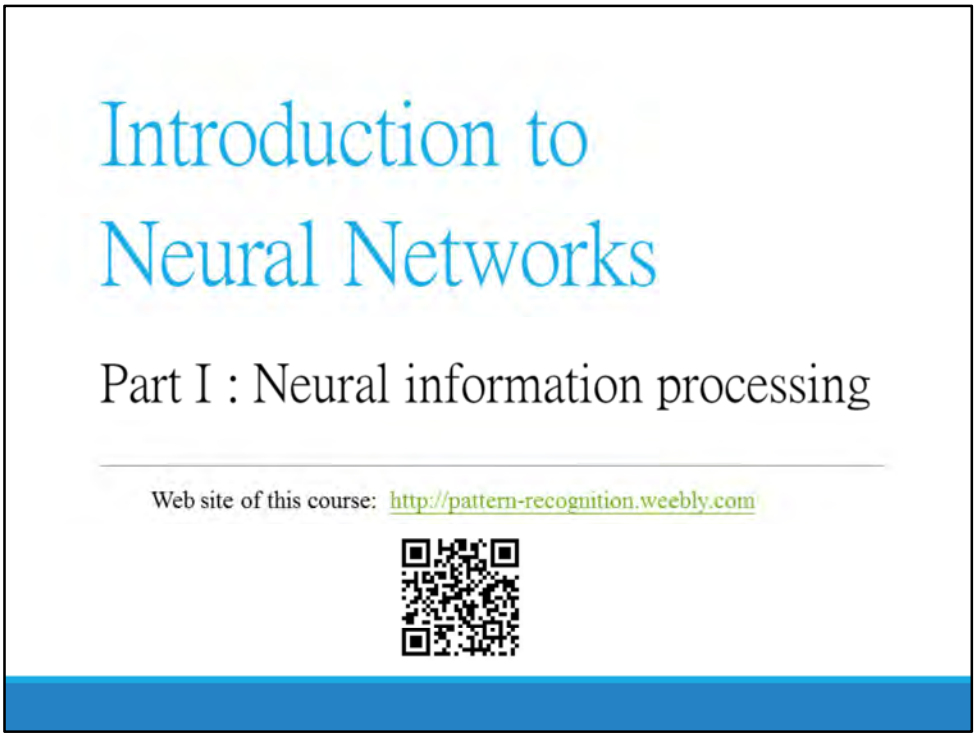
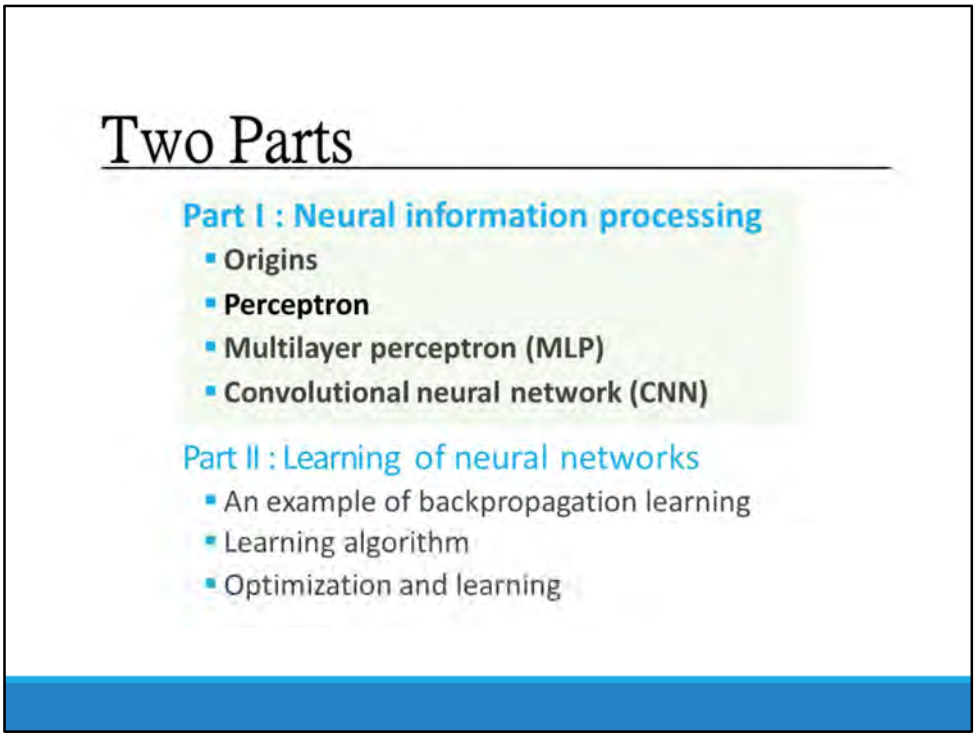
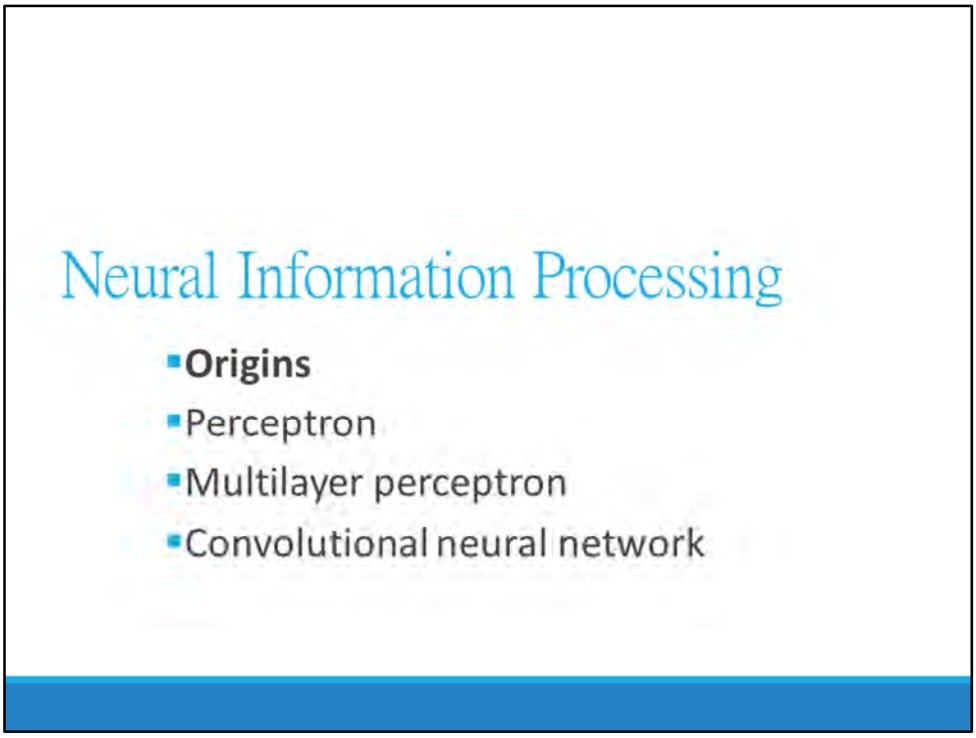


- This is the fourth lecture note of the course PATTERN RECOGNITION in English in 104-2 semester, Electrical Engineering department, Fu-Jen Catholic University.
- This course is taught by Prof. Wang, Yuan-Kai.
- In this lecture note, I will introduce neural networks.
- Web site of this course: http://pattern-recognition.weebly.com.
- Note: URL address of web page will be provided with QR code.
- Reference materials
  - **"Chapter 11 Multilayer Perceptrons," Introduction to Machine Learning, 2nd, E. Alpaydin, MIT Press, 2010.**
  - **Multilayer neural networks, by Leon Bottou, MLSS Tuebingen, 2013.**
  - "Chapter 4 Artificial Neural Networks", Machine Learning, T. Mitchell, McGraw-Hill, 1997. (46 pages)
  - "Chapter 3 The Multi-Layer Perceptron," Machine Learning : an algorithmic perspective, S. Marsland, Chapman and Hall/CRC, 2009.

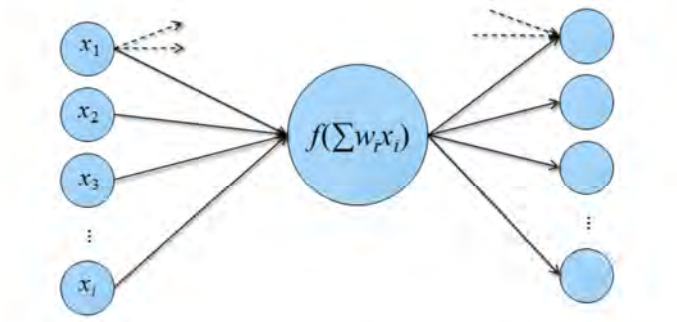


- The first lecture: neural information processing, includes 4 sections. This lecture will be lectured by 1 week.
    - Origins of neural networks
    - Perceptron
    - MLP
    - CNN
- The second lecture: learning of neural networks, includes 4 sections. The second lecture will also be lectured by 1 week.
- What is neural networks?
    - Neural networks are massively parallel, distributed processing systems representing a new computational technology built on the analogy to the human information processing system.



