Learning Deep Generative Models

9.520 Class 19

Ruslan Salakhutdinov
BCS and CSAIL, MIT

Talk Outline

1. Introduction.

1. Autoencoders, Boltzmann Machines.

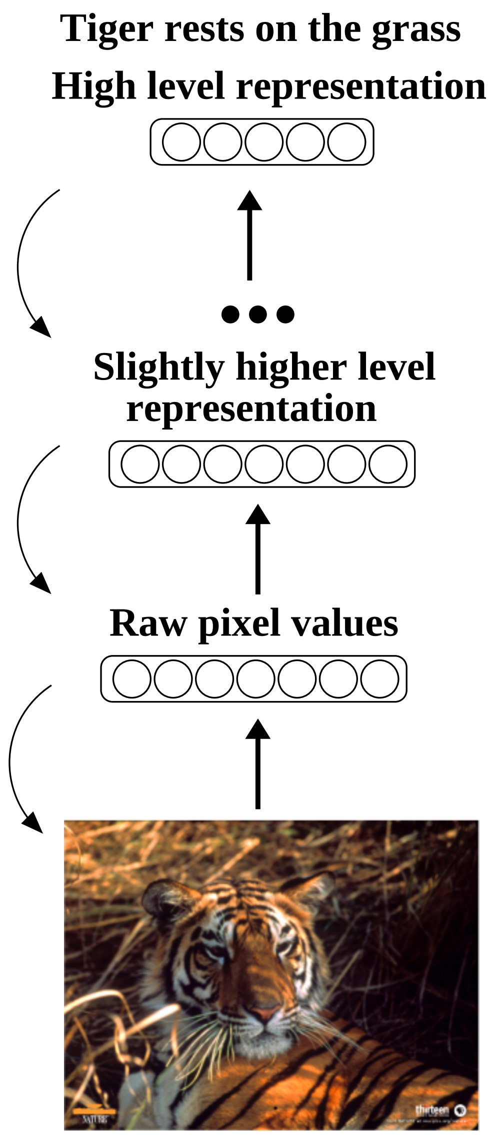
2. Deep Belief Networks (DBN’s).

3. Learning Feature Hierarchies with DBN’s.

4. Deep Boltzmann Machines (DBM’s).

5. Extensions.

Long-term Goal


Tiger rests on the grass
High level representation
Slightly higher level representation
Raw pixel values

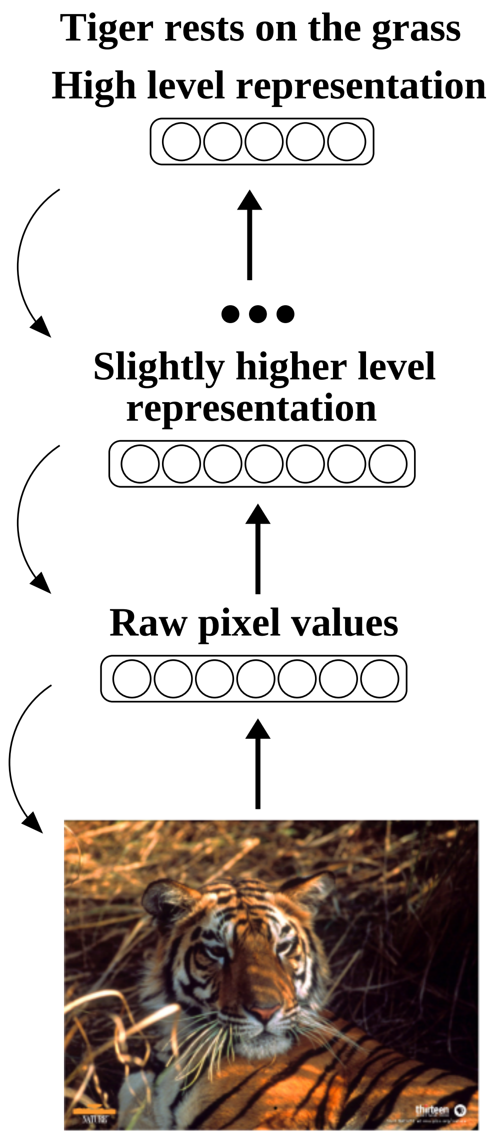
Learn progressively complex high-level representations.

Use bottom-up + top-down cues.

Build high-level representations from large unlabeled datasets.

Labeled data is used to only slightly adjust the model for a specific task.

Challenges


Tiger rests on the grass
High level representation
Slightly higher level representation
Raw pixel values

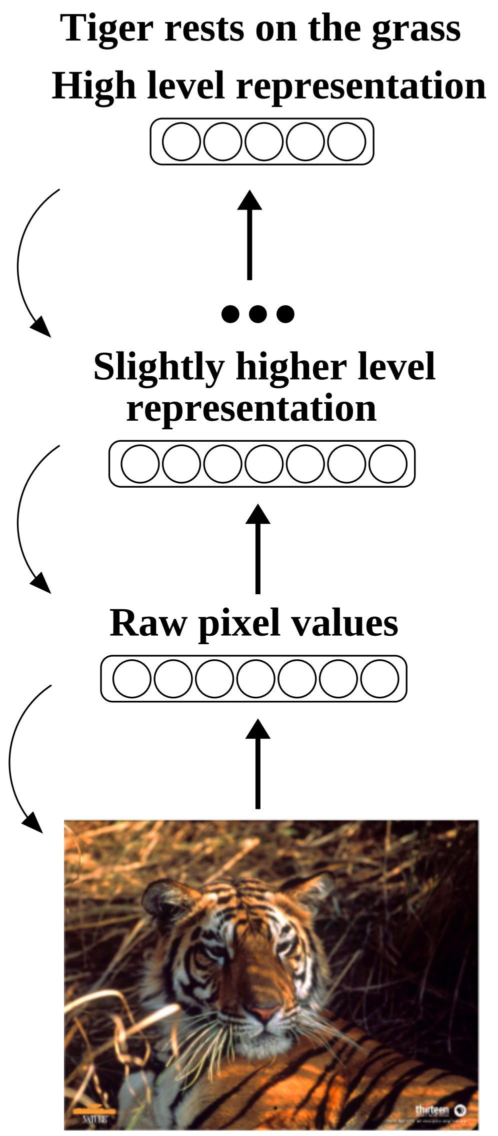
Deep models are composed of several layers of nonlinear modules.

Associated loss functions are almost always non-convex.

Many bad local optima makes deep models very difficult to optimize.

Idea: learn one layer at a time.

Key Requirements


Tiger rests on the grass
High level representation
Slightly higher level representation
Raw pixel values

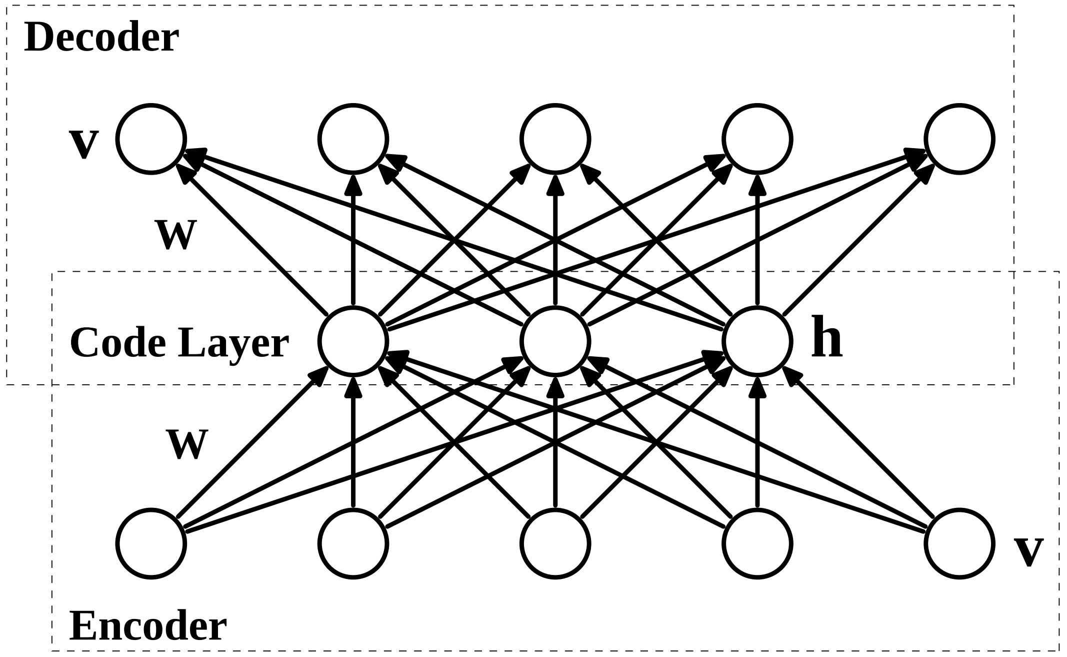
Online Learning.

Learning should scale to large datasets, containing millions or billions examples.

Inferring high-level representation should be fast: fraction of a second.

Demo.

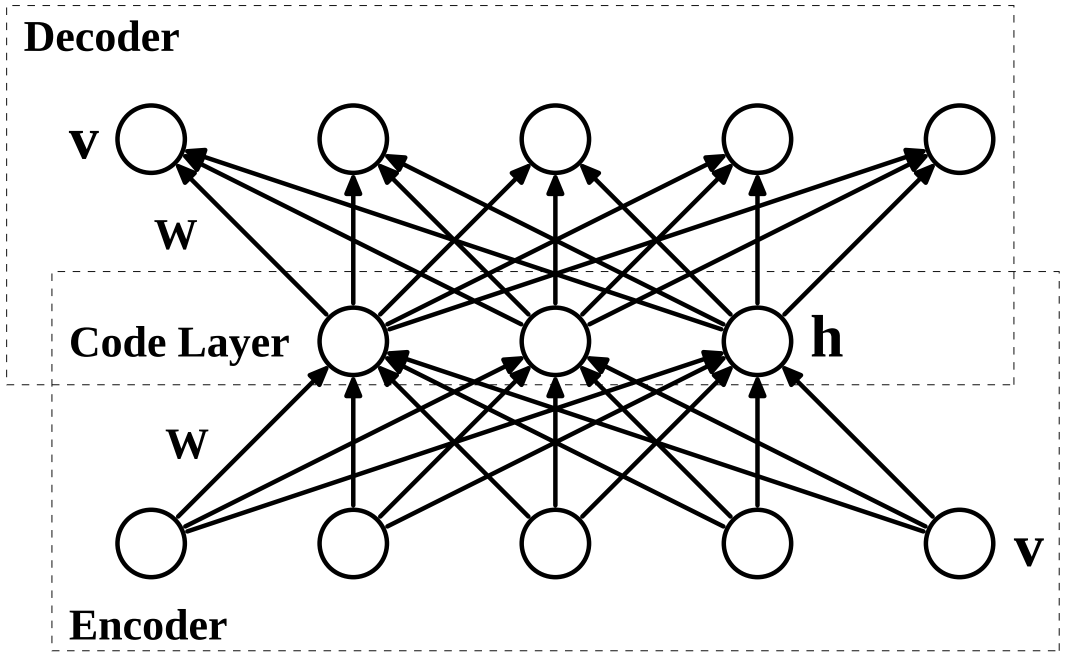
Autoencoders



Consider having D binary visible units $\mathbf{v}$ and K binary hidden units $\mathbf{h}$.

Idea: transform data into a (low-dimensional) code and then reconstruct the data from the code.

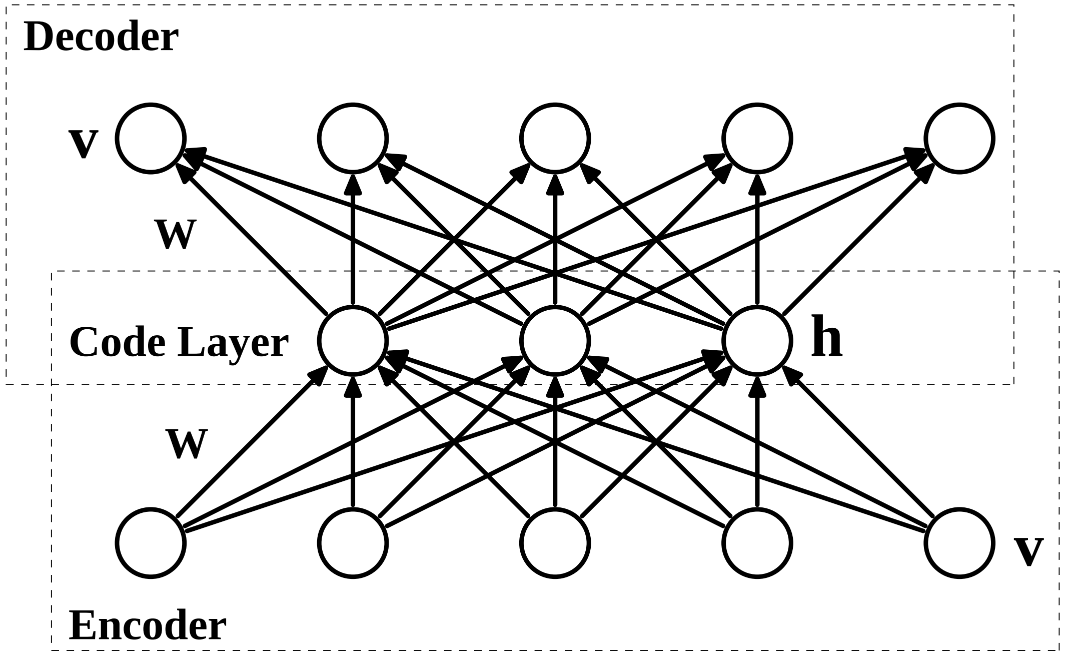
Autoencoders



Encoder: $h_j = \dfrac{1}{1+\exp(-\sum_i v_i W_{ij})}, \; j = 1, ..., K.$

Decoder: $\hat{v}_i = \dfrac{1}{1+\exp(-\sum_j h_j W_{ij})}, \; i = 1, ..., D.$

Autoencoders



Minimize reconstruction error:

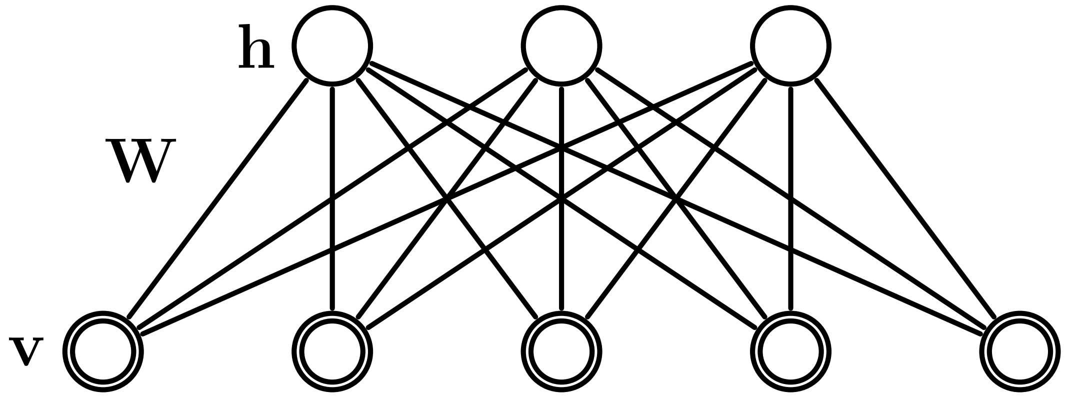
$$\min_W \mathrm{Loss}(\mathbf{v}, \hat{\mathbf{v}}, W) + \mathrm{Penalty}(\mathbf{h}, W)$$

Loss functions: cross-entropy or squared loss.

