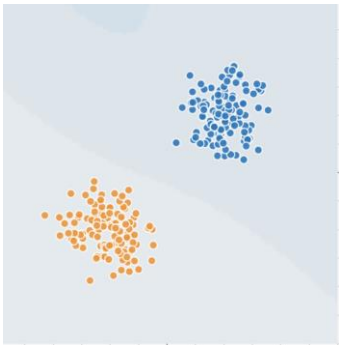
Learning from a Blackbox Using NN

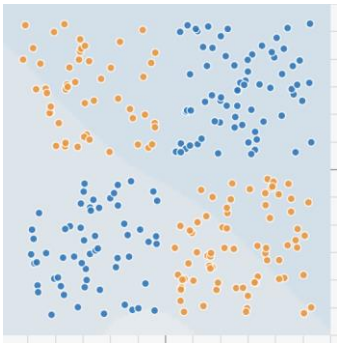
- Training data: more than >2 attributes
- Multiple Classes: more than binary class
- Design your neural networks
  - Choose the number of hidden layers and hidden nodes
- Training your neural networks
  - Implement the Backpropagation algorithm
  - Select the number of epochs and your learning rate
  - You can divide your training set into “batches” to train your NN

Google's Neural Network Playground

- https://www.playground.tensorflow.org/







Modeling the Brain

General Comments: Artificial Neural Networks

Artificial Neural Networks provide…

- A new computing paradigm

- A technique for developing trainable classifiers, memories, dimension-reducing mappings, etc

- A tool to study brain functions

Converging Frameworks

- **Artificial intelligence (AI):** build a "packet of intelligence" into a machine
- **Cognitive psychology:** explain human behavior by interacting processes (schemas) "in the head" but not localized in the brain
- **Brain Theory:** interactions of components of the brain -

  - computational neuroscience

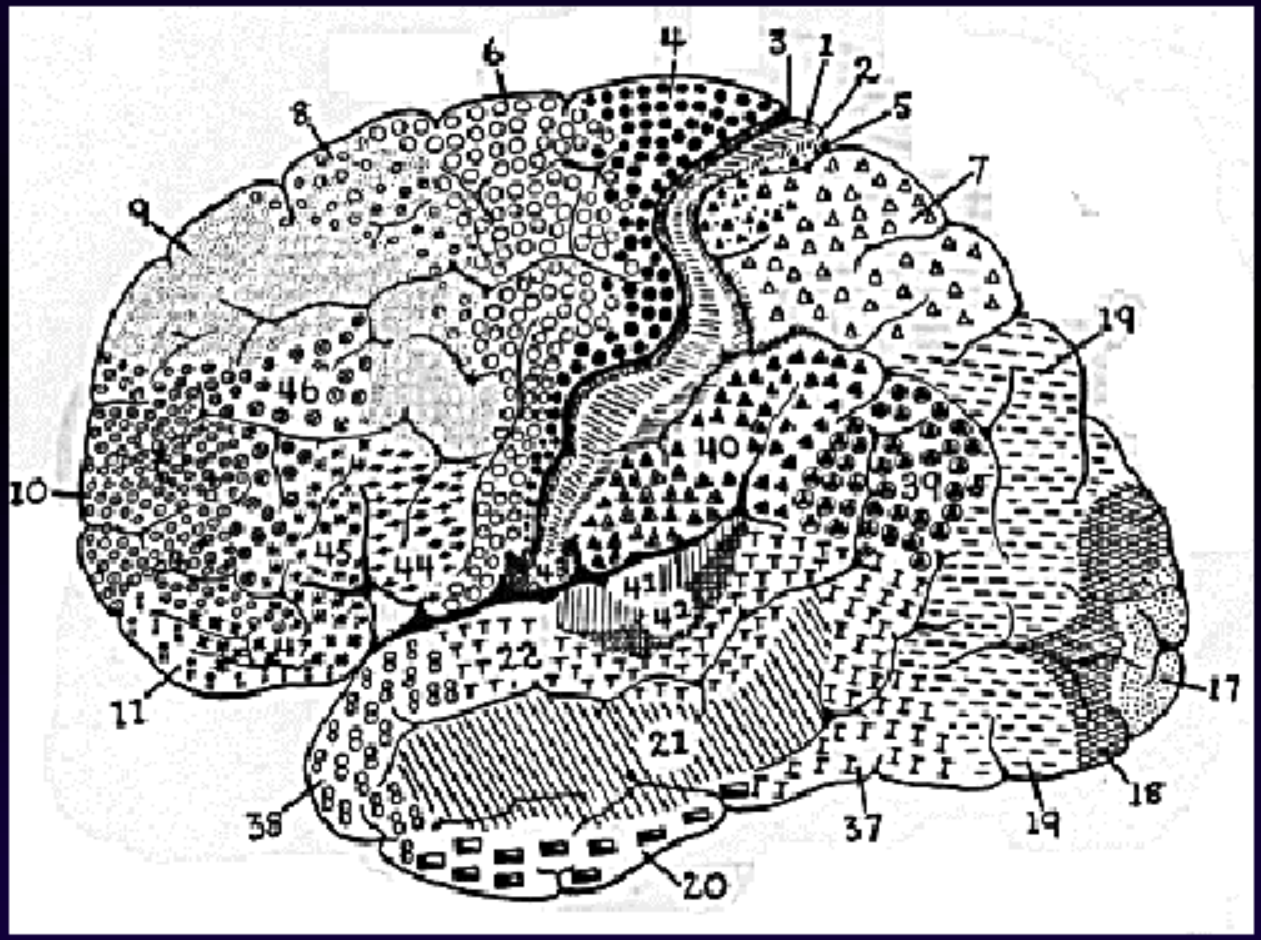
  - neurologically constrained-models

- and abstracting from them as both **Artificial intelligence** and **Cognitive psychology:**

  - connectionism: networks of trainable "quasi-neurons" to provide "parallel distributed models" little constrained by neurophysiology

