Music Machine Learning

XI – Deep learning

Master ATIAM - Informatique

Philippe Esling (esling@ircam.fr)

Maître de conférences – UPMC

Equipe représentations musicales (IRCAM, Paris)

Deep learning motivation

- Supervised training of models is **difficult** (optimization)
- **Shallow** models (SVMs, NNets, boosting, etc…) are **unlikely candidates for learning high-level** abstractions needed for AI
- Unsupervised learning could do "local-learning"
- Each module **tries its best to model what it sees**
- Inference (and learning) is intractable in directed graphical models with many hidden variables
- Current unsupervised learning methods dont easily extend to learn multiple levels of representation

Deep architectures?

- **Deep architectures:** compositions of many layers of adaptive non-linear components.

  Difficulty: parameter searching (local minima)

- **Deep belief nets:** probabilistic generative models that are composed of multiple layers of stochastic, latent variables.

Why using deep architectures?

- Insufficient depth can hurt

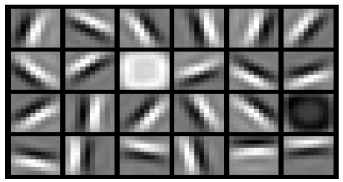
Experiences (past courses) tell us that one-layer machine

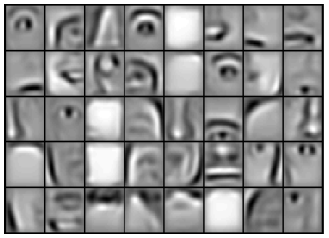
  - only gives us a set of general dictionary elements
  - unless a huge number of dictionary elements.

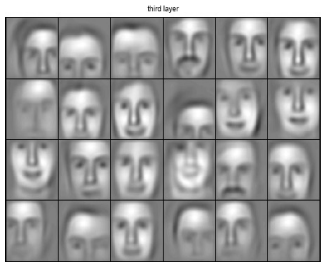
- The brain has a deep architecture
- Cognitive processes seem deep
- Learn a feature hierarchies or
- Functions that represent high-level abstractions

Pixels → Edges → Motifs → Parts → Objects → Scenes

Deep learning intuition







Pixels

Edges

Object parts
(combinations of edges)

Faces
(combinations of parts)

?

Representation learning

- Each level is an **abstraction**
- Abstractions at one level are various **ways to correlate lower-level abstractions**
- We can construct optimal abstractions by **discriminating the important variations**
- Idea is that if we can deconstruct an object and then reconstruct it from fewer parts, we are **learning its structure**
- Knowing the higher levels allow us to re-generate from such networks.

Pixels

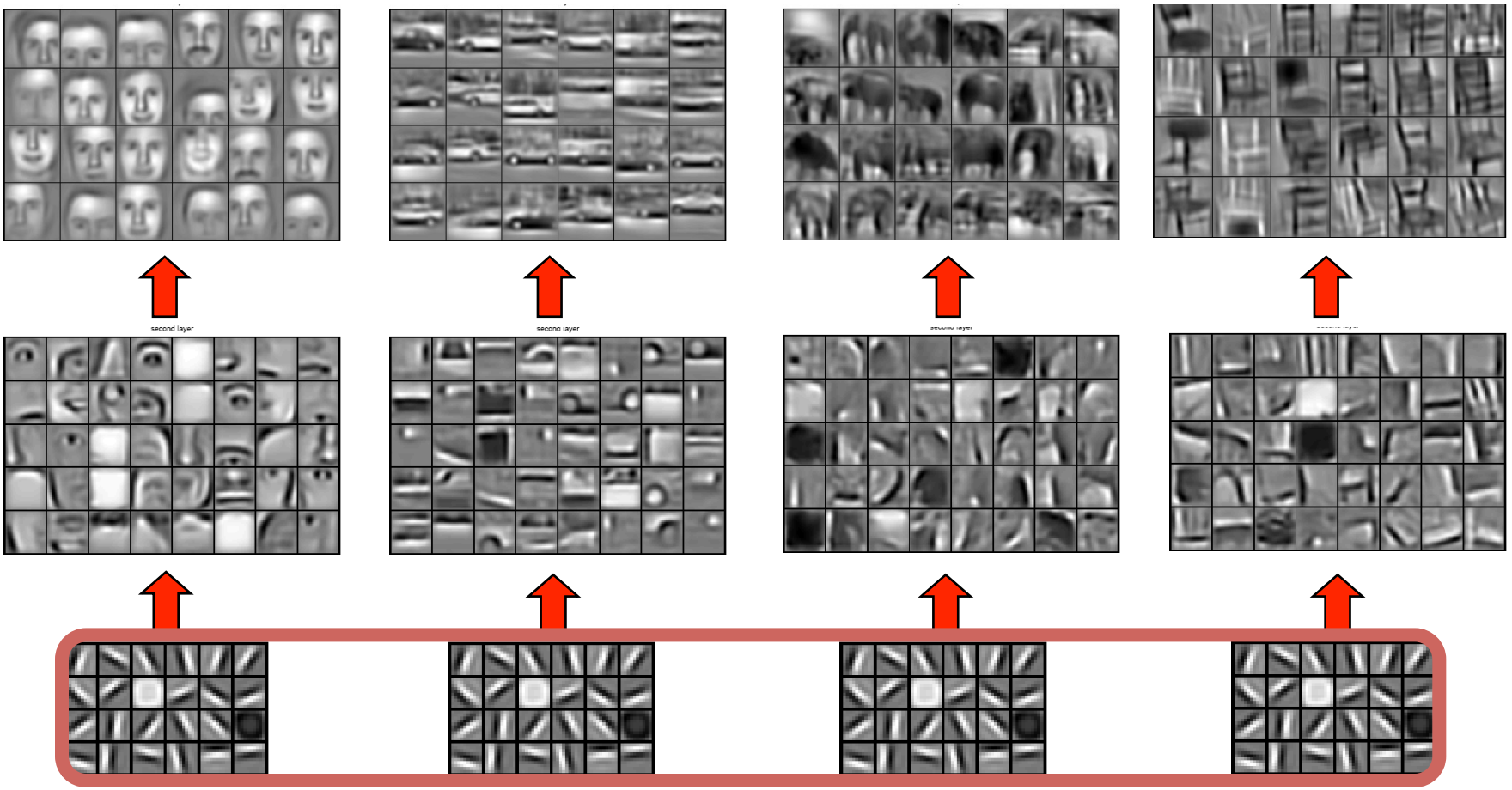
?

Edges

Object parts
(combinations of edges)

Faces
(combinations of parts)

Deep learning intuition



Shared « edges » layer (cf. unsupervised learning)

Why **deep** learning

- The mammal brain organized in a deep architecture
- Given input represented at multiple levels of abstraction,
- Each level corresponding to a different area of cortex
- An architecture with insufficient depth can require many more computational elements …
- Potentially exponentially more (with respect to input size)
- As number of computational elements depends on the number of training examples available to tune or select
- Consequences are not just computational but also statistical:
- Poor generalization may be expected if insufficiently deep

Depth of existing approaches

- Linear/logistic regression = **1 layer**
- Boosting = **2 layers**
  - L 1: base learner
  - L 2: vote or linear combination of layer 1
- Decision tree, LLE, KNN, Kernel SVM = **2 layers**
  - L 1: matching degree to a set of local templates.
  - L 2: Combine these degrees $b + \sum_i \alpha_i K(\mathbf{x}, \mathbf{x}_i)$
- Neural Network = **3 layers**
  - Could be of any depth but **gradient diffusion problem**
- Brain = **5-10 layers**

When a function can be compactly represented by a deep architecture, it might need a very large architecture to be represented by an insufficiently deep one.

Depth 2 for SVM/KNN/Trees ?

- Rely on partition of input space.
- Local estimator use separation for each region.
- Each region is associated with a leaf.
- Need as many as training samples as variations of interest in the target function.
- Not good for highly varying functions.
- Number of training sample is exponential

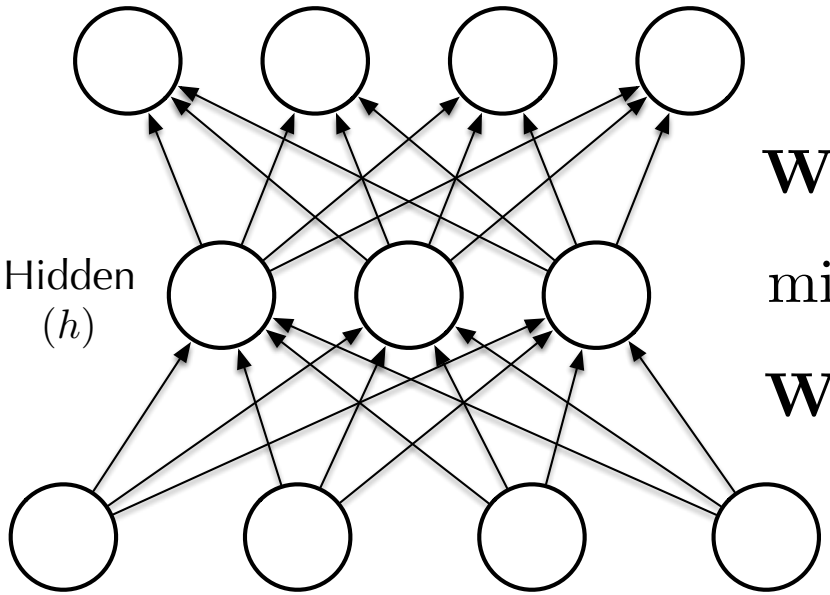
(to number of dimensions in order to achieve a fixed error rate)

Divide and conquer multilayer learn

- **Re-representing the data:** Each time the base learner is called, it passes a transformed version of the data to the next learner.
  - Can we learn a deep, dense DAG one layer at a time, starting at the bottom, and still **guarantee that learning each layer improves the overall model of the training data**?
    - This seems very unlikely. Surely we need to know the weights in higher layers to learn lower layers?

Auto-encoders

$$d : \mathbb{R}^{d_h} \rightarrow \mathbb{R}^{d_x} \quad \mathbf{y} = d\left(\mathbf{h_x}\right) = s_d\left(\mathbf{W}_d\mathbf{h_x} + \mathbf{b}_d\right)$$



$\mathbf{W}_d$

$\min\left[\mathcal{L}(x, y)\right]$

$\mathbf{W}_e$

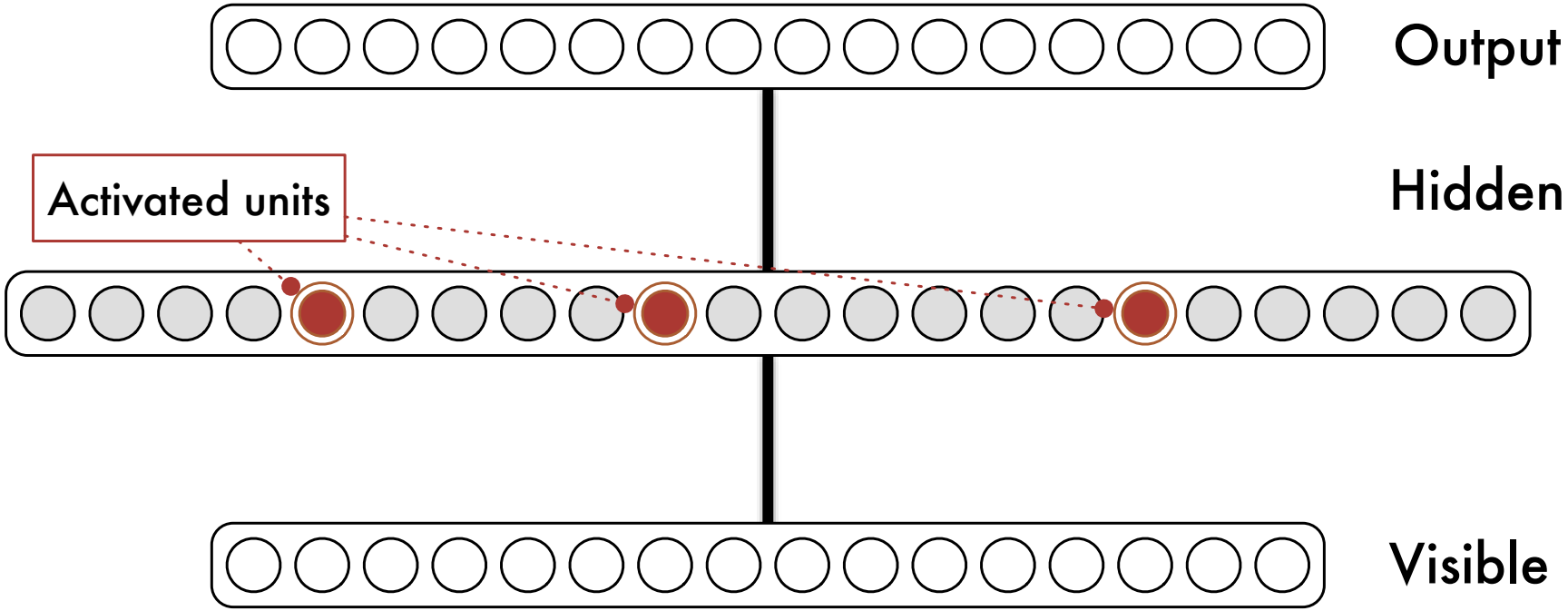
$$\mathbf{W}_d = \mathbf{W}_e^T$$

$$\mathcal{L}(x, y) = \|x - y\|^2$$

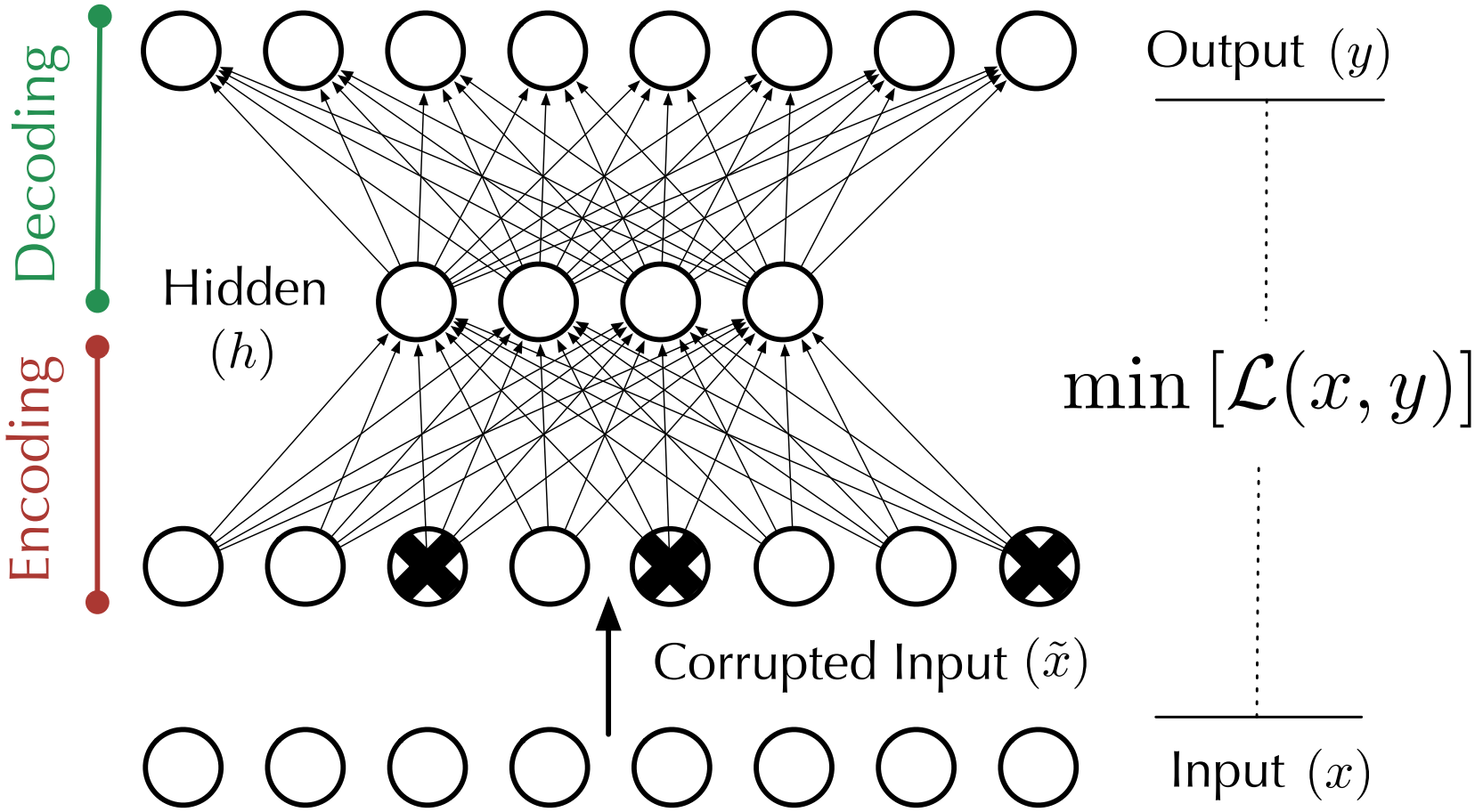
$$e : \mathbb{R}^{d_x} \rightarrow \mathbb{R}^{d_h} \quad \mathbf{h_x} = e\left(\mathbf{x}\right) = s_e\left(\mathbf{W_e}\mathbf{x} + \mathbf{b}_e\right)$$

$$\mathcal{J}_{AE}\left(\theta\right) = \sum_{\mathbf{x} \in \mathcal{D}_n} \mathcal{L}\left(\mathbf{x}, d\left(e\left(\mathbf{x}\right)\right)\right) + \beta \sum_i \mathrm{KL}\left(\rho, \hat{\rho}_i\right)$$

