

How to deal with spurious  
minima in Hopfield networks

# Spurious minima

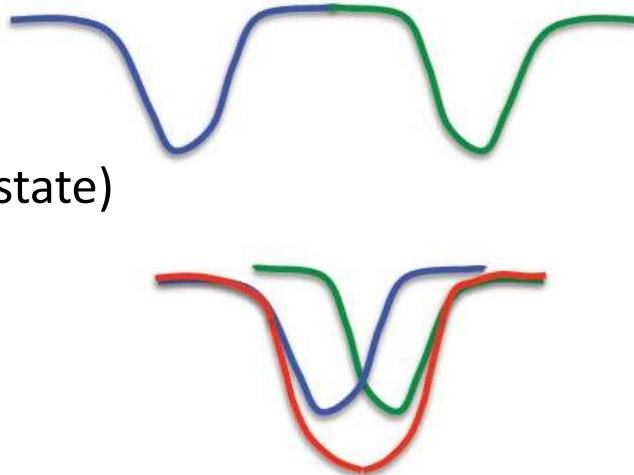
- Using Hopfield's storage rule the capacity of a totally connected net with  $N$  units is only about  $0.15N$  memories.
  - This does not make efficient use of the bits required to store the weights.

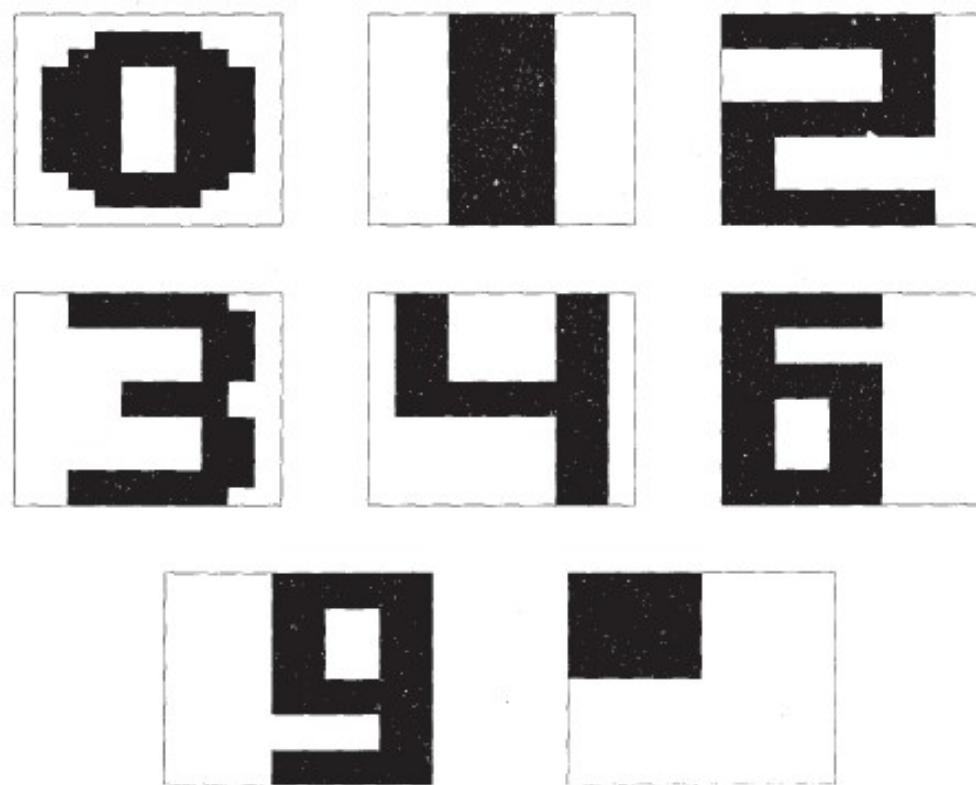
(For some slides thanks to Geoff Hinton)

# Spurious minima

- Each time we memorize a configuration, we hope to create a new energy minimum (stable state)

- But what if two nearby minima merge to create a minimum at an intermediate location?
  - This limits the capacity of a Hopfield net.





**FIGURE 14.17** Set of handcrafted patterns for computer experiment on the Hopfield network.

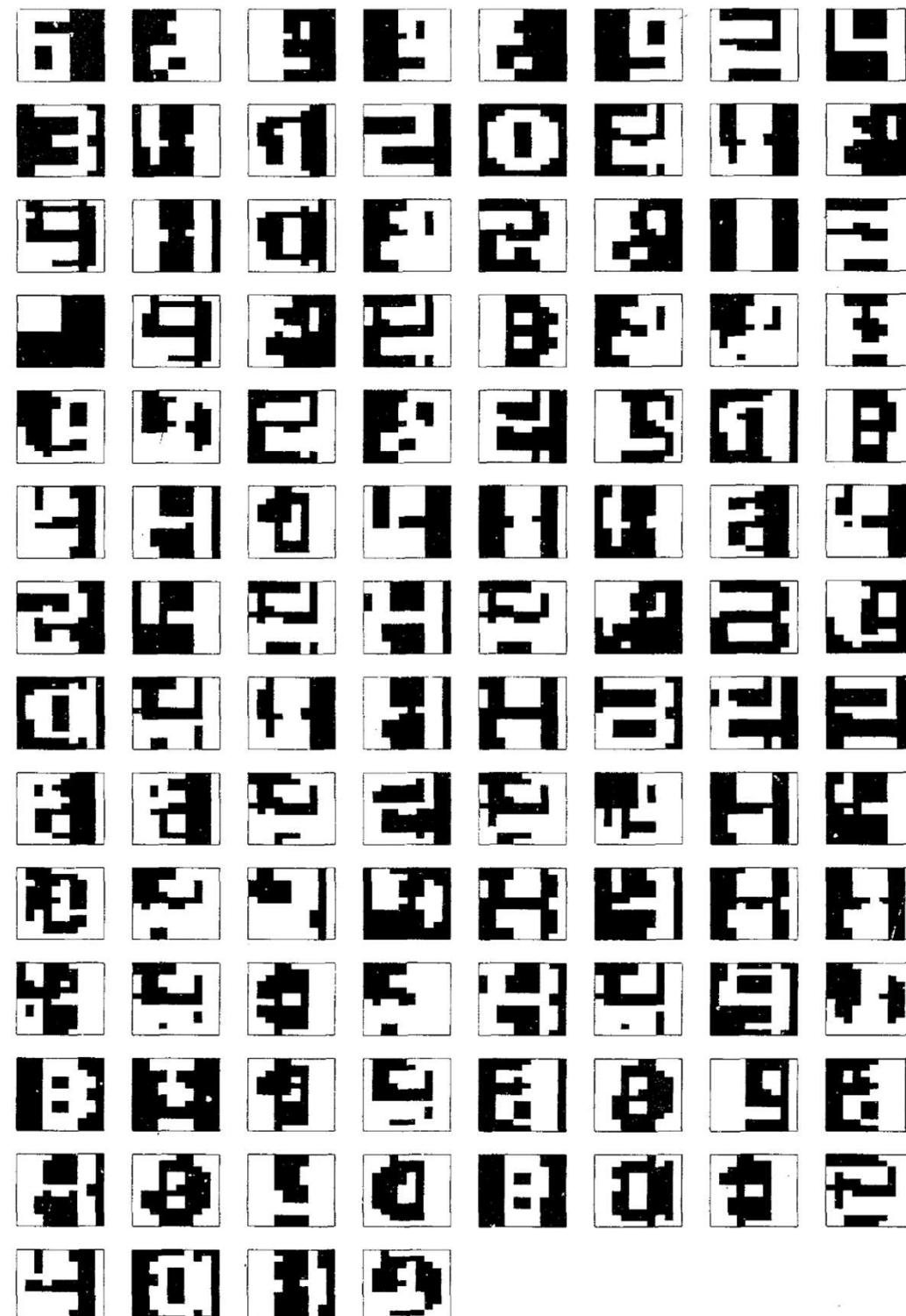


FIGURE 14.20 Compilation of the spurious states produced in the computer experiment

# Spurious minima

- Physicists have studied a lot on how to increase Hopfield storage capacity.

# Spurious minima

Hopfield, Feinstein and Palmer suggested the following strategy:

- Let the net settle from a random initial state and then do unlearning.
- This will get rid of deep, spurious minima and increase memory capacity.
- They showed that this worked.
- But they had no analysis.

## The function of dream sleep

Francis Crick\* & Graeme Mitchison\*

*We propose that the function of dream sleep (more properly rapid-eye movement or REM sleep) is to remove certain undesirable modes of interaction in networks of cells in the cerebral cortex. We postulate that this is done in REM sleep by a reverse learning mechanism (see also p.158), so that the trace in the brain of the unconscious dream is weakened, rather than strengthened, by the dream.*

MANKIND has always been fascinated by dreams. As might be expected, there have been many attempts to assign a purpose or significance to them. Although we dream

is explained in more detail below. Without it we believe that the mammalian cortex could not perform so well.

We first describe our ideas about the cor-

to go into large-amplitude instabilities<sup>5</sup>.  
Neuronal networks  
Now, if one asks what functions such richly

- Crick and Mitchison proposed unlearning as a model of what dreams are for.
  - That's why you don't remember them (unless you wake up during the dream)
- But how much unlearning should we do?
  - Can we derive unlearning as the right way to minimize some cost function?

# Towards RBM

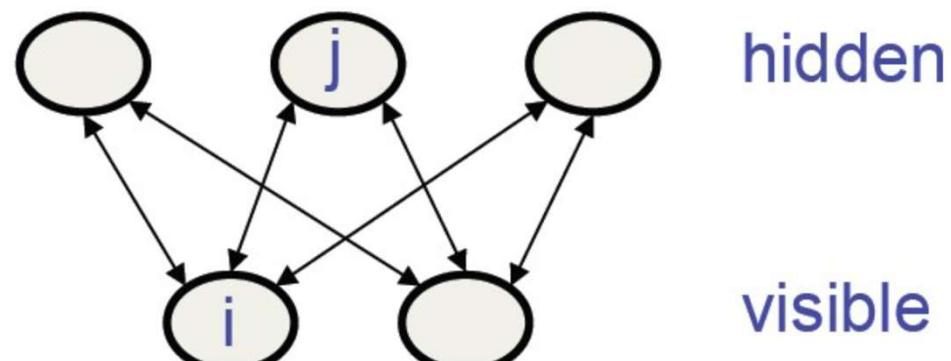
- Three ideas:
  - Visible vs. hidden units
  - Stochastic Units
  - New learning algorithm to correct errors (fantasies)



Geoff  
Hinton

# Restricted Boltzmann Machines (RBM)

- Connections only between visible and hidden units: all visible units are connected to all hidden units.
- Weights are symmetrical  $w_{ij} = w_{ji}$ .
- Possible neurons' activation states are 0 or 1.
- Neurons are stochastic.



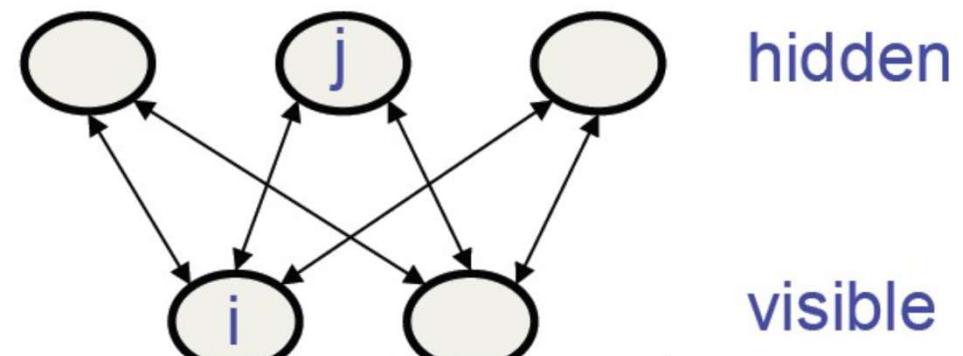
- neurons are separated in visible and hidden
- neurons are stochastic (this is the main feature of rbms) so in order to calculate the activation we need to factor in an amount of noise

Terminology: presenting an **input** to the RBM= Imposing a configuration of activations to the units in the visual layer that corresponds to the input (e.g. B/W Image).  
The RBM generates an **output** at the visible level: this is the pattern of visible units' activations.

# Visible vs. Hidden Units

## A different computational role for Hopfield nets

- Instead of using the net to store memories, use it to construct interpretations of sensory input.
  - The input is represented by the visible units.
  - The interpretation is represented by the states of the hidden units.

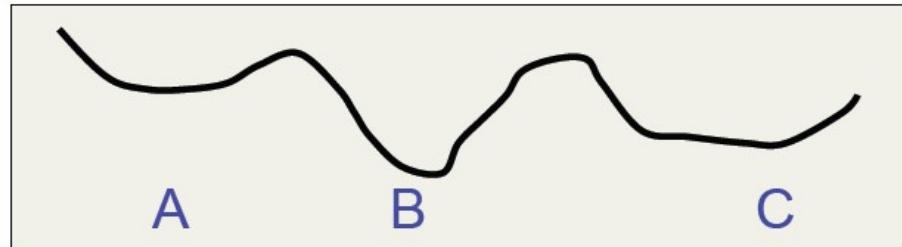


Construct interpretations: detect and represent regularities in visual activations.  
Hidden units specialize in recognising (and generating) portions of visual patterns.

# Introducing stochastic neurons

## Noisy networks find better energy minima

- A Hopfield net always makes decisions that reduce the energy.
  - This makes it impossible to escape from local minima.



