CSE 5526: Introduction to Neural Networks

Deep Networks

Motivation

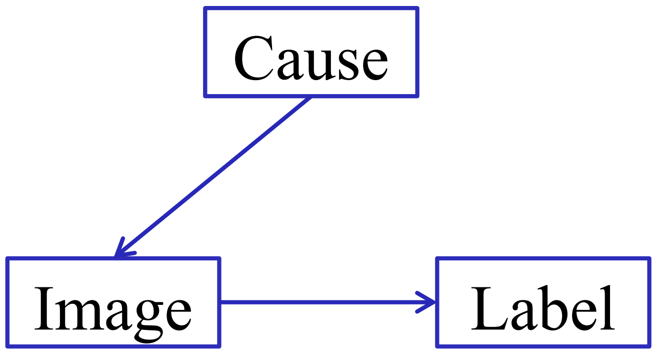
- Shallow nets involve one (hidden) layer of feature detectors followed by an output layer, e.g. MLP with one hidden layer, RBF, and SVM
- Deep nets use more than one hidden layer, e.g. convolutional nets
  - Ex: The addition of two 3-bit binary numbers. The most natural solution would use a small net of two hidden layers. A shallow net can do the job, but with a large net

Motivation (cont.)

- Backprop is applicable to an arbitrary number of hidden layers. But in practice, it exhibits vanishing gradients in shallower (near the input layer) layers. As a result, training is slow and tends to overfit the data
- To overcome this problem, we train a deep net by dividing training into two phases:
  - Use unsupervised, generative pre-training to initialize a deep net
  - Use discriminative fine-tuning to finalize the net

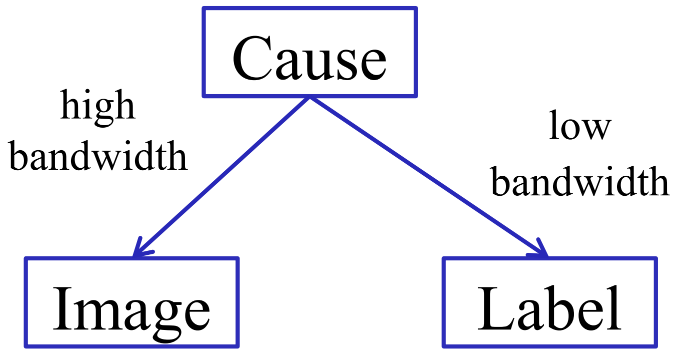
Motivation (cont.)

- Why two phases?



Traditional learning

vs.