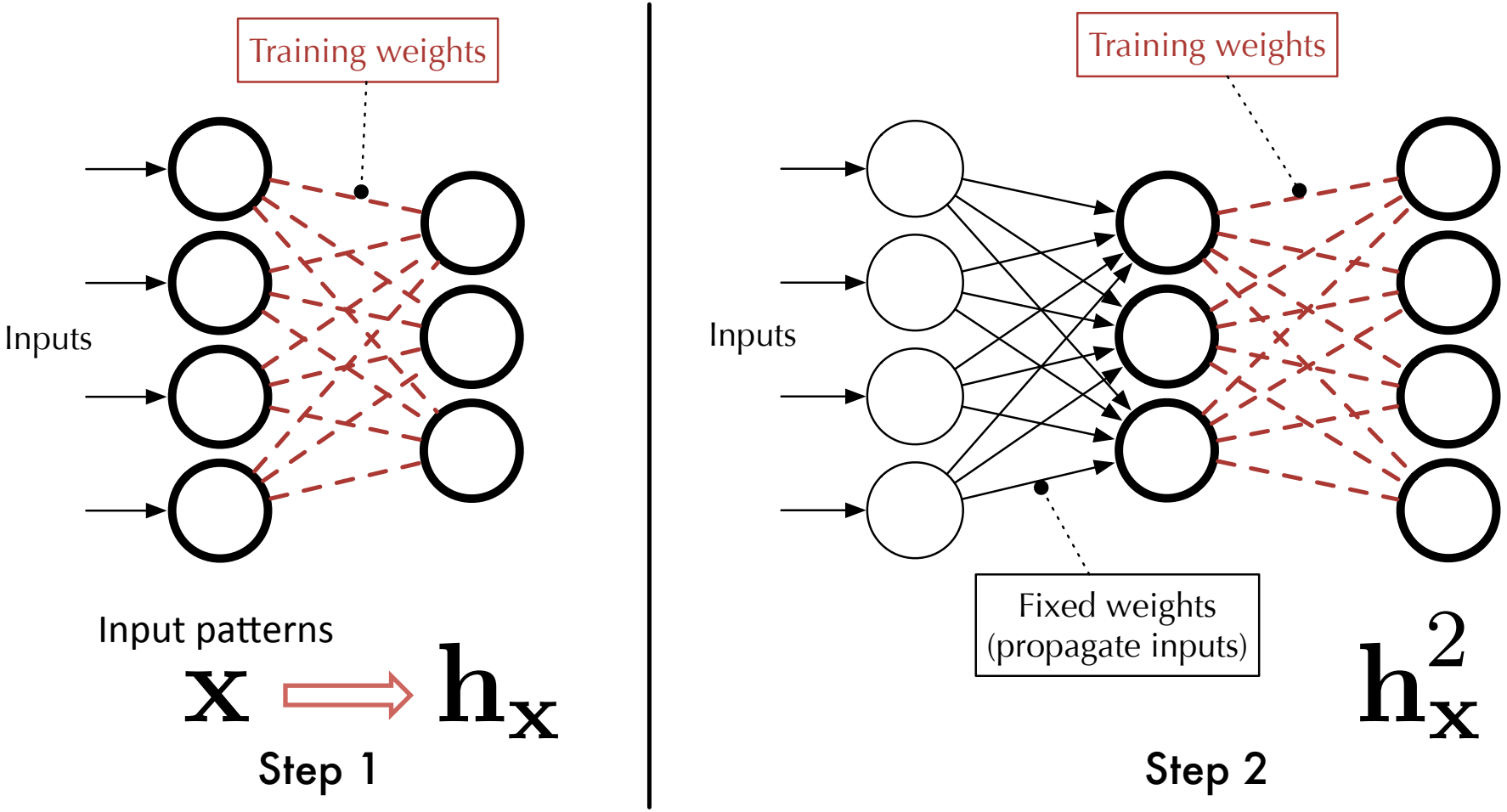
Sparse auto-encoders



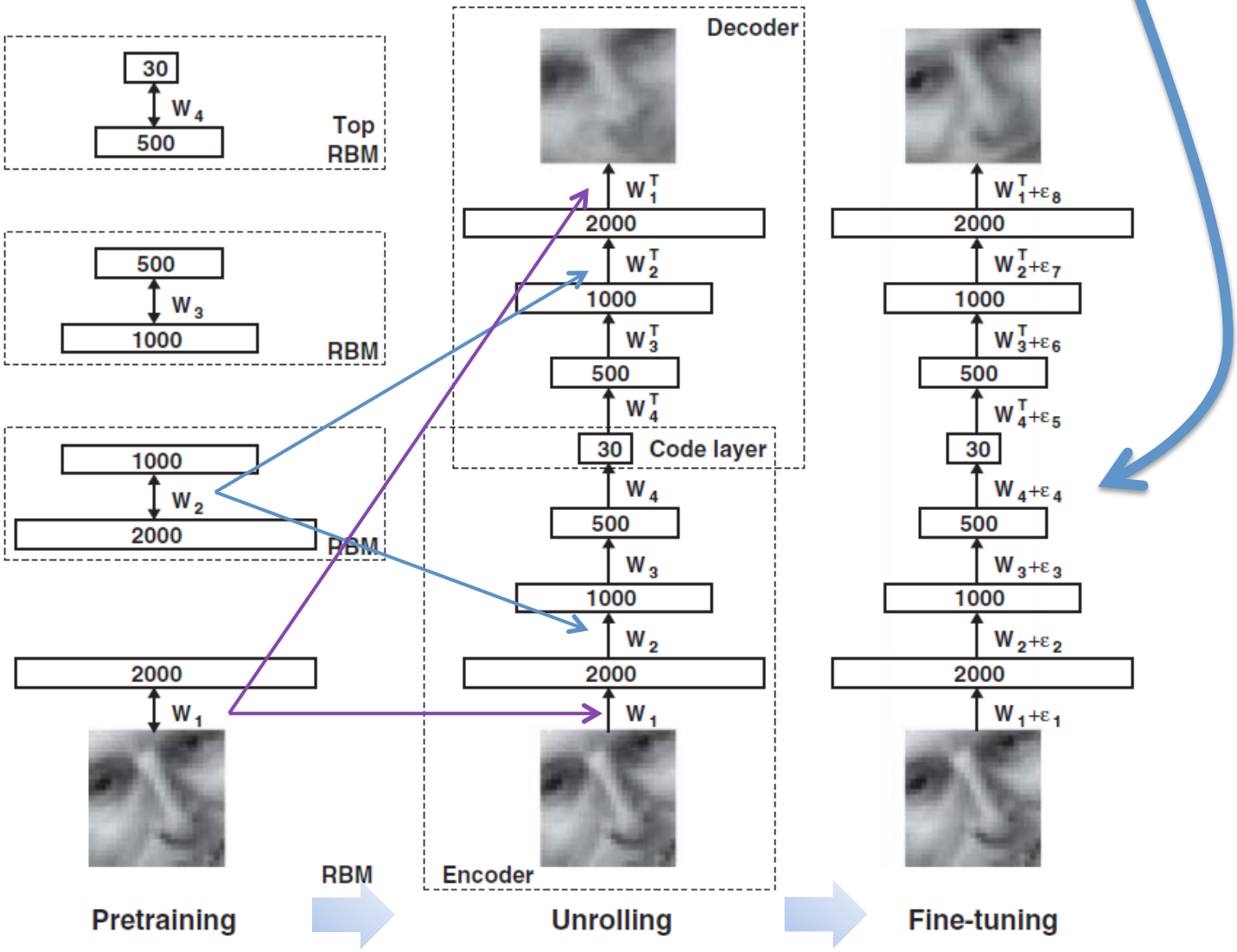
Denoising auto-encoders



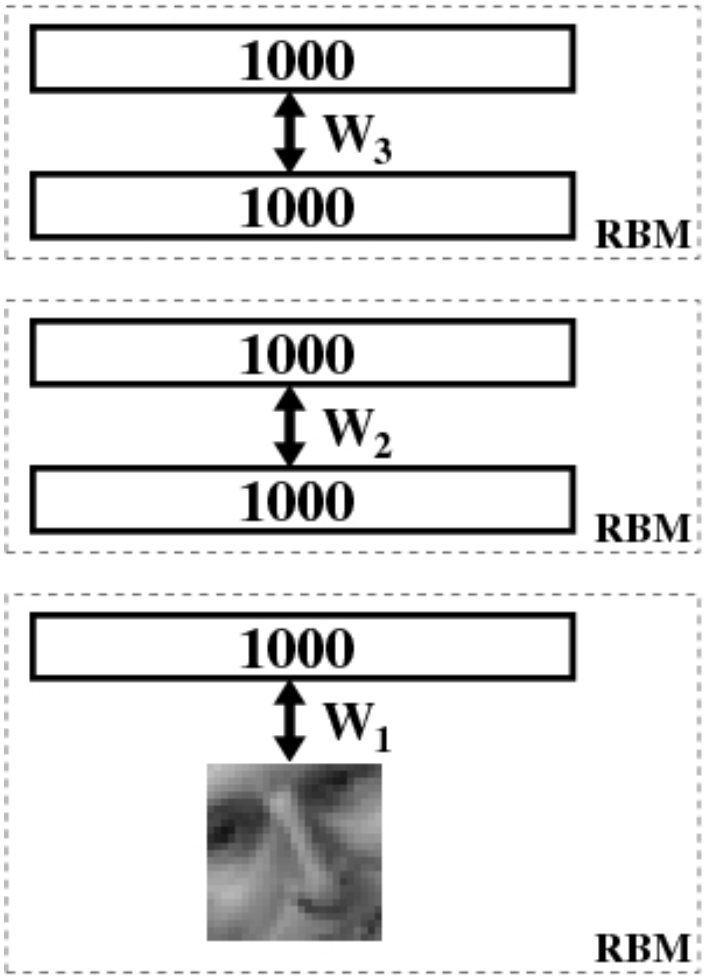
Stacked auto-encoders



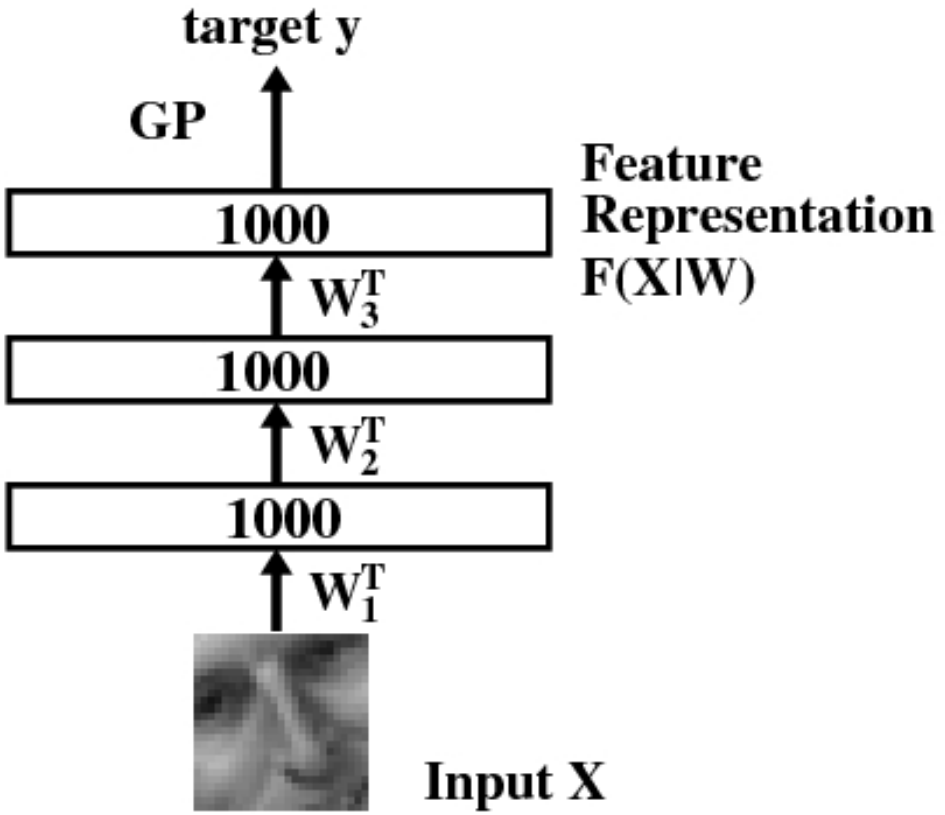
Pretrain, fine-tune (**autoencoder**)



Pre-train fine-tune and infer



y: identity or rotation degree



Supervised learning

Cars

Motorcycles

Semi-supervised learning

Unlabeled images (all cars/motorcycles)

Car

Motorcycle

Self-taught learning

If your labeled dataset is small, can give a huge **performance boost**

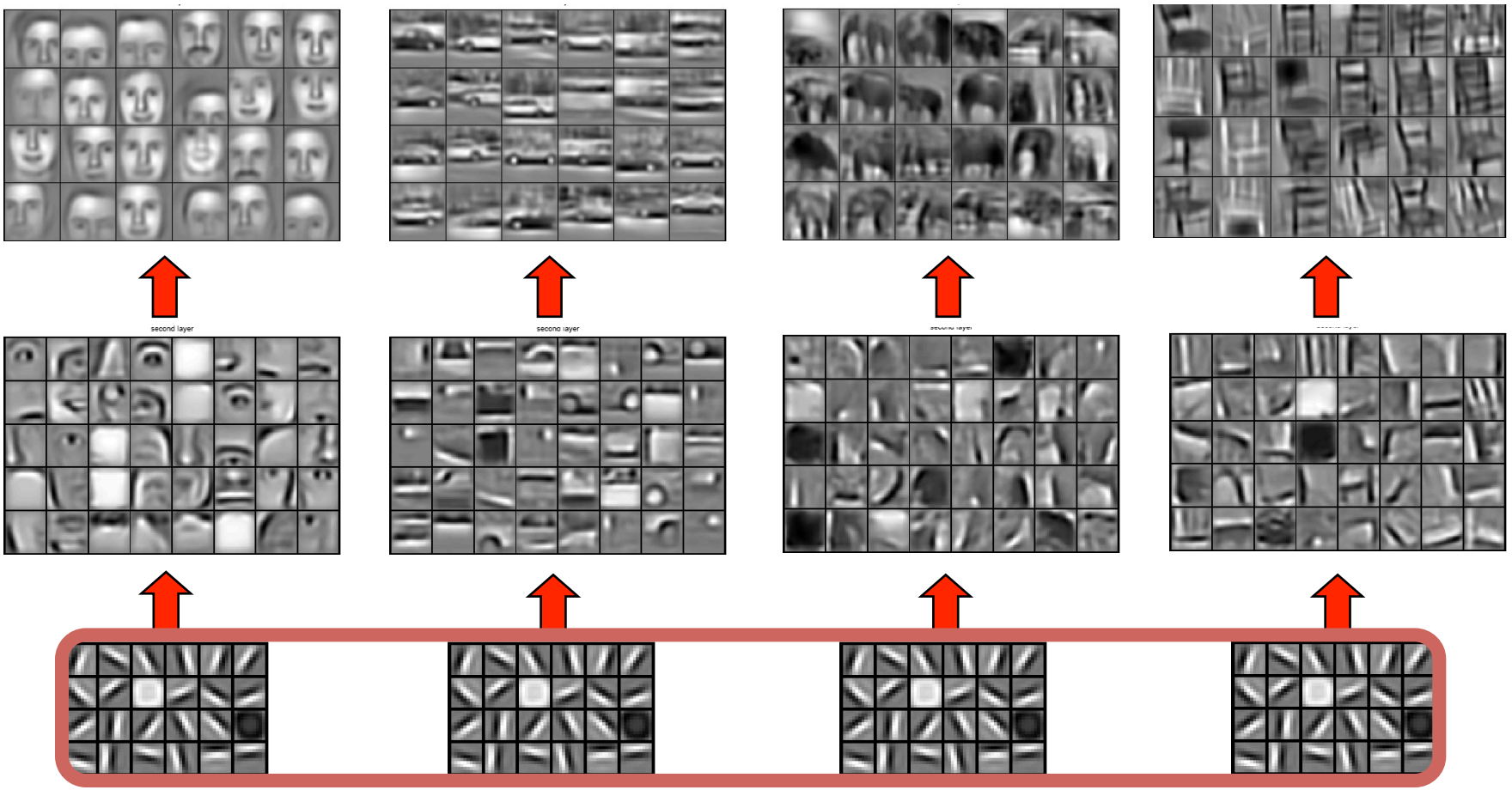


Unlabeled images (random internet images)