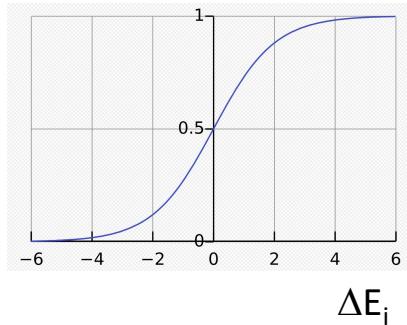
# Restricted Boltzmann Machines

## Stochastic binary units

- Replace the binary threshold units by binary stochastic units that make biased random decisions.
  - The “temperature” controls the amount of noise
  -

$$p(s_i=1) = \frac{1}{1 + e^{-\Delta E_i/T}}$$

$$p(s_i=1)$$



$$\text{Energy gap} = \Delta E_i = E(s_i = 0) - E(s_i = 1)$$

$$E = -\sum_k \sum_j w_{kj} v_k h_j$$

( T= temperature. We always take T=1, therefore we ignore it)

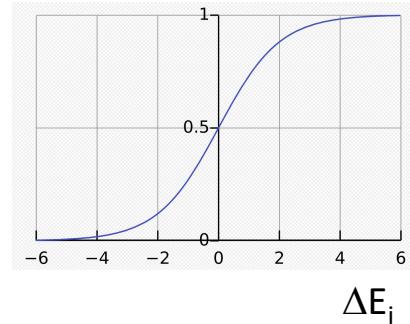
# Restricted Boltzmann Machines

## Stochastic binary units

- Replace the binary threshold units by binary stochastic units that make biased random decisions.
  - The “temperature” controls the amount of noise
  -

$$p(s_i=1) = \frac{1}{1 + e^{-\Delta E_i/T}}$$

$p(s_i=1)$



$$\text{Energy gap} = \Delta E_i = E(s_i = 0) - E(s_i = 1)$$

$$E = -\sum_k \sum_j w_{kj} v_k h_j$$

$$\Delta E_i = \sum w_{ij} s_j \text{ for all } j \text{ connected to } i$$

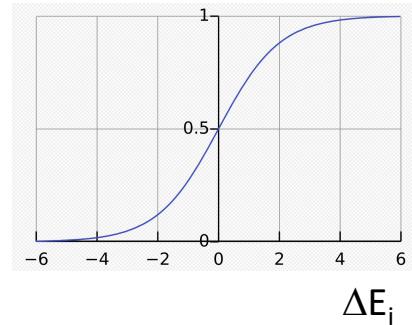
# Restricted Boltzmann Machines

## Stochastic binary units

- Replace the binary threshold units by binary stochastic units that make biased random decisions.
  - The “temperature” controls the amount of noise
  -

$$p(s_i=1) = \frac{1}{1 + e^{-\Delta E_i/T}}$$

$$p(s_i=1)$$



$$\text{Energy gap} = \Delta E_i = E(s_i = 0) - E(s_i = 1)$$

$$E = -\sum_k \sum_j w_{kj} v_k h_j$$

$$\Delta E_i = \sum w_{ij} s_j \text{ for all } j \text{ connected to } i$$

NOTATION:  $s_i$  = activation state (0 or 1) of the generic neuron  $i$   
( $v_i$  = as  $s_i$  limited to visible units and  $h_j$  = as  $s_i$  limited to hidden units)

# Restricted Boltzmann Machines

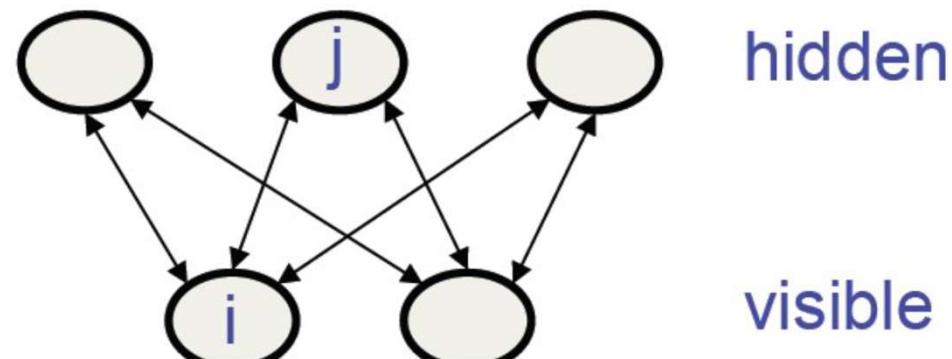
## Sampling

Once calculated that  $P(h_1=1)$  (or a distribution of probabilities for several units considered in parallel )

**Sampling** is used to attribute a state (0 or 1) using the probability

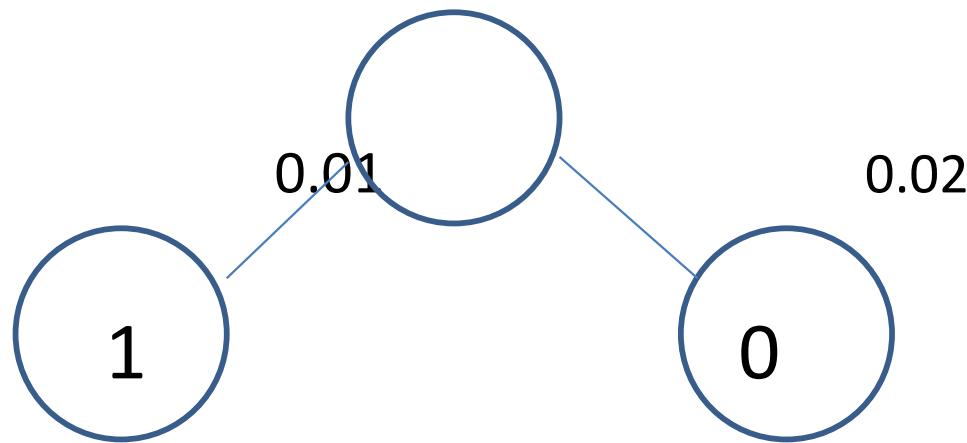
# Restricted Boltzmann Machines

- Use:
  - present a visual input.
  - Calculate activation at the hidden level
  - Recalculate activation at the visual level
  - Possibly, start again
  - .....



# Exercise

- Calculate  $p(h_1=1)$  ( $e^{-0.01}=0.99$ )



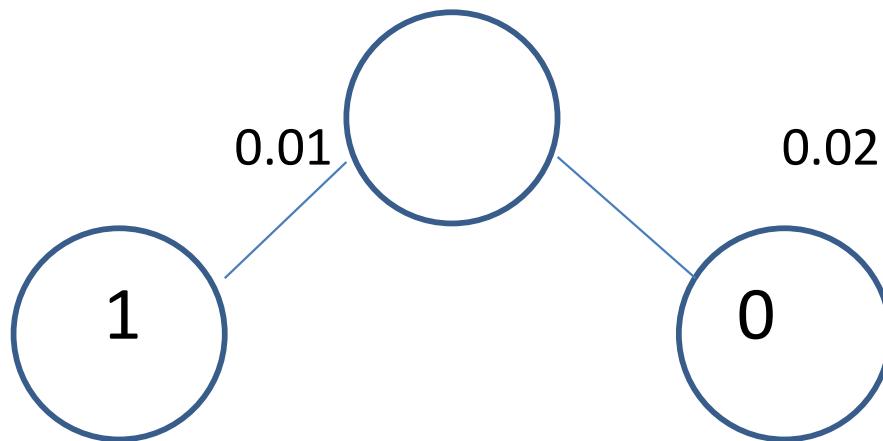
---

$$p(s_i=1) = \frac{1}{1 + e^{-\Delta E_i/T}}$$

$$\Delta E_i = \sum w_{ij} s_j$$

# Exercise- Solution

- Calculate  $p(h_1=1)$  ( $e^{-0.01}=0.99$ )



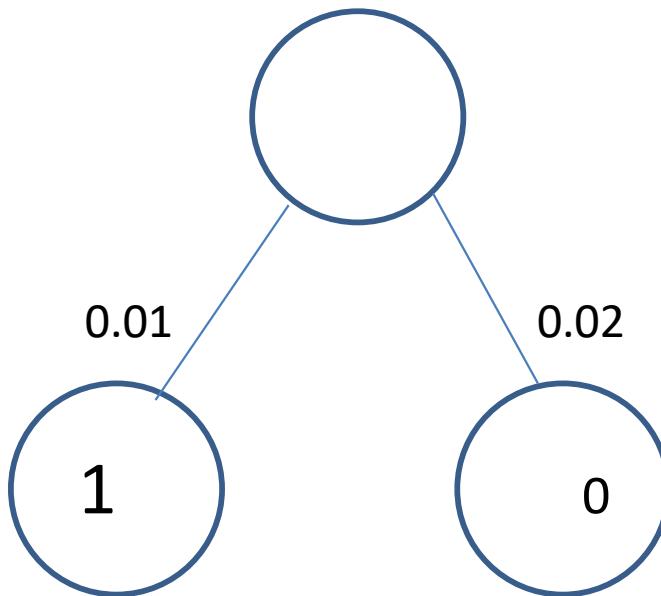
---

$$p(s_i=1) = \frac{1}{1 + e^{-\Delta E_i/T}}$$

$$P(h_1=1)=0.5025$$

$$\Delta E_i = \sum w_{ij} s_j$$

# Sampling

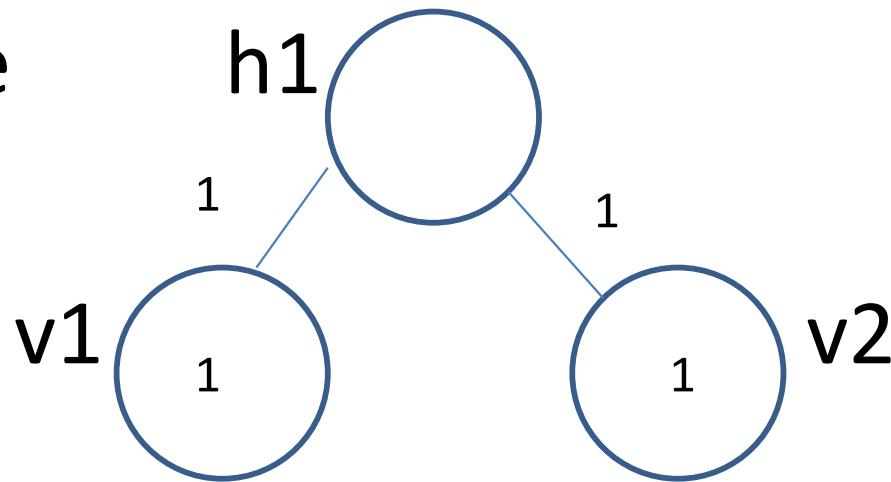


Once calculated that  $P(h_1=1)=0.5025$  (or a distribution for several units considered in parallel )

**Sampling** is used to attribute a state (0 or 1) using the probability

# Hidden units detect and force visual patterns

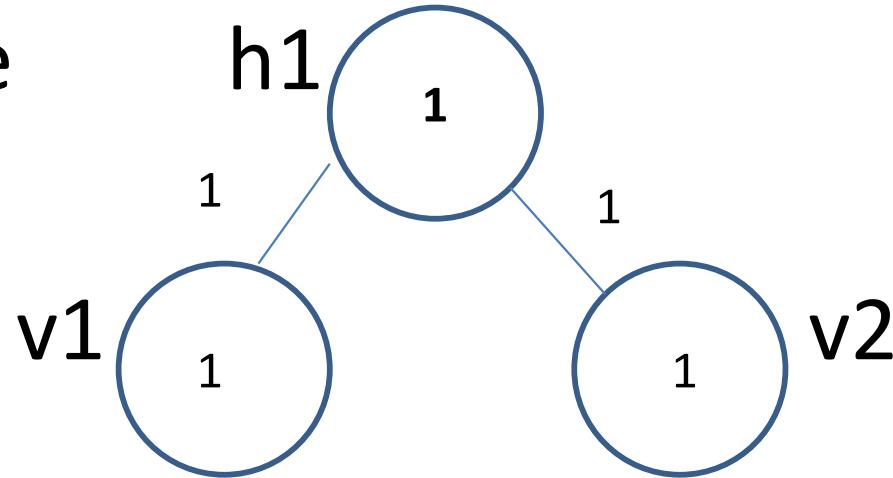
- Example



Given these weights, if  $v_1=1$  and  $v_2=1$ ,  $P(h_1=1) = 1/(1+e^{-\Delta E_i}) = 1/(1+e^{-2})=0.88$

# Hidden units detect and force visual patterns

- Example

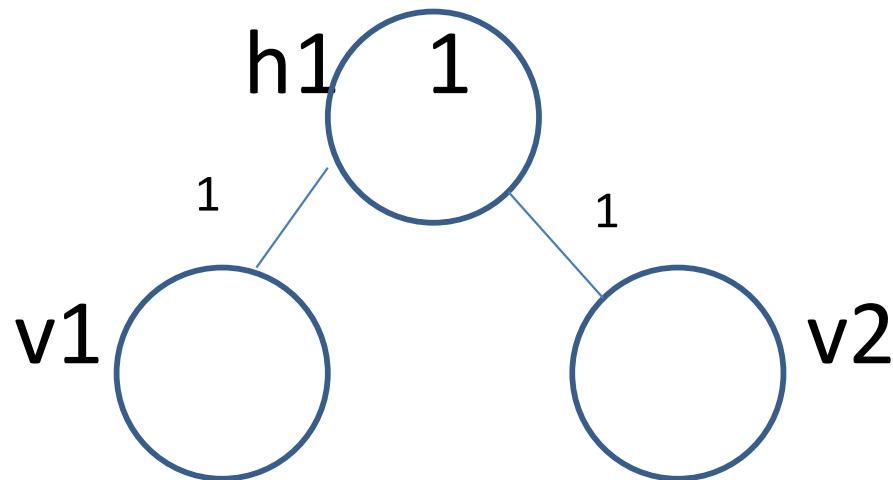


Given these weights, if  $v_1=1$  and  $v_2=1$ ,  $P(h_1=1) = 1/(1+e^{-\Delta E_i}) = 1/(1+e^{-2})=0.88$

