

# A Tutorial on Deep Learning

Kai Yu

Multimedia Department, Baidu



# **Background**

# Classification Models Since Late 80's

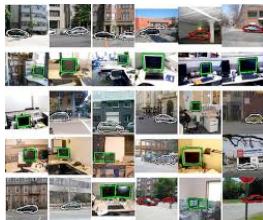
- Neural Networks
- Boosting
- Support Vector Machines
- Maximum Entropy
- ...

# Since 2000 – Learning with Structures

- Kernel Learning
- Transfer Learning
- Semi-supervised Learning
- Manifold Learning
- Sparse Learning
- Matrix Factorization
- Structured Input-Output Prediction
- ...

# Mission Yet Accomplished

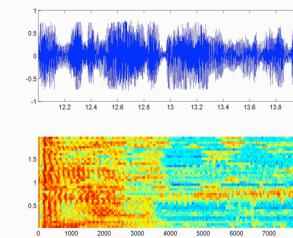
Images & Video



Text & Language



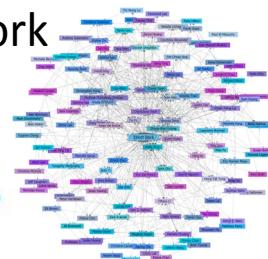
Speech & Audio



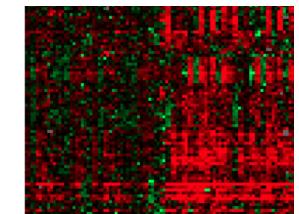
Product  
Recommendation



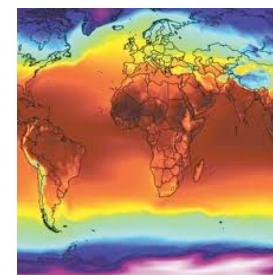
Relational Data/  
Social Network



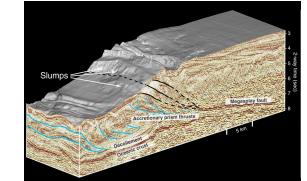
Gene Expression



Climate Change

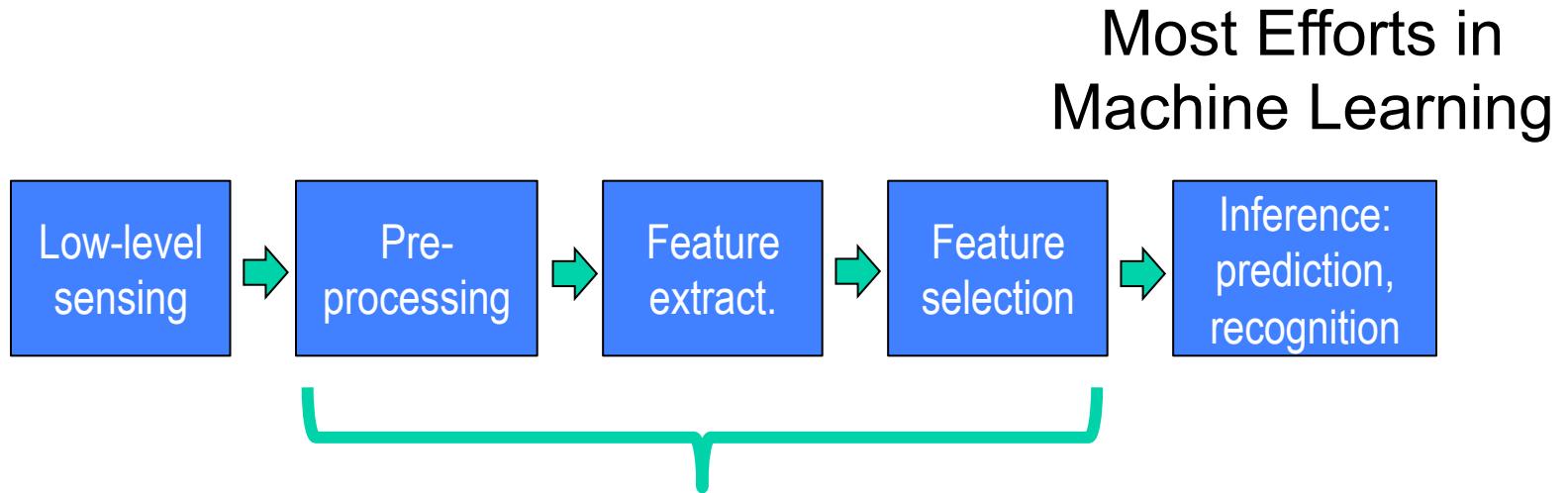


Geological Data



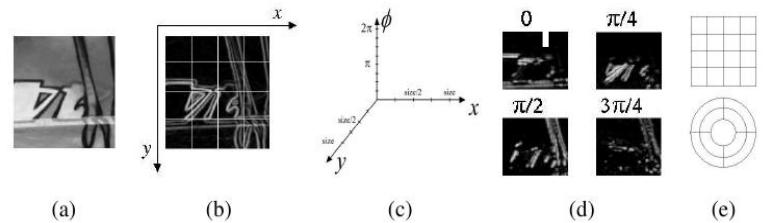
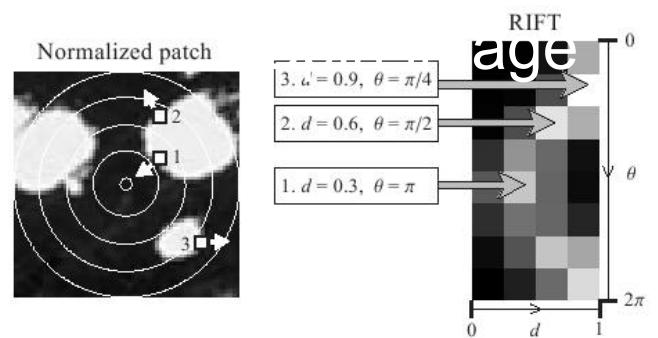
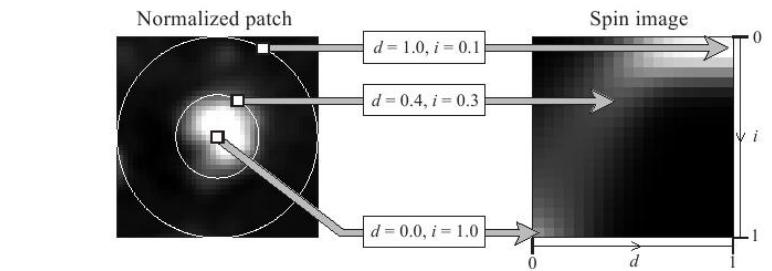
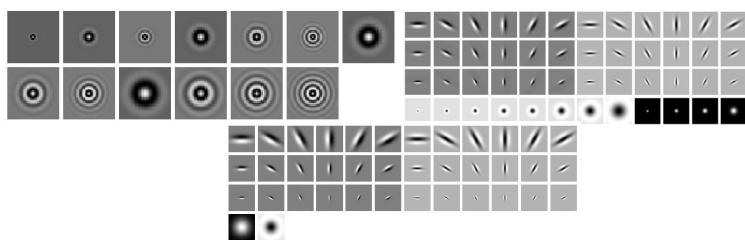
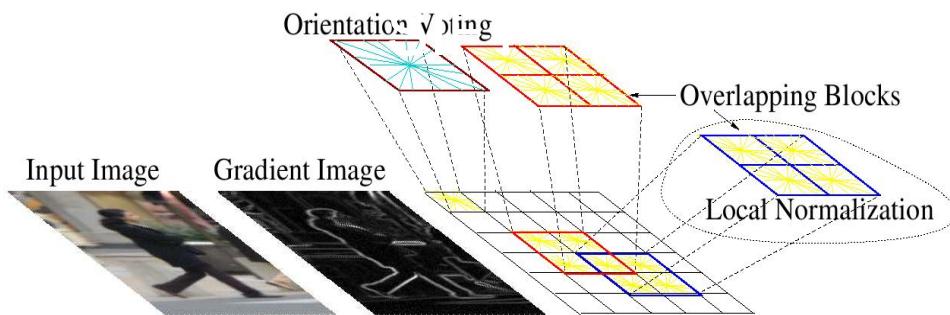
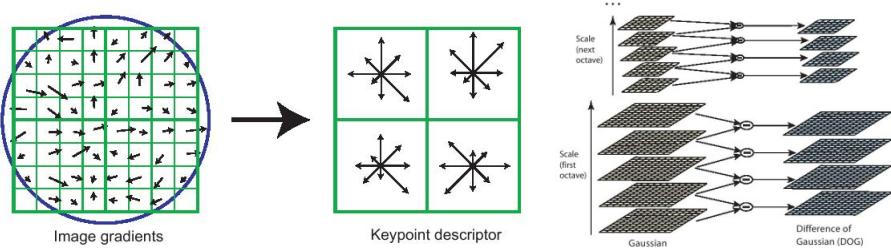
Slide Courtesy: Russ Salakhutdinov

# The pipeline of machine visual perception



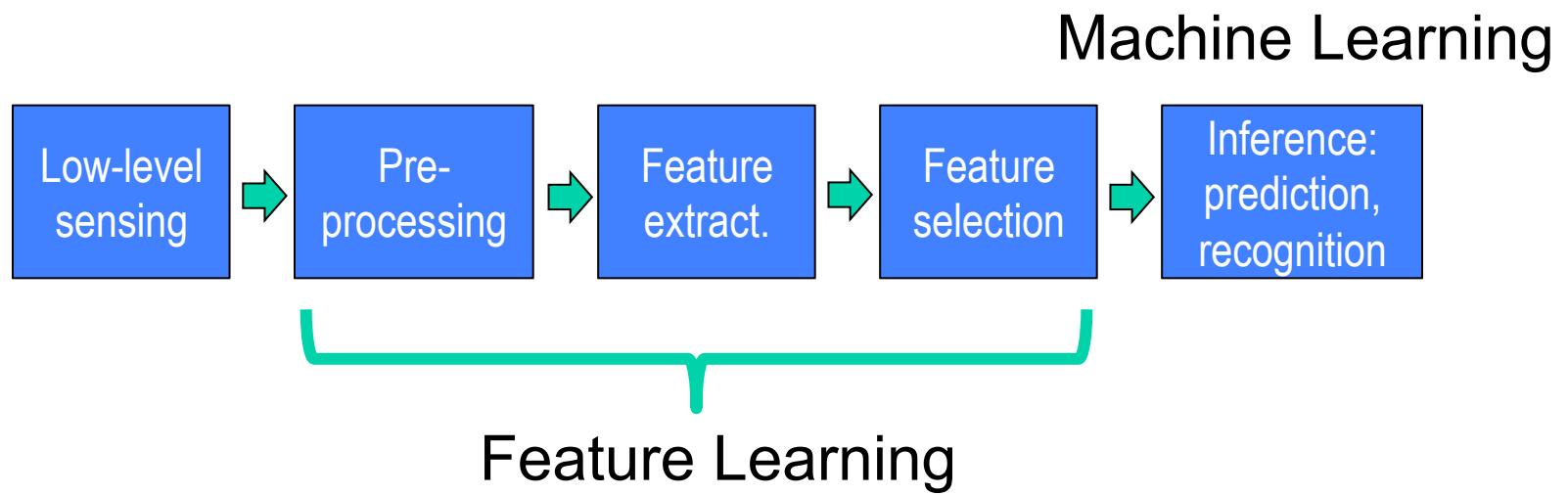
- Most critical for accuracy
- Account for most of the computation for testing
- Most time-consuming in development cycle
- Often hand-craft in practice

# Computer vision features

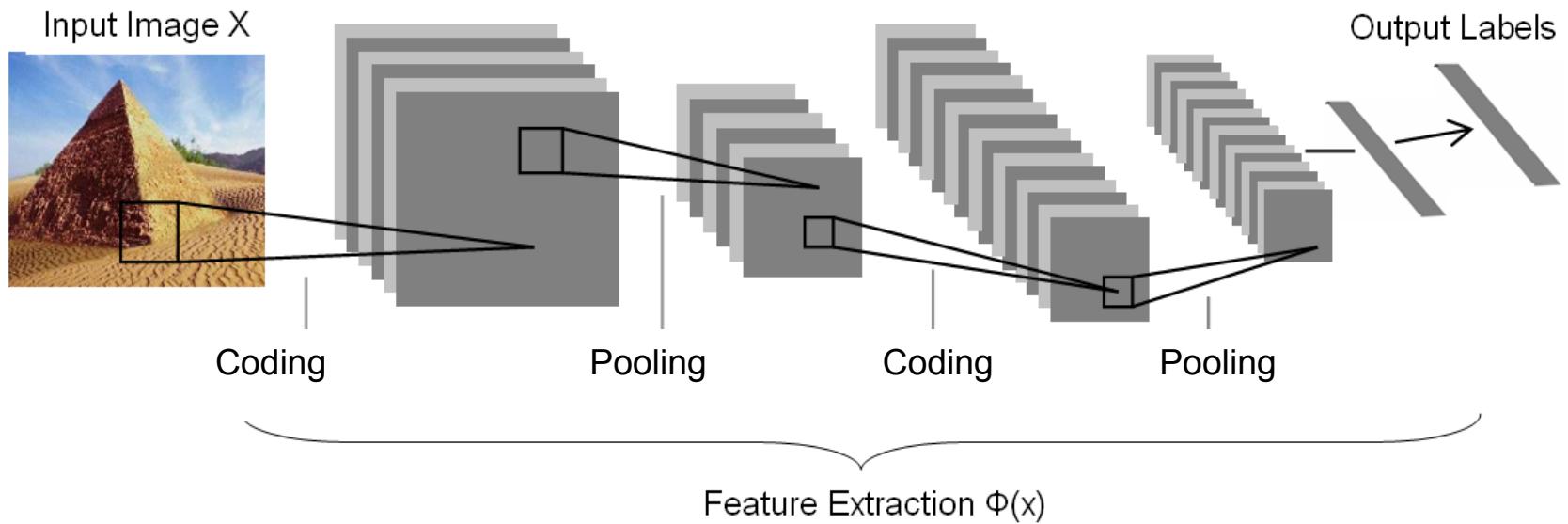


Slide Courtesy: Andrew Ng

# Learning features from data



# Convolution Neural Networks



Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1989.

# “Winter of Neural Networks” Since 90’s

- Non-convex
- Need a lot of tricks to play with
- Hard to do theoretical analysis

# **The Paradigm of Deep Learning**

# Deep Learning Since 2006

materials are identical for all configurations. The blue bars in Fig. 1 summarize the measured SHG signals. For excitation of the *LC* resonance in Fig. 1A (horizontal incident polarization), we find an SHG signal that is 500 times above the noise level. As expected for SHG, this signal closely scales with the square of the incident power (Fig. 2A). The polarization of the SHG emission is nearly vertical (Fig. 2B). The small angle with respect to the vertical is due to deviations from perfect mirror symmetry of the SRRs (see electron micrographs in Fig. 1). Small detuning of the *LC* resonance toward smaller wavelength (i.e., to 1.3- $\mu\text{m}$  wavelength) reduces the SHG signal strength from 100% to 20%. For excitation of the Mie resonance with vertical incident polarization in Fig. 1D, we find a small signal just above the noise level. For excitation of the Mie resonance with horizontal incident polarization in Fig. 1C, a small but significant SHG emission is found, which is again po-

## Reducing the Dimensionality of Data with Neural Networks

G. E. Hinton\* and R. R. Salakhutdinov

High-dimensional data can be converted to low-dimensional codes by training a multilayer neural network with a small central layer to reconstruct high-dimensional input vectors. Gradient descent can be used for fine-tuning the weights in such “autoencoder” networks, but this works well only if the initial weights are close to a good solution. We describe an effective way of initializing the weights that allows deep autoencoder networks to learn low-dimensional codes that work much better than principal components analysis as a tool to reduce the dimensionality of data.

Dimensionality reduction facilitates the classification, visualization, communication, and storage of high-dimensional data. A simple and widely used method is principal components analysis (PCA), which

finds the directions of greatest variance in the data set and represents each data point by its coordinates along each of these directions. We describe a nonlinear generalization of PCA that uses an adaptive, multilayer “encoder” network

Neural networks are coming back!

