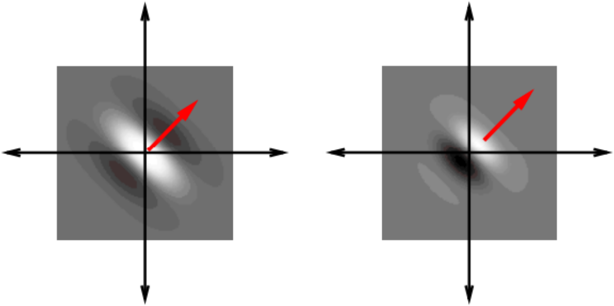
Visual features II

Roland Memisevic

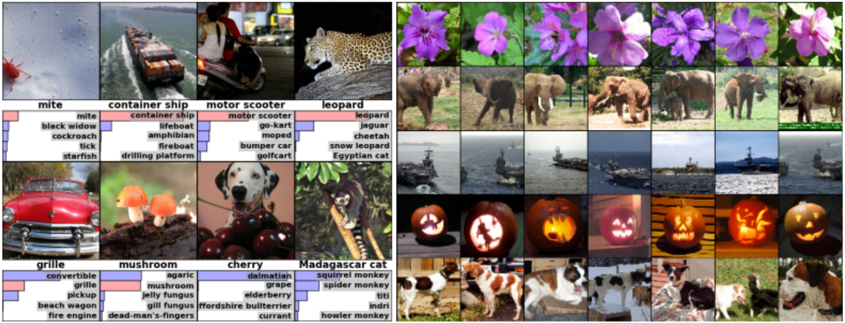
Deep Learning Summer School 2015, Montreal



$\mathbf{w}_k$ $\mathbf{w}_\ell$

figures by Javier Movellan

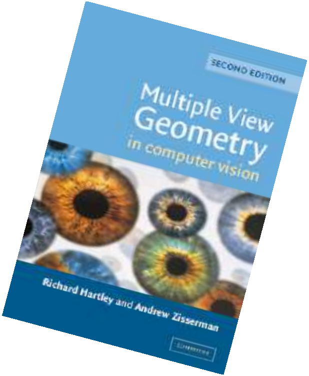
What next?



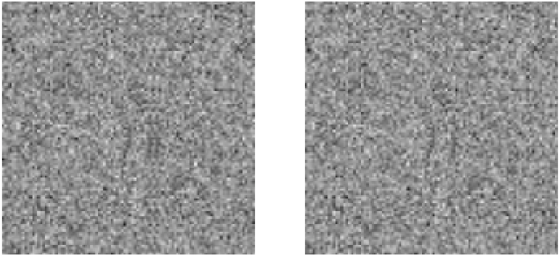
Krizhevsky et al 2012

Vision beyond object recognition

- Many vision (and other cognition) tasks depend on encoding **relations**:
- Geometry, stereo, structure-from-motion, motion understanding, activity analysis, tracking, optical flow, modeling object relations, articulation, odometry, analogy, ...



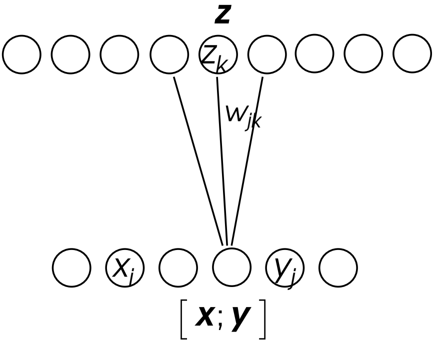
Random dot stereograms



Some things are hard to infer from still images

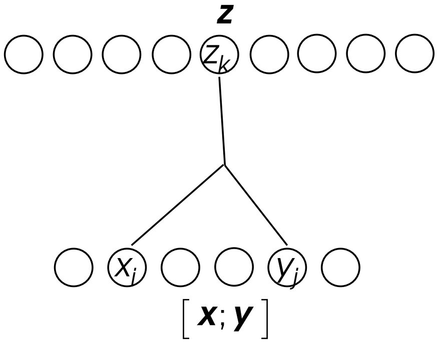
There are things images cannot teach you

Learn relations by concatenating images?



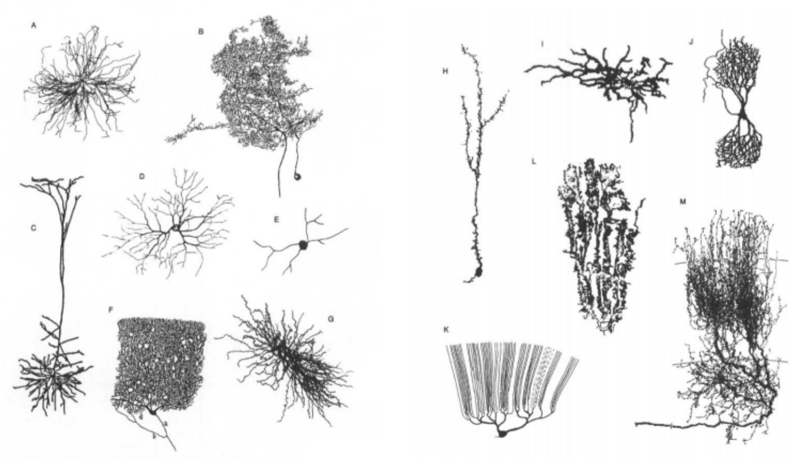
- Problem: This would make unit x_i conditionally independent of unit y_j given $\boldsymbol{z}$.

Learn relations by concatenating images?



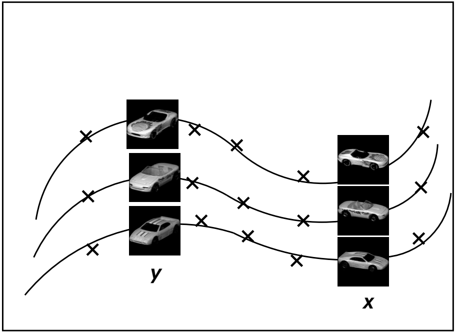
- Solution: Allow x_i and y_j to be in one clique.
- This will require "transistor neurons" that can do more than weighted summation.

$w^T x$?



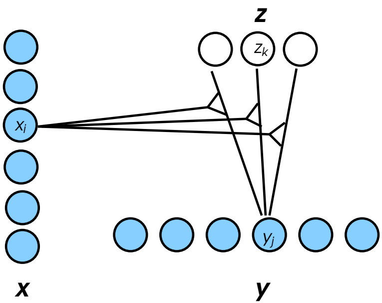
- Mel, 1994

Families of manifolds



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

Bi-linear models

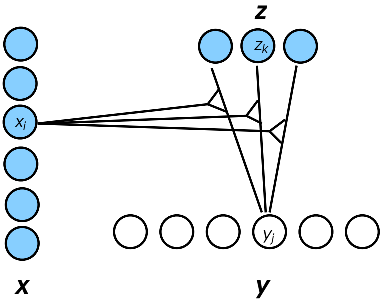


- $w_{jk}(\boldsymbol{x}) = \sum_i w_{ijk} x_i$, so

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

see, for example, (Tenenbaum, Freeman; 2000), (Grimes, Rao; 2005), (Olshausen; 2007), (Memisevic, Hinton; 2007)

Bi-linear models

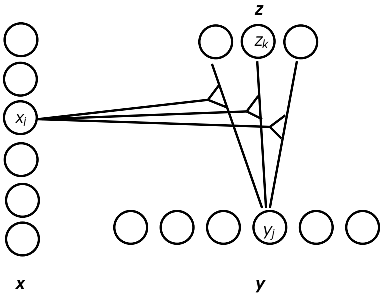


- Similar for $\boldsymbol{y}$:

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

see, for example, (Tenenbaum, Freeman; 2000), (Grimes, Rao; 2005), (Olshausen; 2007), (Memisevic, Hinton; 2007)

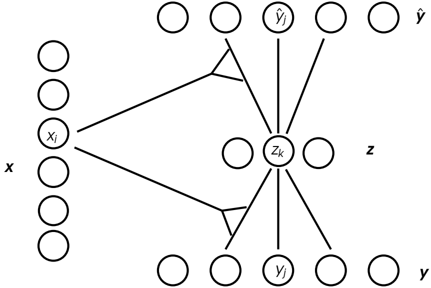
Example: Gated Boltzmann machine



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

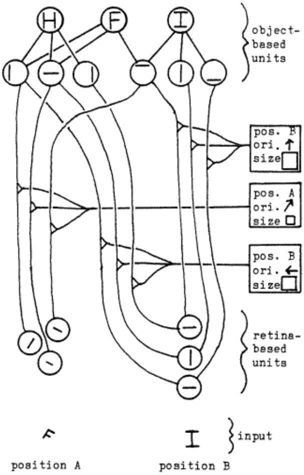
- (Memisevic, Hinton; 2007)

