CONVOLUTIONAL DEEP BELIEF NETWORKS

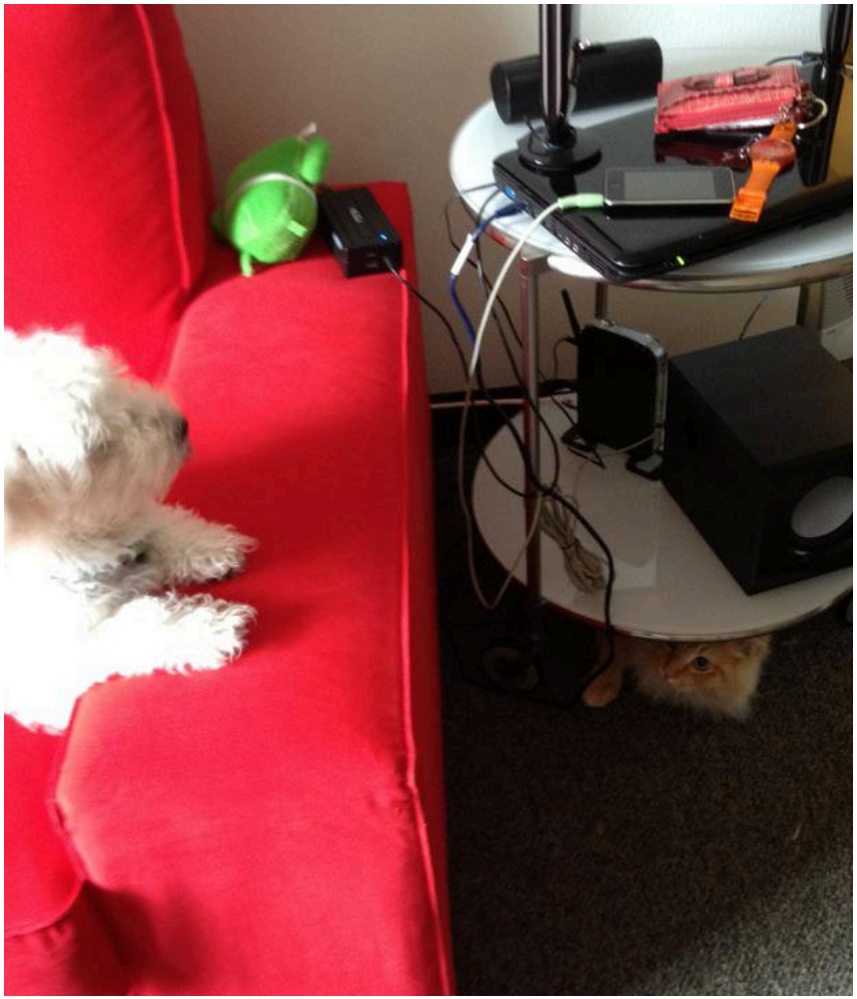
Talk by Emanuele Coviello

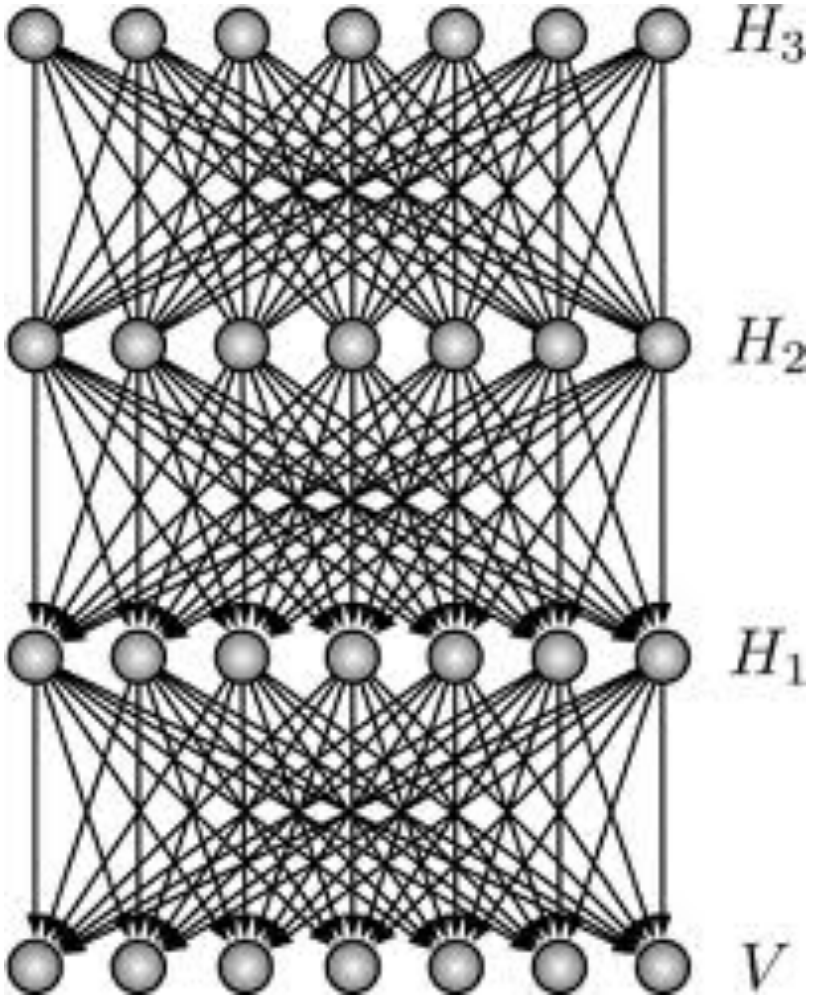
Convolutional Deep Belief Networks
for Scalable Unsupervised Learning of Hierarchical Representations

---

Honglak Lee HLLEE@CS.STANFORD.EDU
Roger Grosse RGROSSE@CS.STANFORD.EDU
Rajesh Ranganath RAJESHR@CS.STANFORD.EDU
Andrew Y. Ng ANG@CS.STANFORD.EDU
Computer Science Department, Stanford University, Stanford, CA 94305, USA

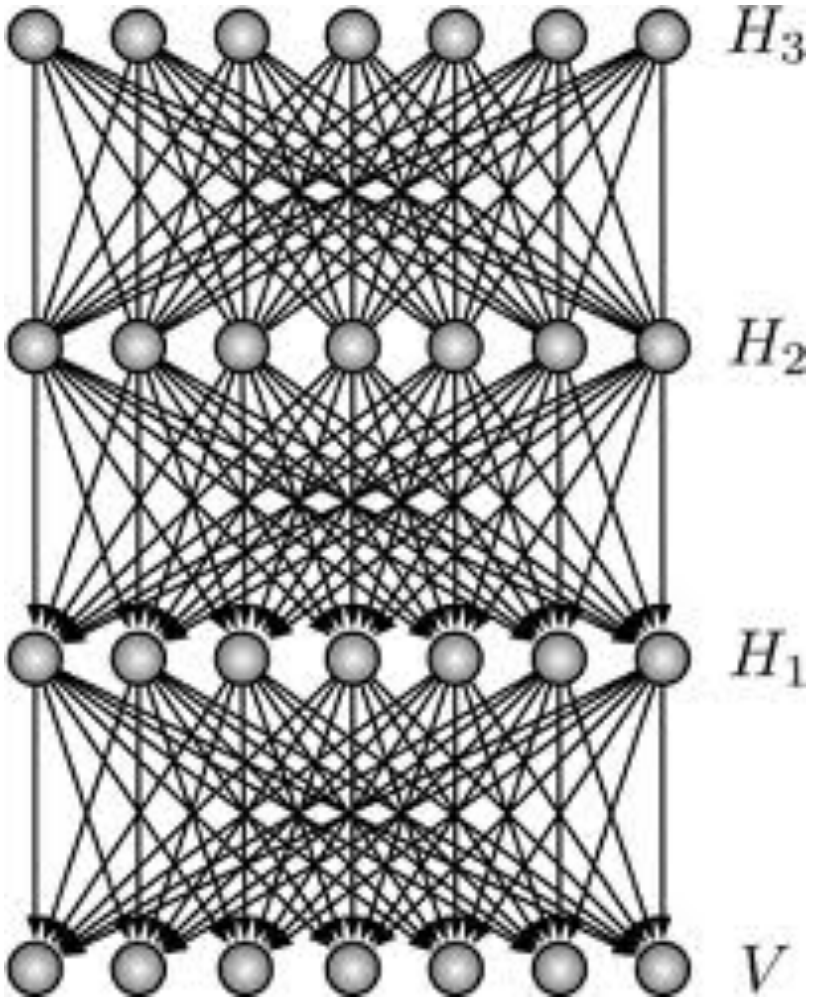
Motivation



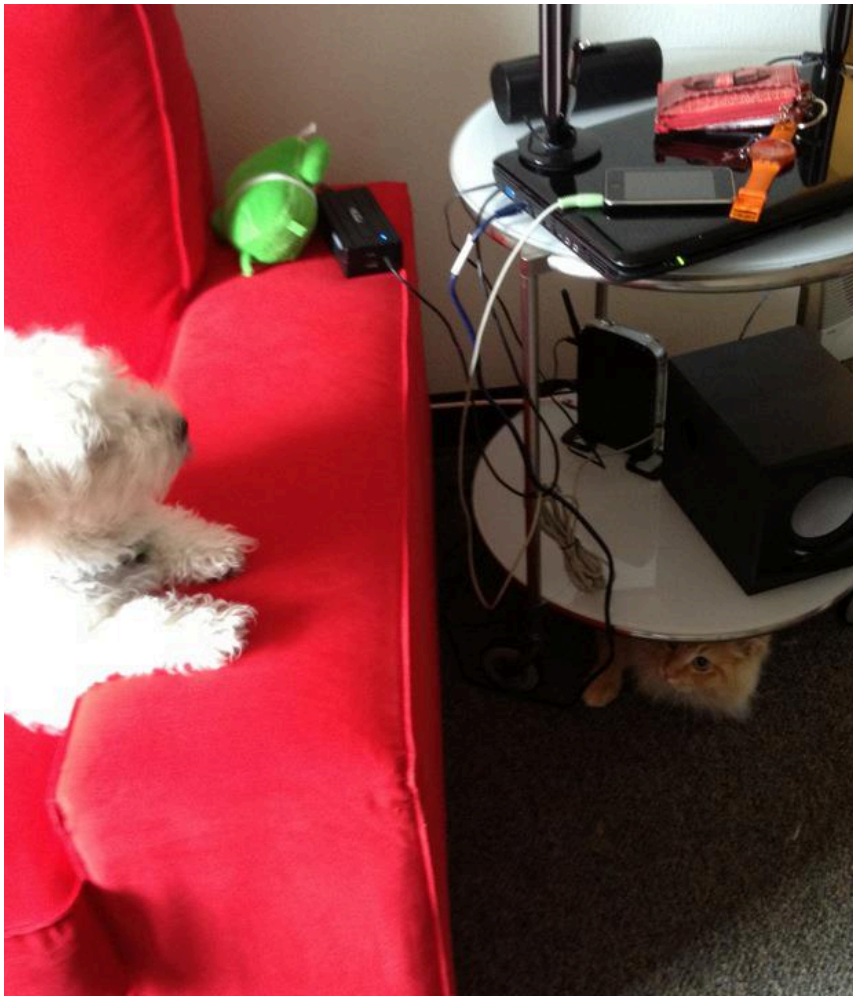


Motivation

- Deep belief Networks (DBM)
- Controlled experiments
  - Hand written digits
- What about **real** images?