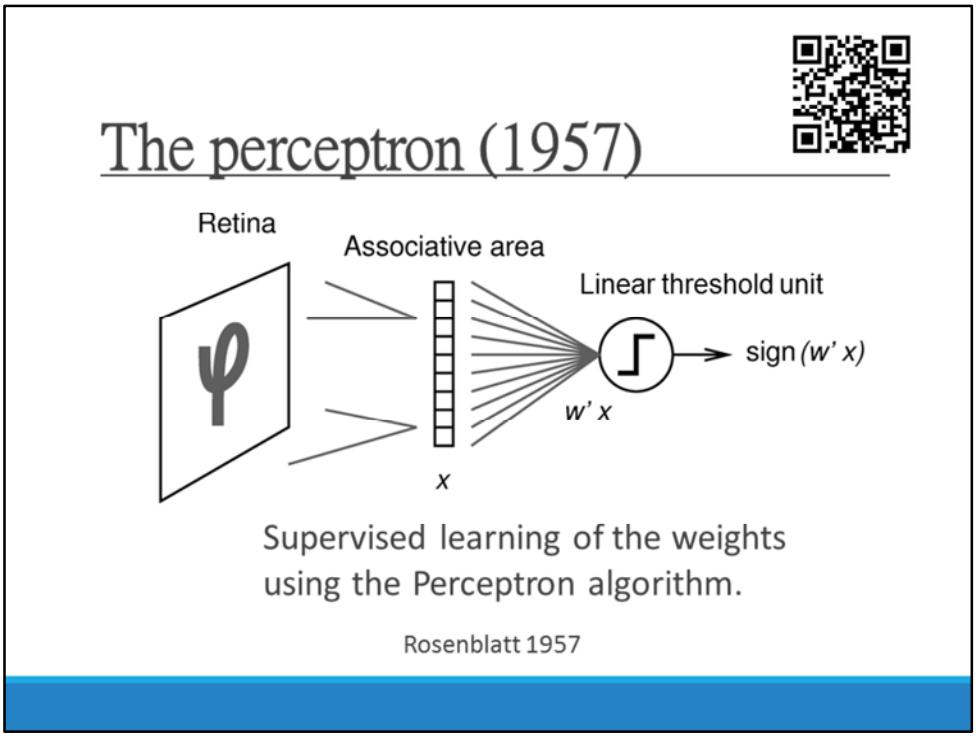
- Origins of neural networks: from 1943 to 1986.
- The evolution of artificial neural networks, from the early idea of neuro-physiologist Heb (1949) about the structure and the behaviour of a biological neural system up to the recent model of artificial neural system, was very long.
- The first cornerstones here were laid down by the neurologists McCulloch and Pitts (1943) who, using formal logic, modelled neural networks using the neurons as binary devices with fixed thresholds interconnected by synapses.
- Nevertheless, the list of pioneer contributors in this field of work is long. It certainly includes the names of distinguished researchers like Rosenblatt (1958), who extended the idea of the ***computing neuron*** to the ***perceptron*** as an element of a self-organizing computational network capable of learning by feedback and by structural adaptation.
- Further pioneer work was also done by Widrow and Hoff (1960), who created and implemented the analogue electronic devices known as ADALINE (Adaptive Linear Element) and MADALINE (Multiple ADALINE) to mimic the neurons, or perceptrons. They used the least mean squares algorithm, simply called the ***delta rule***, to train the devices to learn the pattern vectors presented to their inputs.
- In 1969, Minsky and Papert (1969) portrayed perceptron history in an excellent way but their view, that the multilayer perceptron (MLP) systems had limited learning capabilities similar to the one-layer perceptron system, was later disproved by Rumelhart and McClelland (1986).
- Rumelhart and McClelland (1986) in fact showed that multilayer neural networks have outstanding nonlinear discriminating capabilities and are capable of learning more complex patterns by ***backpropagation learning***. This essentially terminates the most fundamental development phase of perceptron-based neural networks.

McCulloch & Pitts (1943)

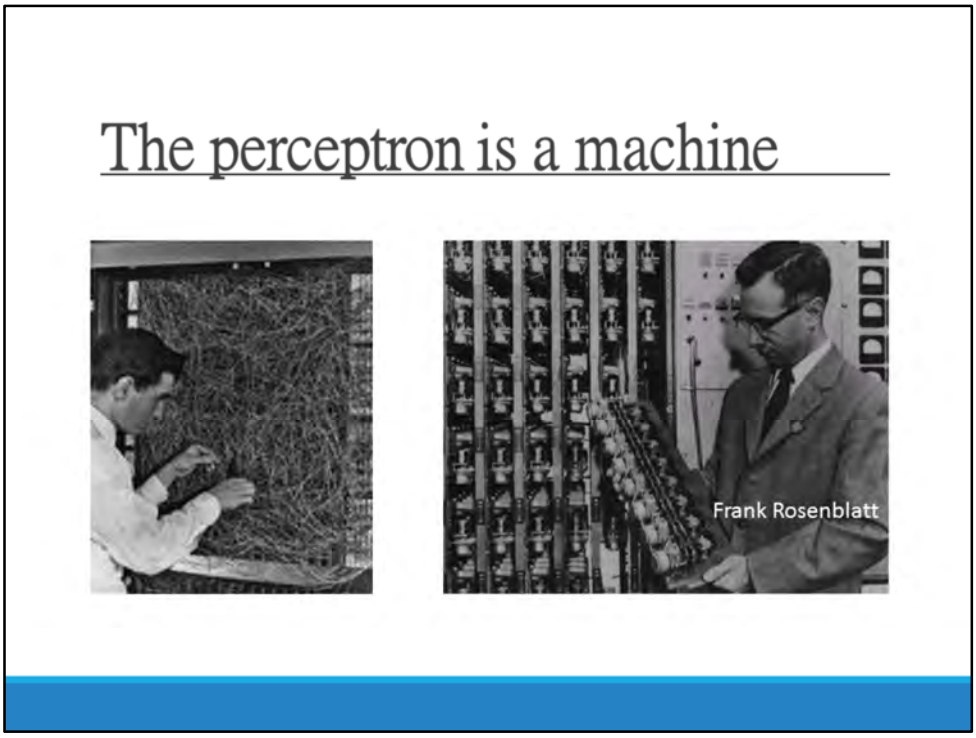


A simplified neuron model: the Linear Threshold Unit.

- The neurologists McCulloch and Pitts (1943) used formal logic to model neural networks using the neurons as binary devices with fixed thresholds interconnected by synapses.

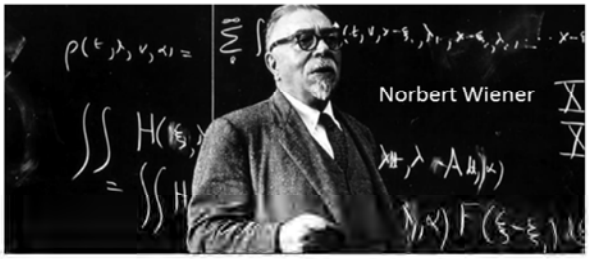


- The perceptron algorithm dates back to the late 1950s; its first implementation, in custom hardware, was one of the first artificial neural networks to be produced.



- https://en.wikipedia.org/wiki/Perceptron
  - The perceptron algorithm was invented in 1957 at the Cornell Aeronautical Laboratory by Frank Rosenblatt,[3] funded by the United States Office of Naval Research.[4] The perceptron was intended to be a machine, rather than a program, and while its first implementation was in software for the IBM 704, it was subsequently implemented in custom-built hardware as the "Mark 1 perceptron". This machine was designed for image recognition: it had an array of 400 photocells, randomly connected to the "neurons". Weights were encoded in potentiometers, and weight updates during learning were performed by electric motors.