Example: Gated autoencoder



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

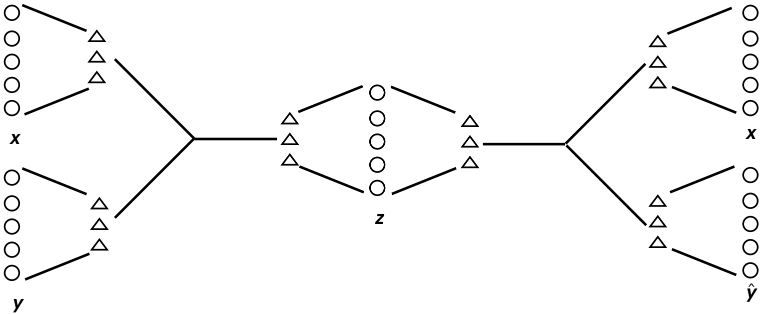
Multiplicative interactions



Hinton 1981; v.d. Malsburg 1981

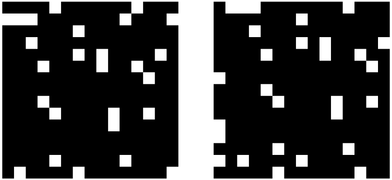
- Binocular+Motion Energy models (Adelson, Bergen; 1985), (Ozhawa, DeAngelis, Freeman; 1990), (Fleet et al., 1994)
- Higher-order neural nets, "Sigma-Pi-units"
- Tensor product binding (Smolensky, 1990), HRR (Plate, 1994)
- Routing circuits (Olshausen; 1994)
- Subspace SOM (Kohonen, 1996)
- Bi-linear models (Tenenbaum, Freeman; 2000), (Grimes, Rao; 2005), (Olshausen; 2007)
- ISA, topographic ICA (Hyvarinen, Hoyer; 2000), (Karklin, Lewicki; 2003): Higher-order within image structure
- (2006 –) GBM, mcRBM, GAE, convISA, applications...

Factored Gated Autoencoder



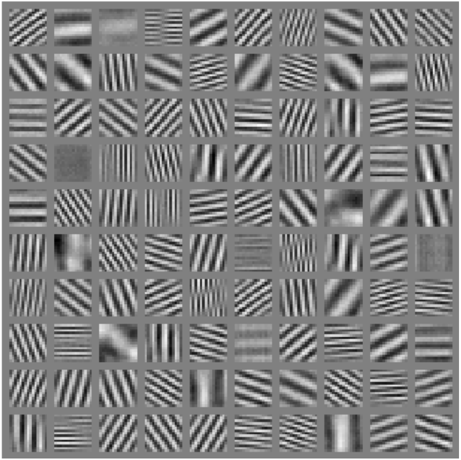
- Projecting onto filters *first* allows us to use fewer products. (Memisevic, Hinton 2010), (Taylor et al 2009)
- This is equivalent to *factorizing* the three-way parameter tensor.

Toy examples

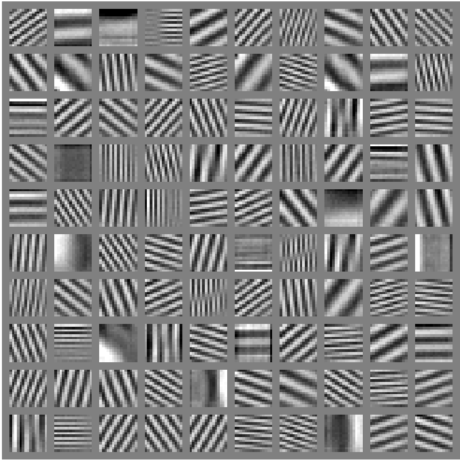


- There is no structure in these images.
- Only in *how they change*.

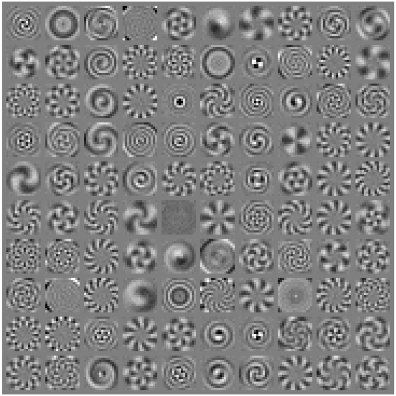
Learned filters w_{if}^{x}



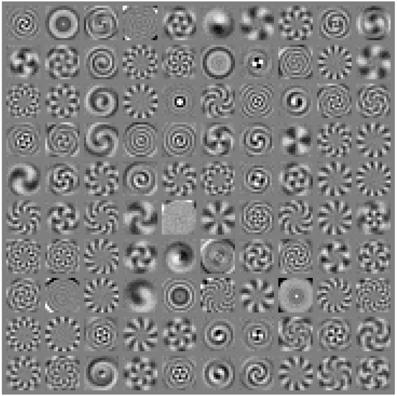
Learned filters w_{jf}^{y}



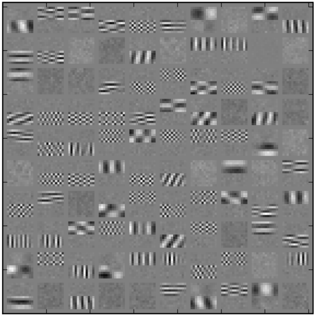
Rotation filters



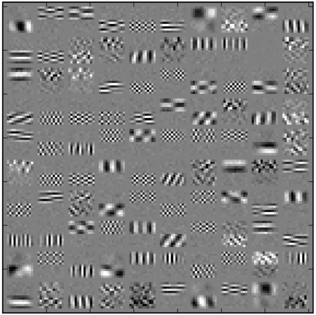
Rotation filters



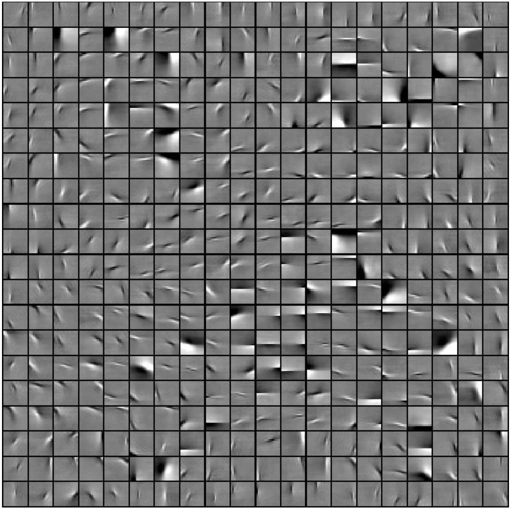
Filters learned from split-screen shifts



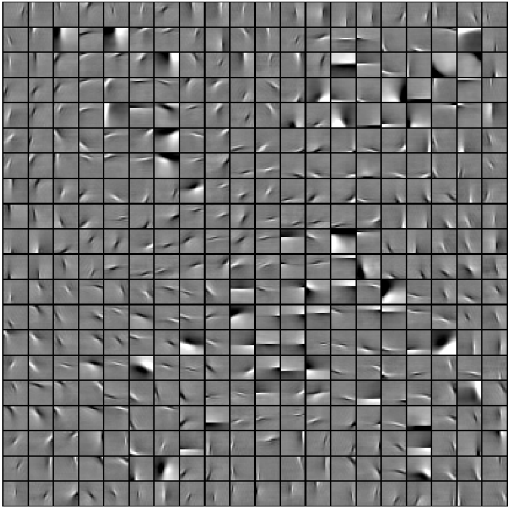
Filters learned from split-screen shifts



Natural video filters



Natural video filters



Understanding gating

- Take a linear transformation, L, in pixel space (a "warp"):

$$\boldsymbol{y} = L\boldsymbol{x}$$

and consider the task:

$$\text{Given } \boldsymbol{x} \text{ and } \boldsymbol{y}\text{, what is } L?$$

Understanding gating

(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.