Motivation



- High dimensional
- Translation invariant

- Need convolutional deep belief networks

Outline

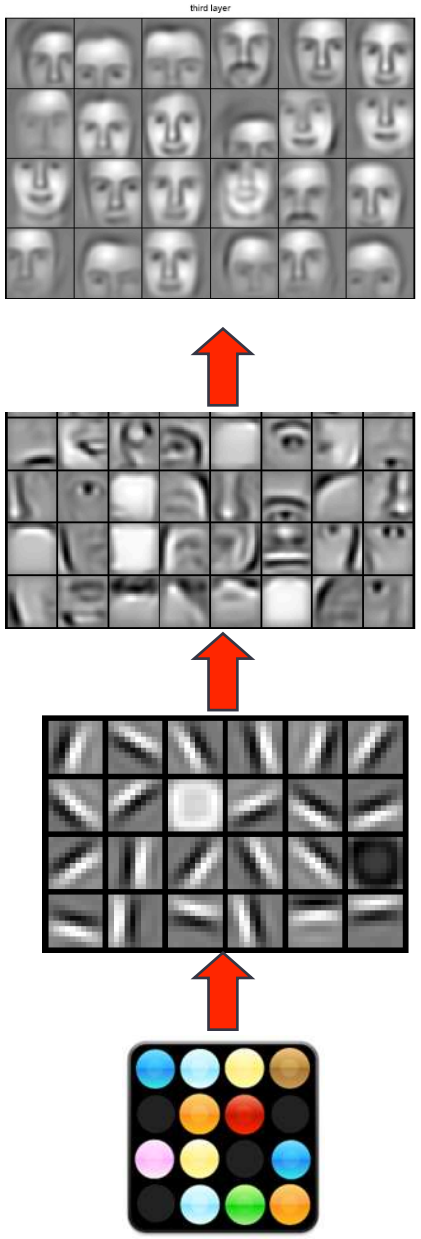
- Freshen up what we know
  - Restricted Boltzman Machines (RBMs)
  - And Deep Belief Networks (DBNs)
- New Machinery
  - Convolutional RBM
  - Probabilistic max-pooling
  - Sparsity regularization
- Convolutional DBNs (CDBM)
- Applications: image and audio

FRESHEN UP

RBMs and DBNs

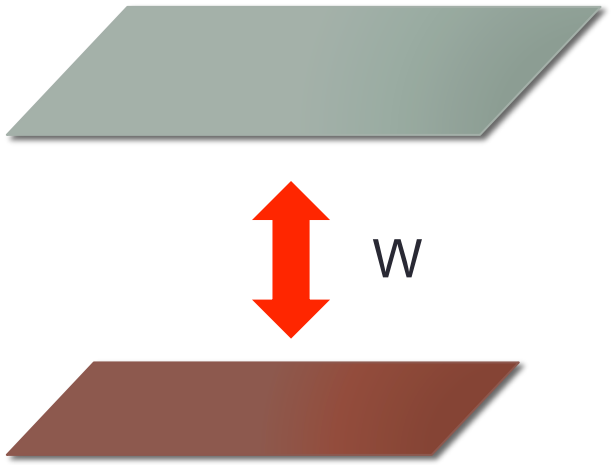
DBNs

- World = many levels
  - Pixel, edges, object parts, objects…
- Hierarchical models represent them all at once
- CBNs = layers of feature detectors



Restricted Boltzman Machine (RBM)

- 2 layer bipartite graph



H = Hidden layer

V = Visible layer

- Binary values

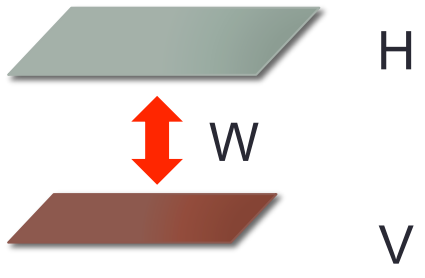
Restricted Boltzman Machine (RBM)

- Energy of a configuration:

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{i,j} v_i W_{ij} h_j - \sum_j b_j h_j - \sum_i c_i v_i,$$

- Probability of a configuration

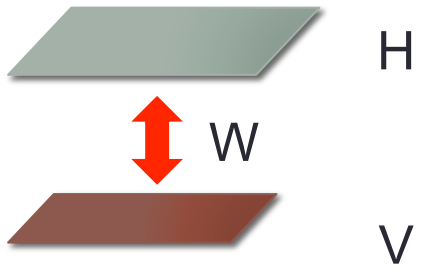
$$P(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} \exp(-E(\mathbf{v}, \mathbf{h})),$$



Restricted Boltzman Machine (RBM)

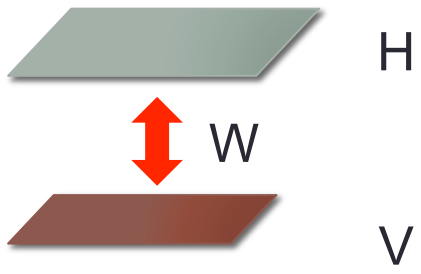
$$E(\mathbf{v}, \mathbf{h}) = -\sum_{i,j} v_i W_{ij} h_j - \sum_j b_j h_j - \sum_i c_i v_i,$$

- Fixed visible units: hidden units independent
- Fixed hidden units: visible units independent
- BLOCK GIBBS SAMPLING
  - Resample h given x
  - Resample x given h



Restricted Boltzman Machine (RBM)

- Parameters can be learned from training data
- Stochastic gradient ascent too difficult
- Contrastive divergence works well in practice

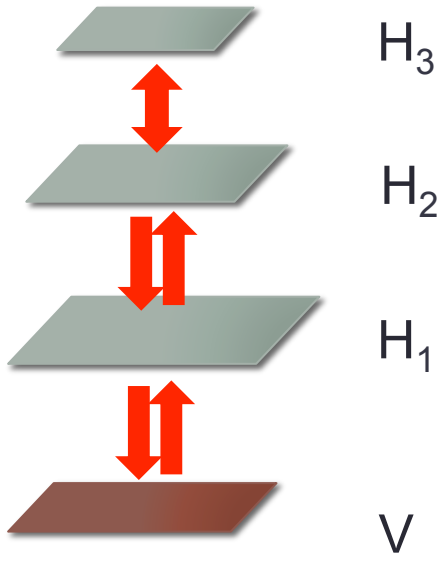


Deep Belief Network

- Stack several RBMs together
- Connection between adjacent layers
- No connections in the same layer

- Training:
- Phase 1) pre-training
- Phase 2) fine tuning
  - Unfold
  - Backprop.
  - min Reconstruction Error

Def: *activation of a unit*
= its expected value



NEW MACHINERY

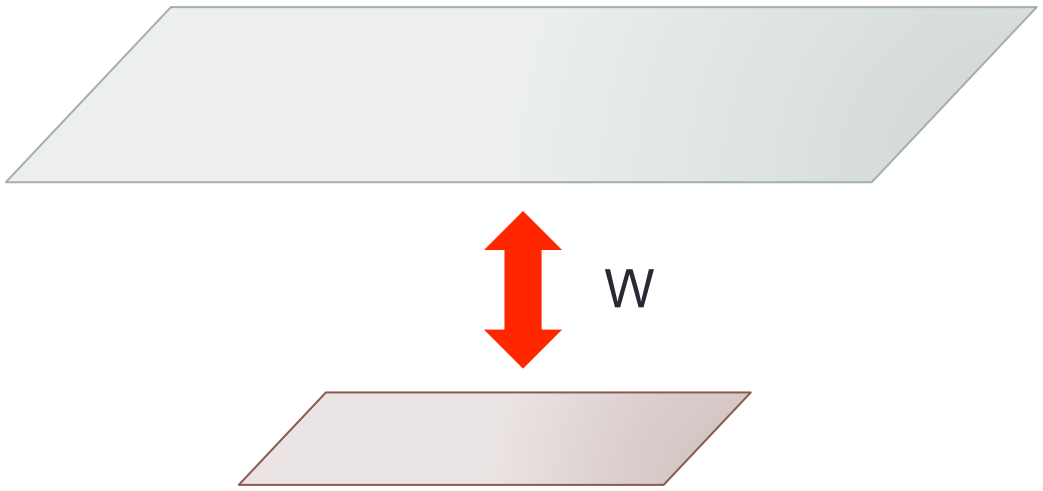
Convolutional RBMs

Probabilistic max-pooling

Sparsity constraint

Convolutional RBM (CRBM)

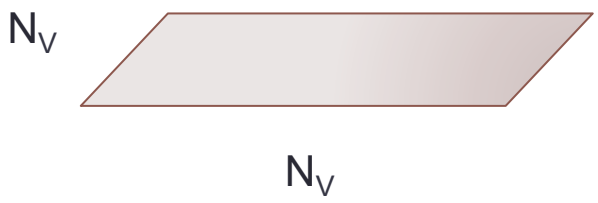
- Similar to RBM
- Weights are shared



H = Hidden layer

V = Visible layer

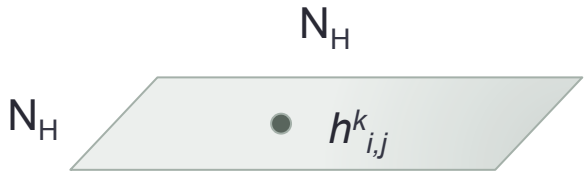
Convolutional RBM (CRBM)



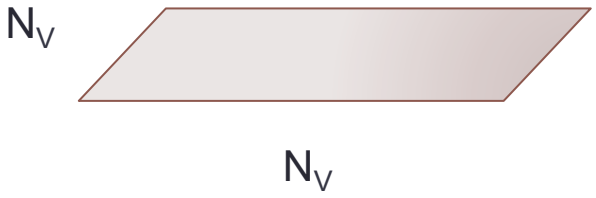
V = Visible layer

Convolutional RBM (CRBM)

- Hidden layer = K groups



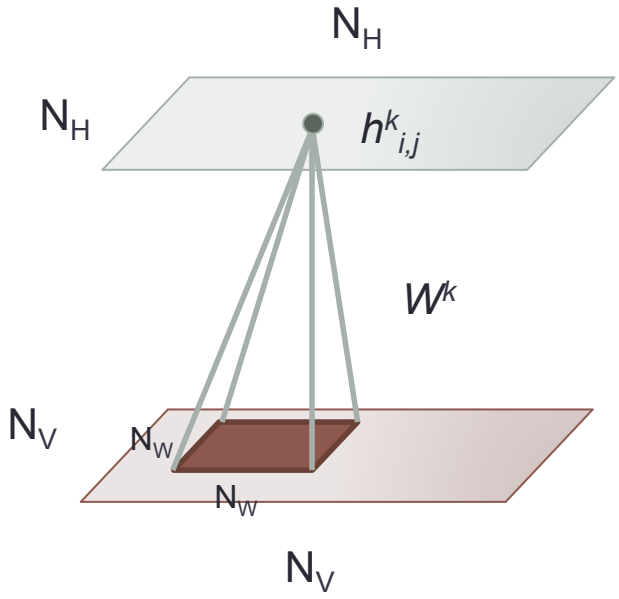
H^k



V = Visible layer

Convolutional RBM (CRBM)