By sampling, with probability 0.88,  $h_1=1$  (the activation of  $h_1$  will be set to 1)

# Hidden units detect and force visual patterns

- And



From  $h_1 = 1$  we can start again and recalculate  $v_1$  and  $v_2$ :

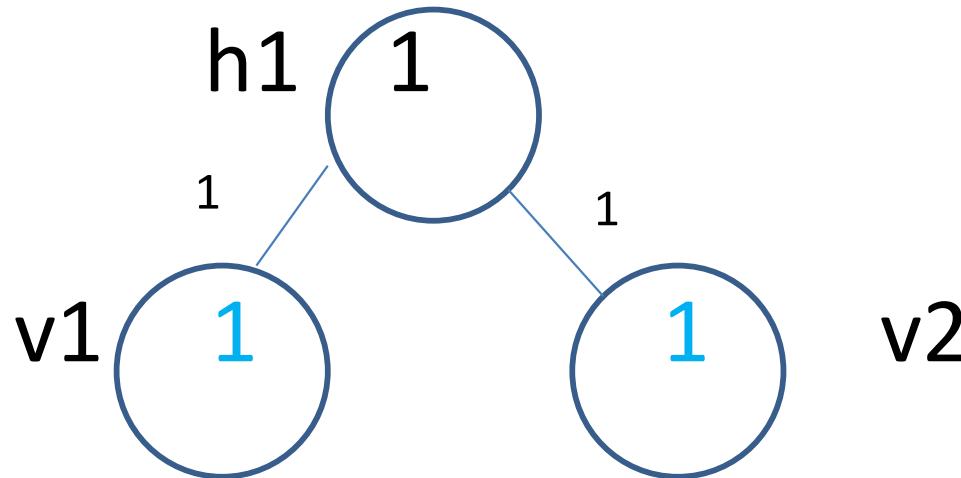
$$P(v_1=1) = 1/(1+e^{-\Delta E_i}) = 1/(1+e^{-1}) = 0.73$$

$$P(v_2=1) = 1/(1+e^{-\Delta E_i}) = 1/(1+e^{-1}) = 0.73$$

With sampling high probability of having  $v_1=1$  and  $v_2=1$

# Hidden units detect and force visual patterns

- And



From  $h_1 = 1$  we can start again and recalculate  $v_1$  and  $v_2$ :

$$P(v_1=1) = 1/(1+e^{-\Delta E_i}) = 1/(1+e^{-1}) = 0.73$$

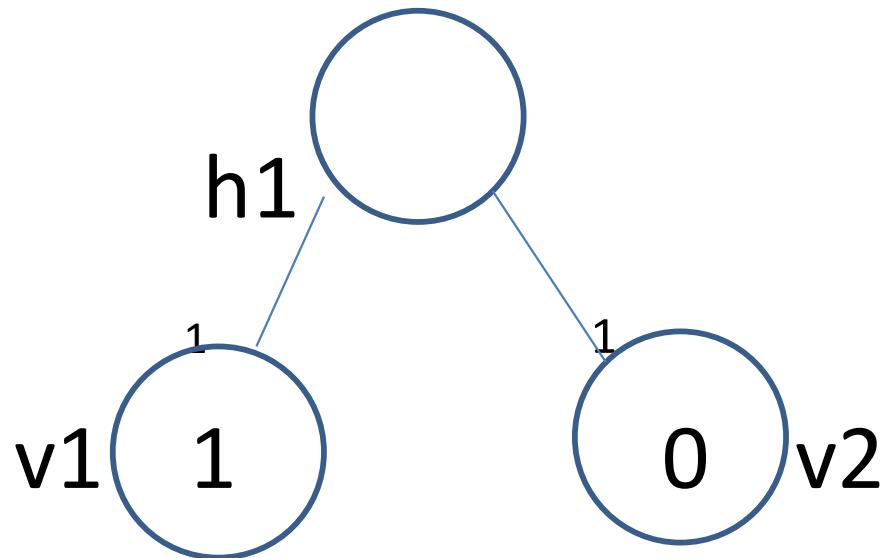
$$P(v_2=1) = 1/(1+e^{-\Delta E_i}) = 1/(1+e^{-1}) = 0.73$$

With sampling high probability of having  $v_1=1$  and  $v_2=1$

→ Hence,  $h_1$  has high probabilities of activating when  $v_1$  and  $v_2$  are 1. Once activated,  $h_1$  forces back the activation of  $v_1$  and  $v_2 = 1$ !

# Hidden units detect and force visual patterns

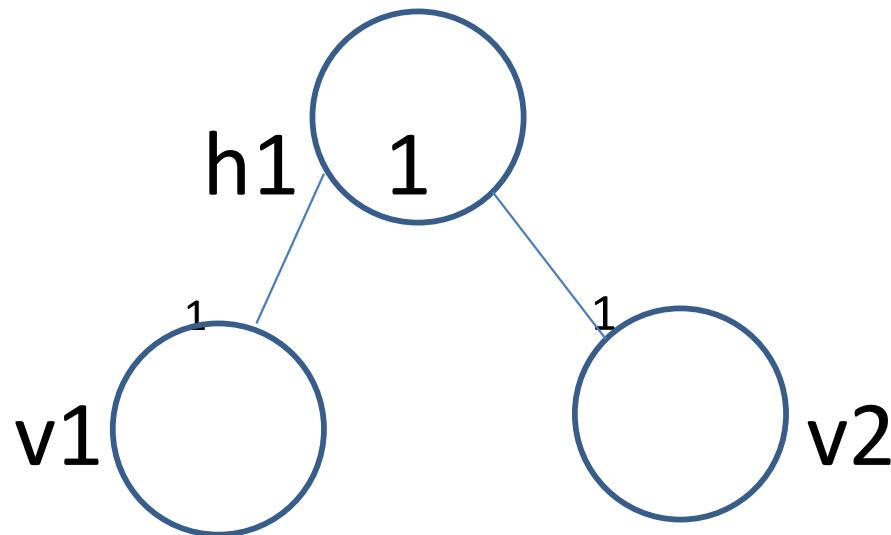
- It also works with different (or corrupted) patterns!
- Input in this case is  $v_1=1$  but  $v_2=0$



$$P(h_1=1) = 1/(1+e^{-\Delta E_i}) = 1/(1+e^{-1}) = 0.73 \text{ Let's assume by sampling } h_1=1$$

# Hidden units detect and force visual patterns

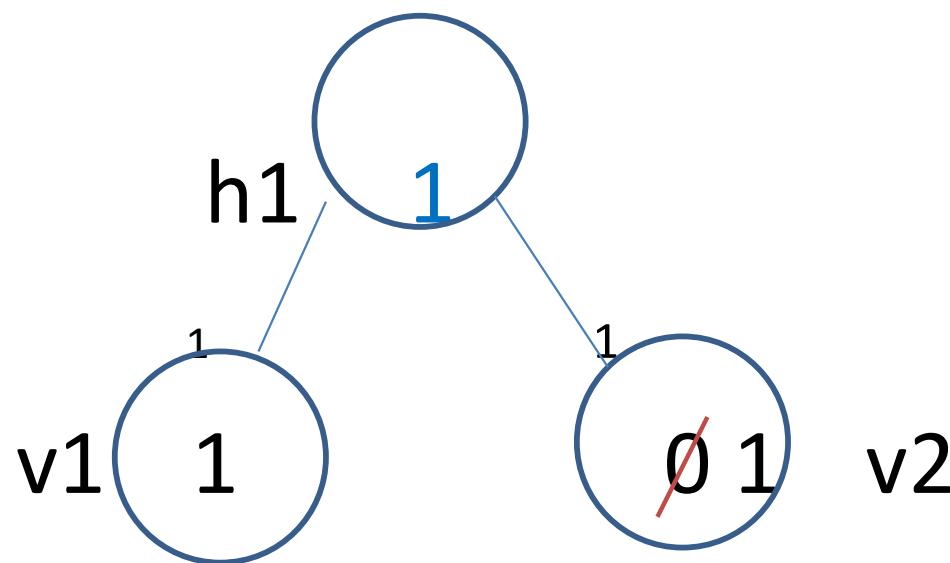
- It also works with different (or corrupted) patterns!
- Input in this case is  $v_1=1$  but  $v_2=0$



$$P(h_1=1) = 1/(1+e^{-\Delta E_i}) = 1/(1+e^{-1}) = 0.73 \text{ Let's assume by sampling } h_1=1$$

# Hidden units detect and force visual patterns

- It also works with different (or corrupted) patterns!
- Input in this case is  $v_1=1$  but  $v_2=0$



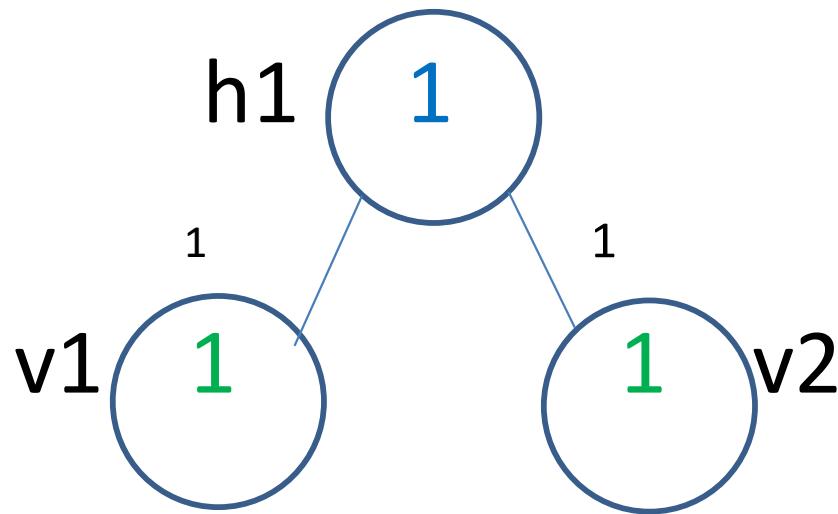
If I recalculate visual activations:

$$P(v_1=1) = 1/(1+e^{-\Delta E_i}) = 1/(1+e^{-1}) = 0.73$$

$$P(v_2=1) = 1/(1+e^{-\Delta E_i}) = 1/(1+e^{-1}) = 0.73$$

→ With high probability after sampling  $v_1=1$  and  $v_2=1$ , i.e. the original pattern [1,0] has been «corrected» as a consequence of h1's action

# Pattern generation



RBM are generative models: they are used to generate visual states.

The process of generating visual states usually requires several steps (calculate visual states, then hidden, then visual again...) until equilibrium, i.e. a state in which there are no big changes in probability distribution.

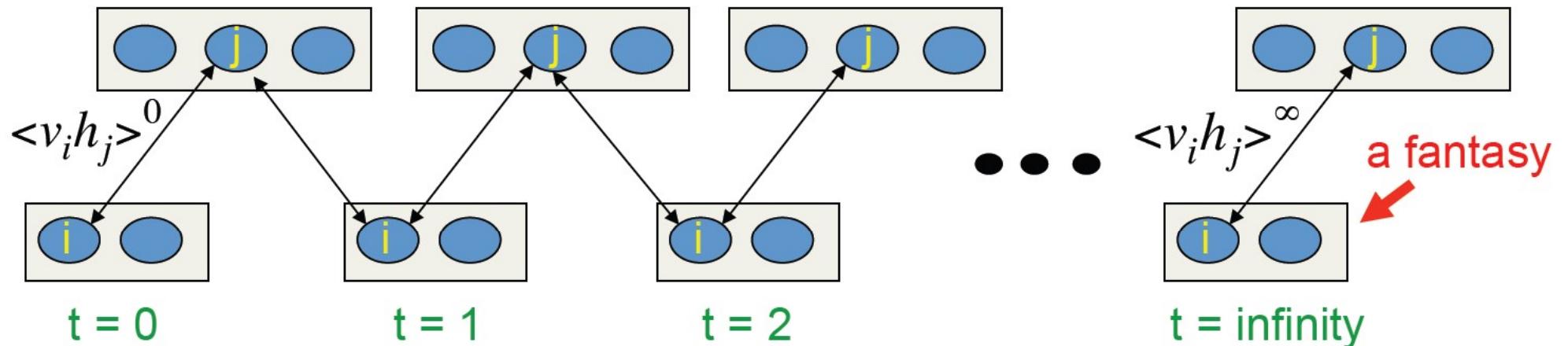
Restricted Boltzmann Machines are Generative Models:  
they learn to Reproduce at the visual level the training set!

