A Fast Learning Algorithm for Deep Belief Nets

Geoffrey E. Hinton, Simon Osindero
Department of Computer Science
University of Toronto, Toronto, Canada

Yee-Whye Teh
Department of Computer Science
National University of Singapore, Singapore

Present by: Seyed Ali Cheraghi

WICHITA STATE UNIVERSITY

WiNES
Wireless, Networking, and Energy Systems Research Laboratory

Features

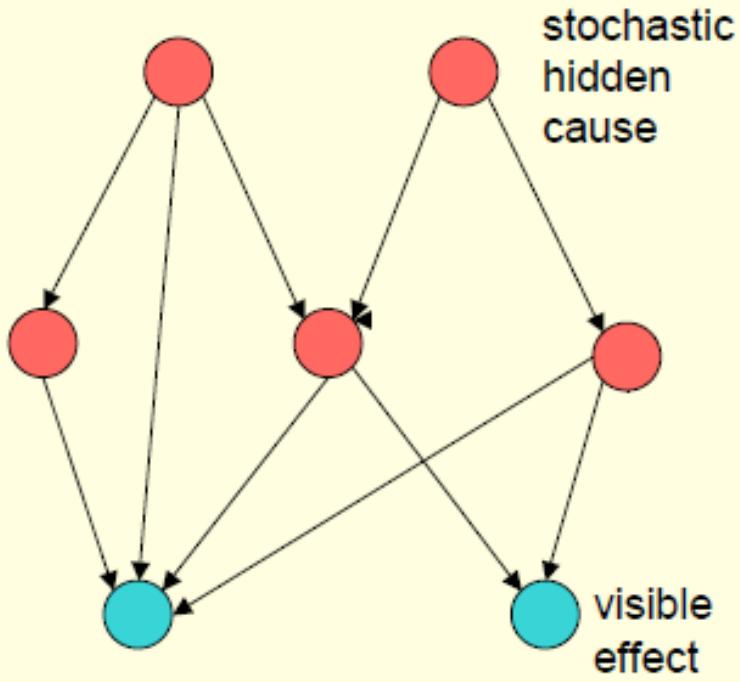
- Fast, greedy learning algorithm that can find a fairly good set of parameters quickly, even in deep networks with millions of parameters and many hidden layers.
- The learning algorithm is unsupervised but can be applied to labeled data by learning a model that generates both the label and the data.
- There is a fine-tuning algorithm that learns an excellent generative model that outperforms discriminative methods on the MNIST database of hand-written digits.
- The generative model makes it easy to interpret the distributed representations in the deep hidden layers.

Features cont.

- The inference required for forming a percept is both fast and accurate.
- The learning algorithm is local. Adjustments to a synapse strength depend on only the states of the presynaptic and postsynaptic neuron.
- The communication is simple. Neurons need only to communicate their stochastic binary states.

Belief Nets

- A belief net is a directed acyclic graph composed of stochastic variables.
- We get to 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.

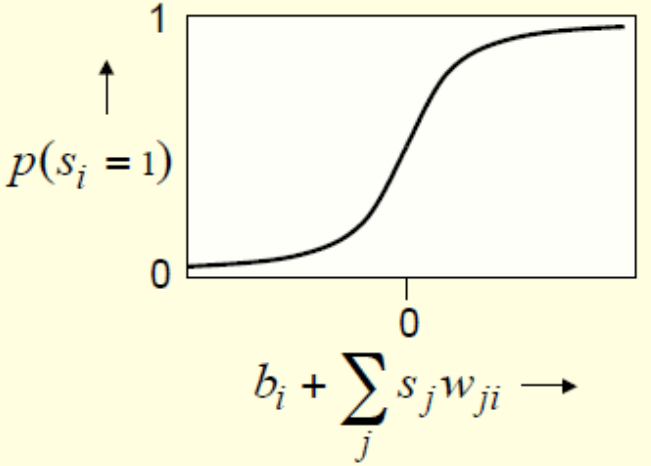


We will use nets composed of layers of stochastic binary variables with weighted connections.

