Outline

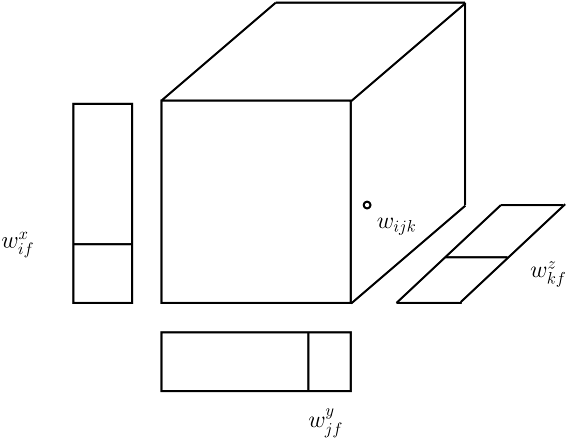
Outline

1. **Introduction**
   - Feature Learning
   - Correspondence in Computer Vision
   - Multiview feature learning
2. **Learning relational features**
   - Encoding relations
   - Learning
3. **Factorization, eigen-spaces and complex cells**
   - Factorization
   - Eigen-spaces, energy models, complex cells
4. **Applications and extensions**
   - Applications and extensions
   - Conclusions

Complexity

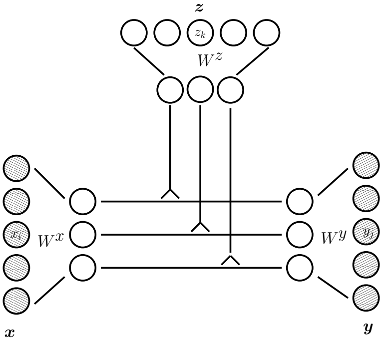
- The number of parameters is about $n \times n \times n$ (!)
- More, if we want sparse, overcomplete hiddens.
- There is a simple, yet far-reaching, way to reduce that number.

Factorization



$$w_{ijk} = \sum_{ijk} \sum_{f} w^x_{if} w^y_{jf} w^z_{kf}$$

Factorization is *filter matching*

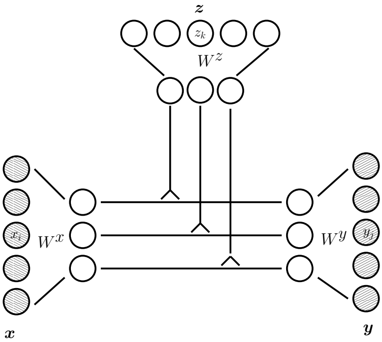


Inference with factorization

$$z_k = \sum_{ij} w_{ijk} x_i y_j \quad = \quad \sum_{ij} \Big(\sum_f w^x_{if} w^y_{jf} w^z_{kf} \Big) x_i y_j$$

$$= \quad \sum_f w^y_{jf} \cdot \Big(\sum_i w^x_{if} x_i \Big) \cdot \Big(\sum_j w^y_{kf} y_j \Big)$$

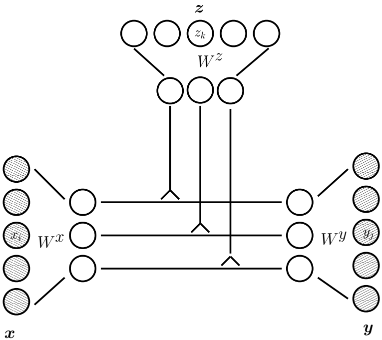
Factorization is *filter matching*



RBM energy

$$E = \sum_{ijk} \Big(\sum_f w^x_{if} w^y_{jf} w^z_{kf}\Big) x_i y_j z_k = \sum_f \Big(\sum_i w^x_{if} x_i\Big)\Big(\sum_j w^y_{jf} y_j\Big)\Big(\sum_k w^z_{kf} z_k\Big)$$

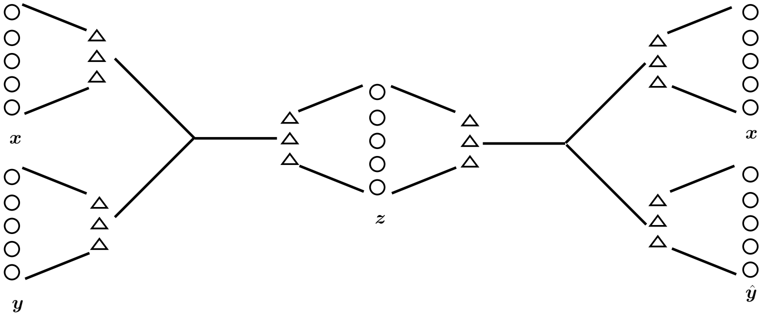
Factorized models



Factored Gated Boltzmann machines

- Exponentiate and normalize energy (just like RBM).
- Learning and inference exactly like before.
- (Taylor, 2009), (Memisevic, Hinton; 2009)

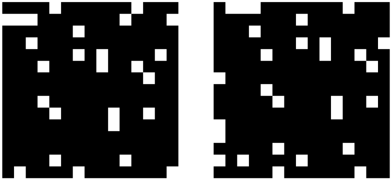
Factorized models



Factored Relational Autoencoders

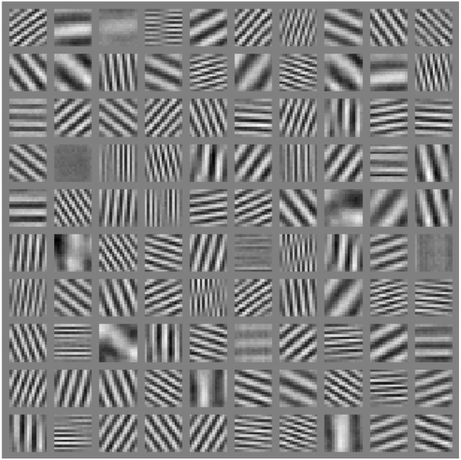
- Everything like before. Back-propagate through the filters.
- Conditional learning trivial as before.
- Joint learning by adding two asymmetric objectives.

Examples

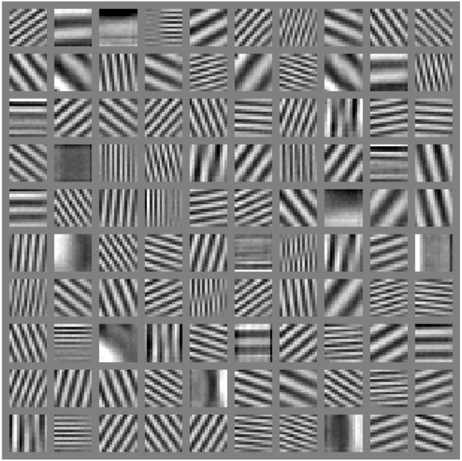


- Toy examples:
- There is no structure in these images.
- Only in *how they change*.

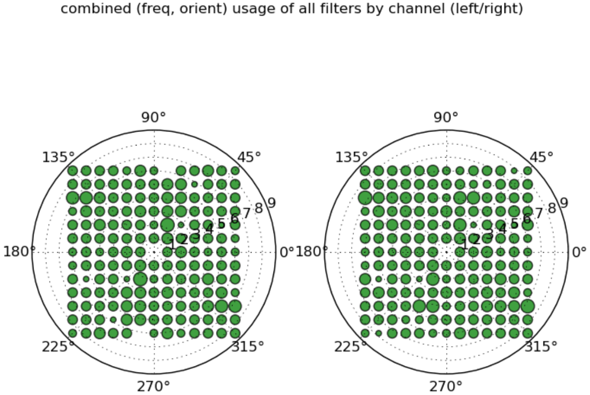
Learned filters w_{if}^{x}



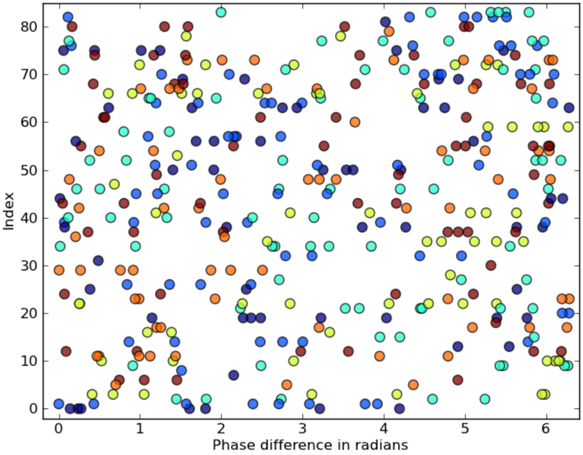
Learned filters w^y_{jf}



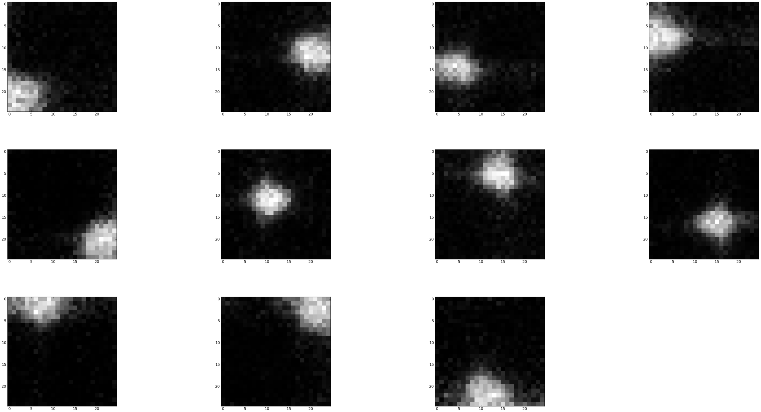
Frequency/orientation histograms



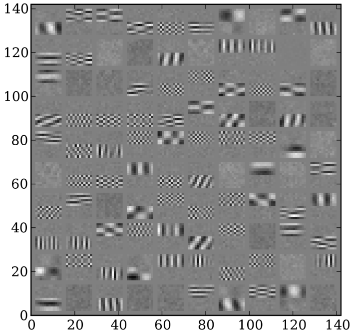
Frequency/orientation histograms



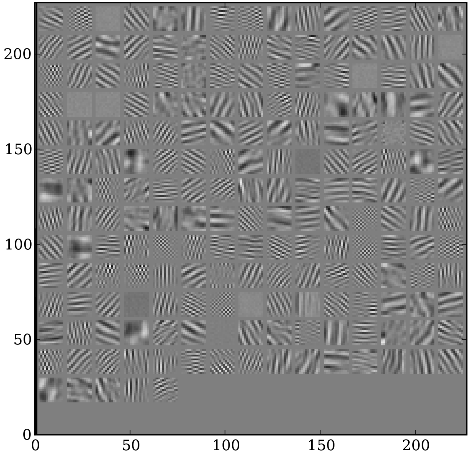
Velocity tuning of mapping units



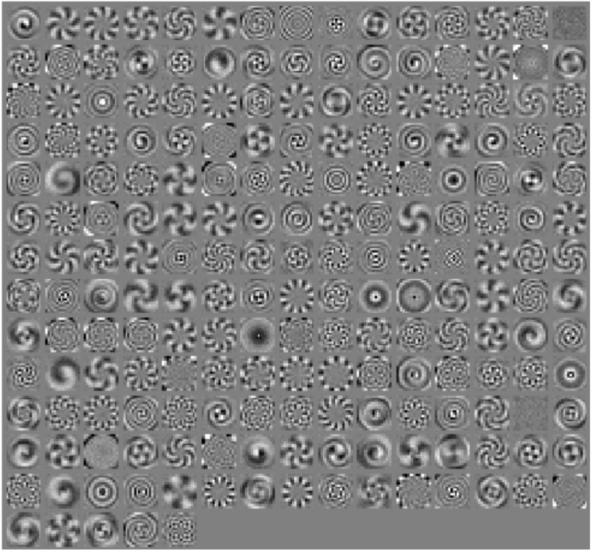
Filters learned from split-screen shifts



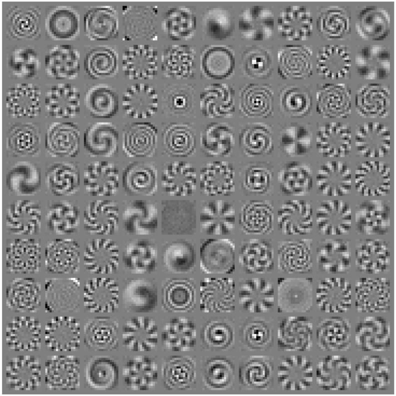
"Filtering"-filters



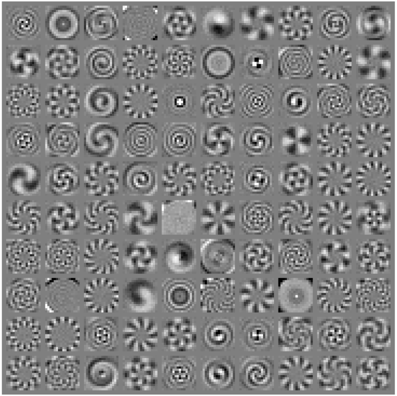
Rotation filters



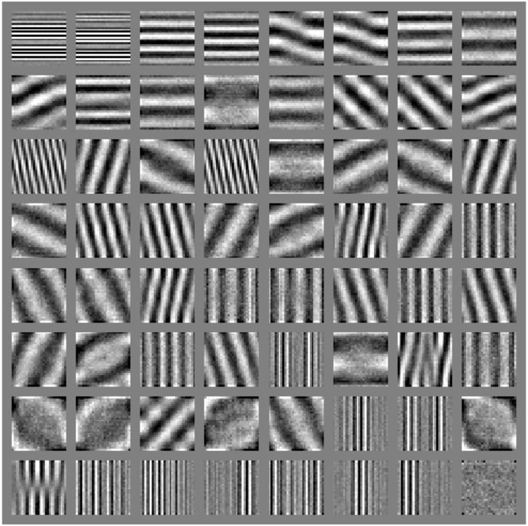
Rotation filters



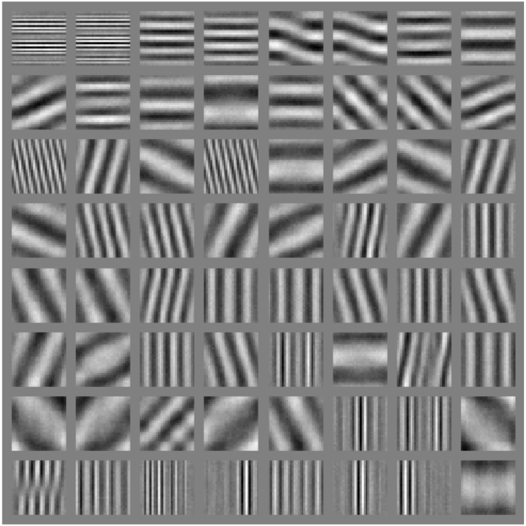
Rotation filters



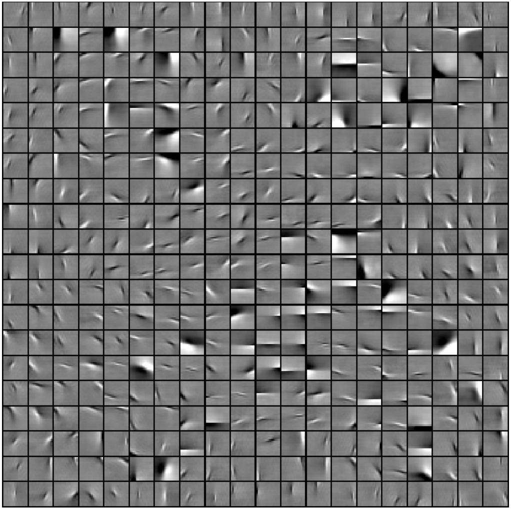
Filters learned by watching TV



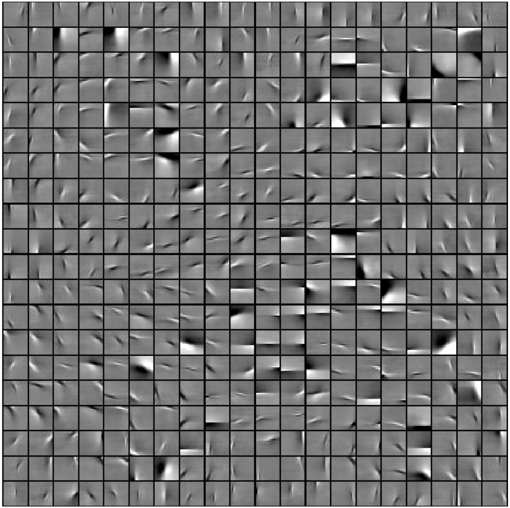
Filters learned by watching TV



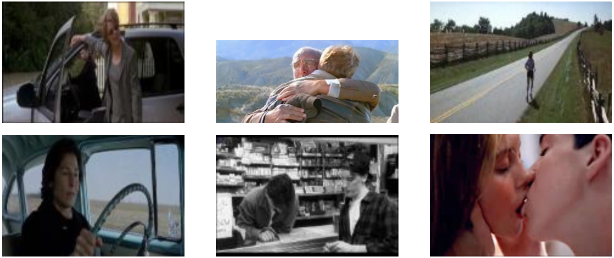
More filters learned by watching TV



More filters learned by watching TV



Action recognition



(Hollywood 2)

- Convolutional GBM (Taylor et al., 2010)

Outline

Linear image warps

- Consider a linear transformation in pixel space ("*warp*"):

$$y = Lx$$

- **Task**:

Given two images (x, y) what is the warp that relates them?

- This is exactly the problem that mapping units should be able to solve.

$y = Lx$

- We restrict our attention to **orthogonal warps**:

$$L^{\mathrm{T}} L = I$$

- Includes all permutations (“shuffling pixels”).
- Orthogonal warps are the *only* transformations we can see anyway, if all our images are *white*:

$$I = C_y = L C_x L^T = L L^T$$

- (Bethge, 2007)

Properties of orthogonal image warps

(I) Orthogonal transformations decompose into 2-D rotations

- An orthogonal matrix is similar to a matrix that performs **axis-aligned two-dimensional rotations:**

$$V^{\mathrm{T}} L V = \begin{bmatrix} R_1 & & \\ & \ddots & \\ & & R_k \end{bmatrix} \qquad R_i = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i) \end{bmatrix}$$

- This follows from the fact that the **eigen-decomposition**

$$L = V D V^{\mathrm{T}}$$

  has complex eigenvalues of length 1.
- The eigenspaces are also known as **invariant subspaces.**

Properties of orthogonal image warps

Example: Translation and the Fourier spectrum

- **Translation** is an example of an orthogonal warp.
- 1-D translation matrices are *circulants*, which have ones along an off-diagonal, like so:

$$L = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

- Eigenspaces are spanned by sine-/cosine-pairs (Fourier features).

Properties of orthogonal image warps

Quadrature pairs

- The invariant subspaces warps are two-dimensional, so **eigenvectors come in pairs**:

$$\big(\boldsymbol{v}_R, \boldsymbol{v}_I\big)$$

- In the case of translation, $\boldsymbol{v}_I$ is a sine and $\boldsymbol{v}_R$ is a cosine feature.
- Waves with 90 degrees phase difference are known as "**quadrature pair**".
- But the concept is more general and applies to all orthogonal matrices.
- The eigenvector pairs of orthogonal transformations have been referred to as "**generalized quadrature pairs**" (Bethge et al., 2007).

Properties of commuting image warps

(II) Commuting transformations share an eigen-basis

- Any two transformations that commute share a single eigen-basis.
- They differ only in their eigen*values*.

- "Proof": Consider A and B with $AB = BA$ and the eigenvector v of B with λ an eigenvalue with multiplicity one. We have

$$BAv = ABv = \lambda Av.$$

So Av is also an eigenvector of B with the same eigenvalue. And therefore, v must be an eigenvector of A, too.

Properties of commuting image warps

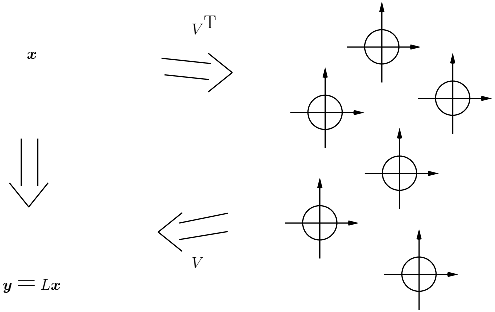
Translation Example continued

- All circulants share the Fourier basis as eigen-basis.

Properties of commuting image warps

Any two **orthogonal, commuting** transformations differ only with respect to the **rotation angles in the eigenpaces**.

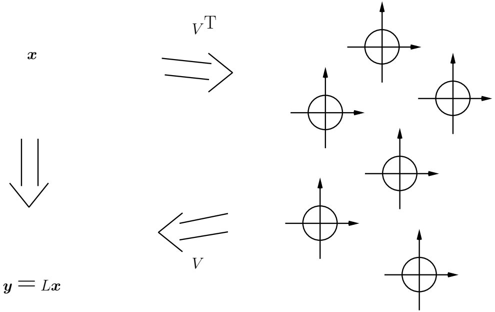
- So to *apply* a transformation you can equivalently perform a set of independent two-D rotations.



Properties of commuting image warps

Any two **orthogonal, commuting** transformations differ only with respect to the **rotation angles in the eigenpaces**.

- So to *apply* a transformation you can equivalently perform a set of independent two-D rotations.

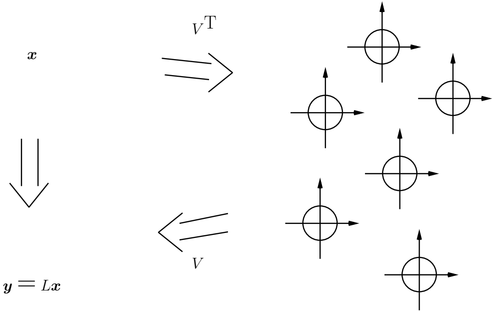


- To *infer* the transformation, given two images x and y: Project x and y onto the eigenvectors, then compute the rotation angles!

Properties of commuting image warps

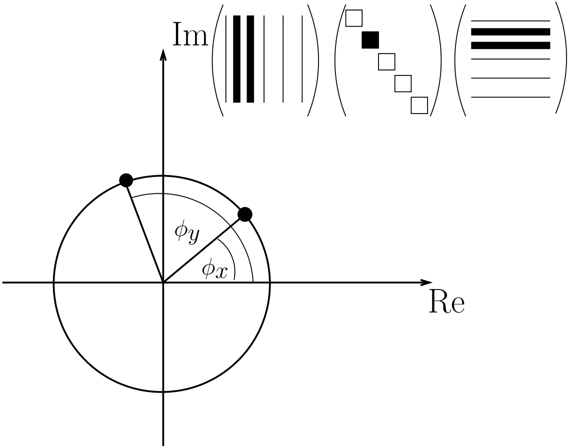
Any two **orthogonal, commuting** transformations differ only with respect to the **rotation angles in the eigenpaces**.

- So to *apply* a transformation you can equivalently perform a set of independent two-D rotations.



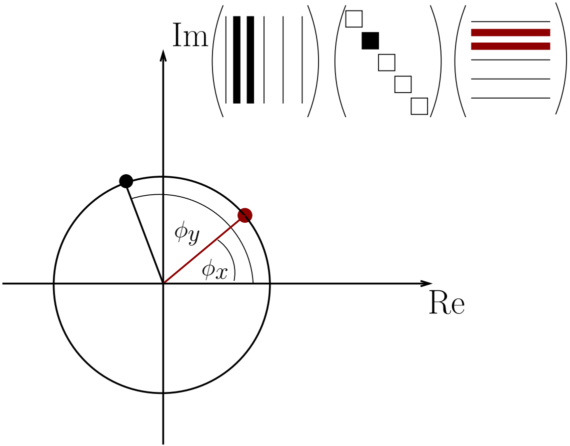
- To *infer* the transformation, given two images $\boldsymbol{x}$ and $\boldsymbol{y}$: Project $\boldsymbol{x}$ and $\boldsymbol{y}$ onto the eigenvectors, then compute the rotation angles!