## The goal of learning

We want to maximize the probability of obtaining at the visible level the vectors of the training set

Training set= set of visible vectors that we want the RBM to learn (in order to reconstruct or generate)

# A picture of an inefficient version of the Boltzmann machine learning algorithm for an RBM

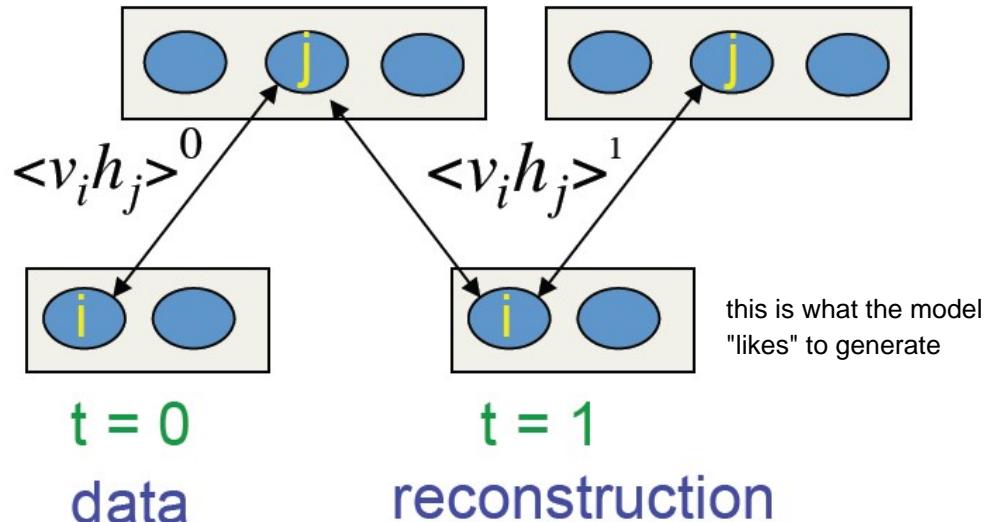


Start with a training vector on the visible units. Then alternate between updating all the hidden units in parallel and updating all the visible units in parallel.

$$\Delta w_{ij} = \varepsilon (\langle v_i h_j \rangle^0 - \langle v_i h_j \rangle^\infty)$$

Contrastive Divergence is the Learning Algorithm for RBM:

## Contrastive divergence: A very surprising short-cut



$$\Delta w_{ij} = \varepsilon (\langle v_i h_j \rangle^0 - \langle v_i h_j \rangle^1)$$

Start with a training vector on the visible units.

Update all the hidden units in parallel.

Update the all the visible units in parallel to get a “reconstruction”.

Update the hidden units again.

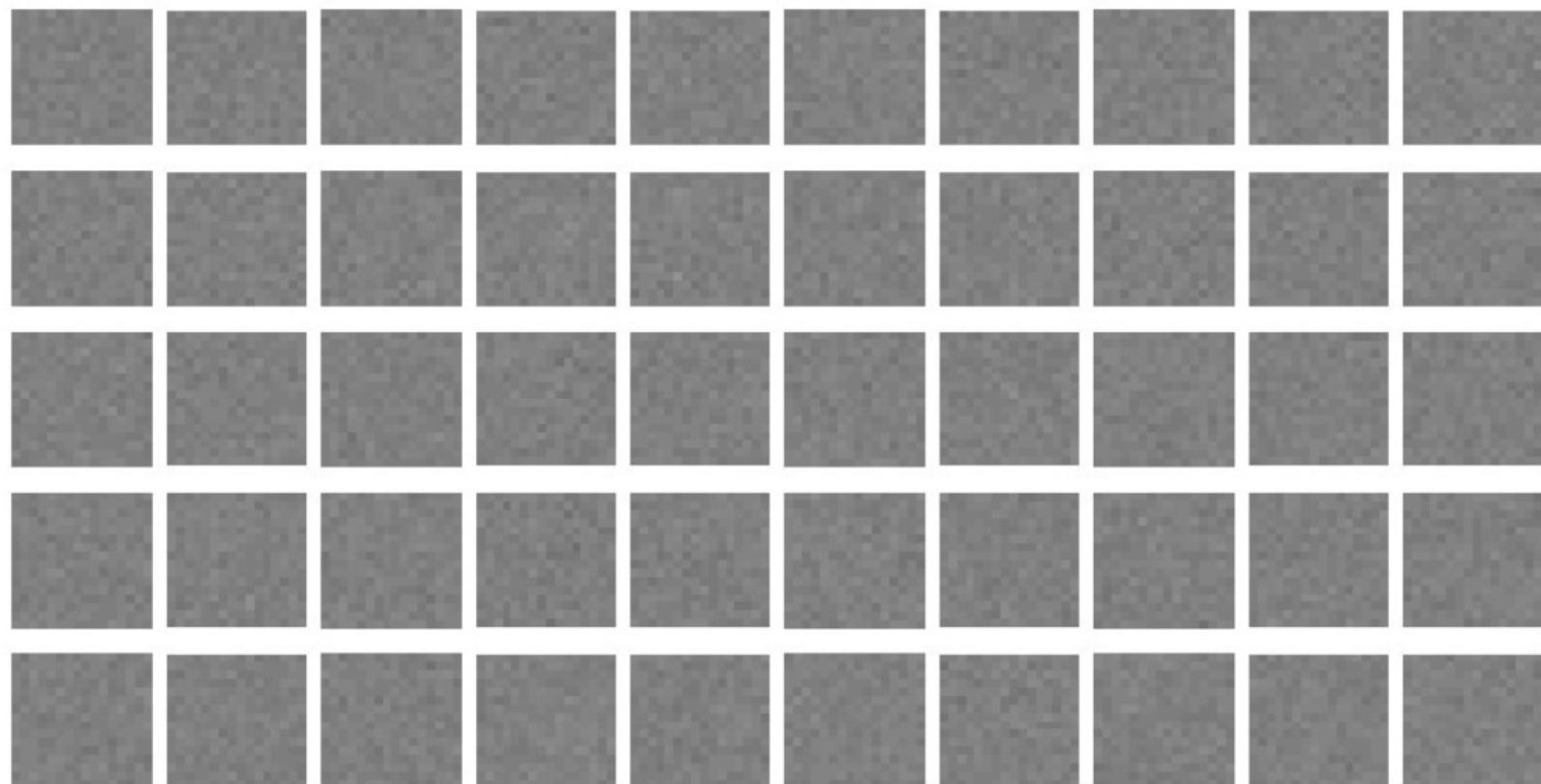
$v_i$  and  $h_j$  are the activations of visual unit  $i$  e hidden unit  $j$ . Apix 0 and 1 refer to the step:  $^0$  at  $t=0$ ,  $^1$  at  $t=1$

- Hidden units specialize in recognising (and generating) portions of patterns.

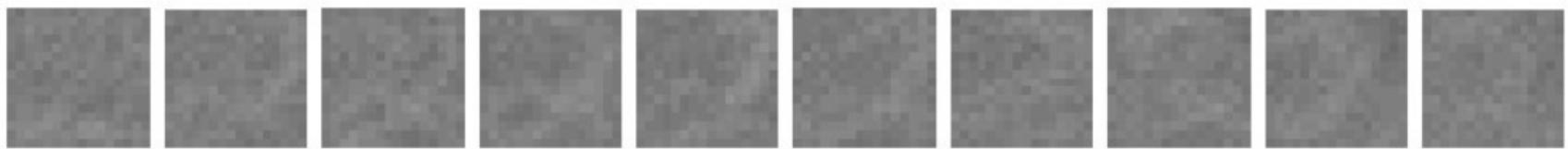
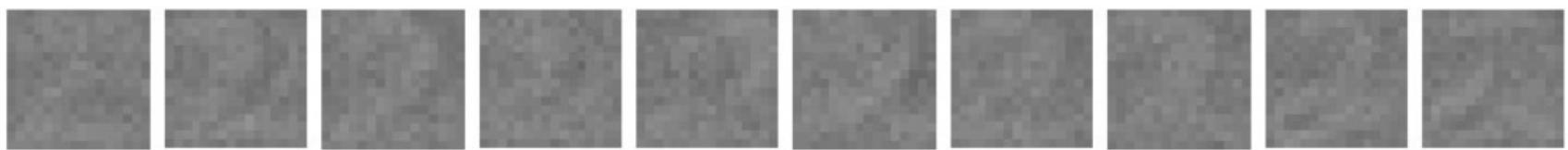
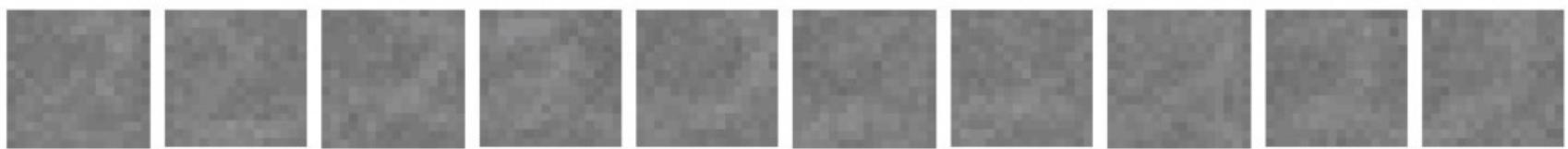
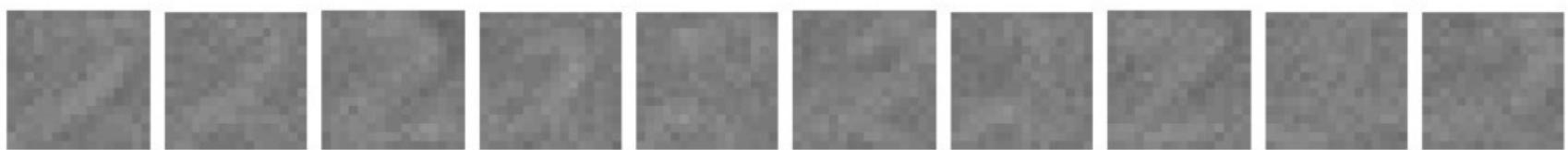
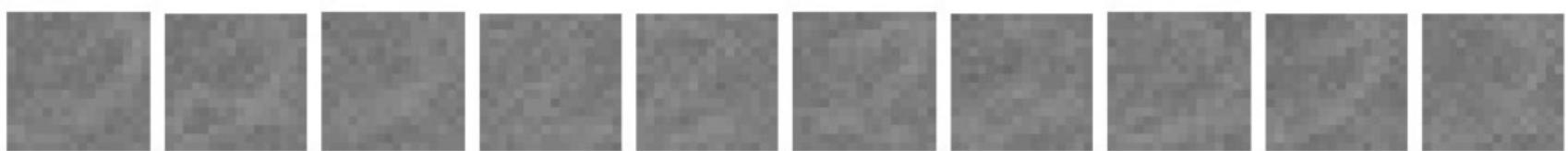
# Importance of weights in RBM

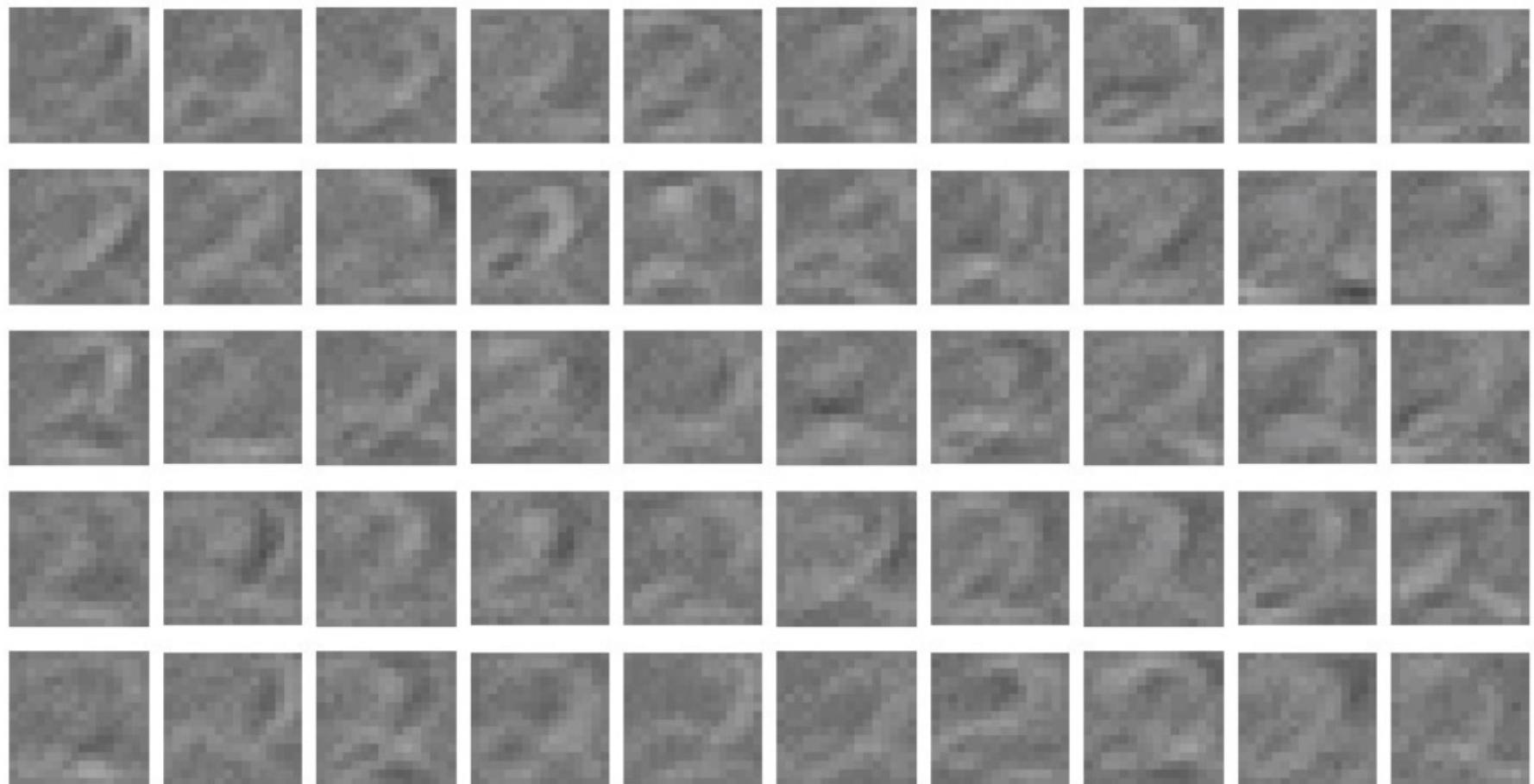
- The probability of unit  $i$  to get state 1 depends on the weights to  $i$ .
- Roughly: positive weight  $w_{ij}$  augments the impact of neuron  $j$  to  $p(s_i=1)$
- By observing weight matrices to hidden units, we can understand which input values augment their probability to activate ( $=1$ )
- This is the topic of our lab: we will use a RBM, and observe the hidden units' matrices.