Understanding gating

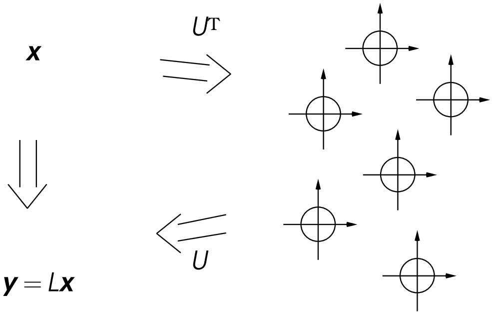
Example: Translation and the Fourier spectrum

- 1-D translation matrices are *circulants*, such as:

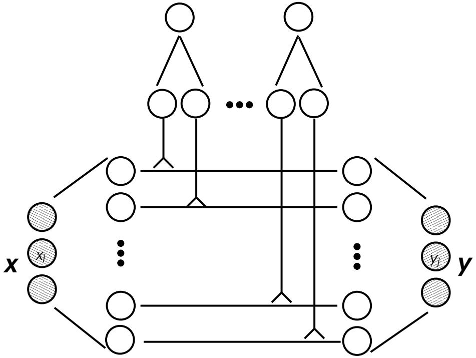
$$L = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

- Their eigenvectors are phasors.
- (Can extend this to images via block-circulants)

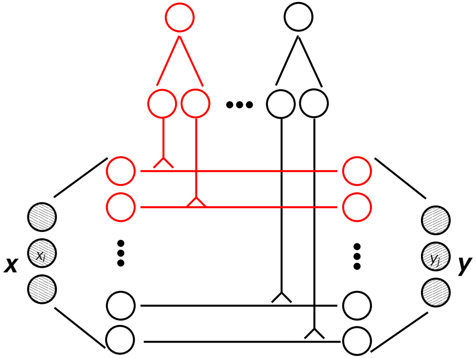
Orthogonal transformations decompose into rotations



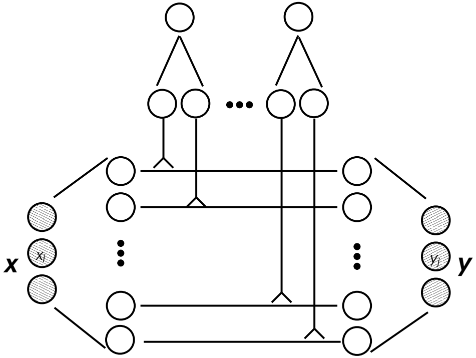
To detect the rotation angle, compute a 2-d inner product



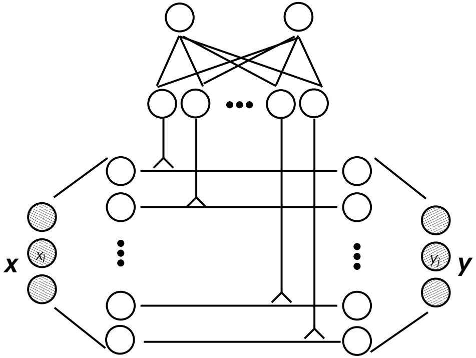
To detect the rotation angle, compute a 2-d inner product



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



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



The aperture problem

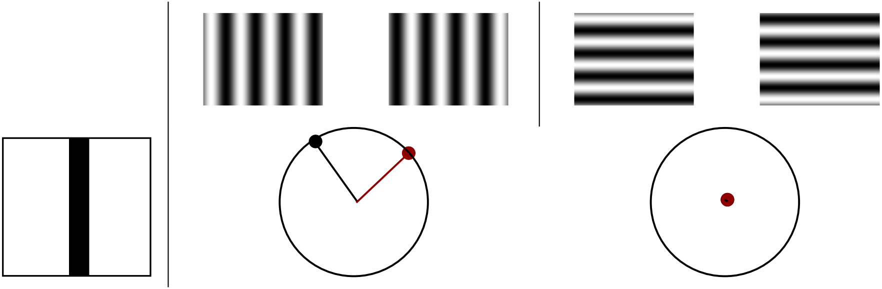
The aperture problem

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

The aperture problem

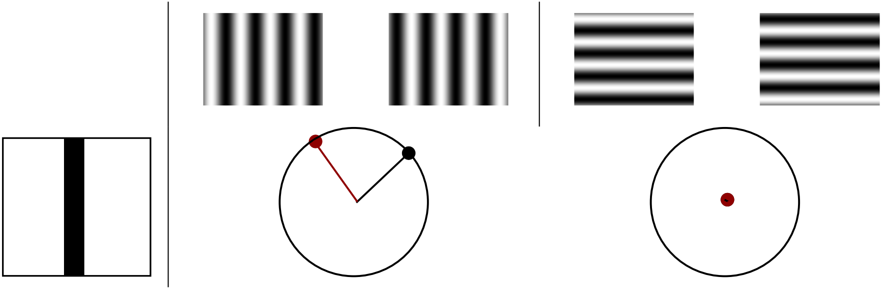
The aperture problem

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



The aperture problem

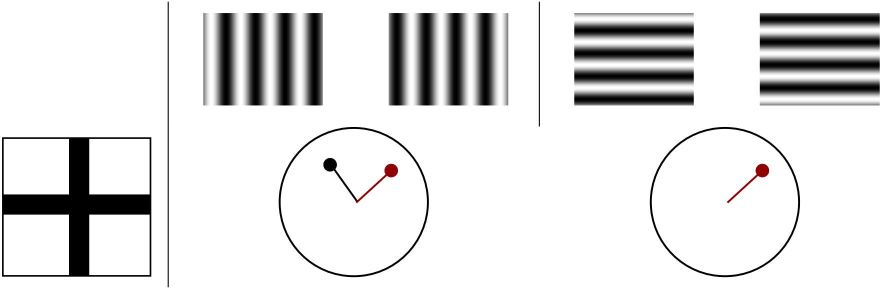
- **Not all images are represented equally well in each subspace.**



The aperture problem

The aperture problem

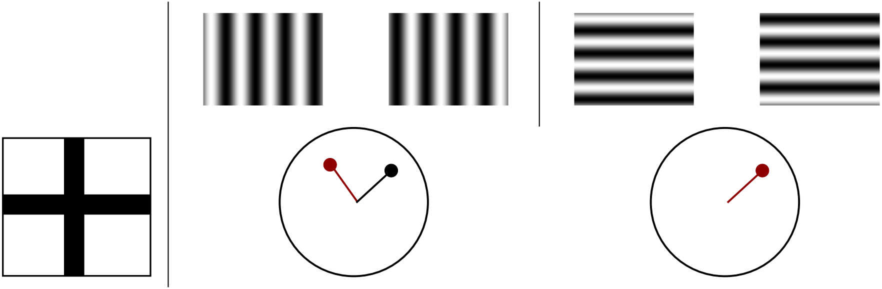
- **Not all images are represented equally well in each subspace.**



The aperture problem

The aperture problem

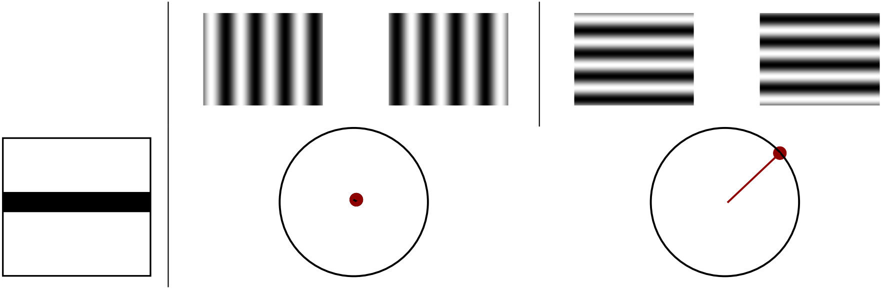
- **Not all images are represented equally well in each subspace.**