Stochastic binary units

- These have a state of 1 or 0.
- The probability of turning on is determined by the weighted input from other units (plus a bias)
- To generate data: $P(s_i = 1)$ = logistic function(s_j and w_{ij})

$$P(s_i = 1) = \frac{1}{1 + \exp(-b_i - \sum_j s_j w_{ij})}$$



bias of unit i

states of its immediate ancestors

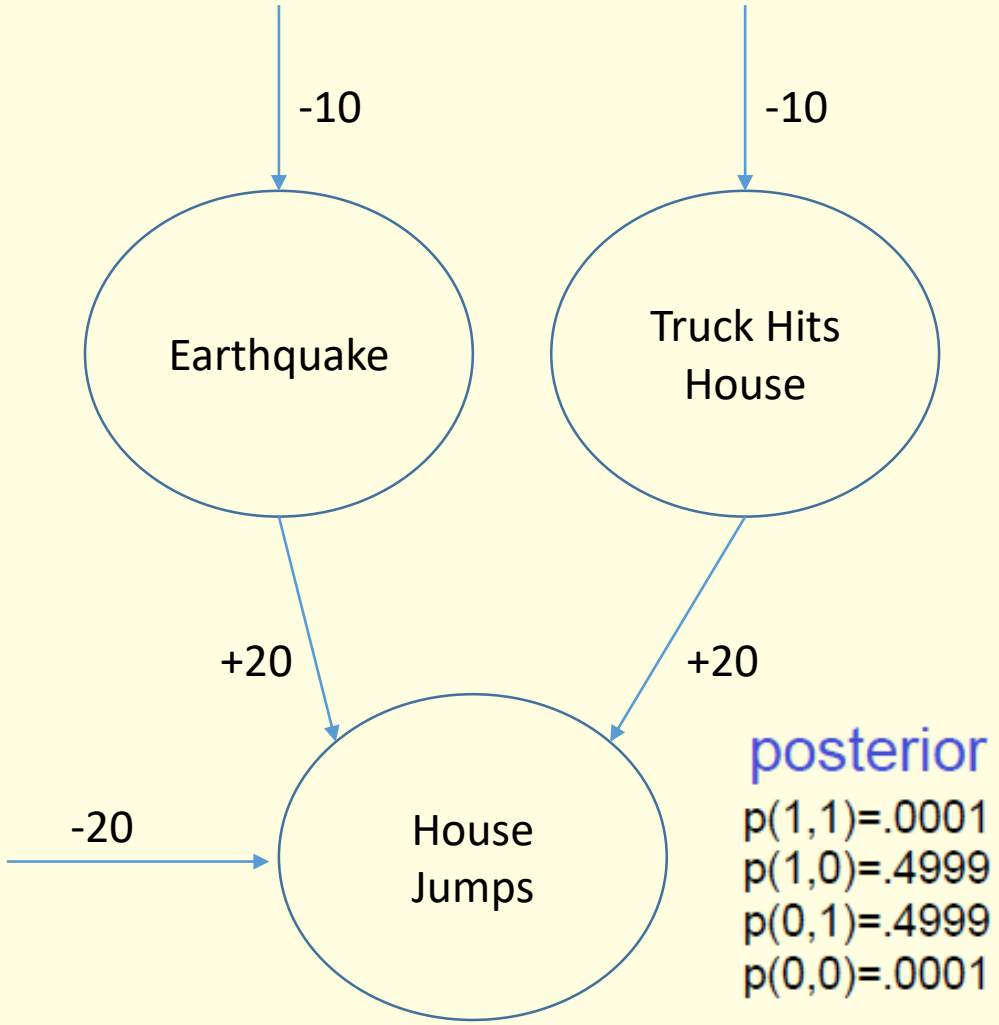
weights on the directed connections from the ancestors

Complementary Priors

- "Explaining away" makes inference difficult in directed belief nets.
- The **posterior** distribution over the hidden variables is intractable
  - Mixture models
  - Linear models with additive Gaussian noise
- Markov chain Monte Carlo methods (Neal, 1992) to sample from the posterior (Time Consuming)
- Variational methods (Neal & Hinton, 1998) use simple approximations to the true conditional distribution,
  - May be poor (the deepest hidden layer)
  - Still requires all of the parameters to be learned together
  - The learning time scale poorly as the number of parameters increases

Explaining away

- Even if two hidden causes are independent, they can become dependent when we observe an effect that they can both influence.
  - If we learn that there was an earthquake it reduces the probability that the house jumped because of a truck.