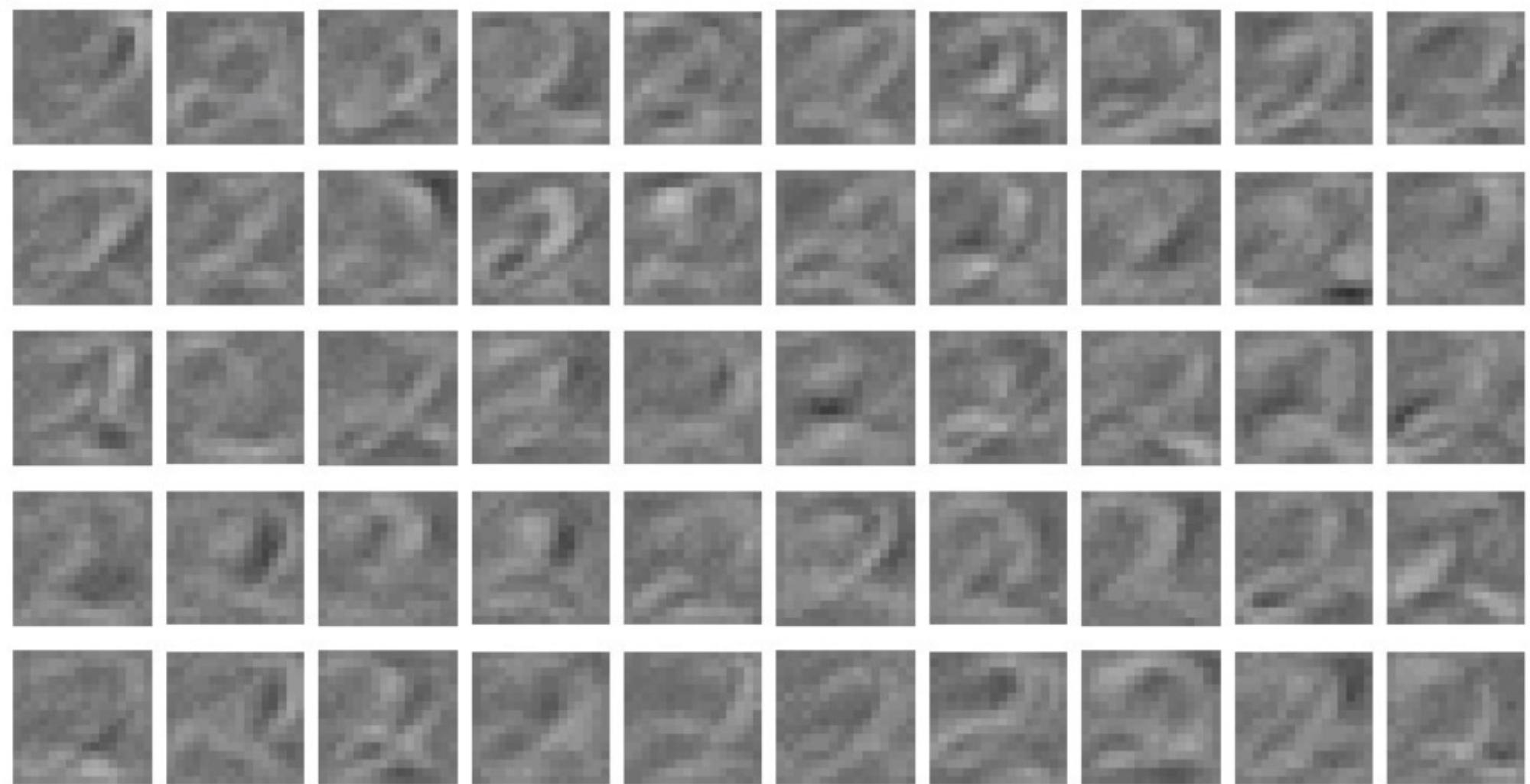
## The weights of the 50 feature detectors