Car

Motorcycle

Deep learning intuition



Shared « edges » layer (cf. unsupervised learning)

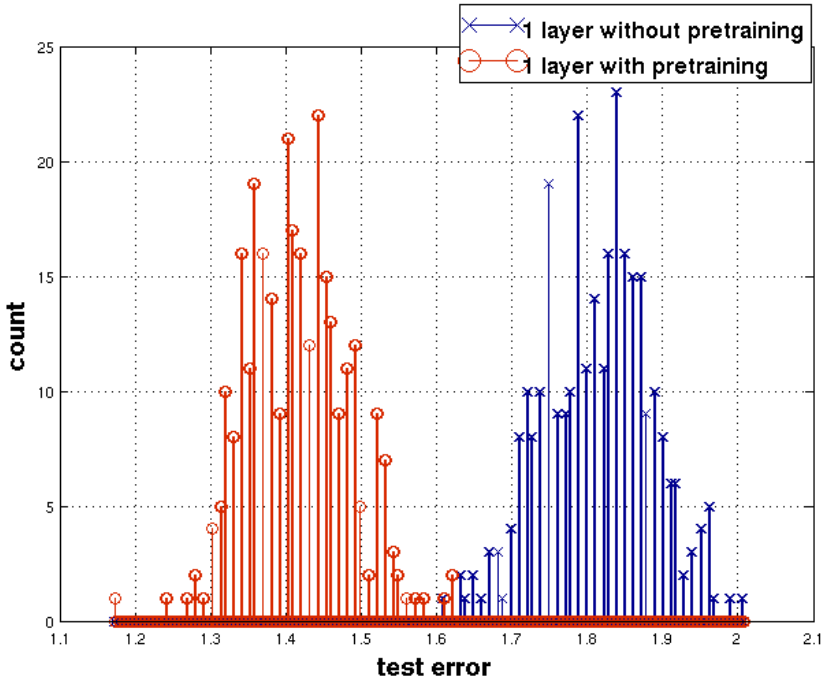
How many layers?

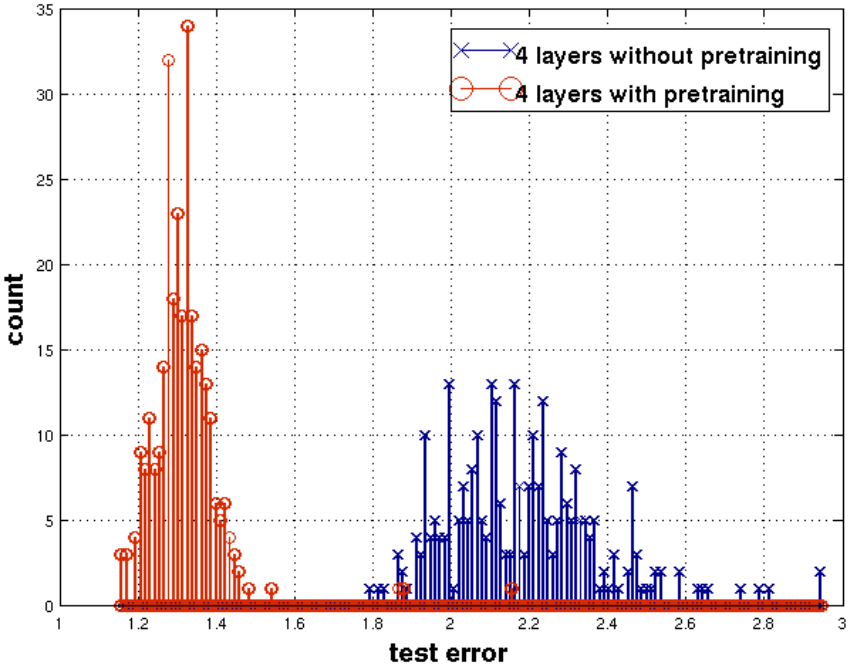
- There might be no universally right depth
  - Several layers is better than one
  - Results are robust against changes in the size of a layer, but top layer should be big
  - A parameter. Depends on your task.
  - With enough narrow layers, we can model any distribution over binary vectors [1]

[1] Sutskever, I. and Hinton, G. E., Deep Narrow Sigmoid Belief Networks are Universal Approximators. Neural Computation, 2007

Effect of unsupervised training

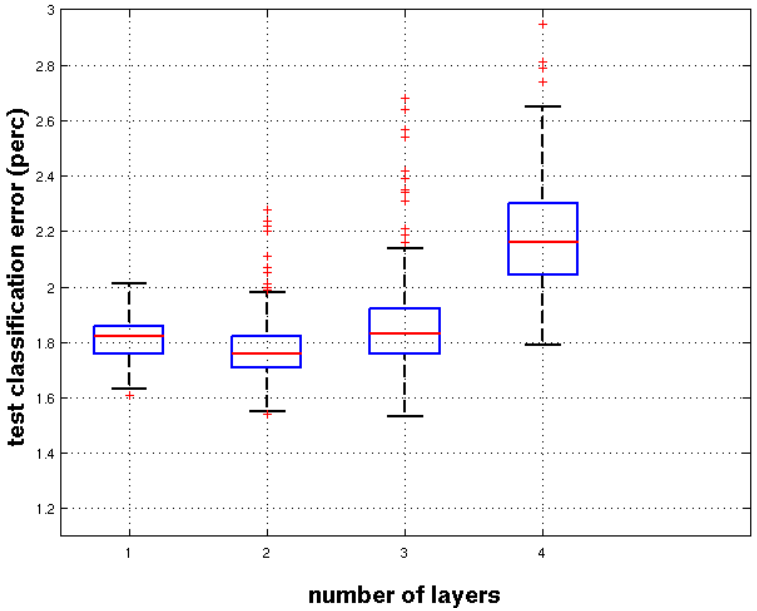
Erhan et. al. AISTATS'2009



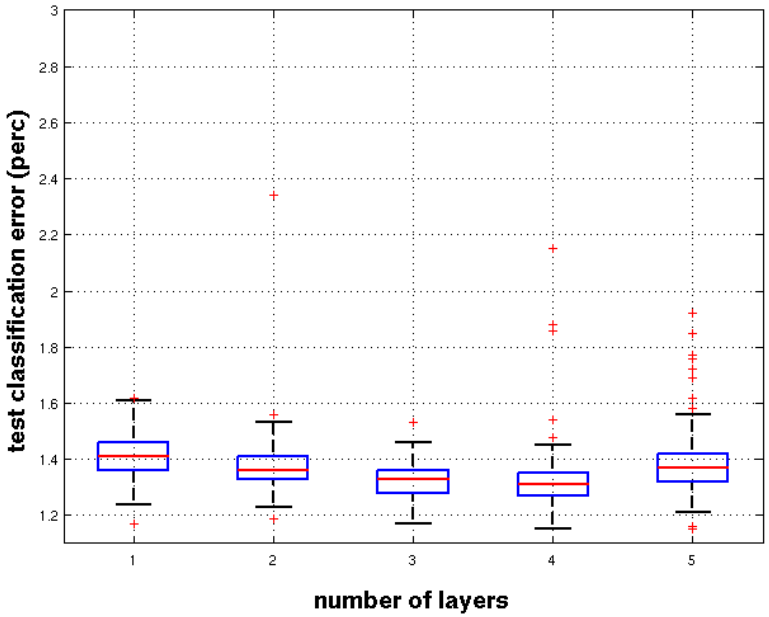


Effect of depth

without pre-training



with pre-training



Beyond layer-wise pretraining

- Layer-wise pretraining is efficient but not optimal.
- It is possible to train parameters for all layers using a contrastive version of the wake-sleep algorithm.
  - Bottom-up in a layer-wise manner
  - Top-down and reffiting the earlier models

Contrastive wake-sleep

After learning many layers of features, we can fine-tune the features to improve generation.

1. Do a stochastic bottom-up pass
   - Adjust the top-down weights to be good at reconstructing the feature activities in the layer below.
2. Do a few iterations of sampling in the top level RBM
   -- Adjust the weights in the top-level RBM.
3. Do a stochastic top-down pass
   - Adjust the bottom-up weights to be good at reconstructing the feature activities in the layer above.

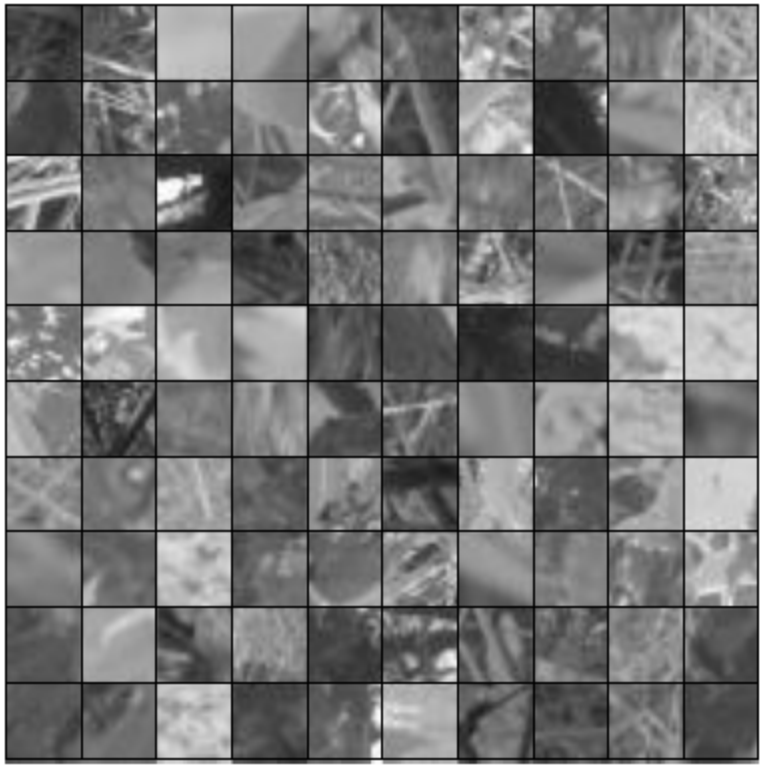
Include lateral connections

- RBM has no connection among layers
- This can be generalized.
- Lateral connections for the first layer [1].
  - Sampling from $P(\mathbf{h}|\mathbf{x})$ is still easy. But sampling from $p(\mathbf{x}|\mathbf{h})$ is more difficult.
- Lateral connections at multiple layers [2].
  - Generate more realistic images.
  - CD is still applicable, with small modification.

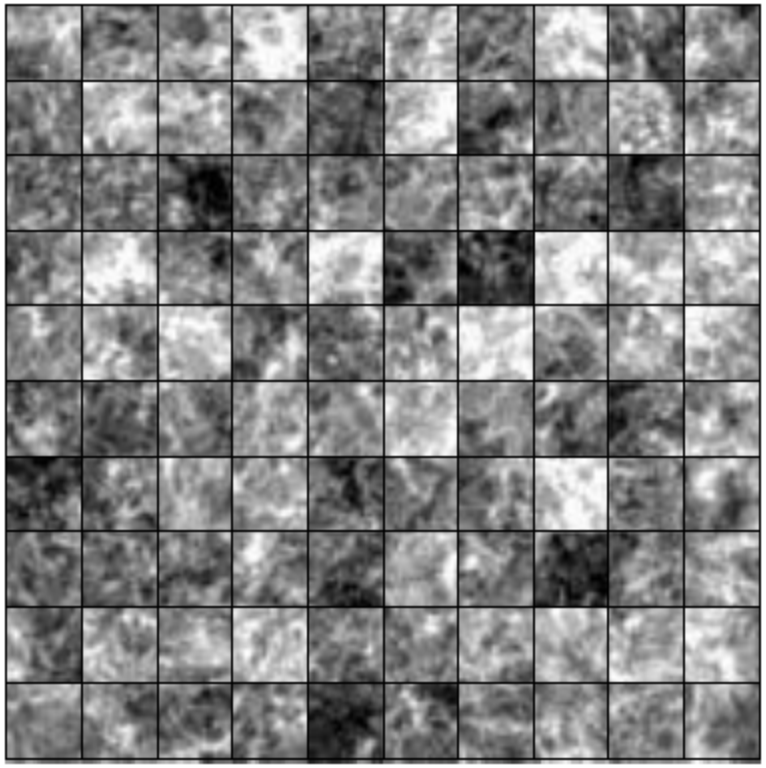
[1] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: a strategy employed by V1?," Vision Research, vol. 37, pp. 3311–3325, December 1997.
[2] S. Osindero and G. E. Hinton, "Modeling image patches with a directed hierarchy of Markov random field," in NIPS, 2007.