Typically, one imposes l_1 regularization on hidden units $\mathbf{h}$ and l_2 regularization on parameters W (related to sparse coding).

Building Block: RBM's

Probabilistic Analog: Restricted Boltzmann Machines.

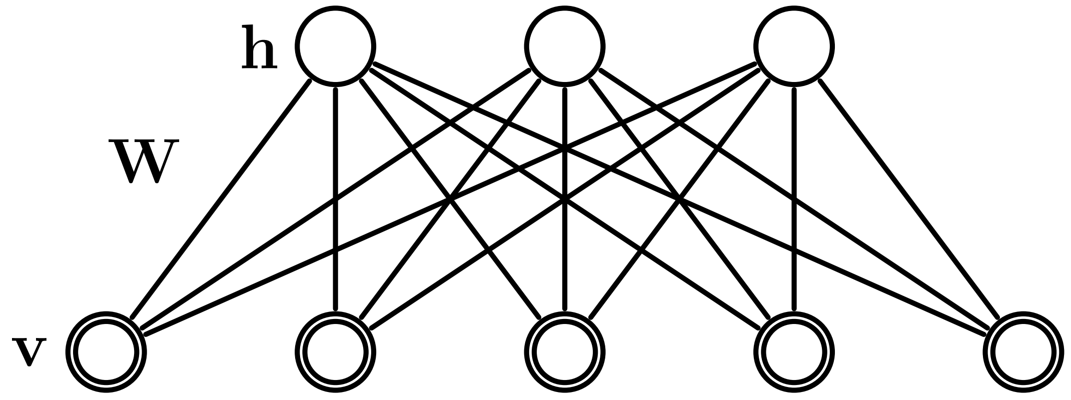


Visible stochastic binary units $\mathbf{v}$ are connected to hidden stochastic binary feature detectors $\mathbf{h}$:

$$P(\mathbf{v}, \mathbf{h}) = \frac{1}{\mathcal{Z}(W)} \exp \Big[\sum_{ij} v_i h_j W_{ij} \Big]$$

Markov Random Fields, Log-linear Models, Boltzmann machines.

Building Block: RBM's



$$P(\mathbf{v},\mathbf{h}) = \frac{1}{\mathcal{Z}(W)} \exp\big[\sum_{ij} v_i h_j W_{ij}\big],$$

where $\mathcal{Z}(W)$ is known as a partition function:

$$\mathcal{Z}(W) = \sum_{\mathbf{h},\mathbf{v}} \exp\big[\sum_{ij} v_i h_j W_{ij}\big].$$

Inference with RBM's

$$P(\mathbf{v}, \mathbf{h}) = \frac{1}{\mathcal{Z}} \exp\big(\sum_{ij} v_i h_j W_{ij}\big)$$

Conditional distributions over hidden and visible units are given by logistic functions:

$$p(h_j = 1|\mathbf{v}) = \frac{1}{1 + \exp(-\sum_i v_i W_{ij})}$$
$$p(v_i = 1|\mathbf{h}) = \frac{1}{1 + \exp(-\sum_j h_j W_{ji})}$$

Key Observation: Given $\mathbf{v}$, we can easily infer the distribution over hidden units.

Learning with RBM's

$$P_{model}(\mathbf{v}) = \sum_{\mathbf{h}} P(\mathbf{v}, \mathbf{h}) = \frac{1}{\mathcal{Z}} \sum_{\mathbf{h}} \exp\Big(\sum_{ij} v_i h_j W_{ij}\Big)$$

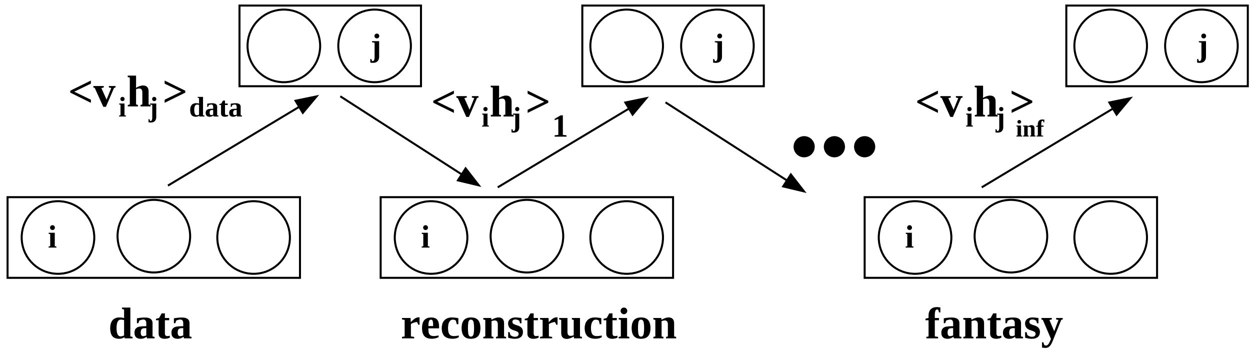
Maximum Likelihood learning:

$$\frac{\partial \log P(\mathbf{v})}{\partial W_{ij}} = \mathrm{E}_{P_{data}}[v_i h_j] - \mathrm{E}_{P_{model}}[v_i h_j],$$

where $P_{\mathrm{data}}(\mathbf{h}, \mathbf{v}) = P(\mathbf{h}|\mathbf{v}) P_{\mathrm{data}}(\mathbf{v})$, with $P_{\mathrm{data}}(\mathbf{v})$ representing the empirical distribution.

However, computing $\mathrm{E}_{P_{model}}$ is difficult due to the presence of a partition function $\mathcal{Z}$.

Contrastive Divergence



Maximum Likelihood learning:

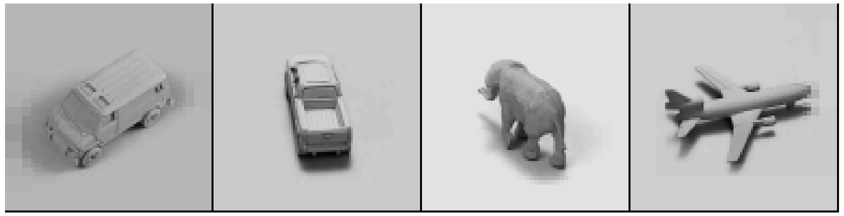
$$\Delta W_{ij} = \mathrm{E}_{P_{data}}[v_i h_j] - \mathrm{E}_{P_{model}}[v_i h_j]$$

Contrastive Divergence learning:

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

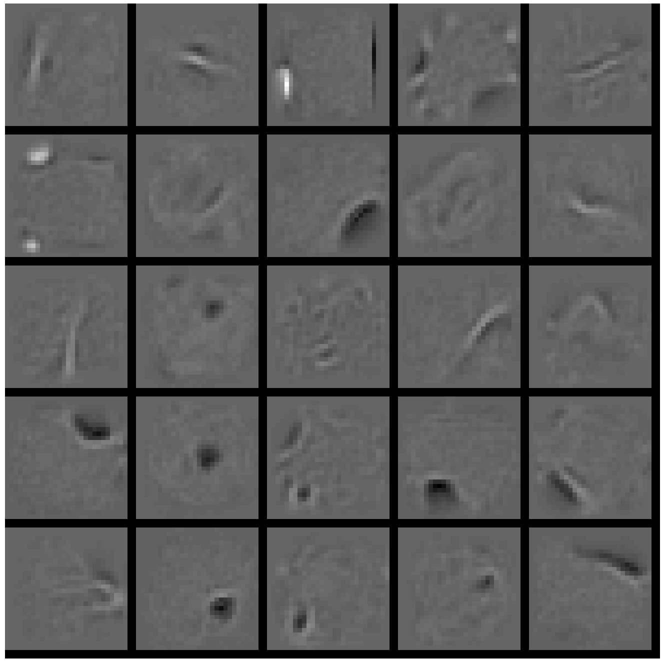
P_T represents a distribution defined by running a Gibbs chain, initialized at the data, for T full steps.