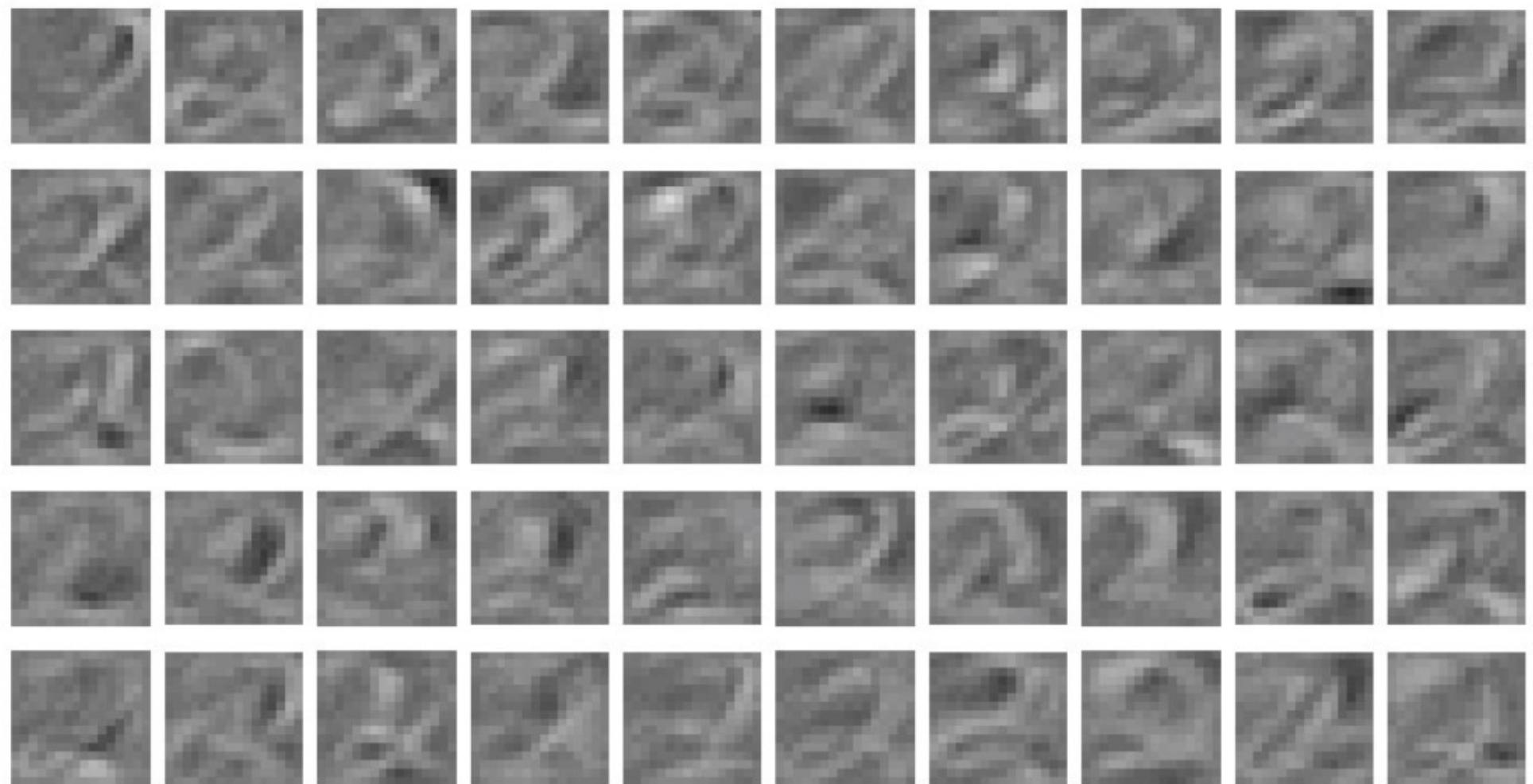


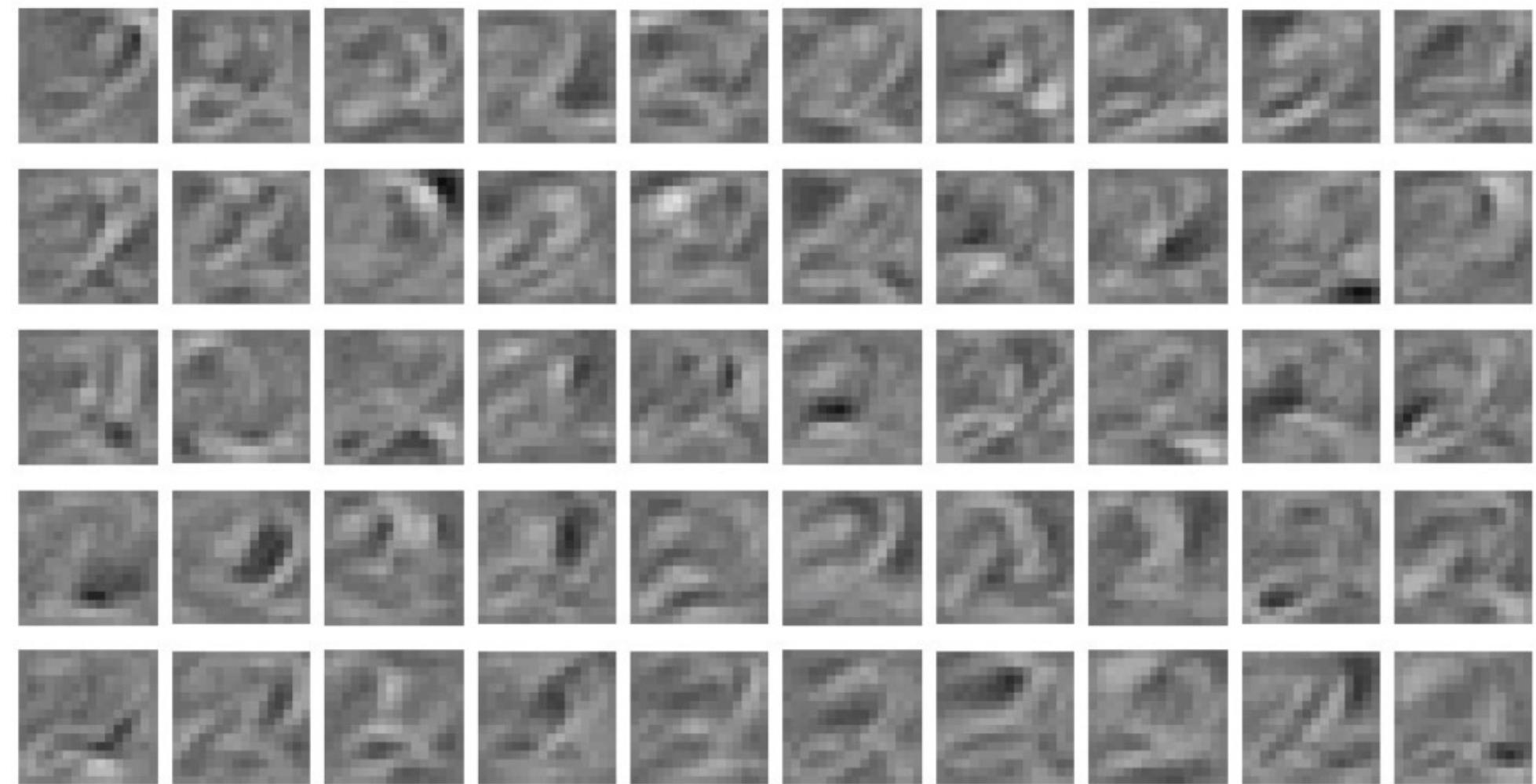
We start with small random weights

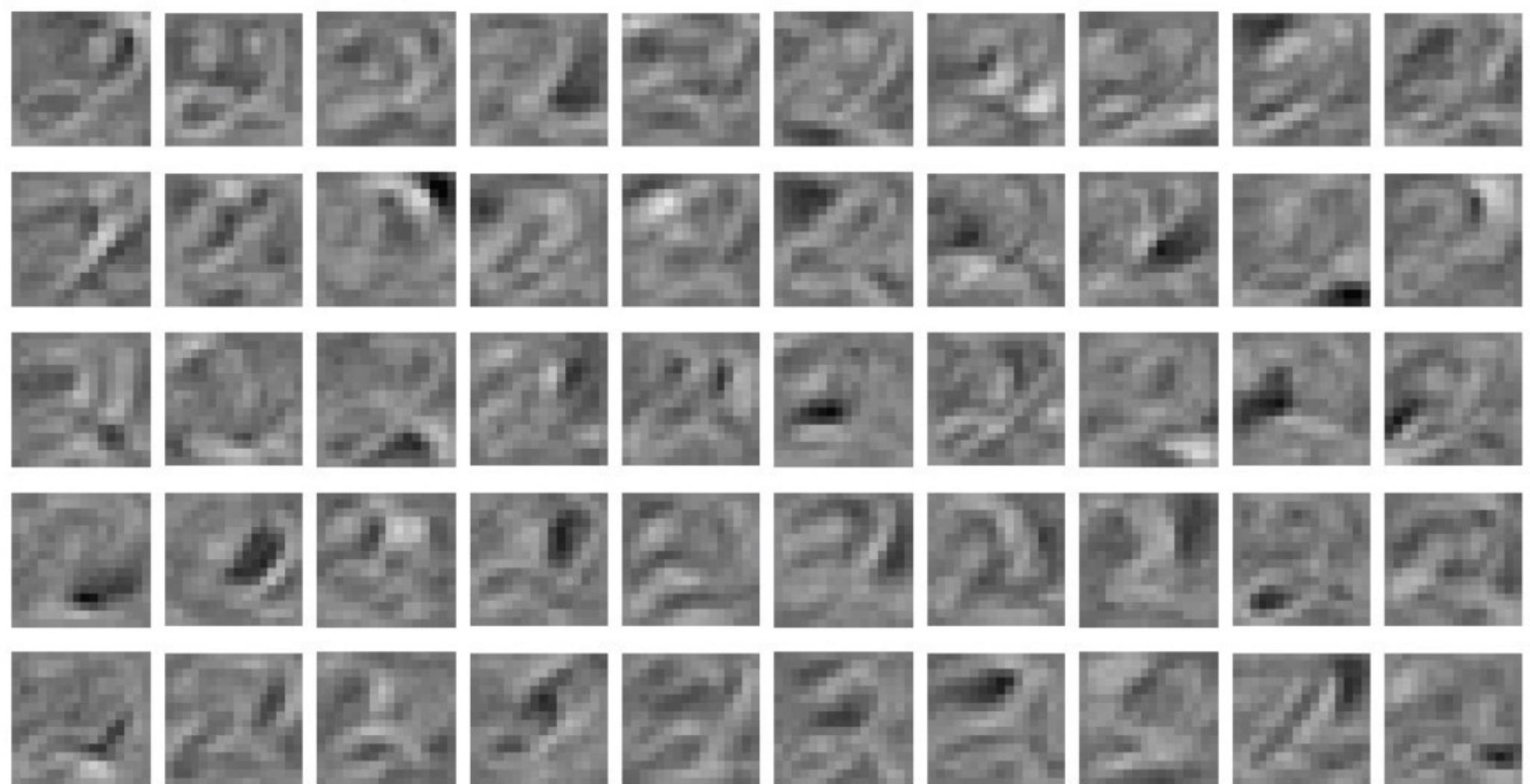


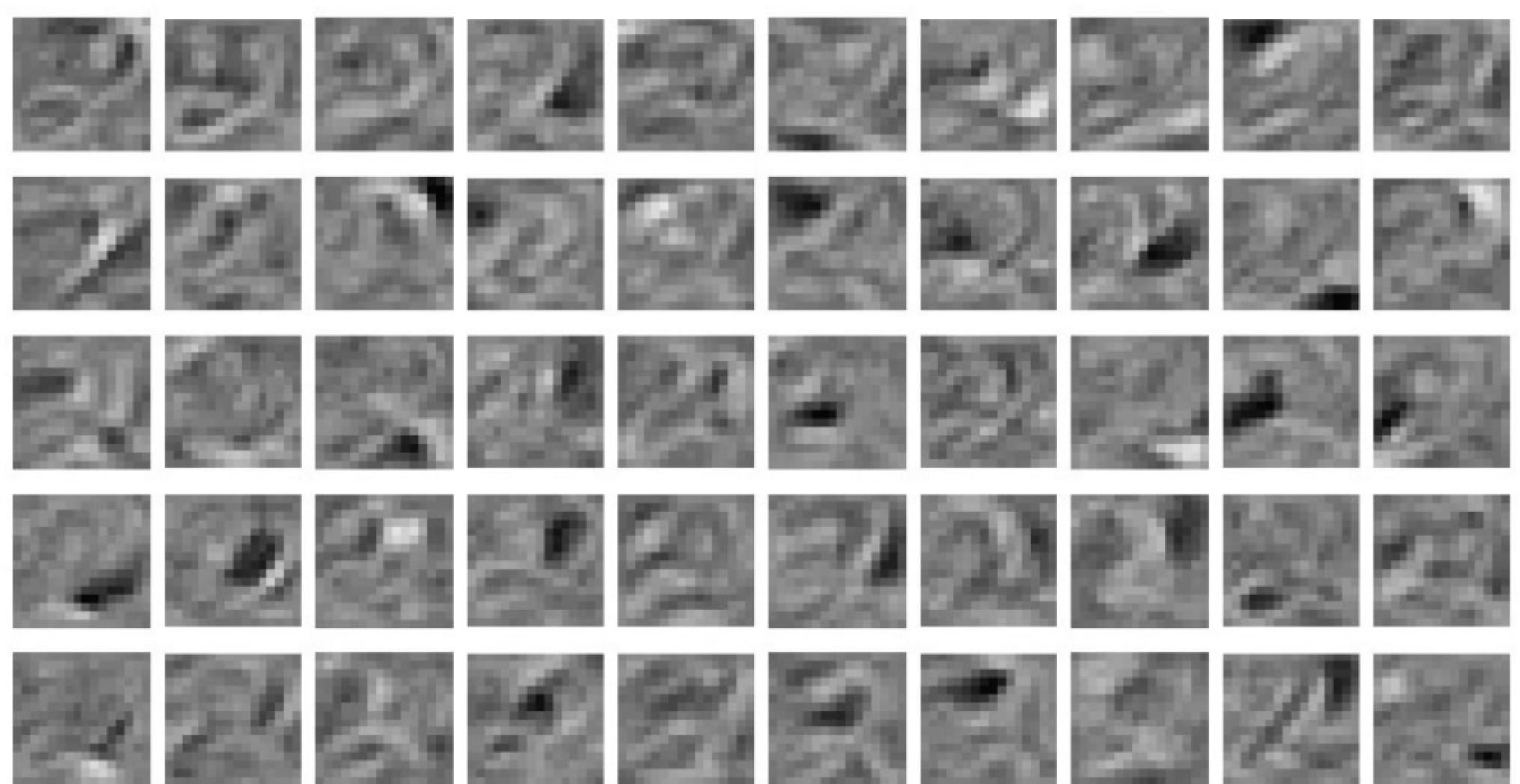




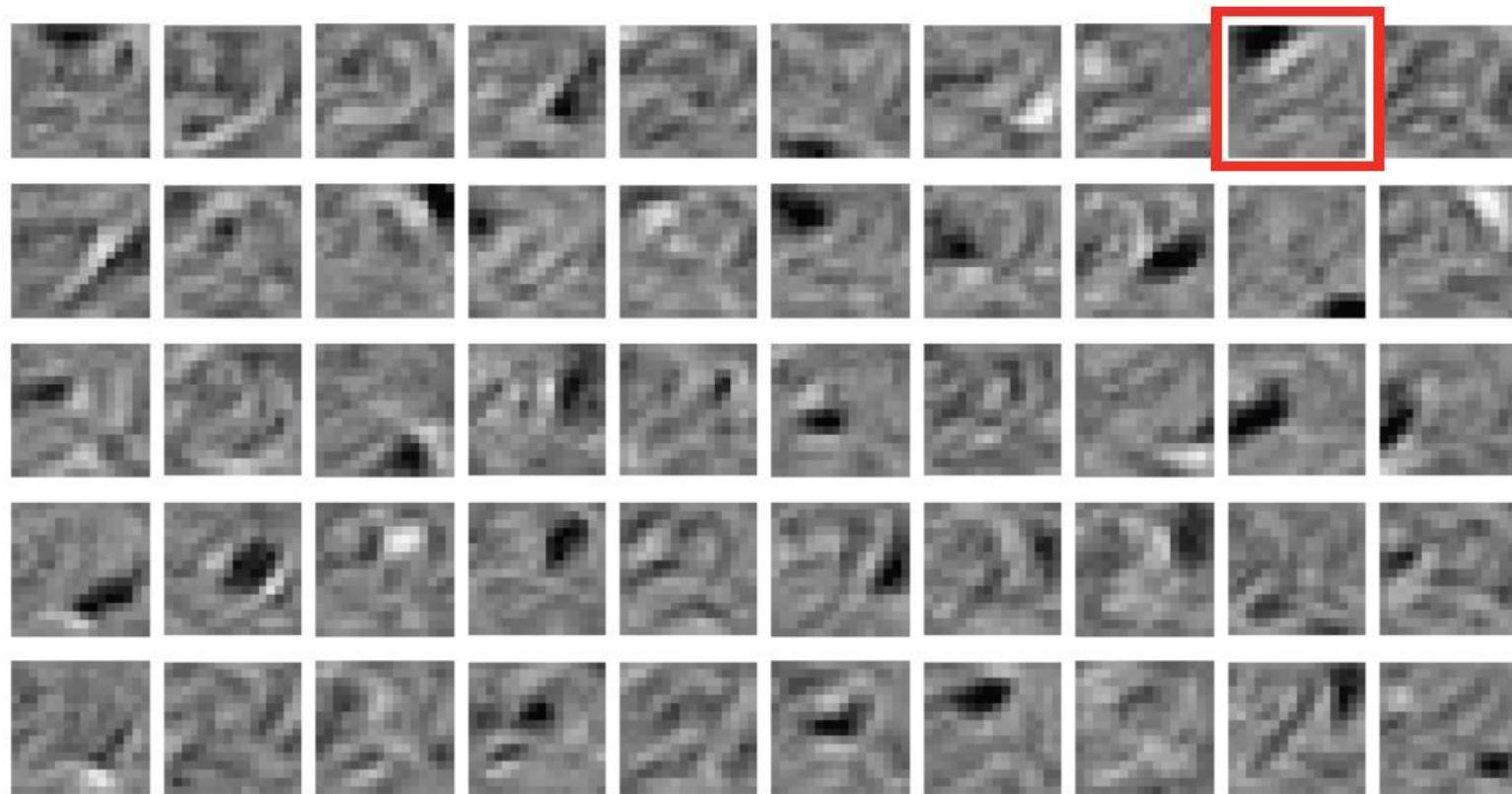




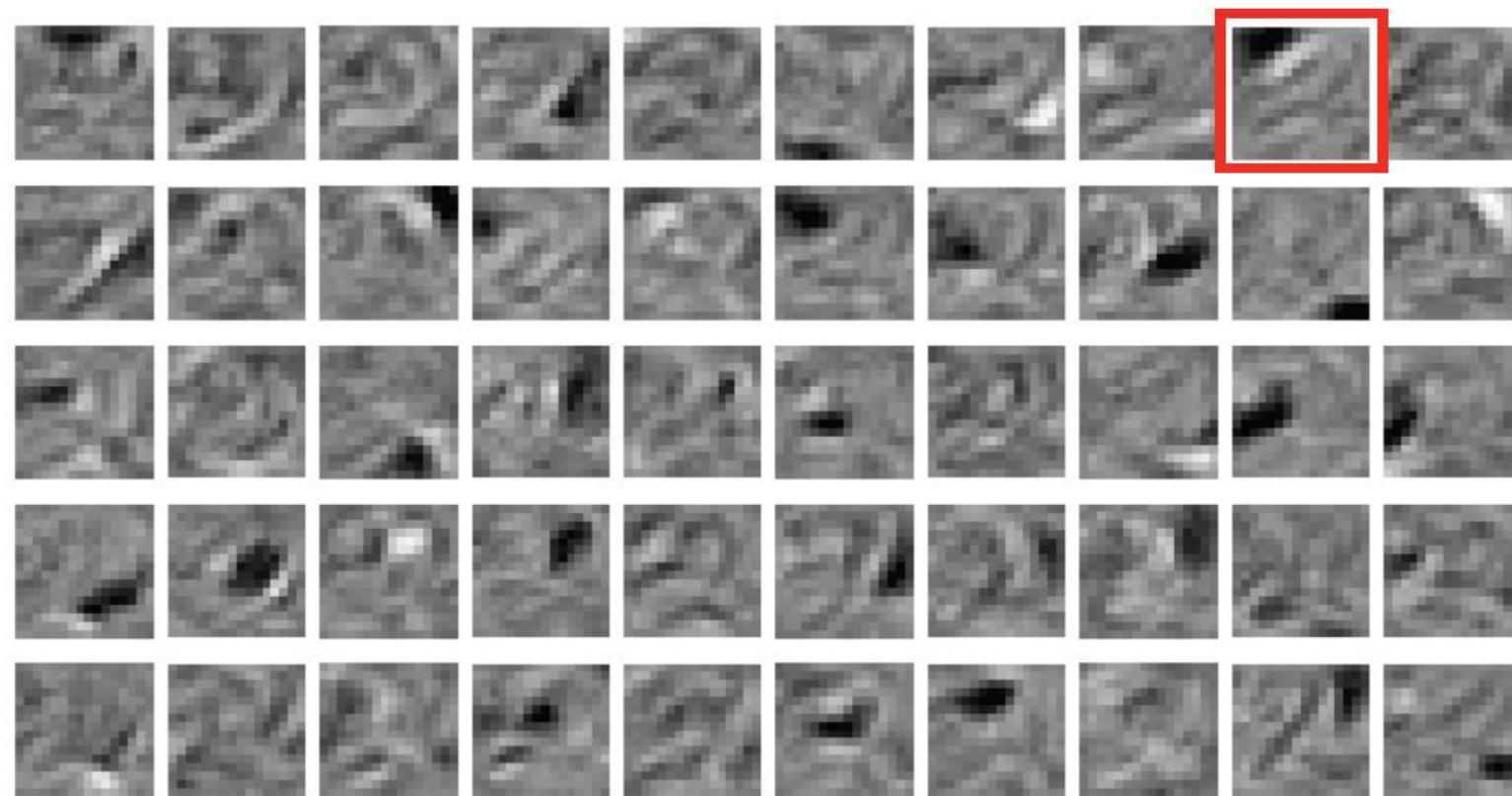




The final  $50 \times 256$  weights: Each neuron grabs a different feature



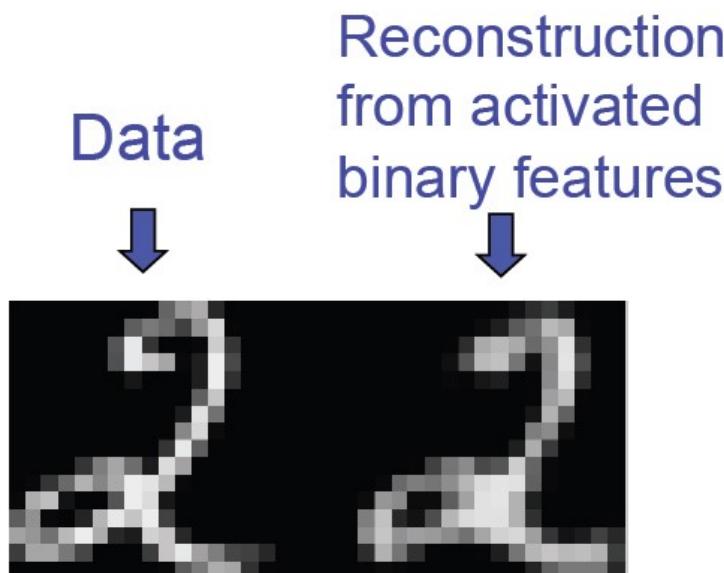
The final  $50 \times 256$  weights: Each neuron grabs a different feature



