Yoshua Bengio
Statistical Learning Algorithms Canada Research Chair, U. Montreal

ERMITES 2011
Ecole de Recherche Multimodale d'Information - Techniques & Sciences
27-29 septembre 2011, Ile de Porquerolles, France

APPRENTISSAGE NON-SUPERVISÉ DE REPRÉSENTATIONS PROFONDES

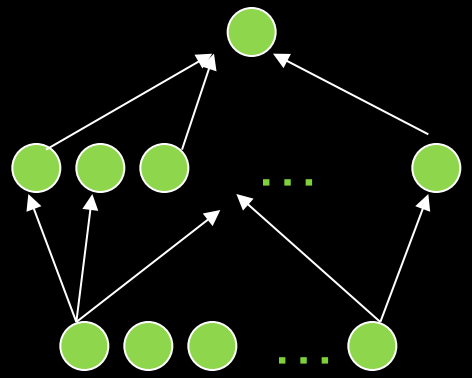
From AI to Deep Learning

- Intelligence requires knowledge
- Knowledge can be implicit
- Explicitly providing knowledge failed (expert systems)
- Learning captures knowledge from data
- Real-world distributions have convoluted unknown structure, not all captured by the principle of local generalization
- Deep Learning: a way to address this by the discovery of multiple levels of representation capturing the underlying factors of variation

What is Generalizing?

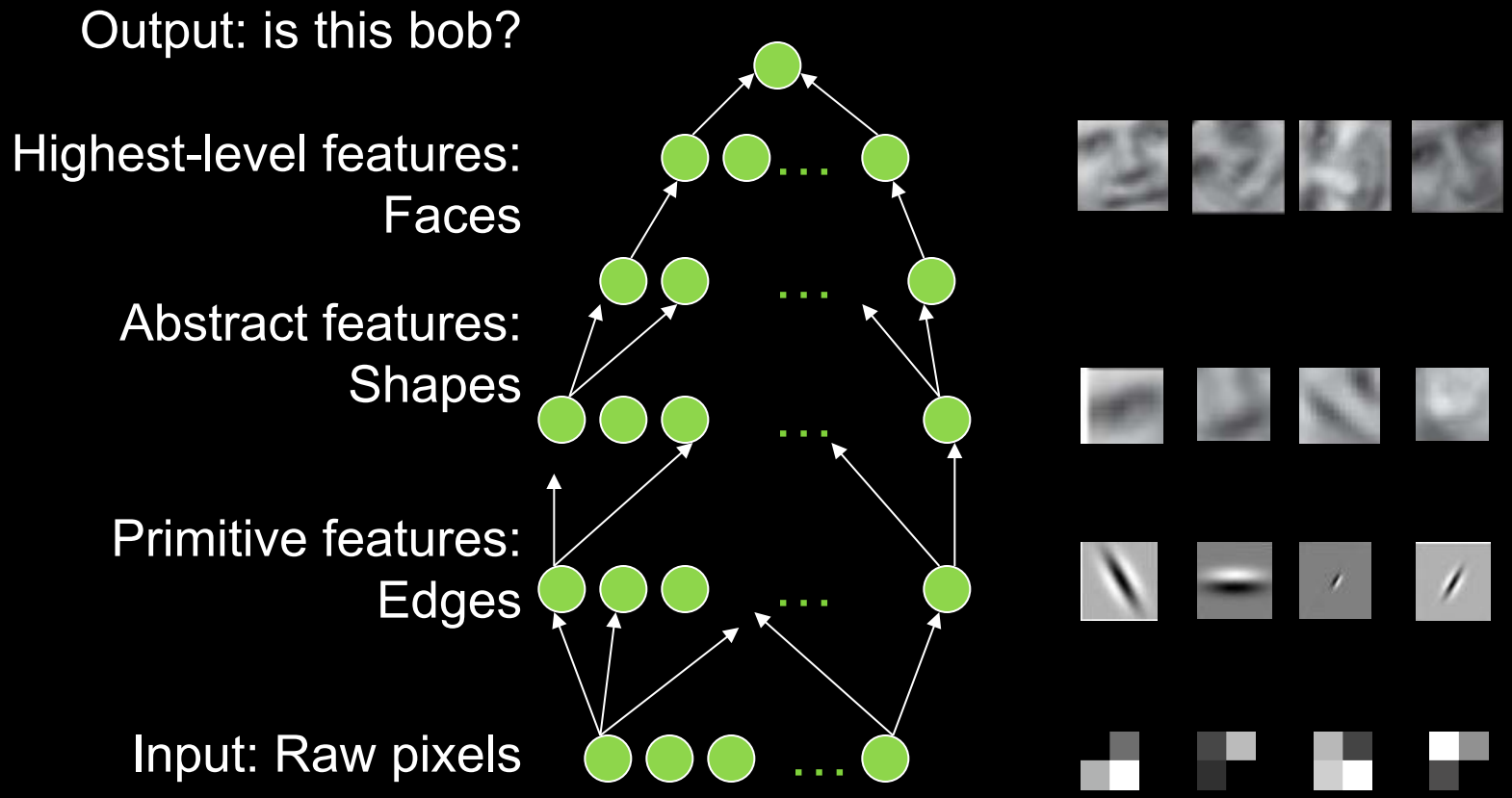
- Capturing dependencies between random variables
- Spreading out the probability mass from the empirical distribution. Where???
- Discovering underlying abstractions / explanatory factors

Shallow learning architecture

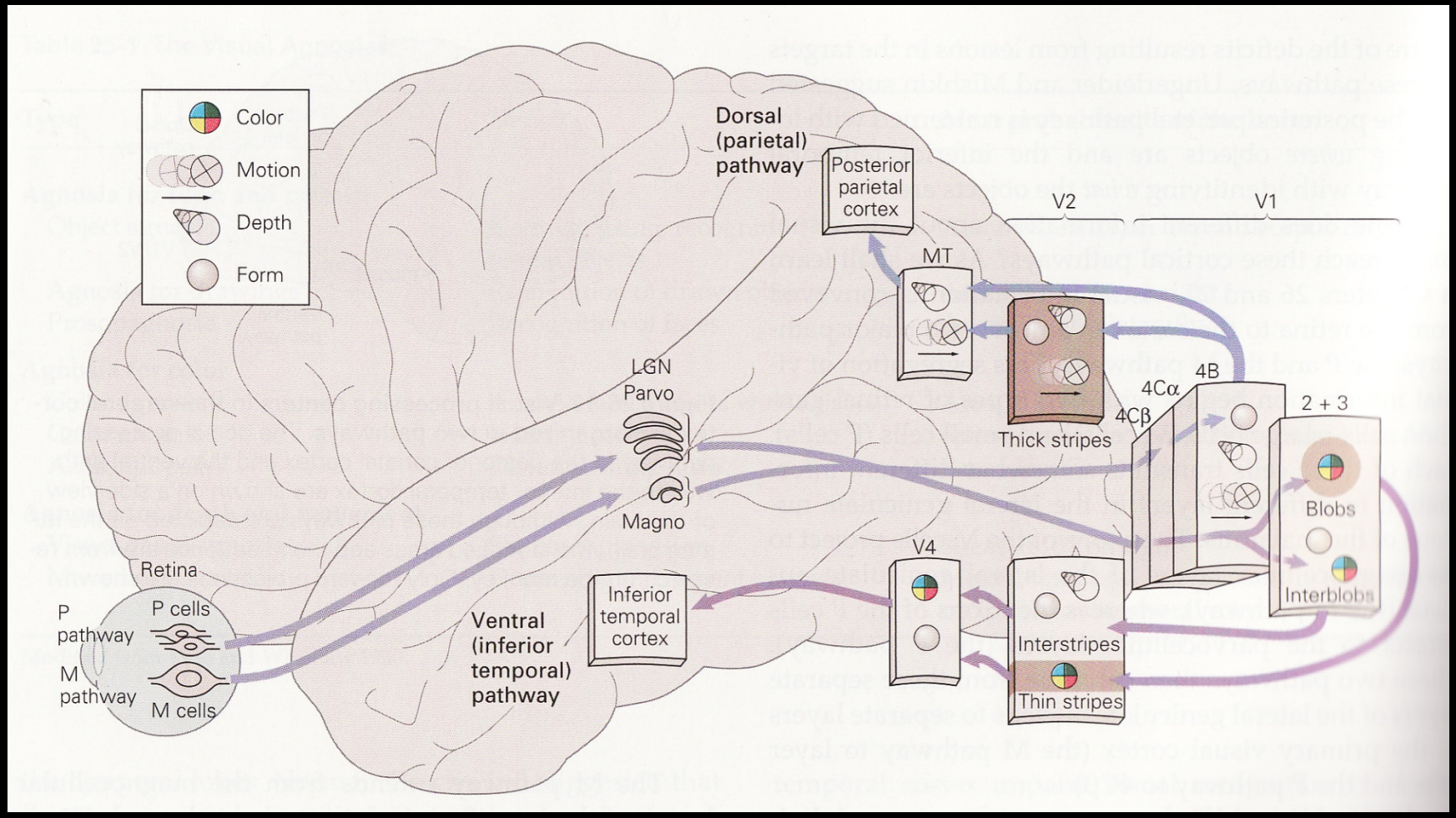


1-layer NNet, SVM, GP predictor, decision tree, boosted stumps, etc.

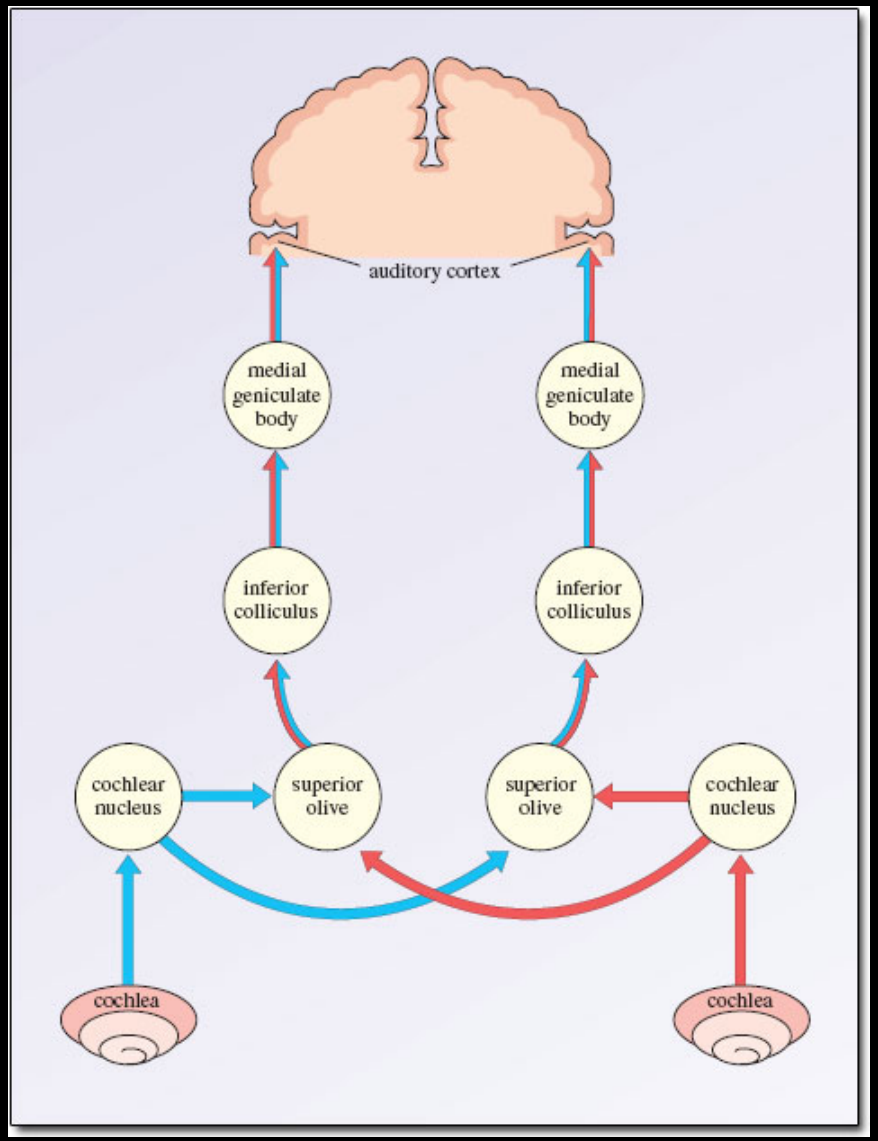
Deep learning architecture

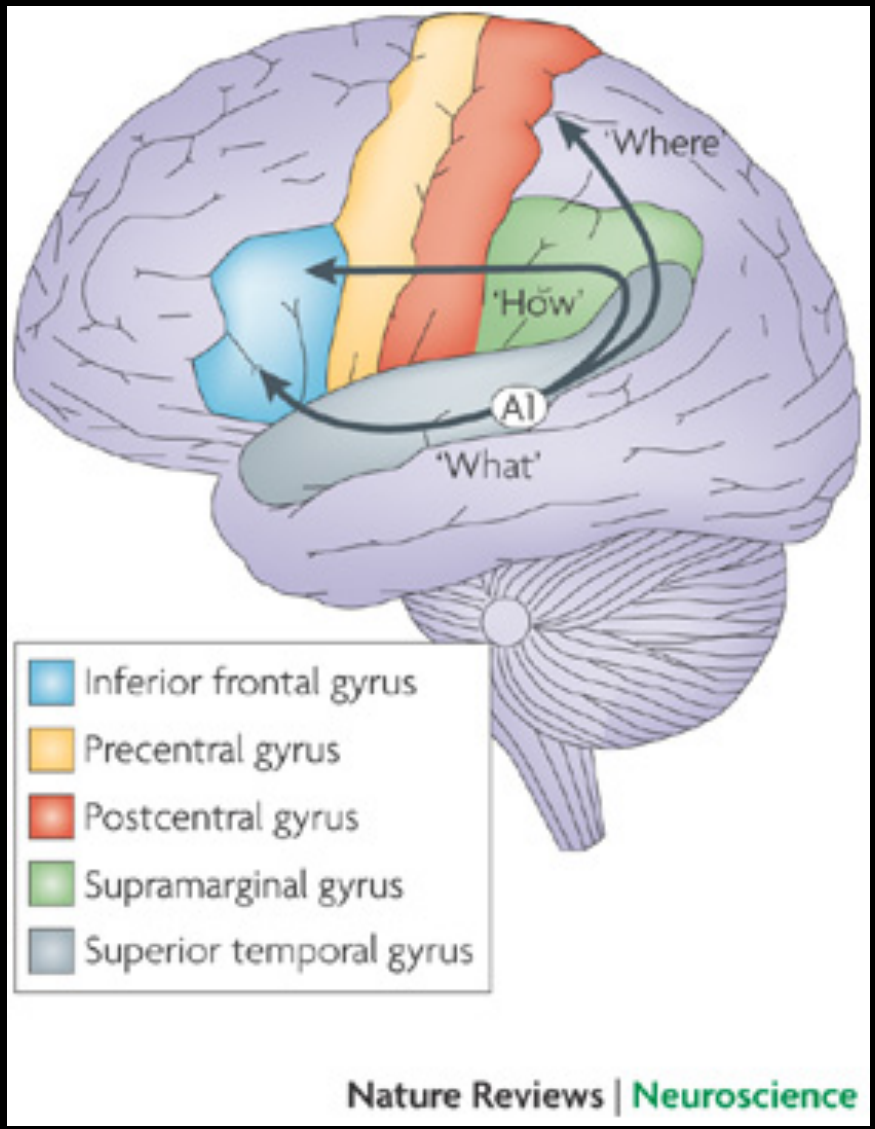


Visual System



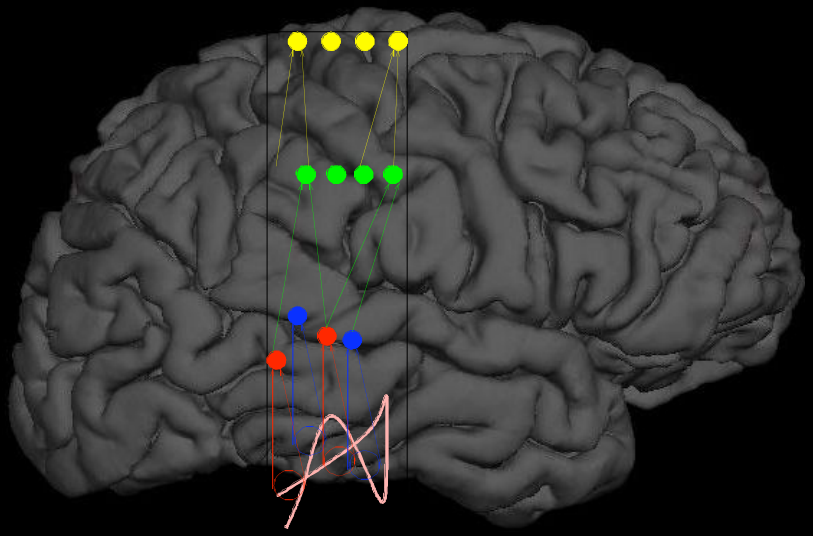
Auditory System

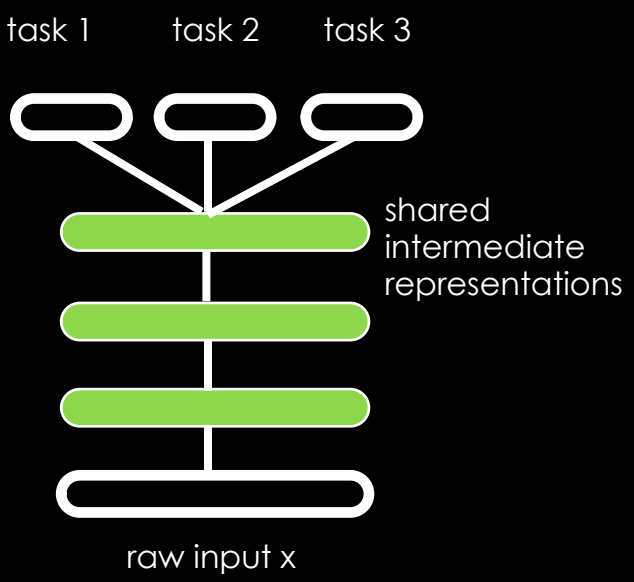




Deep Motivations

- Brains have a deep architecture
- Humans' ideas composed from simpler ones
- Insufficient depth can be exponentially inefficient
- Distributed (possibly sparse) representations necessary for non-local generalization, exponentially more efficient than 1-of-N enumeration of latent variable values
- Multiple levels of latent variables allow combinatorial sharing of statistical strength

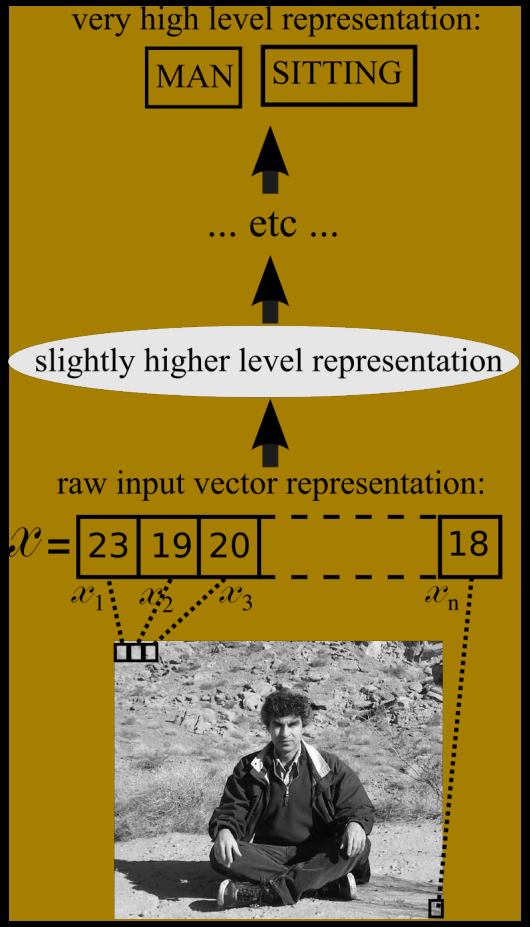




Deep Architecture in our Mind

- Humans organize their ideas and concepts hierarchically
- Humans first learn simpler concepts and then compose them to represent more abstract ones
- Engineers break-up solutions into multiple levels of abstraction and processing
- It would be nice to learn / discover these concepts

(knowledge engineering failed because of limits of introspection?)



Deep Learning Hypotheses

- Hypothesis 1: deep hierarchy of features useful to efficiently represent and learn complex abstractions needed for AI and mammal intelligence.
  - **Computational & statistical efficiency**

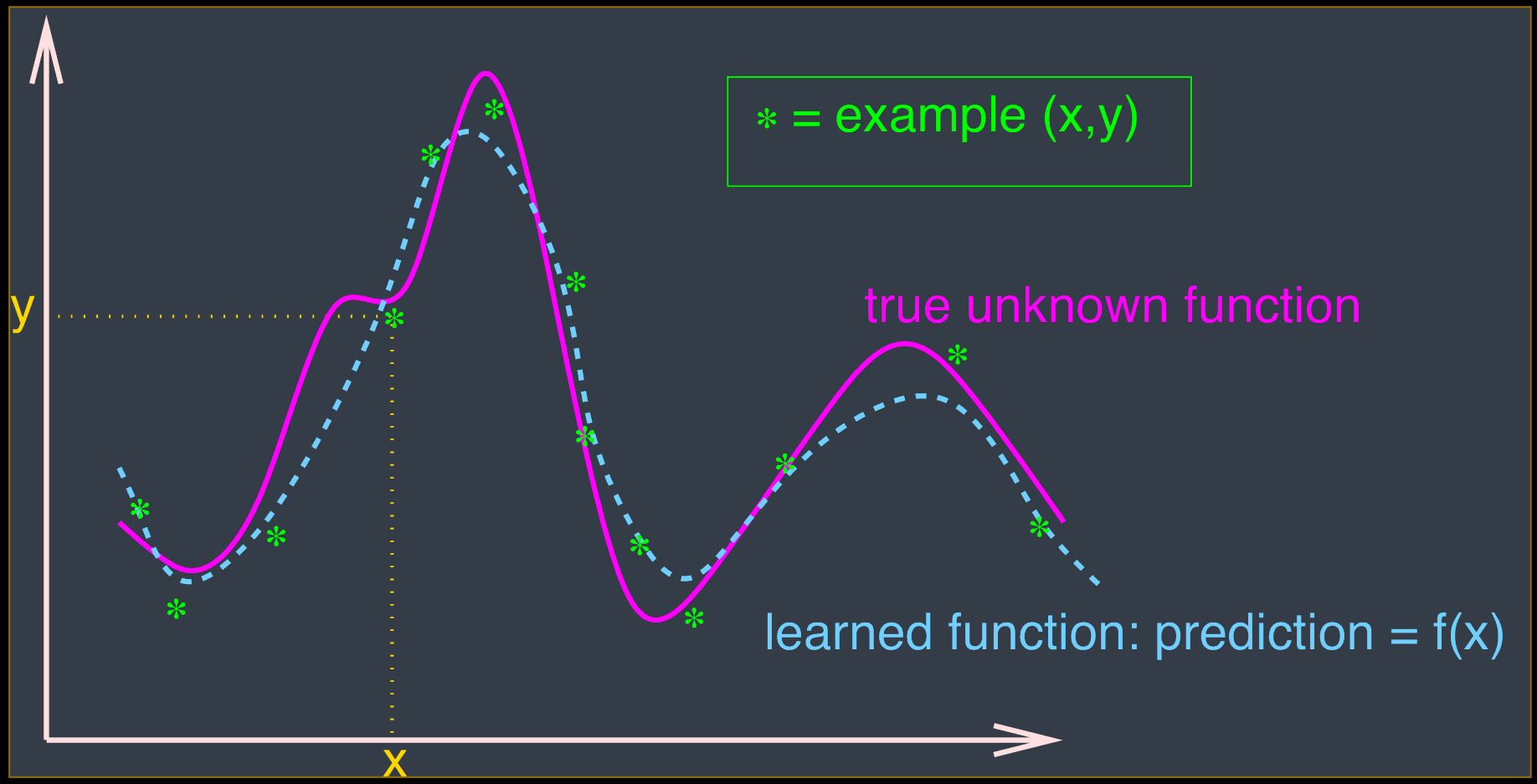
- Hypothesis 2: unsupervised **learning of representations** is a crucial component of the solution.
  - **Optimization & regularization.**

Even if we had labels for all examples

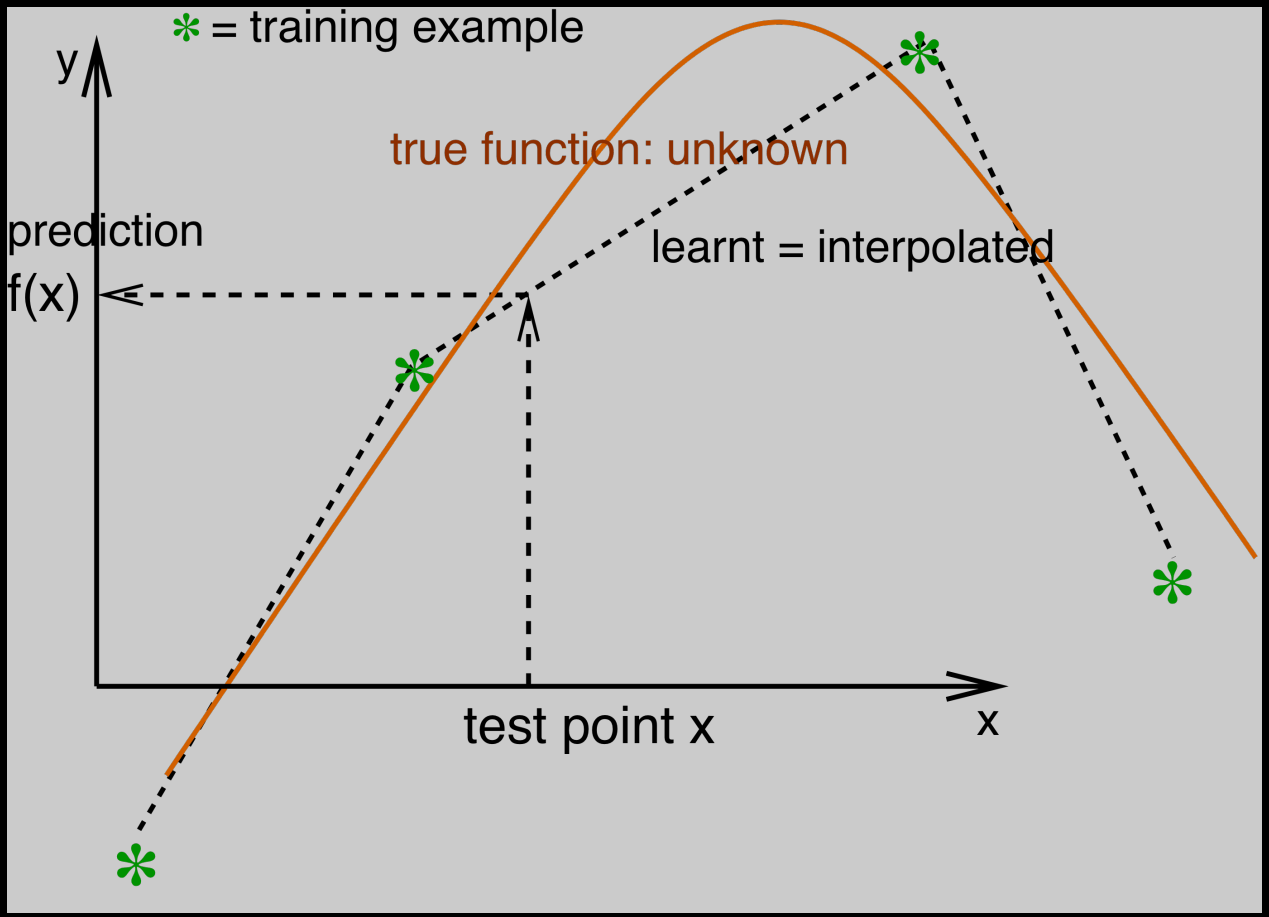
- Theoretical and ML-experimental support for both.

Challenge #1: Non-local learning of the interactions of many factors of variation

Easy Learning



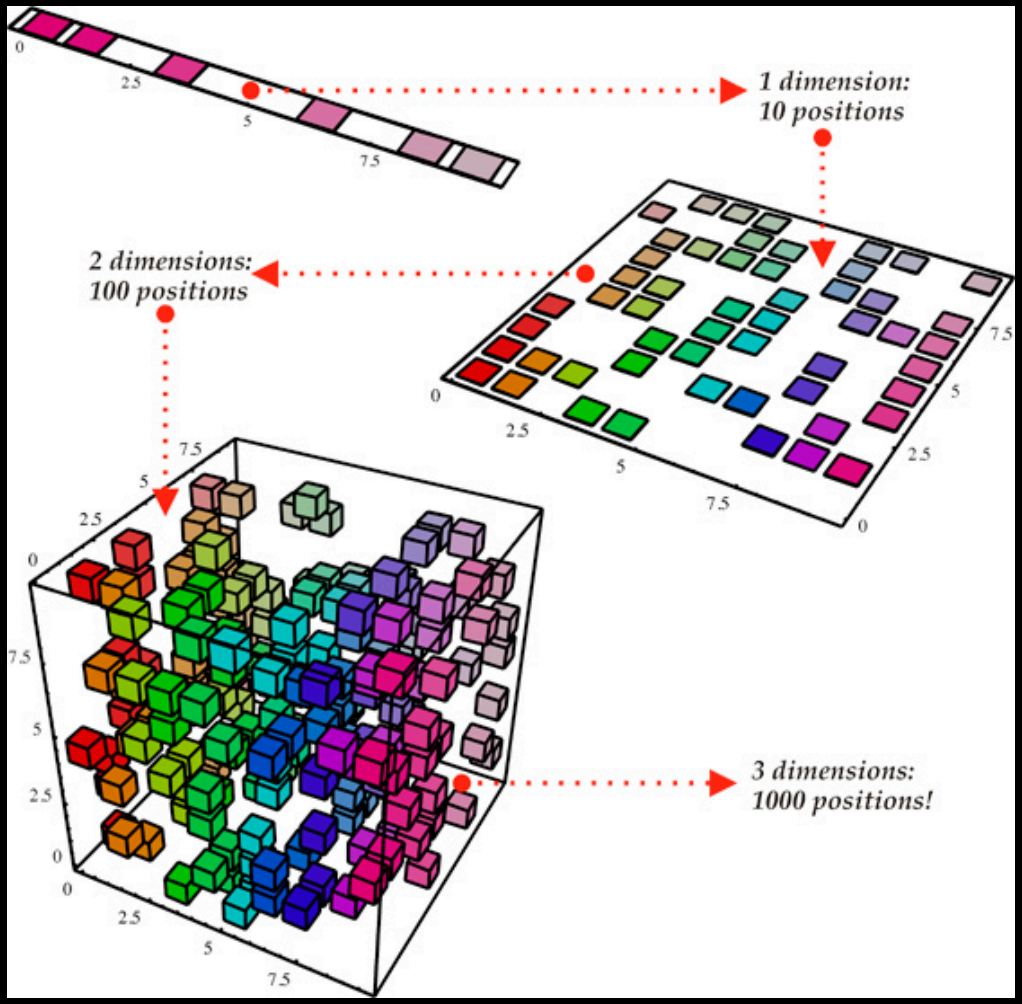
Principle of Local Generalization



The Curse of Dimensionality

To generalize locally, need representative examples for all relevant variations!

Classical solution: hope for a smooth enough target function

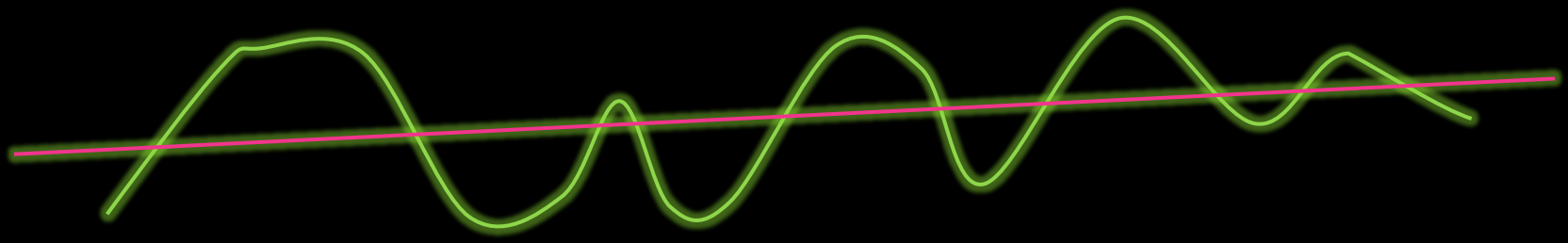


Limits of Local Generalization: Theoretical Results


(Bengio, Delalleau & Le Roux 2007)

e.g. Gaussian (RBF) SVM

- Theorem: Gaussian kernel machines need at least *k* examples to learn a function that has *2k* zero-crossings along some line



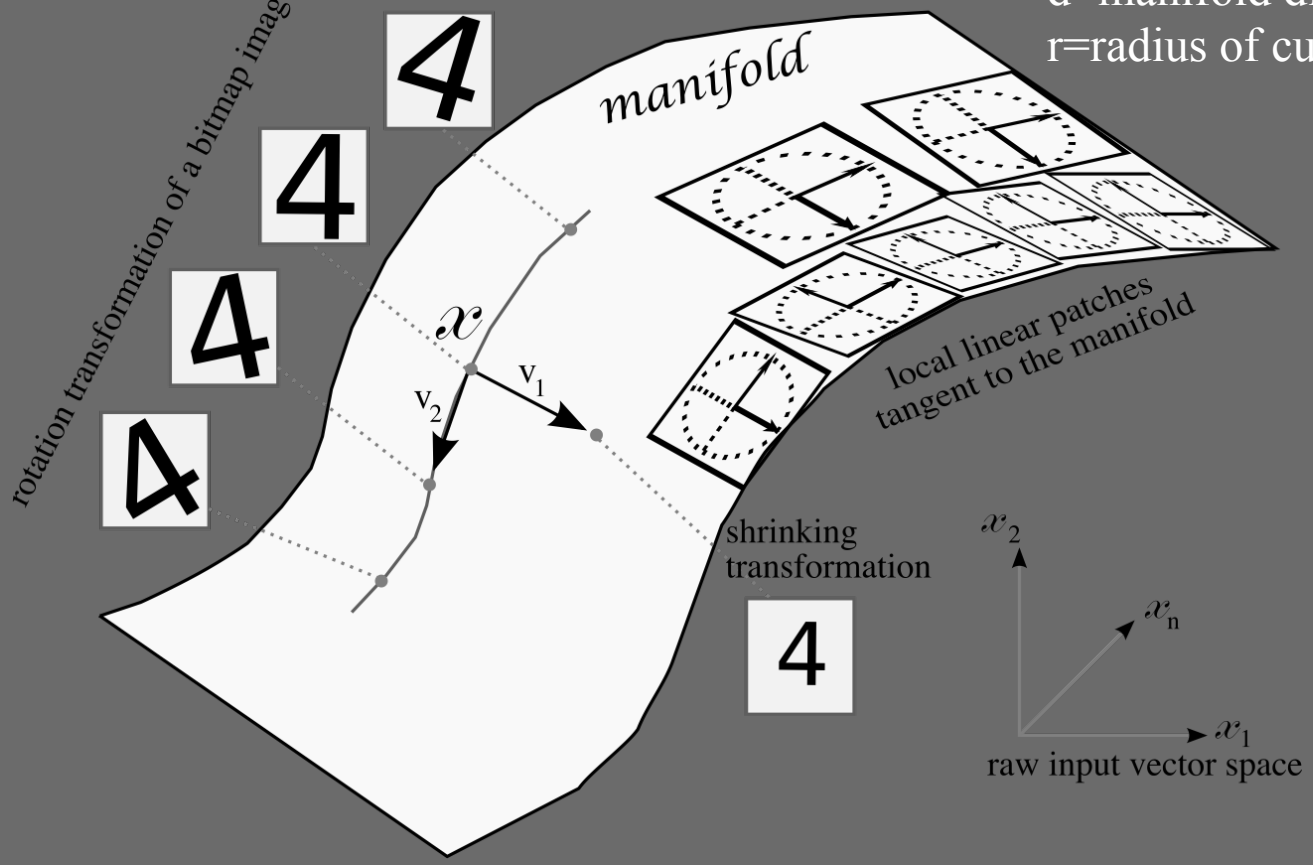
- Theorem: For a Gaussian kernel machine to learn some maximally varying functions over *d* inputs requires $O(2^d)$ examples

Curse of Dimensionality When Generalizing Locally on a Manifold

(Bengio et al 2006)

$O(d\ r^d)$ examples are needed

d=manifold dim.
r=radius of curvature



How to Beat the Curse of Many Factors of Variation?

Compositionality: exponential gain in representational power

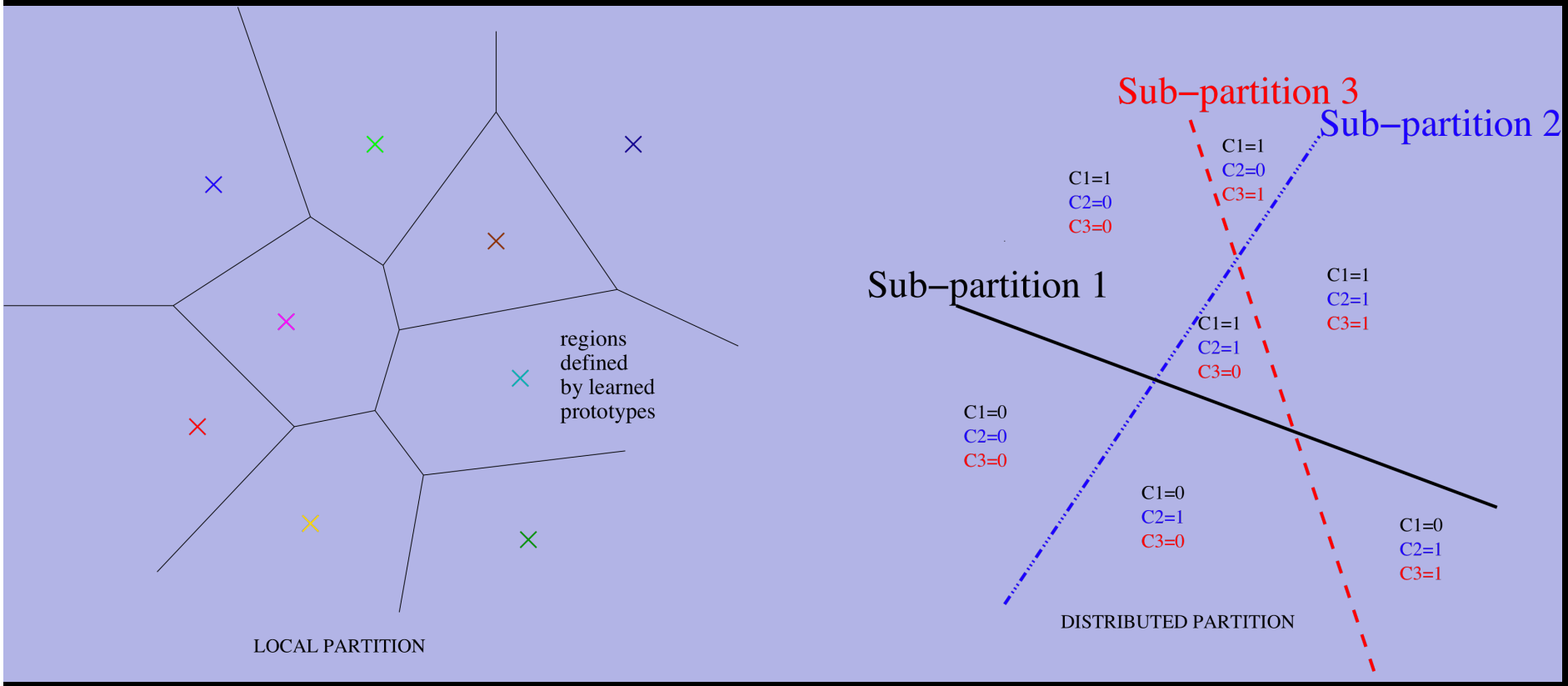
- Distributed representations / embeddings: feature learning
- Deep architecture: multiple levels of feature learning

Can generalize to new configurations

Distributed Representations

- Many neurons active simultaneously
- Input represented by the activation of a set of features that are not mutually exclusive
- Can be exponentially more efficient than local representations
- = FEATURE LEARNING instead of only manual feature-engineering

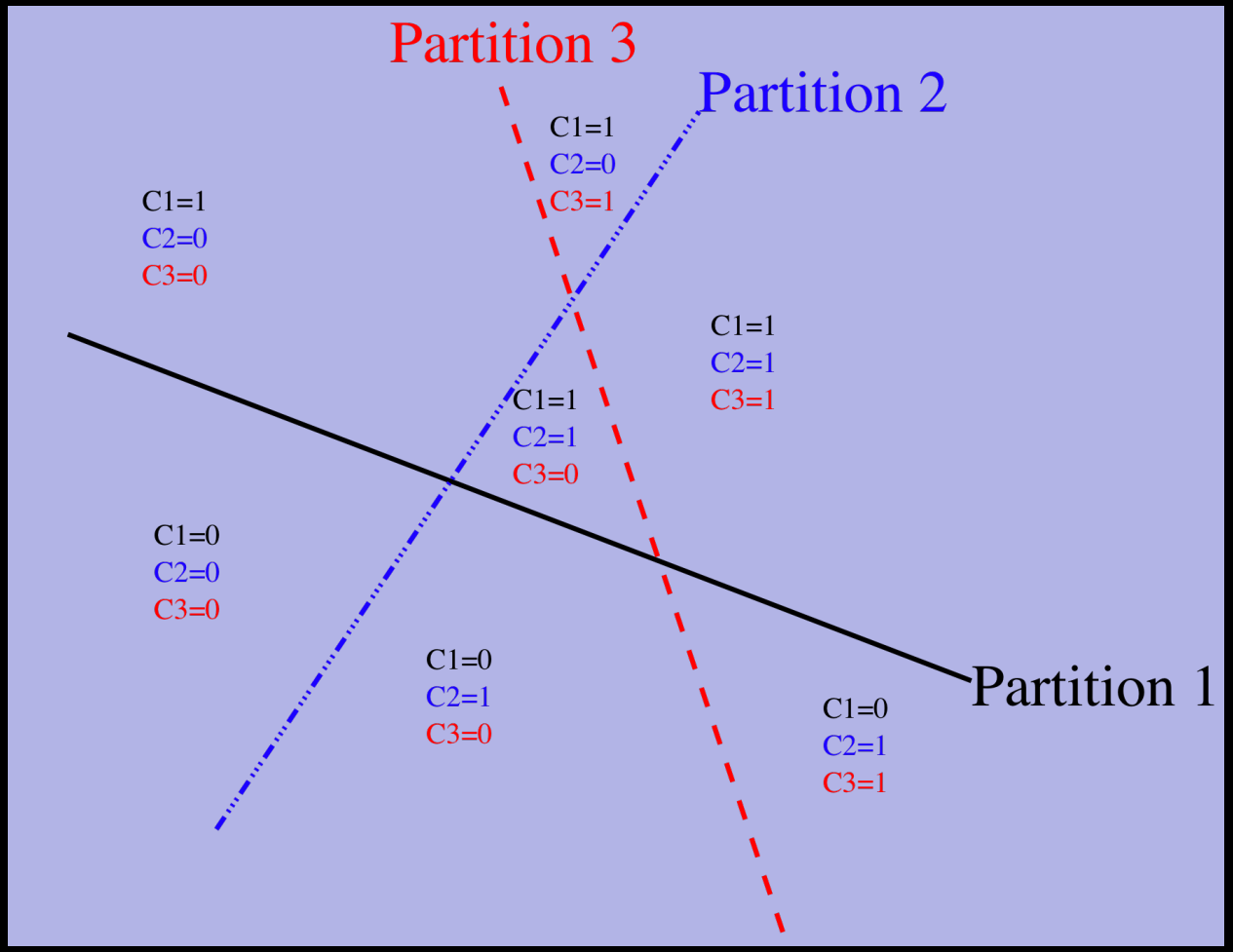
Local vs Distributed Latent Variables

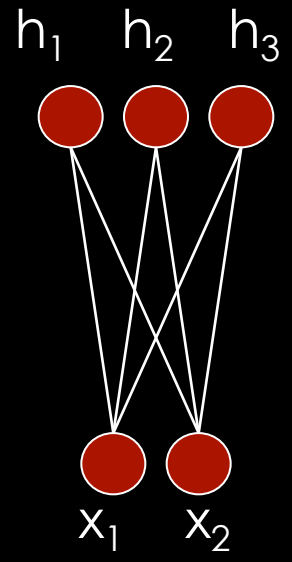


Clustering

Multi-clustering

RBM Hidden Units Carve Input Space



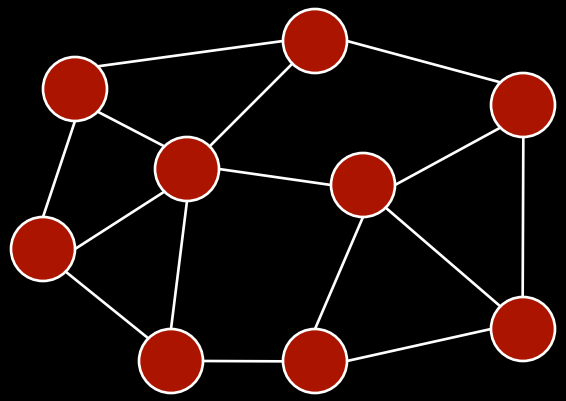


Boltzman Machines and MRFs

- Boltzmann machines:
  (Hinton 84)

$$P(x) = \frac{1}{Z} e^{-\text{Energy}(x)} = \frac{1}{Z} e^{c^T x + x^T W x}$$

- Markov Random Fields:



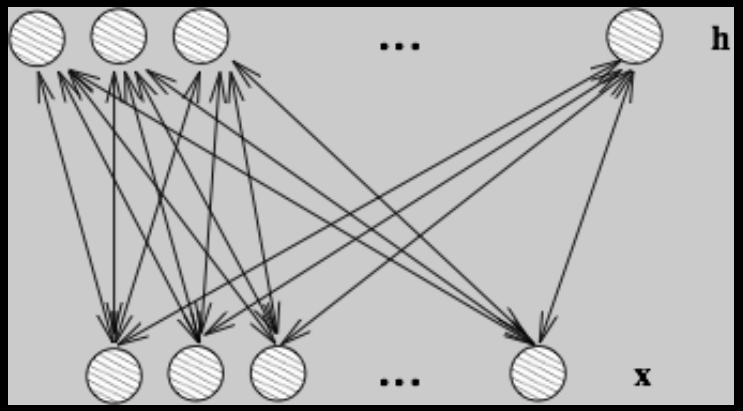
$$P(x) = \frac{1}{Z} e^{\sum_i w_i f_i(x)}$$

- More interesting with latent variables!