- One filter to each hidden group
  - Filter weights shared between units

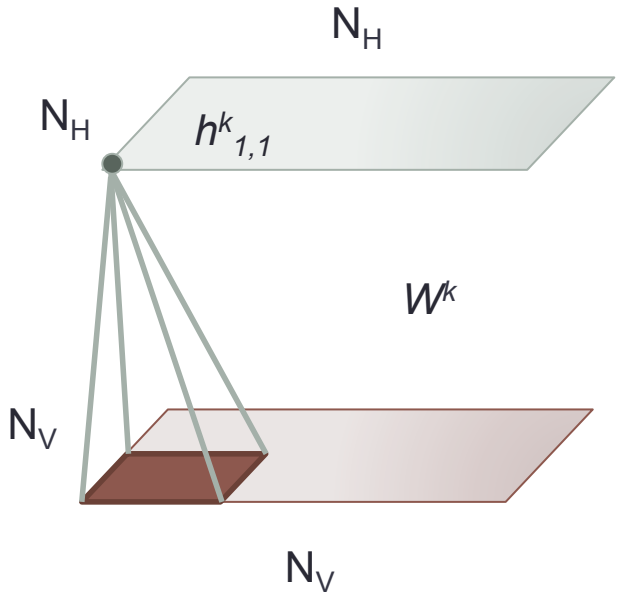


H^k (hidden layer)

V = Visible layer

Convolutional RBM (CRBM)

- One filter to each hidden group
  - Filter weights shared between units

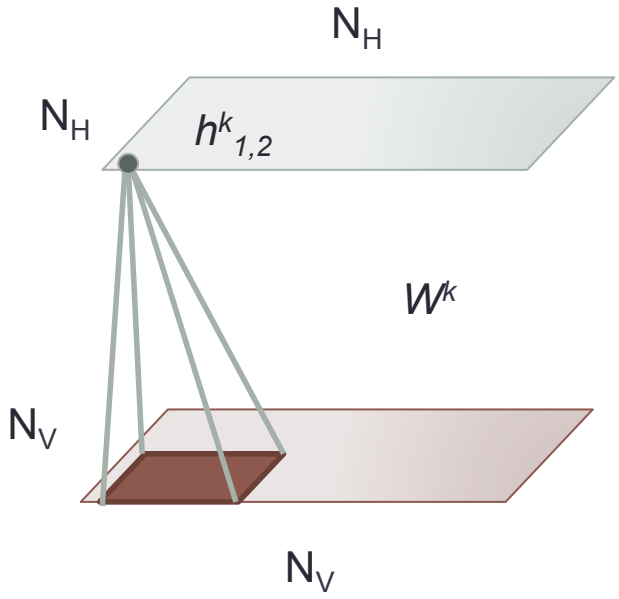


H^k (hidden layer)

V = Visible layer

Convolutional RBM (CRBM)

- One filter to each hidden layer
  - Filter weights shared between units

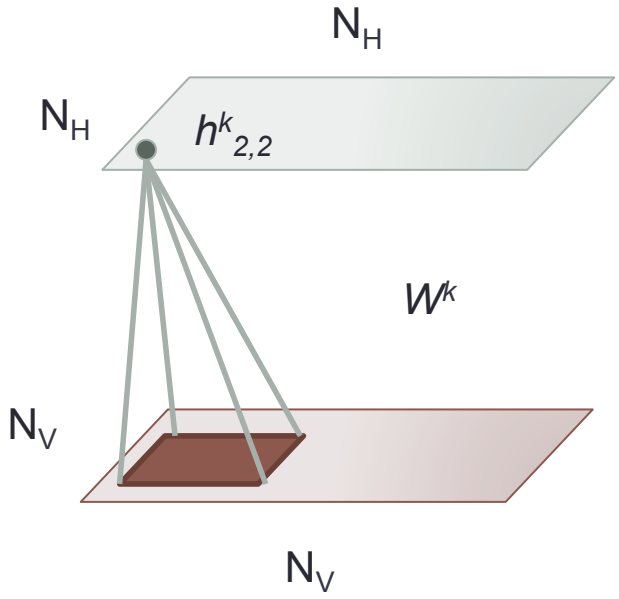


H^k (hidden layer)

V = Visible layer

Convolutional RBM (CRBM)

- One filter to each hidden layer
  - Filter weights shared between units

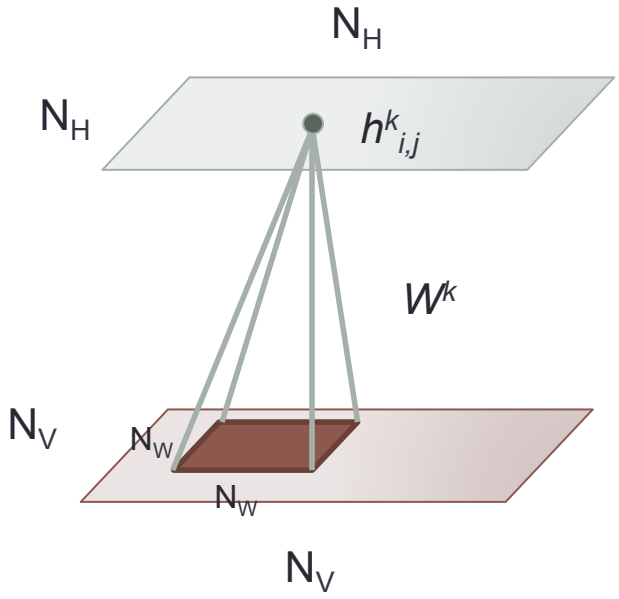


H^k (hidden layer)

V = Visible layer

Convolutional RBM (CRBM)

- One filter to each hidden group
  - Filter weights shared between units



H^k (hidden layer)

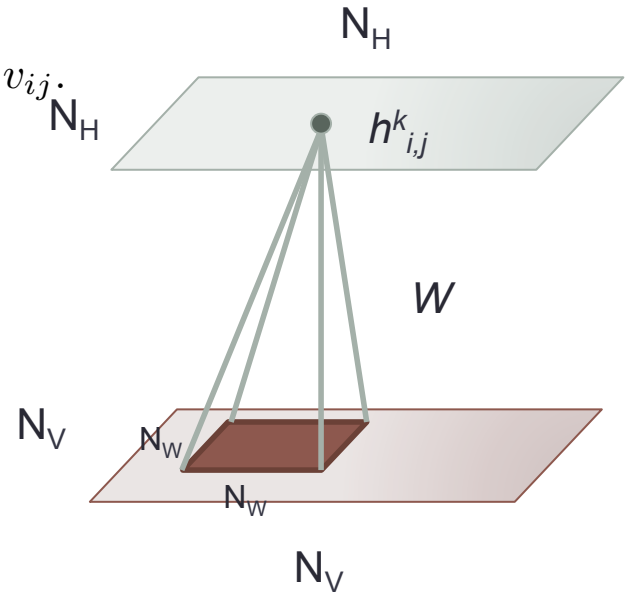
V = Visible layer

Convolutional RBM (CRBM)

$$E(\mathbf{v},\mathbf{h}) = -\sum_{k=1}^{K}\sum_{i,j=1}^{N_H}\sum_{r,s=1}^{N_W} h_{ij}^k W_{rs}^k v_{i+r-1,j+s-1} - \sum_{k=1}^{K} b_k \sum_{i,j=1}^{N_H} h_{ij}^k - c\sum_{i,j=1}^{N_V} v_{ij}.$$

$$E(\mathbf{v},\mathbf{h}) = -\sum_{k=1}^{K} h^k \bullet (\tilde{W}^k * v) - \sum_{k=1}^{K} b_k \sum_{i,j} h_{i,j}^k - c\sum_{i,j} v_{ij}.$$

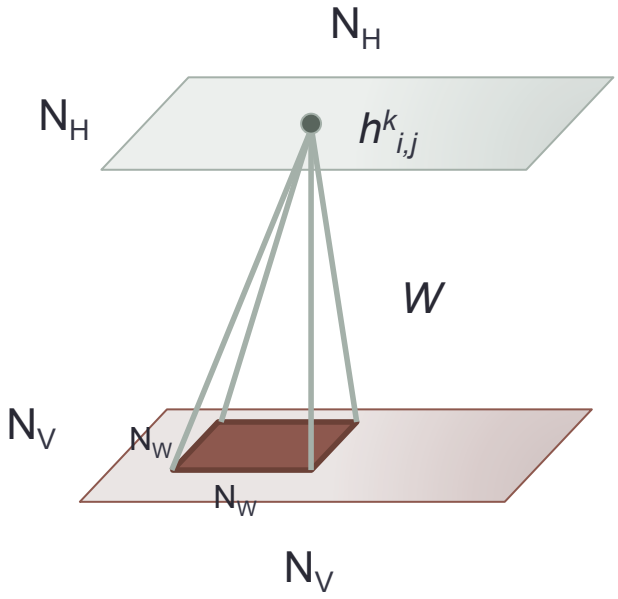
$$P(\mathbf{v},\mathbf{h}) = \frac{1}{Z}\exp(-E(\mathbf{v},\mathbf{h})),$$



Convolutional RBM (CRBM)

- Same conditional independence as with RBMs
- Can use block Gibbs sampling

$$P(h_{ij}^k = 1|\mathbf{v}) = \sigma((\tilde{W}^k * v)_{ij} + b_k)$$
$$P(v_{ij} = 1|\mathbf{h}) = \sigma((\sum_k W^k * h^k)_{ij} + c)$$



