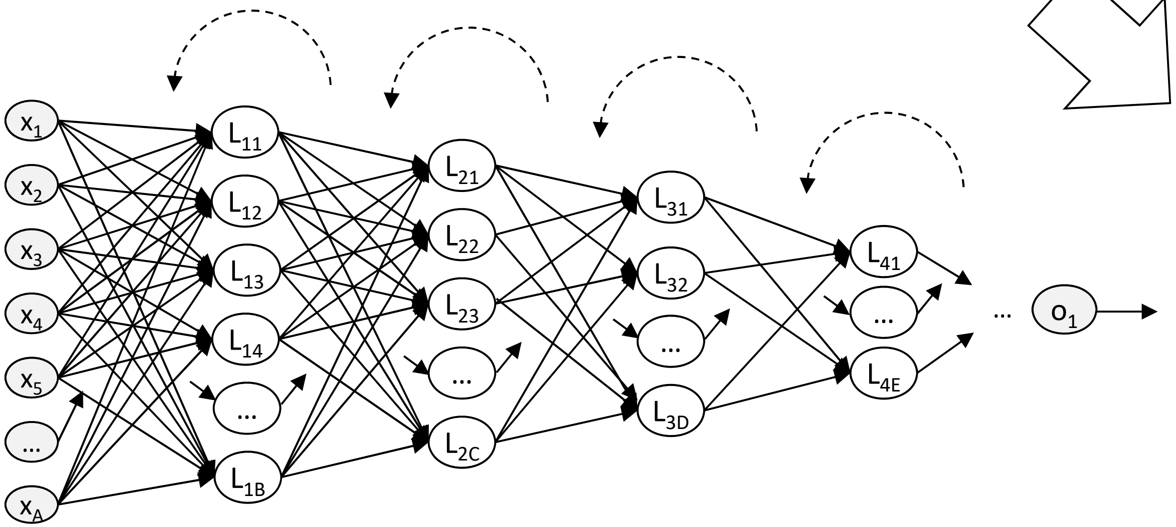
Neuroengineering (I)
5. Deep Learning with AutoEncoders

- **Scuola di Ingegneria Industriale e dell'Informazione** – Politecnico di Milano
- Prof. Pietro Cerveri

Troubles when training multi-layer fully connected NN

Vanishing gradient problem in the back-propagation

$$0 \leftarrow \frac{\partial E}{\partial w_{ij}^{(l-k)}} < \frac{\partial E}{\partial w_{ij}^{(l-1)}} < \frac{\partial E}{\partial w_{ij}^{(l)}} < \frac{\partial E}{\partial w_{ij}^{(l+1)}} <$$



1. Early hidden layers learn much more slowly than later hidden layers
2. Weights in the early hidden layers may undergo erratic updating

Solution: layer-by-layer analysis but .....
we loose supervision

Supervised vs unsupervised learning

Supervised learning

Data: (x, y)
x is input, y is label (category, numerical value)

Classification
Function modeling

Goal: learn a function to map x -> y

Unsupervised learning

Data: x
No labels

Clustering (k-means),
Dimensionality reduction (PCA),
Feature learning (**Autoencoders**)
Data density estimation (GAN)

Goal: learn some underlying hidden structure of the data

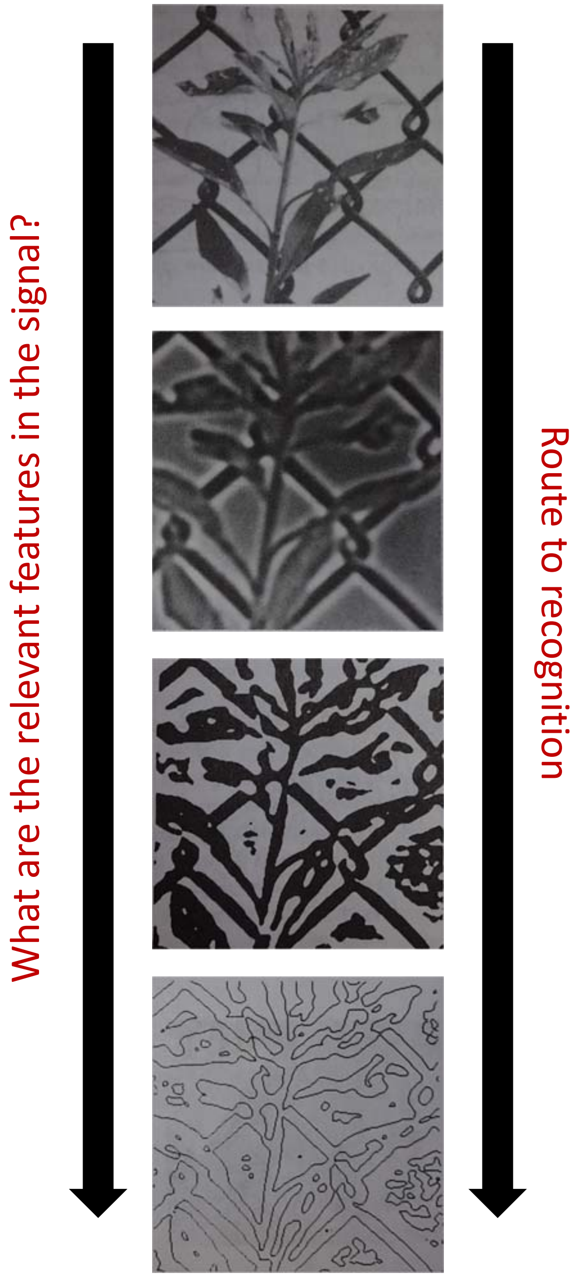
Autoencoders

An <u>autoencoder</u> network is typically a feedforward neural network aiming at learning a compressed, distributed representation (encoding) of a dataset.

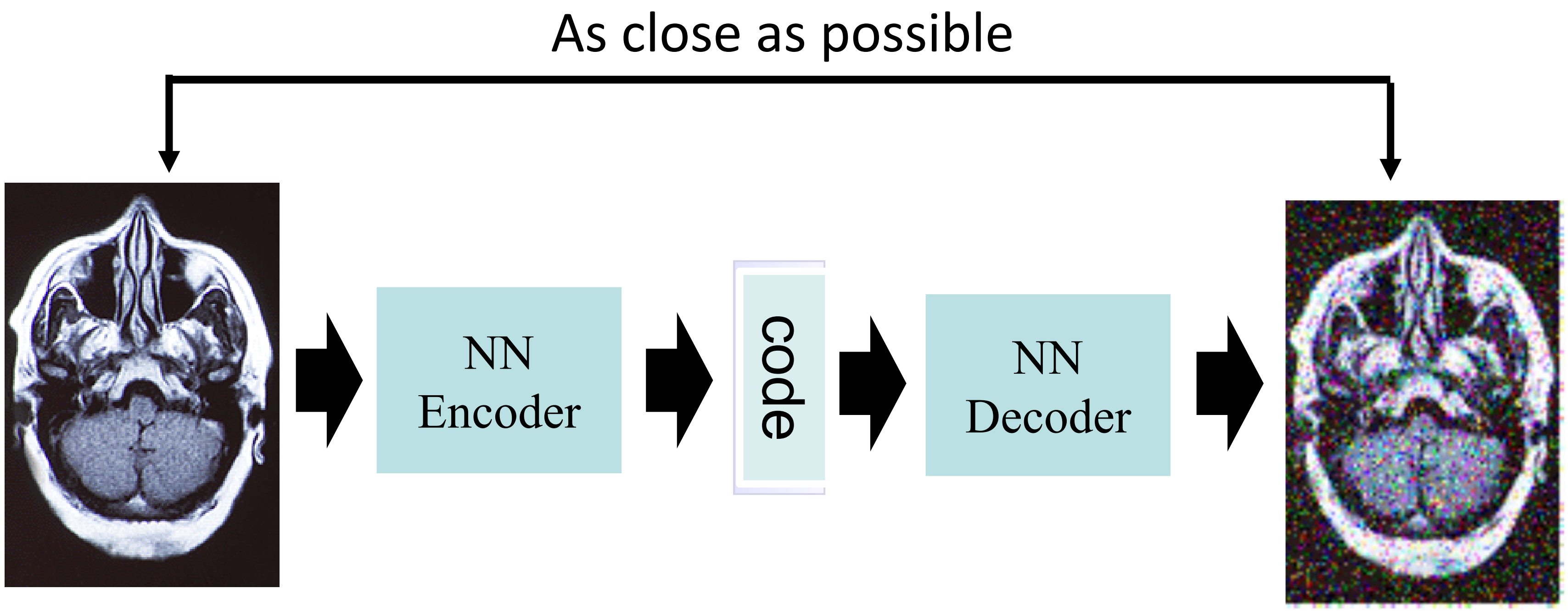
The network does not learn a “mapping” between the training data and its labels, but learns instead the internal structure of the data itself.