Restricted Boltzmann Machine

- The most popular building block for deep architectures

$$P(x,h) = \frac{1}{Z} e^{b^T h + c^T x + h^T W x}$$



Hidden units

Visible units

- Bipartite undirected graphical model
- Inference is trivial:

P(h | x) & *P(x | h)* factorize

RBM Conditionals Factorize

$$
\begin{aligned}
P(\mathbf{h}|\mathbf{x}) &= \frac{\exp(\mathbf{b}'\mathbf{x} + \mathbf{c}'\mathbf{h} + \mathbf{h}'W\mathbf{x})}{\sum_{\tilde{\mathbf{h}}} \exp(\mathbf{b}'\mathbf{x} + \mathbf{c}'\tilde{\mathbf{h}} + \tilde{\mathbf{h}}'W\mathbf{x})} \\
&= \frac{\prod_i \exp(\mathbf{c}_i\mathbf{h}_i + \mathbf{h}_i W_i\mathbf{x})}{\prod_i \sum_{\tilde{\mathbf{h}}_i} \exp(\mathbf{c}_i\tilde{\mathbf{h}}_i + \tilde{\mathbf{h}}_i W_i\mathbf{x})} \\
&= \prod_i \frac{\exp(\mathbf{h}_i(\mathbf{c}_i + W_i\mathbf{x}))}{\sum_{\tilde{\mathbf{h}}_i} \exp(\tilde{\mathbf{h}}_i(\mathbf{c}_i + W_i\mathbf{x}))} \\
&= \prod_i P(\mathbf{h}_i|\mathbf{x}).
\end{aligned}
$$

RBM Energy Gives Binomial Neurons

With $\mathbf{h}_i \in \{0, 1\}$, recall $\text{Energy}(\mathbf{x}, \mathbf{h}) = -\mathbf{b}'\mathbf{x} - \mathbf{c}'\mathbf{h} - \mathbf{h}'W\mathbf{x}$

$$
\begin{aligned}
P(\mathbf{h}_i = 1|\mathbf{x}) &= \frac{e^{1\mathbf{c}_i + 1W_i\mathbf{x} + other\ terms}}{e^{1\mathbf{c}_i + 1W_i\mathbf{x} + other\ terms} + e^{0\mathbf{c}_i + 0W_i\mathbf{x} + other\ terms}} \\
&= \frac{e^{\mathbf{c}_i + W_i\mathbf{x}}}{e^{\mathbf{c}_i + W_i\mathbf{x}} + 1} \\
&= \frac{1}{1 + e^{-\mathbf{c}_i - W_i\mathbf{x}}} \\
&= \text{sigm}(\mathbf{c}_i + W_i\mathbf{x}).
\end{aligned}
$$

since $\text{sigm}(a) = \frac{1}{1+e^{-a}}$.

RBM Free Energy

$$P(\mathbf{x}, \mathbf{h}) = \frac{e^{-\text{Energy}(\mathbf{x},\mathbf{h})}}{Z}$$

- Free Energy = equivalent energy when marginalizing

$$P(\mathbf{x}) = \sum_{\mathbf{h}} \frac{e^{-\text{Energy}(\mathbf{x},\mathbf{h})}}{Z} = \frac{e^{-\text{FreeEnergy}(\mathbf{x})}}{Z}$$

- Can be computed exactly and efficiently in RBMs

$$\text{FreeEnergy}(x) = -c^T\mathbf{x} - \sum_i \log \sum_{\mathbf{h}_i} e^{\mathbf{h}_i(b_i + W_i\mathbf{x})}$$

- Marginal likelihood *P(**x**)* tractable up to partition function *Z*

Factorization of the Free Energy

Let the energy have the following general form:

$$\text{Energy}(\mathbf{x}, \mathbf{h}) = -\beta(\mathbf{x}) + \sum_i \gamma_i(\mathbf{x}, \mathbf{h}_i)$$

Then

$$\begin{aligned}
P(\mathbf{x}) &= \frac{1}{Z} e^{-\text{FreeEnergy}(\mathbf{x})} = \frac{1}{Z} \sum_{\mathbf{h}} e^{-\text{Energy}(\mathbf{x},\mathbf{h})} \\
&= \frac{1}{Z} \sum_{\mathbf{h}_1} \sum_{\mathbf{h}_2} \cdots \sum_{\mathbf{h}_k} e^{\beta(\mathbf{x}) - \sum_i \gamma_i(\mathbf{x},\mathbf{h}_i)} = \frac{1}{Z} \sum_{\mathbf{h}_1} \sum_{\mathbf{h}_2} \cdots \sum_{\mathbf{h}_k} e^{\beta(\mathbf{x})} \prod_i e^{-\gamma_i(\mathbf{x},\mathbf{h}_i)} \\
&= \frac{e^{\beta(\mathbf{x})}}{Z} \sum_{\mathbf{h}_1} e^{-\gamma_1(\mathbf{x},\mathbf{h}_1)} \sum_{\mathbf{h}_2} e^{-\gamma_2(\mathbf{x},\mathbf{h}_2)} \cdots \sum_{\mathbf{h}_k} e^{-\gamma_k(\mathbf{x},\mathbf{h}_k)} \\
&= \frac{e^{\beta(\mathbf{x})}}{Z} \prod_i \sum_{\mathbf{h}_i} e^{-\gamma_i(\mathbf{x},\mathbf{h}_i)}
\end{aligned}$$

$$\text{FreeEnergy}(\mathbf{x}) = -\log P(\mathbf{x}) - \log Z = -\beta(\mathbf{x}) - \sum_i \log \sum_{\mathbf{h}_i} e^{-\gamma_i(\mathbf{x},\mathbf{h}_i)}$$

Energy-Based Models Gradient

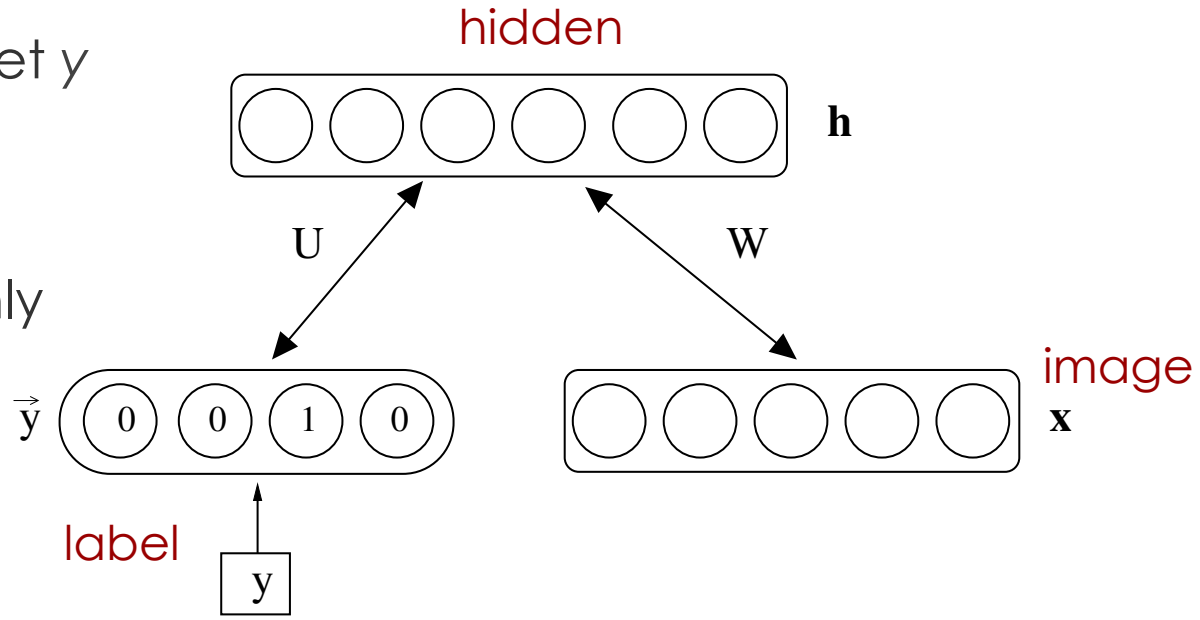
$$P(\mathbf{x}) = \frac{e^{-\text{Energy}(\mathbf{x})}}{Z} \qquad Z = \sum_{\mathbf{x}} e^{-\text{Energy}(\mathbf{x})}$$

$$\frac{\partial \log P(\mathbf{x})}{\partial \theta} = -\frac{\partial \text{Energy}(\mathbf{x})}{\partial \theta} - \frac{\partial \log Z}{\partial \theta}$$

$$\begin{aligned}
\frac{\partial \log Z}{\partial \theta} &= \frac{\partial \log \sum_{\mathbf{x}} e^{-\text{Energy}(\mathbf{x})}}{\partial \theta} \\
&= \frac{1}{Z} \frac{\partial \sum_{\mathbf{x}} e^{-\text{Energy}(\mathbf{x})}}{\partial \theta} \\
&= -\frac{1}{Z} \sum_{\mathbf{x}} e^{-\text{Energy}(\mathbf{x})} \frac{\partial \text{Energy}(\mathbf{x})}{\partial \theta} \\
&= -\sum_{\mathbf{x}} P(\mathbf{x}) \frac{\partial \text{Energy}(\mathbf{x})}{\partial \theta}
\end{aligned}$$

RBM with (image, label) visible units

- Can predict a subset *y* of the visible units given the others *x*
- Exactly if *y* takes only few values
- Gibbs sampling o/w



Boltzmann Machine Gradient

$$P(x) = \frac{1}{Z}\sum_h e^{-\text{Energy}(x,h)} = \frac{1}{Z}e^{-\text{FreeEnergy}(x)}$$

- Gradient has two components:

"positive phase" "negative phase"

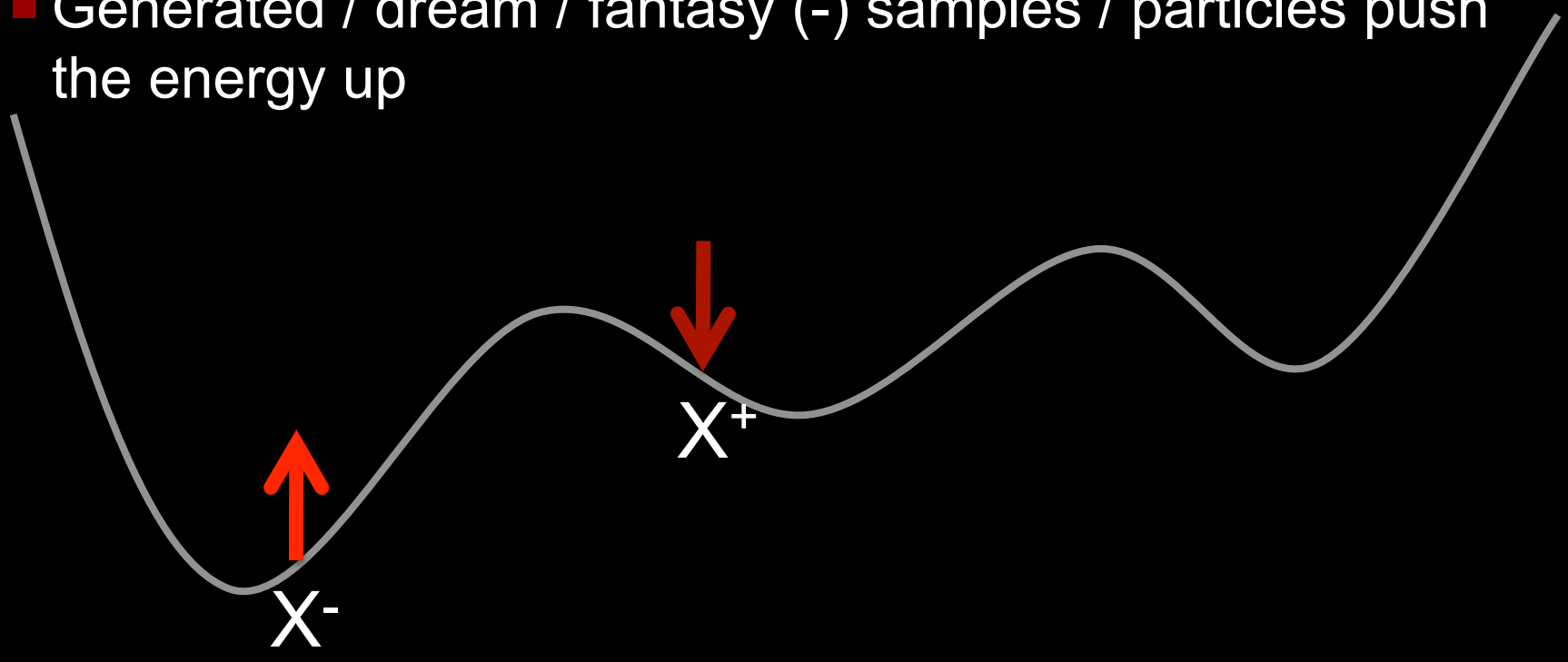
$$\frac{\partial \log P(x)}{\partial \theta} = -\frac{\partial \text{FreeEnergy}(x)}{\partial \theta} + \sum_{\tilde{x}} P(\tilde{x}) \frac{\partial \text{FreeEnergy}(\tilde{x})}{\partial \theta}$$

$$= -\sum_h P(h|x) \frac{\partial \text{Energy}(x,h)}{\partial \theta} + \sum_{\tilde{x},\tilde{h}} P(\tilde{x},\tilde{h}) \frac{\partial \text{Energy}(\tilde{x},\tilde{h})}{\partial \theta}$$

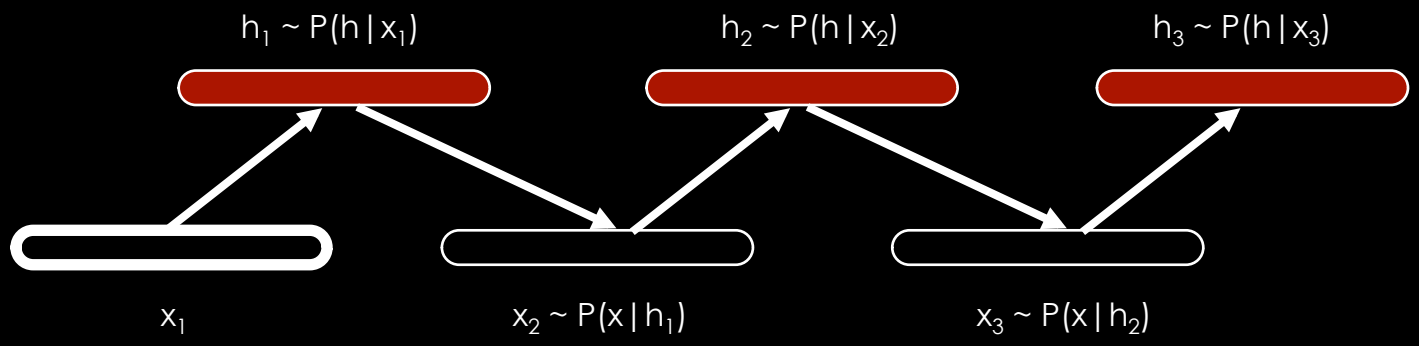
- In RBMs, easy to sample or sum over h | x
- Difficult part: sampling from P(x), typically with a Markov chain

Positive & Negative Samples

- Observed (+) examples push the energy down
- Generated / dream / fantasy (-) samples / particles push the energy up



Gibbs Sampling in RBMs



P(h | x) and P(x | h) factorize

$$P(x,h) = \frac{1}{Z} e^{b^T h + c^T x + h^T W x}$$

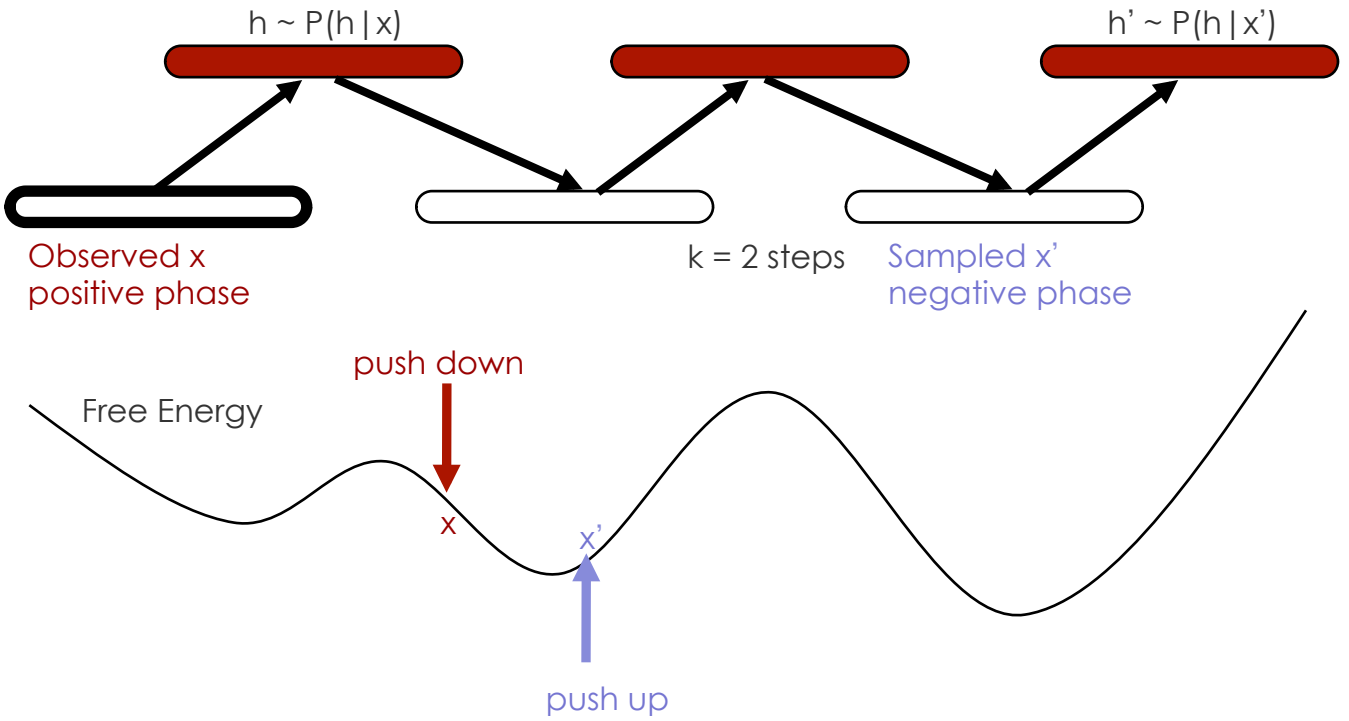
- Easy inference
- Convenient Gibbs sampling
  x→h→x→h…

Training RBMs

Contrastive Divergence: (CD-k)	start negative Gibbs chain at observed x, run k Gibbs steps
Persistent CD: (PCD)	run negative Gibbs chain in background while weights slowly change
Fast PCD:	two sets of weights, one with a large learning rate only used for negative phase, quickly exploring modes
Herding:	Deterministic near-chaos dynamical system defines both learning and sampling
Tempered MCMC:	use higher temperature to escape modes

Contrastive Divergence

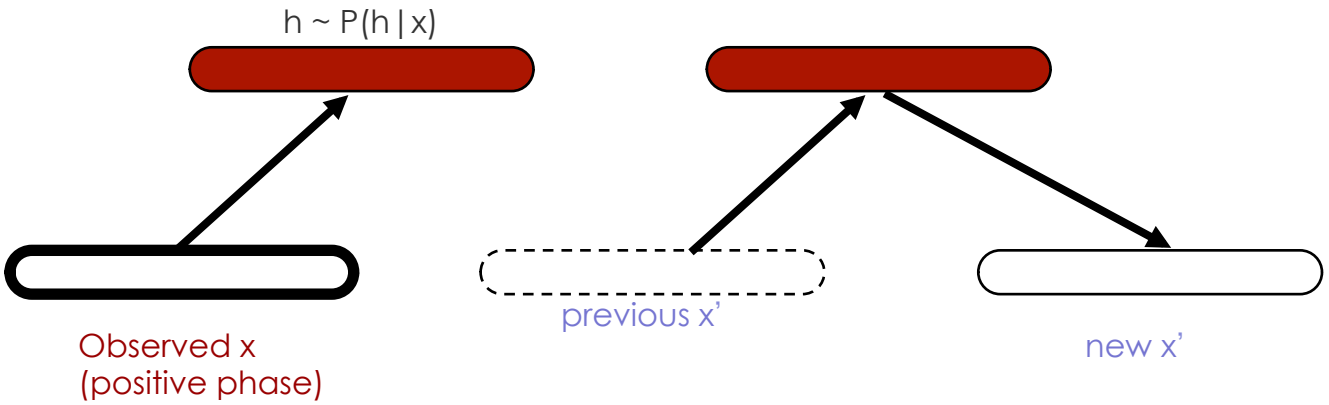
Contrastive Divergence (CD-k): start negative phase block Gibbs chain at observed x, run k Gibbs steps (Hinton 2002)



Persistent CD (PCD)

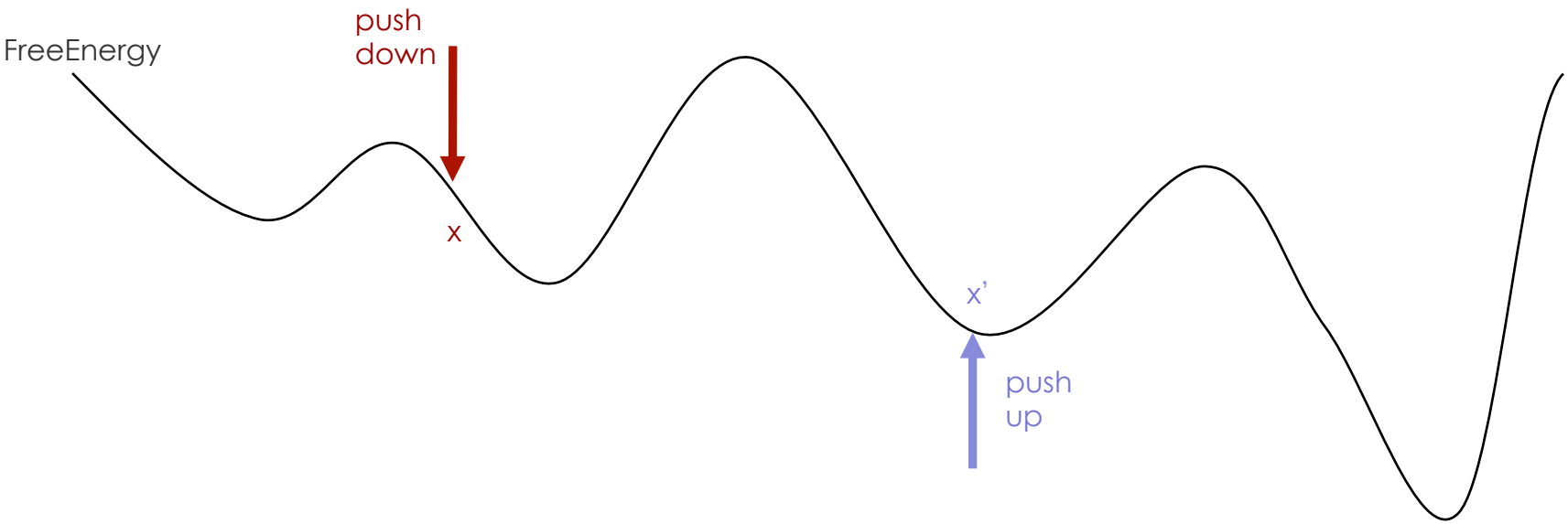
Run negative Gibbs chain in background while weights slowly change (Younes 2000, Tieleman 2008):

- Guarantees (Younes 89, 2000; Yuille 2004)
- If learning rate decreases in 1/t,
  chain mixes before parameters change too much,
  chain stays converged when parameters change



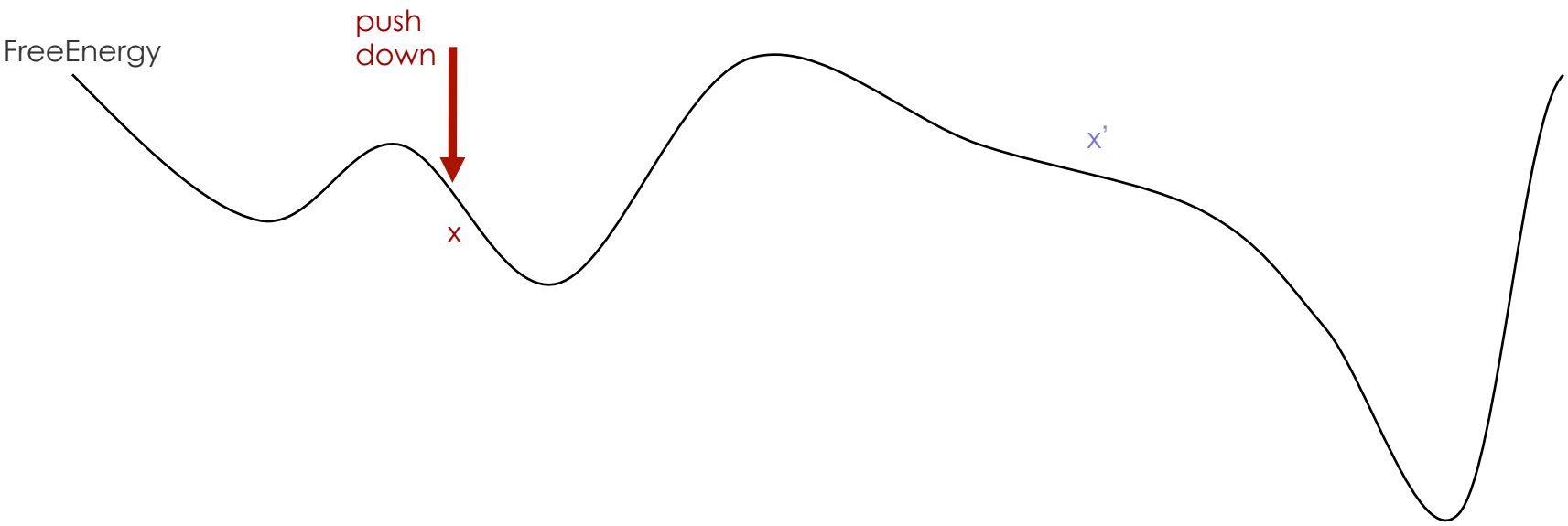
Persistent CD with large learning rate

Negative phase samples quickly push up the energy of wherever they are and quickly move to another mode



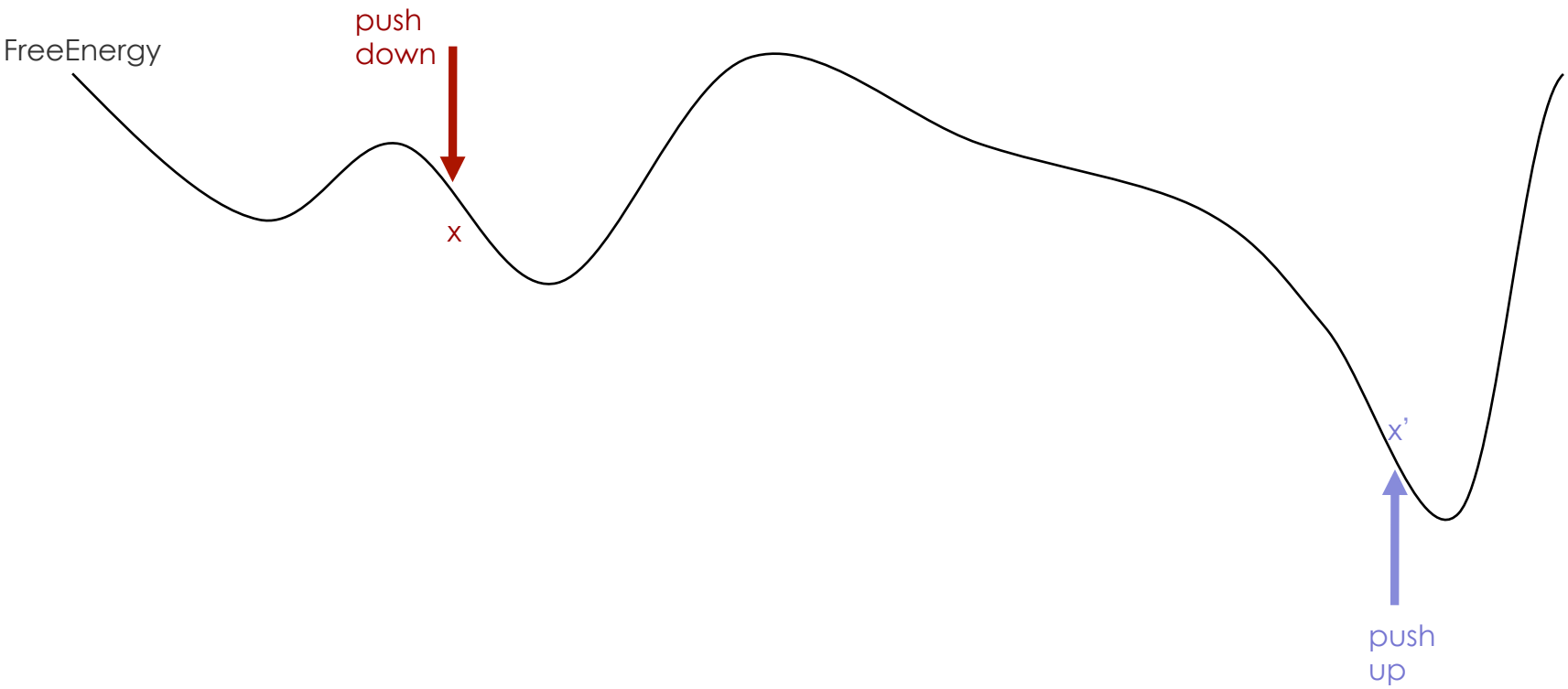
Persistent CD with large step size

Negative phase samples quickly push up the energy of wherever they are and quickly move to another mode



Persistent CD with large learning rate

Negative phase samples quickly push up the energy of wherever they are and quickly move to another mode



Challenge #2: Understanding the expressive power of deep architectures

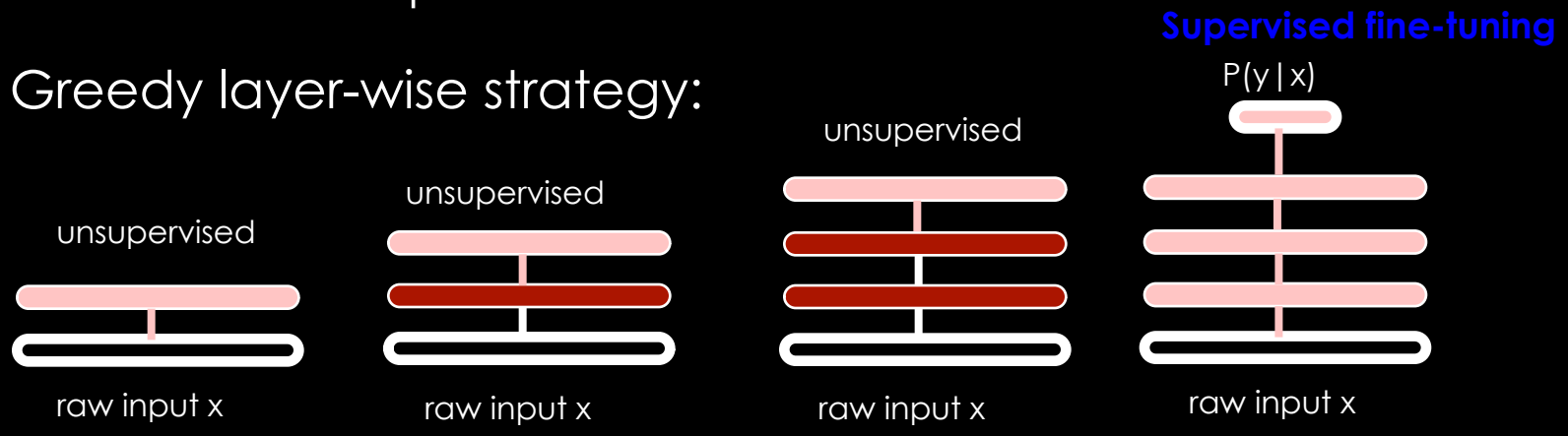
RBMs are Universal Approximators

(Le Roux & Bengio 2008, Neural Comp.)