# Race on ImageNet (Top 5 Hit Rate)



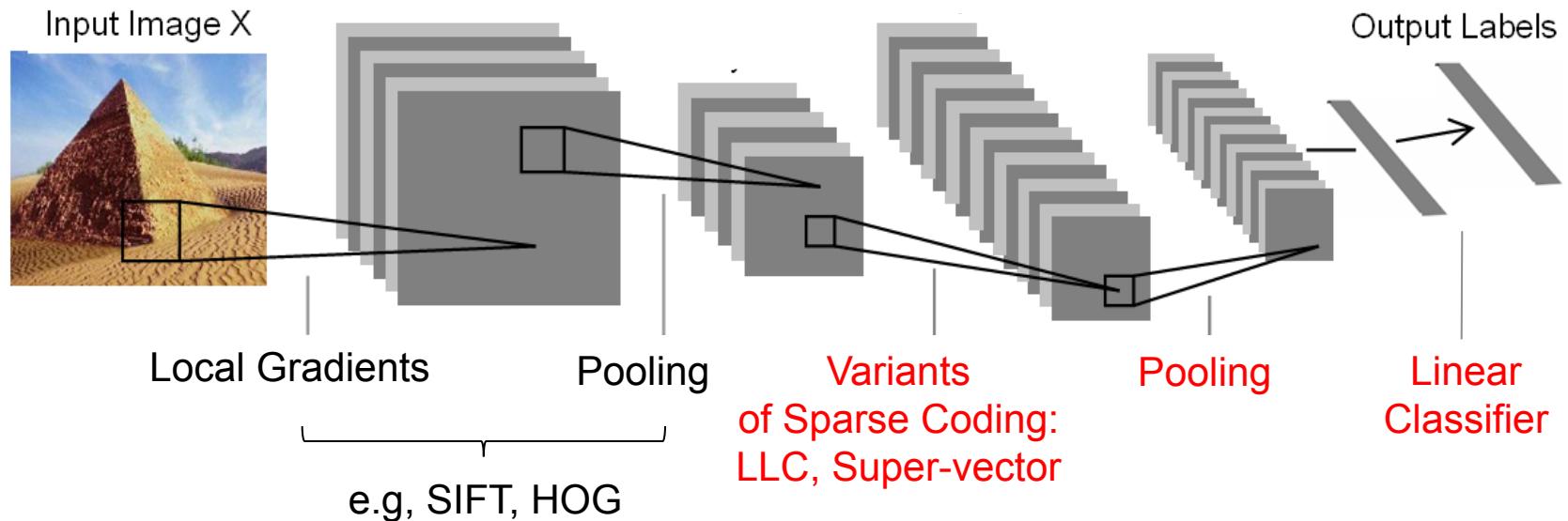
1000 object classes that we recognize

images courtesy of ImageNet (<http://www.image-net.org/challenges/LSVRC/2010/index>)

72%, 2010

74%, 2011

# The Best system on ImageNet (by 2012.10)



- This is a moderate deep model
- The first two layers are hand-designed

# Challenge to Deep Learners

Key questions:

- What if no hand-craft features at all?
- What if use much deeper neural networks?

Our chief critic, Jitendra Malik, has said that this competition is a good test of whether deep neural networks really do work well for object recognition.

-- By Geoff Hinton

# Answer from Geoff Hinton, 2012.10



1000 object classes that we recognize

Images courtesy of ImageNet (<http://www.image-net.org/challenges/LSVRC/2010/index>)

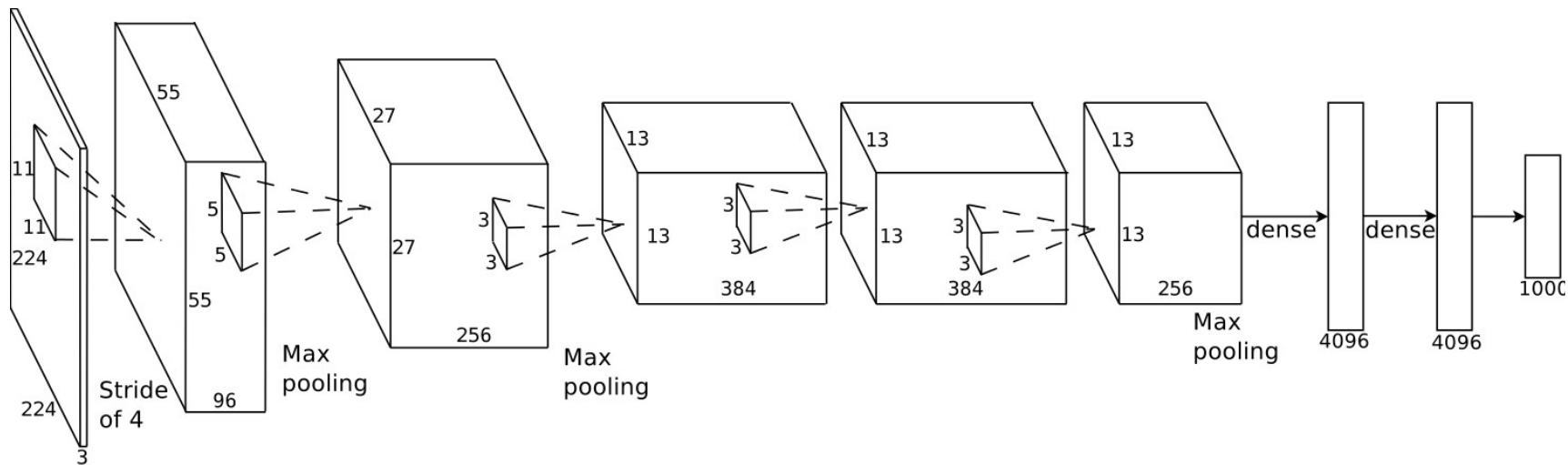
72%, 2010

74%, 2011

85%, 2012

# The Architecture

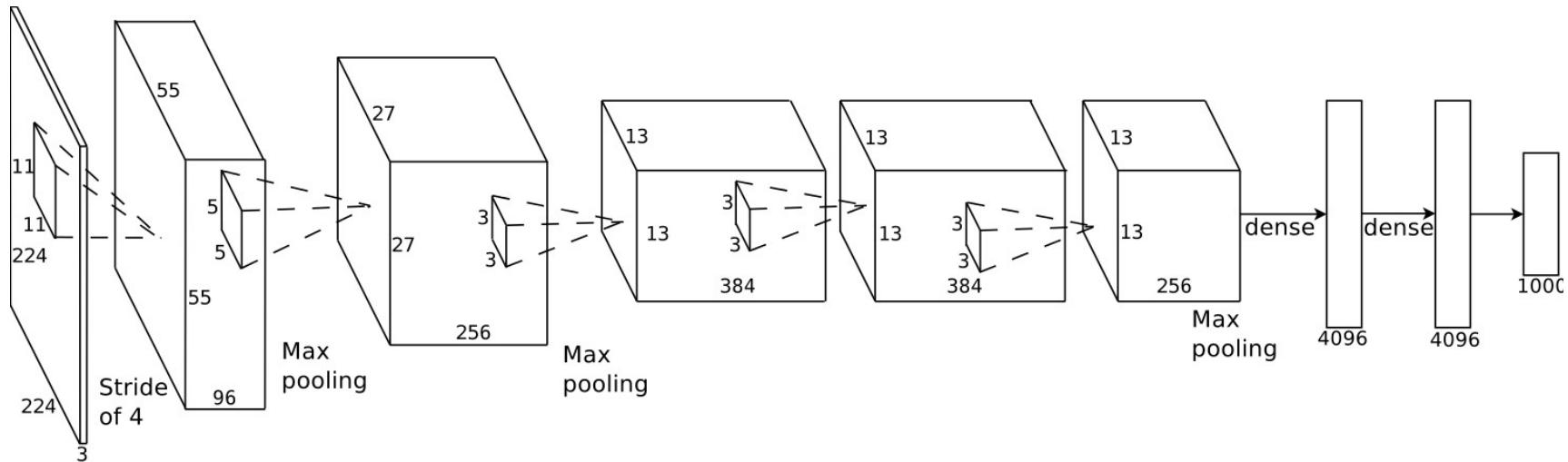
- Max-pooling layers follow first, second, and fifth convolutional layers
- The number of neurons in each layer is given by 253440, 186624, 64896, 64896, 43264, 4096, 4096, 1000



Slide Courtesy: Geoff Hinton

# The Architecture

- 7 hidden layers not counting max pooling.
- Early layers are conv., last two layers globally connected.
- Uses rectified linear units in every layer.
- Uses competitive normalization to suppress hidden activities.



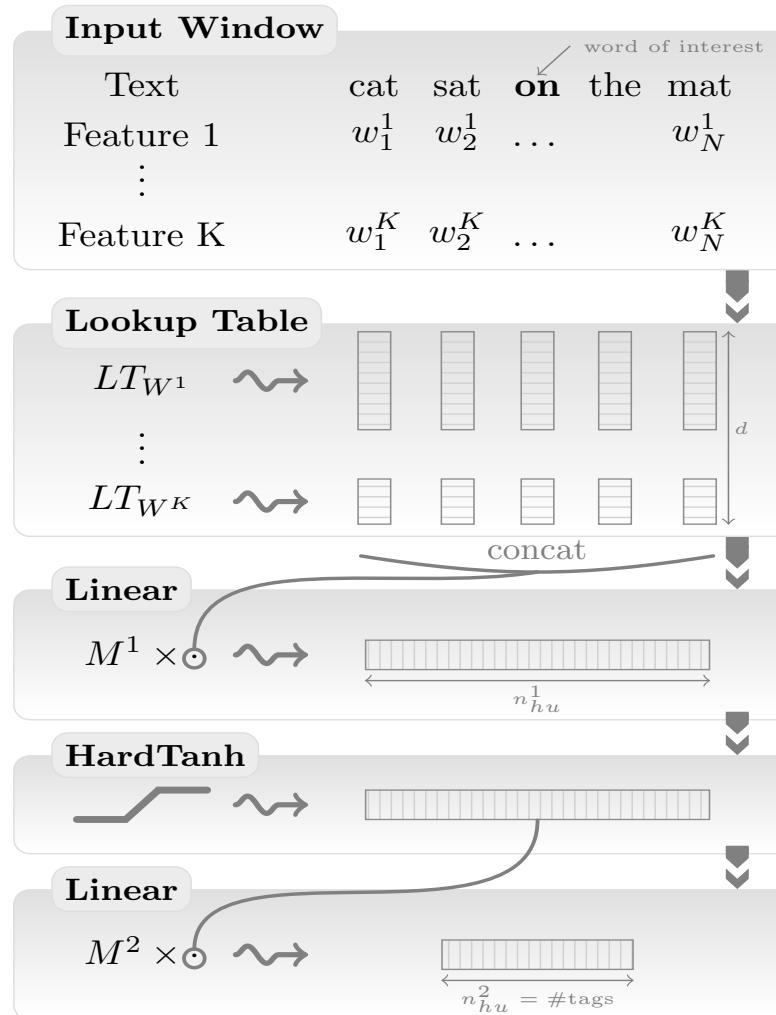
Slide Courtesy: Geoff Hinton

# Revolution on Speech Recognition

task	hours of training data	DNN-HMM	GMM-HMM with same data
Switchboard (test set 1)	309	18.5	27.4
Switchboard (test set 2)	309	16.1	23.6
English Broadcast News	50	17.5	18.8
Bing Voice Search (Sentence error rates)	24	30.4	36.2
Google Voice Input	5,870	12.3	
Youtube	1,400	47.6	52.3

Slide Courtesy: Geoff Hinton

# Deep Learning for NLP



Natural Language Processing (Almost) from Scratch, Collobert et al, JMLR 2011

# **Deep Learning in Industry**

- First successful deep learning models for speech recognition, by MSR in 2009
- Now deployed in MS products, e.g. Xbox



# “Google Brain” Project

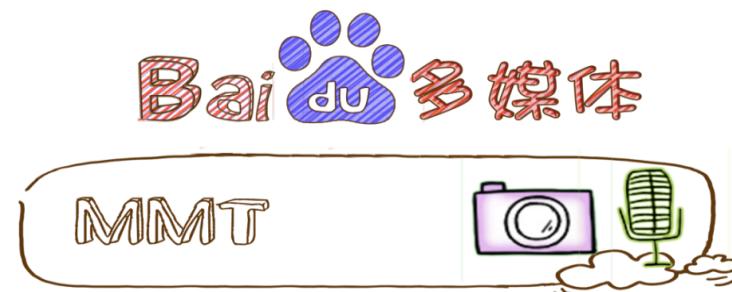
Ley by Google fellow Jeff Dean



- Published two papers, ICML2012, NIPS2012
- Company-wise large-scale deep learning infrastructure
- Big success on images, speech, NLPs

# Deep Learning @ Baidu

- Starts working on deep learning in 2012 summer
- Achieved big success in speech recognition and image recognition, both will be deployed into products in late November.
- Meanwhile, efforts are being carried on in areas like OCR, NLP, text retrieval, ranking, ...



# **Building Blocks of Deep Learning**

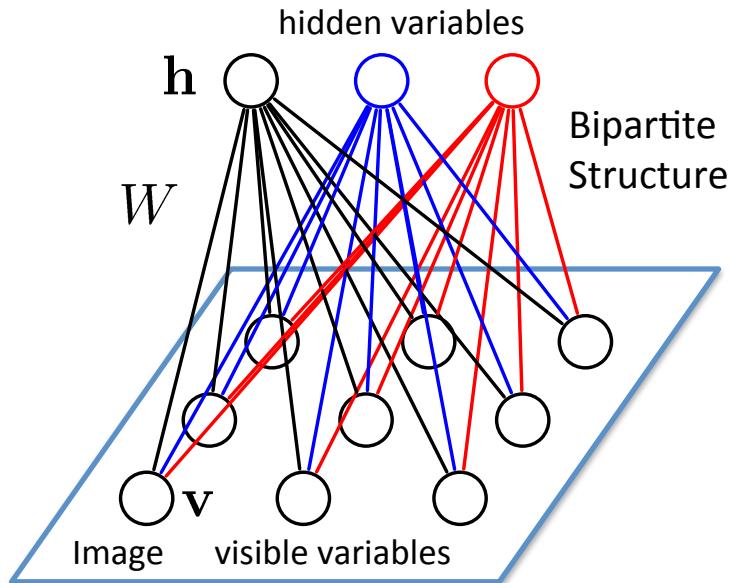
# CVPR 2012 Tutorial: Deep Learning for Vision

---

09.00am:	Introduction	Rob Fergus (NYU)
10.00am:	Coffee Break	
10.30am:	Sparse Coding	Kai Yu (Baidu)
11.30am:	Neural Networks	Marc'Aurelio Ranzato (Google)
12.30pm:	Lunch	
01.30pm:	Restricted Boltzmann Machines	Honglak Lee (Michigan)
02.30pm:	Deep Boltzmann Machines	Ruslan Salakhutdinov (Toronto)
03.00pm:	Coffee Break	
03.30pm:	Transfer Learning	Ruslan Salakhutdinov (Toronto)
04.00pm:	Motion & Video	Graham Taylor (Guelph)
05.00pm:	Summary / Q & A	All
05.30pm:	End	

# **Building Block I - RBM**

# Restricted Boltzmann Machine



Stochastic binary visible variables  $\mathbf{v} \in \{0, 1\}^D$  are connected to stochastic binary hidden variables  $\mathbf{h} \in \{0, 1\}^F$ .

The energy of the joint configuration:

$$E(\mathbf{v}, \mathbf{h}; \theta) = - \sum_{ij} W_{ij} v_i h_j - \sum_i b_i v_i - \sum_j a_j h_j$$

$\theta = \{W, a, b\}$  model parameters.