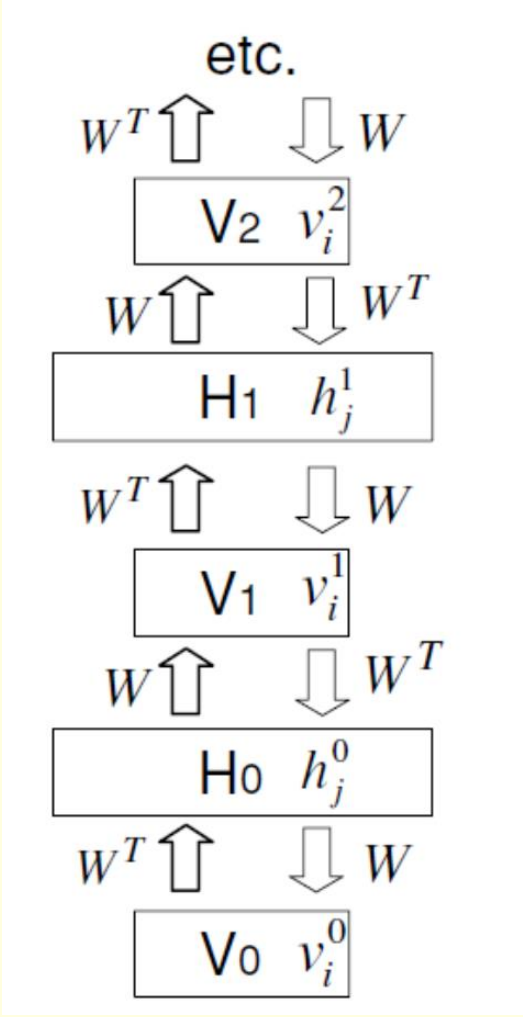


An Infinite Directed Model with Tied Weights

- Generate data from infinite directed net by starting with a random configuration at an infinitely deep hidden layer
- Perform a top-down “ancestral” pass
- Sample from the true posterior distribution over all of the hidden layers
- Sample from the factorial posterior before computing the factorial posterior for the layer above

➢this procedure gives unbiased samples

- The complementary prior at each layer ensures that the posterior distribution really is factorial.



The derivatives of the generative

$$\frac{\partial \log p(\mathbf{v}^0)}{\partial w_{ij}^{00}} = \langle h_j^0 (v_i^0 - \hat{v}_i^0) \rangle$$

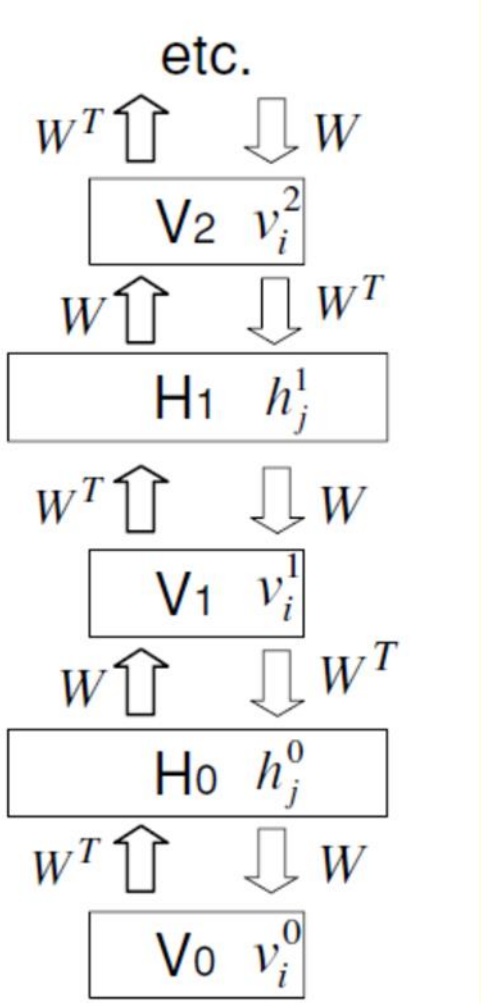
Average over the sampled states

$$\frac{\partial \log p(\mathbf{v}^0)}{\partial w_{ij}^{00}} = \langle h_j^0 (v_i^0 - v_i^1) \rangle.$$

