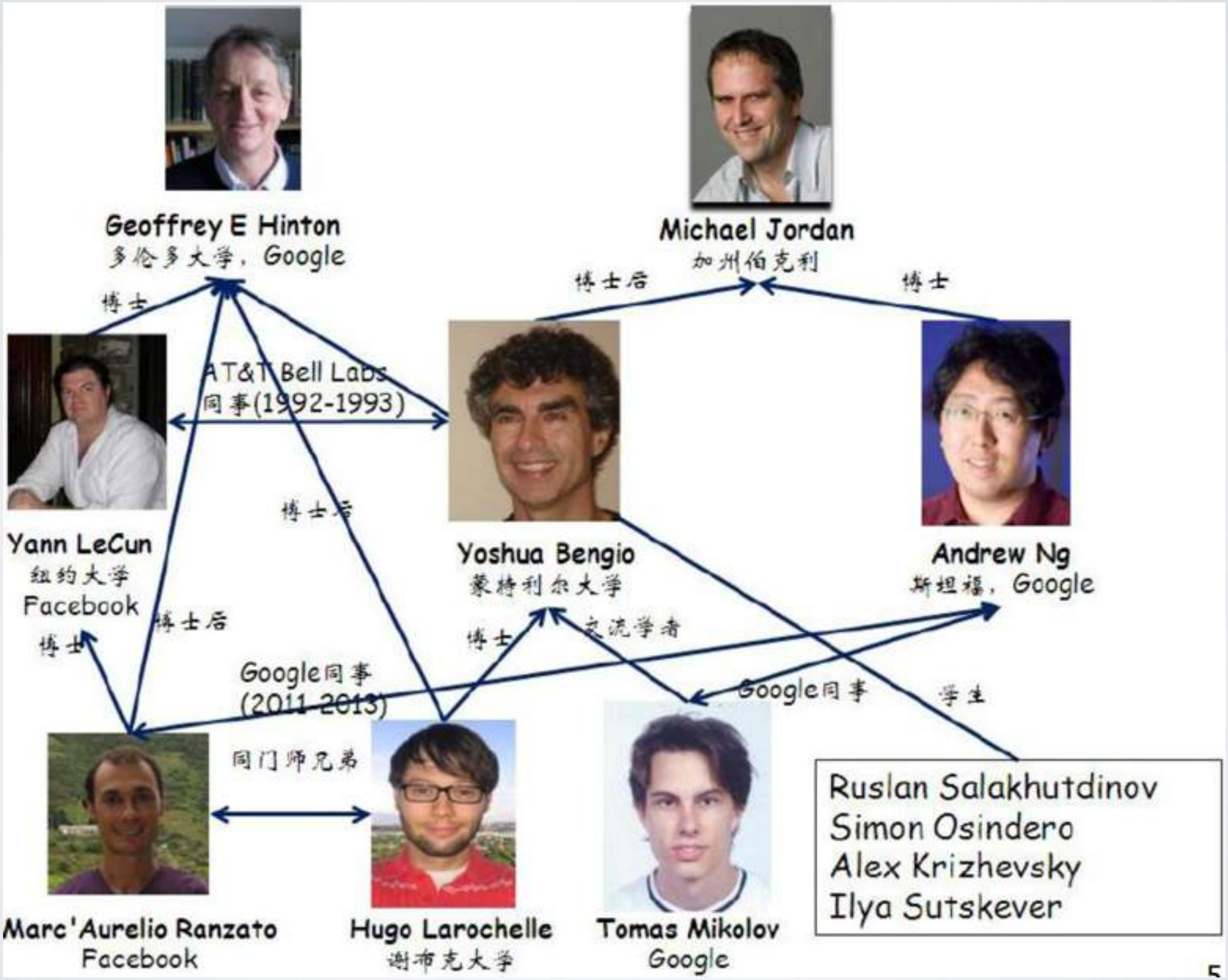
DEEP LEARNING

Junho Shin:
Juno @ AKA STUDY

Warning

Very biased on Geoffrey Hinton's lectures."

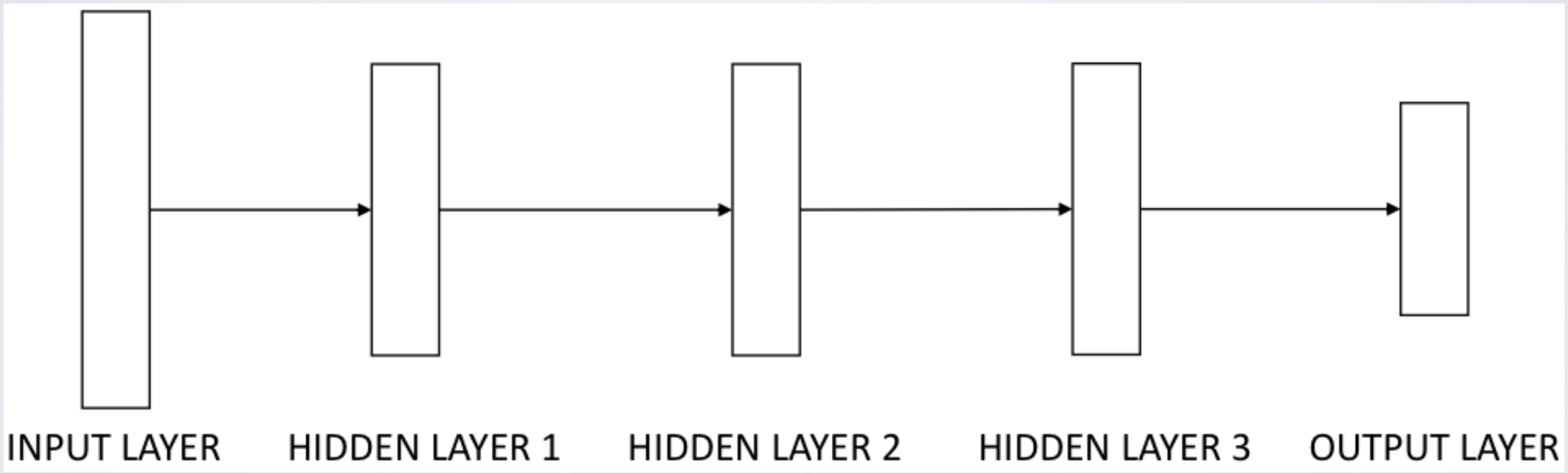


DEEP LEARNING

INTRODUCTION

WHAT IS "DEEP LEARNING"