The network structure should force the network to learn only the most important features and achieves a dimensionality reduction.

Visual recognition is a typical task encompassing distributed and progressively compact representations

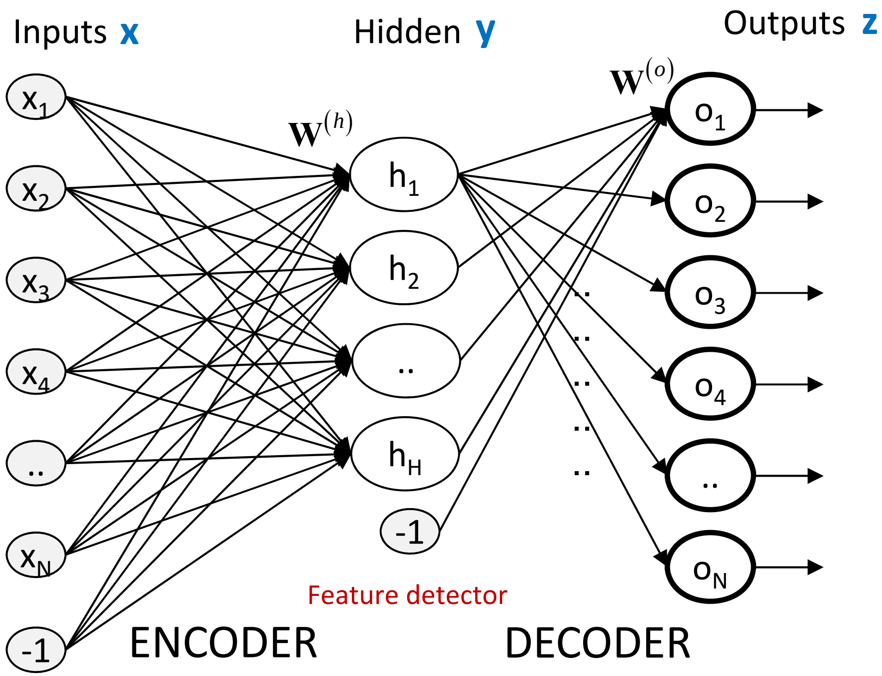


Basic autoencoder



Autoencoder network

- FFNN with 3 layers that recreates the input in the output $Z \equiv X$
- The number of hidden units H is lower than the number of inputs N



φ Sigmoidal activation function

The hidden layer encodes a compressed and distributed representation of the input dataset.

The training corresponds to learn an approximation to the identity function

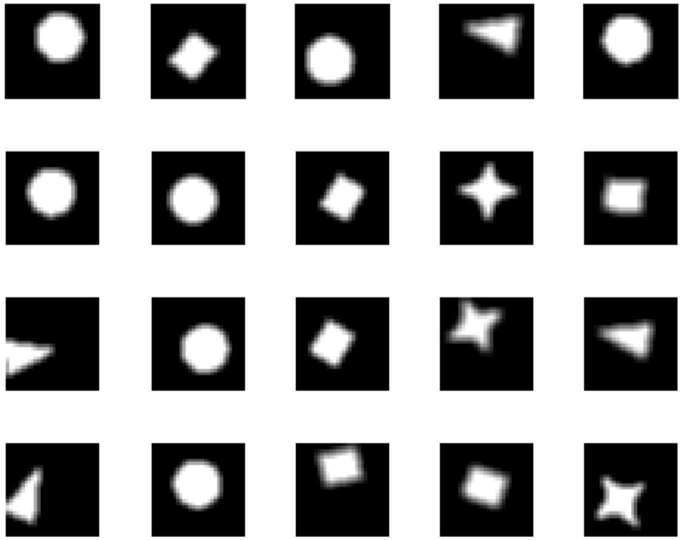
Activation of the hidden neuron *i*

$$y_i = \varphi\left(\sum_j^N w_{ij}^{(h)} x_j + b_i\right) = \varphi_i^{(h)}(\mathbf{x})$$

Activation of the output neuron *i*

$$z_i = \varphi\left(\sum_j^H w_{ij}^{(o)} y_j + c_i\right) = \varphi_i^{(o)}(\mathbf{y})$$

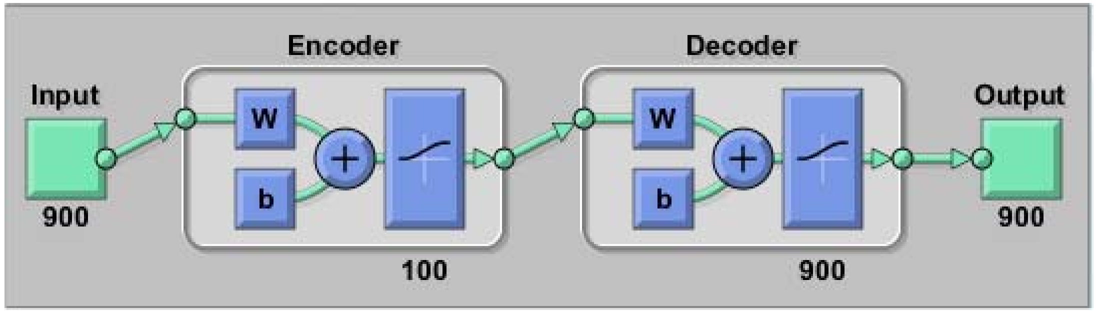
Visual feature encoding



Example: shape recognition on gray images
Four shapes: circle, triangle, rectangle, star