The aperture problem

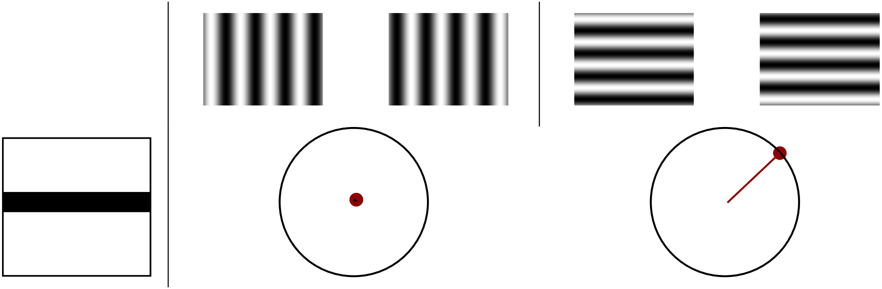
The aperture problem

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

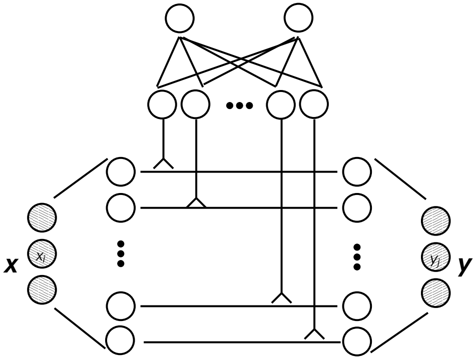


The aperture problem

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

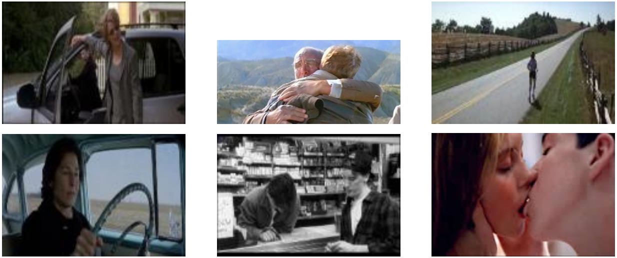


To detect the rotation angle, *pool* over 2-d inner products



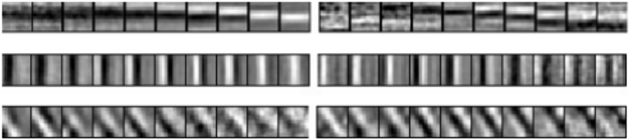
- This is the same as a factored bi-linear model.
- It is also the same as a "square-pooling" model (complex cell) if we let $\boldsymbol{x} = \boldsymbol{y}$.

Action recognition 2011



(Hollywood 2)

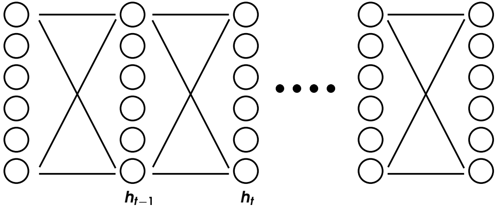
- Convolutional GBM (Taylor et al., 2010)
- hierarchical ISA (Le, et al., 2011)



Other applications

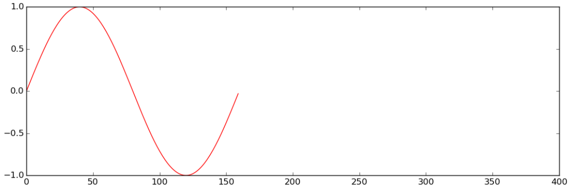
- Invariance from videos (Cadieu, Olshausen 2011); (Zou et al 2012); (Memisevic, Exarchakis 2013)
- Depth inference, eg. (Fleet et al 1994), (Konda, Memisevic 2014)
- Analogy making (Memisevic, Hinton 2010)
- Odometry (Konda, Memisevic 2015)
- ...

Vanishing gradients



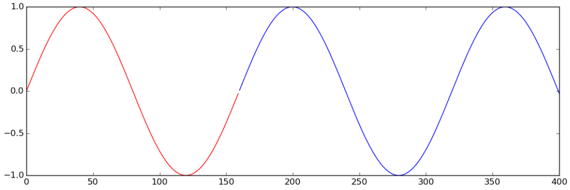
- Back-prop through many layers is hard, because computing the product of many matrices is unstable.
- Orthogonal layers may help, because their eigenvalues have absolute value 1.0 (eg. Saxe et al. 2014)
- Identity initialization (Le et al 2015) works, too (but is a strange choice)

Orthogonal weights create "dynamic memory"



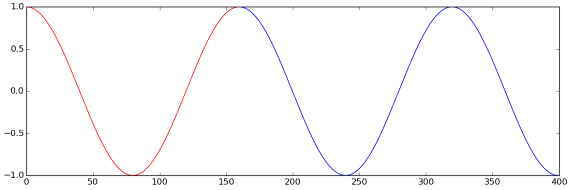
- An infinite sine-wave can be generated by applying the same orthogonal transformation over and over again.
- This will work independently of the initial phase of the sine-wave, if your basis is "steerable" (Bethge et al. 2007).

Orthogonal weights create “dynamic memory”



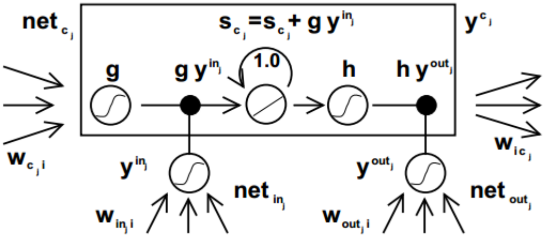
- An infinite sine-wave can be generated by applying the same orthogonal transformation over and over again.
- This will work independently of the initial phase of the sine-wave, if your basis is “steerable” (Bethge et al. 2007).

Orthogonal weights create “dynamic memory”



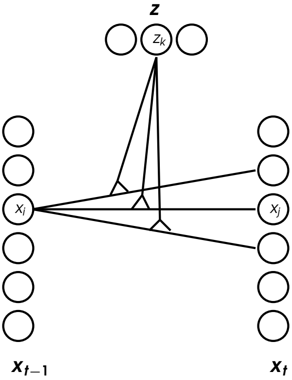
- An infinite sine-wave can be generated by applying the same orthogonal transformation over and over again.
- This will work independently of the initial phase of the sine-wave, if your basis is “steerable” (Bethge et al. 2007).

Why memory needs gating



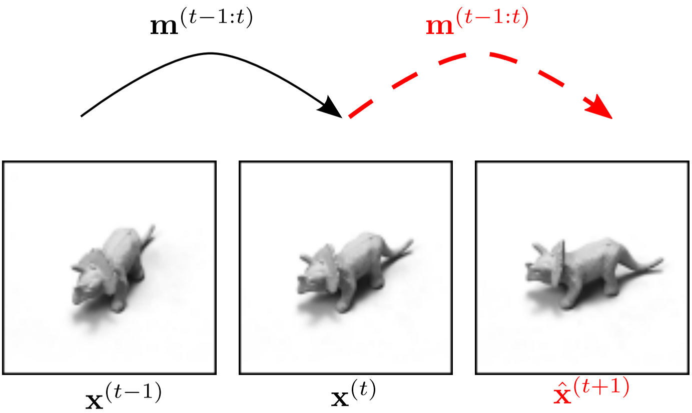
picture from (Hochreiter, Schmidthuber; 1997)

Gating units



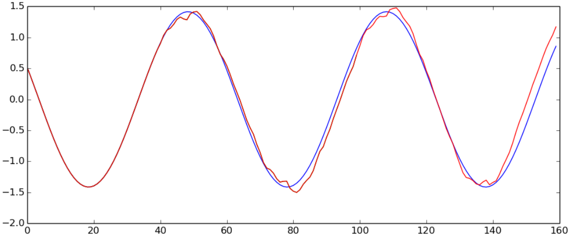
- *Mixtures* of orthogonal transformations can generate arbitrary frequencies (if we know the right mixture coefficients).
- This will still work independently of the initial phase of each respective sine-wave.

Predictive training

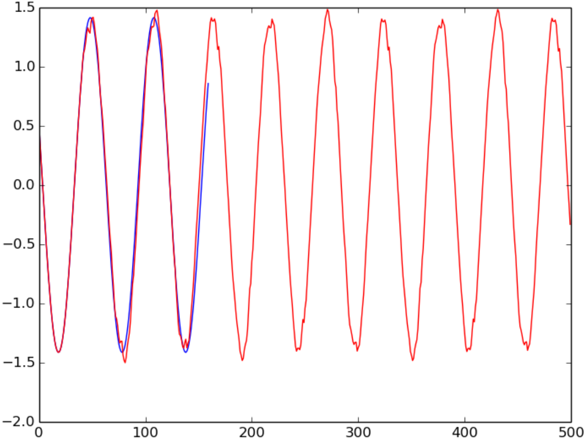


- (Michalski et al., 2014)
- One way to turn a bi-linear model into a recurrent net is by training to **predict** future frames, assuming the transformation to be constant.