  - abstract (computer program or control system) information processing models

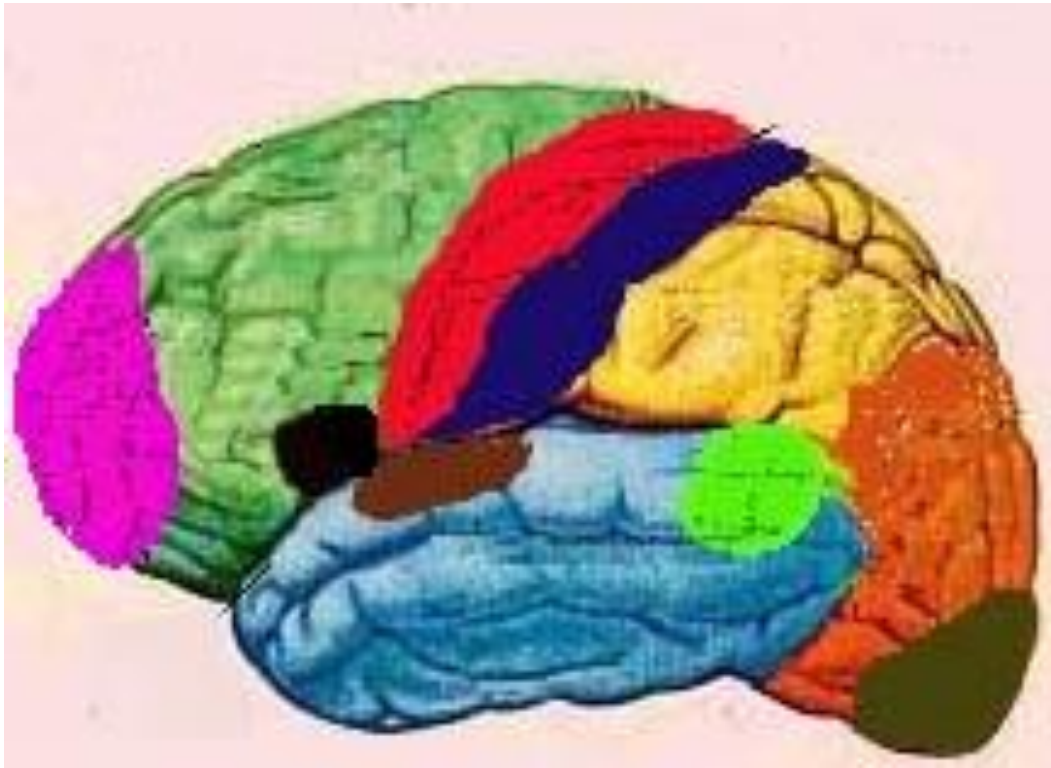
Brodmann's cytoarchitectural map of Cortical Areas



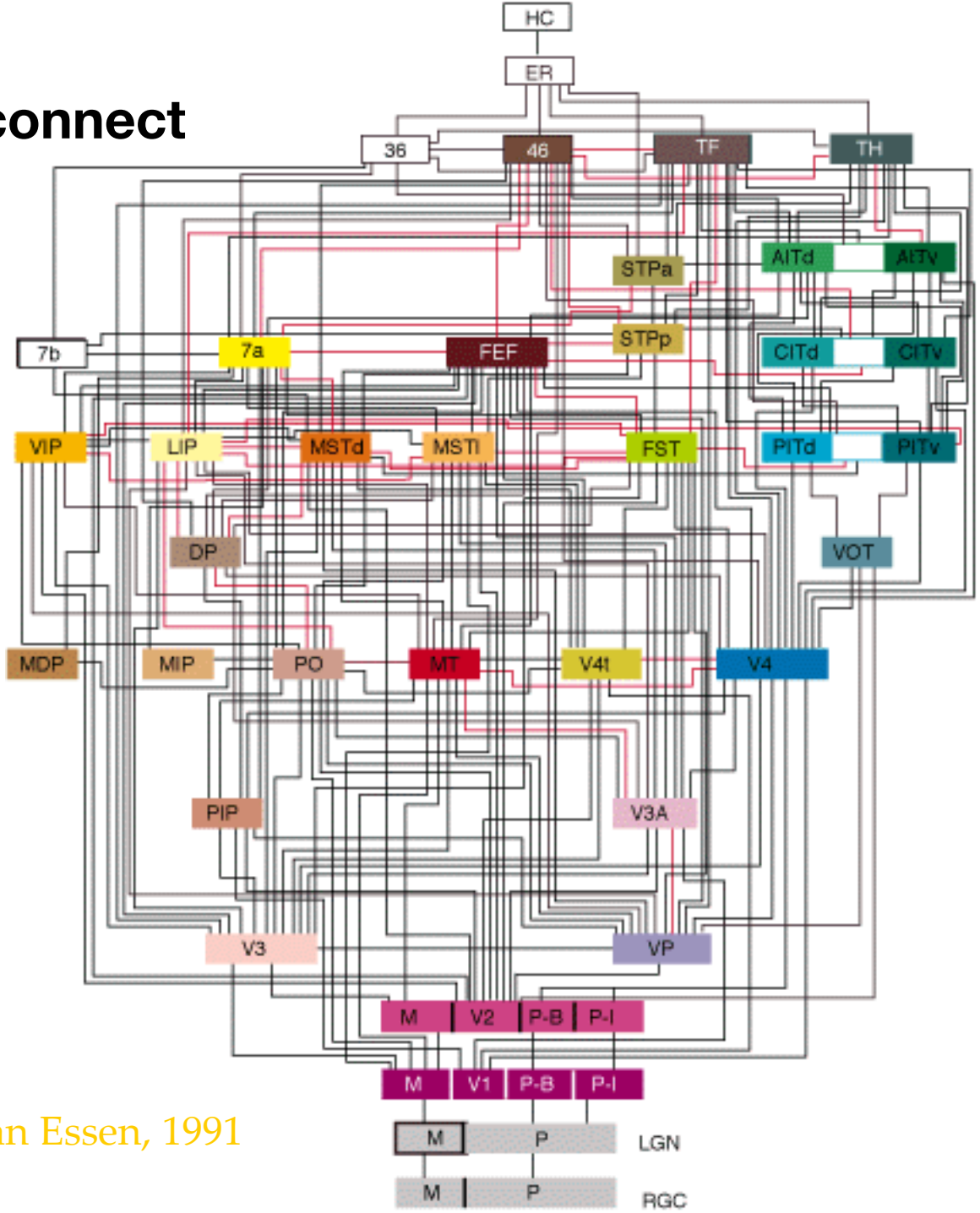
Lateral View

Major Functional Areas

- Primary motor: voluntary movement
- Primary somatosensory: tactile, pain, pressure, position, temp., mvt.
- Motor association: coordination of complex movements
- Sensory association: processing of multisensory information
- Prefrontal: planning, emotion, judgement
- Speech center (Broca's area): speech production and articulation
- Wernicke's area: comprehension of speech
- Auditory: hearing
- Auditory association: complex auditory processing
- Visual: low-level vision
- Visual association: higher-level vision