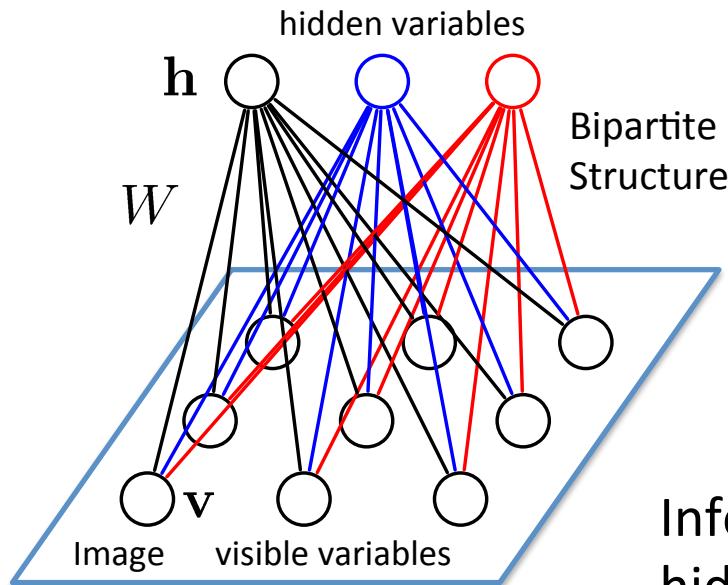
Probability of the joint configuration is given by the Boltzmann distribution:

$$P_\theta(\mathbf{v}, \mathbf{h}) = \frac{1}{Z(\theta)} \exp(-E(\mathbf{v}, \mathbf{h}; \theta)) = \frac{1}{Z(\theta)} \prod_{ij} e^{W_{ij} v_i h_j} \underbrace{\prod_i e^{b_i v_i}}_{\text{partition function}} \underbrace{\prod_j e^{a_j h_j}}_{\text{potential functions}}$$

$$Z(\theta) = \sum_{\mathbf{h}, \mathbf{v}} \exp(-E(\mathbf{v}, \mathbf{h}; \theta))$$

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

# Restricted Boltzmann Machine



**Restricted:** No interaction between hidden variables



Inferring the distribution over the hidden variables is easy:

$$P(\mathbf{h}|\mathbf{v}) = \prod_j P(h_j|\mathbf{v}) \quad P(h_j = 1|\mathbf{v}) = \frac{1}{1 + \exp(-\sum_i W_{ij}v_i - a_j)}$$

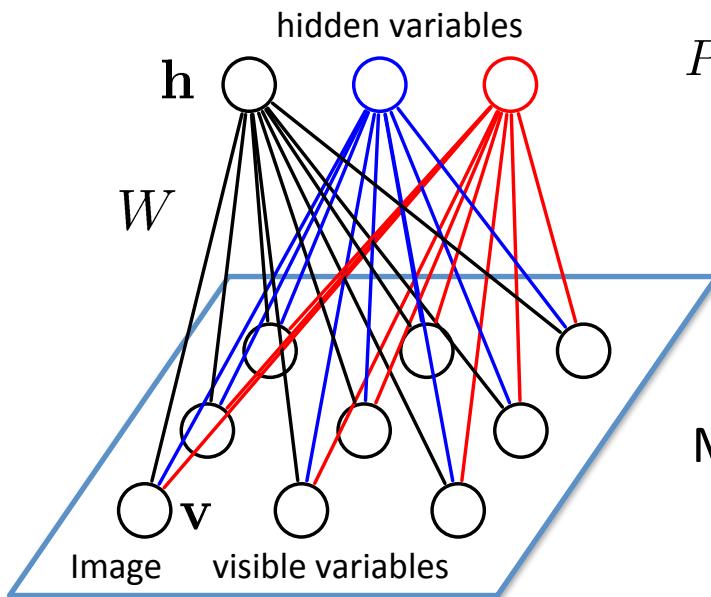
Factorizes: Easy to compute

Similarly:

$$P(\mathbf{v}|\mathbf{h}) = \prod_i P(v_i|\mathbf{h}) \quad P(v_i = 1|\mathbf{h}) = \frac{1}{1 + \exp(-\sum_j W_{ij}h_j - b_i)}$$

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

# Model Parameter Learning



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

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

Maximize (penalized) log-likelihood objective:

$$L(\theta) = \frac{1}{N} \sum_{n=1}^N \log P_{\theta}(\mathbf{v}^{(n)}) - \frac{\lambda}{N} \|W\|_F^2$$

Derivative of the log-likelihood:

$$\frac{\partial L(\theta)}{\partial W_{ij}} = \mathbb{E}_{P_{data}}[v_i h_j] - \mathbb{E}_{P_{\theta}}[v_i h_j] - \frac{2\lambda}{N} W_{ij}$$

Approximate maximum likelihood learning:

Contrastive Divergence (Hinton 2000)

MCMC-MLE estimator (Geyer 1991)

Tempered MCMC

(Salakhutdinov, NIPS 2009)

Pseudo Likelihood (Besag 1977)

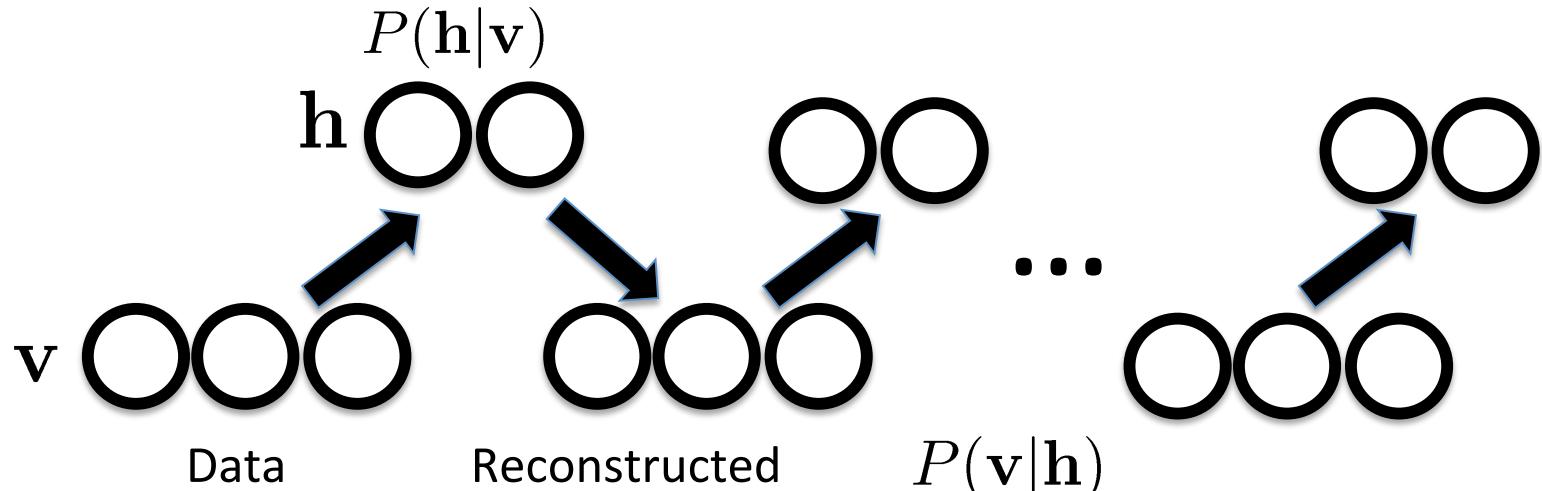
Composite Likelihoods (Lindsay, 1988; Varin 2008)

Adaptive MCMC

(Salakhutdinov, ICML 2010)

# Contrastive Divergence

Run Markov chain for a few steps (e.g. one step):



$$P(\mathbf{h}|\mathbf{v}) = \prod_j P(h_j|\mathbf{v}) \quad P(h_j = 1|\mathbf{v}) = \frac{1}{1 + \exp(-\sum_i W_{ij} v_i - a_j)}$$

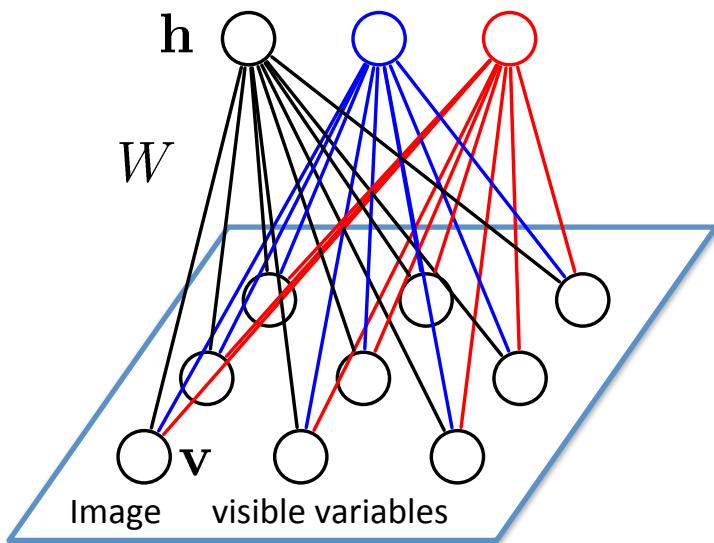
$$P(\mathbf{v}|\mathbf{h}) = \prod_i P(v_i|\mathbf{h}) \quad P(v_i = 1|\mathbf{h}) = \frac{1}{1 + \exp(-\sum_j W_{ij} h_j - b_i)}$$

Update model parameters:

$$\Delta W_{ij} = \mathbb{E}_{P_{data}}[v_i h_j] - \mathbb{E}_{P_1}[v_i h_j]$$

# RBM<sup>s</sup> for Images

Gaussian-Bernoulli RBM:



$$P_{\theta}(\mathbf{v}, \mathbf{h}) = \frac{1}{Z(\theta)} \exp(-E(\mathbf{v}, \mathbf{h}; \theta))$$

Interpretation: Mixture of exponential number of Gaussians

$$P_{\theta}(\mathbf{v}) = \sum_{\mathbf{h}} P_{\theta}(\mathbf{v}|\mathbf{h}) P_{\theta}(\mathbf{h}),$$

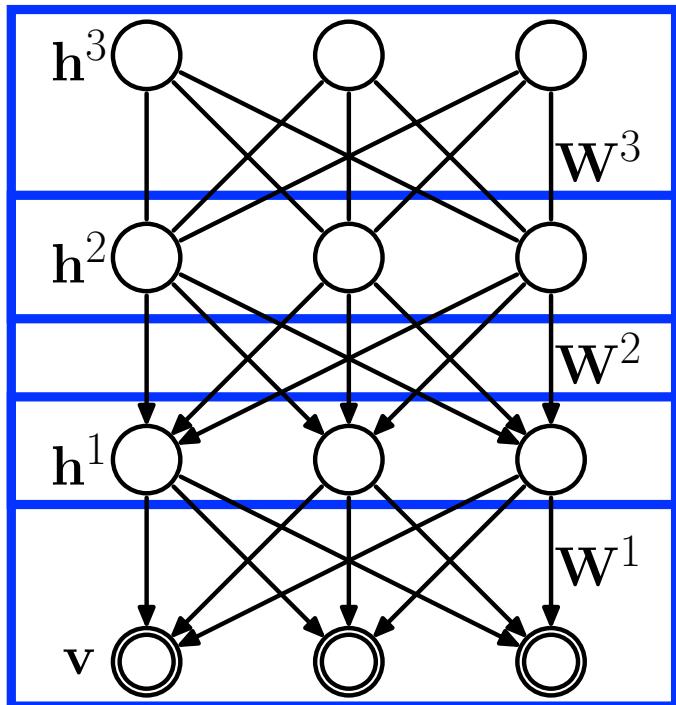
where

$$P_{\theta}(\mathbf{h}) = \int_{\mathbf{v}} P_{\theta}(\mathbf{v}, \mathbf{h}) d\mathbf{v} \quad \text{is an implicit prior, and}$$

$$P(v_i = x | \mathbf{h}) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(-\frac{(x - b_i - \sigma_i \sum_j W_{ij} h_j)^2}{2\sigma_i^2}\right) \quad \text{Gaussian}$$

# Layerwise Pre-training

Deep Belief Network



Efficient layer-wise pretraining algorithm.

$$\begin{aligned} \log P_\theta(\mathbf{v}) &= \sum_{\mathbf{h}^1} P_\theta(\mathbf{v}, \mathbf{h}^1) \geq \sum_{\mathbf{h}^1} Q_\phi(\mathbf{h}^1 | \mathbf{v}) \log \frac{P_\theta(\mathbf{h}^1, \mathbf{v})}{Q_\phi(\mathbf{h}^1 | \mathbf{v})} \\ &= \sum_{\mathbf{h}^1} Q_\phi(\mathbf{h}^1 | \mathbf{v}) \left[ \underbrace{\log P_\theta(\mathbf{v} | \mathbf{h}^1; W^1)}_{\text{Likelihood term}} + \mathcal{H}(Q_\phi(\mathbf{h}^1 | \mathbf{v})) \right] \\ &\quad + \sum_{\mathbf{h}^1} Q_\phi(\mathbf{h}^1 | \mathbf{v}) \log P_\theta(\mathbf{h}^1; W^2) \end{aligned}$$

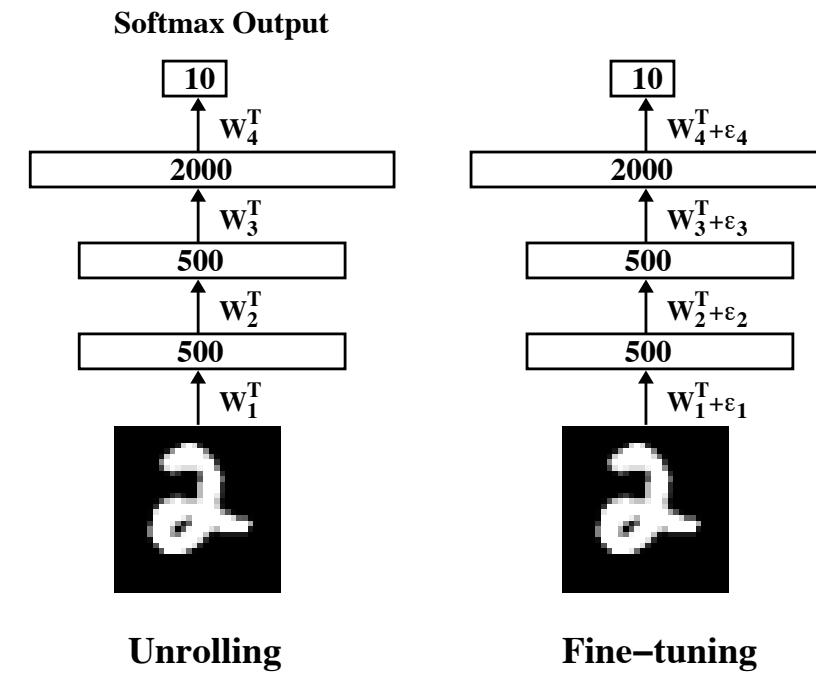
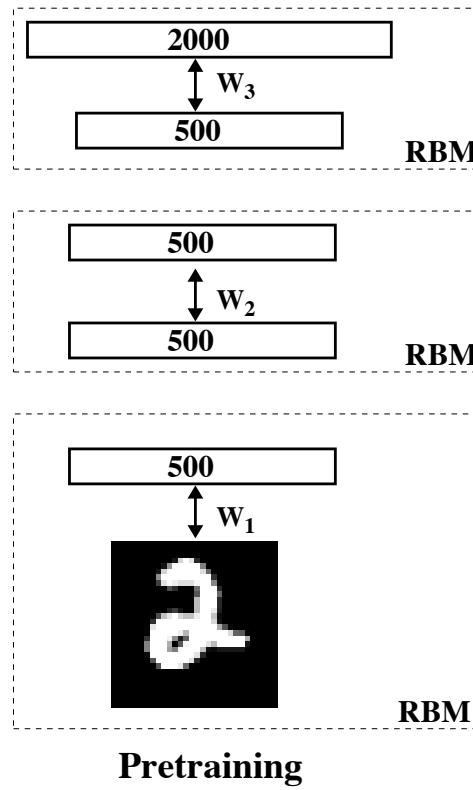
Variational Lower Bound

Entropy functional

Replace with a second layer RBM

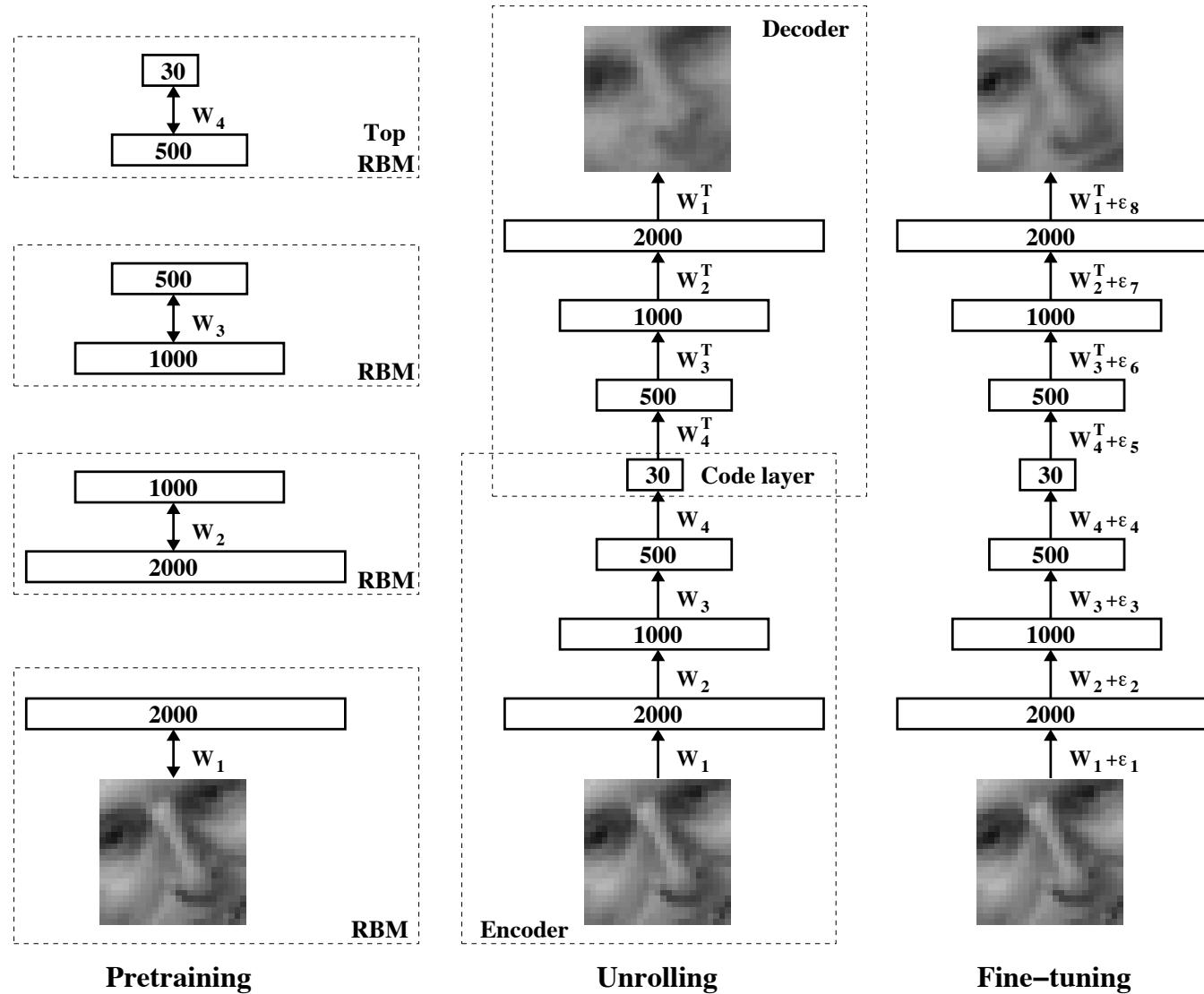
Similar arguments for pretraining a Deep Boltzmann machine