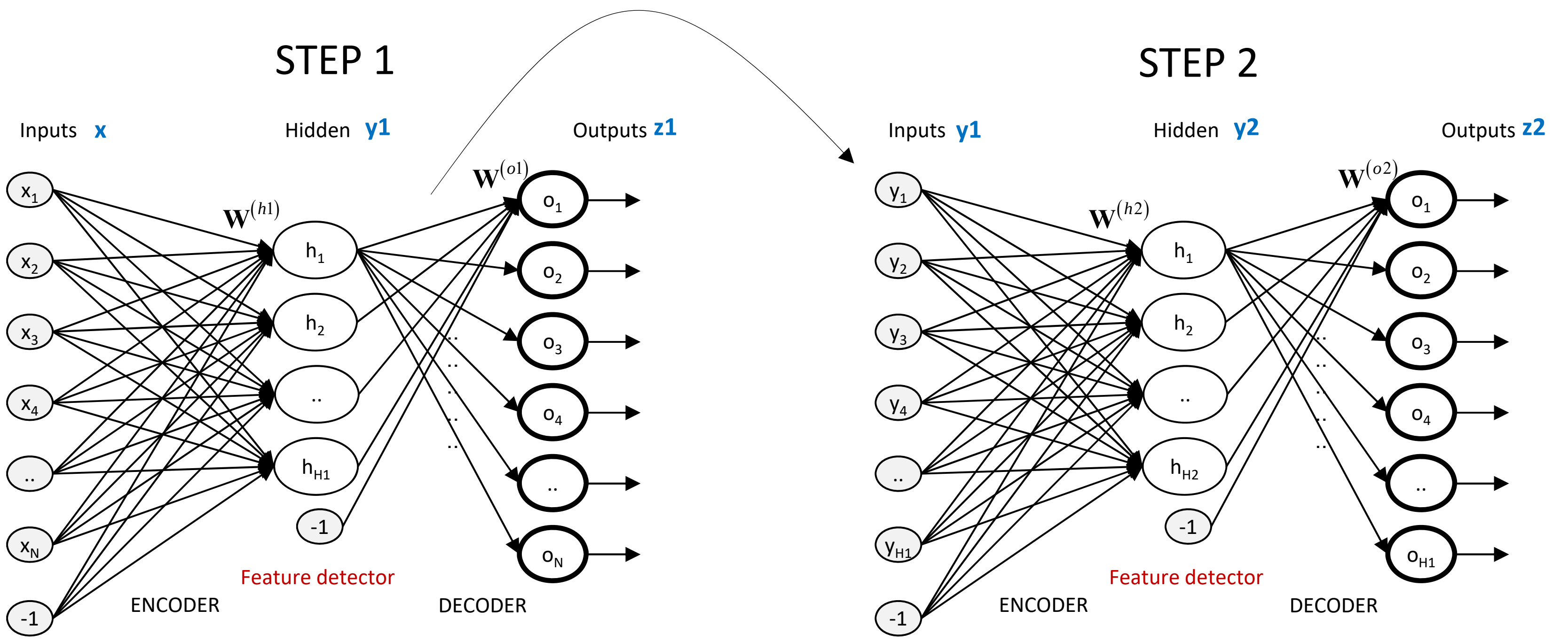
Image size: 30×30 (900 pixels) $\mathbf{x}=[x_1, x_2, \ldots x_{900}]$

Hidden units: 100
Compression ratio: 1/9 $\mathbf{y}=[y_1, y_2, \ldots y_{100}]$



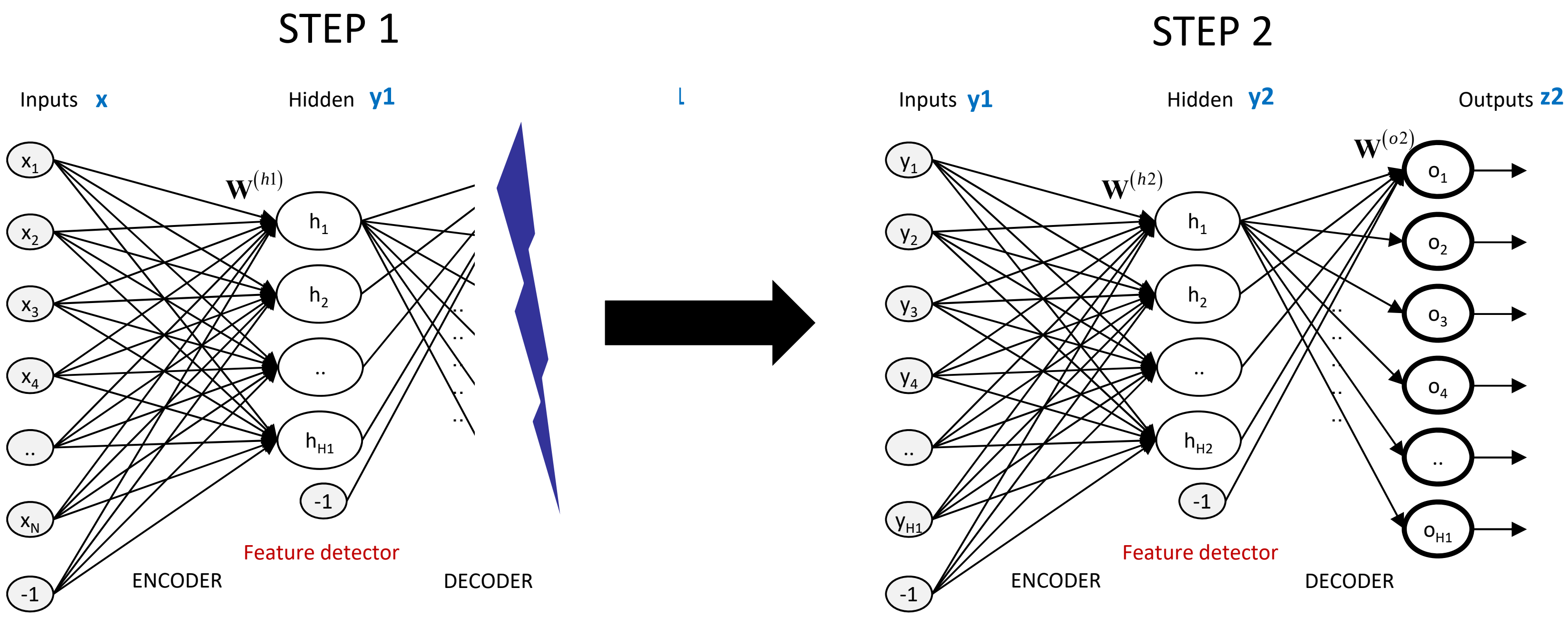
What features are the encoded by signals y_i?

Progressive encoding (stacking net)



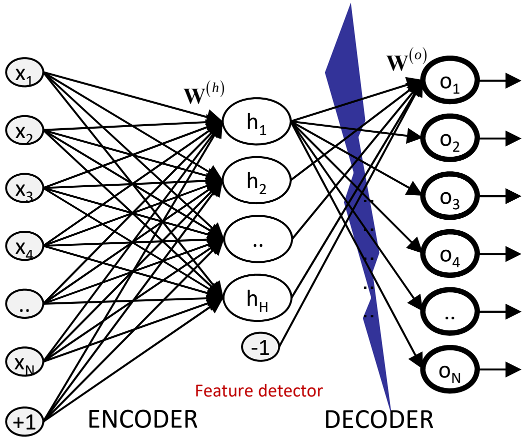
H2 < H1

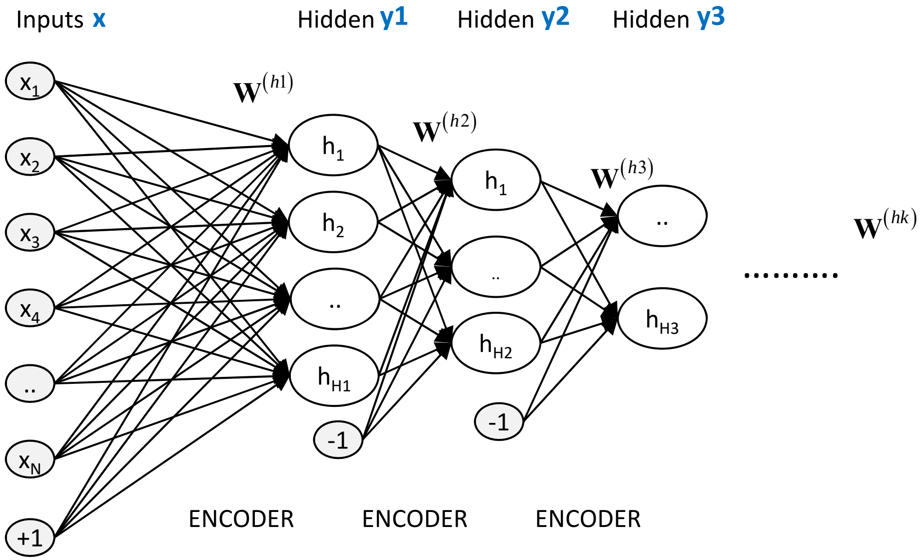
Progressive encoding (stacking net)



H2 < H1

Autoencoder network cascade





HOW to TRAIN?