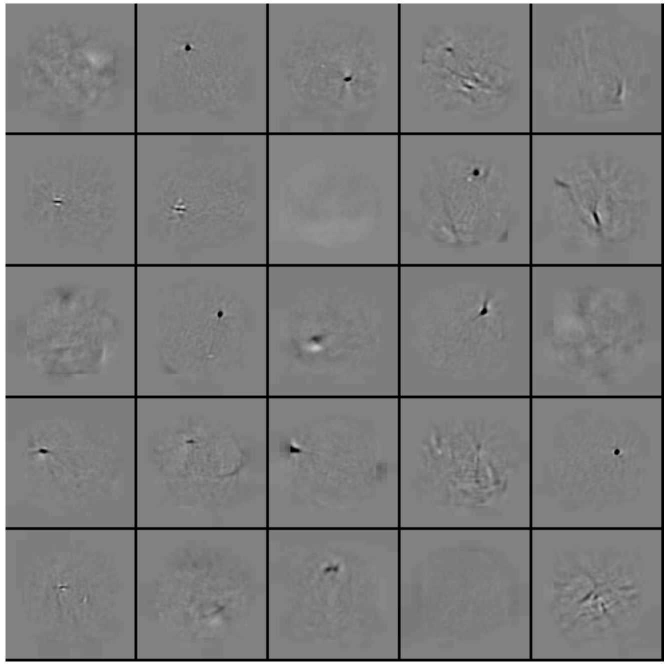
Learning with RBM's



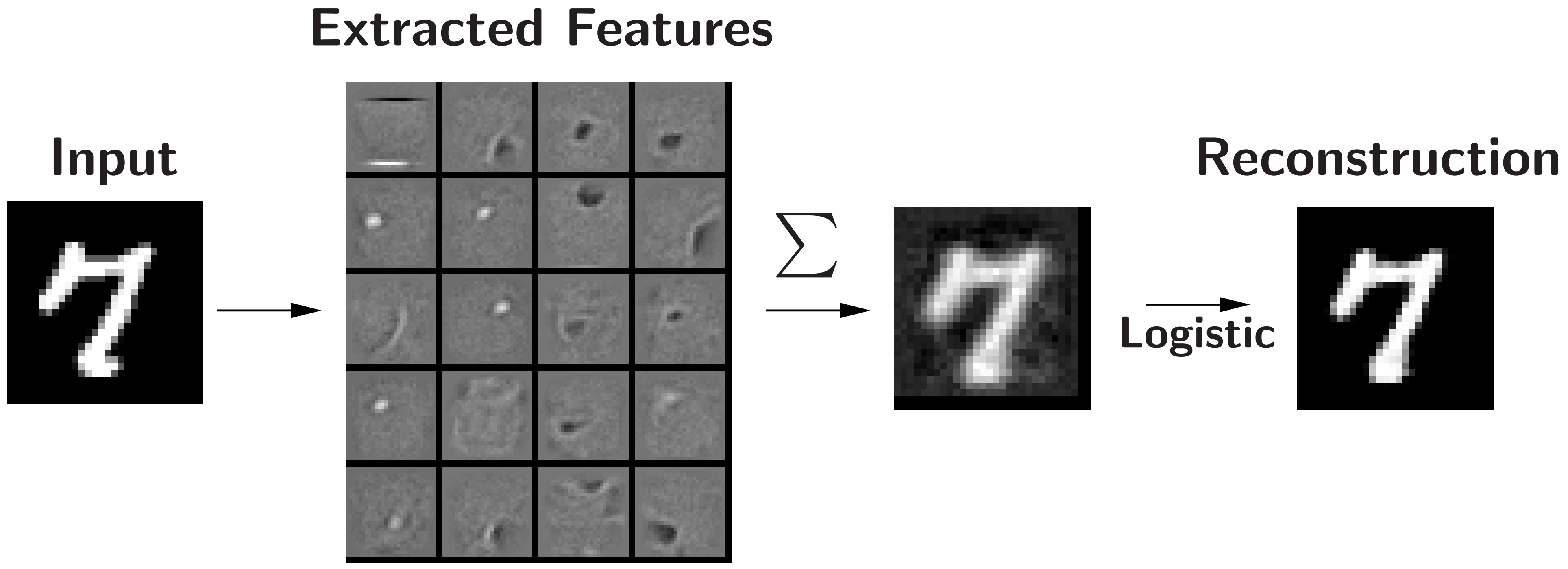
MNIST Digits NORB 3D Objects

Learned W



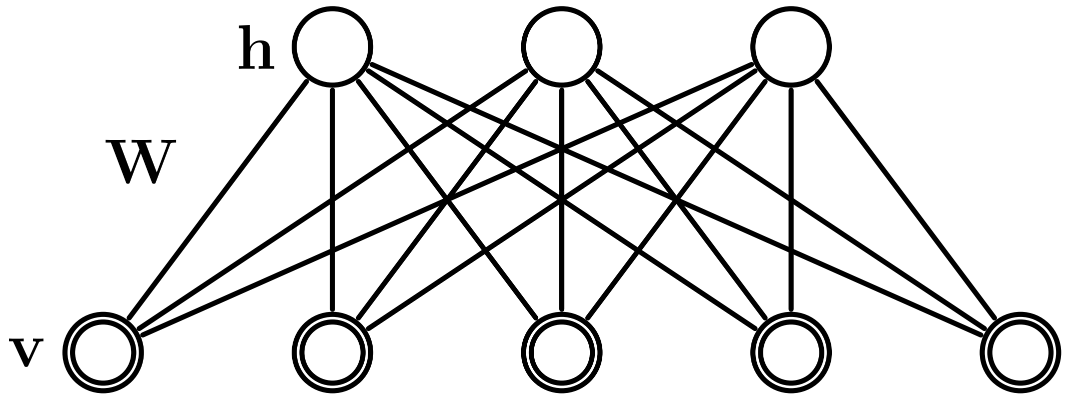


Learning with RBM's



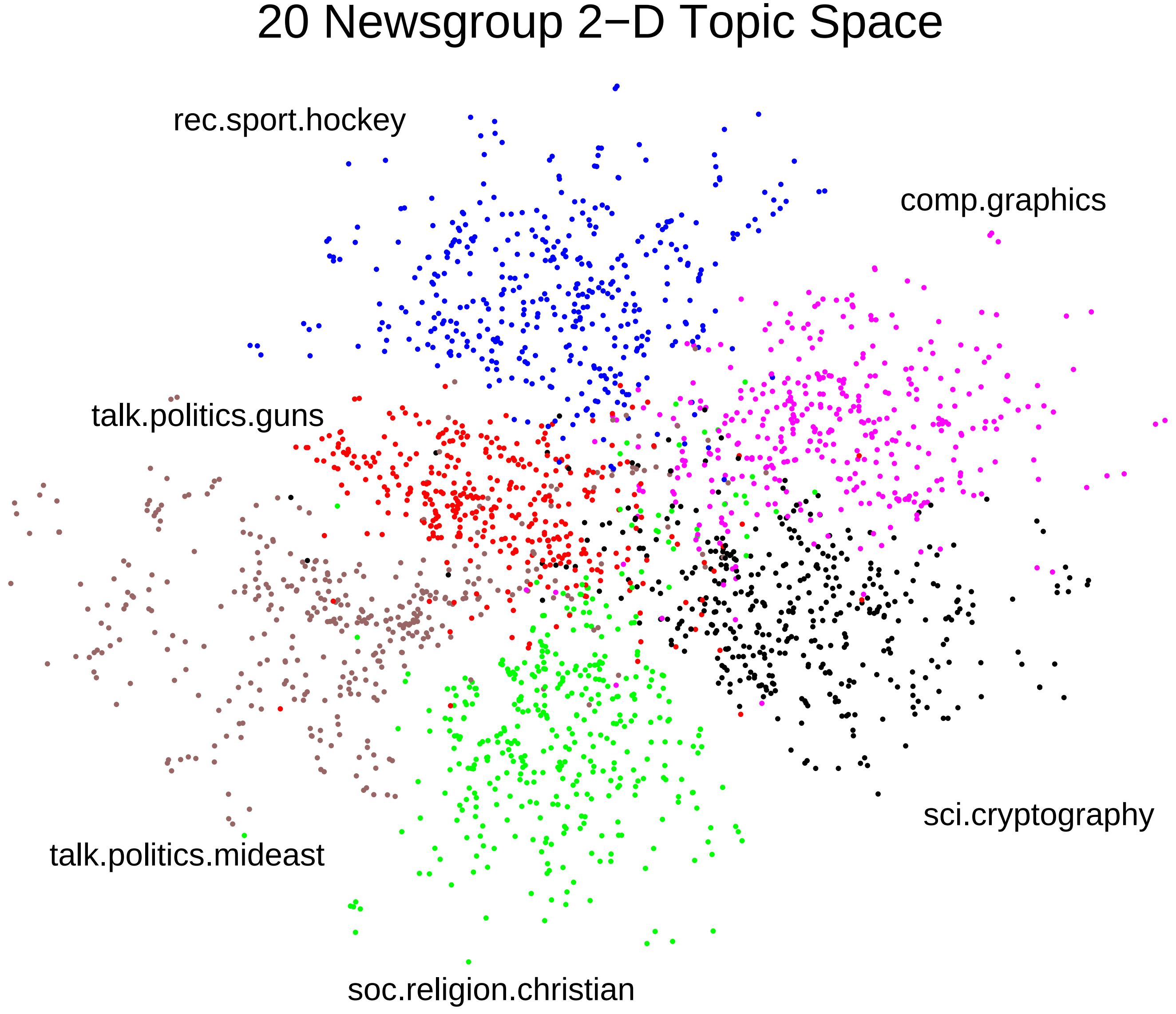
Modeling Documents

Restricted Boltzmann Machines: 2-layer modules.



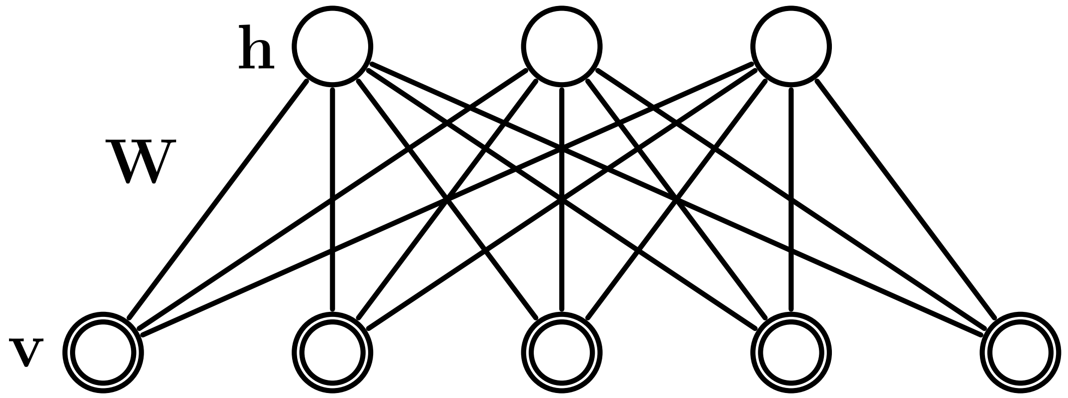
- Visible units are multinomials over word counts.
- Hidden units are topic detectors.

Extracted Latent Topics



Collaborative Filtering

Form of Matrix Factorization.



- Visible units are multinomials over rating values.
- Hidden units are user preference detectors.

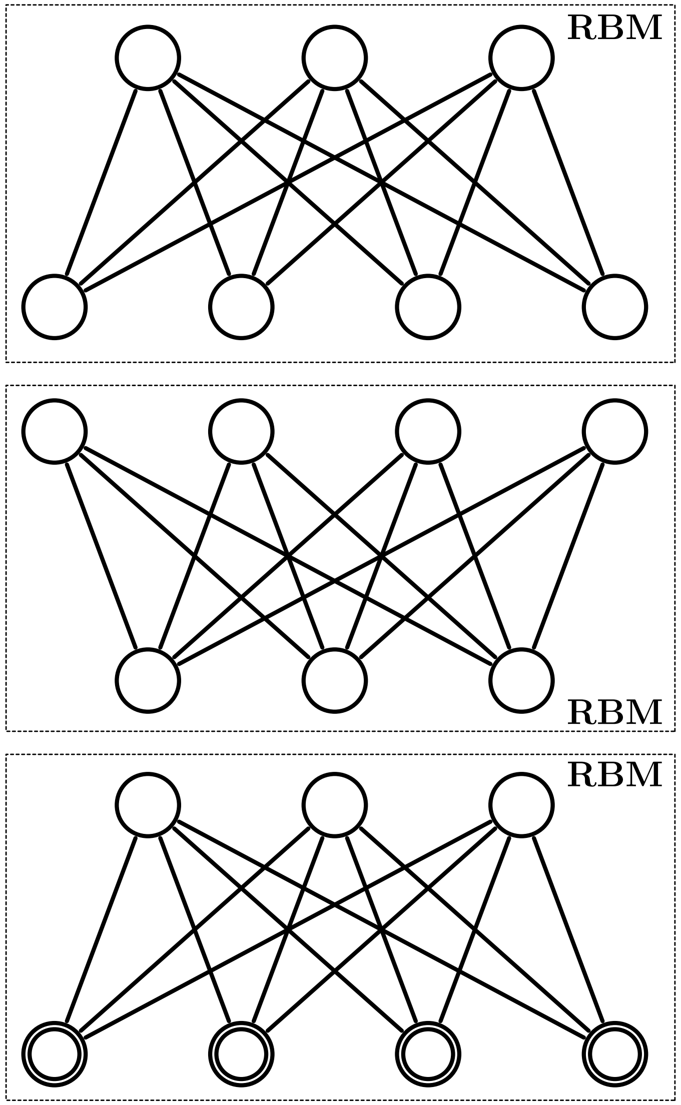
Used in Netflix competition.

Deep Belief Networks (DBN's)

- There are limitations on the types of structure that can be represented efficiently by a single layer of hidden variables.

- We would like to **efficiently** learn multi-layer models using a large supply of high-dimensional highly-structured unlabeled sensory input.

Learning DBN's

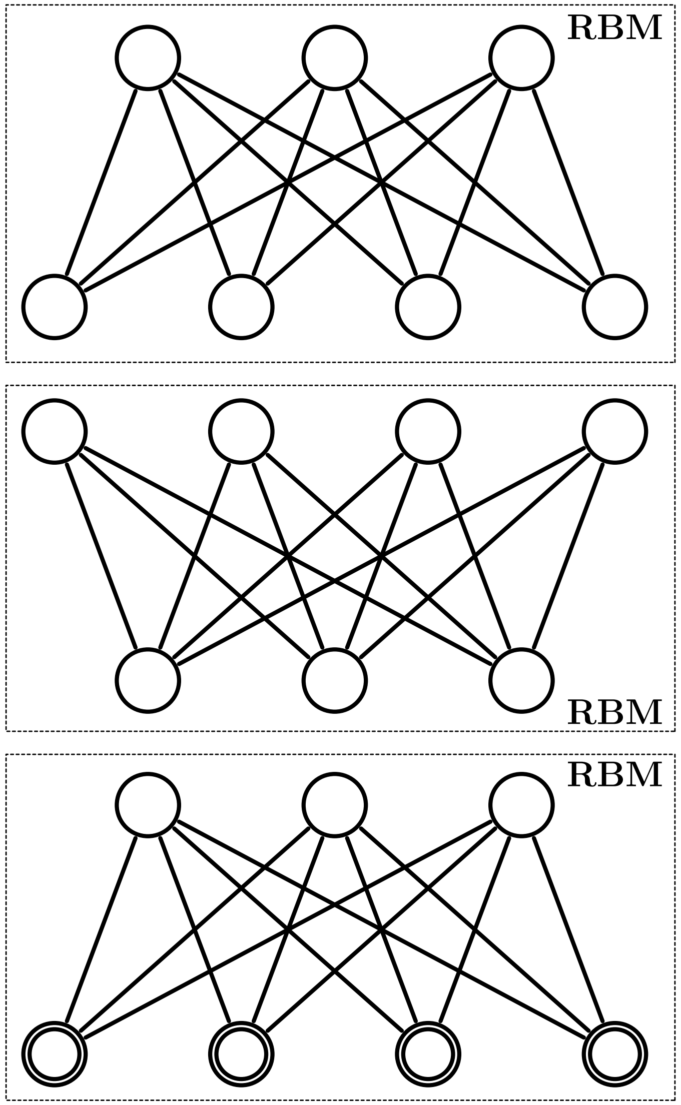


Greedy, layer-by-layer learning:

- Learn and Freeze W^1.
- Sample $\mathbf{h}^1 \sim P(\mathbf{h}^1|\mathbf{v}; W^1)$.
  Treat $\mathbf{h}^1$ as if it were data.
- Learn and Freeze W^2.
- ...

Learn high-level representations.

Learning DBN's

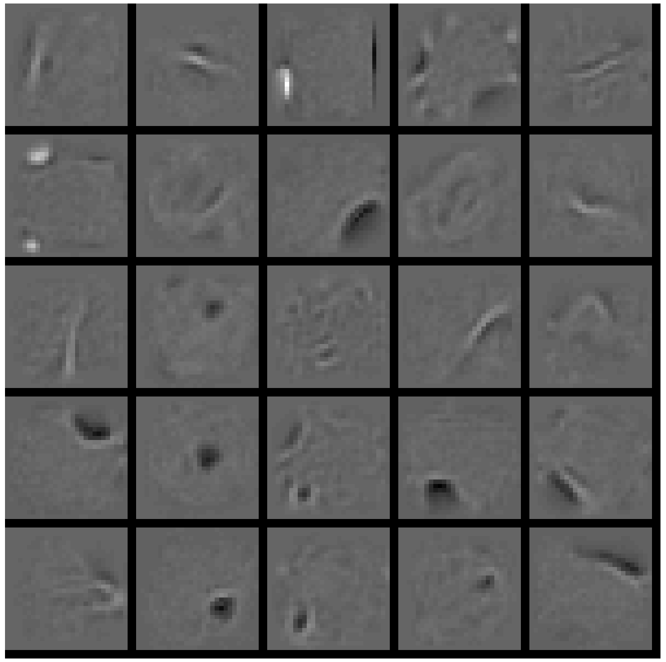


Under certain conditions adding an extra layer always improves a lower bound on the log probability of data.

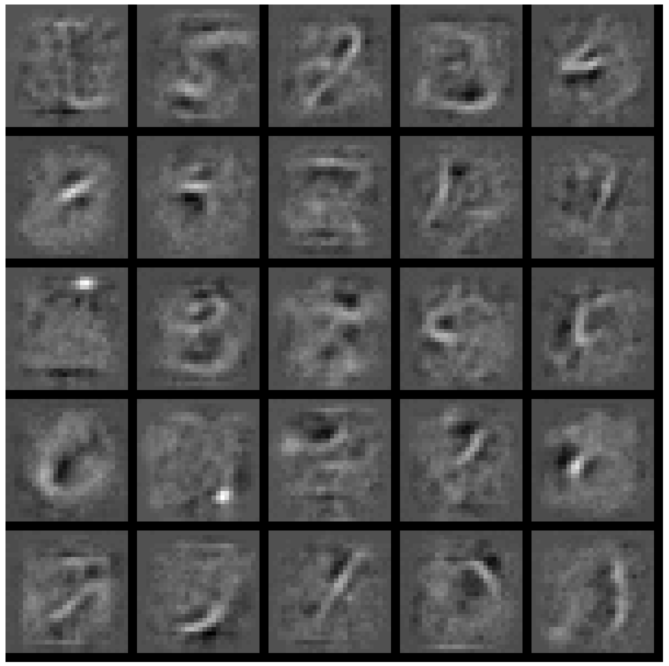
Each layer of features captures high-order correlations between the activities of units in the layer below.

