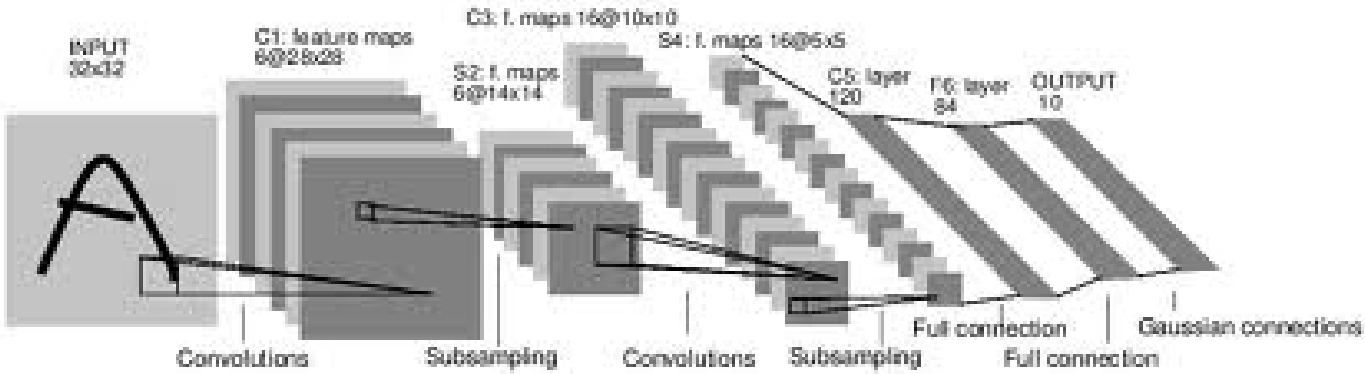
Representing relations

Roland Memisevic

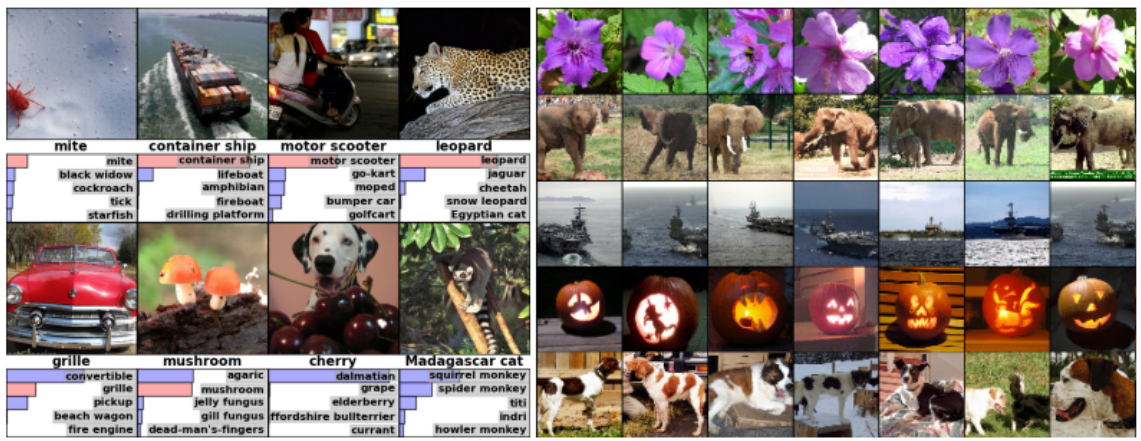
University of Montreal, LISA lab

April 16, 2014

What next?



Le Hammer



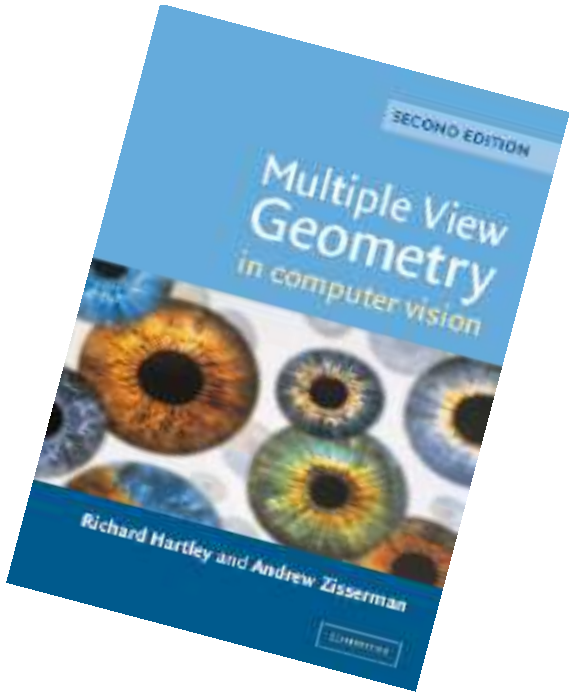
Deep learning of vision?

- Geometry, stereo, structure-from-motion, motion understanding, activity analysis, tracking, optic flow, modeling object relations, scene understanding, articulation, odometry, analogy, ...

Deep learning of vision?

- Geometry, stereo, structure-from-motion, motion understanding, activity analysis, tracking, optic flow, modeling object relations, scene understanding, articulation, odometry, analogy, ...

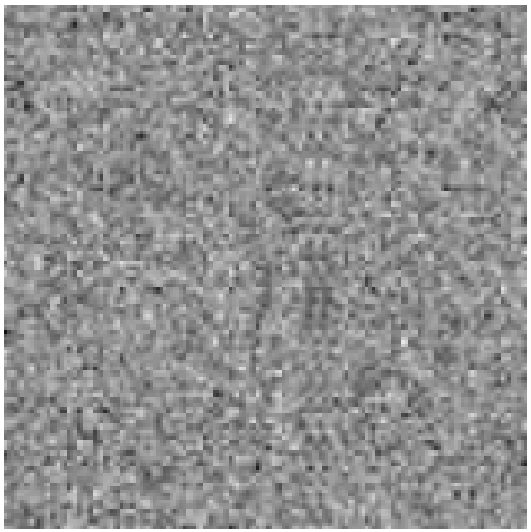
- Need to represent **relations**.

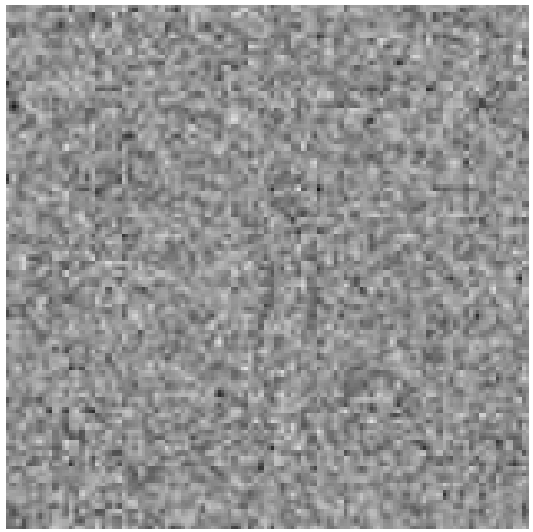


Some things are hard to infer from still images

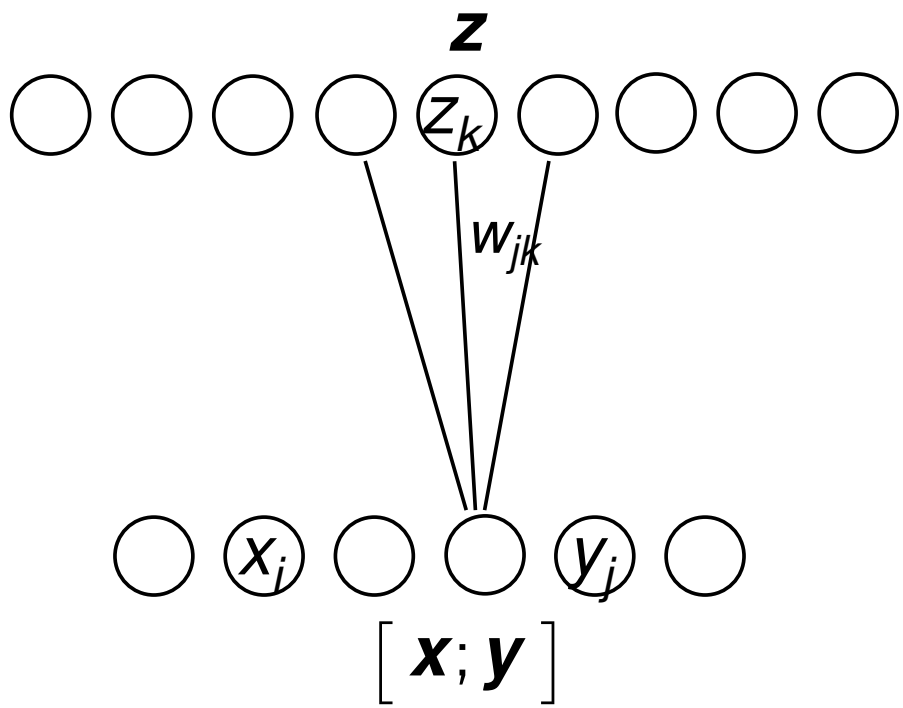
(Ayvaci, Soatto 2012)

Random dot stereograms



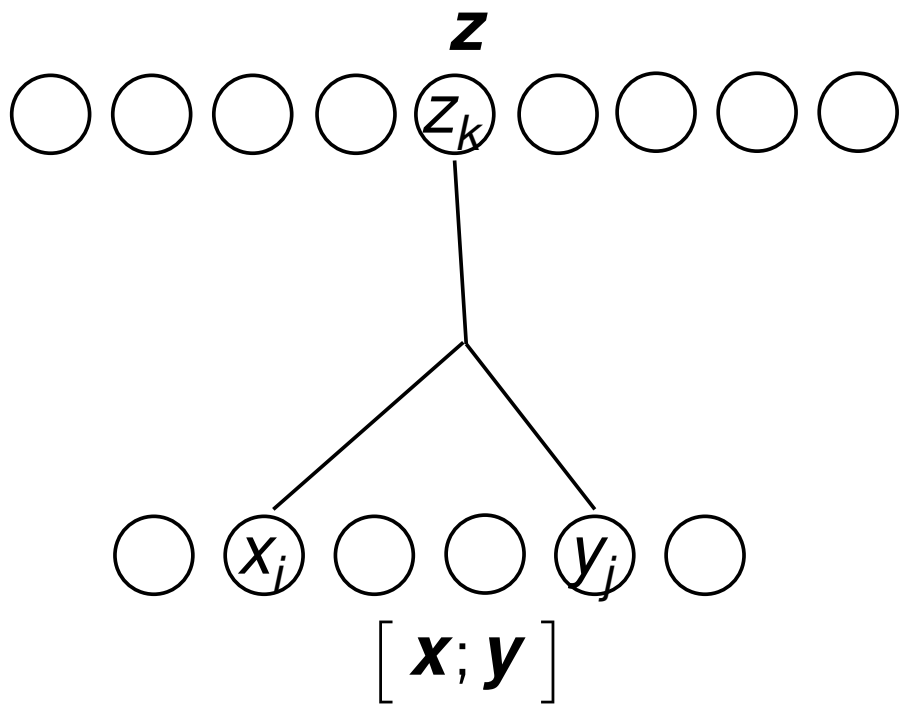


Learn relations by concatenating images?



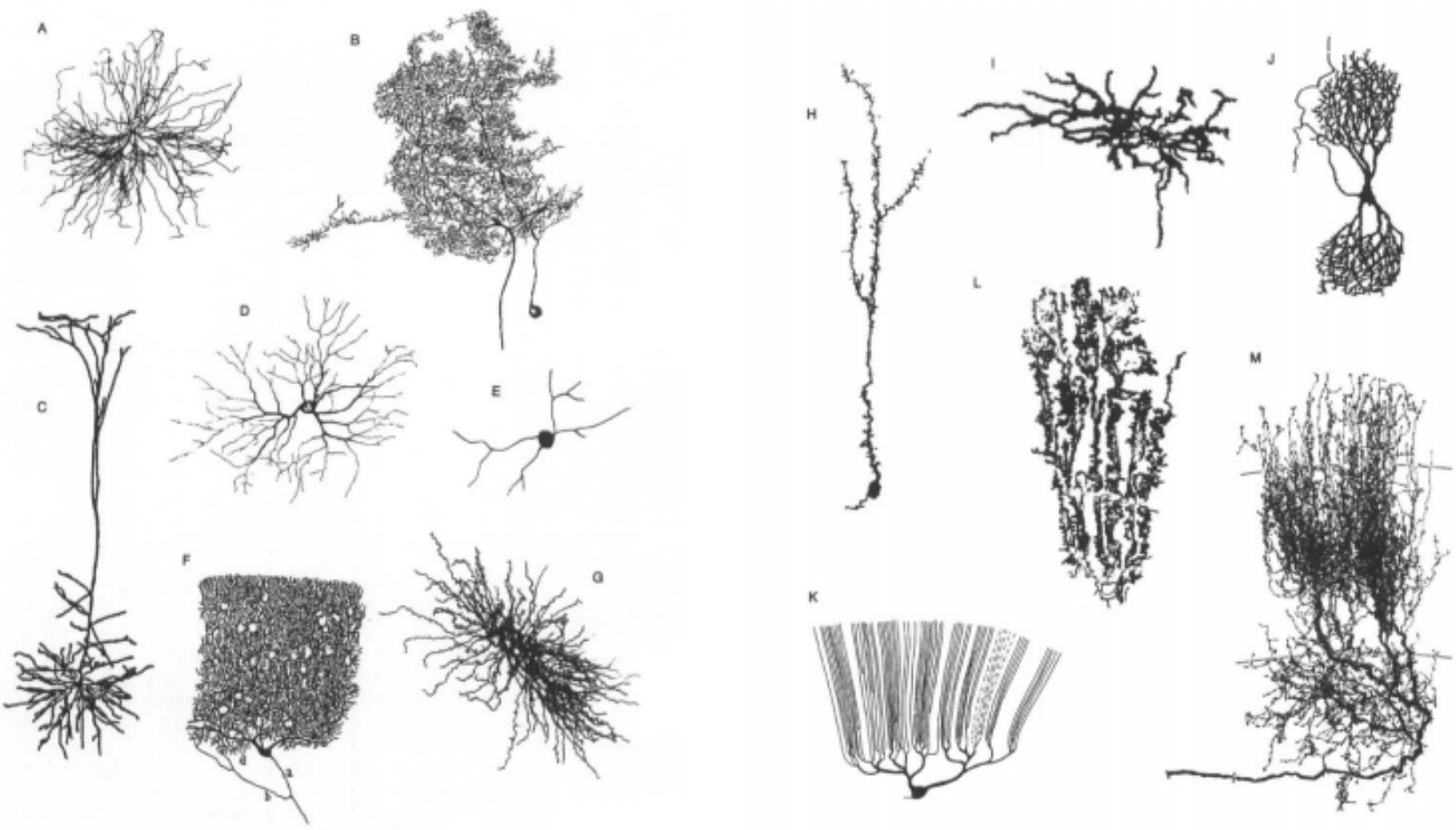
- Problem: This would make unit x_i conditionally independent of unit y_j given $\boldsymbol{z}$.

Learn relations by concatenating images?



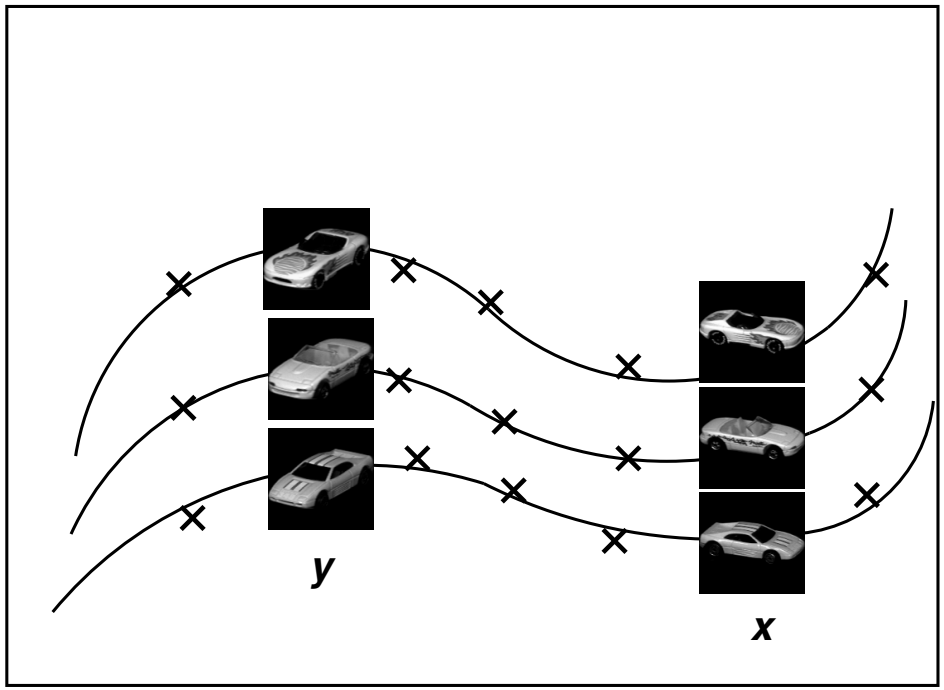
- Solution: Allow x_i and y_j to be in one clique.
- This will require “transistor neurons” that can do more than weighted summation.

