Machine Learning 10-601

Tom M. Mitchell
Machine Learning Department
Carnegie Mellon University

April 15, 2015

Today:
- Artificial neural networks
- Backpropagation
- Recurrent networks
- Convolutional networks
- Deep belief networks
- Deep Boltzman machines

Reading:
- Mitchell: Chapter 4
- Bishop: Chapter 5
- Quoc Le tutorial:
- Ruslan Salakhutdinov tutorial:

Artificial Neural Networks to learn f: X → Y

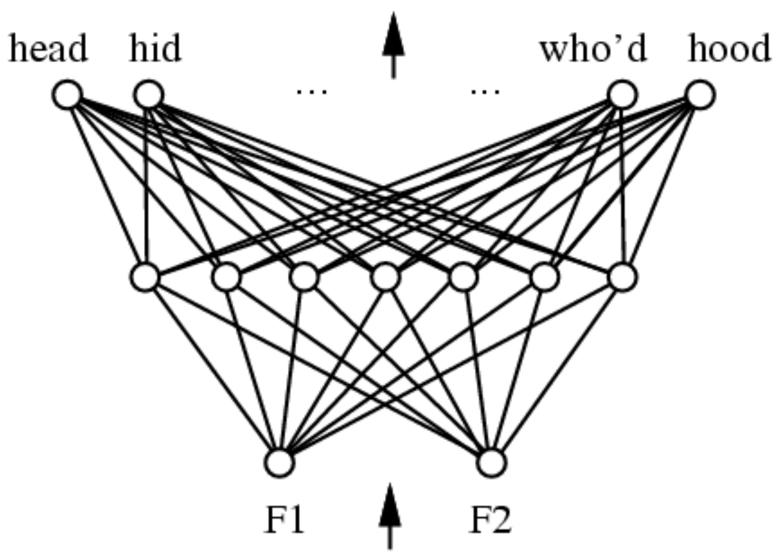
- f might be non-linear function
- X (vector of) continuous and/or discrete vars
- Y (vector of) continuous and/or discrete vars

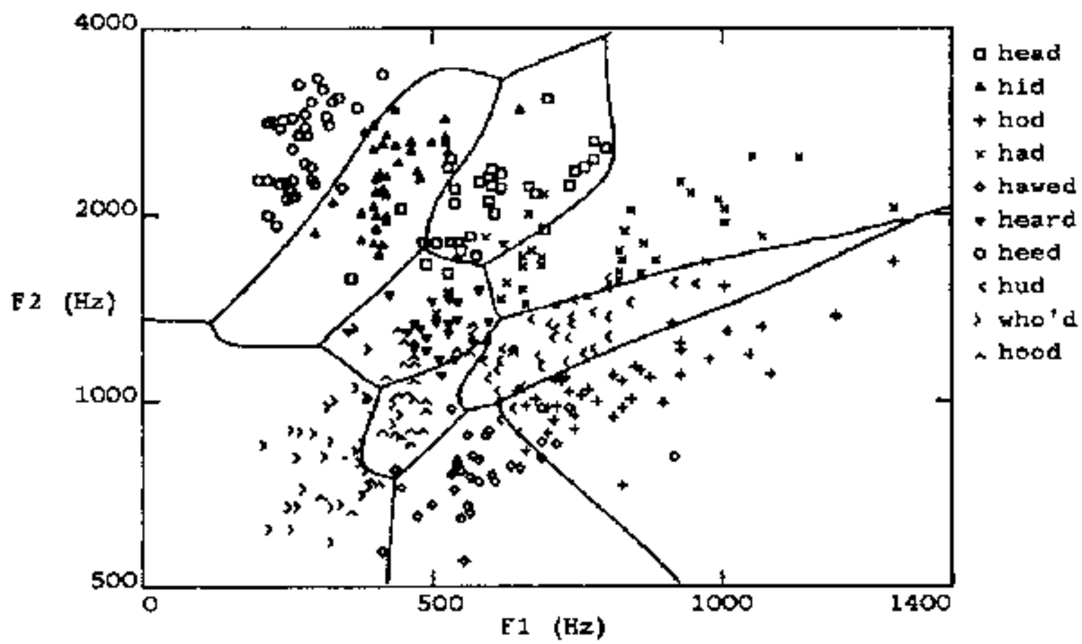
- Represent f by ***network*** of logistic units
- Each unit is a logistic function

$$unit\ output = \frac{1}{1 + exp(w_0 + \sum_i w_i x_i)}$$

- MLE: train weights of all units to minimize sum of squared errors of predicted network outputs
- MAP: train to minimize sum of squared errors plus weight magnitudes

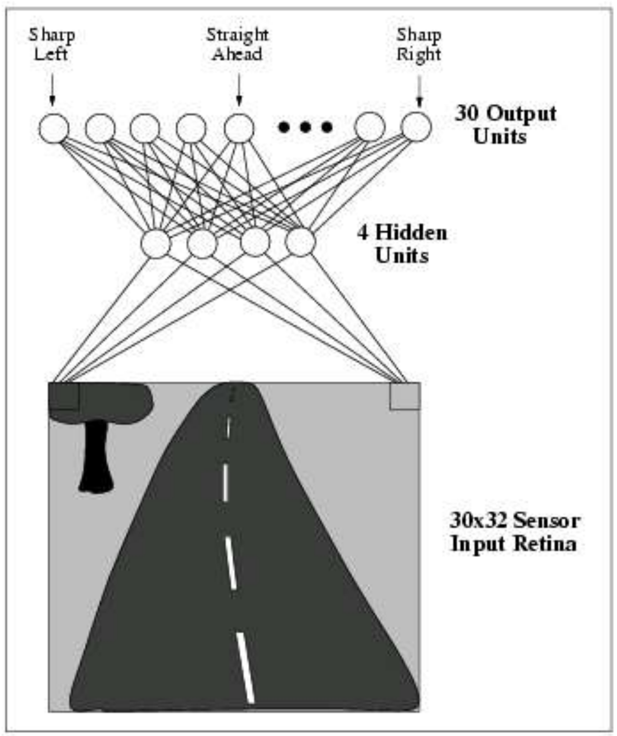
Multilayer Networks of Sigmoid Units

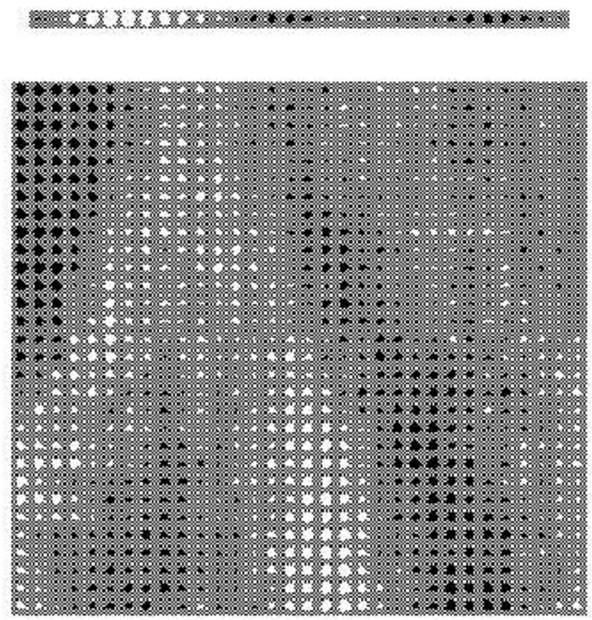




ALVINN

[Pomerleau 1993]





Connectionist Models

Consider humans:

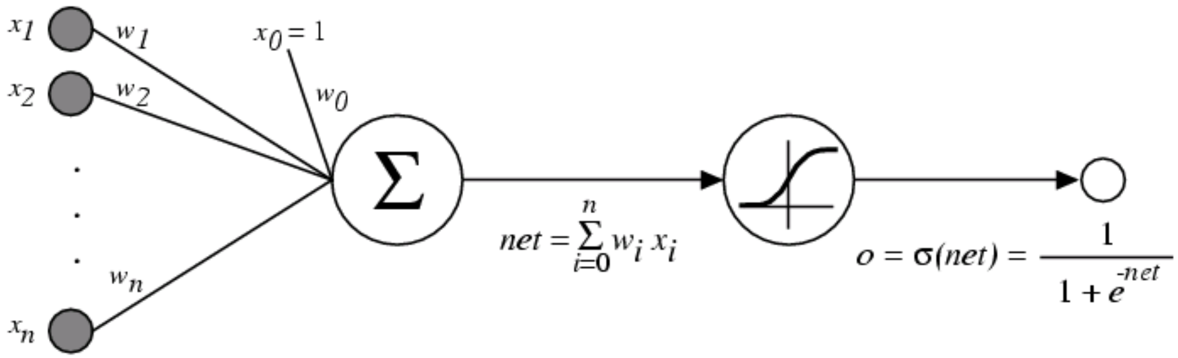
- Neuron switching time ˜ .001 second
- Number of neurons ˜ 10^{10}
- Connections per neuron ˜ 10^{4-5}
- Scene recognition time ˜ .1 second
- 100 inference steps doesn't seem like enough

$\rightarrow$ much parallel computation

Properties of artificial neural nets (ANN's):

- Many neuron-like threshold switching units
- Many weighted interconnections among units
- Highly parallel, distributed process

Sigmoid Unit



$\sigma(x)$ is the sigmoid function

$$\frac{1}{1 + e^{-x}}$$

Nice property: $\frac{d\sigma(x)}{dx} = \sigma(x)(1 - \sigma(x))$

We can derive gradient decent rules to train

- One sigmoid unit
- *Multilayer networks* of sigmoid units $\rightarrow$ Backpropagation

M(C)LE Training for Neural Networks

- Consider regression problem f:X→Y , for scalar Y

$y = f(x) + \varepsilon$

assume noise $N(0,\sigma_\varepsilon)$, iid

deterministic

- Let's maximize the conditional data likelihood

$$W \leftarrow \arg\max_W \ \ln \prod_l P(Y^l | X^l, W)$$

$$W \leftarrow \arg\min_W \ \sum_l (y^l - \widehat{f}(x^l))^2$$

Learned neural network

MAP Training for Neural Networks

- Consider regression problem f:X→Y , for scalar Y

$y = f(x) + \varepsilon$

noise $N(0,\sigma_\varepsilon)$

deterministic

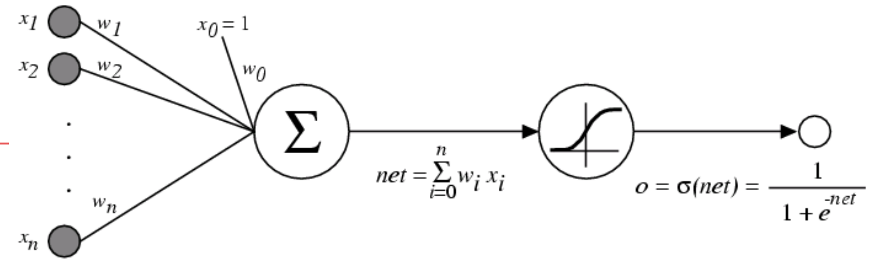
Gaussian P(W) = N(0,σI)

$$W \leftarrow \arg\max_W \ \ln \ P(W) \prod_l P(Y^l | X^l, W)$$

$$W \leftarrow \arg\min_W \ \left[c \sum_i w_i^2 \right] + \left[\sum_l (y^l - \hat{f}(x^l))^2 \right]$$

$\ln P(W) \leftrightarrow c \sum_i w_i^2$

Error Gradient for a Sigmoid Unit



$$
\begin{aligned}
\frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i} \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2 \\
&= \frac{1}{2} \sum_d \frac{\partial}{\partial w_i} (t_d - o_d)^2 \\
&= \frac{1}{2} \sum_d 2(t_d - o_d) \frac{\partial}{\partial w_i} (t_d - o_d) \\
&= \sum_d (t_d - o_d) \left(-\frac{\partial o_d}{\partial w_i} \right) \\
&= -\sum_d (t_d - o_d) \frac{\partial o_d}{\partial net_d} \frac{\partial net_d}{\partial w_i}
\end{aligned}
$$

But we know:

$$\frac{\partial o_d}{\partial net_d} = \frac{\partial \sigma(net_d)}{\partial net_d} = o_d(1 - o_d)$$

$$\frac{\partial net_d}{\partial w_i} = \frac{\partial (\vec{w} \cdot \vec{x}_d)}{\partial w_i} = x_{i,d}$$

So:

$$\frac{\partial E}{\partial w_i} = -\sum_{d \in D} (t_d - o_d) o_d (1 - o_d) x_{i,d}$$

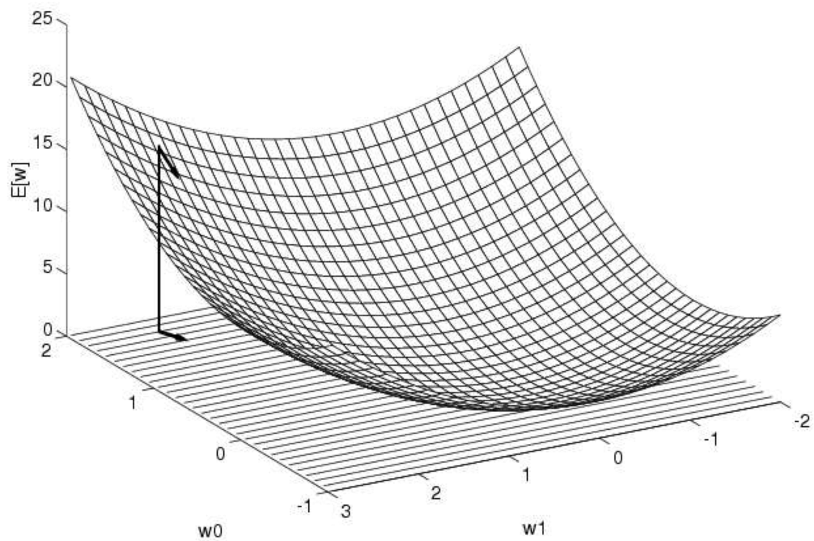
x_d = input

t_d = target output

o_d = observed unit output

w_i = weight i

Gradient Descent



Gradient

$$\nabla E[\vec{w}] \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \cdots \frac{\partial E}{\partial w_n}\right]$$

Training rule:

$$\Delta \vec{w} = -\eta \nabla E[\vec{w}]$$

i.e.,

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

Incremental (Stochastic) Gradient Descent

Batch mode Gradient Descent:
Do until satisfied

1. Compute the gradient $\nabla E_D[\vec{w}]$
2. $\vec{w} \leftarrow \vec{w} - \eta \nabla E_D[\vec{w}]$

Incremental mode Gradient Descent:
Do until satisfied

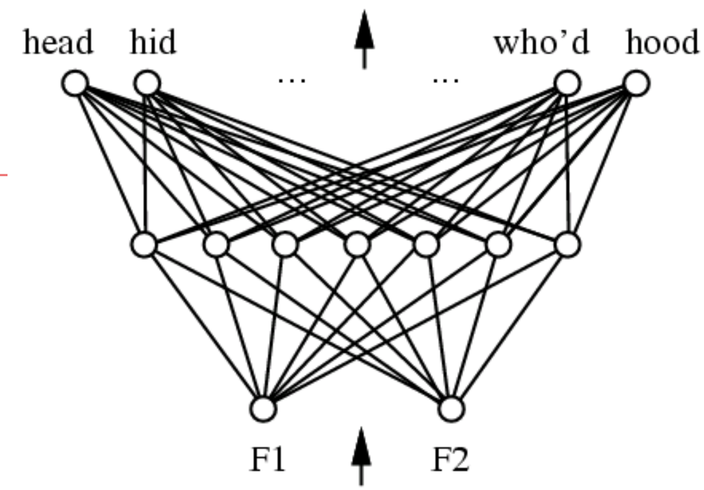
- For each training example d in D
  1. Compute the gradient $\nabla E_d[\vec{w}]$
  2. $\vec{w} \leftarrow \vec{w} - \eta \nabla E_d[\vec{w}]$

$$E_D[\vec{w}] \equiv \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

$$E_d[\vec{w}] \equiv \frac{1}{2} (t_d - o_d)^2$$

Incremental Gradient Descent can approximate *Batch Gradient Descent* arbitrarily closely if η made small enough

Backpropagation Algorithm (MLE)



Initialize all weights to small random numbers.
Until satisfied, Do

- For each training example, Do
  1. Input the training example to the network and compute the network outputs
  2. For each output unit k

  $$\delta_k \leftarrow o_k(1-o_k)(t_k-o_k)$$

  3. For each hidden unit h

  $$\delta_h \leftarrow o_h(1-o_h) \sum_{k \in outputs} w_{h,k}\delta_k$$

  4. Update each network weight $w_{i,j}$

  $$w_{i,j} \leftarrow w_{i,j} + \Delta w_{i,j}$$

  where

  $$\Delta w_{i,j} = \eta \delta_j x_i$$

x_d = input

t_d = target output

o_d = observed unit output

w_{ij} = wt from i to j

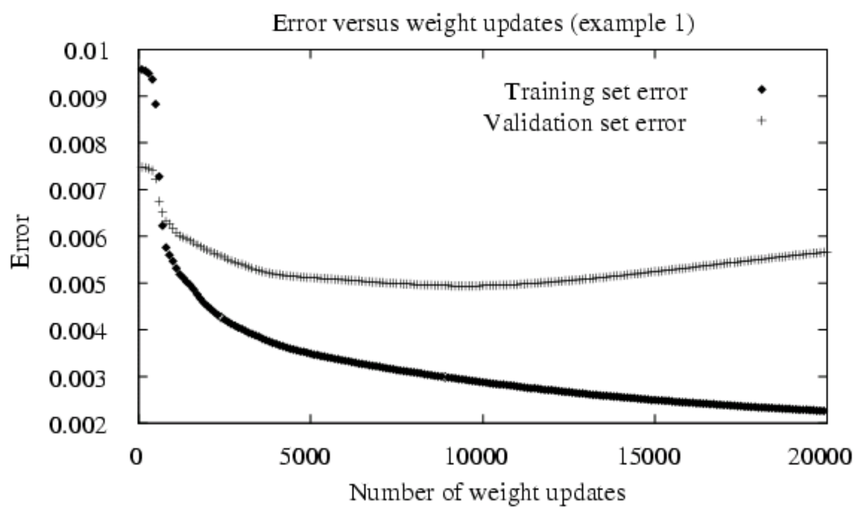
More on Backpropagation

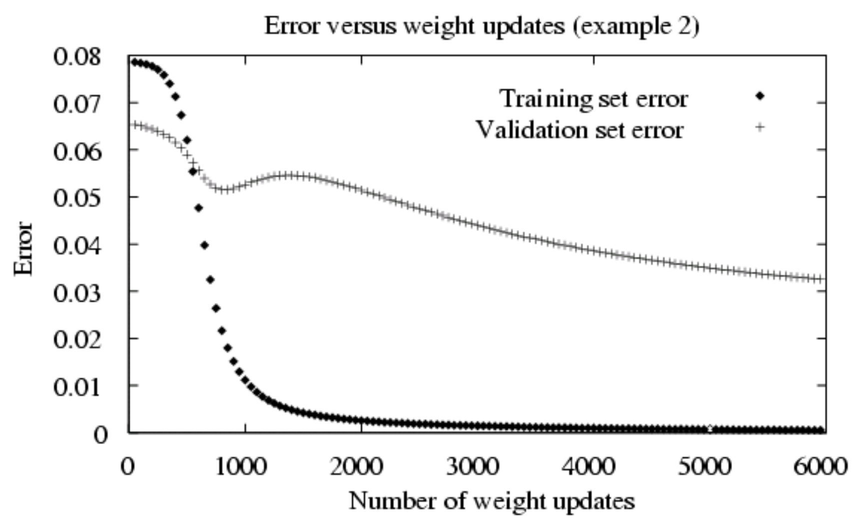
- Gradient descent over entire *network* weight vector
- Easily generalized to arbitrary directed graphs
- Will find a local, not necessarily global error minimum
  - In practice, often works well (can run multiple times)
- Often include weight *momentum* α

$$\Delta w_{i,j}(n) = \eta \delta_j x_{i,j} + \alpha \Delta w_{i,j}(n-1)$$

- Minimizes error over *training* examples
  - Will it generalize well to subsequent examples?
- Training can take thousands of iterations $\rightarrow$ slow!
- Using network after training is very fast

Overfitting in ANNs





Expressive Capabilities of ANNs

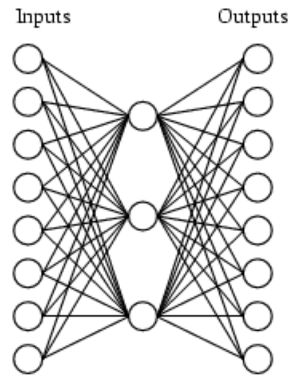
Boolean functions:

- Every boolean function can be represented by network with single hidden layer
- but might require exponential (in number of inputs) hidden units

Continuous functions:

- Every bounded continuous function can be approximated with arbitrarily small error, by network with one hidden layer [Cybenko 1989; Hornik et al. 1989]
- Any function can be approximated to arbitrary accuracy by a network with two hidden layers [Cybenko 1988].