Extracting subspace rotations, naive approach



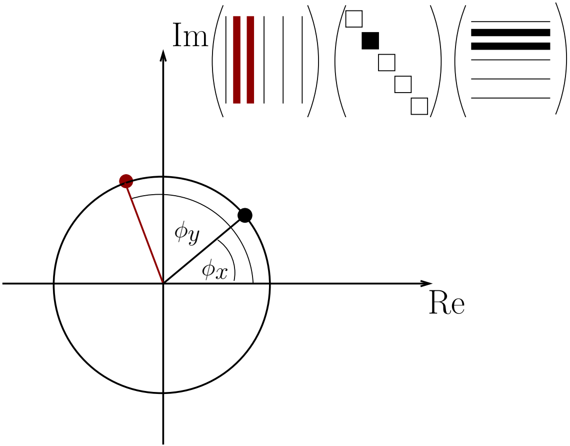
- In each subspace:
- Normalize the 2-D projections to unit norm, then read off the angle between them.

Extracting subspace rotations, naive approach



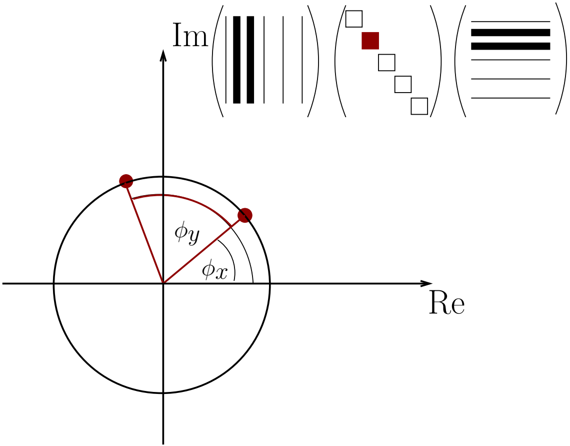
- In each subspace:
- Normalize the 2-D projections to unit norm, then read off the angle between them.

Extracting subspace rotations, naive approach



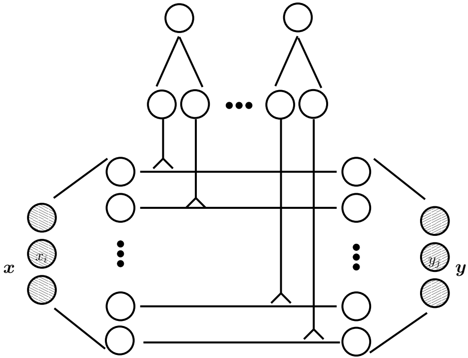
- In each subspace:
- Normalize the 2-D projections to unit norm, then read off the angle between them.

Extracting subspace rotations, naive approach



- In each subspace:
- Normalize the 2-D projections to unit norm, then read off the angle between them.

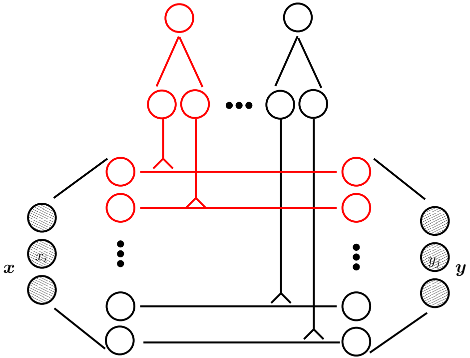
Extracting subspace rotations, naive approach



Extracting rotations by computing angles

- To read off the angle, compute the **inner product**:
- Compute the sum over products of filter responses.

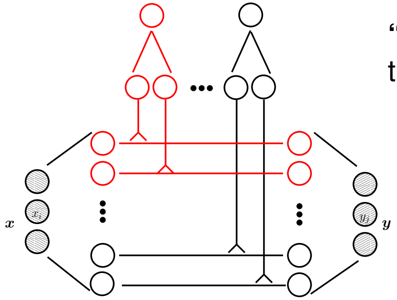
Extracting subspace rotations, naive approach



Extracting rotations by computing angles

- To read off the angle, compute the **inner product**:
- Compute the sum over products of filter responses.

Extracting subspace rotations, naive approach



"cos(angle) == inner product" is the trigonometric identity:

$$\begin{aligned} & \cos(\phi_y - \phi_x) \\ = \; & \cos\phi_y \cos\phi_x + \sin\phi_y \sin\phi_x \\ = \; & ({V_{\cdot 1}}^{\mathrm{T}} \boldsymbol{y})({V_{\cdot 1}}^{\mathrm{T}} \boldsymbol{x}) + ({V_{\cdot 2}}^{\mathrm{T}} \boldsymbol{y})({V_{\cdot 2}}^{\mathrm{T}} \boldsymbol{x}) \end{aligned}$$

Extracting rotations by computing angles

- To read off the angle, compute the **inner product**:
- Compute the sum over products of filter responses.

The aperture problem

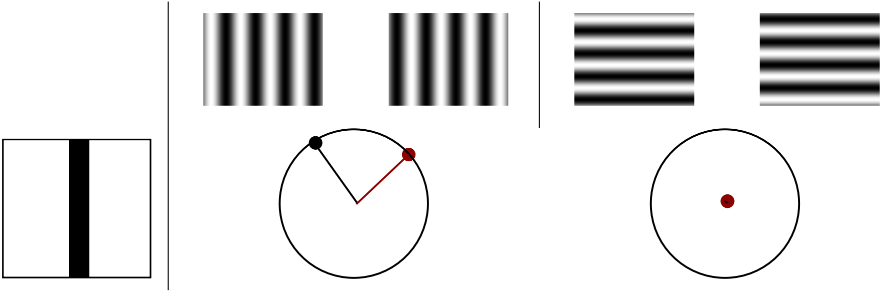
The aperture problem

- **Not all images are represented equally well in each subspace.**

The aperture problem

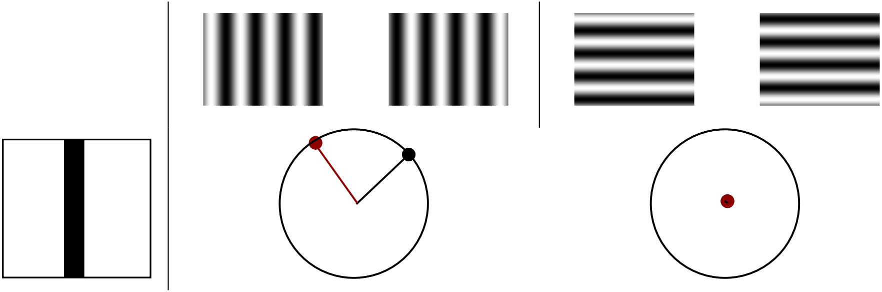
The aperture problem

- **Not all images are represented equally well in each subspace.**



The aperture problem

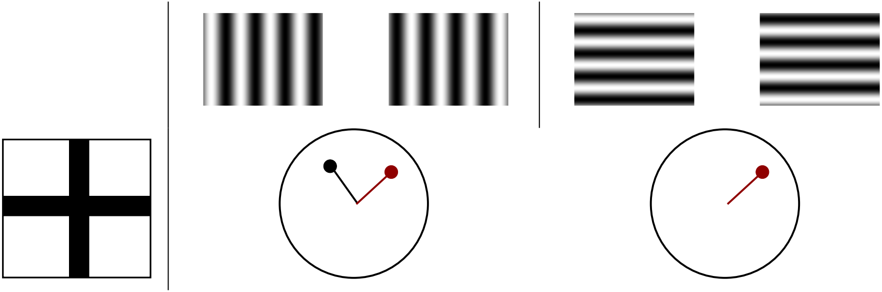
- Not all images are represented equally well in each subspace.



The aperture problem

The aperture problem

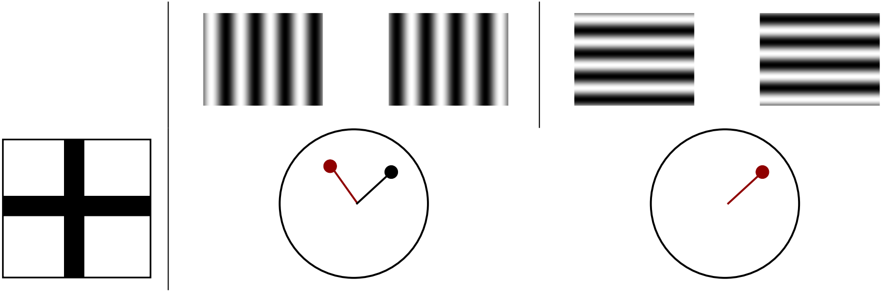
- Not all images are represented equally well in each subspace.



The aperture problem

The aperture problem

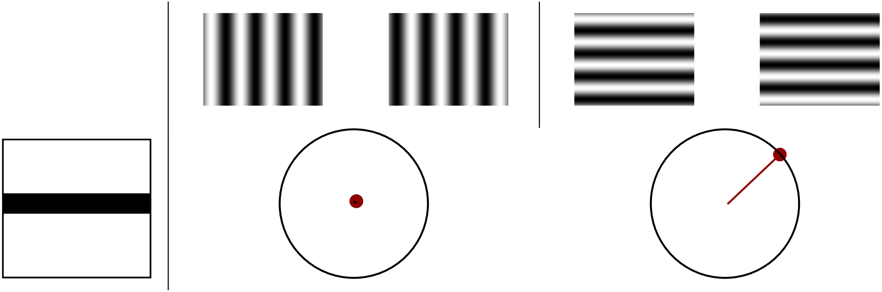
- Not all images are represented equally well in each subspace.



The aperture problem

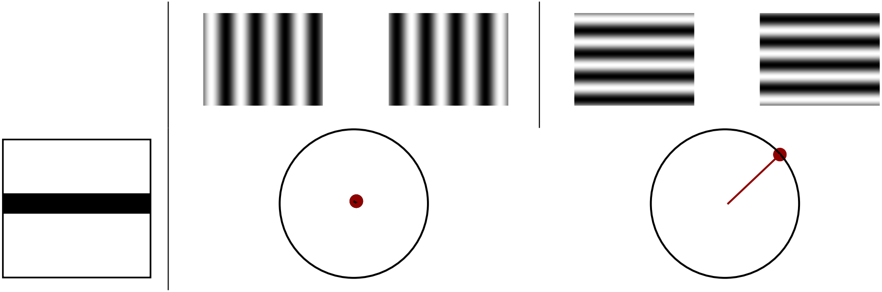
The aperture problem

- Not all images are represented equally well in each subspace.

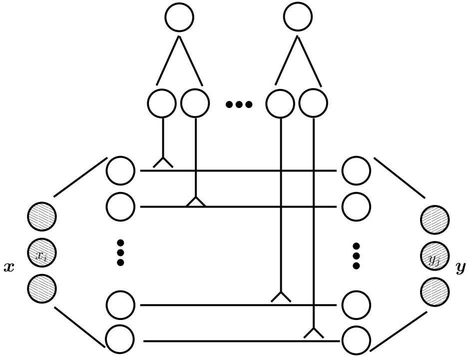


The aperture problem

- Not all images are represented equally well in each subspace.



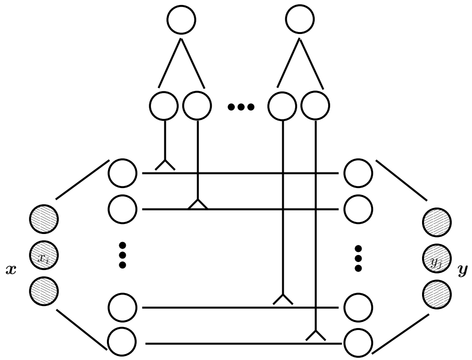
Subspace rotation detectors



How can we get a code that encodes both the presence and our uncertainty about subspace rotations given two images?

- Idea: **Absorb rotations into eigenvectors.**
- This allows us to turn hiddens into rotation *detectors*:

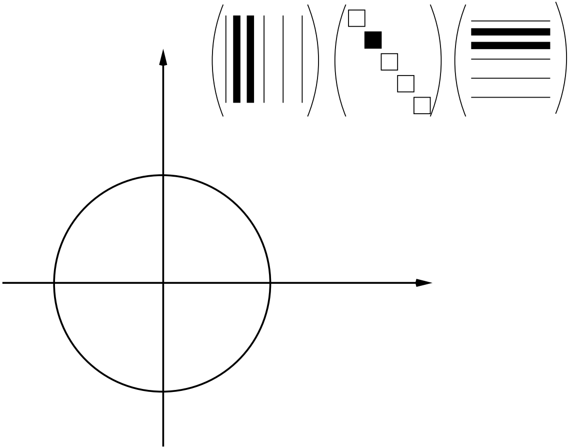
Subspace rotation detectors



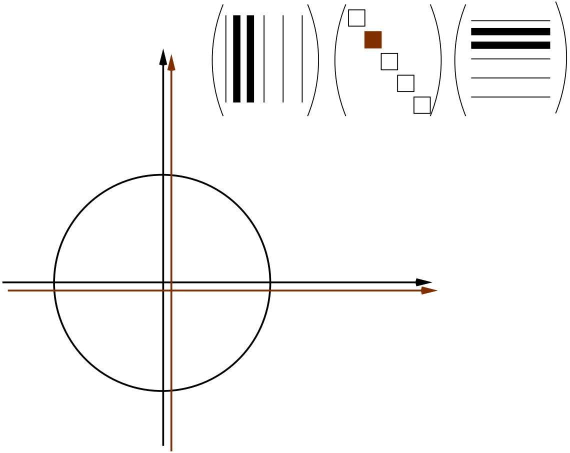
How can we get a code that encodes both the presence and our uncertainty about subspace rotations given two images?

- Idea: **Absorb rotations into eigenvectors.**
- This allows us to turn hiddens into rotation *detectors*:

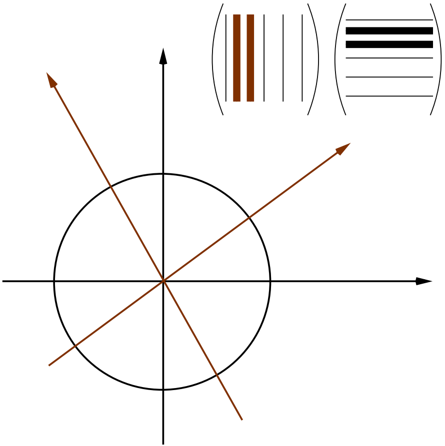
Subspace rotation detectors



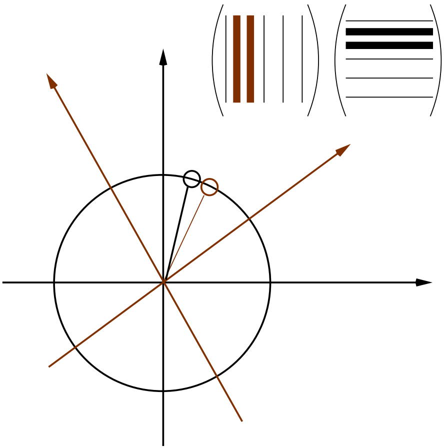
Subspace rotation detectors



Subspace rotation detectors

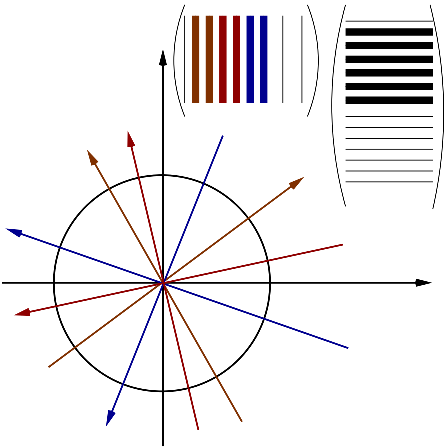


Subspace rotation detectors



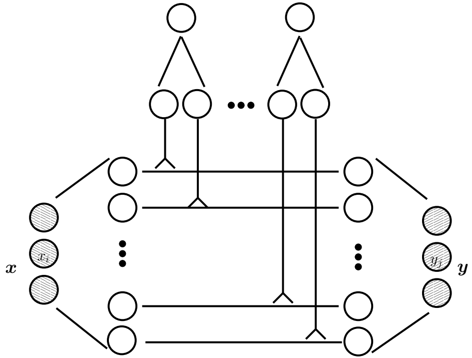
- Inner product is large, when the image transformation matches the absorbed eigenvalue.

Subspace rotation detectors



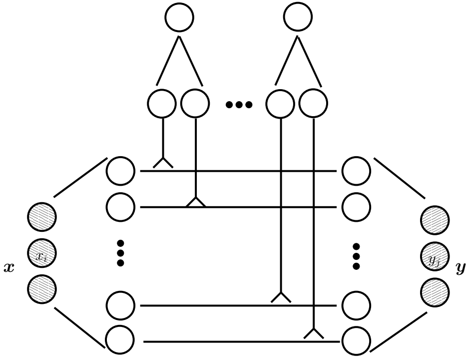
- Idea: For each subspace, use a **panel of mapping units**, each tuned to some angle, θ_i.

Subspace rotation detectors



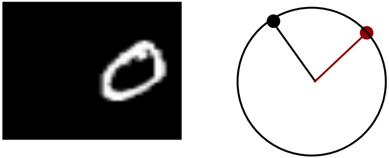
- Transformations encoded in a **population code**.
- A mapping unit is *conservative*: It fires only if a transform is present *and* if it is visible in the image pair.

Subspace rotation detector graphical model

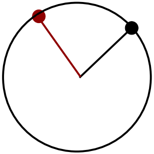


- Most transformations affect multiple subspaces.
- Hiddens should be independent of image content.

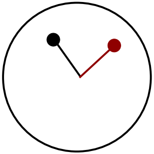
The aperture problem



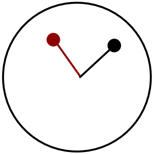
The aperture problem



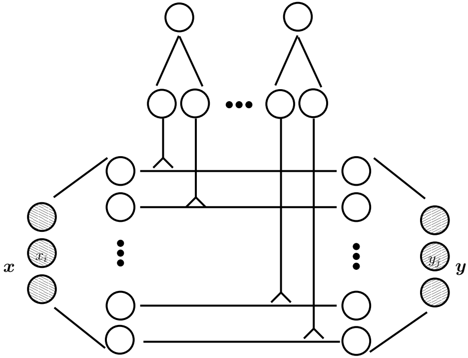
The aperture problem



The aperture problem

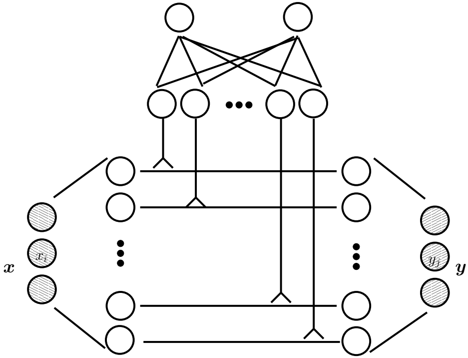


Subspace rotation detector graphical model



- Most transformations affect multiple subspaces.
- Hiddens should be independent of image content.

Subspace rotation detector graphical model

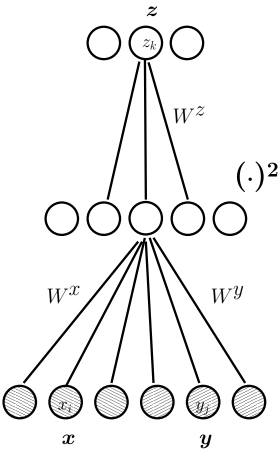


- $\rightarrow$ Let hiddens pool within *and* pool across subspaces.
- This is exactly the factored bilinear model.

Energy models

Square pooling:

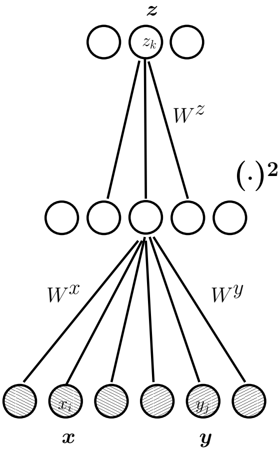
- Another way to learn matched filters is **square pooling** (on concatenation):
  - ASSOM (Kohonen, 1996)
  - ISA (Hyvarinen, 2000)
  - Product of T-distributions (Osindero et al., 2006)
  - (Karklin, Lewicki; 2008)
  - cRBM (Ranzato et al., 2009)
- Often, W^z is constrained so each hidden sees only a few squared inputs. That way hiddens can be thought of as encoding **subspace** norms.



Energy models

Square pooling:

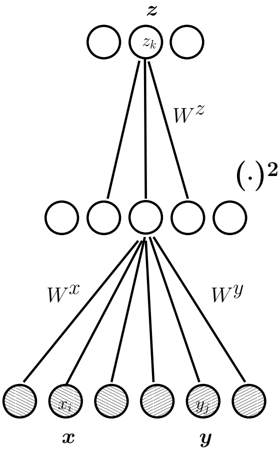
- Why is square pooling the same?
- The activity that a hidden unit gets is:
  $\sum_f w^z_{kf}\left({W^x_{\cdot f}}^{\mathrm{T}}\boldsymbol{x} + {W^y_{\cdot f}}^{\mathrm{T}}\boldsymbol{y}\right)^2$
  $= \sum_f w^z_{kf}\big(2({W^x_{\cdot f}}^{\mathrm{T}}\boldsymbol{x})({W^y_{\cdot f}}^{\mathrm{T}}\boldsymbol{y})$
  $+({W^x_{\cdot f}}^{\mathrm{T}}\boldsymbol{x})^2 + ({W^y_{\cdot f}}^{\mathrm{T}}\boldsymbol{y})^2\big)$
- Inference just adds square terms.
- This may make the rotation detectors more conservative. Otherwise inference is the same!



Energy models

Square pooling:

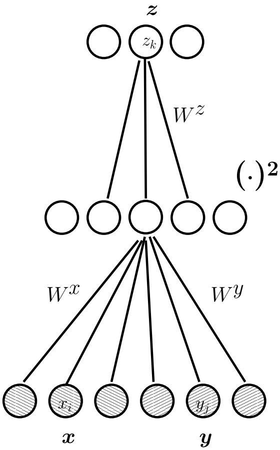
- Why is square pooling the same?
- The activity that a hidden unit gets is:
$\sum_f w^z_{kf}\left({W^x_{\cdot f}}^{\mathrm{T}}\boldsymbol{x} + {W^y_{\cdot f}}^{\mathrm{T}}\boldsymbol{y}\right)^2$
$= \sum_f w^z_{kf}\left(2({W^x_{\cdot f}}^{\mathrm{T}}\boldsymbol{x})({W^y_{\cdot f}}^{\mathrm{T}}\boldsymbol{y})\right.$
$\left.+({W^x_{\cdot f}}^{\mathrm{T}}\boldsymbol{x})^2 + ({W^y_{\cdot f}}^{\mathrm{T}}\boldsymbol{y})^2\right)$
- Inference just adds square terms.
- This may make the rotation detectors more conservative. Otherwise inference is the same!



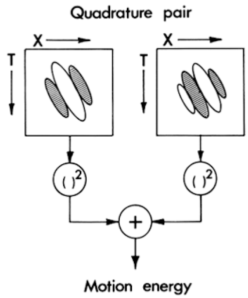
Energy models

Square pooling:

- Learning typically more difficult than with factored gated feature learning.
- Example ISA: Gradient-based, while enforcing ${W^{xy}}^{\mathrm{T}} W^{xy} = I$ after every gradient step (eigen-decomposition).



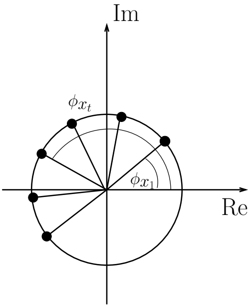
Energy models



The energy model

- (Adelson and Bergen, 1985): Motion
- (Ozhawa, DeAngelis, Freeman; 1990): Disparity
- Equivalence to cross-correlation: See, for example, (Fleet et al.; 1994).