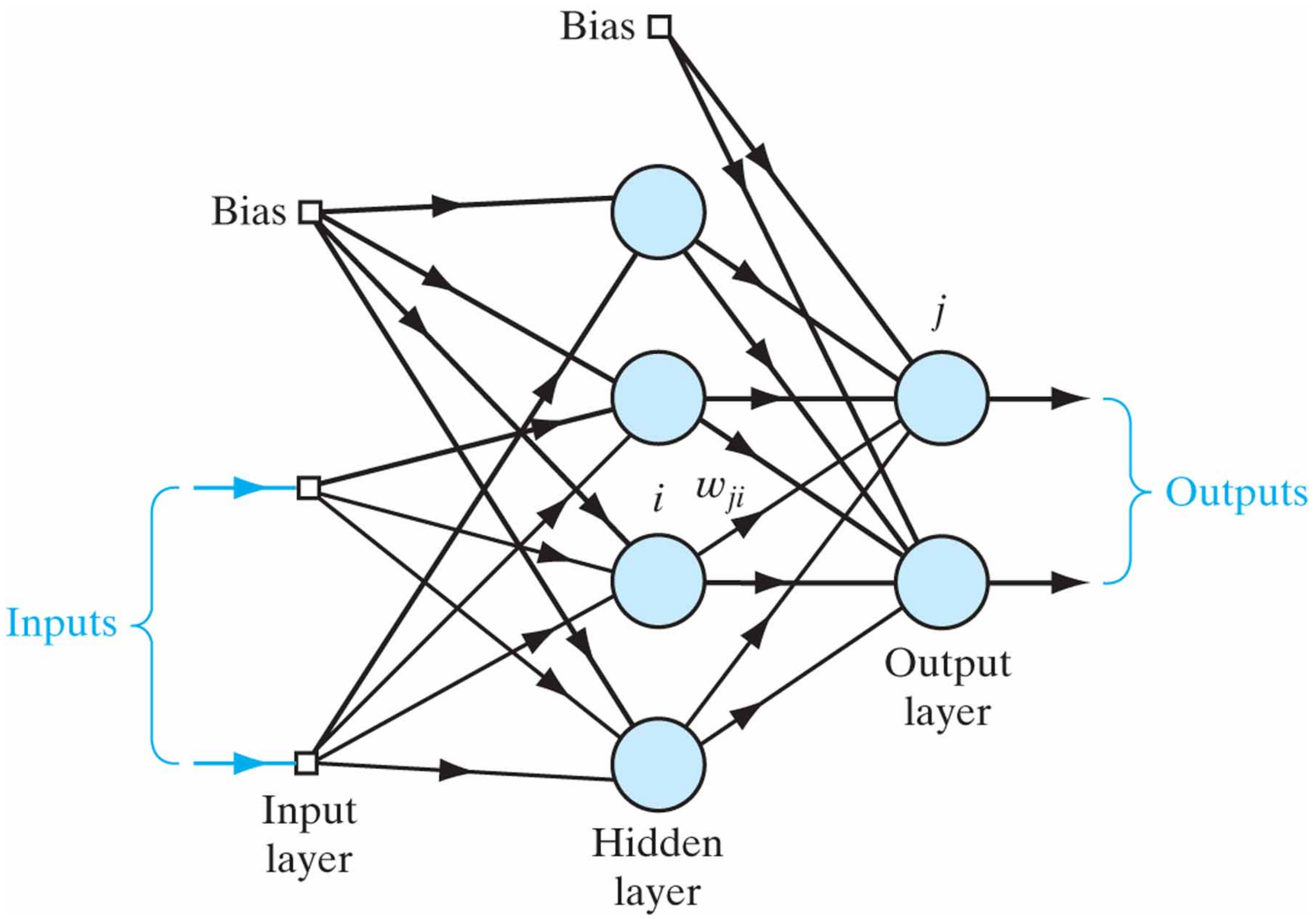
Deep learning

Motivation (cont.)

- Because overfitting is not as problematic for unsupervised learning, we expect deep learning to generalize better
- In addition, Phase 1 allows us to validate what the net has learned by generating patterns from the learned, generative model

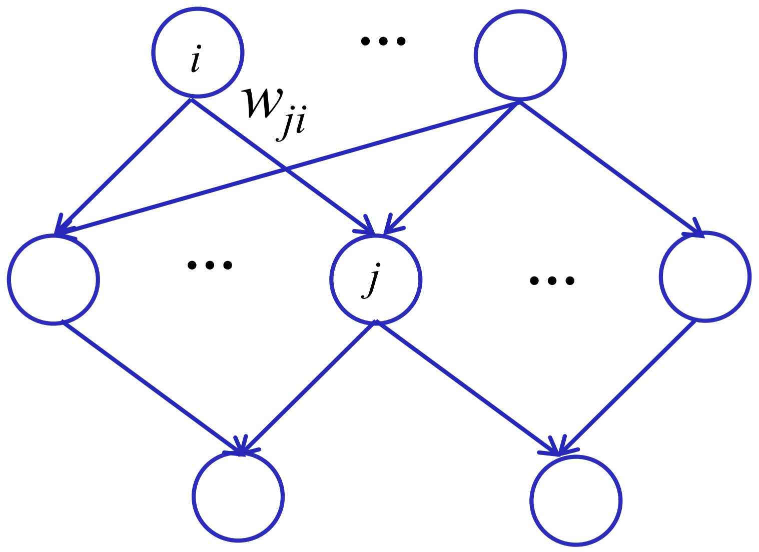
Logistic belief net

- An example logistic belief net



Logistic belief net (cont.)