Learning Hidden Layer Representations



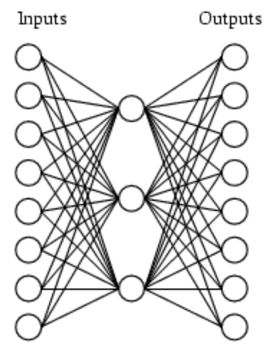
A target function:

Input		Output
10000000	$\rightarrow$	10000000
01000000	$\rightarrow$	01000000
00100000	$\rightarrow$	00100000
00010000	$\rightarrow$	00010000
00001000	$\rightarrow$	00001000
00000100	$\rightarrow$	00000100
00000010	$\rightarrow$	00000010
00000001	$\rightarrow$	00000001

Can this be learned??

Learning Hidden Layer Representations

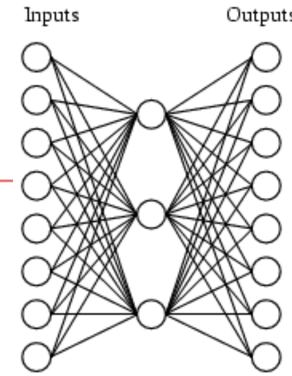
A network:

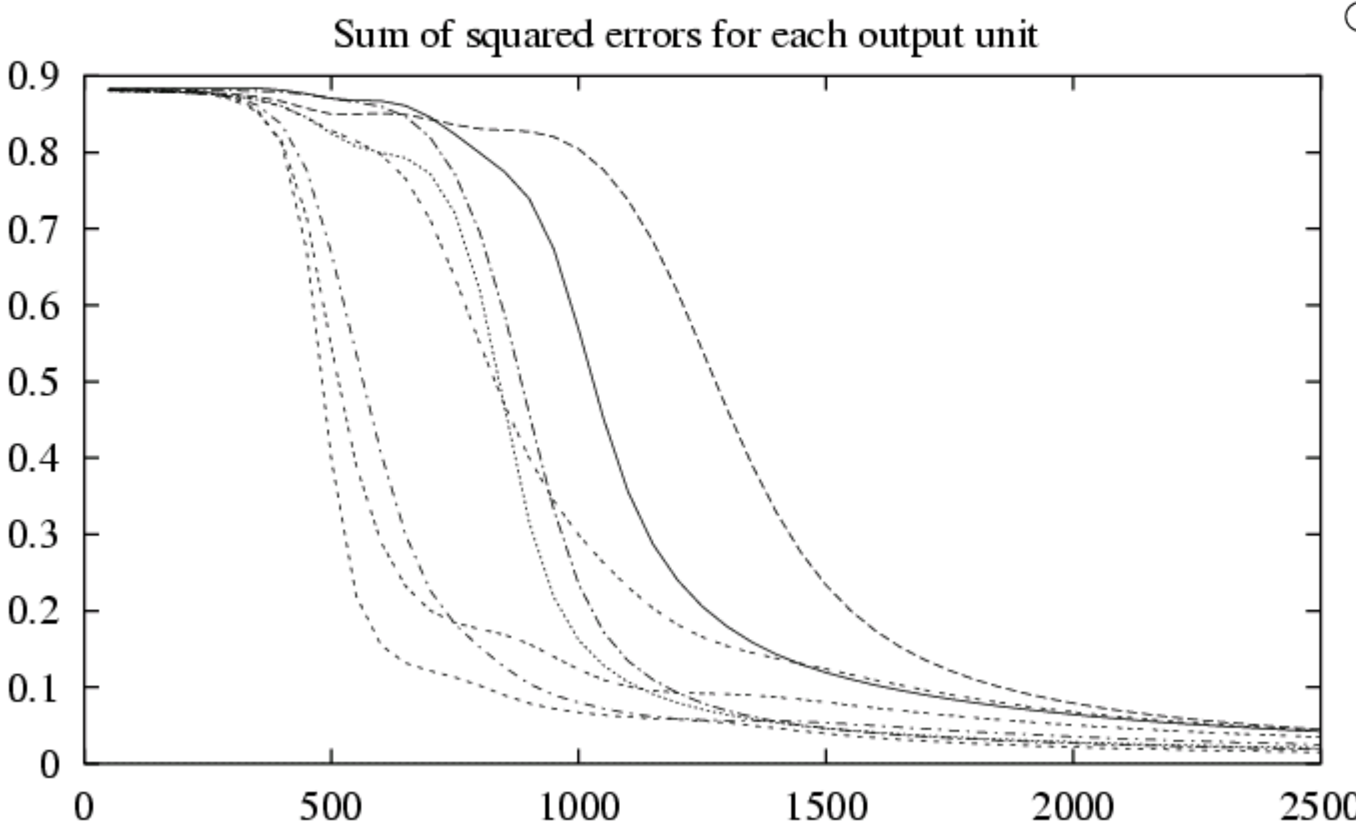


Learned hidden layer representation:

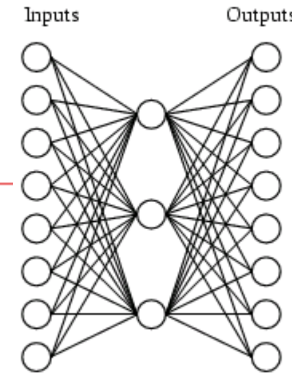
Input		Hidden Values				Output
10000000	$\rightarrow$	.89	.04	.08	$\rightarrow$	10000000
01000000	$\rightarrow$	.01	.11	.88	$\rightarrow$	01000000
00100000	$\rightarrow$	.01	.97	.27	$\rightarrow$	00100000
00010000	$\rightarrow$	.99	.97	.71	$\rightarrow$	00010000
00001000	$\rightarrow$	.03	.05	.02	$\rightarrow$	00001000
00000100	$\rightarrow$	.22	.99	.99	$\rightarrow$	00000100
00000010	$\rightarrow$	.80	.01	.98	$\rightarrow$	00000010
00000001	$\rightarrow$	.60	.94	.01	$\rightarrow$	00000001

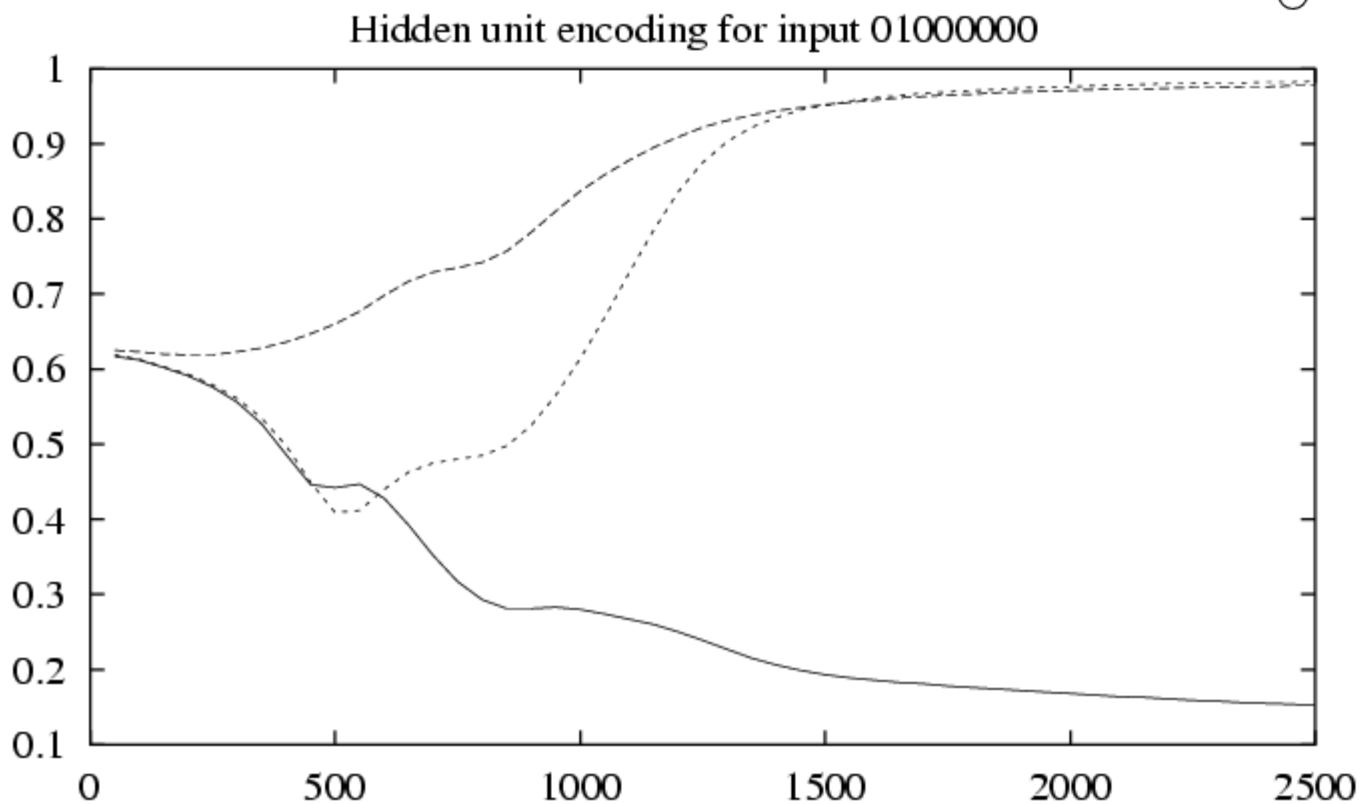
Training


Inputs
Outputs

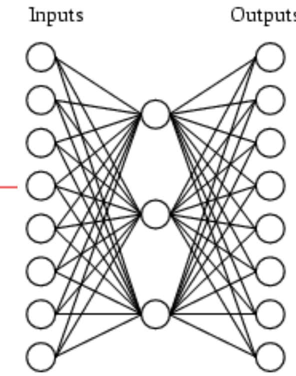

Sum of squared errors for each output unit
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
500
1000
1500
2000
2500

Training

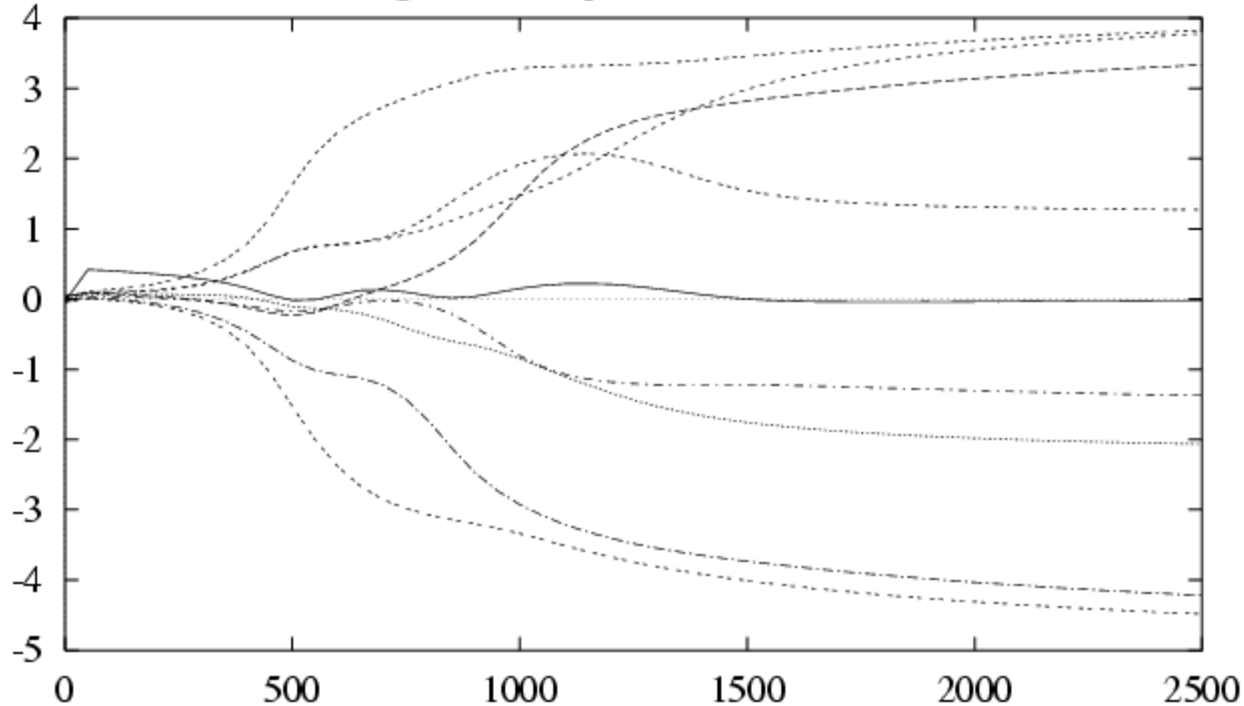

Inputs
Outputs


Hidden unit encoding for input 01000000
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
500
1000
1500
2000
2500