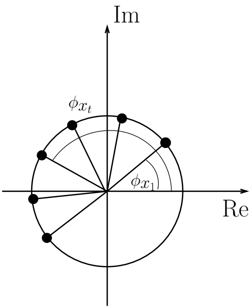
Learning energy models on movies



- What happens when we train energy models on movies?
- Hiddens receive all pairs of products, so they detect the **repeated application of the same eigenvalue**:

$$\Big(\sum_s \boldsymbol{v}^{s\mathrm{T}}\boldsymbol{x}_s\Big)^2 = \sum_s \Big(\boldsymbol{v}^{s\mathrm{T}}\boldsymbol{x}_s\Big)^2 + \sum_{st} \Big(\boldsymbol{v}^{s\mathrm{T}}\boldsymbol{x}_s\Big) \cdot \Big(\boldsymbol{v}^{t\mathrm{T}}\boldsymbol{x}_t\Big)$$

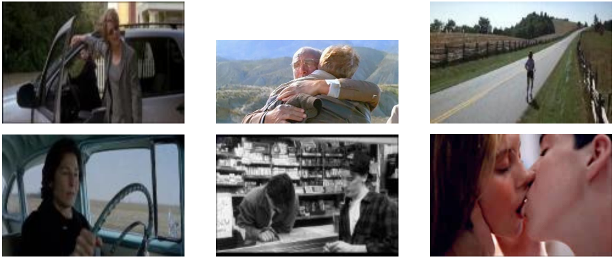
Learning energy models on movies



- What happens when we train energy models on movies?
- Hiddens receive all pairs of products, so they detect the **repeated application of the same eigenvalue**:

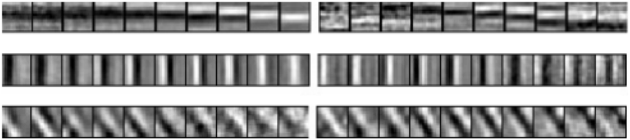
$$\Big(\sum_s \boldsymbol{v}^{s\mathrm{T}}\boldsymbol{x}_s\Big)^2 = \sum_s \Big(\boldsymbol{v}^{s\mathrm{T}}\boldsymbol{x}_s\Big)^2 + \sum_{st}\Big(\boldsymbol{v}^{s\mathrm{T}}\boldsymbol{x}_s\Big)\cdot\Big(\boldsymbol{v}^{t\mathrm{T}}\boldsymbol{x}_t\Big)$$

Action recognition

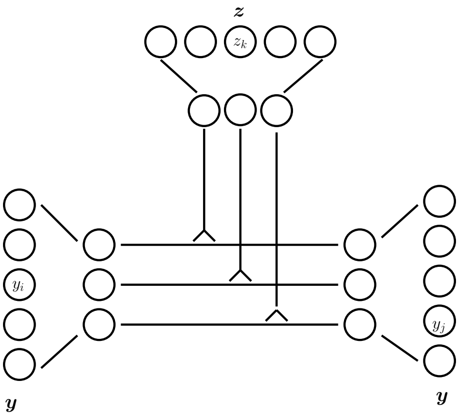


(Hollywood 2)

- Convolutional GBM (Taylor et al., 2010)
- hierarchical ISA (Le, et al., 2011)

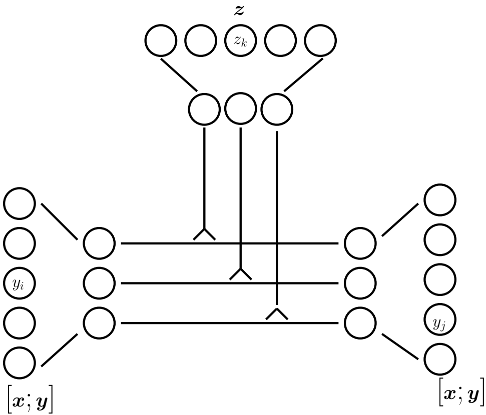


Training energy models via gating



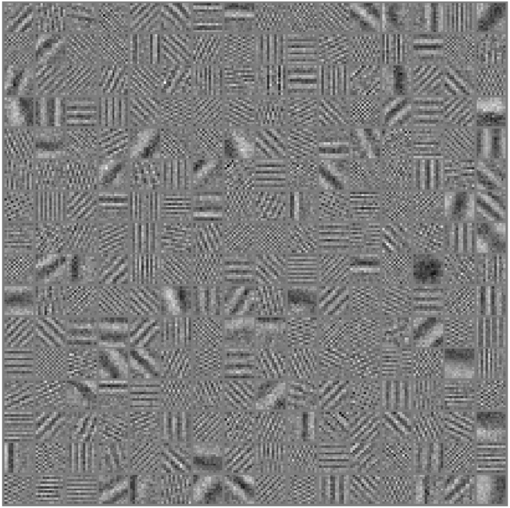
- We can use square pooling to simulate gating.
- But we can also use gating to simulate squaring:
- Plug in *the same* data left and right and tie left and right filters.

Training energy models via gating

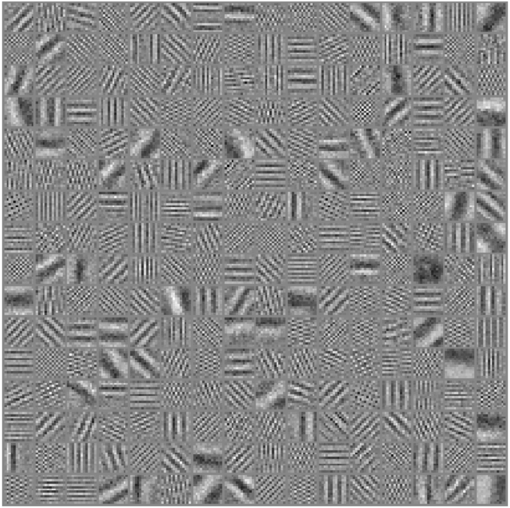


- To train the model on videos, train an energy model on concatenated frames:
- Use **gating via energy via gating**!

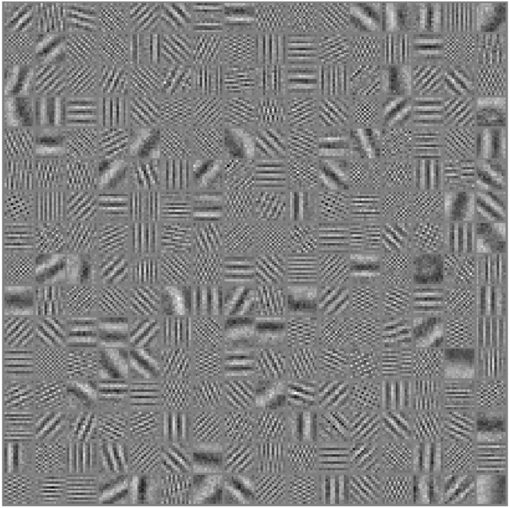
A covariance encoder trained on movies



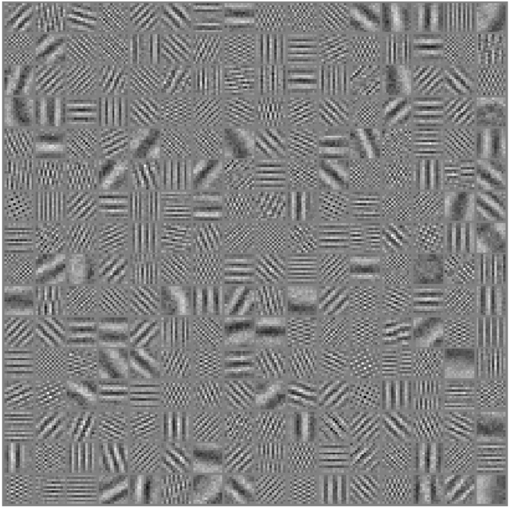
A covariance encoder trained on movies



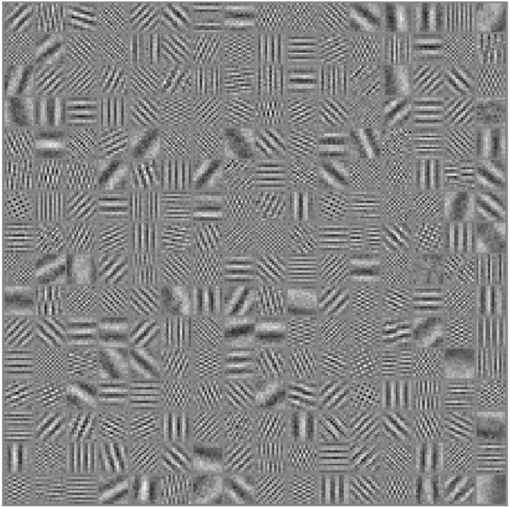
A covariance encoder trained on movies



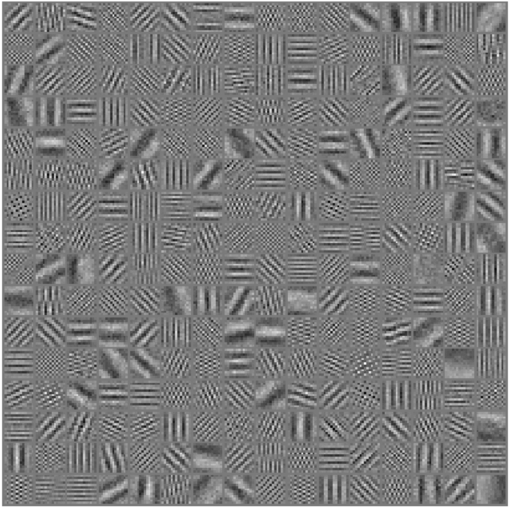
A covariance encoder trained on movies



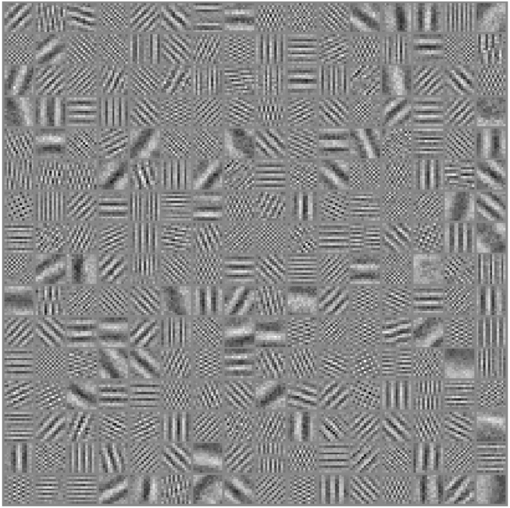
A covariance encoder trained on movies



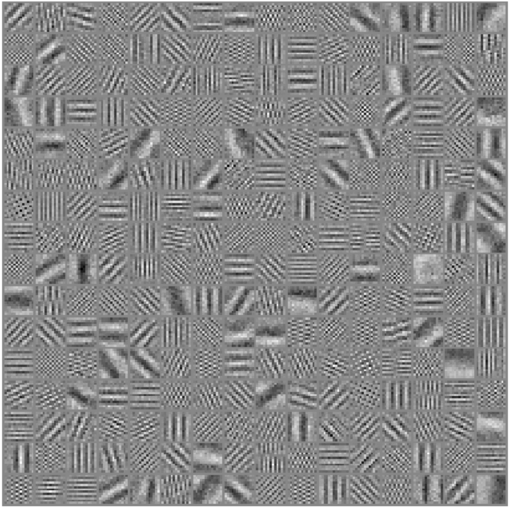
A covariance encoder trained on movies



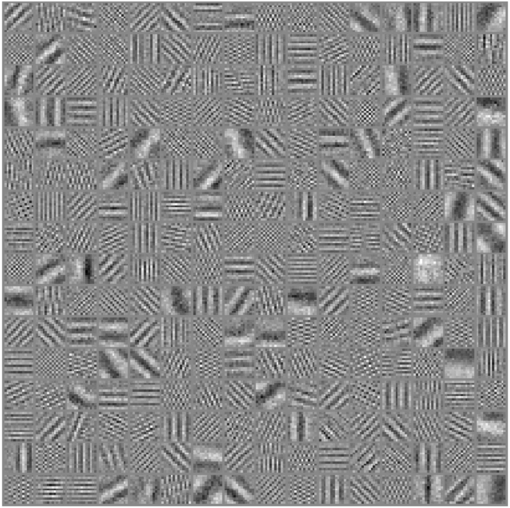
A covariance encoder trained on movies



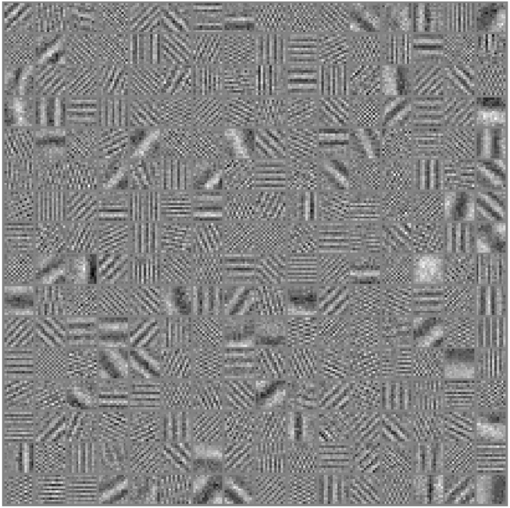
A covariance encoder trained on movies



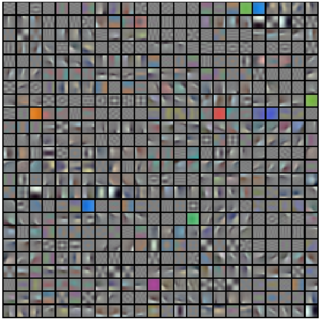
A covariance encoder trained on movies

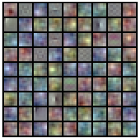


A covariance encoder trained on movies



"Mean-covariance" encoder on single images





#factors / #covar-hiddens / #mean-hiddens	Model	Perf (%)
225 / 225 / 0	cRBM	63.6
225 / 225 / 0	cAE	64.5
900 / 225 / 0	cRBM	64.7
900 / 225 / 0	cAE	65.4
576 / 144 / 81	mcRBM	68.2
576 / 144 / 81	mcAE	67.7

A bag of tricks

Tricks for learning:

- Contrast filters after each gradient step.
- Contrast-normalize input data.
- Optionally, whiten input data.
- Connect top-level hiddens *locally* to the factors.
- Probably even better: make them *locally overlapping* ("Topographic ICA").
- Fast learning: large data-sets essential (use GPU's...).

A bag of tricks

Tricks for learning:

- Contrast filters after each gradient step.
- Contrast-normalize input data.
- Optionally, whiten input data.
- Connect top-level hiddens *locally* to the factors.
- Probably even better: make them *locally overlapping* ("Topographic ICA").
- Fast learning: large data-sets essential (use GPU's...).

A bag of tricks

Tricks for learning:

- Contrast filters after each gradient step.
- Contrast-normalize input data.
- Optionally, whiten input data.
- Connect top-level hiddens *locally* to the factors.
- Probably even better: make them *locally overlapping* ("Topographic ICA").
- Fast learning: large data-sets essential (use GPU's...).

A bag of tricks

Tricks for learning:

- Contrast filters after each gradient step.
- Contrast-normalize input data.
- Optionally, whiten input data.
- Connect top-level hiddens *locally* to the factors.
- Probably even better: make them *locally overlapping* ("Topographic ICA").
- Fast learning: large data-sets essential (use GPU's...).

A bag of tricks

Tricks for learning:

- Contrast filters after each gradient step.
- Contrast-normalize input data.
- Optionally, whiten input data.
- Connect top-level hiddens *locally* to the factors.
- Probably even better: make them *locally overlapping* ("Topographic ICA").
- Fast learning: large data-sets essential (use GPU's...).

A bag of tricks

Tricks for learning:

- Contrast filters after each gradient step.
- Contrast-normalize input data.
- Optionally, whiten input data.
- Connect top-level hiddens *locally* to the factors.
- Probably even better: make them *locally overlapping* ("Topographic ICA").
- Fast learning: large data-sets essential (use GPU's...).

Learning cross-correlation and energy models

Take-home message, factored model

To learn about transformation, let hidden units pool over products of filter responses.

Outline

Outline

Analogy making

$$A : A' \quad :: \quad B : ?$$

Analogy making

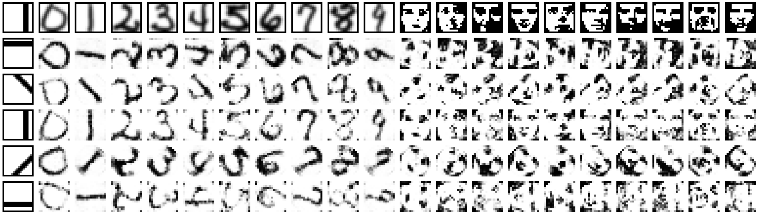
1. Infer transformation from *source* images $x_{\mathrm{source}}, y_{\mathrm{source}}$:

$$z(x_{\mathrm{source}}, y_{\mathrm{source}})$$

2. Apply the transformation to *target* image x_{target}:

$$y(z, x_{\mathrm{target}})$$

Analogy making

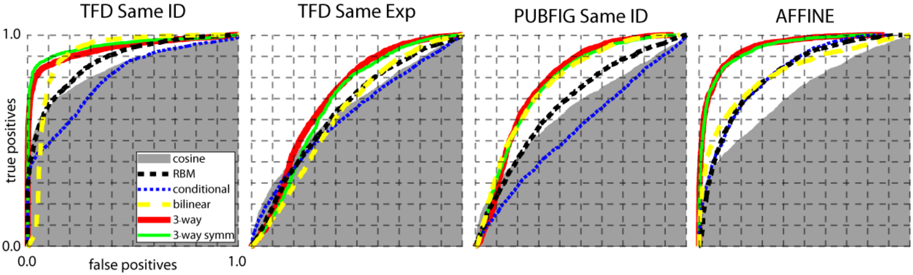


Filters learned from transforming faces

- Filters learned from faces:

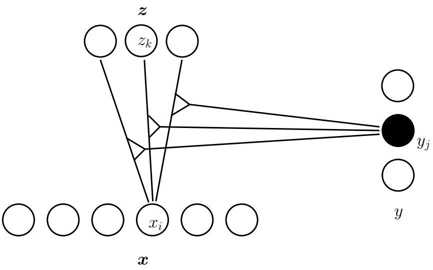
Metric learning and analogy making

- Learning a gated Boltzmann machine on changing facial expressions.
- (Susskind, et al., 2011)
- **Joint density training** allows for matching.



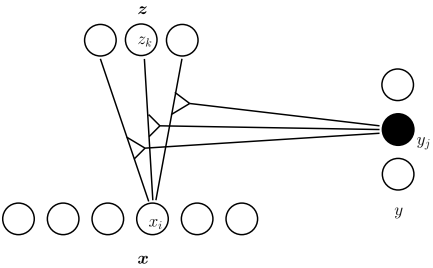
Model/Task	TFD ID	TFD Exp	PUBFIG ID	AFFINE
cosine	0.848	0.663	0.649	0.721
RBM	0.869	0.656	0.647	0.799
conditional	0.805	0.634	0.557	0.825
bilinear	0.905	0.637	**0.774**	0.812
3-way	0.932	**0.705**	0.771	0.930
3-way symm	**0.951**	0.695	0.762	**0.931**

Bi-linear classification



- A special case of a gated Boltzmann machine:
- Replace the output-image by a one-hot-encoded **class-label**.
- This is a classifier, where each *label can blend in it's own model*.

Bi-linear classification

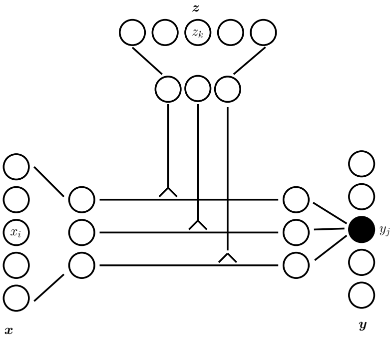


- Marginalization is tractable in closed form

$$
\begin{aligned}
p(y|\boldsymbol{x}) = \sum_{\boldsymbol{z}} p(y, \boldsymbol{z}|\boldsymbol{x}) \quad &\propto \quad \sum_{\boldsymbol{z}} \exp(\boldsymbol{x}^{\mathrm{t}} w_y \boldsymbol{z}) = \sum_{\boldsymbol{z}} \exp(\sum_{ik} w_{yik} x_i h_k) \\
&= \quad \prod_k (1 + \exp(\sum_i w_{yik} x_i))
\end{aligned}
$$

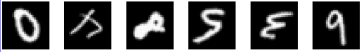
- This is a mixture of 2^K logistic regressors (Nair, 2008), (Memisevic, et al.; 2010), (Warrell et al.; 2010)

Bi-linear classification

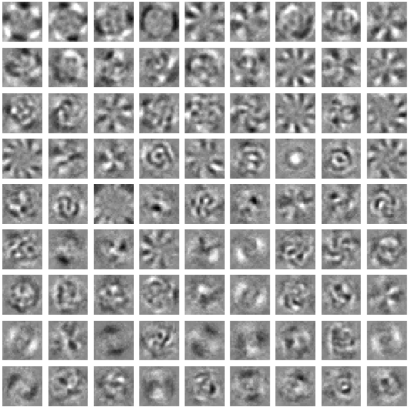


- Factorization allows classes to share features.
- The activity of a factor, f, given class j, is now exactly equal to the parameter value w^y_{jf}.
- Thus the weights can be thought of as the responses of **virtual class-templates**. (Zach 2010, pers. comm.)

Rotated digit classification



- Data from the “deep learning-challenge” [Larochelle et al., 2007].
- Learned rotation-invariant filters:

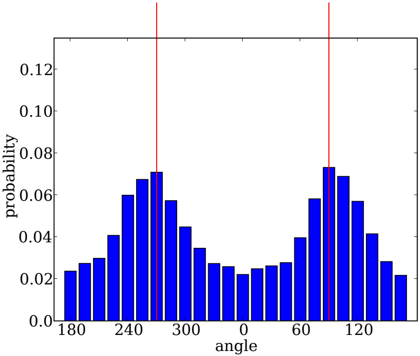


Bi-linear classification

- Deep Learning challenge (Larochelle et al., 2008).

	SVMs		NNet	RBM	DEEP		GSM	
dataset/model:	SVMRBF	SVMPOL	NNet	DBN1	DBN3	SAA3	**GSM**	(unfact)
rectangles	2.15	2.15	7.16	4.71	2.60	2.41	0.83	(0.56)
rect.-images	24.04	24.05	33.20	23.69	22.50	24.05	22.51	(23.17)
mnistplain	3.03	3.69	4.69	3.94	3.11	3.46	3.70	(3.98)
convexshapes	19.13	19.82	32.25	19.92	18.63	18.41	17.08	(21.03)
mnistbackrand	14.58	16.62	20.04	9.80	6.73	11.28	10.48	(11.89)
mnistbackimg	22.61	24.01	27.41	16.15	16.31	23.00	23.65	(22.07)
mnistrotbackimg	55.18	56.41	62.16	52.21	47.39	51.93	55.82	(55.16)
mnistrot	11.11	15.42	18.11	14.69	10.30	10.30	11.75	(16.15)

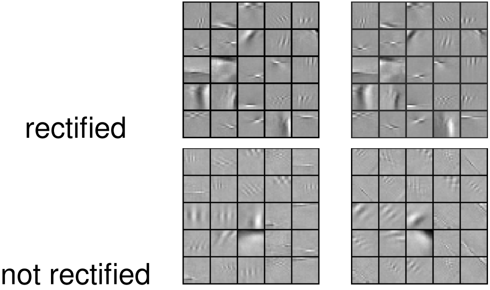
Transparent motion



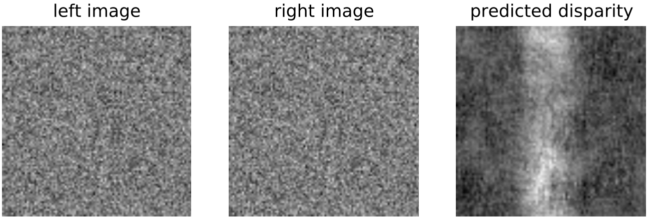
- Hidden variables make extracting multiple, simultaneous motions easy.
- When they fail, they do so in a similar way as humans:
- Better disrimination at large angles, averaging at very small angles, “motion repulsion”.
- (eg., Treue et al., 2000)

Depth as a latent variable

- Learning a dictionary for stereo:
- Generate left-right camera pairs with known disparities.
- *Predict* disparity from the hidden units.
- This gives rise to a three-layer network, that may be trained with Hebbian-like learning.

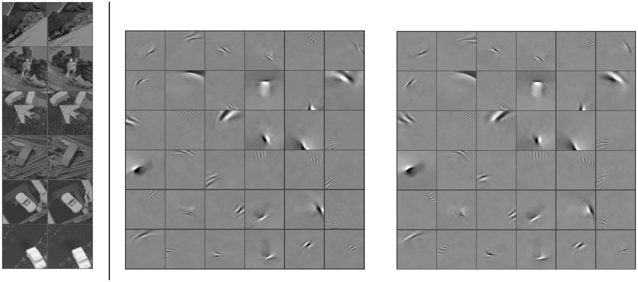


Hiddens learn to encode disparities



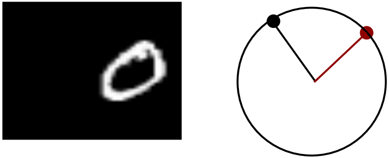
- Can use this to encode 3d-structure implicitly, for example, for multi-view recognition.

Norb stereo features

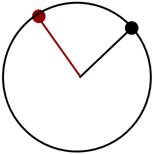


	NORB training subset:			NORB testset:		
RBMmon	RBMbin	cc	cc+bin	RBMbin	cc	cc+bin
73.65	60.43	34.85	31.48	63.28	38.91	36.80

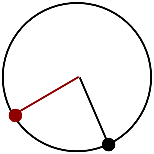
Transformations are transformation invariant



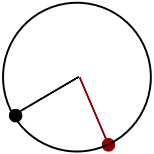
Transformations are transformation invariant



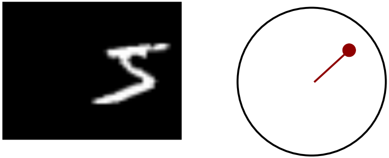
Transformations are transformation invariant



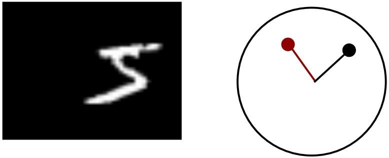
Transformations are transformation invariant



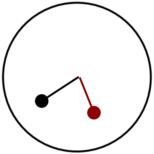
Transformations are transformation invariant



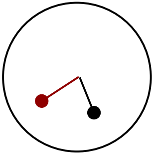
Transformations are transformation invariant



Transformations are transformation invariant



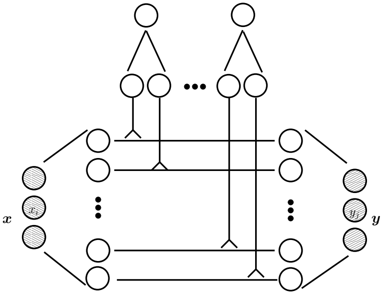
Transformations are transformation invariant



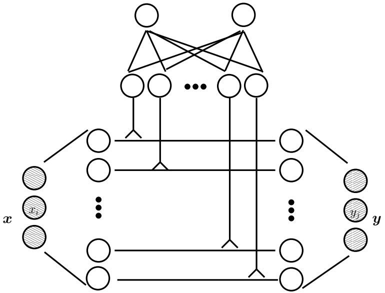
Transformations are transformation invariant

- We used across-subspace pooling to remove dependencies on image content.
- Each subspace itself is dependent on image content.