Neural Network with deep architecture

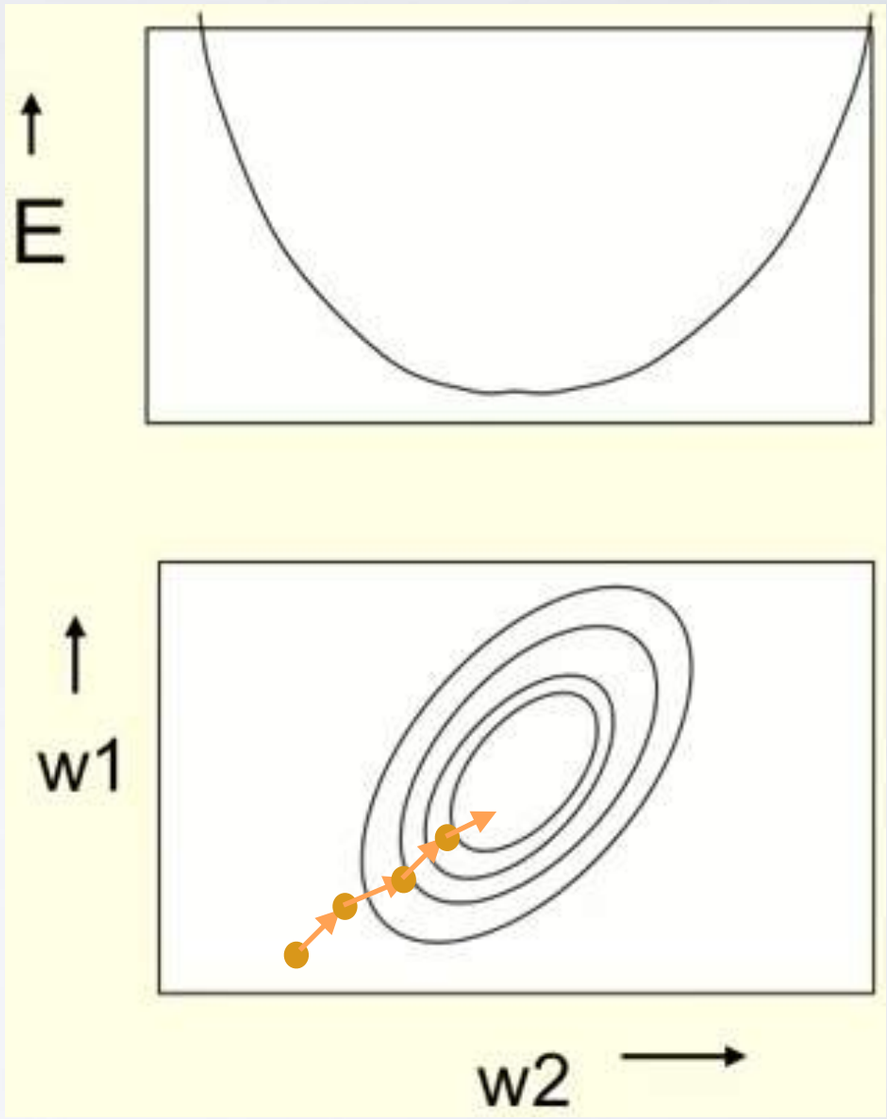


HISTORY

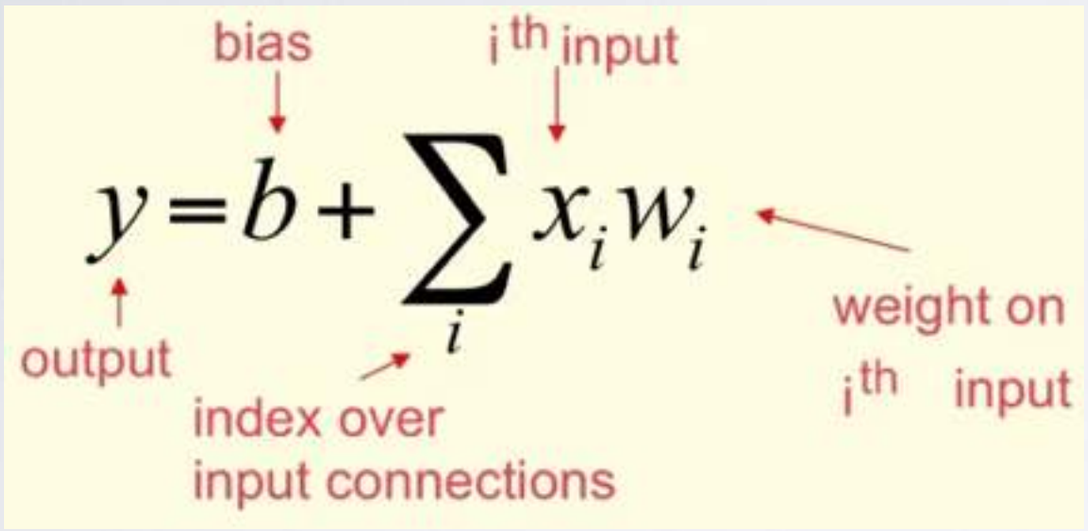
- Back-propagation algorithm: Learning multiple layers
  - Bryson & Ho (1969): Linear
  - Werbos (1974)
  - Rumelhart et al. (1981) ← Hinton
  - Parker (1985)
  - LeCun (1985)
  - Rumelhart et al. (1985)
- In late 1990s, most serious researchers had given up

NN BASIC: LEARNING

- **Goal**
  - To minimize the error summed over all training cases
- **Iterative approach**
  - Gradient descent