# DBNs for MNIST Classification



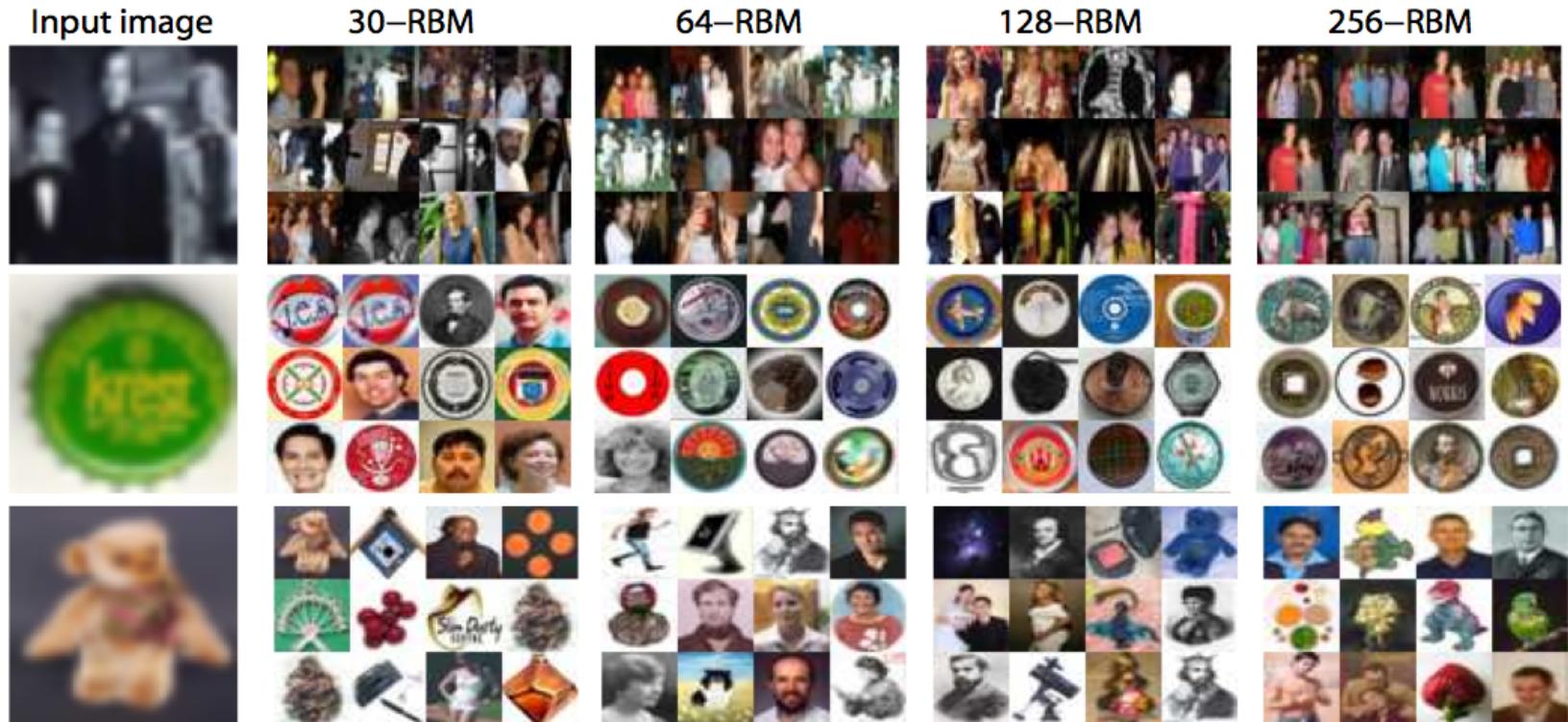
- After layer-by-layer **unsupervised pretraining**, discriminative fine-tuning by backpropagation achieves an error rate of 1.2% on MNIST. SVM's get 1.4% and randomly initialized backprop gets 1.6%.

# Deep Autoencoders for Unsupervised Feature Learning



# Image Retrieval using Binary Codes

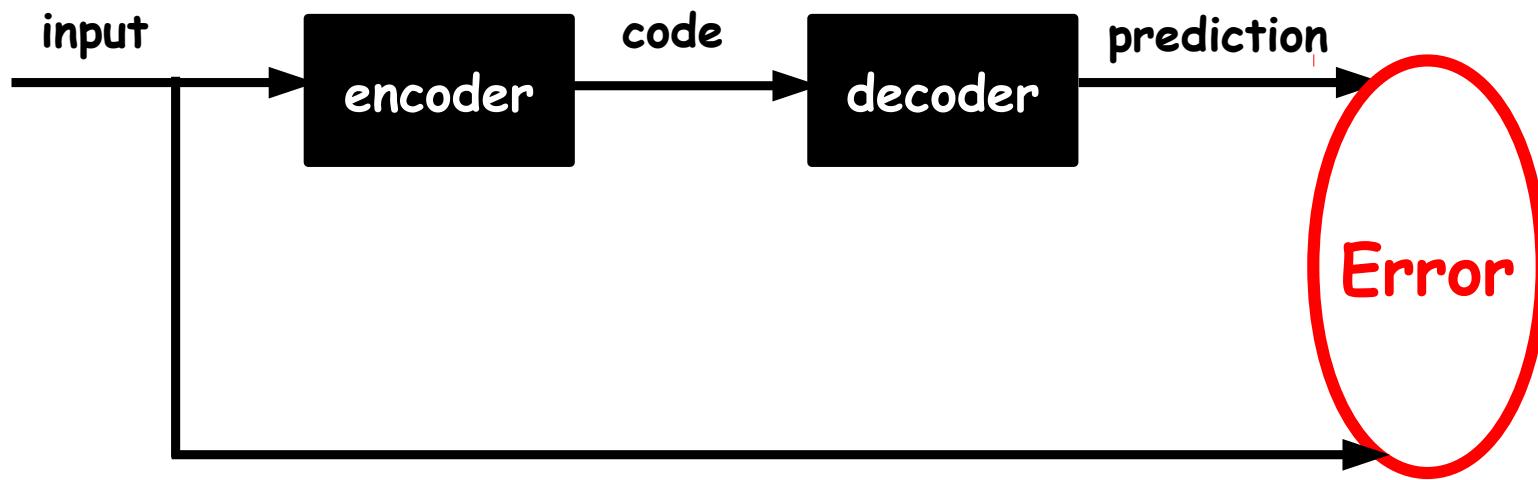
- Map images into binary codes for fast retrieval.



- Small Codes, Torralba, Fergus, Weiss, CVPR 2008
- Spectral Hashing, Y. Weiss, A. Torralba, R. Fergus, NIPS 2008
- Kulis and Darrell, NIPS 2009, Gong and Lazebnik, CVPR 2011
- Norouzi and Fleet, ICML 2011,

## **Building Block 2 – Autoencoder Neural Net**

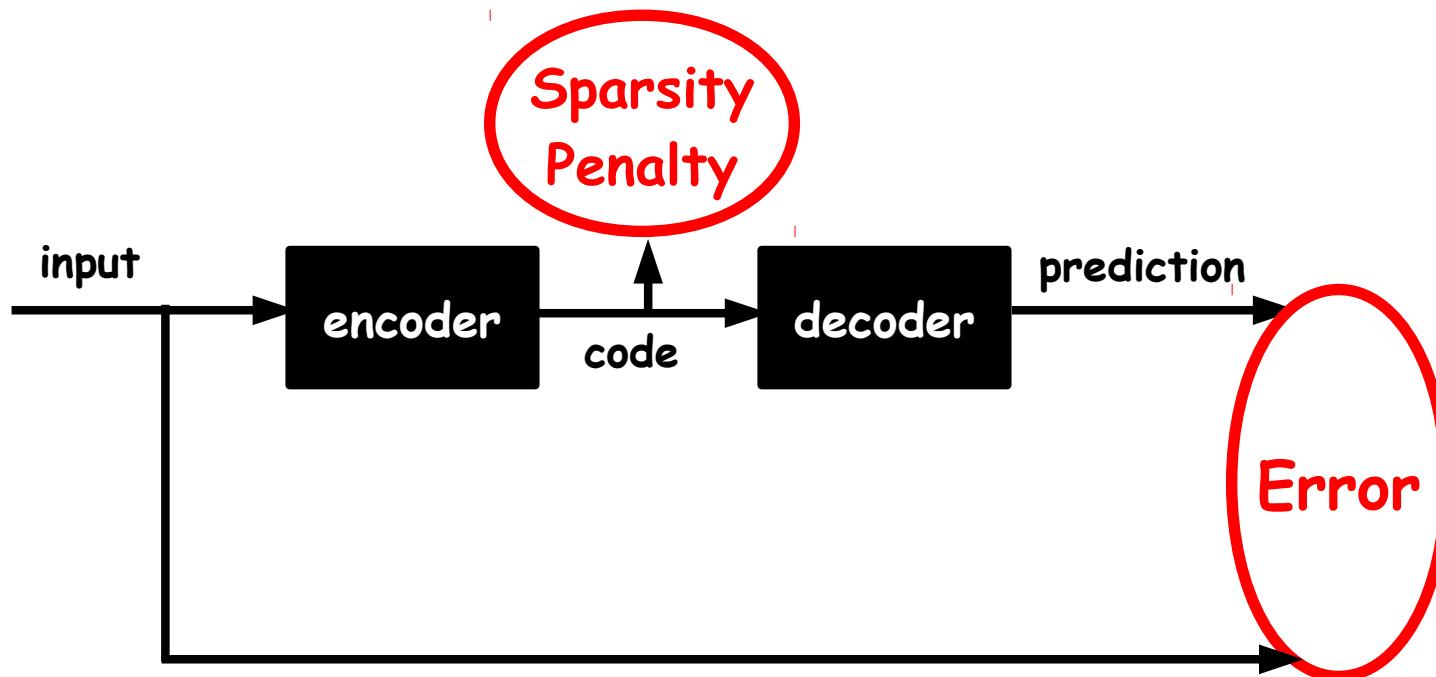
# Autoencoder Neural Network



- input higher dimensional than code
- error:  $\|\text{prediction} - \text{input}\|^2$
- training: back-propagation

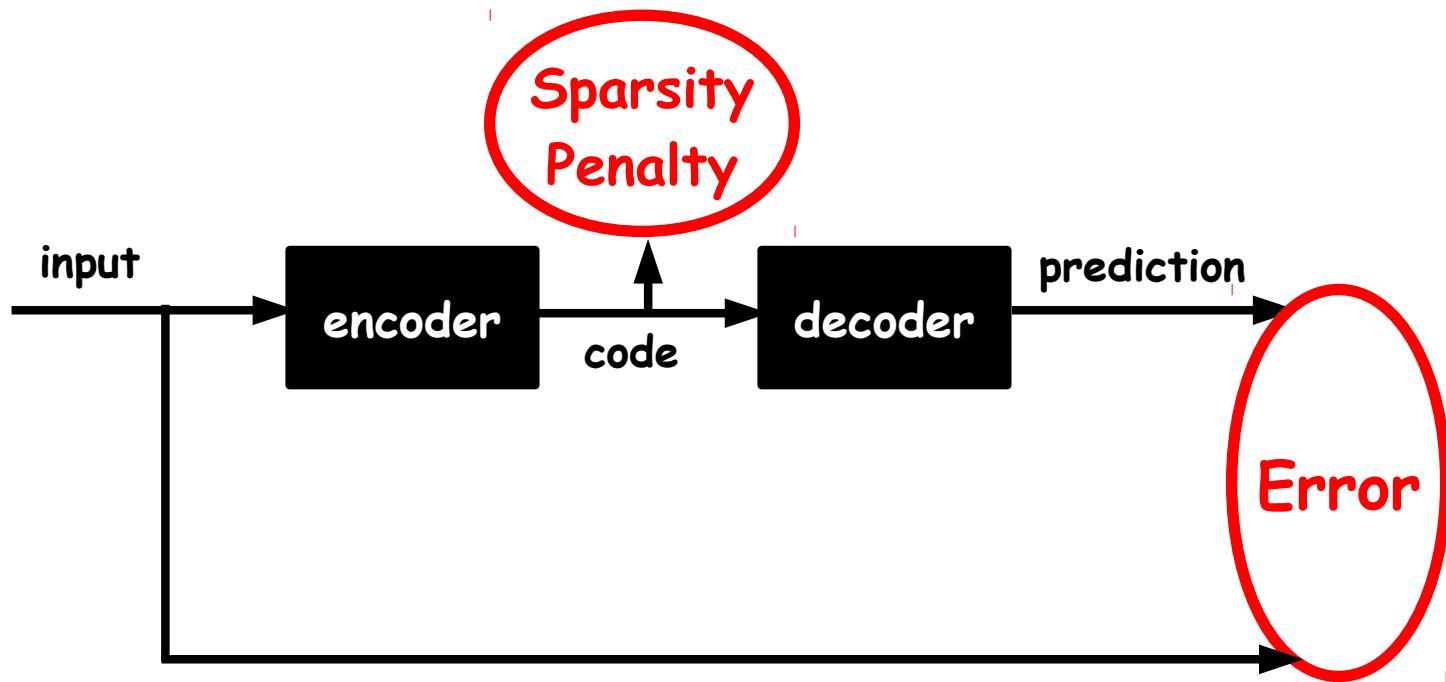
116

# Sparse Autoencoder



- sparsity penalty:  $\|\text{code}\|_1$
- error:  $\|\text{prediction} - \text{input}\|^2$
- loss: sum of square reconstruction error and sparsity
- training: back-propagation

# Sparse Autoencoder



- input:  $X$  code:  $h = W^T X$

- loss:  $L(X; W) = \|W h - X\|^2 + \lambda \sum_j |h_j|$

*Le et al. "ICA with reconstruction cost.." NIPS 2011*

## **Building Block 3 – Sparse Coding**

# Sparse coding

Sparse coding (Olshausen & Field, 1996). Originally developed to explain early visual processing in the brain (edge detection).

**Training:** given a set of random patches  $x$ , learning a dictionary of bases  $[\Phi_1, \Phi_2, \dots]$

**Coding:** for data vector  $x$ , solve LASSO to find the sparse coefficient vector  $a$

$$\min_{a, \phi} \sum_{i=1}^m \left\| x_i - \sum_{j=1}^k a_{i,j} \phi_j \right\|^2 + \lambda \sum_{i=1}^m \sum_{j=1}^k |a_{i,j}|$$

# Sparse coding: training time

Input: Images  $x_1, x_2, \dots, x_m$  (each in  $\mathbb{R}^d$ )

Learn: Dictionary of bases  $\phi_1, \phi_2, \dots, \phi_k$  (also  $\mathbb{R}^d$ ).

$$\min_{a, \phi} \sum_{i=1}^m \left\| x_i - \sum_{j=1}^k a_{i,j} \phi_j \right\|^2 + \lambda \sum_{i=1}^m \sum_{j=1}^k |a_{i,j}|$$

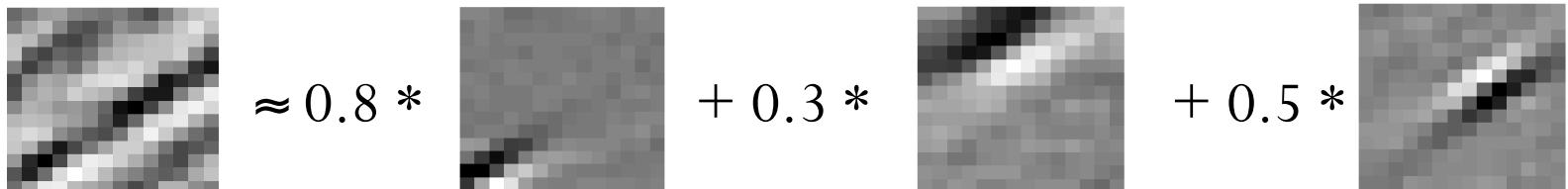
Alternating optimization:

1. Fix dictionary  $\phi_1, \phi_2, \dots, \phi_k$ , optimize  $a$  (a standard LASSO problem)
2. Fix activations  $a$ , optimize dictionary  $\phi_1, \phi_2, \dots, \phi_k$  (a convex QP problem)

# Sparse coding: testing time

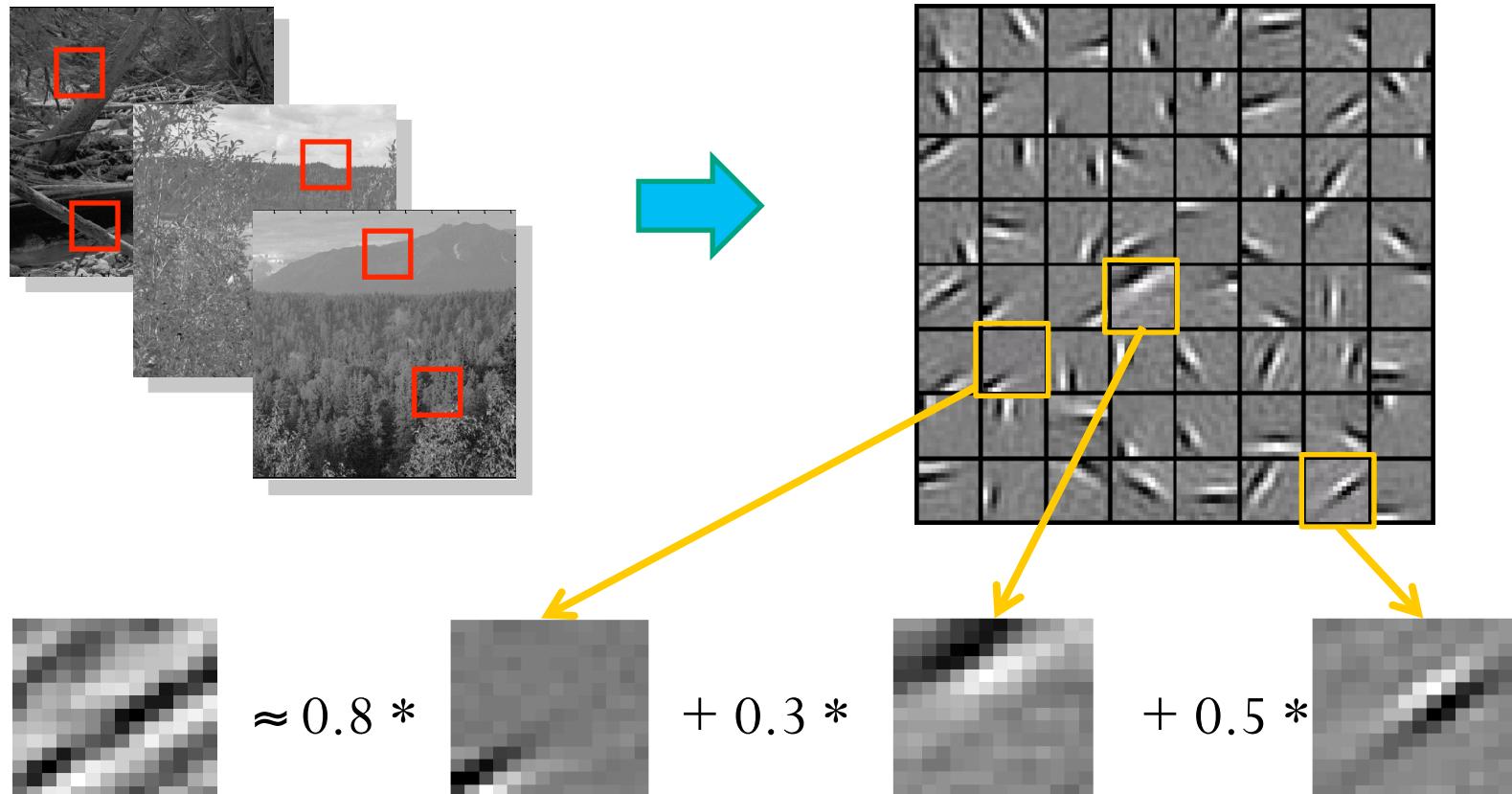
Input: Novel image patch  $x$  (in  $R^d$ ) and previously learned  $\phi_i$ 's  
Output: Representation  $[a_{i,1}, a_{i,2}, \dots, a_{i,k}]$  of image patch  $x_i$ .

$$\min_a \sum_{i=1}^m \left\| x_i - \sum_{j=1}^k a_{i,j} \phi_j \right\|^2 + \lambda \sum_{i=1}^m \sum_{j=1}^k |a_{i,j}|$$



Represent  $x_i$  as:  $a_i = [0, 0, \dots, 0, \mathbf{0.8}, 0, \dots, 0, \mathbf{0.3}, 0, \dots, 0, \mathbf{0.5}, \dots]$

# Sparse coding illustration



$[a_1, \dots, a_{64}] = [0, 0, \dots, 0, \mathbf{0.8}, 0, \dots, 0, \mathbf{0.3}, 0, \dots, 0, \mathbf{0.5}, 0]$   
(feature representation)

# RBM & autoencoders

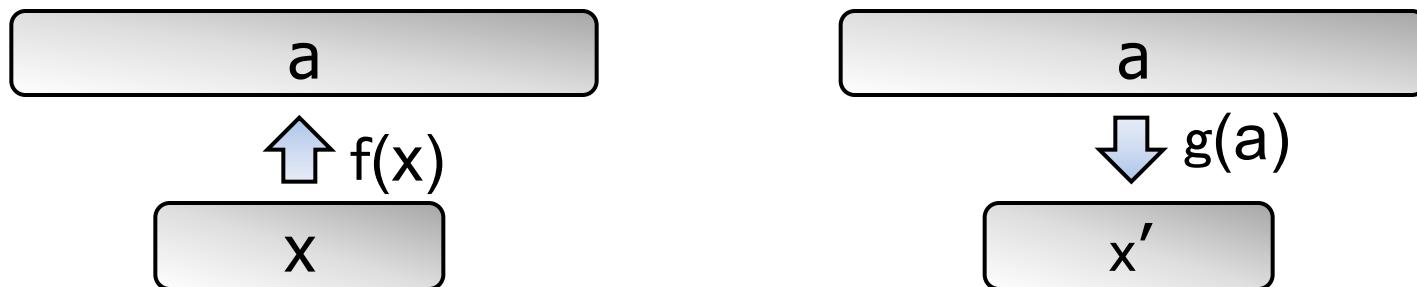
- also involve activation and reconstruction
- but have explicit  $f(x)$
- not necessarily enforce sparsity on  $a$
- but if put sparsity on  $a$ , often get improved results  
[e.g. sparse RBM, Lee et al. NIPS08]



# Sparse coding: A broader view

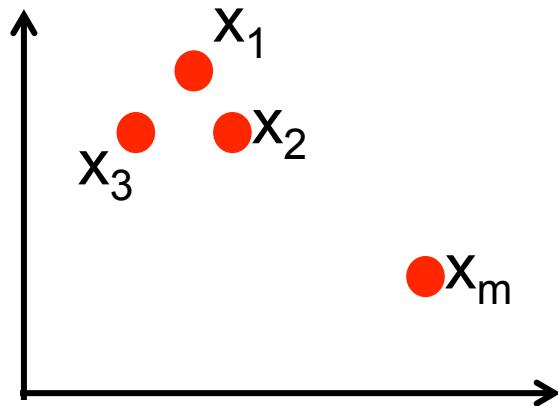
Any feature mapping from  $x$  to  $a$ , i.e.  $a = f(x)$ , where

- $a$  is sparse (and often higher dim. than  $x$ )
- $f(x)$  is nonlinear
- reconstruction  $x' = g(a)$ , such that  $x' \approx x$



Therefore, sparse RBMs, sparse auto-encoder, even VQ can be viewed as a form of sparse coding.

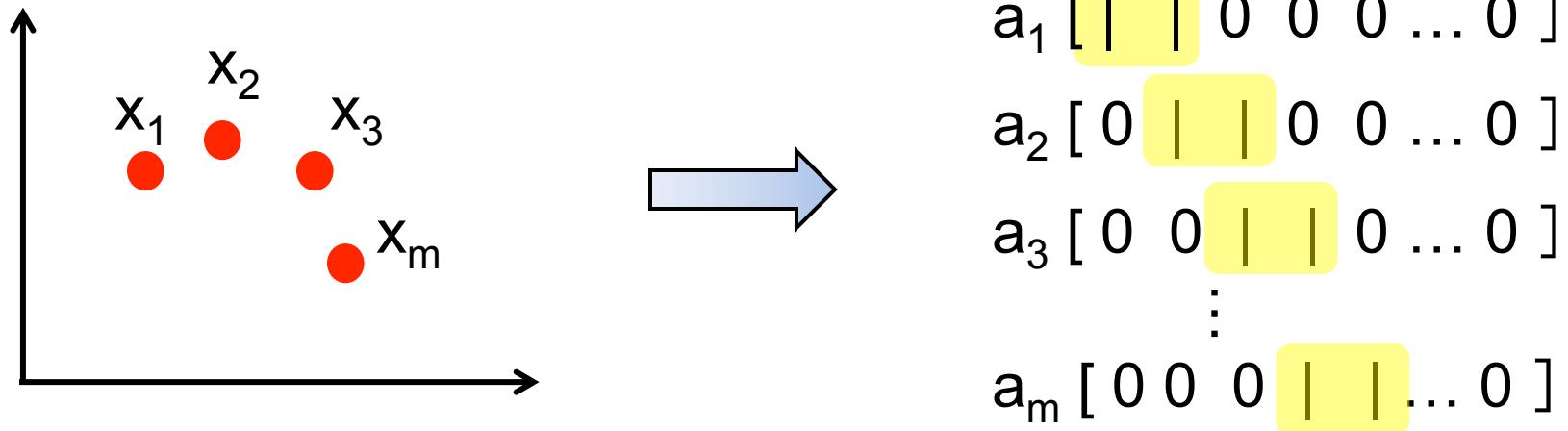
# Example of sparse activations (sparse coding)



$a_1$	[	0			0	0	...	0
$a_2$	[			0	0	0	...	0
$a_3$	[		0		0	0	...	0
$a_m$	[	0	0	0			...	0

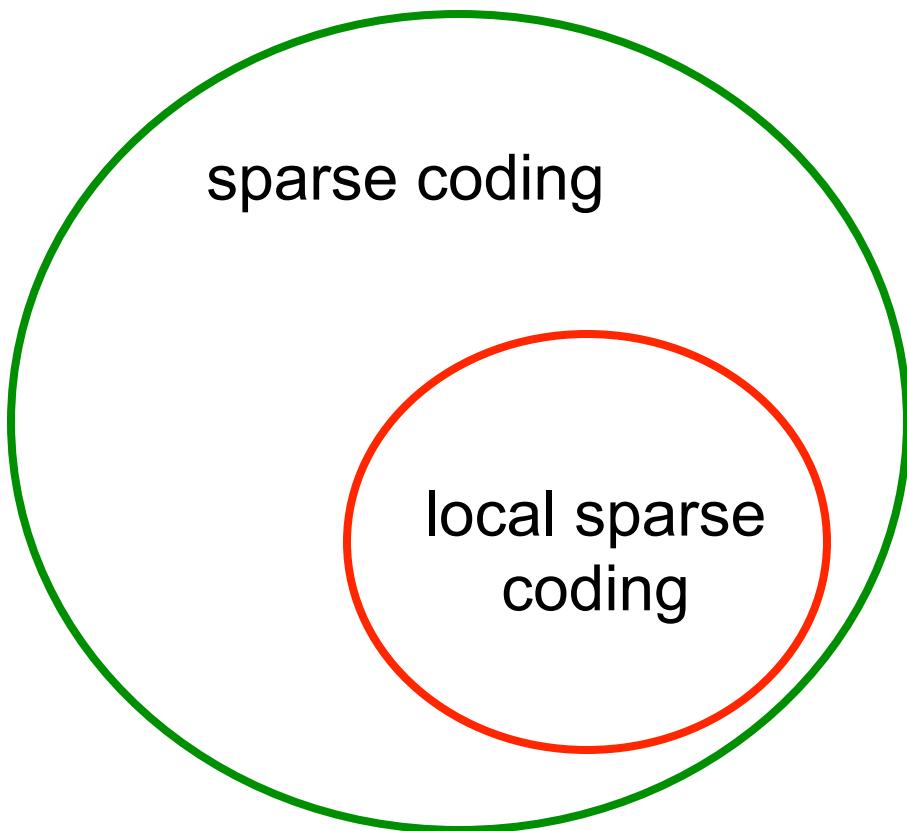
- different  $x$  has different dimensions activated
- **locally-shared sparse representation:** similar  $x$ 's tend to have similar non-zero dimensions

# Example of sparse activations (sparse coding)



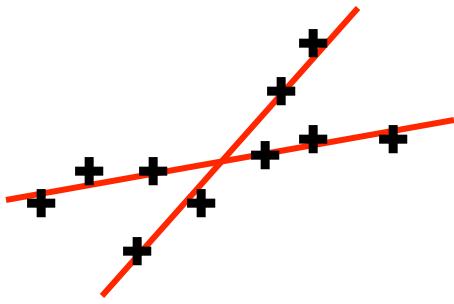
- another example: preserving manifold structure
- more informative in highlighting richer data structures, i.e. clusters, manifolds,

# Sparsity vs. Locality



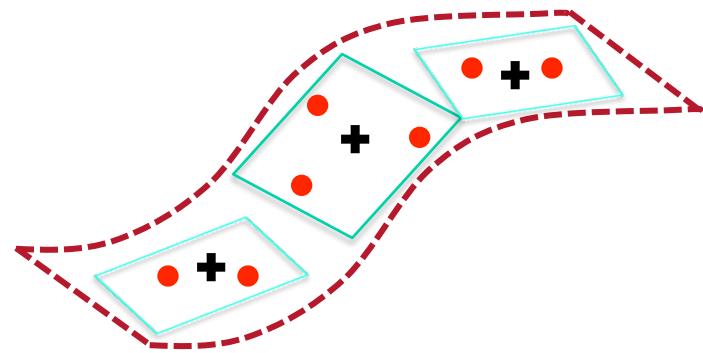
- **Intuition:** similar data should get similar activated features
- **Local sparse coding:**
  - data in the same neighborhood tend to have shared activated features;
  - data in different neighborhoods tend to have different features activated.

# Sparse coding is not always local: example



Case 1  
independent subspaces

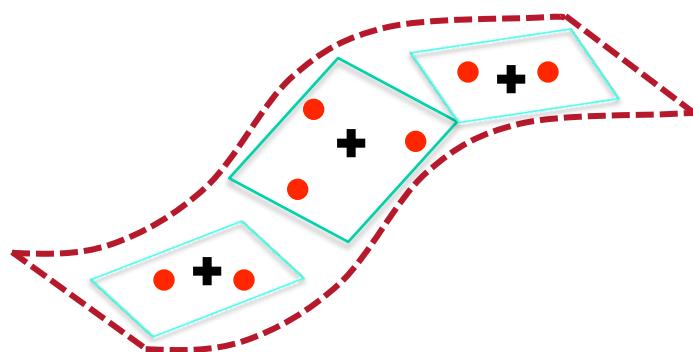
- Each basis is a “**direction**”
- **Sparsity**: each datum is a linear combination of only several bases.



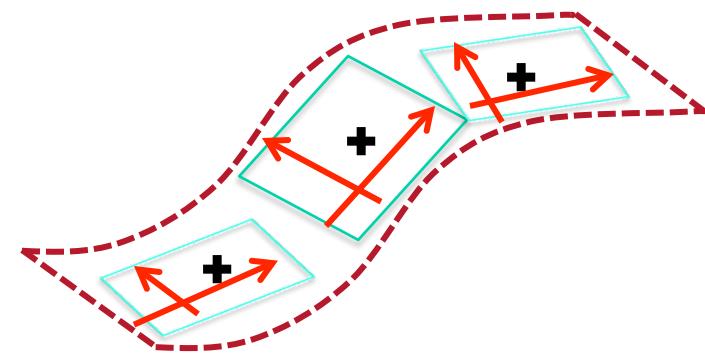
Case 2  
data manifold (or clusters)

- Each basis an “**anchor point**”
- **Sparsity**: each datum is a linear combination of neighbor anchors.
- Sparsity is caused by locality.

# Two approaches to local sparse coding



Approach 1  
Coding via local anchor points



Approach 2  
Coding via local subspaces

## Local coordinate coding

Learning locality-constrained linear coding for image classification, Jingjun Wang, Jianchao Yang, Kai Yu, Fengjun Lv, Thomas Huang. In **CVPR 2010**.

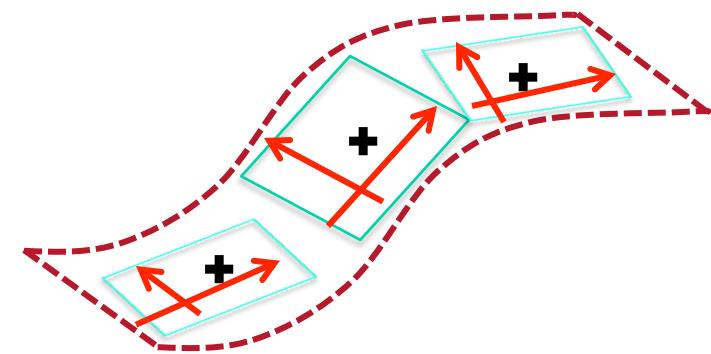
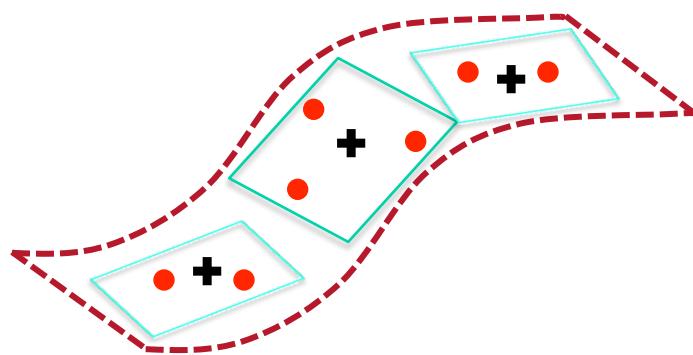
Nonlinear learning using local coordinate coding, Kai Yu, Tong Zhang, and Yihong Gong. In **NIPS 2009**.

## Super-vector coding

Image Classification using Super-Vector Coding of Local Image Descriptors, Xi Zhou, Kai Yu, Tong Zhang, and Thomas Huang. In **ECCV 2010**.

Large-scale Image Classification: Fast Feature Extraction and SVM Training, Yuanqing Lin, Fengjun Lv, Shenghuo Zhu, Ming Yang, Timothee Cour, Kai Yu, LiangLiang Cao, Thomas Huang. In **CVPR 2011**

# Two approaches to local sparse coding



Approach 1

Coding via local anchor points

Approach 2

Coding via local subspaces

Local coordinate coding

Super-vector coding

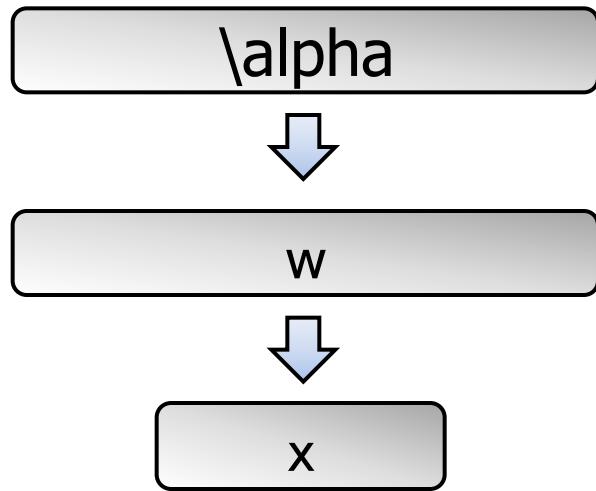
- Sparsity achieved by explicitly ensuring locality
- Sound theoretical justifications
- Much simpler to implement and compute
- Strong empirical success

# Hierarchical Sparse Coding

Yu, Lin, & Lafferty, CVPR 11

$$(\widehat{W}, \widehat{\alpha}) = \arg \min_{W, \alpha} L(W, \alpha) + \frac{\lambda_1}{n} \|W\|_1 + \gamma \|\alpha\|_1$$

subject to  $\alpha \succeq 0$ ,



$$L(W, \alpha) =$$

$$\frac{1}{n} \sum_{i=1}^n \left\{ \frac{1}{2} \|x_i - Bw_i\|^2 + \lambda_2 w_i^\top \Omega(\alpha) w_i \right\}$$

$$\Omega(\alpha) \equiv \left( \sum_{k=1}^q \alpha_k \text{diag}(\phi_k) \right)^{-1}$$

# Hierarchical Sparse Coding on MNIST

Yu, Lin, & Lafferty, CVPR 11

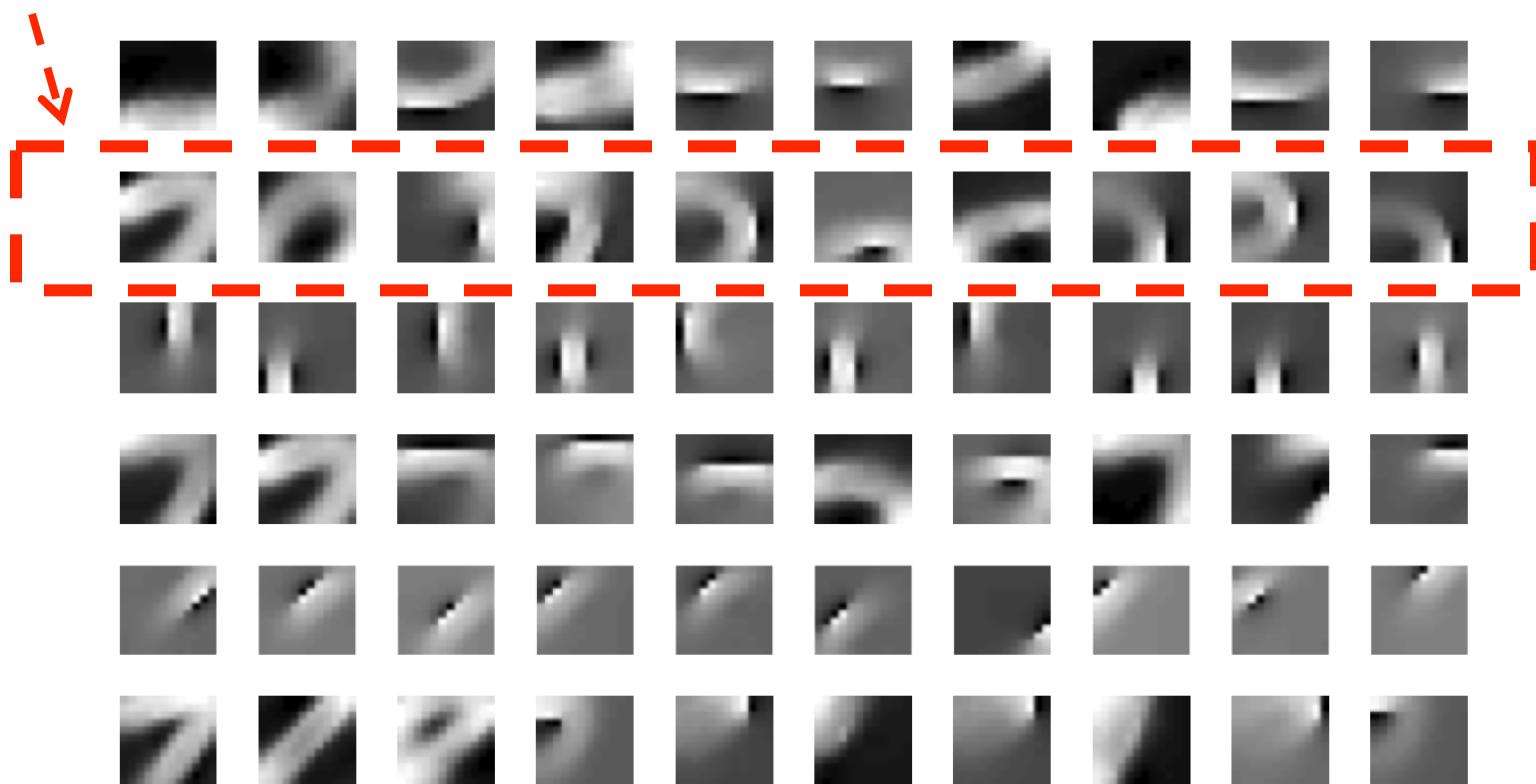
Methods	Error rate (%)
Sparse coding (unsupervised)	2.10
Local coordinate coding (unsupervised) [21]	1.90
Extended local coordinate coding (unsupervised) [21]	1.64
Differentiable sparse coding (supervised) [5]	1.30
Discriminative sparse coding (supervised) [15]	1.05
One-layer sparse coding (unsupervised)	0.98
Convolutional neural network (supervised) [11]	0.82
<b>Hierarchical sparse coding</b> (unsupervised)	<b>0.77</b>

**HSC vs. CNN:** HSC provide even better performance than CNN  
☺☺☺ more amazingly, HSC learns features in **unsupervised** manner!

# Second-layer dictionary

Yu, Lin, & Lafferty, CVPR 11

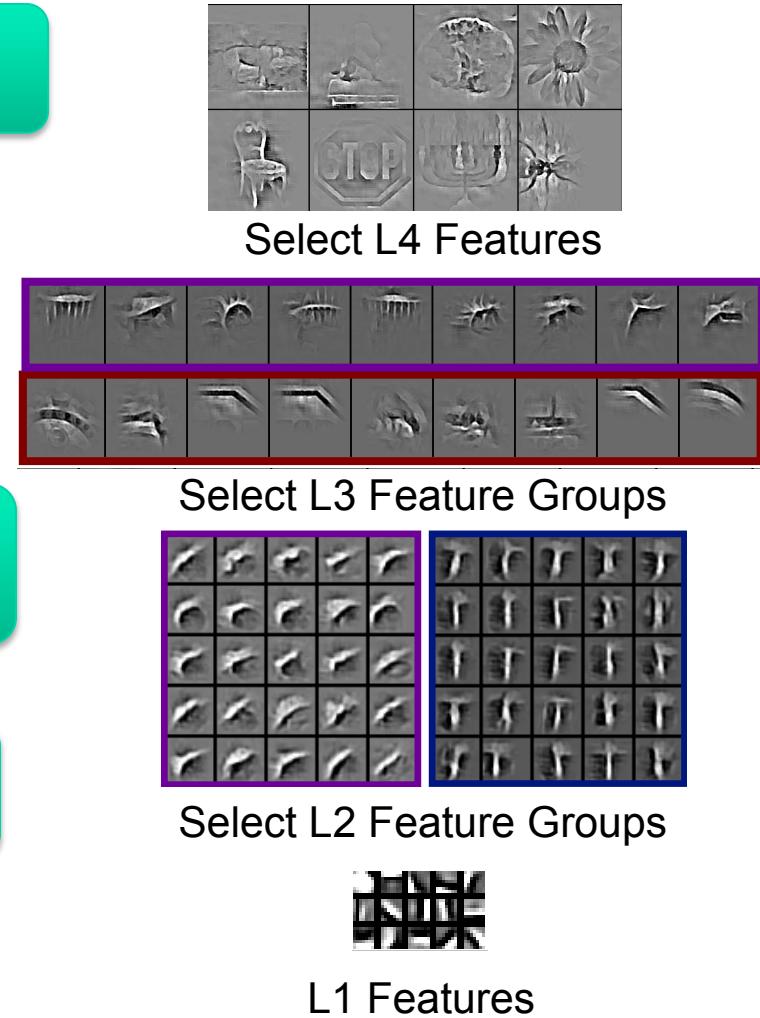
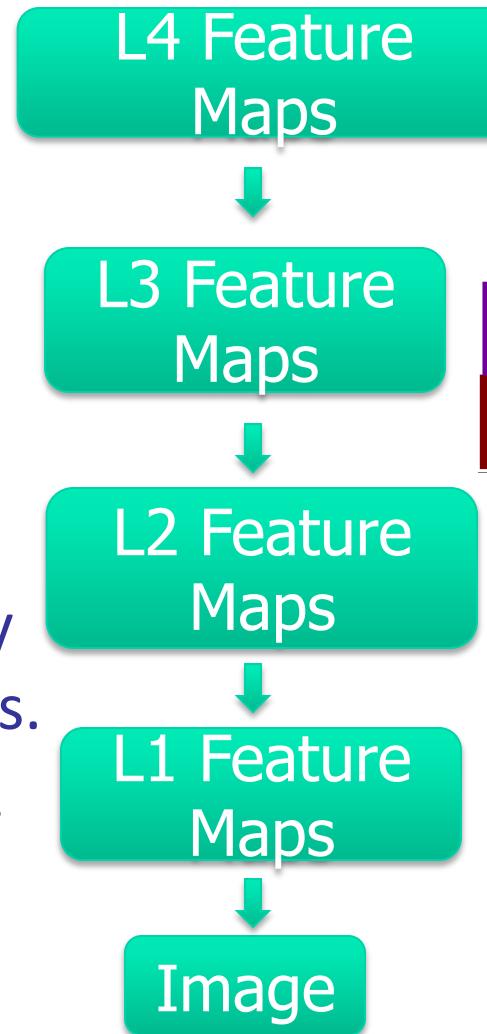
A hidden unit in the second layer is connected to a unit group in the 1<sup>st</sup> layer: **invariance to translation, rotation, and deformation**



# Adaptive Deconvolutional Networks for Mid and High Level Feature Learning

Matthew D. Zeiler, Graham W. Taylor, and Rob Fergus, ICCV 2011

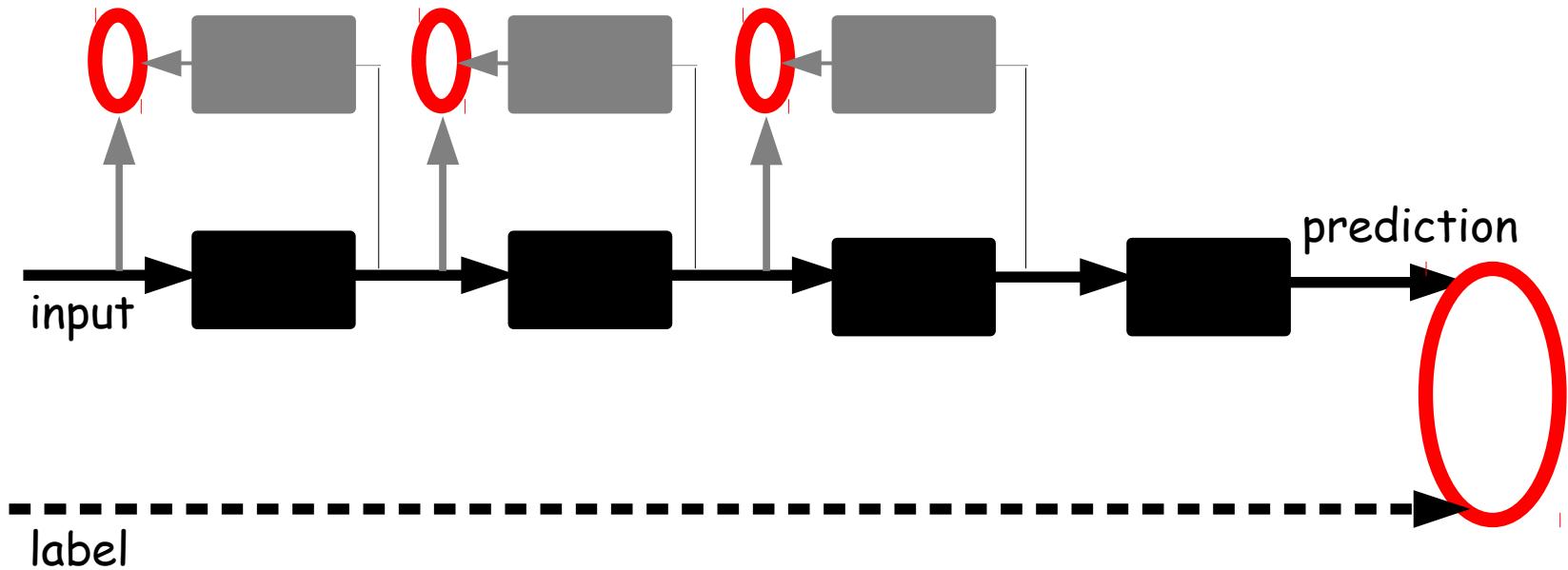
- Hierarchical Convolutional Sparse Coding.
- Trained with respect to image from all layers (L1-L4).
- Pooling both spatially and amongst features.
- Learns invariant mid-level features.