v_i^1 Sample from a Bernoulli random variable with probability $\hat{v}_i^0$

The dependence of v_i^1 on h_j^0 is unproblematic in the derivation because $\hat{v}_i^0$ is an expectation that is conditional on h_j^0

$$\frac{\partial \log p(\mathbf{v}^0)}{\partial w_{ij}} = \langle h_j^0 (v_i^0 - v_i^1) \rangle + \langle v_i^1 (h_j^0 - h_j^1) \rangle + \langle h_j^1 (v_i^1 - v_i^2) \rangle + \cdots$$



$$\frac{\partial \log p(\mathbf{v}^0)}{\partial w_{ij}} = \langle h_j^0 (v_i^0 - v_i^1) \rangle + \langle v_i^1 (h_j^0 - h_j^1) \rangle + \langle h_j^1 (v_i^1 - v_i^2) \rangle + \cdots$$

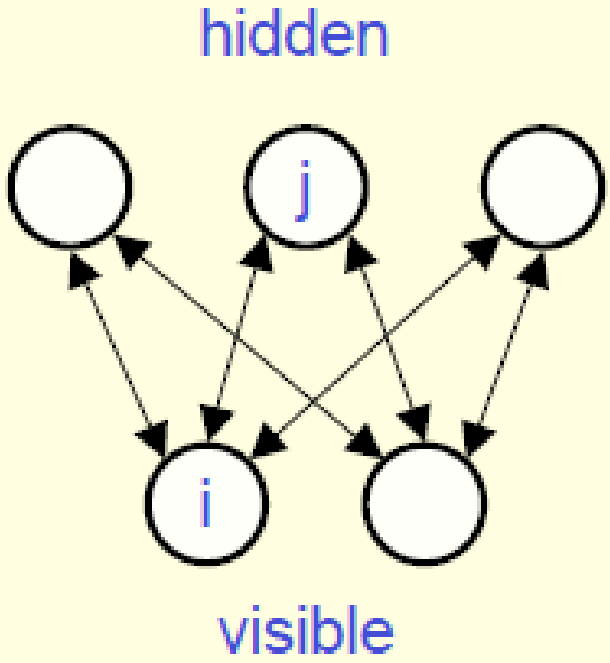
Pairwise products except the first and last cancel

$$\frac{\partial \log p(\mathbf{v}^0)}{\partial w_{ij}} = \langle v_i^0 h_j^0 \rangle - \langle v_i^\infty h_j^\infty \rangle$$

Boltzmann machine learning rule

Restricted Boltzmann Machines

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



RBM

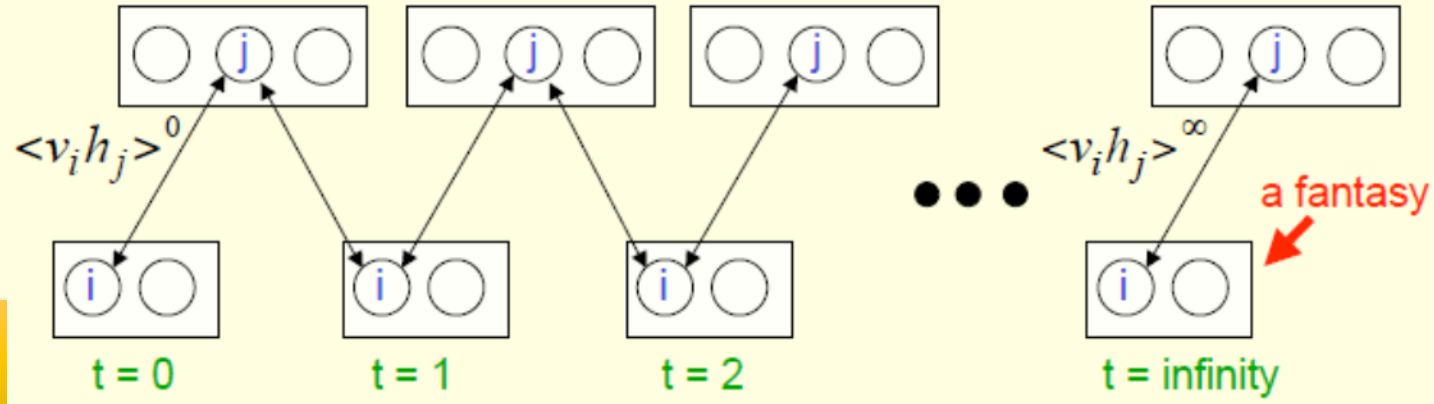
- To generate data from an RBM:

1. Start with a random state in one of the layers
2. Perform alternating Gibbs sampling
3. All of the units in one layer are updated in parallel given the current states of the units in the other layer
4. Repeat until sampling from its equilibrium distribution

Maximum likelihood learning rule in an RBM

- Difference between two correlations $\frac{\partial \log p(v)}{\partial w_{ij}} = <v_i h_j>^0 - <v_i h_j>^\infty$

1. Measure the correlation $\langle v_i^0 h_j^0 \rangle$
2. Hidden states are sampled from their conditional distribution, which is factorial
3. Using alternating Gibbs sampling, run the Markov chain until it reaches its stationary distribution and measure the correlation $\langle v_i^\infty h_j^\infty \rangle$



Contrastive Divergence Learning

- Maximizing the log probability of the data is exactly the same as minimizing the Kullback-Leibler divergence

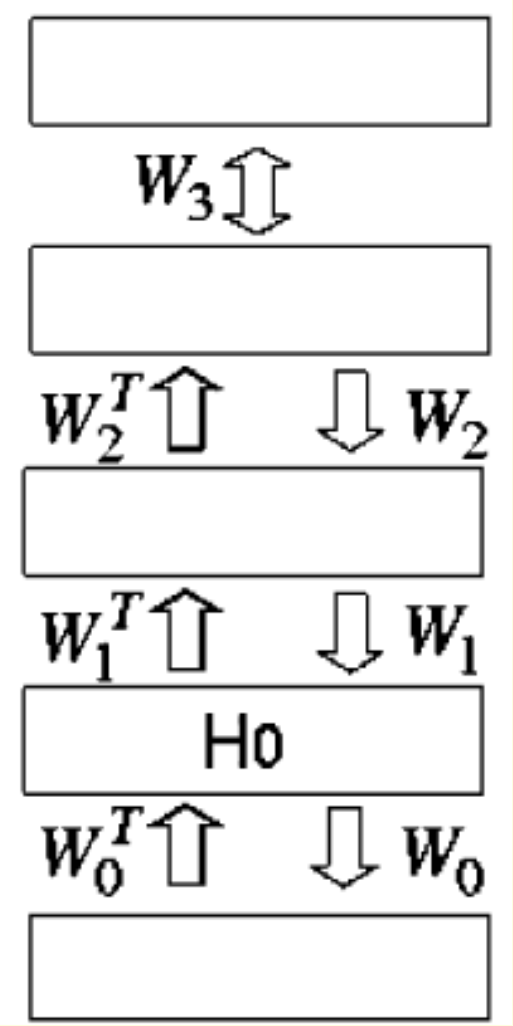
$$KL(P^0||P_\theta^\infty)$$

  - P^0: Distribution of the data
  - P_θ^∞: The equilibrium distribution defined by the model
- Run Markov chain for *n* full steps before measuring the second correlation (Each full step consists of updating **h** given **v**, then updating **v** given **h**)
- Contrastive divergence learning minimizes:

$$KL(P^0||P_\theta^\infty) - KL(P_\theta^n||P_\theta^\infty)$$

A Greedy Learning Algorithm for Transforming Representations

- The idea behind this algorithm is to allow each model in the sequence to receive a different representation of the data.
- Nonlinear transformation on its input vectors and produces as output the vectors that will be used as input for the next model in the sequence.
- To simplify the analysis, all layers have the same number of units.
- The RBM will not be able to model the original data perfectly



A Greedy Learning Algorithm for Transforming Representations

1. Learn W_0 assuming all the weight matrices are tied.

- 2. Freeze W_0 and commit ourselves to using W_0^T to infer factorial approximate posterior distributions over the states of the variables in the first hidden layer, even if subsequent changes in higher-level weights mean that this inference method is no longer correct.
- 3. Keeping all the higher-weight matrices tied to each other, but untied from W_0, learn an RBM model of the higher-level "data" that was produced by using W_0^T to transform the original data.
- In practice, replace the maximum likelihood Boltzmann machine learning algorithm by contrastive divergence learning because it works well and is much faster.