Learning DBN's

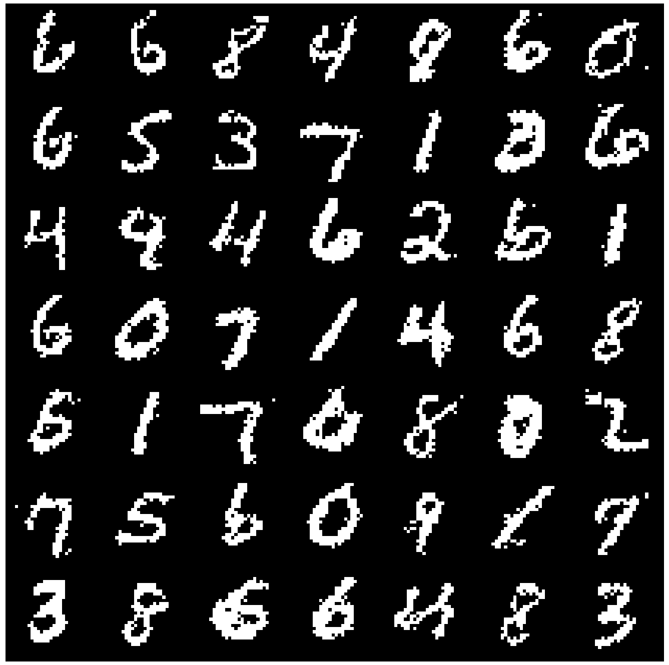
1^{st}-layer features



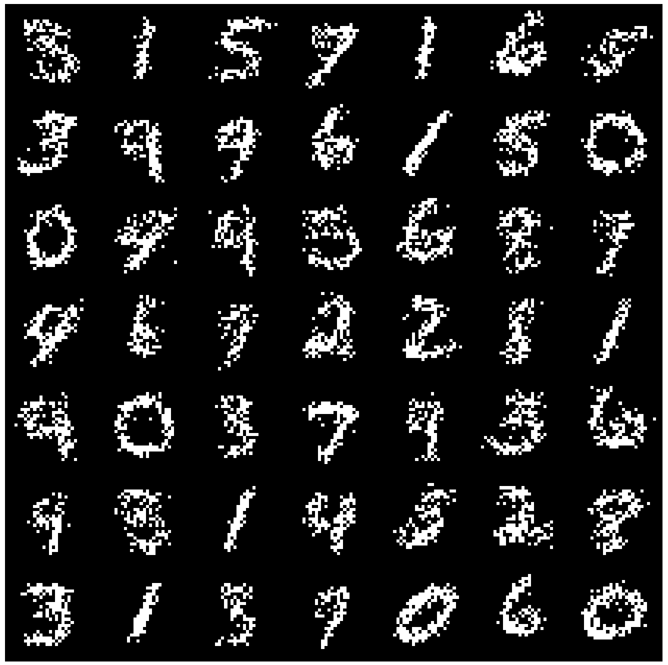
2^{nd}-layer features



Density Estimation



DBN samples



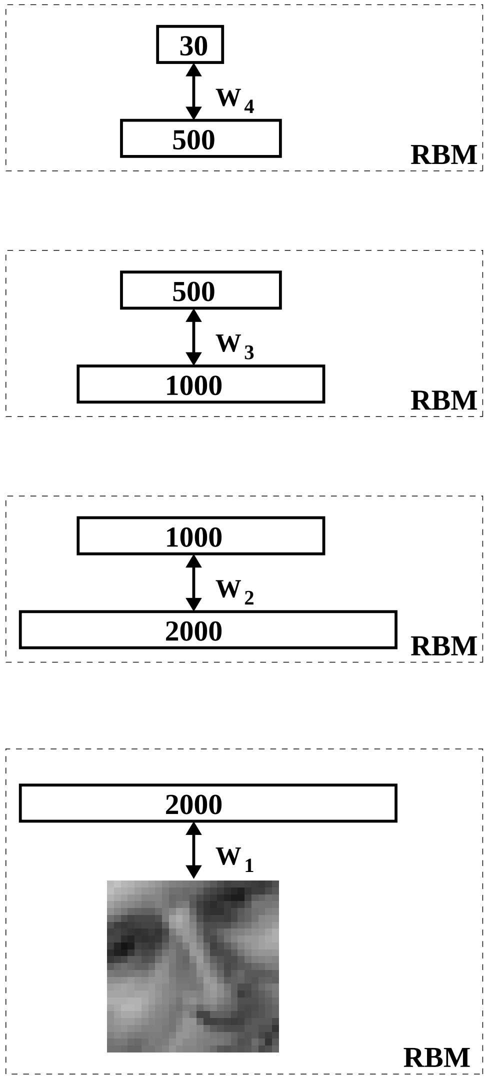
Mixture of Bernoulli's

MoB, test log-prob: -137.64 per digit.

DBN, test log-prob: -85.97 per digit.

Difference of over 50 nats is quite large.

Learning Deep Autoencoders

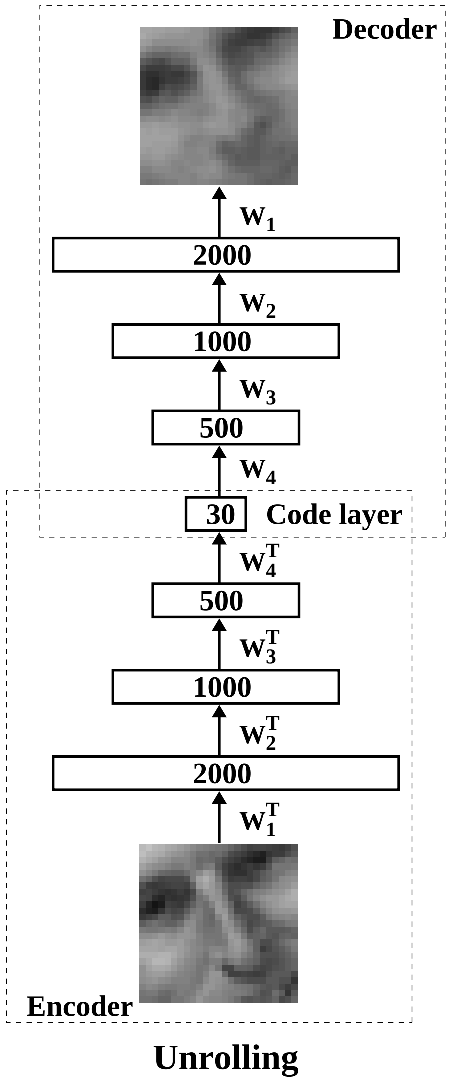


Pretraining consists of learning a stack of RBMs.

Each RBM has 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.

Learning Deep Autoencoders

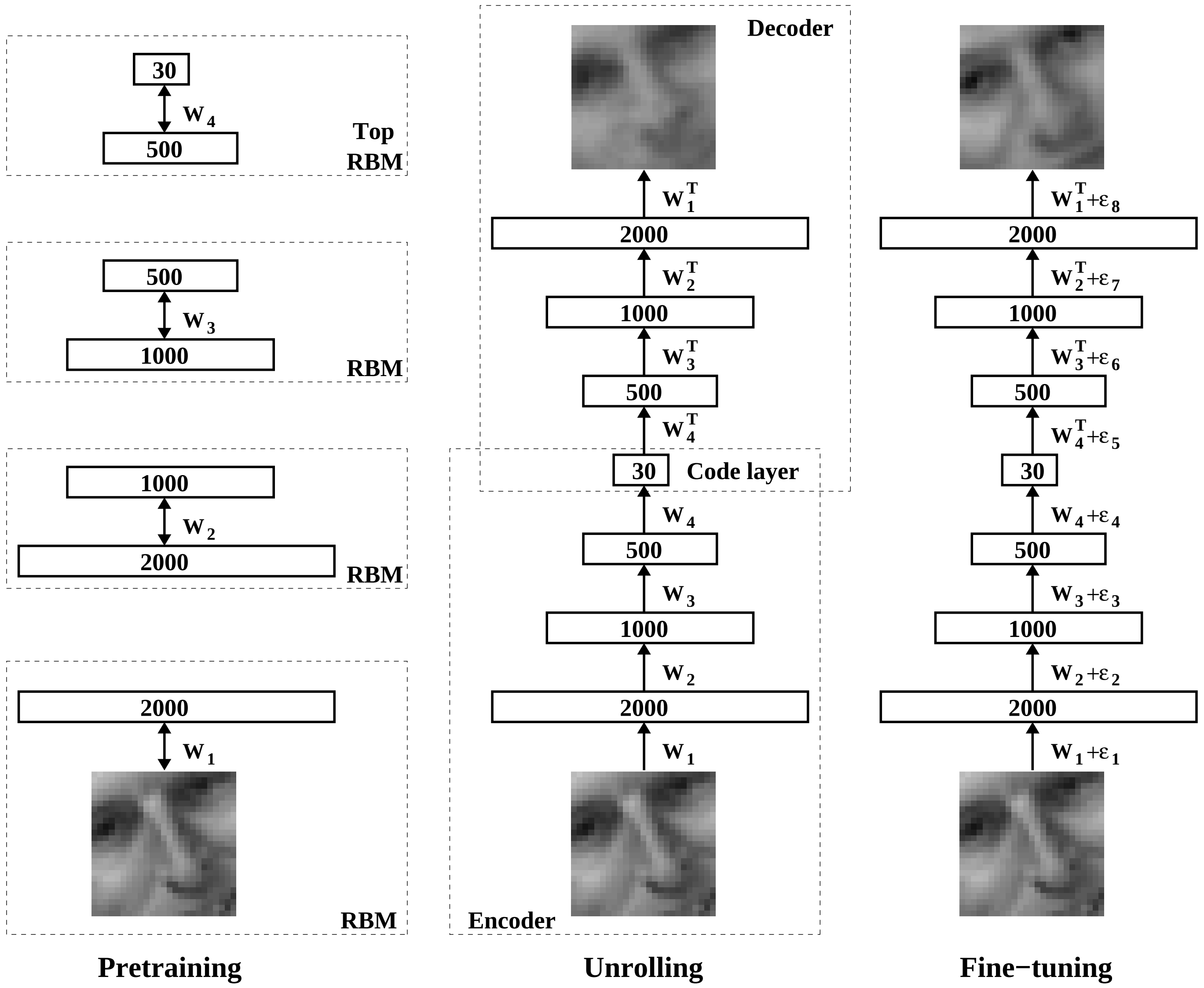


After pretraining multiple layers, the model is unrolled to create a deep autoencoder

Initially encoder and decoder networks use the same weights.

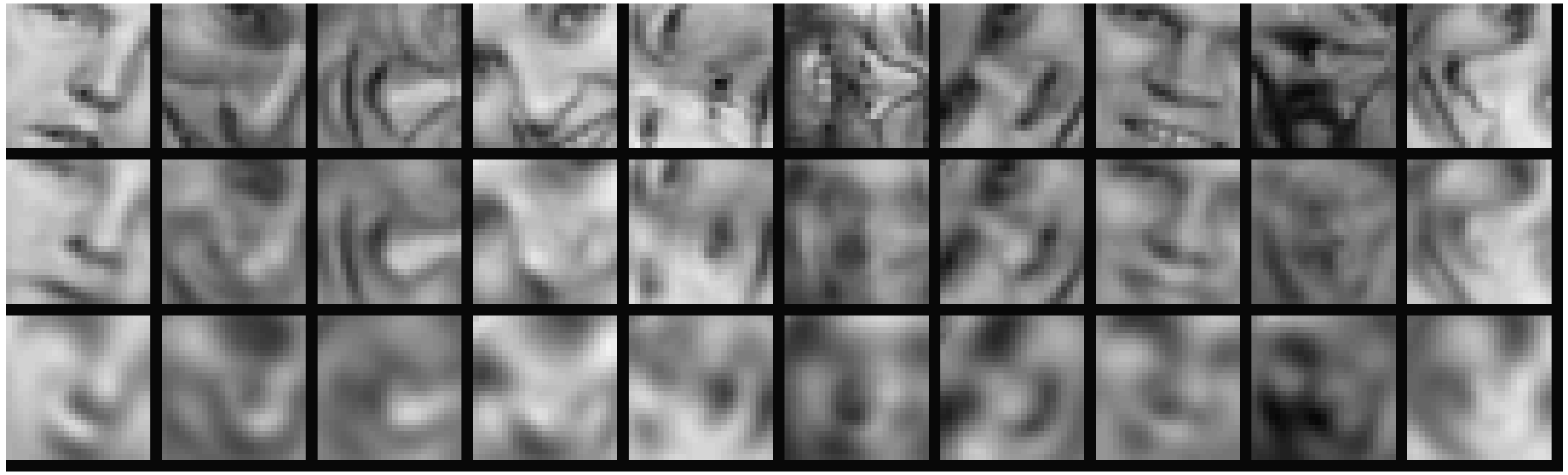
The global fine-tuning uses backpropagation through the whole autoencoder to fine-tune the weights for optimal reconstruction.

Learning Deep Autoencoders



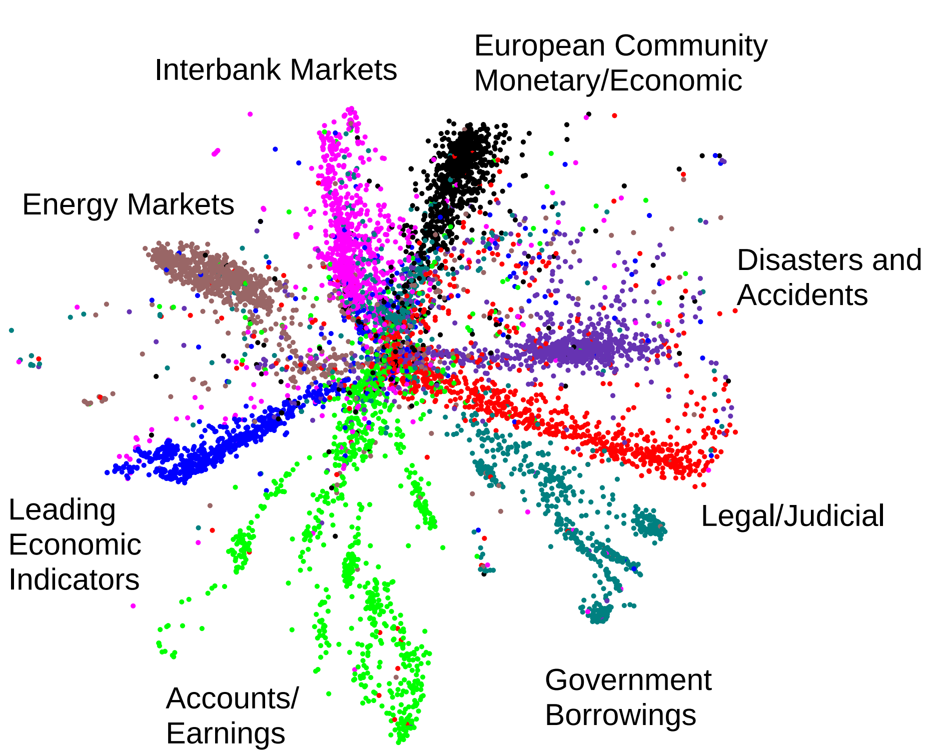
Learning Deep Autoencoders

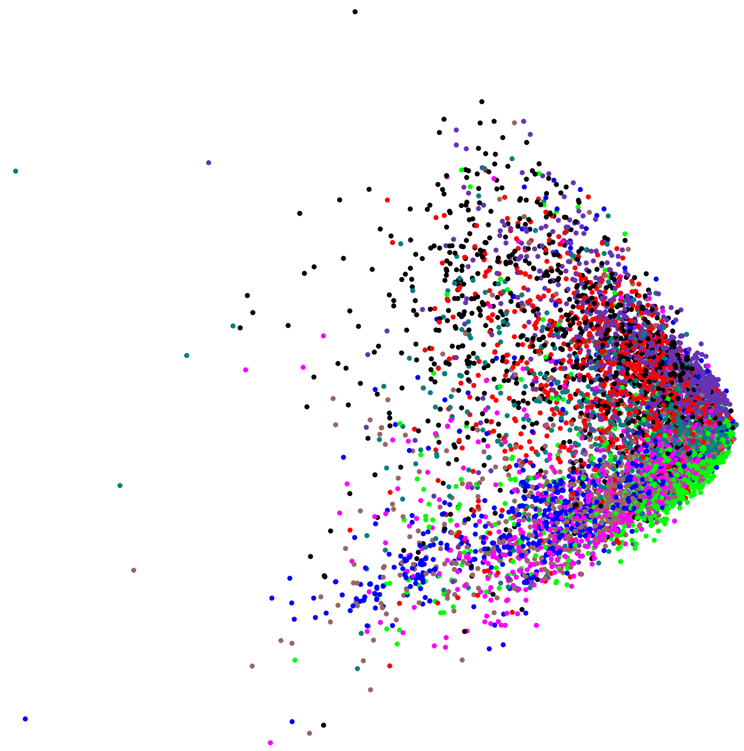
We used a 25×25-2000-1000-500-30 autoencoder to extract 30-D real-valued codes for Olivetti face patches (7 hidden layers is usually hard to train).