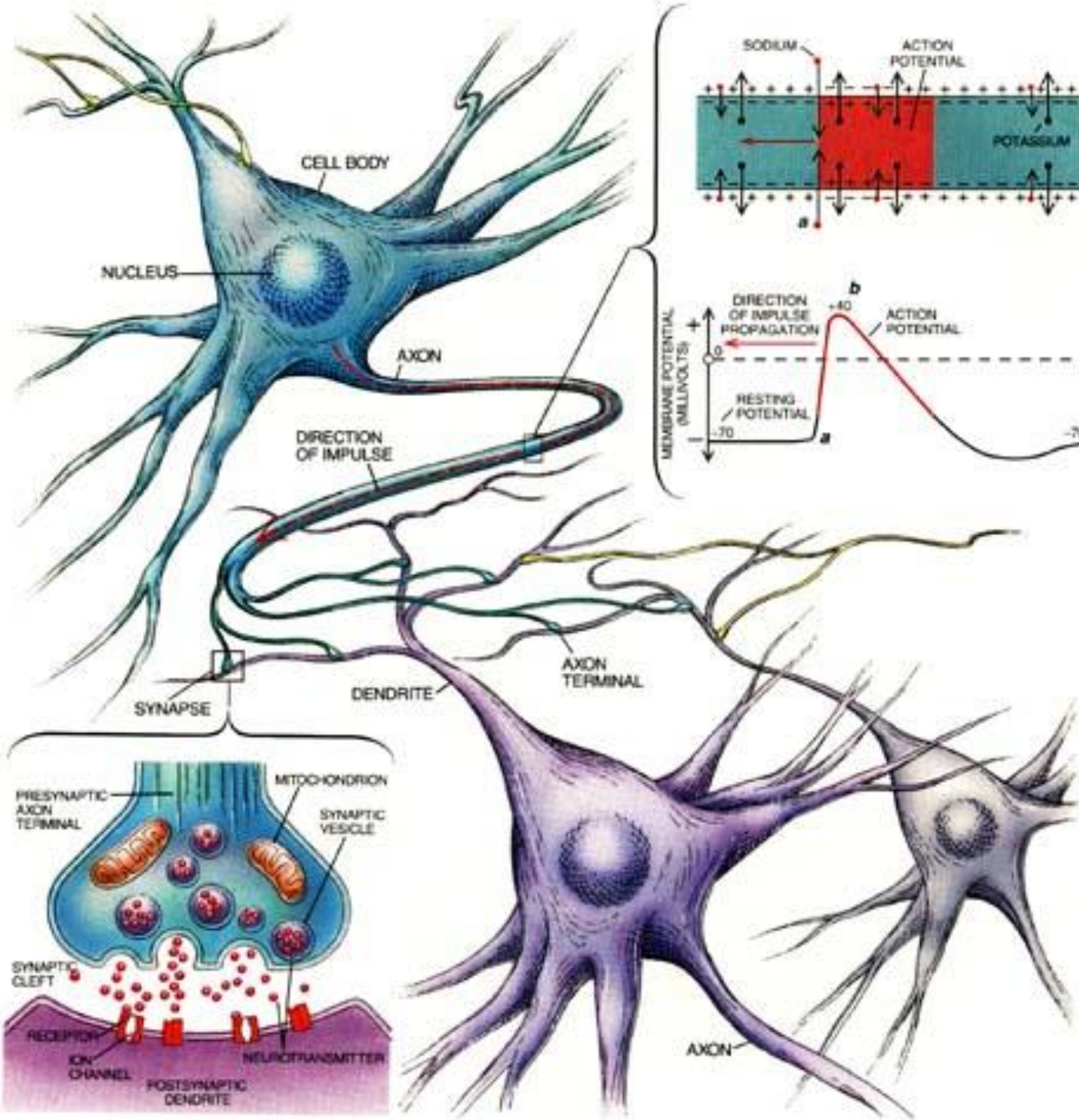
Interconnect



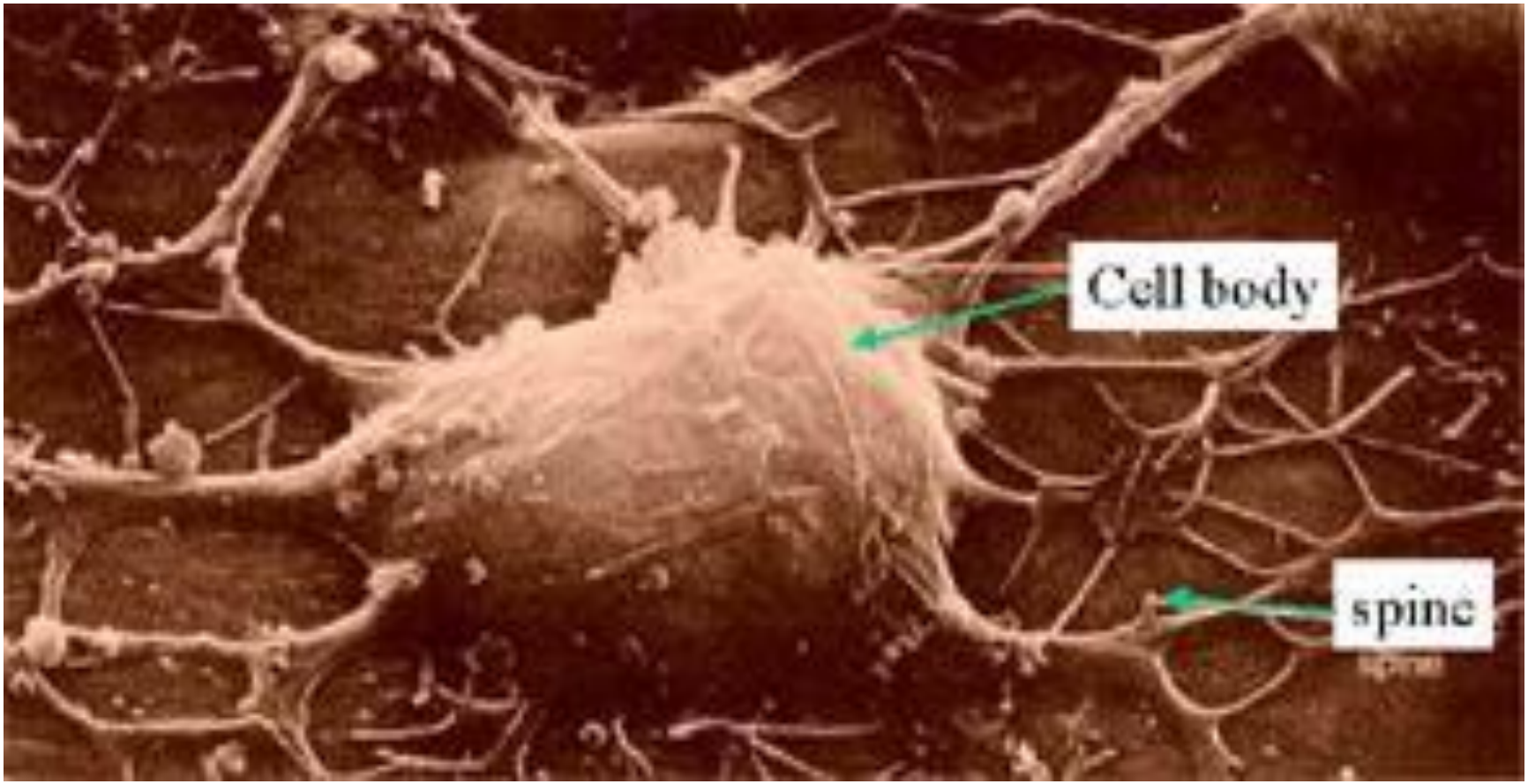
Felleman & Van Essen, 1991

Remember?
Neurons & synapses

Key terms:
Axon
Dendrites
Synapses
Soma (cell body)