NN BASIC: LEARNING



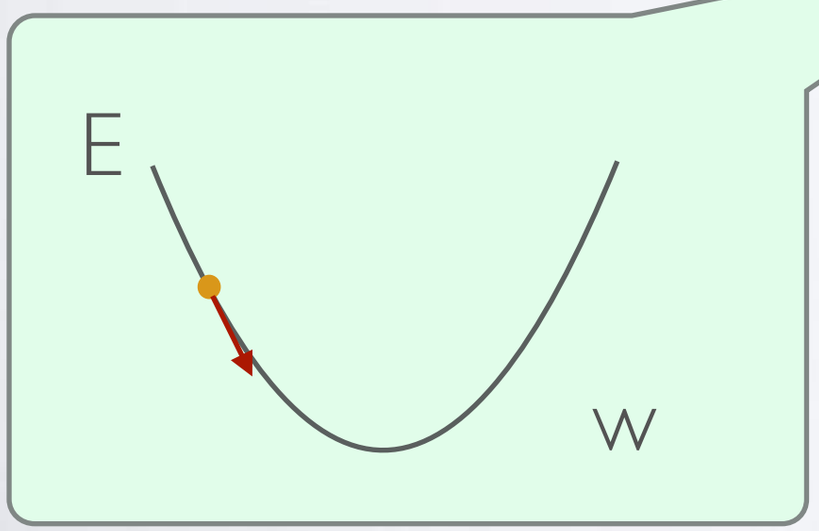
Cost function

$$E = \frac{1}{2} \sum_{n \in training} (t^n - y^n)^2$$

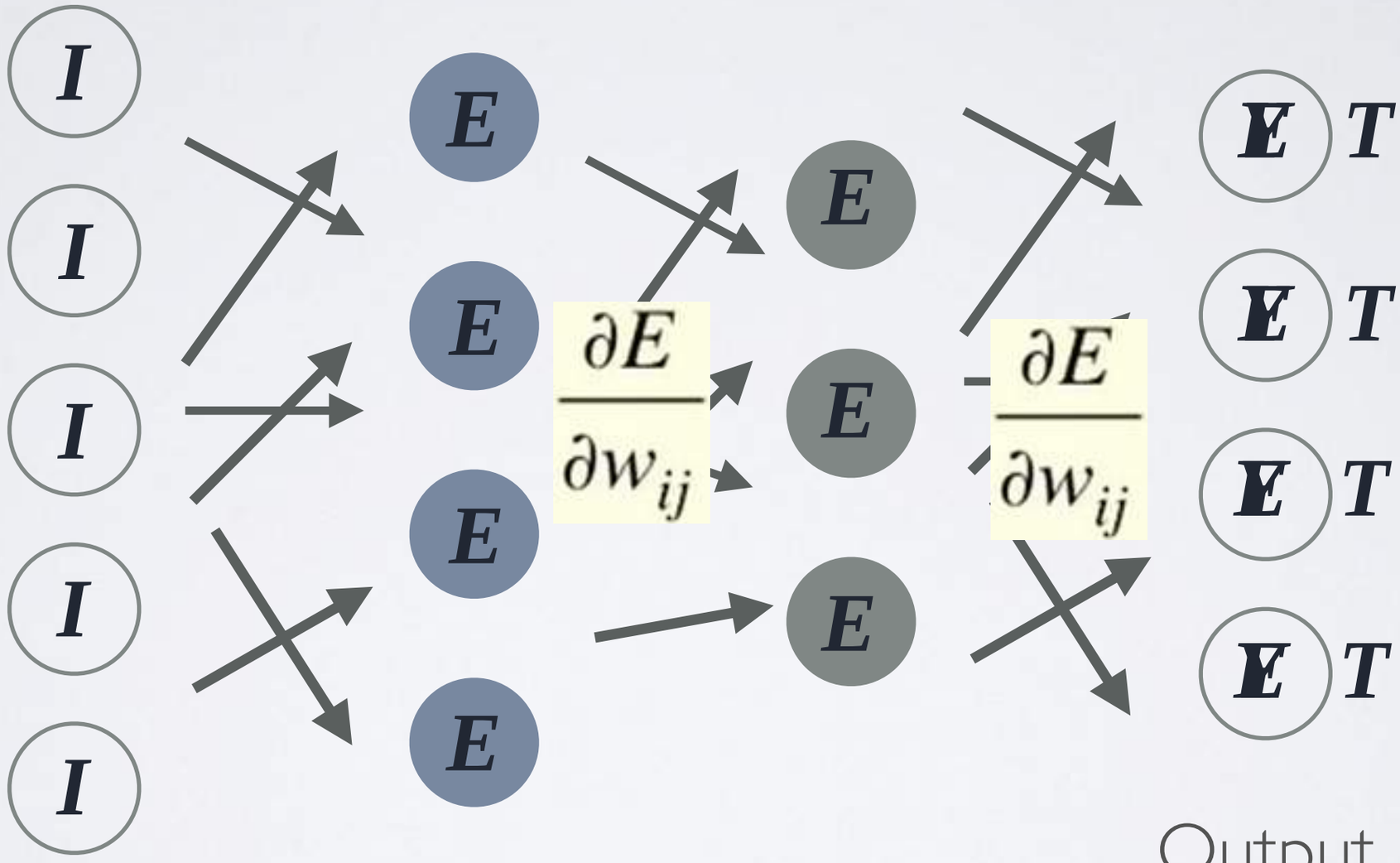
$$\frac{\partial E}{\partial w_i} = \frac{1}{2} \sum_n \frac{\partial y^n}{\partial w_i} \frac{dE^n}{dy^n}$$

$$= -\sum_n x_i^n (t^n - y^n)$$

$$\Delta w_i = -\varepsilon \frac{\partial E}{\partial w_i} = \sum_n \varepsilon \, x_i^n (t^n - y^n)$$



BACK-PROPAGATION



* **i, j** here denotes **i** *th*, **j** *th* layer

WHY NN WAS NOT PROMISING IN 90S

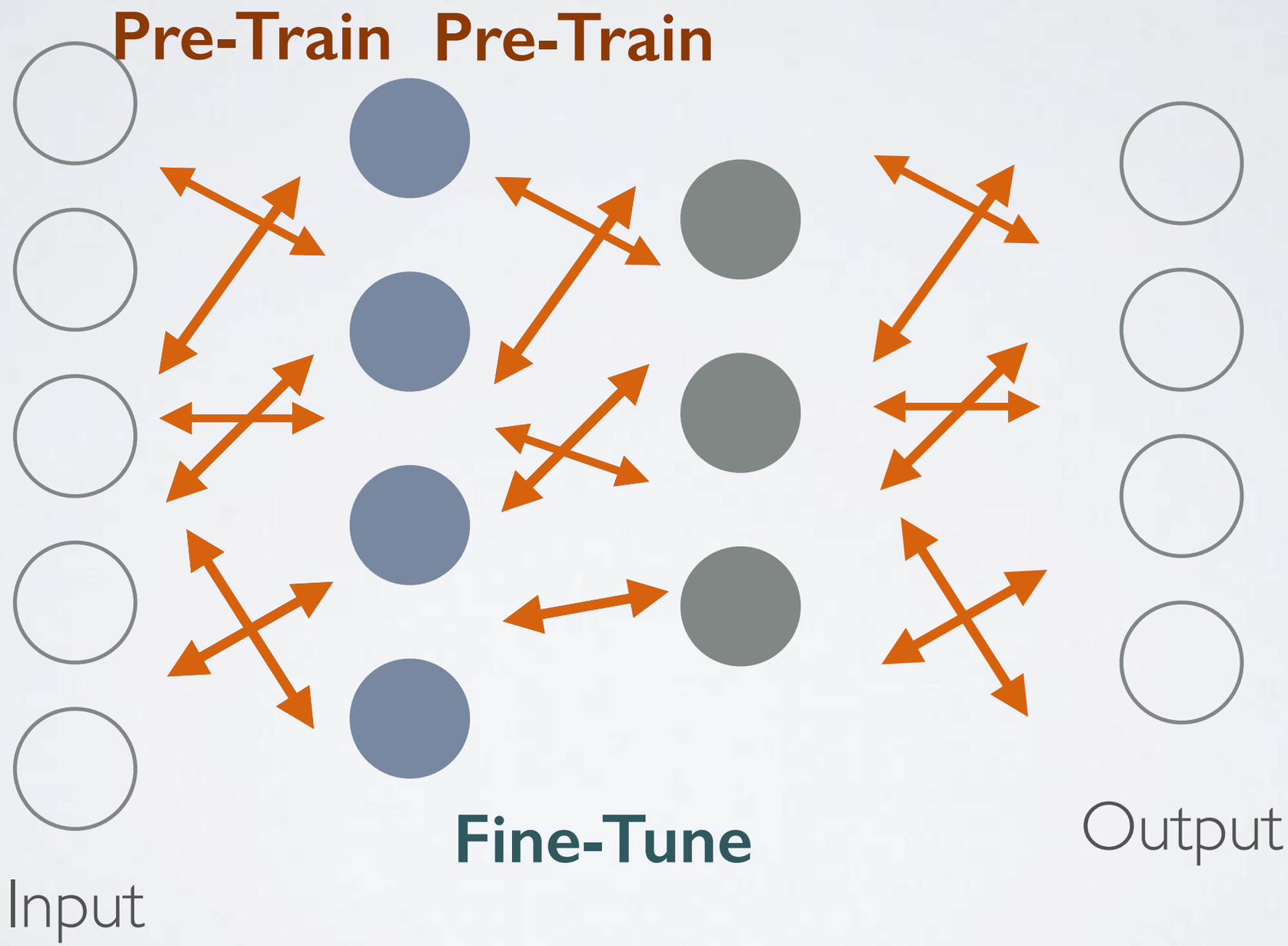
- **Popular explanation**
  - Could not make good use of multiple hidden layers (Recurrent networks, deep auto-encoders, etc.)
  - Support vector machine worked well
- **Real reasons**
  - Computers were very slow
  - Labeled datasets were too small
  - Deep networks were small and not initialized sensibly