Top Random samples from the test dataset; **Middle** reconstructions by the 30-dimensional deep autoencoder; and **Bottom** reconstructions by 30-dimensional PCA.

Dimensionality Reduction





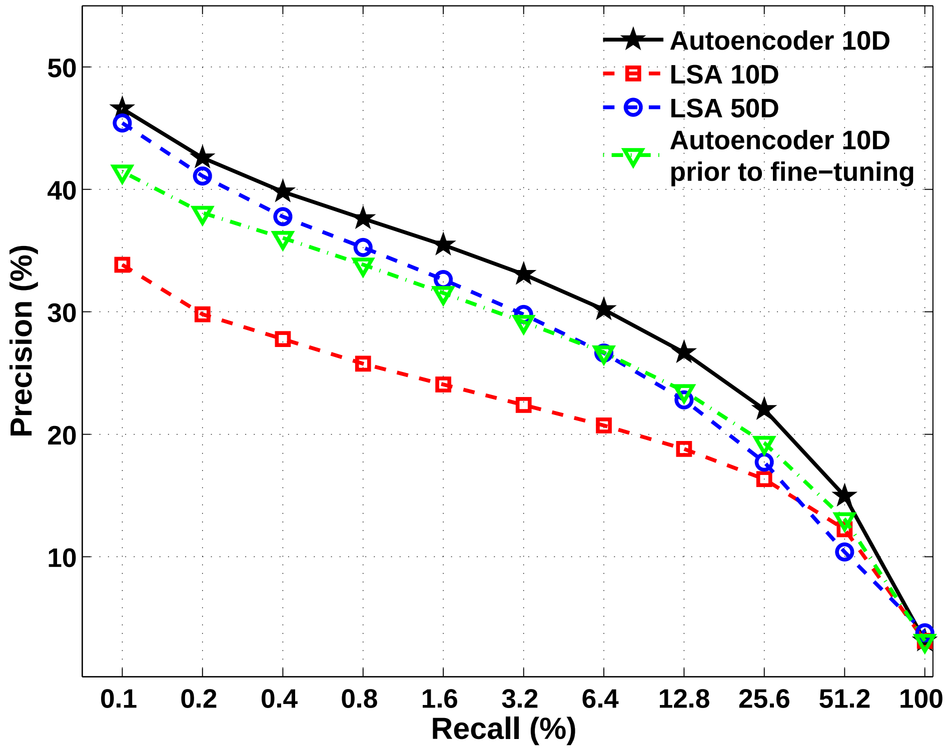
The Reuters Corpus: 804,414 newswire stories.

Simple “bag-of-words” representation.

2000-500-250-125-2 autoencoder.

Document Retrieval

Precision-recall curves when a 10-D query document from the test set is used to retrieve other test set documents, averaged over 402,207 possible queries.



Semi-supervised Learning

- Given a set of $i.i.d$ labeled training samples $\{\mathbf{x}_l, y_l\}$.
- Discriminative models model $p(y_l|\mathbf{x}_l;\theta)$ directly (logistic regression, Gaussian process, SVM's).
- For many applications we have a large supply of high-dimensional unlabeled data $\{\mathbf{x}_u\}$.
- Need to make some assumptions about the input data $\{\mathbf{x}_u\}$.
- Otherwise unlabeled data is of no use.

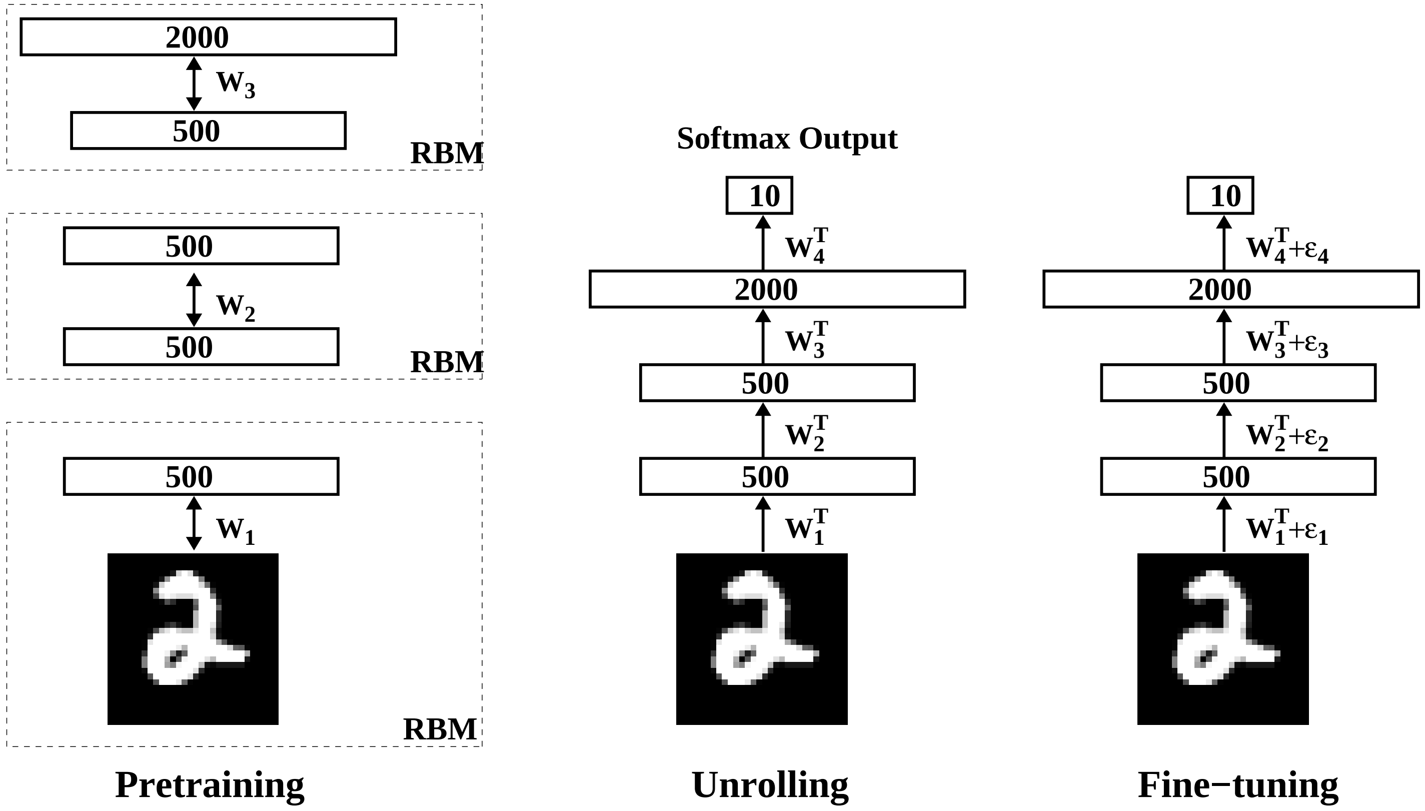
Semi-supervised Learning

Key points of learning deep generative models:

- Learn probabilistic model $p(\mathbf{x}_u;\theta)$.
- Use learned parameters to initialize a discriminative model $p(y_l|\mathbf{x}_l;\theta)$ (neural network).
- Slightly adjust discriminative model for a specific task.

No knowledge of subsequent discriminative task during unsupervised learning. Most of the information in parameters comes from learning a generative model.

Classification Task

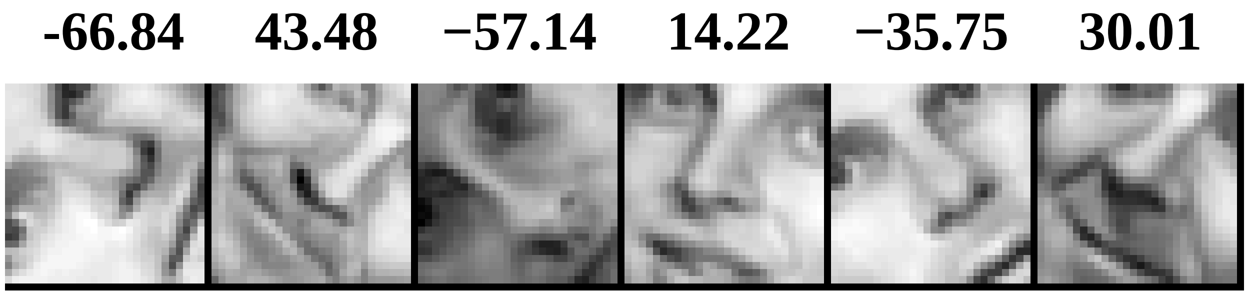


Classification error is 1.2% on MNIST, SVM's get 1.4% and randomly initialized backprop gets 1.6%.

Pretraining helps generalization – most of the information in the weights comes from modeling the input data. (Code available online)

Regression Task

Predicting the orientation of a face patch.



- Labeled Training Data:

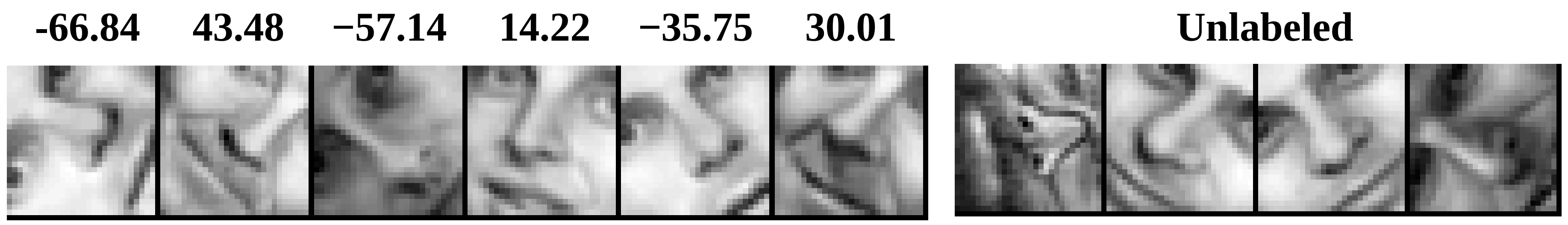
  Input: 1000 labeled training patches Output: orientation

- Labeled Test Data:

  Input: 1000 labeled test patches . Predict: orientation
  of new people.

- Gaussian Processes with exponential kernel achieves a RMSE of 16.36° ($\pm 0.45^\circ$).

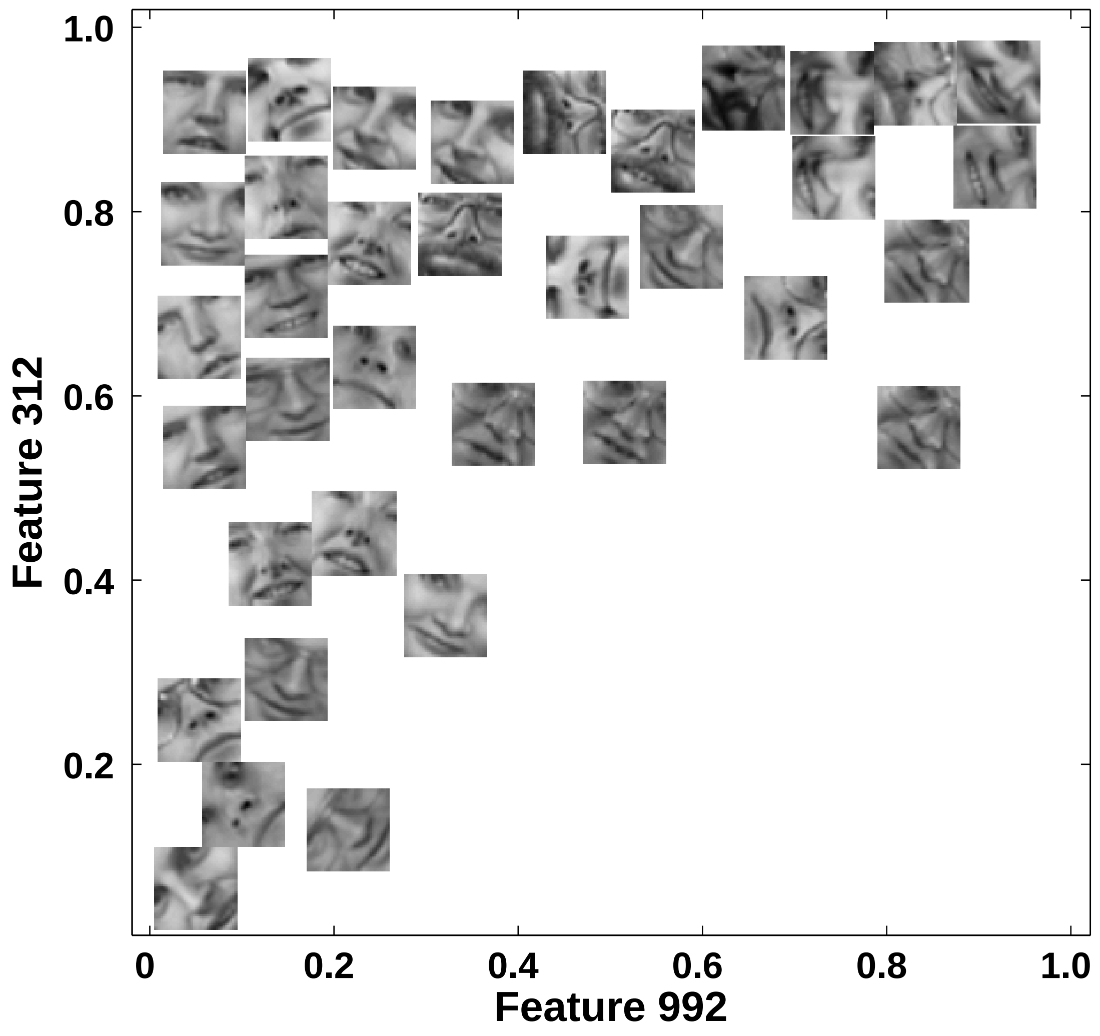
Regression Task



- Additional Unlabeled Training Data: 12000 face patches.
- Pretrain a stack of RBM's: 784-1000-1000.
- **Features were extracted with no idea of the final task.**

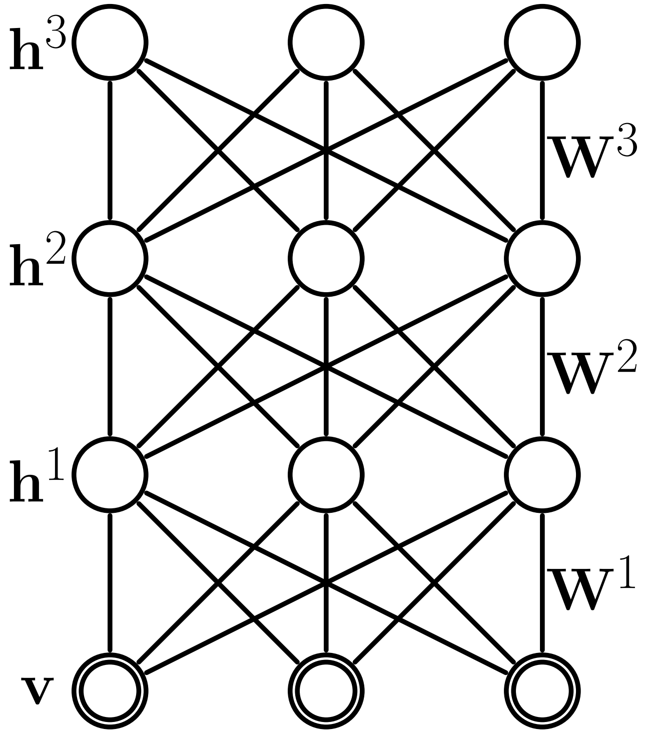
The same GP on the top-level features:	RMSE $11.22°$.
Learn the covariance function of GP:	RMSE $6.42°$.