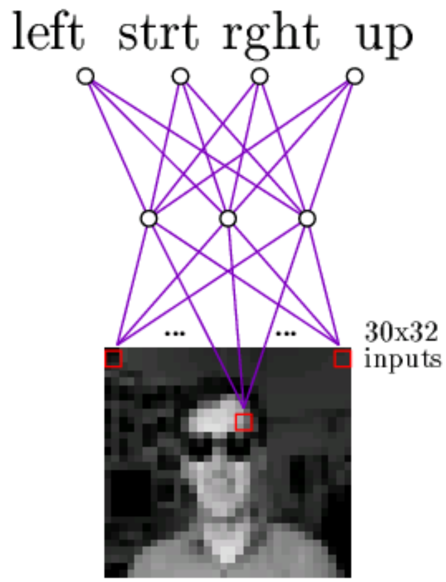
Training



Weights from inputs to one hidden unit



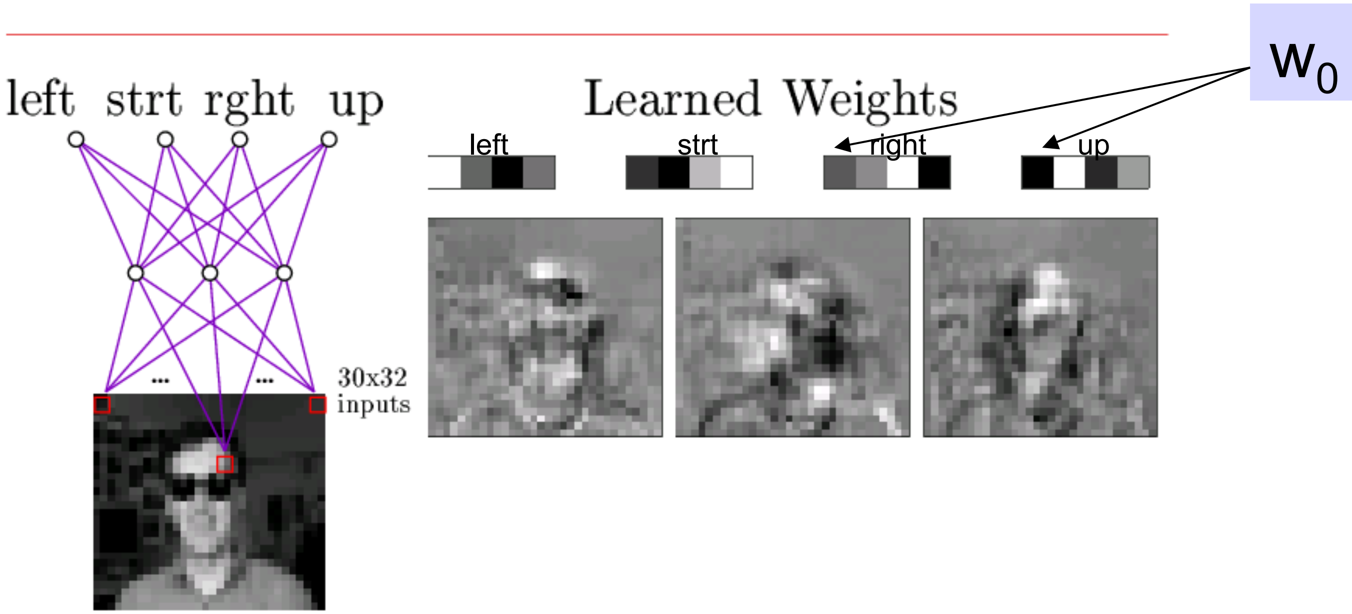
Neural Nets for Face Recognition



Typical input images

90% accurate learning head pose, and recognizing 1-of-20 faces

Learned Hidden Unit Weights

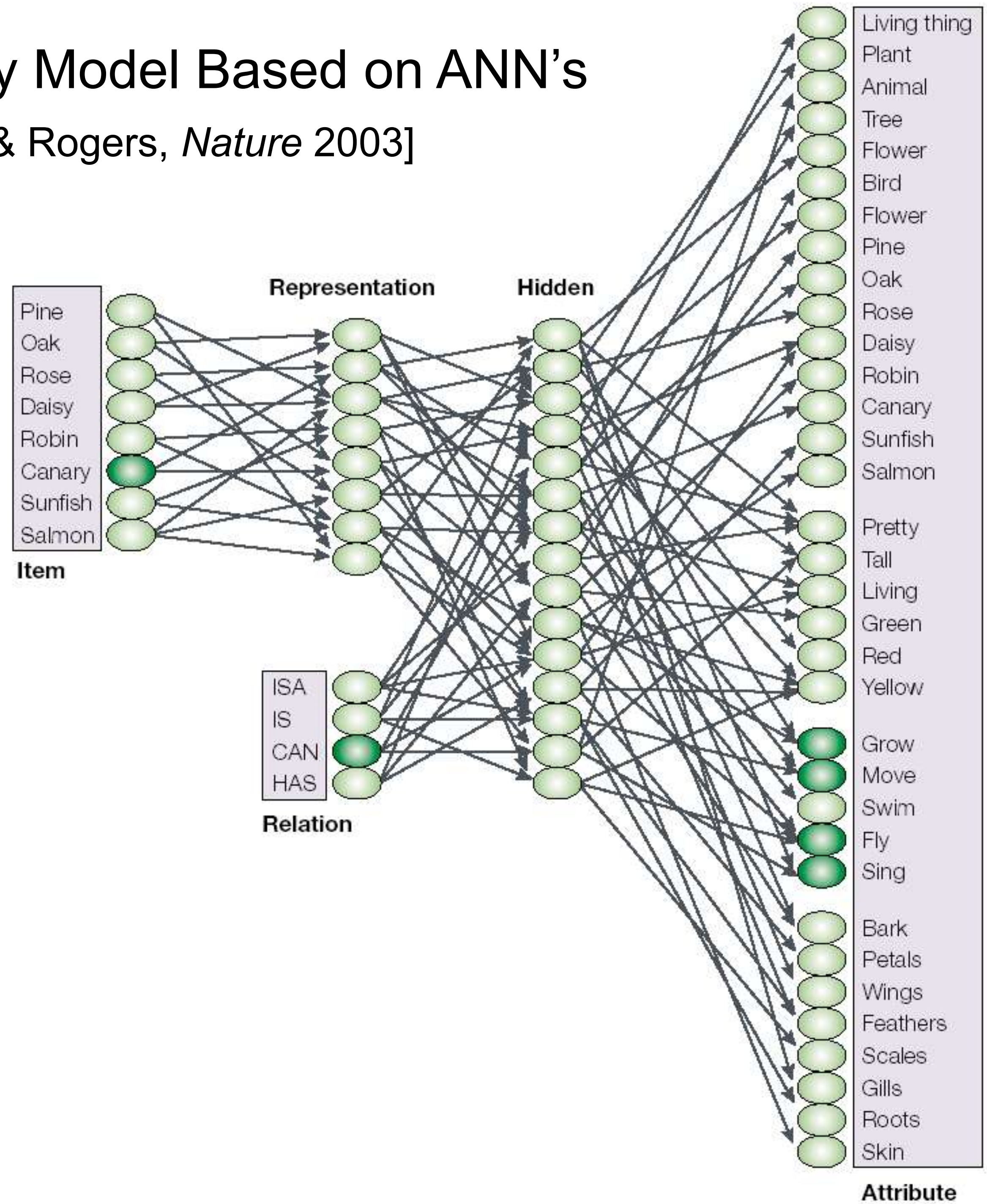


Typical input images

http://www.cs.cmu.edu/~tom/faces.html

Semantic Memory Model Based on ANN's

[McClelland & Rogers, *Nature* 2003]



No hierarchy given.

Train with assertions,
e.g., Can(Canary,Fly)

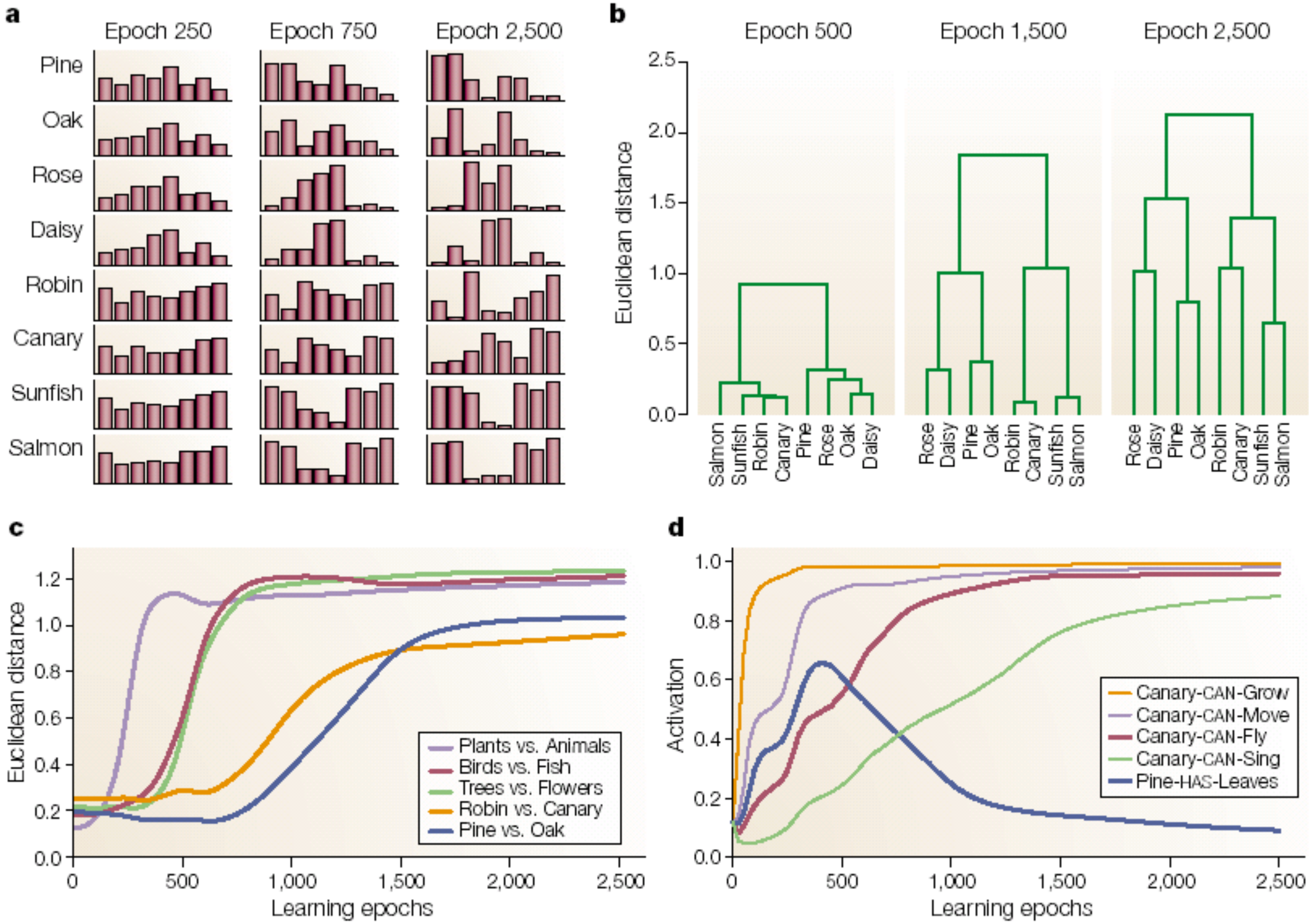


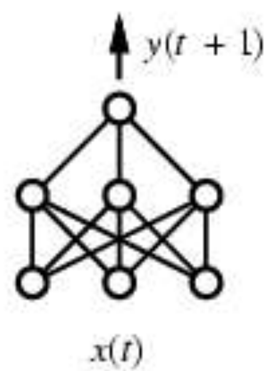
Figure 4 | **The process of differentiation of conceptual representations.** The representations are those seen in the feedforward network model shown in FIG. 3. **a** | Acquired patterns of activation that represent the eight objects in the training set at three points in the learning process (epochs 250, 750 and 2,500). Early in learning, the patterns are undifferentiated; the first difference to appear is between plants and animals. Later, the patterns show clear differentiation at both the superordinate (plant–animal) and intermediate (bird–fish/tree–flower) levels. Finally, the individual concepts are differentiated, but the overall hierarchical organization of the similarity structure remains. **b** | A standard hierarchical clustering analysis program has been used to visualize the similarity structure in the

Training Networks on Time Series

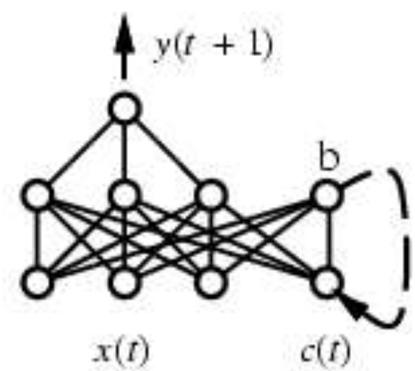
- Suppose we want to predict next state of world
  - and it depends on history of unknown length
  - e.g., robot with forward-facing sensors trying to predict next sensor reading as it moves and turns

Recurrent Networks: Time Series

- Suppose we want to predict next state of world
  - and it depends on history of unknown length
  - e.g., robot with forward-facing sensors trying to predict next sensor reading as it moves and turns
- Idea: use hidden layer in network to capture state history



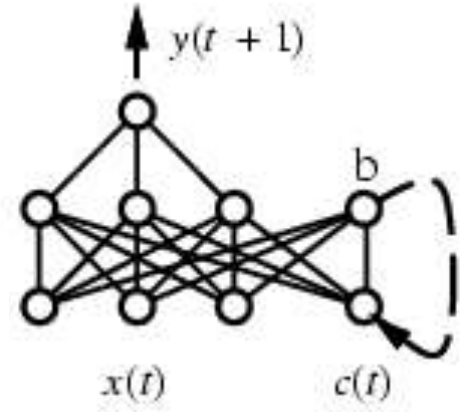
(a) Feedforward network

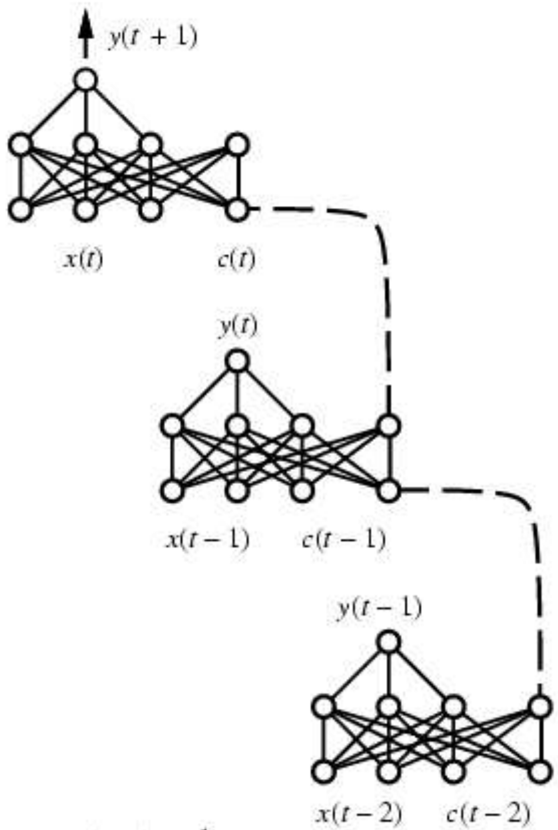


(b) Recurrent network

Recurrent Networks on Time Series

How can we train recurrent net??

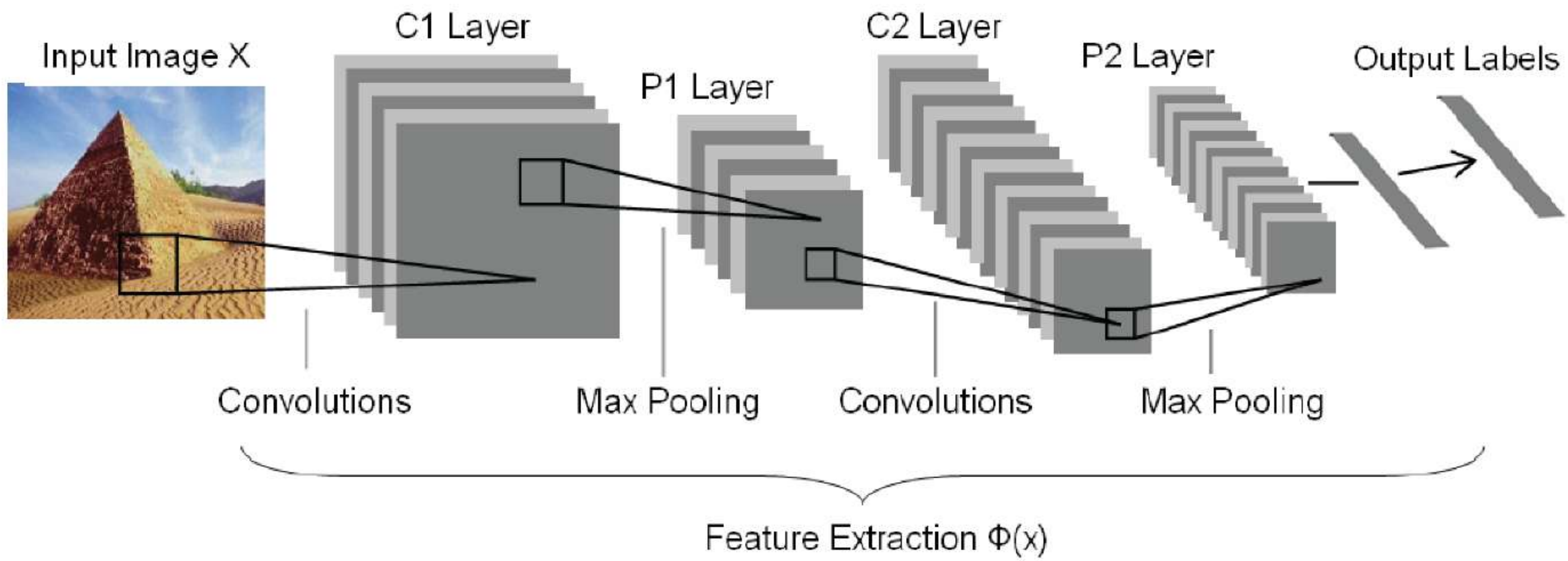




(c) Recurrent network
unfolded in time

Convolutional Neural Nets for Image Recognition

[Le Cun, 1992]



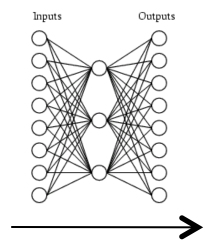
- specialized architecture: mix different types of units, not completely connected, motivated by primate visual cortex
- many shared parameters, stochastic gradient training
- very successful! now many specialized architectures for vision, speech, translation, …

Deep Belief Networks

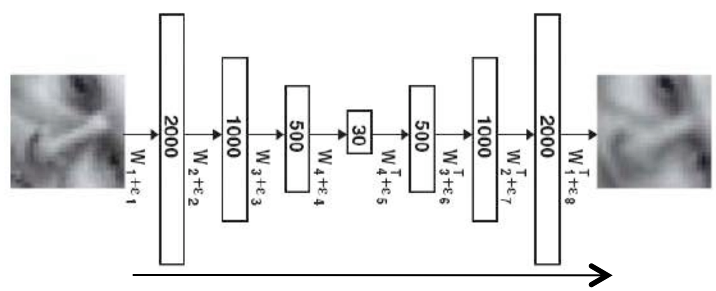
[Hinton & Salakhutdinov, 2006]

- Problem: training networks with many hidden layers doesn't work very well
  - local minima, very slow training if initialize with zero weights

- Deep belief networks
  - autoencoder networks to learn low dimensional encodings



  - but more layers, to learn better encodings



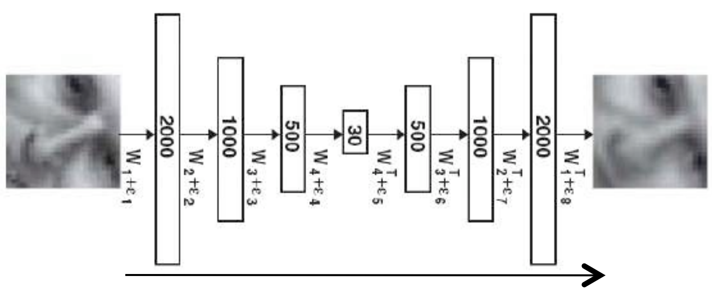
Deep Belief Networks

[Hinton & Salakhutdinov, 2006]

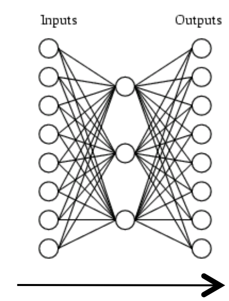
original image

reconstructed from 2000-1000-500-30 DBN

reconstructed from 2000-300, linear PCA



versus



[Hinton & Salakhutdinov, 2006]

