Fakulta matematiky, fyziky a informatiky

Univerzita Komenského v Bratislave

Image Classification using Artificial Neural Networks

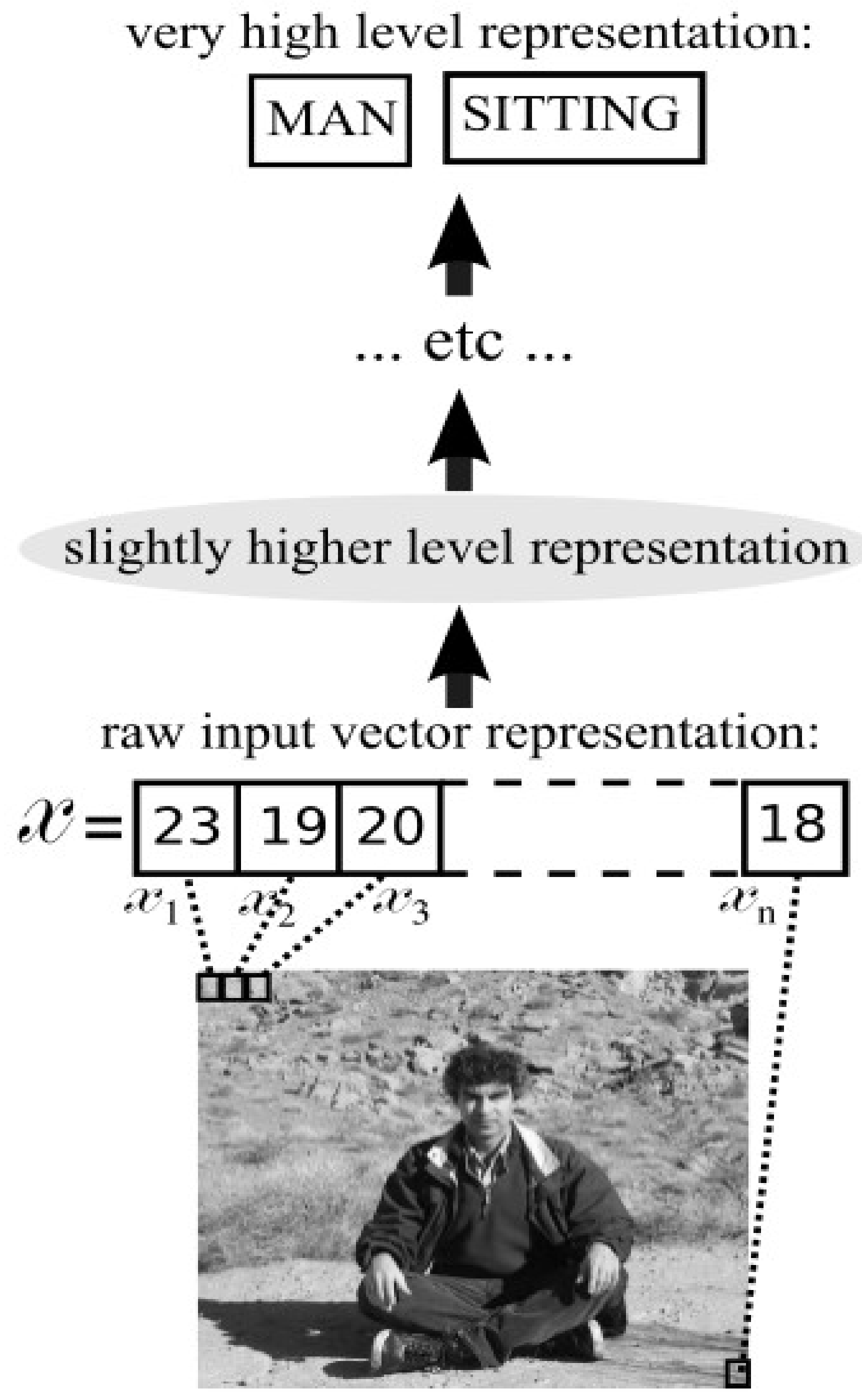
Igor Farkaš

2012

Deep architectures

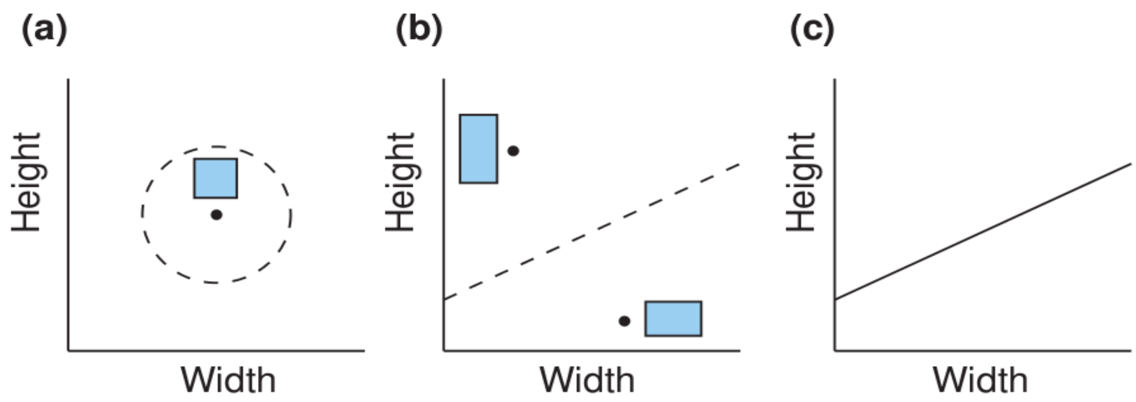
- How to recognize complex objects from raw data?
- Problem of variability (position, rotation, size)

- Deep architectures important:
  - in artificial intelligence
  - in biological systems
  - allow to make a cascade of nonlinear transformations → deep learning



Methods using artificial neural networks

- brain-inspired
- basic building blocks (computing elements) – artificial neurons:
  - deterministic (perceptron, RBF) → discriminatory models (c)
  - stochastic (probabilistic) → generative models (a,b)



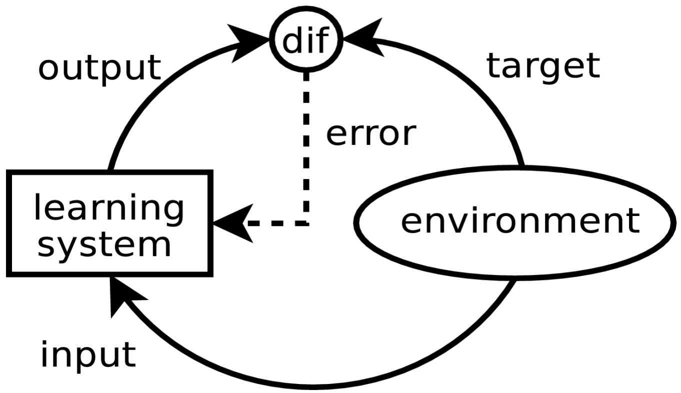
- multi-layered feedforward architectures
- model parameters are learned using training data
- model performance evaluated on testing data (generalization)

Brief history of connectionism

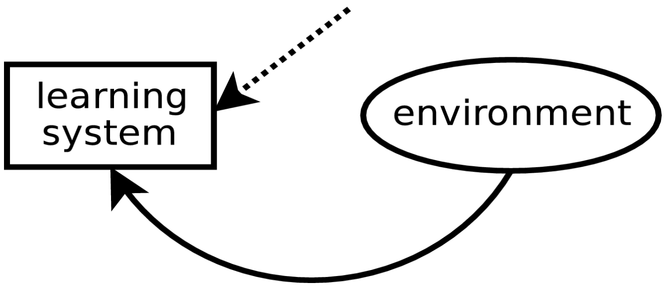
- classical connectionism (until 1940s)
  - within philosophy, psychology
- old connectionism (1950s-1970s) – birth of computer era
  - beginning of theory of artificial neural networks
  - linked to cognitive science revolution
- new connectionism (from 1986)
  - parallel distributed processing $\rightarrow$ subsymbolic processing
  - multi-layer NN models (incl. recurrent)
- even newer connectionism (late 1990s)
  - multilayer generative models (probabilistic approach)

Learning paradigms in NN

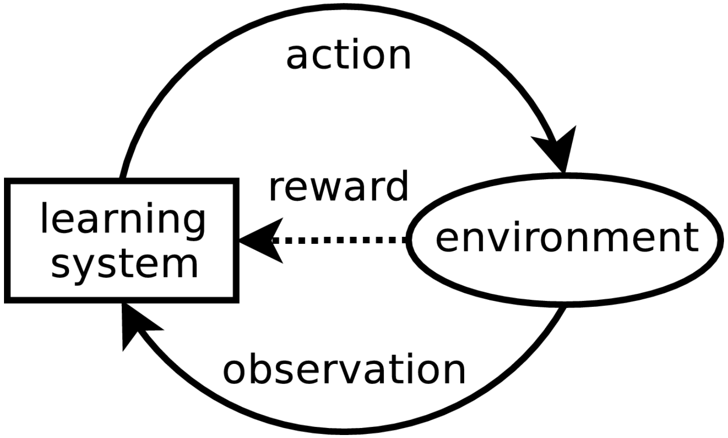
supervised (with teacher)



unsupervised (self-organized)



reinforcement learning (partial feedback)

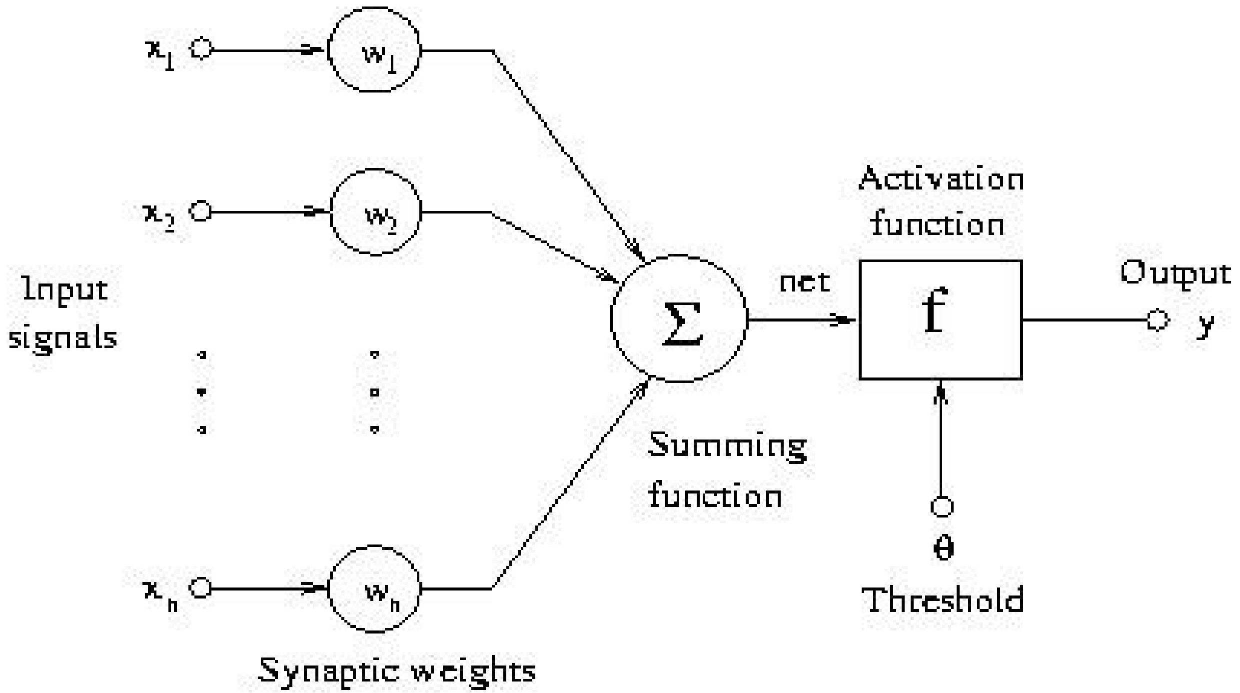


Typical artificial neuron model

1. receives signals from other neurons (or sensors)
2. processes (integrates) incoming signals
3. sends the processed signal to other neurons (or muscles)