- Each unit is bipolar (binary) and stochastic. Generating data from the belief net is easy



State of each unit

- Given the bipolar states of the units in layer k, we generate the state of each unit in layer $k-1$:

$$P\left(h_j{}^{(k-1)} = 1\right) = \varphi\left(\sum_i w_{ji}{}^{(k)} h_i{}^{(k)}\right)$$

where superscript indicates layer number and

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

a logistic activation function

Learning rule

- The bottom layer $\mathbf{h}^{(0)}$ is the same as the visible layer $\mathbf{v}$
- Learning in a belief net is to maximize the likelihood of generating the input patterns applied to $\mathbf{v}$. Similar to Boltzmann learning, we have

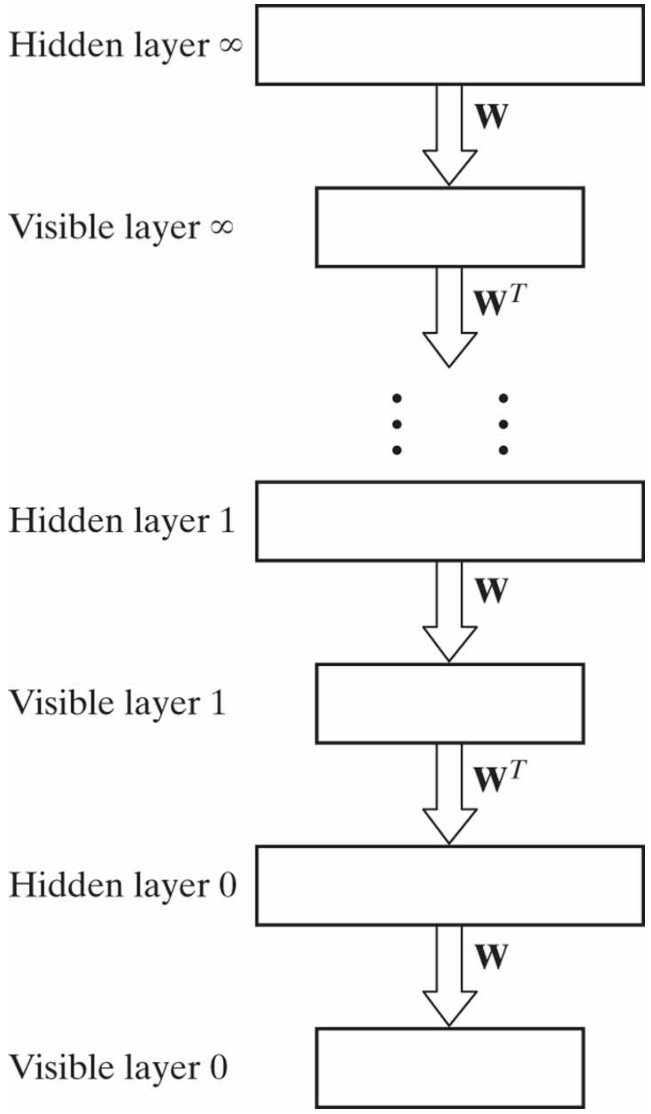
$$\Delta w_{ji} = < h_i{}^{(k)} \left\{ h_j{}^{(k-1)} - 2[P\left(h_j{}^{(k-1)} = 1\right) - \frac{1}{2}] \right\} >$$

- The difference term in the above equation includes an evaluation of the posterior prob. given the training data
- Computing posteriors is, unfortunately, very difficult

A special belief net

- However, for a special kind of belief net, computing posteriors is easy
- Consider a logistic belief net with an infinite no. of layers and tied weights
  - That is, a deep belief net (DBN)

Infinite logistic net



- In such a net, sampling the posterior prob. is the same as generating data in belief nets