$w^T x$?



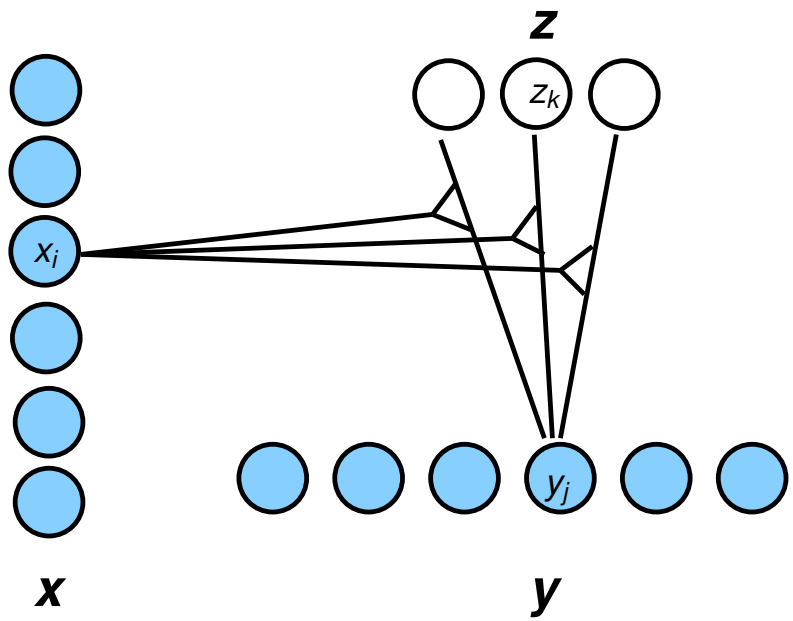
- Mel, 1994

Families of manifolds



- If ***y*** is a transformed version of ***x***, then ***y*** will be on a **conditional manifold**.
- **Idea:** Learn a model for ***y***, but let parameters be **a function of *x***.

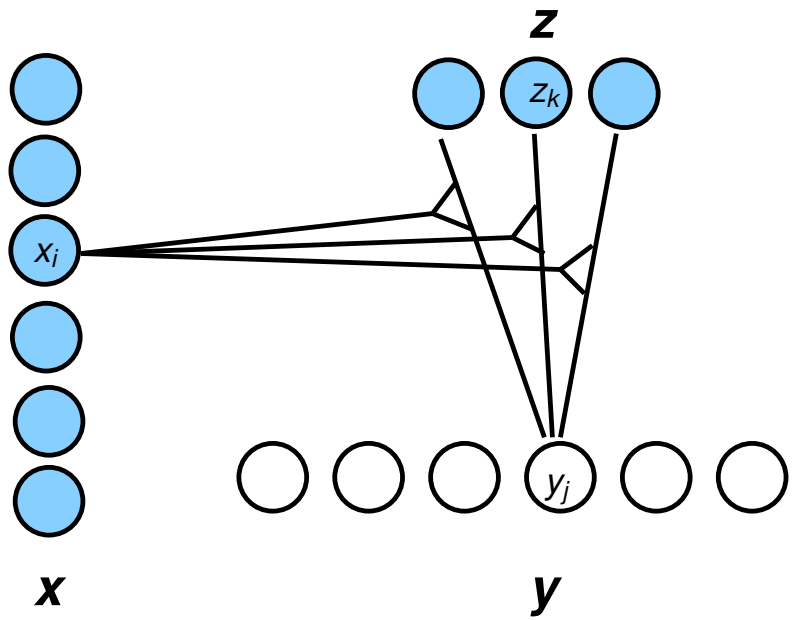
Bi-linear models



- With $w_{jk}(\boldsymbol{x}) = \sum_i w_{ijk} x_i$ we have

$$z_k = \sum_j w_{jk} y_j = \sum_j \left(\sum_i w_{ijk} x_i \right) y_j = \sum_{ij} w_{ijk} x_i y_j$$

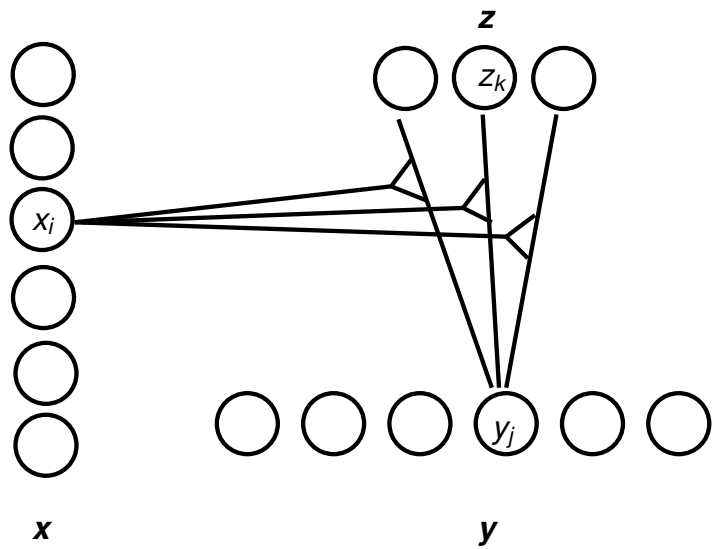
Bi-linear models



- Similar for $\boldsymbol{y}$:

$$y_j = \sum_k w_{jk} z_k = \sum_k \left(\sum_i w_{ijk} x_i \right) z_k = \sum_{ik} w_{ijk} x_i z_k$$

Learning by predicting y from x

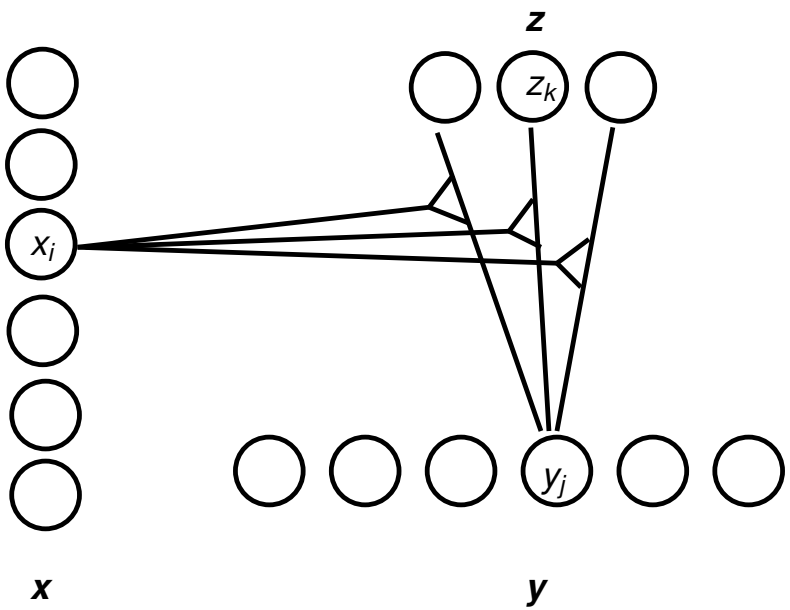


- Predictive cost:

$$\sum_j \left(y_j - \sum_{ik} w_{ijk} x_i z_k\right)^2$$

- (Tenenbaum, Freeman; 2000), (Grimes, Rao; 2005), (Olshausen; 2007), (Memisevic, Hinton; 2007)

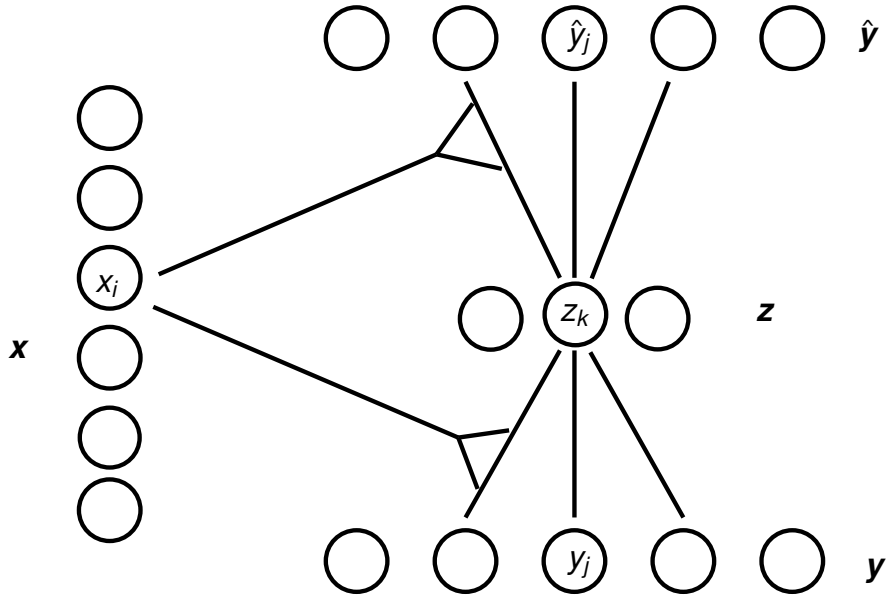
Example: Gated Boltzmann machine



$$E(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = \sum_{ijk} w_{ijk} x_i y_j z_k$$
$$p(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})} \exp\left(E(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})\right)$$
$$Z(\boldsymbol{x}) = \sum_{\boldsymbol{y},\boldsymbol{z}} \exp\left(E(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})\right)$$
$$p(z_k|\boldsymbol{x}, \boldsymbol{y}) = \text{sigmoid}(\sum_{ij} W_{ijk} x_i y_j)$$
$$p(y_j|\boldsymbol{x}, \boldsymbol{z}) = \text{sigmoid}(\sum_{ik} W_{ijk} x_i z_k)$$

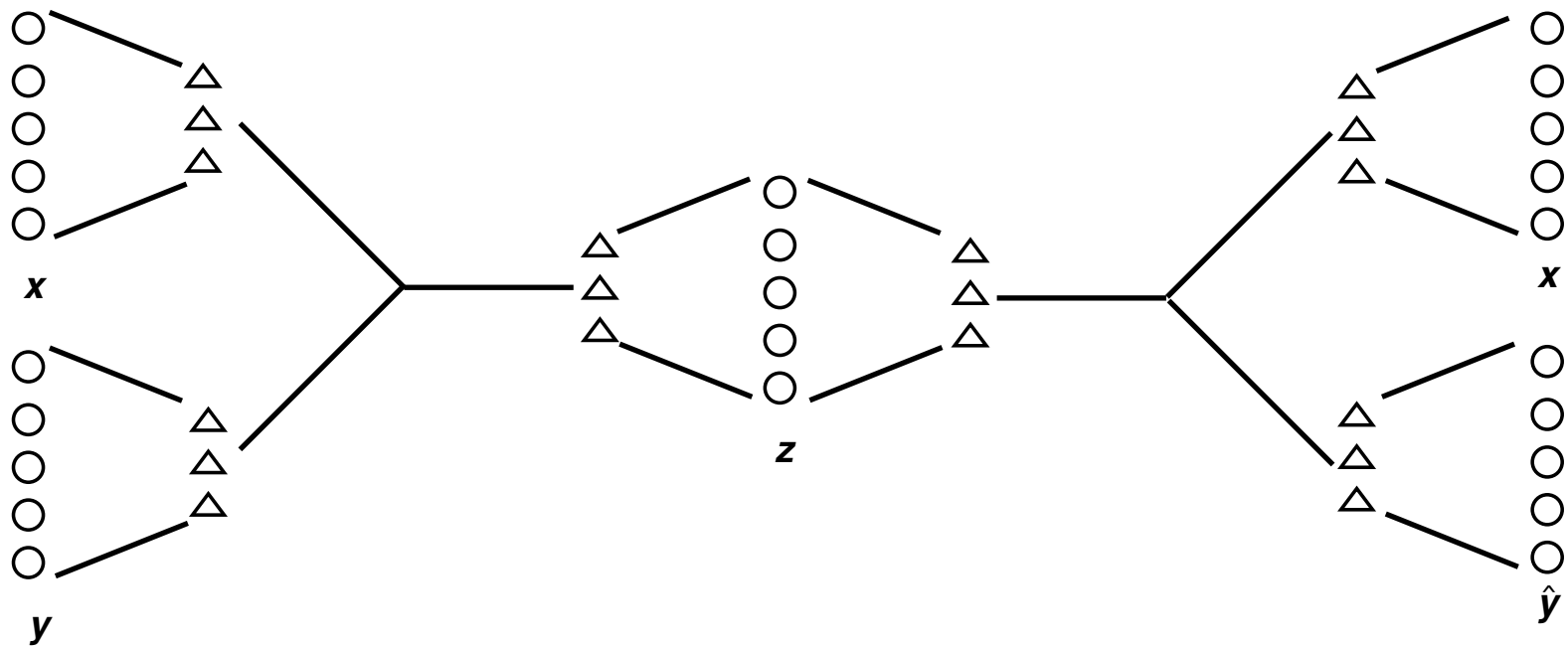
- An RBM that is defined at test-time (Memisevic, Hinton; 2007)

Example: Gated autoencoder



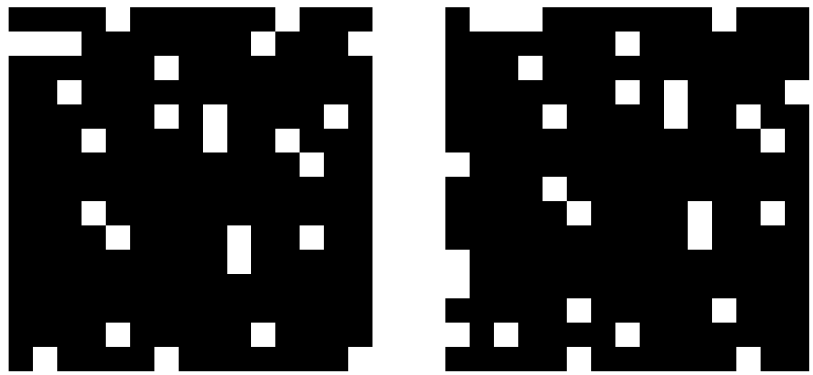
- Encoder and decoder weights become a function of $\boldsymbol{x}$.
- Training with back-prop (Memisevic, 2008)

Factored Gated Autoencoder



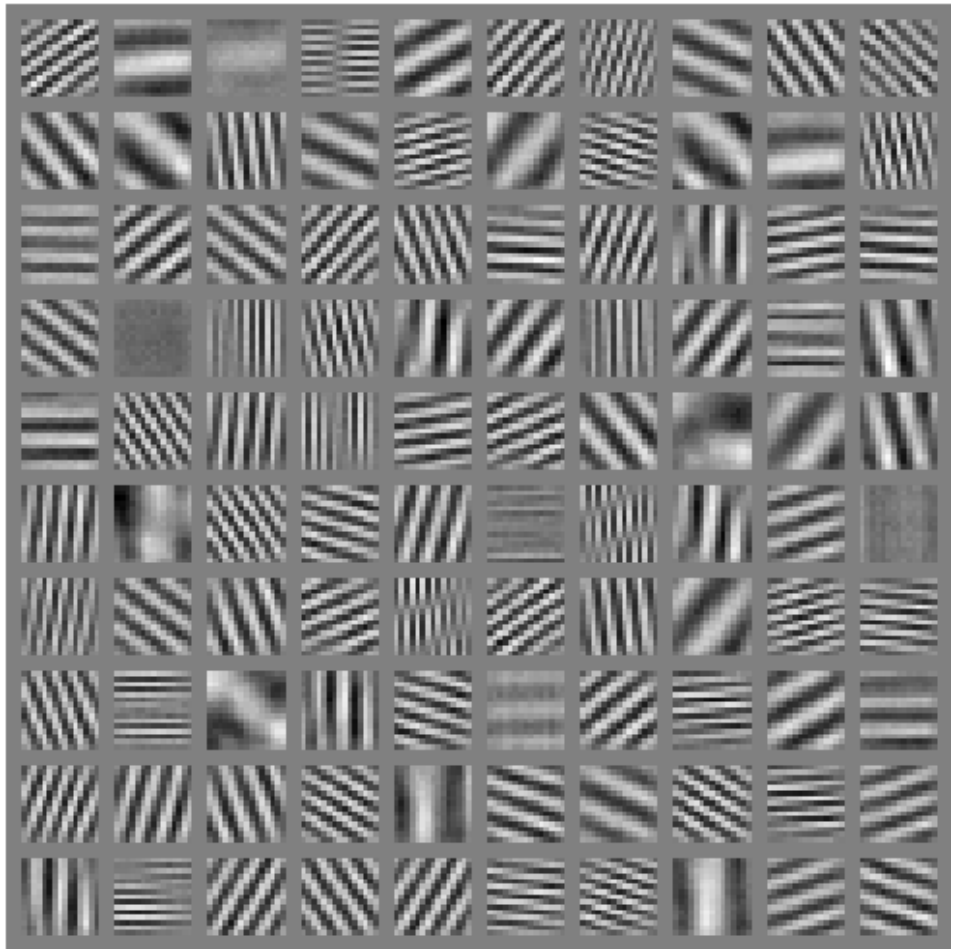
- Projecting onto filters *first* allows us to use fewer products. (Memisevic, Hinton 2010), (Taylor et al 2009)
- This is equivalent to *factorizing* the three-way parameter tensor.