Deterministic model

$$y = f\left(\sum_i w_i x_i - \theta\right)$$



Stochastic model

$$P(s=+1) = 1/(1+\exp(-\sum_i w_i x_i + \theta))$$

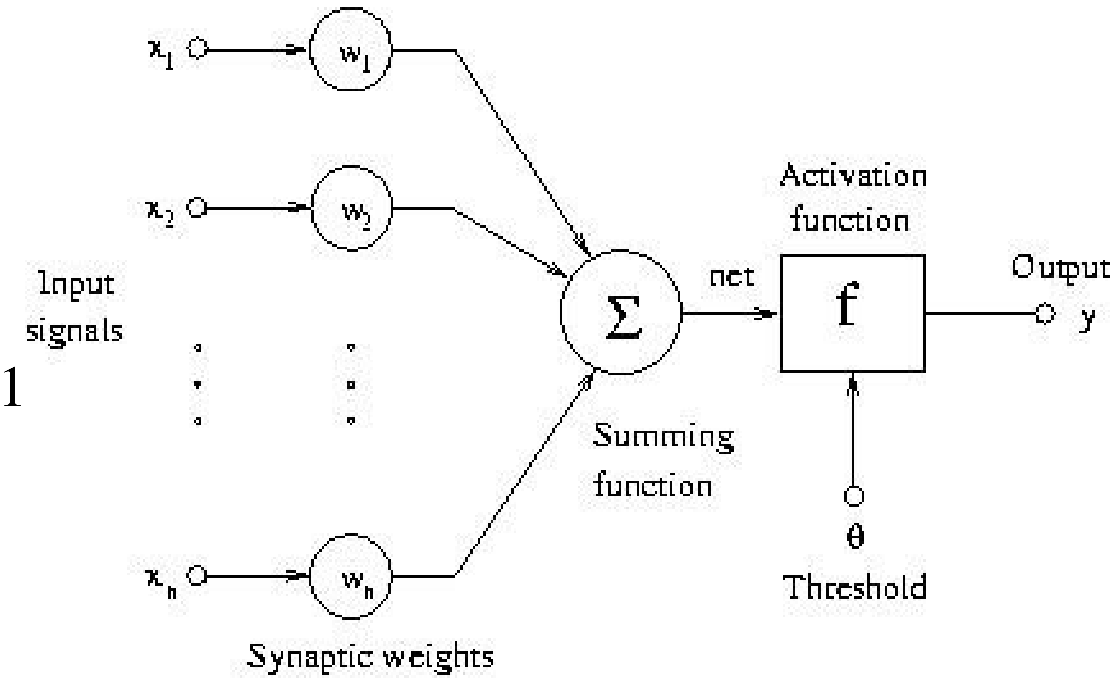
Discrete perceptron

(Rosenblatt, 1962)

- Inputs ***x*** , weights ***w***, output *y*
- Activation:

$$y = f(\textstyle\sum^{n}_{j=1} w_j x_j - \theta)$$

$$y = f\left(\sum_{j=1}^{n+1} w_j x_j\right) \quad x_{n+1} = -1$$



- *f* = threshold function: unipolar {0,1} or bipolar {-1,+1}
- Supervised learning – uses teacher signal *d*
- Learning rule:

$$w_j(t+1) = w_j(t) + \alpha\,(d - y)\,x_j$$

F. Rosenblatt (1962). *Principles of Neurodynamics*, Spartan, New York, NY.

Summary of perceptron algorithm

Given: training data: input-target $\{\boldsymbol{x}, d\}$ pairs, unipolar perceptron
Initialization: randomize weights, set learning rate

Training:

1. choose input $\boldsymbol{x}$, compute output y, set $E = 0$
2. evaluate error function $e(t) = ½ (d - y)^2$, $E \leftarrow E + e(t)$
3. adjust weights using delta rule (if $e(t) > 0$)
4. if all patterns used, then goto 5, else go to 1
5. if $E = 0$ (all patterns in the set classified correctly), then end

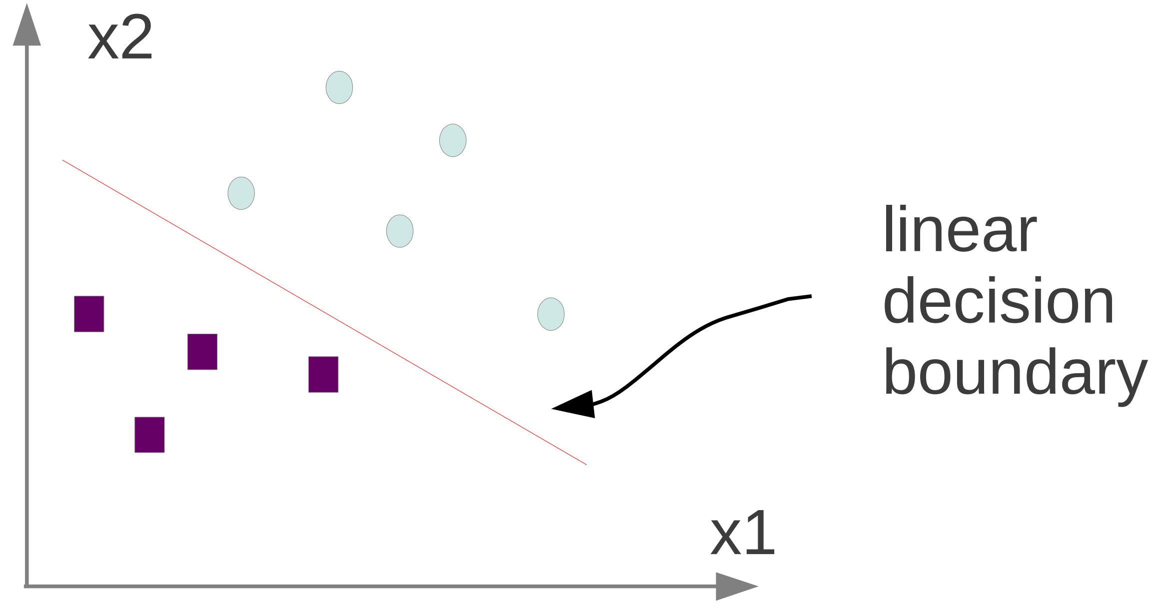
   else reorder inputs, $E \neg 0$, go to 1

Perceptron classification capacity

$$w_1x_1 + w_2x_2 + \ldots + w_nx_n = \theta$$

linear separability of two classes

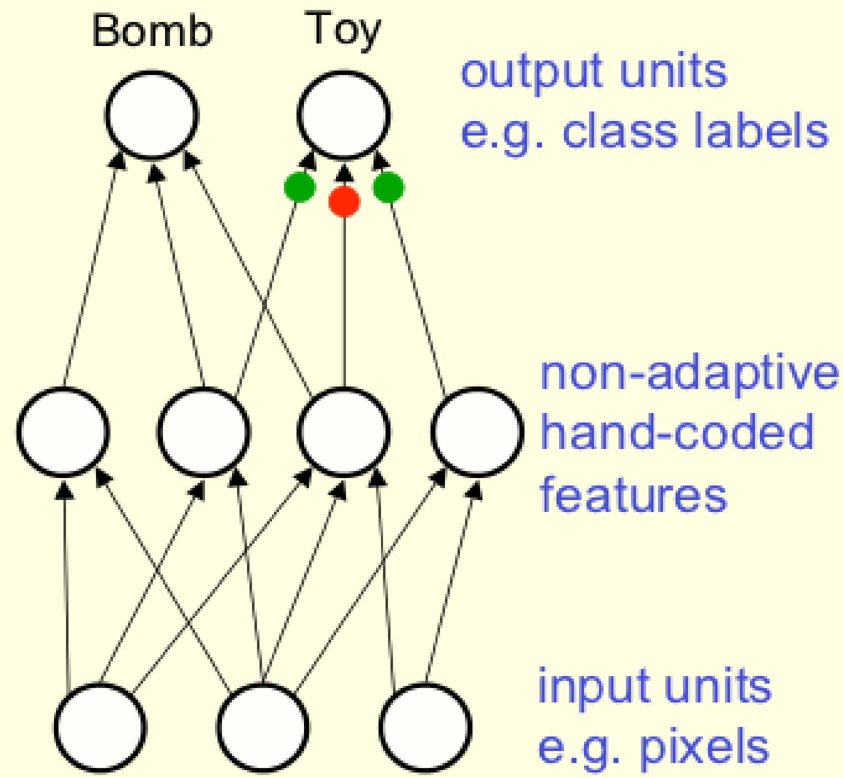
2D example



Fixed-increment convergence theorem (Rosenblatt, 1962): *"Let the classes A and B are finite and linearly separable, then perceptron learning algorithm converges (updates its weight vector) in a finite number of steps."*