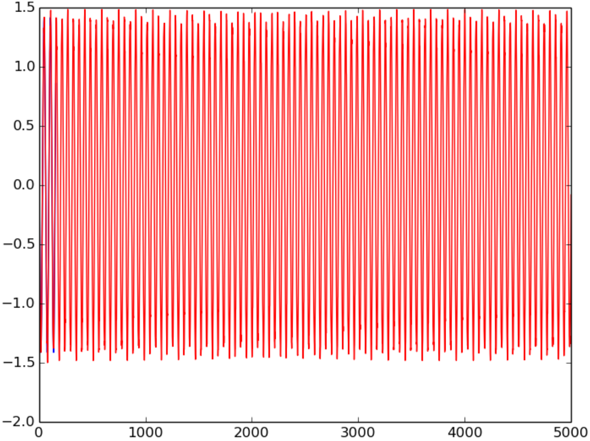
sine waves



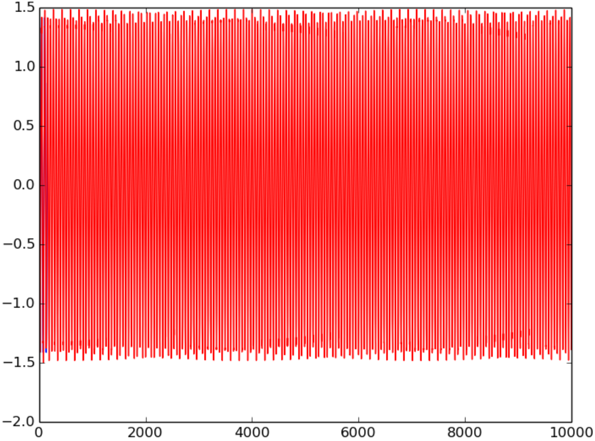
sine waves



sine waves

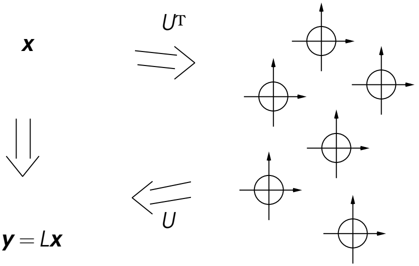


sine waves

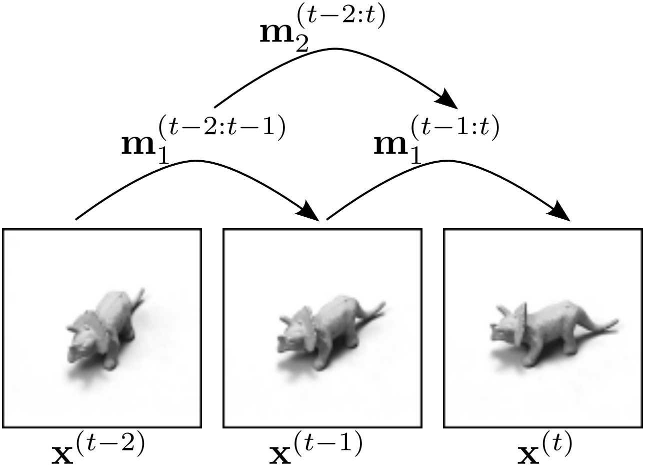


The model learns rotational derivatives

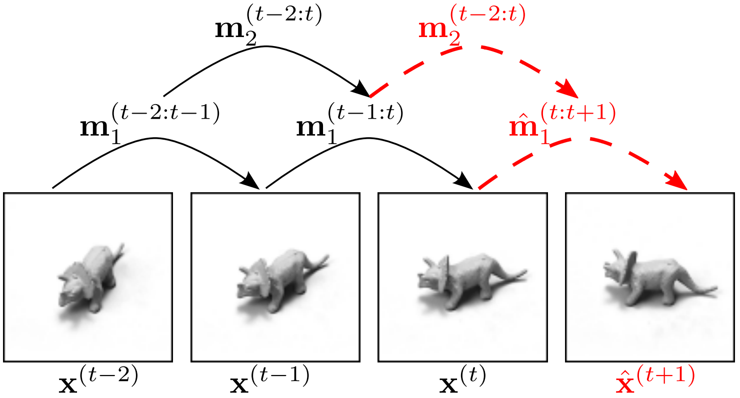
$$U^{\mathrm{T}} L U = \begin{bmatrix} R_1 & & \\ & \ddots & \\ & & R_k \end{bmatrix} \qquad R_i = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i) \end{bmatrix}$$



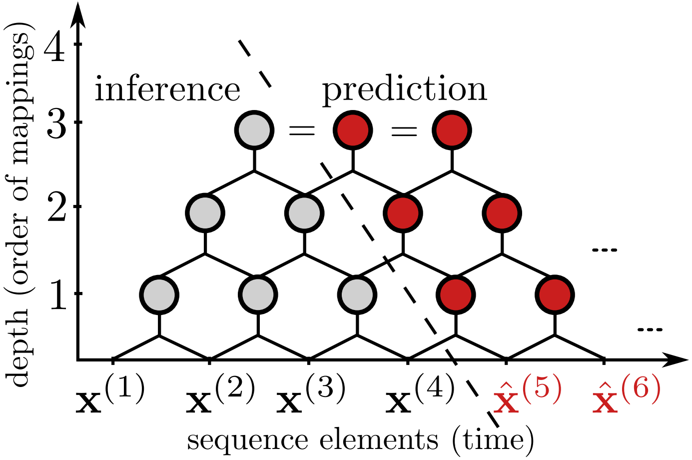
Learning higher-order derivatives (acceleration)



Learning higher-order derivatives (acceleration)

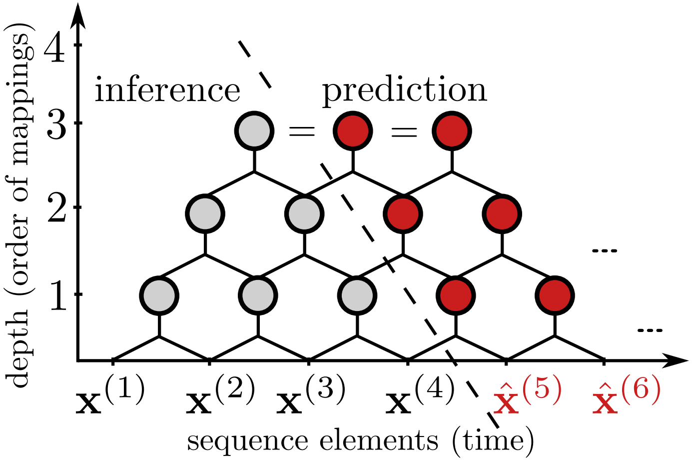


snap, crackle, pop



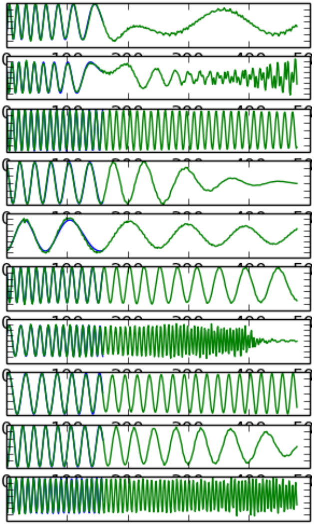
- The model is orthogonal in time, contractive in layers
- Sigmoids represent invariance, linear features equivariance
- 3-way connections similar to tensor nets (Socher et al 2013)

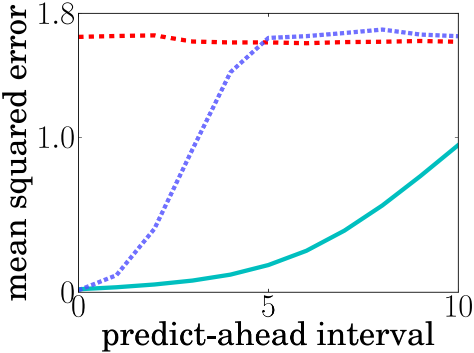
Annealed teacher forcing



- Should a unit get bottom-up or top-down information?
- Given it both, but reduce the bottom-up information over time. Eg. by adding more and more corruption.

