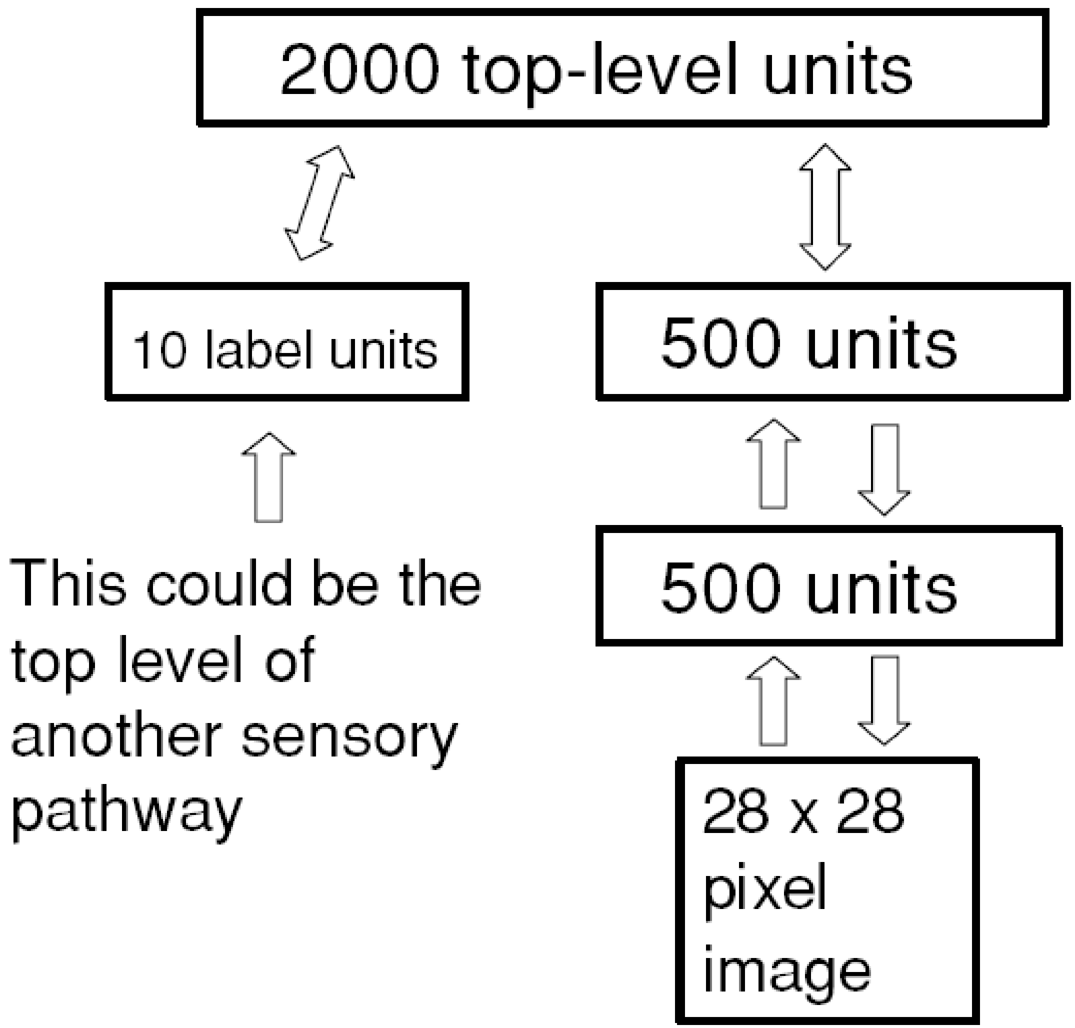
- Energy models and Deep Belief Networks
  - Definition
    - examples
    - overcomplete ICA: generative vs. energy-based models
  - Learning
    - contrastive divergence
    - (score matching)
  - Restricted Boltzmann Machines
  - Deep Belief Networks
    - infinite networks <-> RBMs
    - learning: greedy + wake-sleep phase

Deep Belief Networks

- Demo: http://www.cs.toronto.edu/~hinton/digits.html
- Other datasets: http://www.cs.toronto.edu/~roweis/data.html

Goal architecture



Ingredient 1

- Write code to perform inference and CD learning in a RBM
  – Test that, given weights and visible unit v, you get a correct distribution over h (both probabilities and samples); same for v given h
  – Test that, given v and perturbing correct w, the algorithm comes back to minimum
- Optional: insert decay and momentum term

Ingredient 2

- Write code to perform inference and CD learning in a RBM with softmax labels
  - Define simple model with 2 labels, 2 hidden units, 4 input units; label 1 generates either [1,0,0,0] or [0,1,0,0]; label 2 generates [0,0,1,0] or [0,0,0,1]
  - Verify that generating with fixed labels gives you correct input patterns
  - Verify that after learning, labels are inferred correctly in all 4 cases
- Optional: insert decay and momentum term

Greedy learning

- At this point you can already train the DBN with the greedy algorithm
  - ! The input to each RBM is given by the probabilities over the hidden states at the previous layer
  - Begin using a few letters from the small dataset `binaryalphadigs.mat` (100 units per layer should be ok)
  - Try to generate letters by clamping one of the labels
  - Given all examples of one class, how invariant is the representation in the various layers? Does that change if you learn a model without labels?
  - If you want to use the MNIST dataset, divide the data in balanced mini-batches (e.g., 10 examples from each class)

Classification

- Try to classify the input letters:
  - easy, inaccurate: use the recognition model, set uniform probabilities over labels; look at the probabilities over labels at equilibrium
  - hard, more accurate: get representation at top hidden layer, compute free energy for that state and each label lit in turn (Teh, Hinton, 2001)

Up-down algorithm

- Write code to refine the greedy solution using the up-down algorithm
  - see Appendix B in (Hinton et al., 2006)
  - increase the number of Gibbs iterations at the top during learning
- Optional: how does accuracy depend on the architecture? How good can one get by just greedily learning many layers?