Probabilistic max-pooling

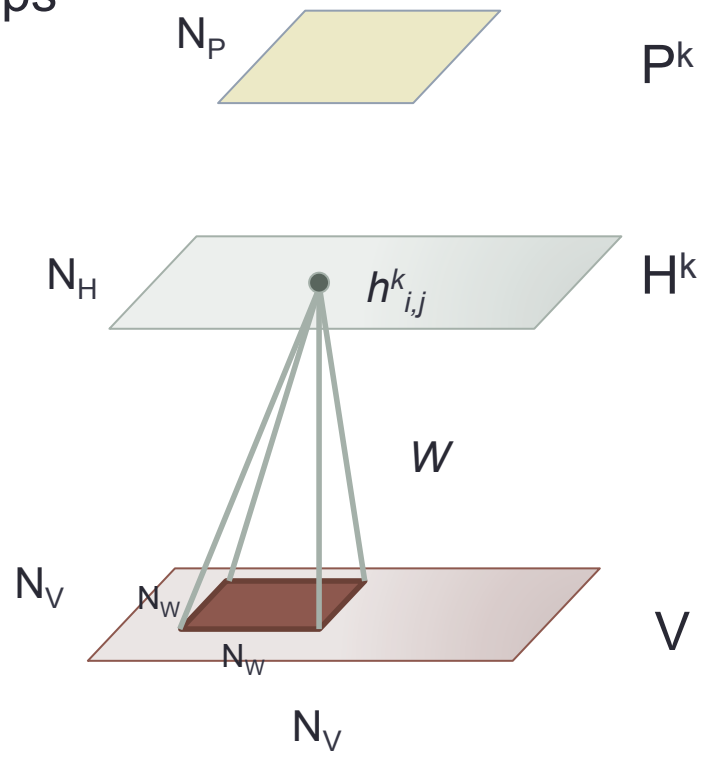
- Why not stack CRBMs together?
- WAIT: high-level detectors on larger and larger regions

- SOLUTION: prob max-pooling
- Alternate:
  - Detection layer D
  - Pooling layer P
    - units in P compute maximum activation of the units in small regions of D
- BENEFITS= translation invariance + more efficiency

Probabilistic max-pooling

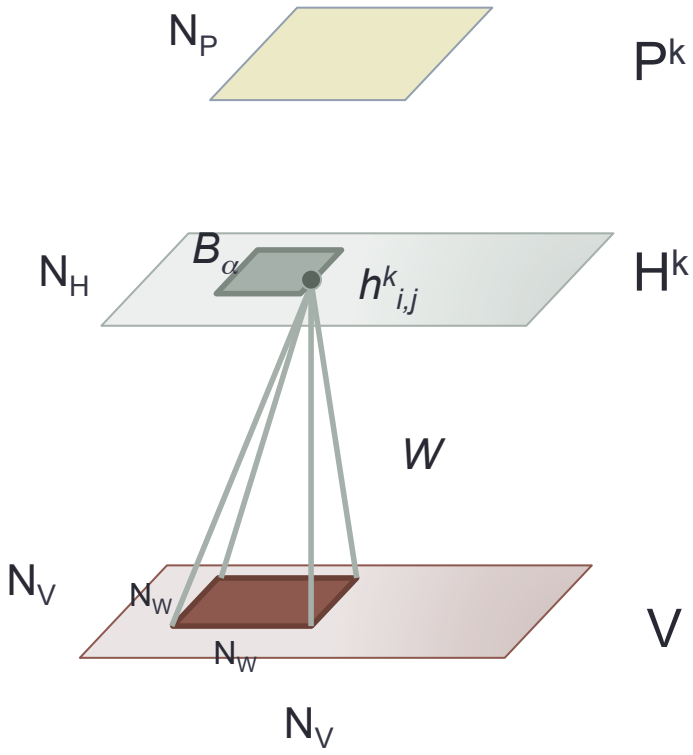
- Detection and pooling layers have K groups
- P^k shrinks H^k
- By a factor of C

- Constraints to relate P^k and H^k



Probabilistic max-pooling

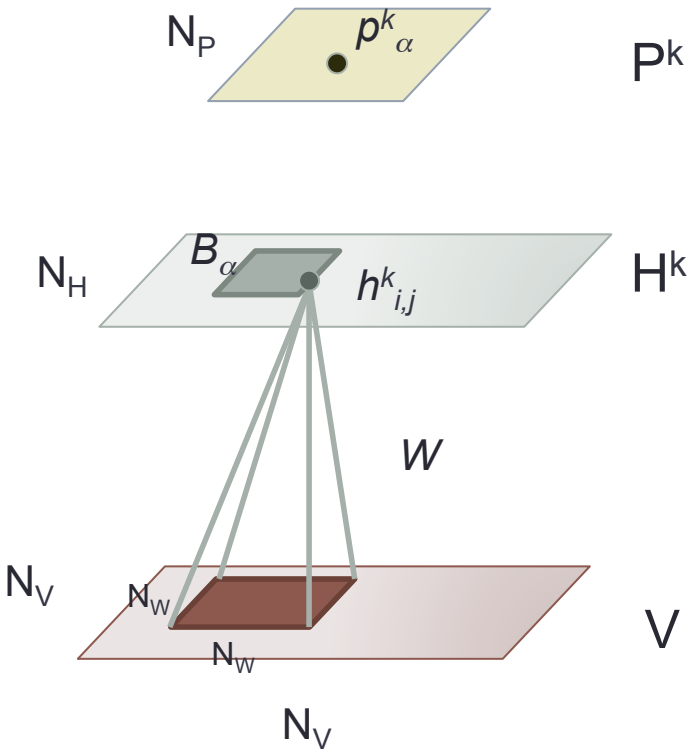
- H^k partitioned in C x C blocks
- B_α = {(i,j), h^{kij} in block α}



Probabilistic max-pooling

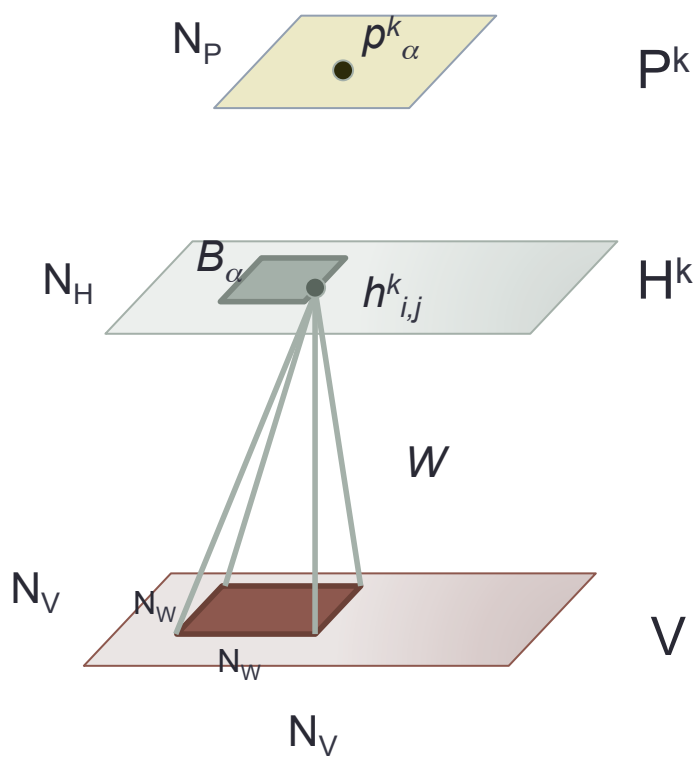
- H^k partitioned in C x C blocks
- B_α = {(i,j), h^{kij} in block α}
- Each block α connected to one $p^k{}_\alpha$

$$N_P = N_H / C$$



Probabilistic max-pooling

- Detection units in B_α and pooling unit $p^k{}_\alpha$
- Connected by a single potential

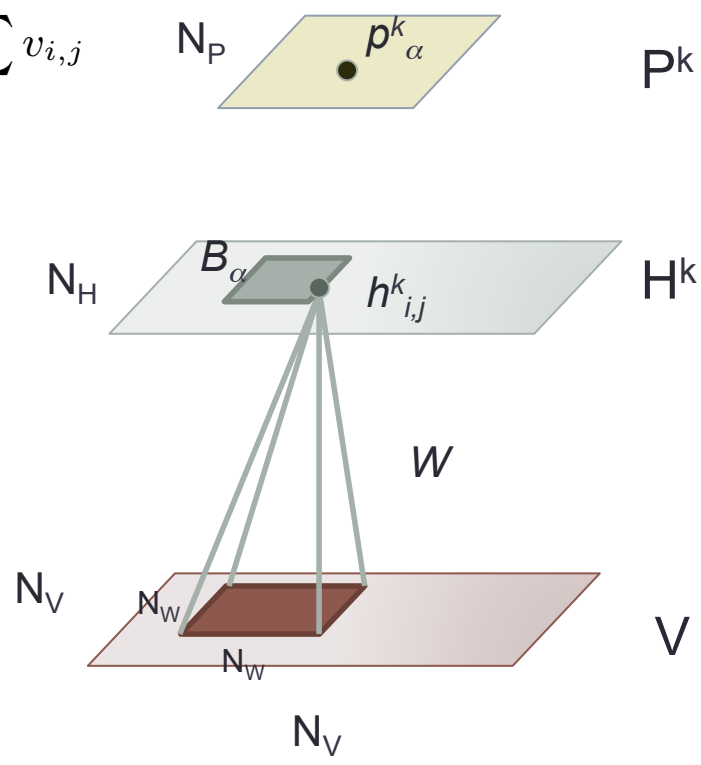


(1) At most one of the detection units ON
(2) Pooling unit ON ←→ a detection unit ON

Probabilistic max-pooling

$$E(\mathbf{v},\mathbf{h}) = -\sum_k \sum_{i,j} \left(h^k_{i,j} (\tilde{W}^k * v)_{i,j} + b_k h^k_{i,j} \right) - c \sum_{i,j} v_{i,j}$$

$$\text{subj. to} \quad \sum_{(i,j)\in B_\alpha} h^k_{i,j} \leq 1, \quad \forall k, \alpha.$$

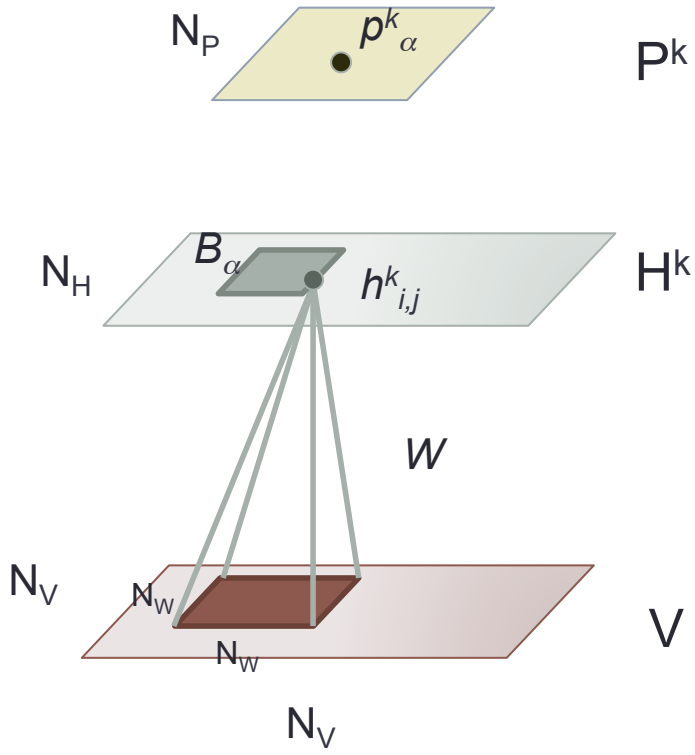


(1) At most one of the detection units ON
(2) Pooling unit ON ⟷ a detection unit ON

Probabilistic max-pooling

- This makes things easier
- FROM C^2+1 r.v.'s with 2^{C^2}
- TO only 1 r.v. with C^2+1 values