Without lateral connections

real data

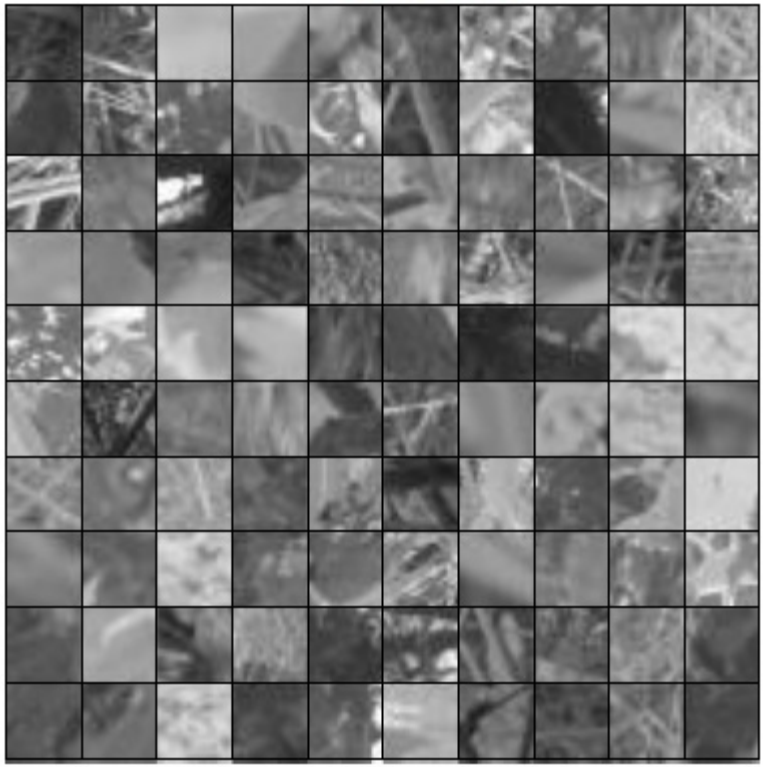


samples from model

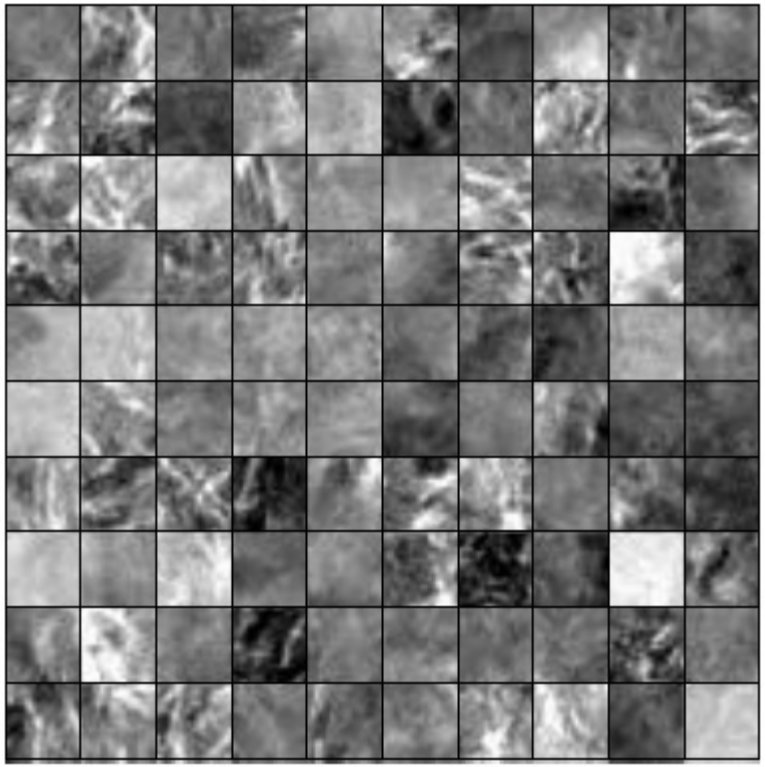


With lateral connections

real data

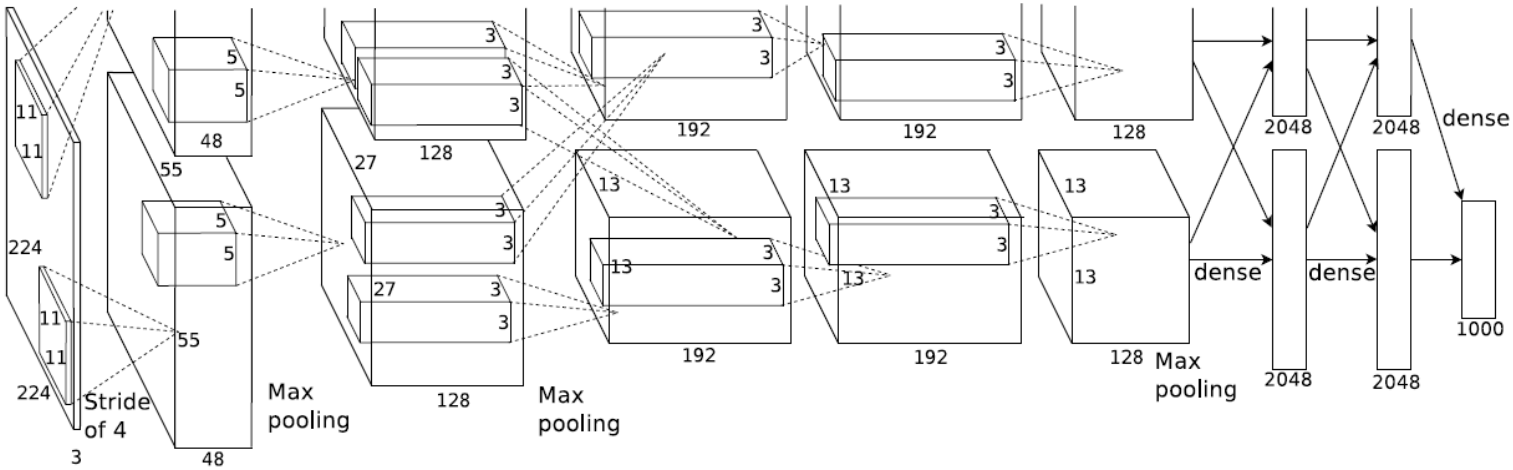


samples from model



Object recognition

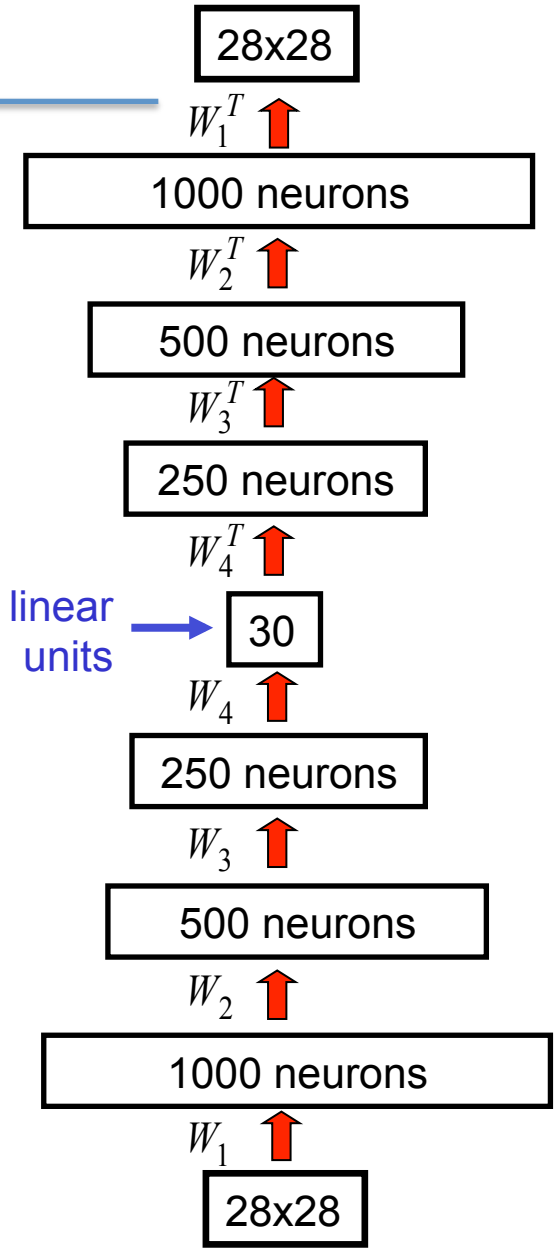
ImageNet



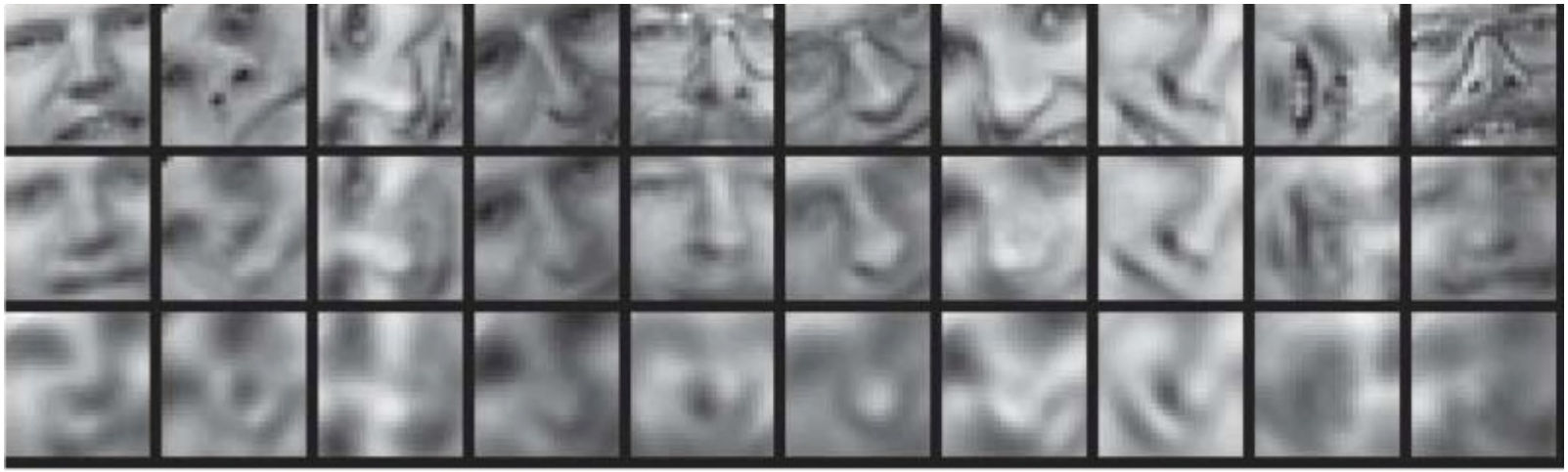
Rank	Name	Error rate	Description
1	**U. Toronto**	0.15315	Deep learning
2	U. Tokyo	0.26172	Hand-crafted features and learning models. Bottleneck.
3	U. Oxford	0.26979	
4	Xerox/INRIA	0.27058	

Deep autoencoder

- They always looked like a really nice way to do non-linear dimensionality reduction:
  - But it is very difficult to optimize deep autoencoders using backpropagation.
- We now have a much better way to optimize them:
  - First train a stack of 4 RBM's
  - Then "unroll" them.
  - Then fine-tune with backprop.