# Recap of Deep Learning Tutorial

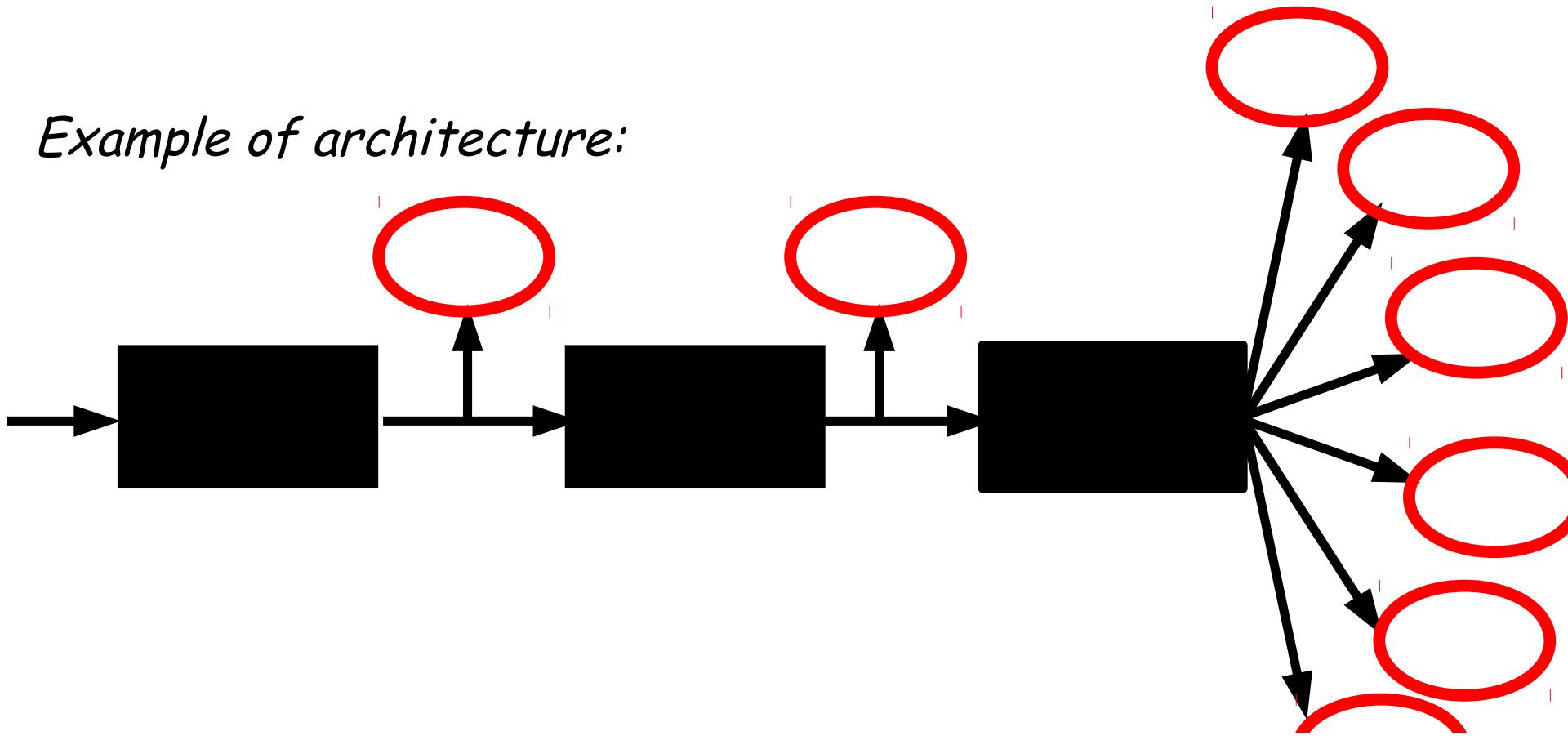
- Building blocks
  - RBMs, Autoencoder Neural Net, Sparse Coding
- Go deeper: Layerwise feature learning
  - Layer-by-layer unsupervised training
  - Layer-by-layer supervised training
- Fine tuning via Backpropogation
  - If data are big enough, direct fine tuning is enough
- Sparsity on hidden layers are often useful.

# Layer-by-layer unsupervised training + fine tuning



# Layer-by-Layer Supervised Training

*Example of architecture:*



## **Biological & Theoretical Justification**

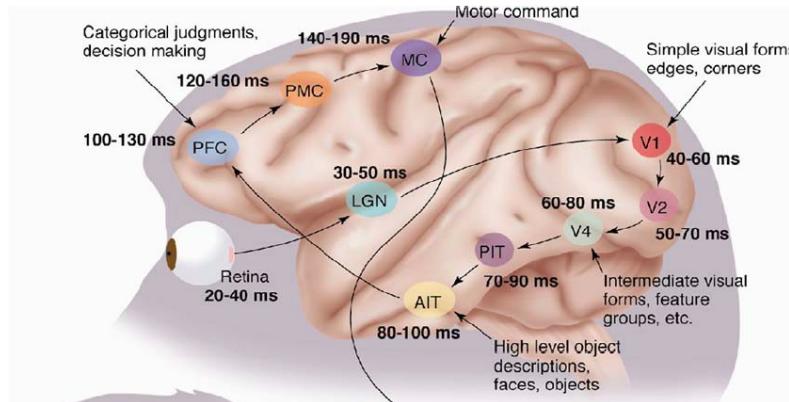
# Why Hierarchy?

---

## Theoretical:

“...well-known depth-breadth tradeoff in circuits design [Hastad 1987]. This suggests many functions can be much more efficiently represented with deeper architectures...” [Bengio & LeCun 2007]

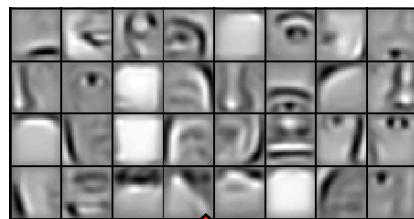
## Biological: Visual cortex is hierarchical (Hubel-Wiesel Model)



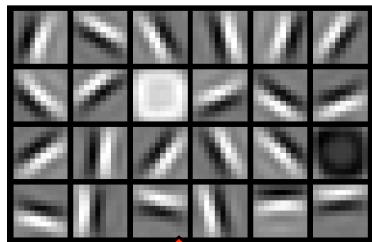
# Sparse DBN: Training on face images



object models



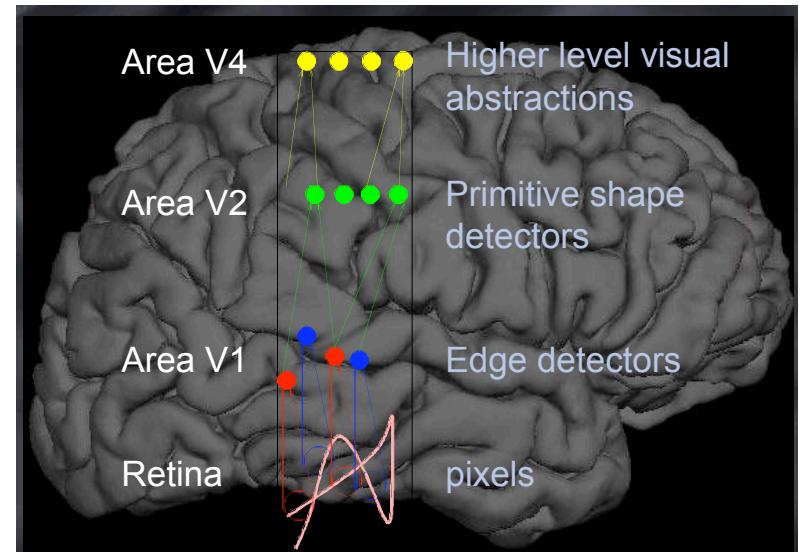
object parts  
(combination  
of edges)



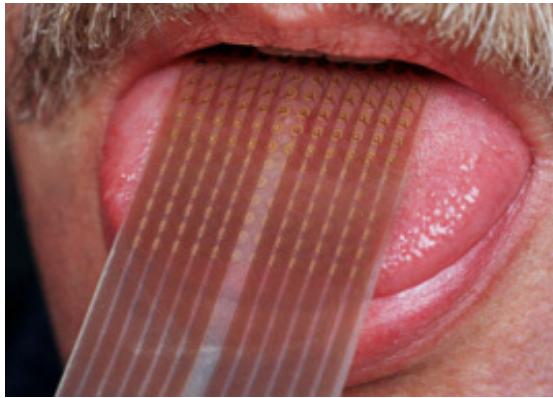
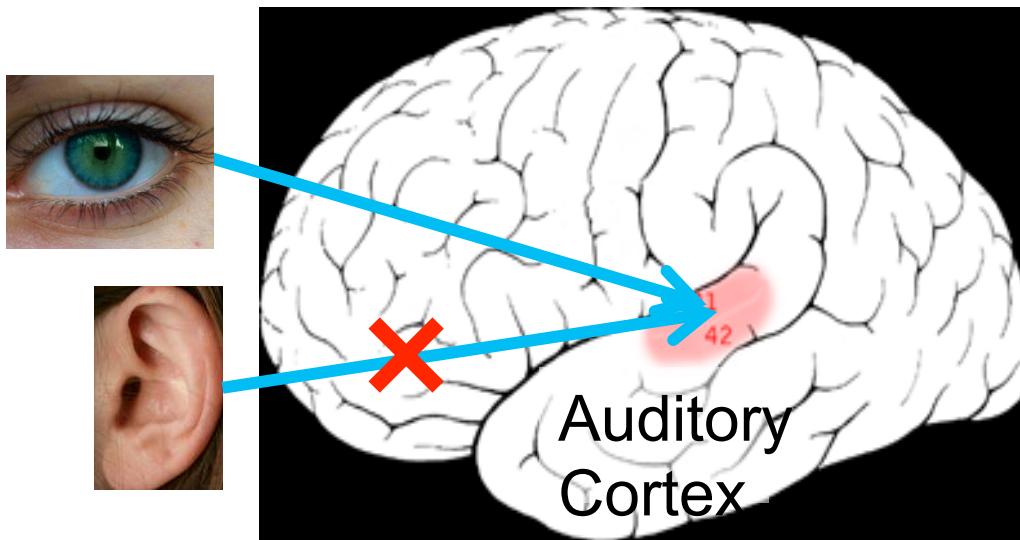
edges



pixels



# Sensor representation in the brain



# **Large-scale training**

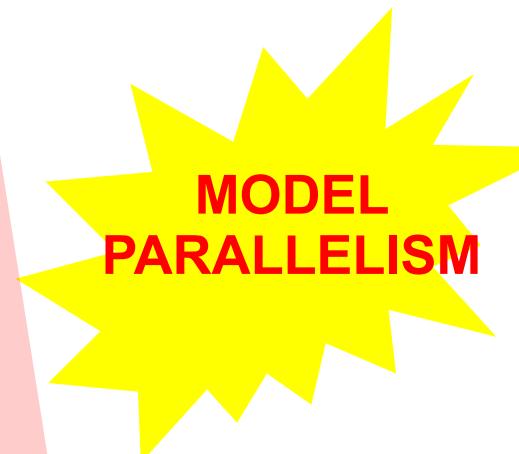
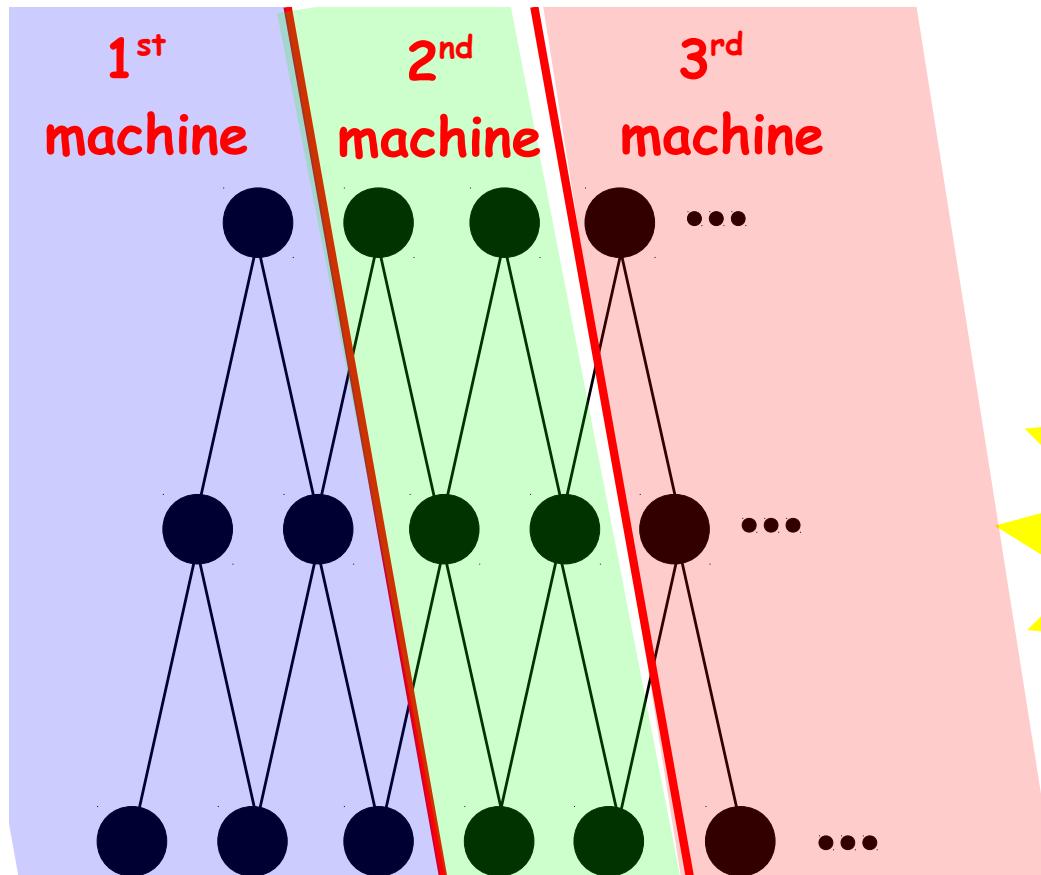
# The challenge

A Large Scale problem has:

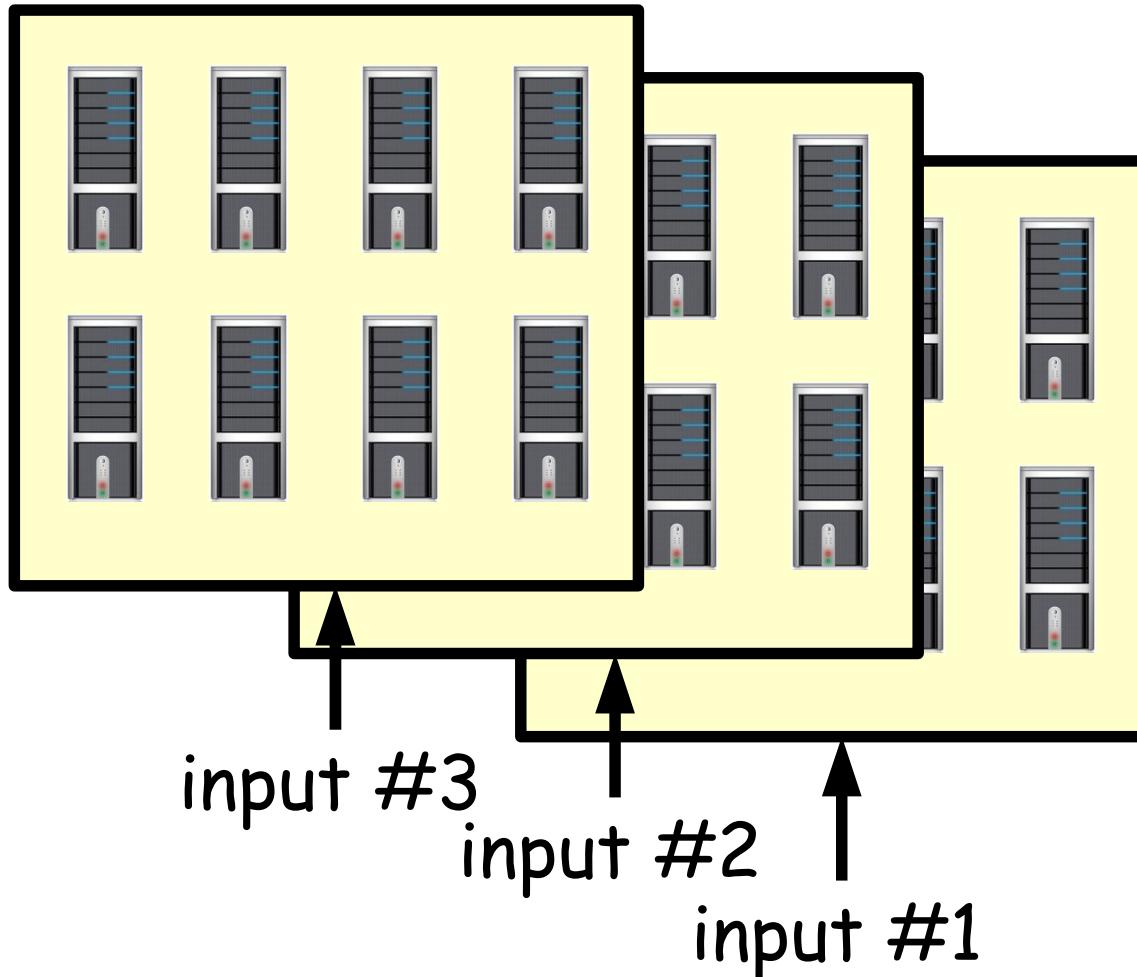
- lots of training samples ( $>10M$ )
- lots of classes ( $>10K$ ) and
- lots of input dimensions ( $>10K$ ).