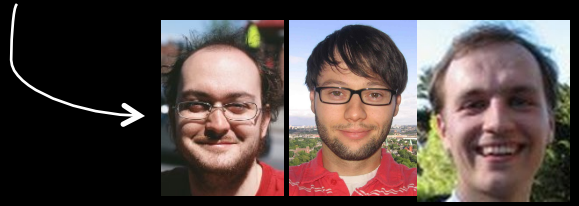
- Adding one hidden unit (with proper choice of parameters) guarantees increasing likelihood
- With enough hidden units, can perfectly model any discrete distribution
- RBMs with variable nb of hidden units = non-parametric

Unsupervised and Semi-Supervised Deep Feature Learning

- Classical: pre-process data with PCA = leading factors
- New: learning multiple levels of features/factors, often over-complete
- Greedy layer-wise strategy:

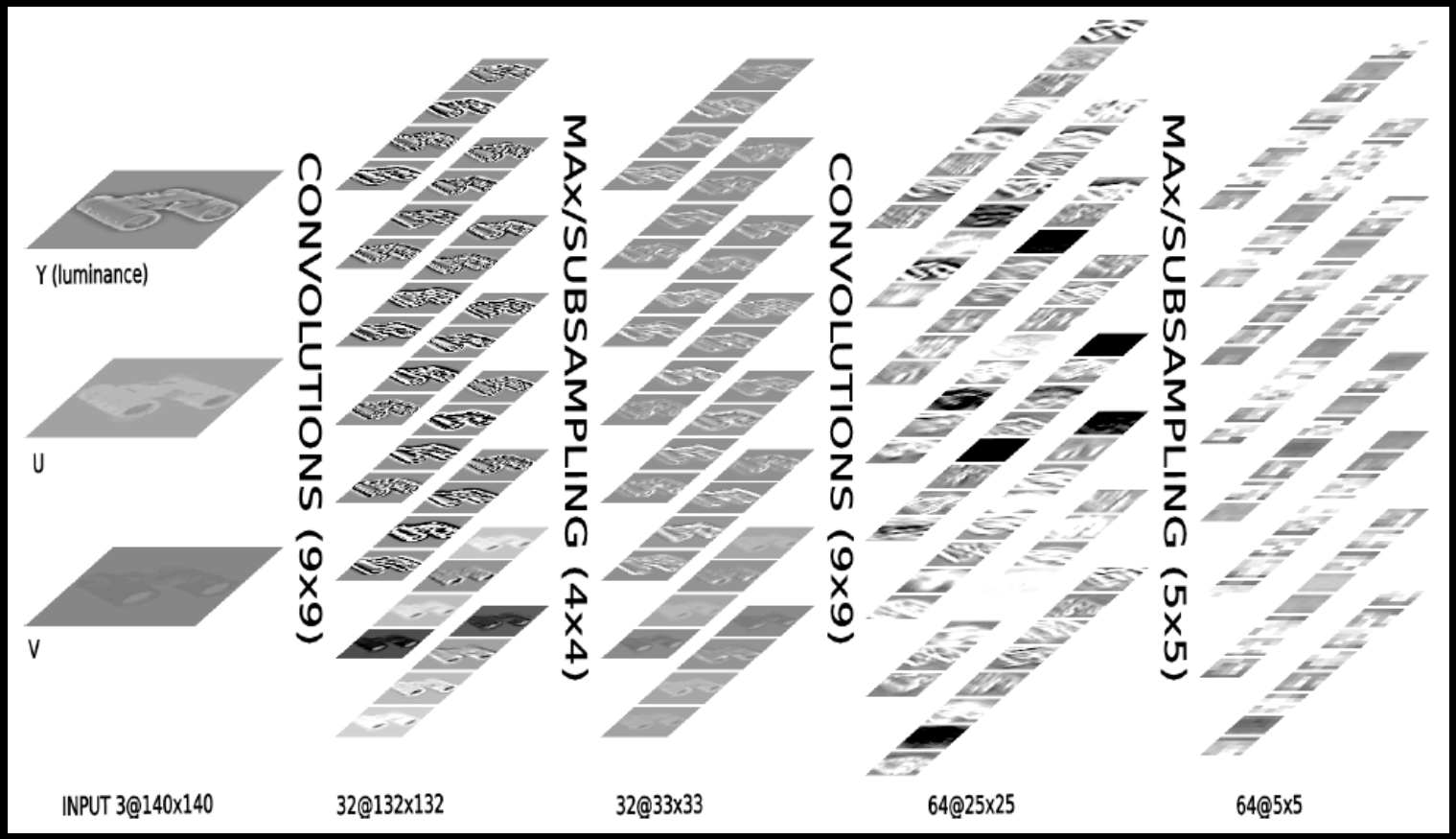


(Hinton et al 2006, Bengio et al 2007, Ranzato et al 2007)



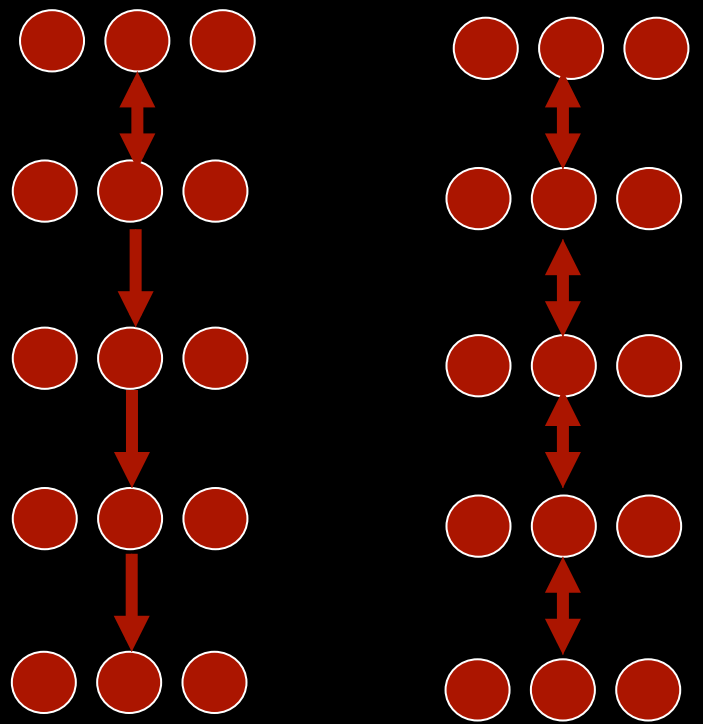
Deep Convolutional Architectures

Mostly from Le Cun's group (NYU), also Ng (Stanford):
state-of-the-art on MNIST digits, Caltech-101 objects, faces



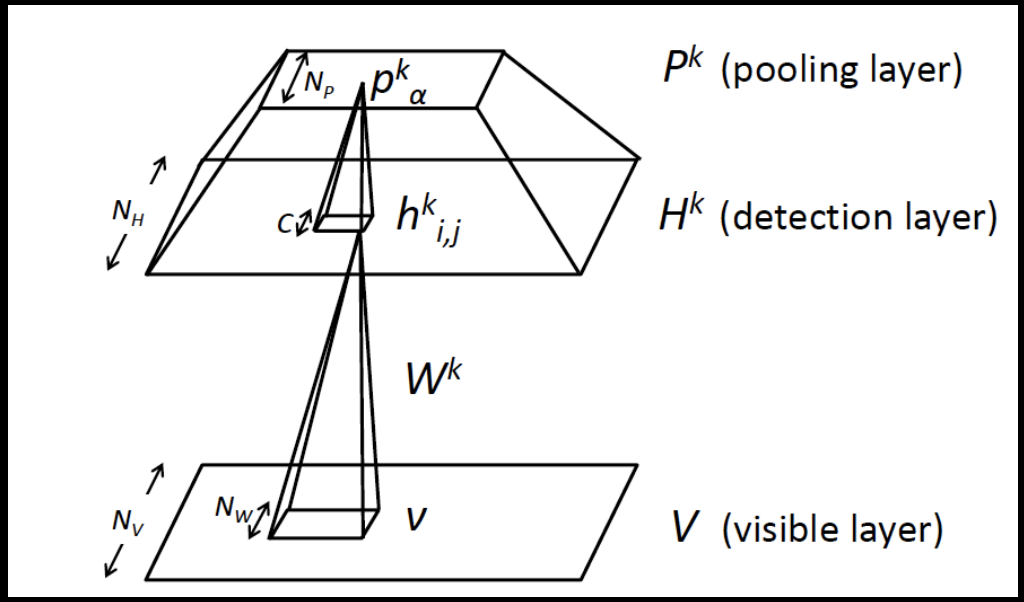
Deep Belief Nets and Deep Boltzmann Machines

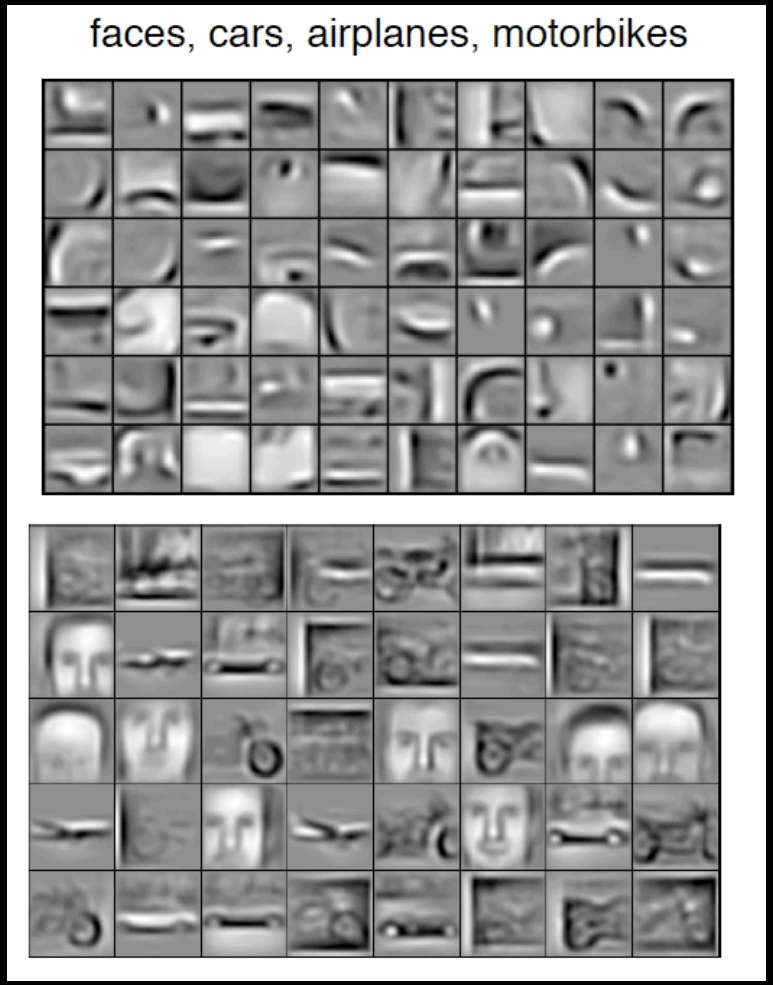
- DBN: Stack RBMs; top k of n layers = prior for last hidden of bottom n-k
- DBM: top levels *modify* the prior of last hidden of bottom n-k



Convolutional DBNs

(Lee et al, ICML'2009)

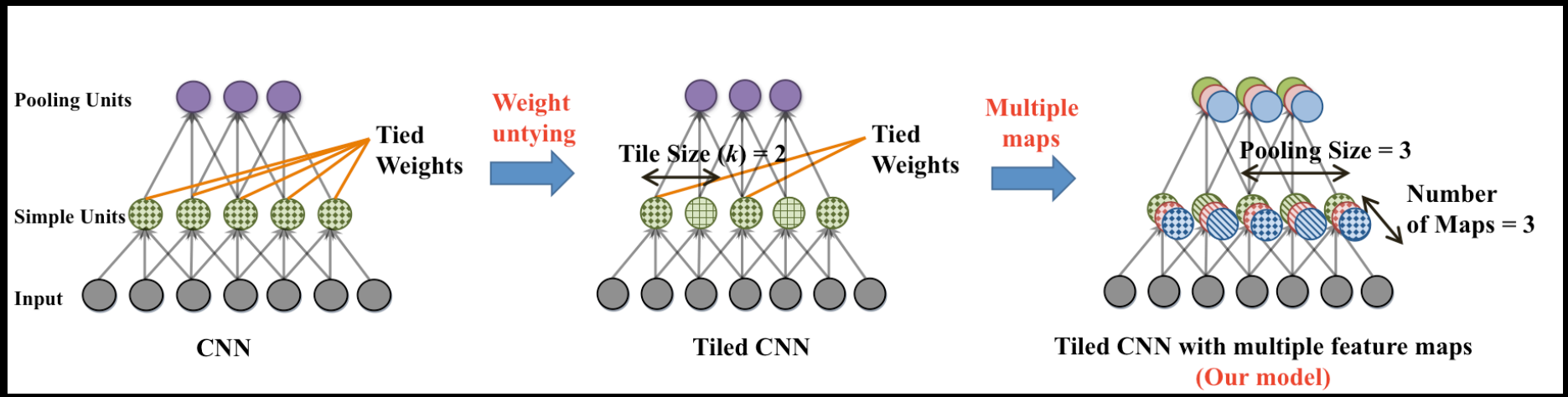




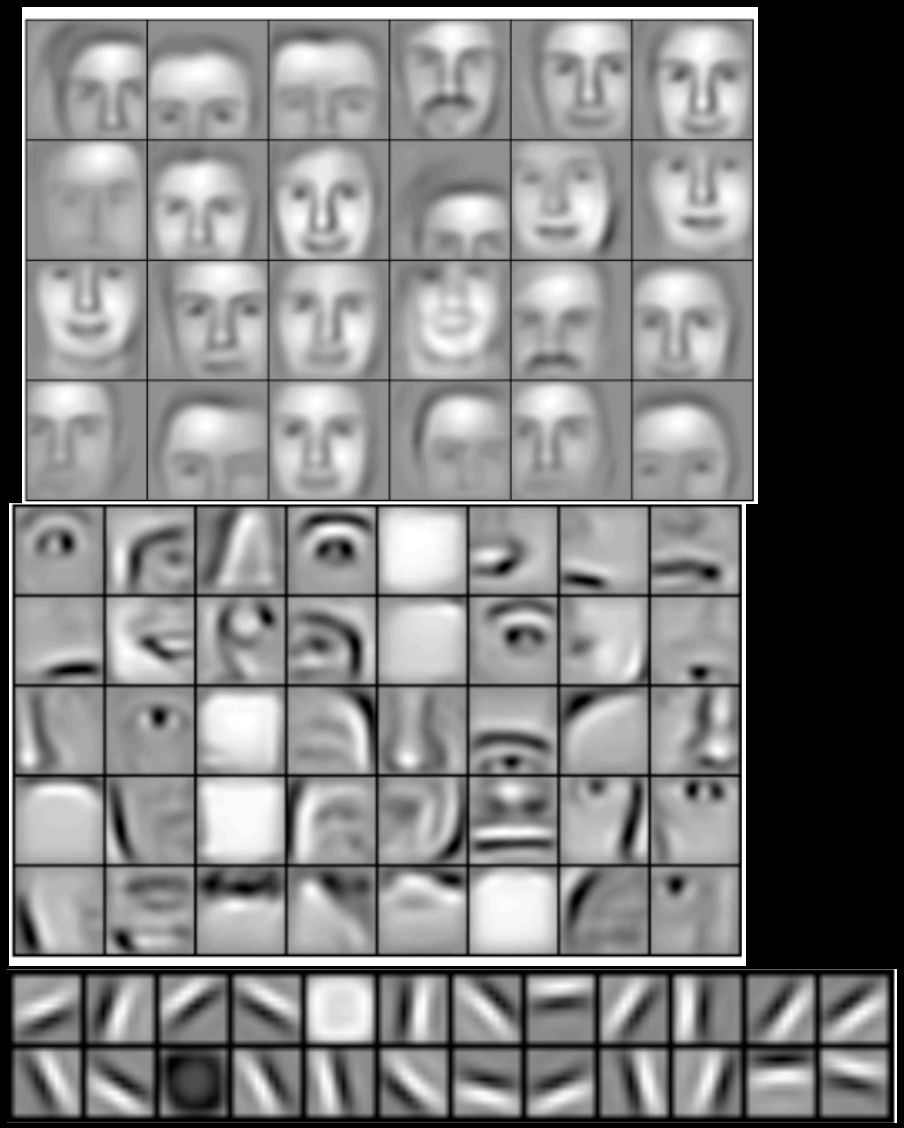
Tiled Convolutional Networks

Quoc et al NIPS 2010

Like convolutional but without the sharing, allows to increase capacity without increasing computation much (but increases memory)



Parts Are Composed to Form Objects



Layer 3: objects

Layer 2: parts

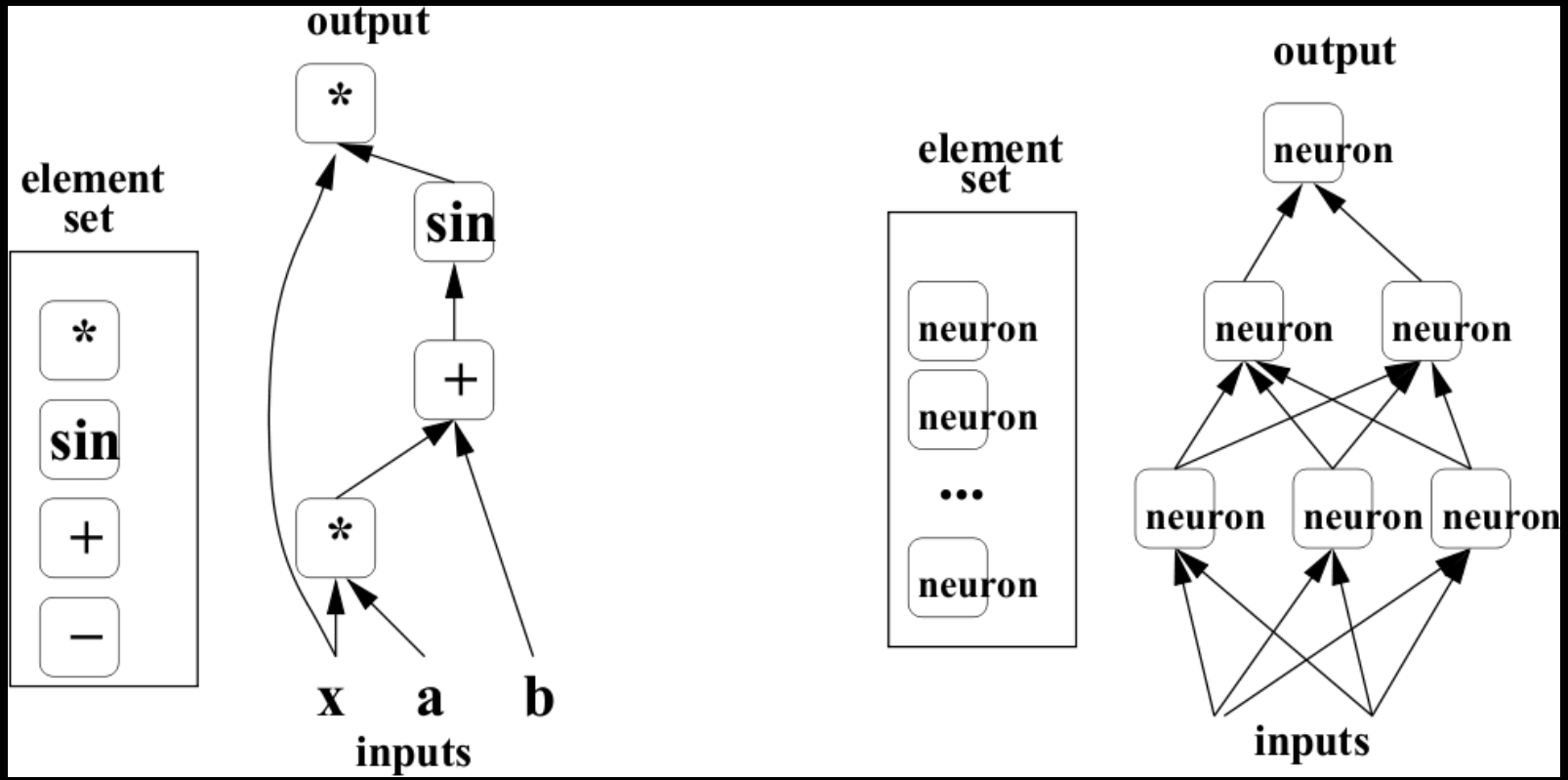
Layer 1: edges

Lee et al. ICML'2009

Representational Power of Deep Architectures

Shallow versus Deep Sum-Product Networks, Bengio & Delalleau, Learning Workshop 2011. Delalleau & Bengio paper submitted to NIPS 2011.

Architecture Depth



Depth = 4

Depth = 3

Deep Architectures are More Expressive

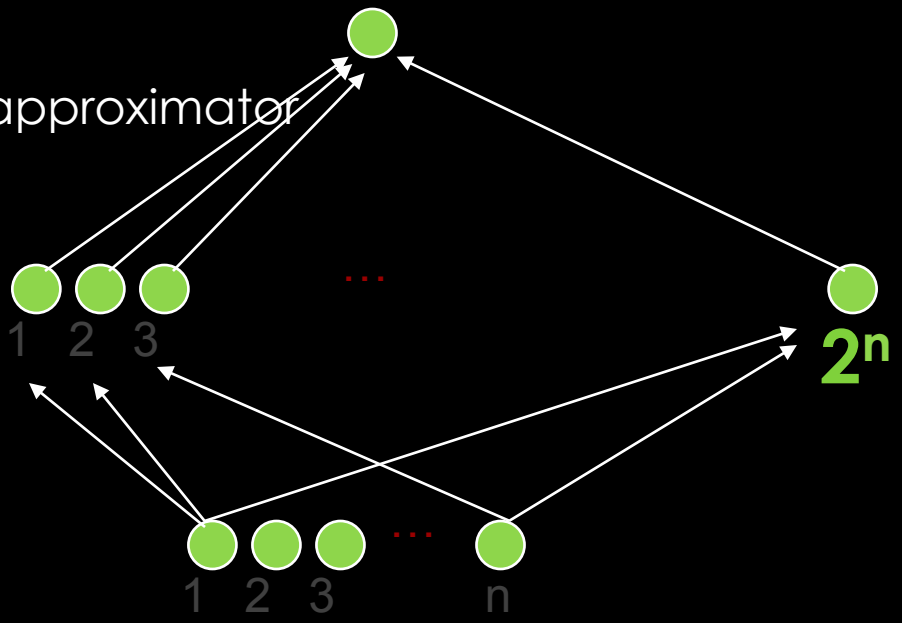

Theoretical arguments:

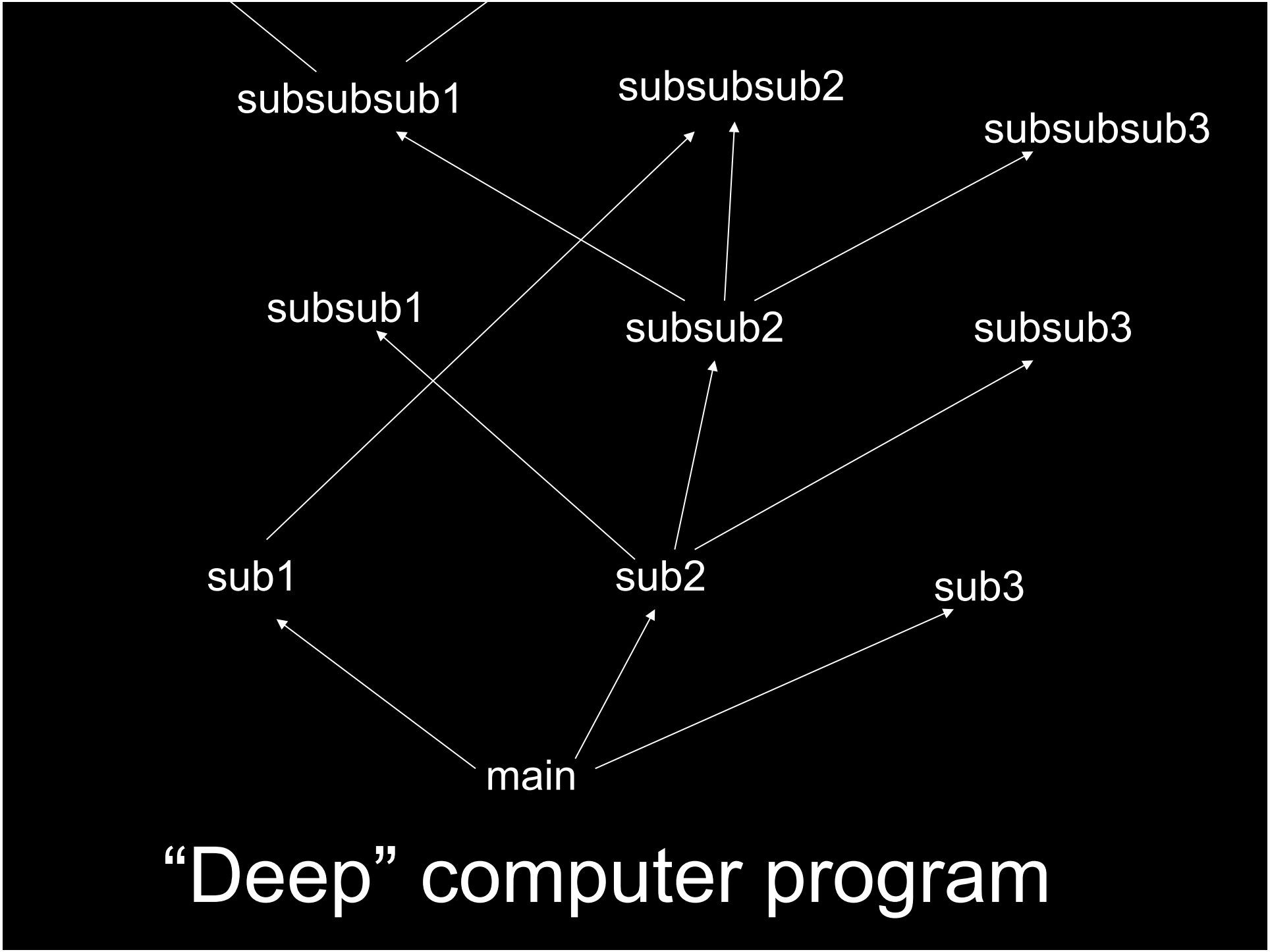
2 layers of { Logic gates / Formal neurons / RBF units } = universal approximator

RBMs & auto-encoders = universal approximator

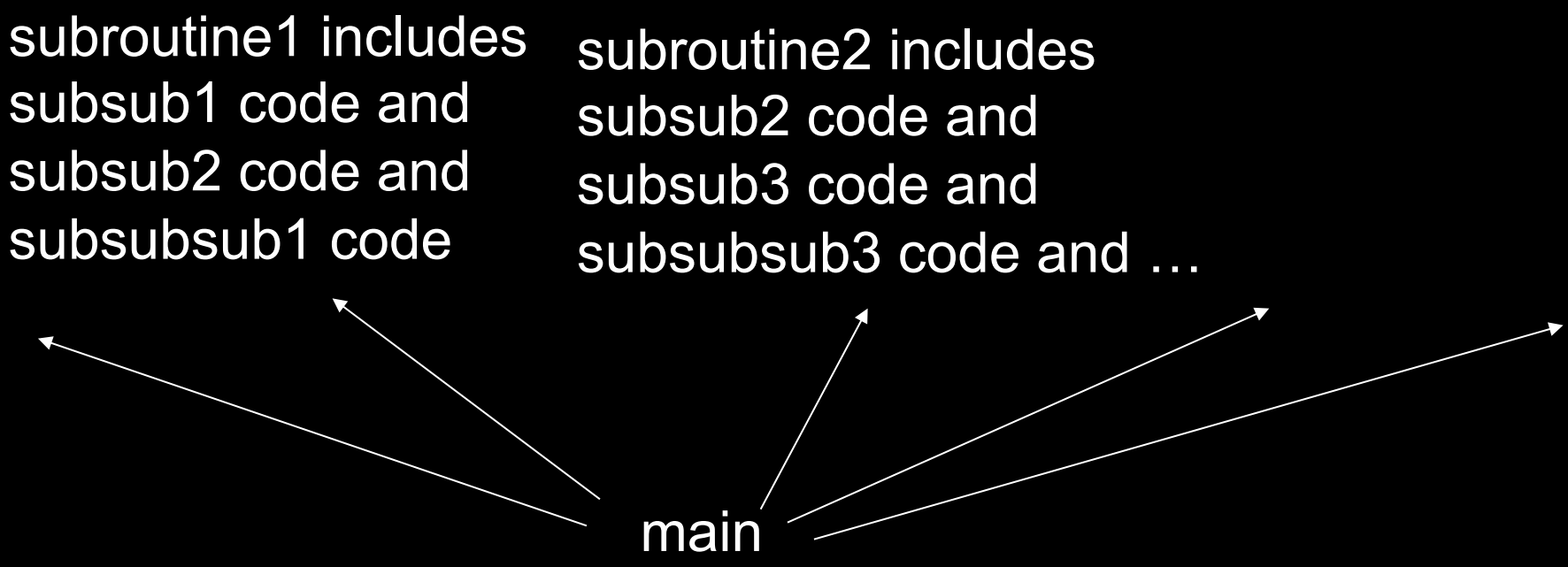
Theorems on advantage of depth:
(Hastad et al 86 & 91, Bengio et al 2007, Bengio & Delalleau 2011, Braverman 2011)

Functions compactly represented with k layers may require exponential size with 2 layers





“Deep” computer program



“Shallow” computer program

“Deep” circuit

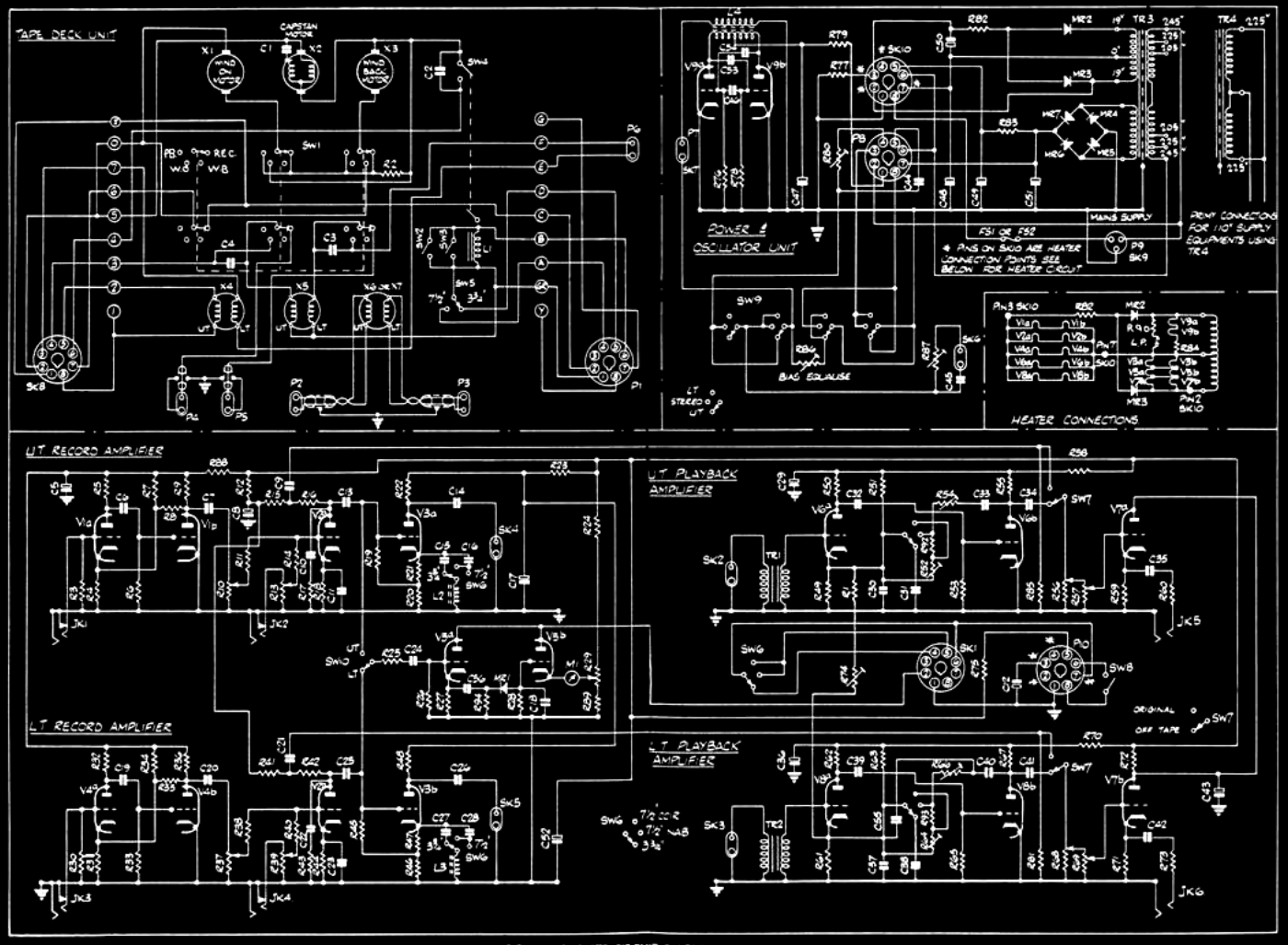
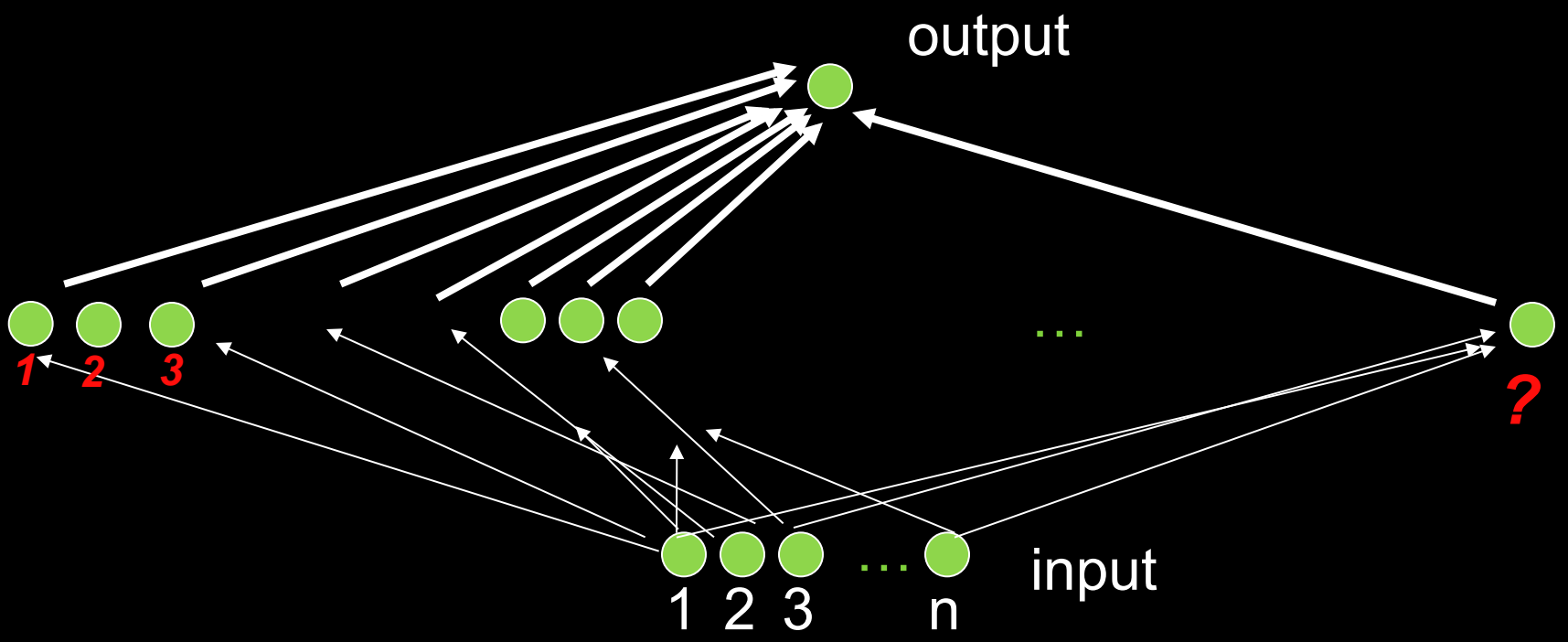


FIG. 16. COMPLETE CIRCUIT DIAGRAM, SERIES 420

“Shallow” circuit

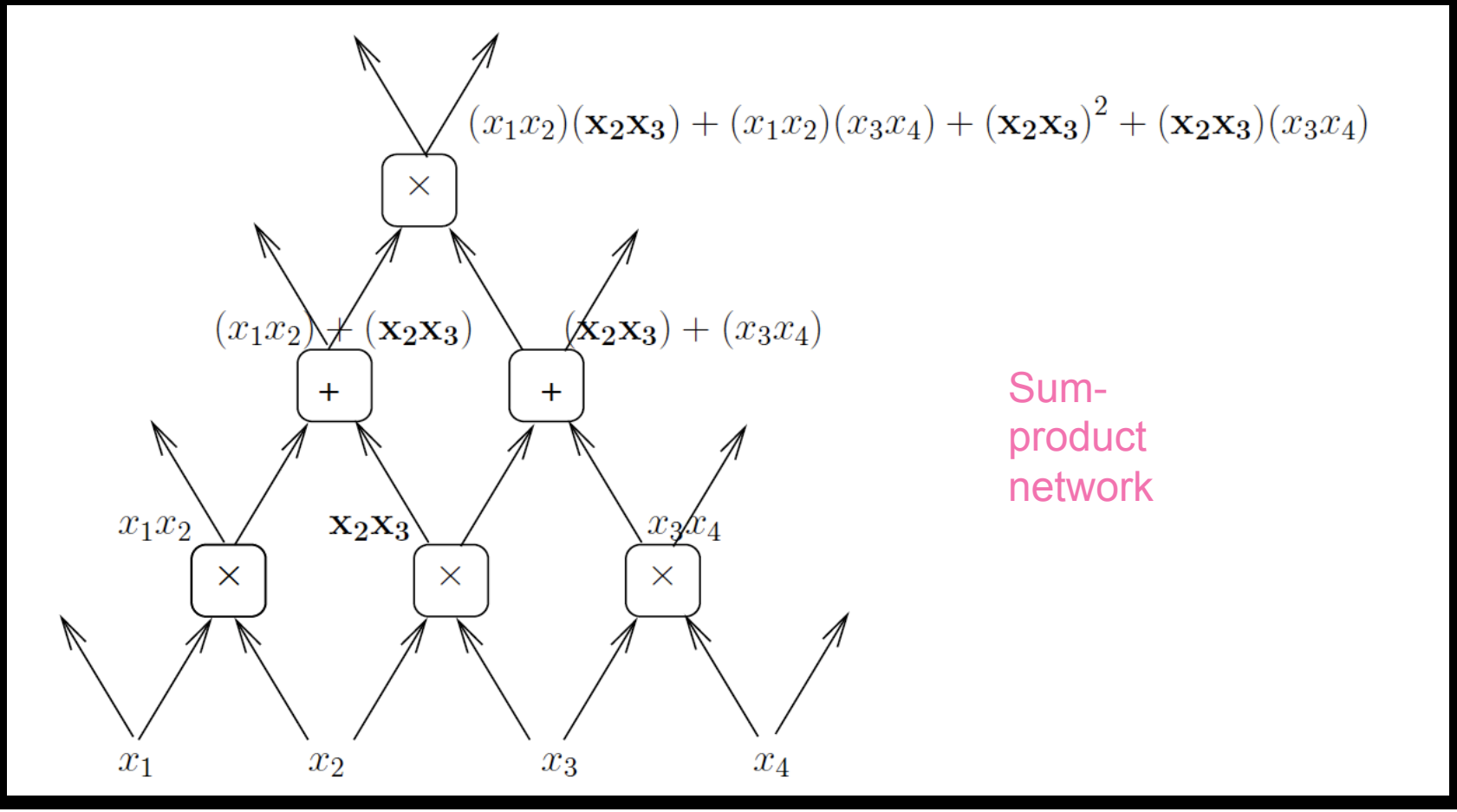


Falsely reassuring theorems: one can approximate any reasonable (smooth, boolean, etc.) function with a 2-layer architecture

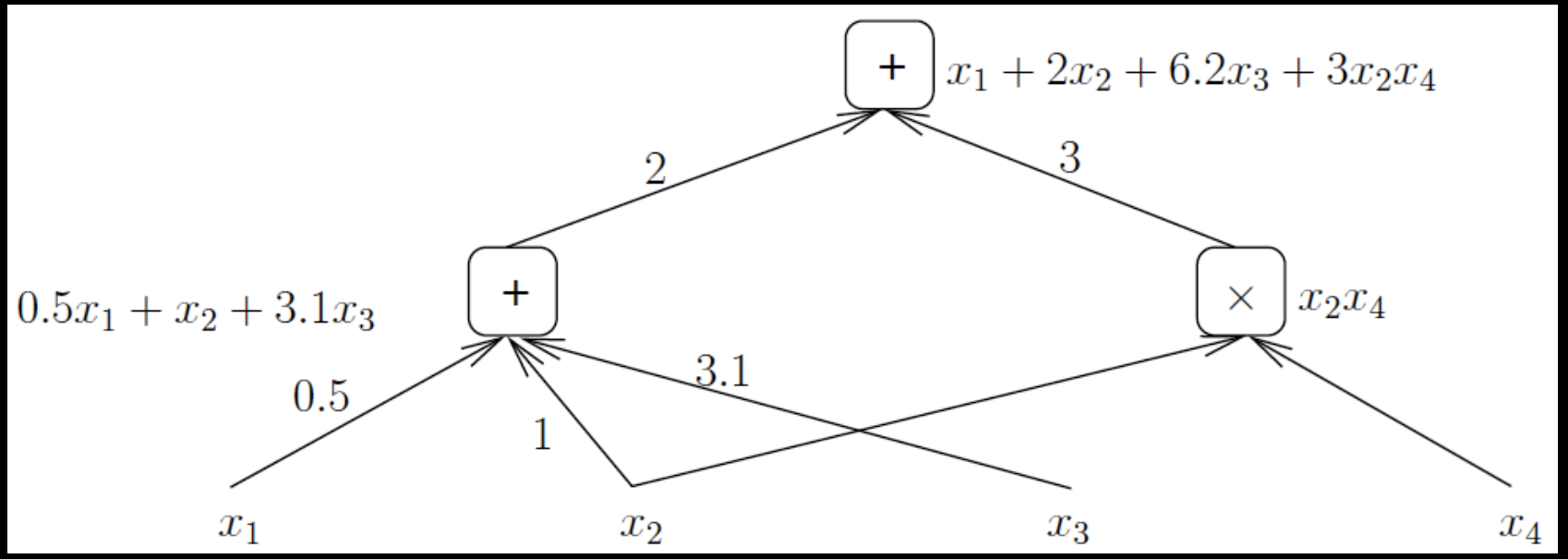
Sharing Components in a Deep Architecture

Polynomial expressed with shared components:

advantage of depth may grow exponentially

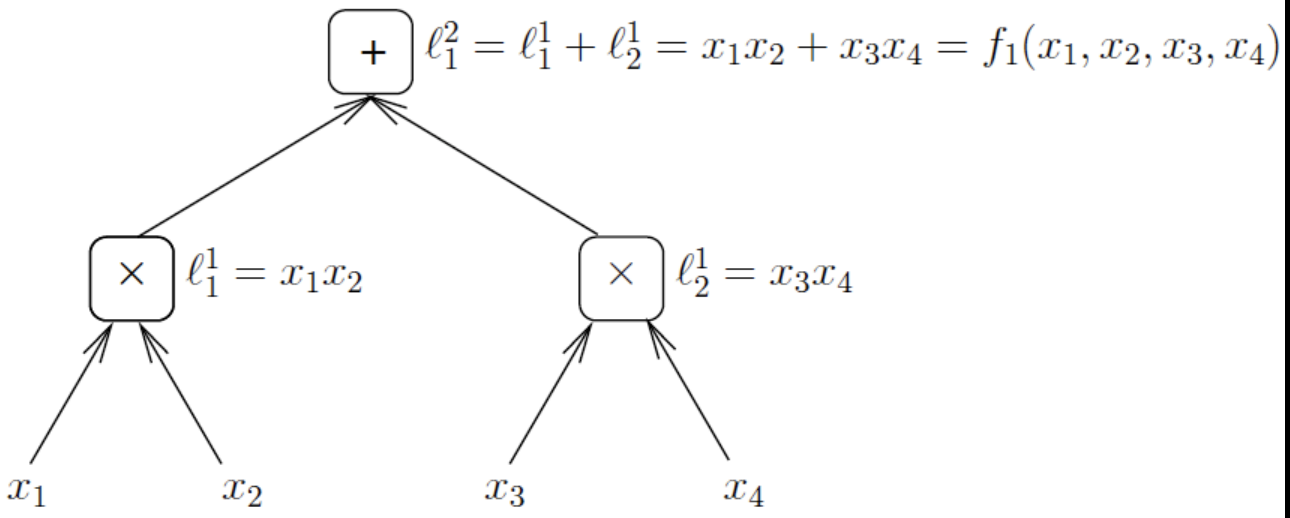


Sum-Product Networks



- Depth 2 suffices to represent any finite polynomial (sum of products)
- (Poon & Domingos 2010) use deep sum-product networks to efficiently parametrize partition functions

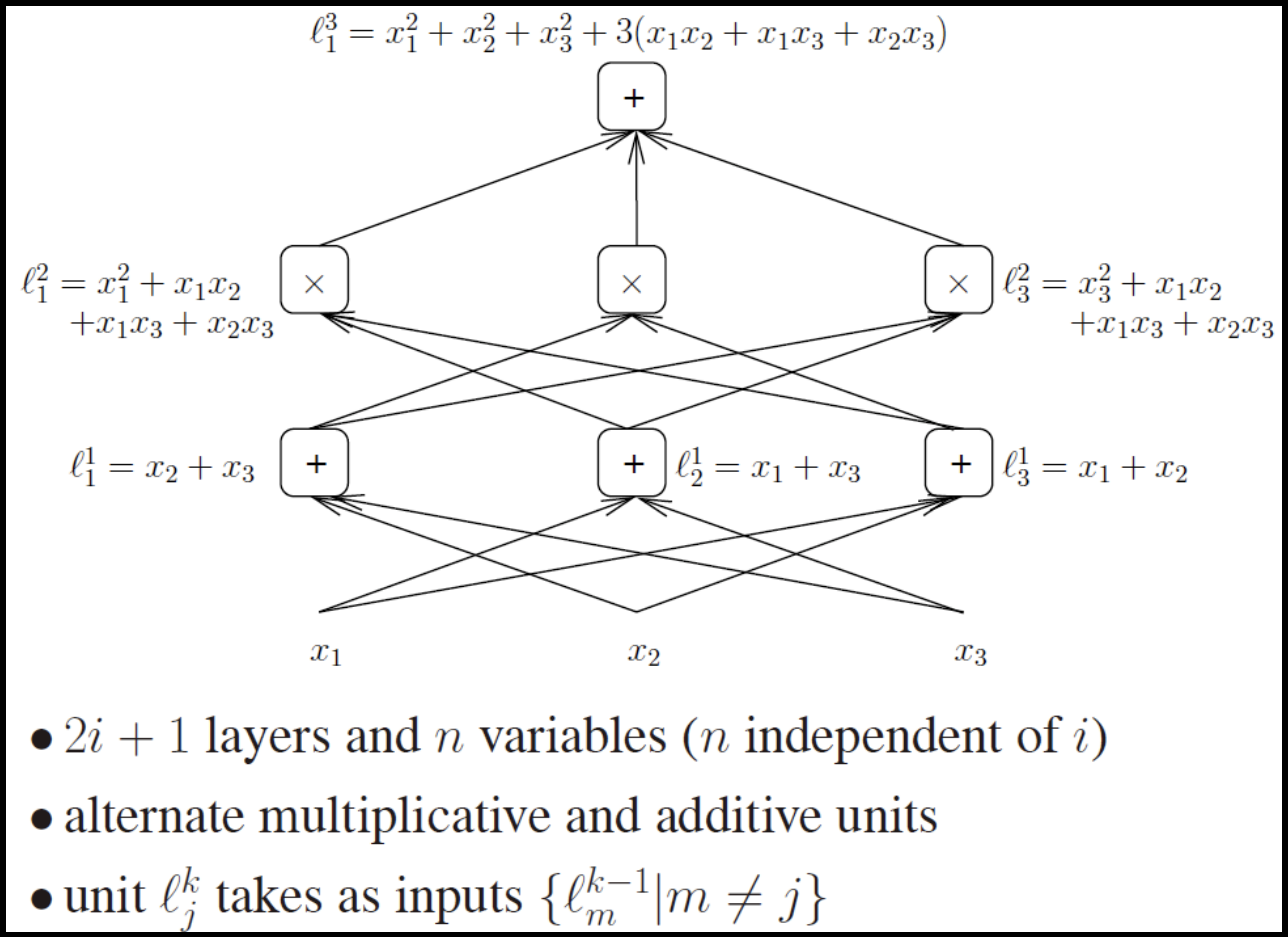
Polynomials that Need Depth



- $2i$ layers and $n = 4^i$ input variables
- alternate additive and multiplicative units
- unit ℓ_j^k takes as inputs ℓ_{2j-1}^{k-1} and ℓ_{2j}^{k-1}

- Need O(n) nodes with depth log(n) circuit
- Need $O(2^{\sqrt{n}})$ nodes with depth-2 circuit

More Polynomials that Need Depth



- $2i + 1$ layers and n variables (n independent of i)
- alternate multiplicative and additive units
- unit ℓ_j^k takes as inputs $\{\ell_m^{k-1} | m \neq j\}$

- Need O(dn) nodes with depth d circuit
- Need O(n^d) nodes with depth-2 circuit

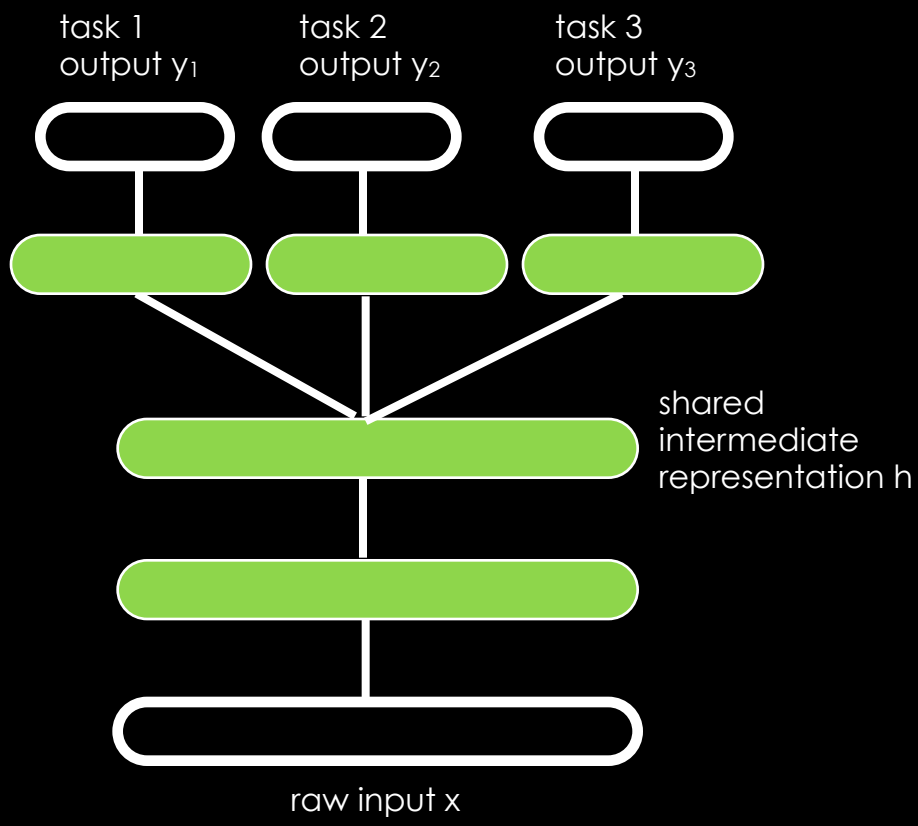
More Deep Theory

Poly-logarithmic Independence Fools Bounded-Depth Boolean Circuits, Braverman, CACM 54(4), April 2011.