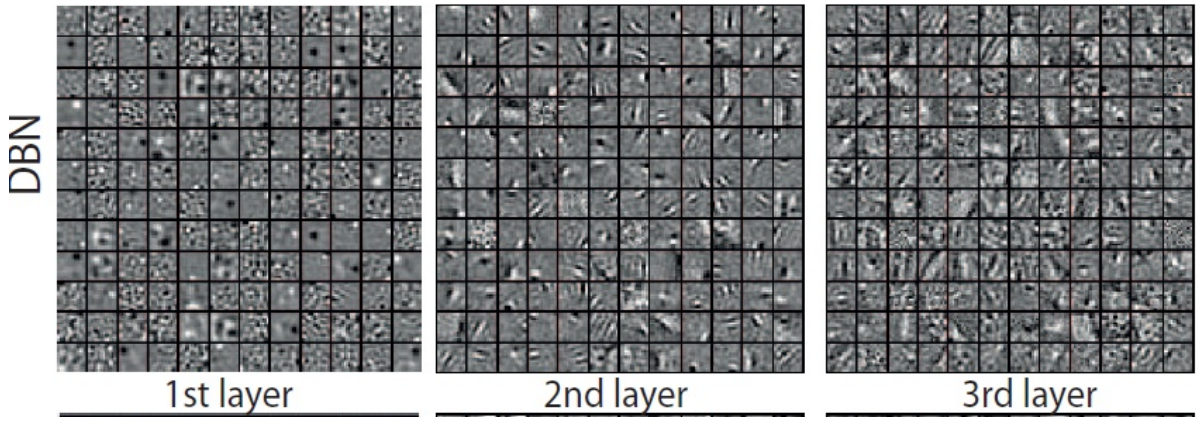
Deep autoencoder



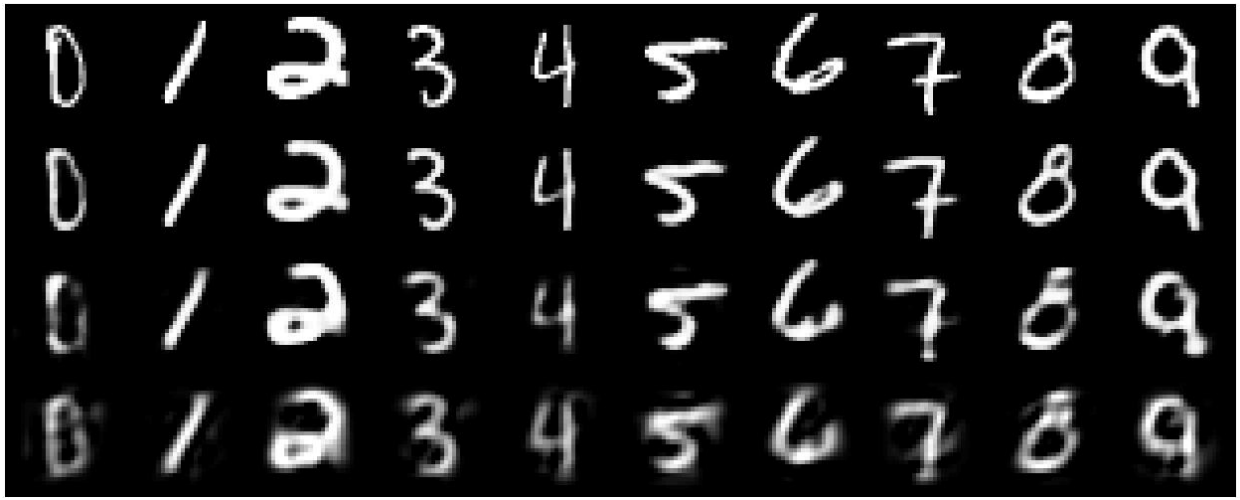
real data

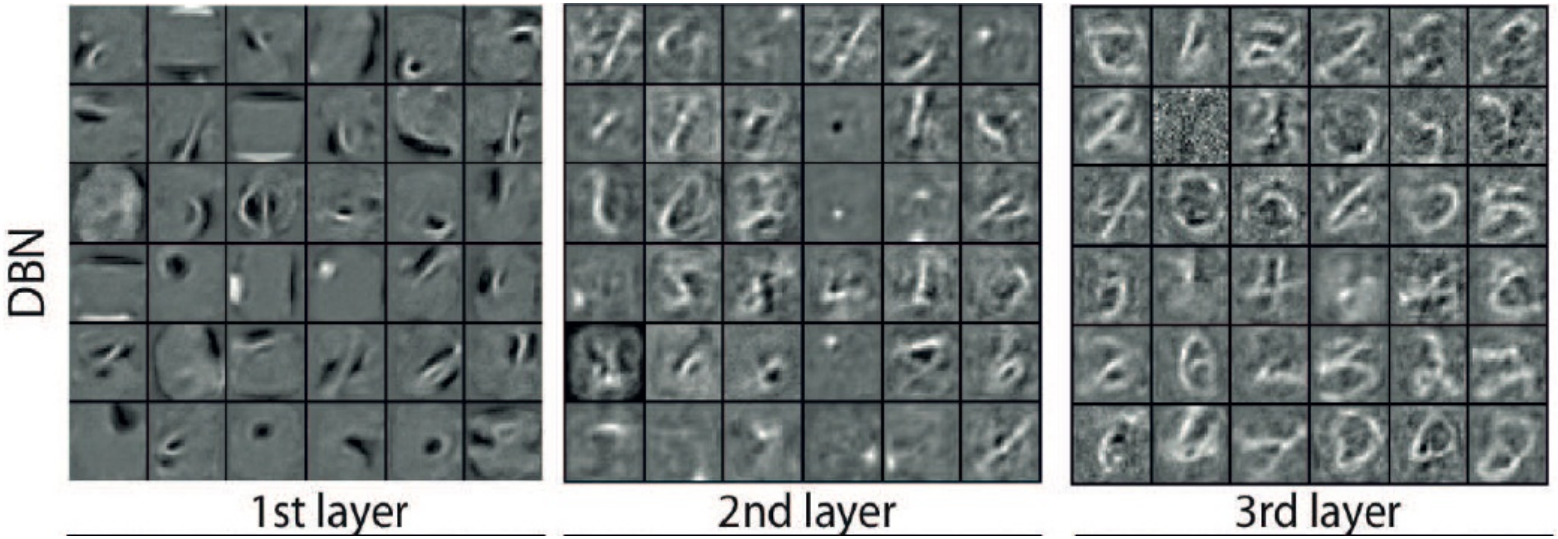
30-D deep

30-D PCA



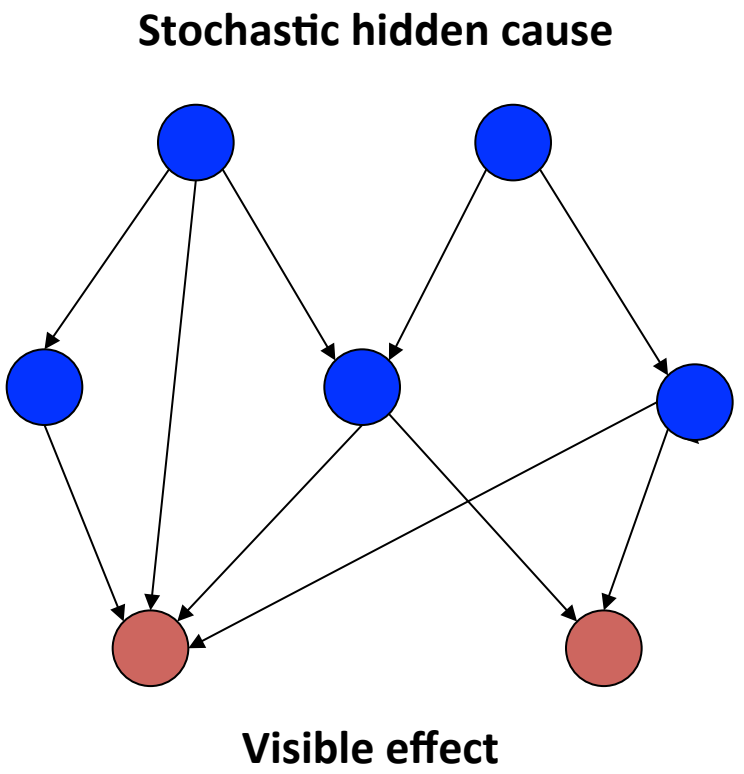
Deep Belief Network





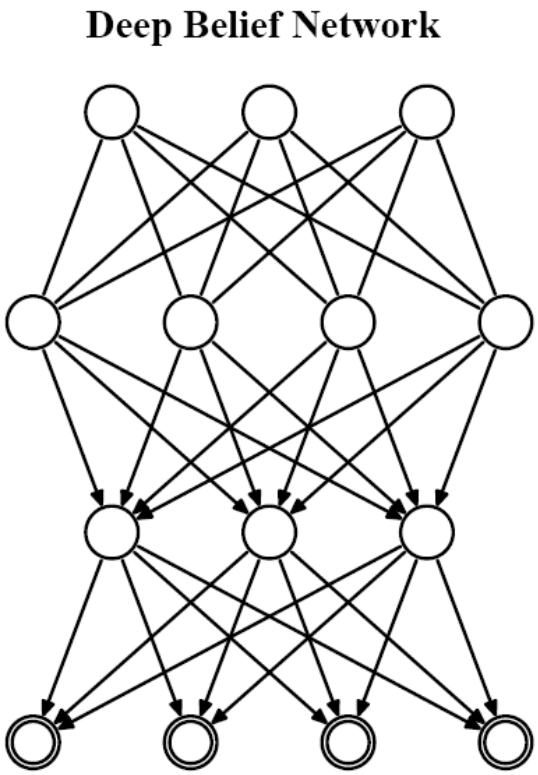
Belief network

- A belief net is a **directed acyclic graph** composed of **stochastic variables.**
- Can observe some of the variables and we would like to solve two problems:
- The **inference** problem: Infer the states of the unobserved variables.
- The **learning** problem: Adjust the interactions between variables to make the network more likely to generate the observed data.

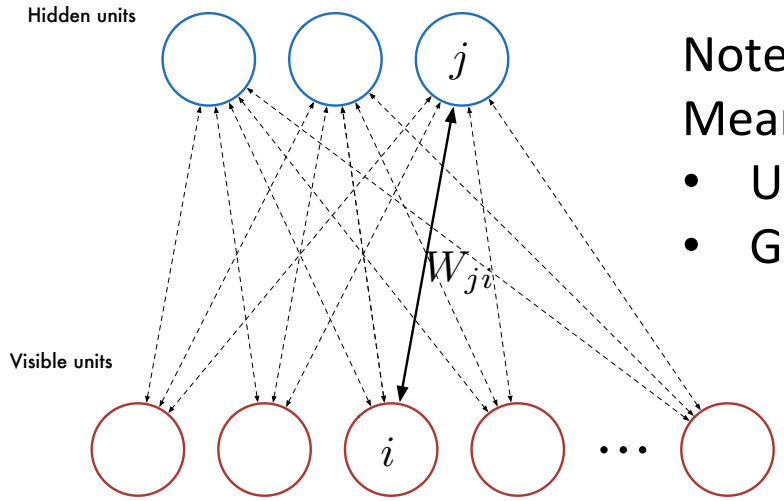


Use **nets composed of layers of stochastic binary variables** with weighted connections. Later, we will generalize to other types of variable.

Learning a DBN?



Restricted Boltzmann Machine



Note the **symmetric weights.**
Means that we can simultaneously

- Understand what the network has learned
- Generate new music from it !

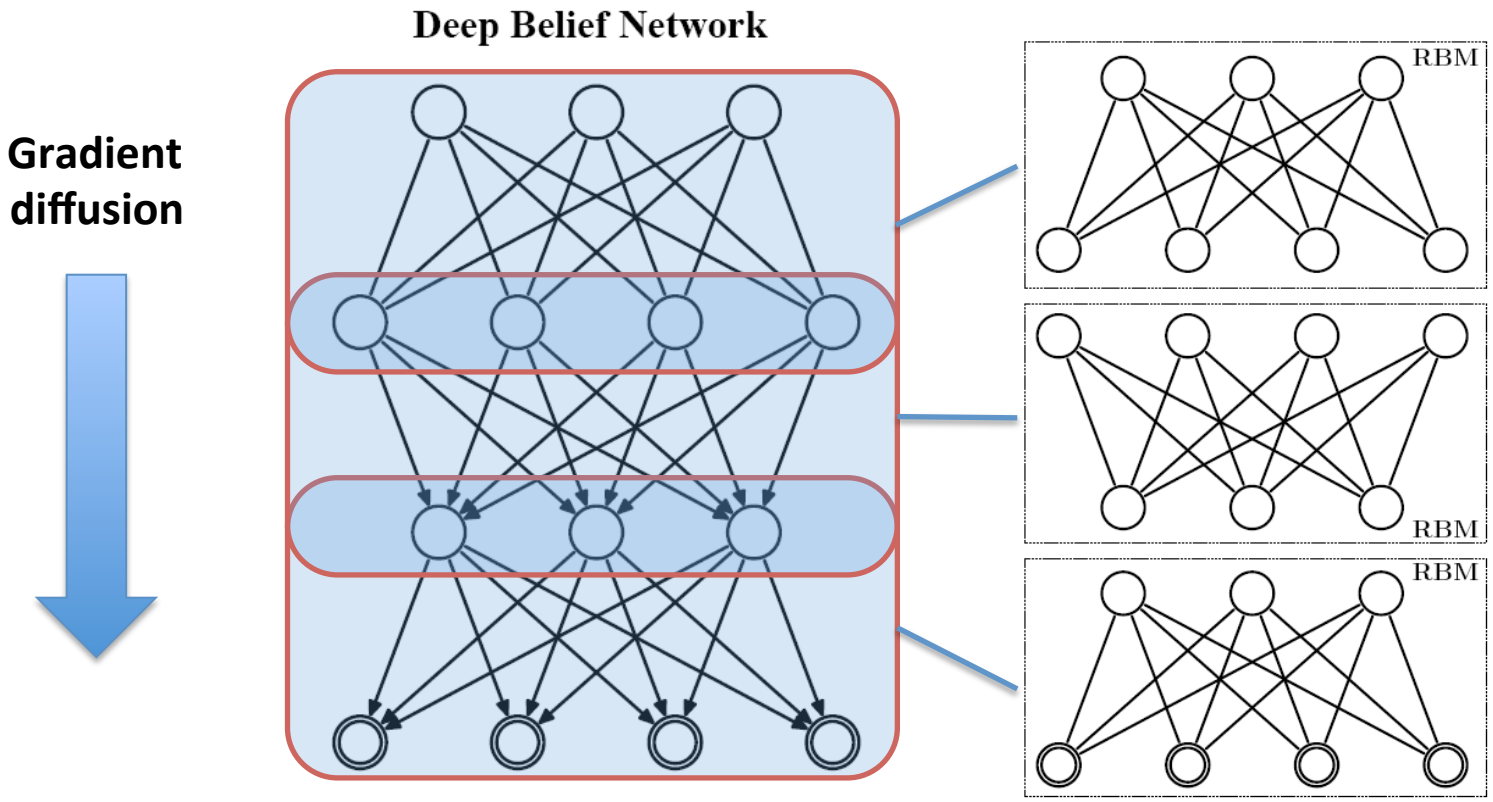
The *energy* of a configuration (pair of boolean vectors) (v,h) is given as

$$E(v,h) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_i \sum_j v_i w_{i,j} h_j$$

The (marginal) probability of a visible (input) vector is the **sum over all possible hidden layer configurations**

$$P(v) = \frac{1}{Z} \sum_h e^{-E(v,h)}$$

Learning a DBN?

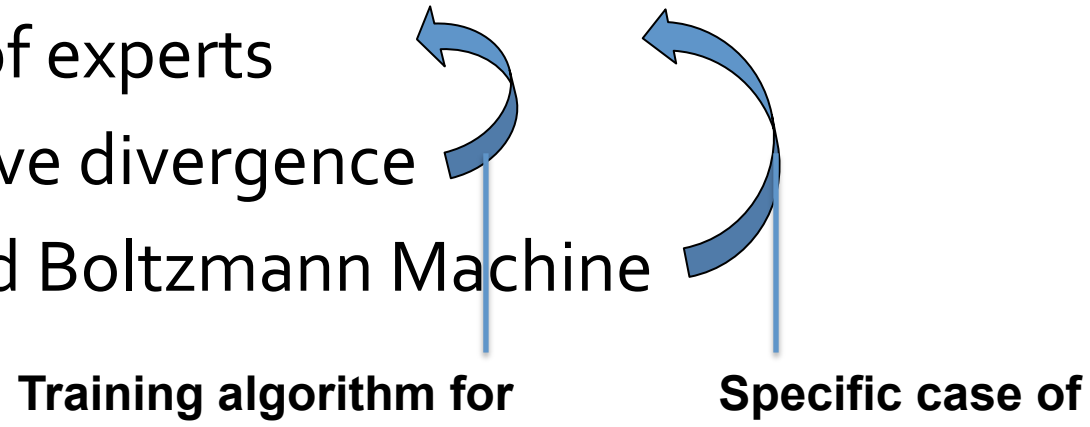


Restricted Boltzmann Machines

RBM in the different layers can be independently trained.

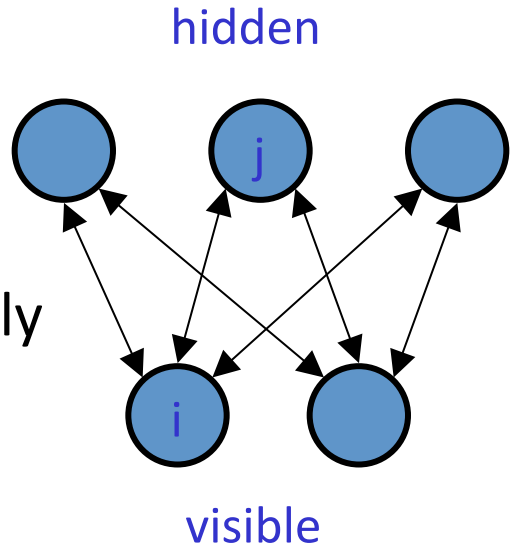
Understanding RBM

- We saw that the goal is to construct **deep** architectures
- To do so, we need to train each layer independently
- This gets rid of the gradient diffusion problem
- But we still need some information before going there
- Restricted Boltzmann machine
  - Product of experts
  - Contrastive divergence
  - Restricted Boltzmann Machine



Restricted Boltzmann Machine

- Restrict connectivity to make learning easier
  - Only one layer of hidden units.
    - Deal with more layers later
  - No connections between hidden units.
- In an RBM, the hidden units are conditionally independent given the visible states.
  - So can quickly get an unbiased sample from the posterior distribution when given a data-vector.
  - This is a big advantage over directed belief nets

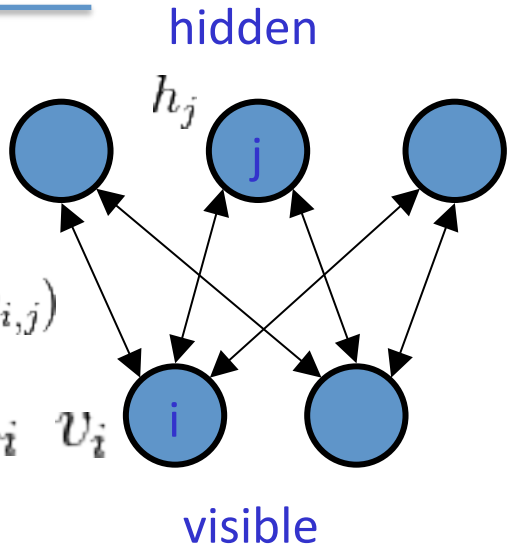


Restricted Boltzmann Machine

As well as a set of **biases** (thresholds) b_j

Still a matrix of (reciprocal) **weights** $W = (w_{i,j})$

As well as a set of **biases** (thresholds) a_i v_i



The *energy* of a configuration (pair of boolean vectors) (v,h) is given as

$$E(v,h) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_i \sum_j v_i w_{i,j} h_j$$

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

bias of unit i

binary state of unit i in joint configuration v, h

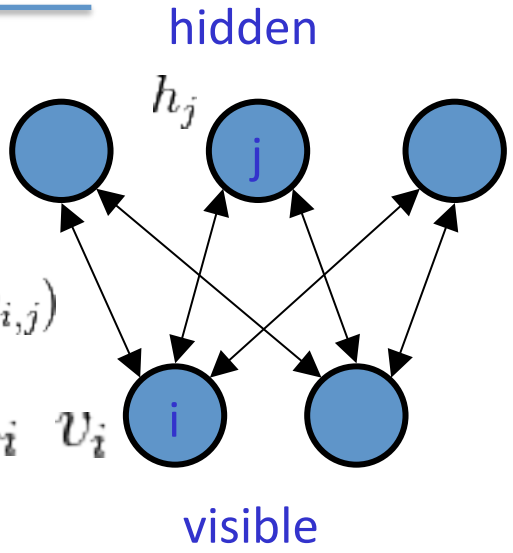
weight between units i and j

Restricted Boltzmann Machine

As well as a set of **biases** (thresholds) b_j

Still a matrix of (reciprocal) **weights** $W = (w_{i,j})$

As well as a set of **biases** (thresholds) a_i v_i



The *energy* of a configuration (pair of boolean vectors) (v,h) is given as

$$E(v,h) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_i \sum_j v_i w_{i,j} h_j$$

Remember that we rely on this **energy function** $P(v,h) = \frac{1}{Z} e^{-E(v,h)}$

The (marginal) probability of a visible (input) vector is the **sum over all possible hidden layer configurations** $P(v) = \frac{1}{Z} \sum_h e^{-E(v,h)}$

Given the structure we have mutual independence in units ! $P(v|h) = \prod_{i=1}^{m} P(v_i|h) \quad P(h|v) = \prod_{j=1}^{n} P(h_j|v)$

Hard equilibrium to achieve

- With:

$$\frac{\partial \log p(\mathrm{D} \mid \theta_1, \ldots, \theta_n)}{\partial \theta_m} \propto \left\langle \frac{\partial \log f_m\left(\vec{d} \mid \theta_m\right)}{\partial \theta_m} \right\rangle_{P^0} - \left\langle \frac{\partial \log f_m\left(\vec{c} \mid \theta_m\right)}{\partial \theta_m} \right\rangle_{P_\theta^\infty}$$

  we can now train our Product of Experts model.

- But… there's a problem:
  - P_θ^∞ is computationally infeasible to obtain
  - To obtain a stopping criterion … we should converge to **the exact** target distribution. Often this takes a *very* long time!