(1) At most one of the detection unit ON
(2) Pooling unit ON ⟷ a detection unit ON

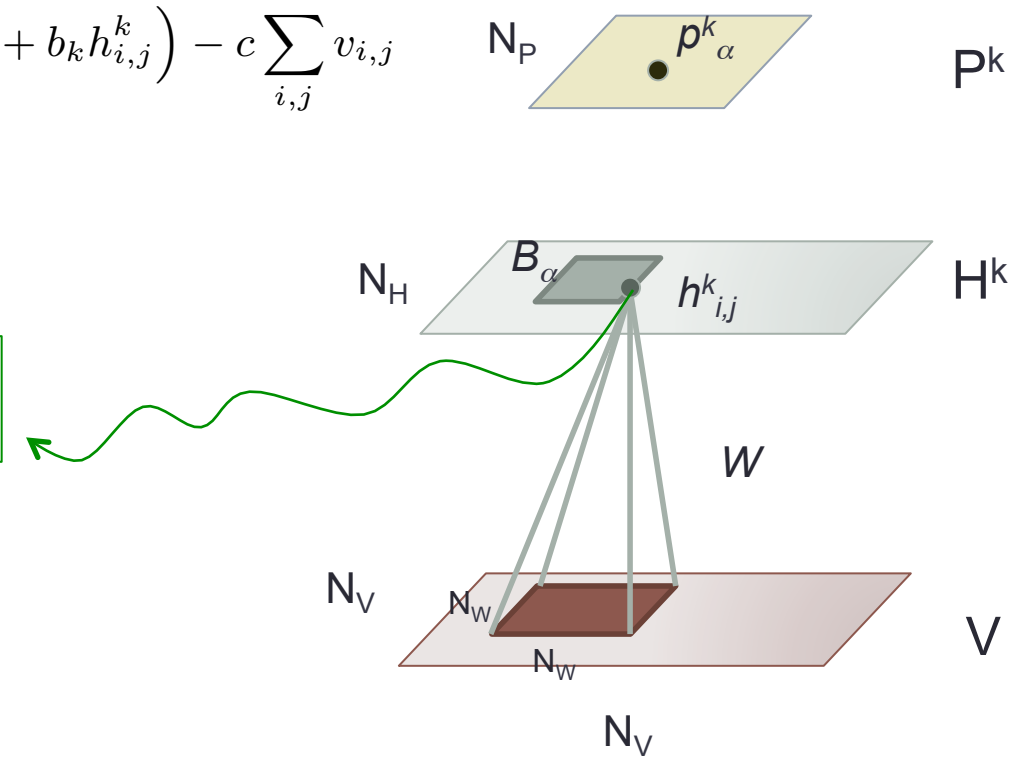


Sampling H and P given V

$$E(\mathbf{v},\mathbf{h}) = -\sum_{k}\sum_{i,j}\left(h^k_{i,j}(\tilde{W}^k * v)_{i,j} + b_k h^k_{i,j}\right) - c\sum_{i,j} v_{i,j}$$

$$\text{subj. to} \quad \sum_{(i,j)\in B_\alpha} h^k_{i,j} \le 1, \quad \forall k, \alpha.$$

$$I(h^k_{ij}) \triangleq b_k + (\tilde{W}^k * v)_{ij}.$$

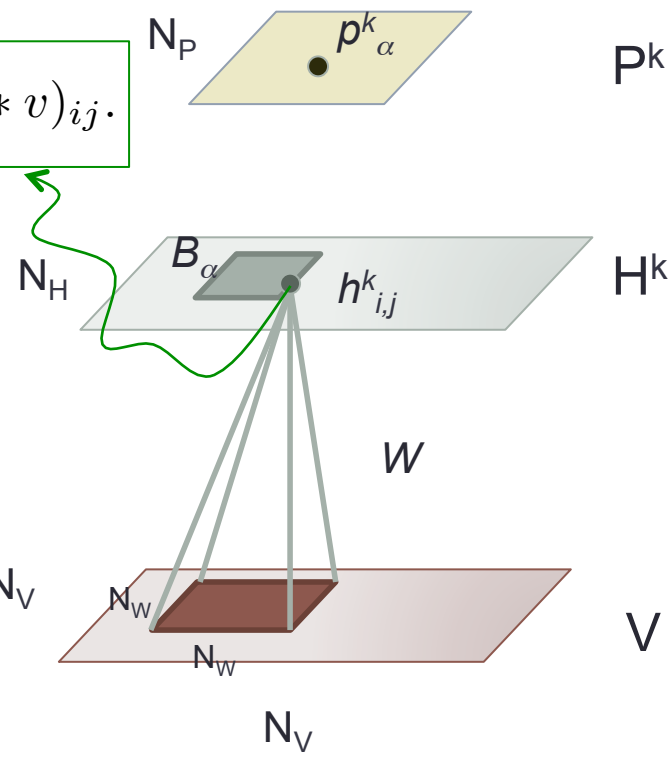


Sampling H and P given V

$$I(h_{ij}^k) \triangleq b_k + (\tilde{W}^k * v)_{ij}.$$

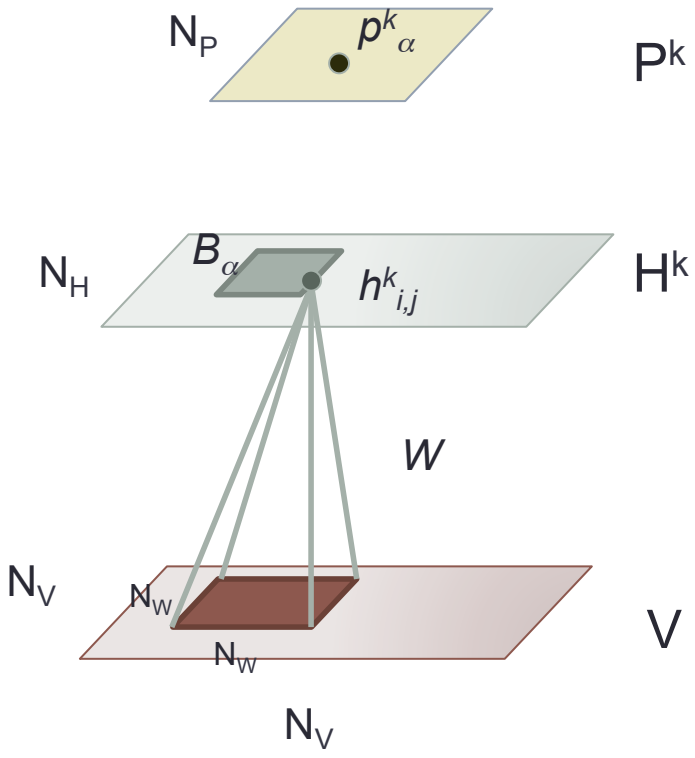
$$P(h_{i,j}^k = 1|\mathbf{v}) = \frac{\exp(I(h_{i,j}^k))}{1 + \sum_{(i',j') \in B_\alpha} \exp(I(h_{i',j'}^k))}$$

$$P(p_\alpha^k = 0|\mathbf{v}) = \frac{1}{1 + \sum_{(i',j') \in B_\alpha} \exp(I(h_{i',j'}^k))}.$$



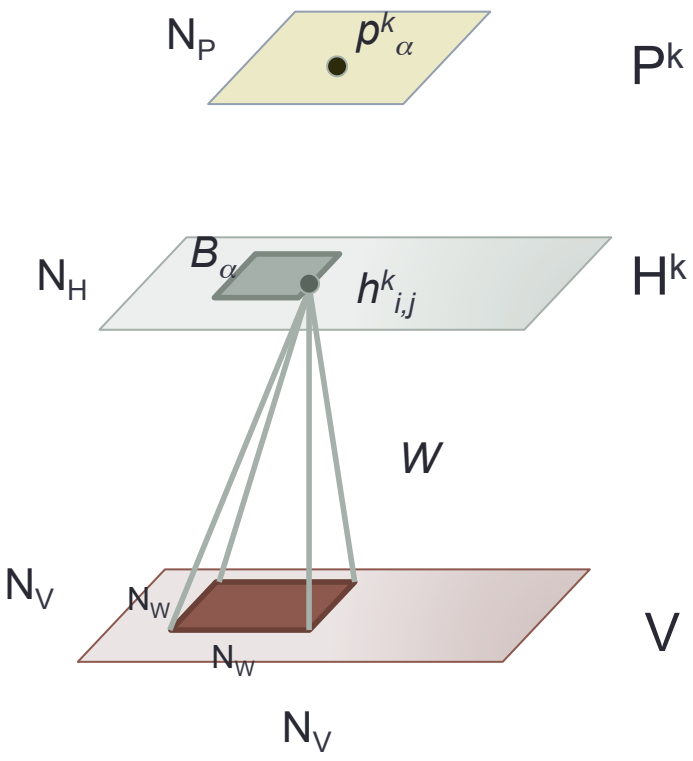
Sampling V given H

$$P(v_{ij} = 1|\mathbf{h}) \quad = \quad \sigma((\sum_k W^k * h^k)_{ij} + c)$$



Sparsity regularization

- PROBLEM: the model is overcomplete
- By a factor of *K*
  - $|H^k| \sim |V|$
- RISK learning trivial solutions
- SOLUTION: force sparsity
- Hidden units ← low activations

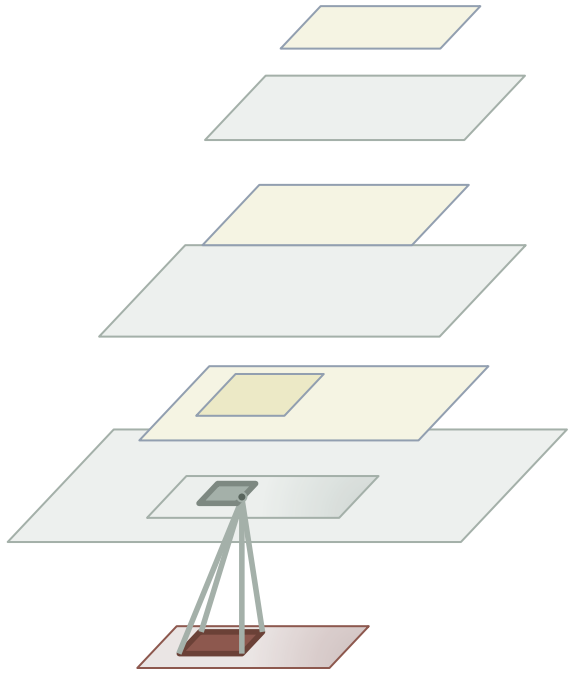


AND FINALLY…

Convolutional Deep Belief Networks!

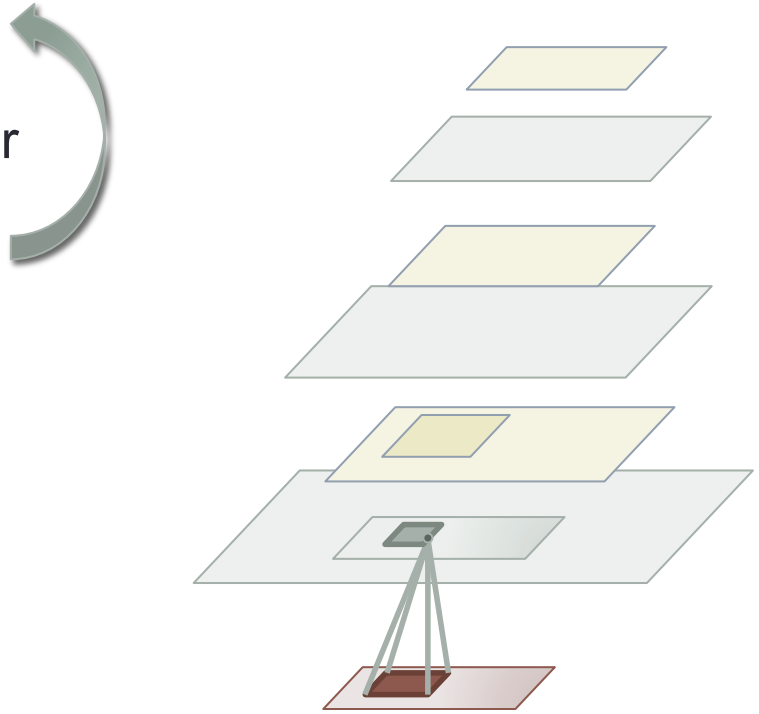
Convolutional Deep Belief Networks

1. Hierarchical generative model for full-sized images
2. Stack together several max-pooling CRBMs
3. Scales gracefully to realistic images
4. Translation invariant



Training a CDBN

- Energy = sum of the energy at each level
- Training = greedy pre-training
  - Start from bottom layer
  - Train weight
  - Freeze weight
  - Use activation as input to next layer
  - Move up one layer

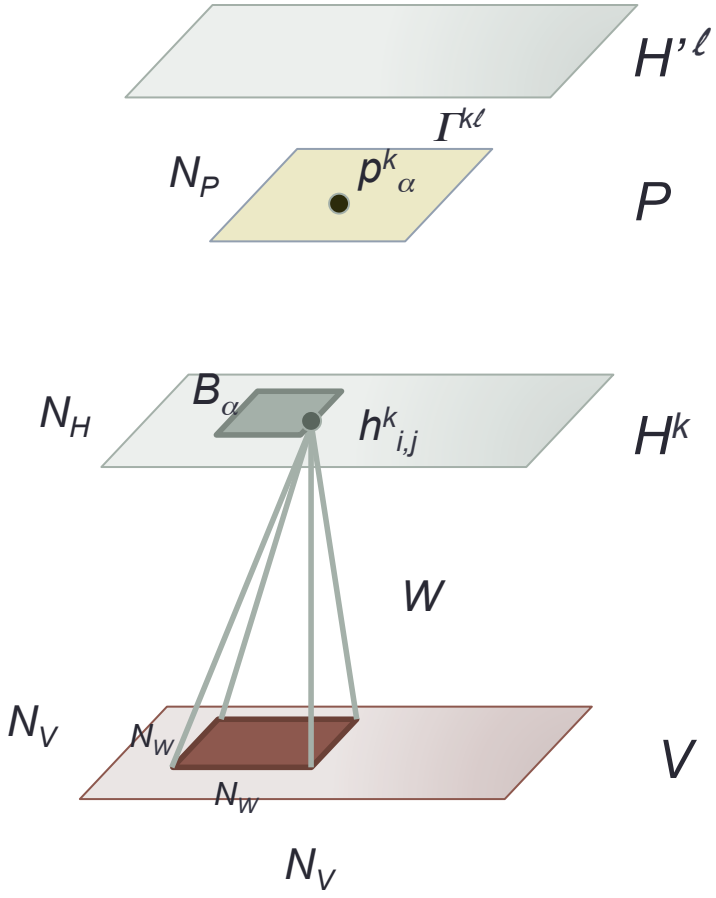


Representing an image with a CDBN

- Parameters are now fixed
- Sample from distribution of the hidden layers conditioned on the image
- Use Gibbs sampling

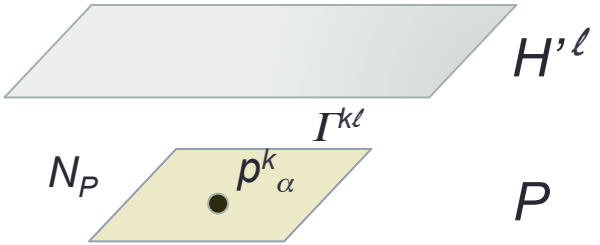
Representing an image with a CDBN

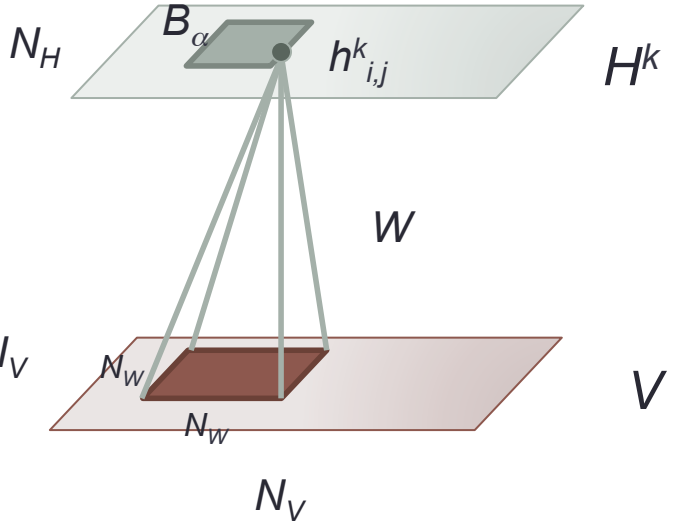
- CDBN with
  - Visible layer *V*
  - Detection layer *H*
  - Pooling layer *P*
  - Detection layer *H'*
- *H'* has *K'* groups
- Shared weights $\Gamma = \{\Gamma^{1,1} \ldots \Gamma^{K,K'}\}$
- $\Gamma^{k,\ell}$ connects:
  - Pooling unit P^k
  - Detection unit H^ℓ



Representing an image with a CDBN

- *H'* has *K'* groups
- Shared weights $\Gamma = \{\Gamma^{1,1} \ldots \Gamma^{K,K'}\}$
- $\Gamma^{k,\ell}$ connects:
  - Pooling unit P^k
  - Detection unit H^ℓ





Interaction *V - H*

$$E(\mathbf{v},\mathbf{h},\mathbf{p},\mathbf{h}') = -\sum_k v \bullet (W^k * h^k) - \sum_k b_k \sum_{ij} h^k_{ij}$$

$$-\sum_{k,\ell} p^k \bullet (\Gamma^{k\ell} * h'^{\ell}) - \sum_{\ell} b'_{\ell} \sum_{ij} h'^{\ell}_{ij}$$

Interaction *P – H'*

Representing an image with a CDBN

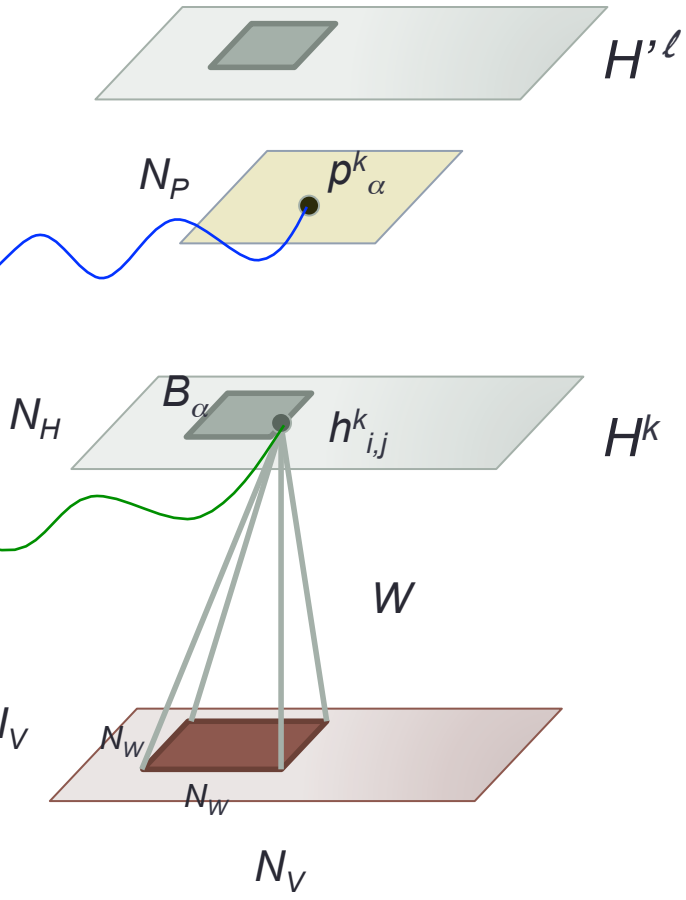
$$I(p^k_\alpha) \triangleq \sum_\ell (\Gamma^{k\ell} * h'^\ell)_\alpha$$

$$I(h^k_{ij}) \triangleq b_k + (\tilde{W}^k * v)_{ij}.$$

Interaction *V - H*

$$E(\mathbf{v},\mathbf{h},\mathbf{p},\mathbf{h}') = -\sum_k v \bullet (W^k * h^k) - \sum_k b_k \sum_{ij} h^k_{ij}$$
$$-\sum_{k,\ell} p^k \bullet (\Gamma^{k\ell} * h'^\ell) - \sum_\ell b'_\ell \sum_{ij} h'^\ell_{ij}$$