If all marginals of the input distribution involving at most k variables are uniform, higher depth makes it exponentially easier to distinguish the joint from the uniform.

Deep Architectures and Sharing Statistical Strength, Multi-Task Learning

- Generalizing better to new tasks is crucial to approach AI
- Deep architectures learn good intermediate representations that can be shared across tasks
- Good representations make sense for many tasks



Parts Are Re-Used to Form Different Objects

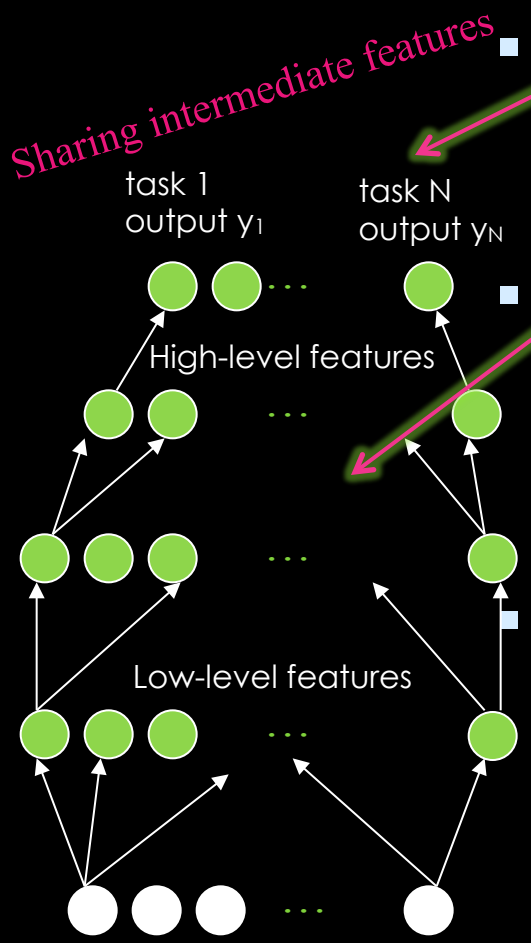
Layer 3: objects

Layer 2: parts

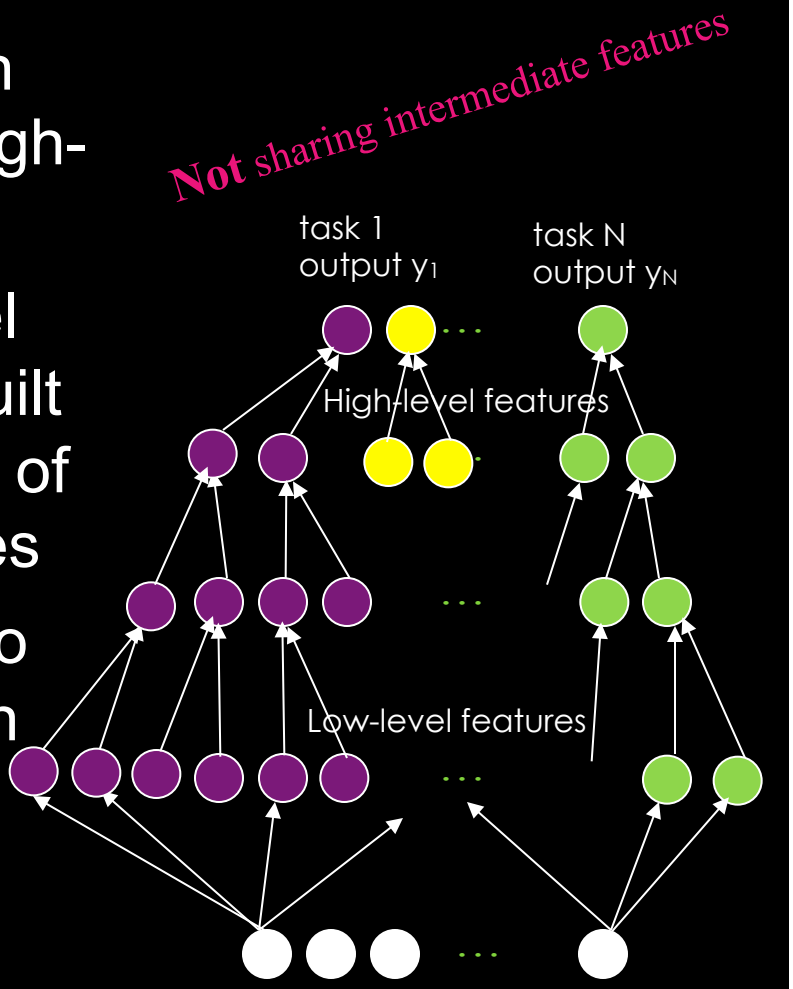
Layer 1: edges

Lee et al. ICML'2009

Feature and Sub-Feature Sharing

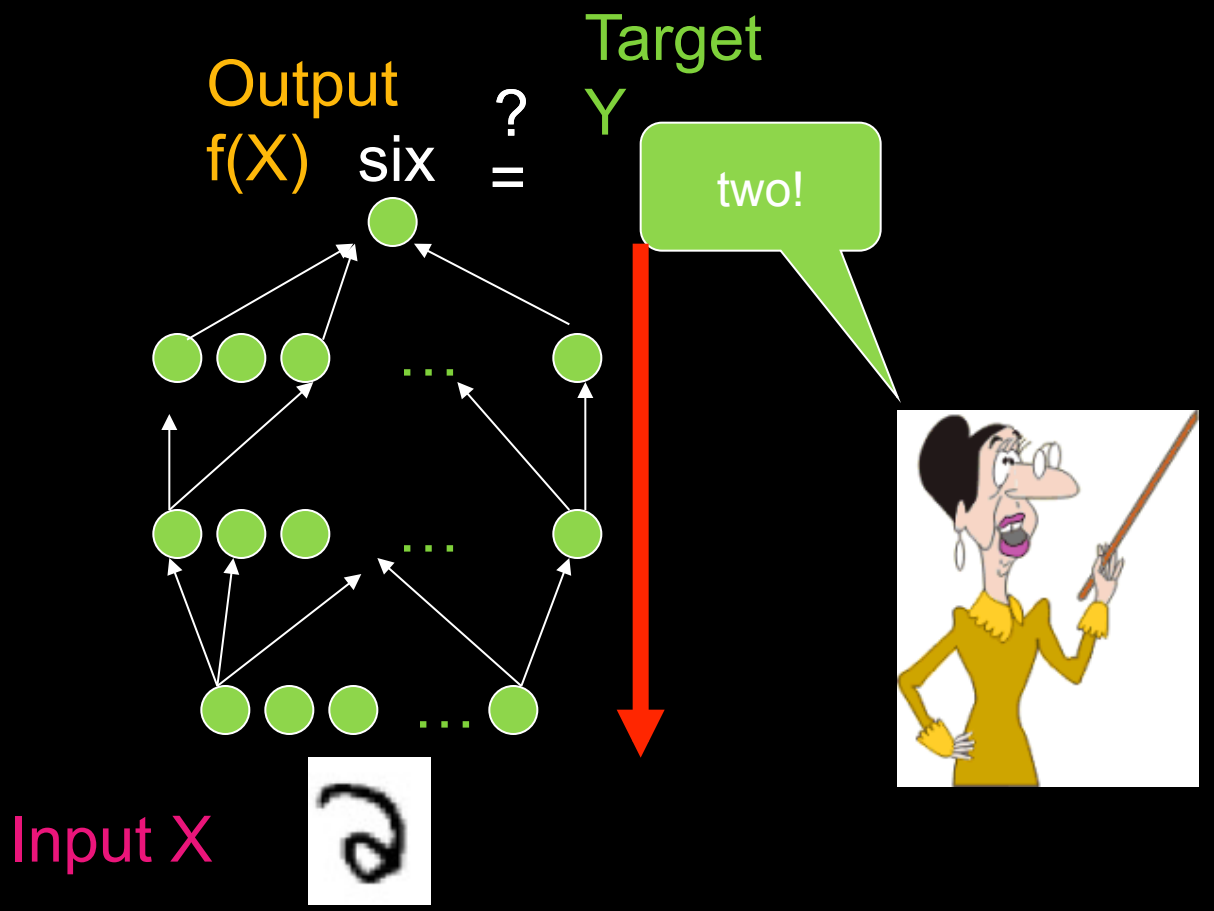


- Different tasks can share the same high-level features
- Different high-level features can be built from the same set of lower-level features
- More levels = up to exponential gain in representational efficiency

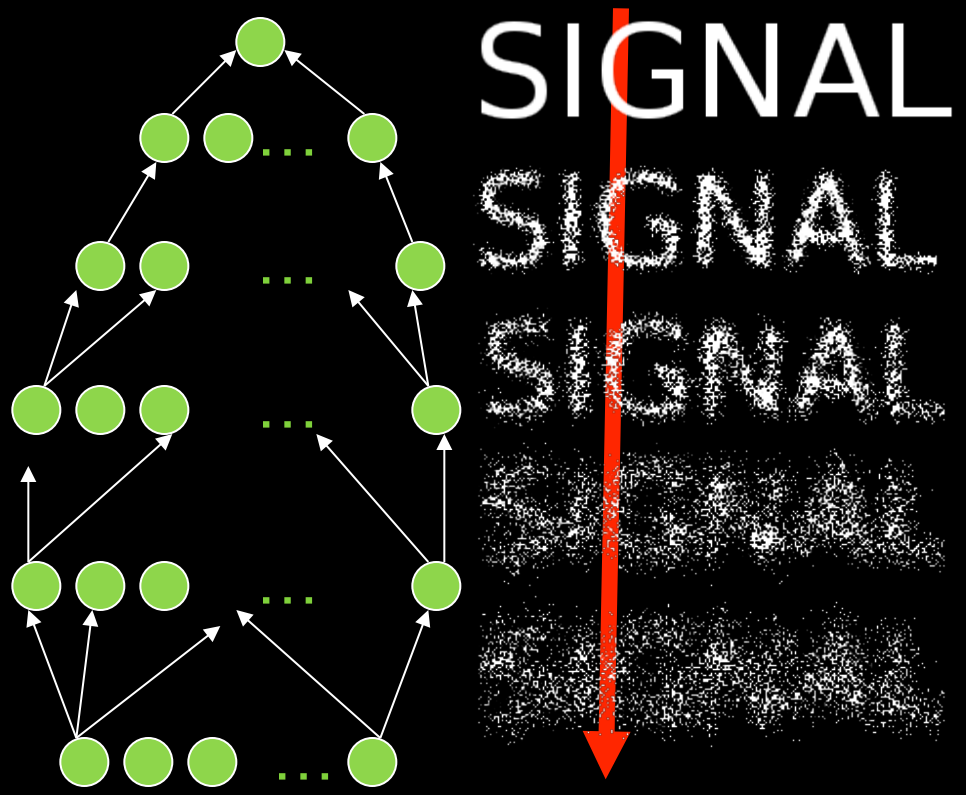


Challenge #3: training deep architectures

Gradient descent



Problem on deep architectures



Before 2006

Failure of deep architectures

2006
Breakthrough!

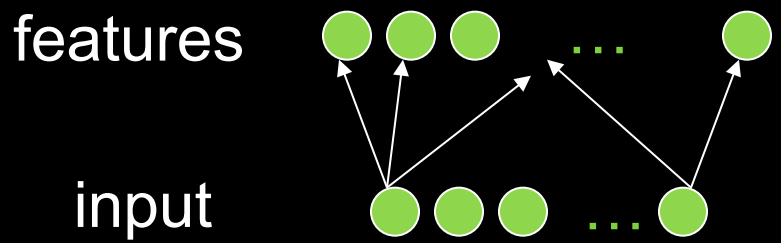
2006: The Deep Breakthrough

- Before 2006, training deep architectures was unsuccessful, except for convolutional neural nets

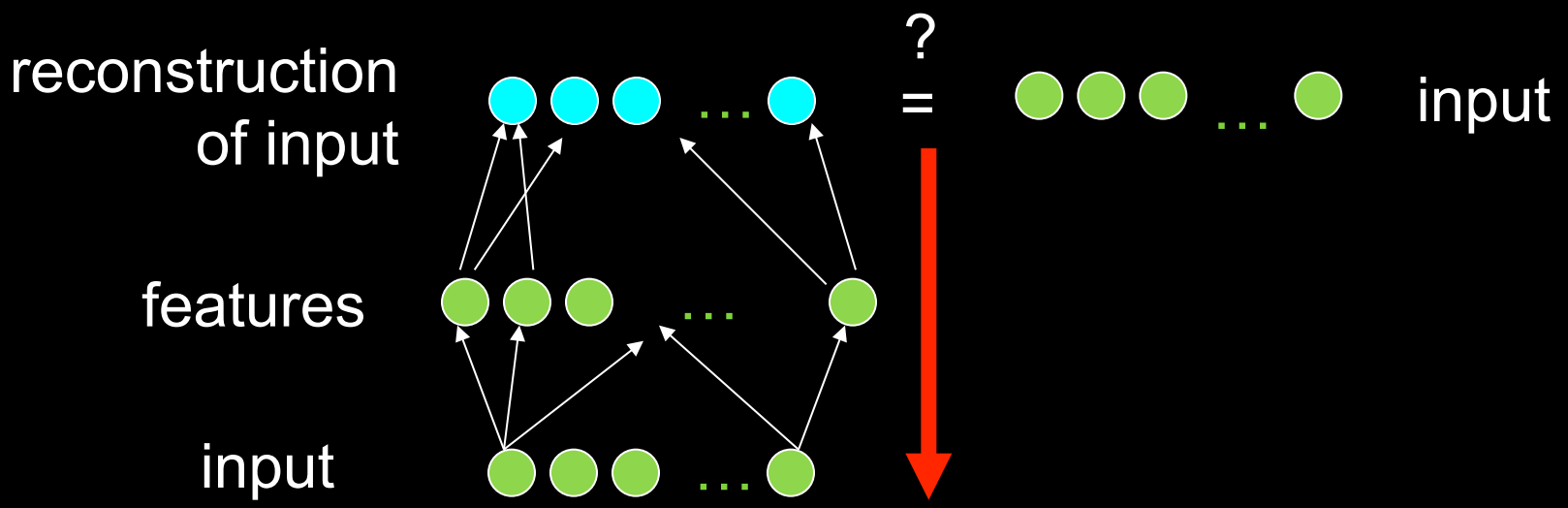
- Hinton, Osindero & Teh « A Fast Learning Algorithm for Deep Belief Nets », *Neural Computation*, 2006
- Bengio, Lamblin, Popovici, Larochelle « Greedy Layer-Wise Training of Deep Networks », *NIPS'2006*
- Ranzato, Poultney, Chopra, LeCun « Efficient Learning of Sparse Representations with an Energy-Based Model », *NIPS'2006*

Deep training

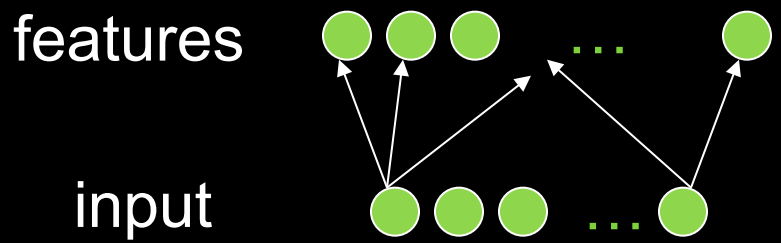
Layer-Wise Unsupervised Pre-training



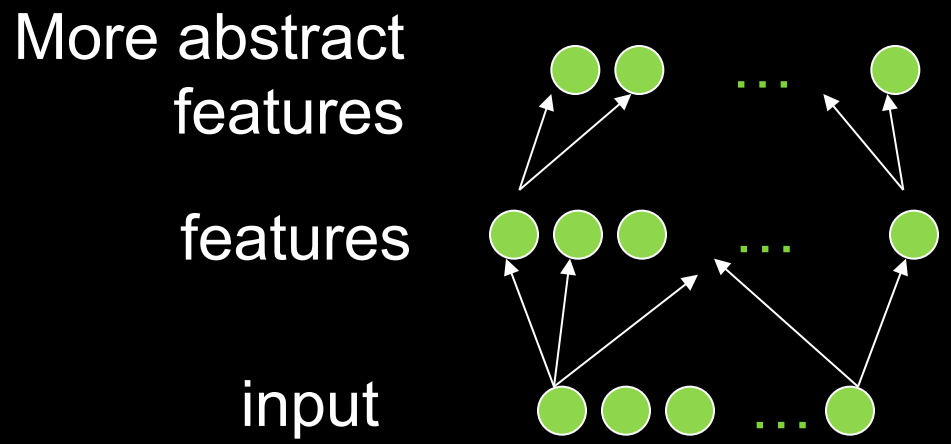
Layer-Wise Unsupervised Pre-training



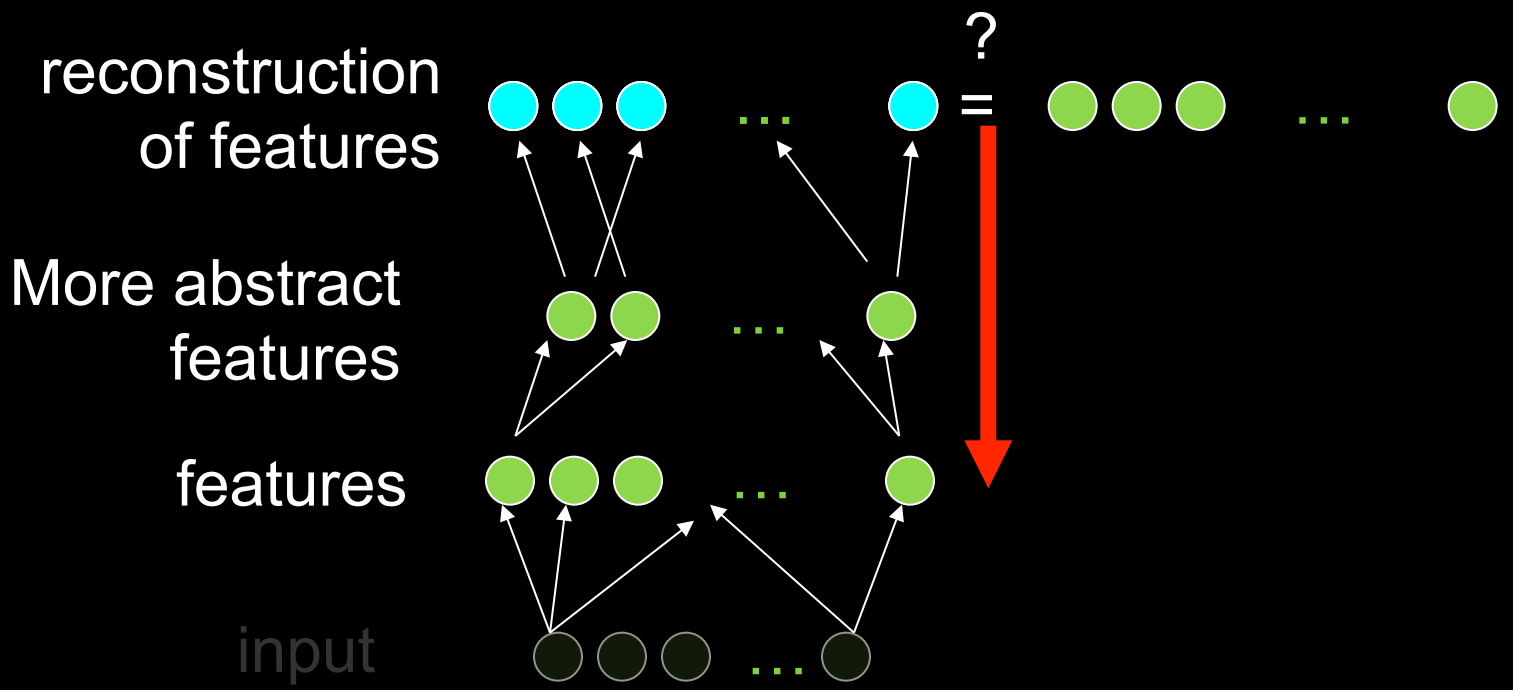
Layer-Wise Unsupervised Pre-training



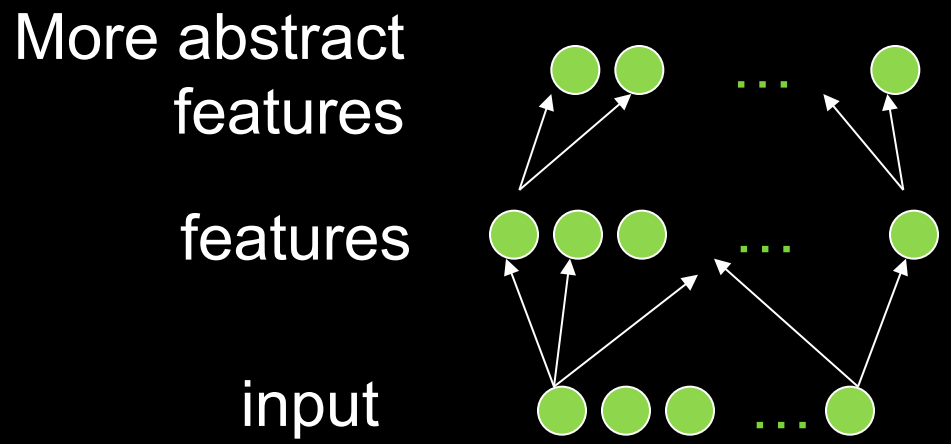
Layer-Wise Unsupervised Pre-training



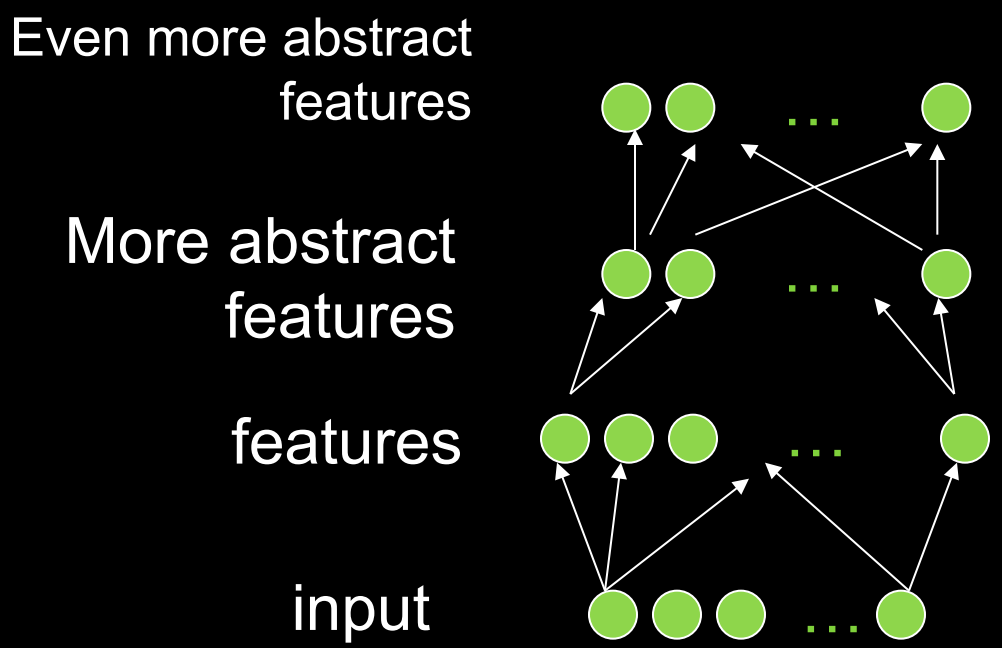
Layer-Wise Unsupervised Pre-training



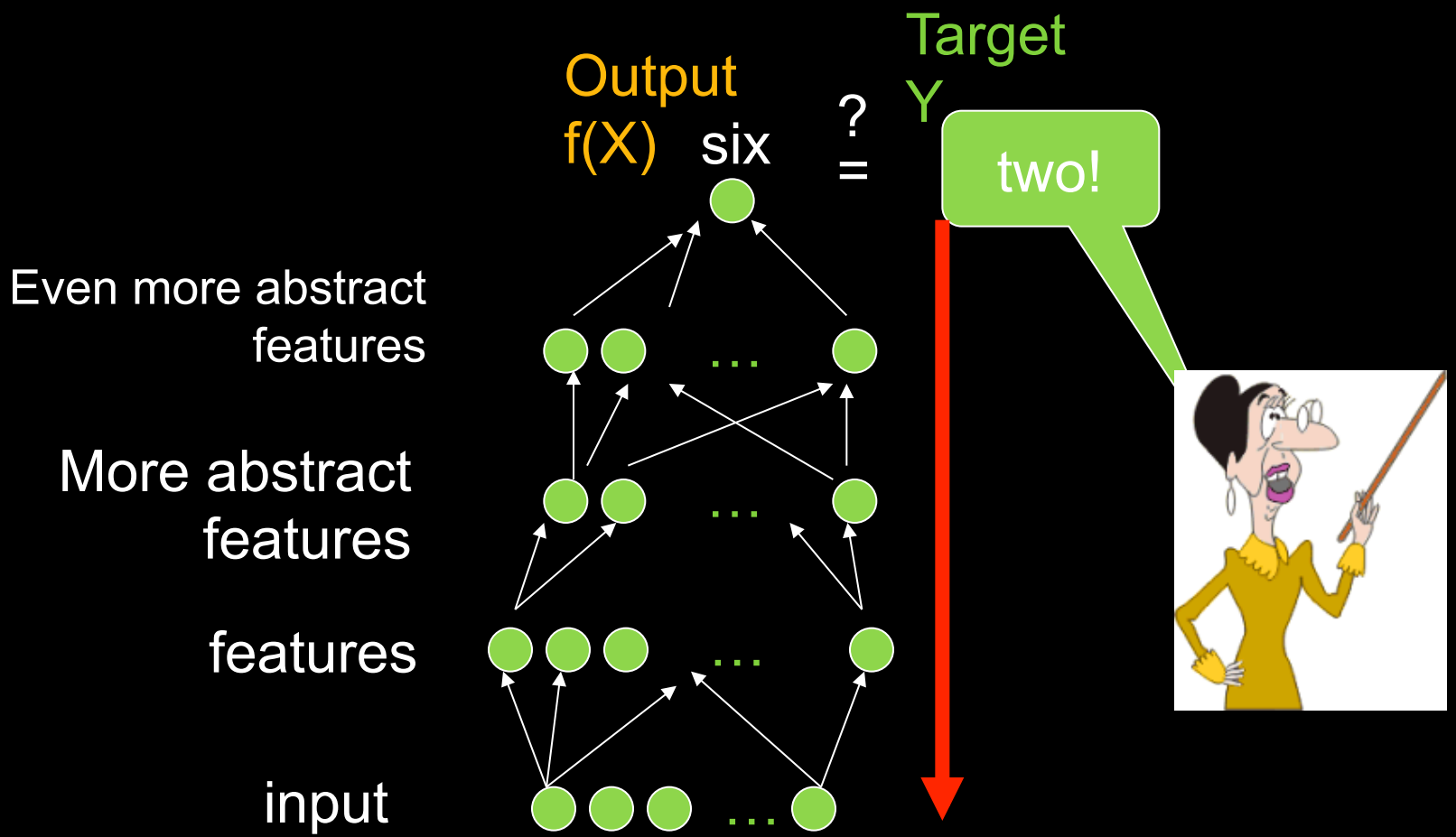
Layer-Wise Unsupervised Pre-training



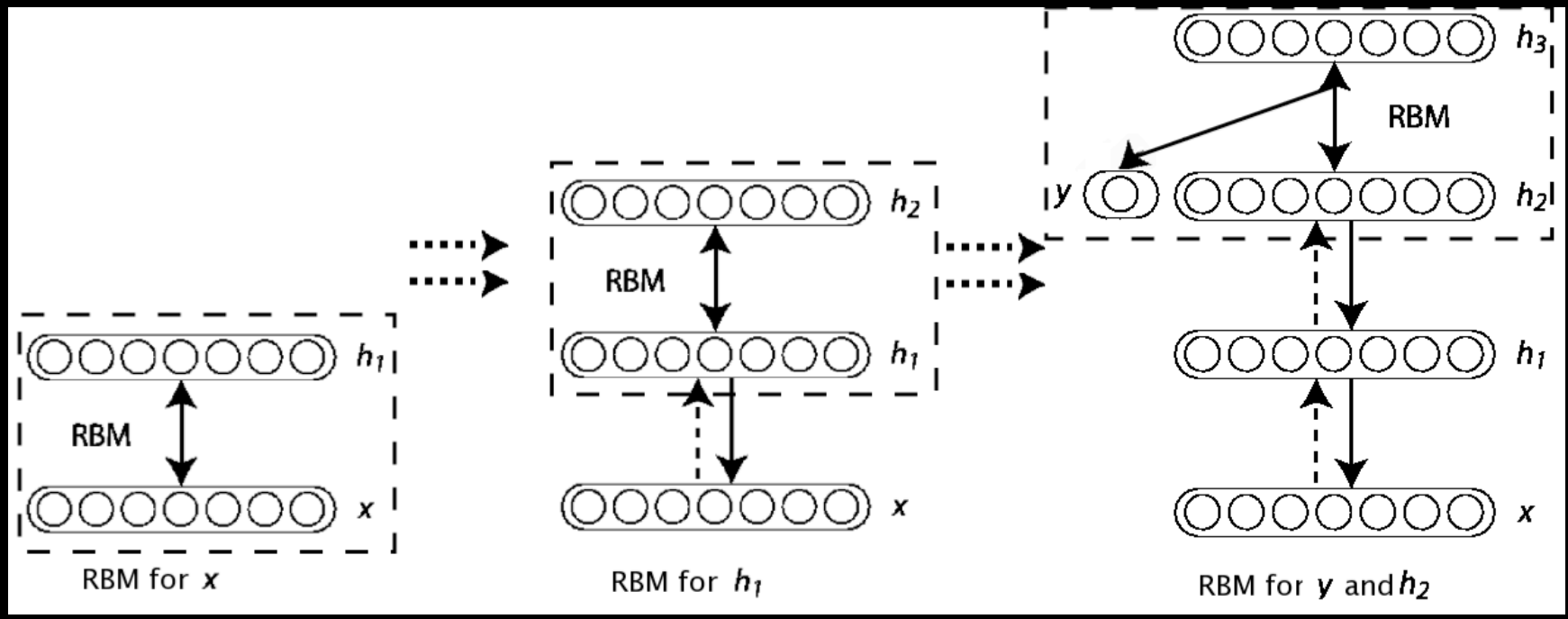
Layer-Wise Unsupervised Pre-training



Supervised Fine-Tuning

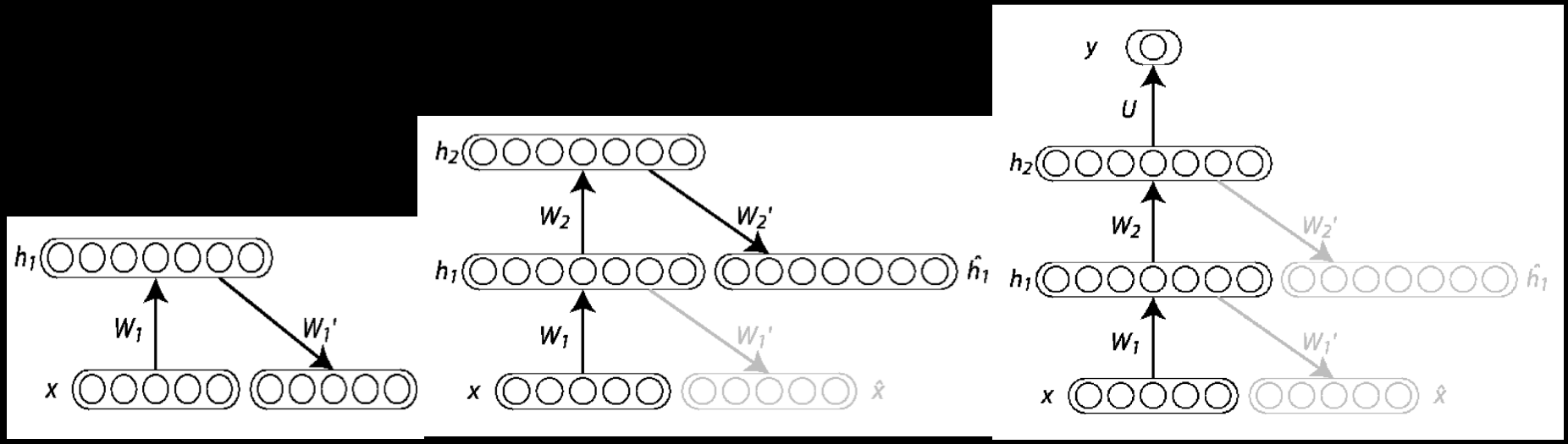


Greedy Layer-Wise Pre-Training

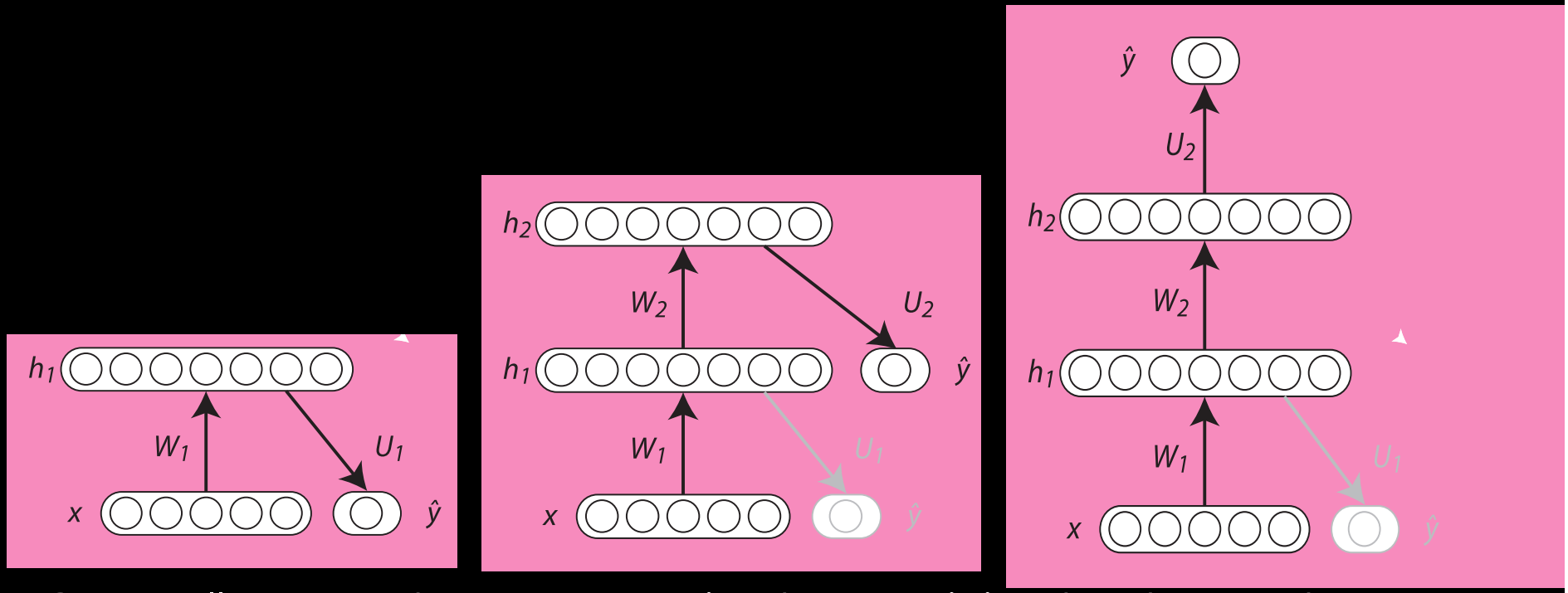


Stacking Restricted Boltzmann Machines (RBM) → Deep Belief Network (DBN)

Stacking Auto-Encoders



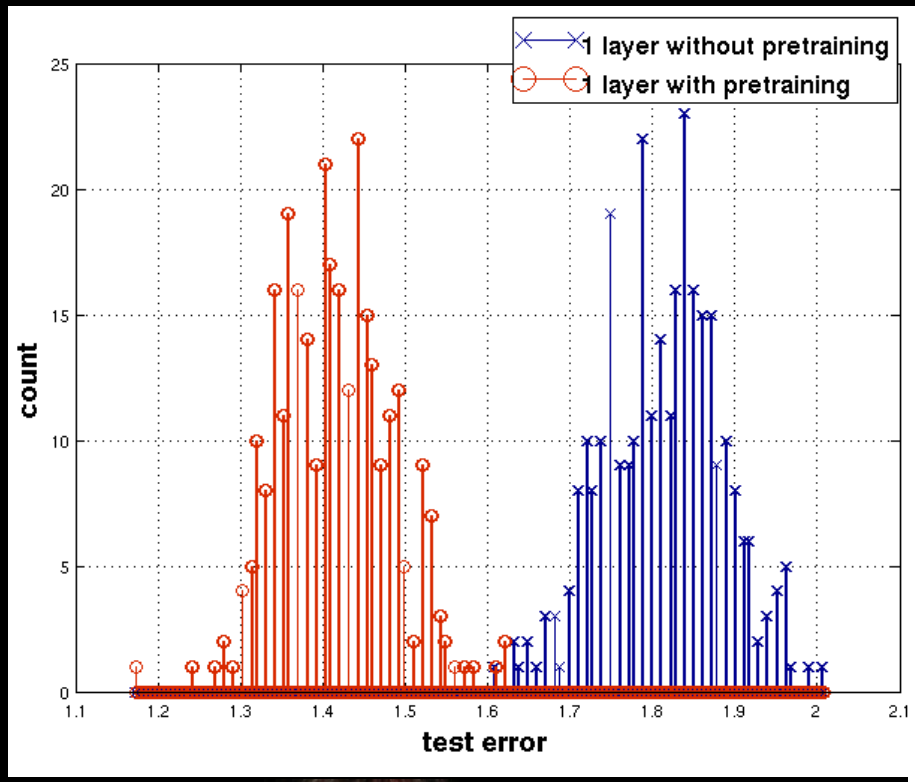
Greedy Layerwise Supervised Training

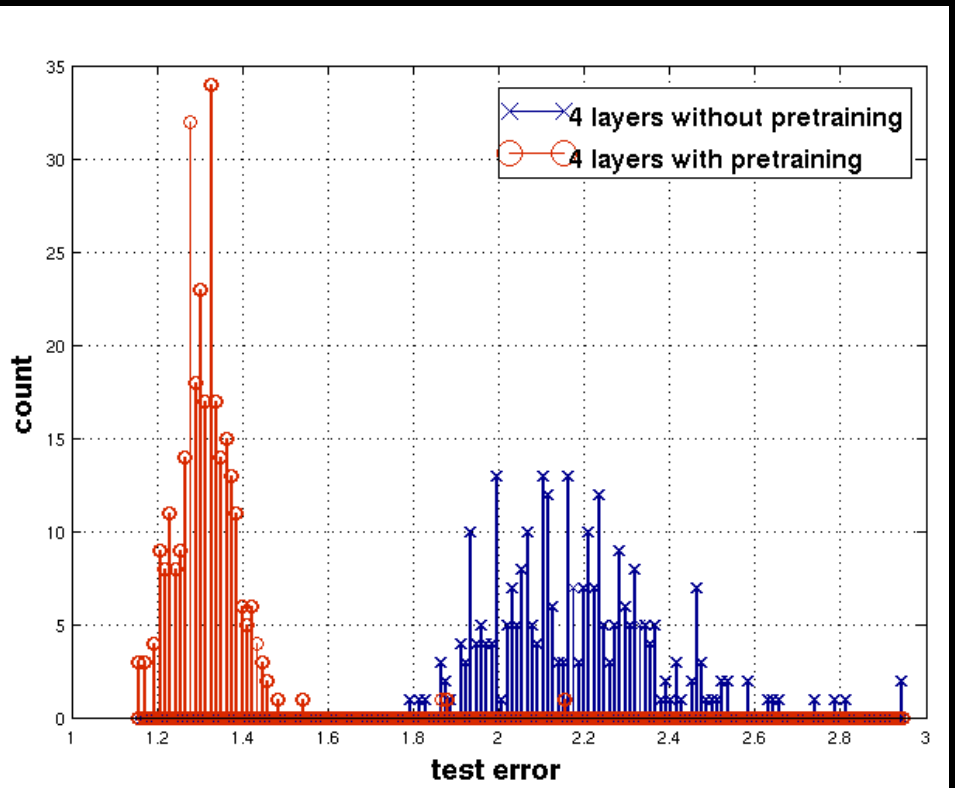


Generally worse than unsupervised pre-training but better than ordinary training of a deep neural network (Bengio et al. 2007).