Toy examples

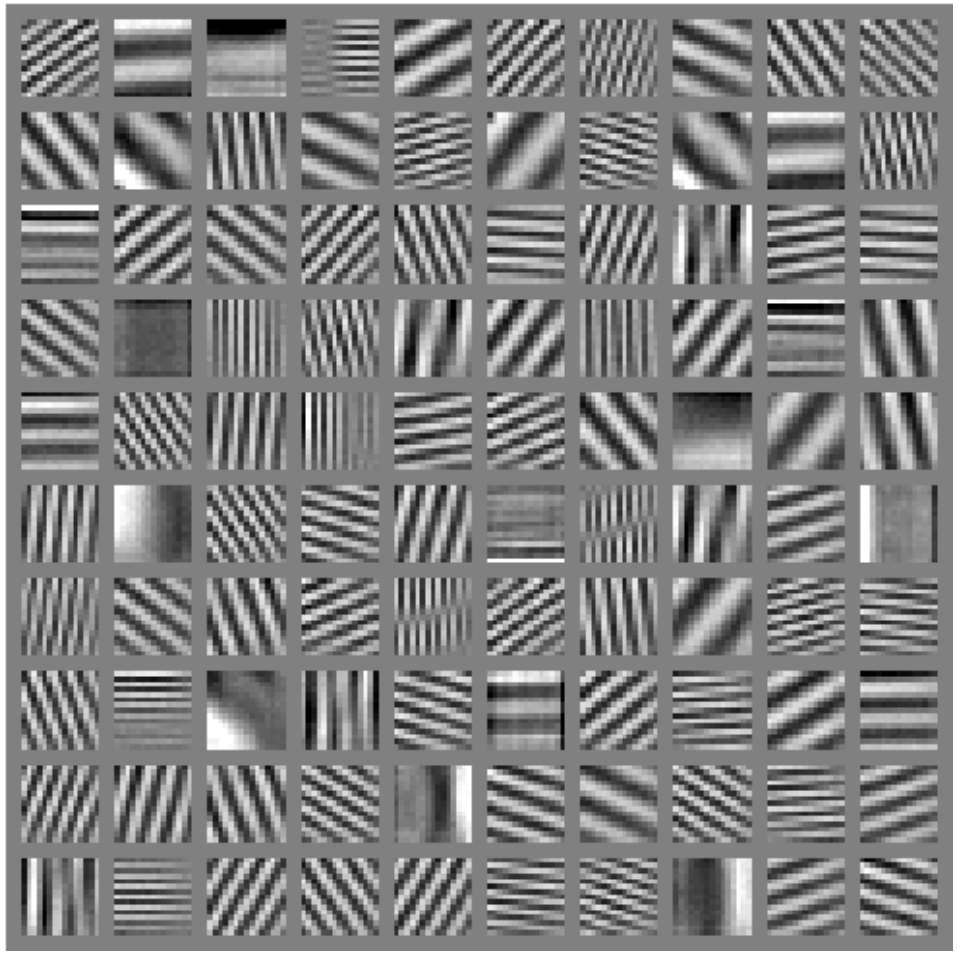


- There is no structure in these images.
- Only in *how they change*.