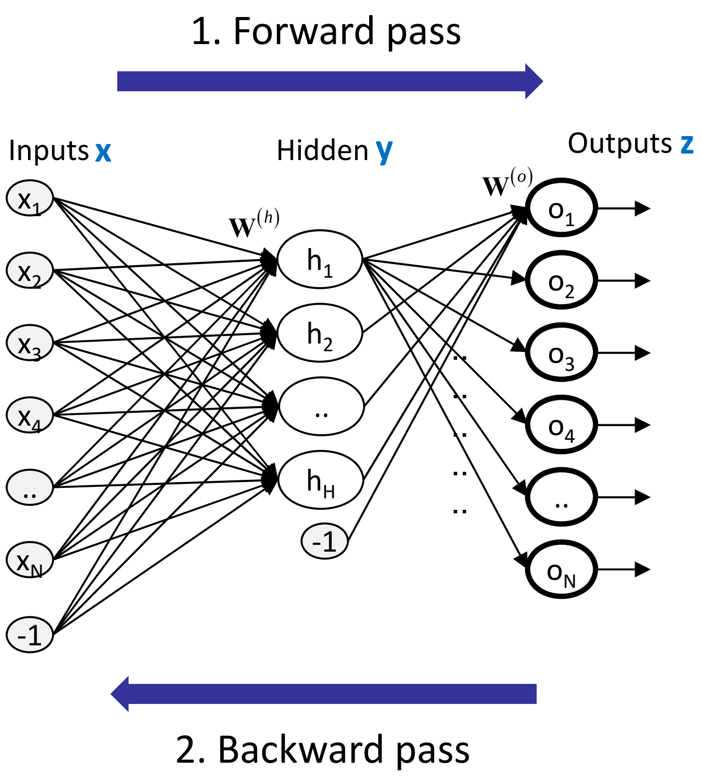
Autoencoder training

"Unsupervised" backpropagation

$$E = \sum_{k=1}^{R} \sum_{i=1}^{N} \frac{1}{2} \left(x_i^{(k)} - z_i^{(k)} \right)^2$$

$$\Delta w_{ij}^{(l)} = \eta \sum_{k=1}^{R} \delta_i^{(l),(k)} e_j^{(l-1),(k)}$$

- ❑ Learning the identity function is ill-posed problem when the number of hidden units is lower than the number of inputs
- ❑ Constraining the weight optimization

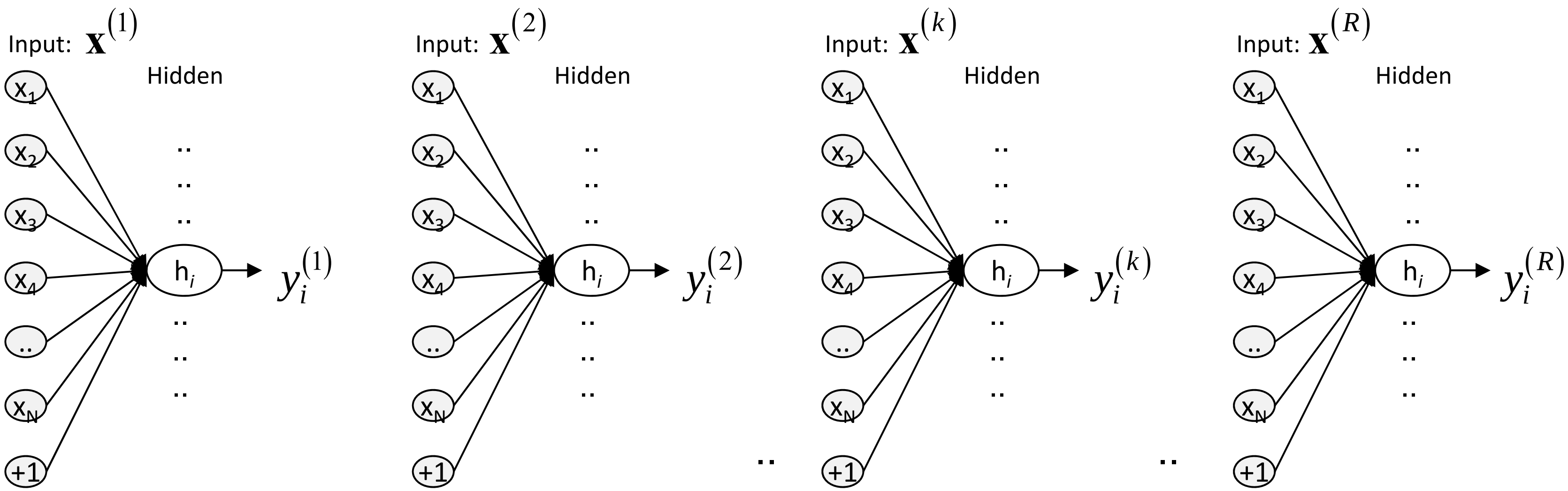


Activation of the hidden neuron

$$y_i = \varphi\left(\sum_{j}^{N} w_{ij}^{(h)} x_j + b_i\right) = \varphi_i^{(h)}(\mathbf{x})$$

Activation of the hidden neuron *i* in correspondence of the *k* input vector

Training set: $\left\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(R)}\right\}$ $\qquad y_i^{(k)} = \varphi_i^{(h)}\left(\mathbf{x}^{(k)}\right)$



Sparse autoencoders

- the term neural coding is adopted to denote the patterns of electrical activity of neurons induced by a stimulus

- Sparse coding in its turn is one kind of pattern

- A coding paradigm is said to be sparse when a stimulus (like an image) yields the activation of just a relatively small number of neurons, that combined represent it in a sparse way

- In machine learning, the same criterion can be used to implement **sparse** autoencoders, which are regular autoencoders trained with a sparsity constraint

Optimization constraint for learning

Autoencoder training principle

SPARSITY constraint on hidden units

Assumption

Active (firing) neuron → output close to 1
Inactive neuron → output close to 0 (-1)

Average activation of hidden unit *i* over the training dataset

$$\hat{\rho}_i = \frac{1}{R}\sum_{k}^{R} y_i^{(k)} = \frac{1}{R}\sum_{k}^{R} \varphi_i^{(h)}\left(\mathbf{x}^{(k)}\right) = \frac{1}{R}\sum_{k}^{R}\left(\sum_{j}^{N} w_{ij}^{(h)} x_j^{(k)} + b_i\right)$$

Constraining the hidden neurons to be inactive most of the time

Average activation $\hat{\rho}_i = \rho$ Sparsity parameter (e.g. $\rho = 0.05$)

Backpropagation

$$E = \sum_{k=1}^{R} E^{(k)} = \sum_{k=1}^{R}\sum_{i=1}^{N}\frac{1}{2}\left(t_i^{(k)} - u_i^{(k)}\right)^2$$

$e_j^{(l-1),(k)}$ is the output signal of the *j*-th neuron of the (*l*-1)-th layer in correspondence of the *k*-th pattern of the training set

$$\delta_i^{(l),(k)} = \left(t_i^{(k)} - u_i^{(k)}\right) f'\left(P_i^{(k)}\right) \quad if \;\; l = L$$

P_i is the action potential of neuron *i*

$$\delta_i^{(l),(k)} = f'\left(P_i^{(k)}\right)\sum_{r=1}^{M_{l+1}}\left(\delta_r^{(l+1),(k)} w_{ri}^{(l+1)}\right) \quad if \;\; l < L$$

M_{l+1} is the number of neurons in the (l+1)-th layer

$$\Delta w_{ij}^{(l)} = \eta \sum_{k=1}^{R} \delta_i^{(l),(k)} e_j^{(l-1),(k)} \Longrightarrow w_{ij}^{(l)} = w_{ij}^{(l)} + \Delta w_{ij}^{(l)}$$

Backpropagation with weight constrain

$$E = \sum_{k=1}^{R} E^{(k)} = \sum_{k=1}^{R}\sum_{i=1}^{N} \frac{1}{2}\left(t_i^{(k)} - u_i^{(k)}\right)^2$$

$e_j^{(l-1),(k)}$ is the output signal of the *j*-th neuron of the (*l*-1)-th layer in correspondence of the *k*-th pattern of the training set

$$\delta_i^{(l),(k)} = \left(t_i^{(k)} - u_i^{(k)}\right) f'\left(P_i^{(k)}\right) \quad if\ \ l = L$$