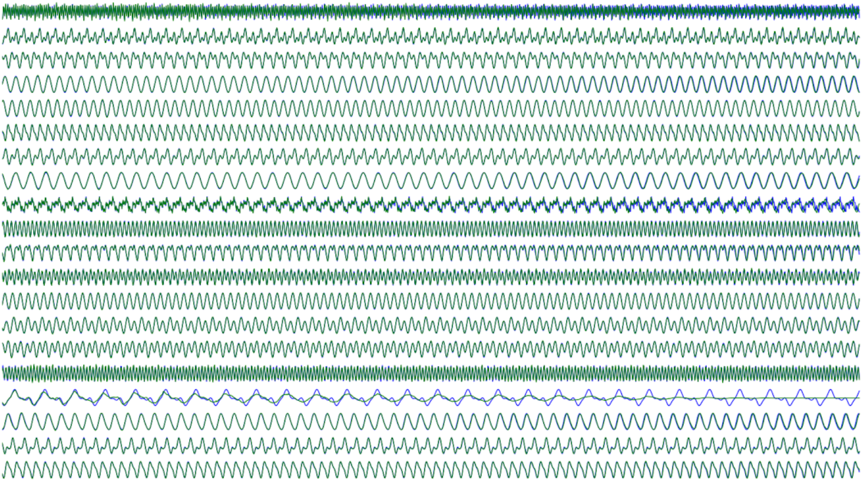
chirps



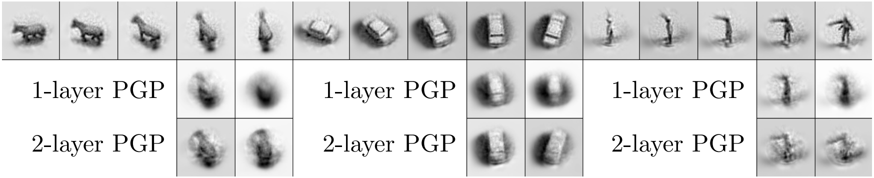


(CRBM vs RNN vs grammar cells)

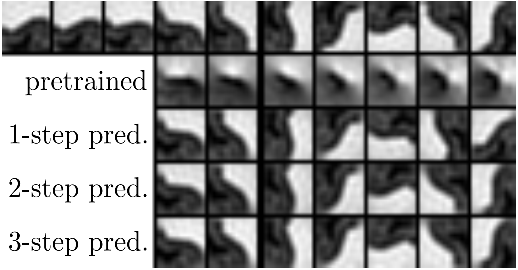
Harmonics

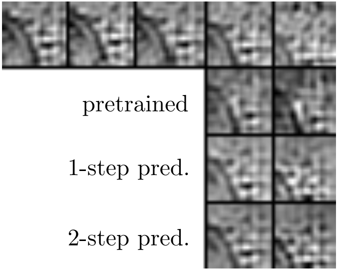


NORB videos


1-layer PGP
2-layer PGP
1-layer PGP
2-layer PGP
1-layer PGP
2-layer PGP

Multi-step prediction helps

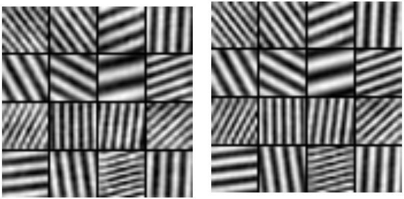




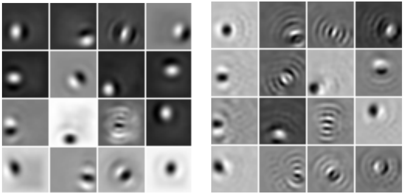
Recognizing accelerations

Data set	$m[1]1:2$	$m[1]2:3$	$(m[1]1:2, m[1]2:3)$	$m[2]1:3$
AccRot	18.1 (19.4)	29.3 (30.9)	74.0 (64.9)	74.4 (53.7)
AccShift	20.9 (20.6)	34.4 (33.3)	42.7 (38.4)	80.6 (63.4)

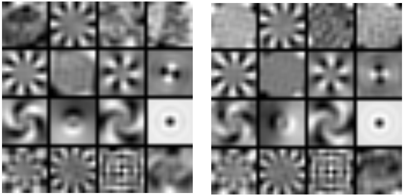
Learned filters



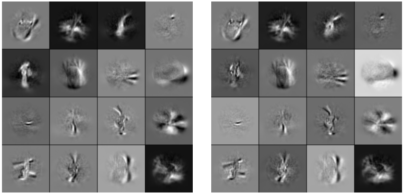
accelerated shifts



bouncing balls

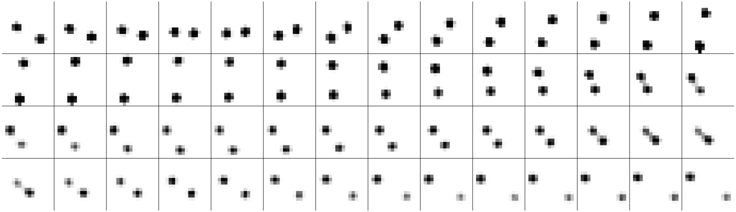


accelerated rotations

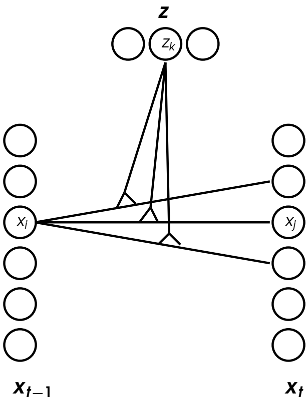


NORBVideos

bouncing balls (Mnih et al), (Sutskever et al)

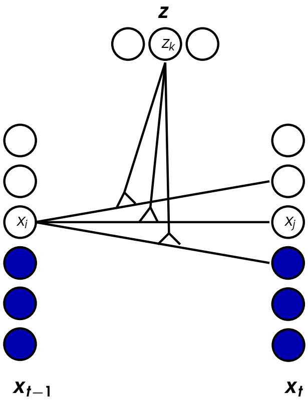


Adding hidden "notebook" units



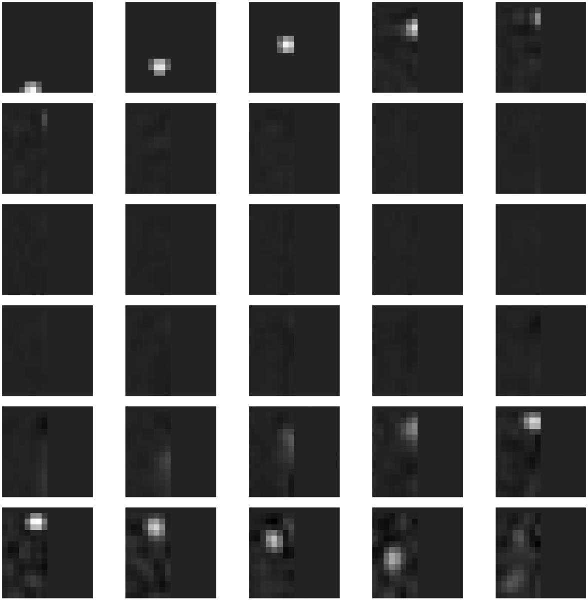
- If we add hidden units to $\mathbf{x}$, each transformation will be able to
  1. write information into the hiddens
  2. read out from the hiddens
  3. transform the hiddens
  4. transform the observables

Adding hidden "notebook" units

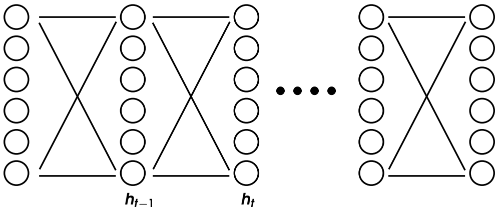


- If we add hidden units to $\mathbf{x}$, each transformation will be able to
  1. write information into the hiddens
  2. read out from the hiddens
  3. transform the hiddens
  4. transform the observables

bouncing ball with occlusion

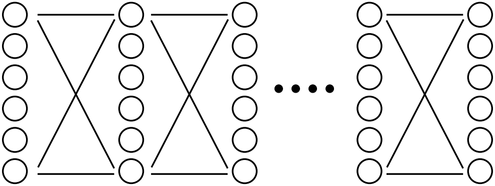


Vanishing gradients



- Back-prop through many layers is hard, because computing the product of many matrices is unstable.
- Orthogonal layers may help, because their eigenvalues have absolute value 1.0 (eg. Saxe et al. 2014)
- Identity initialization (Le et al 2015) works, too (but is a strange choice)

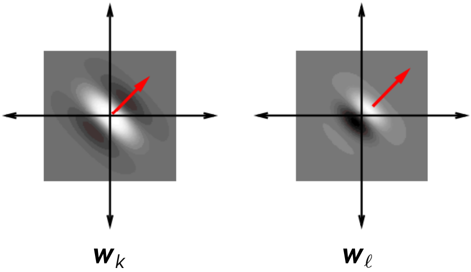
Vanishing gradients



- Back-prop through many layers is hard, because computing the product of many matrices is unstable.
- Orthogonal layers may help, because their eigenvalues have absolute value 1.0 (eg. Saxe et al. 2014)
- Identity initialization (Le et al 2015) works, too (but is a strange choice)

What you should really want are
orthogonal active paths
through the network.