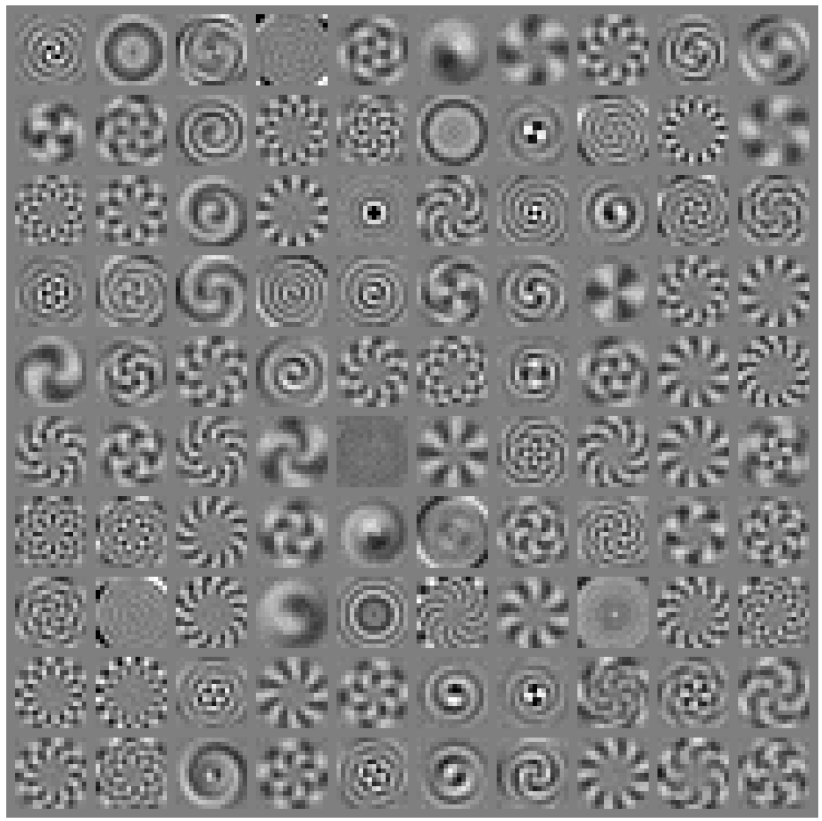
Learned filters w_{if}^{x}



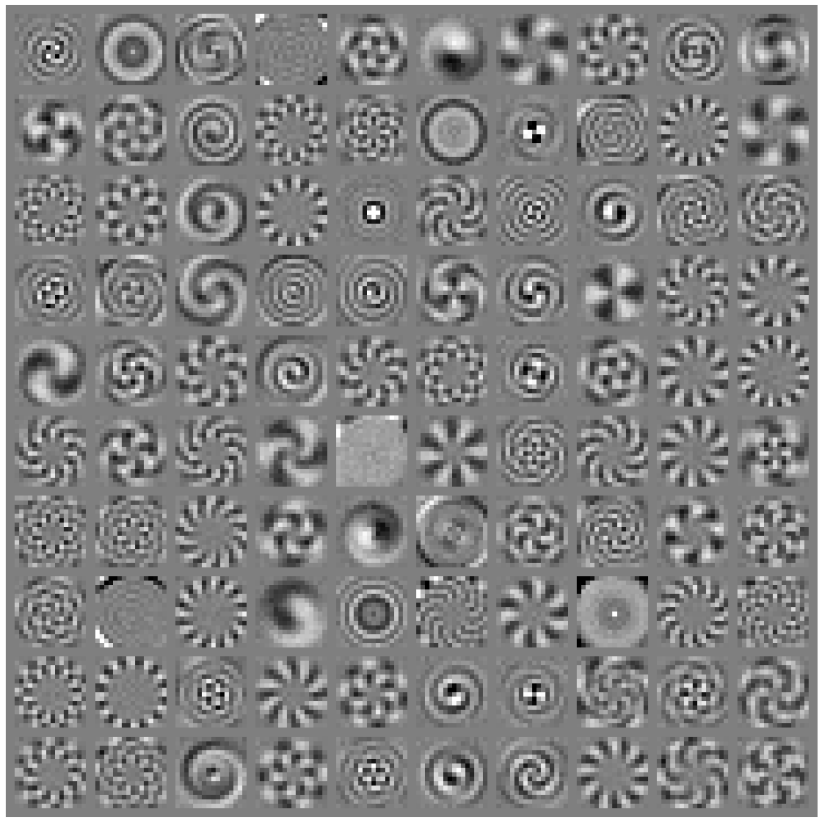
Learned filters w_{jf}^{y}



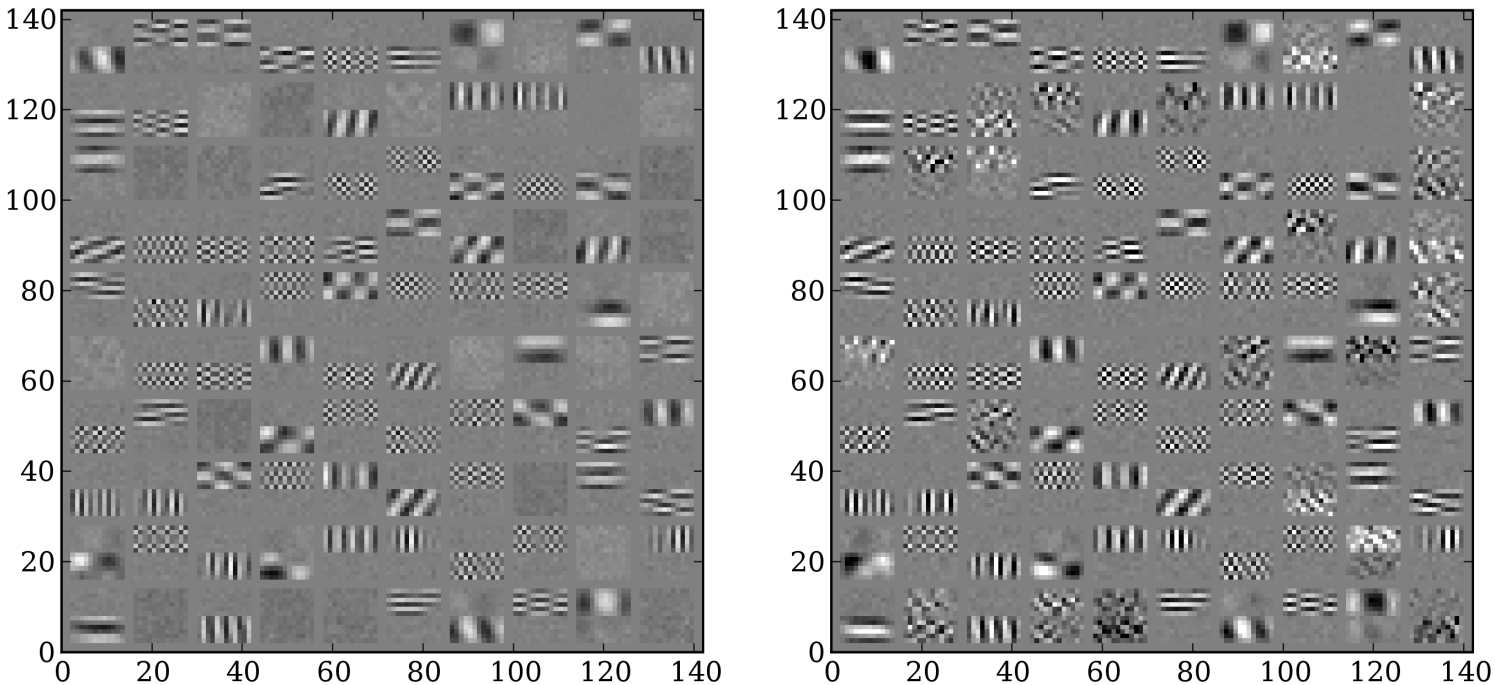
Rotation filters



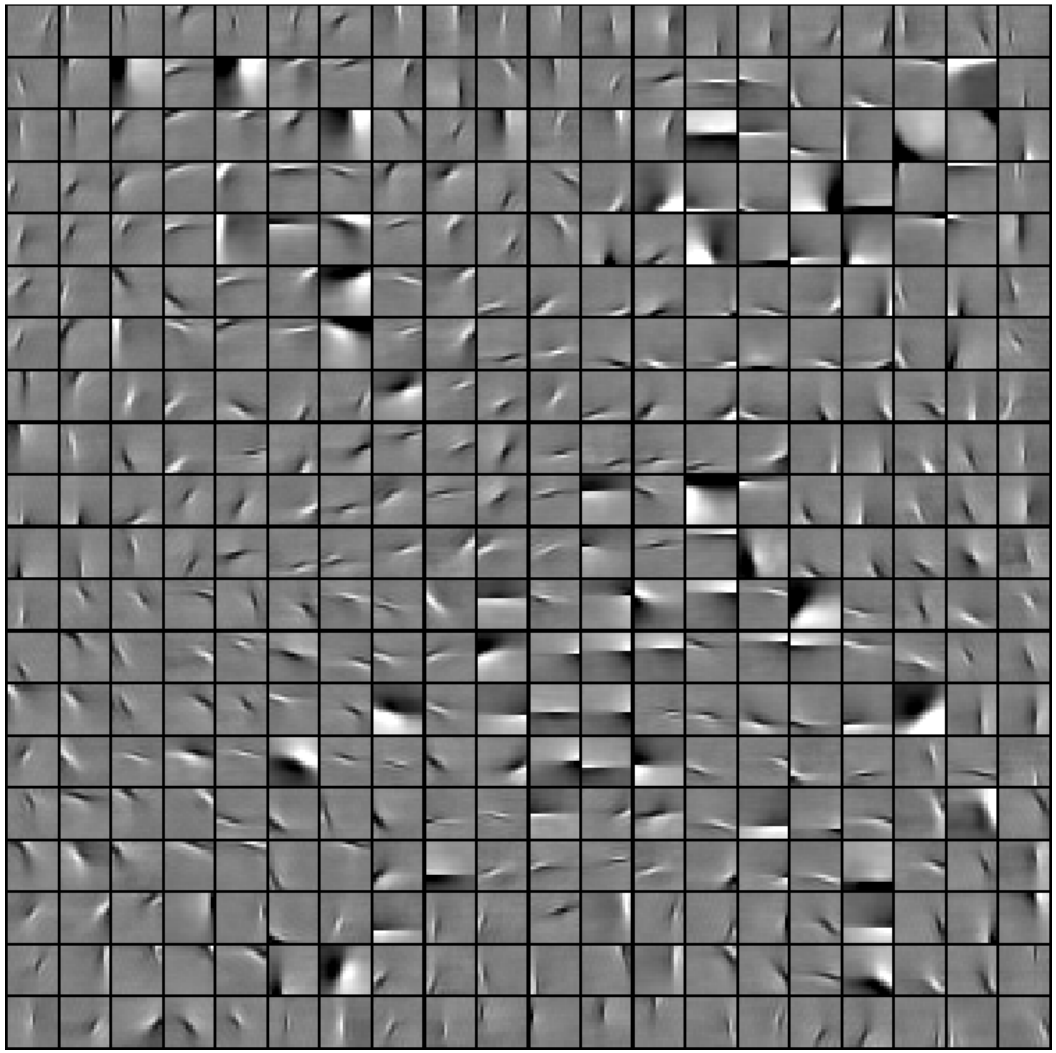
Rotation filters



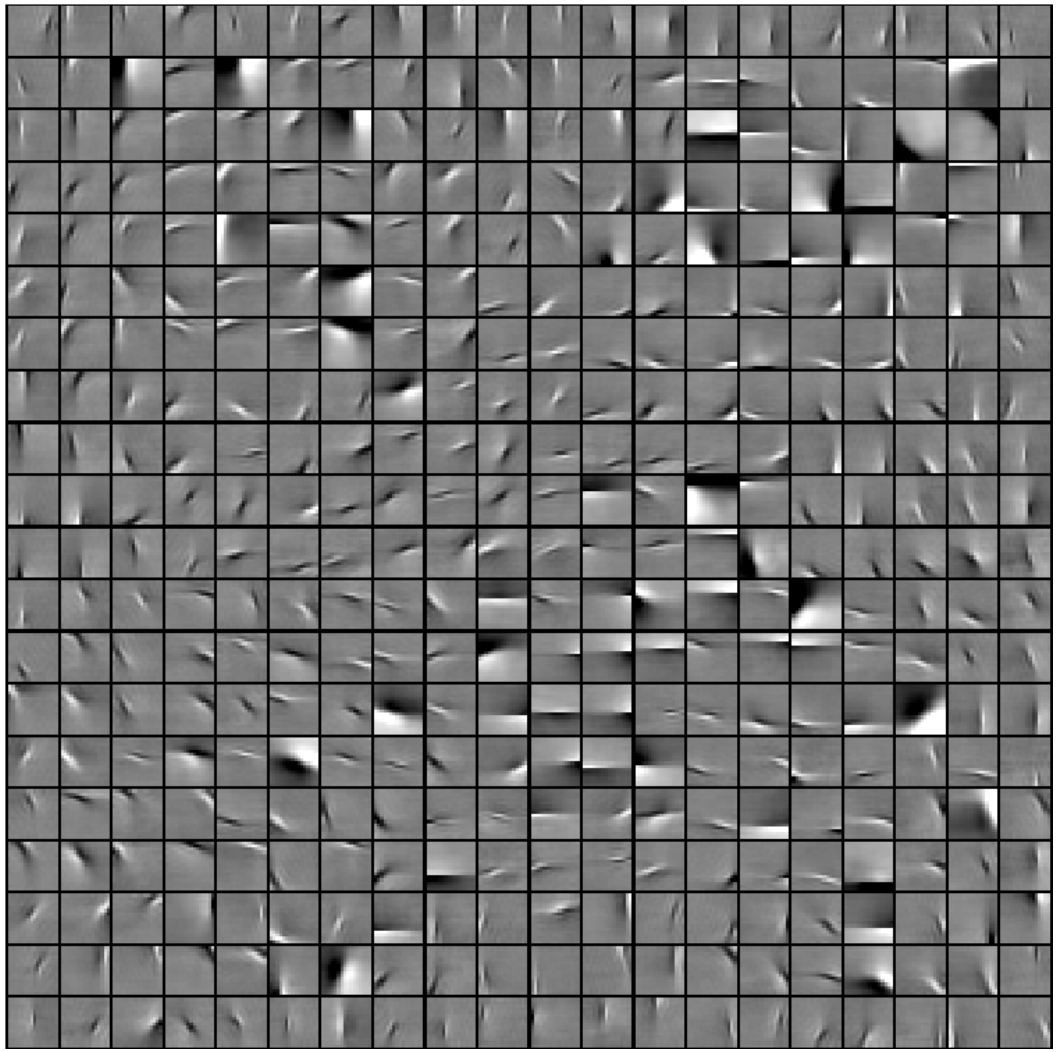
Filters learned from split-screen shifts



Natural video filters



Natural video filters



More theory

(I) Orthogonal transformations decompose into 2-D rotations:

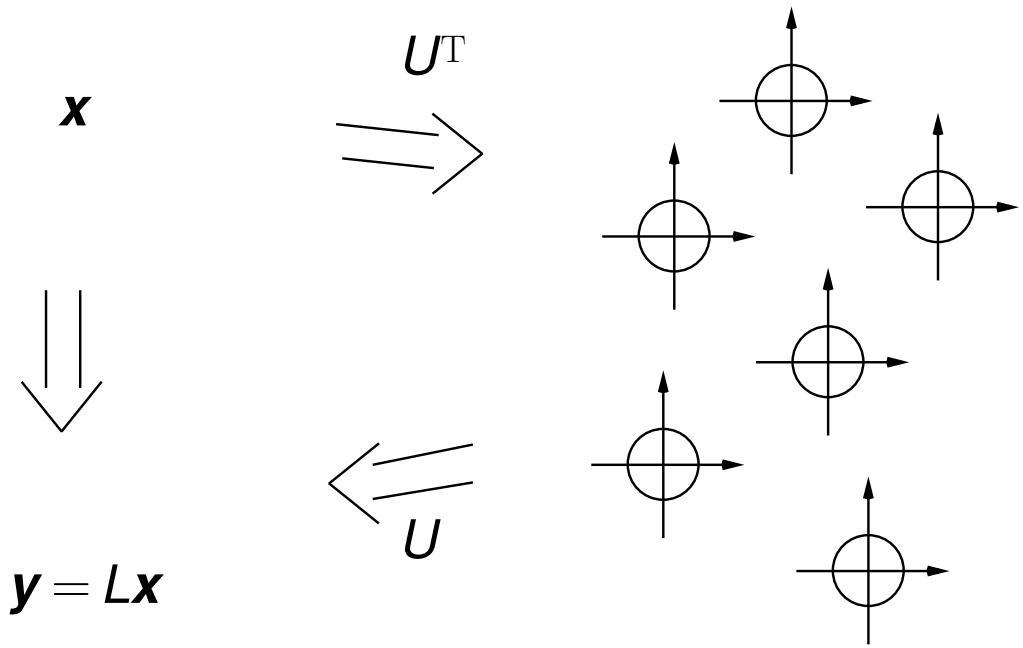
$$U^{\mathrm{T}} L U = \begin{bmatrix} R_1 & & \\ & \ddots & \\ & & R_k \end{bmatrix} \qquad R_i = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i) \end{bmatrix}$$

- (Eigen-decomposition $L = UDU^{\mathrm{T}}$ has complex eigenvalues of length 1.)

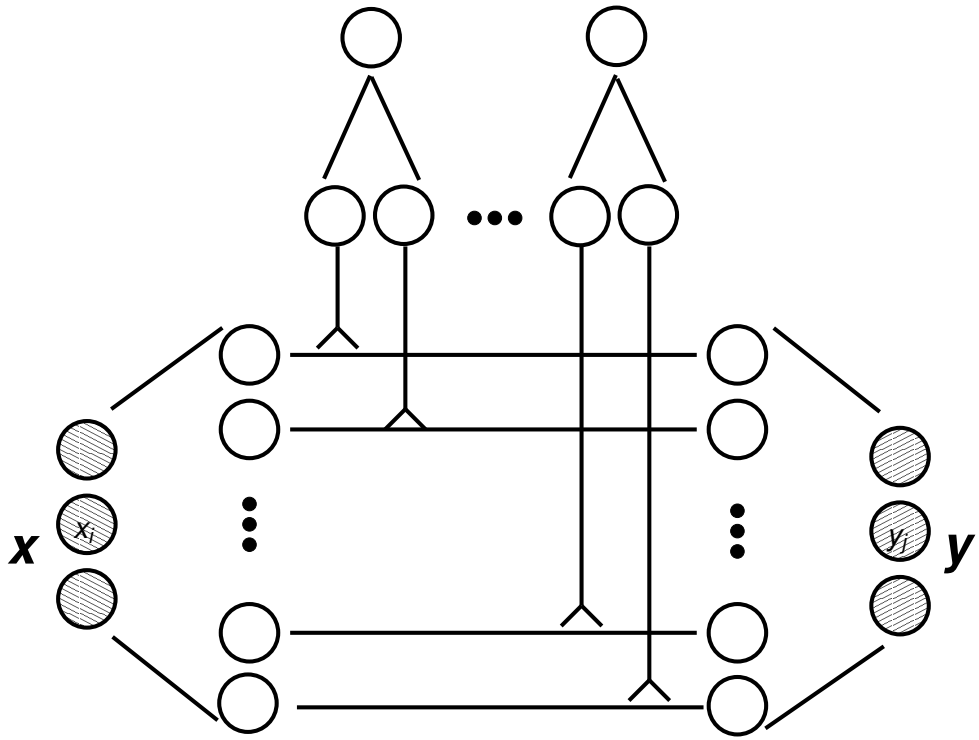
(II) Commuting transformations share an eigen-basis:

- They differ only with respect to the rotation-angle they apply in their eigenspace.

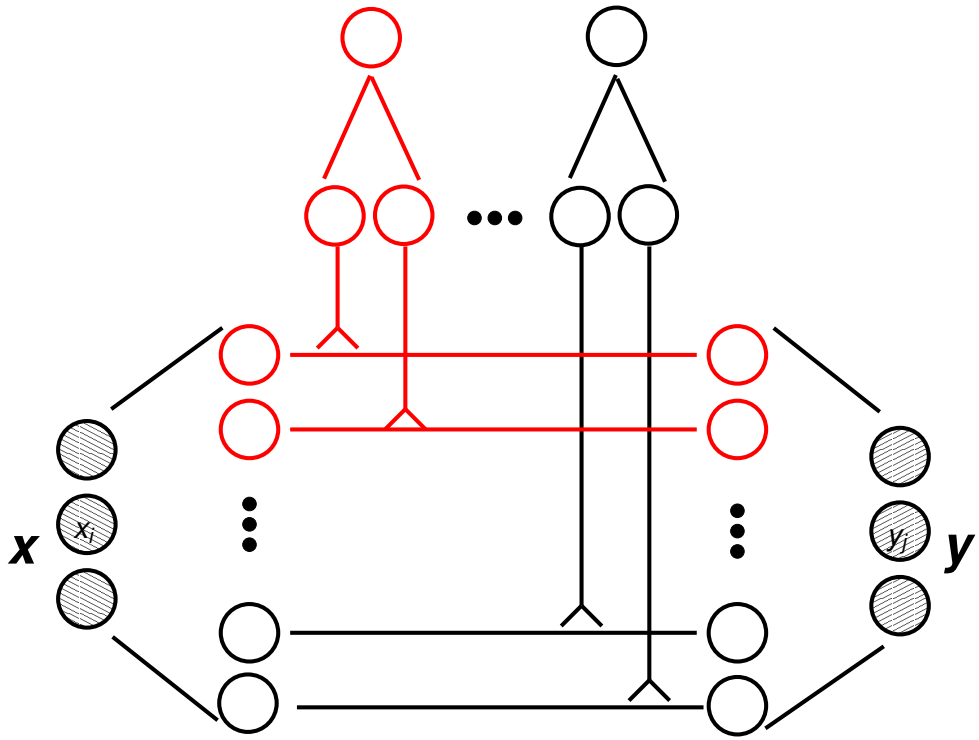
(I)+(II)



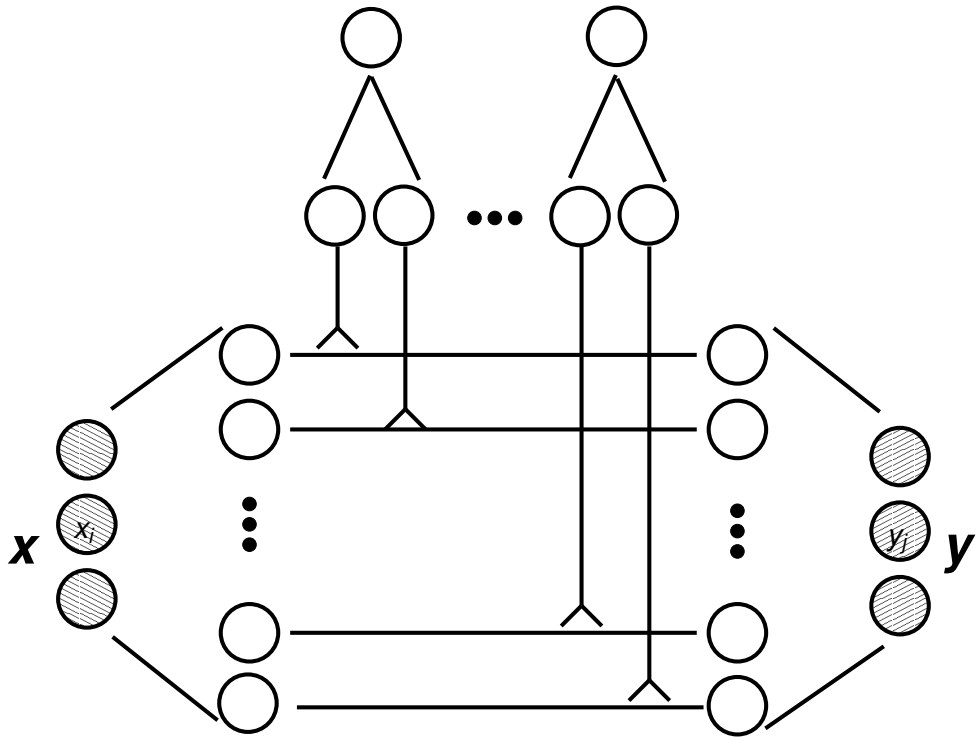
To detect the rotation angle, compute a 2-d inner product



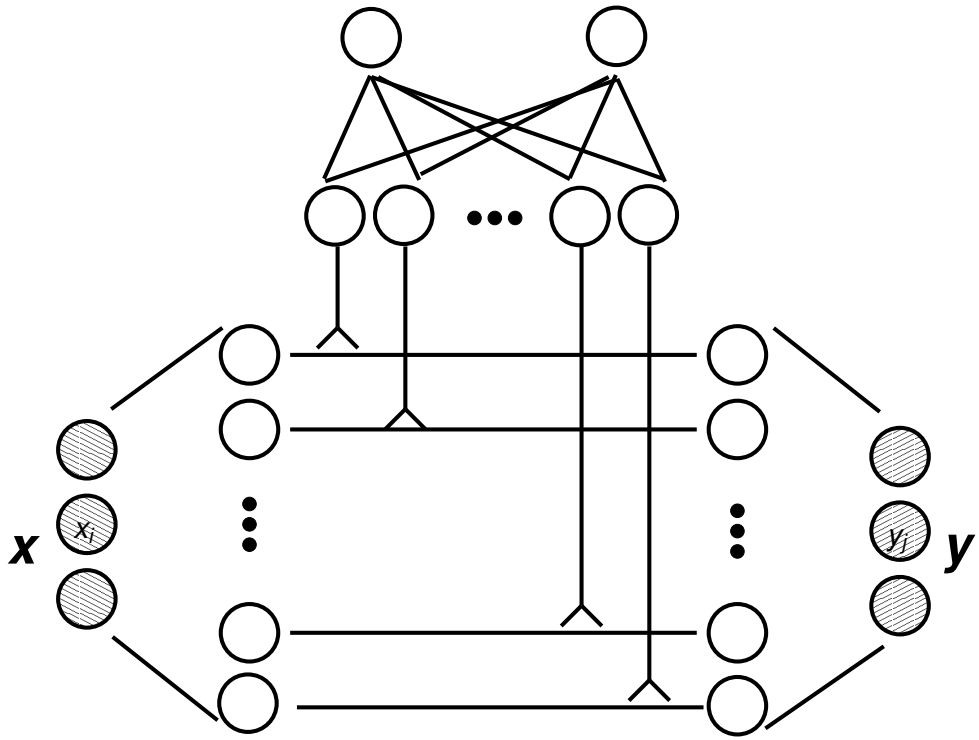
To detect the rotation angle, compute a 2-d inner product



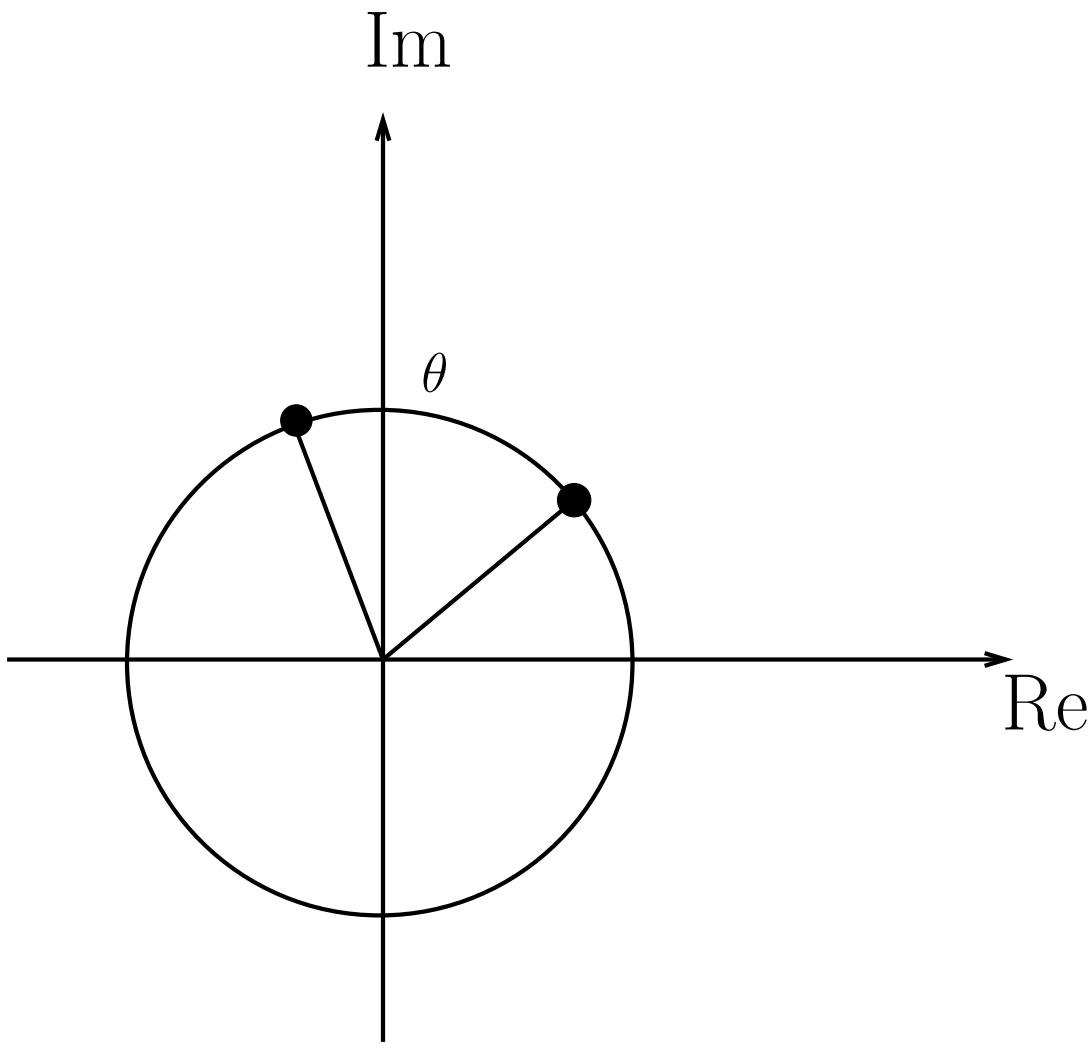
To detect the rotation angle, compute a 2-d inner product



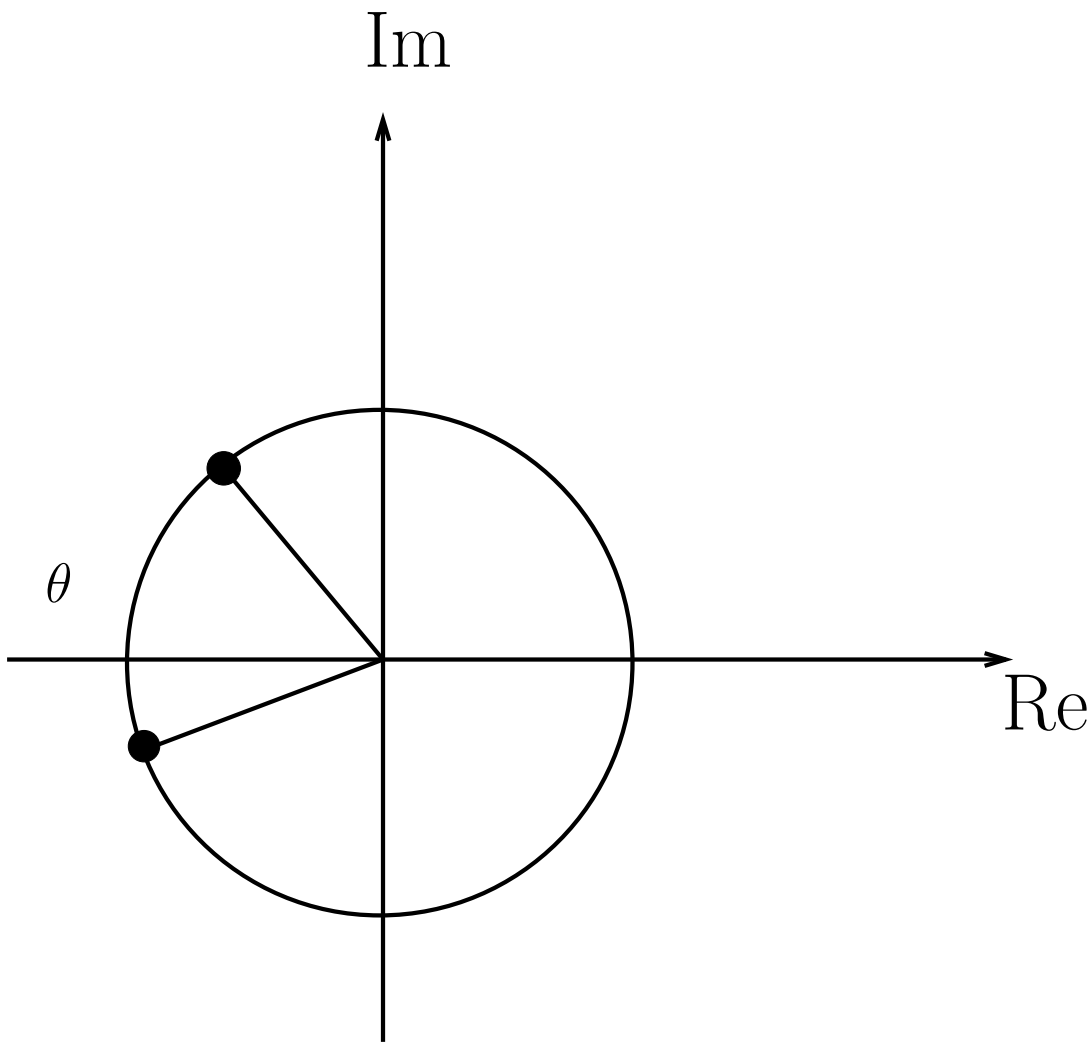
To detect the rotation angle, compute a 2-d inner product



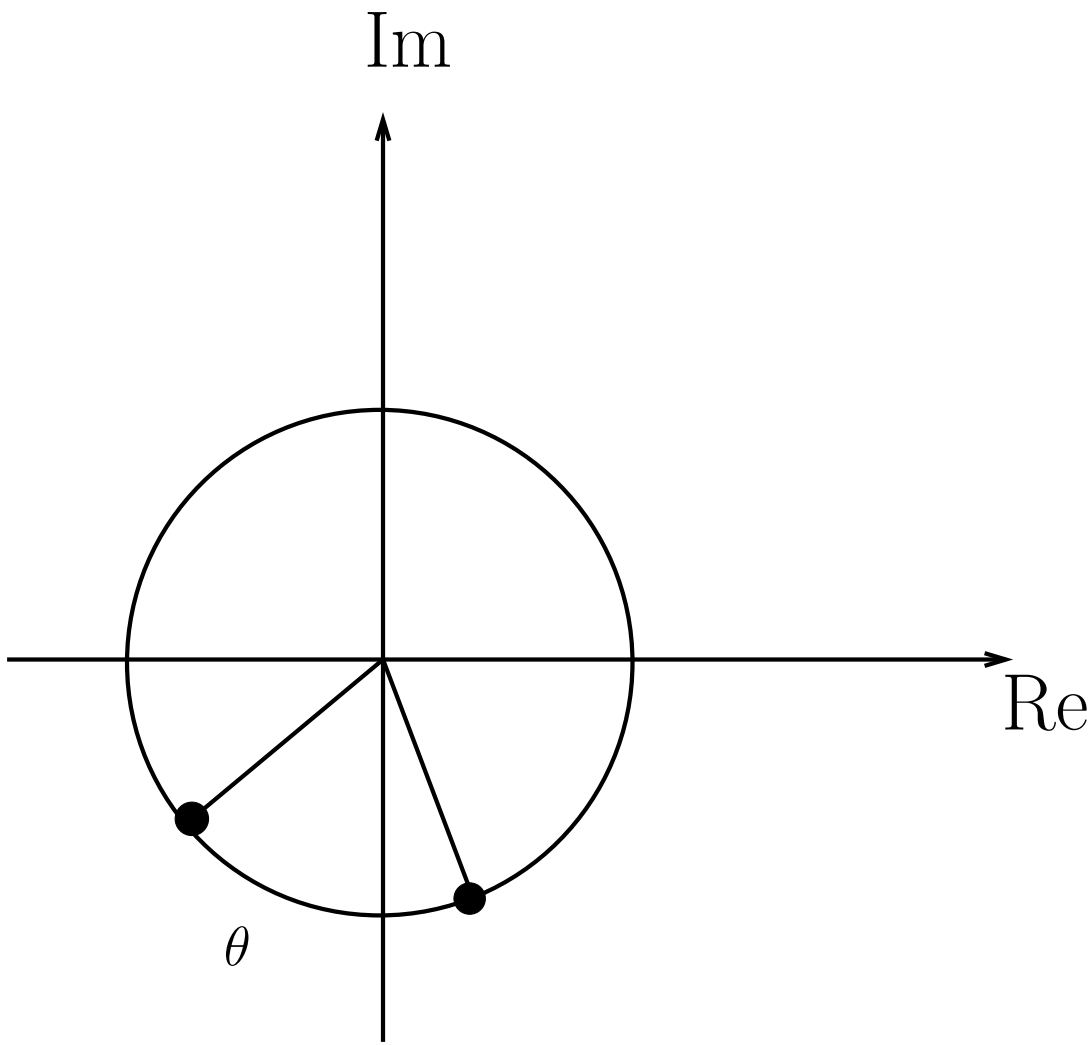
Transformations are transformation-invariant



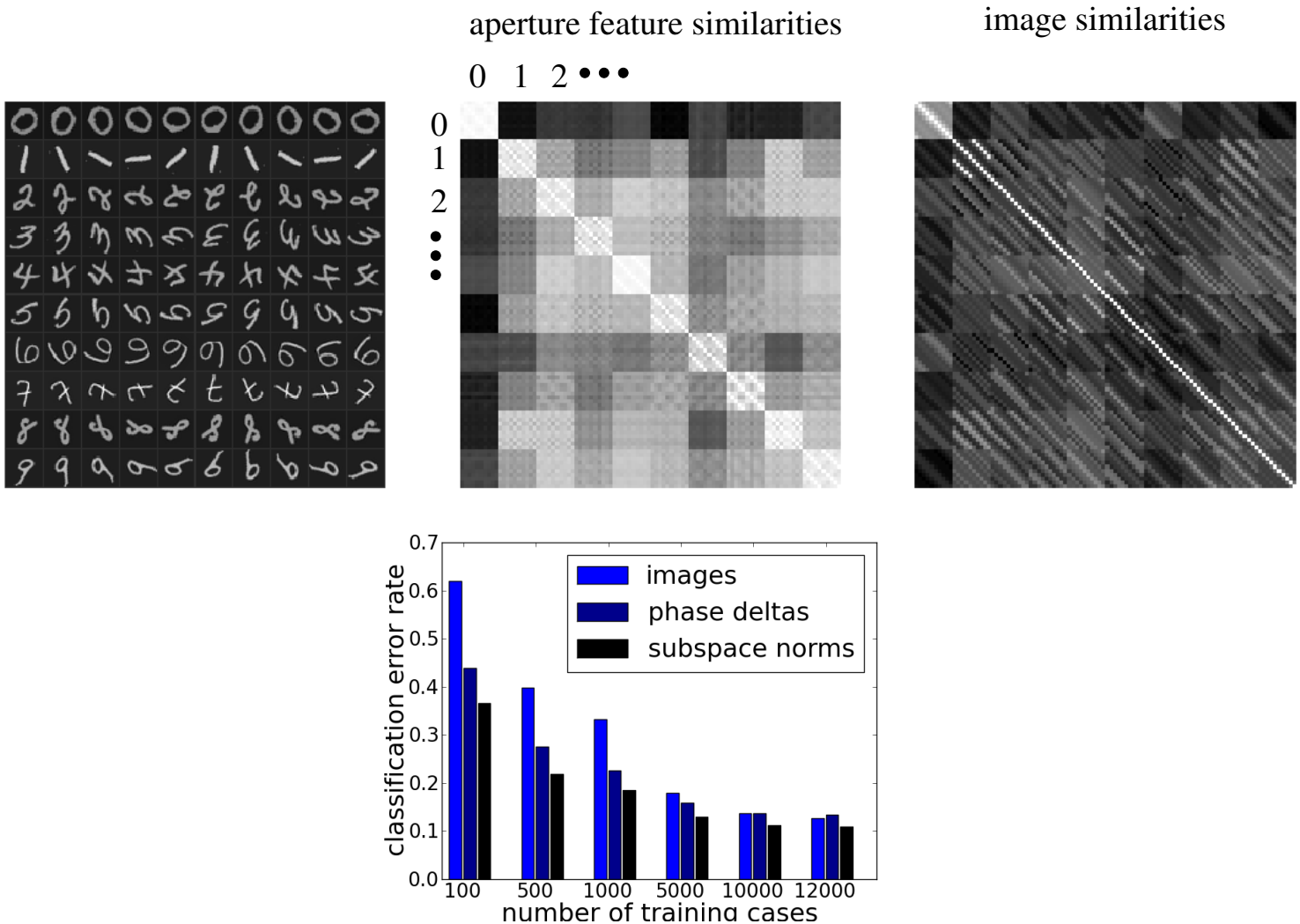
Transformations are transformation-invariant



Transformations are transformation-invariant



Use videos not images at test-time



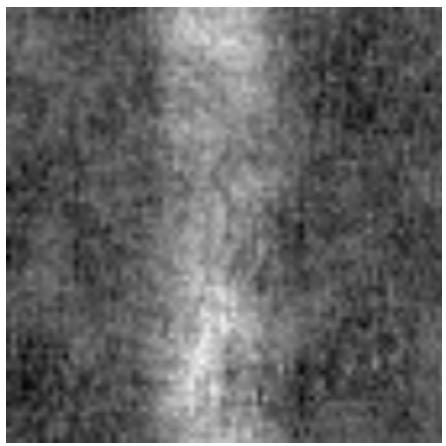
- (Memisevic, Exarchakis 2013)
- Cf. (Cadieu, Olshausen 2011), (Zou et al 2012)

Deep learning for stereo

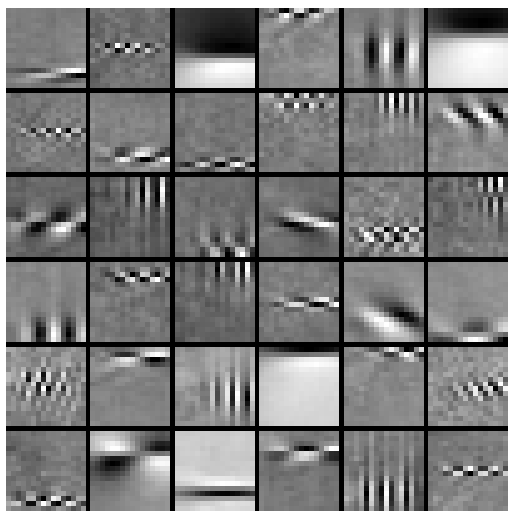
left image

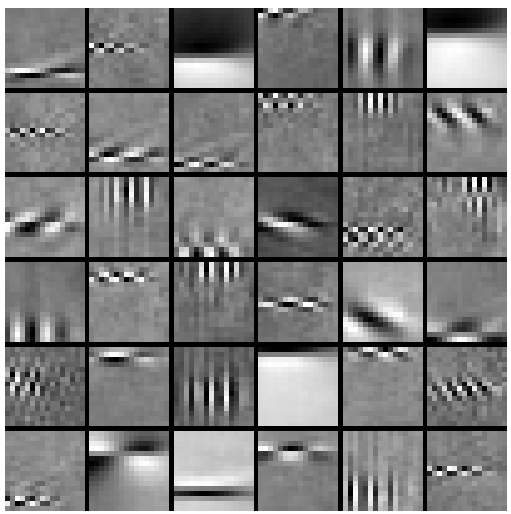
right image

predicted disparity

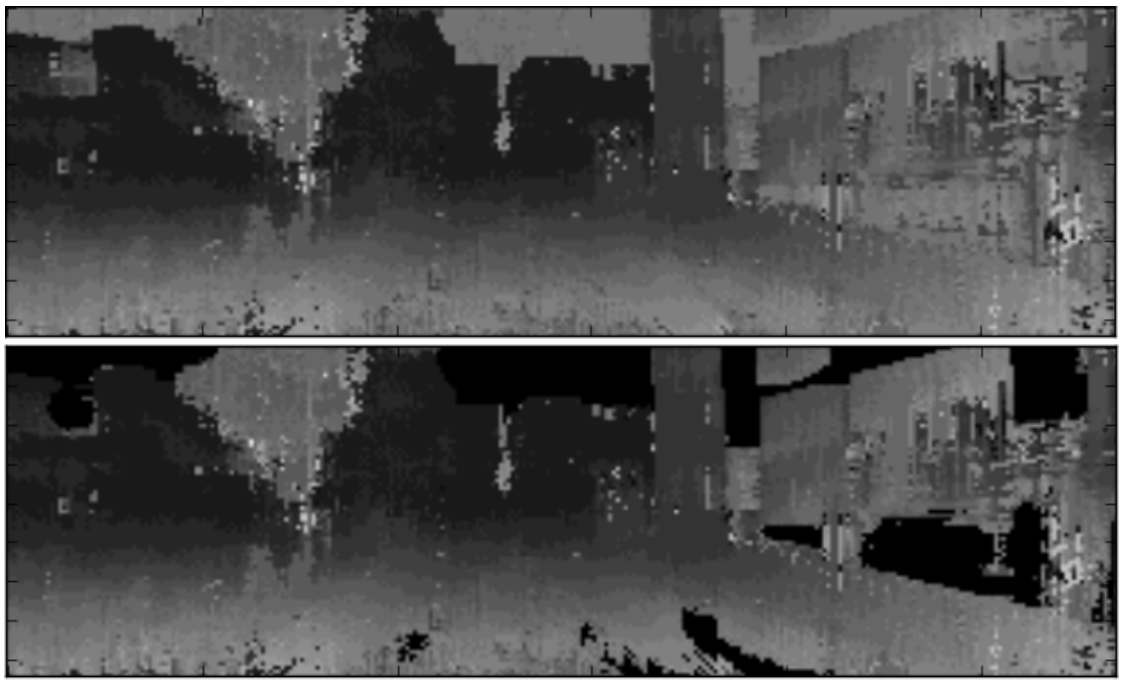


Deep learning for stereo (Konda et al 2013)





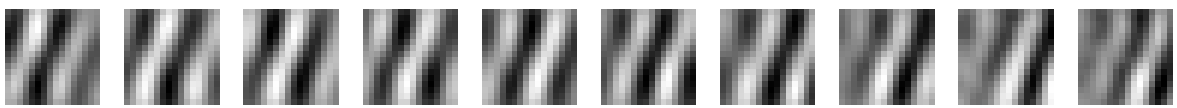
Inferred depth map

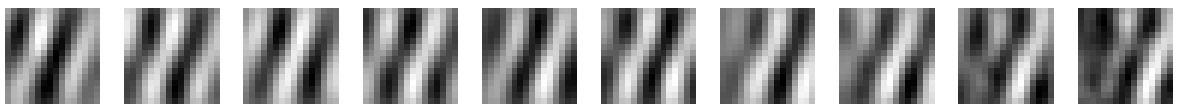


- Top: inferred depth map, bottom: thresholded to remove uncertain regions

Hollywood 3D (Hadfield, et al. 2013)

- filters:

x_{left} : 

x_{right} : 

Precision rates by action class

Action	SAE-MD	SAE-MD(Av)	SAE-MD(Ct)	SAE-M	SAE-D	ISA	3D-Ha	4D-Ha	3.5D-Ha
NoAction	12.10	12.77	13.10	**15.73**	12.15	12.27	12.1	12.9	13.7
Run	52.56	50.44	51.45	45.38	**56.07**	24.91	19.0	22.4	27.0
Punch	**41.09**	38.01	32.68	33.86	36.17	31.17	10.4	4.8	5.7
Kick	9.41	7.94	6.86	6.63	**11.84**	9.96	9.3	4.3	4.8
Shoot	30.26	35.51	30.49	30.52	**40.72**	32.48	27.9	17.2	16.6
Eat	5.85	7.03	6.78	7.29	**9.03**	6.89	5.0	5.3	5.6
Drive	52.65	59.62	51.35	61.61	45.19	54.47	24.8	69.3	**69.6**
UsePhone	22.79	**23.92**	19.01	23.60	23.36	17.67	6.8	8.0	7.6
Kiss	15.03	16.40	16.12	**17.86**	17.06	14.94	8.4	10.0	10.2
Hug	6.64	7.02	7.61	7.38	9.27	9.48	4.3	4.4	**12.1**
StandUp	**37.35**	34.23	37.01	29.16	15.01	26.71	10.1	7.6	9.0
SitDown	6.51	6.95	7.53	7.40	**9.06**	5.13	5.3	4.2	5.6
Swim	16.58	**29.48**	17.60	29.45	26.70	16.09	11.3	5.5	7.5
Dance	43.15	36.26	44.59	29.64	25.12	**53.72**	10.1	10.5	7.5
mean AP	25.14	**26.11**	24.45	24.61	24.05	22.55	12.6	13.3	14.1

Analogy making

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

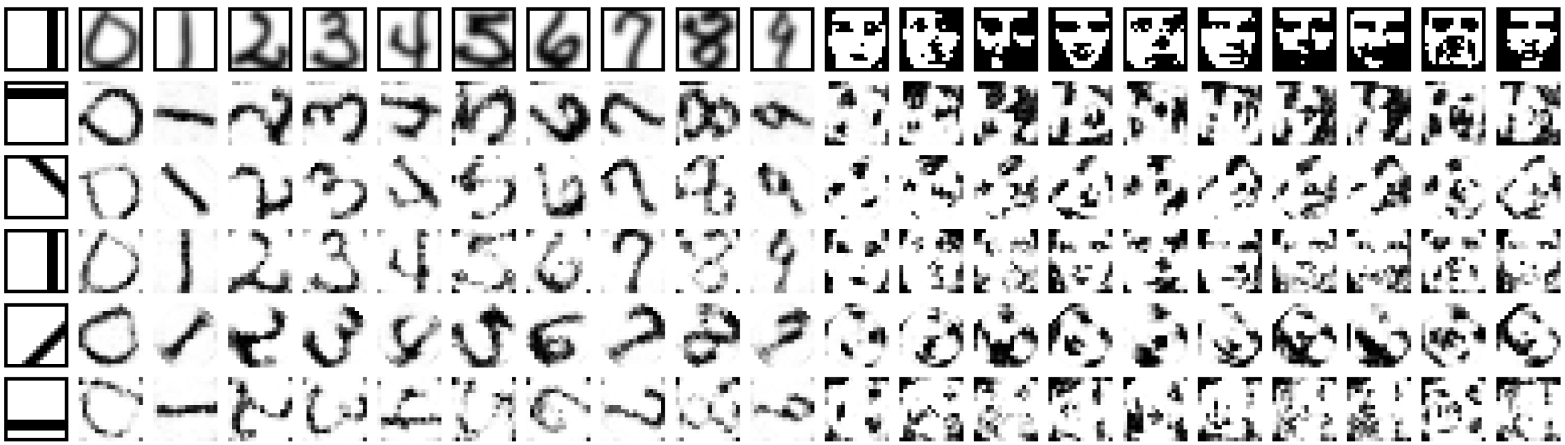
1. Infer transformation:

$$\boldsymbol{z}(\boldsymbol{x}_{\mathbf{source}}, \boldsymbol{y}_{\mathbf{source}})$$

2. Apply transformation:

$$\boldsymbol{y}(\boldsymbol{z}, \boldsymbol{x}_{\mathbf{target}})$$

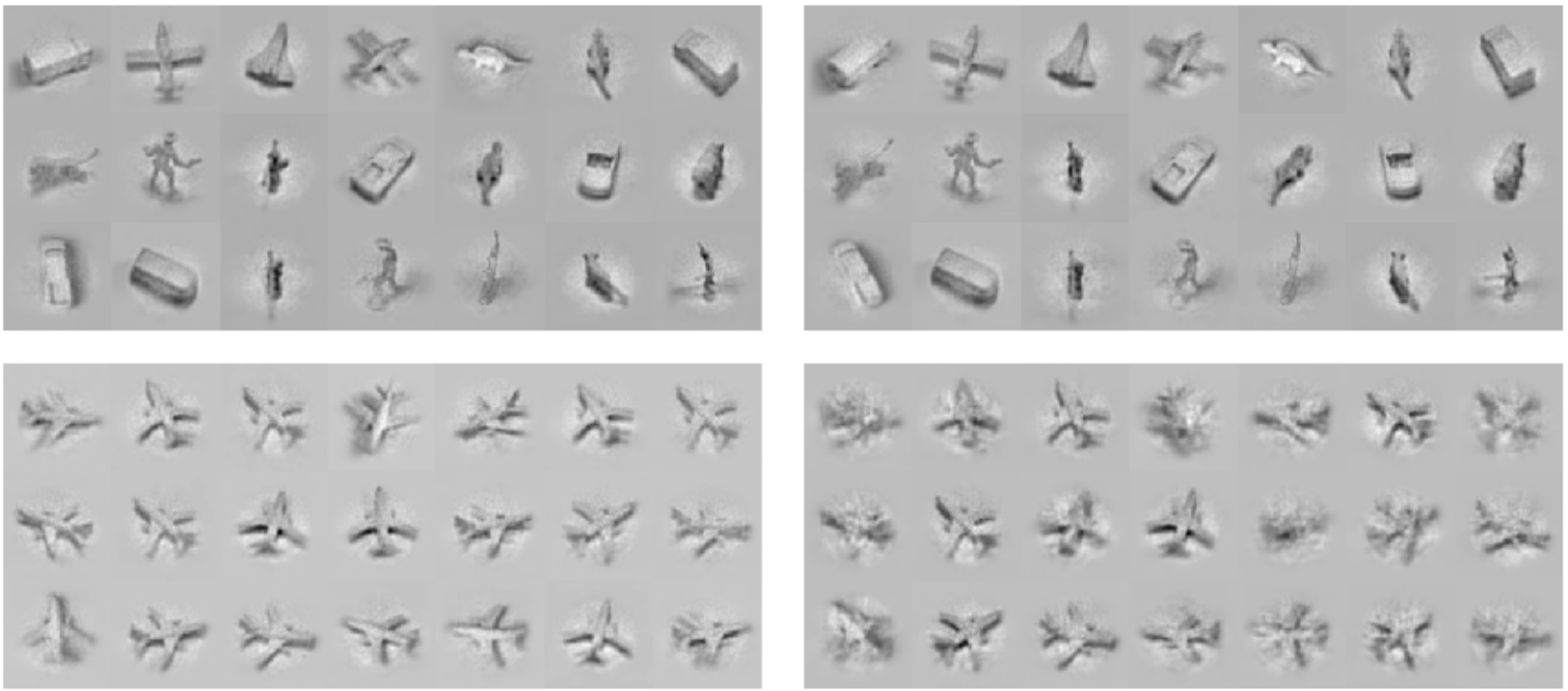
Analogy making



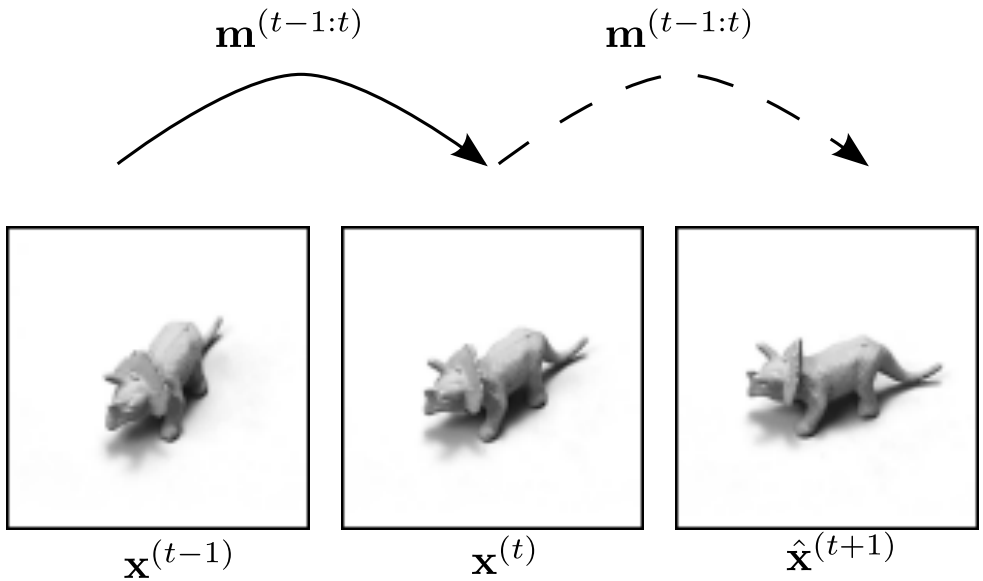
Analogy making for expression transfer

- (Susskind, et al., 2011)

NORB analogies

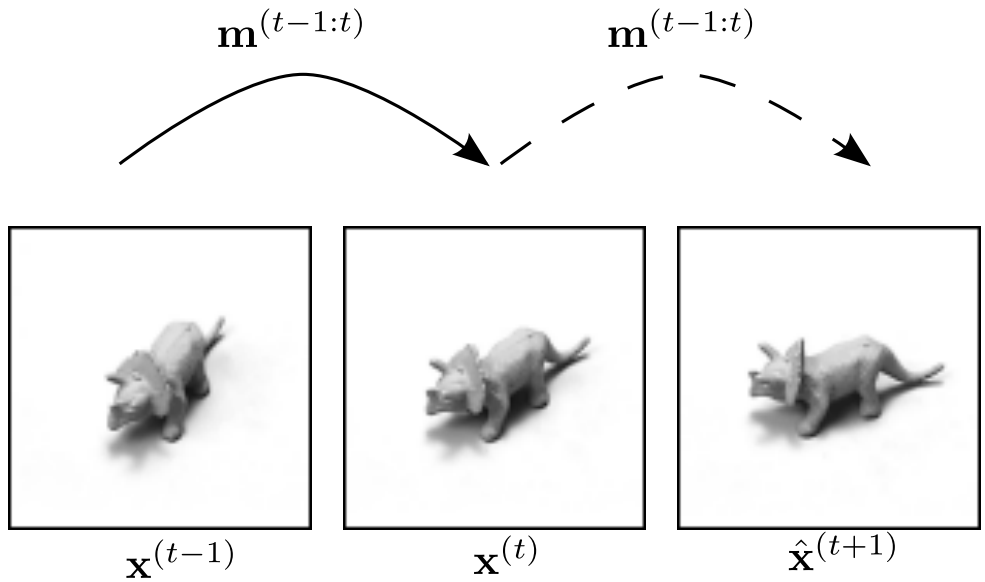


Using analogies for training (Michalski, 2013)



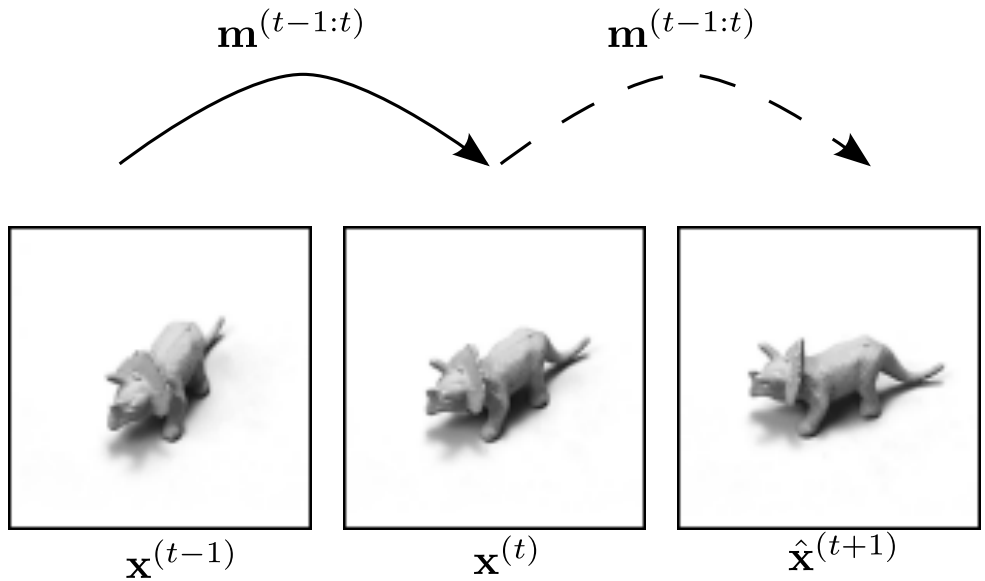
- infer transform, predict future, then backprop-through-time