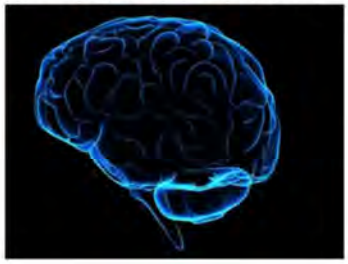
Cybernetics (1948)



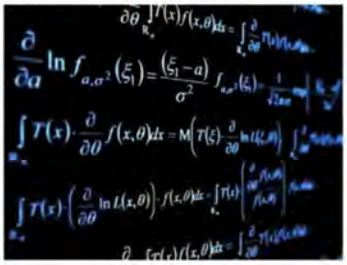
- Mature communication technologies, nascent computing technologies
- Redefining the man-machine boundary

Two camps to design computers

Biological computer



Mathematical computer

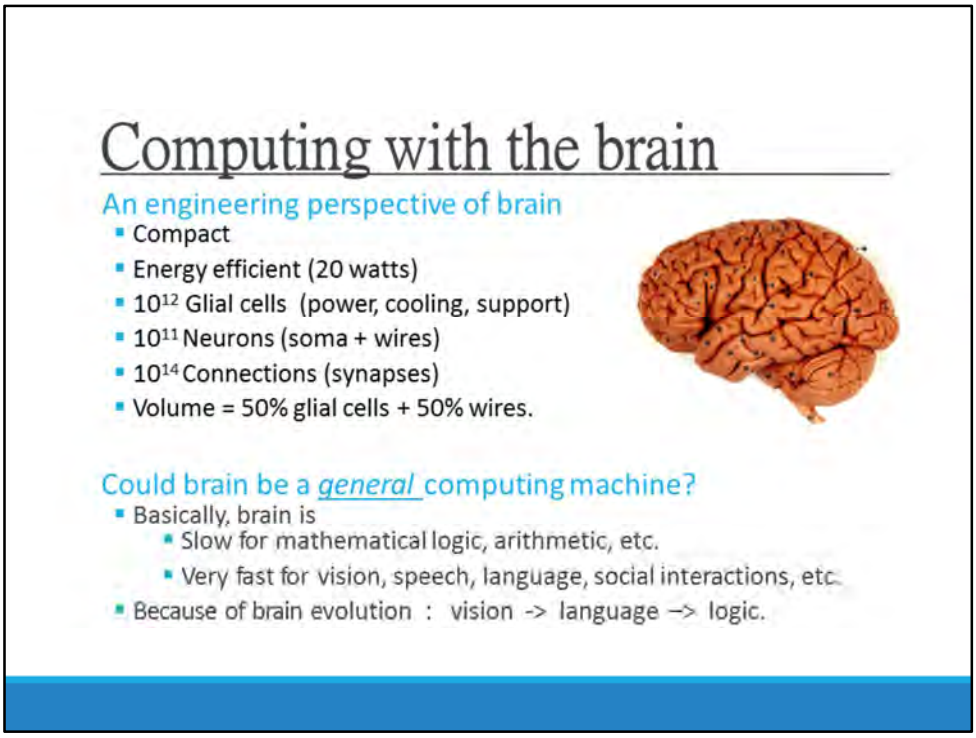


- Which model to emulate : brain or mathematical logic ?
- Mathematical logic won.

Computing with symbols

General computing machines
- Turing machine
- von Neumann machine

Engineering
- Programming
  (reducing a complex task into
  a collection of simple tasks.)
- Computer language
- Debugging
- Operating systems
- Libraries



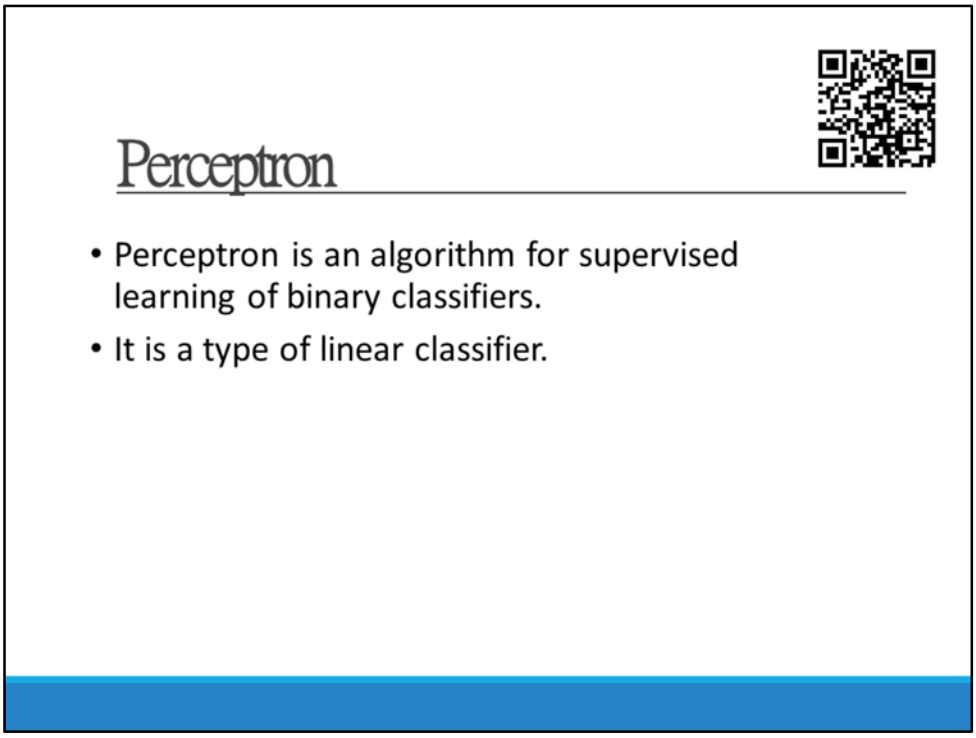
- In engineering, our aim is not to understand the brain per se, but to build useful machines. We are interested in *artificial neural networks* because we believe that they may help us build better computer systems. The brain is an information processing device that has some incredible abilities and surpasses current engineering products in many domains—for example, vision, speech recognition, and learning, to name three. These applications have evident economic utility if implemented on machines. If we can understand how the brain performs these func- tions, we can define solutions to these tasks as formal algorithms and implement them on computers.
- The human brain is quite different from a computer. Whereas a com- puter generally has one processor, the brain is composed of a very large (1011) number of processing units, namely, *neurons*, operating in parallel. Though the details are not known, the processing units are believed to be much simpler and slower than a processor in a computer. What also makes the brain different, and is believed to provide its computational power, is the large connectivity. Neurons in the brain have connections, called *synapses*, to around 104 other neurons, all operating in parallel. In a computer, the processor is active and the memory is separate and passive, but it is believed that in the brain, both the processing and mem- ory are distributed together over the network; processing

is done by the neurons, and the memory is in the synapses between the neurons.

- MNIST:
- ImageNet:

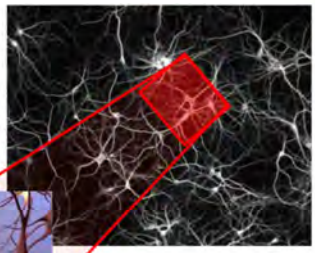
Neural Information Processing

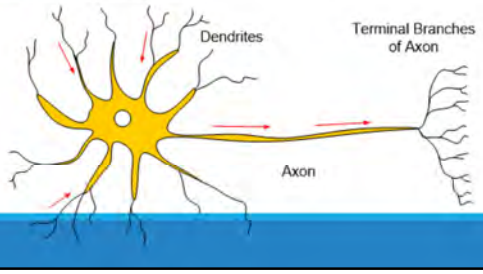
- Origins
- **Perceptron**
- Multilayer perceptron
- Convolutional neural network



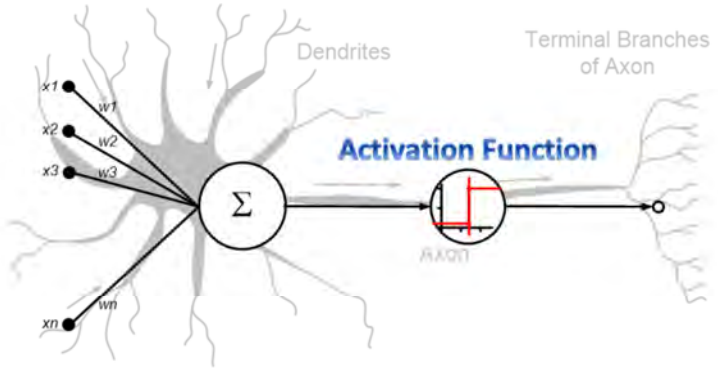
- The perceptron algorithm was invented in 1957 at the Cornell Aeronautical Laboratory by Frank Rosenblatt
- In machine learning, the perceptron is an algorithm for supervised learning of binary classifiers: functions that can decide whether an input (represented by a vector of numbers) belongs to one class or another.
- It is a type of linear classifier, i.e. a classification algorithm that makes its predictions based on a linear predictor function combining a set of weights with the feature vector. The algorithm allows for online learning, in that it processes elements in the training set one at a time.