P_i is the action potential of neuron *i*

$$\delta_i^{(l),(k)} = f'\left(P_i^{(k)}\right) \sum_{r=1}^{M_{l+1}} \left(\delta_r^{(l+1),(k)} w_{ri}^{(l+1)}\right) \quad if\ \ l < L$$

M_{l+1} is the number of neurons in the (l+1)-th layer

$$\Delta w_{ij}^{(l)} = \eta \sum_{k=1}^{R} \delta_i^{(l),(k)} e_j^{(l-1),(k)} \longrightarrow w_{ij}^{(l)} = w_{ij}^{(l)} + \Delta w_{ij}^{(l)}$$

$$E = \sum_{k=1}^{R}\left(\sum_{i=1}^{N} \frac{1}{2}\left(t_i^{(k)} - u_i^{(k)}\right)^2\right) +$$

Regularization term:

It tends to decrease the magnitude of the weights, and helps prevent overfitting

$$\frac{\alpha}{2} \sum_{l=2}^{L} \sum_{i}^{N_l} \sum_{j}^{N_{l-1}} \left(w_{ij}^{(l)}\right)^2$$

$\alpha << 1$

$$\Delta w_{ij}^{(l)} = \eta \left(\sum_{k=1}^{R} \delta_i^{(l),(k)} e_j^{(l-1),(k)} + \alpha w_{ij}^{(l)}\right) \longrightarrow w_{ij}^{(l)} = w_{ij}^{(l)} + \Delta w_{ij}^{(l)}$$

Sparsity in error function (three-layer autoencoder network)

$$E = \sum_{k=1}^{R}\left(\sum_{i=1}^{N}\frac{1}{2}\left(x_i^{(k)} - z_i^{(k)}\right)^2\right) + \frac{\alpha}{2}\sum_{l=2}^{L=3}\sum_{i}^{N_l}\sum_{j}^{N_{l-1}}\left(w_{ij}^{(l)}\right)^2 + \beta G\left(\{\hat{\rho}_i\},\rho\right)$$

Sparsity parameter

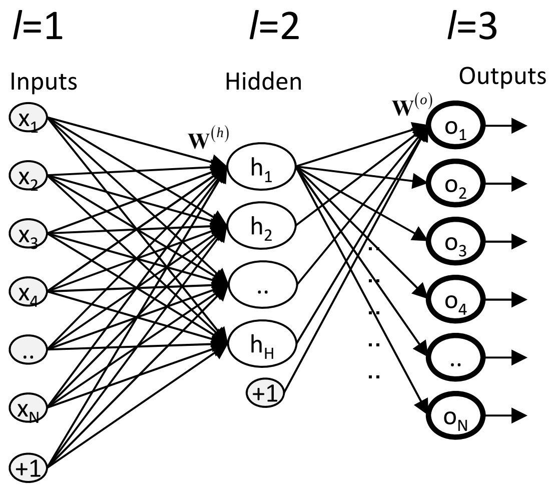
Regularization weight (>0)

Penalty factor G

Average activation of hidden units

$$\hat{\rho}_i = \frac{1}{R}\sum_{k}^{R} y_i^{(k)} = \frac{1}{R}\sum_{k}^{R}\varphi_i\left(\sum_{j}^{N} w_{ij}^{(h)} x_j^{(k)}\right) = f\left(w_{ij}\right)$$

Weights of the hidden neurons



Penalty factor $G\left(\{\hat{\rho}_i\},\rho\right)$

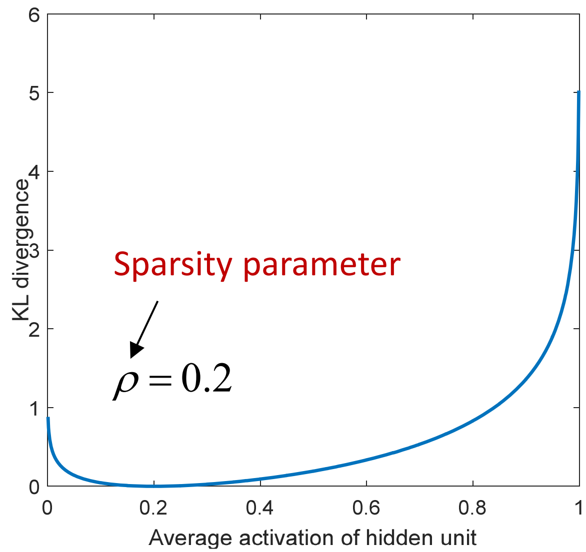
$$G\left(\{\hat{\rho}_i\},\rho\right)=\sum_{i}^{H}\left(\rho\log\frac{\rho}{\hat{\rho}_i}+(1-\rho)\log\frac{(1-\rho)}{(1-\hat{\rho}_i)}\right)$$

H: number of hidden units

$$KL=\rho\log\frac{\rho}{\hat{\rho}_i}+(1-\rho)\log\frac{(1-\rho)}{(1-\hat{\rho}_i)}$$

Kullback-Leibler (KL) divergence

- a standard function for measuring how different two different distributions are.



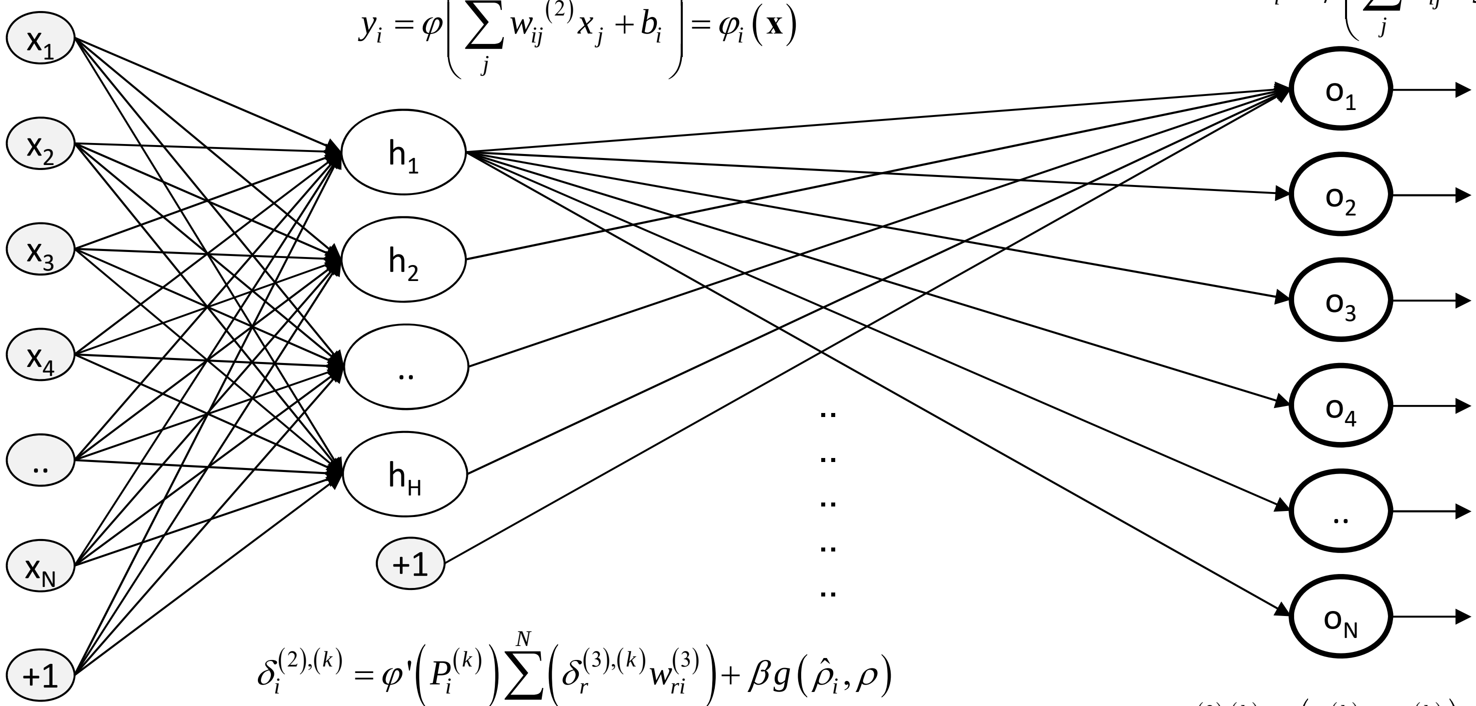
$$\hat{\rho}_i->\rho \;\Rightarrow\; KL->0$$
$$\hat{\rho}_i->1 \;\Rightarrow\; KL->\infty$$
$$\hat{\rho}_i->0 \;\Rightarrow\; KL->\infty$$