Electron Micrograph of a Real Neuron



Approaches to Neural Modeling

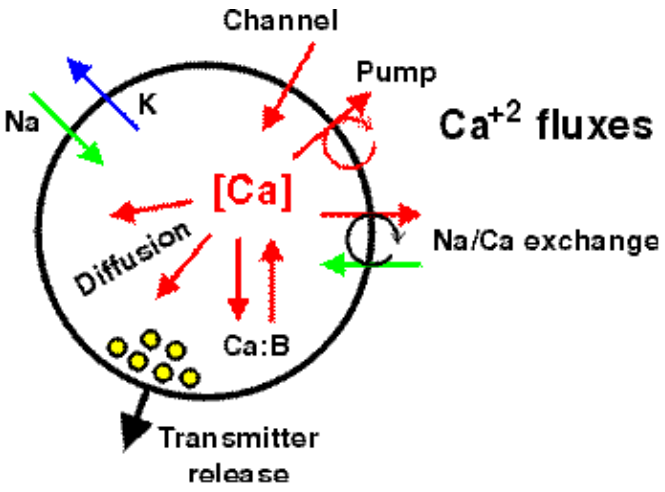
- Biologically-realistic, detailed models
  - E.g., cable equation, multi-compartment models
  - The Hodgkin-Huxley model
  - Simulators like NEURON (Yale) or GENESIS (Caltech)

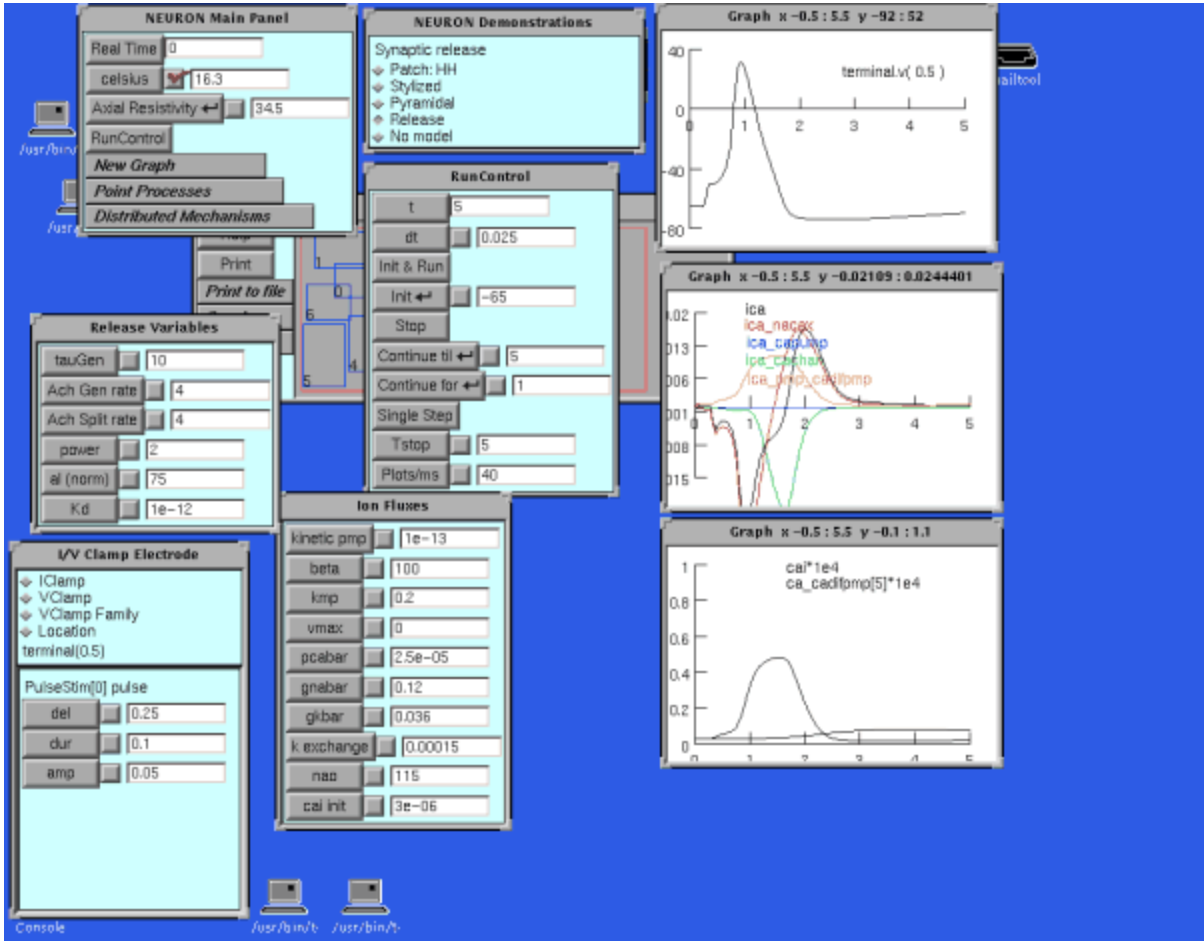
- More abstract models, still keeping realism in mind
  - E.g., integrate & fire model, simple and low detail but preserves spiking behavior

- Highly abstract models, neurons as operators
  - E.g., McCulloch & Pitts model
  - Classical "neural nets" modeling