Biological Neuron

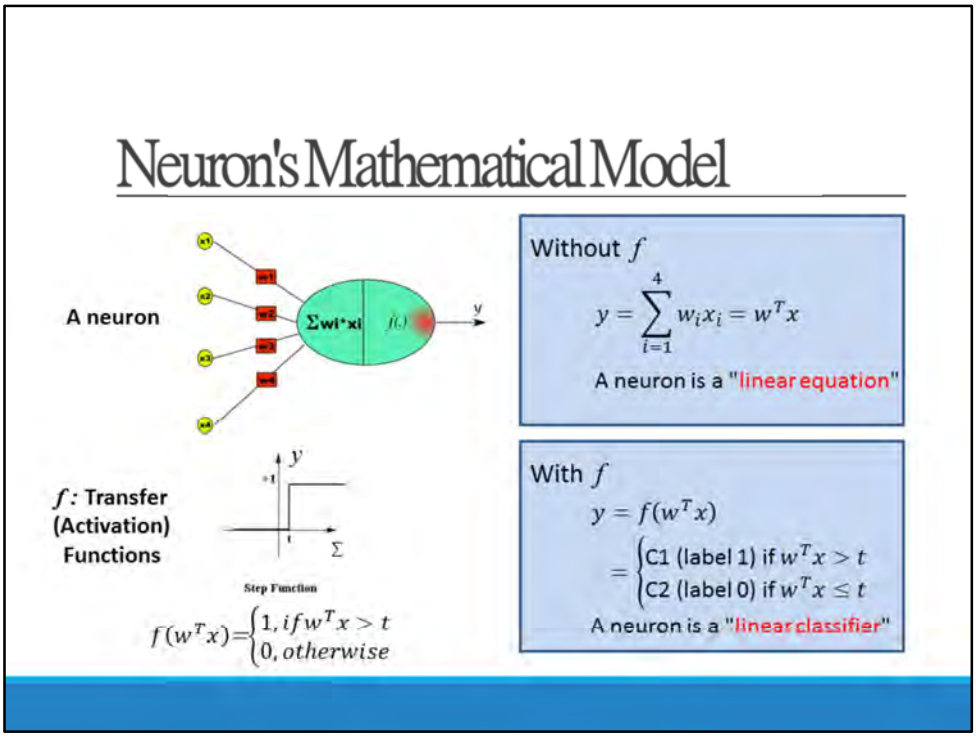




Artificial Neuron



Slide credit : Andrew L. Nelson

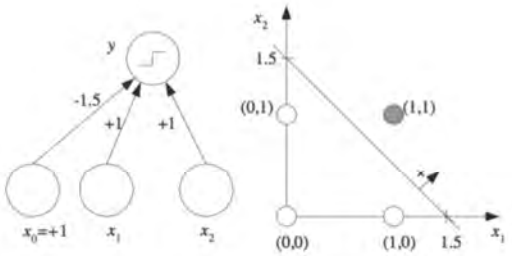


•**The Perceptron was introduced in 1957 by Frank Rosenblatt.**

"And" function Classification

x_1	x_2	r
0	0	0
0	1	0
1	0	0
1	1	1

Input and output for the AND function.



The perceptron that implements AND and its geometric interpretation.

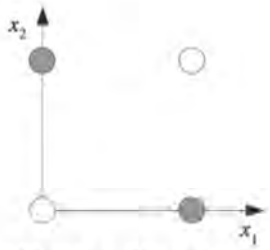
- The discriminant is $y = f(x1 + x2 - 1.5)$.

- In a Boolean function, the inputs are binary and the output is 1 if the corresponding function value is true and 0 otherwise.
- Therefore, it can be seen as a two-class classification problem.
- As an example, for learning to AND two inputs, the table of inputs and required outputs is given in the table.
- An example of a perceptron that implements AND and its geometric interpretation in two dimensions is given in figure.
- The discriminant is $y = f(x1 + x2 - 1.5)$.
- Note that $y = f(x1+x2-1.5)$ satisfies the four constraints given by the definition of AND function in table, for example, for $x1 = 1$, $x2 = 0$, $y = f(-0.5) = 0$.
- Similarly it can be shown that $y = f(x1 + x2 - 0.5)$ implements OR.

"XOR" Function Classification

x_1	x_2	r
0	0	0
0	1	1
1	0	1
1	1	0

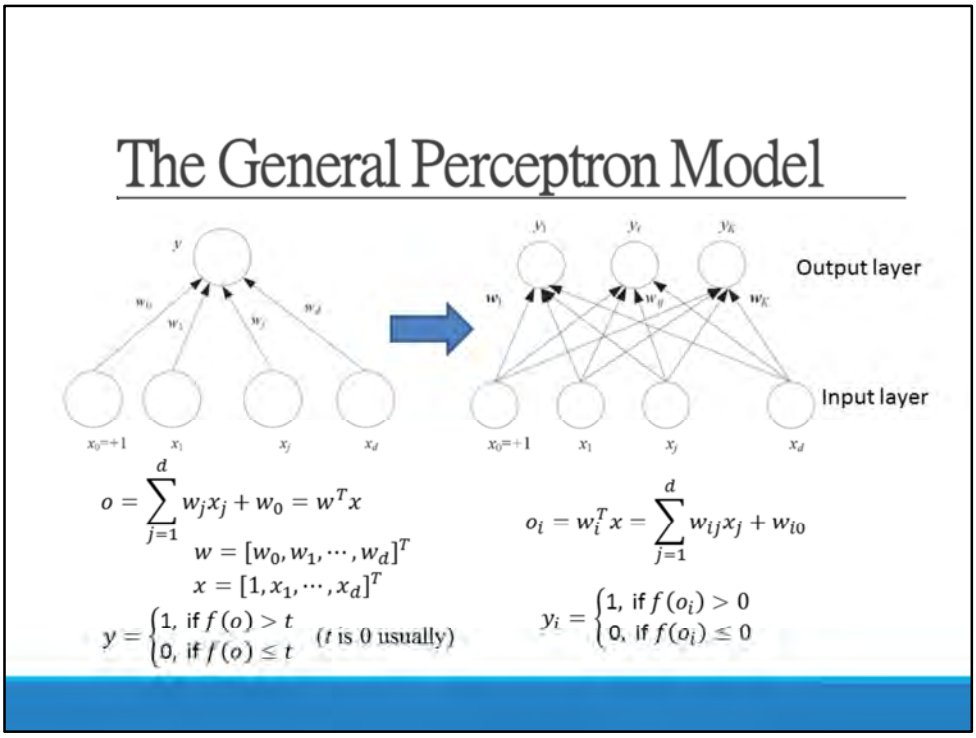
Input and output for the XOR function



XOR problem is not linearly separable. We cannot draw a line where the empty circles are on one side and the filled circles on the other side.

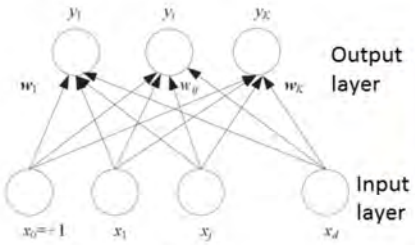
Perceptron can not solve the XOR classification problem

- Though Boolean functions like AND and OR are linearly separable and are solvable using the perceptron, certain functions like XOR are not.
- The table of inputs and required outputs for XOR is given in table.
- As can be seen in figure, the problem is not linearly separable.
- This can also be proved by noting that there are no $w0$, $w1$, and $w2$ values that satisfy the following set of inequalities:
  - $w0 \leq 0$
  - $w2 + w0 > 0$
  - $w1 + w0 > 0$
  - $w1 + w2 + w0 \leq 0$
- A perceptron that has a single layer of weights can only approximate linear functions of the input and cannot solve problems like the XOR, where the discrimininant to be estimated is nonlinear.



- Left:
  - Simple perceptron. x_j , $j = 1, \ldots, d$ are the input units. x_0 is the bias unit that always has the value 1. y is the output unit. w_j is the weight of the directed connection from input x_j to the output.
- Right:
  - K parallel perceptrons. $x_j, j = 0,\ldots,d$ are the inputs and $y_i, i = 1, \ldots, K$ are the outputs. w_{ij} is the weight of the connection from input x_j to output y_i. Each output is a weighted sum of the inputs. When used for K-class classification problem, there is a postprocessing to choose the maximum, or softmax if we need the posterior probabilities.

Perceptron Model for Classification



For a K-class classification problem

- For a given x with unknown class
- $x \in$ class k, $if\ y_k = 1, y_j = 0\ for\ all\ j \neq k$
- Only one y_k can be 1

$$o_i = w_i^T x = \sum_{j=1}^{d} w_{ij} x_j + w_{i0}$$

$$y_i = \begin{cases} 1, \text{ if } f(o_i) > t_i \\ 0, \text{ if } f(o_i) \leq t_i \end{cases}$$

A simplified K-class classification problem

- $x \in$ class k, $if\ y_k = max_i y_i$
- $y_i = w_i^T x = \sum_{j=1}^{d} w_{ij} x_j + w_{i0}$

- During testing, with given weights, $\boldsymbol{w}$, for input $\boldsymbol{x}$, we compute the output y.
- To implement a given task, we need to *learn* the weights $\boldsymbol{w}$, the parameters of the system, such that correct outputs are generated given the inputs.