Computing δ values

Layer 1 Layer 2 Layer 3

$$y_i = \varphi\left(\sum_j^N w_{ij}^{(2)} x_j + b_i\right) = \varphi_i(\mathbf{x})$$

$$z_i = \varphi\left(\sum_j^H w_{ij}^{(3)} y_j + c_i\right) = \varphi_i(\mathbf{y})$$



$$\delta_i^{(2),(k)} = \varphi'\left(P_i^{(k)}\right)\sum_{r=1}^{N}\left(\delta_r^{(3),(k)} w_{ri}^{(3)}\right) + \beta g\left(\hat{\rho}_i, \rho\right)$$

$$\Delta w_{ij}^{(2)} = \eta \sum_{k=1}^{R} \delta_i^{(2),(k)} x_j^{(k)}$$

$$\delta_i^{(3),(k)} = \left(x_i^{(k)} - z_i^{(k)}\right)\varphi'\left(P_i^{(k)}\right)$$

$$\Delta w_{ij}^{(3)} = \eta \sum_{k=1}^{R} \delta_i^{(3),(k)} y_j^{(k)}$$

Derivative of the penalty factor *G* wrt w_{ij}

Derivative $g\left(\left\{\hat{\rho}_i\right\},\rho\right)$

$$G\left(\left\{\hat{\rho}_i\right\},\rho\right)=\sum_i^H\left(\rho\log\frac{\rho}{\hat{\rho}_i}+(1-\rho)\log\frac{(1-\rho)}{(1-\hat{\rho}_i)}\right)$$

H: number of hidden units

$$\hat{\rho}_i=\frac{1}{R}\sum_k^R y_i^{(k)}=\frac{1}{R}\sum_k^R\varphi_i\left(\mathbf{x}^{(k)}\right)=\frac{1}{R}\sum_k^R\left(\sum_j^N w_{ij}^{(2)}x_j^{(k)}\right)$$

Average activation

R: number of training data

$$\frac{\partial G\left(\hat{\rho}_i,\rho\right)}{\partial w_{ij}}=g\left(\hat{\rho}_i,\rho\right)=\frac{\partial\left(\rho\log\frac{\rho}{\hat{\rho}_i}+(1-\rho)\log\frac{(1-\rho)}{(1-\hat{\rho}_i)}\right)}{\partial w_{ij}}=\frac{\partial\left(\rho\log\frac{\rho}{\hat{\rho}_i}+(1-\rho)\log\frac{(1-\rho)}{(1-\hat{\rho}_i)}\right)}{\partial\hat{\rho}_i}\frac{\partial\hat{\rho}_i}{\partial w_{ij}}$$

$$\left(\frac{(1-\rho)}{(1-\hat{\rho}_i)}-\frac{\rho}{\hat{\rho}_i}\right)\frac{\partial\hat{\rho}_i}{\partial w_{ij}}=\left(\frac{(1-\rho)}{(1-\hat{\rho}_i)}-\frac{\rho}{\hat{\rho}_i}\right)\frac{1}{R}\sum_k^R\varphi'\left(P_i\right)$$

$$\Downarrow$$

$$\delta_i^{(2),(k)}=\varphi'\left(P_i^{(k)}\right)\left(\sum_{r=1}^N\left(\delta_r^{(3),(k)}w_{ri}^{(3)}\right)+\beta\left(\frac{(1-\rho)}{(1-\hat{\rho}_i)}-\frac{\rho}{\hat{\rho}_i}\right)\right)\qquad\Delta w_{ij}^{(2)}=\eta\sum_{k=1}^R\delta_i^{(2),(k)}x_j^{(k)}$$

Schema

Initialize weights
(both 2nd and 3rd layers)

FORWARD step

Compute neuron outputs → $\left\{ y_i^{(k)} \right\}, \left\{ z_i^{(k)} \right\}$

Compute average activation → $\hat{\rho}_i = \frac{1}{R}\sum_k^R y_i^{(k)}$

BACKWARD step

Update neural weights → Δw_{ij}

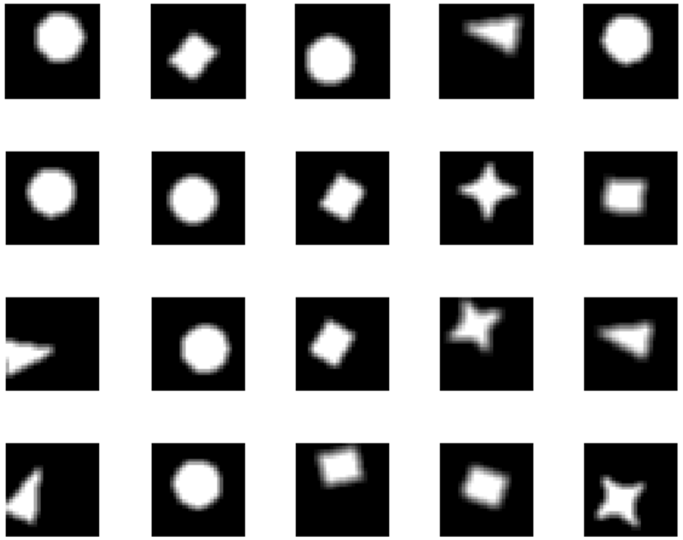
Iteration #1

……….

……….

……….

Training example



Training dataset

1000 labeled images

1 0 0 0: circle
0 1 0 0: triangle
0 0 1 0: rectangle
0 0 0 1: star

Image size: 30×30 pixel

Training example

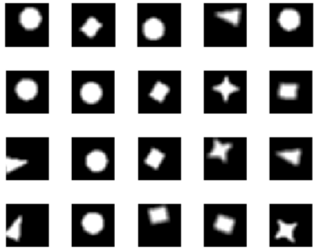
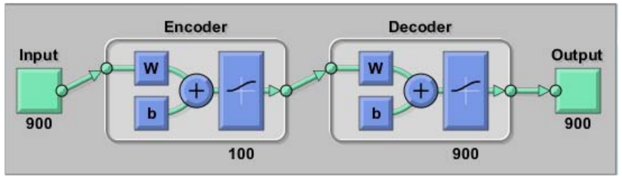
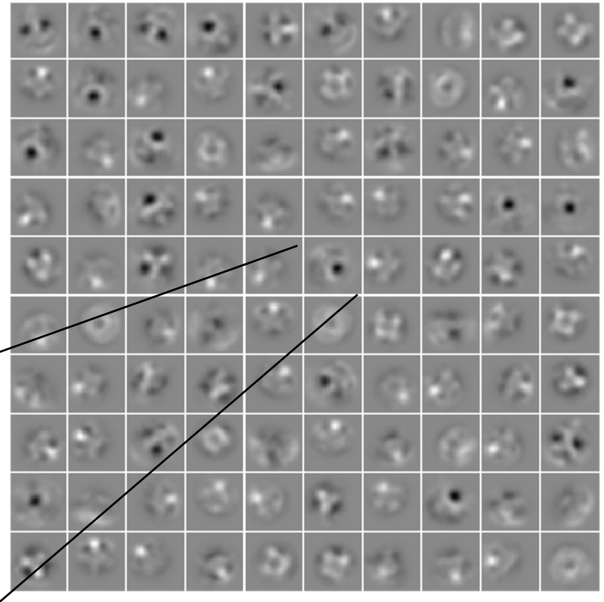
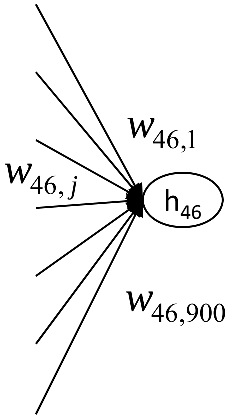