A 2-d subspace

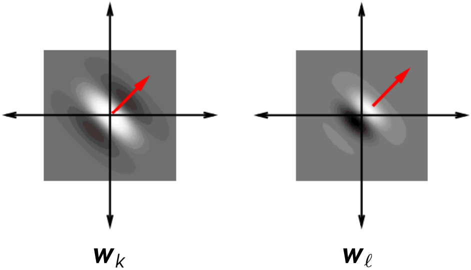


$$\boldsymbol{x} \approx a_k \boldsymbol{w}_k + a_l \boldsymbol{w}_\ell$$

$$a_k = ?, \quad a_\ell = ?$$

figures by Javier Movellan

A 2-d subspace



$$\boldsymbol{x} \approx a_k \boldsymbol{w}_k + a_l \boldsymbol{w}_\ell$$

$$a_k = \boldsymbol{w}_k^{\mathrm{T}} \boldsymbol{x}, \quad a_\ell = \boldsymbol{w}_\ell^{\mathrm{T}} \boldsymbol{x}$$

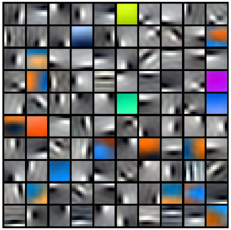
figures by Javier Movellan

Do autoencoders orthogonalize weights?

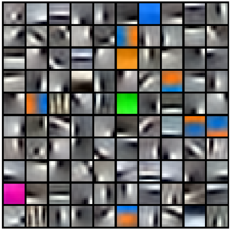
- Autoencoders minimize
$$\big(\boldsymbol{r}(\boldsymbol{x}) - \boldsymbol{x}\big)^2$$
using the reconstruction
$$\boldsymbol{r}(\boldsymbol{x}) = W\boldsymbol{h}(\boldsymbol{x}) = \sum_{k:h_k \neq 0} h_k \boldsymbol{w}_k$$
where h_k is the output of hidden unit k
- For orthonormal active weights the optimal coefficients would be:
$$h_k = \boldsymbol{w}_k^{\mathrm{T}} \boldsymbol{x}$$
- In reality, a ReLU autoencoder uses
$$h_k = \boldsymbol{w}_k^{\mathrm{T}} \boldsymbol{x} + b_k$$

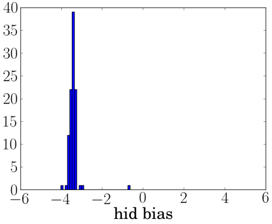
Autoencoders learn negative biases

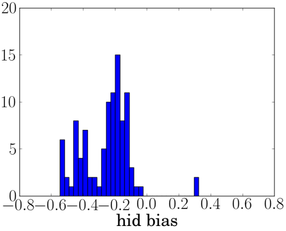
contractive AE (sigmoid)



denoising AE (ReLU)

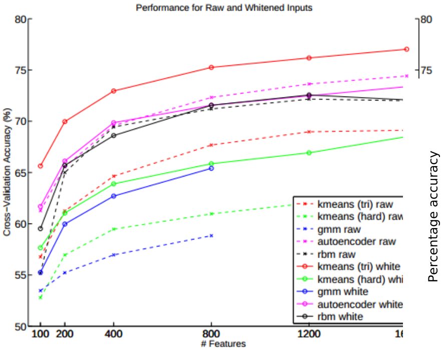




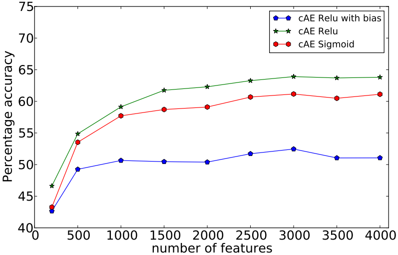


- see also M. Ranzato, et al. 2007, K. Kavukcuoglu, et al., 2008.

Zero-bias ReLUs are hard to beat



CIFAR-10 performance (Coates et al., 2011)



CIFAR-10 permutation-invariant (Konda et al., 2015)

The energy function of a ReLU autoencoder

- ReLU autoencoder:

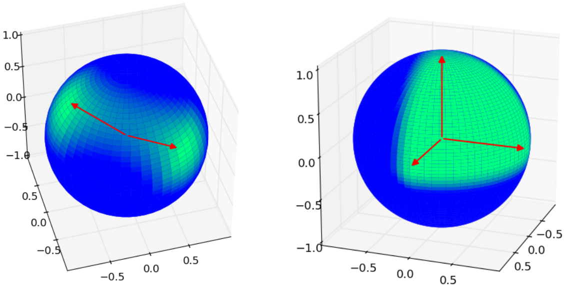
$$\mathcal{F}(\boldsymbol{x}) = \frac{1}{2}(\boldsymbol{x}+\boldsymbol{a}_{\boldsymbol{x}})^{\mathrm{T}} W_{\boldsymbol{x}}^{\mathrm{T}} W_{\boldsymbol{x}} (\boldsymbol{x}+\boldsymbol{a}_{\boldsymbol{x}}) - \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{c}\|^2$$

  with $\boldsymbol{a}_{\boldsymbol{x}} = \frac{1}{2}(W_{\boldsymbol{x}}^{\mathrm{T}} W_{\boldsymbol{x}})^{-1} W_{\boldsymbol{x}}^{\mathrm{T}} b_{\boldsymbol{x}}$ and $W_{\boldsymbol{x}}$ contains the active weight vectors for $\boldsymbol{x}$.

- Zero-bias ReLU autoencoder:

$$\mathcal{F}(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^{\mathrm{T}} W_{\boldsymbol{x}}^{\mathrm{T}} W_{\boldsymbol{x}} \boldsymbol{x} - \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{c}\|^2$$

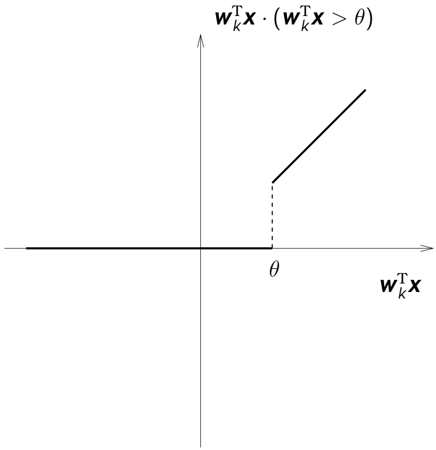
The energy function of a ReLU autoencoder



- Orthogonal transformations are **“steerable”** (Bethge et al. 2007)

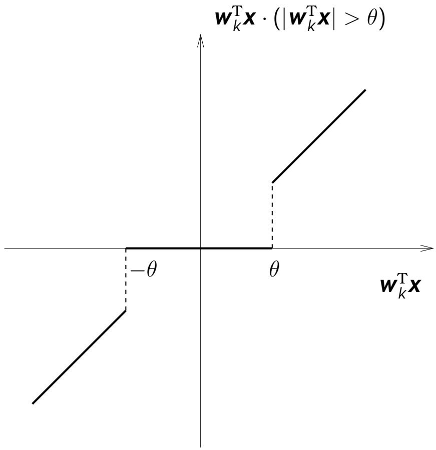
figure by David Krueger

Truncated rectified unit (Trec)



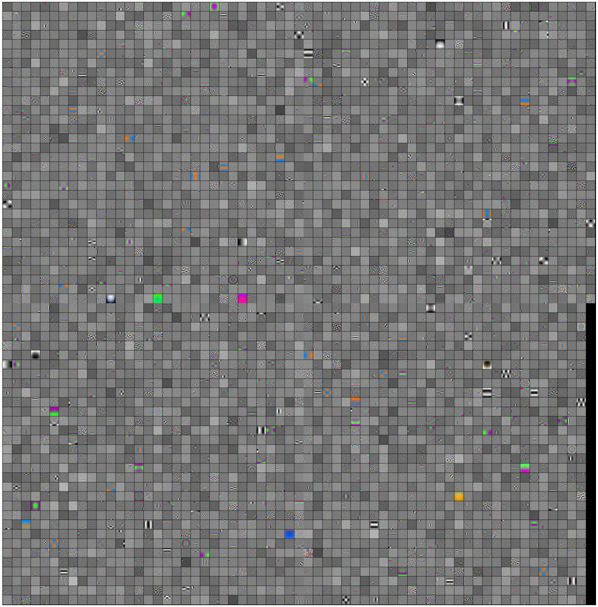
- Like spike-and-slab, hard-threshold, “coring”