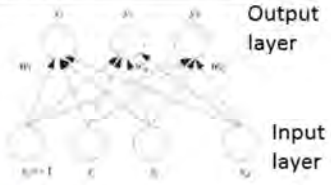
Training the perceptron

Testing for K-class classification problem

- For a given x with unknown class
- $x \in$ class k, $if\ y_k = max_i y_i$
- $y_i = w_i^T x = \sum_{j=1}^{d} w_{ij} x_j + w_{i0}$

That is

- A W represents a perceptron
- Given a W, then we can classify a pattern x



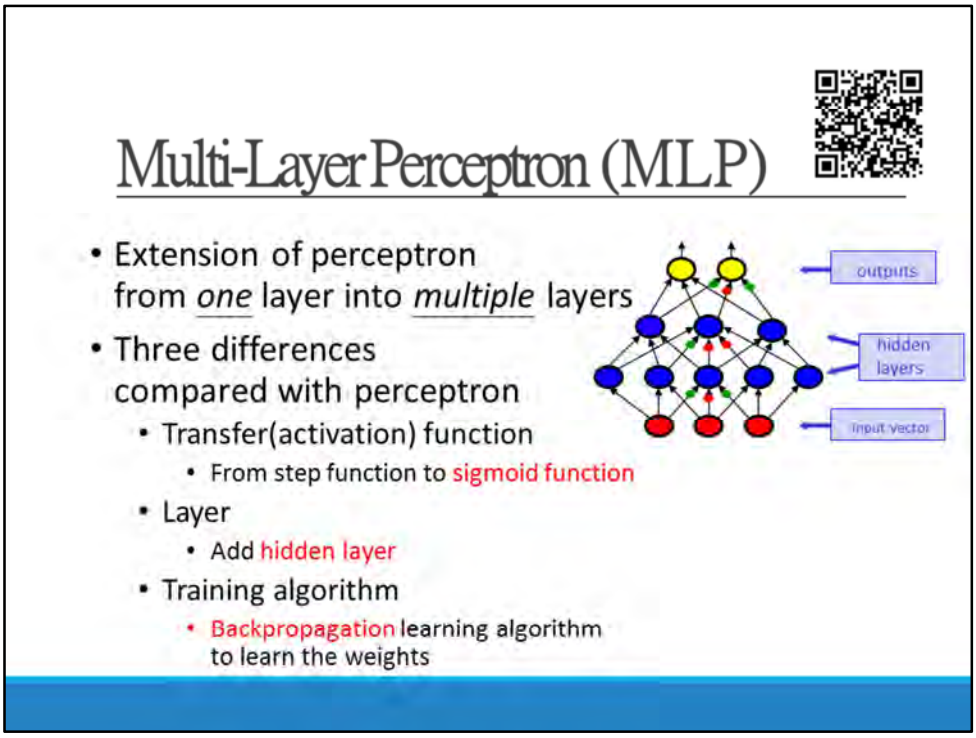
$$W = [w_1, w_2, \cdots, w_K] = \begin{bmatrix} w_{11} & \cdots & w_{K1} \\ \vdots & \ddots & \vdots \\ w_{1d} & \cdots & w_{Kd} \end{bmatrix}$$

A Machine Learning problem:
how to obtain the W of a perceptron ?

- We need a set of training patterns (X, Y)
- We need a learning algorithm A to learn W by (X, Y)
  => Perceptron learning algorithm A: $W=A(X, Y)$

Neural Information Processing

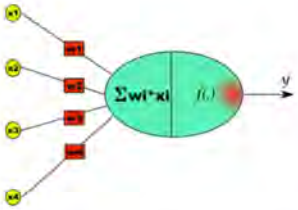
- Origins
- Perceptron
- **Multilayer perceptron (MLP)**
- Convolutional neural network



- A perceptron that has a single layer of weights can only approximate linear functions of the input and cannot solve problems like the XOR, where the discrimininant to be estimated is nonlinear.
- This limitation does not apply to feedforward networks with intermediate or *hidden layers* between the input and the output layers. If used for classification, such *multilayer perceptrons* (MLP) can implement nonlinear discriminants and, if used for regression, can approximate nonlinear functions of the input.
- A multilayer perceptron (MLP) is a feedforward artificial neural network model that maps sets of input data onto a set of appropriate outputs. An MLP consists of multiple layers of nodes in a directed graph, with each layer fully connected to the next one. Except for the input nodes, each node is a neuron (or processing element) with a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training the network.[1][2] MLP is a modification of the standard linear perceptron and can distinguish data that are not linearly separable.

Neuron with Sigmoid Function

Neuron:

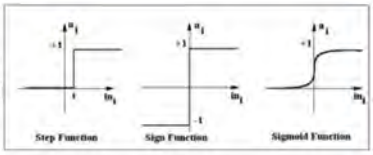


f is a *step* function

$$y = f(w^T x) = \begin{cases} \text{C1 (label 1) if } w^T x > t \\ \text{C2 (label 0) if } w^T x \le t \end{cases}$$

A neuron is a "linear classifier"

f: Transfer (Activation) Functions

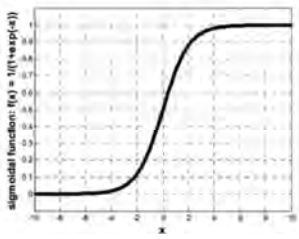


f is a *sigmoid* function

$$y = f(w^T x) = \begin{cases} \text{C1 if } f(w^T x) > 0.5 \\ \text{C2 if } f(w^T x) \le 0.5 \end{cases}$$

A neuron is still a "linear classifier"