Extracted Features

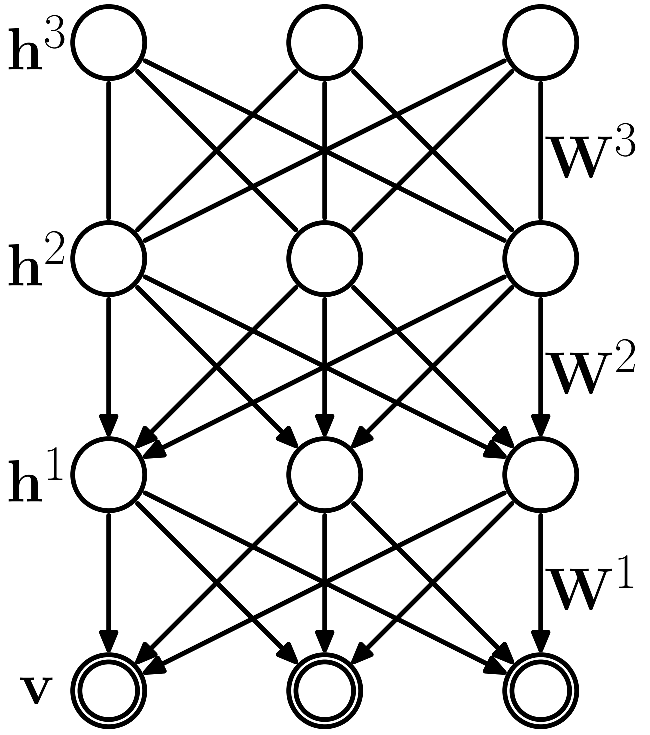


Units 312 and 992 are very responsive to face orientation.

DBM's vs. DBN's



Deep Boltzmann Machine



Deep Belief Network

Deep Boltzmann Machines

- DBM is a type of Markov random field, where all connections between layers are undirected.

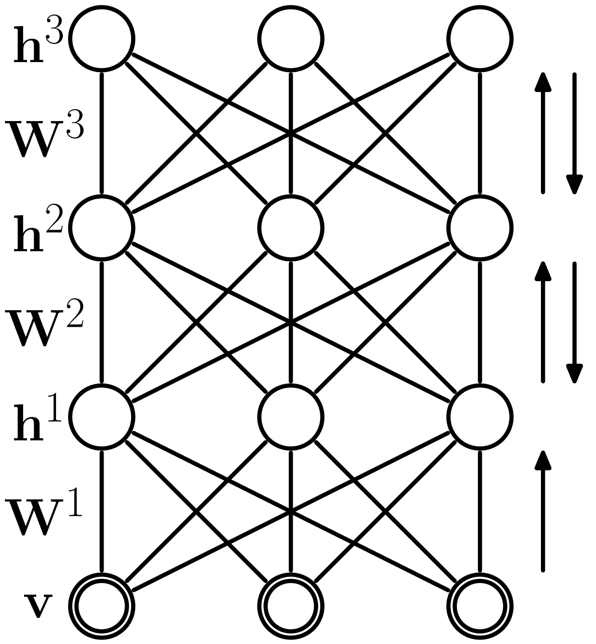
- DBM's have the potential of learning internal representations that become increasingly complex at higher layers, which is a promising way of solving object and speech recognition problems.

- The approximate inference procedure, in addition to a bottom-up pass, can incorporate top-down feedback, allowing DBM's to better propagate uncertainty about ambiguous inputs.

- The entire model can be trained online, processing one example at a time.

Deep Boltzmann Machines

$$P(\mathbf{v}) = \sum_{\mathbf{h}^1,\mathbf{h}^2,\mathbf{h}^3} \frac{1}{\mathcal{Z}} \exp\left[\mathbf{v}^\top W^1 \mathbf{h}^1 + {\mathbf{h}^1}^\top W^2 \mathbf{h}^2 + {\mathbf{h}^2}^\top W^3 \mathbf{h}^3\right].$$



Complex representations.

Fast greedy initialization.

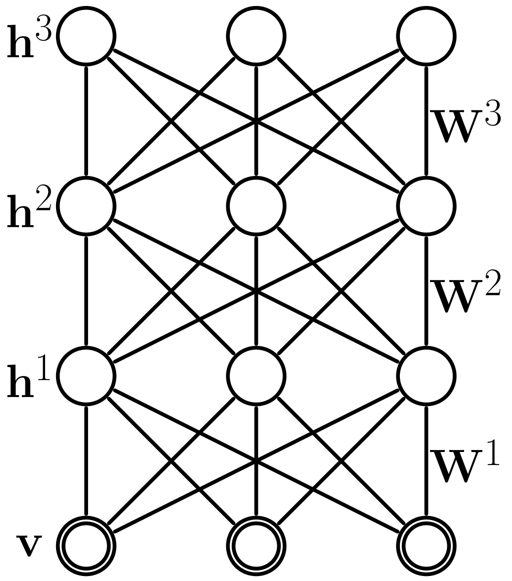
Bottom-up + Top-down.

High-level representations are built from unlabeled inputs.

Labeled data is used to only slightly fine-tune the model.

DBM's: Learning

$$P_\theta(\mathbf{v}) = \sum_{\mathbf{h}} \frac{1}{\mathcal{Z}(\theta)} \exp\left[\mathbf{v}^\top W^1 \mathbf{h}^1 + \mathbf{h}^{1\top} W^2 \mathbf{h}^2 + \mathbf{h}^{2\top} W^3 \mathbf{h}^3\right].$$



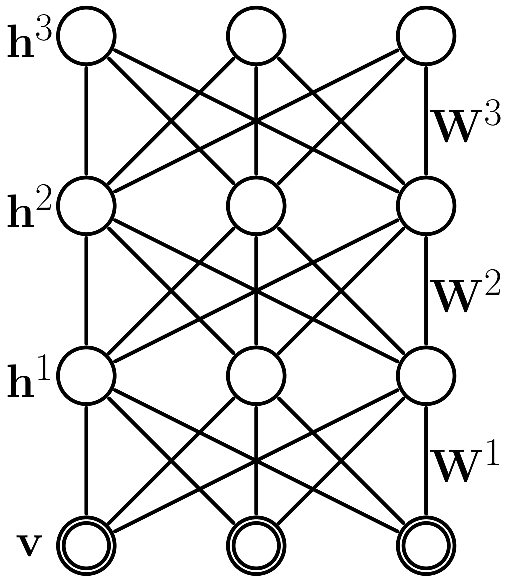
Let $\theta = \{W^1, W^2, W^3\}$.
Maximum Likelihood Learning:

$$\frac{\partial \log P_\theta(\mathbf{v})}{\partial \theta} = \mathrm{E}_{\mathrm{P}_{\mathrm{data}}}[\mathbf{v}\mathbf{h}^\top] - \mathrm{E}_{\mathrm{P}_\theta}[\mathbf{v}\mathbf{h}^\top].$$

$$P_{\mathrm{data}}(\mathbf{h}, \mathbf{v}) = P_\theta(\mathbf{h}|\mathbf{v}) P_{\mathrm{data}}(\mathbf{v}).$$

DBM's: Learning

$$P_\theta(\mathbf{v}) = \sum_{\mathbf{h}} \frac{1}{\mathcal{Z}(\theta)} \exp\left[\mathbf{v}^\top W^1 \mathbf{h}^1 + {\mathbf{h}^1}^\top W^2 \mathbf{h}^2 + {\mathbf{h}^2}^\top W^3 \mathbf{h}^3\right].$$



Let $\theta = \{W^1, W^2, W^3\}$.
Maximum Likelihood Learning:

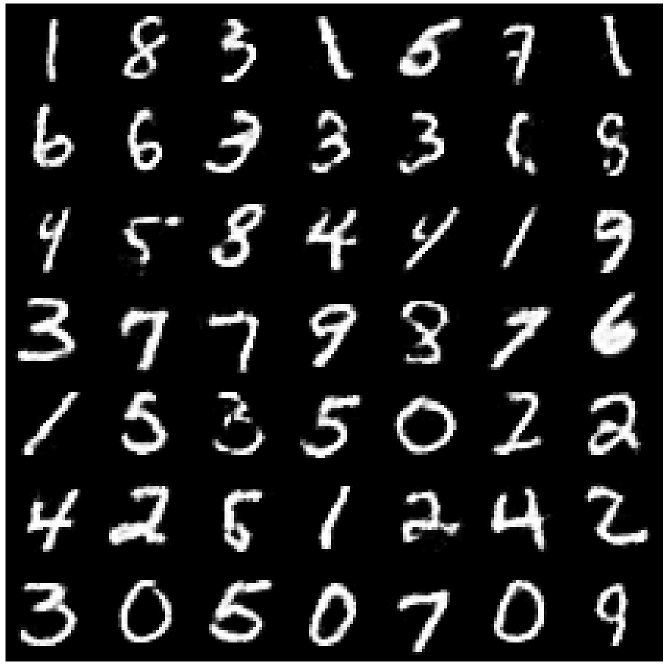
$$\frac{\partial \log P_\theta(\mathbf{v})}{\partial \theta} = \mathrm{E}_{\mathrm{P}_{\mathrm{data}}}[\mathbf{v}\mathbf{h}^\top] - \mathrm{E}_{\mathrm{P}_\theta}[\mathbf{v}\mathbf{h}^\top].$$

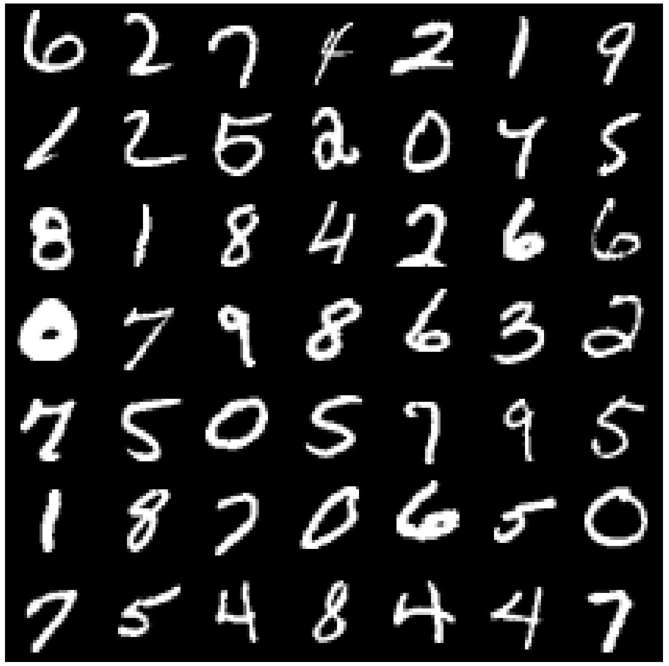
$$P_{\mathrm{data}}(\mathbf{h}, \mathbf{v}) = P_\theta(\mathbf{h}|\mathbf{v}) P_{\mathrm{data}}(\mathbf{v}).$$

Key Idea: (Variation inference + MCMC will be covered later.)

- Variational Inference: Approximate $\mathrm{E}_{\mathrm{P}_{\mathrm{data}}}[\cdot]$.
- MCMC: Approximate $\mathrm{E}_{\mathrm{P}_\theta}[\cdot]$.

Good Generative Model?

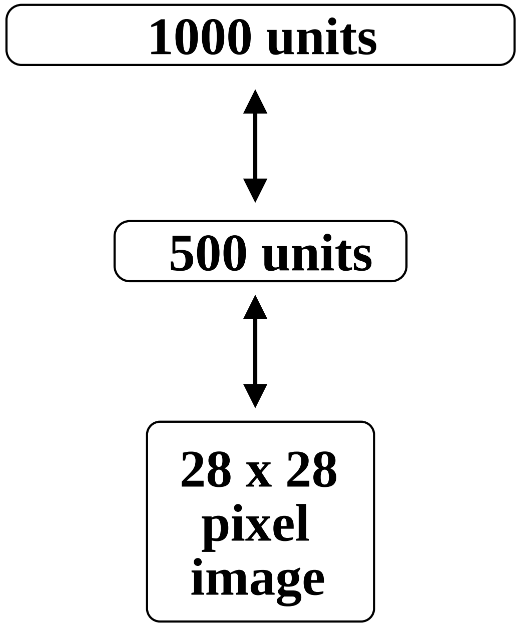




500 hidden and 784 visible units (820,000 parameters).

Samples were generated by running the Gibbs sampler for 100,000 steps.

MNIST: 2-layer BM



0.9 million parameters,
60,000 training and 10,000 test examples.