Effect of Unsupervised Pre-training

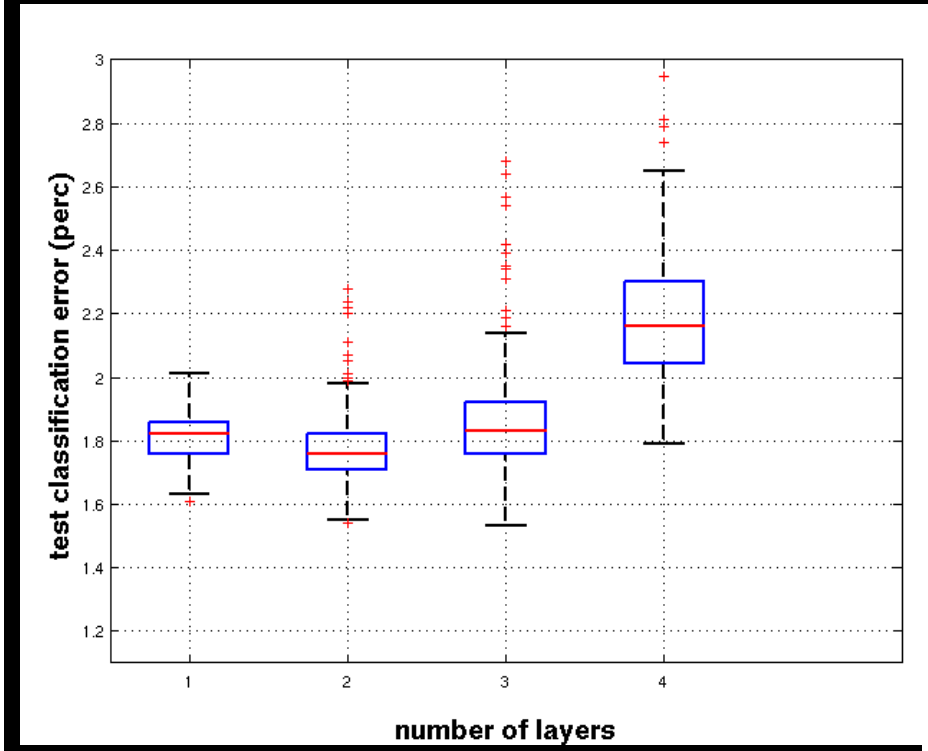
AISTATS'2009+JMLR 2010, with Erhan, Courville, Manzagol, Vincent, S. Bengio



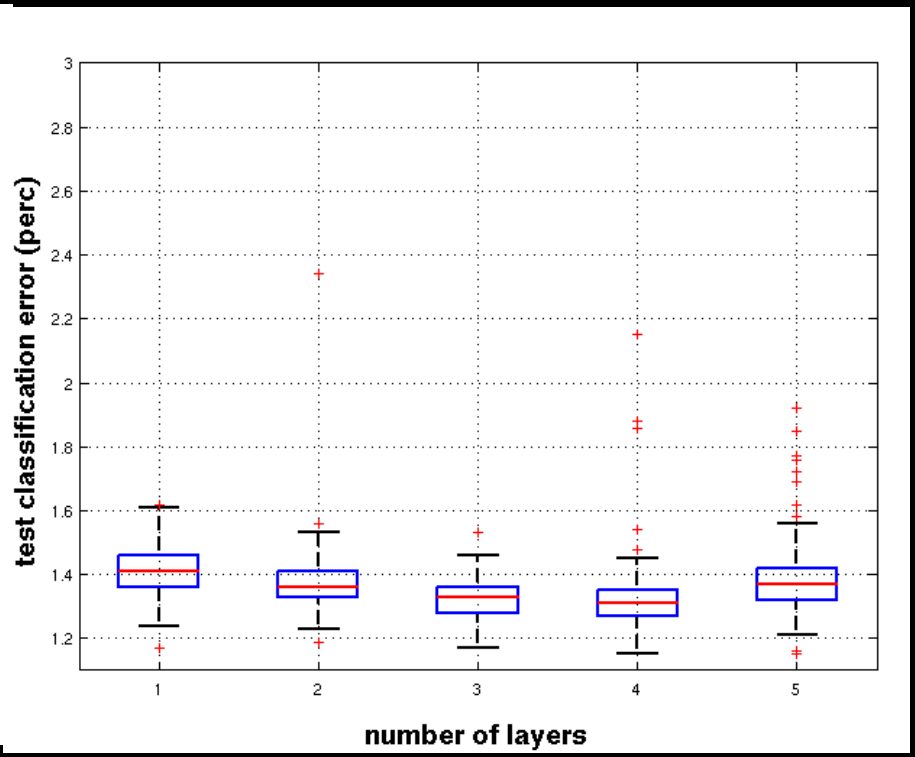


Effect of Depth

w/o pre-training



with pre-training



Level-Local Learning is Important

- Initializing each layer of an unsupervised deep Boltzmann machine helps a lot
- Initializing each layer of a supervised neural network as an RBM, auto-encoder, denoising auto-encoder, etc helps a lot
- Helps most the layers further away from the target
- Not just an effect of unsupervised prior
- Jointly training all the levels of a deep architecture is difficult
- Initializing using **a level-local learning algorithm** is a useful trick

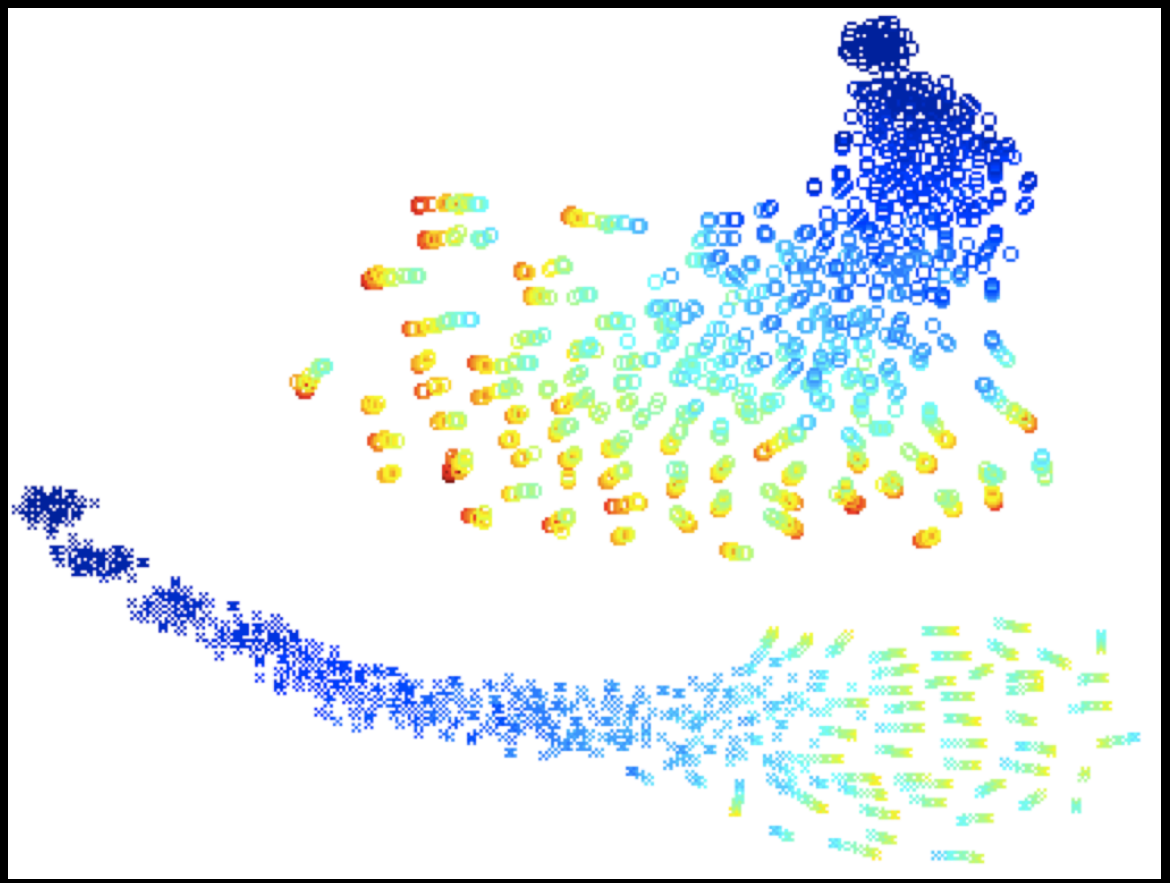
Why is ***Unsupervised*** Pre-Training Working So Well?

(with Erhan, Courville, Manzagol, Vincent, Bengio: JMLR, 2010)

- Regularization hypothesis:
  - Unsupervised component forces model close to P(x)
  - Representations good for P(x) are good for P(y|x)
- Optimization hypothesis:
  - Unsupervised initialization near better local minimum of supervised training error
  - Can reach lower local minimum otherwise not achievable by random initialization

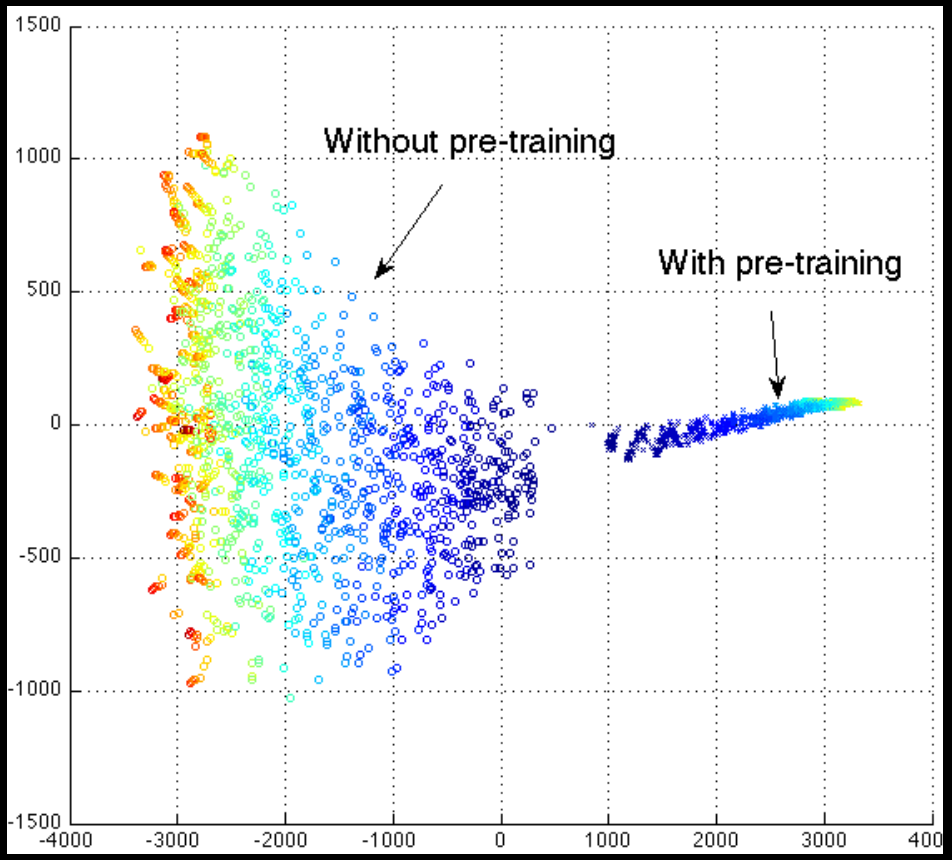
Learning Trajectories in Function Space
(Erhan et al, JMLR, 2010)

- Each point is a model in function space
- Color = epoch
- Top: trajectories w/o pre-training
- Each trajectory converges in different local min.
- No overlap of regions with and w/o pre-training



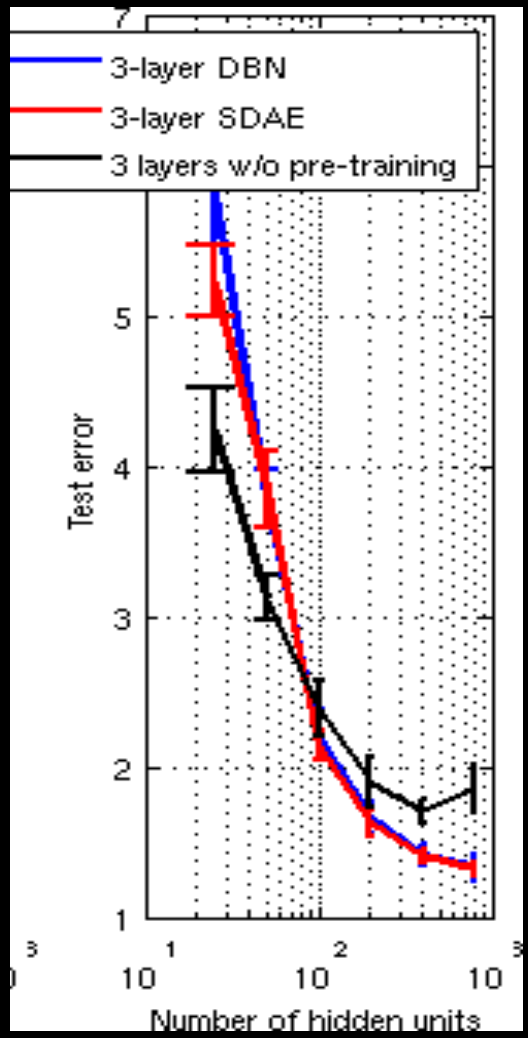
Visualization in Function Space

- Using ISOMAP instead of t-SNE, preserve distances
- Pre-training: small volume compared to without.



Unsupervised Learning as Regularizer

- Adding extra regularization (reducing # hidden units) hurts more the pre-trained models
- Pre-trained models have less variance wrt training sample
- Regularizer = infinite penalty outside of region compatible with unsupervised pre-training

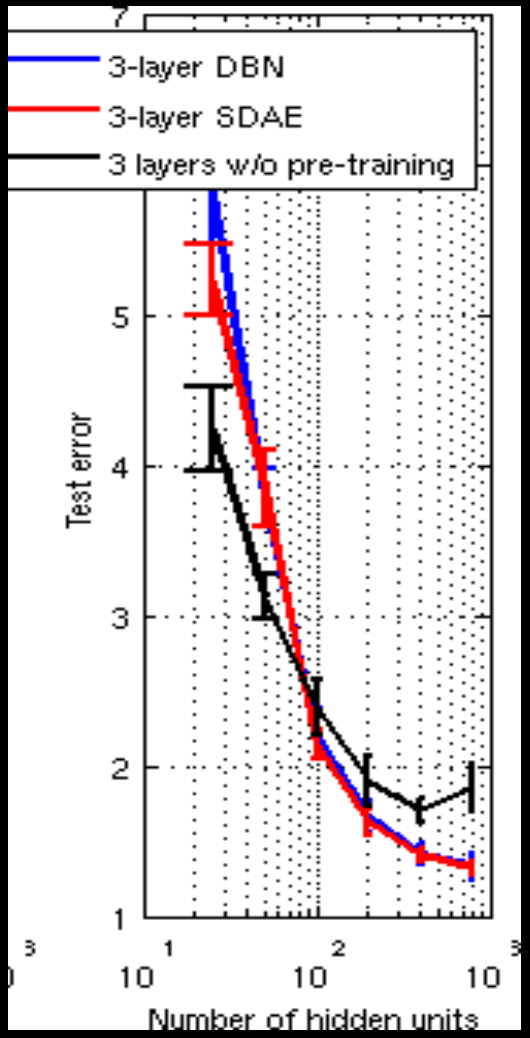


Unsupervised Disentangling of Factors of Variation

- **(Untested) Explanatory theory**:
  - Stacked RBMs & DAE disentangle factors of variation in *P(x)* (Goodfellow et al, NIPS'09)
  - Most salient factors are unrelated to *y*, but some factors are highly predictive of *y*

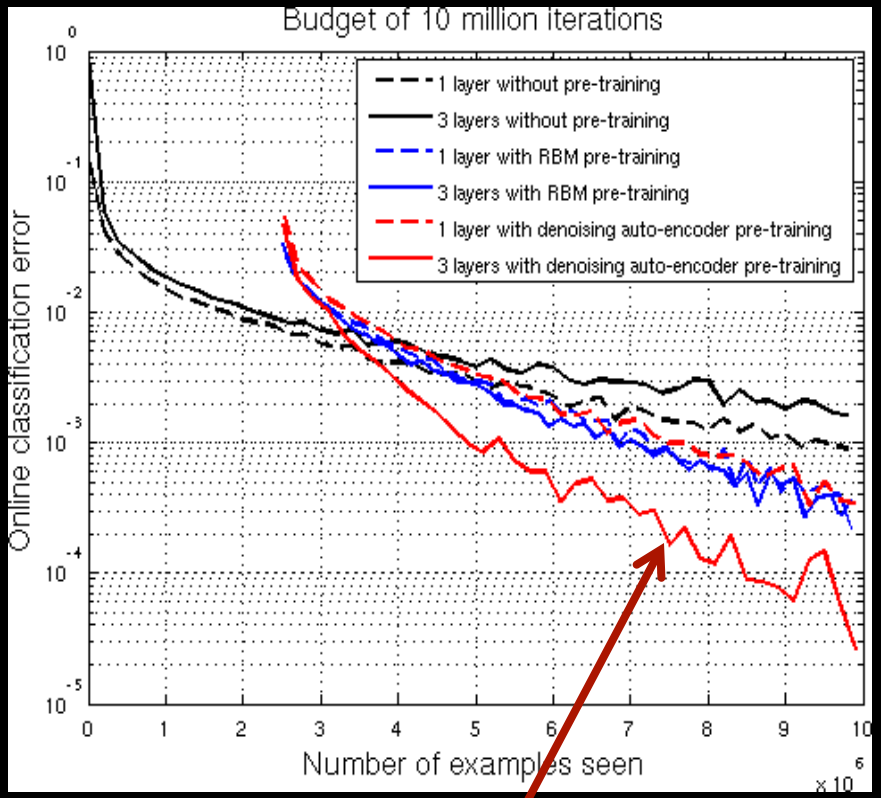
  → RBMs with too few units learn features worse at predicting *y* than randomly initialized networks

  → RBMs with many hidden units are much more predictive of *y*

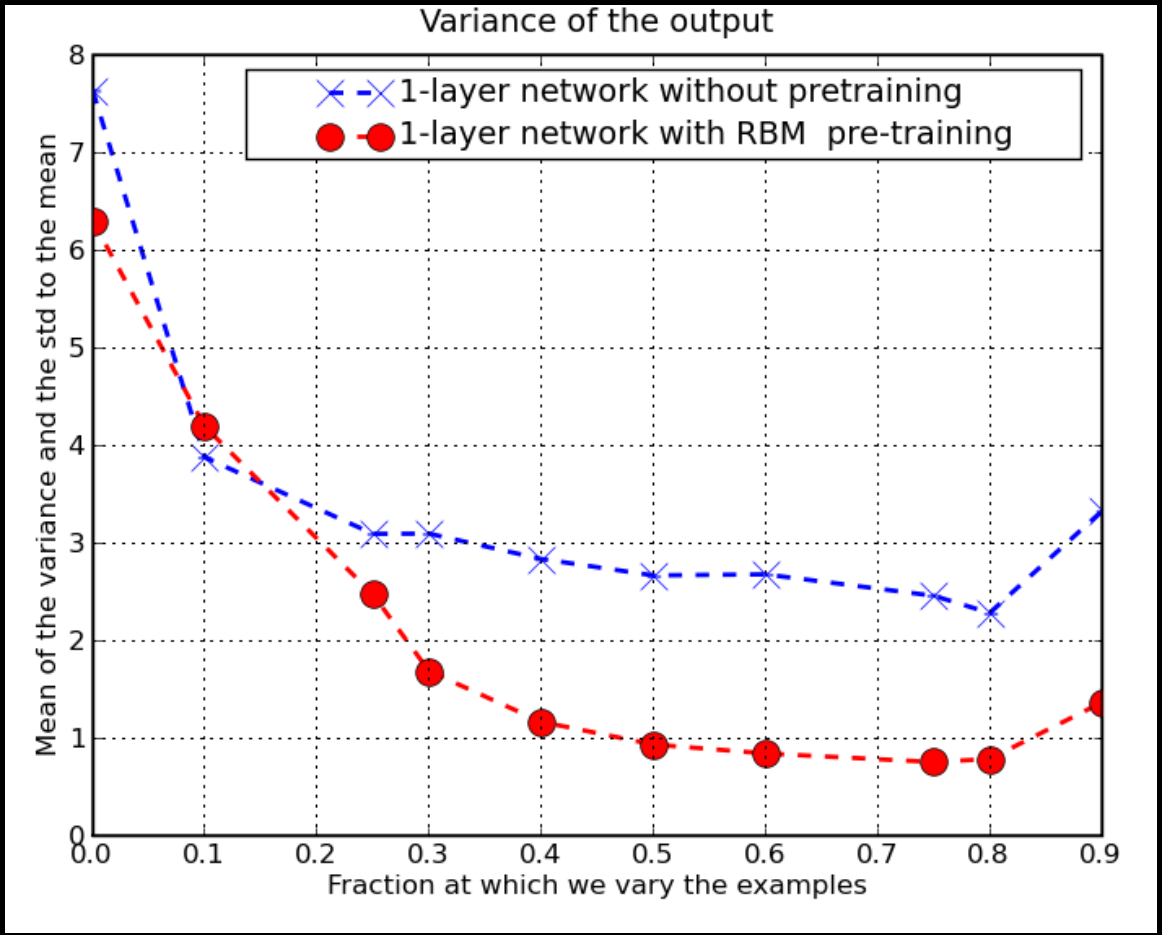


Better Optimization of Online Error

- Both training and online error are smaller with unsupervised pre-training
- As # samples → ∞ training err. = online err. = generalization err.
- Without unsup. pre-training: can't exploit capacity to capture complexity in target function from training data

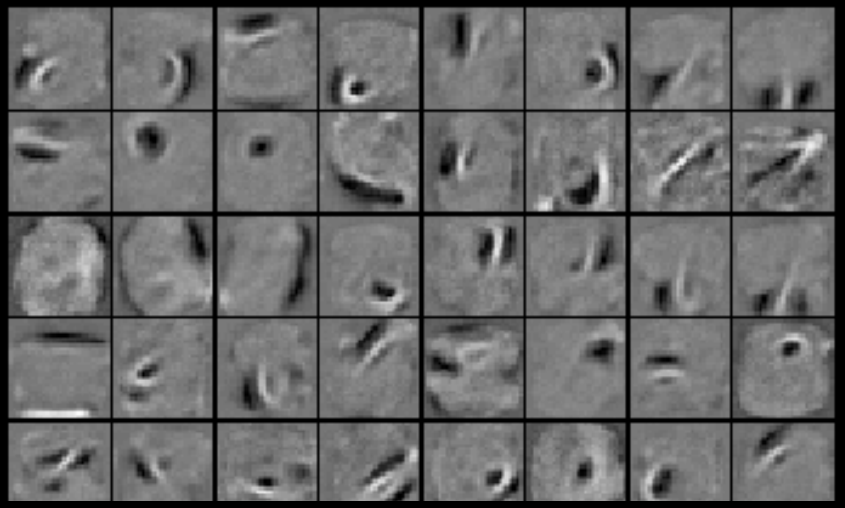


Initial Examples Matter More (critical period?)

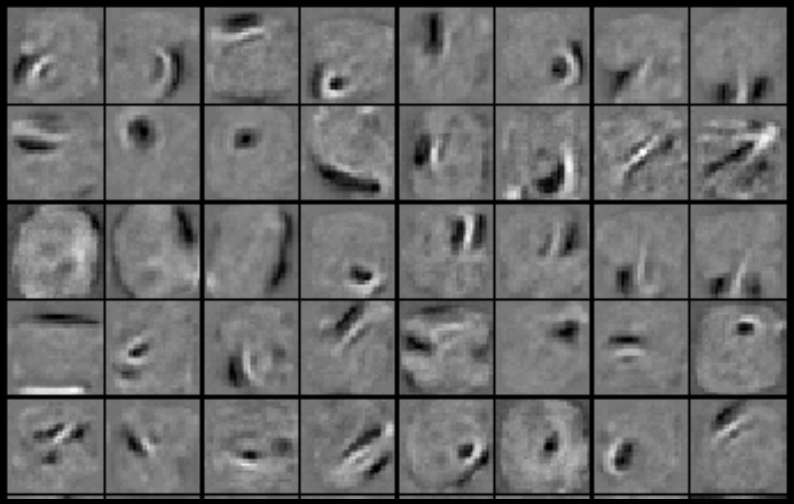


Vary 10% of the training set at the beginning, middle, or end of the online sequence. Measure the effect on learned function.

Learning Dynamics of Deep Nets

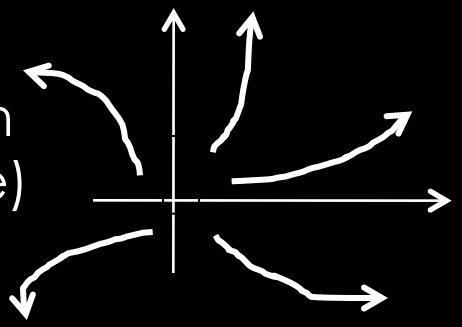


Before fine-tuning



After fine-tuning

- As weights become larger, get trapped in basin of attraction (sign does not change)
- Critical period. Initialization matters.



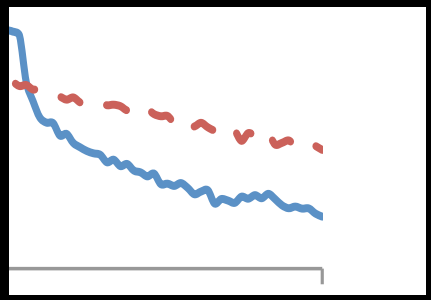
Order & Selection of Examples Matters

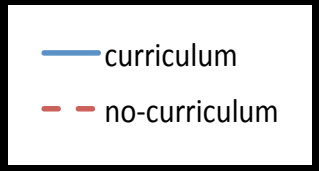
(with Louradour, Collobert & Weston, ICML'2009)

- Curriculum learning

(Bengio et al, ICML'2009; Krueger & Dayan 2009)

- Start with easier examples
- Faster convergence to a better local minimum in deep architectures
- Also acts like a regularizer with optimization effect?
- Influencing learning dynamics can make a big difference





New Developments in Optimizing Deep Architectures

- Hessian-Free (HF) optimization
  - Applied to deep auto-encoders (Martens, ICML 2010)
  - Applied to recurrent nets & modeling text (Martens & Sutskever (& Hinton), ICML 2011)
- Large minibatches (also at Stanford)
- High-curvature directions correlated with small (but important) components of gradient

Unsupervised Learning: Disentangling Factors of Variation

- *(Goodfellow et al NIPS'2009)*: some hidden units more invariant (with more depth) to input geometry variations
- *(Glorot et al ICML'2011)*: some hidden units specialize on one aspect (domain) while others on another (sentiment)
- We don't want invariant representations because it is not clear to what aspects, but disentangling factors would help a lot
- Sparse/saturated units seem to help
- Why?
- **How to train more towards that objective?**

Temporal Coherence and Scales

- One of the hints from nature about different explanatory factors:
  - Rapidly changing factors (often noise)
  - Slowly changing (generally more abstract)
  - Different factors at different time scales

- We should exploit those hints!
- (Becker & Hinton 1993, Wiskott & Sejnowski 2002, Hurri & Hyvarinen 2003, Berkes & Wiskott 2005, Mobahi et al 2009, Bergstra & Bengio 2009)

Advantages of Sparse Representations

- Information disentangling (compare to dense compression).
- More likely to be linearly separable (high-dimensional space).
- Locally low-dimensional representation = local chart
- Efficient variable size representation.

Few bits of information

Many bits of information

Sparsity as a Disentangling Hint

- Look for a few ‘explanations’
- Mixing a sparse signal = entangling
- Sparse representations: add a sparsity penalty
- Group sparsity with different $L_{p,q}$ on different types of coefficients can be used to induce a separation between them (Kowalski)

Challenge #4: What criteria or gradient estimators to train unsupervised non-linear feature extractors (since straight maximum-likelihood is not straightforward at all)

The Partition Function Gradient

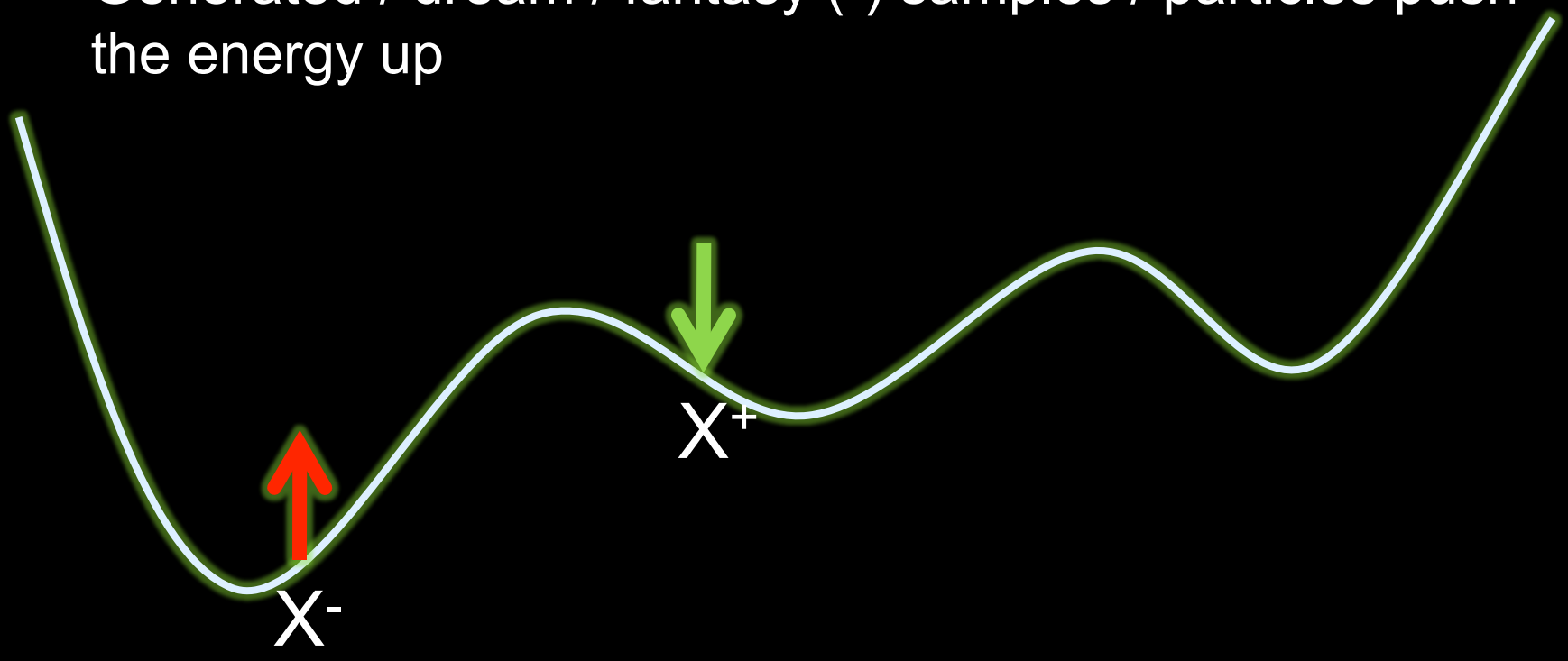
$$p_\theta(x) = \frac{e^{-\text{energy}(x,\theta)}}{\sum_x e^{-\text{energy}(x,\theta)}} = \frac{e^{-\text{energy}(x,\theta)}}{Z(\theta)}$$

$$\begin{aligned}\frac{\partial \log p_\theta(x)}{\partial \theta} &= -\frac{\partial \text{energy}(x,\theta)}{\partial \theta} - \frac{\partial Z(\theta)}{\partial \theta} \\ &= -\frac{\partial \text{energy}(x,\theta)}{\partial \theta} + \sum_x p_\theta(x) \frac{\partial \text{energy}(x,\theta)}{\partial \theta}\end{aligned}$$

- Untractable sum (or integral)
- Positive example (observed *x*) vs negative example (sampled *x*)

Positive & Negative Samples

- Observed (+) examples push the energy down
- Generated / dream / fantasy (-) samples / particles push the energy up



Palette of Tricks to Train Energy-Based Models

Partition function expensive (vocab.) or intractable

- Contrastive Divergence
- PCD / SML + MCMC tricks
  - Tempering
  - Mean-field / variational, etc.
- (regularized) Score Matching / denoising
- Sparse coding / Sparse Predictive Decomposition
- Ratio Matching
- Pseudo-likelihood
- Ranking / margin-based criteria
- Noise contrastive estimation

- **Most rely on + vs – examples contrast**

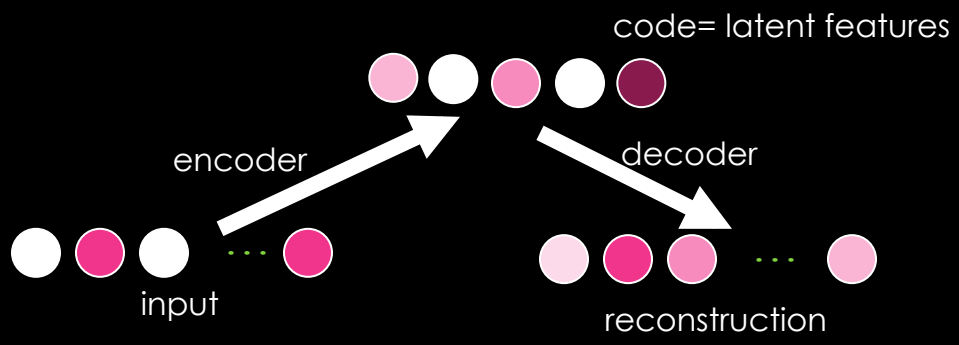
See my book / review paper (F&TML 2009): Learning Deep Architectures for AI

Auto-Encoders

- Reconstruction=decoder(encoder(input)), e.g.

$$
\begin{aligned}
h &= tanh(b + Wx) \\
reconstruction &= tanh(c + W^T h) \\
cost &= ||reconstruction - x||^2
\end{aligned}
$$

- Probable inputs have small reconstruction error
- Linear decoder/encoder = PCA up to rotation
- Minimizing reconstruction error ensures that hidden units capture the directions of largest variation
- Can be stacked successfully (Bengio et al 2006) to form highly non-linear representations, sparse ones increasing disentangling (Goodfellow et al, NIPS 2009)
- What is the corresponding probabilistic model?



Sparse Auto-Encoders

- Successfully used by Andrew Ng's group at Stanford (e.g. ICML 2011)
- Used by Google in their Google Goggles vision system
- Sparsity penalty = binomial KL div. between mean output prob. (over minibatch) and small target prob. (0.05), which works also on RBMs
- Prevents units from becoming always stuck at 0

Link Between Contrastive Divergence and Auto-Encoder Reconstruction Error Gradient

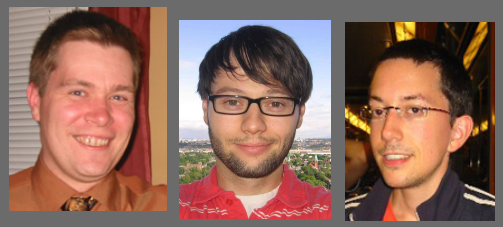
- (Bengio & Delalleau 2009):

  - CD-2k estimates the log-likelihood gradient from 2k diminishing terms of an expansion that mimics the Gibbs steps
  - reconstruction error gradient looks only at the first step, i.e., is a kind of mean-field approximation of CD-0.5

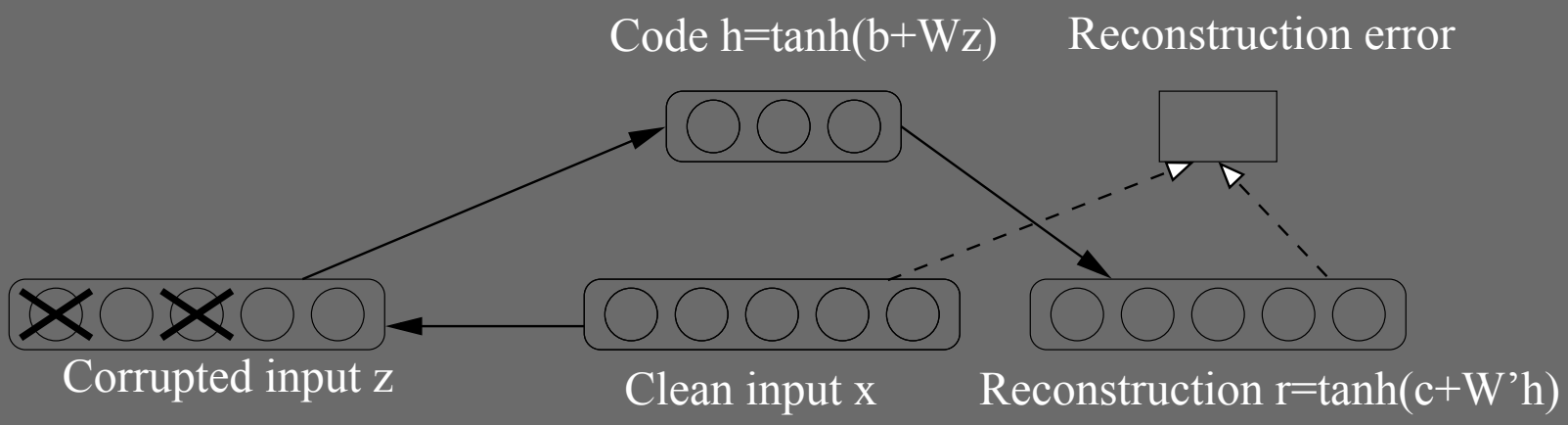
$$\frac{\partial \log P(x_1)}{\partial \theta} = \sum_{s=1}^{t-1} \left(E\left[\frac{\partial \log P(x_s|h_s)}{\partial \theta} \middle| x_1 \right] + E\left[\frac{\partial \log P(h_s|x_{s+1})}{\partial \theta} \middle| x_1 \right] \right) + E\left[\frac{\partial \log P(x_t)}{\partial \theta} \middle| x_1 \right]$$

Denoising Auto-Encoder

(Vincent et al 2008, 2010)

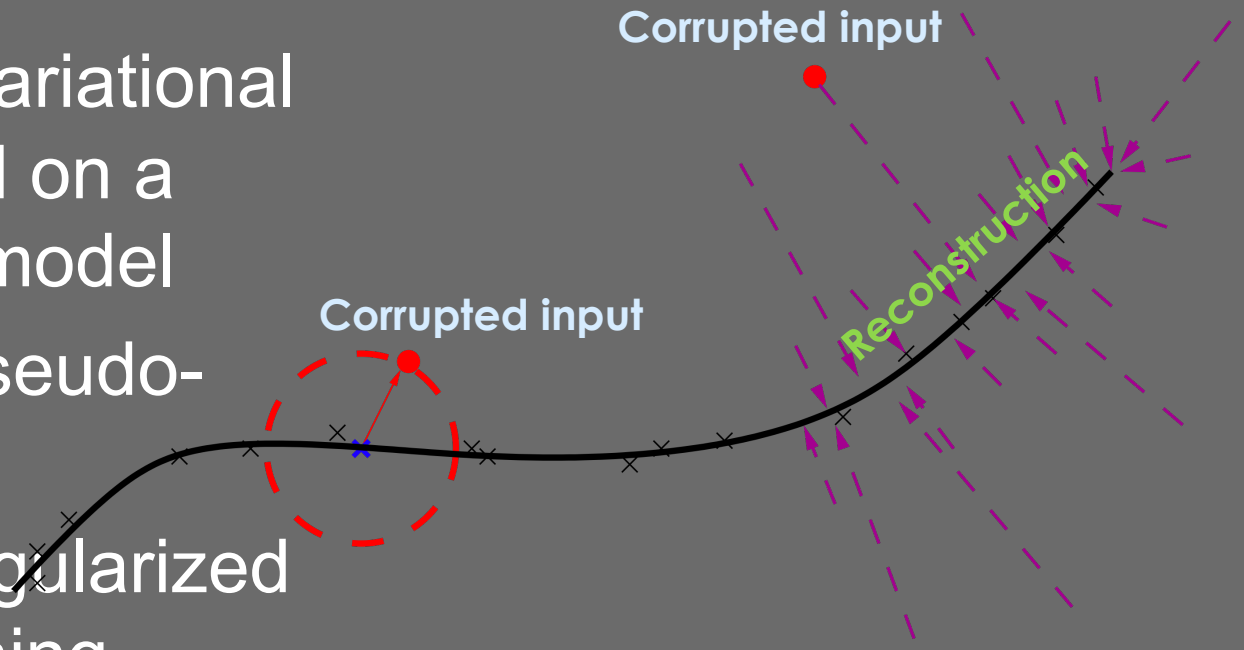


- Stochastically corrupt the input
- Reconstruction target = clean input



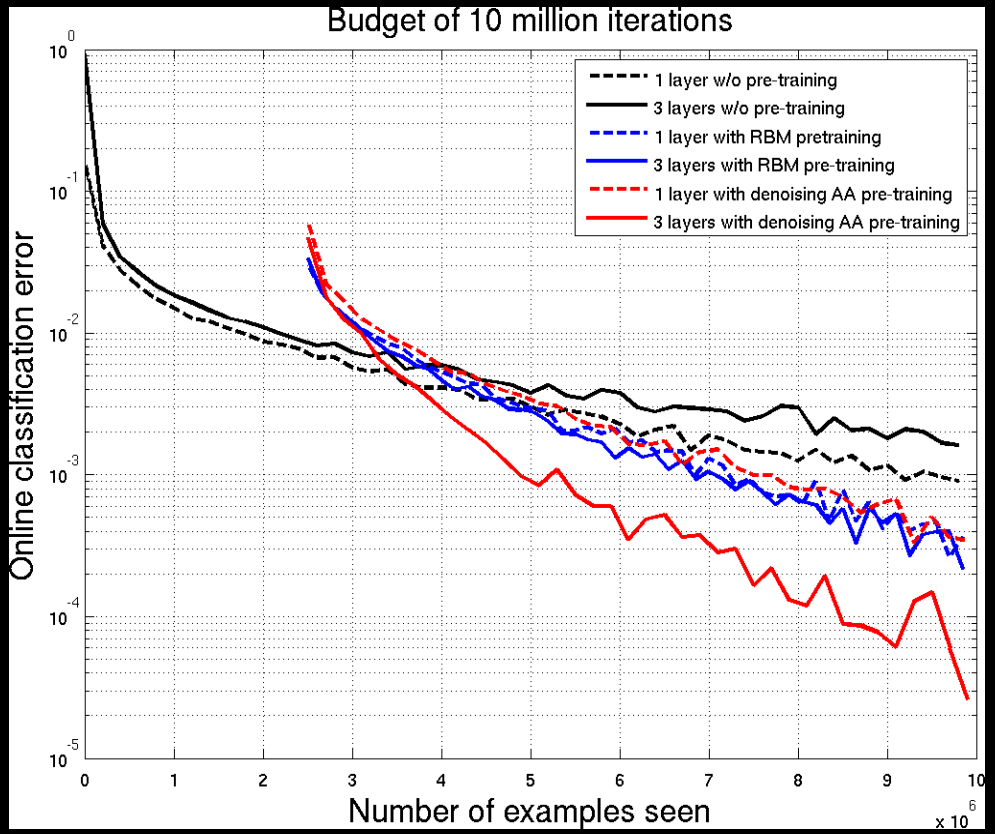
Denoising Auto-Encoder

- Learns a vector field towards higher probability regions
- Minimizes variational lower bound on a generative model
- Similar to pseudo-likelihood
- A form of regularized score matching