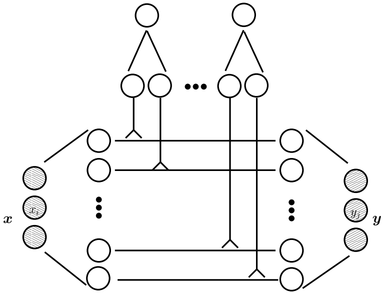
Harnessing the aperture problem



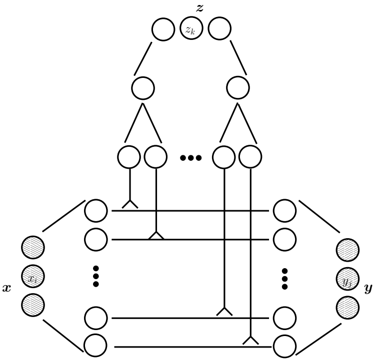
Harnessing the aperture problem



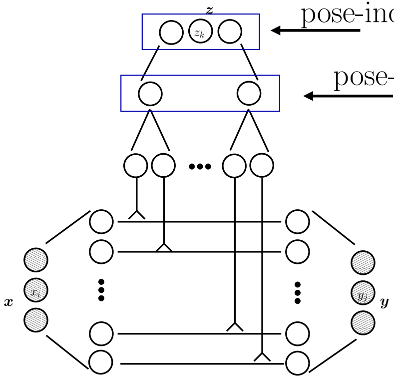
Harnessing the aperture problem



Harnessing the aperture problem



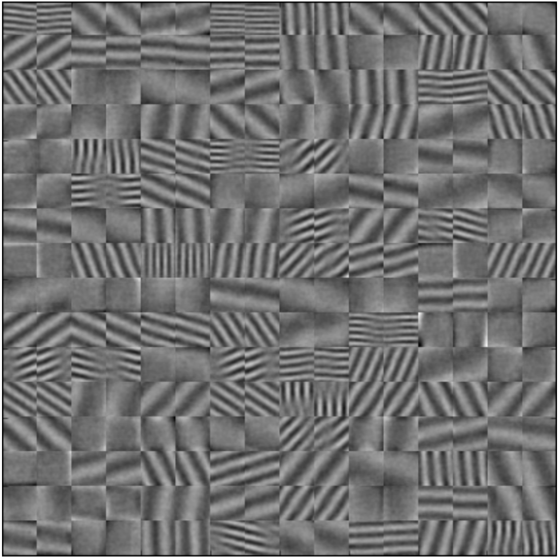
Harnessing the aperture problem



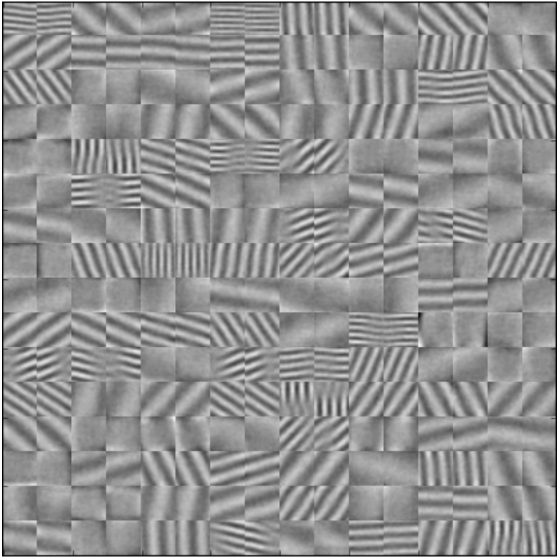
pose-independent, content-independent

pose-independent, content-dependent

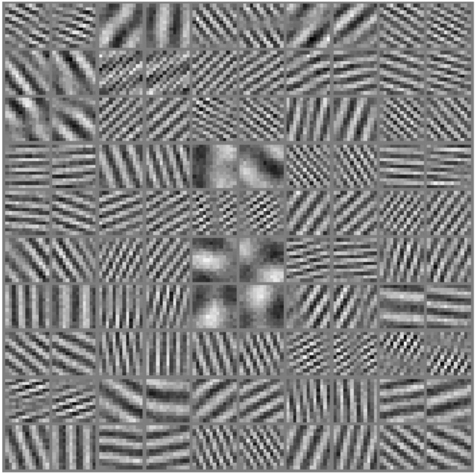
Learning quadrature features



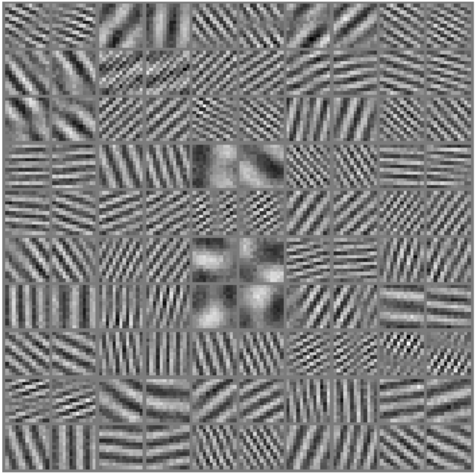
Learning quadrature features



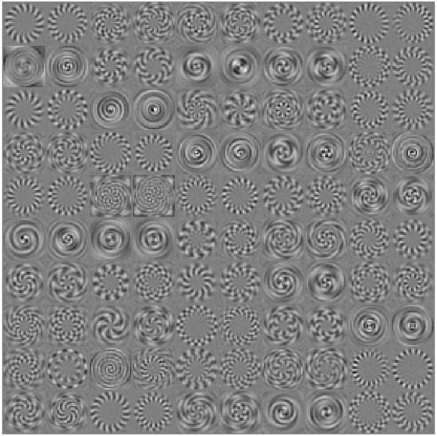
Learning quadrature features



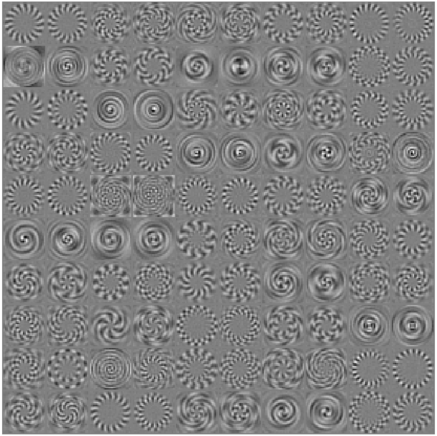
Learning quadrature features



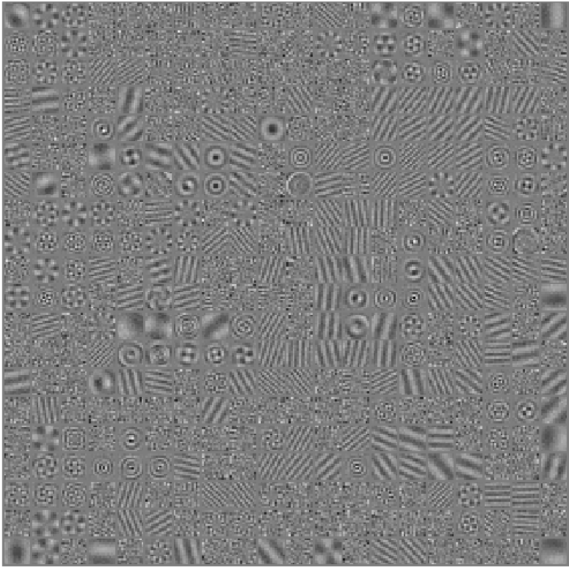
Rotation quadrature filters



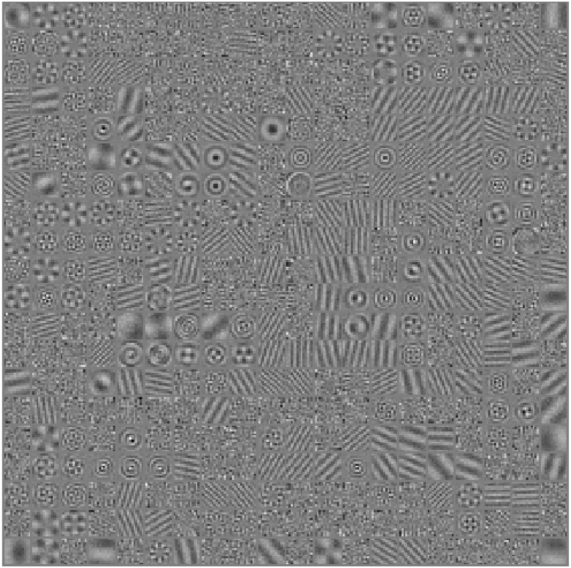
Rotation quadrature filters



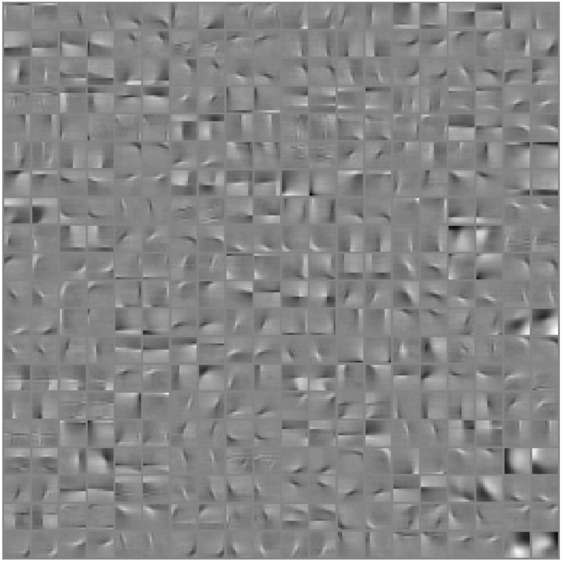
Mixed transformations



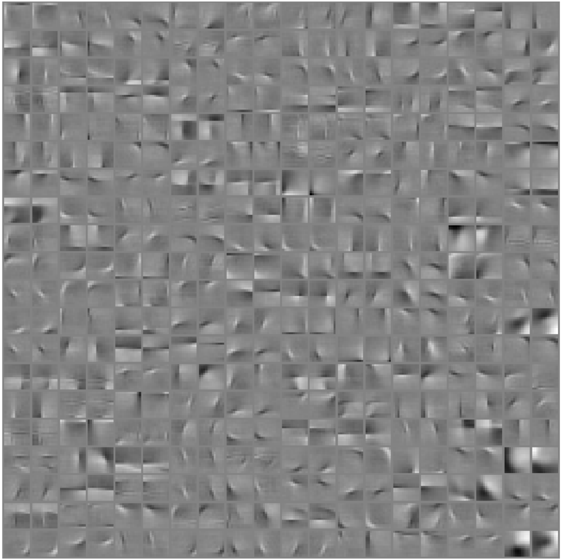
Mixed transformations



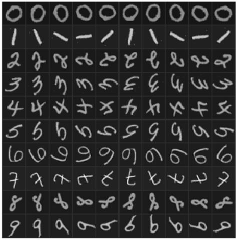
Quadrature features from natural video

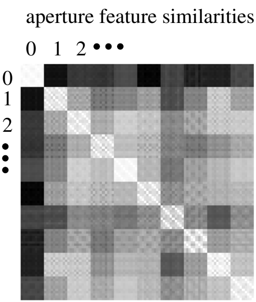


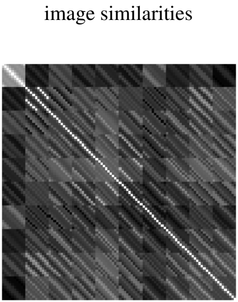
Quadrature features from natural video



Representing digits using rotation aperture features

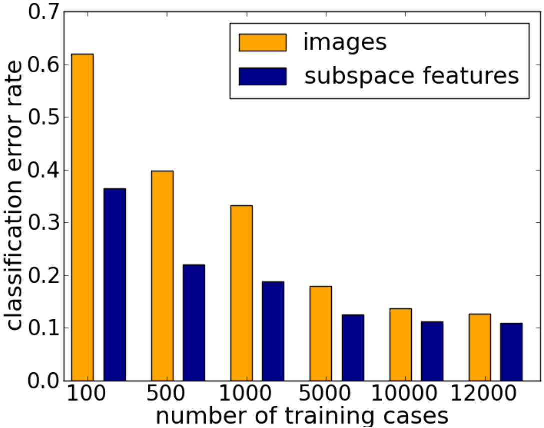






- Learn rotation features. Represent digits using aperture features.
- No video available? Fill video buffer with copies of the same image: Represent the non-transformation.

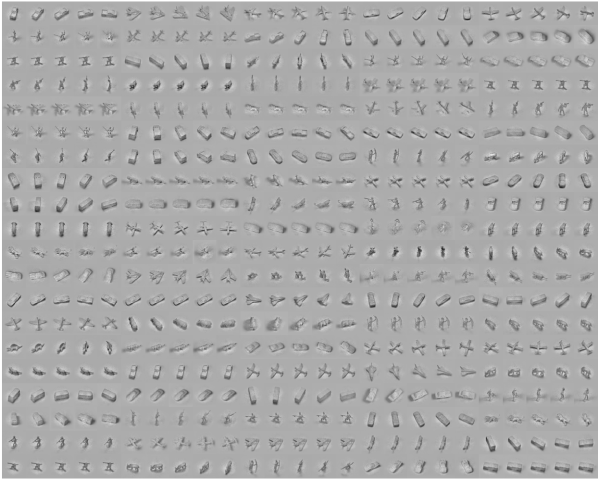
Rotated MNIST error rates



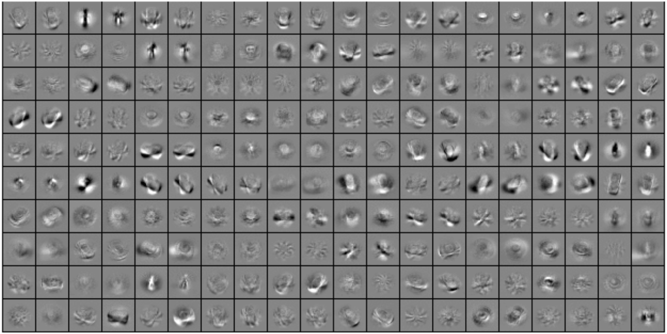
Video object features

- Humans do not recognize still images but videos of objects.
- The way in which an object changes can convey useful information about the object, including 3-D structure.
- $\rightarrow$ **Learn features from videos not still images.** See eg. (Lee and Soatto, 2011).

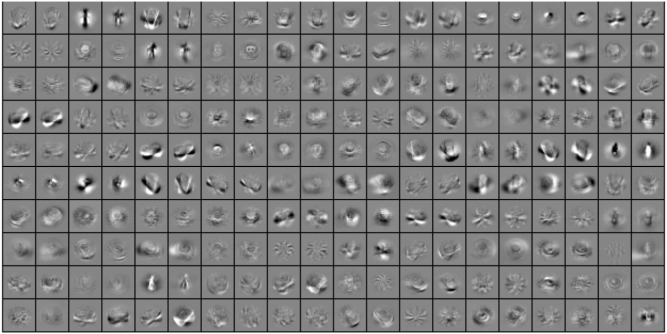
The “norbjects” video dataset



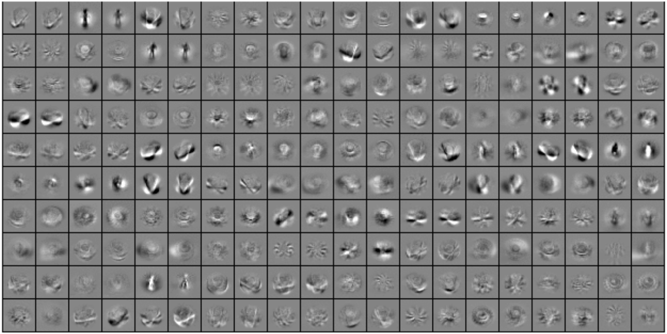
3-D rotation subspaces



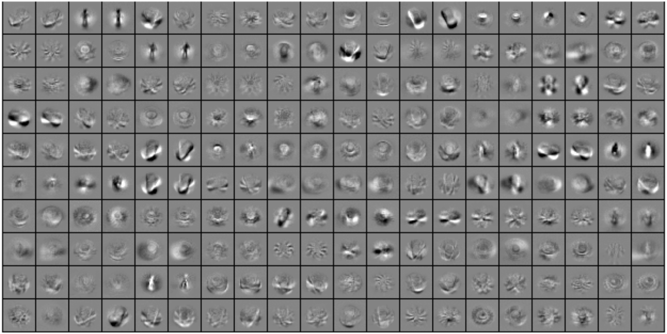
3-D rotation subspaces



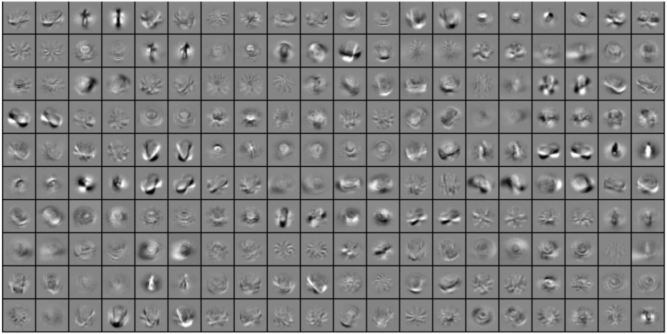
3-D rotation subspaces



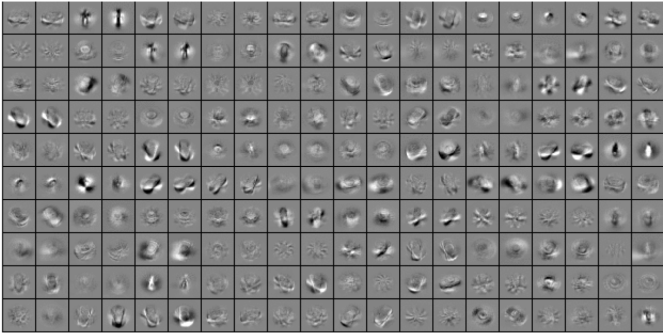
3-D rotation subspaces



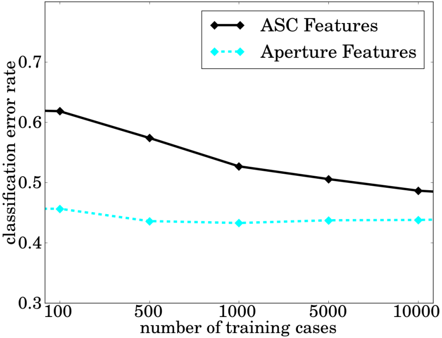
3-D rotation subspaces



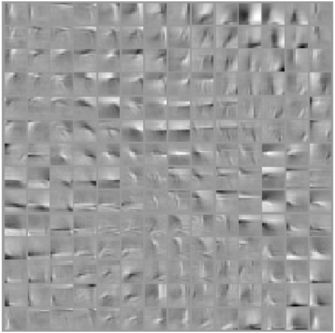
3-D rotation subspaces

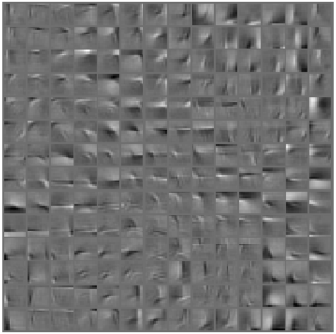


Classification with aperture features



Topographic representations emerge from *local gating*





Outline

Lessons

1) We can learn relations by letting hidden variables compute sums over products of filter responses.
2) Each hidden has to pool across multiple 2-d subspaces.
3) Learning requires contrast normalization + keeping the scales of filters roughly the same.
4) We can replace products with squares and vice versa.
5) Complex cell models approximately jointly diagonalize a set of commuting matrices.
6) Complex cells support selectivity not just invariance.
7) Energy model features can only represent the repeated application of the same transformation.
8) Transformations are transformation-invariant.

Conclusions

- *Learning* is a way to support simplicity and homogeneity of complex, intelligent systems.
- *Feature* learning even more so.
- *Relational* feature learning even more:

- Learning "verbs", not just "nouns", can help address more tasks with a single kind of model.
- This seems like a good reason to have complex cells.
- One reason, why looking for *correspondences* – across frames, across views, across modalities, etc. – is a common operation, is that mappings between modalities are often *one-to-many*.
- The theory provides a strong inductive bias for products and/or squaring non-linearities when building deep learning models.

Thank you

`http://www.cs.toronto.edu/~rfm/multiview-feature-learning-cvpr/index.html`