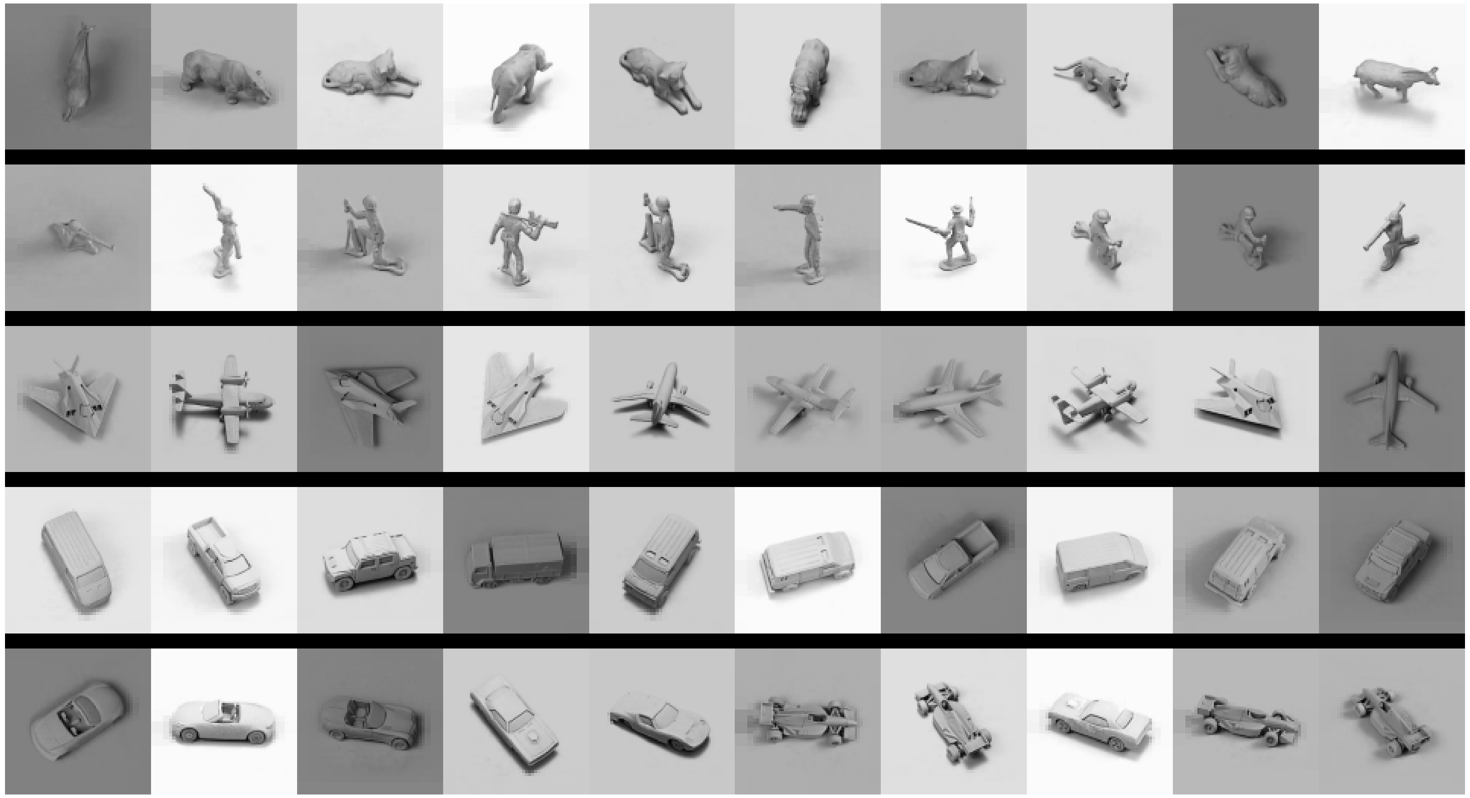
Discriminative fine-tuning: test error of 0.95%.
DBN's get 1.2%, SVM's get 1.4%, backprop gets 1.6%.

Semi-Supervised Learning

Classification error rates on MNIST test set when only a small fraction of labeled data is available.

	Two-Layer DBM	Nonlinear NCA	Autoencoder	KNN
1% (600)	4.82%	8.81%	9.62%	13.74%
5% (3000)	2.72%	3.24%	5.18%	7.19%
10% (6000)	2.46%	2.58%	4.46%	5.87%
100% (60K)	0.95%	1.00%	2.41%	3.09%

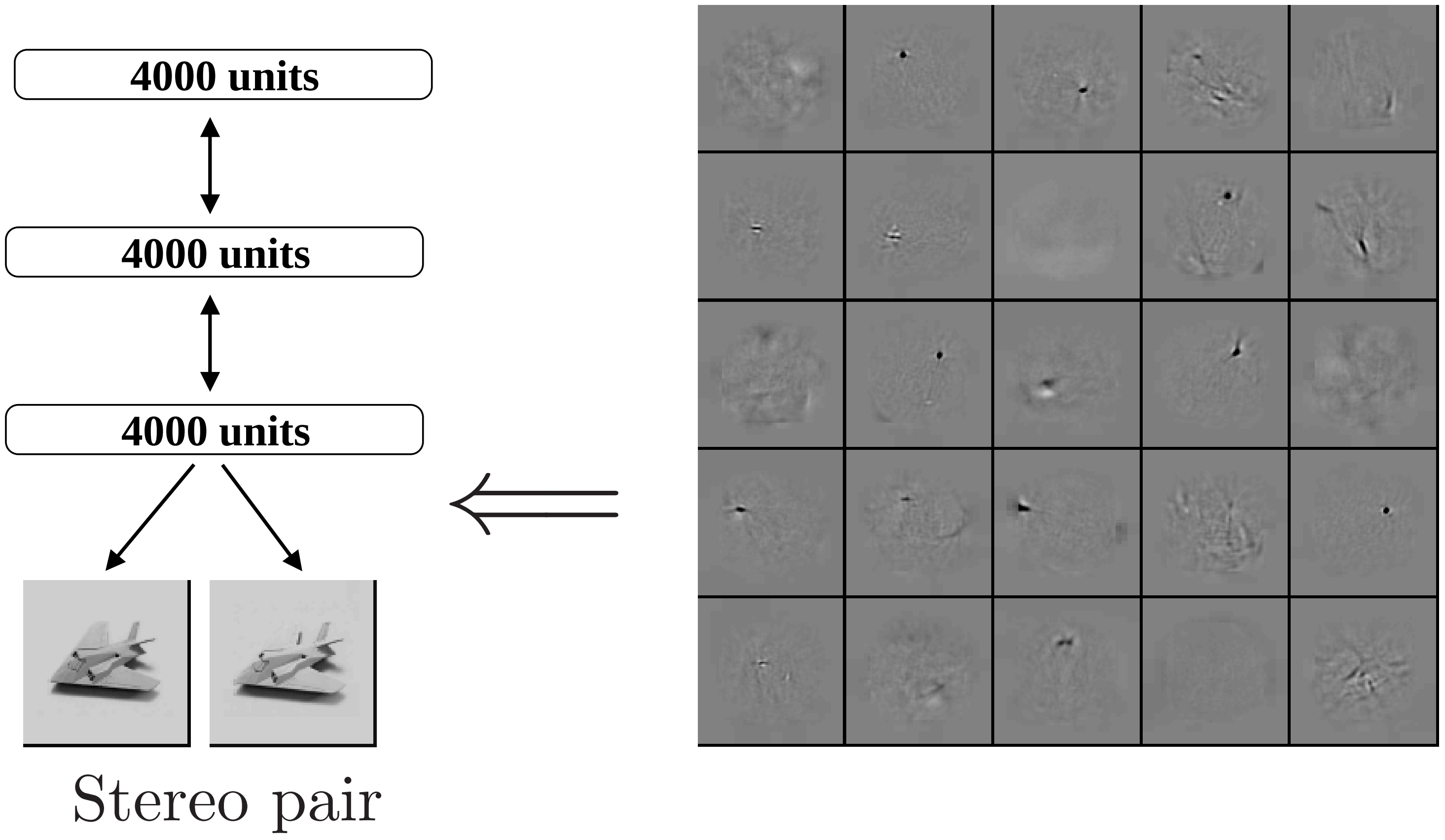
NORB data



5 object categories, 5 different objects within each category, 6 lightning conditions, 9 elevations, 18 azimuth.

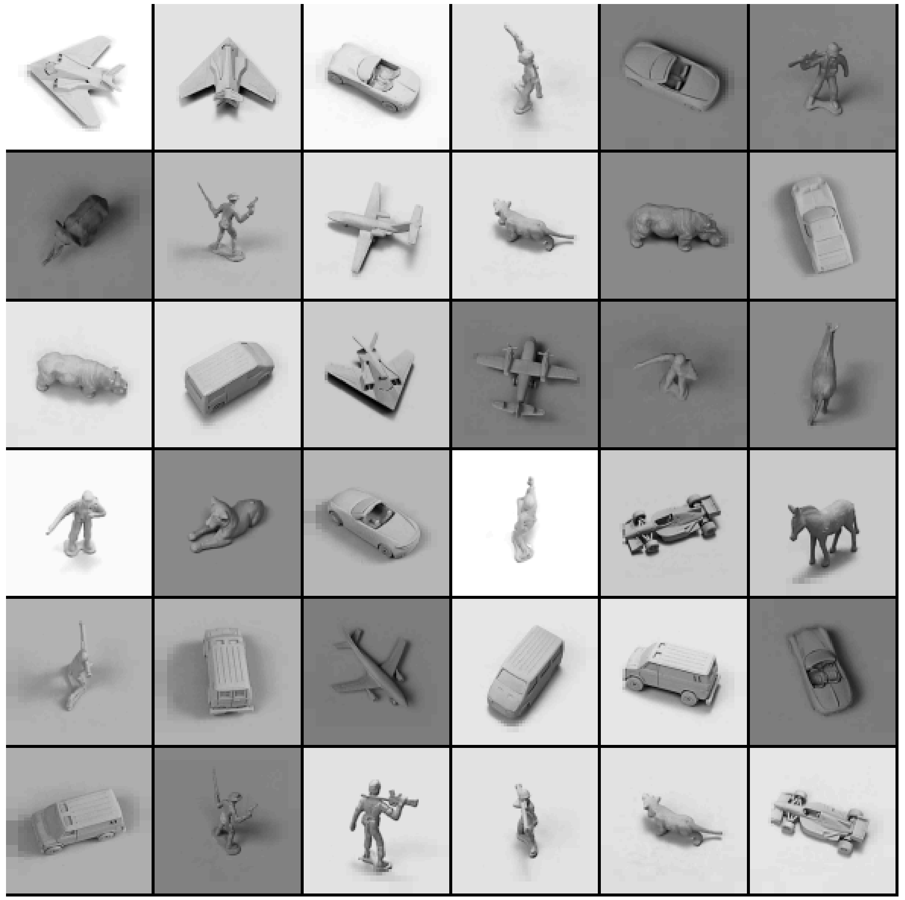
24,300 training and 24,300 test cases.

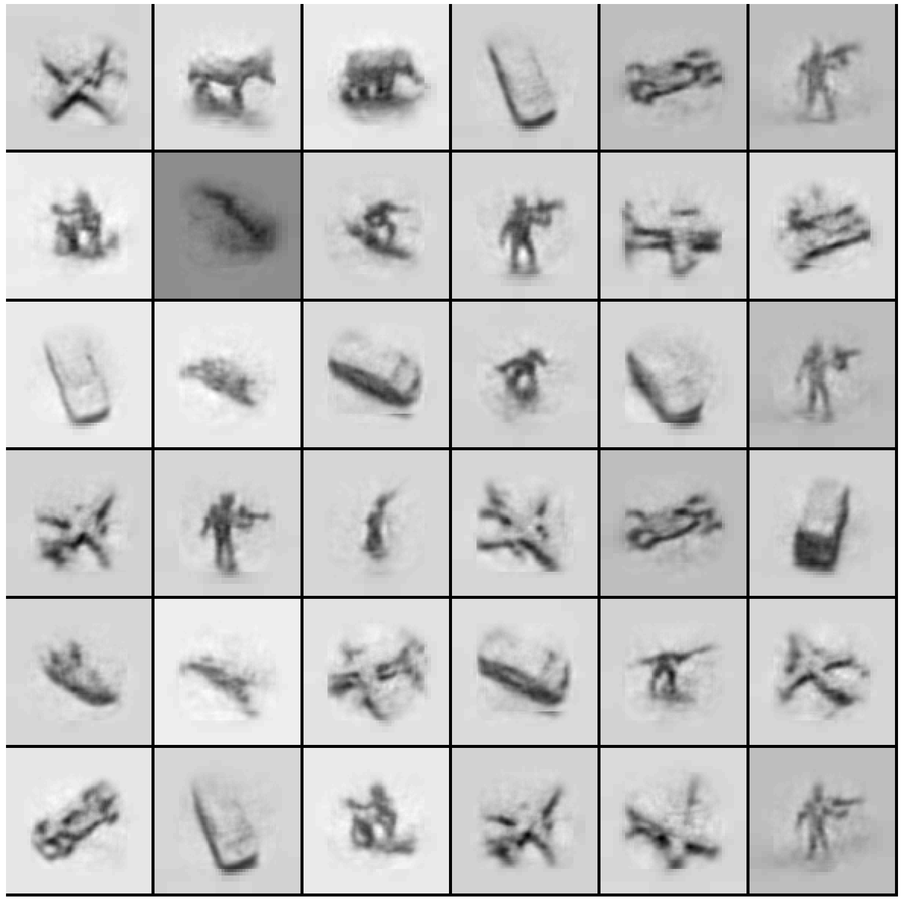
Deep Boltzmann Machines



About 68 million parameters.

Model Samples





Discriminative fine-tuning: test error of 7.2%.

SVM's get 11.6%, logistic regression gets 22.5%.