Learning in DBN

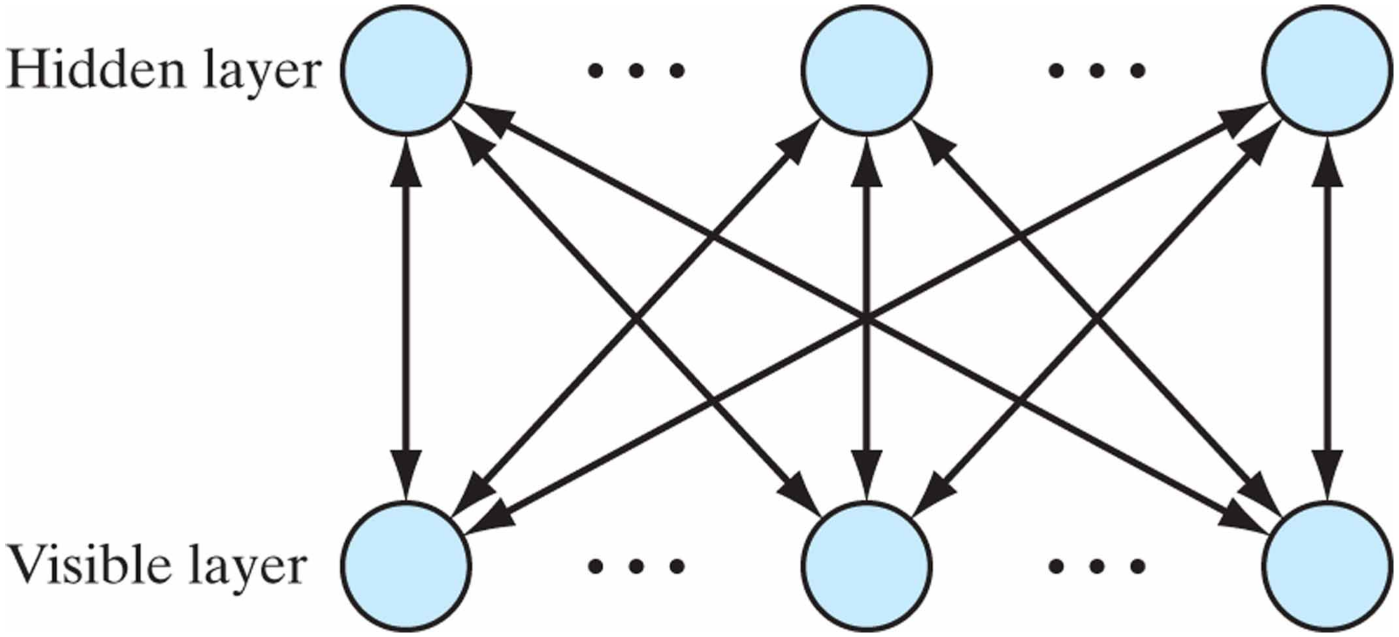
- Because of the tied weights,

$$\frac{\partial L(\mathbf{w})}{\partial w_{ji}} =< h_i{}^{(0)}\left(v_j{}^{(0)} - v_j{}^{(1)}\right) >+< v_j{}^{(1)}\left(h_i{}^{(0)} - h_i{}^{(1)}\right) >+$$
$$< h_i{}^{(1)}\left(v_j{}^{(1)} - v_j{}^{(2)}\right) > \cdots$$

$$=< h_i{}^{(0)} v_j{}^{(0)} >-< h_i{}^{(\infty)} v_j{}^{(\infty)} >$$

Restricted Boltzmann machines

- A restricted Boltzmann machine (RBM) is a Boltzmann machine with one visible layer and one hidden layer, and no connection within each layer