Transfer functions : Sigmoid

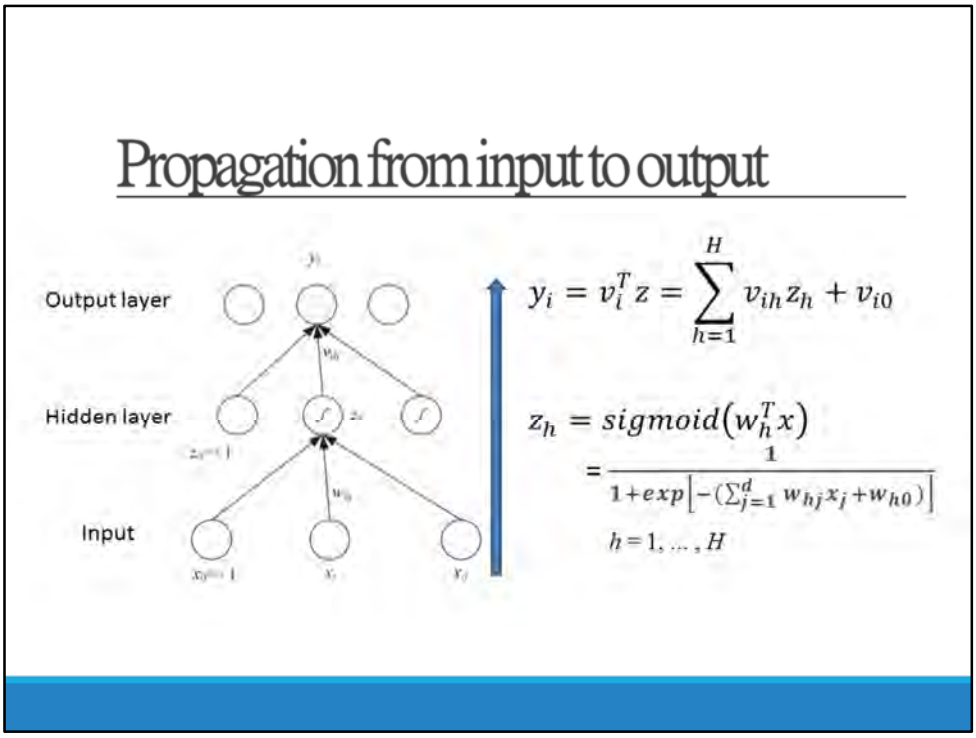


$$y = f(x) = \frac{1}{1+e^{-x}}$$

$$y = f(w^T x) = \frac{1}{1+e^{-w^T x}}$$

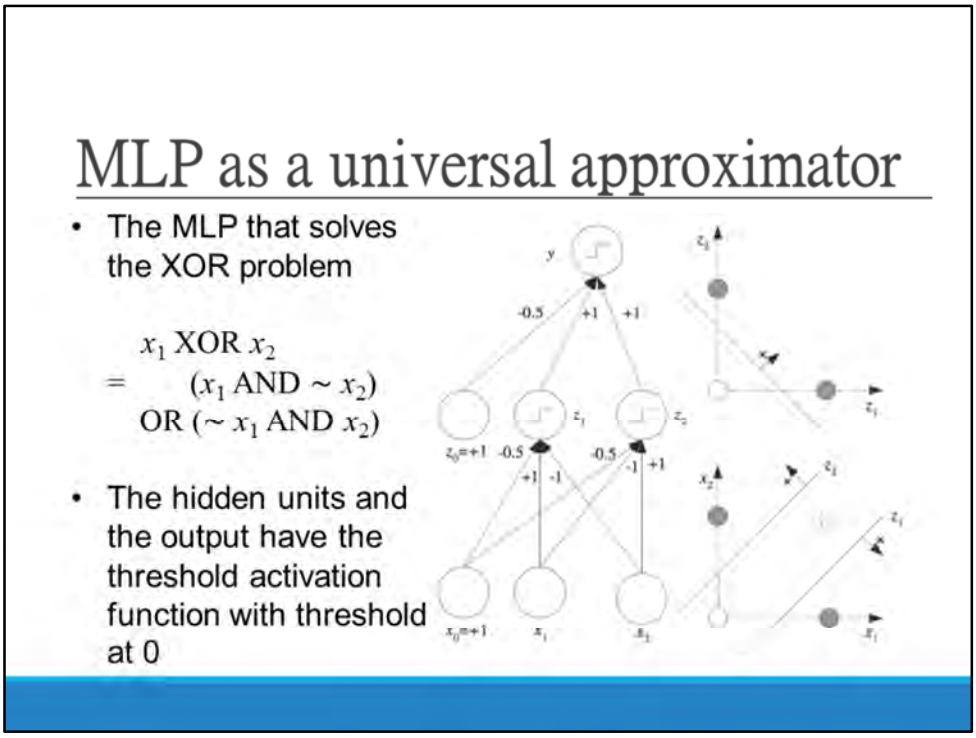
Sigmoid function

Activation function

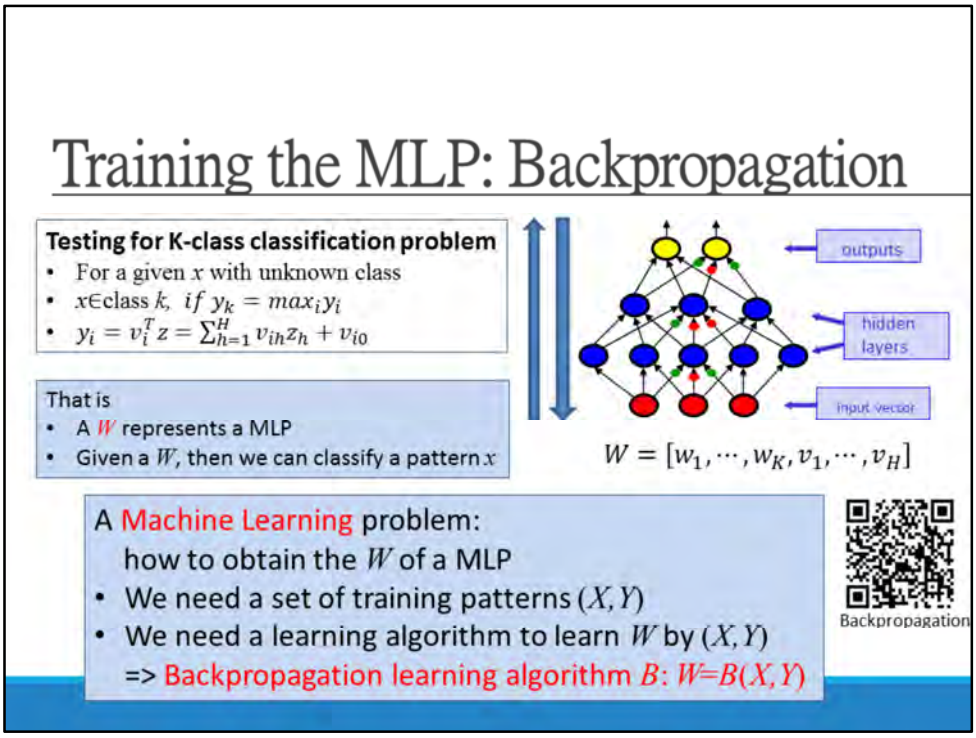


- The structure of a multilayer perceptron.
  - *xj , j* = 0*, . . . , d* are the inputs,
  - *zh , h* = 1*, . . . , H* are the hidden units where *H* is the dimensionality of this hidden space. *z*0 is the bias of the hidden layer.
  - *yi,i* = 1,...,*K* are the output units. *whj* are weights in the first layer, and *vih* are the weights in the second layer.
- Input ***x*** is fed to the input layer (including the bias), the "activation" propagates in the forward direction, and the values of the hidden units *zh* are calculated (see figure 11.6). Each hidden unit is a perceptron by itself and applies the nonlinear sigmoid function to its weighted sum.
- The output *yi* are perceptrons in the second layer taking the hidden units as their inputs.
- The output is a linear combination of the nonlinear basis function val- ues computed by the hidden units. It can be said that the hidden units make a nonlinear transformation from the *d*-dimensional input space to the *H*-dimensional space spanned by the hidden units, and, in this space, the second output layer implements a linear function.
- Note
  - This is a two-layer network, not a three-layer network.

- The input layer of *xj* is not counted as a layer since no computation is done there and when there is a hidden layer.



- We can represent any Boolean function as a disjunction of conjunctions, and such a Boolean expression can be implemented by a multilayer per- ceptron with one hidden layer. Each conjunction is implemented by one hidden unit and the disjunction by the output unit.
    - For example, *x*1 XOR*x*2 =*(x*1 AND ~*x*2*)*OR*(*~*x*1 AND*x*2*)*
- We have seen previously how to implement AND and OR using perceptrons.
- So two perceptrons can in parallel implement the two AND, and another perceptron on top can OR them together (see the figure).
    - We see that the first layer maps inputs from the *(x*1*, x*2*)* to the *(z*1*, z*2*)* space defined by the first-layer perceptrons.
    - Note that both inputs, (0,0) and (1,1), are mapped to (0,0) in the *(z*1*,z*2*)* space, allowing linear separability in this second space.
- Thus in the binary case, for every input combination where the output is 1, we define a hidden unit that checks for that particular conjunction of the input. The output layer then implements the disjunction.
- More detailed description of universal approximator by MLP: please see Section 11.6 "MLP as a Universal Approximator" in the book: **Introduction to Machine Learning, 2nd, E. Alpaydin, MIT Press, 2010. pp. 248.**



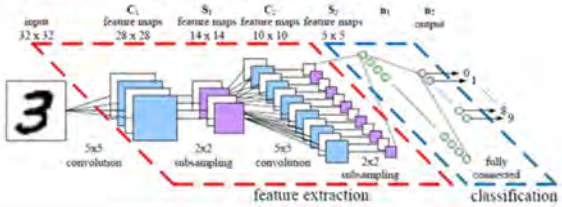
- **Propagation**: from input layer to output layer
  - For testing of classification
- **Back-propagation**: from output layer to input layer
  - For learning of classification

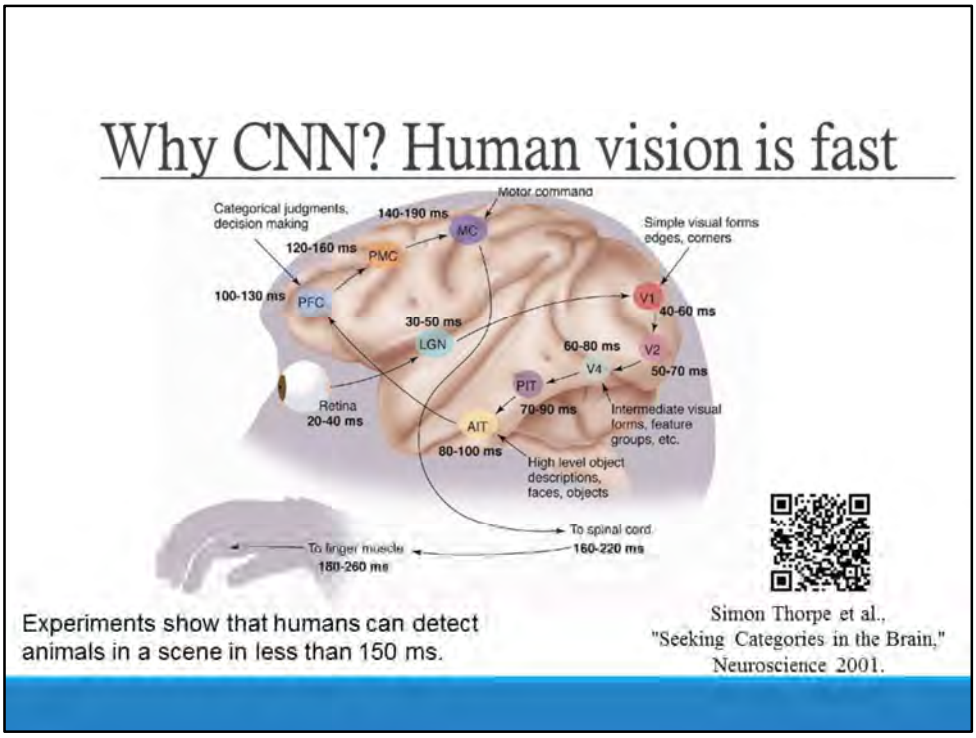
Neural Information Processing

- Origins
- Perceptron
- Multilayer perceptron network
- **Convolutional neural networks (CNN)**

What Is CNN

- An extension of MLP
  - *More hidden layers* to extract *features* of signal (image, speech)
  - Depth of network is "deep"
- CNN is one famous model of **deep neural networks (DNN)**





- Simon J. Thorpe and Michale Fabre-Thorpe, "Seeking Categories in the Brain," NEUROSCIENCE, Vol. 291, No. 5502, Issue of 12 Jan 2001, pp. 260-263. http://monkeylogic.uchicago.edu/old/Science.htm.
- The model is characterized by its hierarchical and feedforward organization (**Figure 1**). Neurons in lower visual areas, with small receptive fields, are sensitive to basic visual features.
- For example, neurons in V1 respond predominantly to edges and lines. These neurons project to neurons at the next stage of the hierarchy, which code for more complex features. By V4, the neurons are selective for basic shapes, and by IT they respond in a viewpoint-invariant manner to full objects. Decisions making happens in the frontal cortex.
- This basic scenario has a well-defined set of char- acteristics. Processing is hierarchical, feedforward, and local on each level, i.e., only neighboring neurons, coding for neighboring parts in the visual field, project to a common higher-level neuron
- Proces ample, a hypo 'lower" neuror

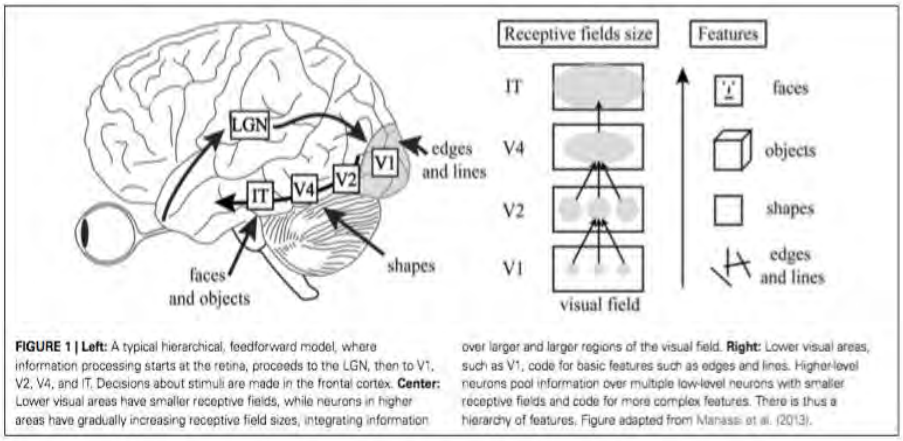


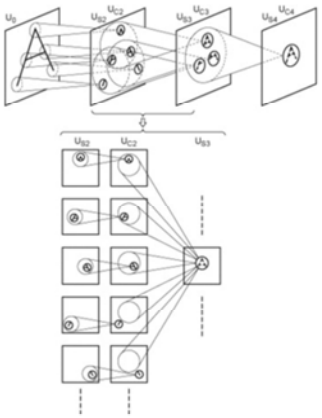
FIGURE 1 | Left: A typical hierarchical, feedforward model, where information processing starts at the retina, proceeds to the LGN, then to V1, V2, V4, and IT. Decisions about stimuli are made in the frontal cortex. **Center:** Lower visual areas have smaller receptive fields, while neurons in higher areas have gradually increasing receptive field sizes, integrating information over larger and larger regions of the visual field. **Right:** Lower visual areas, such as V1, code for basic features such as edges and lines. Higher-level neurons pool information over multiple low-level neurons with smaller receptive fields and code for more complex features. There is thus a hierarchy of features. Figure adapted from Manassi et al. (2013).

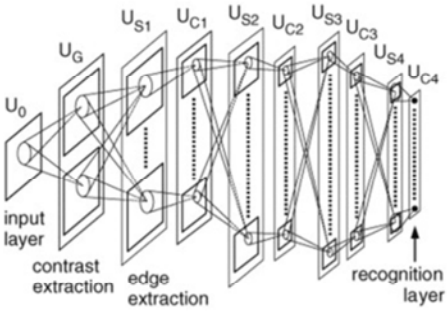
Hubel & Wiesel (1962)

Insights about early image processing in the brain

- Simple cells detect local features
- Complex cells pool local features in a retinotopic neighborhood

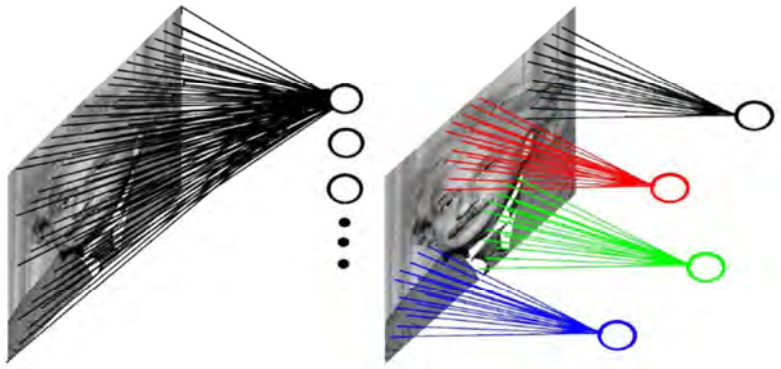
The Neocognitron



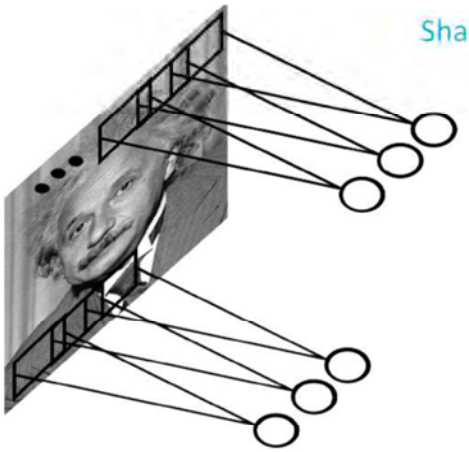


(Fukushima 1974-1982)

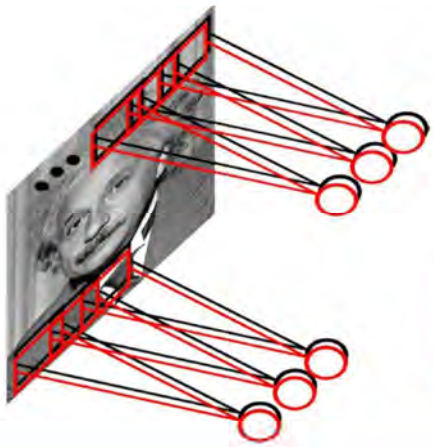
Local connections



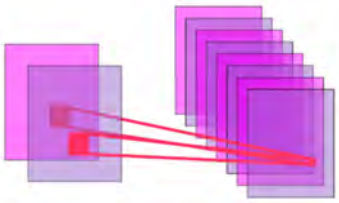
Convolution



Multiple convolutions

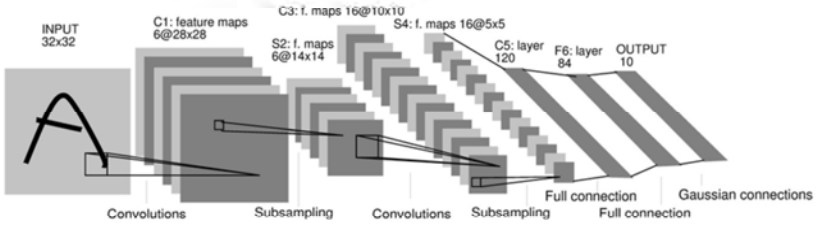


CNN:
Convolutional neural network
ConvNet

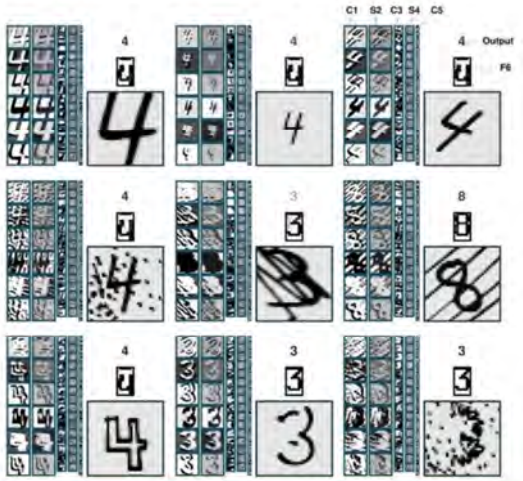


CNNs in the 1990s

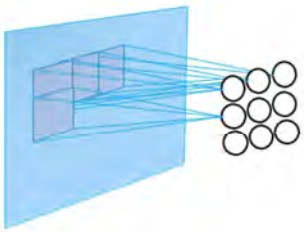
- 1989 Isolated handwritten character recognition (AT&T Bell Labs)
- 1991 Face recognition. Sonar image analysis. (Neuristique)
- 1993 Vehicle recognition. (Onera)
- 1994 Zip code recognition (AT&T Bell Labs)
- 1996 Check reading (AT&T Bell Labs)