SOLUTIONS

- **Techniques for deep network** (Especially for Recurrent NN)
  - Careful initialization of weights
  - Long short term memory
  - Hessian free optimization: Deal with vanishing gradient
  - Echo state networks
- **Deep architecture with energy-based model**

NEW ARCHITECTURE

* Pre-train: unsupervised learning

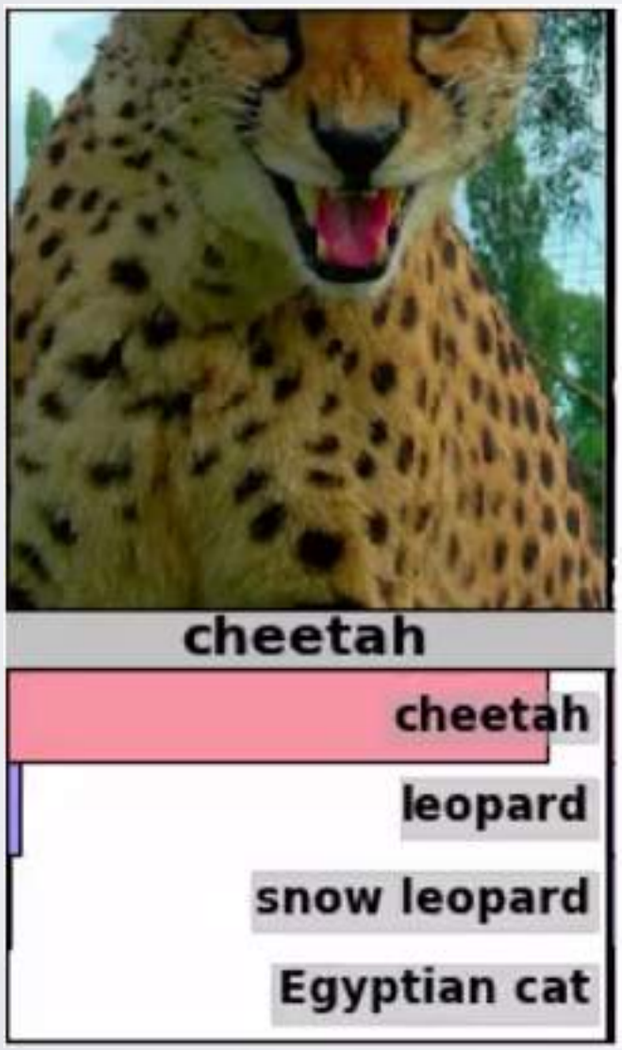


WHY NEW ARCHITECTURE WORKS

- Pre-train
  - Learn features (good representations) from inputs
- Fine-tune with back propagation
  - Start from pre-trained weights
  - Optimize network for discrimination

SUCCESS STORIES

- Competition on ImageNet



SUCCESS STORIES

- Competition on ImageNet

Error rates on the ILSVRC-2012 competition

	classification	classification &localization
• University of Tokyo	• 26.1%	53.6%
• Oxford University Computer Vision Group	• 26.9%	50.0%
• INRIA (French national research institute in CS) + XRCE (Xerox Research Center Europe)	• 27.0%	
• University of Amsterdam	• 29.5%	
University of Toronto (Alex Krizhevsky)	• 16.4%	34.1%

SUCCESS EXAMPLES

- Speech recognition test

Word error rates from MSR, IBM, & Google
(Hinton et. al. IEEE Signal Processing Magazine, Nov 2012)

The task	Hours of training data	Deep neural network	Gaussian Mixture Model	GMM with more data
Switchboard (Microsoft Research)	309	18.5%	27.4%	18.6% (2000 hrs)
English broadcast news (IBM)	50	17.5%	18.8%	
Google voice search (android 4.1)	5,870	12.3% (and falling)		16.0% (>>5,870 hrs)

DEEP LEARNING THEORY

Warning

You don’t really need to understand the next part.

BACKGROUND KNOWLEDGE FOR DNN

Types of neurons:
Linear, Binary threshold, Rectified linear, Sigmoid, Stochastic binary

Gradient descent

Weight initialization

Regularization

Back-propagation

Probabilistic Graphical Models:
Conditional probabilities
Markov Chain Monte Carlo, Gibbs Sampling

Markov models

Types of neural networks:
Feed-forward, Recurrent, Symmetrically-connected

Fast convergence methods

Types of learning:
Supervised, Unsupervised

Energy network models

Mini-batch & online gradient descent

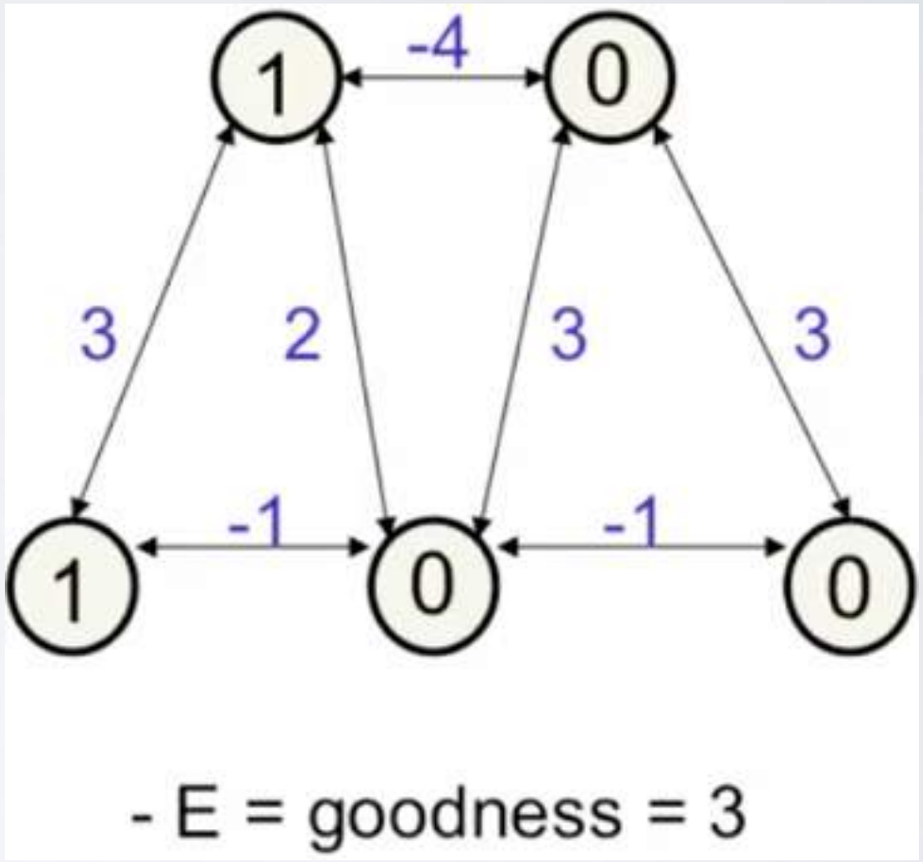
Softmax function

Types of cost functions:
Logistic (Sigmoid), Cross entropy, …

HOPFIELD NET

- Energy Model
  - Symmetric
  - Binary threshold unit

$$E = -\sum_i s_i b_i - \sum_{i<j} s_i s_j w_{ij}$$



- Goal: Memorize given inputs (Binary configuration)