Energy function of RBM

- The energy function is:

$$E(\mathbf{v},\mathbf{h}) = -\sum_i \sum_j w_{ji} v_j h_i$$

- Setting $T = 1$, we have

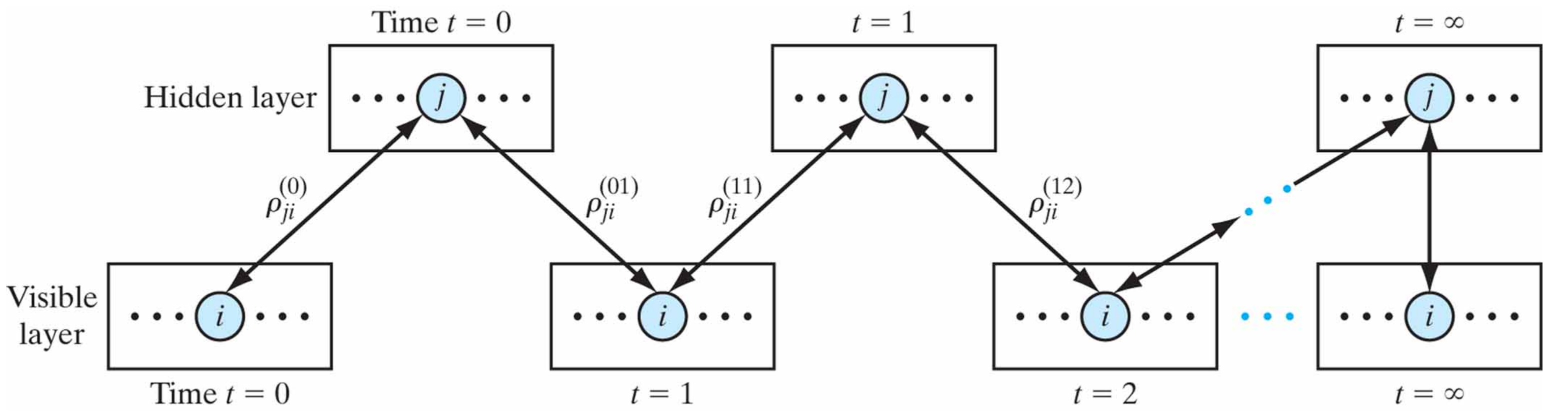
$$\frac{\partial L(\mathbf{w})}{\partial w_{ji}} = {\rho_{ji}}^{+} - {\rho_{ji}}^{-}$$

$$= < {h_i}^{(0)} {v_j}^{(0)} > - < {h_i}^{(\infty)} {v_j}^{(\infty)} >$$

  - The second correlation is computed using alternating Gibbs sampling until thermal equilibrium

Learning in RBM

- This rule is exactly the same as the one for the infinite logistic belief net
  - Hence the equivalence between learning a DBN and an RBM



Contrastive divergence

- In practice, a quick way to learn RBM:

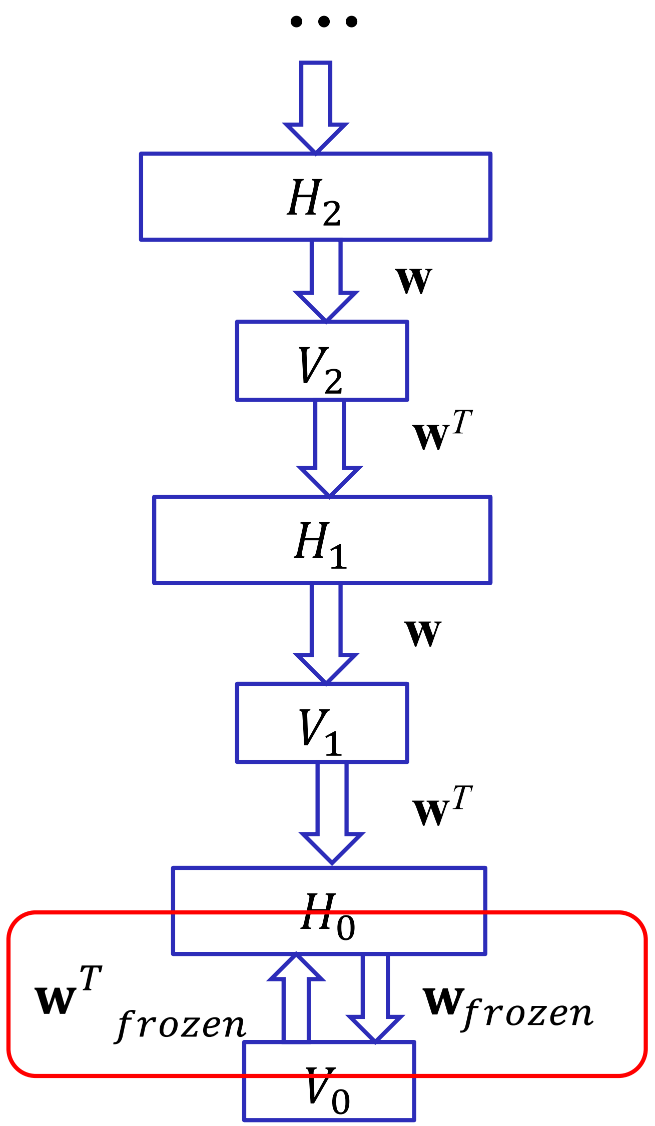
$$\vartriangle w_{ji} = \eta(< h_i{}^{(0)} v_j{}^{(0)} > - < h_i{}^{(1)} v_j{}^{(1)} >)$$