Deep Belief Networks: Training

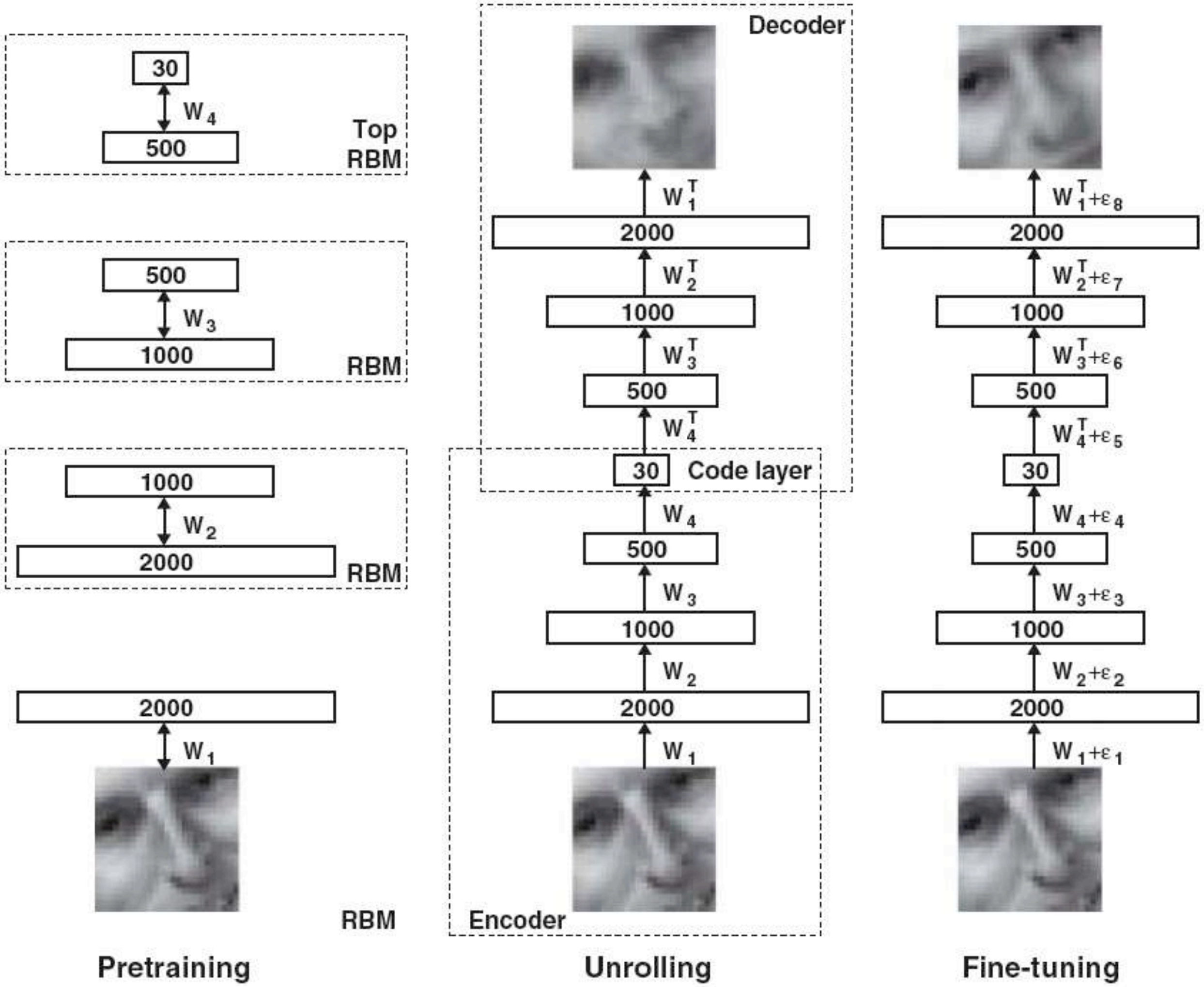


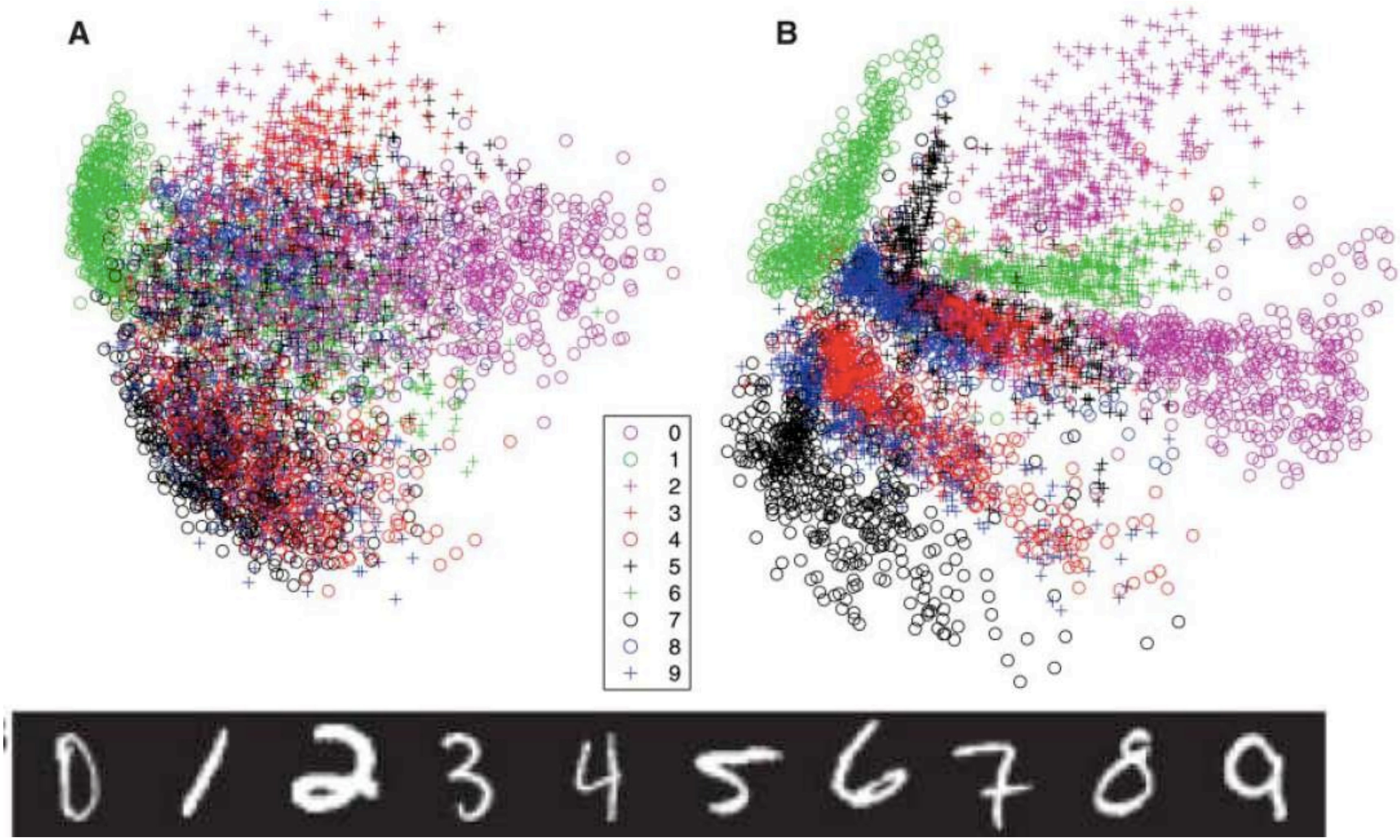
Fig. 1. Pretraining consists of learning a stack of restricted Boltzmann machines (RBMs), each having only one layer of feature detectors. The learned feature activations of one RBM are used as the "data" for training the next RBM in the stack. After the pretraining, the RBMs are "unrolled" to create a deep autoencoder, which is then fine-tuned using backpropagation of error derivatives.

Encoding of digit images in two dimensions

[Hinton & Salakhutdinov, 2006]

784-2 linear encoding (PCA)

784-1000-500-250-2 DBNet



Very Large Scale Use of DBN's

[Quoc Le, et al., *ICML*, 2012]

Data: 10 million 200x200 unlabeled images, sampled from YouTube

Training: use 1000 machines (16000 cores) for 1 week

Learned network: 3 multi-stage layers, 1.15 billion parameters

Achieves 15.8% (was 9.5%) accuracy classifying 1 of 20k ImageNet items

Real images that most excite the feature:

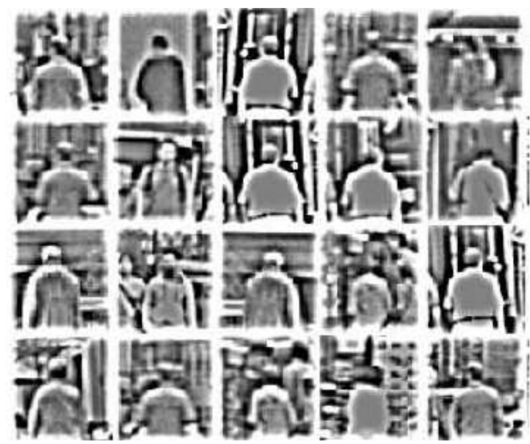
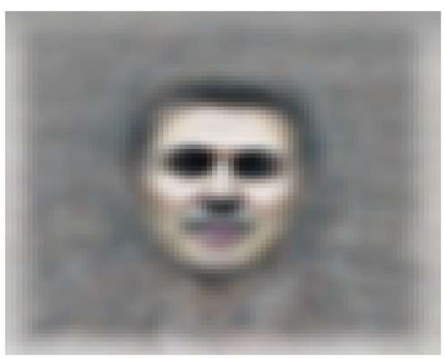
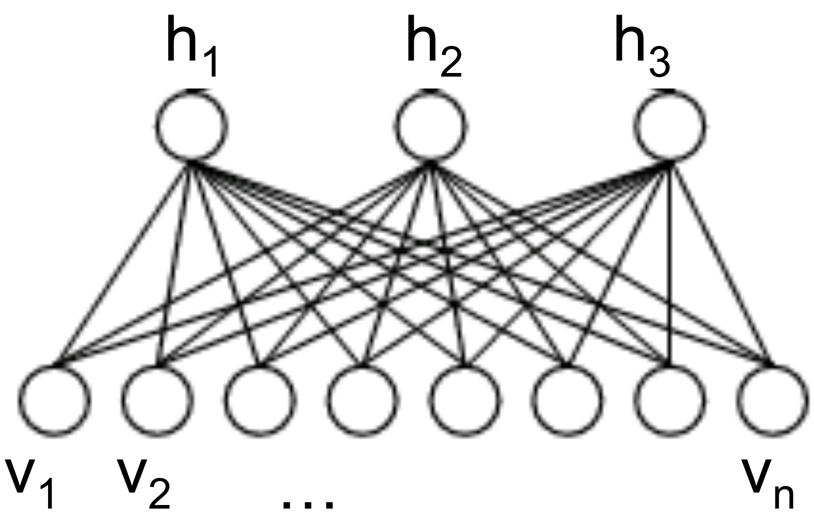


Image synthesized to most excite the feature:



Restricted Boltzman Machine

- Bipartite graph, logistic activation
- Inference: fill in any nodes, estimate other nodes
- consider v_i, h_j are boolean variables



$$P(h_j = 1|\mathbf{v}) = \frac{1}{1+\exp(\sum_i w_{ij} v_i)}$$

$$P(v_i = 1|\mathbf{h}) = \frac{1}{1+\exp(\sum_j w_{ij} h_j)}$$

Impact of Deep Learning

- Speech Recognition
- Computer Vision
- Recommender Systems
- Language Understanding
- Drug Discovery and Medical Image Analysis

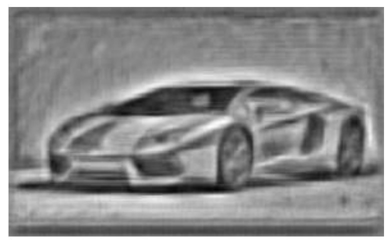
[Courtesy of R. Salakhutdinov]

Feature Representations: Traditionally

Data → Feature extraction → Learning algorithm

Object detection

Image

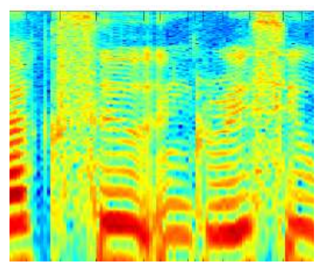


vision features

Recognition

Audio classification

Audio

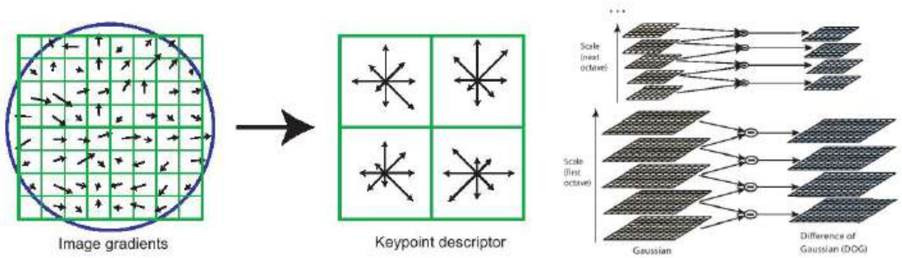


audio features

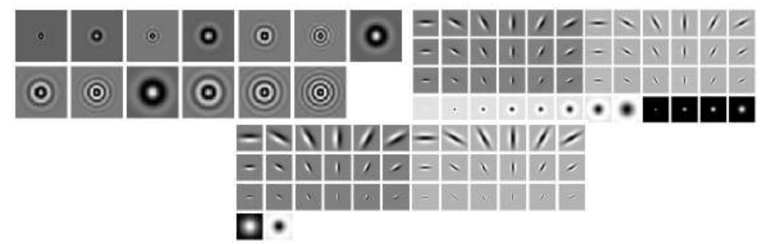
Speaker identification

[Courtesy of R. Salakhutdinov]

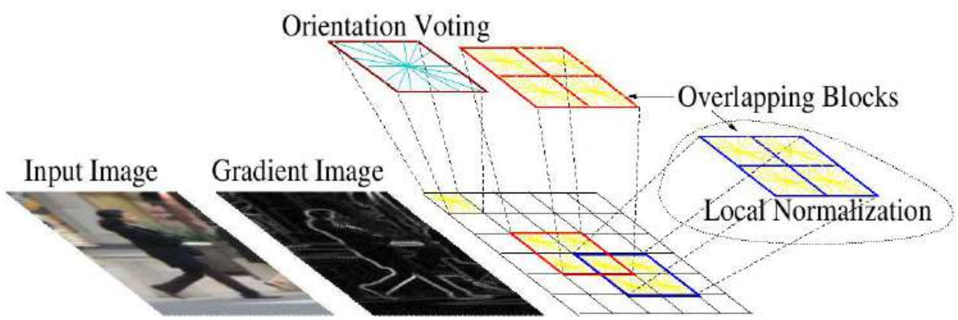
Computer Vision Features



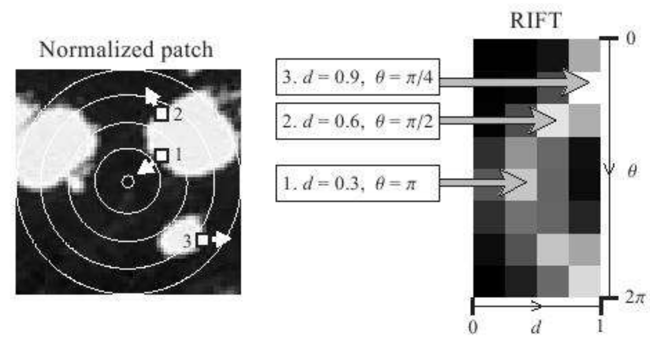
SIFT



Textons

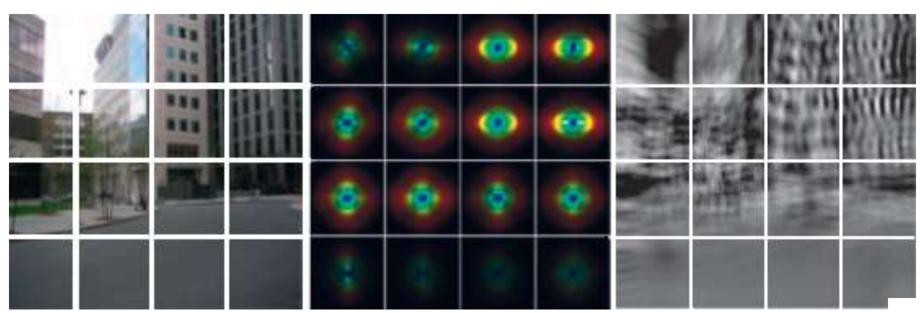


HoG



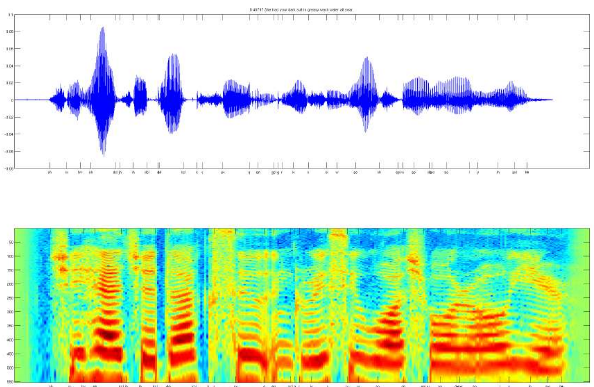
RIFT

GIST

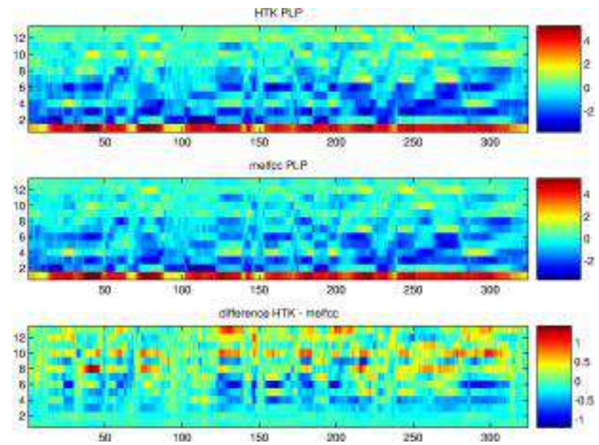


[Courtesy, R. Salakhutdinov]

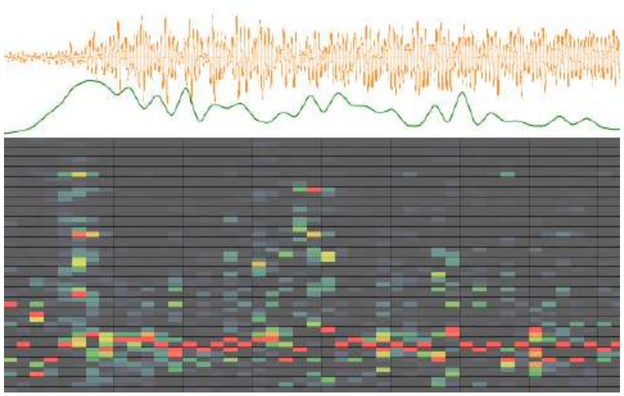
Audio Features



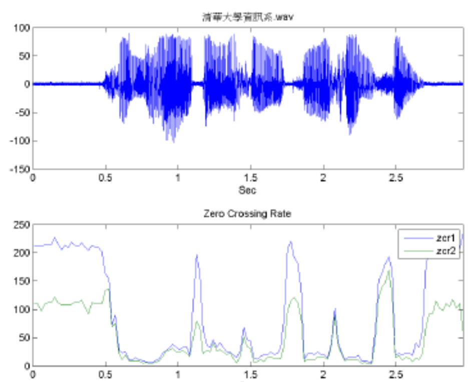
Spectrogram



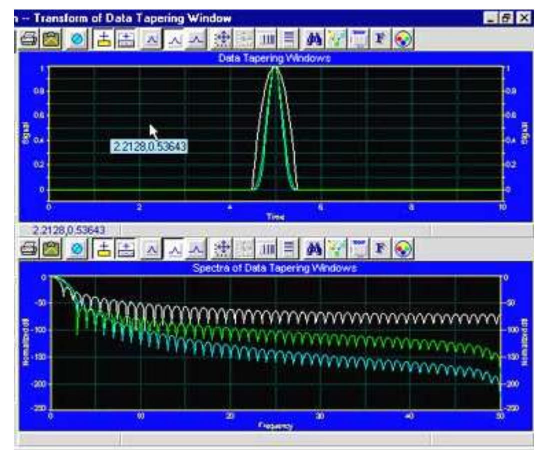
MFCC



Flux



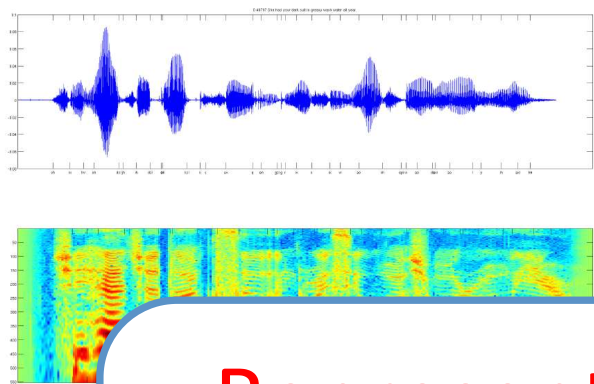
ZCR

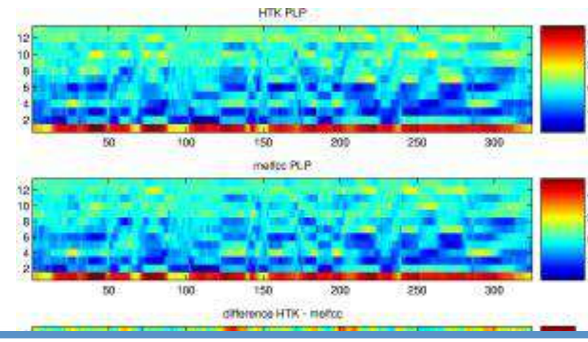


Rolloff

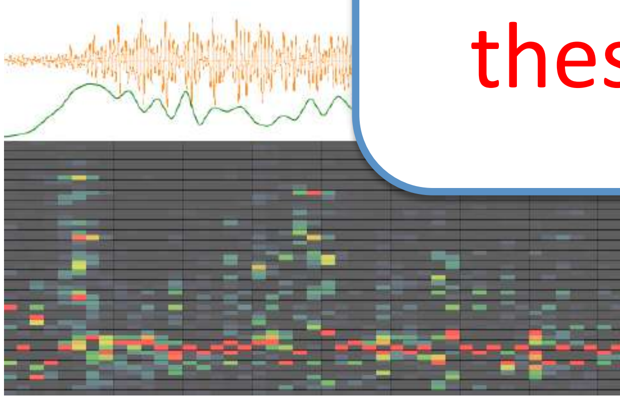
[Courtesy, R. Salakhutdinov]

Audio Features

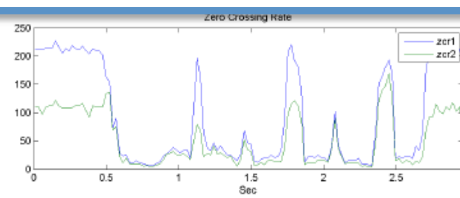




Representation Learning:
Can we automatically learn
these representations?



Flux



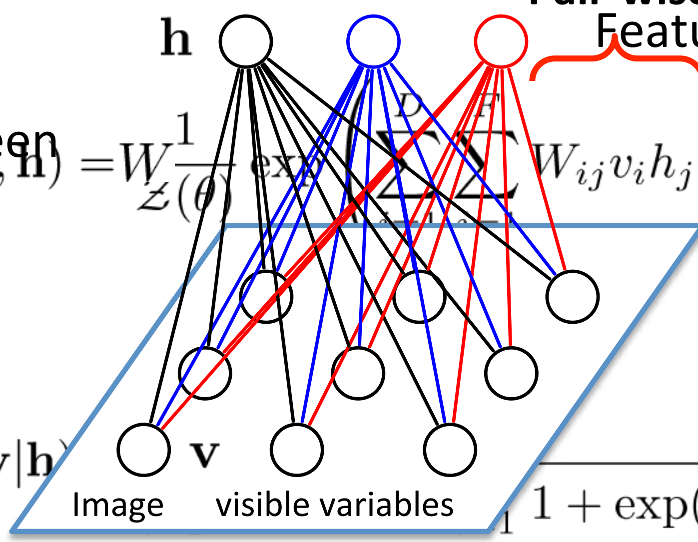
ZCR

Rolloff

[Courtesy, R. Salakhutdinov]

Restricted Boltzmann Machines

Graphical Models: Powerful framework for representing dependency structure between random variables.



$$P_\theta(\mathbf{v},\mathbf{h}) = \frac{1}{\mathcal{Z}(\theta)} \exp\Bigg(\underbrace{\sum_{i=1}^{D}\sum_{j=1}^{F} W_{ij}v_ih_j}_{\textbf{Pair-wise}} + \underbrace{\sum_{i=1}^{D} v_ib_i + \sum_{j=1}^{F} h_ja_j}_{\textbf{Unary}} \Bigg)$$

$$\theta = \{W, a, b\}$$

$$P_\theta(\mathbf{v}|\mathbf{h}) = \frac{1}{1+\exp(-\sum_{j=1}^{F} W_{ij}v_ih_j - b_i)}$$

RBM is a Markov Random Field with:

- Stochastic binary visible variables $\mathbf{v} \in \{0,1\}^D$.
- Stochastic binary hidden variables $\mathbf{h} \in \{0,1\}^F$.
- Bipartite connections.

[Courtesy, R. Salakhutdinov]

Markov random fields, Boltzmann machines, log-linear models.

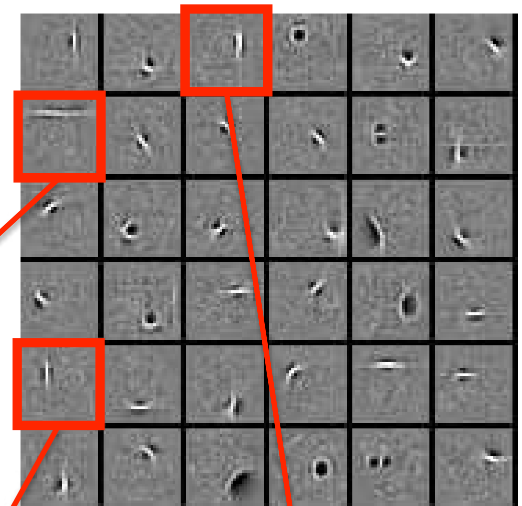
Learning Features

Observed Data

Subset of 25,000 characters

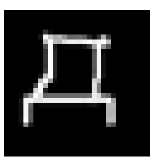
Learned W: "edges"

Subset of 1000 features

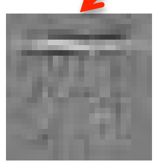


Sparse representations

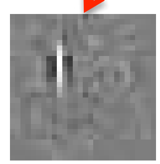
New Image: $p(h_7 = 1|v)$ $p(h_{29} = 1|v)$



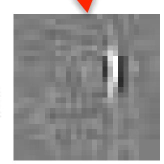
$= \sigma\Big(0.99 \times$



$+ \; 0.97 \times$



$+ \; 0.82 \times$



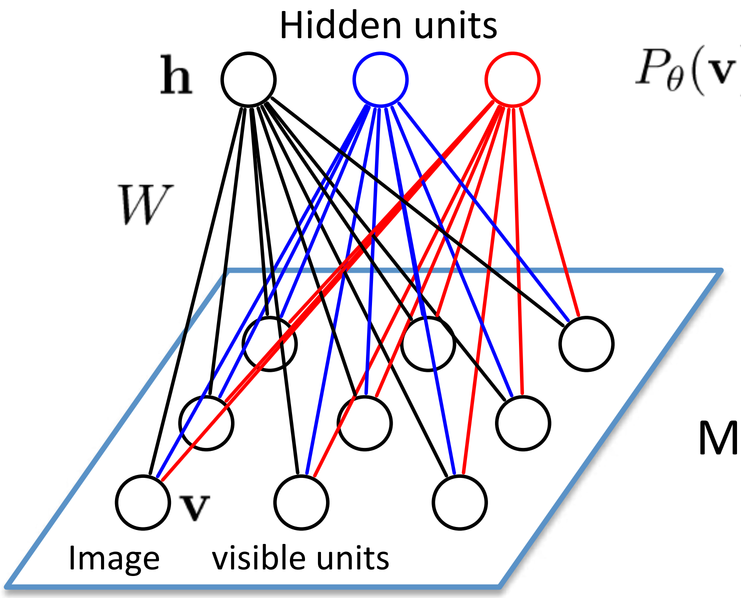
$\cdots \Big)$

$\sigma(x) = \frac{1}{1+\exp(-x)}$

Logistic Function: Suitable for modeling binary images

[Courtesy, R. Salakhutdinov]

Model Learning



$$P_\theta(\mathbf{v}) = \frac{P^*(\mathbf{v})}{\mathcal{Z}(\theta)} = \frac{1}{\mathcal{Z}(\theta)} \sum_{\mathbf{h}} \exp\left[\mathbf{v}^\top W \mathbf{h} + \mathbf{a}^\top \mathbf{h} + \mathbf{b}^\top \mathbf{v}\right]$$

Given a set of *i.i.d.* training examples $\mathcal{D} = \{\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, ..., \mathbf{v}^{(N)}\}$, we want to learn model parameters $\theta = \{W, a, b\}$.

Maximize log-likelihood objective:

$$L(\theta) = \frac{1}{N} \sum_{n=1}^{N} \log P_\theta(\mathbf{v}^{(n)})$$

Derivative of the log-likelihood:

$$\frac{\partial L(\theta)}{\partial W_{ij}} = \frac{1}{N} \sum_{n=1}^{N} \frac{\partial}{\partial W_{ij}} \log\left(\sum_{\mathbf{h}} \exp\left[\mathbf{v}^{(n)\top} W \mathbf{h} + \mathbf{a}^\top \mathbf{h} + \mathbf{b}^\top \mathbf{v}^{(n)}\right]\right) - \frac{\partial}{\partial W_{ij}} \log \mathcal{Z}(\theta)$$

$$= \mathrm{E}_{P_{data}}[v_i h_j] - \underbrace{\mathrm{E}_{P_\theta}[v_i h_j]}$$

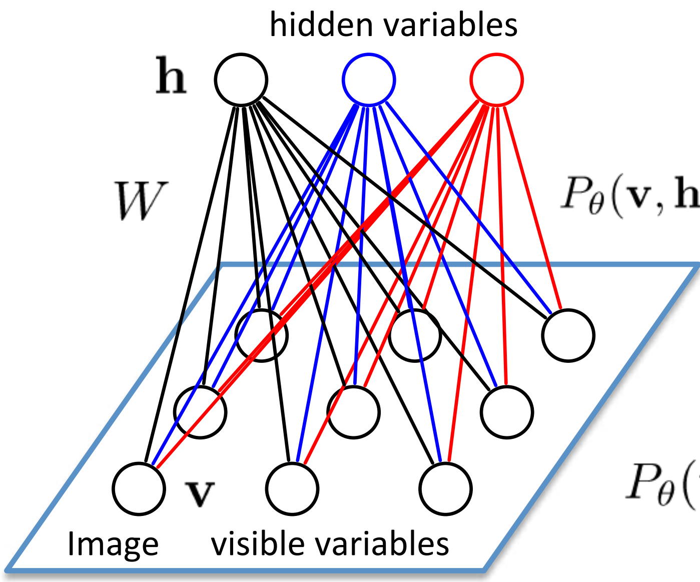
Difficult to compute: exponentially many configurations

$$P_{data}(\mathbf{v}, \mathbf{h}; \theta) = P(\mathbf{h}|\mathbf{v}; \theta) P_{data}(\mathbf{v})$$

$$P_{data}(\mathbf{v}) = \frac{1}{N} \sum_{n} \delta(\mathbf{v} - \mathbf{v}^{(n)})$$

[Courtesy, R. Salakhutdinov]

RBMs for Real-valued Data



$$P_\theta(\mathbf{v},\mathbf{h}) = \frac{1}{\mathcal{Z}(\theta)} \exp\left(\overbrace{\sum_{i=1}^{D}\sum_{j=1}^{F} W_{ij} h_j \frac{v_i}{\sigma_i}}^{\text{Pair-wise}} + \overbrace{\sum_{i=1}^{D} \frac{(v_i - b_i)^2}{2\sigma_i^2} + \sum_{j=1}^{F} a_j h_j}^{\text{Unary}} \right)$$

$$\theta = \{W, a, b\}$$

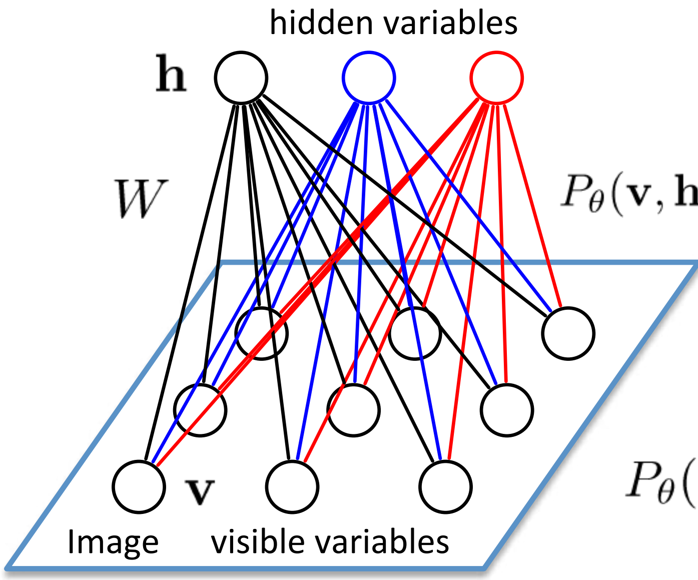
$$P_\theta(\mathbf{v}|\mathbf{h}) = \prod_{i=1}^{D} P_\theta(v_i|\mathbf{h}) = \prod_{i=1}^{D} \mathcal{N}\left(b_i + \sum_{j=1}^{F} W_{ij} h_j, \sigma_i^2 \right)$$

Gaussian-Bernoulli RBM:

- Stochastic real-valued visible variables $\mathbf{v} \in \mathbb{R}^D$.
- Stochastic binary hidden variables $\mathbf{h} \in \{0,1\}^F$.
- Bipartite connections.