Using analogies for training (Michalski, 2013)



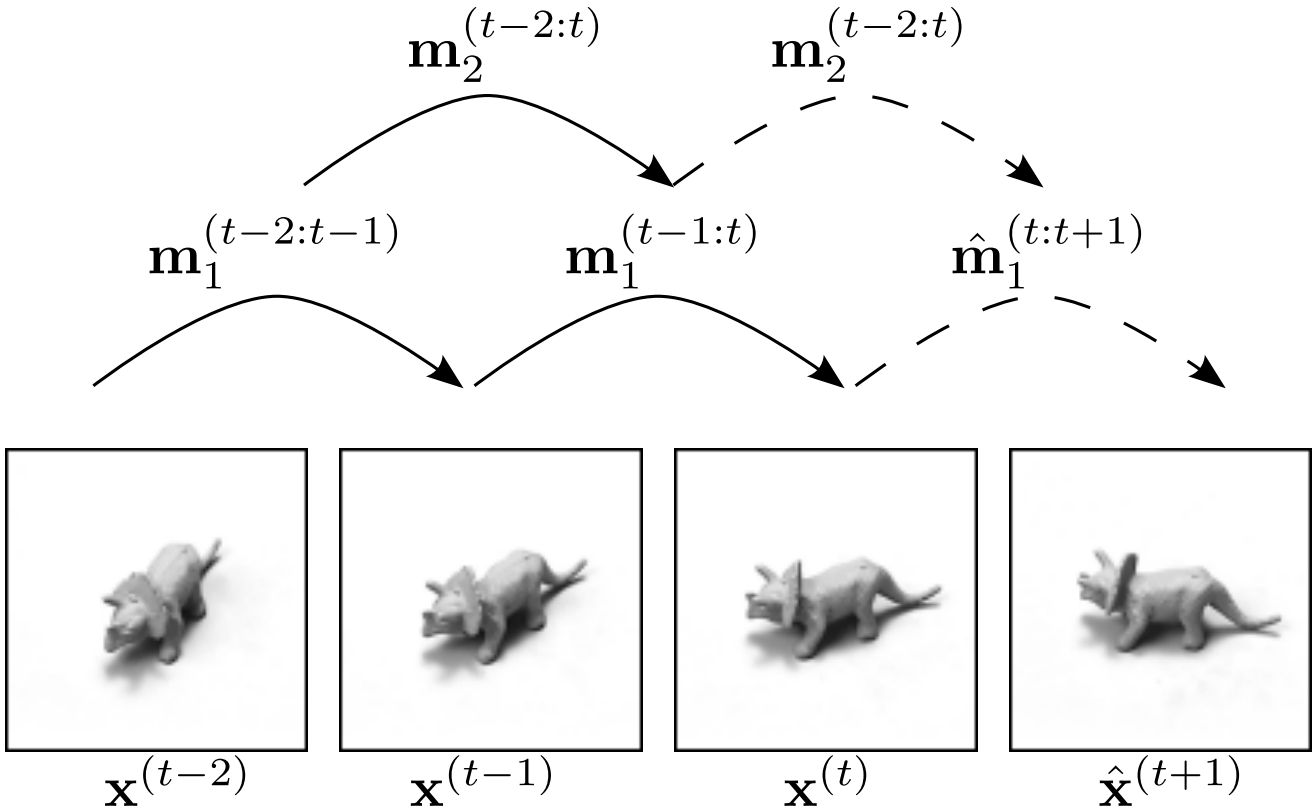
- infer transform, predict future, then backprop-through-time
- problem: transformation assumed to be constant

Using analogies for training (Michalski, 2013)



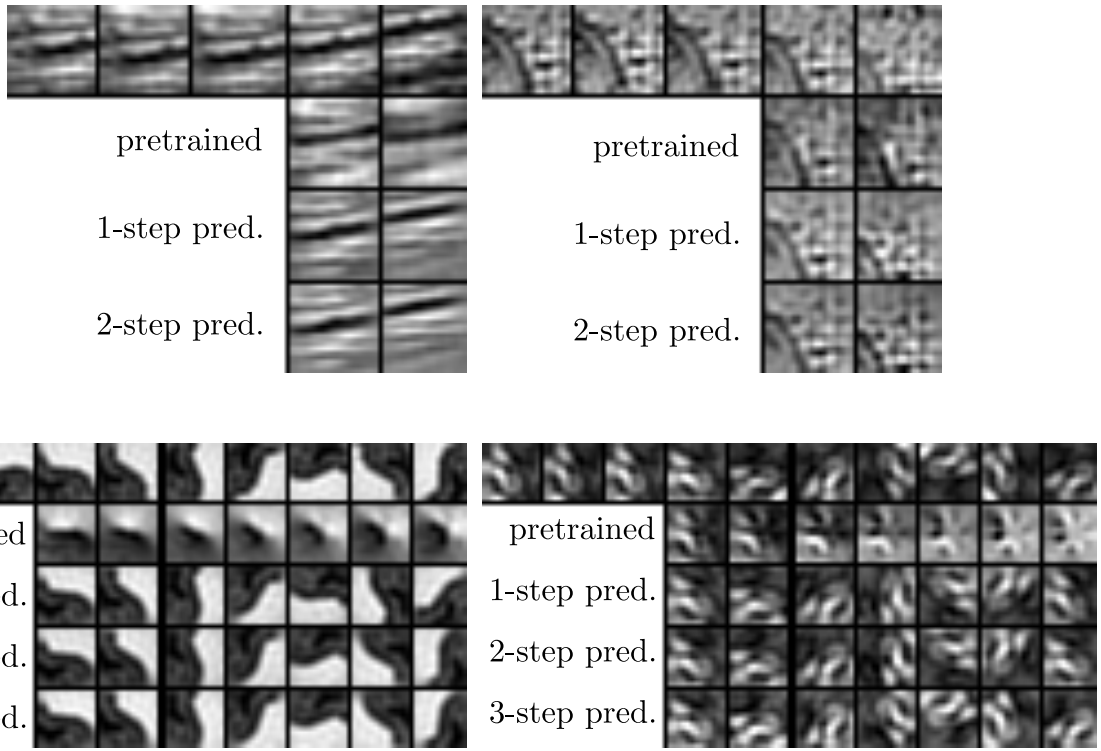
- infer transform, predict future, then backprop-through-time
- problem: transformation assumed to be constant
- solution: add a layer (or more)

Higher-order predictions (Michalski, 2013)

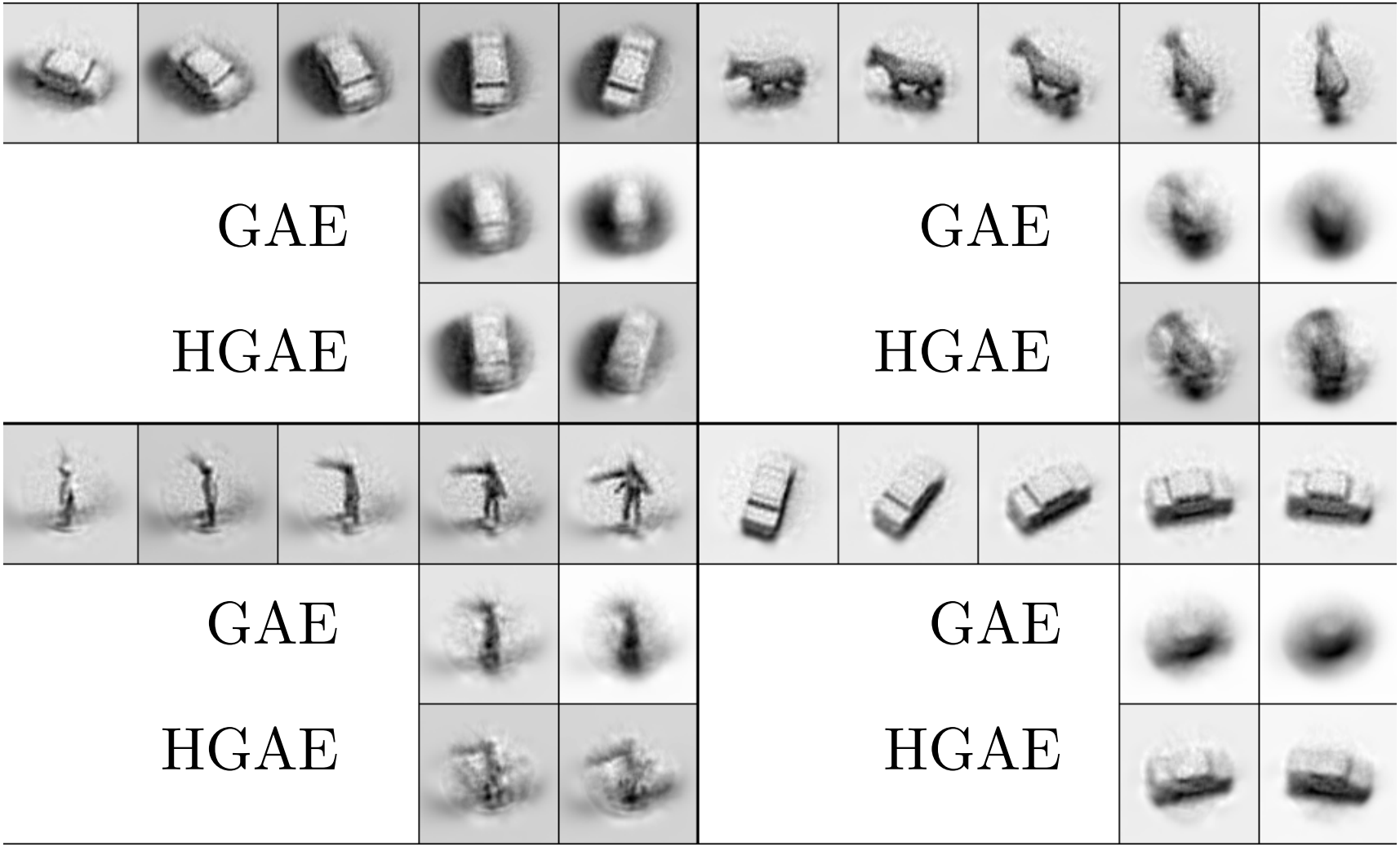


- the first additional layer will learn accelerations
- learn long-term correlations *without* explicit memory units