Detailed Neural Modeling

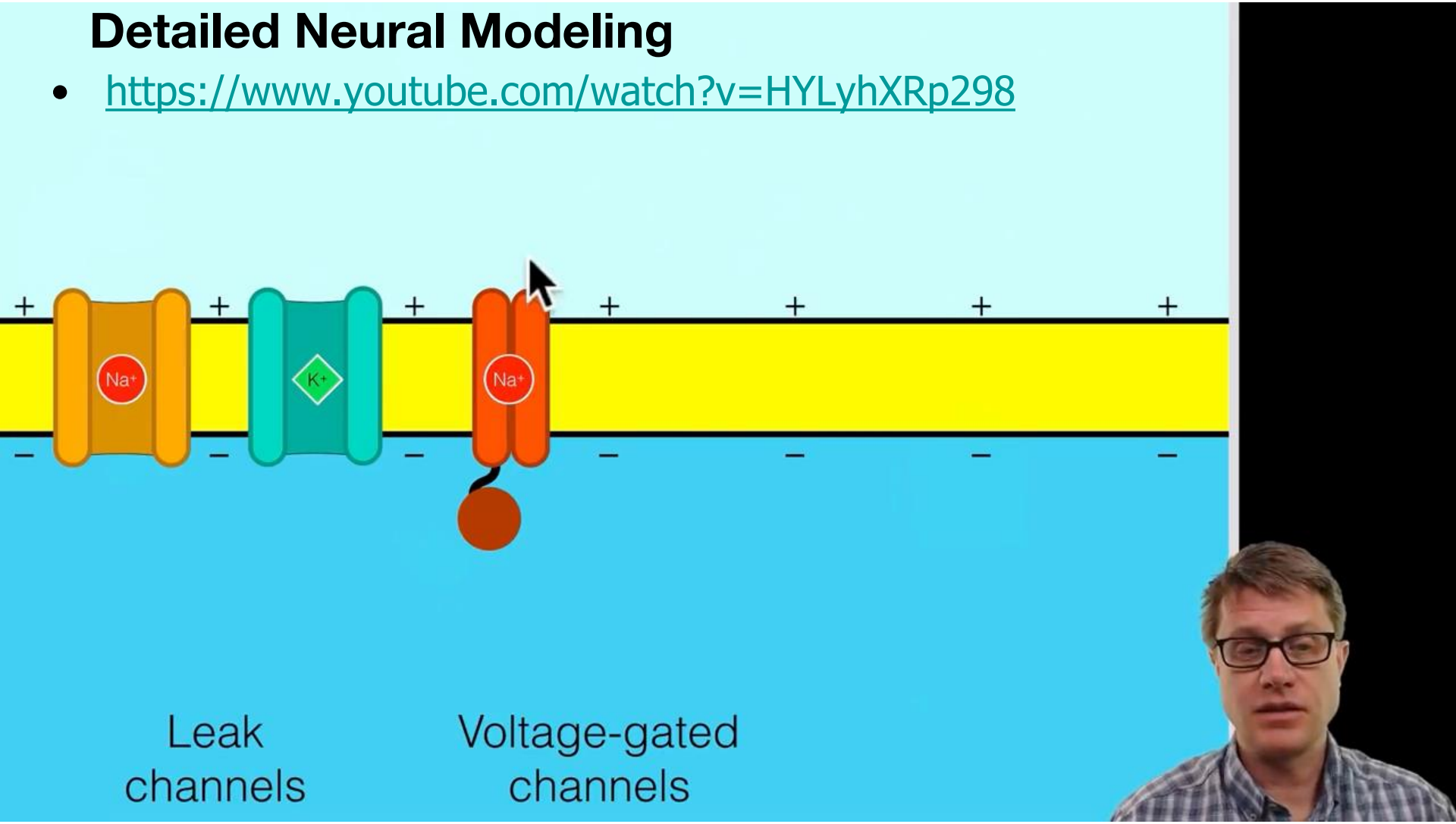
A simulator, called "Neuron" has been developed at Yale to simulate the Hodgkin-Huxley equations, as well as other membranes/channels/etc. See http://www.neuron.yale.edu/





Detailed Neural Modeling

- https://www.youtube.com/watch?v=HYLyhXRp298

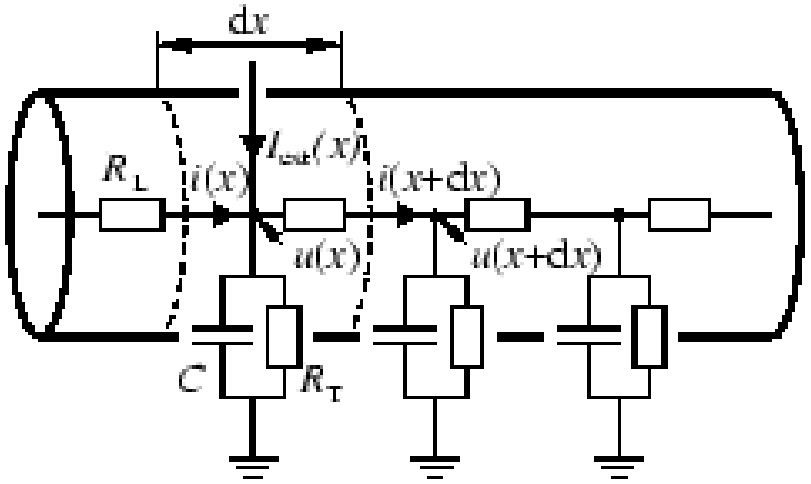


The Cable Equation

- See http://diwww.epfl.ch/~gerstner/SPNM/SPNM.html

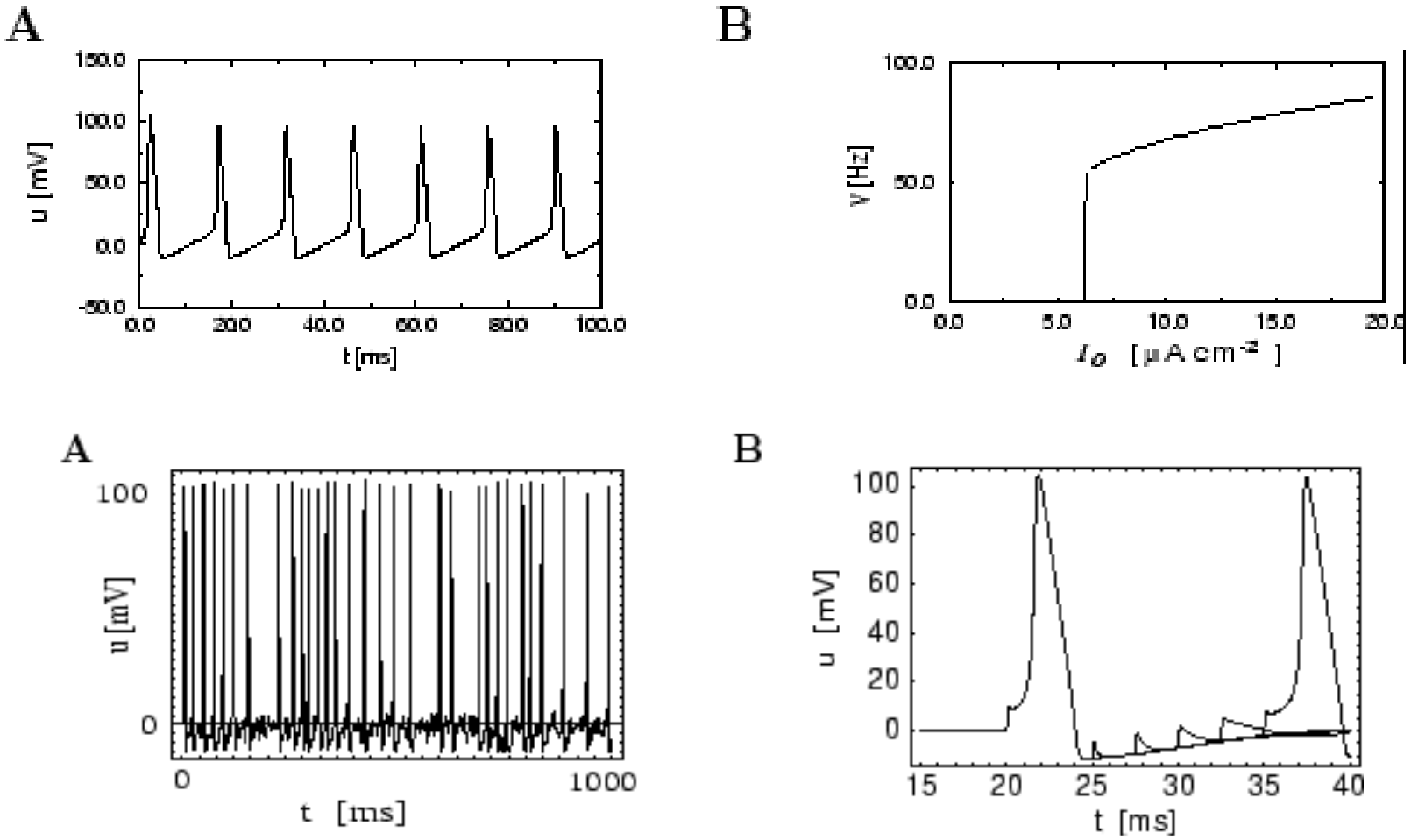
for excellent additional material (some reproduced here).