Surprising result

- Everything that one weight needs to know about the other weights and the data to do maximum likelihood learning
- Is contained in the difference of two correlations.

$$\frac{\partial \log p(\mathbf{v})}{\partial w_{ij}} = \left\langle s_i s_j \right\rangle_{\mathbf{v}} - \left\langle s_i s_j \right\rangle_{free}$$

Derivative of log probability of one training vector

Expected value of product of states at thermal equilibrium when the **training vector is clamped on the visible units**

Expected value of product of states at thermal equilibrium **when nothing is clamped**

Batch learning algorithm

- Positive phase
  - Clamp a data vector on the visible units.
  - Let the hidden units reach thermal equilibrium at a temperature of 1
  - Sample $s_i s_j$ for all pairs of units
  - Repeat for all data vectors in the training set.
- Negative phase
  - Do not clamp any of the units
  - Let the whole network reach thermal equilibrium at a temperature of 1
  - Sample $s_i s_j$ for all pairs of units
  - Repeat many times to get good estimates
- Weight updates
  - Update each weight by an amount proportional to the difference in $< s_i s_j >$ in the two phases.

Solution = **contrastive divergence**

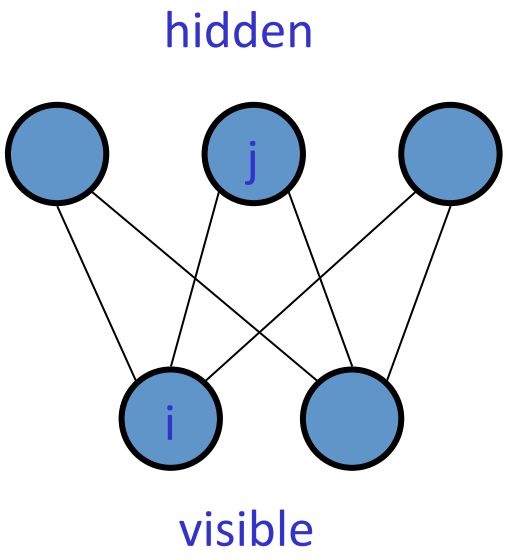
$$\frac{\partial \log p(\mathrm{D}\,|\,\theta_1,\ldots,\theta_n)}{\partial \theta_m} \propto \left\langle \frac{\partial \log f_m\left(\vec{d}\,|\,\theta_m\right)}{\partial \theta_m} \right\rangle_{P^0} - \left\langle \frac{\partial \log f_m\left(\vec{c}\,|\,\theta_m\right)}{\partial \theta_m} \right\rangle_{P_\theta^1}$$

- Now we dont have to run the process to convergence, instead we can stop after 1 iteration (a few iterations more typically)
- Why does this work?
  - Attempts to minimize the ways that the model distorts the data.

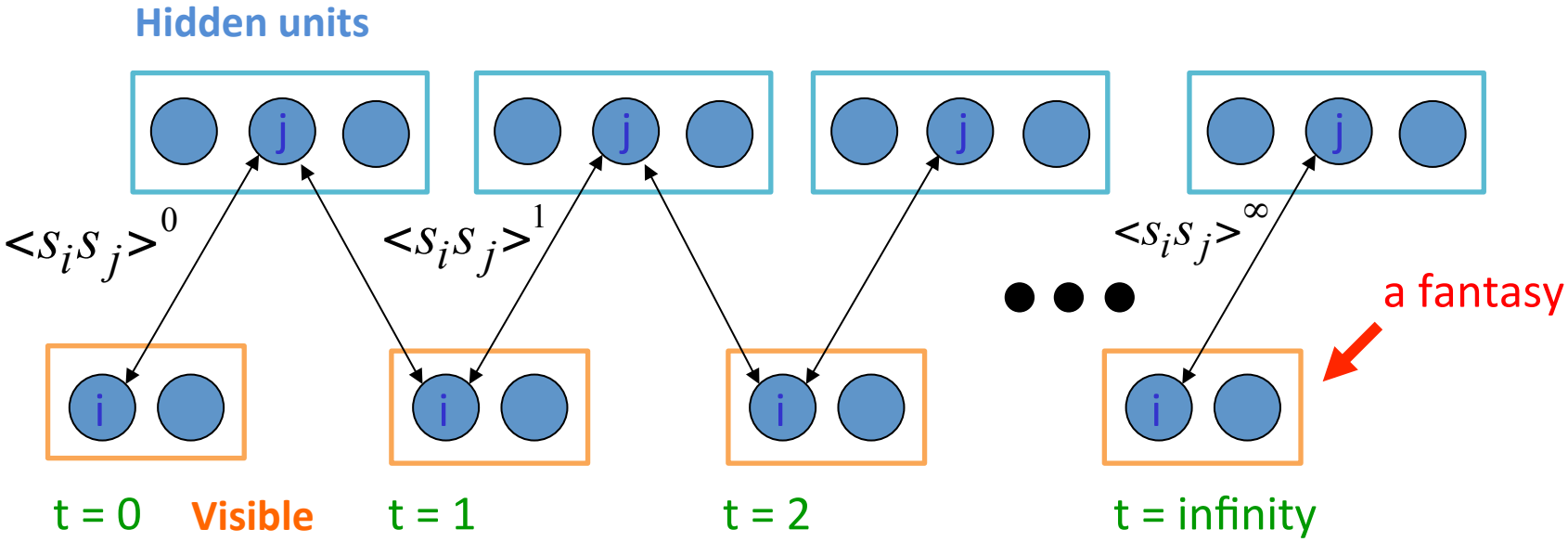
Restricted Boltzmann Machine

- In an RBM, the **hidden units are conditionally independent given the visible states**. It only takes one step to reach thermal equilibrium when the visible units are clamped.
  - Can quickly get the exact value of :

Impossible d'afficher l'image. Votre ordinateur manque peut-être de mémoire pour ouvrir l'image ou l'image est endommagée. Redémarrez l'ordinateur, puis ouvrez à nouveau le fichier. Si le x rouge est toujours affiché, vous devrez peut-être supprimer l'image avant de la réinsérer.



Training an RBM

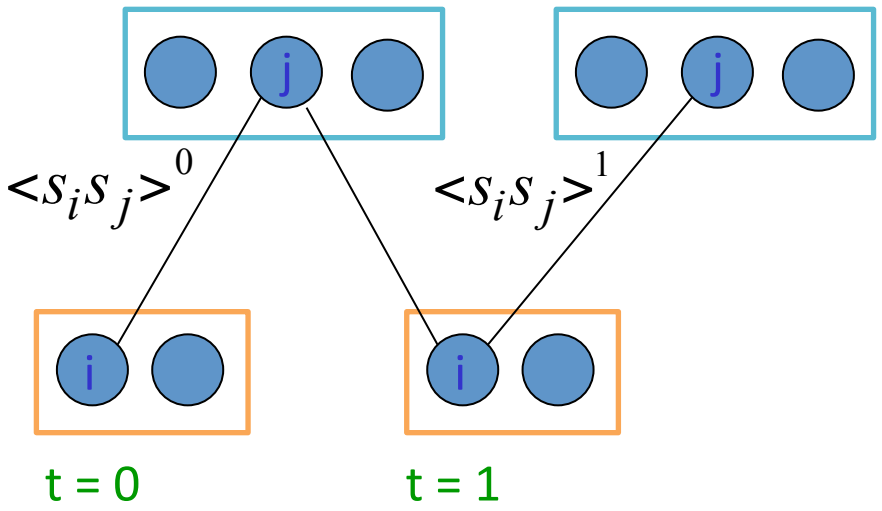


Start with a training vector on the visible units.

Then alternate between updating all the hidden units in parallel and updating all the visible units in parallel.

$$\Delta w_{ij} = \varepsilon \left(<s_i s_j>^0 - <s_i s_j>^\infty \right)$$

Contrastive divergence learning



Start with a training vector on the visible units.

Update all the hidden units in parallel

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

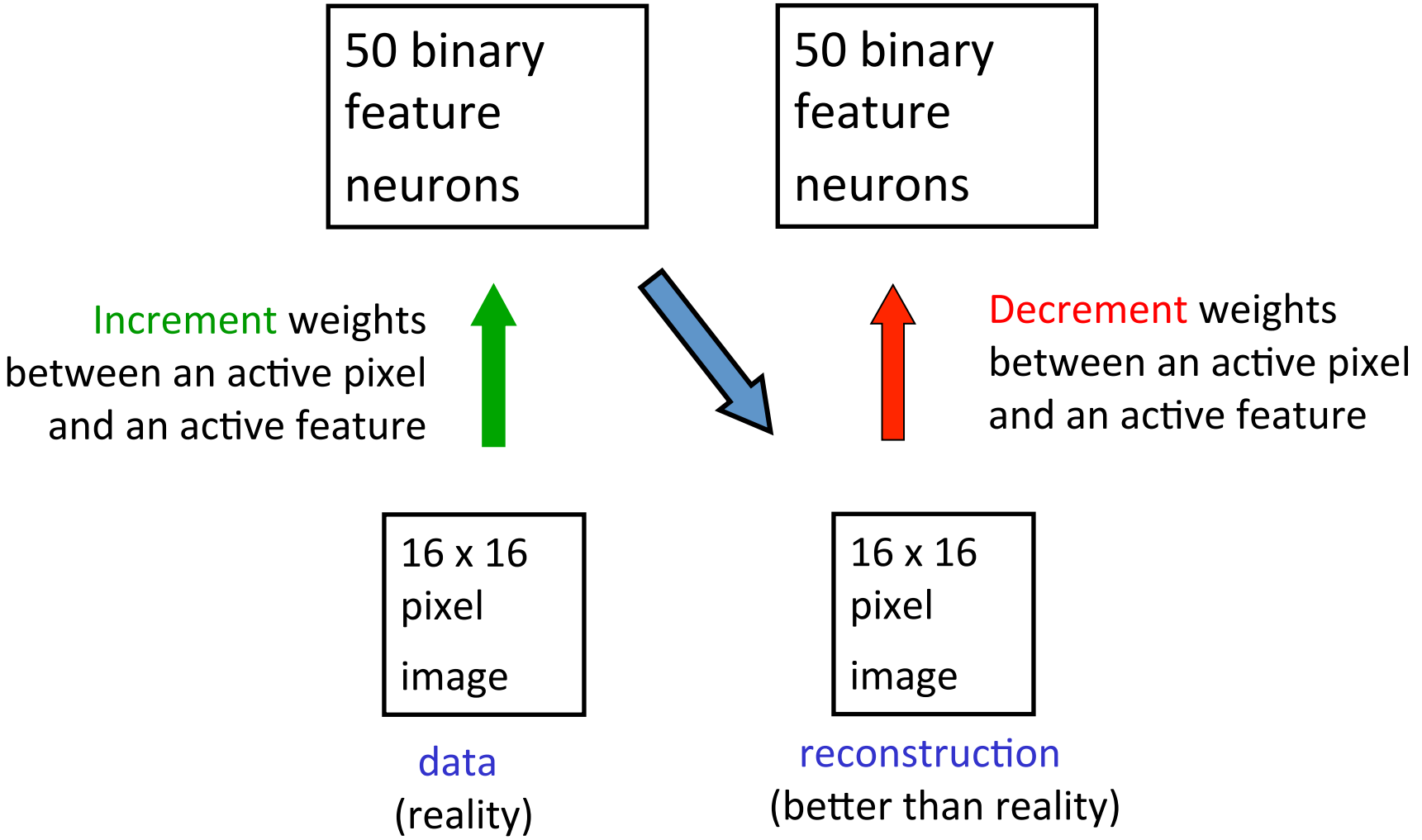
Update the hidden units again.

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

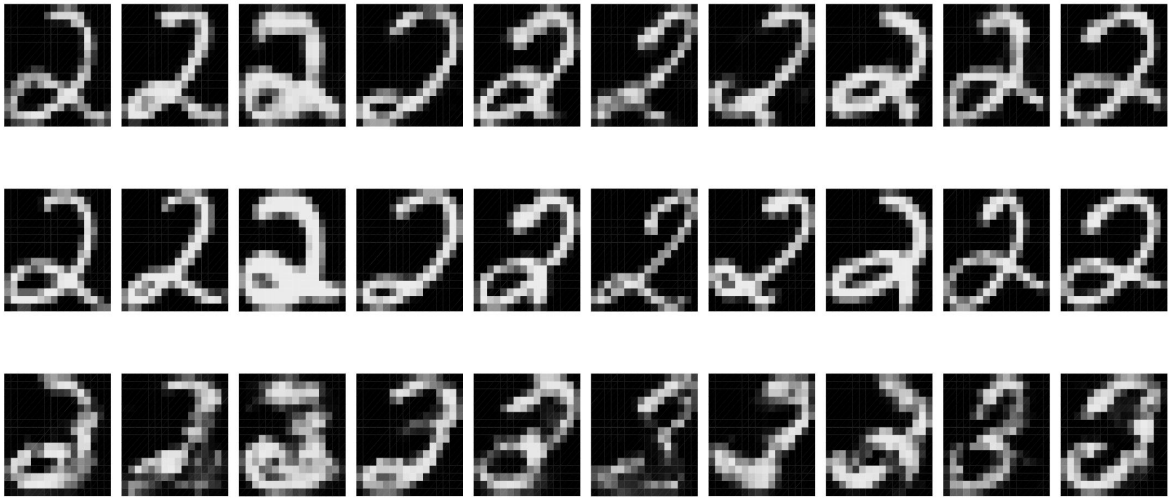
This is not following the gradient of the log likelihood.

Considering infinite directed nets it will see why it works.

Learning in vision



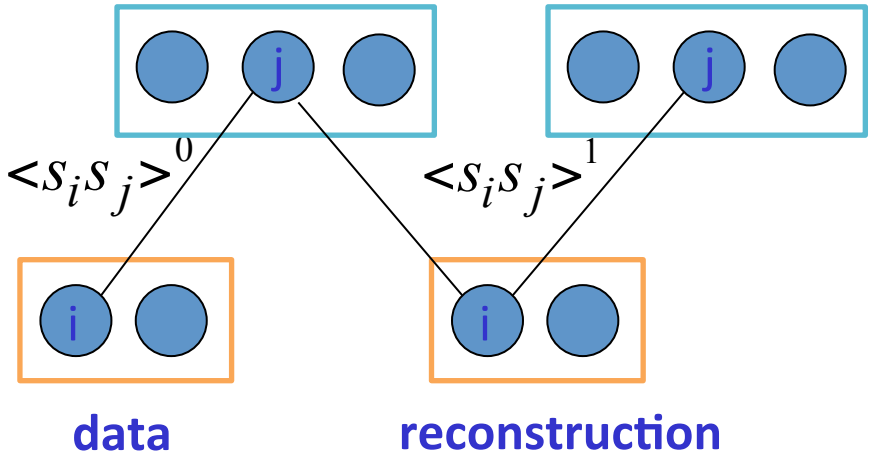
Learning digit classes



Reconstructions by model trained on 2's

Data

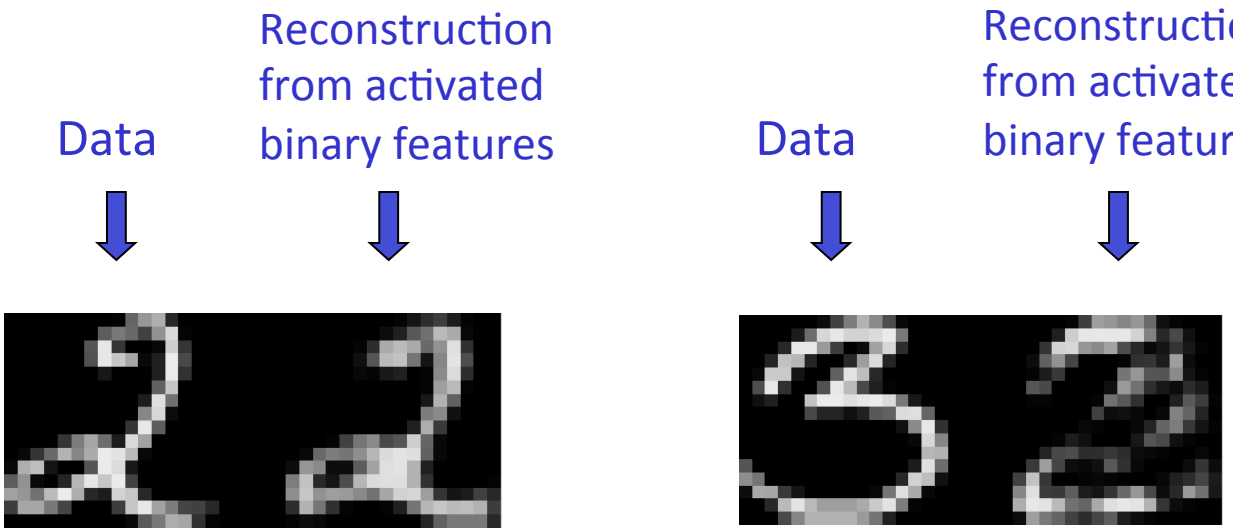
Reconstructions by model trained on 3's



100 hidden units (features)