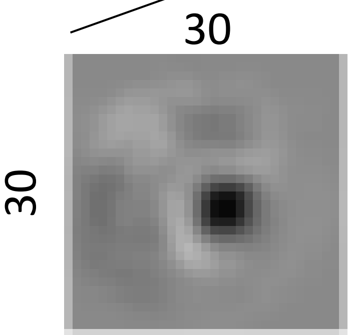
Image size: 30×30 (900 pixels)
Hidden units: 100
Training set: 1000 labeled images
β: 4
ρ: 0.15



HIDDEN LAYER WEIGHTS

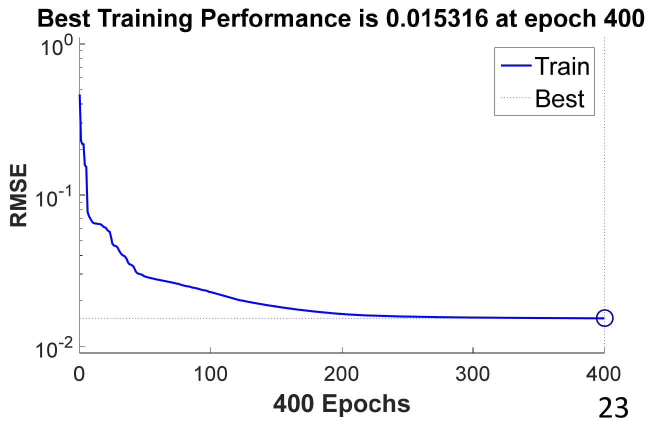






46th hidden neuron

Each weight is mapped to a gray value



Training example

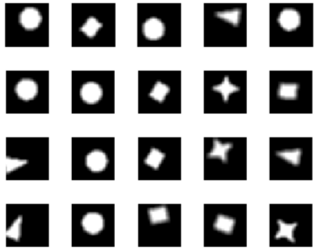
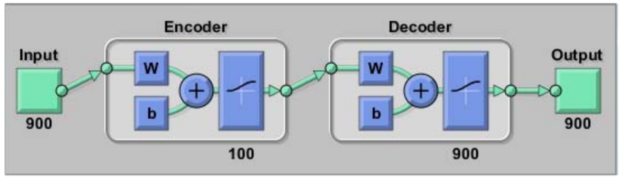


Image size: 30×30 (900 pixels)
Hidden units: 100
Training set: 1000 labeled images
β: 4
ρ: 0.05



HIDDEN LAYER WEIGHTS

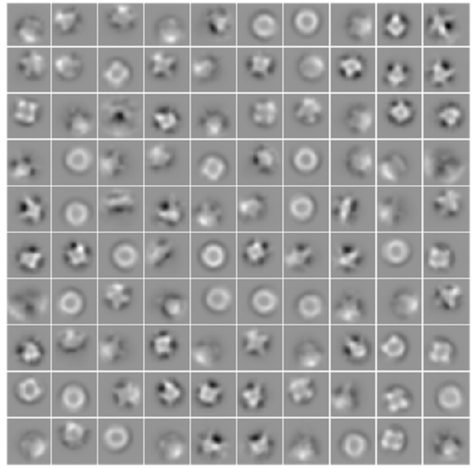
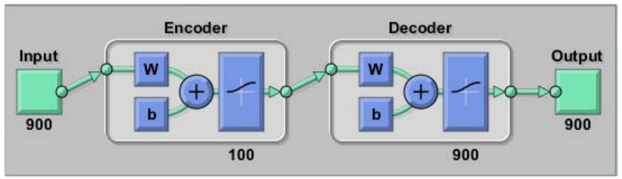
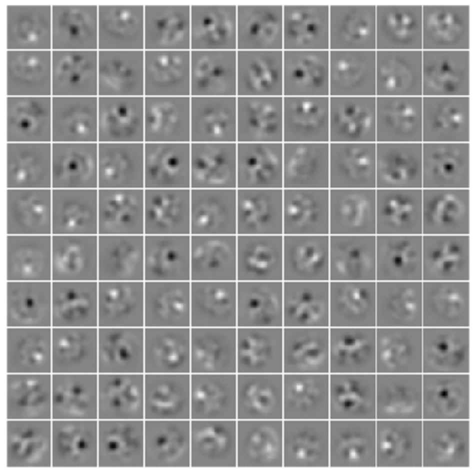


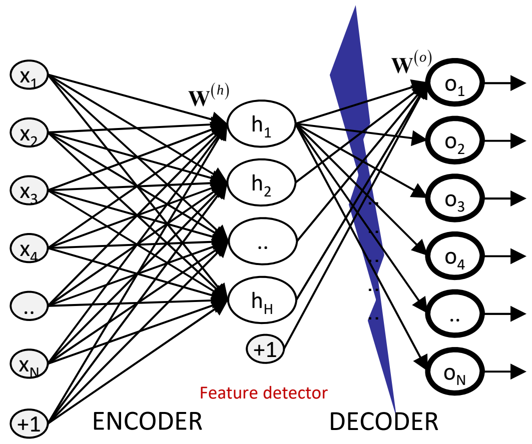
Image size: 30×30 (900 pixels)
Hidden units: 100
Training set: 1000 labeled images
β: 4
ρ: 0.25