- Just a piece of passive dendrite can yield complicated differential equations which have been extensively studied by electronicians in the context of the study of coaxial cables (TV antenna cable):



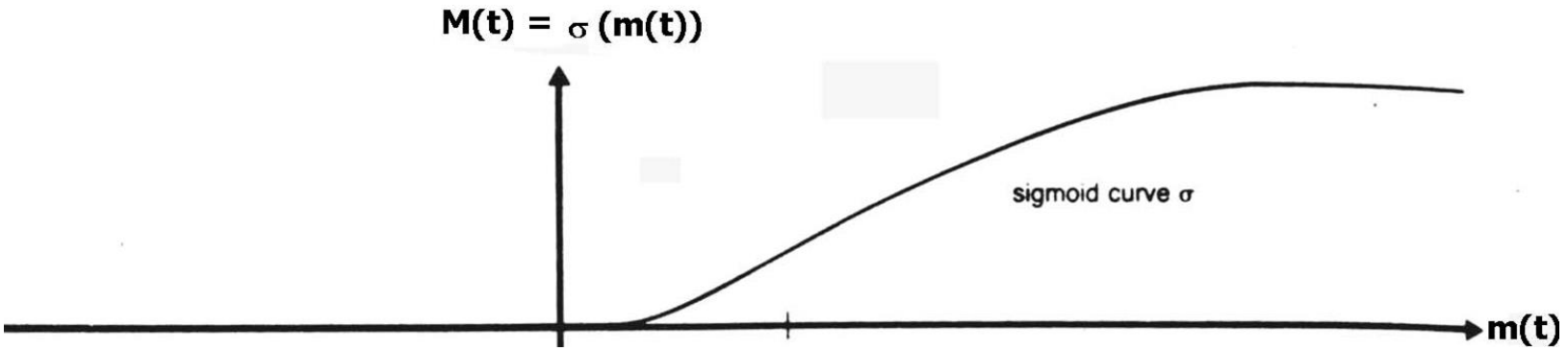
The Hodgkin-Huxley Model

Example spike trains obtained…



Leaky Integrator Neuron

- The simplest "realistic" neuron model is a continuous time model based on using the firing rate (e.g., the number of spikes traversing the axon in the most recent 20 msec.) as a continuously varying measure of the cell's activity
- The state of the neuron is described by a single variable, the membrane potential.
- The firing rate is approximated by a sigmoid, function of membrane potential.



Leaky Integrator Model

$$\tau \dot{m}(t) = - m(t) + h$$

has solution $m(t) = e^{-t/\tau} m(0) + (1 - e^{-t/\tau})h$

$\rightarrow h$ for time constant $\tau > 0$.

- We now add synaptic inputs to get the

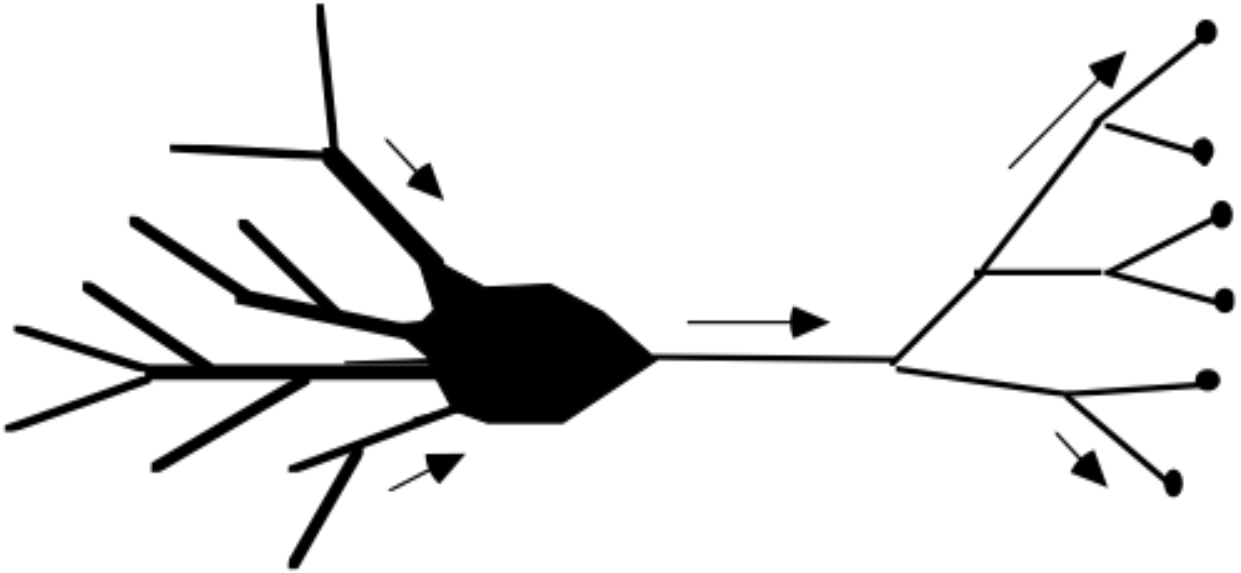
Leaky Integrator Model:

$$\tau \dot{m}(t) = - m(t) + \Sigma_i w_i X_i(t) + h$$

where $X_i(t)$ is the firing rate at the i^{th} input.

- Excitatory input ($w_i > 0$) will increase $\dot{m}(t)$
- Inhibitory input ($w_i < 0$) will have the opposite effect.
- $X(t) = g(m(t))$ with $g()$ a sigmoid relates output to membrane potential

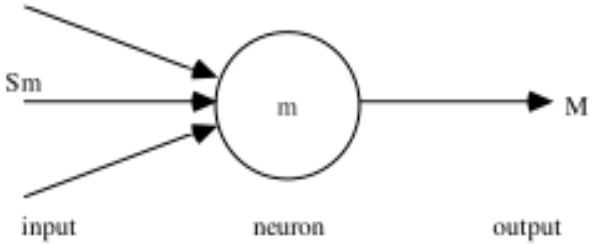
The “Basic” Biological Neuron



- The soma and dendrites act as the input surface; the axon carries the outputs.
- The tips of the branches of the axon form synapses upon other neurons or upon effectors (though synapses may occur along the branches of an axon as well as the ends). The arrows indicate the direction of "typical" information flow from inputs to outputs.