# USE OF RBM (After learning is complete)

- After having learnt the weights starting from a training set, RBM is ready to be used:
- **For generation** (generate a visual activation pattern from random hidden activation pattern)
- **For reconstruction** (start with a visible input-i.e.- visual activation pattern-and see what visual activation pattern the RBM reproduces after a few passages visual-hidden-visual.. Etc)

# How well can we reconstruct digit images from feature activations?



New test image from the digit class that the model was trained on

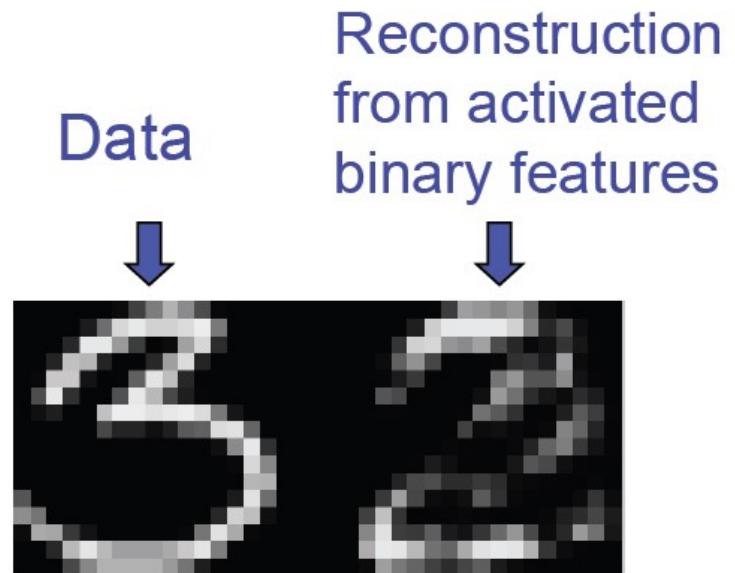
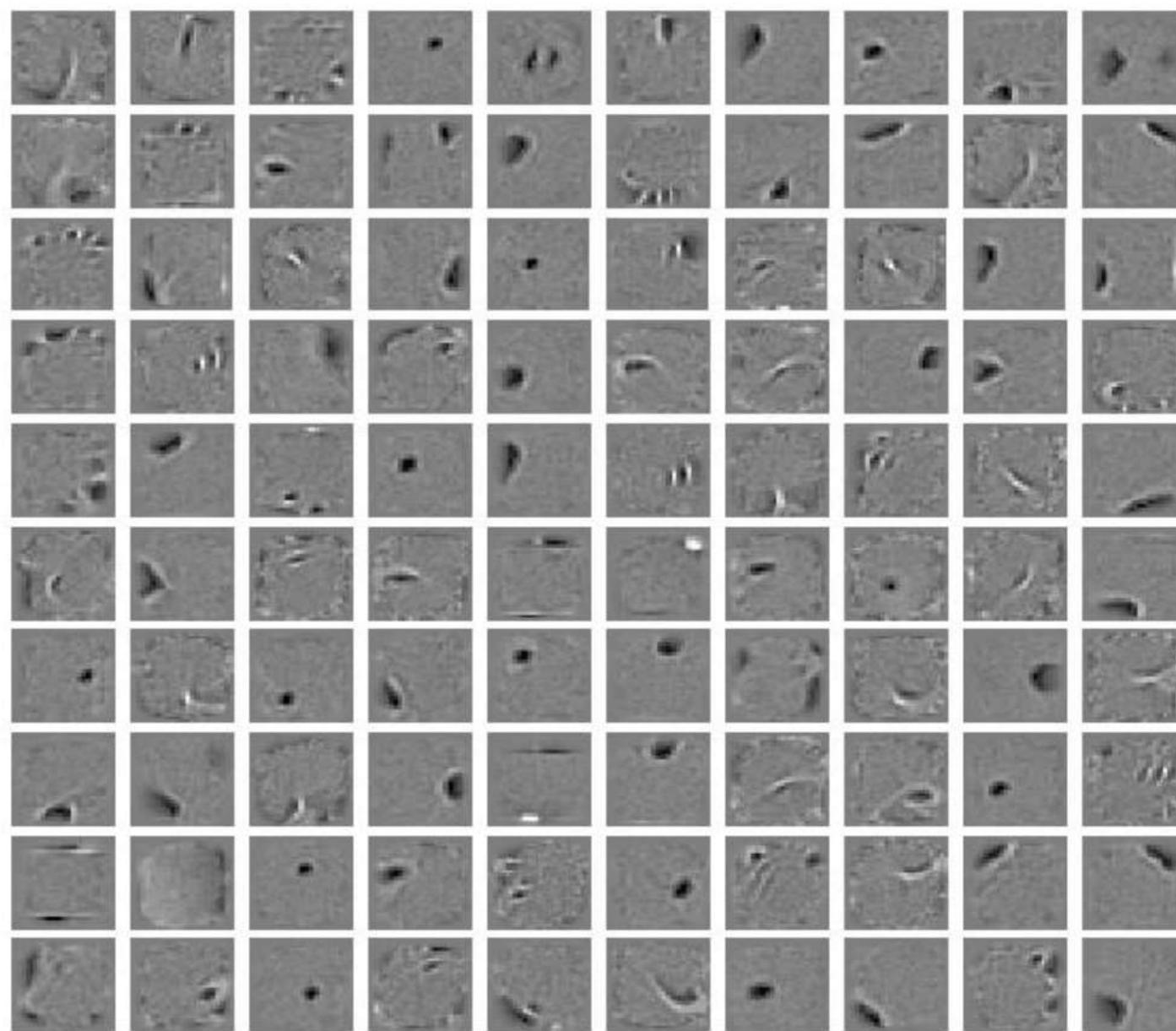
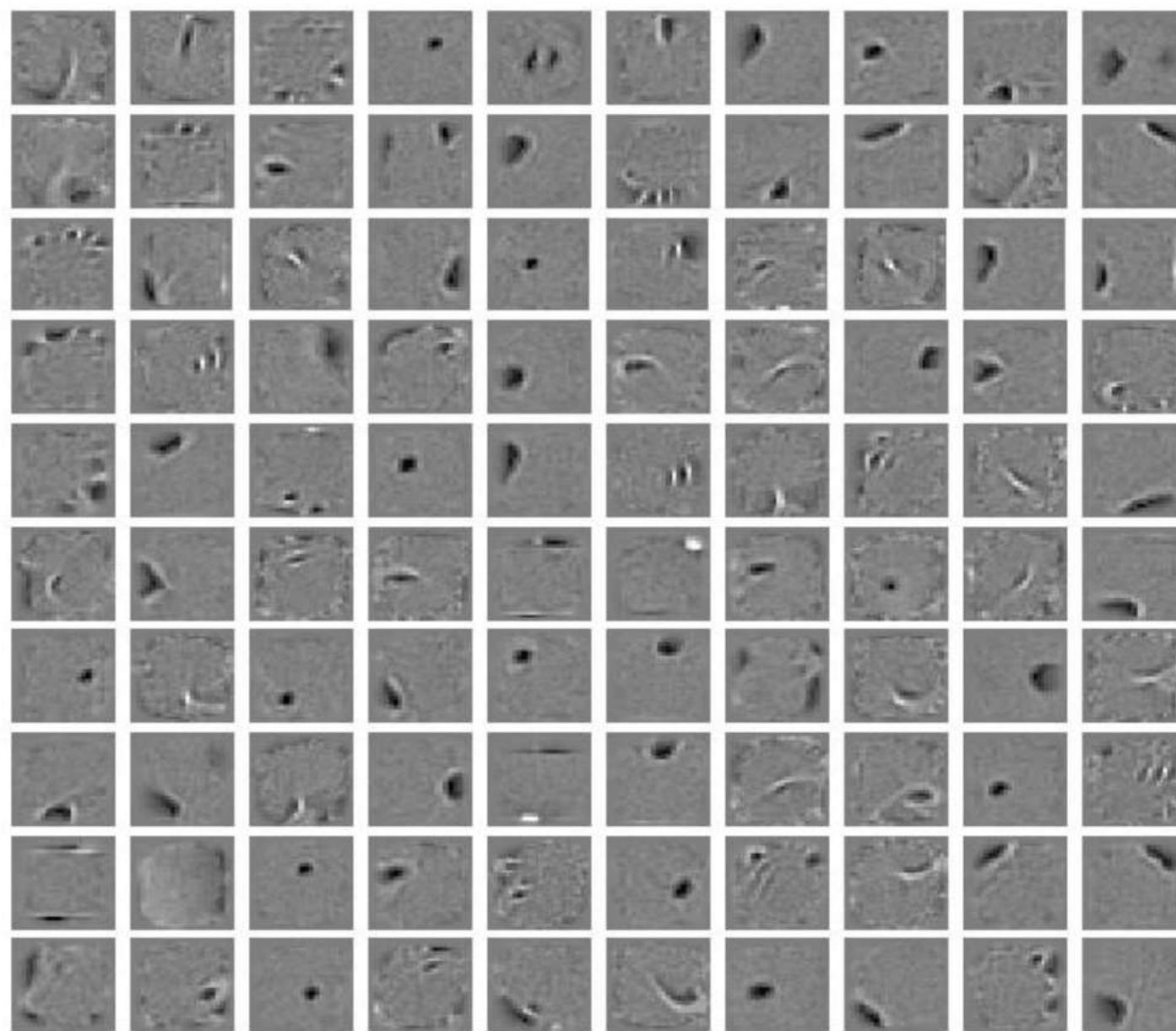


Image from an unfamiliar digit class  
The network tries to see every image as a 2.

this is actually really coooooool



Some features learned in the first hidden layer of a model of all 10 digit classes using 500 hidden units.



Some features learned in the first hidden layer of a model of all 10 digit classes using 500 hidden units.

# Further References:

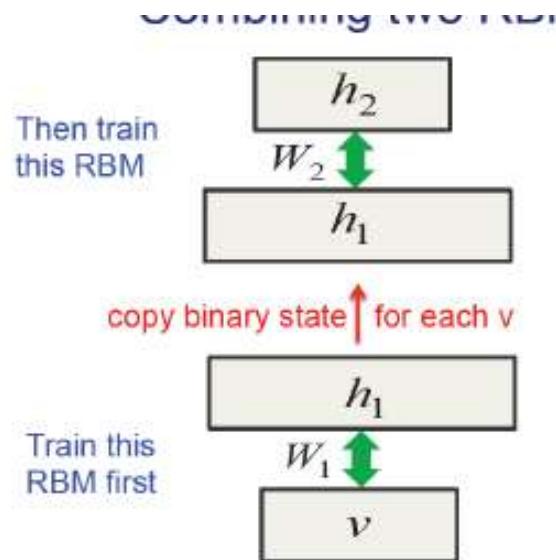
Hinton. A Practical Guide to Training Restricted Boltzmann Machines.

# DEEP RBM

- In the following slides there are some examples of Deep RBM (out of program for 2023/2024)

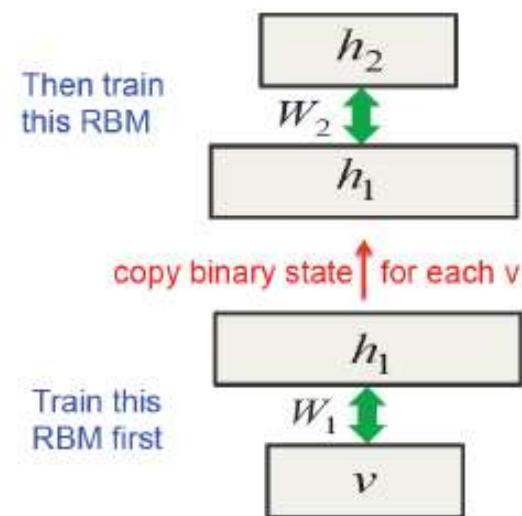
# Stacked RBMs

- First train a layer of features that receives input directly from the pixels.
- This layer discovers patterns of activities, regularities, in the layer below.
- Then treat the activations of the trained features as if they were pixels and learn features of features in a second hidden layer.
- This second hidden layer discovers patterns of activities in the layer below correlations between features discovered at the previous level.
- Each level models the correlation between activities in the level below
- Then do it again



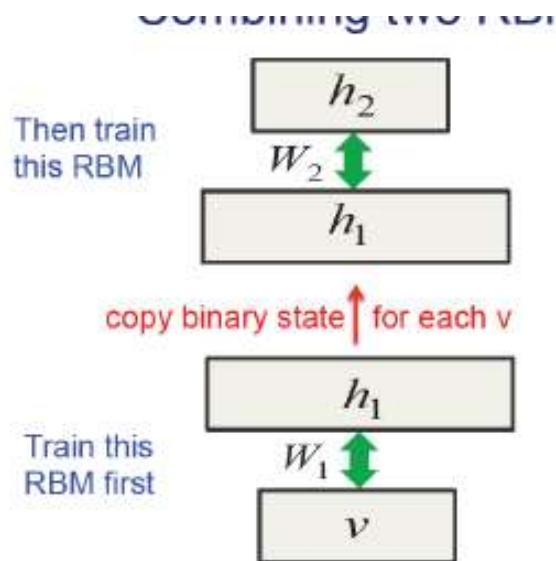
# Stacked RBMs

- It can be proven that each time we add a layer of features we improve the probability for generating the training data.