[Courtesy, R. Salakhutdinov] (Salakhutdinov & Hinton, NIPS 2007; Salakhutdinov & Murray, ICML 2008)

RBMs for Real-valued Data



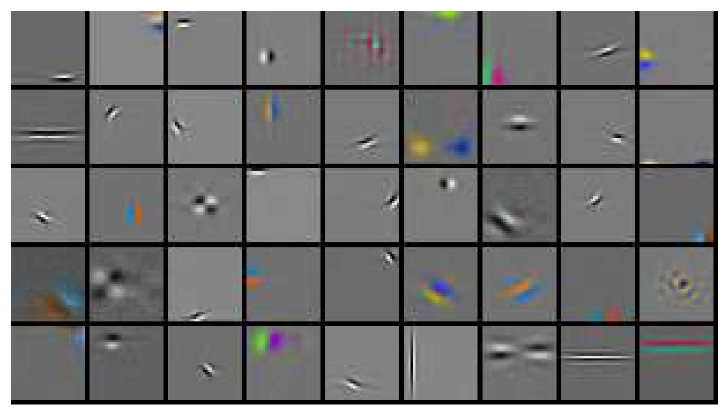
$$P_\theta(\mathbf{v},\mathbf{h}) = \frac{1}{\mathcal{Z}(\theta)} \exp\left(\sum_{i=1}^{D}\sum_{j=1}^{F} \overbrace{W_{ij} h_j \frac{v_i}{\sigma_i}}^{\text{Pair-wise}} + \sum_{i=1}^{D} \overbrace{\frac{(v_i - b_i)^2}{2\sigma_i^2}}^{\text{Unary}} + \sum_{j=1}^{F} \overbrace{a_j h_j}^{\text{Unary}} \right)$$

$$\theta = \{W, a, b\}$$

$$P_\theta(\mathbf{v}|\mathbf{h}) = \prod_{i=1}^{D} P_\theta(v_i|\mathbf{h}) = \prod_{i=1}^{D} \mathcal{N}\left(b_i + \sum_{j=1}^{F} W_{ij} h_j, \sigma_i^2 \right)$$

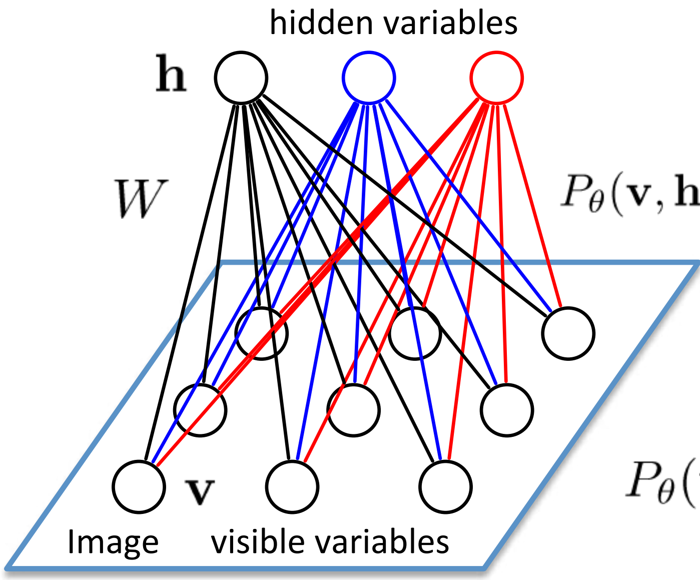
4 million **unlabelled** images

Learned features (out of 10,000)



[Courtesy, R. Salakhutdinov]

RBMs for Real-valued Data



$$P_\theta(\mathbf{v},\mathbf{h}) = \frac{1}{\mathcal{Z}(\theta)} \exp\left(\overbrace{\sum_{i=1}^{D}\sum_{j=1}^{F} W_{ij} h_j \frac{v_i}{\sigma_i}}^{\text{Pair-wise}} + \overbrace{\sum_{i=1}^{D} \frac{(v_i - b_i)^2}{2\sigma_i^2} + \sum_{j=1}^{F} a_j h_j}^{\text{Unary}} \right)$$

$$\theta = \{W, a, b\}$$

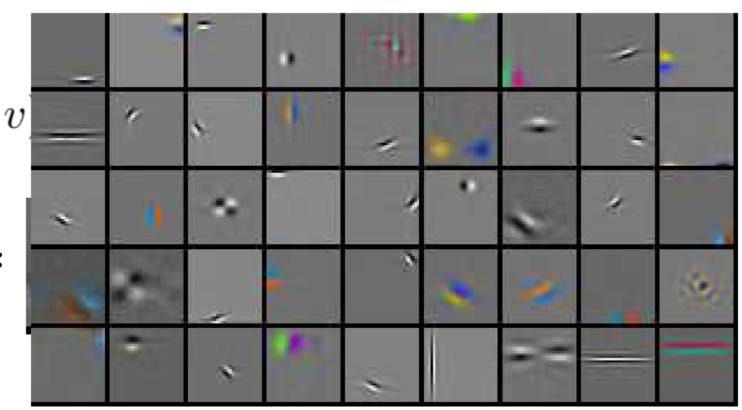
$$P_\theta(\mathbf{v}|\mathbf{h}) = \prod_{i=1}^{D} P_\theta(v_i|\mathbf{h}) = \prod_{i=1}^{D} \mathcal{N}\left(b_i + \sum_{j=1}^{F} W_{ij} h_j, \sigma_i^2\right)$$

4 million **unlabelled** images

$p(h_{29} = 1|v$

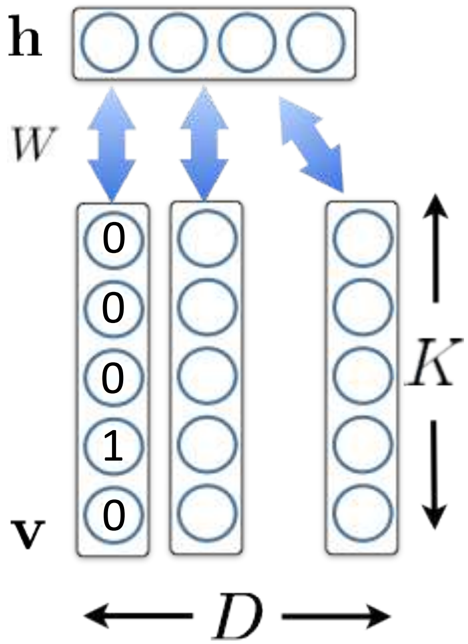
$+\ 0.8\ *$

Learned features (out of 10,000)



[Courtesy, R. Salakhutdinov]

RBMs for Word Counts



$$P_\theta(\mathbf{v},\mathbf{h}) = \frac{1}{\mathcal{Z}(\theta)} \exp\left(\overbrace{\sum_{i=1}^{D}\sum_{k=1}^{K}\sum_{j=1}^{F} W_{ij}^k v_i^k h_j}^{\text{Pair-wise}} + \overbrace{\sum_{i=1}^{D}\sum_{k=1}^{K} v_i^k b_i^k + \sum_{j=1}^{F} h_j a_j}^{\text{Unary}} \right)$$

$$\theta = \{W, a, b\}$$

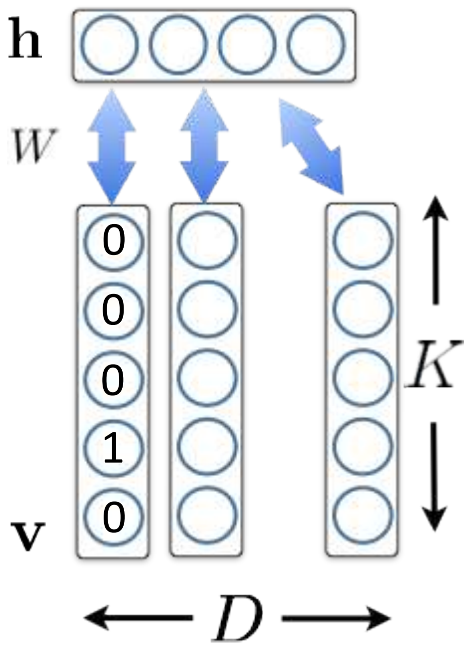
$$P_\theta(v_i^k = 1|\mathbf{h}) = \frac{\exp\left(b_i^k + \sum_{j=1}^{F} h_j W_{ij}^k\right)}{\sum_{q=1}^{K} \exp\left(b_i^q + \sum_{j=1}^{F} h_j W_{ij}^q\right)}$$

Replicated Softmax Model: undirected topic model:

- Stochastic 1-of-K visible variables.
- Stochastic binary hidden variables $\mathbf{h} \in \{0,1\}^F$.
- Bipartite connections.

[Courtesy, R. Salakhutdinov] Salakhutdinov & Hinton, NIPS 2010, Srivastava & Salakhutdinov, NIPS 2012)

RBMs for Word Counts



$$P_\theta(\mathbf{v},\mathbf{h}) = \frac{1}{\mathcal{Z}(\theta)} \exp\left(\overbrace{\sum_{i=1}^{D}\sum_{k=1}^{K}\sum_{j=1}^{F} W_{ij}^{k} v_i^k h_j}^{\textbf{Pair-wise}} + \overbrace{\sum_{i=1}^{D}\sum_{k=1}^{K} v_i^k b_i^k}^{\textbf{Unary}} + \overbrace{\sum_{j=1}^{F} h_j a_j} \right)$$

$$\theta = \{W, a, b\}$$

$$P_\theta(v_i^k = 1|\mathbf{h}) = \frac{\exp\left(b_i^k + \sum_{j=1}^{F} h_j W_{ij}^k\right)}{\sum_{q=1}^{K}\exp\left(b_i^q + \sum_{j=1}^{F} h_j W_{ij}^q\right)}$$

WIKIPEDIA The Free Encyclopedia

REUTERS

AP Associated Press

Reuters dataset:
804,414 **unlabeled**
newswire stories

Bag-of-Words

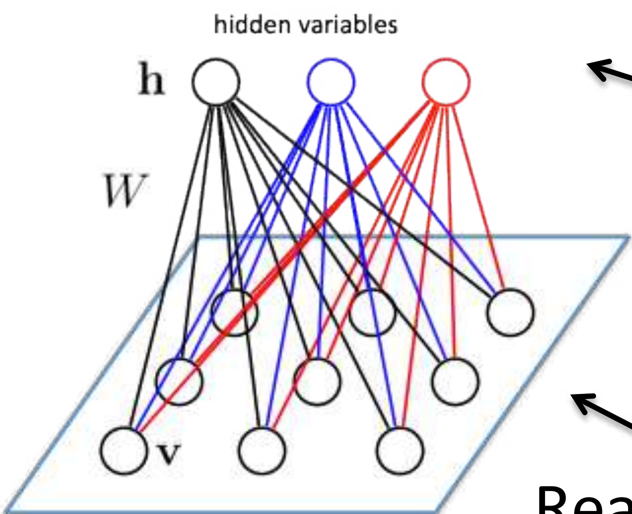
Learned features: ``topics''

russian	clinton	computer	trade	stock
russia	house	system	country	wall
moscow	president	product	import	street
yeltsin	bill	software	world	point
soviet	congress	develop	economy	dow

[Courtesy, R. Salakhutdinov]

Different Data Modalities

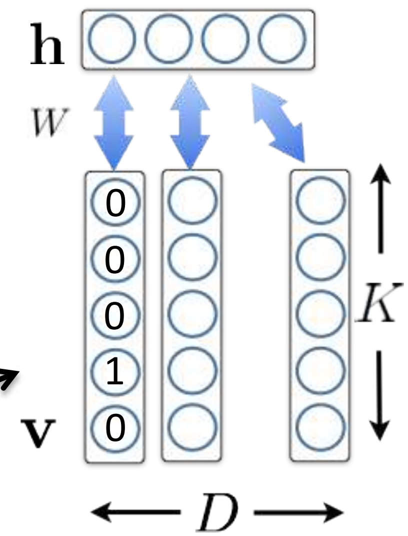
- Binary/Gaussian/Softmax RBMs: All have binary hidden variables but use them to model different kinds of data.



Binary

Real-valued

1-of-K



- It is easy to infer the states of the hidden variables:

$$P_\theta(\mathbf{h}|\mathbf{v}) = \prod_{j=1}^{F} P_\theta(h_j|\mathbf{v}) = \prod_{j=1}^{F} \frac{1}{1 + \exp(-a_j - \sum_{i=1}^{D} W_{ij} v_i)}$$

[Courtesy, R. Salakhutdinov]

Product of Experts

The joint distribution is given by:

$$P_\theta(\mathbf{v},\mathbf{h}) = \frac{1}{\mathcal{Z}(\theta)} \exp\Big(\sum_{ij} W_{ij} v_i h_j + \sum_i b_i v_i + \sum_j a_j h_j\Big)$$

Marginalizing over hidden variables:

$$P_\theta(\mathbf{v}) = \sum_{\mathbf{h}} P_\theta(\mathbf{v},\mathbf{h}) = \frac{1}{\mathcal{Z}(\theta)} \prod_i \exp(b_i v_i) \prod_j \overbrace{\Big(1 + \exp(a_j + \sum_i W_{ij} v_i)\Big)}^{\textbf{Product of Experts}}$$

government	clinton	bribery	oil	stock	…
auhority	house	corruption	barrel	wall	
power	president	dishonesty	exxon	street	
empire	bill	putin	putin	point	
putin	congress	fraud	drill	dow	

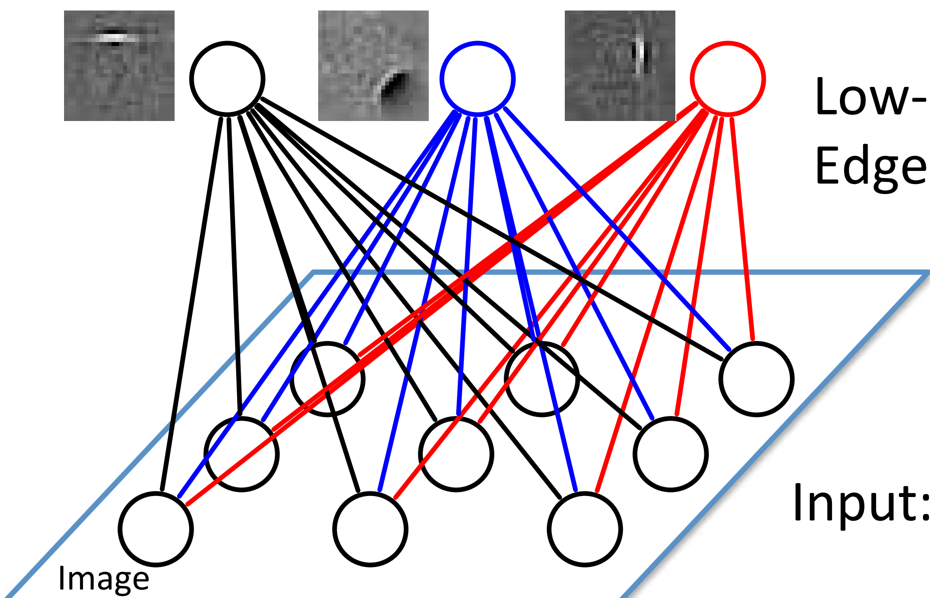
Putin

Topics “government”, ”corruption” and ”oil” can combine to give very high probability to a word “Putin”.

[Courtesy, R. Salakhutdinov]

(Srivastava & Salakhutdinov, NIPS 2012)

Deep Boltzmann Machines



Low-level features:
Edges

Built from **unlabeled** inputs.

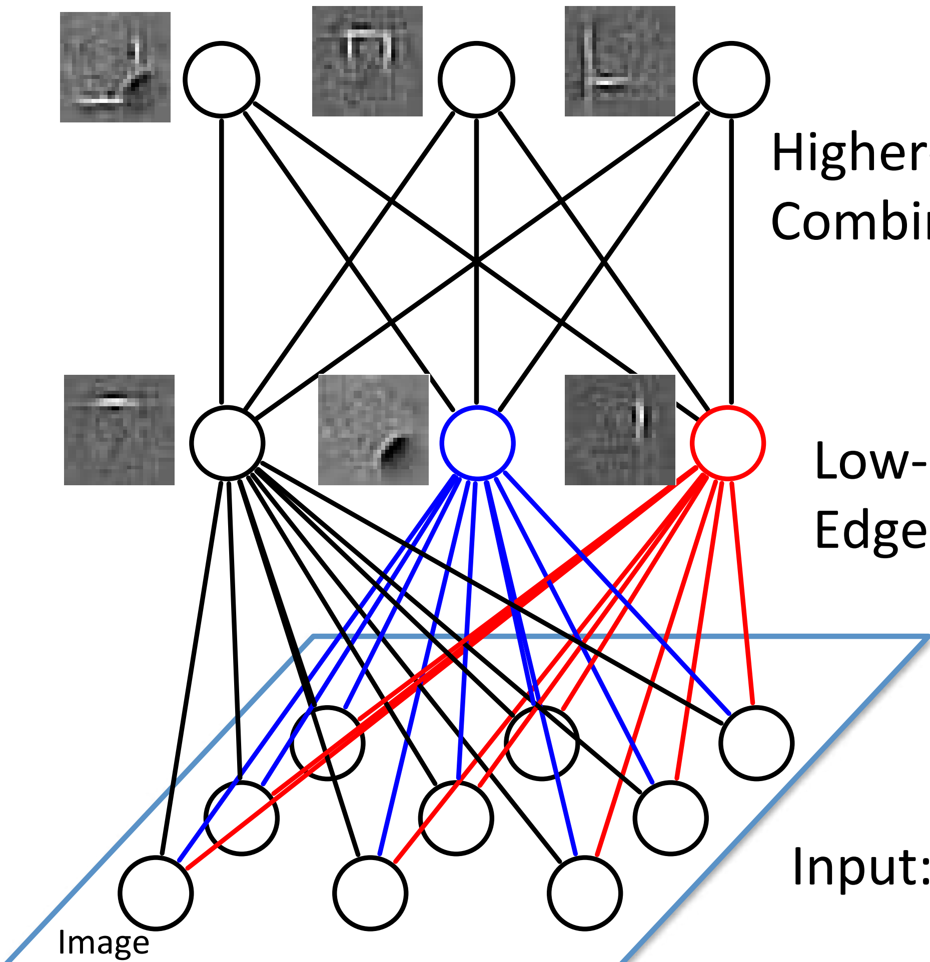
Input: Pixels

[Courtesy, R. Salakhutdinov]

(Salakhutdinov & Hinton, Neural Computation 2012)

Deep Boltzmann Machines

Learn simpler representations,
then compose more complex ones



Higher-level features:
Combination of edges

Low-level features:
Edges

Built from **unlabeled** inputs.