Truncated linear unit (TLin)

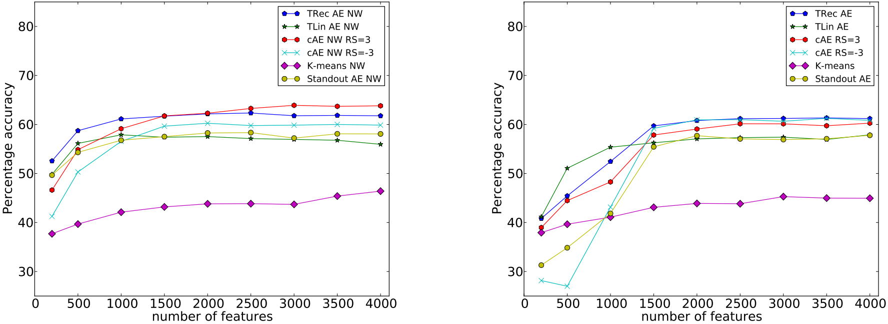


- Like spike-and-slab, hard-threshold, "coring"

ZAE features from tiny images (Torralba et al.)

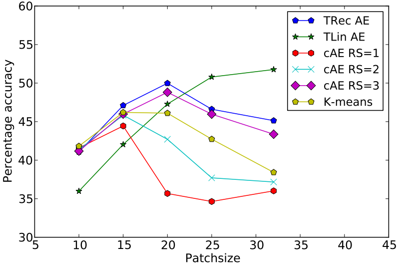


Perm-invariant CIFAR-10

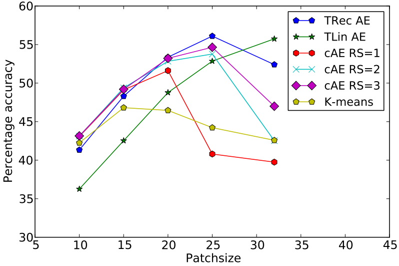


- Zero-bias ReLU at test time.
- With fine-tuning and dropout: 64.1%

Perm-invariant CIFAR-10 patches

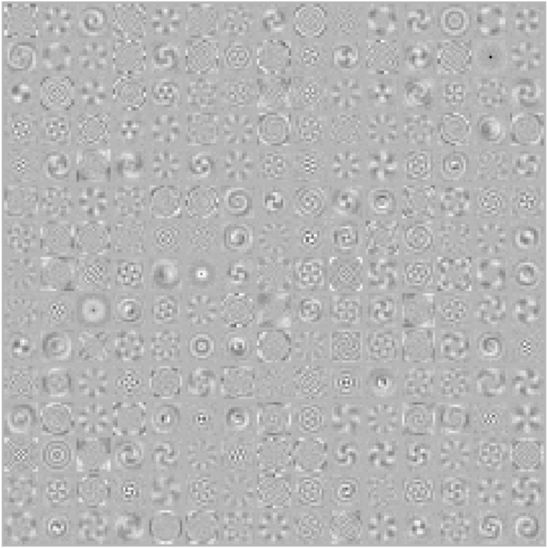


500 hiddens

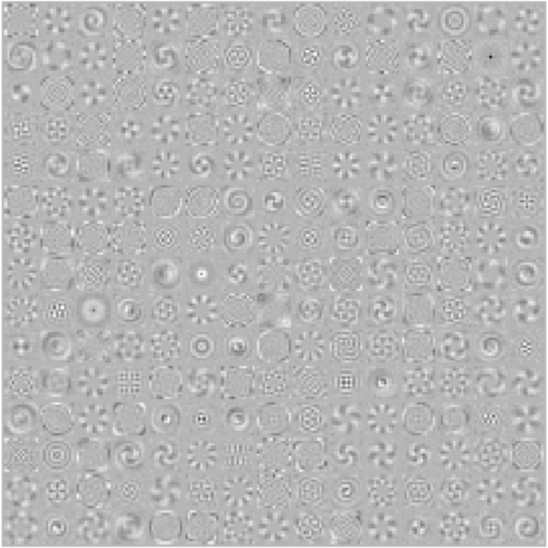


1000 hiddens

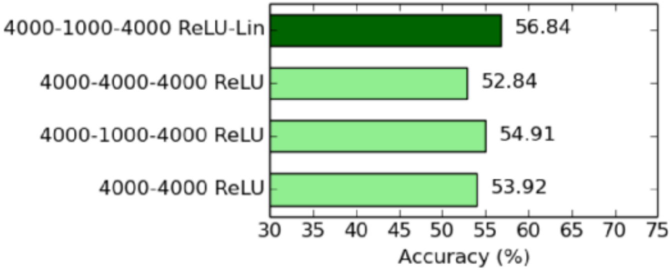
Rotation filters



Rotation filters

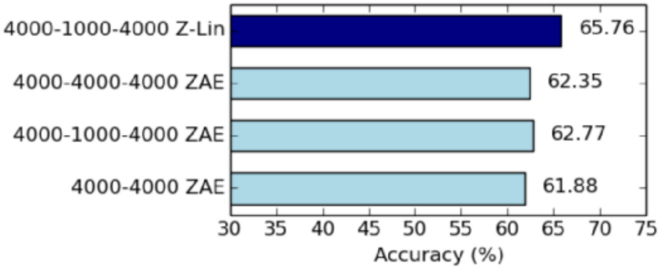


Deep fully-connected CIFAR-10



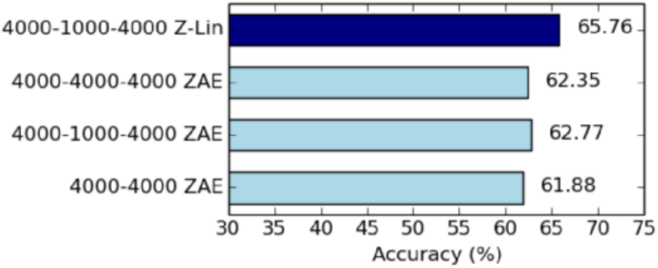
(Zhouhan Lin)

Deep fully-connected CIFAR-10



(Zhouhan Lin)

Deep fully-connected CIFAR-10

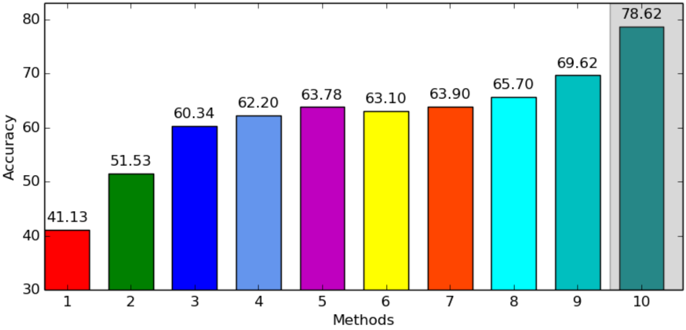


(Zhouhan Lin)

8 layers and dropout: **69.62%**

Training with deformations (not perm-invariant): **78.62%**

Deep fully-connected CIFAR-10



1. Logistic Regression on whitened data (Krishevsky);
2. Pure backprop on a 782-10000-10 network (Krishevsky);
3. Pure backprop on a 782-10000-10000-10 network (Krishevsky);
4. RBM with 2 hidden layers of 10000 hidden units each, plus alogistic regression (Krishevsky);
5. RBM with 10000 hiddens plus logistic regression (Krishevsky);
6. Fastfood FFT model (13);
7. Zerobias autoencoder of 4000 hidden units with logistic regression (10);
8. 782-4000-1000-4000-10 Z-Lin network trained without dropout;
9. 782-4000-1000-4000-1000-4000-1000-4000-10 Z-Lin network, trained with dropout
10. Z-Lin network the same as (8) but trained with dropout and data augmentation

Thank you

Questions?

Analogy making

- (Susskind, et al., 2011)