- The above rule is called contrastive divergence

Training a general deep net layer-by-layer

1. First learn **w** with all weights tied
2. Freeze (fix) **w**, which represents the learned weights for the first hidden layer
3. Learn the weights for the second hidden layer by treating responses of the first hidden layer to the training data as "input data"
4. Freeze the weights for the second hidden layer
5. Repeat steps 3-4 as many times as the prescribed number of hidden layers

Illustration



- This training process is the same as repeatedly training RBMs layer-by-layer, which is adopted due to its simplicity

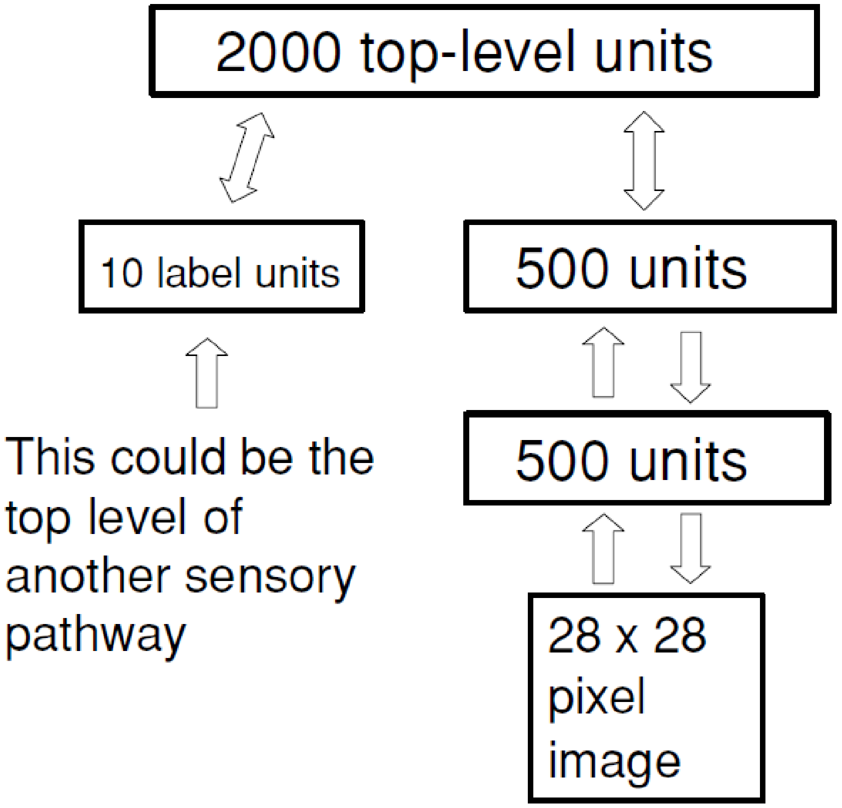
Remarks

- As the number of layers increases, the maximum likelihood approximation of the training data improves
- For discriminative training (e.g. for classification) we add an output layer on top of the learned generative model, and train the entire net by a discriminative algorithm
  - e.g. backprop or SVM
- Although much faster than Boltzmann machines (e.g. no simulated annealing), pretraining is still quite slow, and involves a lot of design as for MLP