256 visible units (pixels)

Quality of reconstruction

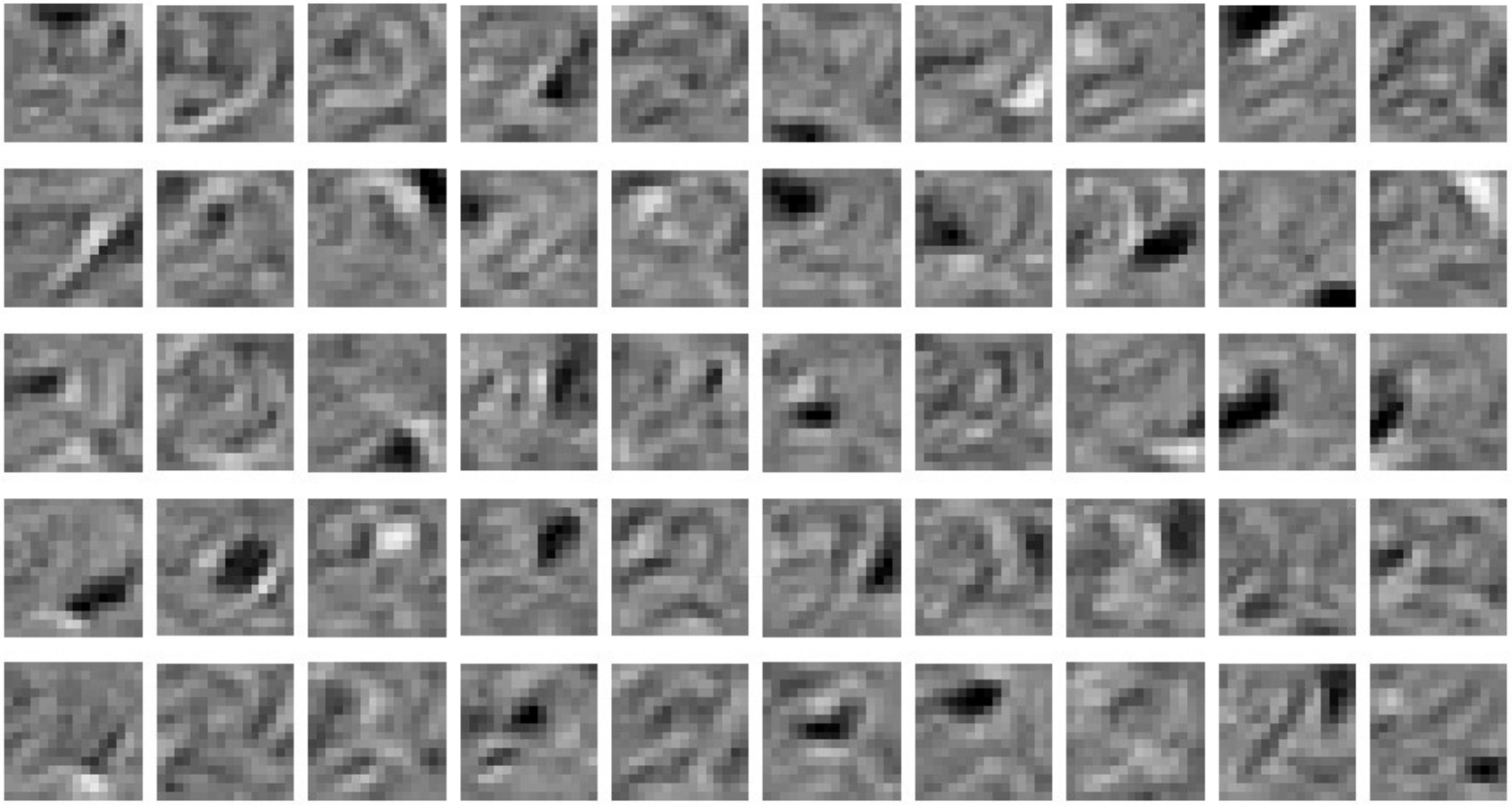


New test images from the digit class that the model was trained on

Images from an unfamiliar digit class

(the network tries to see every image as a 2)

Final weights for each neuron



Each neuron grabs a different feature.

Fine-tuning

After learning many layers of features, we can fine-tune the features to improve generation.

1. Do a stochastic bottom-up pass
   - Adjust the top-down weights to be good at reconstructing the feature activities in the layer below.
2. Do a few iterations of sampling in the top level RBM

   -- Adjust the weights in the top-level RBM.
3. Do a stochastic top-down pass
   - Adjust the bottom-up weights to be good at reconstructing the feature activities in the layer above.
4. **Not required! But helps the recognition rate (-ml).**

RBM itself …

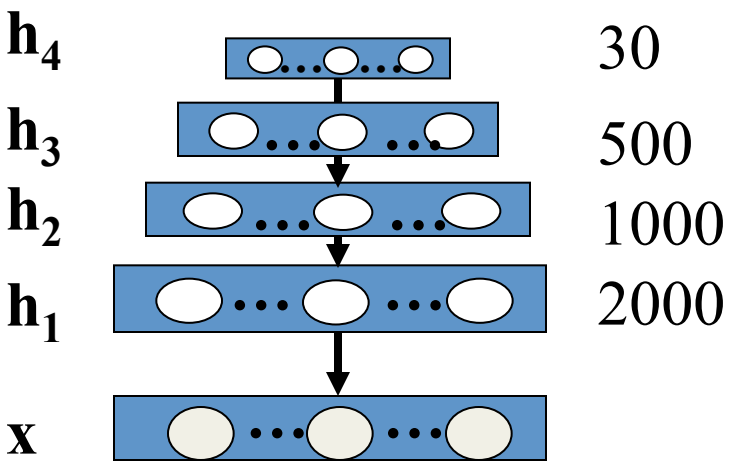
Has many applications !

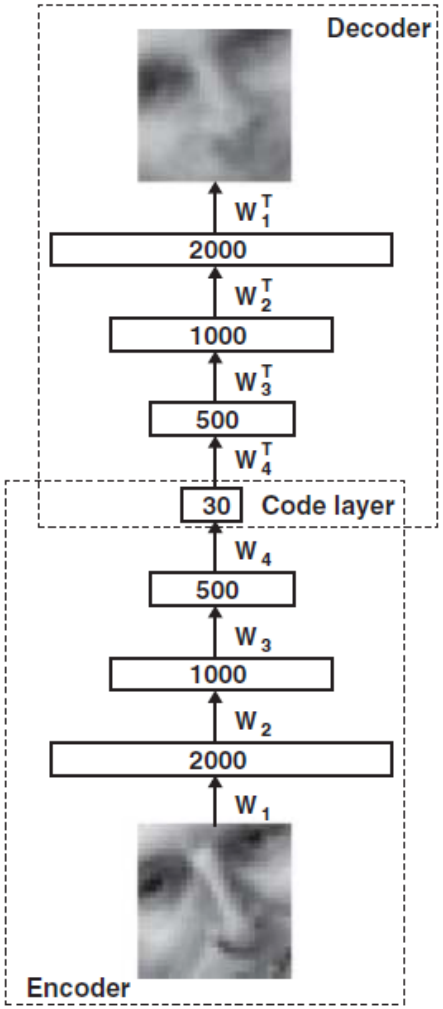
- Multiclass classification
- Collaborative filtering
- Motion capture modeling
- Information retrieval
- Modeling natural images
- Segmentation

Y Li, D Tarlow, R Zemel, Exploring compositional high order pattern potentials for structured output learning, CVPR 2013
V. Mnih, H Larochelle, GE Hinton , Conditional Restricted Boltzmann Machines for Structured Output Prediction, Uncertainty in Artificial Intelligence, 2011.
Larochelle, H., & Bengio, Y. (2008). Classification using discriminative restricted boltzmann machines. ICML, 2008.
Salakhutdinov, R., Mnih, A., & Hinton, G. E. (2007). Restricted Boltzmann machines for collaborative filtering. ICML 2007.
Salakhutdinov, R., & Hinton, G. E. (2009). Replicated softmax: an undirected topic model., NIPS 2009.
Osindero, S., & Hinton, G. E. (2008). Modeling image patches with a directed hierarchy of markov random field., NIPS 2008

Inference in DBN

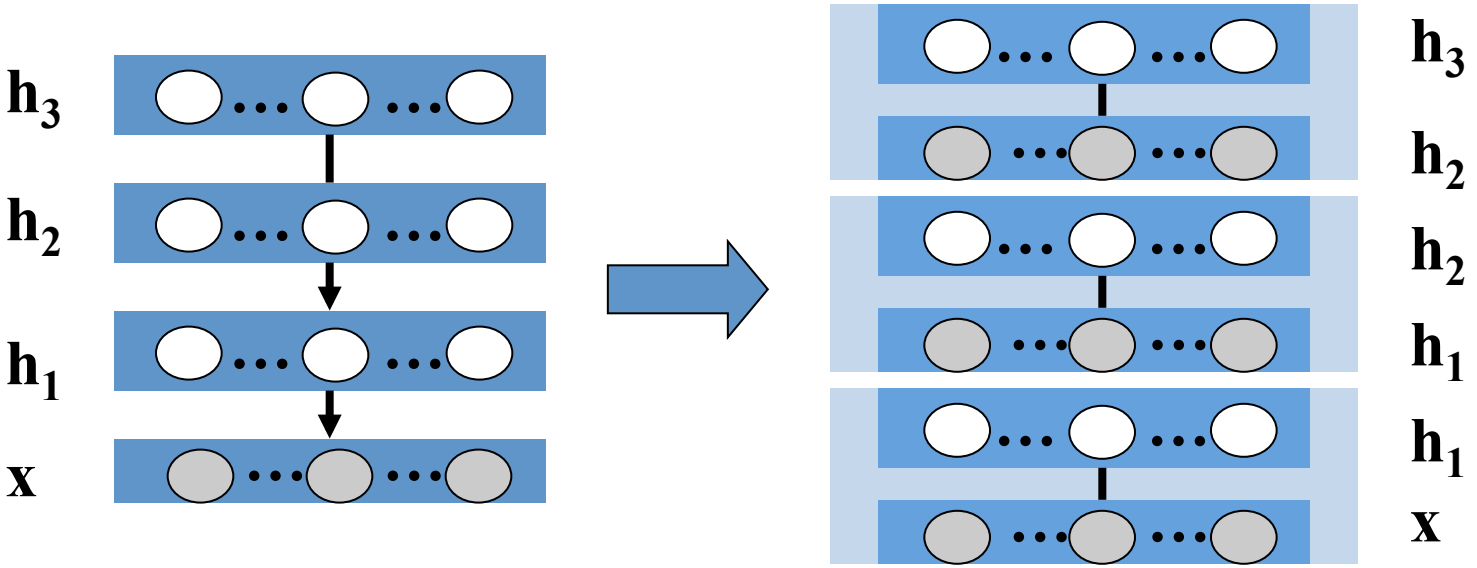
- Inference problem (the problem of explaining away)
  - Solution: Complementary prior





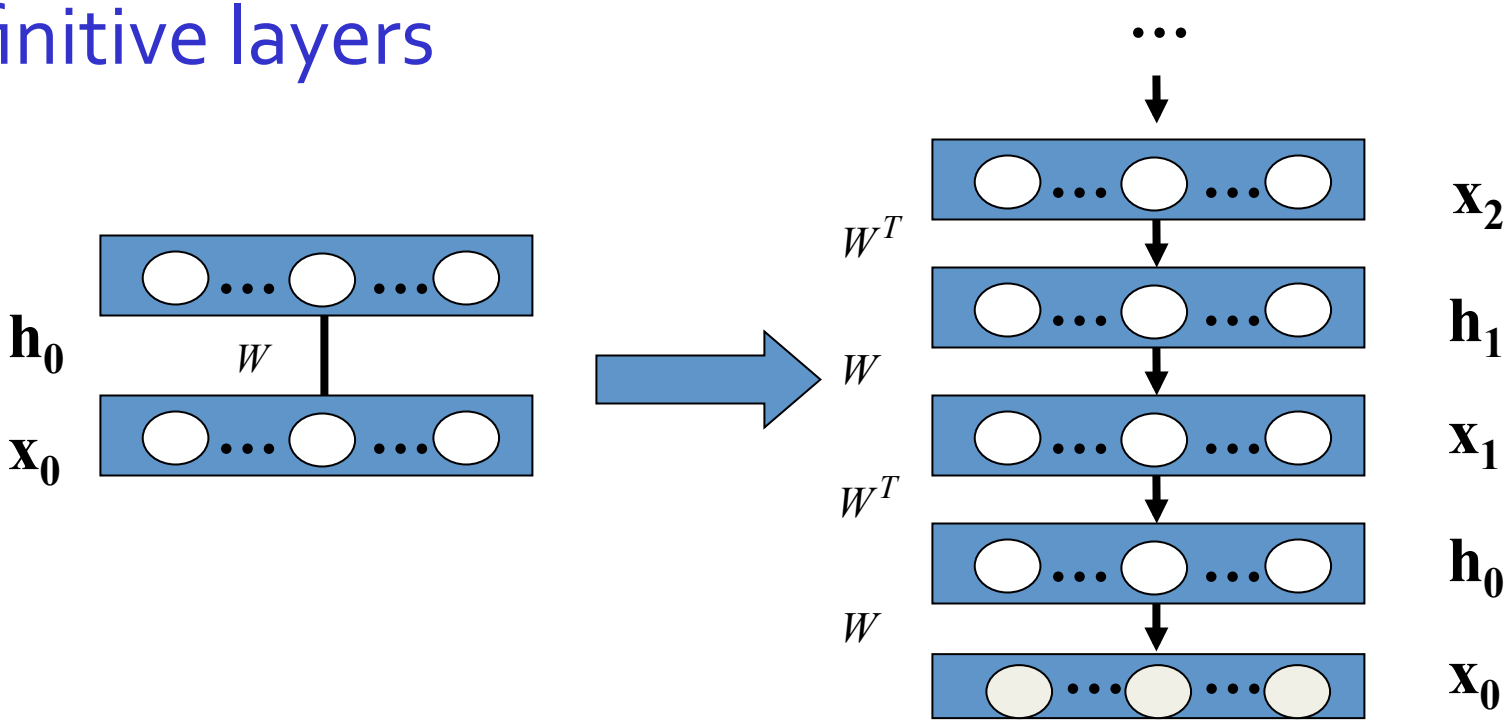
Deep Belief Net

- Explaining away problem of Inference
  Sol: Complementary prior
- Learning problem
  - Greedy layer by layer RBM training (optimize lower bound) and fine tuning
  - Contrastive divergence for RBM training