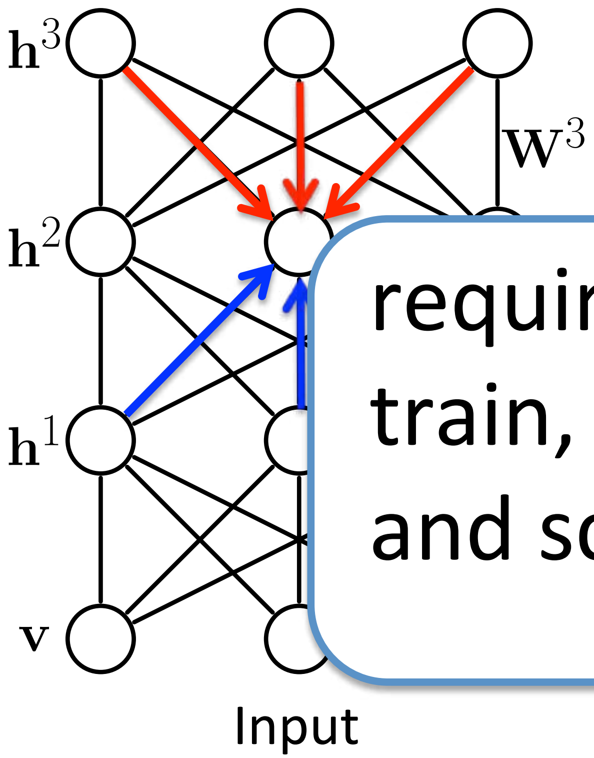
Input: Pixels

[Courtesy, R. Salakhutdinov]

(Salakhutdinov 2008, Salakhutdinov & Hinton 2012)

Model Formulation

$$P_\theta(\mathbf{v}, \mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \mathbf{h}^{(3)}) = \frac{1}{\mathcal{Z}(\theta)} \exp\left[\underbrace{\mathbf{v}^\top W^{(1)} \mathbf{h}^{(1)}}_{\textbf{Same as RBMs}} + \underline{\mathbf{h}^{(1)\top} W^{(2)} \mathbf{h}^{(2)}} + \underline{\mathbf{h}^{(2)\top} W^{(3)} \mathbf{h}^{(3)}}\right]$$



$\theta = \{W^1, W^2, W^3\}$ model parameters

requires approximate inference to train, but it can be done…
and scales to millions of examples

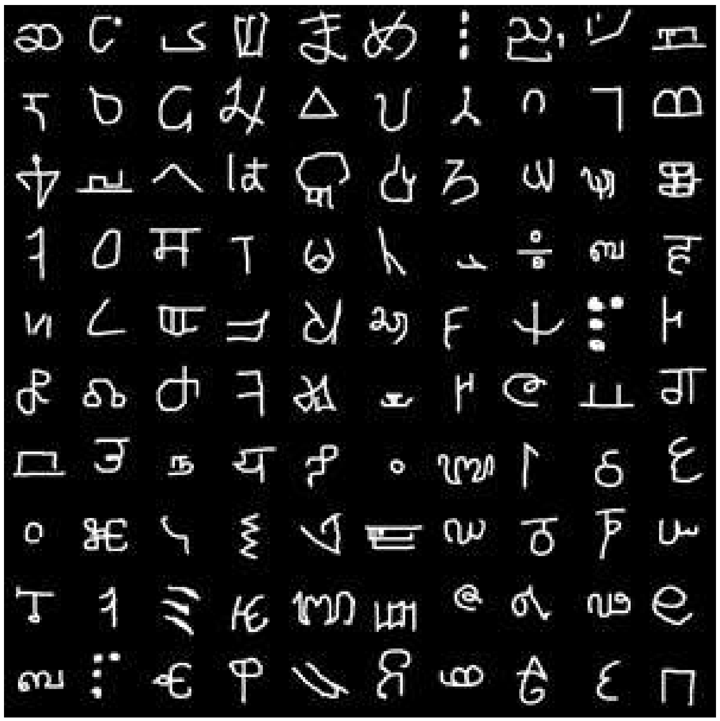
Top-down

Bottom-up

[Courtesy, R. Salakhutdinov]

Samples Generated by the Model

Training Data



Model-Generated Samples

[Courtesy, R. Salakhutdinov]

Handwriting Recognition

MNIST Dataset
60,000 examples of 10 digits

Learning Algorithm	Error
Logistic regression	12.0%
K-NN	3.09%
Neural Net (Platt 2005)	1.53%
SVM (Decoste et.al. 2002)	1.40%
Deep Autoencoder (Bengio et. al. 2007)	1.40%
Deep Belief Net (Hinton et. al. 2006)	1.20%
DBM	**0.95%**

Optical Character Recognition
42,152 examples of 26 English letters

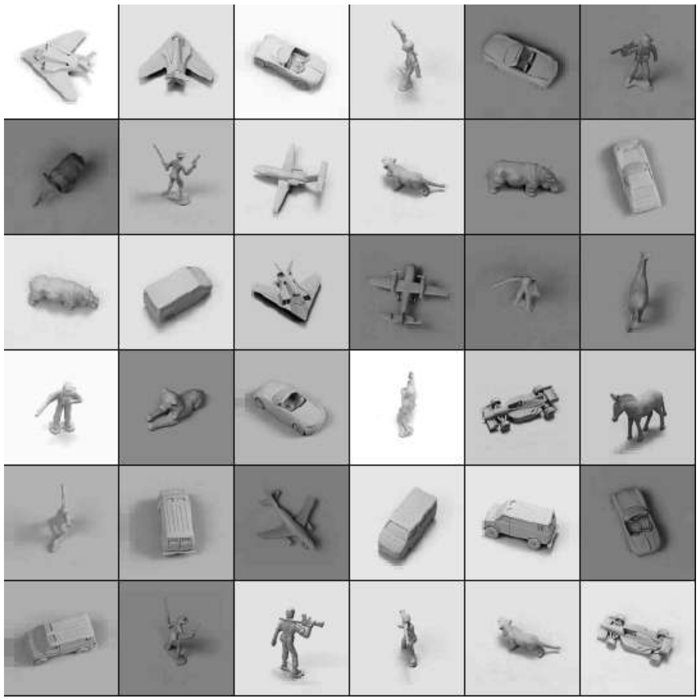
Learning Algorithm	Error
Logistic regression	22.14%
K-NN	18.92%
Neural Net	14.62%
SVM (Larochelle et.al. 2009)	9.70%
Deep Autoencoder (Bengio et. al. 2007)	10.05%
Deep Belief Net (Larochelle et. al. 2009)	9.68%
DBM	**8.40%**

Permutation-invariant version.

[Courtesy, R. Salakhutdinov]

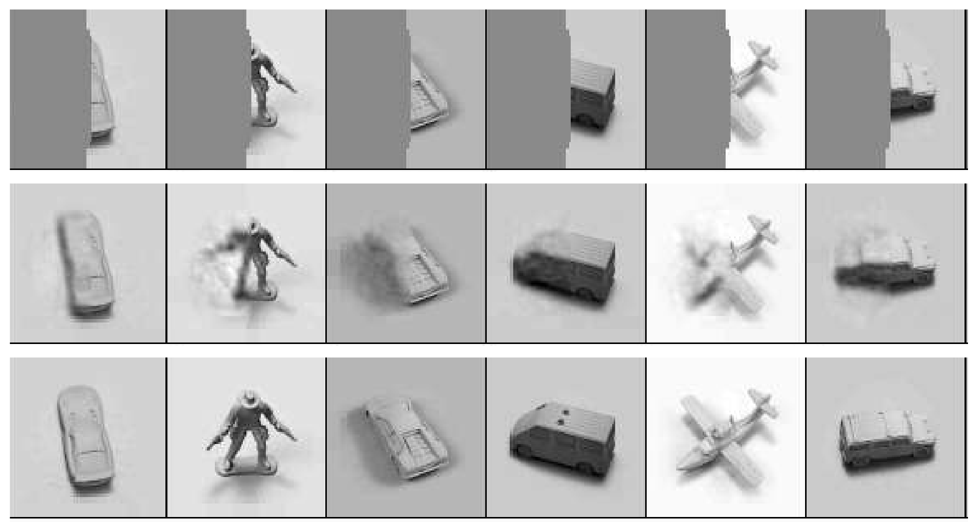
3-D object Recognition

NORB Dataset: 24,000 examples



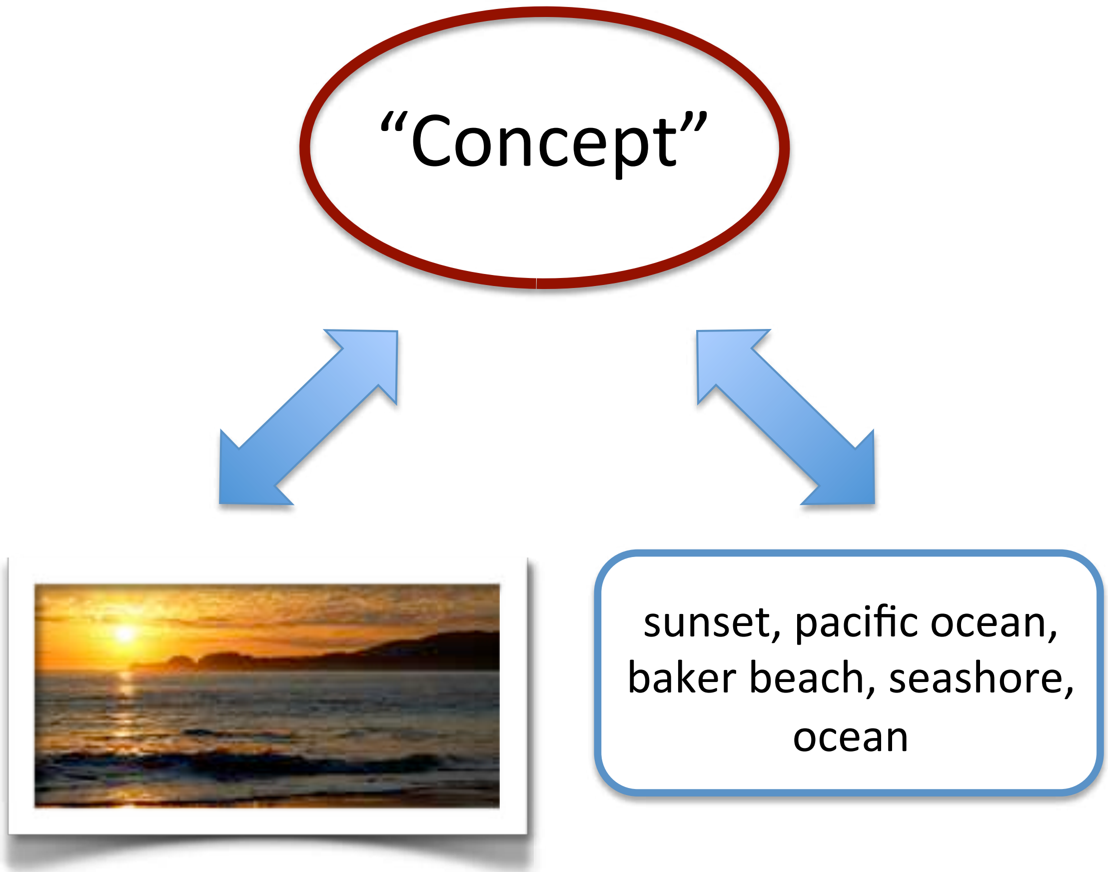
Learning Algorithm	Error
Logistic regression	22.5%
K-NN (LeCun 2004)	18.92%
SVM (Bengio & LeCun 2007)	11.6%
Deep Belief Net (Nair & Hinton 2009)	9.0%
DBM	**7.2%**

Pattern Completion



[Courtesy, R. Salakhutdinov]

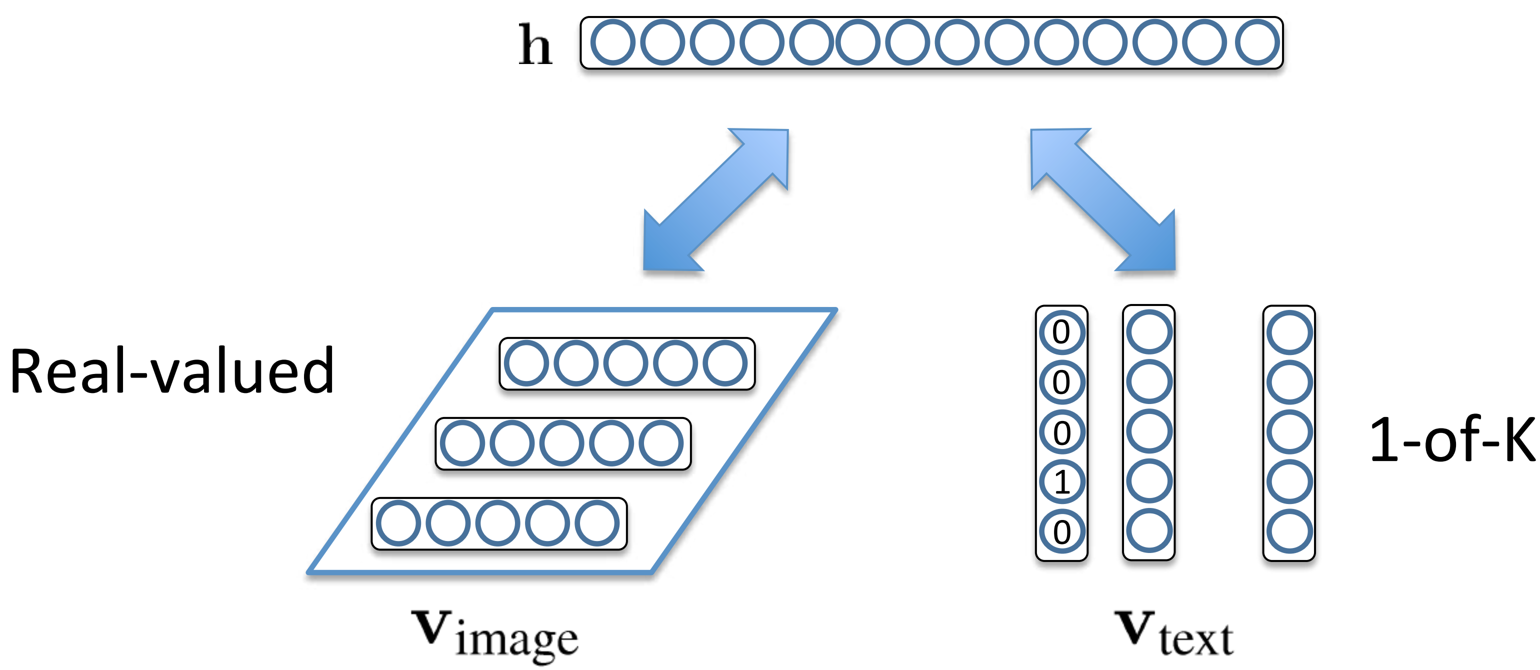
Learning Shared Representations Across Sensory Modalities



[Courtesy, R. Salakhutdinov]

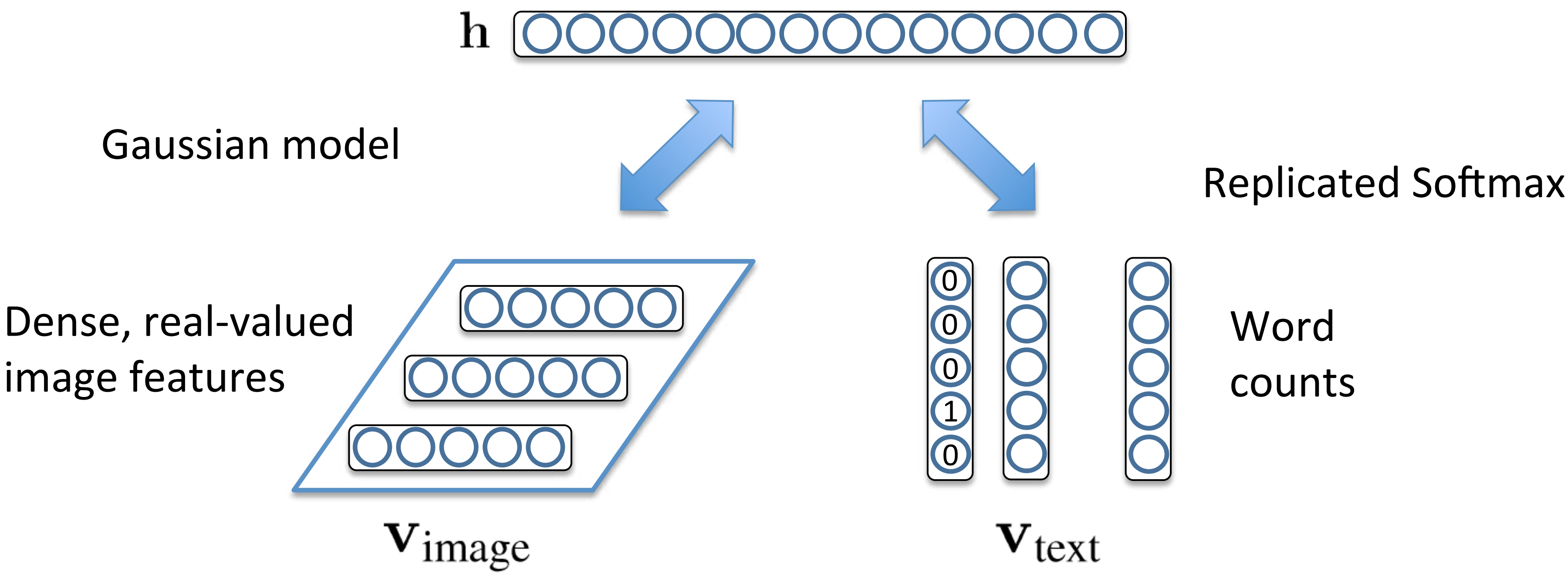
A Simple Multimodal Model

- Use a joint binary hidden layer.
- **Problem**: Inputs have very different statistical properties.
- Difficult to learn cross-modal features.