Back-Fitting with the Up-Down Algorithm

1. Greedily learning good initial values for the weights
2. A variant of the wake-sleep algorithm described in Hinton et al. (1995) can then be used to allow the higher-level weights to influence the lower-level ones.

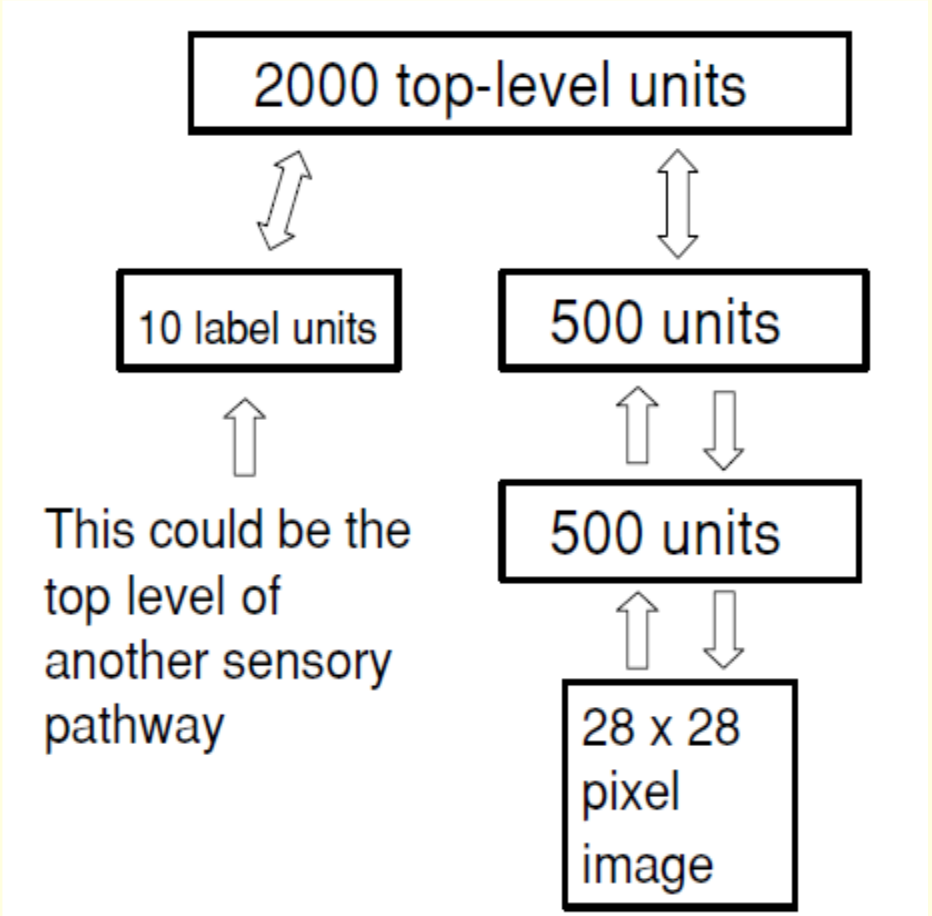
- "up-pass":
  - The weights on the directed connections are adjusted using ML learning
  - The weights on the undirected connections are learned as before
- "down-pass":
- Undirected connections and the generative directed connections are not changed. Only the bottom-up recognition weights are modified.

Performance on the MNIST Database

- Handwritten digits contains 60,000 training images and 10,000 test images.
- Basic version of the MNIST learning task
  - No knowledge of geometry is provided
  - No special preprocessing or enhancement of the training set
- An unknown but fixed random permutation of the pixels would not affect the learning algorithm.
- For this “permutation invariant” version of the task, the generalization performance was 1.25% errors on the official test set.

Performance on the MNIST Database

- Was trained on 44,000 of the training images that were divided into 440 balanced mini-batches, each containing 10 examples of each digit class. The weights were updated after each mini-batch.
- Each layer was trained for 30 sweeps through the training set (called “epochs”).
- The error rate on the test set was 2.49%



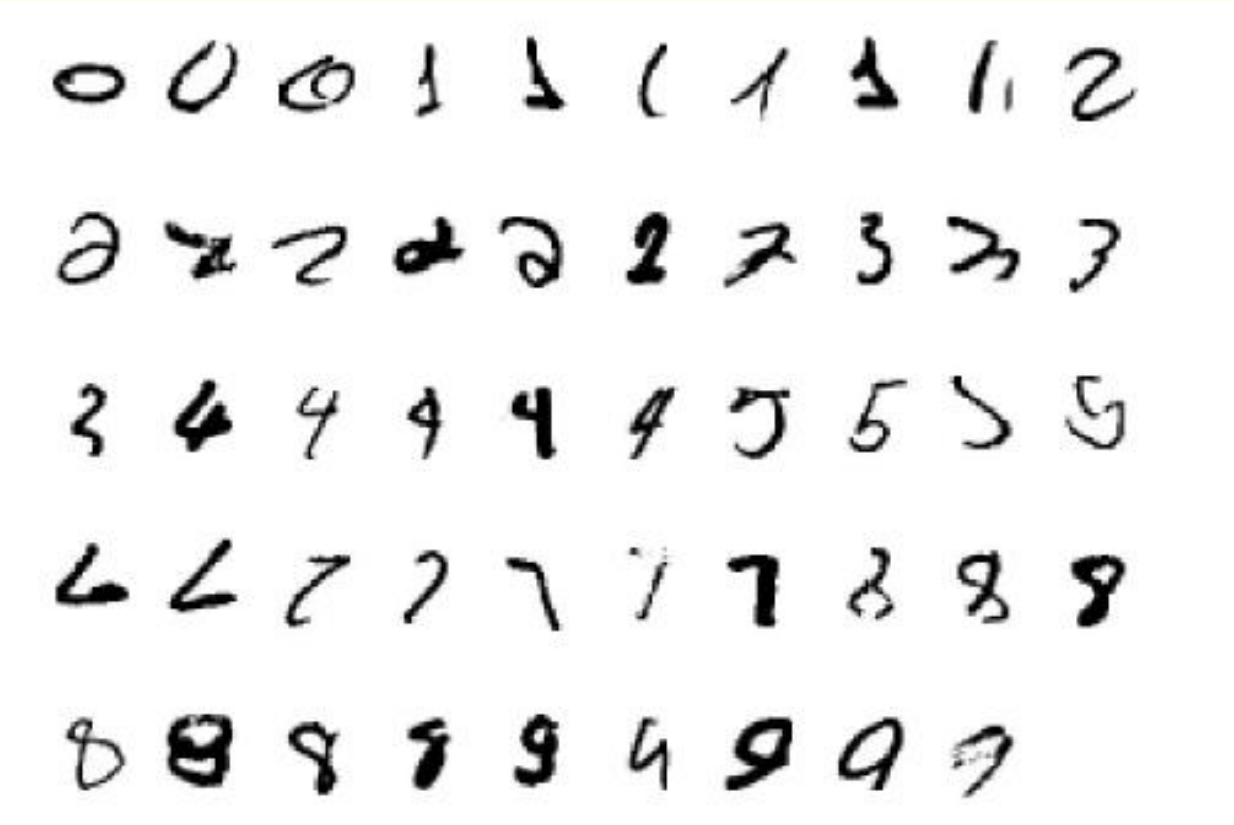
Performance on the MNIST Database

- Network training:
- 300 epochs using the up-down algorithm
- The learning rate, momentum, and weight decay were chosen by training the network several times and observing its performance.
- On a separate validation set of 10,000 images were taken from the remainder of the full training set
  - First 100 epochs: Three full iterations of alternating Gibbs sampling
  - Second 100 epochs: 6 iterations were performed
  - Last 100 epochs: 10 iterations were performed
- Each time the number of iterations of Gibbs sampling was raised, the error on the validation set decreased noticeably.

Performance on the MNIST Database

- The network that performed best on the validation set was tested and had an error rate of 1.39%.
- This network was then trained on all 60,000 training images until its error rate on the full training set was as low as its final error rate had been on the initial training set of 44,000 images.