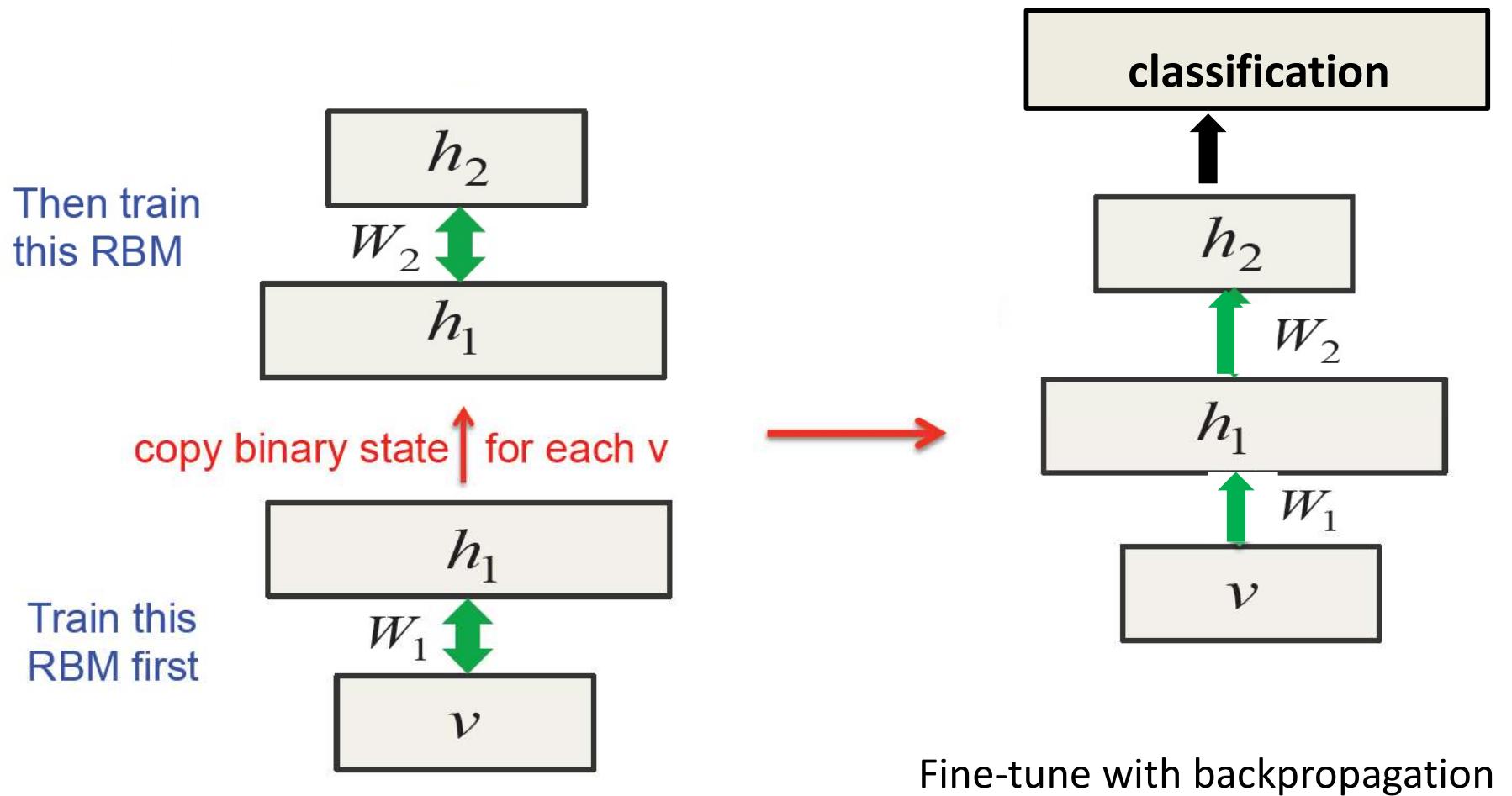


# Stacked RBMs

- It can be proven that each time we add a layer of features we improve the probability of generating the training data.
- Stacked Boltzmann machines can be **fine tuned** differently according to the desired applications.
  - If used for classification, by adding a classification layer → Better results than with random weights (similar to our lab!)
  - If used for generation, with deep belief nets
  - If used for finding few very descriptive features with autoencoders

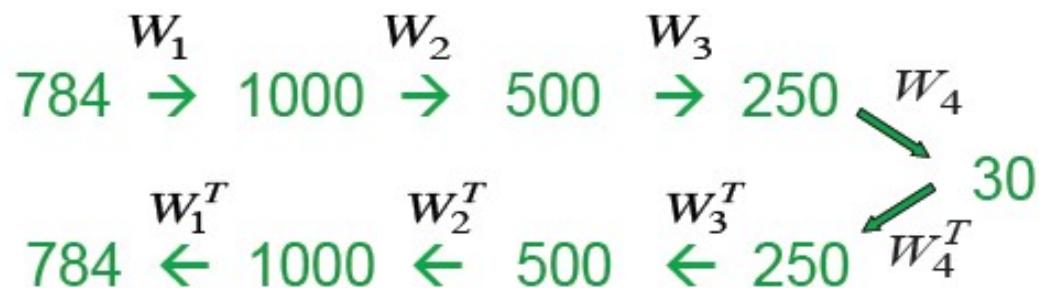


# Stacked RBMs for Discriminative Learning



# DEEP AUTOENCODERS

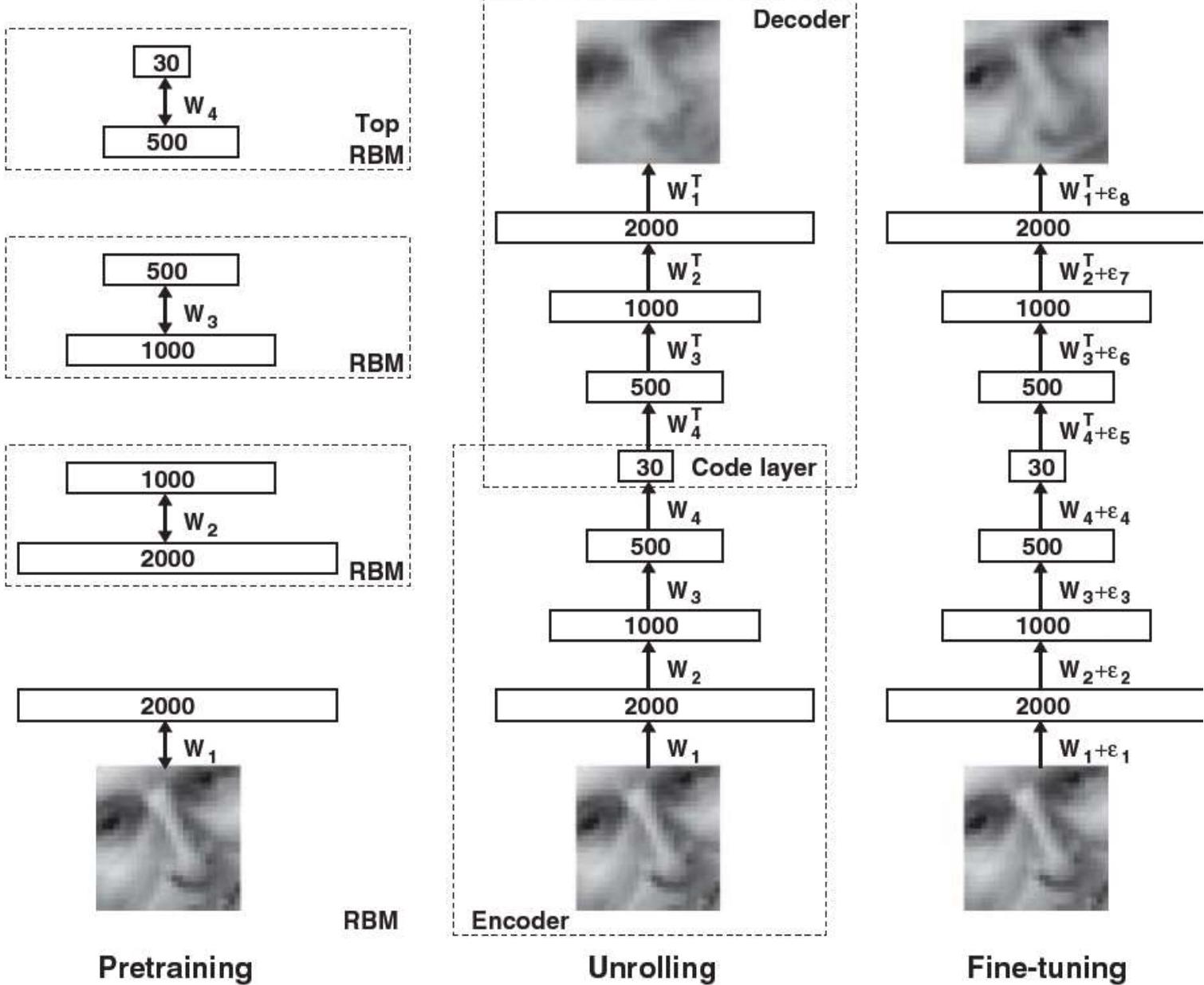
The first really successful deep autoencoders  
(Hinton & Salakhutdinov, Science, 2006)



We train a stack of 4 RBM's and then “unroll” them.  
Then we fine-tune with gentle backprop.

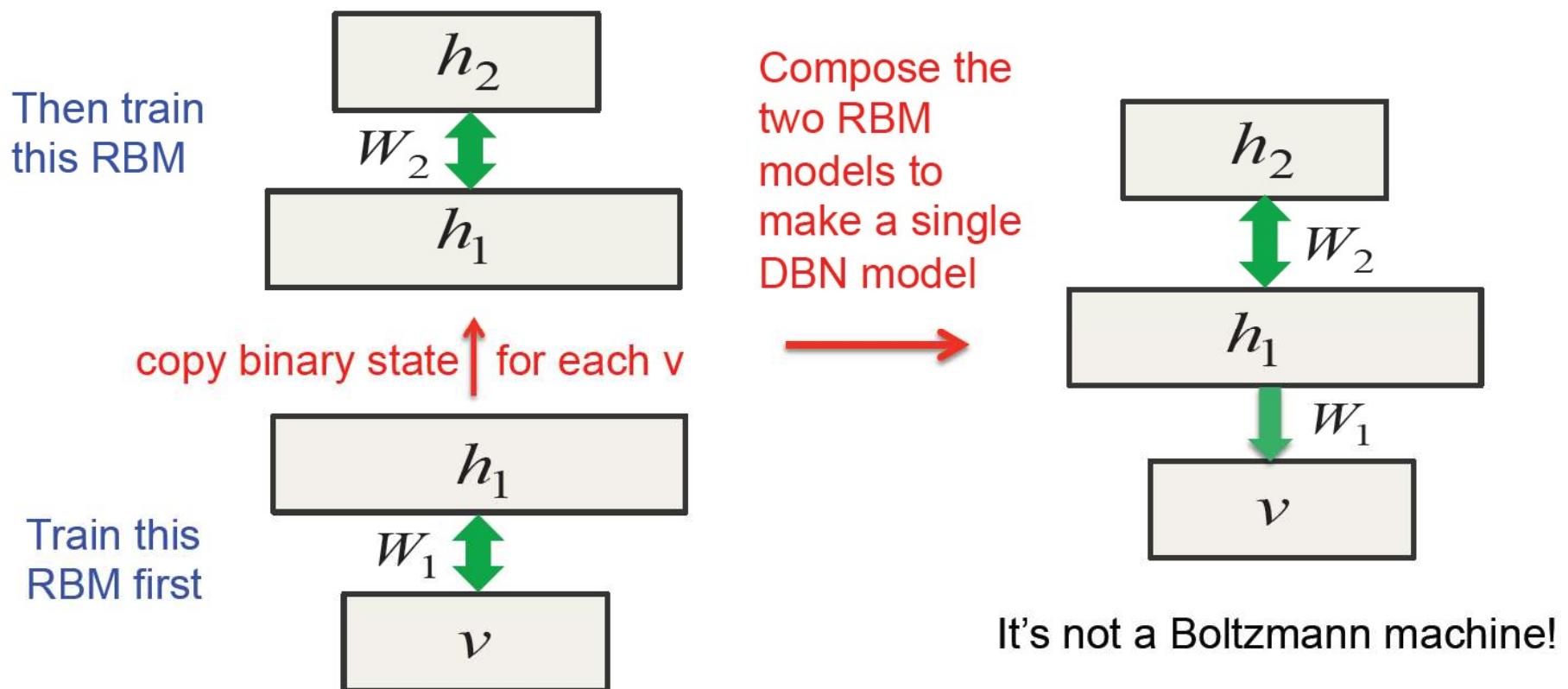
# DEEP AUTOENCODERS

•



# Stacked RBMs: Deep Belief Nets (DBN) for Generation

Combining two RBMs to make a DBN



# Stacked RBMs: Deep Belief Nets (DBN) for Generation

The generative model after learning 3 layers

To generate data:

1. Get an equilibrium sample from the top-level RBM |
2. Perform a top-down pass to get states for all the other layers.

The lower level bottom-up connections are **not** part of the generative model.