NORB Train vs. Test

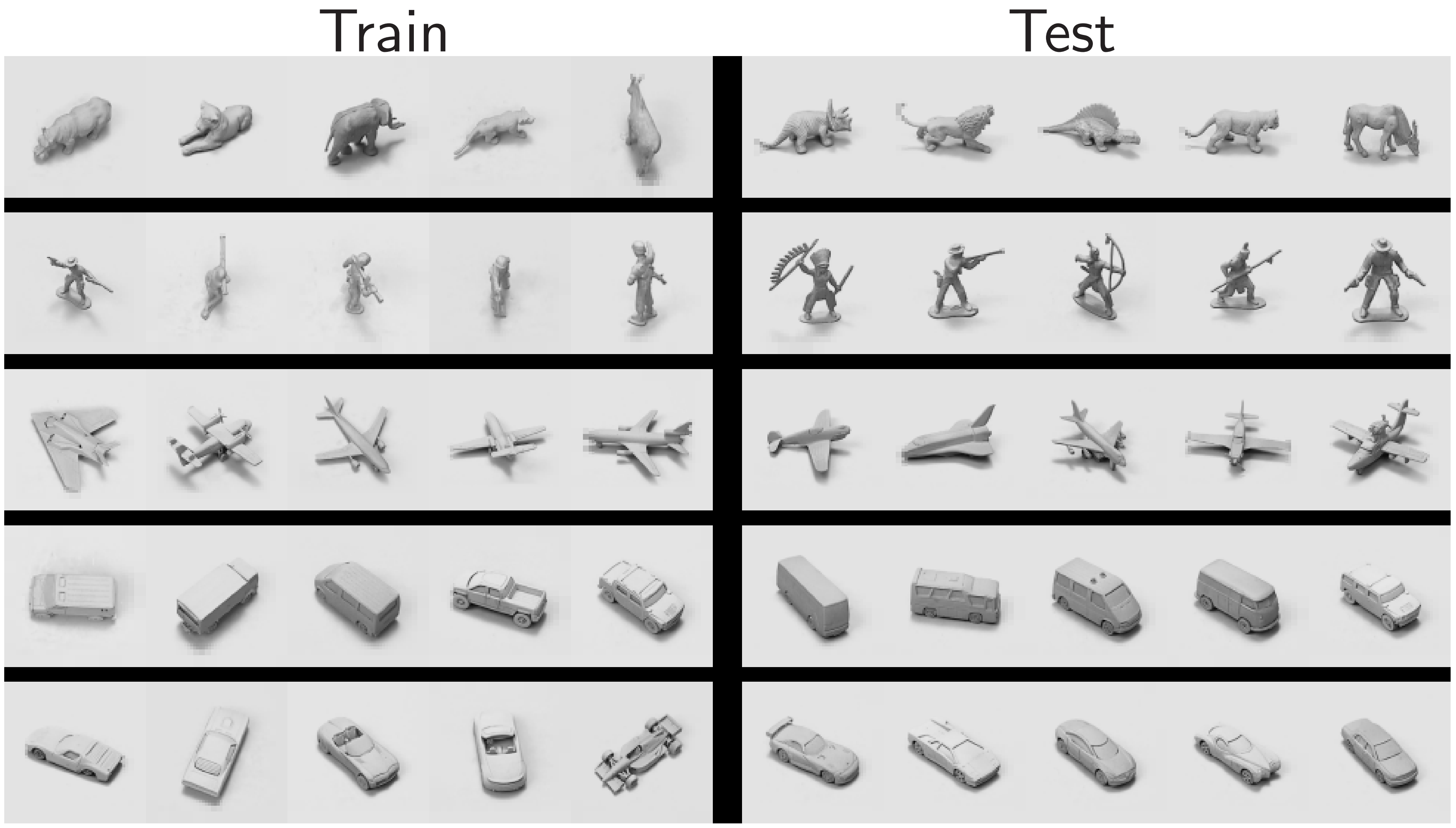
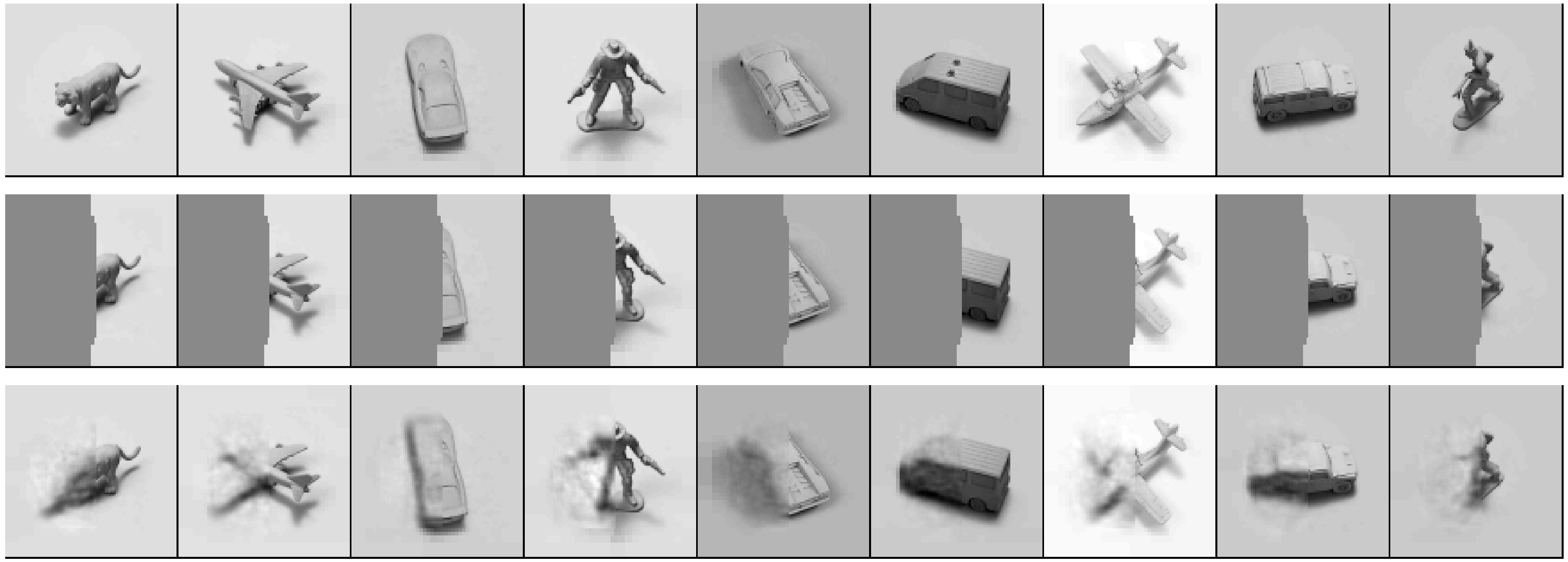
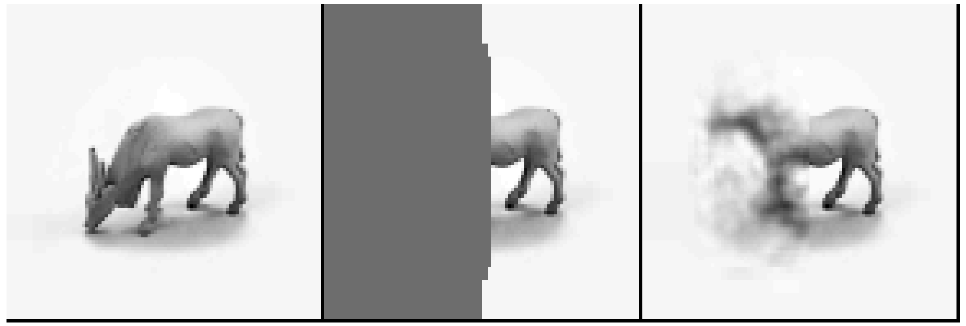


Image Completion





Extensions

- **Better Learning Algorithms for Deep Models**:
  - Better MCMC algorithms within stochastic approximation that can explore highly mutimodal distributions: Tempered Transitions, Simulated/Parallel Tempering.
  - Faster and more accurate variational inference: loopy belief propagation, generalized BP, power expectation propagation.

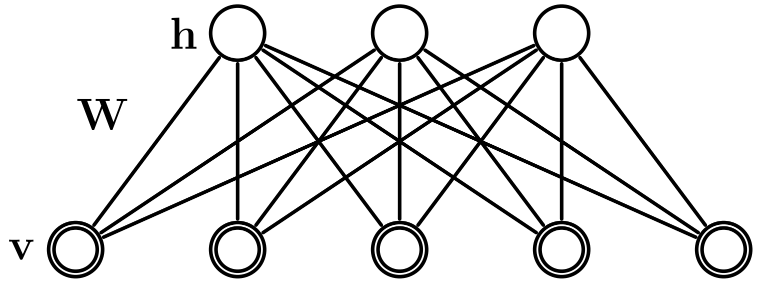
- **Kernel Learning**: Deep models can be used to define kernel or similarity function for many discriminative methods, including logistic regression, SVM's, kernel regression, Gaussian processes.

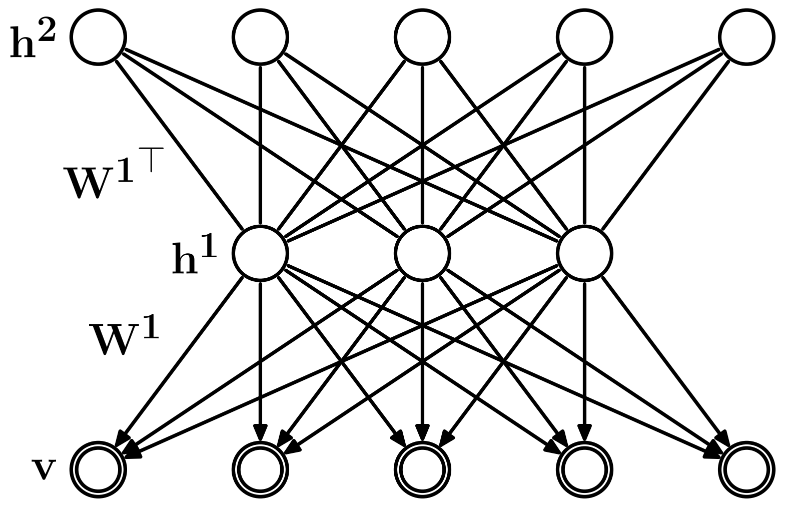
Extensions

- **Large Scale Object Recognition and Information Retrieval:** Deep models have the potential to extract features that become progressively more complex, which is a promising way to solve object recognition and information retrieval problems.
- **Discovering a hierarchical structure of object categories**: Learned high-level representations can be used as the input to more structured hierarchical Bayesian models, which would allow us to better incorporate our prior knowledge.
- **Extracting Structure from Time**: Modeling complex nonlinear dynamics of high-dimensional time series data, such as video sequences

Thank you.

Learning DBN's

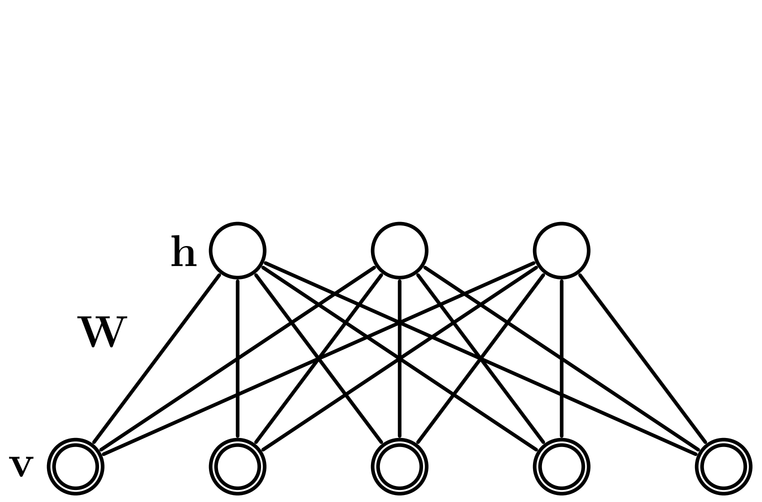


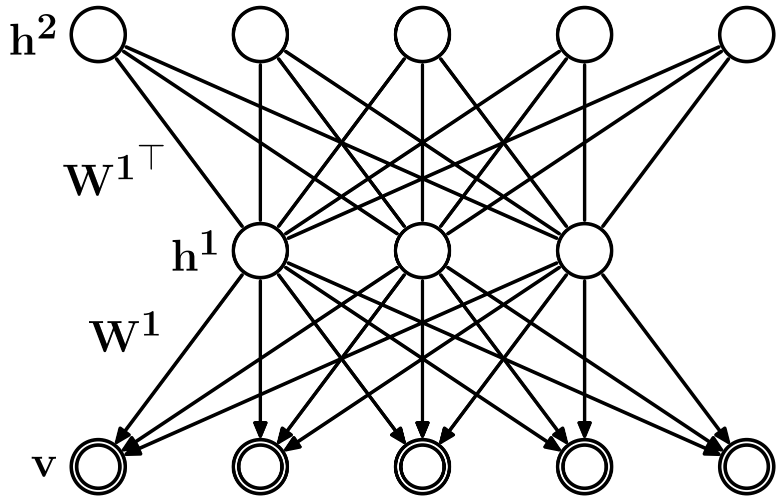


With tied weights, the distribution over $\mathbf{v}$ defined by an RBM and a DBN are the same.

$$\sum_{\mathbf{h}^1} P_{\mathrm{R}BM}(\mathbf{v}, \mathbf{h}^1; W^1) = \sum_{\mathbf{h}^1, \mathbf{h}^2} P_{\mathrm{D}BN}(\mathbf{v}, \mathbf{h}^1, \mathbf{h}^2; W^1)$$

Learning DBN's



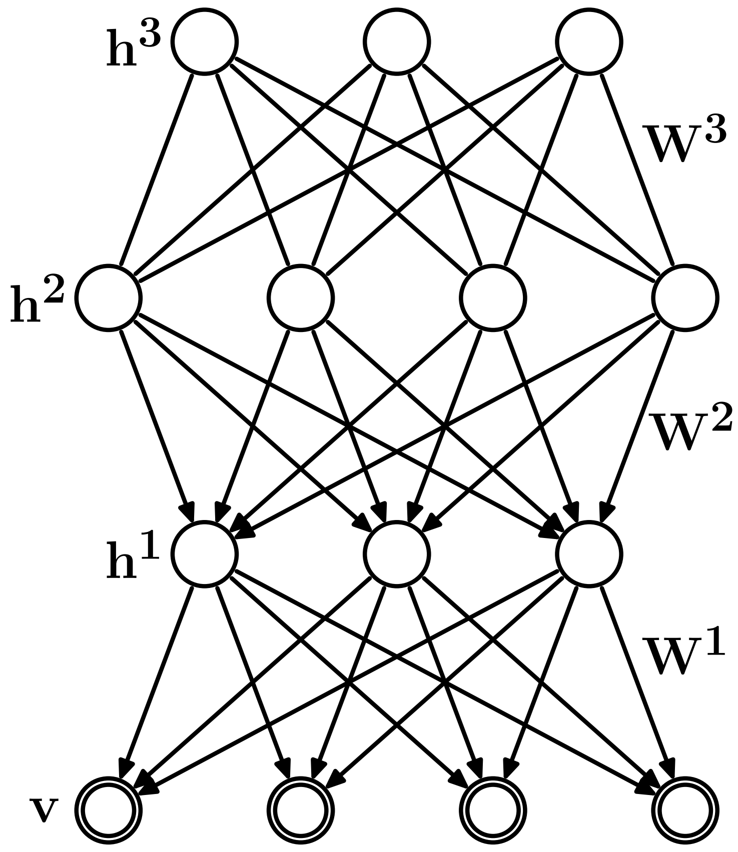


$$\begin{aligned}\log P(\mathbf{v};\theta) &\geq \sum_{\mathbf{h}^1} Q(\mathbf{h}^1|\mathbf{v})\Big[\log P(\mathbf{v},\mathbf{h}^1;\theta)\Big] + \mathcal{H}(Q(\mathbf{h}^1|\mathbf{v}))\\ &= \sum_{\mathbf{h}^1} Q(\mathbf{h}^1|\mathbf{v})\Big[\log P(\mathbf{h}^1;W^2)\ + \log P(\mathbf{v}|\mathbf{h}^1;W^1)\Big] + \mathcal{H}(Q(\mathbf{h}^1|\mathbf{v}))\end{aligned}$$

Freeze $Q(\mathbf{h}^1|\mathbf{v})$, and $P(\mathbf{v}|\mathbf{h}^1;W^1)$ and maximize the lower bound w.r.t. $P(\mathbf{h}^1;W^2)$.

Deep Belief Nets

$$P(\mathbf{v},\mathbf{h};\theta) = P(\mathbf{v}|\mathbf{h}^1)P(\mathbf{h}^1|\mathbf{h}^2)P(\mathbf{h}^2,\mathbf{h}^3).$$



$$\mathbf{h} = \{\mathbf{h}^1,\mathbf{h}^2,\mathbf{h}^3\},$$

$$P(\mathbf{v};\theta) = \sum_{\mathbf{h}} P(\mathbf{v},\mathbf{h};\theta).$$

Set of visible $\mathbf{v}$ and hidden $\mathbf{h}$ binary stochastic units.

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

Inference and maximum likelihood learning are hard.