Version of MNIST Task	Learning Algorithm	Test Error %
Permutation invariant	Our generative model: $784 \rightarrow 500 \rightarrow 500 \leftrightarrow 2000 \leftrightarrow 10$	1.25
Permutation invariant	Support vector machine: degree 9 polynomial kernel	1.4
Permutation invariant	Backprop: $784 \rightarrow 500 \rightarrow 300 \rightarrow 10$ cross-entropy and weight-decay	1.51
Permutation invariant	Backprop: $784 \rightarrow 800 \rightarrow 10$ cross-entropy and early stopping	1.53
Permutation invariant	Backprop: $784 \rightarrow 500 \rightarrow 150 \rightarrow 10$ squared error and on-line updates	2.95
Permutation invariant	Nearest neighbor: all 60,000 examples and L3 norm	2.8
Permutation invariant	Nearest neighbor: all 60,000 examples and L2 norm	3.1
Permutation invariant	Nearest neighbor: 20,000 examples and L3 norm	4.0
Permutation invariant	Nearest neighbor: 20,000 examples and L2 norm	4.4
Unpermuted images; extra data from elastic deformations	Backprop: cross-entropy and early-stopping convolutional neural net	0.4
Unpermuted de-skewed images; extra data from 2 pixel translations	Virtual SVM: degree 9 polynomial kernel	0.56
Unpermuted images	Shape-context features: hand-coded matching	0.63
Unpermuted images; extra data from affine transformations	Backprop in LeNet5: convolutional neural net	0.8
Unpermuted images	Backprop in LeNet5: convolutional neural net	0.95



49 cases in which the network guessed right but had a second guess whose probability was within 0.3 of the probability of the best guess.

Testing the Network

- One way to test the network is to use a stochastic up-pass from the image to fix the binary states of the 500 units in the lower layer of the associative memory
- The second is to repeat the stochastic up-pass 20 times and average either the label probabilities or the label log probabilities over the 20 repetitions before picking the best one.
- The two types of average give almost identical results, and these results are also very similar to using a single deterministic up-pass, which was the method used for the reported results.

Looking into the Mind of a Neural Network

- To generate samples from the model:

1. We perform alternating Gibbs sampling in the top-level associative memory until the Markov chain converges to the equilibrium distribution.
2. We use a sample from this distribution as input to the layers below and generate an image by a single down-pass through the generative connections.
3. If we clamp the label units to a particular class during the Gibbs sampling, we can see images from the model's class-conditional distributions.

- Each row shows 10 samples from the generative model with a particular label clamped on. The top-level associative memory is run for 1000 iterations of alternating Gibbs sampling between samples.

Looking into the Mind of a Neural Network

- To generate samples from the model:

1. We perform alternating Gibbs sampling in the top-level associative memory until the Markov chain converges to the equilibrium distribution.
2. We use a sample from this distribution as input to the layers below and generate an image by a single down-pass through the generative connections.
3. If we clamp the label units to a particular class during the Gibbs sampling, we can see images from the model's class-conditional distributions.
4. We can also initialize the state of the top two layers by providing a random binary image as input.

Each row shows 10 samples from the generative model with a particular label clamped on. The top-level associative memory is initialized by an up-pass from a random binary image in which each pixel is on with a probability of 0.5. The first column shows the results of a down-pass from this initial high level state. Subsequent columns are produced by 20 iterations of alternating Gibbs sampling in the associative memory.

Conclusion

- It is possible to learn a deep, densely connected belief network one layer at a time.
- The higher layers do not exist when learning the lower layers
  - Not compatible with the use of simple factorial approximations to replace the intractable posterior distribution
- They exist but have tied weights that are constrained to implement a complementary prior that makes the true posterior exactly factorial.
  - An undirected model that can be learned efficiently using contrastive divergence

Limitations

- It is designed for images in which nonbinary values can be treated as probabilities (which is not the case for natural images)
- Its use of top-down feedback during perception is limited to the associative memory in the top two layers
- It does not have a systematic way of dealing with perceptual invariances
- It assumes that segmentation has already been performed
- It does not learn to sequentially attend to the most informative parts of objects when discrimination is difficult.

Advantages

- Generative models can learn low-level features without requiring feedback from the label, and they can learn many more parameters than discriminative models without overfitting
- It is easy to see what the network has learned by generating from its model.
- It is possible to interpret the nonlinear, distributed representations in the deep hidden layers by generating images from them.
- The superior classification performance of discriminative learning methods holds only for domains in which it is not possible to learn a good generative model. This set of domains is being eroded by Moore's law.