[Courtesy, R. Salakhutdinov]

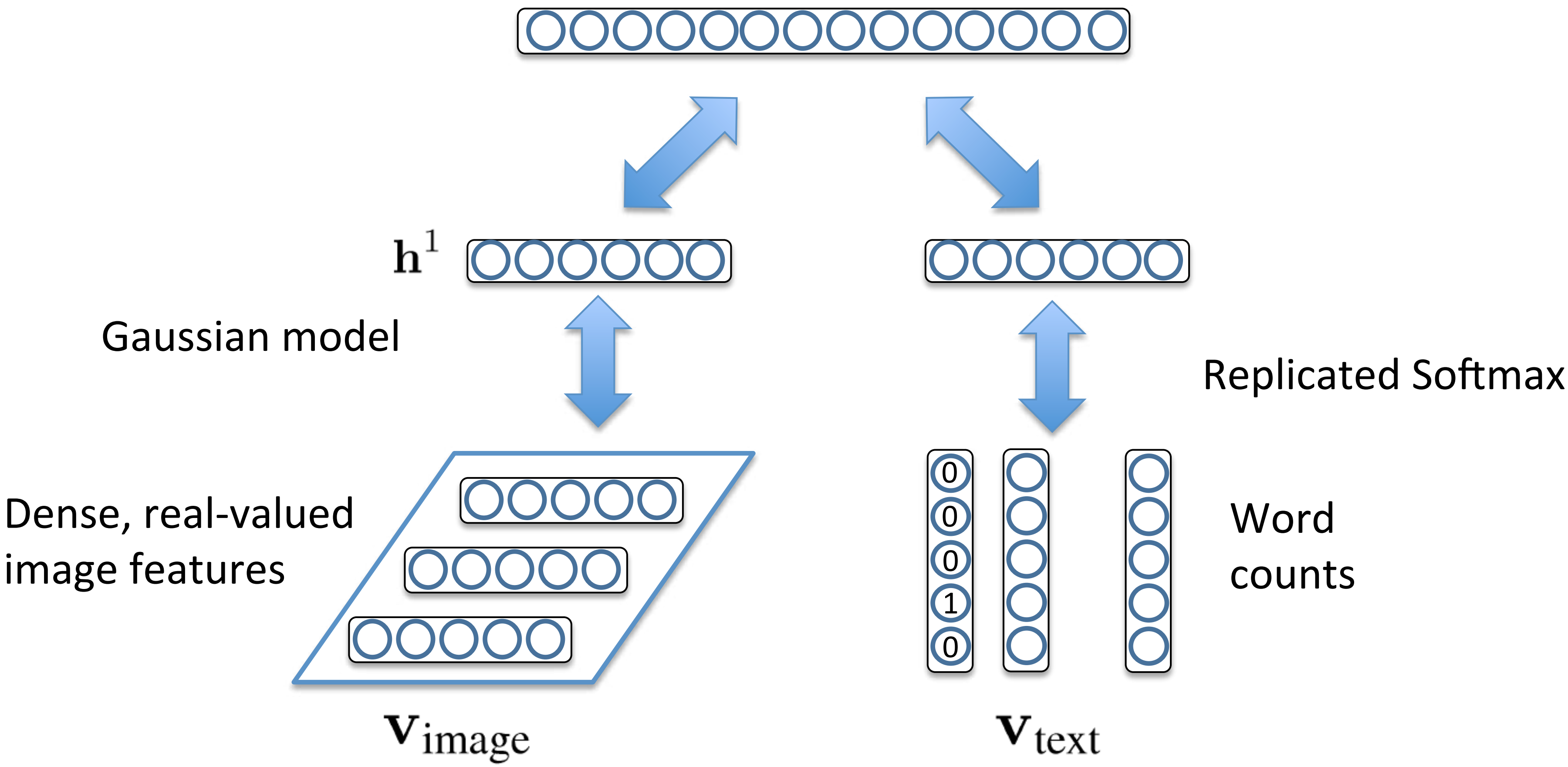
Multimodal DBM



[Courtesy, R. Salakhutdinov]

(Srivastava & Salakhutdinov, NIPS 2012, JMLR 2014)

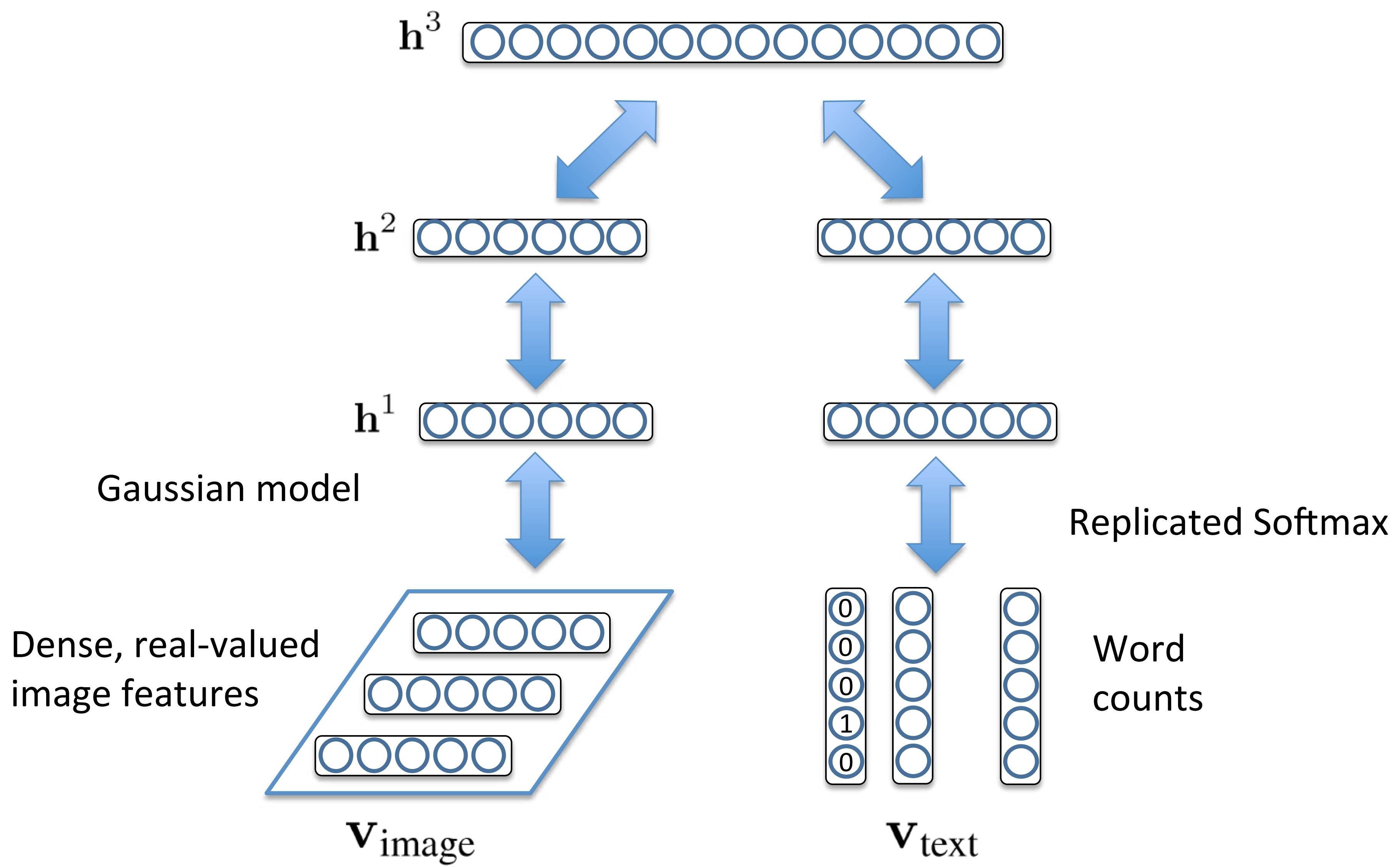
Multimodal DBM



(Srivastava & Salakhutdinov, NIPS 2012, JMLR 2014)

[Courtesy, R. Salakhutdinov]

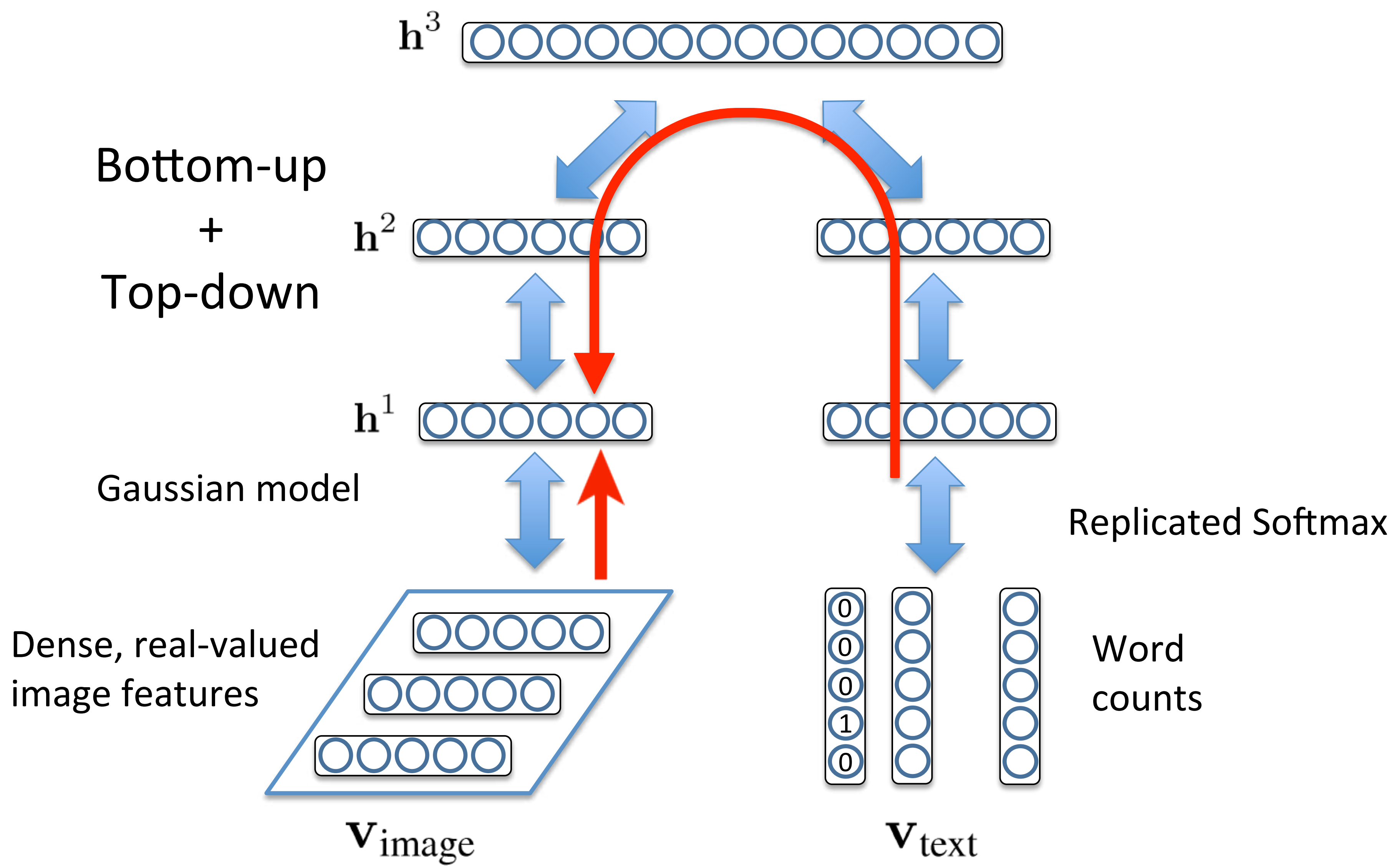
Multimodal DBM



[Courtesy, R. Salakhutdinov]

(Srivastava & Salakhutdinov, NIPS 2012, JMLR 2014)

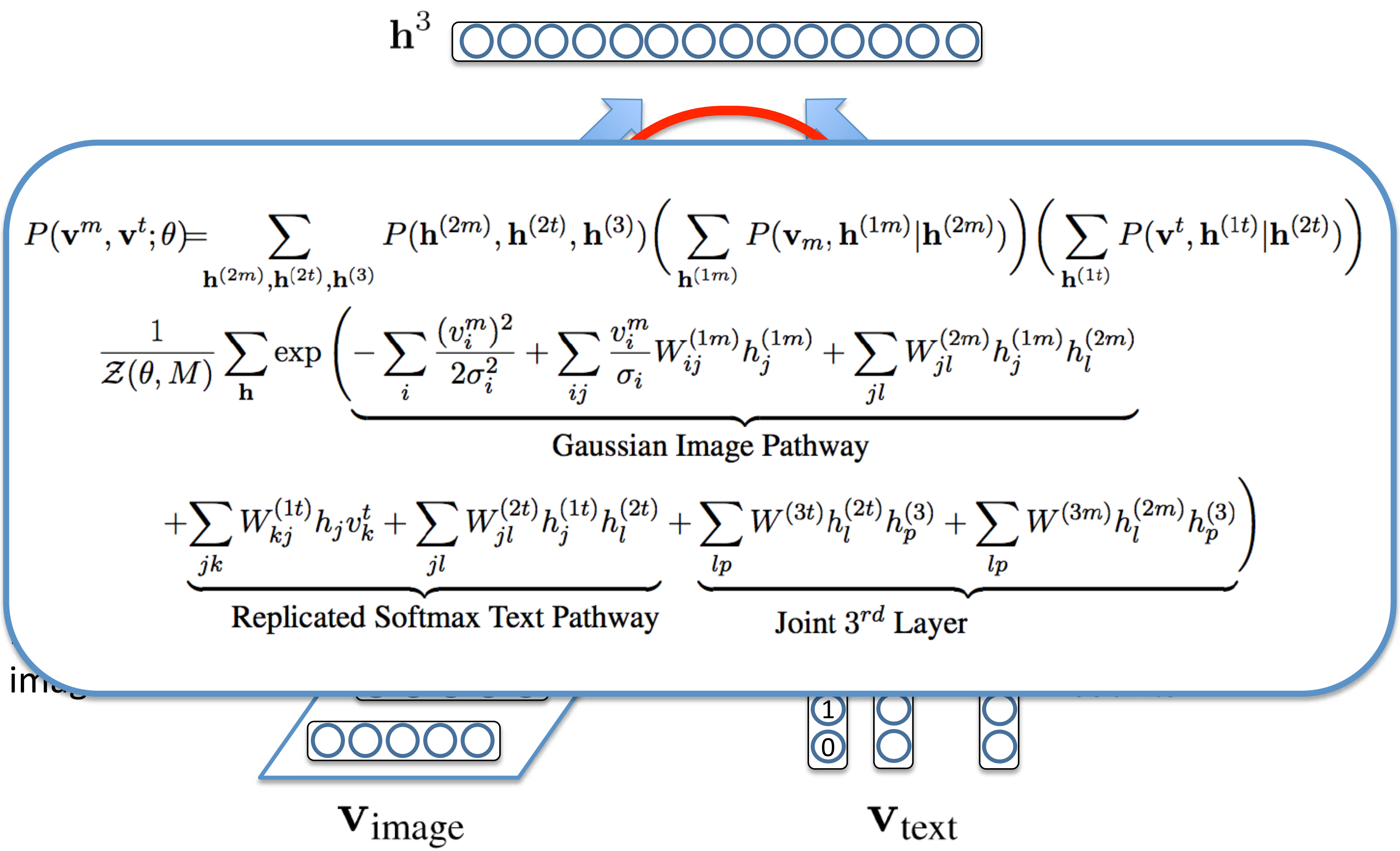
Multimodal DBM



(Srivastava & Salakhutdinov, NIPS 2012, JMLR 2014)

[Courtesy, R. Salakhutdinov]

Multimodal DBM



$$P(\mathbf{v}^m,\mathbf{v}^t;\theta) = \sum_{\mathbf{h}^{(2m)},\mathbf{h}^{(2t)},\mathbf{h}^{(3)}} P(\mathbf{h}^{(2m)},\mathbf{h}^{(2t)},\mathbf{h}^{(3)}) \left(\sum_{\mathbf{h}^{(1m)}} P(\mathbf{v}_m,\mathbf{h}^{(1m)}|\mathbf{h}^{(2m)})\right)\left(\sum_{\mathbf{h}^{(1t)}} P(\mathbf{v}^t,\mathbf{h}^{(1t)}|\mathbf{h}^{(2t)})\right)$$

$$\frac{1}{\mathcal{Z}(\theta,M)}\sum_{\mathbf{h}}\exp\Bigg(\underbrace{-\sum_i \frac{(v_i^m)^2}{2\sigma_i^2} + \sum_{ij}\frac{v_i^m}{\sigma_i}W_{ij}^{(1m)}h_j^{(1m)} + \sum_{jl}W_{jl}^{(2m)}h_j^{(1m)}h_l^{(2m)}}_{\text{Gaussian Image Pathway}}$$

$$\underbrace{+\sum_{jk}W_{kj}^{(1t)}h_j v_k^t + \sum_{jl}W_{jl}^{(2t)}h_j^{(1t)}h_l^{(2t)}}_{\text{Replicated Softmax Text Pathway}} \underbrace{+\sum_{lp}W^{(3t)}h_l^{(2t)}h_p^{(3)} + \sum_{lp}W^{(3m)}h_l^{(2m)}h_p^{(3)}\Bigg)}_{\text{Joint } 3^{rd} \text{ Layer}}$$

[Courtesy, R. Salakhutdinov]

(Srivastava & Salakhutdinov, NIPS 2012, JMLR 2014)

Text Generated from Images

Given	Generated
	dog, cat, pet, kitten, puppy, ginger, tongue, kitty, dogs, furry
	sea, france, boat, mer, beach, river, bretagne, plage, brittany
	portrait, child, kid, ritratto, kids, children, boy, cute, boys, italy

Given	Generated
	insect, butterfly, insects, bug, butterflies, lepidoptera
	graffiti, streetart, stencil, sticker, urbanart, graff, sanfrancisco
	canada, nature, sunrise, ontario, fog, mist, bc, morning

[Courtesy, R. Salakhutdinov]

Text Generated from Images

Given | Generated

portrait, women, army, soldier, mother, postcard, soldiers

obama, barackobama, election, politics, president, hope, change, sanfrancisco, convention, rally

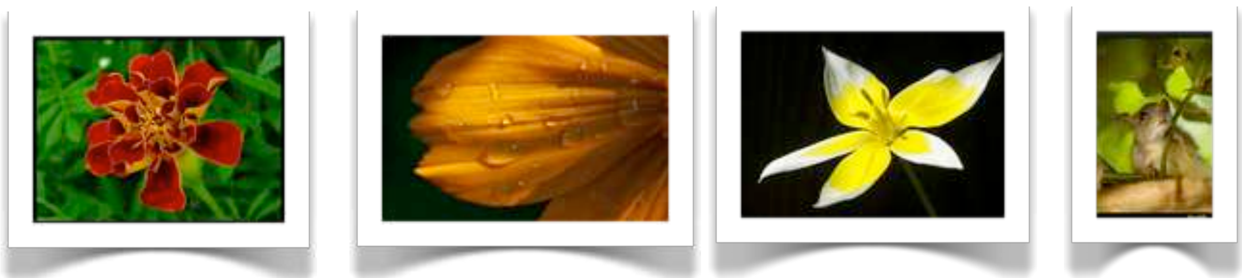
water, glass, beer, bottle, drink, wine, bubbles, splash, drops, drop

Images Generated from Text

Given | Retrieved

water, red, sunset

nature, flower, red, green



blue, green, yellow, colors

chocolate, cake

[Courtesy, R. Salakhutdinov]

MIR-Flickr Dataset

- 1 million images along with user-assigned tags.

sculpture, beauty, stone

d80

nikon, abigfave, goldstaraward, d80, nikond80

food, cupcake, vegan

anawesomeshot, theperfectphotographer, flash, damniwishidtakenthat, spiritofphotography

nikon, green, light, photoshop, apple, d70

white, yellow, abstract, lines, bus, graphic

sky, geotagged, reflection, cielo, bilbao, reflejo

Huiskes et. al.

[Courtesy, R. Salakhutdinov]

Results

- Logistic regression on top-level representation.
- Multimodal Inputs

Mean Average Precision

Learning Algorithm	MAP	Precision@50
Random	0.124	0.124
LDA [Huiskes et. al.]	0.492	0.754
SVM [Huiskes et. al.]	0.475	0.758
DBM-Labelled	0.526	0.791
Deep Belief Net	0.638	0.867
Autoencoder	0.638	0.875
DBM	0.641	0.873

Labeled 25K examples

+ 1 Million unlabelled

[Courtesy, R. Salakhutdinov]

Artificial Neural Networks: Summary

- Highly non-linear regression/classification
- Hidden layers learn intermediate representations
- Potentially millions of parameters to estimate
- Stochastic gradient descent, local minima problems

- Deep networks have produced real progress in many fields
  - computer vision
  - speech recognition
  - mapping images to text
  - recommender systems
  - …
- They learn very useful non-linear representations