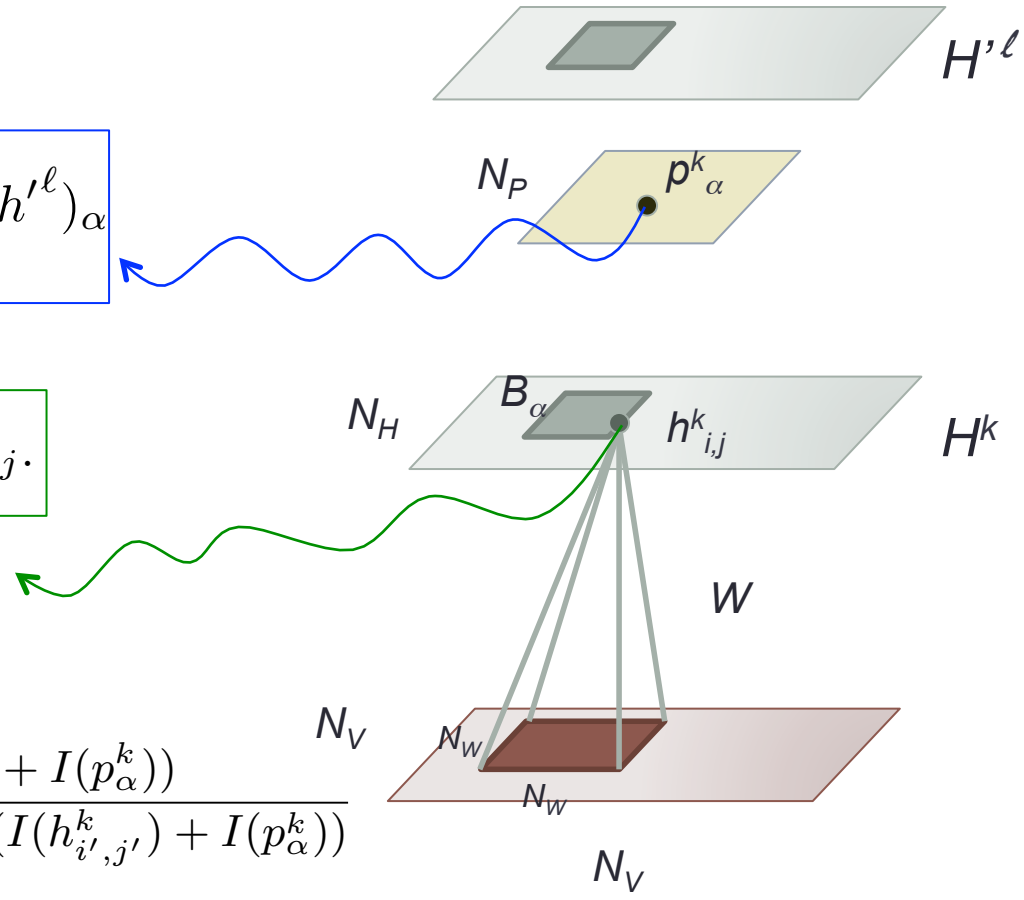
Interaction *P – H'*



Representing an image with a CDBN

$$I(p_\alpha^k) \triangleq \sum_\ell (\Gamma^{k\ell} * h'^{\ell})_\alpha$$

$$I(h_{ij}^k) \triangleq b_k + (\tilde{W}^k * v)_{ij}.$$



$$P(h_{i,j}^k = 1|\mathbf{v}, \mathbf{h}') = \frac{\exp(I(h_{i,j}^k) + I(p_\alpha^k))}{1 + \sum_{(i',j') \in B_\alpha} \exp(I(h_{i',j'}^k) + I(p_\alpha^k))}$$

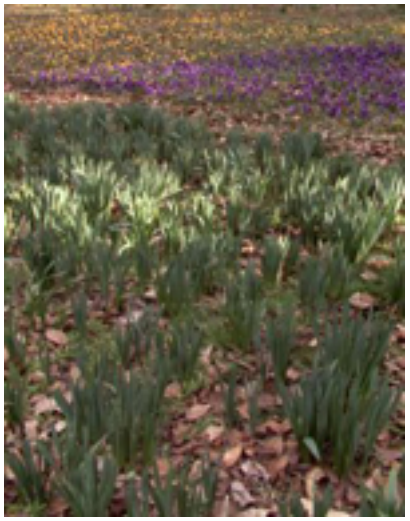
$$P(p_\alpha^k = 0|\mathbf{v}, \mathbf{h}') = \frac{1}{1 + \sum_{(i',j') \in B_\alpha} \exp(I(h_{i',j'}^k) + I(p_\alpha^k))}.$$

APPLICATIONS

Images

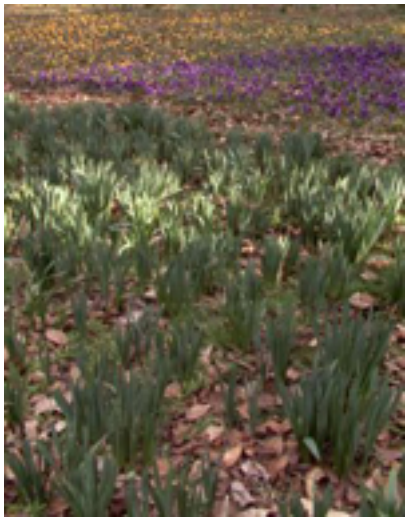
Hierarchical representation from natural scenes

- DATA = Kyoto natural image dataset
- CDBN with 2 layers
- *K* = *24* groups at layer 1
  - *10 x 10* pixel filters (bases)
- *K'* = *100* groups at layer 2
  - *10 x 10* pixels bases
- *C* = *2*

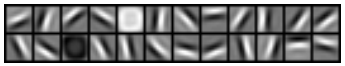


Hierarchical representation from natural scenes

- Kyoto natural image dataset

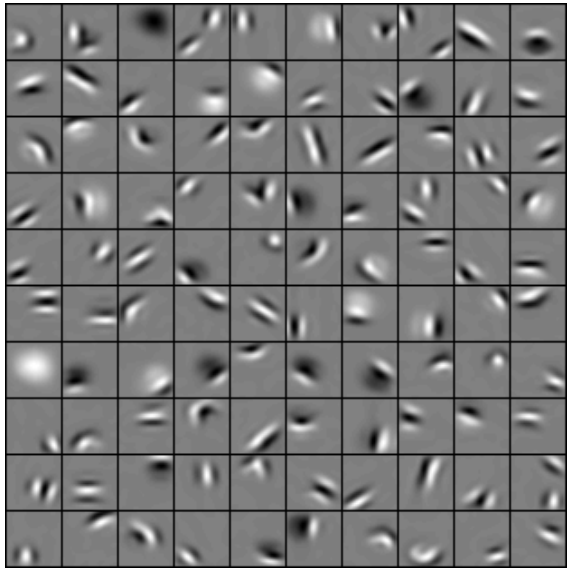


1st layer bases

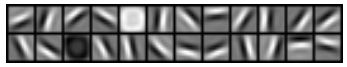


Hierarchical representation from natural scenes

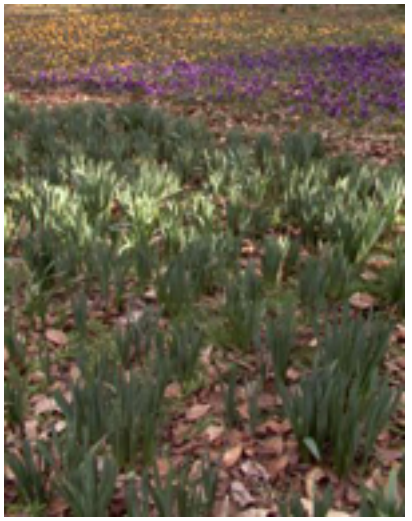
- Kyoto natural image dataset



2nd layer bases



1st layer bases



Self taught learning

- DATA = Caltech-101

- Use unlabeled data improve performance in classification

Self taught learning

- DATA = Caltech-101

- For each Caltech-101 image
- Compute activations of CDBN learned on Kyoto dataset
- These are the features
- Use an SVM with spatial pyramid kernel

Self taught learning

- DATA = Caltech-101

Table 1. Classification accuracy for the Caltech-101 data

Training Size	15	30
CDBN (first layer)	53.2±1.2%	60.5±1.1%
CDBN (first+second layers)	57.7±1.5%	65.4±0.5%
Raina et al. (2007)	46.6%	-
Ranzato et al. (2007)	-	54.0%
Mutch and Lowe (2006)	51.0%	56.0%
Lazebnik et al. (2006)	54.0%	64.6%
Zhang et al. (2006)	59.0±0.56%	66.2±0.5%

1. Combining two layers better than first layer alone
2. Competitive with highly-specialized features (e.g., SIFT)
3. CDBM was trained on natural scenes
   - CDBM = highly general representation of images

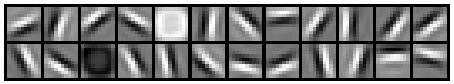
Unsupervised Learning of object parts

- DATA = Caltech-101

3rd layer bases

2nd layer bases

1st layer bases



APPLICATIONS

Music

Music auto-tagging

TEMPORAL POOLING AND MULTISCALE LEARNING FOR AUTOMATIC ANNOTATION AND RANKING OF MUSIC AUDIO

Philippe Hamel, Simon Lemieux, Yoshua Bengio
DIRO, Université de Montréal
Montréal, Québec, Canada
{hamelphi,lemiesim,bengioy}@iro.umontreal.ca

Douglas Eck
Google Inc.
Mountain View, CA, USA
deck@google.com

Music annotation and ranking

- Use tags to describe music
  - (genres, instruments, moods)
- Annotation = assign tags to a song
- Ranking = finding a song that best correspond to a tag

- A computer can do that.

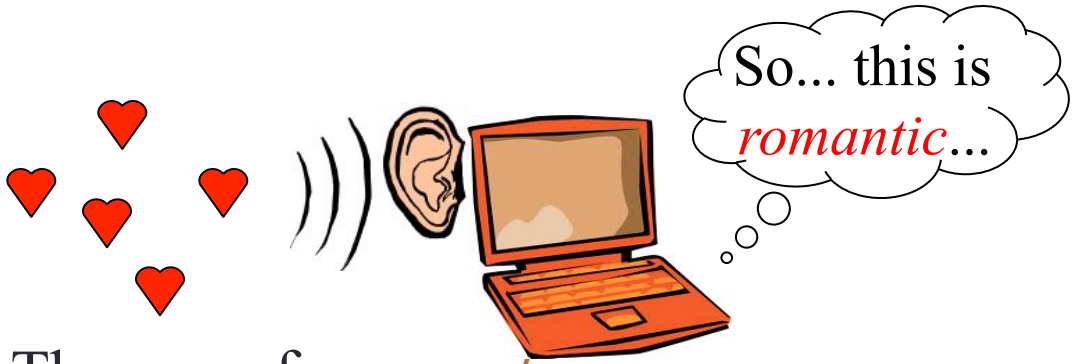
Music automatic annotation and ranking

- A auto-tagger can do all of this…

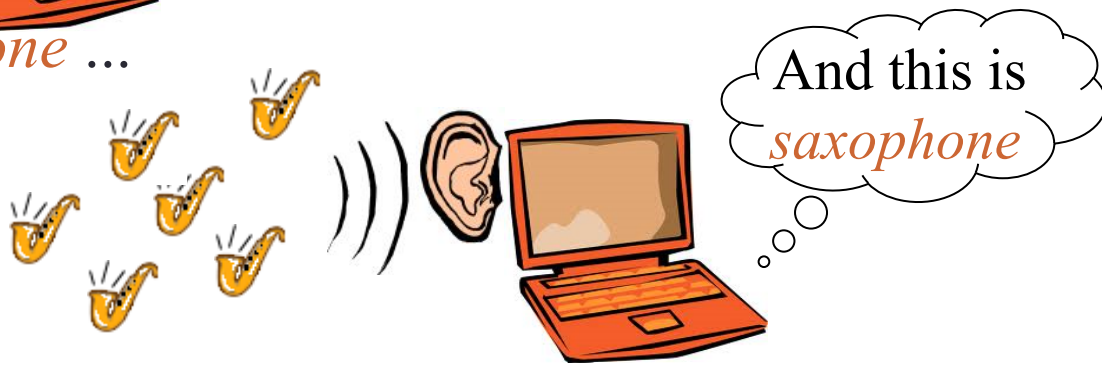
1. extract relevant features from music audio
2. infer abstract concepts from these features.