Applications

- DNNs have been successfully applied to a large number of tasks
- Ex: MNIST handwritten digit recognition
  - A DNN with two hidden layers achieves 1.25% error rate, vs. 1.4% for SVM and 1.5% for MLP

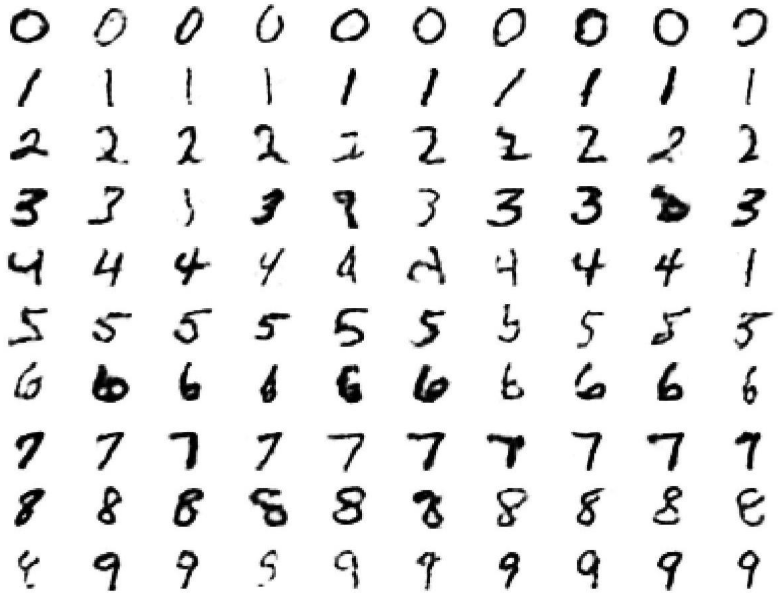
Digit recognition DNN



- The network used to model the joint distribution of digit images and digit labels

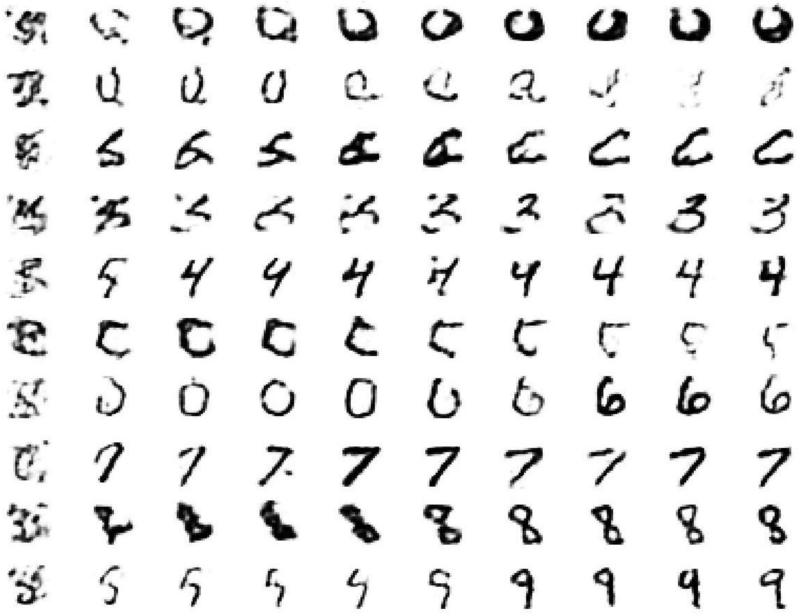
Digit recognition illustration

- Samples from the learned generative model



  - 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

More samples



- 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