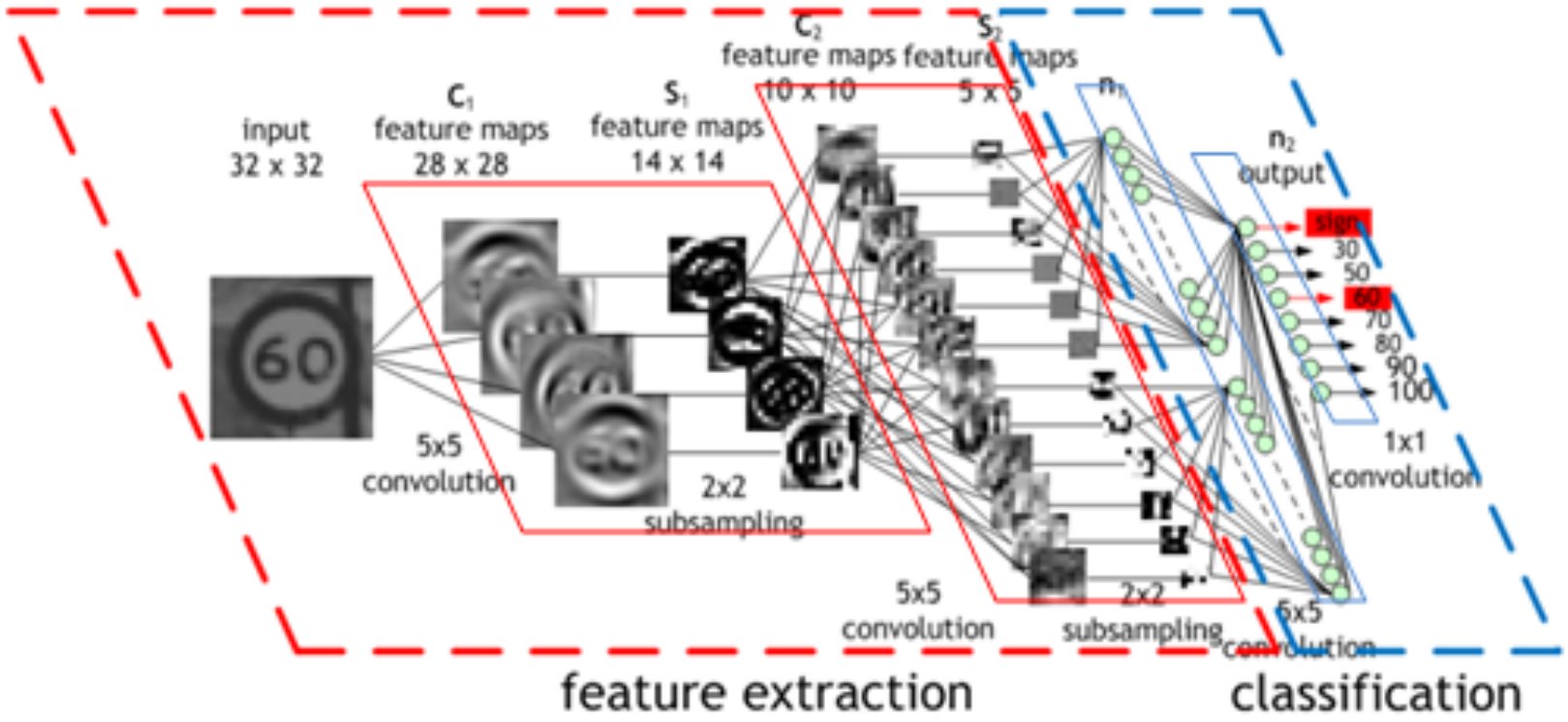


Deep Belief Net and RBM

- RBM can be considered as DBN that has infinitive layers



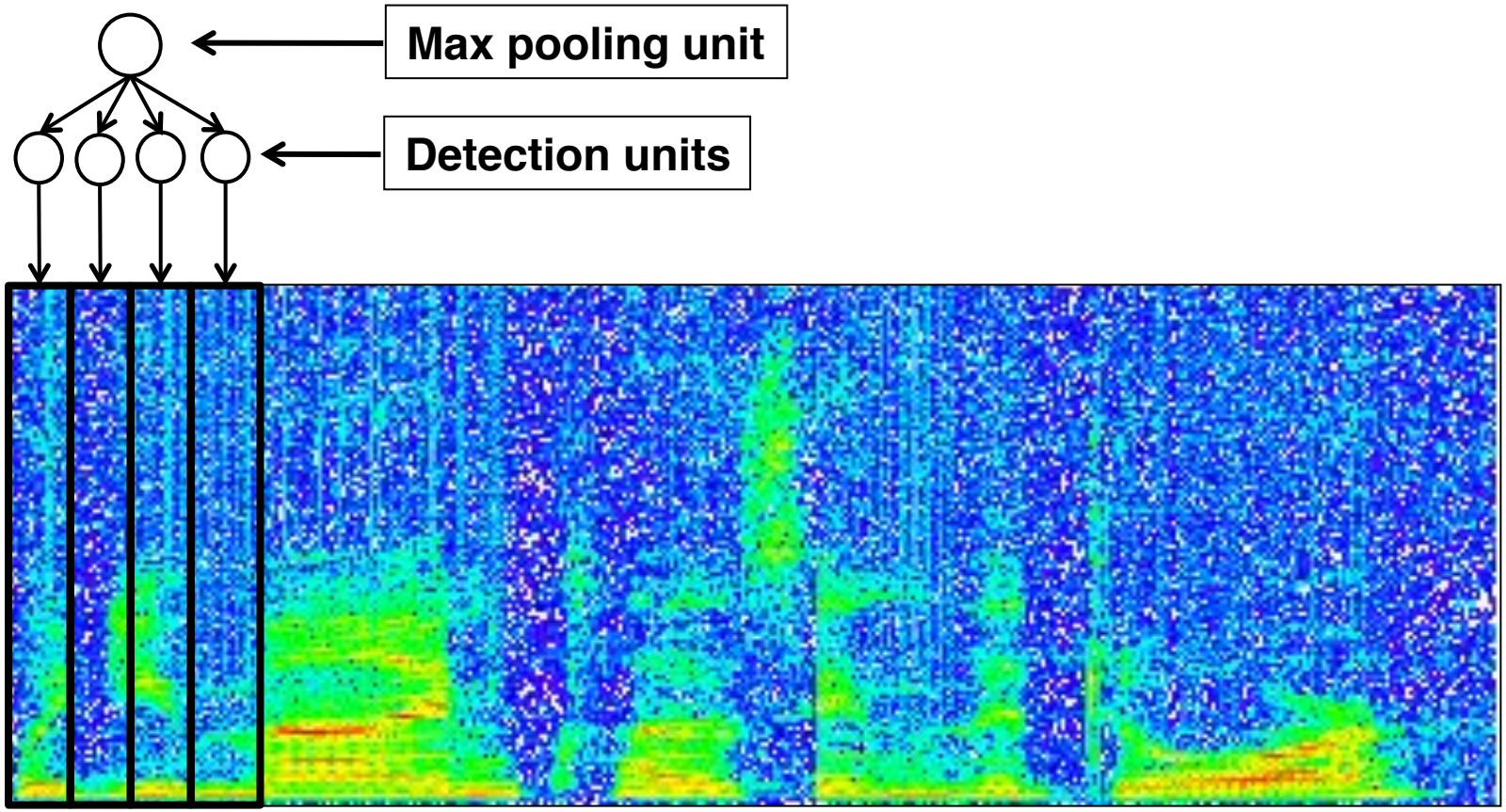
Convolutional Network



- A special case of feed-forward multilayer neural network
- The architecture can be deep or not, the term convolutional comes from

Individual neurons are tiled in such a way that they respond to overlapping regions

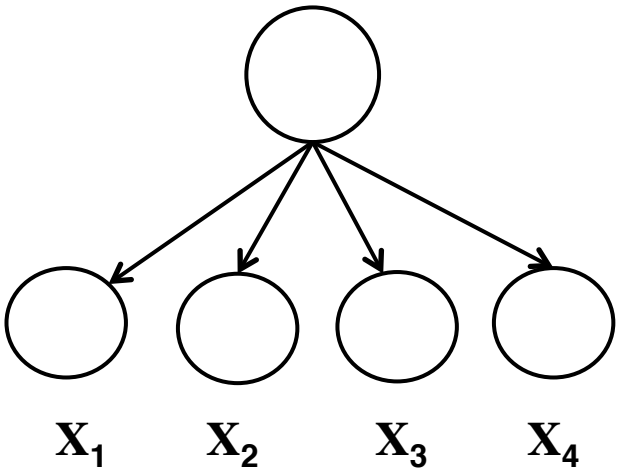
Convolutional DBN for audio



Probabilistic max pooling

Convolutional Neural net:

$$\max \{x_1, x_2, x_3, x_4\}$$



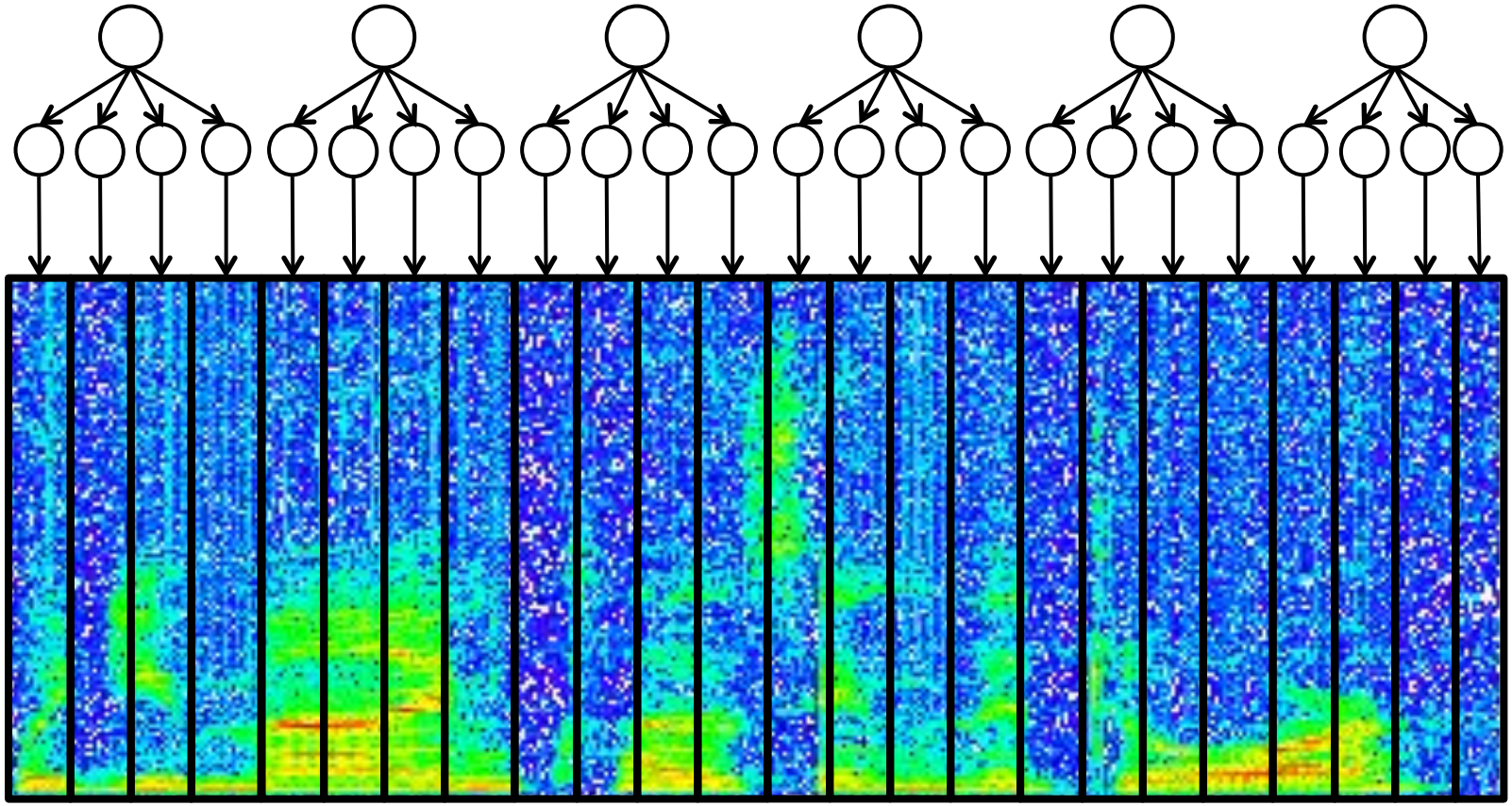
Where x_i are real numbers.

Convolutional DBN

x_i are {0,1}, and *mutually exclusive*.
Thus, 5 possible cases
{0,0,0,0} => 0
{1,0,0,0} => 1
{0,1,0,0} => 1
{0,0,1,0} => 1
{0,0,0,1} => 1

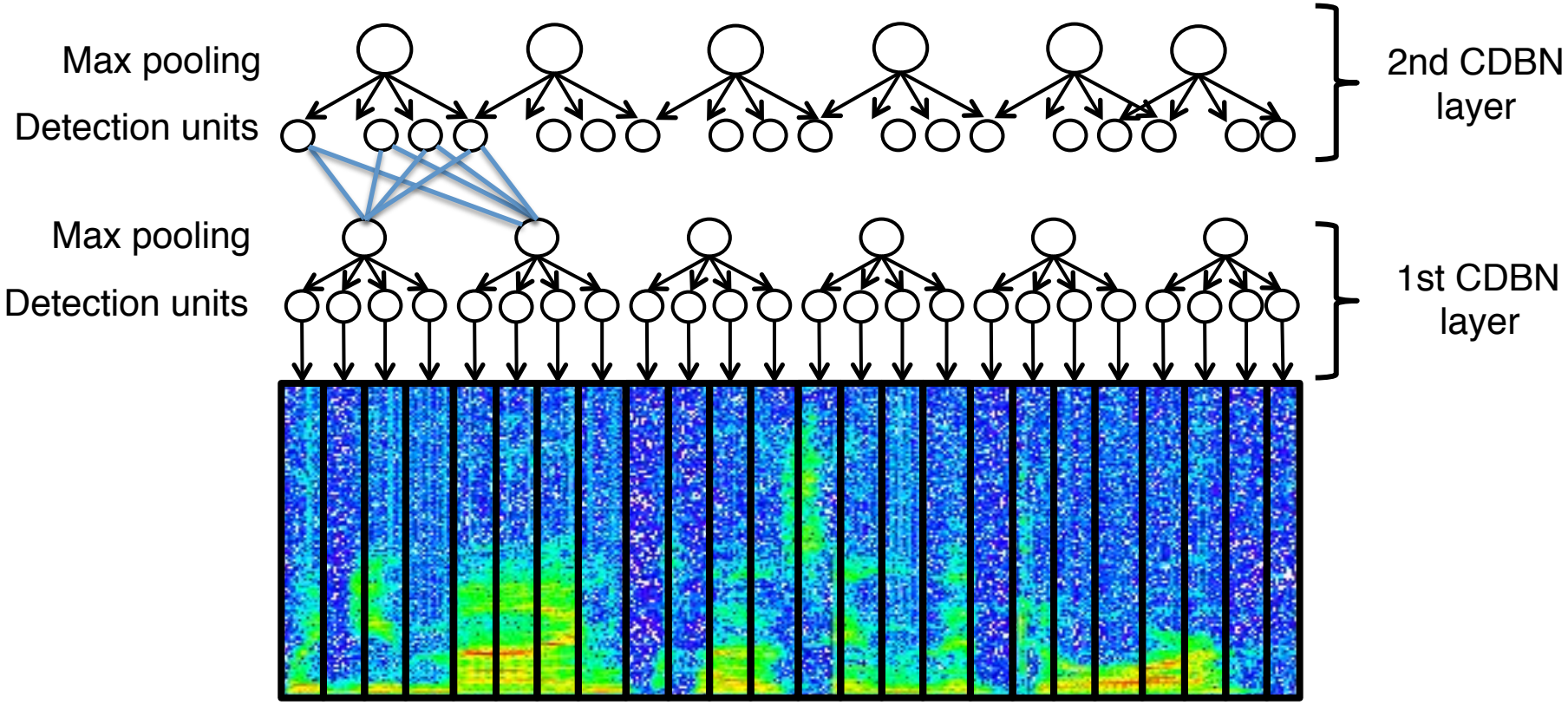
Collapse 2^n configurations into n+1 configurations. Permits bottom up and top down inference.

Convolutional DBN for audio



Spectrogram

Convolutional DBN for audio



Summary

- Deep belief net (DBN)
  - is a network with deep layers, which provides strong representation power;
  - Can be viewed as composition of simple unsupervised networks (RBM or autoencoders)
  - is a generative model;
  - can be learned by layerwise (RBM using Contrastive Divergence)
  - has many applications and more applications is yet to be found.