Stacked Denoising Auto-Encoders

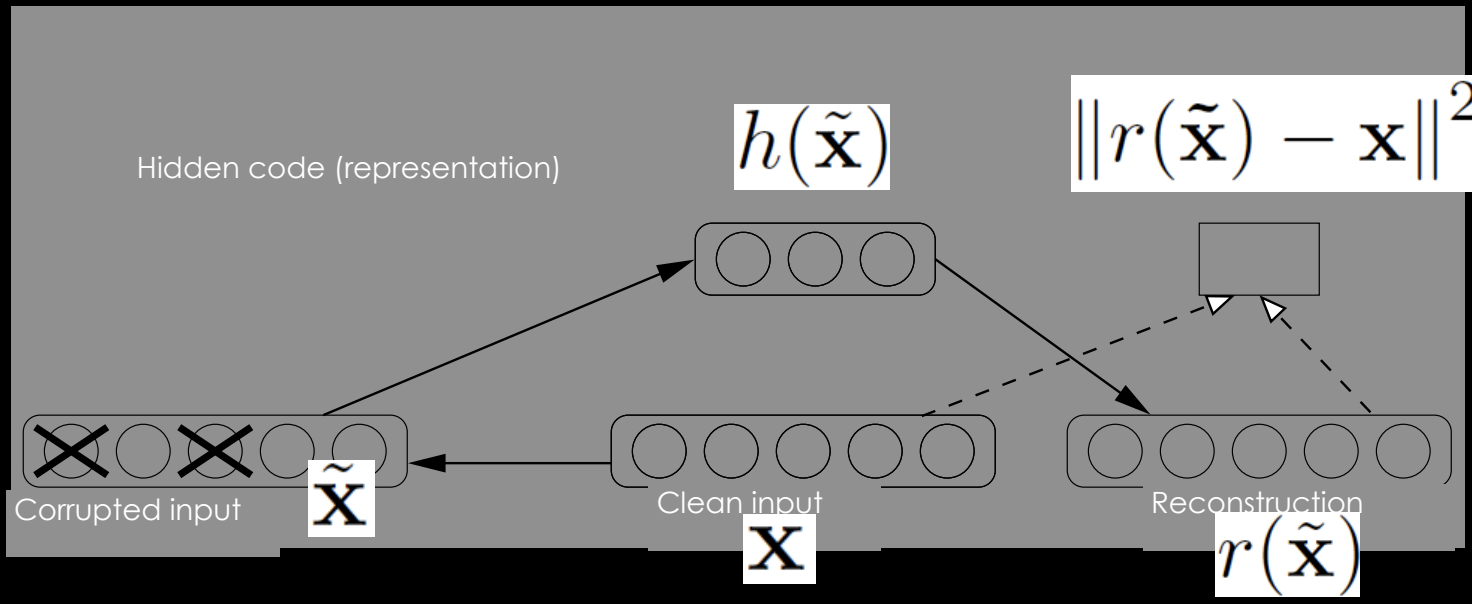
- No partition function, can measure training criterion
- Encoder & decoder: any parametrization
- Performs as well or better than stacking RBMs for unsupervised pre-training



Infinite MNIST

Stacked Denoising Auto-Encoders

- Layerwise unsupervised pre-training *(Vincent et al, ICML'08)*
- Corrupt the input $\tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) = N(\tilde{\mathbf{x}}; \mathbf{x}, \sigma^2 I)$
- Try reconstructing the clean (uncorrupted) input
- Better results with noise variance away from 0
- Use uncorrupted encoding as input to next level



Score Matching (Hyvarinen 2005)

- Score of model *p*: dlogP(x)/dx does not contain partition fn Z
- Matching score of *p* to target score: ?

$$\mathbb{E}_{q(\mathbf{x})}\left[\frac{1}{2}\left\|\frac{\partial \log p(\mathbf{x})}{\partial \mathbf{x}} - \frac{\partial \log q(\mathbf{x})}{\partial \mathbf{x}}\right\|^2\right]$$

- Hyvarinen shows it equals

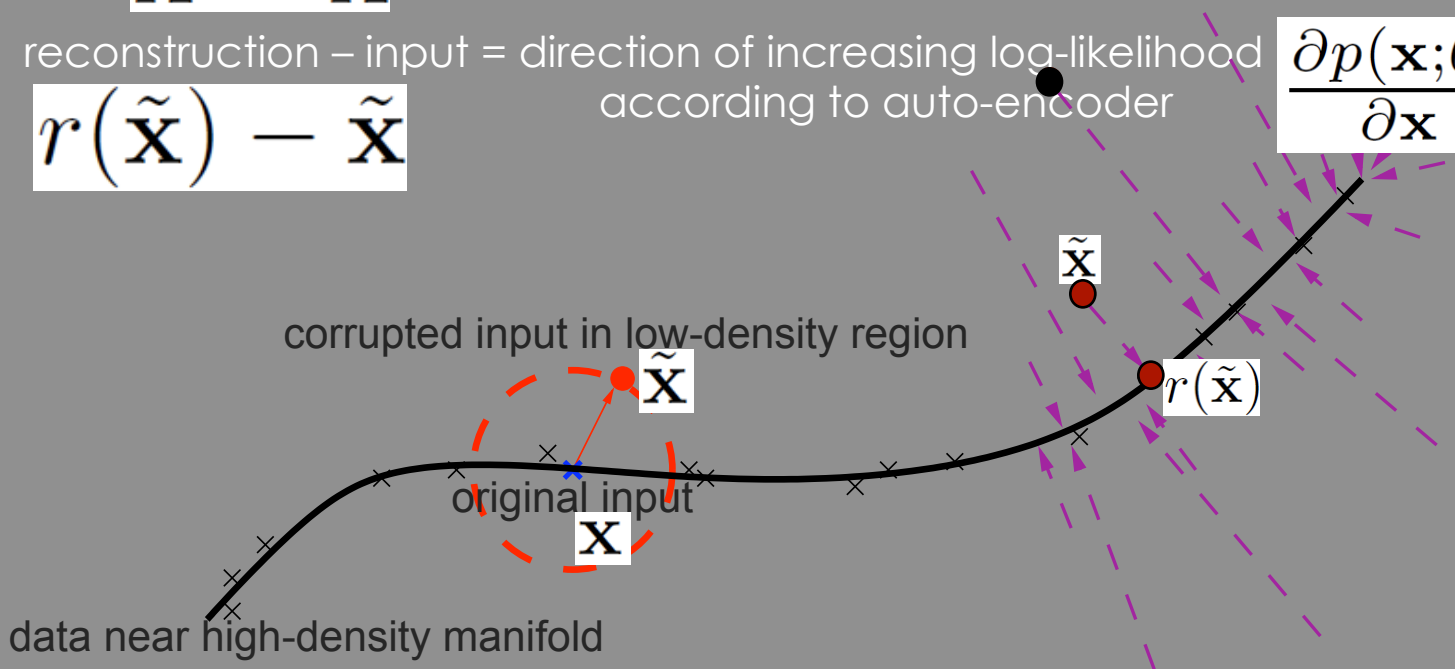
$$\mathbb{E}_{q(\mathbf{x})}\left[\frac{1}{2}\left\|\frac{\partial \log p(\mathbf{x})}{\partial \mathbf{x}}\right\|^2 + \sum_i \frac{\partial^2 \log p(\mathbf{x})}{\partial \mathbf{x_i^2}}\right] + const$$

- and proposes to minimize corresponding empirical mean
- Shown to be asymptotically unbiased to estimate parameters
- **Requires O(#parameters x #inputs) computation!**

Denoising Auto-Encoders doing Score Matching on Gaussian RBMs

P. Vincent, 2010

- clean input - corrupted input = direction of increasing log-likelihood in generating density $\mathbf{x} - \tilde{\mathbf{x}}$ $\frac{\partial \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})}{\partial \tilde{\mathbf{x}}}$
- reconstruction – input = direction of increasing log-likelihood according to auto-encoder $r(\tilde{\mathbf{x}}) - \tilde{\mathbf{x}}$ $\frac{\partial p(\mathbf{x};\theta)}{\partial \mathbf{x}}$



- Denoising error = $\left\| (r(\tilde{\mathbf{x}}) - \tilde{\mathbf{x}}) - (\mathbf{x} - \tilde{\mathbf{x}}) \right\|^2 = \left\| r(\tilde{\mathbf{x}}) - \mathbf{x} \right\|^2$

Sparse Predictive Decomposition

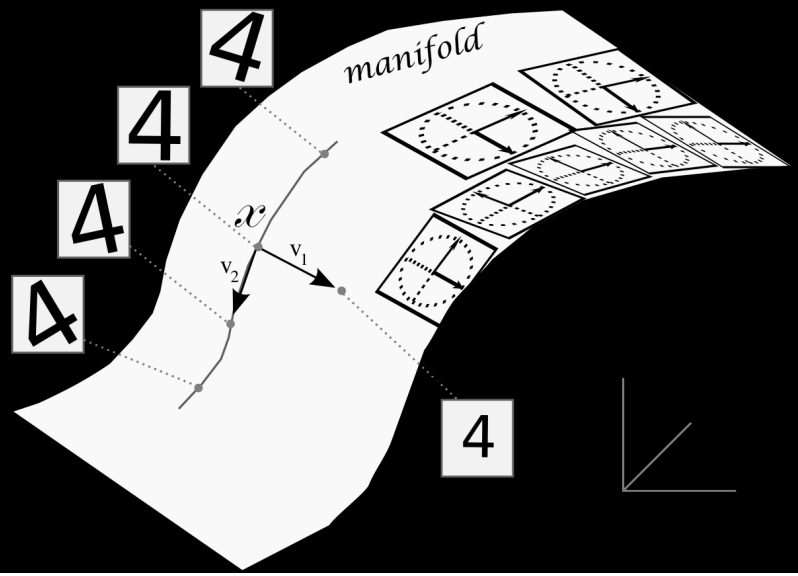
- From LeCun's group over last 5 years
- Sparse coding + parametric encoder + penalty between output of parametric encoder and non-parametric sparse code:

$$||x - W\,z||^2 + \lambda_1\,|z|_1 + \lambda_2\,||z - W'x||^2$$

- Initialize FISTA with W'x: much faster encoding than sparse coding, much faster dictionary learning
- Successful (deep) representation learning in object recognition (MNIST, Caltech, pedestrian detection)

Representations as Coordinate Systems

- PCA: removing low-variance directions → easy but what if signal has low variance? We would like to disentangle factors of variation, keeping them all.
- Overcomplete representations: richer, even if underlying distribution concentrates near low-dim manifold.
- Sparse/saturated features: allows for variable-dim manifolds. Different few sensitive features at *x* = local chart coordinate system.

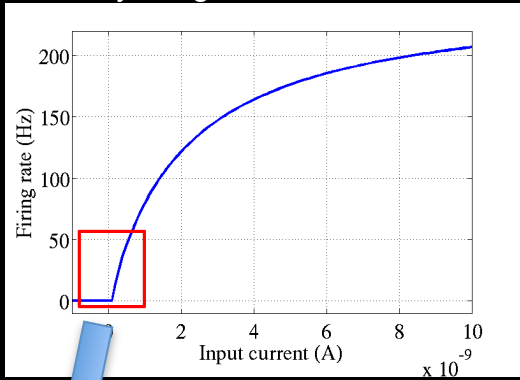


Deep Sparse Rectifier Neural Networks

X. Glorot, A. Bordes and Y. Bengio, following up on (Nair & Hinton 2010)

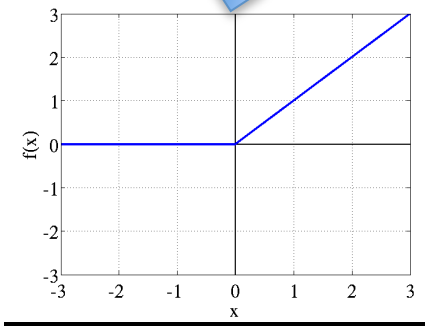
Neuroscience motivations

Leaky integrate-and-fire model

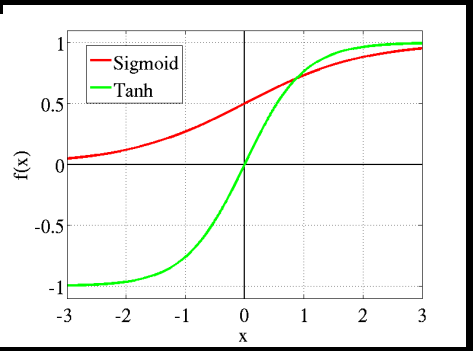


Machine learning motivations

- Sparse representations
- Sparse and linear gradients



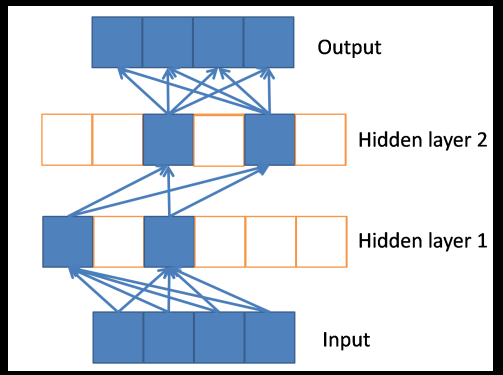
Rectifier



Commonly used functions

$f(x)=\max(0,x)$

- *One-sided*
- Real zeros
- "default" regime at 0



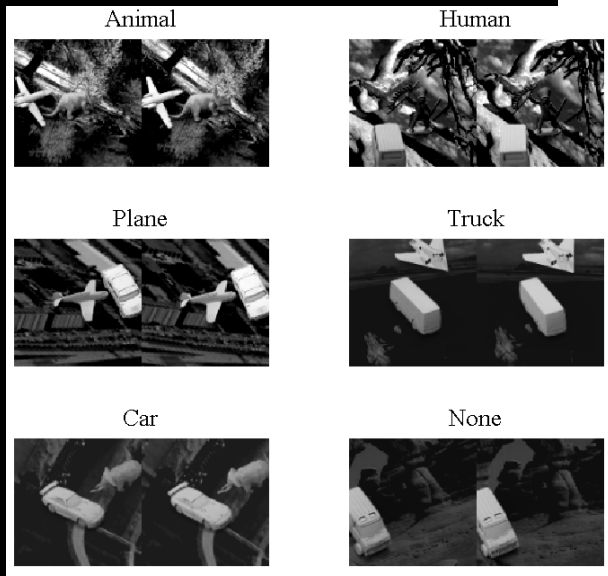
Deep Sparse Rectifier Neural Nets: Results

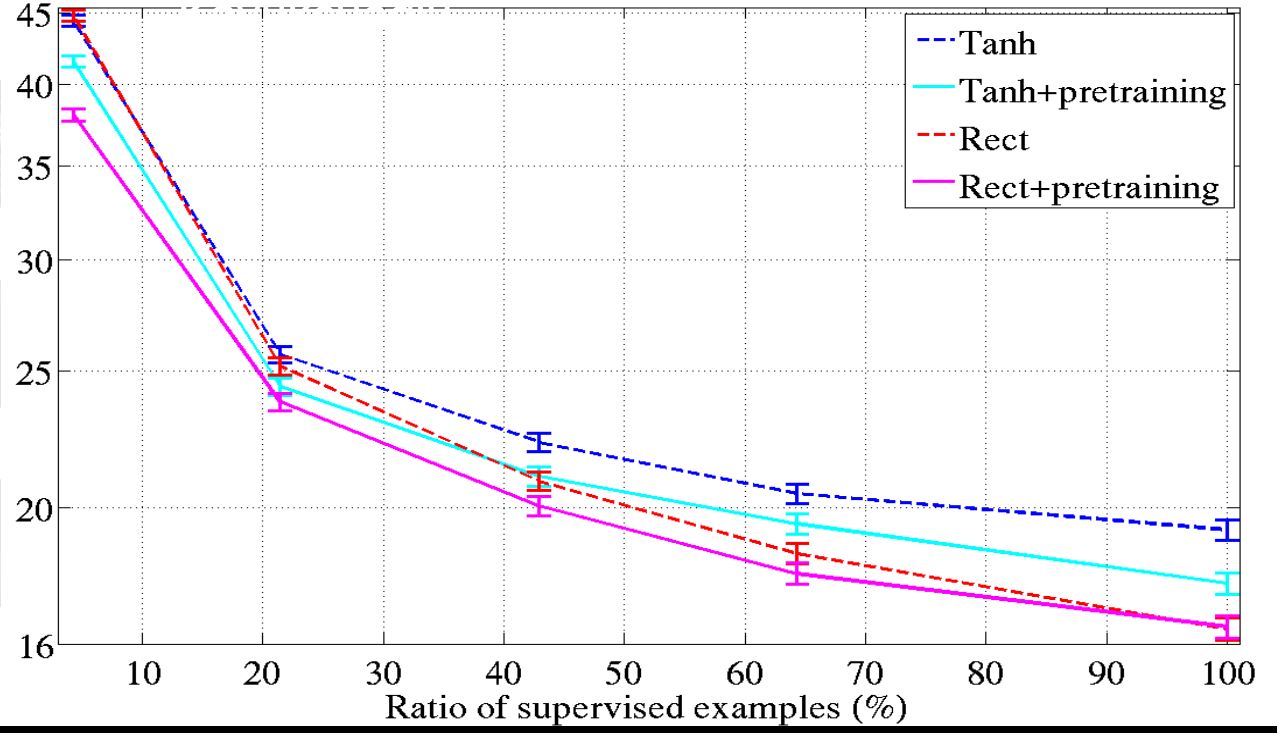
Experiments and results

- Stacked denoising autoencoder
- 4 image recognition and 1 sentiment analysis datasets
- Better generalization than hyperbolic tangent networks
- Rectifier networks achieve their best performance without needing unsupervised pre-training
- Unsupervised pre-training is beneficial in the semi-supervised setting

Neuron	MNIST	CIFAR10	NISTP	NORB
With unsupervised pre-training				
Rectifier	**1.20%**	**49.96%**	**32.86%**	**16.46%**
Tanh	**1.16%**	**50.79%**	35.89%	17.66%
Softplus	**1.17%**	**49.52%**	**33.27%**	19.19%
Without unsupervised pre-training				
Rectifier	**1.43%**	**50.86%**	**32.64%**	**16.40%**
Tanh	1.57%	52.62%	36.46%	19.29%
Softplus	1.77%	53.20%	35.48%	17.68%

NISTP



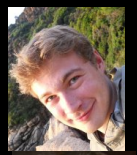


Sparse Auto-Encoders & Sparse Predictive Decomposition

(Ranzato et al, 2007; Ranzato et al 2008, Kavukcuoglu et al 2009, 2010)

- Sparsity penalty on the intermediate codes
- Like sparse coding but with efficient run-time encoder
- Sparsity penalty pushes up the free energy of all configurations (proxy for minimizing the partition function)
- Impressive results in object classification (convolutional nets):
  - MNIST .4% error = record-breaking
  - Caltech-101 65% correct = state-of-the-art (Jarrett et al, ICCV 2009)
- Similar results obtained with a convolutional DBN (Lee et al, ICML'2009)

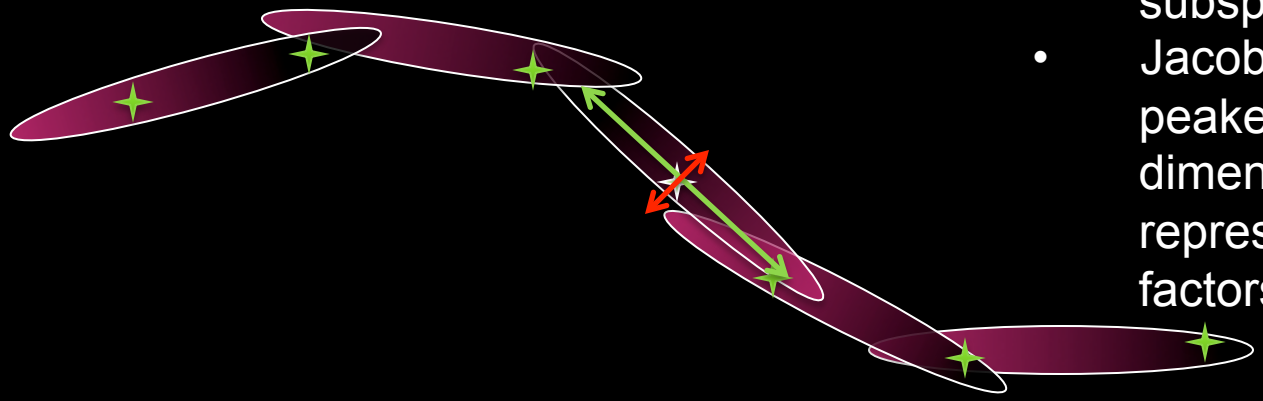



Contractive Auto-Encoders

- *Contractive Auto-Encoders: Explicit Invariance During Feature Extraction*, Rifai, Vincent, Muller, Glorot & Bengio, ICML 2011.
- *Higher Order Contractive Auto-Encoders*, Rifai, Mesnil, Vincent, Muller, Bengio, Dauphin, Glorot, ECML 2011.
- *Part of winning toolbox in final phase of the Unsupervised & Transfer Learning Challenge 2011*

Contractive Auto-Encoders



- Few active units represent the active subspace (local chart)
- Jacobian's spectrum is peaked = local low-dimensional representation / relevant factors

cannot afford contraction in manifold directions

wants contraction in all directions

Training criterion:

$$\mathcal{J}_{\mathrm{CAE}}(\theta) = \sum_{x \in D_n} \left(L(x, g(h(x))) + \lambda \sum_{ij} \left(\frac{\partial h_j(x)}{\partial x_i} \right)^2 \right.$$

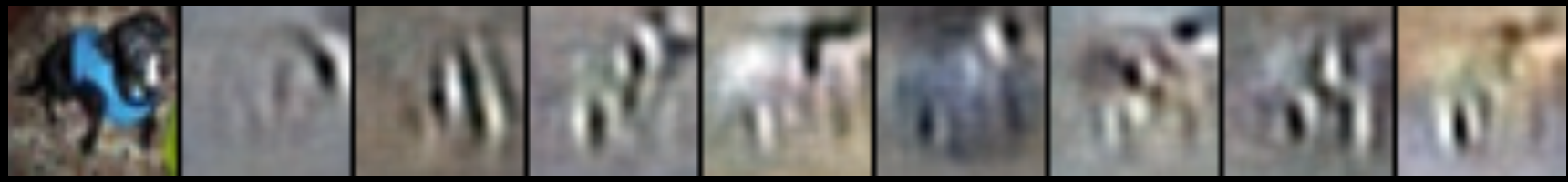
Contractive Auto-Encoders

Data Set	SVM_{rbf}	SAE-3	RBM-3	DAE-b-3	CAE-1	CAE-2
basic	**3.03**$\pm$0.15	3.46$\pm$0.16	3.11$\pm$0.15	2.84$\pm$0.15	2.83$\pm$0.15	**2.48**$\pm$0.14
rot	11.11$\pm$0.28	10.30$\pm$0.27	10.30$\pm$0.27	**9.53**$\pm$0.26	11.59$\pm$0.28	**9.66**$\pm$0.26
bg-rand	14.58$\pm$0.31	11.28$\pm$0.28	**6.73**$\pm$0.22	10.30$\pm$0.27	13.57$\pm$0.30	10.90 $\pm$0.27
bg-img	22.61$\pm$0.379	23.00$\pm$0.37	16.31$\pm$0.32	16.68$\pm$0.33	16.70$\pm$0.33	**15.50**$\pm$0.32
bg-img-rot	55.18$\pm$0.44	51.93$\pm$0.44	47.39$\pm$0.44	**43.76**$\pm$0.43	48.10$\pm$0.44	45.23$\pm$0.44
rect	**2.15**$\pm$0.13	2.41$\pm$0.13	2.60$\pm$0.14	1.99$\pm$0.12	1.48$\pm$0.10	**1.21**$\pm$0.10
rect-img	24.04$\pm$0.37	24.05$\pm$0.37	22.50$\pm$0.37	**21.59**$\pm$0.36	**21.86**$\pm$0.36	**21.54**$\pm$0.36

- Most hidden units saturate
- One a few are active and represent the active subspace (local chart)
- Jacobian's spectrum is peaked = local low-dimensional representation / relevant factors

Distributed vs Local (CIFAR-10 unsupervised)

- CAE manifold directions



- Local PCA directions

Learned Tangent Prop: the Manifold Tangent Classifier

3 hypotheses:

1. Semi-supervised hypothesis (P(x) related to P(y|x))
2. Unsupervised manifold hypothesis (data concentrates near low-dim. manifolds)
3. Manifold hypothesis for classification (low density between class manifolds)

Algorithm:

1. Estimate local principal directions of variation U(x) by CAE (principal singular vectors of dh(x)/dx)
2. Penalize f(x)=P(y|x) predictor by || df/dx U(x) ||

Manifold Tangent Classifier Results

- Leading singular vectors on MNIST, CIFAR-10, RCV1:

Trading & Markets	+gilt +yen +usda	-slow -term -debt	+matur +auction +treasur	-percent -sent -pressure	+bln +coupon +discount	-anti -predict -belgian	+interest +calcul +overnight	-sen -californ -introduc

- **Knowledge-free MNIST: 0.81% error**

K-NN	NN	SVM	DBN	CAE	DBM	CNN	MTC
3.09%	1.60%	1.40%	1.17%	1.04%	0.95%	0.95%	**0.81%**

- Semi-sup.

	NN	SVM	CNN	TSVM	DBN-rNCA	EmbedNN	CAE	MTC
100	25.81	23.44	22.98	16.81	-	16.86	13.47	**12.03**
600	11.44	8.85	7.68	6.16	8.7	5.97	6.3	**5.13**
1000	10.7	7.77	6.45	5.38	-	5.73	4.77	**3.64**
3000	6.04	4.21	3.35	3.45	3.3	3.59	3.22	**2.57**

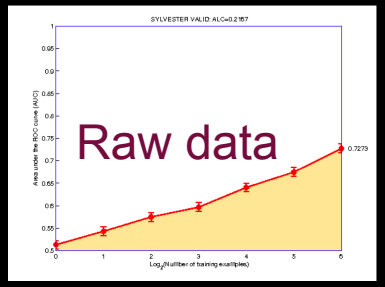
- Forest (500k examples)

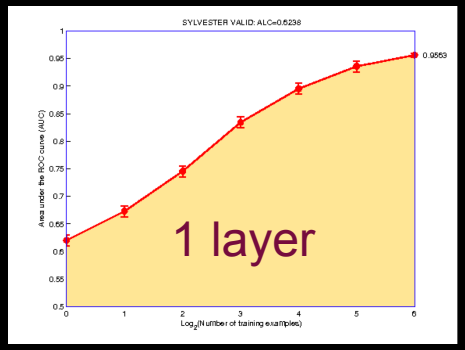
SVM	Distributed SVM	MTC
4.11%	3.46%	**3.13%**

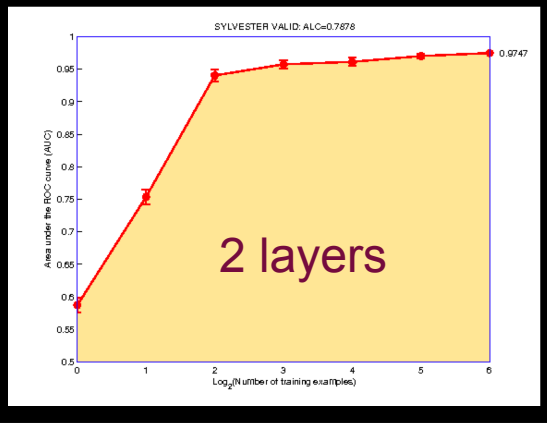
Unsupervised and Transfer Learning Challenge

- 5 datasets, 73 entrants, knowledge-free
- Goal: learning good representations from unlabeled examples of training classes so that they generalize well to unknown test classes.
- Protocol: given 4096 test inputs, provide their representation. Server's Hebbian classifier trained on top. No label of test classes given.

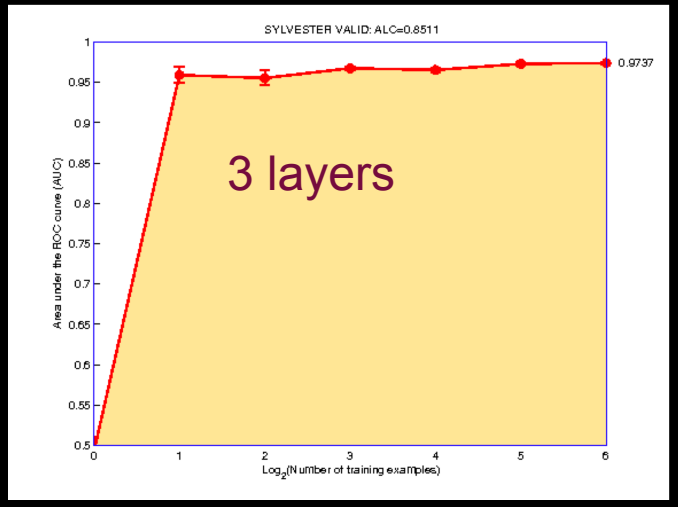
Unsupervised and Transfer Learning Challenge: 1st Place in Final Phase

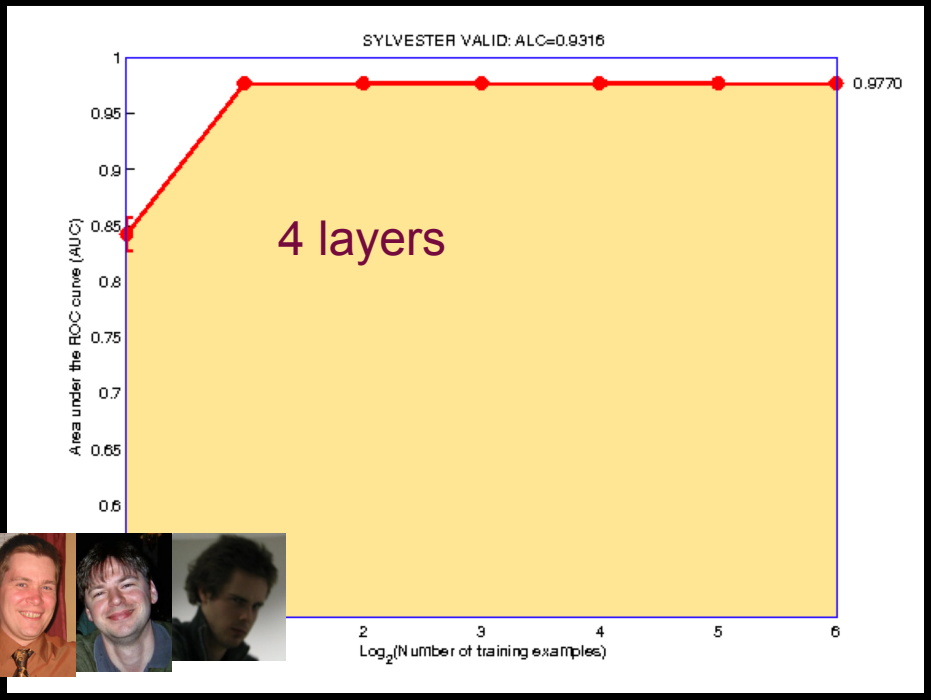






ICML'2011 workshop on UTL





Opportunity #1: DL Applications and Architectures

Recent DL Highlights

- Google Goggles uses stacked sparse auto-encoders (Hartmut Neven @ ICML 2011)
- UofT breaks old accuracy ceiling in TIMIT phoneme detection
- Microsoft (Li Deng) breaks speech recognition records (WER) using deep architectures
- Stanford breaks records in video / gesture classification
- NYU breaks records in traffic sign class.
- Montreal wins Unsupervised & Transfer Learning Challenge
- IBM working with LeCun's lab on DBNs for speech
- Mikolov (Czech Rep. + JHU & Microsoft people) RNN LM reduces Broadcast News WER by 10% vs 4-gram (13.1→11.8%)
- DARPA Deep Learning program (LeCun, Bengio, Ng)

Spike & Slab RBMs

- *A Spike and Slab Restricted Boltzmann Machine*, Courville, Bergstra & Bengio, AISTATS 2011.

- *Unsupervised Models of Images by Spike-and-Slab RBMs*, Courville, Bergstra & Bengio, ICML 2011.

- **Latent = binary r.v.** (spike) *x* **cont. r.v** (slab)
- Much better than Gaussian RBM to deal with continuous-valued inputs
- Part of our winning entry to Unsupervised and Transfer Learning Challenge.

Spike & Slab RBMs

$$E(v, s, h) = -\sum_{i=1}^{N} v^T W_i s_i h_i + \frac{1}{2} v^T \left(\Lambda + \sum_{i=1}^{N} \Phi_i h_i \right) v + \frac{1}{2} \sum_{i=1}^{N} \alpha_i s_i^2 - \sum_{i=1}^{N} \alpha_i \mu_i s_i h_i - \sum_{i=1}^{N} b_i h_i + \sum_{i=1}^{N} \alpha_i \mu_i^2 h_i,$$

- Model conditional covariance of pixels (given hidden units) $C_{v|h} = \left(\Lambda + \sum_{i=1}^{N} \Phi_i h_i - \sum_{i=1}^{N} \alpha_i^{-1} h_i W_i W_i^T \right)^{-1}$
- Hidden representation decomposed into a product s*h, h is binary, s is real
- s*h is often 0 (naturally sparse)

Spike & Slab RBMs

$$P(h_i = 1 \mid v) = \sigma(\hat{b}_i - \frac{1}{2}(v - \xi_{v|h_i})^T C_{v|h_i}^{-1}(v - \xi_{v|h_i}))$$

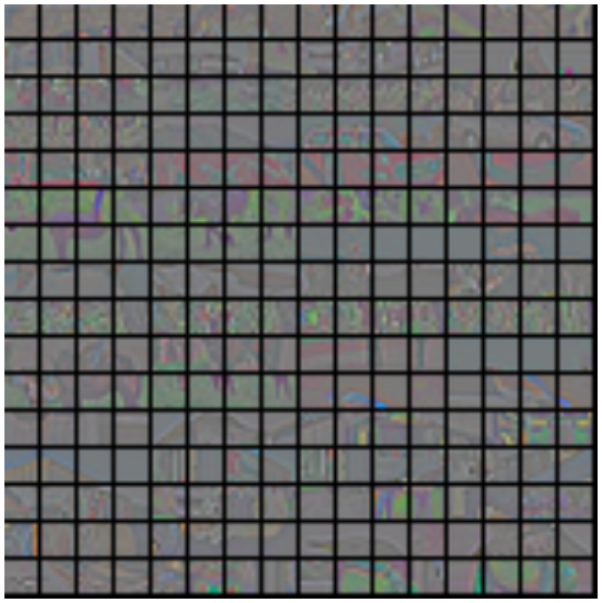
$$p(s \mid v, h) = \prod_{i=1}^{N} \mathcal{N}\left(\left(\alpha_i^{-1} v^T W_i + \mu_i\right) h_i \ , \ \alpha_i^{-1}\right)$$

$$p(v \mid s, h) = \mathcal{N}\left(C_{v|s,h} \sum_{i=1}^{N} W_i s_i h_i \ , \ C_{v|s,h}\right)$$

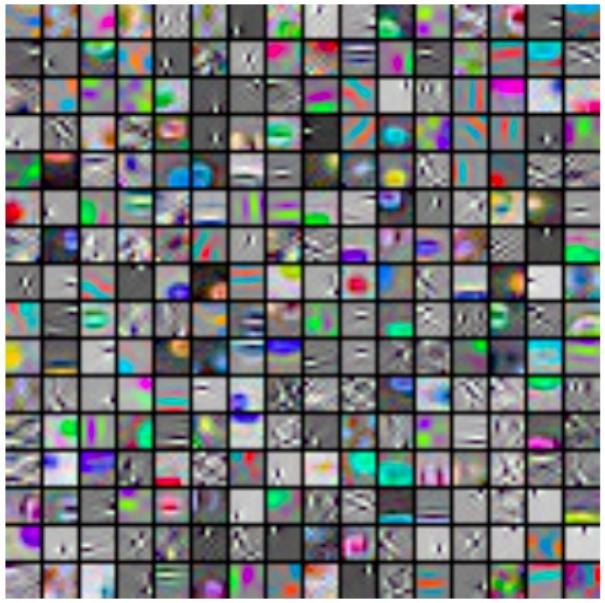
- Can use efficient 3-way Gibbs sampling

Spike & Slab RBMs
CIFAR-10 Filters

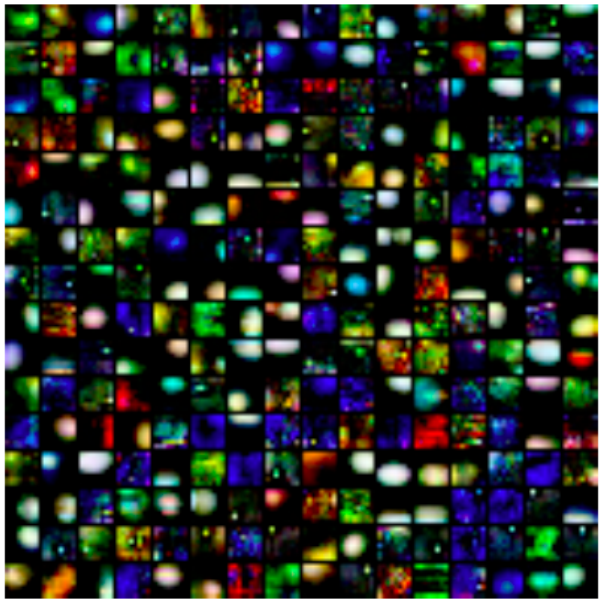
Preprocessed data:



Whitened Patches 8x8

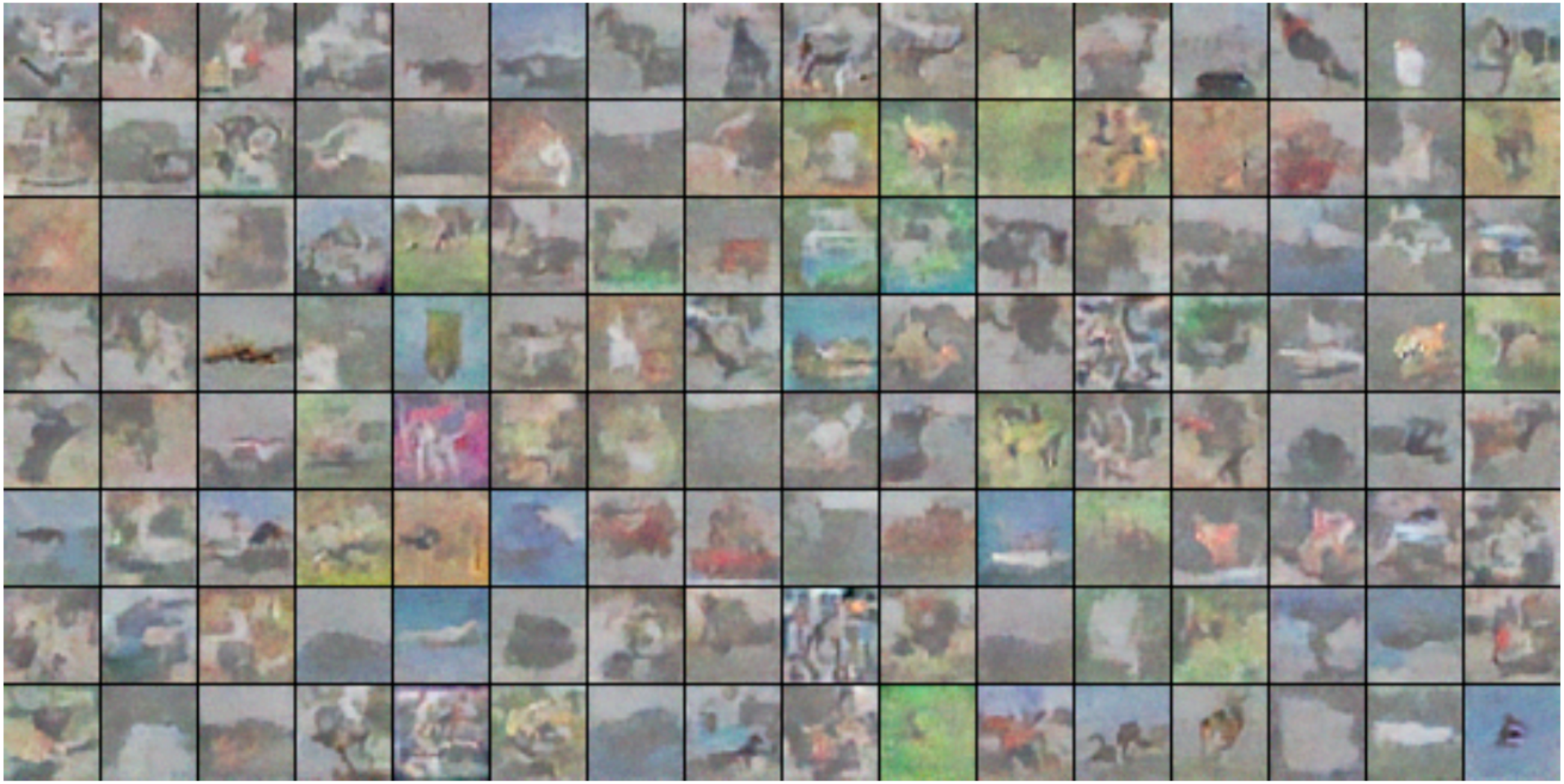


W_i features



Φ_i features

Convolutionally Trained Spike & Slab RBMs Samples



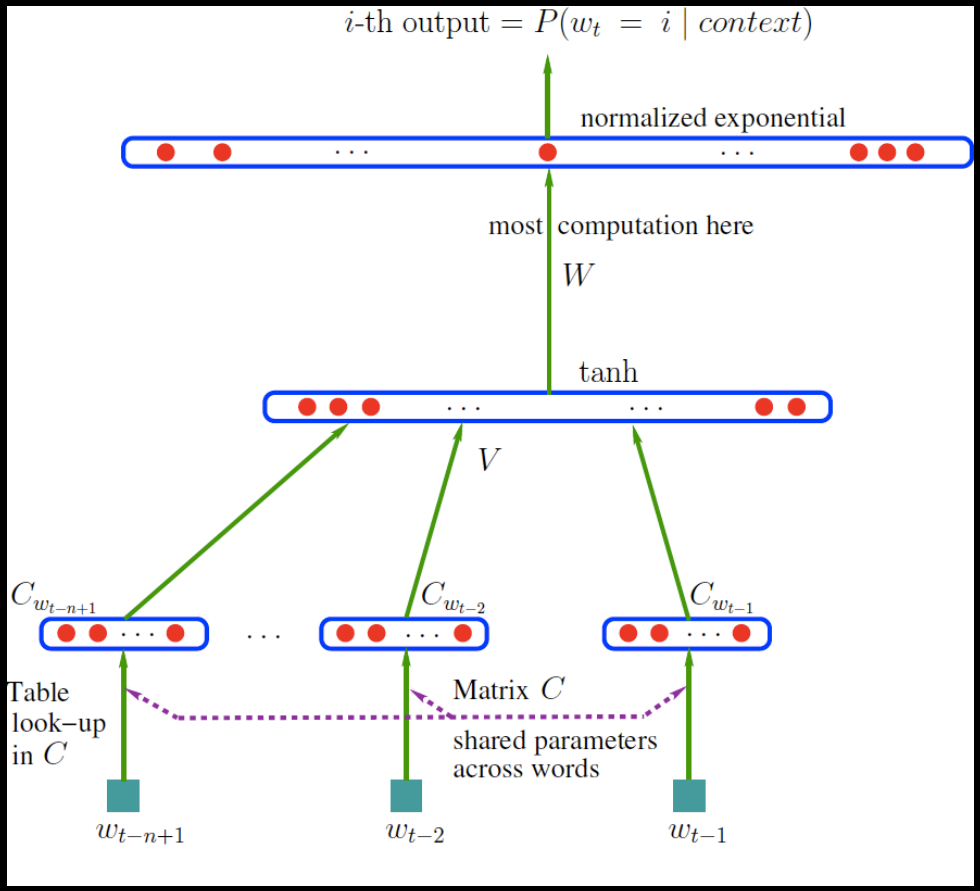
ssRBM is not Cheating

Samples from μ-ssRBM:

Nearest examples in CIFAR:
(least square dist.)