Convnets in the 1990s



Pooling



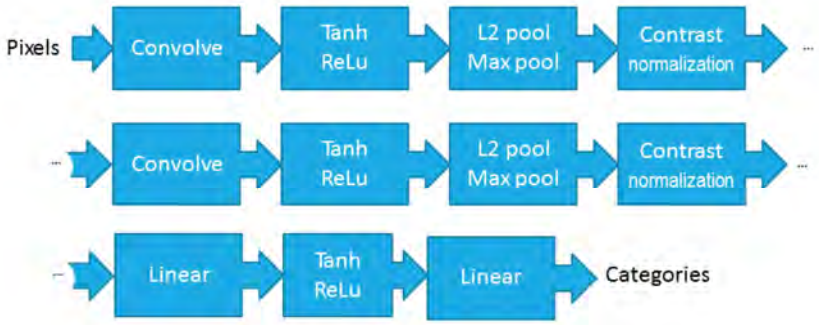
Name	Pooling formula
Average pool	$\frac{1}{s^2}\sum x_i$
Max pool	$\max\{x_i\}$
L2 pool	$\sqrt{\frac{1}{s^2}\sum x_i^2}$
L_p pool	$\left(\frac{1}{s^2}\sum \lvert x_i\rvert^p\right)^{\frac{1}{p}}$

Contrast Normalization

Contrast normalization

- Subtracting a low-pass smoothed version of the layer
- Just another convolution in fact (with fixed coefficients)
- Lots of variants (per feature map, across feature maps, ...)
- Divisive normalization

CNNs in the 2010s

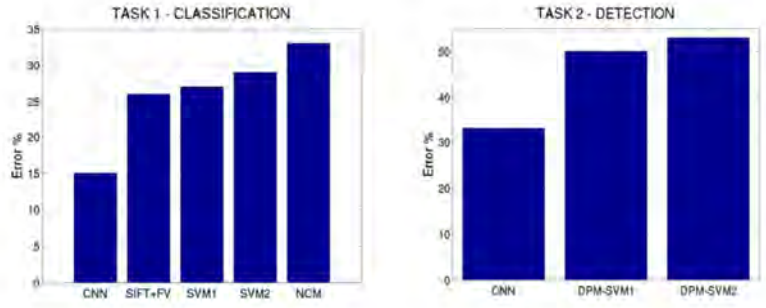


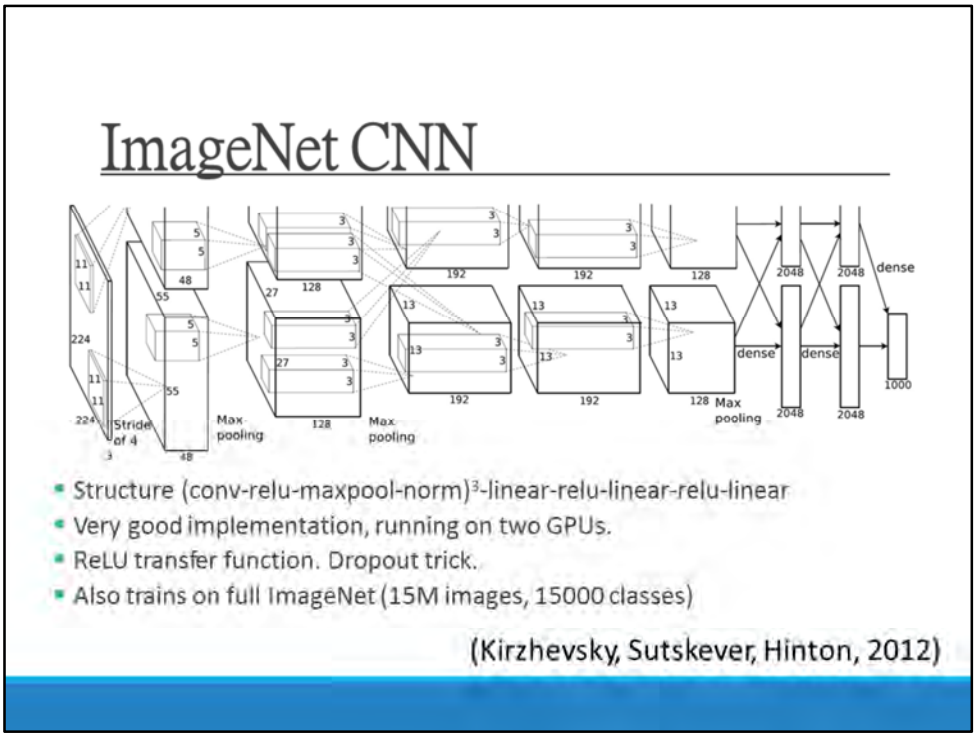
Convnets in the 2000s

- OCR in natural images [2011]. Streetview house numbers (NYU)
- Traffic sign recognition [2011]. GTRSB competition (IDSIA, NYU)
- Pedestrian detection [2013]. INRIA datasets (NYU)
- Volumetric brain segmentation [2009]. Connectomics (MIT)
- Human action recognition [2002,2011]. Smartcatch (NEC), Hollywood II (SF)
- Object recognition [2004,2012]. Norb (NEC), ImageNet (UofT)
- Scene parsing [2010-2012]. Stanford bldg, Barcelona (NEC, NYU)
- Medical image analysis [2008]. Cancer detection (NEC)

ImageNet 2012 competition

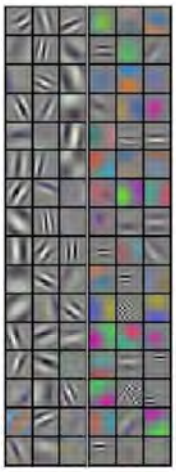
- Object recognition. 1000 categories. 1.2M examples



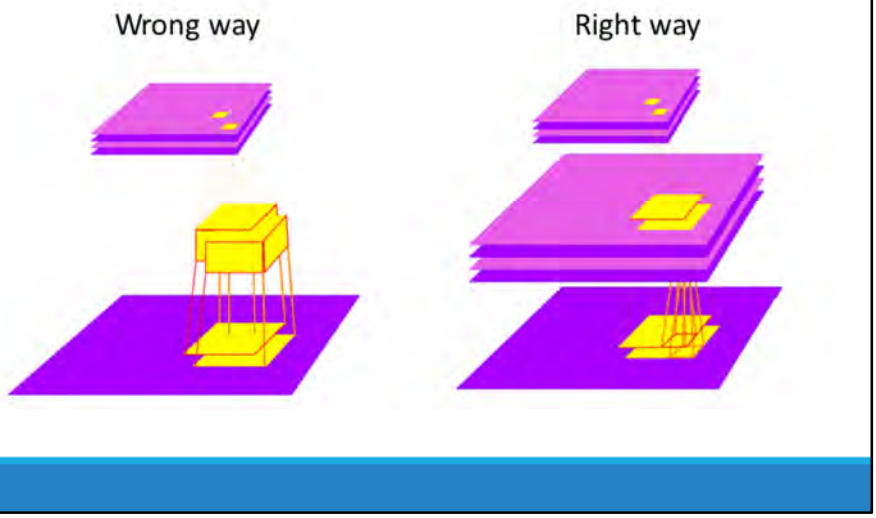


- A. Krizhevsky, I. Sutskever, G.E. Hinton, "Imagenet classification with deep convolutional neural networks."
  Advances in neural information processing systems, pp. 1097-1105, 2012.

ImageNet CNN



Replicated CNNs



Today :
Neural Information Processing

- Origins
- Perceptron
- Multilayer perceptron (MLP)
- Convolutional neural network (CNN)

Next :

Training multilayer networks

- Perceptron learning
- Optimization basics
- Stochastic gradient descent
- Backpropagation learning algorithms