Vision, AI and ANNs

- **1940s: beginning of Artificial Neural Networks**



McCullogh & Pitts, 1942

$\Sigma_i \; w_i x_i \geq \theta$

Perceptron learning rule (Rosenblatt, 1962)

Backpropagation

Hopfield networks (1982)

Kohonen self-organizing maps

…

Excitatory and Inhibitory Synapses

- We call a synapse

  excitatory if $w_i > 0$, and

  inhibitory if $w_i < 0$.

- We also associate a threshold θ with each neuron

- A neuron fires (i.e., has value 1 on its output line) at time t+1 if the weighted sum of inputs at t reaches or passes θ:

$$\mathbf{y(t+1) = 1 \quad \text{if and only if} \quad \Sigma\, w_i x_i(t) \geq \theta}$$

Increasing the Realism of Neuron Models

- The McCulloch-Pitts neuron of 1943 is important

as a basis for

- logical analysis of the neurally computable, and
- current design of some neural devices (especially when augmented by **learning rules** to adjust synaptic weights).

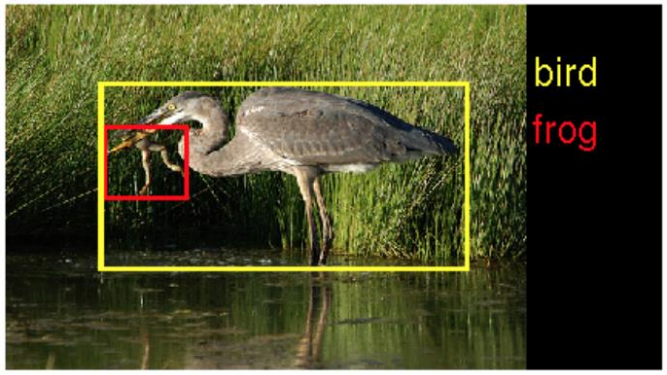
- However, it is no longer considered a useful model for making contact with neurophysiological data concerning real neurons.

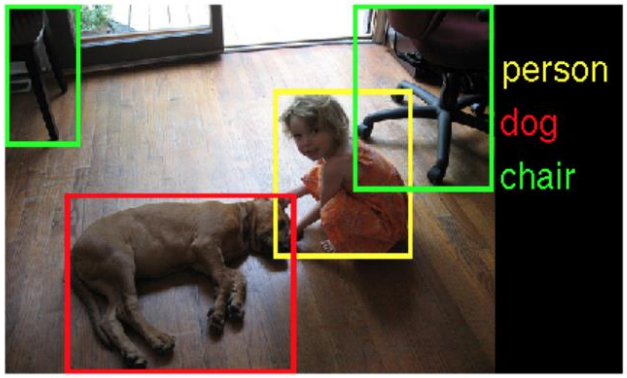
Deep Neural Networks

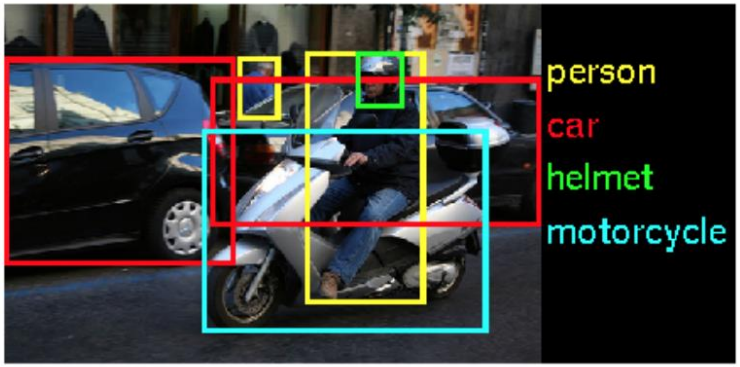
- ImageNet visual recognition challenge: recognize objects, animals, people, etc (200 categories) in photographs.

Comparative scale

		PASCAL VOC 2012	ILSVRC 2014
Number of object classes		20	**200**
Training	Num images	5717	456567
	Num objects	13609	478807
Validation	Num images	5823	20121
	Num objects	13841	55502
Testing	Num images	10991	40152
	Num objects	---	---







Deep Neural Networks

- Top performers for the 2014 challenge (including winner) are deep neural networks!

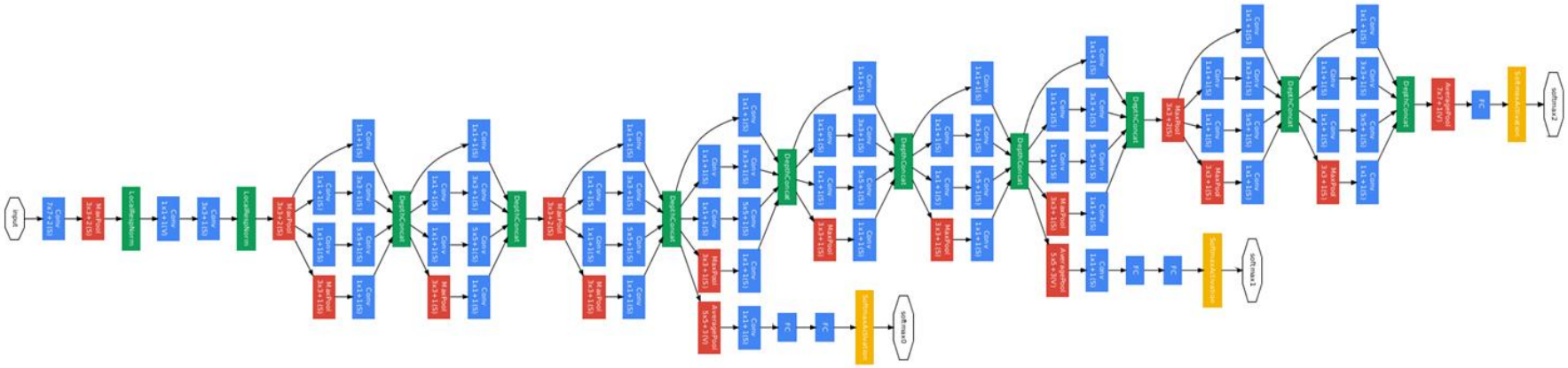
Task 1b: Object detection with additional training data

Object detection with additional training data: Ordered by number of categories won

Team name	Entry description	Description of outside data used	**Number of object categories won**	mean AP
GoogLeNet	Ensemble of detection models. Validation is 44.5% mAP	Pretraining on ILSVRC12 classification data.	142	0.439329
CUHK DeepID-Net	Combine multiple models described in the abstract without contextual modeling	ImageNet classification and localization data	29	0.406659
Deep Insight	Combination of three detection models	Three CNNs from classification task are used for initialization.	27	0.404517
UvA-Euvision	Deep learning with outside data	ImageNet 1000	1	0.354213
Berkeley Vision	R-CNN baseline	The CNN was pre-trained on the ILSVRC 2013 CLS dataset.	1	0.345213
Trimps-Soushen	Two models combined with nms	ILSVRC2012 classification data	0	0.337485
Trimps-Soushen	Four models combination	ILSVRC2012 classification data	0	0.332469

Deep Neural Networks

- Very deep layered network (27 layers) with hundreds of millions of parameters.
- Trained on very large datasets at Google.