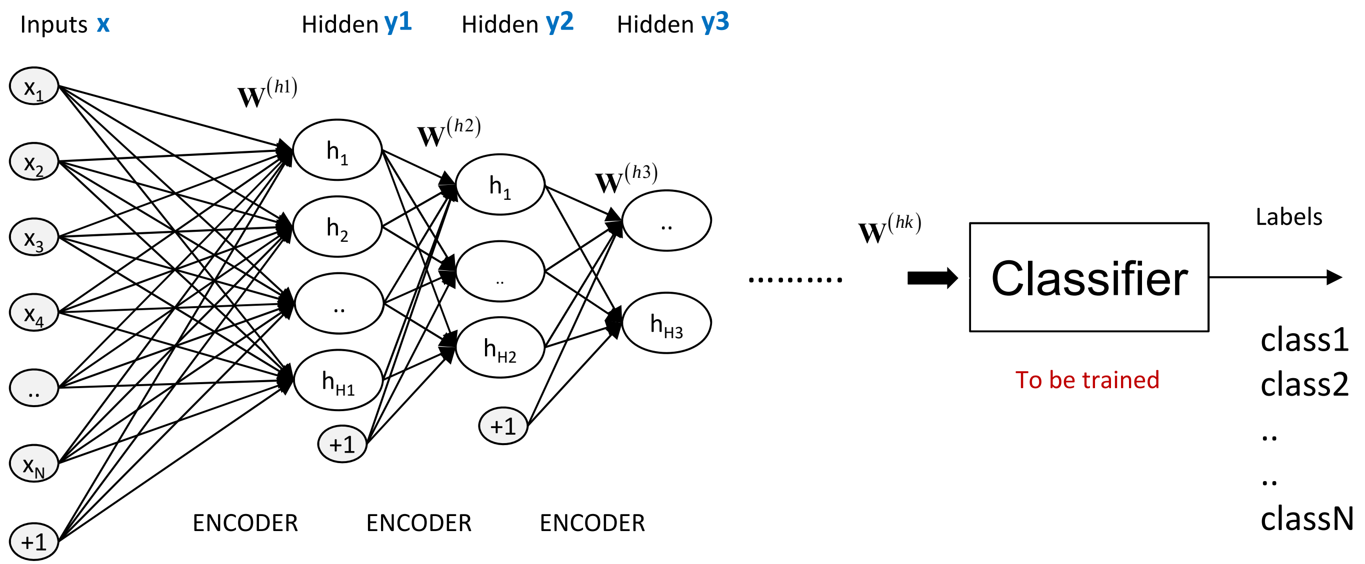


HIDDEN LAYER WEIGHTS

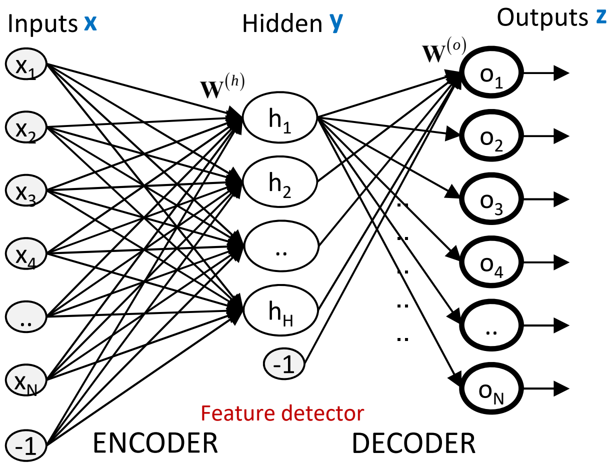


Autoencoder network cascade





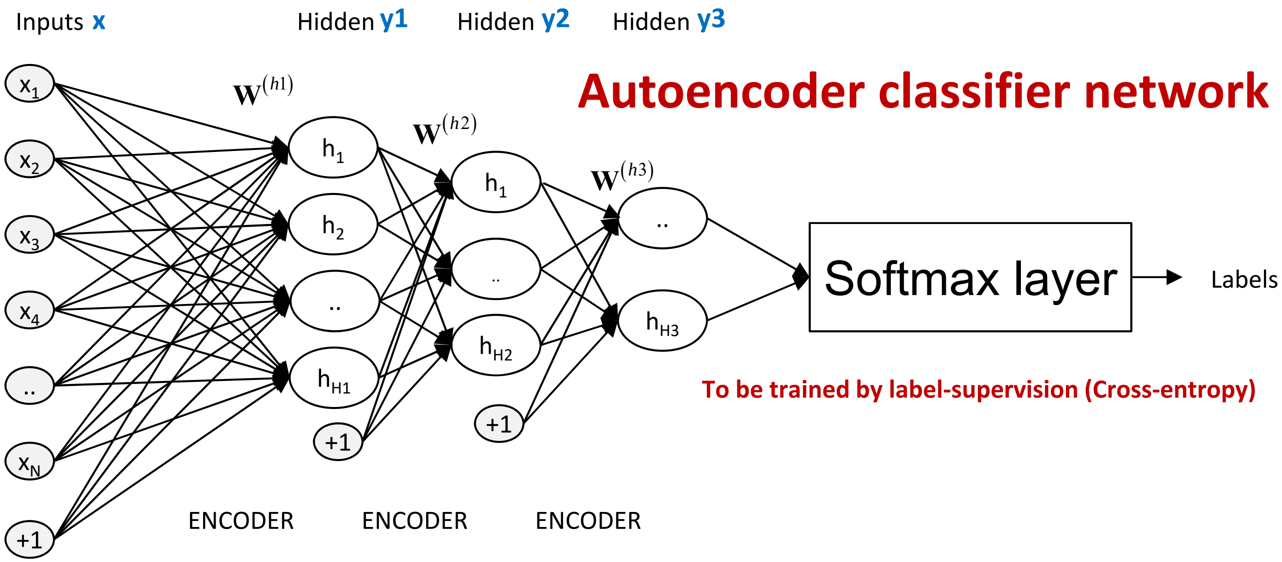
Autoencoder unit



To be trained by self-supervision (RMSE)

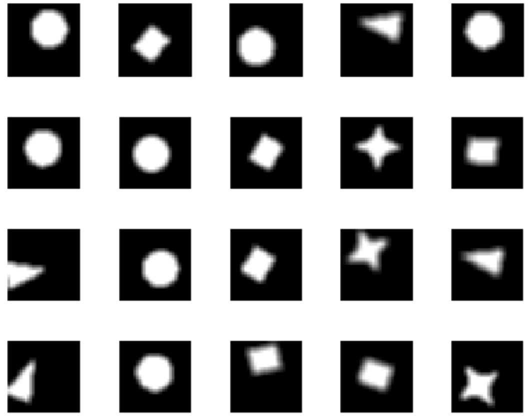
Synthesis

Autoencoder classifier network



To be trained by label-supervision (Cross-entropy)

Shape classifier with the 2 autoencoder layers

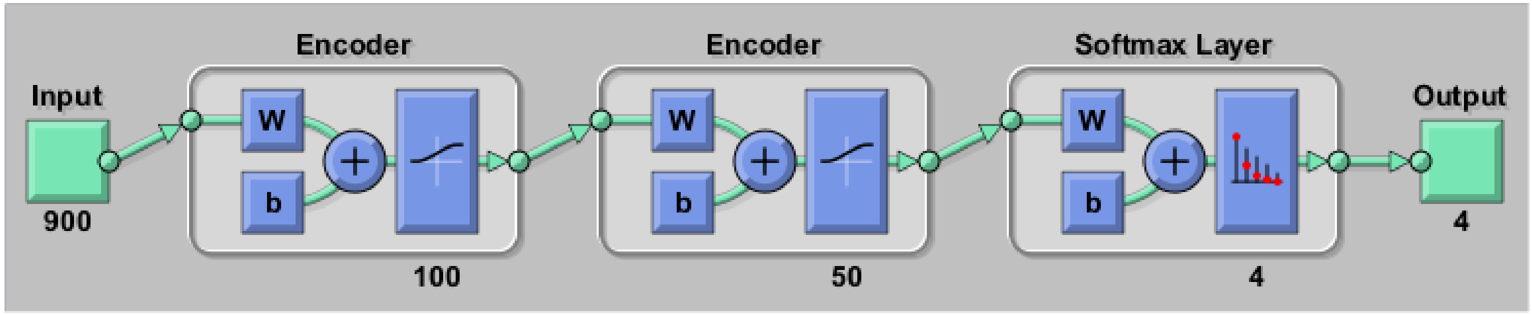


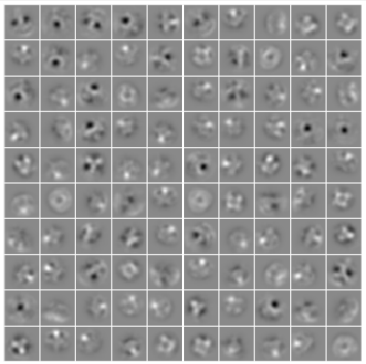
900 pixels

Layer 1
Hidden units: 100

Layer 2
Hidden units: 50

Layer 3
Softmax





1 0 0 0: circle
0 1 0 0: triangle
0 0 1 0: rectangle
0 0 0 1: star

Summary of notation

$\mathbf{x}$	Input vector of the autoencoder network (training example), $\mathbf{x} \in \Re^N$
x_i	Feature of a training example **x** (*i*-th input component)
$\mathbf{z}$	Output vector of the autoencoder network, $\mathbf{z} \in \Re^N$
z_i	Feature of an output vector **z** (*i*-th output component)
$\mathbf{y}$	Output vector of the hidden layer, $\mathbf{y} \in \Re^H$
y_i	Feature of an output vector **y** (*i*-th hidden neuron component)
$\mathbf{W}^{(l)}$	Weight matrix of the layer *l*
$w_{ij}^{(l)}$	Weight factor associated with the connection between unit *j* in the layer *l*, and unit *i* in the layer *l*+1
ρ	Sparsity parameter, which specifies the desired level of sparsity of the activation in the hidden layer
$\hat{\rho}_i$	The average activation of hidden unit *i*
α	Weight (in the error function) of the norm constraint of the neural weights
β	Weight (in the sparse autoencoder error function) of the sparsity penalty term