- best optimizer in practice is on-line SGD which is naturally sequential, hard to parallelize.
- layers cannot be trained independently and in parallel, hard to distribute
- model can have lots of parameters that may clog the network, hard to distribute across machines

# A solution by model parallelism



Le et al. "Building high-level features using large-scale unsupervised learning" ICML 2012



MODEL  
PARALLELISM

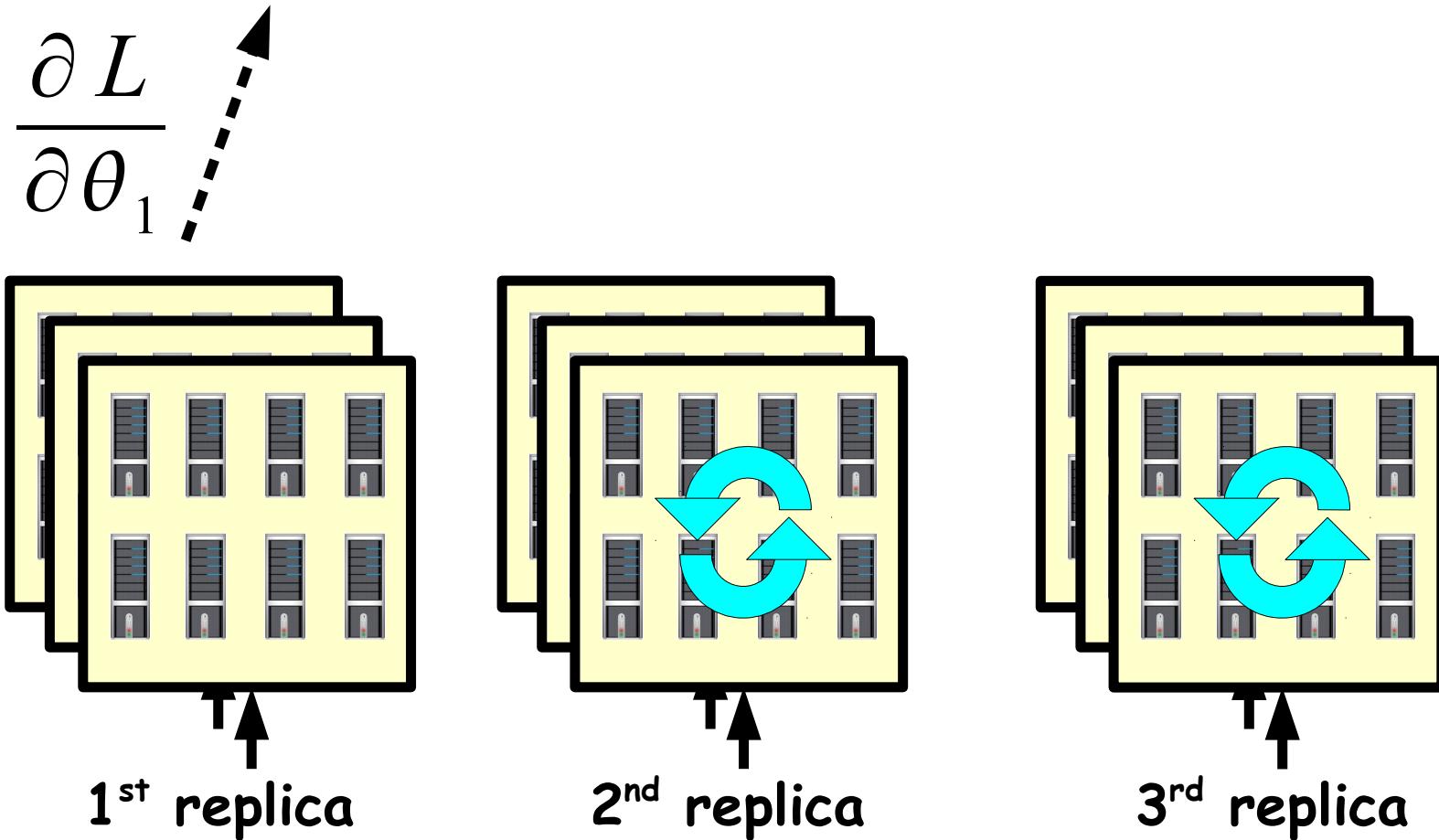
+

DATA  
PARALLELISM

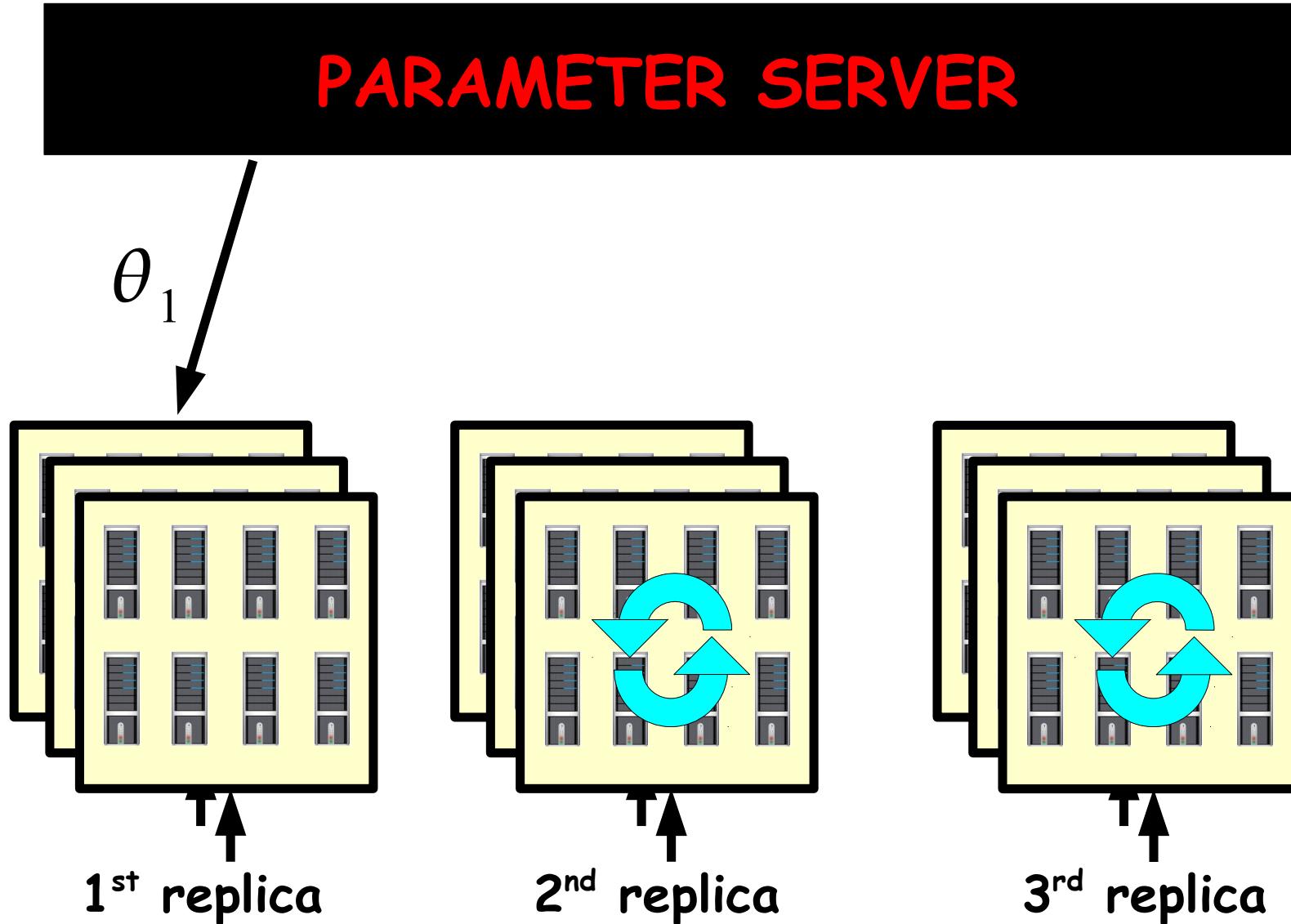
Le et al. "Building high-level features using large-scale unsupervised learning" ICML 2012

# Asynchronous SGD

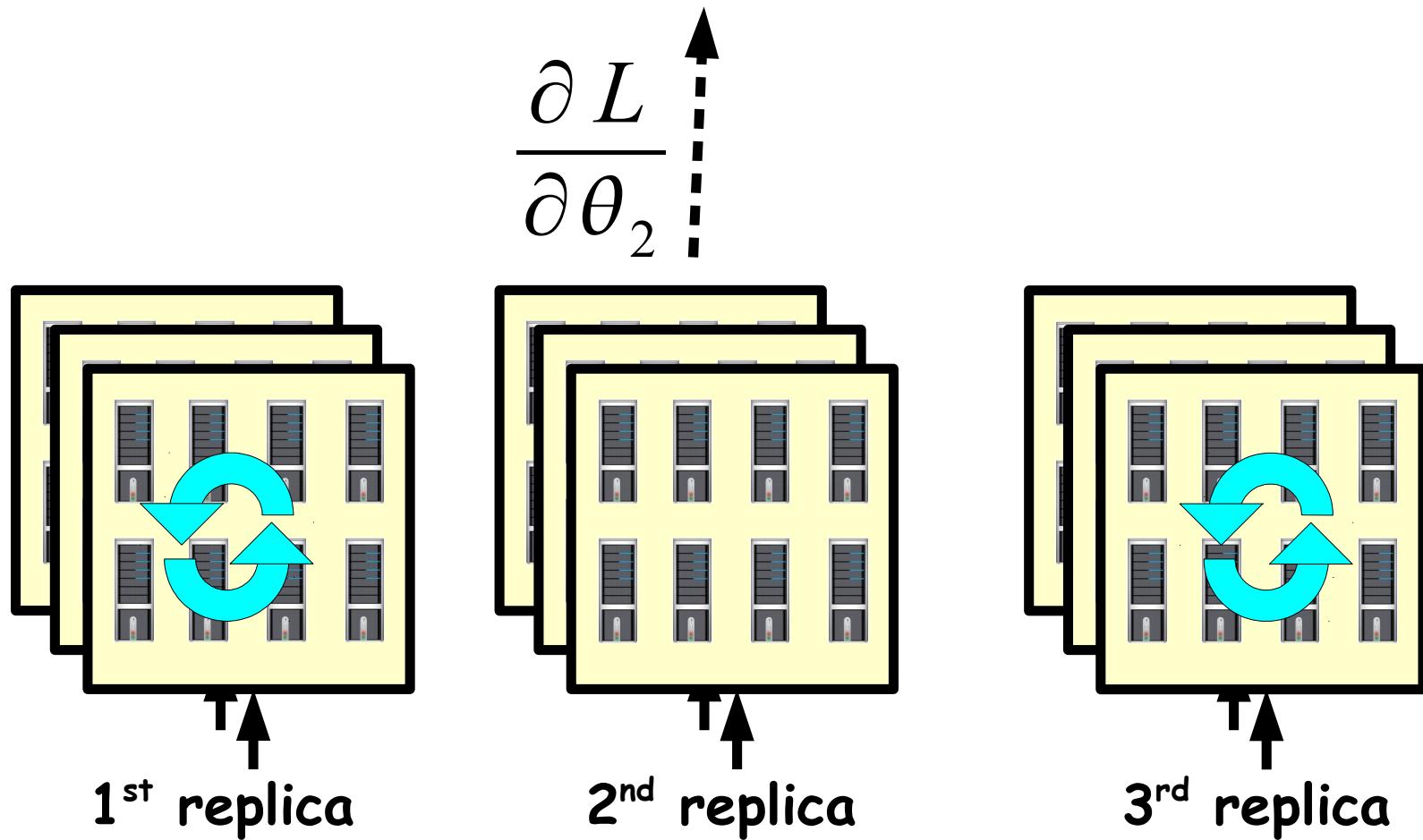
## PARAMETER SERVER



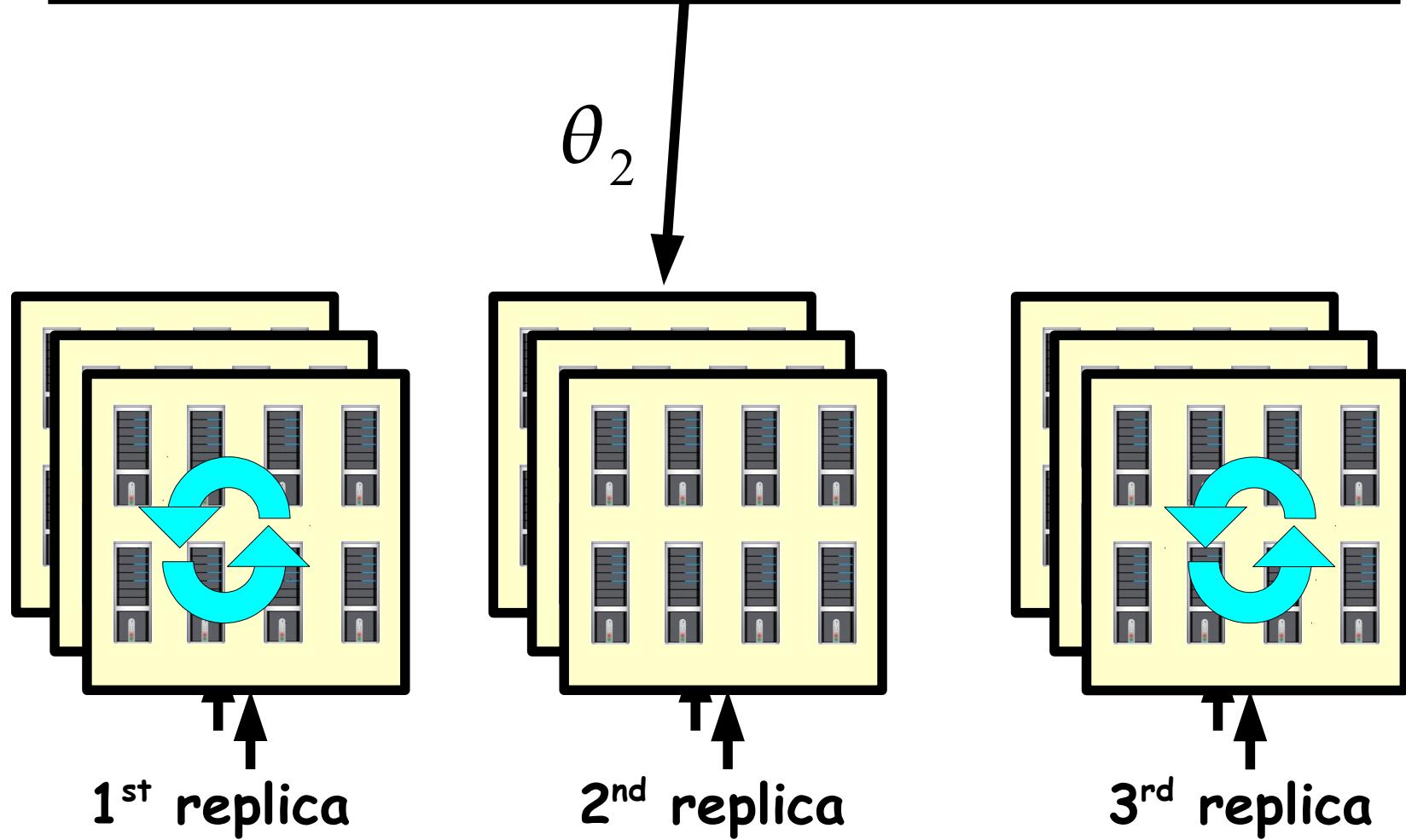
# Asynchronous SGD



# PARAMETER SERVER



# PARAMETER SERVER



# Training A Model with I B Parameters

## Deep Net:

- 3 stages
- each stage consists of local filtering, L2 pooling, LCN
  - $18 \times 18$  filters
  - 8 filters at each location
  - L2 pooling and LCN over  $5 \times 5$  neighborhoods
- training jointly the three layers by:
  - reconstructing the input of each layer
  - sparsity on the code

**Thank you**