Deep & Distributed NLP

- See "*Neural Net Language Models*" ***Scholarpedia*** *entry*
- *NIPS'2000 and JMLR 2003 "A Neural Probabilistic Language Model"*
  - Each word represented by a distributed continuous-valued code
  - Generalizes to sequences of words that are semantically similar to training sequences



Generalization through distributed semantic representation

- Training sentence

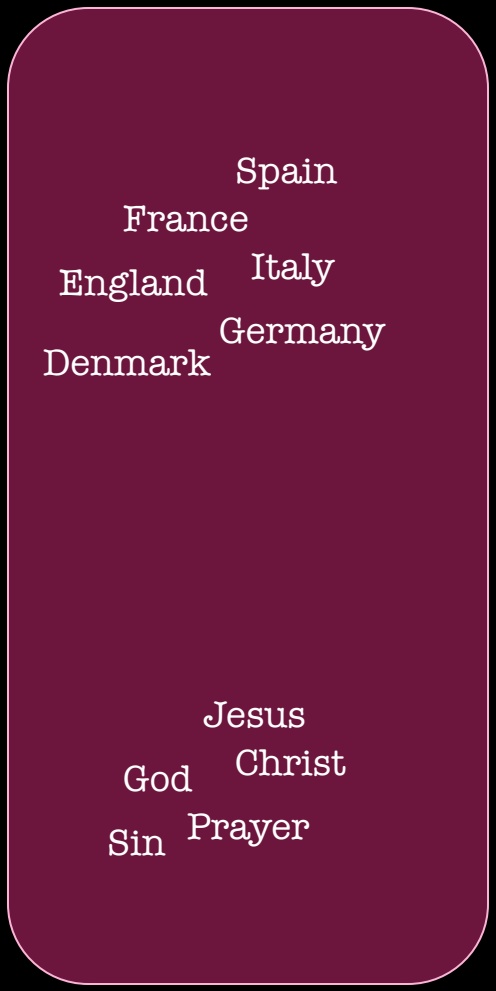
The cat is walking in the bedroom

- can generalize to

A dog was running in a room

- because of the similarity between distributed representations for (a,the), (cat,dog), (is,was), etc.
- Word classes help but are too coarse.

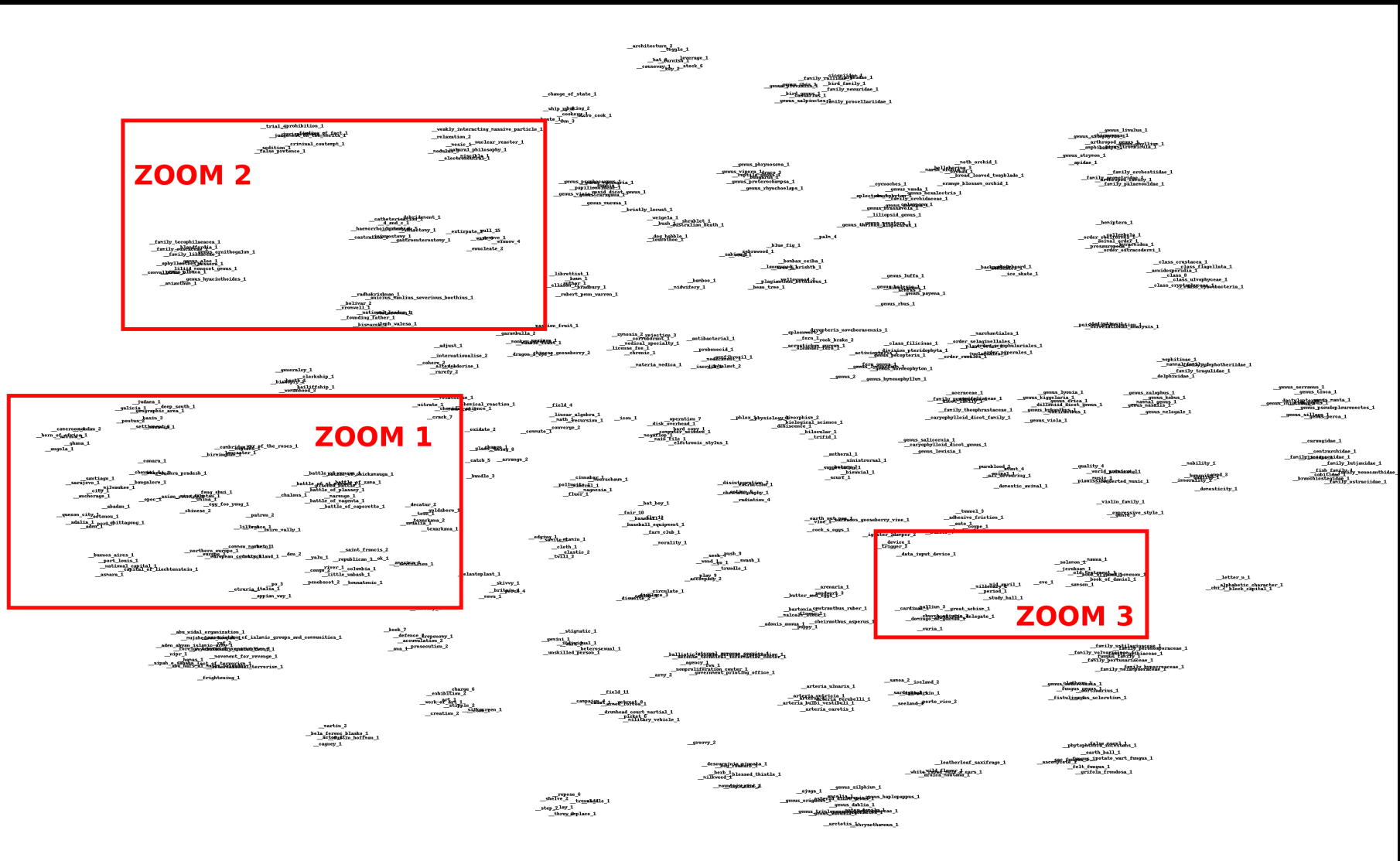
Nearby Words in Semantic Space



Collobert & Weston, ICML'2008

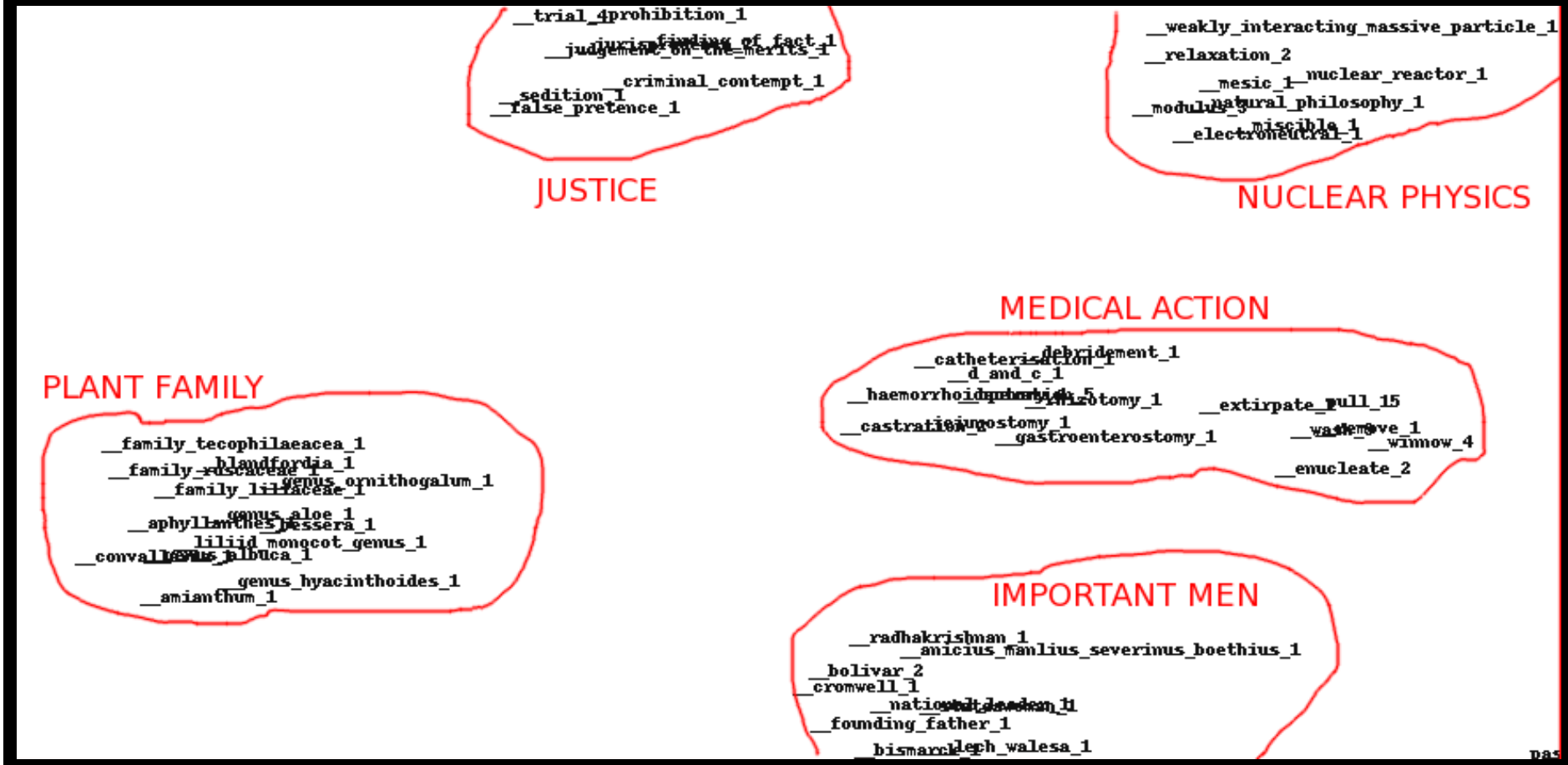
France	Jesus	XBOX	Reddish	Scratched
Spain	Christ	Playstation	Yellowish	Smashed
Italy	God	Dreamcast	Greenish	Ripped
Russia	Resurrection	PS###	Brownish	Brushed
Poland	Prayer	SNES	Bluish	Hurled
England	Yahweh	WH	Creamy	Grabbed
Denmark	Josephus	NES	Whitish	Tossed
Germany	Moses	Nintendo	Blackish	Squeezed
Portugal	Sin	Gamecube	Silvery	Blasted
Sweden	Heaven	PSP	Greyish	Tangled
Austria	Salvation	Amiga	Paler	Slashed

t-SNE of Embeddings

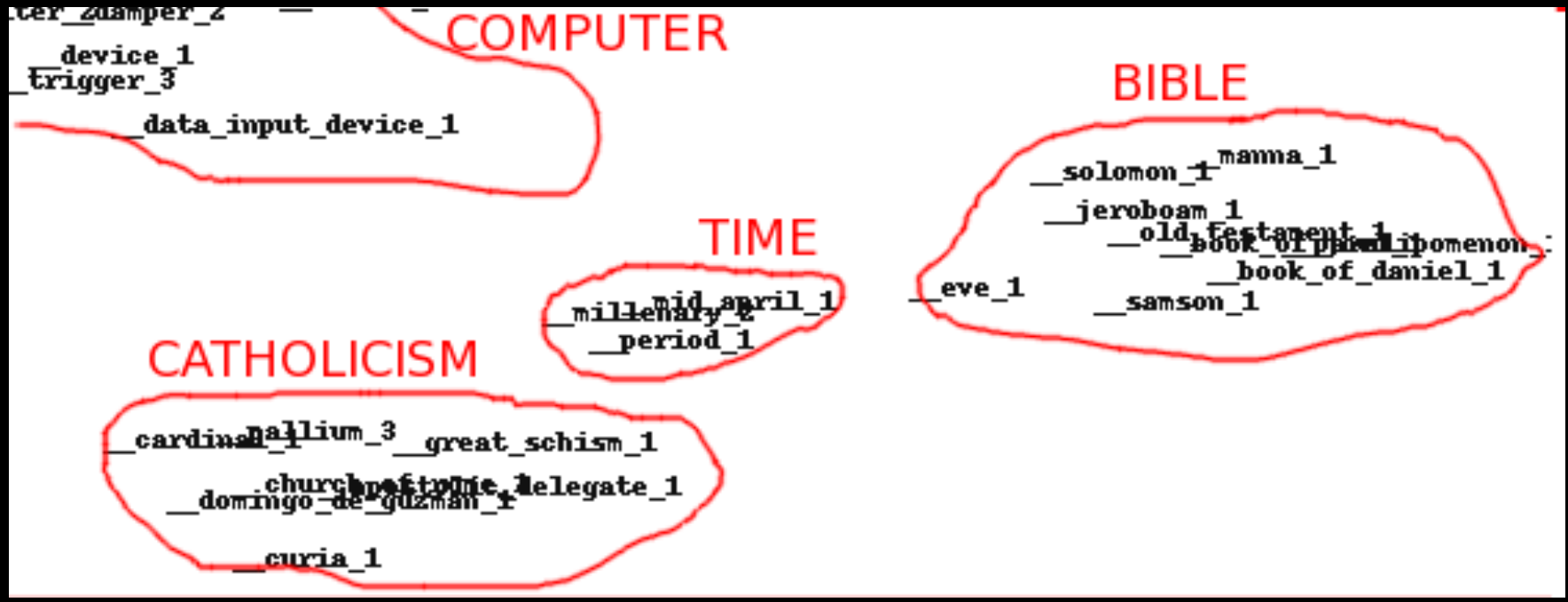


t-SNE of Embeddings: zoom 1

t-SNE of Embeddings: zoom 2



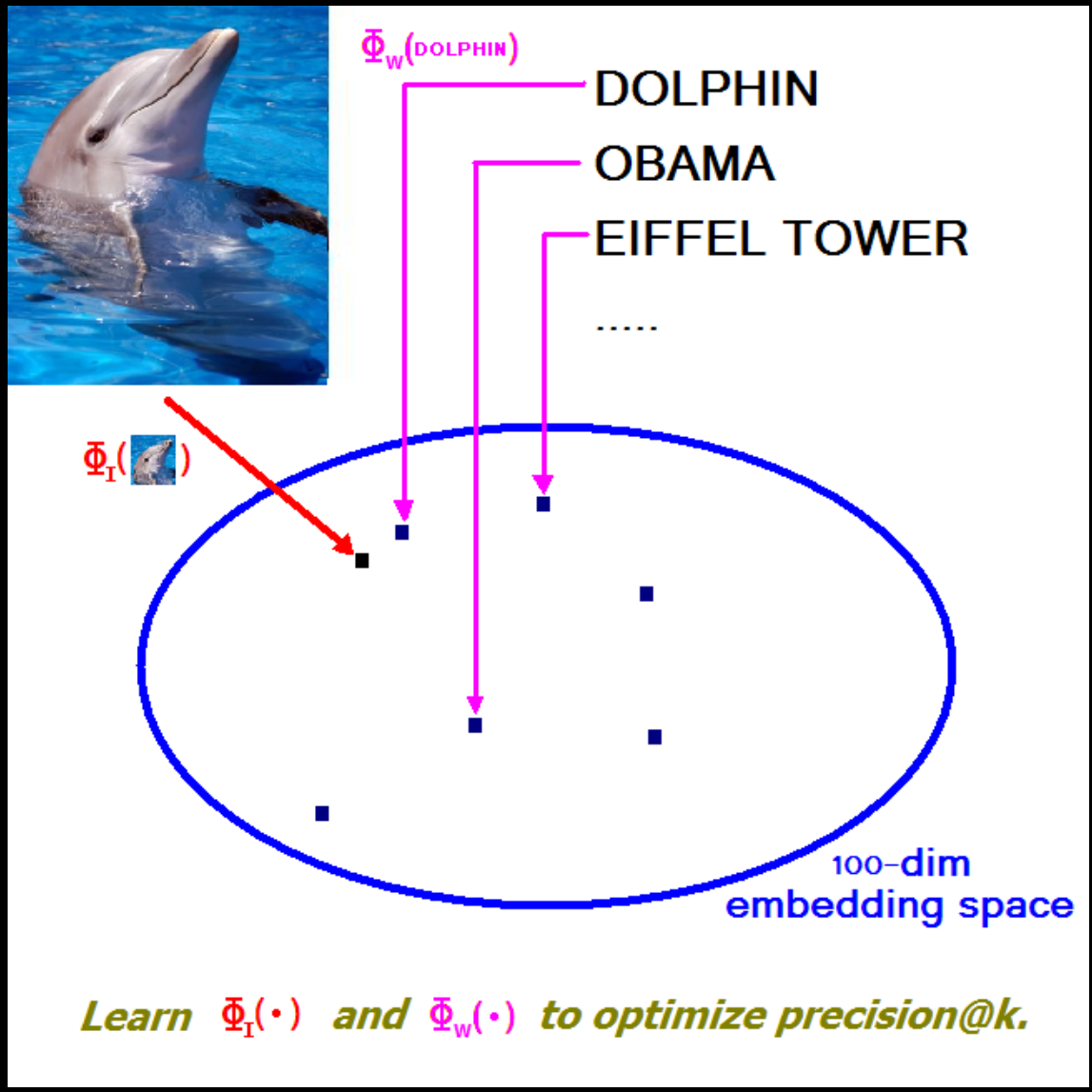
t-SNE of Embeddings: zoom 3



Joint Image-Query Embedding Space

S. Bengio, J. Weston et al @ Google

(NIPS'2010, JMLR 2010, MLJ 2010, NIPS'2009)



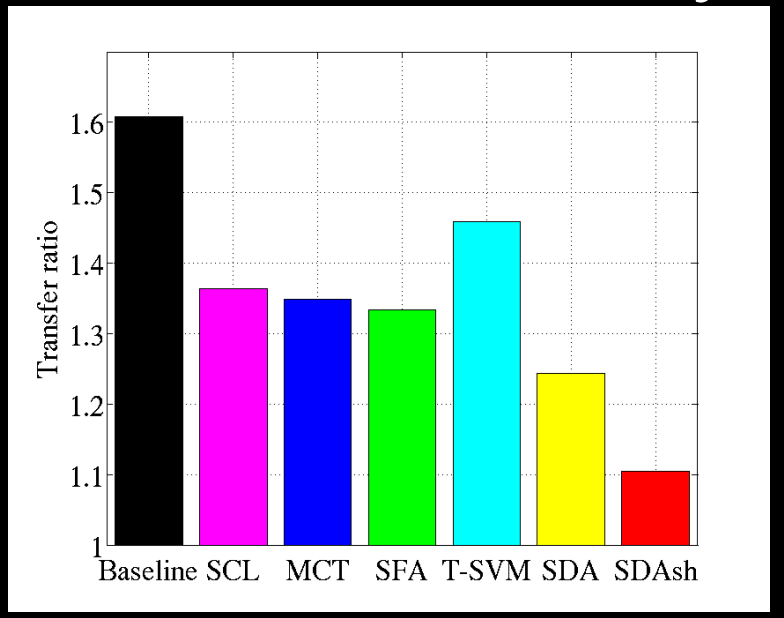
Some results with deep distributed representations for NLP

- *(Bengio et al 2001, 2003)*: beating n-grams on small datasets (Brown & APNews), but much slower
- *(Schwenk et al 2002,2004,2006)*: beating state-of-the-art large-vocabulary speech recognizer using deep & distributed NLP model, with ***real-time*** speech recognition
- *(Morin & Bengio 2005, Blitzer et al 2005, Mnih & Hinton 2007,2009)*: better & faster models through hierarchical representations
- *(Collobert & Weston 2008)*: reaching state-of-the-art in multiple NLP tasks (**SRL**, POS, NER, chunking) thanks to unsupervised pre-training and multi-task learning
- *(Bai et al 2009)*: ranking & semantic indexing (info retrieval).
- *(Collobert 2010):* Deep Learning for Efficient Discriminative Parsing
- (S. Bengio, J. Weston et al @ Google, 2009,2010,2011): joint embedding space for images and keywords, **Google image search**
- *(Sutskever & Martens 2011)*: beating SOA in text compression.
- *(Socher et al 2011)*: parsing with recursive nets, ICML 2011 distinguished application paper award
- *(Mikolov et al 2011)*: beating the SOA in perplexity with recurrence

Domain Adaptation (ICML 2011)

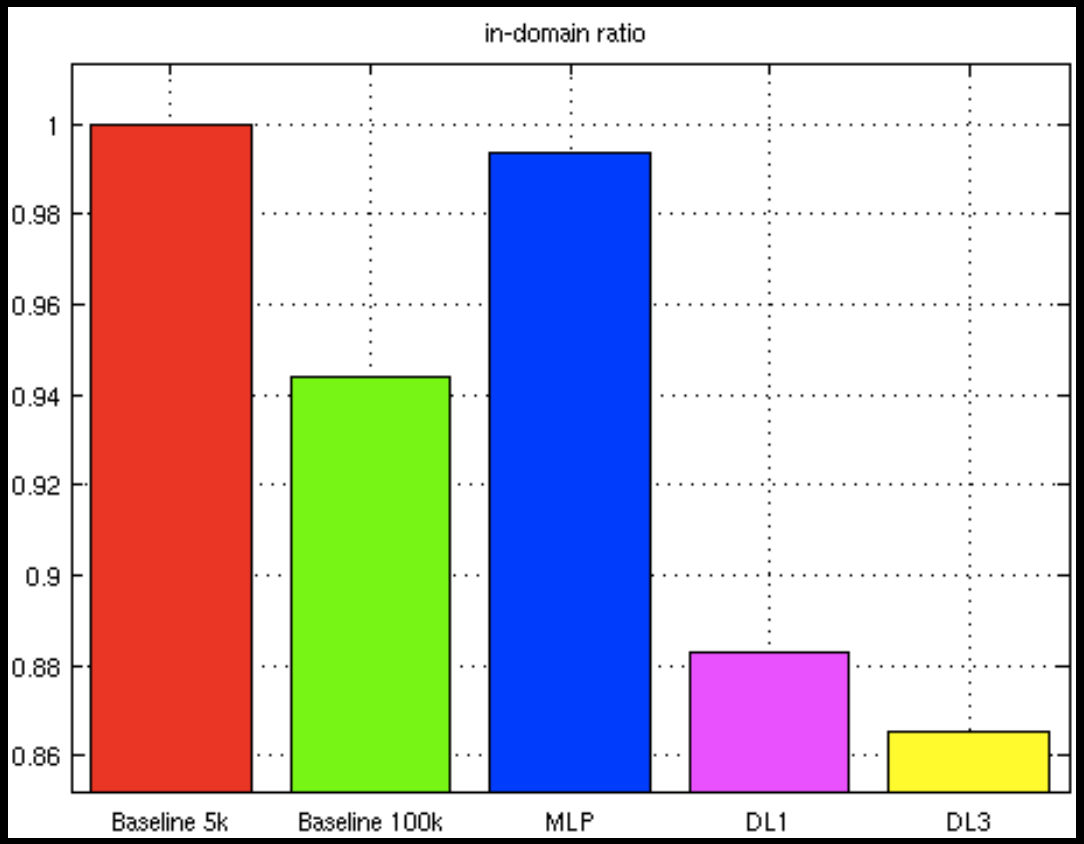
Small (4-domain) Amazon benchmark: we beat the state-of-the-art handsomely

Sparse rectifiers
Stacked Denoising Autoencoders find more features that tend to be useful either for predicting domain or sentiment, not both

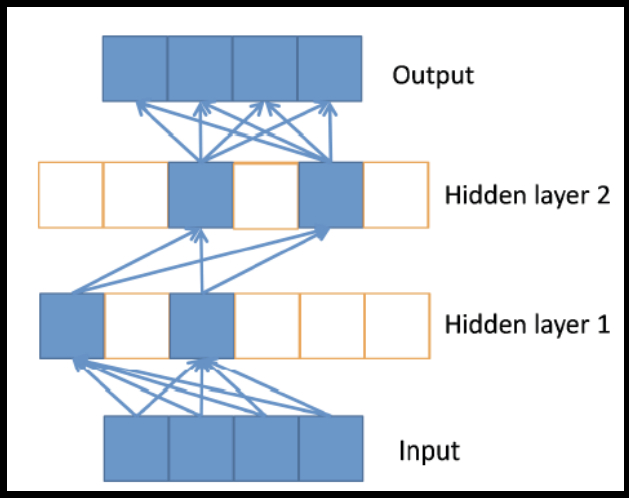


Sentiment Analysis: Transfer Learning

- 25 Amazon.com domains: toys, software, video, books, music, beauty, …
- Unsupervised pre-training of input space on all domains
- Supervised SVM on 1 domain, generalize out-of-domain
- Baseline: bag-of-words + SVM



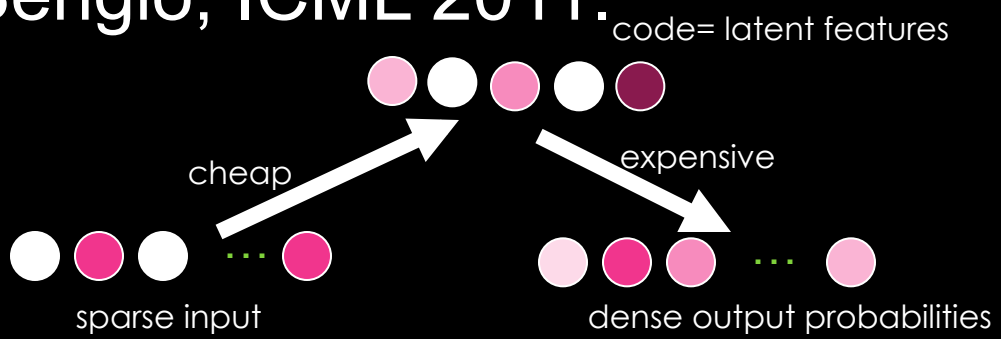
Representing Sparse High-Dimensional Stuff

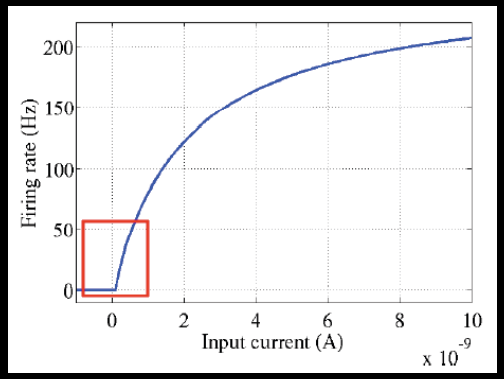


$$f(x) = max(0, x)$$

Deep Sparse Rectifier Neural Networks, Glorot, Bordes & Bengio, AISTATS 2011.

Sampled Reconstruction for Large-Scale Learning of Embeddings, Dauphin, Glorot & Bengio, ICML 2011.





Representing Sparse High-Dimensional Stuff: *Sampled Reconstruction*

$$\hat{L}(\mathbf{x}, \mathbf{z}) = \sum_{k}^{d} \frac{\hat{\mathbf{p}}_k}{\mathbf{q}_k} H(\mathbf{x}_k, \mathbf{z}_k)$$

Stochastic reweighted loss

$$\hat{\mathbf{p}} \in \{0,1\}^d \text{ with } \hat{\mathbf{p}} \sim P(\hat{\mathbf{p}}|\mathbf{x})$$

Sample which inputs to reconstruct

$$\mathbf{q}_k = E[\hat{\mathbf{p}}_k | k, \mathbf{x}, \tilde{\mathbf{x}}]$$

Importance sampling reweighting

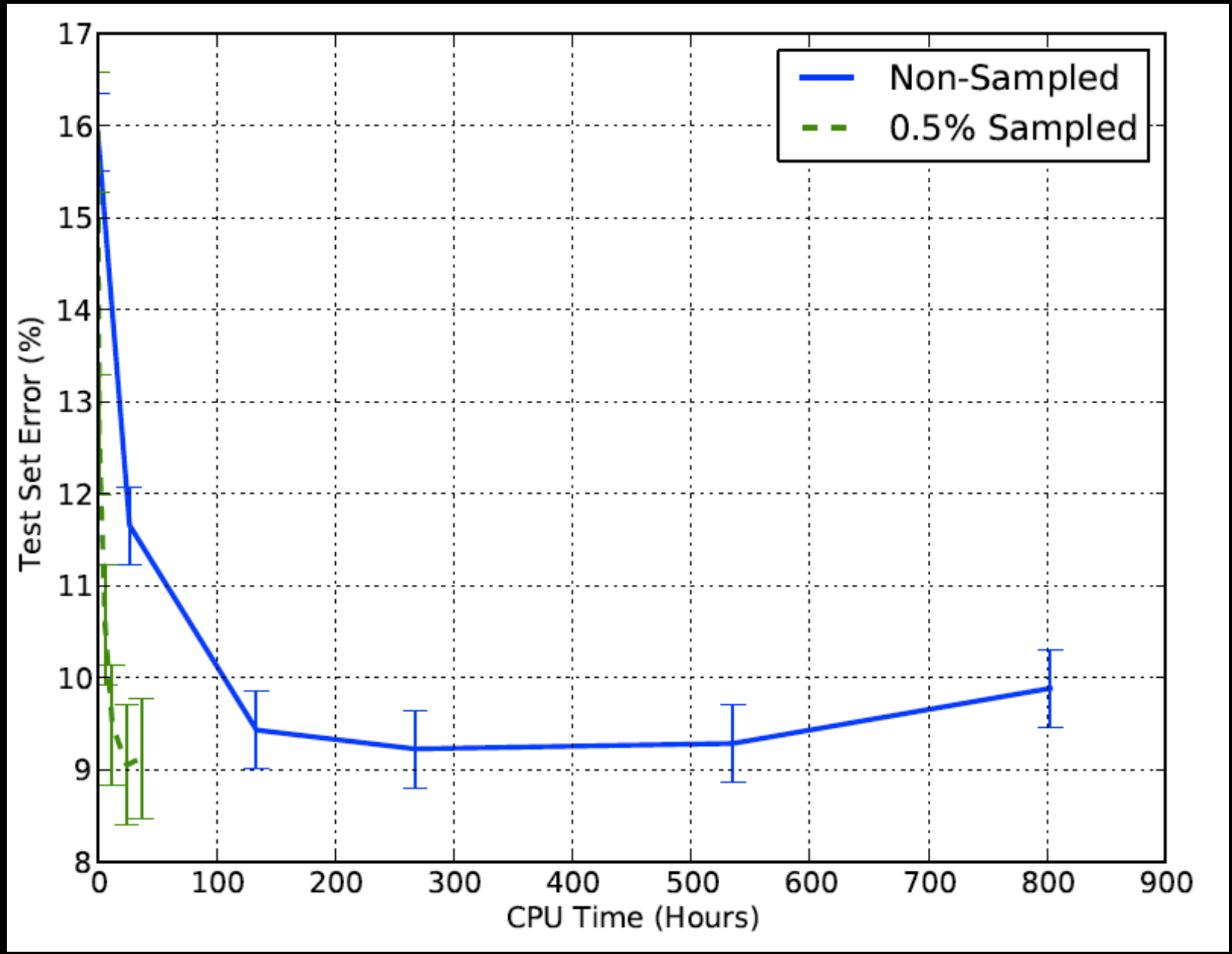
$$\text{Let } \mathcal{C}(\mathbf{x}, \tilde{\mathbf{x}}) = \{k : \mathbf{x}_k = 1 \text{ or } \tilde{\mathbf{x}}_k = 1\}$$

Minimum-variance: guess wrong reconstructions

$$P(\hat{\mathbf{p}}_k = 1 | \mathbf{x}_k) = \begin{cases} 1 & \text{if } k \in \mathcal{C}(\mathbf{x}, \tilde{\mathbf{x}}) \\ |\mathcal{C}(\mathbf{x}, \tilde{\mathbf{x}})|/d_x & otherwise \end{cases}$$

As many randomly chosen other bits

Speedup from *Sampled Reconstruction*

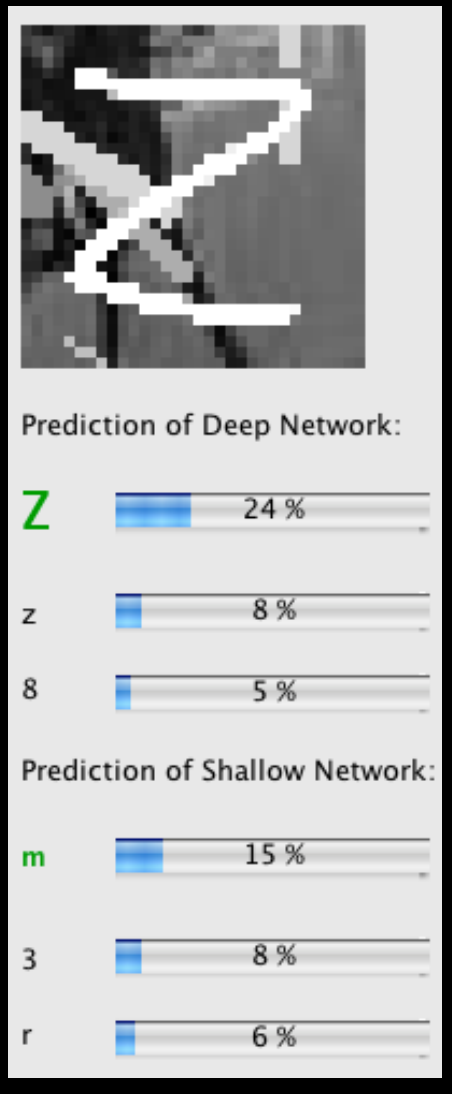


Deep Self-Taught Learning for Handwritten Character Recognition

Y. Bengio & 16 others

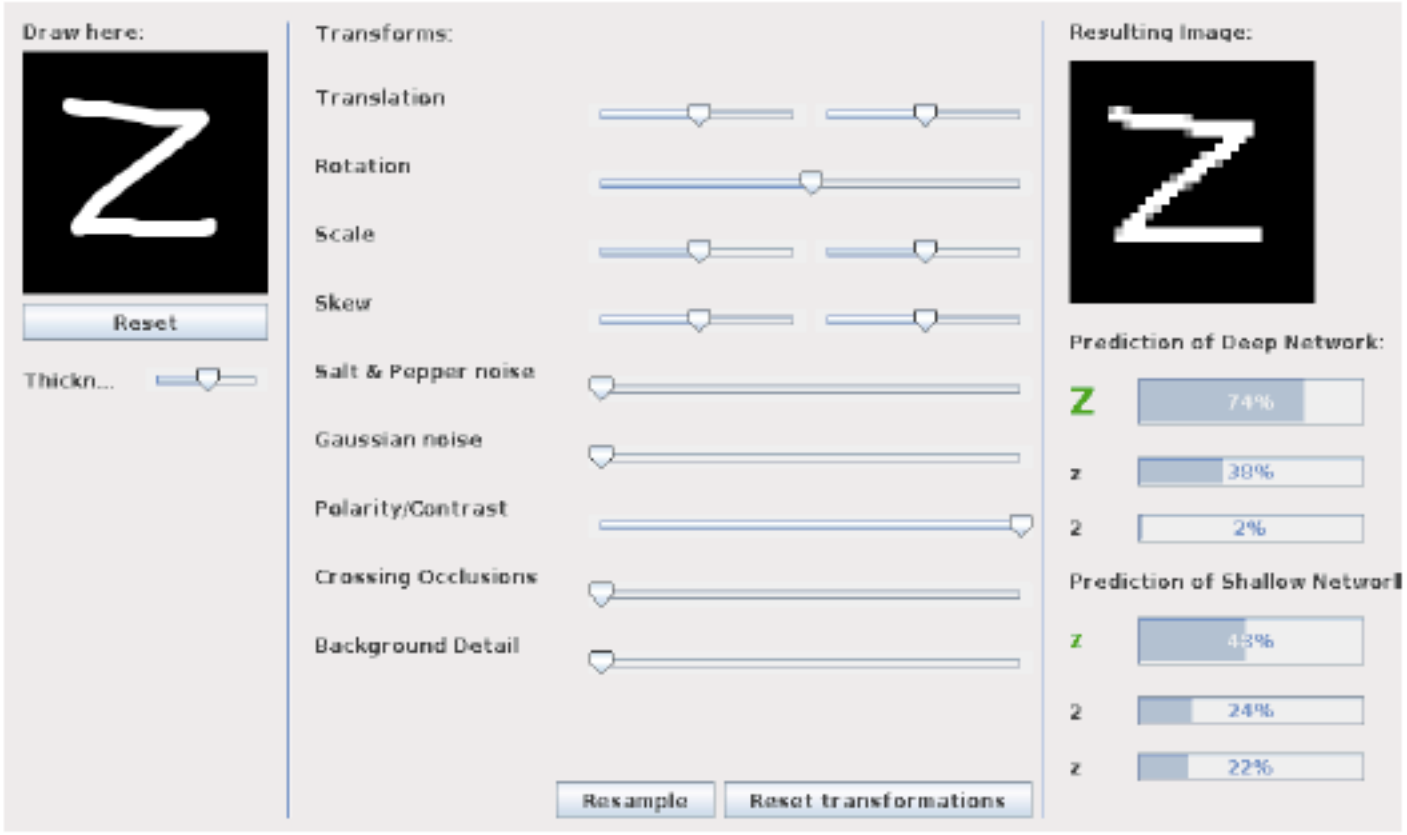
(IFT6266 class project & AISTATS 2011 paper)

- discriminate 62 character classes (upper, lower, digits), 800k to 80M examples
- Deep learners beat state-of-the-art on NIST and reach human-level performance
- Deep learners benefit more from perturbed (out-of-distribution) data
- Deep learners benefit more from multi-task setting



Online Java applet demo

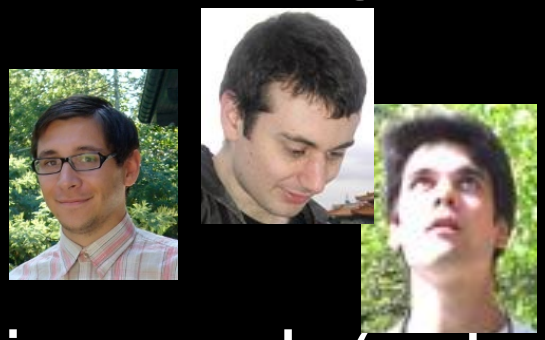
Draw a character, see instant classification. Runs client-side.