Historical background:
First generation neural networks

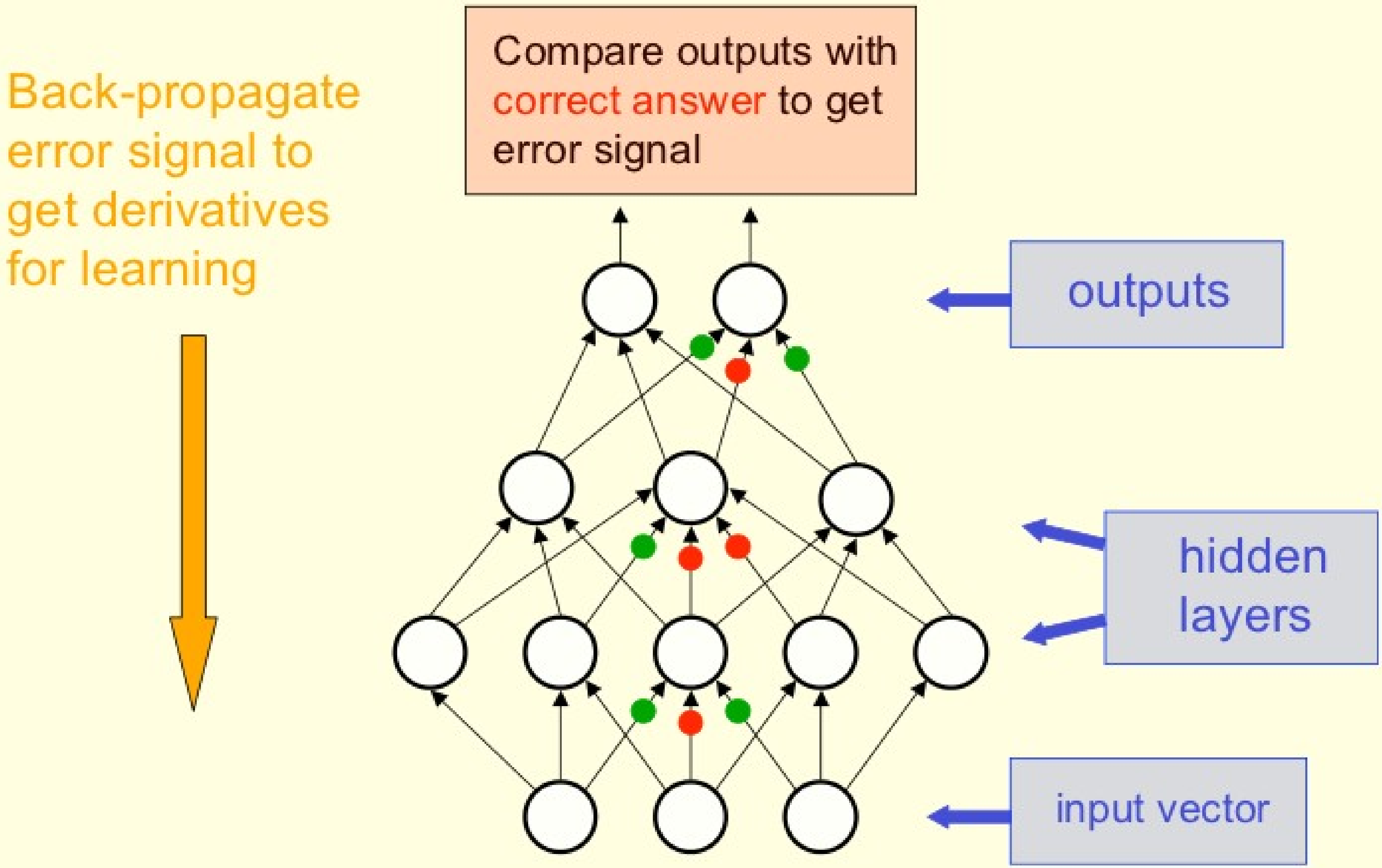
- Perceptrons (~1960) used a layer of hand-coded features and tried to recognize objects by learning how to weight these features.
  – There was a neat learning algorithm for adjusting the weights.
  – But perceptrons are fundamentally limited in what they can learn to do.



Sketch of a typical perceptron from the 1960's

(Hinton, 2007)

Second generation neural networks (~1985)



(Hinton, 2007)

Two-layer perceptron

- Inputs $\boldsymbol{x}$, weights $\boldsymbol{w}$, $\boldsymbol{v}$, outputs $\boldsymbol{y}$
- Nonlinear activation function f
- Unit activation:

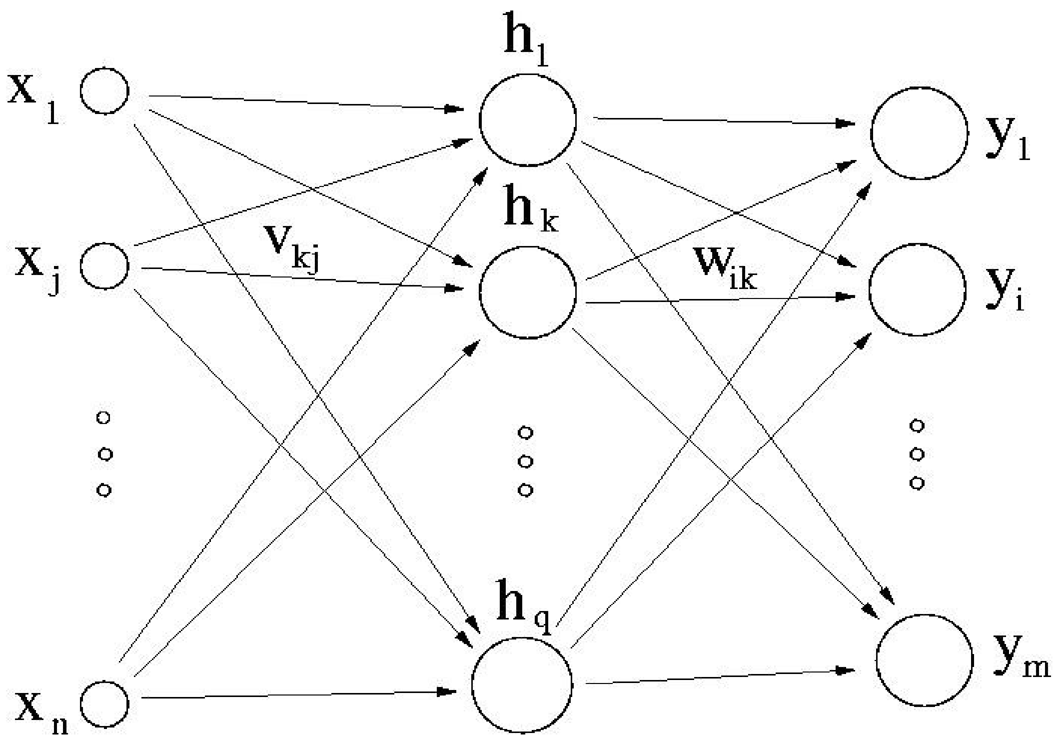
$$h_k = f\left(\sum_{j=1}^{n+1} v_{kj} x_j\right)$$

$$y_i = f\left(\sum_{k=1}^{q+1} w_{ik} h_k\right)$$

- Bias input: $x_{n+1} = h_{q+1} = -1$
- Activation function examples:

$$f(net) = 1 / (1 + \exp(-net))$$

$$f(net) = \tanh(net)$$



Learning equations for original BP

Hidden-output weights:

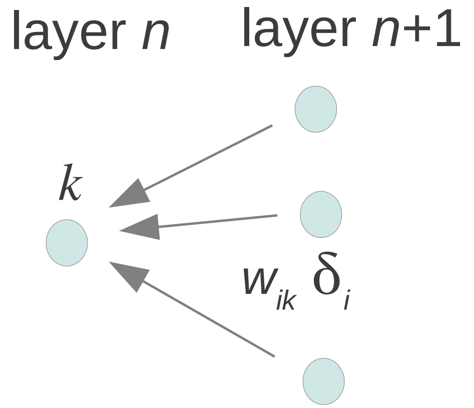
$$w_{ik}(t+1) = w_{ik}(t) + \alpha\delta_i h_k \quad \text{where} \quad \delta_i = (d_i - y_i)\, f_i'$$

Input-hidden weights:

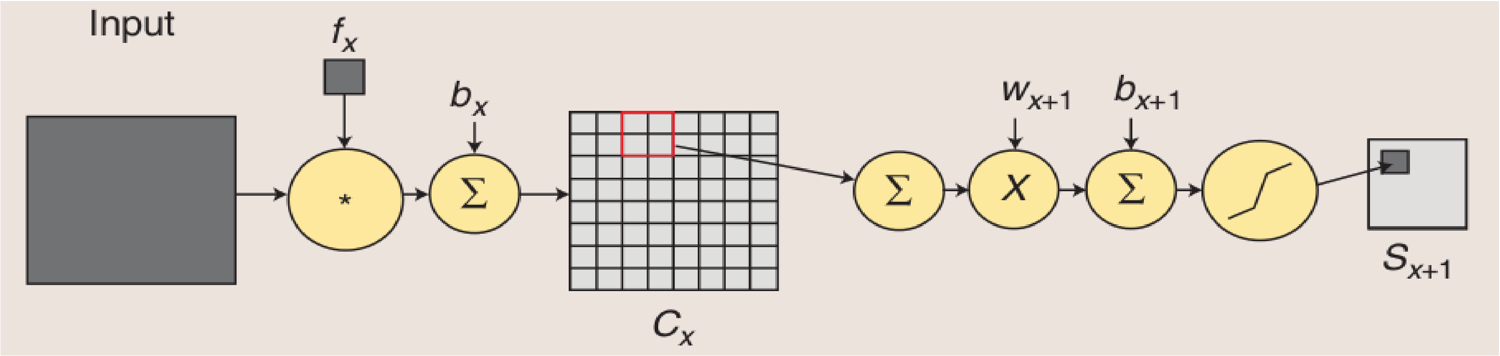
$$v_{kj}(t+1) = v_{kj}(t) + \alpha\, \delta_k x_j \quad \text{where} \quad \delta_k = (\Sigma_i w_{ik}\, \delta_i)\, f_k'$$

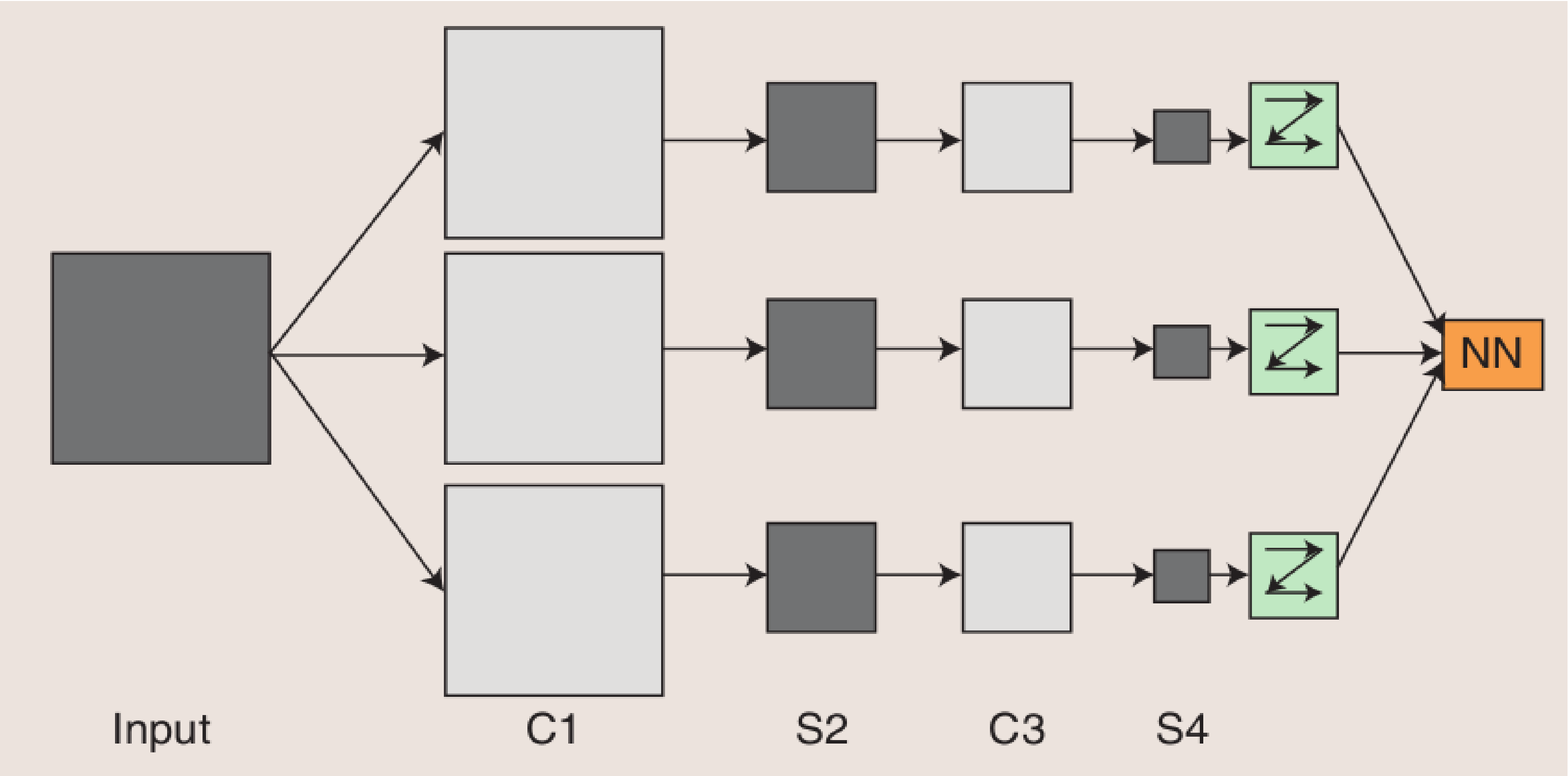
BP provides an “approximation” to the trajectory in weight space computed by the method of steepest descent.