LEARNING

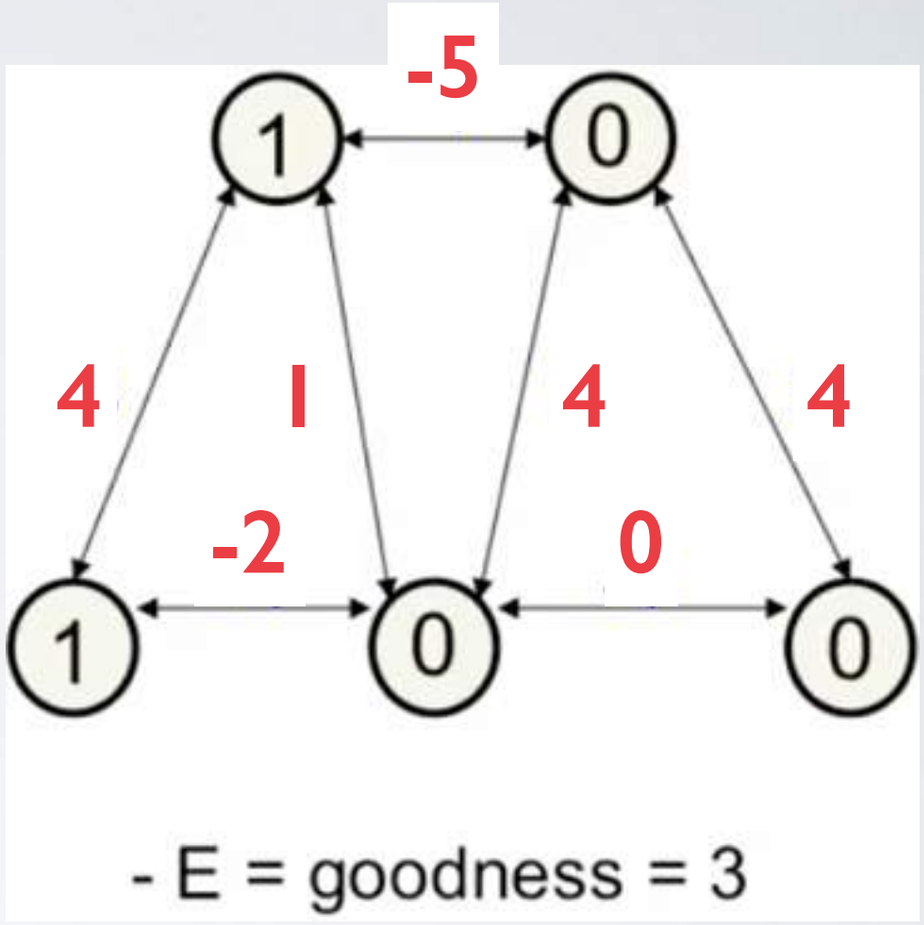
Example: Memorize (1, 0, 1, 0, 0)

$$\Delta w_{ij} = s_i s_j$$

With states of -1, 1

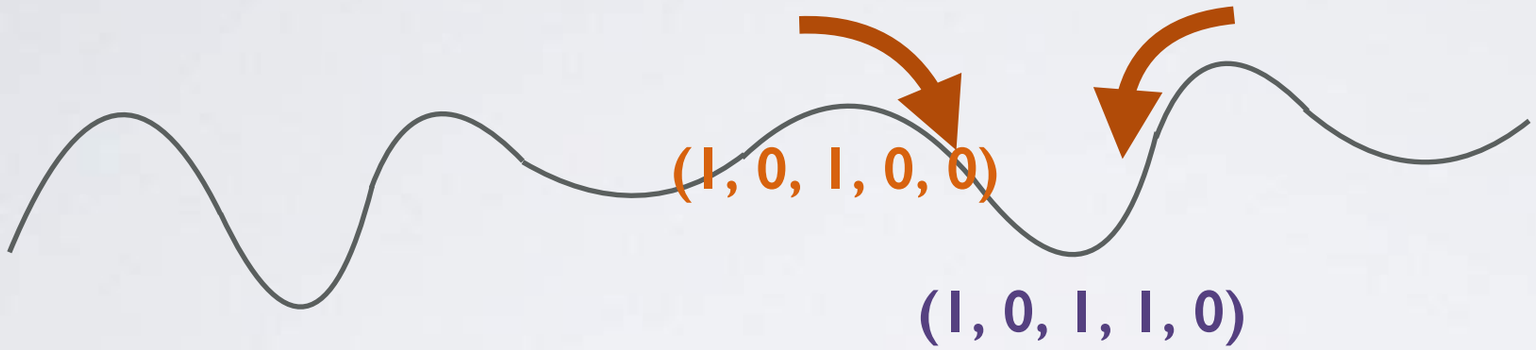
$$\Delta w_{ij} = 4\,(s_i - \tfrac{1}{2})(s_j - \tfrac{1}{2})$$

With states of 0, 1



GETTING A SAMPLE

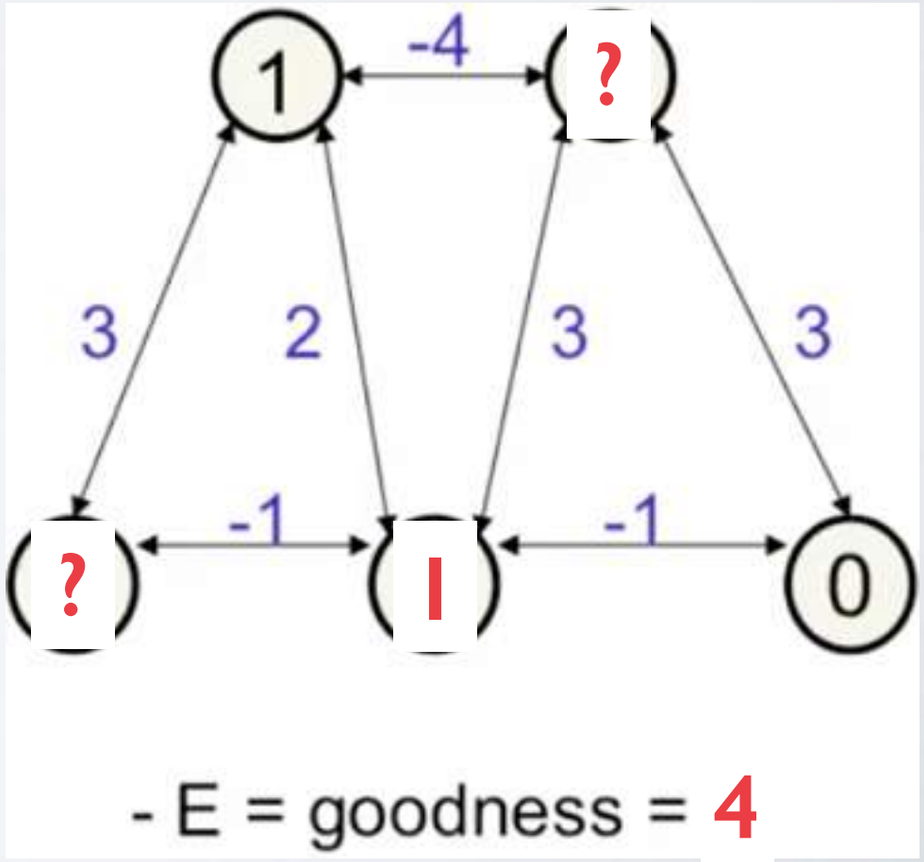
- Get a very probable configuration (by checking energy)



- Binary decision rule
  - To settle to equilibrium (energy min)

$$\Delta E_i = E(s_i = 0) - E(s_i = 1) = b_i + \sum_j s_j w_{ij}$$

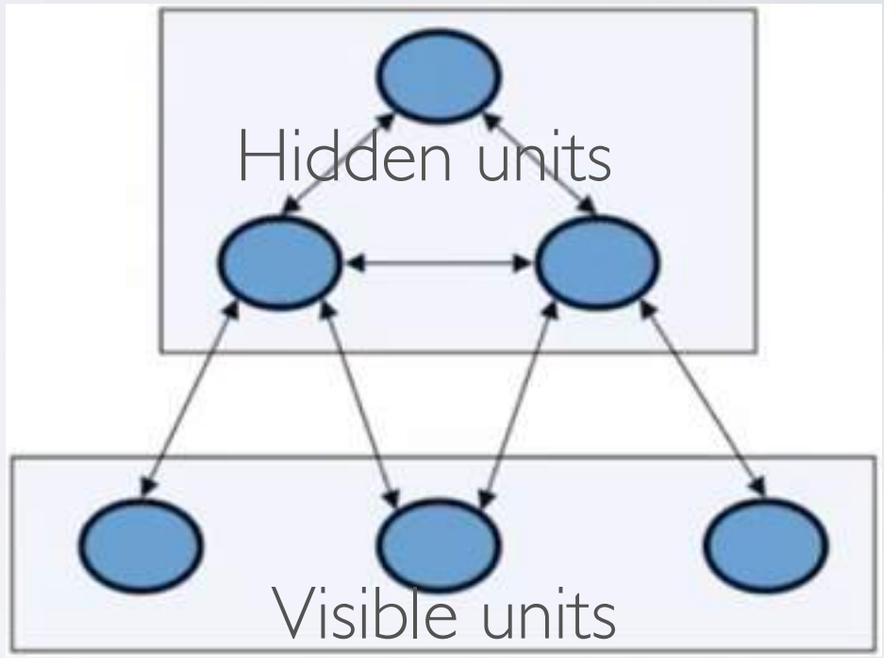
Energy gap



BOLTZMANN MACHINE

- **Hidden units** are internal representation of **visible units**

- Goal : Memorize visible configurations with the most probable hidden configurations



$$-E(\mathbf{v},\mathbf{h}) = \sum_{i \in vis} v_i b_i + \sum_{k \in hid} h_k b_k + \sum_{i<j} v_i v_j w_{ij} + \sum_{i,k} v_i h_k w_{ik} + \sum_{k<l} h_k h_l w_{kl}$$

binary state of unit i in **v**

bias of unit k

Energy with configuration **v** on the visible units and **h** on the hidden units

indexes every non-identical pair of i and j once

weight between visible unit i and hidden unit k

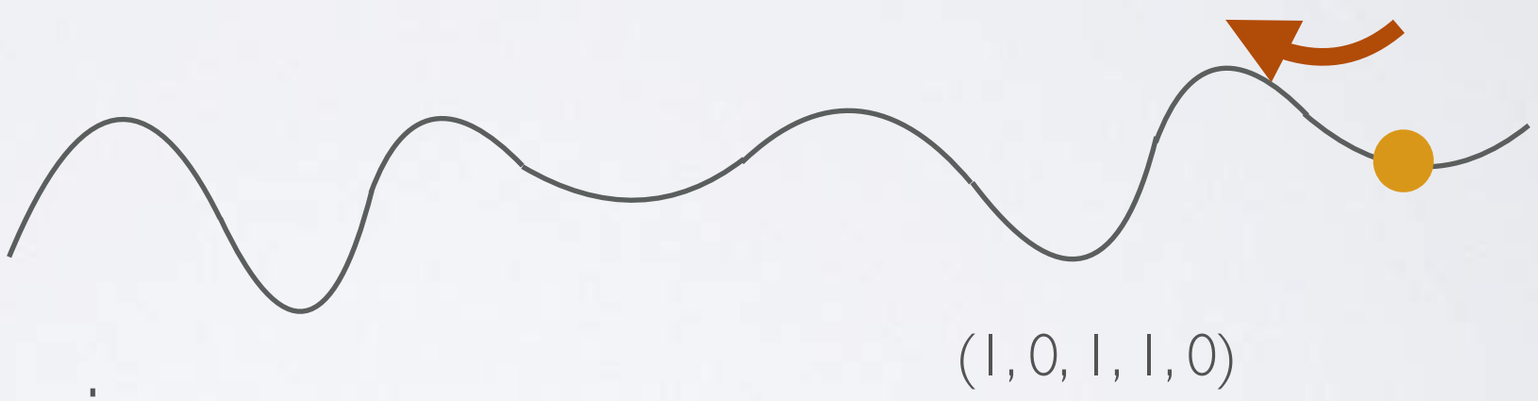
GETTING A SAMPLE

- Markov chain monte carlo
  - Start from random global configurations **(v, h)**
  - Keep picking units at random to **stochastically** update their states based on their energy gaps
  - Get a configuration **(v, h)** when the network reaches thermal equilibrium

E gap > 0 : 1?
E gap < 0 : 0?

HOW TO AVOID LOCAL MINIMUM

- The net always makes decisions that reduce energy
- Solutions
  - Use random noise
  - Use stochastic units



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

LEARNING

- Memorize visible configurations with the most probable hidden configurations

$$\frac{\partial \log p(\mathbf{v})}{\partial w_{ij}} = \langle s_i s_j \rangle_{\mathbf{v}} - \langle s_i s_j \rangle_{model}$$

Derivative of log probability of one training vector, v under the model.

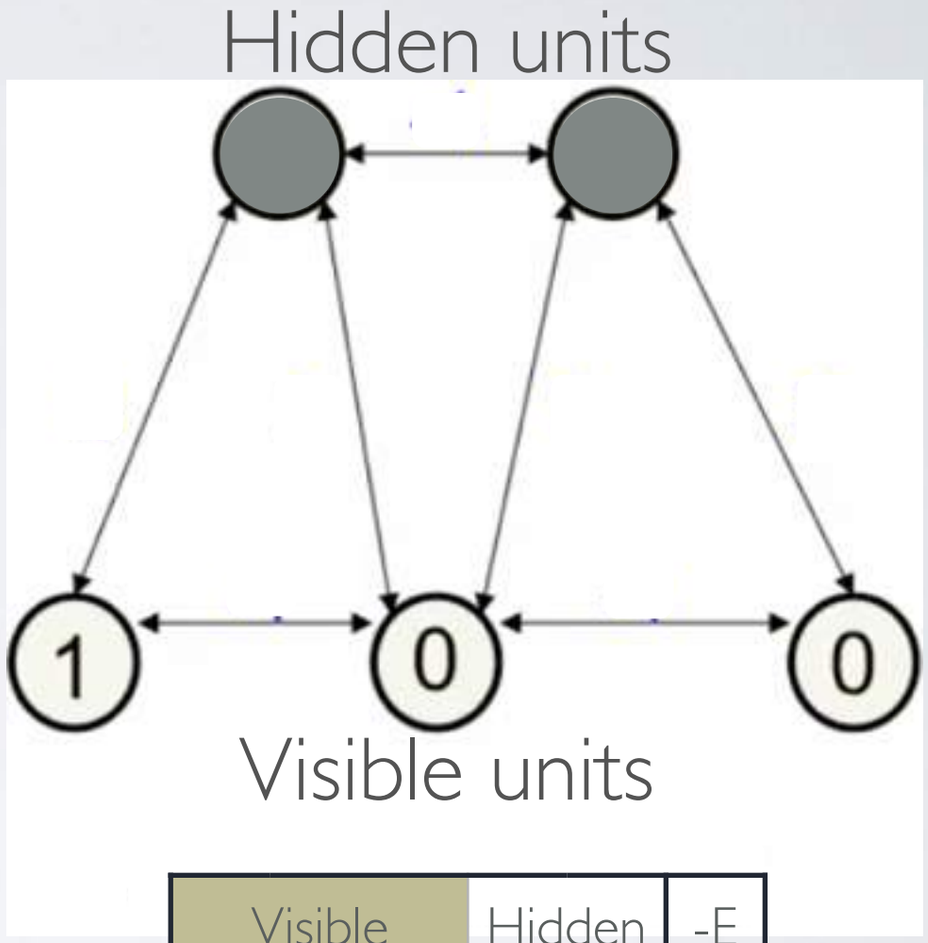
Expected value of product of states at thermal equilibrium when v is clamped on the visible units

Expected value of product of states at thermal equilibrium with no clamping

$$\Delta w_{ij} \propto \langle s_i s_j \rangle_{data} - \langle s_i s_j \rangle_{model}$$

LEARNING

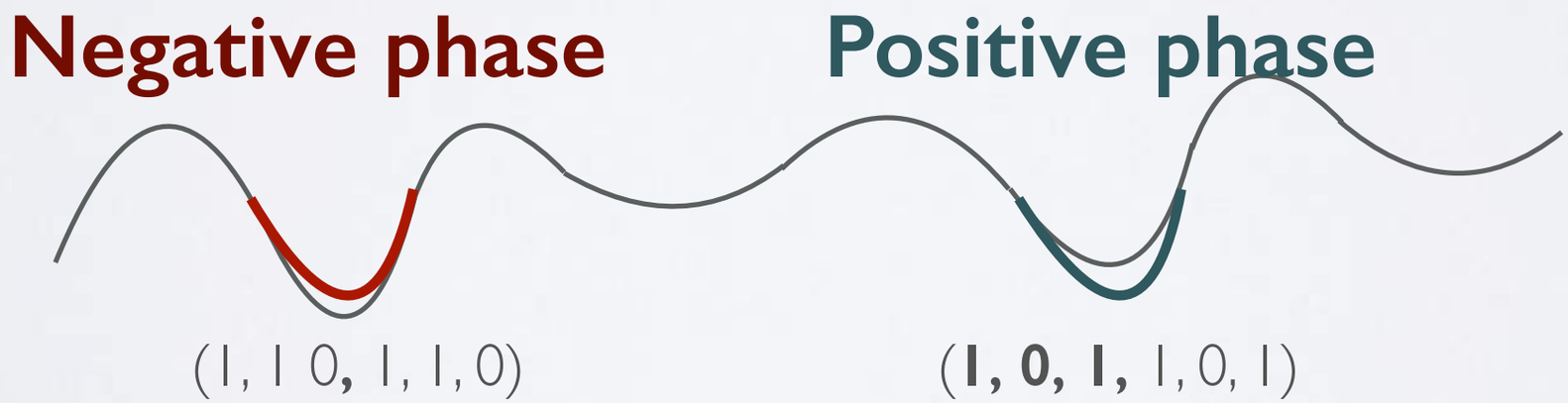
- Positive phase
  - Clamp a data vector on visible units
  - Update hidden units one at a time until the network reaches thermal equilibrium
  - Sample $s_i s_j$
- Negative phase
  - Update all units one at a time (until equilibrium)
  - Sample $s_i s_j$



Visible			Hidden		-E
1	0	0	1	1	4
1	0	0	1	0	3
1	0	0	0	1	2
1	0	0	0	0	3

WHY DO WE NEED NEGATIVE TERM?

- Need **unlearning** to avoid spurious memory
  - Positive phase: Find hidden configurations that work well with v
  - Negative phase: Find the joint configurations that are the best competitors

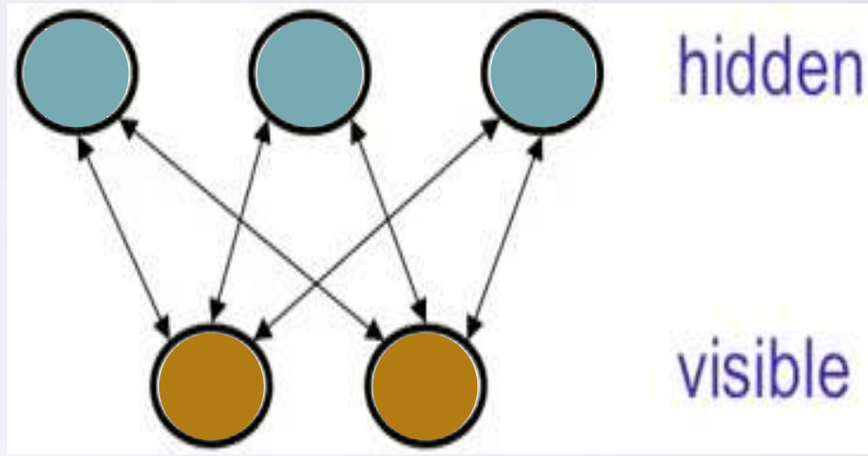


BETTER WAY FOR TRAINING

- Start from whatever state you visited (called **particle**)
- If we were at equilibrium before & if we changed the weights a little
- Only need a few updates to get back to equilibrium
- Didn't work well with mini-batches → Led to ideas of parallel update

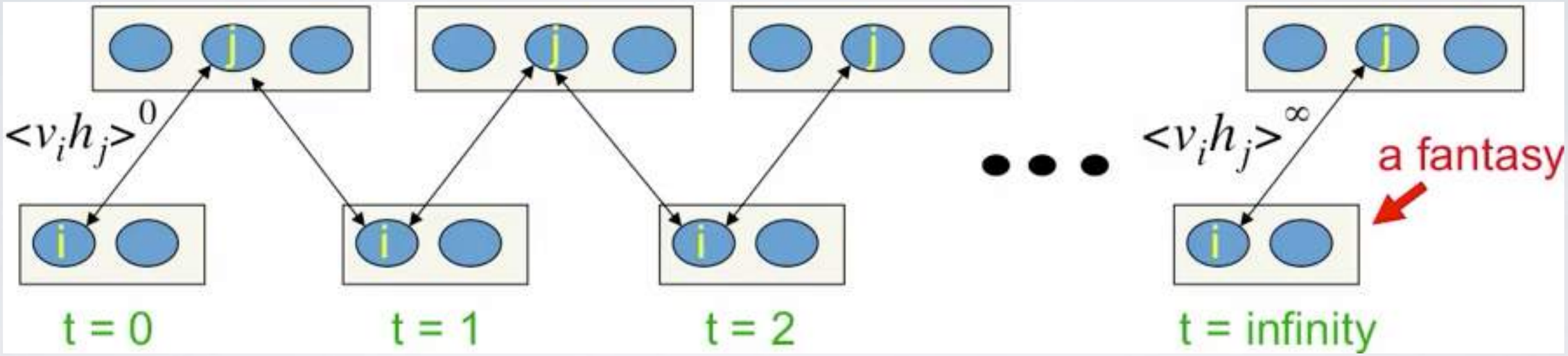
RESTRICTED BOLTZMANN MACHINE

- Restrict the connectivity to make learning easier
  - Only one layer of hidden units
  - No connection between hidden units