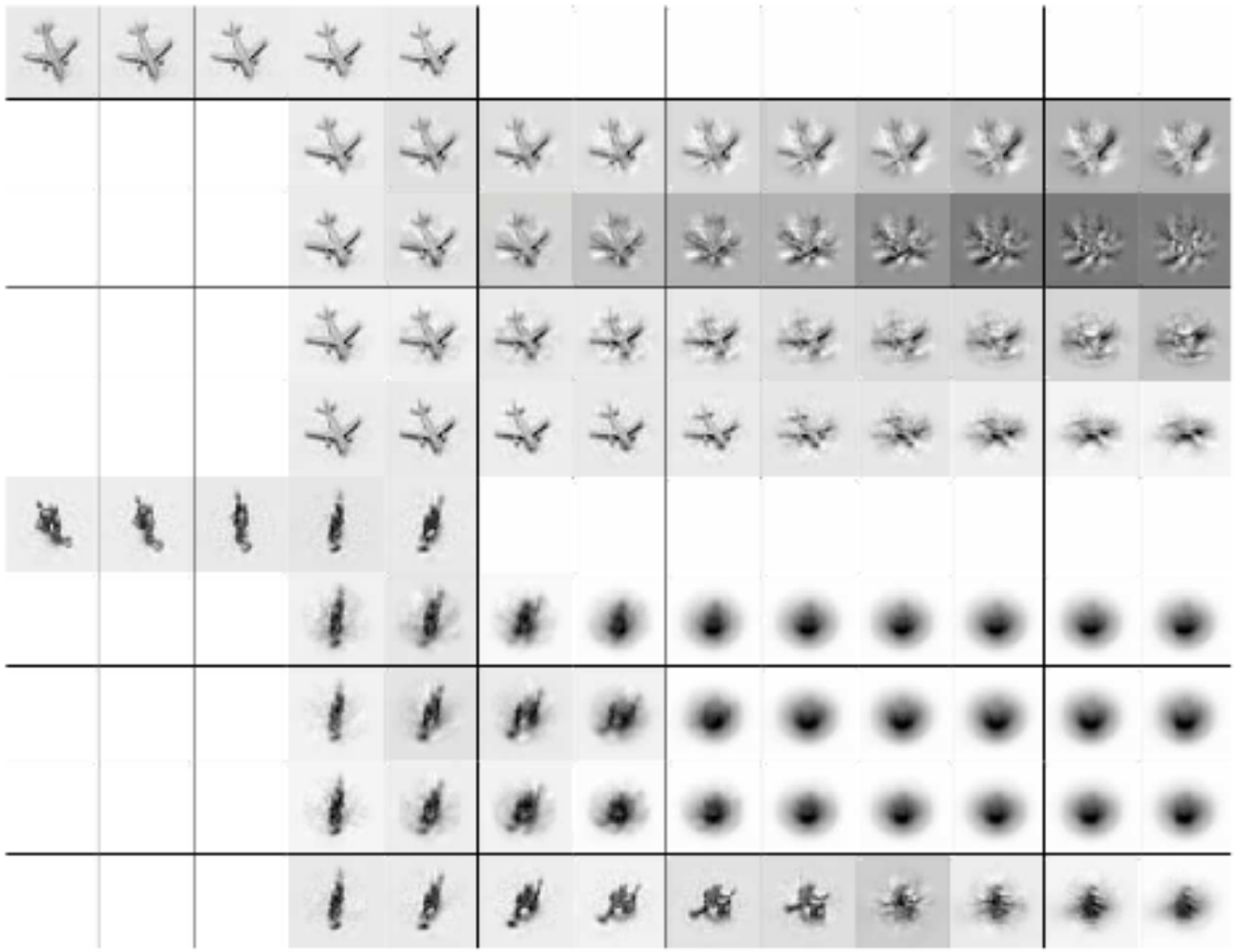
Predictive training



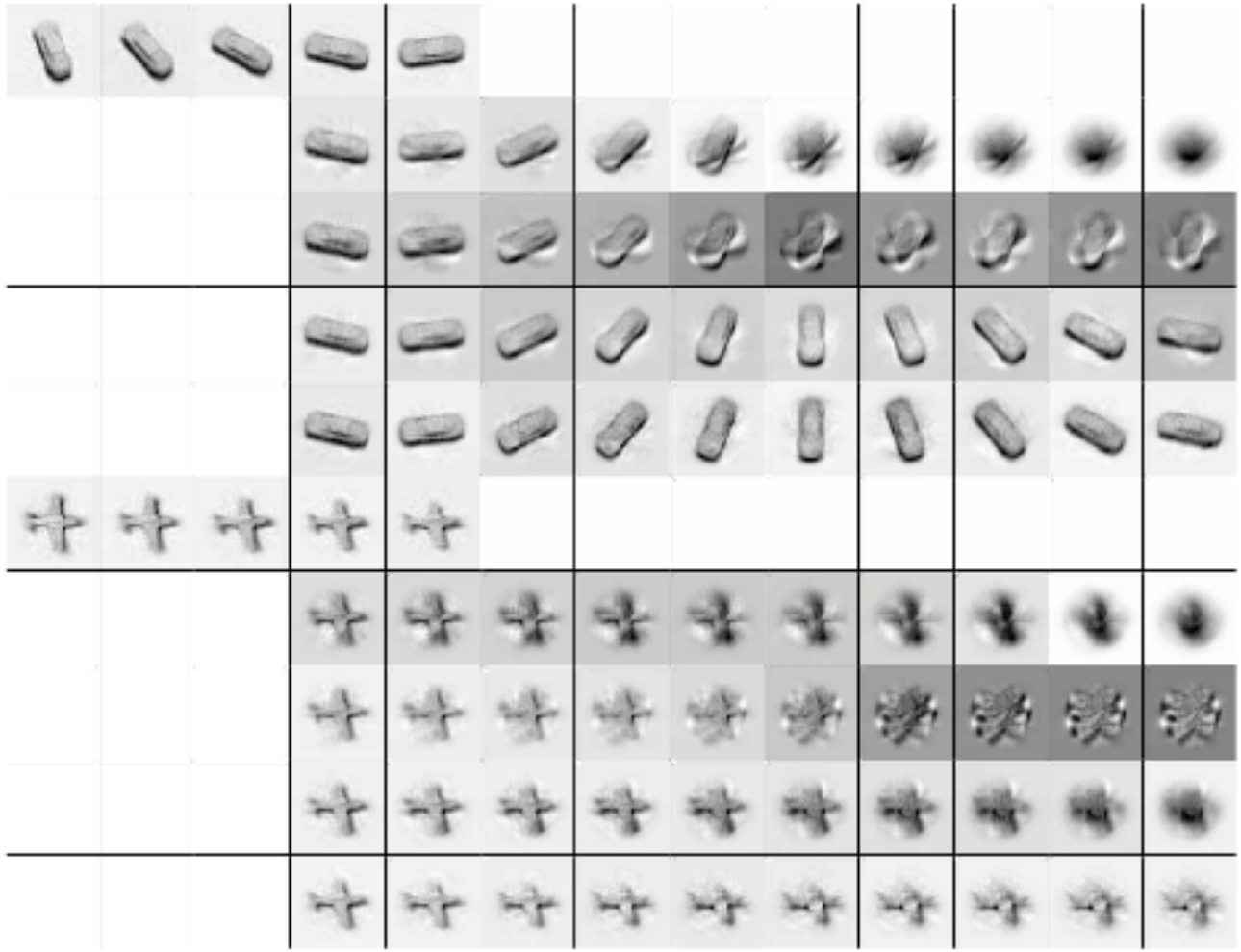
Predictive training



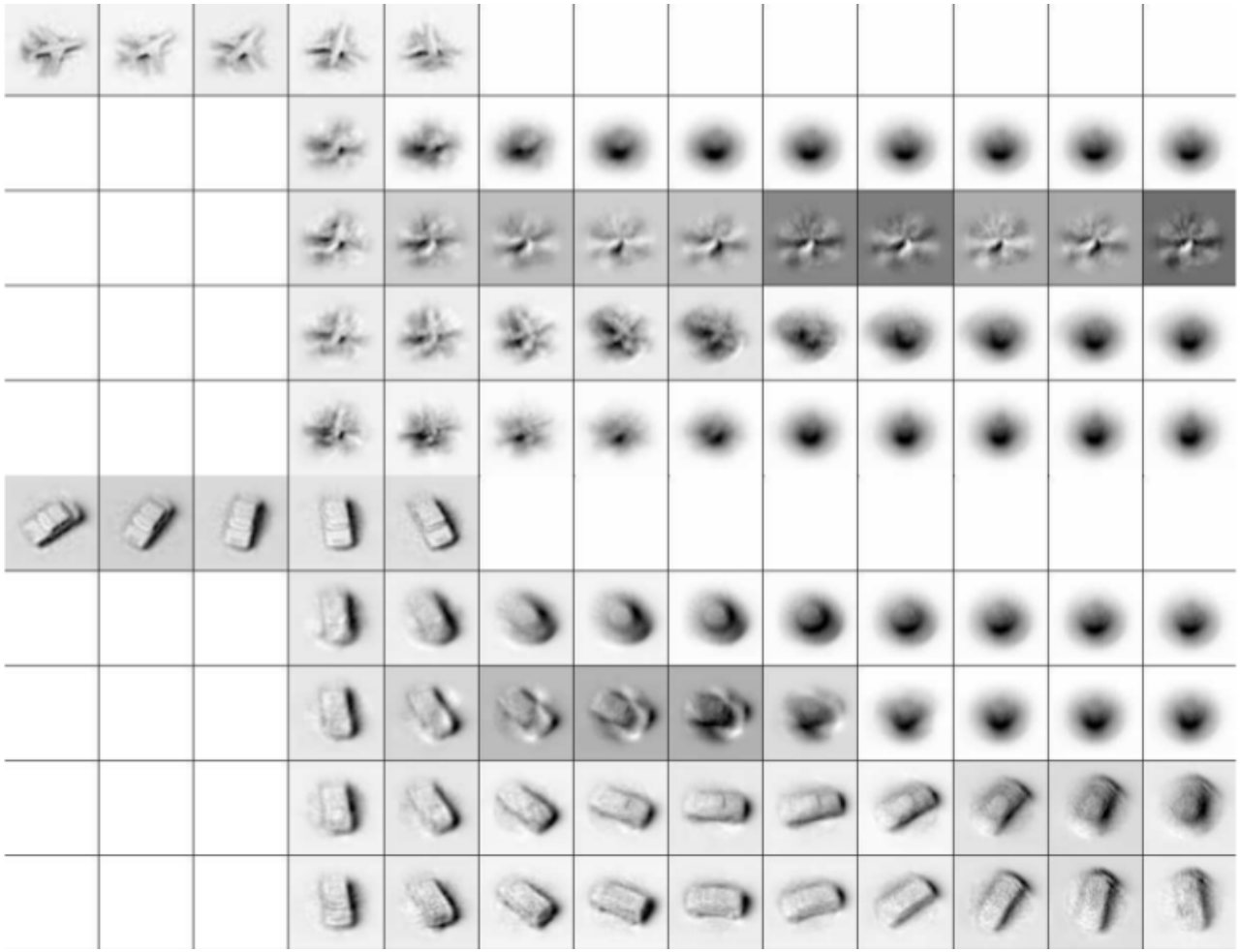
Predictive training



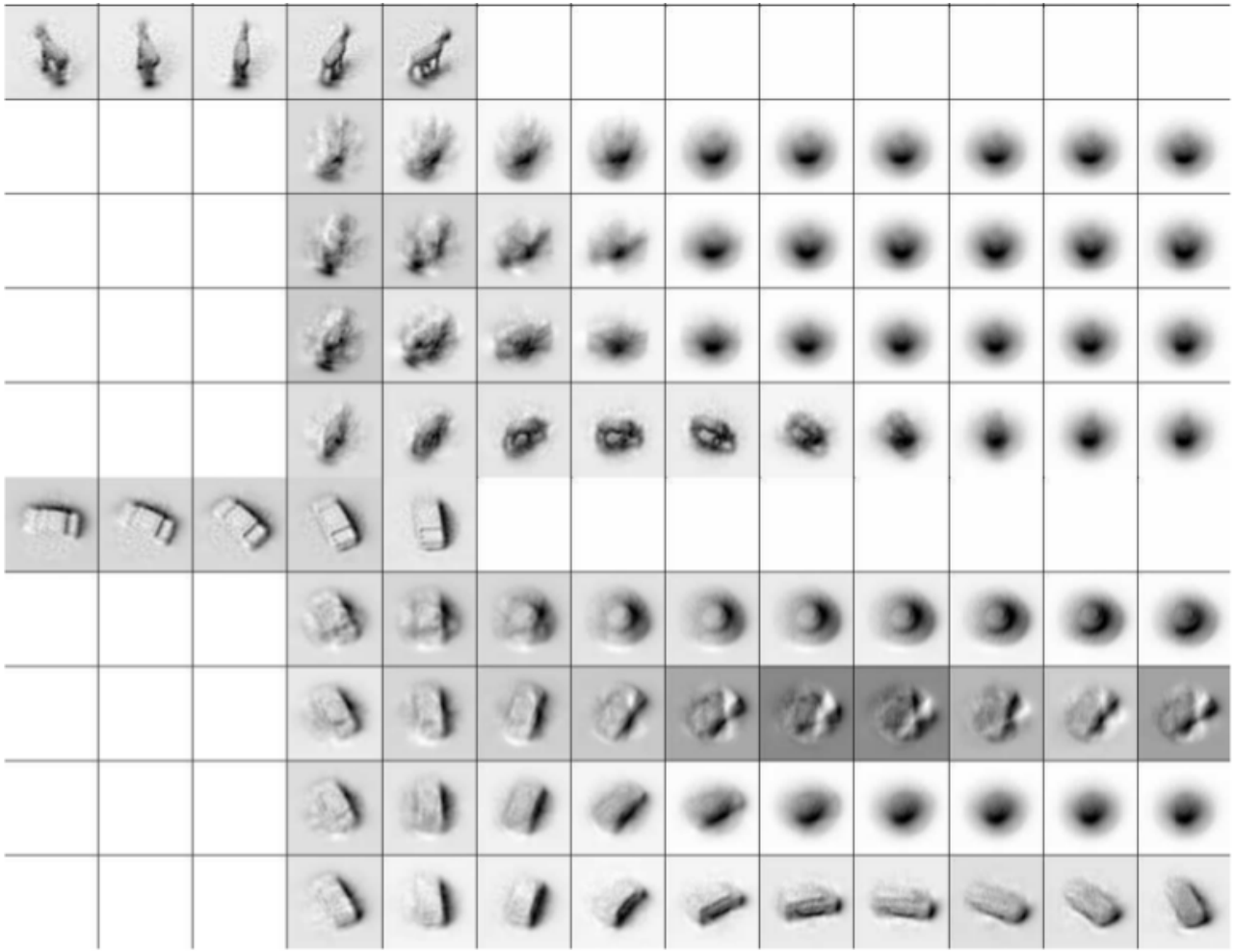
Predictive training



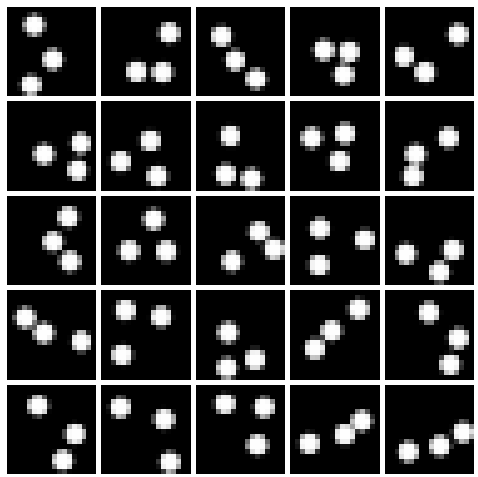
Predictive training



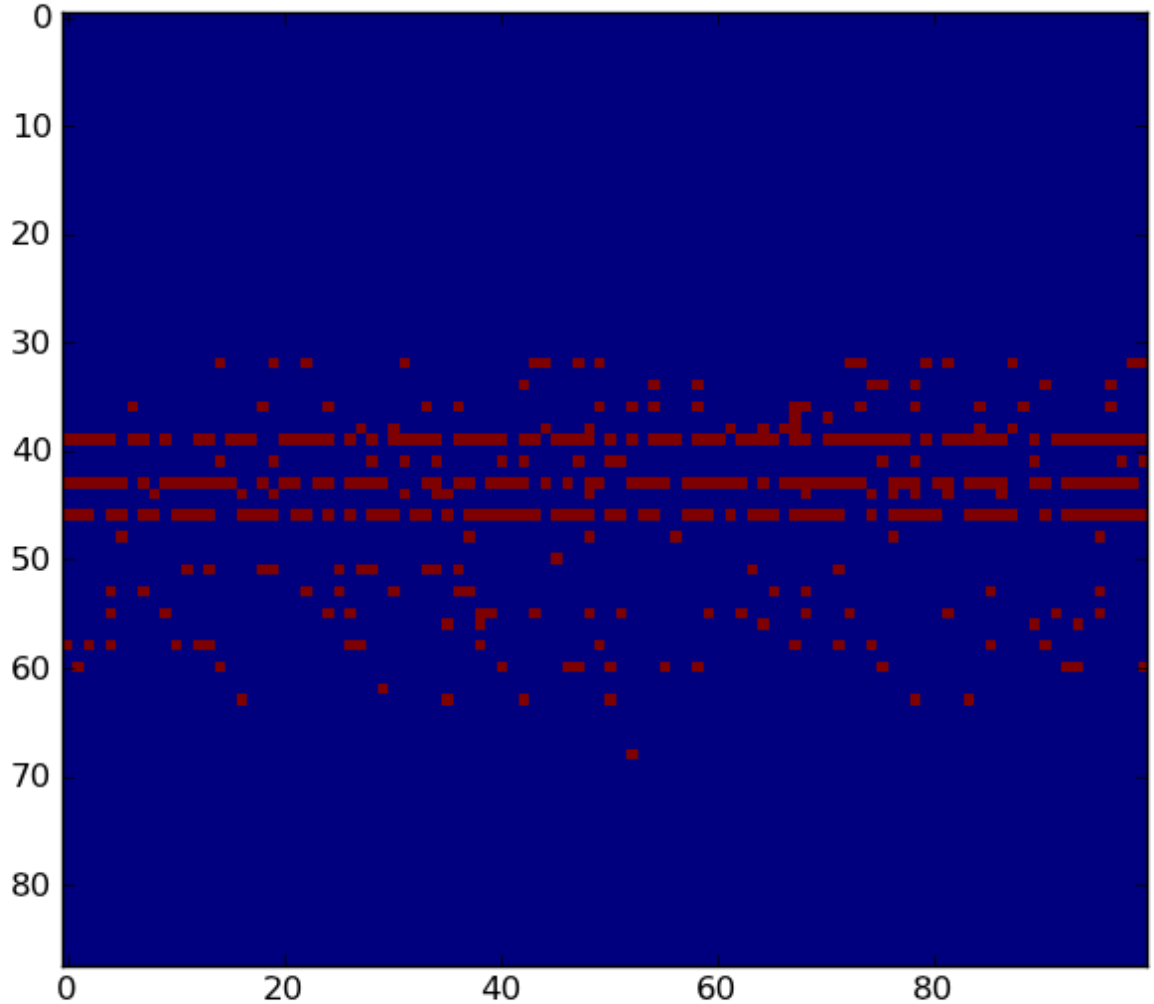
Predictive training



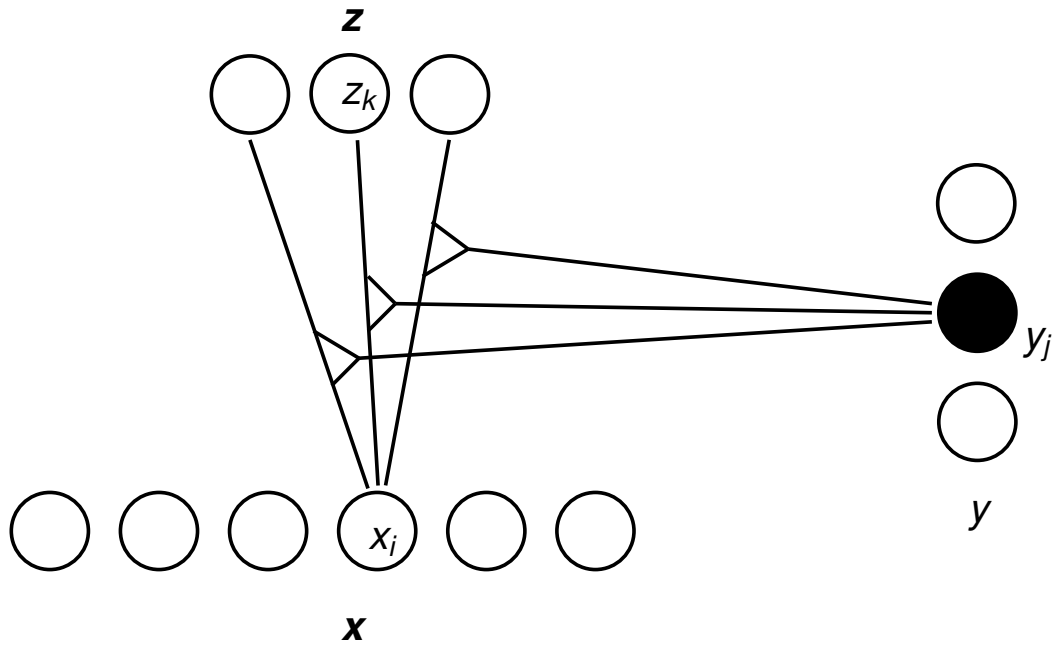
Bouncing balls



Music as 2nd order difference equation



Classification as analogy making



- Class-label maps input into an *interpretation*.
- Alternatively: Hiddens transform input into label
- Closed form for $p(y|\boldsymbol{x})$ (Memisevic, et al.; 2010), (Nair, 2008), (Larochelle et al, 2008)

Conclusions

- Vision (and cognition) make heavy use of relations.
- Making relations first-class objects may allow us to solve many tasks with a single model and with a single learning rule.
- Inductive bias for building deep learning models (eg: get more mileage out of local learning rules)

Thank you!

Questions?