- smoothness of the trajectory depends on α



Convolutional Neural Networks



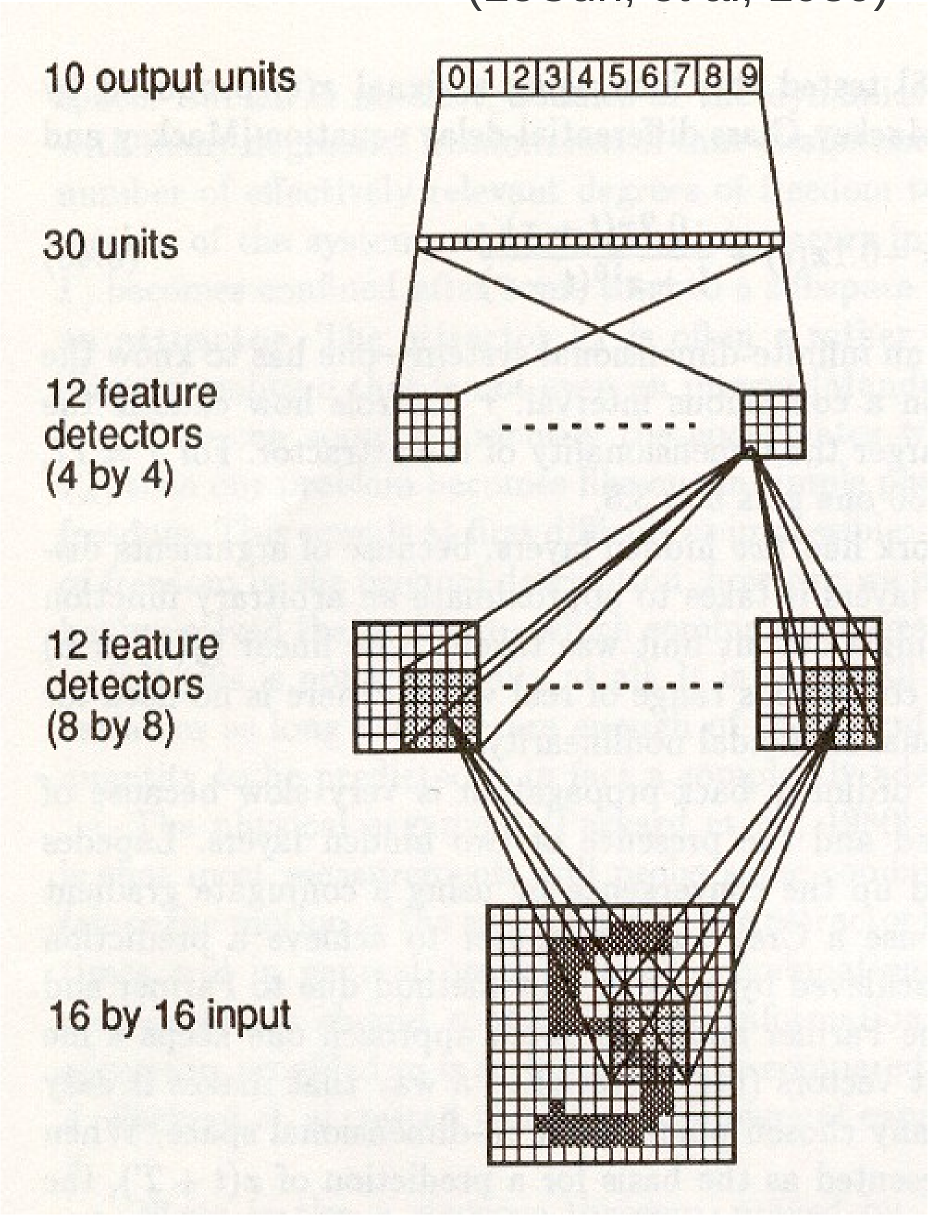


(Arel et al, 2010)

Application: Recognizing hand-written ZIP codes

- Input: 16×16 units, (-1,1) range
- 3 hidden layers (HL)
- Reduction of free parameters by weight sharing on HL1: all 64 units in a group had the same 25 weights
- the same principle used in HL2
- 1256 units and 9760 weights
- Error back-propagation learning used, accelerated with quasi-Newton rule
- 1% error on train set (7,300 digits), 5% on test set (2,000 digits).
- “optimal brain damage” - further elimination of weights to reduce test error

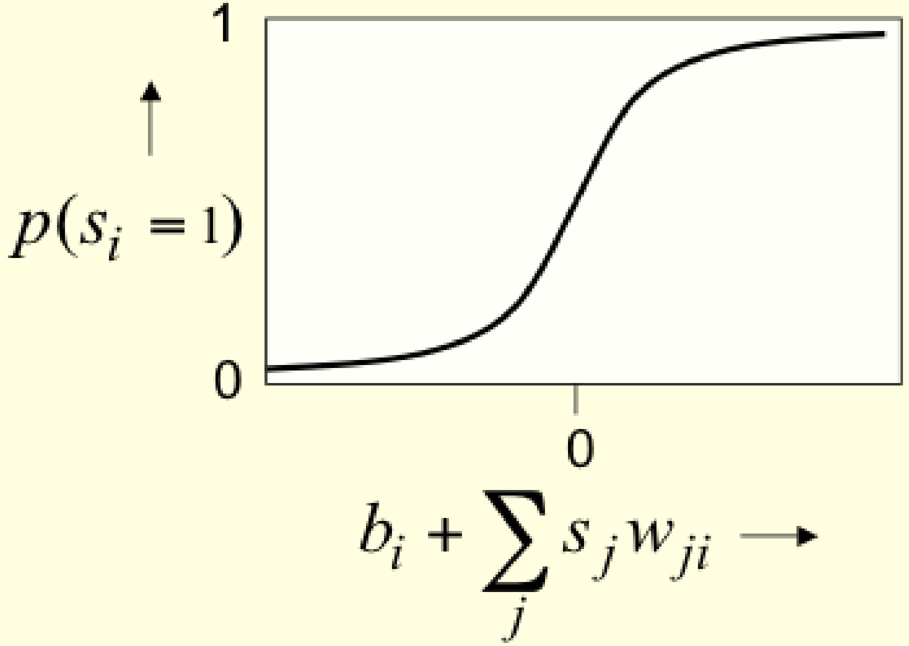
(LeCun, et al, 1989)



Stochastic binary units
(Bernoulli variables)

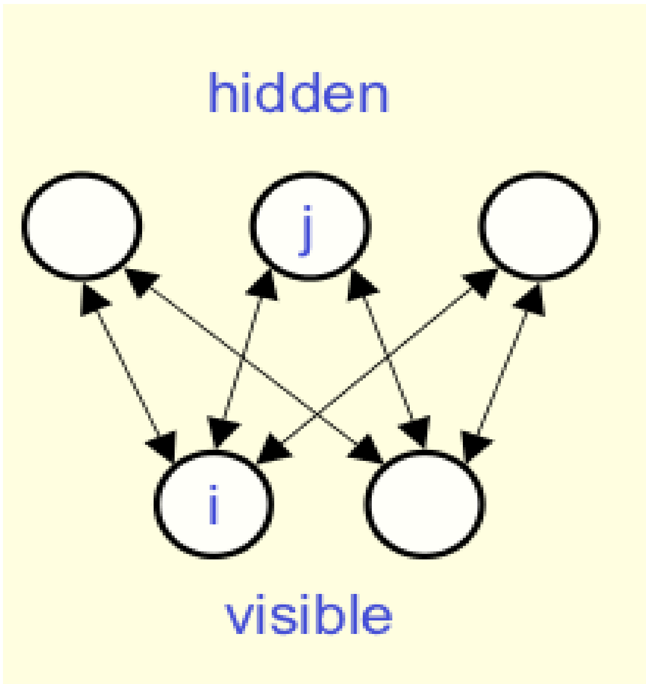
- These have a state of 1 or 0.

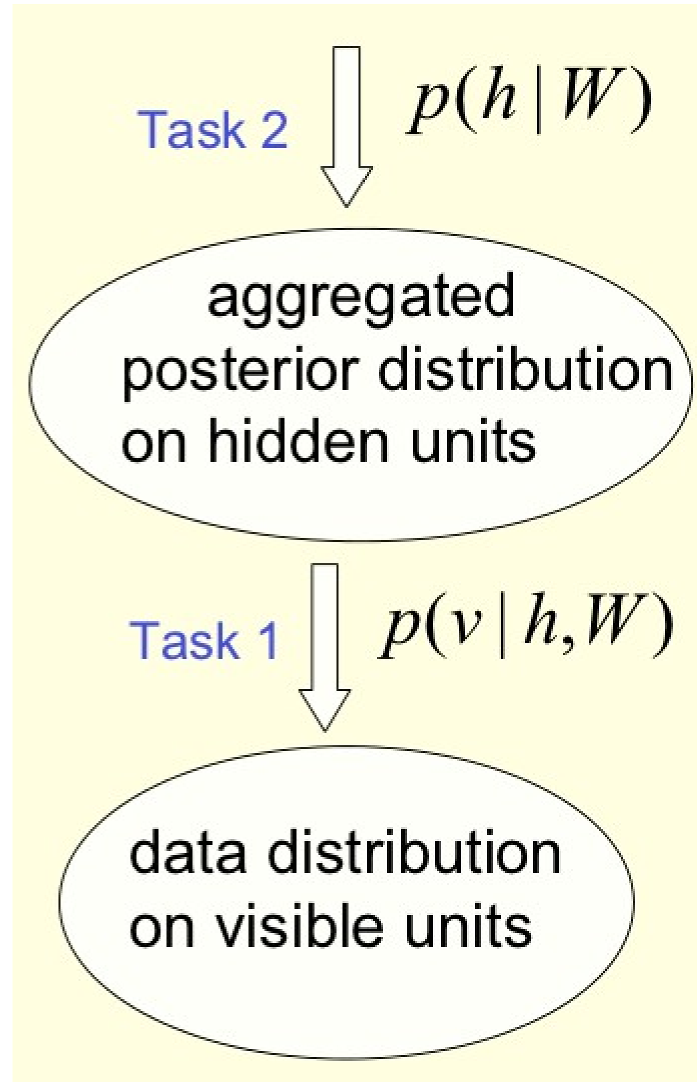
- The probability of turning on is determined by the weighted input from other units (plus a bias)



$$p(s_i = 1) \quad = \quad \frac{1}{1 + \exp(-b_i - \sum_j s_j w_{ji})}$$

Restricted Boltzmann Machine (RBM)