Music automatic annotation and ranking

- To learn/model the tag *romantic*, the auto-tagger needs to "listen to" songs that are associated with this tag...



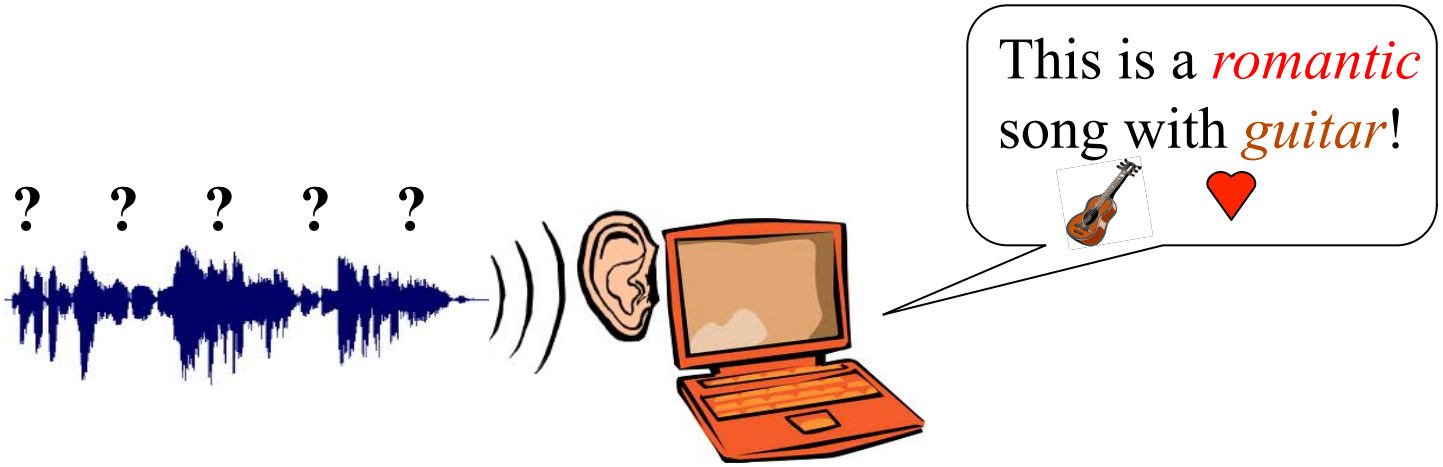
- The same for *saxophone* ...



- And many others:
  - ... *guitar* *hard rock* *driving*

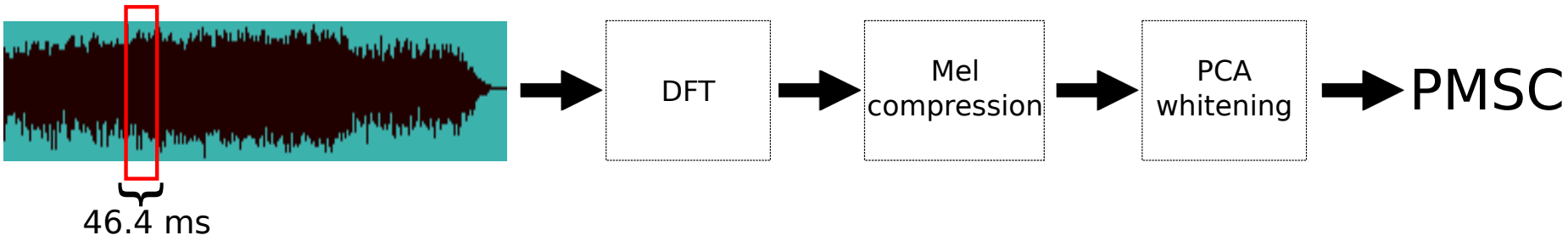
Music automatic annotation and ranking

- Then, it can describe a new song with the most appropriate tags:



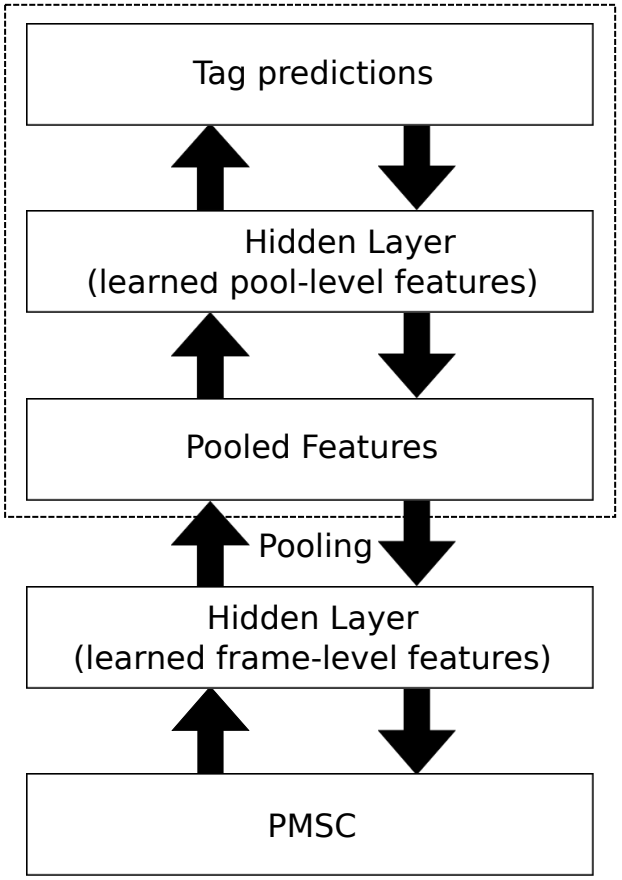
Audio features and preprocessing

1. From audio to spectrum
2. Energy bands
3. PCA
4. = Principal Mel-Spectrum Components (PMSC).



Multi-Time-Scale Learning (MTSL)

- Applies multi-scale learning to music
  - Uses CDBN on sequences of audio features
  - Weights are shared across frames
- Test several pooling functions



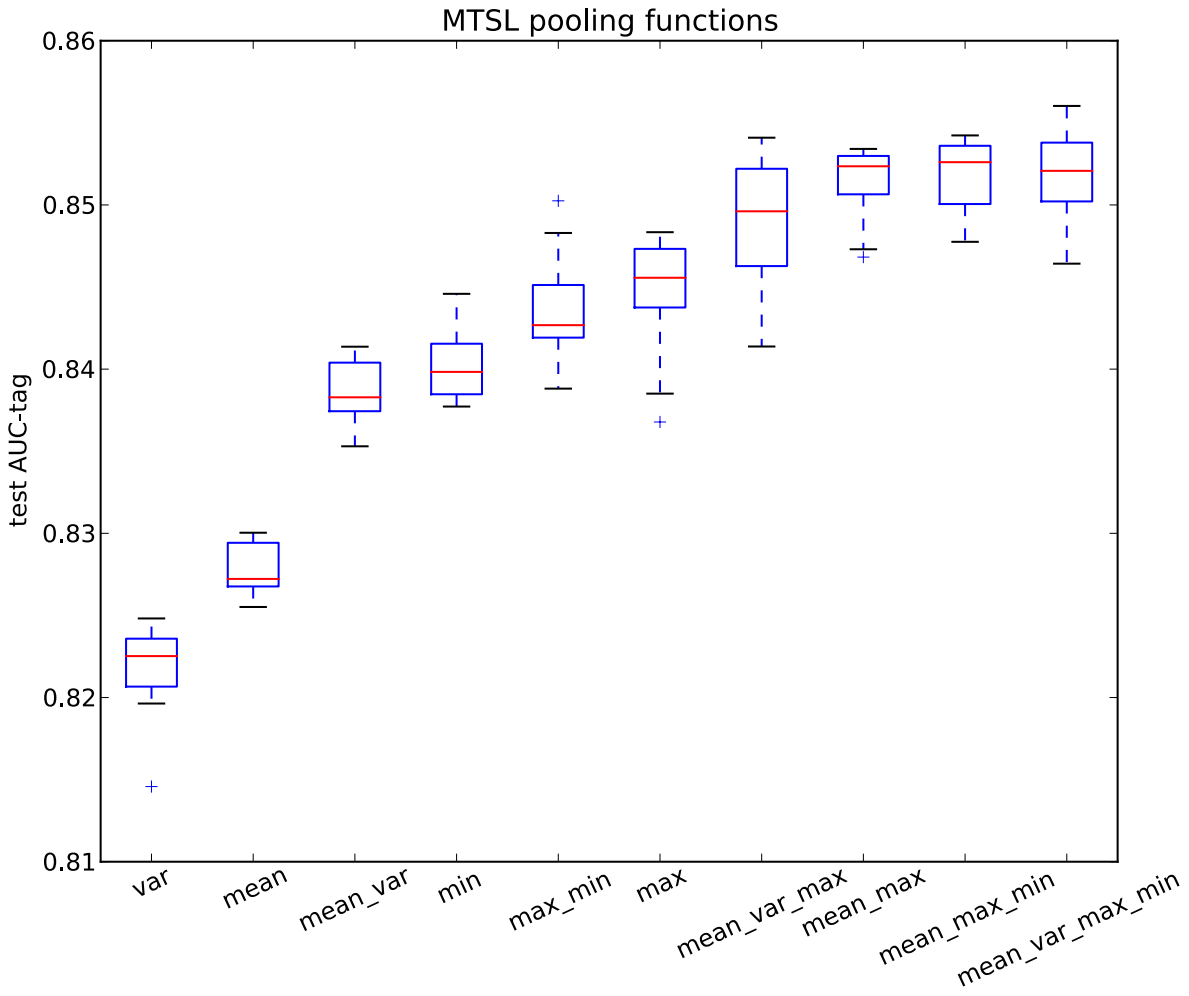
Results

Measure	Manzagol	Zhi	Mandel	Marsyas	PMSC+PFC	PSMC+MTSL
Average AUC-Tag	0.750	0.673	0.821	0.831	0.845	**0.861**
Average AUC-Clip	0.810	0.748	0.886	0.933	0.938	**0.943**
Precision at 3	0.255	0.224	0.323	0.440	0.449	**0.467**
Precision at 6	0.194	0.192	0.245	0.314	0.320	**0.327**
Precision at 9	0.159	0.168	0.197	0.244	0.249	**0.255**
Precision at 12	0.136	0.146	0.167	0.201	0.205	**0.211**
Precision at 15	0.119	0.127	0.145	0.172	0.175	**0.181**

Pooling functions

- Goal is too summarize/shrink information
  - Max, min, mean, var, 3^{rd} moment, 4^{th} moment
- Combinations
- Did not require longer training time

Pooling functions



Conclusions

- Convolutional deep believe network
  - Hierarchical generative model that alternates detection layers and pooling layers
  - Scales to high-dimensional real data
  - Translation invariant
- Application in
  - Computer vision
  - Computer audition
  - … and many others: multimodal data, audio classification…