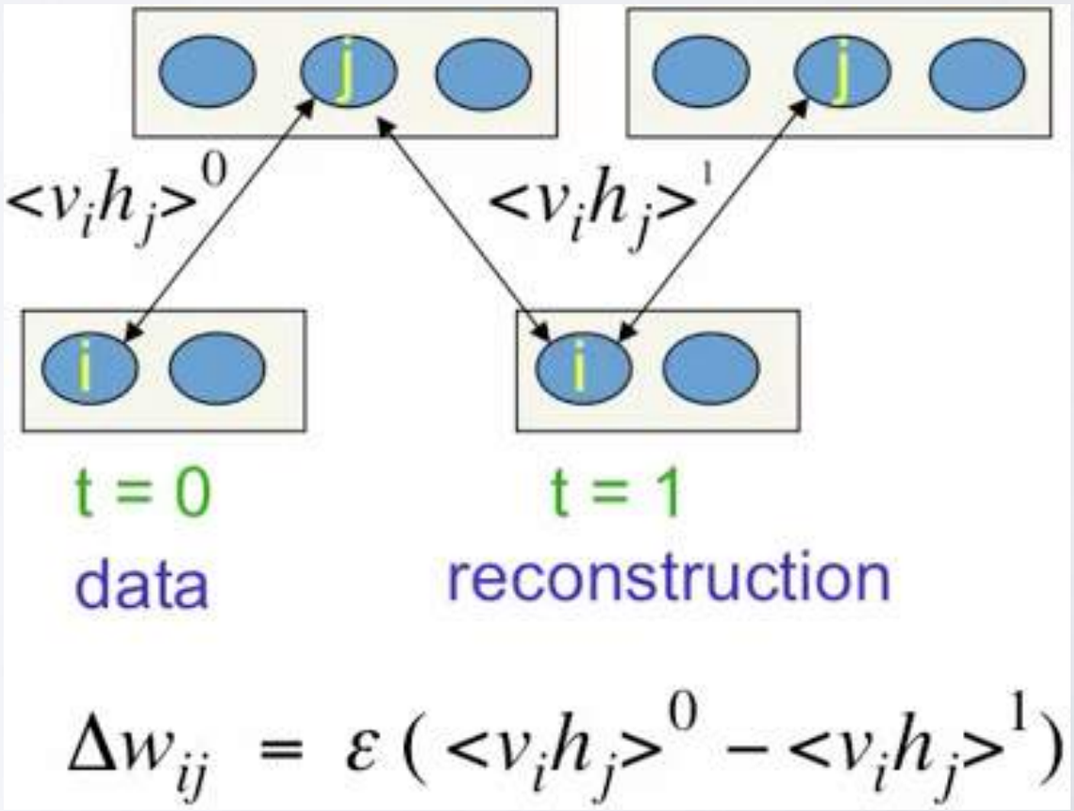
- For positive phase, RBM takes only one step to reach thermal equilibrium

SHORT-CUT: CONTRASTIVE DIVERGENCE

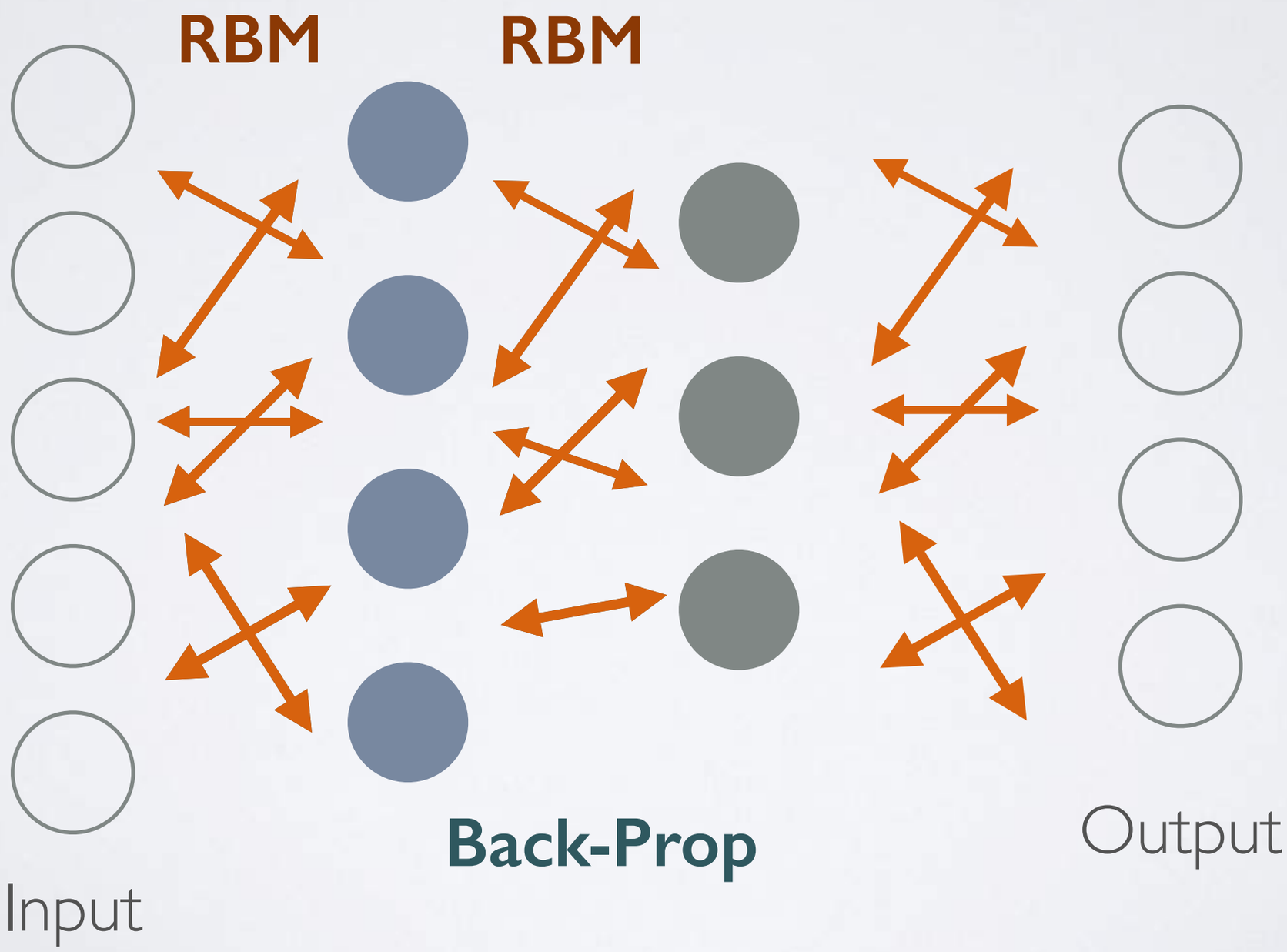


Expedite negative phase

"Don't need to get to equilibrium"



REVISITED:
WHAT CAN WE DO WITH RBM?



“Thank you.”

– Juno @ AKA STUDY