- Each RBM converts its data distribution into an aggregated posterior distribution over its hidden units.
- This divides the task of modeling its data into two tasks:

(Smolensky, 1986)

The Energy of a joint configuration

(ignoring terms to do with biases)

binary state of visible unit i

binary state of hidden unit j

$$E(v,h) = -\sum_{i,j} v_i h_j w_{ij}$$

Energy with configuration v on the visible units and h on the hidden units

weight between units i and j

$$-\frac{\partial E(v,h)}{\partial w_{ij}} = v_i h_j$$

(Hinton, 2007)

Weights → Energies → Probabilities

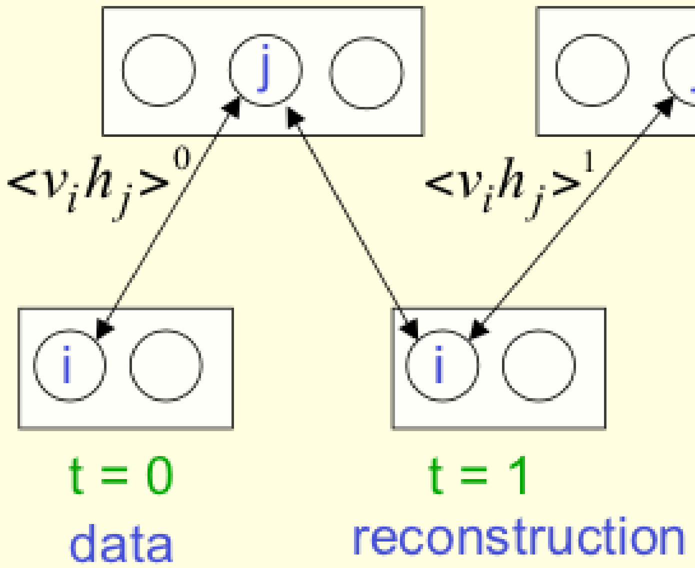
- Each possible joint configuration of the visible and hidden units has an energy
  - The energy is determined by the weights and biases (as in a Hopfield net).
- The energy of a joint configuration of the visible and hidden units determines its probability:

$$p(v,h) \propto e^{-E(v,h)}$$

- The probability of a configuration over the visible units is found by summing the probabilities of all the joint configurations that contain it.

(Hinton, 2007)

A quick way to learn an RBM



Start with a training vector on the visible units.

Update all the hidden units in parallel

Update the all the visible units in parallel to get a "reconstruction".

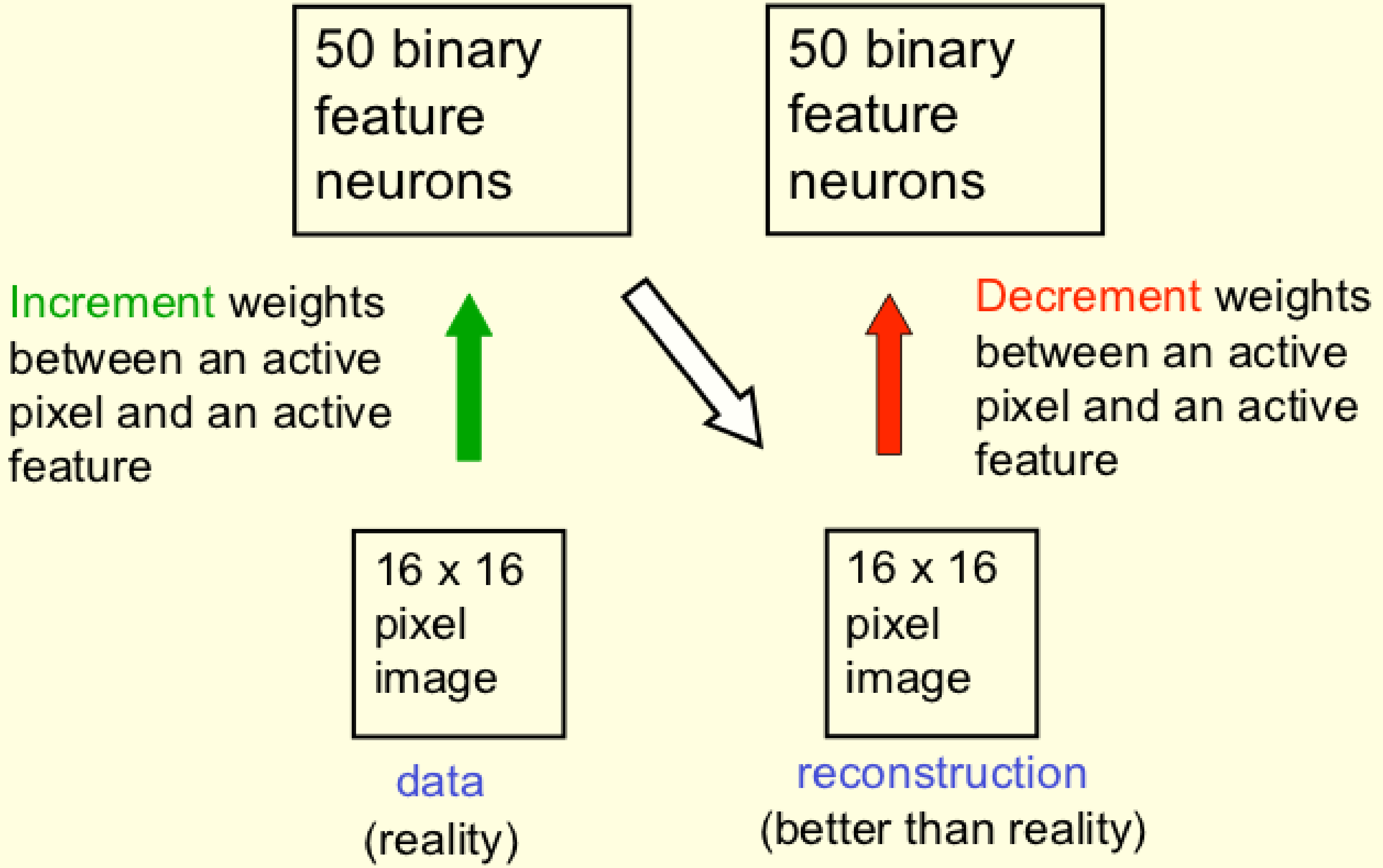
Update the hidden units again.

$$\Delta w_{ij} = \varepsilon \left(<v_i h_j>^0 - <v_i h_j>^1 \right)$$

This is not following the gradient of the log likelihood. But it works well. It is approximately following the gradient of another objective function (Carreira-Perpinan & Hinton, 2005).

(Hinton, 2007)

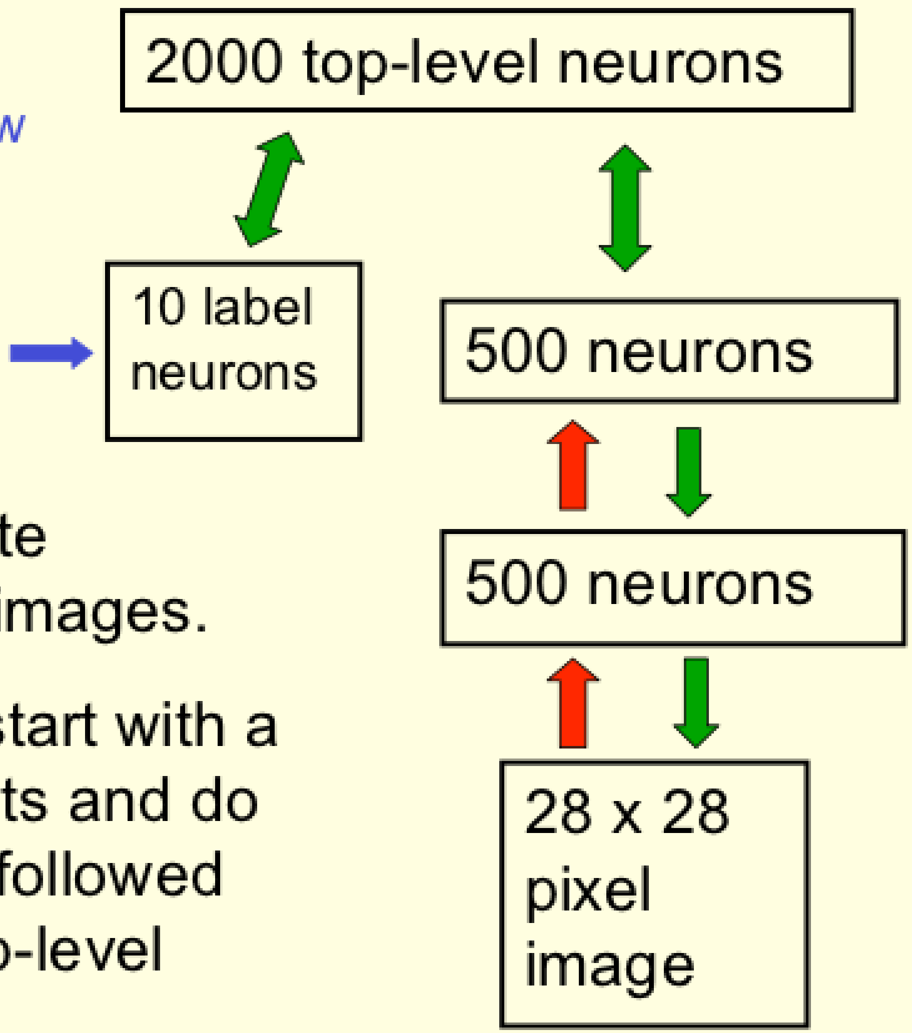
How to learn a set of features that are good for reconstructing images of the digit 2



Deep Belief Network (DBN) = stacked RBMs

The top two layers form an associative memory whose energy landscape models the low dimensional manifolds of the digits.