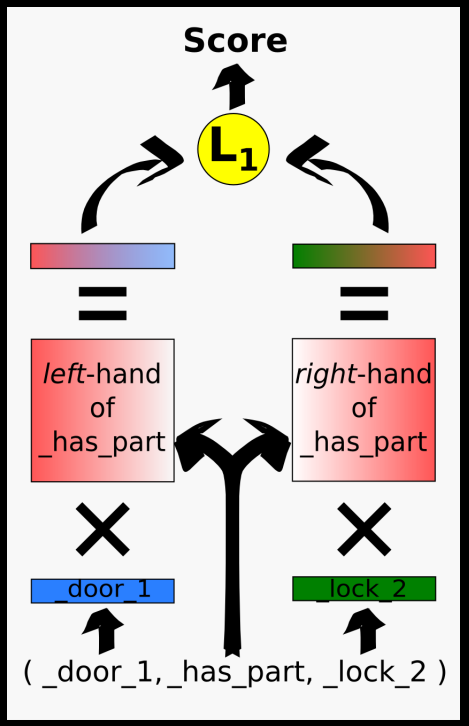
http://deep.host22.com

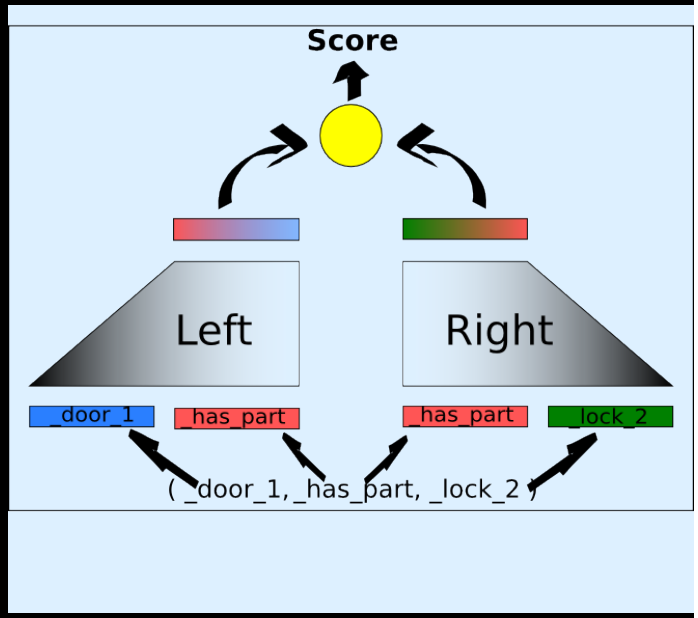
Modeling Semantics

Learning Structured Embeddings of Knowledge Bases, Bordes, Weston, Collobert & Bengio, AAAI 2011

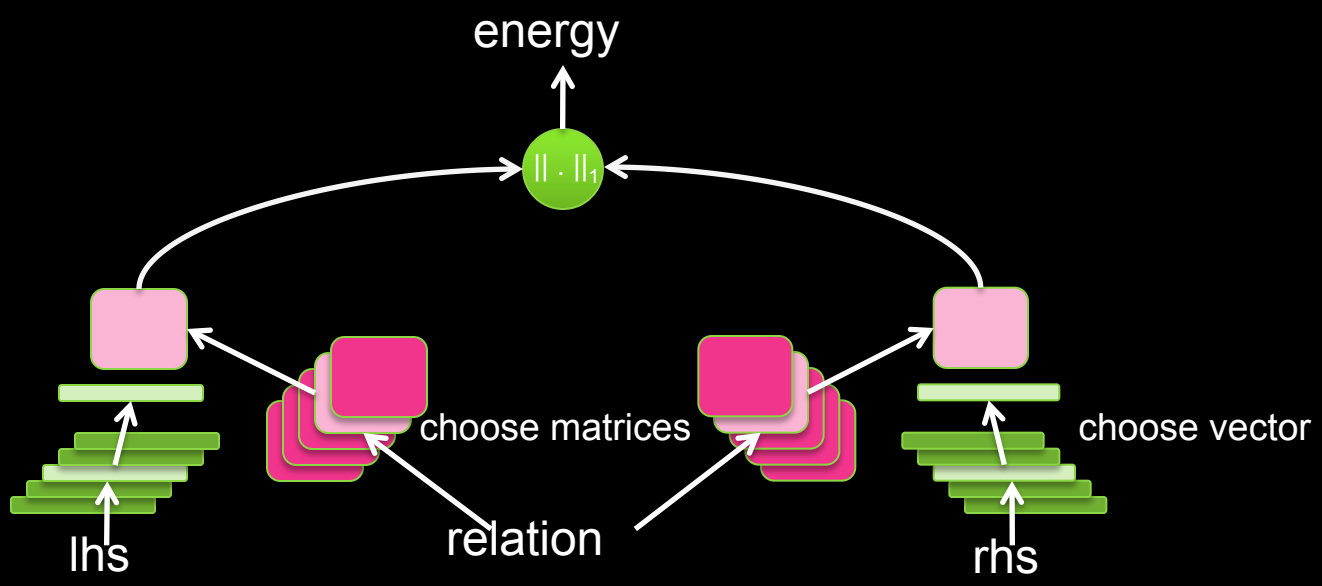


+ ongoing work (submitted)





Modeling Relations with Matrices



Model (lhs, relation, rhs)

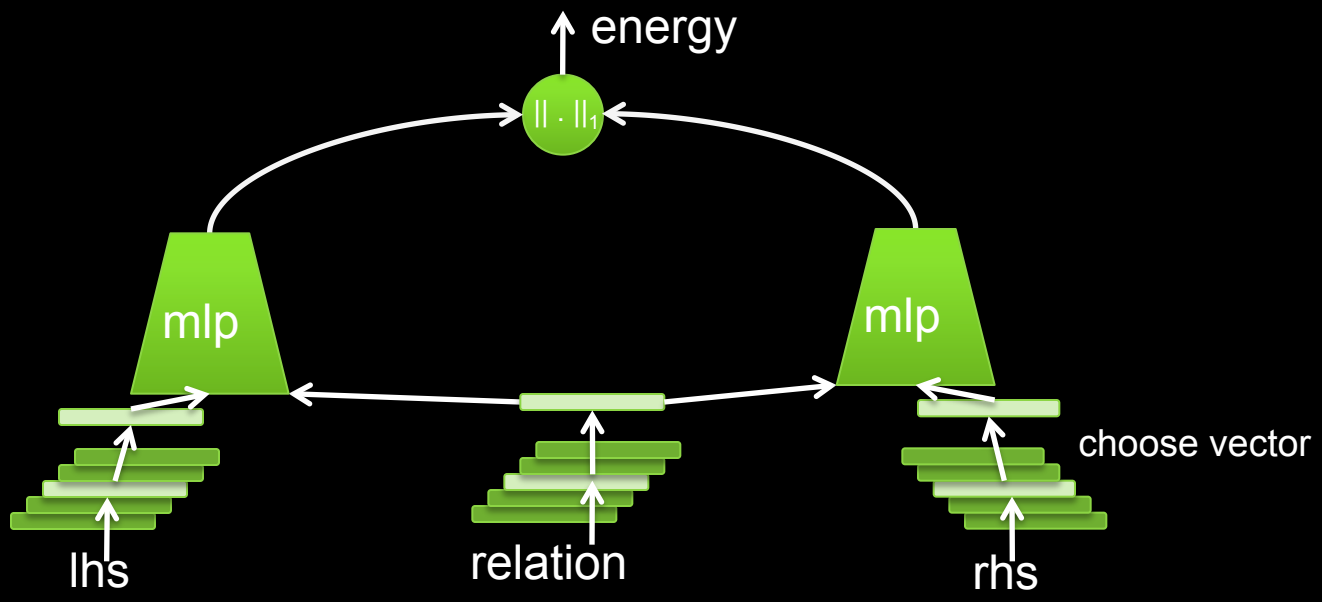
Each concept = 1 embedding vector

Each relation = 2 matrices

Ranking criterion

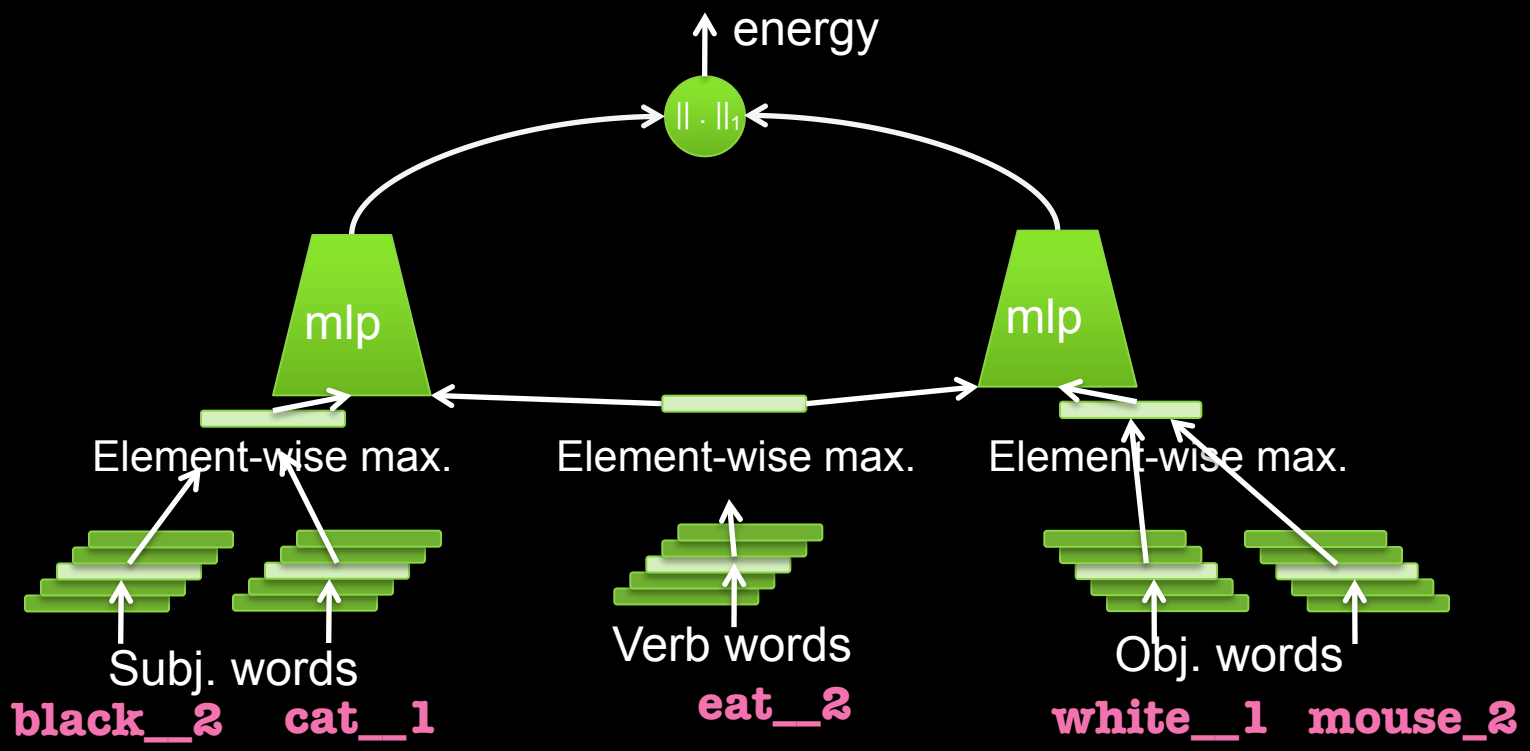
Energy = ….

Allowing Relations on Relations



Verb = relation. Too many to have a matrix each.
Each concept = 1 embedding vector
Each relation = 1 embedding vector
Can handle relations on relations on relations

Training on Full Sentences



→ Use SENNA (Collobert 2010) = embedding-based NLP tagger for Semantic Role Labeling, breaks sentence into (subject part, verb part, object part)

→ Use max-pooling to aggregate embeddings of words inside each part

Combining Multiple Sources of Evidence with Shared Embeddings

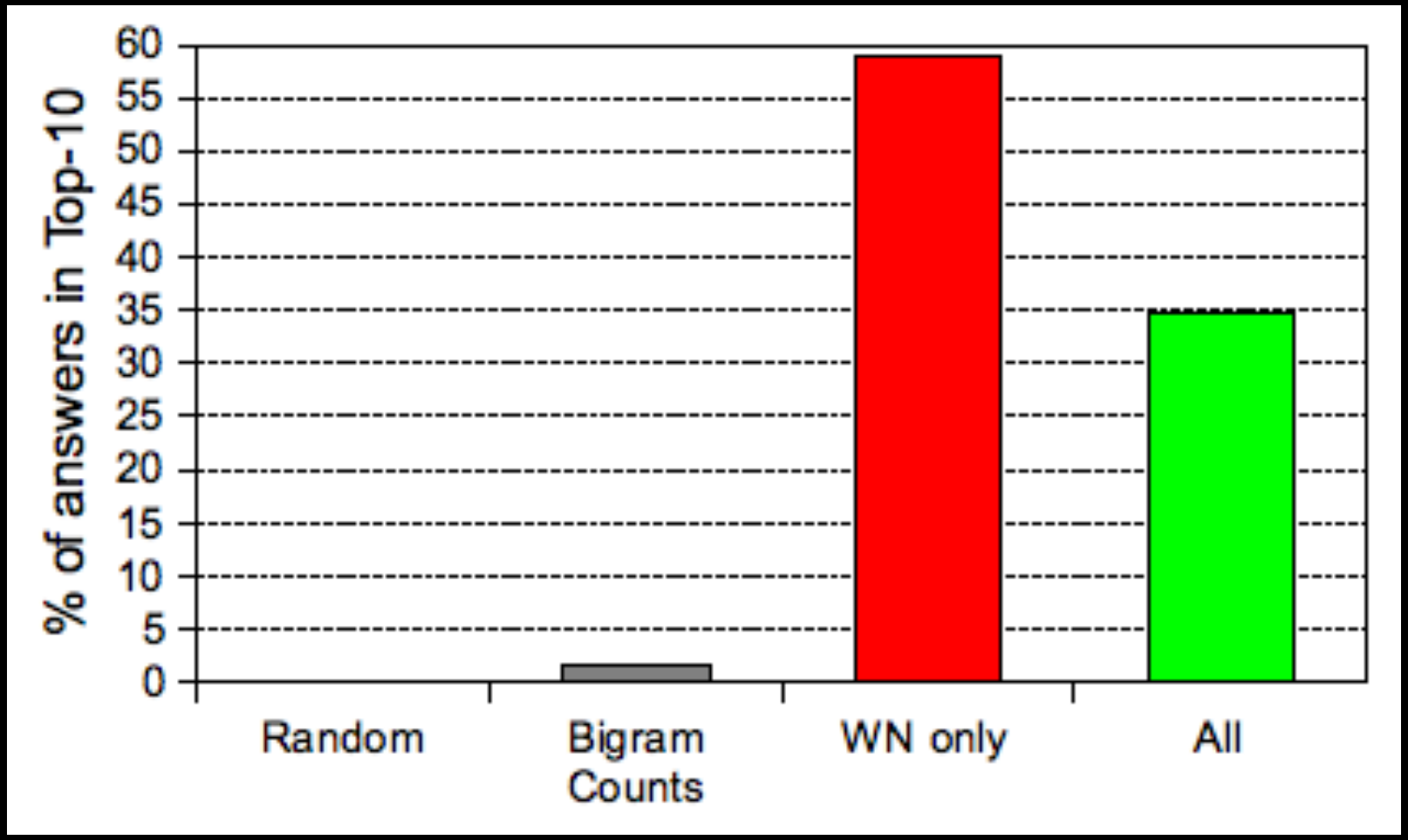
- The undirected graphical model version of relational learning
- With embeddings (shared representations) to help propagate information among data sources: here WordNet, XWN, Wikipedia, FreeBase,…
- Different energy functions can be used for different types of relations, or a generic representation and generic relation symbols used for everything

Question Answering

Generalizing WordNet or **Freebase**, exploiting Wikipedia

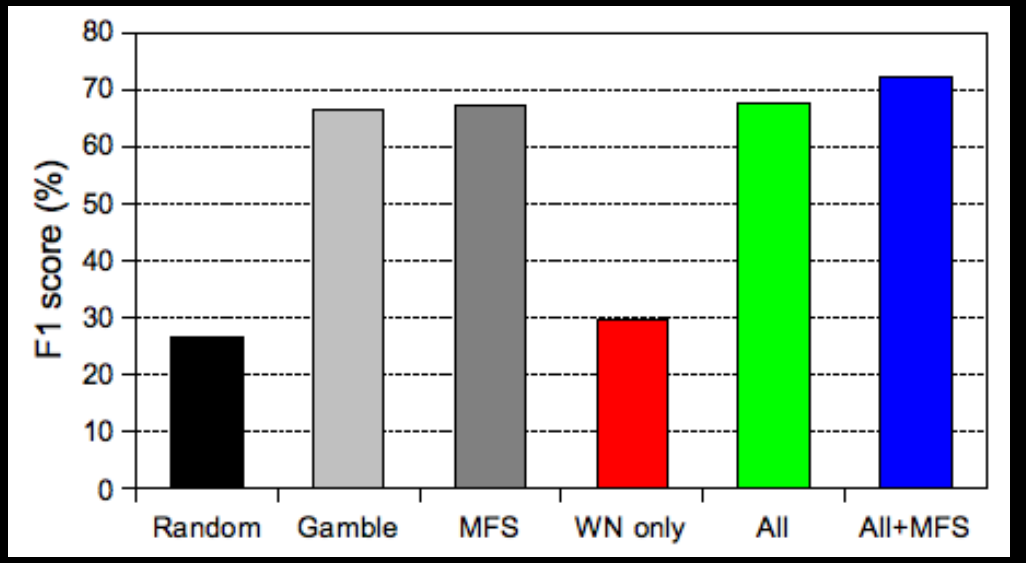
	Model (All)	TextRunner
lhs	_army_NN_1	army
rel	_attack_VB_1	attacked
top ranked rhs	_troop_NN_4 _armed_service_NN_1 _ship_NN_1 _territory_NN_1 _military_unit_NN_1	Israel the village another army the city the fort
top ranked lhs	_business_firm_NN_1 _person_NN_1 _family_NN_1 _payoff_NN_3 _card_game_NN_1	People Players one Students business
rel	_earn_VB_1	earn
rhs	_money_NN_1	money

Question Answering: Ranking Score



Word Sense Disambiguation

- Senseval-3 results



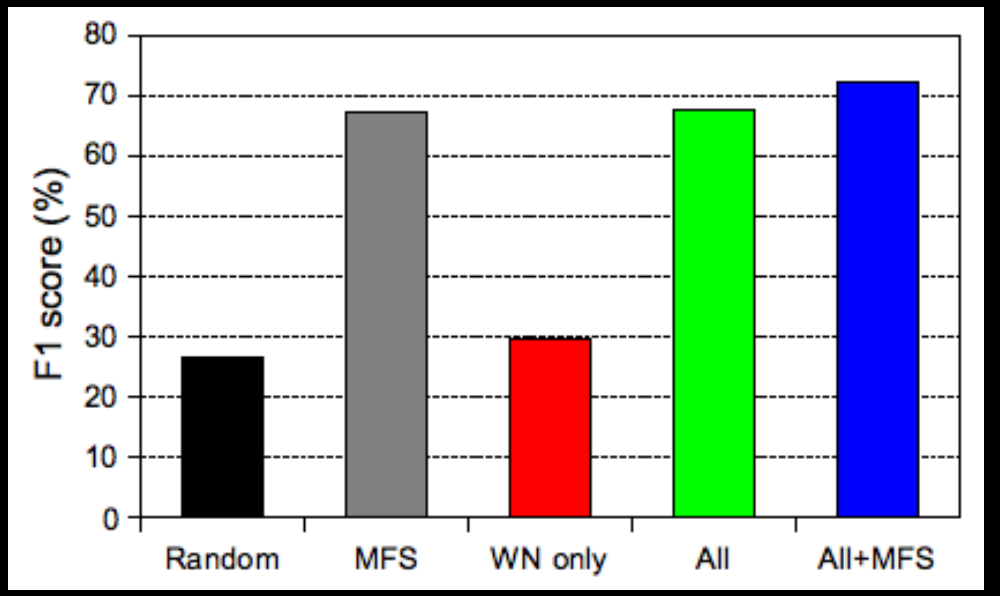
MFS=most frequent sense

All=training from all sources

Gamble=Decadt et al 2004 (Senseval-3 SOA)

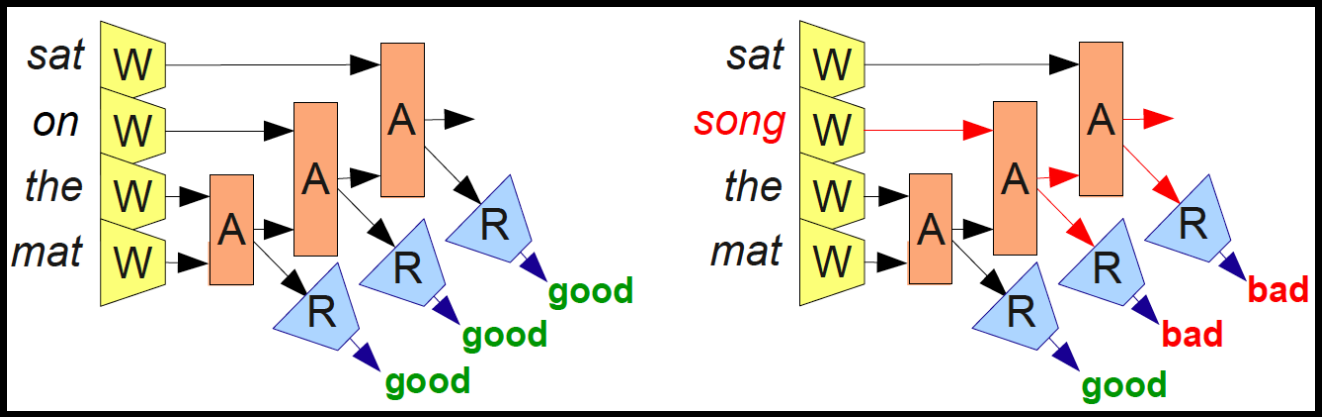
- XWN results

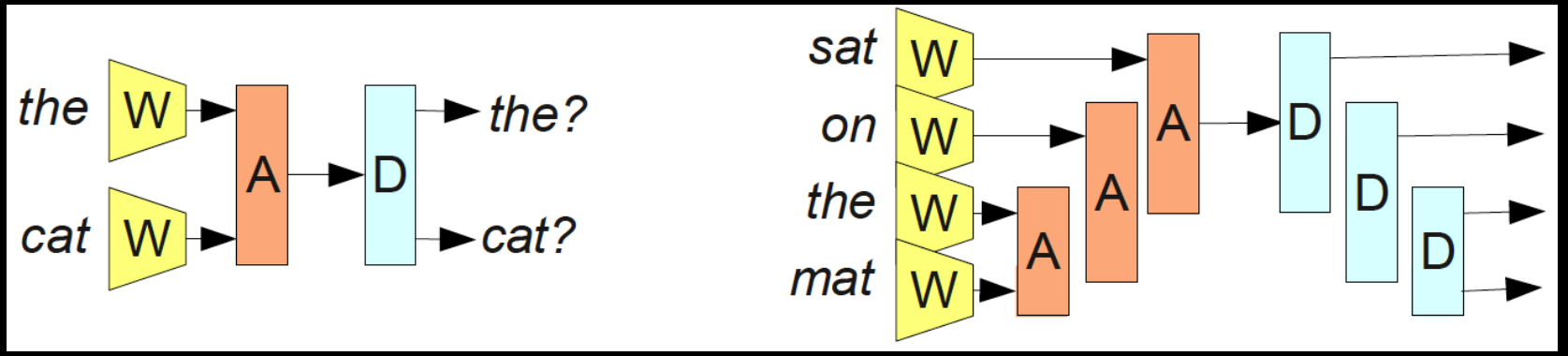
XWN = eXtended WN



Recursive Application of Relational Operators

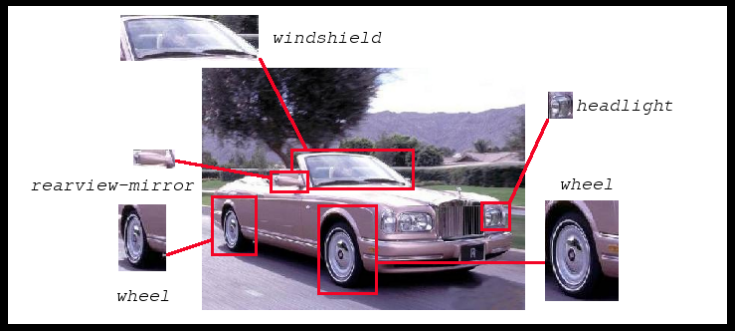
Bottou 2011: ‘From machine learning to machine reasoning’, also Socher ICML2011.





Relations on Multiple Data Types

- Add energy terms associated to relations from different data sources, shared embeddings



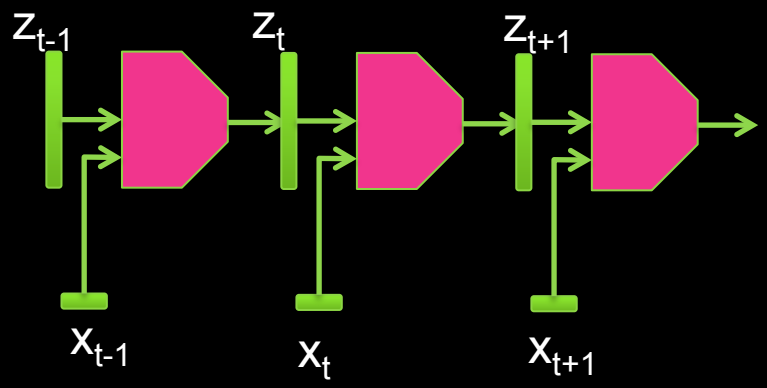
energy(object image, is-a, object label) + energy(part image, is-a, part label) + energy(part image, image-part-of, object image) + energy(part label, label-part-of, object label)

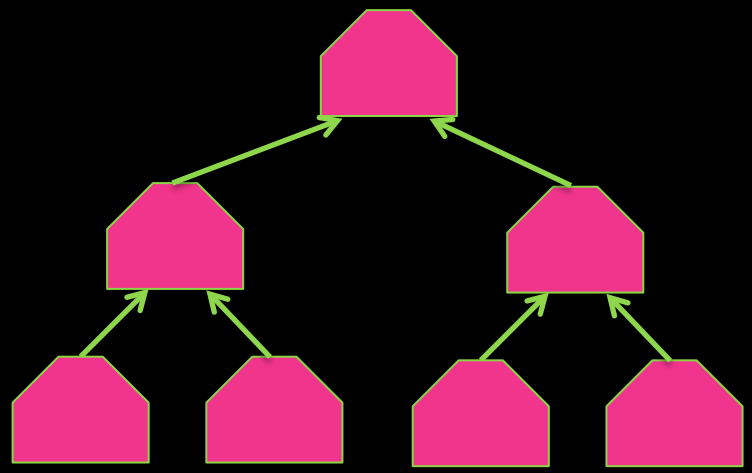
Table 1: **Summary of Test Set Results on ImageNet-WordNet.** Precision at 1 and 10, and Mean Average Precision (MAP) are given. (IW) resp. (I) refers to the (Image,Word) setup resp. (Image).

	Image Annotation			Part-Object Detection			Triplet		
Models	p@1	p@10	MAP	p@1	p@10	MAP	p@1	p@10	MAP
Shared (IW)	9.14%	3.51%	0.1768	**11.48%**	**3.40%**	**0.1892**	26.31%	**9.90%**	0.5545
UnShared (IW)	9.45%	3.68%	0.1847	10.01%	3.02%	0.1669	**33.13%**	9.62%	**0.5595**
Shared (I)	11.21%	3.85%	0.2021	5.13%	1.84 %	0.0955	11.21%	3.85%	0.2021
UnShared (I)	**12.94%**	**4.10%**	**0.2219**	6.08%	2.11%	0.1118	12.94%	4.10%	0.2219
SVM	10.02%	3.72%	0.1864	–	–	–	10.02%	3.72%	0.1864

Recurrent and Recursive Nets

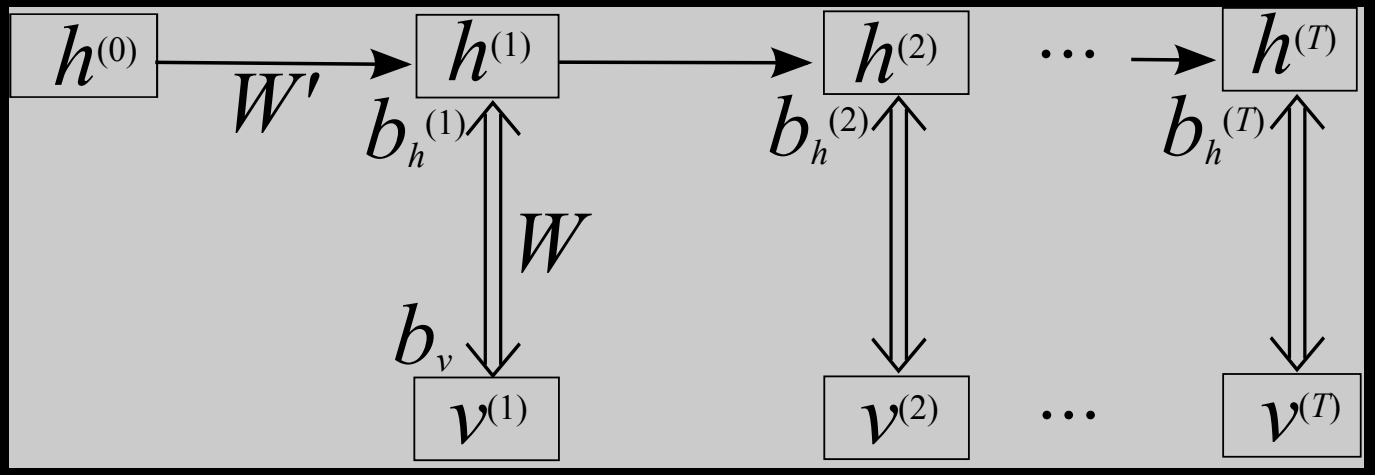
- Replicate a parametrized function over different time steps or nodes of a DAG
- Output state at one time-step / node is used as input for another time-step / node
- Very deep once unfolded!





Combining RBMs and Temporal Recurrence

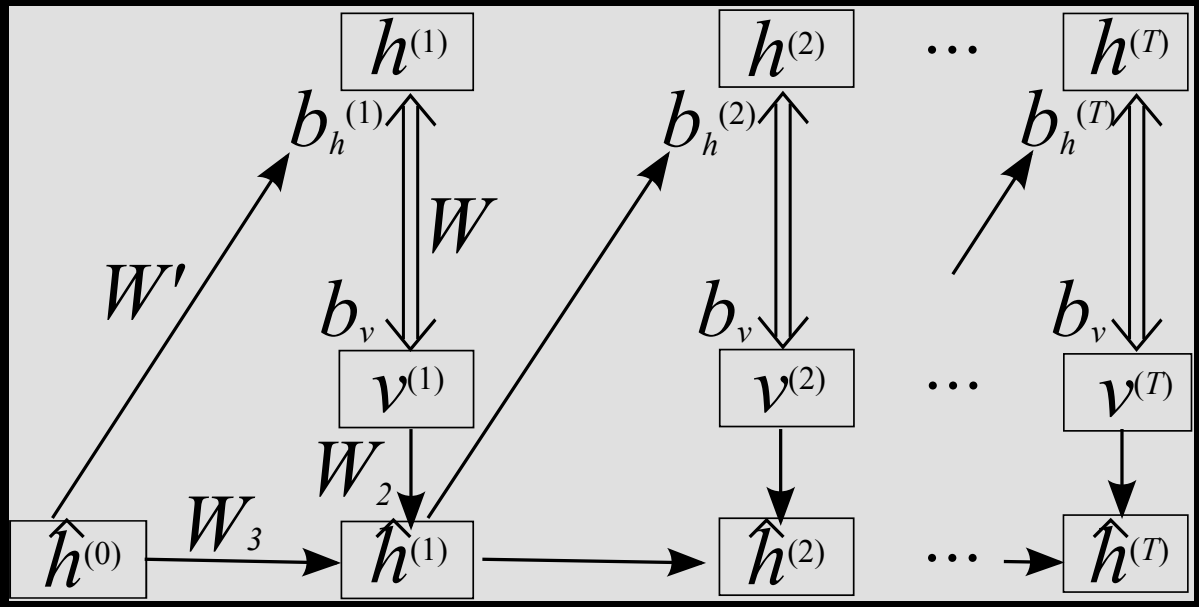
- RTRBM (Sutskever, Hinton & Taylor, NIPS 2008)
- One RBM per time step, modeling visible at t
- Hidden units biases are function of mean-field of previous hidden units, thus introducing a recurrence.



RNN-RBM

(Boulanger, Vincent & Bengio)

Expanding parametrization of RTRBM to remove constraint that recurrent weights correspond to link from recurrent state to RBM hiddens and that visible-hidden weights correspond to recurrent net input weights.

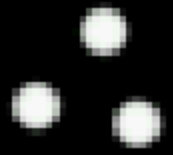


$$b_h^{(t)} = b_h + W'\hat{h}^{(t-1)}$$

$$\hat{h}^{(t)} = \sigma(Wv^{(t)} + b_h^{(t)}) = \sigma(Wv^{(t)} + W'\hat{h}^{(t-1)} + b_h)$$

Experiments with RNN-RBM

- Bouncing balls dataset: RNN-RBM is twice more accurate (MSE 0.005) than RTRBM (MSE 0.01) at predicting next ball position



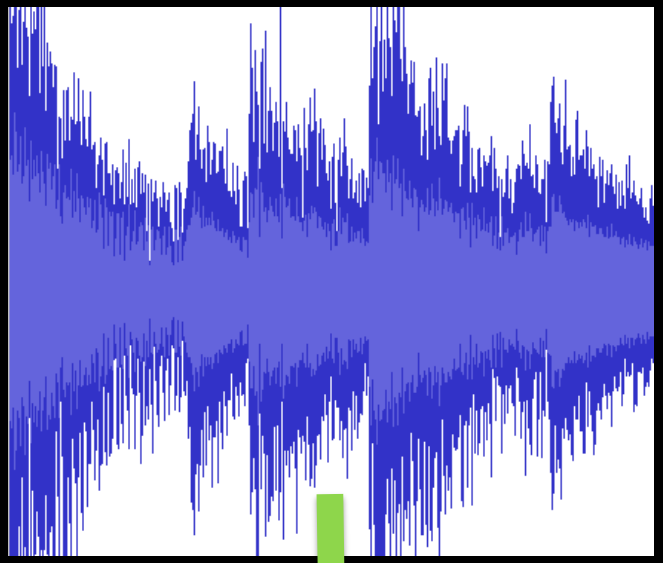
Training example

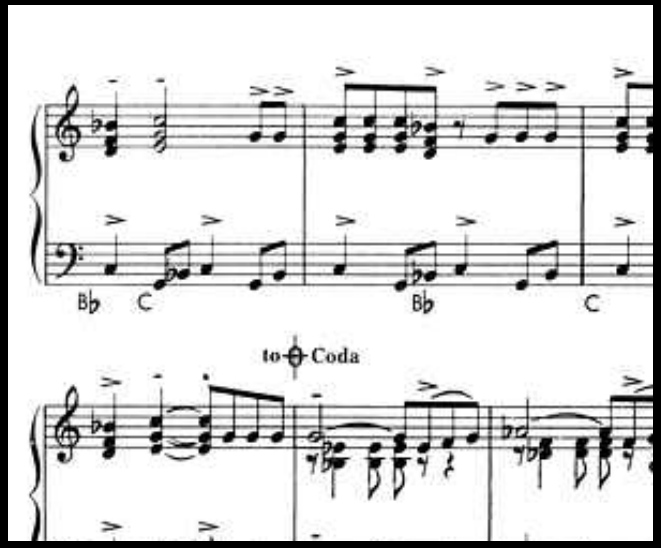
RTRBM-generated sample

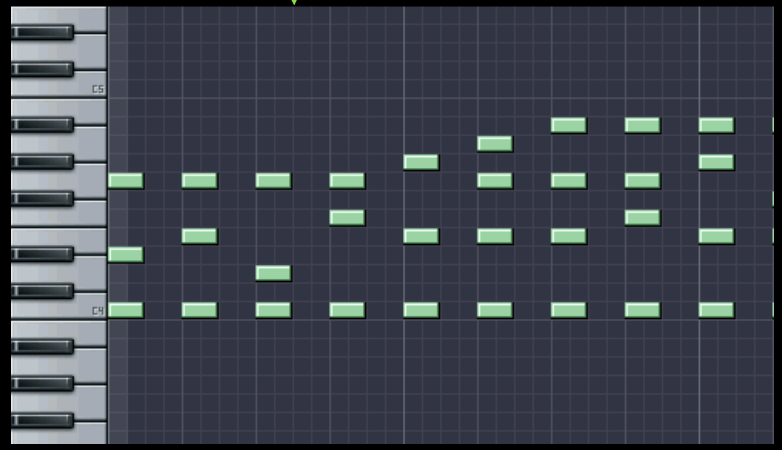
- Motion capture dataset: RNN-RBM (MSE 0.33) vs RTRBM (MSE 0.41)

Music Transcription

Map acoustic signal to sequence of chords (note tuples over time intervals)







Acoustic Models Comparison Pitch Detection

MÉTHODE	FIDÉLITÉ	RAPPEL	PRÉCISION
SVM [36]	38.5 %	93.6 %	38.5 %
Yeh & Röbel [33]	66.3 %	67.6 %	86.0 %
NNC	62.5 %	67.4 %	84.0 %
NNSC	70.6 %	74.6 %	89.1 %
MLP3	71.8 %	77.1 %	88.5 %
DBN	77.5 %	80.2 %	93.2 %

Data from 6 SoundFont 2.0 banks, used for training/validation/test

RNN-RBMs for Music Transcription

- up to 5 notes at a time
- Combine pitch detection + music language model with RNN-RBM with a product of expert model: $\log P_{lang}(\text{notes sequence}) + a * \log P_{audio}(\text{notes}|\text{acoustic})$
- RNN-RBM predicts next chord given previous ones
- Improves state-of-the-art (Mozer dataset) from 68.6% to 77.8% accuracy

Acoustic Transcription: Comparison

	MODÈLE	FIDÉLITÉ	RAPPEL	PRÉCISION
NNSC	Uniforme	70.6 %	74.6 %	89.1 %
	Unigramme	71.6 %	74.2 %	91.7 %
	RNN	72.1 %	74.6 %	92.3 %
	RBM	73.4 %	75.9 %	92.6 %
	RNN-RBM	75.8 %	77.5 %	94.5 %
DBN	Uniforme	77.5 %	80.2 %	93.2 %
	Unigramme	77.9 %	79.4 %	95.0 %
	RNN	78.5 %	79.9 %	95.6 %
	RBM	79.9 %	81.2 %	95.9 %
	RNN-RBM	81.9 %	83.0 %	96.5 %

Generating Music with RNN-RBM

RBM (no temporal modeling)

RNN (no joint model of notes = chords)

RNN-RBM

RNN-RBM, sample with repetitions

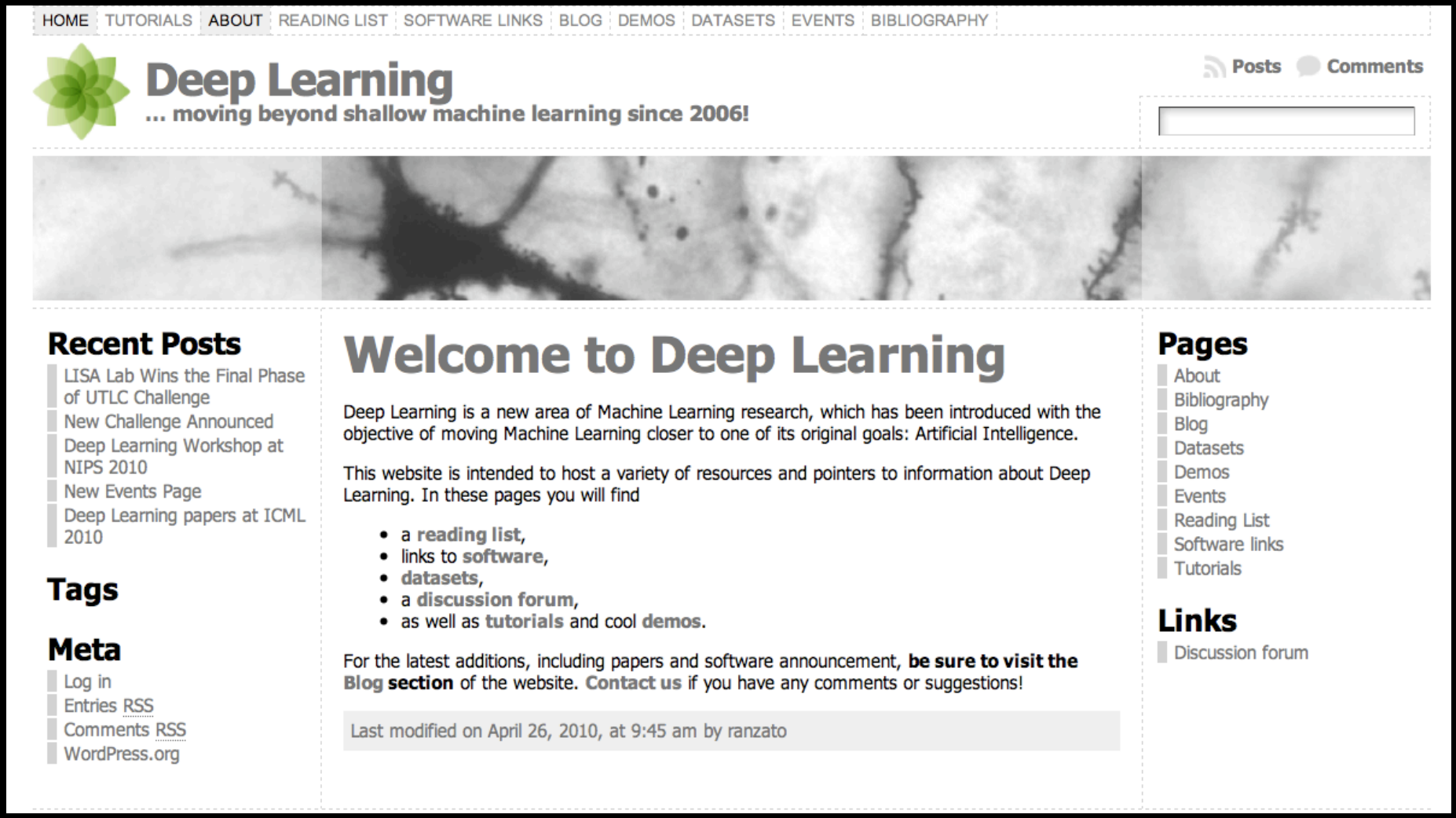
Conclusions

- Deep Learning: powerful arguments & generalization principles
- Unsupervised Feature Learning is crucial: many new algorithms and applications in recent years
- DL particularly suited for multi-task learning, transfer learning, domain adaptation, self-taught learning, and semi-supervised learning with few labels

Deep Questions

- Generic learning algorithms (large-spectrum priors) vs e.g. vision-specific or language-specific architectures?
- Try to filter the noise out vs keep all the information but separate the explanatory factors?
- Why are RBMs and various sparse (auto-en) coding disentangling to some extent?
- What criteria to disentangle the factors of variation?
- Why is sparsity working and helping the disentangling?
- How to avoid the partition fn? Strengths and weaknesses of existing proxys for likelihood?
- How to represent and train recursive / relational learners?
- How could the brain possibly do the equivalent of back-prop through time?
- Are 2^{nd}-order optimization methods really needed when N is large? (e.g. Polyak averaging converges as quickly asymptotically)

http://deeplearning.net



http://deeplearning.net/software/theano : numpy → GPU

Merci! Questions?

LISA team: