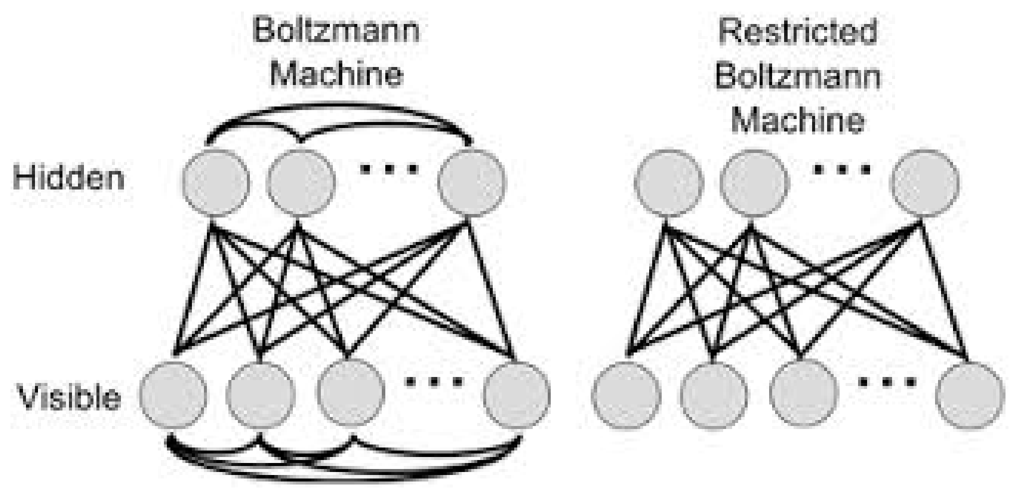
Extra lab 4 support

DD2437

Pawel Herman

CST/EECS/KTH

Restricted Boltzmann machine (RBM)

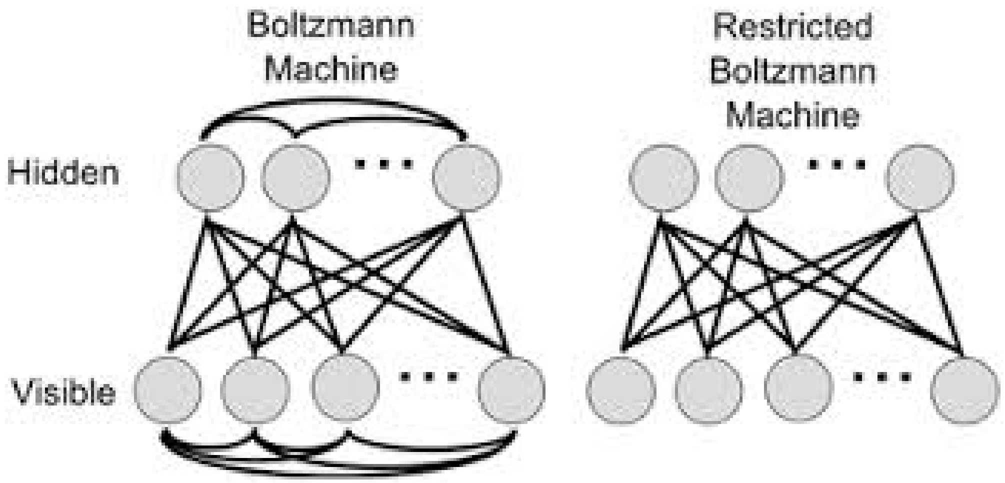


Visible and hidden units are conditionally independent given one another

$$p(\boldsymbol{h} \mid \boldsymbol{v}) = \prod_i p(h_i \mid \boldsymbol{v})$$

$$p(\boldsymbol{v} \mid \boldsymbol{h}) = \prod_j p(v_j \mid \boldsymbol{h})$$

Restricted Boltzmann machine (RBM)



Visible and hidden units are conditionally independent given one another

$$p(\boldsymbol{h} \mid \boldsymbol{v}) = \prod_i p(h_i \mid \boldsymbol{v})$$

$$p(\boldsymbol{v} \mid \boldsymbol{h}) = \prod_j p(v_j \mid \boldsymbol{h})$$

Following the same principle of maximising log likelihood by means of gradient ascent, one obtains:

$$\Delta w_{ji} = \varepsilon \frac{\partial L(\mathbf{W})}{\partial w_{ji}} = \varepsilon \left(\left\langle v_j h_i \right\rangle_{\text{data}} - \left\langle v_j h_i \right\rangle_{\text{model}} \right)$$

Restricted Boltzmann machine (RBM)

Visible and hidden units are conditionally independent given one another

$$p(\boldsymbol{h} \mid \boldsymbol{v}) = \prod_i p(h_i \mid \boldsymbol{v})$$

$$p(\boldsymbol{v} \mid \boldsymbol{h}) = \prod_j p(v_j \mid \boldsymbol{h})$$

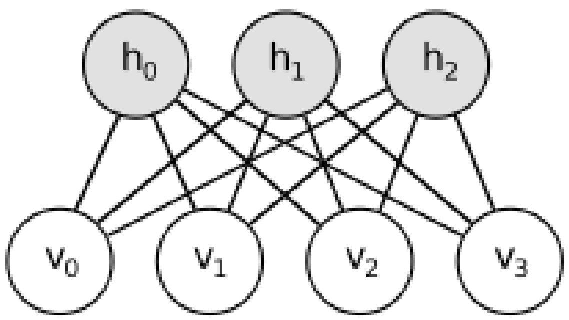
$$P(h_i = 1 \mid \boldsymbol{v}) = \frac{1}{1 + \exp\left(-bias_{h_i} - \boldsymbol{v}^{\mathrm{T}} \mathbf{W}_{:,i}\right)}$$

$$P(v_j = 1 \mid \boldsymbol{h}) = \frac{1}{1 + \exp\left(-bias_{v_j} - \mathbf{W}_{j,:}\, \boldsymbol{h}\right)}$$

Following the same principle of maximising log likelihood by means of gradient ascent, one obtains:

$$\Delta w_{ji} = \varepsilon \frac{\partial L(\mathbf{W})}{\partial w_{ji}} = \varepsilon \left(\left\langle v_j h_i \right\rangle_{\text{data}} - \left\langle v_j h_i \right\rangle_{\text{model}} \right)$$

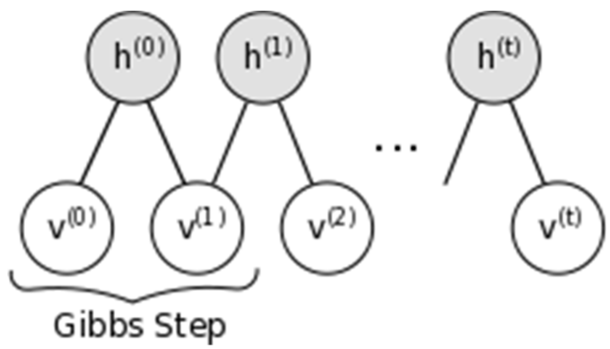
RBM learning with Contrastive Divergence (CD)

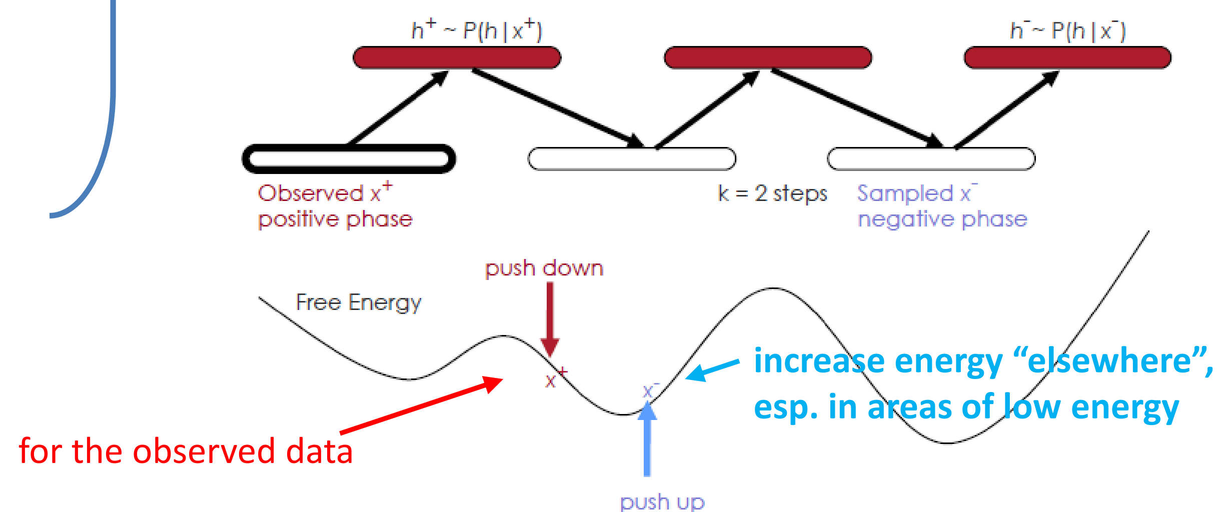


$$P(h_i = 1 \mid \boldsymbol{v}) = \frac{1}{1 + \exp\left(-bias_{h_i} - \boldsymbol{v}^{\mathrm{T}} \mathbf{W}_{:,i}\right)}$$

$$P(v_j = 1 \mid \boldsymbol{h}) = \frac{1}{1 + \exp\left(-bias_{v_j} - \mathbf{W}_{j,:} \boldsymbol{h}\right)}$$

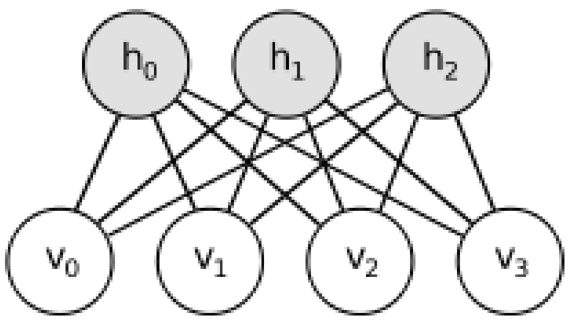
Gibbs sampling





Hinton, 2003

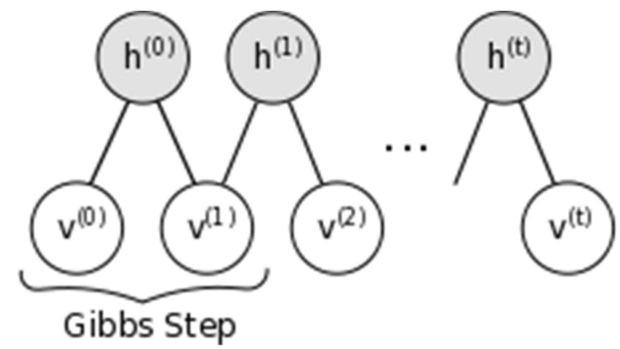
RBM learning with Contrastive Divergence (CD)

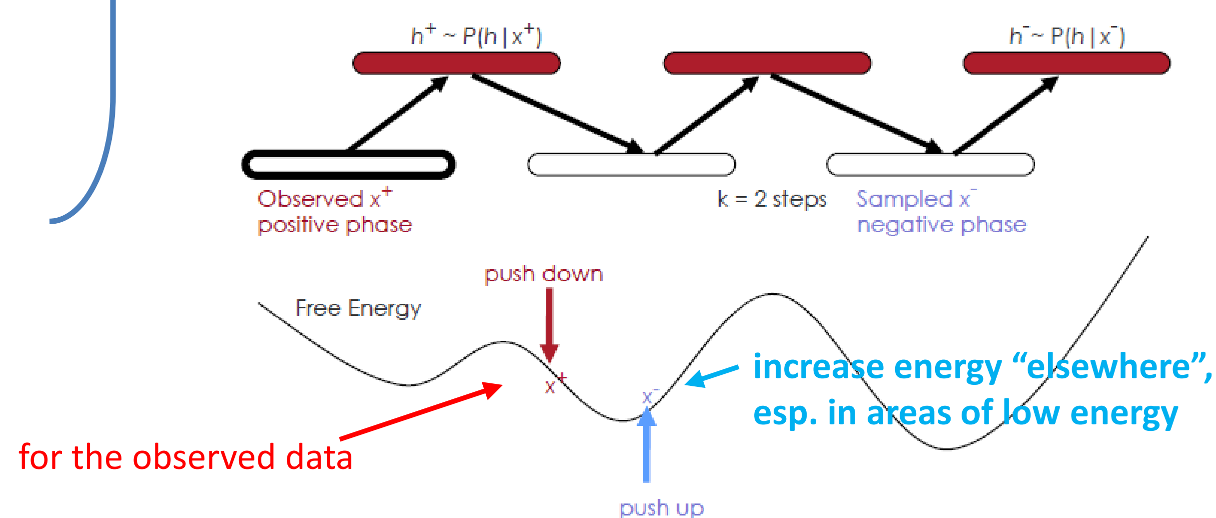


$$P(h_i = 1 \mid \boldsymbol{v}) = \frac{1}{1 + \exp\left(-bias_{h_i} - \boldsymbol{v}^{\mathrm{T}} \mathbf{W}_{:,i}\right)}$$

$$P(v_j = 1 \mid \boldsymbol{h}) = \frac{1}{1 + \exp\left(-bias_{v_j} - \mathbf{W}_{j,:}\,\boldsymbol{h}\right)}$$

Gibbs sampling





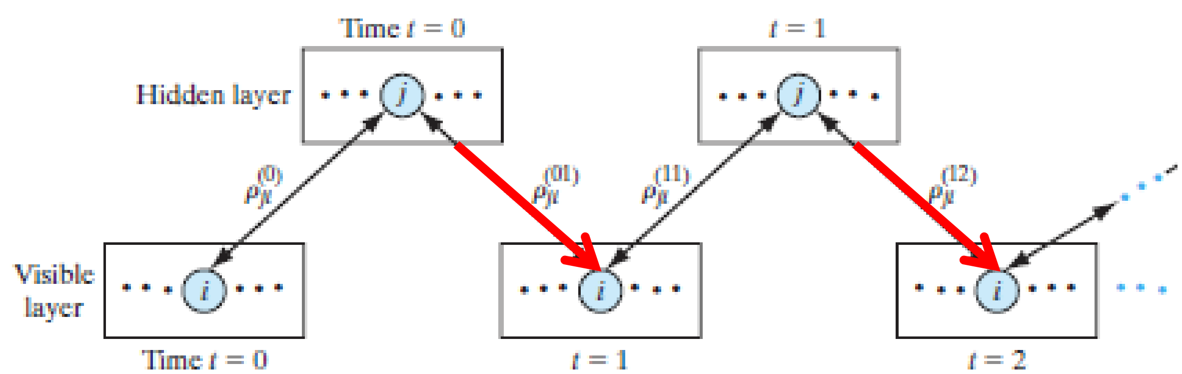
Hinton, 2003

GOOD TO KNOW:

Contrastive Divergence does not optimise the likelihood but it works effectively!

CD_k recipe for training RBM

Gibbs sampling



Objective:

$$\langle v_j h_i \rangle_{\text{data}} \text{ and } \langle v_j h_i \rangle_{\text{model}}$$

1) Set (*clamp*) the visible units with an input vector and update hidden units (binary states).

$$P(h_i = 1 \mid \boldsymbol{v}) = \left(1 + \exp\left(-bias_{h_i} - \boldsymbol{v}^{\mathrm{T}} \mathbf{W}_{:,i}\right)\right)^{-1}$$

2) Update all the visible units in parallel to get a **reconstruction** (probabs can be used).

$$P(v_j = 1 \mid \boldsymbol{h}) = \left(1 + \exp\left(-bias_{v_j} - \mathbf{W}_{j,:} \boldsymbol{h}\right)\right)^{-1}$$

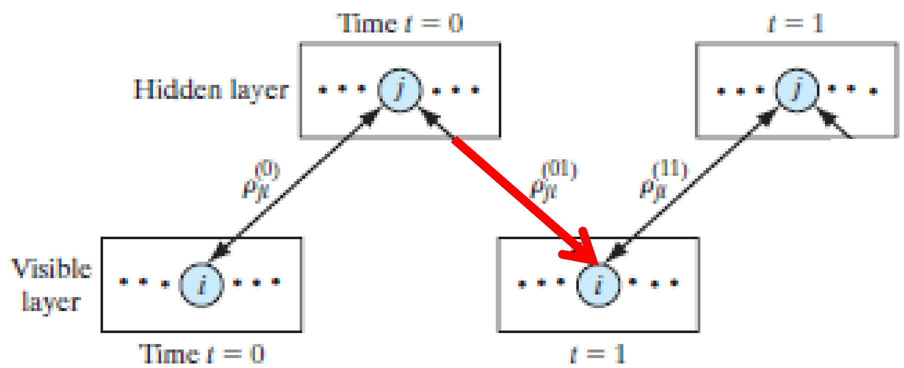
3) Collect the statistics for correlations after *k* steps using mini-batches (*N* samples) and update weights:

$$\Delta w_{j,i} = \frac{1}{N} \sum_{n=1}^{N} \left(v_j^{(n)} h_i^{(n)} - \overbrace{\hat{v}_j^{(n)} \hat{h}_i^{(n)}}^{k\text{-th step}} \right)$$

The final update of the hidden units should use the probability.

CD$_1$ case

Gibbs sampling



$$\Delta w_{j,i} \propto \left\langle v_j^{(0)}\, h_i^{(0)} \right\rangle - \left\langle v_j^{(1)}\, h_i^{(1)} \right\rangle = \frac{1}{N}\sum_{n=1}^{N}\left(v_j^{(0,n)}\, h_i^{(0,n)} - \hat{v}_j^{(1,n)}\, \hat{h}_i^{(1,n)} \right)$$

probabilities

binary samples

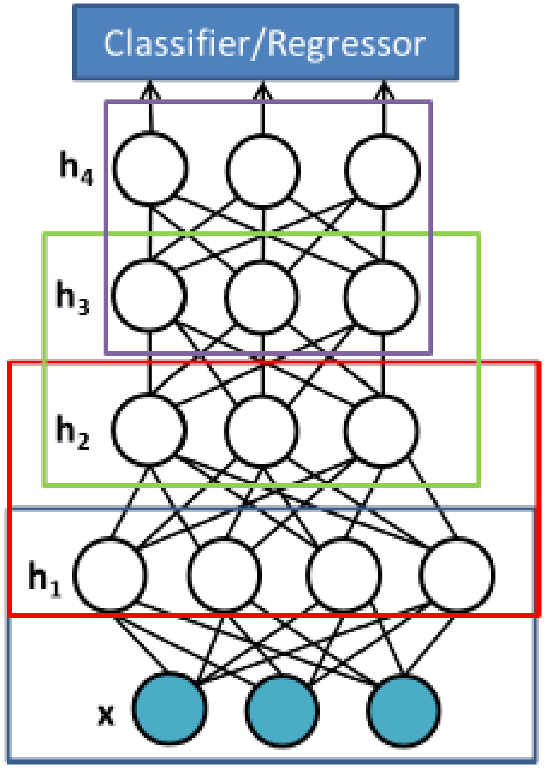
probabilities

$$\Delta bias_j^{(v)} \propto \left\langle v_j^{(0)} \right\rangle - \left\langle v_j^{(1)} \right\rangle$$

$$\Delta bias_i^{(h)} \propto \left\langle h_i^{(0)} \right\rangle - \left\langle h_i^{(1)} \right\rangle$$

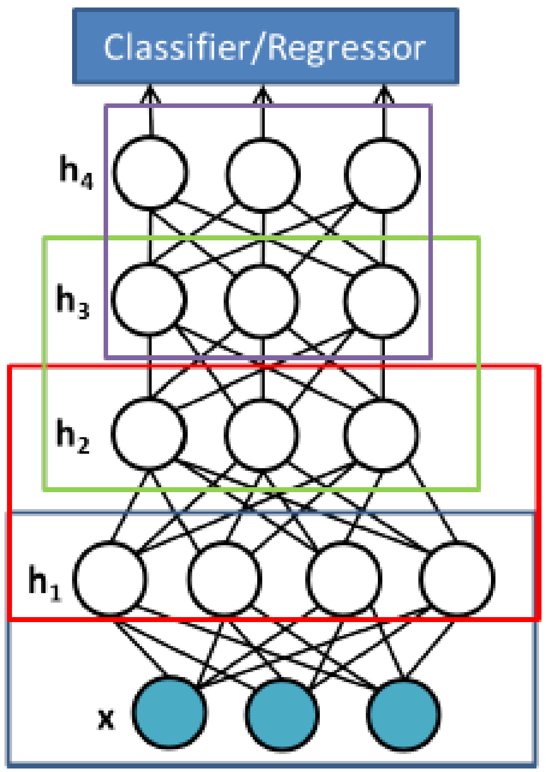
Deep belief nets

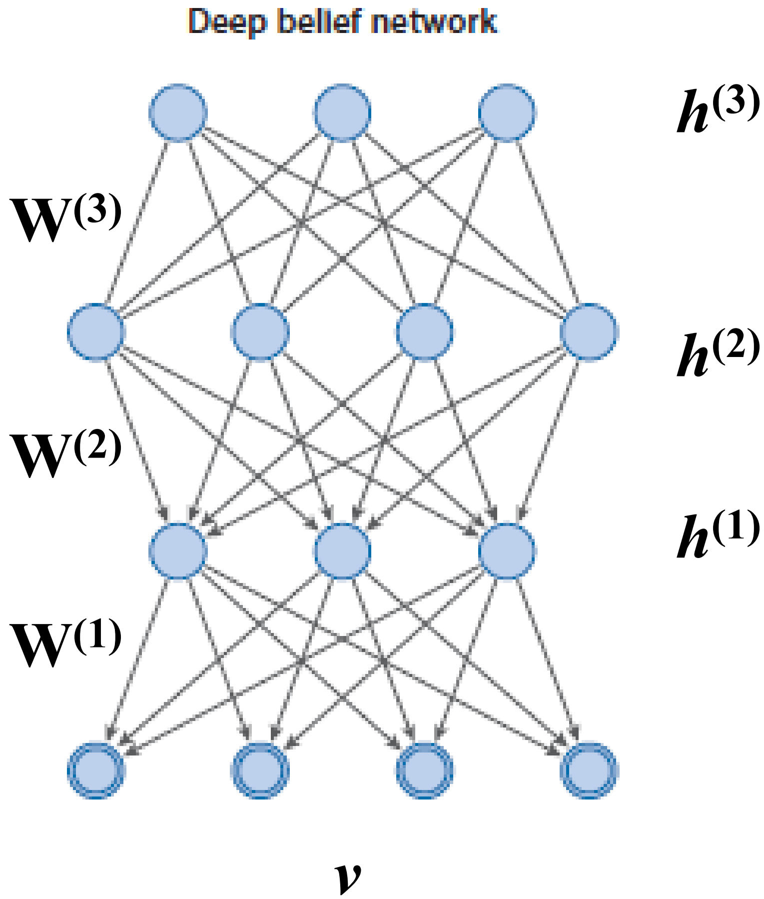
Greedy layer-wise training approach
with the use of RBMs



Deep belief nets

Greedy layer-wise training approach with the use of RBMs

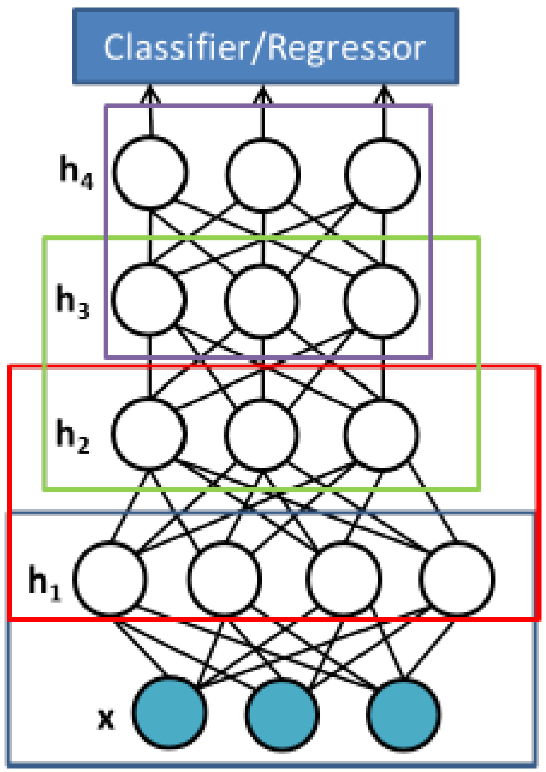


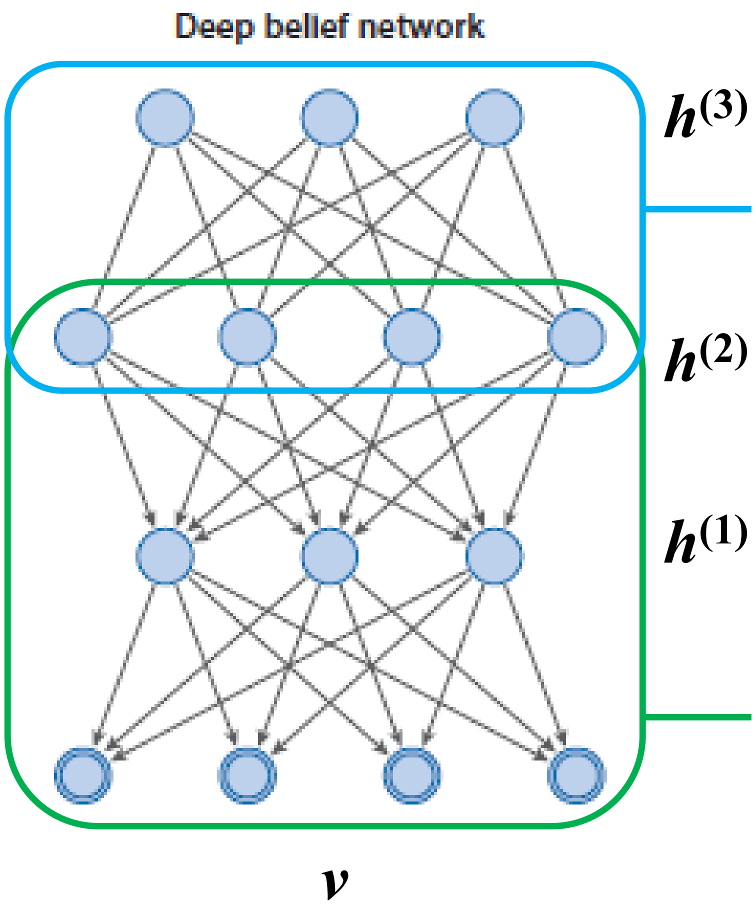


Salakhutdinov, 2015

Deep belief nets

Greedy layer-wise training approach with the use of RBMs





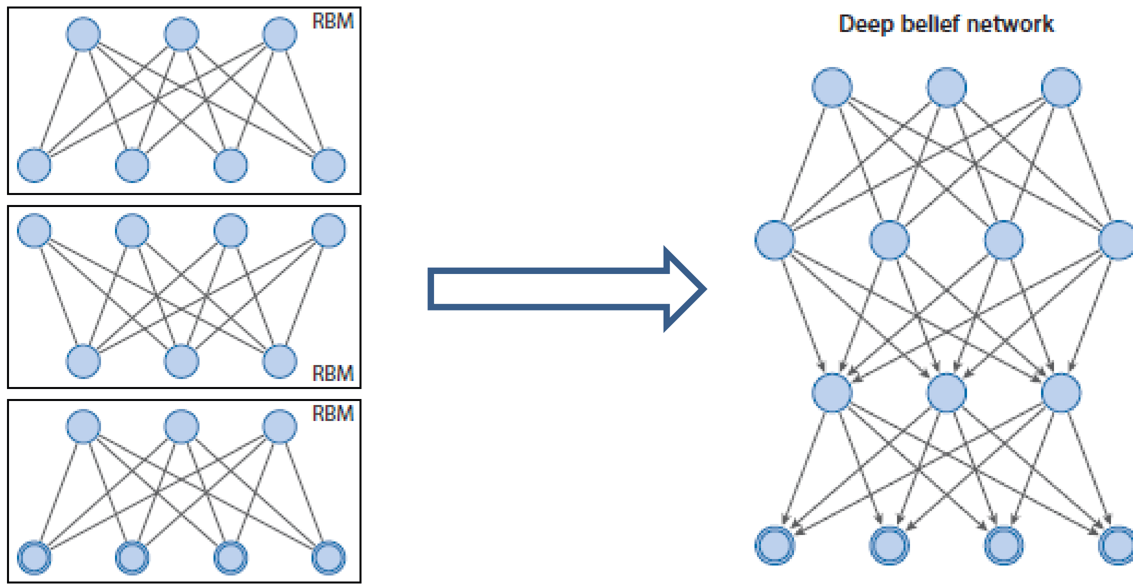
undirected part of the network (bipartite graph of RBM)

directed part of the network (sigmoid belief network)

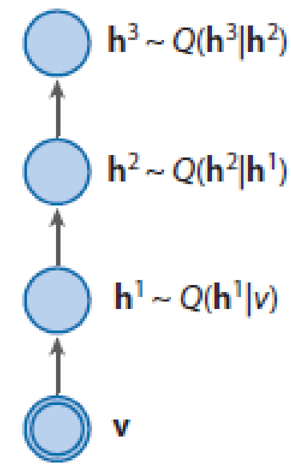
Salakhutdinov, 2015

Deep belief nets

Approach 1



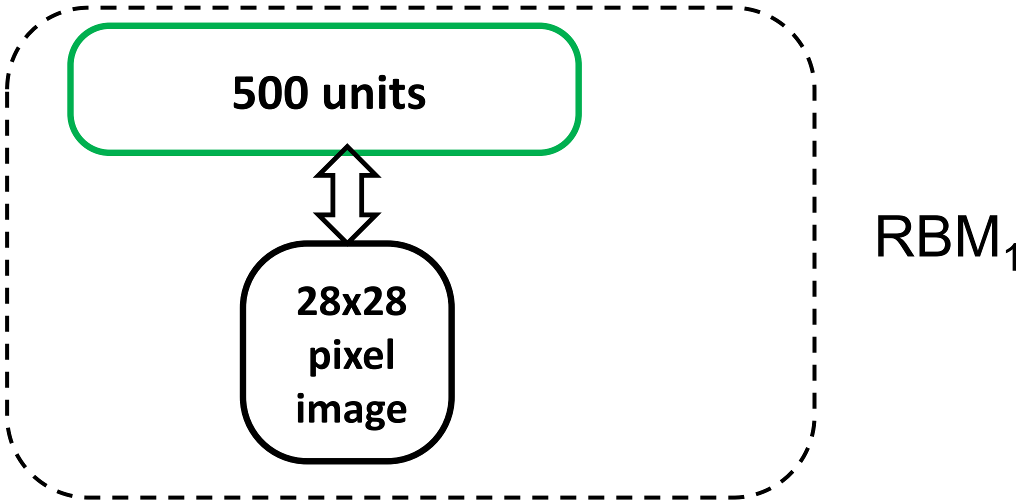
Bottom-up pass by stochastically activating higher layers in time



1: Fit the parameters $W^{(1)}$ of the first-layer RBM to data.
2: Fix the parameter vector $W^{(1)}$, and use samples $\mathrm{h}^{(1)}$ from $Q(\mathrm{h}^{(1)}|\mathrm{v}) = P(\mathrm{h}^{(1)}|\mathrm{v}, W^{(1)})$ as the data for training the next layer of binary features with an RBM.
3: Fix the parameters $W^{(2)}$ that define the second layer of features, and use the samples $\mathrm{h}^{(2)}$ from $\boxed{Q(\mathrm{h}^{(2)}|\mathrm{h}^{(1)}) = P(\mathrm{h}^{(2)}|\mathrm{h}^{(1)}, W^{(2)})}$ as the data for training the third layer of binary features.
4: Proceed recursively for the next layers.

Hinton et al.'s (2006) architecture

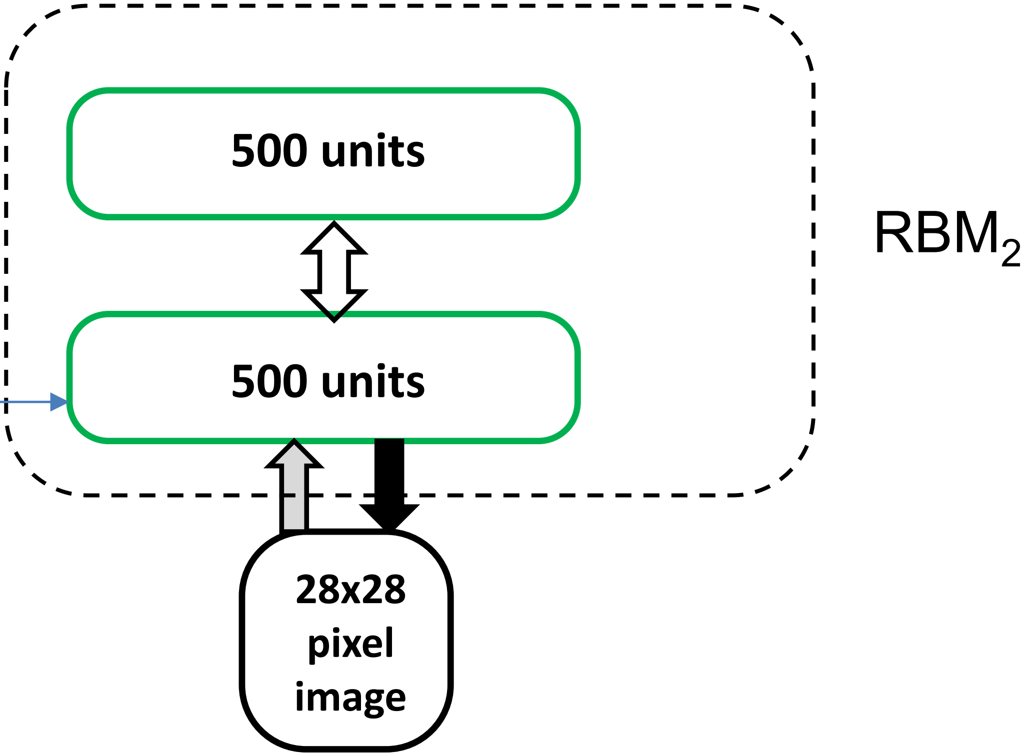
Building the stack of RBMs



Hinton et al.'s (2006) architecture

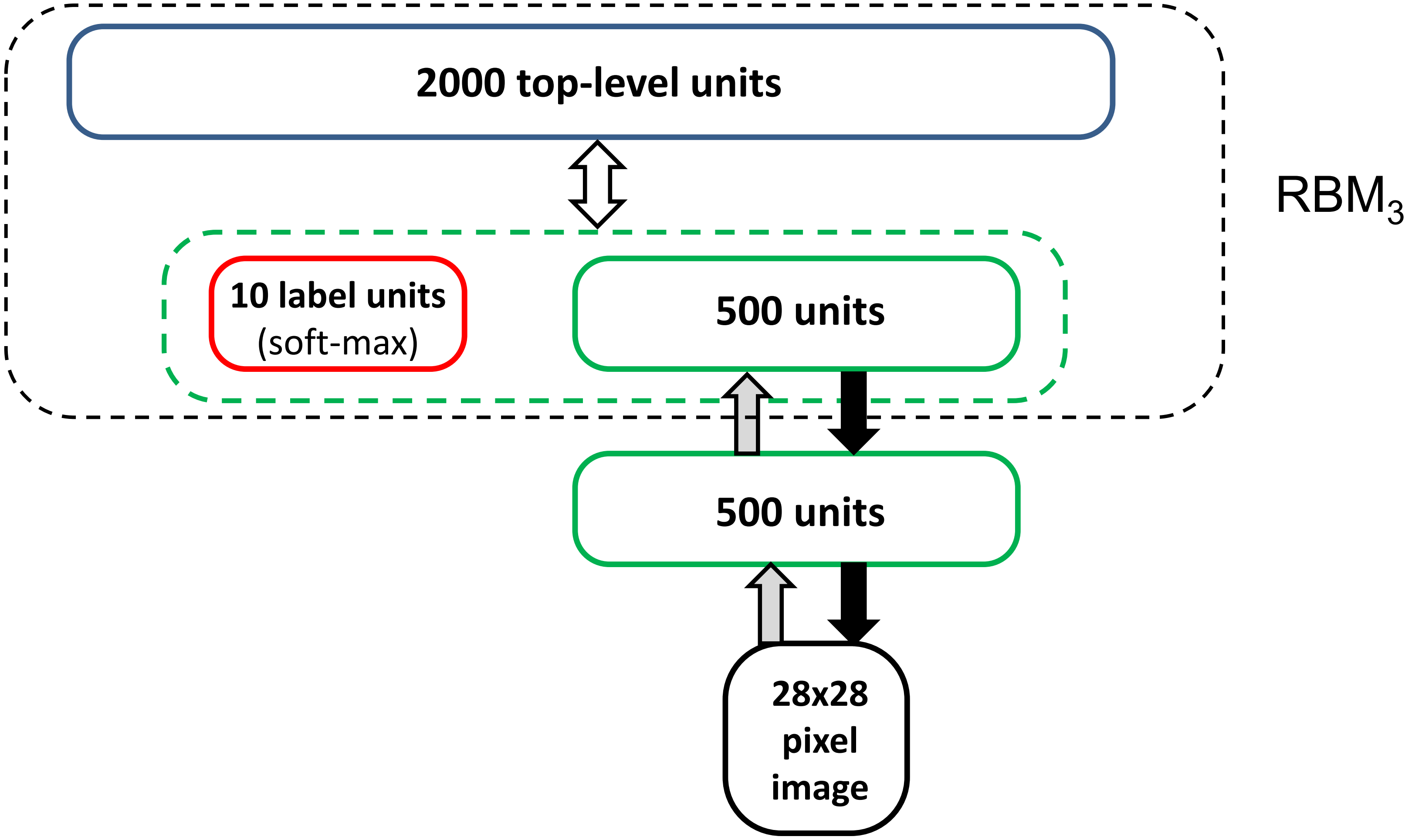
Building the <u>stack of RBMs</u>

The visible layer of RBM_2 is treated as probabilities (just like $v^{(0)}$ in CD)



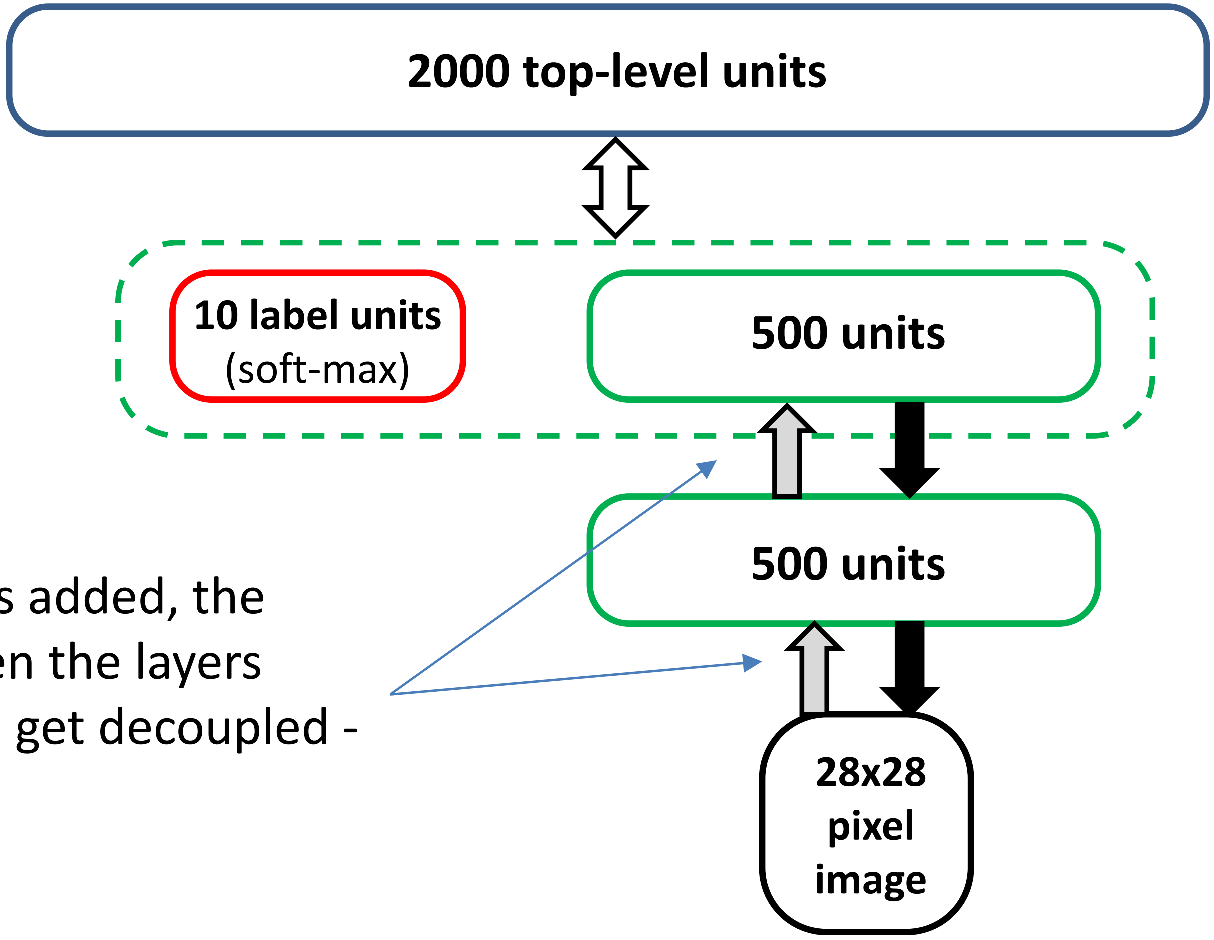
Hinton et al.'s (2006) architecture

Building the stack of RBMs



Hinton et al.'s (2006) architecture

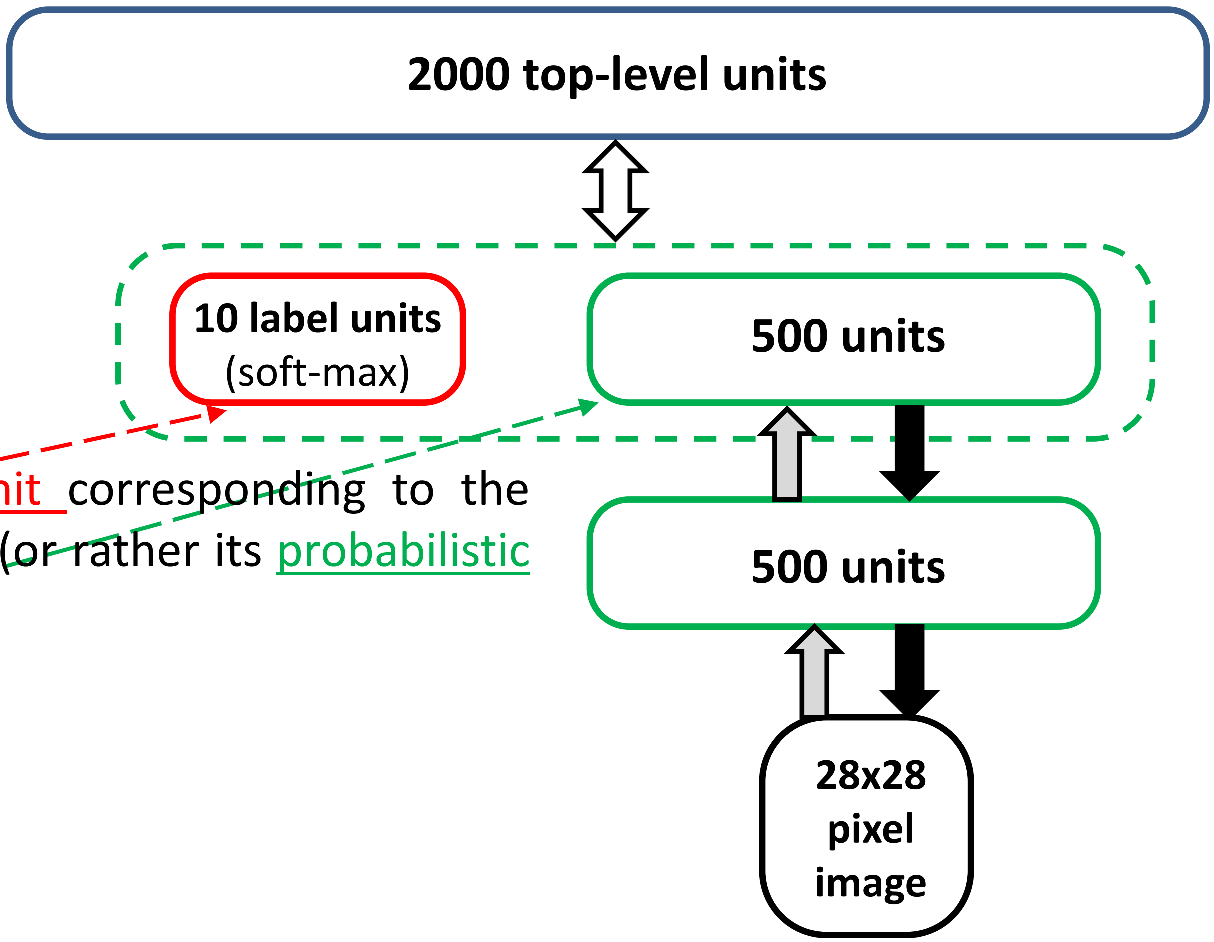
The network used to model joint distribution of digit images and labels.



Once the top layer is added, the connections between the layers below (now hidden) get decoupled - unidirectional

Hinton et al.'s (2006) architecture

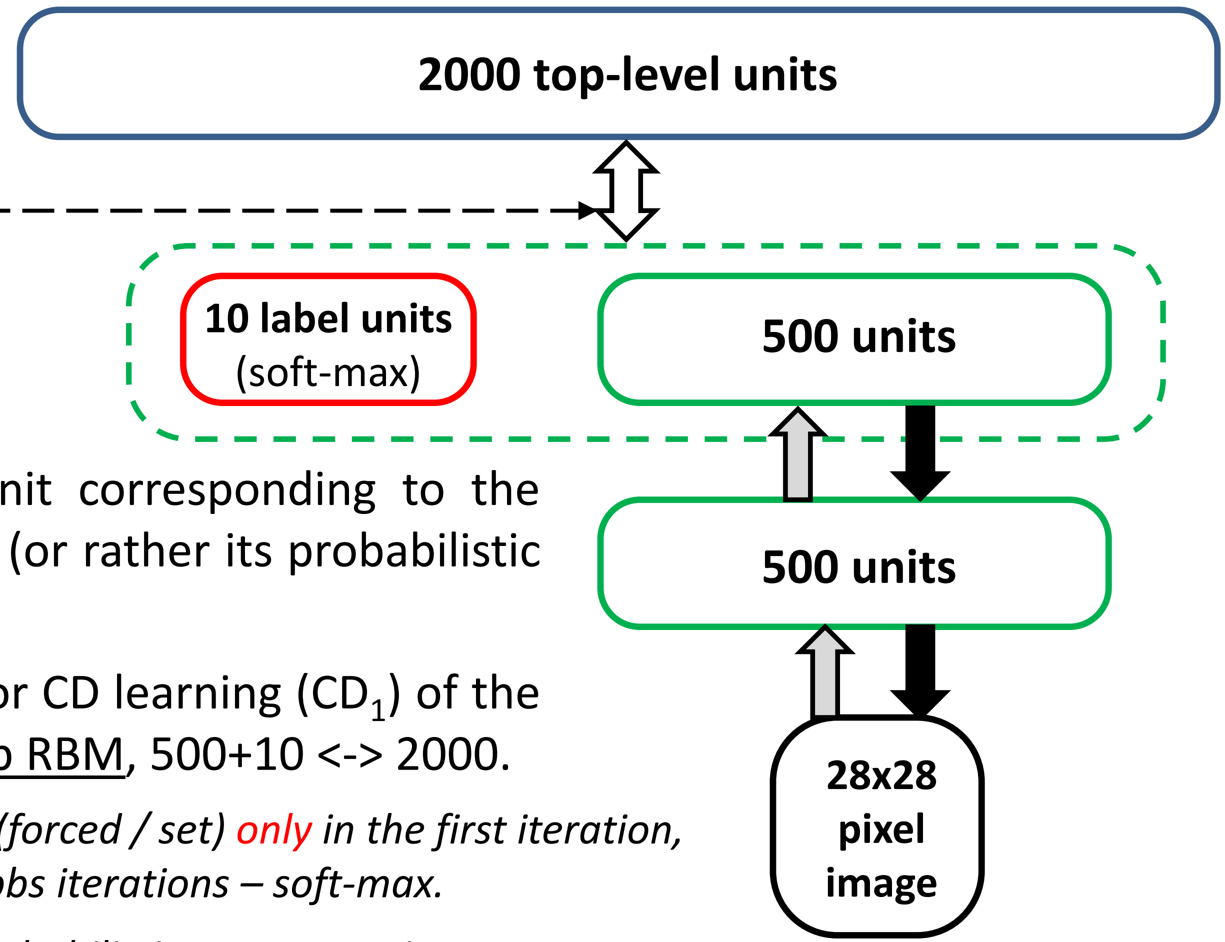
Pretraining with labels once the stack of RBMs has been built



1. Clamp a label unit corresponding to the input digit image (or rather its probabilistic representation).

Hinton et al.'s (2006) architecture

Pretraining with labels once the stack of RBMs has been built



Pretraining with labels

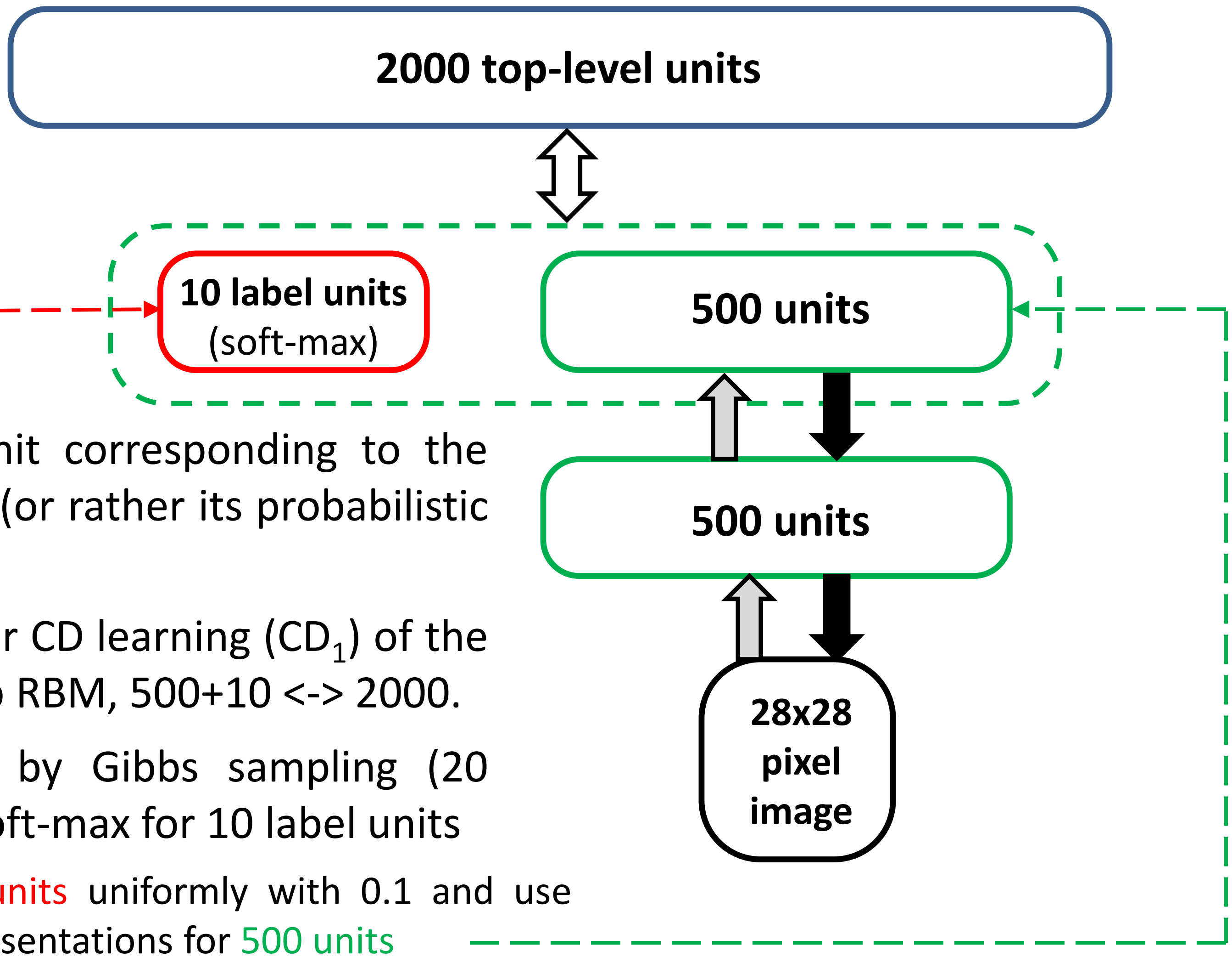
1. Clamp a label unit corresponding to the input digit image (or rather its probabilistic representation).
2. Gibbs sampling for CD learning (CD_1) of the weights in the top RBM, 500+10 <-> 2000.

* *Label units clamped (forced / set) only in the first iteration, then in subsequent Gibbs iterations – soft-max.*

* *500-unit layer is a probabilistic representation (coherently with the notion of CD_1 learning of an RBM).*

Hinton et al.'s (2006) architecture

Pretraining with labels once the stack of RBMs has been built



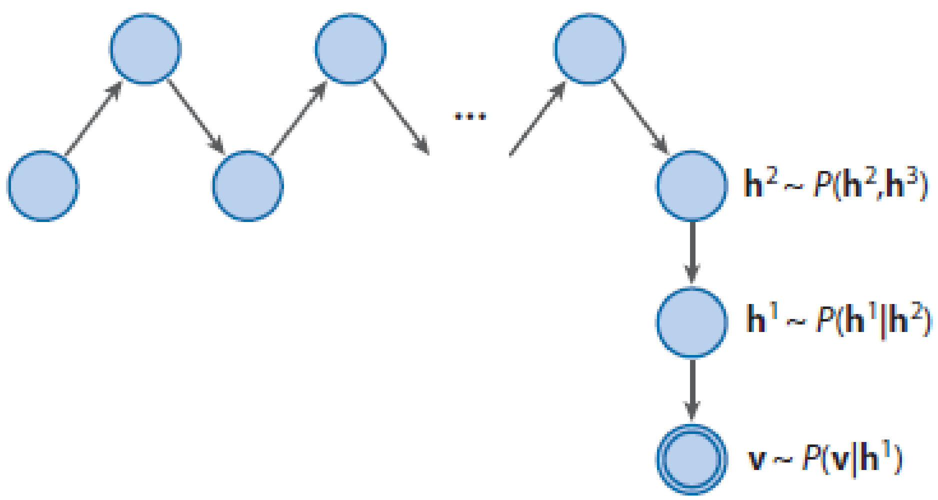
Pretraining with labels

1. Clamp a label unit corresponding to the input digit image (or rather its probabilistic representation).
2. Gibbs sampling for CD learning (CD_1) of the weights in the top RBM, 500+10 <-> 2000.
3. **Test recognition** by Gibbs sampling (20 iterations) with soft-max for 10 label units

Initialise 10 label units uniformly with 0.1 and use *binary* sample representations for 500 units

Approximate sampling from DBN

Gibbs sampling chain in the RBM part

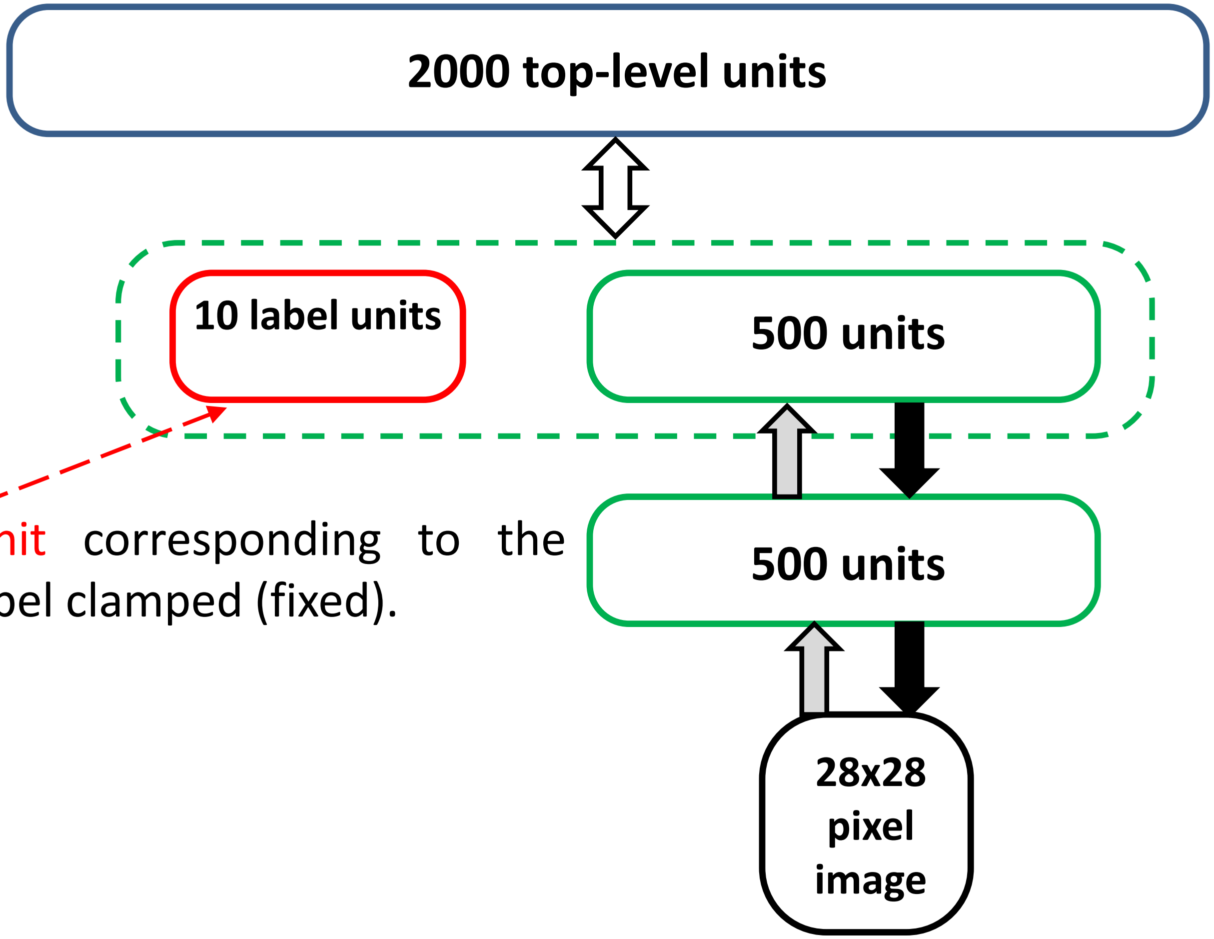


RBM part
(undirected part of the graph)

single-run sampling
(through the directed graph)

Hinton et al.'s (2006) architecture

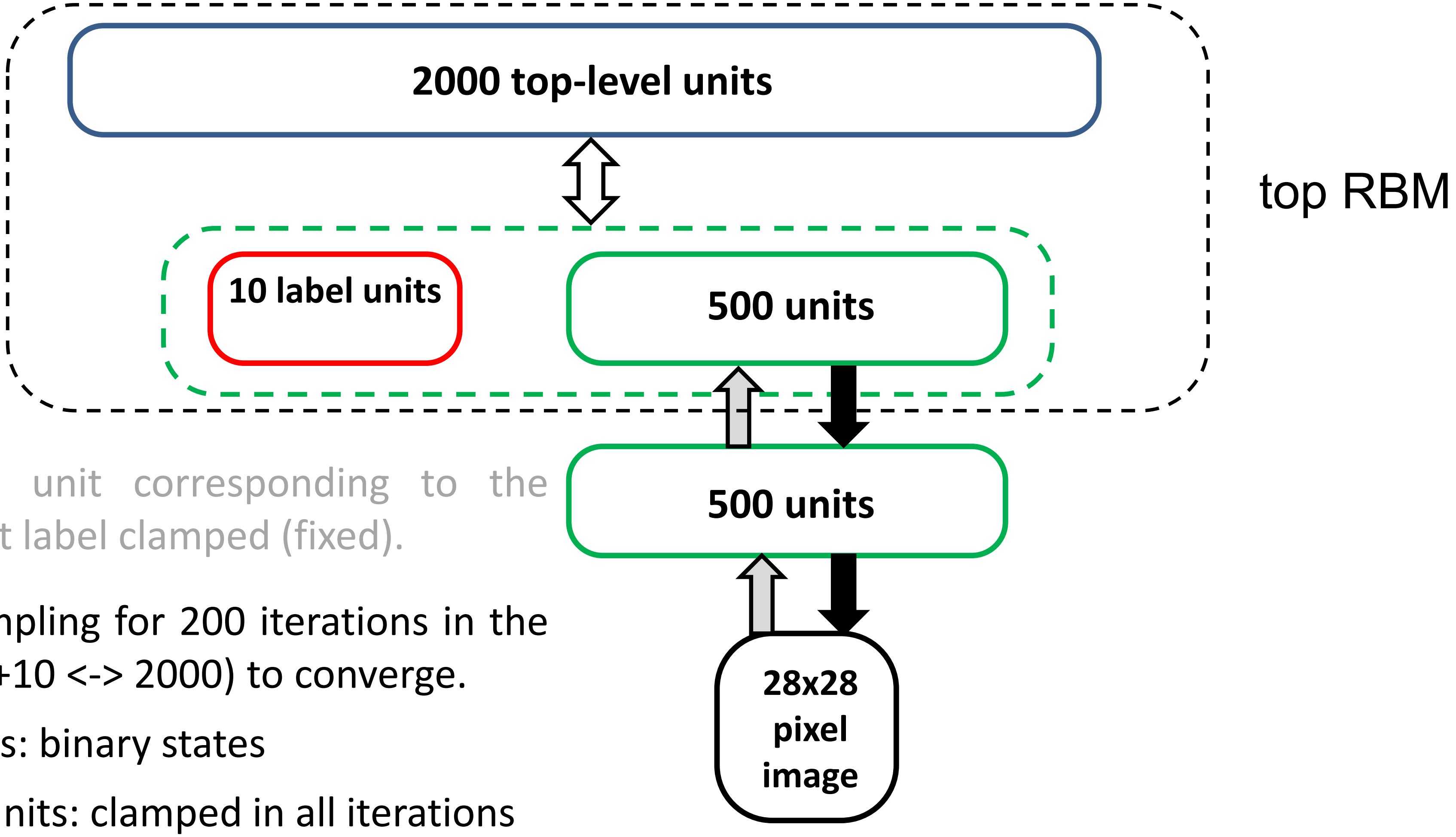
Generating samples



1. Keep a label unit corresponding to the requested digit label clamped (fixed).

Hinton et al.'s (2006) architecture

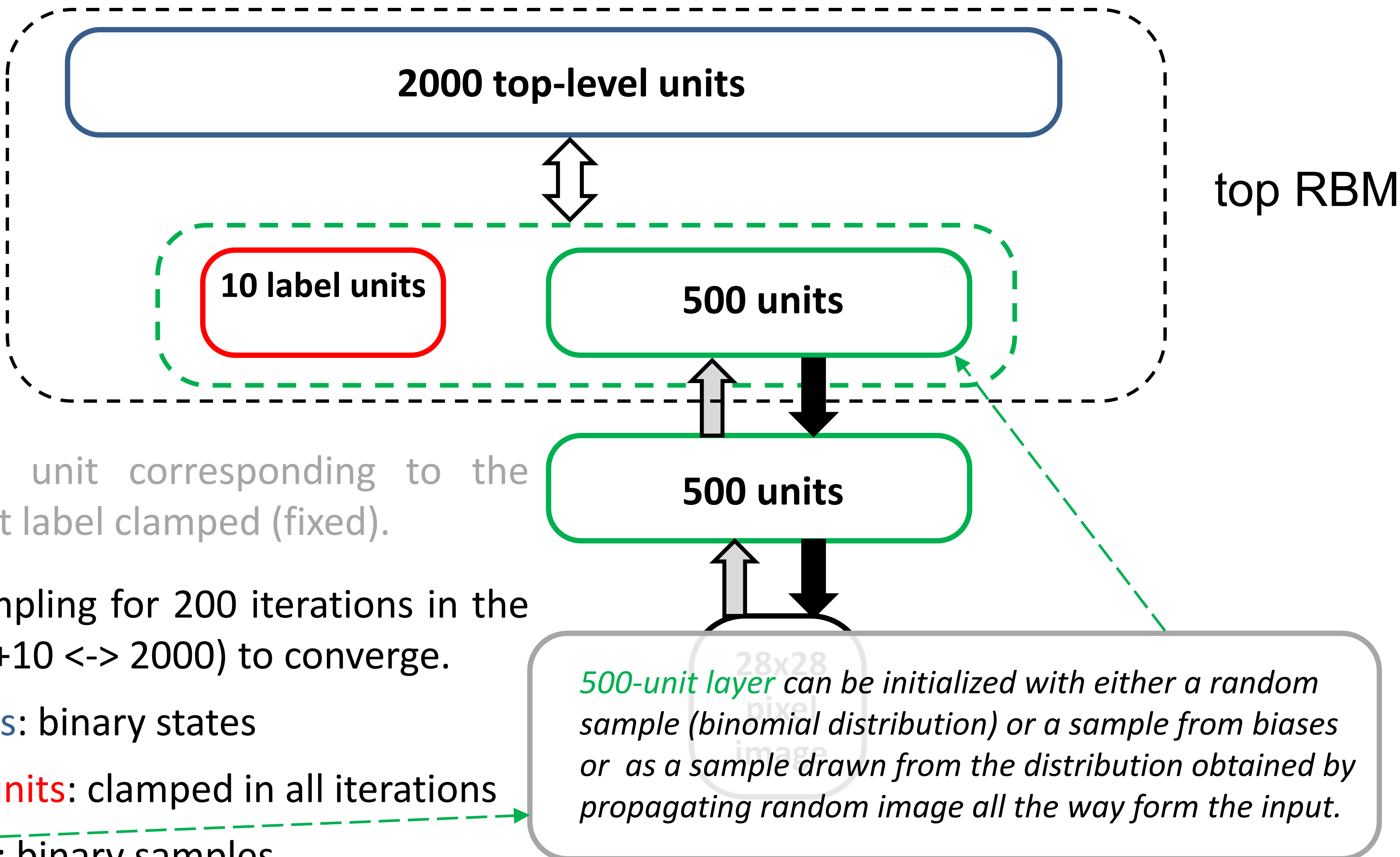
Generating samples



1. Keep a label unit corresponding to the requested digit label clamped (fixed).
2. Run Gibbs sampling for 200 iterations in the top RBM (500+10 <-> 2000) to converge.
   - 2000 units: binary states
   - 10 label units: clamped in all iterations
   - 500 units: binary samples.

Hinton et al.'s (2006) architecture

Generating samples

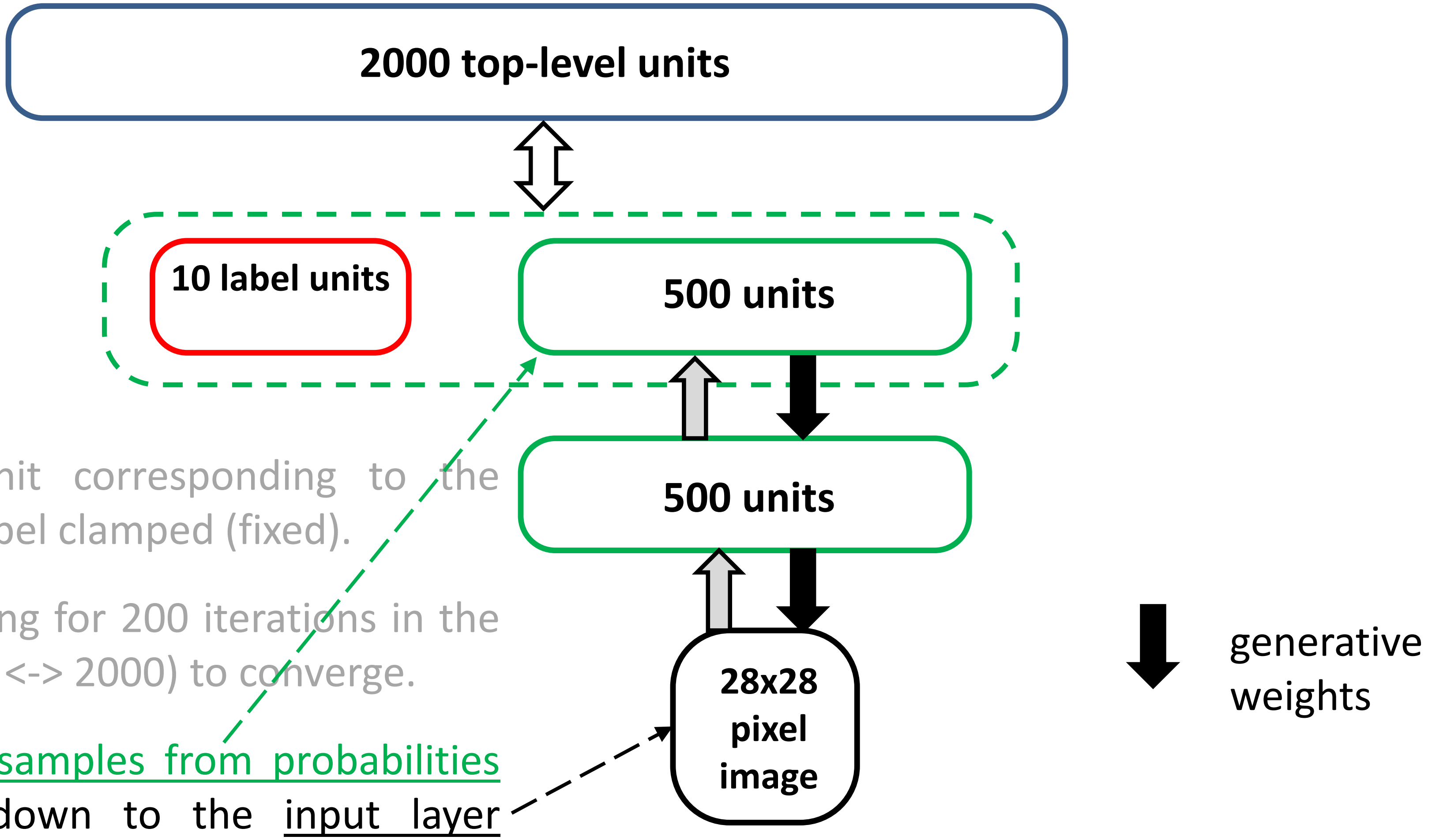


1. Keep a label unit corresponding to the requested digit label clamped (fixed).

2. Run Gibbs sampling for 200 iterations in the top RBM (500+10 <-> 2000) to converge.
   - 2000 units: binary states
   - 10 label units: clamped in all iterations
   - 500 units: binary samples.

Hinton et al.'s (2006) architecture

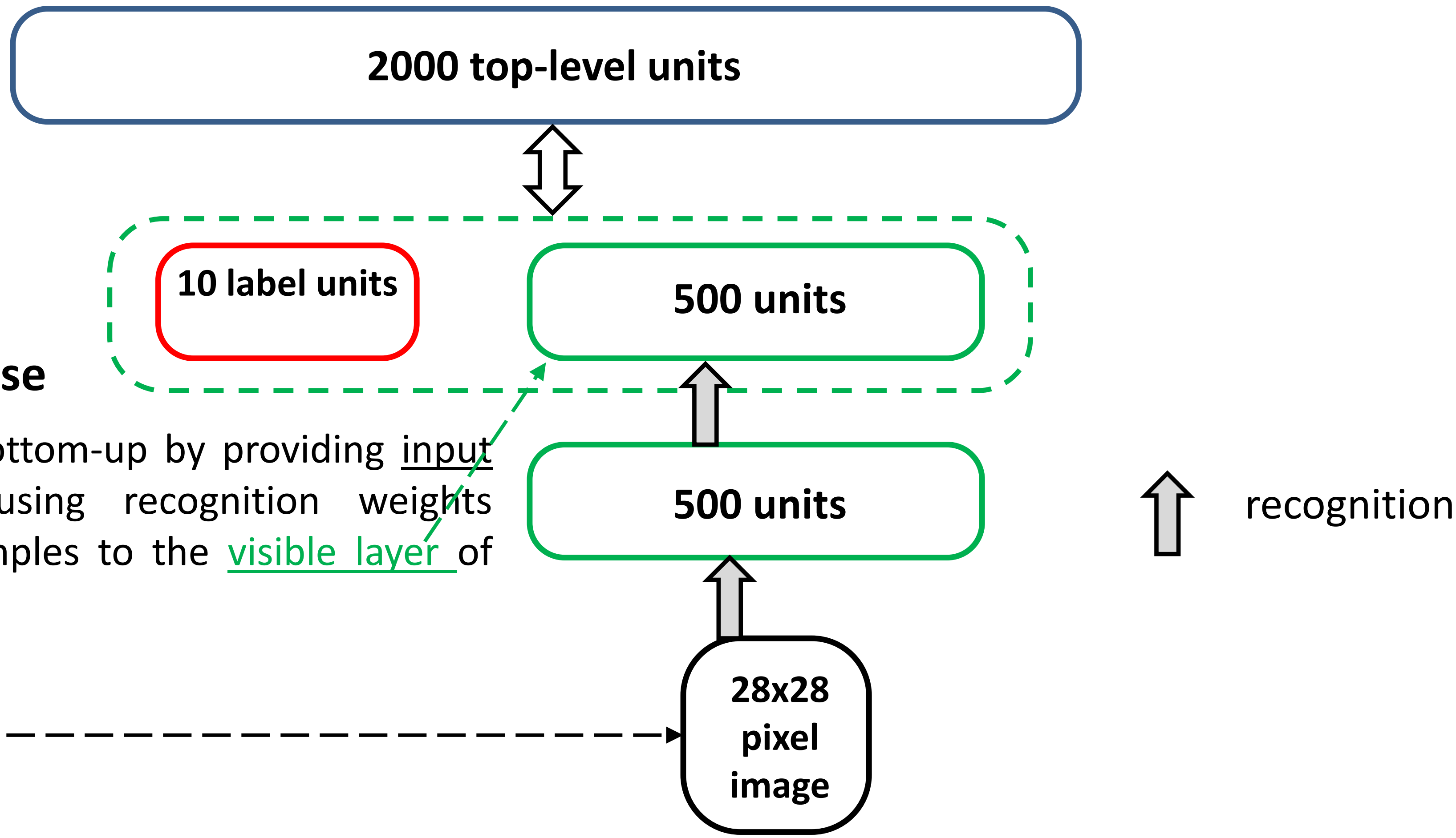
Generating samples



1. Keep a label unit corresponding to the requested digit label clamped (fixed).
2. Run Gibbs sampling for 200 iterations in the top RBM (500+10 <-> 2000) to converge.
3. Generate binary samples from probabilities and propagate down to the input layer where you can see probabs again as images.

Hinton et al.'s (2006) architecture

Fine-tuning with a contrastive wake-sleep algorithm

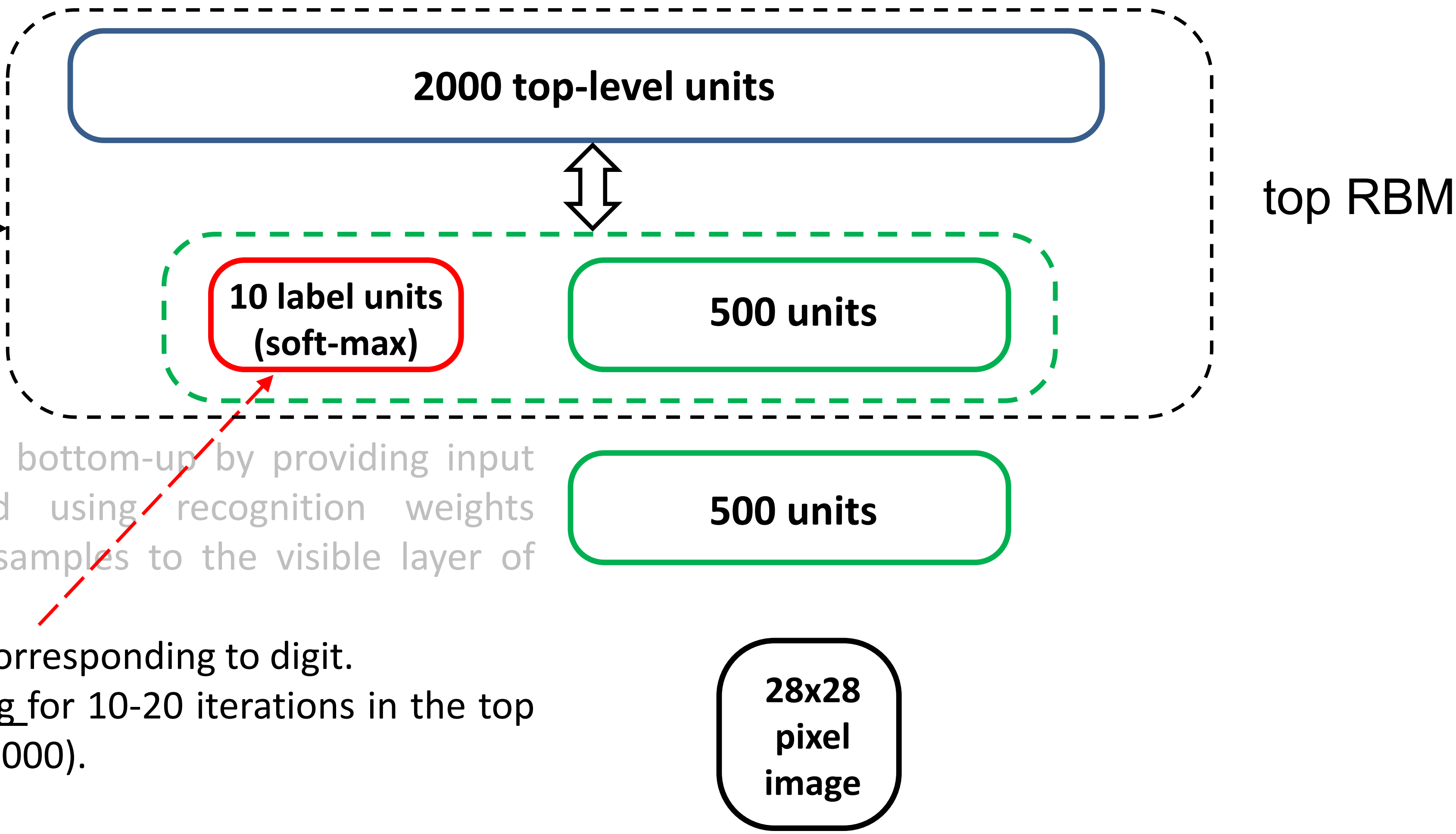


Bottom-up **wake phase**

1. Drive the network bottom-up by providing input digit images and using recognition weights propagate binary samples to the visible layer of the top RBM.

Hinton et al.'s (2006) architecture

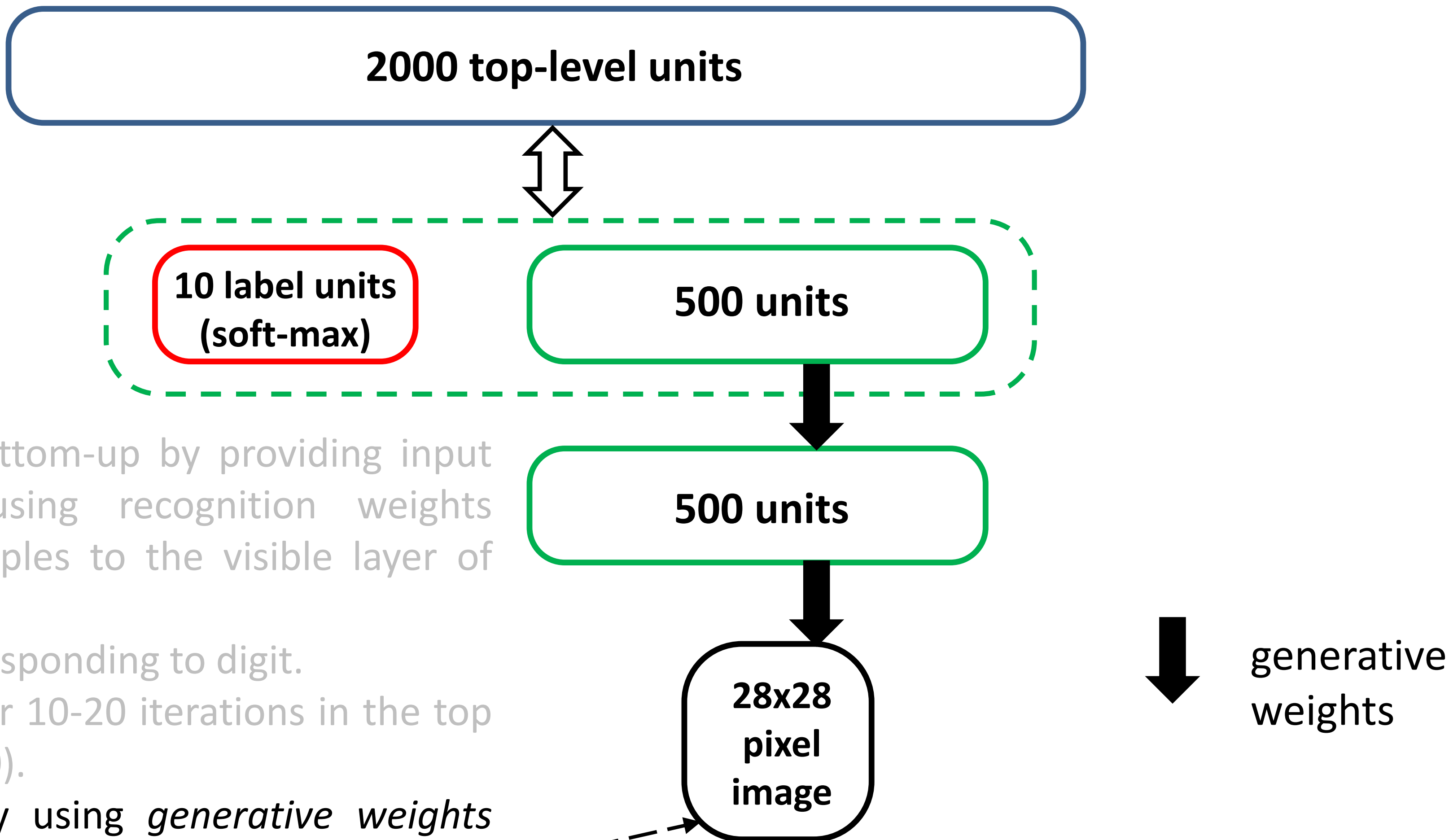
Fine-tuning with a contrastive wake-sleep algorithm



1. Drive the network bottom-up by providing input digit images and using recognition weights propagate binary samples to the visible layer of the top RBM.
2. Input a label unit corresponding to digit.
3. Run Gibbs sampling for 10-20 iterations in the top RBM (500+10 <-> 2000).

Hinton et al.'s (2006) architecture

Fine-tuning with a contrastive wake-sleep algorithm

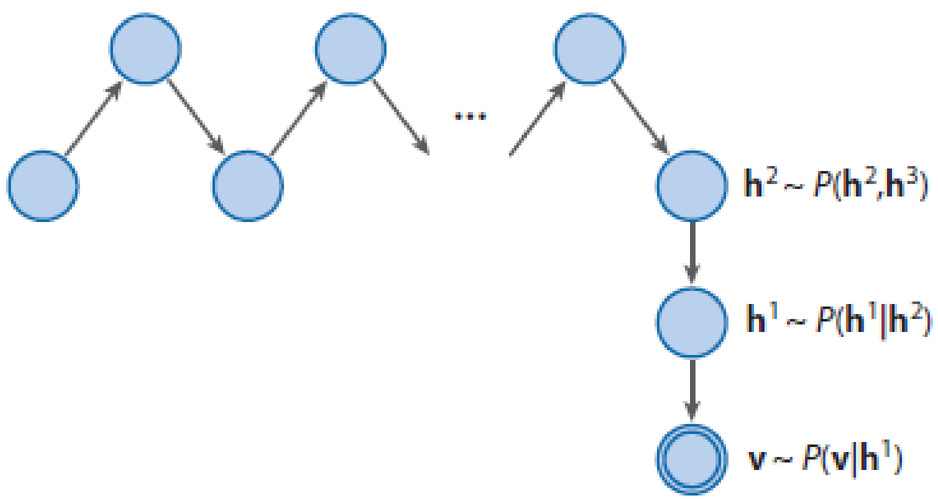


1. Drive the network bottom-up by providing input digit images and using recognition weights propagate binary samples to the visible layer of the top RBM.
2. Input a label unit corresponding to digit.
3. Run Gibbs sampling for 10-20 iterations in the top RBM (500+10 <-> 2000).
4. Propagate the activity using *generative weights* (binary sampling all the way) to the input layer represented with probabs.

Top down **sleep phase**

Approximate sampling from DBN

Gibbs sampling chain in the RBM part



RBM part
(undirected part of the graph)

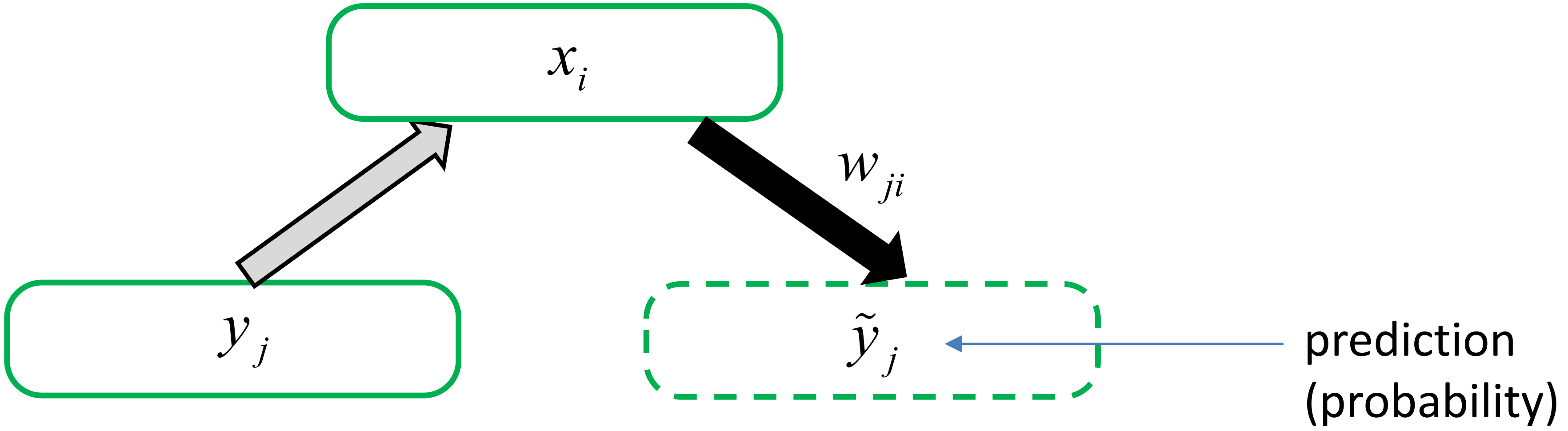
single-run sampling
(through the directed graph)

Hinton et al.'s (2006) architecture

Fine-tuning with a contrastive wake-sleep algorithm

Learning that results from the **wake phase**

(based on network activities sampled during wake phase)

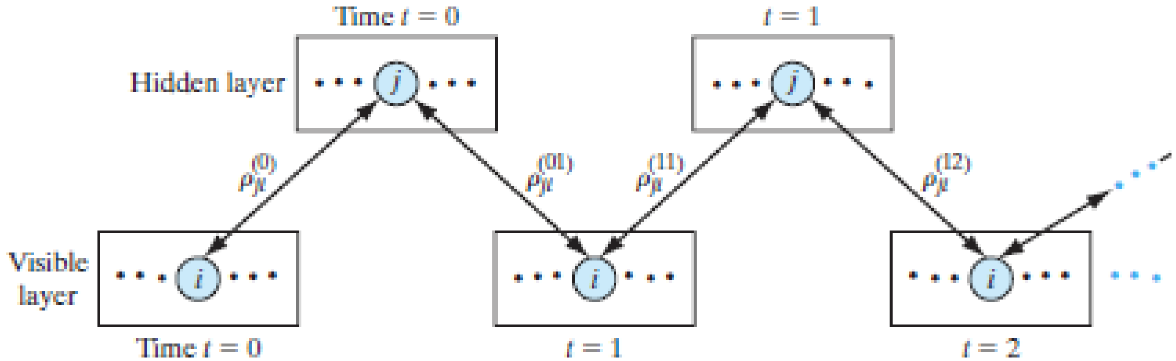


$$\Delta w_{ji} \propto x_i \left(y_j - \tilde{y}_j \right)$$

Hinton et al.'s (2006) architecture

Fine-tuning with a contrastive wake-sleep algorithm

CD$_k$ learning of the top RBM



labels are not clamped here (soft-max is used)

$$\Delta w_{j,i} \propto \left\langle v_j^{(0)} h_i^{(0)} \right\rangle - \left\langle v_j^{(k)} h_i^{(k)} \right\rangle$$

binary states

binary states

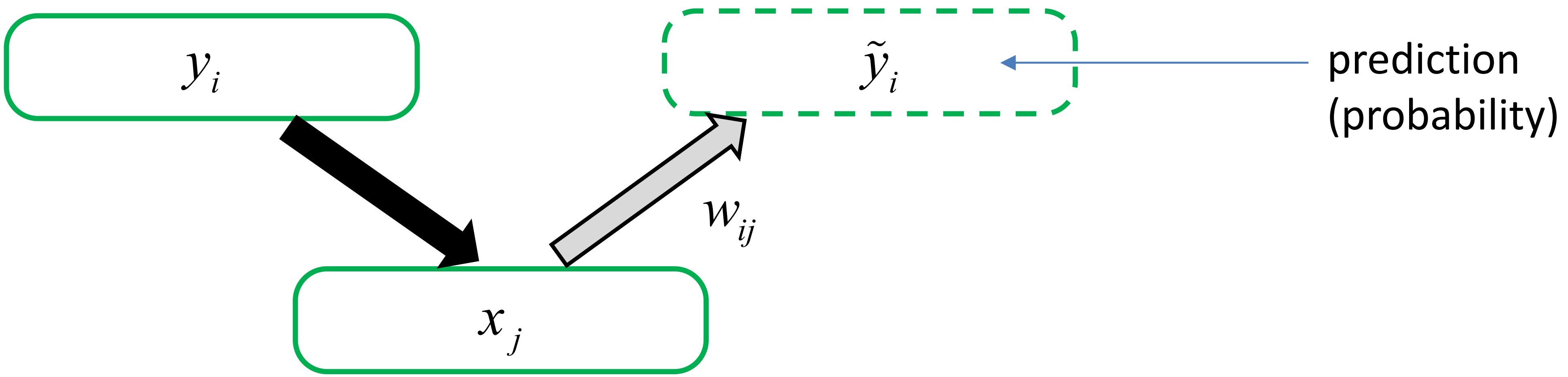
binary states
(probabilities could be used too)

Hinton et al.'s (2006) architecture

Fine-tuning with a contrastive wake-sleep algorithm

Learning that results from the **sleep phase**

(based on network activities sampled during sleep phase)



$$\Delta w_{ij} \propto x_j \left(y_i - \tilde{y}_i \right)$$