The energy valleys have names

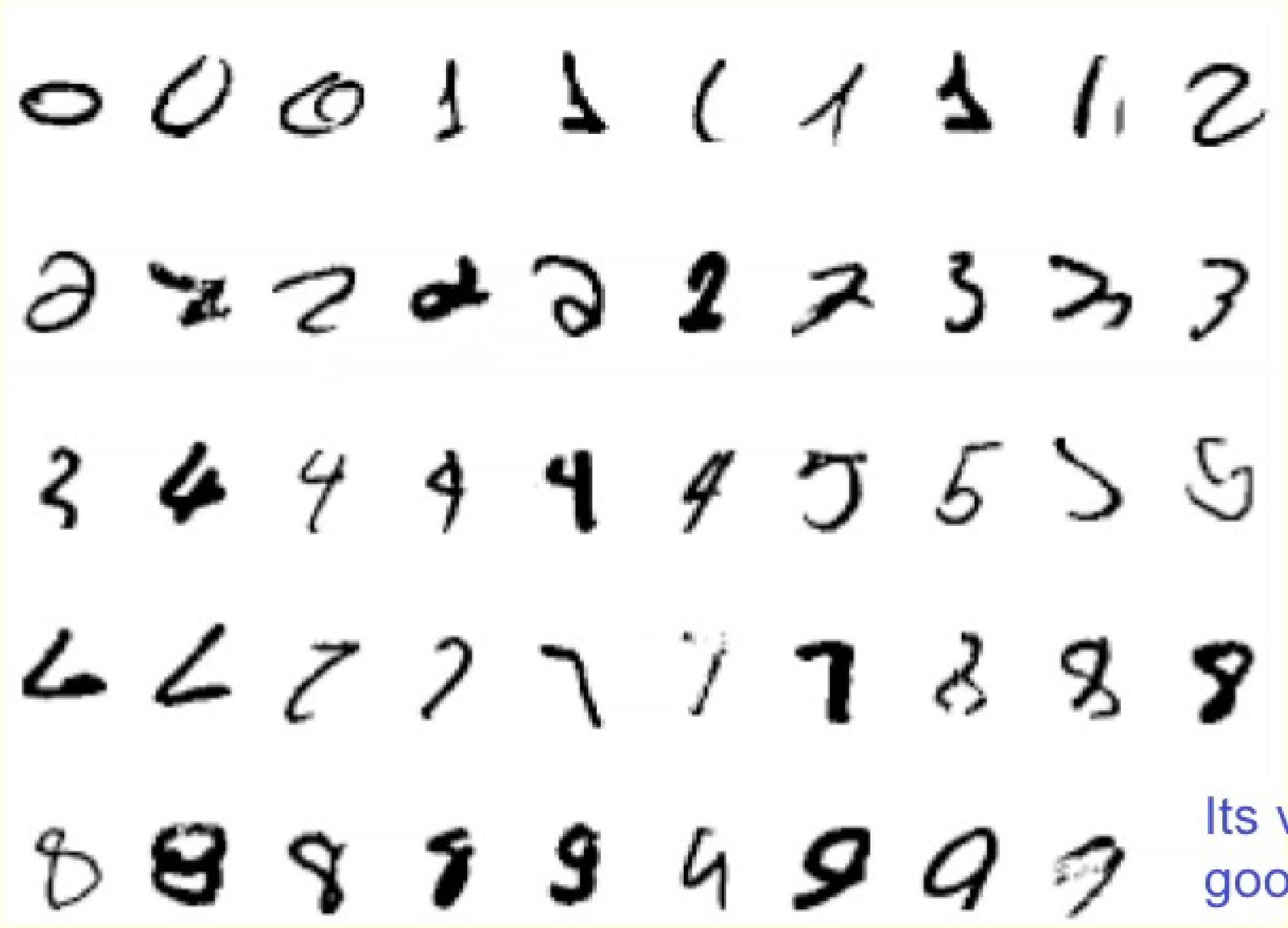


The model learns to generate combinations of labels and images.

To perform recognition we start with a neutral state of the label units and do an up-pass from the image followed by a few iterations of the top-level associative memory.

(Hinton, 2006)

Examples of correctly recognized handwritten digits that the neural network had never seen before



Its very good

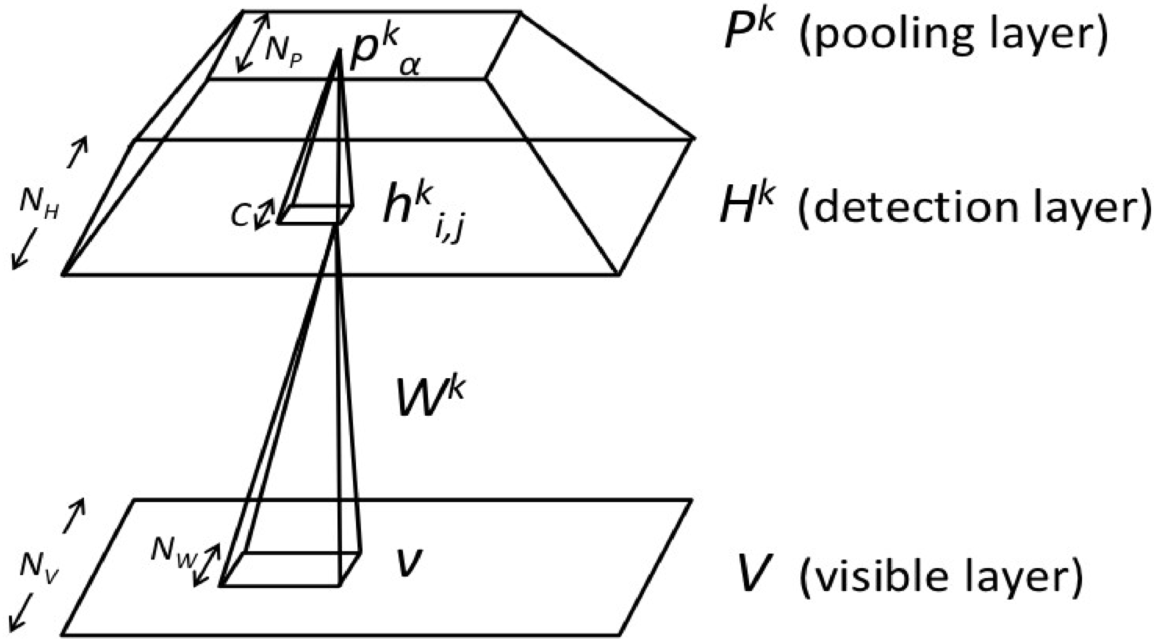
How well does it discriminate on MNIST test set with no extra information about geometric distortions?

• Generative model based on RBM's	1.25%
• Support Vector Machine (Decoste et. al.)	1.4%
• Backprop with 1000 hiddens (Platt)	~1.6%
• Backprop with 500 -->300 hiddens	~1.6%
• K-Nearest Neighbor	~ 3.3%

- See Le Cun et. al. 1998 for more results

- Its better than backprop and much more neurally plausible because the neurons only need to send one kind of signal, and the teacher can be another sensory input.

Convolutional DBN



- sharing weights among all locations within a layer
- Probabilistic max-pooling
- Both features lead to translational invariance and contribute to scalability

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{k=1}^{K} h^k \bullet (\tilde{W}^k * v) - \sum_{k=1}^{K} b_k \sum_{i,j} h^k_{i,j} - c \sum_{i,j} v_{ij}$$

$$\begin{aligned} P(h^k_{ij} = 1|\mathbf{v}) &= \sigma((\tilde{W}^k * v)_{ij} + b_k) \\ P(v_{ij} = 1|\mathbf{h}) &= \sigma((\sum_k W^k * h^k)_{ij} + c), \end{aligned}$$

(Lee et al, 2009)

CDBN performance on MNIST and Caltech-101 datasets

Table 2. Test error for MNIST dataset

Labeled training samples	1,000	2,000	3,000	5,000	60,000
CDBN	2.62±0.12%	2.13±0.10%	1.91±0.09%	1.59±0.11%	0.82%
Ranzato et al. (2007)	3.21%	2.53%	-	1.52%	0.64%
Hinton and Salakhutdinov (2006)	-	-	-	-	1.20%
Weston et al. (2008)	2.73%	-	1.83%	-	1.50%

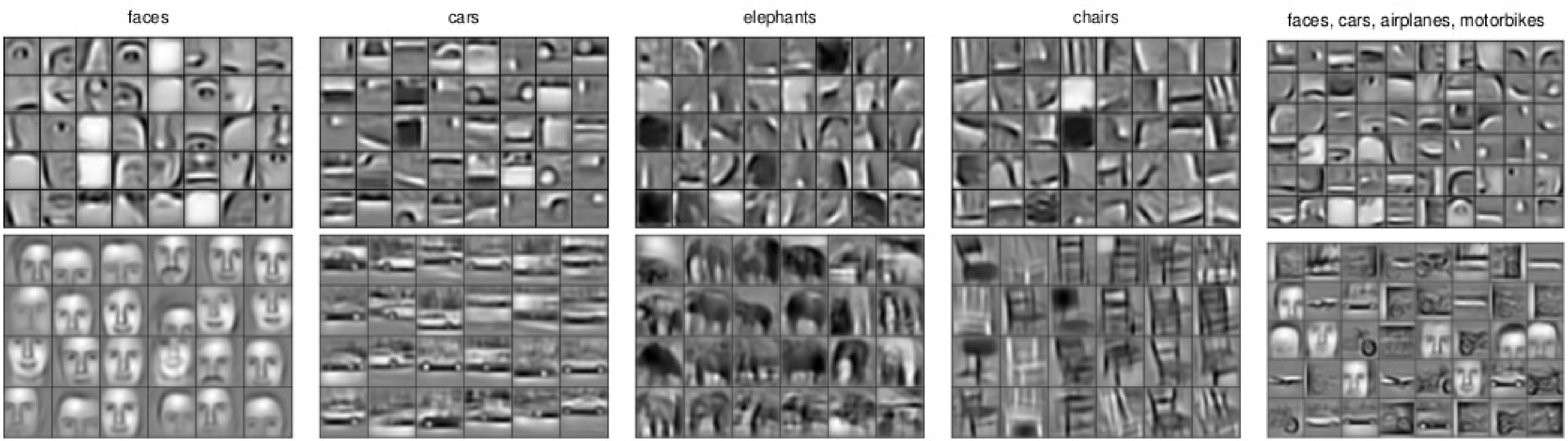


Figure 3. Columns 1-4: the second layer bases (top) and the third layer bases (bottom) learned from specific object categories. Column 5: the second layer bases (top) and the third layer bases (bottom) learned from a mixture of four object categories (faces, cars, airplanes, motorbikes).

(Lee et al, 2009)

CDBN – face reconstruction (Caltech-101)

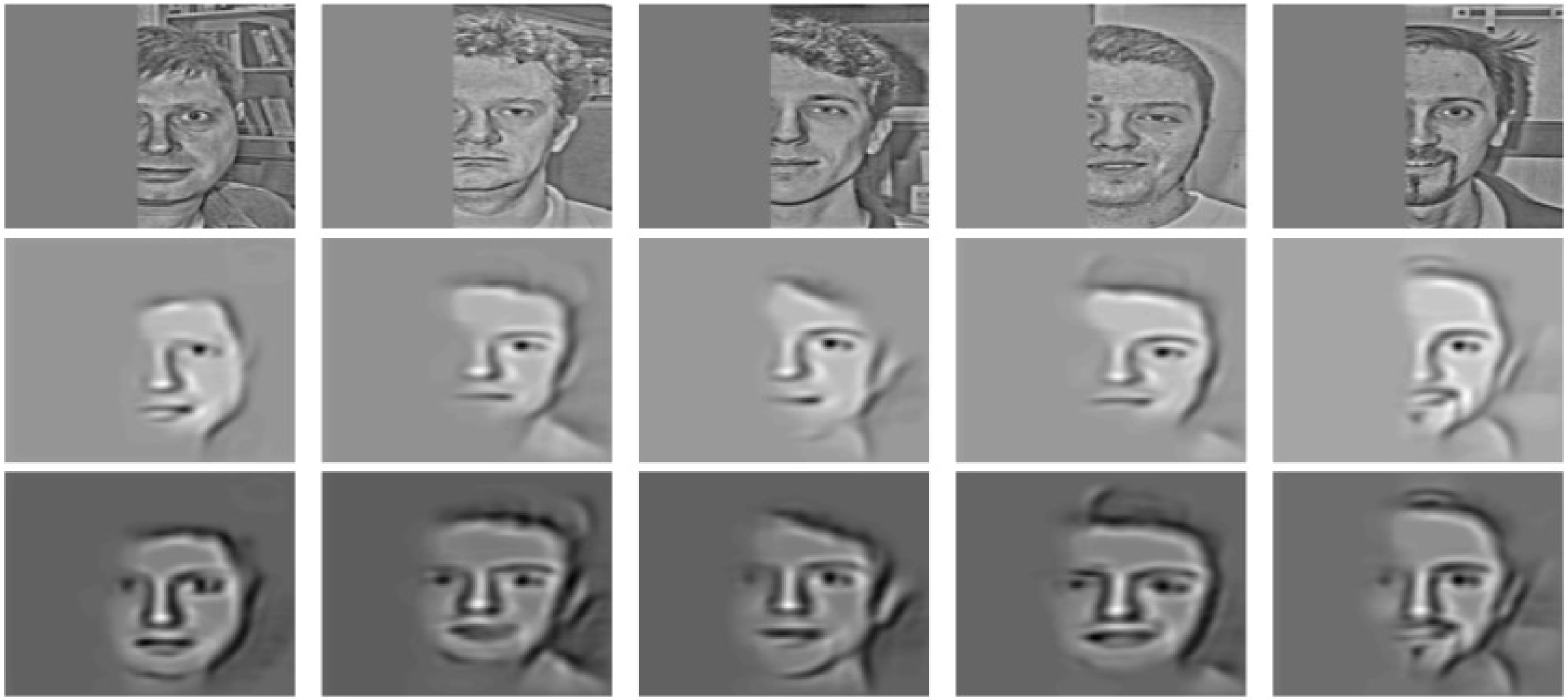
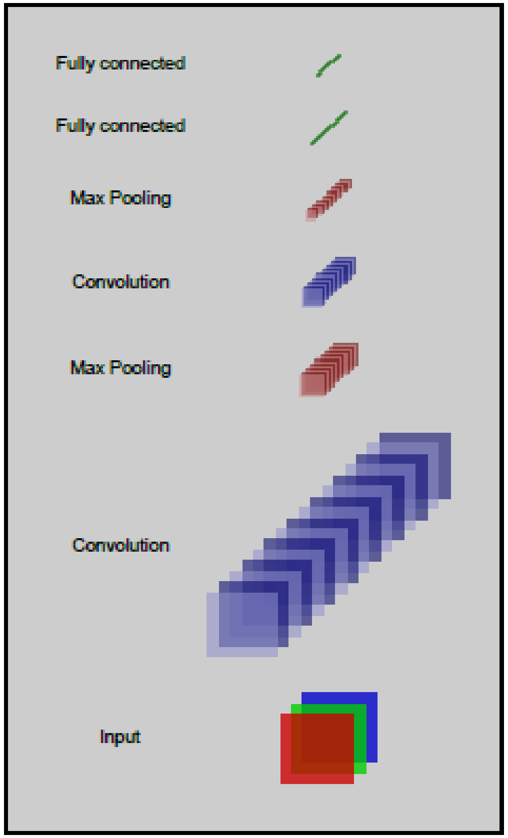
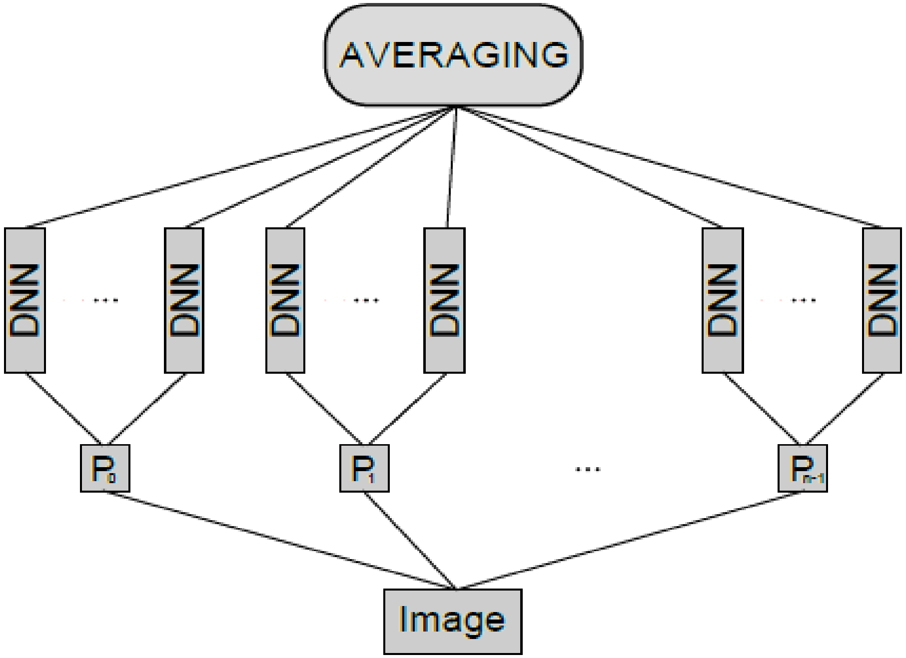


Figure 6. Hierarchical probabilistic inference. For each column: (top) input image. (middle) reconstruction from the second layer units after single bottom-up pass, by projecting the second layer activations into the image space. (bottom) reconstruction from the second layer units after 20 iterations of block Gibbs sampling.

Multi-column deep NN

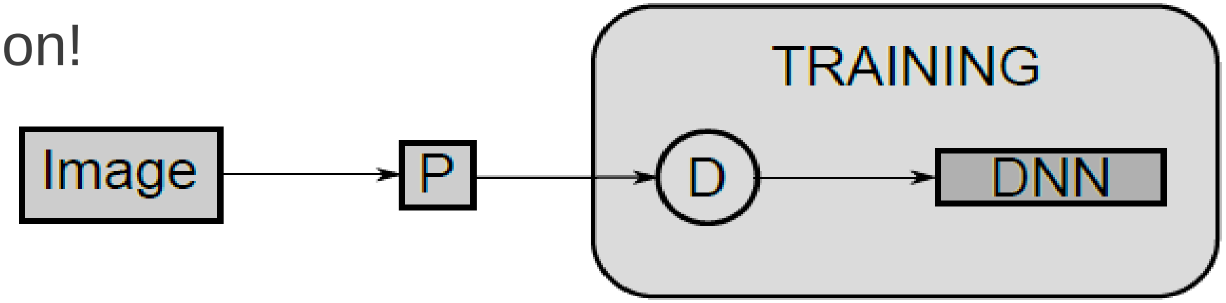


(a)



(b)

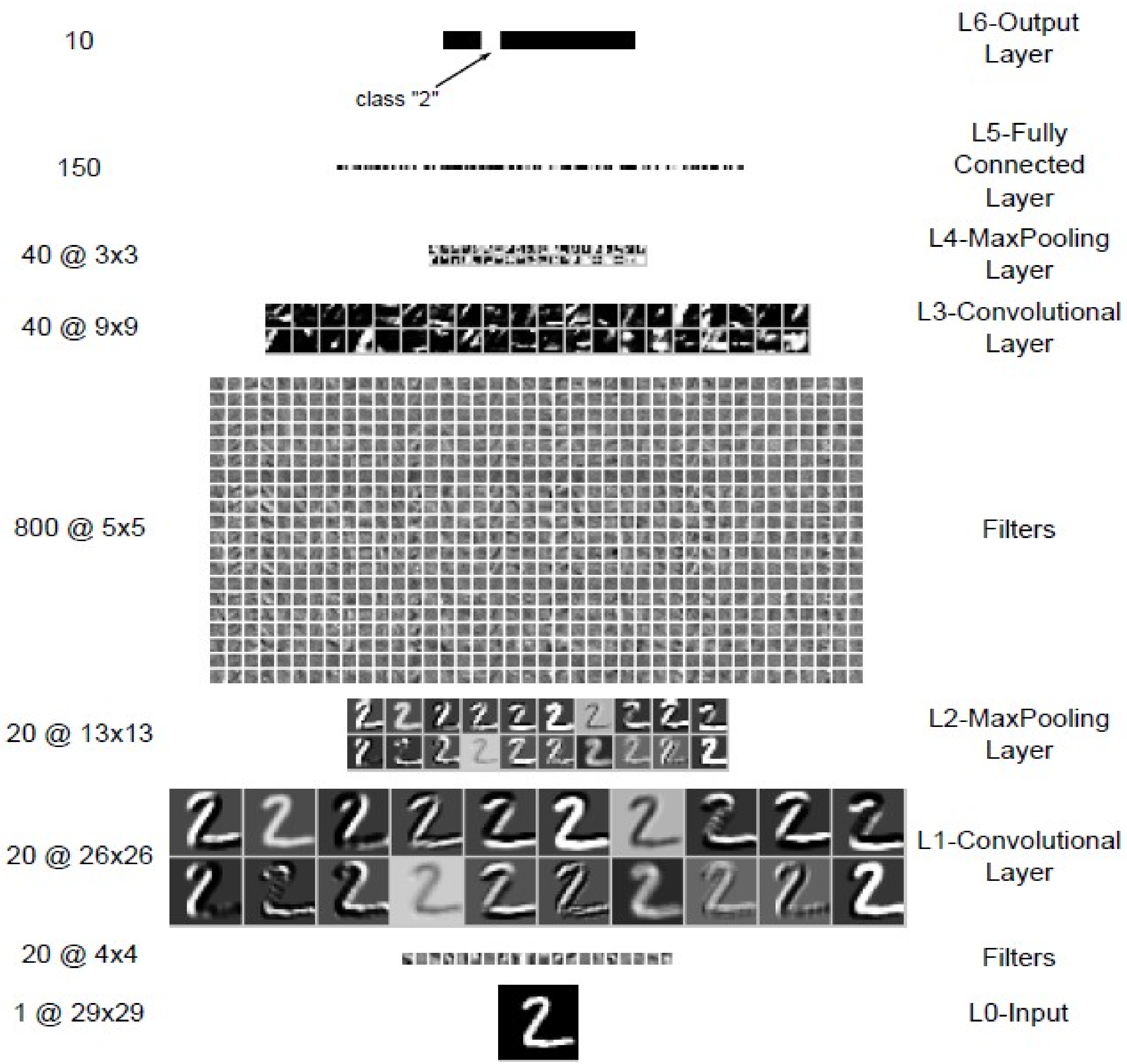
Trained by error back-propagation!