Deep Learning

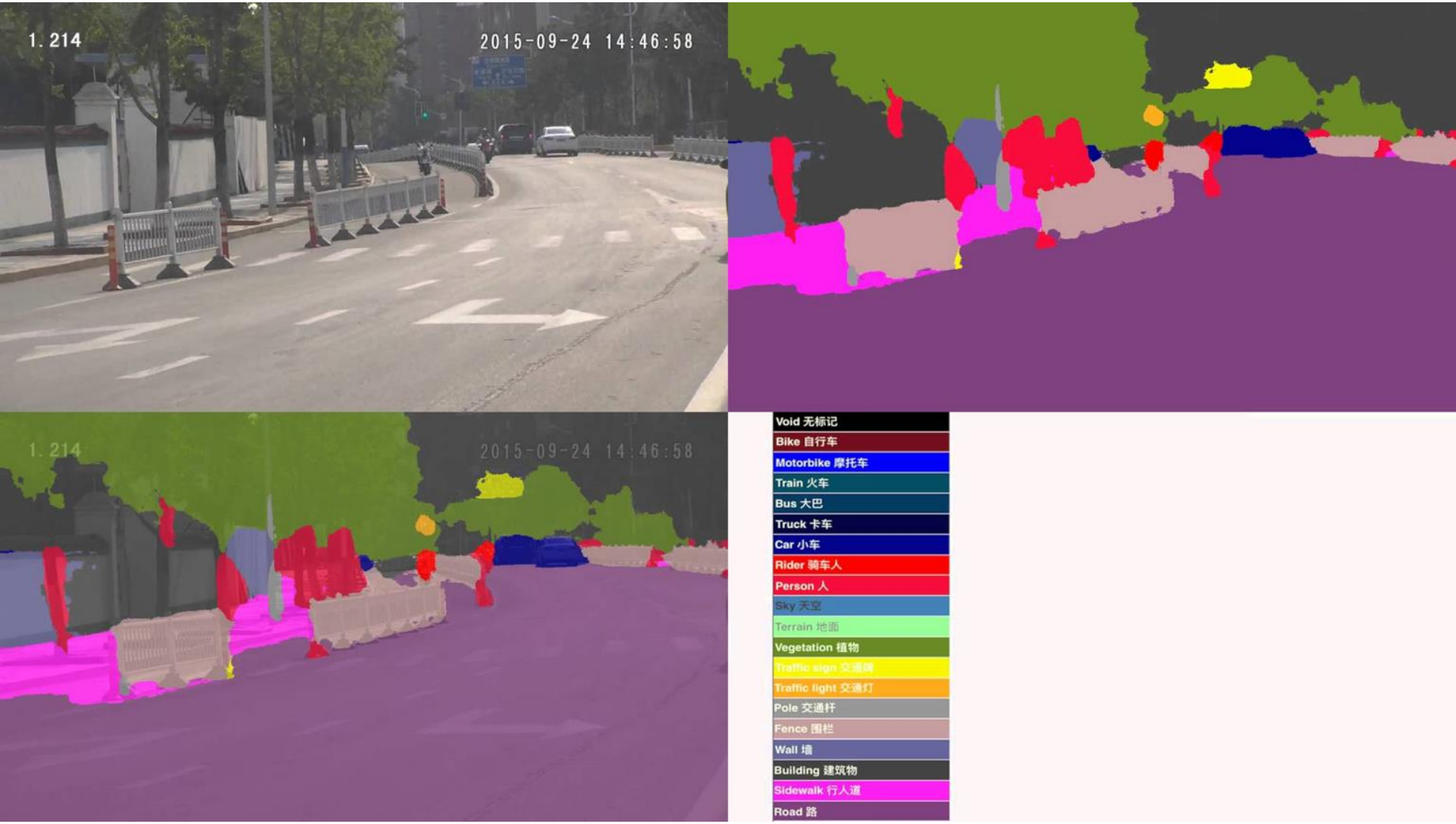
Multilayer neural networks are one form

- Train each layer as if it were a standalone neural network
- A hierarchy of loosely connected learning systems

Basic principles are the same as what you've learned

- But with newer methods for dealing with larger data sets
- Finding representations and structures similar to the brain's
    - e.g., for facial recognition
- And exploiting steady increases in computing power

Scene Understanding



Deep Learning

Train each layer as if it were standalone

- Train a neural network to identify the road
- Train a neural network to identify a car
- Train a neural network to identify a truck
- …

A hierarchy of loosely connected learning systems

- Train a neural network to identify a clear / busy / unsafe road

Applications

Applications: Classification

Business
- Credit rating and risk assessment
- Insurance risk evaluation
- Fraud detection
- Insider dealing detection
- Marketing analysis
- Mailshot profiling
- Signature verification
- Inventory control

Engineering
- Machinery defect diagnosis
- Signal processing
- Character recognition
- Process supervision
- Process fault analysis
- Speech recognition
- Machine vision
- Speech recognition
- Radar signal classification

Security
- Face recognition
- Speaker verification
- Fingerprint analysis

Medicine
- General diagnosis
- Detection of heart defects

Science
- Recognising genes
- Botanical classification
- Bacteria identification

Applications: Modelling

Business

- Prediction of share and commodity prices
- Prediction of economic indicators
- Insider dealing detection
- Marketing analysis
- Mailshot profiling
- Signature verification

Engineering

- Transducer linerisation
- Colour discrimination
- Robot control and navigation
- Process control
- Aircraft landing control
- Car active suspension control
- Printed Circuit auto routing
- Integrated circuit layout
- Image compression

Science

- Prediction of the performance of drugs from the molecular structure
- Weather prediction
- Sunspot prediction

Medicine

- . Medical imaging and image processing

Applicationsh: Forecasting and Detection

Forecasting:

- Future sales
- Production Requirements
- Market Performance
- Economic Indicators
- Energy Requirements
- Time Based Variables

Novelty Detection:

- Fault Monitoring
- Performance Monitoring
- Fraud Detection
- Detecting Rate Features
- Different Cases