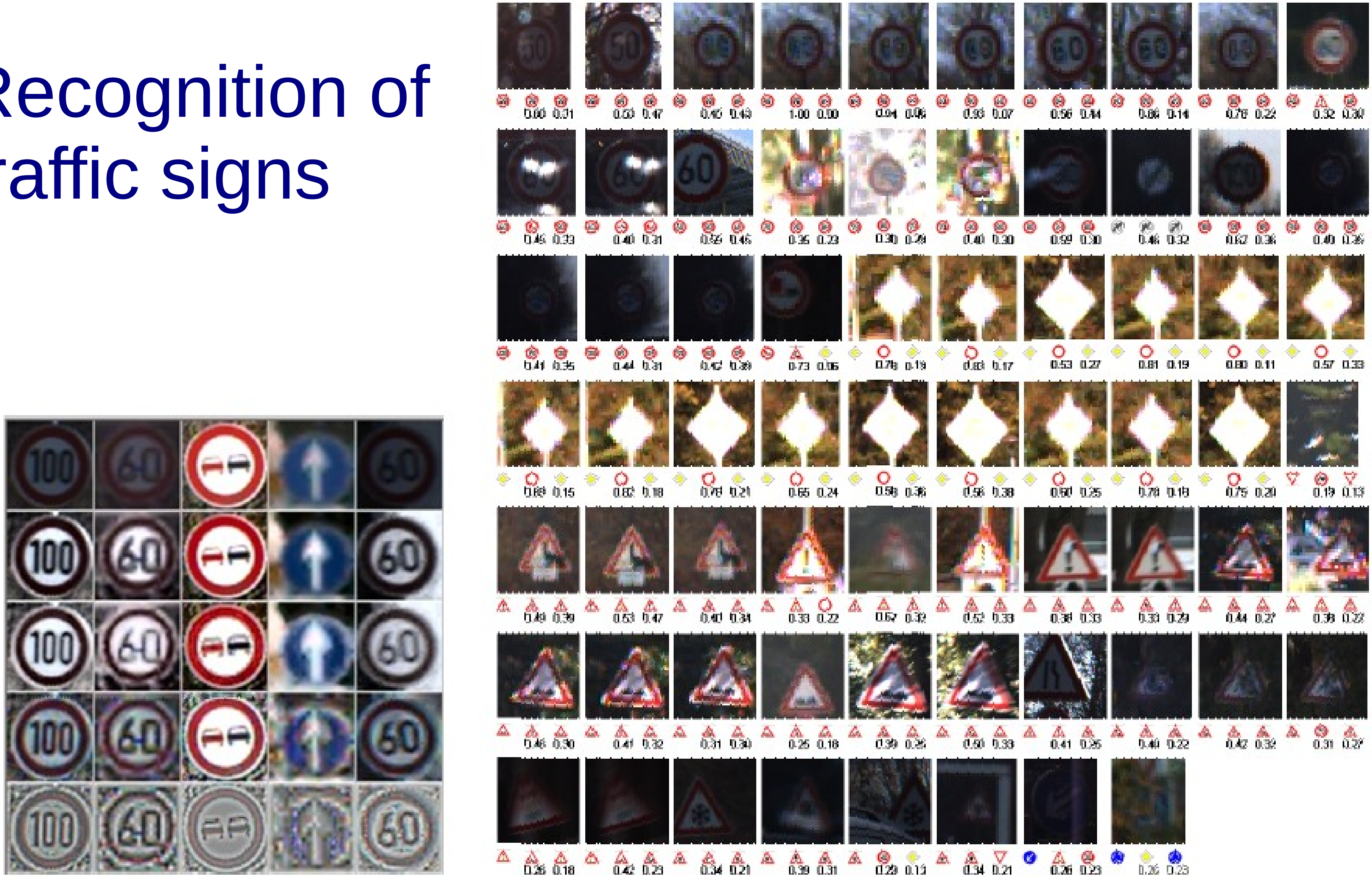
(c)

(Schmidhuber et al, 2010)

Digit recognition - MNIST



Recognition of traffic signs

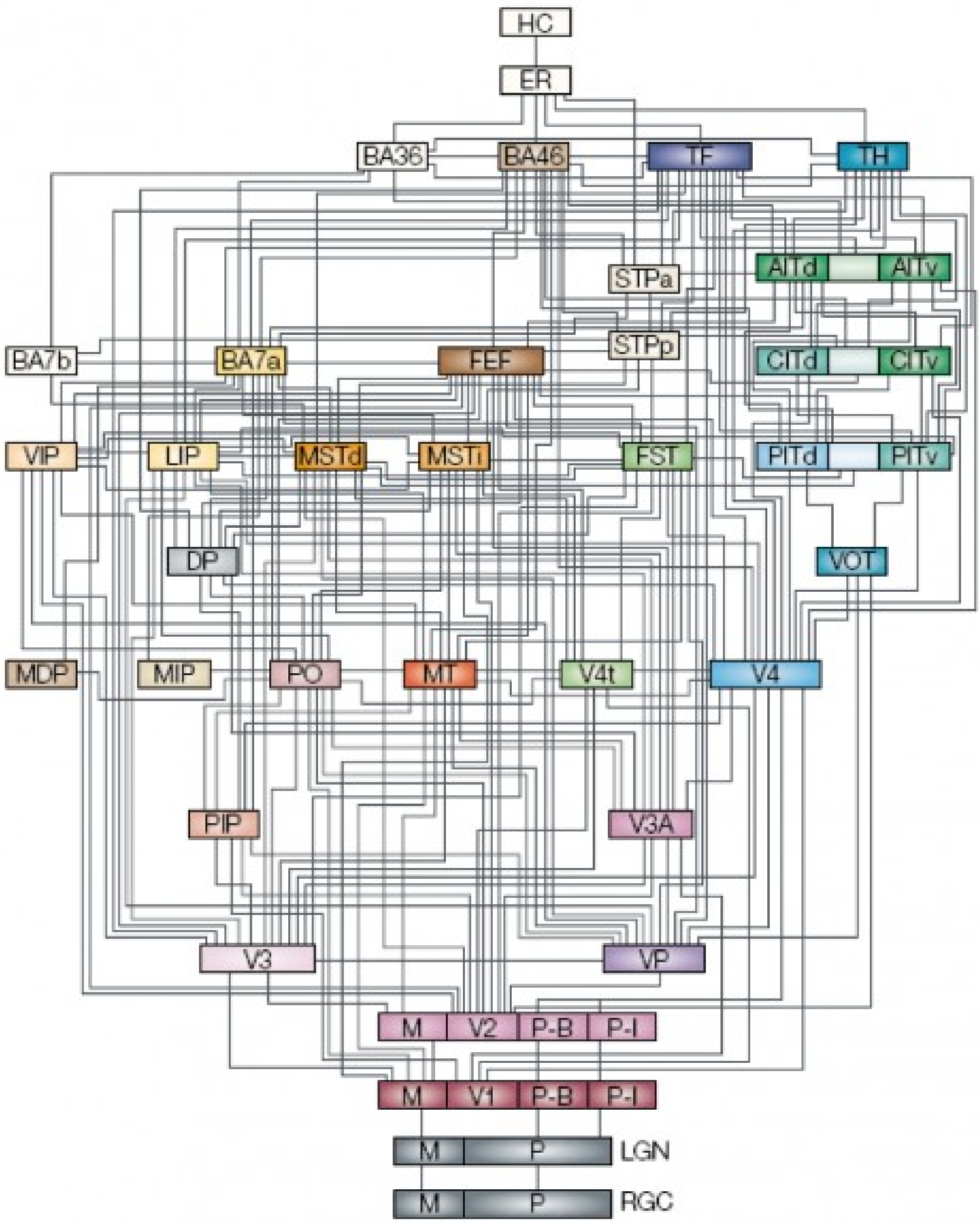


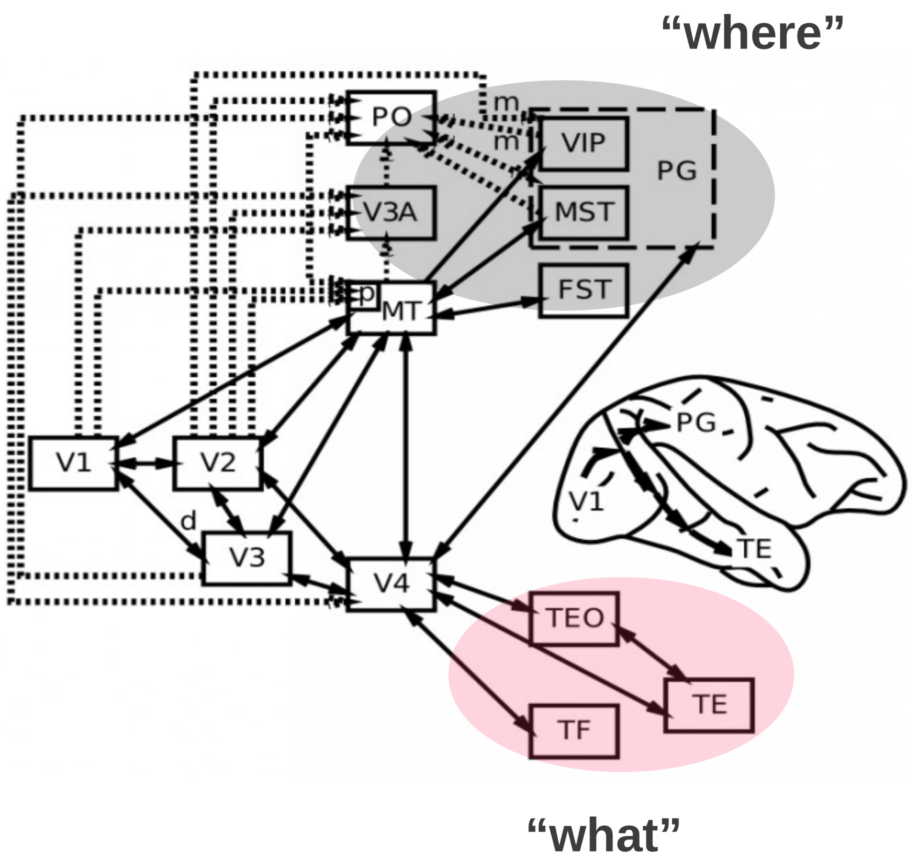
(a) (b)

Figure 3: (a) Preprocessed images, from top to bottom: original, Imadjust, Histeq, Adapthisteq, Conorm. (b) The 68 errors of the MCDNN, with correct label (left) and first and second best predictions (middle and right).

(Ciresan et al, 2012)

Mammalian visual system





Crucial role of spatial attention:
Triggered top-down or bottom-up

Summary

- Complex image recognition – extremely difficult
- Neural network approaches
  - Discriminative (e.g. back-propagation) - reNNaissance
  - Generative (e.g. DBNs, HTM,...)
    - provide added value (biologically plausible)
- Convolution useful in both approaches
- Attentional component inevitable for complex images
- Maybe more inspiration from biology